IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Chris Manuel
24th October 2021

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
- Building a Dashboard with Plotly Dash
- Predictive analysis (Classification)
- Summary of all results

Summary of all results

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

Introduction

Project background

- This project predicts if the SpaceX Falcon-9 first stage will land successfully or not. The cost of a launch by SpaceX is $62MM – other providers can cost up to $165MM. Much of the savings from SpaceX comes from the potential to reuse the first stage.
- If it can be determined if the first stage will land successfully, then the cost of a launch can be determined thereby providing the information necessary to enable SpaceX to be competitive for rocket launches.

Problems that require answering

- What are the factors that influence if the rocket will land successfully?
- What is the relationship with certain rocket variables in determining the success rate of a first stage landing?
- What conditions is SpaceX required to achieve to ensure the largest rocket success landing rate?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology
  - SpaceX Rest API
  - Web Scraping
- Perform data wrangling
  - Data transformation to a dataframe
  - Isolation of Falcon-9 data
  - Deal with missing values
  - Omission of irrelevant attributes
  - One hot encoding to transform categorical variables
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Extraction of pertinent information using SQL.
  - Matplotlib and Seaborn to create various plots for feature analysis.
  - Feature extraction.
- Perform interactive visual analytics using Folium and Plotly Dash
  - In-depth dynamic visualization of various features
  - Extract geographical patterns about launch sites.
- Perform predictive analysis using classification models
  - Build, tune, and evaluate classification models

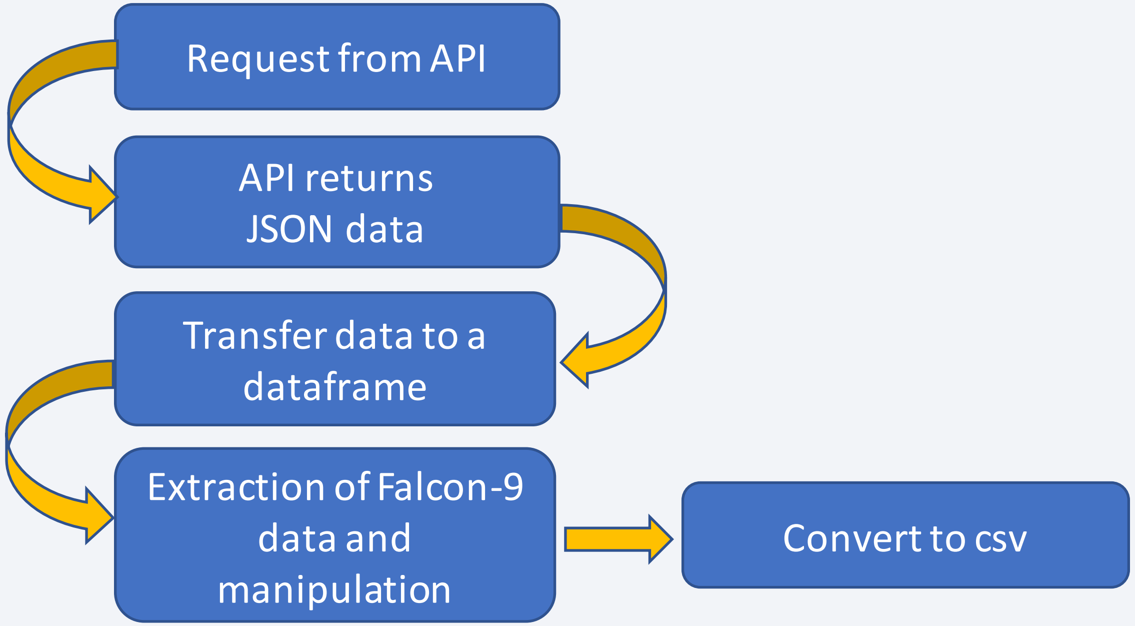
Data Collection

SpaceX launch data was acquired from two sources

1) SpaceX REST API:

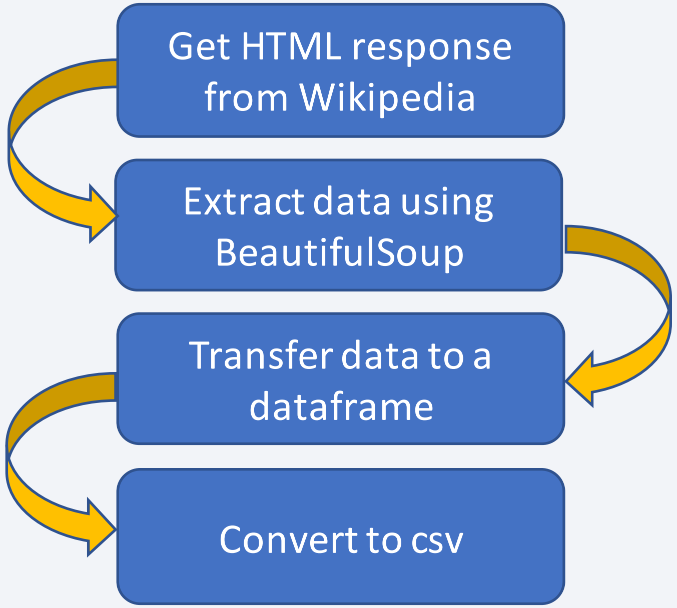
a) The information obtained included the type of rocket, payload masses, launch and landing specifications, landing outcomes, and more.

b) SpaceX API (endpoint = past launches)



Request from API
API returns JSON data
Transfer data to a dataframe
Extraction of Falcon-9 data and manipulation
Convert to csv

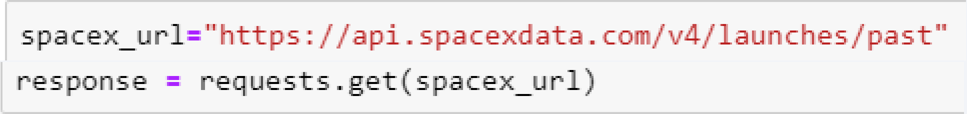
2) Web scraping of the Falcon-9 launch data from Wikipedia using *BeautifulSoup*.



Get HTML response from Wikipedia
Extract data using BeautifulSoup
Transfer data to a dataframe
Convert to csv

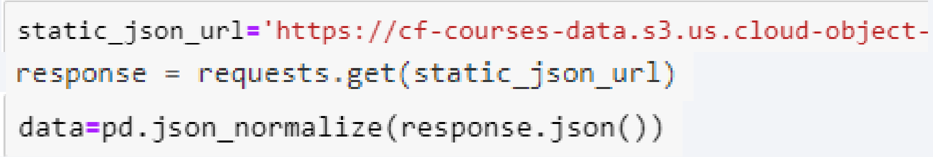
Data Collection – SpaceX API

1. **Get response from URL**



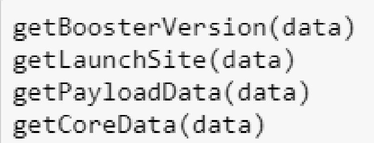
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
response = requests.get(spacex_url)
```

2. **Convert response to JSON**



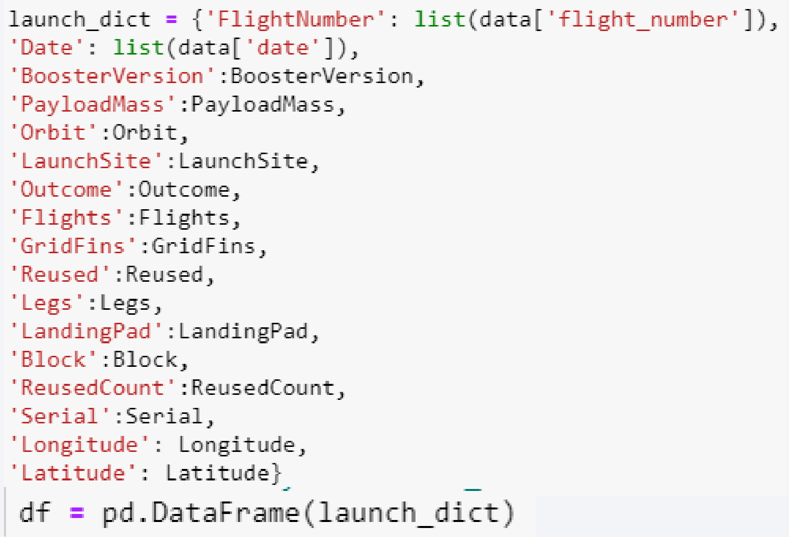
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-
response = requests.get(static_json_url)
data=pd.json_normalize(response.json())
```

3. **Get information about the launches using the IDs given for each launch**



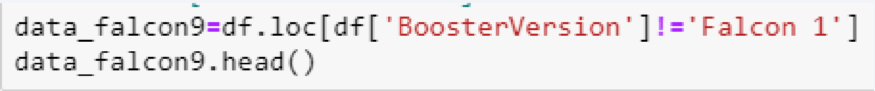
```
getBoosterVersion(data)
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
```

4. **Assign list to dictionary and create a dataframe**



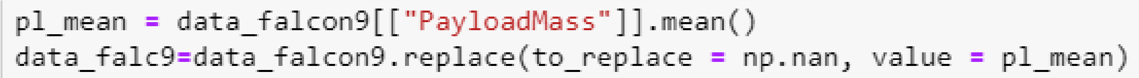
```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,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

5. **Filter the dataframe to only include Falcon 9 launches**



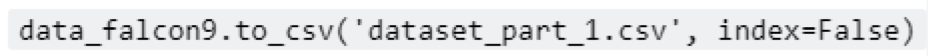
```
data_falcon9=df.loc[df['BoosterVersion']!='Falcon 1']
data_falcon9.head()
```

6. **Data wrangling – dealing with missing values**

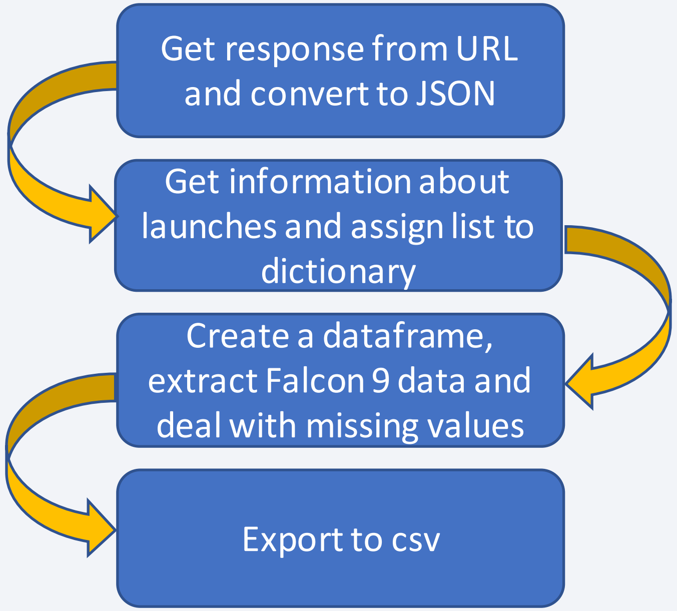


```
pl_mean = data_falcon9[["PayloadMass"]].mean()
data_falc9=data_falcon9.replace(to_replace = np.nan, value = pl_mean)
```

7. **Export to CSV**



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



Get response from URL and convert to JSON

Get information about launches and assign list to dictionary

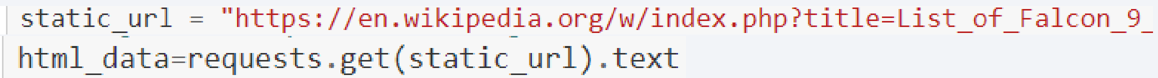
Create a dataframe, extract Falcon 9 data and deal with missing values

Export to csv

Link to GitHub Notebook

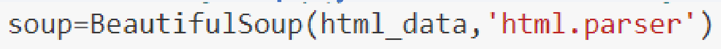
Data Collection – Web Scraping

1. **Get response from HTML**



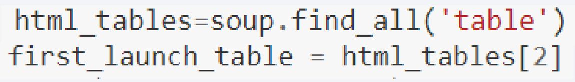
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_
html_data=requests.get(static_url).text
```

2. **Create BeautifulSoup Object**



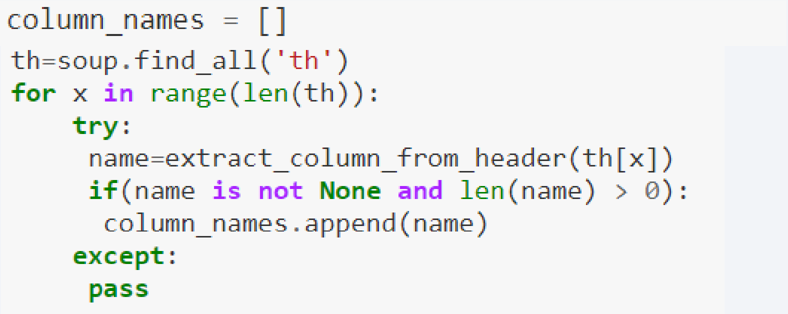
```
soup=BeautifulSoup(html_data,'html.parser')
```

3. **Get tables**



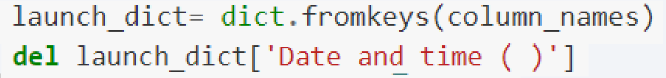
```
html_tables=soup.find_all('table')
first_launch_table = html_tables[2]
```

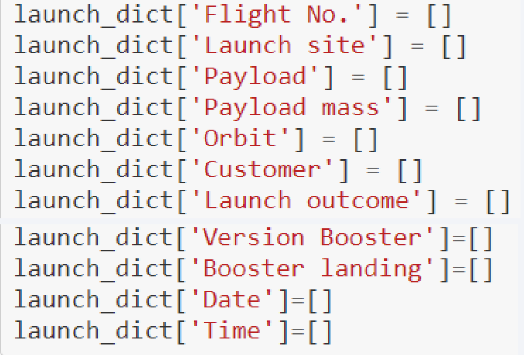
4. **Get column names**



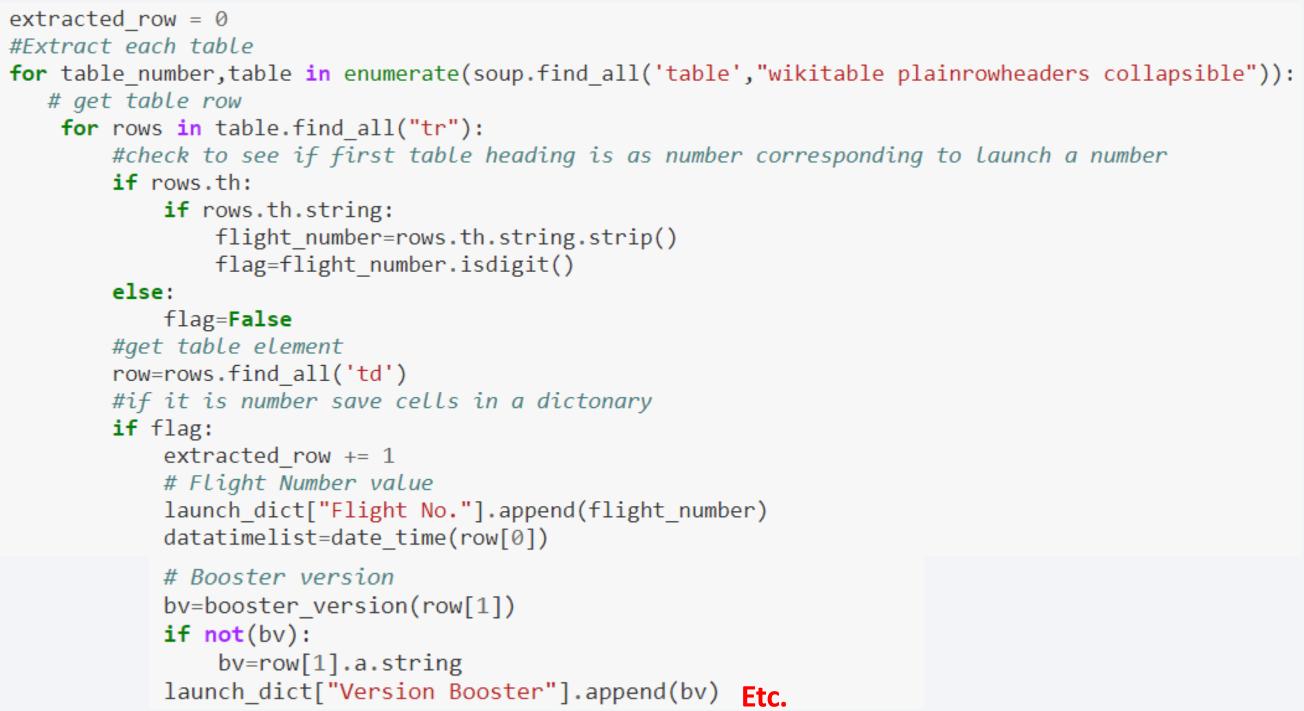
```
column_names = []
th=soup.find_all('th')
for x in range(len(th)):
    try:
     name=extract_column_from_header(th[x])
     if(name is not None and len(name) > 0):
      column_names.append(name)
    except:
     pass
```

5. **Create an empty dictionary then append data**





```
launch_dict= dict.fromkeys(column_names)
del launch_dict['Date and time ( )']
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'] = []
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```



```
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
            launch_dict["Flight No."].append(flight_number)
            datatimelist=date_time(row[0])

            # Booster version
            bv=booster_version(row[1])
            if not(bv):
                bv=row[1].a.string
            launch_dict["Version Booster"].append(bv)
```

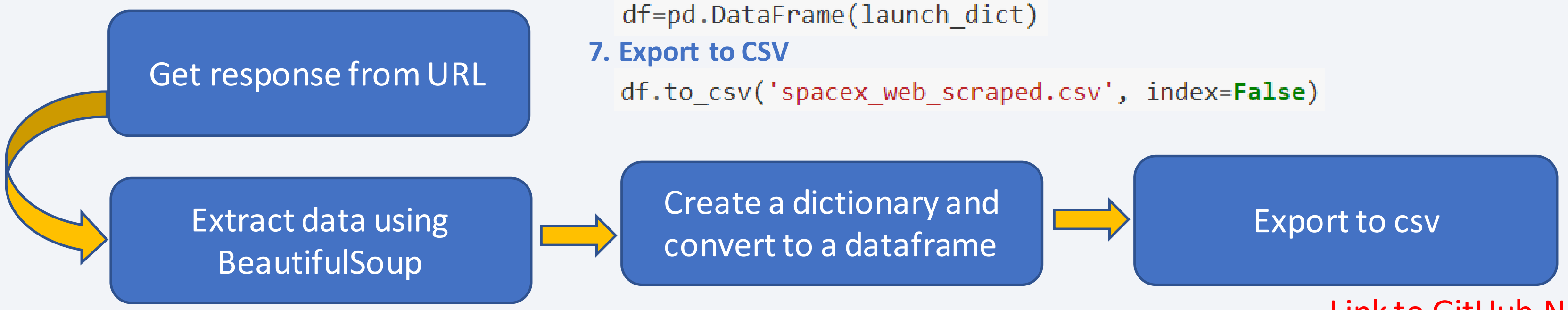
Etc.

6. **Convert to a datafame**

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

7. **Export to CSV**

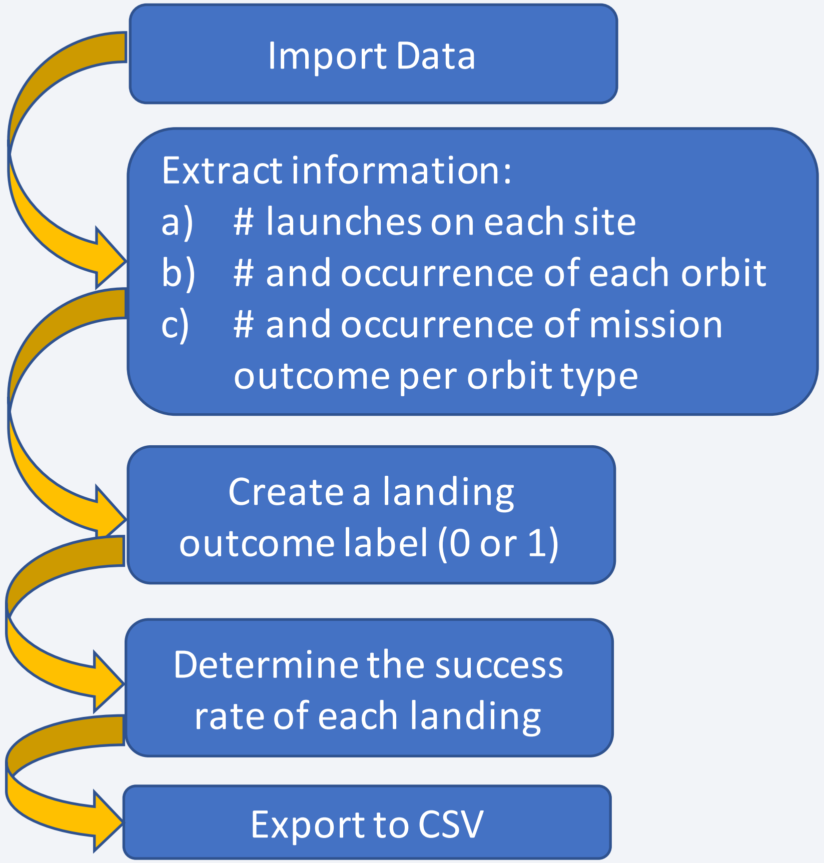
```
df.to_csv('spacex_web_scraped.csv', index=False)
```



Get response from URL → Extract data using BeautifulSoup → Create a dictionary and convert to a dataframe → Export to csv

Link to GitHub Notebook

Data Wrangling



Import Data

Extract information:
a) # launches on each site
b) # and occurrence of each orbit
c) # and occurrence of mission outcome per orbit type

Create a landing outcome label (0 or 1)

Determine the success rate of each landing

Export to CSV

The dataset contains several pieces of information needing to be extracted:

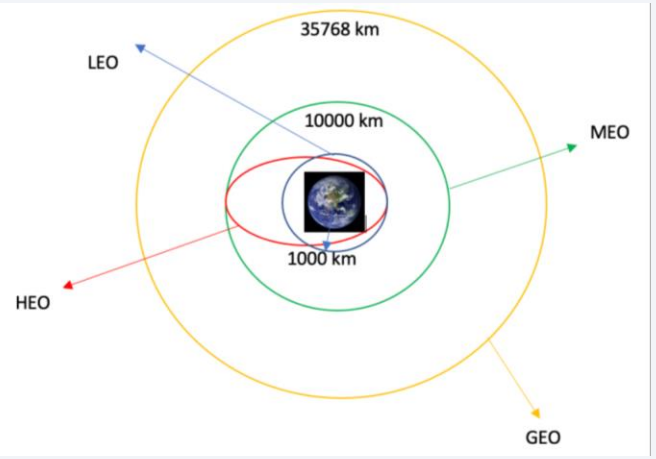
1. Different cases for outcomes of the booster launch:
   a) True / False Ocean: the mission outcome was successfully / unsuccessfully landed to a specific region of the ocean;
   b) True / False RTLS: the mission outcome was successfully / unsuccessfully landed to a ground pad;
   c) True / False ASDS: the mission outcome was successfully / unsuccessfully landed on a drone ship;
   d) “None ASDS” and “None data fame”: represent a failure to land.

Based on the extracted information we create a landing outcome label for the booster: 0 = fail or 1 = success and assign it to “landing_class”.

2. Space X launch facilities contained in the column “Launchsite”:
   a) Cape Canaveral Space Launch Complex 40 VAFB SLC 4E
   b) Vandenberg Air Force Base Space Launch Complex 4E (SLC-4E)
   c) Kennedy Space Center Launch Complex 39A KSC LC 39A

“.value_counts()” is applied to the column “LaunchSite” to determine the number of launches on each site

3. Dedicated orbits for each launch some of which are shown below. We calculate the number and occurrence of mission outcome per orbit type.



We determine the success rate of each landing by taking the mean of each class of landing.

Link to GitHub Notebook

,

EDA with Data Visualization

Two Python libraries were used to create plots for determining potential relationships between variables:

1. Matplotlib
2. Seaborn (based on Matplotlib) provides a high-level interface for drawing attractive and informative statistical graphics

Three types of plots were created to visualize potential relationships:

Scatterplot

A type of plot or mathematical diagram to display how much one variable is correlated with another. An example python command is given as follows:

```
sns.catplot(y="PayloadMass", x="FlightNumber",
            hue="Class", data=df, aspect = 2)
```

Several variables were plotted to determine suitability to use in subsequent feature engineering:

a) FlightNumber vs. PayloadMass
b) FlightNumber vs LaunchSite
c) launch sites vs payload mass.
d) FlightNumber vs Orbit type.
e) Payload vs Orbit type

Bar Chart

Used for plotting categorical data with bars with the lengths or heights proportional to the values they represent. An example of the Python script is as follows:

```
sns.barplot(data = df_mean, y = "Class", x ="Orbit", hue="Orbit" )
```

For this exercise, the bar chart was used to display the *success rate* of the booster landing as a function of the *target orbit* type.

Line Plot

Display data variables plotted against one another where trends can easily be visualized. An example of the Python code is given here:

```
sns.lineplot(data=df_mean, x="Year", y="Class")
```

Successful booster landings vs *Year* were plotted in this form.

Link to GitHub Notebook

EDA with SQL

SQL was used to interrogate the data to extract various pieces of information. Initially, the SpaceX CSV dataset was loaded i nto a table in a Db2 database. SQL commands were then used to extract the information desired from each of the following queries.

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

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

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

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

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

List the names of the boosters which have success in drone ship and 4000 < payload mass <6000

List the total number of successful and failure mission outcomes

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

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

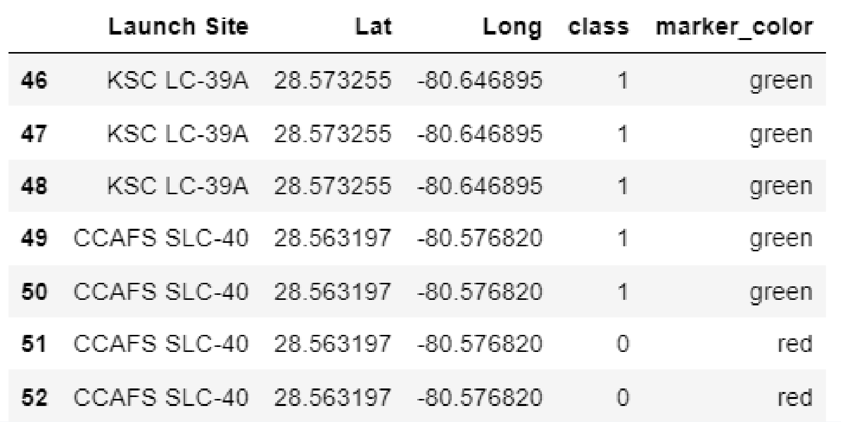
Rank the count of landing outcomes (e.g. Failure (drone ship) or Success (ground pad)) between the dates 2010-06-04 and 2017-03-20, in descending order

Link to GitHub Notebook

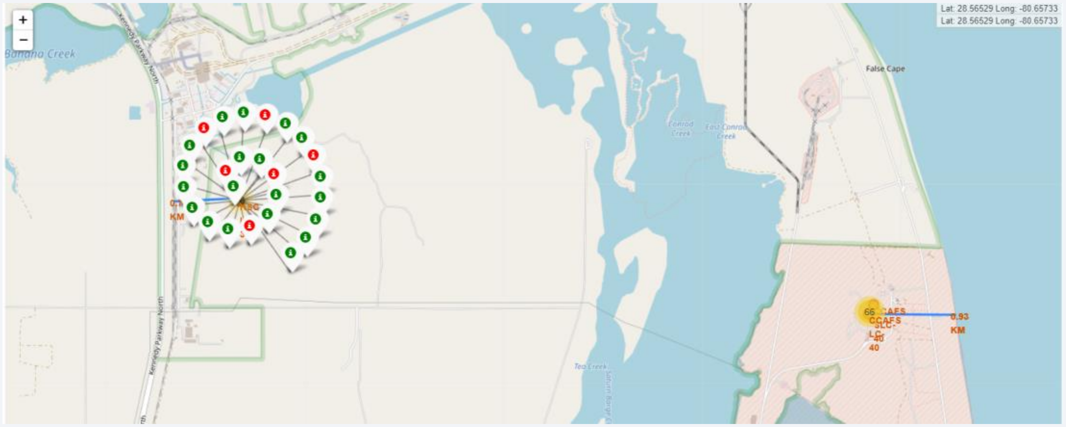
Build an Interactive Map with Folium

The launch success rate may depend on the location of a launch site. Folium can be used to determine if there are any geographical influences on a successful launch:

1. The lat/long of each launch site were added to the map with a *circle* around each launch site including an identifying *marker*.
2. Launch outcomes were added for each site by creating *markers* for all launch records. If a launch was successful (class=1) it was given a green marker and if a launch was a failure (class=0) a red marker was used. *Marker clusters* were used to simplify a map containing many markers having the same coordinate.
3. A "*MousePosition*" was added to on the map to obtain coordinates for any point such as a railway, a highway, or the coastlines. *Lines* are drawn on the map to each of these features and the distance calculated using the Haversine formula (determines the great-circle distance between two points on a sphere given their longitudes and latitudes).



	Launch Site	Lat	Long	class	marker_color
46	KSC LC-39A	28.573255	-80.646895	1	green
47	KSC LC-39A	28.573255	-80.646895	1	green
48	KSC LC-39A	28.573255	-80.646895	1	green
49	CCAFS SLC-40	28.563197	-80.576820	1	green
50	CCAFS SLC-40	28.563197	-80.576820	1	green
51	CCAFS SLC-40	28.563197	-80.576820	0	red
52	CCAFS SLC-40	28.563197	-80.576820	0	red



Link to Github (unfortunately there is a rendering issue)

Link to Jupyter notebook

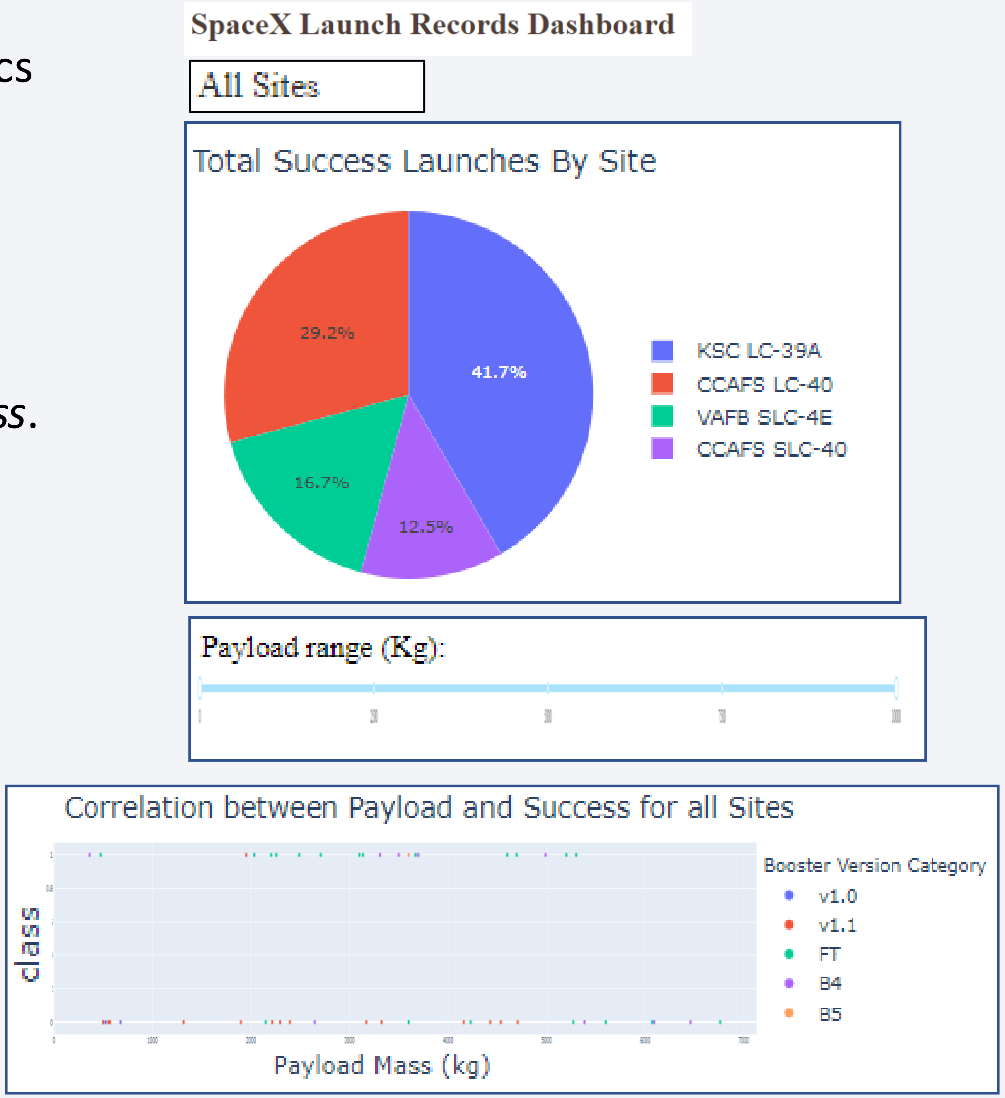
Build a Dashboard with Plotly Dash

Using Plotly Dash a dashboard was created to allow interactive visual analytics on SpaceX launch data in real-time. The dashboard consists of the following components:

- Dropdown list - to enable *Launch Site* selection
- Pie chart - to show the *total successful launches count* for all sites.
- Slider - to *select payload range*
- Scatter chart - to show the *correlation between payload and launch success*.

The intent of the dashboard was to enable insights on the following:

1. Which site has the largest successful launches?
2. Which site has the highest launch success rate?
3. Which payload range(s) has the highest launch success rate?
4. Which payload range(s) has the lowest launch success rate?
5. Which F9 Booster version (v1.0, v1.1, FT, B4, B5, etc.) has the highest launch success rate?

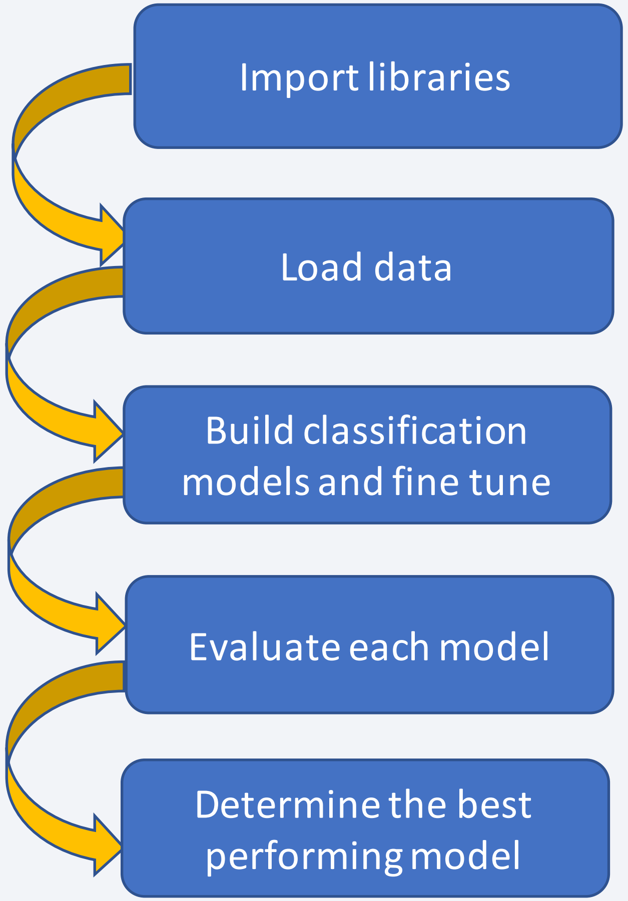


SpaceX Launch Records Dashboard

All Sites

Link to GitHub Notebook

Predictive Analysis (Classification)



Import libraries

Load data

Build classification models and fine tune

Evaluate each model

Determine the best performing model

Import Libraries and Define Auxiliary Functions

Using the library *sklearn* various functionalities are imported to allow:

a) Splitting the data into training and test datasets (*train_test_split*)

b) Refine the parameters of each classification algorithm (*GridSearchCV*)

c) Utilise various classification algorithms (logisitic regression, SVC, decision tree, and KNN)

Load the data

a) Load dataset from csv to pandas dataframe

b) Define a feature matrix (X) and an outcome variable (Y)

c) Standardize the feature matrix

d) Split data into training and test data sets: X_train, X_test, Y_train, Y_test

e) Some of the training data is set aside for model validation.

Build classification models and fine tune

For each of the classification algorithms **-** logistic regression, SVC, decision tree, and KNN – an object is created and used to fit the data while using GridSearchCV to optimize hyperparameters. Use *model.best_params* method to see the output of the grid search.

Evaluate each model

a) Each model is evaluated on the validation data by outputting the accuracy using *model.**best_score***

b) Plot the confusion matrix

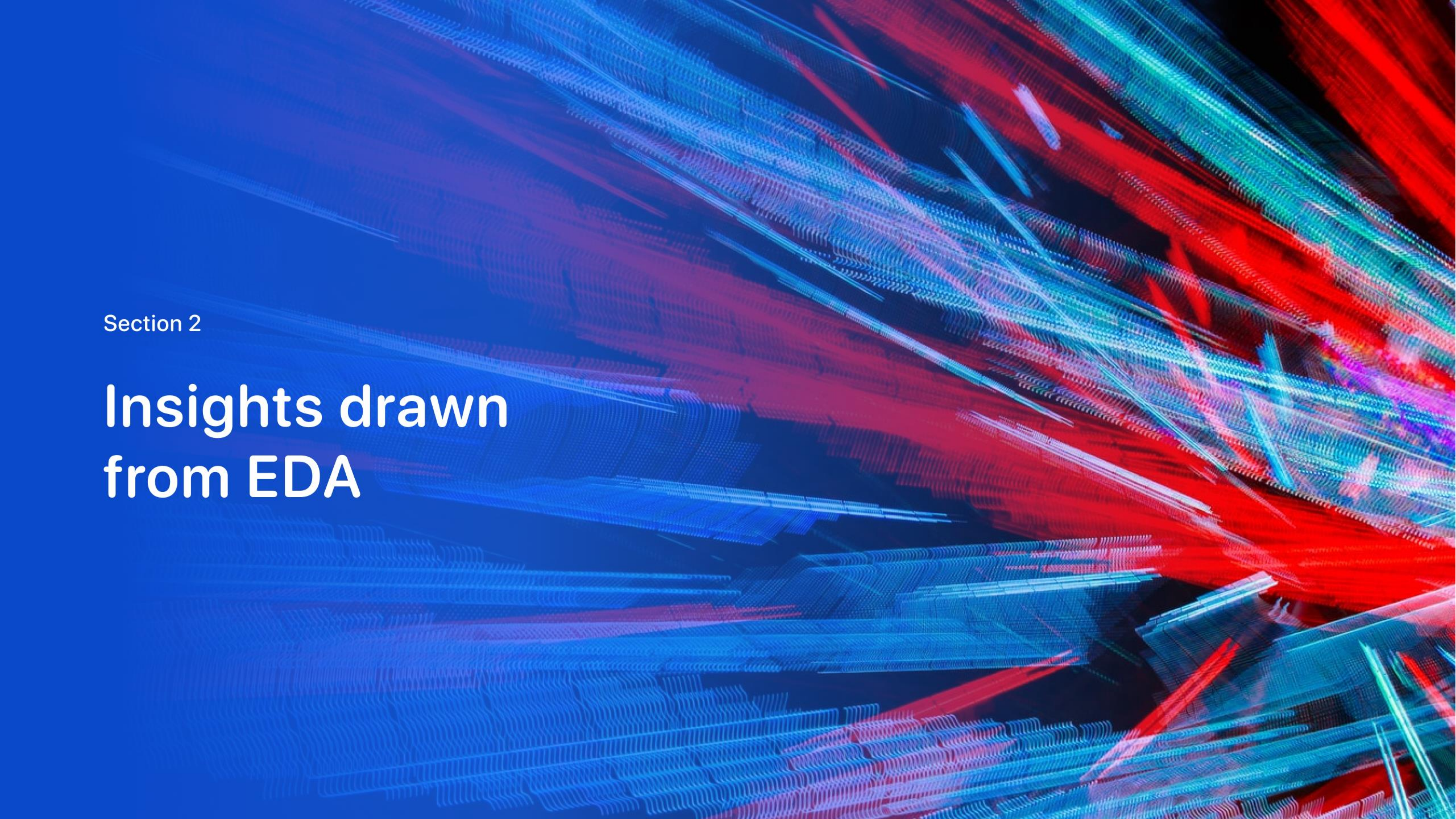
Determine the best performing model

Calculate the Jaccard score, F1 score, and utilize the accuracy from the evaluation to compare which type of classification performed the best.

Link to GitHub Notebook

Results

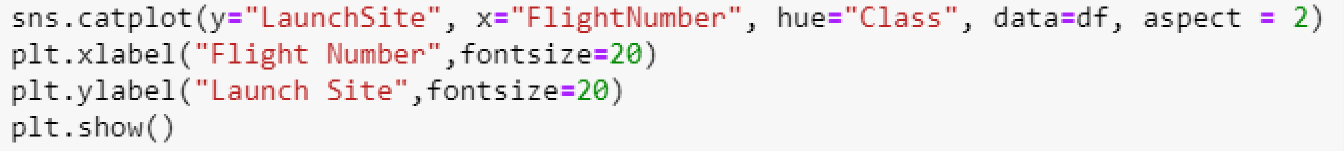
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results



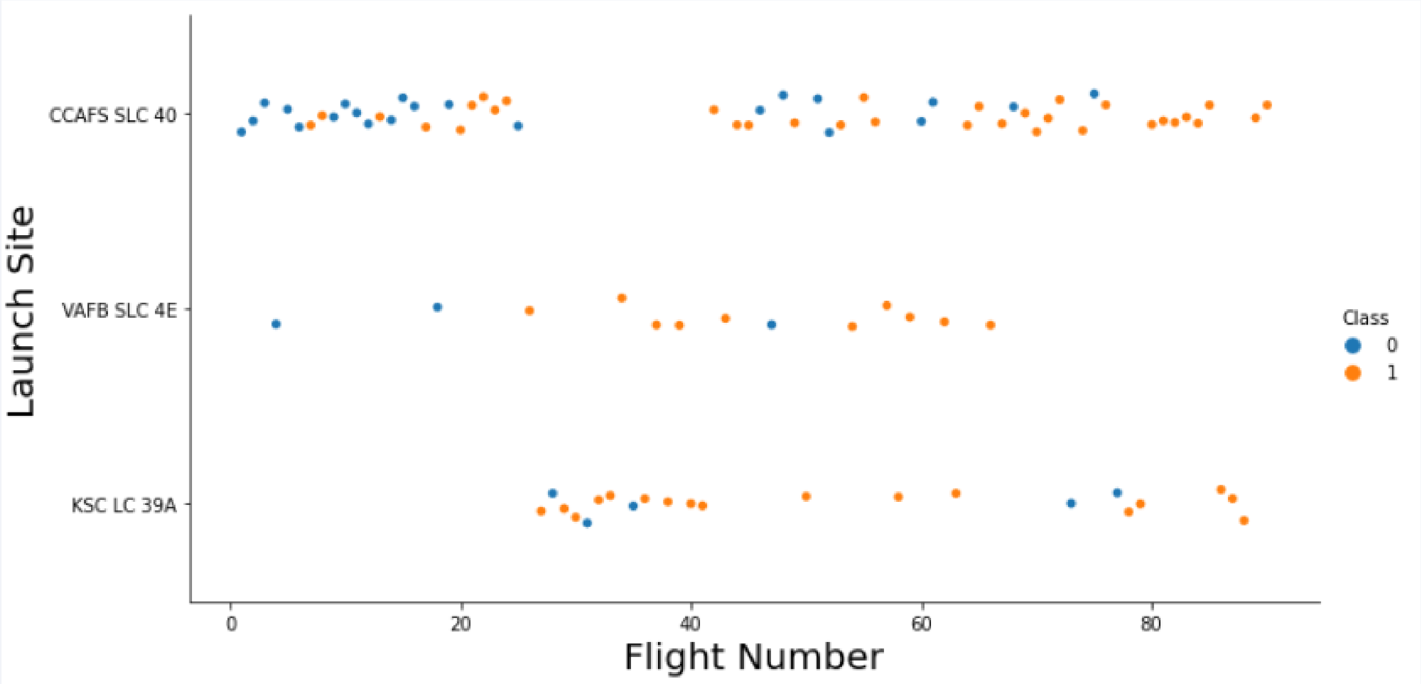
Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

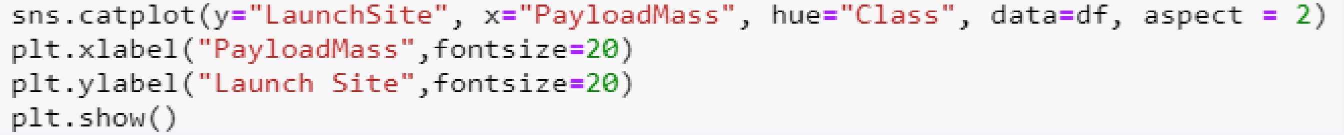


```
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 2)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```

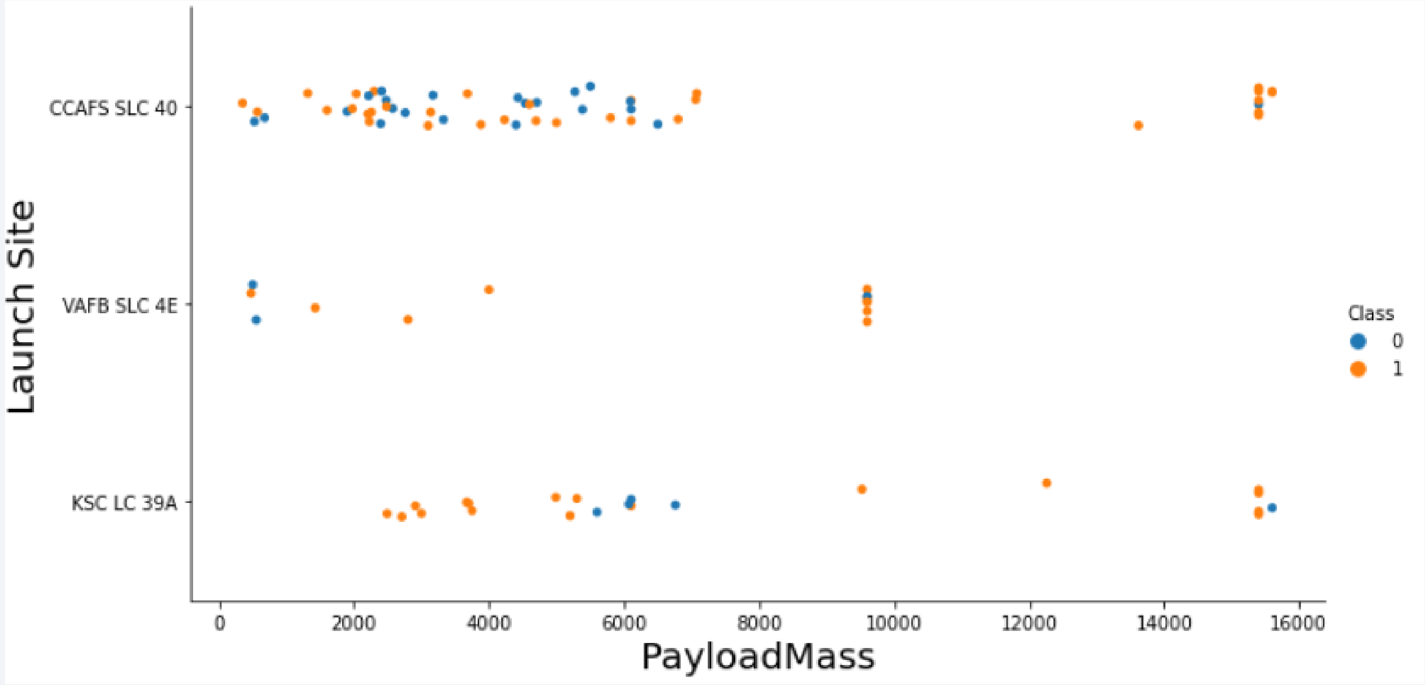


As the number of flights increases at each launch site the success rate increases.

Payload vs. Launch Site



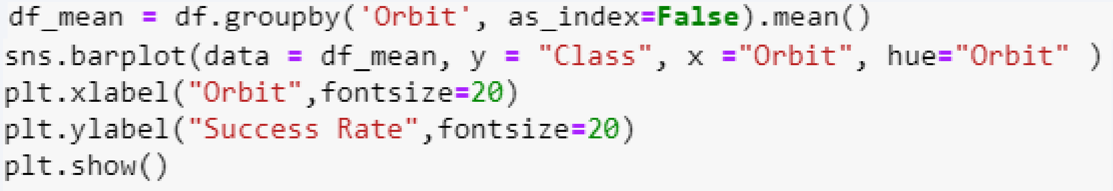
```
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 2)
plt.xlabel("PayloadMass",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



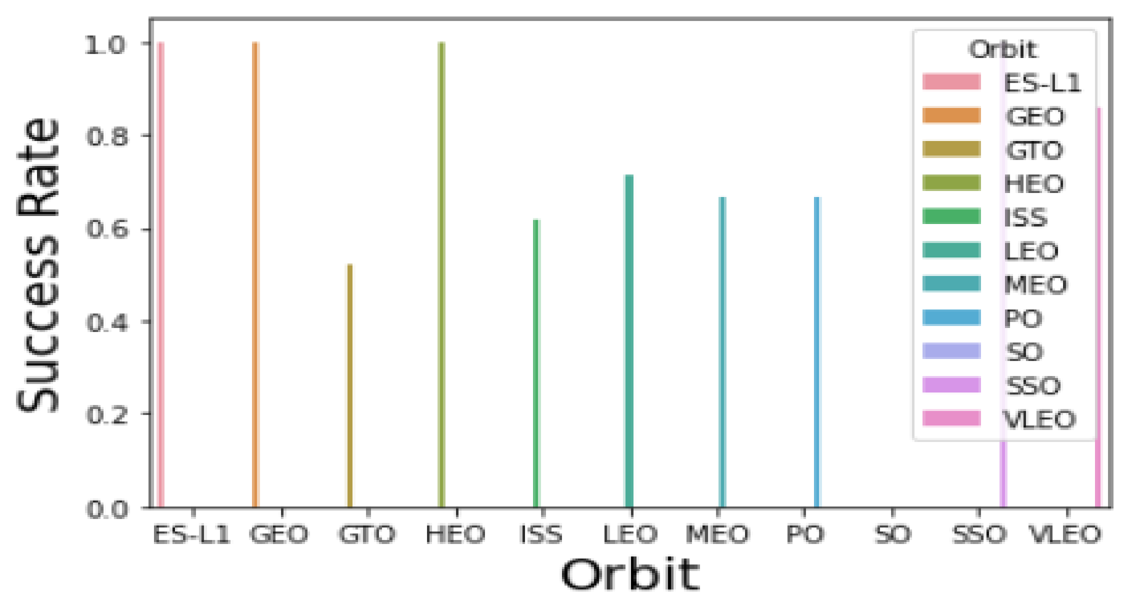
There seems to be good success with most payload masses from these three launch sites. CCAFS SLC 40 is very successful at payloads > 14000 and VAFB SLC 4E has consistent success with payloads ~ 10000.

There doesn’t seem to be enough of a correlation to decide if the launch site is dependant on payload mass for a successful launch.

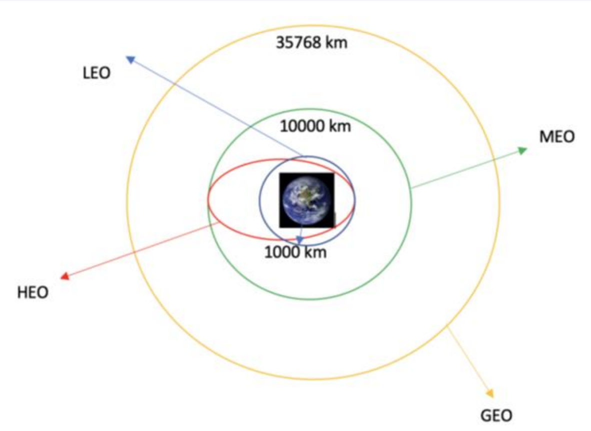
Success Rate vs. Orbit Type



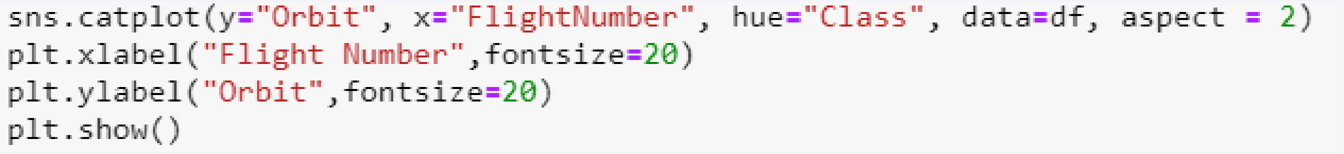
```
df_mean = df.groupby('Orbit', as_index=False).mean()
sns.barplot(data = df_mean, y = "Class", x ="Orbit", hue="Orbit" )
plt.xlabel("Orbit",fontsize=20)
plt.ylabel("Success Rate",fontsize=20)
plt.show()
```



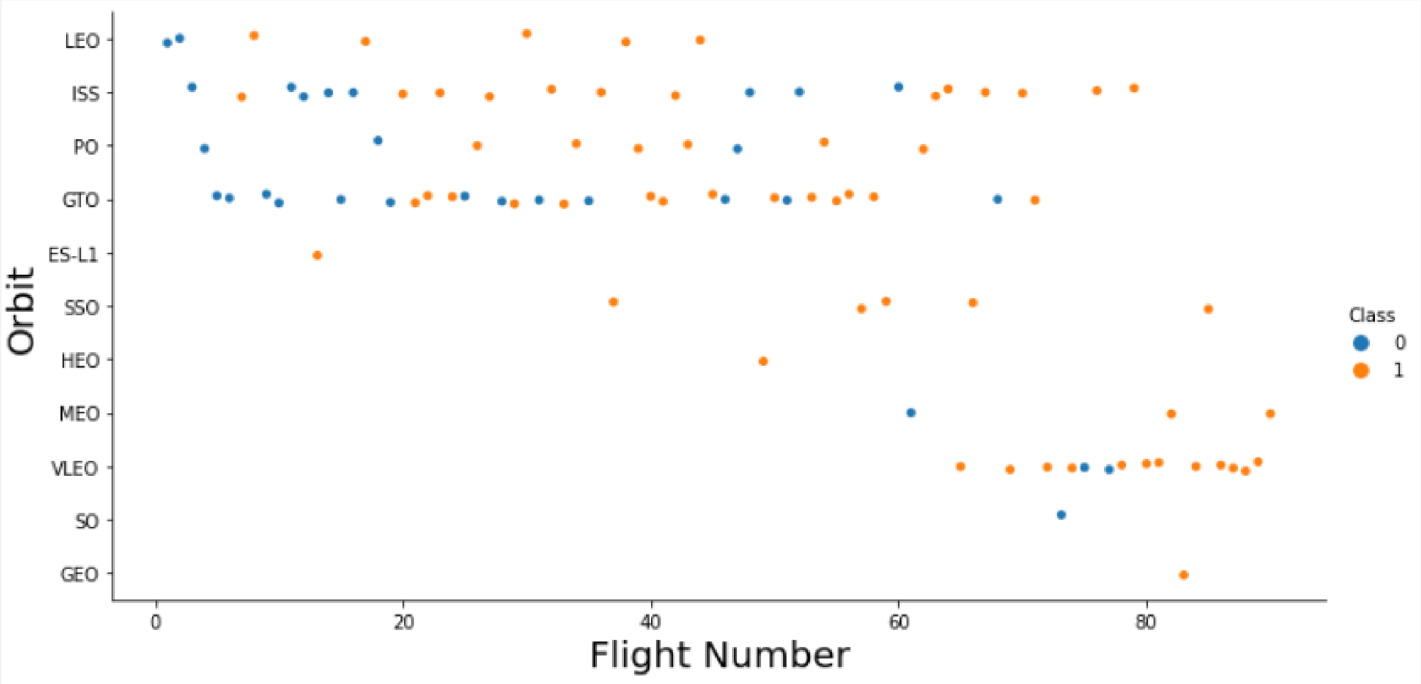
Orbits ES-L1, GEO,HEO, and SSO had 100% success rates. Other orbits ranged between 40 & 60% success rates.



Flight Number vs. Orbit Type

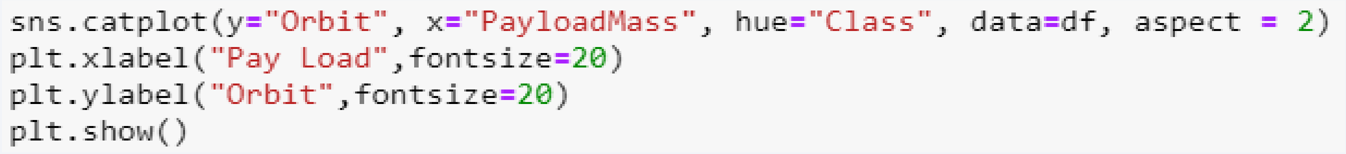


```
sns.catplot(y="Orbit", x="FlightNumber", hue="Class", data=df, aspect = 2)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```

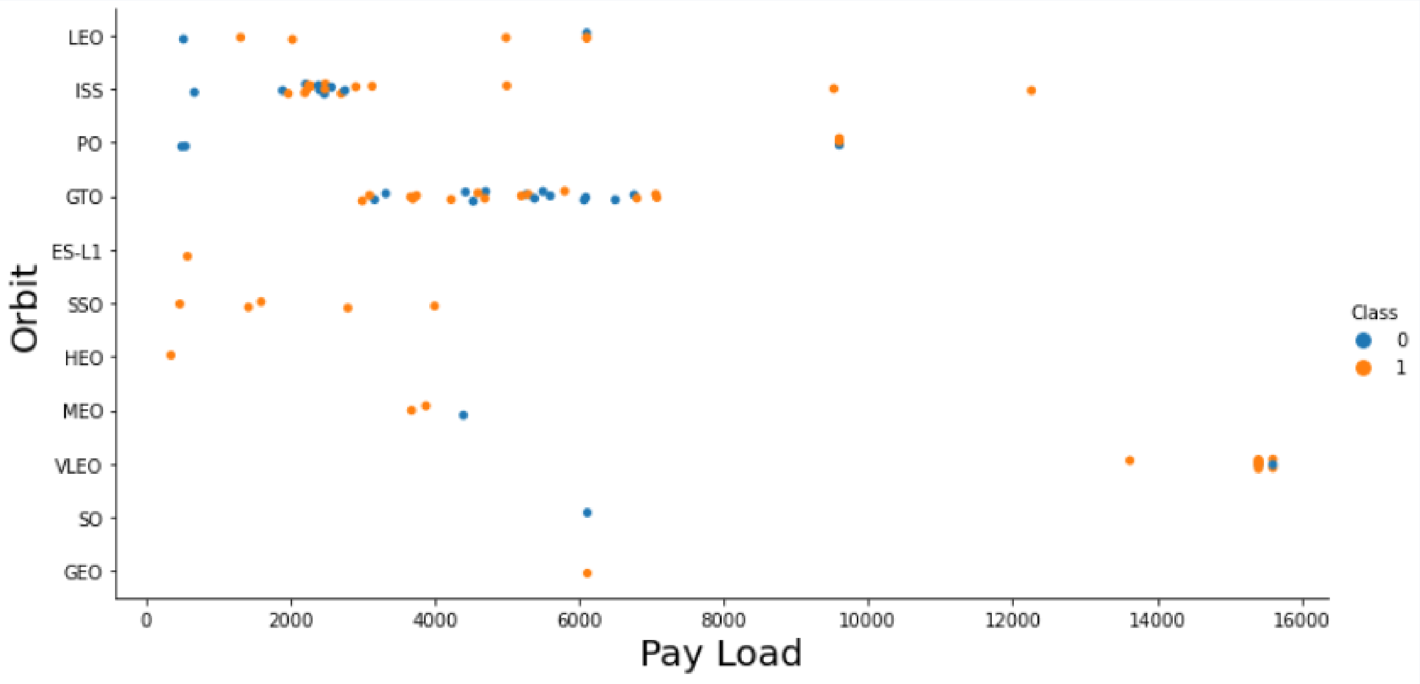


- Targeting the LEO orbit has a good success rate regardless of flight number however it doesn't appear to have been used with later flights.
- Launching to the VLEO orbit has been very successful but only targeted in more recent flights.
- Other orbits have a mixed success rate from launch or too small a statistical base to draw a conclusion.

Payload vs. Orbit Type



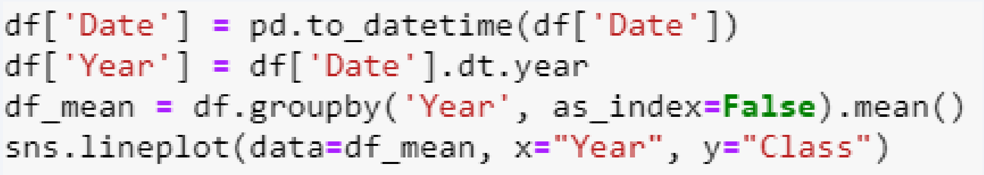
```
sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df, aspect = 2)
plt.xlabel("Pay Load",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



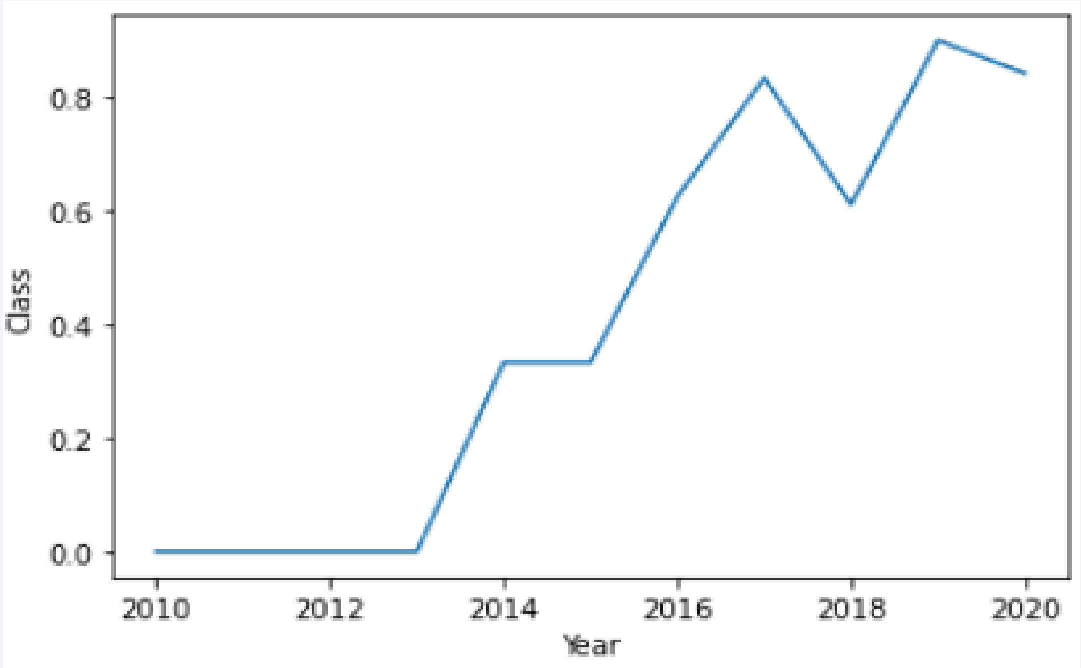
The heaviest payloads targeted the VLEO orbit with most launches successful. Payload has a negative influence on the GTO orbit. Similarly, for the SSO ad HEO orbits although these don't have many launches to them.

Although samples are limited it can be seen that successful launches were made with heavier payloads to the VLEO and ISS orbits.

Launch Success Yearly Trend



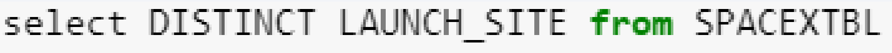
```
df['Date'] = pd.to_datetime(df['Date'])
df['Year'] = df['Date'].dt.year
df_mean = df.groupby('Year', as_index=False).mean()
sns.lineplot(data=df_mean, x="Year", y="Class")
```



- The launch success rate has increased over time since 2013.
- There were no successful launches prior to 2013.
- Although 2018 and 2020 still had large success rate they were less successful than adjacent years.

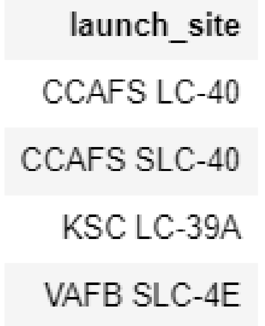
All Launch Site Names

SQL Query



```
select DISTINCT LAUNCH_SITE from SPACEXTBL
```

Query Result



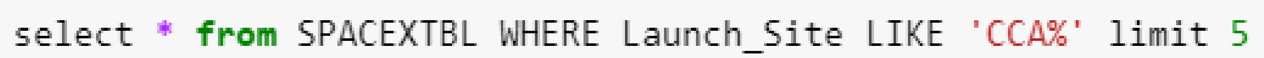
launch_site
CCAFS LC-40
CCAFS SLC-40
KSC LC-39A
VAFB SLC-4E

Query Explanation

The “DISTINCT” query allows unique launch site names to be found – 4 individual sites were determined.

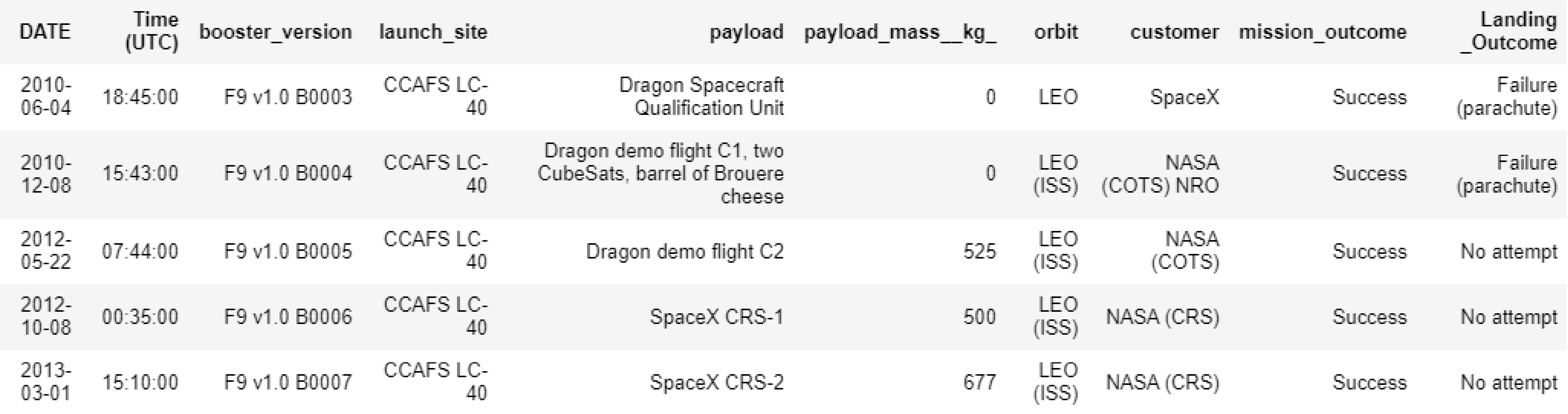
Launch Site Names Begin with 'CCA'

SQL Query



```
select * from SPACEXTBL WHERE Launch_Site LIKE 'CCA%' limit 5
```

Query Result



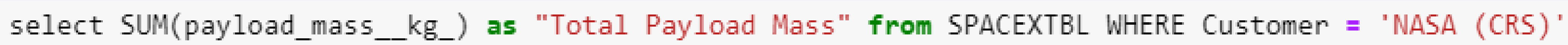
DATE	Time (UTC)	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	Landing _Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
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
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Query Explanation

Five launch sites beginning with “CCA” were determined using the “LIKE” query. The list was limited to the top 5 search results therefore more probably exist.

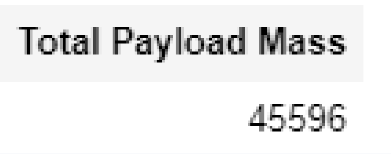
Total Payload Mass Carried by Boosters Launched by NASA (CRS)

SQL Query



```
select SUM(payload_mass__kg_) as "Total Payload Mass" from SPACEXTBL WHERE Customer = 'NASA (CRS)'
```

Query Result



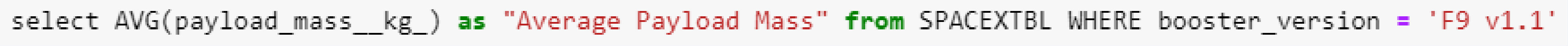
Total Payload Mass
45596

Query Explanation

The query utilizing “SUM” was used to provide the total of the payload mass.
The query was limited to those boosters launched for the customer NASA (CRS)

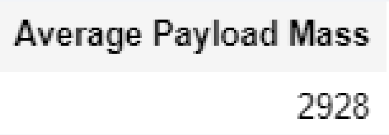
Average Payload Mass Carried by Booster Version F9 v1.1

SQL Query



```
select AVG(payload_mass__kg_) as "Average Payload Mass" from SPACEXTBL WHERE booster_version = 'F9 v1.1'
```

Query Result



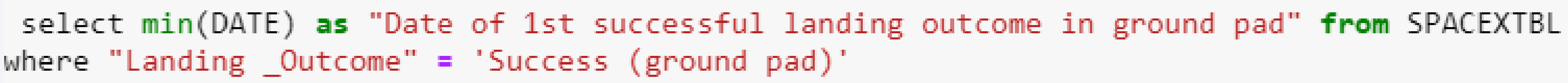
Average Payload Mass
2928

Query Explanation

The query utilizing “AVG” was used to provide the average of the payload mass.
The query was limited to the booster version F9 v1.1

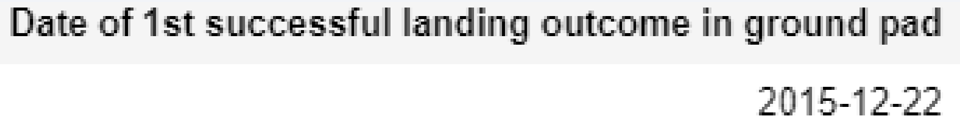
Date of First Successful Landing Outcome on Ground Pad

SQL Query



```
 select min(DATE) as "Date of 1st successful landing outcome in ground pad" from SPACEXTBL
where "Landing _Outcome" = 'Success (ground pad)'
```

Query Result



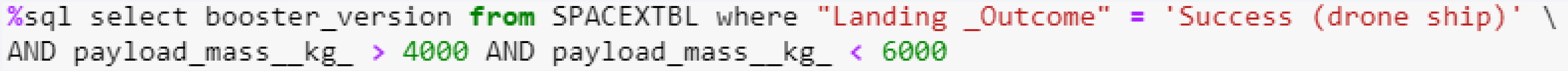
Date of 1st successful landing outcome in ground pad
2015-12-22

Query Explanation

The query utilizing "MIN" was used to determine the data of the *first* successful landing outcome on ground pad.

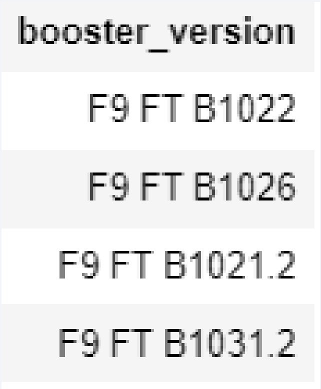
List the names of boosters which have successfully landed on drone ship where 4000 < payload mass < 6000

SQL Query



```
%sql select booster_version from SPACEXTBL where "Landing _Outcome" = 'Success (drone ship)' \
AND payload_mass__kg_ > 4000 AND payload_mass__kg_ < 6000
```

Query Result



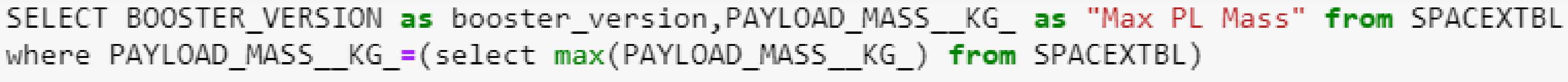
booster_version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

Query Explanation

A query was made to determine the names of the boosters that have successfully landed on a drone ship carrying a payload mass between 4000 and 6000kg.

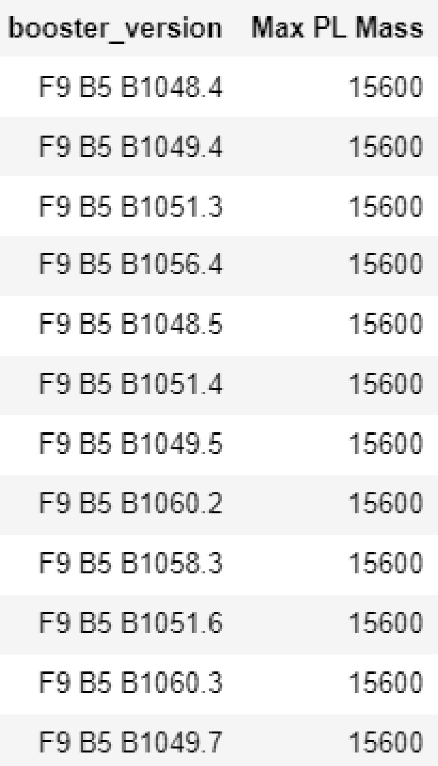
Boosters Carrying Maximum Payload Mass

SQL Query



```
SELECT BOOSTER_VERSION as booster_version,PAYLOAD_MASS__KG_ as "Max PL Mass" from SPACEXTBL
where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from SPACEXTBL)
```

Query Result



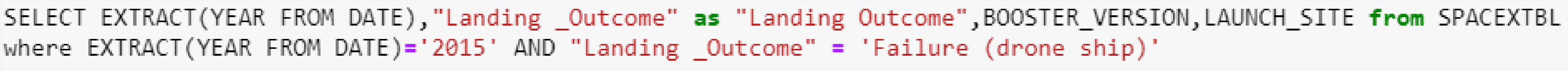
booster_version	Max PL Mass
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

Query Explanation

A sub-query was used to list those boosters which carried the *maximum* payload mass (found to be 15,600kg)

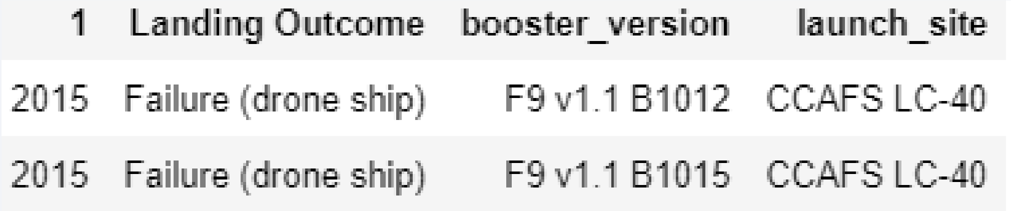
List the failed landing_outcomes in drone ship, their booster versions, and launch site names in 2015

SQL Query



```
SELECT EXTRACT(YEAR FROM DATE),"Landing _Outcome" as "Landing Outcome",BOOSTER_VERSION,LAUNCH_SITE from SPACEXTBL
where EXTRACT(YEAR FROM DATE)='2015' AND "Landing _Outcome" = 'Failure (drone ship)'
```

Query Result



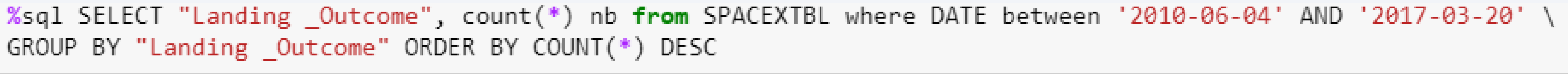
1	Landing Outcome	booster_version	launch_site
2015	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
2015	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

Query Explanation

A query was made to provide the year and the landing outcome but only those where the year was 2015 and the land outcome was a failure on a drone ship.

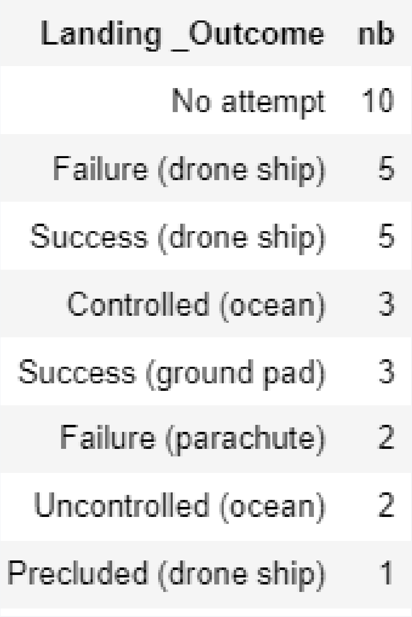
Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

SQL Query



```
%sql SELECT "Landing _Outcome", count(*) nb from SPACEXTBL where DATE between '2010-06-04' AND '2017-03-20' \
GROUP BY "Landing _Outcome" ORDER BY COUNT(*) DESC
```

Query Result



Landing _Outcome	nb
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

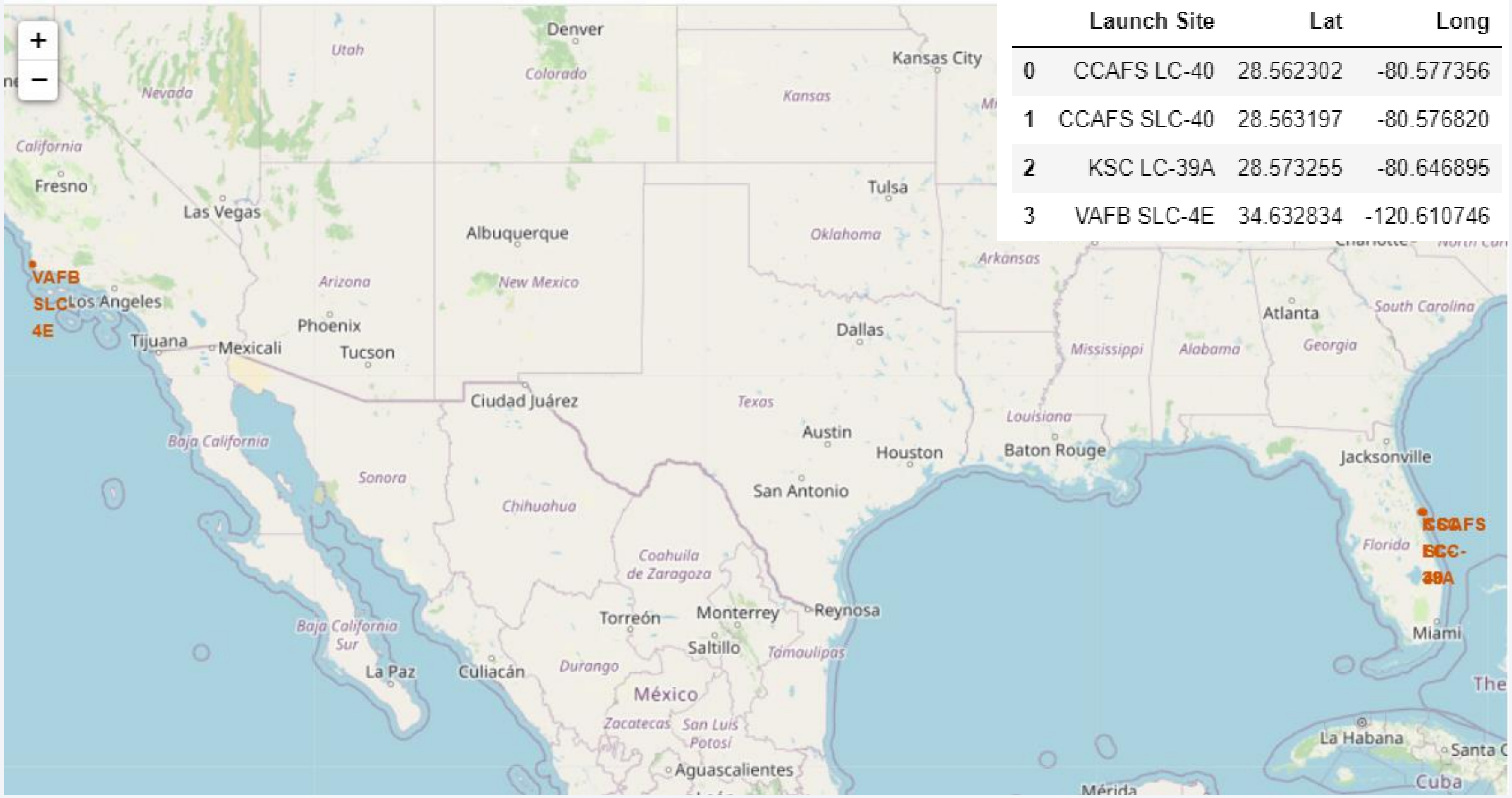
Query Explanation

The total number of landing outcomes by type were ranked between the dates listed.

Section 4

Launch Sites Proximities Analysis

Location of Launch Sites



	Launch Site	Lat	Long
0	CCAFS LC-40	28.562302	-80.577356
1	CCAFS SLC-40	28.563197	-80.576820
2	KSC LC-39A	28.573255	-80.646895
3	VAFB SLC-4E	34.632834	-120.610746

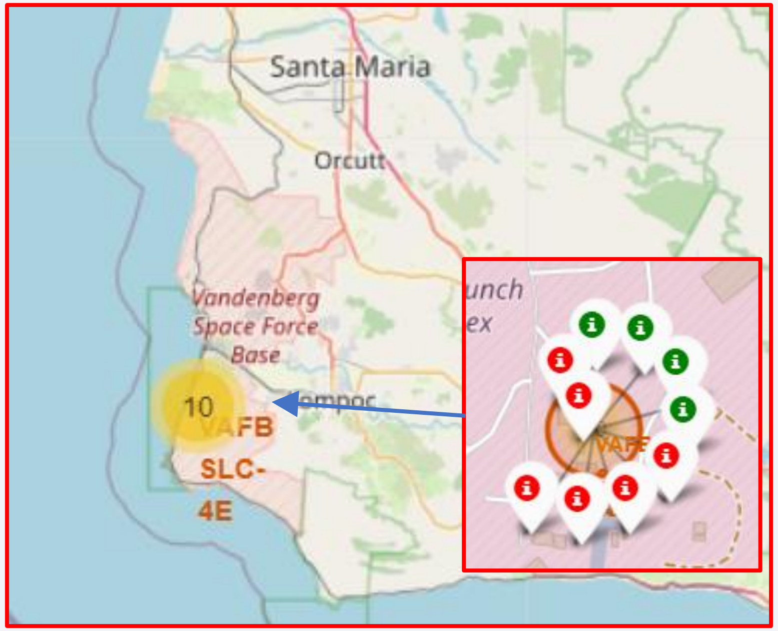
The launch sites are located in:

a) California (VAFB SLC-4E)

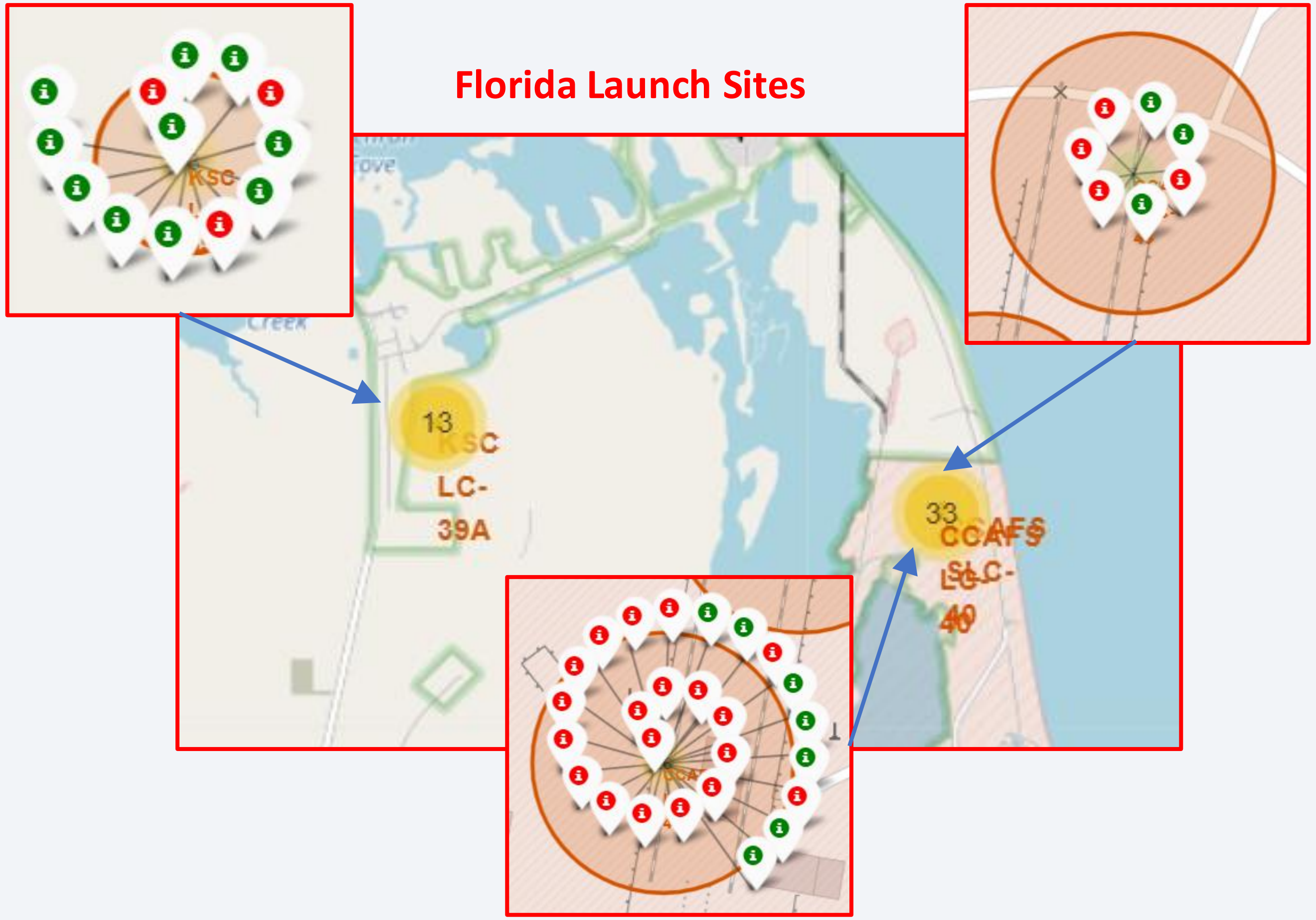
b) Florida (KSC LC-39A, CCAFS LC-40, and CCAFS SLC-40).

Colour Labelled Launch Outcomes

California Launch Site

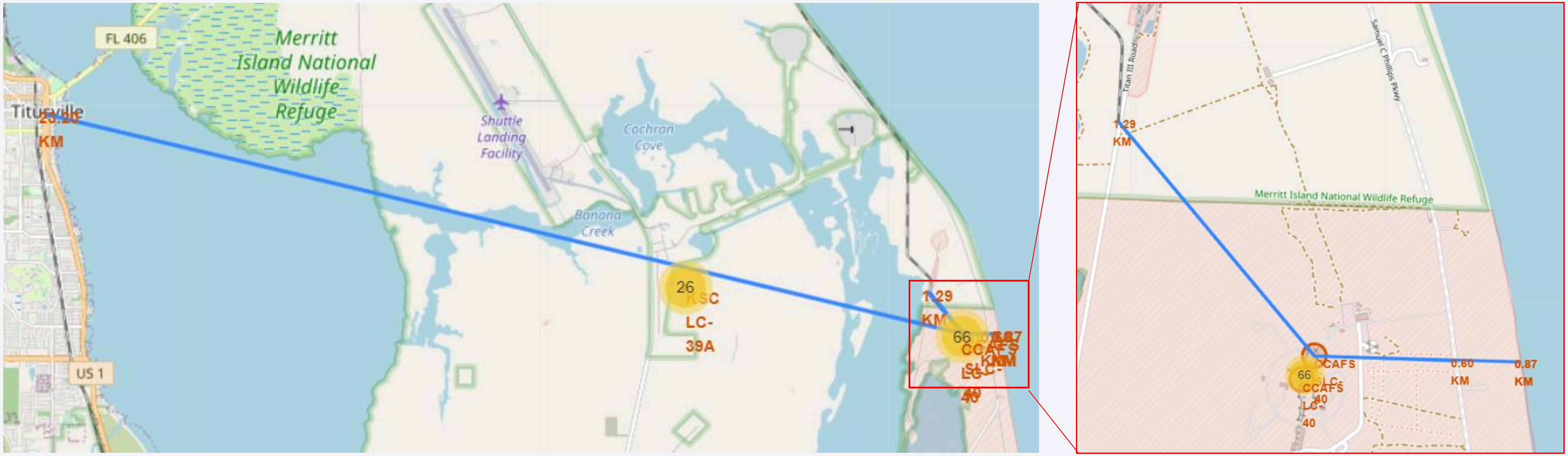


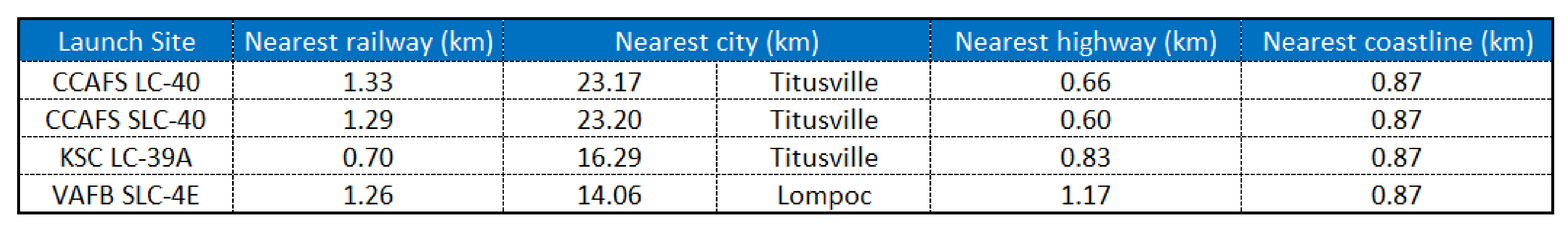
Florida Launch Sites



Launch outcomes are denoted as green (success) and red (failure).
VAFB SLC-4E (California) ~ 40% success from 10 launches
KSC LC-39A (Florida) ~ 77% success from 13 launches
CCAFS LC-40 (Florida) ~ 27% success from 26 launches
CCAFS SLC-40 (Florida) ~ 43% success from 7 launches

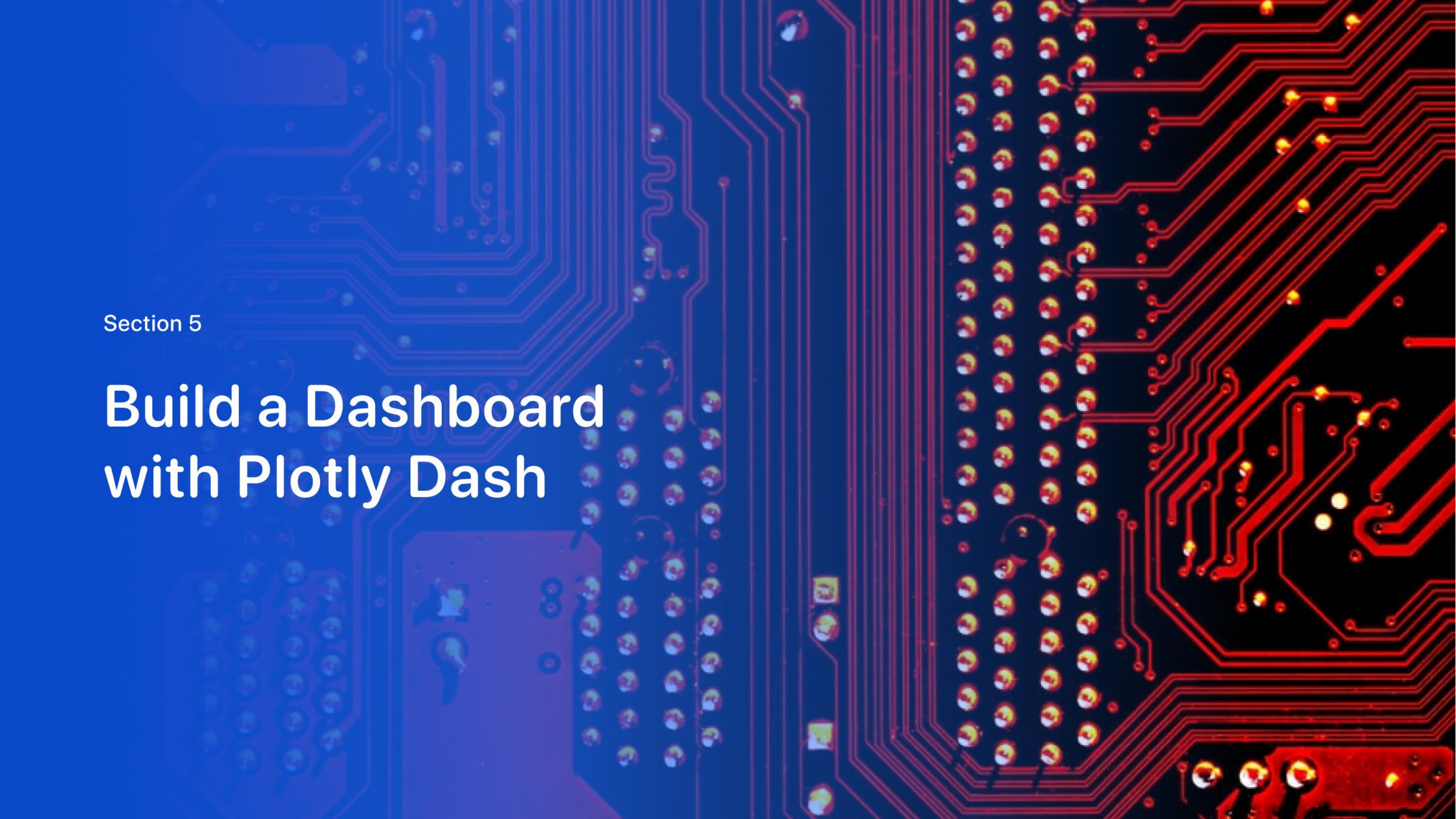
Calculate Proximity of Launch Sites to Various Landmarks





Launch Site	Nearest railway (km)	Nearest city (km)		Nearest highway (km)	Nearest coastline (km)
CCAFS LC-40	1.33	23.17	Titusville	0.66	0.87
CCAFS SLC-40	1.29	23.20	Titusville	0.60	0.87
KSC LC-39A	0.70	16.29	Titusville	0.83	0.87
VAFB SLC-4E	1.26	14.06	Lompoc	1.17	0.87

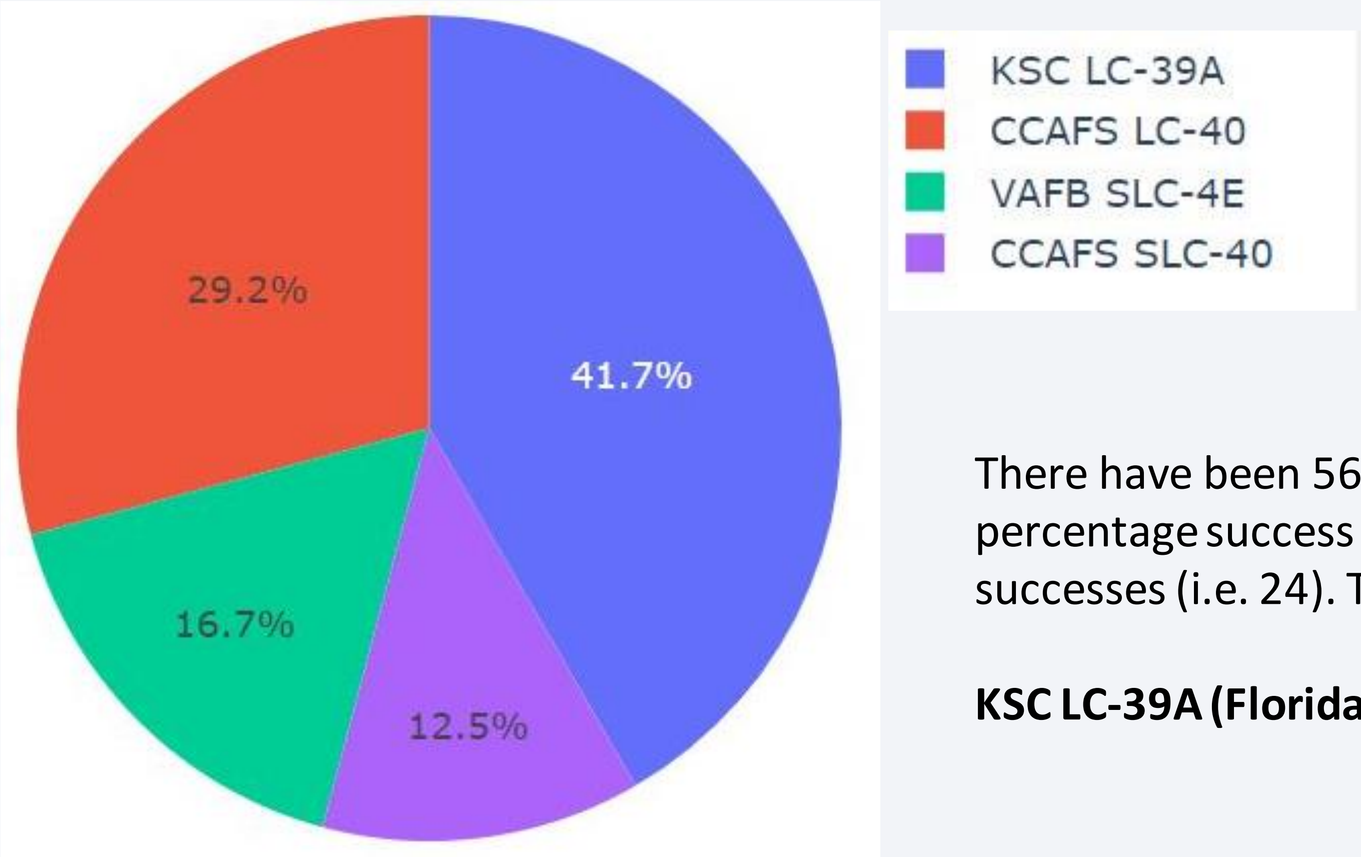
Launch sites are a consistent distance away from the coast (0.87km) and considerable distance away from the nearest city.



Section 5

Build a Dashboard with Plotly Dash

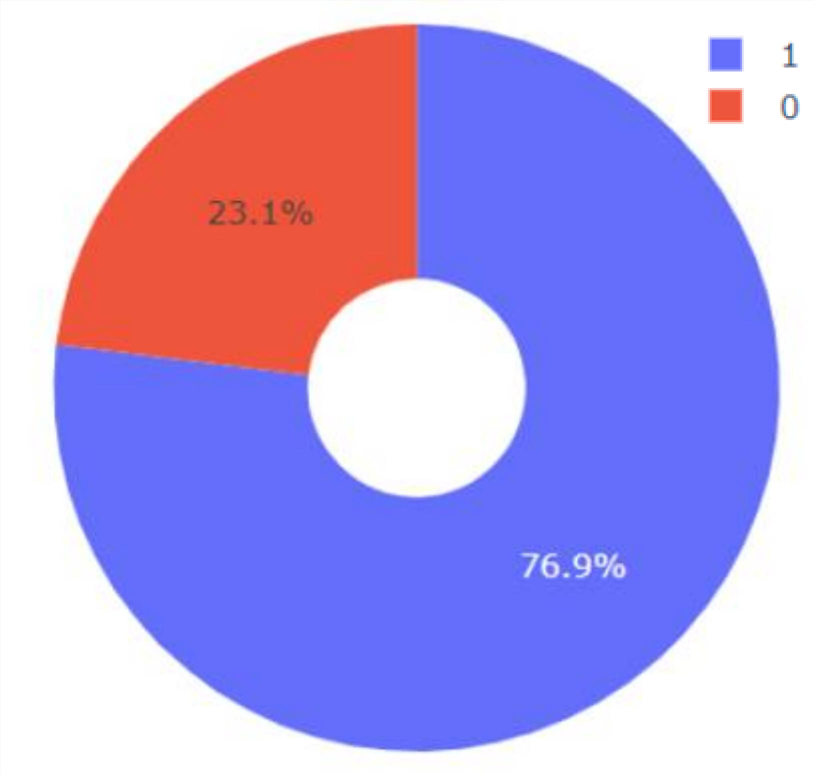
SpaceX Launch Records Dashboard: Launch Success for all Sites



There have been 56 total launches of which 24 were successful. The percentage success from each launch site is given by #successes/total successes (i.e. 24). This is the value given in the pie chart.

KSC LC-39A (Florida) had the most successful launches of all four sites.

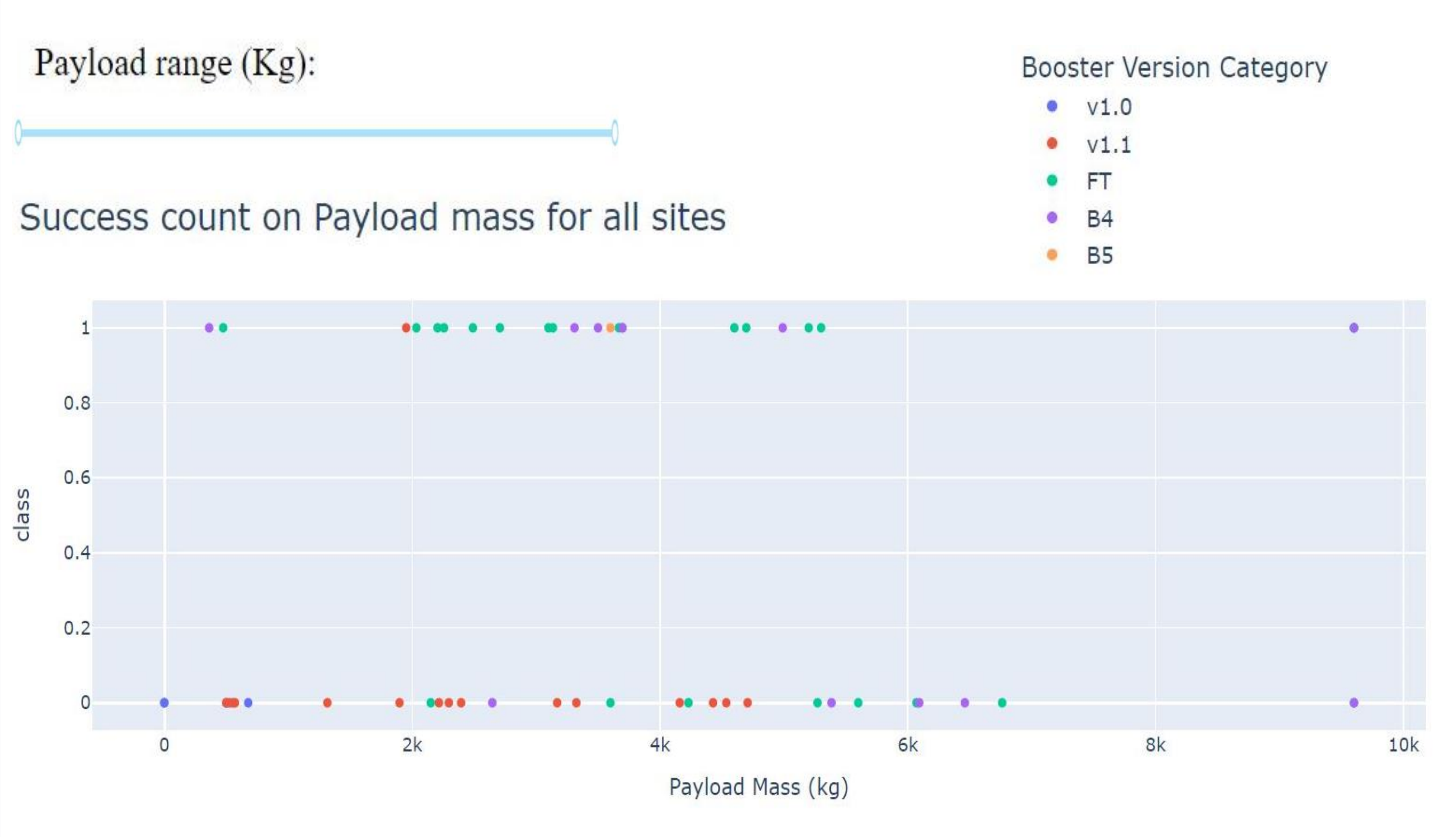
SpaceX Launch Records Dashboard: Launch Success Ratio



Launch site KSC LC-39A had the highest success rate at 76.9% success rate

Other launch success rates in order of descending order are:
CCAFS SLC-40 ~ 43% success
VAFB SLC-4E ~ 40% success
CCAFS LC-40 ~ 27% success

SpaceX Launch Records Dashboard: Payload vs Launch Outcome



Successful launches tend to occur with payload masses between 2000-5000 kg with the majority of these being with the *FT* booster. The booster *v1.1* tended to be associated with a lot of launch failures within this payload range.

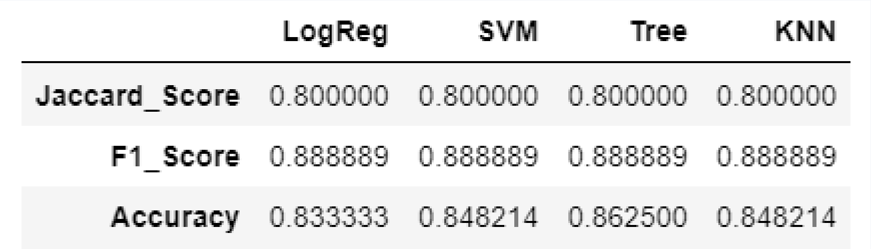
Section 6

Predictive Analysis (Classification)

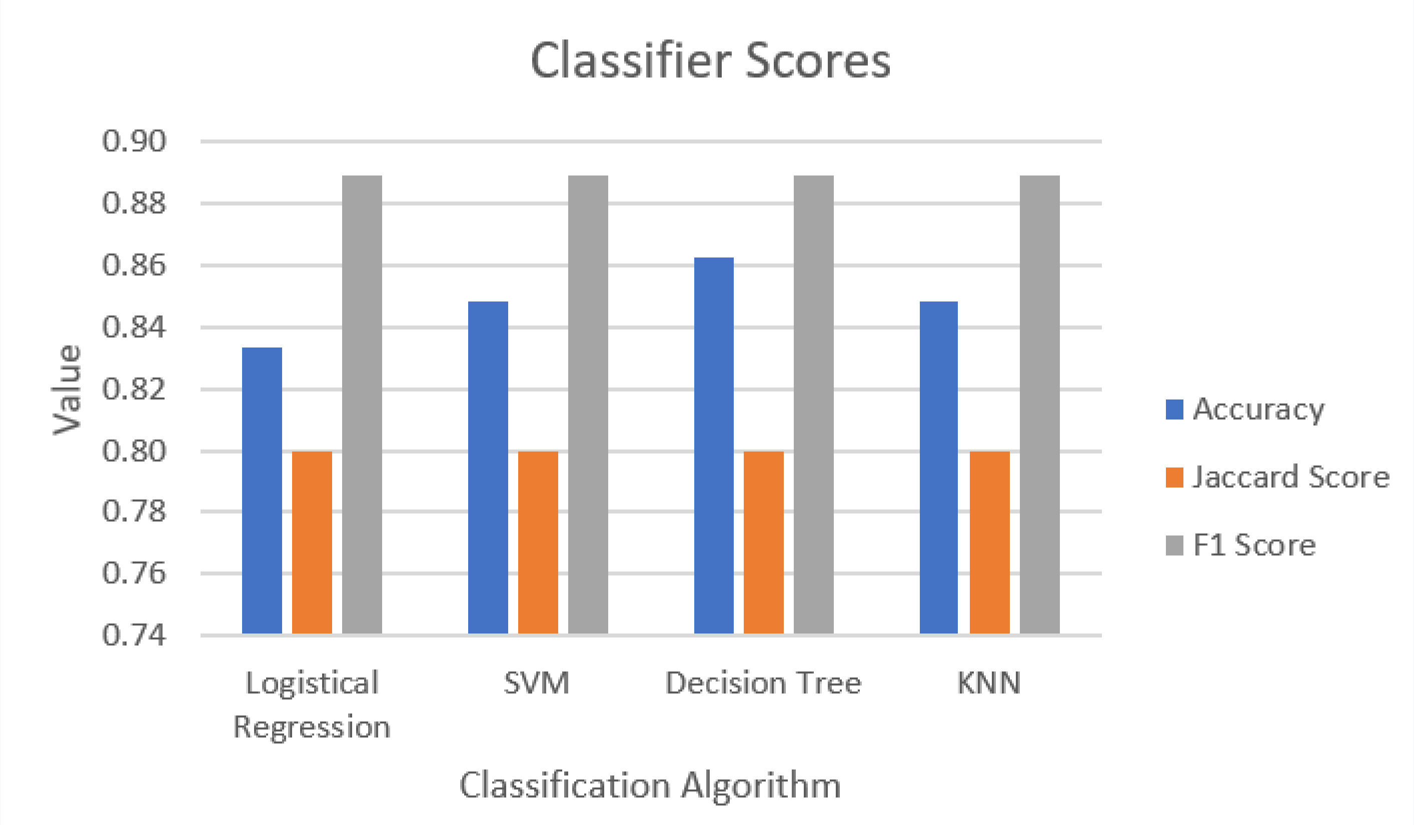
Classification Accuracy

The following features were used for predictive analysis using four different classification methods :
Flight Number, Date, Booster Version, Payload Mass, Orbit, Launch Site, Outcome, Flights, Grid Fins, Reused, Legs, Landing Pad, Block, Reused Count, Serial, Longitude, Latitude, and Class

Various measures were used to determine the effectiveness of each classifier: Jaccard Score, F1-Score, and Accuracy.



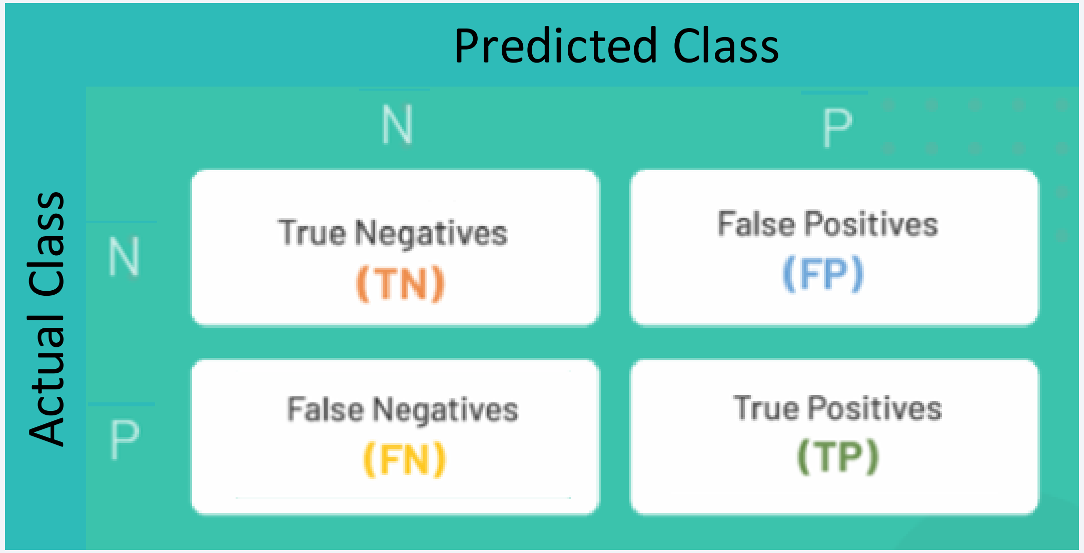
	LogReg	SVM	Tree	KNN
Jaccard_Score	0.800000	0.800000	0.800000	0.800000
F1_Score	0.888889	0.888889	0.888889	0.888889
Accuracy	0.833333	0.848214	0.862500	0.848214



There is no discrimination between the different classification algorithms using the Jaccard or F1 - Scores.

Using the accuracy measure the decision tree classifier turns out to be the best algorithm for training the model.

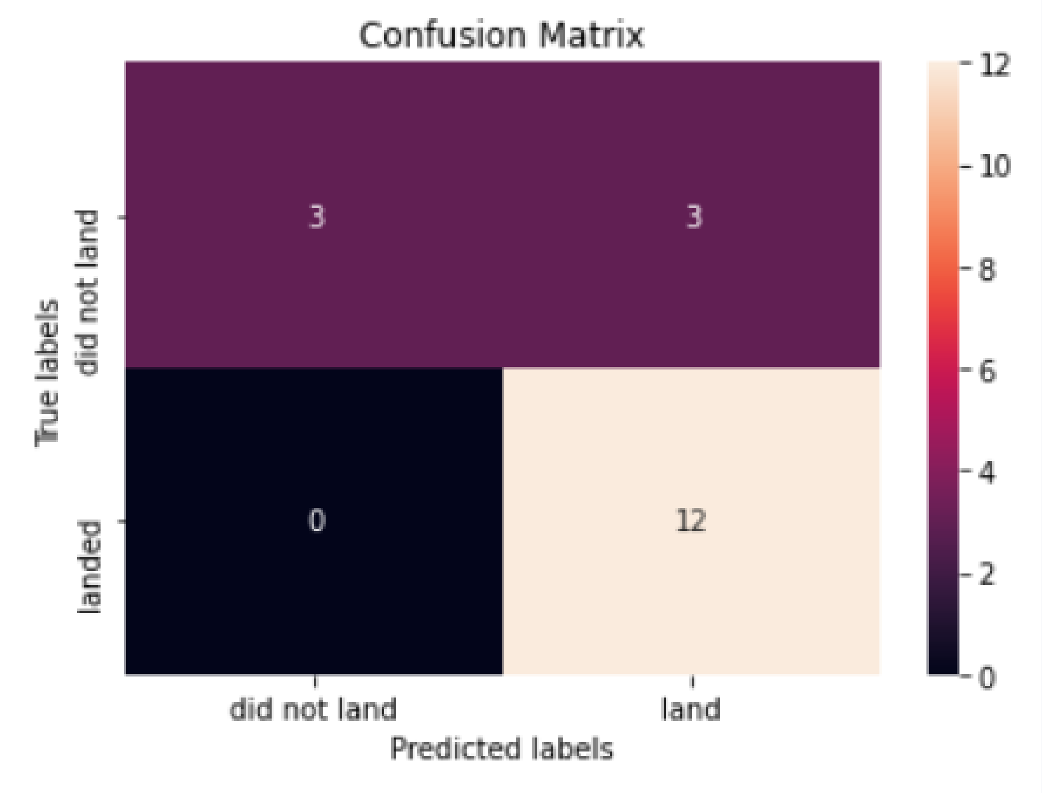
Confusion Matrix for the Decision Tree



Predicted Class

Actual Class

	N	P
N	True Negatives (TN)	False Positives (FP)
P	False Negatives (FN)	True Positives (TP)



The confusion matrix for the decision tree indicates that
The only problem is with false positives where the model predicts a successful landing yet the data says the landing was unsuccessful.

For interest, the decision tree classifier has a precision of 80% and a recall of 100%. This means that 80% of the results are relevant and 100% of the relevant results are correctly classified.

Conclusions

- The success rate for SpaceX launches is heavily reliant on the time since the first launch. Based on the data used the number of unsuccessful launches will decline as more launches are performed in the future.
- Other factors which have a strong influence on a successful launch are:
  - Launch site – if a launch occurs from the *KSCLC-39A* site it will be more successful.
  - Orbit – if the target orbit is *ES-L1*, *GEO*, *HEO*, or *SSO* then there is a good chance of a successful launch.
  - Payload Mass – Smaller payload masses result in a more successful launches: specifically, between 2000 and 5000kg.
  - Booster type – The *F1* booster will have a greater success for a launch in the aforementioned payload mass range.
- The decision tree classification algorithm performs the best when predicting outcomes with this dataset.

Appendix