Winning Space Race with Data Science

An Analysis of the cost to launch and a prediction of first stage reuse at SpaceX

Karel Lewis
October 29, 2023

Outline

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

Executive Summary

Summary of methodologies

- Data collection through APIs and Web Scraping
- Data Wrangling
- Exploratory analysis using SQL
- Exploratory data analysis and data visualization using Pandas and Mathplotlib
- Machine learning predictions
- Interactive visual analytics with Folium
- Interactive Dashboard with Plotly Dash

Summary of all results

- Machine learning predictions
- Interactive visual analytics with Folium
- Interactive Dashboard with Ploty Dash

Introduction

Project background and context

Commercial space travel has become popular in recent times. Possibly the most successful provider is SpaceX. They have sent spacecrafts to international space stations and have used this equipment to provide satellite internet. Launching rockets into space is an expensive venture involving many stages. SpaceX advertised on their website that their Falcon 9 rocket launches into space for 62million dollars. Other competitors charge at least 165 million dollars. Their savings come from reusing the first stage of the launch. As a data scientist for a competitor the aim is to determine using machine learning to predict the first stage landing successfully. This information will be used to determine a suitable price to bid against SpaceX.

Problems you want to find answers

- The factors that affect the rocket landing successfully
- The effects of different rocket variables on a successfully landing outcome
- The best conditions to increase the probability of a successful landing

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX REST API and web scrapping from Wikipedia
- Perform data wrangling
  - Data was processed using one-hot encoding
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Build, tune, evaluate classification models

Data Collection

- Data collection is the process of gathering and collecting information on relevant variables from targeted systems. The data will then be processed to answer the relevant questions like predicting the cost of a launch.
- The dataset was collected via SpaceX REST API and Web Scrapping from Wikipedia
- The data was then put in a dataframe and filtered for particular requirements.

Data Collection – SpaceX API

SpaceX API calls notebook code cell and outcome cell

```
[9]: static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

We should see that the request was successfull with the 200 status response code

```
[10]: response.status_code
```

```
[10]: 200
```

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

```
[13]: # Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows that have multiple payloads in a single rocket.
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in the list and replace the feature.
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the time
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

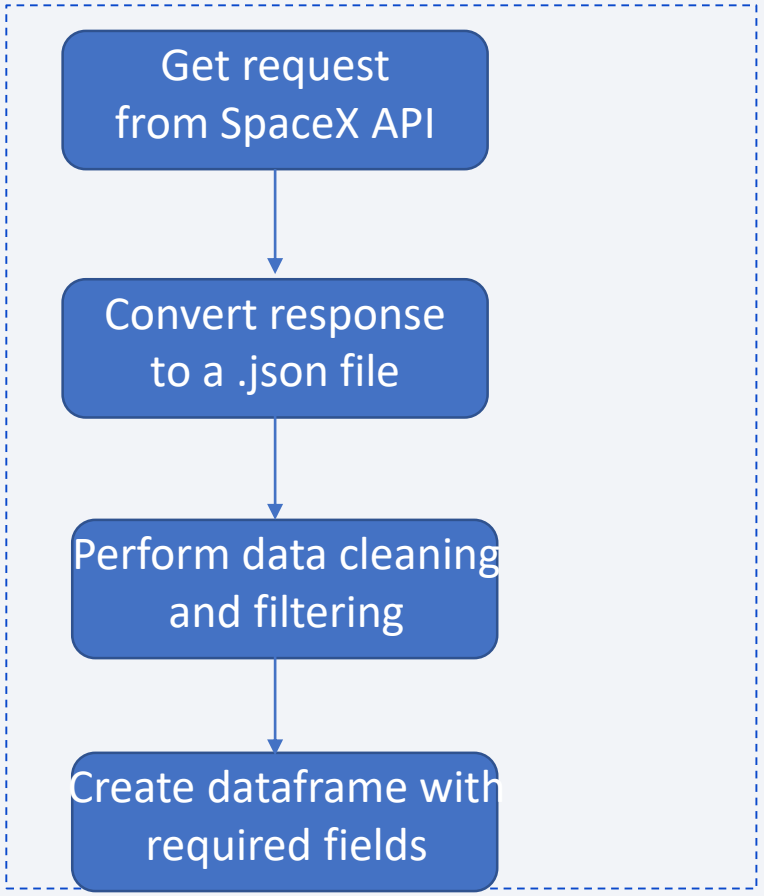
```
[26]: data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

[26]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude
4	1	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003	-80.577366	28.561857
5	2	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0005	-80.577366	28.561857
6	3	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0007	-80.577366	28.561857
7	4	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False	None	1.0	0	B1003	-120.610829	34.632093
8	5	2013-12-03	Falcon 9	3170.0	GTO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B1004	-80.577366	28.561857
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
89	86	2020-09-03	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	2	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	12	B1060	-80.603956	28.608058
90	87	2020-10-06	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	3	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	13	B1058	-80.603956	28.608058
91	88	2020-10-18	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	6	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	12	B1051	-80.603956	28.608058
92	89	2020-10-24	Falcon 9	15600.0	VLEO	CCSFS SLC 40	True ASDS	3	True	True	True	5e9e3033383ecbb9e534e7cc	5.0	12	B1060	-80.577366	28.561857
93	90	2020-11-05	Falcon 9	3681.0	MEO	CCSFS SLC 40	True ASDS	1	True	False	True	5e9e3032383ecb6bb234e7ca	5.0	8	B1062	-80.577366	28.561857

90 rows × 17 columns

Flowchart of SpaceX API calls



GitHub URL:

Data Collection - Scraping

Web Scraping notebook code cell and outcome cell

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

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

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

```
[8]: # 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")
```

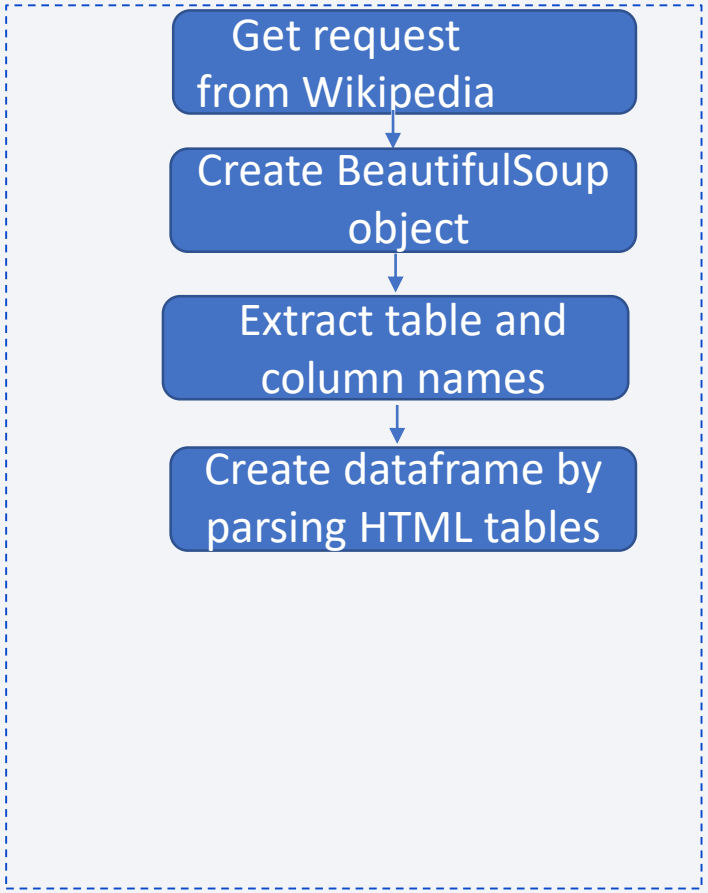
```
[10]: column_names = []

      # Apply find_all() function with `th` element on first_launch_table
      th_list = first_launch_table.find_all('th')
      for th in th_list:
          name = extract_column_from_header(th)
          if name is not None and len(name) > 0:
              # Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
              column_names.append(name)
```

```
[15]: new_df = pd.DataFrame(launch_dict)
      new_df["Launch outcome"]=new_df["Launch outcome"].str.replace("\n","")
      new_df["Booster landing"]=new_df["Booster landing"].str.replace("\n","")
      new_df.tail(50)
```

[15]:

	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
71	72	VAFB	RADARSAT Constellation	4,200 kg	SSO	Canadian Space Agency	Success	F9 B5B1051.2	Success	12 June 2019	14:17
72	73	CCAFS	SpaceX CRS-18	2,268 kg	LEO	NASA	Success	F9 B5B1056.2	Success	25 July 2019	22:01
73	74	CCAFS	AMOS-17	6,500 kg	GTO	Spacecom	Success	F9 B5B1047.3	No attempt	6 August 2019	23:23
74	75	CCAFS	Starlink	15,600 kg	LEO	SpaceX	Success	F9 B5	Success	11 November 2019	14:56
75	76	CCAFS	SpaceX CRS-19	2,617 kg	LEO	NASA	Success	F9 B5	Success	5 December 2019	17:29
76	77	CCAFS	JCSat-18	6,956 kg	GTO	Sky Perfect JSAT	Success	F9 B5B1056.3	Success	17 December 2019	00:10
77	78	CCAFS	Starlink	15,600 kg	LEO	SpaceX	Success	F9 B5	Success	7 January 2020	02:19:21

Flowchart of web scraping



GitHub URL:

Data Wrangling

- Data wrangling involves cleaning up the data so that exploratory analysis can be done and outcomes can be assigned training labels.

Data wrangling notebook code cell and outcome cell

```
[2]: df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_1.csv")
df.head(10)
```

[2]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.561857
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561857

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

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

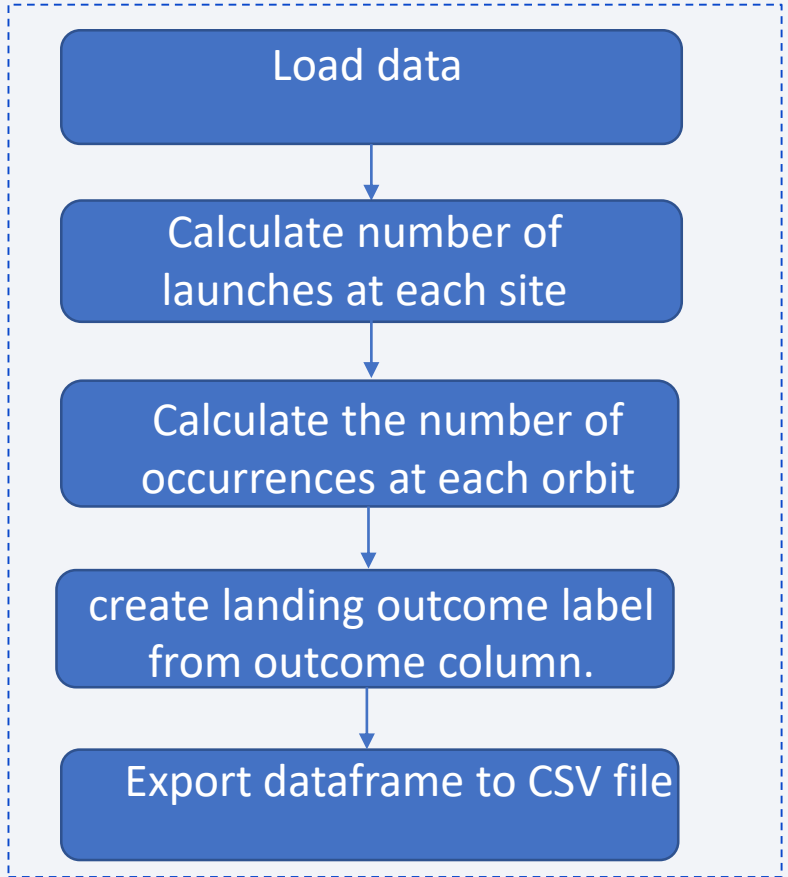
```
[6]: # Apply value_counts on Orbit column
df.Orbit.value_counts()
```

```
[6]: 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
```

```
[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)
```

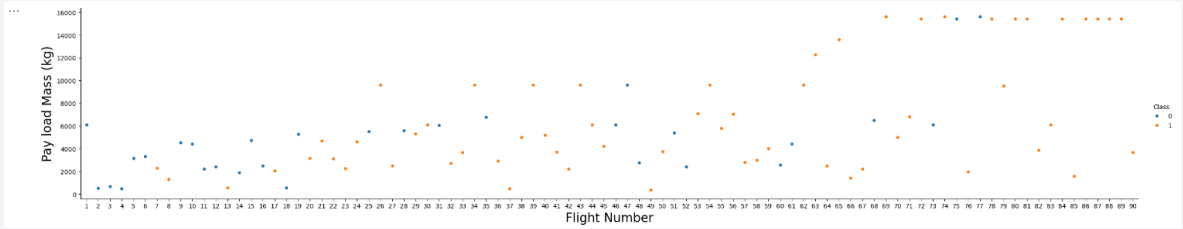
GitHub URL:

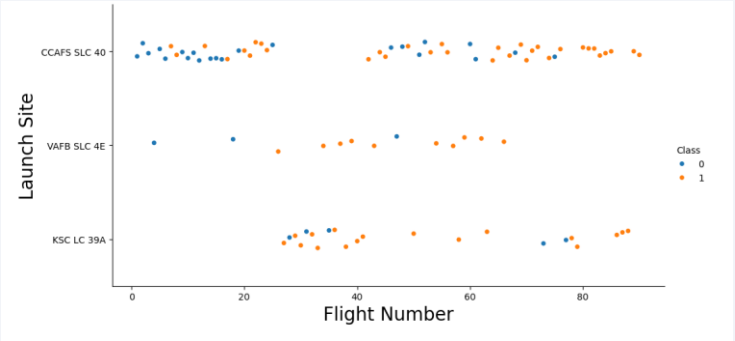
Flowchart of Data Wrangling

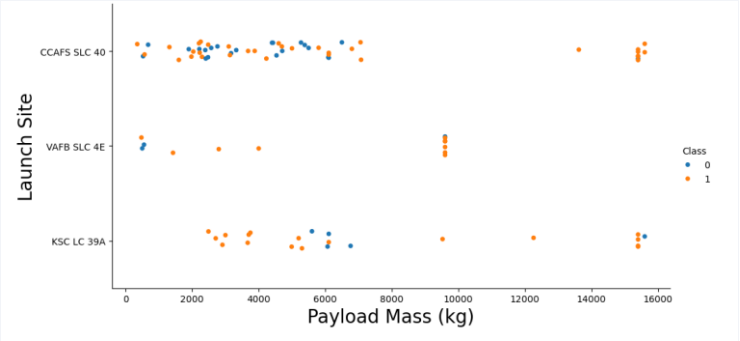


EDA with Data Visualization

- Scatterplots drawn to find the relationship between the attributes such as between:
- Payload and Flight Number.
- Flight Number and Launch Site.
- Payload and Launch Site.
- Flight Number and Orbit Type.
- Payload and Orbit Type.





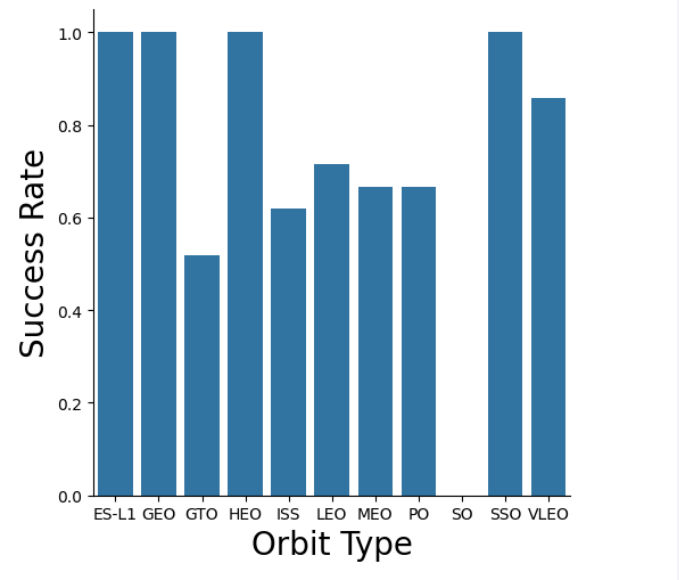


- The scatterplots show the dependency between different attributes. These plots will help to determine the most important factors affecting successful landing outcomes

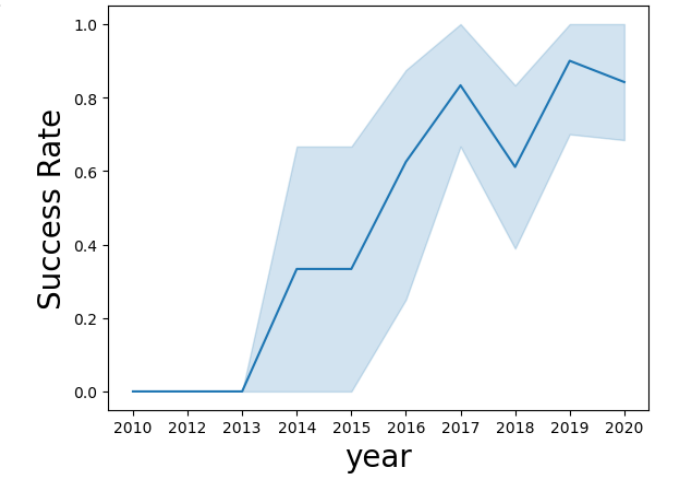
GitHub URL:

EDA with Data Visualization

- Bar chart showing Success rate against Orbit type
- It can be determined easily which Orbit type has the highest success rate



- A line graph is used to show the success rate over time
- This line graph can be used to predict future yearly trends



GitHub URL:

EDA with SQL

SQL queries were used to query the database to gain insights to the data collected and cleaned in previous labs. These queries were performed:

- Using SQL, we had performed many queries to get better understanding of the dataset, Ex:
- Displaying the names of the launch sites.
- Displaying 5 records where launch sites begin with the string ‘CCA’.
- Displaying the total payload mass carried by booster launched by NASA (CRS).
- Displaying the 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.
- Listing the failed landing outcomes in drone ship, their booster versions, and launch sites names for in year 2015.
- Rank the count of landing outcomes or success between the date 2010-06-04 and 2017-03-20, in descending order.

GitHub URL:

Build an Interactive Map with Folium

An interactive map was built with Folium to visualize the location of the launch sites. The latitude and longitude coordinates were plot on a map with a circle marker around each location.

- These map objects were used
- Map Marker, to create marks on the map; code folium.Marker(
- Icon Marker, to create icons on the map; code folium.icon(
- Circle Marker, to create a circle around where markers are placed; code folium.Circle(
- PolyLine, to create a line between points; code folium.PolyLine(
- Marker Cluster Object, put red and green markers to easily identify success and failure launches respectively ; code MarkerCluster()

GitHub URL:

Build a Dashboard with Plotly Dash

Plotly Dash was used to create an interactive dashboard for users to see

- Pie charts showing the percentage success per launch site
- Scatter Graphs showing the relation between outcome and Payload Mass (kg) for different booster versions

These plots were added so that the end user could easily visualize the effects of different variables on the outcome of a launch

GitHub URL:

Predictive Analysis (Classification)

Building A Model

-Load the dataset into Numpy and Pandas
-Standardize and transform dataset
-Split the dataset into training set and test set
-Decide which Machine Learning algorithm to use
-Set the parameters and algorithms to GridSearchCV and train dataset.

Evaluating The Model

-Check the accuracy of each model
-Get the best hyperparameters for each type of algorithms
-Plot the confusion matrix.

Improving The Model

-Use Feature Engineering and Algorithm Tuning

Finding The Best Performing Classification Model

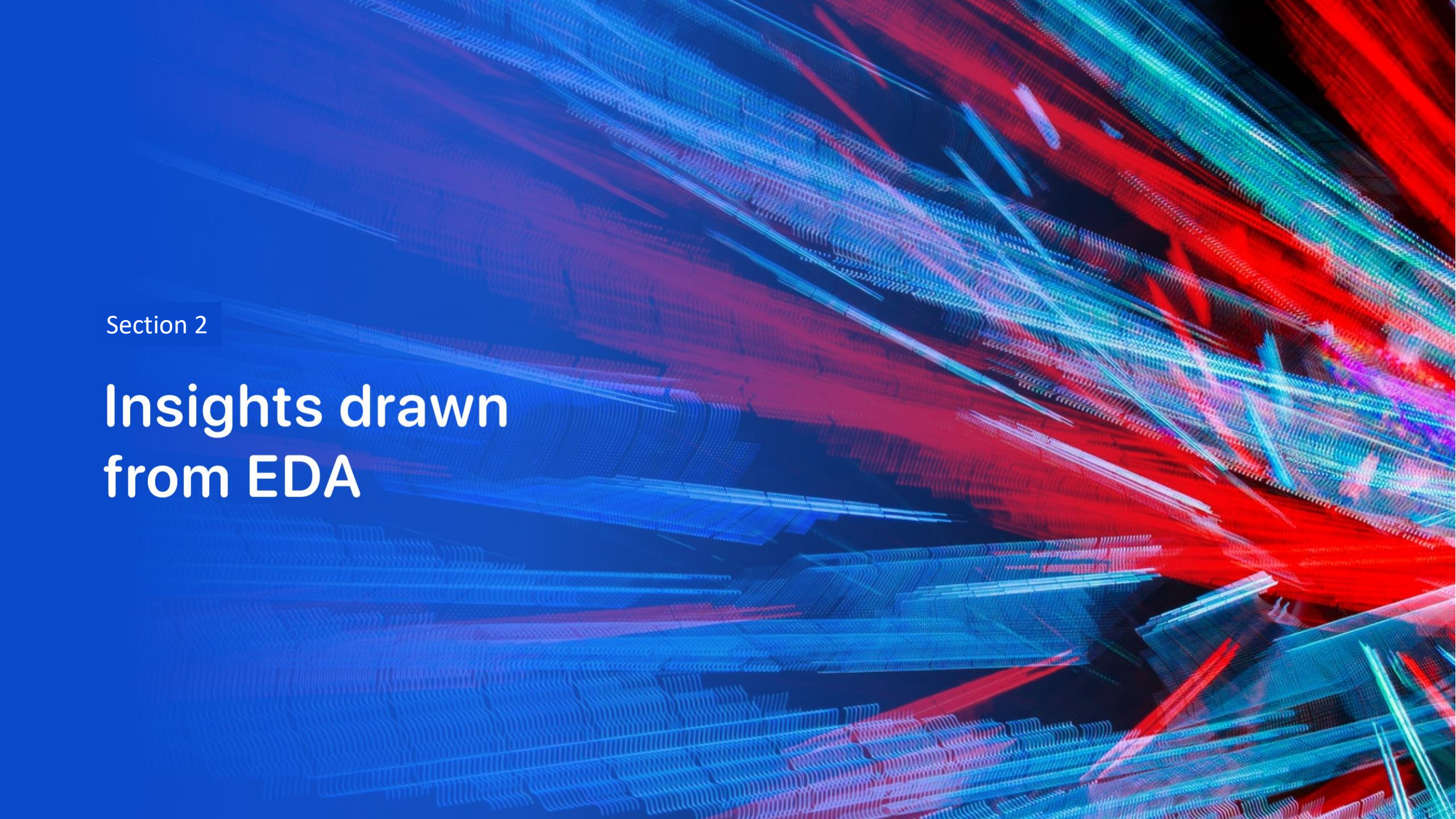
- Select the model with the best accuracy score

GitHub URL:

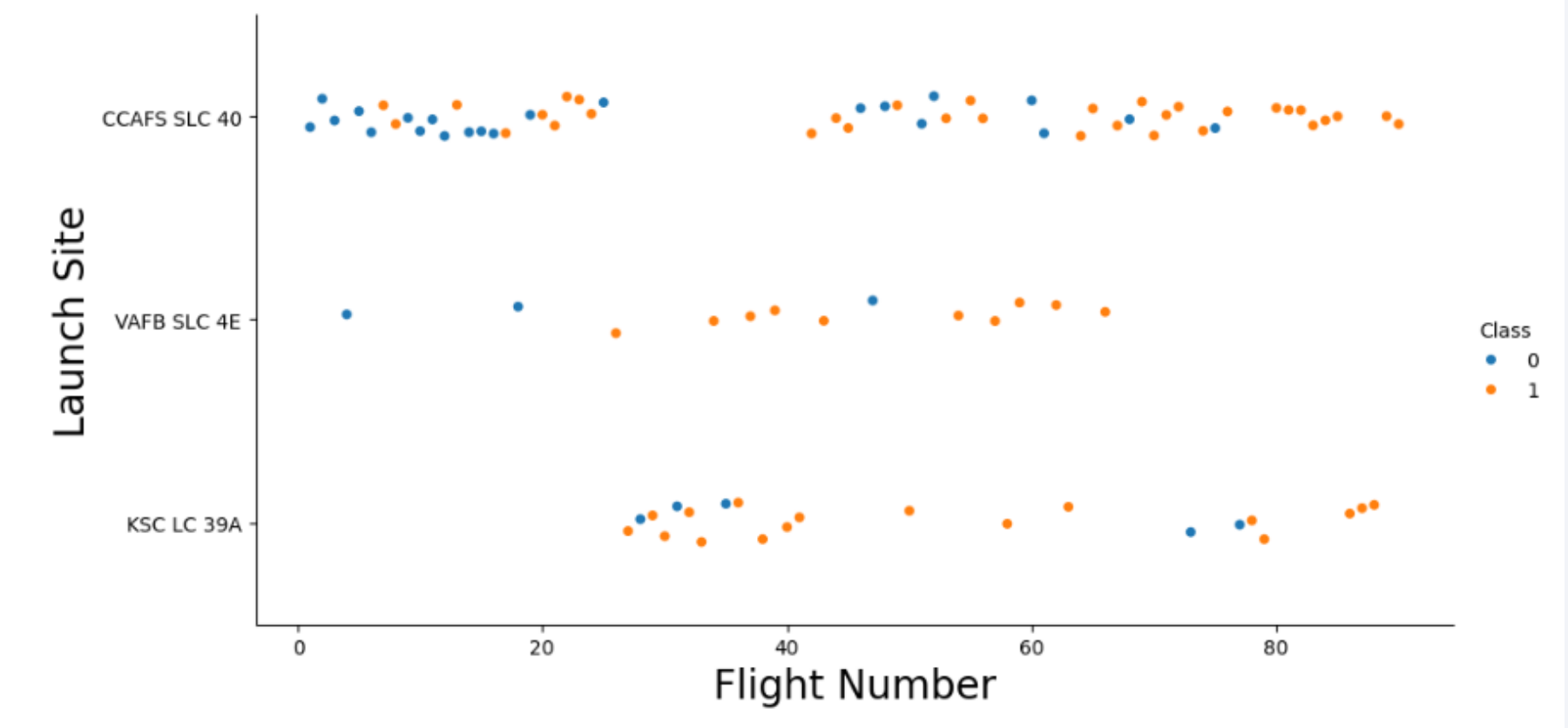
Results

Three categories of results will be shown:

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

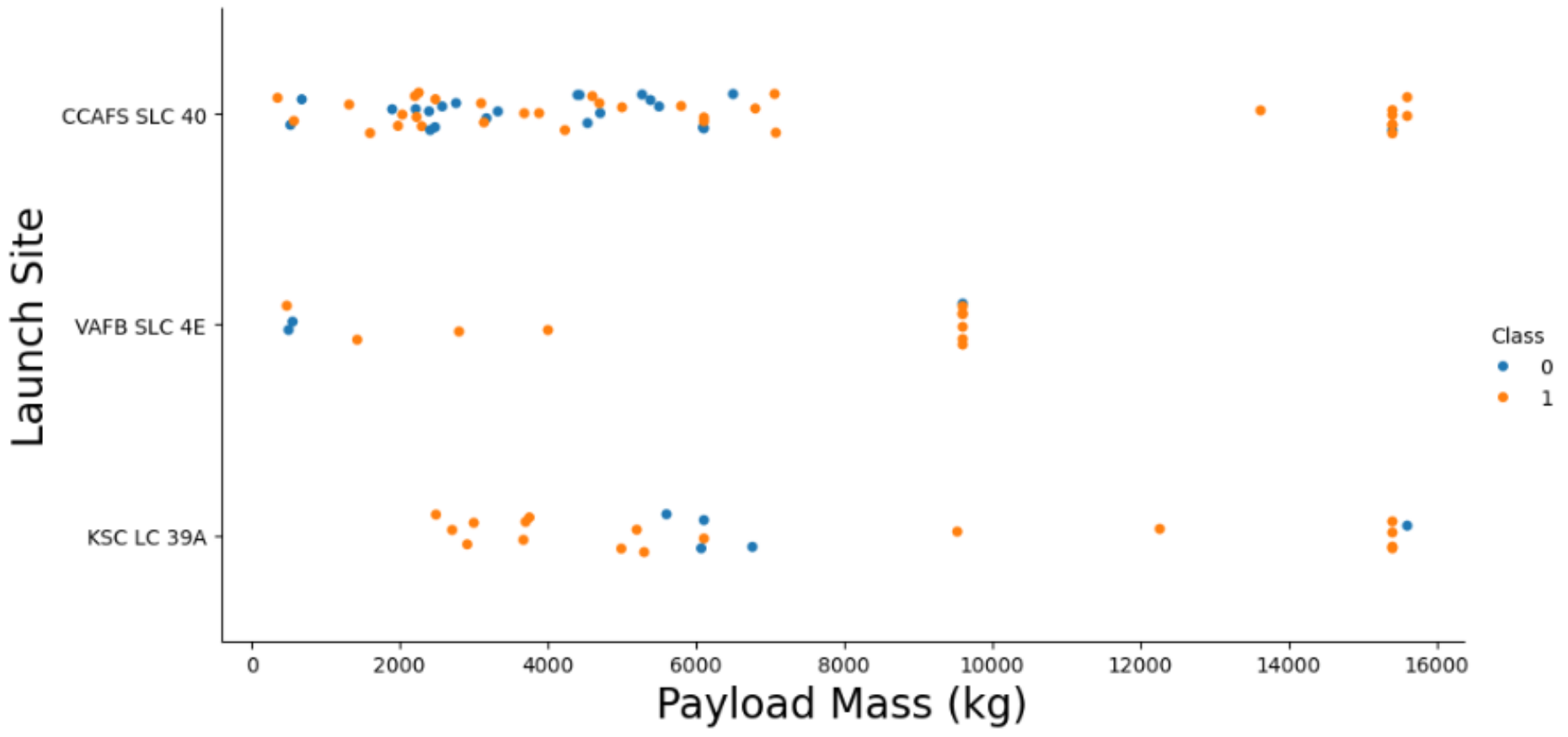

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



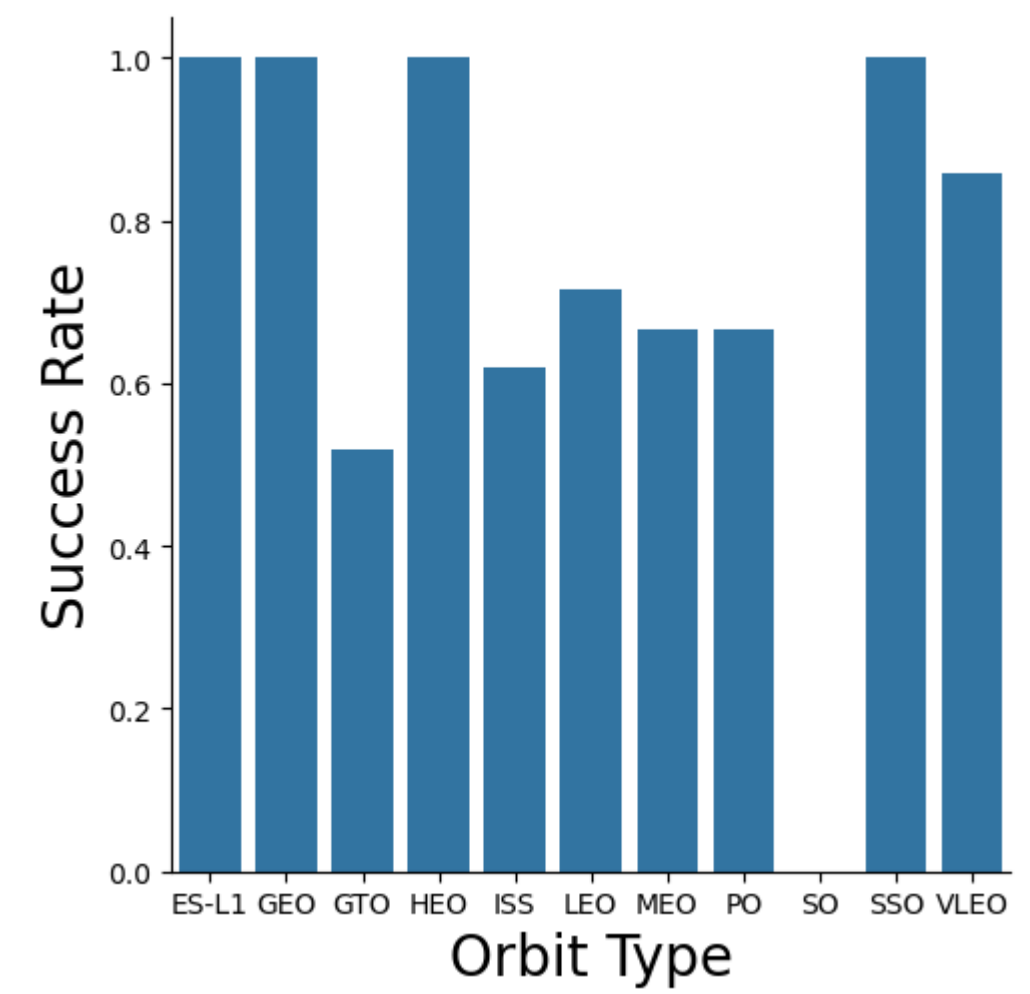
- The scatter plot shows as the number of flights from any launch site increases the number of successful launches also increases (class 1 are successful launches)
- This relationship is clearly seen in flight numbers above 30

Payload vs. Launch Site



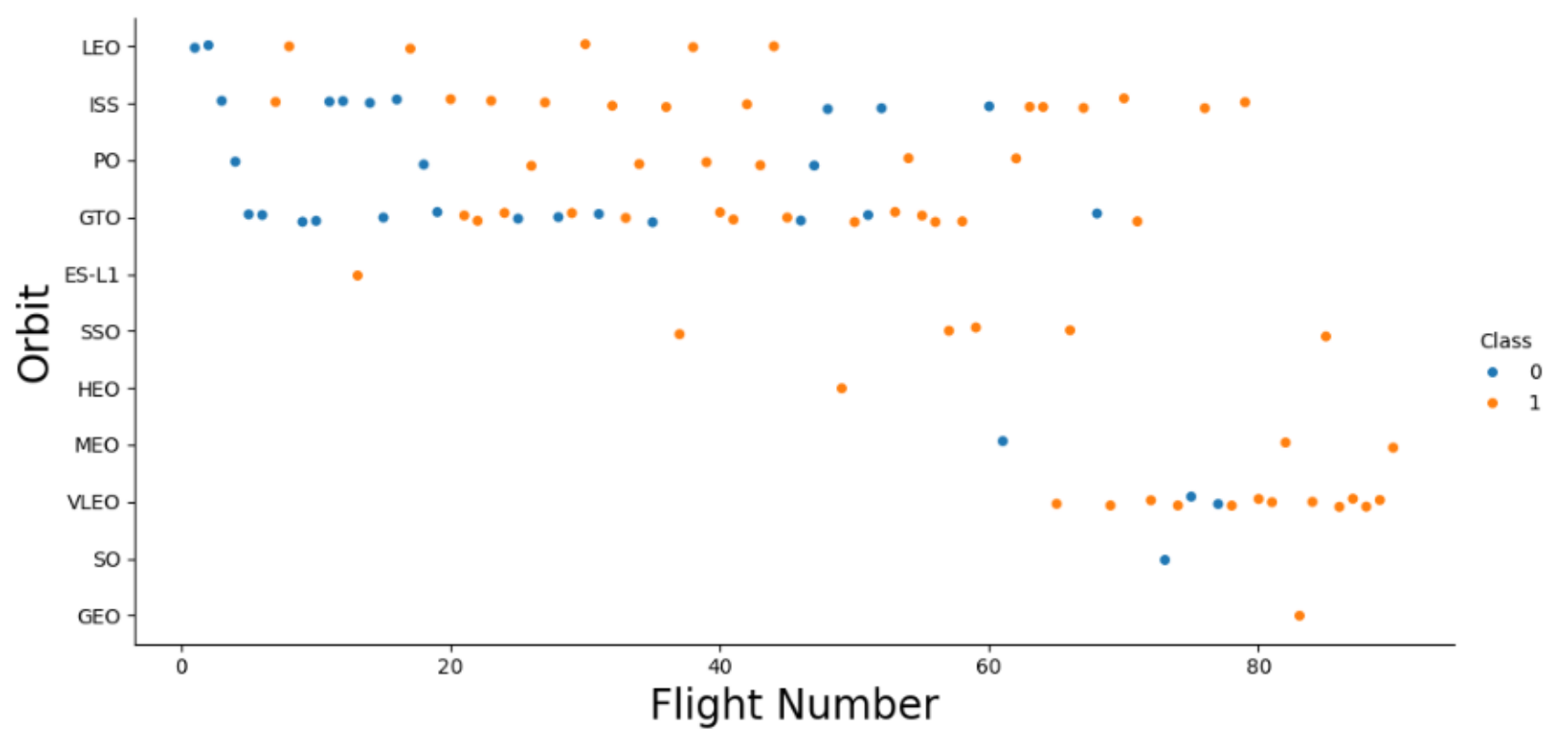
- This scatter plot shows as the Payload Mass (Kg) increases above 7000kg the successful launches increases for all launch sites.

Success Rate vs. Orbit Type



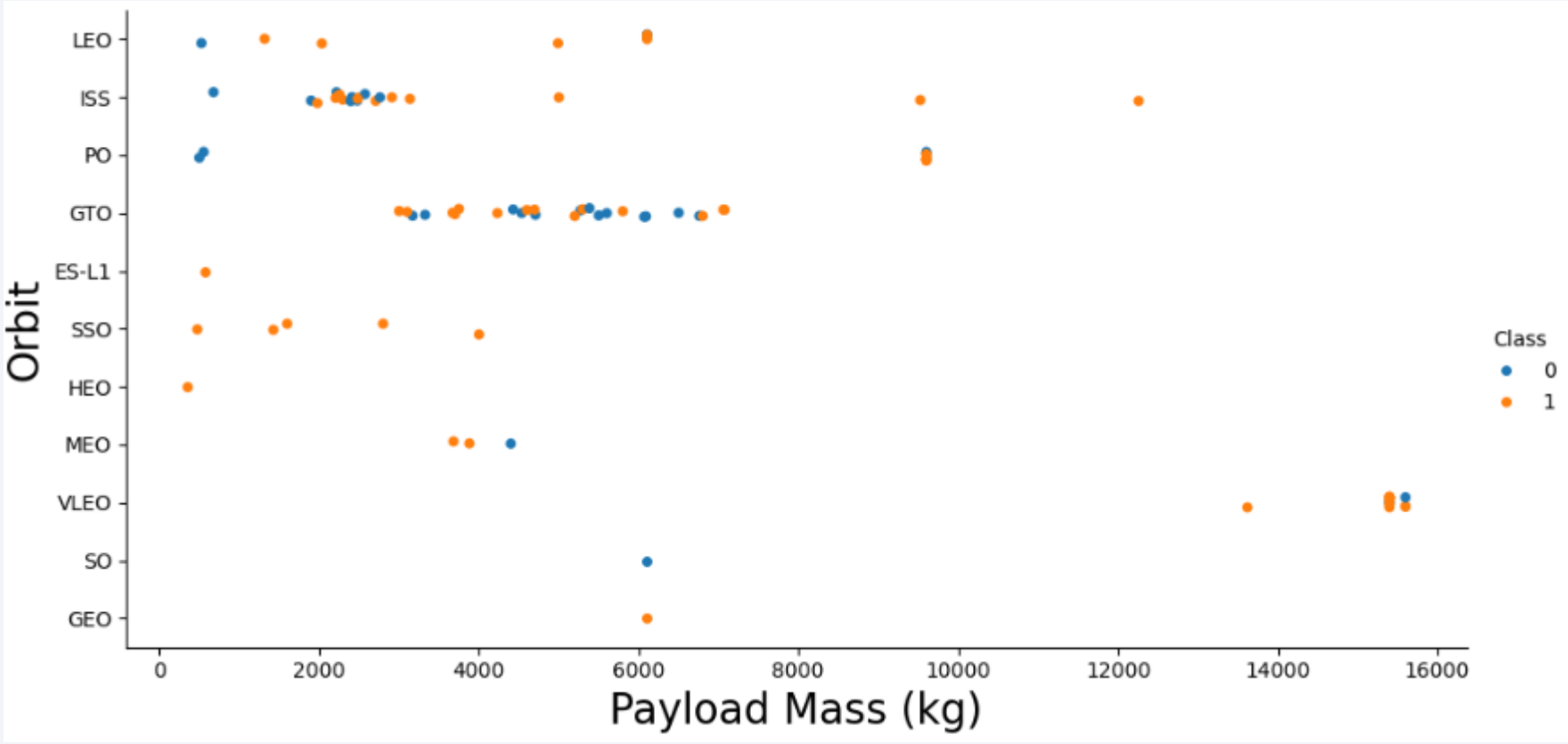
- The bar chart shows that ES-L1, GEO, HEO and SSO orbit types all had a 100% success rate.
- SO had a 0% success rate
- All other orbit types had success rates above 50%
- Further analysis is needed to determine if orbit type was the only factor here affecting the success rate

Flight Number vs. Orbit Type



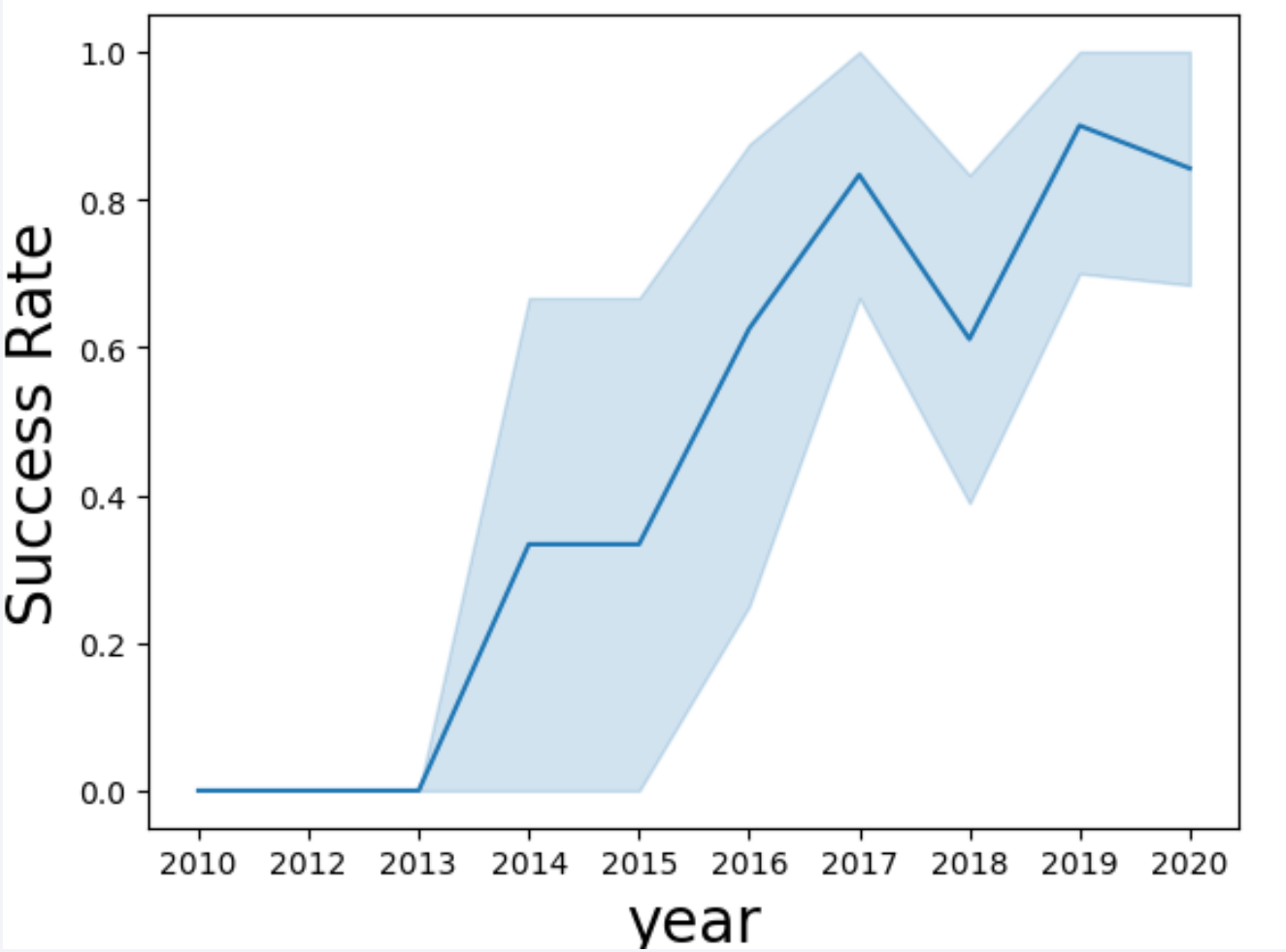
- From the scatter plot there is a general trend of increasing successful launches with flight number for each orbit type. This was clearly seen in the LEO orbit type
- The GTO orbit does not show that increasing flight number increases the launch success rate
- There was not enough data for orbit types SO, GEO and HEO

Payload vs. Orbit Type



- From the scatter plot as the Payload increases so does successful launches for orbit types ISS, LEO and PO
- For orbits MEO and VLEO at higher Payloads there were increased launched failures
- Payload had no effect on orbits SSO, GTO
- The was not sufficient data on the other orbit types to draw any conclusions about the effects of Payload

Launch Success Yearly Trend



- The line graph shows that from 2013 there have been steady increase in the launch success rate. The were two anomalies in 2018 and in 2020 in that these were the only two years after 2013in which the success rate was less than the previous year

All Launch Site Names

```
In [8]: %sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL;
 * sqlite:///my_data1.db
Done.
```

Out[8]:

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

- Using the keyword DISTINCT ensured that that the query would list launch sites from the SpaceX table only once.

Launch Site Names Begin with 'CCA'

In [9]:
```
%sql SELECT * from SPACEXTBL where (LAUNCH_SITE) LIKE 'CCA%' LIMIT 5;
```

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

Out[9]:

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

- In the query using the LIKE keyword in the condition with the wild card ‘CCA%’ ensures that the launch sites selected must begin with letters ‘CCA’
- The keyword LIMIT followed by 5 causes only 5 rows to be fetched from the SpaceX table

Total Payload Mass

In [10]:
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) \
    FROM SPACEXTBL \
    WHERE CUSTOMER = 'NASA (CRS)';
```

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

Out[10]:

SUM(PAYLOAD_MASS__KG_)
45596

- The SUM function calculates the total for the selected column.
- The keyword WHERE was used to select only the NASA (CRS) customers from the customer column
- The total Payload carried by NASA (CRS) is 45596Kg

Average Payload Mass by F9 v1.1

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

```
[11]: %sql SELECT AVG(PAYLOAD_MASS__KG_) \
          FROM SPACEXTBL \
          WHERE BOOSTER_VERSION = 'F9 v1.1';
```

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

t[11]:

AVG(PAYLOAD_MASS__KG_)
2928.4

- The AVG function calculates the average for the selected Payload Mass column.
- The keyword WHERE was used to select only rows with the F9 v1.1 booster version
- The average Payload mass carried by booster version F9 v1.1 is 2928.4Kg

First Successful Ground Landing Date

```
[13]: %sql SELECT MIN(Date) \
      FROM SPACEXTBL \
      WHERE Landing_Outcome = 'Success (ground pad)';

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

t[13]:

MIN(Date)
2015-12-22

- The MIN function is used here to fetch the smallest or first date of occurrence in the date column.
- The keyword WHERE was used to select only rows where the landing outcome was successful
- The date of the first successful landing outcome on ground pad was December 22, 2015

Successful Drone Ship Landing with Payload between 4000 and 6000

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

In [15]:
```
%sql SELECT BOOSTER_VERSION \
FROM SPACEXTBL \
WHERE LANDING_OUTCOME = 'Success (drone ship)' \
AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000;
```

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

Out[15]:

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

- The keyword SELECT was used to select the Boosters version column.
- The WHERE clause and the AND keyword was used to set the criteria for the selection which was both for a successful (drone ship) landing outcome and for a Payload Mass between 4000 and 6000.

Total Number of Successful and Failure Mission Outcomes

```
[17]: %sql SELECT sum(case when MISSION_OUTCOME LIKE '%Success%' then 1 else 0 end) AS "Successful Mission", \
          sum(case when MISSION_OUTCOME LIKE '%Failure%' then 1 else 0 end) AS "Failure Mission" \
      FROM SPACEXTBL;
       * sqlite:///my_data1.db
      Done.
```

[17]:

Successful Mission	Failure Mission
100	1

- The sum function was used to total the number of success and failure missions
- The success missions were identified by using the wildcard %Success% to fetch any records in the mission_outcome column which has the word success in it
- The failure missions were identified by using the wildcard %Failure% to fetch any records in the mission_outcome column which has the word failure in it

Boosters Carried Maximum Payload

```
[18]: %sql SELECT DISTINCT BOOSTER_VERSION \
      FROM SPACEXTBL \
      WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

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

[18]:

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

- A subquery was used.
- The booster version column was selected in the first query the keyword DISTINCT was used to select unique values.
- The condition for the WHERE clause in the first query was a subquery for the maximum Payload.

2015 Launch Records

```
[23]: %sql SELECT substr(Date,6,2) as month, DATE, BOOSTER_VERSION, LAUNCH_SITE, [Landing_Outcome] \
      FROM SPACEXTBL \
      where [Landing_Outcome] = 'Failure (drone ship)' and substr(Date,0,5)='2015';
```

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

[23]:

month	Date	Booster_Version	Launch_Site	Landing_Outcome
10	2015-10-01	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
04	2015-04-14	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

- The SELECT keyword was used to choose the required field of Date, Booster Versio, Launch site and Landing outcome
- The WHERE clause was used to selection the criteria for fetching the records which was for a Failure (drone_ship) and a the date of 2015
- The AND clause stipulates that the two criteria must be true for the records to be selected

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

```
[25]: %sql SELECT [Landing_Outcome], count(*) as count_outcomes \
      FROM SPACEXTBL \
      WHERE DATE between '2010-06-04' and '2017-03-20' group by [Landing_Outcome] order by count_outcomes DESC;
       * sqlite:///my_data1.db
      Done.
```

[25]:

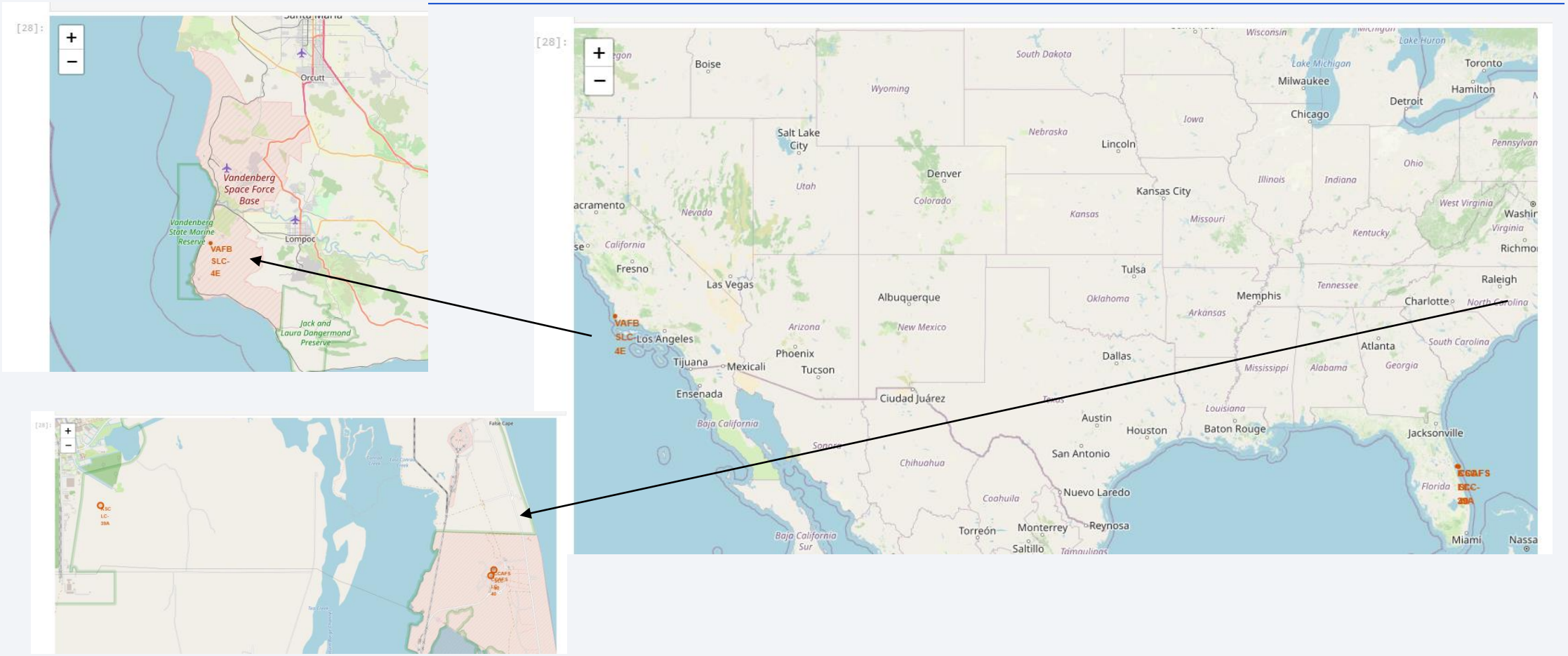
Landing_Outcome	count_outcomes
No attempt	10
Success (ground pad)	5
Success (drone ship)	5
Failure (drone ship)	5
Controlled (ocean)	3
Uncontrolled (ocean)	2
Precluded (drone ship)	1
Failure (parachute)	1

- The SELECT keyword was used to choose the Landing outcome column
- The COUNT function was used to count the landing outcomes from the records
- The WHERE clause was used to select the date criteria
- The GROUP BY clause to group the data by the landing outcomes
- The ORDER BY clause was used to determine how to organized the returned data which was down in descending order using the DESC keyword

Section 3

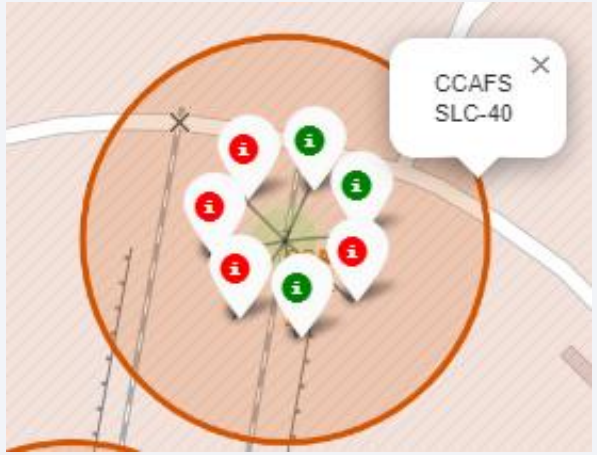
Launch Sites Proximities Analysis

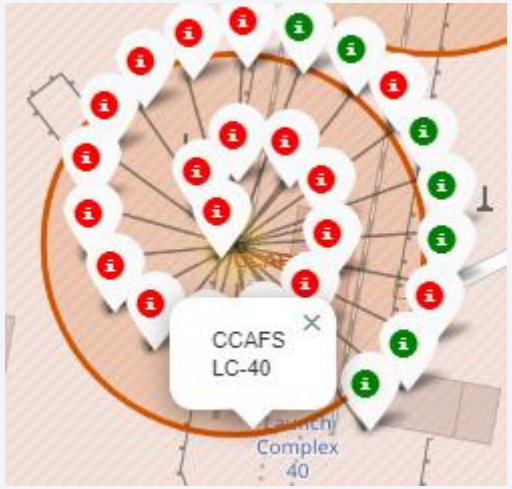
Location Of All SpaceX Launch Sites

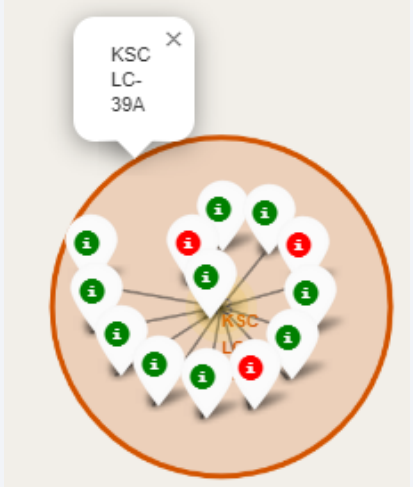


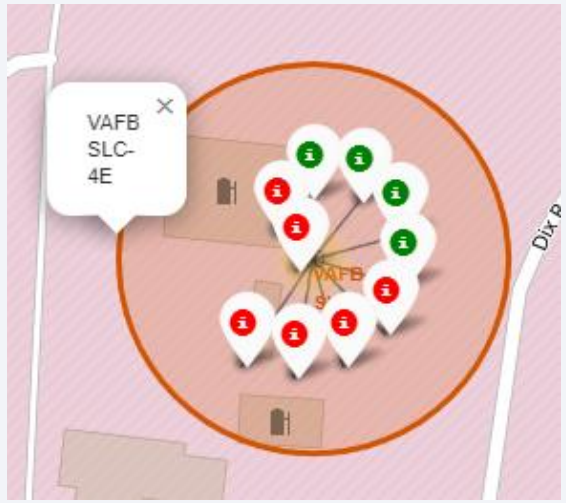
- There are four SpaceX Launch sites which are located within The United states of America in coastal areas

Color Labels Of Success/Failed Launches For Each Site On The Map



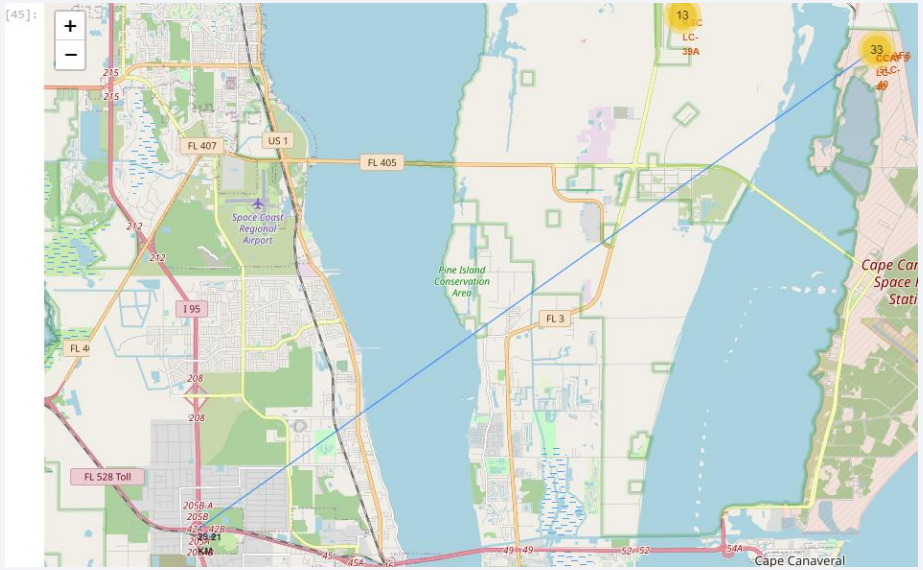


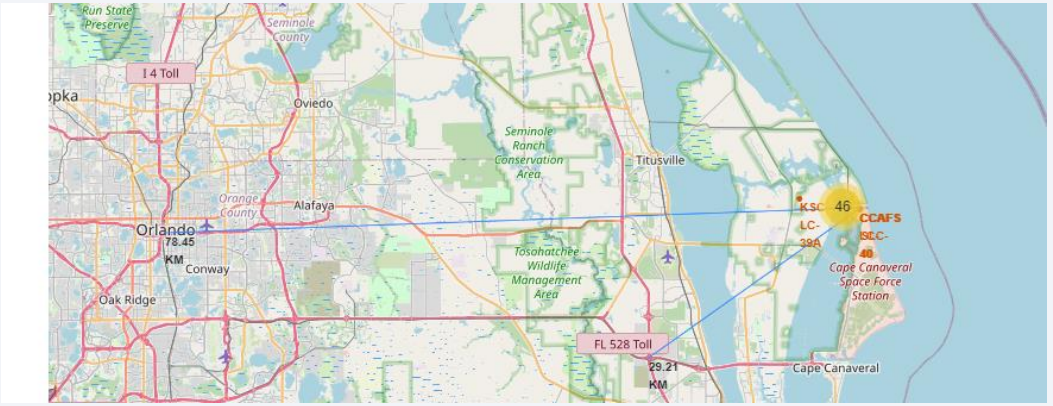




- Green marks show successful launches red marks show failure launches

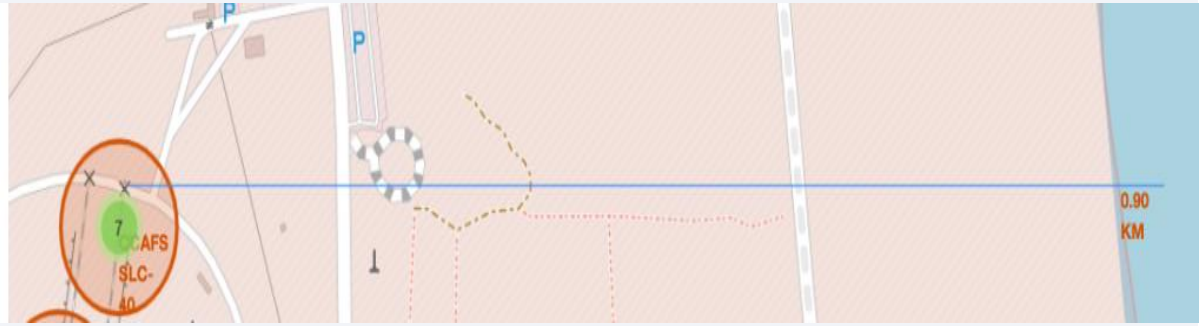
Distances Between A Launch Site To Its Proximities





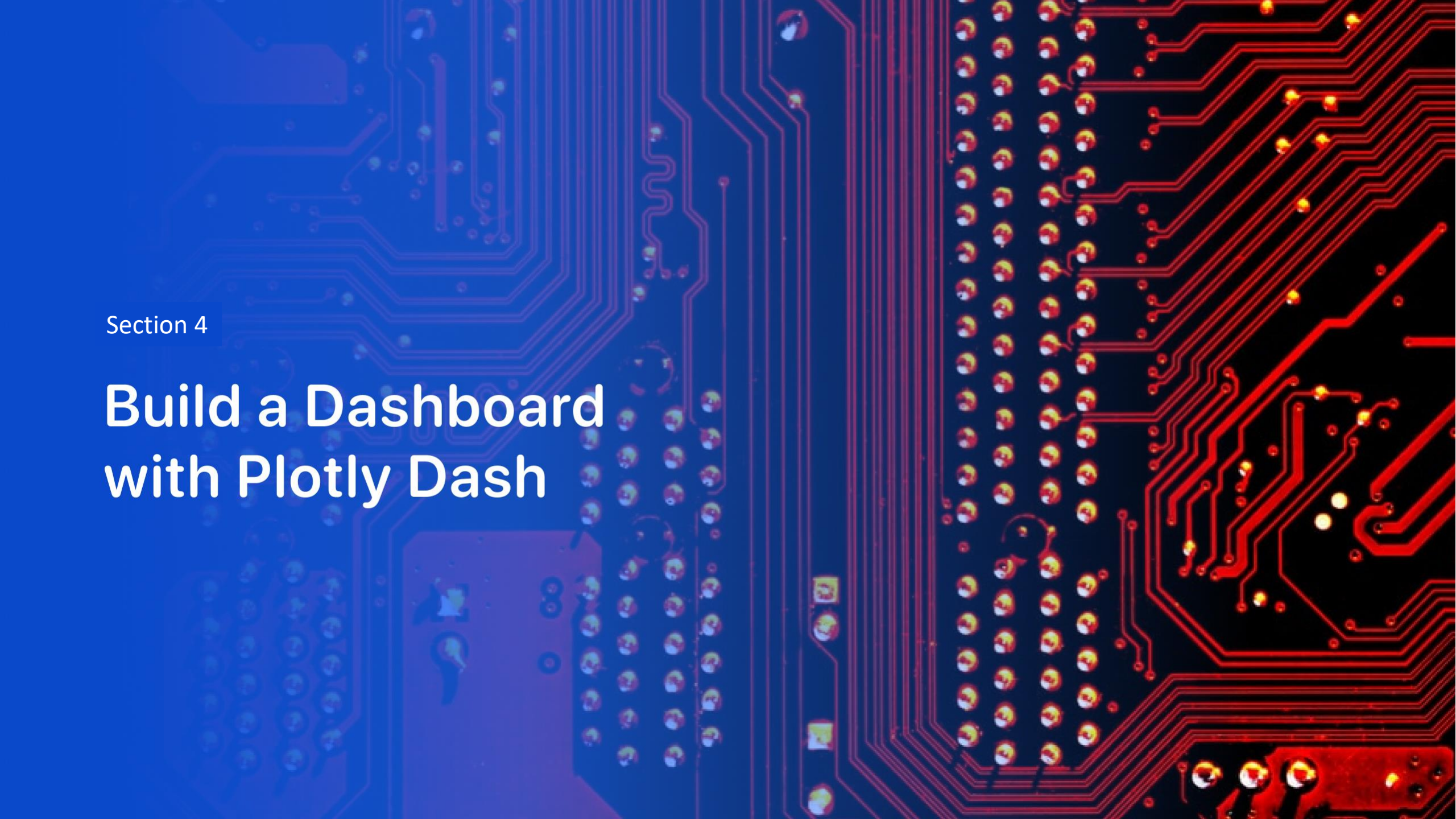
- Distance between CCAFS SLC-40 site and Florida city is 78.45km

- Distance between CCAFS SLC-40 site and Railway is 29.21km



- Distance between CCAFS SLC-40 site and coast line is 0.9km

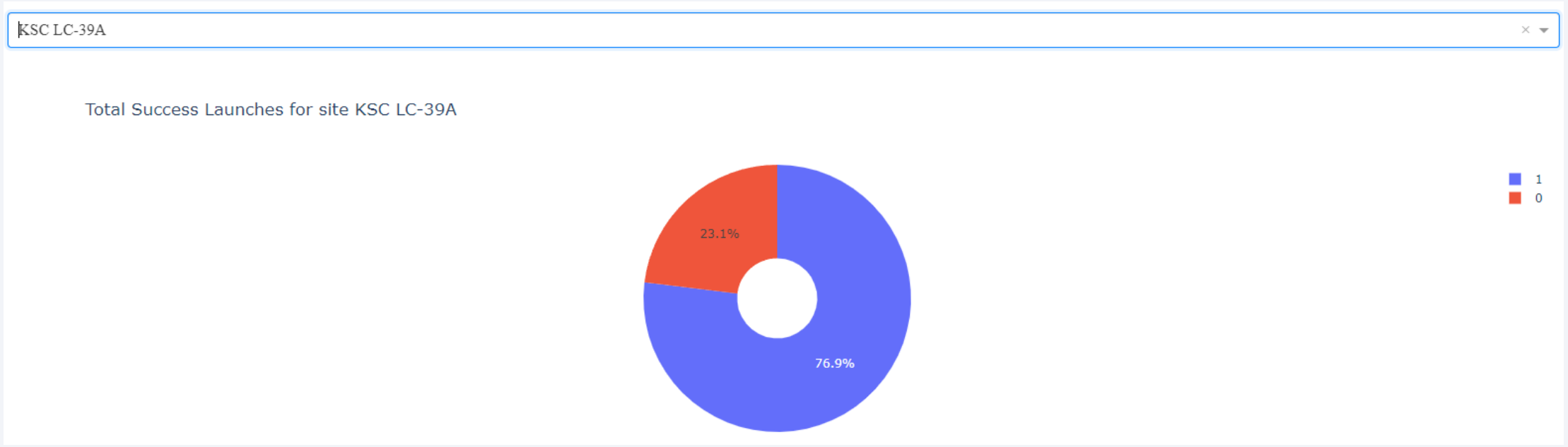
- The CCAFS SLC-40 site is close to a railway
- The CCAFS SLC-40 site far from a city
- The CCAFS SLC-40 site is close to the coast line


Section 4
Build a Dashboard with Plotly Dash

Launch Success Count For All Sites

- From the Pie Chart we see that KSC LC-39A had the most successful launches of all the sites

Launch Site With Highest Launch Success Ratio: KSC LC-39A



- From the Pie Chart we see that KSC LC-39A had a 76.9% launch success and a 23.1% launch failure

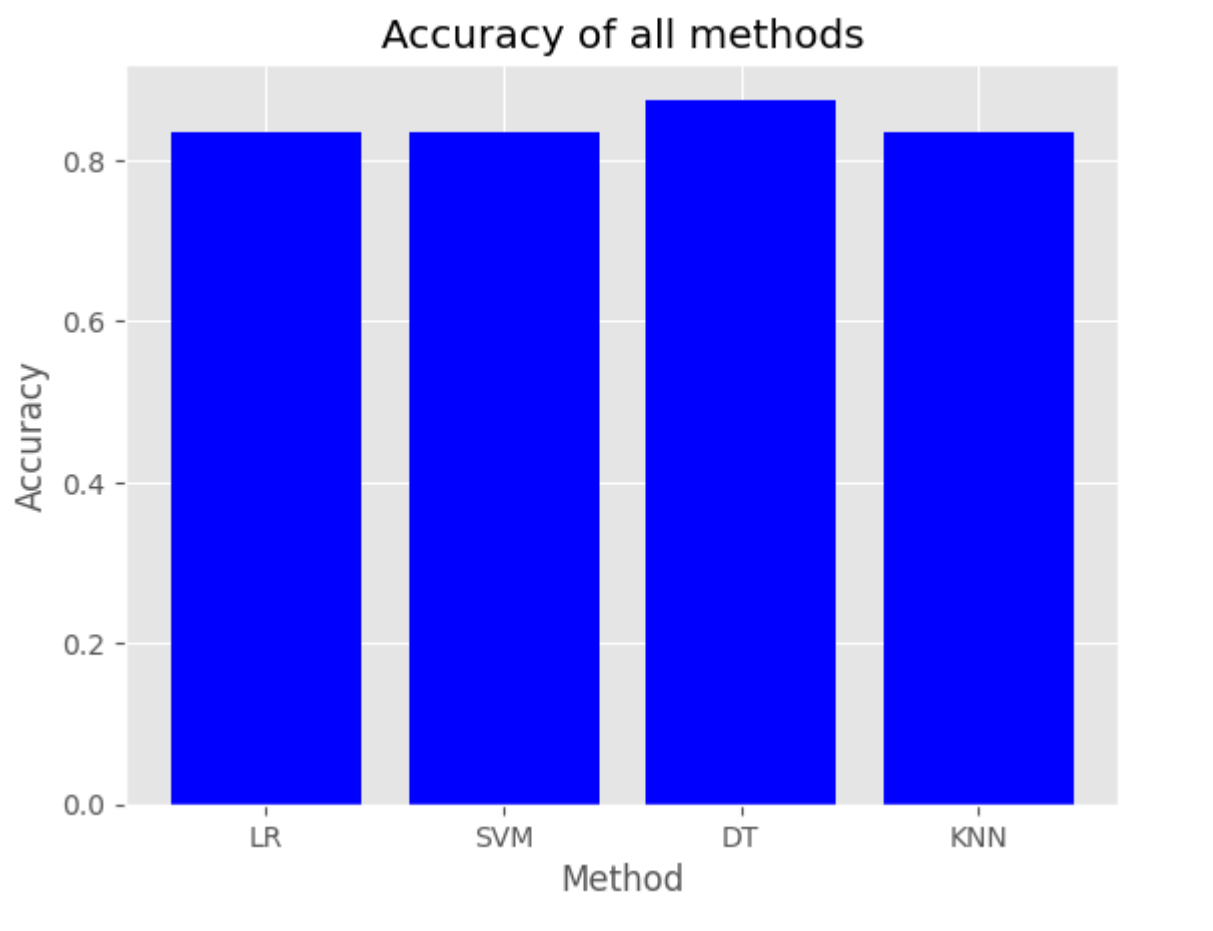
Payload vs Outcome Scatter Plot For All Sites

- From the Scatter Plot we see that for different booster version the highest successful launches occurs between a payload of 2000kg and 5500kg
- The slider was used in the second screenshot to zoom in on these payloads

Section 5

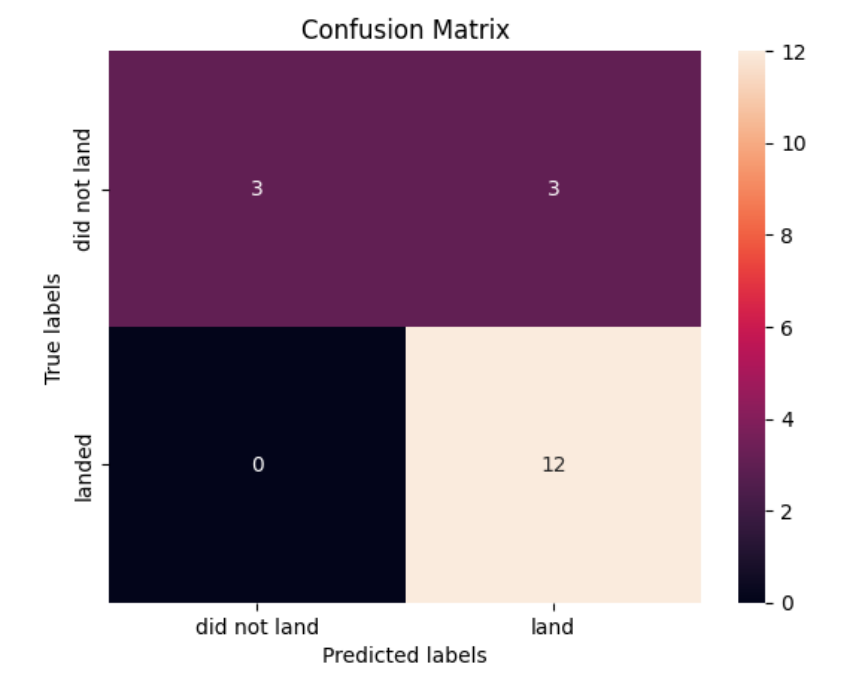
Predictive Analysis (Classification)

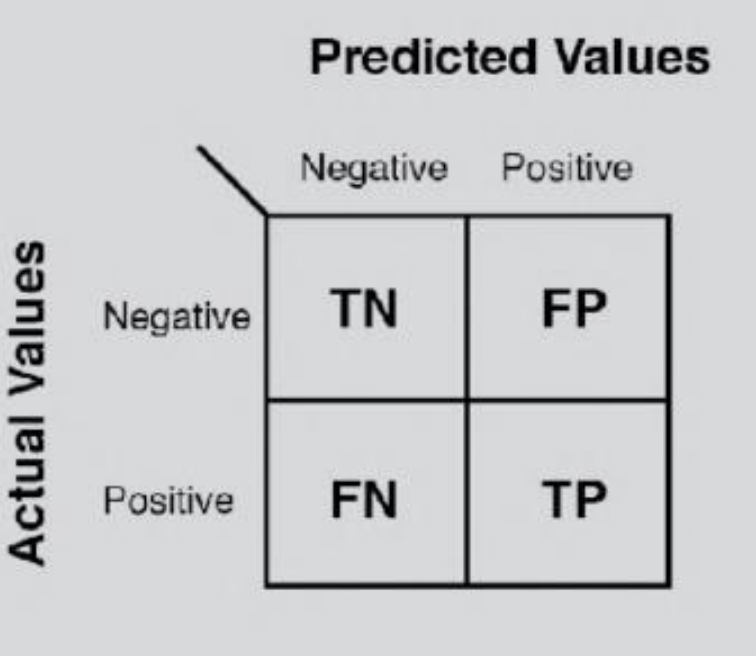
Classification Accuracy



- Based on the Bar chart, we can see that the Decision Tree model performed has the highest classification accuracy which is approximately 90%

Confusion Matrix





- Based on the confusion matrix the decision tree model is able to distinguish between the different classes for true readings the accuracy is (12+3)/18 = 83%. The misclassification rate is 3/18 = 17%. This is a good model

Conclusions

- Orbit types ES-L1, GEO, HEO and SSO had the highest success rates
- From the year 2013, the success rate for SpaceX launches has steadily increased from the previous year with exceptions 2018 and 2020
- KSC LC-39A has the most successful launches of any sites; 76.9%
- Payloads between 2000kg and 5500kg had the highest success rates
- The Decision Tree classifier was the best performing machine learning algorithm for the dataset

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

GitHub URL:

Thank you!