IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Wigih Prasetyo

22 November 2021

Outline

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

Executive Summary

- Summary of methodologies
    1. Data collection
    2. Data wrangling
    3. EDA with data visualization
    4. EDA with SQL
    5. Building an interactive map with Folium
    6. Building a Dashboard with Plotly Dash
    7. Predictive analysis (Classification)
- Summary of all results
    1. Exploratory data analysis results
    2. Interactive analytics demo in screenshots
    3. Predictive analysis results

Introduction

- Project background and context

  In this capstone, we will predict if the Falcon 9 first stage will land successfully. 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. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch. In this lab, you will collect and make sure the data is in the correct format from an API. The following is an example of a successful and launch

- Problems you want to find answers

  1. What is the influences if the rocket will land successfully?
  2. What is the relationship of certain rocket variables with success rate of landing ?
  3. What kind of conditions does the SpaceX must achieve to get the best rt rocket success landing rate ?

Section 1

Methodology

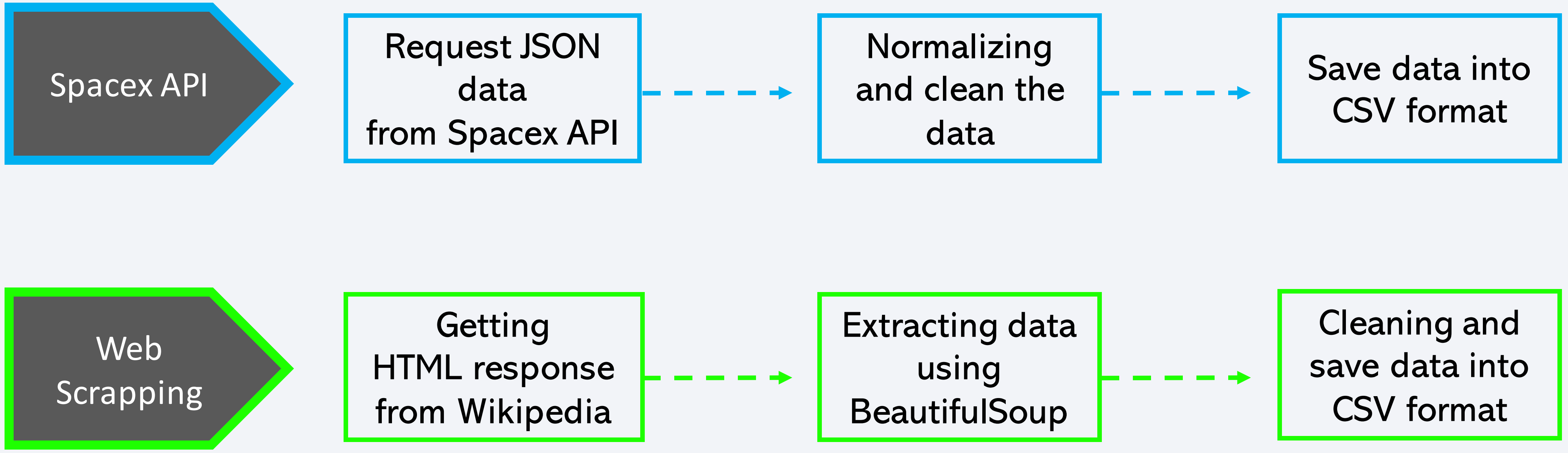
Methodology

Executive Summary

- Data collection methodology:
  1. SpaceX Rest API
  2. Web scrapping from Wikipedia
- Perform data wrangling
  - One Hot Encoding data fields for Machine Learning and dropping irrelevant columns
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
  - Plotting: Scatter Graphs, Bar Graphs to show relationships between variables to show patterns of data
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

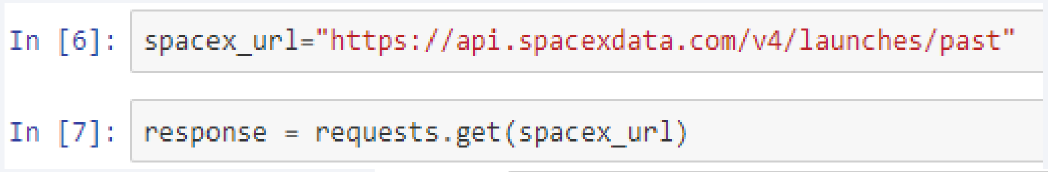
Data Collection

The data is collected from the SpaceX REST API. Then the API will give us data about launches including information about the rocket used, payload d elivered, launch specifications, landing specifications, and landing outcome. Another data is collected from Wikipedia using BeautifulSoup package.

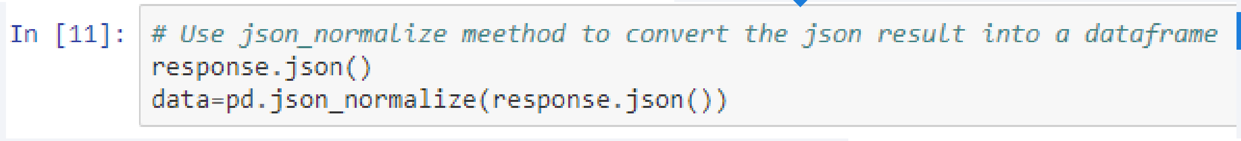


Spacex API
Request JSON data from Spacex API
Normalizing and clean the data
Save data into CSV format
Web Scrapping
Getting HTML response from Wikipedia
Extracting data using BeautifulSoup
Cleaning and save data into CSV format

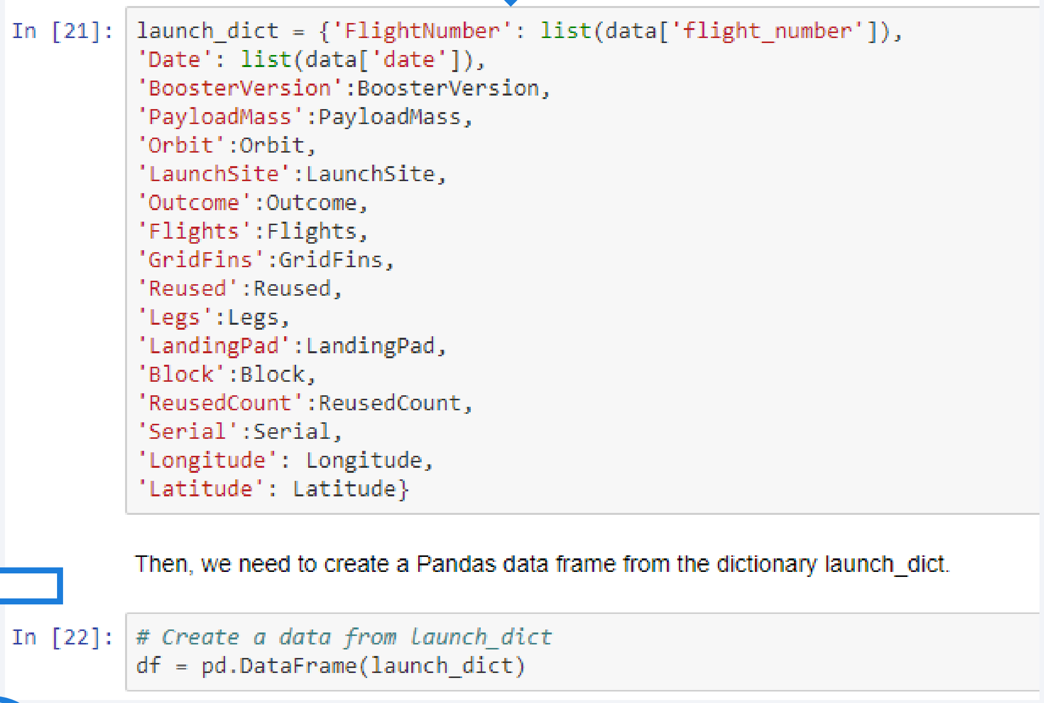
Data Collection – Spacex API



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

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



```
In [11]: # Use json_normalize meethod to convert the json result into a dataframe
         response.json()
         data=pd.json_normalize(response.json())
```

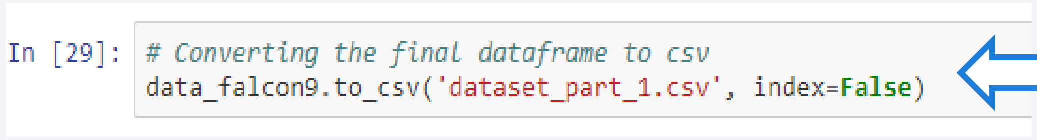


```
In [21]: launch_dict = {'FlightNumber': list(data['flight_number']),
         'Date': list(data['date']),
         'BoosterVersion':BoosterVersion,
         'PayloadMass':PayloadMass,
         'Orbit':Orbit,
         'LaunchSite':LaunchSite,
         'Outcome':Outcome,
         'Flights':Flights,
         'GridFins':GridFins,
         'Reused':Reused,
         'Legs':Legs,
         'LandingPad':LandingPad,
         'Block':Block,
         'ReusedCount':ReusedCount,
         'Serial':Serial,
         'Longitude': Longitude,
         'Latitude': Latitude}
```

Then, we need to create a Pandas data frame from the dictionary launch_dict.

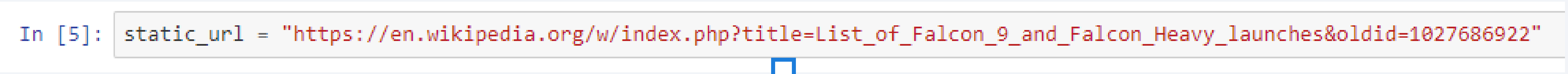
```
In [22]: # Create a data from launch_dict
         df = pd.DataFrame(launch_dict)
```

[Link to the notebook]

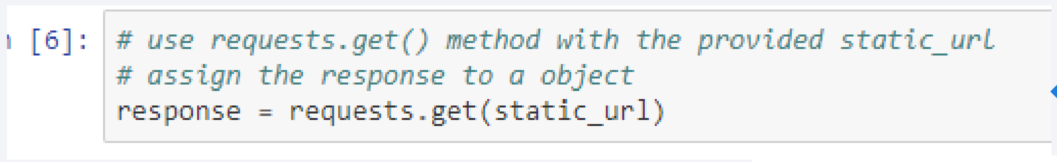


```
In [29]: # Converting the final dataframe to csv
         data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

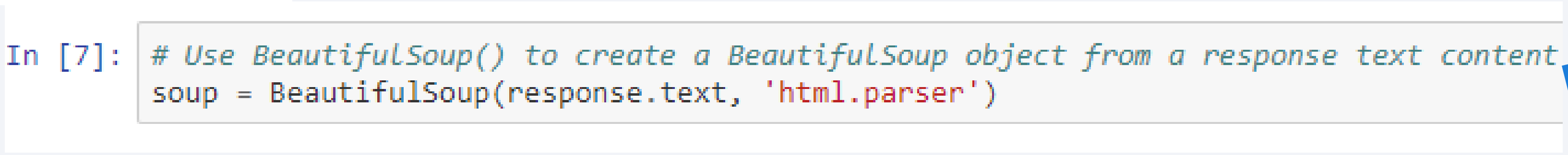
Data Collection - Scraping



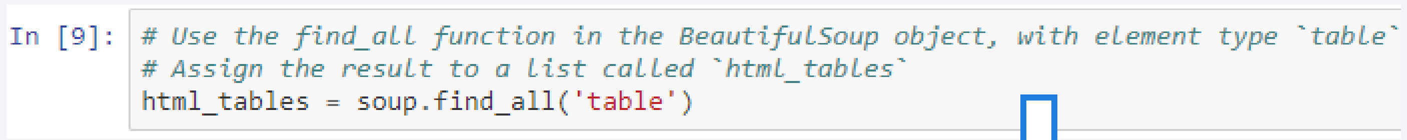
```
In [5]: static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```



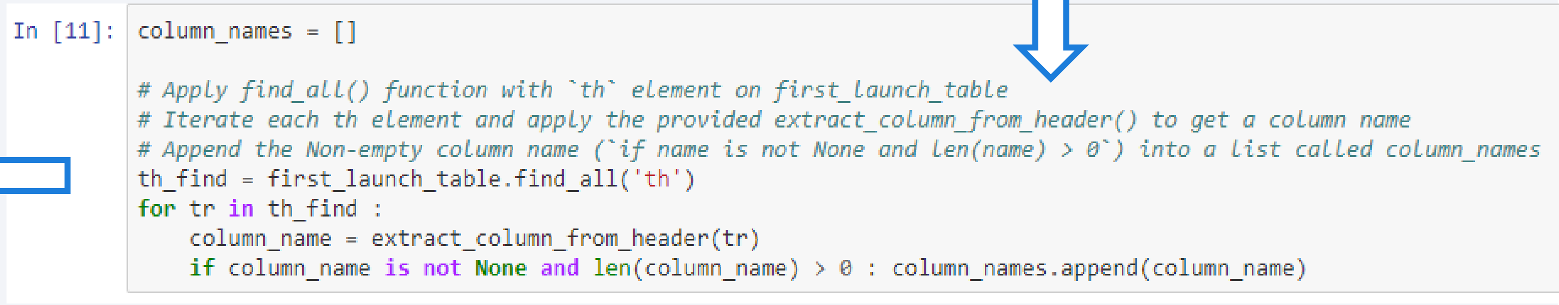
```
[6]: # use requests.get() method with the provided static_url
     # assign the response to a object
     response = requests.get(static_url)
```



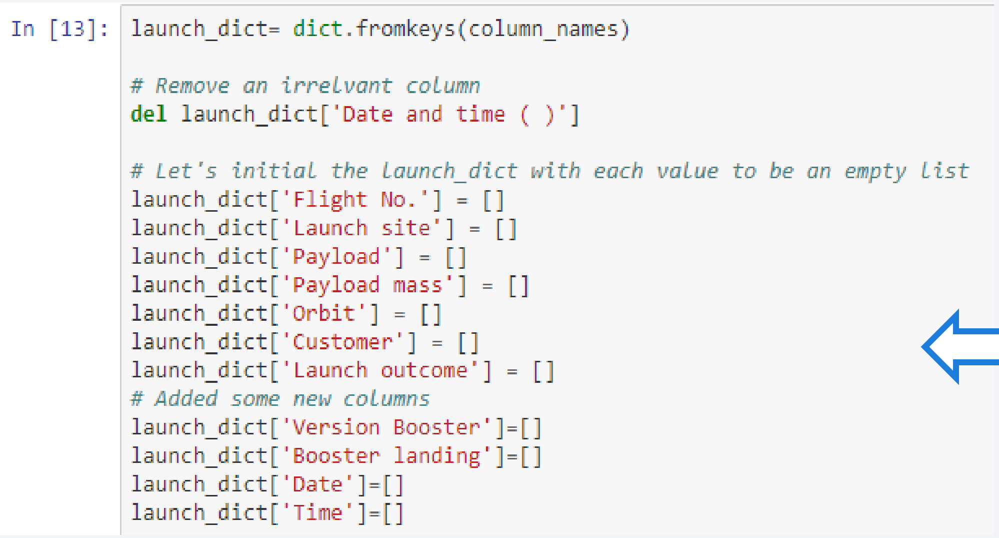
```
In [7]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
        soup = BeautifulSoup(response.text, 'html.parser')
```



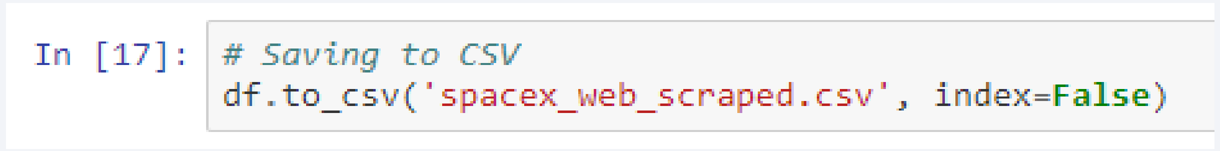
```
In [9]: # Use the find_all function in the BeautifulSoup object, with element type `table`
        # Assign the result to a list called `html_tables`
        html_tables = soup.find_all('table')
```



```
In [11]: 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
         th_find = first_launch_table.find_all('th')
         for tr in th_find :
             column_name = extract_column_from_header(tr)
             if column_name is not None and len(column_name) > 0 : column_names.append(column_name)
```



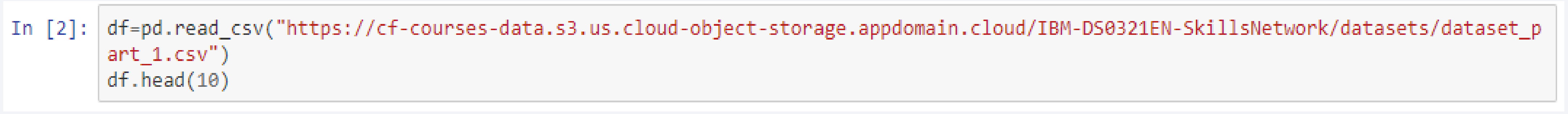
```
In [13]: launch_dict= dict.fromkeys(column_names)

         # 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']=[]
```



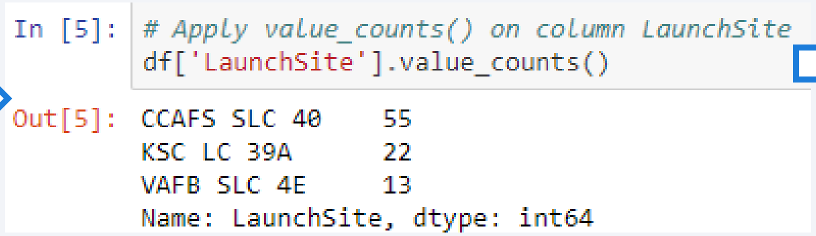
```
In [17]: # Saving to CSV
         df.to_csv('spacex_web_scraped.csv', index=False)
```

Link to the notebook

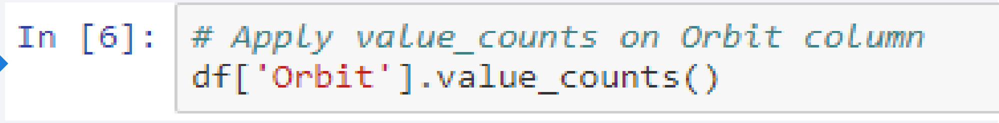
Data Wrangling



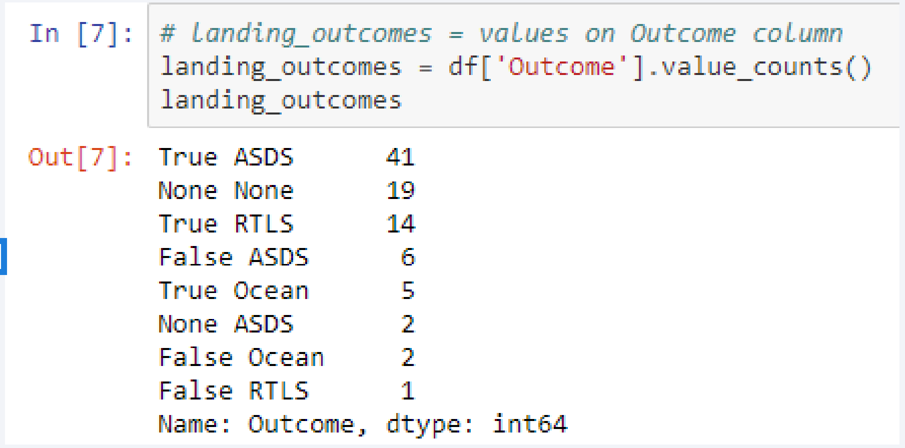
```
In [2]: df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_p
        art_1.csv")
        df.head(10)
```



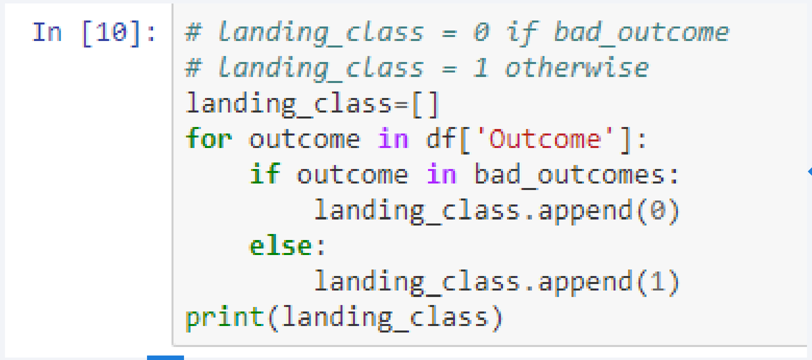
```
In [5]: # Apply value_counts() on column LaunchSite
        df['LaunchSite'].value_counts()

Out[5]: CCAFS SLC 40    55
        KSC LC 39A      22
        VAFB SLC 4E     13
        Name: LaunchSite, dtype: int64
```



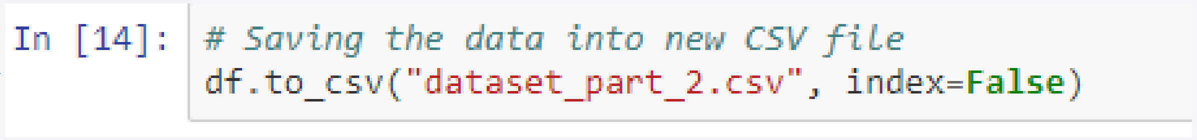
```
In [6]: # Apply value_counts on Orbit column
        df['Orbit'].value_counts()
```



```
In [7]: # landing_outcomes = values on Outcome column
        landing_outcomes = df['Outcome'].value_counts()
        landing_outcomes

Out[7]: True ASDS      41
        None None      19
        True RTLS      14
        False ASDS      6
        True Ocean      5
        None ASDS       2
        False Ocean     2
        False RTLS      1
        Name: Outcome, dtype: int64
```



```
In [10]: # landing_class = 0 if bad_outcome
         # landing_class = 1 otherwise
         landing_class=[]
         for outcome in df['Outcome']:
             if outcome in bad_outcomes:
                 landing_class.append(0)
             else:
                 landing_class.append(1)
         print(landing_class)
```



```
In [14]: # Saving the data into new CSV file
         df.to_csv("dataset_part_2.csv", index=False)
```

1. Data Analyzing
2. Calculating the number of launches on each site
3. Calculating the number and occurrence of each orbit
4. Calculating the number and occurrence of mission outcome per orbit type
5. Creating a landing outcome label from outcome column

Link to the notebook

EDA with Data Visualization

1. Scatter graphs demonstrate how one variable is influenced by another. Correlation is the term used to describe the relationship between two variables. Scatter plots are typically made up of a larger amount of data. Fight Number VS. Payload Mass, Fight Number VS. Launch Site, Payload VS. Launch Site, Orbit VS. Fight Number, Payload VS. Orbit Type, Orbit VS. Payload Mass, Orbit VS. Payload Mass, Orbit VS. Payload Mass, Orbit VS. Payload Mass, Orbit VS. Payload Mass, Orbit VS. Payload Mass, Orbit VS. Payload Mass, Orbit VS.

2. A bar graph allows to compare sets of data between different groups. On one axis, the graph shows categories, and on the other is a discrete value. The purpose is to demonstrate the connection between the two axis. Bar charts can also be used to depict significant changes in data over time. Orbit vs. Mean

3. Line graphs are valuable because they clearly display data variables and patterns, and they can aid in making predictions about the outcomes of data that has not yet been recorded.

Link to the notebook

EDA with SQL

- Connecting to the database
- Displaying the names of the unique launch sites in the space mission
- Displaying 5 records where launch sites begin with the string 'CCA'
- Displaying the total payload mass carried by boosters launched by NASA (CRS)
- Displaying average payload mass carried by booster version F9 v1.1
- Listing the date when the first successful landing outcome in ground pad was achieved
- Listing the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- Listing the total number of successful and failure mission outcomes
- Listing the names of the booster_versions which have carried the maximum payload mass. Use a subquery
- Listing the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
- Sorting the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

Link to the notebook

Build an Interactive Map with Folium

- Summarize what map objects such as markers, circles, lines, etc. you created and added to a folium map
- Explain why you added those objects
  - Marking all launch sites on a map: Each launch site's latitude and longitude coordinates can be used to place a Circle Marker around it with the name of the launch site labeled on it.
  - Marking the success/failed launches for each site on the map: In a MarkerCluster, the dataframe launch outcomes (failures, successes) can be utilized to classify 0 and 1 with Green and Red markers on the map ()
  - Calculating the distances between a launch site to its proximities: The distance between various landmakers and the Launch Site can be determined using Haversine's formula. This is handy for locating various items and measuring distances around the Launch Site. To estimate distance to lanmarks, lines are drawn on the map.

Link to the notebook

Build a Dashboard with Plotly Dash

- A pie chart shows the total launches by a specific site or all sites, display relative proportions of various class of data. The size of the circle can be made proportional to the total quantity of that represented.

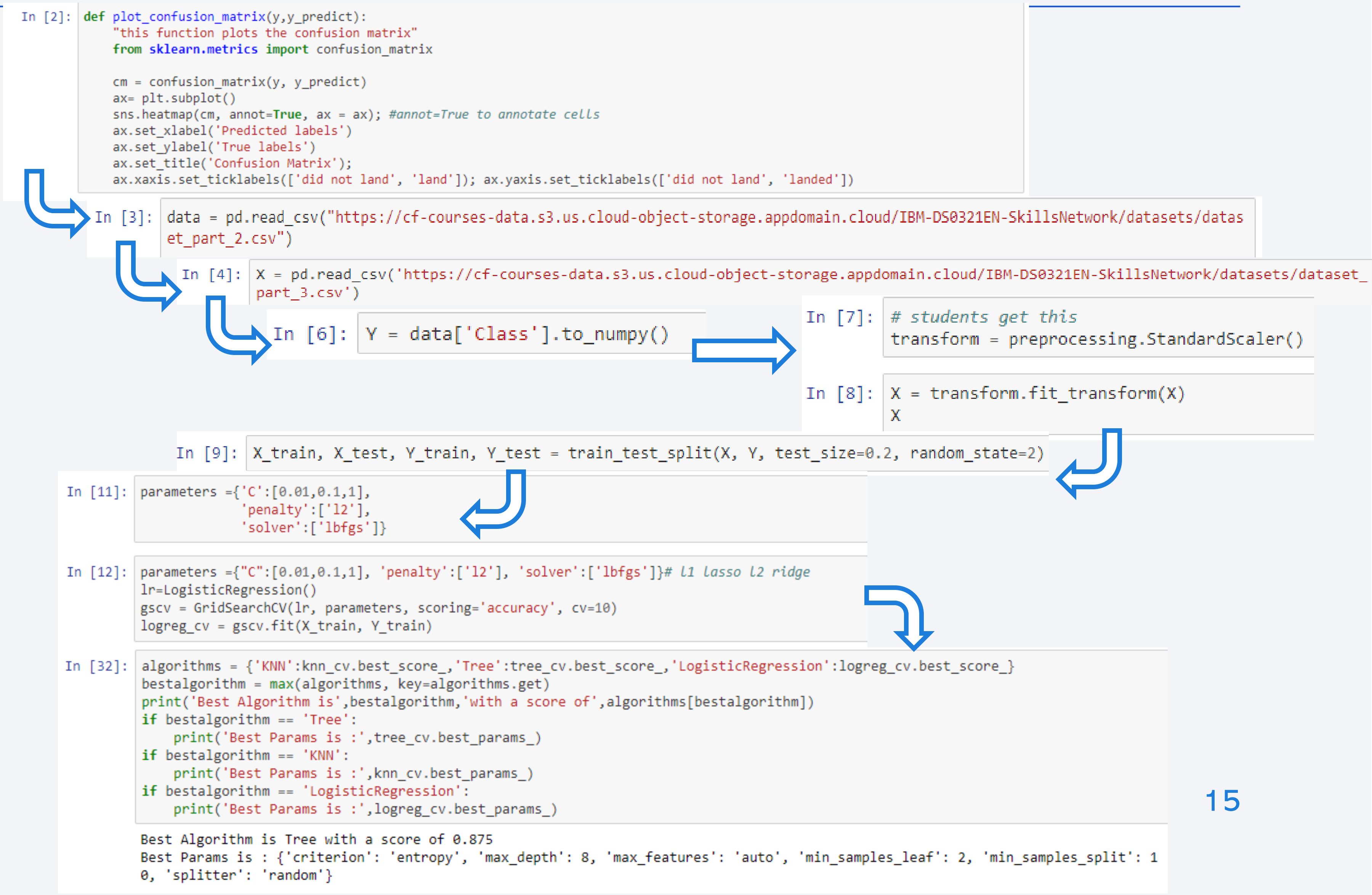
- The scatter graph depicts the association between payload mass (kg) and outcome for several booster models. It is the most effective approach for displaying a non-linear pattern since the range of data flow, observation, and reading are simple.

Link to the notebook

Predictive Analysis (Classification)

[Link to the notebook]

- Building model: Numpy and Pandas are used to load the dataset. Transform the data, divide it into training and test data sets, count the number of test samples, choose the machine learning methods to use, set the parameters and algorithms in GridsearchCV, fit the datasets into GridsearchCV objects, and train the dataset.
- Evaluating model: Checking each model's accuracy, tuning hyperparameters for each type of method, and plotting the confusion matrix
- Improving model: Feature engineering and algorithm tuning
- Finding the best performing classification model: The best performing model is the one with the highest accuracy score. A vocabulary of algorithms is included in the notebook, along with scores at the bottom.



```
In [2]: def plot_confusion_matrix(y,y_predict):
            "this function plots the confusion matrix"
            from sklearn.metrics import confusion_matrix

            cm = confusion_matrix(y, y_predict)
            ax= plt.subplot()
            sns.heatmap(cm, annot=True, ax = ax); #annot=True to annotate cells
            ax.set_xlabel('Predicted labels')
            ax.set_ylabel('True labels')
            ax.set_title('Confusion Matrix');
            ax.xaxis.set_ticklabels(['did not land', 'land']); ax.yaxis.set_ticklabels(['did not land', 'landed'])
```

```
In [3]: data = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/datas
        et_part_2.csv")
```

```
In [4]: X = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_
        part_3.csv')
```

```
In [6]: Y = data['Class'].to_numpy()
```

```
In [7]: # students get this
        transform = preprocessing.StandardScaler()
```

```
In [8]: X = transform.fit_transform(X)
        X
```

```
In [9]: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
```

```
In [11]: parameters ={'C':[0.01,0.1,1],
                      'penalty':['l2'],
                      'solver':['lbfgs']}
```

```
In [12]: parameters ={"C":[0.01,0.1,1], 'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
         lr=LogisticRegression()
         gscv = GridSearchCV(lr, parameters, scoring='accuracy', cv=10)
         logreg_cv = gscv.fit(X_train, Y_train)
```

```
In [32]: algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
         bestalgorithm = max(algorithms, key=algorithms.get)
         print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
         if bestalgorithm == 'Tree':
             print('Best Params is :',tree_cv.best_params_)
         if bestalgorithm == 'KNN':
             print('Best Params is :',knn_cv.best_params_)
         if bestalgorithm == 'LogisticRegression':
             print('Best Params is :',logreg_cv.best_params_)

         Best Algorithm is Tree with a score of 0.875
         Best Params is : {'criterion': 'entropy', 'max_depth': 8, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 1
         0, 'splitter': 'random'}
```

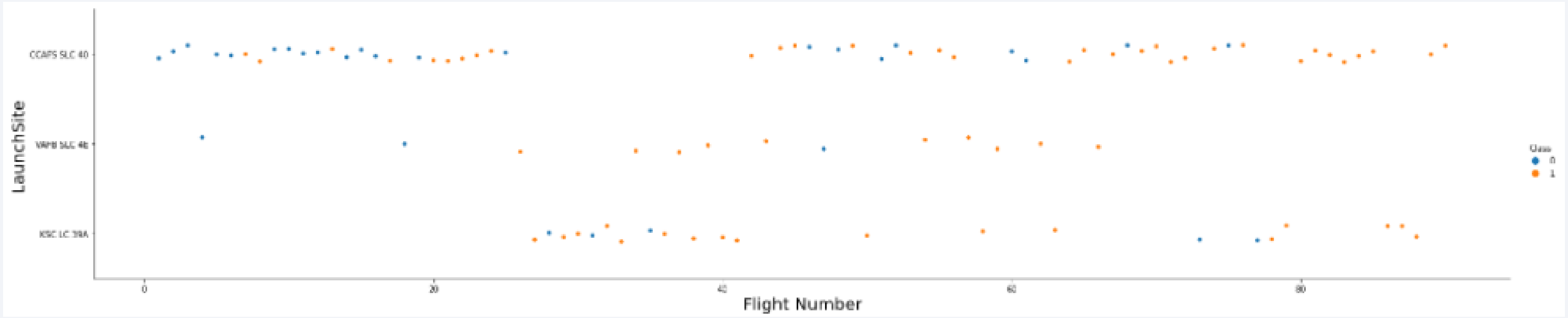
Results

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

Section 2

Insights drawn from EDA

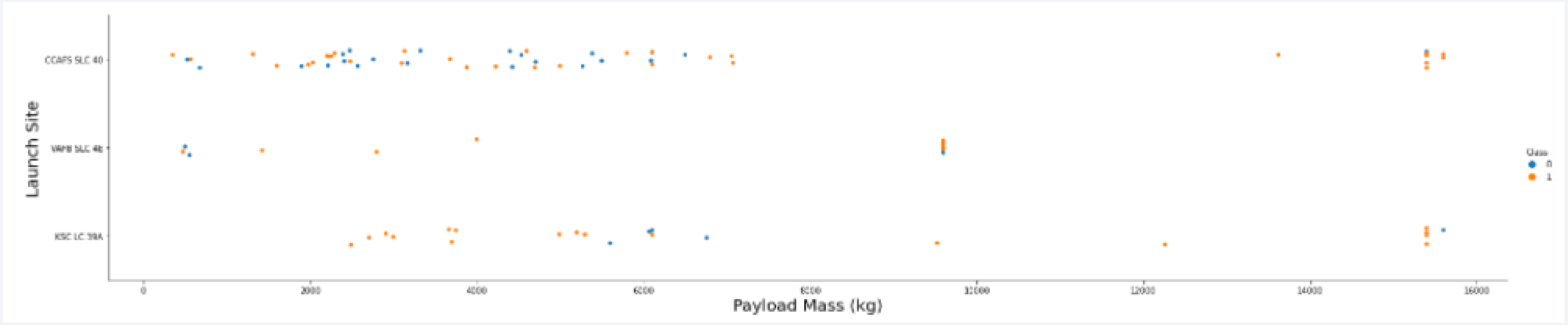
Flight Number vs. Launch Site



Explanations:

1. The earliest flights all failed while the latest flights all succeeded.
2. The CCAFS SLC 40 launch site has about a half of all launches.
3. VAFB SLC 4E and KSC LC 39A have higher success rates.
4. It can be assumed that each new launch has a higher rate of success .

Payload vs. Launch Site



Explanation:

For every launch site the higher the payload mass, the higher the success rate.

Most of the launches with payload mass over 7000 kg were successfull.

KSC LC 39A has a 100% success rate for payload mass under 5500 kg too.

In Payload Vs. Launch Site scatter point chart, for the VAFB-SLC launch site there are no rockets launched for heavypayload mass(greater than 10000).

Success Rate vs. Orbit Type

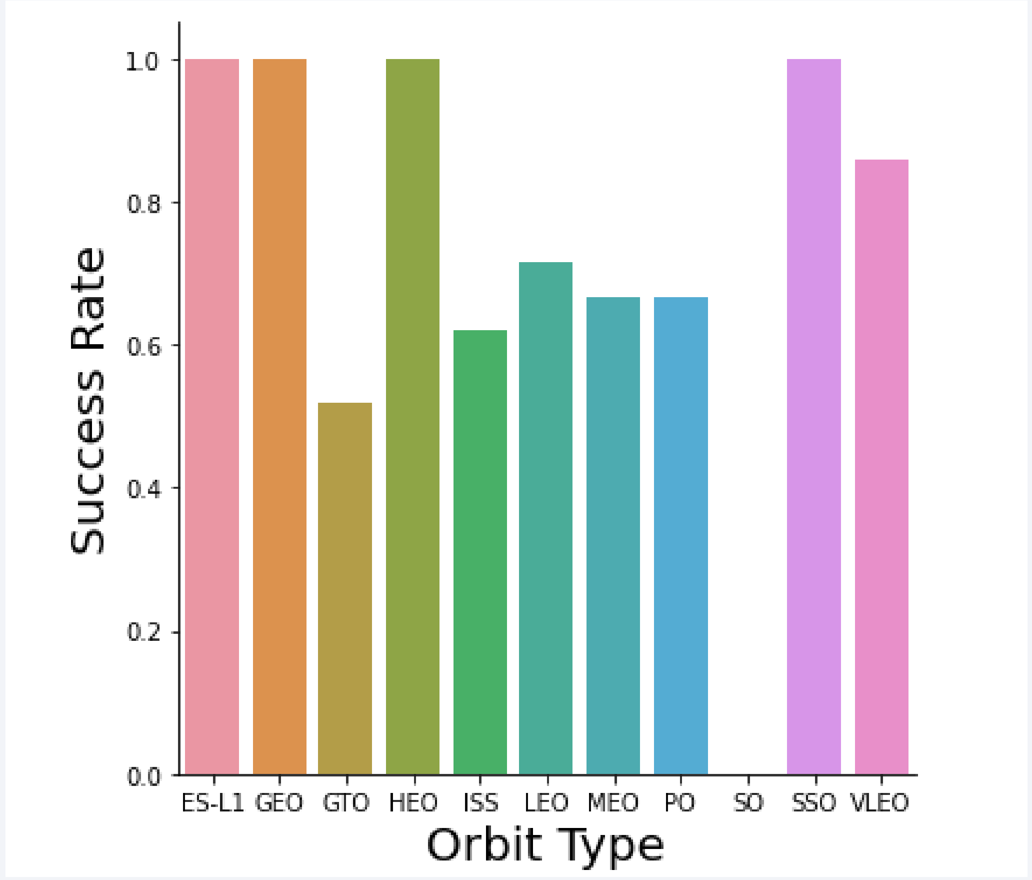
Explanation:

Orbits with 100% success rate are:

ES-L1 GEO HEO SSO

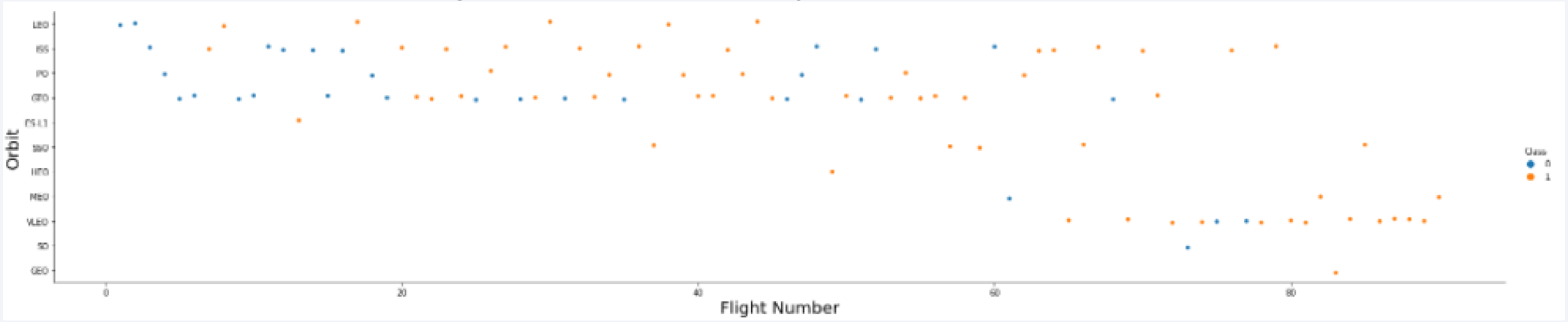
Orbits with 0% success rate are:

SO Orbits with success rate between 50% and 85%: GTO ISS LEO MEO PO



Flight Number vs. Orbit Type

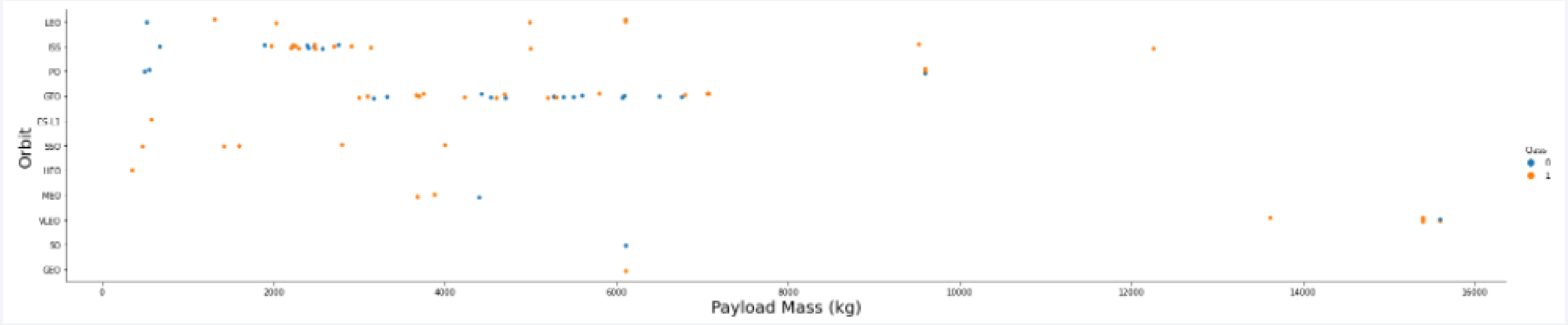
- Show a scatter point of Flight number vs. Orbit type



- Show the screenshot of the scatter plot with explanations

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

Payload vs. Orbit Type



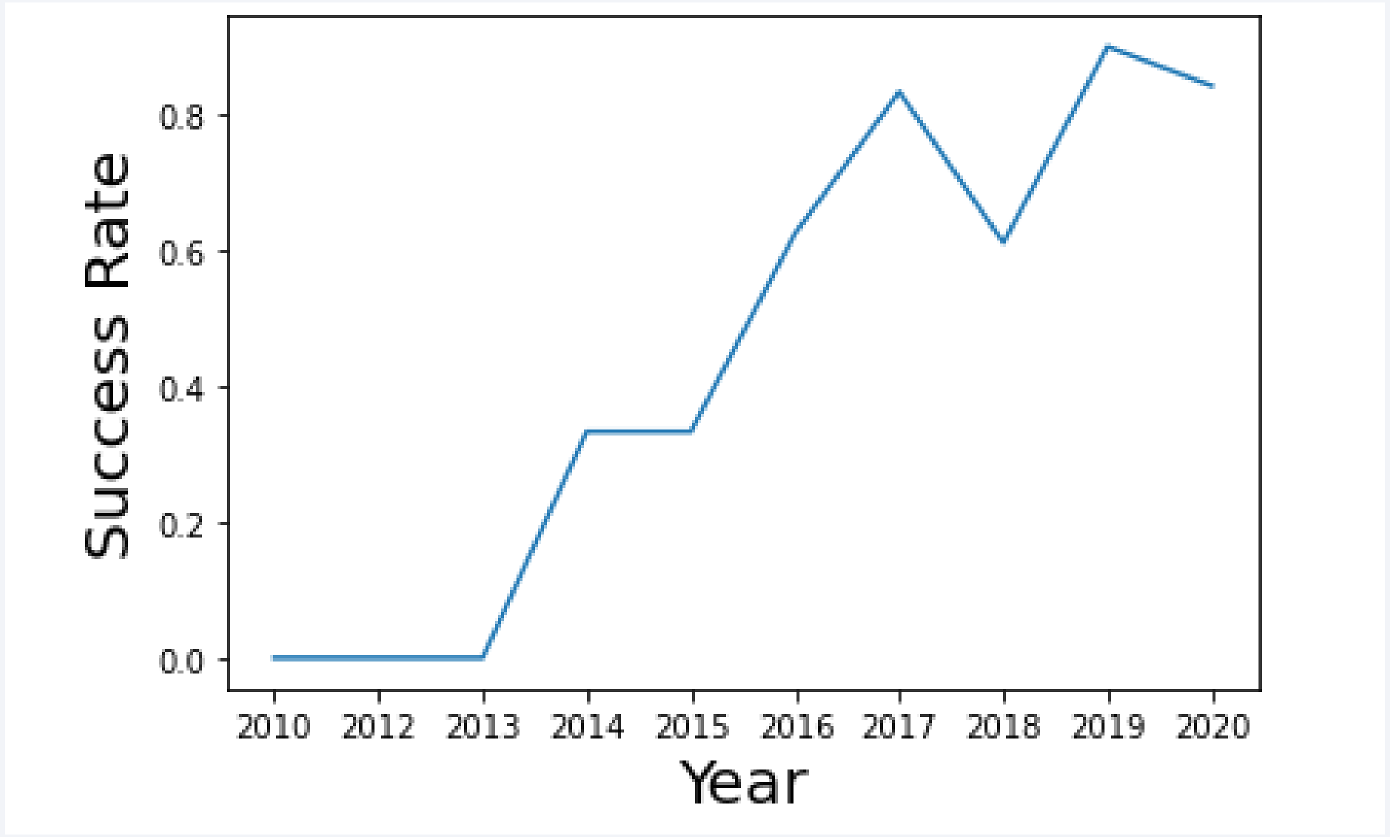
Explanations:

1. With heavy payloads, the successful landing or positive landing rate are more for Polar, LEO and ISS.
2. However, for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.

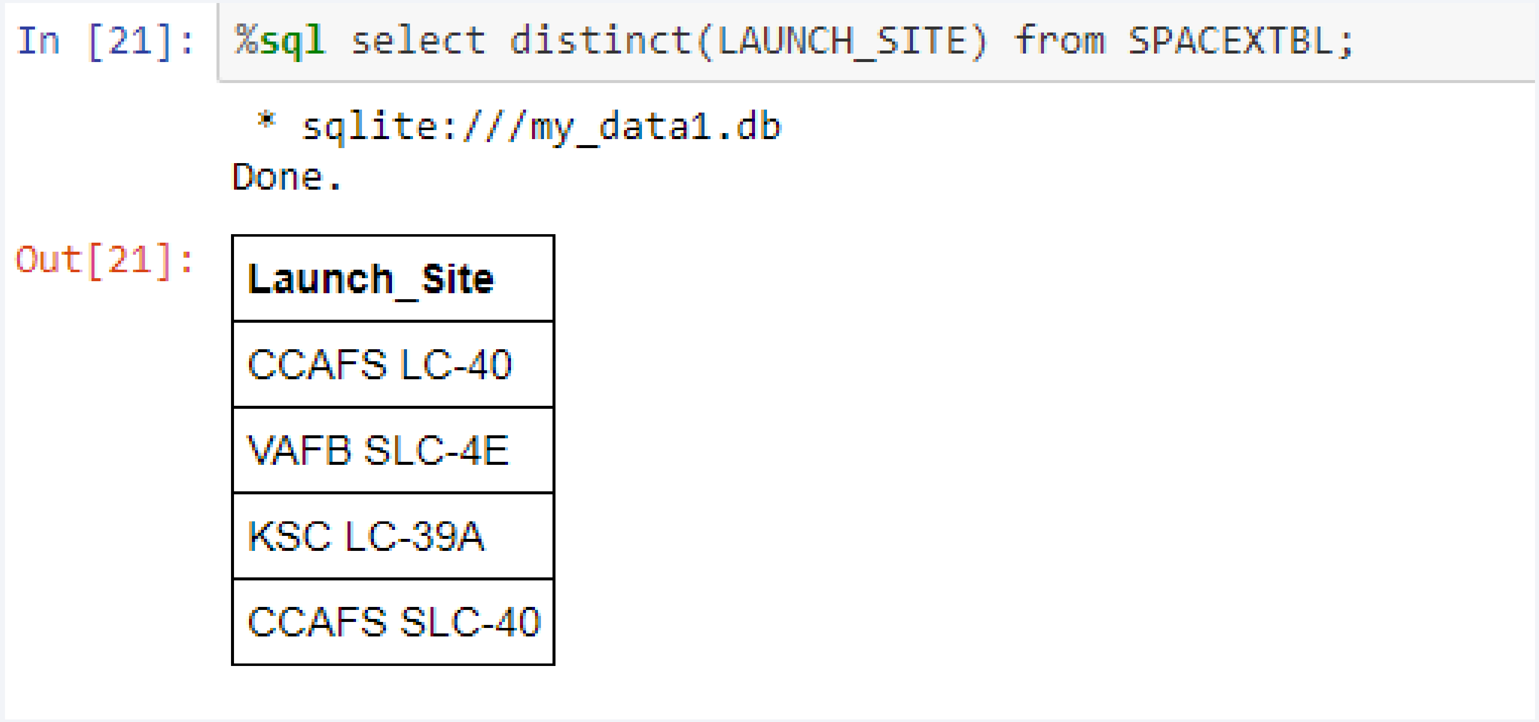
Launch Success Yearly Trend

Explanations:

The sucess rate since 2013 kept increasing untill 2020



All Launch Site Names

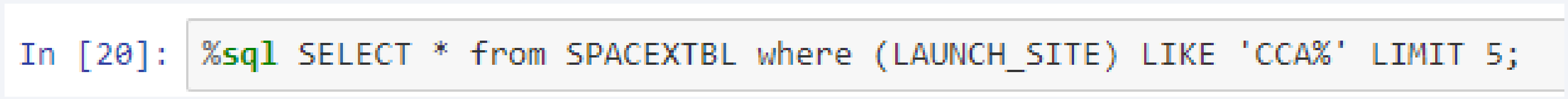


```
In [21]: %sql select distinct(LAUNCH_SITE) from SPACEXTBL;

 * sqlite:///my_data1.db
Done.

Out[21]:
```

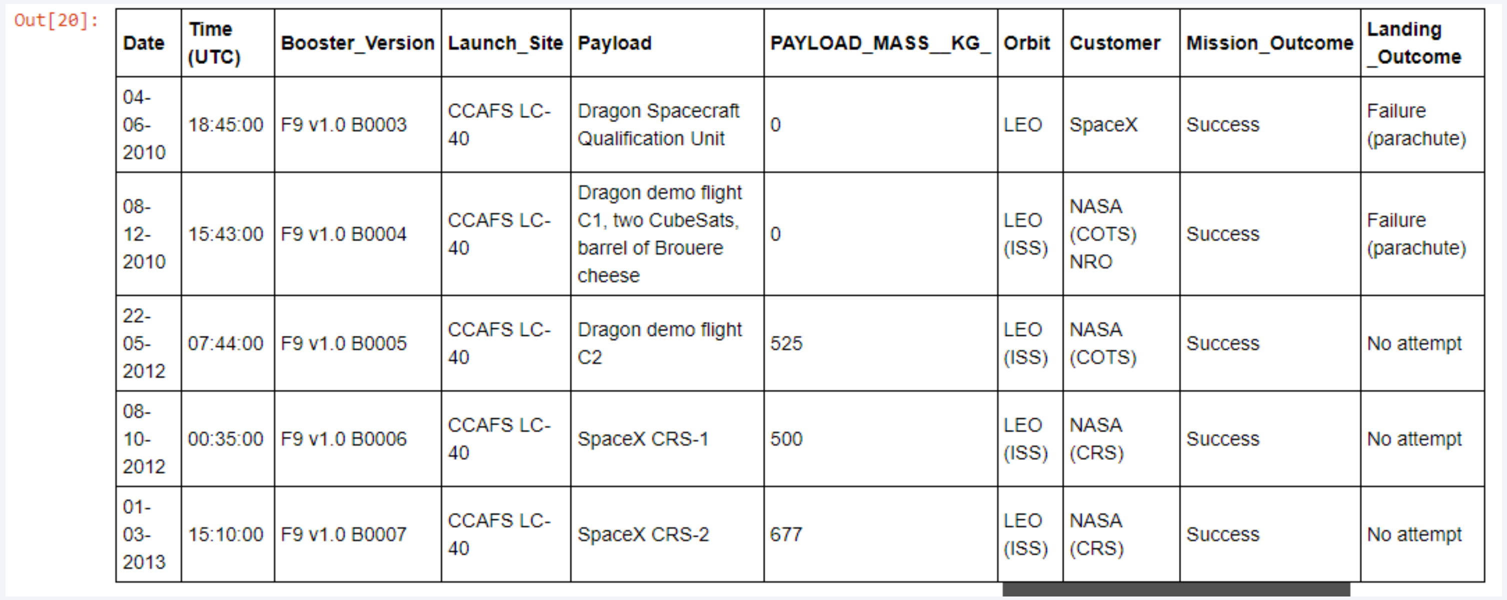
Launch_Site
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40

The word "DISTINCT" in the query means that it will only return unique result from the LAUNCH_SITE column of SPACEXDATASET

Launch Site Names Begin with 'CCA'



```
In [20]: %sql SELECT * from SPACEXTBL where (LAUNCH_SITE) LIKE 'CCA%' LIMIT 5;
```

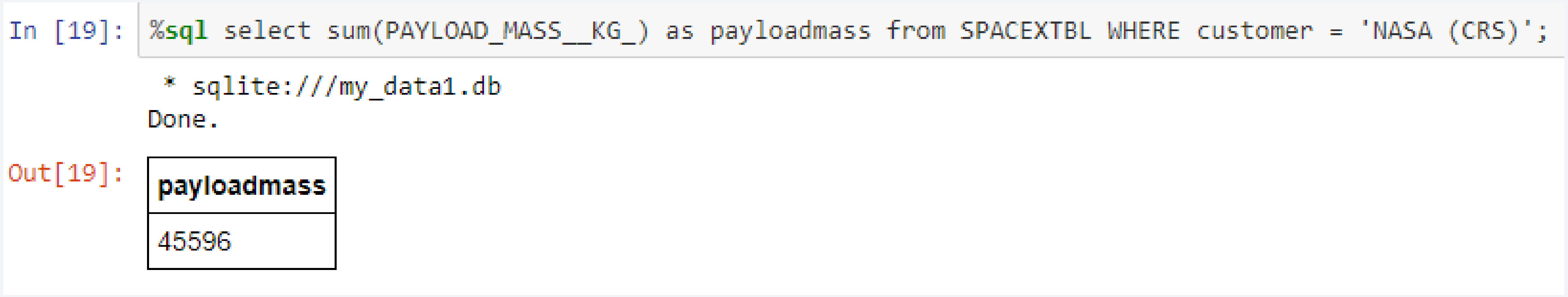


Out[20]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	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)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

The word "LIMIT 5" in the query means that it will only return 5 records from SPACEXDATASET. The words "where" and "LIKE" are to select only data that contains word "CCA". The percentage in the end of CCA indicates that the launch site name must start with CCA

Total Payload Mass


```
In [19]: %sql select sum(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL WHERE customer = 'NASA (CRS)';
```

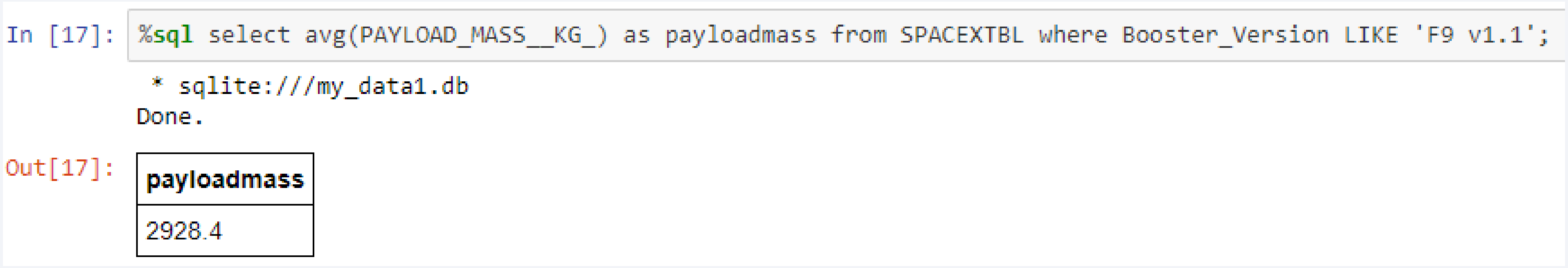
```
 * sqlite:///my_data1.db
Done.
```

Out[19]:

payloadmass
45596

The SUM function sums all of the value in the payload mass kg_ column. The word "where" restricts calculations to NASA customers only (CRS)

Average Payload Mass by F9 v1.1

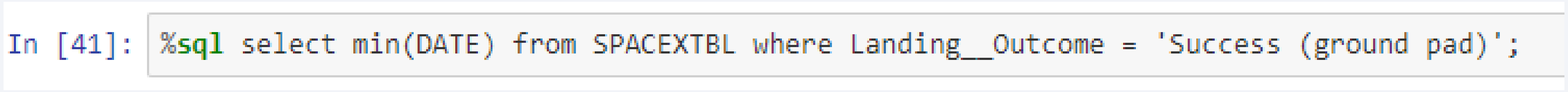


```
In [17]: %sql select avg(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL where Booster_Version LIKE 'F9 v1.1';

 * sqlite:///my_data1.db
Done.

Out[17]:
```

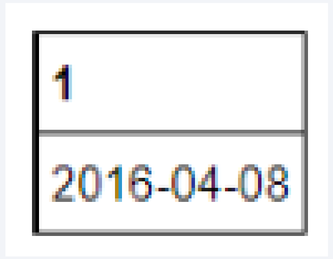
payloadmass
2928.4

The AVG function averages all of the values in the PAYLOAD MASS KG column. The words "where" and "LIKE" limit the dataset so that only Booster Version ='F9 v1.1' is calculated.

First Successful Ground Landing Date



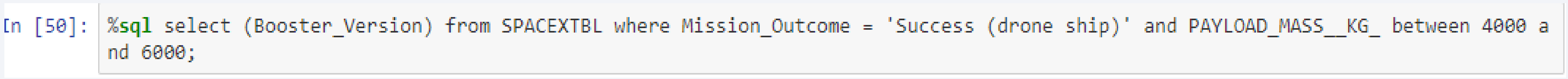
```
In [41]: %sql select min(DATE) from SPACEXTBL where Landing__Outcome = 'Success (ground pad)';
```



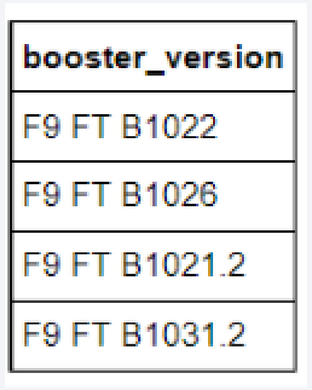
1
2016-04-08

The function MIN select the minimum data in the column DATE. The word "where" limits to only executes calculation on landing outcome = Success (ground pad).

Successful Drone Ship Landing with Payload between 4000 and 6000



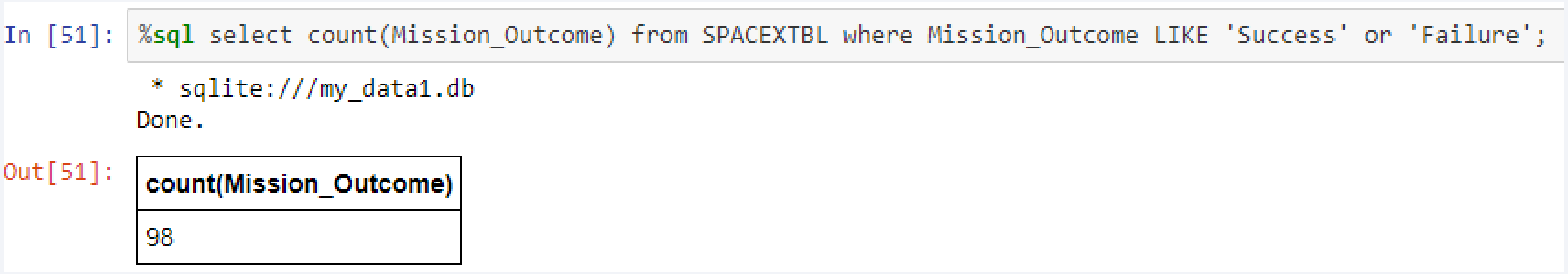
```
In [50]: %sql select (Booster_Version) from SPACEXTBL where Mission_Outcome = 'Success (drone ship)' and PAYLOAD_MASS__KG_ between 4000 a
nd 6000;
```



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

First, selecting only Booster_Version column. Second, the word "where " filters the dateset so that Landing__Outcome = "Success (drone ship)" is returned. Third, the word "and" specifies additional filter conditions PAYLOAD_MASS__KG_ ranges from 4000 to 6000.

Total Number of Successful and Failure Mission Outcomes

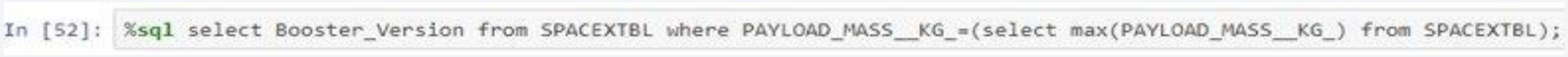


```
In [51]: %sql select count(Mission_Outcome) from SPACEXTBL where Mission_Outcome LIKE 'Success' or 'Failure';

 * sqlite:///my_data1.db
Done.

Out[51]:
```

count(Mission_Outcome)
98

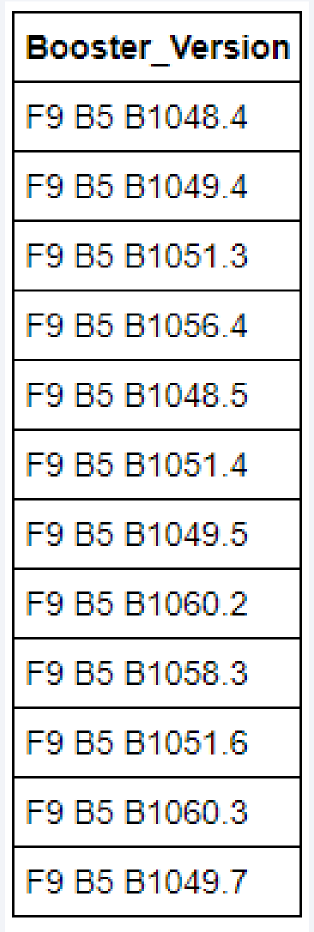
The word "count" is to counting all data in the Mission_Outcome column from dataset. The phrases "where" and "LIKE" restrict the calculation to counting only data that containing words "Success" or "Failure"

Boosters Carried Maximum Payload



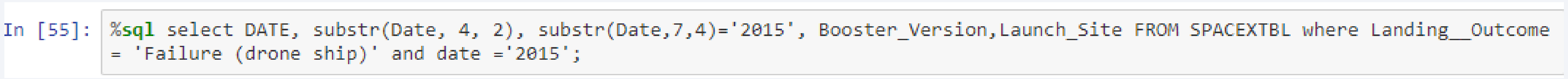
```
In [52]: %sql select Booster_Version from SPACEXTBL where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from SPACEXTBL);
```

First, select the Booster_Version column from dataset. Second, the word "where" restricts only boosters that contain maximum payload mass

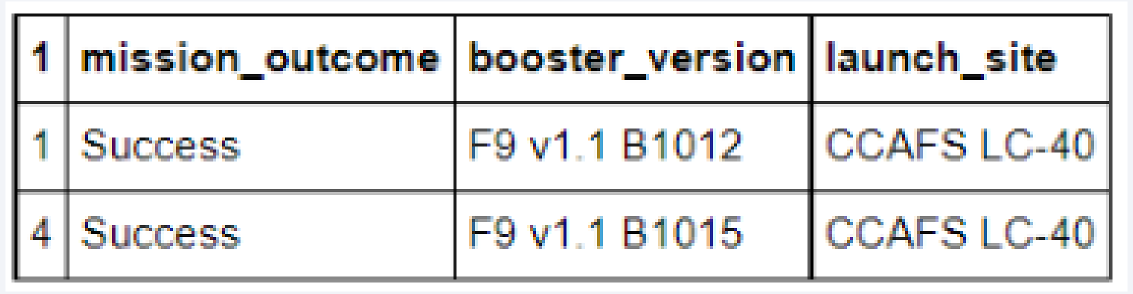


Booster_Version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

2015 Launch Records



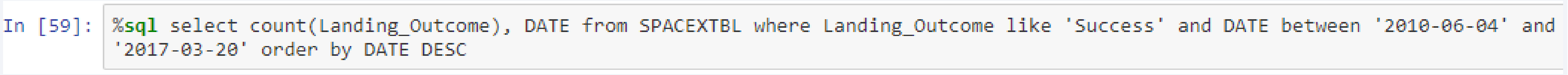
```
In [55]: %sql select DATE, substr(Date, 4, 2), substr(Date,7,4)='2015', Booster_Version,Launch_Site FROM SPACEXTBL where Landing__Outcome
         = 'Failure (drone ship)' and date ='2015';
```



1	mission_outcome	booster_version	launch_site
1	Success	F9 v1.1 B1012	CCAFS LC-40
4	Success	F9 v1.1 B1015	CCAFS LC-40

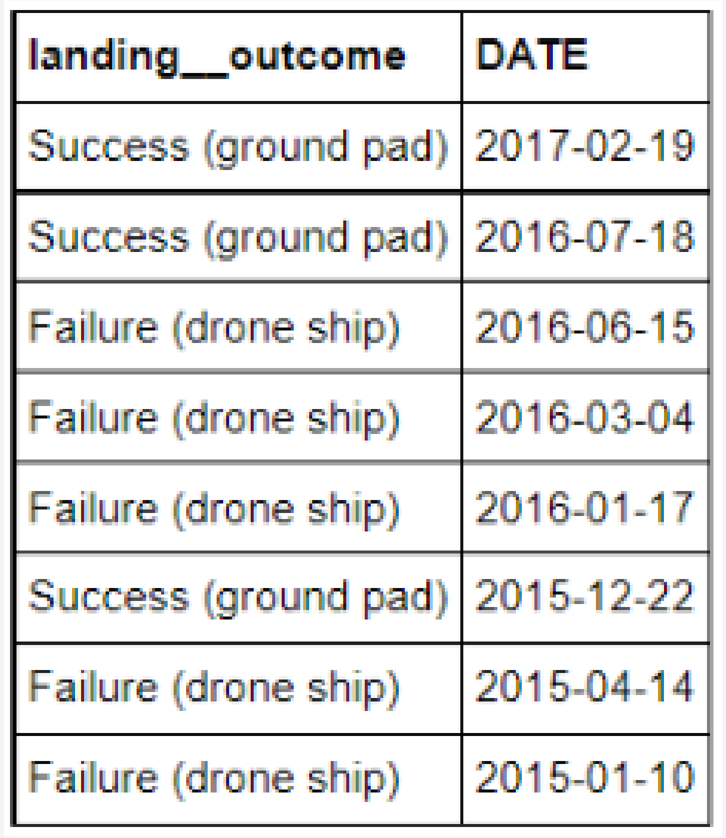
The query is to selecting date, month, year, Booster version and launch site from the dataset. Because SQLLite does not support monthnames, the word phrases "substr(Date, 4, 2)" is needed as month to get the months and phrases "substr(Date,7,4)='2015' " is for the year. The phrases "word", "and" filter to select only drone ship landing outcome failure in the year of 2015.

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



```
In [59]: %sql select count(Landing_Outcome), DATE from SPACEXTBL where Landing_Outcome like 'Success' and DATE between '2010-06-04' and
         '2017-03-20' order by DATE DESC
```

The query is to counting data in landing outcome column and selecting date from data set. The words " where, like" are to filter only success landing outcome. The word "and" specifies the date that is between 2010-06-04 and 2017-03-20. Finally, all data is sorted in descending order by the query " order by DATE DESC".

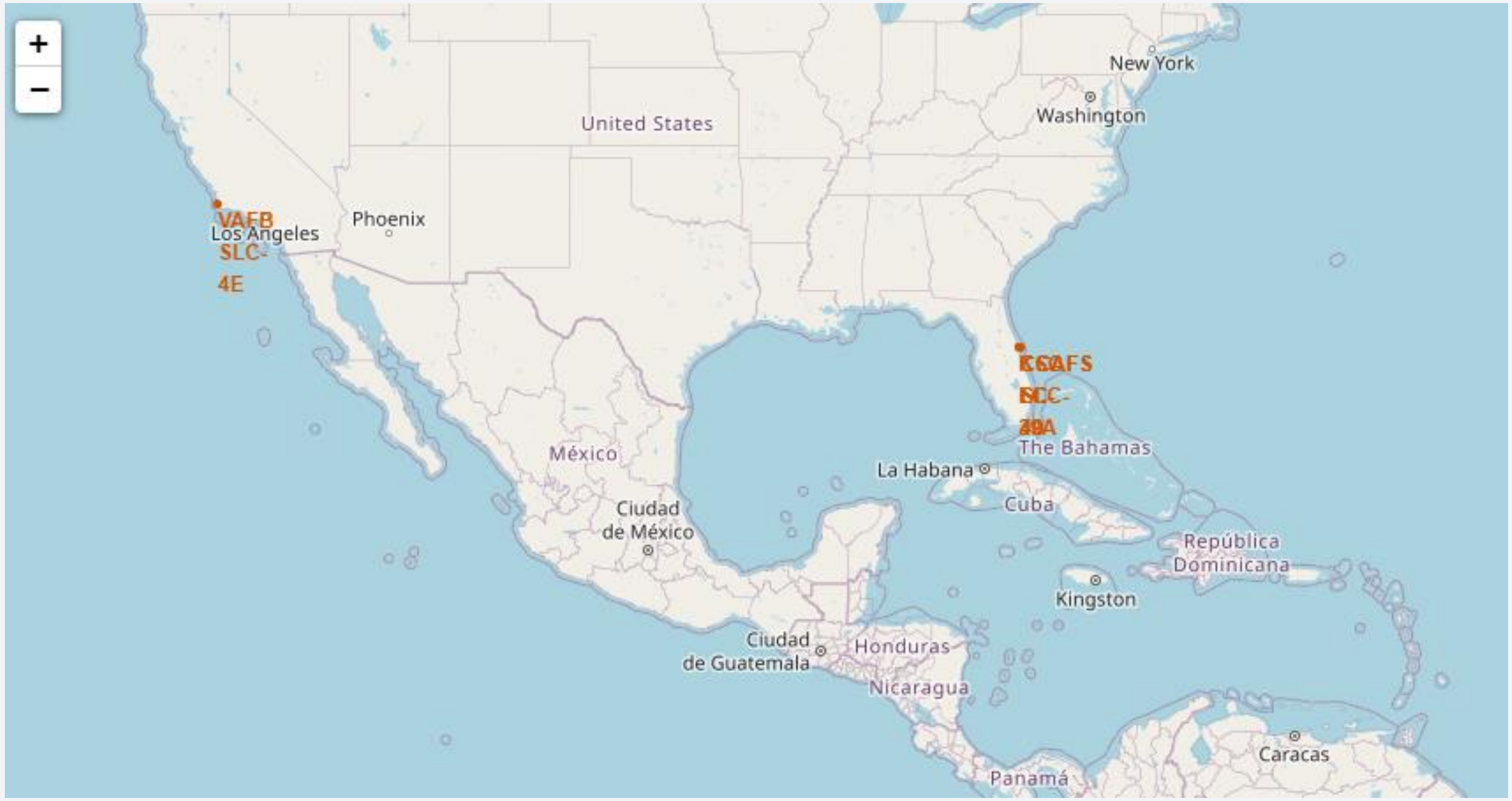


landing__outcome	DATE
Success (ground pad)	2017-02-19
Success (ground pad)	2016-07-18
Failure (drone ship)	2016-06-15
Failure (drone ship)	2016-03-04
Failure (drone ship)	2016-01-17
Success (ground pad)	2015-12-22
Failure (drone ship)	2015-04-14
Failure (drone ship)	2015-01-10

Section 4

Launch Sites Proximities Analysis

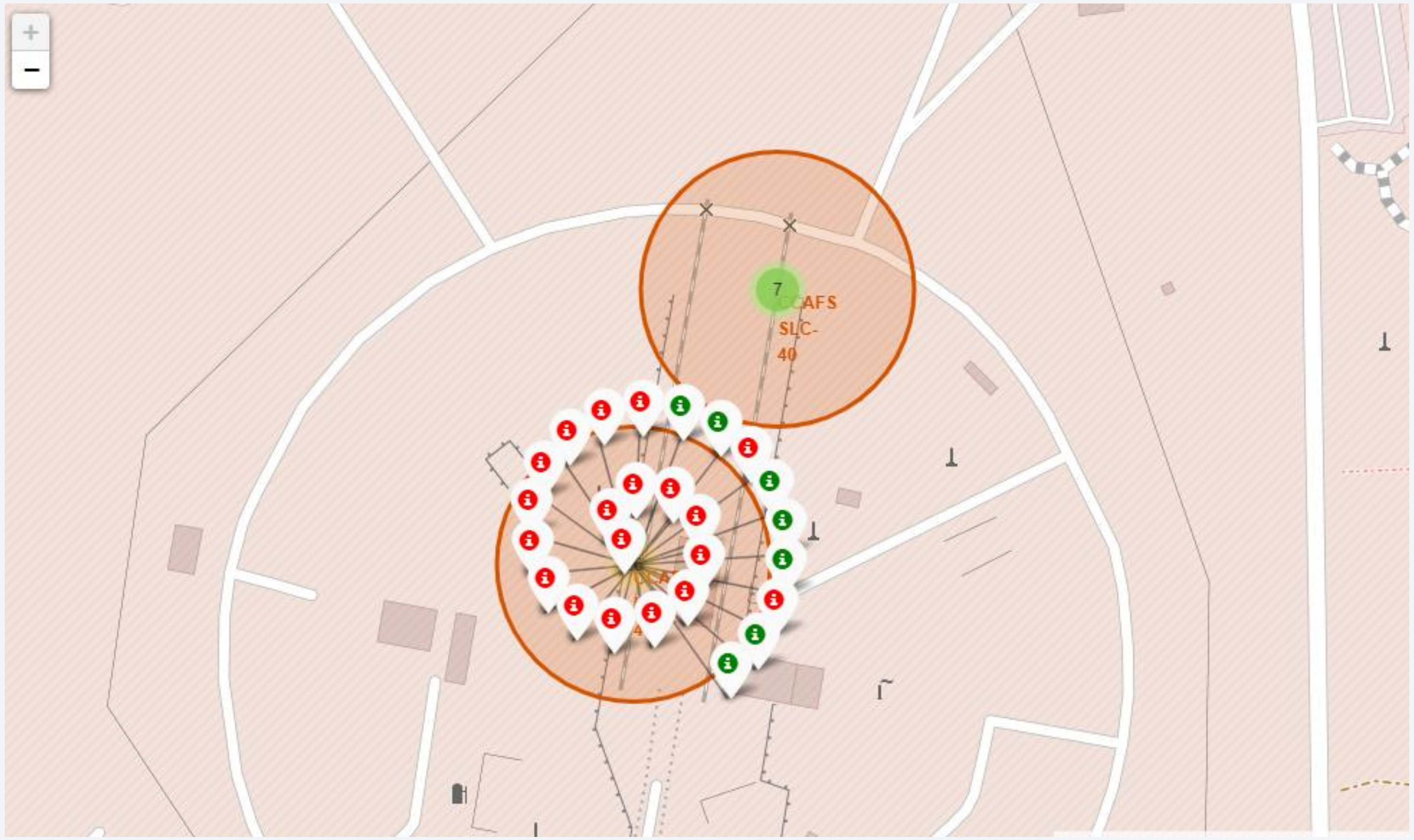
Mark all Launch Sites on a Map



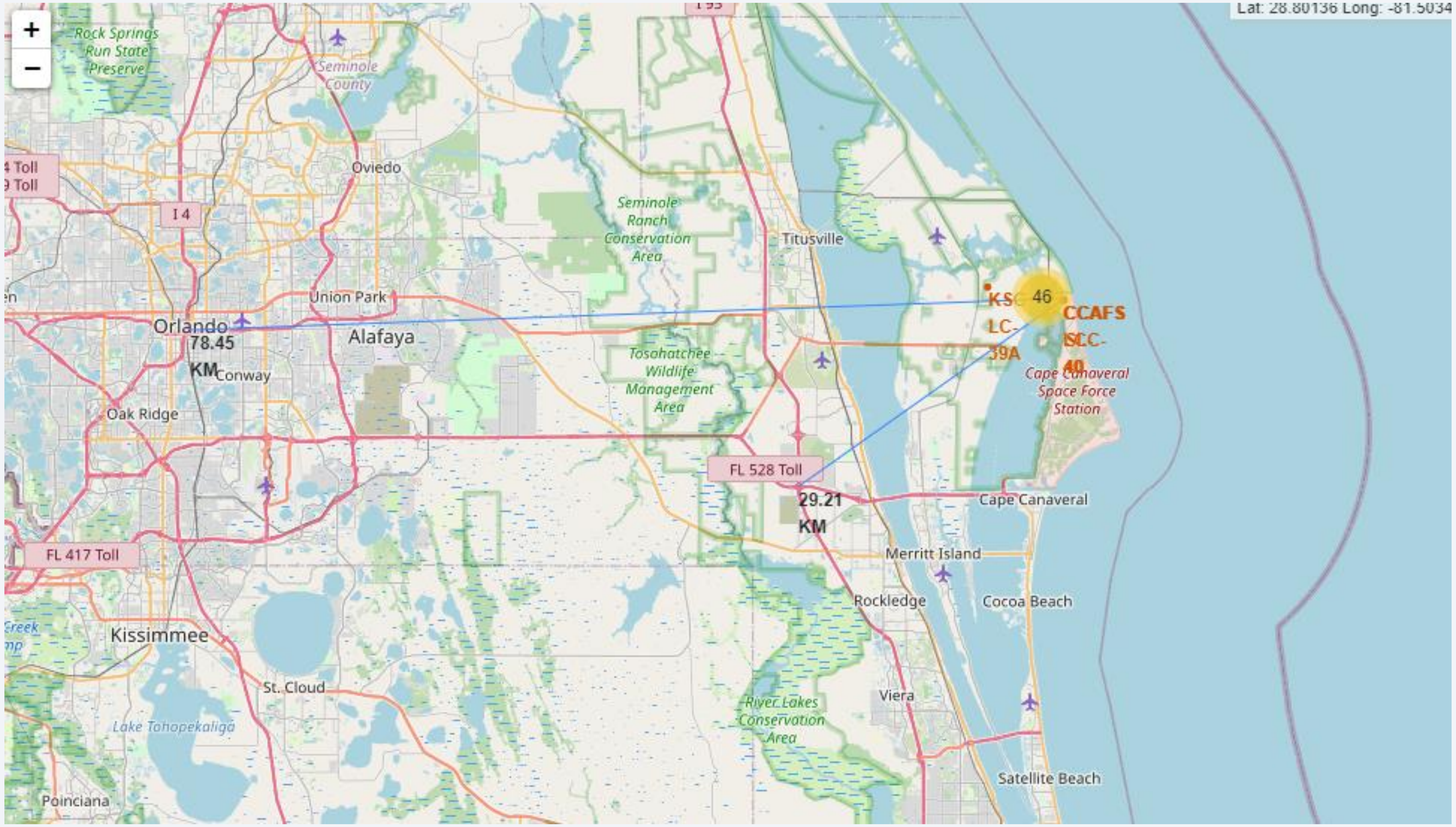
SpaceX launch sites are located in America coasts, Florida, and California.

Mark the Success or Faile Launches for Each Site on the Map

The green mark refers to successful launches, while the red mark refers to failed launches



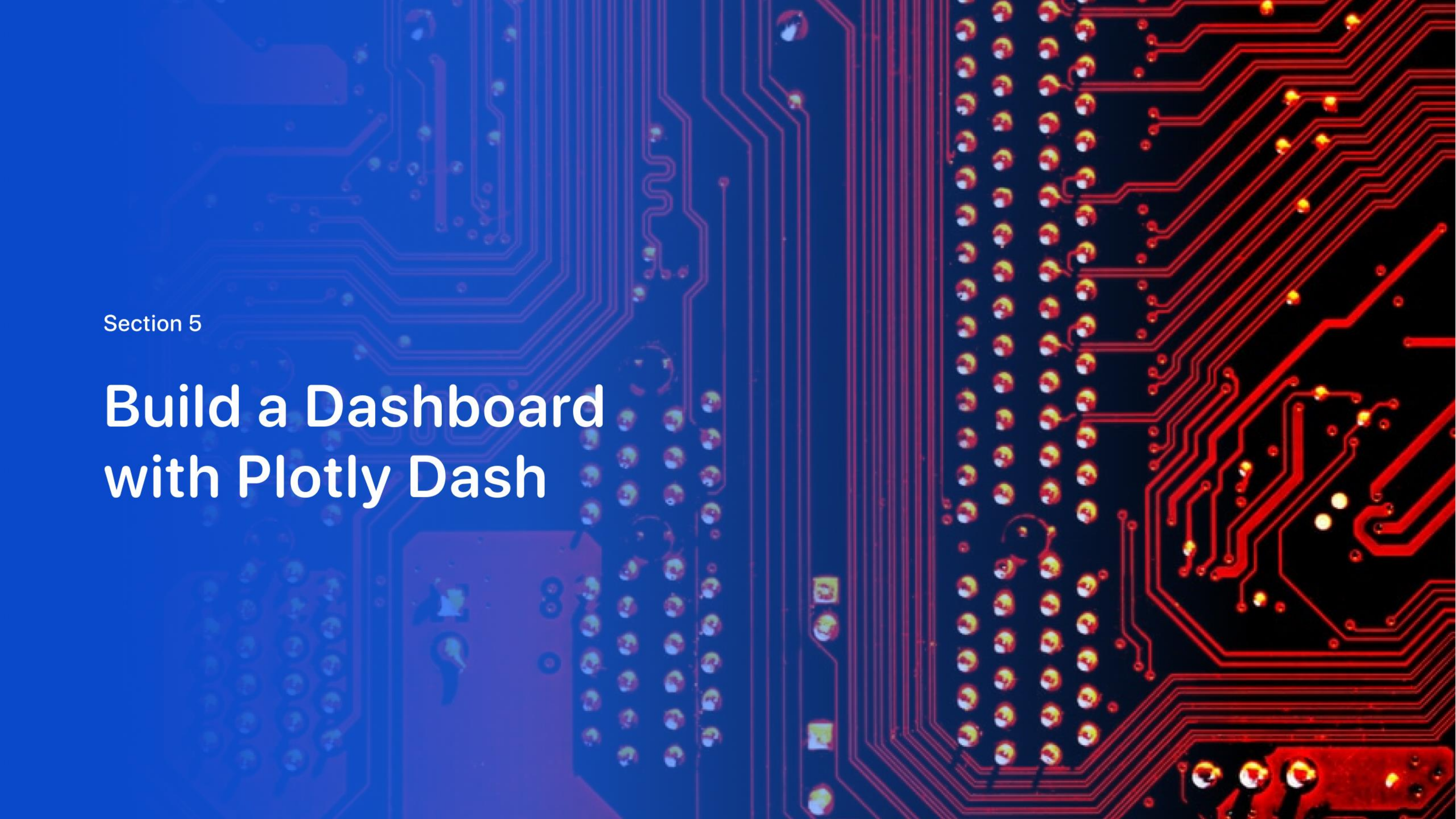
Calculate the Distances between a Launch Site to Its Proximities



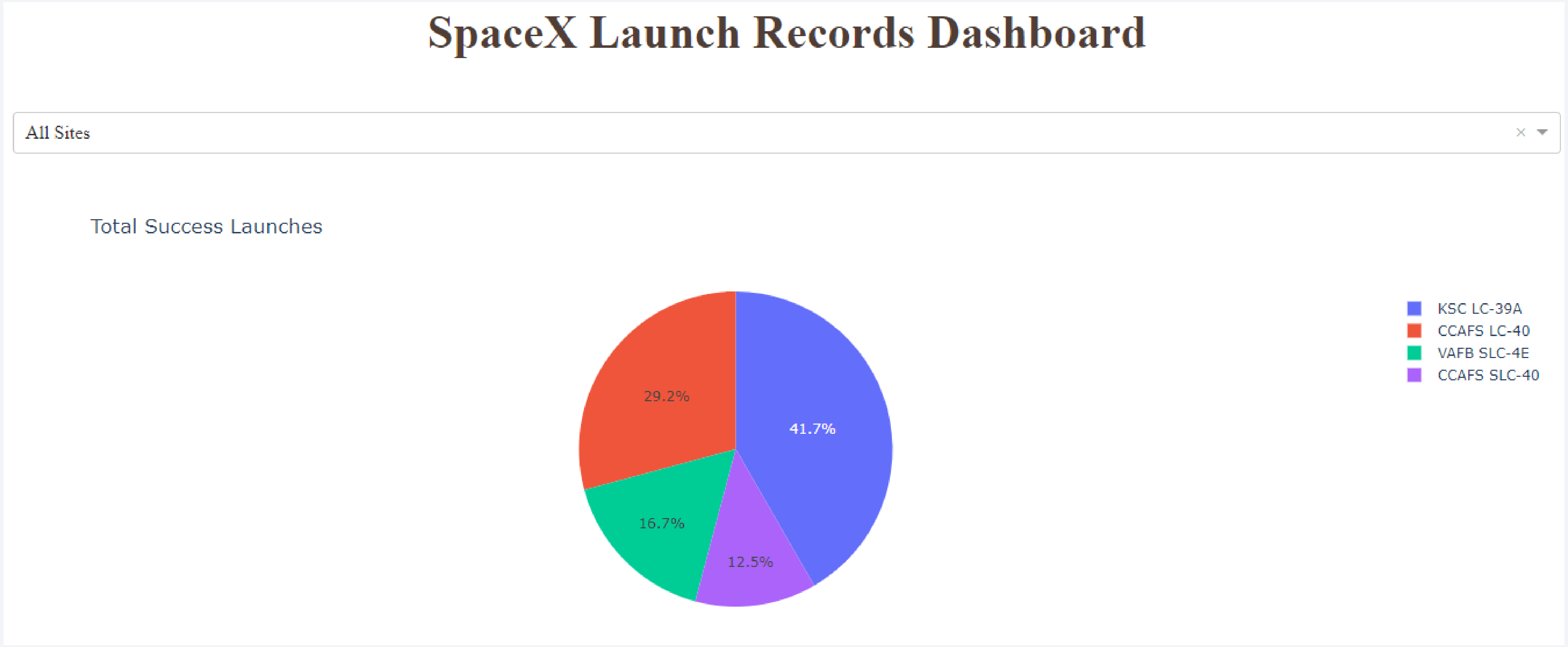
The distance from CCAFS-SLC 40 launch site to a highway is 29,21 km and 78,45 km to Florida (Orlando) City

Section 5

Build a Dashboard with Plotly Dash



Total Successful Launches By all sites

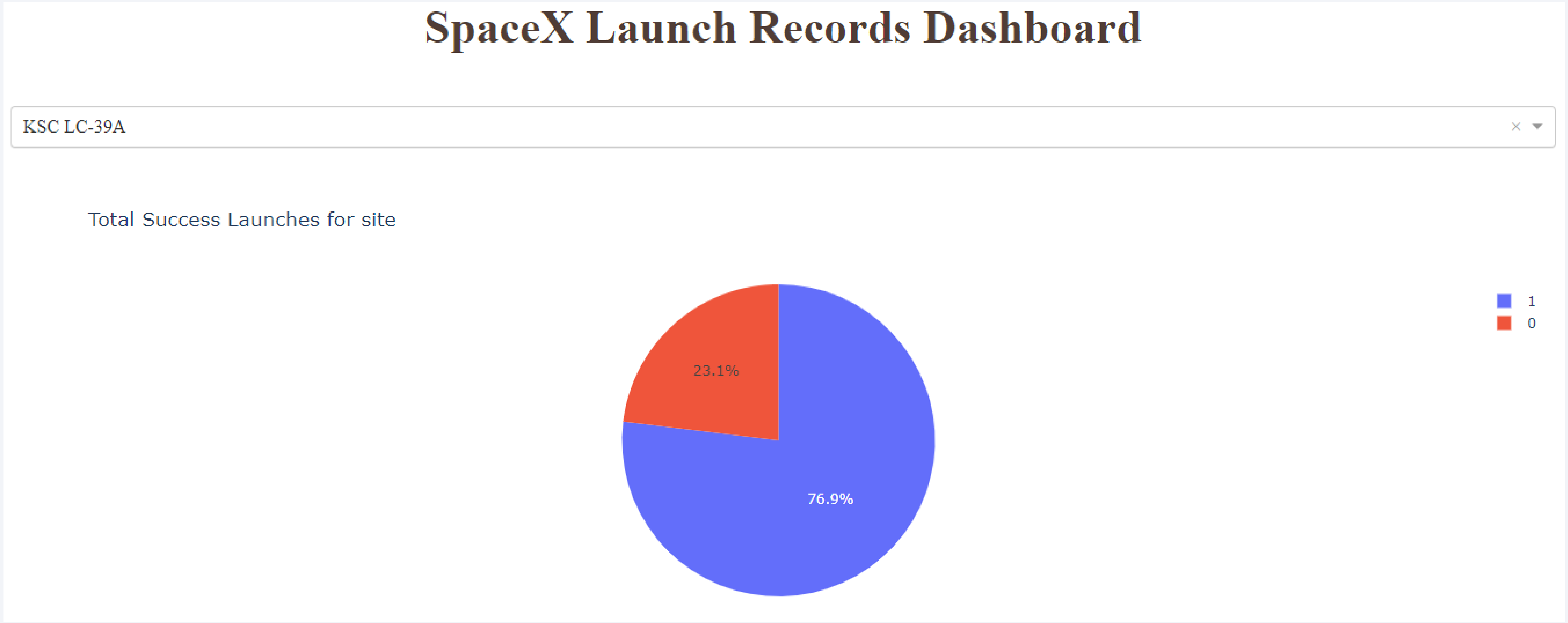

SpaceX Launch Records Dashboard

All Sites

Total Success Launches

KSC LC-39A launch site is the most successful launches.

Total Success Launches for site KSC LC-39A



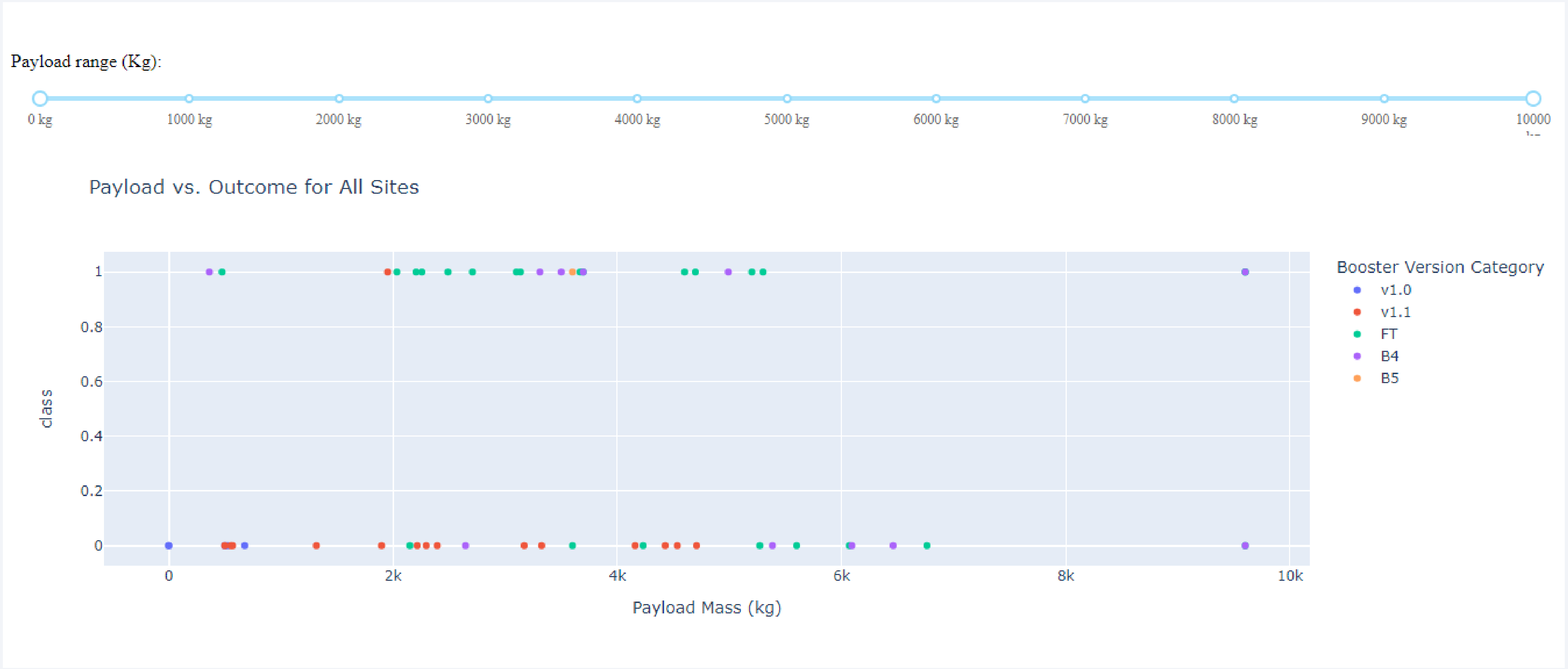
SpaceX Launch Records Dashboard

KSC LC-39A

Total Success Launches for site

The KSC LC-39A launch site achieved 76.9% success rate but has 23.1% failure rate

Payload VS. Mass with Different Payload

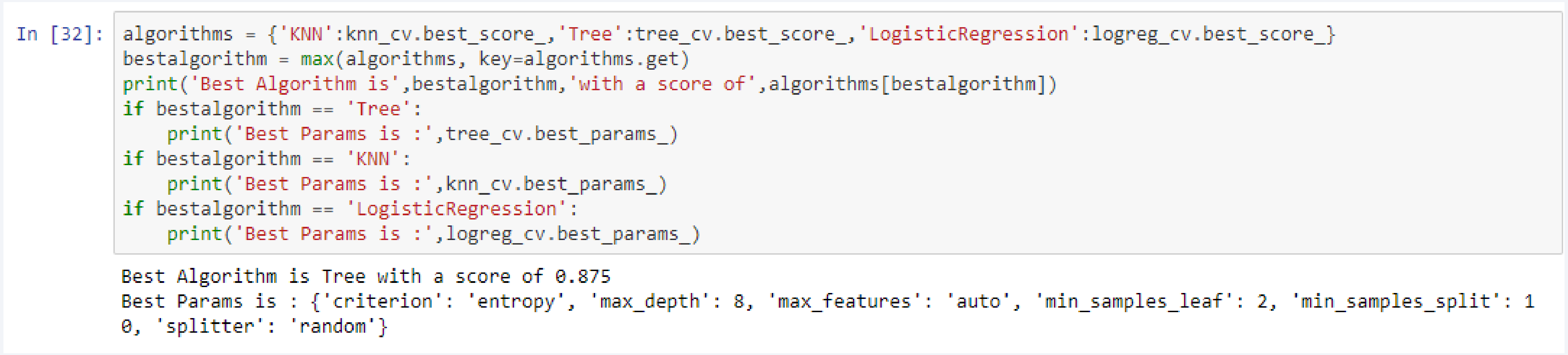


Launches with weighing less than 8000 kg were mostly successful.

Section 6

Predictive Analysis (Classification)

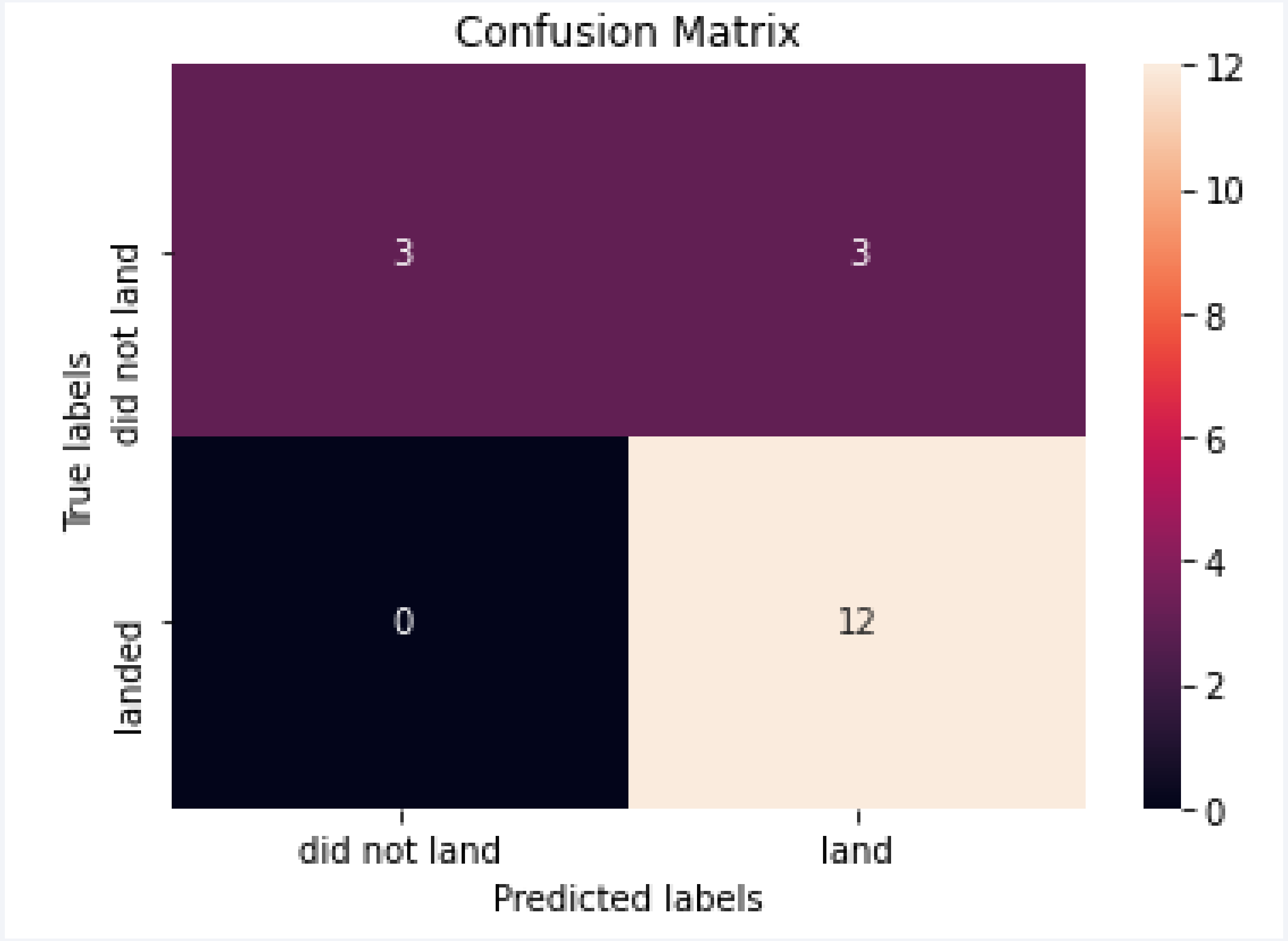
Classification Accuracy


```
In [32]: algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
         bestalgorithm = max(algorithms, key=algorithms.get)
         print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
         if bestalgorithm == 'Tree':
             print('Best Params is :',tree_cv.best_params_)
         if bestalgorithm == 'KNN':
             print('Best Params is :',knn_cv.best_params_)
         if bestalgorithm == 'LogisticRegression':
             print('Best Params is :',logreg_cv.best_params_)
```

```
Best Algorithm is Tree with a score of 0.875
Best Params is : {'criterion': 'entropy', 'max_depth': 8, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 1
0, 'splitter': 'random'}
```

The best model is Tree because has 0.875 accuracy score

Confusion Matrix



The Tree model can distinguish between the different classes. In the confusion matrix, the major problem is false positives.

Conclusions

- For this SpaceX Dataset, the Tree Classifier Algorithm is the best model for Machine Learning.
- Payloads with a low weight perform better success rate than payloads with a higher weight.
- The bigger the payload mass, the higher the success rate for each launch site. Launches weighing less than 8000 kg were mostly successful.
- Orbit ES-L1, GEO, HEO, SSO has the best Success Rate
- Over time, the success rate of SpaceX launches has increased from 2013 to 2020. This demonstrates that SpaceX is a high-performing corporation that is always improving its products.
- With a success percentage of 76.9%, the KSC LC-39A launch site is the most successful of all the sites.

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!