Data Scientist

STUDENT GUIDE

PEGA Academy

This document is the property of:
Pegasystems Inc.
1 Main St, Cambridge, MA 02142-1531
Phone: (617) 374-9600
Fax: (617) 374-9620
www.pega.com

Mission: Data Scientist

Products: Pega Platform™ ’26

URL: https://academy.pega.com/mission/data-scientist/v10

Date: August 12, 2026

Contents

Customer Decision Hub overview

Description

Familiarize yourself with the one-to-one customer engagement paradigm and discover how Pega's omnichannel AI delivers the right action during every customer interaction.

Discover Next-Best-Action Designer, a powerful tool that guides decisioning architects in creating personalized customer experiences across all channels. Define business structures, implement constraints, configure prioritization, and deliver the next best actions through inbound, outbound, and paid channels.

Explore the Impact Analyzer tool, that conducts experiments by implementing alternative prioritization and engagement policy filters on a small control group of customers, to assess the effectiveness of next best actions.

Learning objectives

- Explain the basics of the Next-Best-Action approach.
- Describe the purpose of the Next-Best-Action Designer components: Taxonomy, Constraints, Engagement policy, Arbitration, and Channels.
- Analyze the decisions for a specific customer in Customer Profile Viewer.
- Use Impact Analyzer to monitor the performance of your next best actions.

One-to-one Customer Engagement paradigm

1:1 Customer Engagement paradigm

The optimal outcome of every customer interaction is to provide a great experience while maximizing the value of the customer to the company. To achieve this outcome, you must perform the right action in the right channel at the right moment for each customer. In Pega Customer Decision Hub™, this feature is known as One-to-one Customer Engagement.

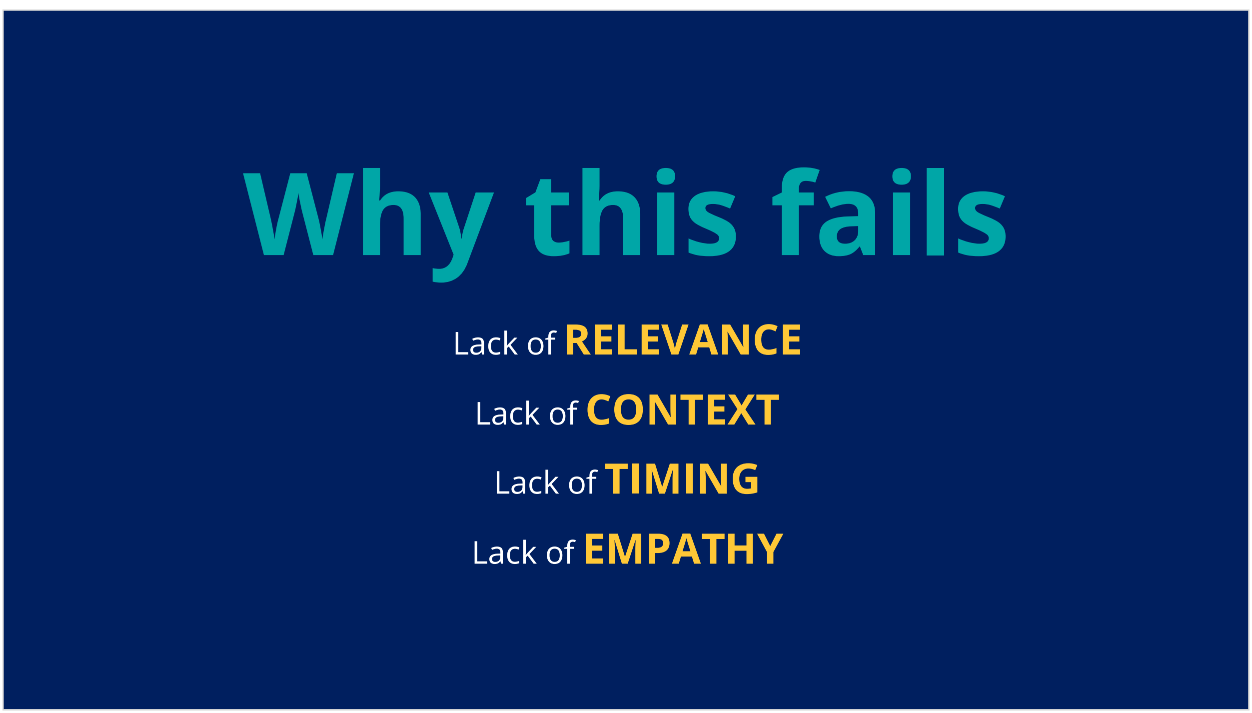
Transcript

The optimal outcome of every customer interaction is to provide a great experience while maximizing the value of the customer to the company.

In a traditional approach marketers search for potential customers from a database based on target demographics, geographies, or financial means to make a purchase. Then, marketers target this customer segment across all channels, and all these customers receive offers for a specific product.

The problem with this approach is that only a low percentage of customers respond or make a purchase. Marketers might hit their short-term goal, but in the long term, this approach affects the relationship with customers.

As a result, the traditional marketing approach fails because of a lack of relevance, context, timing, and empathy.



Customers are more empowered than ever before. As a result, they have very high expectations for the experiences they receive from their service providers. Their experiences must make sense within the context of their lives and this means they must be meaningful, consistent, and personalized across every channel with which they interact.

In business, the optimal outcome of every customer interaction is to provide a great experience while maximizing the customer's value to the company. To achieve this, you must perform the right action in the right channel at the right moment for each customer.

Traditional mass-marketing techniques that use segments, batches, and campaigns do not move the needle anymore; they are antiquated and unsustainable. The market has recentered around real-time technologies and one-to-one customer interactions.

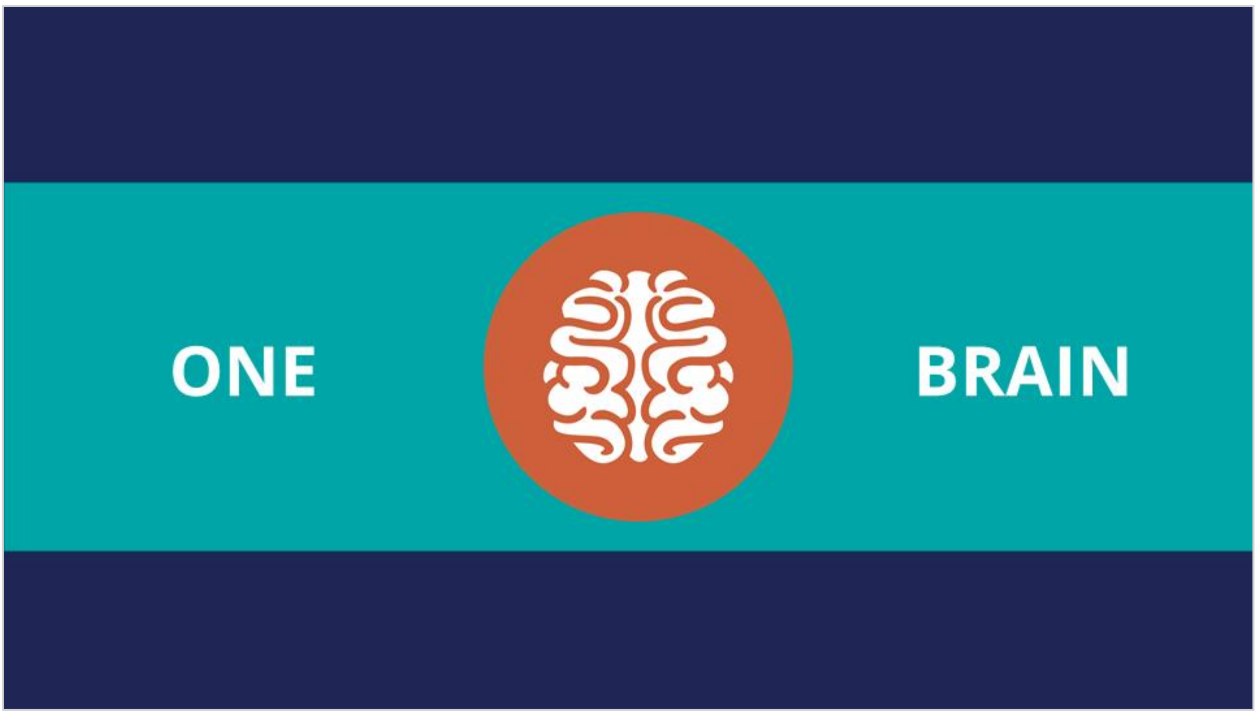
The Pega approach to these interactions is through One-to-one Customer Engagement.

Through One-to-one Customer Engagement, companies can transition their marketing away from a traditional one-to-many campaign-driven approach. A one-to-one approach allows companies to have consistent, contextual, and relevant conversations with individual customers across any channel or touch point.

The key to achieving One-to-one Customer Engagement is one centralized brain.

In other words, one piece of intelligence acts as a single decision authority across your application ecosystem.

Each channel or system profits from this single source of customer intelligence and can use it to gain insights or perform relevant actions.



In Pega Customer Decision Hub, this centralized brain is the core feature that uses AI to enable One-to-one Customer Engagement.

In Pega Infinity, Pega Customer Decision Hub forms the core of the customer engagement platform, which sits at the center of existing systems and channels in an enterprise.

The “brain” collects data from every customer engagement across the enterprise to create predictions and decisions about every interaction in every channel.

Pega Customer Decision Hub can directly integrate with third-party content management platforms such as Adobe Experience Manager and use the content that you develop in these platforms for personalized customer engagement.

Continuous learning and decision-making are the foundation of a One-to-one Customer Engagement solution.

Customer Decision Hub combines analytics, business rules, customer data, and data collected during each customer interaction to create a set of actionable insights that it uses to make intelligent decisions. These decisions are known as the next best actions.

Every next best action weighs customer needs against business needs to optimize decisions based on priorities set by the business manager.

In the milliseconds before interacting with a customer, Customer Decision Hub combines thousands of business rules and predictive and adaptive models that determine customer needs and interests to ensure the next best action is relevant by keeping it personal, timely, contextual, and empathetic.

Customer Decision Hub identifies the best moments to make a sale, provide a service, make a retention offer, inform about an offer, or do nothing at all (for example, if no offer is relevant enough to warrant the customer's attention). The system distributes next-best-action decisions in real time to your channels, such as the web, mobile, and contact center. Pega Customer Decision Hub can also distribute next best actions to real-time paid channels such as Google, YouTube, Facebook, LinkedIn, and Instagram. Pega Customer Decision Hub also integrates with non-real-time outbound channels such as data management platforms (DMPs) and email.

After the system distributes the next best actions and the "brain" receives customer responses, the process begins again. Customer Decision Hub distributes new next best actions in milliseconds. It captures every customer interaction in every channel to ensure consistency and an optimized customer experience across channels.

For example, consider a customer, Miranda. With the centralized brain in place, instead of looking only at sales offers, the system begins with a list of all potential actions, such as service, retention, nurture, or a hardship message, everything that you can do for this customer now.

Although AI drives the next best actions, AI does not run rampant; it experiments on every person with every topic. Customer Decision Hub still allows marketers to maintain control and establish the criteria that AI must meet to consider beginning one of these conversations.

For example, you establish eligibility rules that state the business cannot sell a card to someone under the age of 18.

Then there are applicability rules to define if the action is appropriate at a given time. If Miranda already owns a competing or more valuable product, the business does not offer this product to her even though she is eligible.

Suitability rules determine whether an action is in the best interest of a customer. For example, Miranda might be eligible for a card, and her current card does not offer as high of a cash-back offer. But because Miranda cannot make her monthly payments and might end up in collections, the business does not offer the card to her even though it can.

So, if a customer fails to meet the conditions that you establish for an action, the AI does not consider that action.

Next-Best-Action Decisions

Sales Offers	Service Nudges
Retention Offers	Nurture Messages

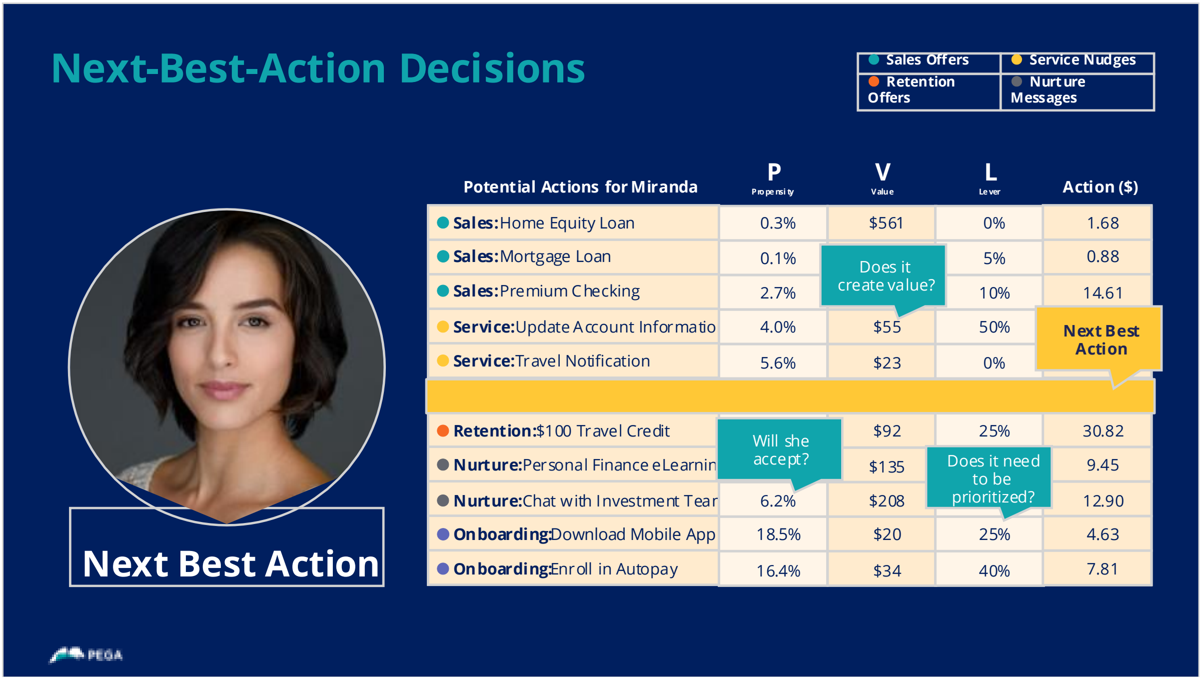
Potential Actions for Miranda	Eligible?	Applicable?	Suitable?	Arbitrate?
~~**Sales:** Rewards Card~~	✓	✓	X	No
Sales: Home Equity Loan	✓	✓	✓	Yes
Sales: Mortgage Loan	✓	✓	✓	Yes
Sales: Premium Checking	✓	✓	✓	Yes
Service: Update Account Informatio	✓	✓	✓	Yes
~~**Service:** Fraud Alert~~	✓	X	—	No
Service: Travel Notification	✓	✓	✓	Yes
Retention: Waive Annual Fee	✓	✓	✓	Yes
~~**Retention:** Double Rewards Points~~	X	—	—	No
Retention: $100 Travel Credit	✓	✓	✓	Yes
Nurture: Personal Finance eLearnin	✓	✓	✓	Yes
Nurture: Chat with Investment Tear	✓	✓	✓	Yes
Onboarding: Download Mobile App	✓	✓	✓	Yes
Onboarding: Enroll in Autopay	✓	✓	✓	Yes

Should we arbitrate?

PEGA

Once the system narrows down the list, AI takes over. AI first determines how likely each action is desirable to the customer. Next, it determines the value each option generates for the business. What is the impact if the customer reads the content or accepts an offer? Does it give you more revenue or reduce your costs?
You can add levers to make adjustments based on the current business situation. For example, the business can decide to nudge an offer if it needs to meet its financial goals or ramp down an offer if it runs low on inventory.

For Miranda, the action that the system selects as the next best action is the one with the highest P*V*L value.

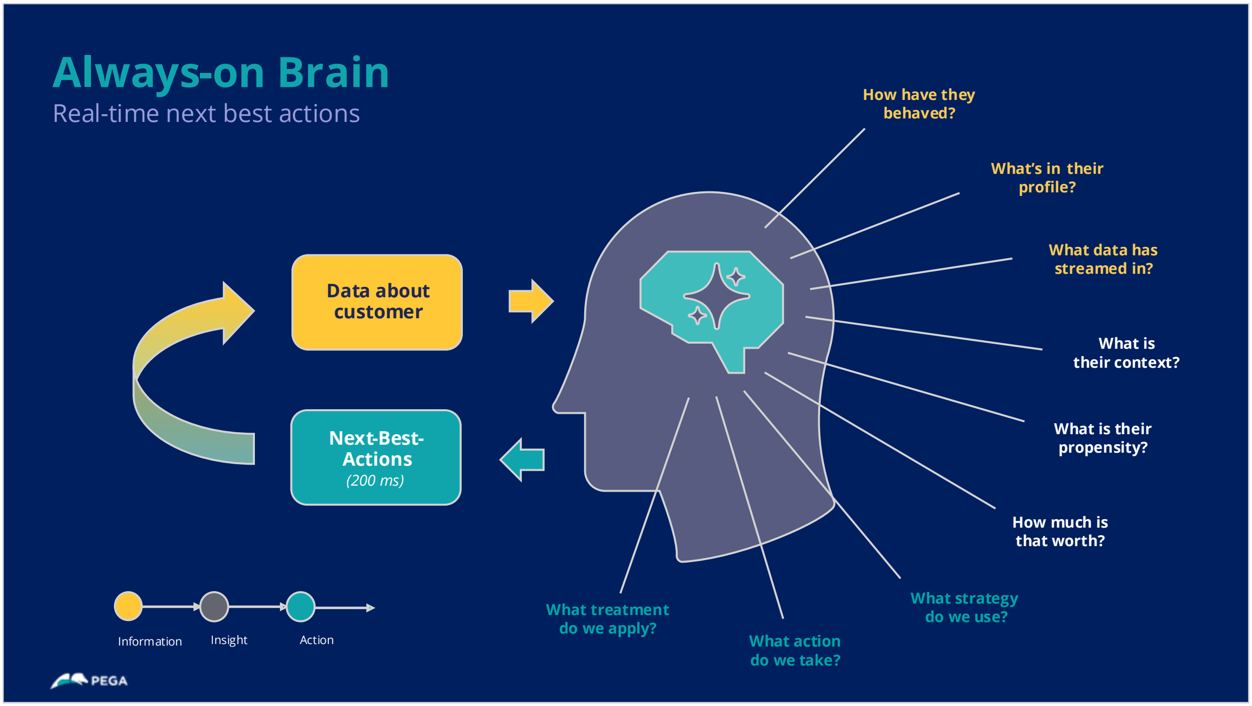


In summary, Customer Decision Hub is the always-on brain that acts as a single, centralized decision authority.

It uses data about the customer, including past interactions, as input.

It uses advanced AI techniques to make predictions.

Decision strategies (which combine traditional business rules with predictive, adaptive, and text analytics) deliver consistent and personalized next best actions across all channels.



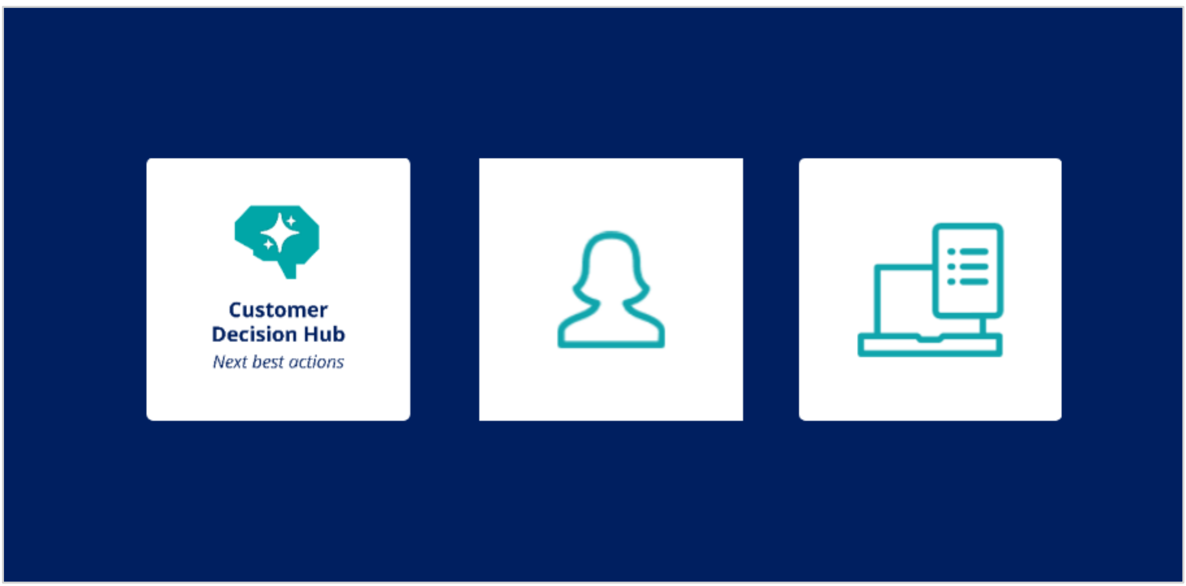
You have reached the end of this video.

The cross-sell on a website use case

U+ Bank uses Pega Customer Decision Hub™ to ensure that the customers of U+ Bank see the tailor-made offers when visiting its website. The centralized decision management "brain" of Customer Decision Hub selects the next best action that is displayed for a customer based on configurations implemented by the Decisioning Architect.

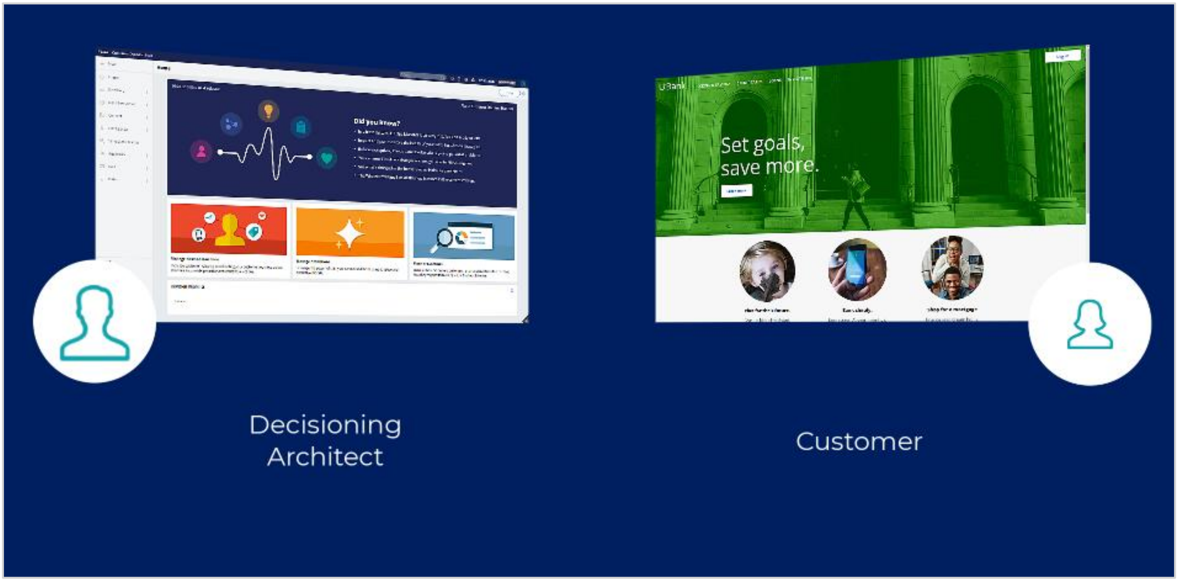
Transcript

Customer Decision Hub can deliver the next-best-action recommendations through various inbound and outbound channels. Outbound messages involve proactively reaching out to customers, while inbound channels demand action from a customer. One such inbound channel is the web. For example, when a customer visits the website, they see the intended offers.



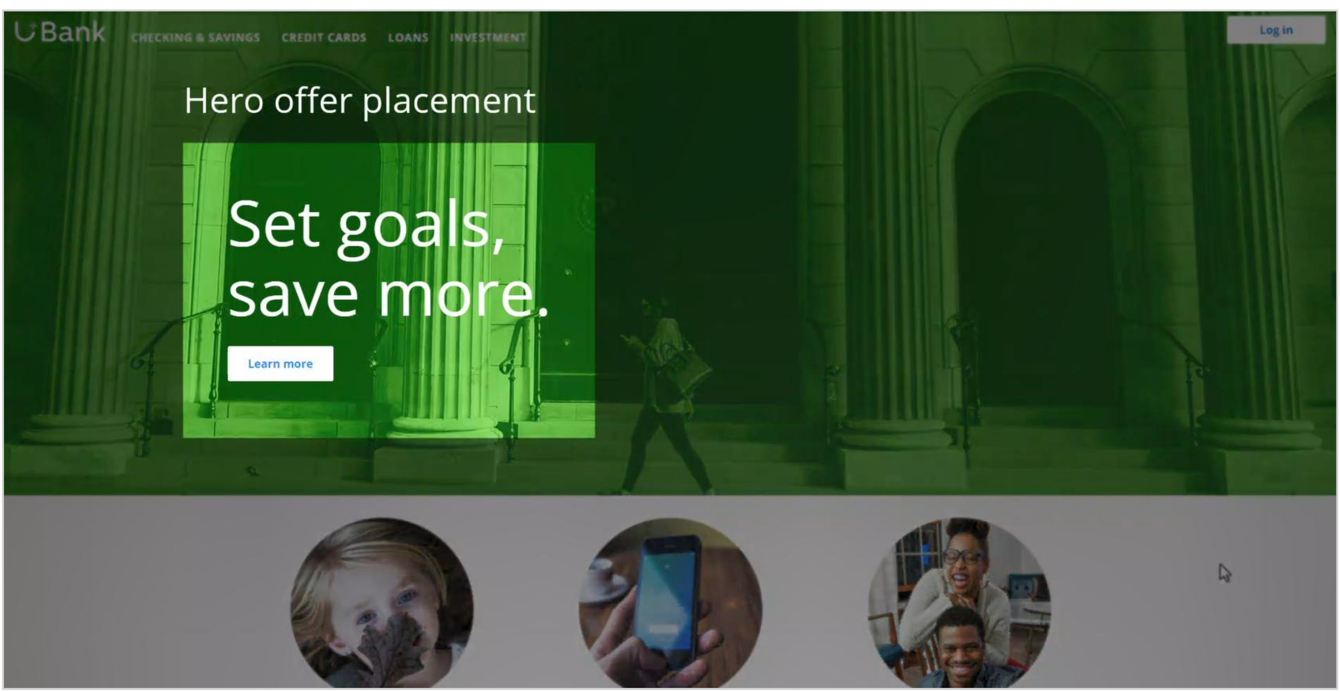
Consider the following web channel scenario, which is a typical cross-selling use case.

The next-best-action recommendations help to ensure that the customers of U+ Bank can see the tailor-made offer when visiting the website of the bank. The centralized decision management "brain" of Customer Decision Hub selects the next best action that is displayed for a customer based on configurations implemented by the Decisioning Architect.

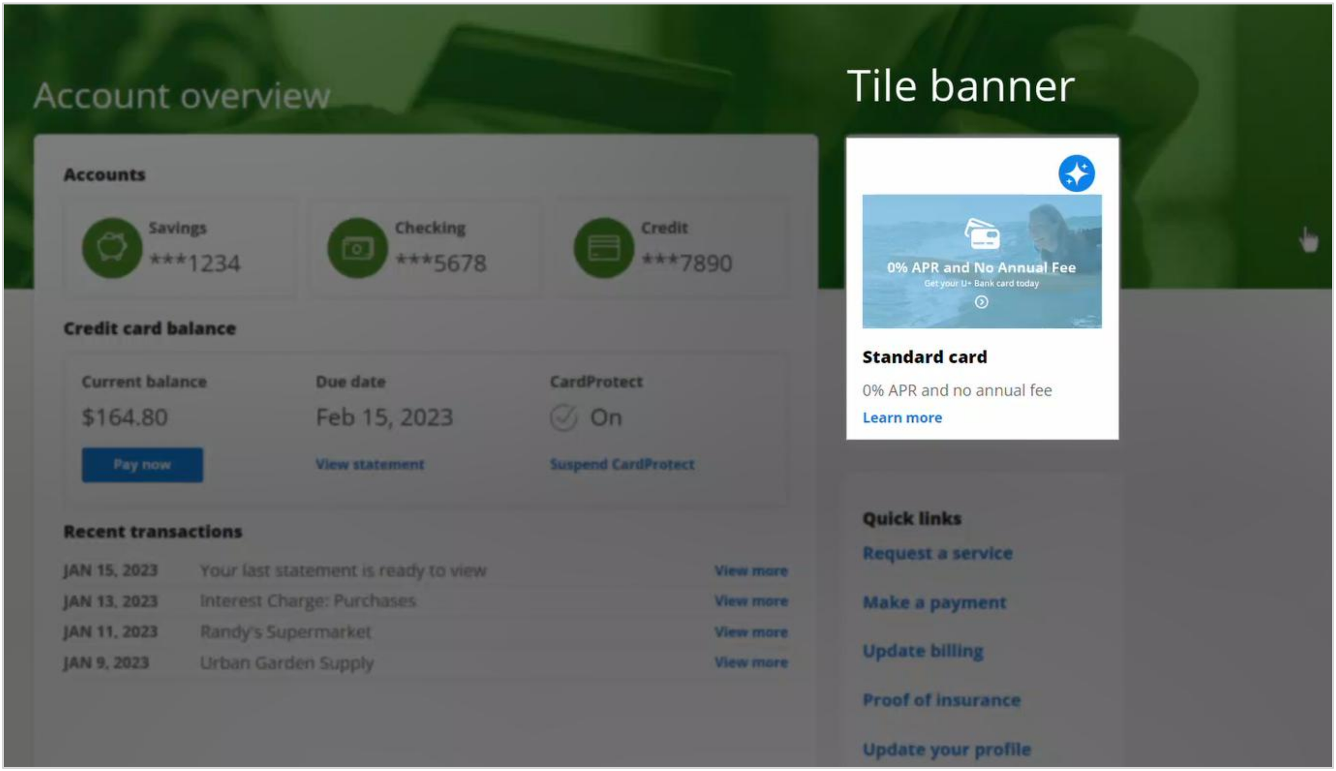


U+ is a retail bank that wants to use its website as a marketing channel to improve One-to-one Customer Engagement, drive sales, and deliver next best actions in real time. The bank has decided to use Pega Customer Decision Hub™ to recommend more relevant banner ads to its customers when they visit the website.

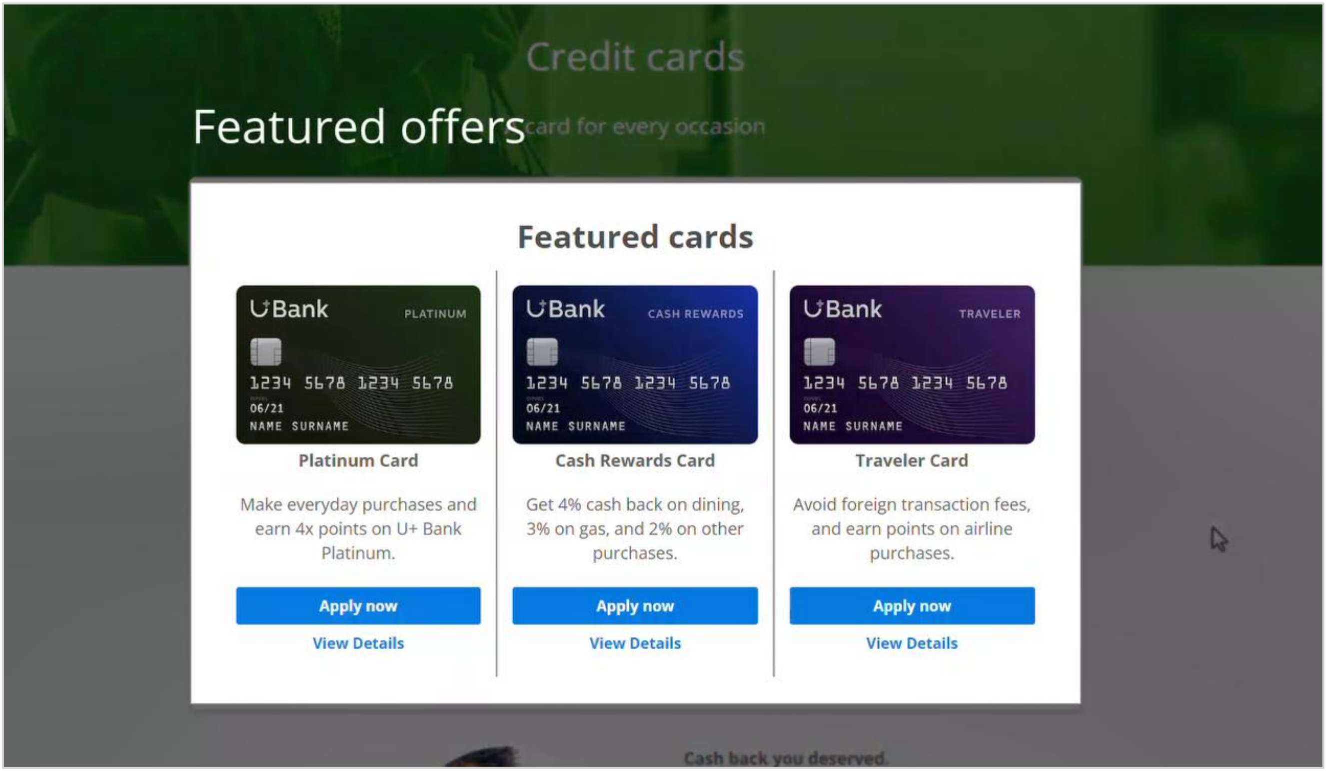
When visiting the U+ Bank website, Troy, a customer, can see banner ads on various pages. For example, on the home page, U+ displays a hero banner at the top of the page, which is typically a larger image with a larger typeface. Under that banner, there is space to display several tile banners, which are typically smaller.



When Troy logs in to his personal portal, he also sees a tile banner on the **Account overview** page.

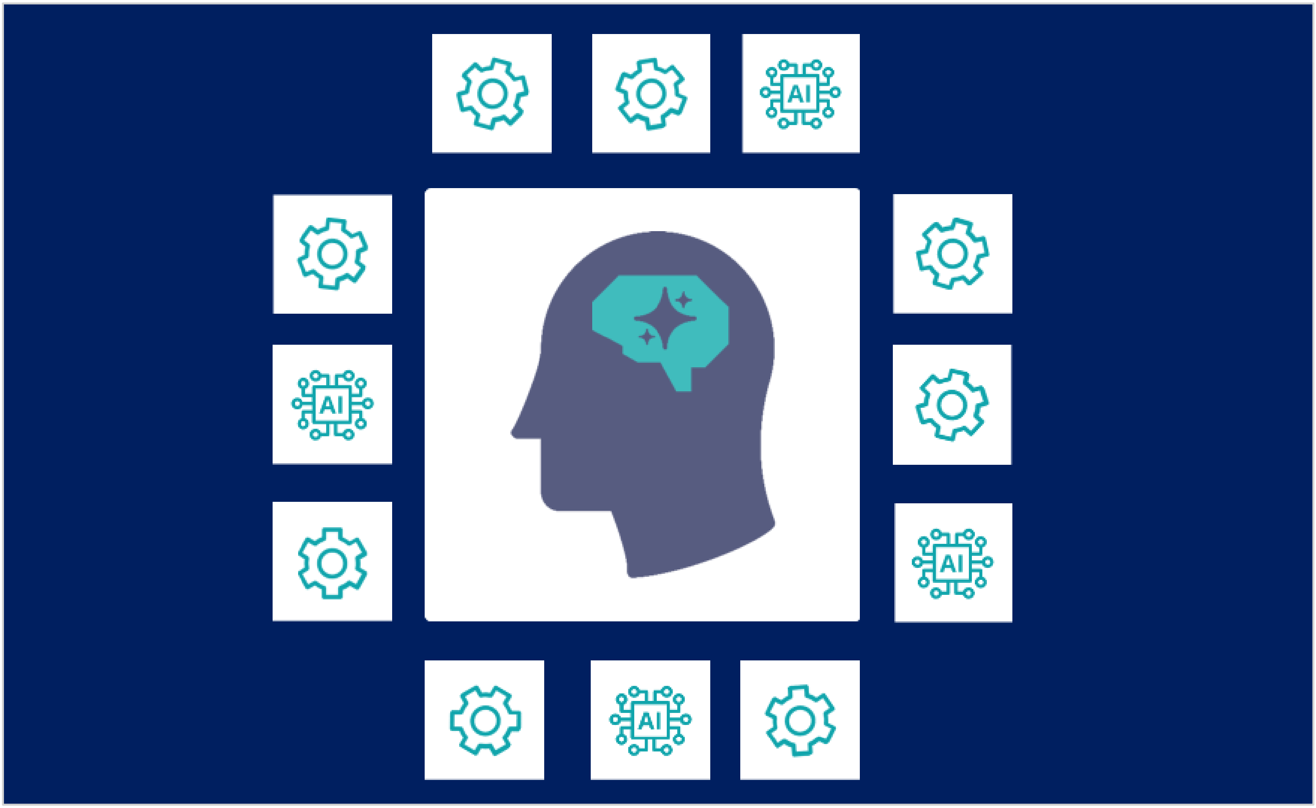


The main goal of U+ at this stage is to increase customer web engagement. When Troy clicks the **Learn more** link, the action shows his interest. This interaction is recorded as a click-through and helps measure the web engagement of the customer. After clicking **Learn more**, Troy can see featured credit card offers for which he can apply.



Behind the scenes of the next best actions, the complex decision engine is working to rank and select the best offer to display for each customer who visits the website.

A combination of artificial intelligence (AI) and other business rules determine the options.



To summarize, when visiting the bank's website, the customer can see the next best action that the "always on" centralized decision management "brain" of Customer Decision Hub selected from a set of actions defined and configured by the Decisioning Architect.

You have reached the end of this video.

Exploring Next-best-Action decisions

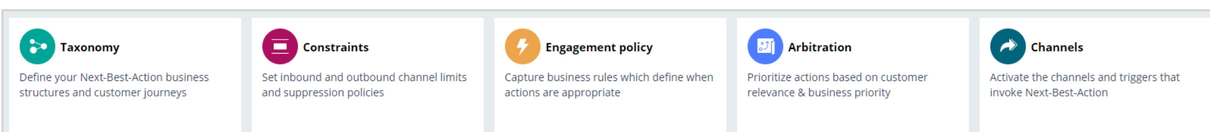
Next-Best-Action Designer organizes the high-level sequence of steps needed to customize the next-best-action strategy to suit the business requirements. Pega Customer Decision Hub™ uses these settings to generate an underlying next-best-action strategy framework and the enterprise-level strategies that come with it.

These decision strategies determine the personalized set of next best actions for each customer. To observe the decisions made for a specific customer, explore the Customer Profile Viewer.

Transcript

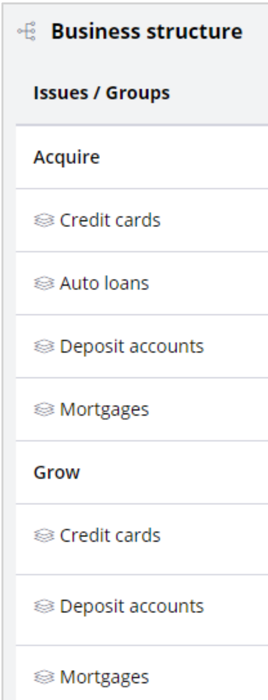
Next-Best-Action Designer guides the decisioning architect by creating a next-best-action strategy that delivers personalized customer experiences across all channels. The tabs across the top of the user interface represent the steps to define the next best actions:

- Use the **Taxonomy** component to define the business structure for your organization.
- Use the **Constraints** component to implement channel limits and constraints.
- Use the **Engagement policy** component to define the rules that control for which actions customers qualify for.
- Use the **Arbitration** component to configure the prioritization of actions.
- Use the **Channels** component to configure when and where the next best actions initiate.



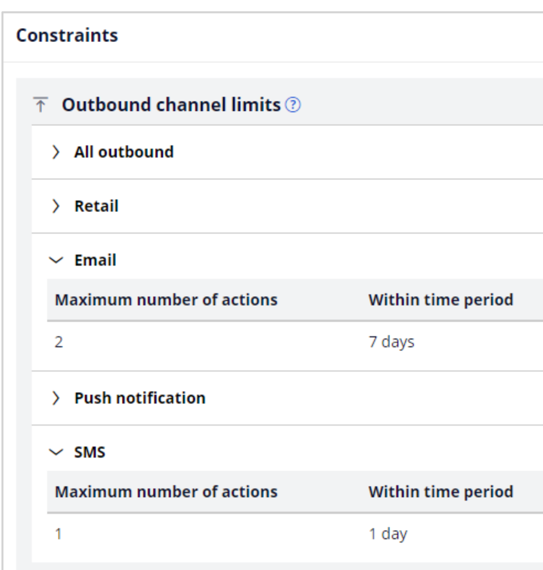
The system uses these definitions to create an underlying Next-Best-Action Strategy framework. These decision strategies are a combination of the business rules and AI models that form the core of Customer Decision Hub, which determines the personalized set of next best actions for each customer.

Use the **Taxonomy** tab to define the hierarchy of business issues and groups to which an action belongs.



A business issue is the purpose behind the actions that you offer to customers. For example, actions to acquire new customers belong to the **Acquire** business issue. The **Grow** business issue groups the actions to cross-sell to existing customers. Business groups organize customer actions into categories. You can create groups for products, such as credit cards, mortgages, or auto loans, to offer these to potential customers.

Use **Constraints** to specify contact limits and limit overexposure to a specific action or group of actions. For example, you do not want your customers to receive more than two emails per week or one SMS message daily.

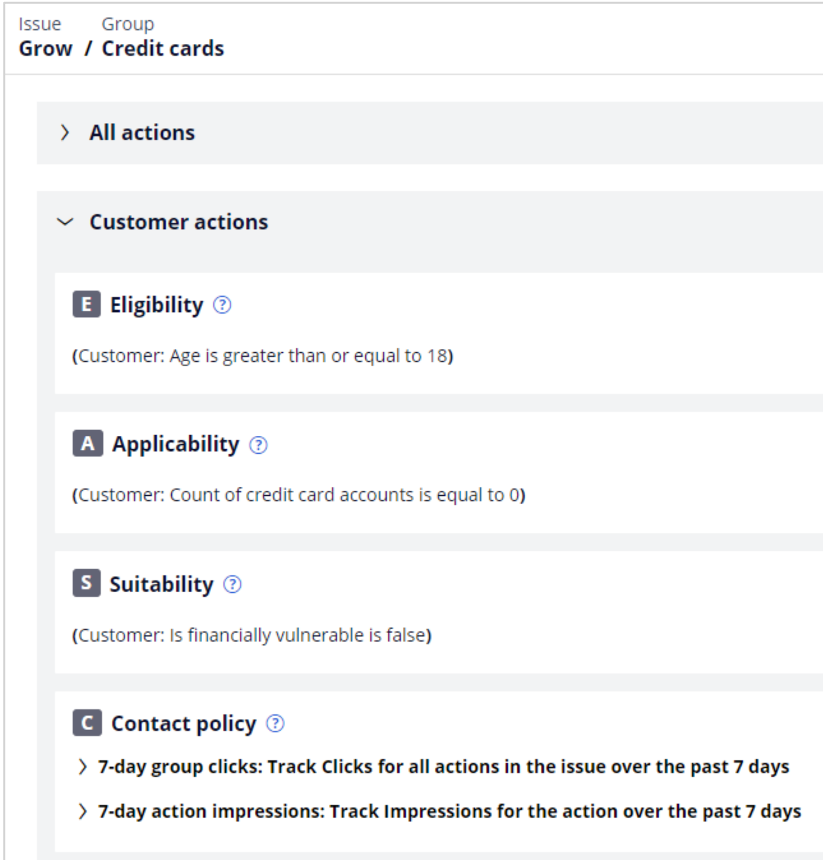


You can define more extensive suppression rules by creating Contact Policy rules in the library. Contact Policy rules are reusable across all business issues and groups. For example, you suppress an action for a customer for seven days after the customer has seen an ad for that action five times.

Use **Engagement policies** to define when specific actions or groups of actions are appropriate for customers. There are four types of engagement policies:

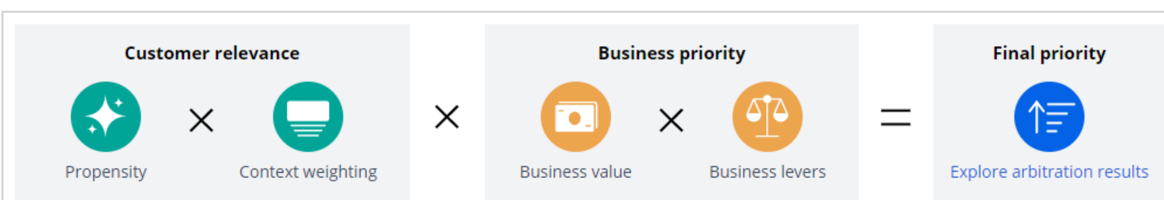
- **Eligibility** determines whether a customer qualifies for an action or group of actions. For example, an action might only be available for customers over a specific age or who live in a specific geographic location.
- **Applicability** determines whether an action or group of actions is relevant for a customer at a particular point in time. For example, a discount on a specific credit card might not be relevant for a customer who already owns a card.
- **Suitability** determines whether an action or group of actions is appropriate for a customer for ethical or empathetic reasons. For example, a credit card offer might not be appropriate for a customer who is financially vulnerable, even though it might be profitable for the bank.

- **Contact Policies** determine when to suppress an action or group of actions and for how long. For example, you can suppress an action after the customer receives a specific number of



promotional messages.

Arbitration determines how Customer Decision Hub prioritizes the list of eligible and appropriate actions from each group.



The factors that the system weighs in arbitration are **Propensity**, **Context weighting**, **Business value**, and **Business levers**. Numerical values represent the factors.

Propensity is the likelihood that a customer responds positively to an action, and AI calculates the propensity. For example, clicking an offer banner or accepting an offer in the contact center are considered positive behaviors.

Context weighting allows you to assign a weighting to a specific context value for all actions within an issue or group. For example, if a customer contacts the bank to change the home address, the weight of the **Service** context increases, and the highest priority action is to ensure that the system delivers the relevant service to the customer.

Business value enables you to assign a financial value to an action and prioritize high-value actions over low-value ones. For example, promoting an unlimited data plan might be more profitable for the company than a limited data plan.

Business levers enable you to accommodate urgent business priorities by specifying a weight for an action, group, or business issue. A simple formula determines a prioritization value, which is used to select the top actions.

Next-Best-Action Designer enables the delivery of next best actions through inbound, outbound, and paid channels. A trigger is a mechanism where an external channel, for example, a website, invokes the execution of a Next-Best-Action decisioning process for specific issues and groups. The result returns to the invoking channel. A real-time container is a placeholder for content in an external real-time channel. For example, a website invokes a real-time container, *TopOffers*, before loading the account landing page.

Triggers

Real-time containers

Status	Name	Description	Business structure level
ACTIVE	NextBestAction	NextBestAction	All issues / All groups
ACTIVE	CreditCards	CreditCards	Grow / Credit cards
ACTIVE	TopOffers	TopOffers	Grow / All groups
ACTIVE	Collections	Collections	Collections / All groups

You can explore the arbitration matrix in the Customer Profile Viewer report to understand what actions the system offers to customers. For example, U+ Bank uses Customer Decision Hub to decide which credit card offer to show when customers log in to the bank's website. Joanna is a U+ Bank customer.

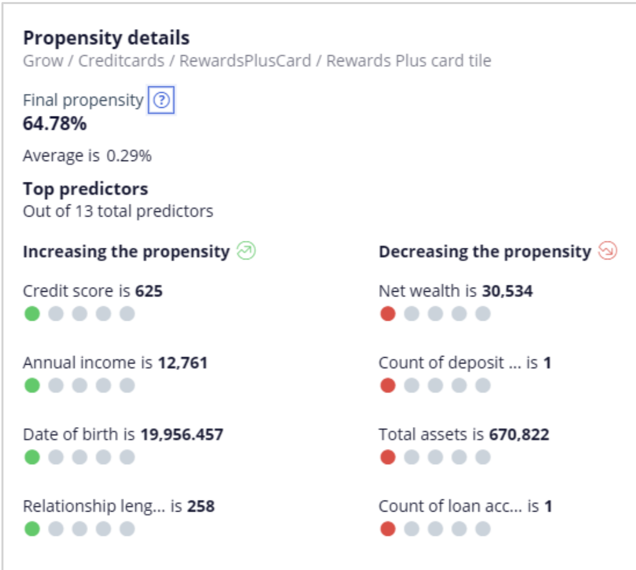
For this use case, the direction is inbound, the channel is the web, and TopOffers is the real-time container service that manages communication between Customer Decision Hub and the website. Joanna meets all criteria for two credit card offers: the Rewards Plus card, and the Premium Rewards card.

The **Explain prioritization** view shows the values for the arbitration formula components: propensity, context weight, business value, and lever weight. The business value of the Premier Rewards card is higher than the value of the Rewards Plus card. But the likelihood that Joanna is interested in a credit card offer is highest for the Rewards Plus card, which is the top offer.

Explain prioritization | Group | Fields | Export complete report

Name	Treatment	Channel	Journey	Stage	Final propensity	Total context weight	Value	Total lever weight	Priority	Control group name	Reason	Rank	results
RewardsPlusCard	Rewards Plus card tile	Web	—	—	64.78%	1.00	40.0	1.4	36.28	—	—	1	PASSED
PremierRewardsCard	Premier Rewards card tile	Web	—	—	48.83%	1.00	45.0	0.85	18.68	—	—	2	PASSED

To aid the transparency of the AI, you can view the predictors that are most influential in the calculation of the propensity.



To understand how the computation for the action propensity works, inspect the **Explain propensity** view, where you can see the model evidence, the number of positive responses, the initial model raw propensity and the final propensity that Customer Decision Hub uses in arbitration.

Explain propensity | Group | Fields | Export complete report

Name	Treatment	Channel	Model control group	Model evidence	Model positives	Model propensity	Final propensity	Rank	results
RewardsPlusCard	Rewards Plus card tile	Web	NBA	3928	1147	65.74%	64.78%	1	PASSED
PremierRewardsCard	Premier Rewards card tile	Web	NBA	3835	1316	48.04%	48.83%	2	PASSED

You have reached the end of this video. You have learned:

- How Next-Best-Action Designer is organized according to the following high-level sequence of steps that are necessary to configure the next-best-action strategy:
    - Defining the business structure for your organization.
    - Implementing the channel limits and constraints.
    - Defining the contact policies that control for which actions a customer qualifies.
    - Configuring the prioritization of actions.
    - Configuring when and where to initiate the next best actions.
- How you can analyze the decisions for a specific customer in Customer Profile Viewer to aid model transparency.

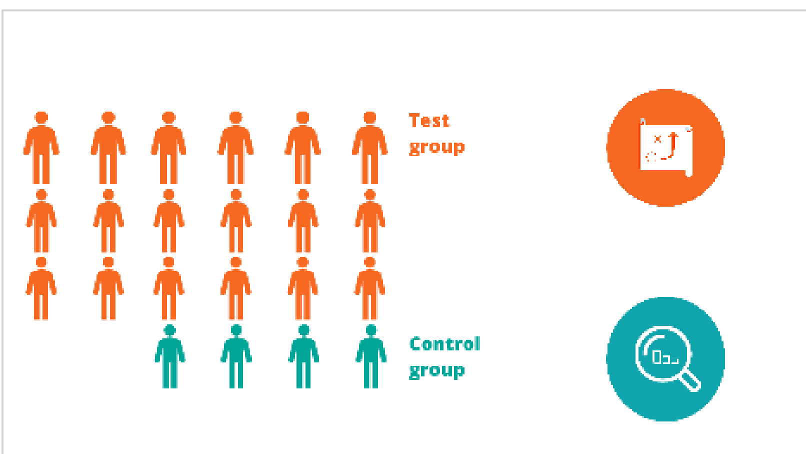
Monitoring the health of the system using Impact Analyzer

Impact Analyzer is a proactive monitoring tool that shows the overall health of your actions and highlights opportunities for improvement. The tool provides insight into the performance of your next best actions through the results of experiments that tell you what actions and configurations are working as expected, and where you can still improve.

Transcript

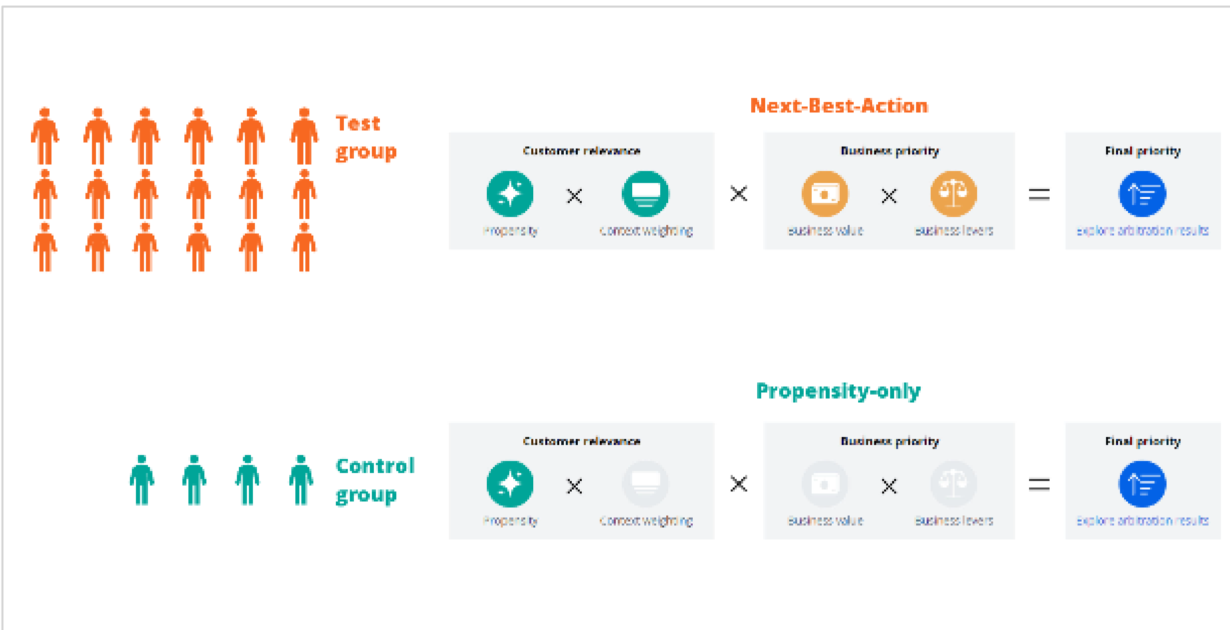
Impact Analyzer is a feature of Pega Customer Decision Hub™ that uses experimentation to track the effectiveness and overall health of next best actions. By utilizing this tool, you can gain insights into the effects of arbitration. Impact Analyzer conducts experiments with alternative prioritization and engagement policy filters on a small control group of customers. This enables you to assess whether or not the next best action is producing a positive effect. In addition to monitoring for basic lift, these control group experiments can also reveal opportunities to optimize and align the next best actions with your business goals.

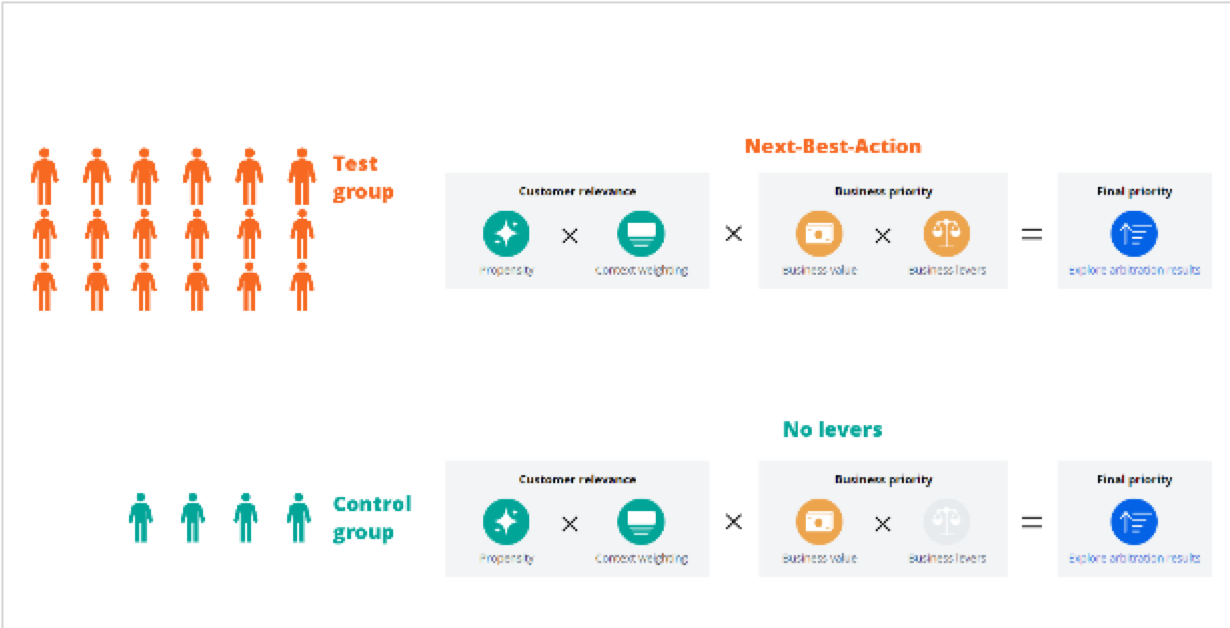
An experiment is a test with two sets of participants, the test group and the control group, in which the test group receives the originally arbitrated action and the control group receives an alternate action based on the type of experiment.

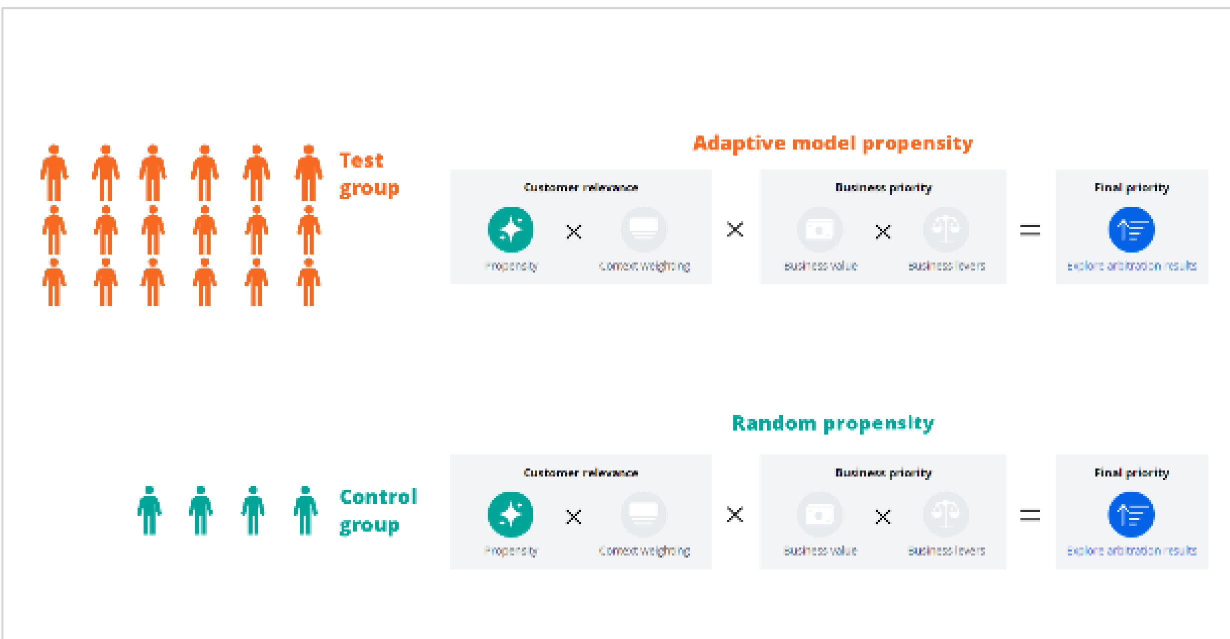


You can use Impact Analyzer to carry out a variety of tests. For example, you can assess the effectiveness of the next best action against a random relevant action or actions, chosen based on propensity alone. Additionally, you can measure the performance of the next best action when arbitrating with no levers. You can also evaluate the performance of the adaptive model propensity against a random propensity.


Test group
Next-Best-Action
Customer relevance
Business priority
Final priority
Random, relevant action
Control group
Customer relevance
Business priority
Final priority

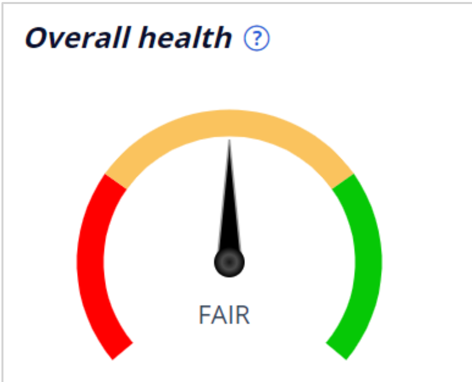

Test group
Next-Best-Action
Customer relevance
Business priority
Final priority
Propensity-only
Control group
Customer relevance
Business priority
Final priority





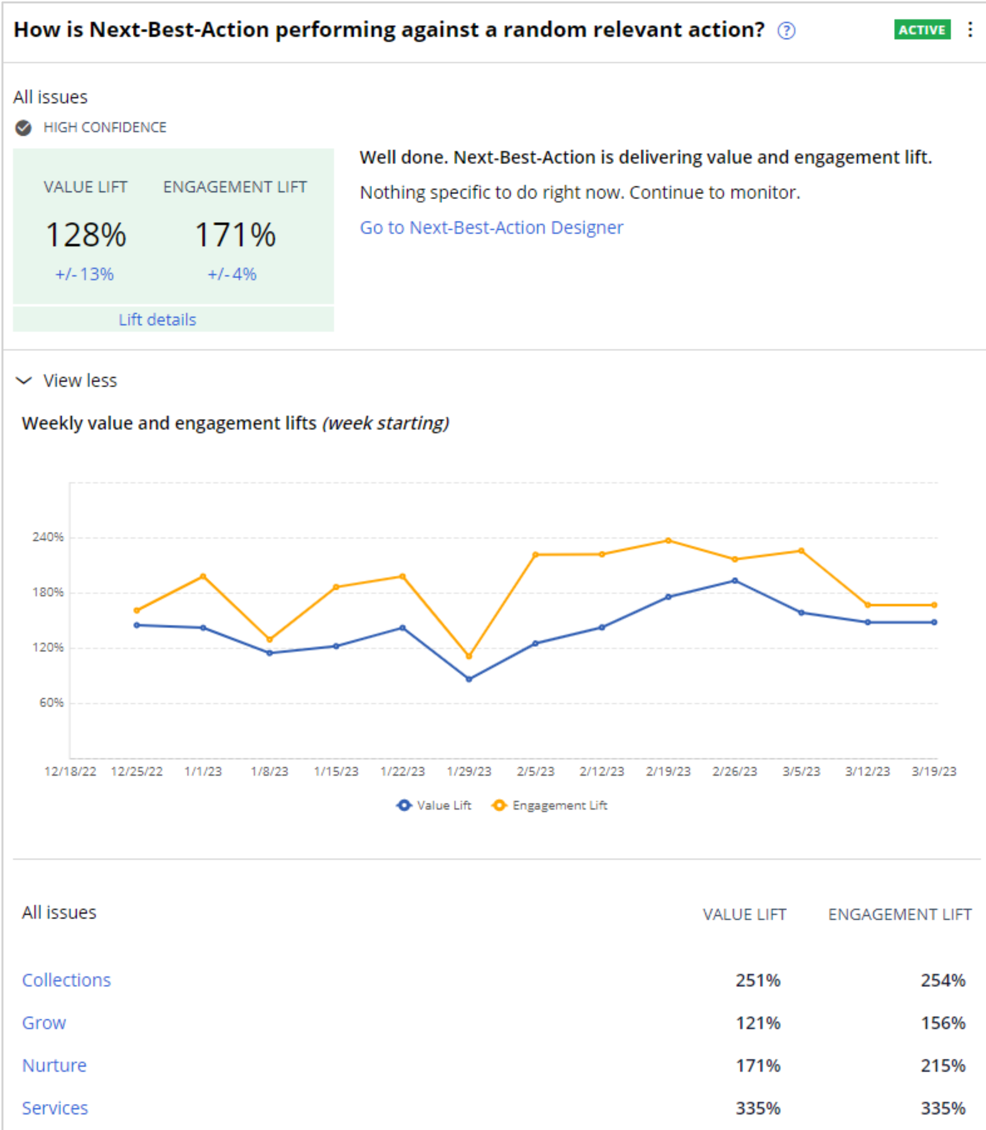
Impact analyzer continuously monitors the health of the next best actions in the experiment, and presents the results in the form of two key performance indicators (KPIs): value lift, and engagement lift. Value lift measures how much a business can make from suggesting certain next best actions to customers, compared to a random relevant action. Engagement lift measures how much more likely customers are to click or accept a suggested next best action, compared to the random relevant action.

When a next best action is delivering lift, the overall health indicator highlights in green. The middle position of the indicator shows that there is room for improvement to better align actions to the business. The red highlight [the section on the left] means that something is wrong and needs your attention. The indicator is colored gray when the general health is not available due to insufficient data in one or more active tests.



Impact Analyzer is available under **Discovery** tab in the left menu of Customer Decision Hub.

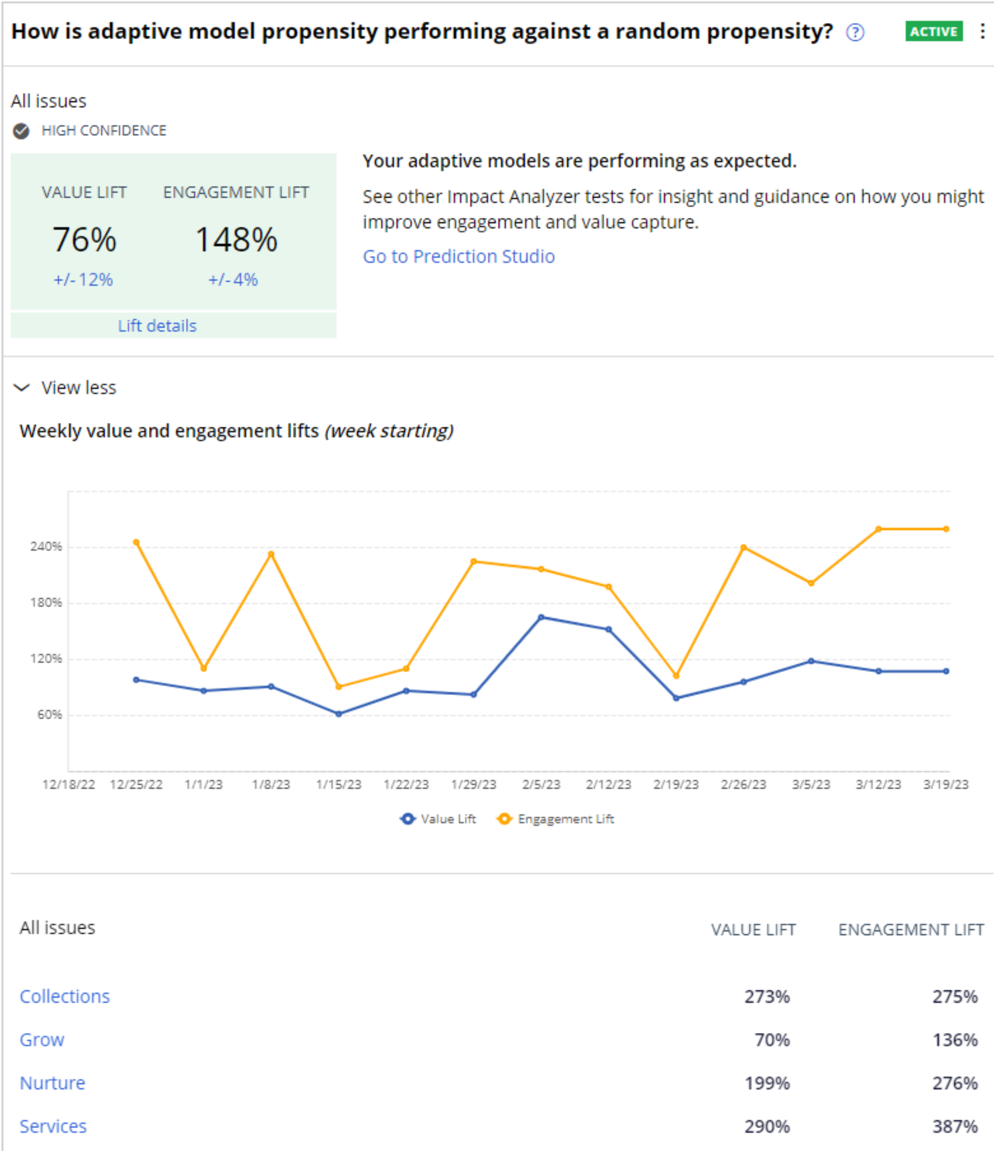
One of the experiments that you can run is called *How is Next-Best-Action performing against a random relevant action?*



This test serves as a baseline check to ensure that the next best action is beating the random relevant action. At the very least, we expect the value lift - and ideally, both the value and engagement lift - to be positive, at the highest level across all issues and groups aggregated. If both value and engagement lifts are negative, there is an issue. This could mean that there is a problem with the underlying adaptive models, or that your engagement policies are too targeted. Another reason might be that your levers are excessively biased, and the next best action lacks the aptitude to optimize engagement or to capture value.

The test widget presents a value lift of 128 percent and an engagement lift of 171 percent. Both values are positive, and the test is highlighted in green.

Another experiment is called *How is adaptive model propensity performing against a random propensity?*



This experiment compares only Pega AI-driven model propensity with no context weight, no business value, and no business levers, against only randomly generated propensity. The experiment is another baseline check to ensure that the adaptive models that a next best action uses are delivering lift over random propensity.

To monitor for basic lift, both presented experiments should be on at all times.

Impact Analyzer offers a range of benefits to its users. One of the most significant benefits is the ability to collect data in real-time, allowing for real-time monitoring and testing of KPIs. Impact Analyzer also enables users to look for opportunities that they may have otherwise missed.

The primary user of Impact Analyzer is the Next Best Actions operations team, who are responsible for ensuring that the system runs optimally. However, the tool brings value to various lines of business, sales, marketing, and Data Scientists.

You have reached the end of this video. You have learned:

- How Impact Analyzer continuously monitors the performance and health of the system.
- How Impact Analyzer conducts experiments by implementing alternative prioritization and engagement policy filters on a small control group of customers, to assess the effectiveness of next best actions.
- How the experiments reveal opportunities to optimize and align next best actions with business goals.

Customer Decision Hub predictions

Description

Discover the Pega Customer Decision Hub™ predictions in the Prediction Studio portal, a comprehensive workspace for Data Scientists who manage customer interactions. Delve into predefined predictions for communication channels, adaptive models, and customer-level decisions to optimize customer experiences and effectively respond to changes in customer behavior.

Familiarize yourself with the tools and processes used to manage business changes in the Pega 1:1 Customer Engagement solution.

Learning objectives

- Describe how adaptive models drive the predefined Pega Customer Decision Hub™ predictions.
- Describe how the outcomes of supporting predictive models can function as predictors for the adaptive models.
- Describe the tasks of a Data Scientist in a Customer Decision Hub project.
- Describe the steps involved in managing business changes in one-to-one customer engagement.

Customer Decision Hub predictions

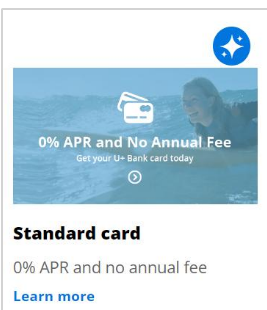
Prediction Studio is the workspace in which you manage the predictions that Pega Customer Decision Hub™ uses to optimize customer interactions.

Transcript

This video provides an overview of the Customer Decision Hub predictions in the Prediction Studio portal. Prediction Studio is the workspace for data scientists, offering tools to manage predictions used by Customer Decision Hub for optimizing customer interactions.

Decision strategies utilize predictions to determine the optimal action and treatment for customers across all interaction channels. Customer Decision Hub includes predefined predictions for each relevant communication channel. The predefined *Predict Inbound CallCenter Propensity* prediction calculates the likelihood of a customer accepting a proposition offered by a Customer Service Representative when contacting the call center. The *Outbound Email Propensity* prediction calculates the likelihood that a customer clicks a link in an email.

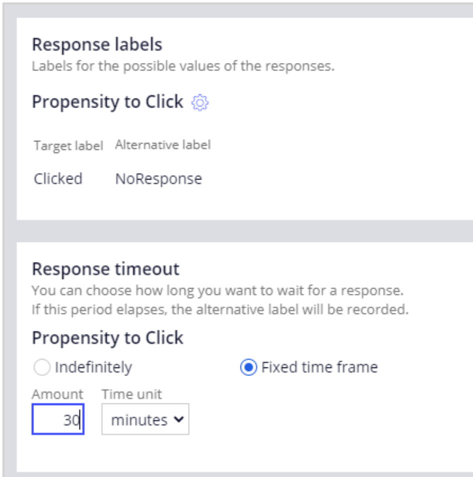
The prediction that calculates the propensity of a customer clicking on a web banner is named *Predict Web Propensity*. For instance, in a cross-sell scenario, U+Bank leverages its website as a marketing tool. Upon customer login, the website presents a personalized credit card offer in a banner.



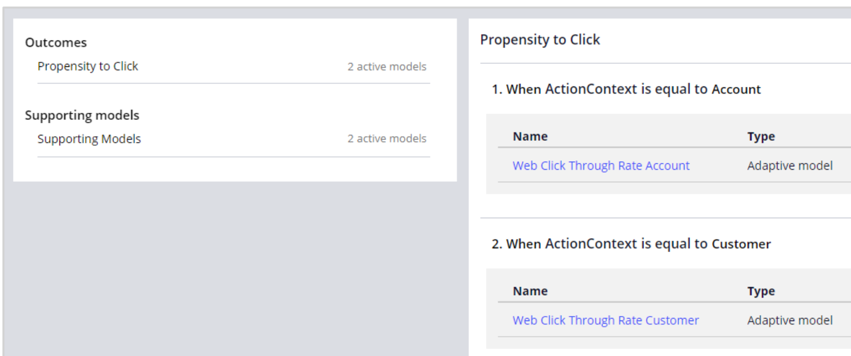
If a customer is eligible for multiple credit cards, the *Predict Web Propensity* prediction calculates the propensity of receiving a positive response from the customer for each of these cards. Customer Decision Hub determines which credit card to offer based on business rules, interaction context, and propensity.

The context of the *Predict Web Propensity* prediction includes the treatment that a customer receives, in this case, a personalized banner in the web channel. This allows the capture of variations in customer preferences, such as the background color of a credit card offer in a web banner.

The prediction's response labels are “Clicked” for the target behavior and “NoResponse” for the alternative behavior. The NoResponse outcome is recorded after the response timeout expires.



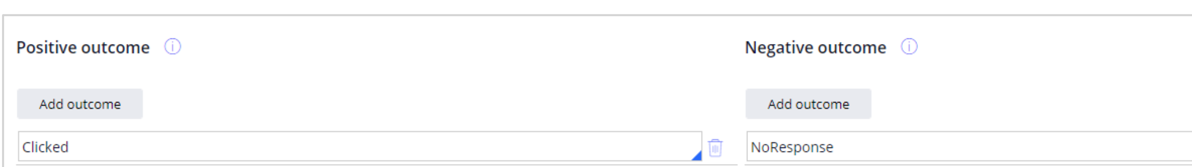
Predictive models drive predictions.



The propensity predictions integrated into Customer Decision Hub utilize adaptive models that learn from customer responses and receive automatic updates every hour. This ensures that U+Bank automatically responds to changes in customer behavior while the adaptive models self-optimize.

Customer Decision Hub makes customer-level decisions, while account-level decisions are made for a specific account associated with the customer. For example, a customer-level decision might be to send a promotional email to a customer based on the overall engagement with the company, while an account-level decision might be to send a reminder email to a customer based on their account balance.

The *Predict Web Propensity* prediction propagates the outcomes of a model based on the *Web Click Through Rate* adaptive model configuration. The outcomes of the adaptive model configurations align with the response labels of the prediction.



Business requirements might necessitate the use of a predictive model trained on historical data or a highly transparent scorecard model. These models can be incorporated as supporting models within the prediction, with their outcomes serving as predictors for the adaptive models.

You have reached the end of this video. What did it demonstrate?

- The predictions that come with Customer Decision Hub in the Prediction Studio portal.
- The *Predict Web Propensity* prediction utilizes the *Web Click Through Rate* adaptive model configuration to determine the outcomes.
- The outcomes of supporting predictive models can function as predictors for the adaptive models.

Personalizing Prediction Studio Notifications

In today's data-driven landscape, successful organizations distinguish themselves through intelligent, real-time decision-making at scale. Pega's Customer Decision Hub enables this capability, but the true value lies in ongoing optimization through robust notification management. Prediction Studio notifications serve as your organization's early warning system, providing essential visibility to maintain model health and performance before issues impact business outcomes. The challenge lies in balancing staying informed while avoiding information overload.

Transcript

Hello! I'm Iris, and today I want to share something that can make the difference between a reactive and proactive approach to managing your predictive models in Pega's Customer Decision Hub.

We're going to explore how to configure Prediction Studio notifications effectively, ensuring your data science team stays ahead of potential issues rather than scrambling to fix problems after they've impacted your business decisions.

Prediction Studio serves as your central workspace for creating, monitoring, and updating predictions along with the predictive models that power them.

Think of it as your mission control center for all things related to predictive analytics in Customer Decision Hub.

But here's the key insight: the real power lies not just in building great models, but in maintaining their performance over time through intelligent monitoring and timely interventions.

Let me walk you through the notification ecosystem that keeps your data science team informed and responsive.

Prediction Studio generates a comprehensive variety of notifications that provide crucial insights into three critical areas: the performance of your predictions, the health of your adaptive models, and the effectiveness of your predictive models overall.

Accessing these notifications is straightforward - simply click the Notifications icon in the Prediction Studio header.

But here's where it gets interesting: these notifications aren't just status updates; they're your early warning system for maintaining model quality and ensuring your Customer Decision Hub continues making optimal decisions.

Now, you might be thinking, "Do I need to constantly log into Prediction Studio to stay informed?"

The answer is absolutely not.

Pega has designed a smart solution: daily email summaries that highlight high-impact notifications.

This means you can maintain awareness of your system's health without being tethered to the interface, allowing you to focus your attention where it's truly needed.

To activate and configure the email summaries, go to the Monitor and notifications setting in Prediction Studio.

When it comes to the Performance category notifications, let me highlight the ones that should be on every data scientist's radar.

Watch for sudden drops in predictive performance - these often signal data drift or environmental changes affecting your models.

Pay attention to models showing consistently low performance, as they may need retraining or feature engineering.

Monitor changes in lift or accuracy metrics, as these directly impact the business value your models deliver.

The beauty of this system lies in its flexibility. The specific notifications you prioritize will depend on your unique use case, business requirements, and risk tolerance.

Here's a common concern I hear: "Won't this create notification overload?"

Pega has anticipated this challenge.

Many notifications come with customizable thresholds that you can adjust based on your team's preferences and operational needs.

This threshold customization allows you to define precisely when you want to receive specific types of notifications, creating a personalized alert system that matches your team's workflow and urgency levels.

Let me put this all together in a practical framework.

When implementing a new Customer Decision Hub project, your data scientist team should approach notification configuration as a strategic activity, not just a technical setup task.

You're essentially creating a monitoring strategy that will alert your team to important issues impacting models and predictions before they affect business outcomes.

These notifications serve as your team's ability to promptly address critical matters, transforming reactive troubleshooting into proactive model management.

This shift from reactive to proactive monitoring can significantly improve both model performance and business results.

To maximize the value of Prediction Studio notifications, I recommend starting with conservative thresholds and gradually refining them based on your team's experience and business needs.

Remember, the goal is to ensure every notification you receive is actionable and valuable.

The time you invest in properly configuring these notifications will pay dividends in improved model performance, reduced downtime, and more confident decision-making across your Customer Decision Hub implementation.

That’s it from me. Until next time!

Change management process

Introduction

When business requirements change rapidly, the software development process needs to be more agile, while still producing high-quality, reliable software. This topic covers the importance of the change management process, and how it works in a one-to-one customer engagement project for Pega Cloud® customers.

Transcript

This video describes the change management process in a one-to-one customer engagement project.

Let’s start by understanding some background on change management in enterprise software development projects.

In recent decades, organizations have been using computer software to automate many traditionally manual tasks.

Business processes need to constantly evolve as customer behavior and market conditions change. Naturally, the software that supports business practices needs to evolve as well. As the pace of change in business requirements increases, the software development process needs to be more agile, while still producing high-quality and reliable software.

Here’s a simplified view of an enterprise software development cycle. It consists of four high-level stages.

Developers develop new software or update existing software.

The work from several developers is merged into a single system in the integration phase.

The new software version goes through testing, and the final, approved software (or a software change) is deployed into production.

This cycle repeats for new as well as incremental updates to existing software.

A software development process is supported by different environments.

- A development environment is one in which developers create new versions of the application by adding enhancements or fixing issues. It is referred to as system of record (SOR) for Pega applications.
- A staging environment is used for various testing such as functional testing, unit testing, and user acceptance testing.
- The business operations environment (BOE) is a replica of the production environment. However, it contains only a sample of the production data. This is where the business operations team creates and tests new business artifacts and conducts simulations.
- The production environment is the main system that propagates the next best actions to external channels, collects customer responses, and is where the AI learning happens. It is also used for live monitoring of key performance indicators.

In a one-to-one customer engagement project, changes to the application can be classified into two categories: enterprise changes and business changes.

Enterprise changes are the changes that developers make to the Pega application. An example of enterprise changes are extensions to the core Pega application and its integration points with external systems.

Developers make these changes in the development environment. Changes to the application are pushed to other environments through the enterprise-change pipeline managed by the Pega Deployment Manager™.

Business changes are small, day-to-day updates made as part of normal business operations. Examples of business changes include creating a new action or updating an existing action with new treatments or engagement policies. Also, this environment is used to carry out various simulations and analyses, for example, to test if there is an ethical bias in the decisions made by the next-best-action strategy framework. Business changes are made by the business operations team in the business operations environment.

The business content team uses the 1:1 Operations Manager portal to initiate changes in the business operations environment.

In the BOE, the changes are carried out in an application overlay. The application overlay defines the scope in which business users can change the application (for example, by modifying actions, treatments, decision strategies, and so on) to accommodate the constantly changing business conditions and requirements.

Changes from the business operations environment are pushed to the development environment and from there to other environments through the business change pipeline.

You have reached the end of this video which showed you:

- The importance of the change management process in an enterprise software development project.
- The high-level software development cycle.
- The cloud environments provided by Pega for a one-to-one customer engagement project using Pega Customer Decision Hub.
- The flow of enterprise and business changes through the enterprise and business change pipelines.

Adaptive models

Description

Online, adaptive models are crucial to next-best-action decision strategies in Pega Customer Decision Hub™. Adaptive Decision Manager is a key component of Pega Decision Management and helps organizations deliver tailored experiences that balance customer relevance and business priorities.

Discover the crucial role of input fields as predictor data in enhancing the predictive performance of adaptive models. Adaptive models automatically optimize by selecting the best subset of predictors from various relevant data sources.

Learn how to safely monitor a new adaptive model configuration by shadowing an active model.

Learning objectives

- Describe how ADM generates and updates Bayesian and Gradient boosting models.
- Explain how adaptive analytics supports Customer Decision Hub in the selection of the next best action for each customer.
- Explain how arbitration works.
- Request a decision for a customer in Customer Profile Viewer.
- Explore the original model propensity and model evidence.
- Shadow an adaptive model to safely monitor a new adaptive model configuration.

Adaptive Decision Manager

Online, adaptive models are central to Pega's next-best-action decision strategies, predicting customers' propensity to accept available actions. These models drive the delivery of highly personalized, relevant actions to each customer, enabling brands to achieve true 1:1 Customer Engagement in Pega Customer Decision Hub™

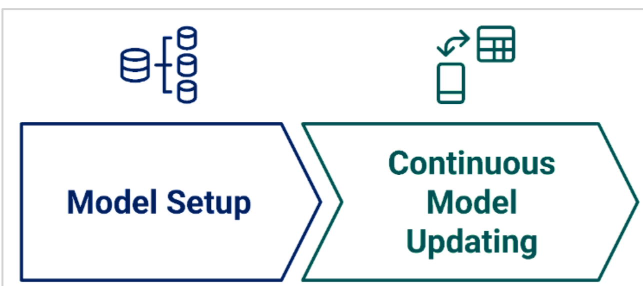
Pega's Adaptive Decision Manager, accessible through Prediction Studio, provides data scientists with a comprehensive toolset for creating, training, and managing self-learning models."

Transcript

Welcome! In this video, we'll dive into Pega's Adaptive Decision Manager. This is the component of Pega Decision Management that predicts customer behavior, for example the probability that a customer will click a web banner on a self-service channel.

Adaptive Decision Manager offers two modeling techniques: Bayesian and Gradient boosting. Both techniques are online learners, which means they continuously learn from new outcomes.

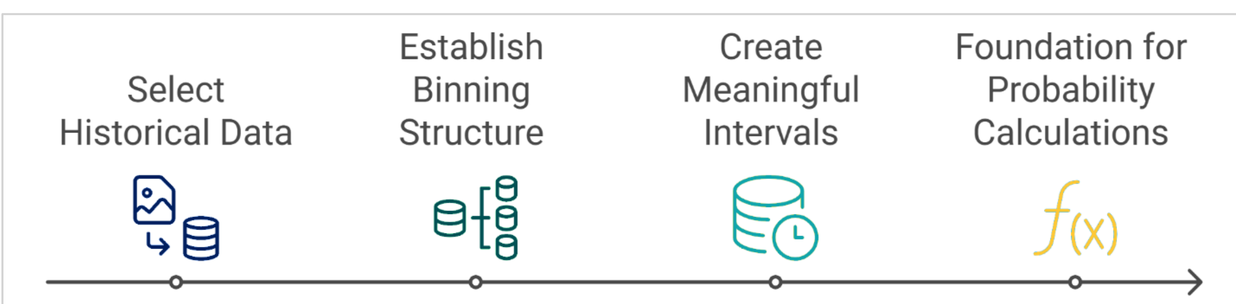
The Naïve Bayes modeling technique in Adaptive Decision Manager consists of two main steps: model setup and continuous model updating.



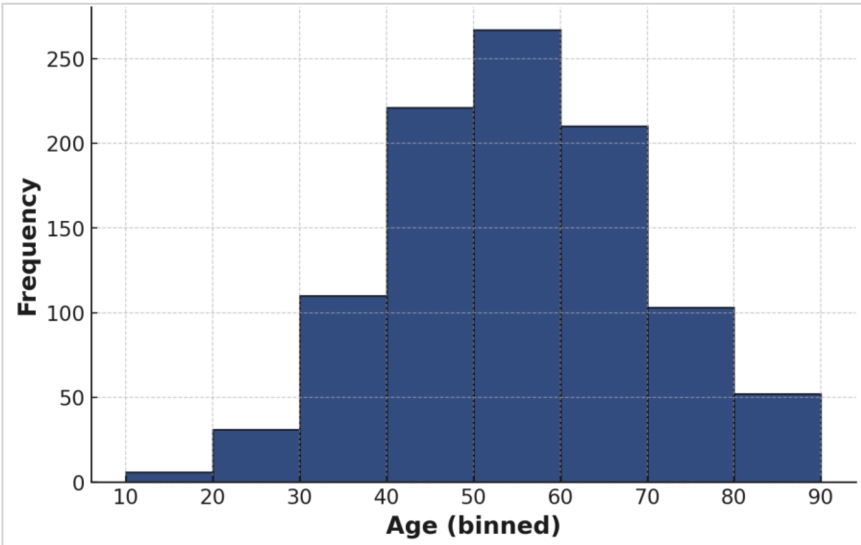
In the model setup phase, you select the potential predictors, preferably from various sources, including customer profile and behavioral data.

You then deploy the model to production.

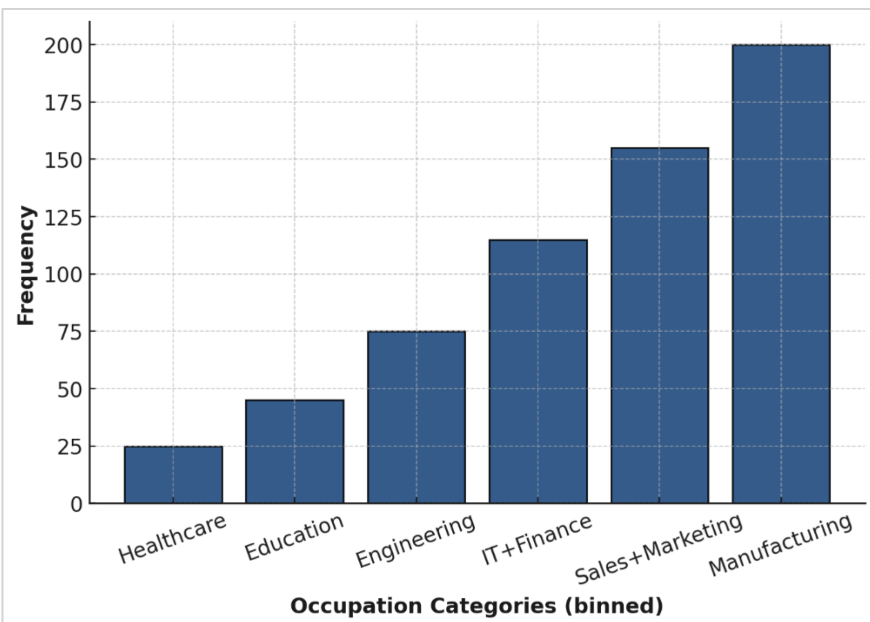
Each update of the model starts with the algorithm taking a batch of recent historical data and using it to establish the binning structure. This crucial step groups similar behaviors together, creating meaningful intervals that lower complexity, and serve as the foundation for the probability calculations.



For numeric predictors, such as age or income, Adaptive Decision Manager identifies intervals where customer behavior remains consistent. Customers aged 42 and aged 43 may have similar propensities to show target behavior and, after binning, reside in the same interval.



For symbolic predictors, such as product categories or customer segments, it combines categorical values that exhibit similar response patterns.



Feature selection follows preprocessing, focusing on identifying the most relevant predictors for the model. This step evaluates each predictor's individual correlation with the outcome using the area under the ROC curve, or AUC. Predictors must demonstrate significant correlation, exceeding a threshold of 0.52 AUC, to be included in the model.

Predictors | Score distribution | Trend

Correlated predictors are grouped and the best performing predictors become active in the model.

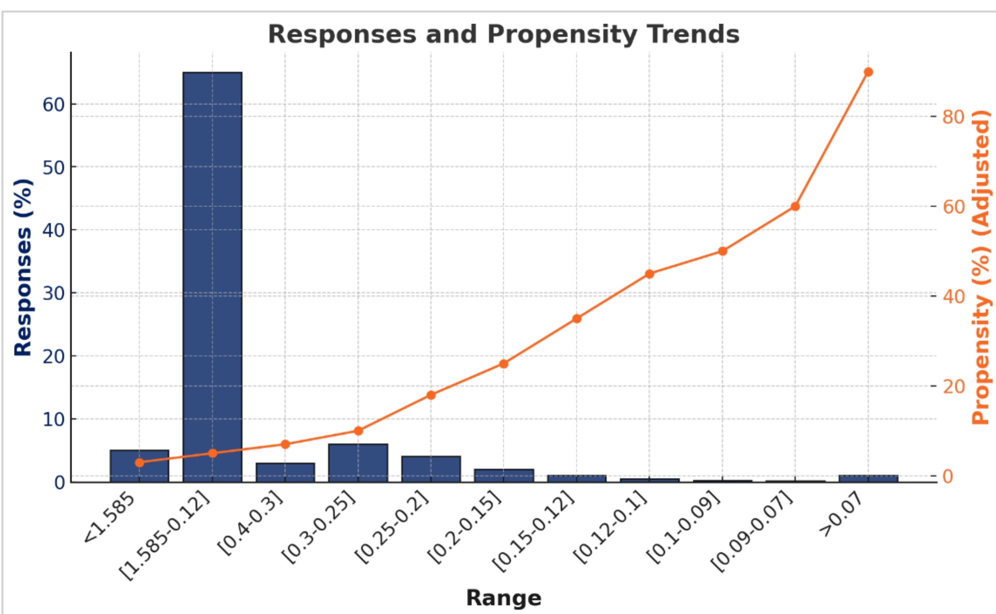
Predictors	Status	Type	Performance (AUC)	Range/Symbols(#)	Bins(#)
Customer.CreditScore	Active	Numeric	75.15	[0.0; 999.0]	7
Customer.AnnualIncome	Active	Numeric	73.36	[6000.0; 14998.0]	9
Customer.TotalAssets	Active	Numeric	55.02	[129.0; 999874.0]	8
Customer.NetWealth	Active	Numeric	54.98	[26.0; 99980.0]	9
Customer.NumDepositAccount	Active	Numeric	54.51	[0.0; 9.0]	10
Customer.MKTCLVValue	Active	Numeric	54.06	[0.0; 999.0]	6
Customer.NumLoanAccount	Active	Numeric	53.86	[0.0; 9.0]	9
Customer.DebtToIncomeRatio	Active	Numeric	53.24	[0.01; 48.0]	6
Customer.Prefix	Active	Symbolic	52.18	5	4
Customer.NumInvestmentAccount	Active	Numeric	52.12	[0.0; 9.0]	8
▾ Customer.IsCustomerActive	Inactive	Symbolic	51.85	3	3
Customer.IsProspect	Inactive	Symbolic	51.85	3	3
Customer.CLV	Inactive	Symbolic	51.68	3	3

Additionally, the system identifies groups of highly correlated predictors and selects the best representative from each group, supporting the independence assumption fundamental to Naïve Bayes modeling.

The core model building process then updates the probability counts that form the mathematical foundation of the model and are stored as the model's core parameters.

The final postprocessing step transforms the Naïve Bayes posterior probability to a propensity.

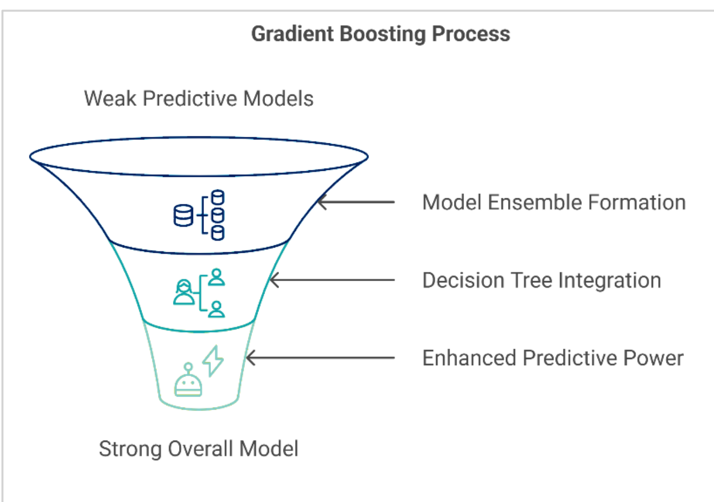
In this calibration step, the algorithm creates score intervals in such a way that the propensity for each next bin always increases to optimize the accuracy of the models.



The next-best-action strategy then uses the updated model to drive the delivery of highly personalized, relevant actions to each customer.

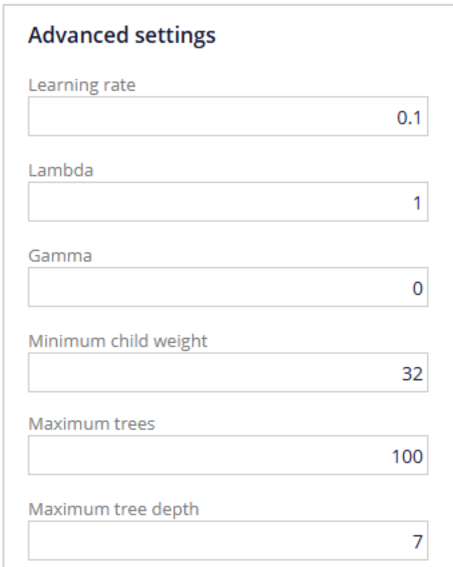
Adaptive Decision Manager incrementally updates the probabilities based on observed outcomes, allowing it to adapt to emerging patterns in customer behavior.

The second option, Gradient boosting, is a powerful machine learning technique that builds an ensemble of weak predictive models, decision trees, to create a strong overall model. It uses adaptive sliding windows to detect changes in customer behavior and determines when to grow and when to prune, eliminating the need to explicitly choose the adaptiveness.

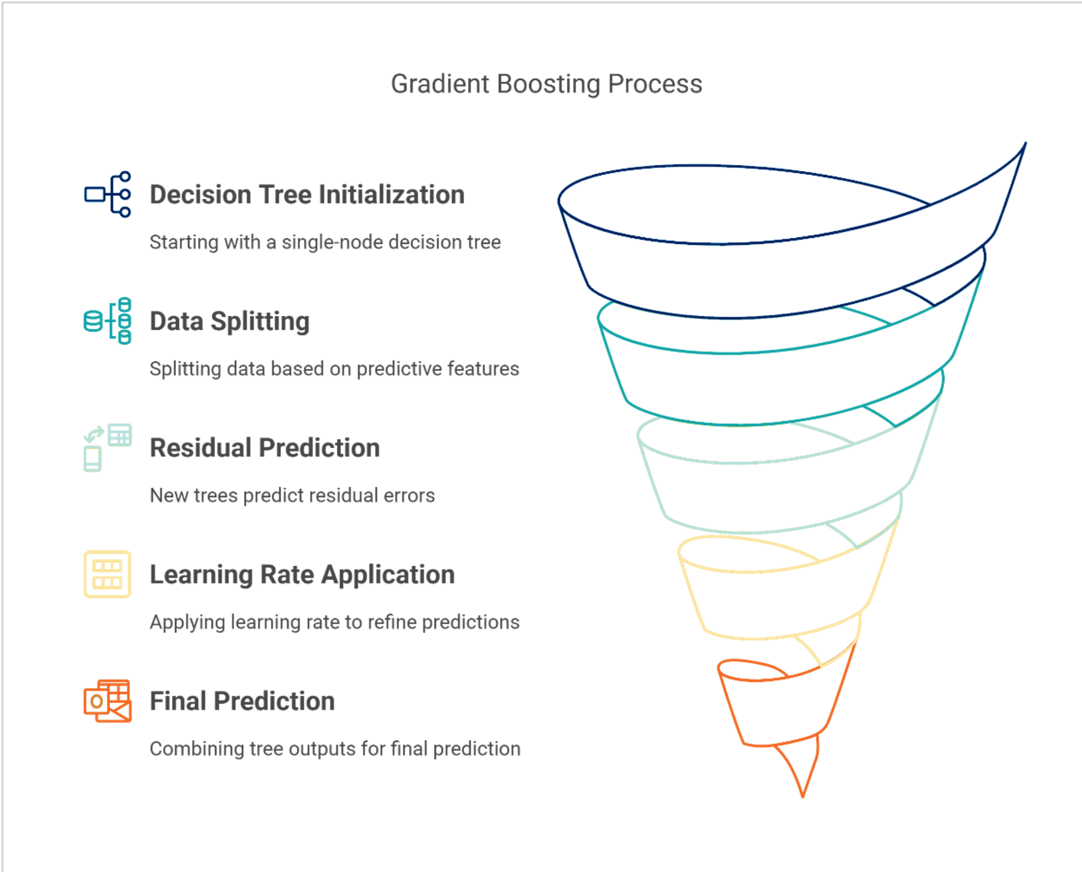


The process begins with a single-node decision tree, and based on the data arriving from the data stream this is extended to more trees and branches within those trees. The decision tree splits the data into branches based on features that provide the most improvement of the predictive power. The best splits are determined by comparing how much information is gained by making a split, compared to not making it. The trees adapt, as branches are pruned, and other branches grow as new data is received.

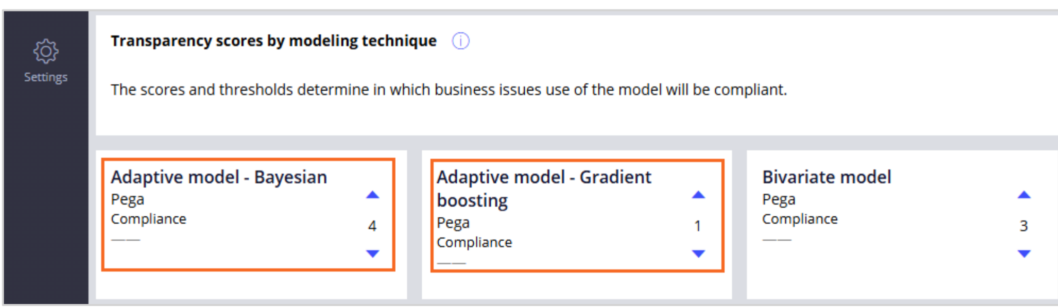
The difference between the predictions and the actual values coming in, known as residuals, represents the error. Next, new decision trees are trained to predict these residual errors to correct the mistakes made by the previous trees and refine the overall prediction. A parameter called the learning rate controls how much each new tree contributes to the final prediction.



A maximum to the number of trees ensures that the Gradient boosting model does not become unnecessarily complex. The final propensity score is obtained by adding the scores of all trees, multiplied by the learning rate.



The higher predictive accuracy of a Gradient boosting model comes with a trade-off in transparency due to the high complexity of the model.



Although Gradient boosting models are harder to interpret, you can gain valuable insights from the predictor importance, which is the impact of an individual predictor on the model score.

That wraps up my presentation on Adaptive Decision Manager. Thanks for joining and until next time!

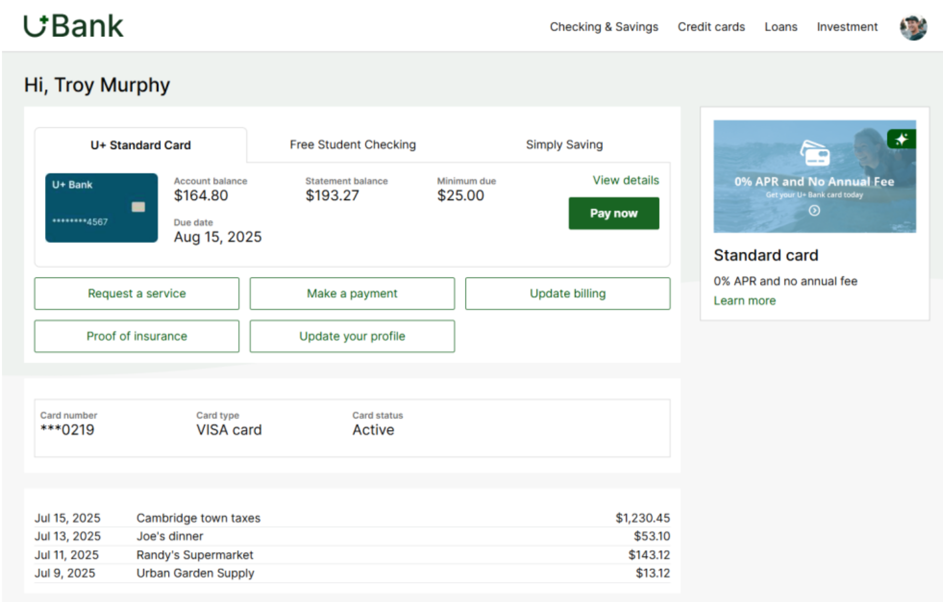
Adding predictors to adaptive models

U+ Bank's uses Customer Decision Hub™ for the credit card cross-selling on the web implementation. Predictor enhancement empowers the data science team to continuously improve the adaptive models' effectiveness.

Financial Services clickstream data captures real-time customer website interactions. By combining traditional demographic and transactional data with real-time behavioral patterns, you create a more comprehensive foundation for personalized decision-making that drives measurable business results.

Transcript

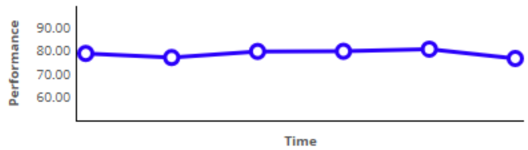
Hi! I'm Iris, and today I'll guide you through enhancing adaptive models in Pega Customer Decision Hub by adding new behavioral predictors. We'll work with U+ Bank's credit card cross-selling implementation, where I'll show you how to incorporate real-time clickstream data to improve prediction accuracy.



As a data scientist working with adaptive models, you'll often need to enhance existing predictions with new data sources. Today's focus is on adding Financial Services clickstream summaries that capture customer website behavior patterns, which can significantly improve your model's ability to predict customer engagement.

Let me start by showing you our prediction setup. In Prediction Studio, we're working with the “Predict Web Propensity prediction, which calculates the likelihood that customers will click on the credit card offers.

Predict Web Propensity 77.04 (AUC)



Decision Request CDH Jul 15, 2025

Open prediction

This prediction considers offer information, including the name, the inbound or outbound direction, and the channel and treatment.

Prediction properties

Prediction name

Predict Web Propensity

Outcome

Propensity to Click

Prediction output

Predicting Propensity to Click for each combination of

	.pyIssue
and	.pyGroup
and	.pyName
and	.pyDirection
and	.pyChannel
and	.pyTreatment

The prediction configuration includes several critical settings. The response labels define "Clicked" as our target behavior and "NoResponse" as the alternative outcome.

Response labels

Labels for the possible values of the responses that the Prediction records. Response labels represent the Outcomes of the Prediction and are used for monitoring and model learning.

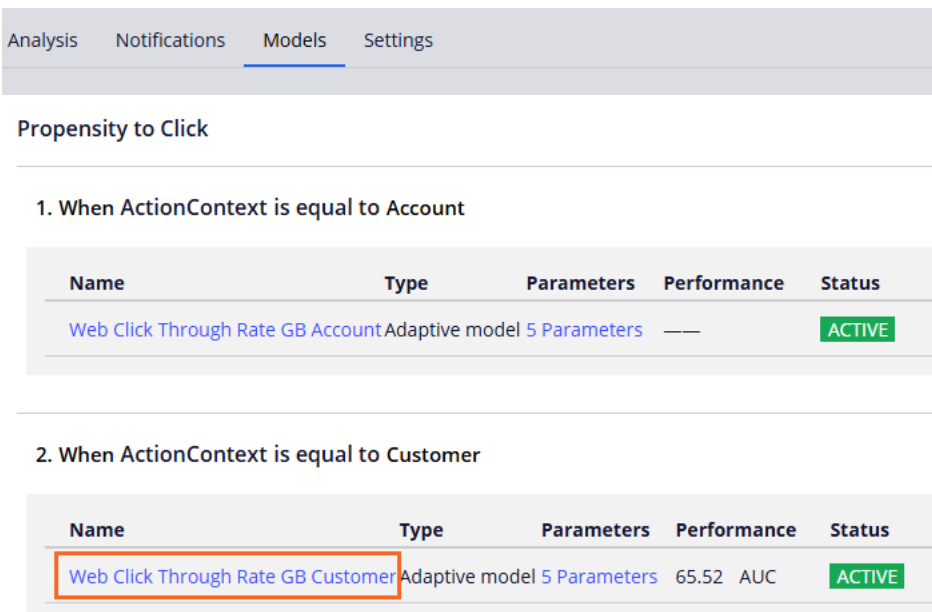
Propensity to Click

Target label	Alternative label
Clicked	NoResponse

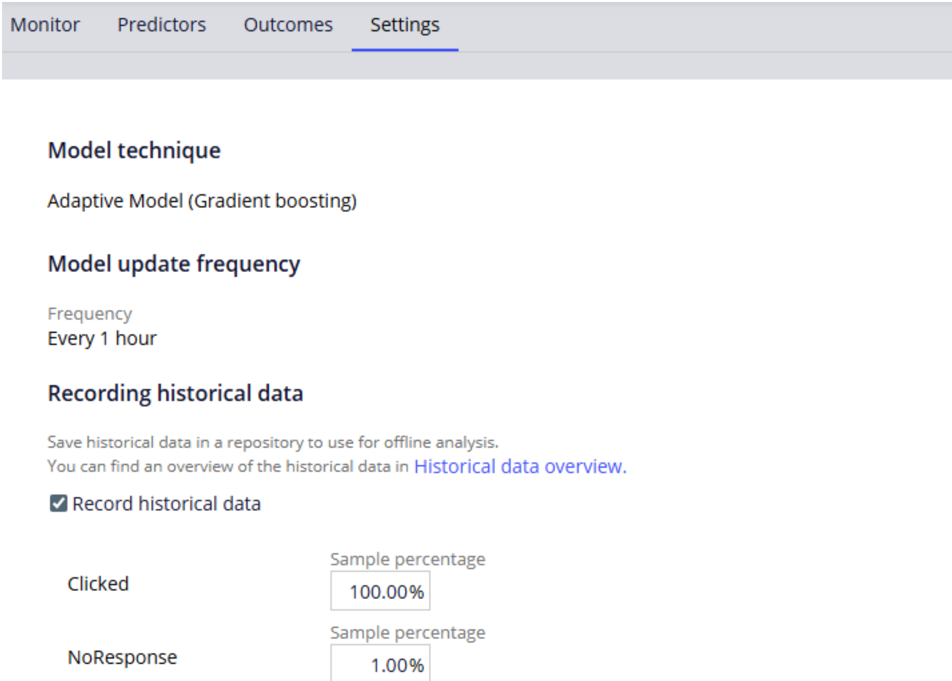
The response timeout is set to 30 minutes, meaning if customers do not interact with an offer within that time period, the system automatically records a NoResponse. This timing begins when the Next-Best-

Action decision is made. Alternatively, you can start the waiting period when the Action is presented to the customer in your channel.

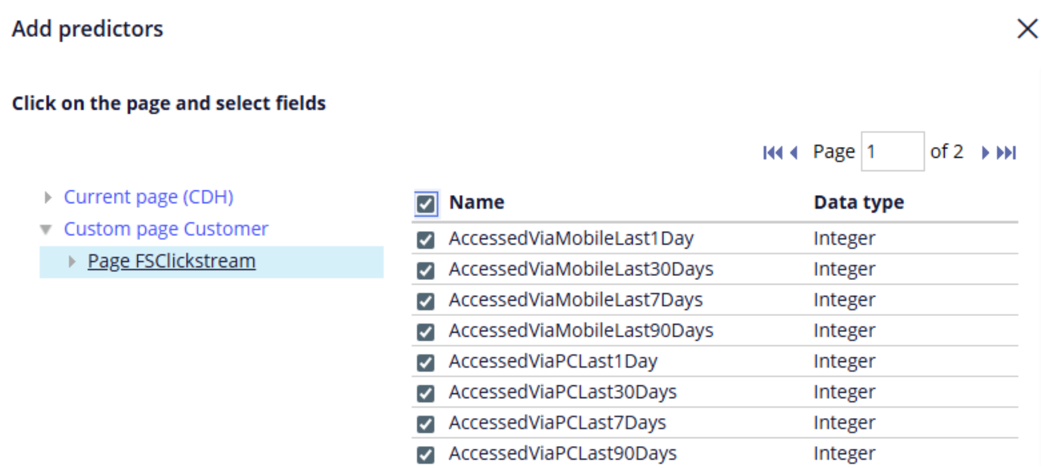
The underlying "Web Click Through Rate GB Customer" model uses gradient boosting as its machine learning technique at the customer level.



The model updates hourly, ensuring rapid learning from new customer interactions. With historical data recording enabled, the system captures predictor values and outcomes for every decision, creating comprehensive learning datasets.

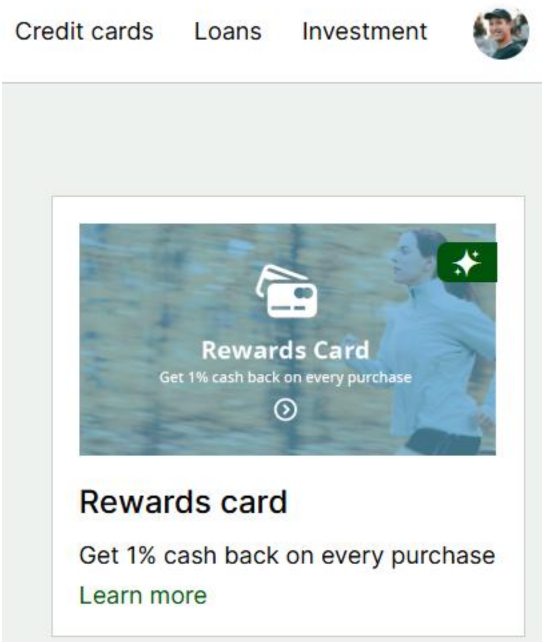


Now let's enhance our model with rich behavioral data. The key is accessing the FSClickstream page, which contains numerous behavioral predictors that provide insights into how customers interact with the website. By selecting all these fields, we're provide the model with comprehensive clickstream data that reveals customer intent and engagement patterns.

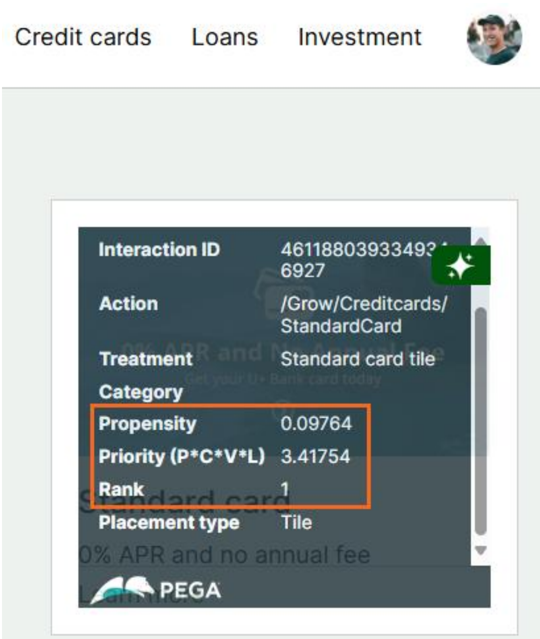


These behavioral predictors complement traditional demographic and transactional data, resulting in a more complete picture of customer behavior. The Financial Services clickstream summaries are particularly valuable because recent web browsing information often correlates strongly with purchase intent.

To see our enhanced model in action, I'll demonstrate the customer experience on the U+ Bank website. When customers like Troy log in, they encounter personalized credit card offers based on our adaptive model's calculations.



Troy is eligible for both the Standard Card and the Rewards Card, and the system displays the most relevant offer based on current propensity scores.



Customer interactions create valuable training data. When Troy clicks "Learn more" within the thirty-minute window, the system records both an Impression outcome when the offer displays and a Clicked outcome when he engages. If he ignores the offer, only the Impression outcome is recorded, followed by a NoResponse after the timeout period. Each interaction feeds back into the model learning process.

The system captures not just the final decision, but also the behavioral context leading up to that decision through the clickstream data we added. For example, when a customer spends some time on the loans page, this may be very relevant for determining the optimal banner content.

The Model Management landing page provides comprehensive monitoring capabilities. In the Latest responses section, you can observe both target and alternative outcomes as they occur.

Latest responses

Time received	Tue Jul 22 09:08:01 GMT 2025
Interaction ID	4611880393349346927
Outcome	NoResponse

Time received	Tue Jul 22 09:07:01 GMT 2025
Interaction ID	4611880393349346927
Outcome	Impression

Time received	Tue Jul 22 07:58:41 GMT 2025
Interaction ID	4611880393349346923
Outcome	NoResponse

Time received	Tue Jul 22 07:57:41 GMT 2025
Interaction ID	4611880393349346923
Outcome	Impression

Time received	Mon Jul 21 12:40:11 GMT 2025
Interaction ID	4611880393349346922
Outcome	NoResponse

Clicked offers show both Impression and Clicked outcomes, while ignored offers display Impression followed by NoResponse outcomes. The model details reveal how many responses have been recorded since the last update, providing insight into the learning velocity and data accumulation.

Model details

Response details

Total responses	17177
Positive responses	4733
Negative responses	12444
Recorded responses	1

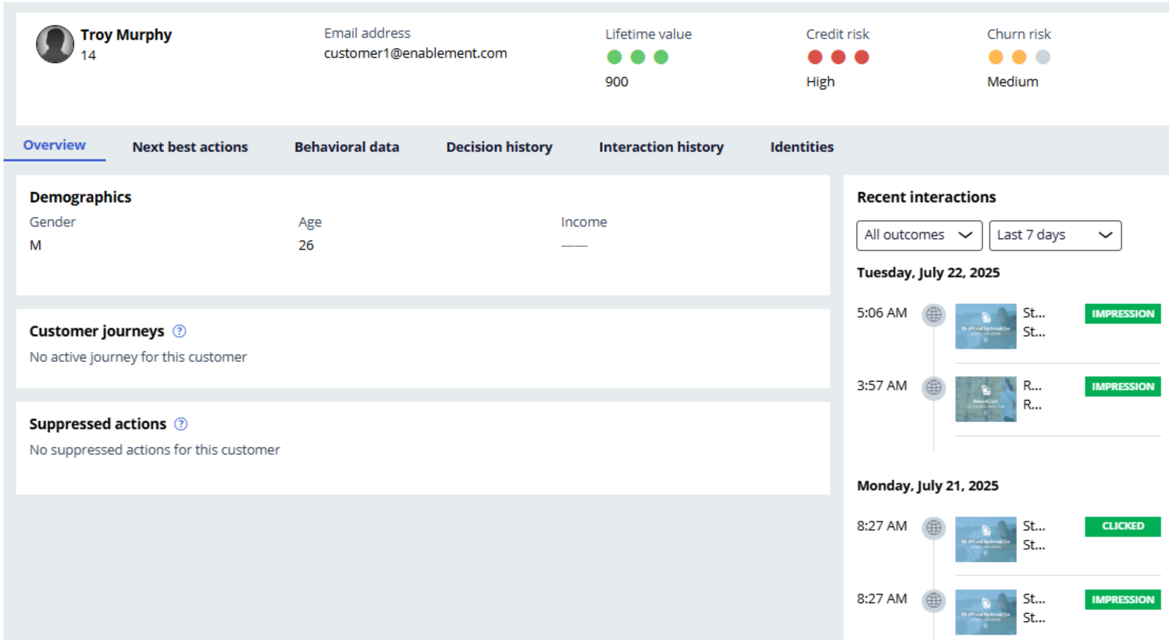
Update schedule

Last update	45 minutes ago
Update frequency	1 hour

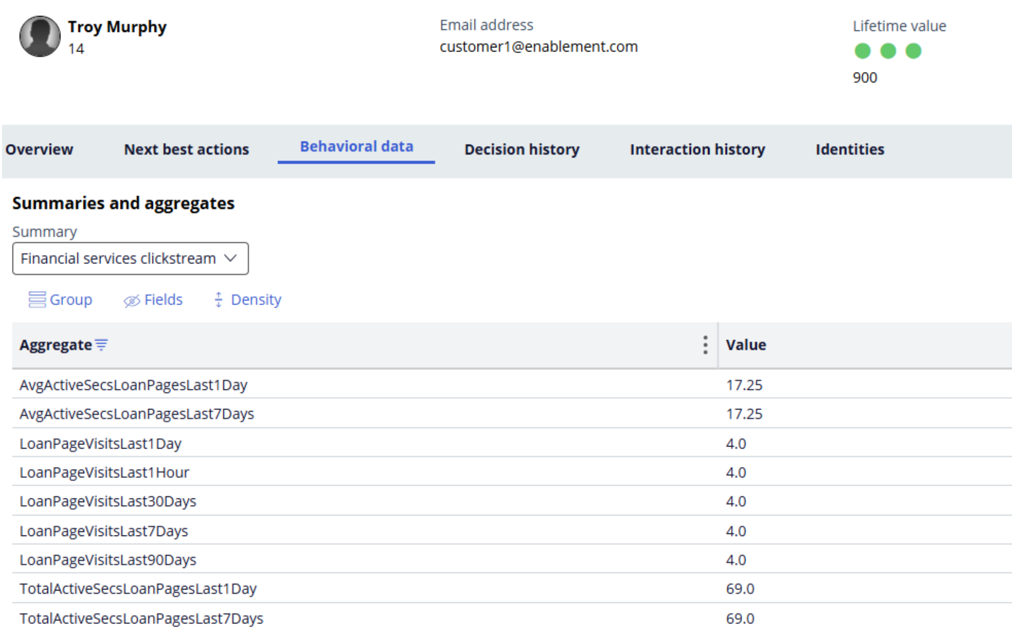
Model context

Model	Web_Click_Through_Rate_GB_Customer

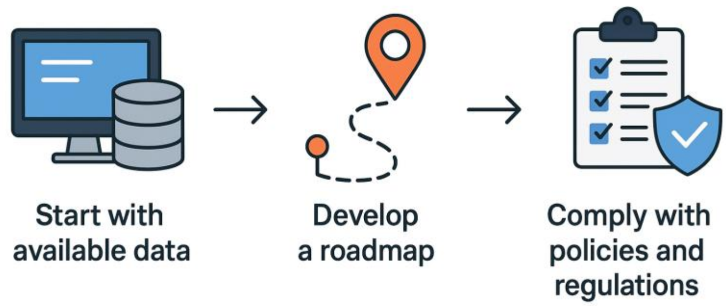
The Customer Profile Viewer offers detailed insights into individual customer journeys. For Troy Murphy, we can examine his complete interaction history, behavioral data summaries, and decision history.



The Financial Services Clickstream section confirms that the system captures comprehensive website navigation patterns, for example Troy's visits to the Loans page.



By integrating FSClickstream predictors, we've enhanced our model's understanding of customer behavior patterns, leading to improved credit card offer targeting and higher engagement rates. The combination of real-time behavioral data with traditional customer profile information creates a comprehensive foundation for personalized decision-making. The key to successful predictor enhancement is starting with data that is already available in your application.



Then, develop a roadmap to gradually incorporate additional data sources, and ensure that you always comply with internal policies and regulatory requirements. This systematic approach to predictor enhancement will improve your adaptive models' accuracy and effectiveness over time.

That’s it for today.

See you next time.

Monitoring adaptive models

Description

It is a regular data scientist task to inspect the health of the out-of-the-box Pega Customer Decision Hub™ predictions and the adaptive models that drive them, and share the findings with the business team. The predictive performance and success rate of individual adaptive models provide information that can help business users and decisioning architects to refine business processes. Learn how to monitor the performance of predictions, adaptive models, and predictors.

Learning objectives

- Describe the lift metrics of a predictions
- Name the key metrics of adaptive models visualized in the bubble chart
- Inspect individual active and inactive predictors
- Explain how predictors with similar predictive performance are grouped
- Examine the propensity distribution and the trend for the whole model
- Use the Prediction Studio APIs

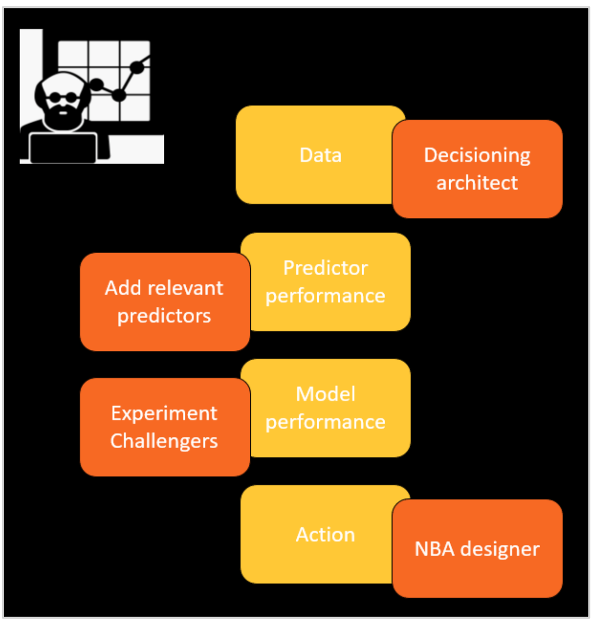
Monitoring predictions

In the production phase of a project, in response to Prediction Studio notifications or as a regular health check, a data scientist analyzes the predictions, models and predictors with the aim of providing insights to the governance team and to fix any issues with prediction performance.

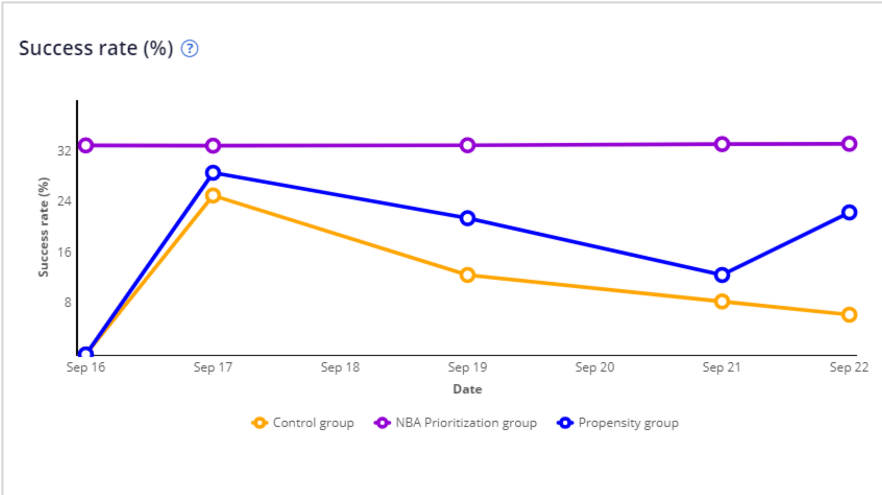
Transcript

This video shows you how to monitor predictions, models, and predictors in the production phase of a project.

In response to the notifications, a data scientist logs in to Prediction Studio in the production environment or in the business operations environment, examines the relevant charts, and checks the suggestions in the notifications. If it is a data quality issue, you refer the work to a decisioning architect. If it is a predictor performance issue, you add more relevant predictors. If it is a model performance issue, you run experiments and introduce challenger models through the MLOps process. If the action, not the model or the predictors, is the issue, you refer the work to the NBA designer team.

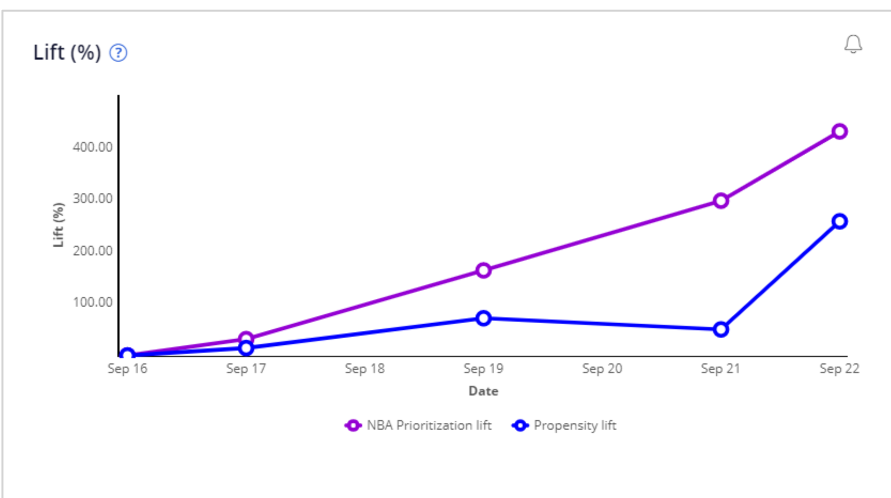


For the analysis of predictions, several charts are available. As an example, U+Bank uses the *Predict Web Propensity* prediction to calculate the likelihood that a customer will click on a web banner that offers one of the credit cards for which the customer is eligible. Success rate measures how successful the credit card offers are. The success rate is defined as the number of times the credit card offers are clicked, divided by the number of times they are offered.



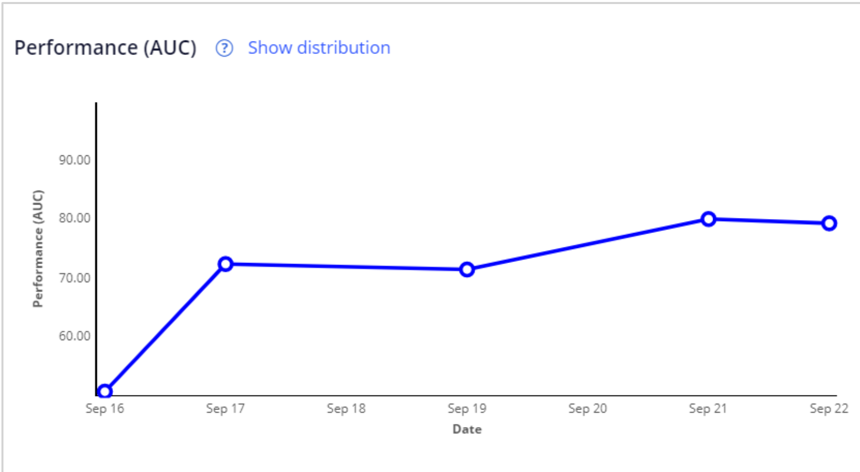
This chart shows the data for offers based on the performance of the Next Best Action strategy, with the majority of the customers in the *NBA Prioritization group*. A small percentage of the customers, who are the *Control group*, receive a random relevant offer. Another small percentage of the customers, who are the *Propensity group*, receive an offer based only on the model propensity with context weight, value and levers disabled. The *Control group* and the *Propensity group* are set to small percentages of the population in *Impact Analyzer*. A prediction needs attention when the success rate decreases over time, as this indicates a problem with the underlying models.

Lift is the impact on engagement that the AI generates. The lift metric can tell us how much better predictions made by a model are when compared to a series of random predictions for customers in the control group.



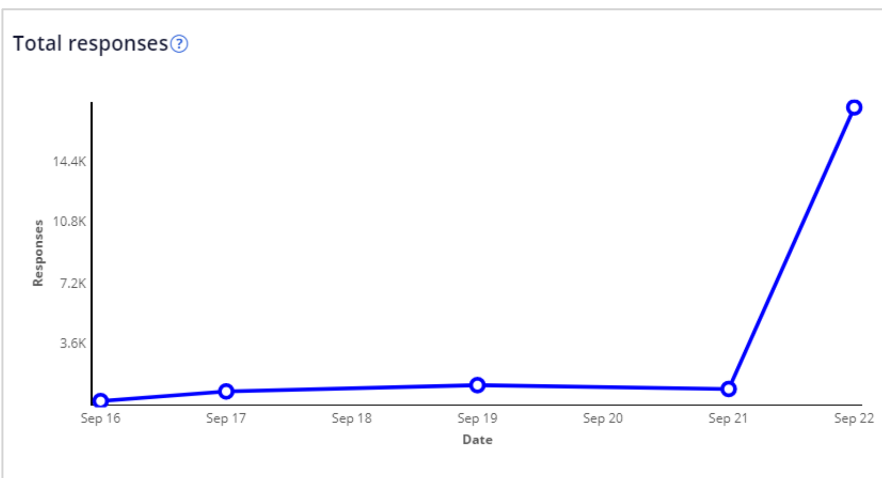
NBA Prioritization lift is the difference in the success rate of the *NBA Prioritization group* over the *Control group*. Similarly, *Propensity lift* compares the *Propensity group* to the *Control group*. The prediction needs attention when the lift decreases over time. The lift may be low if the model is still immature, or if the control group is not representative enough of the actual population.

Performance measures the accuracy of a prediction in predicting an outcome, and ranges from 50 to 100.



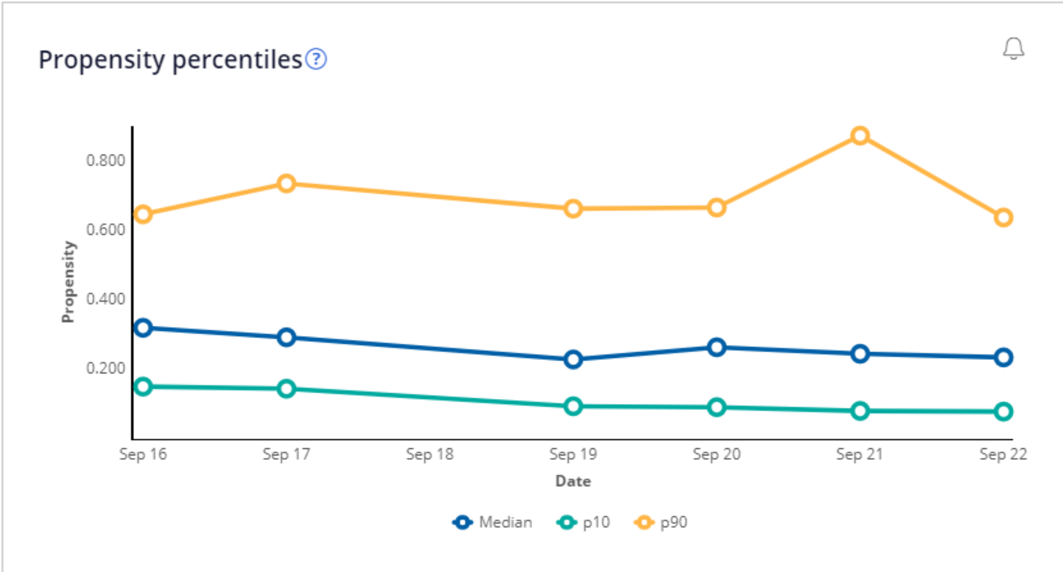
The prediction needs attention when the performance decreases over time. This may indicate issues with the predictor set of the models.

Total responses measures the number of responses the models receive to base their output on.



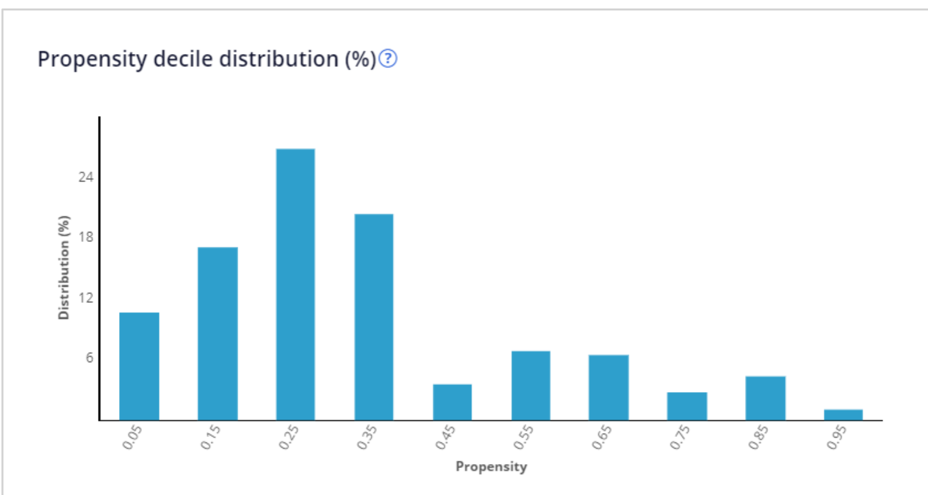
The prediction needs attention when the number of responses received is zero for a week or more. This may have multiple causes and needs to be looked at by the Decisioning Architect.

The Propensity percentiles chart shows the propensity values for the lowest 10%, for the highest 90% and for the median.



This metric needs attention when there is a drastic change in percentile values, especially the median, compared to the previous week. Percentile values can drastically change if the models are experiencing data drift, or when the models receive too many missing values, resulting in percentile drift.

Propensity decile distribution shows the frequency of a certain propensity in a range from 0 (no likelihood), to 1 (certainty).



For inbound use cases, the propensities peak at higher values than outbound use cases and continue reducing over the higher value deciles, indicating that models are more confident in making inbound predictions than outbound predictions. The prediction needs attention when the propensity is very similar across the deciles or moving up towards the higher deciles. This may indicate overfitting of the model towards the data.

Model analysis offers listings of the best and worst performing adaptive models.

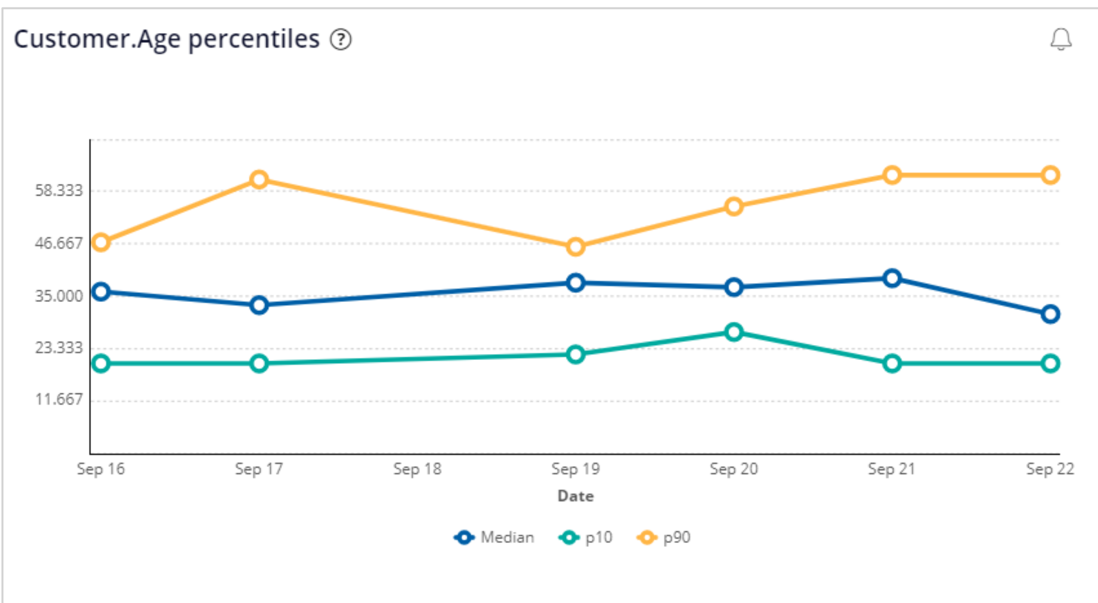
Best performing models

Issue	Group	Action	Direction	Channel	Performance (AUC)
Grow	Creditcards	{"pyName":"PremierRewardsCard","pyTreatment":"Premier Rewards card tile"}	Inbound	Web	83.981
Grow	Creditcards	{"pyName":"RewardsPlusCard","pyTreatment":"Rewards Plus card tile"}	Inbound	Web	75.271
Grow	Creditcards	{"pyName":"RewardsCard","pyTreatment":"Rewards card tile"}	Inbound	Web	63.768
Grow	Creditcards	{"pyName":"StandardCard","pyTreatment":"Standard card tile"}	Inbound	Web	50

Worst performing models

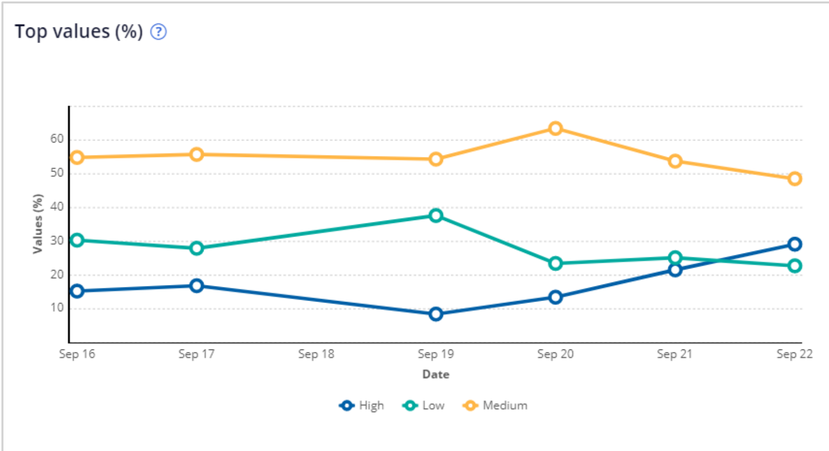
Issue	Group	Action	Direction	Channel	Performance (AUC)
Grow	Creditcards	{"pyName":"StandardCard","pyTreatment":"Standard card tile"}	Inbound	Web	50
Grow	Creditcards	{"pyName":"RewardsCard","pyTreatment":"Rewards card tile"}	Inbound	Web	63.768
Grow	Creditcards	{"pyName":"RewardsPlusCard","pyTreatment":"Rewards Plus card tile"}	Inbound	Web	75.271
Grow	Creditcards	{"pyName":"PremierRewardsCard","pyTreatment":"Premier Rewards card tile"}	Inbound	Web	83.981

The percentiles for the values of the Age predictor give an idea of the range of values for this numerical predictor.



The predictor needs attention when there is a drastic change in percentile values, especially the median, compared to the previous week, as the models experience data drift, or receive too many missing values.

This chart shows the trend of the most frequent values of the symbolic predictor CLV:



The predictor needs attention when a trend line drastically changes, which indicates an underlying change in the data that may cause data drift or concept drift.

This demo has concluded. What did it show you?

- How to monitor Customer Decision Hub predictions.

Monitoring Gradient boosting models

Introduction

Gradient boosting models represent a significant advancement in Pega's decisioning framework, offering superior predictive accuracy compared to traditional approaches. The predictive performance of the model and the success rate of individual actions provide information that can help business users and decisioning consultants to refine the Next-Best-Actions.

Monitoring the health of a Gradient boosting model and its predictors is a regular task of a data scientist that can be performed in Prediction Studio in the production environment or the business operations environment. This monitoring process allows organizations to maintain optimal model performance and ensure reliable decision-making capabilities.

Transcript

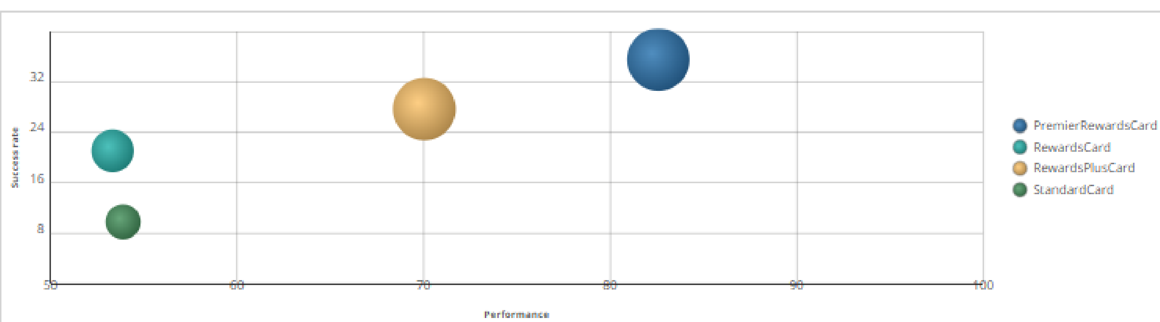
Hi, I'm Iris, and today I'll show you how to inspect the health of a Gradient boosting model and its predictors in Prediction Studio.

As part of Customer Decision Hub™, the Predict Web Propensity prediction calculates the likelihood that a customer clicks on a web banner to optimize customer engagement.

In this example, the model calculates the propensity that a customer will respond positively to a credit card offer.

It uses the Web Click Through Rate Gradient boosting model configuration to determine the outcomes at the customer level.

The Monitor Models tab shows an intuitive bubble chart that serves as your primary dashboard for assessing model health and identifying optimization opportunities.



The bubble chart visualizes the key metrics of the model at the action-treatment combination level.

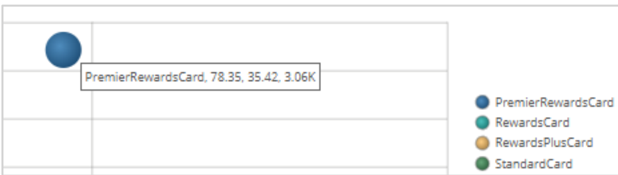
Each bubble represents the metrics for a specific offer, in this example a specific credit card in the Web channel, with the size of the bubble reflecting the number of responses, both positive and negative, that have been used in the adaptive learning process.

The Performance axis shows the model performance expressed in the Area Under the Curve, which ranges from 50 to 100; values above 70 generally indicate good predictive performance, with values above 80 representing excellent model accuracy.

The Success rate axis indicates the success rate of the proposition as a percentage, calculated by dividing the number of positive responses by the total number of responses.

This information can help business users and decisioning consultants to refine the Next-Best-Actions.

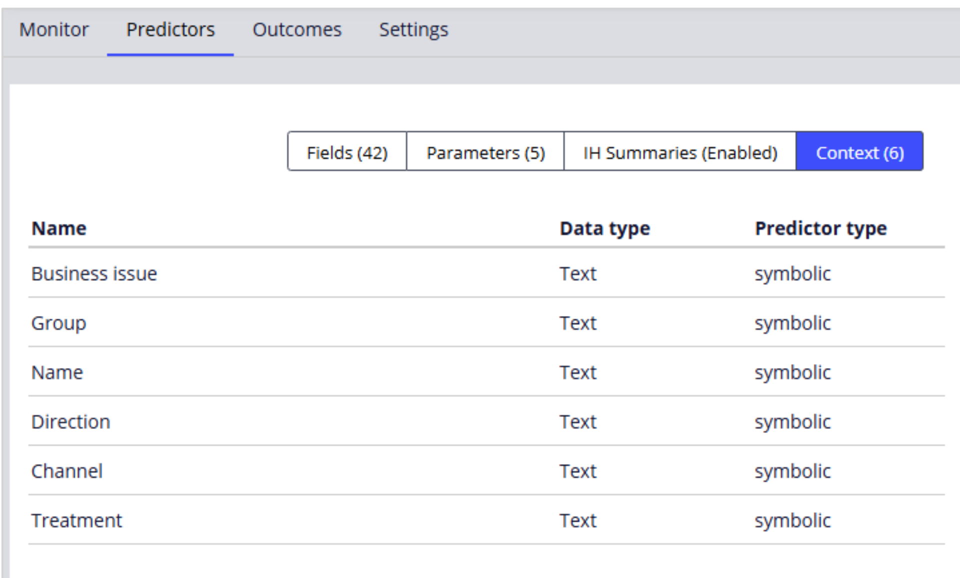
When you hover the cursor over a bubble, you can view the action name, performance metrics, success rate, and response volume.



The information displayed is extracted from the Adaptive data mart, which is generated automatically by background processes that create snapshots at regular intervals of the Adaptive Decision Management server.

This keeps your monitoring data current and reliable.

The model Context predictors includes channel and direction, enabling the single Gradient boosting model to make contextual splits in its decision trees based on properties such as channel and direction.


Monitor Predictors Outcomes Settings

Fields (42) Parameters (5) IH Summaries (Enabled) Context (6)

Name	Data type	Predictor type
Business issue	Text	symbolic
Group	Text	symbolic
Name	Text	symbolic
Direction	Text	symbolic
Channel	Text	symbolic
Treatment	Text	symbolic

From the adaptive model interface, you can drill down into detailed model reports for individual propositions.

The Predictor Importance view shows the relative contribution of each predictor to the model performance as a percentage, providing crucial insights for model optimization.

Predictor importance | Score distribution | Trend

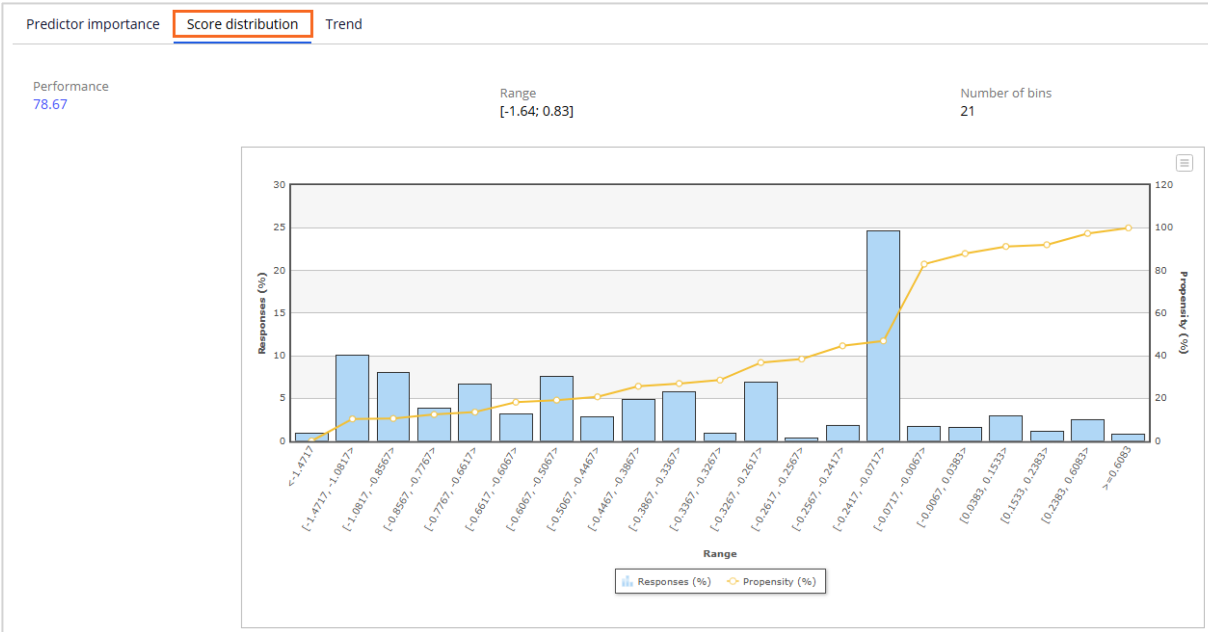
Predictor importance indicates the relative importance of each feature when making a prediction

Predictors	Status	Type	Importance
Customer.AnnualIncome	Active	Numeric	28.47
Customer.CreditScore	Active	Numeric	27.74
Customer.BirthDate	Active	Numeric	18.73
Customer.RelationshipLengthDays	Active	Numeric	14.16
Customer.PrimaryPostalCode	Active	Symbolic	2.87
Customer.ResidentialStatus	Active	Symbolic	2.53
Customer.PrimaryCountryCode	Active	Symbolic	1.41
Customer.OwnershipStatus	Active	Symbolic	1.37
Customer.NumDepositAccount	Active	Numeric	1.12
Customer.OrganizationID	Active	Symbolic	0.94
Customer.PrimaryCity	Active	Symbolic	0.54
Customer.Prefix	Active	Symbolic	0.13
Customer.CLV	Inactive	Symbolic	0.00
Customer.DebtToIncomeRatio	Inactive	Numeric	0.00

For example, Annual Income and Credit Score show a high importance score, and strongly contribute to the model's decision-making process.

Inactive predictors such as CLV and Debt To Income Ratio, have an importance score of zero. Gradient boosting models determine predictor usage through their algorithmic decision tree-building process. This information helps data scientists identify which features may need re-evaluation or removal from the model.

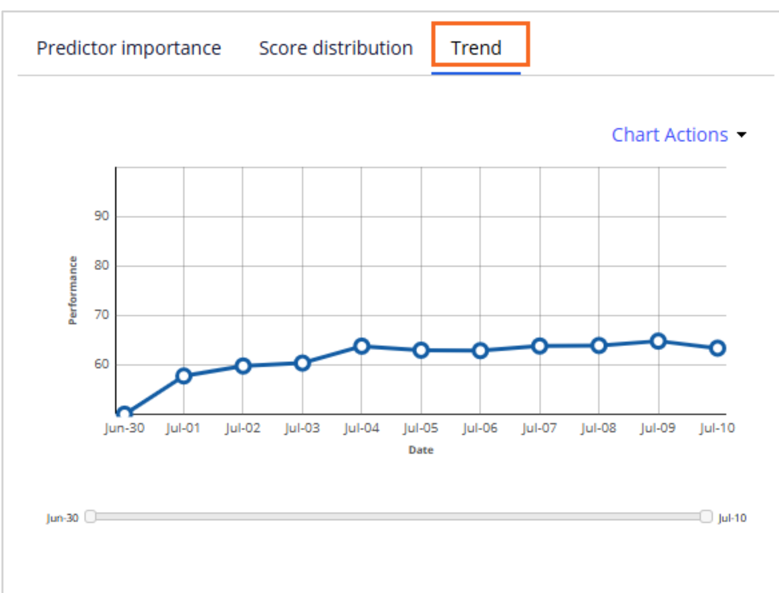
The Score Distribution report enables data scientists to validate their model's predictive accuracy and understand how scores are distributed across different customer segments.



The histogram shows response frequencies and propensity percentages across different score ranges, with the data table providing detailed breakdowns of responses, positives, negatives, and lift metrics for each range.

This granular analysis helps identify optimal score thresholds for decision-making.

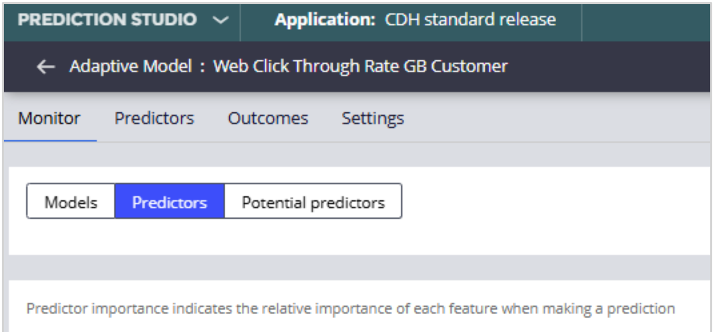
The trend report visualizes model performance over time, showing how the Gradient boosting model adapts and maintains its predictive accuracy.



This temporal analysis is crucial for identifying performance degradation or improvement patterns.

Regular monitoring of these trends enables proactive model maintenance and optimization.

On the Monitor Predictors tab, you can examine the importance scores that show how each feature contributes to the overall Gradient boosting model performance.



The importance scores reflect how frequently and effectively each predictor is used across the entire ensemble of decision trees.

This detailed predictor analysis enables data scientists to understand which customer attributes drive model decisions, supporting both model transparency and regulatory compliance requirements.

That wraps up this video on the regular monitoring of your Gradient boosting models, ensuring they continue to deliver optimal Next-Best-Actions for your customers.

Until next time!

Testing the Prediction Studio APIs

Pega provides APIs that allow you to connect to Prediction Studio from your preferred data science environment. These APIs provide endpoints for various operations, such as accessing prediction attributes, introducing challenger models, exporting data, and more. A System Administrator generates the required artifacts and credentials.

Video

@@@

Transcript

Pega provides APIs that allow you to connect to Prediction Studio from your preferred data science environment. These APIs provide endpoints for many Prediction Studio operations.

To provide data scientists with the option to work from their preferred Python environment, a System Administrator generates the artifacts required to export ADM snapshots and creates the necessary credentials to access the APIs.

To generate the data sets and data flows required for the export of ADM data, you switch to the settings of Prediction Studio and configure the Monitoring database export. Select the Ruleset for the export artifacts and submit to complete the configuration.

Back in Dev Studio, create new client credentials for access to the Prediction Studio APIs. You create an OAuth 2.0 client registration data instance. After creating the client registration, ensure to save the credentials that include the Client ID and Client Secret.

To inspect the API responses, you open the API landing page. From here, select the Prediction APIs service package and authorize the client by entering the saved Client ID and Client Secret.

The tasks are divided into categories like Data Export, Models, Reporting, Predictions, and Prediction Studio. Explore the available models by using the GET List models API to retrieve a list of models within the application.

To delve deeper into a specific model, such as the Web Click Through Rate Customer adaptive model, use the GET Describe Model API with the appropriate model ID to obtain detailed information, including the modeling technique, target labels, and predictors. This model calculates the propensity that a customer clicks on a web banner.

To validate the generated artifact required for the export of ADM snapshots, you use the POST Trigger Datamart Export API. In the repository, the model snapshots and predictor snapshots are stored in the datamart folder. The APIs are now ready for use by the data science team.

Exporting adaptive model data

Description

The reporting datamart of Pega Adaptive Decision Manager (ADM) is an open data model. As a result, data scientists that work on Pega Customer Decision Hub™ projects can export adaptive model data, predictor binning data, and historical data (predictor and outcome values) for further analysis. Learn how to configure data flows to export the model data of interest.

Learning objectives

- Export the historical data from a Customer Decision Hub system.
- Configure data flows to export selected adaptive model data and predictor data.

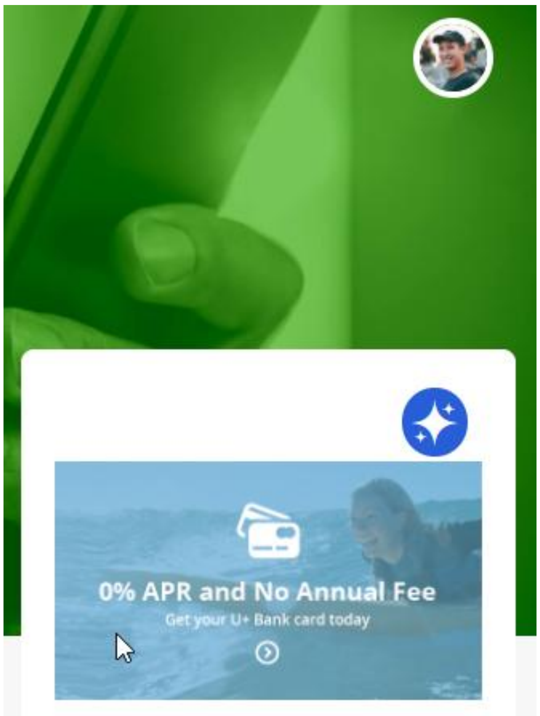
Exporting historical data

Learn how to extract historical data (predictors and outcomes) from adaptive models in your application to perform offline analysis or use the data to build models using the machine learning service of your choice.

Transcript

This demo shows you how to export the customer interaction data that is used by adaptive models to make predictions, including all predictor data and associated outcomes, for offline analysis.

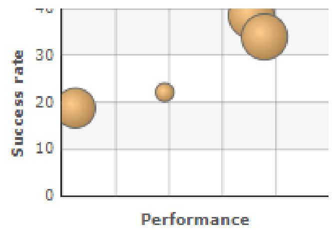
U+ Bank has implemented Pega Customer Decision Hub™ to show a personalized banner on their website that advertises credit card offers.



When a customer is eligible for multiple credit cards, adaptive models decide which card to show. When the customer ignores the banner, the adaptive model that drives the decision regards this as negative behavior. When a customer clicks on the banner, the model regards this as a positive behavior.

As a data scientist, you may want to inspect the raw predictor data used by an adaptive model and the customer interaction outcome to validate data assumptions and check for concept drift. You can also use the data to build various predictive models externally.

All models are managed in Prediction Studio. The adaptive model that drives the decision over which banner to display is the **Web Click Through Rate** model.



Web_Click_Through_Rate

To extract the data, you enable the recording of historical data for a selected adaptive model. A web banner typically has a low click-through rate and a significantly lower number of positive responses than negative responses. In such cases, you can sample all positive outcomes and just one percent of the negative outcomes to limit the storage space needed.

Recording historical data

Save historical data in a repository to use for offline analysis.
You can find an overview of the historical data in
Historical data overview.

☑ Record historical data

	Sample percentage
Clicked	100.00%
NoResponse, Impression	1.00%

The sample percentages determine the likelihood that a customer response is recorded. The system stores the predictor data and outcome as a JSON file in a repository of your choice. By default, the data is stored for 30 days in the **defaultstore** repository. However, this repository points to a temporary directory, and a system architect should switch to a resilient repository to avoid data loss. Supported repository types include Microsoft Azure and Amazon S3.

Select repository type

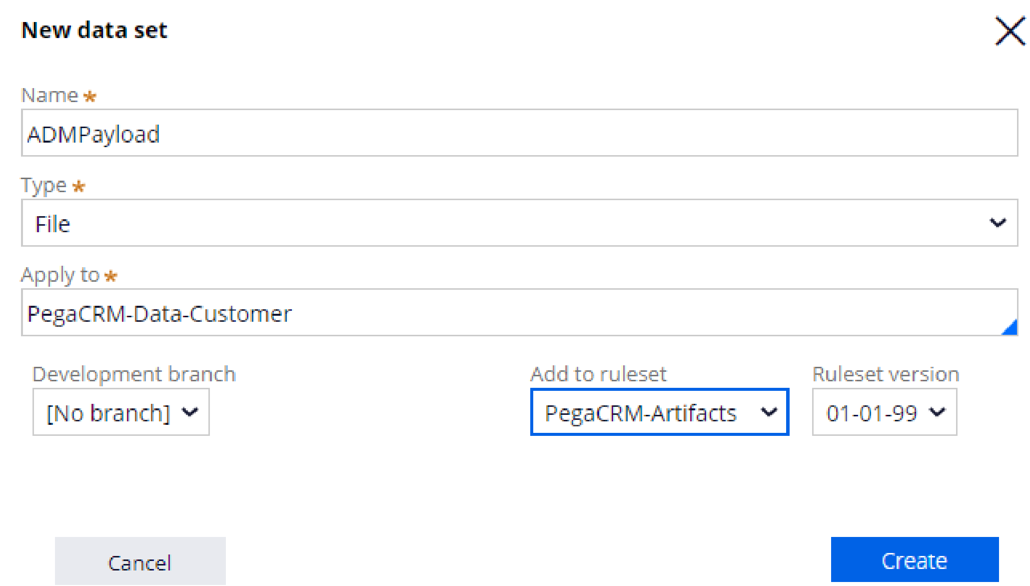
Artifactory

S3

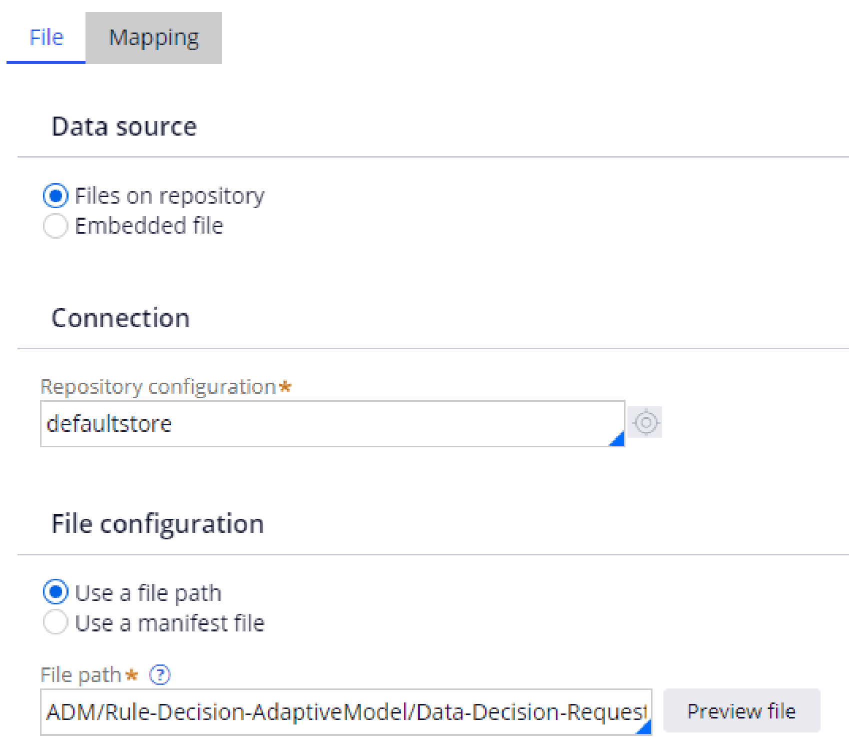
File system

Azure

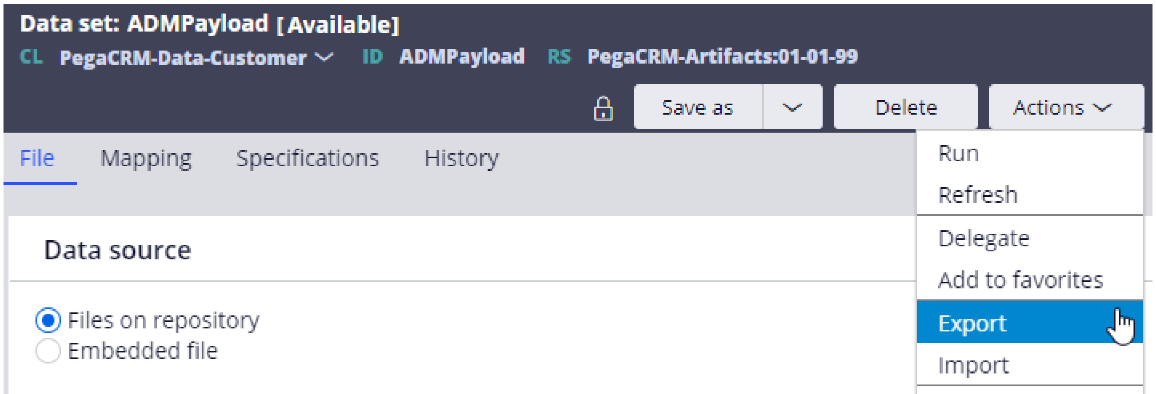
For this demo, we use the default store repository and create a data set to export the data.



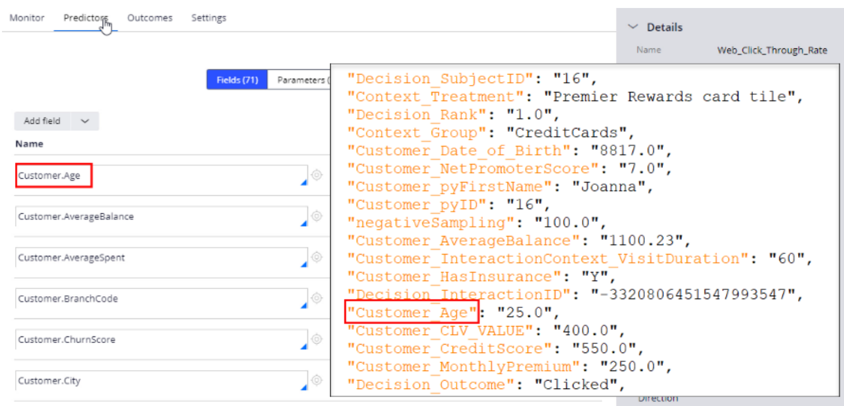
The data set is mapped to the file that contains the recorded historical data.



With these settings in place, the input data used for the prediction and associated outcome are stored in the configured data set when customers see an offer and click on an offer. The system architect can download the data set in DEV Studio.



Every record contains the predictor values used for the prediction, as well as the context and the decision properties, including the outcome of the interaction. All property names are automatically converted to comply with the JSON format.



To use the JSON file for further analysis, import the file into a third-party analytics tool. Keep in mind that when many customers visit the website, the file size becomes very large in a short time. To limit the storage space needed, you can lower the sample percentages.

You have reached the end of this demo. What did it show you?

- How to export the raw data that is used by adaptive models.
- What data is captured during a customer interaction.

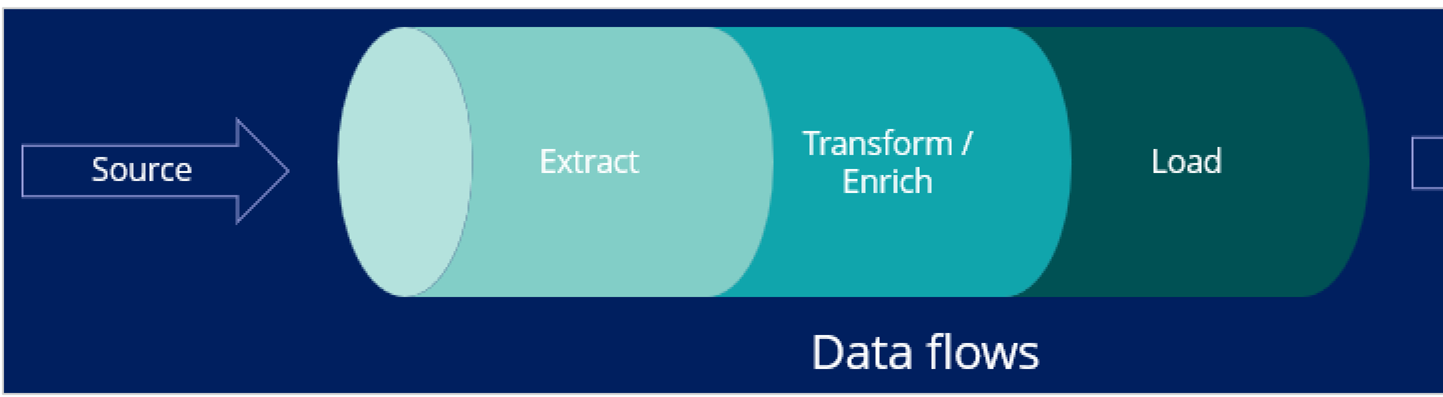
Understanding and implementing Data Flows

Use Data Flows to process and move data between data sources. Reference business Rules to do complex data operations.

Transcript

Data Flows are pipelines that receive data from an input source and output data to one or more destinations. Within a Data Flow, you can transform or enrich the data, for example, by joining with data from other sources. They are crucial to any Pega implementation and are optimized for performance to handle large quantities of data.

In the context of Pega Customer Decision Hub™, they play a critical role and factor in various key activities, such as ingesting customer data, exporting customer interaction details, and orchestrating inbound/outbound next-best-actions. For example, in a customer service scenario, a Data Flow can retrieve audio transcripts from a Pega Voice AI service™. This Data Flow runs on a single thread, retrieves audio transcripts from the Pega server, and processes the data.



Data Flows consist of various components that transform and enrich the data in the pipeline. The components run concurrently to handle the data, starting from the source and moving towards the destination. A Data Flow can use another Data Flow for its source or destination, create Cases, trigger strategies, and more.

There are three types of Data Flows:

- Batch Data Flows (Batch processing)
- Real-time Data Flows (Real-time processing)
- Single Case Data Flows (Single Case processing)

Batch Data Flows process a finite number of records from their source, such as a database table, report definition, or file. You can schedule them to run at specific times, with a given frequency, or on demand. When the Data Flow run begins, it accesses and reads the data in its source, processes it, and finally sends it to the destination. A typical example is ingesting customer records from a CSV file, manipulating the data, and populating the Customer Insights Cache in

Customer Decision Hub or running a 1:1 Next Best Action outbound schedule for a given audience.

Real-time Data Flows process an infinite number of records from their source. They are always active and continuously process an incoming stream of data. For example, they can process data generated by a customer's web activity, where the system aggregates the stream of clicks and page navigation data into meaningful data summaries, or write all customer interactions to a repository as they occur.

Single Case Data Flows always have an abstract data source. They are used to process inbound data, for example, when the call center channel calls Pega Customer Decision Hub to determine the next best actions for a customer. The system runs a Single Case Data Flow to retrieve the customer's data and runs another to invoke the strategies that determine the best action for the customer.

You have reached the end of this video.

Exporting selected model data for external analysis

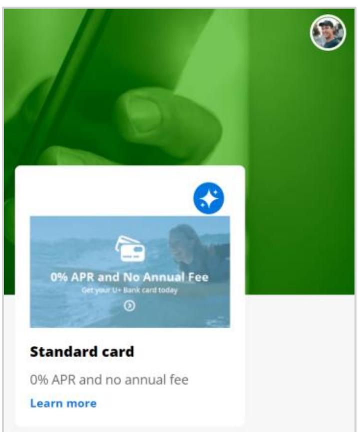
Data scientists can export snapshots of the adaptive models that drive Pega Customer Decision Hub™ predictions from the Adaptive Decision Manager (ADM) data mart for further analysis in their favorite analytical tools.

To limit the scope of the export to the data of interest, learn how to customize the export to populate data sets that contain only data that are relevant to you.

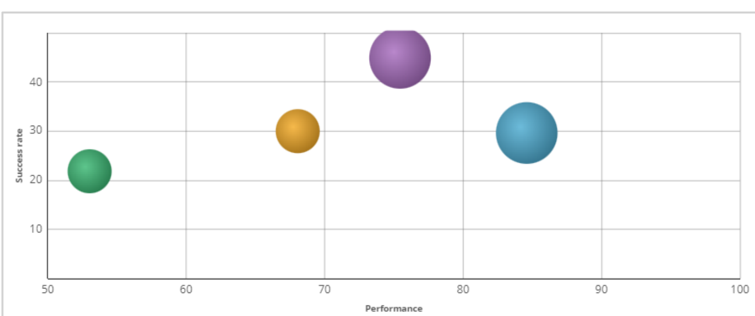
Transcript

This video shows you how to customize the export of adaptive model data for offline analysis in analytical tools such as Python and R. The publicly available GitHub repository Pega Data Scientist Tools helps to build meaningful plots and more with the exported data.

U+ Bank uses Customer Decision Hub to determine which credit card offer to show on their website when a customer logs in. For each offer, an adaptive model determines the likelihood that the customer will click on the web banner.



In Prediction Studio, Data Scientists continuously monitor the state of their predictions and the adaptive models that drive them.



For offline analysis of the adaptive model data, they can export the data from the Adaptive Decision Manager data mart. Two database tables contain the required monitoring information, and these tables populate two data sets.

The *ADM snapshot* data set contains snapshots that include the model ID, the model name, the configuration name, and model attributes such as the number of predictors, the model performance, and many others.

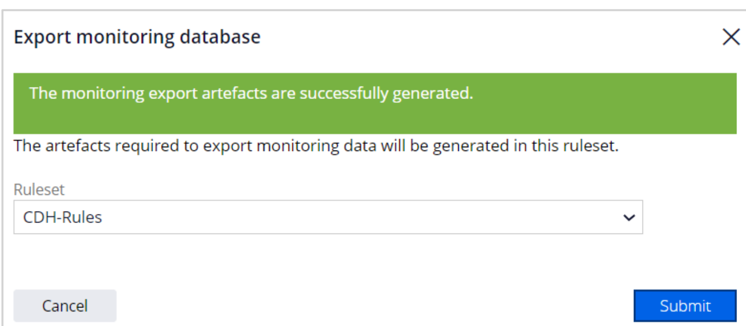
Snapshot Time	Model ID	Model Name	Configuration Name	Predictors	Performance	Etc.
20202212 11.49	5c2d8678-660c	Web Treatment A	Web Click Through Rate	153	0.74	
20202212 11.49	6h4s9812-546d	Email Treatment C	Email Click Through Rate	68	0.63	
20202212 11.49	7f3g3401-841a	Retail Treatment B	Retail Click Through Rate	203	0.69	
20202211 22.53	9g7r8688-550f	Web Treatment C	Web Click Through Rate	59	0.82	
20202211 22.53	6h4s9812-546d	Email Treatment A	Email Click Through Rate	134	0.62	
20202211 22.53	3f5g3401-841g	Retail Treatment A	Retail Click Through Rate	177	0.71	

The *ADM predictor* data set contains snapshots of the binning of individual predictors. The data sets have the Model ID key in common.

Snapshot Time	Model ID	Predictor Name	Predictor Type	Positives	Negatives	Etc.
20202212 11.49	5c2d8678-660c	Predictor A	Numeric	2371	4562	
20202212 11.49	6h4s9812-546d	Predictor G	Symbolic	3498	8921	
20202212 11.49	7f3g3401-841a	Predictor D	Symbolic	1232	9011	
20202211 22.53	9g7r8688-550f	Predictor H	Numeric	4581	7923	
20202211 22.53	6h4s9812-546d	Predictor B	Numeric	3092	6453	
20202211 22.53	3f5g3401-841g	Predictor C	Numeric	2089	5673	

Both tables can grow very large, but you typically need only the data for a selection of the models. For example, you may only be interested in the models for the application you are working on, or in just a particular channel. This demo shows you how to customize the export of the two data sets to your repository.

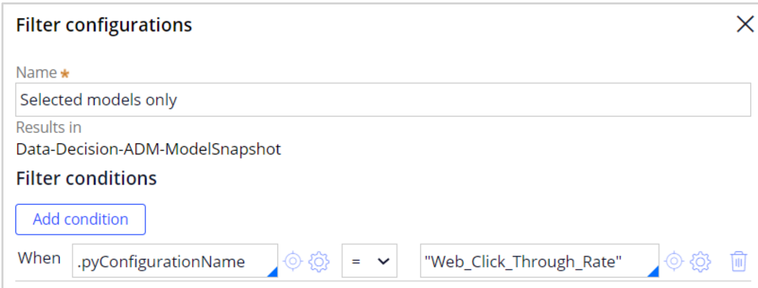
In the implementation phase of the project, you generate the artifacts required for the export of adaptive and predictive model snapshots, including the required data flows in Prediction Studio.



Data flows are scalable and resilient data pipelines that you can use to ingest, process, and move data to one or more destinations.

To limit the size of the exports, you may want to select only the data that you are interested in before exporting. For example, to select the data of models based on the *Web Click Through Rate* model configuration, adjust the two relevant data flows.

The *ADM snapshot data export* data flow exports the model snapshots to the repository. To configure the data flow to only export the relevant snapshots, add a filter component that only passes on snapshots on the condition that the model configuration name equals *Web Click Through Rate*.

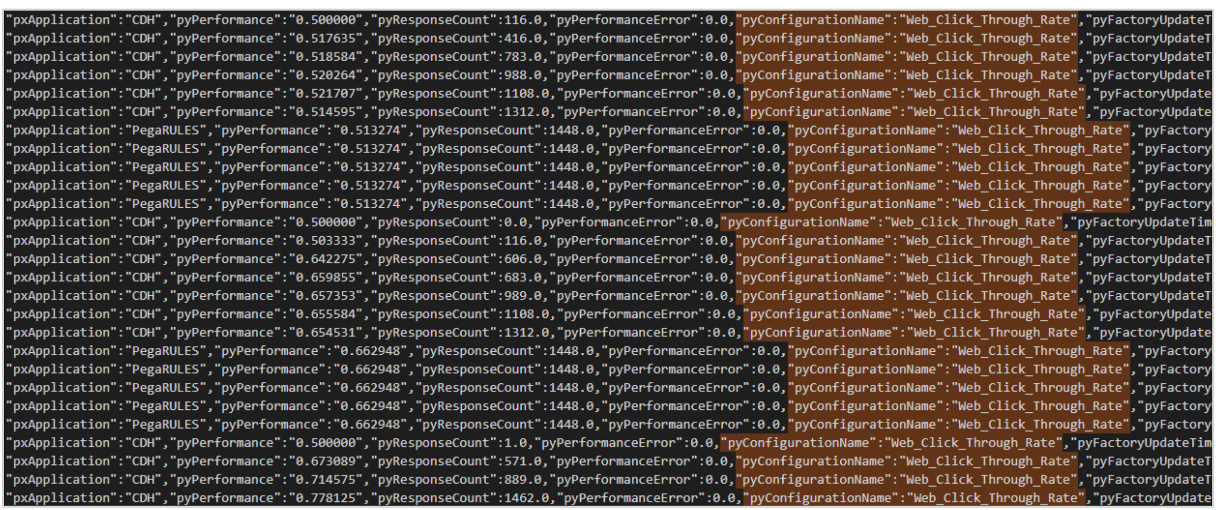


The *ADM predictor* data set does not contain the model configuration name, but it does contain the model ID. To select the predictor binning snapshots for the selected models, create a data set that only contains the selected model IDs. Add the new data set as a second destination to the *ADM snapshot data export* data flow.

The *ADM predictor data export* data flow exports the predictor binning snapshots. To filter only the relevant snapshots, merge the predictor binning snapshots and the data set that contains the selected model IDs that you created for this purpose.

Add a Convert component to match the two classes of the source data set, which is a prerequisite for the merge operation. The data flow merges the data sets on the condition that the model IDs match, and the destination data set only contains the relevant predictor binning snapshots.

The actual export of the monitoring data is typically done in the Business Operations Environment. The exported data in the repository contains only the model snapshots of interest.



```
"pxApplication":"CDH","pyPerformance":"0.500000","pyResponseCount":116.0,"pyPerformanceError":0.0,"pyConfigurationName":"Web_Click_Through_Rate","pyFactoryUpdateT
"pxApplication":"CDH","pyPerformance":"0.517635","pyResponseCount":416.0,"pyPerformanceError":0.0,"pyConfigurationName":"Web_Click_Through_Rate","pyFactoryUpdateT
"pxApplication":"CDH","pyPerformance":"0.518584","pyResponseCount":783.0,"pyPerformanceError":0.0,"pyConfigurationName":"Web_Click_Through_Rate","pyFactoryUpdateT
"pxApplication":"CDH","pyPerformance":"0.520264","pyResponseCount":988.0,"pyPerformanceError":0.0,"pyConfigurationName":"Web_Click_Through_Rate","pyFactoryUpdateT
"pxApplication":"CDH","pyPerformance":"0.521707","pyResponseCount":1108.0,"pyPerformanceError":0.0,"pyConfigurationName":"Web_Click_Through_Rate","pyFactoryUpdate
"pxApplication":"CDH","pyPerformance":"0.514595","pyResponseCount":1312.0,"pyPerformanceError":0.0,"pyConfigurationName":"Web_Click_Through_Rate","pyFactoryUpdate
"pxApplication":"PegaRULES","pyPerformance":"0.513274","pyResponseCount":1448.0,"pyPerformanceError":0.0,"pyConfigurationName":"Web_Click_Through_Rate","pyFactory
"pxApplication":"PegaRULES","pyPerformance":"0.513274","pyResponseCount":1448.0,"pyPerformanceError":0.0,"pyConfigurationName":"Web_Click_Through_Rate","pyFactory
"pxApplication":"PegaRULES","pyPerformance":"0.513274","pyResponseCount":1448.0,"pyPerformanceError":0.0,"pyConfigurationName":"Web_Click_Through_Rate","pyFactory
"pxApplication":"PegaRULES","pyPerformance":"0.513274","pyResponseCount":1448.0,"pyPerformanceError":0.0,"pyConfigurationName":"Web_Click_Through_Rate","pyFactory
"pxApplication":"PegaRULES","pyPerformance":"0.513274","pyResponseCount":1448.0,"pyPerformanceError":0.0,"pyConfigurationName":"Web_Click_Through_Rate","pyFactory
"pxApplication":"CDH","pyPerformance":"0.500000","pyResponseCount":0.0,"pyPerformanceError":0.0,"pyConfigurationName":"Web_Click_Through_Rate","pyFactoryUpdateTim
"pxApplication":"CDH","pyPerformance":"0.503333","pyResponseCount":116.0,"pyPerformanceError":0.0,"pyConfigurationName":"Web_Click_Through_Rate","pyFactoryUpdateT
"pxApplication":"CDH","pyPerformance":"0.642275","pyResponseCount":606.0,"pyPerformanceError":0.0,"pyConfigurationName":"Web_Click_Through_Rate","pyFactoryUpdateT
"pxApplication":"CDH","pyPerformance":"0.659855","pyResponseCount":683.0,"pyPerformanceError":0.0,"pyConfigurationName":"Web_Click_Through_Rate","pyFactoryUpdateT
"pxApplication":"CDH","pyPerformance":"0.657353","pyResponseCount":989.0,"pyPerformanceError":0.0,"pyConfigurationName":"Web_Click_Through_Rate","pyFactoryUpdateT
"pxApplication":"CDH","pyPerformance":"0.655584","pyResponseCount":1108.0,"pyPerformanceError":0.0,"pyConfigurationName":"Web_Click_Through_Rate","pyFactoryUpdate
"pxApplication":"CDH","pyPerformance":"0.654531","pyResponseCount":1312.0,"pyPerformanceError":0.0,"pyConfigurationName":"Web_Click_Through_Rate","pyFactoryUpdate
"pxApplication":"PegaRULES","pyPerformance":"0.662948","pyResponseCount":1448.0,"pyPerformanceError":0.0,"pyConfigurationName":"Web_Click_Through_Rate","pyFactory
"pxApplication":"PegaRULES","pyPerformance":"0.662948","pyResponseCount":1448.0,"pyPerformanceError":0.0,"pyConfigurationName":"Web_Click_Through_Rate","pyFactory
"pxApplication":"PegaRULES","pyPerformance":"0.662948","pyResponseCount":1448.0,"pyPerformanceError":0.0,"pyConfigurationName":"Web_Click_Through_Rate","pyFactory
"pxApplication":"PegaRULES","pyPerformance":"0.662948","pyResponseCount":1448.0,"pyPerformanceError":0.0,"pyConfigurationName":"Web_Click_Through_Rate","pyFactory
"pxApplication":"PegaRULES","pyPerformance":"0.662948","pyResponseCount":1448.0,"pyPerformanceError":0.0,"pyConfigurationName":"Web_Click_Through_Rate","pyFactory
"pxApplication":"CDH","pyPerformance":"0.500000","pyResponseCount":1.0,"pyPerformanceError":0.0,"pyConfigurationName":"Web_Click_Through_Rate","pyFactoryUpdateTim
"pxApplication":"CDH","pyPerformance":"0.673089","pyResponseCount":571.0,"pyPerformanceError":0.0,"pyConfigurationName":"Web_Click_Through_Rate","pyFactoryUpdateT
"pxApplication":"CDH","pyPerformance":"0.714575","pyResponseCount":889.0,"pyPerformanceError":0.0,"pyConfigurationName":"Web_Click_Through_Rate","pyFactoryUpdateT
"pxApplication":"CDH","pyPerformance":"0.778125","pyResponseCount":1462.0,"pyPerformanceError":0.0,"pyConfigurationName":"Web_Click_Through_Rate","pyFactoryUpdate
```


This demo has concluded. What did it show you?

- How to generate the monitoring database export artifacts.
- How to configure the auto-generated data flows to export a subset of the ADM data mart.
- How to trigger the monitoring database export.

Creating predictions

Description

Predicting customer churn is a crucial challenge, as losing customers can significantly impact profitability. To address this requirement, you can use a scorecard or a predictive model to drive a churn prediction. Learn how to connect to models in Amazon SageMaker and Google Vertex AI machine learning services, as well as how to challenge the active model with a candidate model.

Learning objectives

- Build a scorecard to drive a churn prediction in Prediction Studio.
- Create a new prediction that uses a predictive model.
- Connect to machine learning services.
- Challenge a predictive model.

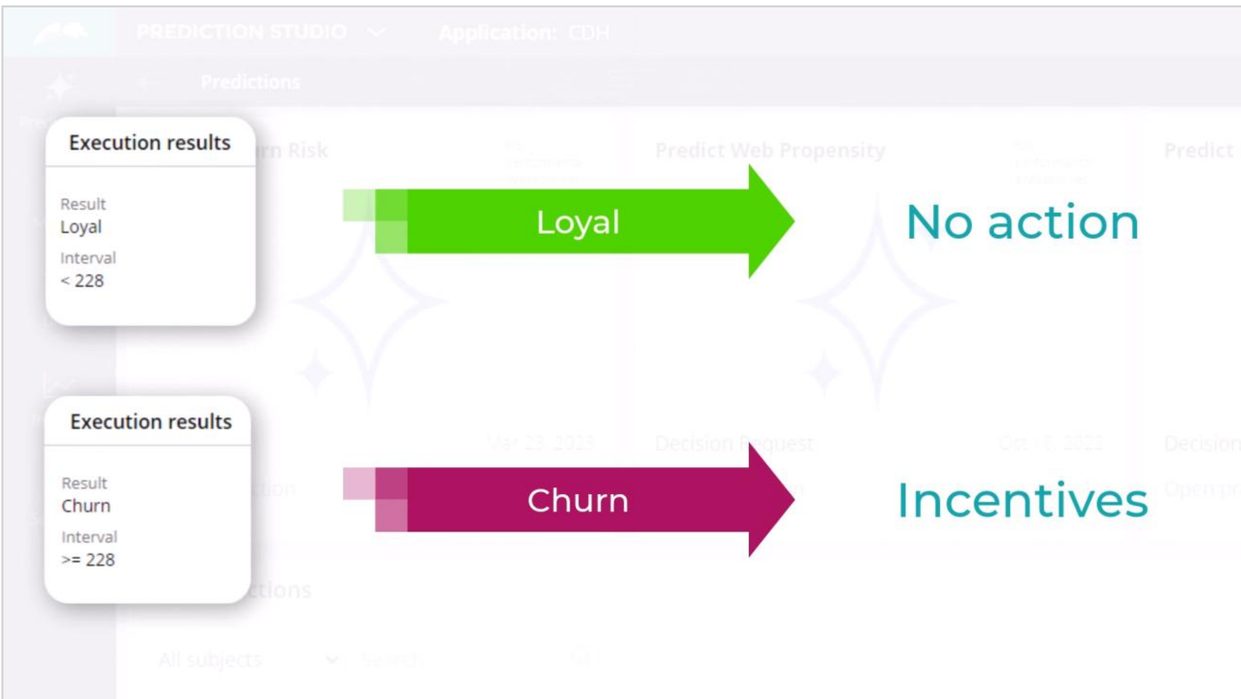
Creating a churn prediction using a scorecard

Predicting customer churn is a crucial challenge, as losing customers can significantly impact profitability. To address this requirement, you can use a scorecard. A scorecard is a transparent predictive model that can drive a churn prediction.

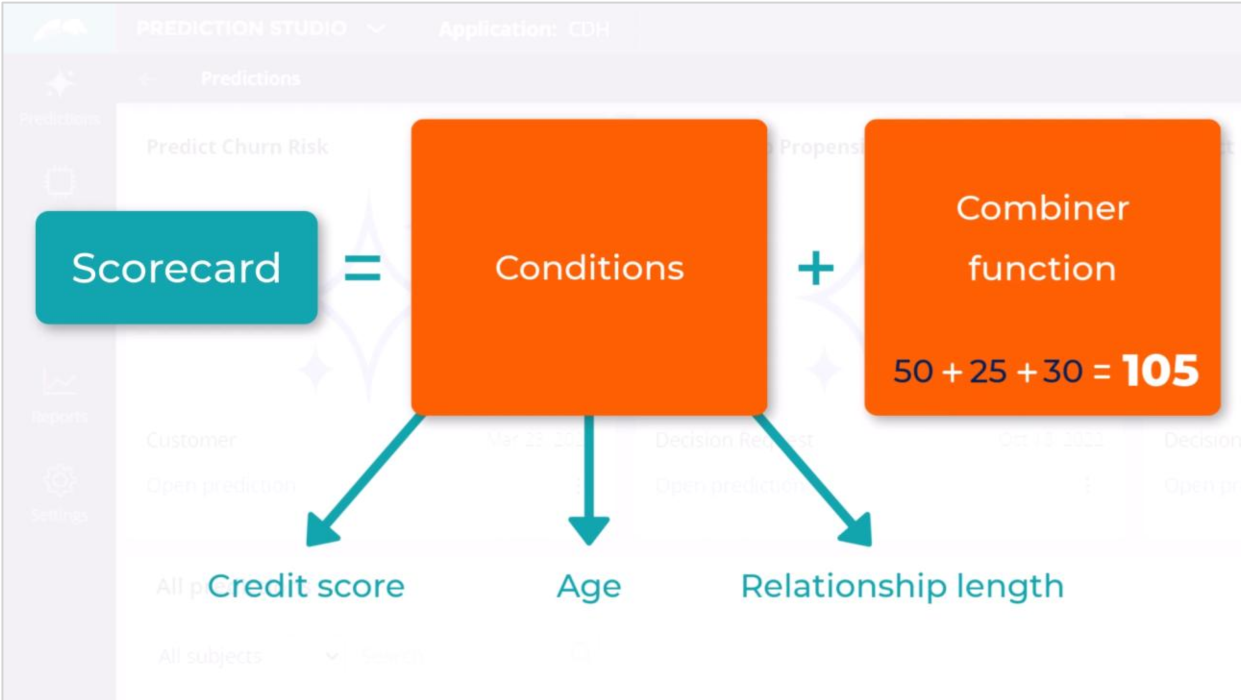
Transcript

In this video, you will explore the use of scorecards to drive predictions in Pega Customer Decision Hub™.

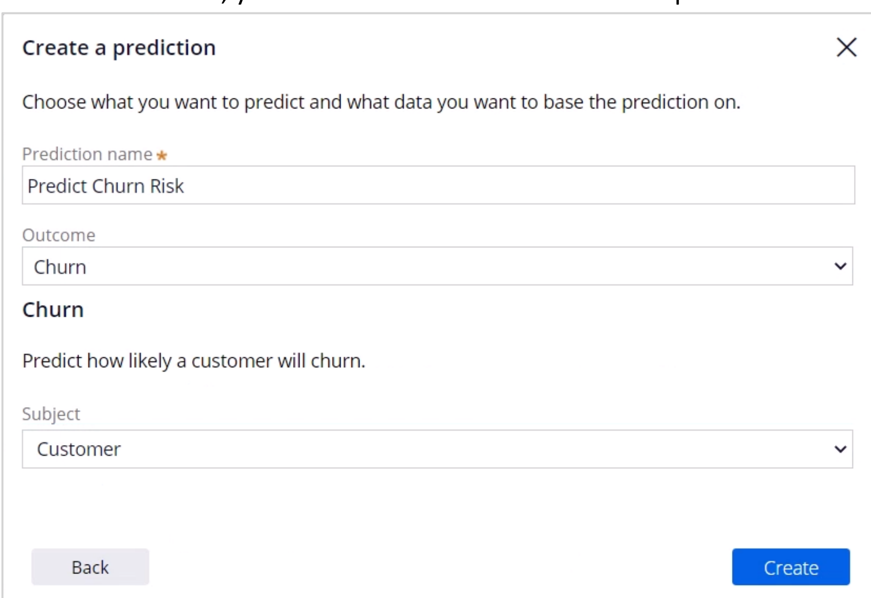
U+Bank implements Pega Customer Decision Hub to optimize customer engagement on the banks' website. The way they reduce the number of customers leaving the bank is through a churn prediction created by a data scientist in Prediction Studio. This prediction is then used in engagement policies, allowing the bank to proactively offer incentives to customers that are likely to leave.



Predictive models, pre-calculated fields, and scorecards can all drive a prediction. A scorecard is a transparent predictive model that assigns a score to each customer based on specific conditions for each predictor. Customers receive points based on these conditions, and the score is calculated using a combiner function, for example, by summing up the points.

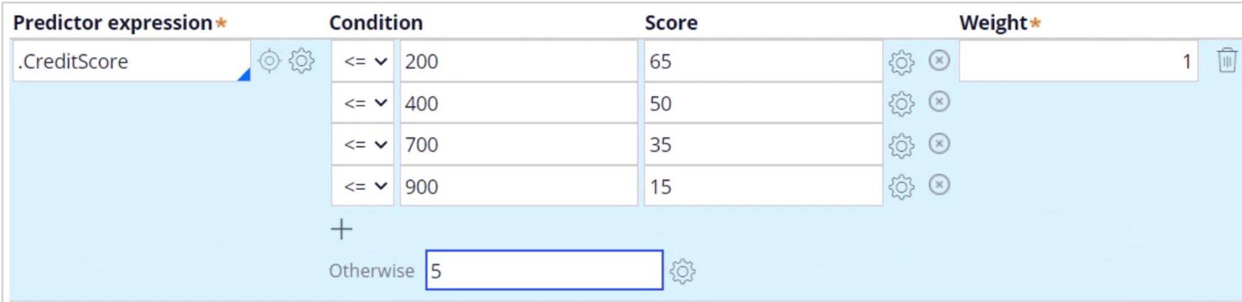


To use a scorecard, you create a Customer Decision Hub prediction in Prediction Studio.

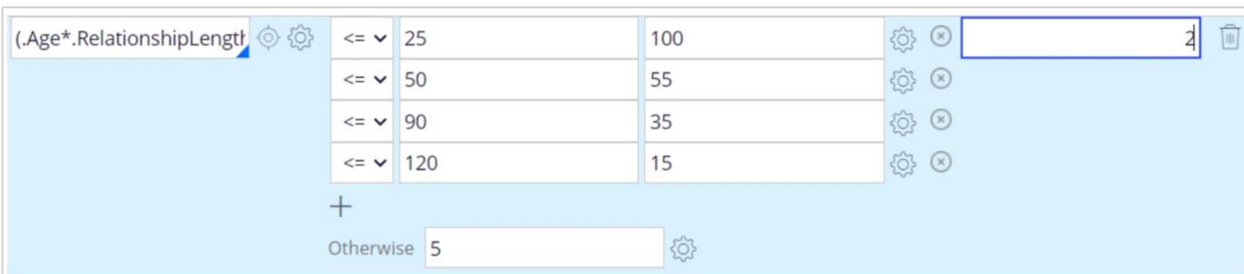


Default churn prediction includes an out-of-the-box template for a scorecard, that you can edit from the Models tab. Let's add the first predictor field - **.CreditScore**.

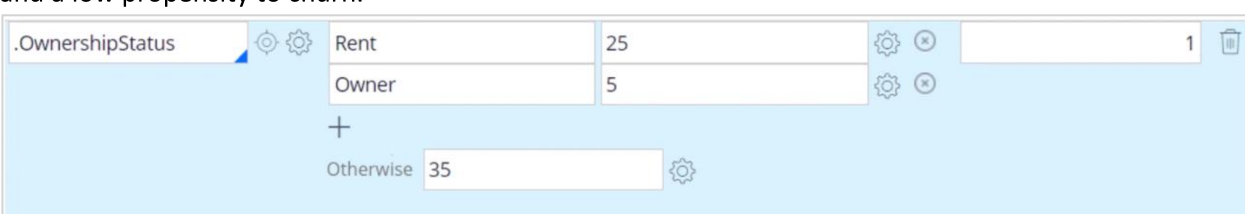
For the numerical predictor **.CreditScore**, customers with a credit score below or equal to 200 will get 65 score points, between 200 and 400, or equal to 400, will get 50 score points, and so on. Lower credit scores may indicate a higher risk of leaving, as customers may have more difficulty obtaining loans or other financial products from the bank.



Age and length of the customer relationship are important factors in credit risk assessment. The scorecard allows for complex expressions that can involve multiple predictors and calculations. We use both the **.Age** and **.RelationshipLengthDays** predictors to create a predictor field expression. The score is calculated by multiplying the predictor values and dividing the result by 100. Younger customers and those with shorter relationships receive more points, as they may be less financially stable and more likely to churn. You can set a weight to this predictor to assign its relative importance.



The scorecard also allows you to add categorical predictors, where you can assign a score for an individual value. For the categorical predictor **.OwnershipStatus**, a customer that meets the condition to be a house owner scores less points, as house ownership may indicate financial maturity, and a low propensity to churn.

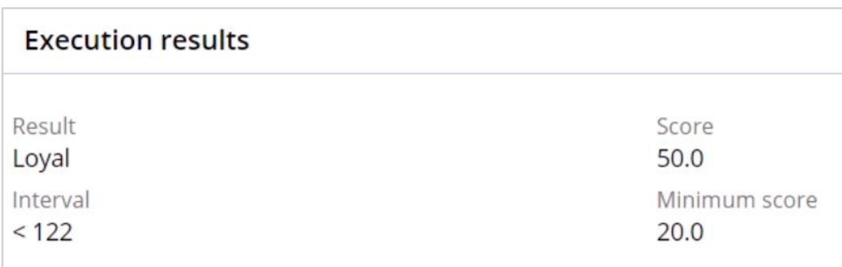


The Combiner function enables you to select a method for combining scores, where in this scorecard, the points assigned for each predictor are summed.

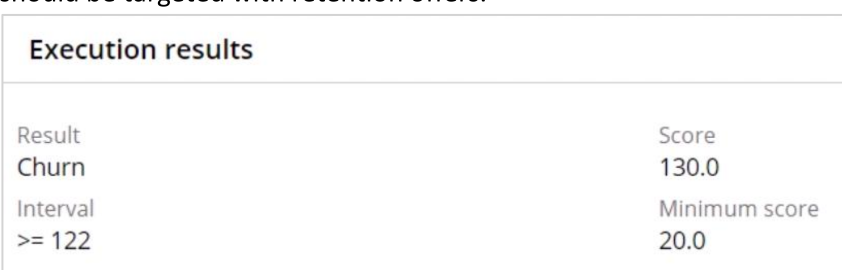
On the Results tab, you can map the Cutoff value to distinguish potential churners from loyal customers. In this case, the scorecard predicts that customers with less than 122 points are likely to remain loyal to

U+Bank, while customers with higher points are likely to churn.

To test the scorecard, you can apply a data transform to run it for different customers. For example, Barbara. In the execution details section, you can see the points assigned for each predictor field. Note the **.Age** and **.RelationshipLengthDays** predictor expression. The final points double the score because of the weight value. The combiner function sums up the points to give Barbara a score of 50, indicating that she is a loyal customer.



Next, we run the scorecard for Robert. Robert's score is 130, which suggests that he is likely to churn and should be targeted with retention offers.



You have reached the end of this video. You have learned:

- How to create a prediction in Pega Customer Decision Hub to predict the risk of churning.
- How to build a scorecard to drive the churn prediction in Prediction Studio.

Using machine learning services

Prediction Studio provides out-of-the-box support for Amazon SageMaker and Google Vertex AI machine learning services. This is useful when you need to use highly specialized algorithms or frameworks that are not built into Pega by default, giving you much greater flexibility.

Transcript

Hi there and welcome. My name is Iris, and I am an intern with the Pega Enablement Team. I am excited to be here today to walk you through configuring and using external machine learning services in Pega Customer Decision Hub™. In this session we will cover supported cloud providers, detailed configuration steps, performance considerations, and best practices.

So, let's start with the basics. When we talk about external machine learning services, we're referring to the ability to integrate Pega with powerful AI and ML models hosted on cloud platforms. Unlike native models that run directly in Prediction Studio, this approach sends data to a service like Amazon SageMaker for processing. The results are then sent back to Pega. This is incredibly useful when you need to use highly specialized algorithms or frameworks that are not built into Pega by default, giving you much greater flexibility.

Pega Platform provides out-of-the-box support for Amazon SageMaker and Google Vertex AI, two of the biggest players in the cloud ML space. SageMaker is Amazon's comprehensive service. Vertex AI unifies Google's ML offerings into a single platform. Both allow you to tap into a vast ecosystem of tools and will look at the specific frameworks Pega supports for each one next.

Diving into Amazon SageMaker, you will see Pega supports a variety of popular frameworks. This includes deep learning with TensorFlow, Gradient Boosting with XG Boost, and several algorithms for clustering, classification, and anomaly detection. For most of these, you will be working with CSV for input and JSON for the output predictions.

For Google Vertex AI, the support is slightly different. Pega integrates well with Google's AutoML capabilities, which is great for rapidly developing models. You can also connect to custom built models and those using the popular Scikit-learn Library, and XG Boost is available here as well. I want to call out one important limitation. Currently, Pega does not support PyTorch or TensorFlow models, specifically within the Google Vertex AI integration. This is a key difference to remember when choosing your platform.

Before you can configure anything in Prediction Studio, you must set up the proper authentication in Dev Studio. For Amazon SageMaker, you will create an AWS authentication profile, providing it with credentials that have the necessary permissions to invoke your SageMaker model endpoints. For Google Vertex AI, the process uses an OAuth 2.0 profile to securely connect to your GCP service account, using standard grant types like client credentials.

Once authentication is handled, the configuration process in Prediction Studio follows these four main steps. First, you establish and test the connection to the service. Second, you create a new predictive model rule in Pega. The third and most technical step is configuring the model metadata via a JSON file, which we will discuss in more detail. Finally, you map the models inputs to the corresponding properties in your Pega application.

We will walk through these steps now. In Prediction Studio settings, you'll create the machine learning service connection. This is where you link the authentication profile you made earlier and provide the endpoint details. The test connection button is your best friend here. Always use it to confirm everything is working before you move on. Once the connection is green, you can proceed to create the actual predictive model rule, selecting your new service from the list.

Step three is arguably the most important and technical part of the configuration, the model metadata file. This JSON file acts as a contract, telling Pega exactly how to interact with the external model. It defines the model's objective, what kind of outcome to expect, like a binary yes-no or a continuous number, and a list of all the input predictors. My biggest piece of advice here is to have your data science team generate this file automatically as part of their model building pipeline. This prevents typos and mismatches that can be very difficult to debug later.

After all that configuration, how do you actually use the model? The good news is it's very straightforward. Within a Next Best Action strategy, you use the standard predictive model shape and select the model you just configured. From a strategy designer's perspective, it feels just like a native Pega model. Behind the scenes, Pega handles the entire data flow, collecting the necessary data, securely sending it to the external service, and then integrating the returned prediction back into your strategy execution.

While external models are powerful, there are practical trade-offs to consider. The biggest one is latency. Every call has to travel over the network, get processed, and travel back. Which takes more time than a native model. You also need to be mindful of the cost. As these cloud services are typically pay-as-you-go. Therefore, these services are ideal for scenarios where the need for a highly complex or specialized model. Like for churn prediction or credit risk, justifies the potential latency and cost.

To wrap up, let's cover some common issues and best practices. If you run into problems, the issue is often in one of three areas. The connection itself, the data format, or authentication. Systematically check your author profiles, network settings, and that critical metadata file. For long-term success, always enforce security best practices like using encrypted connections and rotating credentials. And finally, treat this as a living integration. Regularly monitor its performance and cost, and be sure to update the metadata in Pega whenever your data science team retains or changes the external model.

That's it for me on using external machine learning services within Pega. Thanks for watching, see you next time.

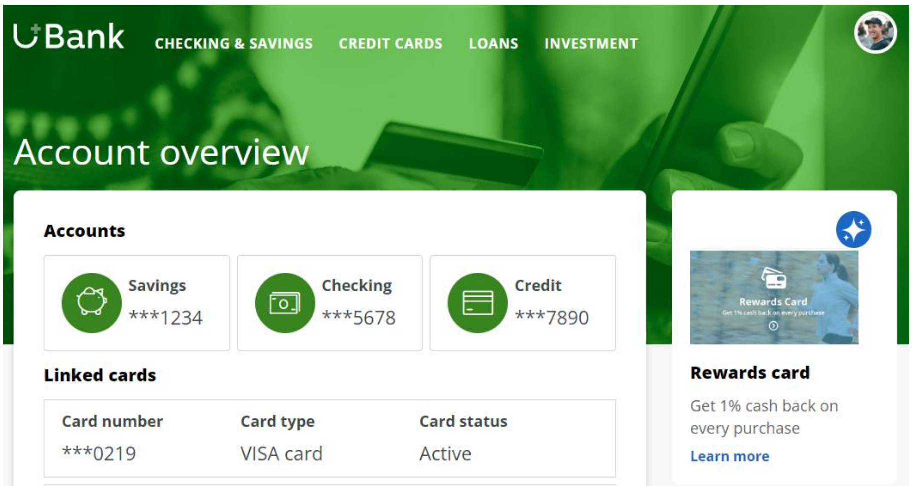
Challenging a predictive model

In the rapidly evolving landscape of data-driven decision-making, organizations are continually seeking ways to enhance their predictive models and ensure optimal outcomes. Learn how to shadow and challenge a Champion model, and ultimately promote the Challenger model to Active status in Pega Customer Decision Hub™.

Transcript

Hi, I'm Bella, a Lead Decisioning Architect working on a Customer Decision Hub project. In this demo, I will show you how to introduce a candidate model for a churn prediction.

U+ Bank uses Customer Decision Hub to optimize customer engagement on the bank's website.

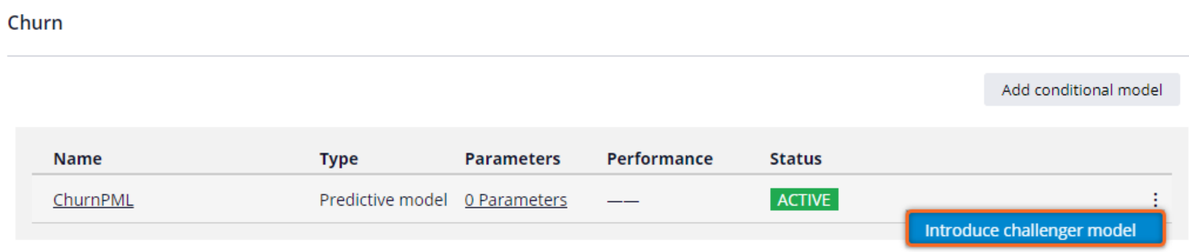


To reduce the number of customers leaving the bank, Customer Decision Hub uses the Predict Churn Propensity prediction. This prediction calculates the likelihood that a customer will switch to a competitor soon. A predictive churn model built on historical data drives the prediction.

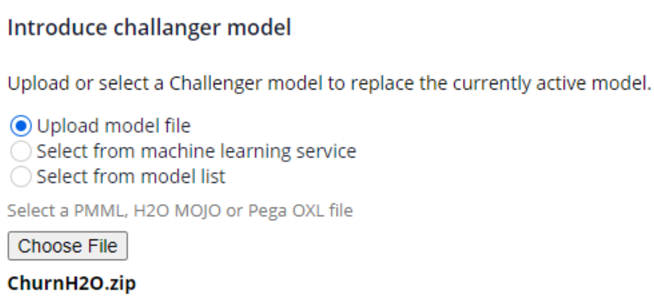
The bank personalizes interactions with customers that have a high churn risk, like Troy, and prioritizes retention offers. In this case, Troy receives a 'free extra miles' offer in the hope of retaining him. In contrast, customers with a low churn risk, like Barbara, receive a credit card offer. The predictive power of the churn model declines over time as customer behavior changes, and the model needs to be updated regularly.

You can gently introduce an updated model in a series of deployment cycles. Begin by placing the candidate model in shadow mode, so it does not impact business outcomes. When you submit your changes for deployment, an automatic change request is created in the open revision. The Revision Manager deploys the revision, and the changes become effective in production. Next, you can utilize a Champion Challenger pattern to gradually increase the utilization of the Challenger model instead of the Champion model over time. Lastly, promote the Challenger model to Active status to replace the Champion model.

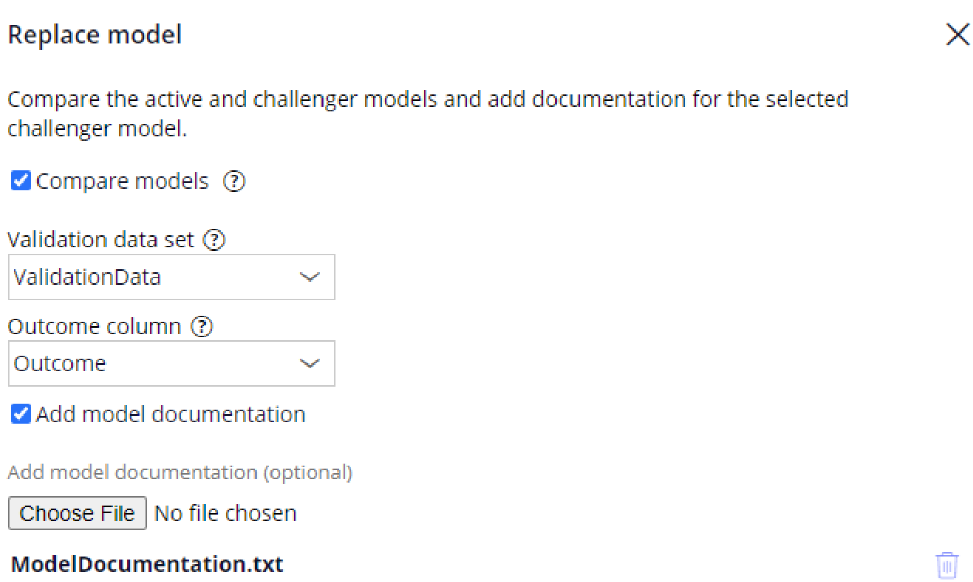
When you introduce a candidate model, you have the option to validate the candidate model and compare it with the Active model. For this, you need a validation dataset. This validation data set contains the predictor values and the outcome for a set of customers.



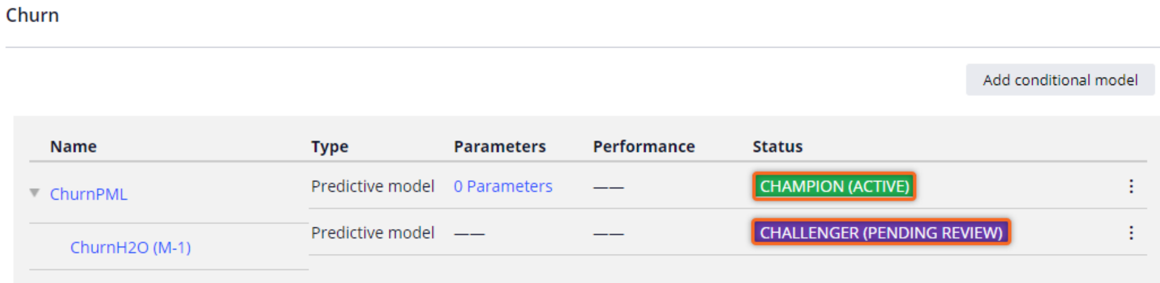
When introducing a Challenger model, you have different options to select the model. You can use a file-based model, connect to a machine learning service like Google AI or Amazon SageMaker, or select a model that is available in your application.



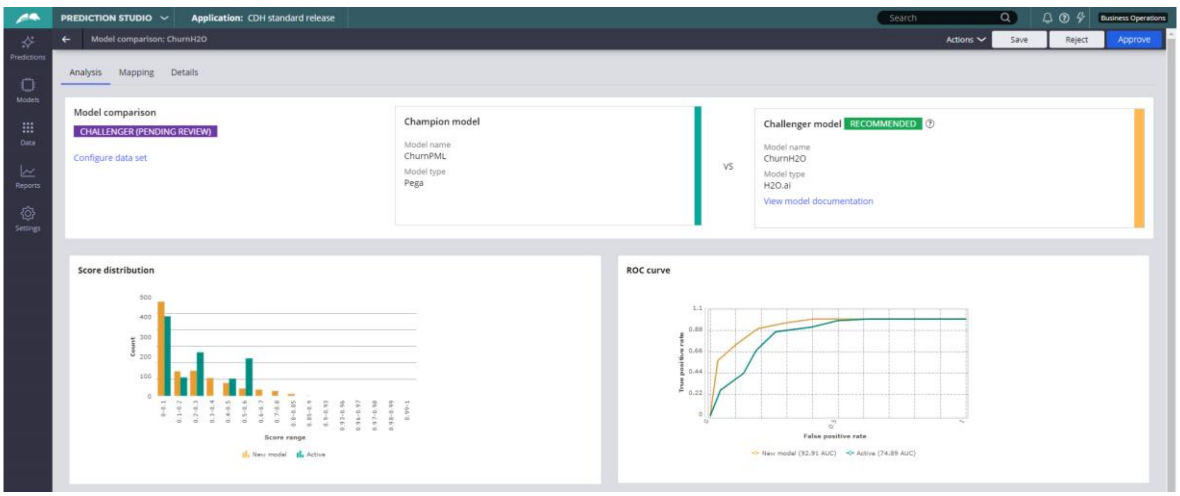
To compare the Active and Challenger models, use your validation data set. Optionally, add supporting documents that provide detailed information about the model, its purpose, and its components. These documents help stakeholders understand the structure and functionality of the model, making it easier to maintain, update, and share with others.



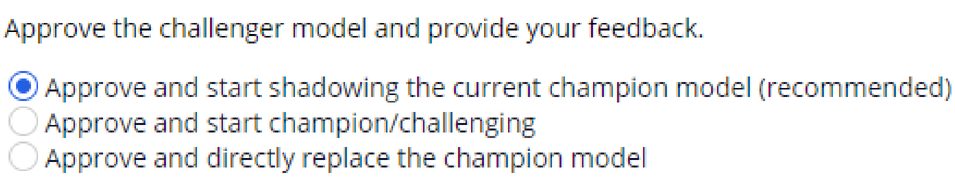
The system configures the challenger model in a pending review state.



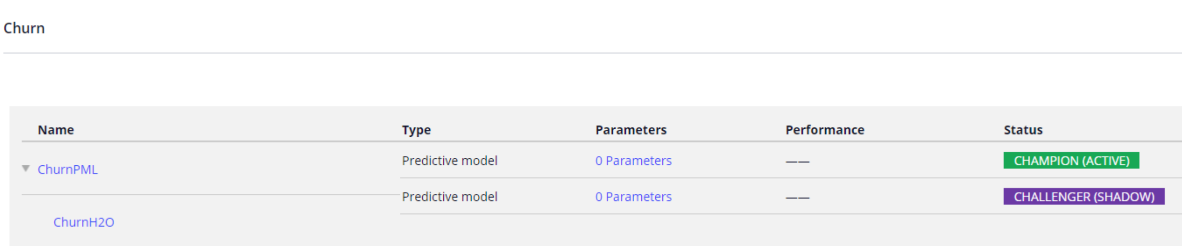
Review the predictive power of the Challenger model compared to the current Champion model, and then approve or reject the Challenger model.



You have the option to shadow, challenge, or directly replace the Active model with the Challenger model.

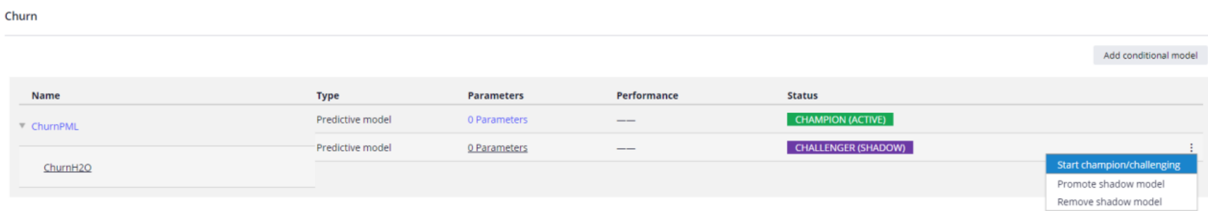


It is recommended to start a Challenger model in shadow mode. In shadow mode, the challenger model runs alongside the current model, receiving production data and generating outcomes without impacting business decisions.

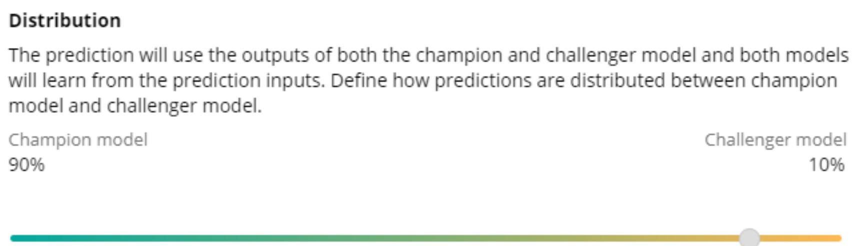


When you submit for deployment, a change request containing all changes made to the prediction is created in the open revision. When the Revision Manager deploys the revision, the changes take effect in production. Note that your validation data set can be included in the revision.

After running the Challenger model in shadow mode for some time, and the model performs without errors and the range of scores is as expected, you switch to a Champion Challenger pattern.

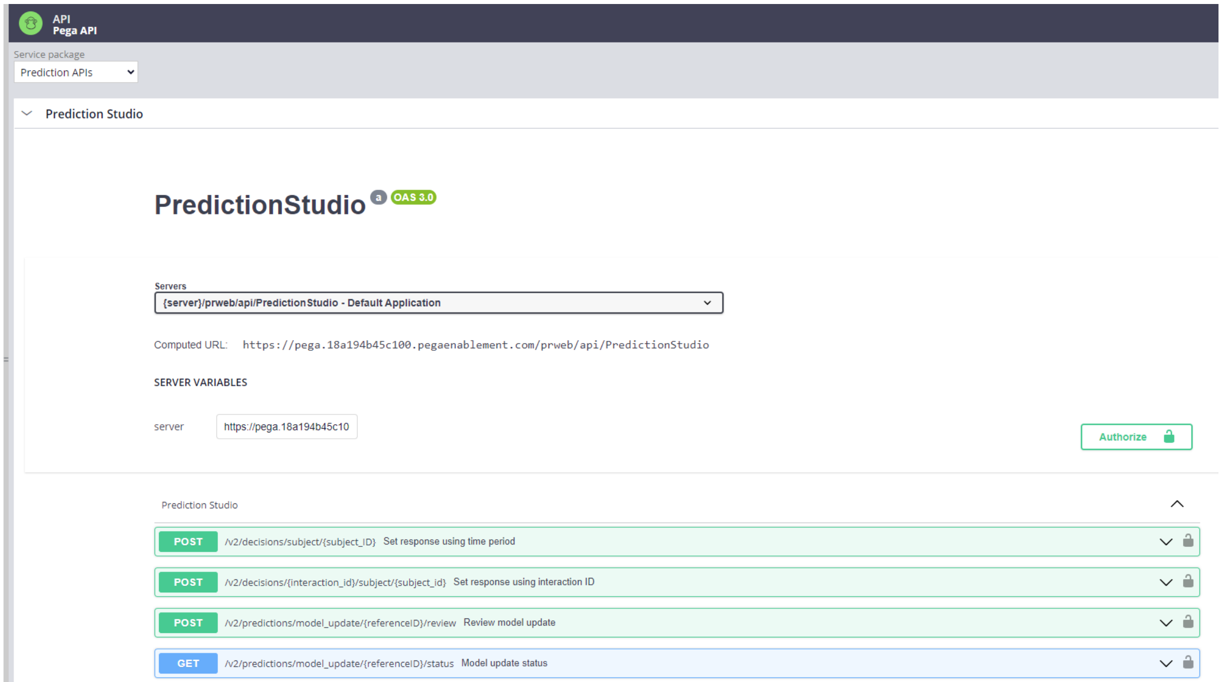


When you challenge the Active model, in a small percentage of cases the Challenger model is used instead of the Active model.



You can edit the distribution to gradually increase this percentage to give the Challenger model more exposure over time and then initiate the promotion of the Challenger model to Active status. By promoting the challenger model, you replace the Champion model with the Challenger model.

To introduce a Challenger model, you can also use the available Prediction APIs.



The API provides options to add a model to your application, review and approve or reject a model update, and retrieve the status of a model update.

This allows you to integrate the model approval process in Prediction Studio with any external model deployment process.

To summarize:

- You can shadow the Active model with a candidate model in production.
- When you submit your changes for deployment, a change request is automatically created in the open revision.
- When the Challenger model functions as expected in production, you can gradually increase the use of the Challenger model.
- You can promote the Challenger model to Active status and replace the Champion model.

Decision Strategies Overview

Description

Decision strategies optimize business processes by using data-driven, real-time arbitration to select the best actions for specific contexts. This module explains how to create and run a decision strategy, detailing the use of various decision strategy components. It covers the importance of understanding component interactions, building expressions, and testing strategies to ensure customer-centric interactions that enhance customer experience and business outcomes.

Learning objectives

- Create and configure decision strategies from scratch using various decision components.
- Utilize various components to build effective decision strategies.
- Develop and validate expressions within decision strategies to calculate and prioritize actions.
- Test and refine decision strategies to ensure they meet business requirements and improve customer interactions.

Creating decision strategies

Introduction

You can optimize decision points in a business process or in a case life cycle with the help of decision strategies. Decision strategies allow you to perform data-driven, real-time arbitration to select the best action for a given context. Learn the type of decision components and how they are used to create decision strategies. Gain hands-on experience designing and executing your own Next-Best-Action decision strategy.

Transcript

This demo will show you how to create a new decision strategy.

It will also describe three important decision components and the types of properties available for use in expressions during strategy building.

In this demo you will build a Next-Best-Label strategy. The Next-Best-Label strategy is a sample strategy, used to illustrate the mechanics of a decision strategy.

Start by creating a new strategy from scratch.

Decision strategies output actions, utilizing the so-called Strategy-Results class.

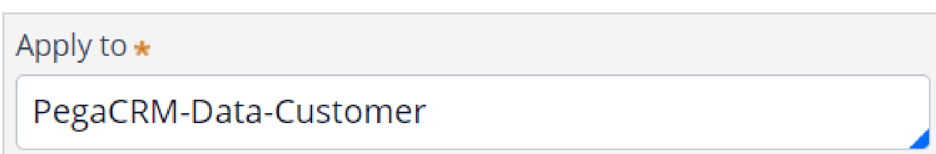
The Strategy-Results class limits the output of the strategy to the actions contained in the Business issue and Group.

The strategy you build will select a Label action from a set of predefined actions. The Label action selected will be the one with the lowest printing cost.

Notice that the complete definition of the Next-Best-Label strategy needs to include a reference to the PegaCRM-Data-Customer class.

This is the ‘Apply to’ class and it indicates the context of the strategy.

It ensures that from within the strategy, you have access to customer-related properties such as Age, Income, Address, Name, etc.

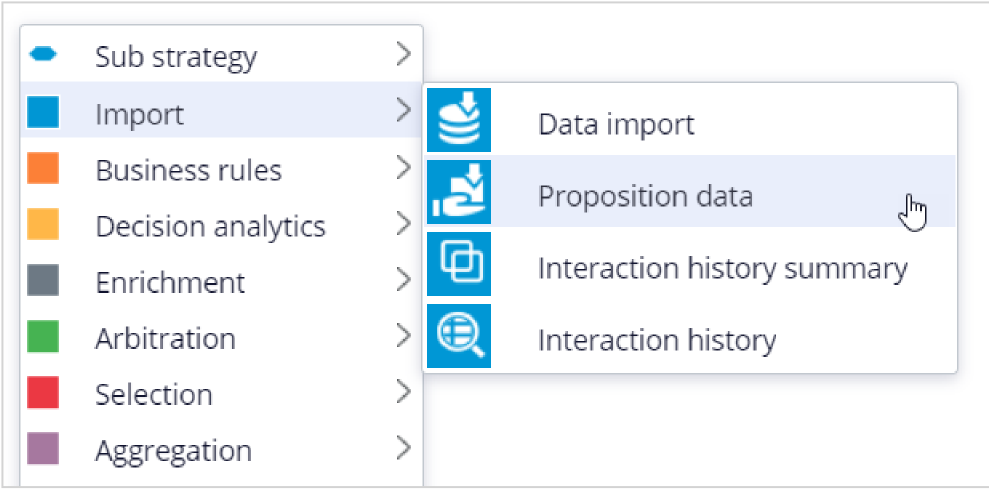


You can now start building the strategy. Right-click on the canvas to get the Context menu, which shows all component categories.

The first component to add is an Import component.

By expanding the Import category, you can see the Import component types available.

In this case you need a Proposition Data component to define the actions that will be considered by the strategy.



Now you need to configure the component. First, right-click to open the Proposition Data properties panel.

Notice that the Business issue and Group are grayed out.

This cannot be changed because the Enablement Business issue and Labels Group have already been selected for this decision strategy.

By default, the strategy will import all actions within that Group, unless you select a specific action.

For this component, you only want to import the Green Label, so let's select that.

Selecting the action from the drop-down menu automatically gives the component the appropriate name.

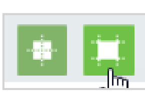
The description, which will appear under the component on the canvas, will also be generated automatically.

If you want to create your own description, you can do so by clicking the 'Use custom' radio button.

Now you want to import a second action into the strategy. You can use the Copy and Paste buttons to quickly add more Proposition Data components to the canvas.

You can use Alignment Snapping and Grid Snapping for easy placement of the components.

By turning these off, you can place a component anywhere on the canvas, but it makes it more difficult to align the shapes.



Now you need to add the next component in the strategy, which is an Enrichment component called Set Property.

You can add this component to the canvas by selecting it from the component menu.

Next, connect it to the Proposition Data components.

Ultimately, the result of this strategy should be the Label action with the lowest printing cost.

This printing cost is the sum of a base printing cost, which is specific to each label, and a variable cost, which depends on the number of letters.

The Set Property component is where you will calculate the printing cost for each of the actions.

The information in the ‘Source components’ tab is populated automatically by the Proposition Data components connected to this component.

Notice that the Black Label action is in the first row.

On the Target tab you can add properties for which values need to be calculated.

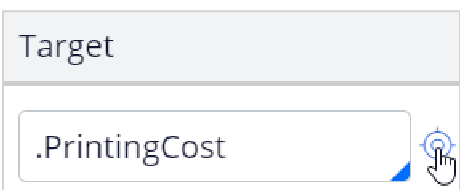
Click ‘Add Item’ to create the equation that will calculate the printing cost for each of the components.

Begin by setting the Target property to ‘dot’ PrintingCost.

In Pega, all inputs begin with a dot. This is called the dot-operator and it means that you are going to use a strategy property.

The PrintingCost property is a new strategy property that does not yet exist.

To create the new PrintingCost strategy property, click on the icon next to the Target field.



By default, the property type is Text. In Pega, there are various types supported. In this case, the PrintingCost is a numeric value, so change its type to Decimal.

Next, you need to make PrintingCost equal to the calculation you create. To create the calculation, click on the icon next to the Source field.

Using the Expression builder, you can create all sorts of complex calculations, but in this use case, the computation is very basic.

PrintingCost should equal BaseCost + 5 * LetterCount.

To access the BaseCost you type a dot. Notice that when you type the dot, a list of available and relevant strategy properties appears.

This not only makes it easy to quickly find the property names you’re looking for; it also avoids spelling mistakes.

In a decision strategy, you have two categories of properties available to use in Expressions.

The first category contains the strategy properties, which can be one of two types.

An Action property is defined in the Action form. Examples are the BaseCost and LetterCount properties you are using here.

These properties have a value defined in the Action form and are available in the decision strategy via the Proposition Data component.

The property values can be overridden in the decision strategy but will often be used as read only.

The second type of strategy property is a calculation like the one you just created, PrintingCost. Such calculations are often created and set in the decision strategy.

These types of properties are either used as transient properties, for temporary calculations, or for additional information you want the strategy to output.

The second category contains properties from the strategy context, also called customer properties.

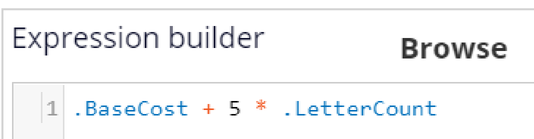
Suppose you want to use a customer property in your Expression, such as Age or Income.

In that case, you would have to type the prefix ‘Customer dot’, instead of just dot.

This is the list of available properties from the strategy context, also known as Customer properties.

For now, you calculate the printing cost for each action that does not use customer properties.

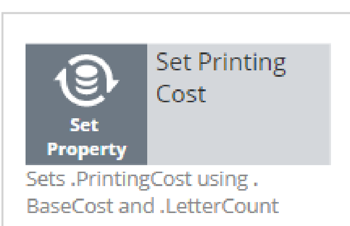
Finalize the Expression.



Even though you used the dot-operator to build your Expression, it’s best practice to validate it, so click Test.

If the Expression isn’t valid, you will receive an error message on screen.

On the canvas, you can see the automatically generated description for the component: Sets PrintingCost using BaseCost and LetterCount.



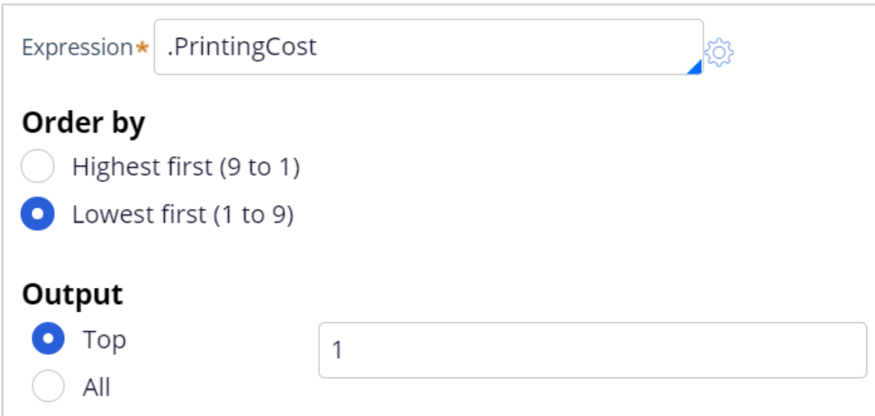
Now you want to ensure that the actions will be prioritized based on the lowest printing cost. So, you need to add the Prioritize component from the Arbitration category.

The prioritization can either be based on an existing property, or it can be based on an equation. Let’s select an existing property using the dot construct.

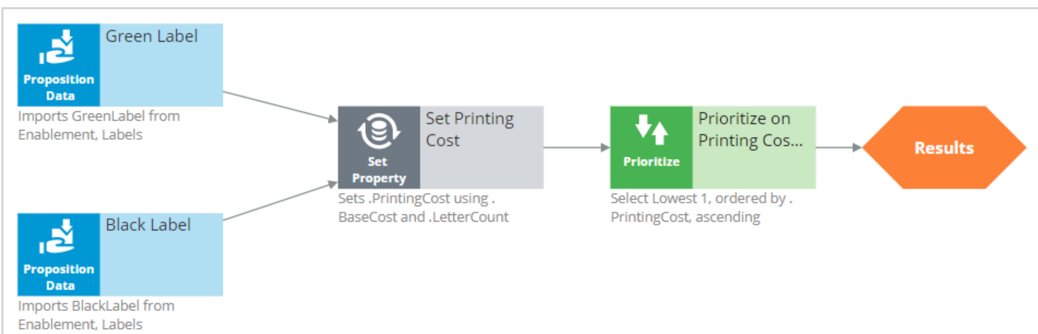
Here you can select the order in which the top actions are presented. Since you are interested in the lowest printing costs, configure it accordingly.

You can also select the number of actions that will be returned by the strategy.

If you want to output only one label, select Top 1 here.



Now you can connect the components and save the strategy.



To test the strategy, first check it out. Then, expand the right-hand side test panel and click ‘Save & Run’ to examine the results.

You can view results for any of the components by selecting that component.

If more than one action is present, each one is presented as a Page.

For the Set Property component, the Results contain a page for the Black Label and one for the Green Label.

For the Black Label the PrintingCost is 70.

For the Green Label the PrintingCost is 60.

On the canvas, you can show values for strategy properties such as Printing Cost.

For this exercise, you execute this strategy against a Data Transform called UseCase1.

If you open UseCase1, you can see the customer data the strategy uses when you run it.

To test the strategy on a different use case, you can create a Data Transform with different properties.

You can also select a Data Set that points to an actual live database table.

This demo has concluded. What did it show you?

- How to create a decision strategy from scratch.
- How to configure Proposition Data, Set Property and Prioritize decision components.
- How to build expressions in strategies.
- The two categories of properties available for expressions.
- How to test a decision strategy using a use case stored in a data transform.

Decision strategy execution

Introduction

Using Pega Decision Management, you do not need to be an expert in programming, math or data science to design and execute sophisticated decision strategies that engage your customers throughout the customer journey. With its highly intuitive graphical canvas, Pega Decision Management enables you to easily embed Pega or third-party predictive models into your decision strategies. The result is customer-centric interactions that improve the customer experience while increasing customer value, retention and response rates.

Transcript

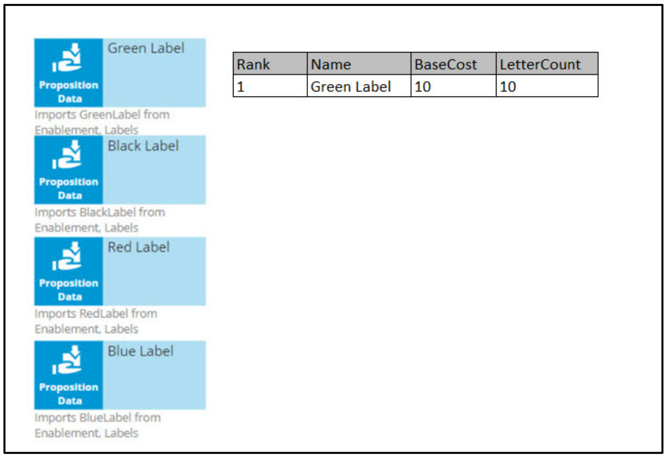
This demo explains what's going on inside each component when a Decision Strategy is executed.

For example, what happens 'under the covers' when a Filter component is executed, and how does it interact with the components around it?

In the interest of keeping it simple, this example is limited to four actions. In reality, decision strategies will involve many more actions than that.

Here are our 4 actions: 'Green Label', 'Black Label', 'Red Label' and 'Blue Label'; they are represented by a Data Import or, more specifically, a Proposition Data component.

In this example, the Proposition Data components import three data properties for each action: Name, BaseCost and LetterCount.

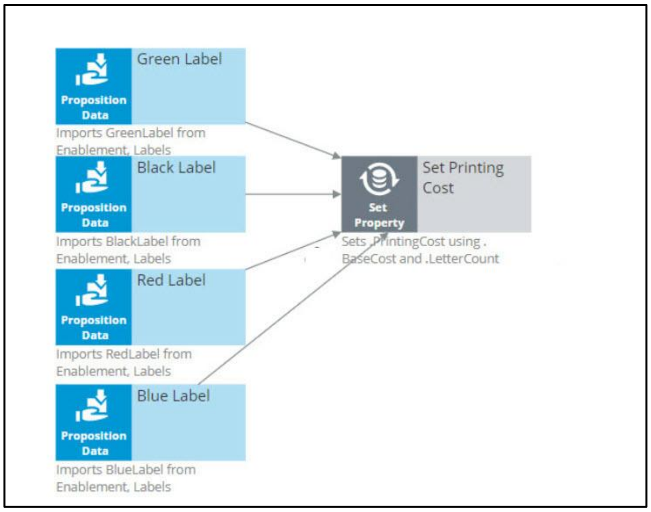


The first action’s Name is Green Label, its BaseCost is 10, and its LetterCount is 10.

Likewise, the other actions have a Name, BaseCost and LetterCount.

One property is automatically populated for you; this is the Rank. We will come back to this later, but notice that, as separate components, each action has a Rank of 1.

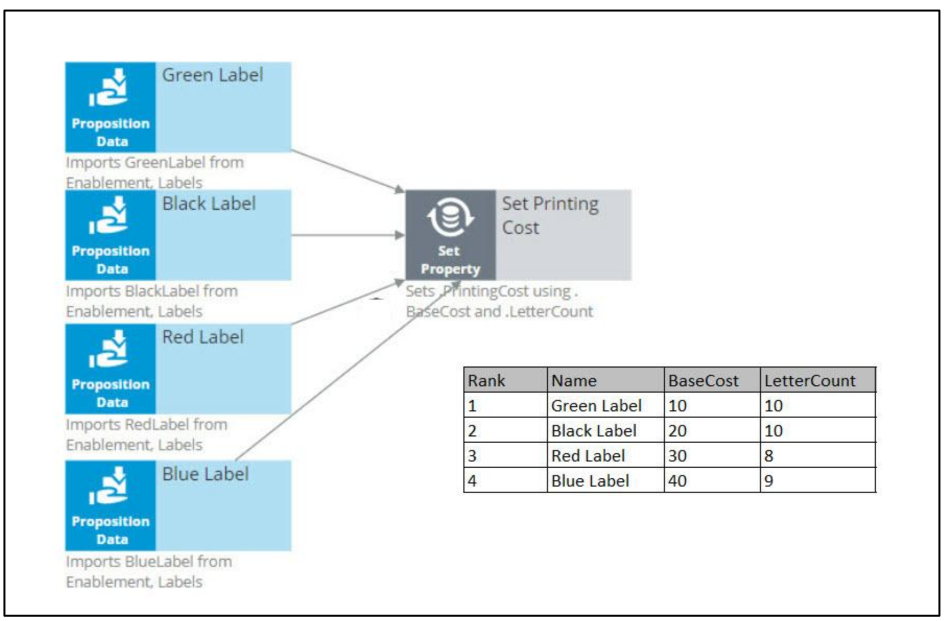
On the strategy canvas, components are connected by drawing arrows from component to component. So, what do these arrows mean exactly?



Well, when you draw an arrow, what happens is that, at runtime, all information in the component you're drawing the arrow from is available as a data source to the component you're drawing the arrow to.

So now, the Name, BaseCost and LetterCount for all of the actions are available in a single Set Property component.

The only data element that changes is the row number, or as we call it in the strategies, the Rank. In each decision component, the Rank value is automatically computed.



Rank	Name	BaseCost	LetterCount
1	Green Label	10	10
2	Black Label	20	10
3	Red Label	30	8
4	Blue Label	40	9

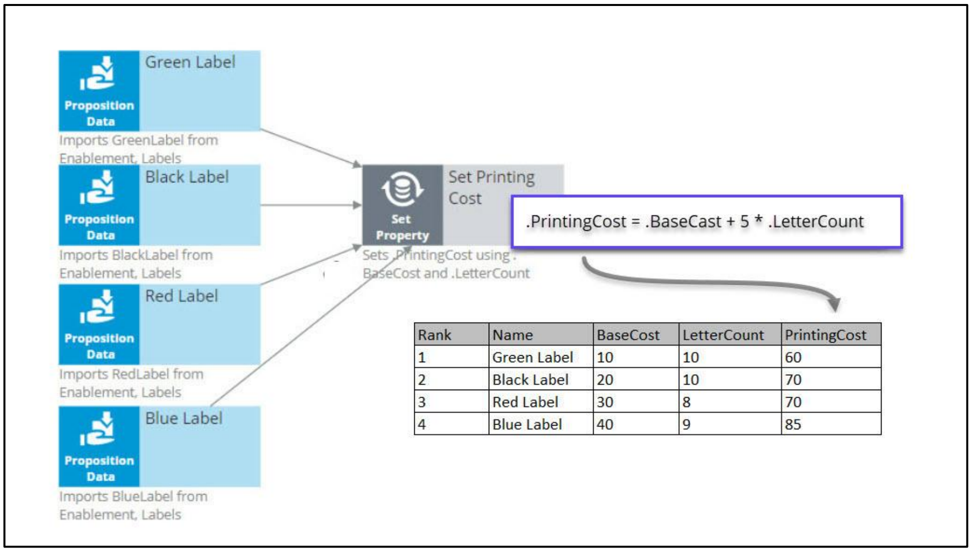
In the Set Property component, the Rank is determined by the order in which the actions are received by the component.

As a result, in this instance, the Green Label action has a Rank of 1, Black has a Rank of 2, Red has a Rank of 3, and Blue has a Rank of 4.

Ultimately, you want to select the best Label action. That is the Label with the lowest printing cost.

The printing cost of a Label is the sum of the BaseCost and a variable cost based on the LetterCount.

You configure the Set Property component to compute the printing cost of each Label action.



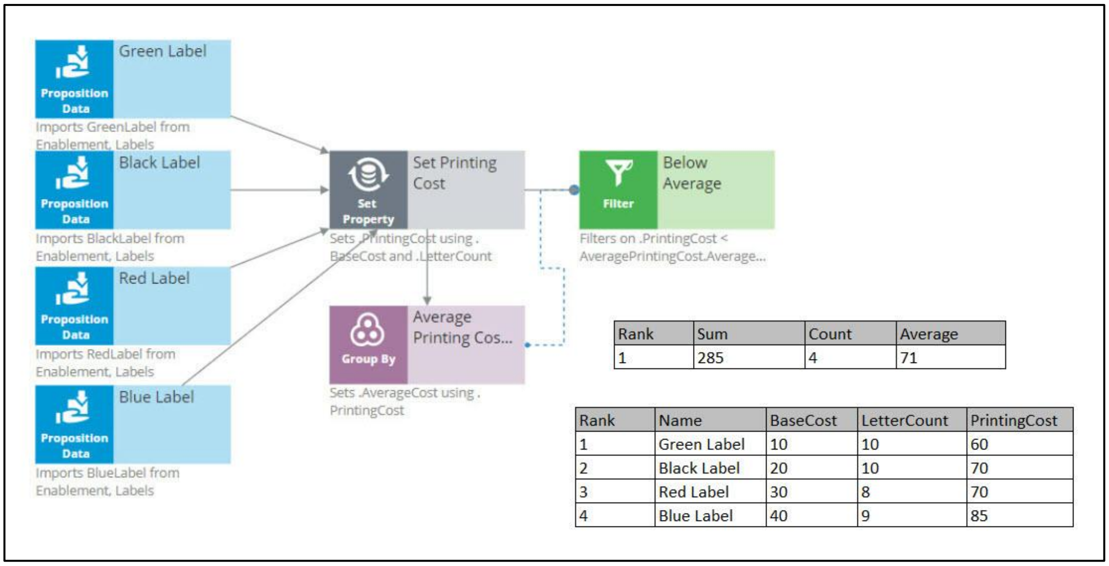
Because we are combining the data in our four Proposition Data components into one Set Property component, we only need to add one PrintingCost property to the new component, and it automatically computes the printing cost for all four actions.

For the Green Label action, PrintingCost equals a BaseCost of 10 plus 5 times the LetterCount of 10 which equals 60.

Similarly, the PrintingCost for the Black and Red Label actions is 70, and for the Blue Label action is 85.

Now, let’s say the business rule is to select only Label actions with a printing cost lower than the average printing cost of all labels. For this requirement we use a ‘Group by’/Filter component combination.

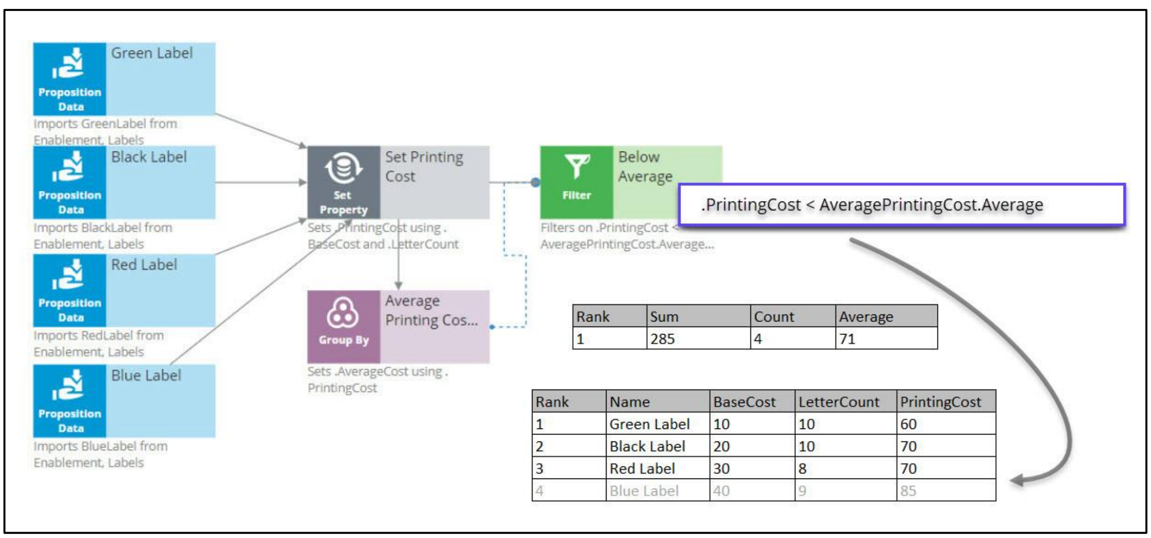
A ‘Group by’ component offers essential aggregation capabilities, like Sum and Count, that are used in many decision strategies. We will use it to calculate the average printing cost.



Again, we have our set of actions, each with their own specific PrintingCost value. The ‘Group by’ component combines all actions into one row. How does that work?

Well, it sums the PrintingCost values for all the actions, it counts the actions, and it calculates the average printing cost by dividing the summed printing cost by the count.

In this example, the sum of the PrintingCost values is 285, and the count of the actions is 4, so the average printing cost is 71.



Now that you have calculated the average printing price using a ‘Group by’ component, configure the Filter component to filter out actions that have a printing cost equal to or higher than this average.

So far in this strategy, we’ve seen only the solid line arrows, which copy information from one component to another. But now we also see a dotted line arrow.

This tells us that a component refers to information in another component.

Here, the Filter component is referencing the average printing cost that exists inside the Aggregation component. This is an important capability to understand.

The Filter component filters out actions when the printing cost for that action is equal to or above the average printing cost and propagates the other actions.

First, via the solid arrow, the filter looks at the actions sourced from the Set Property component.

Then, it applies the filter condition, which references the average printing cost in the 'Group by' component via the dotted arrow.

The Filter Condition in the Filter component is the Expression: 'dot PrintingCost is smaller than AveragePrintingCost dot Average'.

By using this ComponentName dot Property construct, any decision component can be referenced by any other component by name.

Important to note that the Filter component lets actions through when the condition Expression evaluates to **true** and filters out actions when the condition Expression is not met.

When you refer to a component, you always refer to the first element in the component, the one with Rank 1.

In this case, you are referring to the one and only row in the 'Group by' component, which naturally has Rank 1.

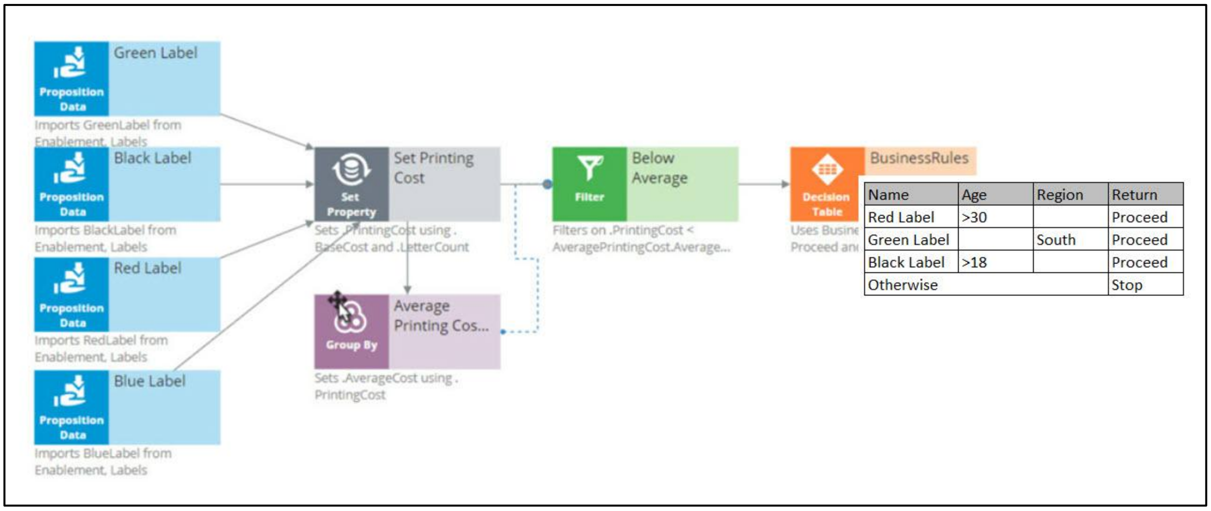
The Rank 1 average equals 71 in the 'Group by' component. This means that the filter will allow Label actions through that have a printing cost lower than 71.

By this standard, the printing cost of the Blue Label action is too high, so it is filtered out. The printing cost of the other Label actions are below 71, so they survive.

The result is that the table contains three surviving actions: Green Label with Rank 1, Black Label with Rank 2, and Red Label with Rank 3.

The next component is a Decision Table. A Decision Table in Pega is an artifact that can be used to implement business requirements in table format.

In a Decision Table, the business rules are represented by a set of conditions and a set of Return values.



The Decision Table receives information about the remaining actions via the solid arrow from the Filter component.

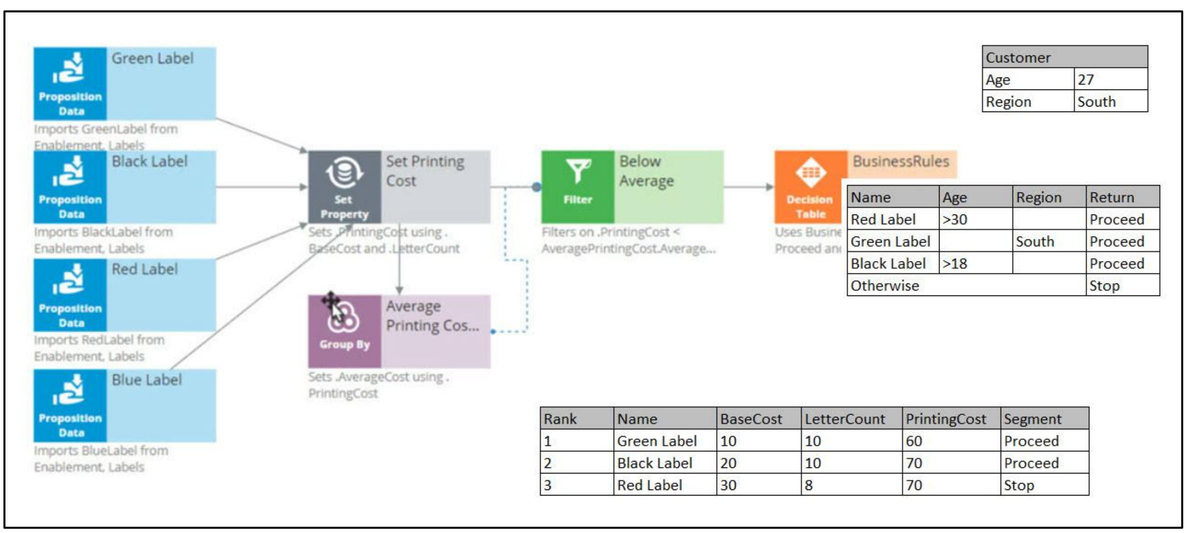
The business criteria say that the Red Label action can be offered if the customer's age is over 30 and they are from any region. If these criteria are met, the Return value is 'Proceed'.

The Decision Table also says that the Green Label action can be offered to anyone in the Southern region. So, if the Region value is South, the Return value for Green is 'Proceed'.

The Black Label action can be offered to anyone over the age of 18.

But in all other cases, or, Otherwise, no Label action meets the criteria, and the Return value is 'Stop'.

As an example, consider a customer with Age 27 and Region South.



Now, the Decision Table applies the business criteria for each action against the customer information and returns a value. The value returned by a Decision Table is also called a Segment.

The Decision Table checks the Green Label action with Rank 1 first, and in this case, it can proceed because the customer's Region is South.

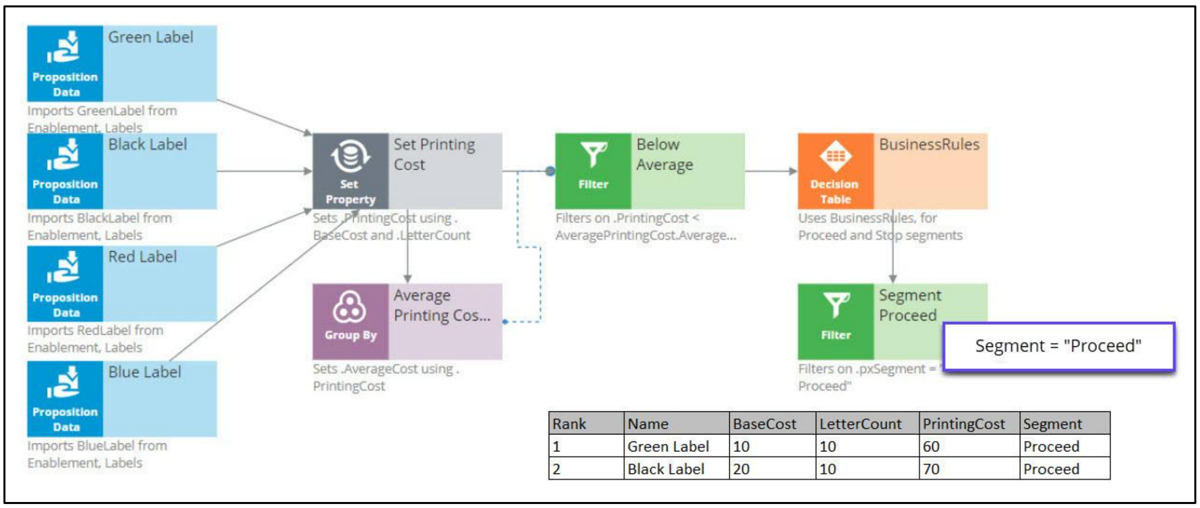
Next, it looks at the Black action and sees that the criteria for Black is that the customer's age is greater than 18. This customer is 27.

Black doesn't care about the Region, so the Segment value for the Black action is 'Proceed'.

Finally, it looks at the Red action, and the Age criteria don't match up, so the Segment value for Red is 'Stop'.

The result of the component is that you get a new segmentation column that flags which of the actions comply with the business rules.

You're now going to filter out the actions that do not match the business rules. This happens in the 'Segment Proceed' Filter component.



Rank	Name	BaseCost	LetterCount	PrintingCost	Segment
1	Green Label	10	10	60	Proceed
2	Black Label	20	10	70	Proceed

Again, via the solid arrow, the strategy copies the data over from the Decision Table component into the Filter component.

Now each action has a Rank, Name, BaseCost, LetterCount, PrintingCost and Segment. The filter condition is applied to this data.

The filter condition says: allow this action through if the Segment value equals 'Proceed'.

What this Filter component now does is go through the list of actions to find the actions with value 'Proceed' in their Segment property.

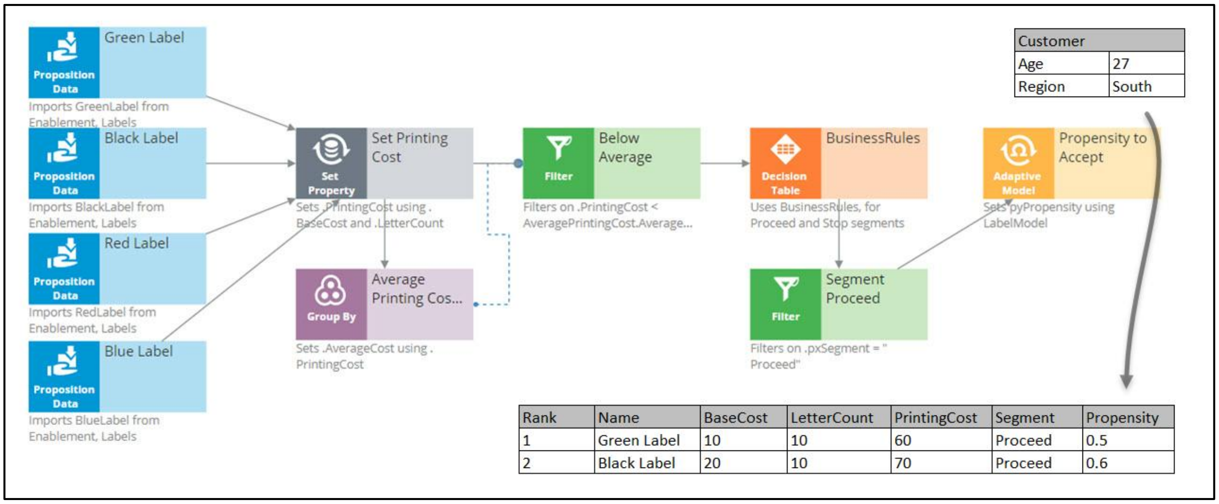
First is the Green Label. Green is allowed through, which means its properties will be available in the new component.

Then the Black Label. It is also allowed through because it also has 'Proceed' in its Segment property.

But the Red Label action is not allowed through, because Red has 'Stop' in its Segment property. Therefore, Red is not part of the output.

The strategy so far has selected two of our original actions, Green and Black.

Now, in the Adaptive Model component, you will use predictive analytics to determine the propensity of each of the remaining actions.



Propensity is the probability that a customer will accept an action, or their likelihood of interest in it.

In order to calculate the propensity, we use an Adaptive Model component. The referenced model is configured to monitor customer characteristics such as Age and Region.

In this case our test customer has an Age of 27 and is from the South Region.

Again, just to keep it simple, we are using a model that makes predictions based on only this information. In reality, models will take into account many more properties.

The Adaptive Model determines the propensity.

First, we supply the action and the customer profile to the Adaptive Model, and the model says: ‘Oh, it’s the Green Label action; we have some evidence that young people like the Green Label action, but people from the South don’t like it.’

Combining both factors, we get an overall propensity of 0.5 for the Green Label action.

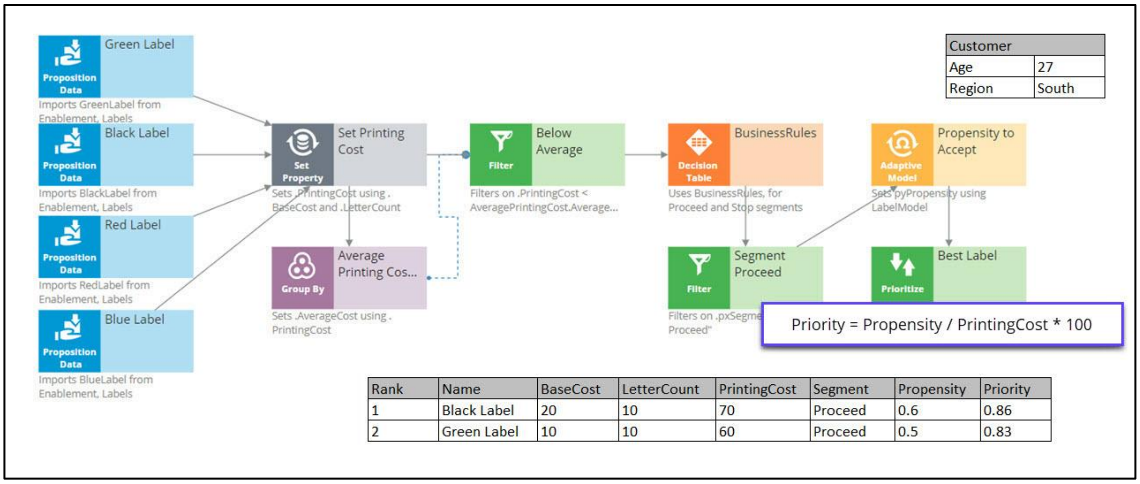
For the Black Label action, the likelihood turns out to be 0.6.

After consulting the Adaptive Model, the Propensity to Accept component sets the Propensity property value for each action.

Remember, the propensity is always a number between zero and 1.

It shows something along the lines of, half of the customers that are like this customer accepted the Green Label action in the past, and 3 out of 5 customers like this customer accepted the Black action last month.

The next component in our chain, called Best Label, is the Prioritize component. This component determines the priority of each action and ranks them. Let’s see how this works.



A key element of this component is the priority Expression, which calculates a priority value for each action. According to this Expression, the higher the value, the higher the priority and rank.

In this case, the priority calculation weighs likelihood of acceptance in its equation: 'Propensity divided by PrintingCost times 100'.

When performing this calculation on the Black Label action, we can see that it has a PrintingCost of 70 and a Propensity of 0.6, therefore its Priority is 0.86.

The Green Label action has a lower PrintingCost and a lower Propensity, resulting in a Priority of 0.83.

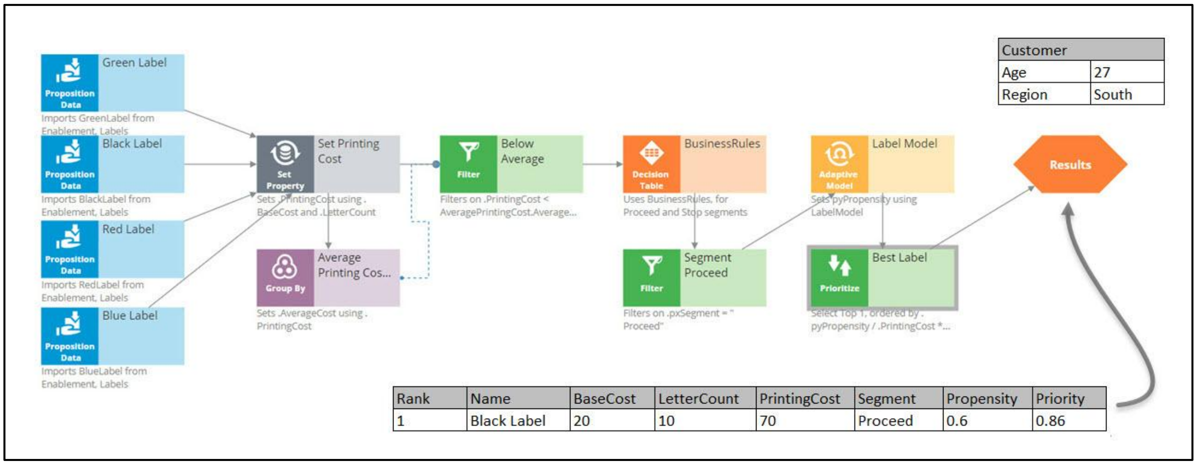
Because 0.86 is higher than 0.83, the Black Label action is now ranked number one.

So, even though the printing cost of the Black Label action is higher than that of the Green Label action, the Black Label action still comes out on top.

In this case, the Priority component reversed the Ranks of the two actions. Black is now the primary action and Green is the secondary action.

The same Prioritization component is also configured to output only the top action.

Therefore, it filters out the Green action altogether, and at the end of our strategy chain, the Black Label is left as our best action.



Customer	
Age	27
Region	South

Rank	Name	BaseCost	LetterCount	PrintingCost	Segment	Propensity	Priority
1	Black Label	20	10	70	Proceed	0.6	0.86

Defining prediction patterns

Description

Learn how to improve the predictive power of your adaptive models by configuring additional potential predictors in Pega Customer Decision Hub™. For example, you can make input fields that are not directly available in the customer data model accessible to the models by configuring these fields as parameterized predictors.

Gain experience using a prediction in a decision strategy and learn how to arbitrate between different groups of actions to display more relevant offers to customers.

Learning objectives

- Describe how to use predictions in a decision strategy.
- Configure parameterized predictors.
- Arbitrate between business issues by using applicability rules in Next-Best-Action Designer.

Creating parameterized predictors

Pega Customer Decision Hub™ uses Adaptive Models that use a wide range of predictors to optimize customer interactions. You can make input fields that are not directly available in the customer Data Model accessible to the models by configuring these fields as parameterized predictors. Learn how to create parameterized predictors for Expressions, offline model scores, and online model scores.

Transcript

This demo shows you how to create parameterized predictors to make input fields that are not directly available in the customer Data Model accessible to Adaptive Models. We will cover three examples: Expressions, offline model scores, and online model scores.

U+ Bank is cross-selling its credit cards on the web by using Pega Customer Decision Hub™. All customer data, including financial clickstream summary attributes, such as webpage visits, is available to the Adaptive Models that determine which offer to display for a particular customer.

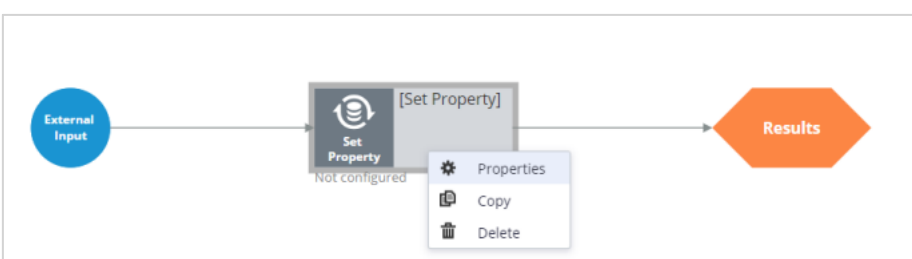
The first parametrized predictor is the ratio of two clickstream summary attributes that denote the number of visits to the U+ Bank Investment webpage in the last 30 days and the last 90 days. An increasing ratio might indicate growing customer interest.

The *Financial Services Clickstream* Data Set contains the clickstream data, which is a record of the user's activity on the website. These entries reflect the *Investment* webpage visits of the customer.

Investment page visits last 1 hour	Investment page visits last 30 days	Investment page visits last 7 days	Investment page visits last 90 days
0.0	0.0	0.0	0.0
2.0	2.0	2.0	2.0

To populate a parameterized predictor, you configure the *NBA Pre-Processing Extension Point* Decision strategy that is included in a change request. The pre-processing strategy runs before the Prediction strategy so that the output of the extension strategy is available as input to the Prediction.

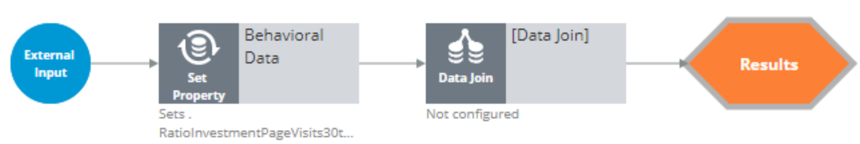
The first example of a parameterized predictor is an Expression that references two clickstream fields. To add a field, you add a Set Property component to the strategy and create a new property in the *Strategy Results* class. Set the property type to **Decimal** to store the ratio of customer visits to a web page in the last 30 days and in the last 90 days.



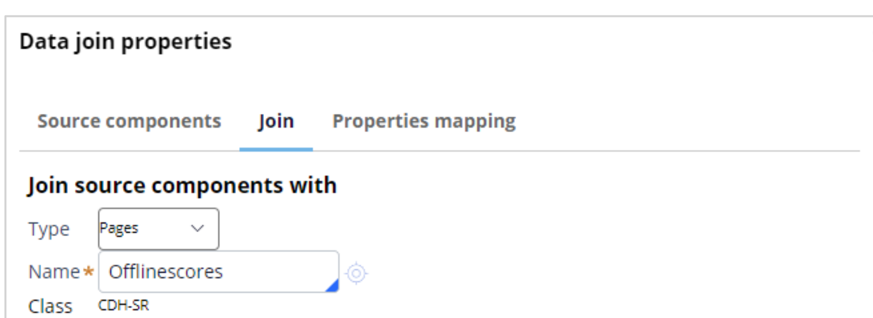
This Expression returns a value of zero when the number of Investment page visits in the last 90 days is zero. Otherwise, it returns the ratio of the Investment page visits in the last 30 days to the last 90 days. A high value might indicate the increasing interest of the customer in the content of this page.

The second example of a parameterized predictor is offline model scores. A Data Set that is associated with the customer context makes the data available as a data source. Each customer can have multiple scores for different categories or different channels.

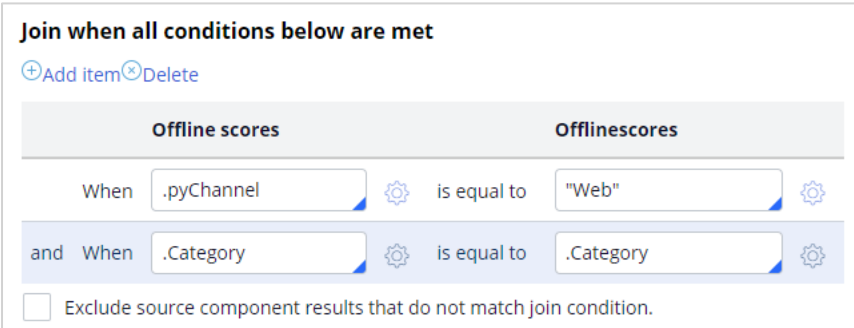
To add the model scores as potential predictors to the Adaptive Models, you add a Data Join component to the extension strategy.



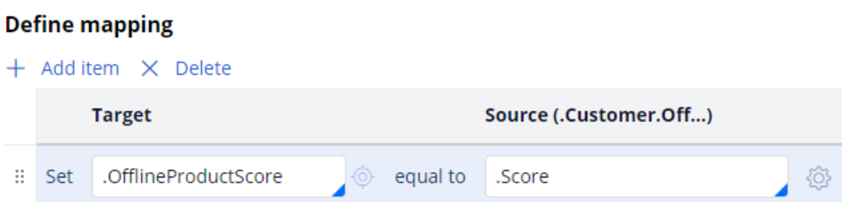
The component joins the *Offlinescores* page that contains the model scores to the available data.



To output only relevant scores, you specify the channel and the product category to match the Prediction.



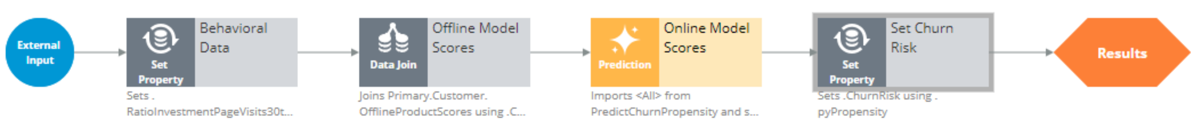
You then create a Decimal property and map it to accommodate the model scores.



The third example of a parameterized predictor is the score of a churn Prediction that runs in Customer Decision Hub. The Data Scientist team develops Predictive Models based on the latest insights and data that drive the churn Prediction.

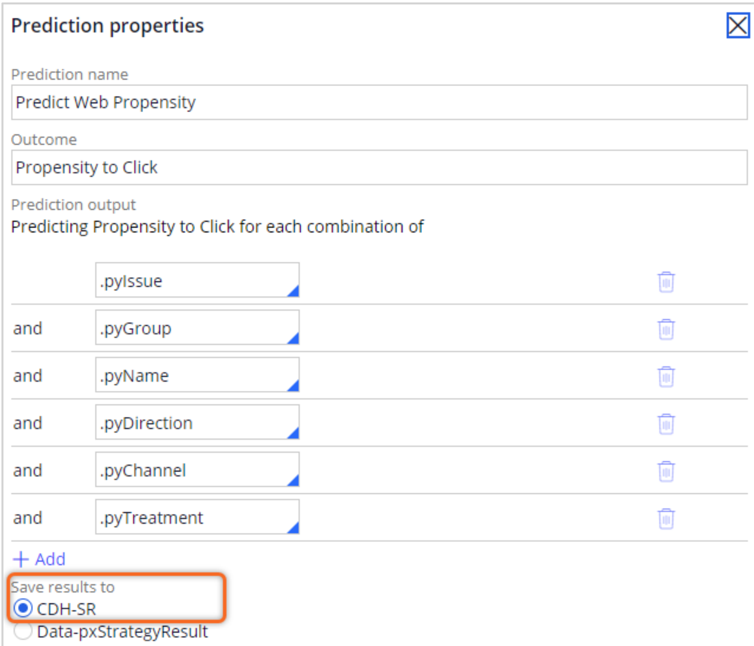
To make the churn score available to the Adaptive Models as a parameterized predictor, add a Prediction component to the extension strategy. Reference the churn Prediction in the *Customer* class.

Add a Set Property component to the extension strategy to create a new property and map it to the propensity calculated by the churn Prediction.

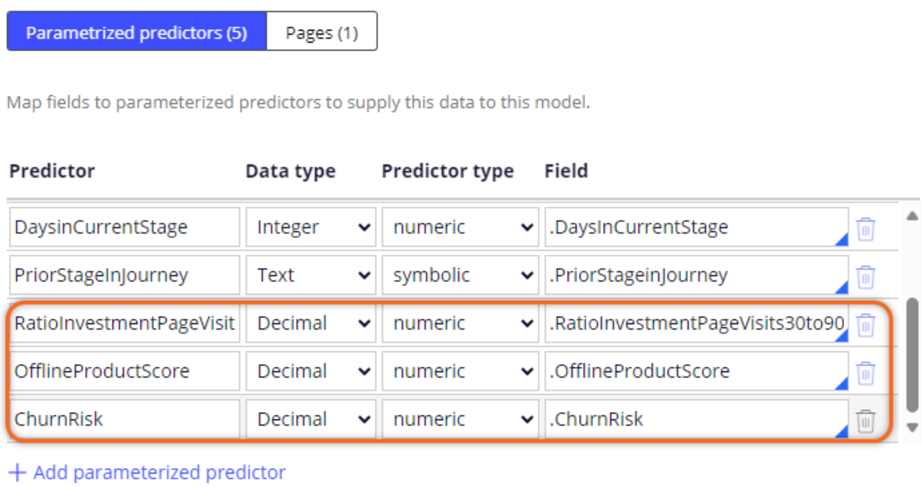


Run the strategy by using a test customer profile to verify that the strategy outputs include the real-time churn risk score, the offline product scores, and the Expression.

Next, you use Prediction Studio to map three new parameterized predictors in the Prediction. The Predict Web Propensity Prediction calculates the likelihood that a customer clicks on a web banner. Like the new properties, the system saves the Prediction results to the *CDH-SR* class.



You then create three new parameterized predictors and map them to the new properties that represent customer behavior, offline scores, and online scores.



When you submit your work for deployment, the application saves all changes made in Predictions to a *Change prediction* change request.

After a review by the Team Lead, a Revision Manager deploys the changes as part of a revision.

You have reached the end of this video. You have learned:

- How to configure the pre-processing extension strategy for parameterized predictors.
- How to add properties as parameterized predictors to the Prediction.
- How to submit the changes for deployment.

Using predictions in engagement strategies

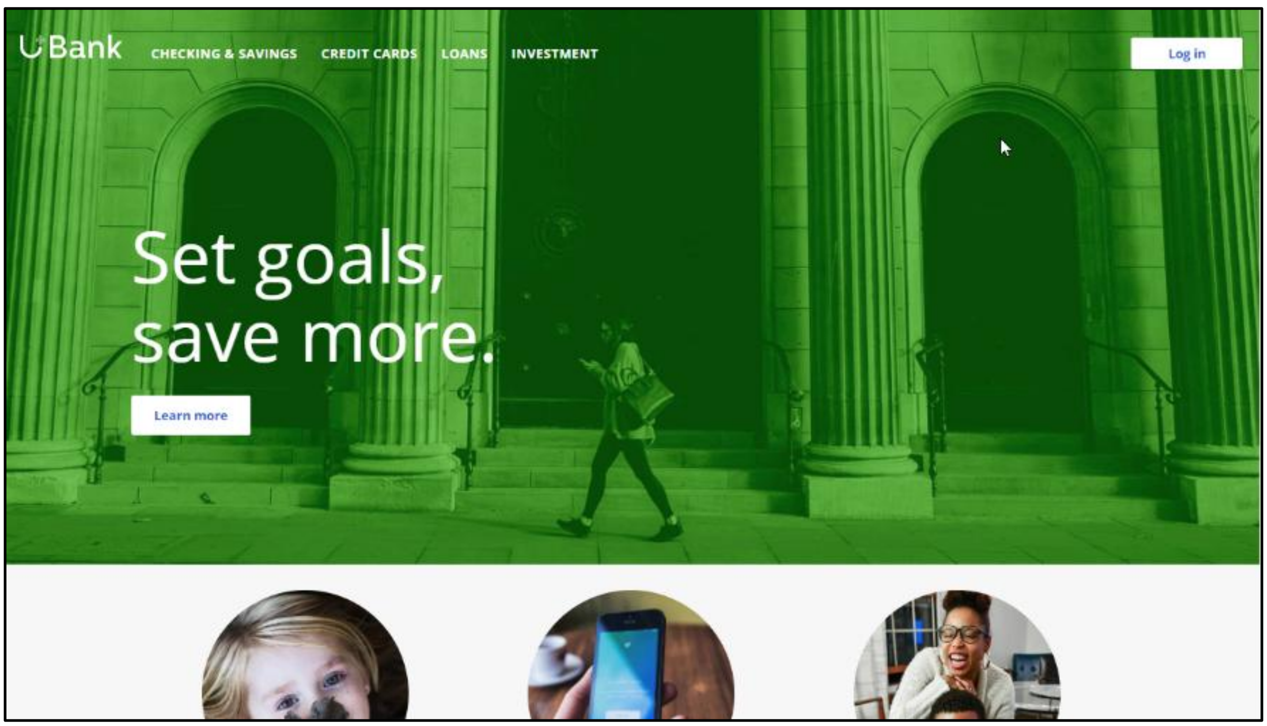
Introduction

A prediction is used to predict customer behavior such as offer acceptance and churn based on characteristics such as credit risk, income, and product subscriptions. Learn how to arbitrate between different groups of actions to display more relevant offers to customers. Gain experience using a prediction in a decision strategy and learn how applicability rules can be defined to reflect the bank's requirements in a decision strategy.

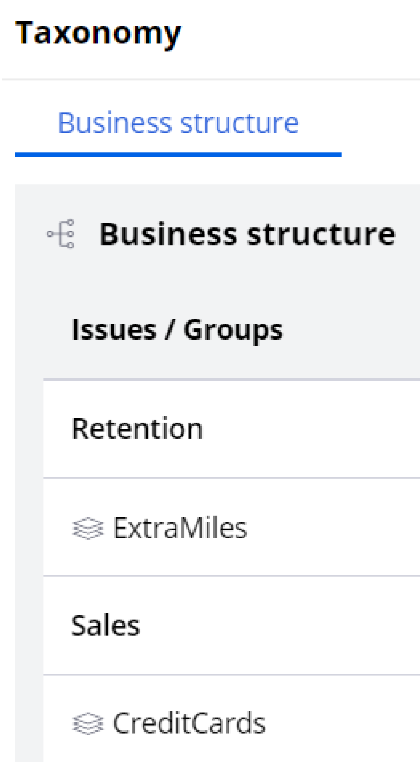
Transcript

This demo shows you how to use a prediction in an engagement strategy to determine customer applicability for a retention offer.

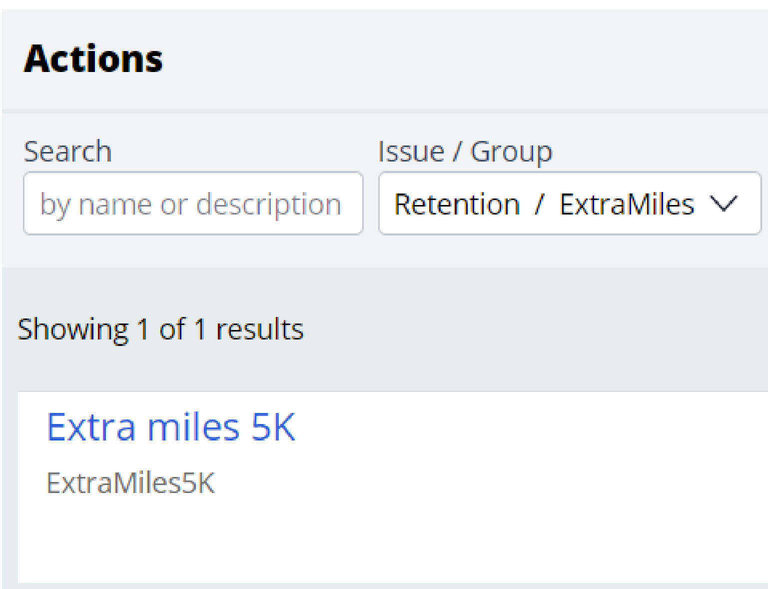
Currently, U+ Bank is cross-selling on the web by showing various credit cards to eligible customers who log in to its website. The bank now wants to show a retention offer, instead of a credit card offer, to customers who are likely to churn in the near future. The credit card offers are shown only to loyal customers.



To meet this business requirement, a decisioning administrator has already set up the taxonomy by defining a new business issue called Retention, and an offer group.

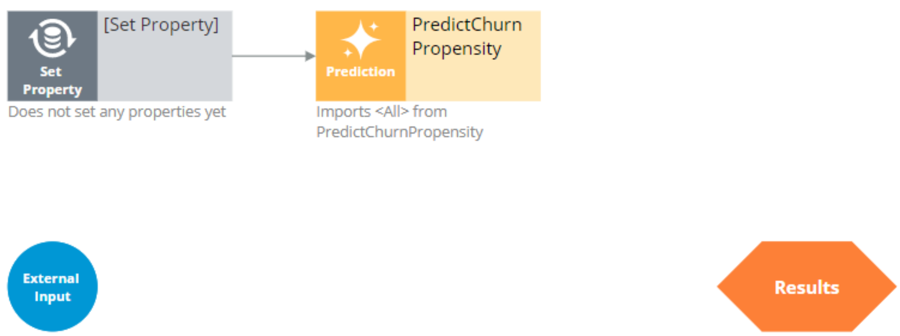


This ExtraMiles group contains a retention offer, Extra Miles 5K.

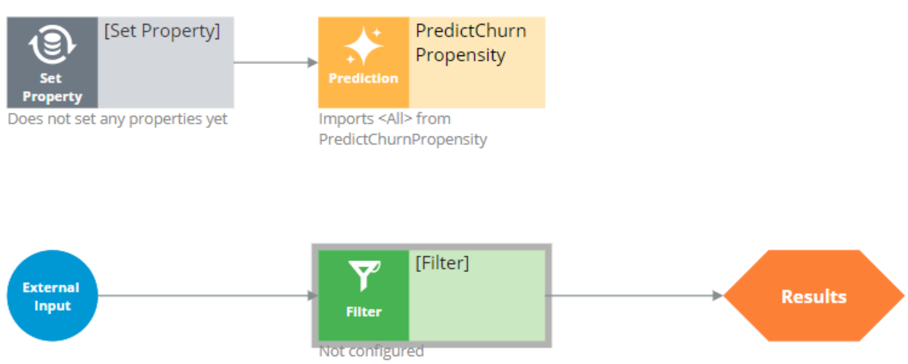


The next step is to create an applicability condition that makes a customer qualify for a retention offer when there is a high likelihood that the customer might churn. A data scientist has created a prediction that identifies these high-risk customers. When you open the prediction in Prediction Studio, notice that the possible response labels are **Churn** and **Loyal** to predict customer behavior. The result of the prediction is stored in the pxSegment property.

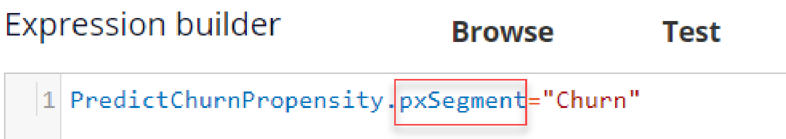
To define the applicability condition, you create a decision strategy to output a retention offer only if the response label of the prediction is **Churn**. Add a **Prediction** component to the canvas and configure it to reference the churn prediction. Add a **Set Property** component and connect it to the Prediction component. You can configure the Set Property component at a later point to accommodate parameterized fields.

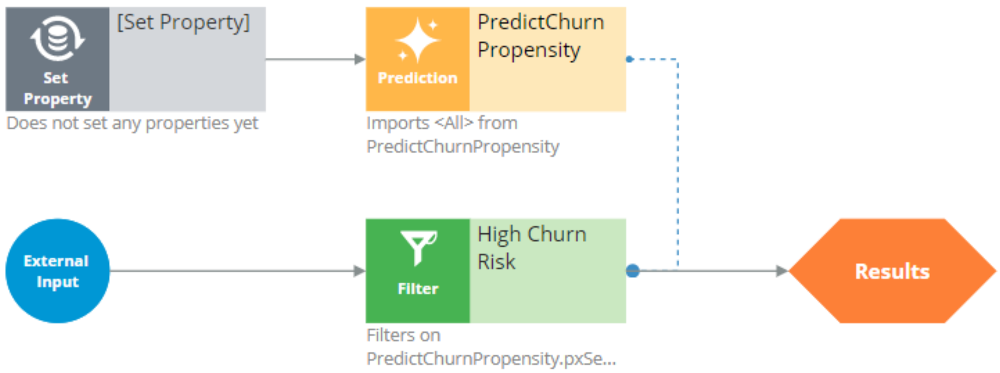


Next, add a filter component to filter out the loyal customers and pass retention offers to high churn risk customers only.

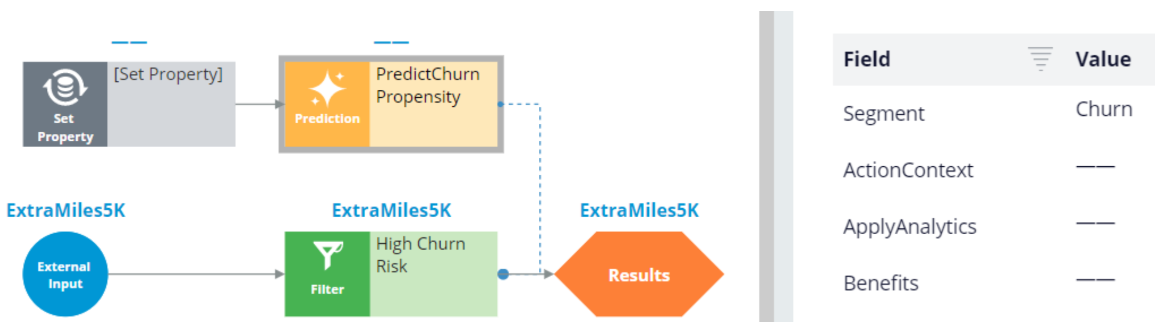


The filter condition is defined to output a retention offer when the pxSegment property of the prediction is equal to **Churn**.

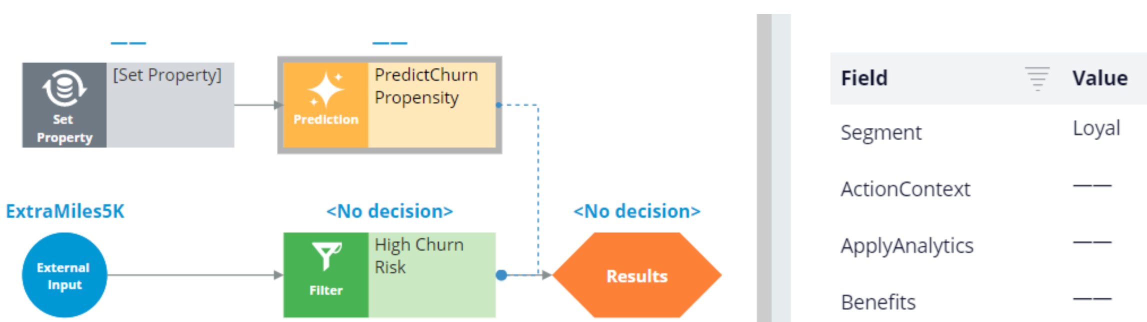




Next, test the strategy using two customer profiles, **Troy** and **Barbara**. For external inputs, consider all available retention offers. The strategy outputs a result for **Troy** because the result of the prediction is **Churn**.



The strategy does not have a result for **Barbara**, because the Segment value is **Loyal**.



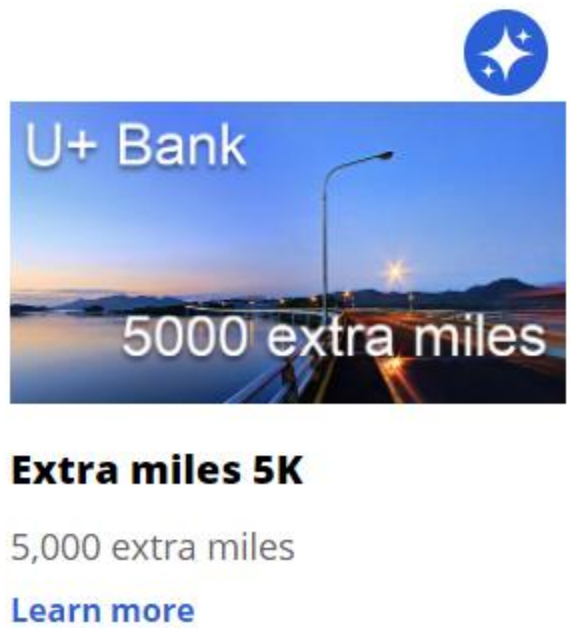
By checking in the strategy, you commit your changes so that they go into effect. You can now use this strategy in the Next-Best-Action Designer engagement policy as an applicability condition.

The first business rule you need to implement is that the **ExtraMiles** group is applicable only to high churn risk customers. To implement this rule, in the **Applicability** section, define a condition for the customer field. Select the **RetentionStrategy**. The condition is: the RetentionStrategy has results for the High Churn Risk component.

The second business rule you need to implement is: U+ Bank wants to show credit card offers to low-risk customers only; meaning the **CreditCards** group is not applicable for high-risk customers. To implement this rule, modify the **Applicability** section of the **CreditCards** group. The condition is: the RetentionStrategy doesn't have results for the High Churn Risk component.

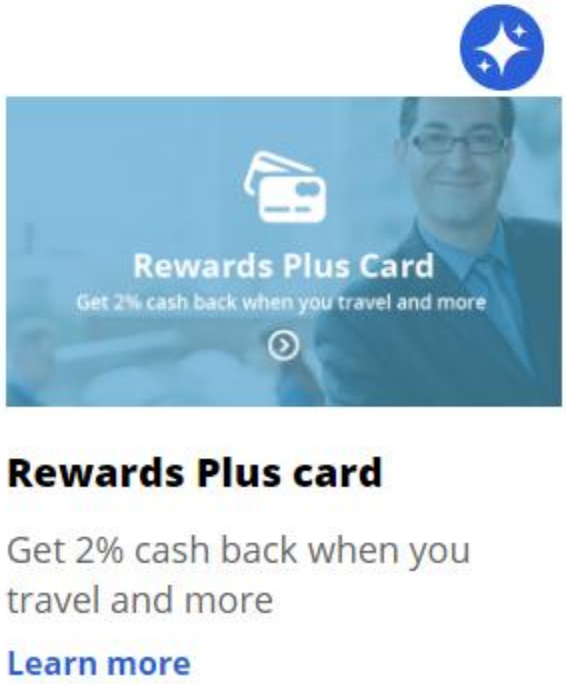
Once the applicability conditions are defined, you need to amend the **Channels** configuration. Because U+ Bank introduced a new group, **ExtraMiles**, which belongs to a new business issue, **Retention**, you need to select the results from the appropriate business structure level. In this case, the bank wants to arbitrate between two different business issues: Sales and Retention. Therefore, select All Issues/All Groups from the business structure level. Saving the configuration implements the business requirement.

On the U+ Bank website, when you log in as **Troy**, notice that the retention offer is displayed because



Troy is predicted to churn in the near future.

Now, when you log in as **Barbara**, notice that the credit card offer is displayed because she is predicted to remain loyal for now.



You have reached the end of this demo. What did it show you?

- How to use a prediction in a decision strategy
- How to arbitrate between different groups of actions to display more relevant offers to customers
- How to define applicability rules using a decision strategy in Next-Best-Action Designer

Model governance

Description

AI has the potential to deliver significant benefits, but improper controls can result in regulatory issues, public relations problems, and liability. The Pega T-Switch™ settings in Prediction Studio, which define the transparency thresholds for business issues, help companies to enable users to deploy AI algorithms responsibly and safely.

U+ Bank uses Pega Customer Decision Hub™ to personalize the offers that customers see when they log in to the U+ Bank website.

Customer Decision Hub provides tools to give local and global explanations for the behavior of the adaptive model, such as Customer Profile Viewer for local explanations and the ADM predictor importance report for global explanations. An ethical bias simulation enables you to test your engagement policies for unwanted bias.

Learning objectives

- Explain how Prediction Studio reflects the company's model transparency policy
- Provide local explanations in the Customer Profile Viewer for a single propensity calculation of an adaptive model
- Inspect the ADM predictor importance report to provide global explanations of the factors that drive predictions
- Explain the function of an ethical bias simulation
- Detect unwanted bias in engagement policies

Model transparency

AI has the potential to deliver significant benefits, but without proper controls, it can lead to regulatory issues, public relations problems, and liability.

In Prediction Studio, a senior Data Scientist can set the transparency thresholds within their business to enable users to deploy AI algorithms responsibly and safely. The Pega T-Switch™ settings help companies mitigate potential risks, maintain regulatory compliance, and responsibly provide differentiated experiences to their customers.

Transcript

Companies in various industries use AI to make predictions and decisions based on those predictions.

To use AI models responsibly, it's important to make sure that the decisions they make can be easily understood by the people who are impacted by them.

Responsible use is especially critical in industries such as finance and healthcare, where AI-driven decisions can have a significant impact on customers. Consumers and regulators demand a high level of trust and transparency in the models that drives these impactful decisions.

Predictive models can be either opaque, making it difficult or impossible to understand how a model reached a decision, or more transparent.

Various factors, such as the complexity and interpretability of the model determine the transparency of a model type.

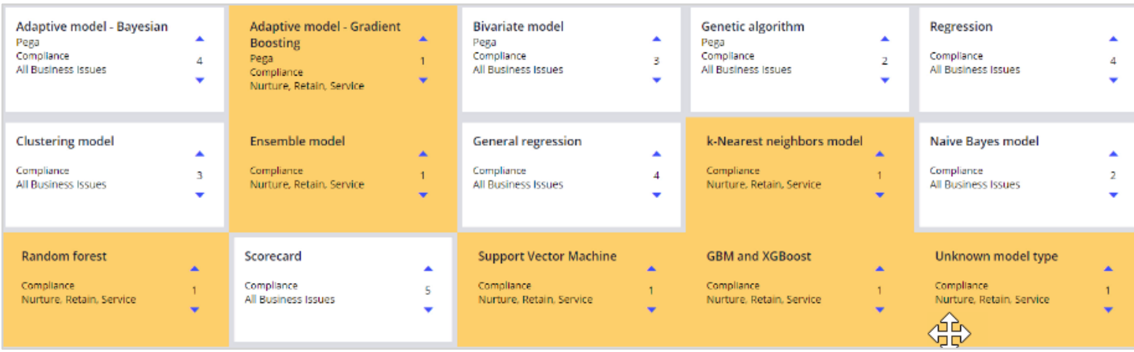
Pega Customer Decision Hub™ includes the tools with which companies can employ AI models with confidence in their implementations and avoid ethical and regulatory issues. Prediction Studio has a dedicated page for model transparency.

Each model type that Pega Platform™ includes receives a recommended transparency score on a scale of 1 to 5, with 1 being the least transparent and 5 being the most transparent. For example, linear regression models are considered more transparent than neural network models because they are simpler and easier to interpret. The adaptive Bayesian models that drive the out-of-the-box Customer Decision Hub predictions are highly transparent as they include model monitoring, reporting, and propensity explanations.

The required transparency may differ between business issues. Marketing might call for the use of more complex models. Still, when dealing with a business issue such as collections or a risk assessment, decisions must be highly explainable to both regulators and customers.

By default, all models are allowed for all business issues. In the development phase of a Pega Customer Decision Hub project, the project lead coordinates with stakeholders to collect the requirements for the transparency policy.

In Prediction Studio, a senior Data Scientist then implements the transparency policy by adjusting the business issue thresholds. Depending on the company policy, model techniques are marked as compliant or non-compliant for a specific business issue.



You have reached the end of this video. You have learned:

- How each model type that comes with Pega Platform has a recommended transparency score.
- How Prediction Studio reflects the model transparency policy of the company.

Explainable AI

U+ Bank uses Pega Customer Decision Hub™ to personalize the offers that customers see when they log in to the U+ Bank website. The adaptive models that calculate the propensity that a customer clicks on the offer are highly transparent. Customer Decision Hub includes tools that give local and global explanations for the adaptive model behavior.

Local explanations enhance the ability to understand the reasoning behind a single propensity calculation made by the model. Customer Profile Viewer offers local explanations of the decisions made by Customer Decision Hub, including the propensities calculated by the adaptive models that drive these decisions.

Global explanations refer to predictor importance in an adaptive model. Customer Decision Hub has built-in support to calculate predictor importance by using decision trees and random forests.

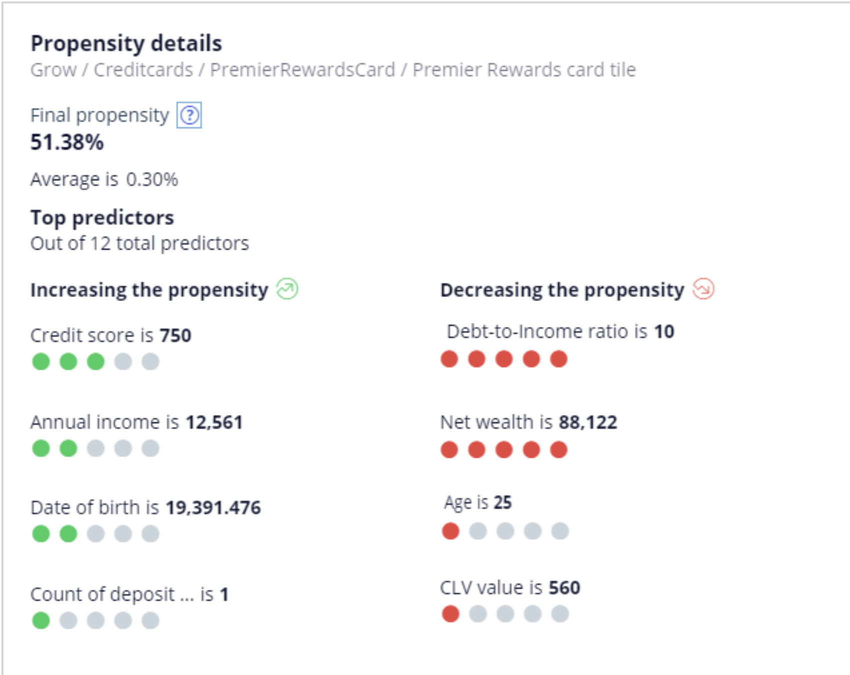
Transcript

Local and global explanations support transparency in how predictive models calculate propensity. Local explanations enable understanding the rationale behind a single propensity calculation made by a predictive model, and they focus on understanding how the input predictors influence that single calculation. Global explanations focus on the overall behavior of a model and provide an understanding of which predictors are most important across all customers. Both local and global explanations are important to ensure that the models behave as expected and to build trust with customers and regulators in the use of AI.

U+ Bank uses Pega Customer Decision Hub to optimize customer interactions on the bank's website, and the Customer Profile Viewer report provides local explanations of the propensities calculated by the out-of-the-box adaptive models. You can review basic customer information for Joanna, including key indicators and demographic data. You can review recent interactions with Joanna in the timeline and how she responded to the decisions made by Customer Decision Hub. You can filter the interactions by the outcome and time frame.

Customer Profile Viewer also shows the current prioritization of next-best-action results. For the U+ Bank website channel, the direction is inbound, and TopOffers is the real-time container that connects with the website. Joanna is eligible for two credit card offers as determined by engagement policies and constraints. The adaptive models calculate the propensities for these offers. The final propensity used in the arbitration formula may differ from the model propensity, for example, when the customer is part of a model control group or when the adaptive model is still in the initial learning phase.

For example, U+ Bank can explain the decision to offer customer Joanna the offer with the highest propensity, the RewardsPlus credit card. The credit score and annual income predictors increase the propensity for Joanna to click on the personalized credit card offer. In contrast, the debt-to-income ratio and net wealth predictors decrease it. This explanation is valid for the RewardsPlus credit card offer to Joanna at this point in time.



In machine learning, predictor importance refers to the relative importance of each predictor in predicting the target variable. In Prediction Studio, the ADM predictor importance report lists the predictor importance score for each predictor, which indicates their relative importance in the model. By analyzing predictor importance, you can offer global explanations of the factors that drive the predictions made by the adaptive models. For example, when you select the Age predictor, you see that the importance of this predictor is relatively high in the adaptive model for the Premier Rewards credit card offer. You can export the report for further analysis.

You have reached the end of this video. You have learned:

- How Customer Profile Viewer provides data on past interactions with a customer.
- How Customer Profile Viewer provides local explanations for the decisions that Customer Decision Hub makes for a customer.
- How Prediction Studio supports global explanations of the factors that drive predictions.

Ethical bias

An ethical bias simulation enables you to test your engagement policies for unwanted bias. That is, you can check if your conditions discriminate based on age, gender, ethnicity, or any other attributes specific to your business scenario. This is particularly useful during the testing stage, while developing and changing actions.

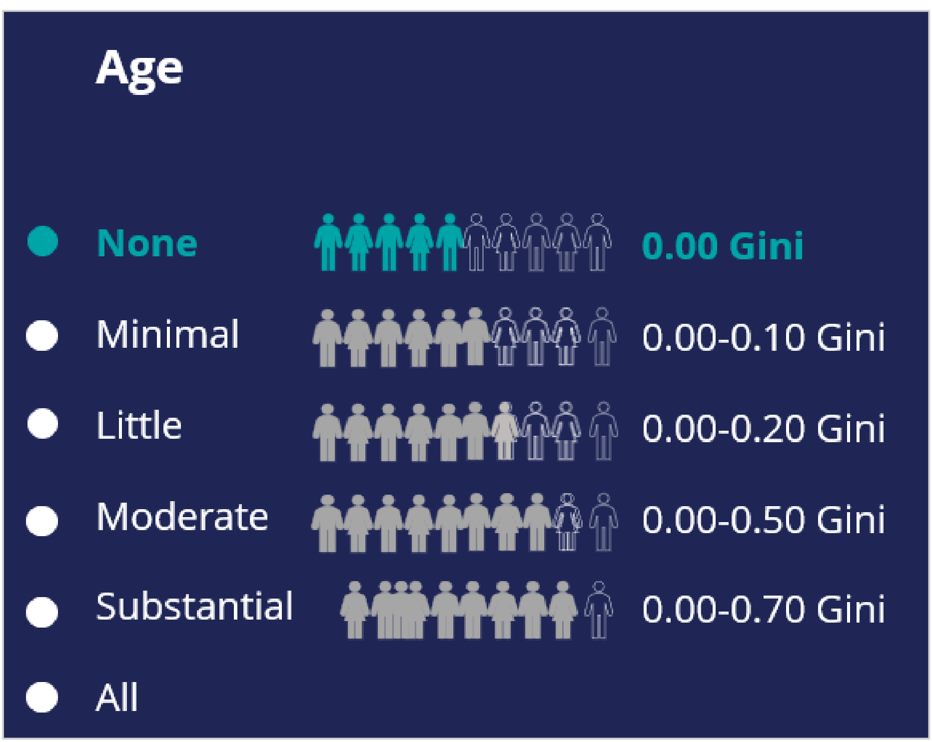
Transcript

This video explains what ethical bias is and how you can avoid unwanted bias in your engagement policies.

Ethical bias testing helps check your engagement policies for unwanted bias. That is, you can test if your conditions discriminate based on age, gender, ethnicity or any other attributes specific to your business scenario.

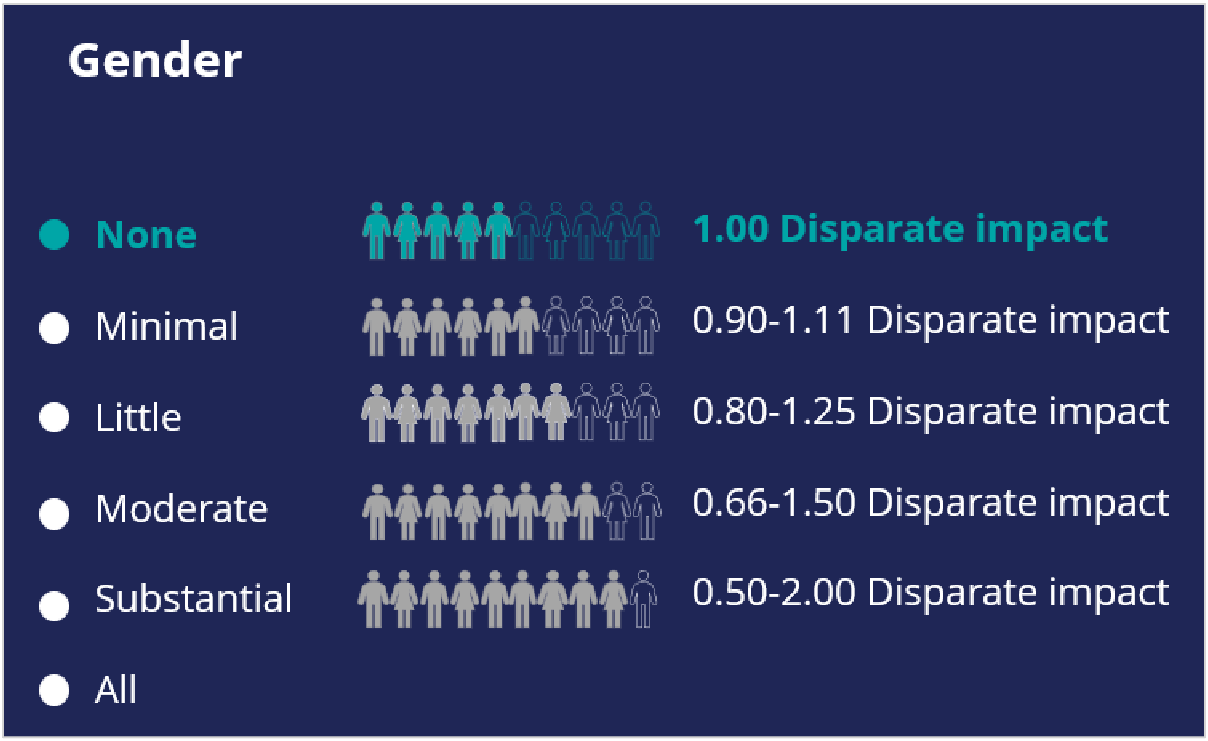
An ethical bias policy forms the base of an ethical bias simulation. This policy will include the bias fields and the thresholds for each field. You can include any property from your customer class. For instance, age and gender properties are typical properties you might want to include in bias testing.

As age is a numerical field, a Gini coefficient is used to calculate the bias. This is a method of measuring the statistical inequality of a value distribution. A Gini coefficient of 0 represents perfect distribution equality. You can select a warning threshold between 0 (warn if any bias is detected) and 0.7 (warn only if very high bias is detected). You can also choose not to check for bias within a particular business issue.



As Gender is a nominal value, a disparate impact is used to determine bias. A disparate impact is used to calculate bias for categorical fields by comparing the number of customers who were selected for an action to those not selected for an action, and then correlating the result to the selected bias field.

A disparate impact of 1 represents perfect distribution equality. You can select a warning threshold between 1 (warn if any bias is detected) and 0.50 - 2.00 (warn only if very high bias is detected). You can also choose to ignore this bias field for a particular issue in your business structure.



Even though no bias is ideal, it is not practical to set the threshold at that level as that would block any actions from reaching customers. Hence, in real business scenarios, a little bias is often allowed to prevent customers from receiving no actions.

Detecting unwanted bias

Ethical bias testing checks your engagement policies for unwanted bias. That is, you can test if your conditions discriminate based on age, gender, ethnicity, or any other attributes specific to your business scenario.

Transcript

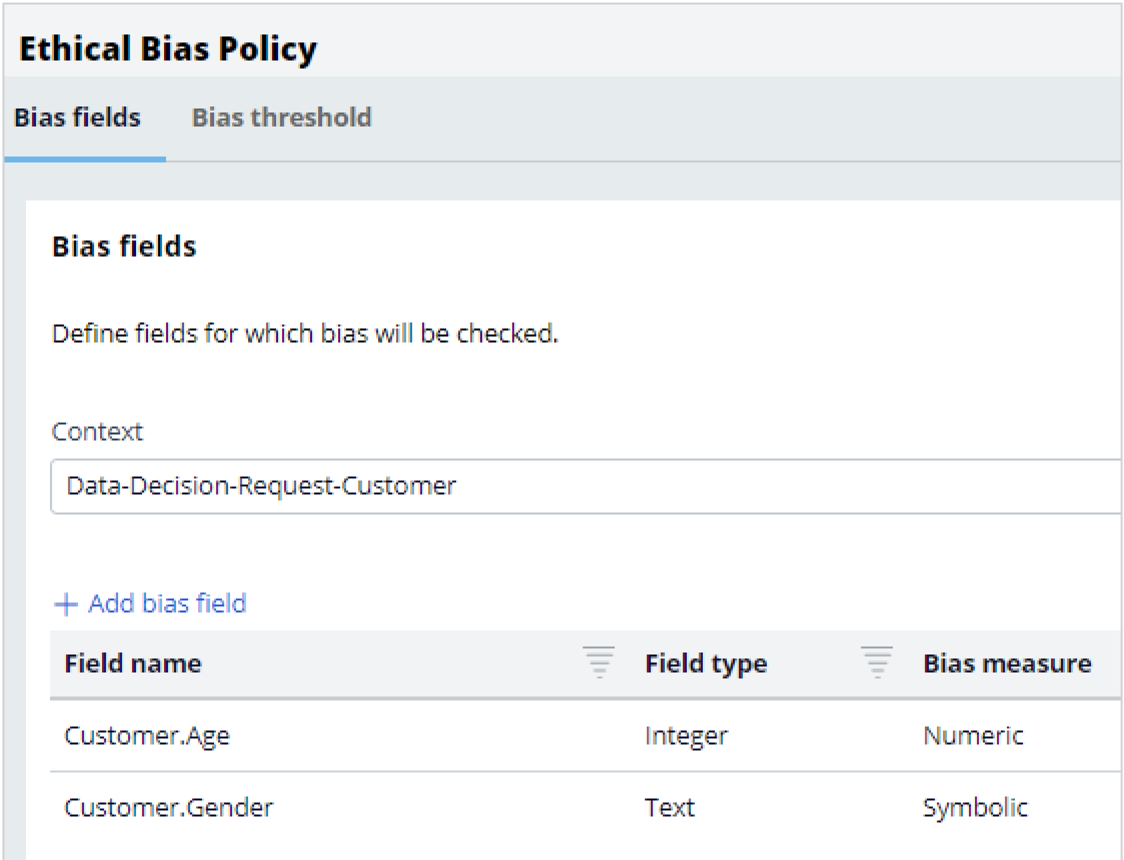
This demo will show you how to run an ethical bias simulation and identify any unwanted bias in the engagement policy conditions.

U+, a retail bank, recently updated their engagement policy conditions to present credit card offers to qualified customers. Now they would like to run an ethical bias simulation to ensure there is no unwanted bias based on age or gender before they push the changes to the live production environment.

This is the Pega Customer Decision Hub™ portal.

To create a simulation, first configure the ethical bias policy. This policy will include the bias fields and threshold. You can select any property from your customer class. In this case, you will use the age and gender properties for the test, which are the most commonly used properties for bias testing. The age property value is a number. In the **Add bias** field window, add age as a bias field.

Next, add gender as a bias field.



Now, on the **Bias threshold** tab, review and configure the bias threshold settings for each property you selected. This configuration is done at the business issue level. This means, for every business issue, you can decide how much bias to allow. For instance, for the risk issue, you might want to disallow any bias, whereas for grow, you might allow some bias, if you designed a special offer for a specific age category or a specific gender. The bias threshold measurement depends on the type of field that you select. By default, the threshold is set for no detection for these fields.

In this case, the bank does not want to discriminate on age, so select the appropriate threshold value.

The bank also does not want to discriminate on gender, so select the appropriate threshold value. Save the changes.

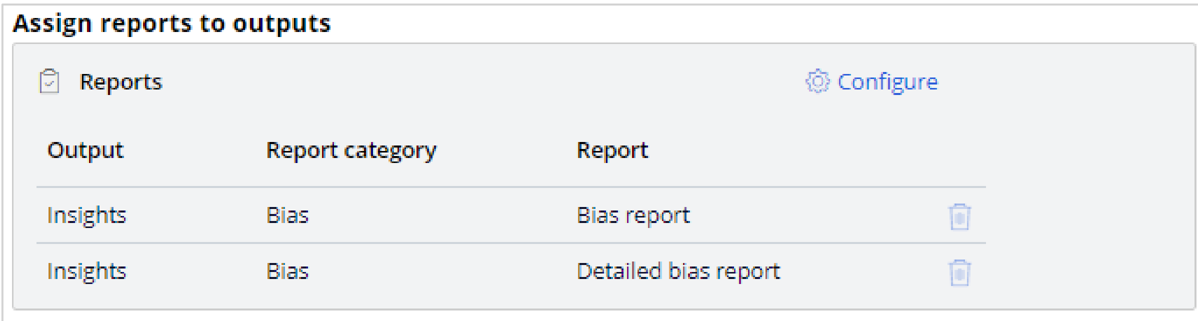
Navigate to the **Simulations** landing page to create and run an **Ethical bias** simulation.

On the Simulation Creation page, select the top-level strategy on which you would like to run the simulation. Then, select the input population on which you want to execute the simulation.

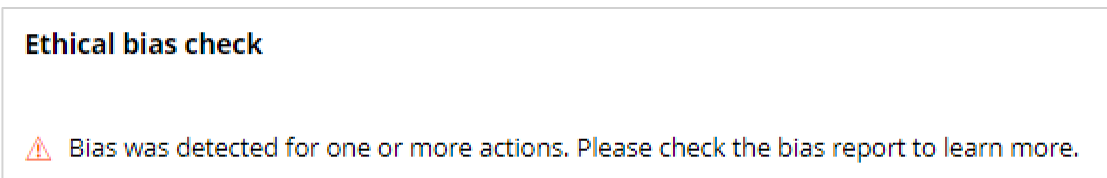
Now, rename the simulation. This will help you easily identify specific simulation runs.

Note that the simulation results are output to the **Insights** data set. This is an internal dataset where the results will be available for an hour by default.

The ethical bias simulation has two bias reports automatically available as output. The results can be examined post simulation run.



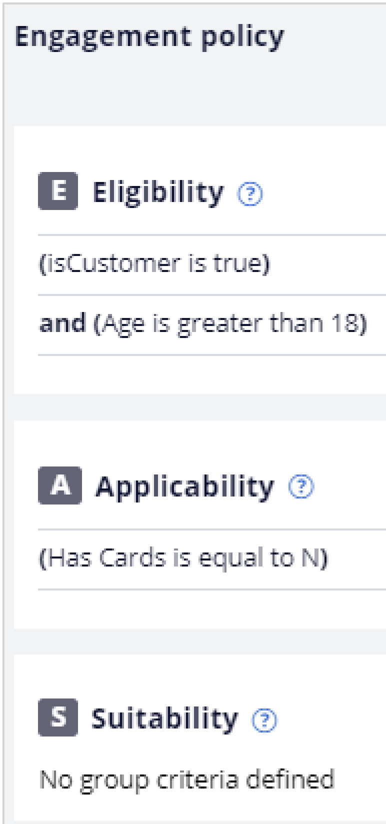
Run the simulation. Once the run is complete, there will be an indication if any unwanted bias is identified. In this scenario, although business did not want any bias, it seems some bias has been detected.



Open the generated reports and view the information in detail to understand where the bias was detected. You can sort the report in the bias detected column. Notice that there is bias detected on age. You can check the **Bias value** against the **Confidence interval** to check if the bias value is within the bias threshold range.

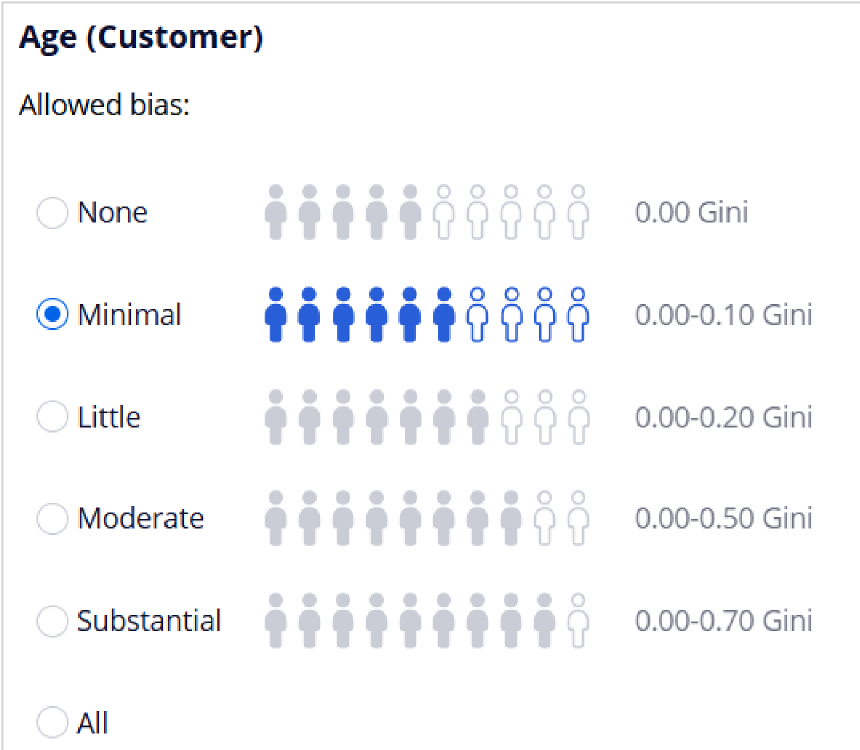
Bias detected ↓	Issue	Group	Action	Bias field	Category	Bias measure	Bias value	Confidence interval	Bias threshold
true	Grow	Creditcards	PremierRewardsCard	.Customer.Age	--	Gini coefficient	0.15	0.05 - 0.36	0.0
true	Grow	Creditcards	RewardsCard	.Customer.Age	--	Gini coefficient	0.22	0.09 - 0.44	0.0
true	Grow	Creditcards	RewardsPlusCard	.Customer.Age	--	Gini coefficient	0.15	0.05 - 0.36	0.0
true	Grow	Creditcards	StandardCard	.Customer.Age	--	Gini coefficient	0.22	0.09 - 0.44	0.0

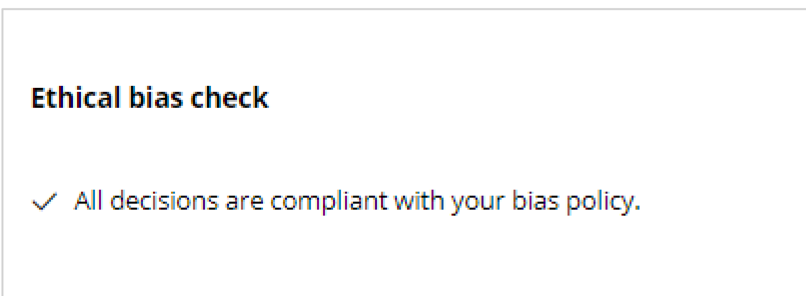
Navigate to Next-Best-Action Designer to view the engagement policy conditions. Note that there is an eligibility condition that uses age. Credit cards are valid only for customers with an age greater than 18. This is a hard eligibility rule that cannot be ignored. It is likely that bias has been detected due to this condition.



Based on the bank’s policies and regulations, you must review the bias threshold and decide how much bias the business will allow.

To modify the policy, open the Ethical Bias policy. Due to the eligibility condition, which cannot be ignored, business decided to allow a maximum bias of 0.1 Gini. So, increase the bias threshold, and re-run the same ethical bias simulation.





Since the age bias threshold was increased, no bias should be detected.

Bias detected	Issue[1] ↑	Group[2] ↑	Action[3] ↑	Bias field[4] ↑	Category	Bias measure	Bias value	Confidence interval	Bias threshold
false	Grow	Creditcards	PremierRewardsCard	.Customer.Age	--	Gini coefficient	0.15	0.05 - 0.36	0.0 - 0.1
false	Grow	Creditcards	PremierRewardsCard	.Customer.Gender	F	Rate ratio	1.17	0.77 - 1.78	1.0
false	Grow	Creditcards	PremierRewardsCard	.Customer.Gender	M	Rate ratio	0.84	0.54 - 1.31	1.0
false	Grow	Creditcards	PremierRewardsCard	.Customer.Gender	U	Rate ratio	1.02	0.67 - 1.56	1.0
false	Grow	Creditcards	RewardsCard	.Customer.Age	--	Gini coefficient	0.22	0.09 - 0.44	0.0 - 0.1
false	Grow	Creditcards	RewardsCard	.Customer.Gender	F	Rate ratio	1.06	0.58 - 1.95	1.0
false	Grow	Creditcards	RewardsCard	.Customer.Gender	M	Rate ratio	1.16	0.64 - 2.07	1.0
false	Grow	Creditcards	RewardsCard	.Customer.Gender	U	Rate ratio	0.82	0.44 - 1.52	1.0
false	Grow	Creditcards	RewardsPlusCard	.Customer.Age	--	Gini coefficient	0.15	0.05 - 0.36	0.0 - 0.1
false	Grow	Creditcards	RewardsPlusCard	.Customer.Gender	F	Rate ratio	1.17	0.77 - 1.78	1.0
false	Grow	Creditcards	RewardsPlusCard	.Customer.Gender	M	Rate ratio	0.84	0.54 - 1.31	1.0
false	Grow	Creditcards	RewardsPlusCard	.Customer.Gender	U	Rate ratio	1.02	0.67 - 1.56	1.0

Now, let's run an ethical bias simulation at the **Trigger_NBA_Sales_CreditCards** strategy level. This strategy includes engagement policy conditions, arbitration, adaptive analytics, constraints, and treatments and channels processing.

Configure inputs

Strategy Configure

Trigger_NBA_Sales_CreditCards

Context

Data-Decision-Request-Customer

Results in

CRM-SR-Sales-CreditCards

Audience Configure

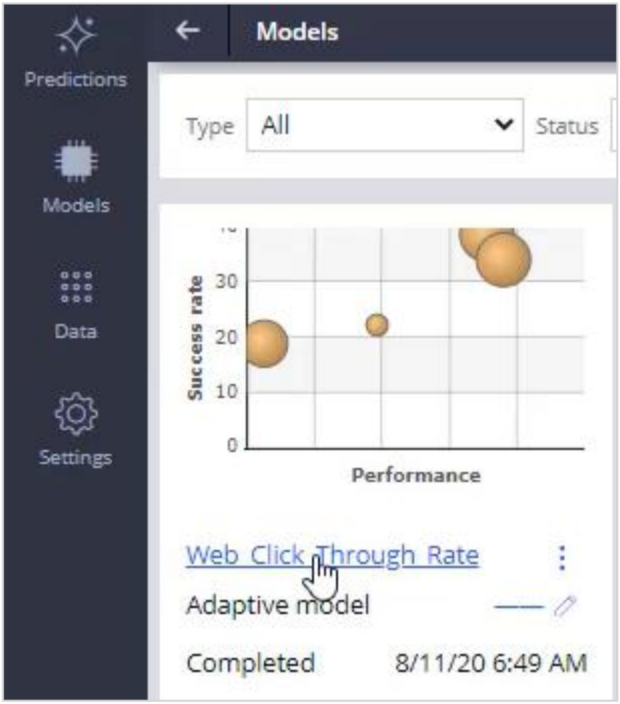
Sampled Customers

Data set

Note that bias is detected on gender. However, it is at the CreditCards group level, with only engagement policies in scope, thus, there was no bias.

Bias detected	Issue[1] ↑	Group[2] ↑	Action[3] ↑	Bias field[4] ↑	Category
false	Sales	CreditCards	PremierRewardsCard	.Customer.Age	--
true	Sales	CreditCards	PremierRewardsCard	.Customer.Gender	F
true	Sales	CreditCards	PremierRewardsCard	.Customer.Gender	M
false	Sales	CreditCards	RewardsCard	.Customer.Age	--
false	Sales	CreditCards	RewardsCard	.Customer.Gender	F
false	Sales	CreditCards	RewardsCard	.Customer.Gender	M
false	Sales	CreditCards	RewardsPlusCard	.Customer.Age	--
true	Sales	CreditCards	RewardsPlusCard	.Customer.Gender	F
true	Sales	CreditCards	RewardsPlusCard	.Customer.Gender	M
false	Sales	CreditCards	StandardCard	.Customer.Age	--
false	Sales	CreditCards	StandardCard	.Customer.Gender	F
false	Sales	CreditCards	StandardCard	.Customer.Gender	M

Navigate to **Prediction studio** to view the **Web Click Through Rate** adaptive model.

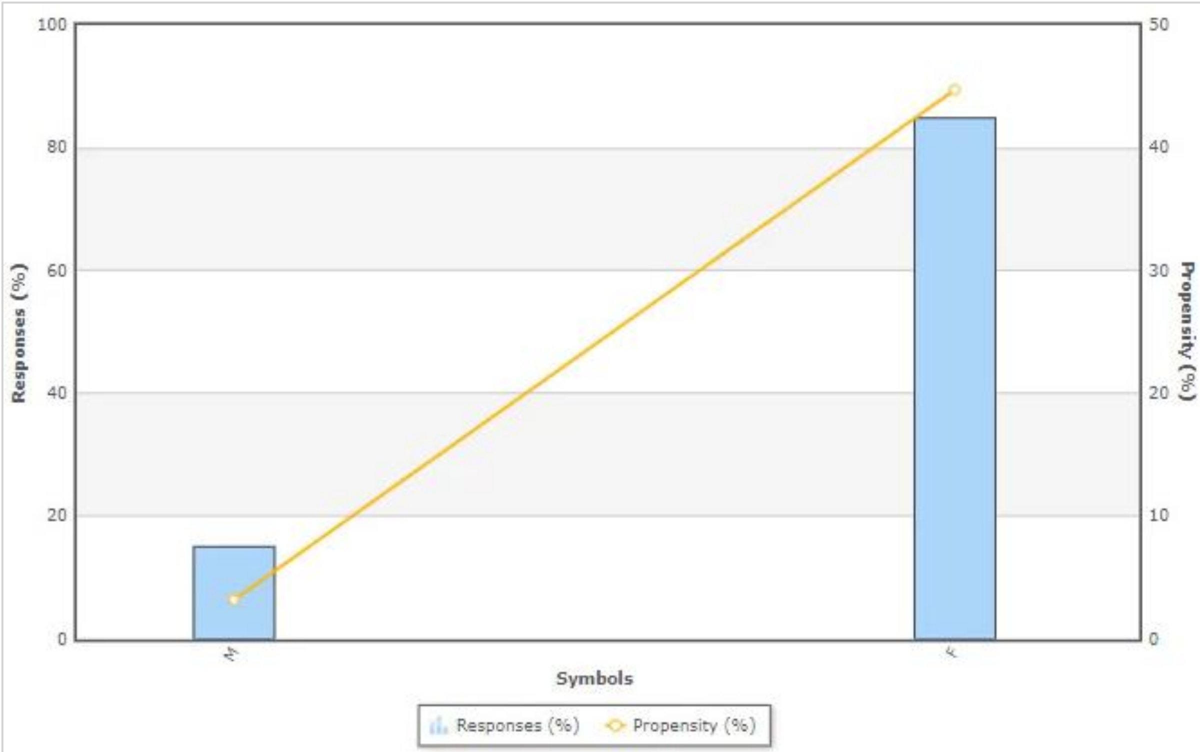


Now, open the Premier rewards card model report to view the predictors.

Predictors | Score distribution | Trend

Predictors	Status	Type	Performance (AUC)	Range/Symbols(#)	Bins(#)
Customer.AverageSpent	Active	Numeric	76.84	[1001.42; 1999.4]	12
Customer.Age	Active	Numeric	69.58	[19.0; 80.0]	9
Customer.Gender	Active	Symbolic	61.24	2.00	2

Notice that gender is one of the top predictors. As gender is a predictor, and the model is learning constantly, bias is detected even though there is no engagement policy condition on gender. Open the predictor to view the customer responses. Here you can see that there is a clear difference in the number of times the offer was presented to male customers versus female customers.



If you do not want the adaptive model to learn based on gender, ensure gender is not included as a predictor.

This demo has concluded. What did it show you?

- How to configure an ethical bias policy.
- How to run an ethical bias simulation.
- How to view the auto-generated ethical bias reports to understand the bias threshold deviation.

Pega Process AI overview

Description

Gain a greater understanding of the key features, capabilities, and benefits of Prediction Studio in Pega Process AI™ context. Prediction Studio is the dedicated workspace for data scientists to control the life cycles of predictions and the predictive models that drive them. Configure the predictions that are deployed in Pega Process AI to increase efficiency and effectiveness in case management.

Learning objectives

- Describe the use of Pega Process AI in case management
- Explain the types of predictions that are available in Prediction Studio

Pega Process AI overview

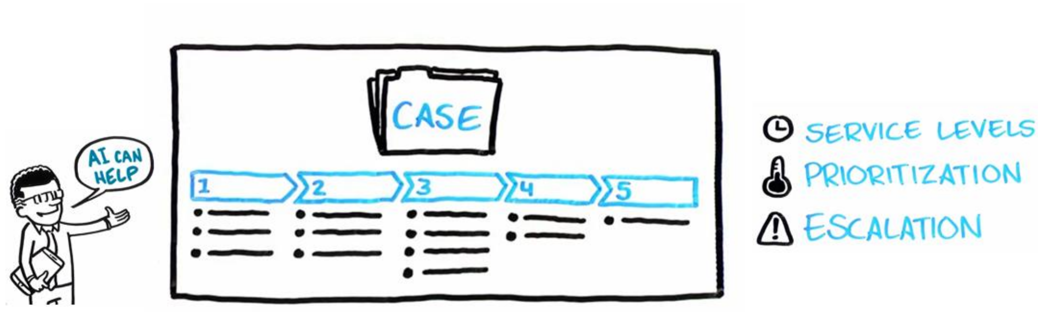
Introduction

In recent years, artificial intelligence has moved out of the labs and helped enterprises generate proven business value. At the same time, operationalizing AI can be a bottleneck. Pega Process AI™ tackles this problem by using AI to self-optimize processes and applying your own AI in Pega case management.

Transcript

This video provides an overview of the Pega Process AI capabilities in intelligent automation.

Process management aims to optimize business processes by increasing efficiency, consistency, and transparency, which decreases costs and improves quality.



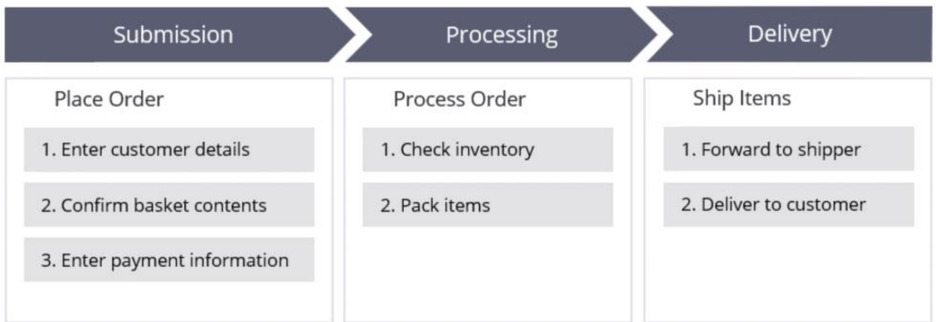
For example, consider an online order process. The customer submits an order, and the company processes and then delivers the order.

A Pega Platform application that models the online order process follows the same sequence as a series of stages. A **case type** is the abstract model of that process.

Case types model repeatable business transactions that might refer to a customer, or another entity, such as a machine in a maintenance case type.

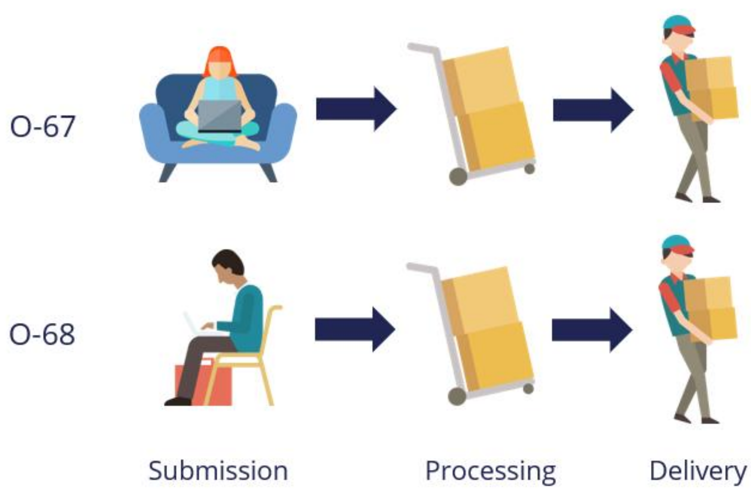
The **case life cycle** for a case type helps to visualize the work to complete as part of a business transaction.

Each stage in the life cycle contains the steps required to complete it and move to the next stage.



A **case** is a specific transaction instance of the case type.

Each time a user submits an online order, Pega Platform creates an order case and assigns the case a unique identifier.

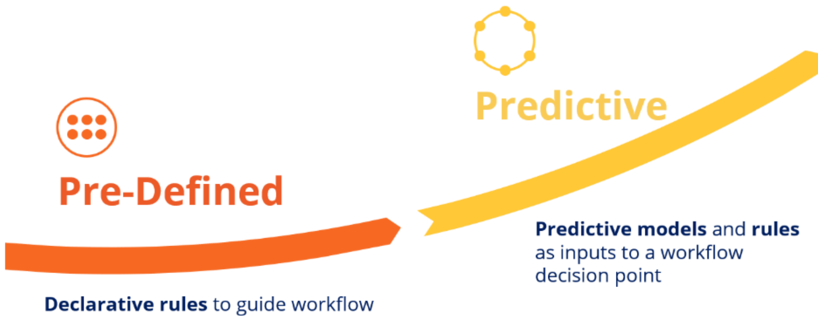


A case type can use declarative rules to manage the workflow, for example, to confirm that the order contains a valid shipping address or the order amount threshold to qualify for free shipping.

Pega Process AI can improve the quality of the decisions in the workflow by weighting in predictions, driven by predictive models.

The first approach is to operationalize existing predictive models that have proven their efficiency, to support the decisions that benefit from predictions, such as credit risk in a sales case or fraud risk in a claims case.

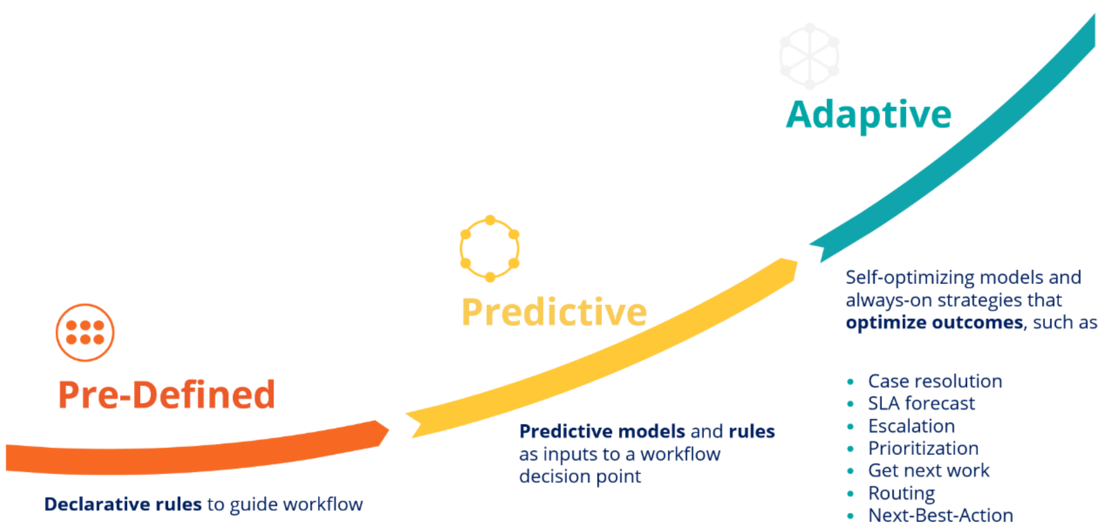
For example, the decision not to process an order can be based on a high credit risk score, and then the application can route the dubious claim for closer inspection.



The inputs for such a predictive model can be attributes of the case itself, such as the claimed amount in a claims case type, but they can also include data such as the number of claims submitted recently by the same customer.

You can build predictive models in Prediction Studio, import the models in the PMML and H2O formats, or run externally on the Amazon SageMaker and Google Vertex AI platforms to drive a prediction.

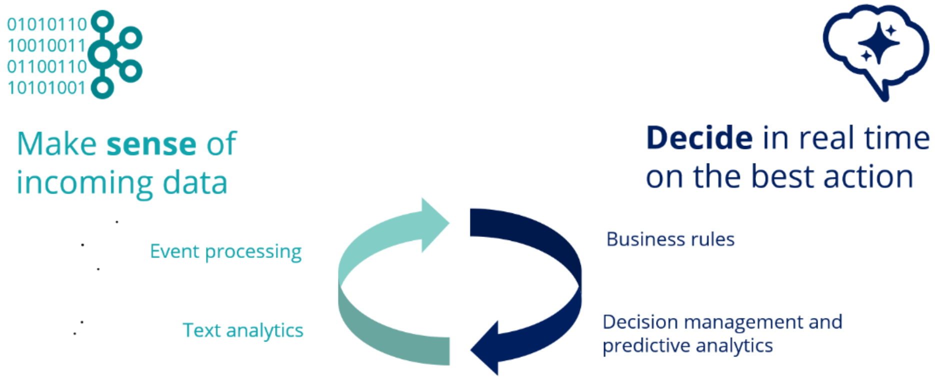
To optimize case outcomes, use adaptive models that can predict outcomes, such as case resolution, or intelligently prioritize and route cases to optimize business value and customer experience.



Adaptive models self-optimize by learning from the previous case outcomes that they capture.

The objective of Pega Process AI is to make sense of the incoming data and then decide on the best action to take in a specific stage of the case.

You can enhance the incoming data analysis by event processing to detect patterns of interest in real-time data streams and by natural language processing of incoming text.



The decision is based on the business rules and supported by predictive analytics. This process is repeated every time that the case requests a decision.

As the number of processed cases increases and model evidence accumulates, the predictive power of the models increases over time.

To summarize, Pega Process AI uses artificial intelligence in case management to produce better business outcomes.

You can use real-time, adaptive case outcome predictions and your own AI models in custom predictions.

Process AI predictions

Introduction

With the decision management capability of Pega Platform™, you can enhance applications to help optimize business processes, predict customer behavior, analyze natural language, and make informed decisions to better meet the needs of customers and achieve positive business outcomes.

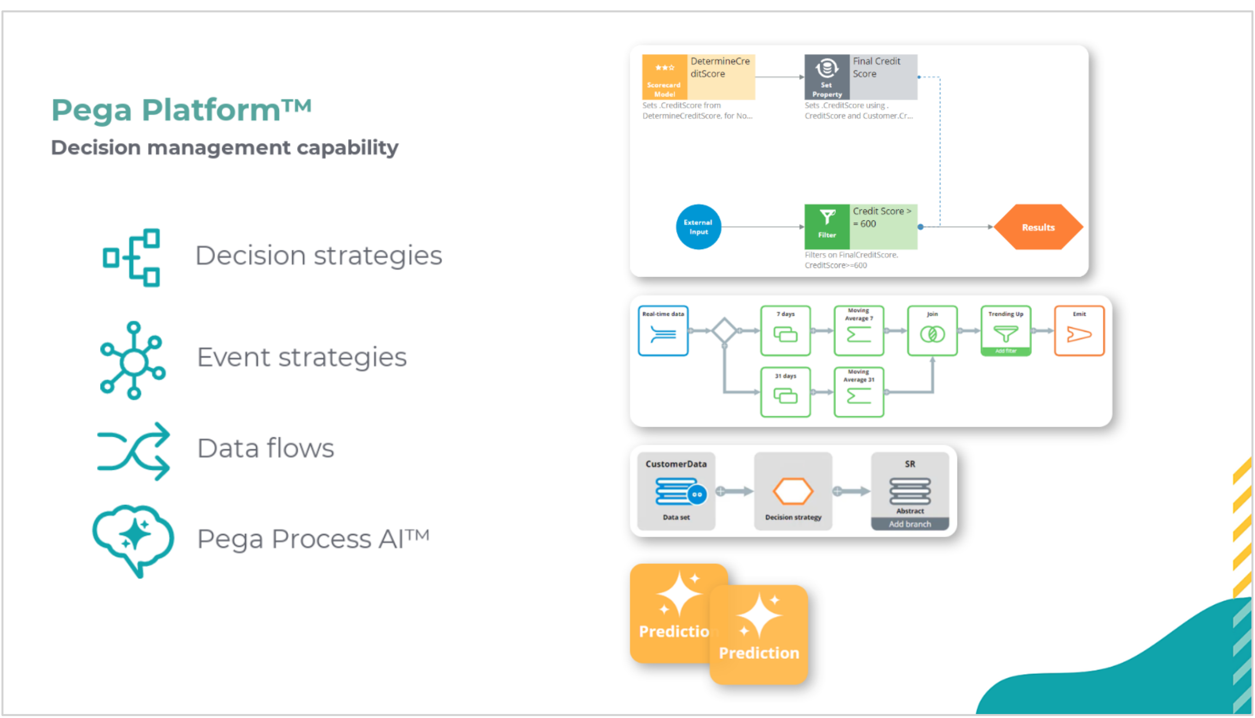
Transcript

This video introduces you to Pega AI, a feature of the decision management capability of Pega Platform.

Other decisioning features of Pega Platform include:

- Decision strategies that feature a business- and user-friendly canvas with which you can create decision logic that uses behavioral and operational data to improve intelligent processes.
- Event strategies to detect patterns in data streams and react to them.
- Data flows as scalable and resilient data pipelines to ingest, process, and move data from one or more sources to one or more destinations.

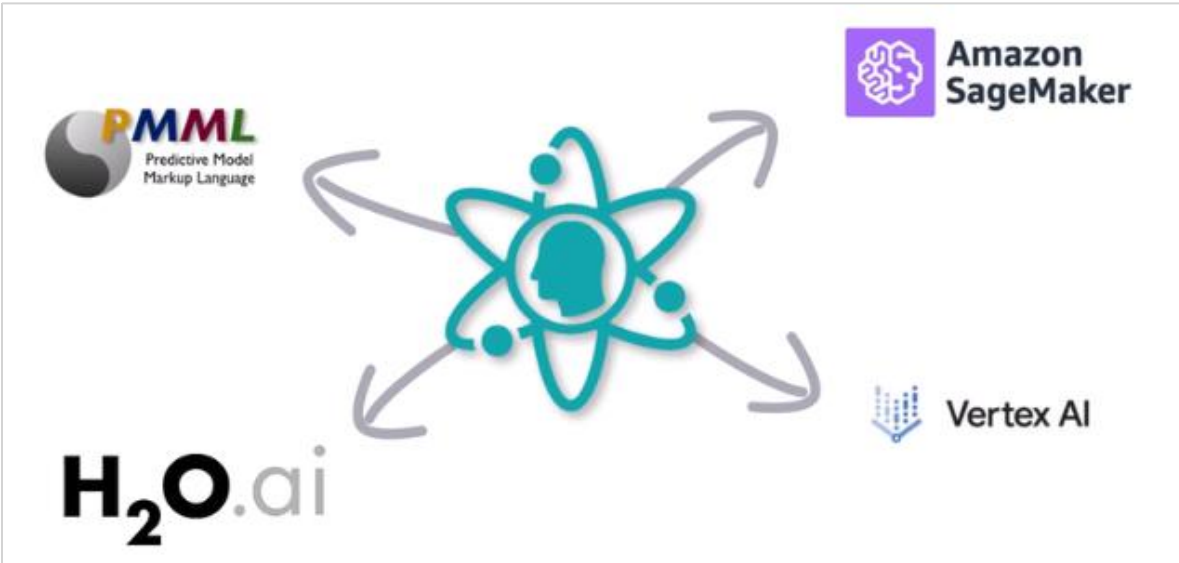
Decision management uses Pega AI to make predictions about the possibility of fraud, successful case completion, and other subjects to make decisions more relevant.



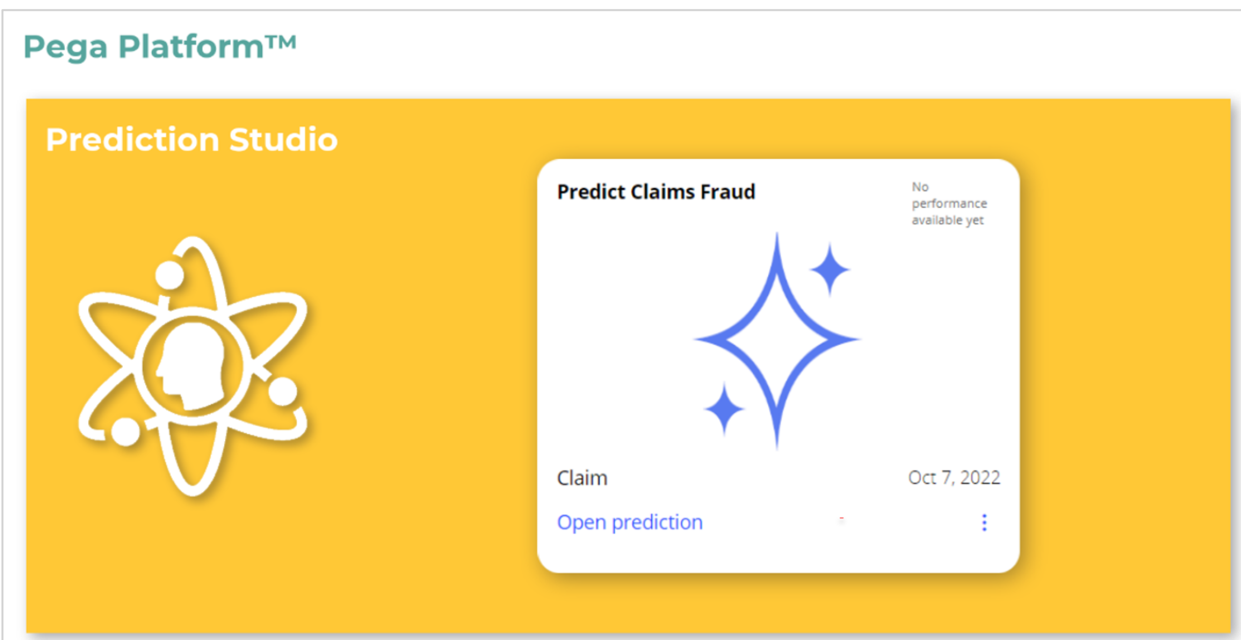
Decision management is a Pega Platform capability. You can apply decision management to any application that is built on Pega Platform.

Various and versatile predictions are available, but one or more predictive models drive them all.

For example, a data scientist can create a predictive model in Pega Platform or an external environment that can export the model as a PMML or H2O file back to Pega Platform. Another option is to connect to a machine learning service such as AWS SageMaker or Google Vertex AI.

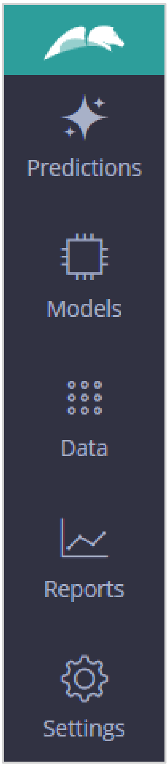


If an insurance company wants to use Pega Process AI™ to route incoming claims that might be fraudulent to an expert, based on the outcome of a predictive model, the data scientist creates a fraud model to drive a new case management prediction in Prediction Studio.

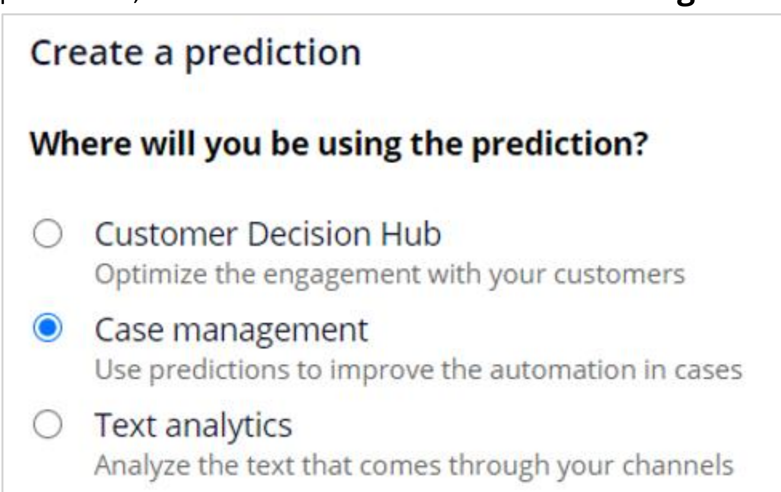


Prediction Studio is the dedicated workspace where you manage the life cycle of predictive models and the predictions that they drive.

The workspace provides data scientists with everything they need to author, deploy, govern, monitor, and change predictions. Prediction Studio has five work areas: **Predictions**, **Models**, **Data**, **Reports**, and **Settings**.



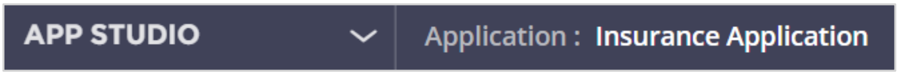
On the Predictions landing page, you create and manage predictions. There are three types of predictions, but Process AI focuses on the **Case management** prediction.



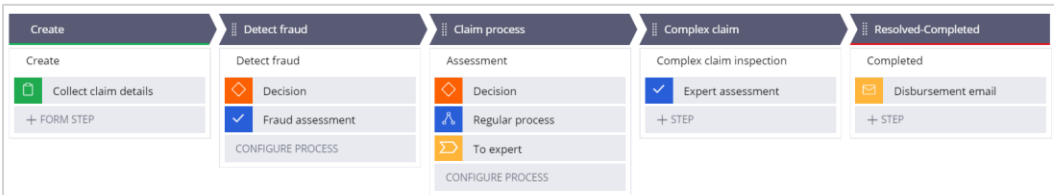
Case management predictions are used in case types to support decisions in business processes. For example, predictive models can help to predict whether an insurance claim is fraudulent or distinguish regular from complex claims.

This dependence routes cases more accurately and strengthens the separation of concerns.

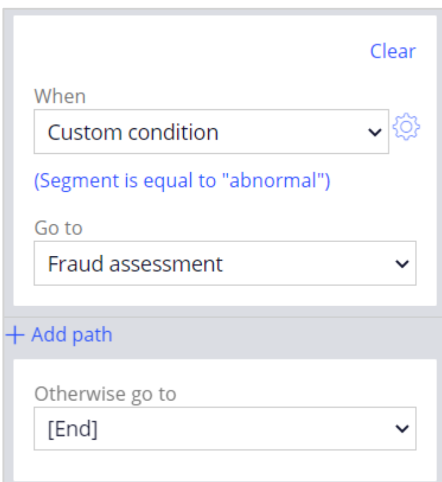
The decision step in a case type uses **case management** predictions.



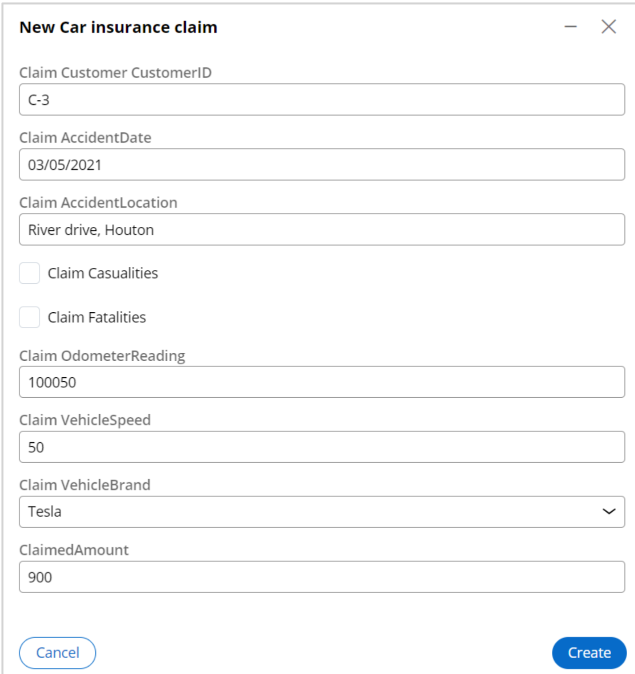
Consider the following case type, which handles incoming car insurance claims:



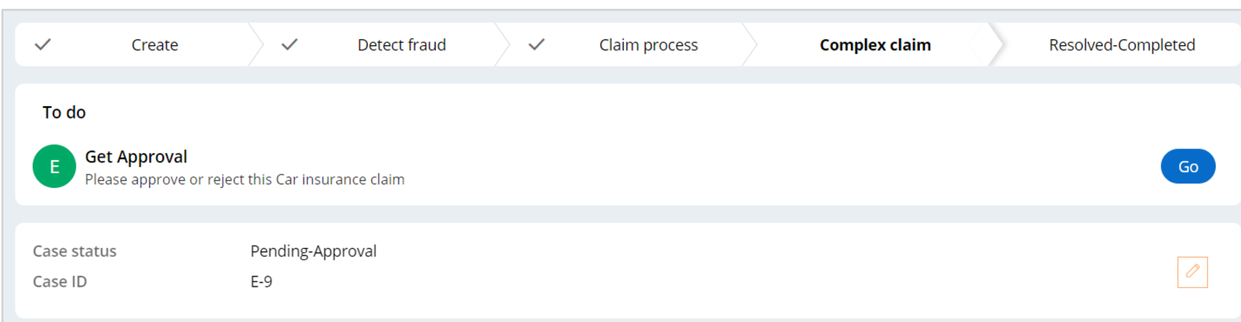
An application developer can use the outcome of the prediction in the condition of a decision step instead of a business rule. Based on the condition, the system routes a case to a fraud expert when the prediction flags the claim as abnormal.



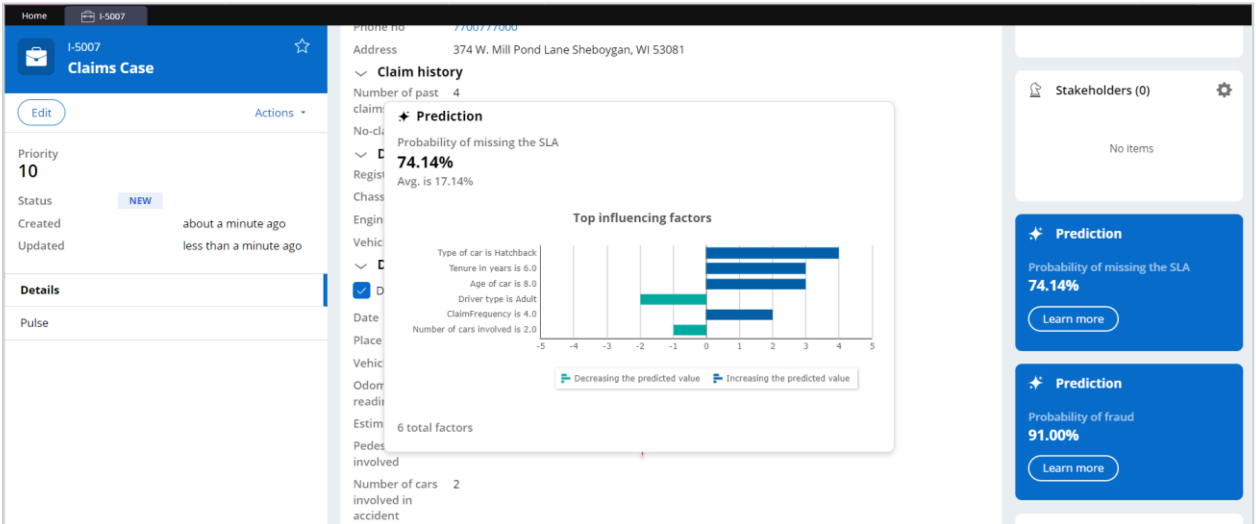
Pega Process AI can also help to distinguish regular from complex claims. It helps speed up the process by identifying such cases early and routing them to the right person.



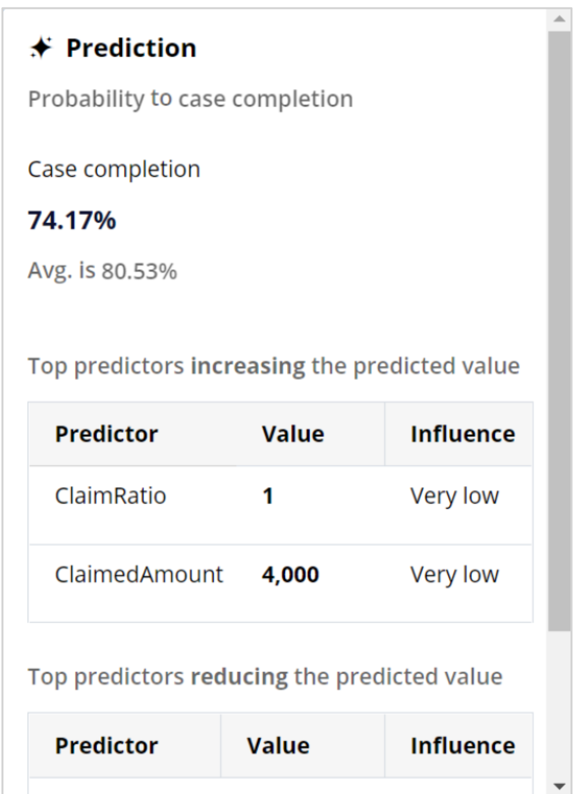
In the following case, the data scientists create a prediction that aims to identify cases that are likely to miss their deadlines. Then, an application developer uses the prediction outcome when configuring the case type so that the system can automatically route a complex case to a senior employee for evaluation.



Some additional helpful information widgets are available in the case view. Notice the prediction widget in the claims case. The widget conveniently displays the output of the trained prediction models, in this case, fraud or missing service-level agreement (SLA) probability. By clicking **Learn more**, you can see the details of the model and whether it received training with adequate data (in this example, the probability of case completion).



The widget also displays which predictors contribute positively (increasing the predicted value), and which contribute negatively, (decreasing the predicted value).



You have reached the end of this video. You have learned:

- How Pega AI allows you to improve business processes by using predictions.
- How predictive models drive predictions.
- How to create and manage predictions in Prediction Studio.

- How to use predictions in a case type to improve business processes.

Applying NLP for Case classification

Description

Pega Process AI utilizes natural language processing to automatically categorize and route claims based on customer descriptions, eliminating delays and errors associated with manual processing. Learn how to enable AI Case classification, define topics, train models, and validate the accuracy of automatic categorization to streamline claims processing and enhance operational efficiency.

Learning objectives

- Enable AI Case classification in Pega Process AI by creating a new accident category field and defining topics.
- Train and test models in Prediction Studio
- Validate the automatic AI accident category prediction and Case routing by creating a new Claims Case for a test Persona.
- Verify the effectiveness of intelligent Case routing, which automatically routes Cases to the correct Work Queue based on the predicted accident category.

Applying NLP for Case classification

In the complex and dynamic world of insurance, processing claims is an important task. It requires a high degree of precision, efficiency, and speed. However, the traditional approach, which involves manual categorization and routing of Cases, often leads to delays and errors. These issues can negatively impact customer satisfaction and operational efficiency.

Pega Process AI, equipped with advanced features, addresses these challenges. By using the power of natural language processing (NLP), Pega Process AI can automatically identify the accident category based on the Case description provided by the customer in their insurance claim. As shown in the following demo, Pega Process AI not only eliminates the need for manual intervention but also ensures swift and accurate routing of the Case to the correct Work Queue.

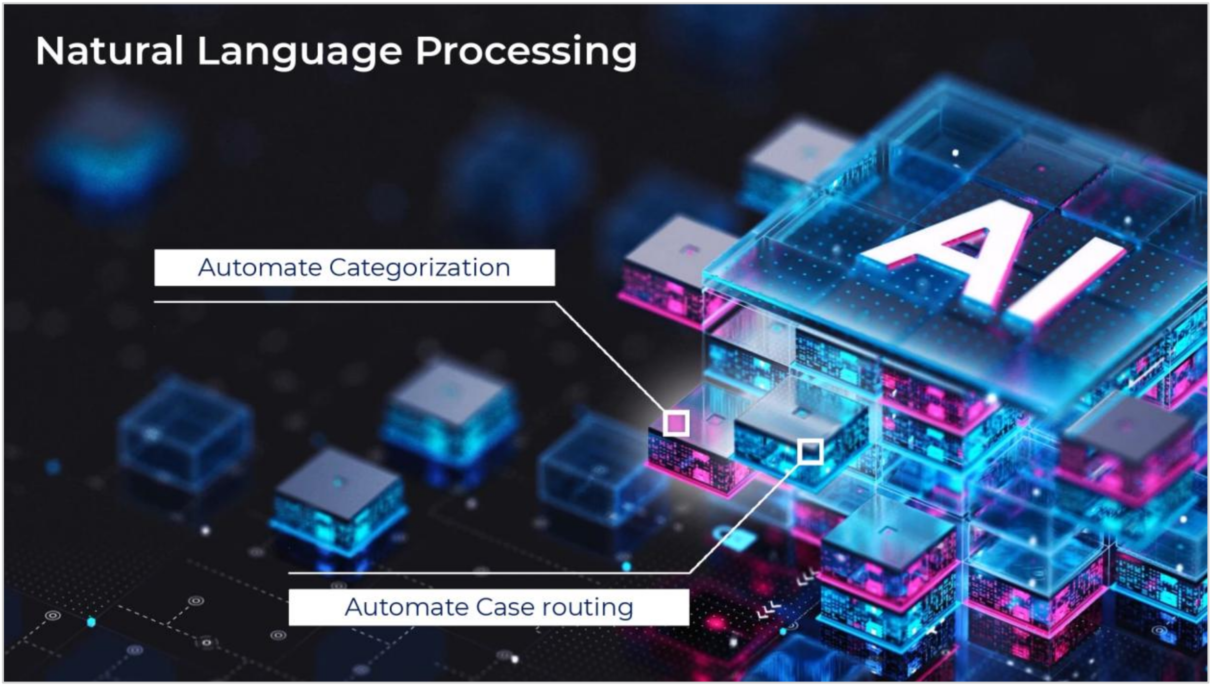
Transcript

This demo shows you how to enable Case classification with AI and test it in a real-world scenario.

Consider a scenario that involves an insurance company, U+ Insurance. This company is currently handling a high volume of car insurance claims. The process they follow is traditional and manual; experts review all Cases and manually categorize and route them based on the description provided by customers. This process often results in delays and errors due to the sheer volume of Cases and the human element involved.

Recognizing the need for a more efficient and accurate process, the company decides to utilize natural language processing in Pega Process AI.

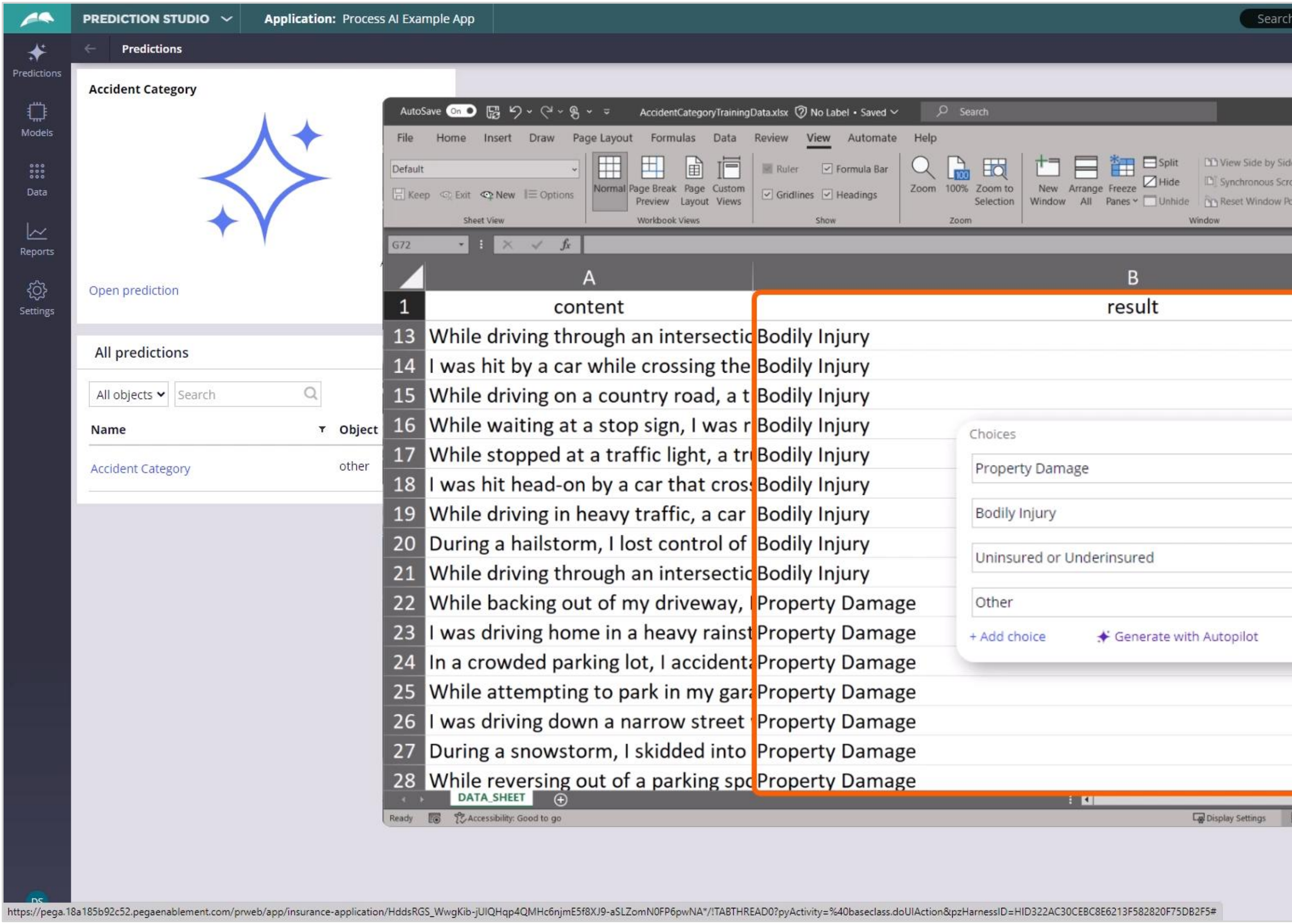
The goal is to automate the categorization and Case routing, which reduces the time it takes to process claims and minimizes the possibility of errors as a result.



An Application Developer must enable the AI-powered Case classification prediction. If the desired category field is nonexistent in the application, the developer begins by creating it in the Case and defining the topics. The system uses the choices set by the Application Developer as topics for the new Accident Category Prediction. This is a crucial step as it sets the foundation for the AI to understand and categorize the Cases. Now, the Application Developer must enable AI and specify the Case description provided by the customer as the input text for AI analysis.

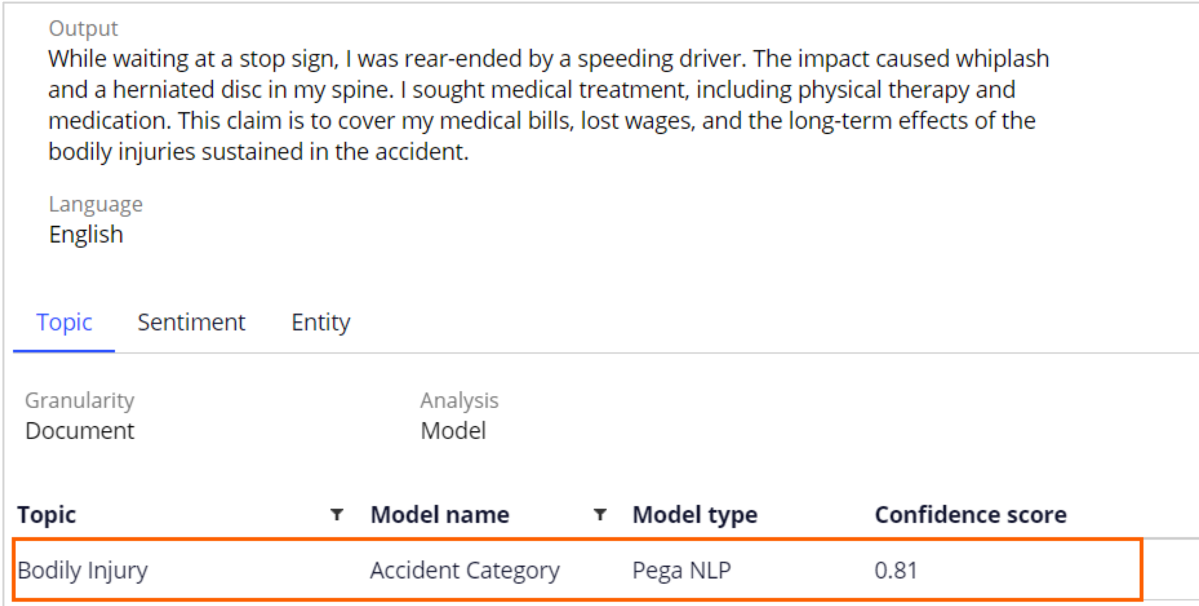
After the Application Developer enables AI, the system automatically creates a new Prediction in Prediction Studio. When insurance claims arrive, the Prediction uses AI models to recognize the topic of each claim and categorize it based on the Case description provided by the customer. The topics correspond with the choices set in the new Accident Category field.

At first, the models are not trained. Data Scientist trains and builds the models in Prediction Studio. Data Scientist uses a provided training data file, which contains examples of Case descriptions and their corresponding topics. The topics match accident categories that the prediction needs to detect.

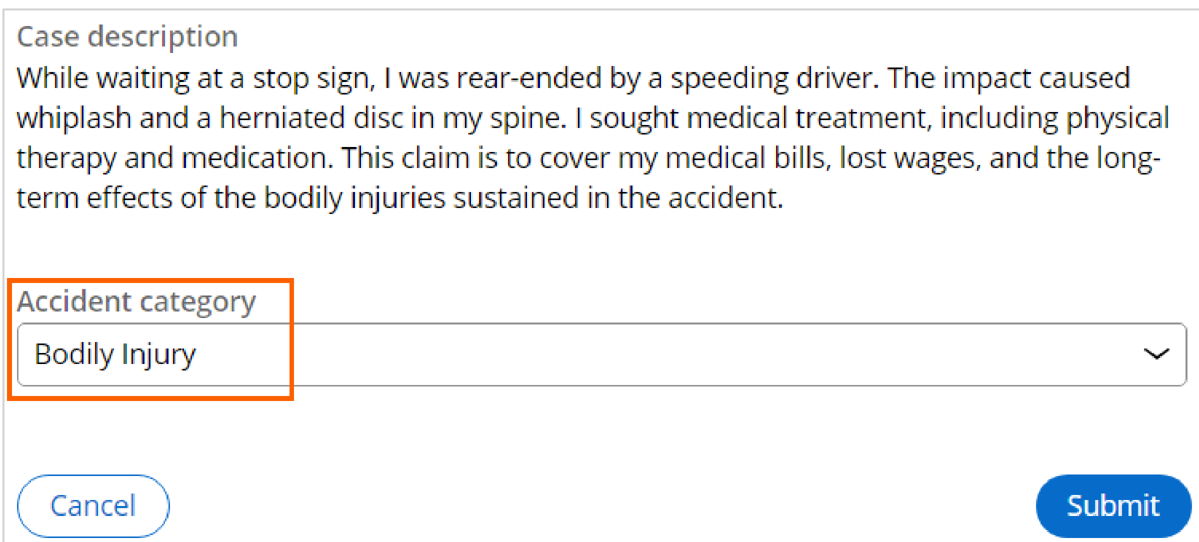


After the models train on the data, the text Prediction undergoes testing in Prediction Studio. The Data Scientist confirms that the AI correctly detects the topic of each message by providing the AI with sample

Case descriptions and checking if the AI correctly identifies the accident category.

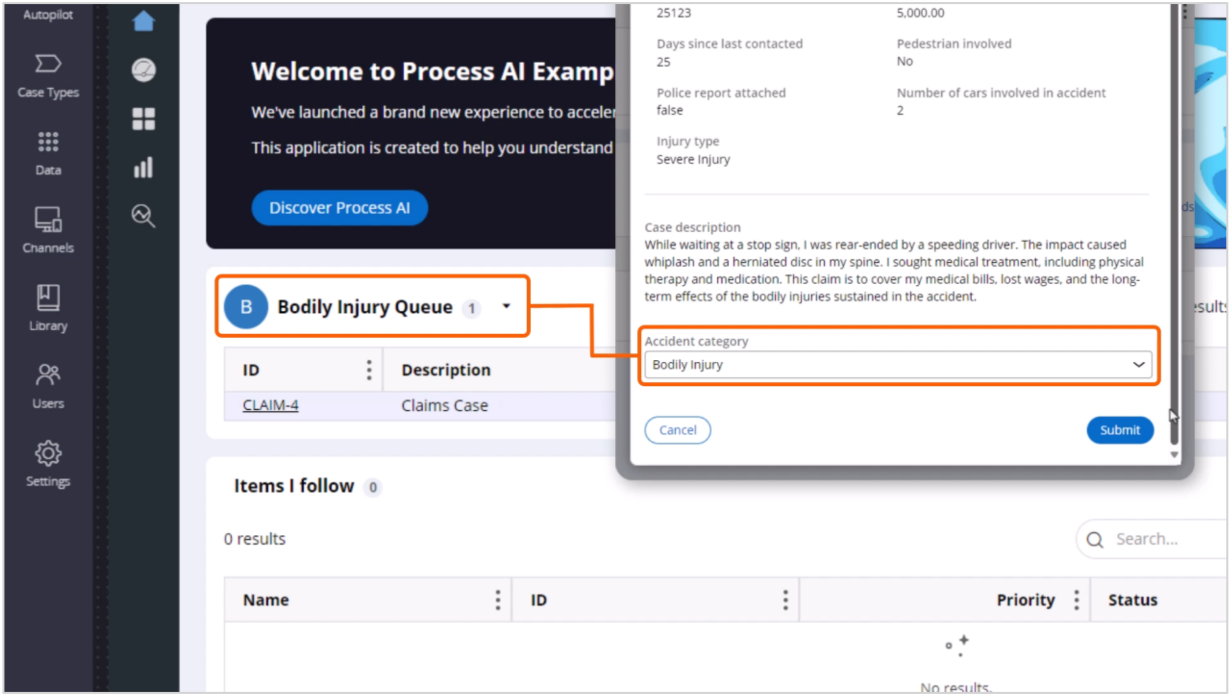


After the models complete training and testing, the Application Developer steps back in to validate the functionality of the automatic AI accident category prediction and Case routing in the Process AI Example Application. The developer creates a new Claims Case for each Persona and validates the detected category. This step helps ensure that the AI is accurately categorizing and routing the Cases in a real-world scenario.



Finally, the Application Developer verifies the pre-configured intelligent Case routing. The system automatically routes the Cases to the correct Work Queue based on the predicted accident category. This action eliminates the need for manual classification and ensures that the appropriate teams handle

Cases, which improves the efficiency of the claims processing workflow.



This demo has concluded. What did it show you?

- How to enable AI Case classification in Pega Process AI by creating a new accident category field and defining topics.
- How to train and test models in Prediction Studio.
- How to validate the automatic AI accident category prediction and Case routing by creating a new Claims Case for a test Persona.
- How to verify the effectiveness of intelligent Case routing, which automatically routes Cases to the correct Work Queue based on the predicted accident category.

Predicting fraud

Description

Occasionally, an insurance claim might be erroneous or even fraudulent. To detect fraud and optimize the way in which the application routes work and meets business goals, learn how to use your own predictive models in case management.

Learning objectives

- Create a prediction to detect fraud
- Use the new prediction in a case type

Predicting fraud

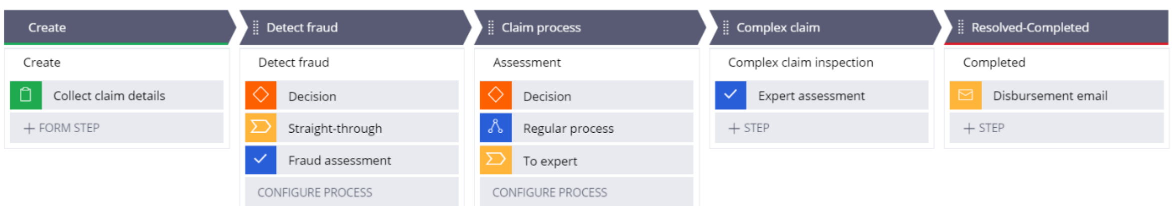
Introduction

Pega Process AI™ lets you bring your own predictive models to Pega. Use predictions in case types to optimize the way in which your application processes work and to meet your business goals. Learn how to use a predictive fraud model to effectively route suspicious claims for closer inspection.

Transcript

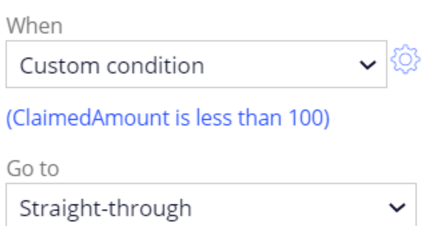
This demo will show you how to use a predictive fraud model in a case type to route suspicious claims to an expert.

U+ Insurance uses Pega Platform™ for case management. The life cycle of the case type that processes incoming car insurance claims contains a fraud detection stage, a regular process stage, and a complex claim process stage.

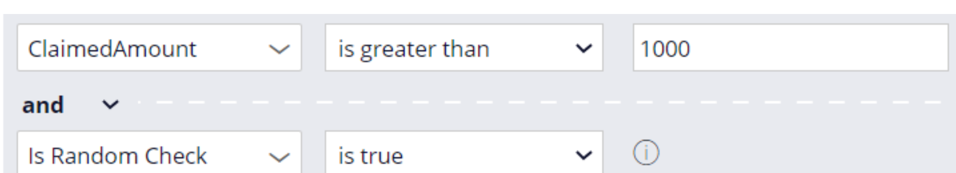


When the case is resolved, the claimant receives an email that communicates the decision.

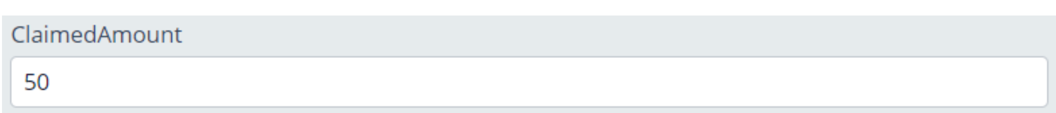
The decision step in the **Detect fraud** stage routes cases with a low claimed amount for straight-through processing.



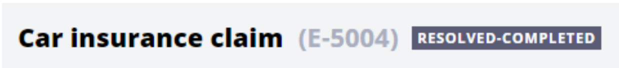
A set percentage of claims with a high claimed amount is routed to an expert for fraud assessment.

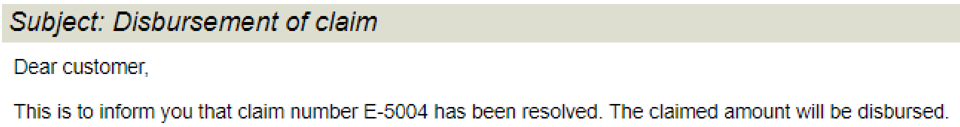


Consider this car insurance claim. The claimed amount is 50.



The claim qualifies for straight-through processing as the claimed amount is below the threshold. The case is automatically resolved, and the claimant receives an email that states that the claimed amount will be disbursed.

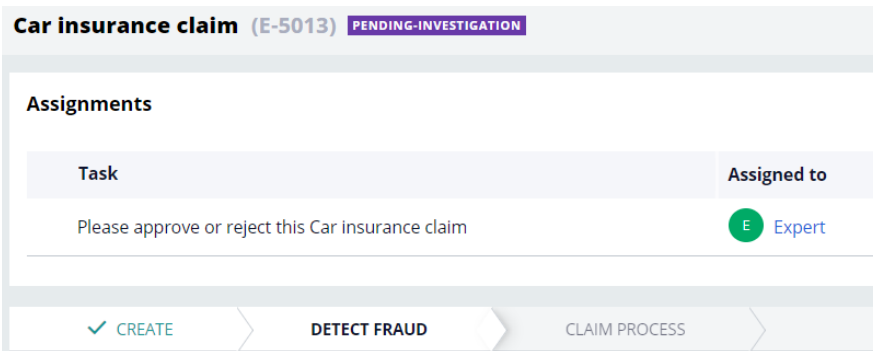



Subject: Disbursement of claim

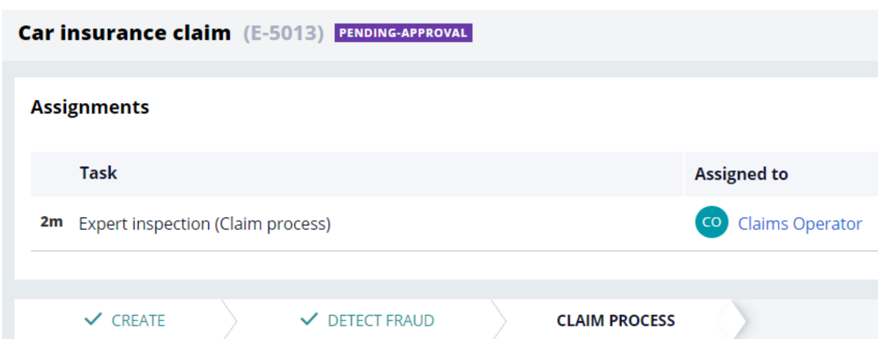
Dear customer,

This is to inform you that claim number E-5004 has been resolved. The claimed amount will be disbursed.

A fraud expert inspects a set percentage of cases with a high claimed amount.



After approval, the system routes the case to the regular claim process.



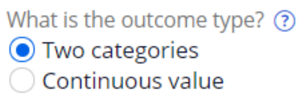
U+ Insurance wants to improve the effectiveness of fraud detection by using a predictive model that calculates the fraud risk of each claim.

The business requirements are that claims only qualify for straight-through processing if the fraud risk score is very low, while all claims with a high fraud risk score are inspected by the fraud expert. The routing of randomly selected cases to the fraud expert must remain in place to create a control group.

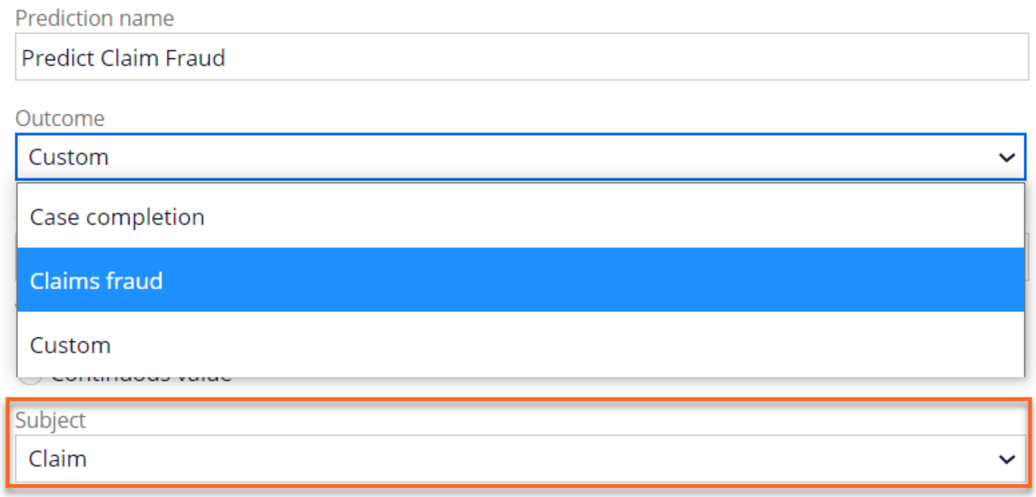
The data scientist team of U+ Insurance has developed a fraud model on the H2O.ai platform and has validated the model against historical data that the company captured.

The system qualifies a claim as abnormal if the probability of fraud exceeds the threshold; otherwise, the system classifies the case as normal.

To implement the fraud model, you create a new case management prediction. You can create a custom prediction that can forecast binary or numerical outcomes.



For fraud detection, Process AI provides an out-of-the-box template. The claim is the subject of the prediction.

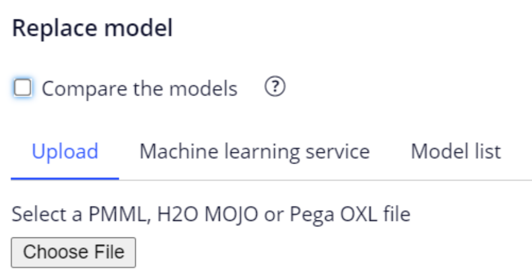


A placeholder scorecard initially drives the prediction.

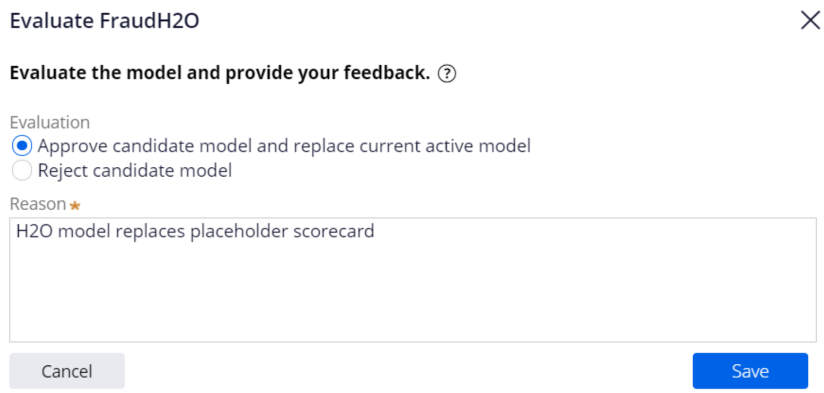
Claims fraud

Name	Type	Performance	Status
Predict Fraud	Scorecard	——	ACTIVE

When the predictive fraud model replaces the scorecard, the prediction is ready for implementation in the Car insurance claim case type. You replace the placeholder with a machine learning model, a scorecard, or a field that contains a precalculated score. You can upload a machine learning model as a PMML or H2O file. Alternatively, you can connect to online machine learning services.



You can select predictive models that are available in the application in the model list. When the model is ready for review, approve the model to replace the scorecard.

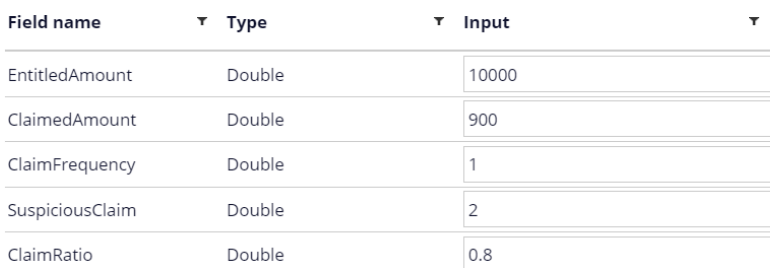


The fraud model now drives the prediction.

Claims fraud

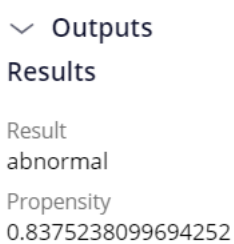
Name	Type	Performance	Status
FraudH2O	Predictive model	——	ACTIVE

When you run the model with these input values, the model qualifies the claim as abnormal.



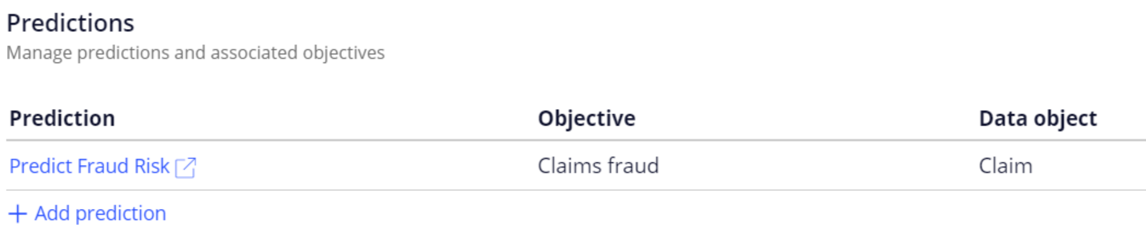
Field name	Type	Input
EntitledAmount	Double	10000
ClaimedAmount	Double	900
ClaimFrequency	Double	1
SuspiciousClaim	Double	2
ClaimRatio	Double	0.8

The model predicts the claim to be abnormal because the propensity value is above the threshold.

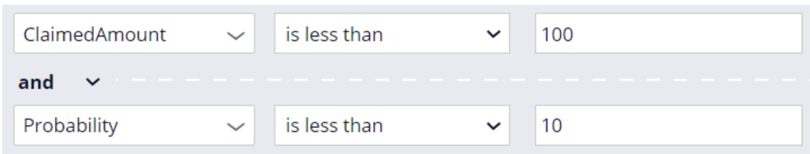


Predictors of the model include the claim data, such as location and claimed amount, but can also cover customer behavior data, such as the number of recent claims.

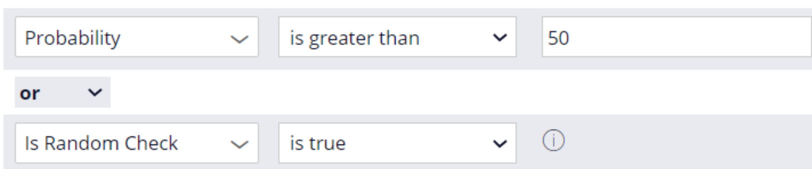
As an application developer, you can implement the fraud prediction to route claims based on the fraud risk calculated by the model.



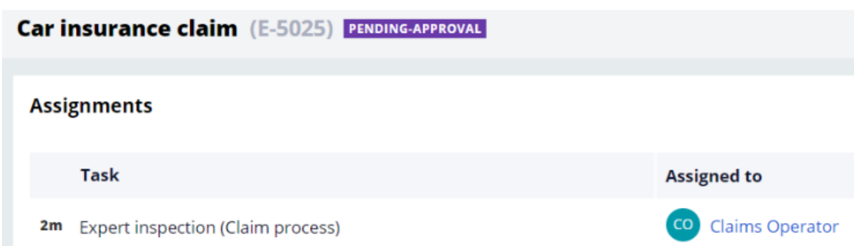
In the **Decision** step in the **Detect fraud** stage of the life cycle, implement the prediction. Add the condition that only claims with a very low predicted fraud risk qualify for straight-through processing.



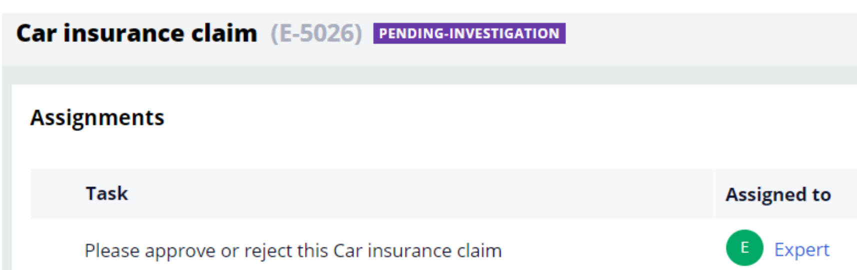
Replace the condition that routes a claim to a fraud expert based on the claimed amount with a condition that is based on the outcome of the fraud model and change the logical operator to generate the control group.



When you run the same claim that previously qualified for straight-through the claim now disqualifies because the condition that fraud risk is very low is not met and the system consequently routes the case for regular processing.



When a claim with the same predictor values as previously tested in Prediction Studio is run, the system routes the case to the fraud expert.



This demo has concluded. What did it show you?

- How to create a case management prediction driven by a predictive model.
- How to use a prediction in a case type.

Predicting missing the Service-Level Agreement

Description

Pega Process AI™ can help to distinguish regular from complex claims. Complex claims often escalate into a lengthy process, which is not only costly, but also leads to poor customer experiences.

Learn how to use Process AI to create an adaptive model to route complex cases to an experienced handler and leave many of the claims for straight-through processing. As the adaptive model learns from the outcome of each case, it becomes more accurate at predicting which claims to escalate, and in that way to self-optimize the process.

Learning objectives

- Create a prediction that predicts case outcomes
- Use the new prediction to route complex cases to an expert

Predicting missing the Service-Level Agreement

Introduction

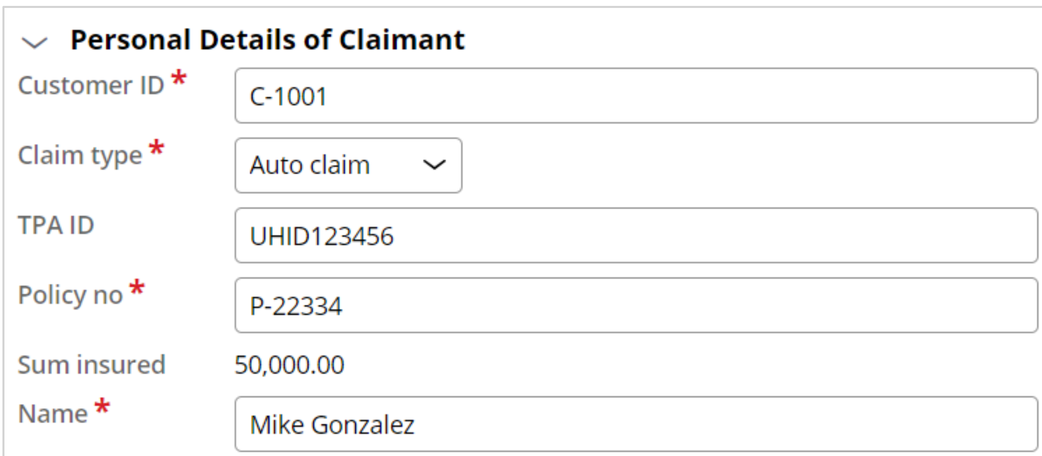
Pega Process AI™ can help to distinguish regular from complex claims. Complex claims often escalate into a lengthy process, which is costly and leads to a bad customer experience. The distinction lets you detect these claims early and address them at once.

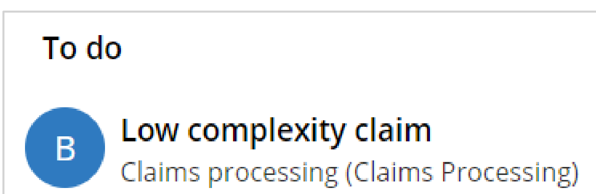
Learn how to create a prediction that aims to identify cases that are likely to miss their deadlines and route them to a senior employee to handle them more efficiently and improve the customer experience.

Transcript

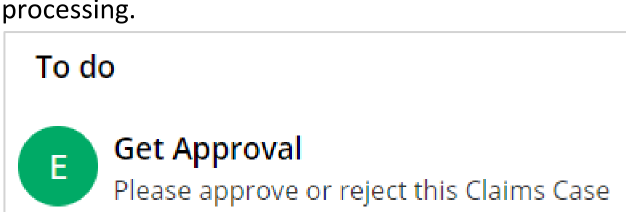
This demo shows you how to use adaptive models to predict missing the Service-Level Agreement (SLA).

U+ Insurance uses Pega Platform™ for case management. An incoming car insurance claim is straight through processed, or routed to a claims operator, who approves or rejects the claim to resolve the case.

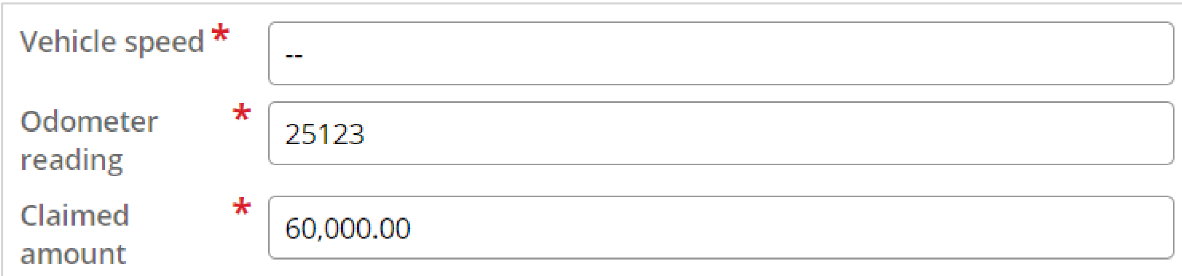




A case is escalated to an expert when the claim is not completed in the allotted time for regular processing.

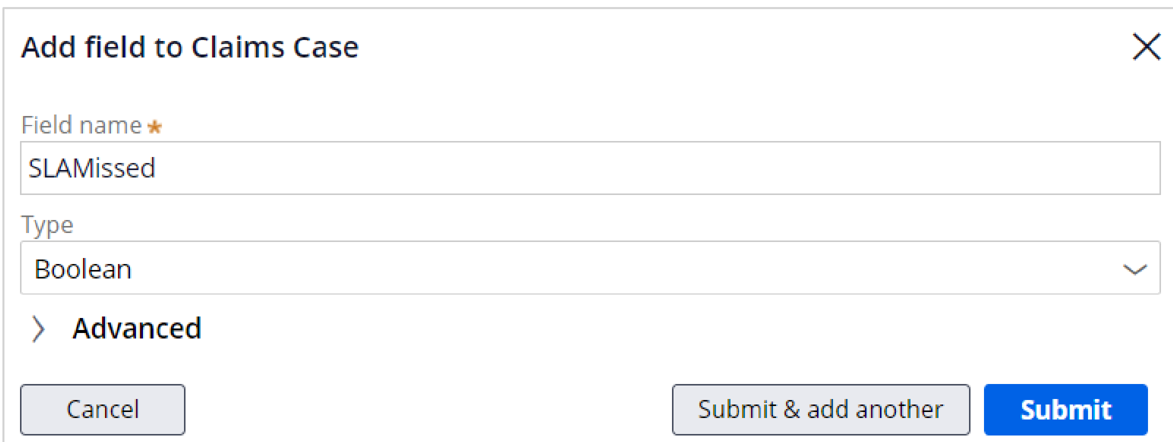


In the current configuration, claims that exceed 45000 are considered highly complex and are always investigated by an expert as a precaution.

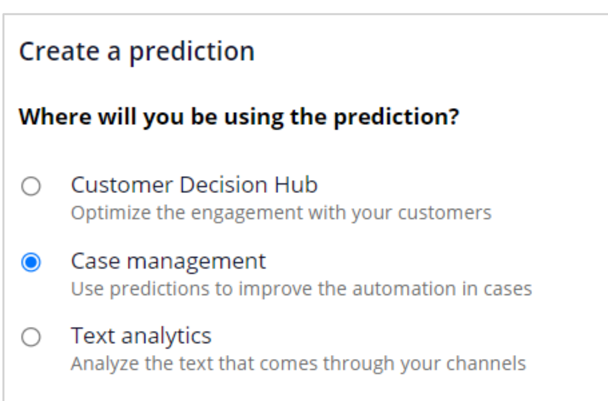


However, decisioning using hard business rules, like in this case based on a simple cutoff value is not efficient, because even cases that exceed 45000 can be often resolved on time in the regular claims process. As a result, the experts consequently spend valuable time on relatively simple claims.

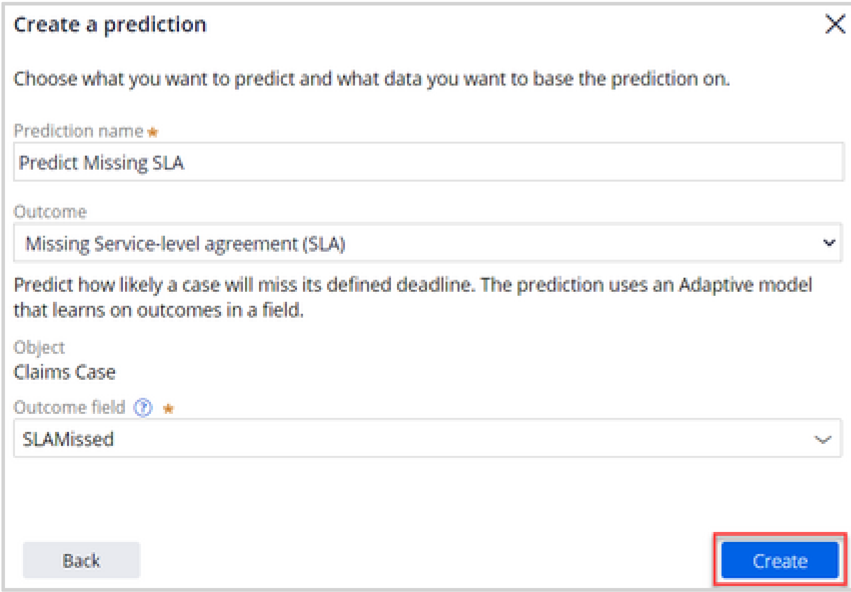
Process AI can help optimize the process by predicting the likelihood that a case is resolved before the deadline in the regular workflow and otherwise, route it to an expert irrespective of the cause of the complexity of the claim. First, it is an Application Developer's task to create a Boolean outcome field. It serves the adaptive model as the outcome field and allows it to distinguish cases that missed the SLA. You add the outcome field in the case type data model settings to make it available in that case type.



Next, to allow the model to learn from future outcomes, in Goal & deadline settings of the case type, the Application Developer configures a condition that automatically sets the outcome as missed when the deadline expires. Finally, a Data Scientist creates a case management prediction that calculates the propensity of whether the case is likely to miss the SLA.



Process AI offers a wizard to create Missing Service-Level Agreement (SLA) predictions.



The Outcome field reflects the Boolean field that the Application Developer creates. This associates the prediction with the case type.

Next step for the Data Scientist is to add potential predictors. Best practice is to include many unrelated fields, including the claim properties. It is also important to exclude predictors that are irrelevant and do not have any predicting power, like **ChassisNo**, **CustomerID**, **CustomerPhoto**, **Name**, **PhoneNo**, **PolicyNo**, and **RegistrationNo**.

Add predictors

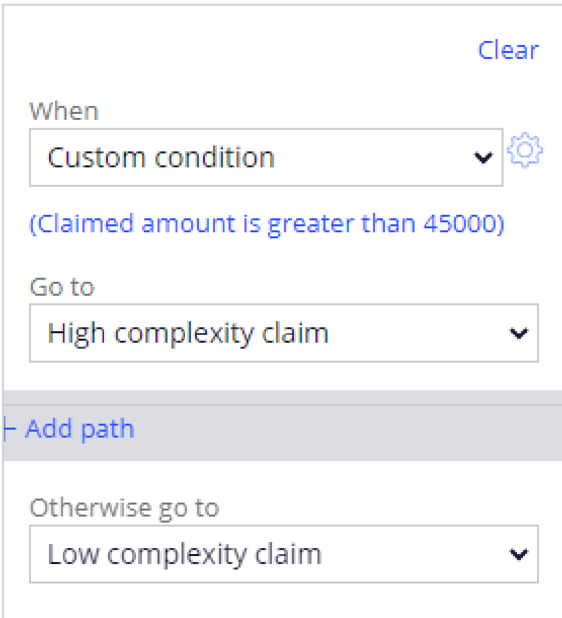
Click on the page and select fields

- Current page (InsuranceApp)
  - Page AccidentCategory_ea2f6
  - Page Customer
  - Page PredictMissingSLA

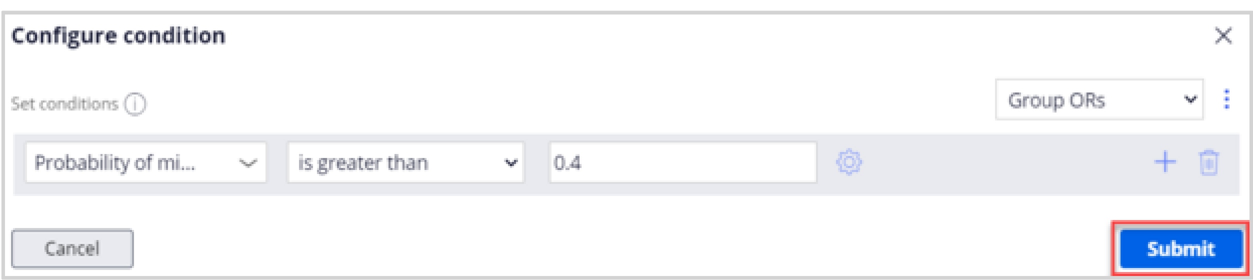
☐	Name	Data type
☑	AccidentDate	Date
☑	Address	Text
☑	Age	Integer
☑	AgeOfCar	Double
☐	ChassisNo	Text
☑	ClaimedAmount	Double
☑	ClaimFrequency	Integer
☑	ClaimRatio	Double
☑	ClaimType	Text
☐	CustomerID	Identifier
☐	CustomerPhoto	Text
☑	DaySinceLastContacted	Integer
☑	DOB	Date
☑	DocumentsSubmitted	TrueFalse
☑	DriverType	Text
☑	EngineNo	Text
☑	EntitledAmount	Double
☑	Gender	Text
☑	HospitalizationExpenses	Decimal
☑	InjuryStatus	Text
☐	Name	Text
☑	NoClaimBonus	Integer
☑	NoOfCarsInvolved	Integer
☑	Occupation	Text
☑	OdometerReading	Text
☑	PedestrianInvolved	Text
☑	PharmacyBills	Decimal
☐	PhoneNo	Text
☑	Placeofaccident	Text
☑	PoliceReportAttached	Text
☐	PolicyNo	Identifier
☑	PreHospitalization	Decimal
☐	RegistrationNo	Text
☑	RelationToInsured	Text
☑	Suspiciousclaim	Integer
☑	Tenure	Double
☑	TPAID	Text
☑	TypeOfCar	Text
☑	VehicleModel	Text
☑	VehicleSpeed	Text

The adaptive model learns from previous cases and automatically activates predictors that perform above a threshold and deactivates predictors when their performance drops over time. The prediction is ready to be implemented in the Claims case case type by an application developer.

In the current configuration, the Decide complexity decision step categorizes claims as low or high complexity depending only on the claimed amount. As a result, claims that exceed 45000 are categorized as complex.



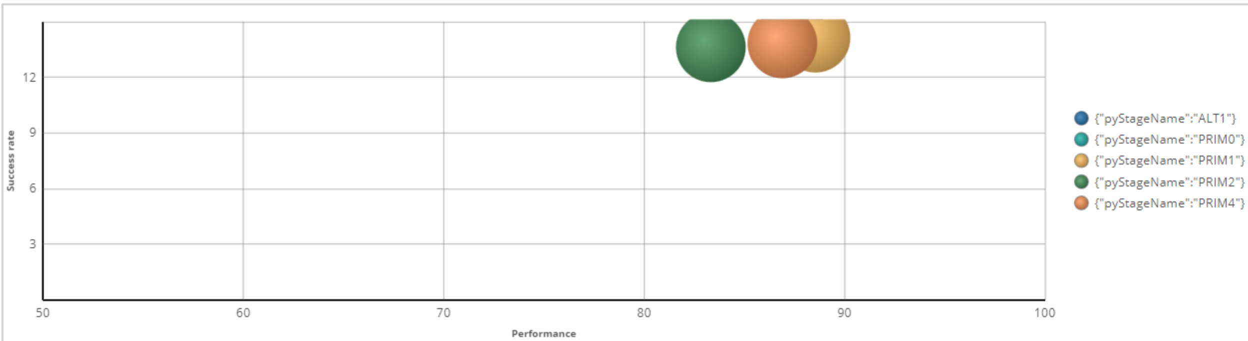
This condition requires an edit to meet the new business requirement that the routing decision is based on the propensity calculated by the Missing SLA prediction. To categorize a claim as a high complexity claim in the **Decide complexity** decision step, the propensity to miss the SLA needs to exceed a threshold. In this case: 0.4.



When a claims operator handles a claim, the case status changes to **Resolved-Completed** or **Resolved-Rejected**, and the outcome of the case maps to the alternative label (MissedSLA = False) for the prediction. When a complex claim misses the deadline, the outcome of the case maps to the target label (MissedSLA = True) for the prediction. The model learns using this information and as a result, depending on the outcome, the missing SLA propensity for a similar case in the future increases or decreases.

A claim with a high propensity to miss SLA is immediately routed to an expert. The claim is routed to the regular workflow when the expert assesses the claim and does not consider it a complex case. This reassignment allows the adaptive model to learn from cases that are incorrectly routed to the expert.

An adaptive model is created for each primary and alternative stage in the case type. A decision request in a stage uses the model that is specific to that stage to calculate the propensity.



At the very beginning, the models have no predictive power. The models learn and self-optimize with every captured case outcome.

This demo has concluded. What did it show you?

- How to create a missing SLA prediction.
- How to implement a missing SLA prediction to improve efficiency.

Pega NLP overview

Description

Gain a greater understanding of the key features, capabilities, and benefits of Pega natural language processing in decision management context. Businesses are exploring ways to utilize language processing. With Pega NLP you can analyze and extract meaningful information from a large amount of text by using text analytics to improve business performance and customer experience.

Learning objectives

- Explain text analytics in Pega NLP context.
- Describe practical use cases where text analytics can be used.

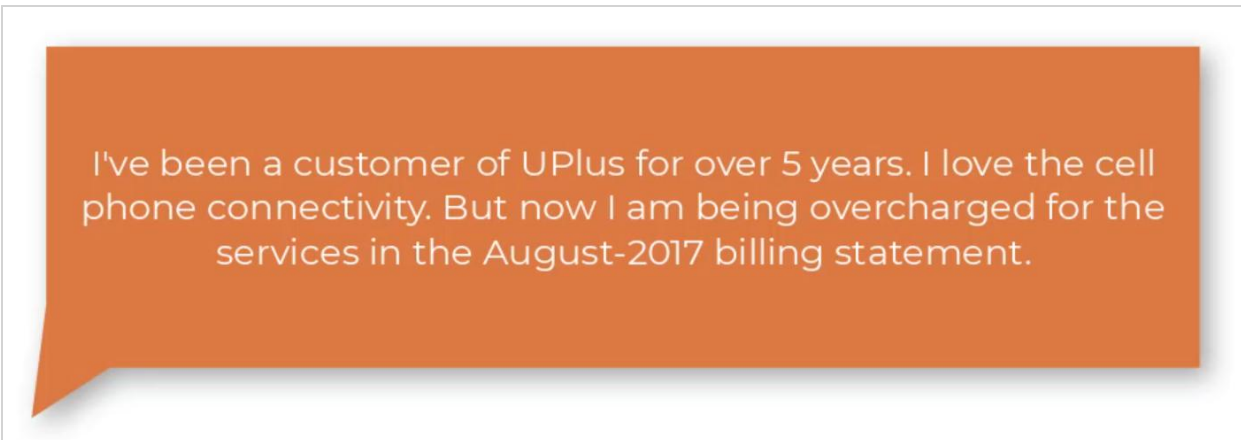
Text analytics

Today, digital sources such as news, blogs, social media, emails, and chatbots generate continuous text. Text analytics helps to transform large amounts of text into a structured format so that your applications can use the text to optimize processes such as email routing, automatic chatbot interactions, or entity extraction.

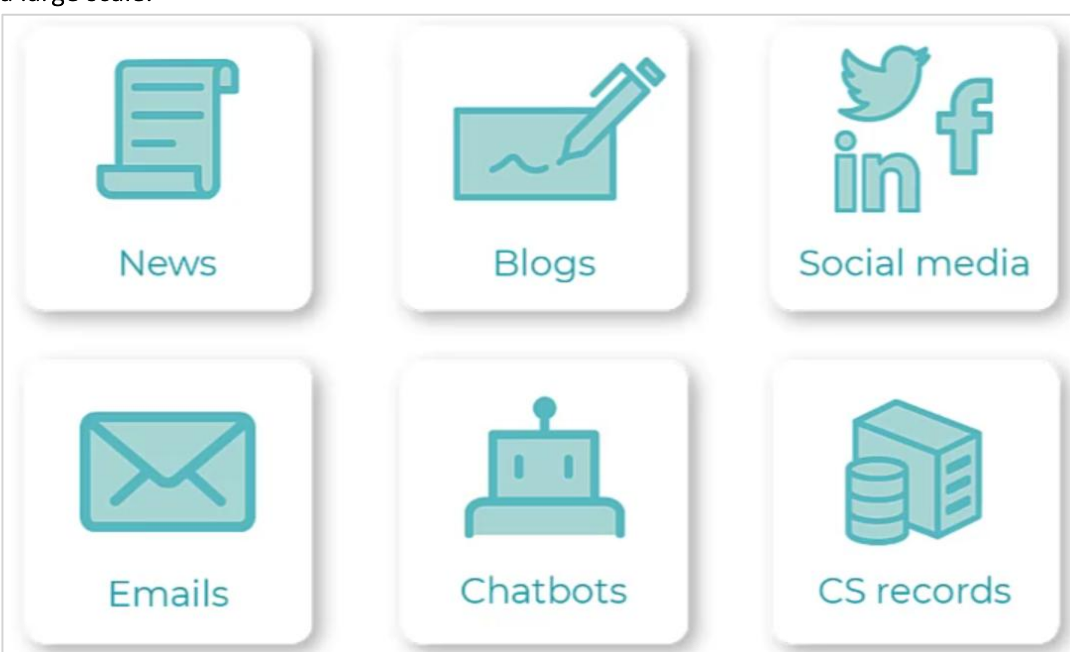
Transcript

Learn about text analytics and how it is helps determine customer sentiments and needs.

Consider this complaint from a customer to a customer service department.



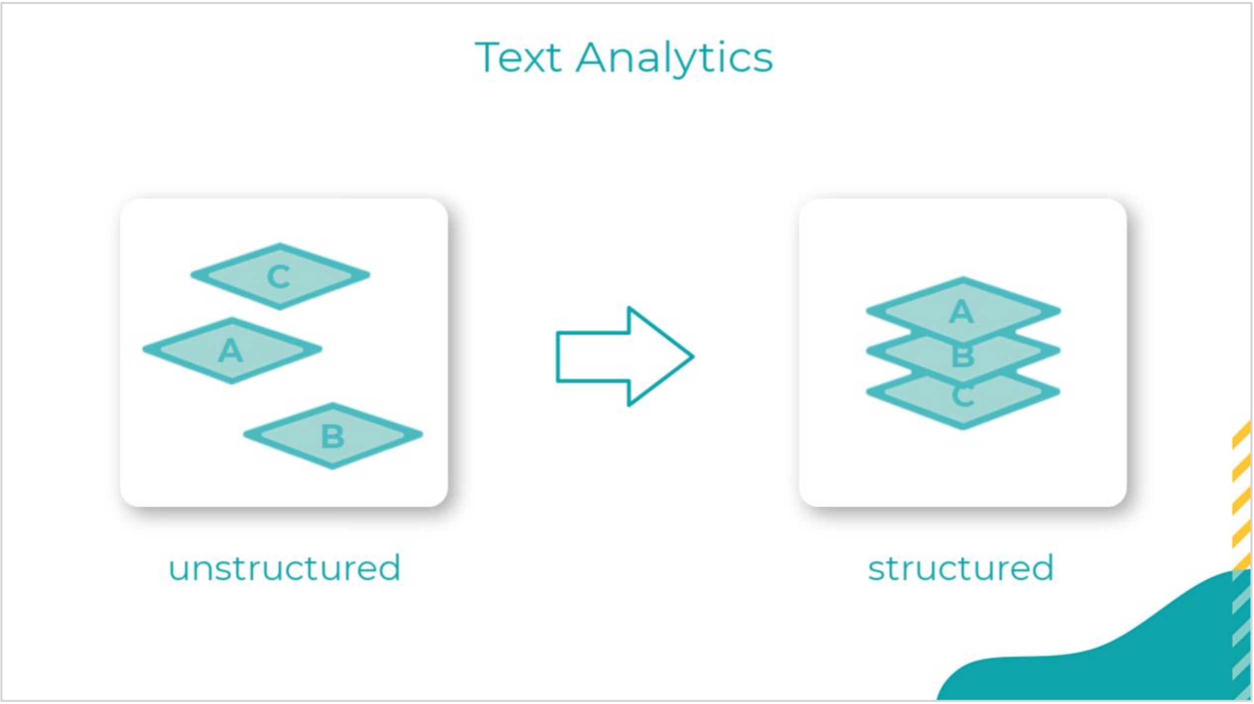
Humans can understand the implications of a piece of text by summarizing, elaborating, or even rephrasing it while preserving its original meaning. The challenge for humans comes when text exists on a large scale.



Consider a company with tens of thousands of customer service records in text form.

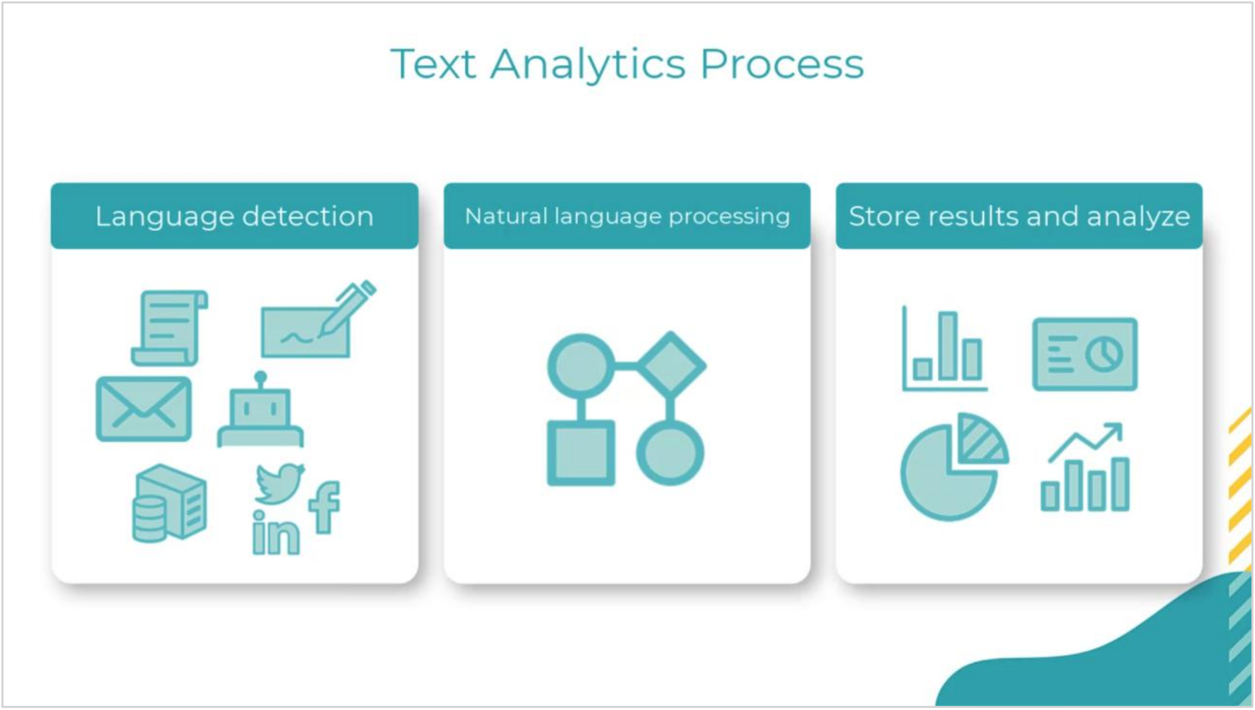
The company wants to route all complaints automatically to the customer service department. Then, it wants a report to understand whether customers are satisfied with various products and services. The company must find a way to process and extract the text, and text analytics is the most efficient and effective way to accomplish this task.

Text analytics is the process of deriving valuable information from text.



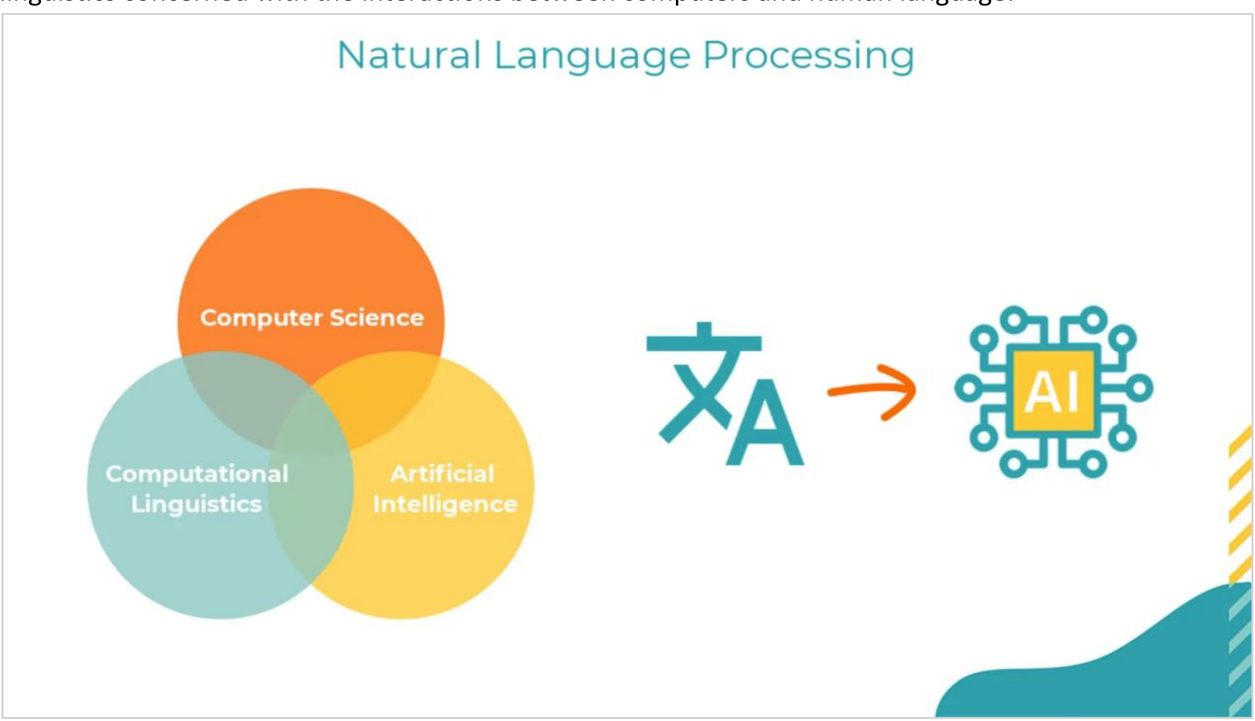
It involves converting large amounts of text into structured data that can be analyzed using statistical methods. There are three main steps involved in text analytics:

1. Language detection
2. Natural language processing
3. Store results and analyze



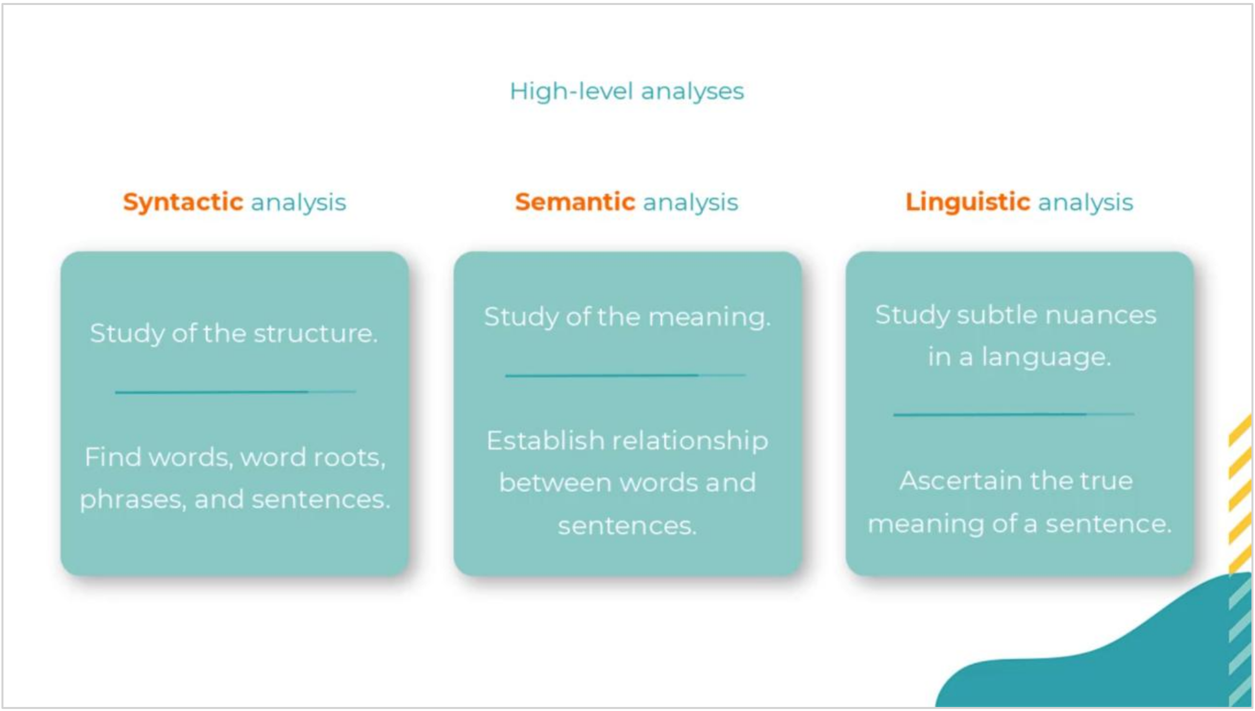
First, the system collects data from a source, such as social media, emails, chatbots, or customer service records. Second, natural language processing techniques are applied to extract specific attributes from the text and present them as structured data. And third, the extracted structured data is stored and analyzed.

Natural language processing is a field of computer science, artificial intelligence, and computational linguistics concerned with the interactions between computers and human language.

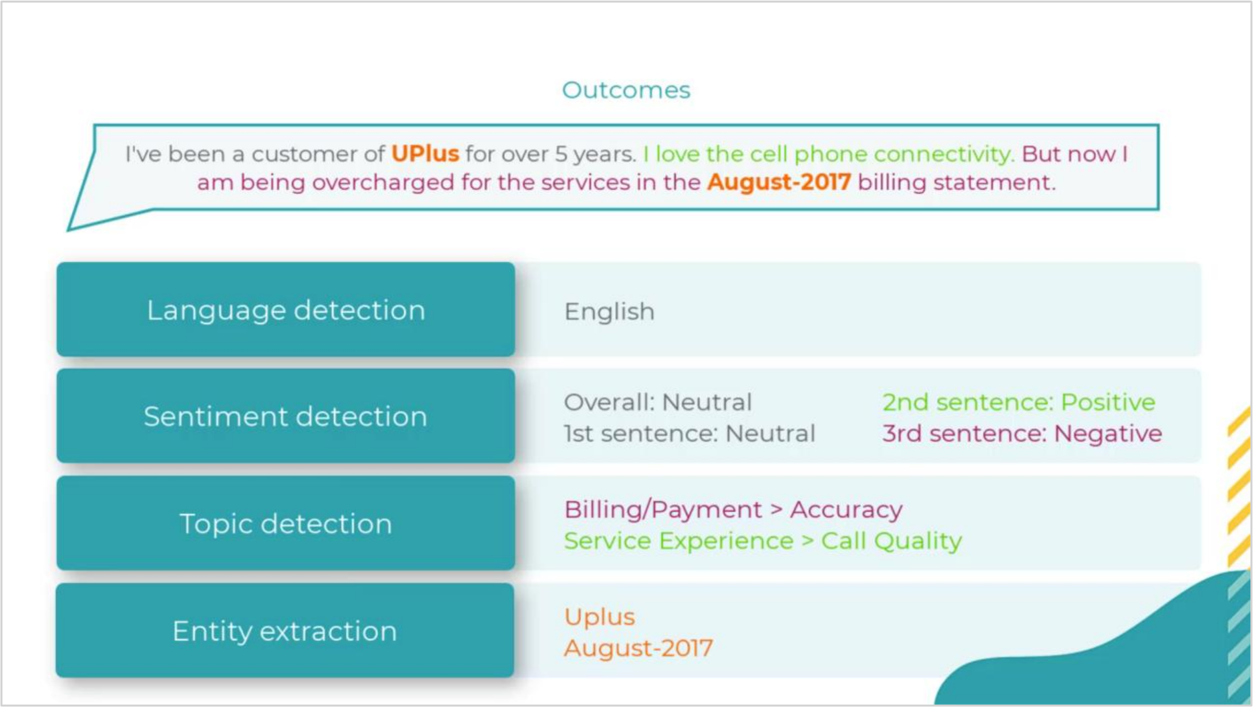


In particular, it is concerned with programming computers to process large volumes of natural language text fruitfully. At a high level, the following analyses occur during natural language processing.

Syntactic analysis analyzes the structure of text and recognizes, for example, nouns, verbs, and adjectives. Semantic analysis establishes the meaning of a piece of text and the relationship between words and sentences. The linguistic analysis phase determines the exact meaning of the text.



The processing of the text from the customer who claims that they were overcharged produces a series of outcomes. First, the analysis detects the language because this action is a prerequisite for the next steps.



Then, the analysis detects the sentiment of the text as positive, negative, or neutral. The sentiment is the general attitude expressed towards a subject by the author of the message. The analysis reports the overall sentiment and sentiment for each sentence.

The analysis classifies the text into one or more predefined topics, such as the business functions of a company. It also reports the sentiment of the corresponding sentence for each category that it detects. Entities refer to the proper nouns found in the text (such as names of people, places, dates and times, and organizations), which helps establish the subject of the text.

You have reached the end of this video. You have learned:

- What text analytics is.
- How text analytics process works.
- What natural language processing is and how it works in text analytics.

Pega NLP overview

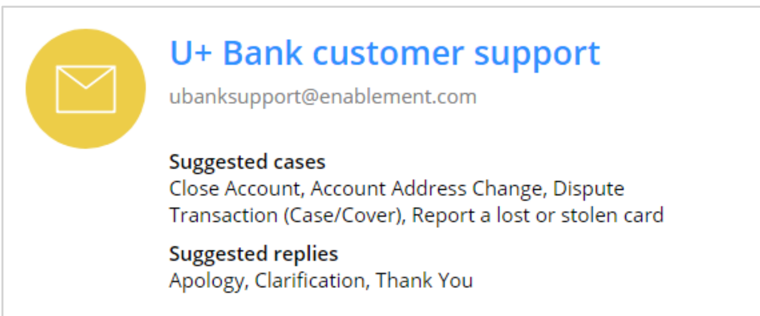
Introduction

Pega natural language processing (NLP) focuses on text categorization and text extraction. Discover how the system uses Pega NLP to route incoming emails based on the topic of the email, and how NLP drives a chatbot to automatically detect the topic of the message, extract necessary entities, create a case, and then route it to the correct customer service workbasket.

Transcript

This video introduces you to Pega NLP, a feature of the decision management capability of Pega Platform. You can apply decision management to any application that uses Pega Platform, which makes it highly versatile.

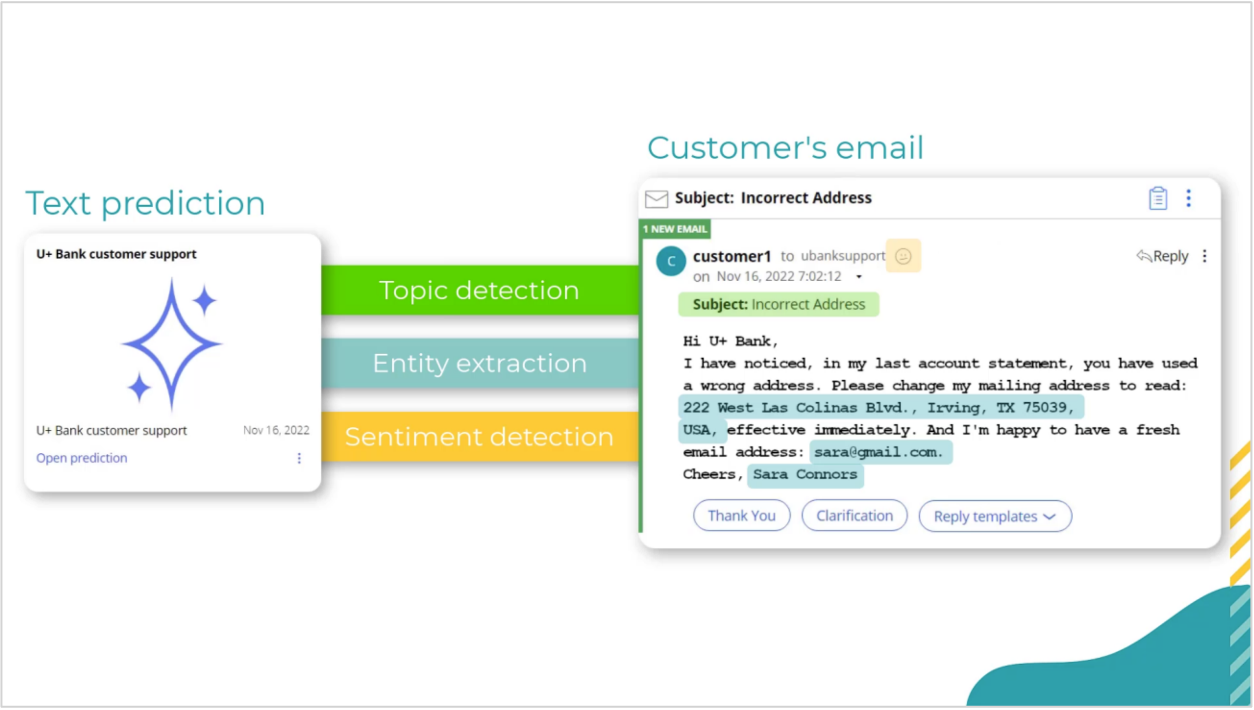
Pega NLP uses text analytics predictions to detect the topic and entities that are included in the message. For example, a company has an email channel where its customers can send any email.



A text prediction drives the email channel. Consider this email sent by the customer through the company's email channel. The text prediction aims to detect the topic of the email and route it to the appropriate department.

Apart from the topic detection model, the prediction also includes an entity extraction model that can extract entities from an email. It can automatically detect and extract certain entities such as an account number, email address, or street address. This functionality allows to automatically process or prioritize emails.

In addition to topic detection and entity extraction, Pega NLP detects the sentiment of an email based on its content. The sentiment score ranges from -1 to 1 and is negative, positive, or neutral.

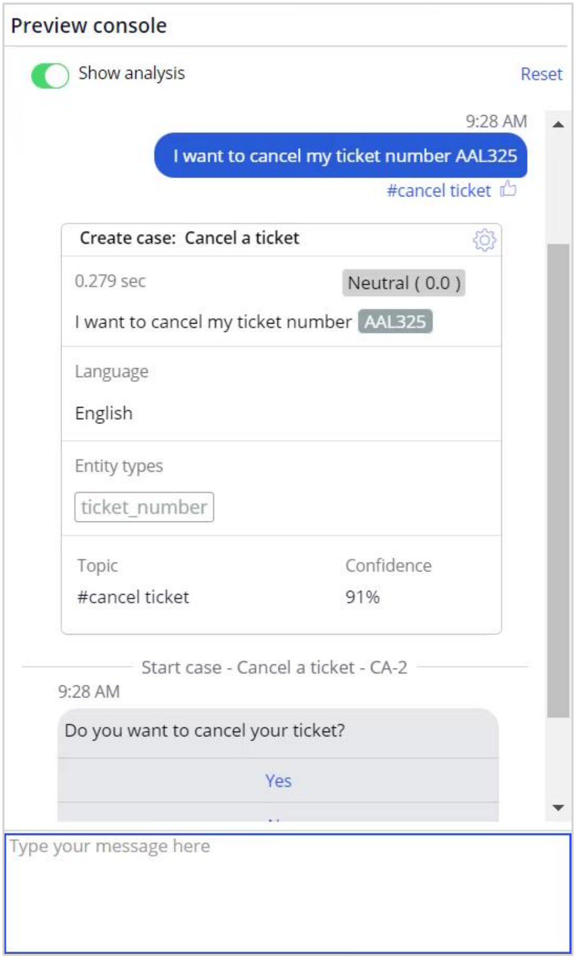


This functionality offloads work from customer service representatives, speeds up the process, and improves the customer experience.

Multiple channels can use the Pega NLP feature. For example, an airline sets up a chatbot channel to speed up customer service by automating the ticket cancellation process.

When Troy, a customer, sends a message to the Airline chatbot that he wants to cancel his ticket, the text prediction detects the topic of the message and runs the `Cancel a ticket` case type that an application developer preconfigured in the system. The case type contains the conversation flow that the chatbot uses in conversation with Troy. The chatbot uses entity extraction models to detect entities such as the ticket number, includes all necessary information in a case, and routes it to a correct customer service workbasket.

In App Studio, you can test the chatbot in the preview console.



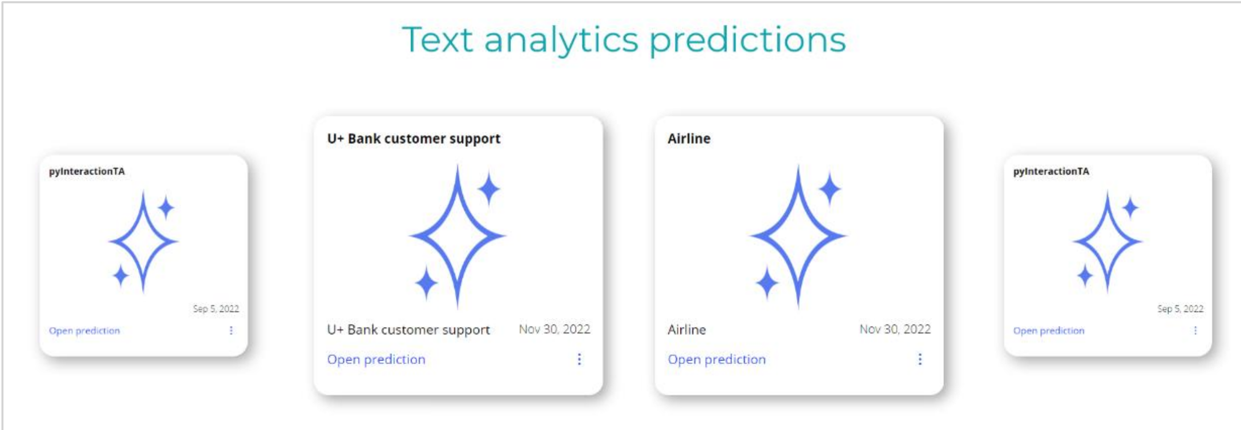
You can turn on the Show Analysis switch to see the chatbot details. Notice the customer's message. The topic detection model detects the topic of this message with 91 percent confidence, and the entity extraction model identifies the ticket number. Chatbot collects this information, includes it in a case, and then automatically routes it to a customer service representative for further processing.

Predictive models drive predictions. A data scientist manages predictions in Prediction Studio.

Prediction Studio is the dedicated workspace in Pega Platform where you manage the life cycle of predictive models and the predictions that they drive.

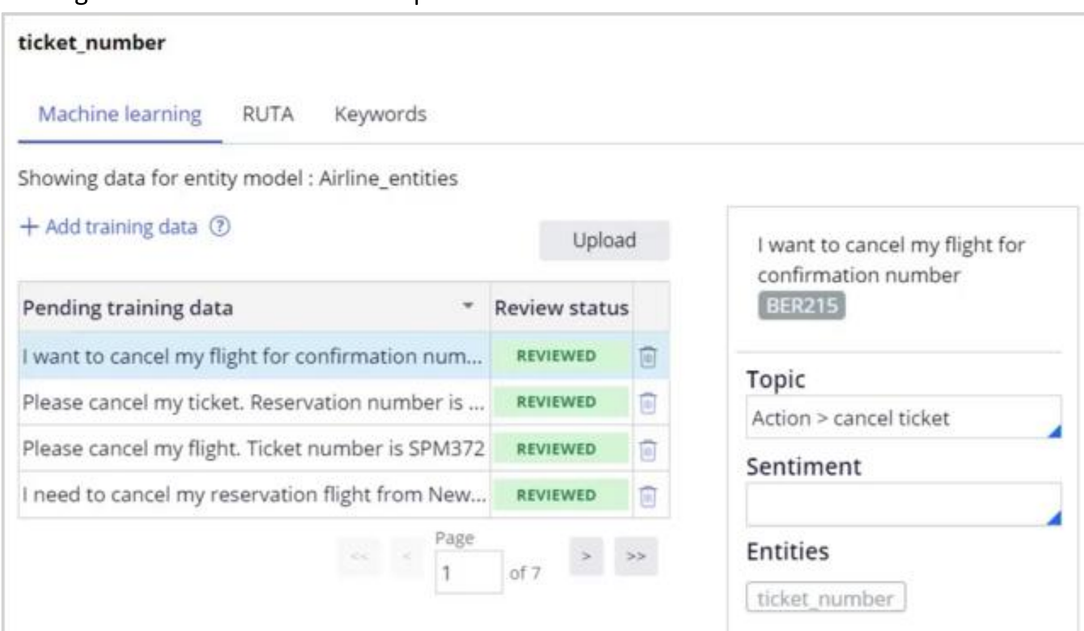
The workspace provides data scientists with everything they need to monitor and change predictions.

This is the Predictions landing page, where you manage predictions. Pega NLP uses Text analytics predictions.



Data scientists build topic detection model in a text prediction by using training data that they add manually or as a batch.

Data scientists also configure entity extraction in a text prediction. Several methods are available, including machine learning, Ruta scripts, or keyword-based entity extraction. It is also possible to use Google AutoML models. The machine learning-based model is the most versatile but requires training data to function. When adding entity extraction training data, you can also specify the topic of each message to further enhance the topic detection model.



You have reached the end of this video. You have learned:

- How Pega NLP allows you to improve business processes by using predictions.
- How topic and sentiment detection models and entity extraction models drive text analytics predictions.

- That machine learning models require training data.

Text analytics for email routing

Description

Humans can effortlessly interpret a single tweet but are unable to parse a large volume of information efficiently. Businesses are exploring ways to use machine learning to extract meaningful information from a large number of text messages. Learn how a text prediction can work to detect topics, extract entities, and identity the sentiment for incoming emails.

Learning objectives

- Explain text analytics
- Describe practical applications where text analytics can be used
- Describe the role of machine learning in text analytics
- Explain how text predictions are trained on classified messages

Email routing

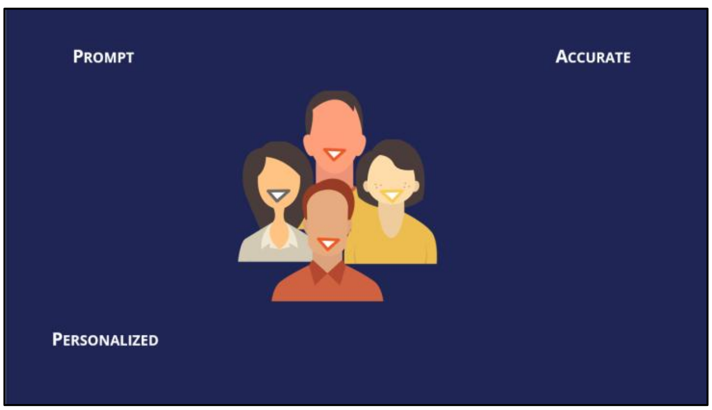
Introduction

A company has an email channel where its customers can send any email — from service requests to compliments to sales inquiries — to the product support team. This email variety makes it difficult to provide each customer with a prompt and personalized response. Pega Infinity™ uses AI-powered text analytics to perform intelligent email routing.

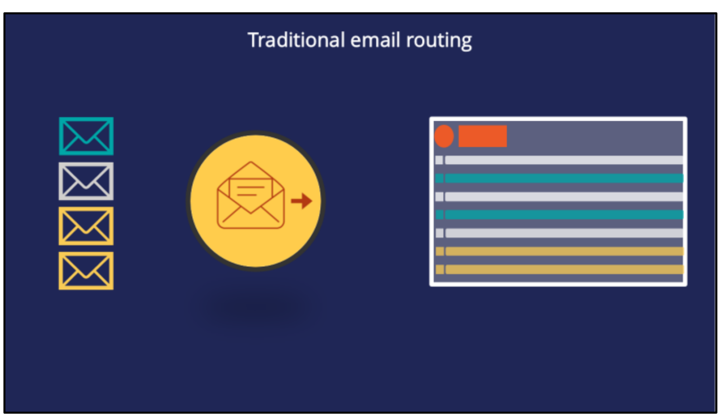
Transcript

In this video, you will learn how Pega Infinity™ uses AI-powered text analytics to do intelligent email routing.

Customer satisfaction is a reflection of what a customer expects from a company vs. what they experience from the company. Meeting, or even better, exceeding customer expectations means addressing their service requests and complaints promptly, accurately, and with personalized solutions. Doing this will ensure the customer has a great experience.

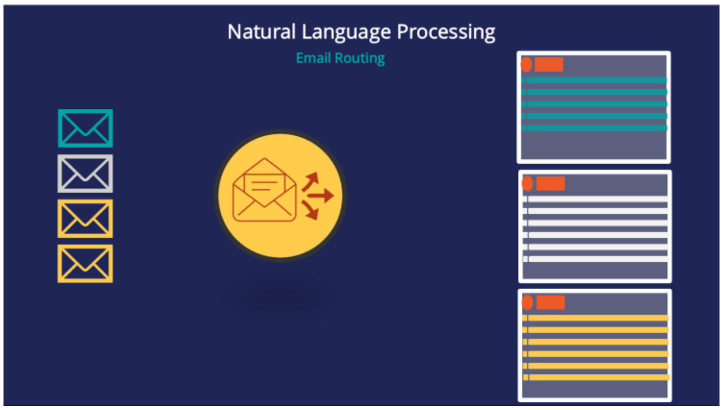


Assume a company has an email channel in which its customers can send any type of email—from service requests to compliments to sales inquiries—to the product support team. These emails are often routed to the same container and are uncategorized. This makes it difficult to provide each customer with a prompt and personalized response.

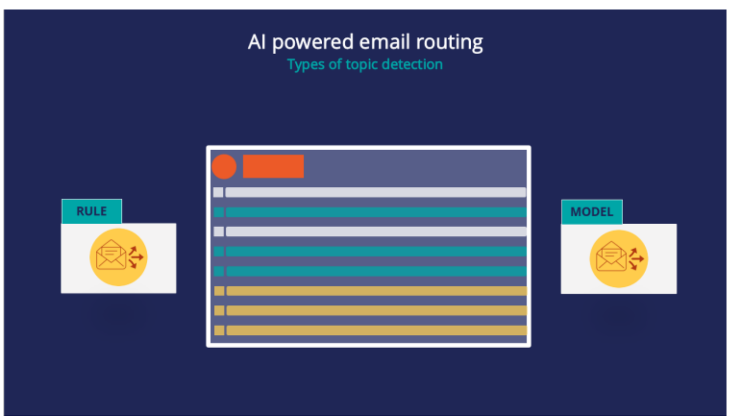


In Pega Infinity, this issue can be addressed by routing the emails using artificial intelligence. Pega Infinity uses AI-powered Natural Language Processing to detect the topic of an email and route the email to the appropriate container.

Consider emails from customers requesting an account address change, making a compliment, or requesting a new credit card. With the help of AI-powered text analytics, Pega Infinity is able to read and understand the content of each email and route it appropriately. This means customer service representatives can be alerted to any account-related service requests and resolve them quickly.

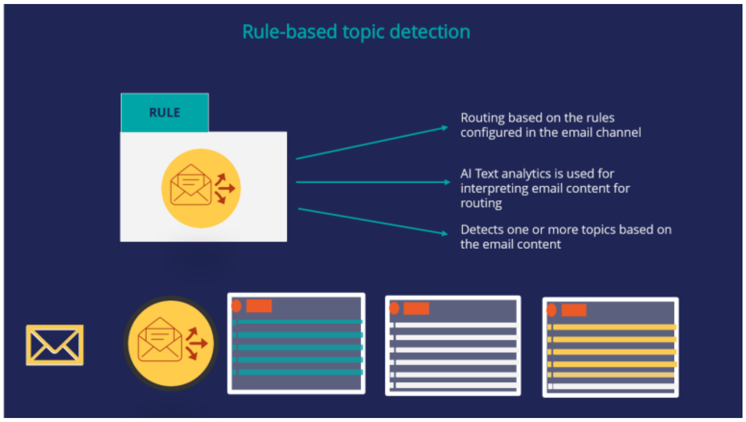


Email routing is done using the topic detection mechanism. The two types of topic detection are rule-based and model-based.



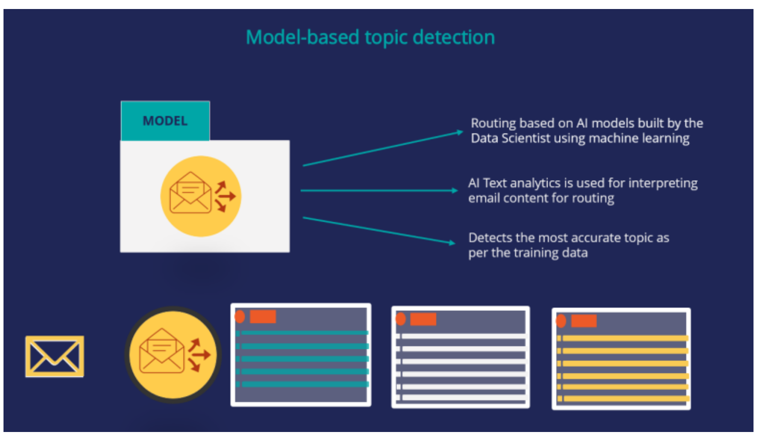
In rule-based topic detection the routing is based on the rules configured in the email channel. AI-powered text analytics is used to detect the topic of the email, and the channel rules route it to the right container. This type of topic detection may detect one or more topics if the email contains words associated with more than one topic.

Let's consider a service request email from a customer. The email content is analyzed and routed to the right container. If an email from another customer contains words that are associated with two topics, the rule-based topic detection detects both topics. The email can then be routed to two different containers depending on how the channel rules are configured.

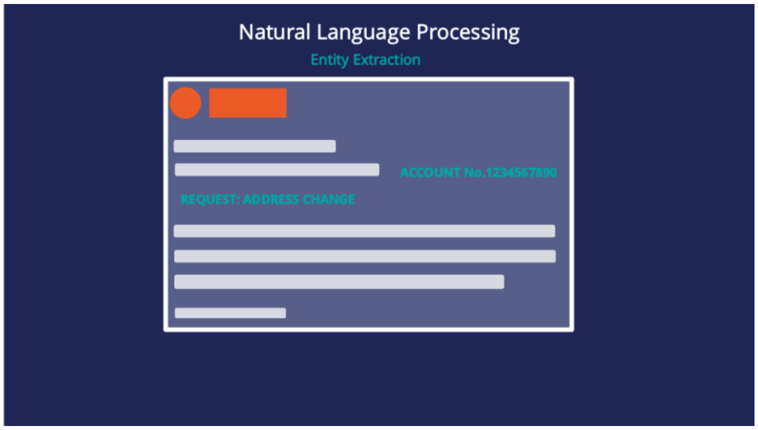


In model-based topic detection the routing is based on AI models built by a Data Scientist using machine learning. Building these models requires a training data set and a test data set. The data sets consist of a list of emails and the associated topic for each email. This type of topic detection identifies the most accurate topic based on the AI model and training set used by the Data Scientist.

Let's consider the same service request email from the customer. The email's content is analyzed and routed to the right container. If an email from another customer contains words associated with two topics during the training of the models, the model-based topic detection detects both topics but typically with a different accuracy factor. In this case, the topic with the highest accuracy factor is chosen.



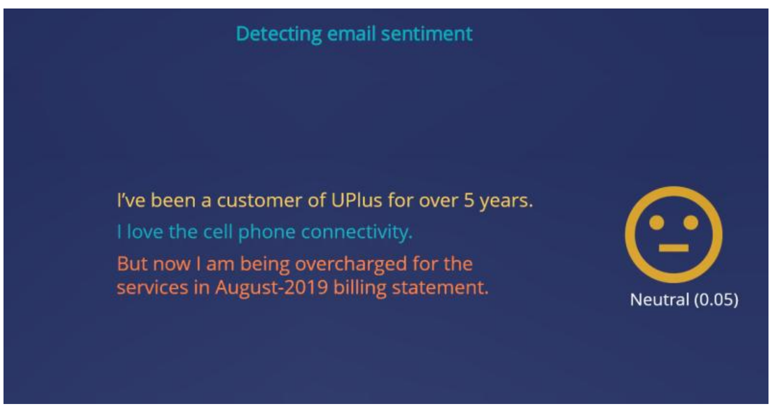
Pega Infinity also enables you to extract entities from an email. This means that when an email is sent, certain entities such as account number, email address, street address, etc. can be automatically detected and extracted. This allows certain emails to be automatically processed or given priority.



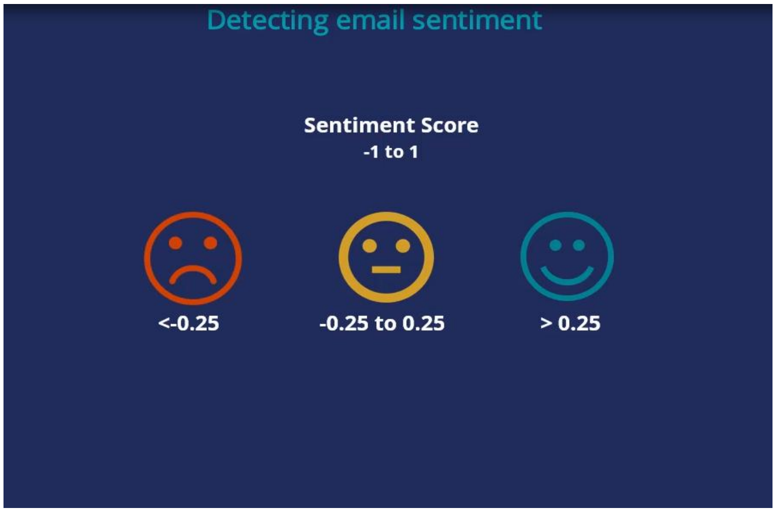
Besides Topic detection and Entity Extraction, Pega Infinity uses its AI-powered text analytics to enable you to detect the sentiment of an email based on its content.

Suppose a customer sends the following email to customer service.

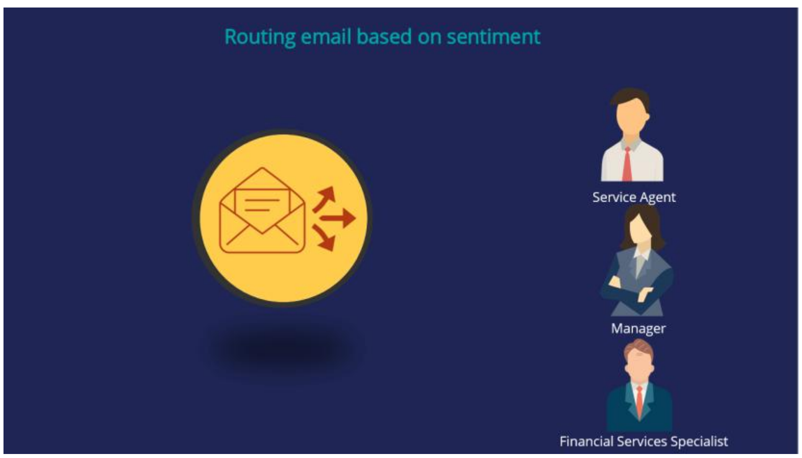
The email's content is a combination of different types of sentences. The first sentence has a neutral sentiment. In the second sentence, the customer expresses his satisfaction with the connectivity, so it has a positive sentiment. The third sentence is negative, as the customer is complaining about an overcharge. The overall sentiment of the email is determined based on the weight of the content sentiments. In this specific example the overall sentiment is neutral with a sentiment score of 0.05.



The sentiment score is a value between -1 and 1. In the out-of-the-box configuration, a sentiment score <-0.25 results in a Negative sentiment, a sentiment score between -0.25 and 0.25 results in a Neutral sentiment, and a sentiment score above 0.25 results in a positive sentiment.



Once the email sentiment is detected, you can configure the email channel to route a specific topic with a specific sentiment to a specialized agent for a quick and personalized response. For example, you could route an address change with a neutral sentiment to a Service Agent, a complaint email with a negative sentiment to a Manager, and a credit card inquiry with a negative sentiment to a Financial Services Specialist.



In summary, Pega Infinity's AI-based email routing capability enables customer service representatives to be more productive, reduces request processing time, and improves the customer experience by providing prompt and personalized service.

Training a topic model to improve email routing

Introduction

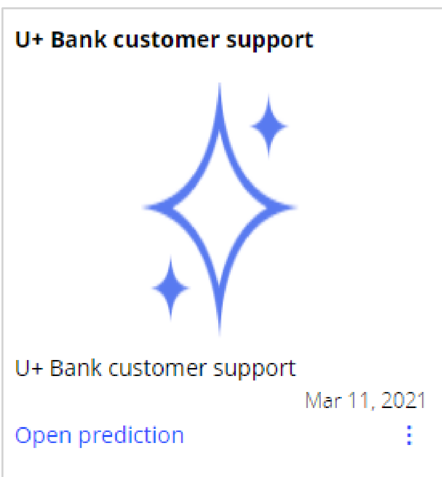
U+ Bank uses Pega Customer Service™ to route incoming emails to the appropriate department based on the topic of the email. For several use cases (for example, an address change), emails are routed based on keywords that are detected in the message. To improve the email routing, learn how to train the text prediction with a data set that contains classified messages.

Transcript

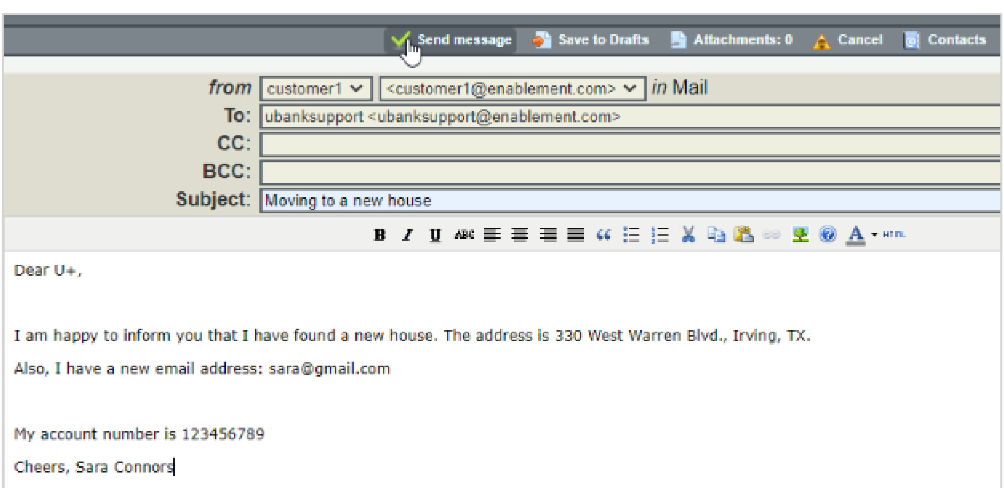
This demo shows you how to train a text prediction to improve email routing.

U+ Bank uses machine learning to route inbound messages in the email channel to the appropriate department based on the topic of the email.

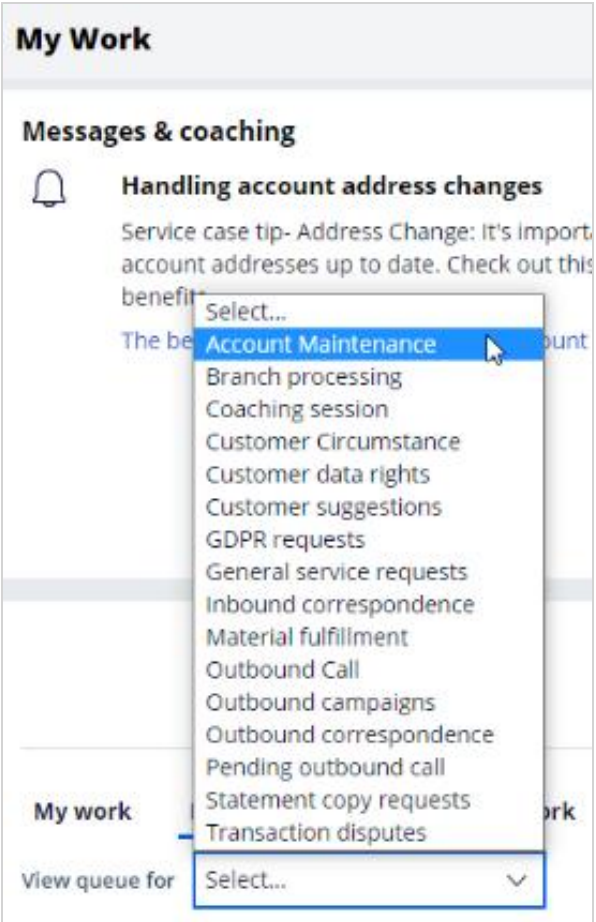
A text prediction that aims to detect the topic of the message drives the routing.



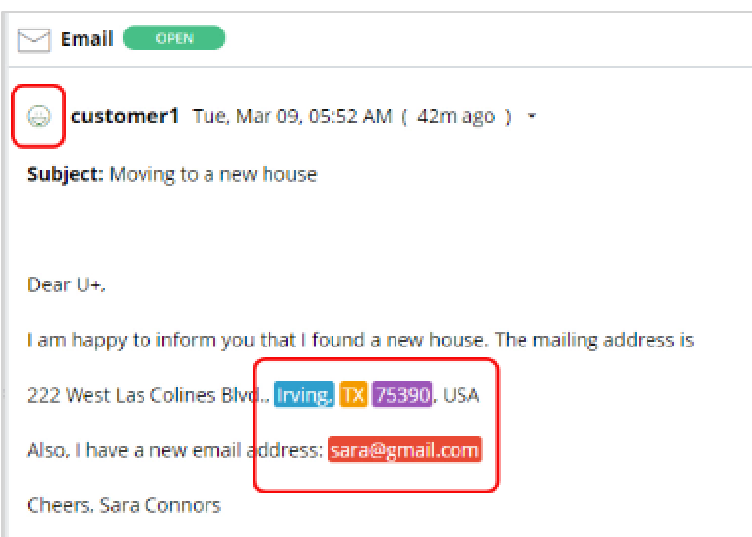
For example, when Sara writes an email to inform U+ Bank that she has moved to a new house, the text prediction detects an address change as the topic.



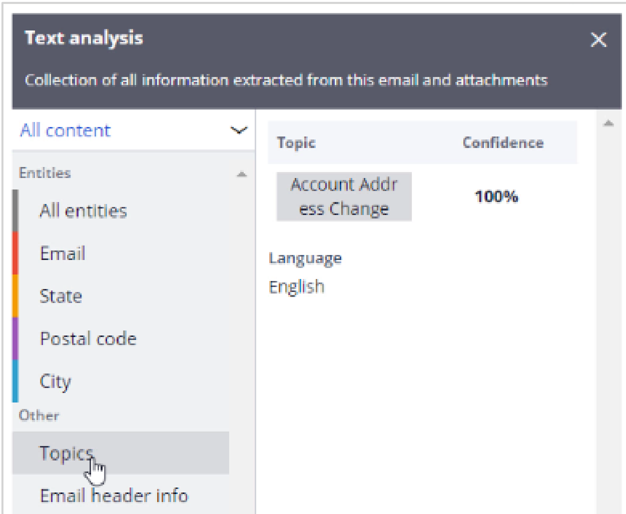
The **Account maintenance** department receives all emails where an address change is detected as the topic of the message.



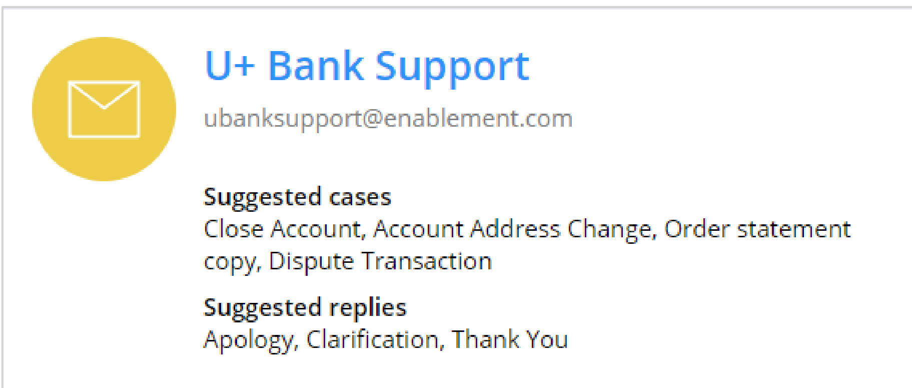
The text prediction also detects entities such as a ZIP Code and an email address and the overall sentiment of the message.



Notice that the topic Account Address Change is detected with 100% confidence.



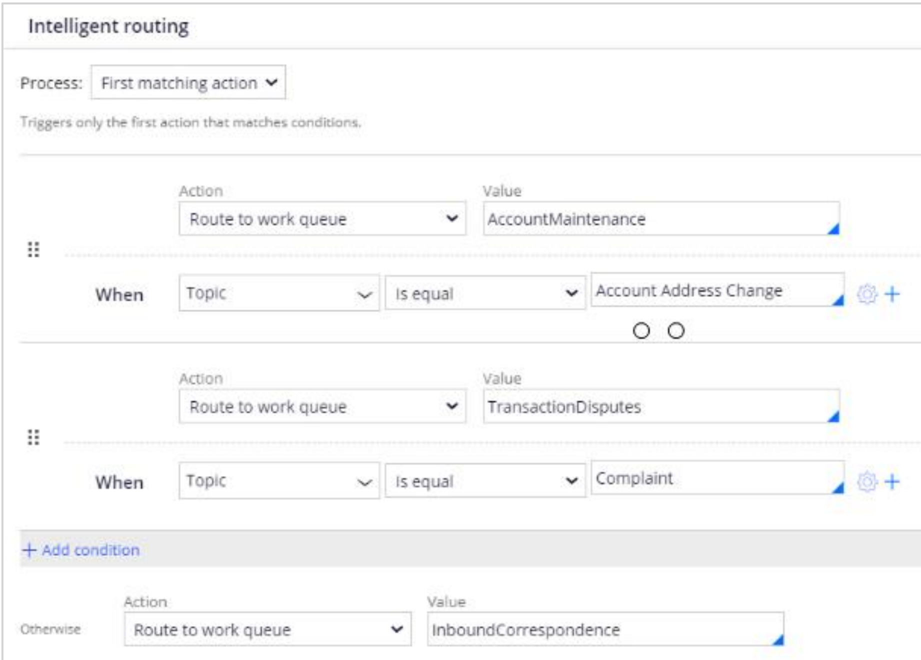
The intelligent routing is set up in the email channel, in App Studio.



An email is routed to the work queue of the Account maintenance department when the detected topic is an address change.

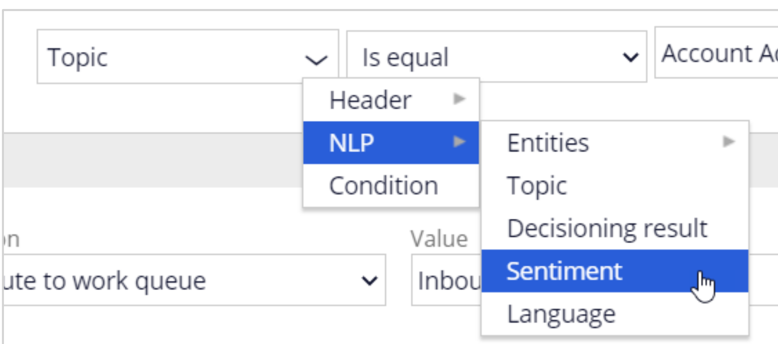
Complaints are routed to the **Transaction Disputes** department.

If the address change and complaint topics are not detected, the email is routed to a default work queue.

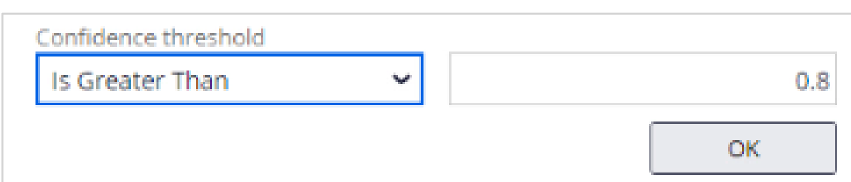


Note that in the **When** condition of the email routing, you use the outcome of the text prediction.

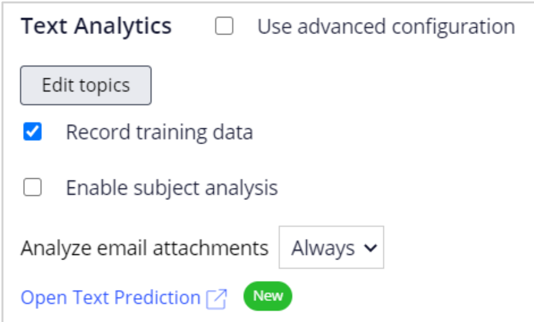
Optionally, you can also use the sentiment of the email in the routing conditions.



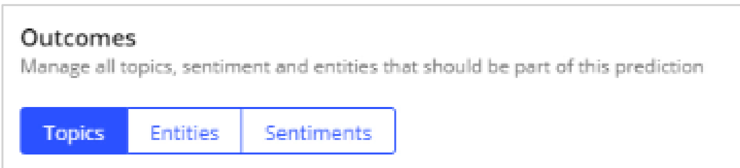
You can set a threshold for the confidence with which a topic must be detected to trigger routing.



Every channel is associated with a text prediction.



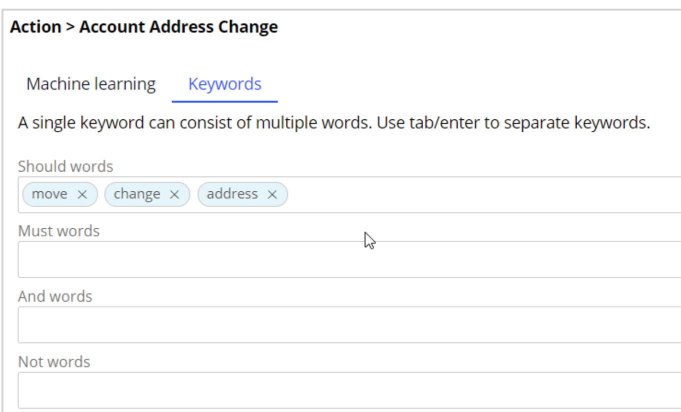
A text prediction is driven by predictive models that detect topics, entities, and sentiments.



Many entity extraction models, and a sentiment model, are available out of the box.

For topic detection, Prediction Studio supports keyword models as well as machine learning models based primarily on training data.

A keyword model uses **Should words**, **Must words**, **And words**, and **Not words**.



If any of the **Should words** appear in a piece of text, topic detection assigns that text to the corresponding topic. To achieve accurate results, create an exhaustive list of **Should words**.

Only if all **Must words** appear in a piece of text, will the topic detection assign that text to the corresponding topic.

Use **And words** to distinguish between similar topics while using identical **Must words**.

If a **Not word** appears in a piece of text, the text is not assigned to the corresponding topic.

You can test the output of a prediction on a sample message.

Two topics are correctly detected: an address change and a complaint.

Notice that the confidence score for both topics is **1**. The keyword model performs a Boolean match based on the presence or absence of words and detects the topic with absolute certainty.

Topic detection

Granularity	Analysis
Document	Rule

Topic	Sentiment	Sentiment score	Model name	Model type	Confidence score
Action > Account Address Change	Neutral	0	pyInteractionTaxonomy	Pega NLP	1.00

It is highly recommended to use a machine learning model for topic detection. It determines the confidence score based on evidence.

A machine learning model is based on training data that consists of categorized texts.

You can get training data from two sources. The first is training data accumulated through the application that is running in production.

When a customer service representative corrects the topic of an incoming email, this change is added to the training data.

After the messages are reviewed and approved, they are used to rebuild the topic model.

Also, you can choose to import data, provided you have accumulated data in the past.

The file must contain the message and the associated topic.

	A	B	C
1	**content**	**result**	**type**
20	Effective 11/28/17 please change mailing address to	Action > Account Address Change	Test
21	Please amend the mailing/billing address to the follov	Action > Account Address Change	Test
22	Confirm - mailing address: 1430 E 33rd St., Signal Hill	Action > Account Address Change	Test
23	Effective 8/28/17, please change the insured's mailin	Action > Account Address Change	Test
24	Per our insureds request, please issue the following c	Action > Account Address Change	
25	Effective 8/25/17, please correct the mailing address	Action > Account Address Change	
26	Effective 8-16-17 please amend the mailing address a	Action > Account Address Change	
27	Please amend the mailing address.	Action > Account Address Change	
28	Effective 8/28/17 please change the mailing and billir	Action > Account Address Change	
29	AMEND insured's MAILING ADDRESS to read:	Action > Account Address Change	
30	Effective: 09/01/2017 Please amend mailing address	Action > Account Address Change	
31	Please update mailing address per the below:	Action > Account Address Change	
32	We have received a request to change your mailing a	Action > Account Address Change	
33	Please change mailing address to read: 197 Warren A	Action > Account Address Change	
34	Please change the mailing address as per the attache	Action > Account Address Change	
35	Please amend mailing & billing address to 3253 Phillip	Action > Account Address Change	

Notice that training data for the address change and the complaint topics are in the pending state.

You can now rebuild the models.

You can select individual models or rebuild all models across model types and languages.

While training is in progress, you have the option to cancel.

When the process completes, you can view the latest model report on the **Models** tab.

The report contains the validation data, the confusion matrix, and the score sheet.

The topic detection now uses machine learning models based on the training data and the keywords provided for the address change and complaint topics.

Outcomes

Manage all topics, sentiment and entities that should be part of this prediction

Topics | Entities | Sentiments

Language

English

Topics	Total keywords	Total training data	F-score
Action > Account Address Change	10	82 (0 pending)	1.00
Action > Close Account	10	0 (0 pending)	
Action > Complaint	7	84 (0 pending)	1.00
Action > Report a lost or stolen card	13	0 (0 pending)	

The **Should words** and **Must words** act as positive features for matching text to a topic, while the **Not words** act as negative features.

But the training and testing data have the greatest impact on your machine learning model, while keywords have a smaller impact.

Once the models are rebuilt, you can test the prediction.

Notice that the topics are detected with a confidence score below one.

In this case, the topic message is recognized as a complaint with high confidence. The address change topic is detected, but with a lower confidence score.

When multiple topics are detected in a message, comparison of the confidence scores allows selection of the one with the highest priority.

Topic	Sentiment	Sentiment score	Model name	Model type	Confidence score
Action > Complaint	Neutral	-0.25	U+ Bank customer support	Pega NLP	0.75
Action > Account Address Change	Neutral	-0.25	U+ Bank customer support	Pega NLP	0.25
Action > Report a lost or stolen card	Neutral	-0.25	U+ Bank customer support	Pega NLP	0.00

The other outcomes of the text prediction are the entities detected, such as an email address and the sentiment of the message.

You have reached the end of this demo. What did it show you?

- How Pega Customer Service routes incoming emails to the appropriate department based on the topic of the email.
- How text predictions work to predict the topic and sentiment of a message and detect entities.
- How to train a topic model and use machine learning to identify the topic correctly.

Using entity extraction with chatbot channel

Description

Better understand the key features and benefits of entity extraction with chatbot channel. Use this module to learn how to enable the chatbot to handle ticket cancellation requests by training a text prediction that drives the chatbot. The chatbot detects the topic of a message, extracts all relevant entities, and creates a case that the Customer Service Representative (CSR) in the CSR portal can handle later.

Learning objectives

- Describe how Pega Chatbot uses natural language processing to determine the topic of inbound message and extract entities
- Configure the chatbot channel
- Explain the out-of-the-box entity extraction models
- Train the topic detection and entity extraction models

Training predictions for chatbot

Introduction

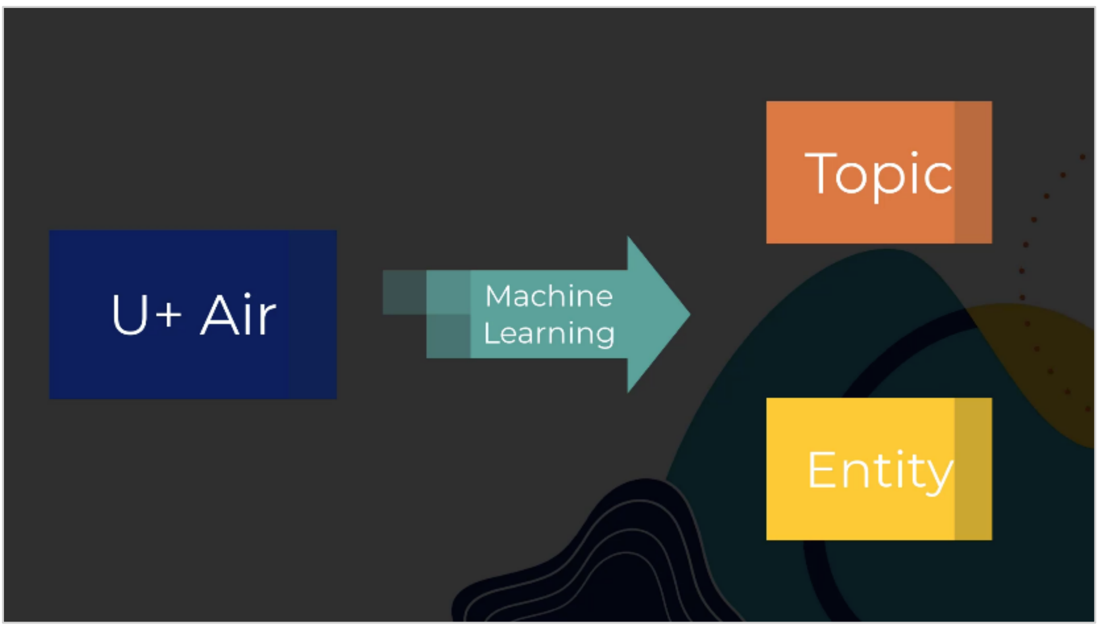
Web Chatbot is a feature in Pega Customer Service™ that allows your customers to enter questions or requests in a chat window that is accessible from a customer-facing web page, and then receive answers and guidance.

In a web chatbot session, the chatbot responds to a customer request, such as *I want to cancel a ticket*, by detecting the right topic with machine learning. The chatbot then walks the customer through a set of questions. The chatbot gathers the required information by leveraging entity extraction models. Learn how to train these topic detection and entity extraction models.

Transcript

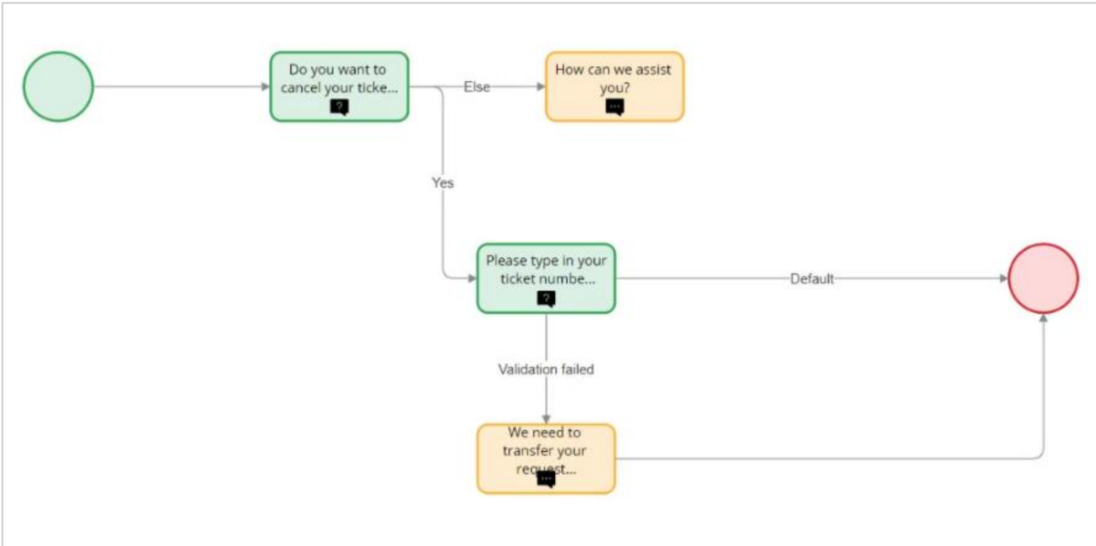
This demo shows you how to train a text prediction to enable the chatbot to handle ticket cancellation requests.

U+ Air uses machine learning to identify the topic of the incoming message and detect entities such as ticket numbers.



A text prediction that aims to detect the topic and entities included in the message drives the chatbot.

For example, when Troy, a customer, sends a message to the Airline chatbot that he wants to cancel his ticket, the text prediction detects the topic of the message and runs the *Cancel a ticket* case type that an application developer preconfigured in the system. The case type contains the conversation flow that the chatbot uses in conversation with Troy. Notice the conversation elements such as *Ask a question* or *Send a message*. Observe how the chatbot follows the conversation flow during interaction with Troy.



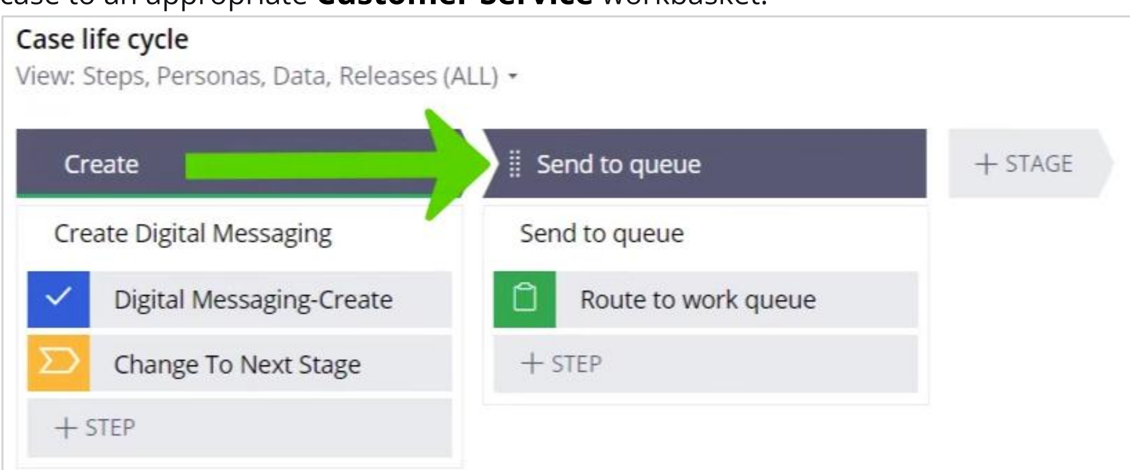
The prediction detects the ticket number entity that Troy provides, and then asks for confirmation.

Notice that the chatbot skips the *Please type in your ticket number* question because the text prediction detected that Troy provided the ticket number in the first message.

See how the chatbot reacts if Troy does not provide the ticket number in the first message.

Instead of skipping the question, the chatbot asks Troy to provide the ticket number.

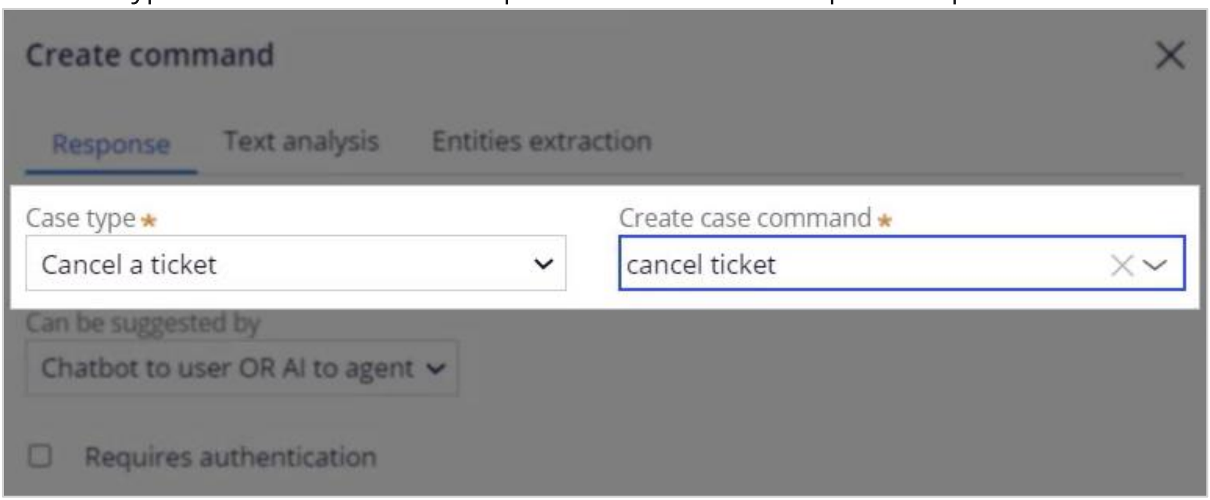
Once the conversation ends, the chatbot saves the necessary information and routes the case to an appropriate **Customer Service** workbasket.



Note that the case contains the ticket number extracted by the text prediction during the conversation with Troy.

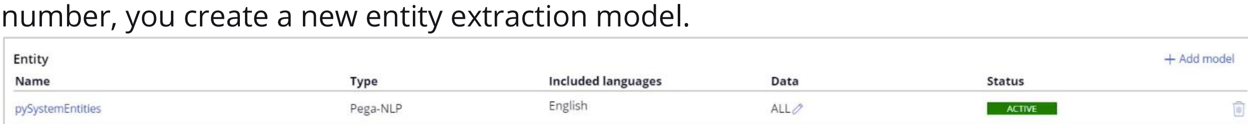
To make the chatbot available to customers, you first configure the digital messaging channel in App Studio.

Notice that the *Cancel a ticket* case type is specified. This is the same case type that contains the conversation flow that the chatbot uses in the conversation with Troy. The **Create case command** represents the topic in the text prediction. The chatbot triggers the case type creation when the text prediction detects this specific topic.



The topic detection and entities extraction are driven by a text prediction that is automatically generated after chatbot channel creation. Data scientist builds the models in the prediction in Prediction Studio. Models learn with training data. In this scenario, the training data contains three topics: *action book ticket*, *action cancel ticket*, and *action reschedule*. Besides topic detection models, the prediction contains entity extraction models that extract necessary information from the incoming chat message.

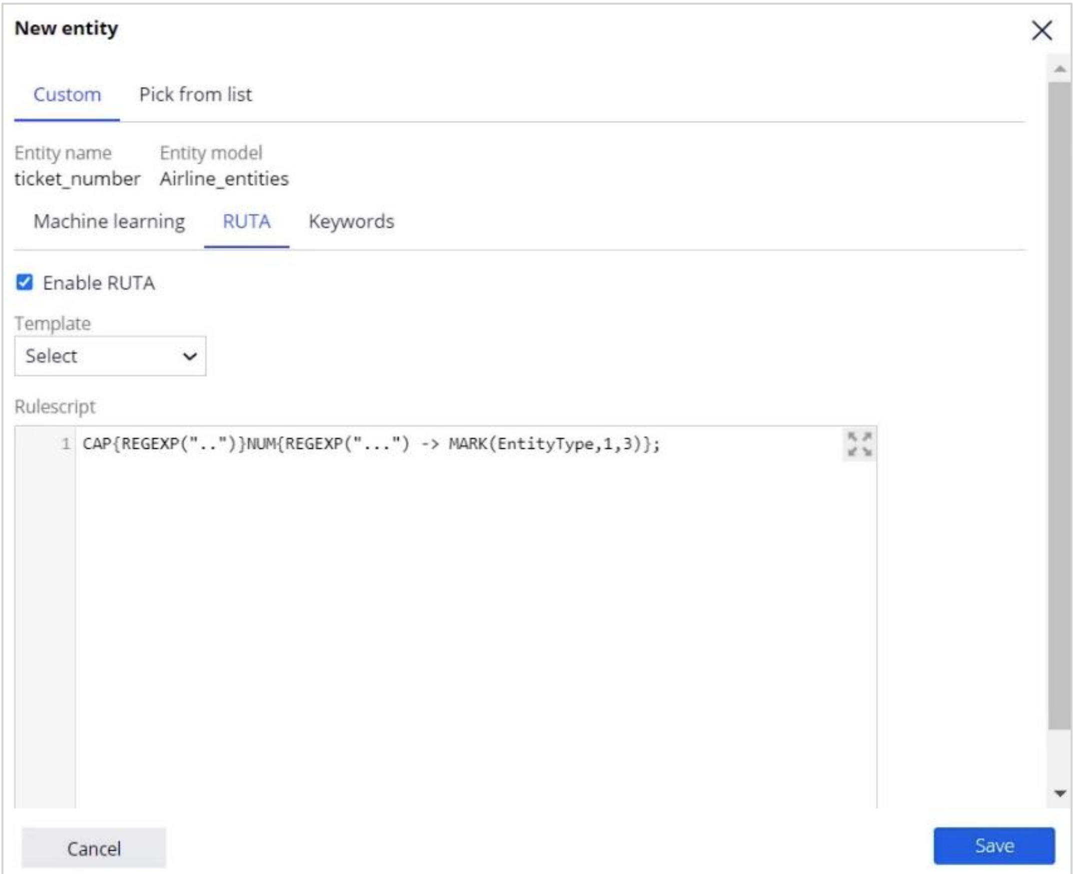
Notice the existing *pySystemEntities* model. It is an out-of-the-box entity extraction model that contains basic entities; you cannot modify it. To add a new entity type such as ticket number, you create a new entity extraction model.



After creation of the new model, you can create a new entity.

U+ Air determines that using Rule-based Text Annotation (Ruta) scripts is the best choice to detect the strict ticket number pattern. Ruta is a rule-based script language which detects keywords and phrases that follow specific patterns.

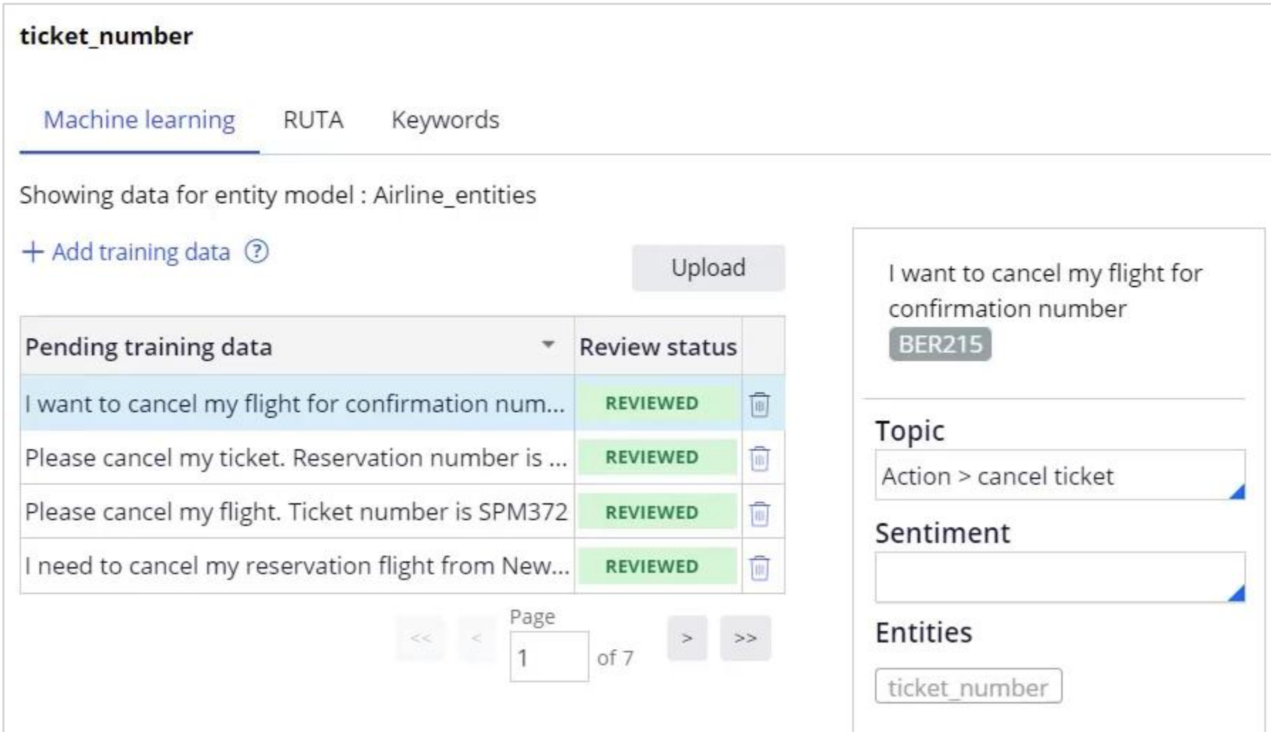
Notice the rulescript that detects the ticket number. This script detects ticket numbers that follow a strict pattern of two letters followed by three digits. In case of a rescheduled ticket or human error, the application also needs to detect unusual ticket number patterns. For example, a dash in between the letters and numbers, or three letters and three digits. Using Ruta with machine learning can achieve this business outcome.



A machine learning-based model must be built to enable machine learning entity extraction. You add training data to the ticket number entity to build the model; you can add training data one record at a time.

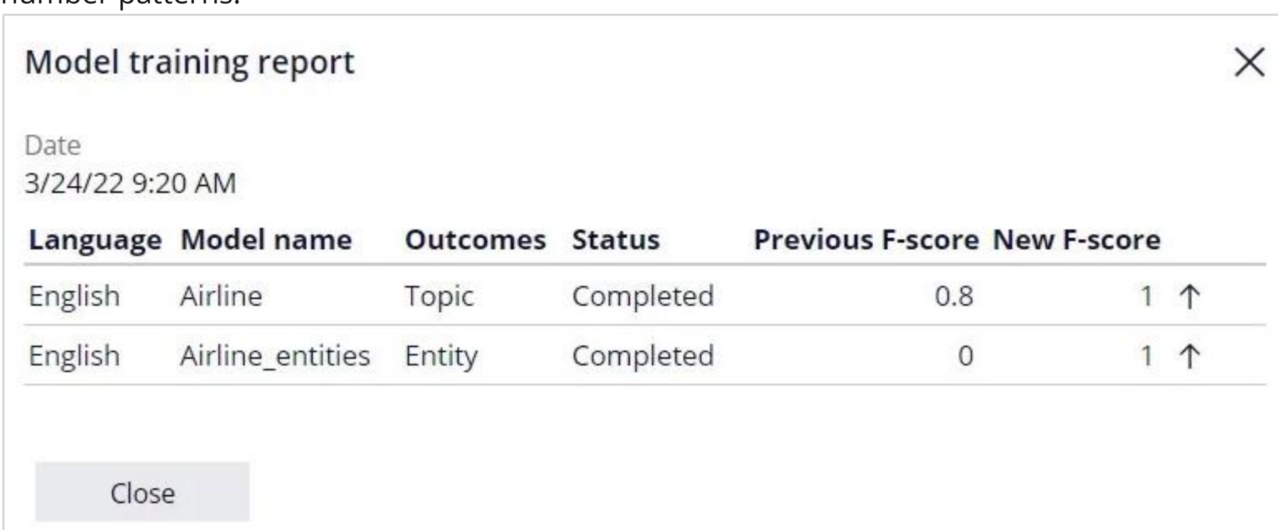
A training data record is a typical chat message that the customer sends. After adding a training data record, you manually identify the entities you want the model to recognize. In this case, JK-294 is a ticket number entity. Optionally, to improve the topic detection model, you can specify the topic of the message that provides additional training data.

You can also use a batch import from a dataset file to provide additional training data. Notice the tagging that identifies a specific entity in each record. It is possible to have multiple tags to train multiple entity types simultaneously.



After uploading the training data file, you can review each record and ensure that it was correctly imported. The correct ticket number entity and topic are flagged.

After uploading the dataset, build the models. Notice the New F-score for the Airline and Airline_entities models. Now, the entity extraction model can also detect unusual ticket number patterns.


Model training report

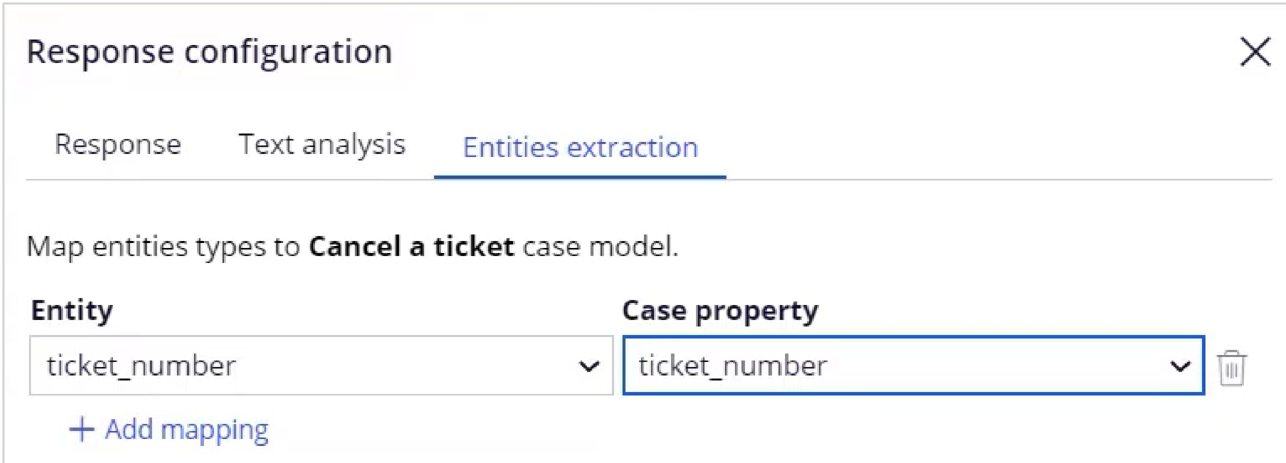
Date
3/24/22 9:20 AM

Language	Model name	Outcomes	Status	Previous F-score	New F-score
English	Airline	Topic	Completed	0.8	1 ↑
English	Airline_entities	Entity	Completed	0	1 ↑

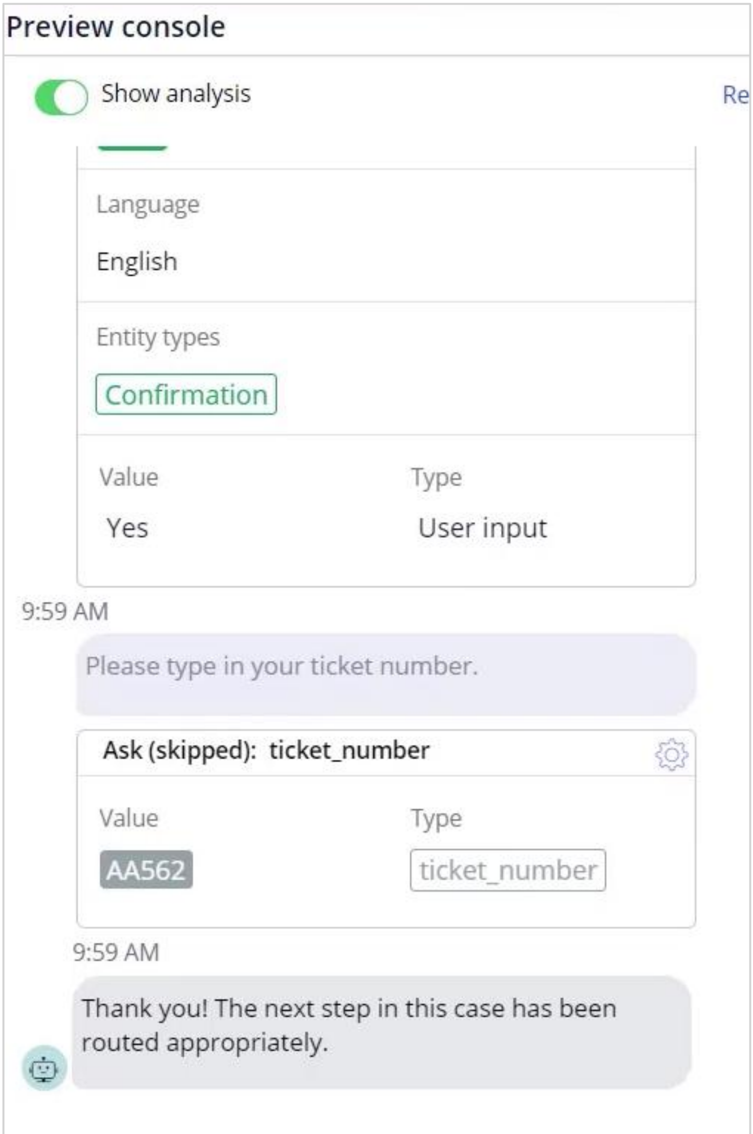
Close

To enable the chatbot to use the extracted entity in the created case, an application developer maps the entity to a case property in a case command.

The system uses this property in the *Cancel a ticket* case type.



With all models in the text prediction built, and all necessary properties mapped, it is possible to see in full detail how Pega natural language processing (NLP) works. The digital messaging interface has a built-in preview console that simulates the chatbot. After sending the *I want to cancel my ticket number AA562* message, you can turn on the **Show Analysis** switch to see the chatbot details. In this case, the chatbot detects the topic of the message with 94% confidence. The confidence score, which is a value from 1 to 100, represents the confidence of the prediction. The chatbot detects the ticket number entity in the first message, skips the ticket number question, and then routes the case to a correct workbasket.



The case that the chatbot creates is available for review by the customer service representative in the CSR portal. Notice that because of the detected topic, the system creates a *Cancel a ticket case*. The case contains the ticket number extracted by the text prediction during the conversation with the customer.

You have reached the end of this video. What did it show you?

- How to create a chatbot channel.
- How to create a topic detection model.
- How to create an entity extraction model with machine learning and Ruta script.