SpaceX Capstone Data Science Track in IBM SE4R program

Sigi Karg

August 15, 2014

OUTLINE

- Executive Summary
- Introduction
- Methodology
- Results
  - Visualization – Charts
  - Dashboard
- Discussion
  - Findings & Implications
- Conclusion
- Appendix

EXECUTIVE SUMMARY

- List Methodology
- Results
  - Data Collection from SpaceX website and Wikipedia
  - Data Wrangling using Exploratory Data Analysis and feature engineering
  - Data Visualization
- Interactive Maps and Dashboards
- Predictive Analysis using Machine Learning Algorithms
- Conclude increase of success rate from 2015 on and
- Decision tree algorithm achieves highest scores (87%)

INTRODUCTION

Motivation and background

- **SpaceX Falcon 9** first stage landing prediction
- Cost savings because SpaceX can reuse the first stage.

Goal of the project

- Predict whether first stage of **SpaceX Falcon 9** will land successfully
- Use machine-learning (ML) algorithms to make predictions

METHODOLOGY

- Collecting data from SpaceX
- Web scraping records from Wikipedia
- Data wrangling performing Exploratory Data Analysis (EDA)
  - Exploring and Preparing Data
  - Data Visualization
  - SQL Database queries
- Interactive Visual Analytics
- Predictive analysis using Machine Learning algorithms

RESULTS – Data Collection

- Collecting data from SpaceX
  - Request data from RESTful API in JSON format
  - Normalize JSON data
  - Convert to Pandas dataframe
  - Replace missing data with mean value

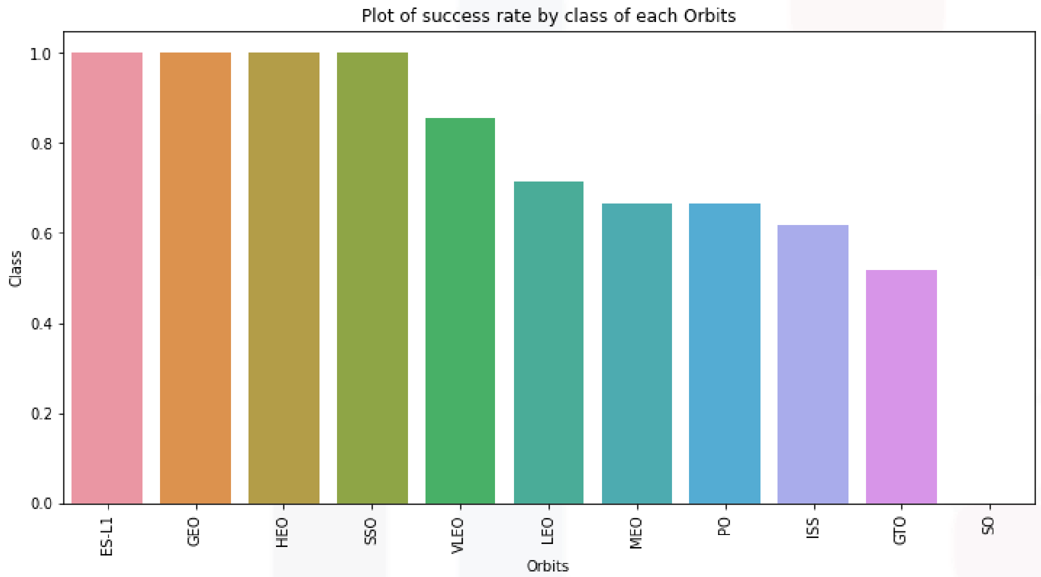
- Web scraping records from Wikipedia
  - Request Wikipedia Falcon 9 launch records
  - Use Python BeautifulSoup package to parse HTML titles and table headers
  - Convert to Pandas dataframe

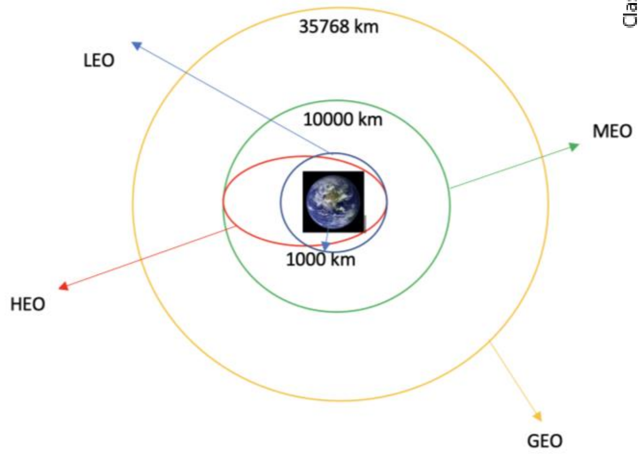
RESULTS – Data Wrangling

- Exploratory Data Analysis (EDA)
  - Find patterns in the data that could be used as label for training supervised ML models
  - Binary training labels indicating whether booster landed successfully or not
  - Setup Database and perform SQL queries

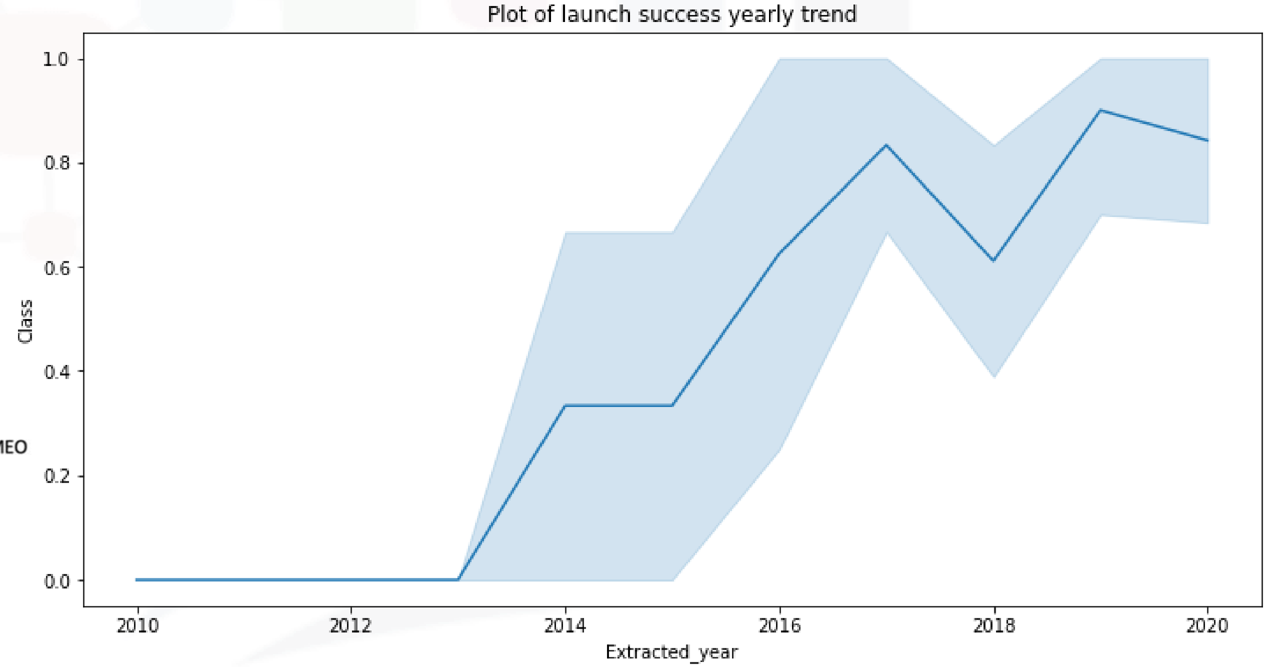
- Data preparation
  - Explore relationships between data features
  - Feature engineering
  - Investigate influence of PayloadMass, Orbit, or LaunchSite on mission success

RESULTS – Data Visualization



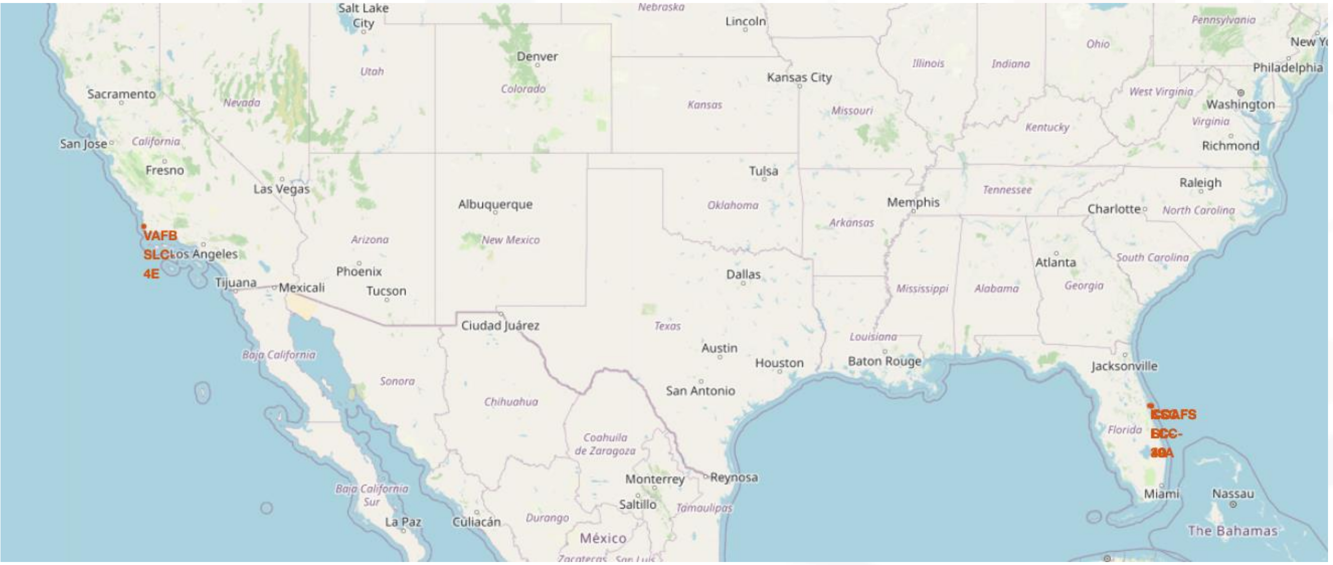


- Visualize interdependencies of features
  - Success rate vs. Orbit or launch year



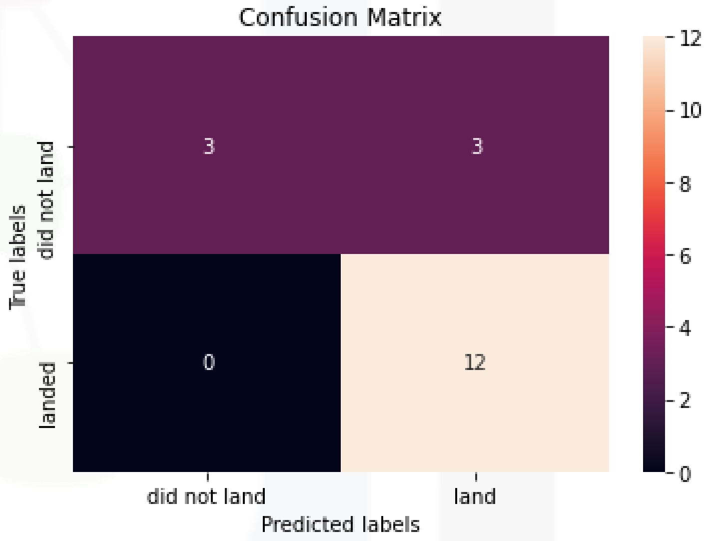
RESULTS – Map and Dashboard

- Interactive Visual Analytics with Folium
  - Mark all launch sites on a map
  - Mark the success/failed launches for each site on the map
  - Calculate the distances between a launch site to its proximities



RESULTS – Predictive Analysis

- Machine Learning Predictions
  - SVM, Classification Trees and Logistic Regression
  - GridSearch for hyperparameter optimization
  - Find the method performs best using test data
  - Accuracy metric
  - Confusion matrix
  - Classification trees achieve highest scores (87%)



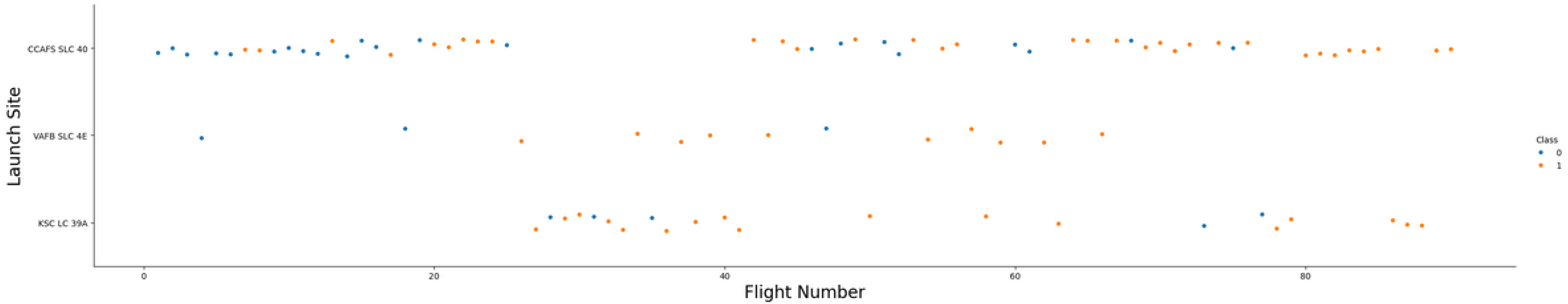
CONCLUSION

- The success rate of booster landings increased substantially from 2015 to 2020
- The probability to recover the booster is larger for orbits further away from earth
- Launch site KSC LC-39A is most successful one
- Decision tree classifier is the best machine learning algorithm for this task.

Detailed Insights drawn from EDA

Flight Number vs. Launch Site

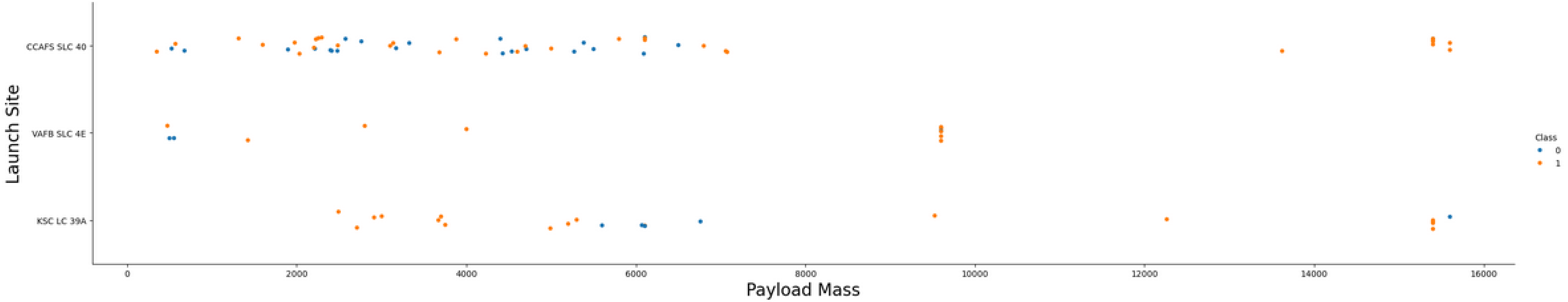
In [5]:
```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch Site", fontsize=20)
plt.show()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



Payload Mass vs. Launch Site

In [6]:

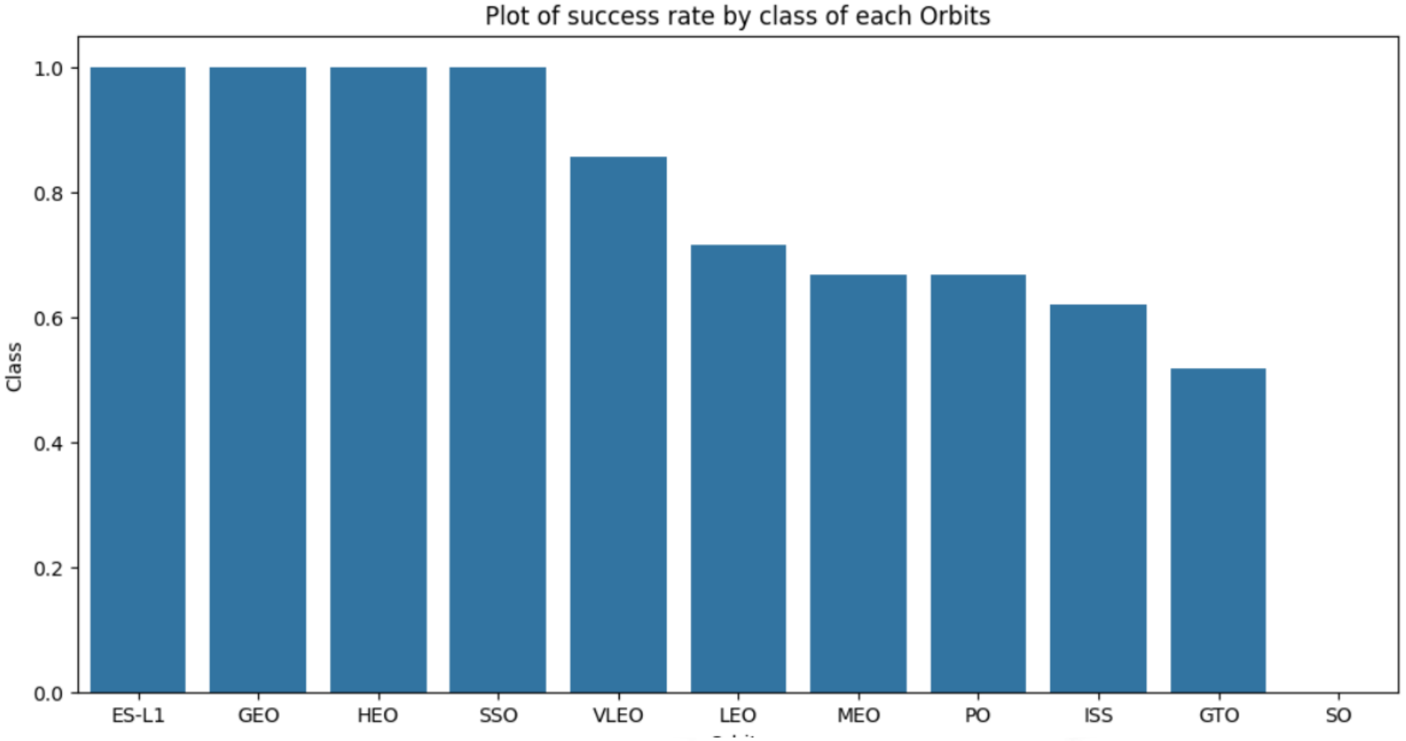
```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the class
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("Payload Mass",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



Success Rate vs. Orbit Type

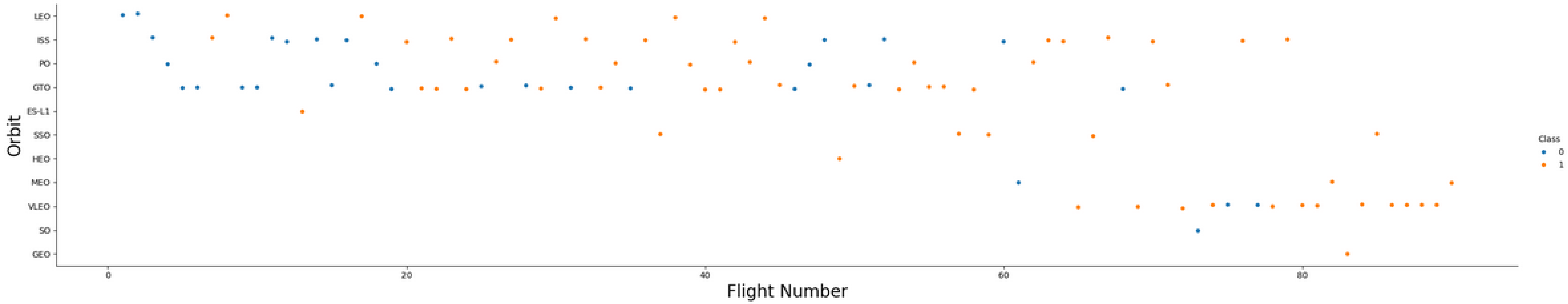
```
df_groupby_orbits = df.groupby('Orbit').Class.mean()
grouped_orbits = df.groupby(by=['Orbit'])['Class'].mean().sort_values(ascending=False).reset_index()
fig, ax=plt.subplots(figsize=(12,6))
ax = sns.barplot(x = 'Orbit', y = 'Class', data=grouped_orbits)
ax.set_title('Plot of success rate by class of each Orbits', fontdict={'size':12})
ax.set_ylabel('Class', fontsize = 10)
ax.set_xlabel('Orbits', fontsize = 10)
```

```
Text(0.5, 0, 'Orbits')
```



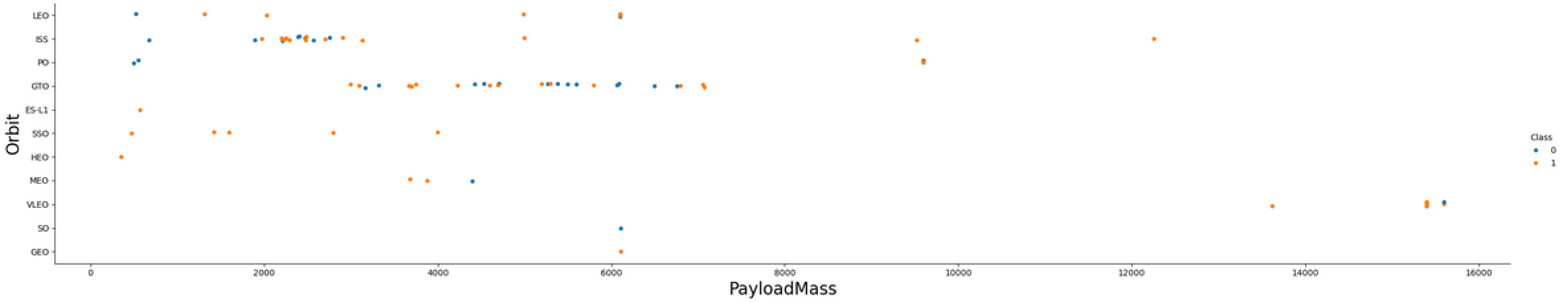
Orbit Type vs. Flight Number

```
# 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("Flight Number",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



Orbit Type vs. Payload mass

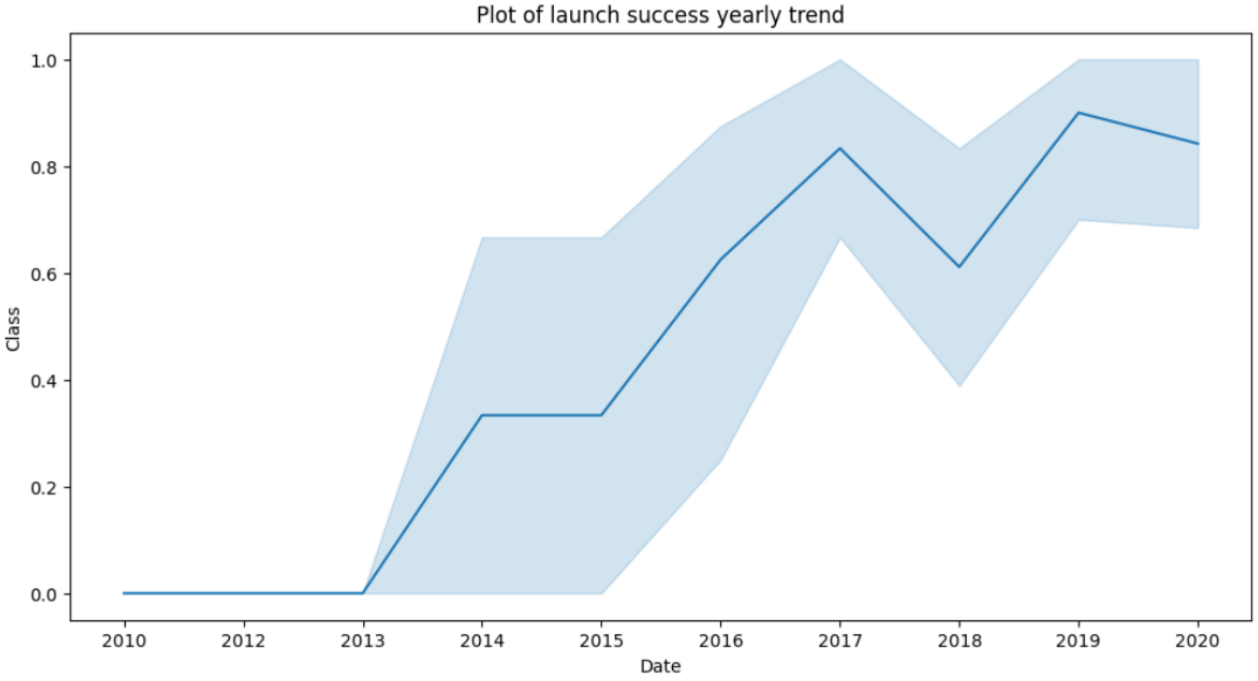
```
# Plot a scatter point chart with x axis to be Payload Mass and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("PayloadMass",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



Launch Success Yearly Trend

```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
#df_copy = df.copy()
#df_copy['Extracted_year'] = pd.DatetimeIndex(df['Date']).year

# plot line chart
fig, ax=plt.subplots(figsize=(12,6))
sns.lineplot(data=df, x='Date', y='Class')
plt.title('Plot of launch success yearly trend');
plt.show()
```



Launch Sites

```
task_1 = '''
        SELECT DISTINCT Launch_Site FROM SPACEXTABLE
'''
cur.execute(task_1)
for row in cur:
    print(f'row = {row}')
```

```
row = ('CCAFS LC-40',)
row = ('VAFB SLC-4E',)
row = ('KSC LC-39A',)
row = ('CCAFS SLC-40',)
```

Launch Sites

```
task_1 = '''
        SELECT DISTINCT Launch_Site FROM SPACEXTABLE
'''
cur.execute(task_1)
for row in cur:
    print(f'row = {row}')
```

```
row = ('CCAFS LC-40',)
row = ('VAFB SLC-4E',)
row = ('KSC LC-39A',)
row = ('CCAFS SLC-40',)
```

Launch Sites beginning with CCA

```
task_2 = '''
        SELECT *
        FROM SPACEXTABLE
        WHERE Launch_Site LIKE 'CCA%'
        LIMIT 5
        '''
cur.execute(task_2)
for row in cur:
    print(f'row = {row}')
```

```
row = ('2010-06-04', '18:45:00', 'F9 v1.0  B0003', 'CCAFS LC-40', 'Dragon Spacecraft Qualification Unit', 0, 'LEO', 'SpaceX',
'Success', 'Failure (parachute)')
row = ('2010-12-08', '15:43:00', 'F9 v1.0  B0004', 'CCAFS LC-40', 'Dragon demo flight C1, two CubeSats, barrel of Brouere che
ese', 0, 'LEO (ISS)', 'NASA (COTS) NRO', 'Success', 'Failure (parachute)')
row = ('2012-05-22', '7:44:00', 'F9 v1.0  B0005', 'CCAFS LC-40', 'Dragon demo flight C2', 525, 'LEO (ISS)', 'NASA (COTS)', 'S
uccess', 'No attempt')
row = ('2012-10-08', '0:35:00', 'F9 v1.0  B0006', 'CCAFS LC-40', 'SpaceX CRS-1', 500, 'LEO (ISS)', 'NASA (CRS)', 'Success', '
No attempt')
row = ('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 by NASA

```
]: task_3 = '''
            SELECT SUM(PAYLOAD_MASS__KG_)
            FROM SPACEXTABLE
            WHERE Customer LIKE 'NASA (CRS)'
            '''
   cur.execute(task_3)
   for row in cur:
       print(f'row = {row}')

row = (45596,)
```

Average Payload Mass with F9 v1.1

```
task_4 = '''
        SELECT AVG(PAYLOAD_MASS__KG_)
        FROM SPACEXTABLE
        WHERE Booster_Version = 'F9 v1.1'
        '''
cur.execute(task_4)
for row in cur:
    print(f'row = {row}')
```

```
row = (2928.4,)
```

First successful landing in ground pad

```
task_5 = '''
        SELECT MIN(Date)
        FROM SPACEXTABLE
        WHERE Landing_Outcome LIKE 'Success (ground pad)'
        '''
cur.execute(task_5)
for row in cur:
    print(f'row = {row}')
```

```
row = ('2015-12-22',)
```

Successful Drone Ship Landing with Payload between 4000 and 6000

```
task_6 = '''
        SELECT Booster_Version
        FROM SPACEXTABLE
        WHERE Landing_Outcome = 'Success (drone ship)'
            AND PAYLOAD_MASS__KG_ > 4000
            AND PAYLOAD_MASS__KG_ < 6000
        '''
cur.execute(task_6)
for row in cur:
    print(f'row = {row}')
```

```
row = ('F9 FT B1022',)
row = ('F9 FT B1026',)
row = ('F9 FT  B1021.2',)
row = ('F9 FT  B1031.2',)
```

Total Number of Successful and Failure Mission Outcomes

```
]:
task_7a = '''
        SELECT COUNT(Mission_Outcome)
        FROM SPACEXTABLE
        WHERE Mission_Outcome LIKE 'Success%'
        '''
cur.execute(task_7a)
for row in cur:
    print(f'success = {row}')
task_7b = '''
        SELECT COUNT(Mission_Outcome)
        FROM SPACEXTABLE
        WHERE Mission_Outcome LIKE 'Failure%'
        '''
cur.execute(task_7b)
for row in cur:
    print(f'failure = {row}')
```

```
success = (100,)
failure = (1,)
```

Boosters Carried Maximum Payload

```
]:  task_8 = '''
            SELECT Booster_Version, PAYLOAD_MASS__KG_
            FROM SPACEXTABLE
            WHERE PAYLOAD_MASS__KG_ = (
                                        SELECT MAX(PAYLOAD_MASS__KG_)
                                        FROM SPACEXTABLE
                                        )
            ORDER BY Booster_Version
            '''
    cur.execute(task_8)
    for row in cur:
        print(f'row = {row}')
```

```
row = ('F9 B5 B1048.4', 15600)
row = ('F9 B5 B1048.5', 15600)
row = ('F9 B5 B1049.4', 15600)
row = ('F9 B5 B1049.5', 15600)
row = ('F9 B5 B1049.7 ', 15600)
row = ('F9 B5 B1051.3', 15600)
row = ('F9 B5 B1051.4', 15600)
row = ('F9 B5 B1051.6', 15600)
row = ('F9 B5 B1056.4', 15600)
row = ('F9 B5 B1058.3 ', 15600)
row = ('F9 B5 B1060.2 ', 15600)
row = ('F9 B5 B1060.3', 15600)
```

2015 Failure Launch Records

```
task_9 = '''
        SELECT Booster_Version, Launch_Site, Landing_Outcome, Date
        FROM SPACEXTABLE
        WHERE Landing_Outcome LIKE 'Failure (drone ship)'
            AND Date BETWEEN '2015-01-01' AND '2015-12-31'
        '''
cur.execute(task_9)
for row in cur:
    print(f'row = {row}')
```

```
row = ('F9 v1.1 B1012', 'CCAFS LC-40', 'Failure (drone ship)', '2015-01-10')
row = ('F9 v1.1 B1015', 'CCAFS LC-40', 'Failure (drone ship)', '2015-04-14')
```

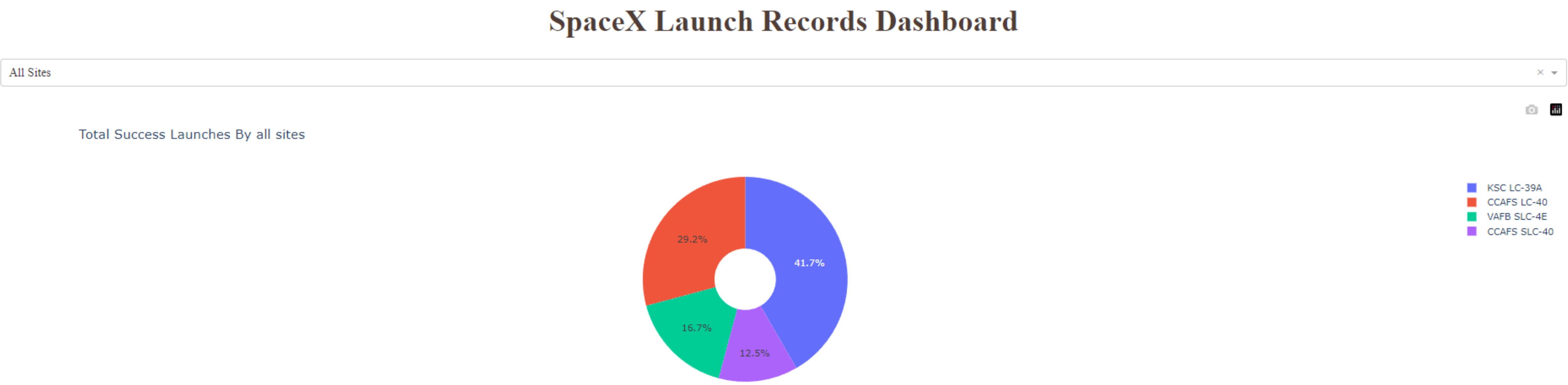
Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

```
task_10 = '''
        SELECT Landing_Outcome, COUNT(Landing_Outcome)
        FROM SPACEXTABLE
        WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
        GROUP BY LandingOutcome
        ORDER BY COUNT(LandingOutcome) DESC
        '''
cur.execute(task_10)
for row in cur:
    print(f'row = {row}')
```

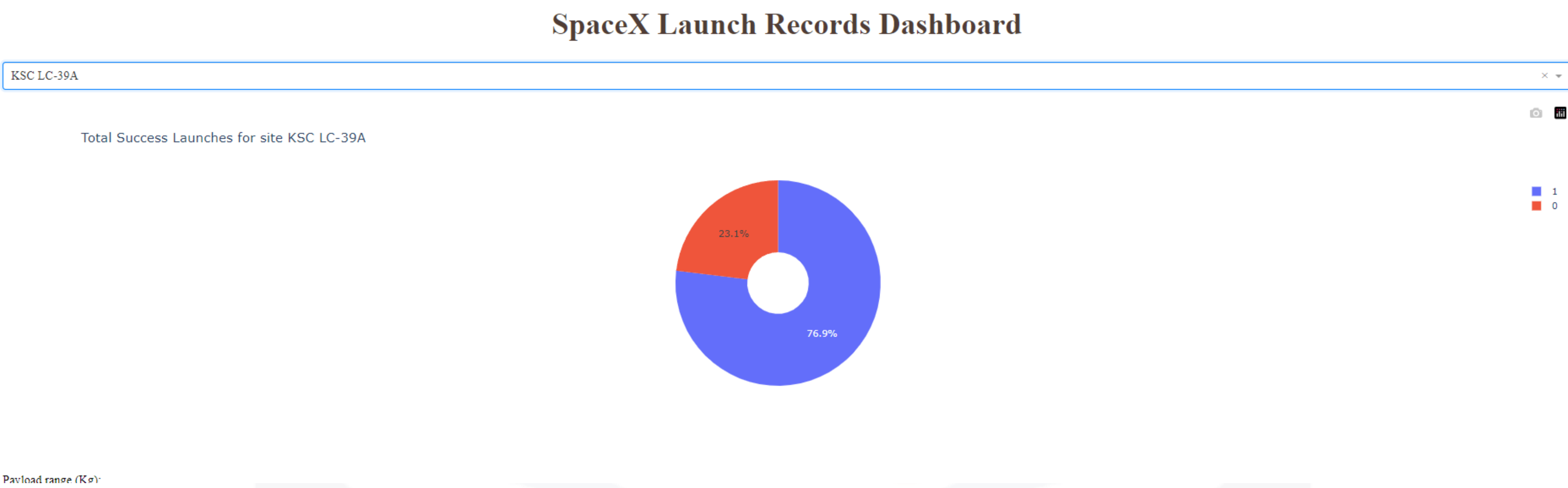
	landingoutcome	count
0	No attempt	10
1	Success (drone ship)	6
2	Failure (drone ship)	5
3	Success (ground pad)	5
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Precluded (drone ship)	1
7	Failure (parachute)	1

Launch Site Analysis

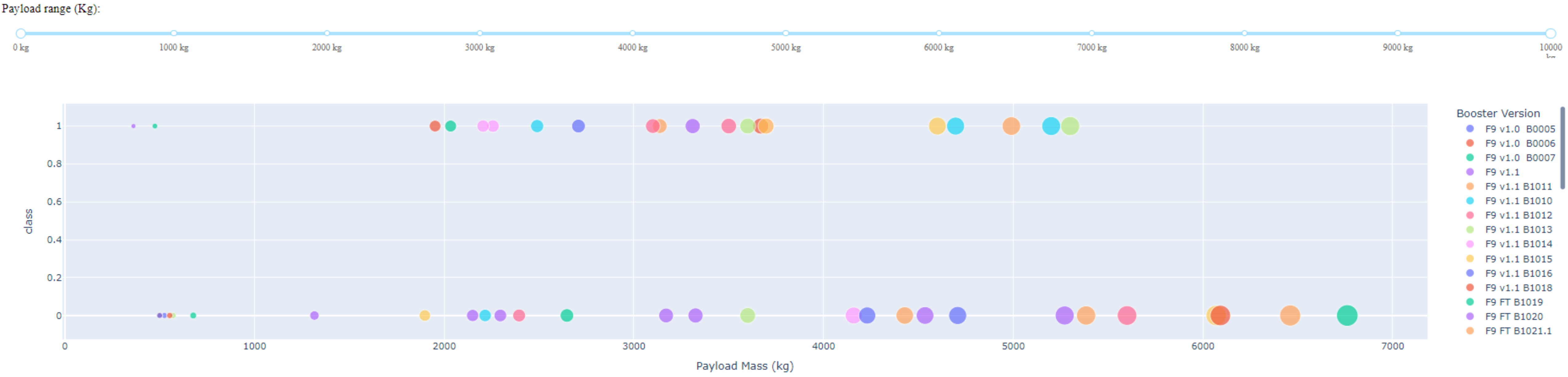
Success counts for all launch sites



Launch site with highest launch success ratio

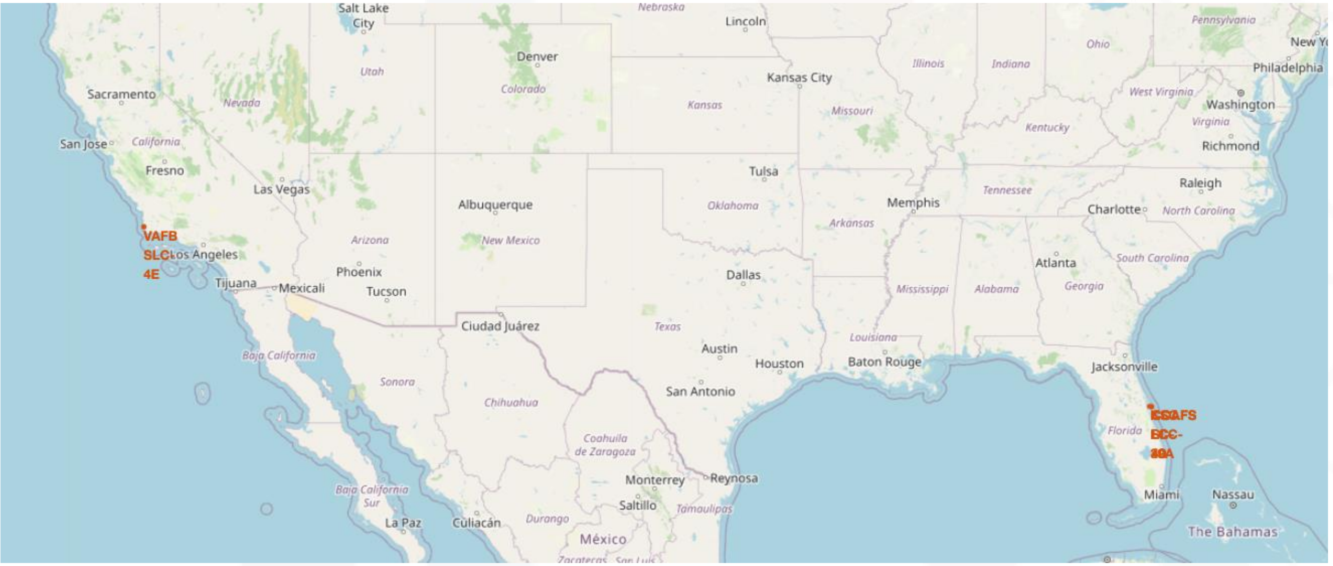


Launch Outcome for all sites and all Payloads



Launch Outcome for all sites and Payloads between 2000 and 7000

Map with Launch Sites



Distance to coast

Success/Failed launches for selected site

Predictive Analysis

Classification Accuracy

```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

```
Best model is DecisionTree with a score of 0.8732142857142856
```