IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Susmitha Babu Vishnu
25/12/2023

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 Analytics with Folium
  - Machine Learning Prediction
- Summary of all results
  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - Predictive Analytics result

Introduction

- Project background and context
  - SpaceX is a ground-breaking corporation that has completely changed the space sector by providing rocket launches, notably Falcon 9, for as little as $62 million, compared to other suppliers that charge as much as $165 million for each launch.
  - The majority of these savings are attributable to SpaceX's brilliant concept to re-land the rocket after the first stage of the launch so that it can be used on a later flight. The price will drop considerably more if this practice is repeated.
  - We, as data scientists in SpaceY, use this data to compete with SpaceX.
- Problems you want to find answers
  - Factors determining if the rocket will land successfully
  - Interaction amongst various features that determine the success rate of a successful landing
  - Operating conditions which needs to be in place to ensure a successful landing program
- Github Link of the Main branch with all files and codes:

https://github.com/Susmithavishnu/Applied-Data-Science-Capstone-Project/tree/main

Section 1

Methodology

Methodology

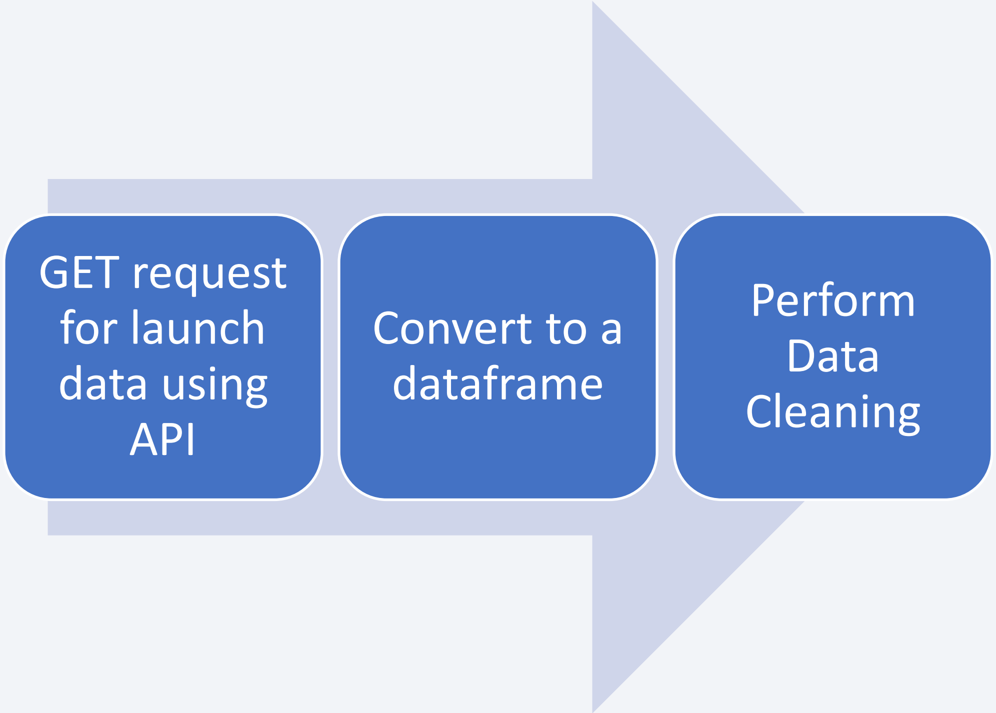
Executive Summary

- 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 for categorical features
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - The collected data were normalized, divided into training and test sets and evaluated by four different classification models with their accuracies.

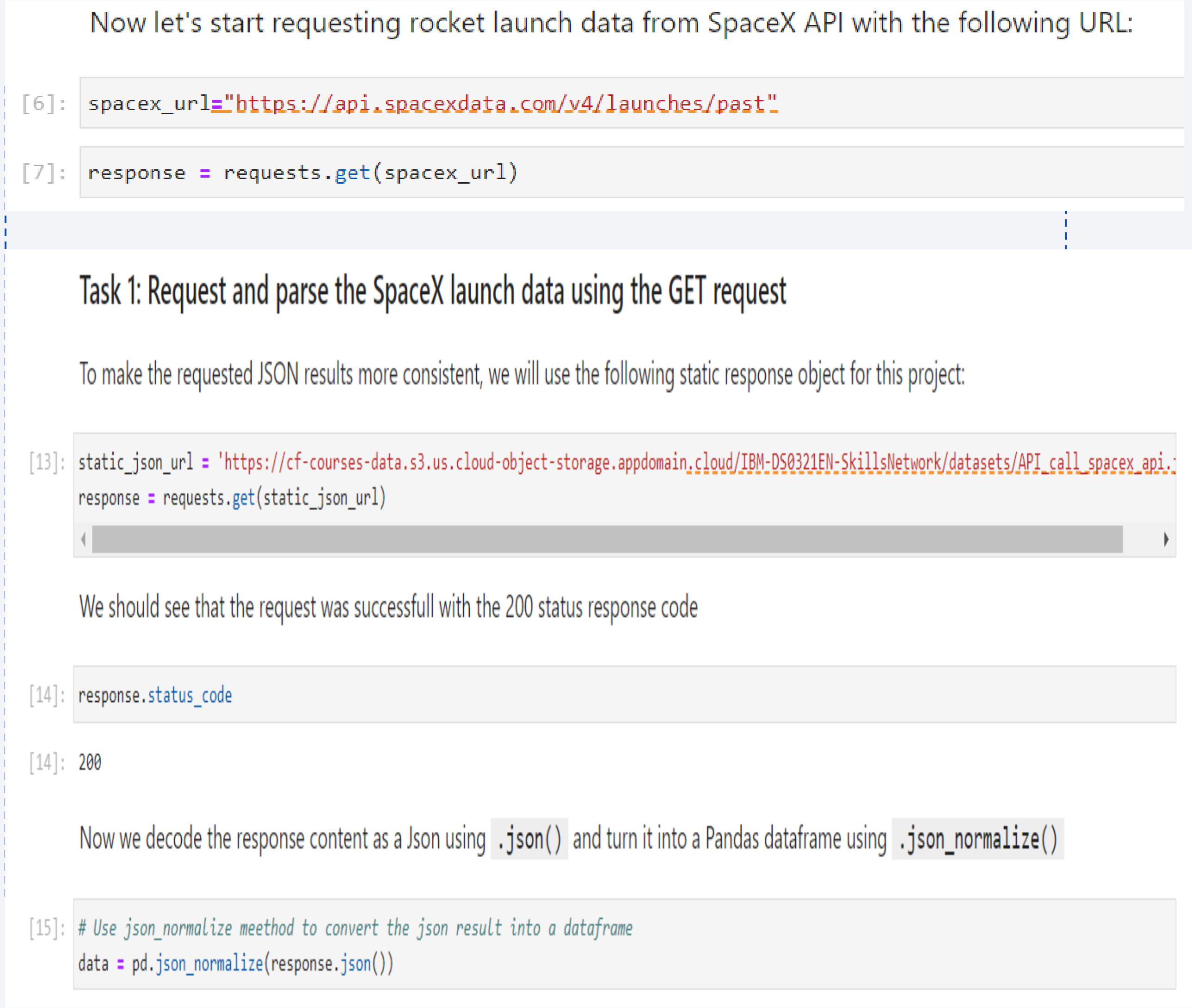
Data Collection

- Data sets were collected from:
  - Space X API (https://api.spacexdata.com/v4/rockets/) &
  - Wikipedia (https://en.wikipedia.org/wiki/List_of_Falcon/_9/_and_Falcon_Heavy_launches), using web scraping techniques.
- Data has been collected using the below steps -
  - Collected the data using GET request to the SpaceX API
  - Decoded the response content as a JSON using .json() function call and converted it into a pandas dataframe using .json_normalize()
  - Data cleaning was done. Checked for missing values and filled them wherever necessary
  - Web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup

Data Collection – SpaceX API



- Github Link: https://github.com/Susmithavishnu/Applied-Data-Science-Capstone-Project/blob/main/jupyter-labs-spacex-data-collection-api.ipynb



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

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

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

Task 1: Request and parse the SpaceX launch data using the GET request

To make the requested JSON results more consistent, we will use the following static response object for this project:

```
[13]: static_json_url = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.
      response = requests.get(static_json_url)
```

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

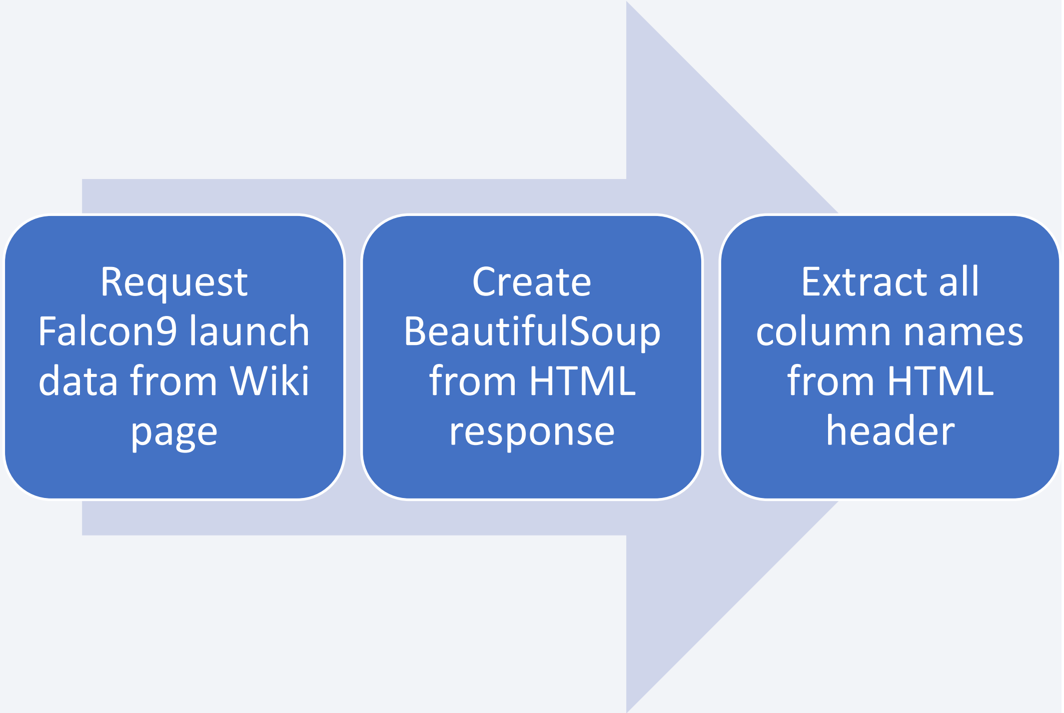
```
[14]: response.status_code
```

```
[14]: 200
```

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

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

Data Collection - Scraping



- Github Link: https://github.com/Susmithavishnu/Applied-Data-Science-Capstone-Project/blob/main/jupyter-labs-webscraping.ipynb

TASK 1: Request the Falcon9 Launch Wiki page from its URL

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

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

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

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

[9]:
```
# 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')
```

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

Next, we just need to fill up the `launch_dict` with launch records extracted from table rows.

Data Wrangling

- Data Wrangling is the process of cleaning and unifying messy and complex data sets for easy access and Exploratory Data Analysis (EDA).
- We will first calculate the number of launches on each site, then calculate the number and occurrence of mission outcome per orbit type.
- We then create a landing outcome label from the outcome column. This will make it easier for further analysis, visualization, and ML. Lastly, we will export the result to a CSV
- Github Link: https://github.com/Susmithavishnu/Applied-Data-Science-Capstone-Project/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

Use the method `value_counts()` on the column `LaunchSite` to determine the number of launches on each site:

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

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

Use the method `.value_counts()` to determine the number and occurrence of each orbit in the column `Orbit`

```
# Apply value_counts on Orbit column
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
```

Data Wrangling

Use the method `.value_counts()` on the column `Outcome` to determine the number of `landing_outcomes`.Then assign it to a variable landing_outcomes.

[9]:
```
# landing_outcomes = values on Outcome column
df['Outcome'].value_counts()
```

[9]:
```
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
```

[12]:
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = df['Outcome'].map(lambda x: 0 if x in bad_outcomes else 1)
```

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

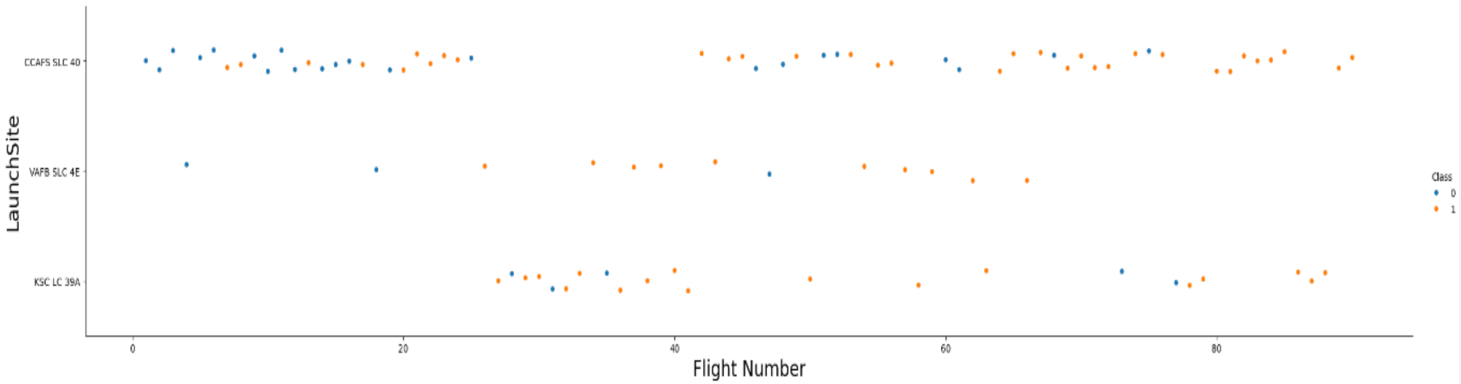
[13]:
```
df['Class']=landing_class
df[['Class']].head(8)
```

[13]:

	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

EDA with Data Visualization

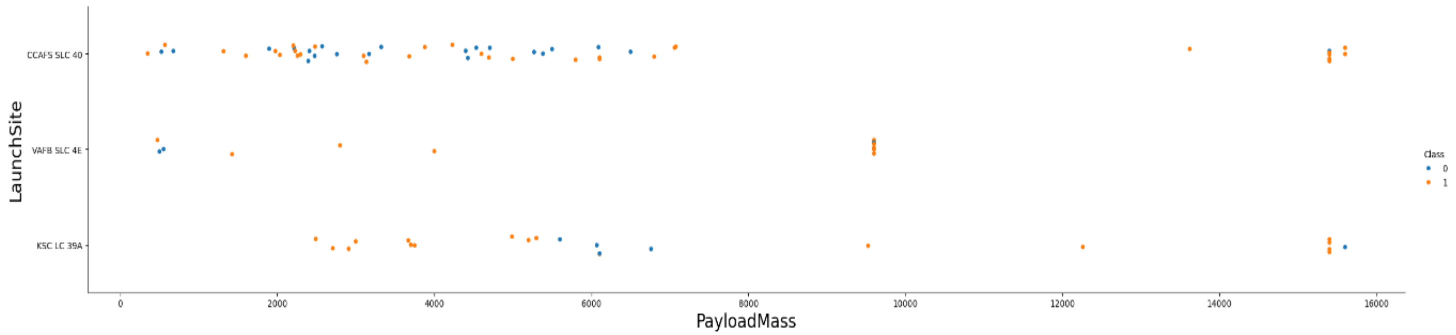
[8]:
```python
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("LaunchSite",fontsize=20)
plt.show()
```



- Flight Number x Launch Site

  scatter graph

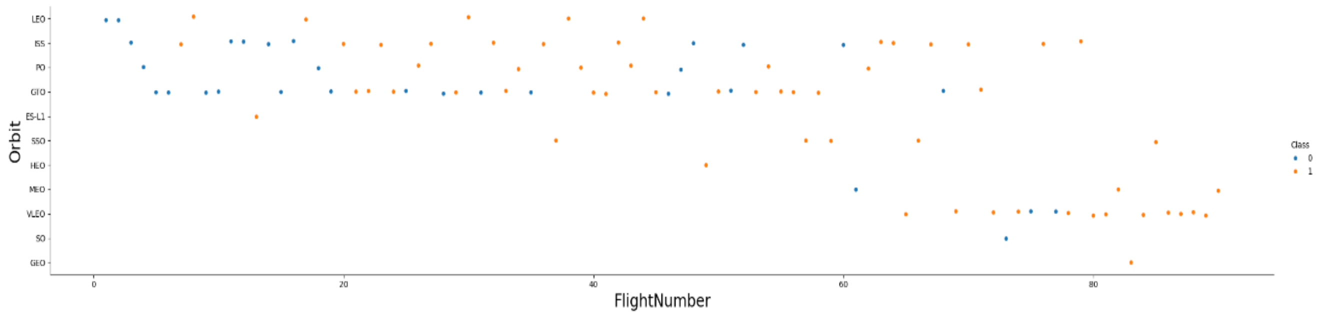
[9]:
```python
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("PayloadMass",fontsize=20)
plt.ylabel("LaunchSite",fontsize=20)
plt.show()
```



- Payload Mass x Launch Site scatter graph

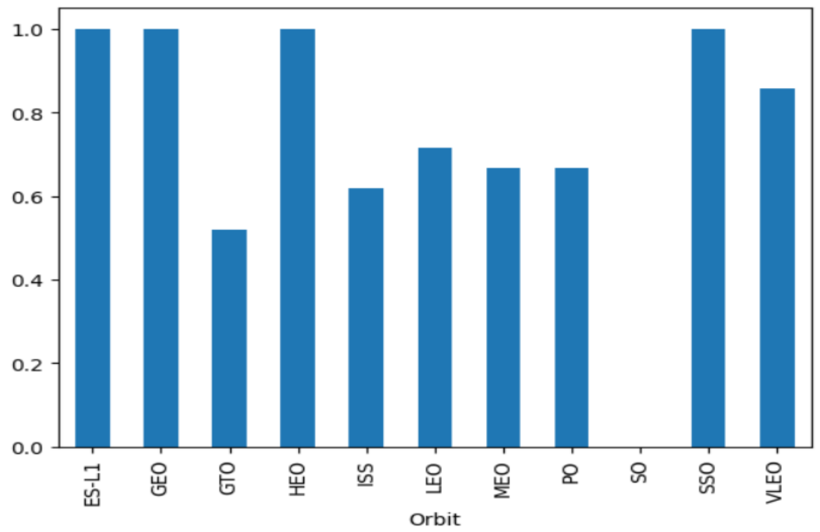
EDA with Data Visualization

```
[11]: # Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("FlightNumber",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



- Flight number x Orbit Scatter plot

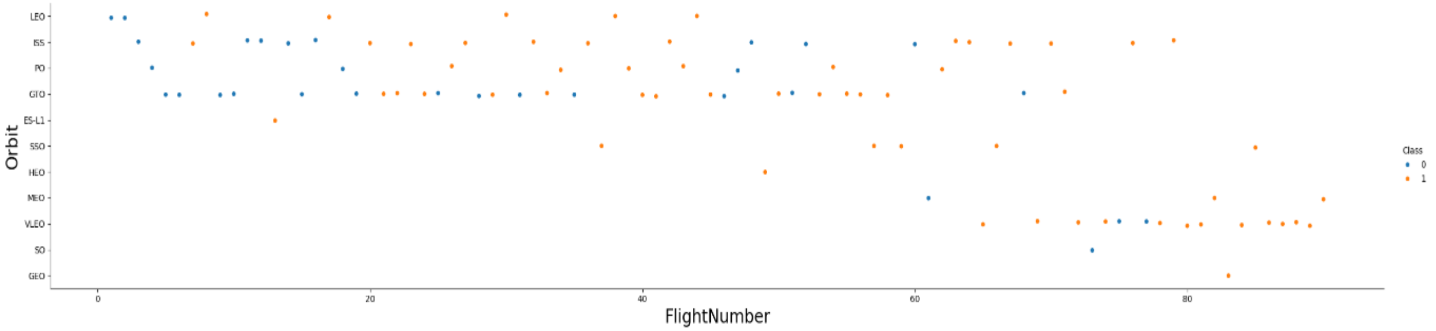
```
[10]: # HINT use groupby method on Orbit column and get the mean of Class column
df.groupby('Orbit')['Class'].mean().plot.bar()

[10]: <AxesSubplot:xlabel='Orbit'>
```



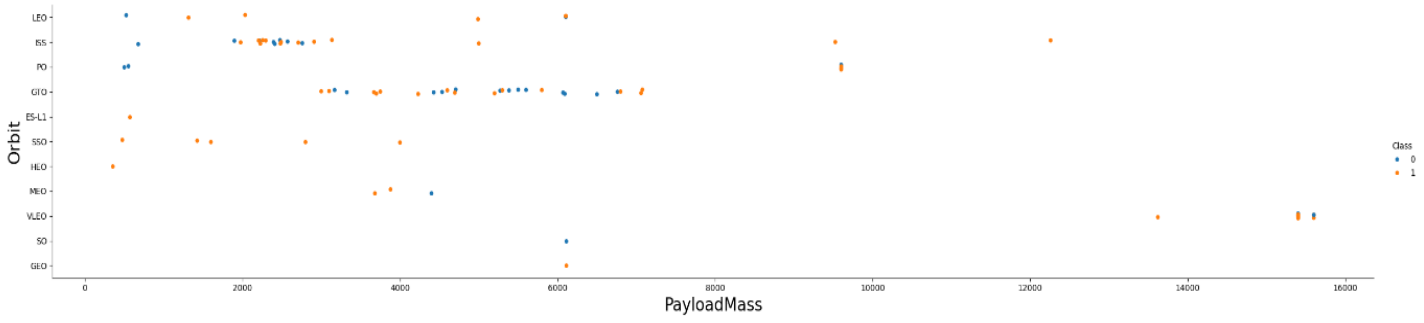
- Bar Chart showing success rate of each orbit

EDA with Data Visualization

```
[11]: # Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("FlightNumber",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```




```
[12]: # Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("PayloadMass",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



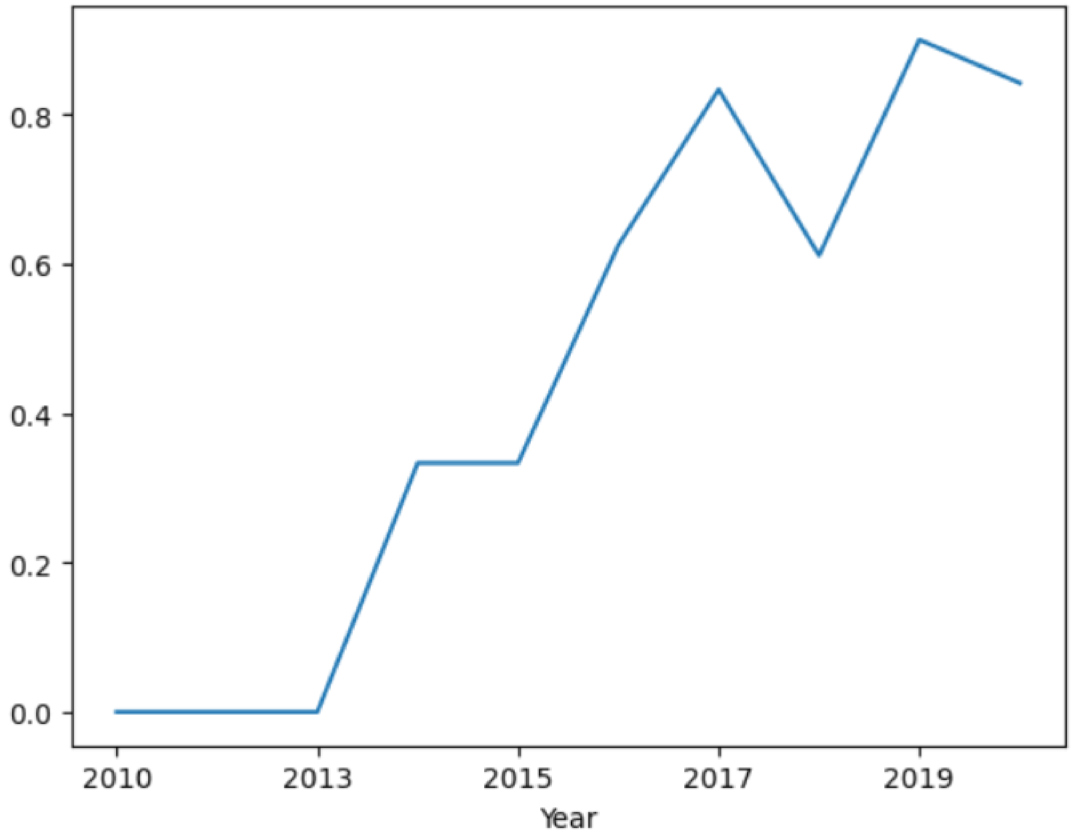
- Flight Number x Orbit scatter plot

- Payload Mass x Orbit scatter plot

EDA with Data Visualization

```
[14]: # Plot a line chart with x axis to be the extracted year and y axis to be the success rate
      temp_df = df.copy()
      temp_df['Year'] = year
      temp_df.groupby('Year')['Class'].mean().plot()
```

```
[14]: <AxesSubplot:xlabel='Year'>
```



- Line chart which plots the year against the success rate
- Github Link for all the Data Viz. graphs: https://github.com/Susmithavishnu/Applied-Data-Science-Capstone-Project/blob/main/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA with SQL

EDA was done with SQL queries for the following prompts

- 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 Link: https://github.com/Susmithavishnu/Applied-Data-Science-Capstone-Project/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- To visualize the launch data into an interactive map. We took 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 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's formula to calculated the distance of the launch sites to various landmark to find answer to the questions of:
  - How close the launch sites with railways, highways and coastlines?
  - How close the launch sites with nearby cities?

  - Github Link: https://github.com/Susmithavishnu/Applied-Data-Science-Capstone-Project/blob/main/lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash which allowing the user to play around with the data as they need.
  - We plotted pie charts showing the total launches by a certain sites.
  - We then plotted scatter graph showing the relationship with Outcome and Payload
- Mass (Kg) for the different booster version.
- Github Link: https://github.com/Susmithavishnu/Applied-Data-Science-Capstone-Project/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

Building the Model

- Transform the data and then split into training and test datasets
- Decide which type of ML algorithms to use
- Set the parameters and algorithms to GridSearchCV and fit it to dataset.

Evaluating the Model

- Check the accuracy for each model
- Get tuned hyperparameters

for each type of algorithms.

- plot the confusion matrix

Improving the Model

- Use Feature Engineering

and Algorithm Tuning

Find the Best Model

Github Link: https://github.com/Susmithavishnu/Applied-Data-Science-Capstone-Project/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Results

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

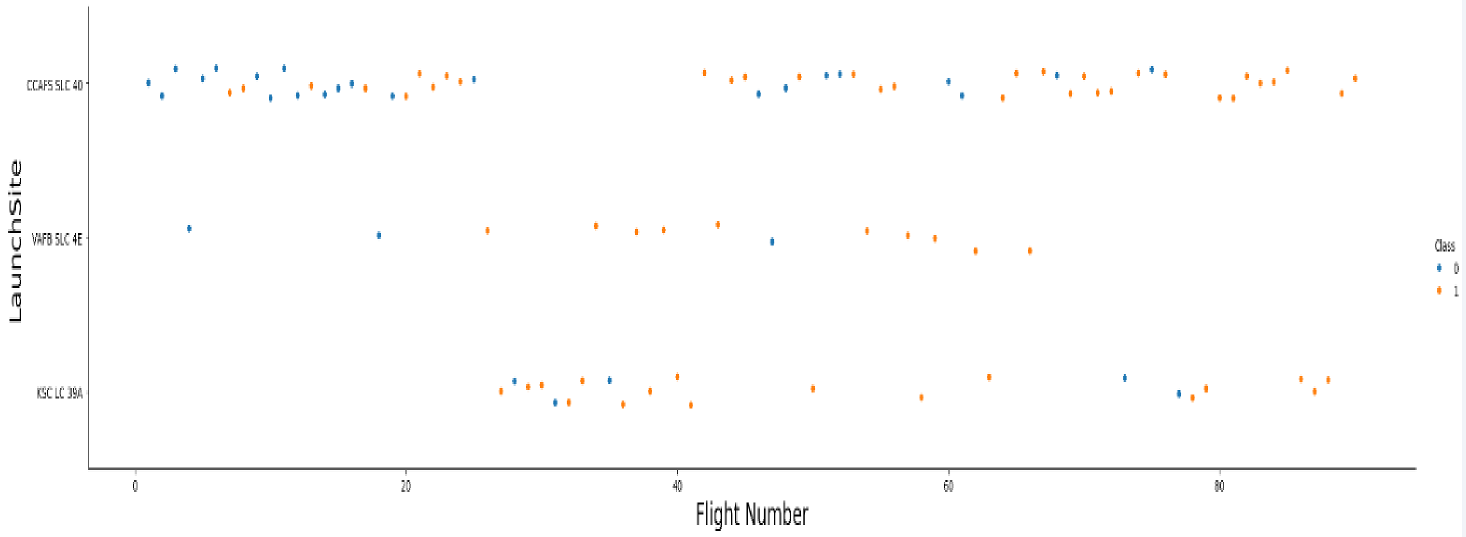
Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

- Larger the flights amount of the launch site, the greater the the success rate will be.
- CCAFS SLC40 shows the least pattern of this. i.e, lesser success rate

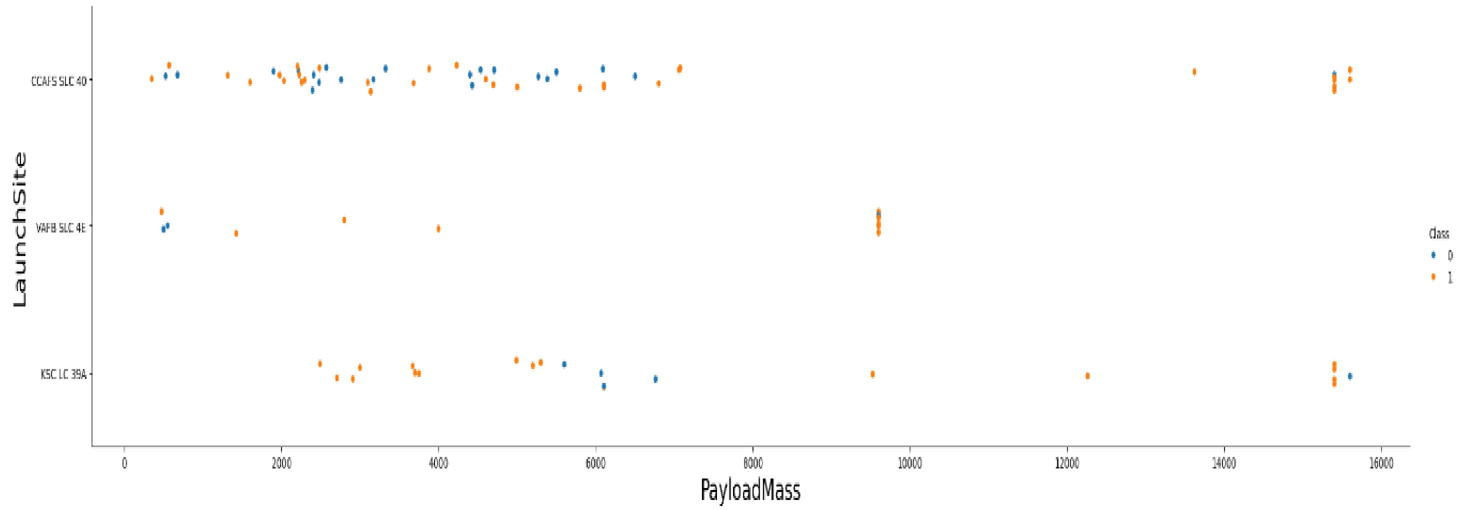
```
[8]: # Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("LaunchSite",fontsize=20)
plt.show()
```



Payload vs. Launch Site

- Once the pay load mass is greater than 7000kg, the probability of the success rate will be highly increased.
- But there is no significant pattern to compare these

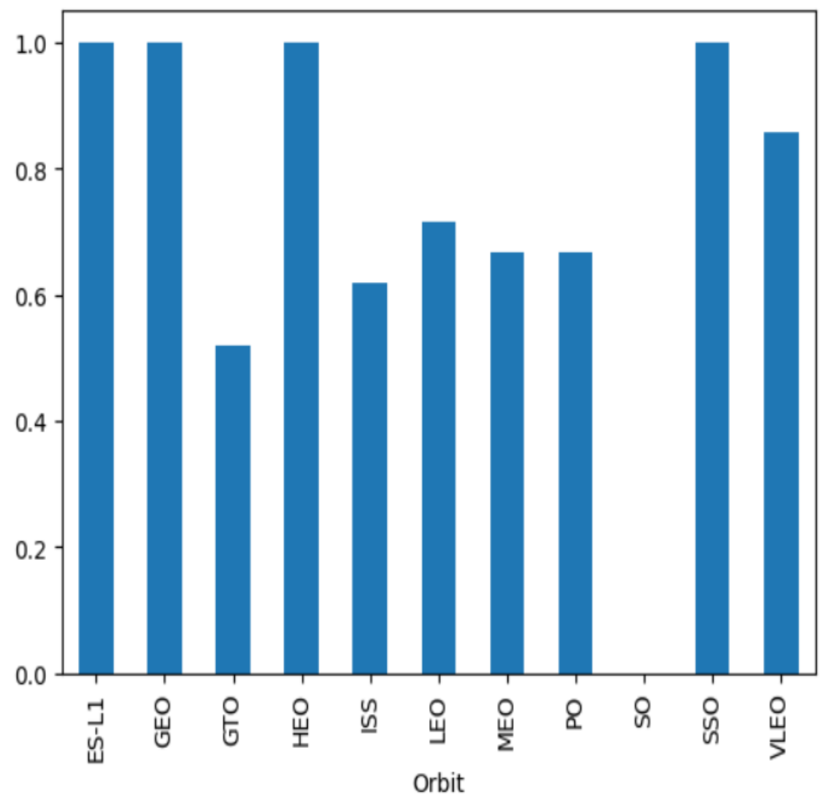
[9]:
```python
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("PayloadMass",fontsize=20)
plt.ylabel("LaunchSite",fontsize=20)
plt.show()
```



Success Rate vs. Orbit Type

- Shows the success rate for each orbit type.
- We can see that SO is the worst/empty and 100% for SSO, HEO, GEO and ES-L1
- But, SSO, HEO, GEO and ES-L1 have only one occurrence each

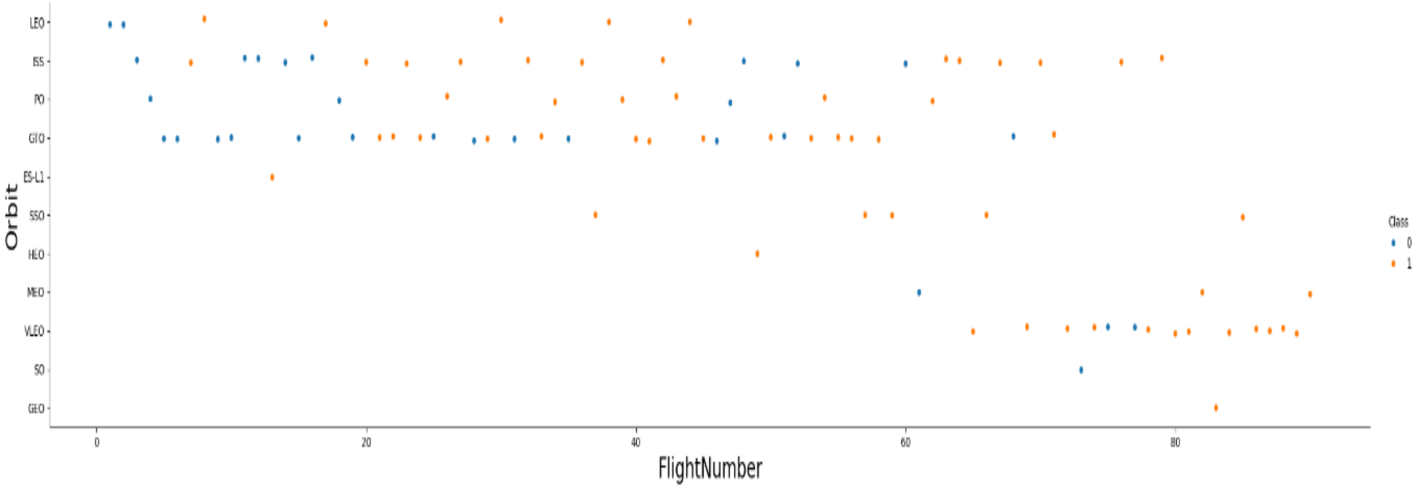
```
[10]: # HINT use groupby method on Orbit column and get the mean of Class column
      df.groupby('Orbit')['Class'].mean().plot.bar()

[10]: <AxesSubplot:xlabel='Orbit'>
```



Flight Number vs. Orbit Type

- We can see that the larger the flight number on each orbits, the greater the success rate except for GTO orbit which depicts no relationship between both attributes.
- We have seen that some orbits only have 1 occurrence so these cannot be seen as good comparisons

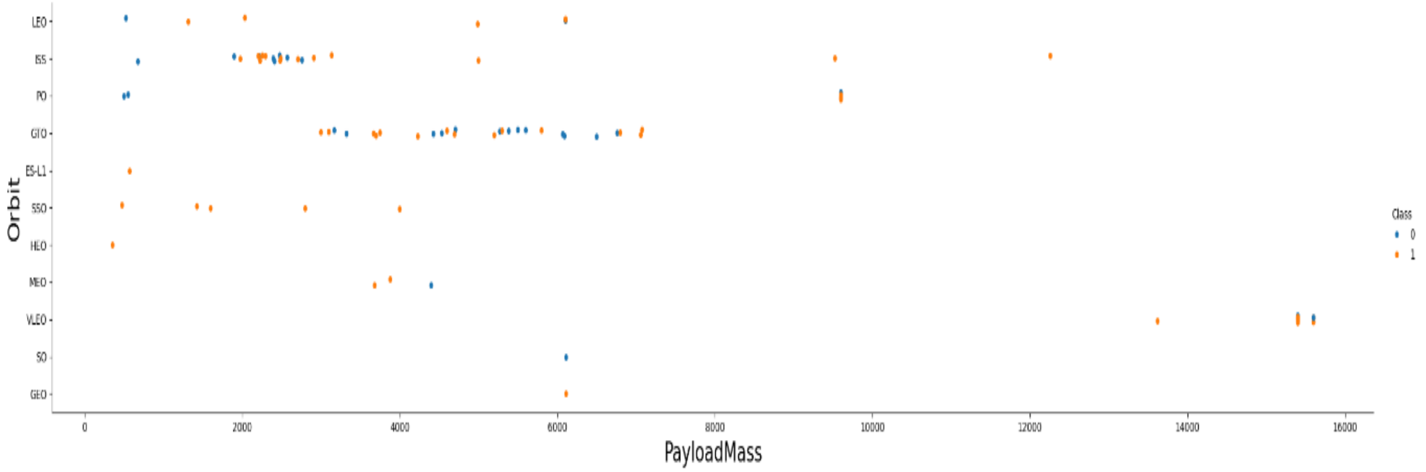
[11]:
```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("FlightNumber",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



Payload vs. Orbit Type

- We can see that heavier payload has positive impact on LEO, ISS and PO orbit.
- However, it has negative impact on MEO and VLEO orbit

[12]:
```python
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("PayloadMass",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```

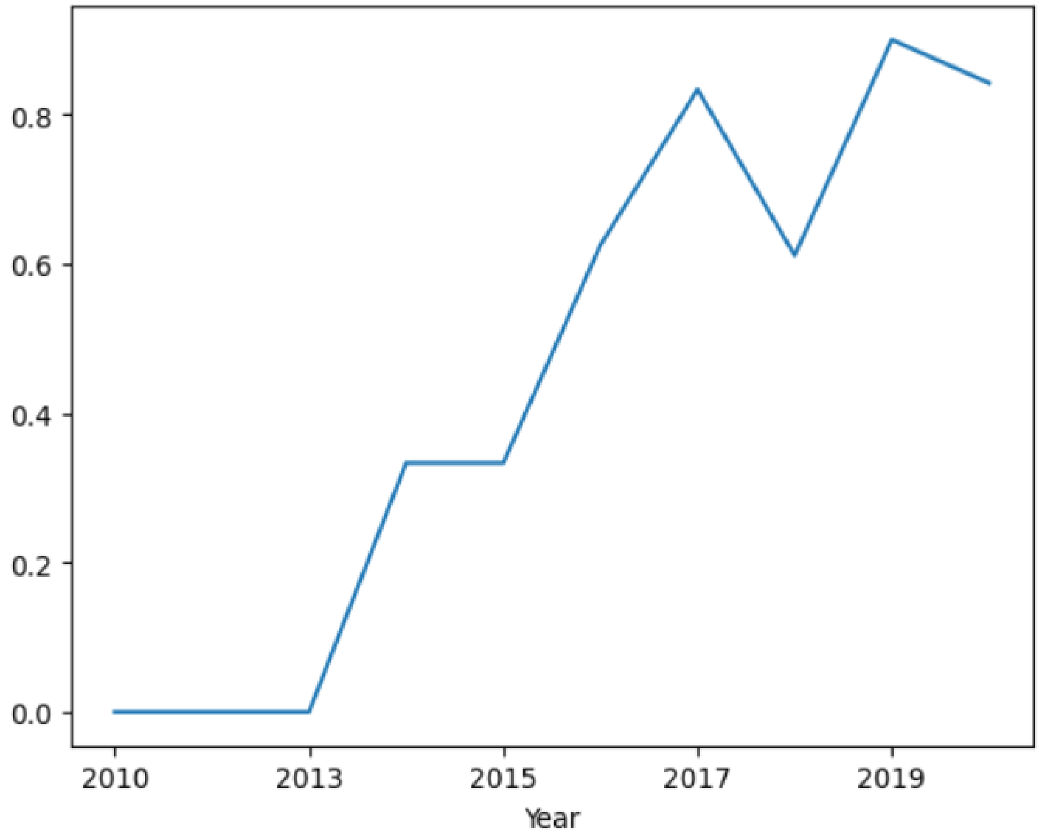


Launch Success Yearly Trend

- We can see an increasing trend from the year 2013 until 2020.
- If this trend possibly continues for the next year onward, the success rate may steadily increase to 100%

```
[14]: # Plot a line chart with x axis to be the extracted year and y axis to be the success rate
temp_df = df.copy()
temp_df['Year'] = year
temp_df.groupby('Year')['Class'].mean().plot()
```

```
[14]: <AxesSubplot:xlabel='Year'>
```



All Launch Site Names

- **Displays the names of the unique launch sites in the space mission**

```
[8]: %sql select DISTINCT launch_site from SPACEXTABLE

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

[8]:

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

Launch Site Names Begin with 'CCA'

- Display 5 records where launch sites begin with the string 'CCA'

```
[11]: %sql select * from SPACEXTABLE where launch_site like 'CCA%' limit 5

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

[11]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	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	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- Display the total payload mass carried by boosters launched by NASA (CRS)

```
[13]: %sql select sum(Payload_mass__kg_) from SPACEXTABLE where customer = 'NASA (CRS)'

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

[13]:

sum(Payload_mass__kg_)
45596

Average Payload Mass by F9 v1.1

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

```
[17]: %sql select AVG(payload_mass__kg_) from SPACEXTABLE where Booster_Version = 'F9 v1.1';

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

[17]:

AVG(payload_mass__kg_)
2928.4

First Successful Ground Landing Date

- List the date when the first succesful landing outcome in ground pad was acheived

```
[19]: %sql SELECT MIN(DATE) FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Success (ground pad)';

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

[19]:

MIN(DATE)
2015-12-22

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

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

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

[21]:

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

Total Number of Successful and Failure Mission Outcomes

- List the total number of successful and failure mission outcomes

```
[24]: %sql SELECT MISSION_OUTCOME, COUNT(*) FROM SPACEXTABLE GROUP BY MISSION_OUTCOME;

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

[24]:

Mission_Outcome	COUNT(*)
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Boosters Carried Maximum Payload

- List the names of the booster_versions which have carried the maximum payload mass

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

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

[31]:

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

2015 Launch Records

- List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015

```
[33]: ql 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.
```

[33]:

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

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

- Rank 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.

```
[30]: %sql SELECT LANDING_OUTCOME, COUNT(*) AS QTY FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY LANDING_OUTCOME ORDER BY QTY DESC;
```

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

[30]:

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

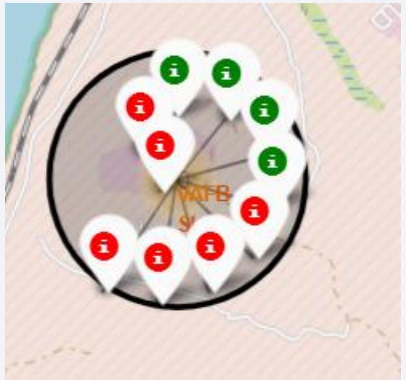
Section 3

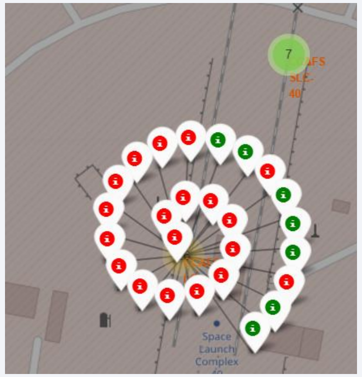
Launch Sites Proximities Analysis

Launch Site Locations

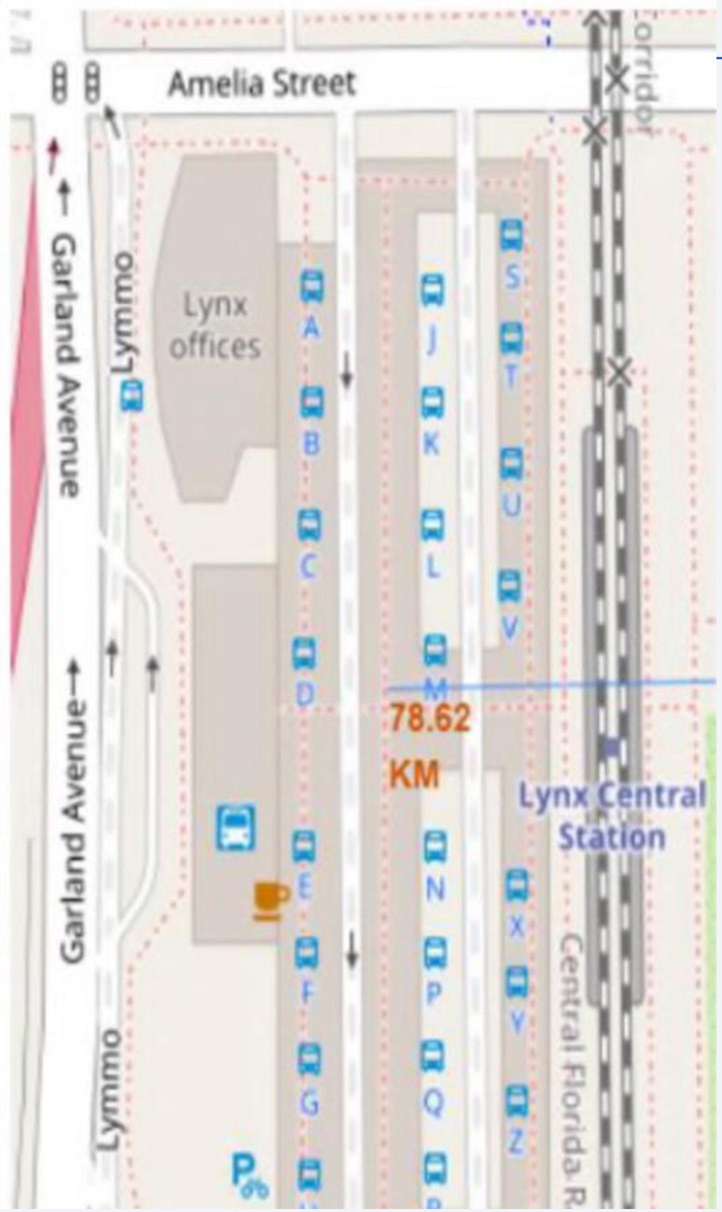
- We can see all the SpaceX launch sites are located inside the United States

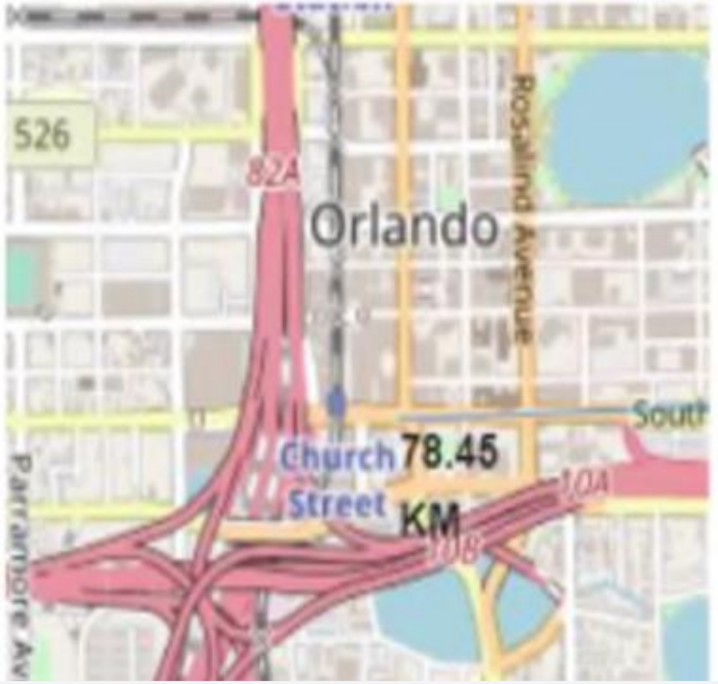
Markers with colored labelled launch sites





Launch sites distance from Landmarks

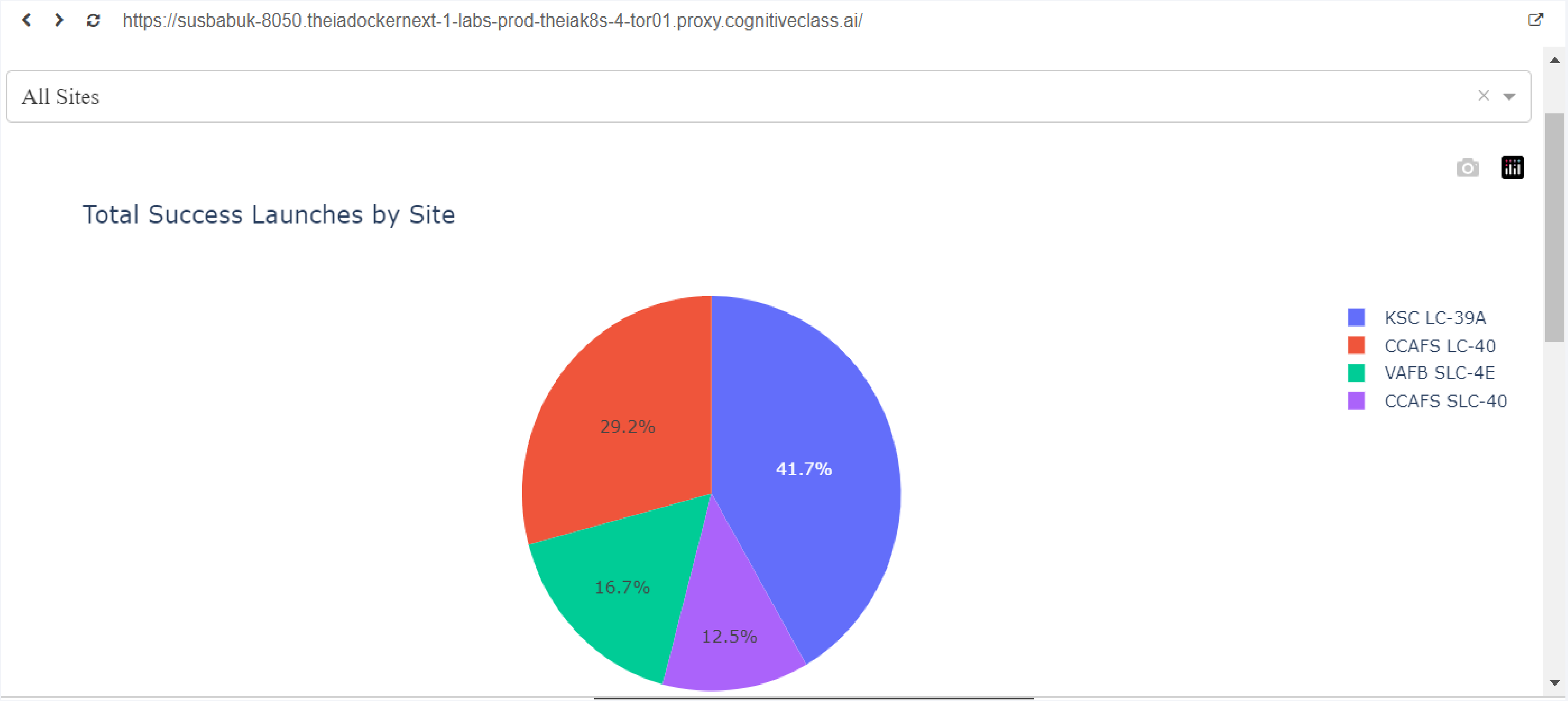




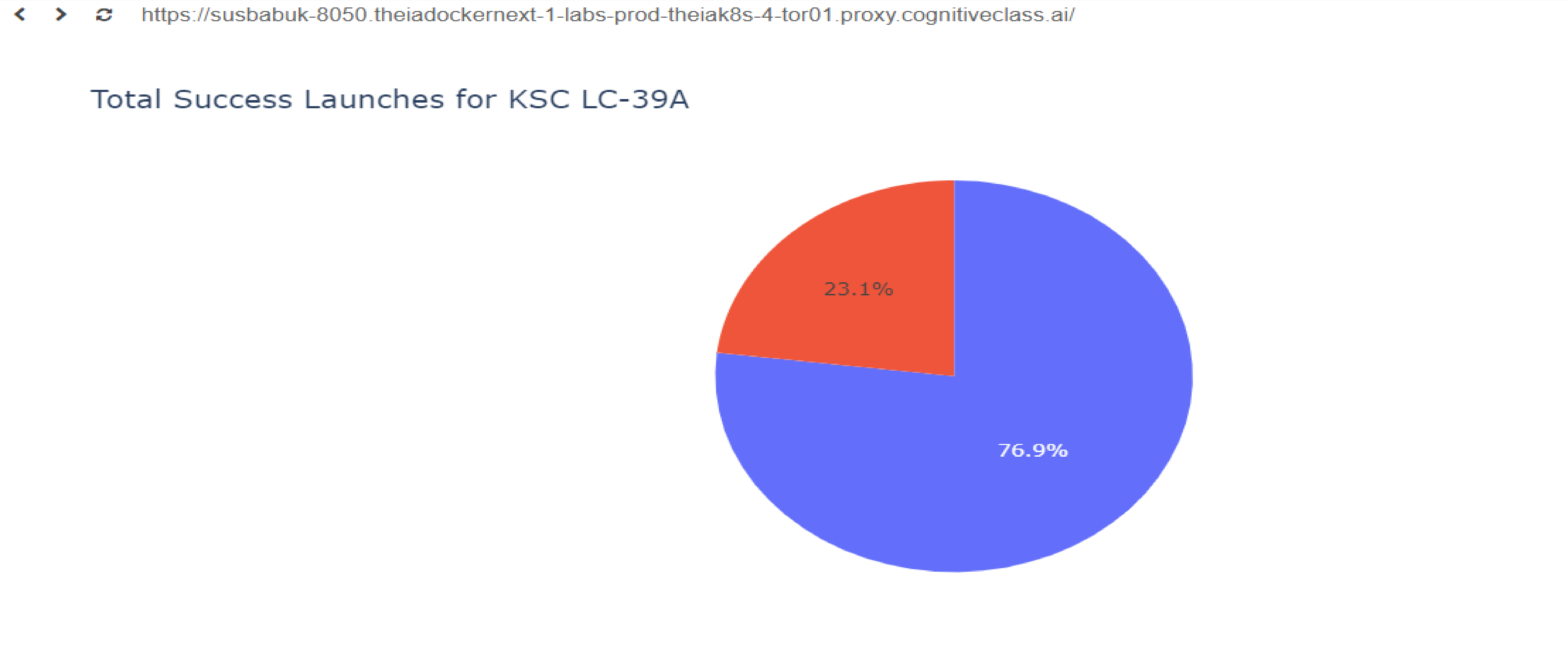
Section 4

Build a Dashboard with Plotly Dash

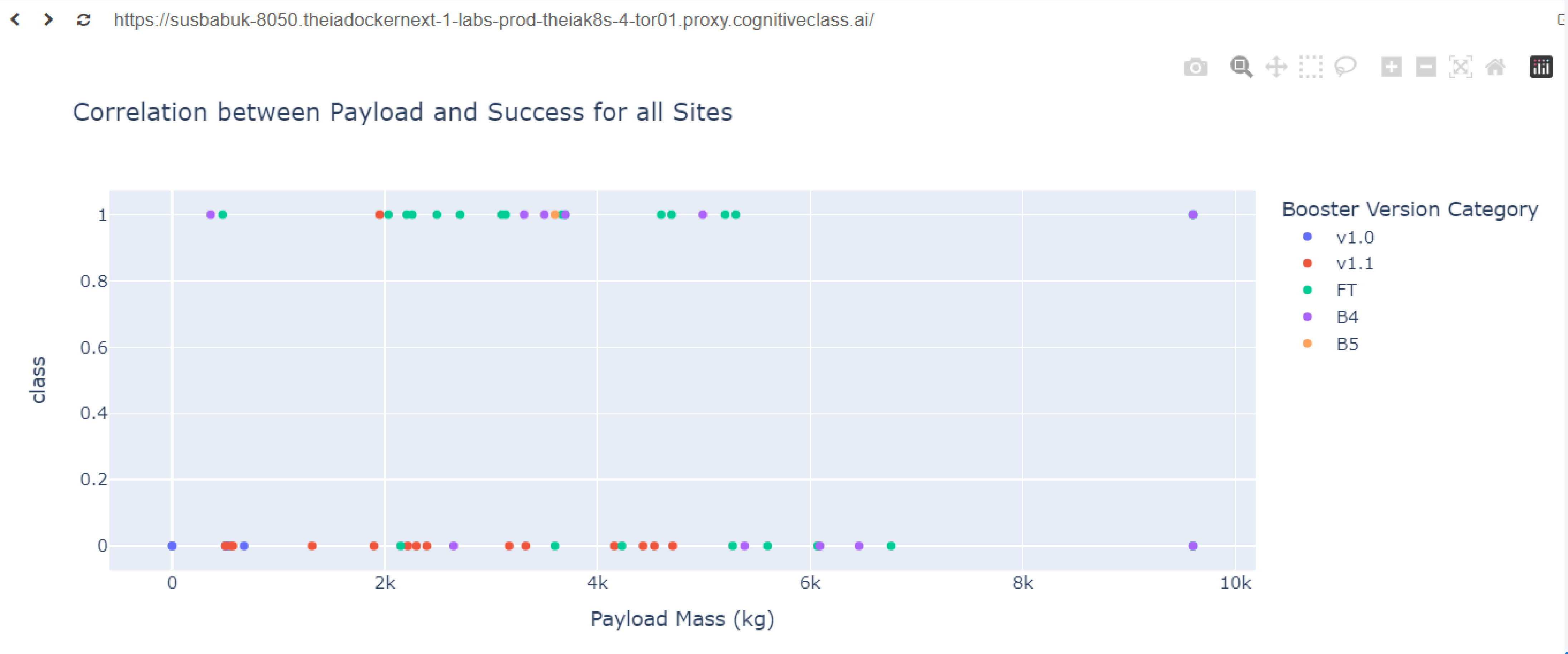
All Sites Success Rates



Highest Success Rate



Payload vs Launch Outcome Scatter plot



Section 5

Predictive Analysis (Classification)

Classification Accuracy

TASK 12

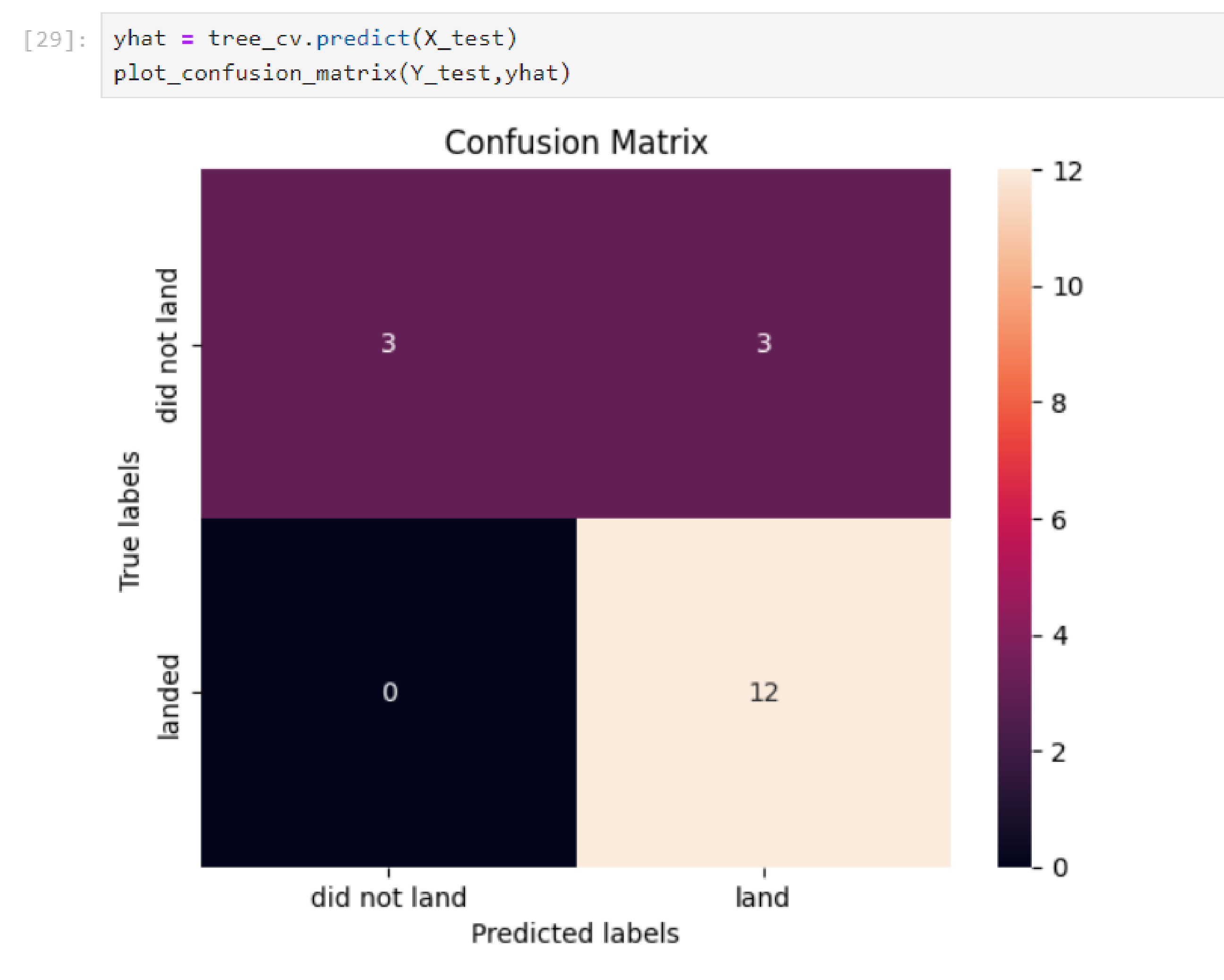
Find the method performs best:

[35]:
```
algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Tree':
    print('Best Params is :',tree_cv.best_params_)
if bestalgorithm == 'KNN':
    print('Best Params is :',knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best Params is :',logreg_cv.best_params_)
```

```
Best Algorithm is Tree with a score of 0.8875
Best Params is : {'criterion': 'gini', 'max_depth': 4, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 10, 'splitter': 'best'}
```

- Decision Tree was shown to have the highest classification accuracy with 88.75% score

Confusion Matrix



- We see 12 True Positive and 3 True Negatives classified which is good. But the 3 False Positives is a problem. Therefore 15/18 were classified accurately
- The confusion matrix given can be divided as
  - Top left – True Negative
  - Top right – False Positive
  - Bottom left – False Negative
  - Bottom right – True Positive

Conclusions

- The low weighted payloads, i.e., <4000, performed better than the heavy weighted payloads.
- Starting from the year 2013, the success rate for SpaceX launches has increased, directly proportional time in years to 2020, which it will eventually perfect the launches in the future.
- KSC LC-39A have the most successful launches of any sites; 76.9%
- SSO orbit had a 100% success rate and had more than one occurrence as compared to the others.
- The Decision Tree Algorithm is the best Machine Learning approach for this dataset.

Thank you!