IBM Skills Academy

Auto Insurance Fraud Analyzed in Jupyter Notebooks

Lab 6 Guide

The information contained in this document has not been submitted to any formal IBM test and is distributed on an "as is" basis without any warranty either express or implied. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will result elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Contents

Section 1. Preface

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.

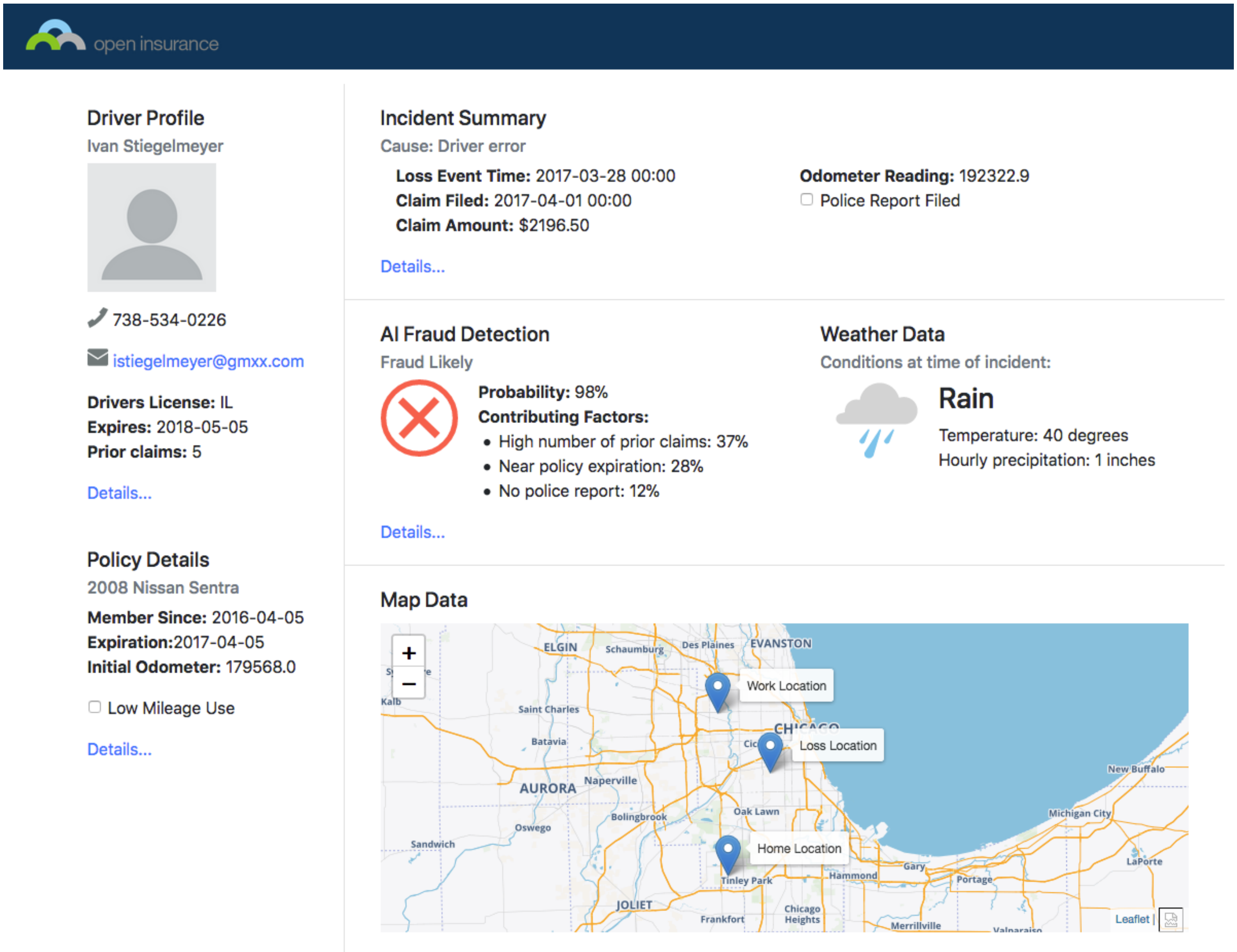
A global fraud report from Experian says that 72% of businesses cite fraud as a growing concern.

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 taks 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.

Watson Studio provides tools to build the data assets used by this app: refine data and build, train, deploy the fraud probability model



Driver Profile

Ivan Stiegelmeyer

☎ 738-534-0226

✉ istiegelmeyer@gmxx.com

Drivers License: IL
Expires: 2018-05-05
Prior claims: 5

Details...

Policy Details

2008 Nissan Sentra

Member Since: 2016-04-05
Expiration:2017-04-05
Initial Odometer: 179568.0

□ Low Mileage Use

Details...

Incident Summary

Cause: Driver error

Loss Event Time: 2017-03-28 00:00
Claim Filed: 2017-04-01 00:00
Claim Amount: $2196.50

Odometer Reading: 192322.9
□ Police Report Filed

Details...

AI Fraud Detection

Fraud Likely

☒ **Probability:** 98%
Contributing Factors:

- High number of prior claims: 37%
- Near policy expiration: 28%
- No police report: 12%

Details...

Weather Data

Conditions at time of incident:

Rain

Temperature: 40 degrees
Hourly precipitation: 1 inches

Map Data

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

Tools

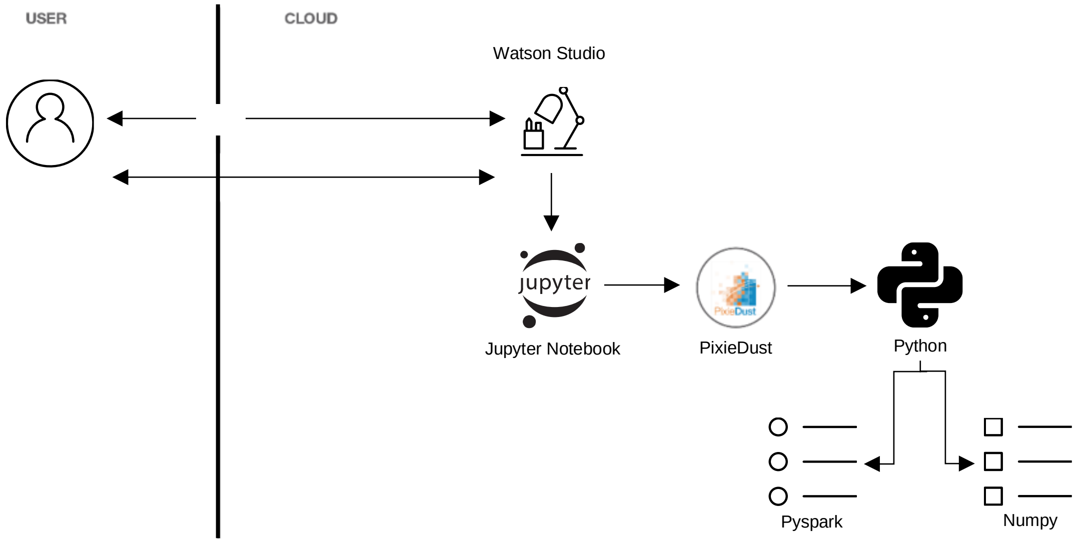
Watson Studio

Jupyter Notebook

PixieDust

Python (sklearn and numpy)

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

This lab assumes that you have completed all prior labs and that you have an active IBM Cloud Account.

Section 2. Create the Notebook

By now you have already registered with IBM Cloud and applied your promo code. Let's begin our journey. If you have been working on prior exercises, the you already have Watson Studio provisioned, hence start from **Step 6**.

If you have done the previous labs…

1. Access the Watson Studio service from the IBM Cloud Dashboard (it's under Services).
2. Open the existing Fraud related project.
3. Begin with Step 10 below.

If you are starting with Watson Studio anew…

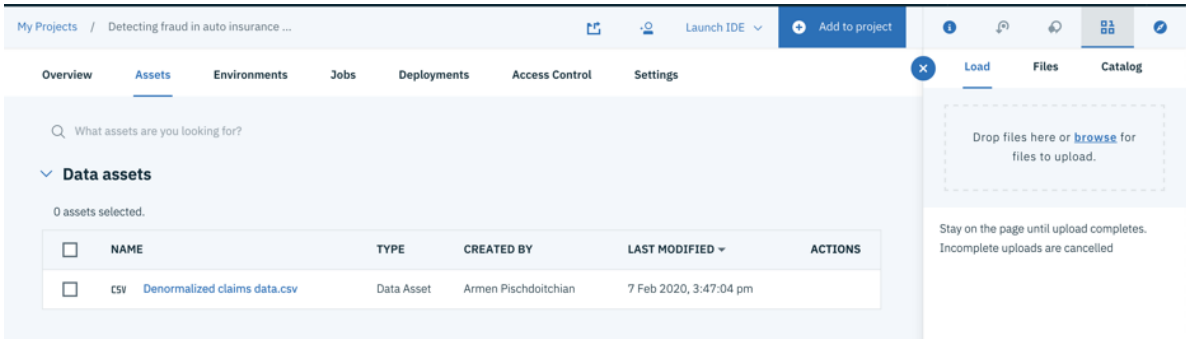
1. Login into IBM Cloud: https://cloud.ibm.com
2. Click the **Catalog** tab.
3. Search for the **Watson Studio** service and click that tile.
4. Click **Create**.
5. Click the **Get Started** button.
6. Click **Create a project**.
7. Select the **Create an empty project** tile
8. Specify a name; for example: **Predicting fraud in auto insurance claims**.
9. 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.

10. Click the **Assets** tab
11. 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

12. **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	

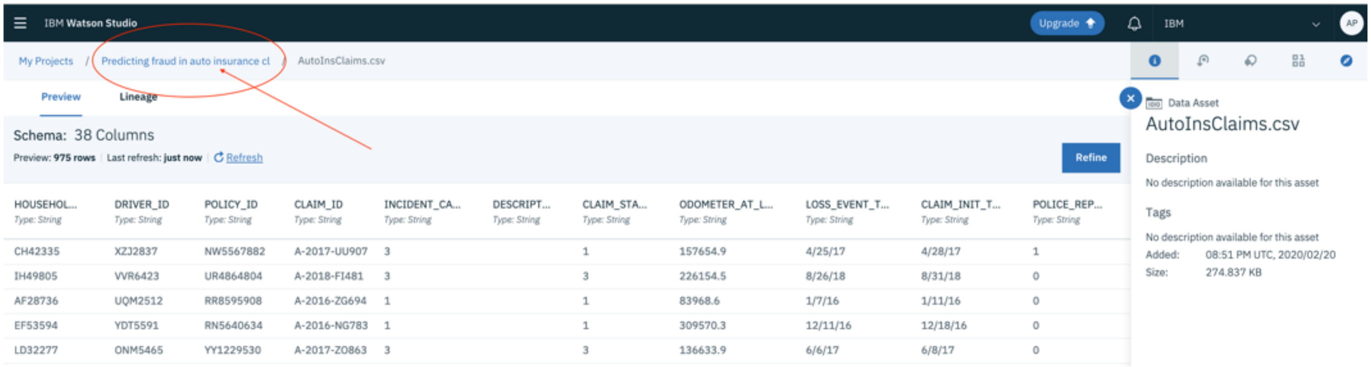
Drop files here or browse for files to upload.

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

13. 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.

14. 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
Refresh
Refine

Schema: 38 Columns

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

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-ZO863	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

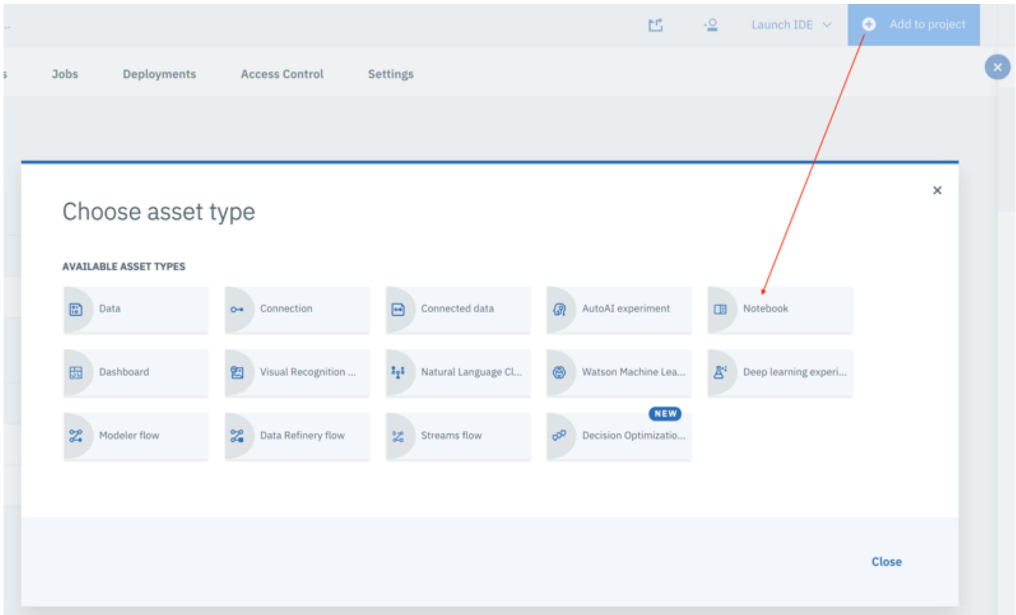
Added: 08:51 PM UTC, 2020/02/20

Size: 274.837 KB

15. 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.

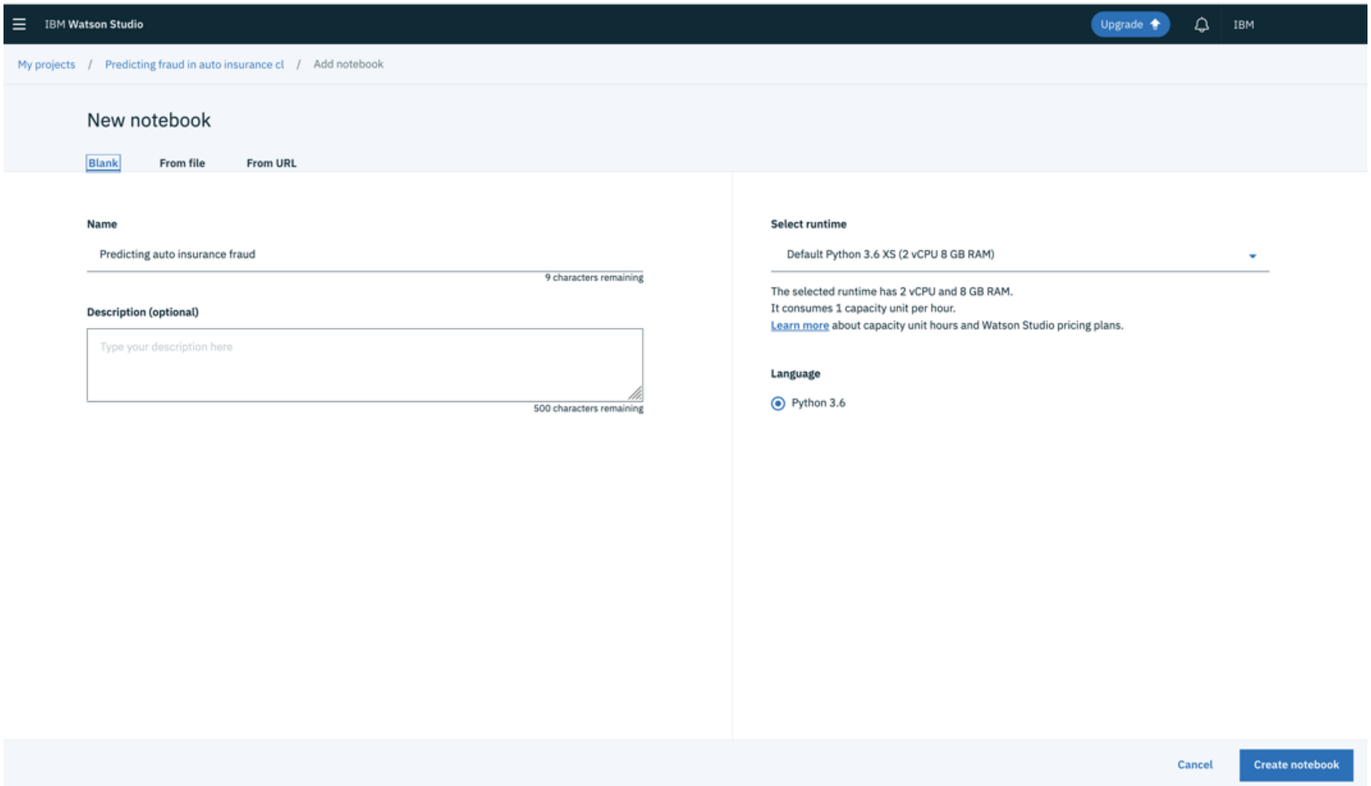
16. Click **Add to project** and select **Notebook**.



17. Select **Blank**

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

19. Click **Create notebook**.


IBM Watson Studio
Upgrade
IBM
My projects / Predicting fraud in auto insurance cl / Add notebook

New notebook

Name

Predicting auto insurance fraud

9 characters remaining

Description (optional)

Type your description here

500 characters remaining

Select runtime

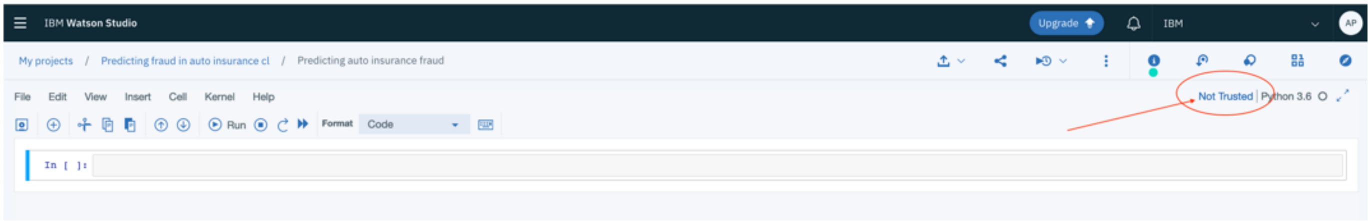
Default Python 3.6 XS (2 vCPU 8 GB RAM)

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

20. 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.



```
In [ ]:
```

About Running Jupyter notebooks

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.

Importing libraries and Understanding the Raw data

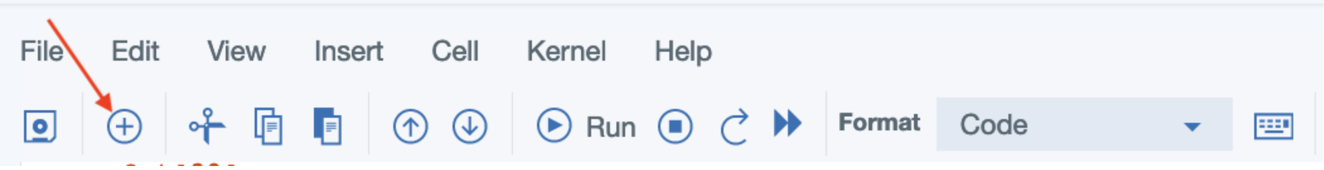
You are about to copy paste the following commands in the notebook cells. Ensure to wait until the star to the left of the cell has turned into a number, then move to the next cell using the + sign (just under the edit menu item)

21. 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 scikit-learn
!pip install --upgrade pixiedust
```

22. 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.

23. Click the + sign from the menu bar to create another cell below.



24. Copy and paste the following command in the newly created cell.

```
import pixiedust
import sklearn
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
import numpy as np
from sklearn.model_selection import train_test_split
from scipy.io import arff
import brunel
from watson_machine_learning_client import WatsonMachineLearningAPIClient
```

Allow for the cell to run completely. This import takes upwards of two minutes. Only after the cell displays [2], then create another cell underneath.

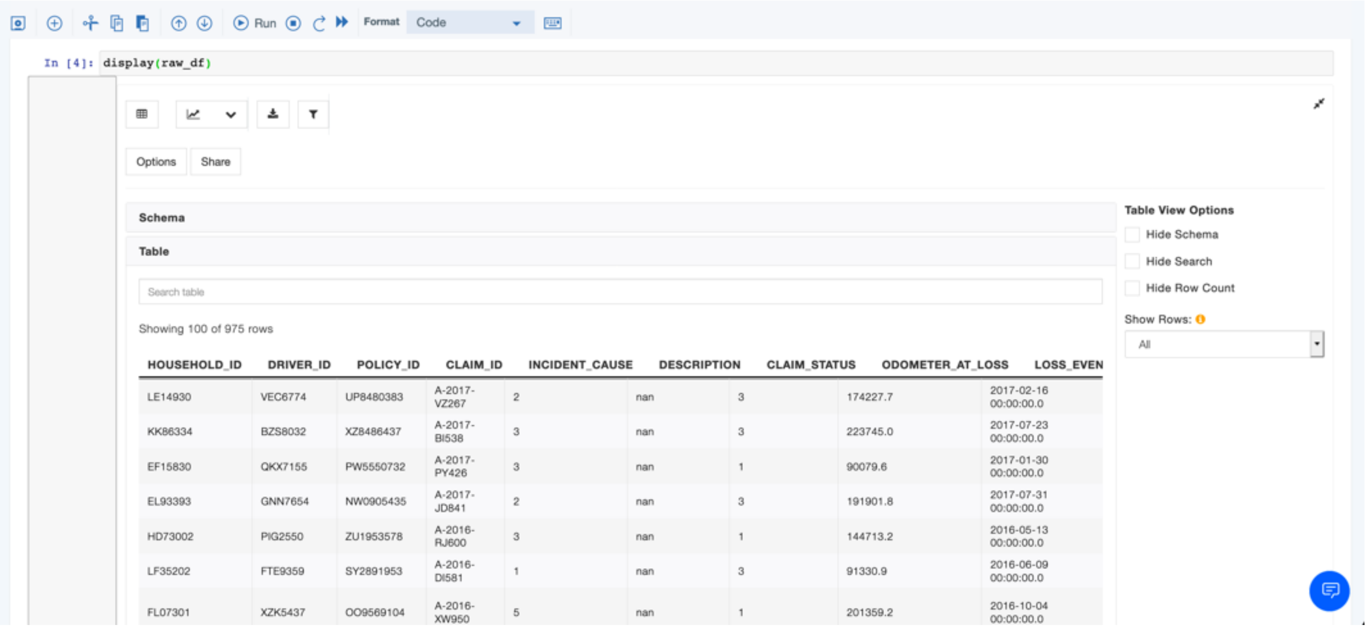
25. Enter the following command in the new cell:

```
raw_df=pixiedust.sampleData('https://raw.githubusercontent.com/apischdo/skillsac
ademy/master/Denormalized%20claims%20data.csv')
```

26. Enter the Display command to view the data set using Pixiedust

```
display(raw_df)
```

27. Take a moment and review the columns and values within the table



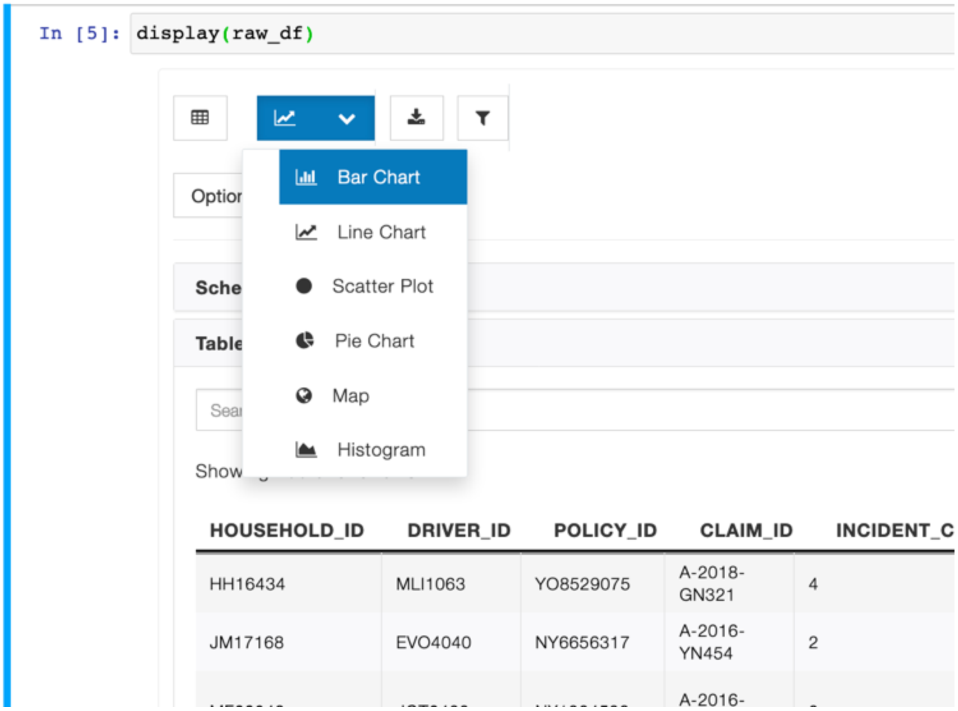
```
In [4]: display(raw_df)
```

Showing 100 of 975 rows

HOUSEHOLD_ID	DRIVER_ID	POLICY_ID	CLAIM_ID	INCIDENT_CAUSE	DESCRIPTION	CLAIM_STATUS	ODOMETER_AT_LOSS	LOSS_EVEN
LE14930	VEC6774	UP8480383	A-2017-VZ267	2	nan	3	174227.7	2017-02-16 00:00:00.0
KK86334	BZS8032	XZ8486437	A-2017-BI538	3	nan	3	223745.0	2017-07-23 00:00:00.0
EF15830	QKX7155	PW5550732	A-2017-PY426	3	nan	1	90079.6	2017-01-30 00:00:00.0
EL93393	GNN7654	NW0905435	A-2017-JD841	2	nan	3	191901.8	2017-07-31 00:00:00.0
HD73002	PIG2550	ZU1953578	A-2016-RJ600	3	nan	1	144713.2	2016-05-13 00:00:00.0
LF35202	FTE9359	SY2891953	A-2016-DI581	1	nan	3	91330.9	2016-06-09 00:00:00.0
FL07301	XZK5437	OO9569104	A-2016-XW950	5	nan	1	201359.2	2016-10-04 00:00:00.0

Understanding the Data

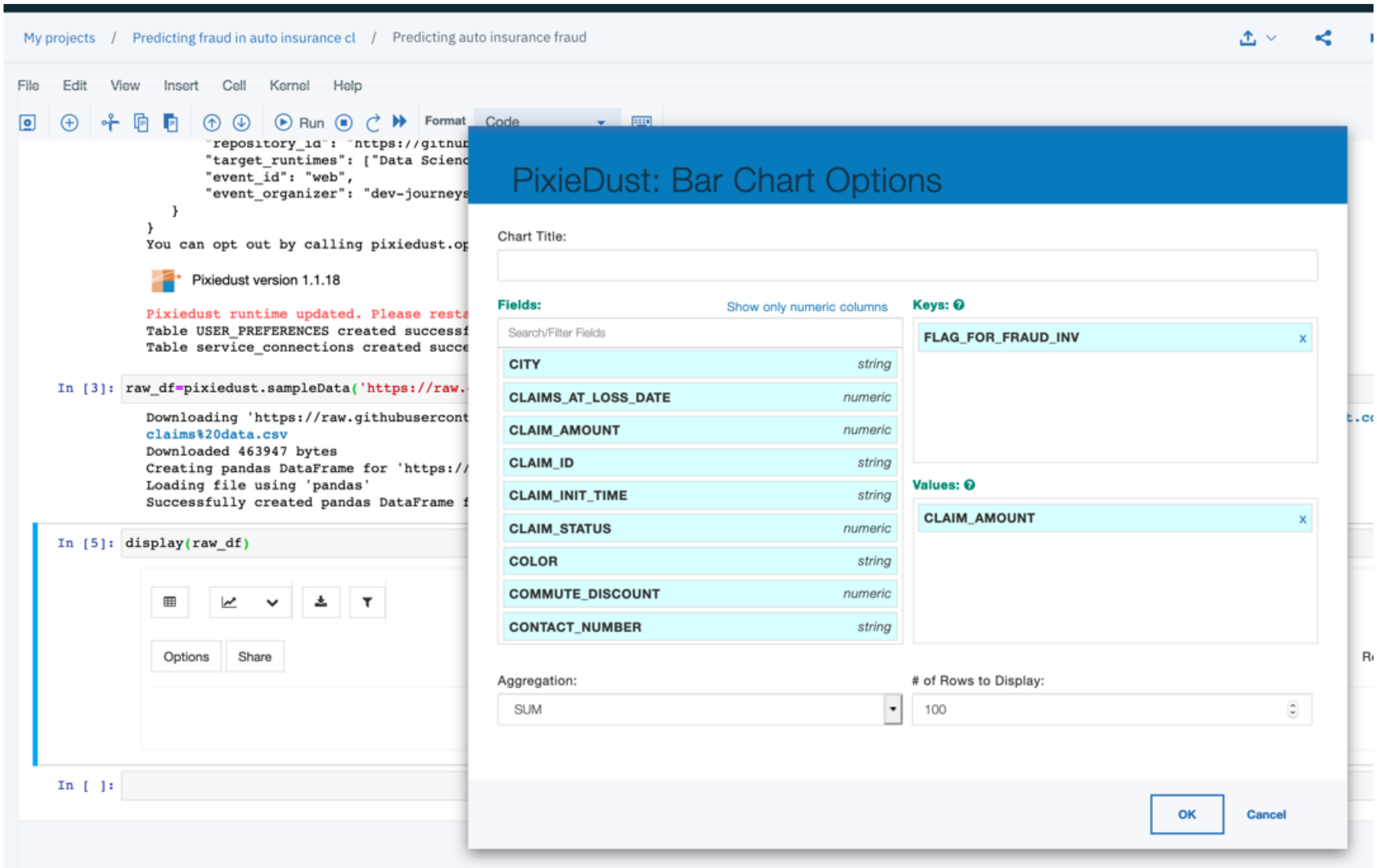
1. From the drop-down menu, select Bar Chart.



```
In [5]: display(raw_df)
```

HOUSEHOLD_ID	DRIVER_ID	POLICY_ID	CLAIM_ID	INCIDENT_C
HH16434	MLI1063	YO8529075	A-2018-GN321	4
JM17168	EVO4040	NY6656317	A-2016-YN454	2
			A-2016-	

2. Find and drag the **FLAG_FOR_FRAUD_INV** to the X axis and the **CLAIM_AMOUNT** to the Y axis and click OK.



My projects / Predicting fraud in auto insurance cl / Predicting auto insurance fraud
File Edit View Insert Cell Kernel Help
Run
Format
Code

```
            "repository_id": "https://githu
            "target_runtimes": ["Data Scienc
            "event_id": "web",
            "event_organizer": "dev-journeys
        }
    }
    You can opt out by calling pixiedust.op

    Pixiedust version 1.1.18

    Pixiedust runtime updated. Please resta
    Table USER_PREFERENCES created successf
    Table service_connections created succe

In [3]: raw_df=pixiedust.sampleData('https://raw.

    Downloading 'https://raw.githubusercont
    claims%20data.csv
    Downloaded 463947 bytes
    Creating pandas DataFrame for 'https://
    Loading file using 'pandas'
    Successfully created pandas DataFrame f

In [5]: display(raw_df)

In [ ]:
```

PixieDust: Bar Chart Options

Chart Title:

Fields: Show only numeric columns

Search/Filter Fields

CITY	string
CLAIMS_AT_LOSS_DATE	numeric
CLAIM_AMOUNT	numeric
CLAIM_ID	string
CLAIM_INIT_TIME	string
CLAIM_STATUS	numeric
COLOR	string
COMMUTE_DISCOUNT	numeric
CONTACT_NUMBER	string

Keys: FLAG_FOR_FRAUD_INV

Values: CLAIM_AMOUNT

Aggregation: SUM

# of Rows to Display: 100

3. Observe the rendering. Change the rendering to brunel. Experiment with other options and renders as well.

4. Perform the visualization tasks with the following dependent and independent variables:

X-axis (keys)	Y-axis (values)
FLAG_FOR_FRAUD_INV	CLAIMS_AT_LOSS_DATE
FLAG_FOR_FRAUD_INV	ODOMETER_AT_LOSS
FLAG_FOR_FRAUD_INV	POLICE_REPORT

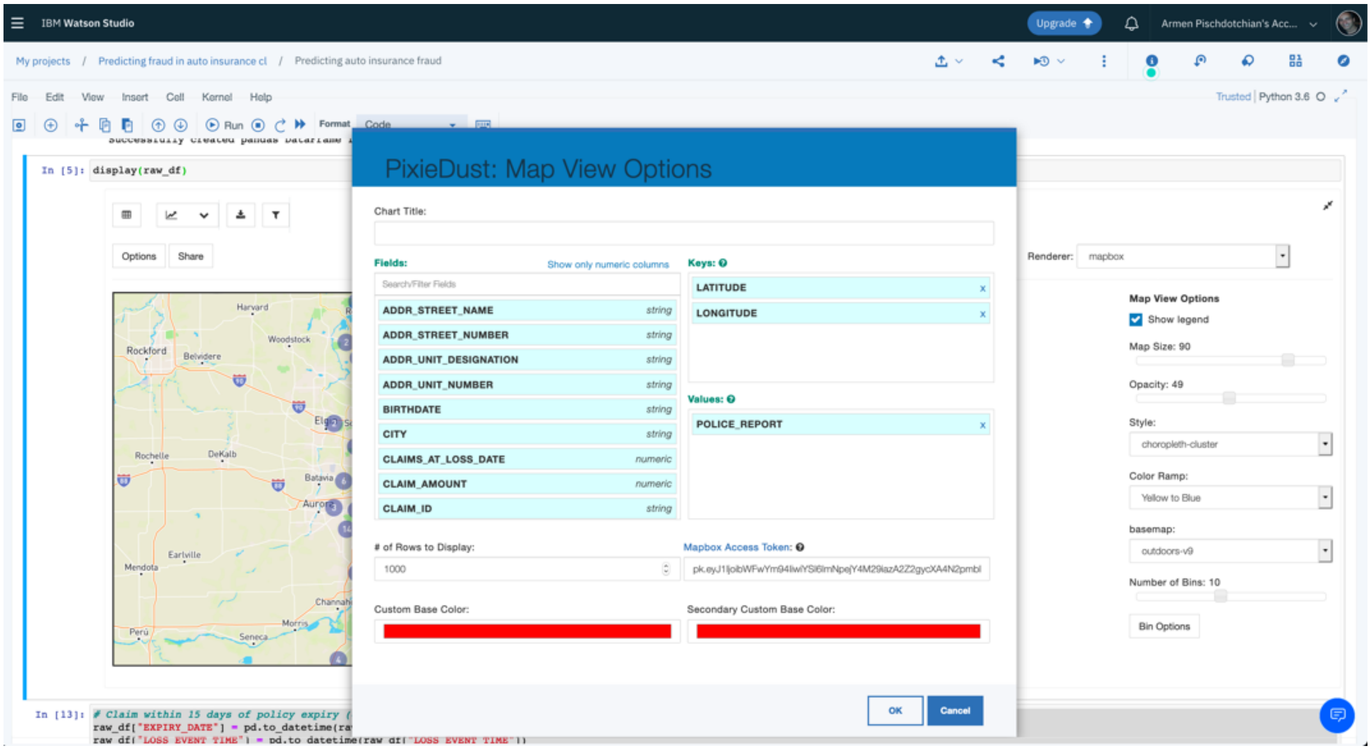
5. Let’s view the presence of police reports on the map.

6. Click **Options** and drag the **LATTITUDE** and **LONGITUDE** to Keys section and the **POLICE_REPORT** to the Values section.

7. If the mapbox API key does not appear, use this key (or you may need to create your own API by following the links provided).

```
pk.eyJ1IjoiYXBpc2NoZG8iLCJhIjoiY2o2cXkxMjUxMDMyaTJ3bGEyajFsZ3Y4cSJ9.mL2PT0XH2vrNiDozb7gO0w
```

Change some of the setting to the right. You can use the settings that you see in the screen capture or your own preferences.



```
In [5]: display(raw_df)
```

PixieDust: Map View Options

Chart Title:

Fields: Show only numeric columns

ADDR_STREET_NAME	string
ADDR_STREET_NUMBER	string
ADDR_UNIT_DESIGNATION	string
ADDR_UNIT_NUMBER	string
BIRTHDATE	string
CITY	string
CLAIMS_AT_LOSS_DATE	numeric
CLAIM_AMOUNT	numeric
CLAIM_ID	string

Keys: LATITUDE, LONGITUDE

Values: POLICE_REPORT

# of Rows to Display: 1000

Mapbox Access Token: pk.eyJ1IjoibWFwYm94IiwiYSI6ImNpejY4M29iazA2Z2gycXA4N2pmbl

Custom Base Color: 🟥

Secondary Custom Base Color: 🟥

Renderer: mapbox

Map View Options

☑ Show legend

Map Size: 90

Opacity: 49

Style: choropleth-cluster

Color Ramp: Yellow to Blue

basemap: outdoors-v9

Number of Bins: 10

```
In [13]: # Claim within 15 days of policy expiry (
raw_df["EXPIRY_DATE"] = pd.to_datetime(ra
raw_df["LOSS_EVENT_TIME"] = pd.to_datetime(raw_df["LOSS_EVENT_TIME"])
```

Notice that the closer the incident was to downtown, the greater the police reports. Perhaps you can think of ways to close the gap between plenty of accidents, yet sparse police reports in rural areas.

What other variables would you peg against the locations of incidents?

Feature Engineering

Feature engineering goes beyond just mere calculations of values in a column; it involves creating new columns that hold ‘engineered’ values that begin to tell the tale of potential fraudulent behavior.
We are going to look at the following hypothesis:

- Claim within 15 days of policy expiry (date of loss - insurance_policy.expiry)
- Expired drivers’ license (if date of loss > insurance_driver.drivers_license_expiry)
- Days living at current address (date of loss - insurance_driver.date_at_current_address)
- Conflict on whether a policyholder with a low mileage discount experienced a loss with high mileage at the point of loss.

Let’s begin our journey with the following commands that you will enter into each cell as depicted below.

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

```
    # Claim within 15 days of policy expiry (date of loss - insurance_policy.expiry)
    raw_df["EXPIRY_DATE"] = pd.to_datetime(raw_df["EXPIRY_DATE"])
    raw_df["LOSS_EVENT_TIME"] = pd.to_datetime(raw_df["LOSS_EVENT_TIME"])

    raw_df["DAYS_FROM_LOSS"] = raw_df["LOSS_EVENT_TIME"] - raw_df["EXPIRY_DATE"]
    raw_df["DAYS_FROM_LOSS"] = abs(raw_df.DAYS_FROM_LOSS.dt.days)

    raw_df.loc[raw_df['DAYS_FROM_LOSS'] >= 15, 'SUSPICIOUS_CLAIM_TIME'] = 1
    raw_df.loc[raw_df['DAYS_FROM_LOSS'] < 15, 'SUSPICIOUS_CLAIM_TIME'] = 0
```

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

```
raw_df["SUSPICIOUS_CLAIM_TIME"].value_counts()
```

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

```
    # Expired drivers license (if date of loss > insurance_driver.drivers_license_expiry)
    raw_df["DRIVERS_LICENSE_EXPIRY"] = pd.to_datetime(raw_df["DRIVERS_LICENSE_EXPIRY"])

    raw_df["DAYS_FROM_L_EXPIRY"] = raw_df["DRIVERS_LICENSE_EXPIRY"] -
    raw_df["LOSS_EVENT_TIME"]
    raw_df["DAYS_FROM_L_EXPIRY"] = raw_df.DAYS_FROM_L_EXPIRY.dt.days

    raw_df.loc[raw_df['DAYS_FROM_L_EXPIRY'] >= 0, 'EXPIRED_LICENSE'] = 0
    raw_df.loc[raw_df['DAYS_FROM_L_EXPIRY'] < 0, 'EXPIRED_LICENSE'] = 1
```

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

```
    # Days living at current address (date of loss -
    insurance_driver.date_at_current_address)
```

```
raw_df["DATE_AT_CURRENT_ADDRESS"] = pd.to_datetime(raw_df["DATE_AT_CURRENT_ADDRESS"])
raw_df["DAYS_AT_ADDRESS"] = raw_df["LOSS_EVENT_TIME"] - raw_df["DATE_AT_CURRENT_ADDRESS"]
raw_df["DAYS_AT_ADDRESS"] = abs(raw_df.DAYS_AT_ADDRESS.dt.days)

raw_df.loc[raw_df['DAYS_AT_ADDRESS'] >= 15, 'SUSPICIOUS_LIVING'] = 1
raw_df.loc[raw_df['DAYS_AT_ADDRESS'] < 15, 'SUSPICIOUS_LIVING'] = 0
```

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

```
raw_df["SUSPICIOUS_LIVING"].value_counts()
```

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

```
#7500/year
raw_df["START_DATE"] = pd.to_datetime(raw_df["START_DATE"])
#find number of days between policy creation and accident
raw_df["LENGTH_OF_POLICY"]=(raw_df["LOSS_EVENT_TIME"] - raw_df["START_DATE"]).dt.days

#convert to years
raw_df["LENGTH_OF_POLICY"]=raw_df["LENGTH_OF_POLICY"]/365

#divide Odometer at loss by years
raw_df["MILES/YEAR"] = raw_df["ODOMETER_AT_LOSS"]/raw_df["LENGTH_OF_POLICY"]
raw_df["MILES/YEAR"].value_counts()
```

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

```
# Conflict on whether a policyholder with a low mileage discount experienced a loss with
high mileage at the point of loss
raw_df.loc[raw_df["MILES/YEAR"] <7500, 'LOW_MILEAGE_AT_LOSS'] = 1
raw_df.loc[raw_df["MILES/YEAR"] >=7500, 'LOW_MILEAGE_AT_LOSS'] = 0
```

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

```
raw_df.loc[raw_df["LOW_MILEAGE_USE"]==raw_df["LOW_MILEAGE_AT_LOSS"],
'SUSPICIOUS_MILEAGE'] = 0
raw_df.loc[raw_df["LOW_MILEAGE_USE"]!=raw_df["LOW_MILEAGE_AT_LOSS"],
'SUSPICIOUS_MILEAGE'] = 1
```

9. 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
```

10. 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']
```

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

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

12. 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)
```

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

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

14. 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","EXCESSIVE_CLAIM_AMOUNT"]]
yVar = df_model["FLAG_FOR_FRAUD_INV"]
```

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

```
xVar.head()
```

16. 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)
```

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

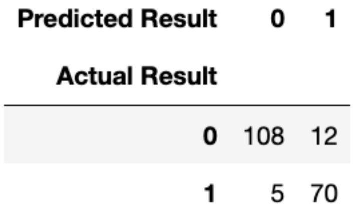
```
#train model
clf = RandomForestClassifier(n_jobs=2, random_state=0)
```

```
clf.fit(X_train, y_train)
```

18. 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

X axis is actual value
Y axis is what is predicted

108 is TP (type 1 errors)
5 is FN, (type 2 errors)
70 TN and
12 is FP