



Adobe Experience Platform

Data Science Workspace Overview – Nov 2019

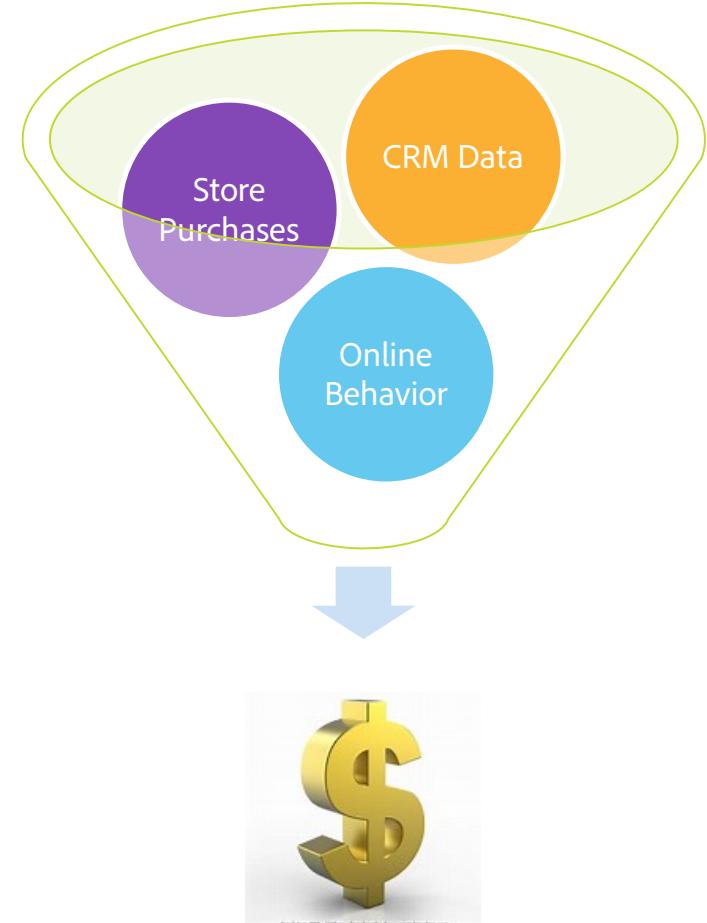
Agenda

- What is Data Science
- DSW Features
- DSW Terminologies
- How it works - Demo
- Use Cases
- Q&A

What is Data Science?

“almost **everything** that has something to do with **data**”*

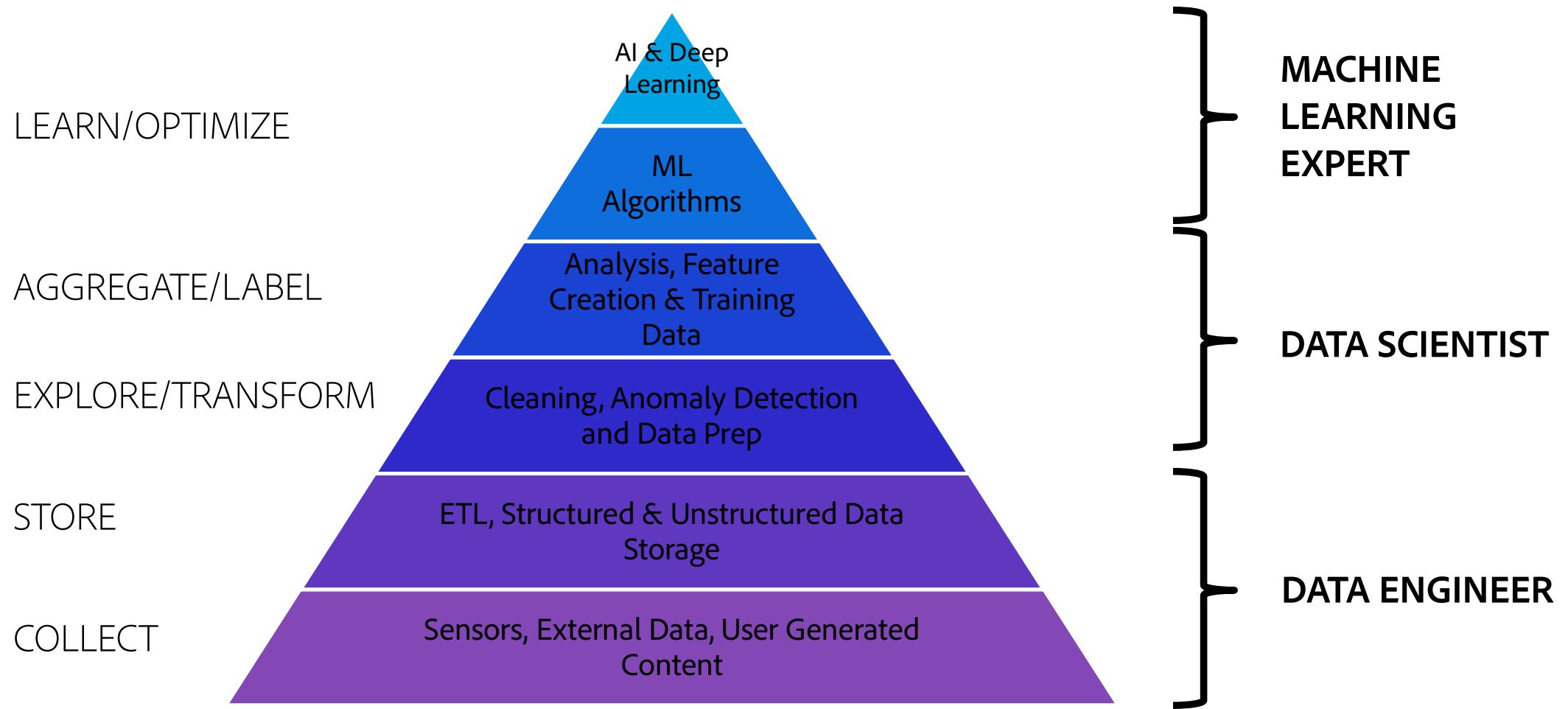
“Data science is the study of **where** information comes from, **what** it represents and **how** it can be turned into a **valuable** resource in the creation of business and IT **strategies**.**



* Journal of Data Science

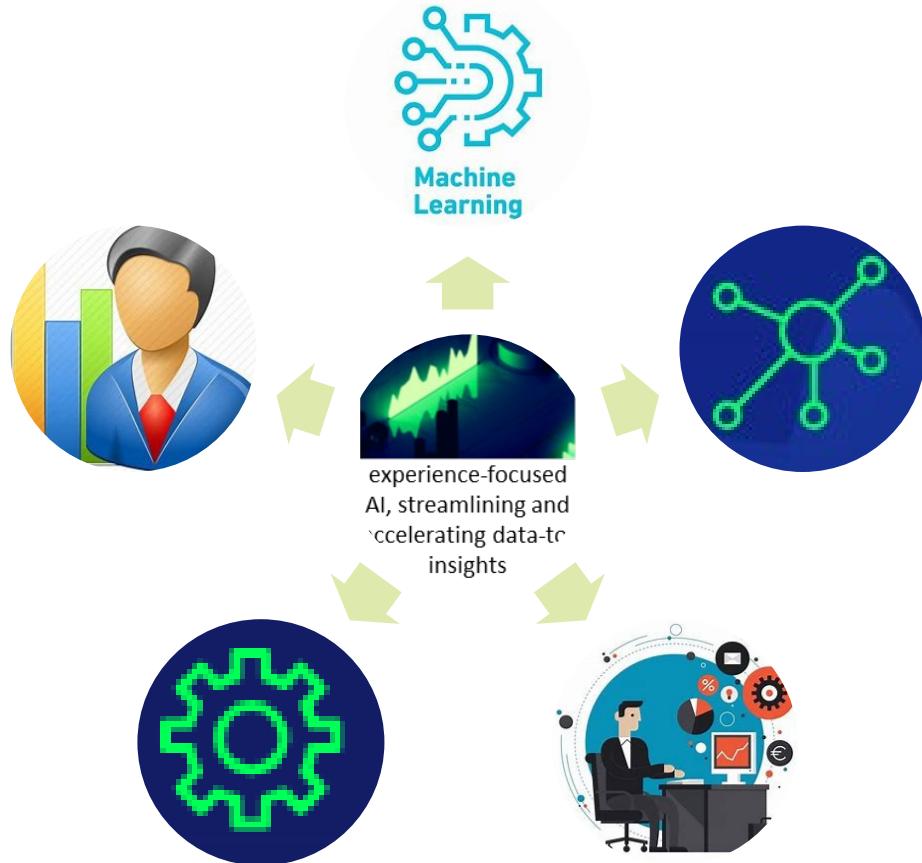
** <https://searchenterpriseai.techtarget.com>

Data Science Hierarchy of Needs



Data Science Workspace

Building Intelligent Services on Adobe Experience Platform



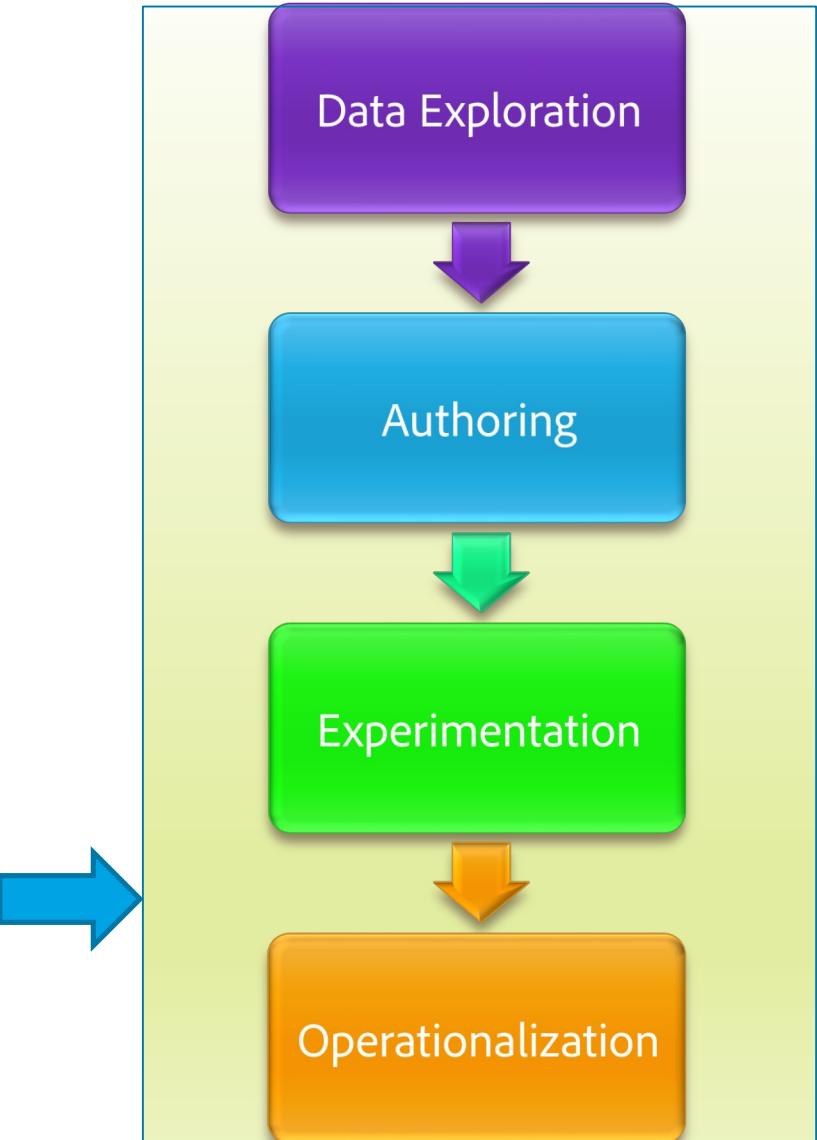
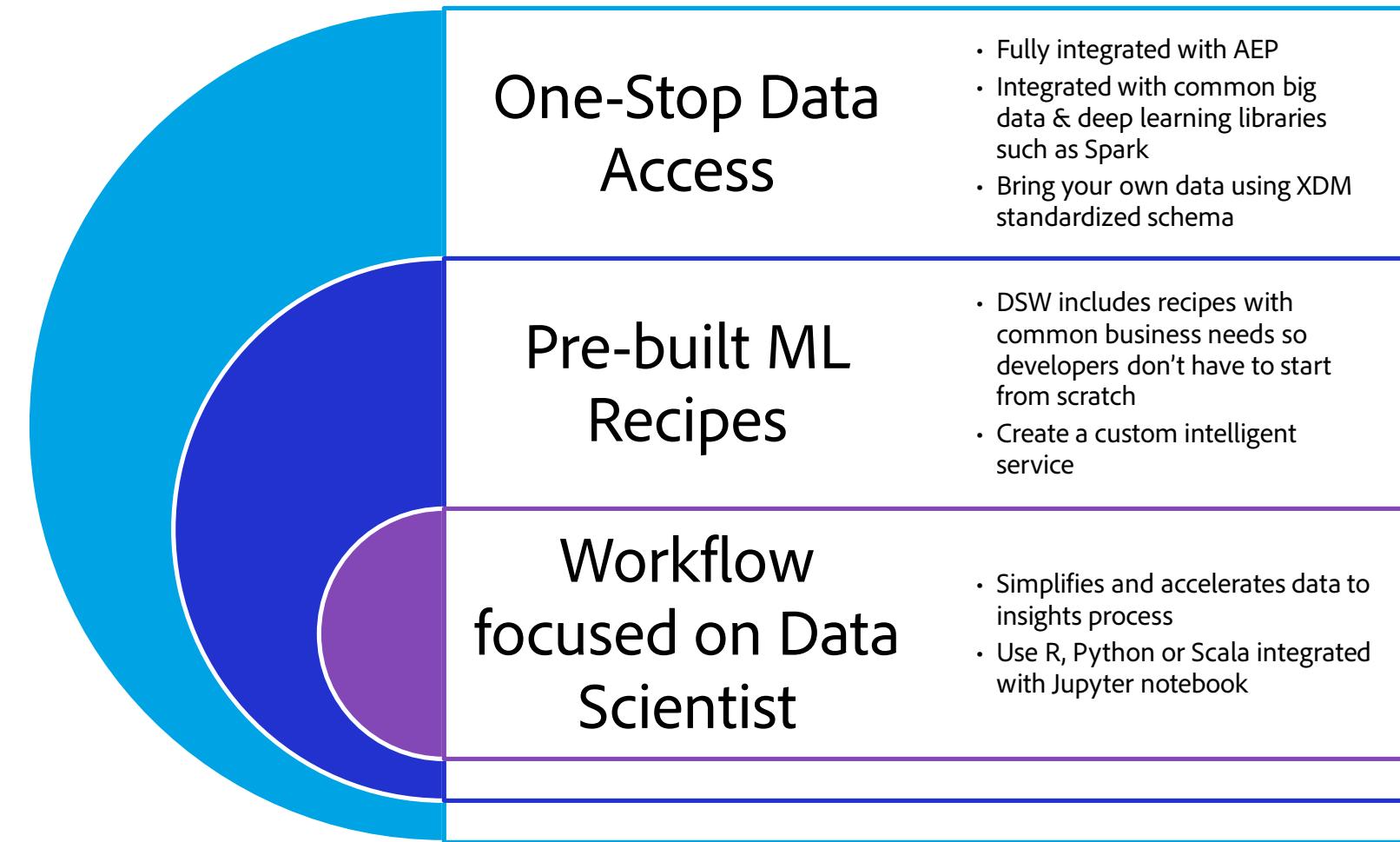
With DSW, AEP allows you to streamline and accelerate data-to-insights with

- ✓ Integrated access to data stored in Platform
- ✓ The computing power essential for machine learning/AI & managing big datasets
- ✓ Prebuilt ML recipes to accelerate into AI-driven experiences
- ✓ Simplified authoring, reuse, and modification of recipes for data scientists of varied skill levels
- ✓ Intelligent service publishing and sharing in just a few clicks - and monitoring & retraining for continuous optimization of personalized experiences

Data scientists of all skill levels will achieve insights faster and more effective digital experiences sooner.

Why Data Science Workspace?

Data to insights



DSW Terminologies?

Lets look at some terminologies used in Data Science Workspace

Algorithm

Standard DS techniques such as classification, regression, k-means clustering, etc.

Feature

An individual measurable property or characteristics of a phenomenon being observed

Feature Engineering

Process of converting raw data into usable form for analysis using domain knowledge

Recipe

A propriety algorithm or an ensemble of algorithms to help solve specific business problems

Instance

An occurrence of the recipe configured with the right data definition to help solve specific biz problems – one recipe can create many instances

Trained Model

An instance of the recipe that is trained using historical data to learn from. The trained model finds patterns in the training data to predict the target

Service

Service that is created from a "trained model" to be used in building experiments

Jupyter Notebook

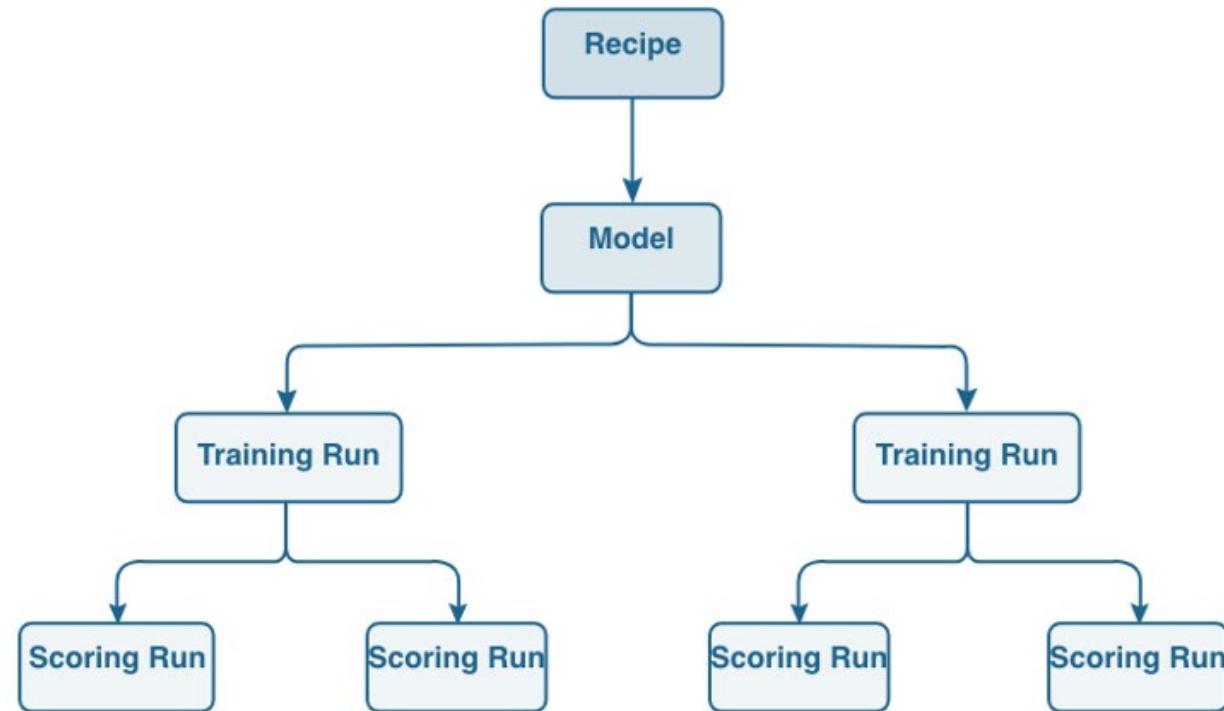
An open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text

Hyperparameters

High level properties of a model, different from standard model parameters that are usually fixed.
Ex: depth of decision tree, number of hidden layers, learning rate, etc.

Relationship between Model, Recipe and Experiments

The following chart outlines the hierarchical relationship between Recipes, Models, Training Runs, and Scoring Runs.



How does it work?

Define the Problem

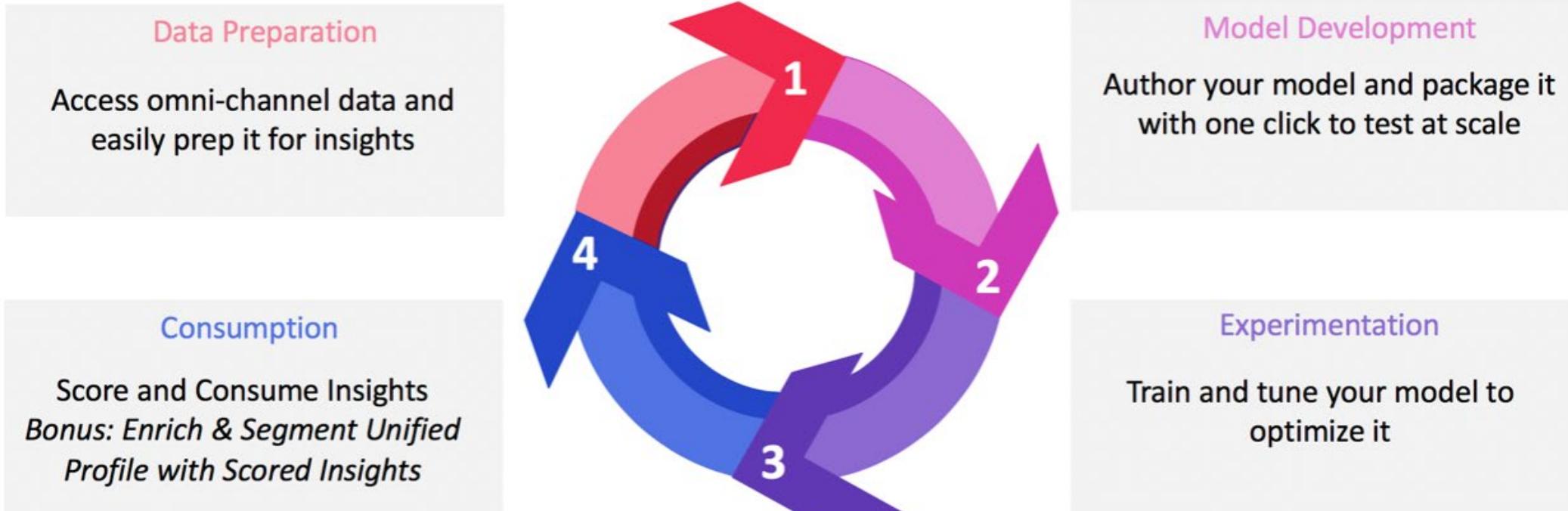
It all starts with a business problem. Lets look at some of the business problems faced by companies today:

- ✓ A retailer might want to predict what products to recommend to a customer
- ✓ A brand might want to predict who would purchase of their products
- ✓ A telecom company might want to predict who would churn and build effective strategies to stop churn
- ✓ A retailer might want to understand their customer better and would like to build customer segments for effective marketing strategies

For today's session, lets consider the business problem of - **"A retailer might want to predict who would stop making a purchase and churn"**

How does it work?

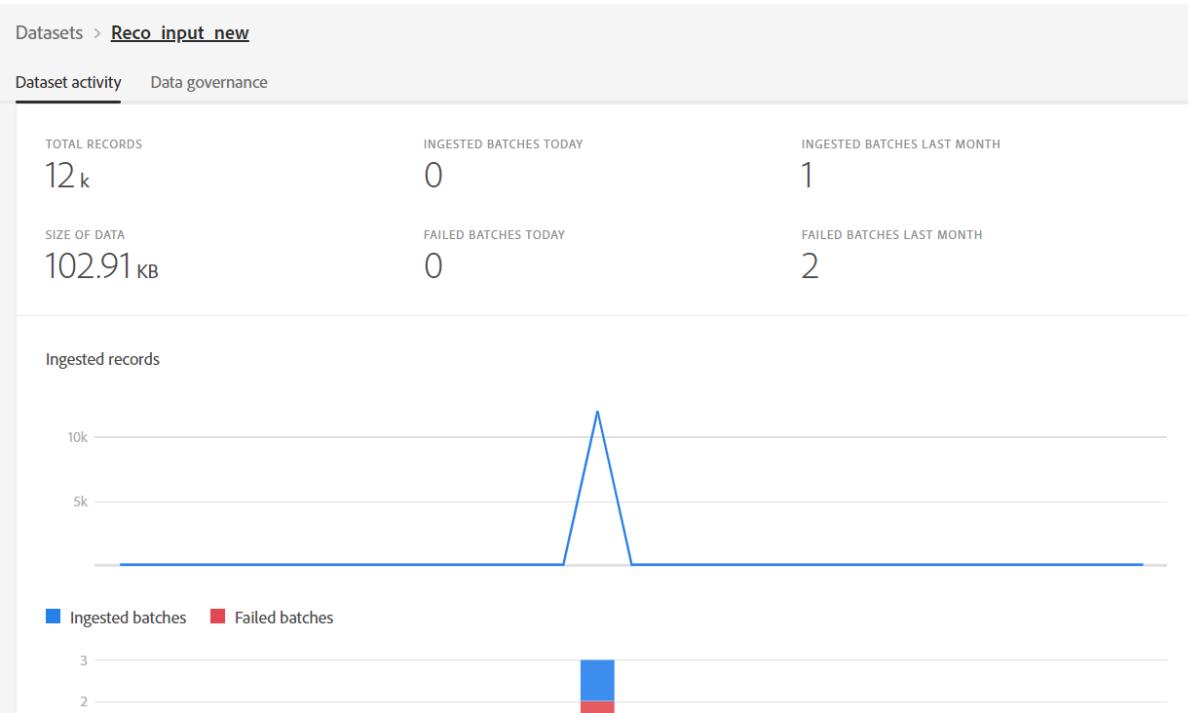
Objective: Consider a fitness retailer, Luma. When customers shop at the website and click on a product, they are presented with personalized product recommendations.



Data Preparation

First step in a model building is data preparation

Lets see how the input data looks like



RECOMMENDATIONS INPUT SCHEMA				
	_aepgdcdevenablement1.itemId	_aepgdcdevenablement1.interactionType	_aepgdcdevenablement1.userId	_id
0	LLWH10	Purchase	matab@adobe.com	matab@adobe.com
1	LLWS01	Purchase	matab@adobe.com	matab@adobe.com
2	LLWP09	Purchase	matab@adobe.com	matab@adobe.com
3	LLMH09	Purchase	punyakot@adobe.com	pbhatt@adobe.com
4	LLMP09	Purchase	punyakot@adobe.com	pbhatt@adobe.com

Data Preparation

First step in a model building is data preparation

Lets look at the Jupyter Notebook and read the data and perform some exploratory data analysis

```
import pandas as pd
import numpy as np

from platform_sdk.dataset_reader import DatasetReader
from datetime import date
dataset_reader = DatasetReader(PLATFORM_SDK_CLIENT_CONTEXT, dataset_id="5d96f6bca74eb91656574254")
# If you do not see any data or would like to expand the default date range, change the following query
data = dataset_reader.read()
data.head()
```

Summary Stats

[3]: data.describe()

	_aepgdcdevenablement1.interactionType	_aepgdcdevenablement1.itemId	_aepgdcdevenablement1.userId	_id	timestamp
count	11989	11989	11989	11989	11989
unique	2	99	865	864	26
top	Purchase	LLMP11	clanchesterel@prweb.com	clanchesterel@prweb.com	1970-01-01T00:00:43.685000
freq	11803	467	47	47	11780

Model Development

After data exploration and preparing the data in the right format, the next step is to build a model

A recommendations model based on Collaborative filtering is created

```
Users = data['_aepgdcdevenablement1.userId'].unique()

Reco = pd.DataFrame()

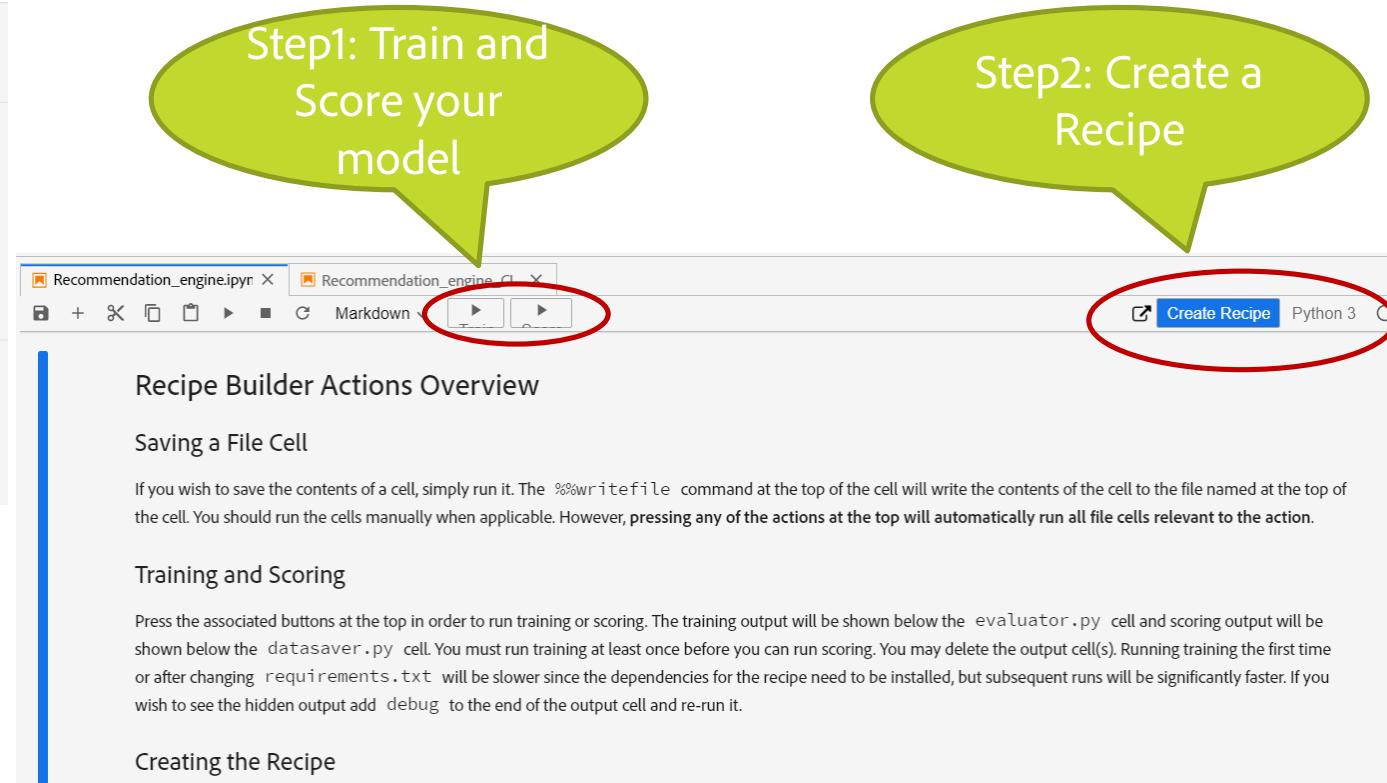
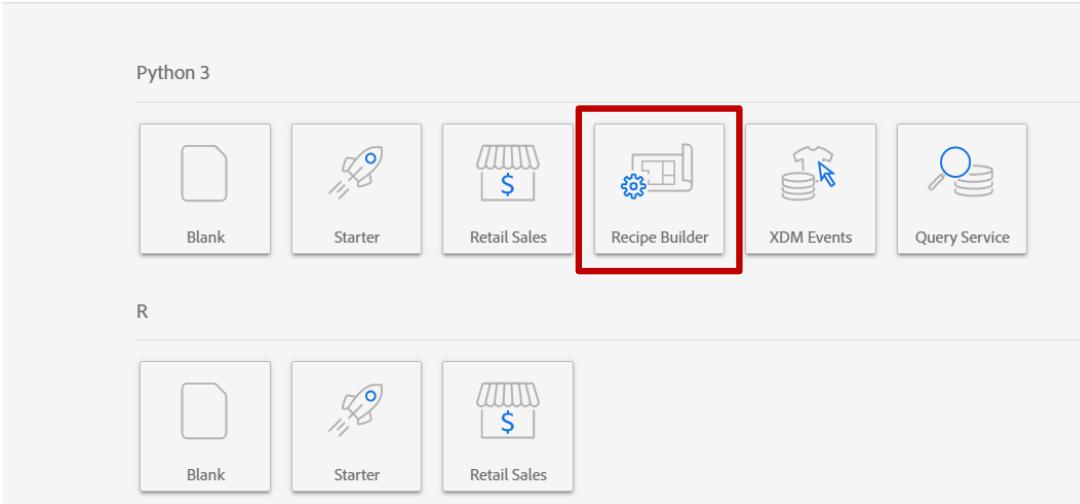
for i in Users:
    Users_a = data[(data['_aepgdcdevenablement1.userId'] == i)]
    Users_a = Users_a.sort_values(['_aepgdcdevenablement1.userId', 'Score'], ascending=[True, True])
    Users_a = Users_a[~Users_a['Item'].isin(data1['_aepgdcdevenablement1.itemId'][data1['_aepgdcdevenablement1.userId']==i])]
    Users_a = Users_a[Users_a.Item != 0]
    Top5 = Users_a.iloc[0:4,:2]
    Reco = Reco.append(Top5)

data = pd.DataFrame(Reco.groupby('_aepgdcdevenablement1.userId')['Item'].apply('#'.join)).reset_index()
data.columns = ['_aepgdcdevenablement1.userId', '_aepgdcdevenablement1.recommendations']
data.head(5)
```

Experimentation

Once you are satisfied with the model you have built, you can create a recipe out of it and perform experiments

Lets look at how a recipe is built



Experimentation

Training and scoring

Create experiments by changing the hyper-tuning parameters

The screenshot shows the Adobe Experimentation interface with the following sections:

- Evaluation metrics:**

NAME	VALUE
Recall	0.7641618497109827
Precision	0.29082369942196534
- Configuration parameters:**

NAME	VALUE
tenant_id	aepgdcdevenablement1
sampling_fraction	0.5
num_recommendations	4
- Training run 2 Properties:**
 - View activity logs**
 - TRAINING**
 - Status: Complete
 - Created: 11/1/2019, 4:19 PM
 - Dataset: Reco_input_new
 - Schema: -

Consumption

Score the data and create Segments with Segment Unified Profile

Score and Preview the scored dataset

RECOMMENDATIONS_OUTPUT_SCHEMA		
	_aepgdcdevenablement1.recommendations	_aepgdcdevenablement1.userId
0	LLMP11#LLWJ09#LLMP12#LLWS09	aandryushind@live.com
1	LLMP11#LLWJ09#LLMP12#LLWS09	aaronowiczot@eventbrite.com
2	LLMP11#LLWJ09#LLMP12#LLWS09	abaccasng@harvard.edu
3	LLMP11#LLWJ09#LLMP12#LLWS09	abehnkefh@goo.gl

Consumption

Score the data and create Segments with Segment Unified Profile

Create segments with the recommendations using Segment UI

The screenshot shows the Adobe Segment UI interface. On the left, there's a query builder titled "Attributes". It contains a single condition: "Include Recommendations contains LLMP". Below this, there's a section for "Include" with a dropdown menu set to "Recommendations" and a "contains" operator, followed by a text input field containing "LLMP" with a green checkmark icon. There's also a "Case sensitive" toggle switch. A "Drag and drop an attribute or segment" placeholder is visible below the query builder. On the right, there's a summary panel titled "Segment properties". It displays "200" estimated qualified profiles, which is 3.25% of the total. There's a "Refresh estimate" button and a note that it was last updated on Nov 17, 2019 at 1:47 PM.

Attributes

Include Recommendations contains LLMP

Include

Recommendations contains LLMP

Case sensitive

Drag and drop an attribute or segment

Segment properties

200 ESTIMATED 3.25% OF TOTAL

Refresh estimate

Last updated: Nov 17, 2019 1:47 PM

Consumption

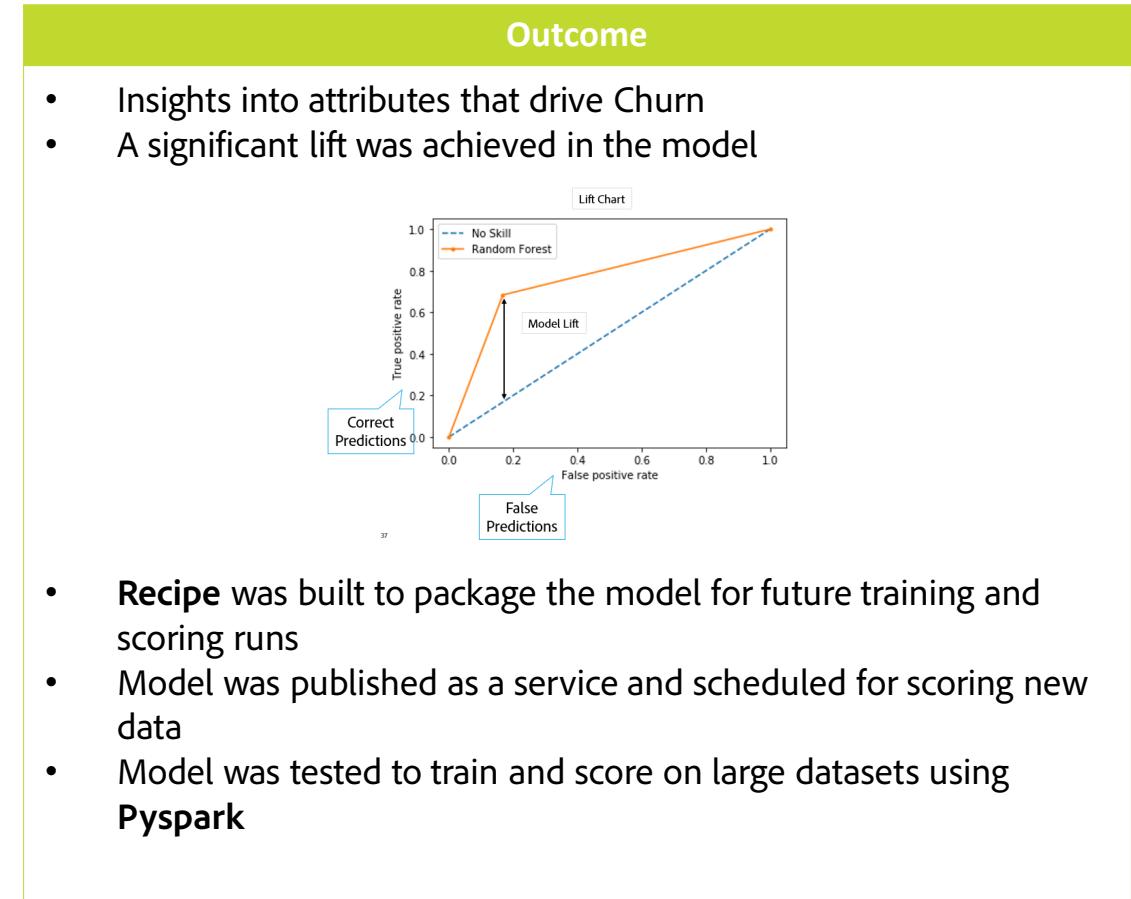
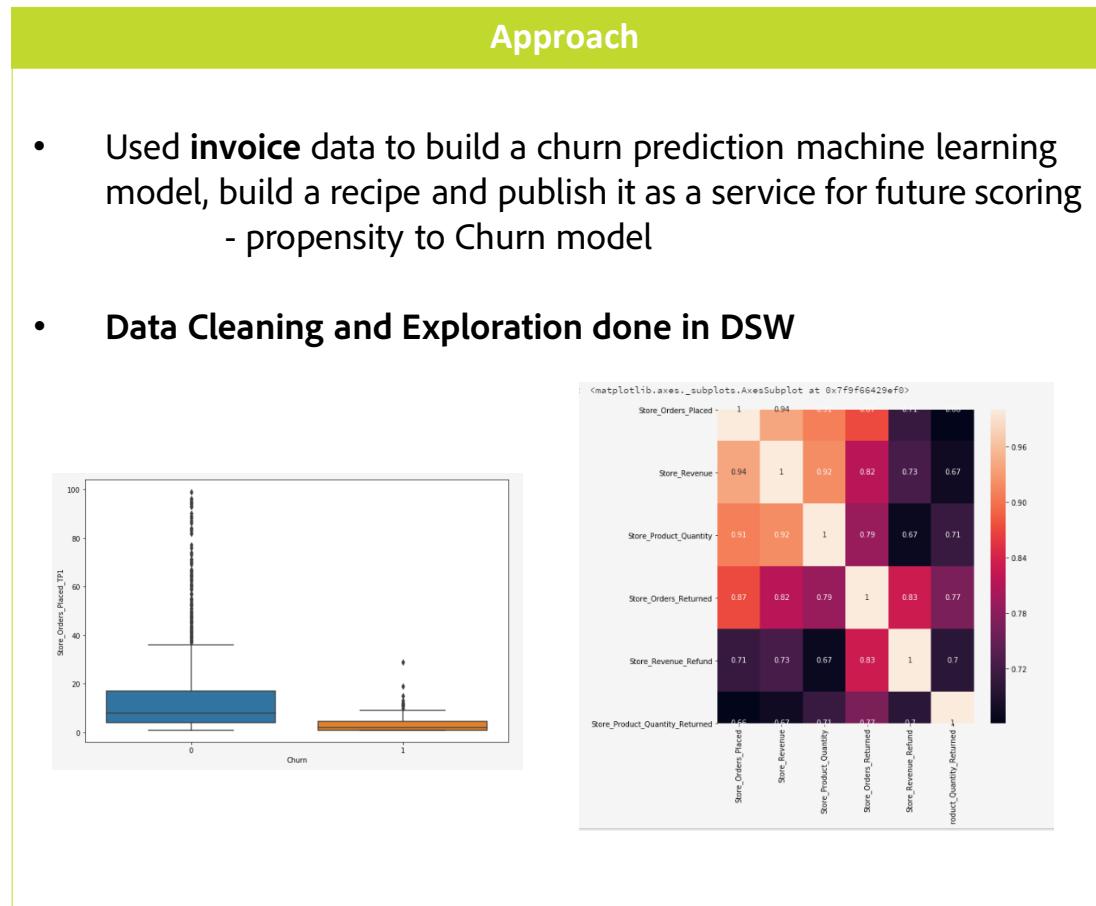
Publish the model as a service

Publish the model as a service and schedule it to run at frequent intervals

The screenshot shows the 'Services > Recommendations Model as Service' overview page. The top navigation bar includes 'Overview', 'Scoring runs', and 'Training runs'. The main content area displays the model details: 'Recommendations Model as Service' (Created 11/17/2019, 12:12 PM by Tina Soni), a 'Score' button, a description field containing a single dash, and a note indicating it was published from model 'test'. Below this, the page is divided into 'Scoring' and 'Training' sections. The 'Scoring' section shows a 'Most Recent' run completed on 11/17/2019 at 12:48 PM using 'Reco_input_new' and resulting in a 'test' dataset. It also shows an 'Upcoming' section with a link to 'Update schedule'. The 'Training' section shows a 'Last trained' run completed on 11/17/2019 at 12:11 PM using 'Reco_input_new', with recall and precision metrics displayed as 0.7641618497109827 and 0.29082369942196534 respectively. It also shows an 'Upcoming' section with a link to 'Update schedule'.

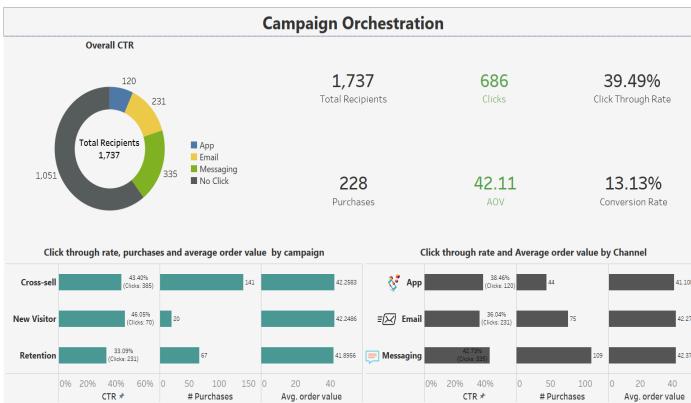
Churn Prediction Model for an American Office Products dealer

Objective: Based on prior customer history and purchase behaviour, build a churn prediction model to predict customers likelihood to churn based on a purchase rhythm



Recommendation engine & propensity to buy model - AEP POC

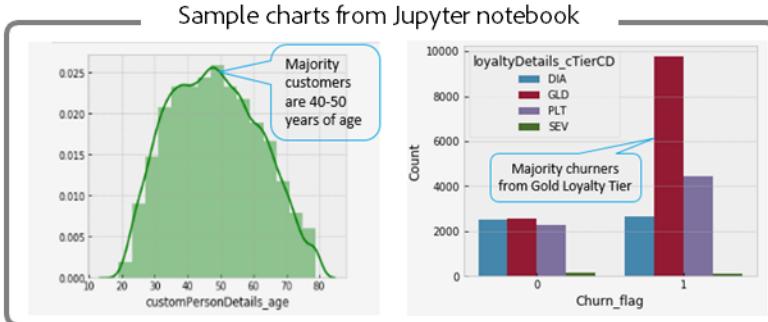
Objective: Based on customers' purchase behavior, Luma, an online retail store wants to send ***triggered campaigns*** to their customers based on their online and offline activity. We leverage DSW for building an engine for product recommendation.

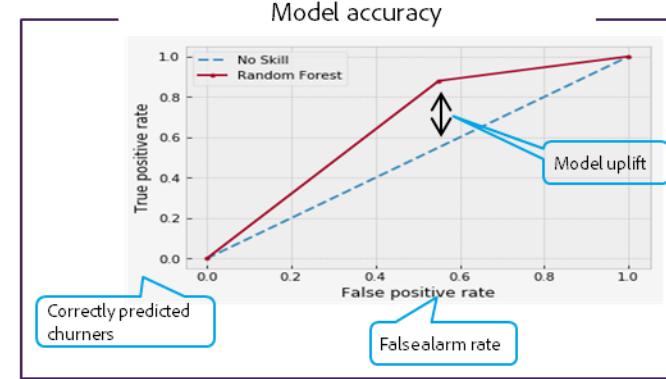
Approach
<ul style="list-style-type: none">Used online and offline data to build two machine learning models using templatized recipe builder available in DSW:<ul style="list-style-type: none">- propensity to buy model- recommendations engine to predict products that can be recommended to a customerConnected AEP and Tableau for Campaign Orchestration Dashboard for customer journeys (recommendations from DSW to conversion and AOV) 

Outcome
<ul style="list-style-type: none">Used the scores and recommendations from this model to create segments that were shared with Adobe CampaignCreated personas for customized targeting such as :<p><i>Janet -- "High propensity to buy, prefers email and recommended products are yoga pants"</i></p>Improved efficiency with integration of feedback into system -- Analyzed the campaign effectiveness, user's purchase behavior, lifetime value and loyalty in near real-time so that marketer can re-strategize, cross-sell and retarget their customers.

Churn prediction for an American gaming hotel and casino corporation

Objective: Predict users' likelihood to churn using AEP, DSW capabilities. Churn scores further used to make segments and target users at high risk.

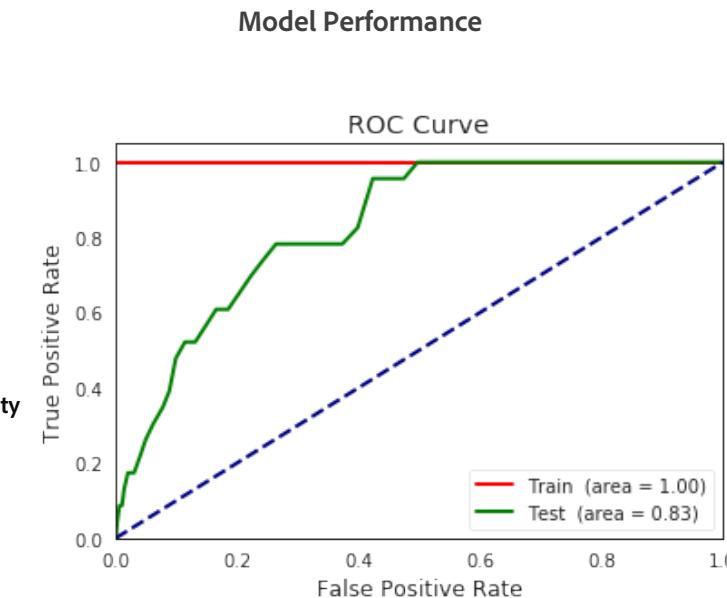
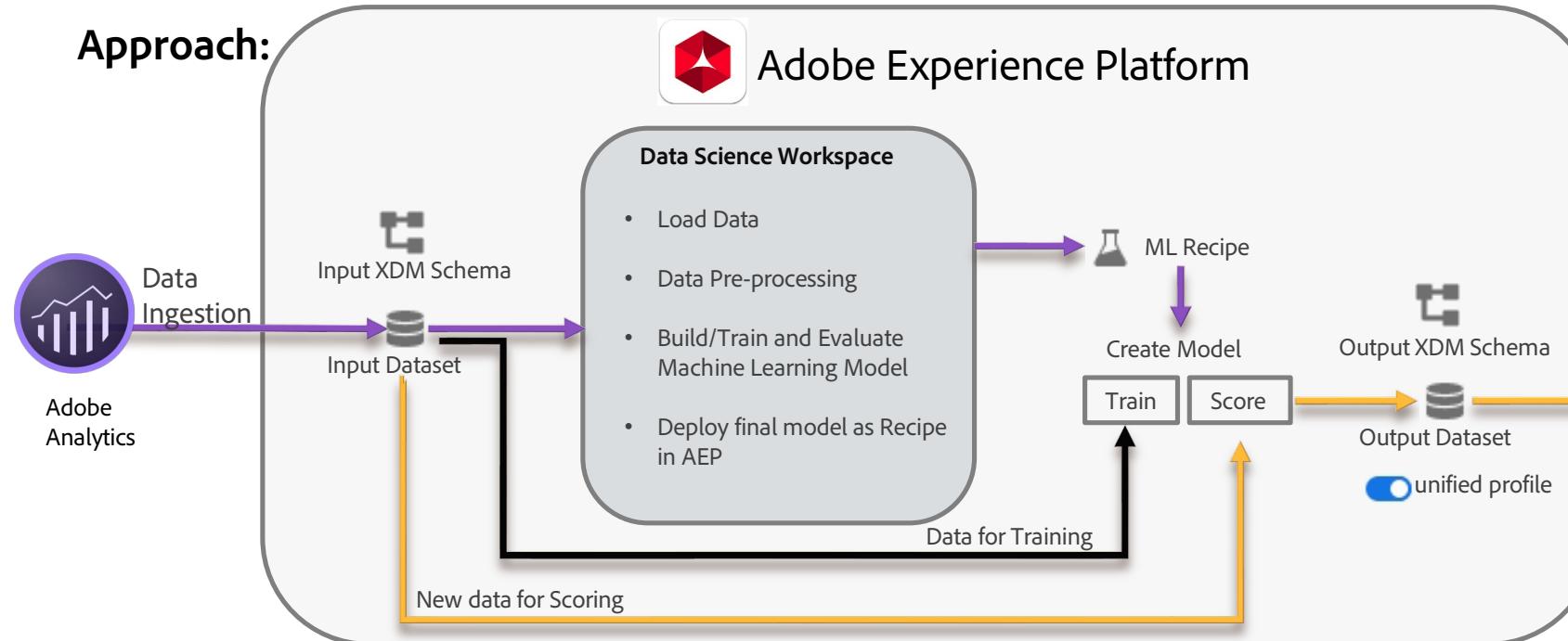
Approach
<ol style="list-style-type: none">1. Leverage the data sets in AEP in the XDM format – Analytics and client-specific datasets ingested in AEP2. Query services used for data aggregation from event level data to profile one-view3. Data cleaning and exploration in DSW <p>Sample charts from Jupyter notebook</p><p>Majority customers are 40-50 years of age</p><p>Majority churners from Gold Loyalty Tier</p><p>loyaltyDetails_cTierCD</p><ul style="list-style-type: none">DIAGLDPLTSEV<p>Count</p><p>Churn_flag</p>4. Insights generation for quick action5. Model building using jupyter notebook within DSW and pre-built recipe notebook6. Algorithm for most robust model – Random Forest

Outcome
<ol style="list-style-type: none">1. Insights on attributes that drive churn behaviour for actioning.2. Model gave a significant lift in accuracy of prediction.  <p>Model accuracy</p> <p>True positive rate</p> <p>No Skill</p> <p>Random Forest</p> <p>Correctly predicted churners</p> <p>Model uplift</p> <p>False alarm rate</p>

Customer POC – Propensity to purchase modeling

Objective: Predict users' propensity to purchase the insurance product using AEP, DSW. Predicted propensity scores will be used further to make segments and target users that can be persuadable.

Approach:



Outcome: Propensity scores of new users are at disposal to the customer.

Output dataset activated to unified profile to make the scores of the users available across all platforms.

Key Takeaways

1

Platform has been architected from the ground up to deliver better personalized experiences at scale.

2

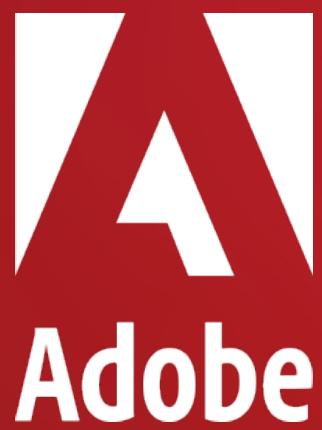
Incremental value with Real-Time Customer Profiles, Data Science Intelligence and External API Activation.

3

Data Science Workspace gives power to companies to deliver real-time, personalized and relevant experiences to their customers

A classroom scene with six children of diverse ethnicities sitting at desks. Five children have their right hands raised, while one boy in the foreground looks down at his paper. They are all smiling. A globe sits on a shelf in the background.

Questions



Overview of Data Science Workspace in Adobe Experience Platform

The screenshot shows the Adobe Experience Platform interface with several components highlighted by purple dashed boxes:

- Data Science Workspace**: A large box on the left side of the page, containing the main navigation menu.
- Data Science Workspace Recipes**: A box at the top of the workspace area.
- Data Science Workspace Notebooks**: A box next to the Recipes box.
- DSW Homepage**: A box highlighting the main content area of the workspace.
- Models**: A box under the Customer section of the sidebar.
- Explore Data Science**: A button under the Models section of the sidebar.
- Get Started**: A button in the DSW Homepage content area.
- More Resources**: A button in the DSW Homepage content area.
- Data to Insights using Data Science Workspace**: A section in the DSW Homepage content area.
- Leverage machine learning and artificial intelligence to develop, train, and tune models to make predictions for personalized, targeted digital experiences.**: Text in the Data to Insights section.
- Data Governance**, **Datasets**, **Identities**, **Unified Profile**, **Destinations**, **Segments**, **Connections**, **File Ingestion**, **Schemas**: Labels for various components shown in a 3D flow diagram.

Overview of Data Science Workspace in Adobe Experience Platform

The screenshot shows the Adobe Experience Platform interface with the Data Science workspace selected. The left sidebar includes sections for Home, Workflows, Customer (Profiles, Segments, Identities), Data Science (Models, Services), and Data Management (Schemas, Datasets, Connections, Queries, Monitoring). The main content area is titled 'Models' and has tabs for Overview, Browse (which is selected), Recipes, and Notebooks. A table lists two models: 'Cartviewreco' and 'Recommendation Engine'. The 'Cartviewreco' model was created on 9/5/2019 at 5:22 PM from 'Product_Recommendation'. The 'Recommendation Engine' model was created on 9/4/2019 at 5:56 PM from 'Product_Recommendation'. A red circle highlights the 'Create Model' button in the top right of the table header. To the right, a sidebar titled 'Models' shows a count of 2.

MODEL NAME	RECIPE SOURCE	LAST UPDATED	CREATED
Cartviewreco	Product_Recommendation	9/5/2019, 5:22 PM	9/5/2019, 5:22 PM
Recommendation Engine	Product_Recommendation	9/4/2019, 5:56 PM	9/4/2019, 5:56 PM

"Browse" shows all the models that are created
There is also an option to create a new model¹

Overview of Data Science Workspace in Adobe Experience Platform

The screenshot shows the Adobe Experience Platform Data Science Workspace interface. The left sidebar includes sections for Home, Workflows, CUSTOMER (Profiles, Segments, Identities), DATA SCIENCE (Models, Services), and DATA MANAGEMENT (Schemas, Datasets, Connections, Queries, Monitoring). The Models section is currently selected. The main content area displays a table of Recipes, with the 'Recipes' tab selected in the navigation bar. The table columns are Recipe Name, Created (sorted by date), and Type. The table contains five entries:

RECIPE NAME	CREATED	TYPE
Cart_reco	9/1/2019, 2:12 PM	Python
Cart_reco_recipe	9/1/2019, 1:55 PM	Python
Product_Recommendation	8/29/2019, 3:02 PM	Python
Prop_to_buy_XGBoost	8/28/2019, 4:51 PM	Python
Prop_to_buy_XGBoost	8/28/2019, 4:50 PM	Python

A red circle highlights the 'Import Recipe' button in the top right corner of the table header. To the right of the table, a large callout box displays the number '5' and the text 'Recipes'. The top right of the page shows the user 'AEP GDC Dev Enablement1' and standard navigation icons.

A “Recipe” is a reusable component of a model which can be deployed for different datasets. Any existing Recipe¹ can also be imported using the “Import Recipe” Option

Overview of Data Science Workspace in Adobe Experience Platform

The screenshot shows the Adobe Experience Platform Data Science Workspace interface. On the left is a sidebar with various icons for Models, Overview, Browse, Recipes, and Notebooks. The Notebooks tab is selected, showing a file browser with a list of files. Above the file browser is a menu bar with File, Edit, View, Run, Kernel, Tabs, Settings, and Help. To the right of the file browser is a 'Launcher' window. The launcher is divided into sections for Python 3, R, PySpark 3, and Spark. Each section contains icons for 'Blank', 'Starter', and other notebook types like 'Retail Sales', 'Recipe Builder', 'XDM Events', 'Query Service', 'XDM Queries', and 'Aggregation'. A purple dashed box highlights the 'Launcher' title and the icons for the different notebook types. Another purple dashed box highlights the file browser area with the text: 'Create a new file, open an existing file, save files or execute a command'. At the bottom of the launcher window, there are status indicators for GPU OFF, CPU 0%, RAM 0.12GB / 4.00GB, Disk Space 41.71GB / 96.88GB, and a help icon. The word 'Launcher' is also at the bottom right of the window.

Create a new file, open an existing file, save files or execute a command

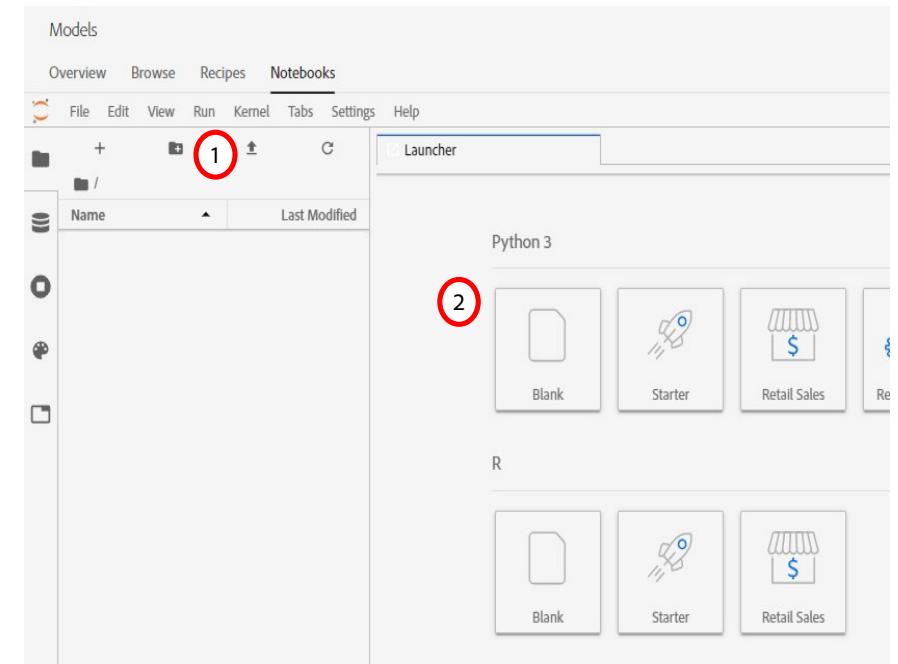
Launcher – Select the program of your choice in Launcher

In the Launcher window, we can see four different types of notebook that can be opened.

Overview of Data Science Workspace in Adobe Experience Platform

Adding Notebooks

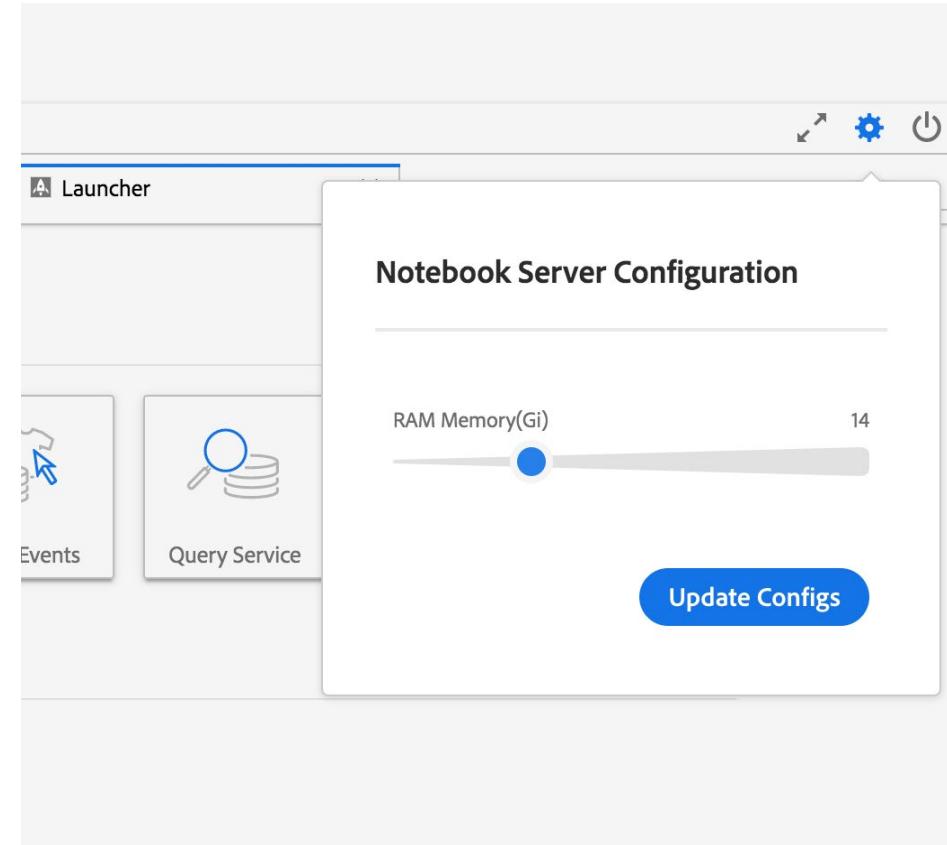
- To add your notebooks to Data Science Workspace, you can manually drag-drop them at the windows pane to your left.
- Else, you could click on the upload¹ symbol and locate your files through explorer.
- You can also start a blank notebook² and start exploring datasets and trying basic commands.



Overview of Data Science Workspace in Adobe Experience Platform

Increasing RAM

While performing powerful computations or executing a model that requires more computing resources, there is an option in AEP to increase RAM memory

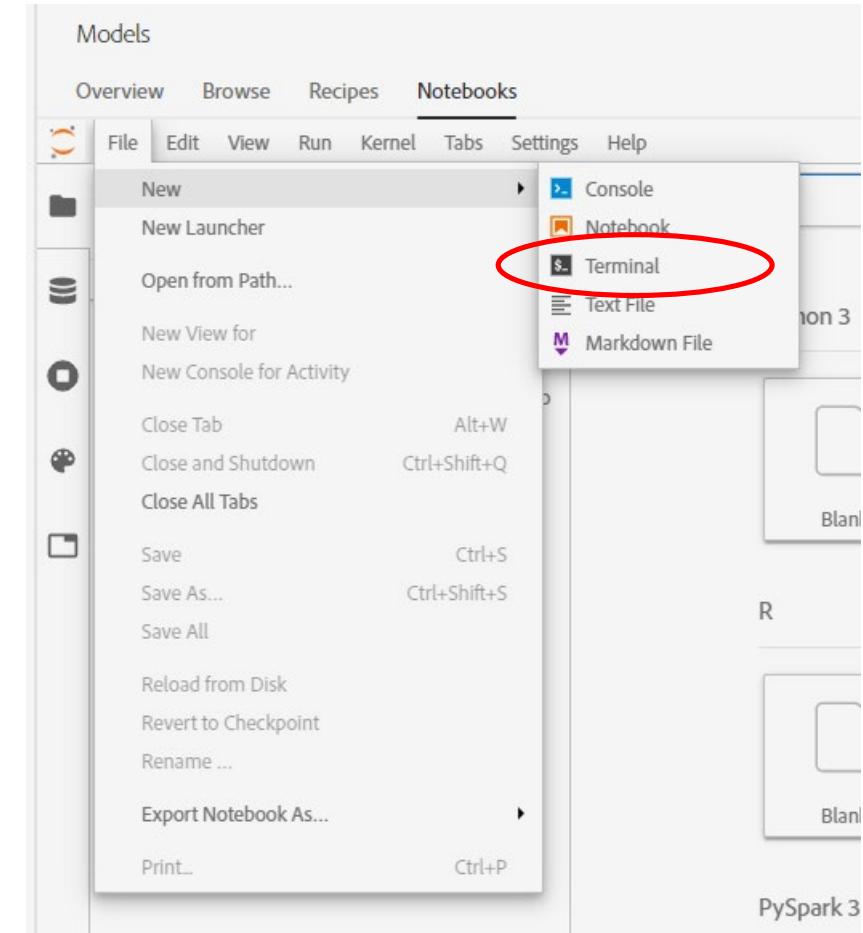


Overview of Data Science Workspace in Adobe Experience Platform

Installing New Libraries

If you want to install new libraries that are not already present, it can be done through the terminal window

File -> New -> Terminal



Overview of Data Science Workspace in Adobe Experience Platform

Executing a Command

Select the cell you want to execute

Press "Shift+Enter"

OR

Click on the "Play" button on top left hand side

```
[1]: import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from data_access_sdk_python.writer import DataSetWriter
writer = DataSetWriter()
```



Overview of Adobe Experience Platform - Datasets

The screenshot shows the Adobe Experience Platform Datasets interface. On the left, a sidebar navigation includes Home, Workflows, CUSTOMER (Profiles, Segments, Identities), DATA SCIENCE (Models, Services), and DATA MANAGEMENT (Schemas, **Datasets**, Connections, Queries, Monitoring). A yellow star icon is next to the Datasets link. The main content area has tabs for Overview and Browse, with 'Browse' selected. It displays two datasets:

NAME	CREATED	SOURCE	SCHEMA	LAST B
Cross-Industry Demo Retail Data postValues	9/20/2019, 1:49 AM	Adobe Analytics	Cross-Industry Demo Retail Data Schema v1	Success
Cross-Industry Demo Retail Data midValues	9/20/2019, 1:49 AM	Adobe Analytics	Cross-Industry Demo Retail Data Schema v1	Success

On the right, there are summary statistics:

- 47 Datasets (circled 7)
- 2.67 GB Total Storage Used (circled 8)
- Most Recently Updated (circled 9):
 - acsgdcteam1 midValues (8/14/2019)
 - acsgdcteam1 postValues (8/14/2019)
 - Cross-Industry Demo ... (9/22/2019)

Datasets lists down all the data that is ingested in AEP¹, the date of creation², Source³, Schema⁴ and Last Modified⁵. There is an option to create a dataset⁶ and the window also provides a view of the number of datasets⁷, storage⁸ and most recently updated⁹.

Overview of Adobe Experience Platform – Dataset characteristics

The screenshot shows the Adobe Experience Platform Datasets interface. On the left, a sidebar navigation includes Home, Workflows, CUSTOMER (Profiles, Segments, Identities), DATA SCIENCE (Models, Services), DATA MANAGEMENT (Schemas, Datasets, Connections, Queries, Monitoring). The Datasets option is selected. The main area displays 'Dataset Activity' for 'Cross-Industry Demo Retail Data postValues'. Key metrics shown are: TOTAL RECORDS (1.03 M), INGESTED BATCHES TODAY (18), FAILED BATCHES TODAY (0), INGESTED BATCHES LAST 7 DAYS (260), FAILED BATCHES LAST 7 DAYS (0). Below these are two charts: 'Ingested Records' (line chart from September 19 to 25) and 'Ingested Batches' (bar chart from September 19 to 25). The 'Preview Dataset' button in the top right is circled in green. To the right, a detailed view of the dataset 'Cross-Industry Demo Retail Data postValues' is shown, including its ID, name, description, table name, schema, and connection information.

AEP GDC Dev Enablement 1

Delete Dataset

Preview Dataset

Cross-Industry Demo Retail Data postValues

Info Add Data

Dataset ID
5d83e2c70303011644074859

Name
Cross-Industry Demo Retail Data post...

Description
Analytics Data Connector for report suite Cross-Industry Demo Retail Data

Table Name
cross_industry_demo_retail_data_postvalues

Unified Profile ⓘ

Schema
Cross-Industry Demo Retail Data Schema v1

Streaming ⓘ

Not Enabled

Source
Adobe Analytics

Connection
AnalyticsConnection

Click on the name of the dataset to open a window with dataset characteristics including number of records, batches ingested, failed batches, Dataset ID, Schema, etc.

Click Preview Dataset to view the actual data

Overview of Adobe Experience Platform – Dataset Preview

▼ Cross-Industry Demo Data Schema v1

- advertising
- search
- device
- _experience
- environment
- marketing
- identityMap
- placeContext
- web
- dataSource
- endUserIDs
- commerce

CROSS-INDUSTRY DEMO DATA SCHEMA V1

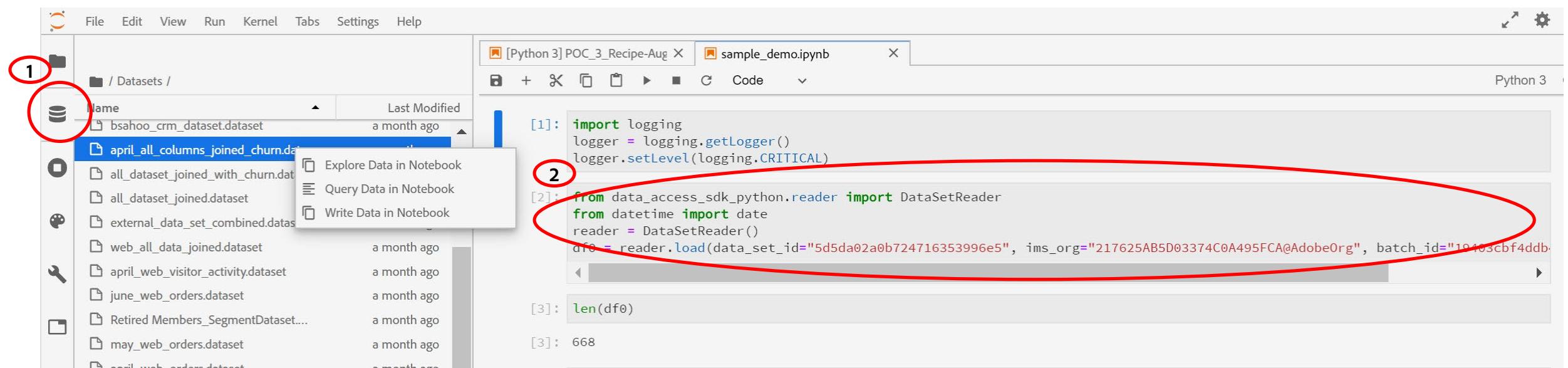
	productListItems.list.element._experience.analytics.customDimensions.eVars.eVar23	_experience.analytics.session.web.webReferrer.type	_experience.analytics.endUser.firstW
0	;prd1185;1;247.8;event9=116.47 event10=	unknown	http://www.adobedemo.com/cont
1	;prd1185;1;247.8;event9=116.47 event10=	unknown	http://www.adobedemo.com/cont
2	;prd1185;1;247.8;event9=116.47 event10=	unknown	http://www.adobedemo.com/cont
3	;prd1185;1;247.8;event9=116.47 event10=	unknown	http://www.adobedemo.com/cont
4	;prd1185;1;247.8;event9=116.47 event10=	unknown	http://www.adobedemo.com/cont
5	-	unknown	http://www.adobedemo.com/cont
6	-	unknown	http://www.adobedemo.com/cont
7	-	unknown	http://www.adobedemo.com/cont
8	;prd1185;1;247.8;event9=116.47 event10=	unknown	http://www.adobedemo.com/cont

Data Exploration

Explore Dataset in AEP Data Science Workspace Jupyter Notebook

There are 2 ways to read a dataset in AEP Jupyter Notebook

1. Select “Datasets” in Notebooks and select “Explore Data in Notebook”
2. Write the code, change the dataset ID and Batch ID. Execute the command

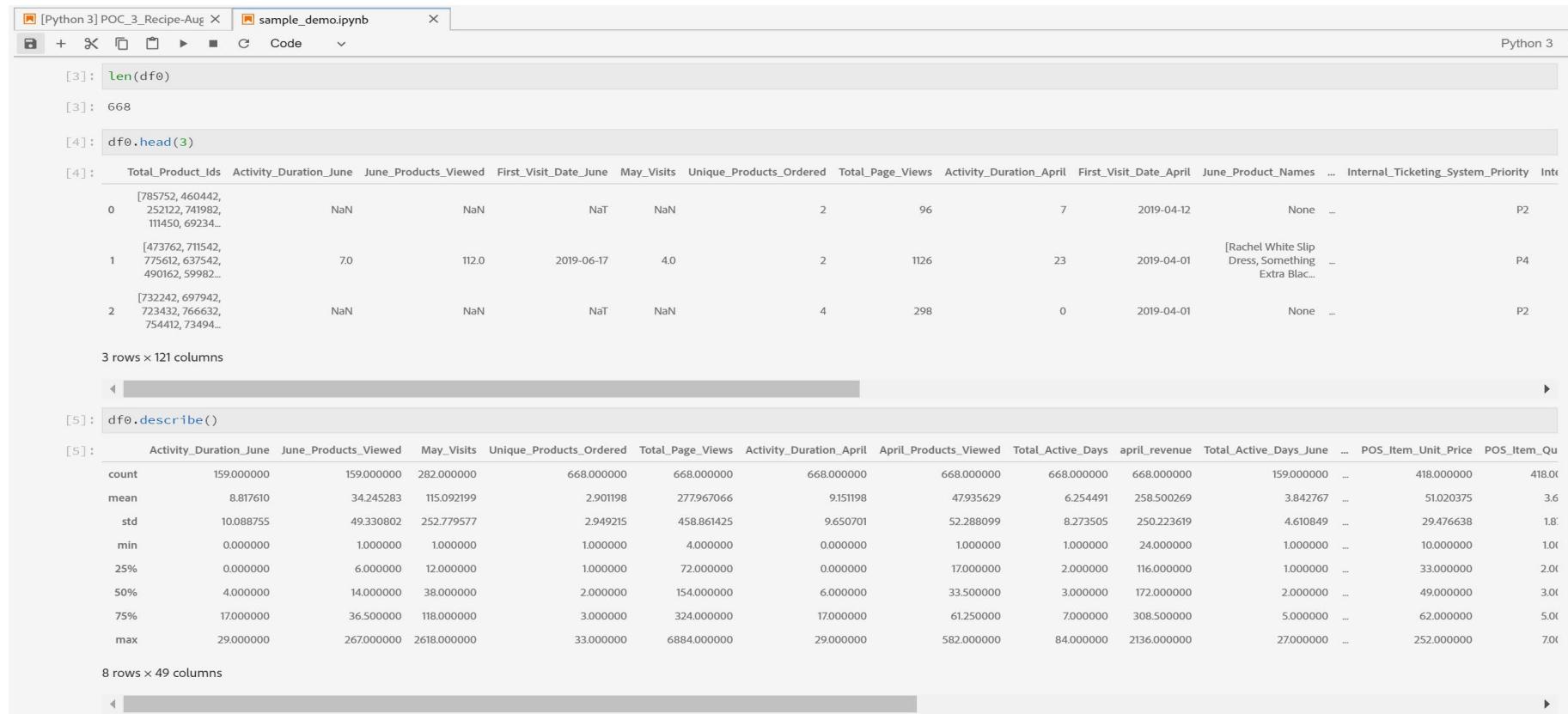


Data Science Workspace

Quick Summary statistics to understand the distribution of different attributes in data.

Use <dataset>.head(10) to view 10 records of the data

<dataset>.describe() gives a summary statistics of the data



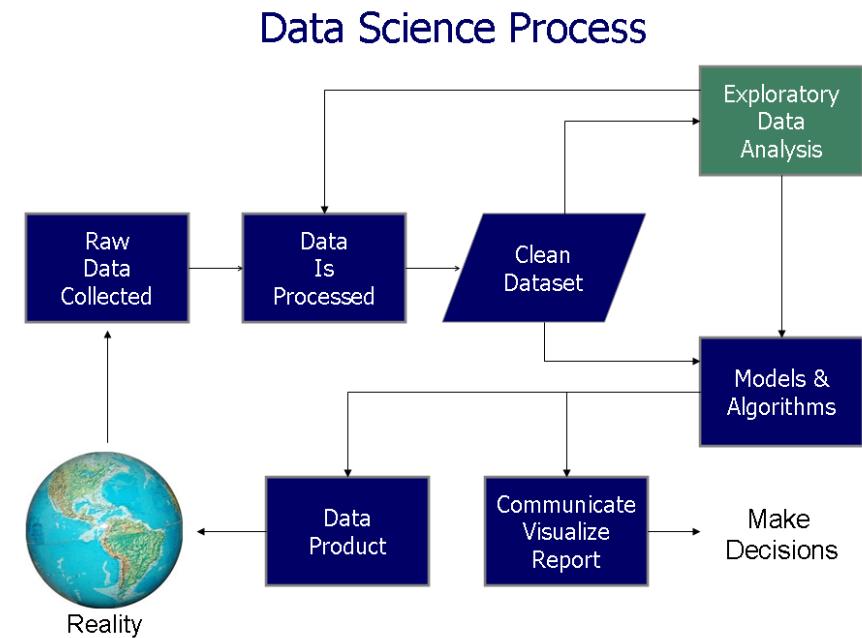
The screenshot shows a Jupyter Notebook interface with two code cells. Cell [4] displays the first three rows of a DataFrame `df0` using `df0.head(3)`. Cell [5] displays the descriptive statistics for all columns using `df0.describe()`.

```
[3]: len(df0)
[3]: 668
[4]: df0.head(3)
[4]:   Total_Product_Ids  Activity_Duration_June  June_Products_Viewed  First_Visit_Date_June  May_Visits  Unique_Products_Ordered  Total_Page_VIEWS  Activity_Duration_April  First_Visit_Date_April  June_Product_Names ... Internal_Ticketing_System_Priority  Int...
[4]:   785752, 460442, 252122, 741982, 111450, 69234...
[4]:   NaN          NaN          NaN          NaT        NaN            2           96             7  2019-04-12        None  ...
[4]:   P2
[4]:   473762, 711542, 775612, 637542, 490162, 59982...
[4]:   7.0         112.0      2019-06-17       4.0            2          1126            23  2019-04-01  [Rachel White Slip Dress, Something Extra Blac...
[4]:   P4
[4]:   732242, 697942, 723432, 766632, 754412, 73494...
[4]:   NaN          NaN          NaN          NaT        NaN            4           298            0  2019-04-01        None  ...
[4]:   P2
[4]: 3 rows x 121 columns
[5]: df0.describe()
[5]:   Activity_Duration_June  June_Products_Viewed  May_Visits  Unique_Products_Ordered  Total_Page_VIEWS  Activity_Duration_April  April_Products_Viewed  Total_Active_Days  april_revenue  Total_Active_Days_June  ...  POS_Item_Unit_Price  POS_Item_Qu...
[5]:   count    159.000000    159.000000  282.000000    668.000000    668.000000    668.000000    668.000000    668.000000    159.000000  ...  418.000000    418.00...
[5]:   mean     8.817610    34.245283  115.092199    2.901198  277.967066    9.151198    47.935629    6.254491  258.500269    3.842767  ...  51.020375    3.6
[5]:   std     10.088755    49.330802  252.779577    2.949215  458.861425    9.650701    52.288099    8.273505  250.223619    4.610849  ...  29.476638    1.8
[5]:   min     0.000000    1.000000  1.000000    1.000000    4.000000    0.000000    1.000000    1.000000    24.000000    1.000000  ...  10.000000    1.0
[5]:   25%     0.000000    6.000000  12.000000    1.000000    72.000000    0.000000    17.000000    2.000000   116.000000    1.000000  ...  33.000000    2.0
[5]:   50%     4.000000   14.000000  38.000000    2.000000   154.000000    6.000000   33.500000    3.000000   172.000000    2.000000  ...  49.000000    3.0
[5]:   75%    17.000000   36.500000  118.000000    3.000000   324.000000   17.000000   61.250000    7.000000   308.500000    5.000000  ...  62.000000    5.0
[5]:   max    29.000000   267.000000  2618.000000   33.000000  6884.000000   29.000000   582.000000   84.000000  2136.000000   27.000000  ...  252.000000    7.0
[5]: 8 rows x 49 columns
```

Exploratory Data Analysis

What is it and why is it important

- ✓ Exploratory data analysis is an approach to analyze and understand the data, discover patterns, anomalies and existing relationships with features present in the data
- ✓ It is often performed through visual methods incorporating techniques like density plots, scatter plots, box plots and correlation analysis
- ✓ This is an important step in building an advanced ML model because every ML algorithm follows a GIGO principle, i.e., Garbage In Garbage Out.



Exploratory Data Analysis

Different types of Visualizations

✓ Few types of data visualizations are:

- ✓ Box Plots
- ✓ Scatter Plots
- ✓ Histograms
- ✓ Correlation Matrix

Data Science Workspace

Data Exploration – Box Plots

A Box plot is a visualization which can help data scientists detect outliers. Outliers tend to skew the model results and it is important to treat the outliers appropriately

In Python, the code used to generate Box plots is as below

```
plt.figure(figsize=(12,6))
sns.boxplot(x='Person_Gender', y='Total_Orders_Placed' , data=df0)
```

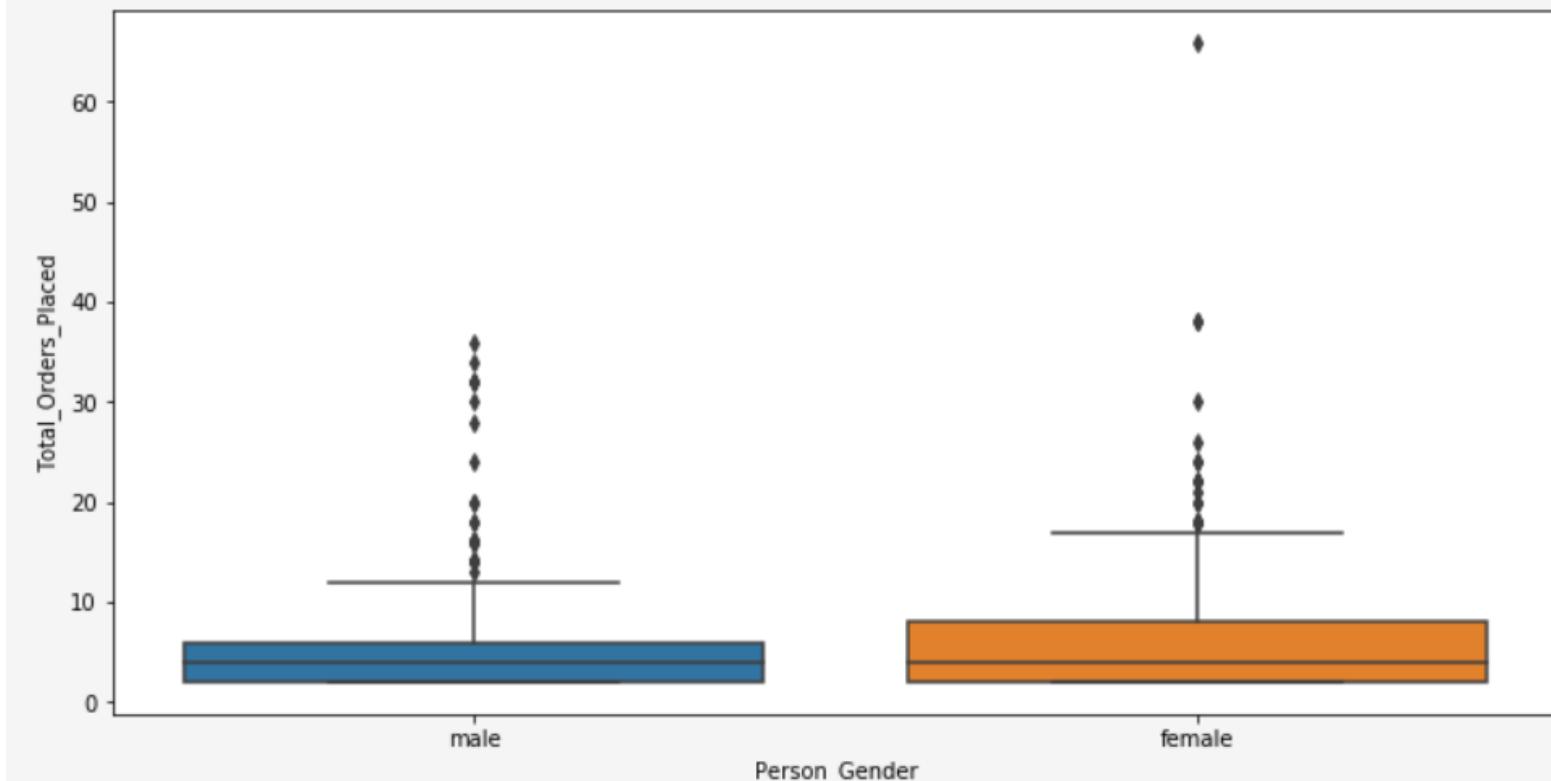
Data Science Workspace

Data Exploration – Box Plots

The below plots show presence of outliers

```
plt.figure(figsize=(12,6))
sns.boxplot(x='Person_Gender', y='Total_Orders_Placed' , data=df0)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe39247d470>
```



Data Science Workspace

Data Exploration – Scatter Plots

Scatter Plots are 2-dimensional plots that can help a data scientist evaluate relationships between 2 dimensions

In Python, the code used to generate scatter plots is as below

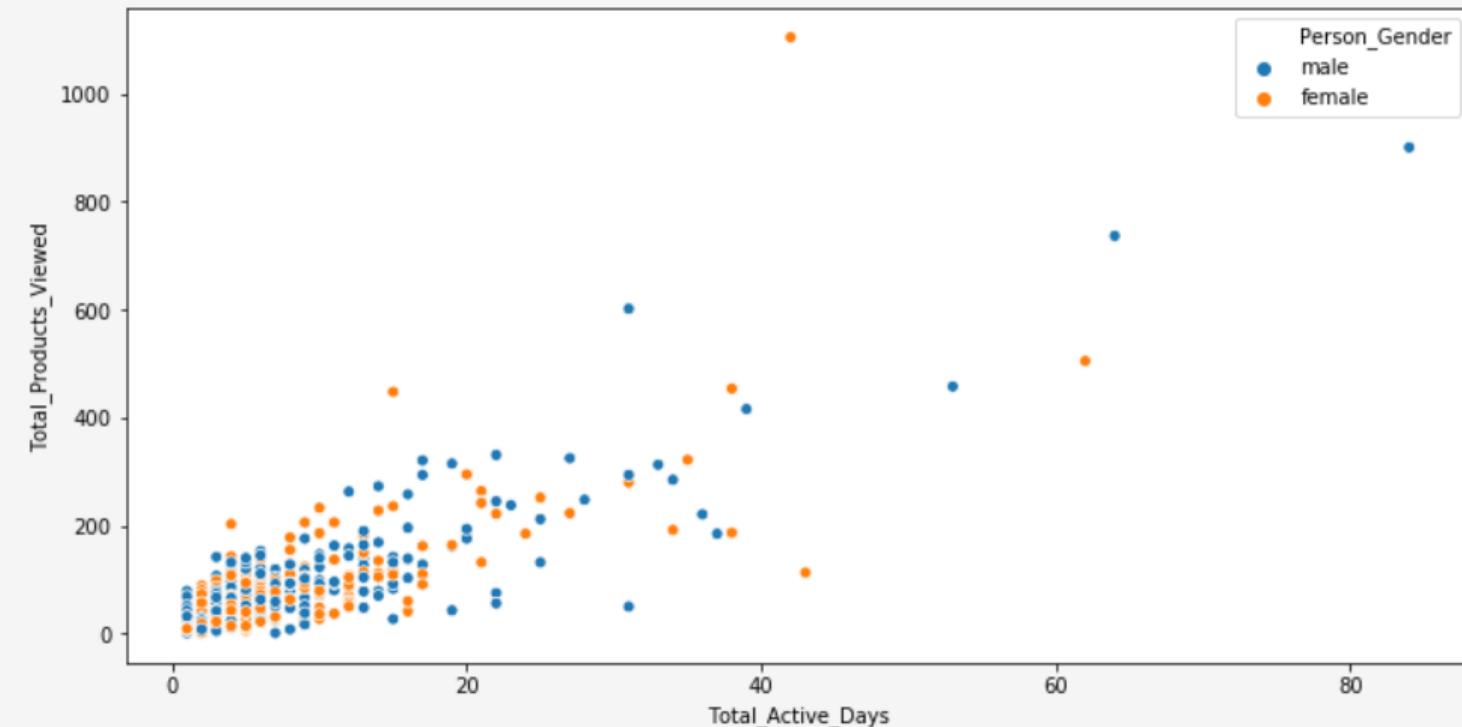
```
plt.figure(figsize=(12,6))
ax = sns.scatterplot(x="Total_Active_Days", y="Total_Products_Viewed", hue="Person_Gender", data=df0)
```

Data Science Workspace

Data Exploration – Scatter Plots

The below plots show presence of outliers

```
: plt.figure(figsize=(12,6))
ax = sns.scatterplot(x="Total_Active_Days", y="Total_Products_Viewed", hue="Person_Gender", data=df0)
```



Data Science Workspace

Data Exploration – Histograms

A histogram is a plot that is used to discover the underlying frequency distribution (shape) of a set of continuous numerical data like number of orders, age, etc.

In Python, the code used to generate Box plots and scatter plots is as below

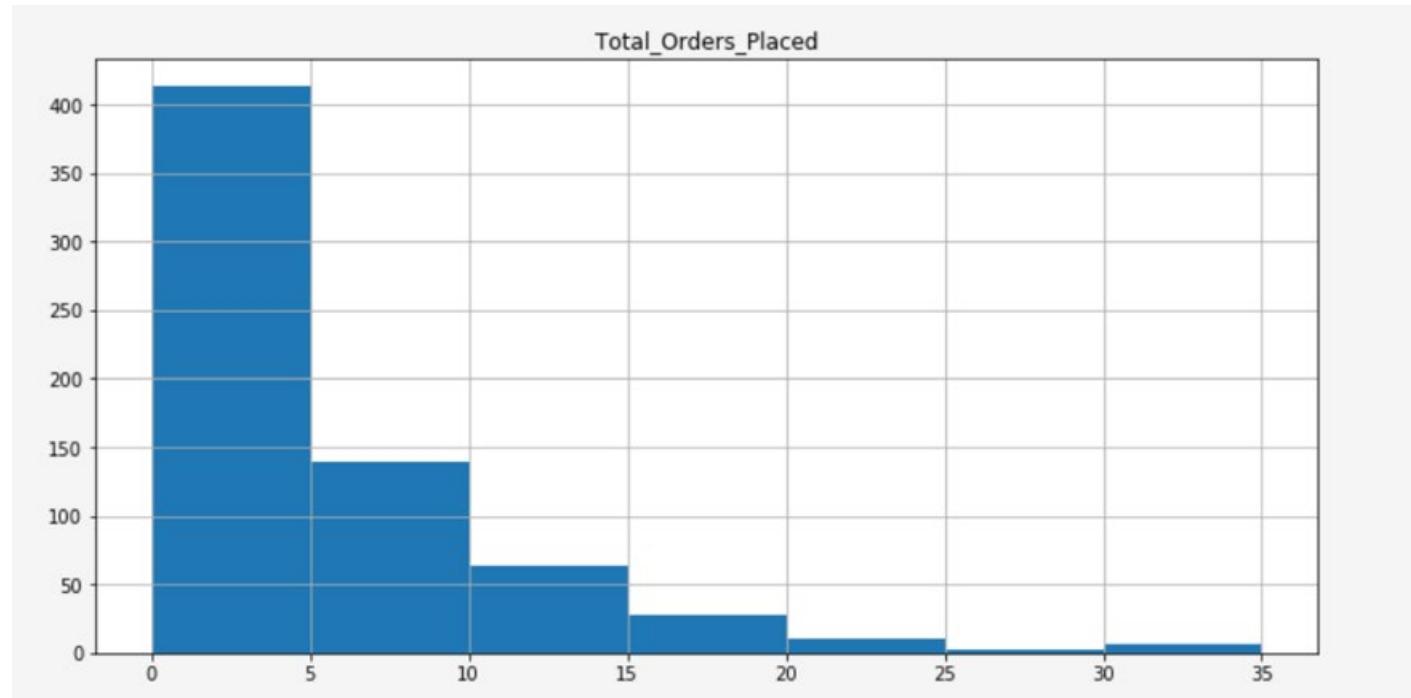
```
import matplotlib.pyplot as plt
%matplotlib inline
plt.figure(figsize=(12,6))
plt.title('Total_Orders_Placed')
df0['Total_Orders_Placed'].hist(bins=[0,5,10,15,20,25,30,35])
```

Data Science Workspace

Data Exploration – Histograms

The output of the code is below

Interpretation: This histogram is done on Total orders placed and the plot shows that maximum customer place <5 orders



Data Science Workspace

Data Exploration – Correlation matrix

A Correlation Matrix is a table that gives correlation coefficients between variables. This matrix is used to summarize data and provide insights into relationships between different variables. It is also used as inputs to advanced analytics

In Python, the code used to generate a correlation matrix is as below

```
import seaborn as sns  
plt.figure(figsize=(5,5))  
sns.heatmap(df1.corr(), annot = True)
```

Data Science Workspace

Data Exploration – Correlation matrix

Interpretation: Correlation coefficients range from -1

to +1, signs indicating the direction of the relation

Below are thumb rules to interpret a correlation

matrix

Absolute Correlation coefficients that are

- 0.8 – 1 – Very Strong correlation
- 0.6 – 0.8 – Strong correlation
- 0.4 – 0.6 – moderate correlation
- <0.4 – No correlation



Interacting with 3rd party tools

Tableau and PowerBI

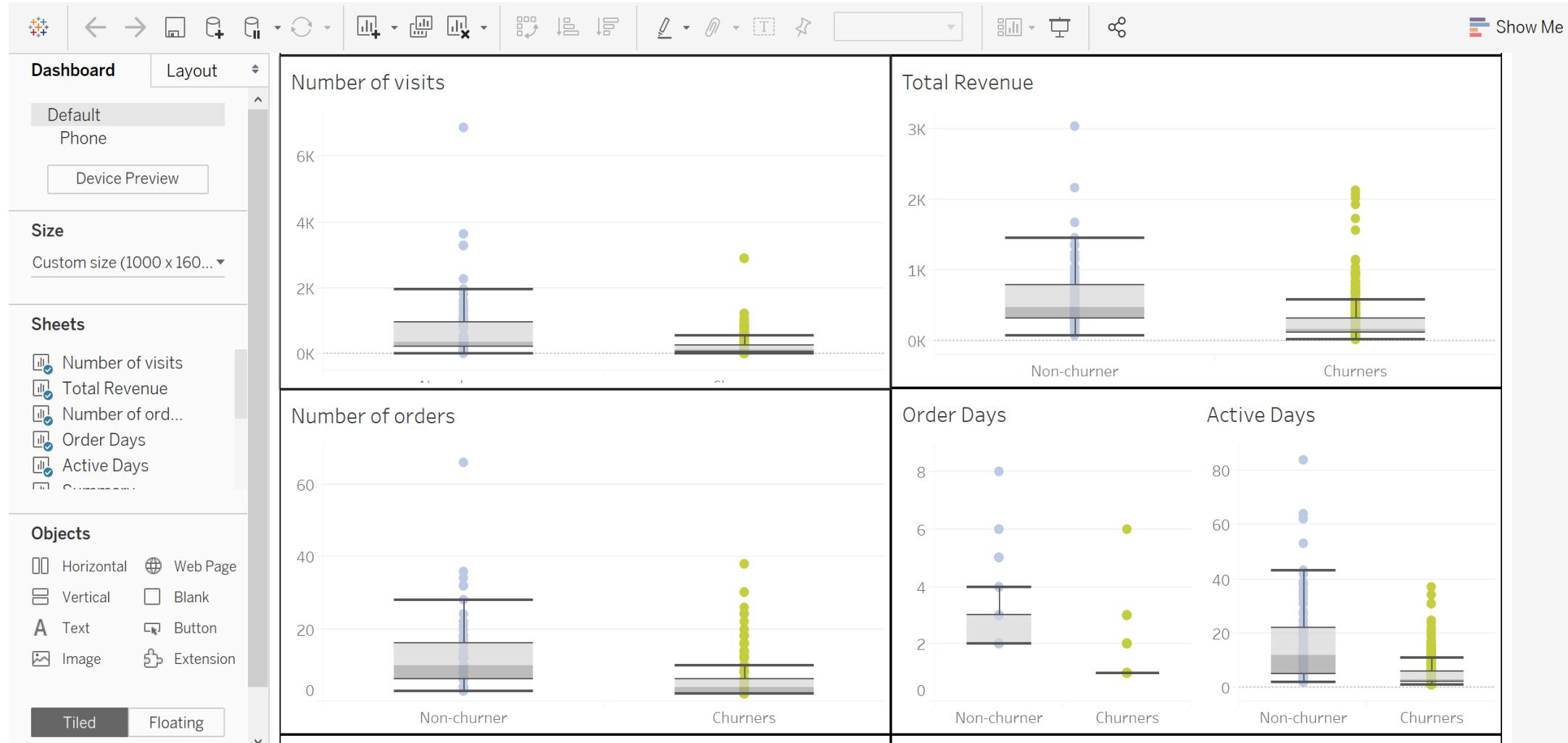
The screenshot shows the Adobe Experience Platform Query interface. On the left, there's a sidebar with various navigation options like Home, Workflows, Unified Profile, Profiles, Segments, Identities, Data Science, ML Models, Data Management, Schemas, Datasets, Connections, Queries (which is selected), and Monitoring. The main area has tabs for Query, Browse, and Configuration, with Configuration selected. Under Configuration, there's a section for "Postgres Credentials" with fields for Host (verizon.platform-query.adobe.io), Port (80), Database (all), Username (777B575E55828EBB7F000101@AdobeOrg), and Password (a long, complex string). Below these fields is a note: "Expires: 4/24/2019 at 9:54 AM Refresh". To the right, there's a "PSQL Command" section containing a long command string starting with "psql *sslmode=require host=verizon.platform-query.adobe.io port=80 dbname=all user=777B575E55828EBB7F000101@AdobeOrg password=eNqrVqowLVWYUsrMLY7PsZTUzUgtIDXUSo4tUURSsxJB8oEBRuZminVAgALngwLeNptUltuozAQ_Rceq9DYxocC...". A note above the command says "Make sure to enable SSL in the client".

BI Tools

- Tableau, PowerBI
- Provides credentials and instructions for connecting supported clients
- Connectivity achieved using existing PostgreSQL drivers

Interacting with 3rd party tools

Build Dashboards in third party data visualization tools such as tableau

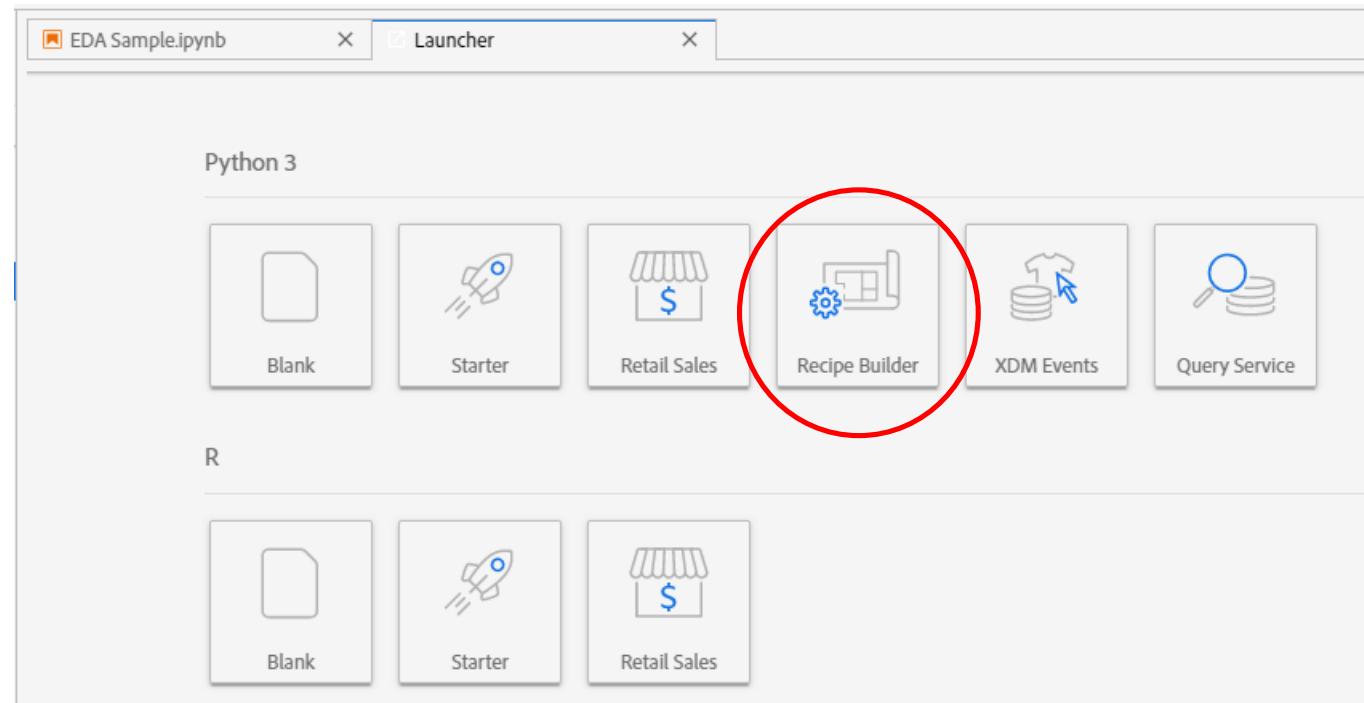


AEP data accessed and read in Tableau

Recipe Building

Build a recipe out of the trained model to be used as a reusable component to score multiple populations

A recipe is a reusable component that is built out of a pre-trained and validated model which can be used to score multiple other populations



Select Recipe Builder
from Launcher

Recipe Building

Build a recipe out of the trained model to be used as a reusable component to score multiple populations

ED A Sample.ipynb [Python 3] Recipe Builder-Copy.ipynb

Train Score Create Recipe

Recipe Builder Actions Overview

Saving a File Cell

If you wish to save the contents of a cell, simply run it. The `%%writefile` command at the top of the cell will write the contents of the cell to the file named at the top of the cell. You should run the cells manually when applicable. However, **pressing any of the actions at the top will automatically run all file cells relevant to the action.**

Training and Scoring

Press the associated buttons at the top in order to run training or scoring. The training output will be shown below the `evaluator.py` cell and scoring output will be shown below the `datasaver.py` cell. You must run training at least once before you can run scoring. You may delete the output cell(s). Running training the first time or after changing `requirements.txt` will be slower since the dependencies for the recipe need to be installed, but subsequent runs will be significantly faster. If you wish to see the hidden output add `debug` to the end of the output cell and re-run it.

Step 1 – Train your
model

Step 2 – Score the
model

Step 3 – Select
“Create Recipe” to
create a new recipe

Recipe Building

Build a recipe out of the trained model to be used as a reusable component to score multiple populations

Step 1 – Train the model

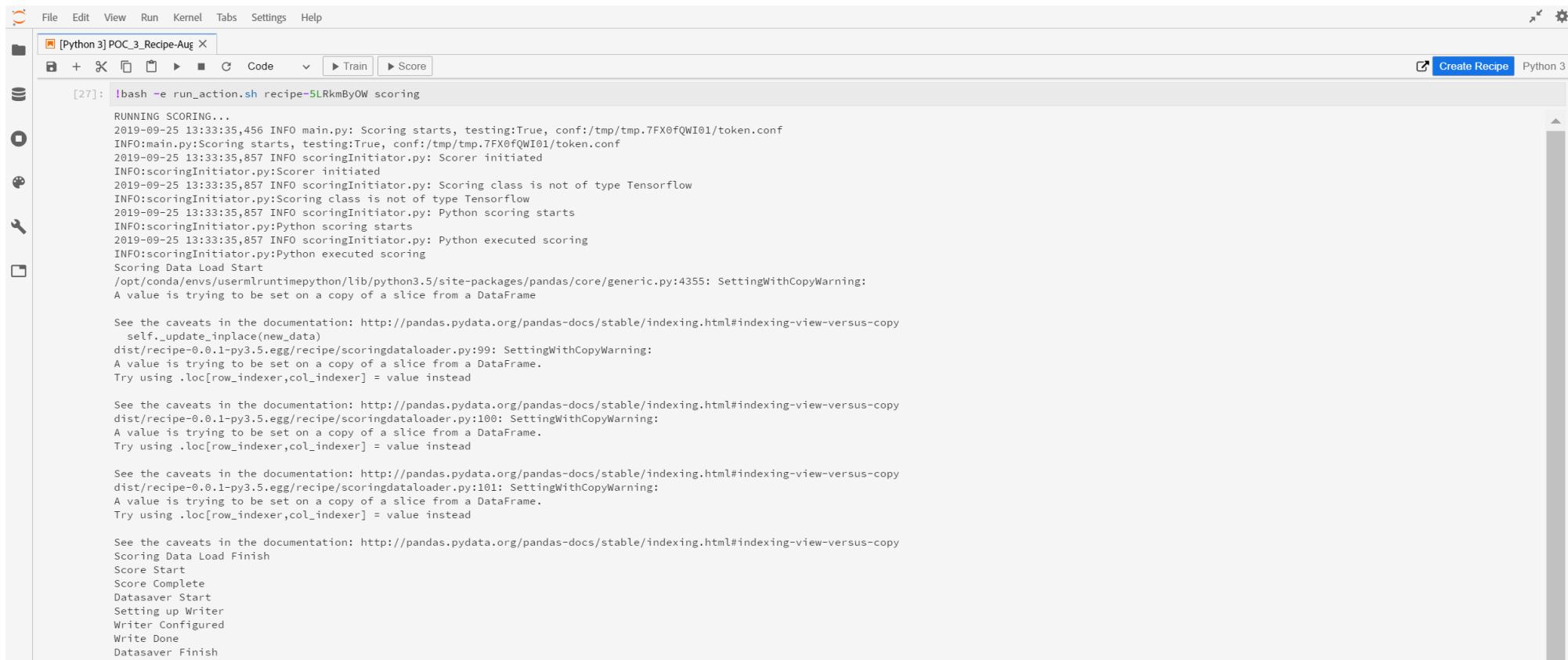
The screenshot shows a Jupyter Notebook interface with the following details:

- File Bar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help.
- Toolbar:** Python 3, Create Recipe, Python 3, and several icons for file operations.
- Code Cell:** [36]: `!bash -e run_action.sh recipe=5LRkmBy0W training`
- Output Cell:** Displays log messages from the training process:
 - RUNNING TRAINING...
 - 2019-09-25 13:35:34,367 INFO trainingInitiator.py: Trainer initiated
 - INFO:trainingInitiator.py:Trainer initiated
 - 2019-09-25 13:35:34,367 INFO trainingInitiator.py: Evaluator initiated
 - INFO:trainingInitiator.py:Evaluator initiated
 - 2019-09-25 13:35:34,367 INFO main.py: Training starts, testing:True, conf:/tmp/tmp.wXZnPot1E0/token.conf
 - INFO:main.py:Training starts, testing:True, conf:/tmp/tmp.wXZnPot1E0/token.conf
 - 2019-09-25 13:35:34,367 INFO trainingInitiator.py: Training class is not of type Tensorflow
 - INFO:trainingInitiator.py:Training class is not of type Tensorflow
 - 2019-09-25 13:35:34,368 INFO trainingInitiator.py: Python Job
 - INFO:trainingInitiator.py:Python Job
 - 2019-09-25 13:35:34,368 INFO trainingInitiator.py: Load the dataframe
 - INFO:trainingInitiator.py:Load the dataframe
 - Training Data Load Start
 - /opt/conda/envs/usermruntimepython/lib/python3.5/site-packages/pandas/core/generic.py:4355: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
 - See the caveats in the documentation: <http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy>
self._update_inplace(new_data)
 - dist/recipe-0.0.1-py3.5.egg/recipe/trainingdataloader.py:97: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
 - See the caveats in the documentation: <http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy>
dist/recipe-0.0.1-py3.5.egg/recipe/trainingdataloader.py:98: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
 - See the caveats in the documentation: <http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy>
dist/recipe-0.0.1-py3.5.egg/recipe/trainingdataloader.py:99: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
 - See the caveats in the documentation: <http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy>
Training Data Load Finish
 - 2019-09-25 13:35:58,824 INFO trainingInitiator.py: Evaluator class is defined for python
 - INFO:trainingInitiator.py:Evaluator class is defined for python
 - Initiate
 - 2019-09-25 13:35:58,826 INFO trainingInitiator.py: Python Training started
 - INFO:trainingInitiator.py:Python Training started
 - Train Start
 - Train Complete
 - 2019-09-25 13:35:58,831 INFO Helper.py: Save Model completed : trainedModel
 - INFO:Helper.py:Save Model completed : trainedModel
 - 2019-09-25 13:35:58,831 INFO trainingInitiator.py: Python Training completed
 - INFO:trainingInitiator.py:Python Training completed
 - 2019-09-25 13:35:58,831 INFO trainingInitiator.py: Evaluation will be on the test data
 - INFO:trainingInitiator.py:Evaluation will be on the test data
 - 2019-09-25 13:35:58,831 INFO trainingInitiator.py: Evaluate config is set to true
 - INFO:trainingInitiator.py:Evaluate config is set to true
 - 2019-09-25 13:35:58,831 INFO evaluateInitiator.py: Starting evaluation
 - INFO:evaluateInitiator.py:Starting evaluation
 - Initiate
 - Evaluation evaluate triggered
- Snipping Tool Overlay:** A floating window titled "Snipping Tool" with options for "New", "Mode", "Delay", "Cancel", and "Options". It says "Select the snip mode using the Mode button or click the New button."

Recipe Building

Build a recipe out of the trained model to be used as a reusable component to score multiple populations

Step 2 –Score the model



The screenshot shows a Jupyter Notebook interface with the following details:

- File Bar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help.
- Toolbar:** Python 3, POC_3_Recipe-Aug, Create Recipe, Python 3.
- Code Cell:** [27]: `!bash -e run_action.sh recipe=SLRkmByOW scoring`
- Output:** The cell outputs the following log messages:
 - RUNNING SCORING...
 - 2019-09-25 13:33:35,456 INFO main.py: Scoring starts, testing=True, conf:/tmp/tmp.7FX0fQWI01/token.conf
 - INFO:main.py:Scoring starts, testing=True, conf:/tmp/tmp.7FX0fQWI01/token.conf
 - 2019-09-25 13:33:35,857 INFO scoringInitiator.py: Scorer initiated
 - INFO:scoringInitiator.py:Scorer initiated
 - 2019-09-25 13:33:35,857 INFO scoringInitiator.py: Scoring class is not of type Tensorflow
 - INFO:scoringInitiator.py:Scoring class is not of type Tensorflow
 - 2019-09-25 13:33:35,857 INFO scoringInitiator.py: Python scoring starts
 - INFO:scoringInitiator.py:Python scoring starts
 - 2019-09-25 13:33:35,857 INFO scoringInitiator.py: Python executed scoring
 - INFO:scoringInitiator.py:Python executed scoring
 - Scoring Data Load Start
 - /opt/conda/envs/usermlruntimepython/lib/python3.5/site-packages/pandas/core/generic.py:4355: SettingWithCopyWarning: A value is trying to be set on a copy of a slice from a DataFrame
 - See the caveats in the documentation: <http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy>
 - self._update_inplace(new_data)
 - dist/recipe-0.0.1-py3.5.egg/recipe/scoringdataloader.py:99: SettingWithCopyWarning:
 - A value is trying to be set on a copy of a slice from a DataFrame.
 - Try using .loc[row_indexer,col_indexer] = value instead
 - See the caveats in the documentation: <http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy>
 - dist/recipe-0.0.1-py3.5.egg/recipe/scoringdataloader.py:100: SettingWithCopyWarning:
 - A value is trying to be set on a copy of a slice from a DataFrame.
 - Try using .loc[row_indexer,col_indexer] = value instead
 - See the caveats in the documentation: <http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy>
 - dist/recipe-0.0.1-py3.5.egg/recipe/scoringdataloader.py:101: SettingWithCopyWarning:
 - A value is trying to be set on a copy of a slice from a DataFrame.
 - Try using .loc[row_indexer,col_indexer] = value instead
 - See the caveats in the documentation: <http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy>
 - Scoring Data Load Finish
 - Score Start
 - Score Complete
 - Datasaver Start
 - Setting up Writer
 - Writer Configured
 - Write Done
 - Datasaver Finish

Recipe Building

Build a recipe out of the trained model to be used as a reusable component to score multiple populations

Step 3 –Build Recipe – Give a name

The screenshot shows a software interface for building machine learning recipes. At the top, there's a menu bar with File, Edit, View, Run, Kernel, Tabs, Settings, and Help. Below the menu is a toolbar with various icons and buttons, including a 'Create Recipe' button which is highlighted in blue. The main window has a title bar '[Python 3] POC_3_Recipe-Aug X'. On the left, there's a sidebar with icons for file operations like New, Open, Save, and a search bar. The central area is titled 'Recipe Builder Actions Overview'.

- Saving a File Cell:** A note says: "If you wish to save the contents of a cell, simply run it. The `%writefile` command at the top of the cell will write the contents of the cell to the file named at the top of the cell. You should run the cells manually when applicable. However, pressing any of the actions at the top will automatically run all file cells relevant to the action."
- Training and Scoring:** A note says: "Press the associated buttons at the top in order to run training or scoring. The training output will be shown below the `evaluator.py` cell and scoring output will be shown below the `datasaver.py` cell. You must run training at least once before you can run scoring. You may delete the output cell(s). Running training the first time or after changing `requirements.txt` will be slower since the dependencies for the recipe need to be installed, but subsequent runs will be significantly faster. If you wish to see the hidden output add `debug` to the end of the output cell and re-run it."
- Creating the Recipe:** A note says: "When you are done editing the recipe and satisfied with the training/scoring output, you can create a recipe. This will generate a `recipe.py` file in your workspace. You must run scoring at least once before you can create the recipe. After pressing it, you will see a progress bar showing how much time is left for the build to finish. If the recipe creation is successful then you can click to navigate to the created recipe."
- Caution!**
 - Do not delete any of the file cells
 - Do not edit the `%writefile` line at the top of the file cells
- Requirements File (Optional):** A note says: "Add additional libraries you wish to use in the recipe to the cell below. You can specify the version number if necessary. The file cell below is a commented out example."

In the bottom right corner of the main window, a modal dialog box is open with the title "Enter Recipe Name". It contains a text input field with the value "Sample_Recipe", a "DISMISS" button, and an "OK" button.

At the bottom of the screen, there's a decorative footer with a colorful abstract pattern and the text "© 2016 Adobe Systems Incorporated. All Rights Reserved. Adobe Confidential." and the Adobe logo.

Recipe Building

Build a recipe out of the trained model to be used as a reusable component to score multiple populations

Step 3 –Build Recipe

The screenshot shows a software interface for building machine learning recipes. At the top, there's a menu bar with File, Edit, View, Run, Kernel, Tabs, Settings, and Help. Below the menu is a toolbar with icons for file operations like Open, Save, and Print, followed by Code, Train, and Score buttons. A progress bar indicates the status of the build process. On the far right, there are buttons for Create Recipe and Python 3.

The main area is titled "Recipe Builder Actions Overview". It contains several sections:

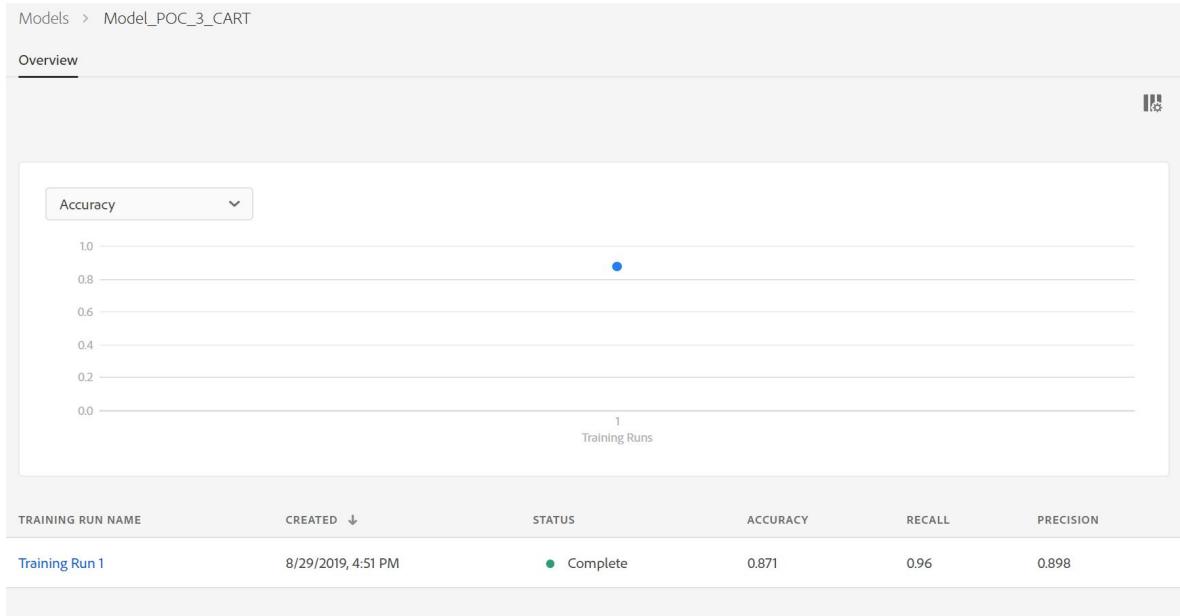
- Saving a File Cell:** A note explaining that running `%%writefile` at the top of a cell will save its contents to a file named at the top of the cell. It also mentions that pressing any action button at the top will run all relevant file cells.
- Training and Scoring:** Instructions for running training or scoring. It says to press the Train or Score button at the top of the notebook to run the associated code cells. Training output is shown below the `evaluator.py` cell, and scoring output is shown below the `datasaver.py` cell. It notes that you must run training at least once before scoring.
- Creating the Recipe:** A note that after saving the file cell, you can run the `%%writefile` command again to update the file. It also says to run training or scoring at least once before creating the recipe.
- Caution!** Two bullet points:
 - Do not delete any of the file cells
 - Do not edit the `%%writefile` line at the top of the file cells
- Requirements File (Optional):** A section for specifying additional libraries. It includes a code cell example:

```
[37]: %%writefile ~/my-workspace/.recipes/recipe-5LRkmBy0W/requirements.txt
# pandas==0.22.0
# numpy
```

The interface also includes a progress bar for the build process and a message box for "Recipe Creation Started".

Train and Evaluate ML Model

Train the model on a subset of the data and evaluate on other subset to check the performance.



Models > Model_POC_3_CART > Training Run 1

Evaluation Scoring Runs

 View Activity Logs

Evaluation Metrics

NAME	VALUE
Accuracy	0.8706467661691543
Recall	0.96
Precision	0.8983957219251337

Configuration Parameters

NAME	VALUE
batch_id	19403cbf4ddb44c4832f8501c352d491
max_depth	5
n_estimators	100
learning_rate	0.1

Score New Data

Use the trained model to score new data coming into the platform and record model predictions in the output dataset.

Models > Model_POC_3_CART > Training Run 1

Score

Evaluation Scoring Runs

SCORING RUN NAME	CREATED	STATUS
Scoring Run 1	8/29/2019, 5:00 PM	● Complete

Scoring Run

Preview Scoring Results Dataset

View Activity Logs

Scoring Run Name
Scoring Run 1

Created
8/29/2019, 5:00 PM

Scoring Run

Churn_Score_Schema

CHURN_SCORE_SCHEMA

	_aepgdcdevenlement3.april_revenue	_aepgdcdevenlement3.ecid_dsw	_aepgdcdevenlement3.Churn_Prediction
0	330.0	22155109217195399804340517356118708926	1
1	157.0	01122191937004221938798456707820403531	1
2	146.0	10582750433803258541801790039664520574	1
3	326.0	21470829511243479413986772949353176918	1
4	194.0	32304225634251848422352730383207858122	1
5	248.0	38366223737022310212197256537036620145	0
6	132.0	41860454351187077421827488454061202809	1
7	554.0	13409225164322502642127195998691666723	0

Output Dataset with predictions

Model as a Service : Publish & Schedule

Publish the model as a service and schedule it to run (both train and score) hourly, daily, weekly etc. on the new set of data.

Services > Service_POC3_Churn

+ Score

Overview Scoring Runs Training Runs

 Service_POC3_Churn
Created 8/29/2019, 5:19 PM by Amit doda

 Score

Description: Service_POC3_Churn
Published from Model: Model_POC_3_CART

 Scoring

 Most Recent
Completed on 9/12/2019, 5:30 AM with april_all_columns_joined_churn
 Result Dataset: POC 3 Churn Output Dataset

 Upcoming
Scheduled for 8/29/2019, 5:19 PM with april_all_columns_joined_churn
 Every hour at 9/24/2019, 5:30 AM until 9/27/2019, 5:29 AM
[Update Schedule](#)

[View All](#)

 Training

 Last Trained
8/29/2019, 5:19 PM with april_all_columns_joined_churn
Accuracy: 0.8557213930348259 Recall: 0.9428571428571428
Precision: 0.8967391304347826

 Upcoming
No Training Scheduled
[Update Schedule](#)

Scoring Runs

Hourly scheduled scoring runs of a sample ML Service.

Overview	Scoring Runs	Training Runs
SCORING RUN NAME	CREATED ↓	STATUS
Scoring Run 25	9/12/2019, 5:30 AM	● Complete
Scoring Run 24	9/12/2019, 4:32 AM	● Complete
Scoring Run 23	9/12/2019, 3:31 AM	● Complete
Scoring Run 22	9/12/2019, 2:30 AM	● Complete
Scoring Run 21	9/12/2019, 1:33 AM	● Complete
Scoring Run 20	9/12/2019, 12:32 AM	● Complete
Scoring Run 19	9/11/2019, 11:31 PM	● Complete
Scoring Run 18	9/11/2019, 10:31 PM	● Complete
Scoring Run 17	9/11/2019, 9:30 PM	● Complete
Scoring Run 16	9/11/2019, 8:30 PM	● Complete



Adobe