Winning Space Race with Data Science

Tejal Swapnil Patil
12/02/2023

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection using API
  - Data collection using Web scraping
  - Data wrangling
  - EDA using Visualization lab
  - EDA using SQL
  - Data visualization using folium
  - Dashboard with Plotly Dash
  - Predictive analysis
- Summary of all results
  - EDA results
  - Interactive analytics
  - Predictive Analysis

Introduction

- **Project background and context**
  - Falcon 9 is a reusable, two-stage rocket designed and manufactured by SpaceX for the reliable and safe transport of people and payloads into Earth orbit and beyond
  - If we can determine if the first stage will land, we can determine the cost of a launch.
  - This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.
  - Also reusability allows SpaceX to refly the most expensive parts of the rocket, which in turn drives down the cost of space access.
- **Problems you want to find answers**
  - Factors affecting the success of first stage landing.
  - What should be the operating conditions for successful landing?
  - Relation between the features which determines the success rate of successful landing.

Section 1

Methodology

Methodology

Executive Summary

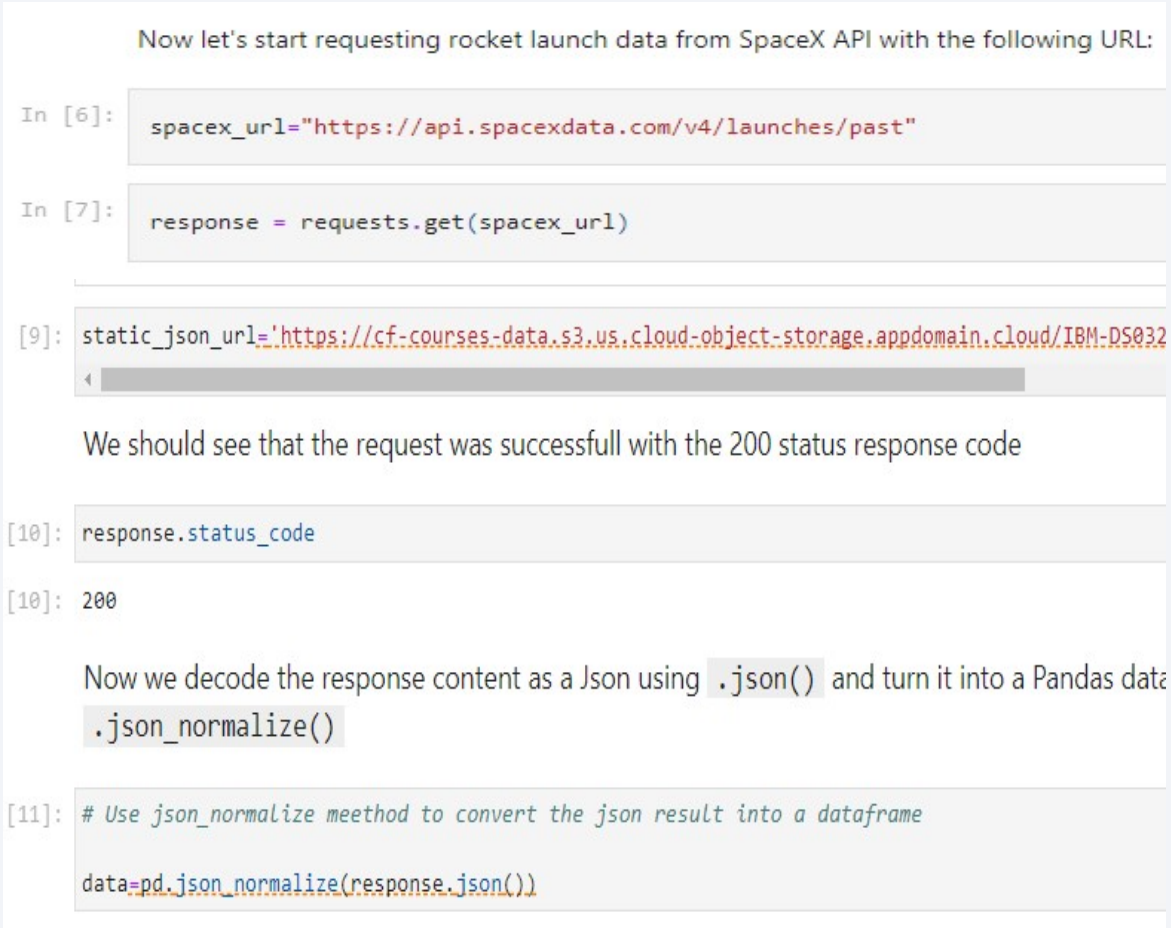
- **Data collection methodology:**
  - Data was collected by get request SpaceX API and Web scraping with Beautifulsoup method.
- **Perform data wrangling**
  - Data is wrangling using one hot encoding for ‘Outcomes’.
- **Perform exploratory data analysis (EDA) using visualization and SQL**
- **Perform interactive visual analytics using Folium and Plotly Dash**
- **Perform predictive analysis using classification models**
  - LR, SVM, Decission tree and KNN have been built
  - Tunes using best_params_ and evaluated by finding accuracy using score method.

Data Collection

- Data was collected by **get request SpaceX API**
- Then decoding of the response content as a Json using .json() was done and turn it into a Pandas dataframe using .json_normalize().
- Next we removed the Falcon 1 launches keeping only the Falcon 9 launches.
- Further missing value from payload mass column was replaced with mean of the same column.
- Additionally, performing **web scraping using 'Beautifulsoup'** Method to collect Falcon 9 historical launch records from a Wikipedia page titled List of Falcon 9 and Falcon Heavy launches.
- Created a data frame by parsing the launch HTML tables

Data Collection – SpaceX API

- Data was collected by get request SpaceX API
- Then decoding of the response content as a Json using .json() was done and turn it into a Pandas dataframe using .json_normalize().
- Next we removed the Falcon 1 launches keeping only the Falcon 9 launches.
- Further missing value from payload mass column was replaced with mean of the same column.
- GitHub URL of the completed SpaceX API calls
- https://github.com/tejal21190/IBM-Data-Science-SpaceX-Project/blob/master/1.%20Data%20collection%20API.ipynb



Data Collection - Scraping

- Performing web scraping using 'Beautifulsoup' Method to collect Falcon 9 historical launch records from a Wikipedia page titled List of Falcon 9 and Falcon Heavy launches.
- Created a data frame by parsing the launch HTML tables
- GitHub URL of the completed web scraping notebook
- https://github.com/tejal21190/IBM-Data-Science-SpaceX-Project/blob/master/2.%20Data%20Collection%20with%20Web%20Scraping.ipynb

In [4]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and
```

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

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

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

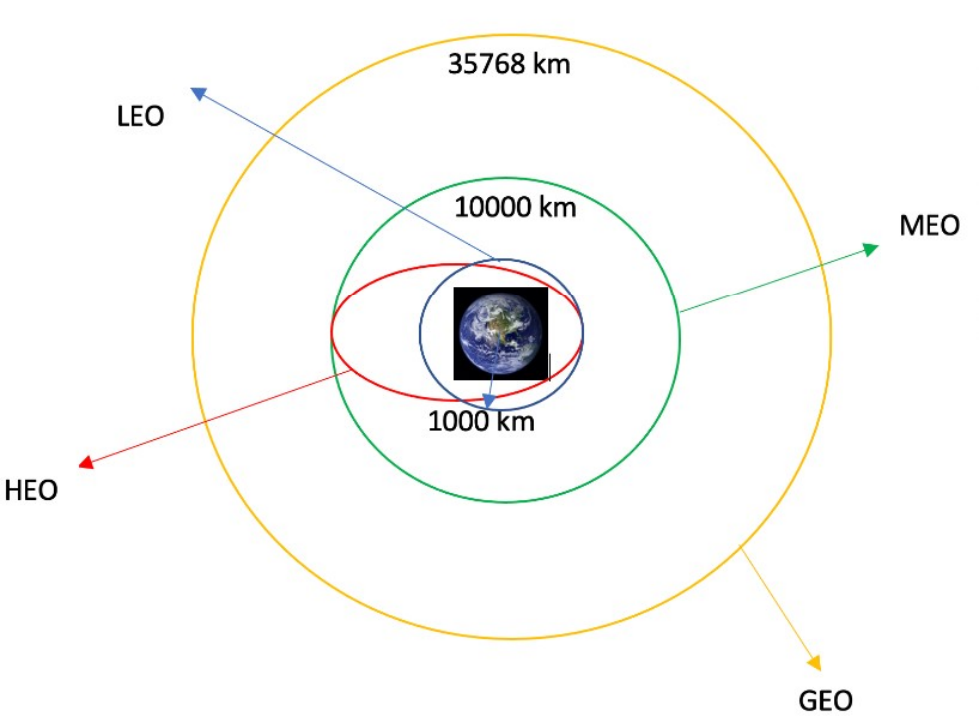
In [10]:
```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables =soup.find_all('table')

html_tables
```

In [36]:
```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
```

Data Wrangling

- We have done exploratory Data Analysis and determine Training Labels
- Using value_counts() following data is calculated
  - The number of launches on each site
  - The number and occurrence of each orbit
  - The number and occurrence of mission outcome per orbit type
- We have created a landing outcome label from Outcome column
- https://github.com/tejal21190/IBM-Data-Science-SpaceX-Project/blob/master/3.%20Data%20wrangling.ipynb



EDA with Data Visualization

Sr. No.	Plot	Parameters
1.	Scatter plot	Flight Number vs. Payload Mass
2.	Scatter plot	Flight Number Vs Launch Site
3.	Scatter plot	Payload Vs Launch Site
4.	Bar Chart	Class Vs Orbit
5.	Scatter Plot	Flight Number Vs Orbit type
6.	Scatter plot	Payload Vs Orbit type
7.	Line chart	Plot of success rate

https://github.com/tejal21190/IBM-Data-Science-SpaceX-Project/blob/master/4.%20EDA%20with%20Visualization%20lab.ipynb

EDA with SQL

SQL queries performed

- Displayed the names of the unique launch sites in the space mission
- Displayed 5 records where launch sites begin with the string 'CCA
- Displayed the total payload mass carried by boosters launched by NASA (CRS)
- Displayed average payload mass carried by booster version F9 v1.1
- Listed the date when the first successful landing outcome in ground pad was achieved
- Listed the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- Listed the total number of successful and failure mission outcomes
- Listed the names of the booster_versions which have carried the maximum payload mass with subquery
- Listed the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
- 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

- https://github.com/tejal21190/IBM-Data-Science-SpaceX-Project/blob/master/5.%20Exploratory%20Data%20analysis%20with%20SQL.sql

Build an Interactive Map with Folium

- Created a folium Map object, with an initial center location to be NASA Johnson Space Center at Houston, Texas.
- Used folium.Circle to add a highlighted circle area with a text label on a specific coordinate.
- Created and add folium.Circle and folium.Marker for each launch site on the site map.
- Created a MarkerCluster object and then added a folium.Marker to marker_cluster.
- To explore and analyze the proximities of launch sites, added a MousePosition on the map to get coordinate for a mouse over a point on the map.
- Created and added a folium.Marker on your selected closest coastline point on the map
- Created a `folium.PolyLine` object using the coastline coordinates and launch site coordinate
- Created and added a folium.Marker on your selected closest city and railway point on the map
- Created a `folium.PolyLine` object using city and railway coordinates and launch site coordinate

https://github.com/tejal21190/IBM-Data-Science-SpaceX-Project/blob/master/6.%20Data%20visiualization%20using%20folium.ipynb

Build a Dashboard with Plotly Dash

- **Pie chart** for ‘total success launches by all sites’ to find relation of success launch with launch sites
- **Scatter Plot** for ‘success-payload-scatter-chart‘ to find relation of payload mass on success launch.

https://github.com/tejal21190/IBM-Data-Science-SpaceX-Project/blob/master/7.spacex_dash_app.py

Predictive Analysis (Classification)

- Loaded data using numpy and pandas, and Standardized the data in X then reassigned it to the variable X using the transform
- We splited the data into training and testing data using the function train_test_split
- We built different machine learning models and tune with best_params_ hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model.

In [4]:
```
from js import fetch
import io

URL1 = "https://cf-courses-data.s3.us.cloud-object-storag
resp1 = await fetch(URL1)
text1 = io.BytesIO((await resp1.arrayBuffer()).to_py())
data = pd.read_csv(text1)
```

In [6]:
```
URL2 = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cl
resp2 = await fetch(URL2)
text2 = io.BytesIO((await resp2.arrayBuffer()).to_py())
X = pd.read_csv(text2)
```

In [9]:
```
# students get this
transform = preprocessing.StandardScaler()
X=transform.fit_transform(X)
```

In [10]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
```

https://github.com/tejal21190/IBM-Data-Science-SpaceX-Project/blob/master/8.SpaceX_Machine_Learning_Prediction.ipynb

Logistic regression

In [14]:
```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
lr=LogisticRegression()
#create a GridSearchCV object logreg_cv
logreg_cv=GridSearchCV(lr,parameters,cv=10)
# Fit the object to find the best parameters from the dictionary parameters
logreg_cv.fit(X_train,Y_train)
```

In [15]:
```
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
print("accuracy :",logreg_cv.best_score_)
```

SVM

In [18]:
```
parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
              'C': np.logspace(-3, 3, 5),
              'gamma':np.logspace(-3, 3, 5)}
svm = SVC()
#create a GridSearchCV object svm_cv
svm_cv=GridSearchCV(svm,parameters,cv=10)
# Fit the object to find the best parameters from the dictionary parameters
svm_cv.fit(X_train,Y_train)
```

In [19]:
```
print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
print("accuracy :",svm_cv.best_score_)
```

Decision tree

In [22]:
```
parameters = {'criterion': ['gini', 'entropy'],
     'splitter': ['best', 'random'],
     'max_depth': [2*n for n in range(1,10)],
     'max_features': ['auto', 'sqrt'],
     'min_samples_leaf': [1, 2, 4],
     'min_samples_split': [2, 5, 10]}

tree = DecisionTreeClassifier()
#create a GridSearchCV object tree_cv
tree_cv=GridSearchCV(tree,parameters,cv=10)
# Fit the object to find the best parameters from the dictionary parameters
tree_cv.fit(X_train,Y_train)
```

In [23]:
```
print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)
```

KNN

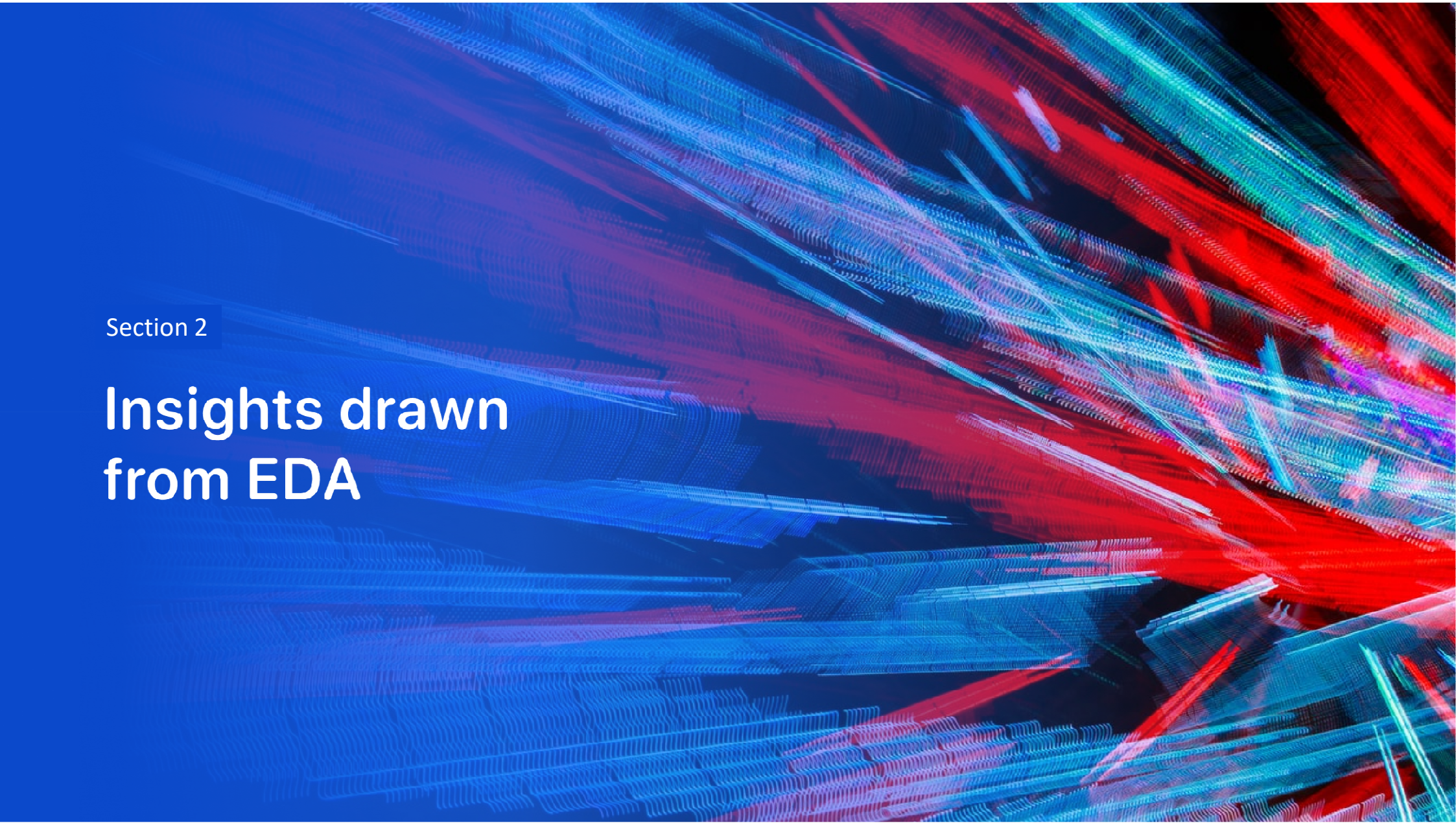
In [26]:
```
parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
              'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
              'p': [1,2]}

KNN = KNeighborsClassifier()
#create a GridSearchCV object knn_cv
knn_cv=GridSearchCV(KNN,parameters,cv=10)
# Fit the object to find the best parameters from the dictionary parameters
knn_cv.fit(X_train,Y_train)
```

In [27]:
```
print("tuned hpyerparameters :(best parameters) ",knn_cv.best_params_)
print("accuracy :",knn_cv.best_score_)
```

Results

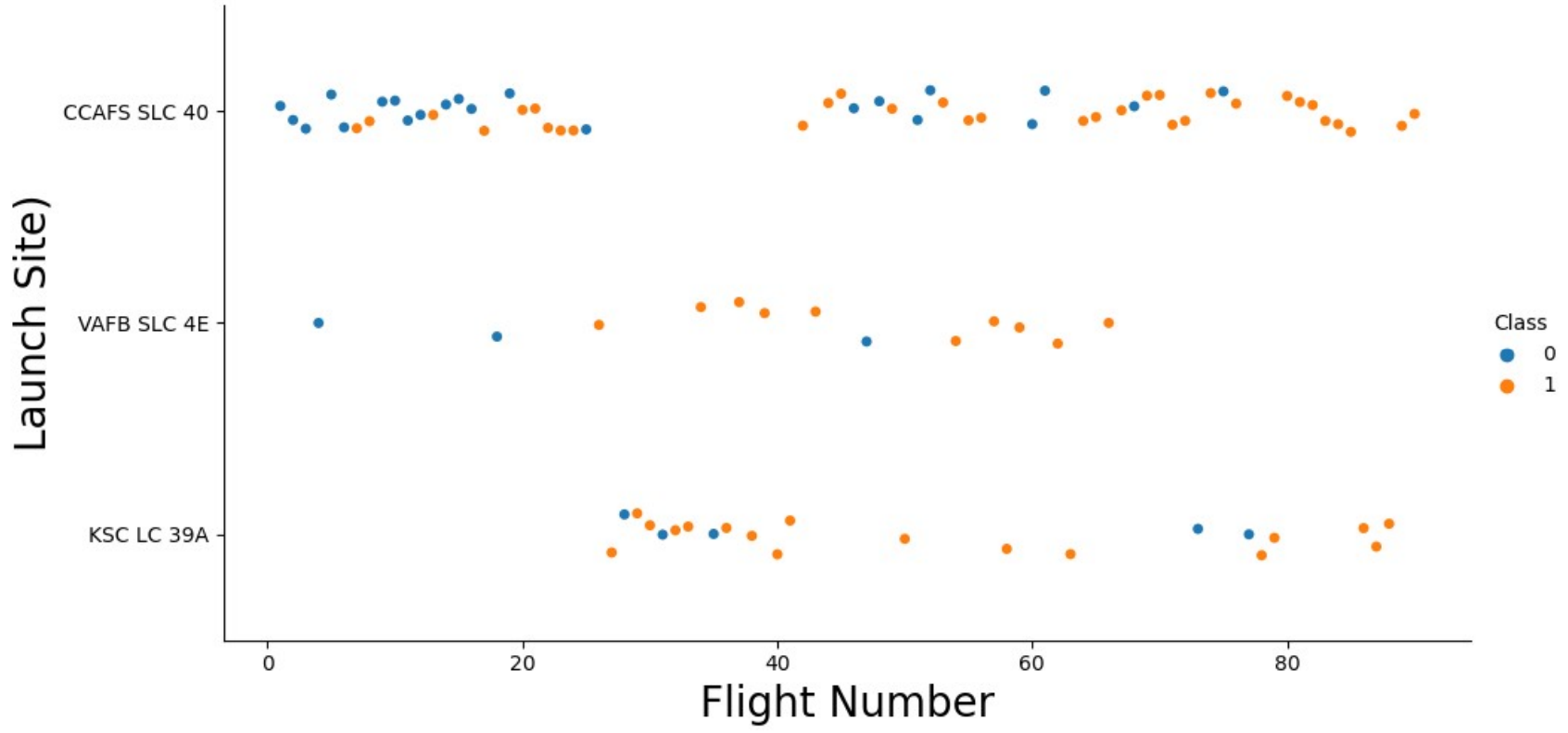
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results



Section 2

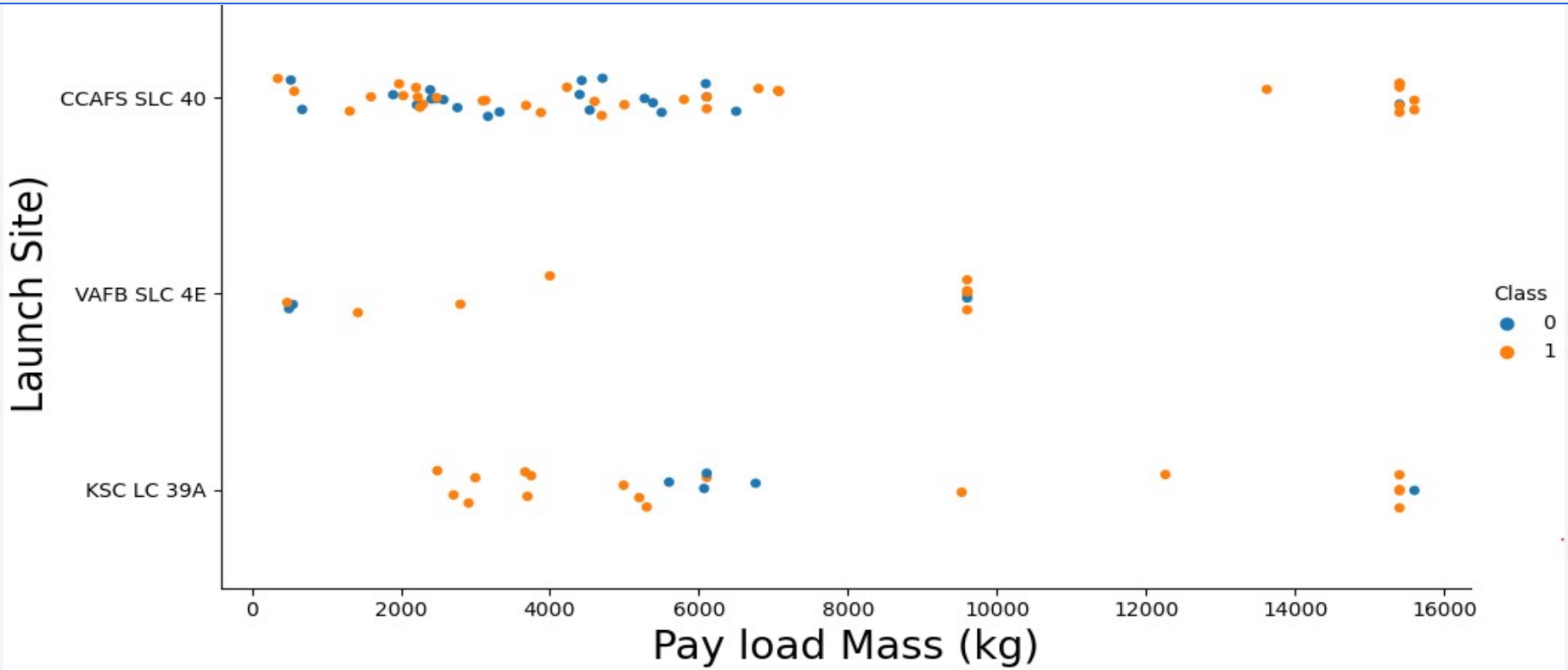
Insights drawn from EDA

Flight Number vs. Launch Site



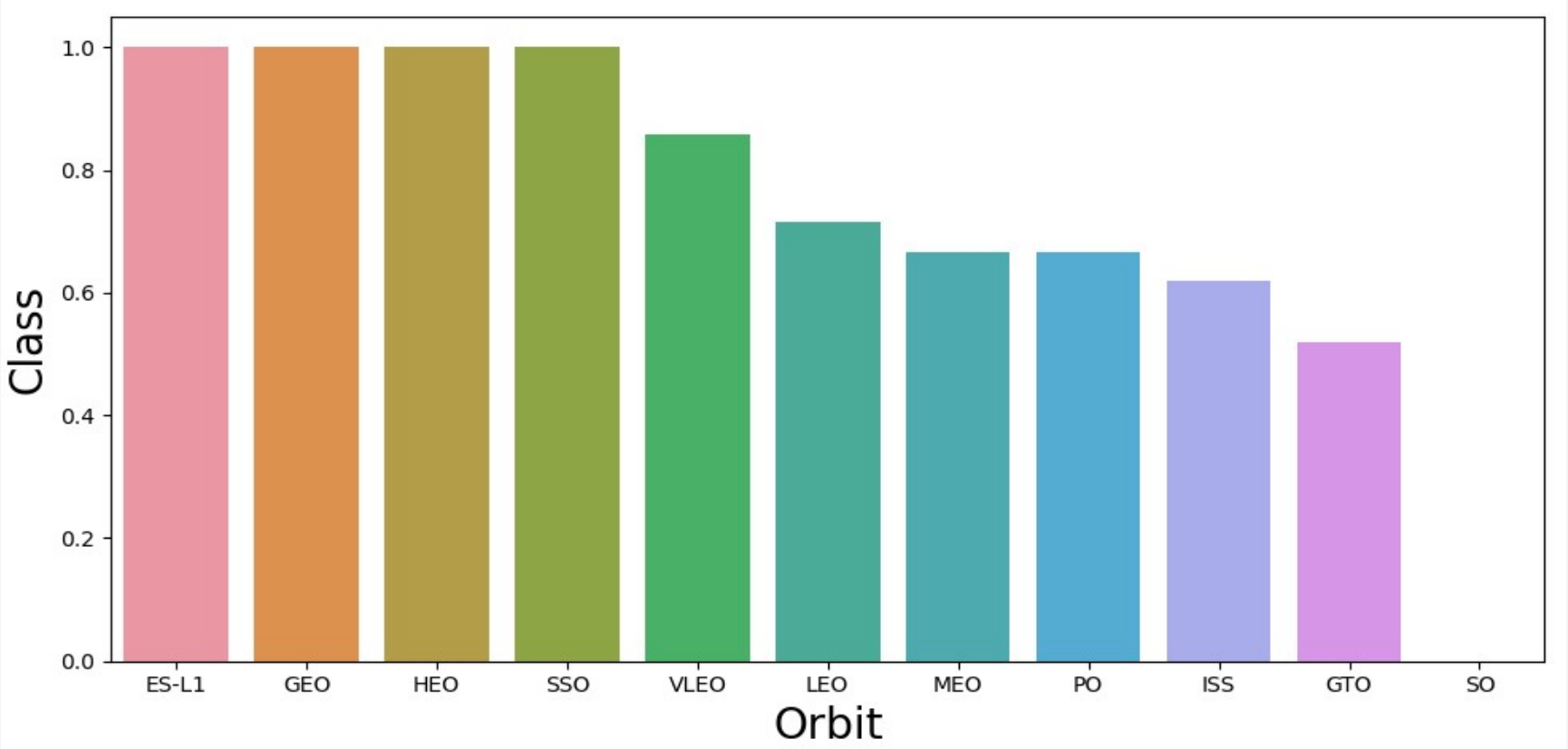
- Scatter plot of Flight Number vs. Launch Site
- It is seen that larger the flight amount at a launch site, the greater the success rate at a launch site

Payload vs. Launch Site



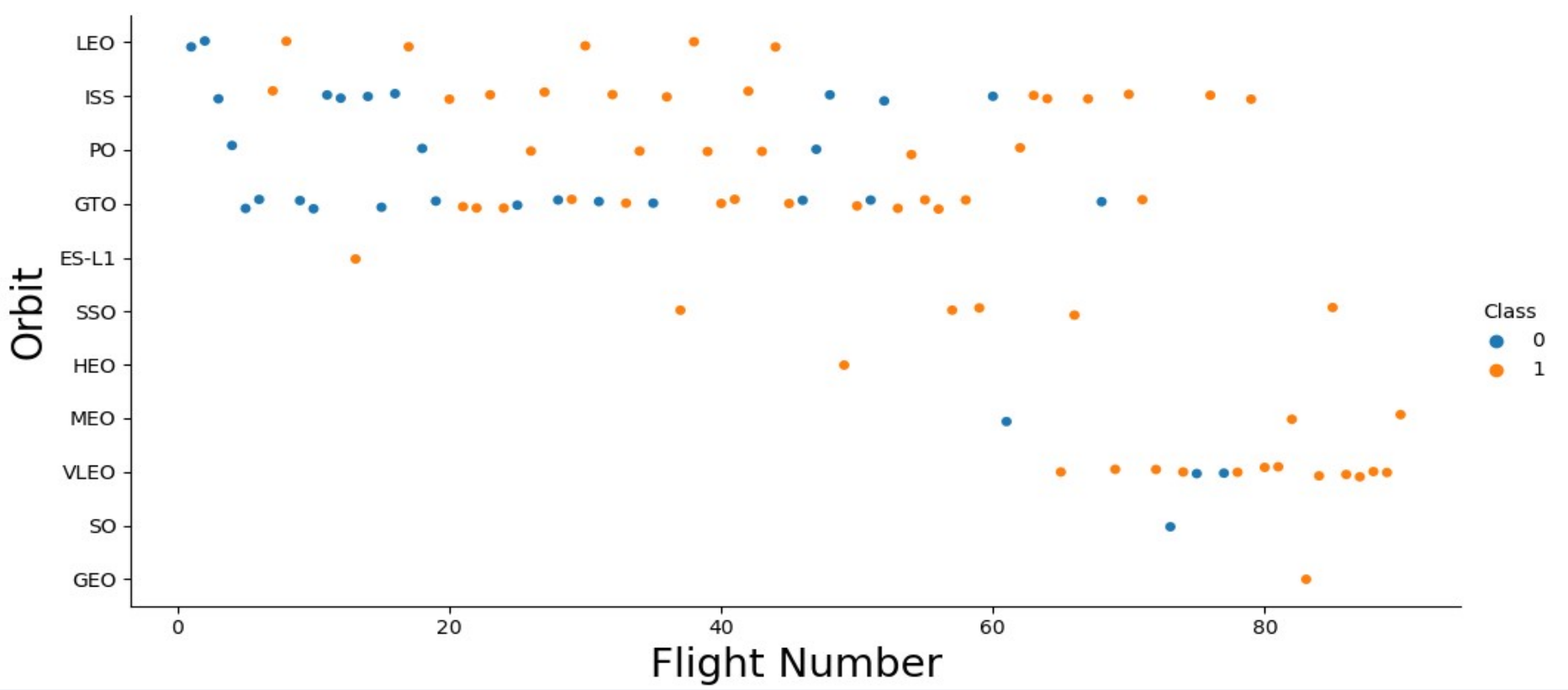
- Scatter plot of Payload vs. Launch Site
- It is seen that for CCAFS SLC 40, higher Payload mass gives higher success rate

Success Rate vs. Orbit Type



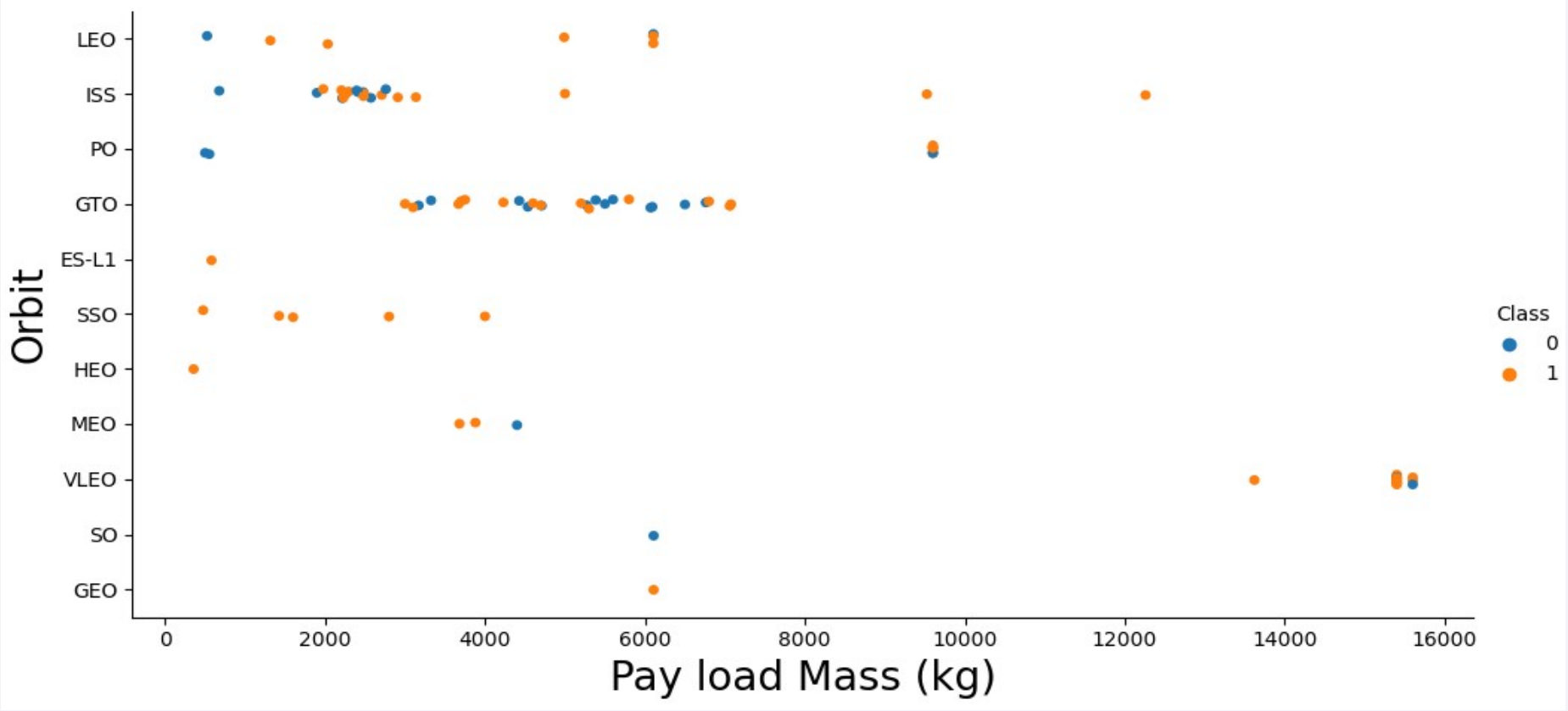
- Bar chart for the success rate of each orbit type
- It is seen that ES-L1, GEO, HEO, SSO, VLEO had the most success rate.

Flight Number vs. Orbit Type



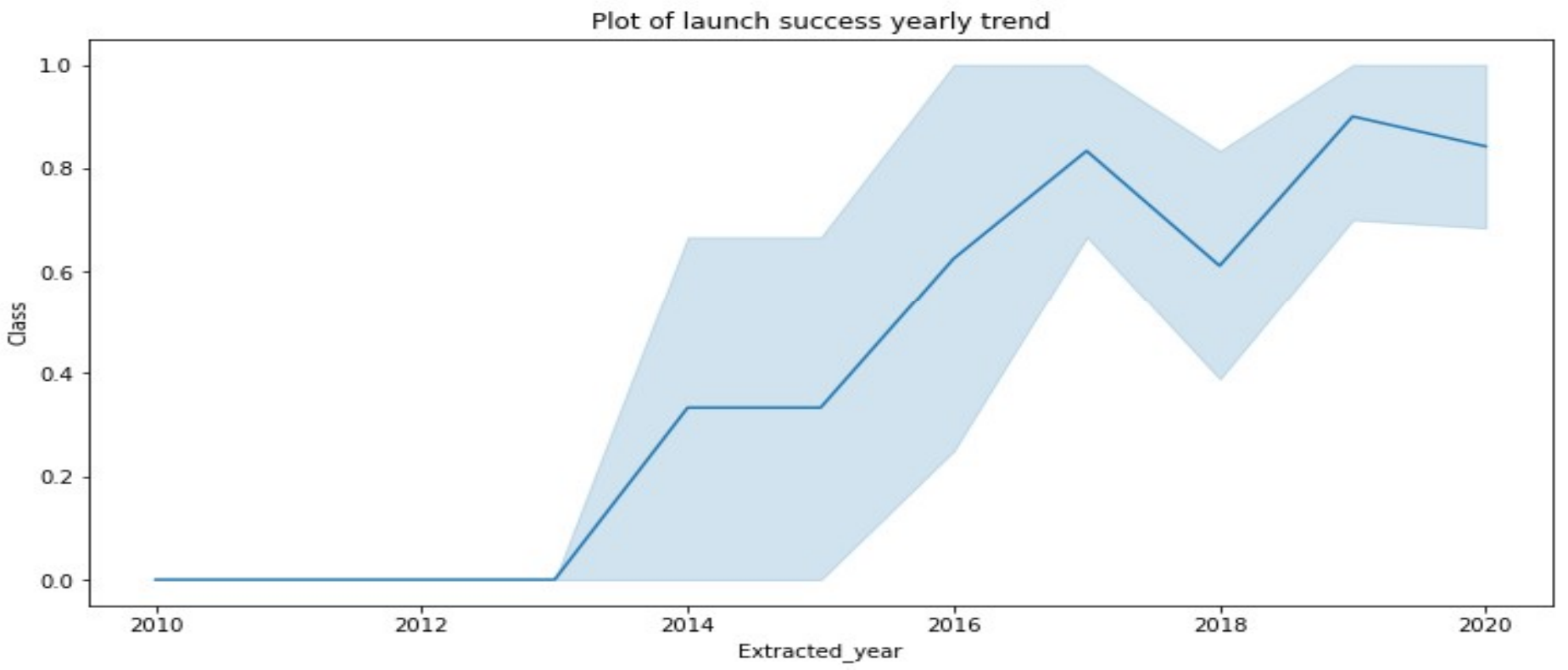
- Scatter point of Flight number vs. Orbit type
- It is seen that Flight Number vs. Orbit type. We observe that in the LEO orbit, success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit

Payload vs. Orbit Type



- Scatter point of payload vs. orbit type
- It is seen that heavy payloads, the successful landing are more for PO, LEO and ISS orbits.

Launch Success Yearly Trend



- Line chart of yearly average success rate
- It is observed that success rate since 2013 kept on increasing till 2020.

All Launch Site Names

- Method **DISTINCT** is used to show only unique launch sites from the SpaceX data.

```
/*Task 1
Display the names of the unique launch sites in the space mission*/
SELECT distinct(Launch_Site) FROM spacex;
```

Result Grid

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

Launch Site Names Begin with 'CCA'

- 5 records where launch sites begin with `CCA` were displayed using shown query.

```
/*Task 2
Display 5 records where launch sites begin with the string 'CCA'*/
SELECT * FROM spacex WHERE Launch_Site like 'CCA%' LIMIT 5;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MAS	Orbit	Customer	Mission_Outc	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft ...	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight ...	0	LEO (ISS)	NASA (COTS...	Success	Failure (parachute)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	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

- Calculated the total payload carried by boosters from NASA using **SUM**

```
/*Task 3
Display the total payload mass carried by boosters launched by NASA (CRS)*/
SELECT SUM(PAYLOAD_MASS__KG_) FROM spacex WHERE Customer='NASA (CRS)';
```

Result Grid | Filter Rows:

	SUM(PAYLOAD_MASS__KG_)
▶	45596

Average Payload Mass by F9 v1.1

- Calculated the average payload mass carried by booster version F9 v1.1 using **AVG**

```
/*Task 4
Display average payload mass carried by booster version F9 v1.1*/
SELECT AVG(PAYLOAD_MASS__KG_) FROM spacex WHERE Booster_Version='F9 v1.1';
```

Result Grid | Filter Rows:

AVG(PAYLOAD_MASS__KG_)
2928.4000

First Successful Ground Landing Date

- The dates of the first successful landing outcome on ground pad was displayed using **MIN**

```
/*Task 5
List the date when the first successful landing outcome in ground pad was acheived.*/
SELECT min(Date) as First_successfull_Landspacex FROM spacex WHERE Landing_Outcome ='Success (ground pad)';
```

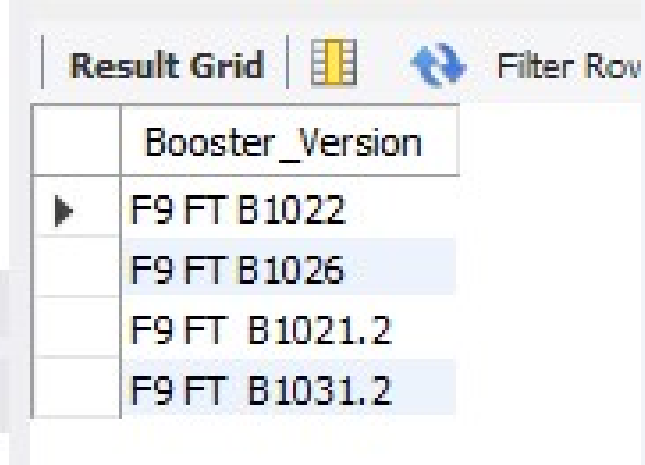
Result Grid | Filter Rows:

	First_successfull_Landspacex
▶	2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- Listed the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

```
/*Task 6
List the names of the boosters which have success in drone ship and have payload mass
greater than 4000 but less than 6000*/
SELECT Booster_Version FROM spacex 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

Total Number of Successful and Failure Mission Outcomes

- Calculated the total number of successful and failure mission outcomes

```
/*Task 7
List the total number of successful and failure mission outcomes*/
SELECT distinct(Mission_Outcome),count(Mission_Outcome) FROM spacex GROUP by Mission_Outcome;
```

Result Grid | Filter Rows: | Export:

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

Boosters Carried Maximum Payload

- Listed the names of the booster which have carried the maximum payload mass

```
/*Task 8
List the names of the booster_versions which have carried the maximum payload mass. Use a subquery*/
SELECT Booster_Version, PAYLOAD_MASS__KG_ FROM spacex
WHERE PAYLOAD_MASS__KG_=(SELECT Max(PAYLOAD_MASS__KG_) FROM spacex);
```

Result Grid | Filter Rows:

Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

2015 Launch Records

- Listed the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

```
/*Task 9
List the failed landing_outcomes in drone ship, their booster versions,
and launch site names for in year 2015*/
SELECT Landing_Outcome,Booster_Version,Launch_Site FROM spacex
WHERE Landing_Outcome='Failure (drone ship)' AND DATE like '2015%';
```

Result Grid | Filter Rows: | Expor

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

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

```
/*Task 10
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*/
SELECT Landing_Outcome,COUNT(Landing_Outcome) FROM spacex
where Date BETWEEN '2010-06-04' and '2017-03-20'
GROUP BY Landing_Outcome ORDER BY COUNT(Landing_Outcome) DESC;
```

Result Grid | Filter Rows:

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

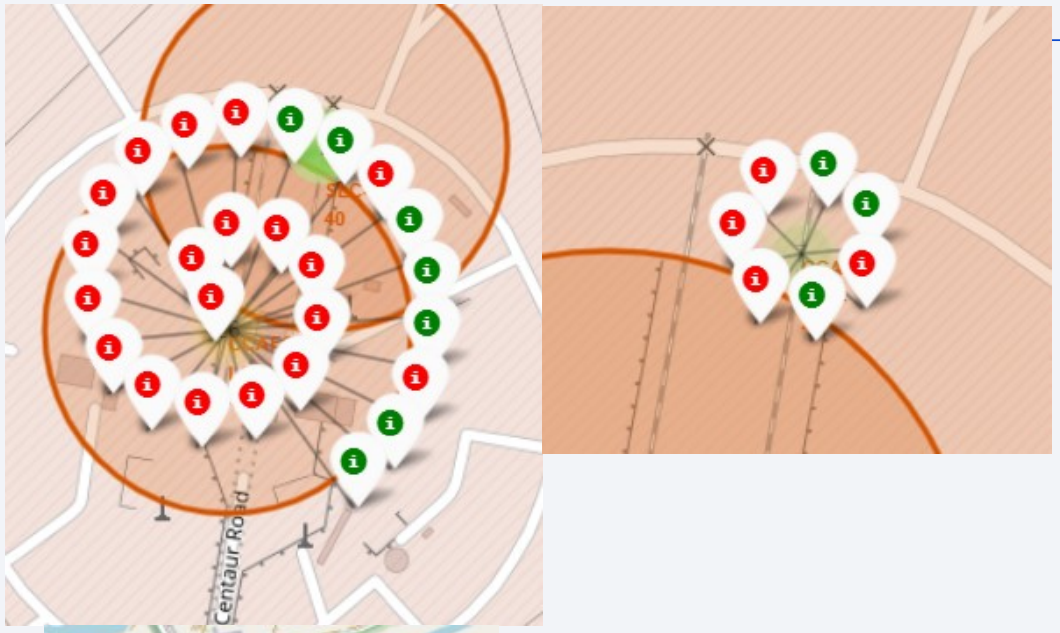
Section 3

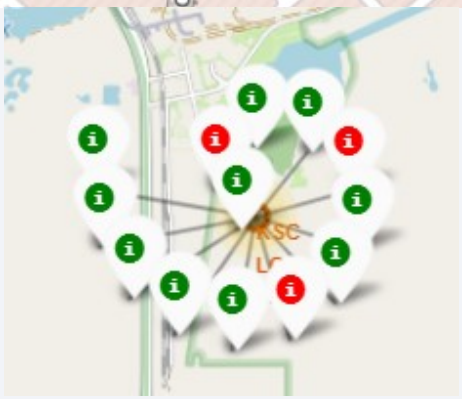
Launch Sites Proximities Analysis

All launch sites on global map

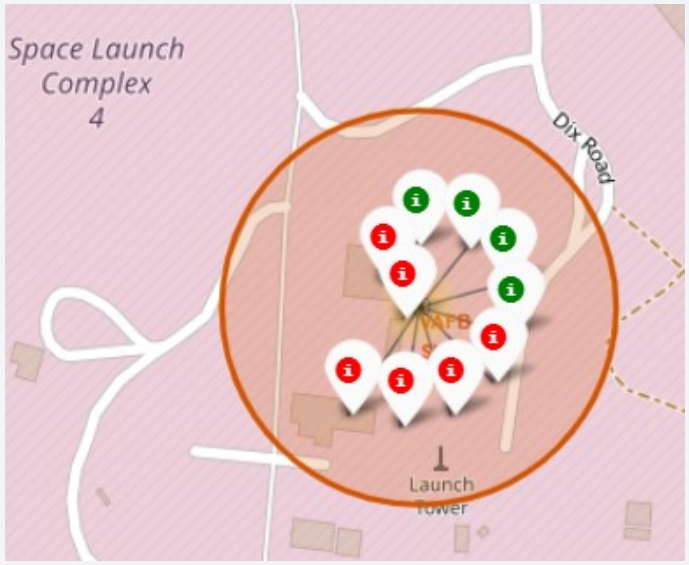
It is seen that all launch sites are in United states of American Coasts. California and Florida.

Markers showing launch sites with colored labels





Launch sites of Florida

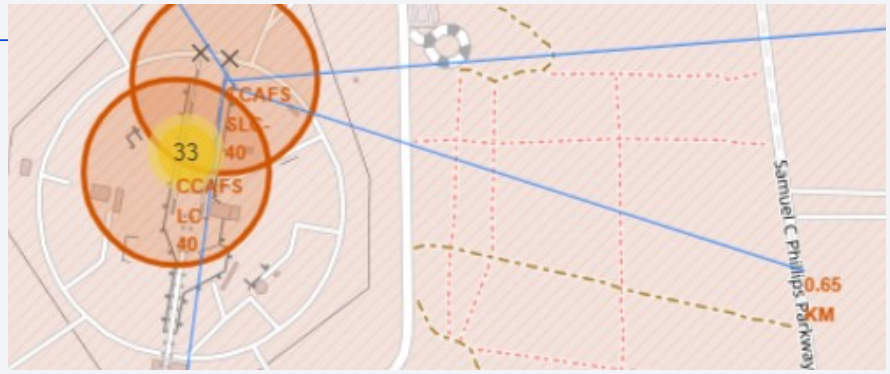


Launch sites of California

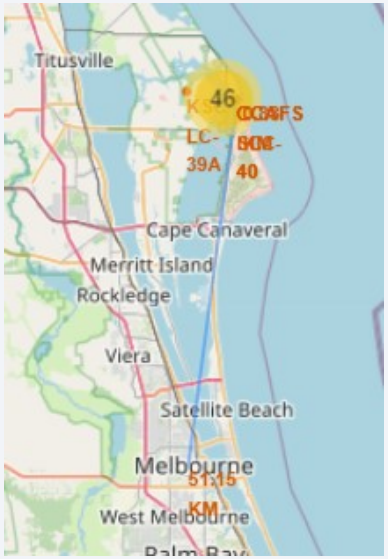
- **Red** colour denotes failed launches
- **Green** colour denotes success launches

Launch Site distance to landmarks

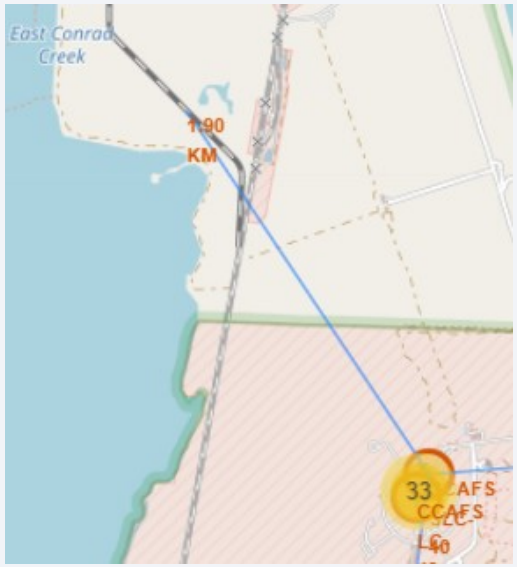
Distance of Coastline from Launch sites



Distance of Highway from Launch site

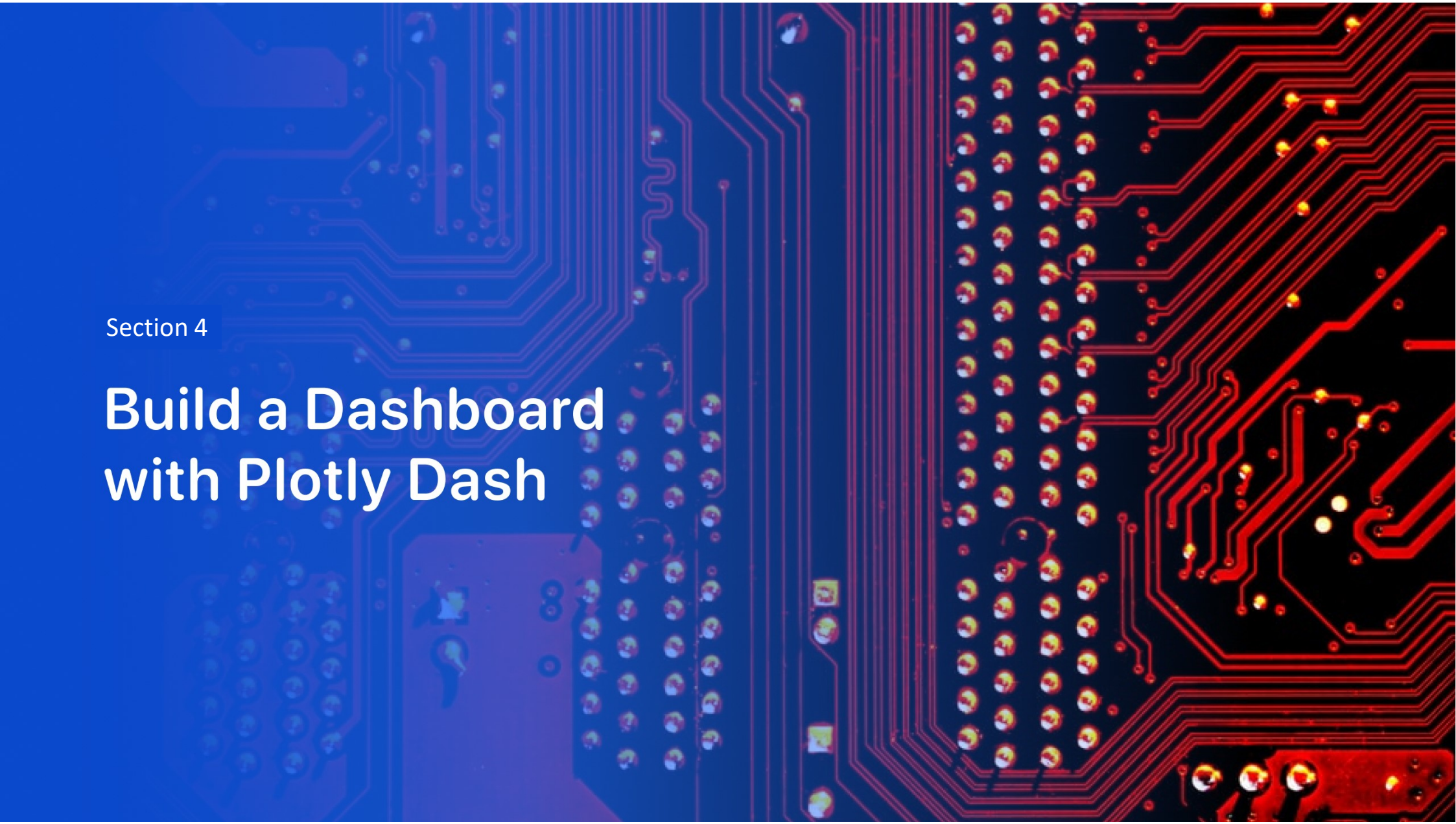


Distance of Melbourne city from launch site

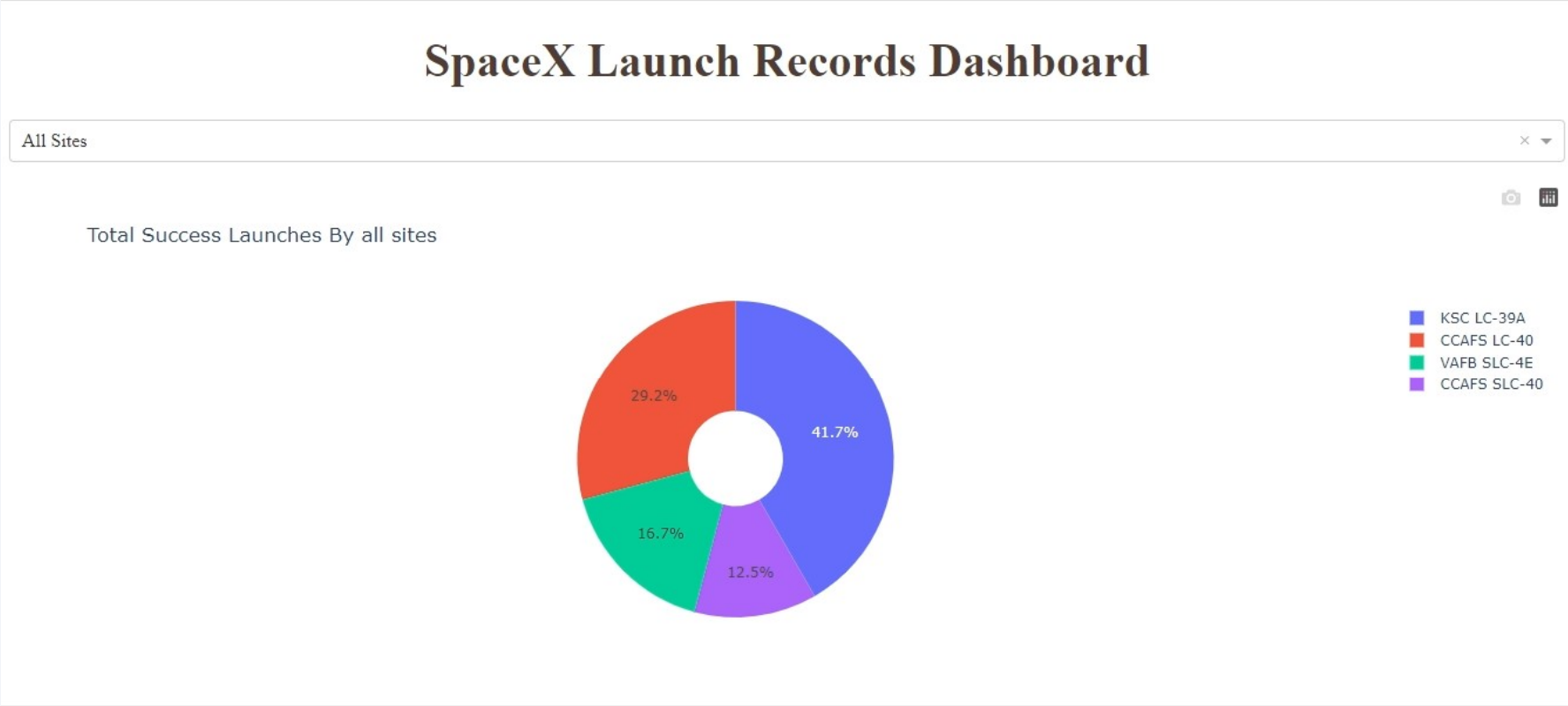


Distance of Railway from Launch site

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

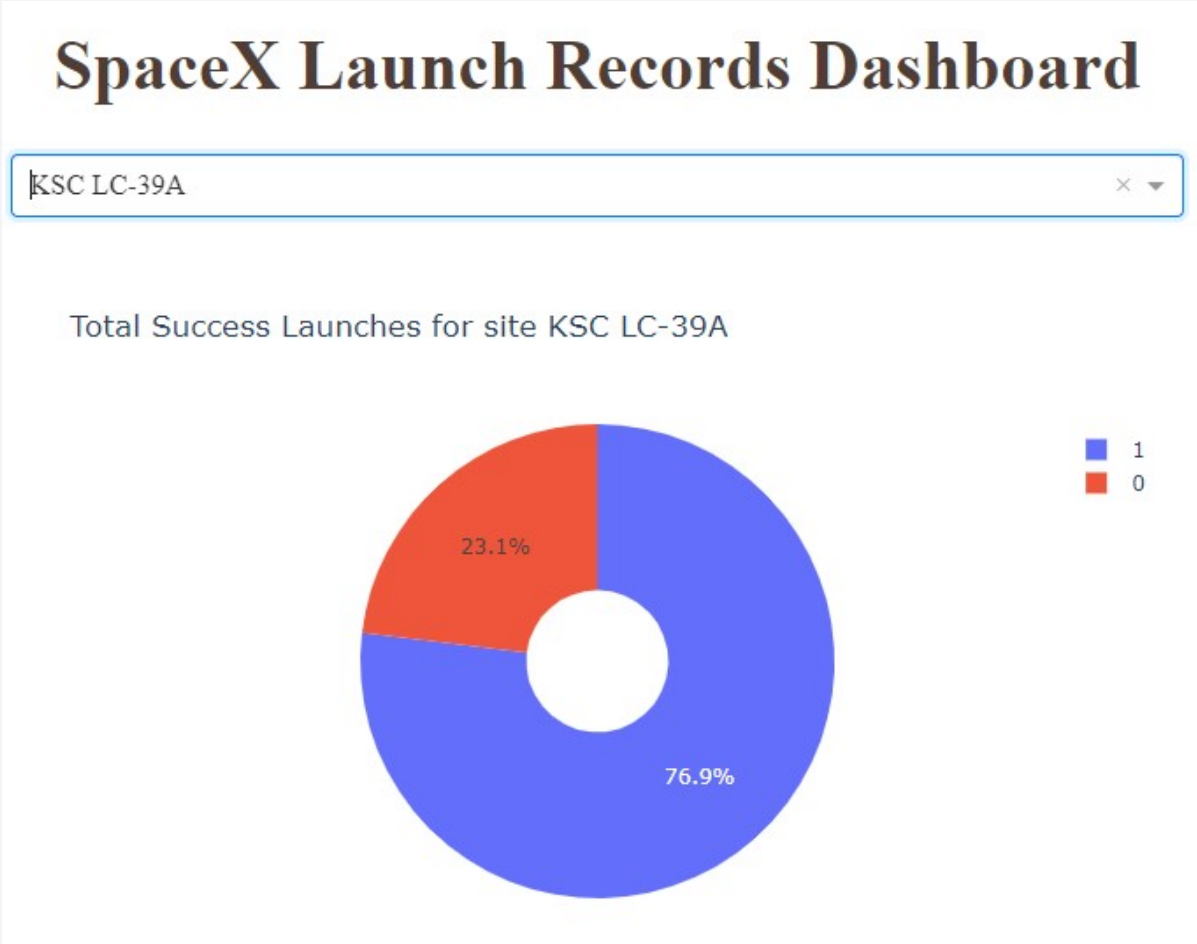

Section 4
Build a Dashboard with Plotly Dash

The success percentage achieved by each launch site



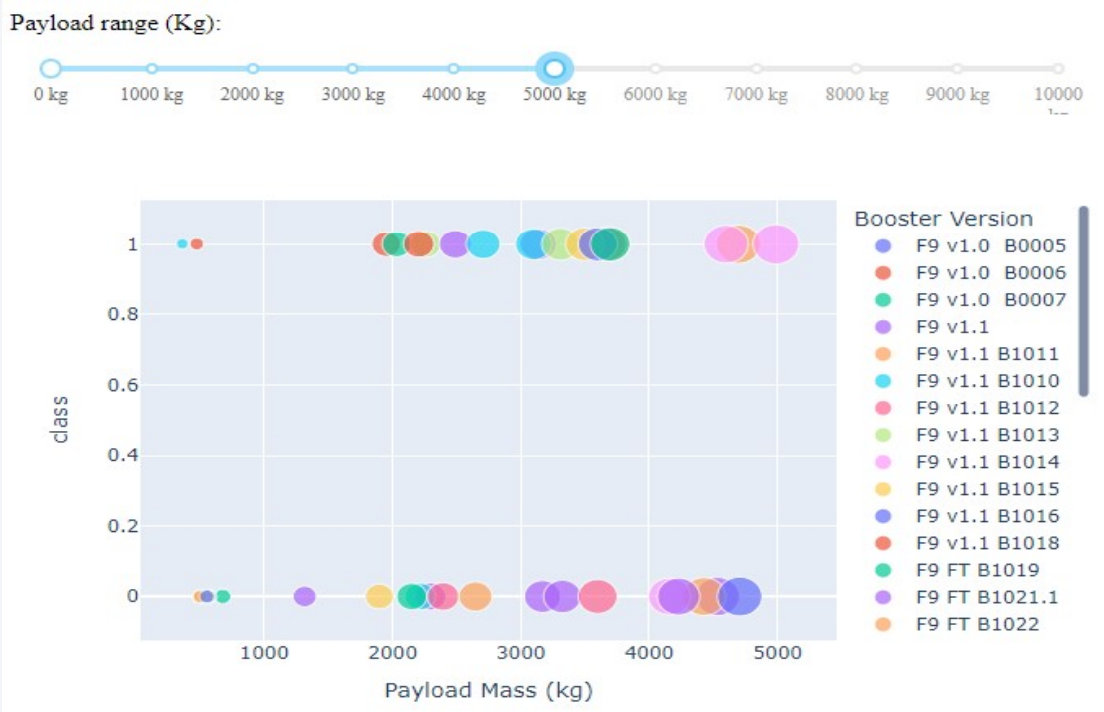
We can see that KSC LC-39A had the most successful launches among all sites

The Launch site with the highest launch success ratio

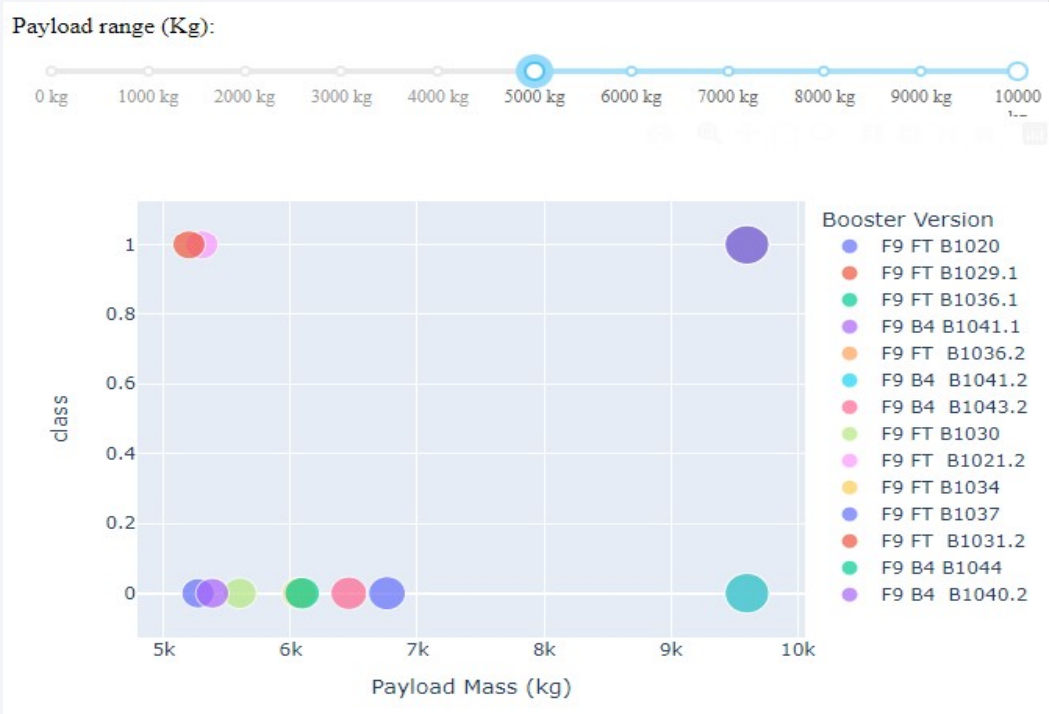


It is seen that KSC LC-39A is having highest i.e 76.9% success rate

Scatter plot of Payload vs Launch Outcome for all sites, with different payload selected in the range slider



Low payload

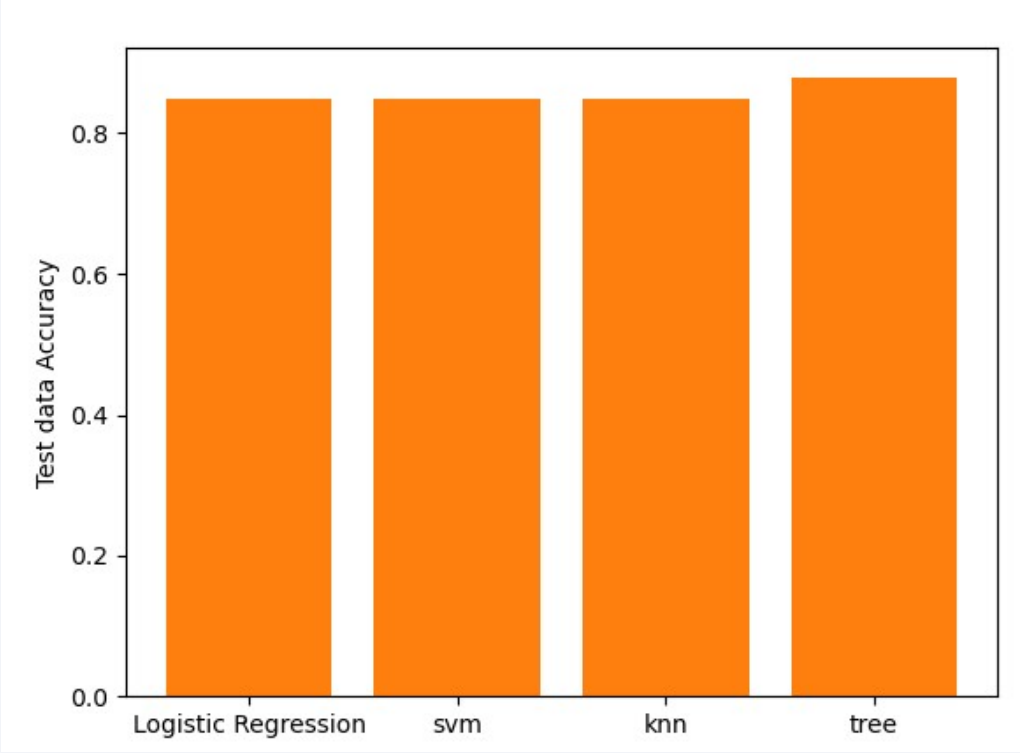


High payload

Section 5

Predictive Analysis (Classification)

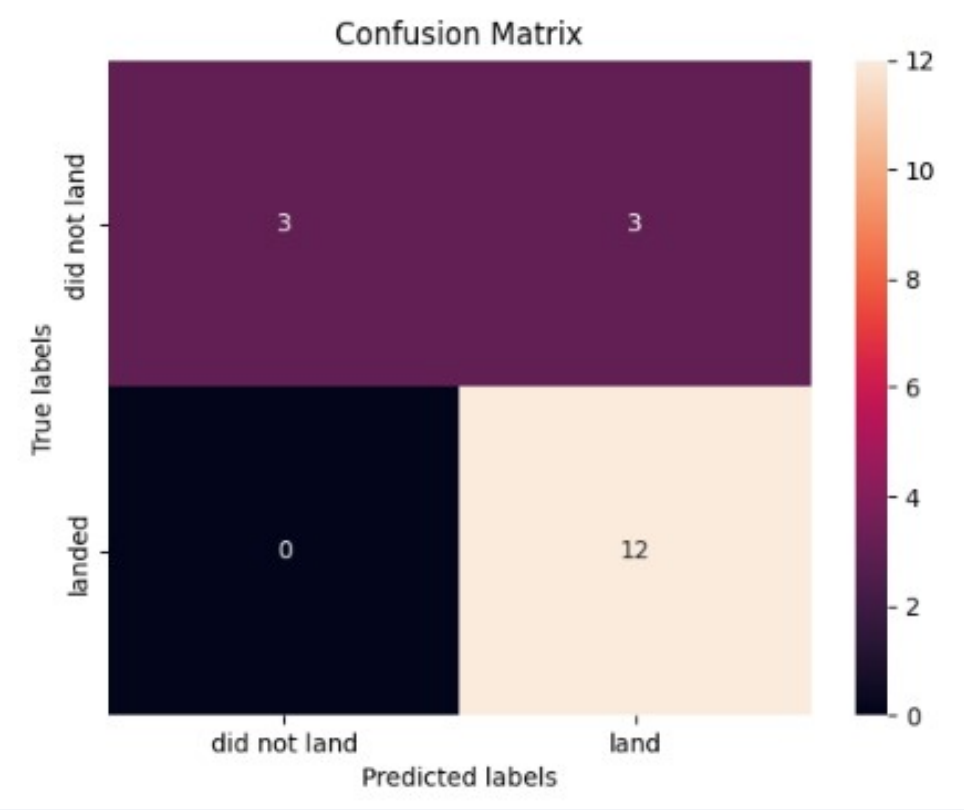
Classification Accuracy



- It is seen that Decision tree has highest accuracy

Confusion Matrix

- The major problem is the false positives .i.e., unsuccessful landing marked as successful landing by the classifier.



Conclusions

- The success rate at launch site is more for large flight amount at a launch site
- Launch success rate started to increase in 2013 till 2020.
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the more success rate.
- KSC LC-39A had the most successful launches on any sites.
- Decision tree model gives the best result

Thank you!