IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Ivan K Ivanov
7/19/2023

Outline

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

Executive Summary

- Summary of methodologies
- Summary of all results

Introduction

- This project is focused on predicting if the Falcon 9 Booster will successfully land after launch
- All data used in the analysis is provided by SpaceX

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data is collected through the SpaceX API for real-time information on missions. SQL databases store and retrieve the data efficiently. Supplementary data from svc files enriches the dataset for comprehensive analysis.
- Perform data wrangling
  - Data wrangling is the process of gathering, transforming, and preparing raw data from different sources to create a structured and usable dataset for analysis and modeling.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  Predictive analysis using classification models involves utilizing algorithms to categorize data into predefined classes, enabling predictions and pattern recognition.

Data Collection

- The SpaceX API at https://api.spacexdata.com/v4 provides real-time data on SpaceX missions in JSON format through HTTP requests.
- To transform the JSON data into a structured format for analysis, Python's pandas library offers the `read_json()` function, converting it into a DataFrame.
- Data from SQL databases is accessed using SQL queries through Python libraries, while svc files can be read directly into pandas DataFrames using the `read_csv()` function for analysis.

Data Collection – SpaceX API

With the API calls were reached the the following endpoints

- Booster version

https://api.spacexdata.com/v4/rockets/

- Launch Site

https://api.spacexdata.com/v4/launchpads/

- Payload

https://api.spacexdata.com/v4/rockets/

- GitHub:

https://github.com/IvanKrumov/DS_Capstone_Project/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

```
def getBoosterVersion(data):
    for x in data['rocket']:
      if x:
       response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
       BoosterVersion.append(response['name'])
```

Data Collection - Scraping

- Present your web scraping process using key phrases and flowcharts
- Add the GitHub URL of the completed web scraping notebook, as an external reference and peer-review purpose

Place your flowchart of web scraping here

Data Wrangling

- Data wrangling from a JSON file involves loading the JSON data, understanding its structure, extracting relevant information, and converting it into a structured format like a DataFrame for further analysis and modeling.

- https://github.com/IvanKrumov/DS_Capstone_Project/blob/main/jupyter-labs-webscraping.ipynb

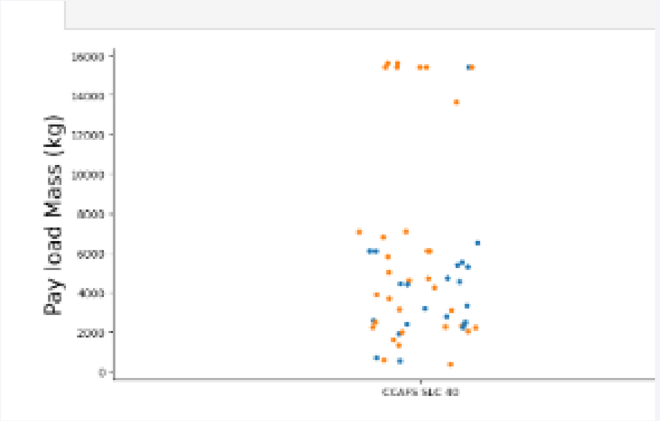
```
data_falcon9.isnull().sum()
```

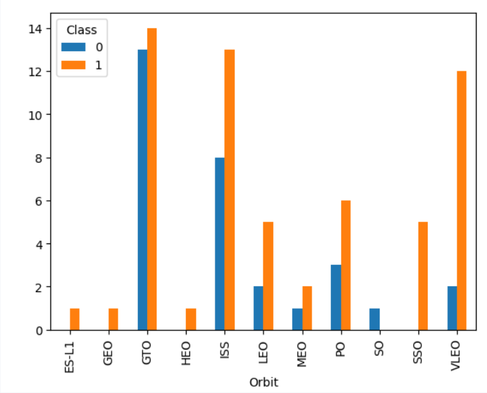
```
FlightNumber      0
Date              0
BoosterVersion    0
PayloadMass       5
```

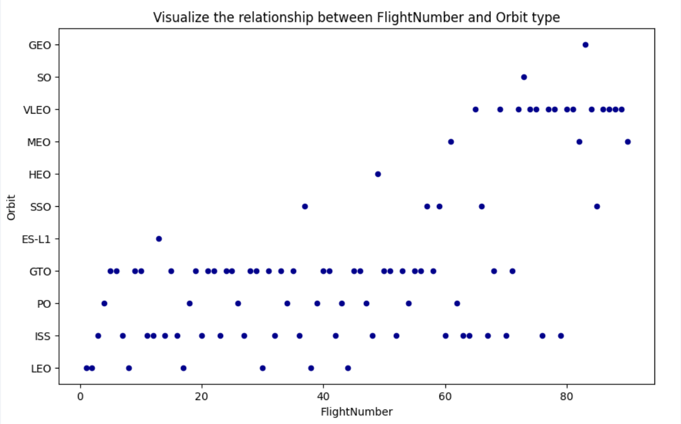
EDA with Data Visualization

- In data science, scatter plots are used to visualize the relationship between two variables, displaying individual data points on a two-dimensional plane. Line plots depict the trend or progression of data points over time or across a continuous variable, while bar charts represent categorical data by displaying rectangular bars with heights proportional to the data's frequency or value.

- https://github.com/IvanKrumov/DS_Capstone_Project/blob/main/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb







EDA with SQL

- select distinct(Launch_Site) from SPACEXTBL;
- select * from SPACEXTBL where Launch_Site like 'CCA%';
- select sum(PAYLOAD_MASS__KG_) from SPACEXTBL where Customer == 'NASA (CRS)' ;
- select min(Date) from SPACEXTBL where Mission_Outcome == 'Success' ;
- select distinct(Booster_Version) from SPACEXTBL where Mission_Outcome == 'Success' AND \
- (PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000) AND Landing_Outcome=='Success (drone ship)';
- select count(Landing_Outcome) from SPACEXTBL where Landing_Outcome == 'Success' or Landing_Outcome == 'Failure';
- sql select * FROM SPACEXTBL where (Landing_Outcome == 'Success (ground pad)' or Landing_Outcome == 'Failure (drone ship)' AND Date between '06/04/2010' \
- AND '20/03/2017') GROUP BY Landing_Outcome ORDER BY COUNT(Landing_Outcome) DESC
- Add the GitHub URL of your completed EDA with SQL notebook, as an external reference and peer-review purpose

https://github.com/IvanKrumov/DS_Capstone_Project/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- In this project's map visualization, I opted for the folium library, a powerful tool for creating interactive maps in Python. To represent the launch sites, I employed circles as markers on the map. Each circle marker was customized to signify a specific launch site, and I added tags or labels to provide clear identification of these sites. This approach allowed viewers to easily discern and associate the launch locations with the corresponding markers on the map, enhancing the overall clarity and usability of the visualization.
- Add the GitHub URL of your completed interactive map with Folium map, as an external reference and peer-review purpose https://github.com/IvanKrumov/DS_Capstone_Project/blob/main/lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

- Within the Plotly Dash board, I incorporated two interactive components, including a dcc.Dropdown menu enabling users to select from a list of launch sites. Additionally, a dcc.RangeSlider was implemented, offering users the ability to adjust the payload range. As part of the dashboard, the data was presented using scatter plots and pie charts, fostering intuitive data exploration and analysis.
- Add the GitHub URL of your completed Plotly Dash lab, as an external reference and peer-review purpose
  https://github.com/IvanKrumov/DS_Capstone_Project/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- The process will involve data preprocessing to standardize the data, followed by the Train_test_split method to divide the data into training and testing sets. The model training will include performing Grid Search to find optimal hyperparameters for improved performance. We will evaluate the accuracy of different models, including Logistic Regression, Support Vector Machines, Decision Tree Classifier, and K-nearest neighbors, using the training data. Finally, we will generate a confusion matrix to assess the models' performance.
- Add the GitHub URL of your completed predictive analysis lab, as an external reference and peer-review purpose
  https://github.com/IvanKrumov/DS_Capstone_Project/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

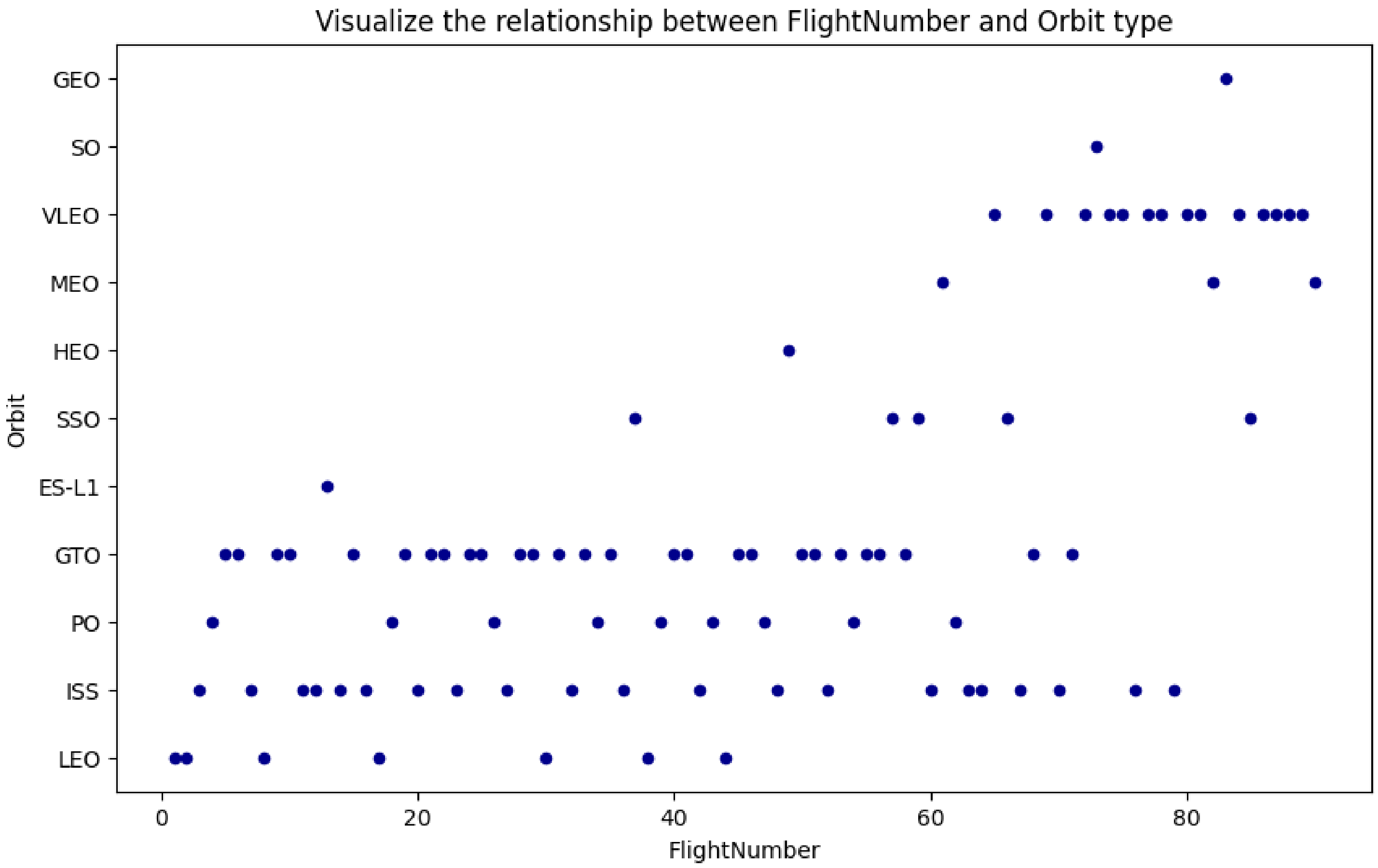
Results

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

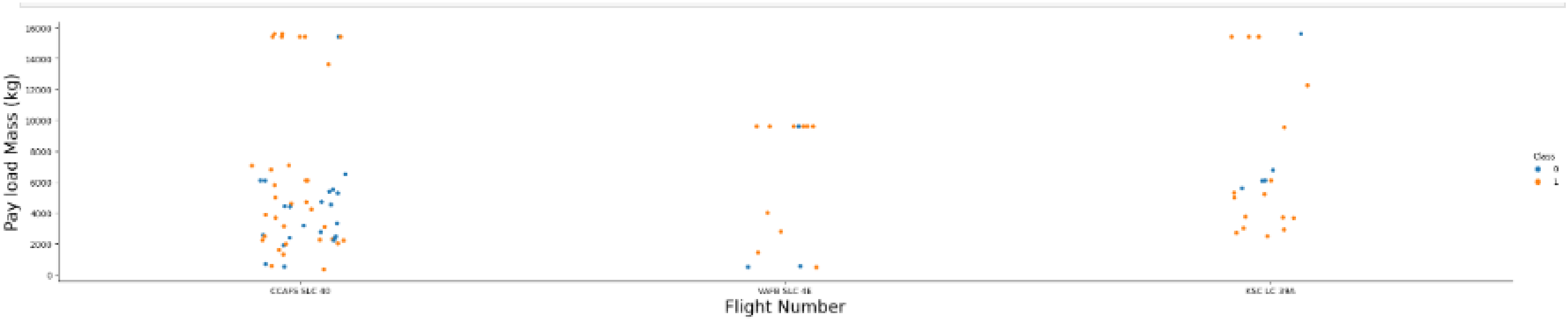
Section 2

Insights drawn from EDA

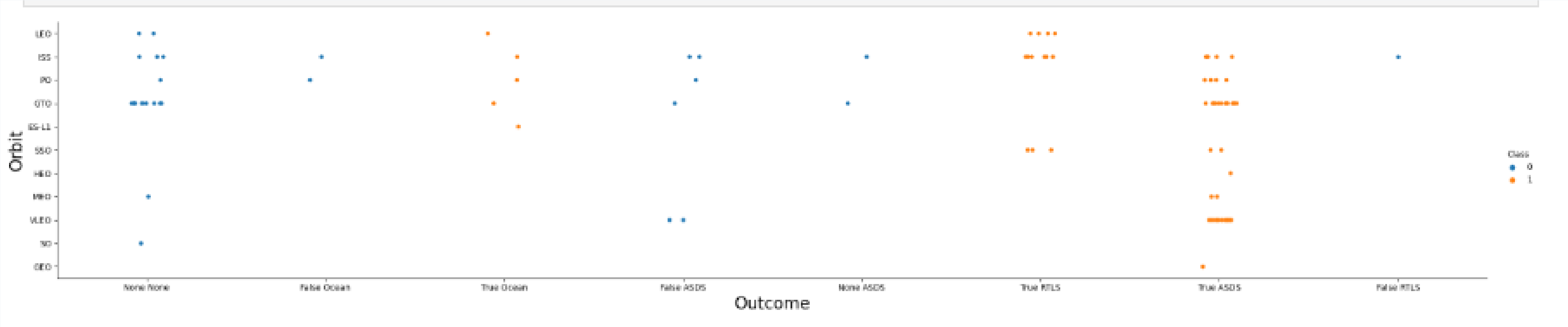
Flight Number vs. Launch Site



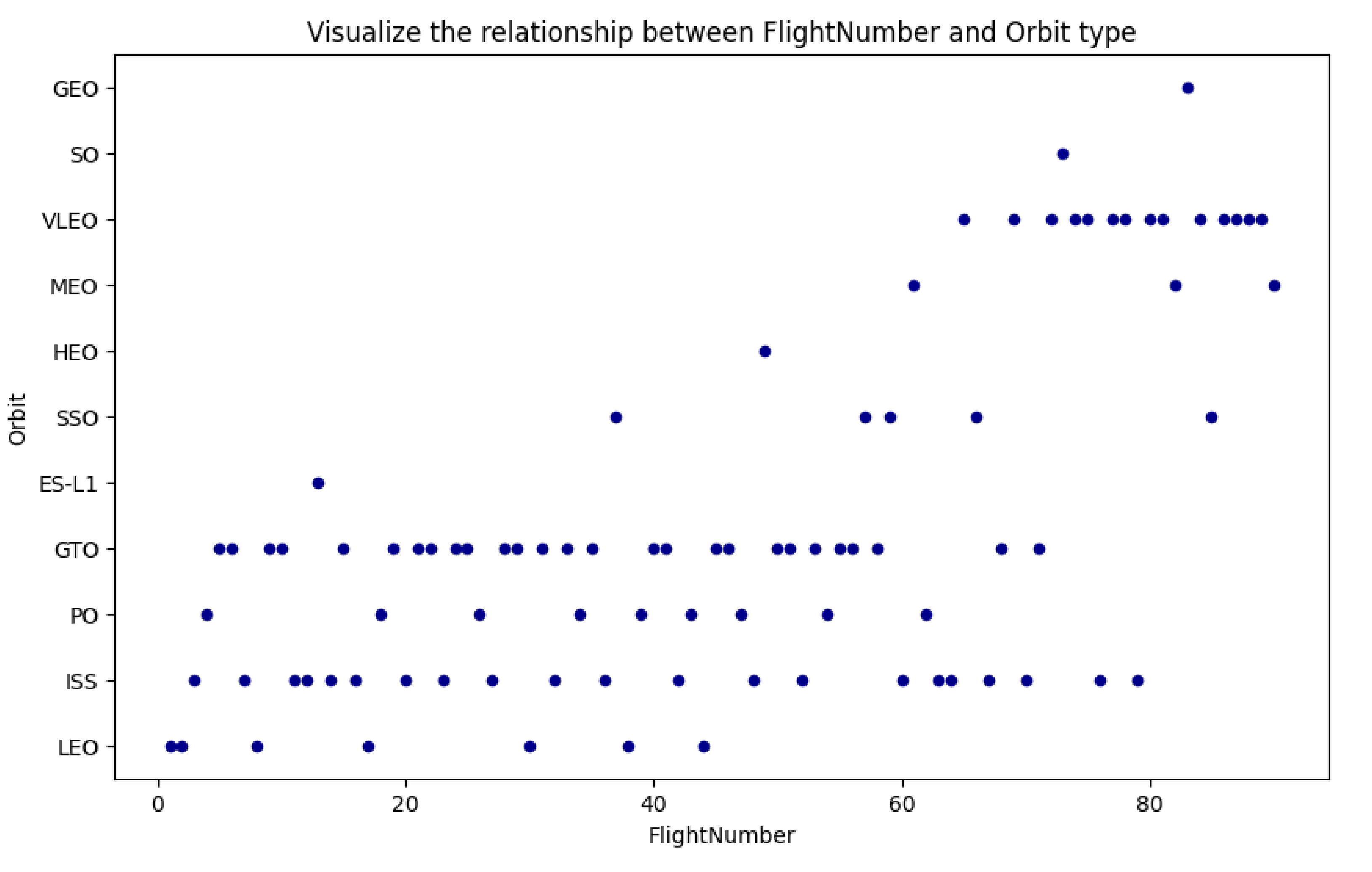
Payload vs. Launch Site



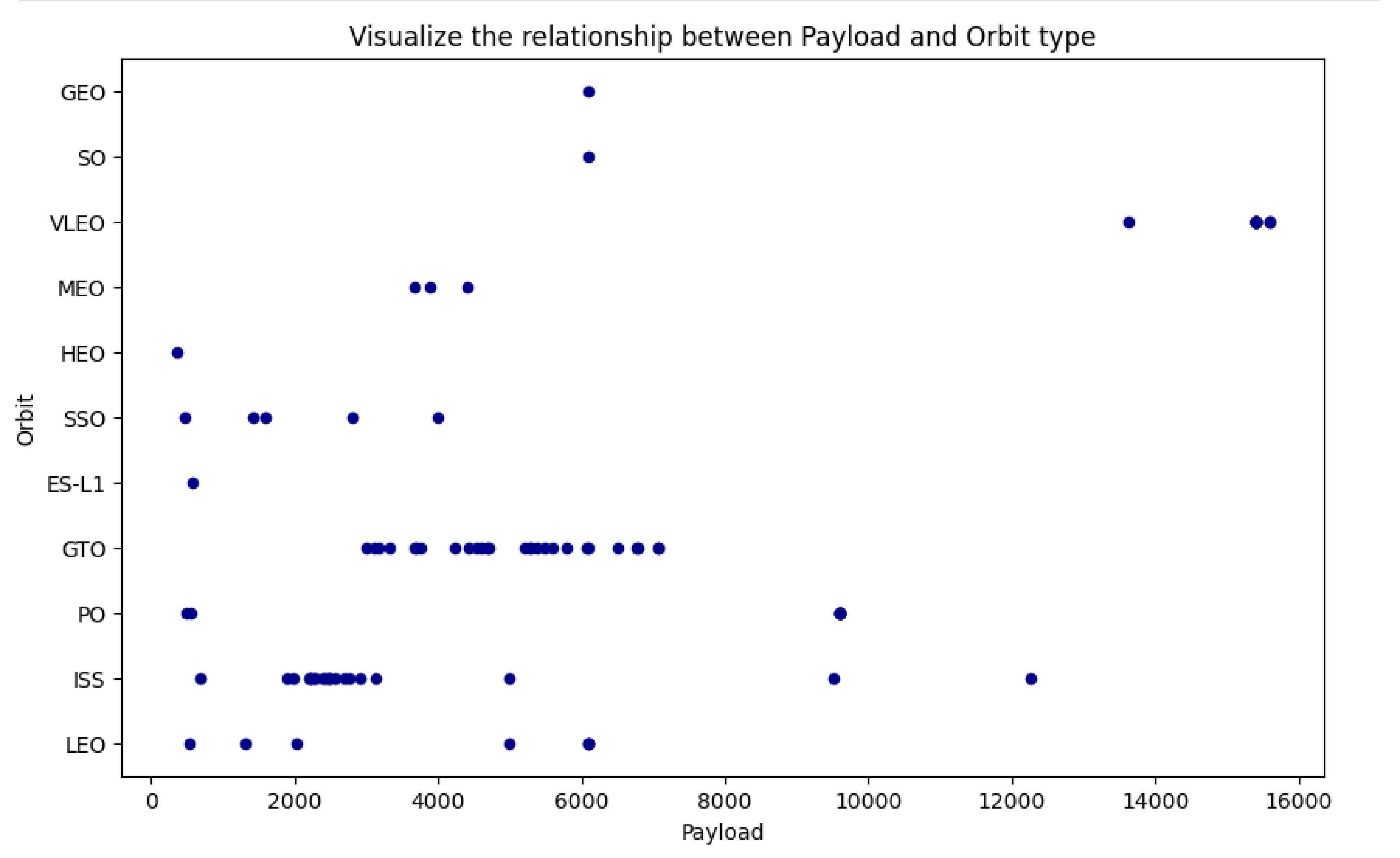
Success Rate vs. Orbit Type



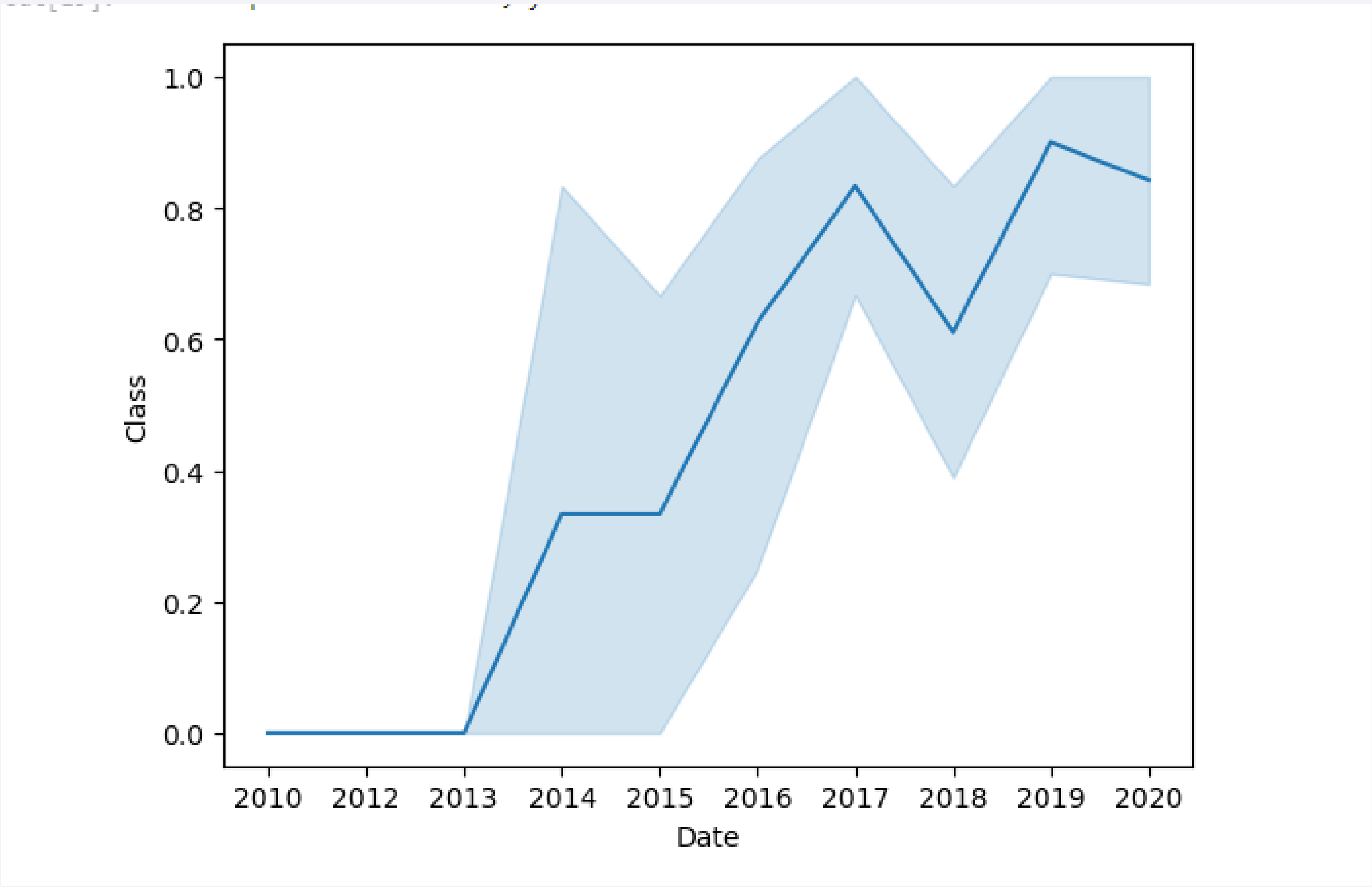
Flight Number vs. Orbit Type



Payload vs. Orbit Type



Launch Success Yearly Trend



All Launch Site Names

```
df = pd.read_sql_query("select 
distinct(Launch_Site) from SPACEXTBL;", con)

#print the dataframe
df
```

Launch Site Names Begin with 'CCA'

```
df = pd.read_sql_query("select * from SPACEXTBL where Launch_Site like 'CCA%';", con)

#print the dataframe
df.head()
```

	Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
0	06/04/2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0.0	LEO	SpaceX	Success	Failure (parachute)
1	12/08/2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0.0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	22/05/2012	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525.0	LEO (ISS)	NASA (COTS)	Success	No attempt
3	10/08/2012	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500.0	LEO (ISS)	NASA (CRS)	Success	No attempt
4	03/01/2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677.0	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

```
df = pd.read_sql_query("select sum(PAYLOAD_MASS__KG_) from SPACEXTBL where Customer == 'NASA (CRS)' ;", con)
# PAYLOAD_MASS__KG_
# Payload
# Customer
#print the dataframe
df
```

	sum(PAYLOAD_MASS__KG_)
0	45596.0

Average Payload Mass by F9 v1.1

Display average payload mass carried by booster version F9 v1.1

```
df = pd.read_sql_query("select avg(PAYLOAD_MASS__KG_) from SPACEXTBL where Booster_Version == 'F9 v1.1' ;", con)
# Booster_Version
# Payload
# Customer
#print the dataframe
df
```

	avg(PAYLOAD_MASS__KG_)
0	2928.4

First Successful Ground Landing Date

```
df = pd.read_sql_query("select min(Date) from SPACEXTBL where Mission_Outcome == 'Success' ;", con)

#print the dataframe
df
```

	min(Date)
0	01/06/2014

Successful Drone Ship Landing with Payload between 4000 and 6000

```python
# df = pd.read_sql_query("select distinct(Landing_Outcome) from SPACEXTBL;", con)
df = pd.read_sql_query("select distinct(Booster_Version) from SPACEXTBL where Mission_Outcome == 'Success' AND \
                        (PAYLOAD_MASS__KG_  > 4000 AND PAYLOAD_MASS__KG_ < 6000) AND Landing_Outcome=='Success (drone ship)';", con)

#print the dataframe
df
```

	Booster_Version
0	F9 FT B1022
1	F9 FT B1026
2	F9 FT B1021.2
3	F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

```
# df = pd.read_sql_query("select sum()* from SPACEXTBL;", con)
# Success Failure
df = pd.read_sql_query("select count(Landing_Outcome) from SPACEXTBL where Landing_Outcome == 'Success' or Landing_Outcome == 'Failure';", con)
df
```

	count(Landing_Outcome)
0	41

Boosters Carried Maximum Payload

```
df = pd.read_sql_query("select * from (select Booster_Version from SPACEXTBL order by Payload DESC limit 10);", con)
# df = pd.read_sql_query("select * from SPACEXTBL;", con)
df
```

	Booster_Version
0	F9 B4 B1043.1
1	F9 v1.1 B1016
2	F9 B4 B1045.1
3	F9 FT B1023.1
4	F9 v1.1
5	F9 B5B1047.1
6	F9 B5B1049.1
7	F9 B5 B1049.3
8	F9 B5 B1051.5
9	F9 B5 B1059.3

2015 Launch Records

```
df = pd.read_sql_query("select Date, Booster_Version, Landing_Outcome, Launch_Site from SPACEXTBL where substr(Date,7,4)='2015' and \
                        Landing_Outcome == 'Failure (drone ship)';", con)

df
```

	Date	Booster_Version	Landing_Outcome	Launch_Site
0	01/10/2015	F9 v1.1 B1012	Failure (drone ship)	CCAFS LC-40
1	14/04/2015	F9 v1.1 B1015	Failure (drone ship)	CCAFS LC-40

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

```
test_score_distribution = %sql select * FROM SPACEXTBL where (Landing_Outcome == 'Success (ground pad)' or Landing_Outcome == 'Failure (drone ship)' AND Date between '06/04/2010' \
                        AND '20/03/2017') GROUP BY Landing_Outcome ORDER BY COUNT(Landing_Outcome) DESC;
```

```
 * sqlite:///my_data1.db
Done.
```

```
dataframe = test_score_distribution.DataFrame()
dataframe
```

	Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
0	22/12/2015	1:29:00	F9 FT B1019	CCAFS LC-40	OG2 Mission 2 11 Orbcomm-OG2 satellites	2034.0	LEO	Orbcomm	Success	Success (ground pad)
1	14/04/2015	20:10:00	F9 v1.1 B1015	CCAFS LC-40	SpaceX CRS-6	1898.0	LEO (ISS)	NASA (CRS)	Success	Failure (drone ship)

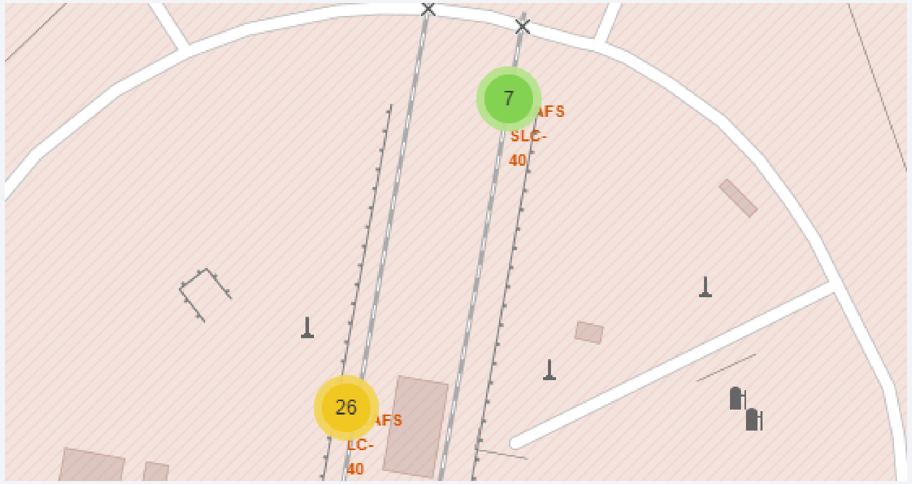
Section 3

Launch Sites Proximities Analysis

SpaceX Launch Sites

SpaceX Successful Launch Map

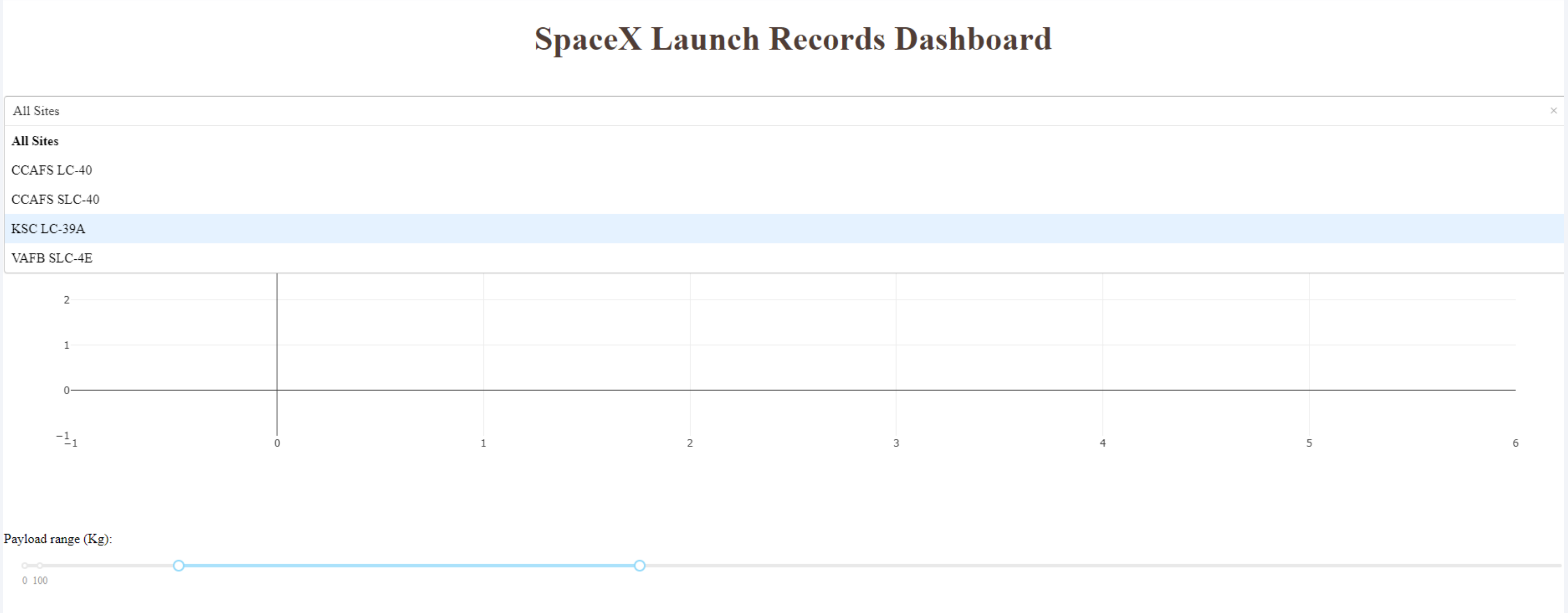
SpaceX Detailed Diagram



Section 4

Build a Dashboard with Plotly Dash

Dashboard Menu



<Dashboard Screenshot 2>

- Replace <Dashboard screenshot 2> title with an appropriate title
- Show the screenshot of the piechart for the launch site with highest launch success ratio
- Explain the important elements and findings on the screenshot

<Dashboard Screenshot 3>

- Replace <Dashboard screenshot 3> title with an appropriate title

- Show screenshots of Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider

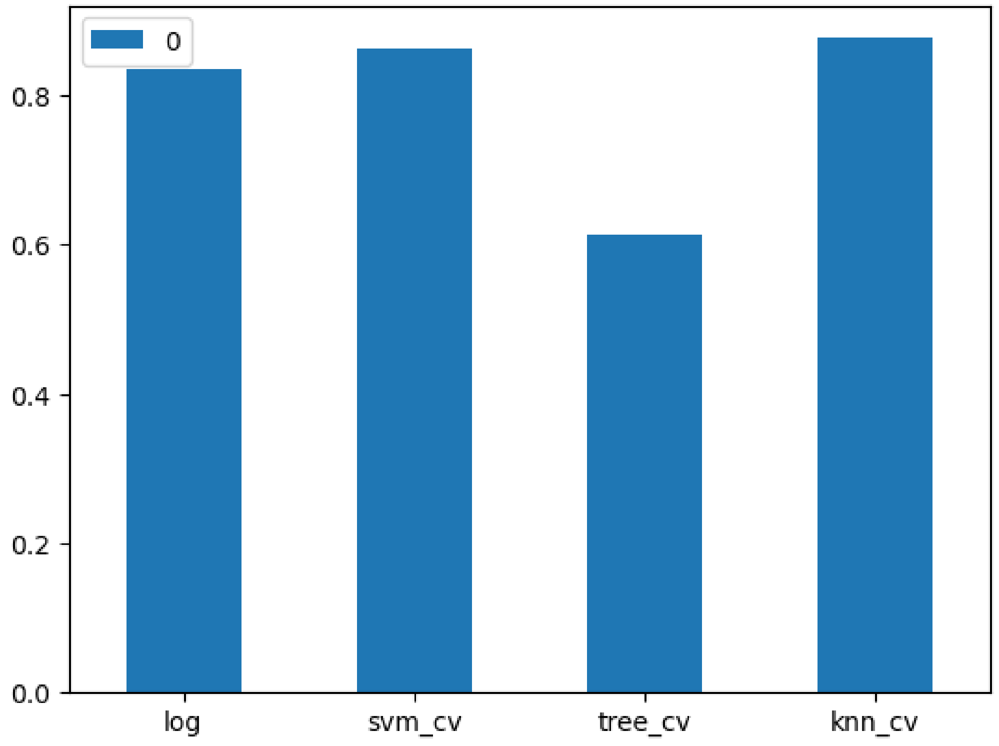
- Explain the important elements and findings on the screenshot, such as which payload range or booster version have the largest success rate, etc.

Section 5

Predictive Analysis (Classification)

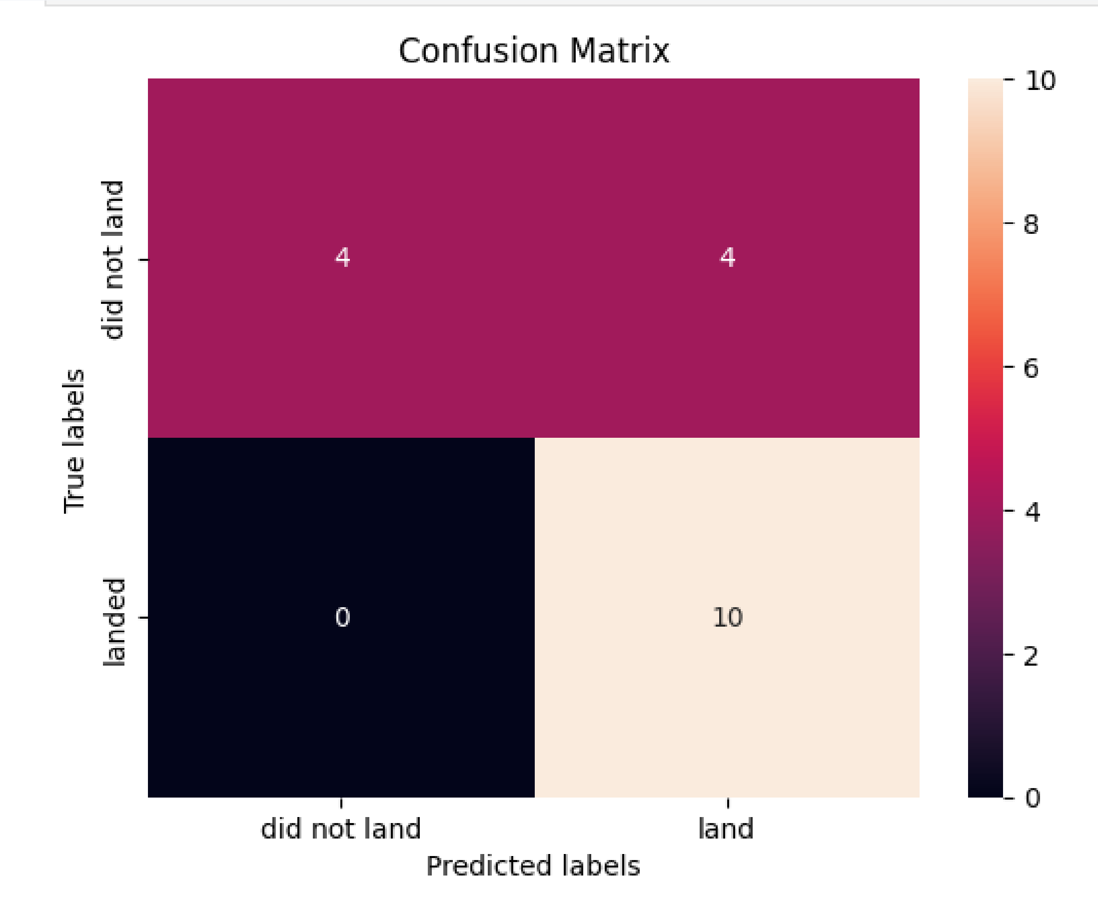
Classification Accuracy

```
[48]: ax = df_acc.plot.bar(rot=0)
```



log	0.835714
svm_cv	0.862500
tree_cv	0.613333
knn_cv	0.876786

Confusion Matrix



Conclusions

- KNN Model has highest model accuracy
- KNN Model has best performing Confusion Matrix
- According to all data Falcon 9 will lend

Appendix

Thank you!