IBM Developer
SKILLS NETWORK

SPACEX

Winning Space Race with Data Science

Sourav Das
06-03-2022

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion
- Appendix

Executive Summary

- **Summary of methodologies**
  - ➢ Data Collection
  - ➢ Data Wrangling
  - ➢ EDA with Data Visualization
  - ➢ EDA with SQL
  - ➢ Building an interactive map with Folium
  - ➢ Predictive Analysis (Classification)
- **Summary of all results**
  - ➢ Exploratory data analysis result
  - ➢ Interactive analysis demo using screenshots
  - ➢ Predictive Analysis Results

Introduction

- **Project background and context**

  We will predict if the Falcon 9 first stage will land successfully. SpaceX advertises Falcon 9 rocket launches on its website, with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. Therefore if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch. In this module, you will be provided with an overview of the problem and the tools you need to complete the course.

- **Problems that needed solving**

  - What influences if the rocket will land successfully ?
  - The effect each relationship with certain rocket variables will impact in determining the success rate of a successful landing .
  - What conditions do SpaceX have to achieve to get the best results and ensure the best rocket success landing rate .

IBM Developer
SKILLS NETWORK

Section 1

Methodology

SPACEX

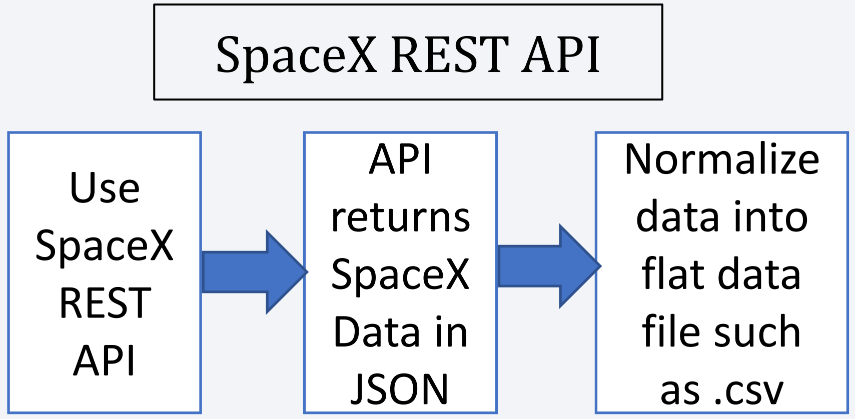
Methodology

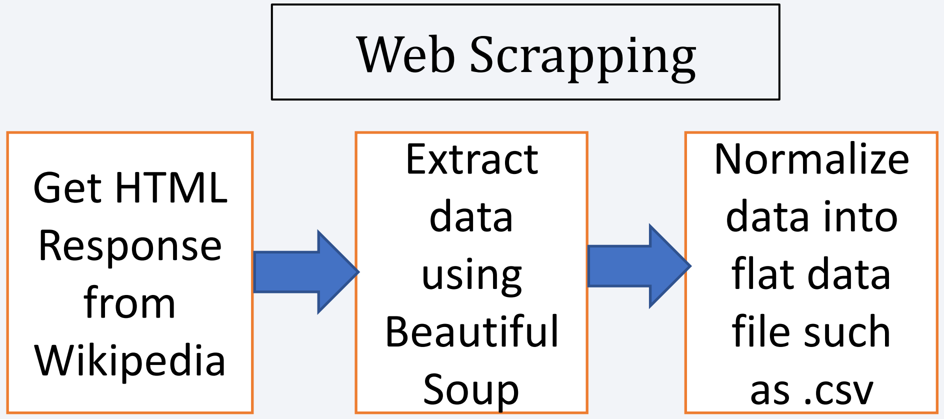
- **Data collection methodology:**
  - ➢ SpaceX Rest API
  - ➢ Web Scrapping from Wikipedia
- **Perform data wrangling**
  - ➢ One Hot Encoding data fields for Machine Learning and dropping irrelevant columns
- **Perform exploratory data analysis (EDA) using visualization and SQL**
  - ➢ Plotting Scatter Graphs, Bar Graphs to show relationship between variables and patterns of data
- **Perform interactive visual analytics using Folium and Plotly Dash**
- **Perform predictive analysis using classification models**
  - ➢ How to build, tune, evaluate classification models

Data Collection

Collection of Datasets was done by following

- Worked with SpaceX launch data that is gathered from the SpaceX Rest API.
- The API gave the data about launches, including information about the rocket used, payload delivered, launch specifications and landing outcome.
- The goal is to use this data to predict whether SpaceX will attempt to land a rocket or not.
- The SpaceX Rest API endpoints or URL starts with api.spacexdata.com/v4/.
- Another popular data source for obtaining Falcon 9 launch data is web scrapping Wikipedia using BeautifulSoup.


SpaceX REST API
Use SpaceX REST API
API returns SpaceX Data in JSON
Normalize data into flat data file such as .csv


Web Scrapping
Get HTML Response from Wikipedia
Extract data using Beautiful Soup
Normalize data into flat data file such as .csv

Data Collection – SpaceX API

IBM Developer SKILLS NETWORK

1. Getting response from API

```
[ ]  spacex_url="https://api.spacexdata.com/v4/launches/past"

[ ]  response = requests.get(spacex_url)
```

2. Converting response to a JSON File

SPACEX

```
# Use json_normalize meethod to convert the json result into a dataframe
response = requests.get(static_json_url).json()
data = pd.json_normalize(response)
```

3. Apply custom functions to clean data

```
# Call getBoosterVersion
getBoosterVersion(data)
```

```
# Call getLaunchSite
getLaunchSite(data)

# Call getPayloadData
getPayloadData(data)

# Call getCoreData
getCoreData(data)
```

4. Assign list to Dictionary then Data Frame

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

```
# Create a data from launch_dict
df = pd.DataFrame.from_dict(launch_dict)
```

GitHub

5. Filter Data Frame and export to flat file (.csv)

```
data_falcon9 = df.loc[df['BoosterVersion']!="Falcon 1"]
```

```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data Collection - Scraping

1. Getting response from HTML

```
response = requests.get(static_url)
```

2. Creating BeautifulSoup object

```
soup = BeautifulSoup(response.text, 'html.parser')
```

3. Finding Tables

```
html_tables = soup.find_all("table")
```

4. Getting Column Names

```
column_names = []
temp = soup.find_all('th')
for x in range(len(temp)):
    try:
     name = extract_column_from_header(temp[x])
     if (name is not None and len(name) > 0):
        column_names.append(name)
    except:
     pass
```

5. Creation of Dictionary

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

6. Appending Data to Keys (refer to notebook Block 16)

```
In [16]:  extracted_row = 0
          #Extract each table
          for table_number,table in enumerate(soup.find_all
             # get table row
              for rows in table.find_all("tr"):
                  #check to see if first table heading is a
                  if rows.th:
                      if rows.th.string:
```

7. Converting Dictionary to Data Frame

```
df = pd.DataFrame.from_dict(launch_dict)
```

8. Export to flat file (.csv)

```
df.to_csv('spacex_web_scraped.csv', index=False)
```

Data Wrangling

IBM Developer SKILLS NETWORK

SPACEX

Introduction

In the data set, there are several different cases where the booster did not land successfully. Sometimes a landing was attempted but failed due to an accident; for example, True Ocean means the mission outcome was successfully landed to a specific region of the ocean while False Ocean means the mission outcome was unsuccessfully landed to a specific region of the ocean. True RTLS means the mission outcome was successfully landed to a ground pad False RTLS means the mission outcome was unsuccessfully landed to a ground pad. True ASDS means the mission outcome was successfully landed on a drone ship False ASDS means the mission outcome was unsuccessfully landed on a drone ship.
We mainly convert those outcomes into Training Labels with 1 means the booster successfully landed 0 means it was unsuccessful.

Process

Perform Exploratory Data Analysis EDA on dataset

Calculate the number of launches at each site

Calculate the number and occurrence of each orbit

Calculate the number and occurrence of mission outcome per orbit type

Export dataset as .CSV

Create a landing outcome label from Outcome column

Work out success rate for every landing in dataset

GitHub

Each launch aims to an dedicated orbit, and here are some common orbit types:

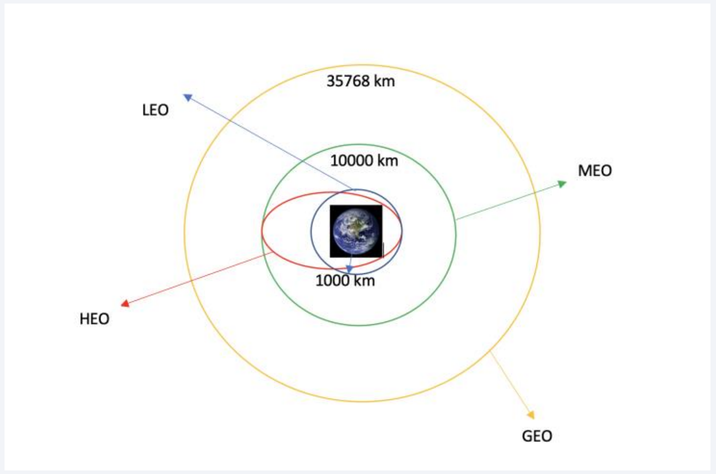


Diagram showing common orbit types SpaceX uses

EDA with Data Visualization

Scatter Graphs being drawn

- Flight Number vs. Payload Mass
- Flight Number vs. Launch Site
- Payload vs. Launch Site
- Orbit vs. Flight Number
- Payload vs. Orbit Type
- Orbit vs. Payload Mass

Scatter plots show how much one variable is affected by another. The relationship between two variables are called their correlation. Scatter plots usually consists of large body of data.

Bar Graph being drawn

Mean vs. Orbit

A bar diagram makes it easy to compare sets of data between different groups at a glance. The graph represents categories on one axis and a distance value in the order. The goal is to show the relationship between the two axes. Bar charts can also show big changes in data over time.

Line Graph being drawn

Success Rate vs. Year

Line graphs are useful in that they show data variables and trends very clearly and can help to make predictions about the results of data not yet recorded.

EDA with SQL

Performed SQL queries to gather information about the dataset

For example of some questions we were asked about the data we needed information about. Which we are using SQL queries to get the answers in the dataset:

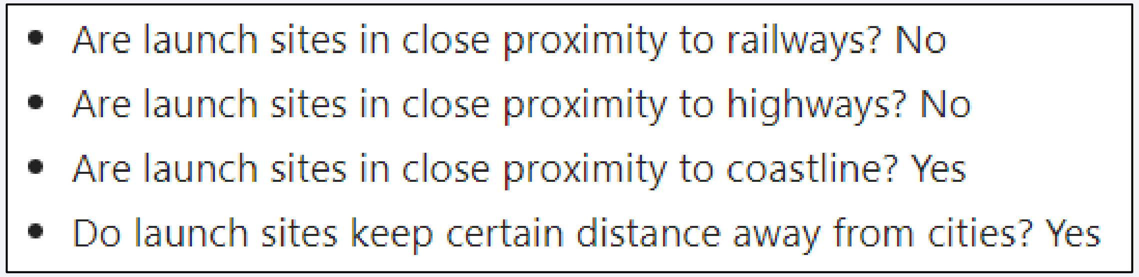
- Displaying the names of the unique launch sites in the space mission
- Displaying 5 records where launch sites begin with the string ‘KSC’
- Displaying the total payload mass carried by boosters launched by NASA (CRS)
- Displaying average payload mass carried by booster version F9 v1.1
- Listing the date where the successful landing outcome in drone ship was achieved.
- Listing the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000
- Listing the total number of successful and failure mission outcomes
- Listing the names of the booster_versions which have carried the maximum payload mass.
- Listing the records which will display the month names, successful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017
- Ranking the count of successful landing_outcomes between the date 2010-06-04 and 2017-03-20 in descending order.

Build an Interactive Map with Folium

To visualize the Launch Data into an interactive map. We took the Latitude and Longitude Coordinates at each launch site and added a Circle Marker around each launch site with a label of the name of the launch site.

We assigned the dataframe launch_outcomes(failures, successes) to classes 0 and 1 with Green and Red markers on the map in a MarkerCluster()

Using Haversine's formula we calculated the distance from the Launch Site to various landmarks to find various trends about what is around the Launch Site to measure patterns. **Lines are drawn** on the map to measure distance to landmarks.



- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

GitHub

Build a Dashboard with Plotly Dash

The dashboard is built with Flask and Dash web framework.

Graphs:

- Pie Chart showing the total launches by a certain site/all sites
- Display relative proportions of multiple classes of data.
- Size of the circle can be made proportional to the total quantity it represents.

Scatter Graph showing the relationship with Outcome and Payload Mass (Kg) for the different Booster Versions :

- It shows the relationship between two variables.
- It is the best method to show you a non-linear pattern.
- The range of data flow, i.e. maximum and minimum value, can be determined.
- Observation and reading are straightforward.

Predictive Analysis (Classification)

1. BUILDING MODEL

- Load our dataset into NumPy and Pandas Transform Data
- Split our data into training and test data sets
- Check how many test samples we have
- Decide which type of machine learning algorithms we want to use
- Set our parameters and algorithms to GridSearchCV
- Fit our datasets into the GridSearchCV objects and train our dataset.

2. EVALUATING MODEL

- Check accuracy for each model
- Get tuned hyperparameters for each type of algorithms
- Plot Confusion Matrix

3. IMPROVING MODEL

- Feature Engineering
- Algorithm Tuning

4. FINDING THE BEST PERFORMING CLASSIFICATION MODEL

- The model with the best accuracy score wins the best performing model
- In the notebook there is a dictionary of algorithms with scores at the bottom of the notebook.

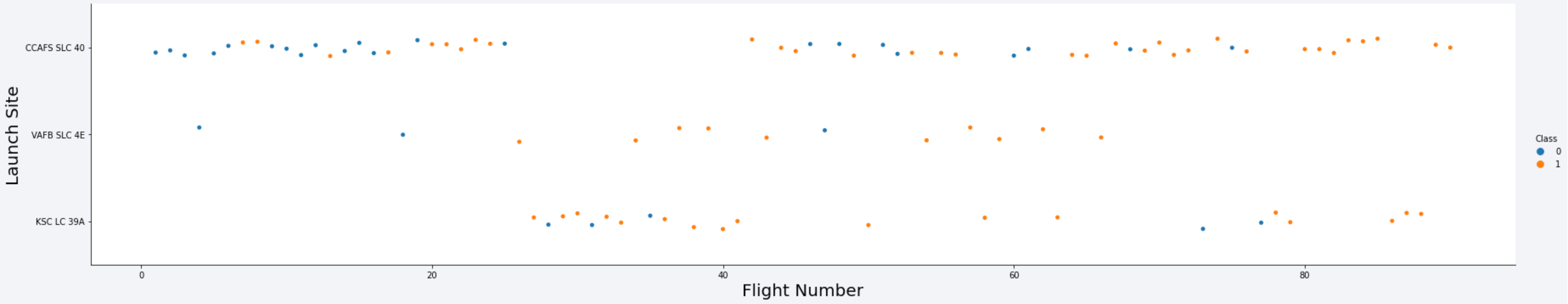
Results

- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

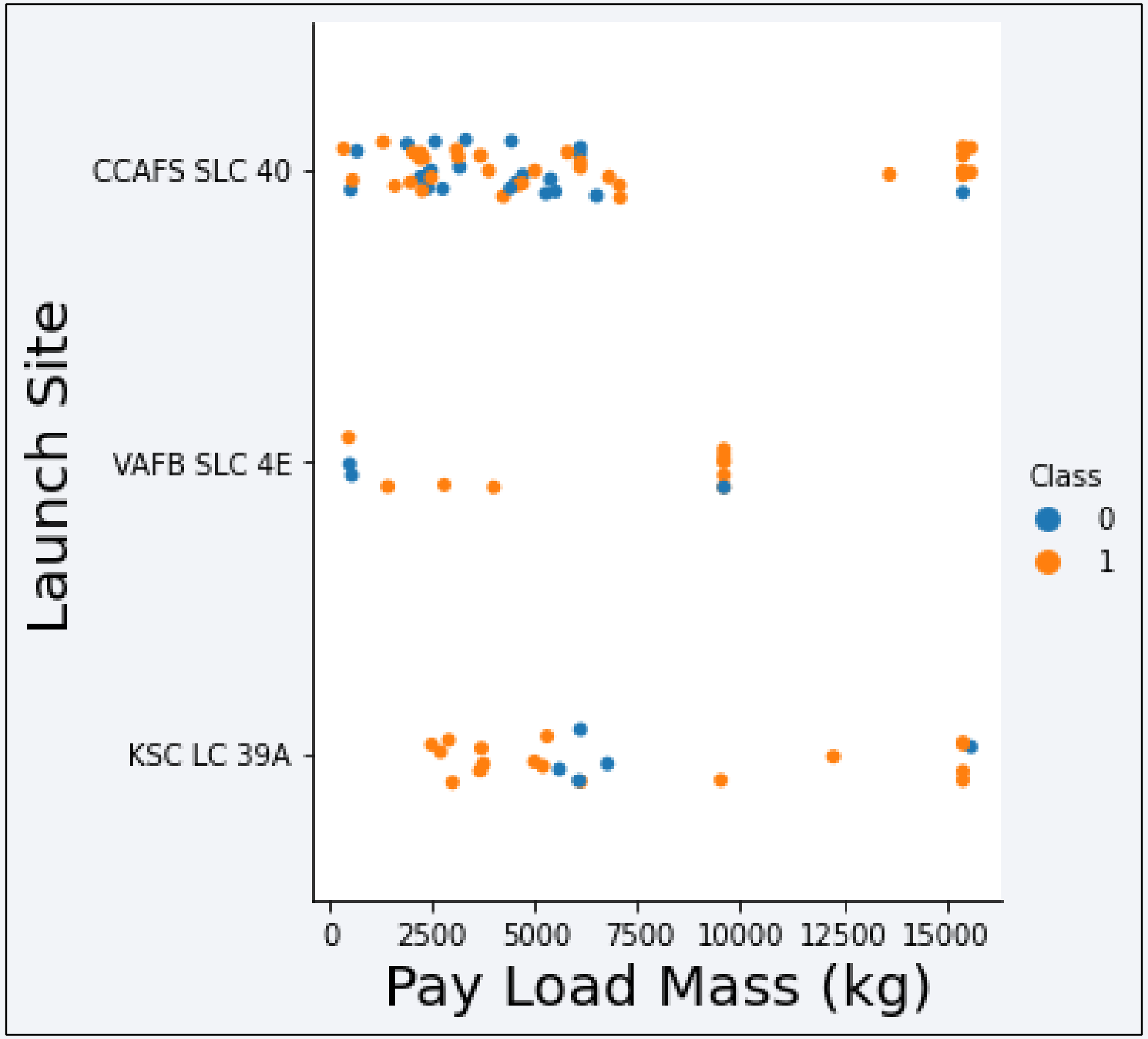
Insights drawn from EDA

Flight Number vs. Launch Site



The more amount of flights at a launch site the greater the success rate at a launch site

Payload vs. Launch Site

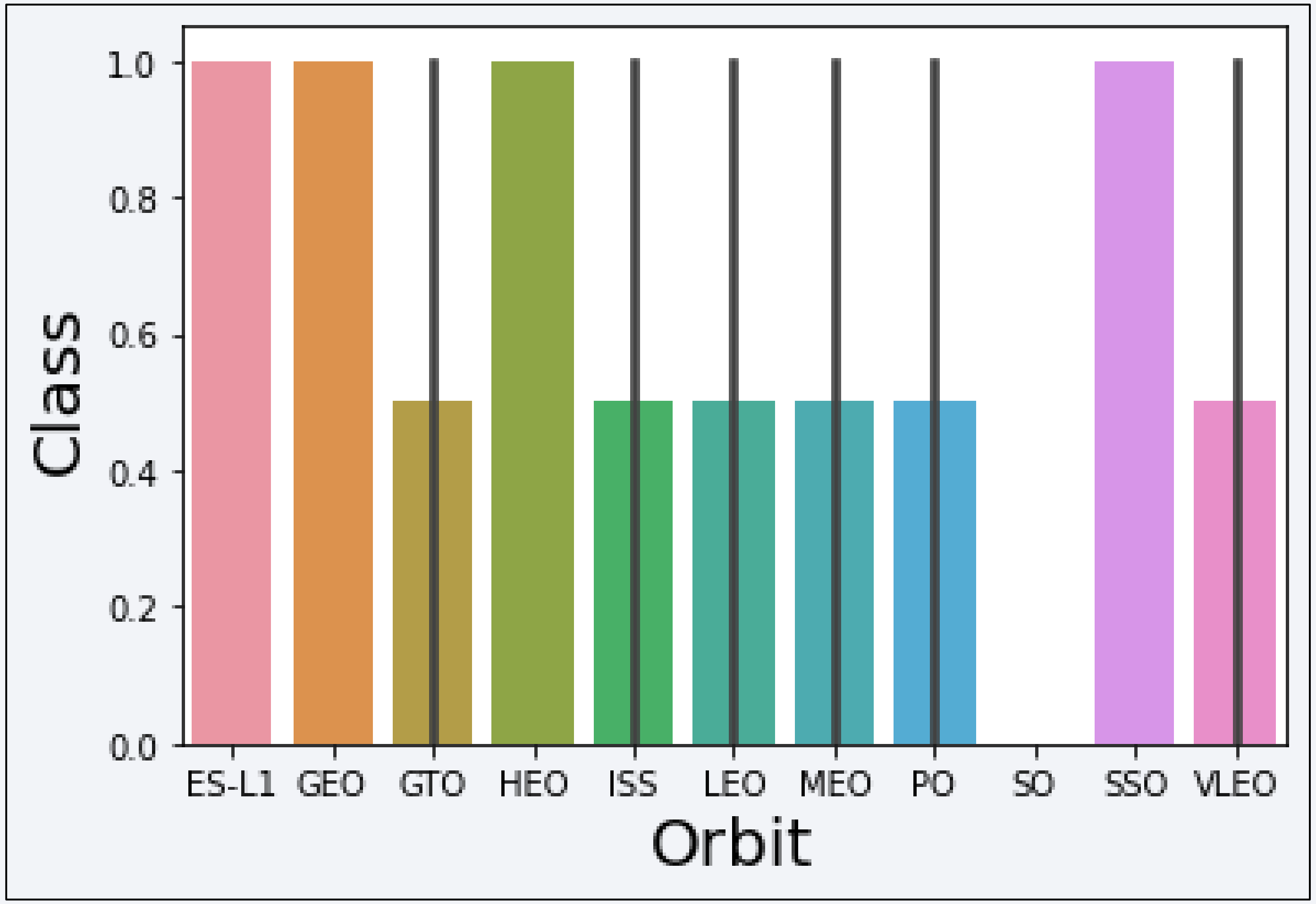


The greater the payload mass for Launch Site CCAFS SLC 40 the higher the success rate for the Rocket. There is not quite a clear pattern to be found using this visualization to make a decision if the Launch Site is dependent on Pay Load Mass for a success launch.

Success Rate vs. Orbit Type

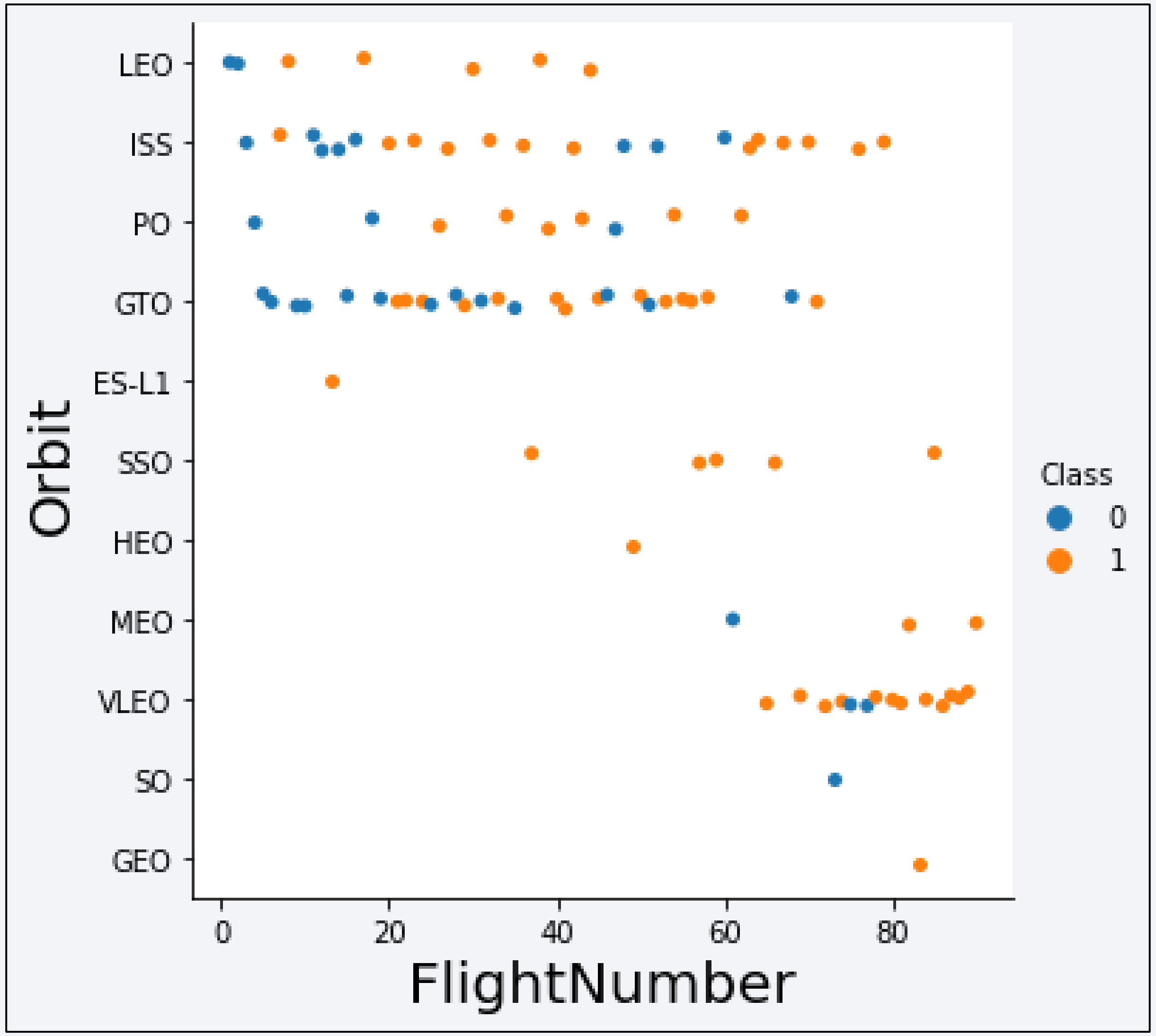
IBM Developer SKILLS NETWORK

SPACEX



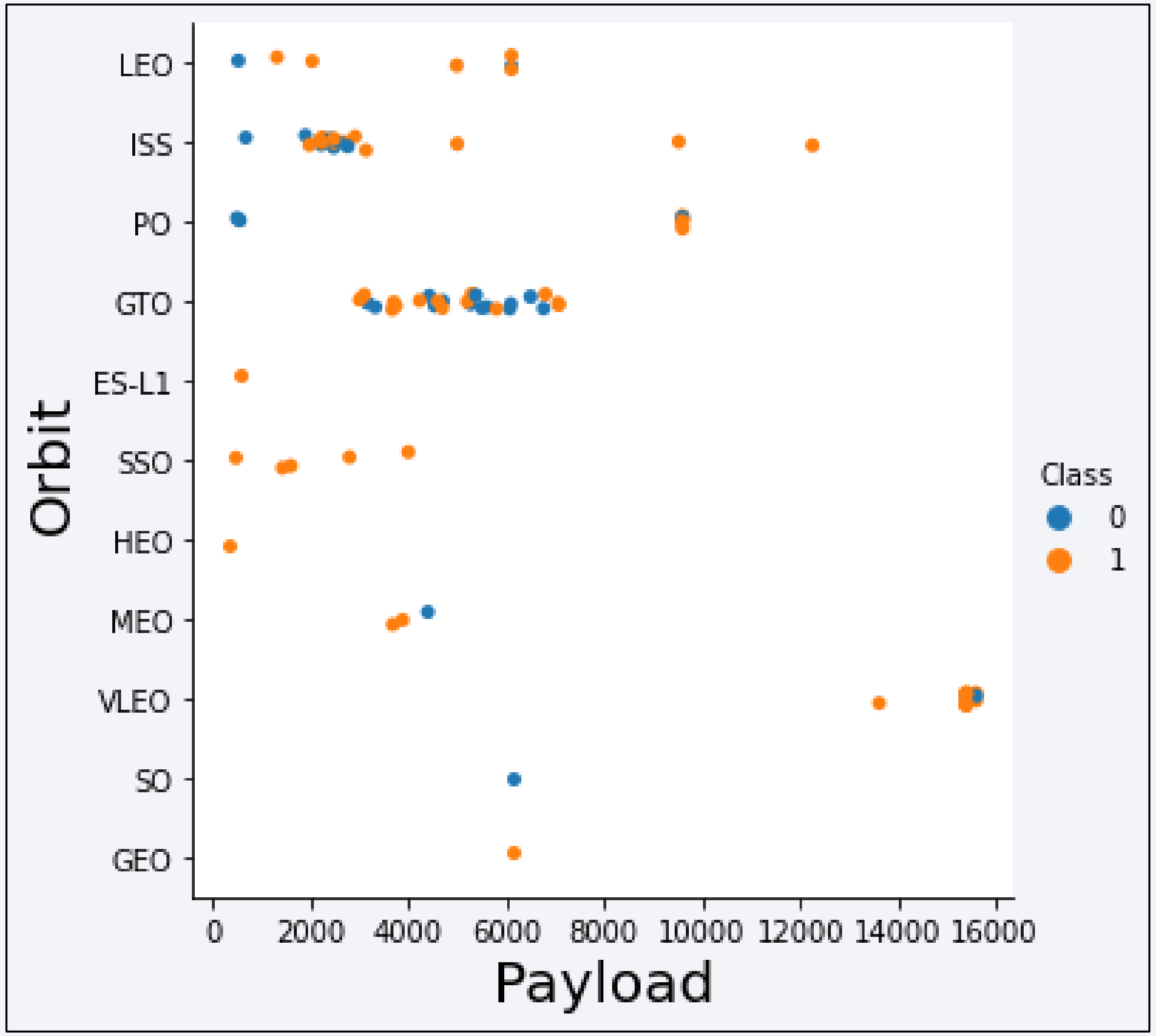
Orbit ES-L1, GEO, HEO and SSO has the highest success rate and orbit SO has the lowest success rate.

Flight Number vs. Orbit Type



In the LEO orbit the success related to the number of flights. On the other hand, there seems to be no relationship between flight number when in GTO orbit.

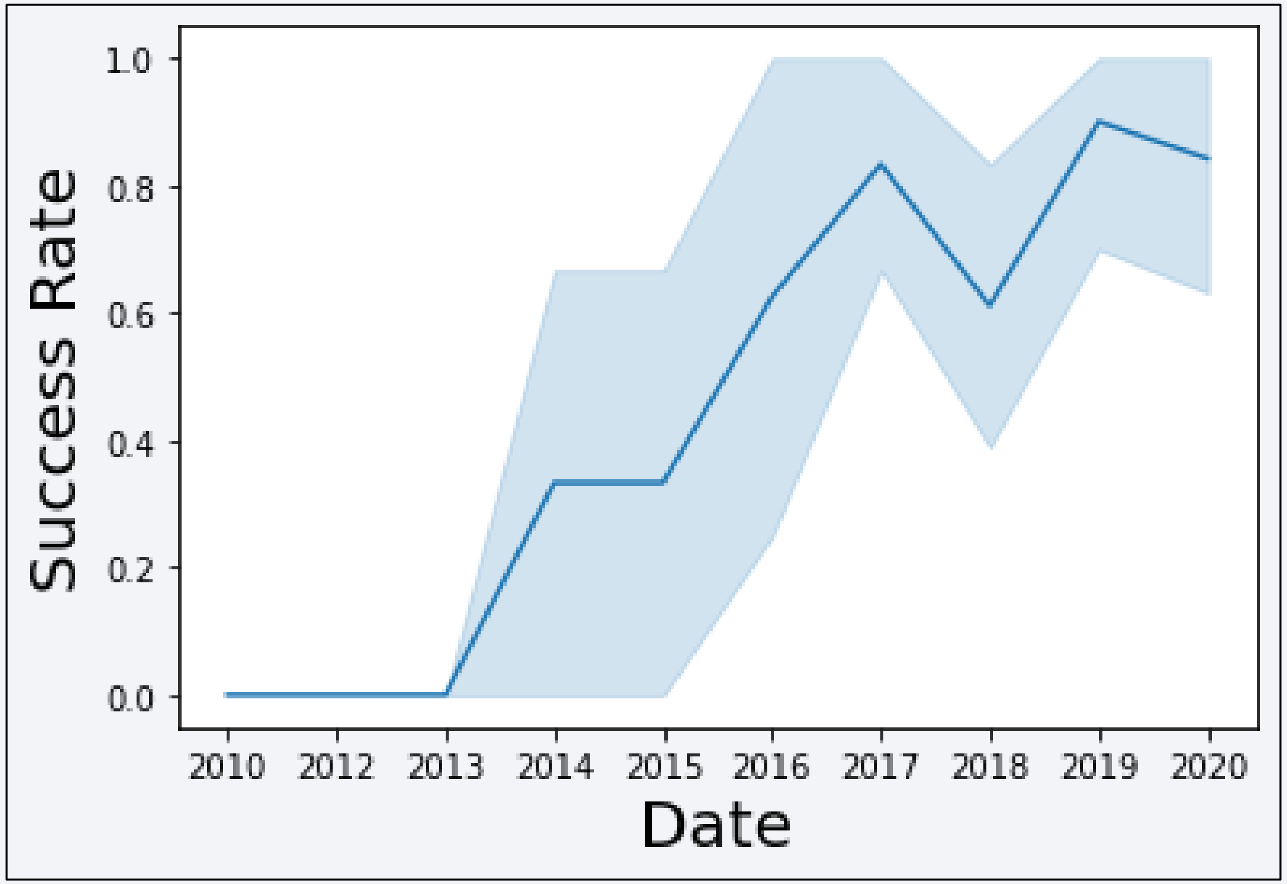
Payload vs. Orbit Type



With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS.

However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.

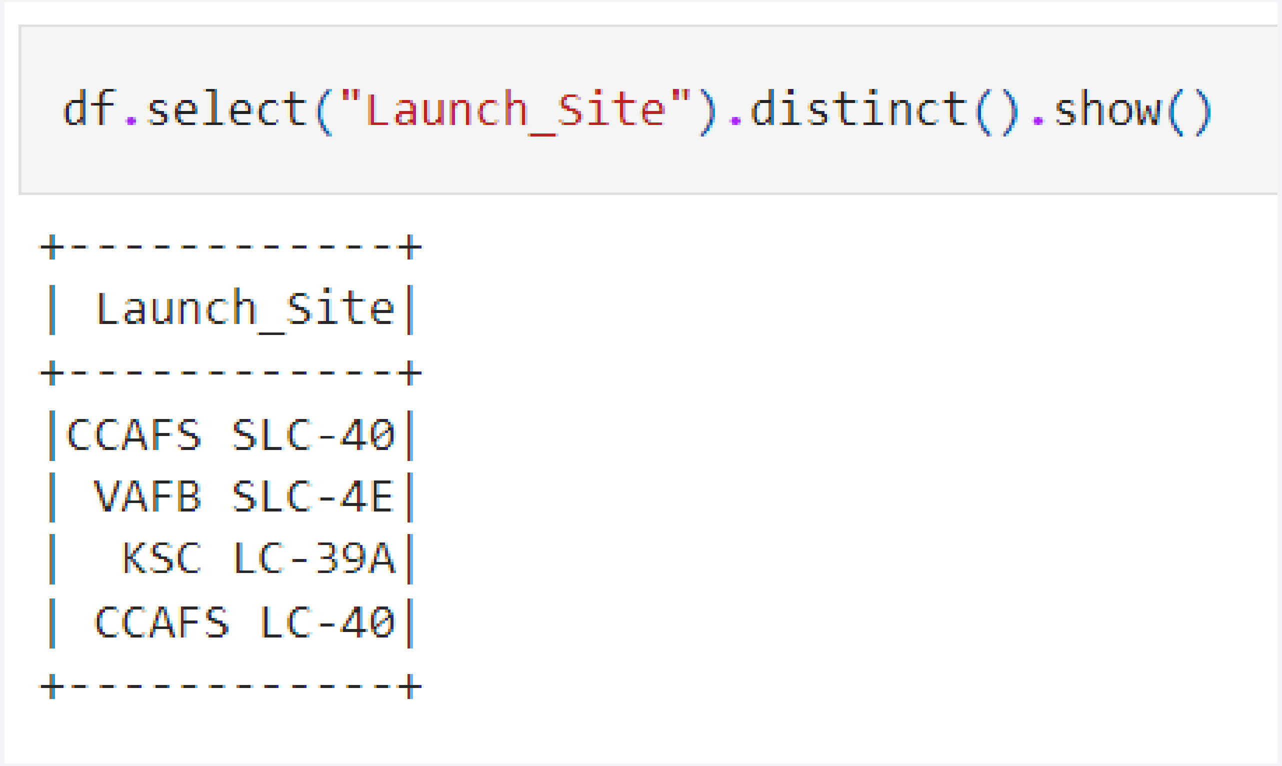
Launch Success Yearly Trend



It can be observed that the success rate since 2013 kept increasing till 2020.

All Launch Site Names

SQL Query



```
df.select("Launch_Site").distinct().show()
```

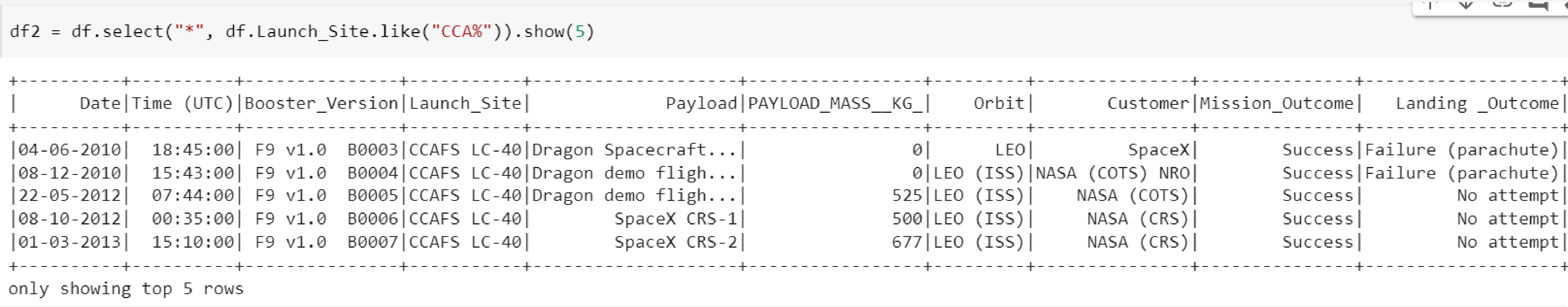
```
+-------------+
|  Launch_Site|
+-------------+
|CCAFS SLC-40|
| VAFB SLC-4E|
|  KSC LC-39A|
| CCAFS LC-40|
+-------------+
```

Query Explanation:

Using distinct() function in the query means that it will only show unique values in the Launch_Site

Launch Site Names Begin with 'CCA'

SQL Query



```
df2 = df.select("*", df.Launch_Site.like("CCA%")).show(5)
```

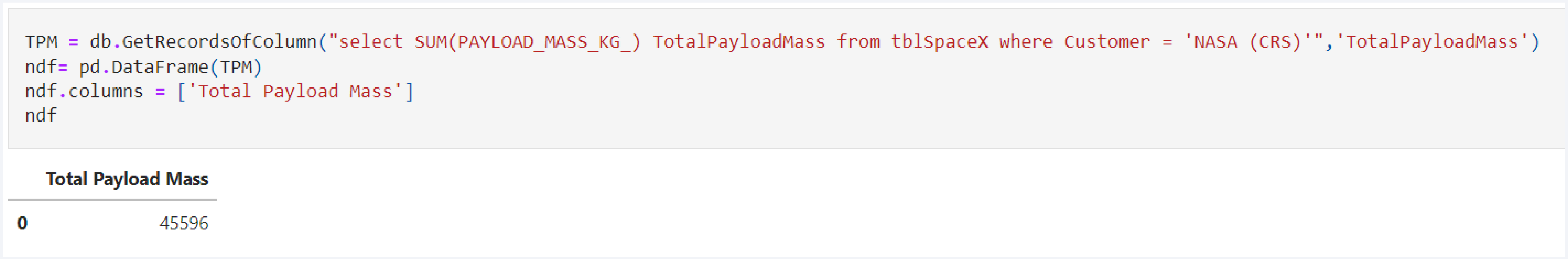
```
+----------+----------+---------------+-----------+--------------------+-----------------+---------+---------------+---------------+-------------------+
|      Date|Time (UTC)|Booster_Version|Launch_Site|             Payload|PAYLOAD_MASS__KG_|    Orbit|       Customer|Mission_Outcome|   Landing _Outcome|
+----------+----------+---------------+-----------+--------------------+-----------------+---------+---------------+---------------+-------------------+
|04-06-2010|  18:45:00| F9 v1.0  B0003|CCAFS LC-40|Dragon Spacecraft...|                0|      LEO|         SpaceX|        Success|Failure (parachute)|
|08-12-2010|  15:43:00| F9 v1.0  B0004|CCAFS LC-40|Dragon demo fligh...|                0|LEO (ISS)|NASA (COTS) NRO|        Success|Failure (parachute)|
|22-05-2012|  07:44:00| F9 v1.0  B0005|CCAFS LC-40|Dragon demo fligh...|              525|LEO (ISS)|    NASA (COTS)|        Success|         No attempt|
|08-10-2012|  00:35:00| F9 v1.0  B0006|CCAFS LC-40|       SpaceX CRS-1|              500|LEO (ISS)|     NASA (CRS)|        Success|         No attempt|
|01-03-2013|  15:10:00| F9 v1.0  B0007|CCAFS LC-40|       SpaceX CRS-2|              677|LEO (ISS)|     NASA (CRS)|        Success|         No attempt|
+----------+----------+---------------+-----------+--------------------+-----------------+---------+---------------+---------------+-------------------+
only showing top 5 rows
```

Query Explanation:

Using the ‘show(5)’ in the query means that it will only show top 5 records and ‘like’ function has a wild card with the words ‘CCA%' the percentage in the end suggests that the Launch Site name must start with CCA.

Total Payload Mass

SQL Query



```
TPM = db.GetRecordsOfColumn("select SUM(PAYLOAD_MASS_KG_) TotalPayloadMass from tblSpaceX where Customer = 'NASA (CRS)'",'TotalPayloadMass')
ndf= pd.DataFrame(TPM)
ndf.columns = ['Total Payload Mass']
ndf
```

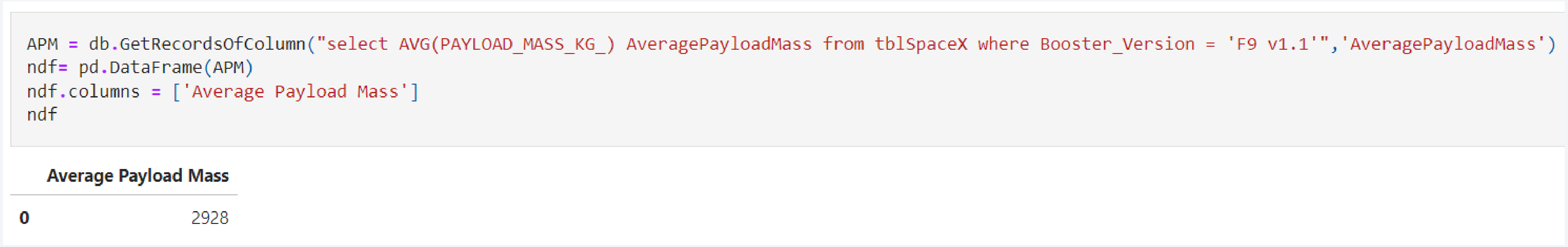
```
   Total Payload Mass
0               45596
```

Query Explanation:

Using the function SUM summates the total in the column PAYLOAD_MASS_KG_.
The WHERE clause filters the dataset to only perform calculations on Customer NASA (CRS).

Average Payload Mass by F9 v1.1

SQL Query



```
APM = db.GetRecordsOfColumn("select AVG(PAYLOAD_MASS_KG_) AveragePayloadMass from tblSpaceX where Booster_Version = 'F9 v1.1'","AveragePayloadMass')
ndf= pd.DataFrame(APM)
ndf.columns = ['Average Payload Mass']
ndf
```

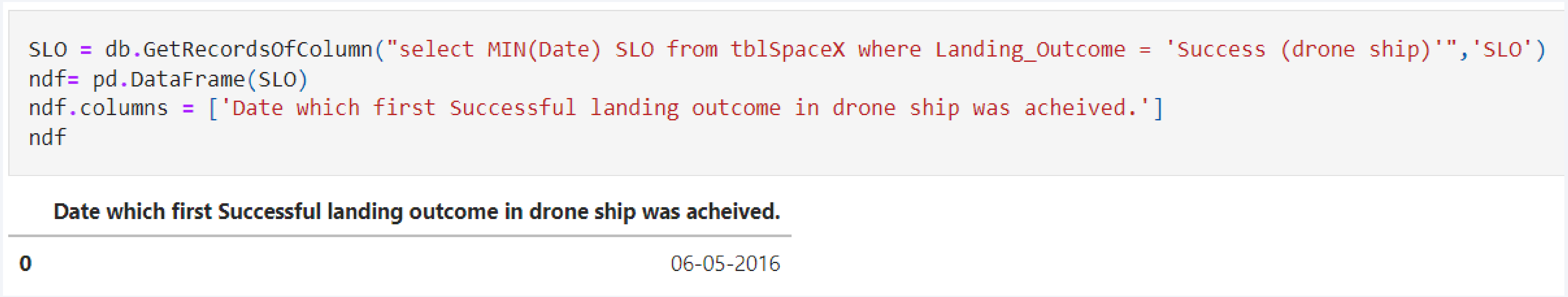
```
   Average Payload Mass
0                  2928
```

Query Explanation:

Using the function AVG works out the average in the column PAYLOAD_MASS_KG_.
The WHERE clause filters the dataset to only perform calculations on Booster_Version F9 v1.1.

First Successful Ground Landing Date

SQL Query



```
SLO = db.GetRecordsOfColumn("select MIN(Date) SLO from tblSpaceX where Landing_Outcome = 'Success (drone ship)'",'SLO')
ndf= pd.DataFrame(SLO)
ndf.columns = ['Date which first Successful landing outcome in drone ship was acheived.']
ndf
```

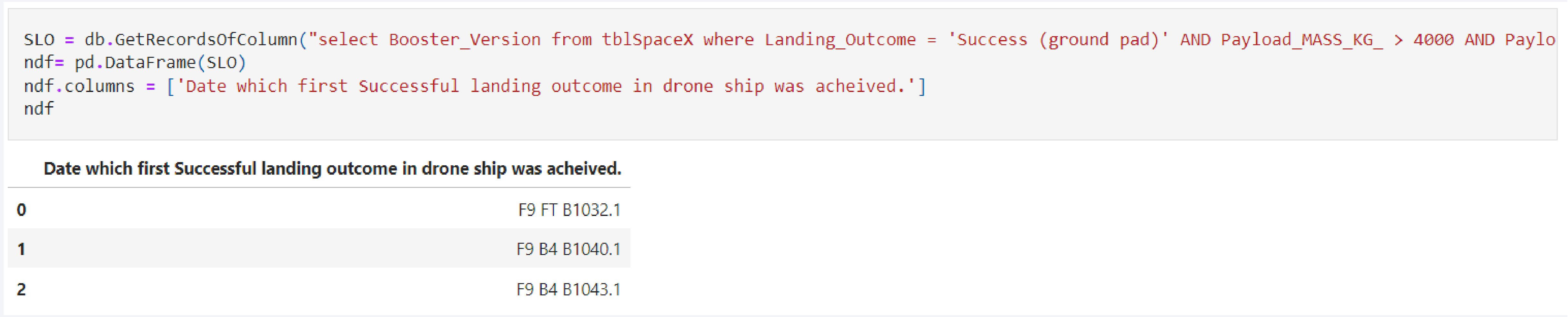
```
   Date which first Successful landing outcome in drone ship was acheived.
0                                                             06-05-2016
```

Query Explanation:

Using the function MIN works out the minimum date in the column Date.
The WHERE clause filters the dataset to only perform calculations on Landing_outcome Success (drone ship).

Successful Drone Ship Landing with Payload between 4000 and 6000

SQL Query



```
SLO = db.GetRecordsOfColumn("select Booster_Version from tblSpaceX where Landing_Outcome = 'Success (ground pad)' AND Payload_MASS_KG_ > 4000 AND Paylo
ndf= pd.DataFrame(SLO)
ndf.columns = ['Date which first Successful landing outcome in drone ship was acheived.']
ndf
```

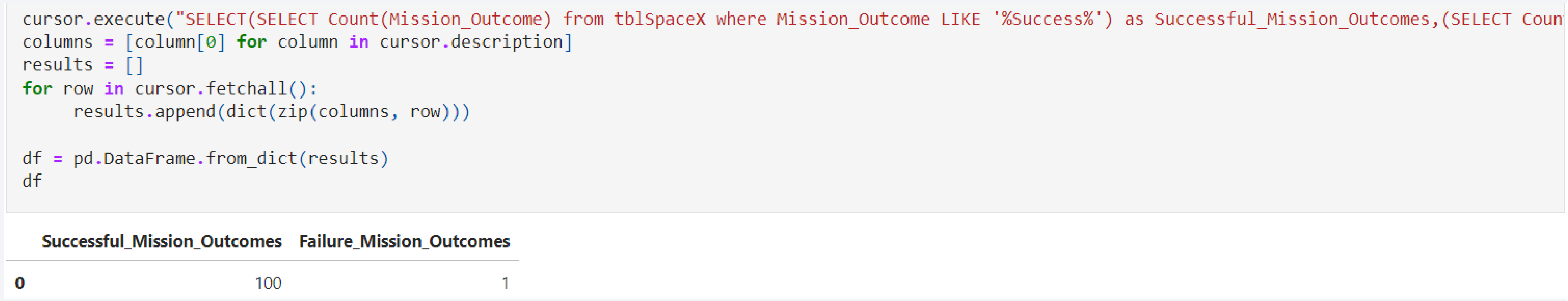
	Date which first Successful landing outcome in drone ship was acheived.
0	F9 FT B1032.1
1	F9 B4 B1040.1
2	F9 B4 B1043.1

Query Explanation:

The WHERE clause filters the dataset to only perform calculations on Landing_outcome Success (drone ship).
The AND clause specifies additional filter conditions Payload_MASS_KG_ > 4000 AND Payload_MASS_KG_ < 6000.

Total Number of Successful and Failure Mission Outcomes

SQL Query



```
cursor.execute("SELECT(SELECT Count(Mission_Outcome) from tblSpaceX where Mission_Outcome LIKE '%Success%') as Successful_Mission_Outcomes,(SELECT Coun
columns = [column[0] for column in cursor.description]
results = []
for row in cursor.fetchall():
    results.append(dict(zip(columns, row)))

df = pd.DataFrame.from_dict(results)
df
```

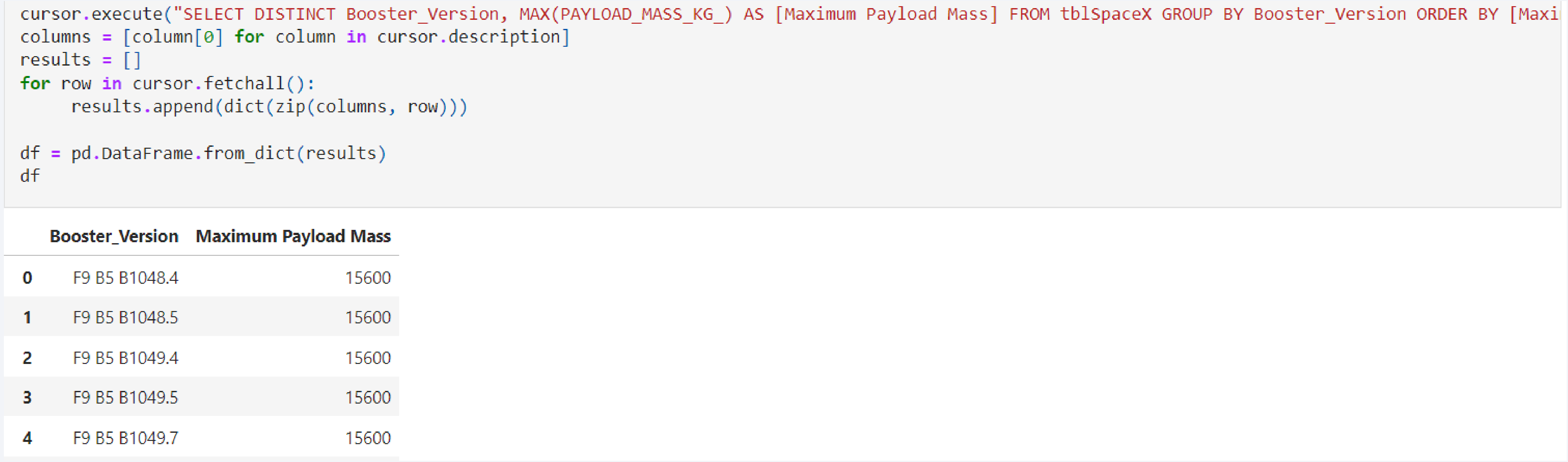
	Successful_Mission_Outcomes	Failure_Mission_Outcomes
0	100	1

Query Explanation:

We used subqueries here to produce the results. The LIKE '%f00%' wildcard shows that in the record the foo phrase is in any part of the string in the records for example.
PHRASE "(Drone Ship was a Success)" LIKE '%Success%' Word 'Success' is in the phrase the filter will include it in the dataset.

Boosters Carried Maximum Payload

SQL Query



```
cursor.execute("SELECT DISTINCT Booster_Version, MAX(PAYLOAD_MASS_KG_) AS [Maximum Payload Mass] FROM tblSpaceX GROUP BY Booster_Version ORDER BY [Maxi
columns = [column[0] for column in cursor.description]
results = []
for row in cursor.fetchall():
    results.append(dict(zip(columns, row)))

df = pd.DataFrame.from_dict(results)
df
```

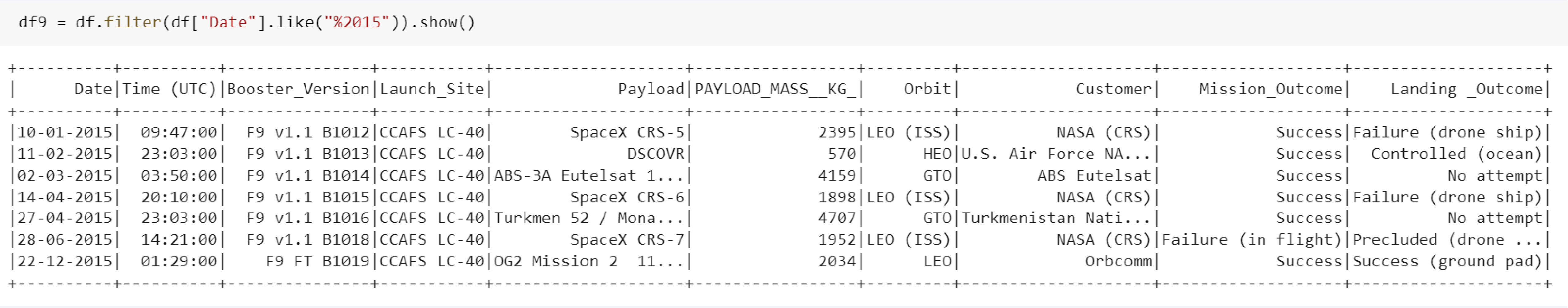
	Booster_Version	Maximum Payload Mass
0	F9 B5 B1048.4	15600
1	F9 B5 B1048.5	15600
2	F9 B5 B1049.4	15600
3	F9 B5 B1049.5	15600
4	F9 B5 B1049.7	15600

Query Explanation:

Using the word DISTINCT in the query means that it will only show Unique values in the Booster Version column. GROUP BY puts the list in order set to a certain condition. DESC means its arranging the dataset into descending order.

2015 Launch Records

SQL Query



```
df9 = df.filter(df["Date"].like("%2015")).show()
```

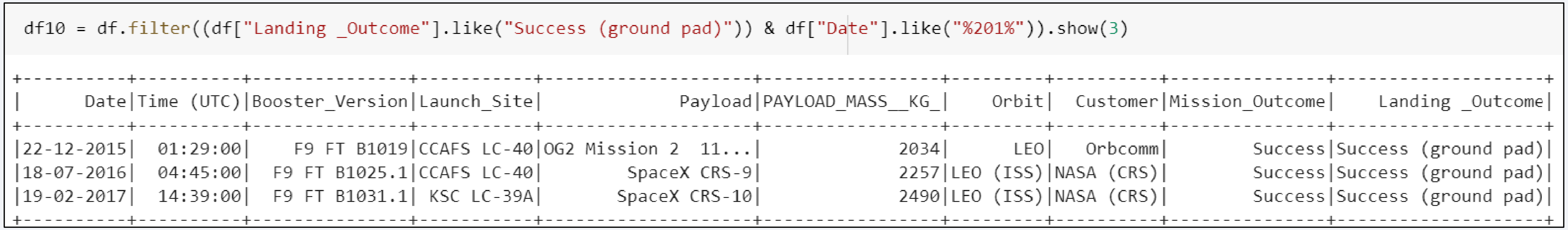
```
+----------+----------+---------------+-----------+--------------------+-----------------+---------+-------------------+------------------+--------------------+
|      Date|Time (UTC)|Booster_Version|Launch_Site|             Payload|PAYLOAD_MASS__KG_|    Orbit|           Customer|   Mission_Outcome|     Landing _Outcome|
+----------+----------+---------------+-----------+--------------------+-----------------+---------+-------------------+------------------+--------------------+
|10-01-2015|  09:47:00|  F9 v1.1 B1012|CCAFS LC-40|        SpaceX CRS-5|             2395|LEO (ISS)|         NASA (CRS)|           Success|Failure (drone ship)|
|11-02-2015|  23:03:00|  F9 v1.1 B1013|CCAFS LC-40|              DSCOVR|              570|      HEO|U.S. Air Force NA...|          Success| Controlled (ocean)|
|02-03-2015|  03:50:00|  F9 v1.1 B1014|CCAFS LC-40|ABS-3A Eutelsat 1...|             4159|      GTO|       ABS Eutelsat|           Success|          No attempt|
|14-04-2015|  20:10:00|  F9 v1.1 B1015|CCAFS LC-40|        SpaceX CRS-6|             1898|LEO (ISS)|         NASA (CRS)|           Success|Failure (drone ship)|
|27-04-2015|  23:03:00|  F9 v1.1 B1016|CCAFS LC-40|Turkmen 52 / Mona...|             4707|      GTO|Turkmenistan Nati...|          Success|          No attempt|
|28-06-2015|  14:21:00|  F9 v1.1 B1018|CCAFS LC-40|        SpaceX CRS-7|             1952|LEO (ISS)|         NASA (CRS)|Failure (in flight)|Precluded (drone ...|
|22-12-2015|  01:29:00|    F9 FT B1019|CCAFS LC-40|OG2 Mission 2  11...|             2034|      LEO|           Orbcomm|           Success|Success (ground pad)|
+----------+----------+---------------+-----------+--------------------+-----------------+---------+-------------------+------------------+--------------------+
```

Query Explanation:

The filter() function filters rows from the data frame.
The 'like' function has a wild card with the words '%2015' the percentage in the first suggests that the Date must end with 2015.

Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

SQL Query



```
df10 = df.filter((df["Landing _Outcome"].like("Success (ground pad)")) & df["Date"].like("%201%")).show(3)

+----------+----------+---------------+-----------+--------------------+-----------------+---------+----------+---------------+--------------------+
|      Date|Time (UTC)|Booster_Version|Launch_Site|             Payload|PAYLOAD_MASS__KG_|    Orbit|  Customer|Mission_Outcome|    Landing _Outcome|
+----------+----------+---------------+-----------+--------------------+-----------------+---------+----------+---------------+--------------------+
|22-12-2015|  01:29:00|    F9 FT B1019|CCAFS LC-40|OG2 Mission 2  11...|             2034|      LEO|   Orbcomm|        Success|Success (ground pad)|
|18-07-2016|  04:45:00|  F9 FT B1025.1|CCAFS LC-40|        SpaceX CRS-9|             2257|LEO (ISS)|NASA (CRS)|        Success|Success (ground pad)|
|19-02-2017|  14:39:00|  F9 FT B1031.1| KSC LC-39A|       SpaceX CRS-10|             2490|LEO (ISS)|NASA (CRS)|        Success|Success (ground pad)|
+----------+----------+---------------+-----------+--------------------+-----------------+---------+----------+---------------+--------------------+
```

Query Explanation:

The filter() function filters rows from the data frame.
The ‘like’ function has a wild card with the words.
The show() function shows the exact result.

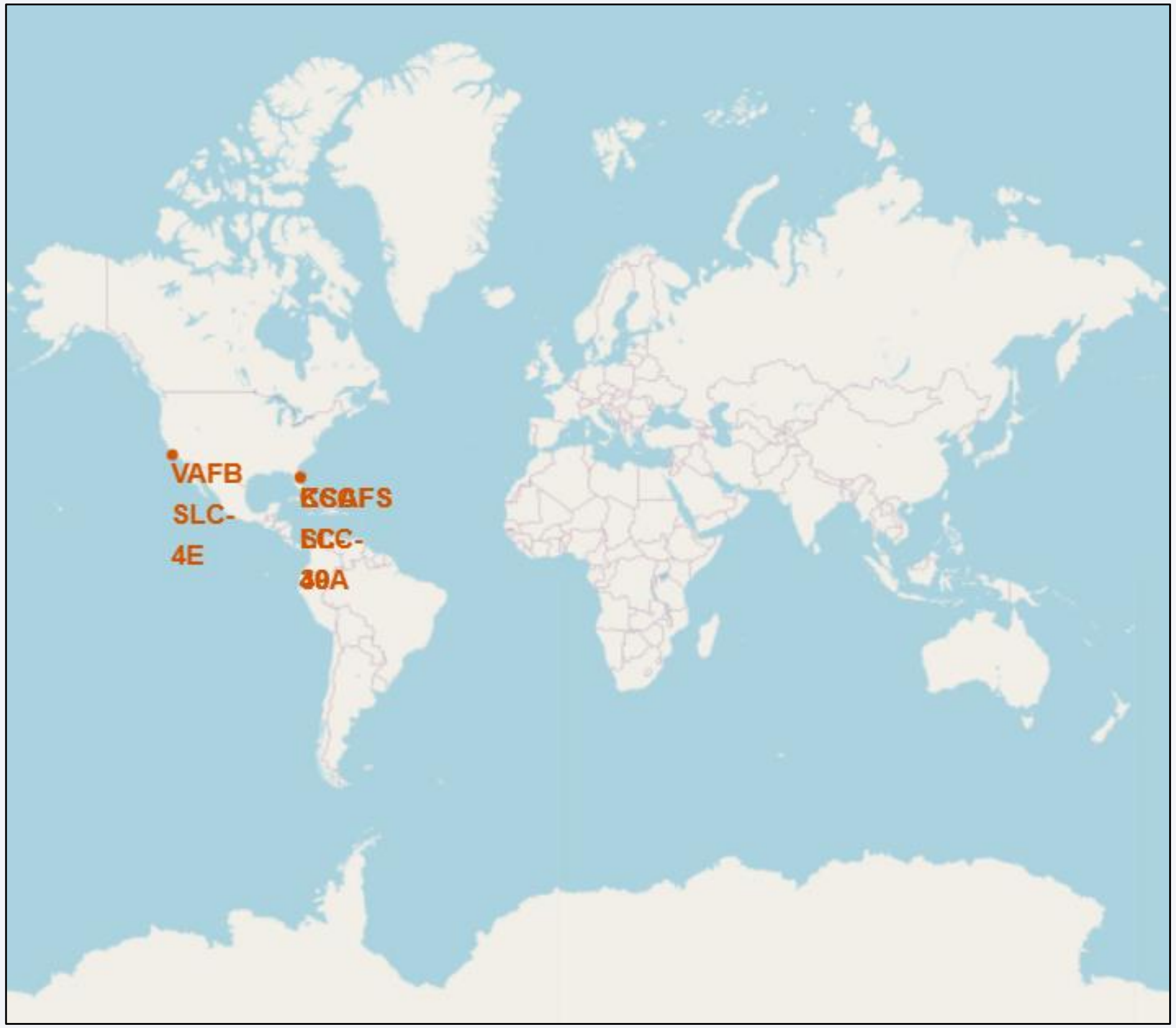
IBM Developer
SKILLS NETWORK

Section 3

Launch Sites Proximities Analysis

SPACEX

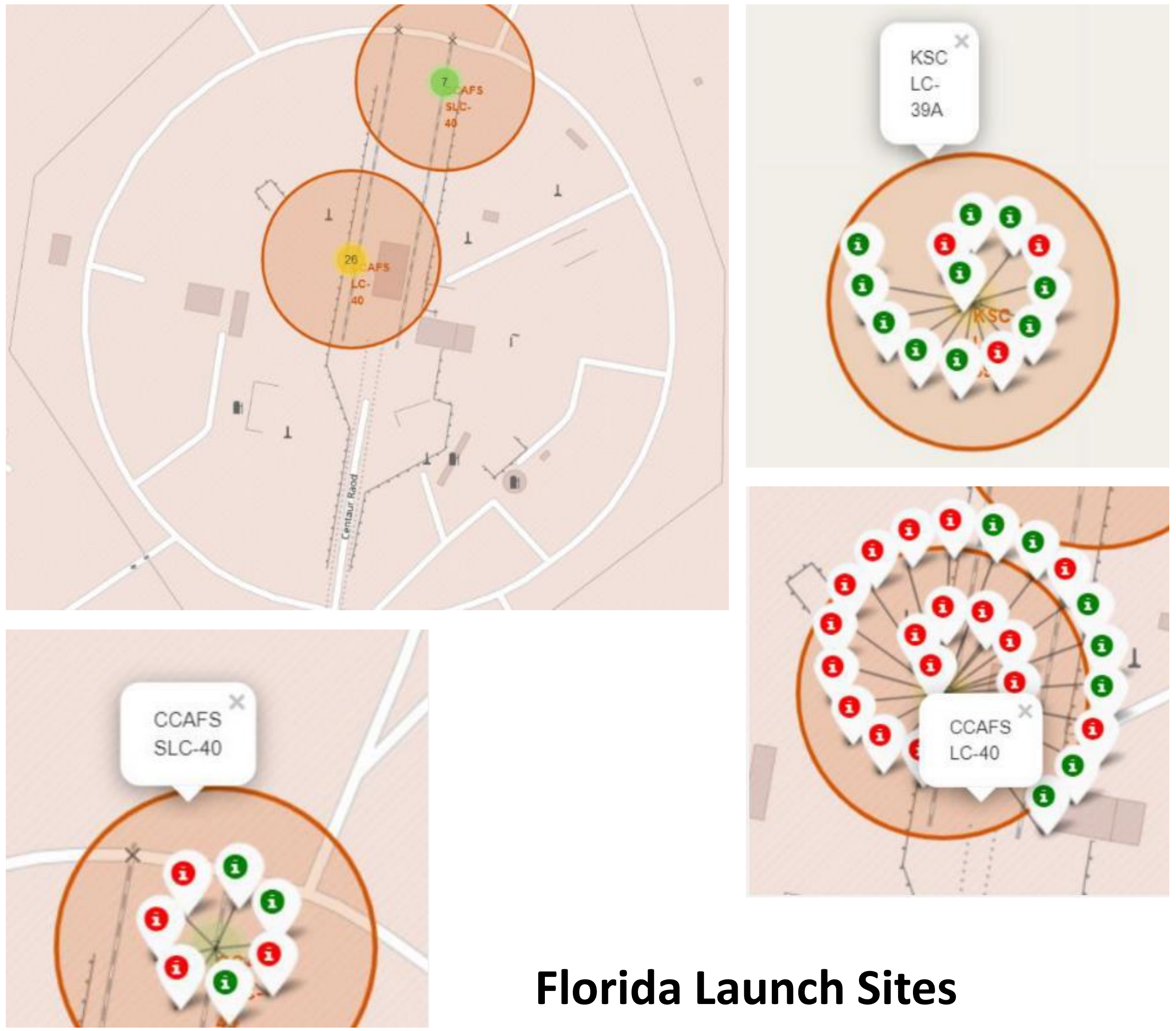
Launch Sites of SpaceX From Global Map



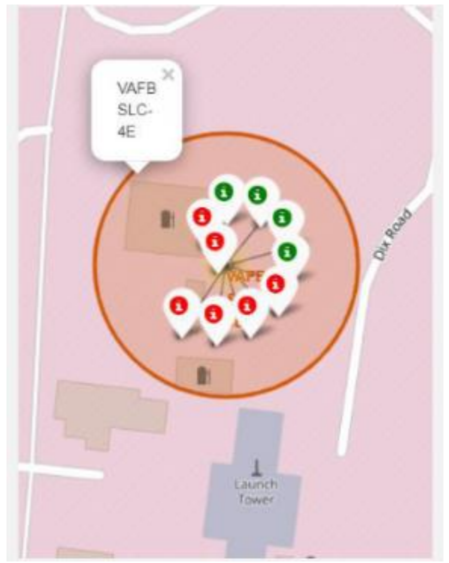
We can see that the SpaceX launch sites are in the United States of America coasts. Florida and California.

We also see that all the launch sites are not closer to Equator line but very close to the coast area.

Color Labelled Markers



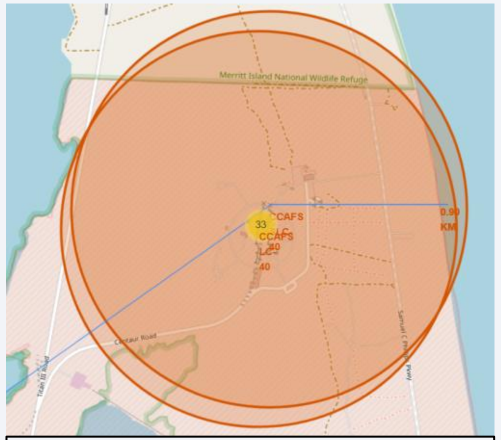
Florida Launch Sites



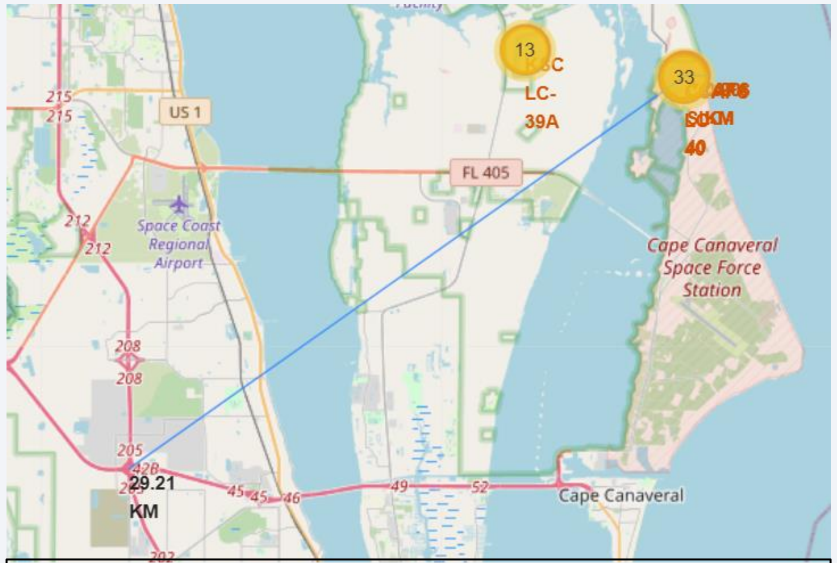
California Launch Site

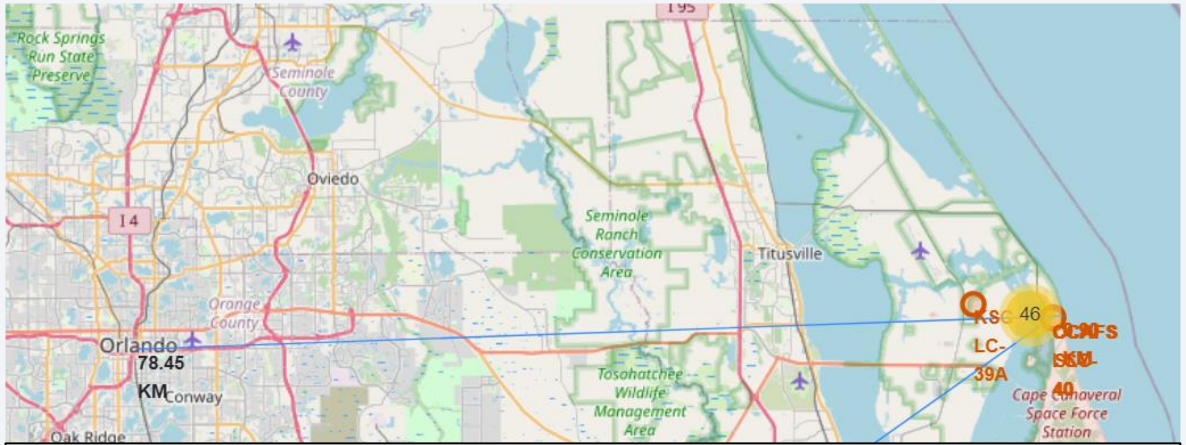
Green Marker shows successful launches and Red Marker shows failure.

Launch Sites Distance to landmarks using CCAFS-SLC-40 as a reference



Distance to Coast Line


13
KSC LC-39A
33
29.21 KM
US 1
FL 405
Space Coast Regional Airport
Cape Canaveral Space Force Station
Cape Canaveral
Distance to closest High Way



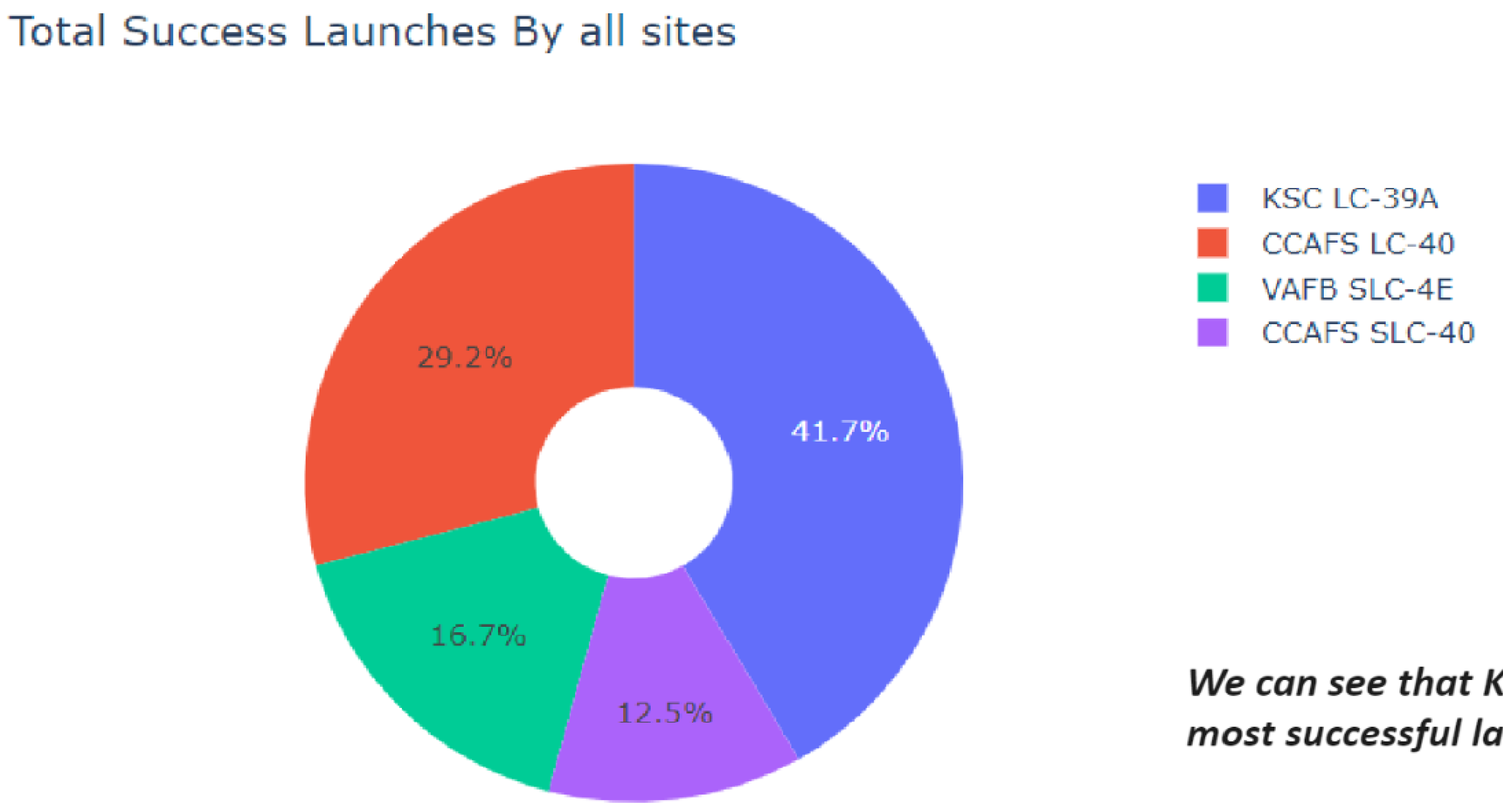
Distance to Florida City

- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

Section 4

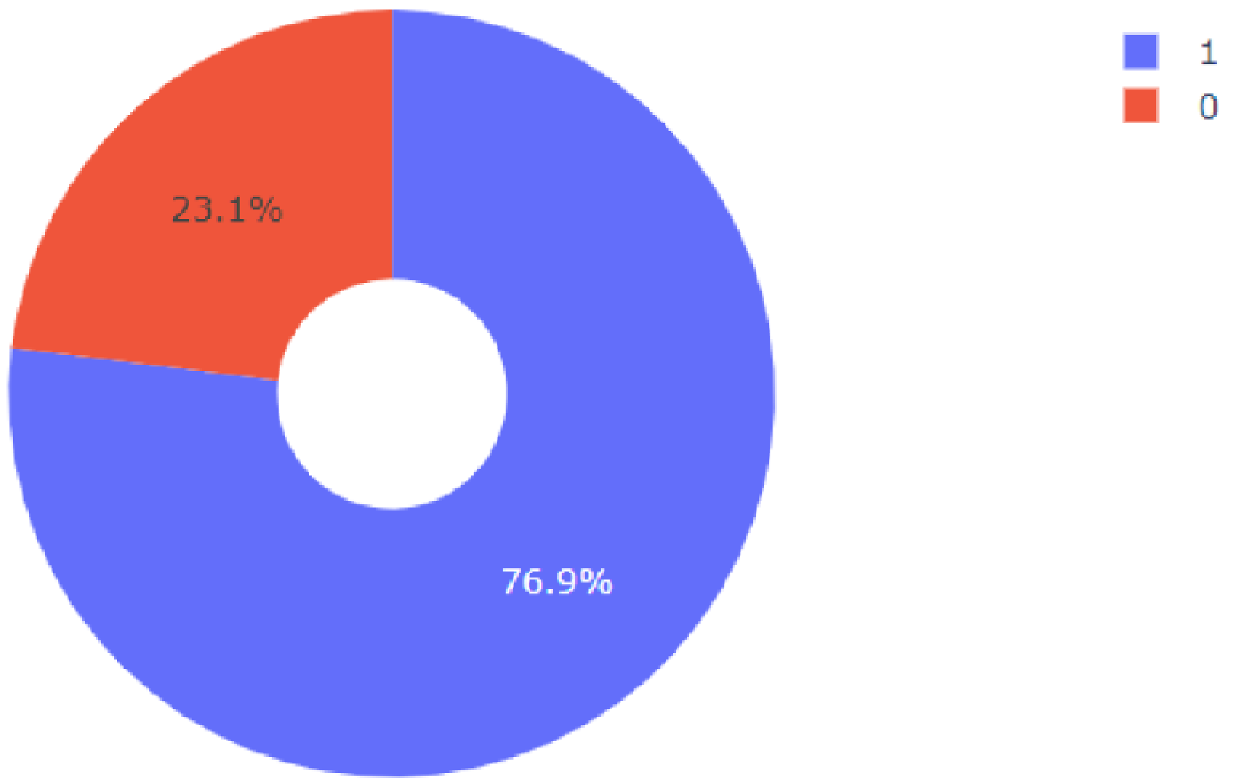
Build a Dashboard with Plotly Dash

Pie Chart Dashboard showing the success rate achieved by each Launch Site



We can see that KSC LC-39A had the most successful launches from all the sites

Pie Chart Dashboard for the Launch Site with highest launch success ratio

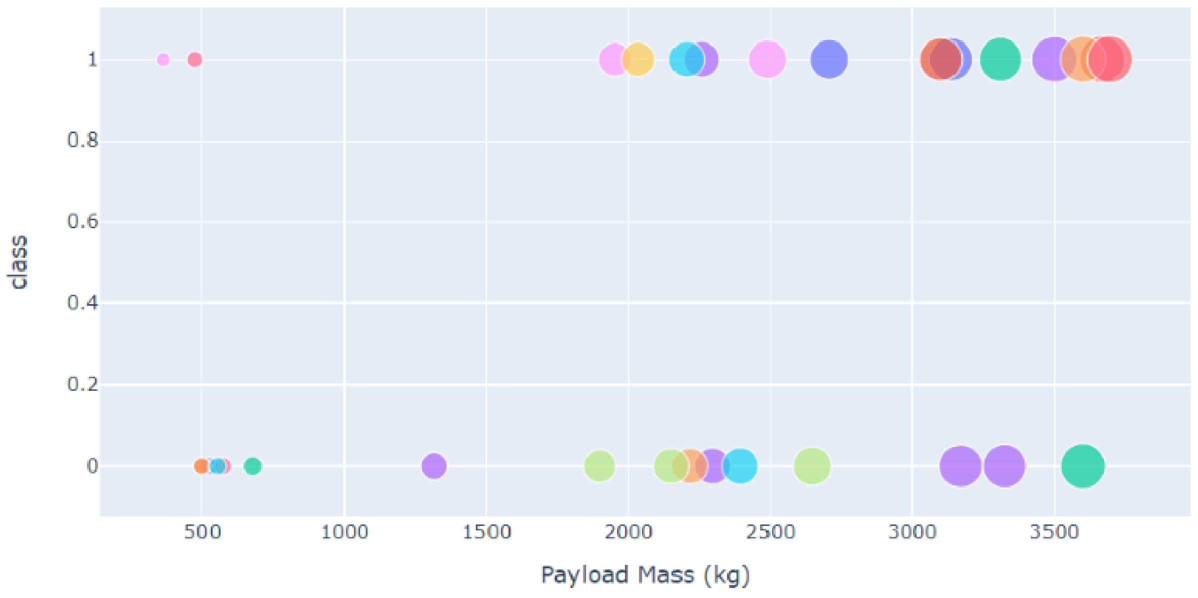


KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

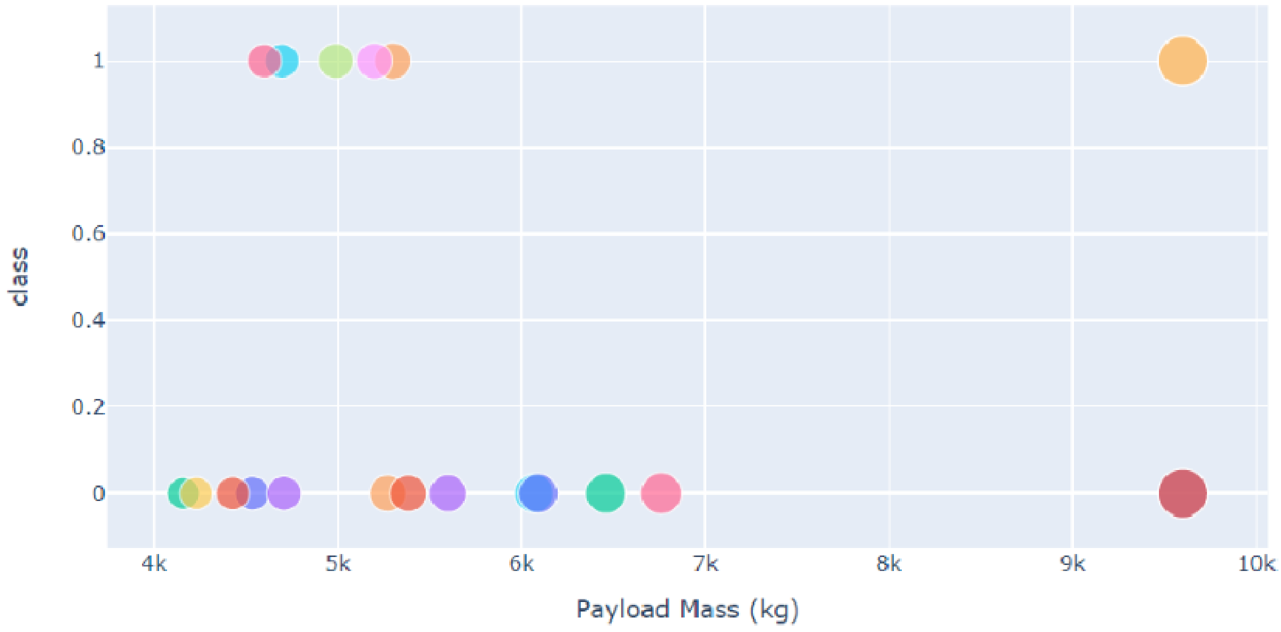
Scatter Plot Dashboard for Payload vs. Launch Outcome for all sites, with different payload selected in the range slider

IBM Developer SKILLS NETWORK

Low Weighted Payload 0kg – 4000kg



Heavy Weighted Payload 4000kg – 10000kg



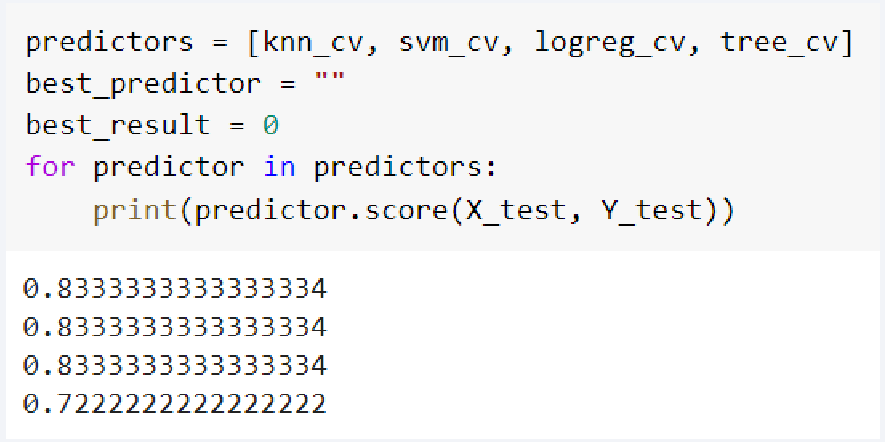
We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

IBM Developer
SKILLS NETWORK

Section 5

Predictive Analysis (Classification)

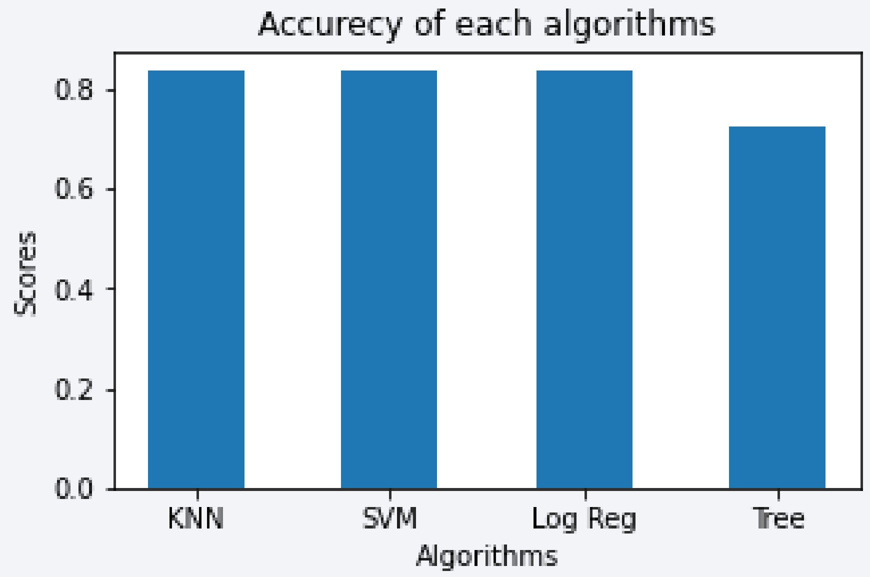
Classification Accuracy



```
predictors = [knn_cv, svm_cv, logreg_cv, tree_cv]
best_predictor = ""
best_result = 0
for predictor in predictors:
    print(predictor.score(X_test, Y_test))
```

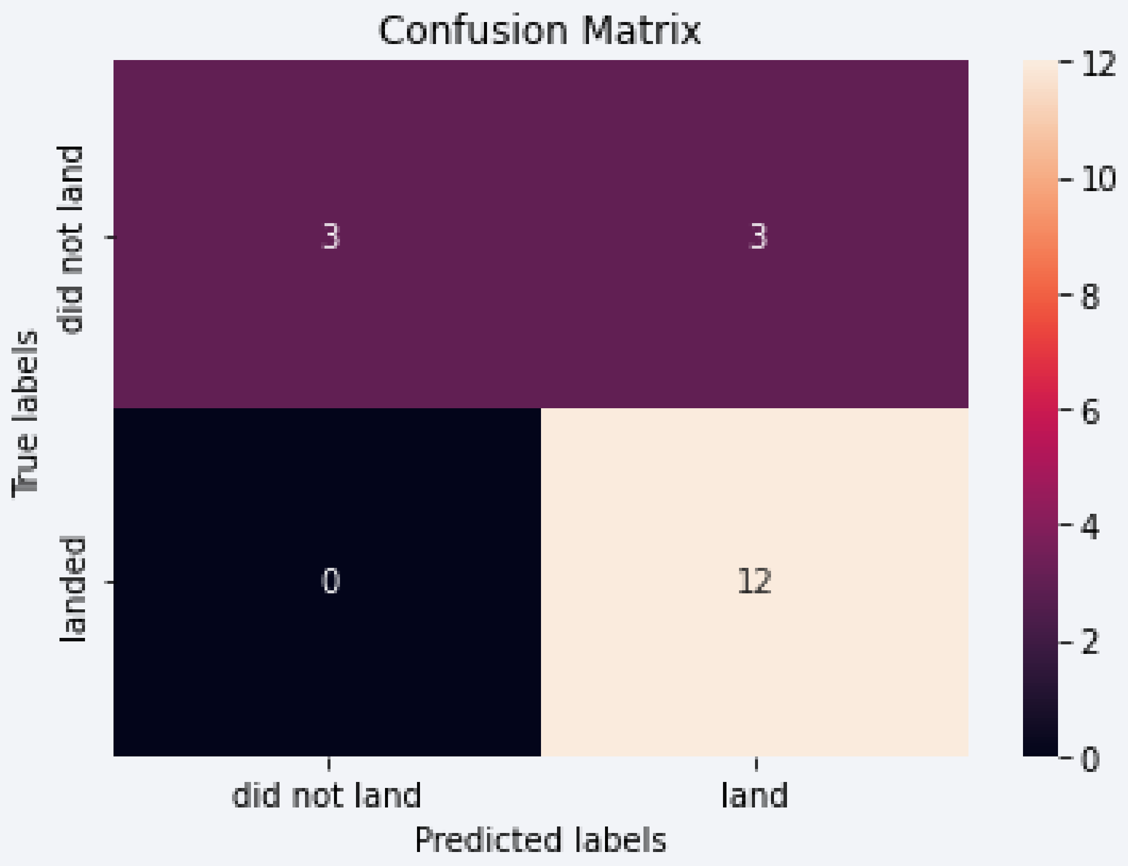
```
0.8333333333333334
0.8333333333333334
0.8333333333333334
0.7222222222222222
```

As you can see the accuracy is extremely close so we choose SVM as our winner.



After selecting the best hyperparameters for the SVM classifier using the validation data, we achieved 83.33% accuracy on the test data.

Confusion Matrix



Examining the confusion matrix, we see that Tree can distinguish between the different classes. We see that the major problem is false positives.

Conclusions

- The Tree Classifier Algorithm is the best for Machine Learning for this dataset
- Low weighted payloads perform better than the heavier payloads
- The success rates for SpaceX launches is directly proportional time in years they will eventually perfect the launches
- We can see that KSC LC-39A had the most successful launches from all the sites
- Orbit GEO,HEO,SSO,ES-L1 has the best Success Rate

Appendix

- Haversine Formula
- Python Dashboard
- SQL Queries
- Machine Learning Prediction

IBM Developer
SKILLS NETWORK

Thank you!

SPACEX