Predicting Insurance Fraud Analyzed in Jupyter Notebooks

Hands-on Lab

Overview

Combatting fraud and performing investigative action demands an end-to-end data science solution. It empowers an organization to scale analysis with ready access to public clouds, private clouds and on-premises. The platform also speeds modeling, training and deployment time and simplifies collaboration with data scientists, risk analysts, investigators, and other subject matter experts while adhering to strong governance and security posture. Further, in order to respond to new types of fraud, waste and abuse while minimizing false negatives and accelerating response, the platform needs to continuously accommodate real-time data, monitor and detect fraudulent activities and adapt as the patterns change and spot anomalies.

The global Fraud Detection and Prevention (FDP) market size is expected to grow from USD 20 billion to 63.5 billion by 2023, according to various analyst reports (i.e. "Fraud Detection and Prevention Market by solution). Predictive analytics segment is projected to be the largest contributor to the FDP market during the forecast period.

Predictive analytics solutions help enterprises identify the possibilities of fraud incidents by analyzing the current data. The solutions are used to identify potential threats, payment frauds, frauds in insurance processes, and credit/debit card frauds. Organizations are trying to impart these solutions for predicting fraud or suspicious activity and their pattern to help drastically reduce losses due to frauds.

A global fraud report from Experian says that 72% of businesses cite fraud as a growing concern.

Digital transformation has created data issues that make it difficult to detect fraud. Silos of data residing in your lines of business, departments, and geos and varying analytical techniques across channels and transaction systems have opened you up to increased risk exposures and attacks from fraudsters.

It's time to track behavior and exposure so you can prevent fraud before it happens. In this lab you will perform all of the steps that you undertook with a UI-driven Watson Studio activities, except that you will perfrom these tasks in Jupyter Notebooks. The code snipets are provided, so need for coding acumen, yet you are welcome to experiment with the code and realize other insights and feature engineering task that may reveal other behavior indicative of fraudulent auto insurance claims.

The app below represents relevant data to the data scientist. The focal point of this lab is about fraud prediction by assigning a probability value to certain behavior that may predict fraudulent behavior.

Estimated Time to Complete: 1 Hour

Objectives

The following constitute hypothesis formulated by subject matter experts (SMEs) in the field of auto insurance:

- Loss event claimed within 15 days of policy expiration
- Expired drivers' license
- Expensive vehicle damages
- Frequent changes of residence
- High mileage at loss event for a policyholder with a low mileage discount
- High number of previous claims
- No police report

You will build models that:

- Find the data that shows the fraud indicators
- Prepare a training data set with the fraud indicators
- Train a fraud prediction model
- Deploy the model for use in the fraud triaging app

The learning goals of this notebook are:

- Load the auto insurance CSV file into the Object Storage Service linked to your Watson Studio project or import the Jupyter Notebook into your project. In this lab you will build the notebook from scratch.
- Create an Apache® Spark machine learning model
- Train and evaluate a model
- Persist a model in a Watson Machine Learning repository

There are five Milestones you must complete:

- Create the Notebook
- About Running Jupyter notebooks
- Importing libraries and Understanding the Raw data
- Understanding the Data
- Feature Engineering

Tools

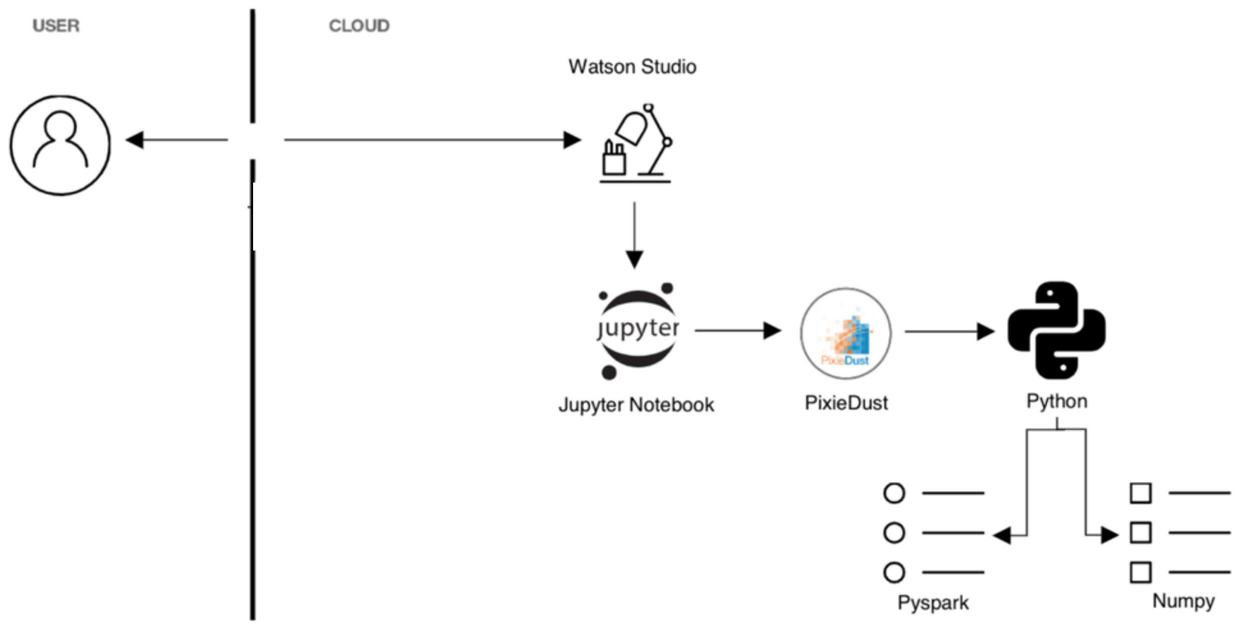
	Watson Studio
	Jupyter Notebook
	PixieDust
	Python (Pyspark and numpy)

Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

PixieDust is an open source Python helper library that works as an add-on to Jupyter notebooks to improve the user experience of working with data. It also fills a gap for users who have no access to configuration files when a notebook is hosted on the cloud.

Python is an interpreted, high-level, general-purpose programming language. Pyspark and Numpy are Python libraries.

Flow



1. The User will create a Watson Studio Service.
2. From Watson Studio connect to Jupyter Notebook.
3. Utilize PixieDust, an open-source visualization program.
4. Manipulate Python code using Python Libraries, such as Pyspark, sklearn and Numpy.

Prerequisites

You must have completed all prior labs before beginning this lab.

Milestone 1: Create the Notebook

Milestone Overview

This lab requires you to complete five Milestones:

1. **Create the Notebook**
2. About Running Jupyter notebooks
3. Importing libraries and Understanding the Raw data
4. Understanding the Data
5. Feature Engineering

By now you have already registered with IBM Cloud and created your Watson Studio service. Let's begin our journey.

Steps

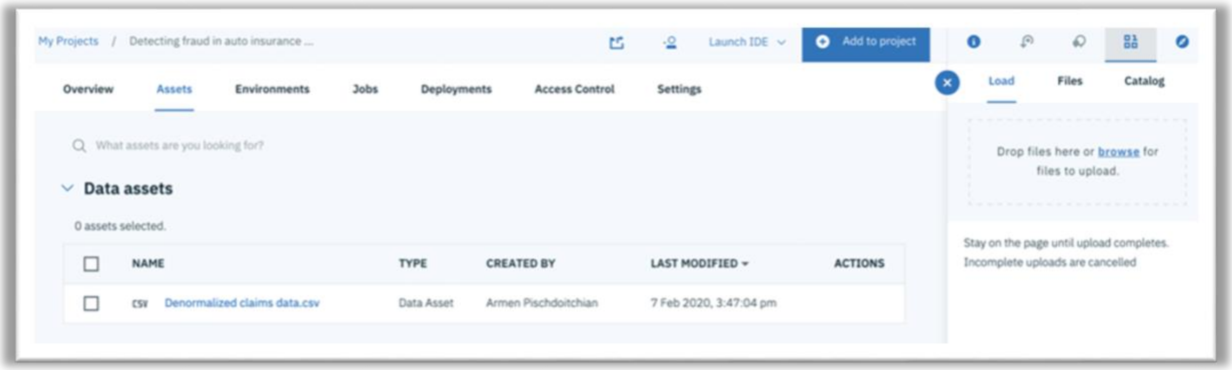
1. Access the Waston Studio service from the IBM Cloud Dashboard
2. Click **Create a project**.
3. Select the **Create an empty project** tile.
4. Specify a name; for example: **Predicting fraud in auto insurance claims**.
5. Click **Create**.

If this is your first-time visiting Watson Studio, you may need to add a **Cloud Object Storage (COS)** if the cloud-object-storage does not appear and the Create button is greyed out. Click the link under Choose project options and after you add the COS, click **Refresh** so you can view the COS instance. This is a one-time event when you first provision the Watson Studio service.

6. Click the **Assets** tab
7. Navigate to the link below and download the CSV file. Once in Github, click **Raw** and then **File -> Save Page As...**

https://github.com/apischdo/skillsacademy/blob/master/Denormalized%20claims%20data.csv

8. **Browse** to your newly downloaded CSV file and click **Open** to upload the file.



Data assets

0 assets selected.

NAME	TYPE	CREATED BY	LAST MODIFIED	ACTIONS
CSV Denormalized claims data.csv	Data Asset	Armen Pischdoitchian	7 Feb 2020, 3:47:04 pm	

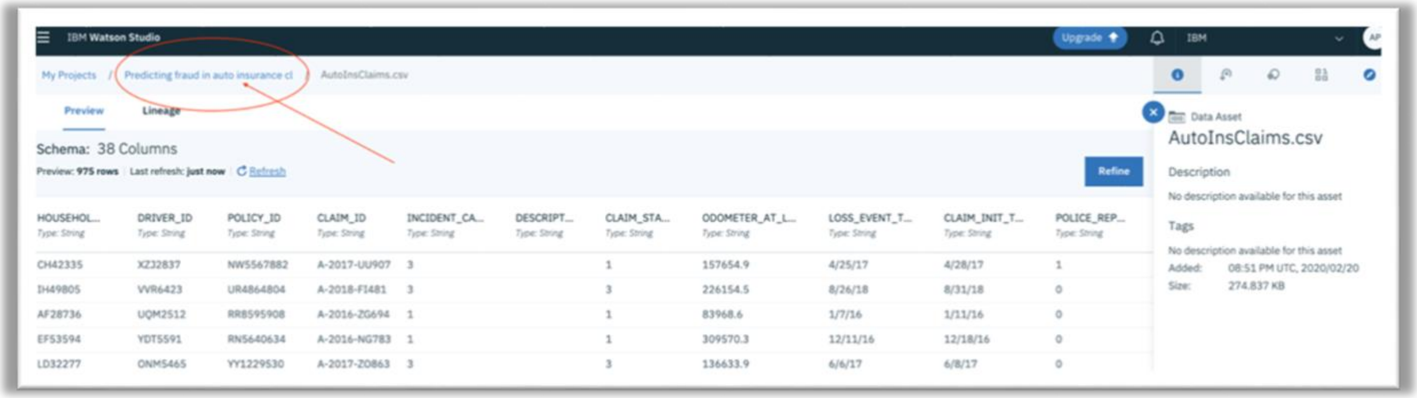
Drop files here or browse for files to upload.

Stay on the page until upload completes. Incomplete uploads are cancelled

Figure 1-1 **My Projects Page**

9. Click the data set (the CSV file) to preview and close the right-side panel so you can view more. Notice that this file has more columns yet depicts the same story as the exercises in Lab 2. In this exercise you will explore more features (columns) and build slightly different models based on some new data.

10. There is no need to click Refinery, you will perform all tasks in Jupyter Notebooks.


IBM Watson Studio
Upgrade
IBM
AP
My Projects / Predicting fraud in auto insurance cl / AutoInsClaims.csv
Preview
Lineage
Refine

Schema: 38 Columns

Preview: 975 rows | Last refresh: just now | Refresh

HOUSEHOL... Type: String	DRIVER_ID Type: String	POLICY_ID Type: String	CLAIM_ID Type: String	INCIDENT_CA... Type: String	DESCRIPT... Type: String	CLAIM_STA... Type: String	ODOMETER_AT_L... Type: String	LOSS_EVENT_T... Type: String	CLAIM_INIT_T... Type: String	POLICE_REP... Type: String
CH42335	XZJ2837	NW5567882	A-2017-UU907	3		1	157654.9	4/25/17	4/28/17	1
IH49805	VVR6423	UR4864804	A-2018-FI481	3		3	226154.5	8/26/18	8/31/18	0
AF28736	UQM2512	RR8595908	A-2016-ZG694	1		1	83968.6	1/7/16	1/11/16	0
EF53594	YDT5591	RN5640634	A-2016-NG783	1		1	309570.3	12/11/16	12/18/16	0
LD32277	ONM5465	YY1229530	A-2017-Z0863	3		3	136633.9	6/6/17	6/8/17	0

Data Asset

AutoInsClaims.csv

Description

No description available for this asset

Tags

No description available for this asset

Added: 08:51 PM UTC, 2020/02/20

Size: 274.837 KB

Figure 1-2 **My Projects Page**

11. There are two common ways that you can work with this data:

You can insert the data into a data frame by using the Pandas or merely call the data set directly from Github. In this example, you will insert it into Jupyter Notebooks using PixieDust.

12. Click **Add to project** and select **Notebook**.

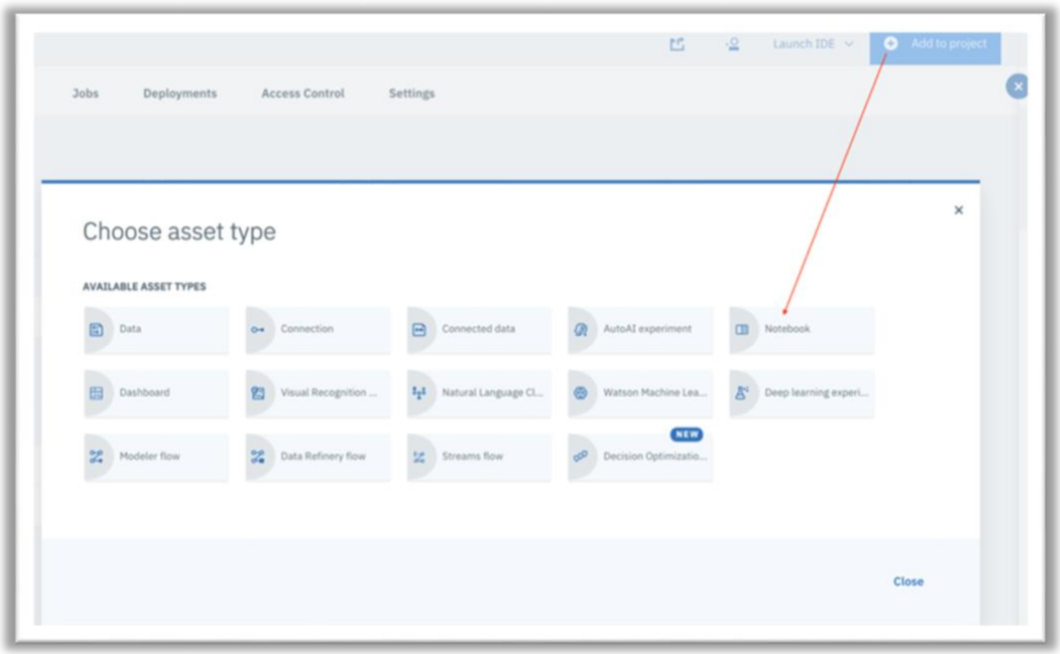
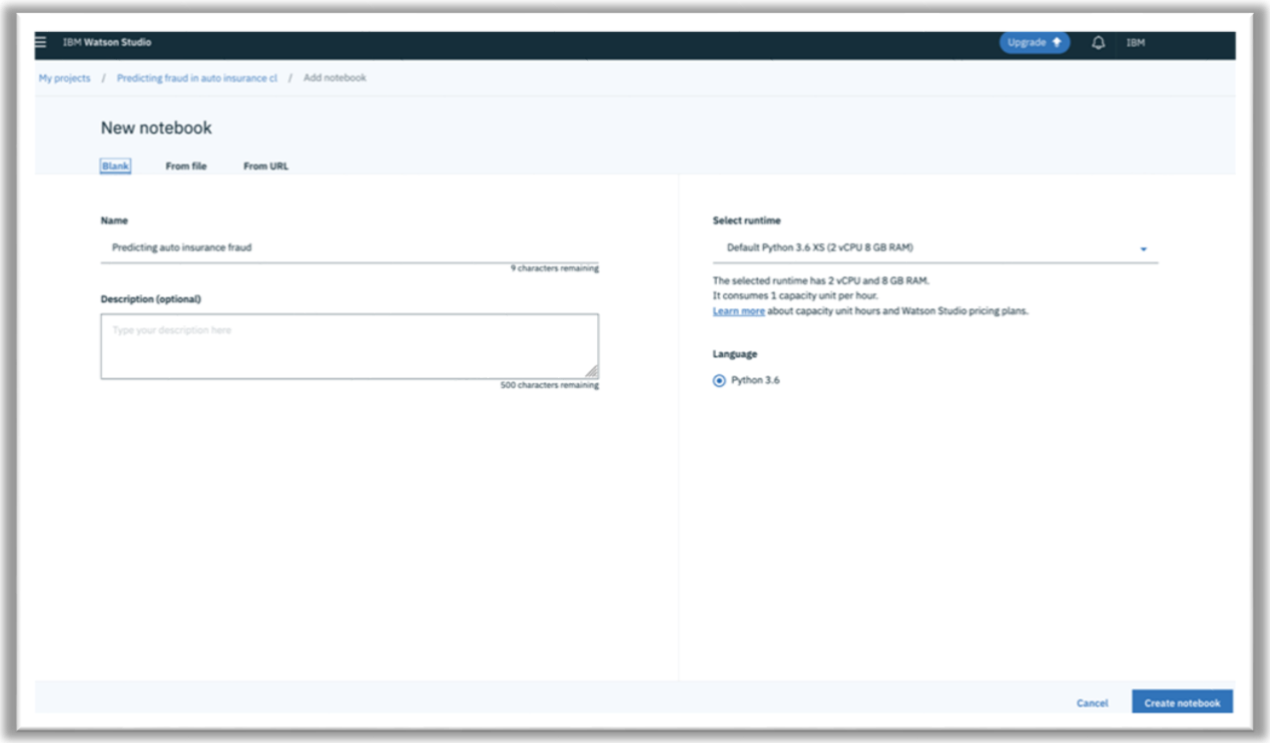


Figure 1-3 **Choose Asset type page**

13. Select **Blank**

14. Give it a meaningful name. For example: Predicting auto insurance fraud

15. Click **Create notebook**.



New notebook

Name

Predicting auto insurance fraud

9 characters remaining

Description (optional)

Type your description here

500 characters remaining

Select runtime

Default Python 3.6 XS (2 vCPU 8 GB RAM)

The selected runtime has 2 vCPU and 8 GB RAM.
It consumes 1 capacity unit per hour.
Learn more about capacity unit hours and Watson Studio pricing plans.

Language

[✓ Python 3.6]

Figure 1-4 **New Notebook**

16. Ensure that the kernel depicts Trusted by clicking Not Trusted and then Trust. All the notebook the few seconds it needs to save that setting.

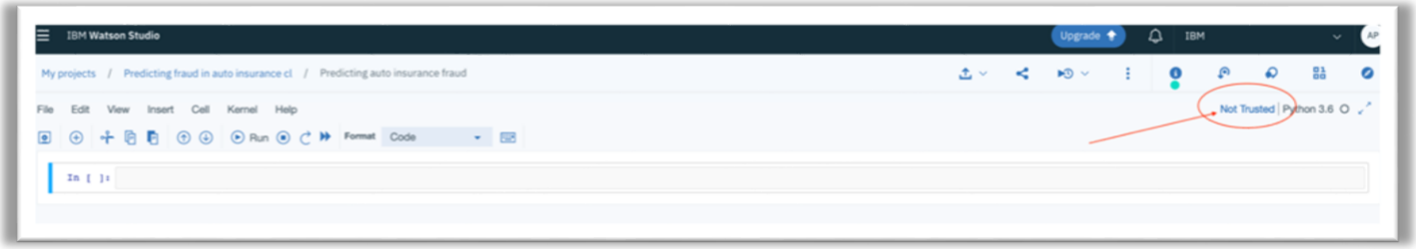

IBM Watson Studio
Upgrade
IBM
My projects / Predicting fraud in auto insurance cl / Predicting auto insurance fraud
File Edit View Insert Cell Kernel Help
Run
Format
Code
Not Trusted
Python 3.6
In [ ]:

Figure 1-5 **Not Trusted button position**

Milestone Summary

In this Milestone you learned how to create a Notebook. Now you can run small pieces of code that process your data, and you can immediately view the results of your computation.

Milestone 2: About Running Jupyter notebooks

Milestone Overview

This lab requires you to complete five Milestones:

1. Create the Notebook
2. **About Running Jupyter notebooks**
3. Importing libraries and Understanding the Raw data
4. Understanding the Data
5. Feature Engineering

When a notebook is executed, what is actually happening is that each code cell in the notebook is executed, in order, from top to bottom.

Each code cell is selectable and is preceded by a tag in the left margin. The tag format is In [x]:. Depending on the state of the notebook, the x can be:

- A blank, this indicates that the cell has never been executed.
- A number, this number represents the relative order this code step was executed.
- A *, this indicates that the cell is currently executing.

There are several ways to execute the code cells in your notebook:

- One cell at a time.
  - Select the cell, and then press the Play button in the toolbar.
- Batch mode, in sequential order.
  - From the Cell menu bar, there are several options available. For example, you can Run All cells in your notebook, or you can Run All Below, that will start executing from the first cell under the currently selected cell, and then continue executing all cells that follow.
- At a scheduled time.
  - Press the Schedule button located in the top right section of your notebook panel. Here you can schedule your notebook to be executed once at some future time, or repeatedly at your specified interval.

After running each cell of the notebook, the results will display.

Milestone 3: Importing libraries and Understanding the Raw data

Milestone Overview

This lab requires you to complete five Milestones:

1. Create the Notebook
2. About Running Jupyter notebooks
3. **Importing libraries and Understanding the Raw data**
4. Understanding the Data
5. Feature Engineering

You are about to copy paste the following commands in the notebook cells. Ensure to wait until the star to the left of the cell has turned into a number, then move to the next cell using the + sign (just under the edit menu item)

Steps

1. Copy and paste the following commands (all of it in the first cell). One of the first things that you do in notebooks, is to install and import libraries that will do various operations for you.

```
!pip install pandas_profiling
```

2. Click Run from the top menu bar. Allow for the cell to run and install scikit and pixiedust. Only after the [*] in a cell has become a whole number, then proceed to the next cell.
3. Click the + sign from the menu bar to create another cell below.

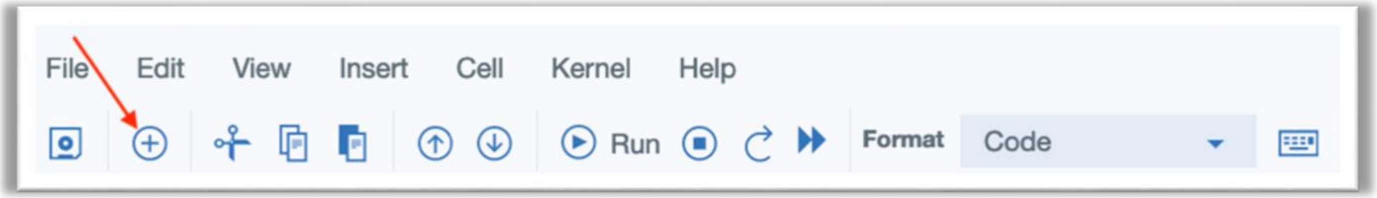


Figure 3-1 **Menu Bar**

4. Copy and paste the following command in the newly created cell to install the LightGBM.

   LightGBM, short for Light Gradient Boosting Machine, is a free and open source distributed gradient boosting framework for machine learning originally developed by Microsoft. It is based on decision tree algorithms and used for ranking, classification and other machine learning tasks.

```
!pip install lightgbm
```

Allow for the cell to run completely. Enter the following commands in the new cell to import libraries such as numpy,

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import ipaddress
import pandas_profiling as pp
%matplotlib inline
from sklearn import preprocessing
plt.rc("font", size=14)
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
import warnings
warnings.filterwarnings("ignore")
import time
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
import lightgbm as lgb
from lightgbm import LGBMClassifier
import seaborn as sns
sns.set(style="white")
sns.set(style="whitegrid", color_codes=True)
```

5. Use the Pandas data framework to insert the csv file by entering the following commands:

```
import types
import pandas as pd
url = 'https://raw.githubusercontent.com/IBM/predict-fraud-using-auto-
ai/master/data/fraud_dataset.csv'
df = pd.read_csv(url)
print(df.head())
print(df.shape)
```

6. Take a moment and review the columns and values within the table
7. Enter (copy/paste) the following command in a new cell:

```
sns.countplot(x='Fraud_Risk',data=df, palette='hls')
```

```
    plt.show()
```

1. Enter (copy/paste) the following command in a new cell:

```
df.groupby('Fraud_Risk').mean()
```

2. Enter (copy/paste) the following command in a new cell:

```
    df.corr(method ='pearson')
```

3. Enter (copy/paste) the following command in a new cell:

```
    X = df[df.columns[0:12]]
    y = df[df.columns[12:]]
```

4. Enter (copy/paste) the following command in a new cell:

```
df.dtypes
```

5. Enter (copy/paste) the following command in a new cell:

```
df.isna()
```

6. Enter (copy/paste) the following command in a new cell:

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
```

7. Enter (copy/paste) the following command in a new cell:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

8. Enter (copy/paste) the following command in a new cell:

```
raw_df.loc[raw_df["CLAIM_AMOUNT"] <3000, 'EXCESSIVE_CLAIM_AMOUNT'] = 0
raw_df.loc[raw_df["CLAIM_AMOUNT"] >=3000, 'EXCESSIVE_CLAIM_AMOUNT'] = 1
```

9. Enter (copy/paste) the following command in a new cell:

```
# dataframes for certain features
features = ['FLAG_FOR_FRAUD_INV',
 'SUSPICIOUS_MILEAGE',
 'EXPIRED_LICENSE',
 'SUSPICIOUS_CLAIM_TIME',
 'SUSPICIOUS_LIVING',
 'EXCESSIVE_CLAIM_AMOUNT']
```

10. Enter (copy/paste) the following command in a new cell:

```
df_model = raw_df[features]
```

11. Enter (copy/paste) the following command in a new cell:

```
#ensure all relevant features are integers
df_model["SUSPICIOUS_LIVING"] = df_model["SUSPICIOUS_LIVING"].astype(int)
df_model["EXPIRED_LICENSE"] = df_model["EXPIRED_LICENSE"].astype(int)
df_model["SUSPICIOUS_CLAIM_TIME"] = df_model["SUSPICIOUS_CLAIM_TIME"].astype(int)
df_model["SUSPICIOUS_MILEAGE"] = df_model["SUSPICIOUS_MILEAGE"].astype(int)
df_model["EXCESSIVE_CLAIM_AMOUNT"] = df_model["EXCESSIVE_CLAIM_AMOUNT"].astype(int)
```

12. Enter (copy/paste) the following command in a new cell:

```
raw_df.groupby("FLAG_FOR_FRAUD_INV", as_index=False).mean()
```

13. Enter (copy/paste) the following command in a new cell:

```
#split data into x and y variables
xVar =
df_model[["EXPIRED_LICENSE","SUSPICIOUS_CLAIM_TIME","SUSPICIOUS_LIVING","SUSPICIOUS_MILEAGE","EXCE
SSIVE_CLAIM_AMOUNT"]]
yVar = df_model["FLAG_FOR_FRAUD_INV"]
```

14. Enter (copy/paste) the following command in a new cell:

```
xVar.head()
```

15. Enter (copy/paste) the following command in a new cell:

```
#split into a test/train set
X_train, X_test, y_train, y_test = train_test_split(xVar, yVar, test_size=0.2)
print (X_train.shape, y_train.shape)
print (X_test.shape, y_test.shape)
```

16. Enter (copy/paste) the following command in a new cell:

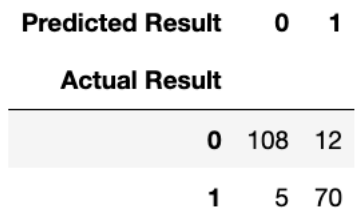
```
#train model
clf = RandomForestClassifier(n_jobs=2, random_state=0)


clf.fit(X_train, y_train)
```

17. Enter (copy/paste) the following command in a new cell:

```
#create confusion matrix to gut check model
preds = clf.predict(X_test)
pd.crosstab(y_test, preds, rownames=['Actual Result'], colnames=['Predicted Result'])
```

Let’s take a moment and understand the significance of the confusion matrix below:



```
Predicted Result    0   1
   Actual Result
               0  108  12
               1    5  70
```

Milestone Summary

In this Milestone you learned the process of using domain knowledge to extract features from raw data via data mining techniques. These features can be used to improve the performance of machine learning algorithm.