Winning Space Race with Data Science

Joaquim Bolós Fernández

29/6/2022

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection from SpaceX REST API
  - Data Collection from Web Scraping Falcon 9 records from Wikipedia
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Data Visualisation and Feature Engineering
  - Interactive Analysis of Geo-spatial data using Folium
  - Interactive Analysis with a Web-based dashboard using Dash and Plotly
  - Machine Learning Prediction with a Classification ML model.
- Summary of all results
  - EDA Results
  - Interactive Analysis Results
  - Predictive Analysis Results

Introduction

Project background and context

Space X advertises Falcon 9 rocket launches on its website, costing 62 million dollars. While other providers cost upward of 165 million dollars each, much of the savings for Space X is because the company can reuse the first stage. Therefore if it can be determined whether the first stage will land, it is possible to 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.

Consequently, this project aims to determine the price of each launch, which will be done by gathering information about Space X and creating dashboards for the team. It will also be determined if SpaceX will reuse the first stage. Instead of using rocket science to determine if the first stage will land successfully, a machine learning model will be trained, and public information will be used to predict if SpaceX will reuse the first stage.

Problems you want to find answers

- What factors determine if the rocket will land successfully?
- The interaction amongst various factors which determines the success rate of a successful landing.
- What factors ensure a successful landing program

Section 1

Methodology

Methodology

Executive Summary

Data collection methodology:

- Data was collected using SpaceX REST APIs and Web Scraping Falcon 9 historical launch records from a Wikipedia page.

Perform data wrangling:

- An Exploratory Data Analysis (EDA) was performed to find some patterns in the data and determine what would be the label for training supervised models.

Perform exploratory data analysis (EDA) using visualisation and SQL

- An Exploratory Data Analysis (EDA) was performed to find some patterns in the data, and Feature engineering was also applied.

Perform interactive visual analytics using Folium and Plotly Dash

Perform predictive analysis using classification models

- After an EDA, the training labels are determined, data is standardised and split into test and training sets, and finally, various Classification methods are evaluated to find the method and its corresponding hyper-parameters that perform best.

Data Collection

Data collection process:

- Data was collected using requests.get to the SpaceX REST API.
- Then, the response content was decoded as a Json using .json() function and then turned into a pandas dataframe using .json_normalize().
- Finally, the data was cleaned only to include Falcon 9 launches, and missing values for the Payload mass were replaced with the average of the non-missing values
- On the other hand, web scraping was used to obtain data from Wikipedia for Falcon 9 launch records with BeautifulSoup.
- The launch records were obtained as an HTML table, parsed and converted into a pandas dataframe for future analysis.

Data Collection – SpaceX REST API

GitHub Link

Data was collected using requests.get to the SpaceX REST API.

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

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

Then, the response content was decoded as a Json using .json() function and then turned into a pandas dataframe using .json_normalize().

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

Finally, the data was cleaned only to include Falcon 9 launches, and missing values for the Payload mass were replaced with the average of the non-missing values

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

# Remove launches that are not from Falcon9
data_falcon9 = df[df['BoosterVersion']!='Falcon 1']
data_falcon9.head()

# Calculate the mean value of PayloadMass column
PayloadMass = data_falcon9['PayloadMass'].mean()

# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, PayloadMass)
```

Data Collection - Scraping

GitHub Link

Web scraping was used to obtain data from Wikipedia for Falcon 9 launch records with BeautifulSoup.

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

# use requests.get() method with the provided static_url
response = requests.get(static_url)

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

# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

The launch records were obtained as an HTML table, parsed and converted into a pandas dataframe for future analysis.

```
[ ]: # Using find_all() and extra progamming, all the data from the HTML table is stored into a Pyhton dictionary (launch_dict)

# Then a dataframe is created from dict
df=pd.DataFrame(launch_dict)
```

Data Wrangling

GitHub Link

Overall, an exploratory data analysis was performed and the training labels were determined.

In the lab I calculated the number of launches at each site, the number and occurrence of each orbits, and the number and occurence of mission outcome per orbit type

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

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

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

Finally, a landing outcome label from Outcome column was created

[ ]:
```
landing_class = []

for count in range(len(df['Outcome'])):

    # landing_class = 0 if bad_outcome
    if df['Outcome'][count] in bad_outcomes:
        landing_class.append(0)

    # landing_class = 1 otherwise
    else:
        landing_class.append(1)
```

EDA with Data Visualisation

GitHub Link

The data was explored by visualising the relationship between:

- 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

EDA with SQL

GitHub Link

SQL Queries:

- All Launch Site Names
- Launch Site Names Begin with ‘CCA'
- Total Payload Mass
- Average Payload Mass by F9 v1.1
- First Successful Ground Landing Date
- Successful Drone Ship Landing with Payload between 4000 and 6000
- Total Number of Successful and Failure Mission Outcomes
- Boosters Carried Maximum Payload
- 2015 Launch Records
- Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

Build an Interactive Map with Folium

GitHub Link

- Firstly, all launch sites on a map were marked with markers and circles.
- Secondly, the success/failed launches for each site on the map were marked using a MarkerCluster object, which added individual markers with an icon property to indicate if the launch was successful or failed (Color-labeled markers).
- Finally, the distances between the launch site to its proximities were calculated using the MousePosition. In addition, the distances were plotted in the Map using Markers and PolyLines.

Build a Dashboard with Plotly Dash

GitHub Link

The interactive dashboard with Plotly dash included:

- Pie charts showing the total amount of launches depending on the sites
- A Scatter graph showing the relationship of the Outcome and Payload Mass (Kg) depending on the booster version.

Predictive Analysis (Classification)

GitHub Link

- The data frame was loaded, and then using Numpy and Pandas, the target label Y and the features X were defined.
- Then, the data was standardised and split into the test and training sets
- Next, different ML Classifiers (Logistic Regression, SVM, Decison Tree and KNN) were built, and their corresponding hyperparameters were tuned using GridSearchCV
- For each model, the accuracy of the train and test data was calculated using the method score, and confusion matrices were obtained.
- Finally, the best performing classification model was obtained

Results

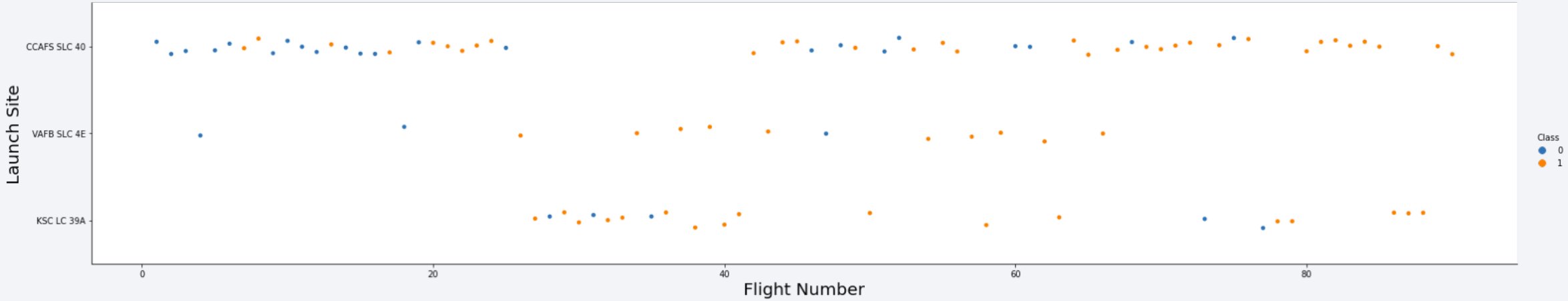
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

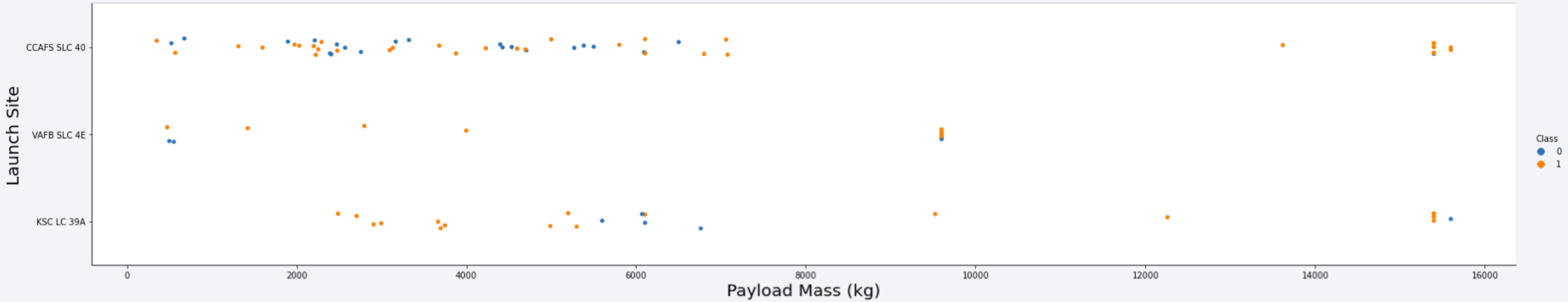
Flight Number vs. Launch Site

- From the plot, it can be seen that the larger the flight number at a launch site, the greater the success rate at a launch site.



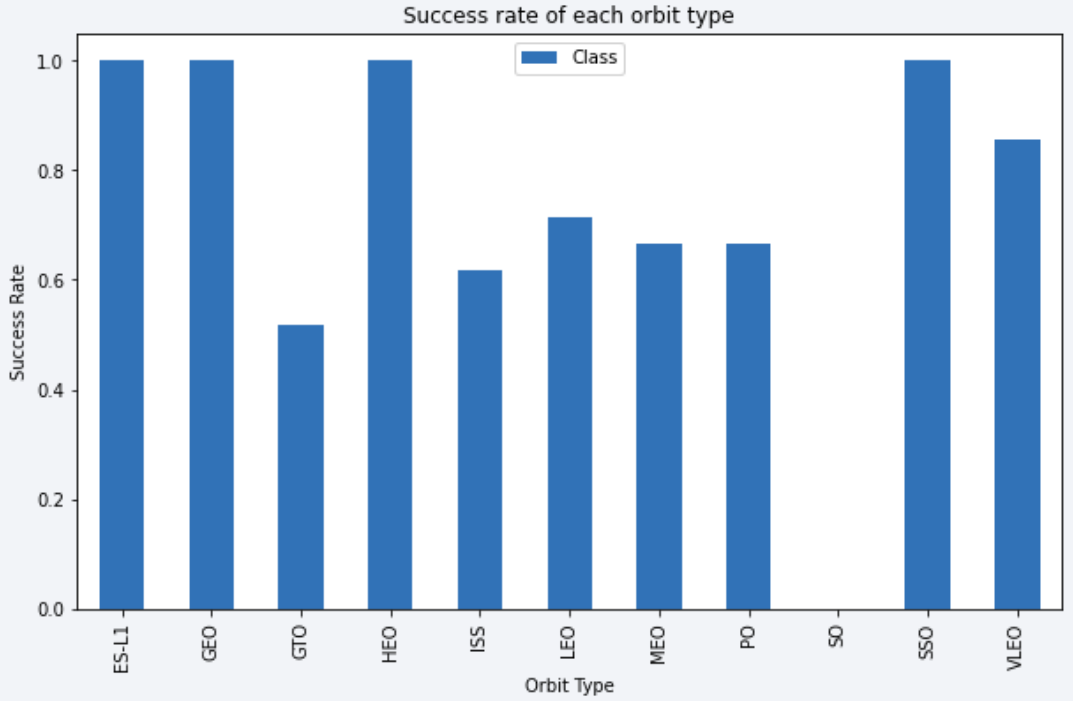
Payload vs. Launch Site

- The greater the payload mass for the launch site CCAFS SLC 40 the higher the success rate for the rocket
- For the VAFB-SLC launch site there are no rockets launched for heavy payload mass(greater than 10000)



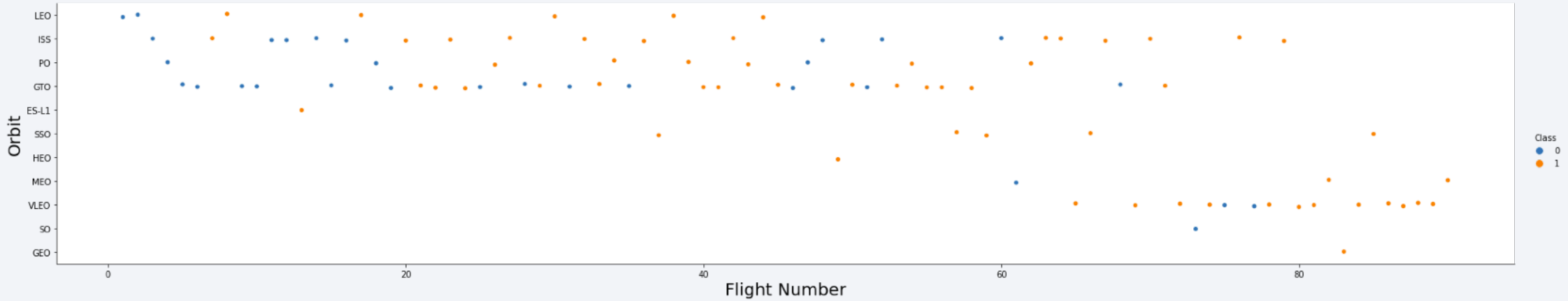
Success Rate vs. Orbit Type

- Orbit Types ES-L1, GEO, HEO, SSO, VLEO have a 100% success rate, whereas the rest oscillates around 60% success rate, and SO has a success rate of 0%



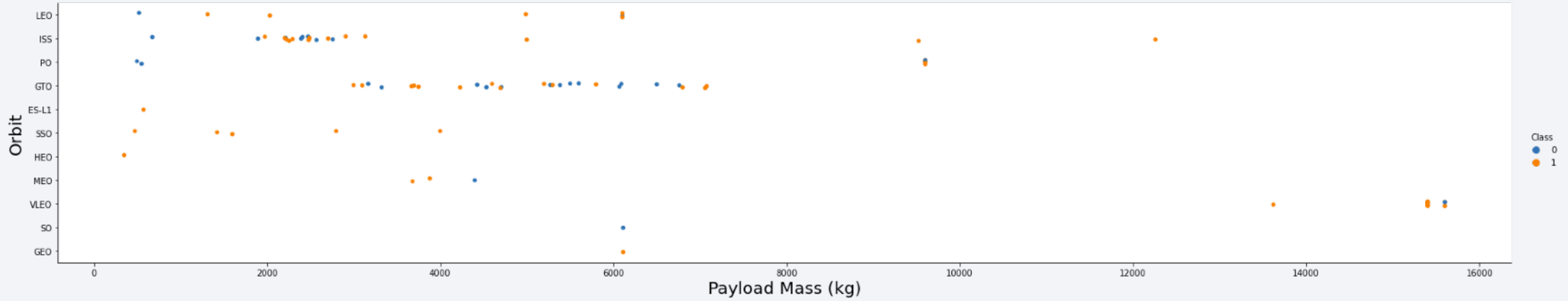
Flight Number vs. Orbit Type

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



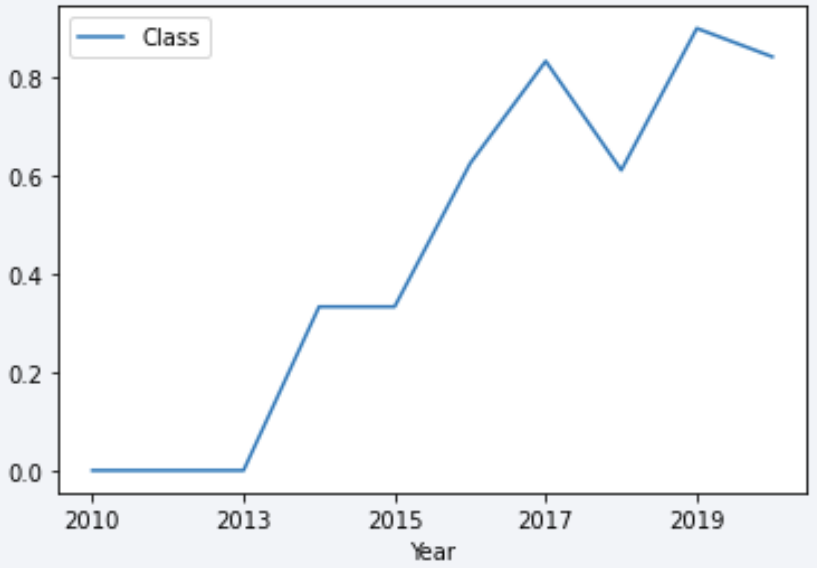
Payload vs. Orbit Type

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



Launch Success Yearly Trend

- Success rate has been increasing since 2013 till 2020.



All Launch Site Names

- The keyword **DISTINCT** is used to show only unique launch sites from the SpaceX data.

Display the names of the unique launch sites in the space mission

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

Out[10]:

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

Launch Site Names Begin with 'CCA'

- The keywords **WHERE** and **LIKE** are used to show only unique launch sites with the String ‘CCA’

Display 5 records where launch sites begin with the string 'CCA'

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

Out[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

- The keywords **SUM** and **LIKE** are used to display the Total Payload Mass

Display the total payload mass carried by boosters launched by NASA (CRS)

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

Out[12]:

	total_payloadmass
0	45596

Average Payload Mass by F9 v1.1

- **AVG** is used to show the Average Payload Mass by F9 v1.1.

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

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

Out[13]:

	avg_payloadmass
0	2928.4

First Successful Ground Landing Date

- **MIN** is used to show the First Successful Ground Landing Date.

In [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)
```

Out[14]:

	firstsuccessfull_landing_date
0	2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- The **WHERE** clause was used to filter the query for boosters which have successfully landed on drone ship. In addition, the **AND** condition determined successful landing with payload mass greater than 4000 but less than 6000 (**BETWEEN** condition could have been used too).

In [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)
```

Out[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

- The keyword **LIKE** is used again to show the Total Number of Successful and Failure Mission Outcomes.

List the total number of successful and failure mission outcomes

In [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:
```

Out[16]:

	failureoutcome
0	1

Boosters Carried Maximum Payload

- **MAX** is used, within a subquery, to show the Boosters which have Carried the Maximum Payload.

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

In [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)
```

Out[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

- A combination of the **WHERE** clause, **LIKE**, **AND**, and **BETWEEN** conditions were used to filter the query for: failed landing outcomes in drone ship, their booster versions, and launch site names within the year 2015

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

In [18]:
```
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)
```

Out[18]:

	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

- A combination of **COUNT, WHERE, BETWEEN, GROUP BY** and **ORDER BY** allowed to obtain the Rank of the 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))

In [19]:
```
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)
```

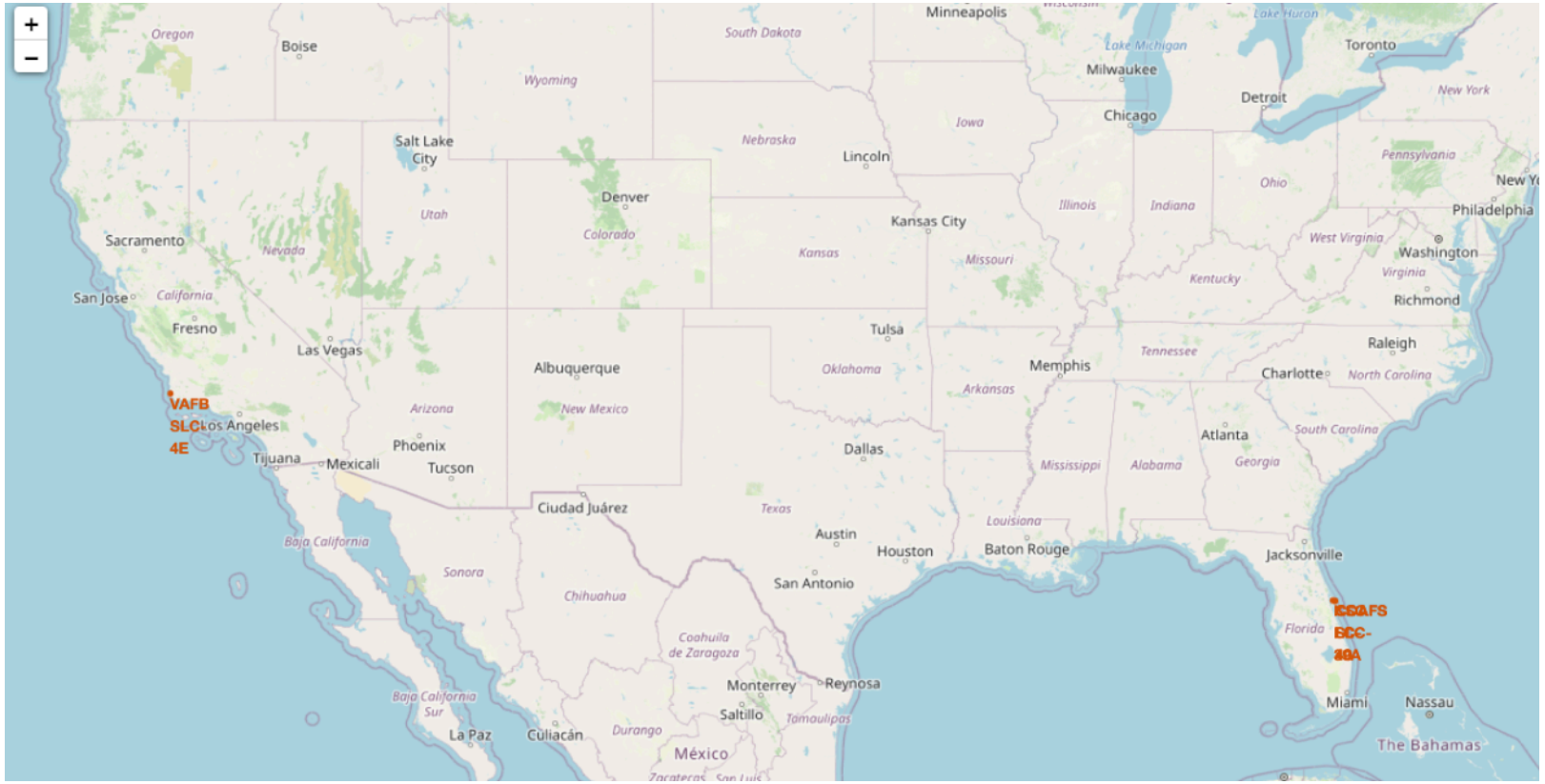
Out[19]:

	landingoutcome	count
0	No attempt	10
1	Success (drone ship)	6
2	Failure (drone ship)	5
3	Success (ground pad)	5
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Precluded (drone ship)	1
7	Failure (parachute)	1

Section 3

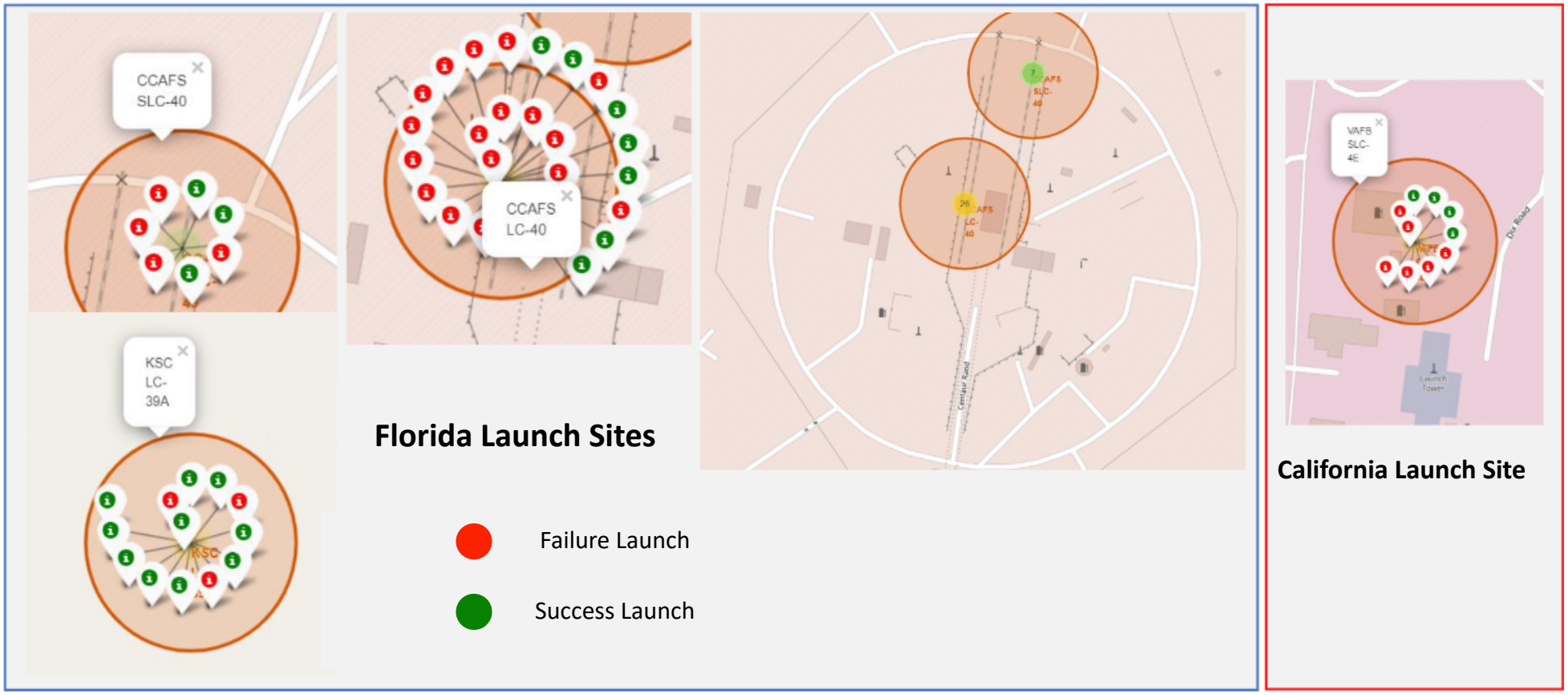
Launch Sites Proximities Analysis

Launch sites on a map

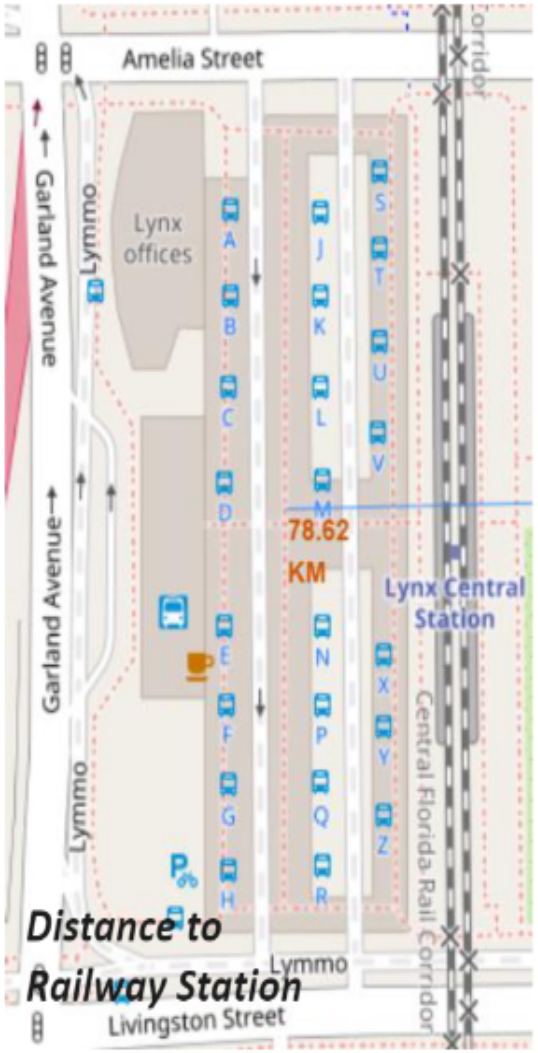


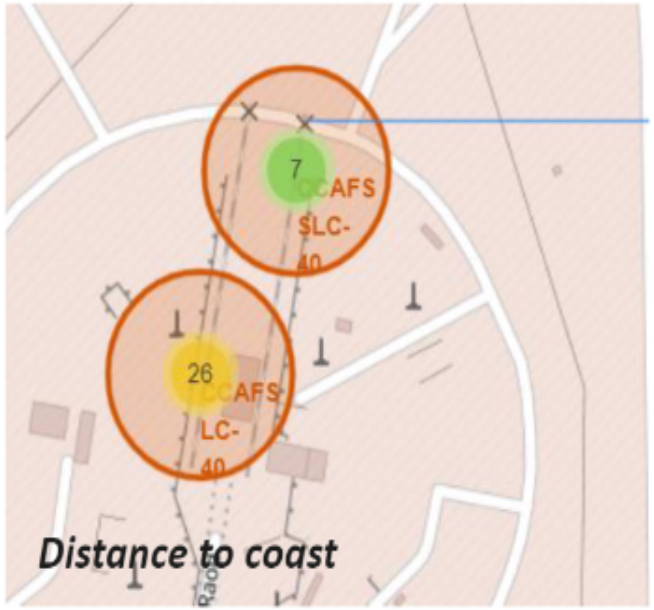
SpaceX launch Sites are in the US East and West Coast

The success/failed launches for each site on the map

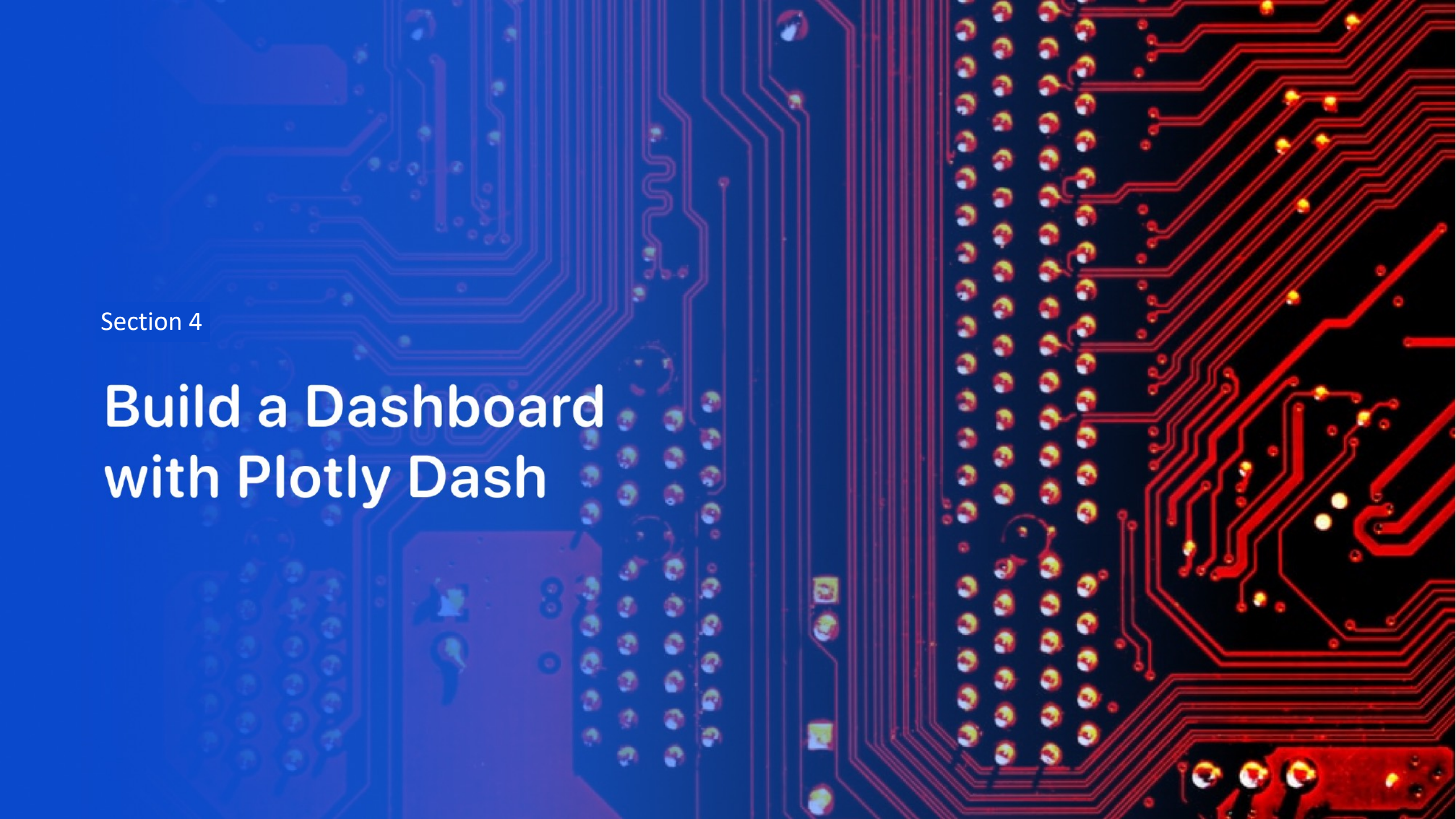


Distances between a launch site to its proximities

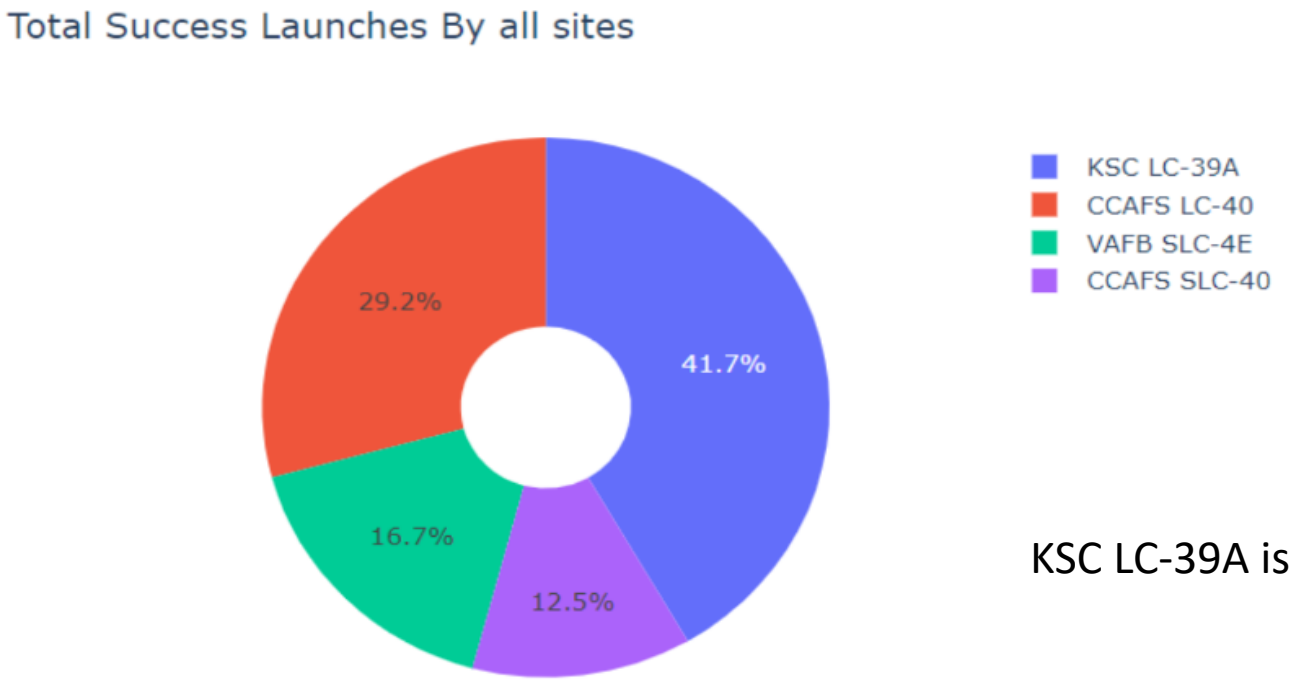




- Are launch sites in close proximity to railways? - No
- Are launch sites in close proximity to highways? - No
- Are launch sites in close proximity to coastline? - Yes
- Do launch sites keep certain distance away from cities? - Yes


Section 4
Build a Dashboard with Plotly Dash

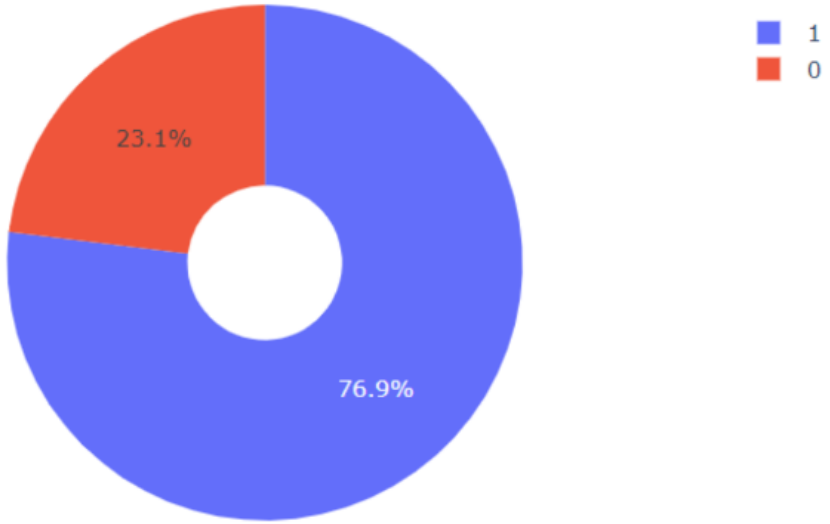
Pie chart showing the success percentage achieved by each launch site



KSC LC-39A is the site with most successful launches

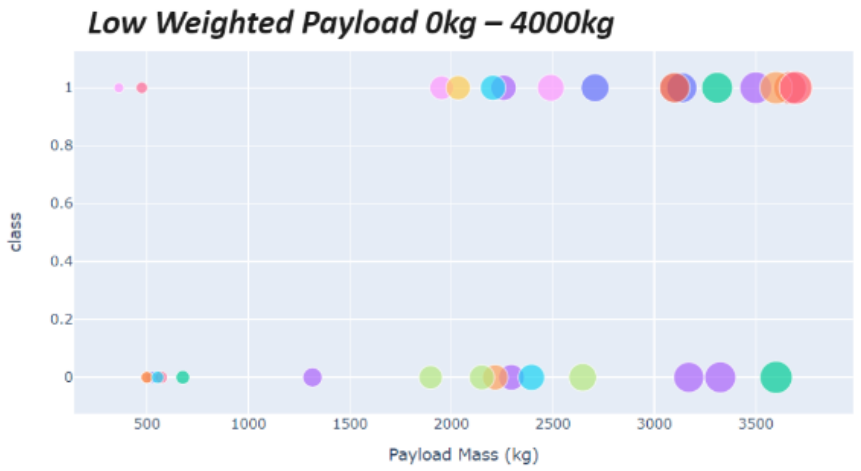
Pie chart showing the Launch site with the highest launch success rate

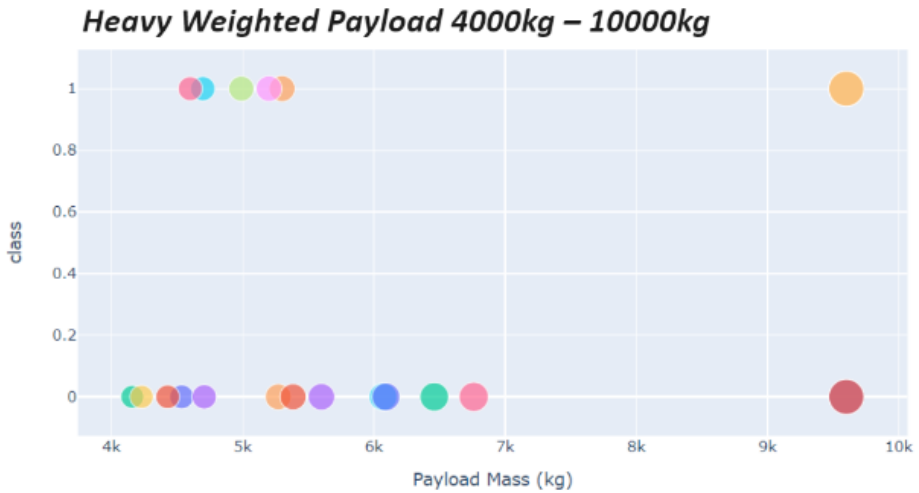
KSC LC-39A has a 76.9% success rate



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

The Success rates for low payload is higher than for heavy payload

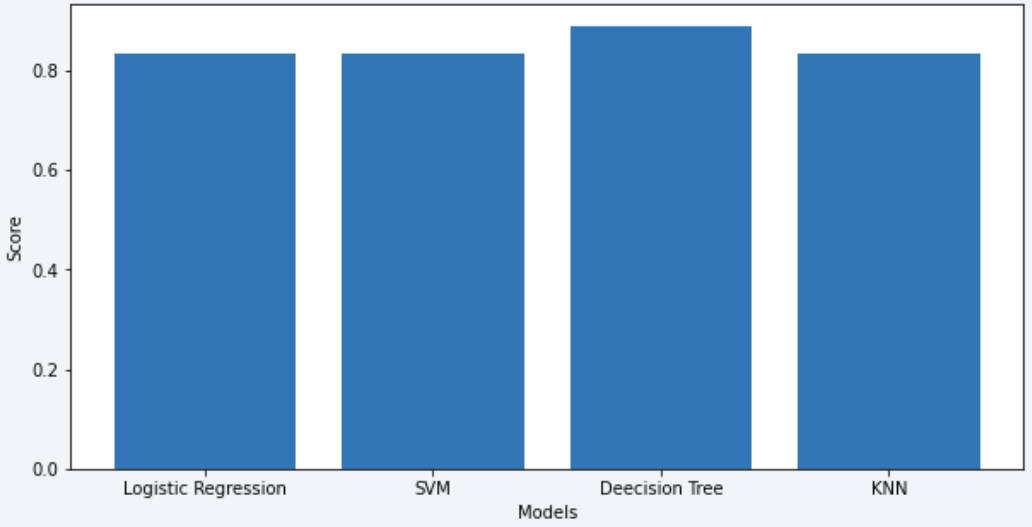




Section 5

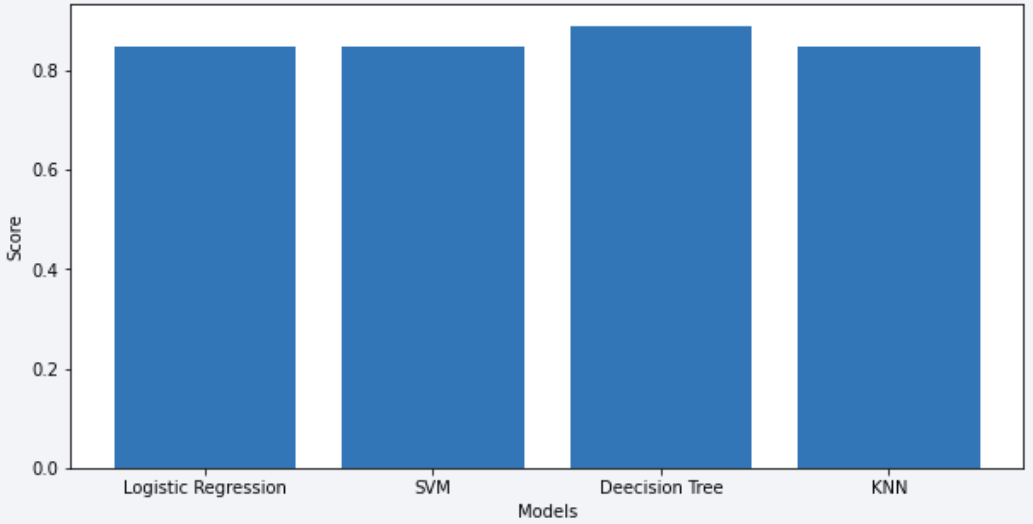
Predictive Analysis (Classification)

Classification Accuracy



Test Scores

	Scores
Logistic Regression	0.833333
SVM	0.833333
Deecision Tree	0.888889
KNN	0.833333



Train Scores

	Scores
Logistic Regression	0.846429
SVM	0.848214
Deecision Tree	0.889286
KNN	0.848214

Confusion Matrix

Examining the confusion matrix, it can be seen that the decision tree can distinguish between the different classes. Nonetheless, it also indicates that the major problem in model's predictions are false positives.

Conclusions

- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- Launch success rate increased in 2013 till 2020.
- Orbits ES-L1, GEO, HEO, SSO, VLEO have a success rate of 100%.
- KSC LC-39A has had more successful launches than the other sites.
- The Decision Tree Classifier is the best machine learning model for predicting the first stage landing outcome.

Thank you!