IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

João Marcos P. P. Salles
08/04/2023

Outline

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

Executive Summary

For this project we used two methods of data collection: direct connection with the SpaceX Falcon 9 data API and data wrangling from the Wikipedia pages.

After the data collection phase, we proceeded with the wrangling phase where the data was checked and prepared for the visualization phase and predictive models ahead.

In the next phase we uploaded the data set to a larger database where we could perform exploratory SQL analysis and the proceed to the visualization phase.

In the visualization phase, we're able to check the validity, in a visual format, of the database in various forms and how it can impact the predictive models, while plotting them on various forms with various advanced graphic libraries.

Finally, we pass to the predictive modelling phase, where we test several models checking their accuracy and deciding on the best one to use to answer our questions.

Github link to all Jupyter notebooks and files: https://github.com/jmppsalles/IBM_Capstone

Introduction

- Project background and context

  Space X advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because Space X can reuse the first stage. Therefore, 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 space X for a rocket launch. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.

- Problems you want to find answers

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

Section 1

Methodology

Data Collection

- The data was collected using various methods
  - Data collection was done using get request to the SpaceX API and normalized into a pandas Dataframe;
  - Additional data was gathered by web scraping the Wikipedia page for Falcon 9 launch records with BeautifulSoup;
  - We then cleaned the data, checked for missing values and fill in missing values where necessary;
  - Finally, we saved the data on a .csv file for future use.
  - All files are saved on GitHub @ https://github.com/jmppsalles/IBM_Capstone

Data Collection – SpaceX API and Webscraping

- We used the get request to the SpaceX API to collect data, clean the requested data
- Used BeautifulSoup to scrape data from Wikipedia page of the Falcon 9.

(1) Connect to API and use get request

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

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

(2) Scrape data from Wikipedia page of Falcon 9 Rocket

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches
```

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

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

Data Collection – Data Wrangling

- Cleaned and wrangled data to a format that would be used in the later phases of data visualization and predictive analysis.
- Saved the data in a .csv format for future use.

(3) Cleaned data of unwanted Falcon 1 references and dealt with missing values

```
data_falcon9 = data[data['BoosterVersion']!='Falcon 1']
```

```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
```

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

```
# Calculate the mean value of PayloadMass column
mean_plm = data_falcon9['PayloadMass'].mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, mean_plm)
```

(4)Finally, saved the file in a .csv format for future references

```
df=pd.DataFrame(launch_dict)
df.to_csv('spacex_web_scraped.csv', index=False)
```

Insights from Queries with SQL

- We loaded the SpaceX dataset into the IBM SQL database by way of the Watson jupyter notebook interface
- We applied several SQL queries to get some insight:
  - The names of unique launch sites in the space mission.

```
%sql Select Unique Launch_Site from SpaceX
```

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

```
%%sql Select customer, sum(payload_mass_kg) as "Total Payload Mass" from
        (Select customer, payload_mass_kg from SpaceX
        where customer LIKE 'NASA (CRS)')
        GROUP BY CUSTOMER
```

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

```
%%sql Select BOOSTER_VERSION, AVG(payload_mass_kg) as "AVERAGE Payload Mass" from
        (Select BOOSTER_VERSION, payload_mass_kg from SpaceX
        where BOOSTER_VERSION LIKE 'F9 v1.1')
        GROUP BY BOOSTER_VERSION
```

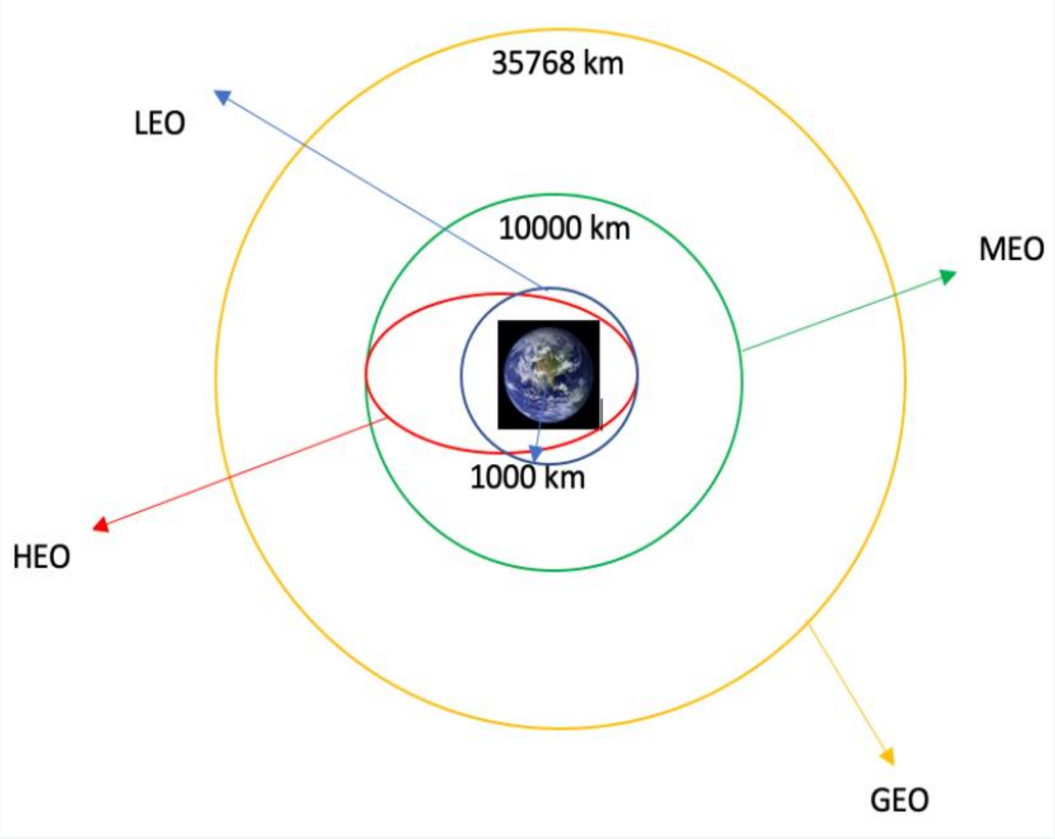
  - The total number of successful and failure mission outcomes

```
%%sql SELECT Mission_Outcome, count(Mission_Outcome) as "Total" FROM SPACEX
Group by Mission_Outcome
```

  - List the names of the booster versions which have carried the maximum payload mass.

```
%%sql SELECT Unique Booster_version, payload_mass_kg FROM SPACEX
where payload_mass_kg = (Select max(payload_mass_kg) from SPACEX)
```

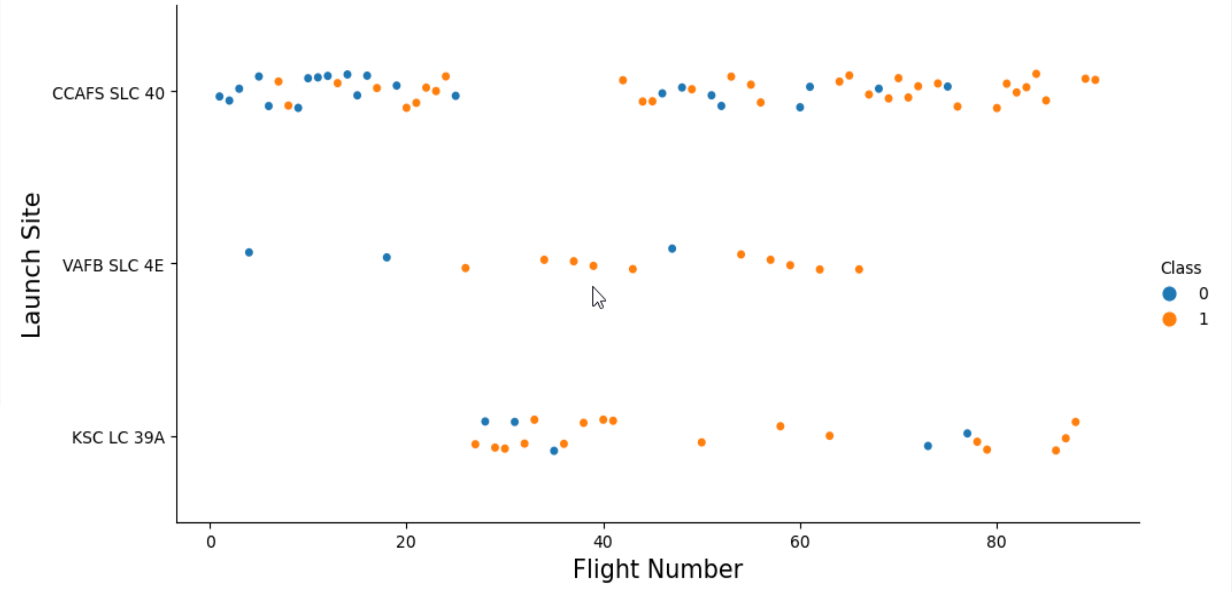
Visual Analytics

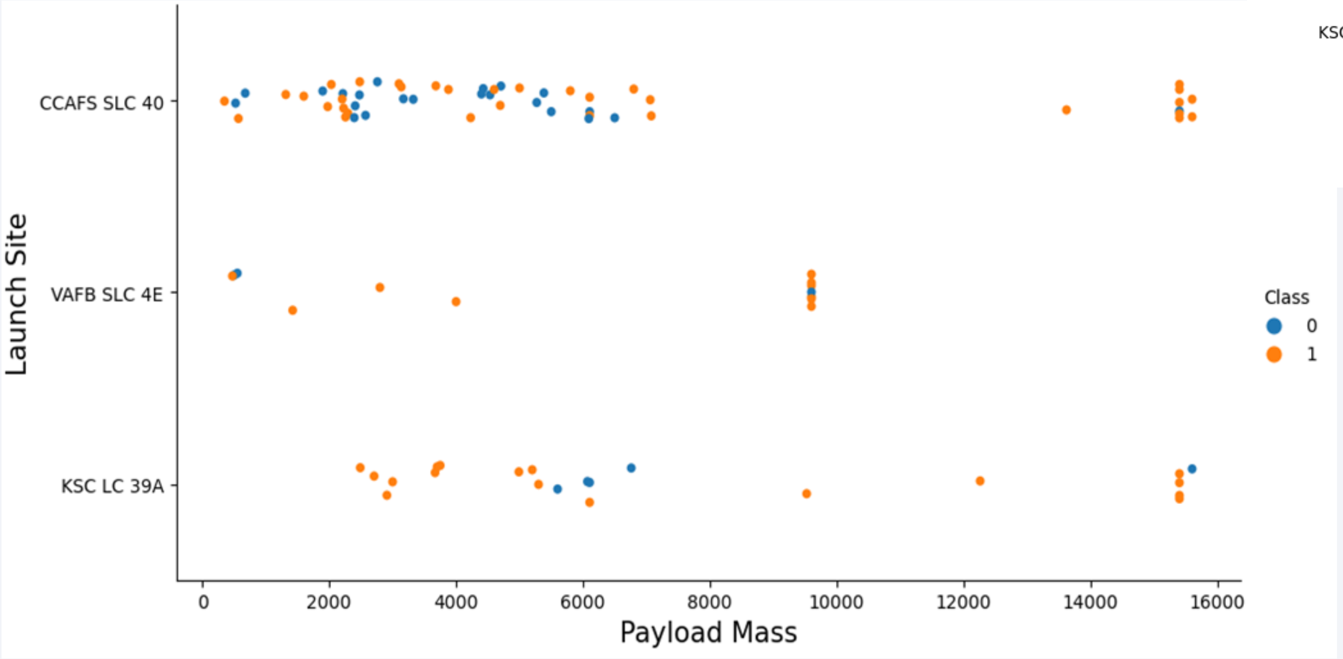


- Using the Numpy, Seaborn and Matplotlib libraries, we performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits (Low, Middle, High and Geosynchronous Earth Orbits)
- We then created a landing outcome label from outcome column and exported the results to a .csv file.
- We explored the data by visualizing the relationship between flight number and launch Site, payload and launch site, success rate of each orbit type, flight number and orbit type, the launch success yearly trend.

EDA with Data Visualization

- The result shows launch site CCFAS SLC 40 with a higher rate of success on lower and higher payloads, whereas KSC LC 39A has more success on middle range payloads.





Building an interactive map with Folium

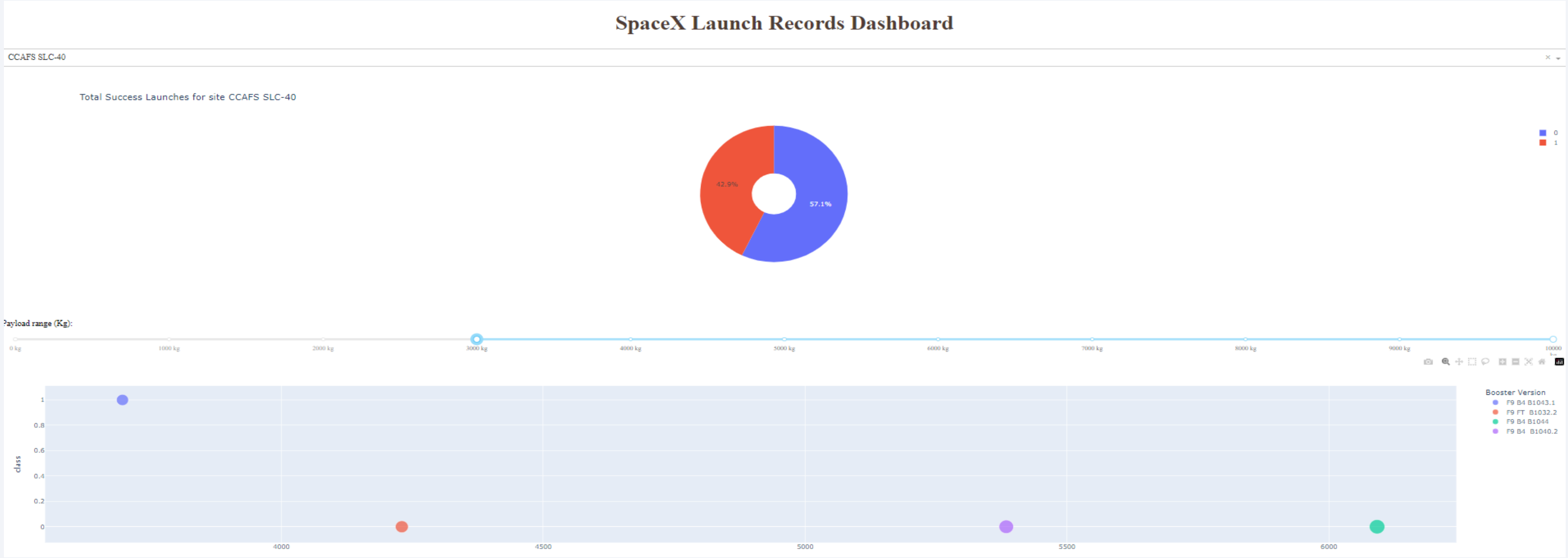
- After getting the geolocation markers for all the launch sites with the provided .csv file (spacex_launch_geo.csv), we also added the NASA JSC coordinates;
- Using the Folium library, we created a map where we marked all sites with each specific success or failure assigned to a 1 or a 0 (we also assigned green and red to the successes and failures respectively);
- Using these color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- We calculated the distances between a launch site to its proximities. We answered some question for instance:
    - Are launch sites near railways, highways and coastlines.
    - Do launch sites keep certain distance away from cities.

Building an interactive map with Folium (cont.)

- Interactive maps:

Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash
- We plotted pie charts showing the total launches by a certain sites
- We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.



Predictive Analysis (Classification)

- Using Pandas, Numpy, Matplotlib, Pyplot, Seaborn and Sklearn, we standardized the data, split it into training and testing sets.
- We chose the following predictive models to test: Logistic Regression (LogReg), K Nearest Neighbor (KNN), Decision Tree (Tree) and Support Vector Machine (SVM)
- 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.

Section 2

Insights drawn from EDA

First Successful Ground Landing Date

- From the Data gathered, we found that the first successful landing happened on December 22nd, 2015.

In [14]:

```
task_5 = '''
        SELECT MIN(Date) AS FirstSuccessfull_landing_date
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Success (ground pad)'
        '''
create_pandas_df(task_5, database=conn)
```

Out[14]:

	firstsuccessfull_landing_date
0	2015-12-22

Total Payload Mass

- Total payload carried for NASA is 45.6 tonnes.

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

In [12]:

```
task_3 = '''
        SELECT SUM(PayloadMassKG) AS Total_PayloadMass
        FROM SpaceX
        WHERE Customer LIKE 'NASA (CRS)'
        '''
create_pandas_df(task_3, database=conn)
```

Out[12]:

	total_payloadmass
0	45596

Average Payload Mass by F9 v1.1

- Average payload mass carried by Falcon9 v1.1 is 2,928.4 Kg

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

In [13]:

```
task_4 = '''
        SELECT AVG(PayloadMassKG) AS Avg_PayloadMass
        FROM SpaceX
        WHERE BoosterVersion = 'F9 v1.1'
        '''
create_pandas_df(task_4, database=conn)
```

Out[13]:

	avg_payloadmass
0	2928.4

Successful Drone Ship Landing with Payload between 4000 and 6000

```
In [15]:
task_6 = '''
        SELECT BoosterVersion
        FROM SpaceX
        WHERE LandingOutcome = 'Success (drone ship)'
            AND PayloadMassKG > 4000
            AND PayloadMassKG < 6000
        '''
create_pandas_df(task_6, database=conn)
```

Out[15]:

	boosterversion
0	F9 FT B1022
1	F9 FT B1026
2	F9 FT B1021.2
3	F9 FT B1031.2

- These are the successful booster landings on drone ships, with mass between 4,000 and 6,000 Kg

Boosters Carried Maximum Payload

- The maximum payload for the Falcon 9 B5 booster subtypes is 15,600 Kg.

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

In [17]:

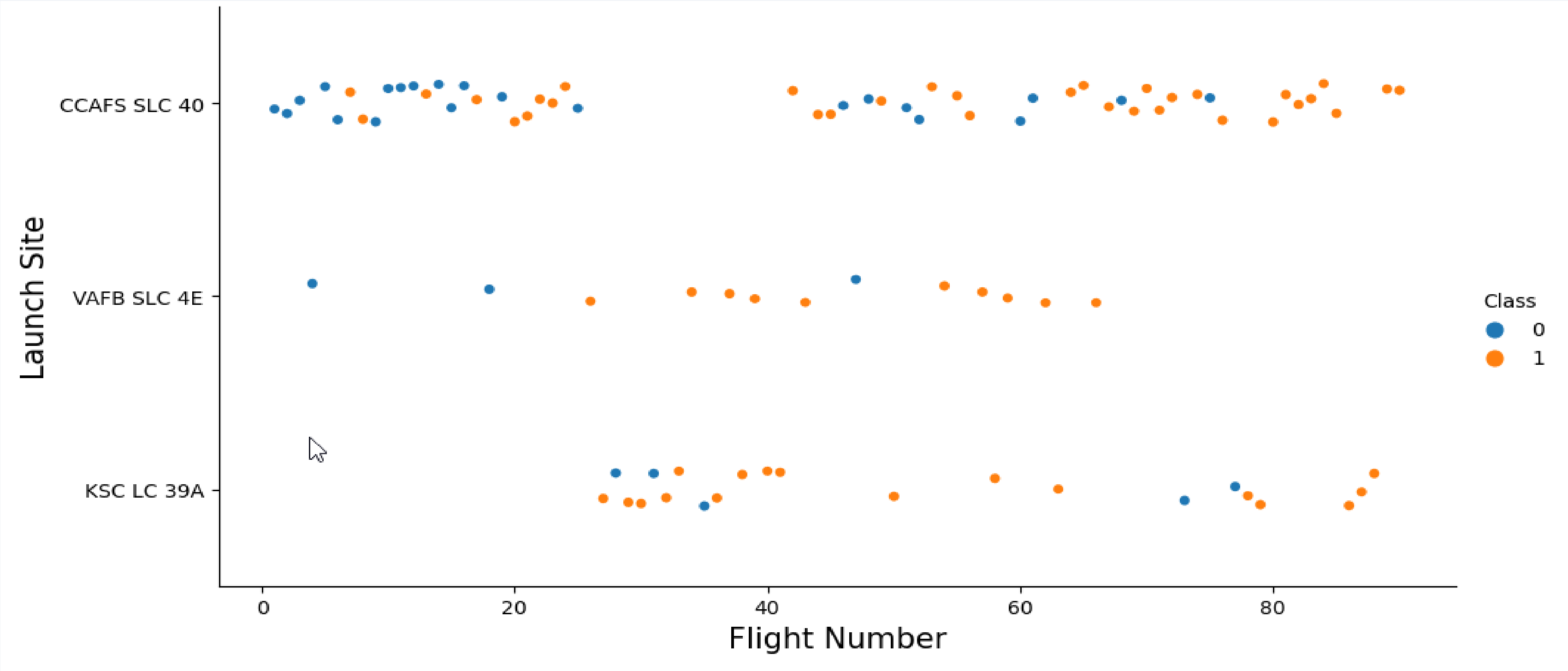
```
task_8 = '''
        SELECT BoosterVersion, PayloadMassKG
        FROM SpaceX
        WHERE PayloadMassKG = (
                                SELECT MAX(PayloadMassKG)
                                FROM SpaceX
                                )
        ORDER BY BoosterVersion
        '''
create_pandas_df(task_8, database=conn)
```

Out[17]:

	boosterversion	payloadmasskg
0	F9 B5 B1048.4	15600
1	F9 B5 B1048.5	15600
2	F9 B5 B1049.4	15600
3	F9 B5 B1049.5	15600
4	F9 B5 B1049.7	15600
5	F9 B5 B1051.3	15600
6	F9 B5 B1051.4	15600
7	F9 B5 B1051.6	15600
8	F9 B5 B1056.4	15600
9	F9 B5 B1058.3	15600
10	F9 B5 B1060.2	15600
11	F9 B5 B1060.3	15600

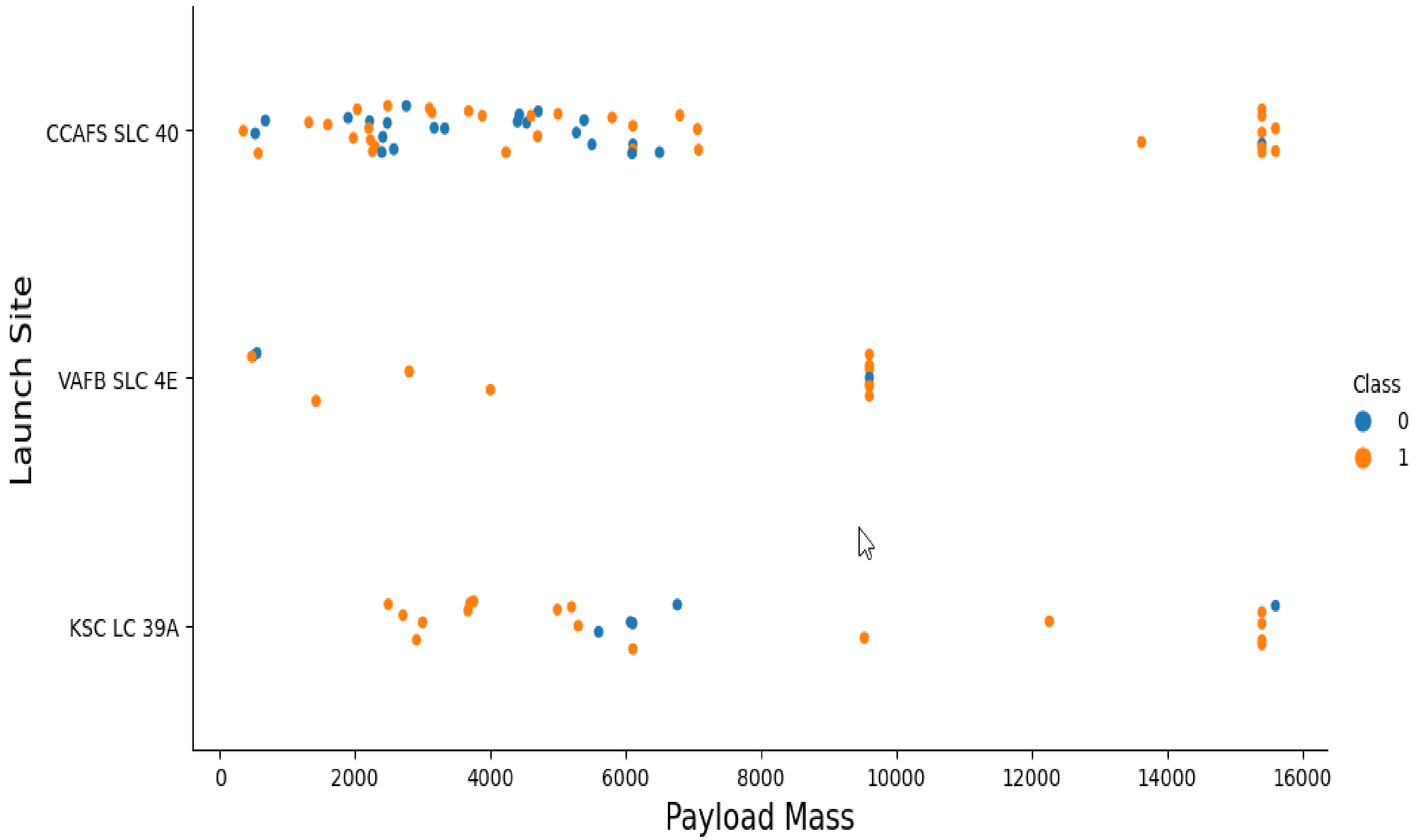
Flight Number vs. Launch Site

- From the plot, we found that the larger the flight amount at a launch site, the greater the success rate at a launch site.



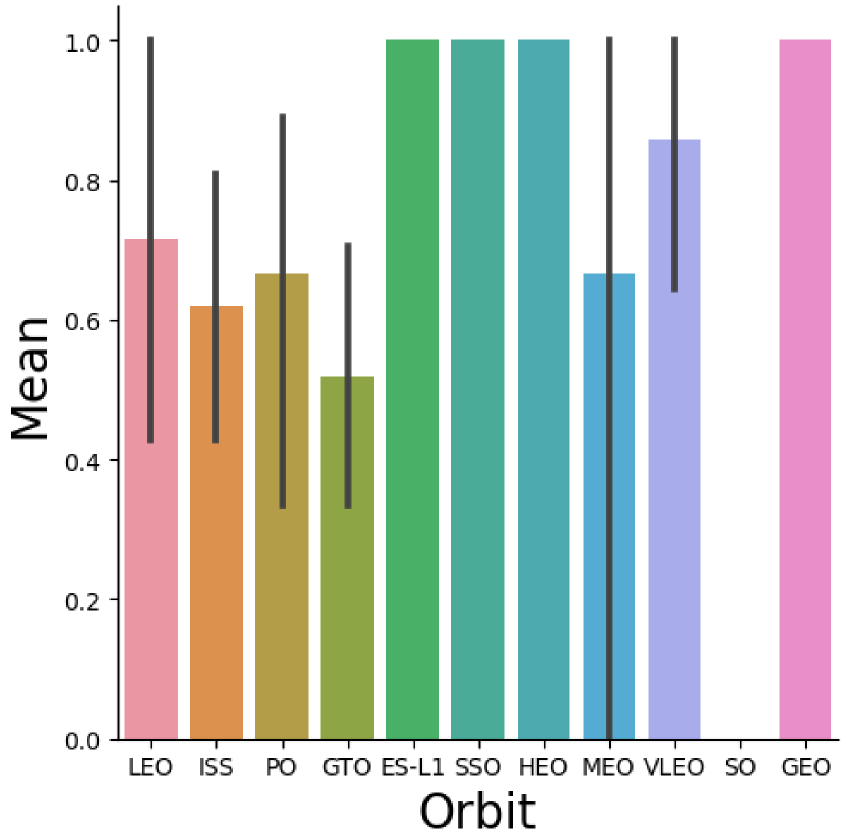
Payload vs. Launch Site

- The higher the payload, the higher the chance for a successful landing
- Launch site CCAFS SLC 40 has the higher rate of successful rate of landings per payload mass.
- Together with the previous slide, we can come to the conclusion that launch site CCAFS SLC 40 is preferable to the other sites.



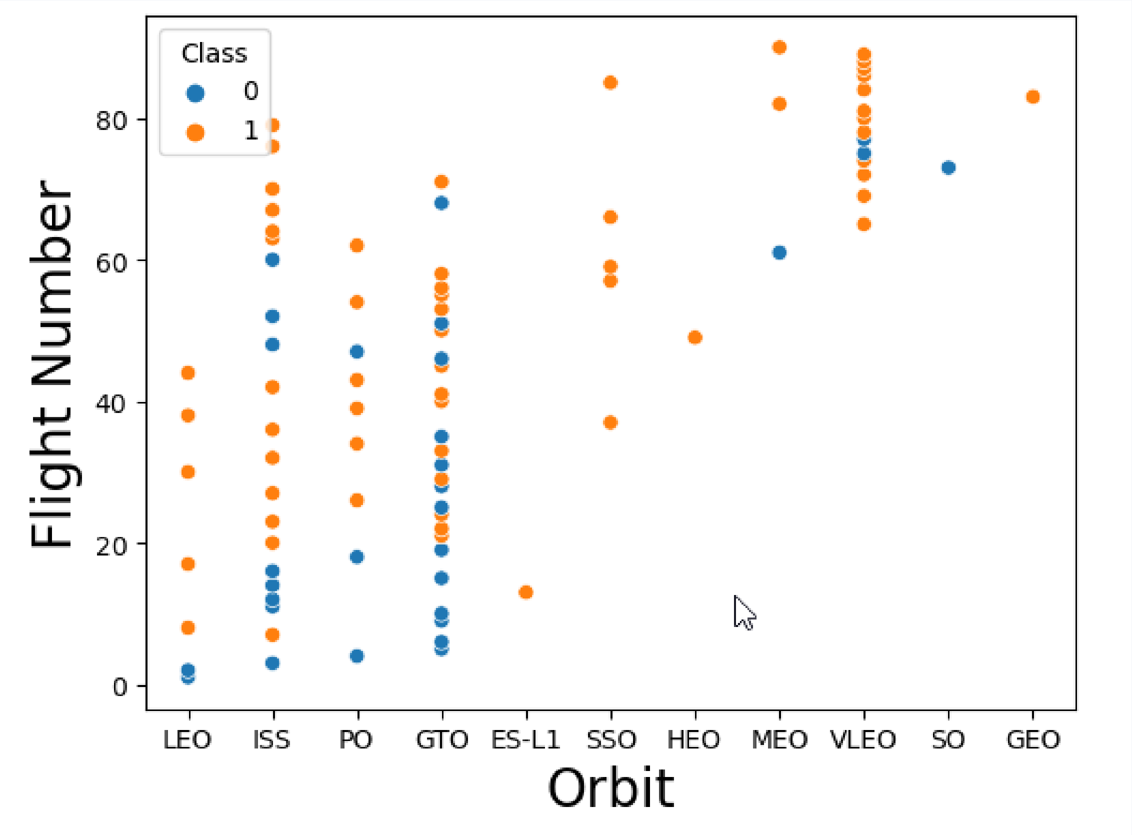
Success Rate vs. Orbit Type

- From this graph, we can conclude that missions targeting ES-L1, GEO, HEO, SSO, VLEO had the most success rate.



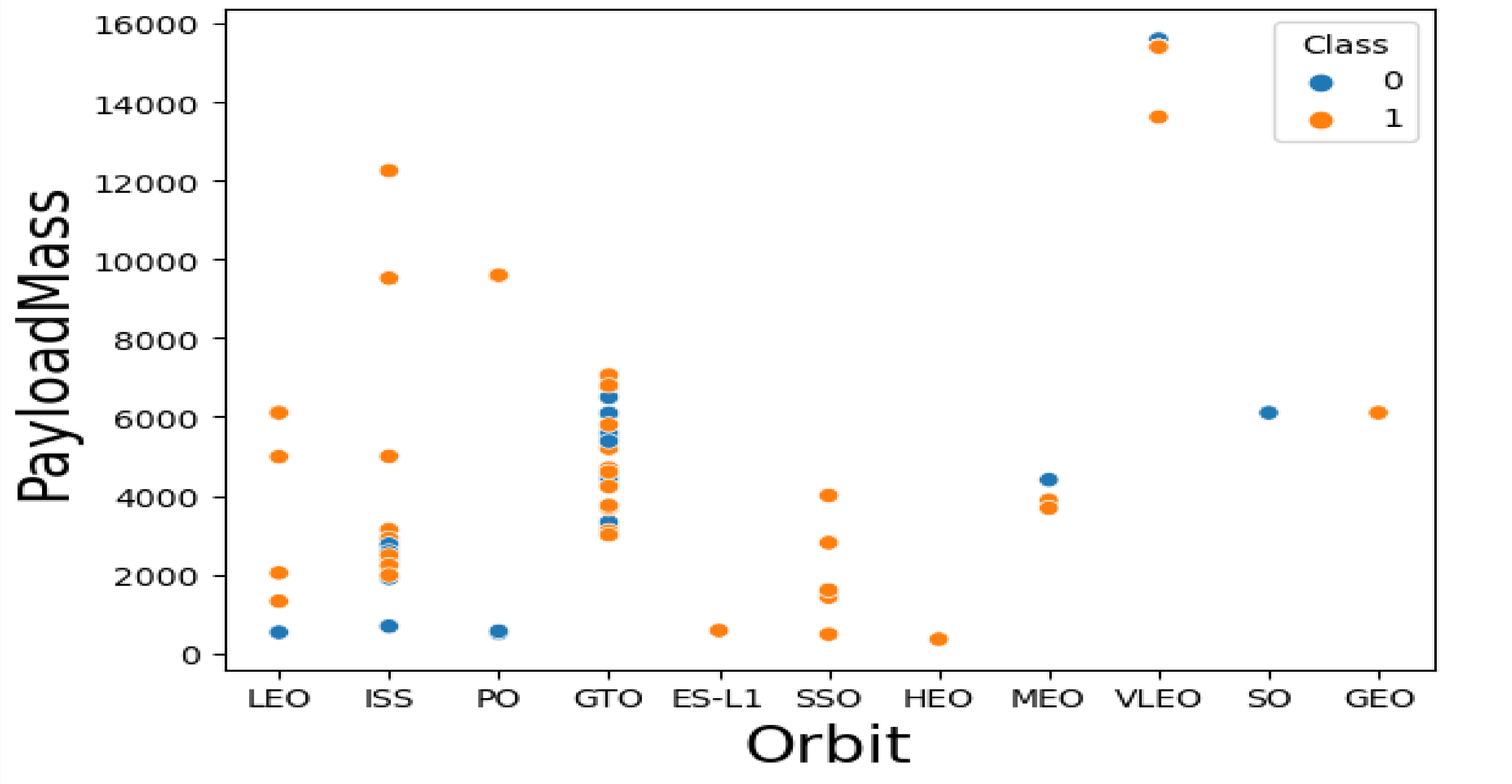
Flight Number vs. Orbit Type

- The graph below shows the Flight Number vs. Orbit type. Here we can see that in the VLEO and LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.



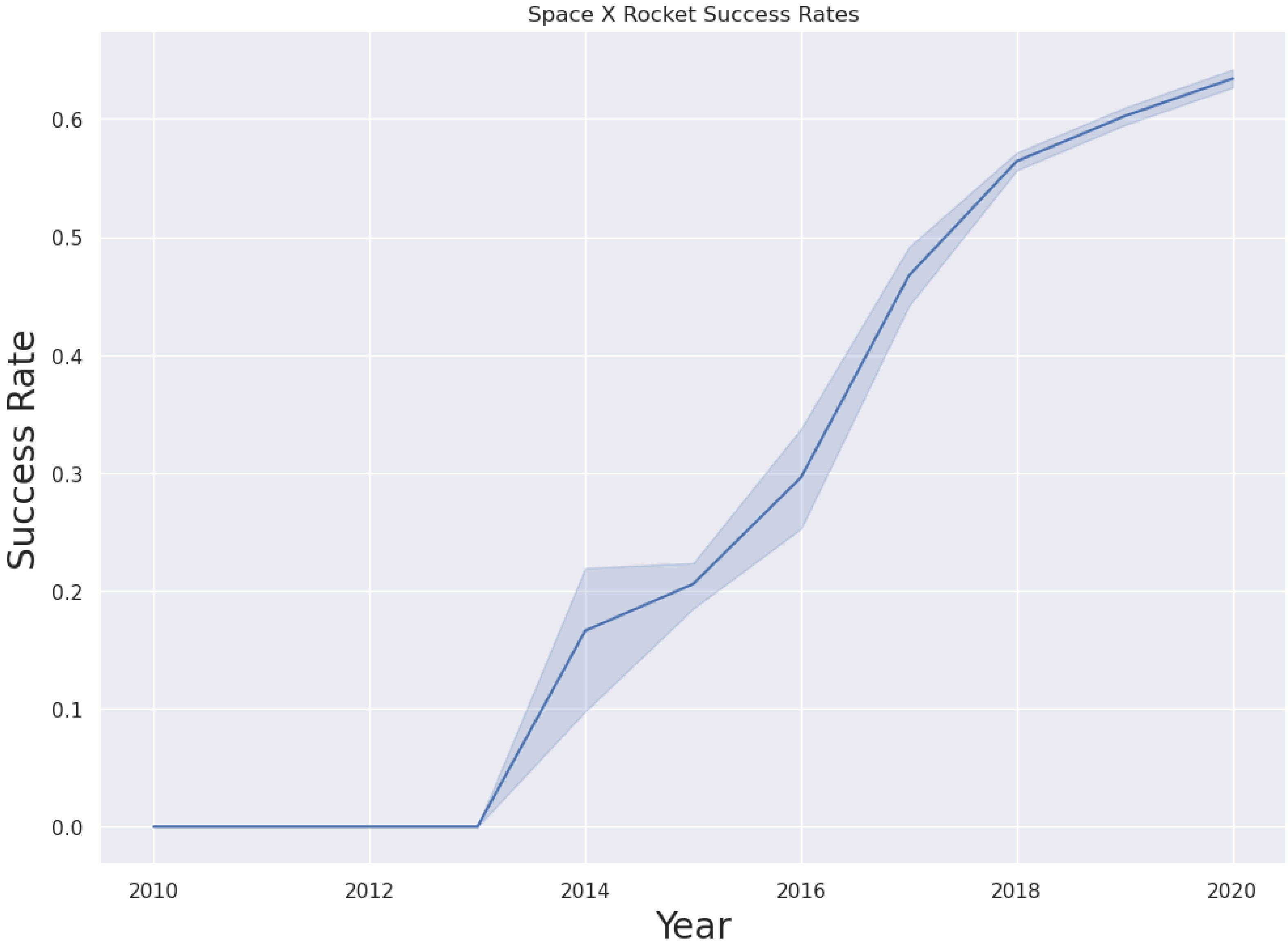
Payload vs. Orbit Type

- We can observe that with heavy payloads, the successful landing are more for PO, LEO and ISS orbits.



Launch Success Yearly Trend

- From this graph, we can observe that success rate since 2014 kept on increasing until 2020, more than tripling in that period



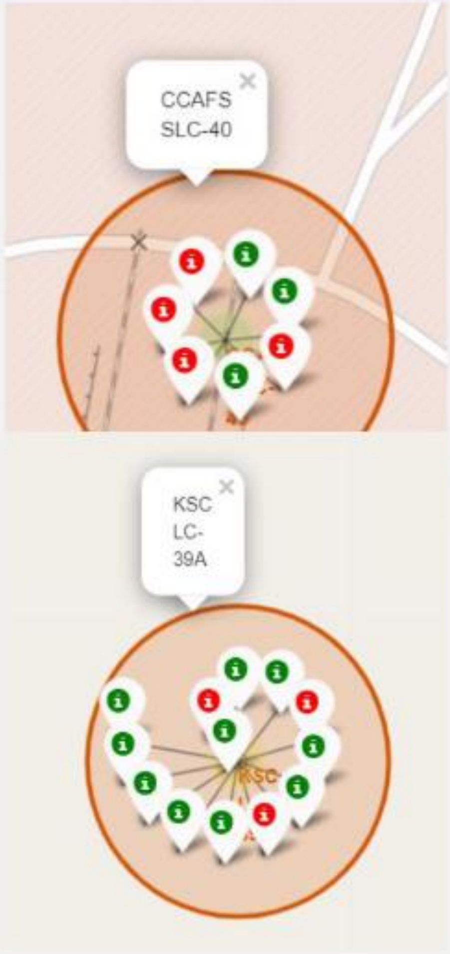
Section 4

Launch Sites Proximities Analysis

All launch sites map markers

Launch sites situated on both coasts

Markers showing launch sites with color labels



Florida Launch Sites

Green Marker shows successful Launches and Red Marker shows Failures

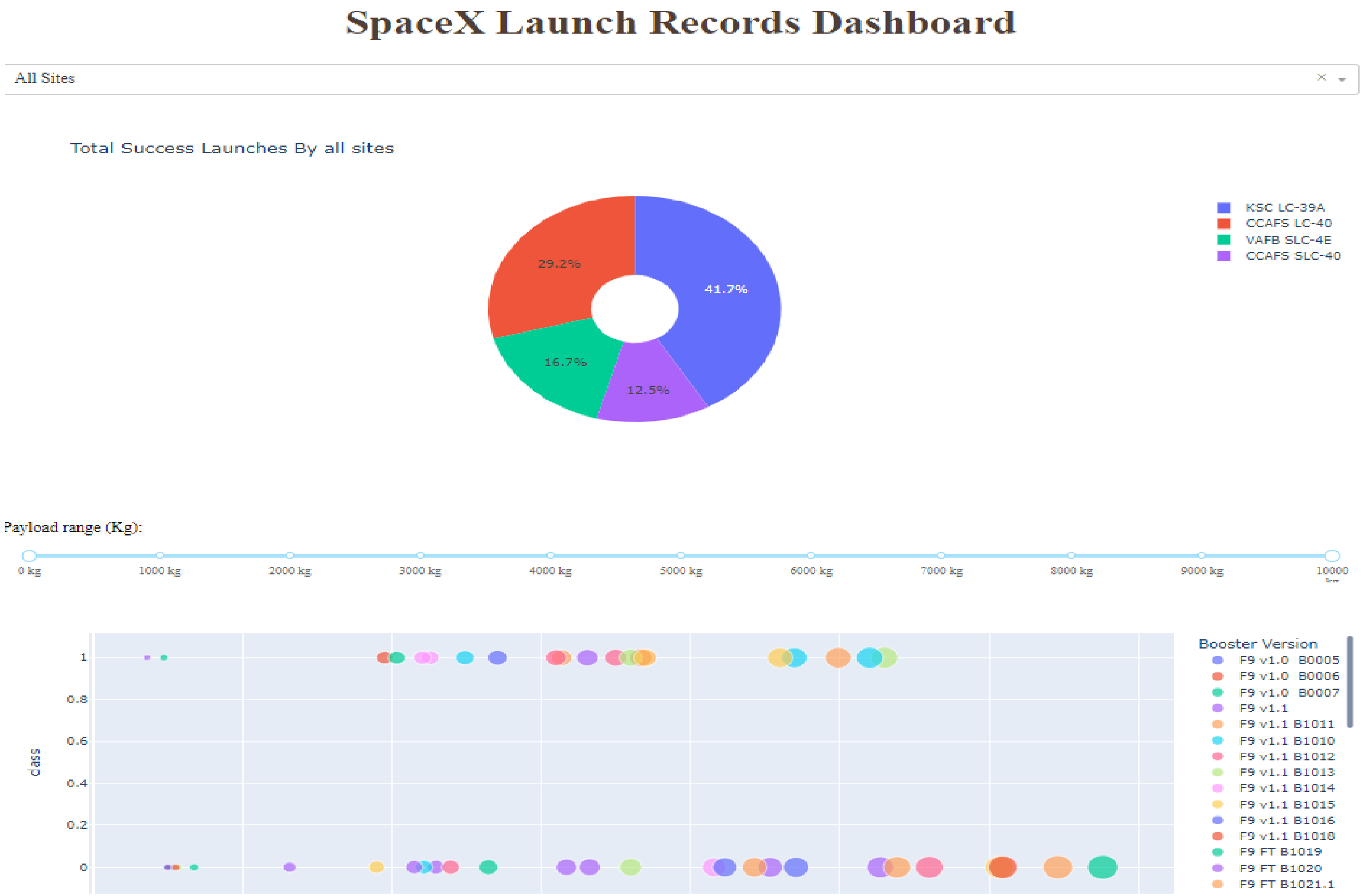
California Launch Site

37

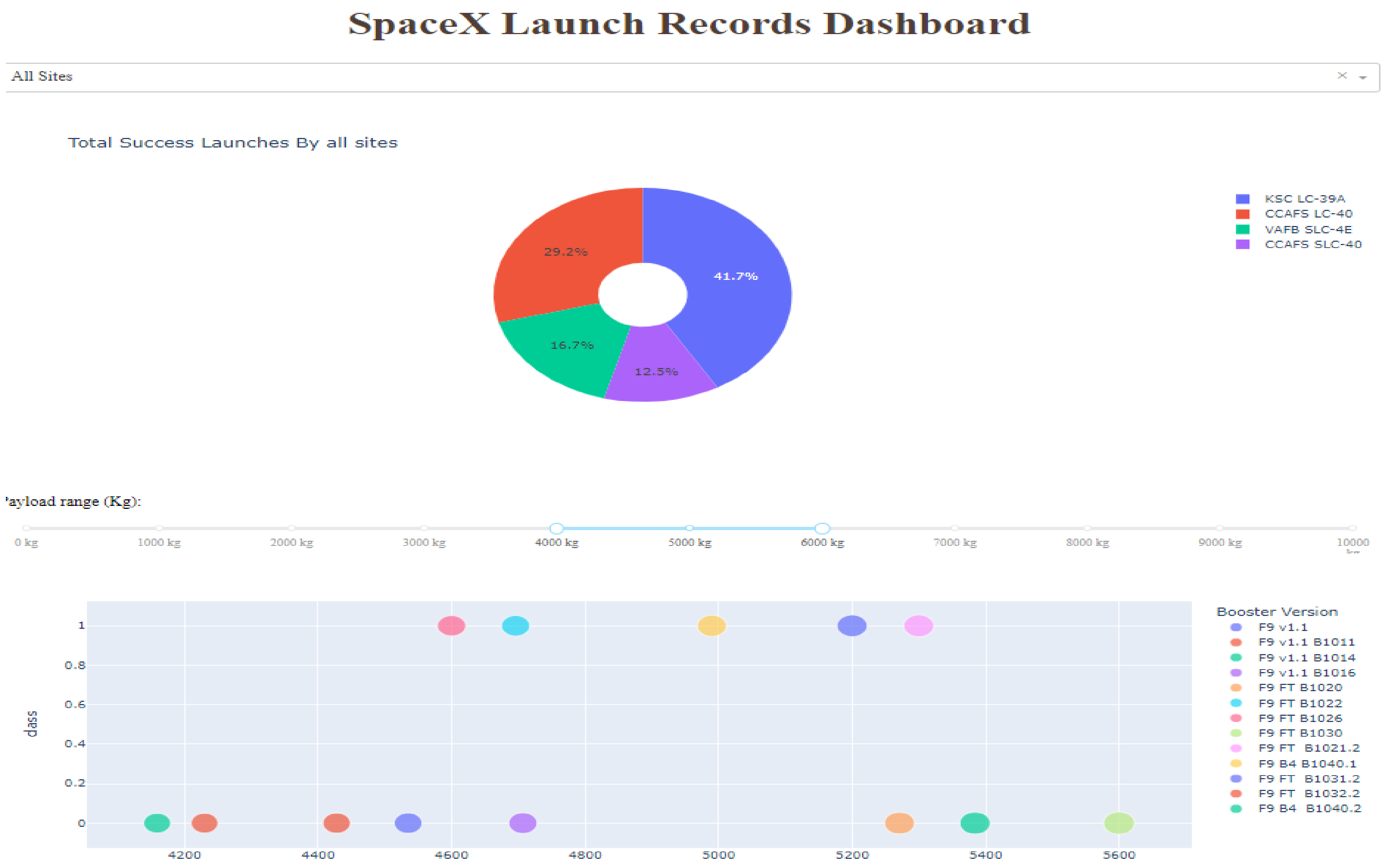
Section 5

Build a Dashboard with Plotly Dash

Complete dashboard showing all sites success rate and payload range

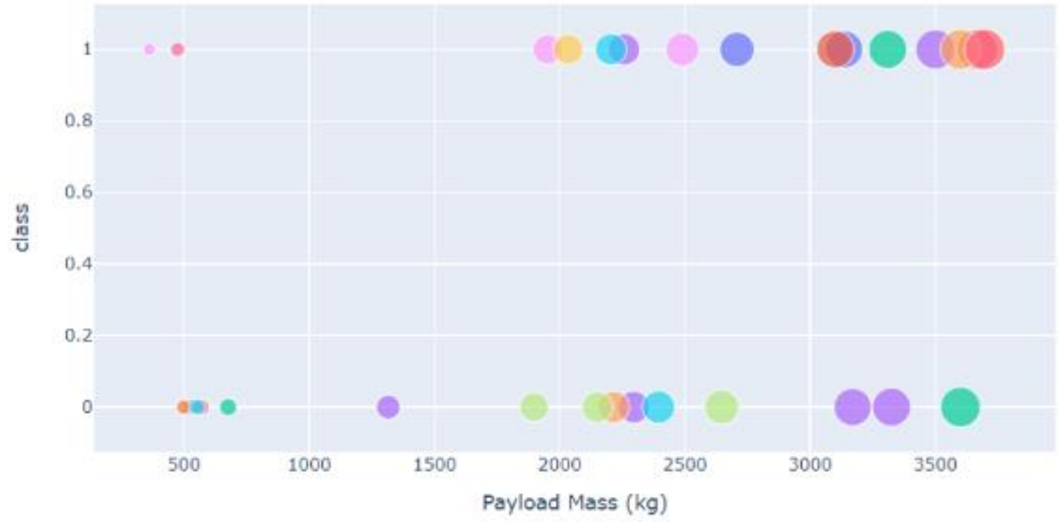


Complete dashboard for all sites with payload between 4,000 and 6,000 Kg

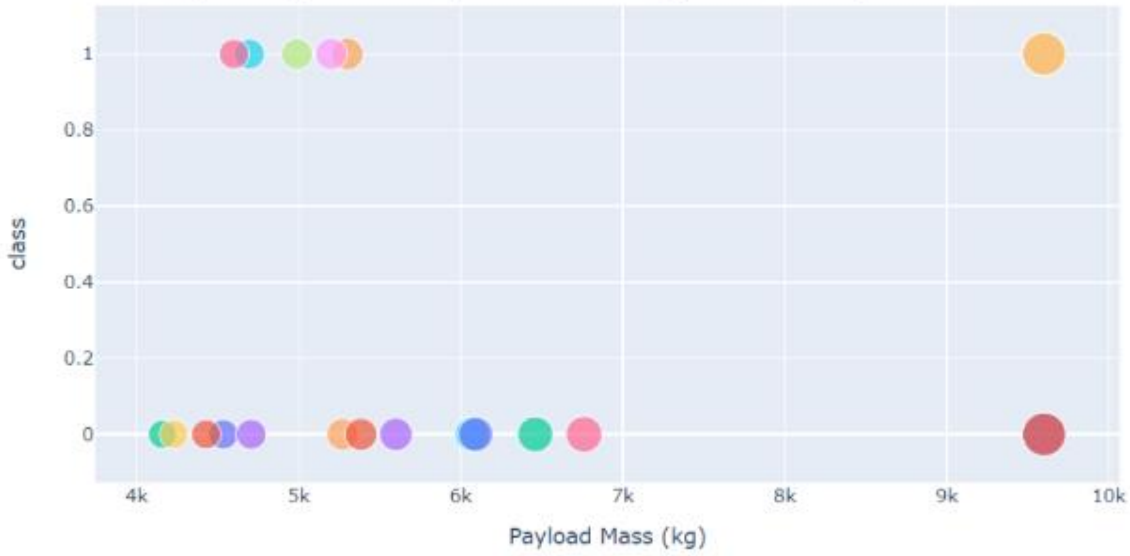


Scatter plot of Payload vs Launch Outcome for all sites, with different payload selected in the range slider

Low Weighted Payload 0kg – 4000kg

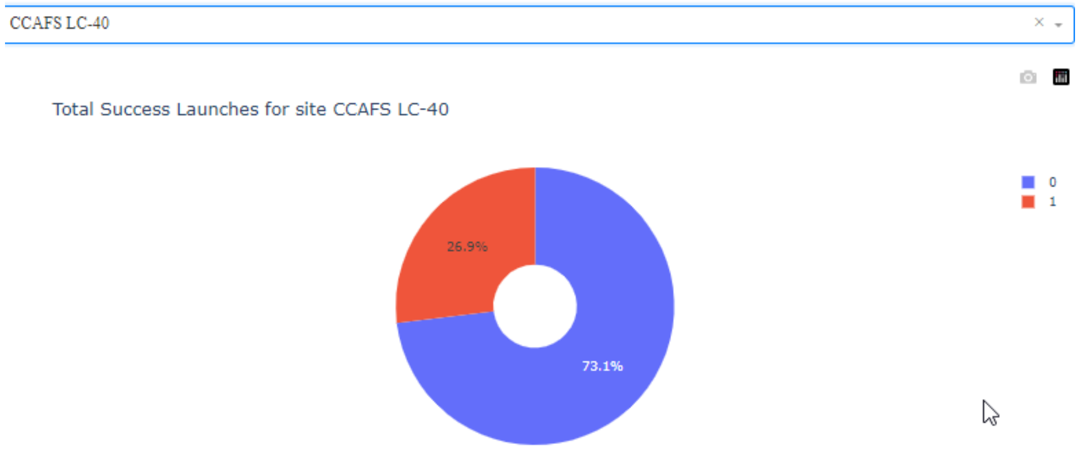


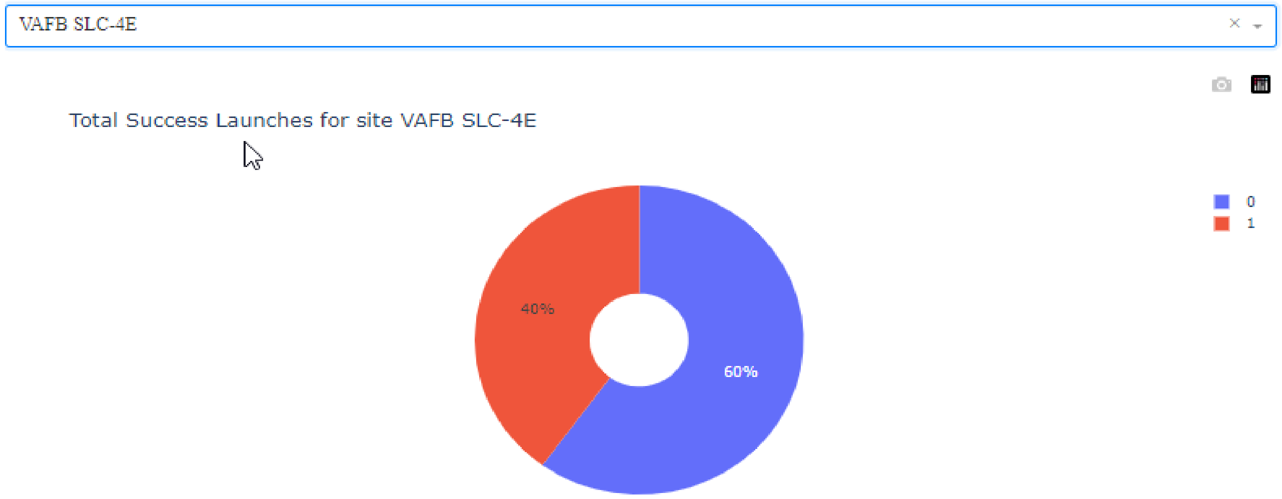
Heavy Weighted Payload 4000kg – 10000kg

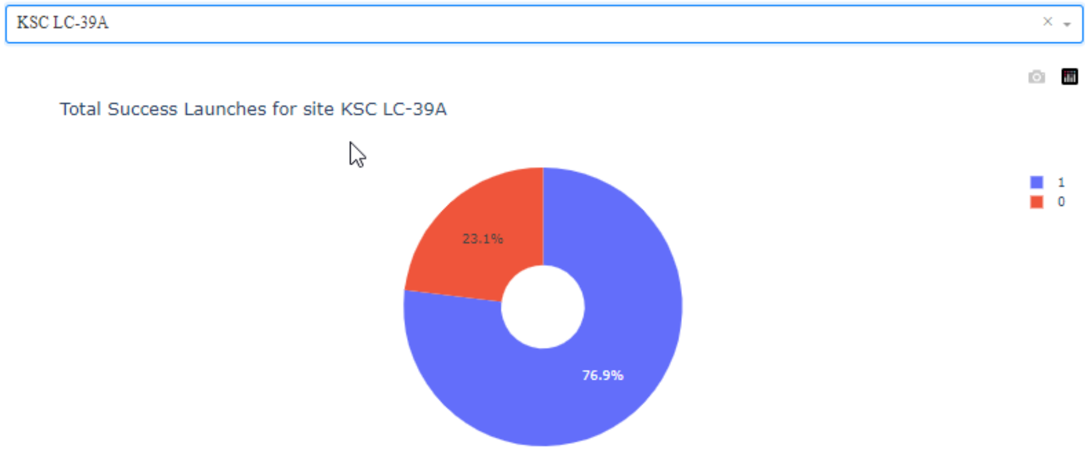


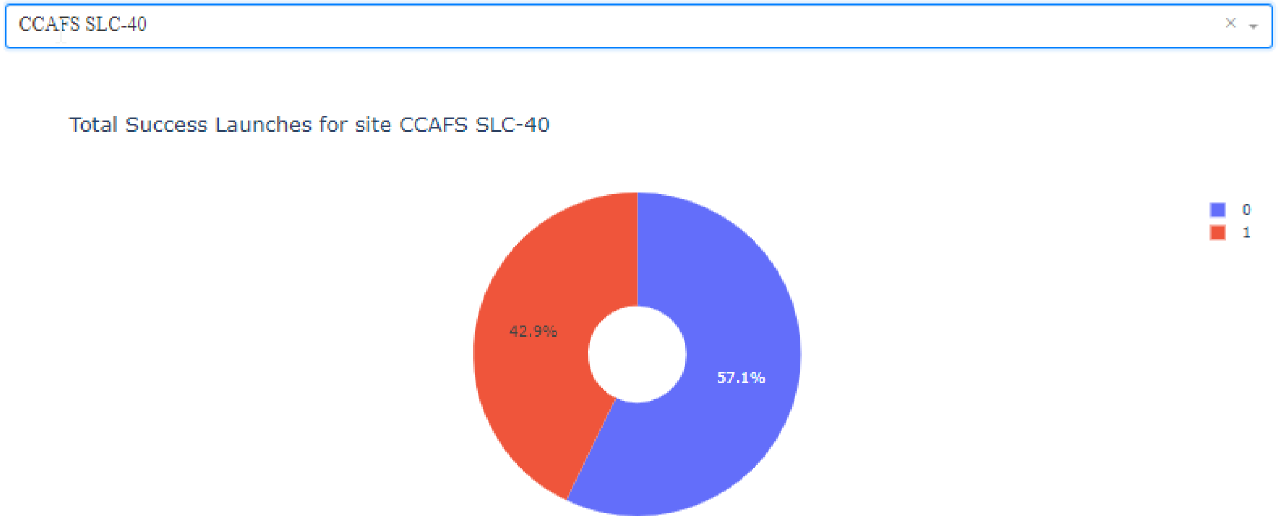
We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

Pie chart comparison of the four launch sites success rates.









Section 6

Predictive Analysis (Classification)

Predictive Analysis

- Predictive Models

- The models chosen for this analysis are:
  - Logistic Regression (LogReg)
  - K Nearest Neighbor (KNN)
  - Decision Tree Classifier (Tree)
  - Support Vector Machine (SVM)

- We split the data in train and test sets with a test size of 0.2 and random state of 0.2.
- Result shape of 18 samples.

Predictive Analysis

- Logistic Regression (LogReg)

- Using the following parameters:

```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
lr=LogisticRegression()
logreg_cv = GridSearchCV(lr,parameters, cv = 10)
logreg_cv.fit(X_train, Y_train)
```

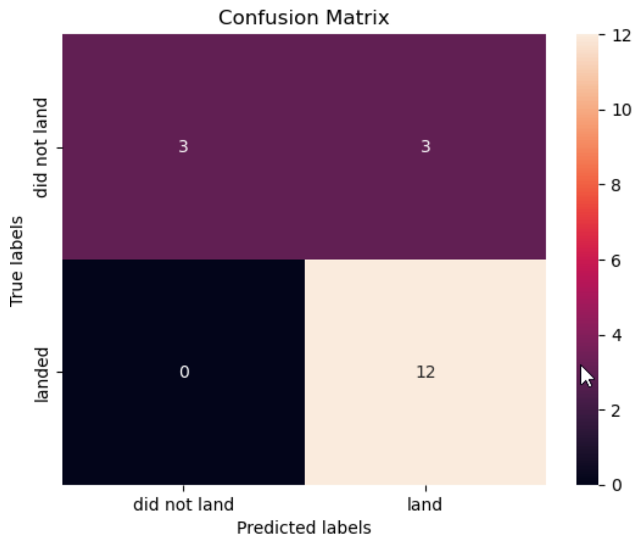
- We arrived at:

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

```
tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8464285714285713
```

```
lr_test_acc = logreg_cv.score(X_test, Y_test)
lr_test_acc
```

```
0.8333333333333334
```



Predictive Analysis

- Support Vector Machine (SVM)

- Using the following parameters:

```
parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
              'C': np.logspace(-3, 3, 5),
              'gamma':np.logspace(-3, 3, 5)}
svm = SVC()
```

```
svm_cv = GridSearchCV(svm,parameters, cv = 10)
svm_cv.fit(X_train, Y_train)
```

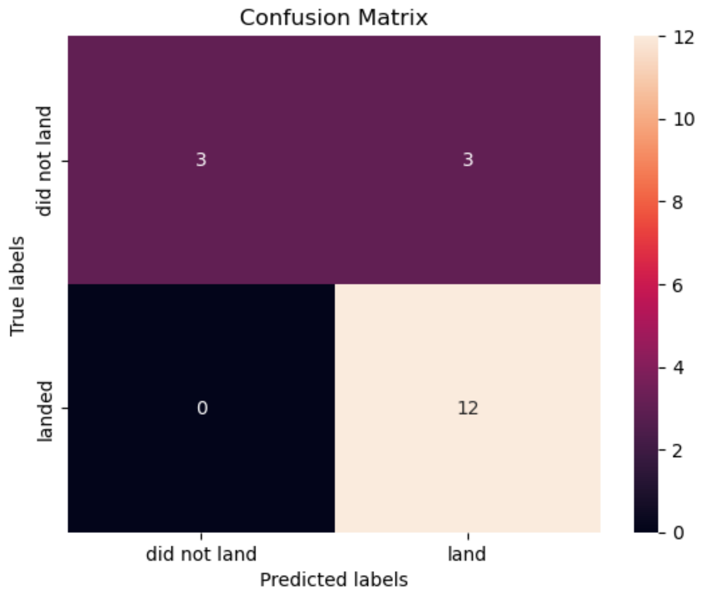
```
print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
print("accuracy :",svm_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
accuracy : 0.8482142857142856
```

```
svm_test_acc = svm_cv.score(X_test, Y_test)
svm_test_acc
```

```
0.8333333333333334
```

- We arrived at:



Predictive Analysis

- Decision Tree Classifier (Tree)

- Using the following parameters:

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

tree = DecisionTreeClassifier()
```

```
tree_cv = GridSearchCV(tree, parameters, cv=10)
tree_cv.fit(X_train, Y_train)
```

- We arrived at:

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

```
tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 4, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 10, 'splitter': 'rand
om'}
accuracy : 0.8767857142857143
```

```
tree_test_acc = tree_cv.score(X_test, Y_test)
tree_test_acc
```

```
0.7777777777777778
```

Predictive Analysis

- K Nearest Neighbor (KNN)

- Using the following parameters:

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

KNN = KNeighborsClassifier()
```

```
knn_cv = GridSearchCV(KNN, parameters, cv = 10)
knn_cv.fit(X_train, Y_train)
```

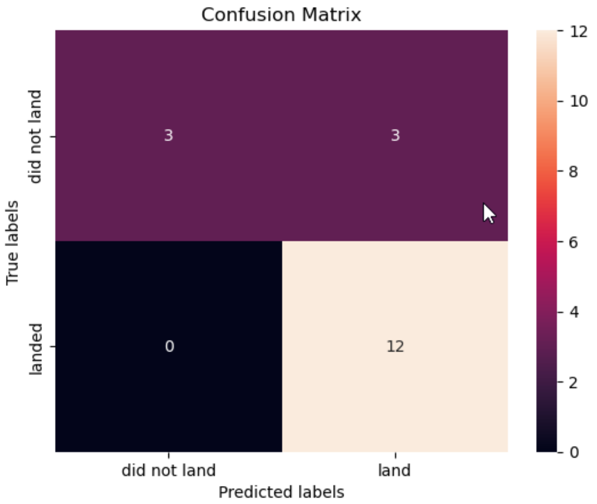
- We arrived at:

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

```
tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
accuracy : 0.8482142857142858
```

```
knn_test_acc = knn_cv.score(X_test, Y_test)
knn_test_acc
```

```
0.8333333333333334
```



Classification Accuracy

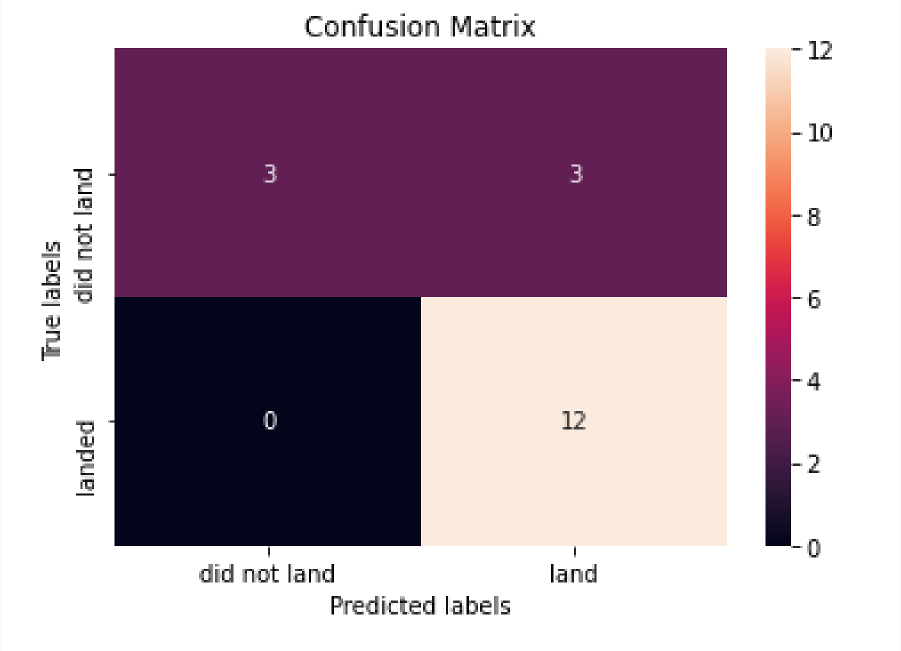
- The decision tree classifier is the model with the highest classification accuracy

```
[33]: mods = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogReg':logreg_cv.best_score_, 'SVM':svm_cv.best_score_}
best = max(mods, key=mods.get)
print('Best Model is',best,'with a score of', mods[best])
if best == 'Tree':
    print('Best Params is :',tree_cv.best_params_)
if best == 'KNN':
    print('Best Params is :',knn_cv.best_params_)
if best == 'LogReg':
    print('Best Params is :',logreg_cv.best_params_)
if best == 'SVM':
    print('Best Params is :',svm_cv.best_params_)
```

```
Best Model is Tree with a score of 0.875
Best Params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

Confusion Matrix

- We can see that all Confusion Matrices have pretty much the same numbers as the accuracy of the four models is very similar.
- The confusion matrix for the decision tree classifier shows that it can distinguish between the different classes. The major problem is the false positives .i.e., unsuccessful landing marked as successful landing.



Conclusions

We can conclude that:

- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- Preferable launch site is CCFS SLC-40, with the higher success landing rate;
- Launch success rate started to increase in 2014 and more than triples up to 2020;
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- The Decision tree classifier is the most suitable predictive model for this task.

Thank you!