IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Bruno C. Storino
April 24th, 2024

Outline

Executive Summary

The objective of this project report is to predict SpaceX future rocket first-stage landing outcomes using machine learning models. The results will support identifying competitive pricing to compete with SpaceX.

- Summary of methodologies
  - Data collection using API and Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis (EDA) with SQL and Data Visualization
  - Interactive Visual Analytics with Folium
  - Machine Learning Prediction
- Summary of all results
  - Exploratory Data Analysis (EDA) Results
  - Interactive Analytics
  - Predictive Analytics Results

Introduction

- Project background and context

  Space X 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 Space X 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 space X for a rocket launch.

- Problems you want to find answers
  - Identifying factors that influence the landing outcome
  - Establish the relationship between each variables and how it is affecting the outcome
  - The best condition needed to increase the probability of successful landing

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - SpaceX REST API and web scrapping from Wikipedia
- Data wrangling
  - One-hot encoding for categorical features
- Exploratory data analysis (EDA)
  - SQL and data visualization
- Interactive visual analytics
  - Folium and Plotly Dash
- Predictive analysis
  - Logistic Regression, Support Vector Machine, Decision Tree, K-Nearest Neighbor.

Data Collection – SpaceX API

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

```python
response = requests.get(spacex_url)
```

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

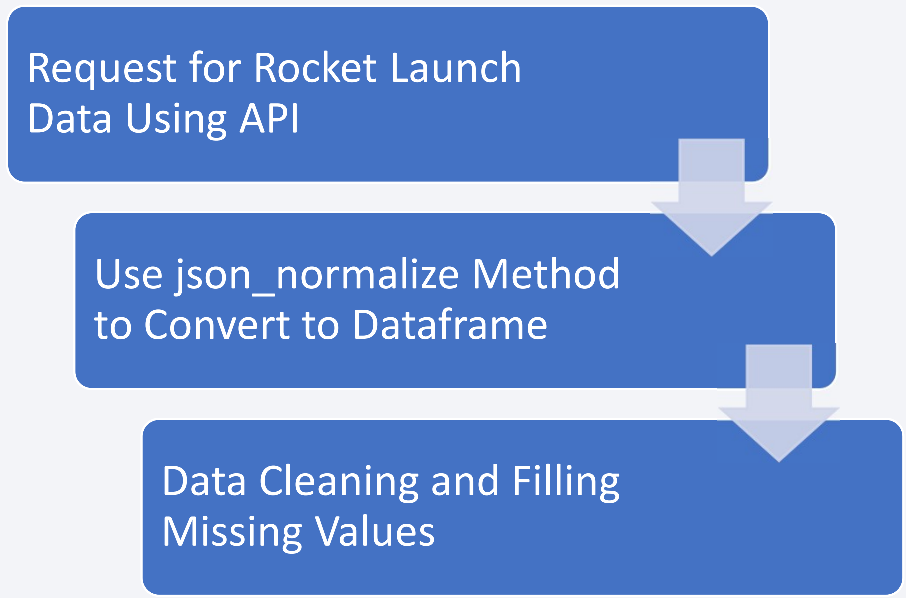
```python
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows that have multiple payloads in a single rocket.
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in the list and replace the feature.
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the time
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```



https://github.com/bstorino/IBM-Data-Science-Final-Presentation/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

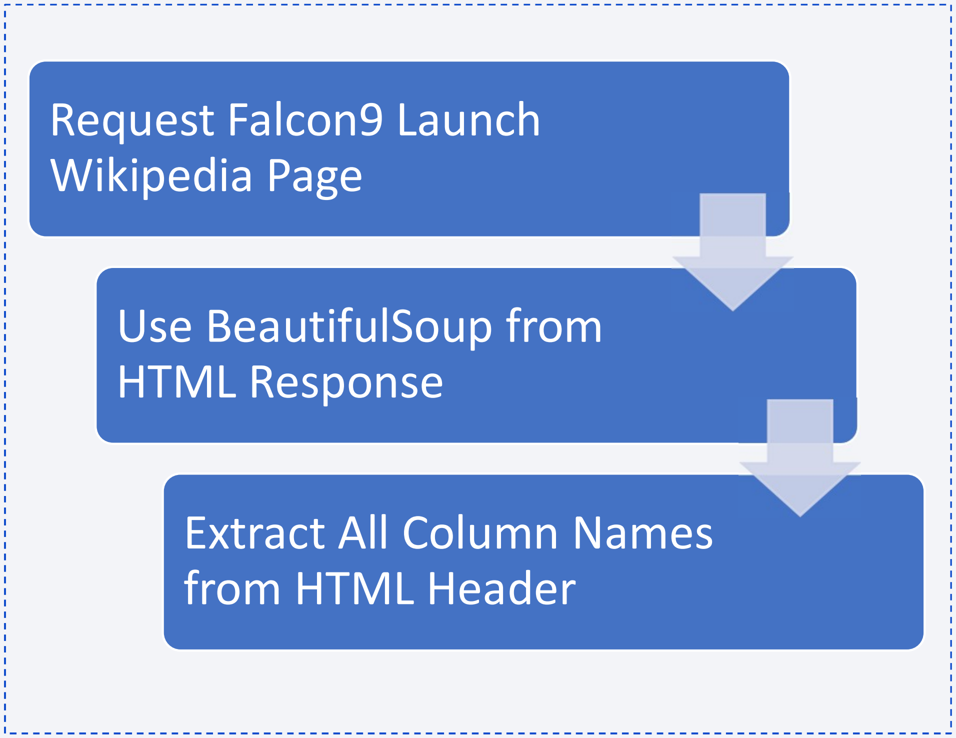
Data Collection - Scraping

```python
# use requests.get() method with the provided static_url
# assign the response to a object
r = requests.get(static_url)
data = r.text
```

```python
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(data,"html.parser")
```

```python
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            #print(flight_number)
            datatimelist=date_time(row[0])

            # Date value
            # TODO: Append the date into launch_dict with key `Date`
            date = datatimelist[0].strip(',')
            #print(date)
```



https://github.com/bstorino/IBM-Data-Science-Final-Presentation/blob/main/jupyter-labs-webscraping.ipynb

Data Wrangling

```
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()
```

```
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
landing_outcomes
```

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = [0 if x in bad_outcomes else 1 for x in df['Outcome']]
# landing_class
df['Class']=landing_class
print(df[['Class']].head(8))
print(df["Class"].mean())  # probability of positive outcome 2/3
print(df.head(5))
```

```
df.to_csv("dataset_part_2.csv", index=False)
```

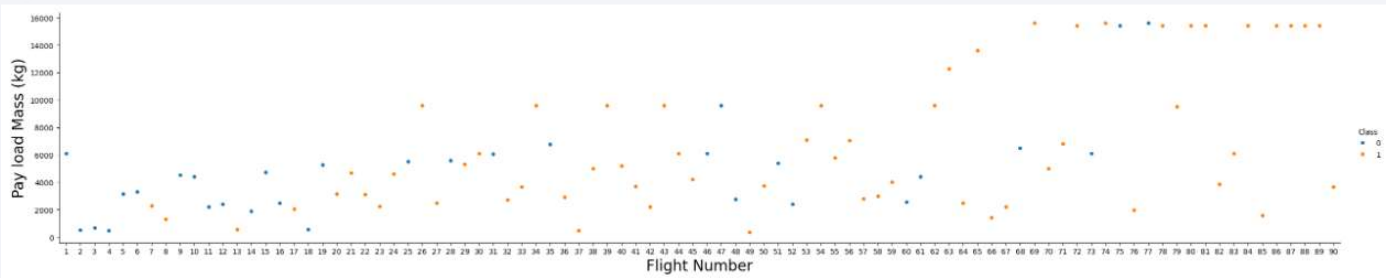
https://github.com/bstorino/IBM-Data-Science-Final-Presentation/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

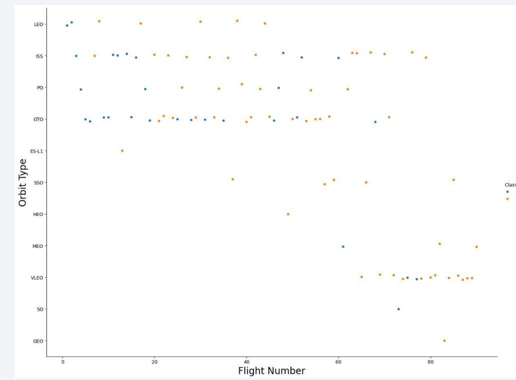
Calculate Number of Launches on Each Site

Calculate the Number and of Mission Outcomes per Orbit Type

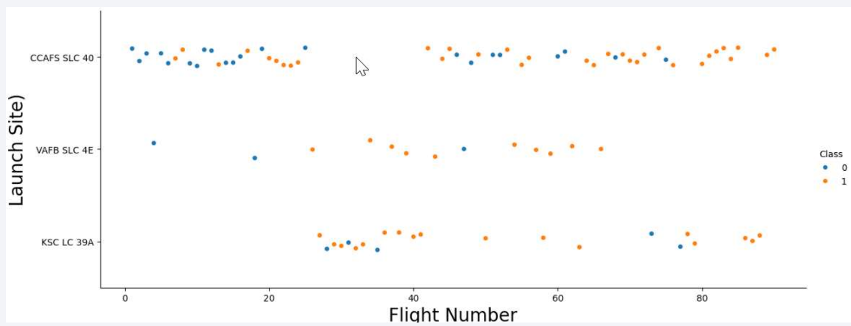
Create Label for Landing Outcome and Export Results to a CSV file

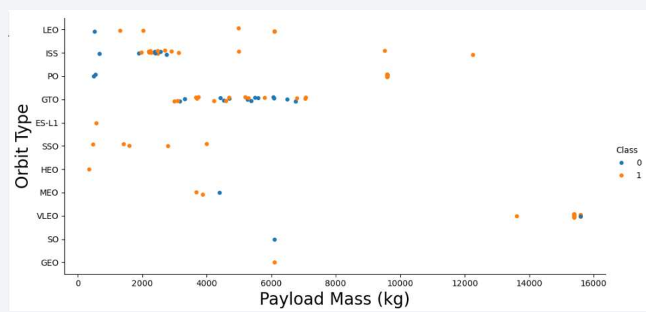
EDA with Data Visualization – Scatter Plots





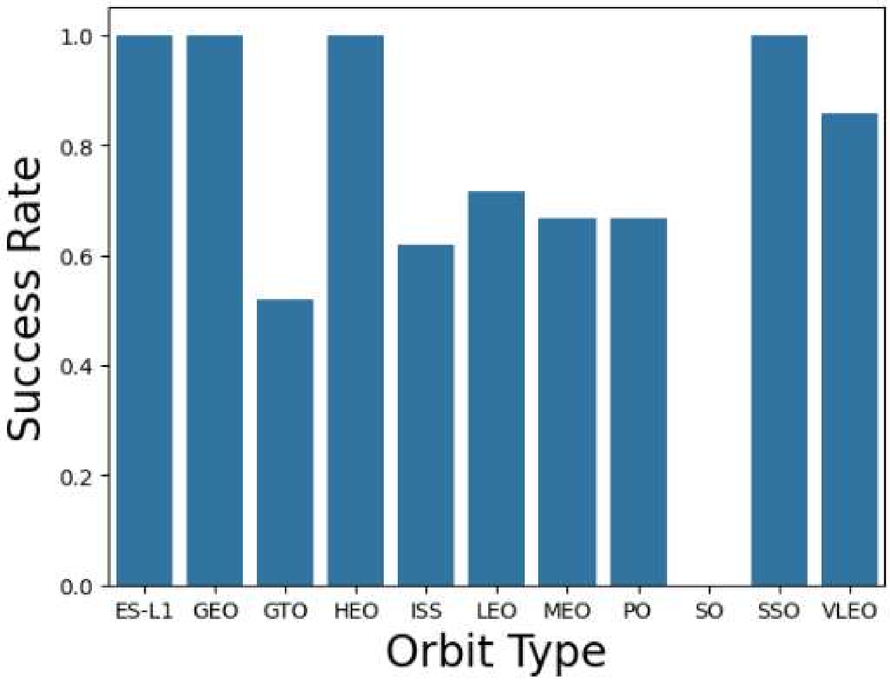
- Scatter Plots:
  - Payload vs. Flight Number
  - Flight Number vs. Launch Site
  - Flight Number vs. Orbit Type
  - Payload vs. Orbit Type

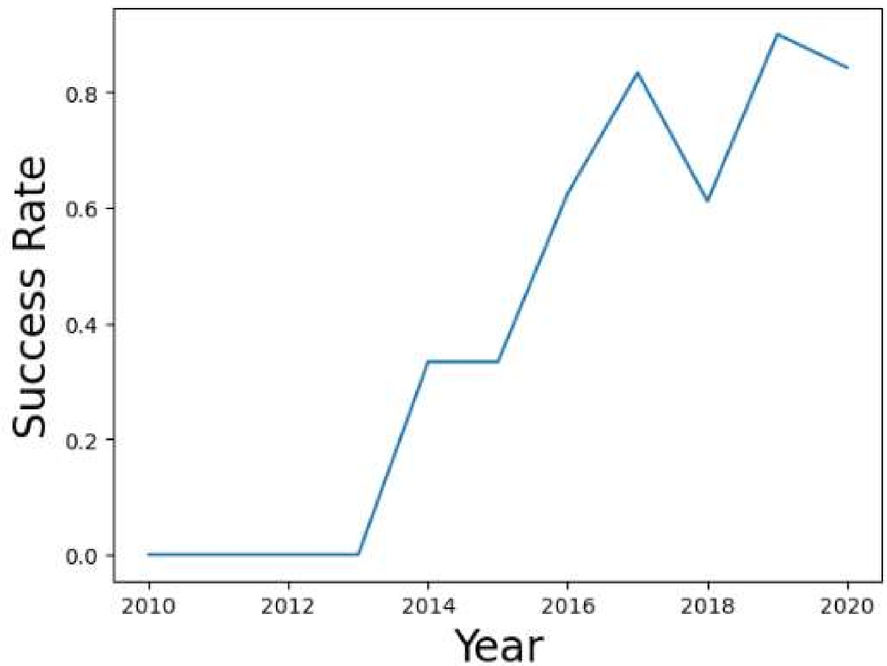




- https://github.com/bstorino/IBM-Data-Science-Final-Presentation/blob/main/edadataviz.ipynb

EDA with Data Visualization – Success Rate





- https://github.com/bstorino/IBM-Data-Science-Final-Presentation/blob/main/edadataviz.ipynb

EDA with SQL

- Performed Querries:
  - Names of launch sites
  - Top 5 records for launch sites beginning with ‘CCA’ string
  - Total payload carried by booster launched by NASA (CRS)
  - Average payload carried by booster version F9
  - Date when first successful landing outcome was achieved
  - List of boosters with successful outcomes landing on drone ships with payload greater than 4,000 but less than 6,000
  - Number of successful and failure mission outcomes
  - List of booster versions that have carried the maximum payload
  - List of failed landing outcomes on drone ships, their booster versions, and launch sites in 2015
  - Rank with the count of landing outcomes or success between 6/4/2010 and 3/20/2017
- https://github.com/bstorino/IBM-Data-Science-Final-Presentation/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

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 then assigned the dataframe launch_outcomes (failure,success) to classes 0 and 1 with Red and Green markers on the map in MarkerCluster().
- We then used the Haversine's formula to calculated the distance of the launch sites to various landmark to find answer to the questions of:
  - How close the launch sites with railways, highways and coastlines?
  - How close the launch sites with nearby cities?

- https://github.com/bstorino/IBM-Data-Science-Final-Presentation/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- Interactive Dashboards:
  - Pie chart with total launches by sites.
  - Scatter plot with relationship between payloads and outcomes for different booster versions
- Explain why you added those plots and interactions
- Add the GitHub URL of your completed Plotly Dash lab, as an external reference and peer-review purpose

Predictive Analysis (Classification)

Build the Model	Evaluate the Model	Improving the Model	Select Best Model
• Load Data into Numpy and Pandas • Transform data • Split data into training and test datasets	• Check each model accuracy • Get tuned hyperparameters • Plot confusion matrices	• Feature engineering • Algorithm tuning	• Evaluate accuracy scores • Best accuracy score will be the best performing model

- https://github.com/bstorino/IBM-Data-Science-Final-Presentation/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

Overall project results in three categories:

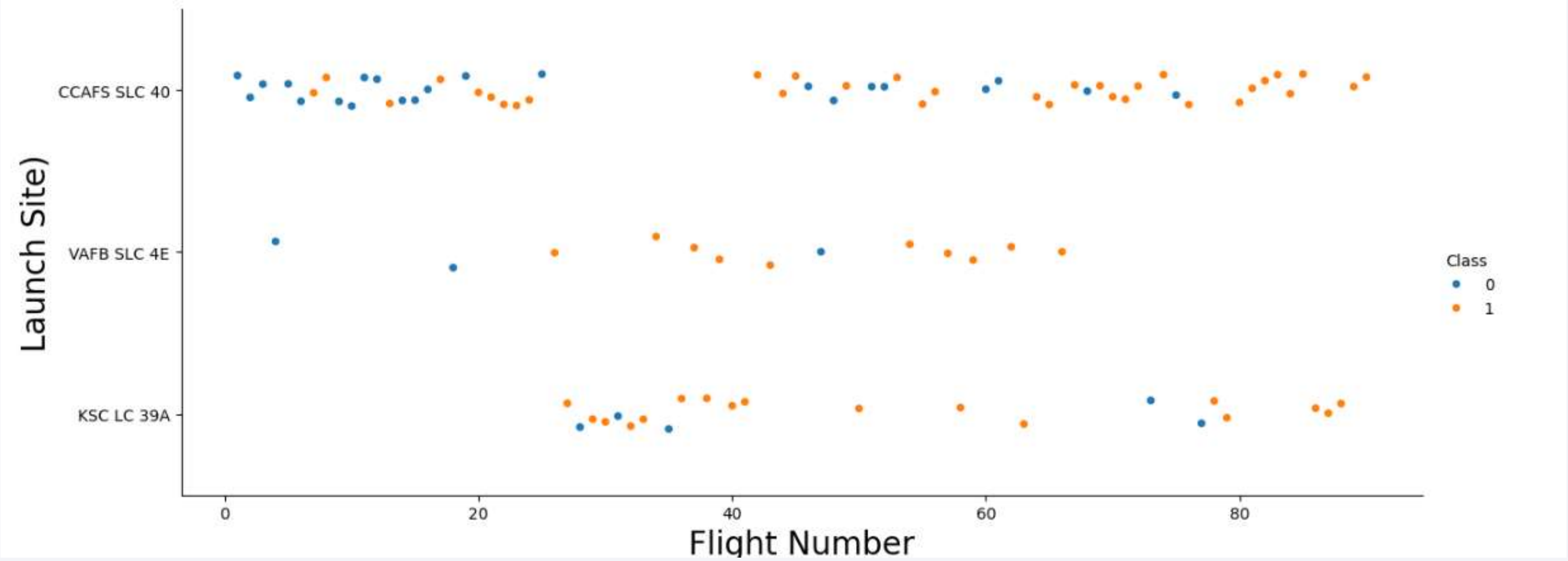
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

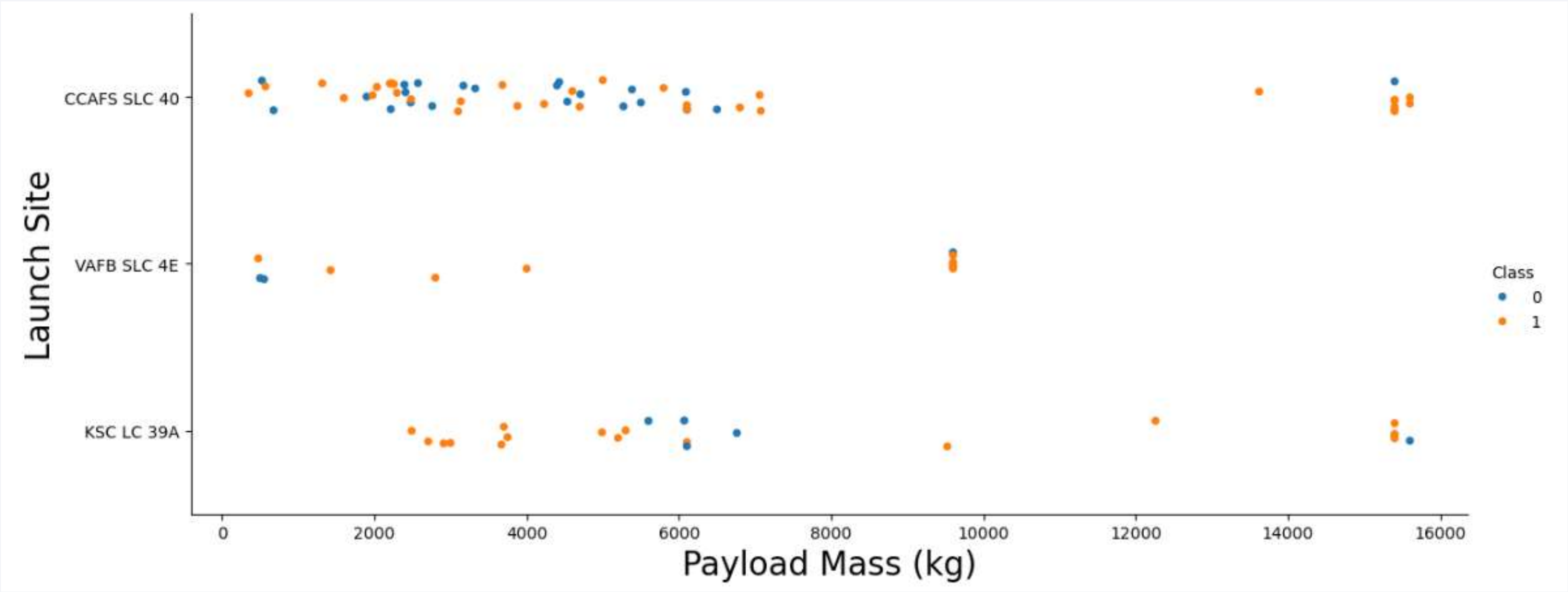
Flight Number vs. Launch Site

- Most initial launches were failures.
- The larger the number of launches, the greater success rate.



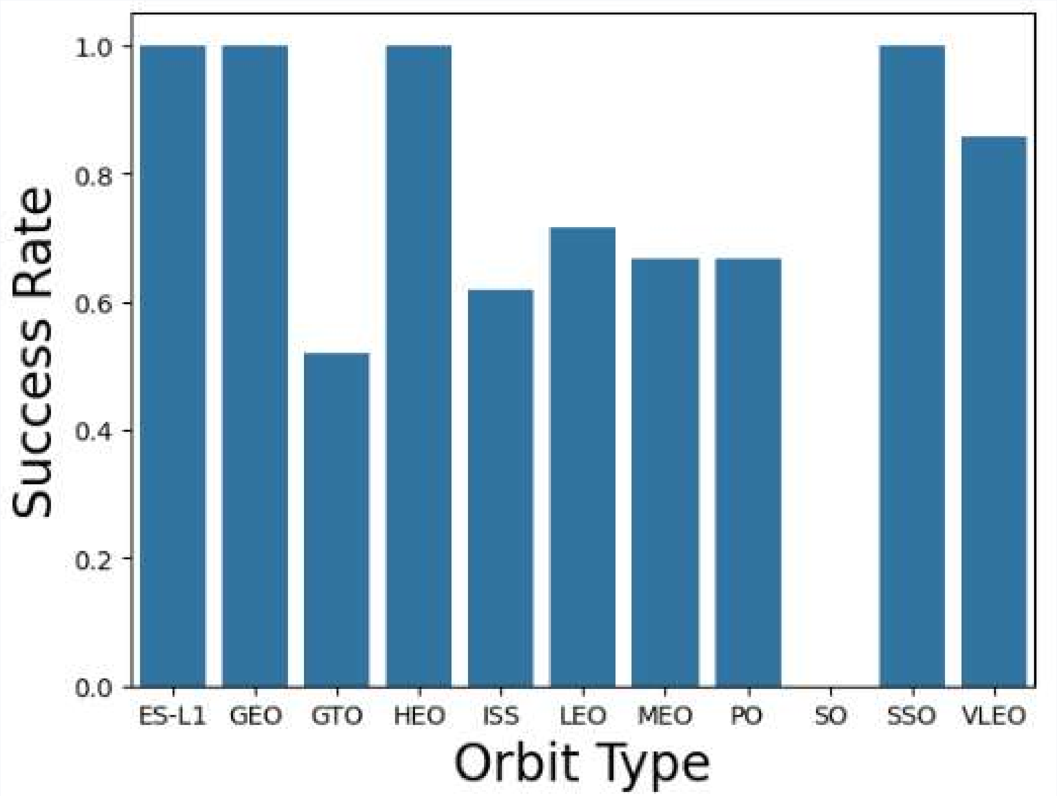
Payload vs. Launch Site

- Payloads greater than 7,000 kg present greater probability of success.
- No clear relationship between launch sites and payload.



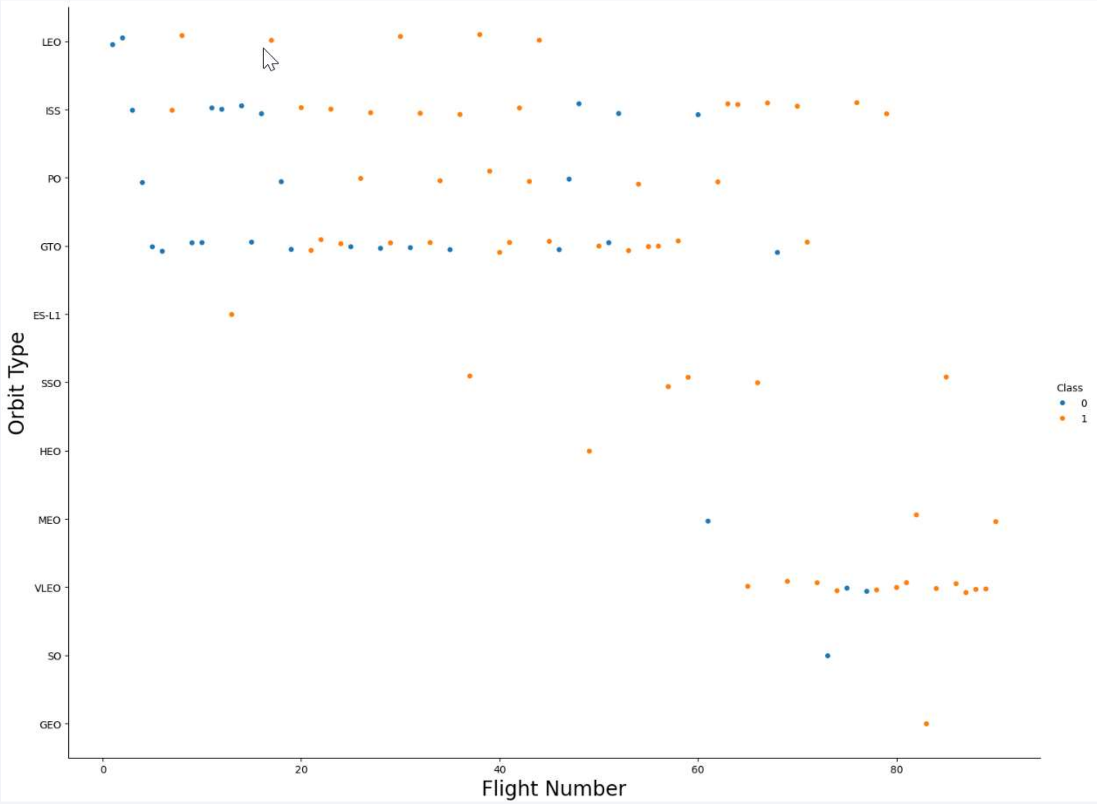
Success Rate vs. Orbit Type

- Success Rate is greater for ES-L1, GEO, HEO, and SSO orbit types



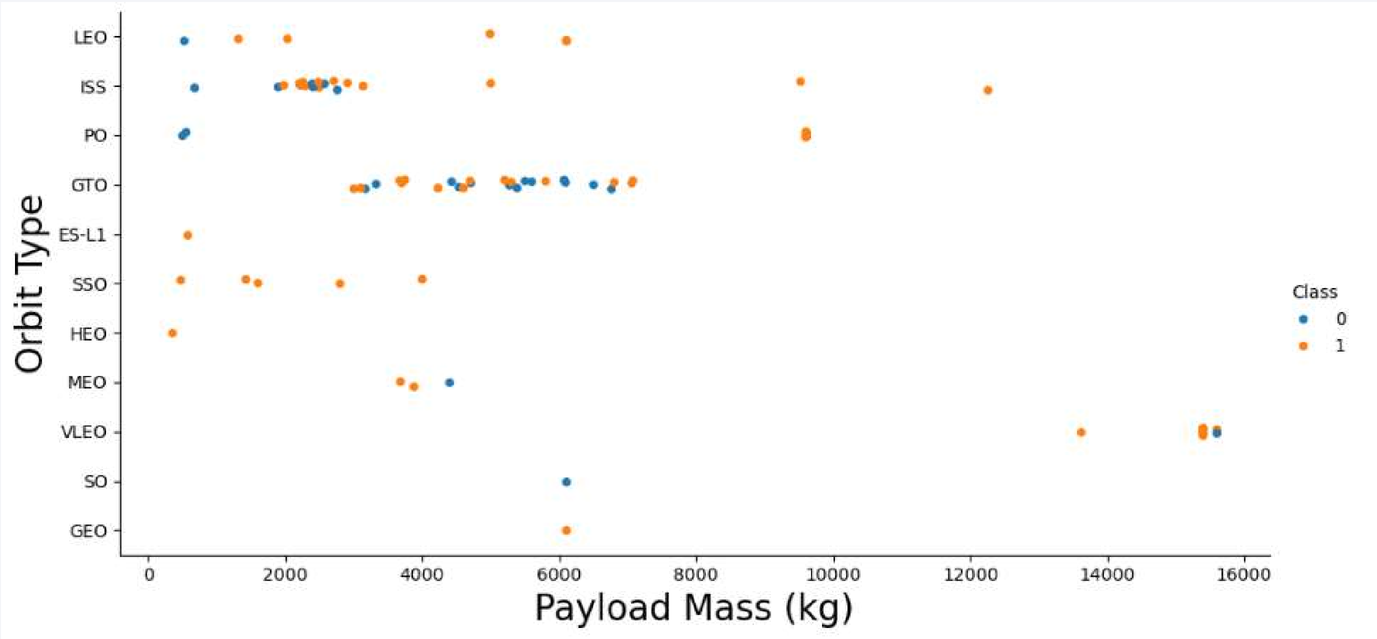
Flight Number vs. Orbit Type

- LEO orbit success rate appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.



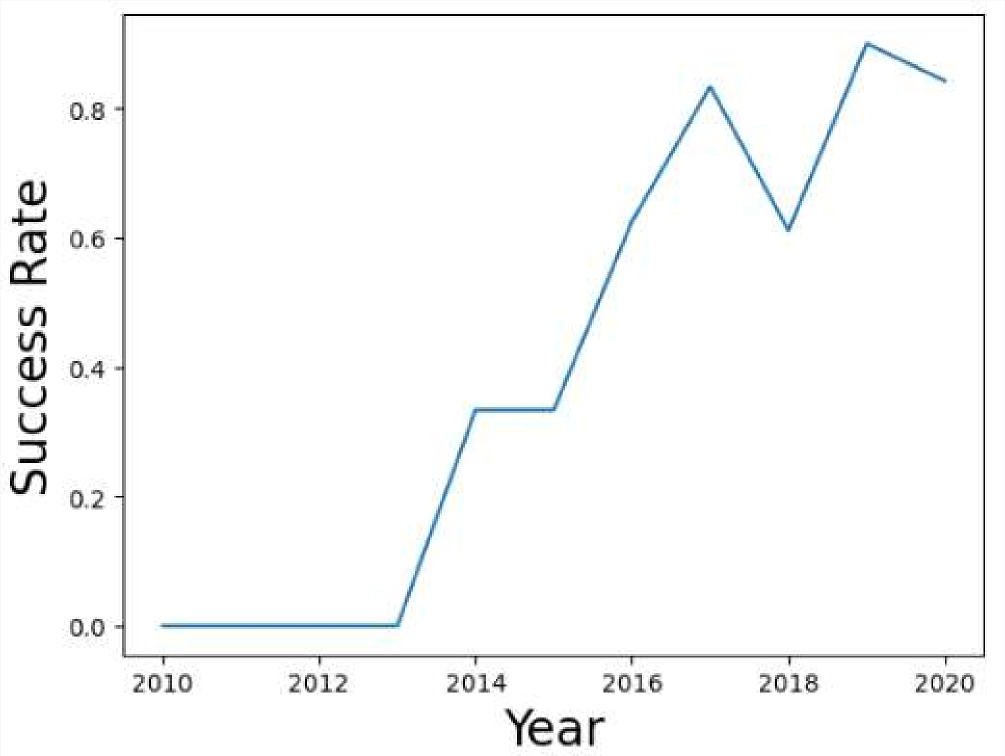
Payload vs. Orbit Type

- With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.
- For GTO we cannot distinguish this relationship as both positive and negative results are present for heavier payloads.



Launch Success Yearly Trend

- Increasing success rates since 2013.



All Launch Site Names

- Names of the unique launch sites

```
q = pd.read_sql('select distinct Launch_Site from spacexdata', conn)
q
```

	Launch_Site
0	CCAFS LC-40
1	VAFB SLC-4E
2	KSC LC-39A
3	CCAFS SLC-40

Launch Site Names Begin with 'CCA'

- Five records where launch sites begin with `CCA`

```
q = pd.read_sql("select * from spacexdata where Launch_Site like 'CCA%' limit 5", conn)
q
```

	index	Date	Time_(UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
0	0	None	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	1	None	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	2	None	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
3	3	None	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	4	None	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- Total payload carried by boosters from NASA

```
q = pd.read_sql("select sum(PAYLOAD_MASS__KG_) from spacexdata where Customer='NASA (CRS)'", conn)
q
```

	sum(PAYLOAD_MASS__KG_)
0	45596

Average Payload Mass by F9 v1.1

- Average payload mass carried by booster version F9 v1.1

```
q = pd.read_sql("select avg(PAYLOAD_MASS__KG_) from spacexdata where Booster_Version='F9 v1.1'", conn)
q
```

	avg(PAYLOAD_MASS__KG_)
0	2928.4

First Successful Ground Landing Date

- Dates of the first successful landing outcome on ground pad

Successful Drone Ship Landing with Payload between 4000 and 6000

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

```
q = pd.read_sql("select distinct Booster_Version from spacexdata where Landing_Outcome='Success (drone ship)' and PAYLOAD_MASS__KG_ between 4000 and 6000", conn)
q
```

	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

- Total number of successful and failure mission outcomes

```
q = pd.read_sql("select substr(Mission_Outcome,1,7) as Mission_Outcome, count(*) from spacexdata  group by 1", conn)
q
```

	Mission_Outcome	count(*)
0	Failure	1
1	Success	100

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass

```
q = pd.read_sql("select distinct Booster_Version from spacexdata where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from spacexdata)", conn)
q
```

	Booster_Version
0	F9 B5 B1048.4
1	F9 B5 B1049.4
2	F9 B5 B1051.3
3	F9 B5 B1056.4
4	F9 B5 B1048.5
5	F9 B5 B1051.4
6	F9 B5 B1049.5
7	F9 B5 B1060.2
8	F9 B5 B1058.3
9	F9 B5 B1051.6
10	F9 B5 B1060.3
11	F9 B5 B1049.7

2015 Launch Records

- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

```
q = pd.read_sql("select distinct Landing_Outcome, Booster_Version, Launch_Site from spacexdata where Landing_Outcome='Failure (drone ship)'", conn)
q
```

	Landing_Outcome	Booster_Version	Launch_Site
0	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
1	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40
2	Failure (drone ship)	F9 v1.1 B1017	VAFB SLC-4E
3	Failure (drone ship)	F9 FT B1020	CCAFS LC-40
4	Failure (drone ship)	F9 FT B1024	CCAFS LC-40

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

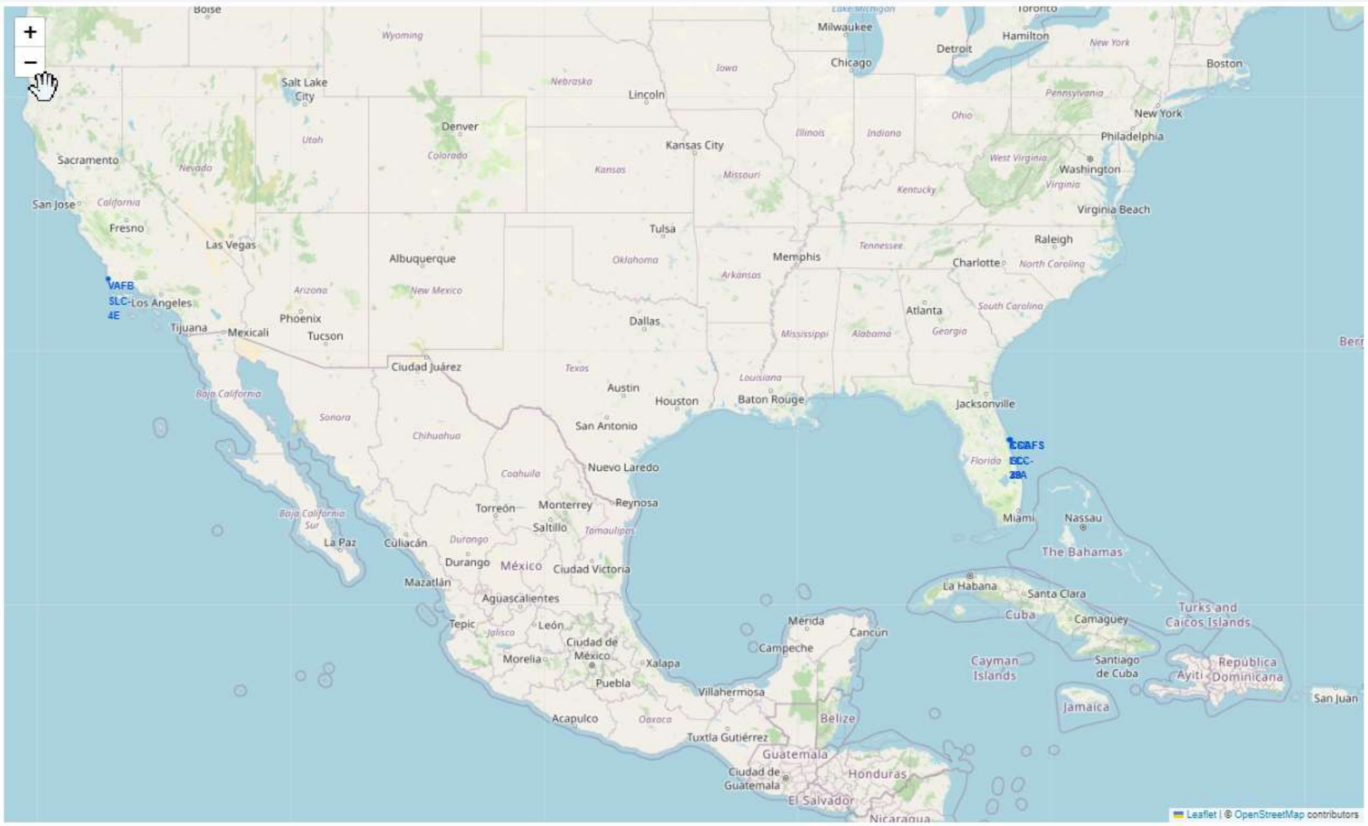
- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

Section 3

Launch Sites Proximities Analysis

Launch Sites Location Map

- Launch sites are located on the east and west coasts of the United States.



Detailed Launch Site Maps

- Launch sites with markers indicating successful and failure outcomes

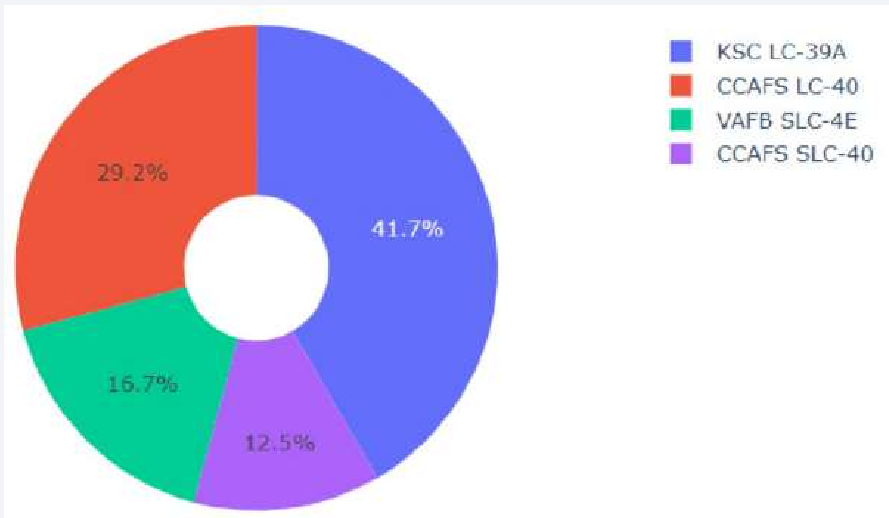
Distance from Launch Site to Coastline

Section 4

Build a Dashboard with Plotly Dash

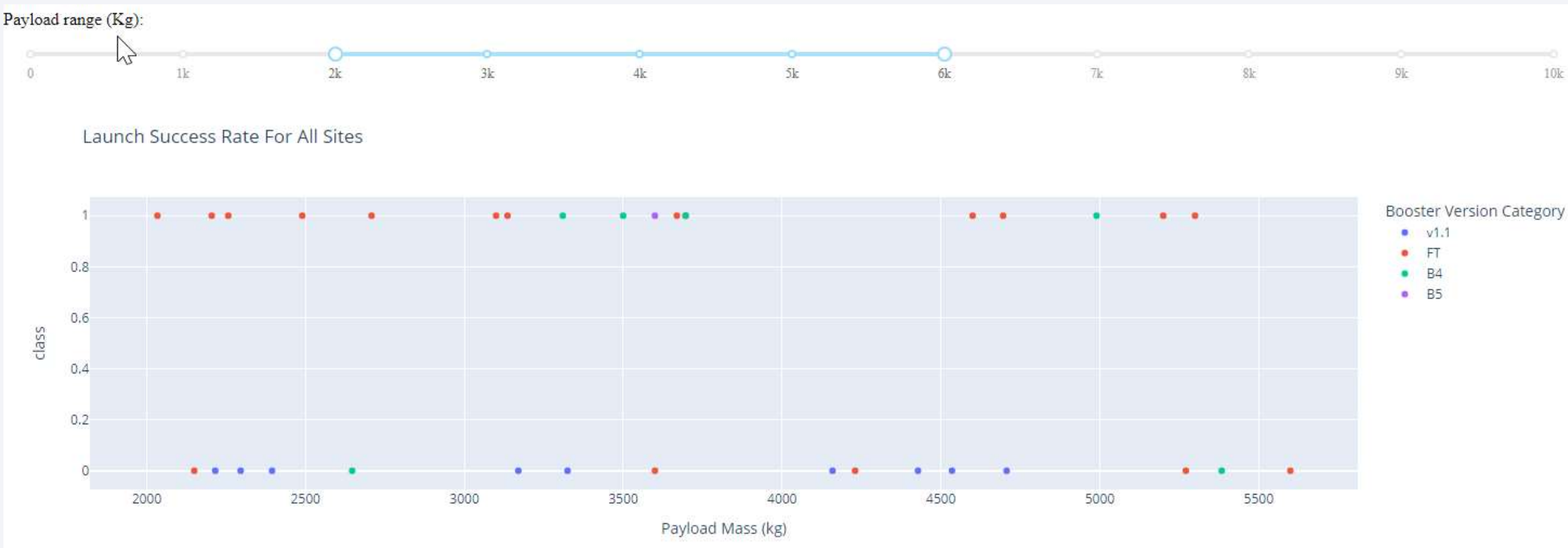
Launch Sites and Success Percentage

- KSC LC-39A has the highest rate of successful launches



Launch Outcome vs Payload

- Outcomes for different payload levels and ranges



Section 5

Predictive Analysis (Classification)

Classification Accuracy

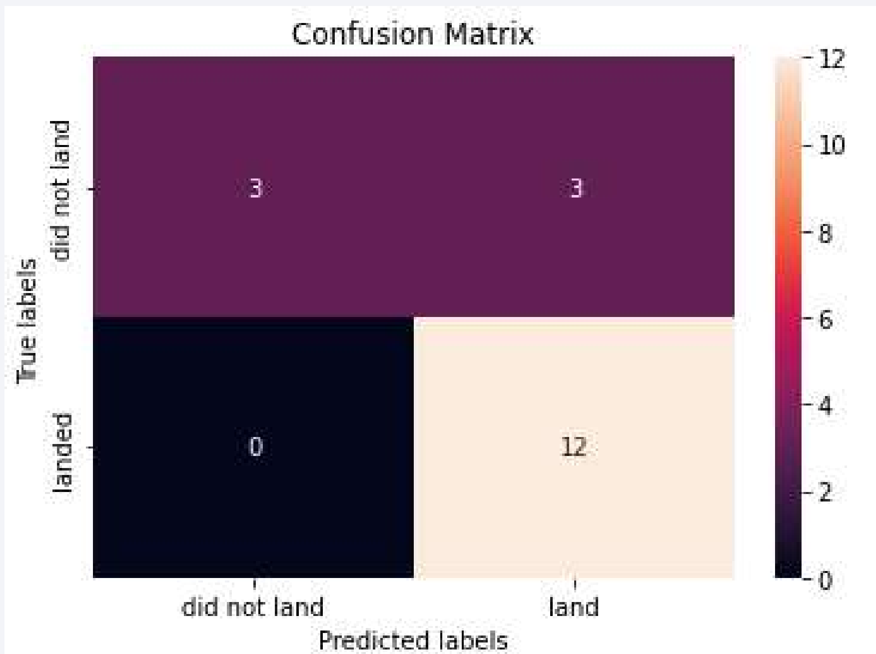
- Find which model has the highest classification accuracy

```
algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Tree':
    print('Best Params is :',tree_cv.best_params_)
if bestalgorithm == 'KNN':
    print('Best Params is :',knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best Params is :',logreg_cv.best_params_)
```

```
Best Algorithm is Tree with a score of 0.9017857142857142
Best Params is : {'criterion': 'entropy', 'max_depth': 10, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_spli
t': 10, 'splitter': 'random'}
```

Confusion Matrix

- The best performing model is the decision tree classifier.
- The confusion matrix shows the major problem is predicting unsuccessful landings as successful (false positives).



Conclusions

- The best classifier methodology for this dataset is Machine Learning approach.
- The heavy weighted payloads (>4,000kg) performed worse than the low weighted payloads.
- The success rate for SpaceX launches increased starting from 2013, showing progress that will eventually perfect the launches in the future.
- KSC LC-39A have the most successful launches of any sites; 76.9%
- SSO orbit have the most success rate; 100% and more than 1 occurrence.

Thank you!