IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Apar Prasad
Oct 19 2023

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection through API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Data Visualization
  - Interactive Visual Analytics with Folium
  - Machine Learning Prediction
- Summary of all results
  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - Predictive Analytics result

Introduction

- Project background and context
  - 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.
- Problems you want to find answers
  - What factors determine if the rocket will land successfully?
  - The interaction amongst various features that determine the success rate of a successful landing.
  - What operating conditions needs to be in place to ensure a successful landing program.

Section 1

Methodology

Methodology

Executive Summary

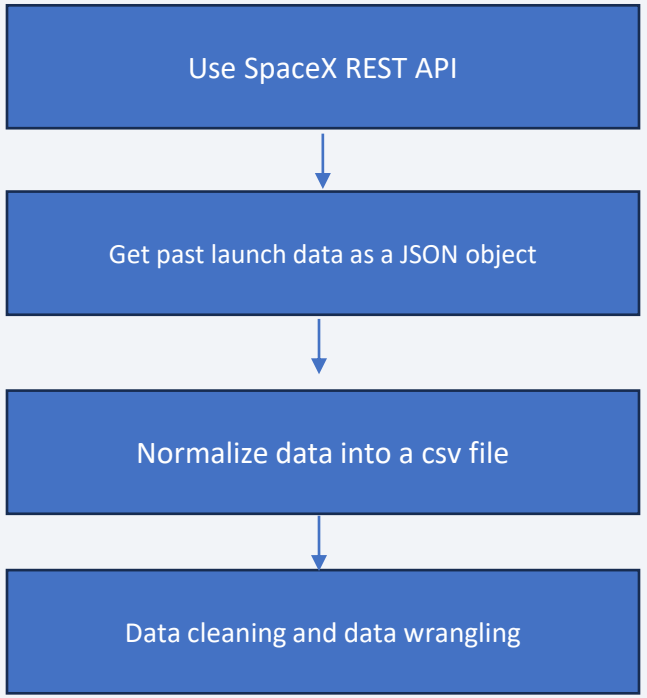
- Data collection methodology:
  - Data was collected using SpaceX API and web scraping from Wikipedia
- Perform data wrangling
  - One-hot encoding was applied to categorical features
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models

Data Collection

- Describe how data sets were collected.
  - Data collection was done using get request to the SpaceX API.
  - Next, we decoded the response content as a Json using .json() function call and turn it into a pandas dataframe using .json_normalize().
  - We then cleaned the data, checked for missing values and fill in missing values where necessary.
  - In addition, we performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.
  - The objective was to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for future analysis.

Data Collection – SpaceX API

Flow-chart



Code snippets

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

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

```
[12]: # Use json_normalize method to convert the json result into a dataframe

      # decode response content as json
      static_json_df = res.json()

[13]: # apply json_normalize
      data = pd.json_normalize(static_json_df)
```

```
[30]: rows = data_falcon9['PayloadMass'].values.tolist()[0]

      df_rows = pd.DataFrame(rows)
      df_rows = df_rows.replace(np.nan, PayloadMass)

      data_falcon9['PayloadMass'][0] = df_rows.values
      data_falcon9
```

NOTE: These snapshots do not contain the full code. For the full code, please refer to the GitHub code at https://github.com/apar20/IBM_Capstone/blob/main/Data%20Collection%20API.ipynb

Data Collection - Scraping

Flow-chart

Get HTML Response from Wikipedia

Extract data using BeautifulSoup

Extract all column names from HTML table header

Create a dataframe by parsing the launch HTML tables

Export data to csv

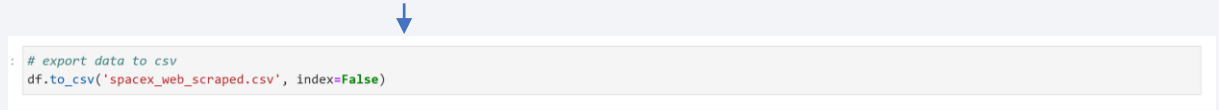
Code snippets

```
[4]: static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"

Next, request the HTML page from the above URL and get a response object

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

[5]: # use requests.get() method with the provided static_url
     # assign the response to a object
     html_data = requests.get(static_url)
     html_data.status_code

[5]: 200
```

```
[6]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
     soup = BeautifulSoup(html_data.text, 'html.parser')
```

```
[10]: column_names = []

      # Apply find_all() function with `th` element on first_launch_table
      # Iterate each th element and apply the provided extract_column_from_header() to get a column name
      # Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names

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

```
[12]: 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']=[]
```

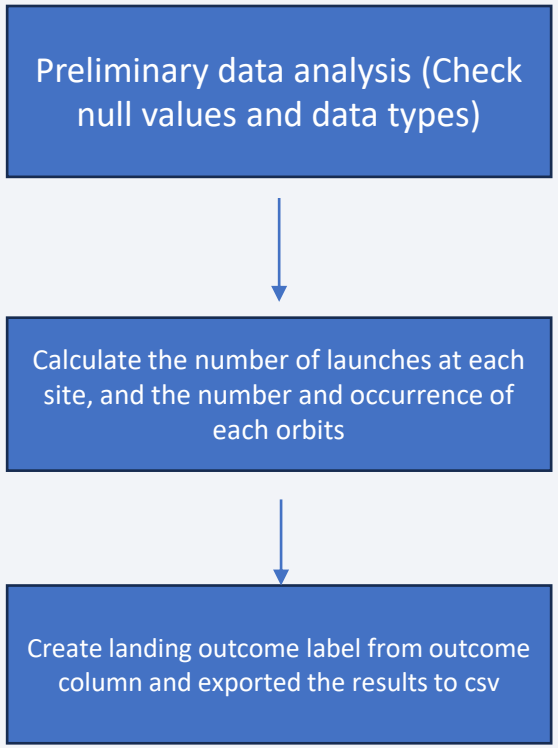


```
: # export data to csv
  df.to_csv('spacex_web_scraped.csv', index=False)
```

NOTE: These snapshots do not contain the full code. For the full code, please refer to the GitHub code at https://github.com/apar20/IBM_Capstone/blob/main/Data%20Collection%20with%20Web%20Scraping.ipynb

Data Wrangling

Flow-chart



Code snippets

```
[3]: # check for null values
     df.isnull().sum()/df.count()*100
```

```
[4]: # check data types
     df.dtypes
```

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

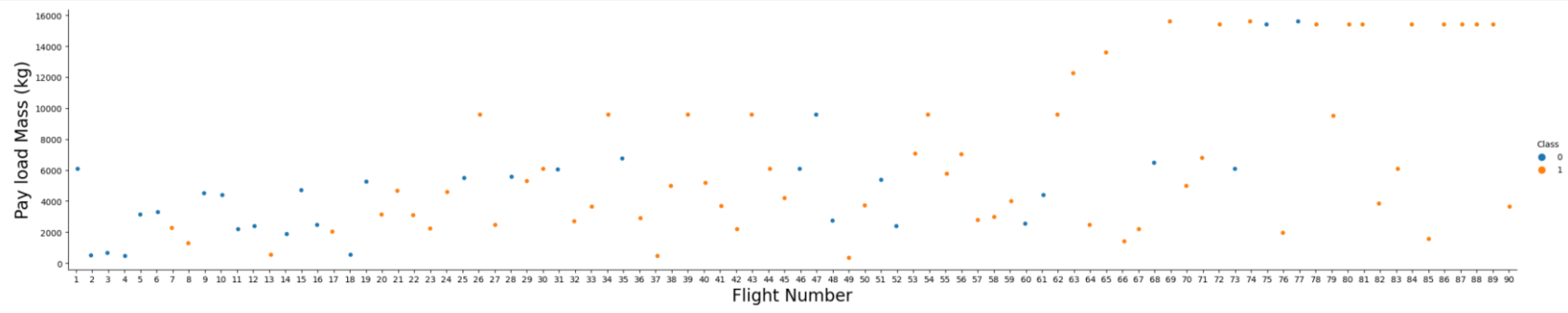
```
[6]: # Apply value_counts on Orbit column
     df['Orbit'].value_counts()
```

```
[10]: # landing_class = 0 if bad_outcome
      # landing_class = 1 otherwise
      #landing_class = [x for x in bad_outcomes if df['Outcome'][x] ]

      landing_class = []

      for key, value in df['Outcome'].items():
          if value in bad_outcomes:
              landing_class.append(0)
          else:
              landing_class.append(1)
```


NOTE: These snapshots do not contain the full code. For the full code, please refer to the GitHub code at https://github.com/apar20/IBM_Capstone/blob/main/Data%20Wrangling.ipynb

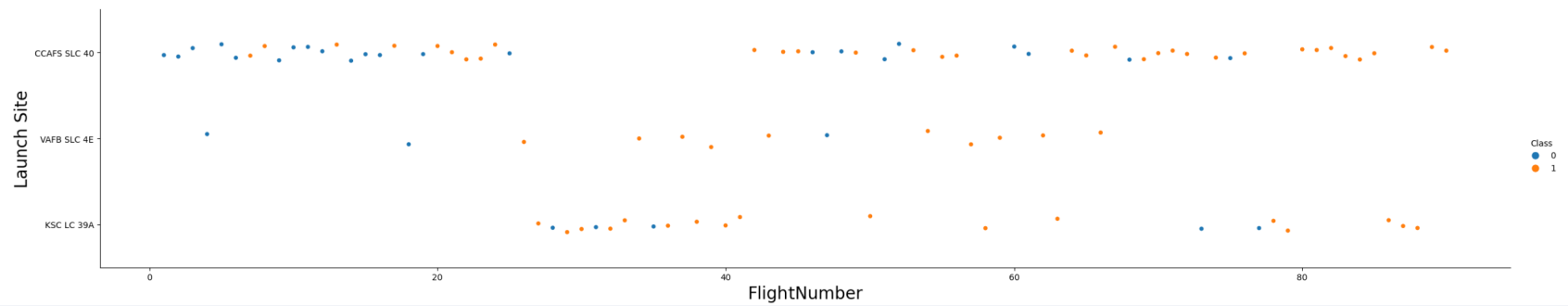
EDA with Data Visualization

Flight number and Payload



Legend
Orange: Successful
Blue: Failure

Flight number and Launch Site

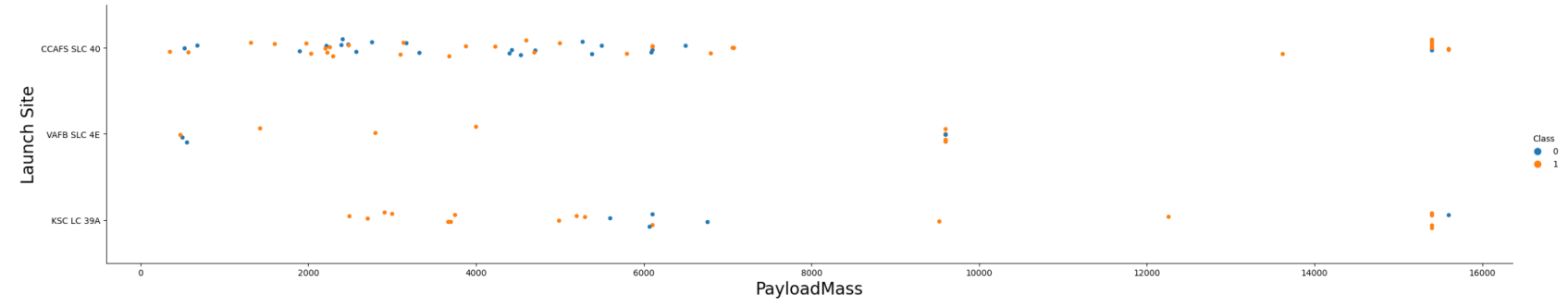


Legend
Orange: Successful
Blue: Failure

GitHub code located at: https://github.com/apar20/IBM_Capstone/blob/main/Data%20Collection%20API.ipynb

EDA with Data Visualization

Payload and Launch Site

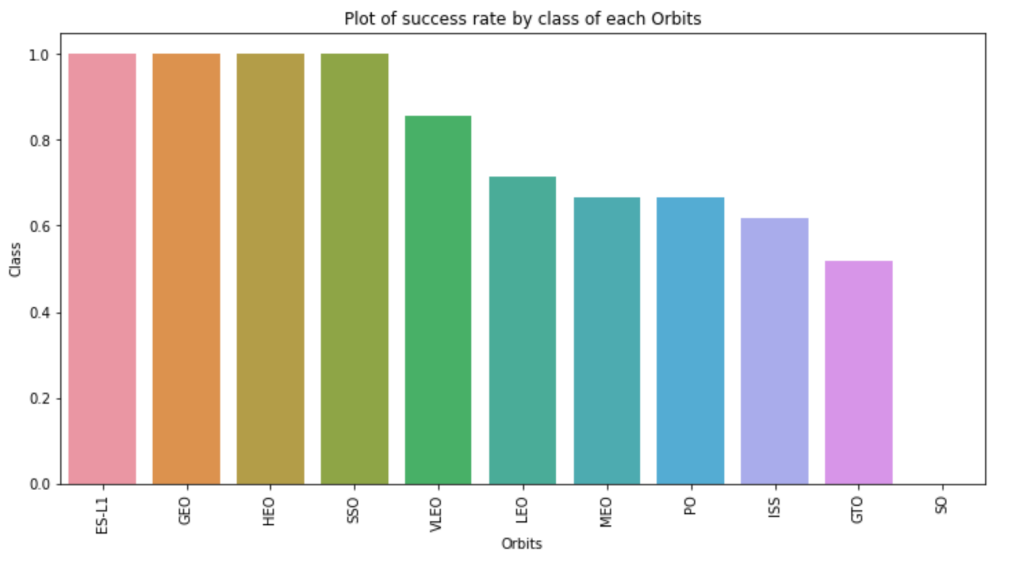


Legend

Orange: Successful

Blue: Failure

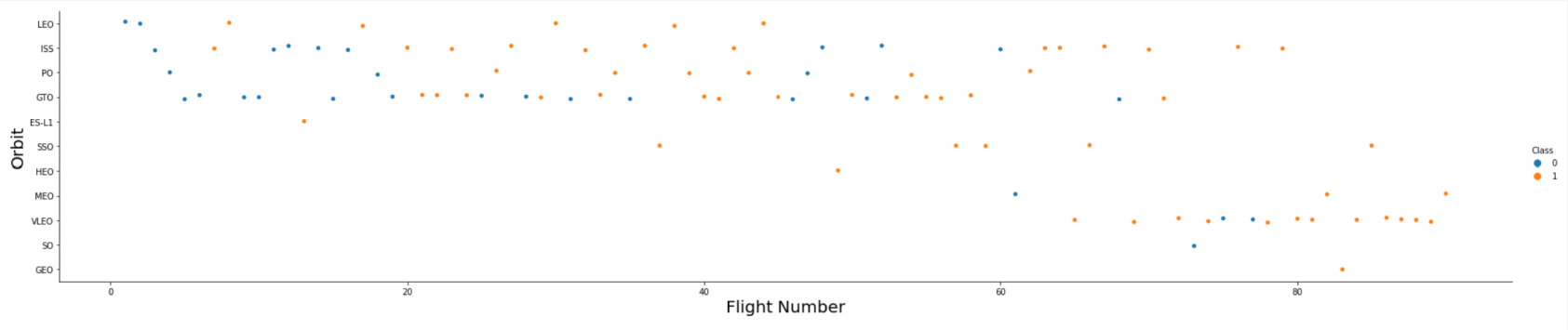
Success rate of each orbit type



GitHub code located at: https://github.com/apar20/IBM_Capstone/blob/main/Data%20Collection%20API.ipynb

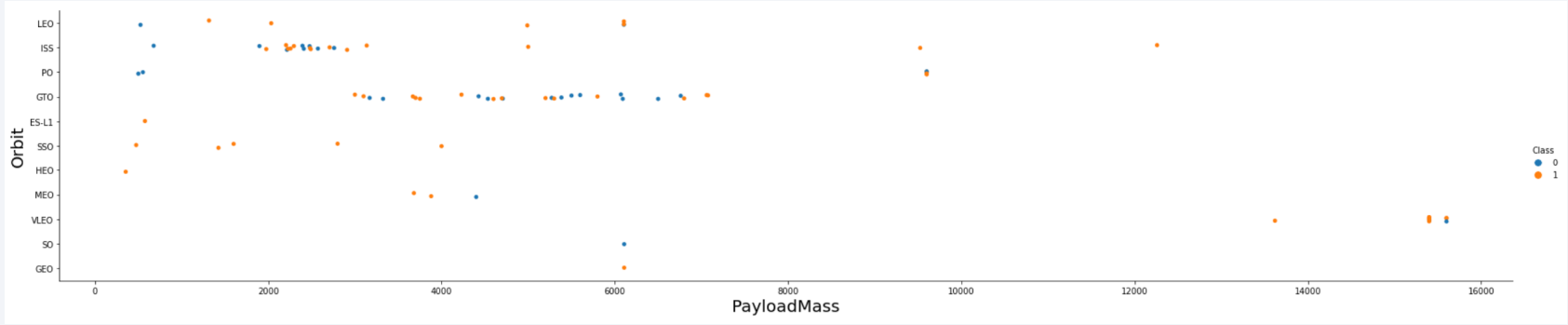
EDA with Data Visualization

Flight number and orbit type



Legend
Orange: Successful
Blue: Failure

Payload and orbit type

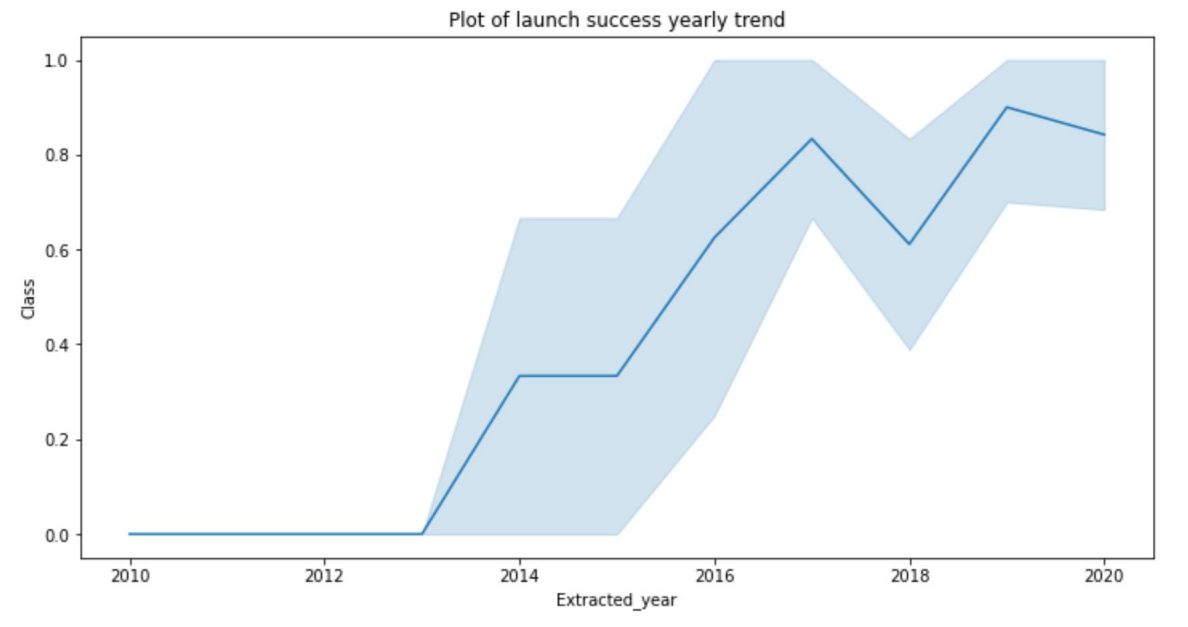


Legend
Orange: Successful
Blue: Failure

GitHub code located at: https://github.com/apar20/IBM_Capstone/blob/main/Data%20Collection%20API.ipynb

EDA with Data Visualization

Launch success yearly trend



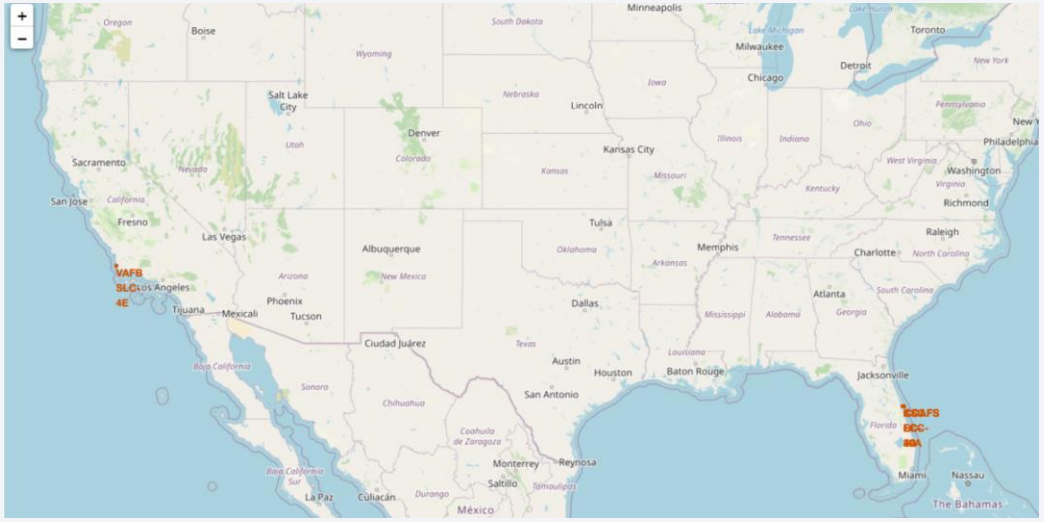
GitHub code located at: https://github.com/apar20/IBM_Capstone/blob/main/Data%20Collection%20API.ipynb

EDA with SQL

- We wrote the following SQL queries to find out for instance:
  - The names of unique launch sites in the space mission.
  - Launch sites begin with the string 'CCA'
  - The total payload mass carried by boosters launched by NASA (CRS)
  - The average payload mass carried by booster version F9 v1.1
  - The total number of successful and failure mission outcomes
  - The failed landing outcomes in drone ship, their booster version and launch site names

GitHub code at: https://github.com/apar20/IBM_Capstone/blob/main/EDA%20with%20SQL%20.ipynb

Build an Interactive Map with Folium

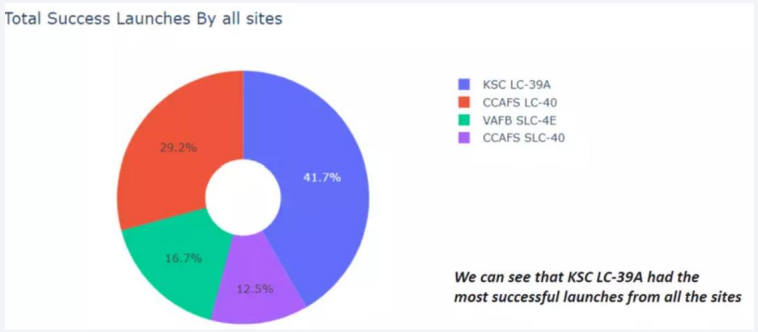


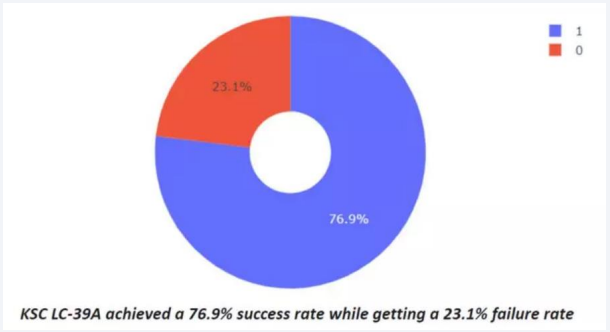
- Marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map
- Assigned the feature launch outcomes (failure or success) to class 0 and 1 (0 for failure, and 1 for success)
- Identified which launch sites have relatively high success rate using the color-labeled marker clusters
- Calculated the distances between a launch site to its proximities. We answered some question for instance:
  - Are launch sites near railways, highways and coastlines.
  - Do launch sites keep certain distance away from cities.

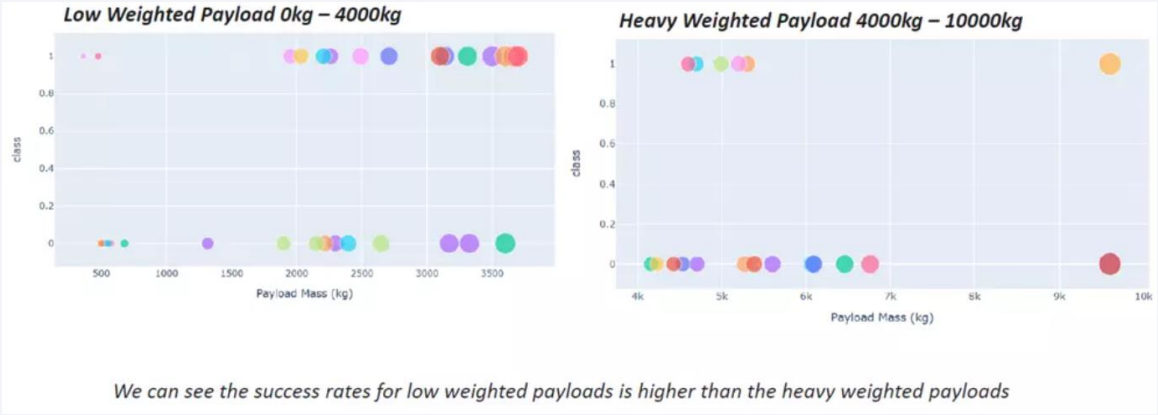
GitHub code located at: https://github.com/apar20/IBM_Capstone/blob/main/Interactive%20Map%20with%20Folium.ipynb

Build a Dashboard with Plotly Dash

- Built an interactive dashboard with Plotly dash
- Plotted pie charts showing the total launches by a certain sites
- Plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version







GitHub code located at: https://github.com/apar20/IBM_Capstone/blob/main/Dashboard%20with%20Plotly%20Dash.py

Predictive Analysis (Classification)

Split label column 'Class' from dataset

Fit and Transform Features using StandardScaler

Train_test_split data

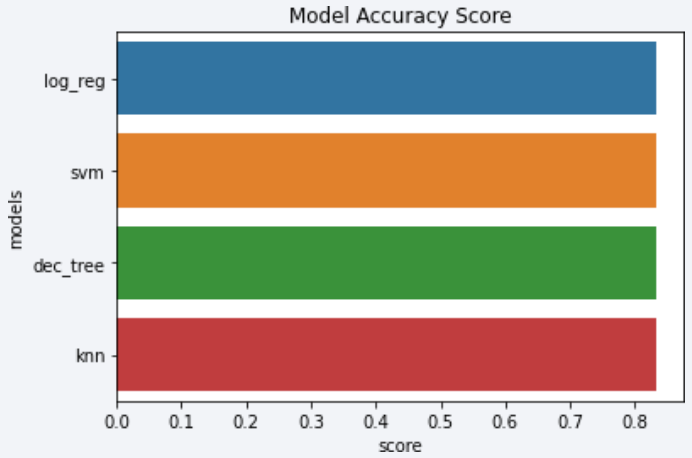
GridSearchCV (cv=10) to find optimal parameters

Use GridSearchCV on Logistic Regression, SVM, Decision Tree and KNN models

Score models on split testset

Confusion Matrix for all models

Barplot to compare scores of all models

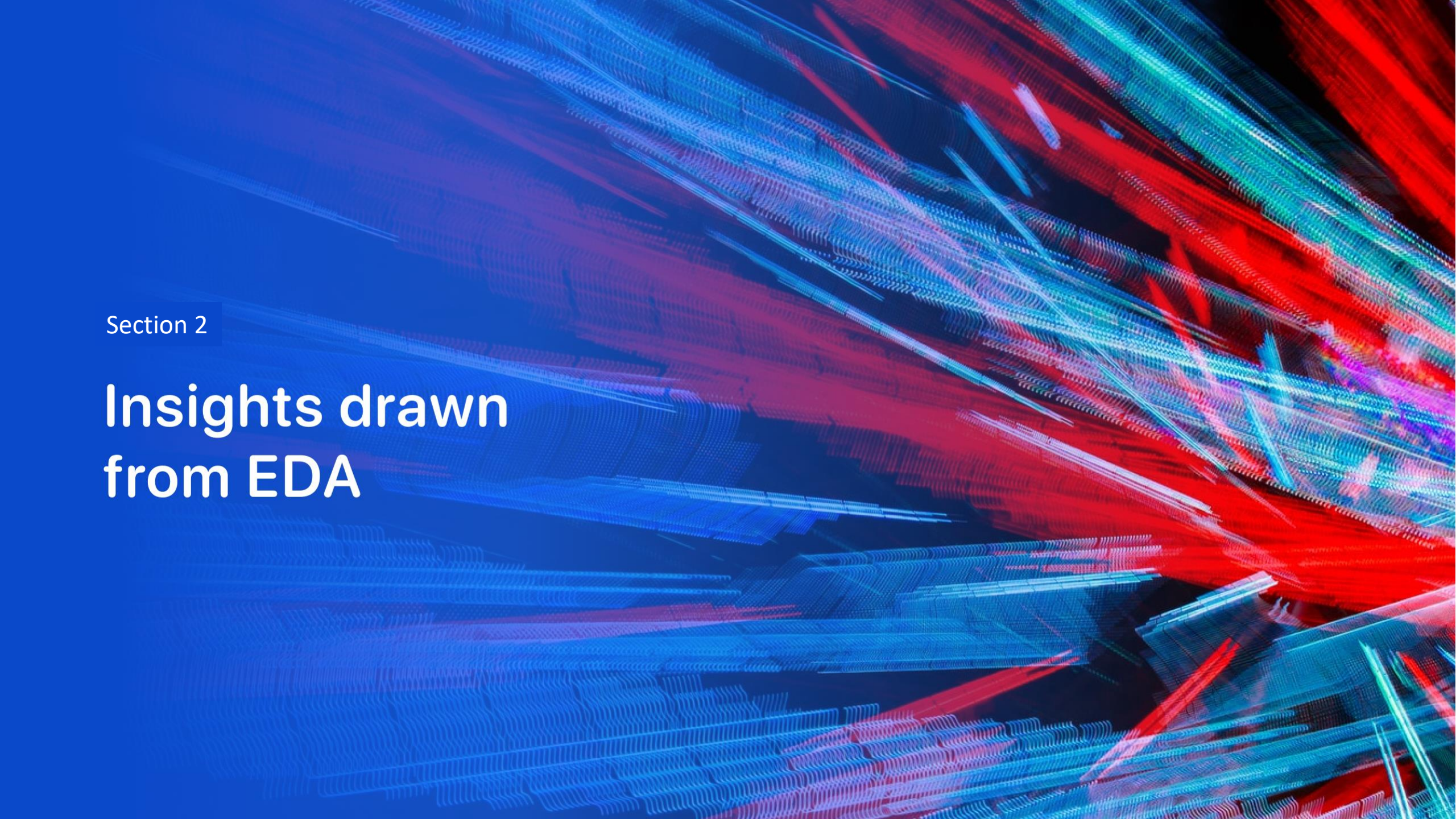


Since all models performed the same for the test set, the confusion matrix is the same across all models

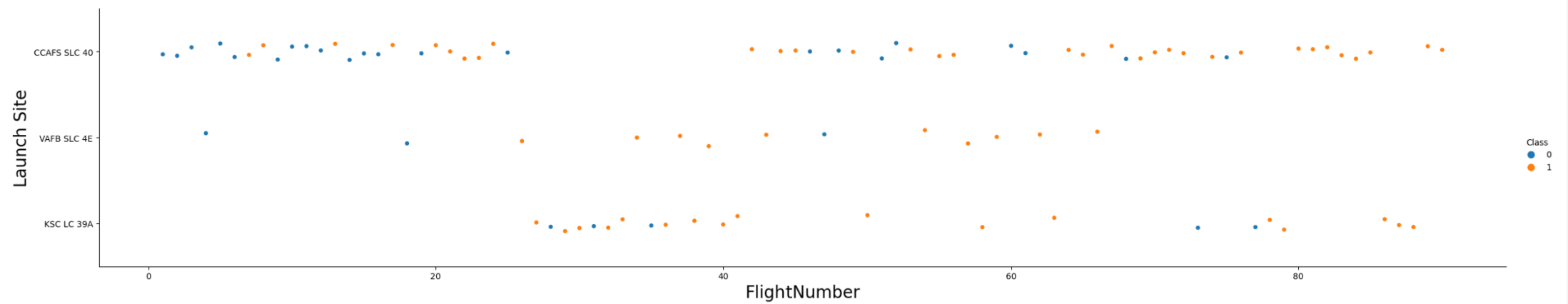
GitHub code available at: https://github.com/apar20/IBM_Capstone/blob/main/Predictive%20Analysis%20(Classification).ipynb

Results

- Low weighted payloads perform better than heavy payloads
- As the success rate of SpaceX launches is increasing, their future launches are likely to be successful
- Orbits ESL1, SSO, GEO and HEO are the most suitable target orbits
- KSC LC-39A is the most suitable site for the next launch
- Any of SVM, KNN, Decision Tree and Logistic Regression model may be used to predict the accuracy of this dataset


Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



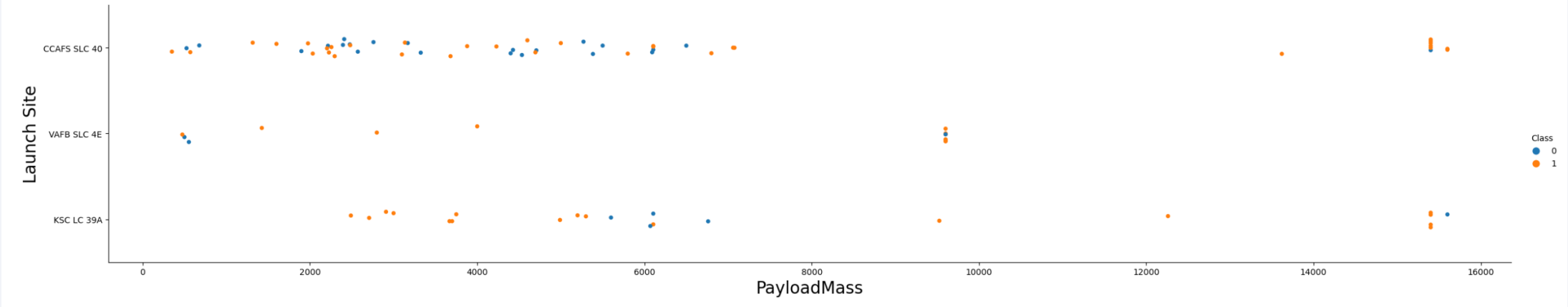
Legend

Orange: Successful

Blue: Failure

From the plot, we found that the larger the flight amount at a launch site, the greater the success rate at a launch site.

Payload vs. Launch Site



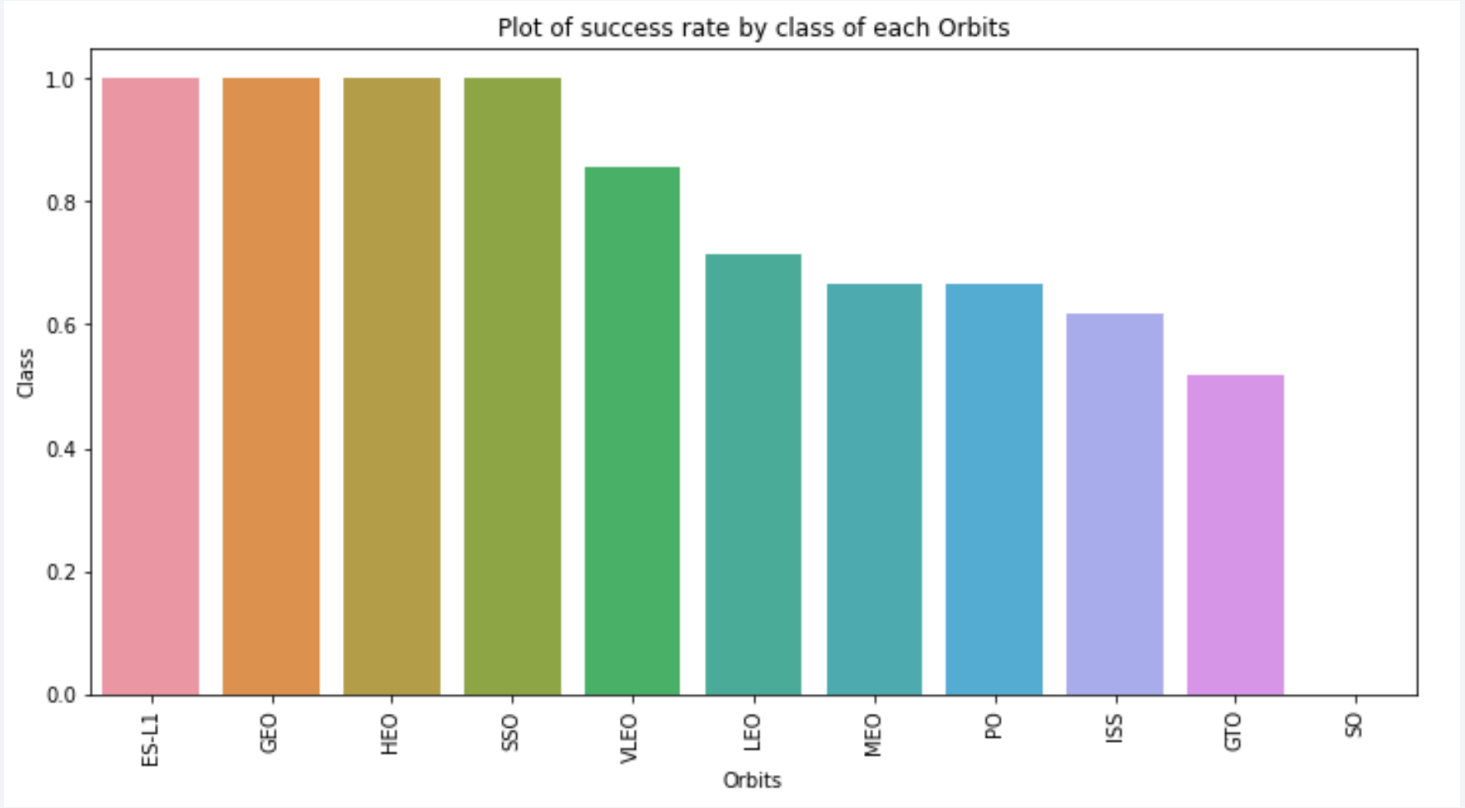
Legend

Orange: Successful

Blue: Failure

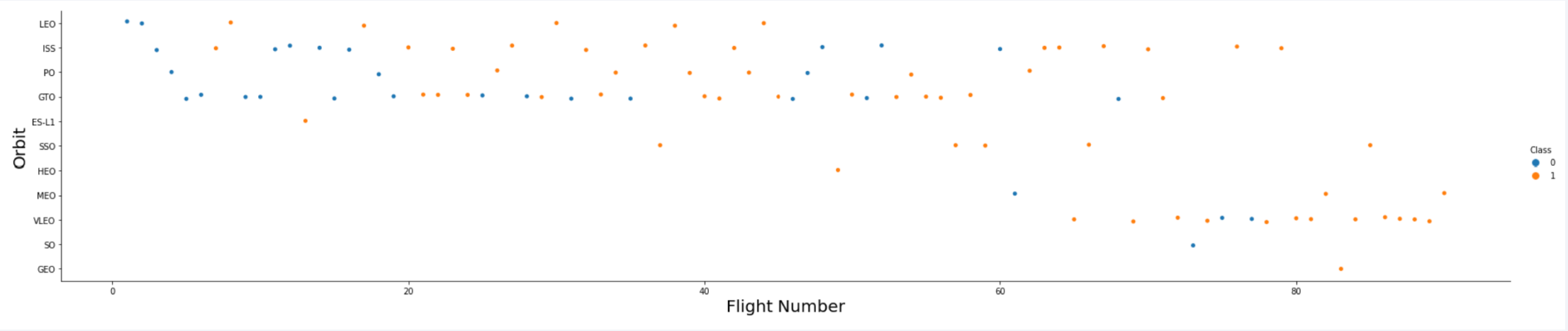
Payload mass appears to fall mostly between 0-6000 kg

Success Rate vs. Orbit Type



ESL1, GEO, HEO and SSO have 100% success rate making them ideal targets for launch

Flight Number vs. Orbit Type



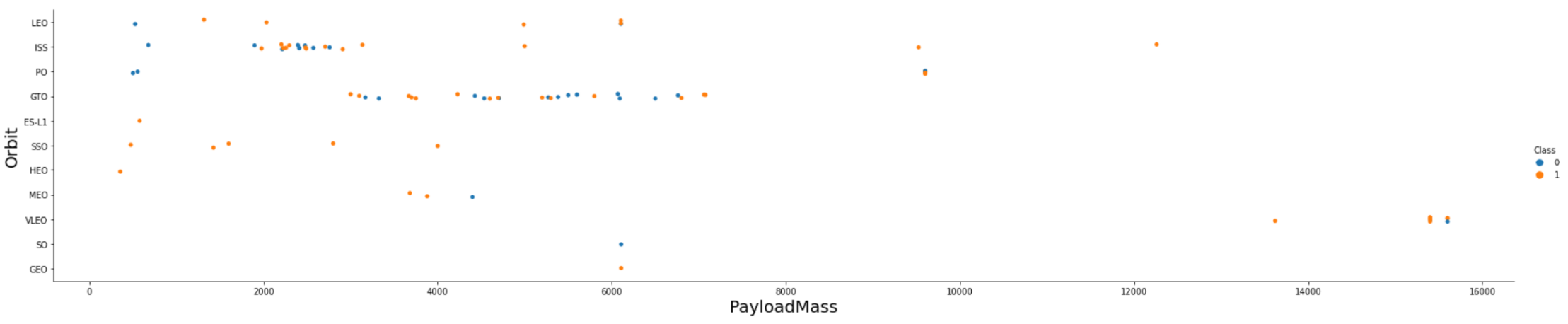
Legend

Orange: Successful

Blue: Failure

- Launch Orbit preferences changed over Flight Number. Launch Outcome seems to correlate with this preference
- SpaceX appears to perform better in lower orbits or Sun-synchronous orbits

Payload vs. Orbit Type



Legend

Orange: Successful

Blue: Failure

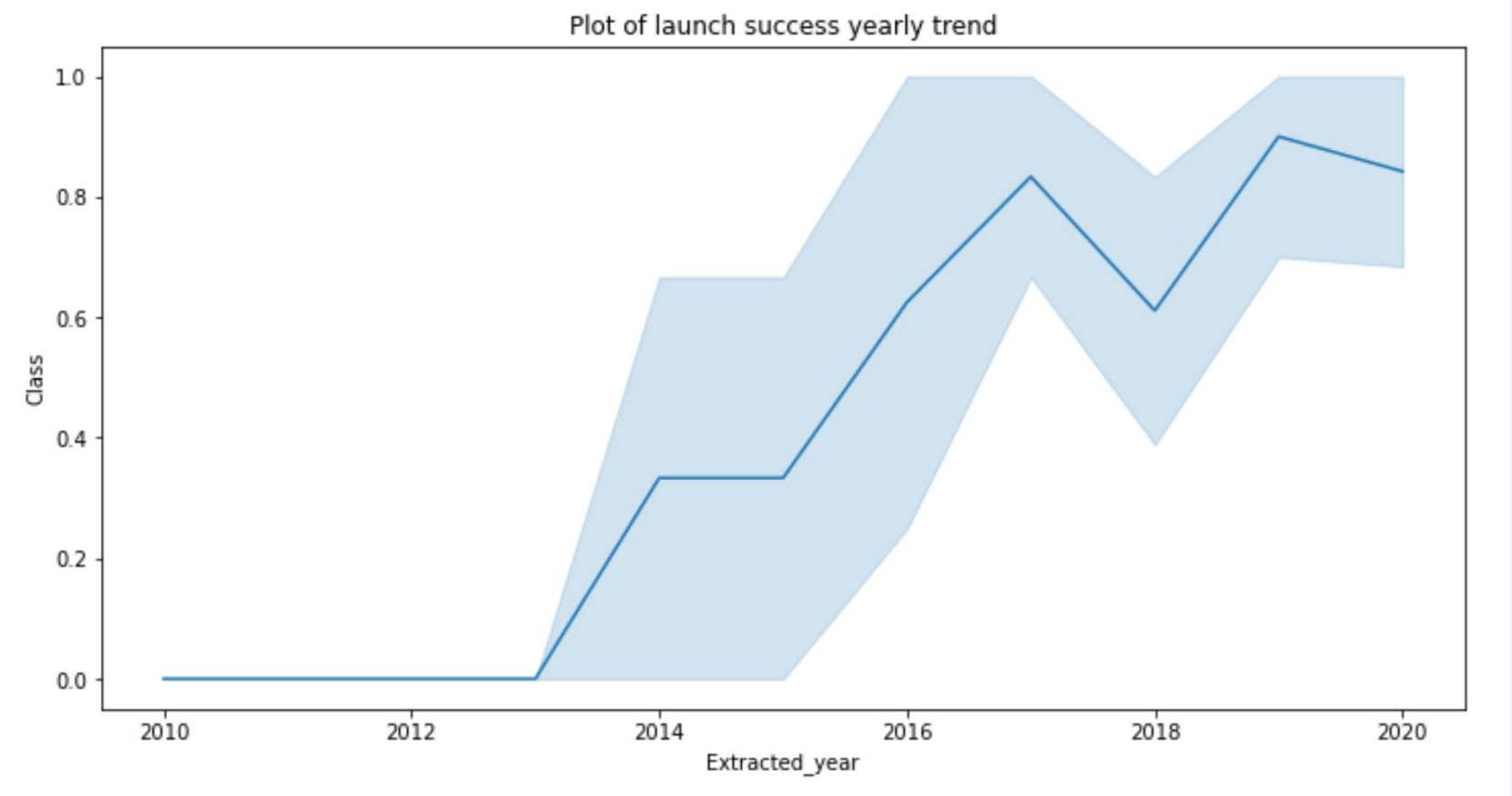
Payload mass seems to correlate with orbit

ES-L1, HEO, MEO and SSO seem to have relatively low payload mass

VLEO only has payload mass values in the higher end of the range

All other orbits have payloads in the low to medium range

Launch Success Yearly Trend



- Success rate has considerably increased since 2013 with a slight dip in 2018
- Success rate in recent years is ~80%

All Launch Site Names

```
[10]: task_1 = '''
        SELECT DISTINCT LaunchSite
        FROM SpaceX
      '''
      create_pandas_df(task_1, database=conn)
```

```
[10]: ,,,,,,,,,,,,,,,,,,,,,,,,,,
```

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

We used the key word **DISTINCT** to show only unique launch sites from the SpaceX data

Launch Site Names Begin with 'CCA'

```
[11]: task_2 = '''
        SELECT *
        FROM SpaceX
        WHERE LaunchSite LIKE 'CCA%'
        LIMIT 5
        '''
create_pandas_df(task_2, database=conn)
```

[11]: ,,

	date	time	boosterversion	launchsite	payload	payloadmasskg	orbit	customer	missionoutcome	landingoutcome
0	2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	2010-08-12	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	2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
3	2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

[12]:
```
task_3 = '''
        SELECT SUM(PayloadMassKG) AS Total_PayloadMass
        FROM SpaceX
        WHERE Customer LIKE 'NASA (CRS)'
        '''
create_pandas_df(task_3, database=conn)
```

[12]: ,,,,,,,,,,,,,,,,

	total_payloadmass
0	45596

Average Payload Mass by F9 v1.1

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

[13]:
```
task_4 = '''
        SELECT AVG(PayloadMassKG) AS Avg_PayloadMass
        FROM SpaceX
        WHERE BoosterVersion = 'F9 v1.1'
        '''
create_pandas_df(task_4, database=conn)
```

[13]: ,,,,,,,,,,,,,,,

	avg_payloadmass
0	2928.4

First Successful Ground Landing Date

Task 5

List the date when the first successful landing outcome in ground pad was acheived.

Hint:Use min function

[14]:
```
task_5 = '''
        SELECT MIN(Date) AS FirstSuccessfull_landing_date
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Success (ground pad)'
        '''
create_pandas_df(task_5, database=conn)
```

[14]: ,,,,,,,,,,,,,,,

	firstsuccessfull_landing_date
0	2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

Task 6

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

[15]:
```
task_6 = '''
        SELECT BoosterVersion
        FROM SpaceX
        WHERE LandingOutcome = 'Success (drone ship)'
            AND PayloadMassKG > 4000
            AND PayloadMassKG < 6000
        '''
create_pandas_df(task_6, database=conn)
```

[15]: ''''''''''''''''''''''''''

	boosterversion
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

Task 7

List the total number of successful and failure mission outcomes

[16]:
```
task_7a = '''
        SELECT COUNT(MissionOutcome) AS SuccessOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Success%'
        '''

task_7b = '''
        SELECT COUNT(MissionOutcome) AS FailureOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Failure%'
        '''
print('The total number of successful mission outcome is:')
display(create_pandas_df(task_7a, database=conn))
print()
print('The total number of failed mission outcome is:')
create_pandas_df(task_7b, database=conn)
```

```
The total number of successful mission outcome is:
,,,,,,,,,,,,,,
```

	successoutcome
0	100

```
,


The total number of failed mission outcome is:
,,,,,,,,,,,,,,
```

[16]:

	failureoutcome
0	1

Boosters Carried Maximum Payload

Task 8

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

[17]:
```
task_8 = '''
        SELECT BoosterVersion, PayloadMassKG
        FROM SpaceX
        WHERE PayloadMassKG = (
                                SELECT MAX(PayloadMassKG)
                                FROM SpaceX
                                )
        ORDER BY BoosterVersion
        '''
create_pandas_df(task_8, database=conn)
```

[17]:

	boosterversion	payloadmasskg
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
5	F9 B5 B1051.3	15600
6	F9 B5 B1051.4	15600
7	F9 B5 B1051.6	15600
8	F9 B5 B1056.4	15600
9	F9 B5 B1058.3	15600
10	F9 B5 B1060.2	15600
11	F9 B5 B1060.3	15600

2015 Launch Records

Task 9

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

```
task_9 = '''
        SELECT BoosterVersion, LaunchSite, LandingOutcome
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Failure (drone ship)'
            AND Date BETWEEN '2015-01-01' AND '2015-12-31'
        '''
create_pandas_df(task_9, database=conn)
```

```
,,,,,,,,,,,,,,,,,,,,,,,,,
```

	boosterversion	launchsite	landingoutcome
0	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
1	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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

```
task_10 = '''
        SELECT LandingOutcome, COUNT(LandingOutcome)
        FROM SpaceX
        WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
        GROUP BY LandingOutcome
        ORDER BY COUNT(LandingOutcome) DESC
        '''
create_pandas_df(task_10, database=conn)
```

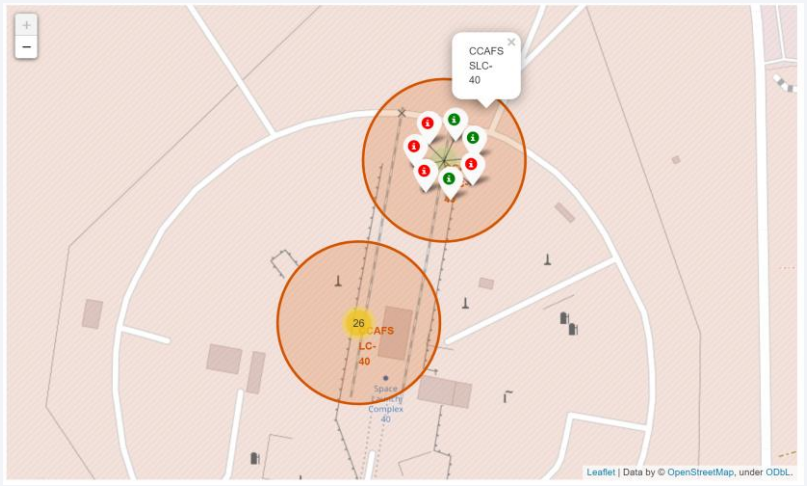
Section 3

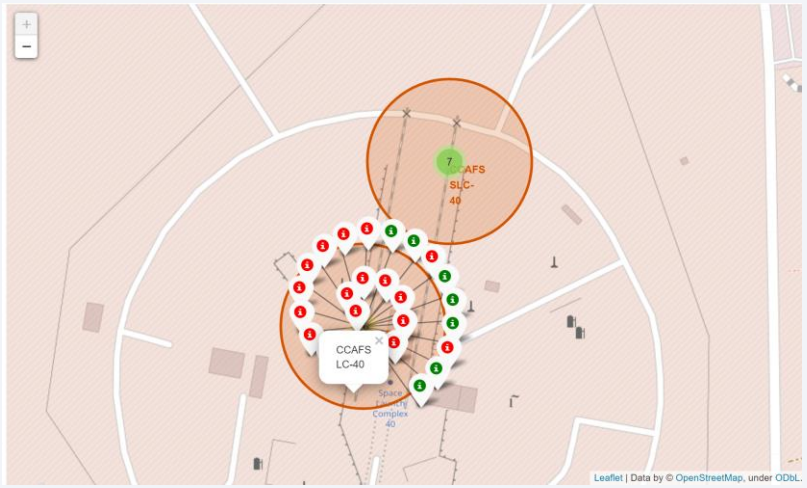
Launch Sites Proximities Analysis

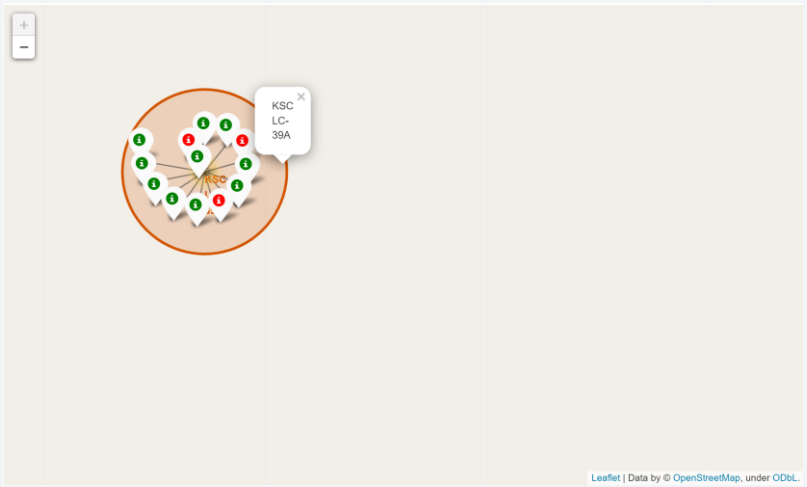
All launch sites global map markers

Launch sites with outcomes

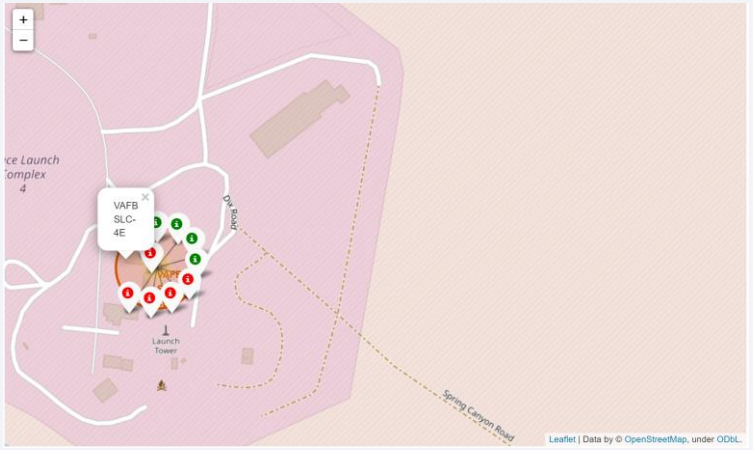
Florida sites







California sites



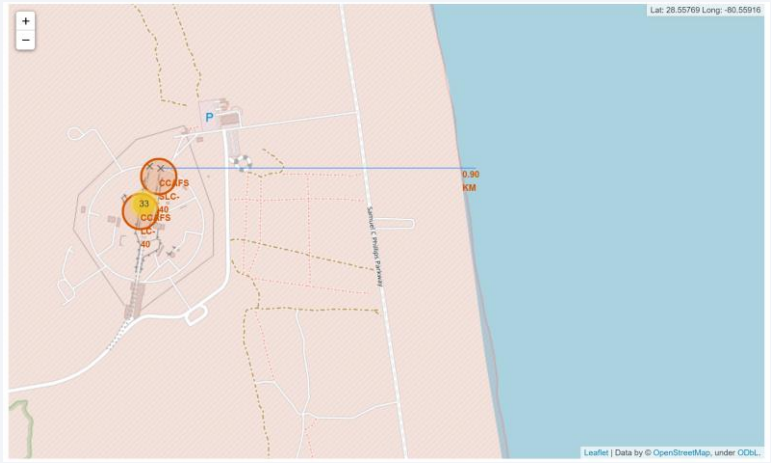
Legend

Green: Successful

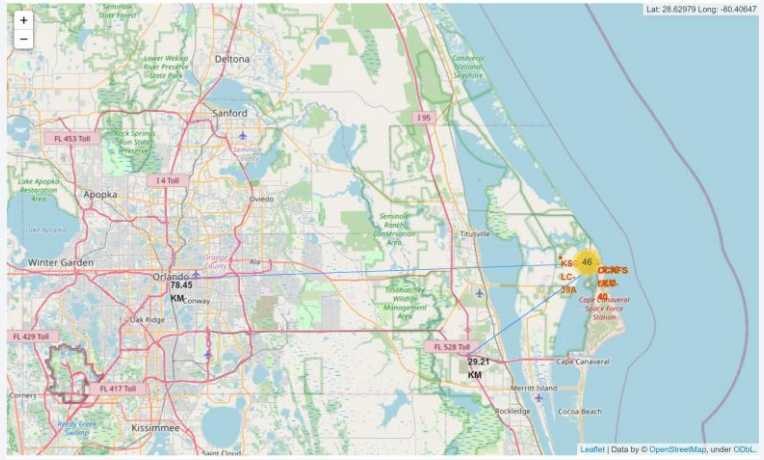
Red: Failure

Launch Site distance to landmarks

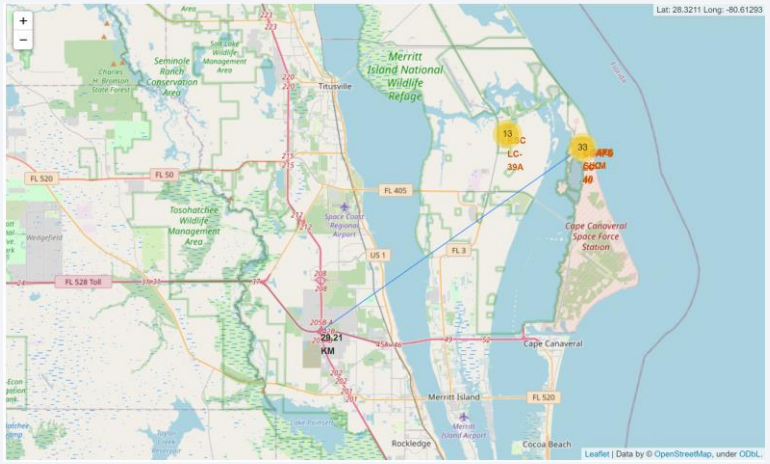
Distance to coastline

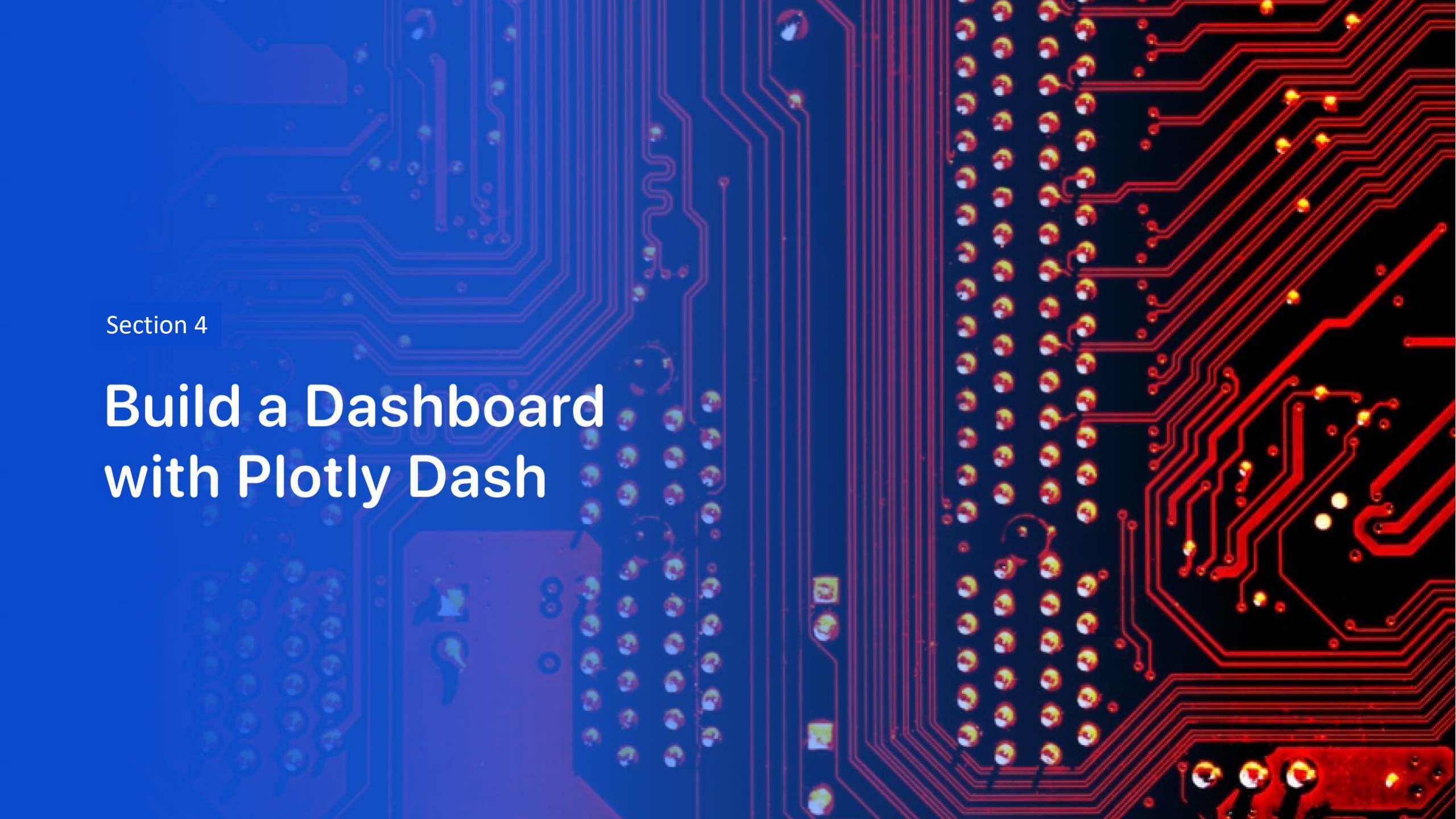


Distance to closest city



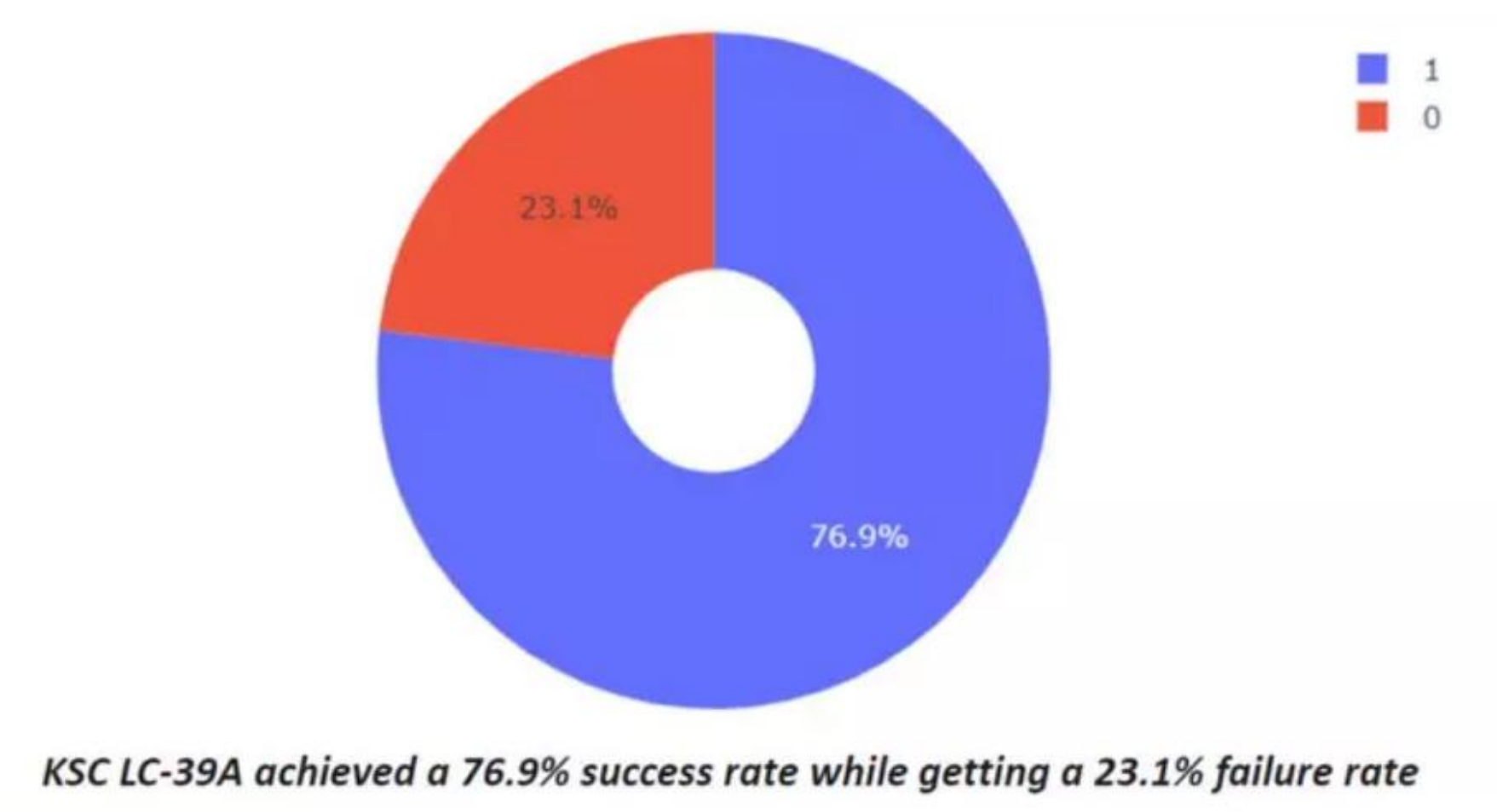
Distance to highway




Section 4
Build a Dashboard with Plotly Dash

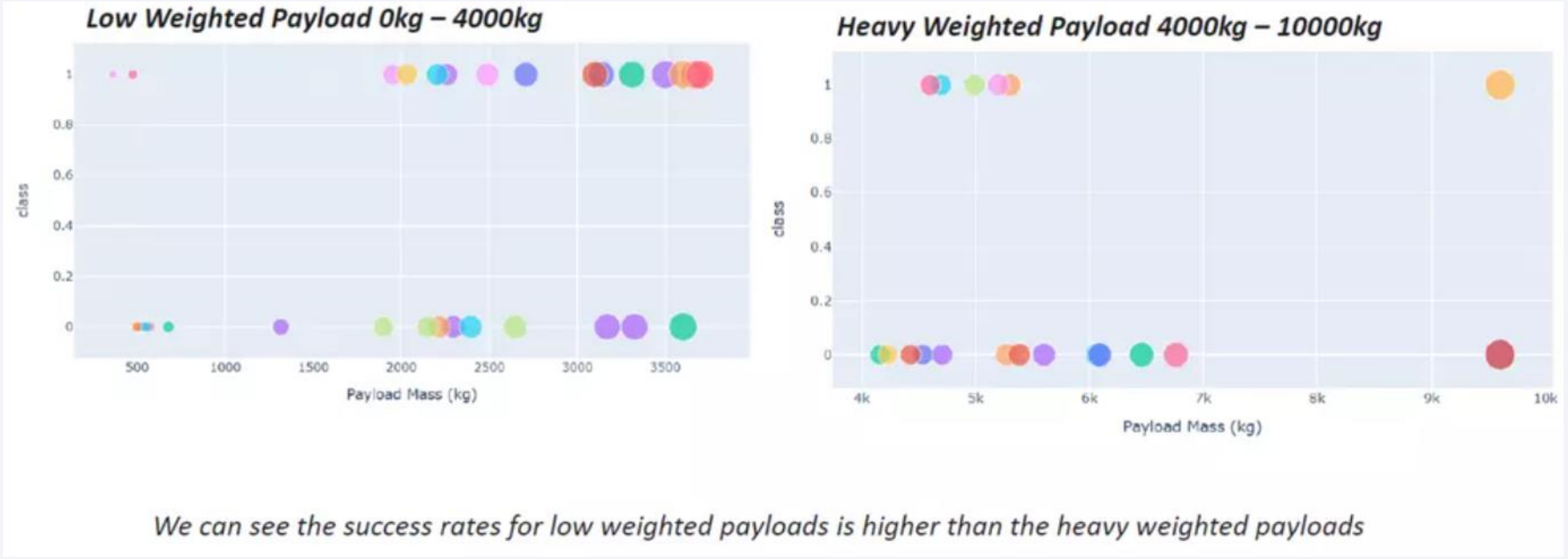
Pie chart showing the success percentage achieved by each launch site

Pie chart showing the Launch site with the highest launch success ratio



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

Scatter plot of Payload vs Launch Outcome for all sites, with different payload selected in the range slider

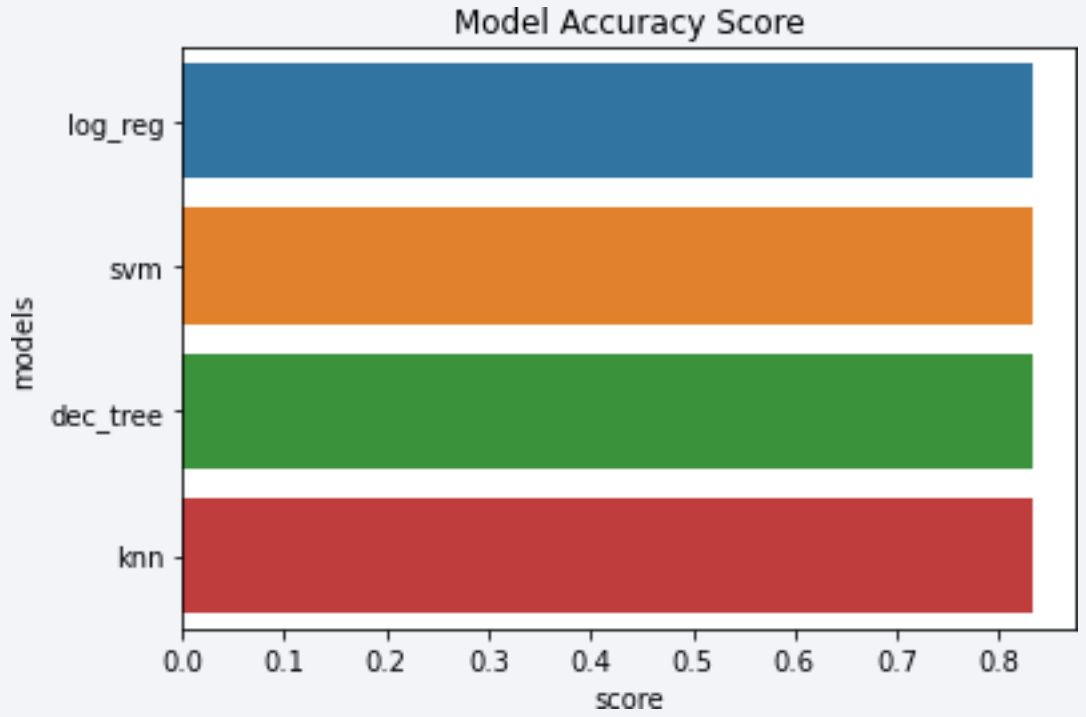


We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

Section 5

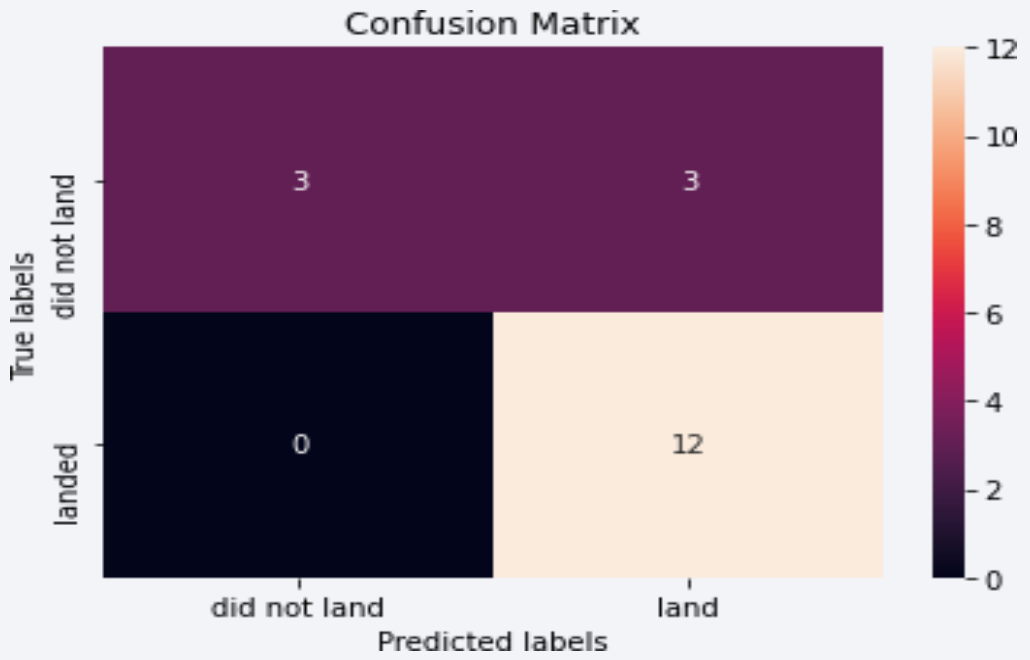
Predictive Analysis (Classification)

Classification Accuracy



- All models had virtually the same accuracy on the test set at 83.33% accuracy
- It should be noted that test size is small at only sample size of 18
- We likely need more data to determine the best model

Confusion Matrix



- Since all models performed the same for the test set, the confusion matrix is the same across all models. The models predicted 12 successful landings when the true label was successful landing.
- The models predicted 3 unsuccessful landings when the true label was unsuccessful landing.
- The models predicted 3 successful landings when the true label was unsuccessful landings (false positives). Our models over predict successful landings.

Conclusions

- Low weighted payloads perform better than heavy payloads
- As the success rate of SpaceX launches is increasing, their future launches are likely to be successful
- Orbits ESL1, SSO, GEO and HEO are the most suitable target orbits
- KSC LC-39A is the most suitable site for the next launch
- Any of SVM, KNN, Decision Tree and Logistic Regression model may be used to predict the accuracy of this dataset

Appendix

- GitHub repository: https://github.com/apar20/IBM_Capstone

Thank you!