



IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Syed Khaleelullah
Data Scientist



Outline

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

Executive Summary

Based on the need to reduce costs and increase the performance of aerospace expeditions, this project aims to predict the successful landing of the first stage of a rocket.

Data from SpaceX Falcon 9 launches was analyzed. The data was collected through an API and web scraping, the steps of cleaning, transforming and then visualizing and manipulating the data through Exploratory Data Analysis (EDA) were performed, for the prediction of future launches, classification models were used with evaluation of the models.

The developed model presented a good accuracy evaluation and returned satisfactory results with great confidence.



Introduction

Background

- This is the final course in the IBM Data Science Professional Certificate.
- This report has been created as part of the Applied Data Science Capstone course.
- In this report, I take the role of a Data Scientist to help a startup compete with SpaceX. With my Data Science findings and models involving data collection, data wrangling, exploratory data analysis, data visualization, model development and model evaluation, the startup SpaceY can make informed decisions against SpaceX for a rocket launch.

* 10th course in the [IBM Data Science Professional Certificate](#)



Introduction

Business Problem

- The strongest rockets are now being made by SpaceX, who manages to keep their costs lower than their rivals. The Owner of SpaceX Elon Musk, has made contributions to science in a variety of fields, including the exploration of the distant, dark reaches of space, the creation of new, energy-efficient automobiles, and the programming of computers to facilitate daily life.
- The aerospace company SpaceX was able to lower the cost of its missions by creating a technology that allowed the rocket's first stage to be reused. Depending, SpaceX may occasionally sacrifice the first stage. on the mission's payload, orbit, and client.
- When the first stage of their rockets can be reused, SpaceX promotes Falcon 9 rocket flights with a cost of 62 million dollars.
- Construction of the first stage is anticipated to cost upwards of \$15 million, excluding profit margin and R&D cost recovery.
- Therefore if we can determine if the first stage will land, we can determine the cost of a launch. We will a machine learning pipeline to predict if the first stage will land given the data.
- Here we will showcase our findings through this report!



Section 1

Methodology

Methodology

Executive Summary

1. Data Collection

- The Data was collected using the SpaceX API and by webscraping the data from the Wikipedia page “Falcon 9 and Falcon Heavy launch list”

2. Data Wrangling

- Clean and Transform the data into a presentable and usable form.

3. Exploratory Data Analysis

- Perform exploratory data analysis (EDA) using visualization and SQL

4. Data Visualization

- Perform interactive visual analytics using Folium and Plotly Dash

5. Model Development

- Perform predictive analysis using classification models
- Build, tune, evaluate classification models

6. Reporting Results

- Present findings and results

Data Collection

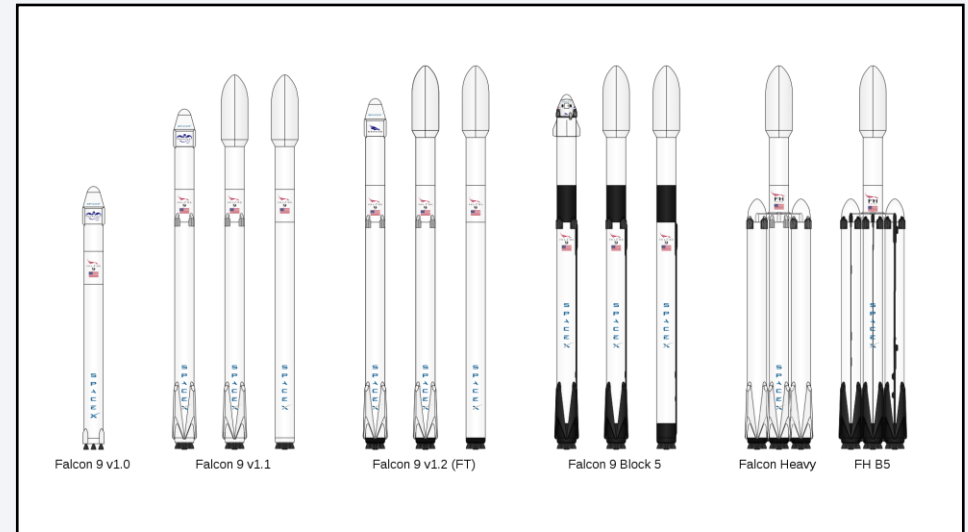
The Datasets were collected through two methods:

1. SpaceX API

- Retrieved historical launch data for SpaceX using open source REST API
- Perform Get request to the SpaceX API and clean the requested data
- Filter the DataFrame to only include Falcon 9 launches


2. Web scraping

- Performed web scraping to collect Falcon 9 historical launch records from a Wikipedia page titled “List of Falcon 9 and Falcon Heavy launches”
- Obtained all columns and variables from the HTML header.
- Parsed the HTML and converted into a Pandas DataFrame.



Data Collection – SpaceX API

- Retrieved historical launch data for SpaceX using open source REST API
- Request and parse the SpaceX launch data using the GET request
- The response was through the JSON language, which was transformed into a Pandas DataFrame.



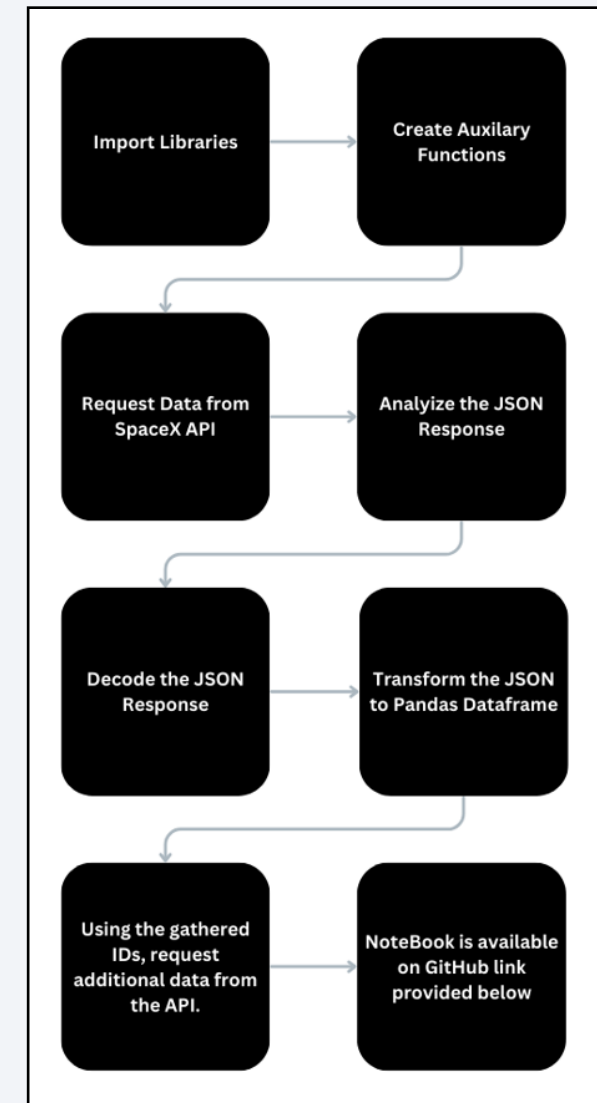
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
response = requests.get(spacex_url)
response.json()
data = pd.json_normalize(response.json())
```

The screenshot shows a Jupyter Notebook window titled "IBM-DataScience-Capstone-SpaceX-Falcon9.py". It contains the following Python code: `spacex_url="https://api.spacexdata.com/v4/launches/past"`, `response = requests.get(spacex_url)`, `response.json()`, and `data = pd.json_normalize(response.json())`. The CodeImage logo is visible in the bottom right corner of the notebook interface.

- Filtered the DataFrame to only include the Falcone9 launches.

Notebook on Data Collection using SpaceX API GitHub link:

[Data Collection SpaceX API](#)



Data Collection – SpaceX API

From the DataFrame, it was noticed that a lot of the data was IDs. Using these IDs, the most relevant information was requested from the SpaceX API.

- Lets construct our dataset using the data we have obtained. We will combine the columns into a dictionary.
- Create a Pandas data frame from the dictionary launch_dict.

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

From the rocket we would like to learn the booster name

From the payload we would like to learn the mass of the payload and the orbit that it is going to

From the launchpad we would like to know the name of the launch site being used, the longitude, and the latitude.

From cores we would like to learn the outcome of the landing, the type of the landing, number of flights with that core, whether gridfins were used, whether the core is reused, whether legs were used, the landing pad used, the block of the core which is a number used to separate version of cores, the number of times this specific core has been reused, and the serial of the core.

IDs	Data Obtained
Rocket	Booster Name
Payload	Mass of the Payload and Orbit
Launchpad	Name of the launch site, the longitude and latitude
Cores	Outcome of landing, type of landing, number of flights, gridfins reused, core reused, legs reused, landing pad used, block of core, number of cores reused and the serial of the core

Data Collection – SpaceX API

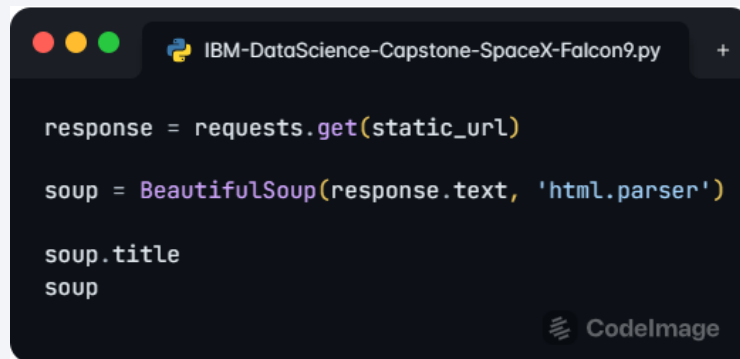
- Finally, we remove the Falcon 1 launches keeping only the Falcon 9 launches. Filtered the DataFrame using **Booster Version**.
- Save the filtered data into a new Pandas DataFrame.

```
IBM-DataScience-Capstone-SpaceX-Falcon9.py +  
  
data_falcon9 = launch_df.loc[launch_df['BoosterVersion'] != 'Falcon 1']  
  
data_falcon9.loc[:, 'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))  
  
CodeImage
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs
4	1	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False
5	2	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False
6	3	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False
7	4	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False
8	5	2013-12-03	Falcon 9	3170.0	GTO	CCSFS SLC 40	None None	1	False	False	False

Data Collection – Web Scraping

- Obtained Falcon 9 historical launch records from a Wikipedia page titled “List of Falcon 9 and Falcon Heavy launches”
- Request the Falcon9 Launch Wiki page from its URL.
- Create a BeautifulSoup object from the HTML response.
- Launch records are stored in a HTML table.
- Extract the launch records HTML table header with BeautifulSoup.
- Parsed the HTML table and converted into a Pandas DataFrame.



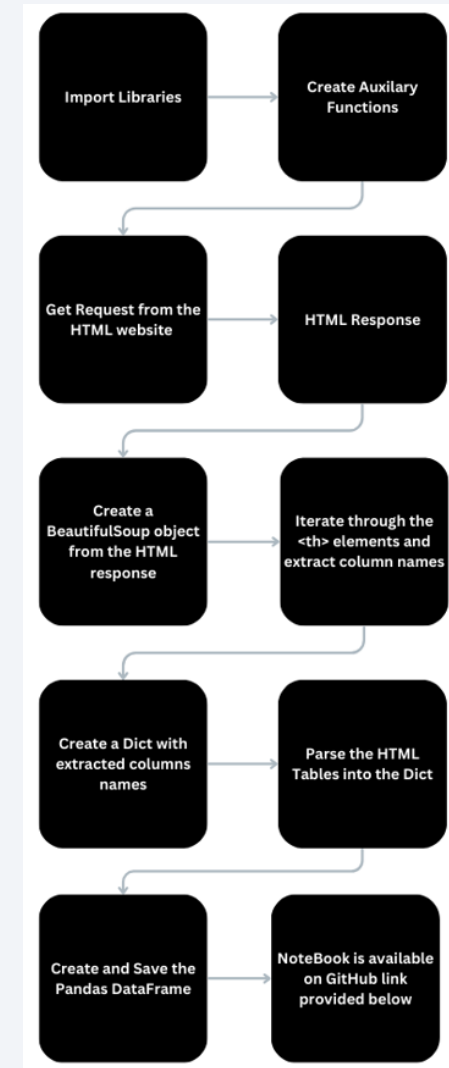
```
response = requests.get(static_url)

soup = BeautifulSoup(response.text, 'html.parser')

soup.title
soup
```

Notebook on Data Collection by Web Scraping GitHub link:

[Data Collection Web Scraping](#)



Data Wrangling - SpaceX API

- From the SpaceX API data collection, the DataFrame which included only Falcon 9 launches was checked for missing values.
- Missing values were observed in two columns: PayloadMass and LandingPad.
- In the PayloadMass column, the missing values were replaced by the mean of the column.

```
IBM-DataScience-Capstone-SpaceX-Falcon9.py +

PayloadMass_mean = data_falcon9['PayloadMass'].mean()

data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, PayloadMass_mean)

data_falcon9.isnull().sum()
```

CodeImage



```
IBM-DataScience-Capstone-SpaceX-Falcon9.py +

FlightNumber      0
Date              0
BoosterVersion    0
PayloadMass       0
Orbit             0
LaunchSite        0
Outcome           0
Flights           0
GridFins          0
Reused            0
Legs              0
LandingPad        26
Block             0
ReusedCount       0
Serial            0
Longitude         0
Latitude          0
dtype: int64
```

CodeImage

Data Wrangling

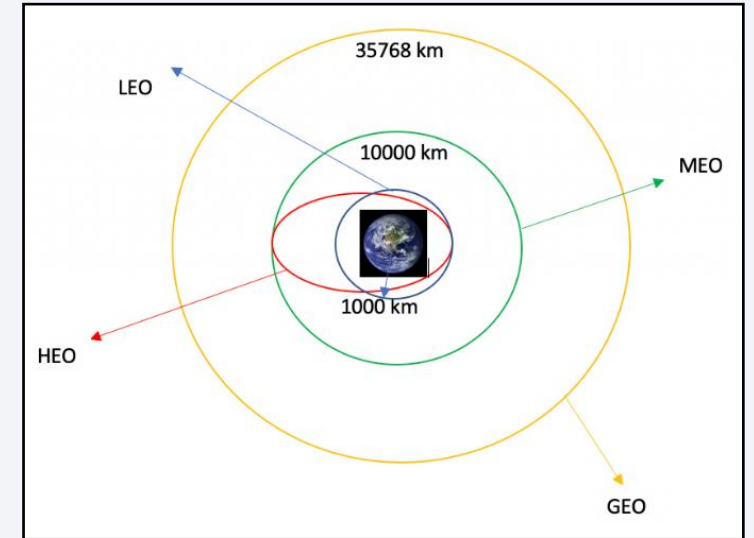
- The process of transforming and mapping data from one "raw" data form into another format with the aim of making it more acceptable and valuable for a number of downstream uses, such as analytics, is known as data wrangling.
- Data wrangling is to provide high-quality, practical data.
- Performed some Exploratory Data Analysis (EDA) to find some patterns in the data and determine what would be the label for training supervised models.

Notebook on Data Wrangling GitHub link:

[Data Wrangling](#)

Data Wrangling

- Defining an outcome variable by analyzing the data.
- After Data Analysis, it was seen to calculate:
 - the number of launches on each site
 - the number and occurrence of each orbit
 - the number and occurrence of mission outcome per orbit type
- After calculations, it was observed:
 - CCAFS SLC 40 was the most used launch site location with 55 launches;
 - More launches were done in direction of the geosynchronous orbit GTO.
 - More than 66% of the launches had a successful first stage landing.



```
df['LaunchSite'].value_counts()

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

```
df['Orbit'].value_counts()

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
```

```
landing_outcomes = df['Outcome'].value_counts()
landing_outcomes

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

Data Wrangling

- A column “Class” was introduced with binary values 0’s and 1’s that show if the first stage landed successfully or not.

Class	Data Information
0	the first stage did not land successfully
1	the first stage landed Successfully

- The target column “Class” was introduced from the column “Outcome”.
 - Class 0; the first stage did not land successfully
 - False RTLS -> ground pad landing failed
 - False Ocean -> ocean landing failed
 - False ASDS -> drone ship landing failed
 - None ASDS -> failure to land
 - None None -> failure to land
 - Class 1; the first stage landed successfully
 - False RTLS -> ground pad landing failed
 - False Ocean -> ocean landing failed

```
IBM-DataScience-Capstone-SpaceX-Falcon9.py +

landing_outcomes = df['Outcome'].value_counts()
landing_outcomes

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

```
IBM-DataScience-Capstone-SpaceX-Falcon9.py +

bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes

Output:
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

Notebook on Data Wrangling GitHub link:

[Data Wrangling](#)

EDA with Data Visualization

- Performed exploratory data analysis between various fields from the data by visualizing these data using Matplotlib and Seaborn visualization libraries.
- Various Charts were plotted:
 1. Scatter plot between Flight Number and Launch Site
 2. Scatter plot between Payload and Launch Site
 3. Bar plot between success rate of each orbit type
 4. Scatter plot between FlightNumber and Orbit type
 5. Scatter plot between Payload and Orbit type
 6. Line Chart between launch success yearly trend

Notebook on EDA with Visualization GitHub link:

[EDA with Visualization](#)

EDA with SQL

- Loaded data into an IBM DB2 instance
- SQL queries performed:
 1. Unique launch sites in the space mission
 2. First 5 records where launch sites begin with the string 'CCA'
 3. The total payload mass carried by boosters launched by NASA (CRS)
 4. The average payload mass carried by booster version F9 v1.1
 5. The date when the first successful landing outcome in ground pad was achieved
 6. The names of the boosters which have success landing in drone ship and have payload mass greater than 4000 but less than 6000
 7. The total number of successful and failure mission outcomes
 8. The names of the booster versions which have carried the maximum payload mass
 9. The failed landing outcomes in drone ship, their booster versions, and launch site names for in year 2015
 10. The ranking of the count of landing outcomes between 2010-06-04 and 2017-03-20, in descending order

Notebook on EDA with SQL GitHub link:

[EDA with SQL](#)

Build an Interactive Map with Folium

- Used an interactive visual analytics python library Folium.
- To get insights in geographical patterns about launch sites, performed visual analytics to determine:
 1. All launch sites on map
 2. The success/failed launches for each site on the map
 3. The distances between a launch site to its proximities
 - From Railways
 - From Highways
 - From Coastlines
 - From Cities



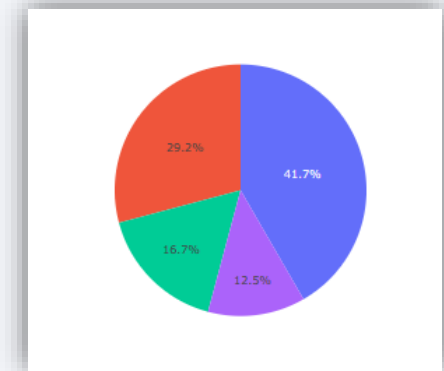
Launch Sites: CCAFS SLC- 40 and
Showing distance from railway, highway,
city and coastline

Notebook on Interactive Map with Folium GitHub link:

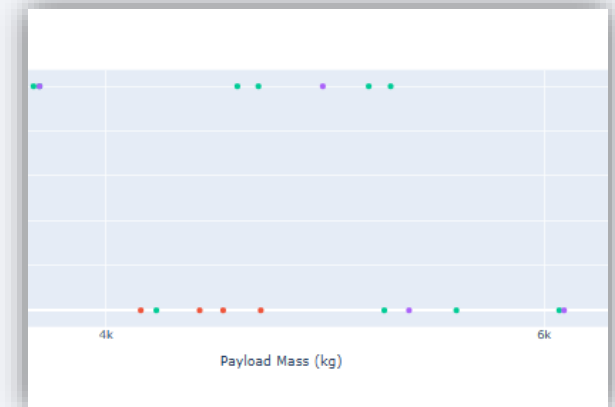
[Interactive Map with Folium](#)

Build a Dashboard with Plotly Dash

- Used a python interactive dashboard library “Plotly”.
- Build a Plotly dashboard application to perform interactive visual analytics on SpaceX launch data in real-time.
- The Interactive dashboard showcases:
 1. A Dropdown on the top, to select each launch site or all sites.
 2. A Pie chart showing success launches for all sites or each site
 3. A Slider to select the payload range in Kgs.
 4. A Scatter plot that shows the Success count on payload mass and outcome per booster for all sites or each site.



Success counts for all launch sites



Success counts on Payload Mass for all sites

Notebook on Dashboard with Plotly Dash GitHub link:

[Dashboard with Plotly Dash](#)

Predictive Analysis (Classification)

- To determine if the first stage of Falcon 9 will land successfully, a machine learning model and pipeline was built based on the previous data collected.
- Objectives:
 - ✓ Perform EDA and determine the training labels
 - ✓ Create a columns for Class
 - ✓ Standardize the data
 - ✓ Split into training and test data
 - ✓ Find the best hyperparameter
 - ✓ Find the method which performs best using test data.
- Classification models used:
 - Logistic Regression
 - Support Vector Machine
 - Decision Tree
 - K Nearest Neighbors



Notebook on Machine Learning Prediction Analysis GitHub link:
[Machine Learning Prediction](#)

Predictive Analysis (Flowchart)

Data Preparation and Model training and Evaluation Process using ML Classification models



Notebook on Machine Learning Prediction Analysis GitHub link:
[Machine Learning Prediction](#)

Predictive Analysis (Classification Models)

- The data was assigned to a variable called X after features engineering. The StandardScaler() method was used to standardize the data in this variable.
- The data from the "Class" column was assigned to a target variable (Y), which was then converted into a Numpyarray.
- Using the function train_test_split to split the data X and Y into training and test data with the parameter test_size to 0.2 and random_state to 2.
- Created Classification models objects for each models used.
- To find the best parameters, GridSearchCV() method was used and fit these parameters into the Classification objects. The best parameters for each model can be seen in the table.
- The accuracy of each models were evaluated using the Test data to determine the best models for this project.

Model	Best Parameters using GridSearchCV()
Logistic Regression	{'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
Support Vector Machine (SVM)	{'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
Decision Tree	{'criterion': 'gini', 'max_depth': 12, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 10, 'splitter': 'random'}
K-Nearest Neighbors (KNN)	{'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}

Notebook on Machine Learning Prediction Analysis GitHub link:
[Machine Learning Prediction](#)

Results

- Performed Exploratory Data Analysis to analyze which of the variables would have high impact in predicting if the falcon 9 first stage would land successfully.
- Performed feature engineering to determine high impacting feature variables such as Flight number, PayloadMass, Orbit, LaunchSite, Flights, GridFins, Reused, legs, LandingPads, serial.
- Created a landing outcome column “Class” which is the training label for prediction. This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully.
- Performed 4 Classification models which were tuned with hyperparameters, among which Decision Tree had a better result with the training set, with an Accuracy of 88%.
- Decision Tree had the best result among other models with the test set, with an Accuracy of 94%.

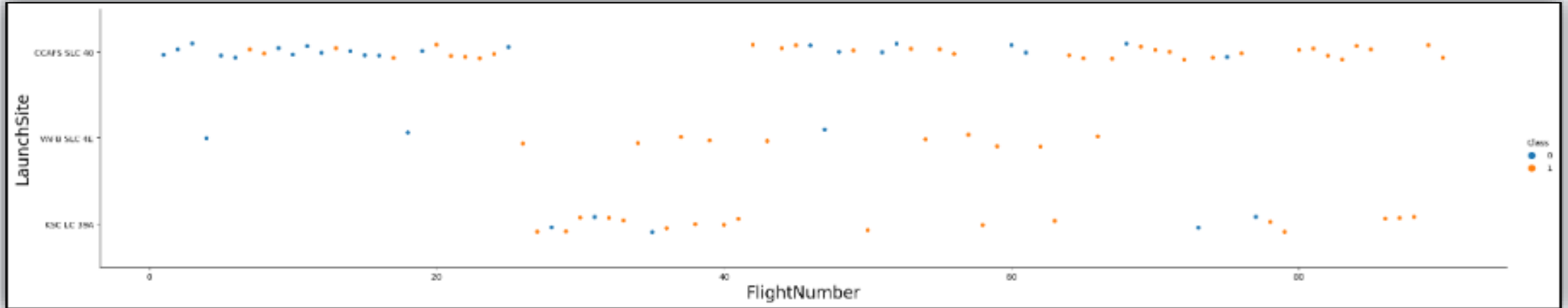
Model	Training Set Score	Test Set Score
Logistic Regression	0.84	0.83
Support Vector Machine (SVM)	0.84	0.83
Decision Tree	0.88	0.94
K-Nearest Neighbors (KNN)	0.84	0.83

The background of the slide is an abstract composition. It features a dark blue base color. Overlaid on this are numerous diagonal streaks in shades of red and cyan. A faint, light blue grid pattern is also visible, particularly in the lower-left quadrant. The overall effect is dynamic and technological.

Section 2

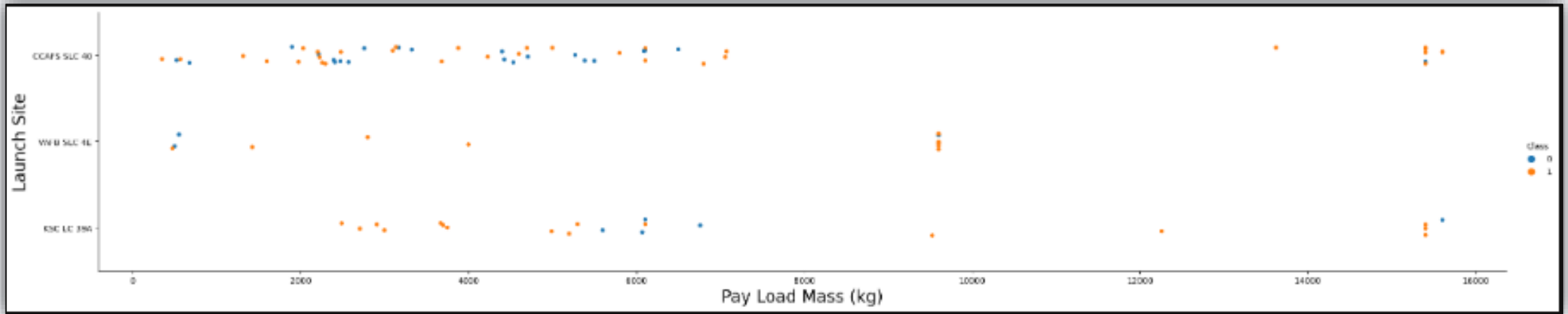
Insights drawn from EDA

Flight Number vs. Launch Site



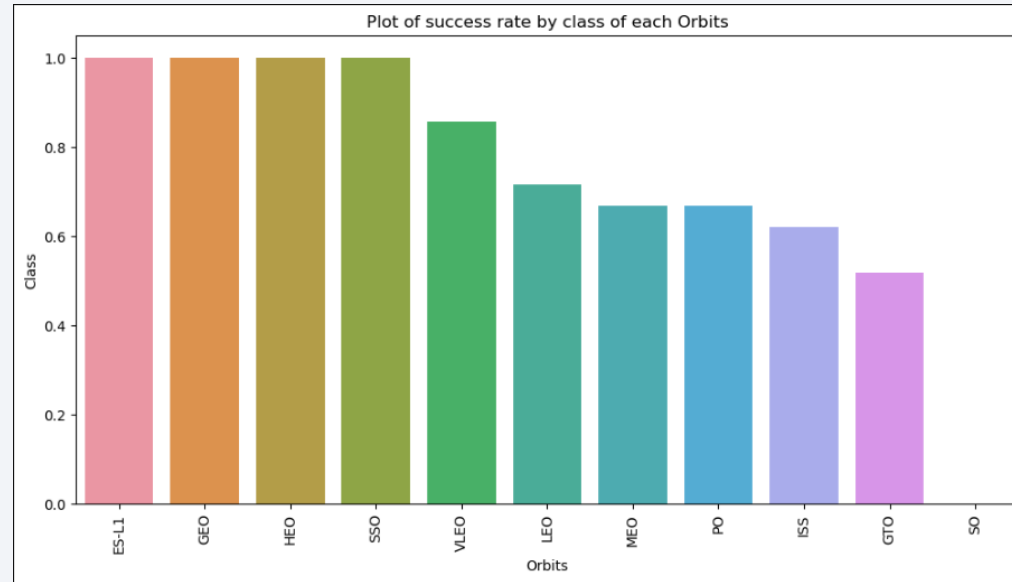
- Most of the launches were done at the location: CCAFS LC-40
- Most of the initial flights were done at the location “CCAFS LC-40”
- Launch site “CCAFS LC-40” seems to had higher success rate than failures
- As observed the gaps at a particular location was filled in with other launch sites, where “CCAFS LC-40” had the most number of launches, with “KSC LC 39A” with the second most and others at “VAFBS SLC 4E” with the least number of launches.

Payload vs. Launch Site



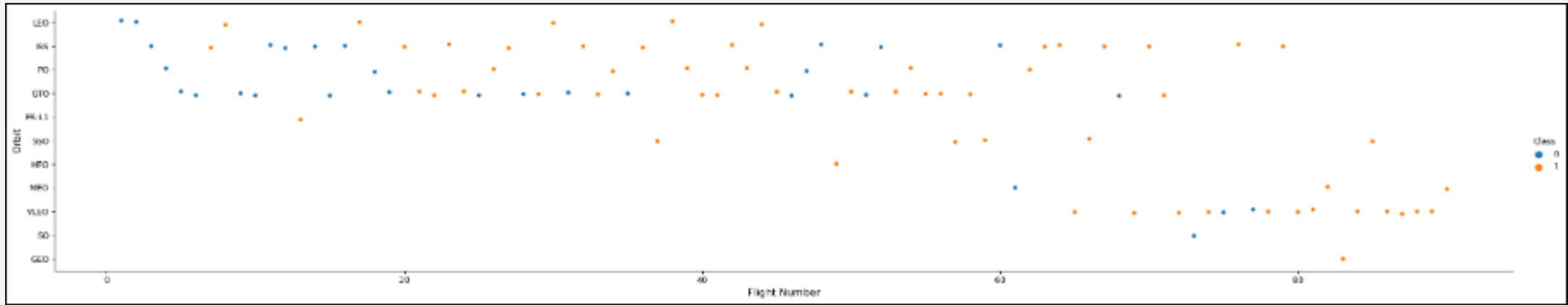
- This plot shows the launch sites with different payload Mass.
- Most of the launches were done at the location: CCAFS LC-40
- At launch site “VAFBS SLC 4E” there were little to no launches done as the Payload Mass increased above 4000kgs and only few were done after 9000kgs, where they were successful.
- No heavy payload mass flights were launched at VAFBS SLC 4E after 10000kgs.
- Most of the launches were less than 10000kgs.

Success Rate vs. Orbit Type



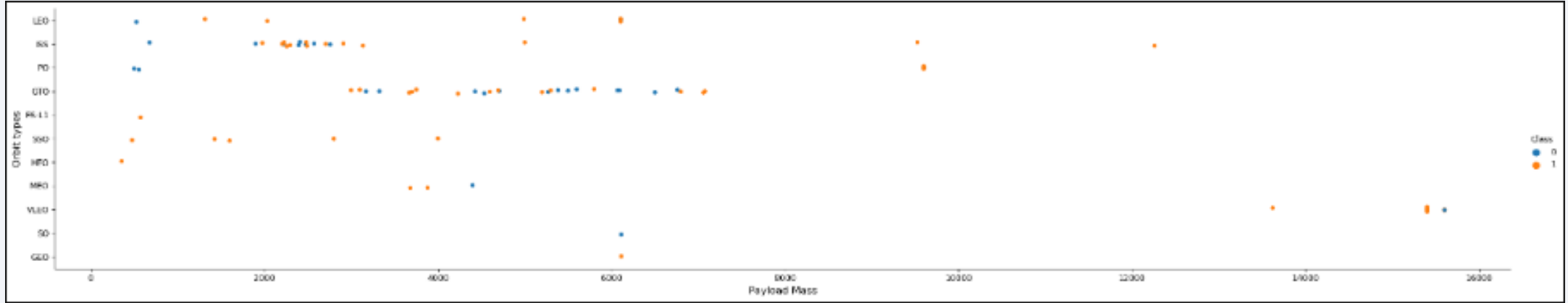
- This plot shows the success rate by class of each object.
- Orbits such as ES-L1, GEO, HEO, SSO had a success rate of the first stage landed successfully through all missions.
- “GTO” orbit had only half of successful landings whereas “SO” orbit had none.

Flight Number vs. Orbit Type



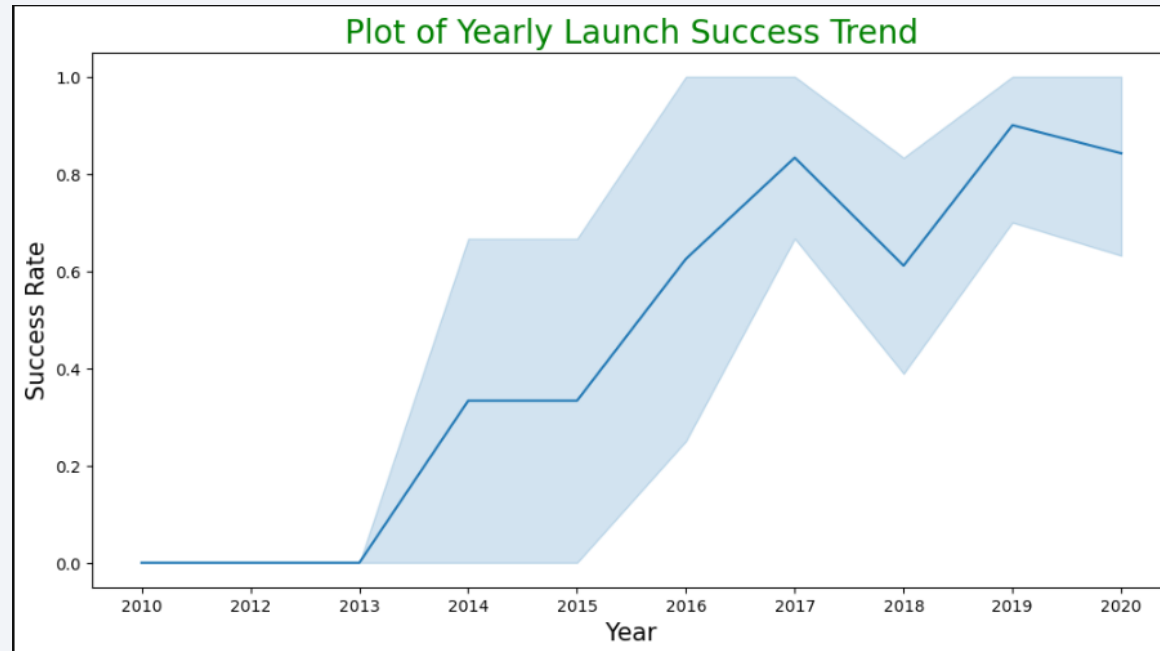
- This plot shows the relationship between the Flight number and each orbit.
- More than 60% of the flights were launched into LEO, ISS, PO and GTO orbits.
- Only after the 50th flight number, launches were started into different orbits as well.
- Among the first 50 flights, LEO orbit seems to have more mission success, whereas GTO orbit have more success missions after the 20th flight and higher number of failed missions of first successful landings.

Payload vs. Orbit Type



- This plot shows the relationship between the Payload Mass and each orbit.
- It appears that for Polar, Leo, and ISS orbits, the success rate of landings is better when the Payload mass exceeds 5000 kg.
- The same could not be seen for GTO because over the whole payload mass range, both successful and unsuccessful landing outcomes are present.

Launch Success Yearly Trend



- This plot shows the yearly Launch Success Trend.
- From the line chart, it appears that from 2013 till 2020, the success rate increased.
- From the chart, we also observe that there was atleast one unsuccessful first stage mission.

All Launch Site Names

- From the “SPACEXTBL”, a column LAUNCH_SITE contained multiple duplicate values. For our requirement, a **DISTINCT** method is used to return only distinct (different) values.
- Below is the query with **DISTINCT** method to select only distinct values as shown.

```
%sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL;
```



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

Launch Site Names Begin with 'CCA'

```
%sql SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```

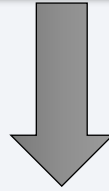
* sqlite:///my_data1.db
Done.

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS_KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

- From the “SPACEXTBL”, a **SELECT** method is used to retrieve records of tables from a database and a **LIMIT** method is used to limit the number of records to show based on a limit value.
- A **LIKE** command is used in a **WHERE** method to return a pattern.

Total Payload Mass

```
%sql SELECT SUM(payload_mass_kg_) FROM SPACEXTBL WHERE CUSTOMER = 'NASA (CRS)'
```

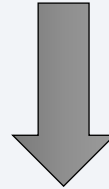


SUM(payload_mass_kg_)
45596

- The **Sum()** function returns the total sum of a numeric column.

Average Payload Mass by F9 v1.1

```
%sql SELECT AVG(payload_mass_kg_) FROM SPACEXTBL WHERE Booster_Version = 'F9 v1.1';
```

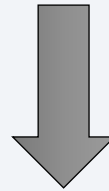


AVG(payload_mass_kg_)
2928.4

- The **Average()** function returns the average of a numeric column.
- This **AVG()** functions returns the average of the Payload Mass of the Booster v1.1 to be 2928.4 kgs.

First Successful Ground Landing Date

```
%sql SELECT MIN(DATE) FROM SPACEXTBL WHERE mission_outcome = 'Success';
```




MIN(DATE)
01-03-2013

- The **Min()** function returns the smallest value of a numeric column.
- This **Min(DATE)** function returns the first date of a success landing to be 01-03-2013 where the outcome of the mission is a Success.

Successful Drone Ship Landing with Payload between 4000 and 6000

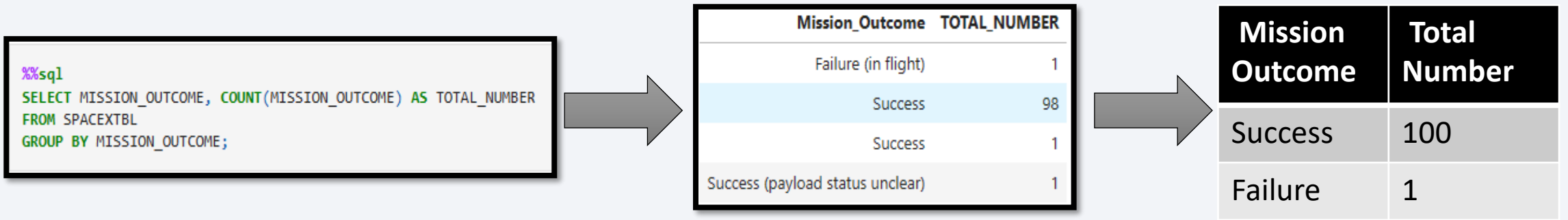
```
%sql SELECT booster_version FROM SPACEXTBL  
WHERE "Landing_Outcome" = 'Success (drone ship)'  
and payload_mass__kg_ BETWEEN 4000 AND 6000;
```



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

- From the table **SPACEXTBL**, we **SELECT** the Booster Version column and show the only records **WHERE** the Landing outcome is 'Success(drone ship)' **AND** the payload mass is **BETWEEN** 4000 and 6000 kgs.
- Use **WHERE**, **AND**, **BETWEEN** methods statements to perform the condition required.

Total Number of Successful and Failure Mission Outcomes



- From the table SPACEXTBL, we **SELECT** the Mission_Outcome column and the **COUNT** of the column “mission_outcome” which is renamed as TOTAL_NUMBER and grouped by the Mission_outcome column to return the required solution.
- The **GROUP BY** statement groups rows that have the same values into summary rows
- Use **COUNT** method to return a count of the column “mission_outcome”.

Boosters Carried Maximum Payload

```
%%sql
SELECT DISTINCT BOOSTER_VERSION
FROM SPACEXTBL
WHERE PAYLOAD_MASS_KG_ = (
    SELECT MAX(PAYLOAD_MASS_KG_)
    FROM SPACEXTBL);
```



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

- From the table SPACEXTBL, we **SELECT** the **DISTINCT** of Booster Version **WHERE** the payload mass is the **MAX**(largest) of payload mass.
- Here, we use a subquery to determine the largest value of payload mass using the **Max()** method.
- Use **DISTINCT** method to select only distinct values of a column.

2015 Launch Records

```
%%sql
SELECT "Landing _Outcome", BOOSTER_VERSION, LAUNCH_SITE
FROM SPACEXTBL
WHERE "Landing _Outcome" = 'Failure (drone ship)' AND DATE LIKE '%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

- From the table SPACEXTBL, we use the combinations of **WHERE**, **AND** and **LIKE** conditions to return the booster version's failure landing outcomes at launch sites in the year 2015.

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

```
%%sql
SELECT "Landing _Outcome", COUNT("Landing _Outcome") as COUNT
FROM SPACEXTBL
WHERE DATE BETWEEN '04-06-2010' AND '20-03-2017'
GROUP BY "Landing _Outcome"
ORDER BY COUNT DESC;
```



Landing _Outcome	COUNT
Success	20
No attempt	10
Success (drone ship)	8
Success (ground pad)	6
Failure (drone ship)	4
Failure	3
Controlled (ocean)	3
Failure (parachute)	2
No attempt	1

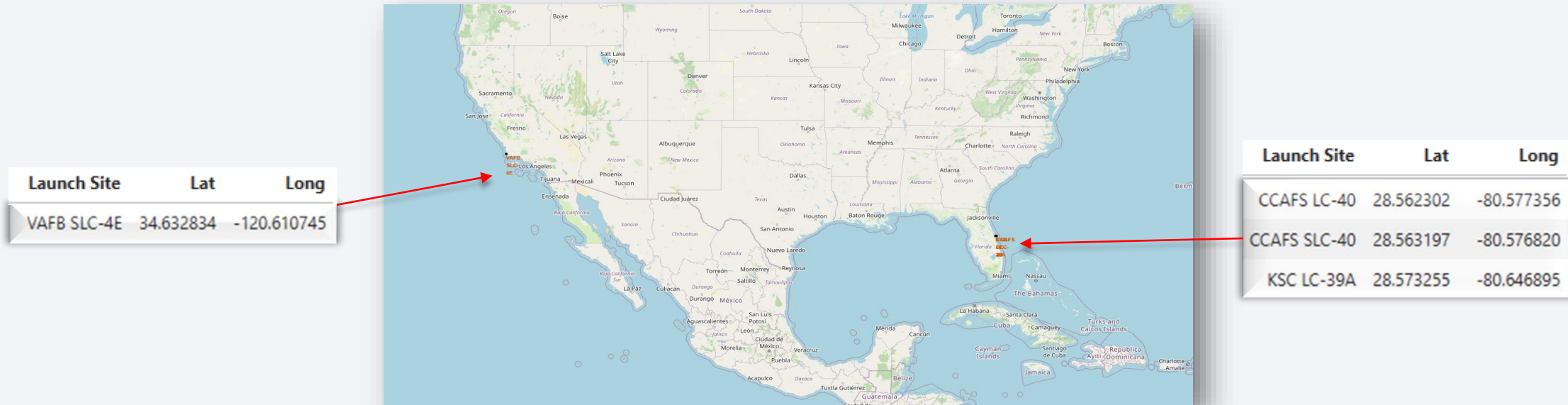
- From the table SPACEXTBL, we use the various combinations such as **COUNT**, **WHERE**, **AND**, **BETWEEN**, **GROUP BY** and **ORDER BY** conditions to rank the landing outcomes between 2010-06-04 and 2017-03-20 with its count.
- To return the details in descending order, we use **DESC** method.

A satellite view of Earth from space, showing the curvature of the planet and city lights at night. The background is a deep blue gradient.

Section 3

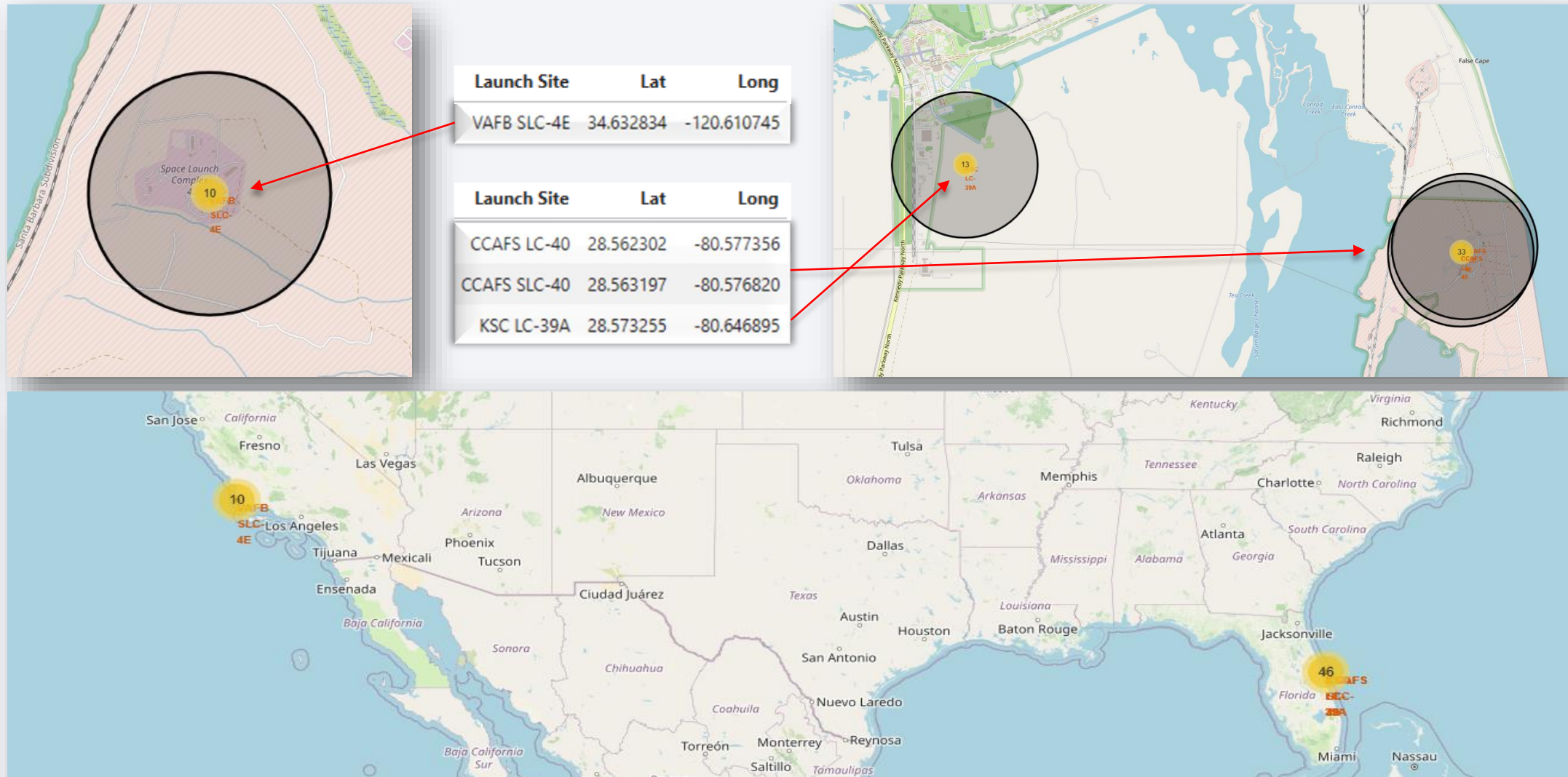
Launch Sites Proximities Analysis

Folium Map: Launch Site Locations



- An interactive folium map object was created with addition of the all the launch site locations. The latitude and longitude of all the launch site locations were included in the dataset.
- Add a folium.Circle and folium.Marker for each launch site on the folium map.
- The launch sites were on the east and west coast of United States.
- This interactive map can be zoomed in/out to see all the launch site locations.

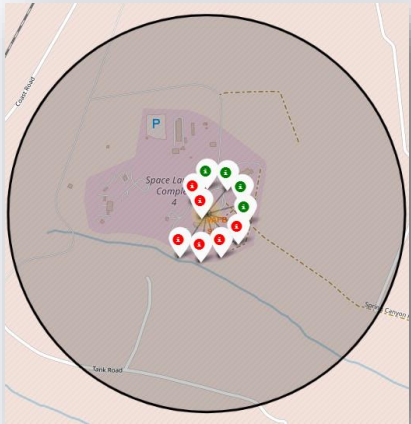
Folium Map: Launch Site Locations Color Labeled



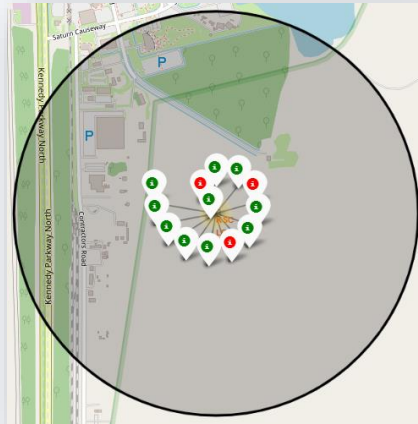
Folium Map: Launch Site Locations Color Labeled

- Added markers and circles to each site locations.
- If launch was successful, we show **Green markers**.
- If launch was unsuccessful, we show **Red markers**.
- We can see that KSC LC-39A had higher successful rate than others.

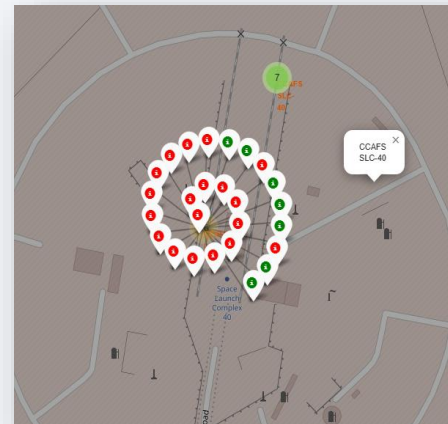
VAFB SLC-4E



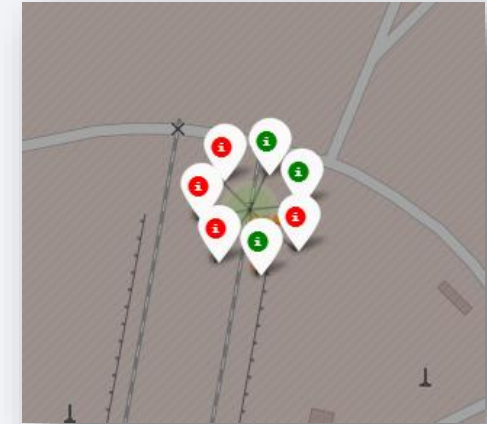
KSC LC-39A



CCAFS LC-40

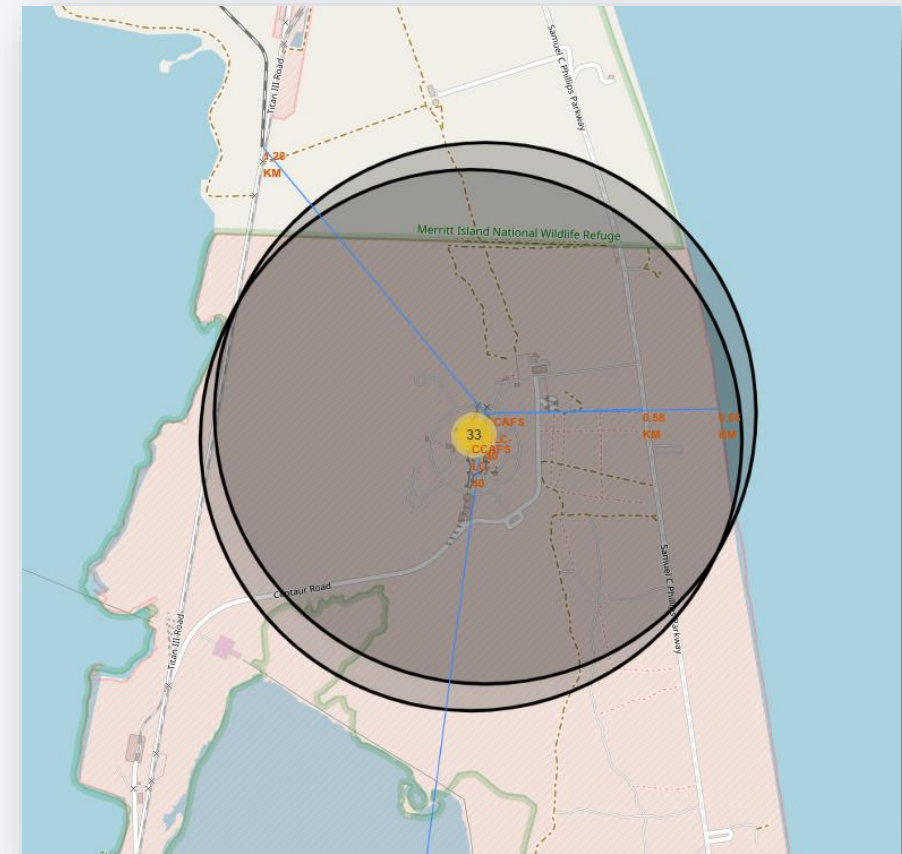


CCAFS SLC-40



Folium Map: Launch Site Distance Analysis

- Added a **Blue** line Marker to calculate the distance of the site locations from railways, highways, coastline and cities.
- For example: CCAFS SLC-40,
 - Railway: 1.28km
 - Highway: 0.58km
 - Coastline: 0.86km
 - City: 51.4km
- From this data, we analyze that all launch sites are **not closer** to the railways, highways, coastlines and cities.



CCAFS SLC-40



Section 4

Build a Dashboard with Plotly Dash

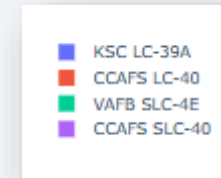
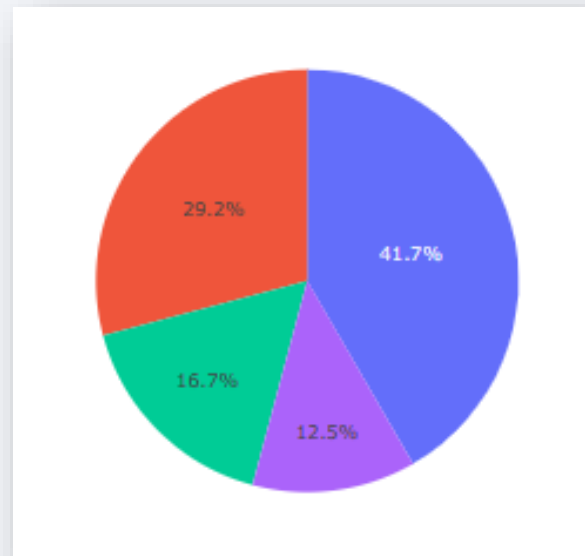
Plotly Dashboard App

All Sites



Drop down to select all/each sites

Success count for all sites

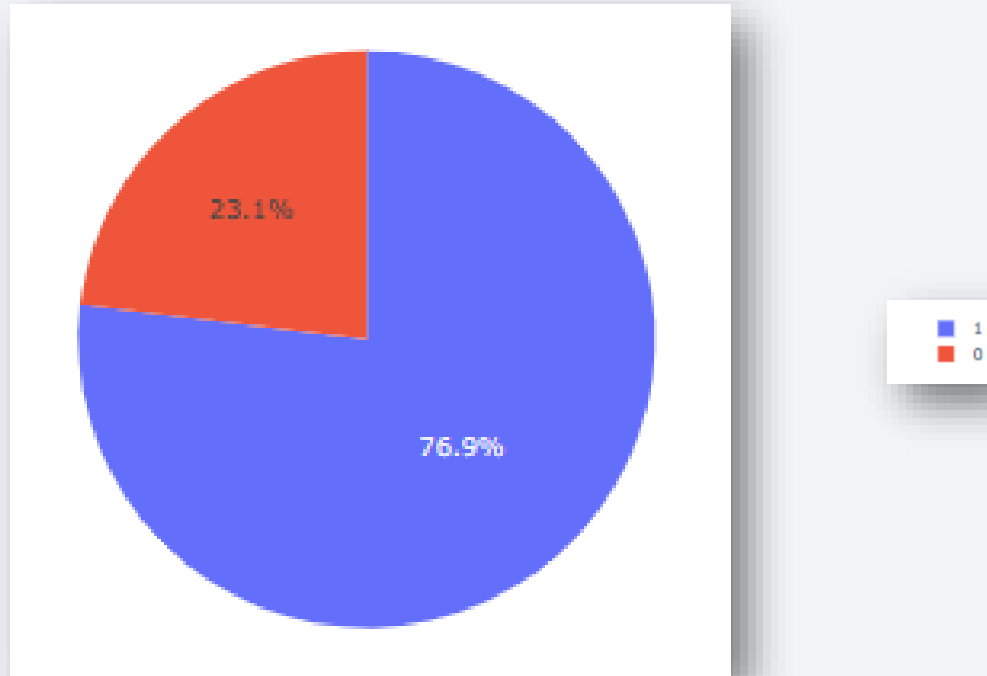


Color coded sites

- As shown above, we see the **KSC LC-39A** location has the most successful launches for all sites followed by CCAFS LC-40, VAFB SLC-4E and CCAFS SLC-40

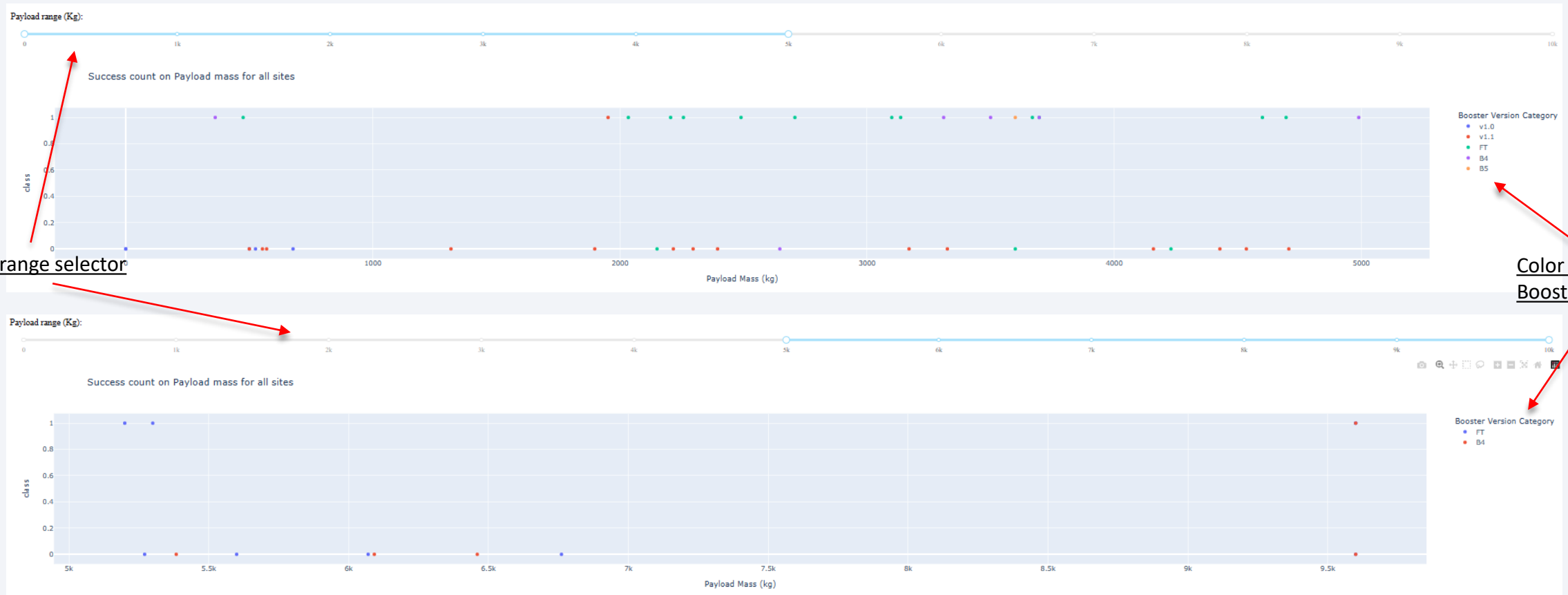
Plotly Dashboard App: **KSC LC-39A** Highest launch success ratio

KSC LC-39A



- As shown in the piechart, we see the **KSC LC-39A** location achieved a 76.9% success rate with just 23.1% failure rate.

Plotly Dashboard App: Payload vs Launch Outcome scatter plot for All Sites



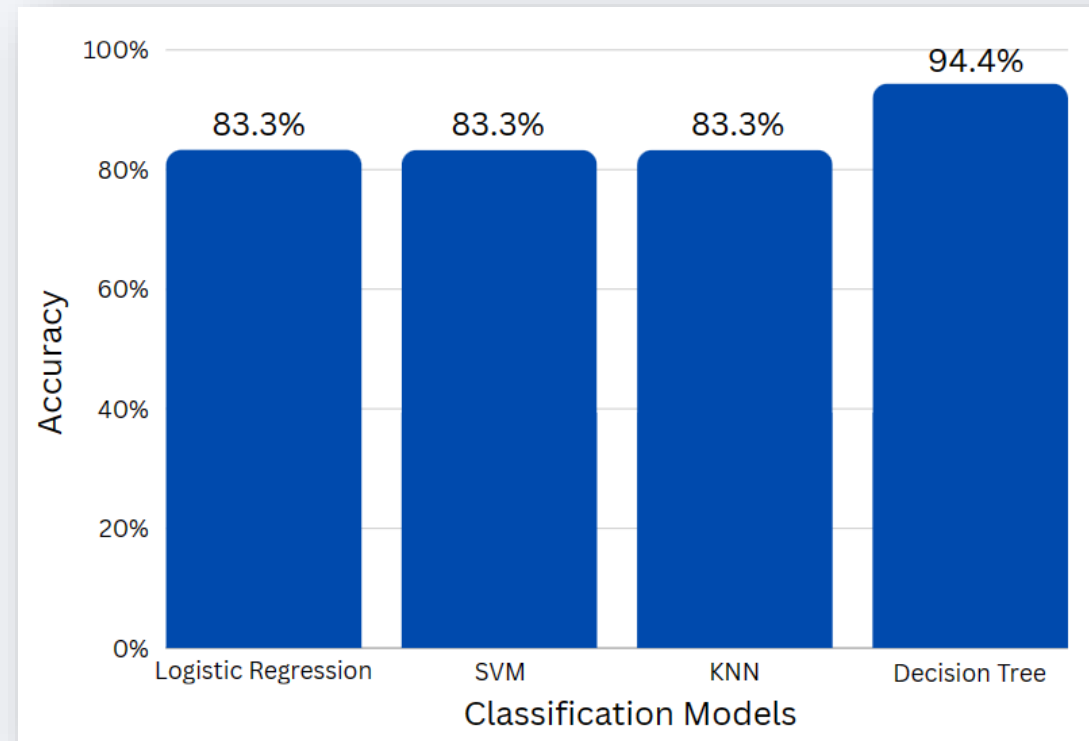
Lower payload ranges (<5000kgs) have **higher success rate** than higher payloads (> 5000kgs)



Section 5

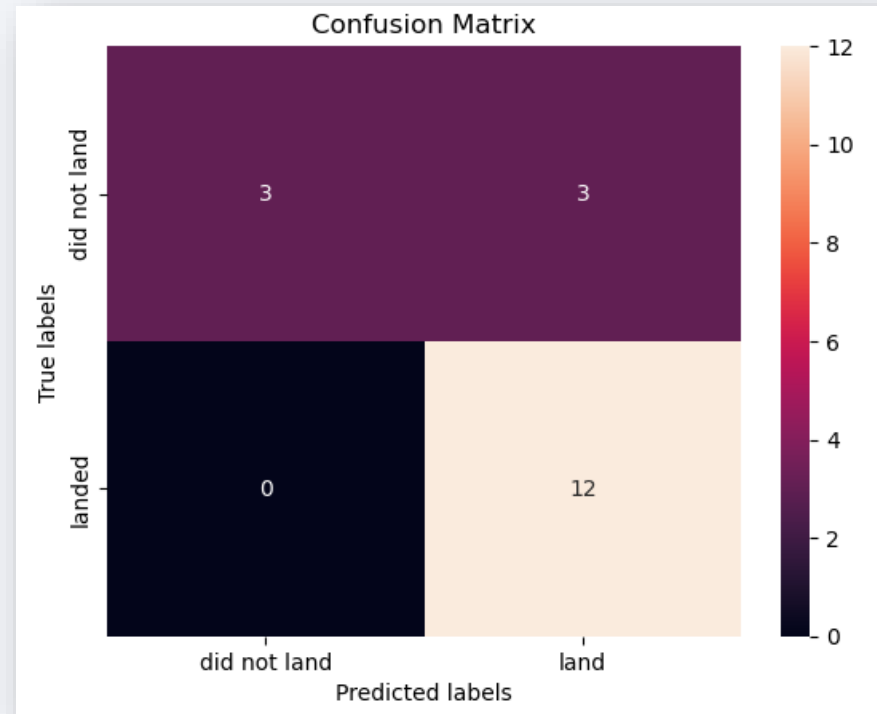
Predictive Analysis (Classification)

Classification Accuracy



Among the 4 classification models used (Logistic Regression, SVM, Decision Tree and KNN), **Decision Tree** model has the highest classification accuracy of **94%** on test data.

Confusion Matrix : Best Model (Decision Tree)



The Confusion matrix of the Decision Tree model shows the landing success where the “Landed” true label with “land” prediction label gives a True positive of 12.

Although, the confusion matrix of all the 4 classifications were same.

Conclusion

- The report showcases data analysis of Falcon 9 launches and uses a machine learning model that predicts if the Falcon 9 SpaceX's first stage will land successfully or not.
- Space X statement stated that 1st stage booster costs 15 million to build.
- Based on this report, Space Y can predict if the first stage of Falcon 9 will land successfully or not and determine the cost of the launch.
- Taking into all these considerations and factors that impact the success of first stage landing, the data tells us that:
 - KSC LC-39A location has the most successful launches with a success rate of 76.9%
 - ES-L1, GEO, HEO, SSO had a success rate of the first stage landed successfully through all missions.
 - CCAFS LC-40 site locations was the most used, where higher payloads above 6000kgs have higher success rate.
 - Machine learning model “Decision Tree” is the best for this case with a accuracy rate of 94.4%

GitHub links - Notebooks

- [IBM-DataScience-Capstone-SpaceX-Falcon9/IBM Capstone Data Collection API Week1.ipynb at master · syedkhaleelullah7/IBM-DataScience-Capstone-SpaceX-Falcon9 · GitHub](#)
- [IBM-DataScience-Capstone-SpaceX-Falcon9/IBM Capstone Data Collection w Web Scraping Week1.ipynb at master · syedkhaleelullah7/IBM-DataScience-Capstone-SpaceX-Falcon9 · GitHub](#)
- [IBM-DataScience-Capstone-SpaceX-Falcon9/IBM Capstone EDA Data Wrangling Week1.ipynb at master · syedkhaleelullah7/IBM-DataScience-Capstone-SpaceX-Falcon9 · GitHub](#)
- [IBM-DataScience-Capstone-SpaceX-Falcon9/IBM Capstone EDA with Visualization Week2.ipynb at master · syedkhaleelullah7/IBM-DataScience-Capstone-SpaceX-Falcon9 · GitHub](#)
- [IBM-DataScience-Capstone-SpaceX-Falcon9/IBM Capstone EDA with SQL Week2.ipynb at master · syedkhaleelullah7/IBM-DataScience-Capstone-SpaceX-Falcon9 · GitHub](#)
- [IBM-DataScience-Capstone-SpaceX-Falcon9/IBM Capstone Vizual Analytics with Folium Week3.ipynb at master · syedkhaleelullah7/IBM-DataScience-Capstone-SpaceX-Falcon9 · GitHub](#)
- [IBM-DataScience-Capstone-SpaceX-Falcon9/IBM Capstone Machine Learning Prediction Week4.ipynb at master · syedkhaleelullah7/IBM-DataScience-Capstone-SpaceX-Falcon9 · GitHub](#)
- [IBM-DataScience-Capstone-SpaceX-Falcon9/spacex_dash_app.py at master · syedkhaleelullah7/IBM-DataScience-Capstone-SpaceX-Falcon9 · GitHub](#)

Thank you!

