

Applied Data Science Capstone

Exploring and working
with SpaceX data



IBM Developer
SKILLS NETWORK

By **BHAVESH JAIN**

Outline

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



Executive Summary

Summary of all results

- Exploratory data analysis results
- Interactive analytics results
- Predictive analysis results

Summary of Methodologies

- Data collection
 - Data wrangling
 - EDA with data visualization
 - EDA with SQL
 - Interactive map with Folium
 - Dashboard with Plotly Dash
 - Predictive analysis (Classification)
-

Introduction to the Problem

Background

SpaceX is a company that is developing and testing reusable rockets for spaceflight.

Context

We here, need to find a way to predict which rocket will land back successfully.

And what all matters to ensure a successful landing by analysing the data

Problem Statements

The Key problems at hand are:

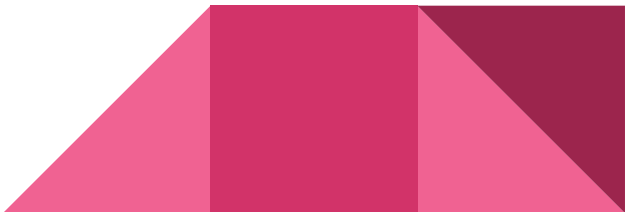
- What factors play a role in determining safe landing
- Relationship between various factors and success

Methodology

Section 1

- Executive Summary
- Timeline
- Data Collection
 - Overview
 - API
 - Web Scraping
- Wrangling
- EDA
 - Matplotlib
 - SQL
- Interactive Visualisation
 - Folium Maps
 - Dash
- Predictive Analysis
- Results

Executive Summary (Methodology)

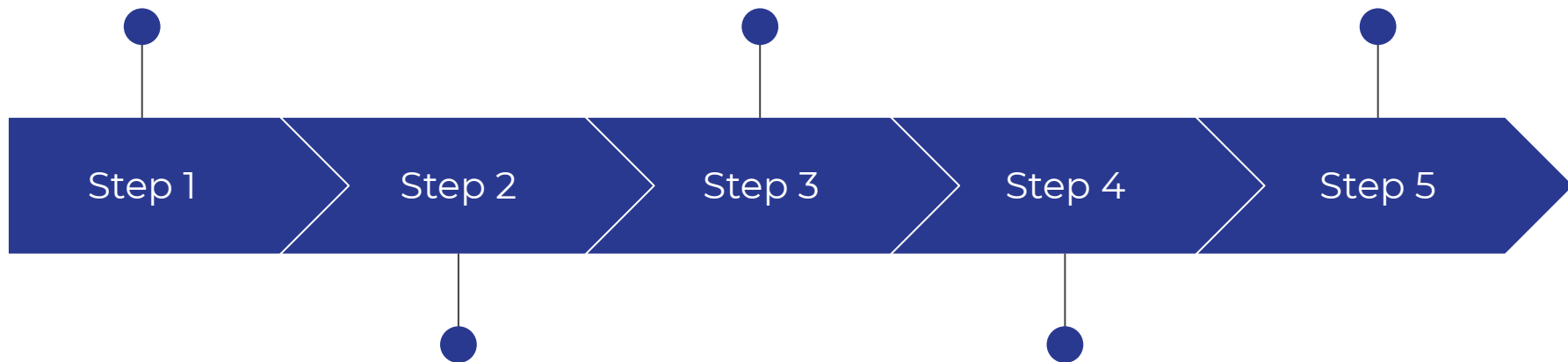
1. Data collection methodology:
 - SpaceX Rest API
 - (Web Scraping) from Wikipedia
 2. Data wrangling
 - One Hot Encoding data fields for Machine Learning and dropping irrelevant columns
 3. Performed exploratory data analysis (EDA) using visualization and SQL
 - Plotting Scatter Graphs, Bar Graphs to show relationships between variables to show patterns of data.
 4. Performed interactive visual analytics using Folium and Plotly Dash
 5. Performed predictive analysis using classification models
 - Models were imported from Scikit Learn libraries
 - Model fine tuned using grid search
- 

Timeline

Data Collection

Exploratory Data Analysis

Predictive Analysis



Data Wrangling

Interactive
Analytics



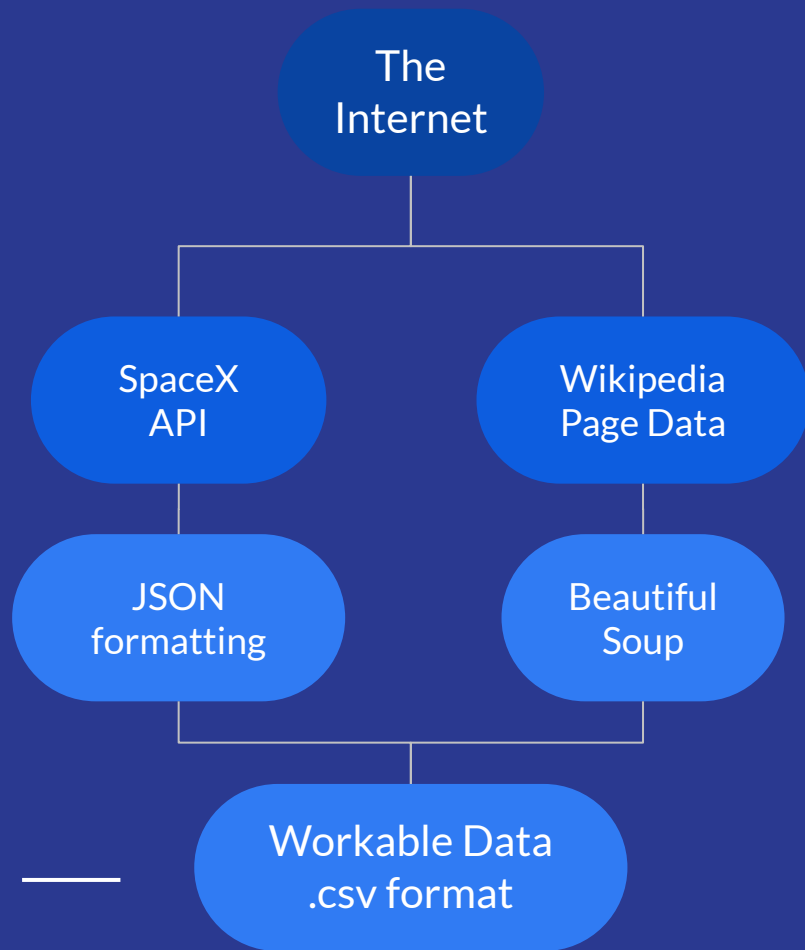
Data Collection (Overview)

There are multiple ways to pull data from the internet.

2 methods are used here to extract web data and convert them to usable **.csv files**.

1. Calling the spacex API and cleaning the data using *json library*.
2. Scraping the data from wikipedia and pulling useful information from the *beautifulsoup library*.

This procedure enables us to get a consolidated dataframe with all the important information we need for further processing like wrangling, visualisation and prediction.



SpaceX API

[Github link](#)

Data was pulled from the SpaceX API

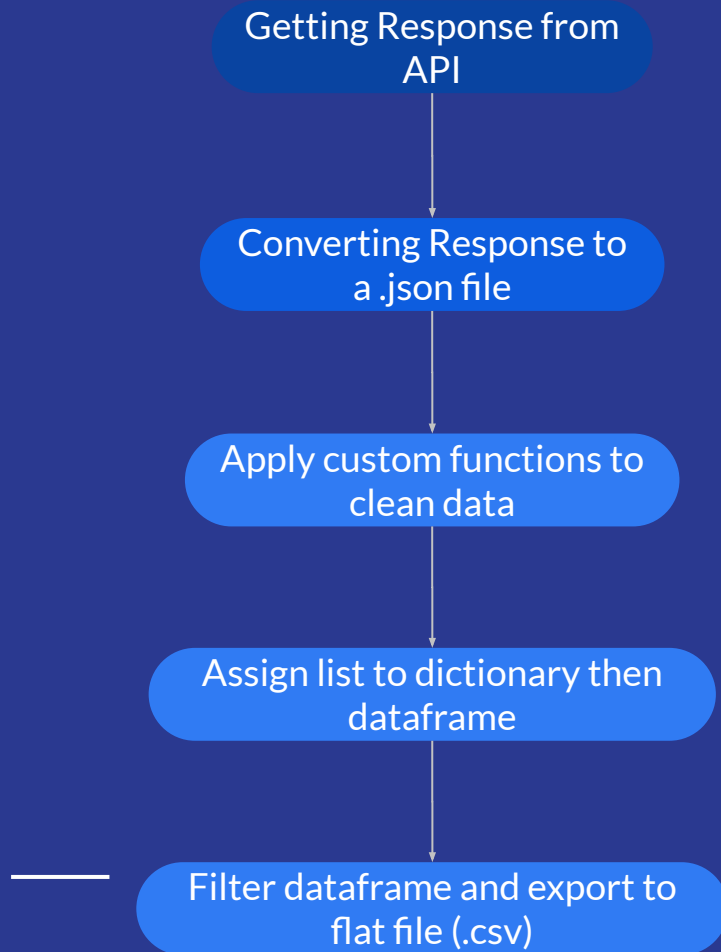
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
response = requests.get(spacex_url)
```

This was followed by normalising data

```
response = requests.get(static_json_url)
data = response.json()
data = pd.json_normalize(data)
```

Then only Falcon 9 data was chosen from the *data* DataFrame and exported as a csv



Web Scrapping

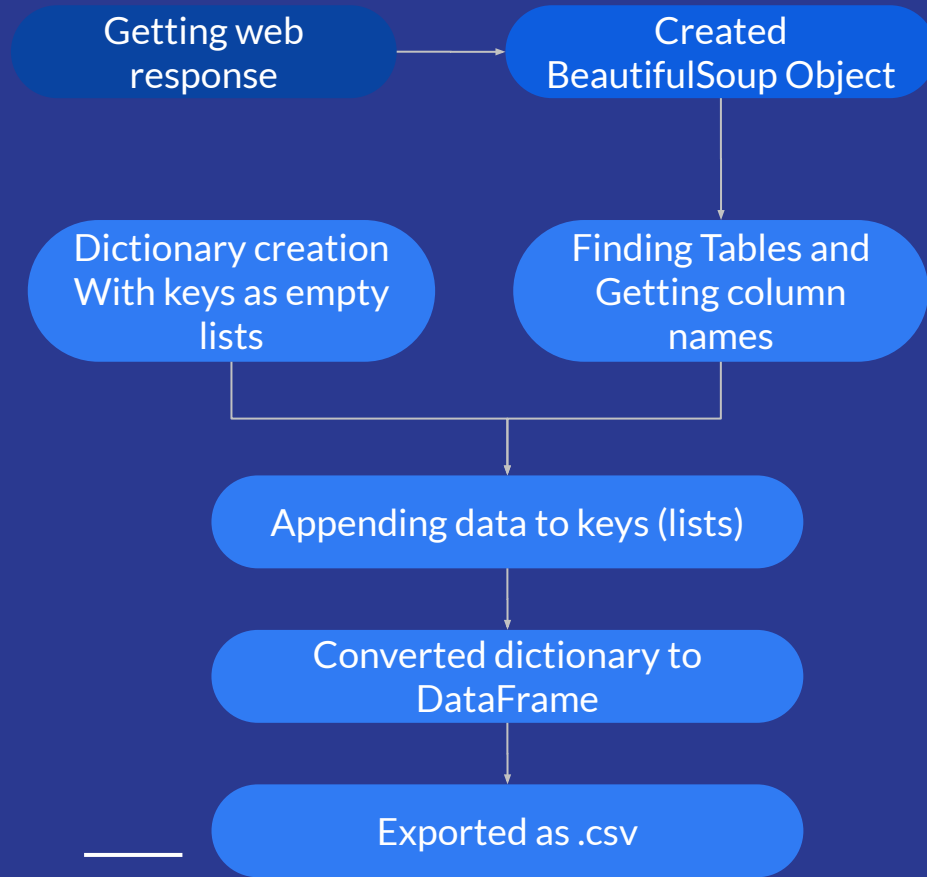
[Github link](#)

Data was pulled from the wikipedia and processed using the *BeautifulSoup* library.

A dictionary is created and empty arrays/lists are initialised

The needed tables were located in the parsed data and content from there was fed into arrays created during dictionary creation.

The dictionary was then converted to a dataframe using the *pandas* library and exported as a csv.



Data Wrangling

[Github link](#)

Data was imported from the csv created in the earlier step.

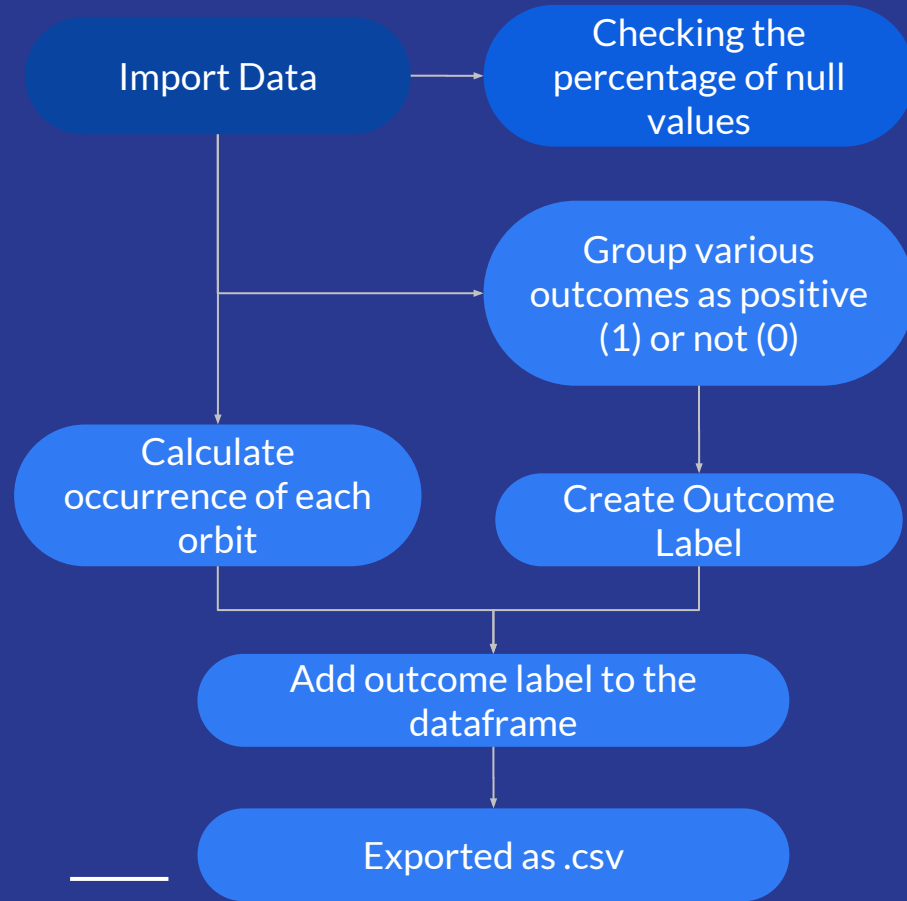
First, each column was checked for the percentage of null values.

Then the number of occurrences of each orbit were examined.

All possible outcomes (8) were identified and classified as positive i.e. successful (1) or not (0).

This was then added as the *Label* column in the data frame that will be our final target variable.

The new and edited data frame was then exported.

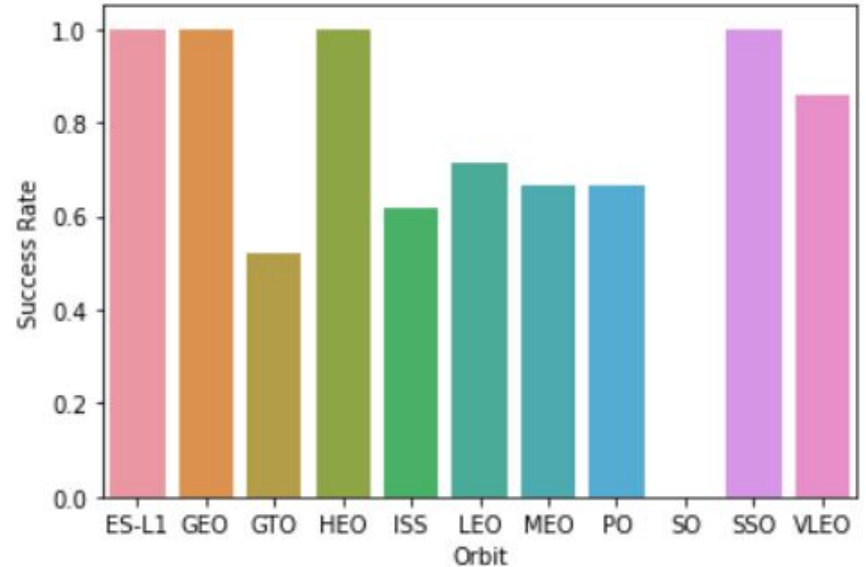


EDA with Data Visualisation

[Github Link](#)

The Graphs used are:

1. ScatterPlot
 - Flight Number vs. Payload Mass
 - Flight Number vs. Launch Site
 - Payload vs. Launch Site
 - Orbit vs. Flight Number
 - Payload vs. Orbit Type
 - Orbit vs. Payload Mass
2. BarChart
 - Orbit vs. Success Rate
3. LineChart
 - Success Rate over the years



The Bar Chart Used

EDA with SQL

[Github Link](#)

The Queries used are:

- Displaying the names of the unique launch sites in the space mission
- Displaying 5 records where launch sites begin with the string 'KSC'
- 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 where the successful landing outcome in drone ship was achieved.
- Listing the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000



EDA with SQL

[Github Link](#)

The Queries used are (contd.):

- 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 records which will display the month names, successful landing_outcomes in ground pad ,booster
- versions, launch_site for the months in year 2017
- Ranking the count of successful landing_outcomes between the date 2010 06 04 and 2017 03 20 in descending order.



Interactive Map with Folium

[Github Link](#)

To visualize the Launch Data into an interactive map, the latitude and longitude coordinates at each launch site and added a **Circle Marker** around each launch site with a **label of the name of the launch site**.

We assigned the dataframe *launch_outcomes*(failures, successes) to classes 0 and 1 with **Green and Red markers** respectively on the map in a *MarkerCluster()*

The distance from the Launch Site to various landmarks was used to find various trends about what is around the Launch Site to measure patterns. Lines are drawn on the map to measure distance to landmarks



Location markers on 2
of the launch sites

Dashboard using Plotly Dash

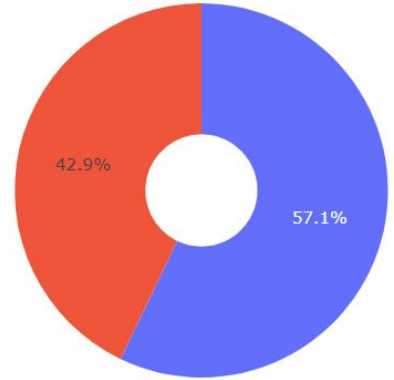
[Github Link](#)

The main purpose to create an interactive dashboard was so that the data could be analysed dynamically on the go.

A **dropdown** is used for selecting the launchbase for which the viewer wants to see the *success ratio* and *Booster vs payload relation*.

The *success ratio* itself is displayed using a **Pie Chart** underneath the dropdown.

A **slider** is used to select the payload range for which a **ScatterPlot** is displayed that shows the *Booster Versions* on Y axis and *Payload weight* on X axis.



Success Ratio Pie Chart



Predictive Analysis

[Github Link](#)

Data Import

The data is imported and split into features and target.

The class column is selected as the target

The data is scaled using *standard scaler* and split into training and test set with test set being 20% of the complete data set

Model Creation and Tuning

The models used and compared were Logistic Regression, SVM, Decision Tree and K nearest neighbor. They were fed the train data and evaluated on their test data performance.

They were fine tuned using grid search where the selected the ebay parameters from a pre defined dictionary

Evaluation

The metrics used for the evaluation of each model were the accuracy, best accuracy and plotting of confusion matrix.

Each of these results was stored in a separate data frame used strictly for evaluation purposes

Selection

From that separate data frame, we could select the performance of all models together and were hence able to select the best on depending on the accuracy.

Results

Exploratory Data Analysis

- The results from the various EDA notebooks will be in subsequent sections

Interactive Visualisation

- The results from the various EDA notebooks will be in subsequent sections

Predictive Analysis

- The best model is the **Decision Tree** classifier

Insights from The Project

Section 2

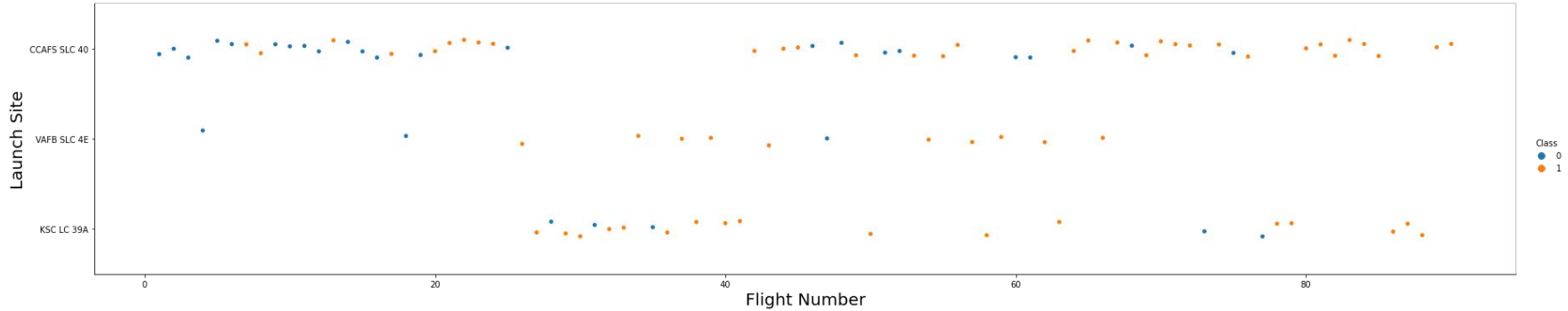
- EDA by Visualisation
- EDA by SQL
- Launch Site Analysis
- Dashboard
- Predictive Analysis

EDA by Visualisation

Insights and plots

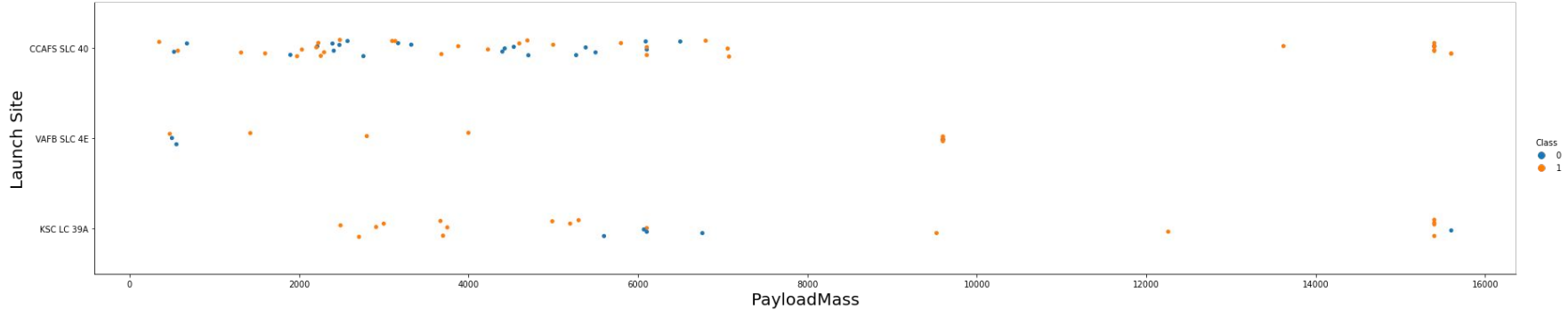
- Flight number vs Launch Site
 - Payload vs Launch Site
 - Success Rate vs Orbit Type
 - Flight Number vs Orbit Type
 - Payload vs Orbit Type
 - Yearly Success Trend
-

Flight Number vs. Launch Site



Success rate at all tree launch sites increases with an increase in flight number i.e. the more the flights, higher the success possibility

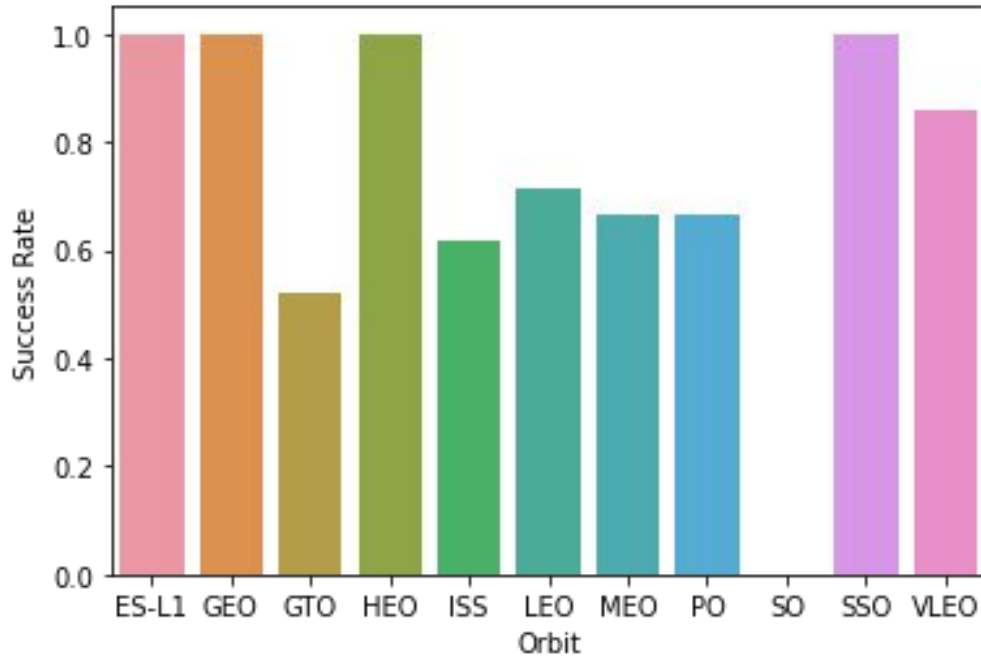
Payload vs. Launch Site



A payload higher than 6500kg will normally be successful irrespective of the launch site

SLC 4E and **LC 39A** have a high success rate in 1000 to 5500 kg range also

Success Rate vs. Orbit Type

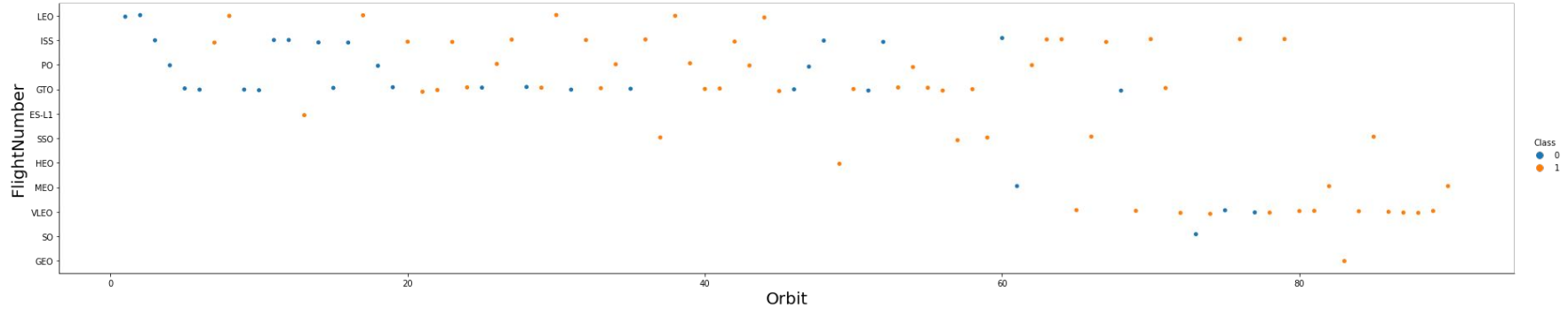


The **ES-L1**, **GEO**, **SSO** and **HEO** have a 100% success rate.

The **SO** orbit has not yet had a successful landing

The success of other orbits lies in the 40% to 80% success rate ballpark

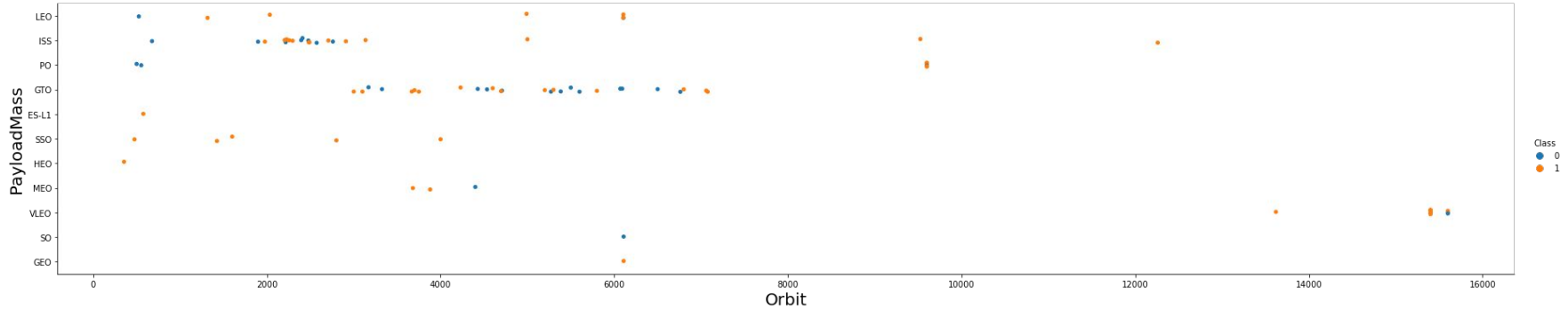
Flight Number vs. Orbit Type



The success rate increases with an increase in the launch number, i.e. with more and more launches, the success possibility also goes up for all orbits

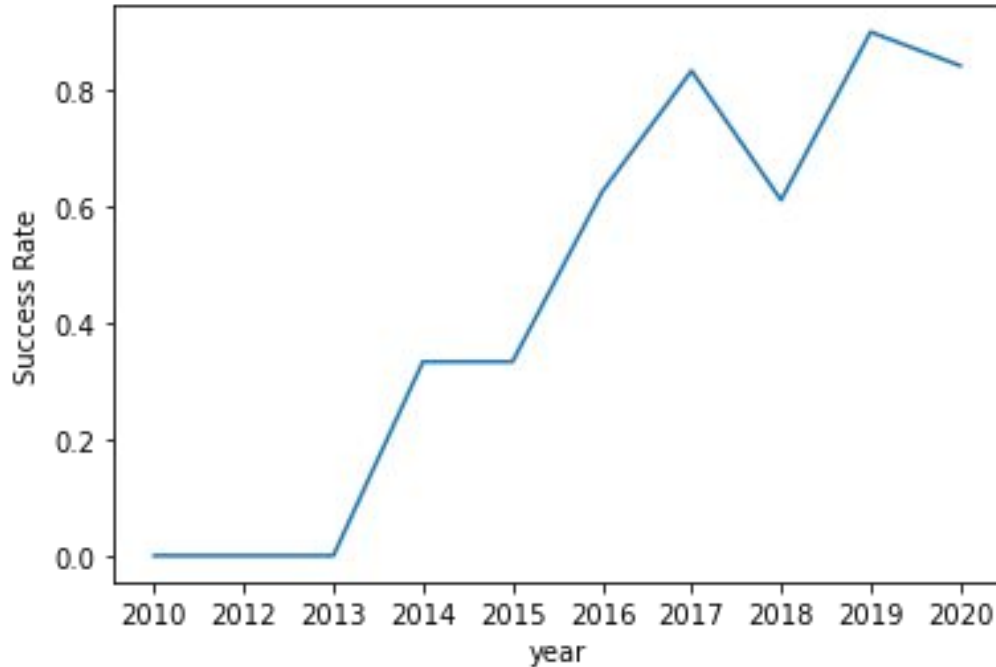
Also most of the later launches belong to the **VLEO** orbit.

Payload vs. Orbit Type



Heavy payloads have a negative influence on GTO orbits and positive on GTO and Polar LEO (ISS) orbits.

Launch Success Yearly Trend



Barring 2018 and 2020, there has always been an increase in success rate with progression of time.

It is important to see that SpaceX went from 0% to roughly 40% in just one year i.e. 2013-2014.

EDA by SQL

Insights and Answers

- Unique Launch Sites
 - Launch Site Names Begin with 'CCA'
 - Total Payload by Nasa
 - Average Payload Mass by F9 v1.1
 - First Successful Ground Landing Date
 - Successful Drone Ship Landing with Payload between 4000 and 6000
 - Total Number of Successful and Failure Mission Outcomes
 - Boosters Carried Maximum Payload
 - 2017 Launch Records
 - Rank Landing Outcomes Between 2010-06-04 and 2017-03-20
-

Unique Launch Sites

Unique Launch Sites
CCAFS LC-40
CCAFS SLC-40
CCAFS SLC-40
KSC LC-39A
VAFB SLC-4E

Using the word **DISTINCT** in the query means that it will only show Unique values in the *Launch_Site* column from **tblSpaceX**



Launch Site Names Begin with 'CCA'

	Date	Time_UTC	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS_KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
0	19-02-2017	2021-07-02 14:39:00.0000000	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
1	16-03-2017	2021-07-02 06:00:00.0000000	F9 FT B1030	KSC LC-39A	EchoStar 23	5600	GTO	EchoStar	Success	No attempt
2	30-03-2017	2021-07-02 22:27:00.0000000	F9 FT B1021.2	KSC LC-39A	SES-10	5300	GTO	SES	Success	Success (drone ship)
3	01-05-2017	2021-07-02 11:15:00.0000000	F9 FT B1032.1	KSC LC-39A	NROL-76	5300	LEO	NRO	Success	Success (ground pad)
4	15-05-2017	2021-07-02 23:21:00.0000000	F9 FT B1034	KSC LC-39A	Inmarsat-5 F4	6070	GTO	Inmarsat	Success	No attempt

Using the word **TOP 5** in the query means that it will only show 5 records from **tblSpaceX** and **LIKE** keyword has a wild card with the character the '%' in the end suggests that the *Launch_Site* name must start with KSC.

Total Payload Mass by NASA

Total Payload Mass	
0	45596

Using the function **SUM** summates the total in the column **PAYLOAD_MASS_KG_**

The **WHERE** clause filters the dataset to only perform calculations on *Customer NASA (CRS)*



Average Payload Mass by F9 v1.1

Average Payload Mass	
0	2928

Using the function **AVG** works out the average in the column **PAYLOAD_MASS_KG_**

The WHERE clause filters the dataset to only perform calculations on

Booster_version F9 v1.1




First Successful Ground Landing Date

Date which first Successful landing outcome in drone ship was acheived.	
0	06-05-2016

Using the function **MIN** works out the minimum date in the column *Date*

The **WHERE** clause filters the dataset to only perform calculations on *Landing_Outcome Success* (drone ship)



Successful Drone Ship Landing with Payload between 4000 and 6000

Selecting only *Booster_Version*

The **WHERE** clause filters the dataset to *Landing_Outcome* = *Success* (drone ship)

The **AND** clause specifies additional filter conditions *Payload_MASS_KG* 4000 **AND** *Payload_MASS_KG* 6000

Date which first Successful landing outcome in drone ship was acheived.

0	F9 FT B1032.1
1	F9 B4 B1040.1
2	F9 B4 B1043.1

Total Number of Successful and Failure Mission Outcomes

Successful_Mission_Outcomes	Failure_Mission_Outcomes
0	100
	1

Here, subqueries were used to produce the results. The **LIKE** '%foo%' wildcard shows that in the record the foo phrase is in any part of the string in the records for example.



Boosters Carried Maximum Payload

	Booster_Version	Maximum Payload Mass
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
...
92	F9 v1.1 B1003	500
93	F9 FT B1038.1	475
94	F9 B4 B1045.1	362
95	F9 v1.0 B0003	0
96	F9 v1.0 B0004	0

97 rows x 2 columns

Using the word **DISTINCT** in the query means that it will only show Unique values in the *Booster_Version* column from *tblSpaceX*

GROUP BY puts the list in order set to a certain condition.

DESC means its arranging the dataset into descending order



2017 Launch Records

Month	Booster_Version	Launch_Site	Landing_Outcome
January	F9 FT B1029.1	VAFB SLC-4E	Success (drone ship)
February	F9 FT B1031.1	KSC LC-39A	Success (ground pad)
March	F9 FT B1021.2	KSC LC-39A	Success (drone ship)
May	F9 FT B1032.1	KSC LC-39A	Success (ground pad)
June	F9 FT B1035.1	KSC LC-39A	Success (ground pad)
June	F9 FT B1029.2	KSC LC-39A	Success (drone ship)
June	F9 FT B1036.1	VAFB SLC-4E	Success (drone ship)
August	F9 B4 B1039.1	KSC LC-39A	Success (ground pad)
August	F9 FT B1038.1	VAFB SLC-4E	Success (drone ship)
September	F9 B4 B1040.1	KSC LC-39A	Success (ground pad)
October	F9 B4 B1041.1	VAFB SLC-4E	Success (drone ship)
October	F9 FT B1031.2	KSC LC-39A	Success (drone ship)
October	F9 B4 B1042.1	KSC LC-39A	Success (drone ship)
December	F9 FT B1035.2	CCAFS SLC-40	Success (ground pad)

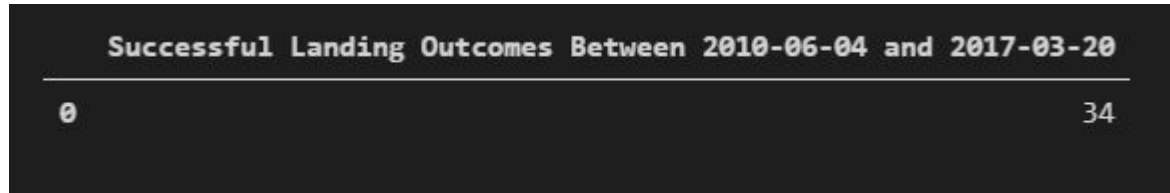
Server stored as **NVARCHAR** the **MONTH** function returns name month.

The function **CONVERT** converts *NVARCHAR* to *Date*

WHERE clause filters Year to be 2017



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



Function COUNT counts records in column **WHERE** filters data

LIKE (wildcard)

AND (conditions)

AND (conditions)

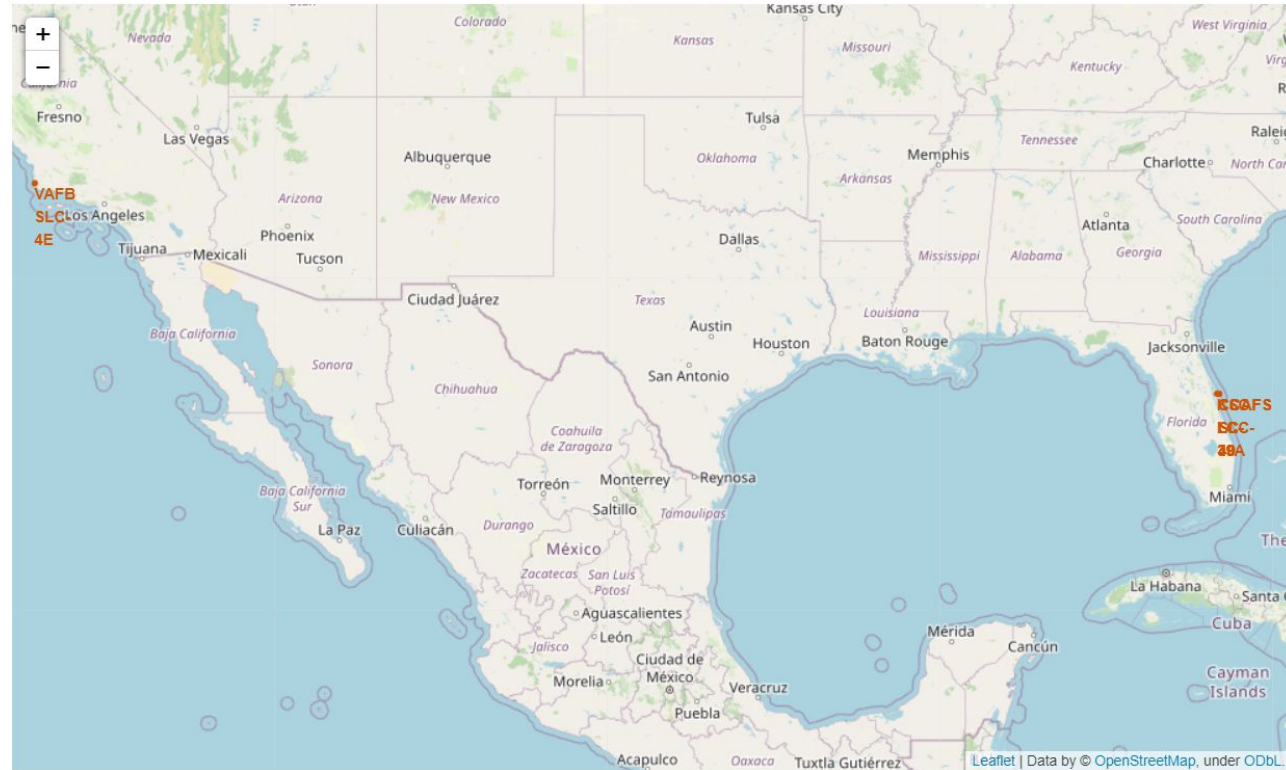


Launch Sites Proximities

Insights and Analysis

- Launch Site Location
- Launch Outcomes (Florida)
- Launch Outcomes (California)
- Launch Site Proximities (Florida)

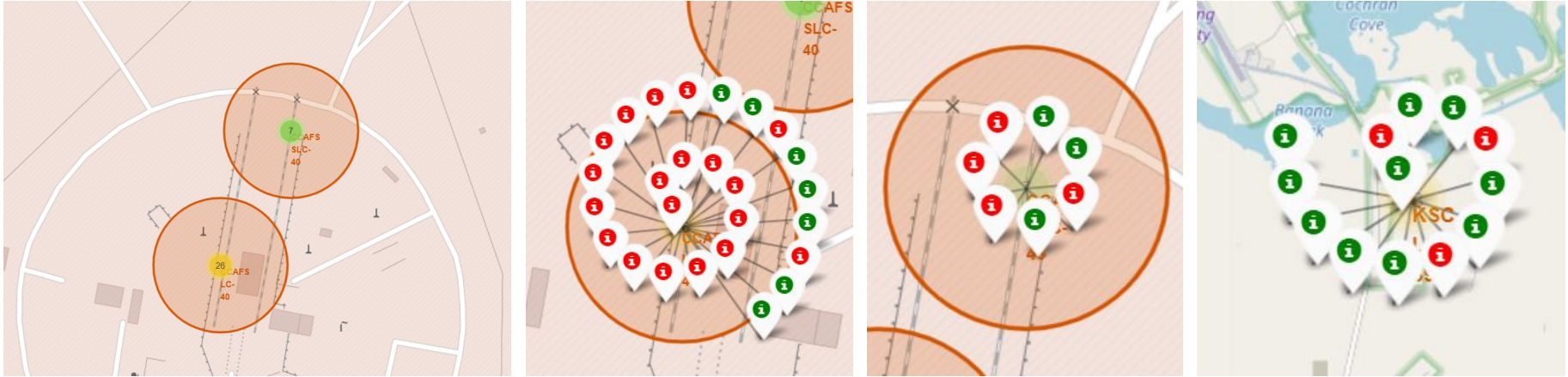
Launch Site Location



The launch sites are on situated on the coasts of **USA.**

They are located in *Florida (East Coast)* and *California (West Coast)*

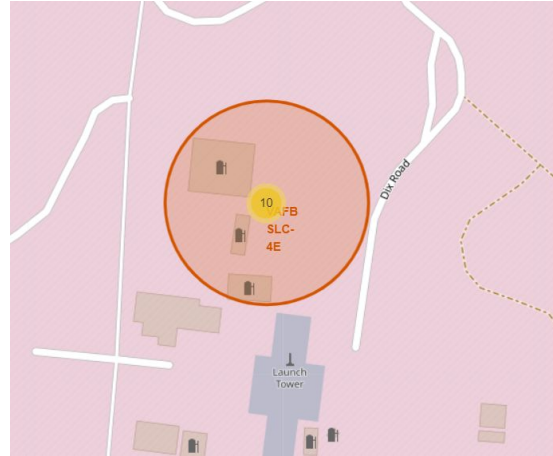
Launch Outcomes (Florida)



Green markers represent successful Launches

Red markers represent unsuccessful ones

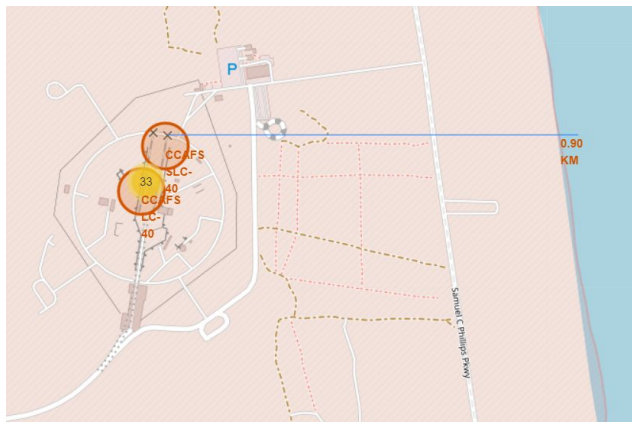
Launch Outcomes (California)



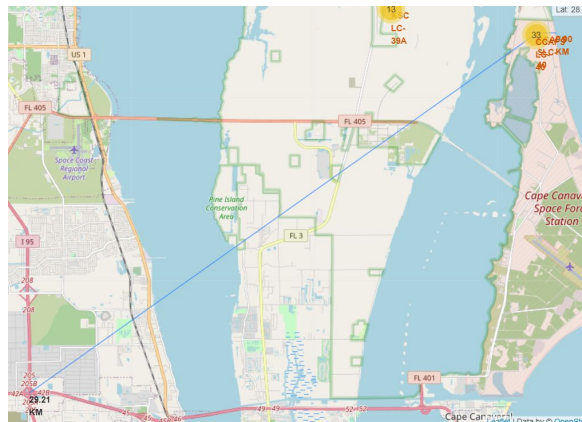
Green markers represent successful Launches

Red markers represent unsuccessful ones

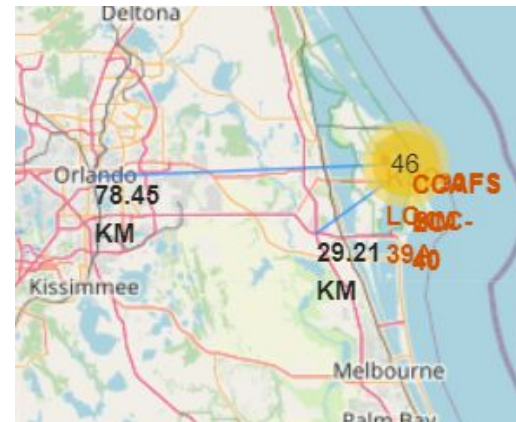
Launch Site Proximities (Florida)



Distance to coast **9.81km**



Distance to highway **29.21km**



Distance to city **78.45km**

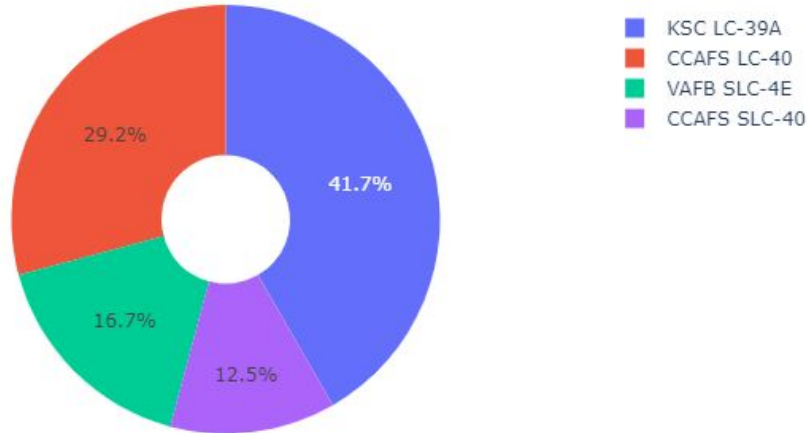
Interactive Dashboard

Insights and Analysis

- Launch Success Overview
- Launch Overview LC-39A
- Light Payload vs Outcome
- Heavy Payload vs Outcome

Launch Success Overview

Total Success Launches By all sites

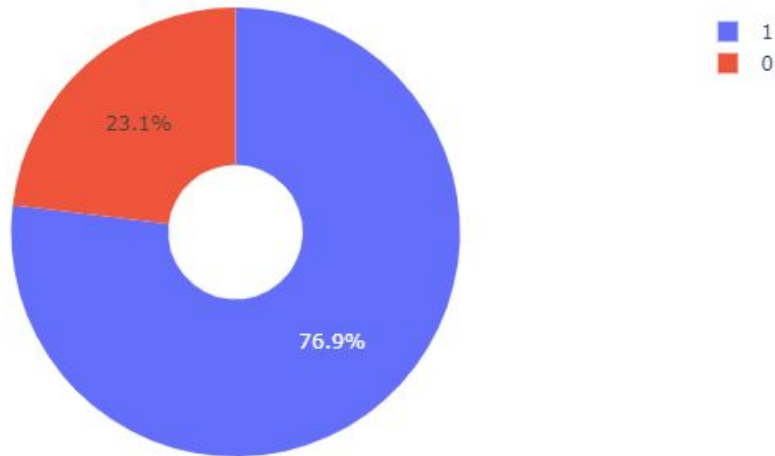


This pie chart shows us the distribution of the *Successful Launches* based on the **Launch Site** used.

LC39A contributes to the most where as **SLC40** contributes to the least successful missions

Launch Overview LC-39A

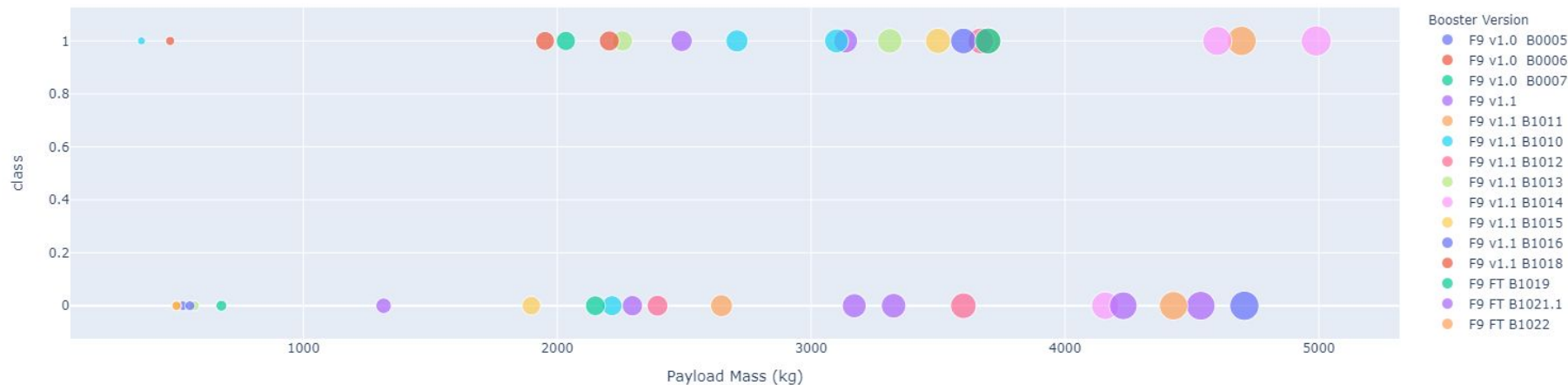
Total Success Launches for site KSC LC-39A



Of all launches at this site, roughly **77%** have been successful.

This is the most successful launch site.

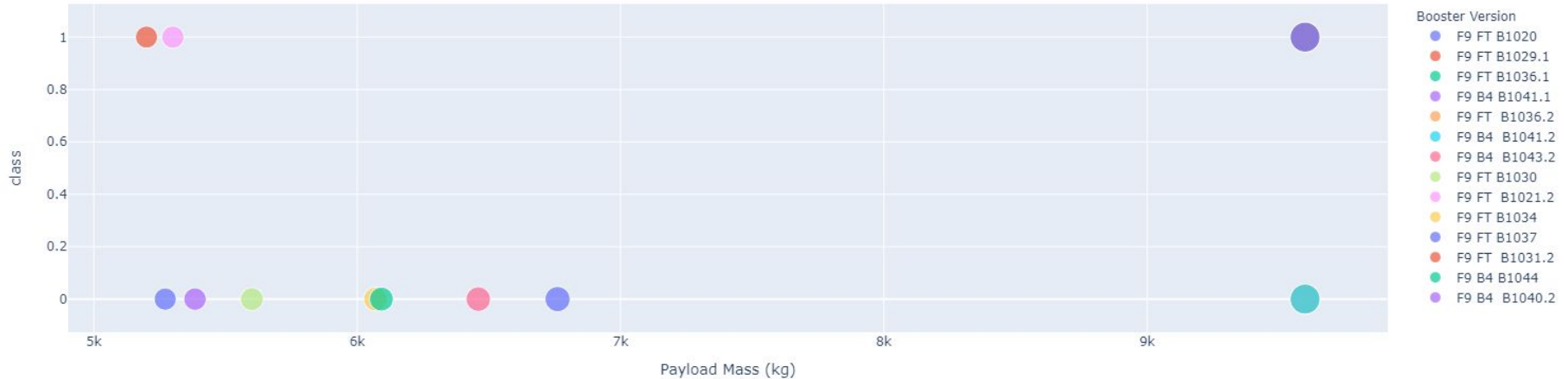
Light Payload vs Outcome



Graph of **Outcome** vs. **Light Payload** (0 to 5000kg)

The colours specify booster versions

Heavy Payload vs Outcome



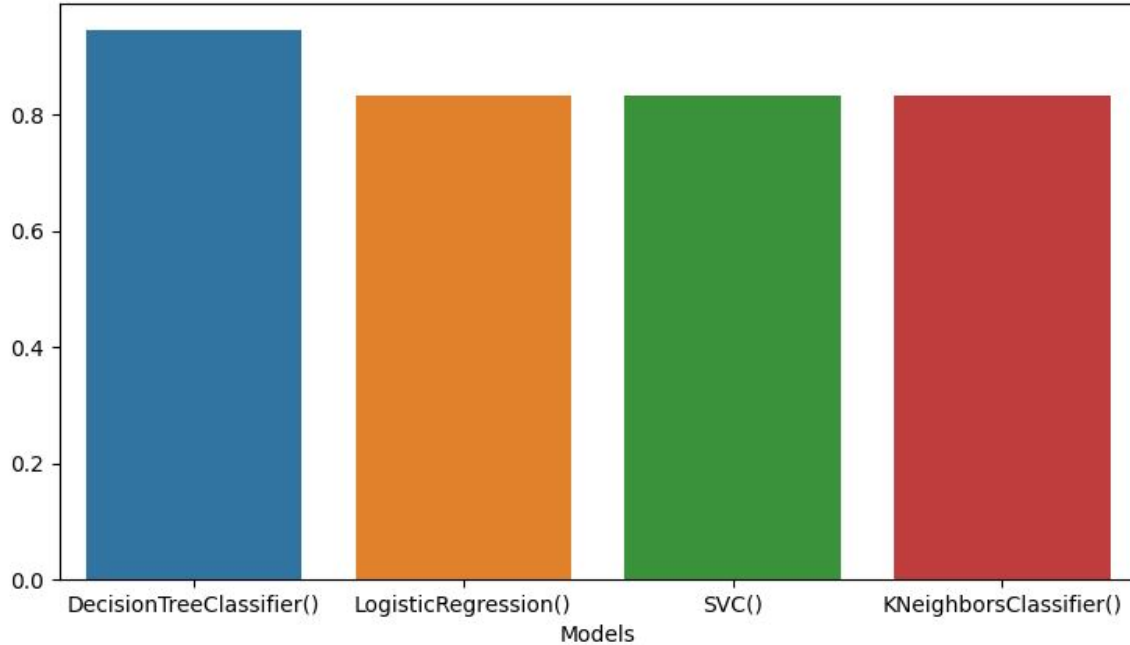
Graph of **Outcome** vs. **Heavy Payload** (5000 kg to 10000 kg)
The colours specify booster versions

Predictive Analysis

Insights and Analysis

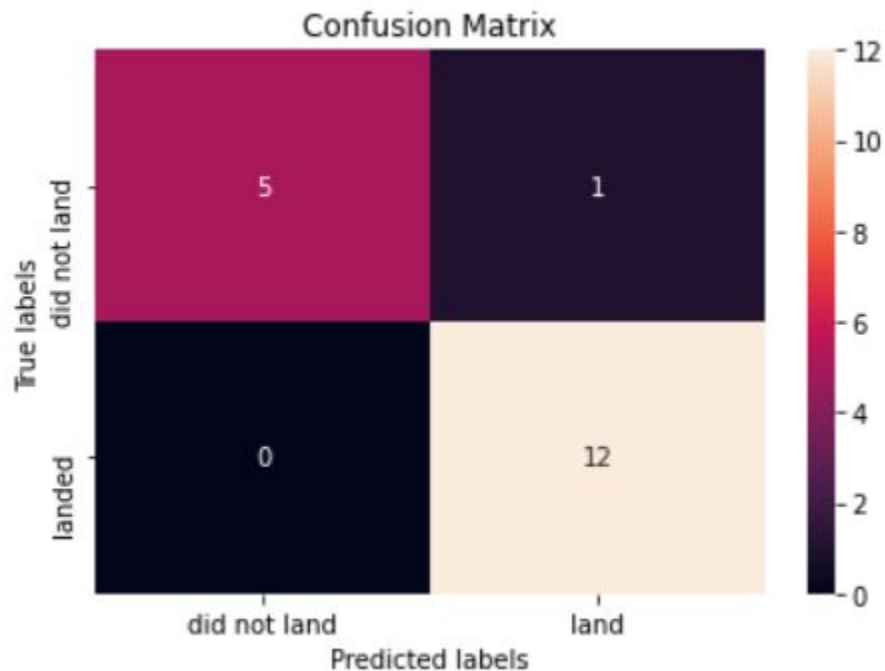
- Score Comparison
- Confusion Matrix

Score Comparison of models



Decision Tree Classifier is the best model as it outperforms all others that have a more or less same accuracy score.

Confusion Matrix



The confusion matrix on left is good way to gauge the performance

Of the 18 identifications, 17 were predicted correctly and the one misidentified was a unsuccessful launch that was predicted as a successful one



94.44%

The accuracy of the **Decision Tree Classifier** to predict if a rocket will land successfully or not

Conclusion

- The success rates for SpaceX launches is directly proportional time in years they will eventually perfect the launches.
- KSC LC 39A had the most successful launches from all the sites.
- Orbit GEO,HEO,SSO,ES L1 has the best Success Rate.
- Decision Tree Classifier is the best model (here) for classification.

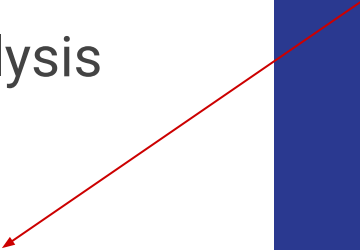
Appendix

A description of custom code snippets used

Models Dataframe

Used in the Predictive Analysis

```
In [160]: model = []  
          best_parameter = []  
          best_score = []  
          score = []
```



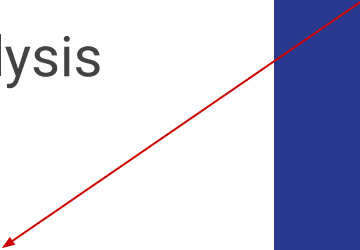
For my own ease, I had created a dataframe wherein I could glance over the performance of all models in one go.

Firstly, 4 empty lists were initialised. These lists were:

- model
 - To store model names
 - best_parameter
 - To store the best set of parameters returned post grid search
 - best_score
 - To store the best score
 - score
 - To store the average score
-

Models Dataframe

Used in the Predictive Analysis
(contd.)



```
#adding the model details to the models dataframe  
model.append(logreg_cv.estimator)  
best_parameter.append(logreg_cv.best_params_)  
best_score.append(logreg_cv.best_score_)  
score.append(logreg_cv.score(X_test,Y_test))
```

To ensure that a dataframe could be created out of them,

After the evaluation and running of each models, the respective parameters were appended to the lists

This ensured me to automatically feed the data and not enter it manually at the end

Example showed here is for **Logistic Regression**

Models Dataframe

Used in the Predictive Analysis
(contd.)

In the end, a data frame with the name **Models** was created out of those filled arrays

It was then sorted by the **Score** column in a descending manner so that the model with the highest average accuracy stays on top and is easier to see.

```
Models = pd.DataFrame({'Model':model,'Score':score,'Best Score':best_score,'Best Parameters':best_parameter})  
Models = Best_Model.sort_values(by='Score', ascending=False).reset_index().drop('index',axis=1)
```

	Model	Score	Best Score	Best Parameters
0	DecisionTreeClassifier()	0.944444	0.903571	{'criterion': 'gini', 'max_depth': 6, 'max_fea...
1	LogisticRegression()	0.833333	0.846429	{'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
2	SVC()	0.833333	0.848214	{'C': 1.0, 'gamma': 0.03162277660168379, 'kern...
3	KNeighborsClassifier()	0.833333	0.848214	{'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}

This is what the completed
dataframe looks like

Thank You



IBM Developer
SKILLS NETWORK

By **BHAVESH JAIN**