IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Rudy Schneeweiss
30/03/2022

Outline

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

NB. I would like to apologize for any formatting errors, this was created using LibreOffice software and saved as a PPT

Executive Summary

- **Methodologies used:**
- Data collection
- Data Wrangling
- EDA with visualization
- EDA with SQL
- Building an interactive map with Folium
- Building a dashboard with Plotly Dash
- Predictive Analysis & Classification

- **Summary of Results:**
- EDA Results
- Interactive Analytics
- Predictive Analysiis

Introduction

The commercial space age is here, companies are making space travel affordable for everyone. Virgin Galactic is providing suborbital spaceflights. Rocket Lab is a small satellite provider. Blue Origin manufactures sub-orbital and orbital reusable rockets. Perhaps the most successful is SpaceX.

SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upwards 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.

Here, we set out to find:
- The cost of a launch
- The probability that the first stage will return

Section 1

Methodology

Methodology

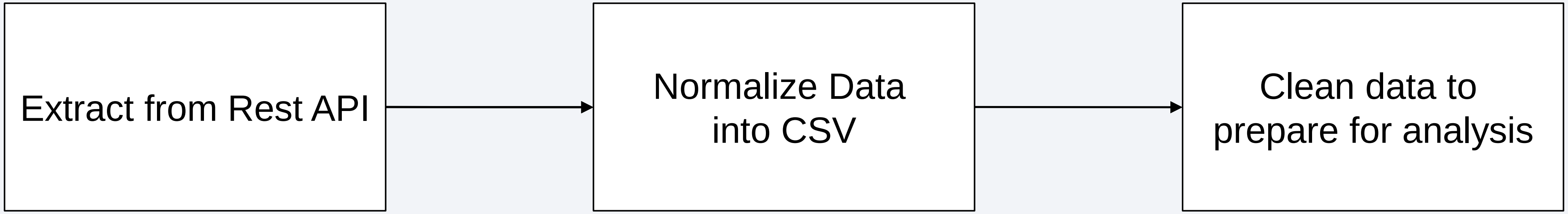
Executive Summary

- Data collection methodology:
  - Pulled from SpaceX Rest API & scraped from Wikipedia
- Perform data wrangling
  - Used One Hot Encoding fields. Cleaned null values & irrelevant columns from initial data.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Different models were evaluated to find the best classifier. (LR, KNN, SVM, DT)

Data Collection

Method 1: SpaceX API

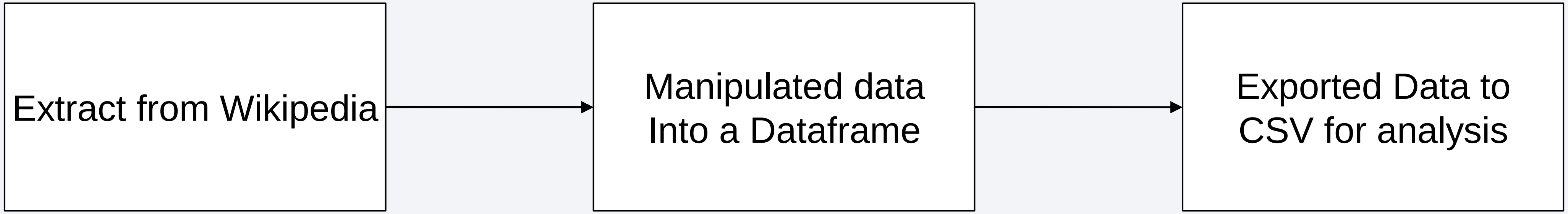
- Requested and Parsed SpaceX data in JSON format directly from web API.
- Removed irrelevant data, such as information relating to Falcon 1 launches
- Replaced some NaN values with the mean values for their respective columns.



Data Collection

Method 2: Web Scraping

- Requested the Falcon9 Launch Wiki page from its URL
- Extracted all column/variable names from the HTML table header using BeautifulSoup
- Created a data frame by parsing the launch HTML tables



Data Collection – SpaceX API

- As pictured, the SpaceX API was called to collect the data. In order to prepare it for analysis, data wrangling and formatting techniques were applied
- https://github.com/RDS-95/PythonCapstone/blob/main/Data%20Collection%20API%20Lab.ipynb

Now let's start requesting rocket launch data from SpaceX API with the following URL:

```
In [7]: spacex_url="https://api.spacexdata.com/v4/launches/past"

In [8]: response = requests.get(spacex_url)
```

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

```
In [12]: # Use json_normalize meethod to convert the json result into a dataframe
         response = requests.get(static_json_url).json()
         data=pd.json_normalize(response)
```

```
In [25]: # Hint data['BoosterVersion']!='Falcon 1'
         data_falcon9 = df[df.BoosterVersion!='Falcon 1']
         data_falcon9.head()
```

Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the mean and the `.replace()` function to replace `np.nan` values in the data with the mean you calculated.

```
In [59]: # Calculate the mean value of PayloadMass column
         PLMmean = data_falcon9["PayloadMass"].mean()
         PLMmean
         # Replace the np.nan values with its mean value
         data_falcon9a = data_falcon9.replace([np.NaN],"6123.547647058824")
         data_falcon9a.isnull().sum()

Out[59]: FlightNumber      0
         Date              0
         BoosterVersion    0
         PayloadMass       0
         Orbit             0
         LaunchSite        0
         Outcome           0
         Flights           0
         GridFins          0
         Reused            0
         Legs              0
         LandingPad        0
         Block             0
         ReusedCount       0
         Serial            0
         Longitude         0
         Latitude          0
         dtype: int64
```

Data Collection - Scraping

- Used HTTP GET to scrape data from wikipedia. Used BeautifulSoup to parse the data before converting it into a Pandas dataframe.
- https://github.com/RDS-95/PythonCapstone/blob/main/Data%20Collection%20with%20Web%20Scraping%20Lab.ipynb

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

In [7]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url)
html = response.content
```

Create a `BeautifulSoup` object from the HTML `response`

In [8]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(html, 'html.parser')
```

In [12]:
```
# 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")
```

Next, we just need to iterate through the `<th>` elements and apply the provided `extract_column_from_header()` to extract column name one by one

In [19]:
```
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

for row in first_launch_table.find_all('th'):
    name = extract_column_from_header(row)
    if (name != None and len(name) > 0):
        column_names.append(name)
```

After you have fill in the parsed launch record values into `launch_dict`, you can create a dataframe from it.

In [23]:
```
df=pd.DataFrame(launch_dict)
```

Data Wrangling

Training labels for the data were determined. The number of launches and the cardinality of each orbit type was calculated.

Following this, a landing outcome label was appended to our data and the resulting data set was exported to a CSV.

https://github.com/RDS-95/PythonCapstone/blob/main/EDA%20lab.ipynb

Identify and calculate the percentage of the missing values in each attribute

```
In [3]: df.isnull().sum()/df.count()*100
```

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

Out[6]: CCAFS SLC 40    55
        KSC LC 39A      22
        VAFB SLC 4E     13
        Name: LaunchSite, dtype: int64
```

```
In [7]: # Apply value_counts on Orbit column
        df.Orbit.value_counts()

Out[7]: GTO      27
        ISS      21
        VLEO     14
        PO        9
        LEO       7
        SSO       5
        MEO       3
        ES-L1     1
        HEO       1
        SO        1
        GEO       1
        Name: Orbit, dtype: int64
```

```
In [11]: # 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)
```

```
In [12]: df['Class']=landing_class
         df[['Class']].head(8)
```

EDA with Data Visualization

Charts were using Seaborn to visualize the relationship between:

1)Flight No. and Launch Site
2)Payload and Launch Site
3)Success Rate and Orbit Type
4)Flight Number and Orbit Type
5)Payload and Orbit Type
6)Yearly success trend

EDA with SQL

The data was loaded into a PostgreSQL database and queries were run to find:

1)Unique Launch Site Names

2)Total Payload Mass borne by NASA boosters

3)Average Payload mass borne by Falcon 9 v1.1 boosters

4)Total number of successes and failures

5)Failed landing outcomes with their booster version and launch site

https://github.com/RDS-95/PythonCapstone/blob/main/EDA%20with%20SQL%20(2).ipynb

Build an Interactive Map with Folium

Firstly, all launch sites were marked using markers. Different markers indicate the success or failure of a launch

These markers were used to identify launch sites with higher success rates

Folium was used to investigate the proximity of the launch site to highways, coastlines and other important features.

https://github.com/RDS-95/PythonCapstone/blob/main/Viz%20with%20Folium.ipynb

Build a Dashboard with Plotly Dash

Plotly Dash was used to create an interactive dashboard containing:

1. Visualization of total launches per site
2. Graphs showing the launch outcomes vs. Payload mass
3. Different booster versions outcomes/payload mass relationship

`

https://github.com/RDS-95/PythonCapstone/blob/main/PlotlyDash

Predictive Analysis (Classification)

Numpy and Pandas were used to load the training data sets.

Different ML models were used to train parameters using GridSearch CV.

By measuring the accuracy of each model, the best classification model was determined

https://github.com/RDS-95/PythonCapstone/blob/main/predictive%20analysis%20lab.ipynb

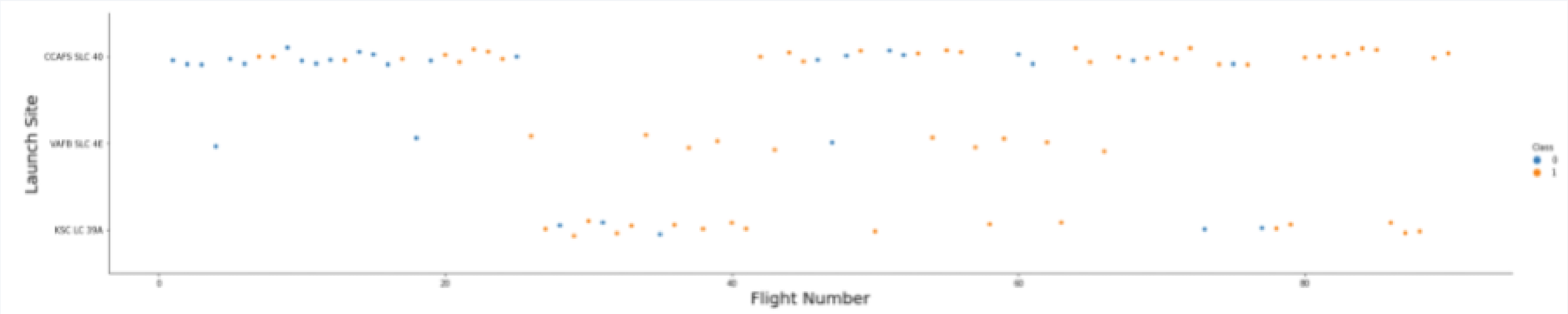
Results

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

Section 2

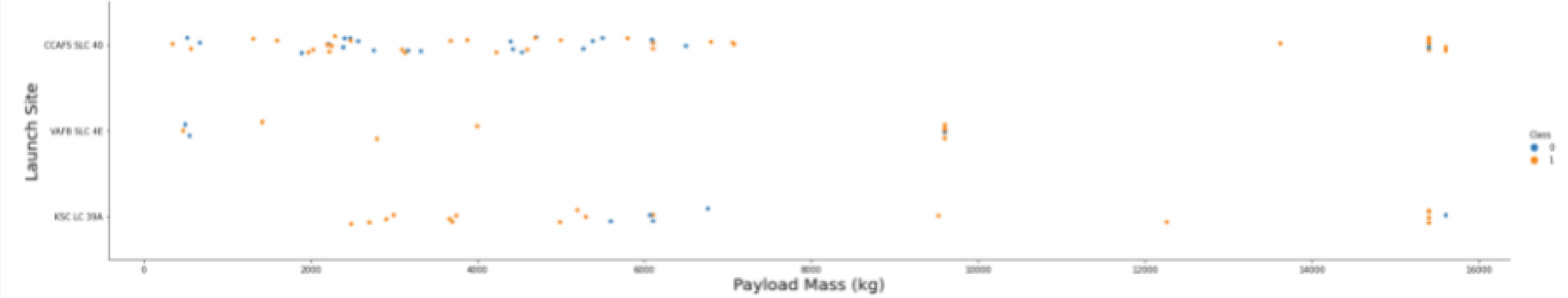
Insights drawn from EDA

Flight Number vs. Launch Site



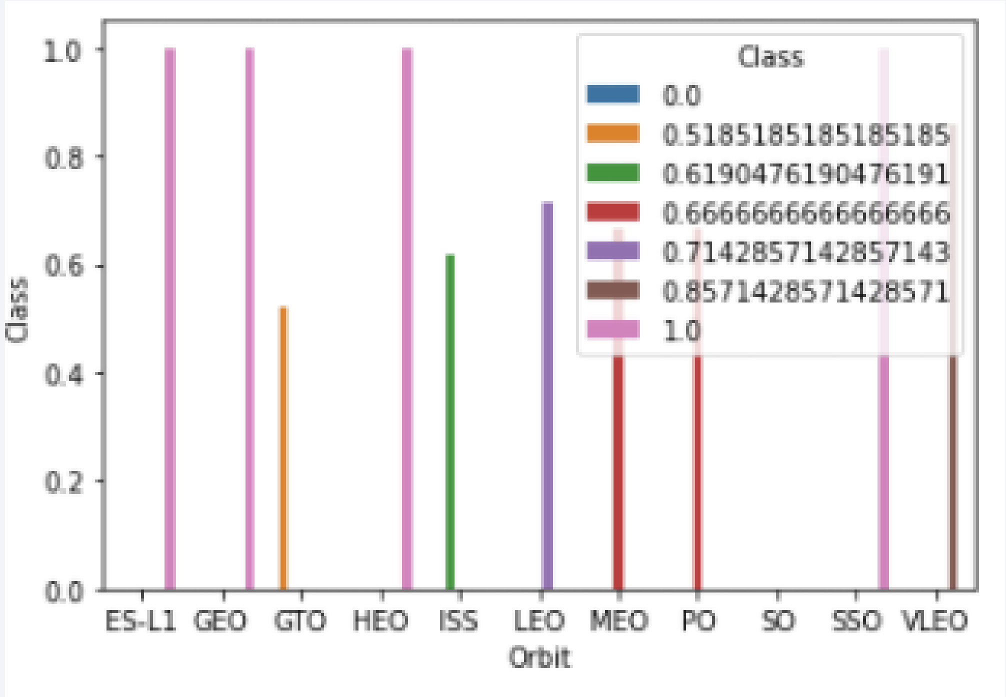
As the flight number increases, the success rate of the launch appears to trend upwards. This is across all launch sites.

Payload vs. Launch Site



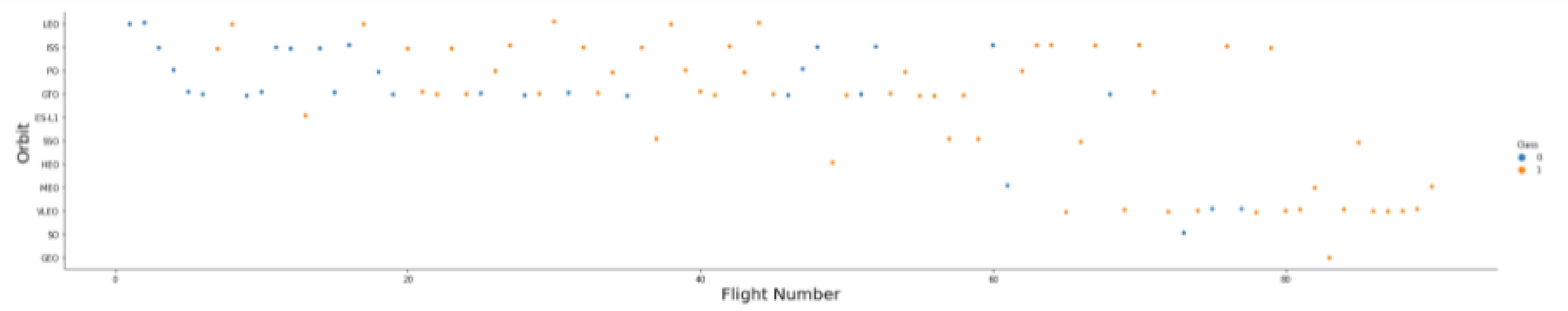
It is unclear if the higher payload mass has an impact on the success rate of the launch.

Success Rate vs. Orbit Type



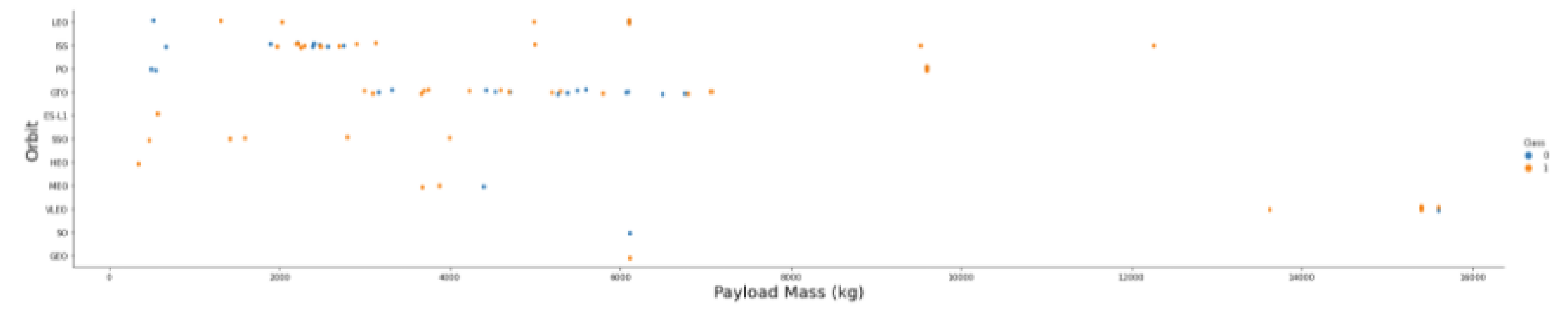
Several orbit types have a 100% success rate. Orbit type GTO has the lowest success rate at just over 50%.

Flight Number vs. Orbit Type



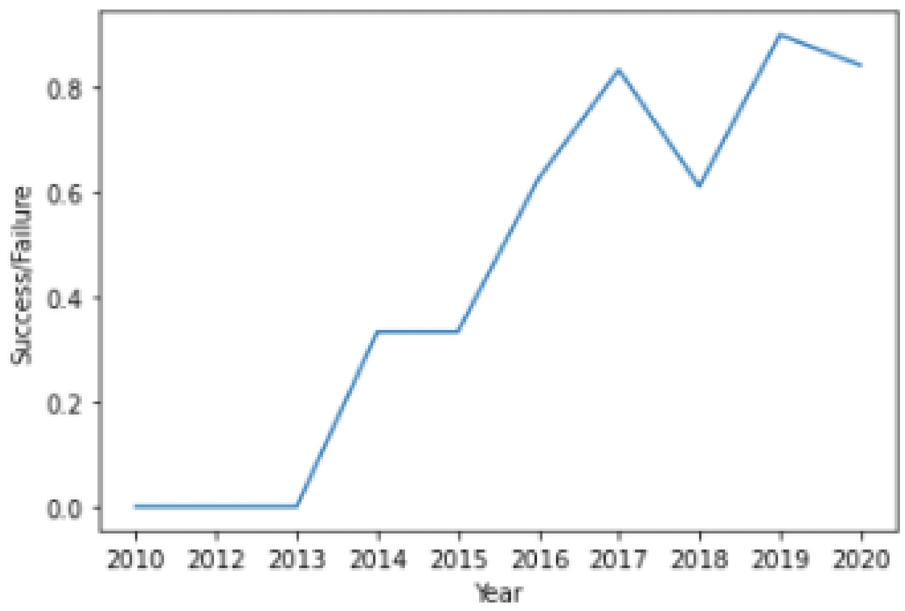
Some orbit types such as SO & ES-LI do not have enough data points to derive conclusions. For GFO & ISS it appears that the success rate increases with flight number.

Payload vs. Orbit Type



Success rater of ISS orbits increases with payload types. Similar for LEO, although there are not as many data points to provide conclusive evidence.

Launch Success Yearly Trend



you can observe that the sucess rate since 2013 kept increasing till 2020

Success rate has clearly been trending upwards.

All Launch Site Names

- DISTINCT will show us the unique names of launch sites.

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'

- Use LIKE to find the launch sites that begin with 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

- Again, we can use LIKE on the customer column to isolate NASA launches.

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

- Use of AVG on PayloadMassKG column finds the average payload

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

- The MIN date is the first successful 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

- Use AND to add clauses to the result. In this case based on the PayloadMassKG column.

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

- Use % to find the successful missions. Two separate tasks were used. One determines # successes and one # failures

```
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 can be used on PayloadMassKG to find the maximum payload.

```
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

- Use BETWEEN in addition to the previous techniques to set the date range in this query.

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

- Use GROUP BY and ORDER BY to create the desired output.

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

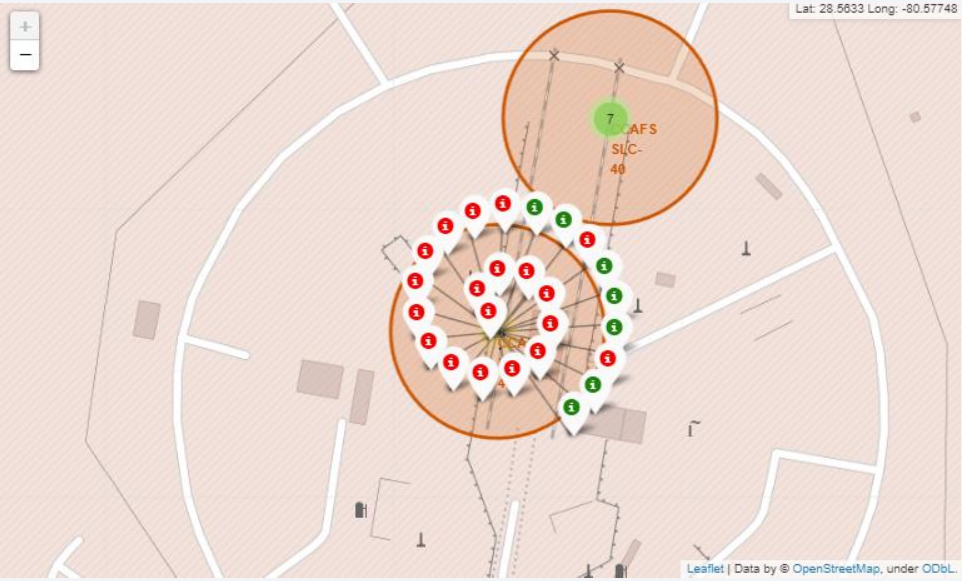
Section 3

Launch Sites Proximities Analysis

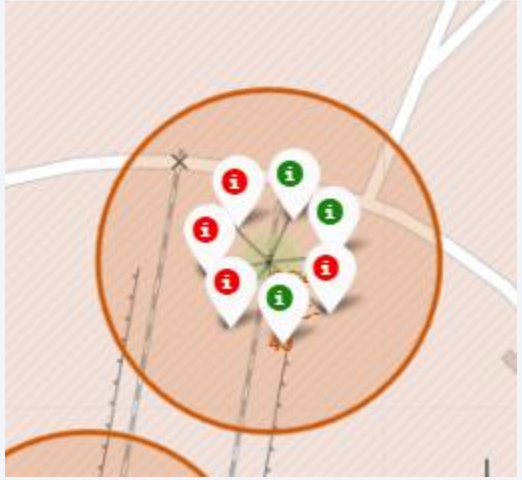
Launch Site Markers

The launch sites are concentrated around two locations; southern California and Florida

Markers on Specific Sites



CCAFS LC-40



CCAFS SLC-40

VAFS SLC-4E

KSC LC-39A

RED = failed launch
GREEN = Successful launch

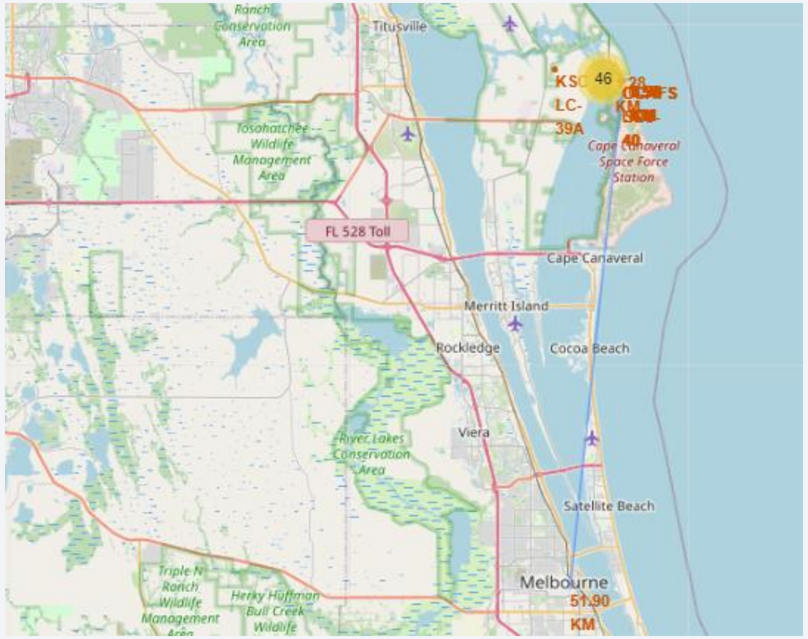
Distance to Strategic Landmarks

Coastline - 0.9km

Highway - 0.58km

Distance to Strategic Landmarks

Railway – 1.28km



City – 51.9km

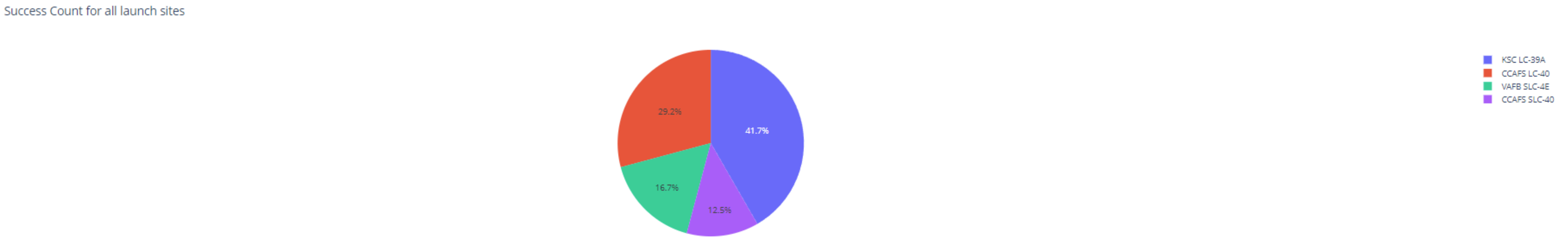
Launch sites are generally close to coastlines, highways and railways. They are ideally located far away from cities

Section 4

Build a Dashboard with Plotly Dash

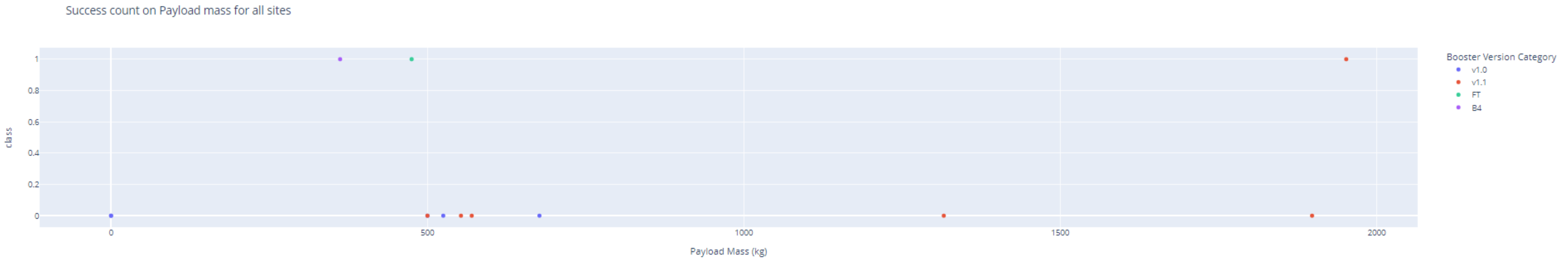
SpaceX Launch Records Dashboard

All Sites



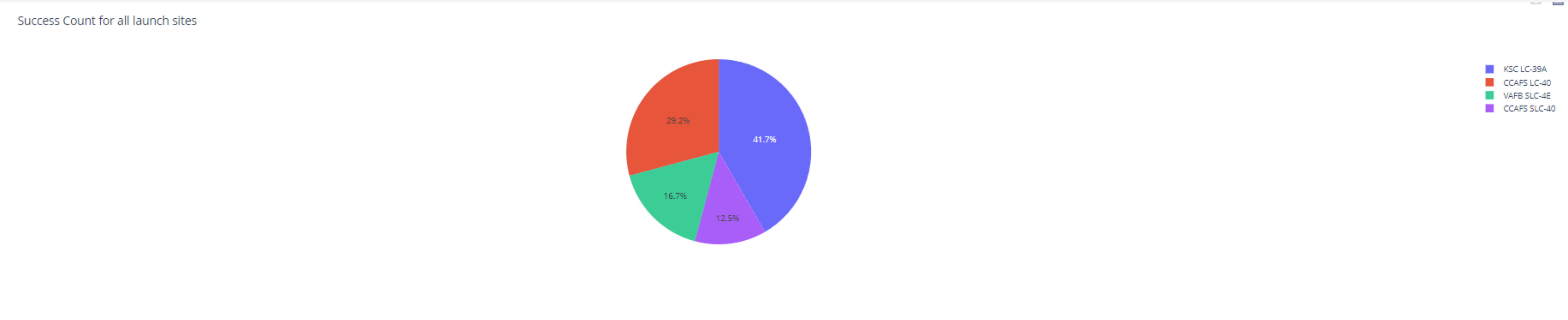
Payload range (Kg):

0 1k 2k 3k 4k 5k 6k 7k 8k 9k 10k



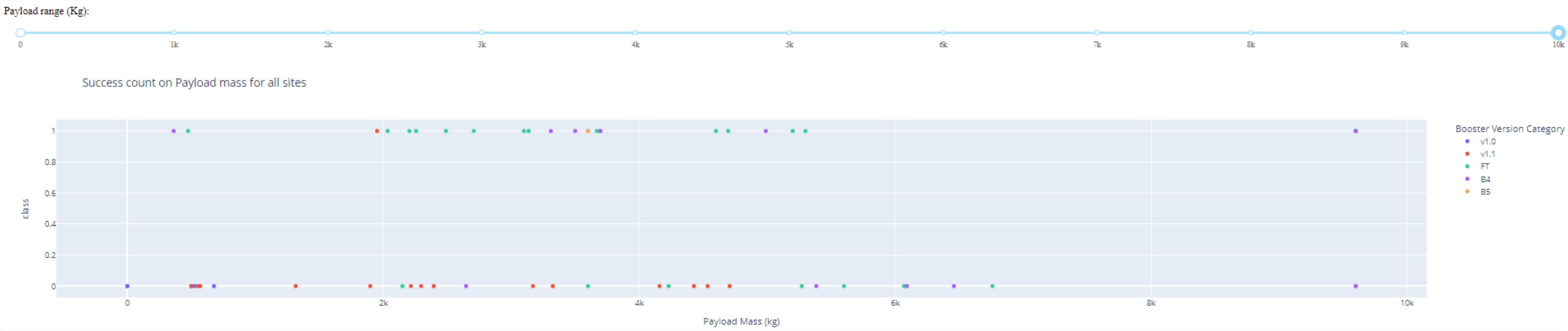
Success Count For all Launch Sites

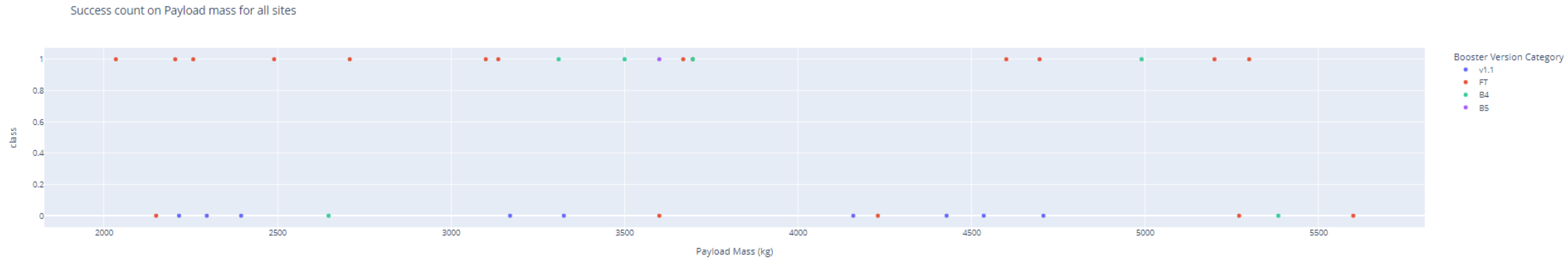
Screenshot below shows the successful launches across all sights. From Plotly Dash



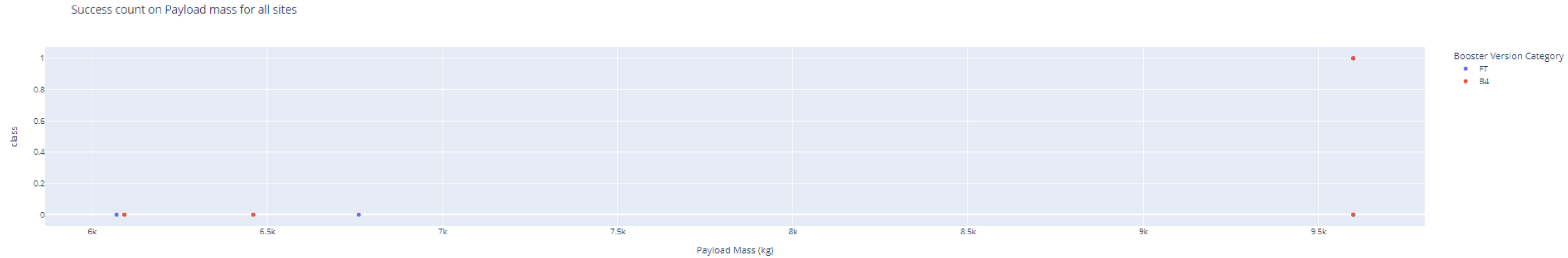
Success count on Payload mass

The screenshot below shows the success count on payload mass. The following slide shows a few different positions for the slider at the top.

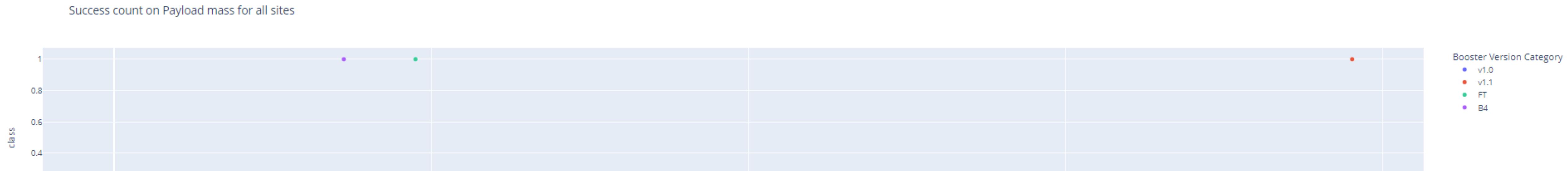




Payload range (Kg):



Payload range (Kg):

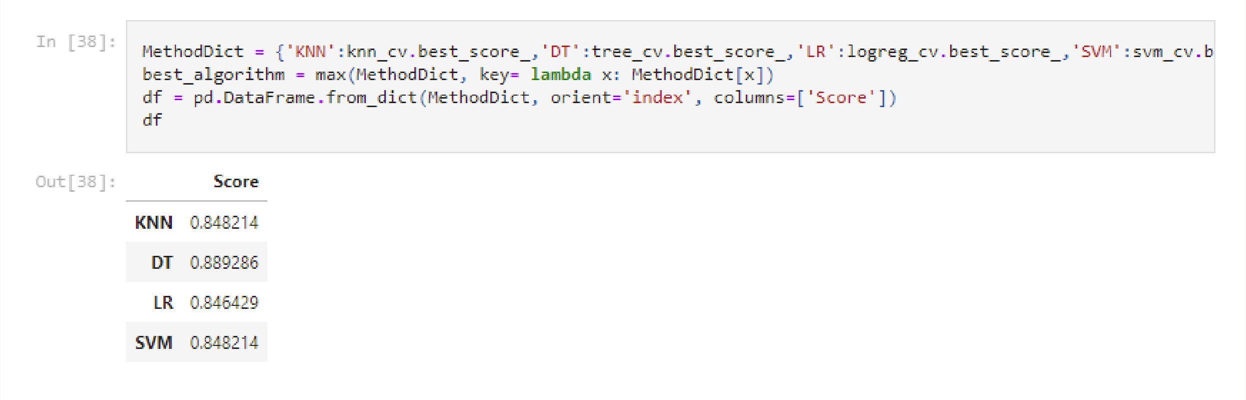


Section 5

Predictive Analysis (Classification)

Classification Accuracy

- Decision Tree has the highest accuracy. NB the scale on the bar chart has been set to accentuate the difference.

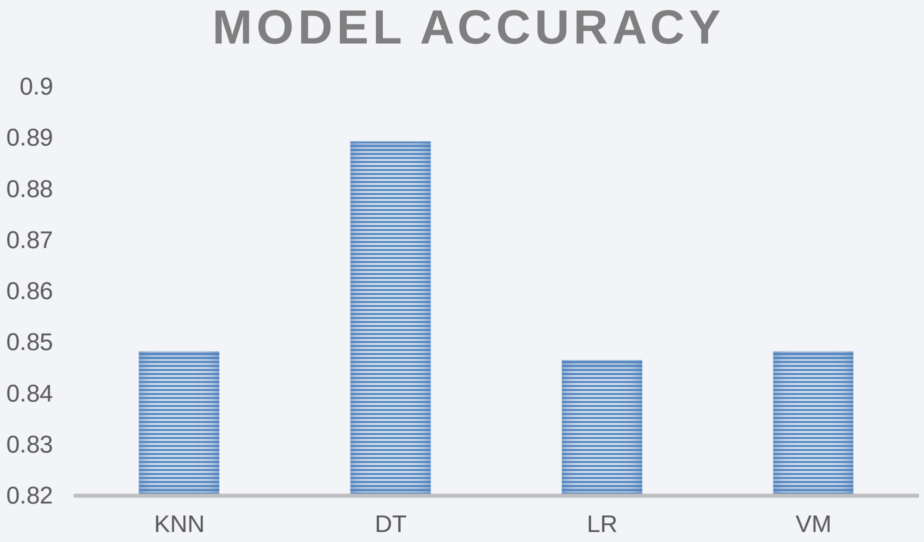


In [38]:

```
MethodDict = {'KNN':knn_cv.best_score_,'DT':tree_cv.best_score_,'LR':logreg_cv.best_score_,'SVM':svm_cv.b
best_algorithm = max(MethodDict, key= lambda x: MethodDict[x])
df = pd.DataFrame.from_dict(MethodDict, orient='index', columns=['Score'])
df
```

Out[38]:

	Score
KNN	0.848214
DT	0.889286
LR	0.846429
SVM	0.848214



Confusion Matrix

- The confusion matrix shows us that the model will accurately predict the outcome most of the time. The source of error is the model predicting a landing where the rocket did not land ie. A false positive.

Conclusions

- The launch success rate has been steadily trending upward since the first launches
- Most successful launch orbits have been ES-L1, HEO, SSO and GEO
- The most successful launch site has been KSC LC-39A
- The Decision Tree classifier provides the most accurate model for this data

Appendix

Thank you!