IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Marcia McDonald
12 November 2023

Outline

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

Executive Summary

- **Summary of methodologies**
  - Data Collection through API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Data Visualization
  - Interactive Visual Analysis with Folium
  - Machine learning prediction

- **Summary of all results**
  - Exploratory Data Analysis result
  - Interactive Analysis results
  - Predictive Analysis results in Machine Learning Lab

Introduction

- **Project background and context**

  - SpaceX advertises Falcon 9 rocket launches on its website, with a cost of 62 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 then determine the cost of a launch. This information can be used if an alternative company wants to bid against SpaceX for a rocket launch.
    In this module, you will be provided with an overview of the problem and the tools you need to complete the course.

- **Problems you want to find answers**
  - What factors determine if the rocket will land successfully?
  - The interaction amongst various features that determines the success rate of a successful landing.
  - What operating conditions needs to be in place to ensure a successful landing program.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX API and Web Scraping from Wikipedia.
- Perform data wrangling
  - Missing value replaced by mean values (Payload Mass)
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Analyse outcome of Orbit type
  - Analyse outcome by payload mass and booster version with SQL
  - Visual Analysis with charts by payload mass, time, orbit type and launch site
- Perform interactive visual analytics using Folium and Plotly Dash
  - Visual Analysis with map by site.
  - Interactive Dashboard: Analysis by site, Payload and booster version
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models using Logistic Regression, SVM, Decision Tree, KNN
    - Parameter Tuning with Grid search

Data Collection

- Data collection is the process of gathering and measuring information on targeted variables in an established system, which then enables one to answer relevant questions and evaluate the outcomes. As mentioned, the dataset was collected by REST API and Web scraping from Wikipedia.

- **The data was collected using various methods**

  - Data collection was done using get request to the SpaceX API

  - Next, we decoded the response content as a Json using .json() function call and turn it into a panda dataframe using .json_normalized()

  - We then cleaned the data, checked for missing values and fill in missing values where necessary.

  - In addition, we performed web scraping from Wikipedia for falcon 9 launch records with BeautifulSoup.

  - The objective was to extract the launch records as HTML table, parse the table and convert it into pandas dataframe for future analysis.

Data Collection – SpaceX API

Get request for rocket launch data using API

Use json_normalize method to convert json result to dataframe

Performed data cleansing and filling the missing value

From:

https://github.com/Kellis47/Project-X/blob/main/1.1%20Data-collection-API.ipynb

1.

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/da
```

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

```
response.status_code
```

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

2.

```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())  # convert to flat table
```

3.

```
# 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 h
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 featur
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)]
```

Data Collection - Scraping

Request the Falcon 9 Launch Wiki page from URL

Create a BeautifulSoup from the HTML response

Extract all columns / variable names from the HTML response

- **From:** https://github.com/Kellis47/Project-X/blob/main/2.%20Data%20Web%20Scraping.ipynb

```python
# use requests.get() method with the provided static_url
# assign the response to a object
r = requests.get(static_url)
data = r.text
```

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

```python
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
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            #print(flight_number)
            datatimelist=date_time(row[0])

            # Date value
            # TODO: Append the date into launch_dict with key `Date`
            date = datatimelist[0].strip(',')
            #print(date)

            # Time value
            # TODO: Append the time into launch_dict with key `Time`
            time = datatimelist[1]
            #print(time)

            # Booster version
```

Data Wrangling

- Data wrangling is the process of removing errors and combining complex data sets to make them more accessible and easier to access and to conduct exploratory Data Analysis (EDA).

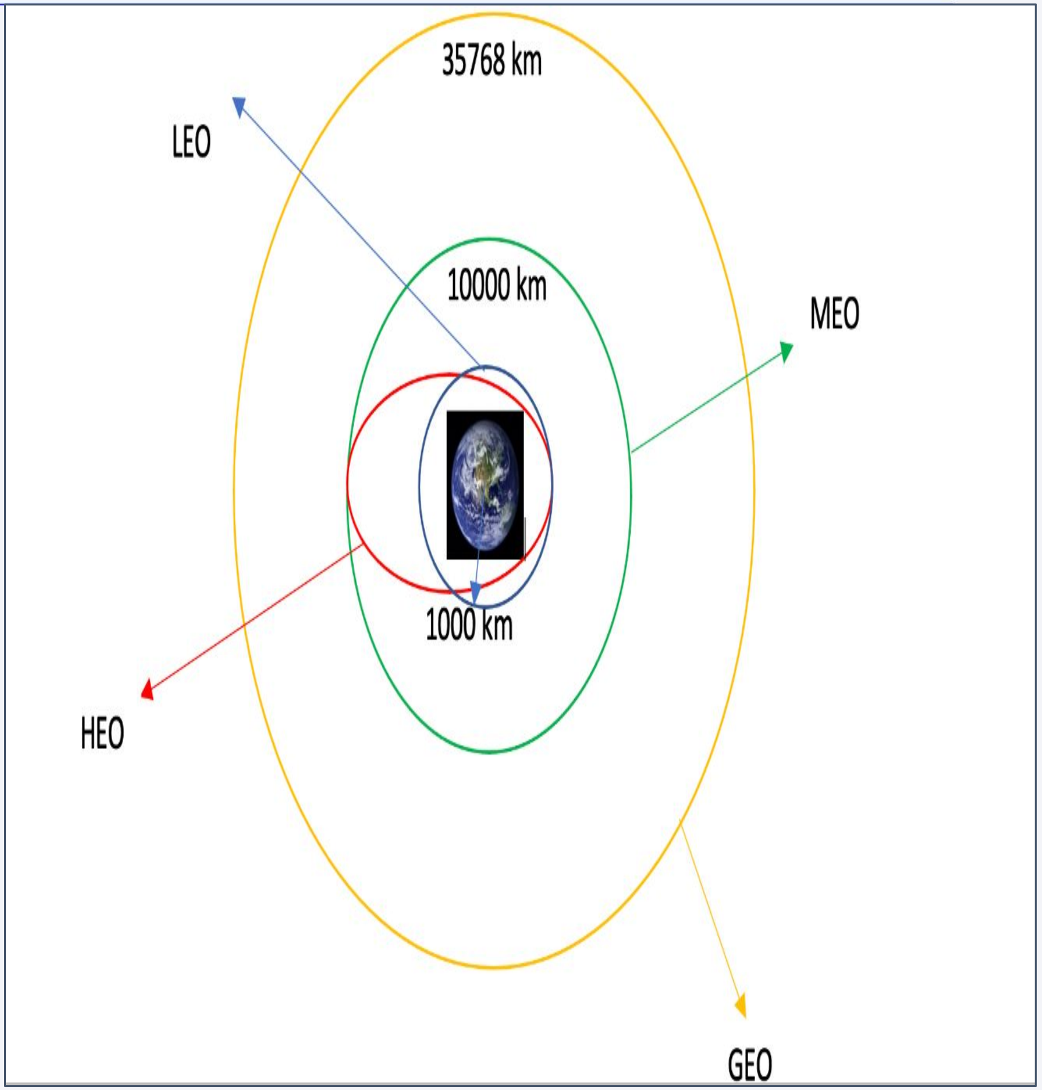
  - How is this calculated?

    We firstly calculate the number of launches on each site, then calculate the number
    and occurrence of the mission outcome per orbit type.

  - We then proceed to create a landing outcome label from the outcome column. This will make it easier for further analysis, visualization, and ML. Finally, we will then export the result to a CSV

  -

From:
https://github.com/Kellis47/Project-X/blob/main/3.%20IBM-Labs%20module%20-Data%20Wrangling%20Jupyter-spacex.ipynb

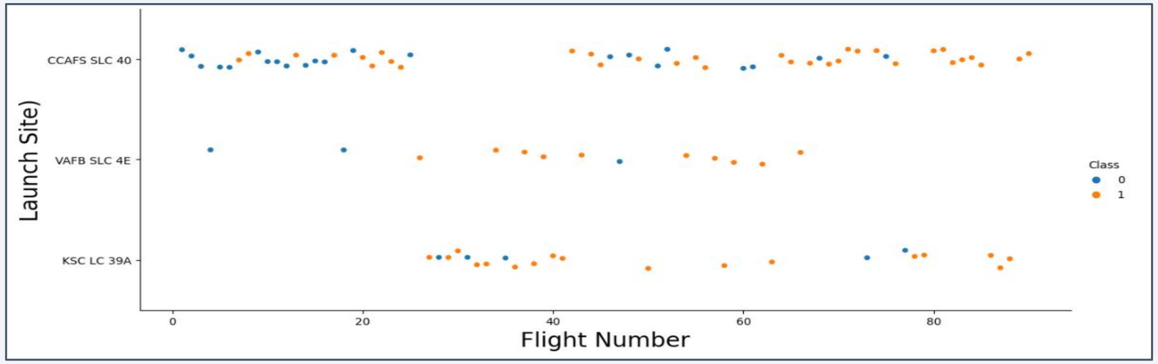


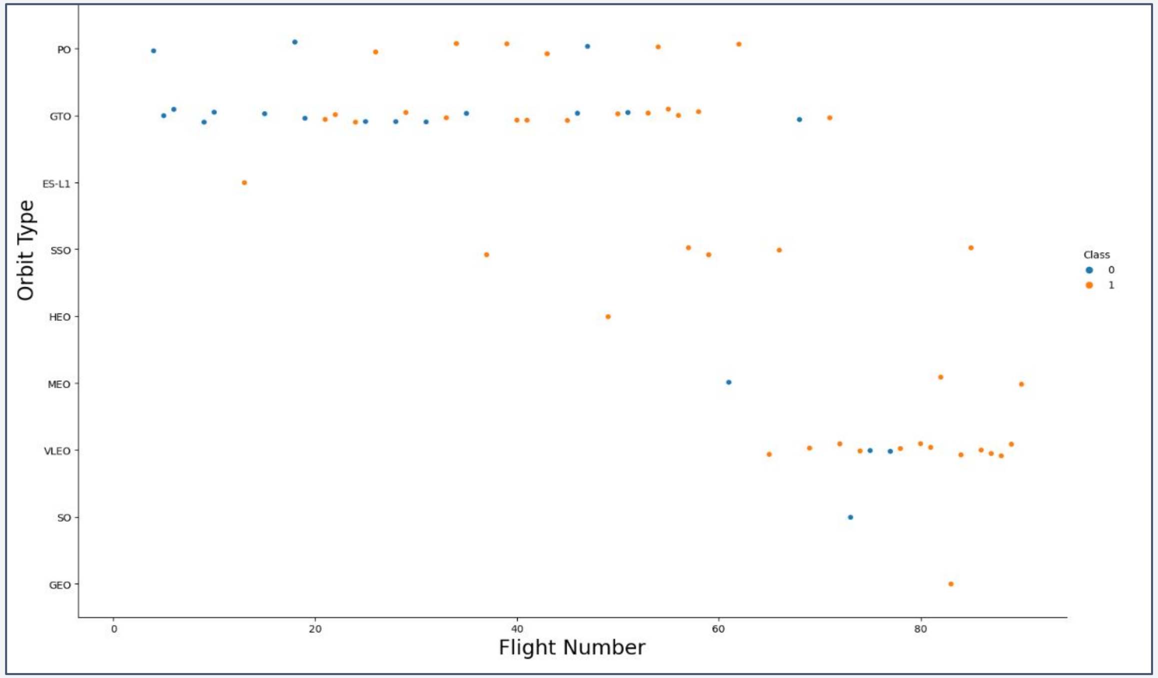
EDA with Data Visualization

- We firstly started by using scatter graphs to find the relationship between the following attributes:
- Payload and flight number
- Flight number and launch site
- Flight number and Orbit Type
- Payload and Orbit Type

Scatter plots are able to show the dependency of the attributes on each other.
Once a pattern is determined from the graphs. It's easy to see which factors are affected the most to the success of the landing outcome.

From:
https://github.com/Kellis47/Project-X/blob/main/5.%20EDA%20with%20Visualisation.ipynb



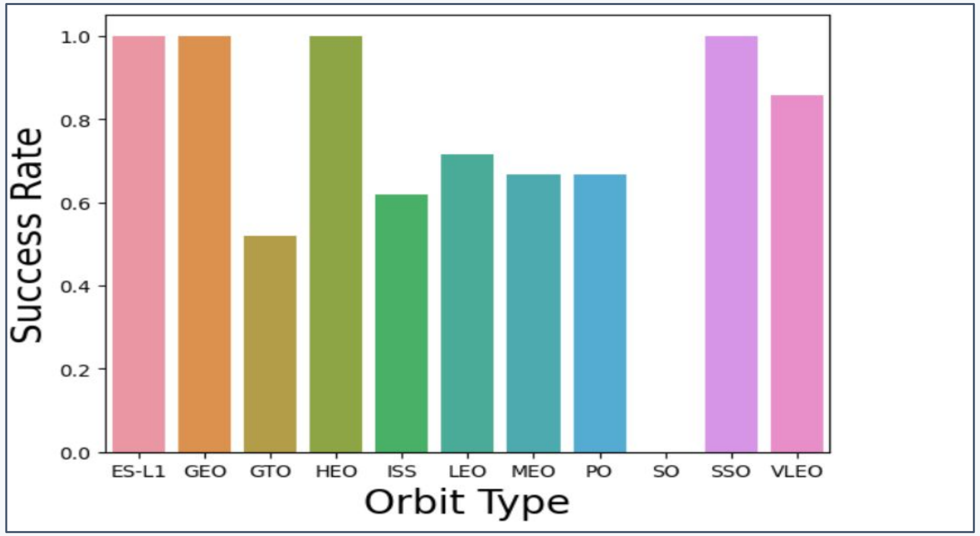


EDA with Data Visualization

Once we have identified the relationships using a Scatter plot.

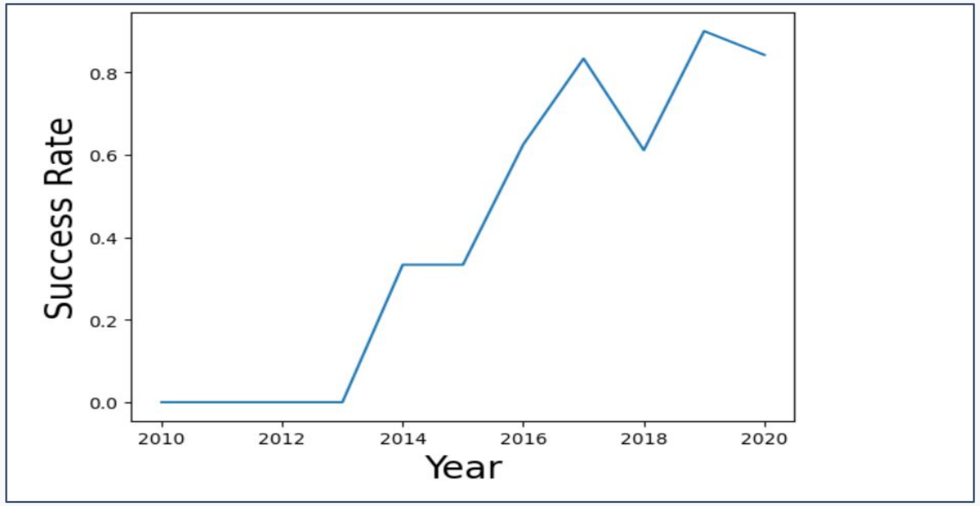
We will then use further visualization tools such as Bar graphs and plot graph for further analysis.

Using Bar graphs is one of the easiest ways to interpret the relationships between the attributes. In this case, we will used bar graph to determine which orbits have the highest probability of success.



We will then, used the line graph to show trends or patterns of the attributes over time, which in this case, is used to identify the launch success yearly trend.

We then used Feature Engineering, to be used in success prediction in the future module, by creating dummy variables for categorical columns.



From:

https://github.com/Kellis47/Project-X/blob/main/5.%20EDA%20with%20Visualisation.ipynb

EDA with SQL

Using SQL, we performed many queries to get a better understanding of the dataset, EX:

- Displaying the names of the launch sites.
- Displaying 5 records of launch sites begins with string ‘CCA’.
- Displaying total payload mass carried by booster launched by NASA (CRS).
- Displaying the average payload mass carried by booster version F9 v 1.1
- Listing the date when the first successful landing outcome in ground pad was achieved.
- Listing the names of boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000.
- Listing the 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 in year 2015.
- Rank the count of landing outcomes or success between the date 04.06.2010 and 20.03.2017, in descending order.

From:

https://github.com/Kellis47/Project-X/blob/main/8.%20Machine%20Learning%20Prediction.ipynb

https://github.com/Kellis47/Project-X/blob/main/4.%20EDA%20with%20SQL%20Notebook%20for%20Peer%20Assignment.ipynb

Build an Interactive Map with Folium

In order to visualise the launch data in an interactive map. We took the latitude and longitude coordinates of each launch site and added a circle marker around each launch site with a label of the name of the launch site.

We then assigned the dataframe launch_outcomes (failure, Success) to classes 0 and 1 with Red and Green markers on the map in MarkerCluster().

We then used the Haversine formula to calculate the distance of the launch sites to various landmarks to find answers to the questions:

- How close are the launch sites with railways, highways and coastline?
- How close are launch sites to nearby cities?

- https://github.com/Kellis47/Project-X/blob/main/6.%20Interactive%20Visual%20Analytics%20with%20Folium.ipynb
- https://github.com/Kellis47/Project-X/blob/main/6.%20Module%203%20%20Interactive%20Visual%20Analytics%20with%20Folium%20(1).ipynb

Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash, which allows the user to play around with the data as and when they need to.
- We then plotted the results onto Pie charts, showing the total launches by conducted at certain sites.
- We then proceeded to plot the results on a scatter graph, which depicted the relationships between the outcome and Payload Mass (Kg) for the different booster version.

https://github.com/Kellis47/Project-X/blob/main/Dash%20App%20.py.pdf

Predictive Analysis (Classification)

1. **Build the Model**

. **Load the dataset into Numpy and Panda.**

. **Transform the data and then split it into training datasets**

. **Decide which type of ML to use.**

. **Set the parameters and algorithms to Grid search CV and fit it to dataset.**

. **Decide which type of ML to use**

. **Set the parameters and algorithms to Gridsearch cv and fit to dataset.**

2. **Evaluating the Model**

. Check the accuracy for each model.

. Get tuned hyperparameters for each type of algorithms.

. Plot the confusion matrix.

3. **Improving the model**

. Use Feature Engineering and Algorithm Tuning

4. **Find the Best Model**

. The model with the best accuracy score will be the best performing model.

From:
https://github.com/Kellis47/Project-X/blob/main/Dash%20App%20.py.pdf

Results

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

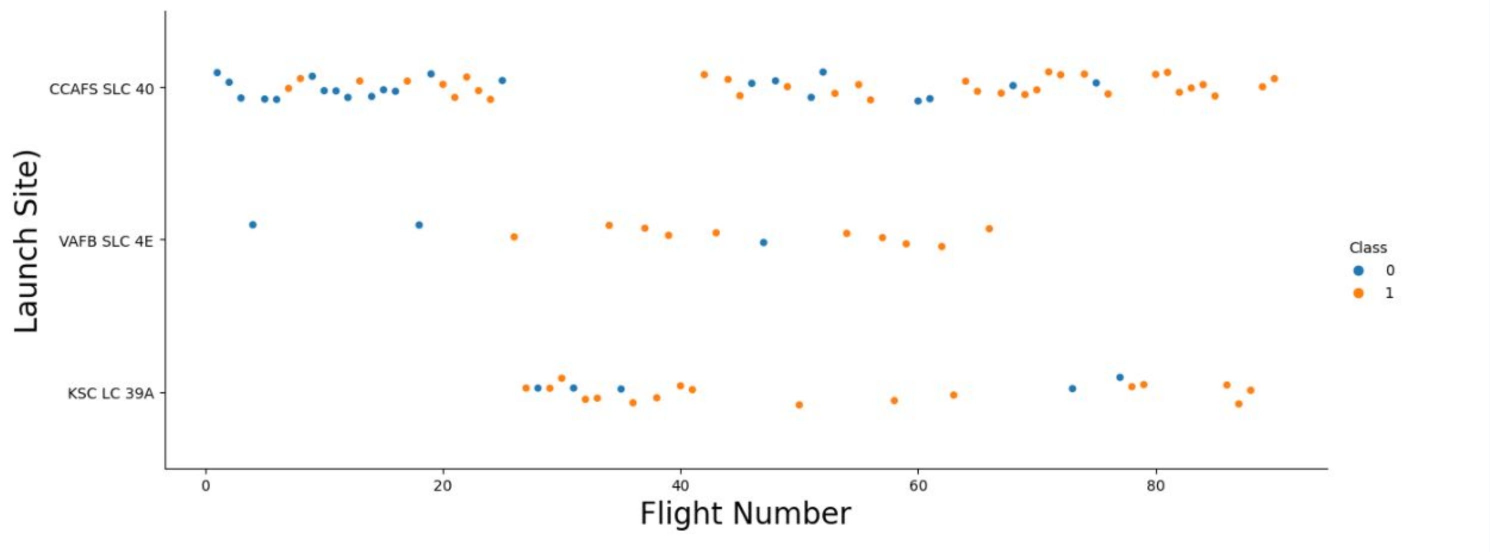
Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

This scatter plot shows that the larger the flight amounts of launch sites, the greater the success rate will be.

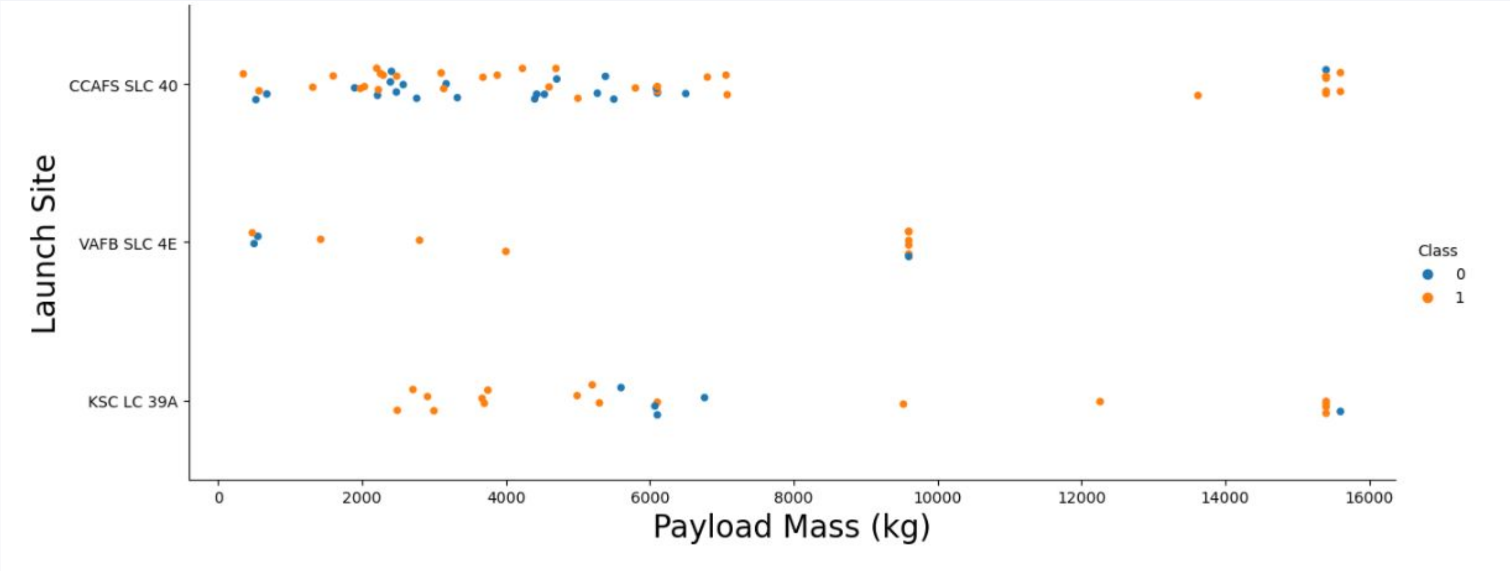
However, site CCAFS SLC40 shows the least pattern of this.



Payload vs. Launch Site

The scatter plot shows once the payload mass is greater than 7000kg, the probability of the success rate will be highly increased.

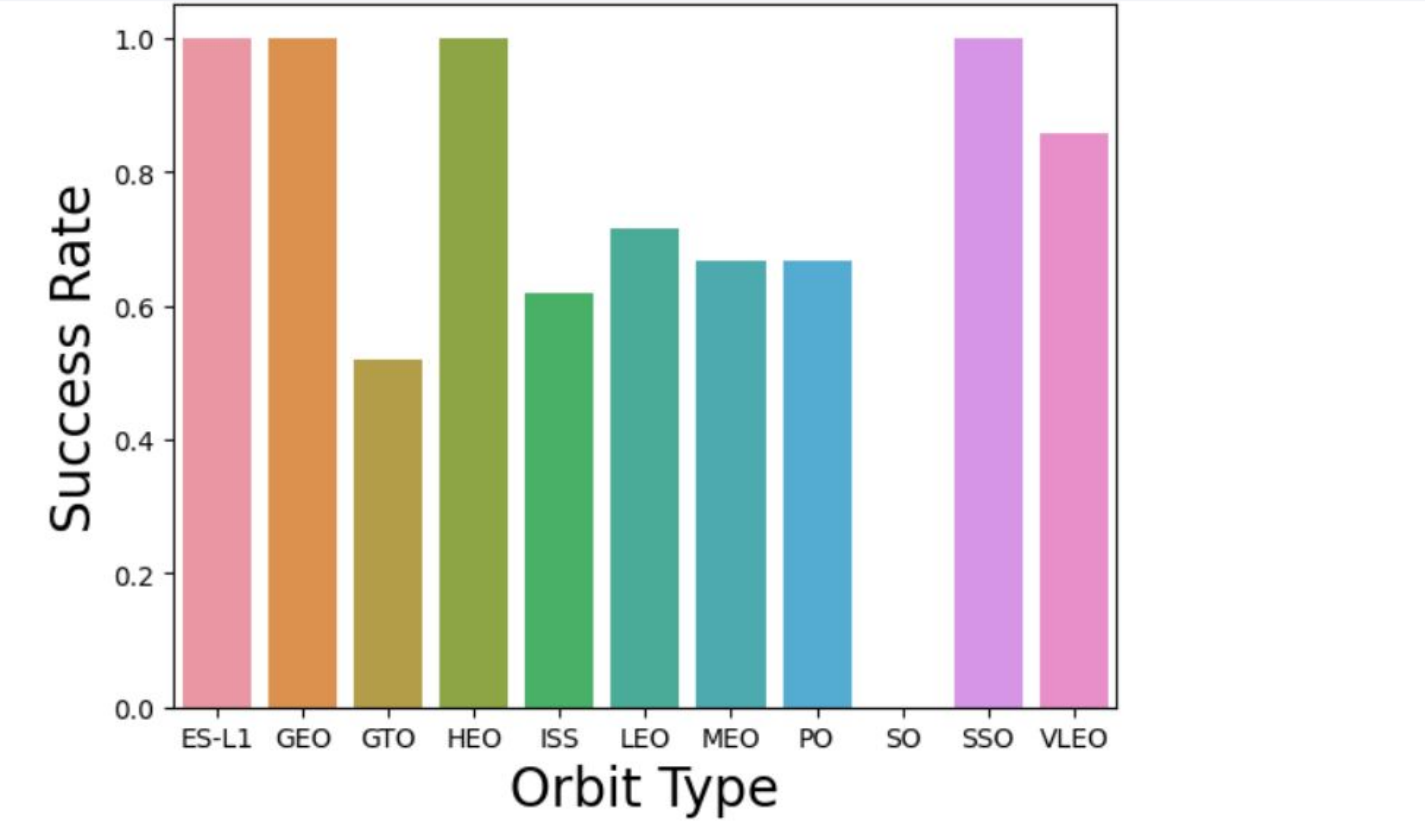
However, there is no clear pattern to say that the launch site is dependent to the payload mass for the success rate.



Success Rate vs. Orbit Type

The Bar Chart depicts the possibility of the Orbits to influence the landing outcomes as some Orbits has 100% success rate such as SSO, HEO, GEO, and ES-LI while SO Orbit produced 0% rate of success.

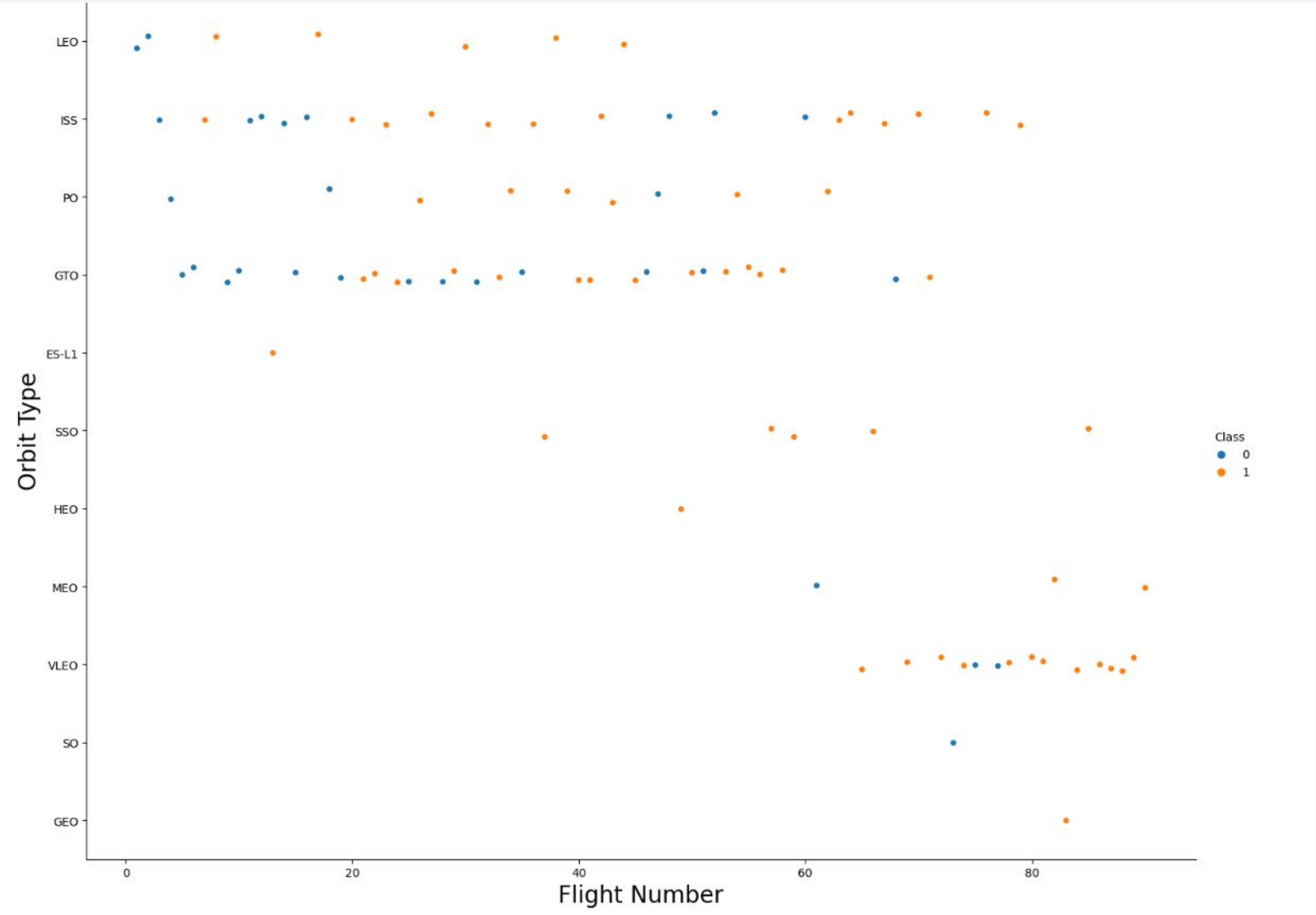
However, under deeper analysis it shows that some of these Orbits has only 1 occurrence such as GEO,SO,HEO and ES-L1 which means the data needs more datasets in order to determine a pattern or trend before we can draw any conclusion.



Flight Number vs. Orbit Type

The scatter plot displays in general terms that the larger the flight number
on each Orbits, the greater the success rate (especially LEO Orbit), except for GTO Orbit which depicts no relationship between both attributes.

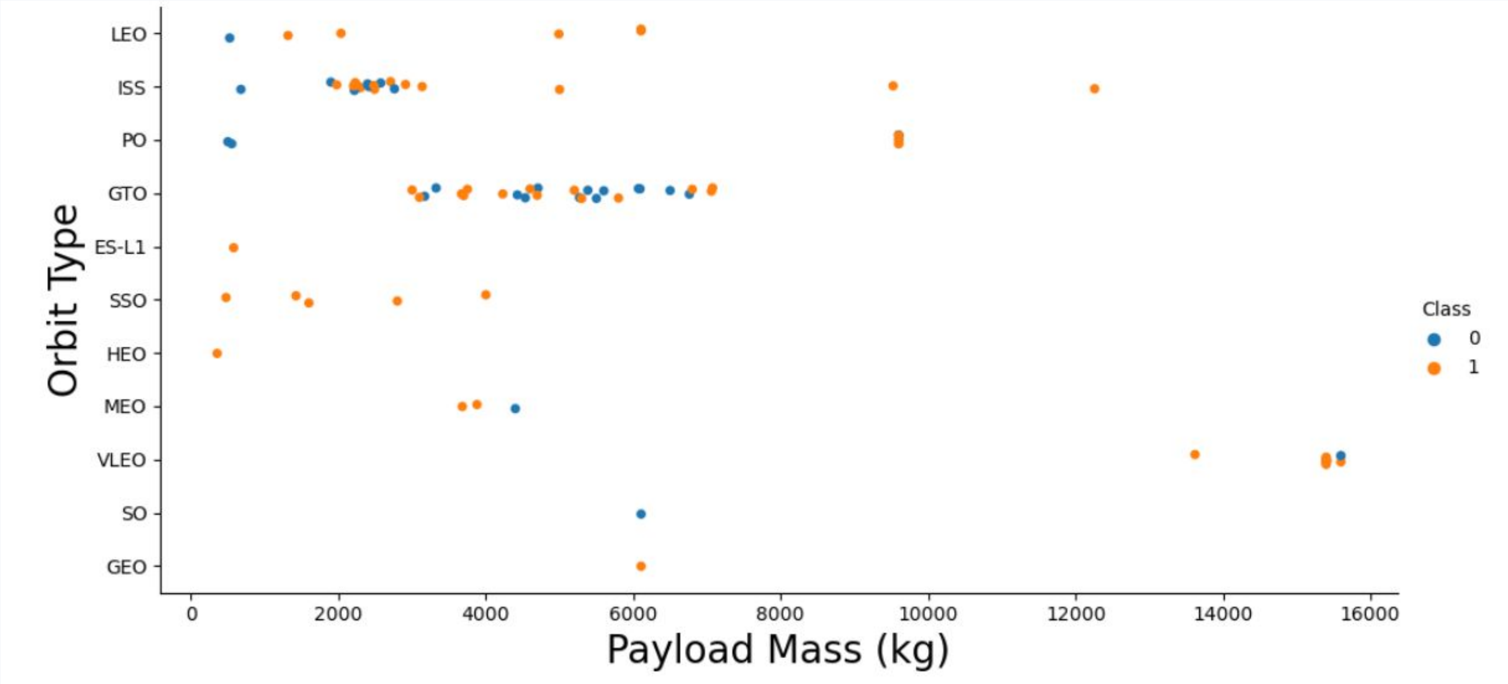
An Orbit that only has 1 occurrence should also be excluded from above statement as it requires more dataset.



Payload vs. Orbit Type

- Show a scatter point of payload vs. orbit type

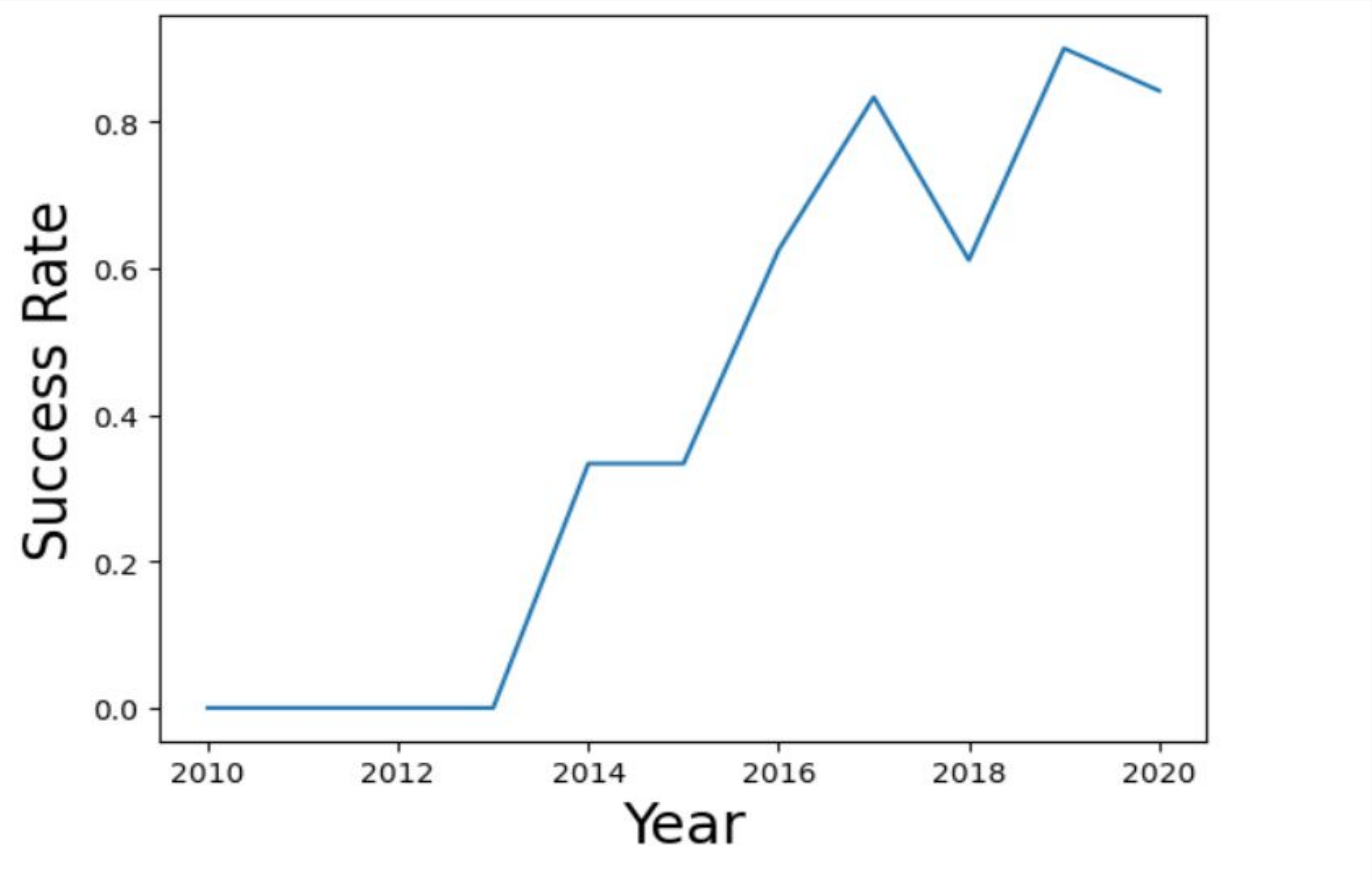
- Show the screenshot of the scatter plot with explanations



Launch Success Yearly Trend

The line chart identifies a clear increasing trend from the year 2013 until 2020.

We can predict, if this trend continue for the next year onwards, the success rate will steadily increase until reaching 1/100 % success rate.



All Launch Site Names

We used the keyword ‘Distinct’ to extract only the unique launch sites from the SpaceX data.

https://github.com/Kellis47/Project-X/blob/main/4.%20EDA%20with%20SQL%20Notebook%20for%20Peer%20Assignment.ipynb

```
In [10]: # Task 1 Display the names of the unique launch sites in the space mission#
         q = pd.read_sql('select distinct Launch_Site from spacexdata', conn)
         q
```

Out[10]:

	Launch_Site
0	CCAFS LC-40
1	VAFB SLC-4E
2	KSC LC-39A
3	CCAFS SLC-40
4	None

Launch Site Names Begin with 'KSC'

- Find 5 records where launch sites' names start with `KSC`

```
q = pd.read_sql("select * from spacexdata where Launch_Site like 'KSC%' limit 5", conn)
q
```

Out[12]:

	index	Date	Time_(UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	La
0	29	None	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490.0	LEO (ISS)	NASA (CRS)	Success	
1	30	None	6:00:00	F9 FT B1030	KSC LC-39A	EchoStar 23	5600.0	GTO	EchoStar	Success	
2	31	None	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300.0	GTO	SES	Success	
3	32	None	11:15:00	F9 FT B1032.1	KSC LC-39A	NROL-76	5300.0	LEO	NRO	Success	
4	33	None	23:21:00	F9 FT B1034	KSC LC-39A	Inmarsat-5 F4	6070.0	GTO	Inmarsat	Success	

Total Payload Mass

Base on our calculations, we found the total payload mass carried by booster from NASA as 45596 using the following query below.

```
q = pd.read_sql("select sum(PAYLOAD_MASS__KG_) from spacexdata where Customer='NASA (CRS)'", conn)
q
```

Out[13]:

	sum(PAYLOAD_MASS__KG_)
0	45596.0

https://github.com/Kellis47/Project-X/blob/main/4.%20EDA%20with%20SQL%20Notebook%20for%20Peer%20Assignment.ipynb

Average Payload Mass by F9 v1.1

- We calculated the average payload mass carried by booster version F9 v1.1 as 2928.4

https://github.com/Kellis47/Project-X/blob/main/4.%20EDA%20with%20SQL%20Notebook%20for%20Peer%20Assignment.ipynb

In [14]:
```
# Task 4 Display average payload mass carried by booster version F9 v1.1¶#
q = pd.read_sql("select avg(PAYLOAD_MASS__KG_) from spacexdata where Booster_Version='F9 v1.1'", conn)
q
```

Out[14]:

	avg(PAYLOAD_MASS__KG_)
0	2928.4

First Successful Ground Landing Date

- We used the Min() to locate the result.
- The results concluded there were no successful landing outcomes achieved in ‘drone ship’.

https://github.com/Kellis47/Project-X/blob/main/4.%20EDA%20with%20SQL%20Notebook%20for%20Peer%20Assignment.ipynb

```
[16]: #Task 5 List the date where the succesful landing outcome in drone ship was acheived.#
      q = pd.read_sql("select min(Date) from spacexdata where Landing_Outcome='Success (drone ship)'", conn)
      q
```

[16]:

	min(Date)
0	None

Successful Drone Ship Landing with Payload between 4000 and 6000

We used the **'WHERE'** clause to filter the search for boosters which have successfully landed on the drone ship and applied the '**AND**' condition to determine the successful landing with payload mass greater than 4000 but less than 6000.

```
%sql SELECT BOOSTER_VERSION FROM SPACEX WHERE LANDING__OUTCOME = 'Success (drone ship)' \
AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

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

Total Number of Successful and Failure Mission Outcomes

We calculated the total number of successful and failure mission outcomes by using count(*) function. Which provided the following results:

[22]:
```
# Task 7 List the total number of successful and failure mission outcomes#

q = pd.read_sql("select substr(Mission_Outcome,1,7) as Mission_Outcome, count(*) from spacexdata  group by 1", conn)
q
```

[22]:

	Mission_Outcome	count(*)
0	None	898
1	Failure	1
2	Success	100

Boosters Carried Maximum Payload

- We used the **'WHERE'** clause to filter to search for boosters_ versions which have carried out the maximum payload mass then applied the '**MAX**' condition to determine the maximum payload mass.
  The names of the booster_version which have carried the maximum payload mass are as follows:

In [21]:
```
#Task 8 List the names of the booster_versions which have carried the maximum payload mass. Use a subquery#

q = pd.read_sql("select distinct Booster_Version from spacexdata where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MA
q
```

Out[21]:

	Booster_Version
0	F9 B5 B1048.4
1	F9 B5 B1049.4
2	F9 B5 B1051.3
3	F9 B5 B1056.4
4	F9 B5 B1048.5
5	F9 B5 B1051.4
6	F9 B5 B1049.5
7	F9 B5 B1060.2
8	F9 B5 B1058.3
9	F9 B5 B1051.6
10	F9 B5 B1060.3
11	F9 B5 B1049.7

2015 Launch Records

List of failed landing outcomes in drone ship, their booster_versions, and launch site names for the year 2015.

Landing_Outcome	Booster_Version	Launch_Site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1017	VAFB SLC-4E
Failure (drone ship)	F9 FT B1020	CCAFS LC-40
Failure (drone ship)	F9 FT B1024	CCAFS LC-40

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

- Based on our calculation we ranked 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 by using the **'Where'** function, as well as using the **'And'** condition

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)
```

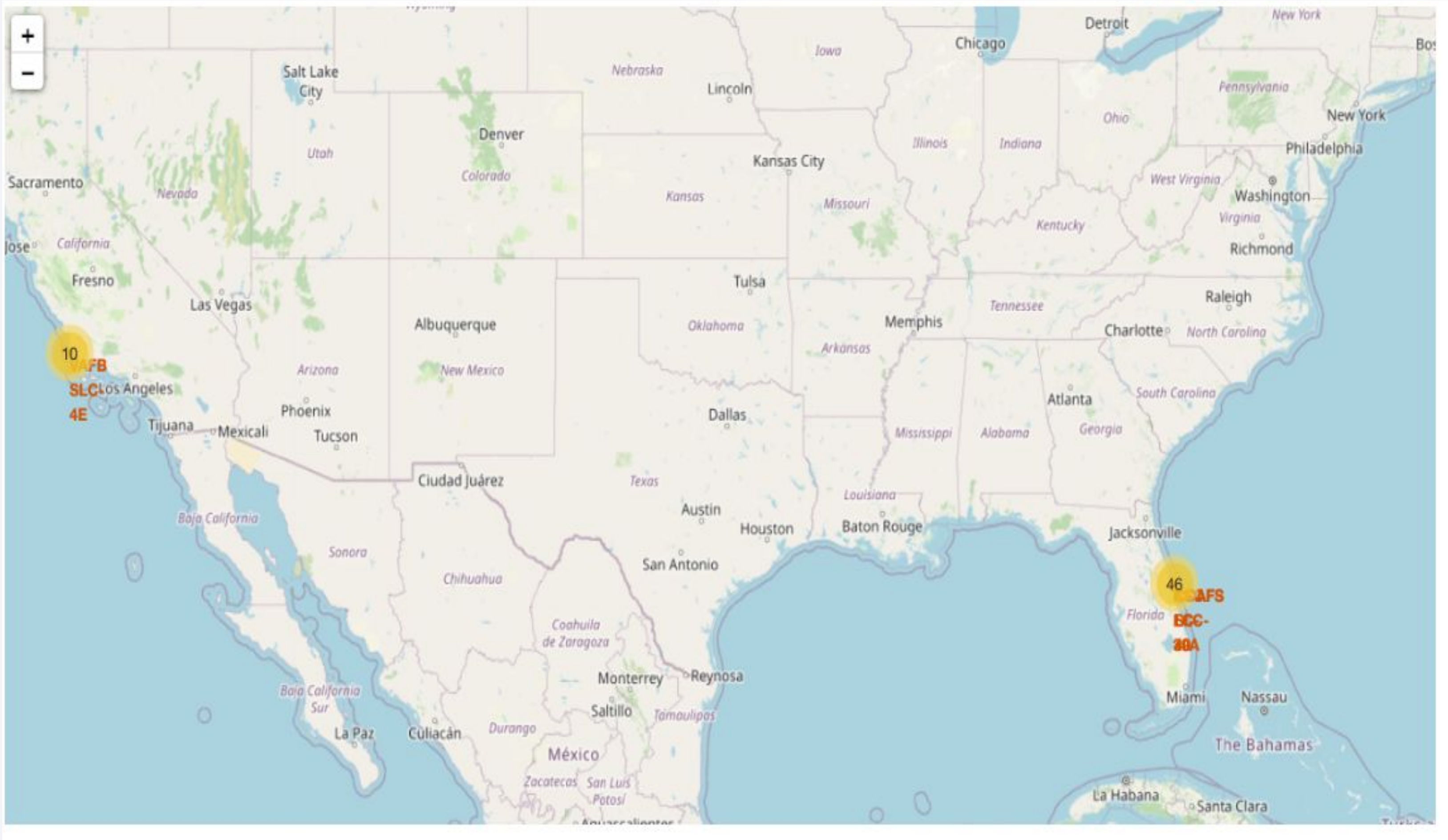
Out[19]:

	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

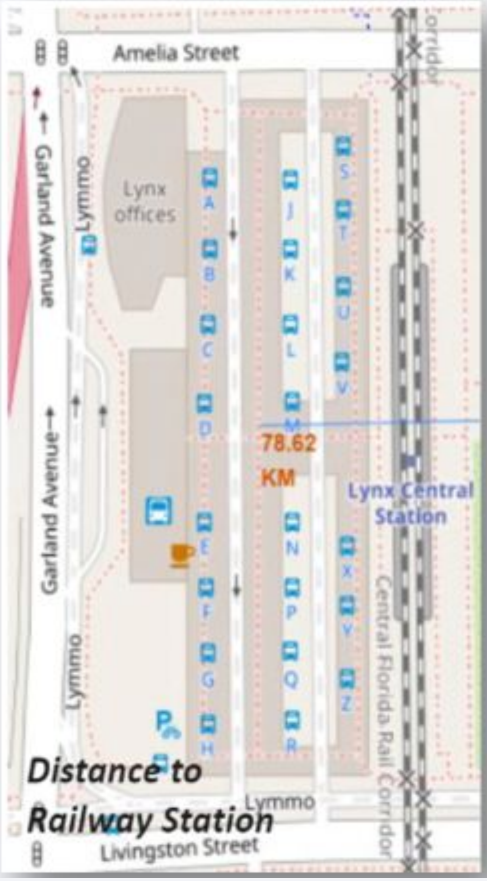
Location of all the Launch Sites

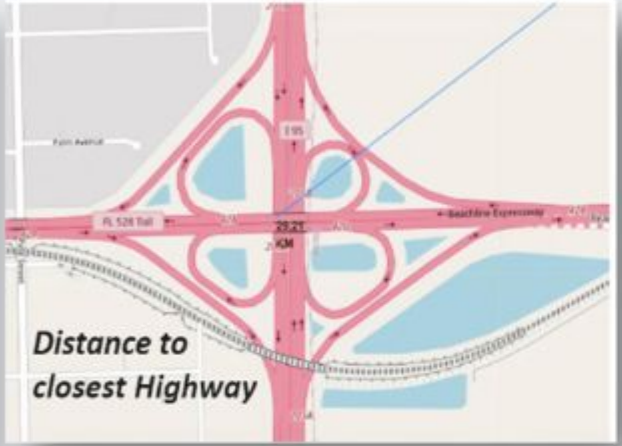


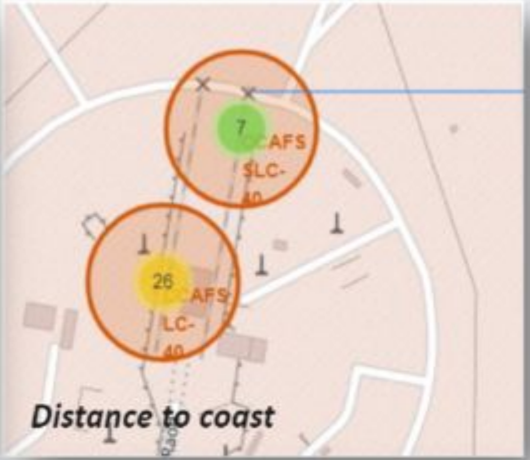
- We can see that all the SpaceX launch sites are located in side the
  United States.

Markers showing launch sites with colour labels

Launch Sites Distances to landmarks





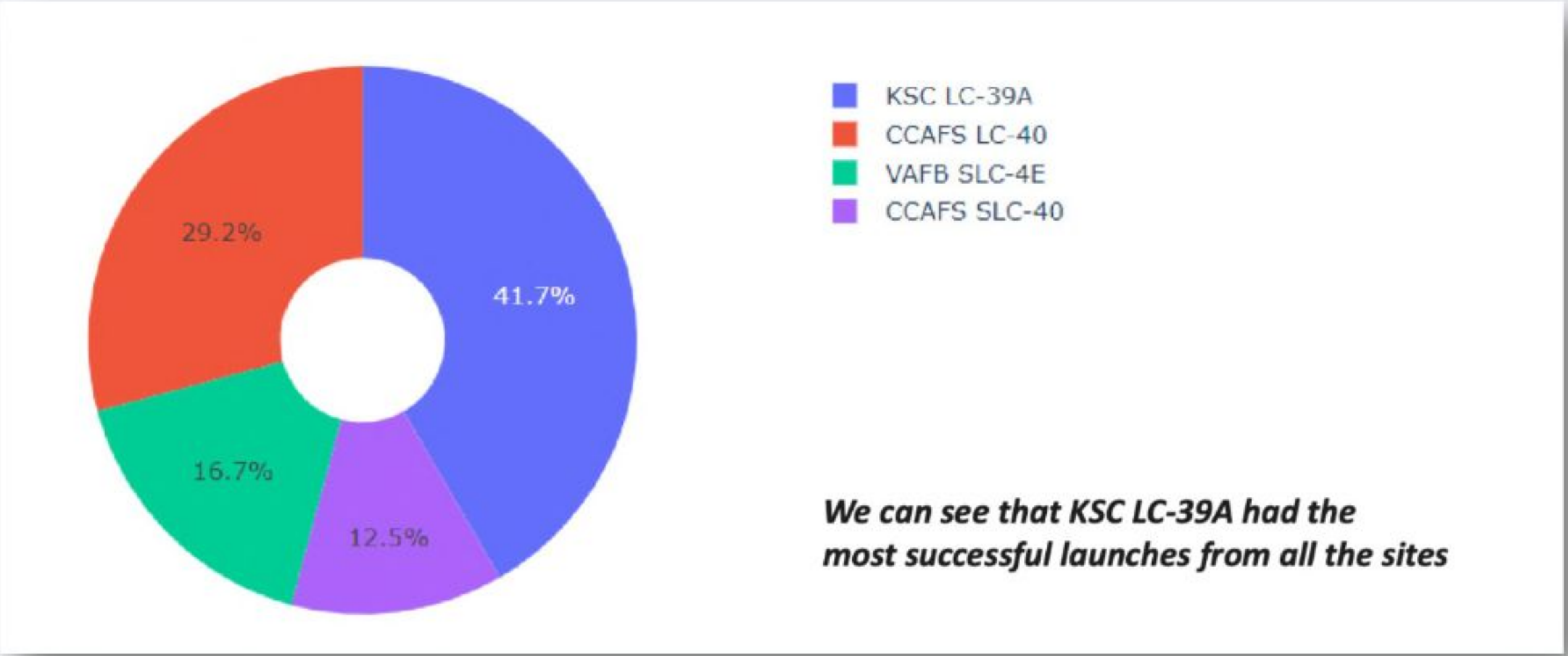


- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

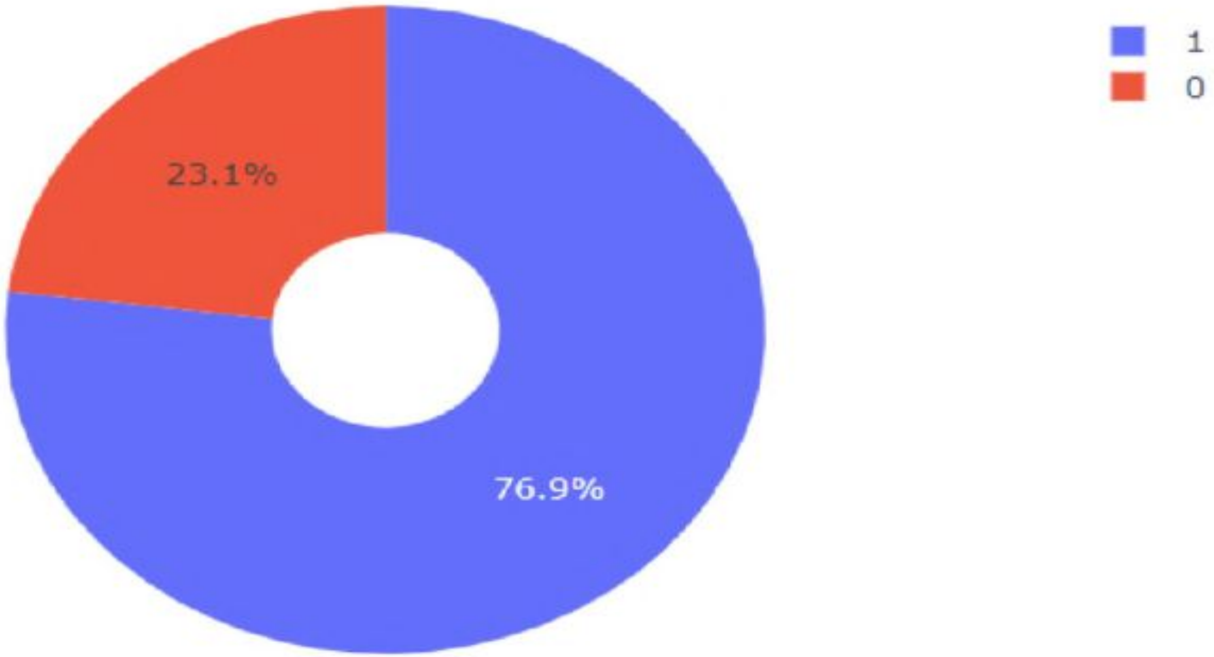
Section 4

Build a Dashboard with Plotly Dash

The success rate by percentage for each site

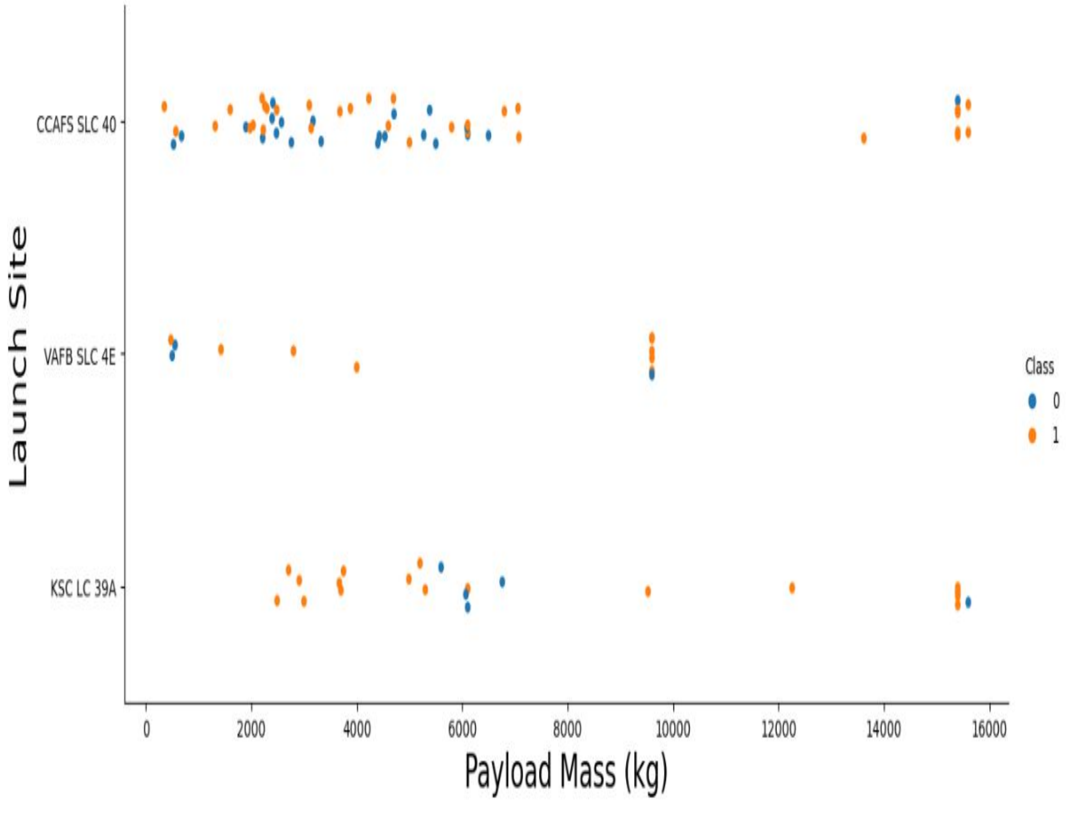


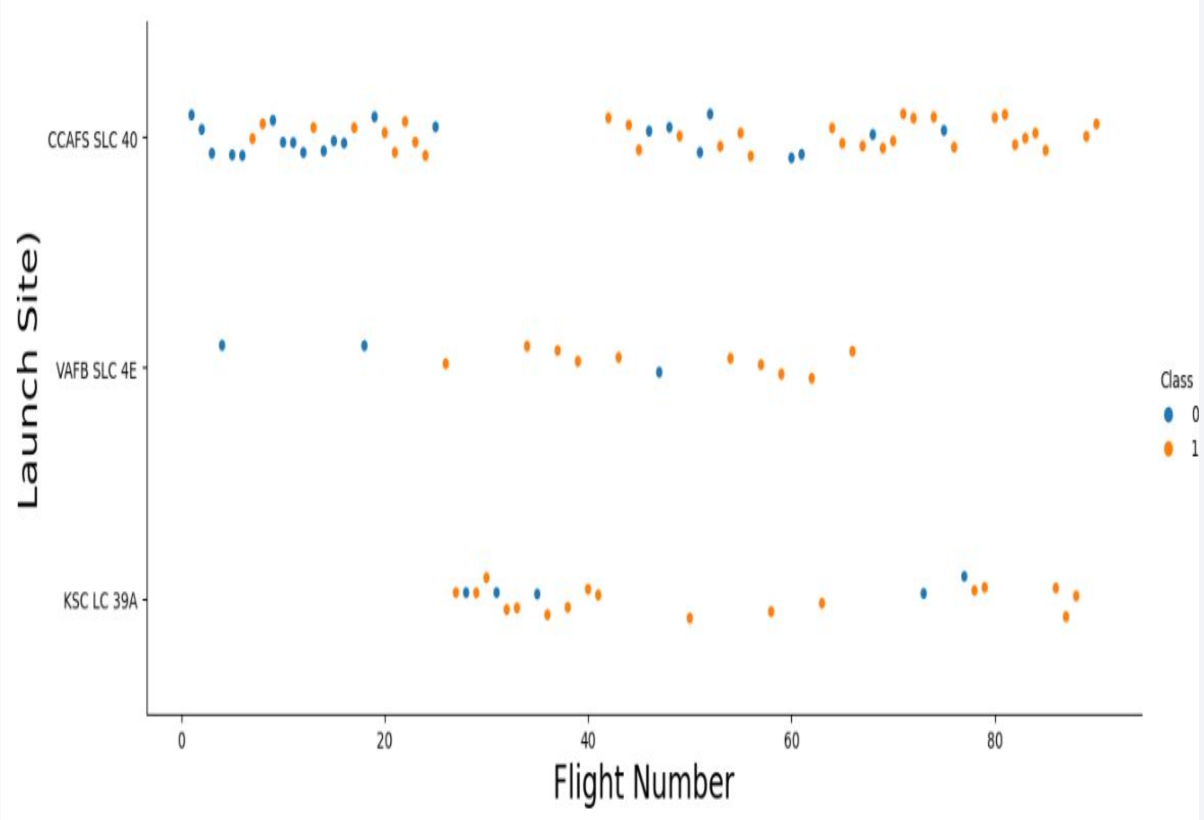
The highest launch ratio: KSC LC -39A



KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

Payload vs Launch Outcome Scatter Plot





Section 5

Predictive Analysis (Classification)

Classification Accuracy

As you can see, by using the code below, we could identify that the best algorithm to be used would be the Tree Algorithm, which have the highest classification accuracy.

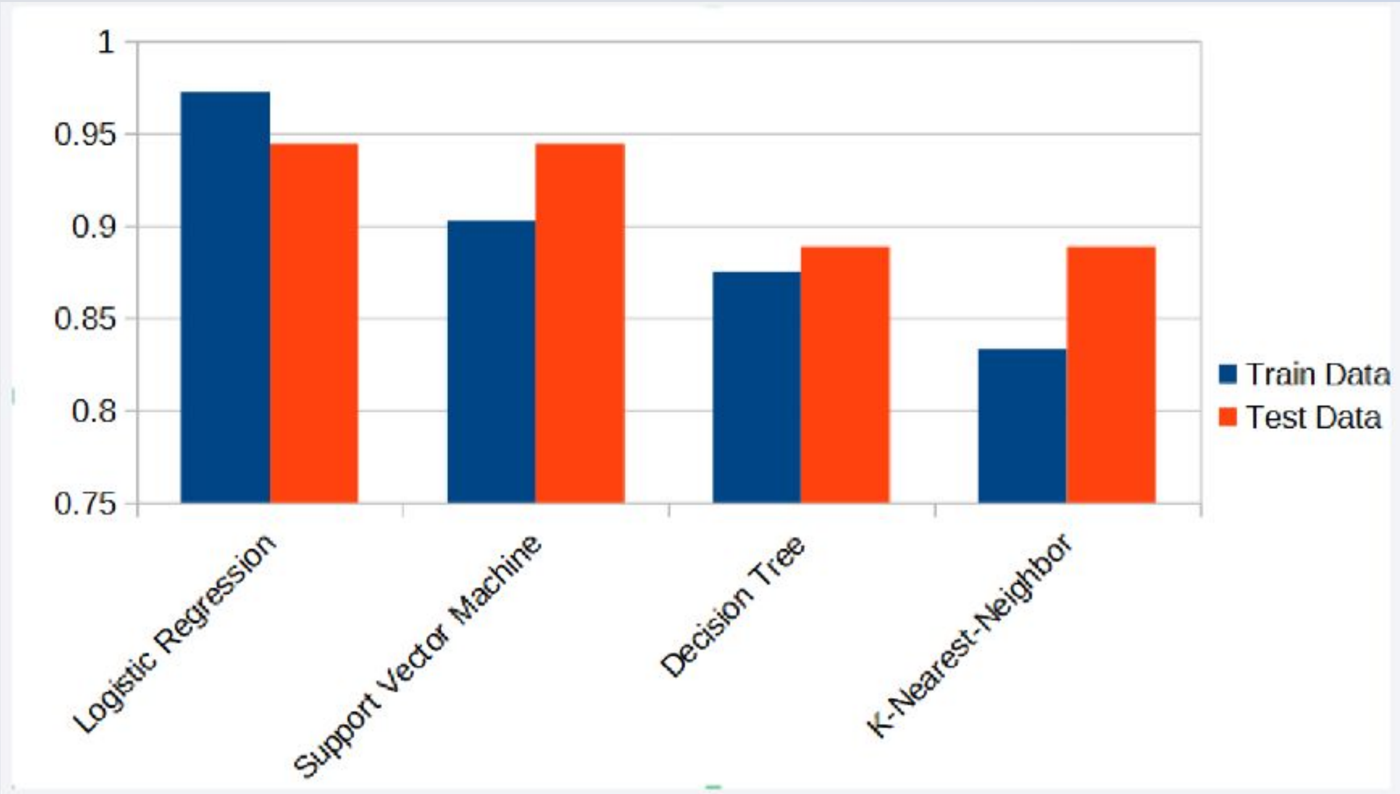
```
#Task 10#

parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
              'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
              'p': [1,2]}

KNN = KNeighborsClassifier()

knn_cv = GridSearchCV(KNN, parameters, cv=10)
knn_cv.fit(X, y)
knn_cv.best_estimator_

print("tuned hpyerparameters :(best parameters) ", knn_cv.best_params_)
print("accuracy :", knn_cv.best_score_)
```

```
#Task 12#

Scores on test data for each method

Logistic Regression: 0.944
SVM: 0.944
Decision Tree: 0.888
KNN: 0.888
Conclusion: Logistic Regression and SVM deliver the best performance on test data.
```

Classification Accuracy

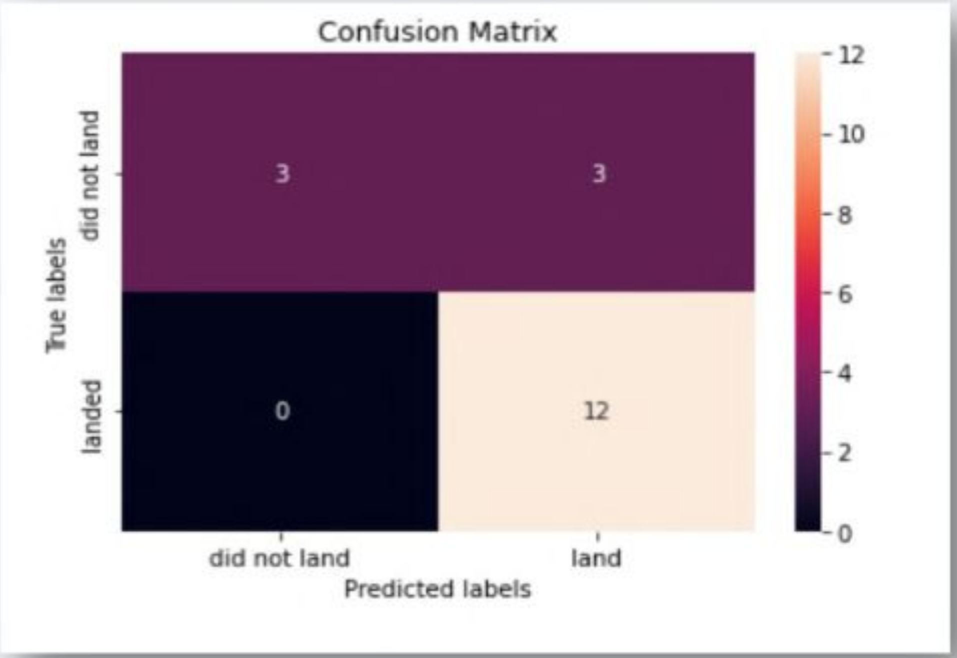
- Logistic Regression has the best result for train data

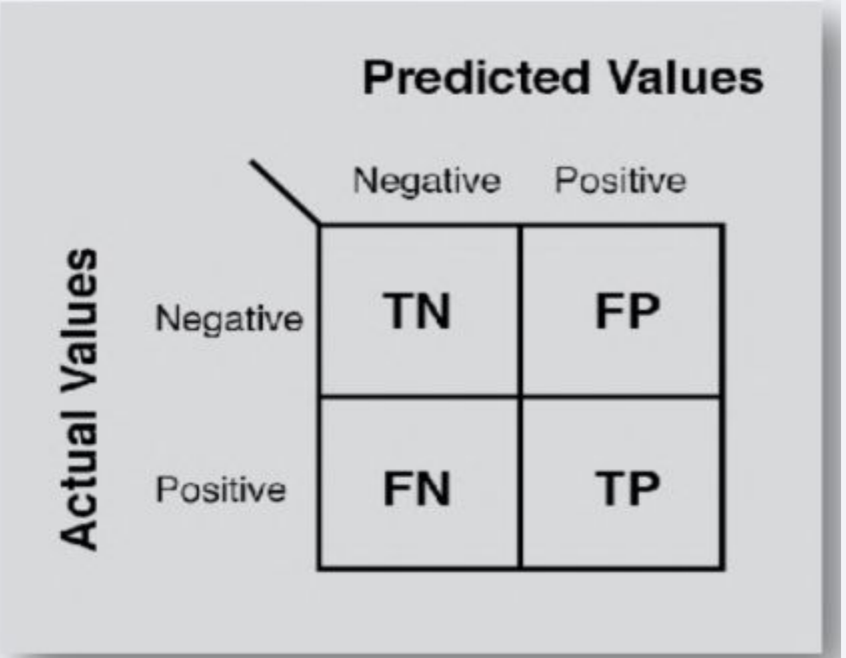
- Logistic Regression and Support Vector Machine have the best rests on test data



Confusion Matrix

The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The major problem is the false positives i.e…unsuccessful landing marked as successful landing by the classifier.





Conclusions

Based on the research conducted, we can conclude the following:

- The tree Classifier Algorithms is the best Machine Learning approach for this dataset
- The lower weighted payloads (which were 4000 kg and below) performed better than the higher weighted payloads.
- Starting from the year 2013, we can see the success rate for SpaceX has increased in direct proportional time in years to 2020, which it will eventually perfect the launch outcomes in the future.
- KSC -LC-39A has had the most successful launches of any of the sites; at 76.9%
- SSO Orbit has had the most success rate; at 100% and more than one occurrence.

Appendix

- All Python code and SQL can be analysed from the following:
  - [Jupyter Notebook]
  - [Plotly Dashboard]
- The current version of this document can be downloaded from the following link

Thank you!