IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Sai Bhargav
2023-02-15

Outline

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

Executive Summary

Summary of methodologies

- Data Collection API
- WebScraping
- Python Data Wrangling
- Exploratory Data Analysis using SQL
- Exploratory Data analysis for Data visualization
- Interactive Visual Analytics and Dashboards
- Machine Learning Prediction

Introduction

- Project background and context
- SpaceX 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 SpaceX can reuse the first stage. 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.
- Problems you want to find answers
- If the Falcon 9 first stage will land successfully.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - SpaceX Rest API
  - Web Scrapping from Wikipedia
- Perform datawrangling
  - Data cleaning using one hot encoding
- Perform exploratory data analysis (EDA) using visualization andSQL
- Perform interactive visual analytics using Folium andPlotlyDash
- Perform predictive analysis using classification models
  - Logistic Regression, KNN, SVM, Decision Tree models are built and evaluated based on confusion matrix and accuary

Data Collection

- Describe how data sets were collected.
  - **Request and parse the SpaceX launch data using the GET request**
  - **Filter the dataframeto only include Falcon 9 launches**
  - **Dealing with Missing Values**

Data Collection –SpaceX API

- Add the GitHub URL of the completed SpaceX API calls notebook (must include completed code cell and outcome cell), as an external reference and peer-review purpose

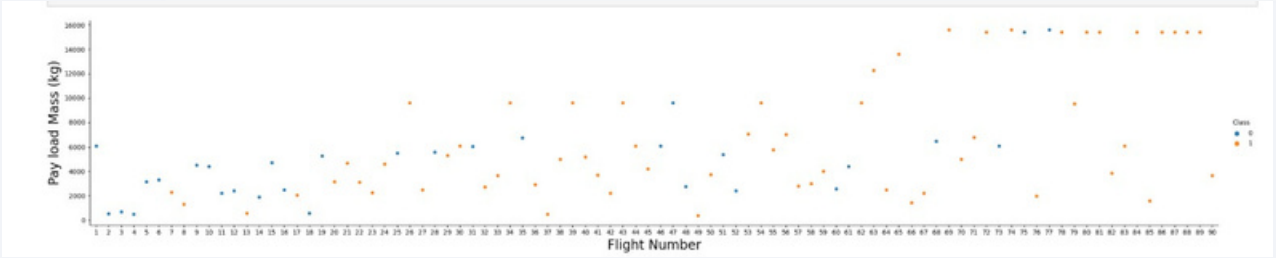
Data Collection -Scraping

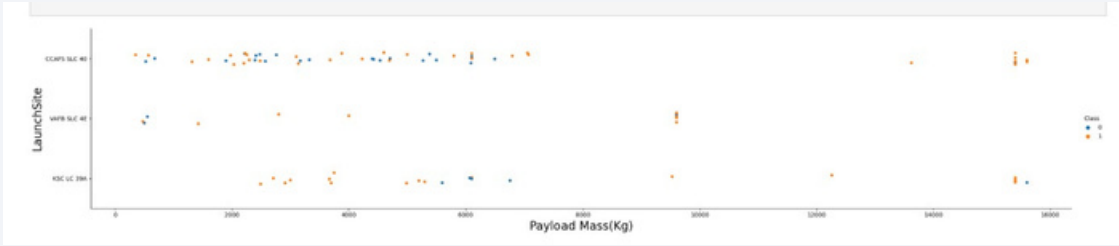
- Add the GitHub URL of the completed web scraping notebook, as an external reference and peer-review purpose

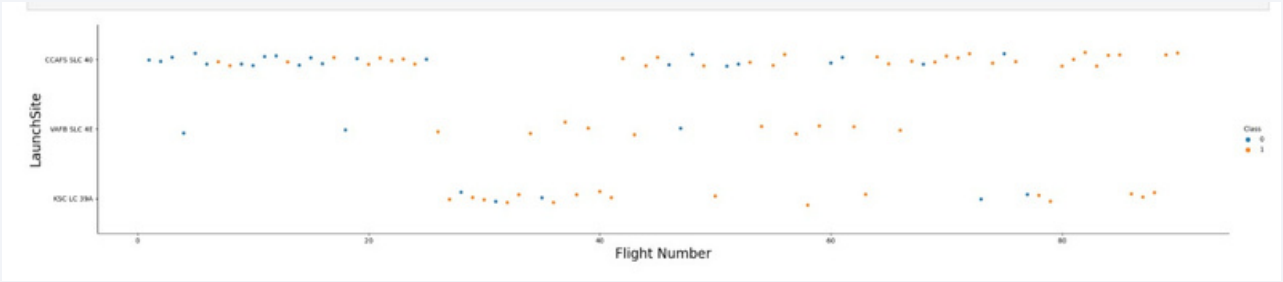
Data Wrangling

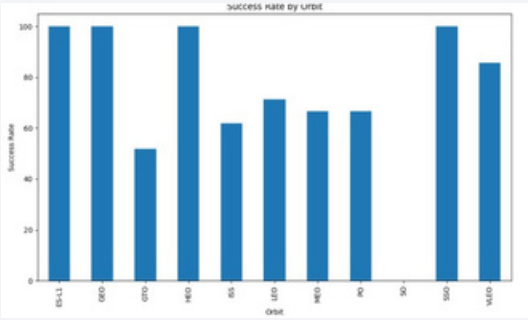
- Add the GitHub URL of your completed data wrangling related notebooks, as an external reference and peer-review purpose

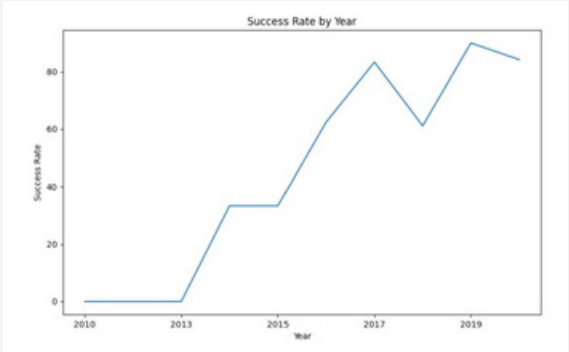
EDA with Data Visualization











- Add the GitHub URL of your completed EDA with data visualization notebook, as an external reference and peer-review purpose

EDA withSQL

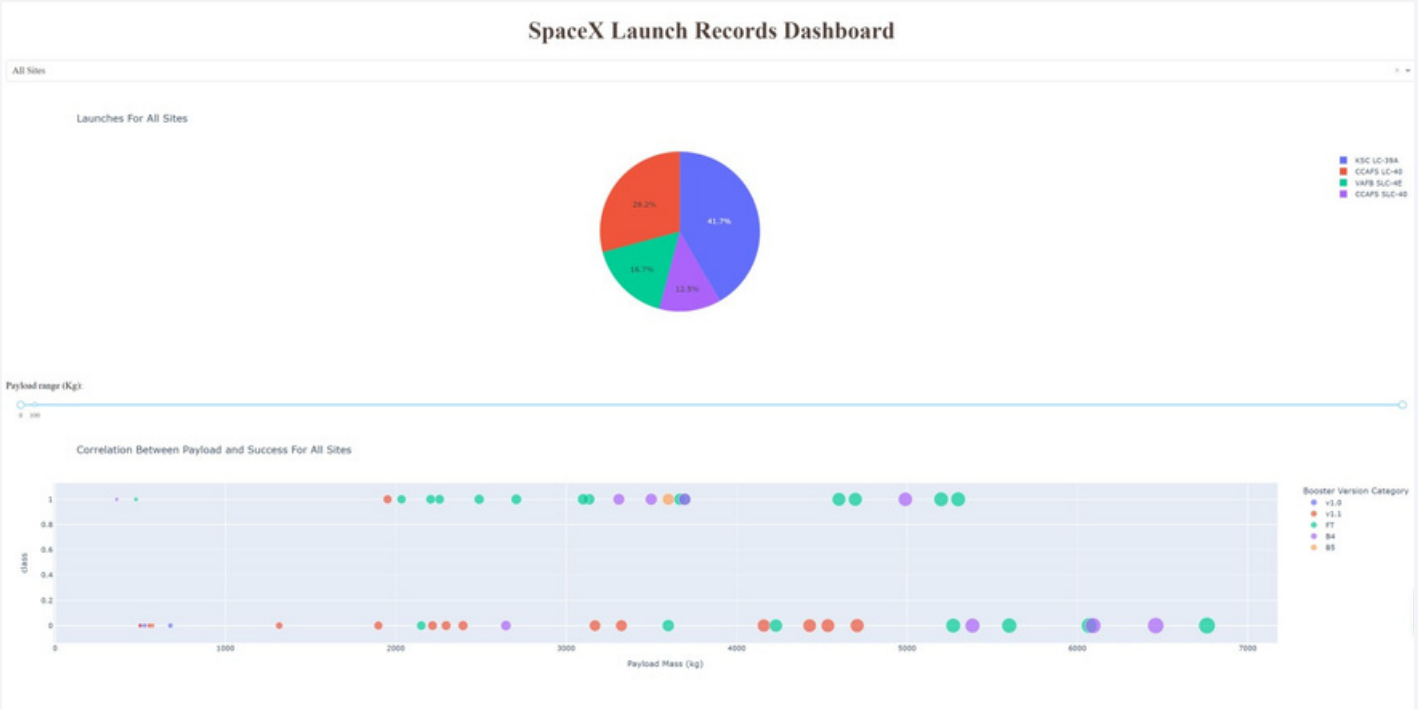
- Add the GitHub URL of your completed EDA with SQL notebook, as an external reference and peer-review purpose

Build an Interactive Map with Folium

- Add the GitHub URL of your completed interactive map with Folium map, as an external reference and peer-review purpose

Build a Dashboard with PlotlyDash

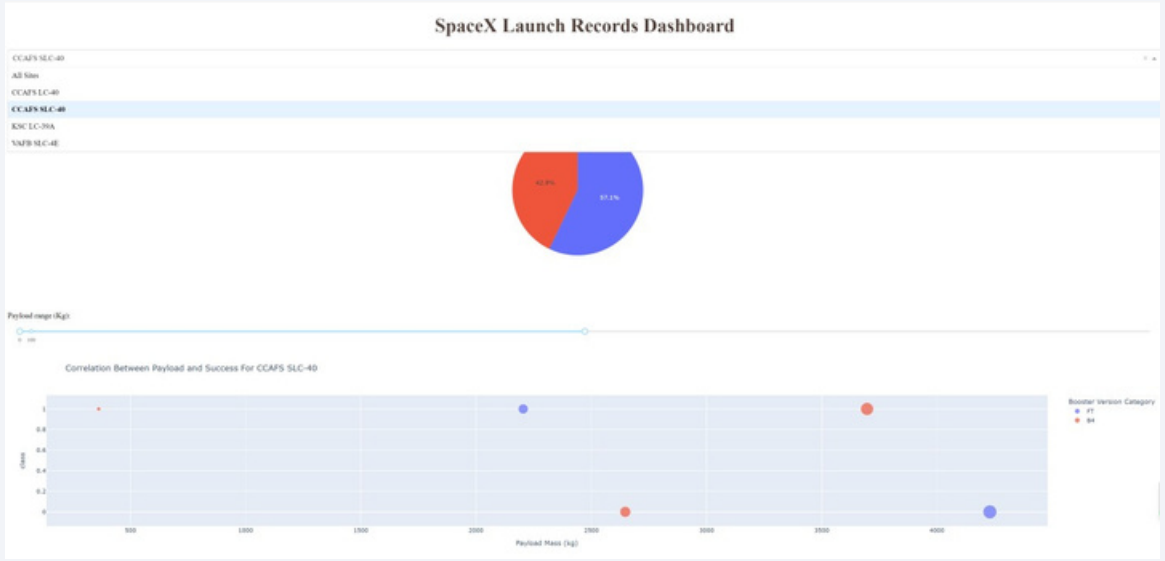
- Add the GitHub URL of your completed PlotlyDash lab, as an external reference and peer-review purpose

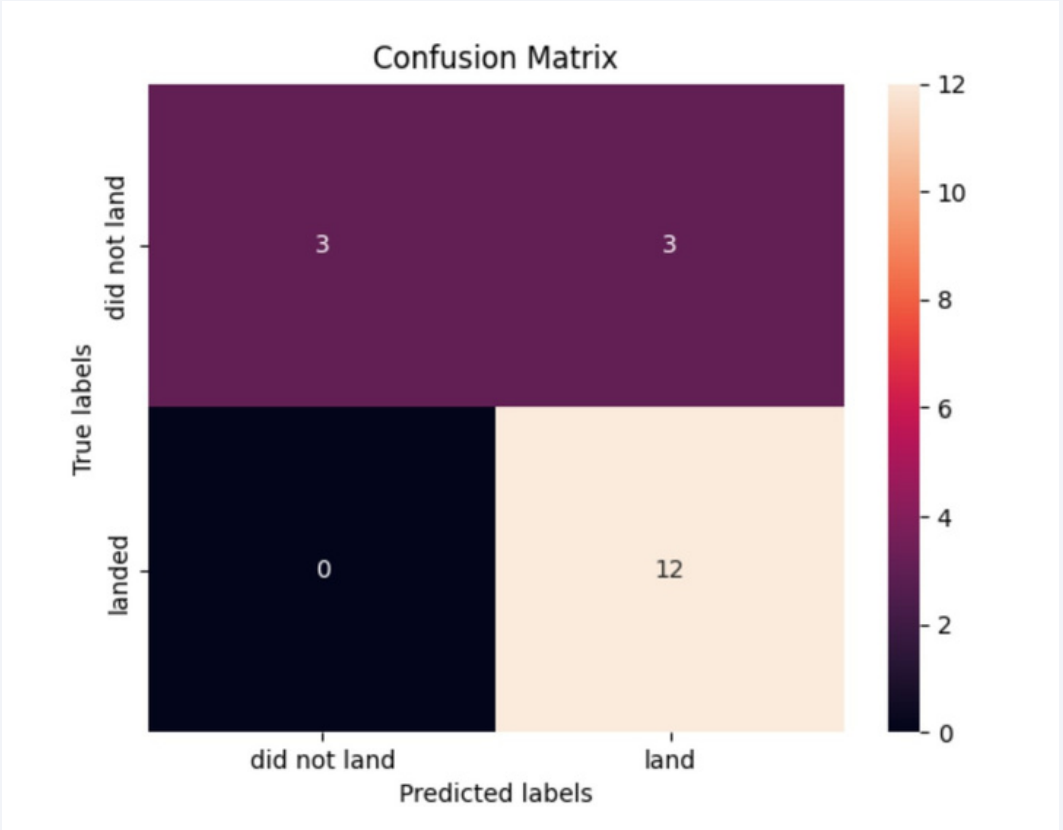


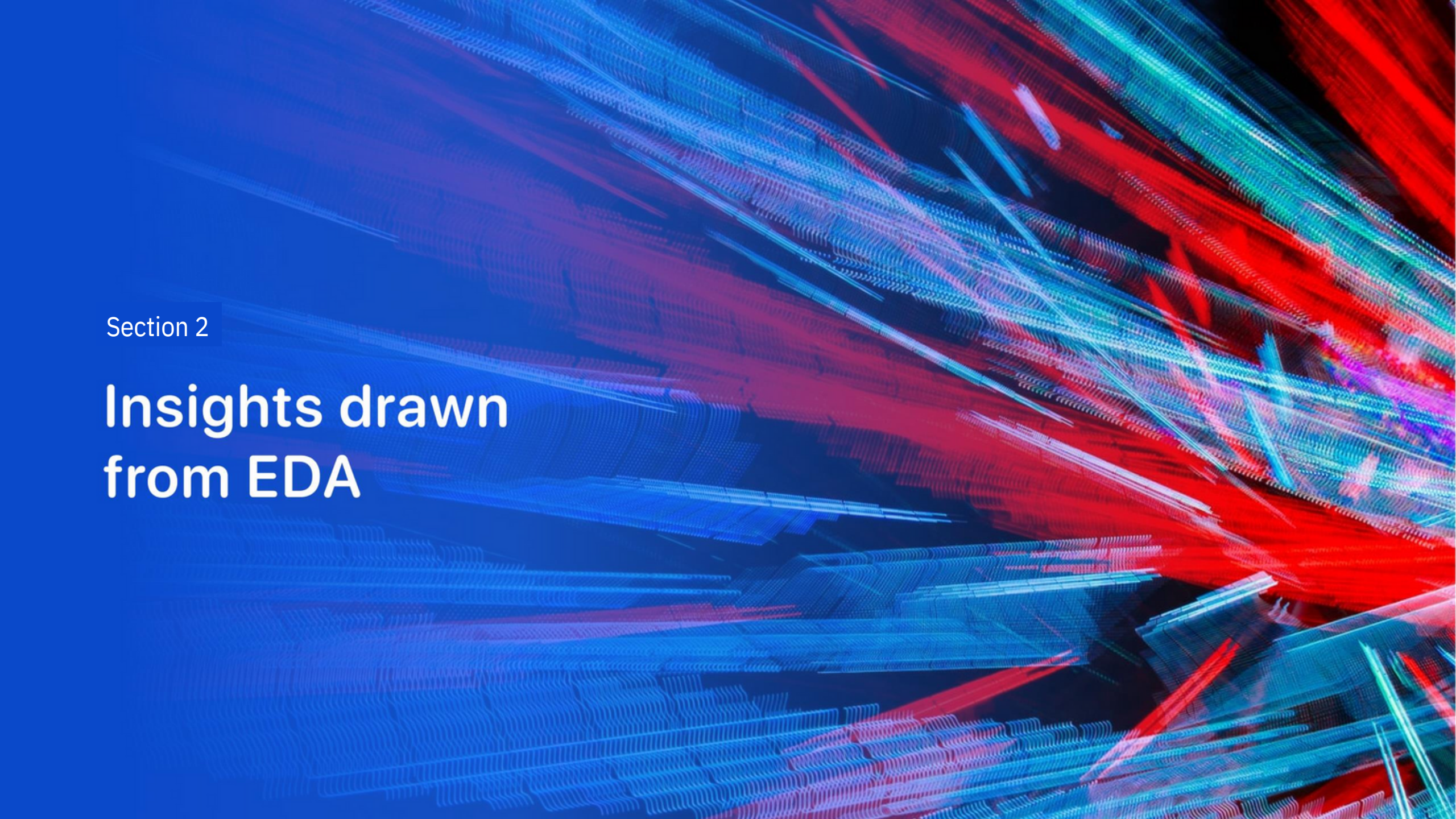
Predictive Analysis (Classification)

- Add the GitHub URL of your completed predictive analysis lab, as an external reference and peer-review purpose

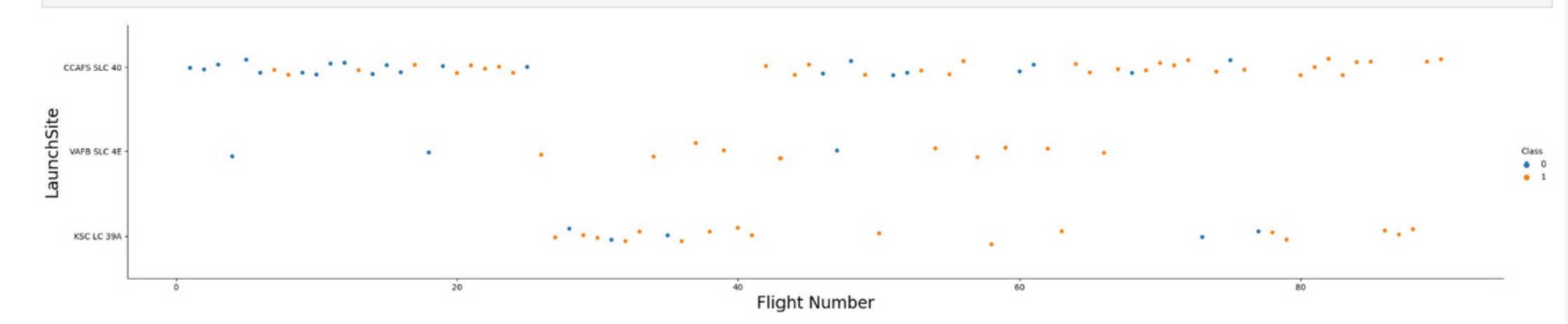
Results


SpaceX Launch Records Dashboard
CCAFS SLC-40
All Sites
CCAFS LC-40
CCAFS SLC-40
KSC LC-39A
VAFB SLC-4E
Payload range (Kg):
Correlation Between Payload and Success For CCAFS SLC-40
Booster Version Category
Payload Mass (kg)

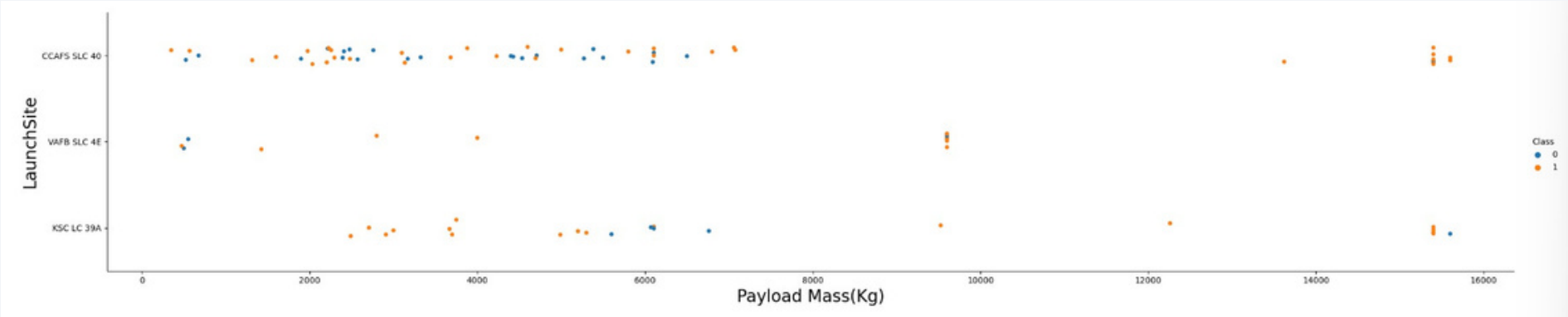

Confusion Matrix
True labels
did not land
landed
3
3
0
12
did not land
land
Predicted labels
12
10
8
6
4
2
0


Section 2
Insights drawn from EDA

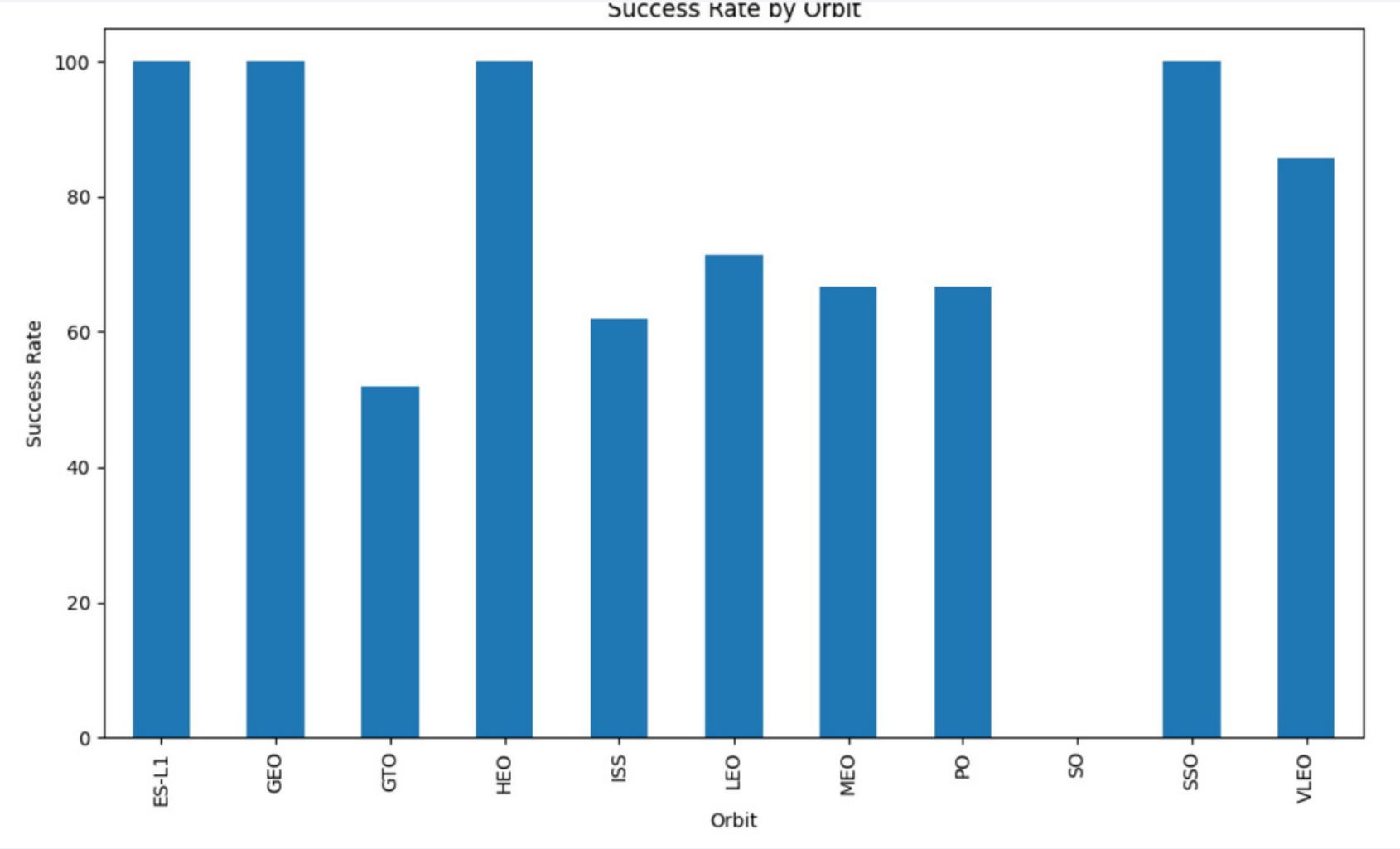
Flight Number vs. LaunchSite



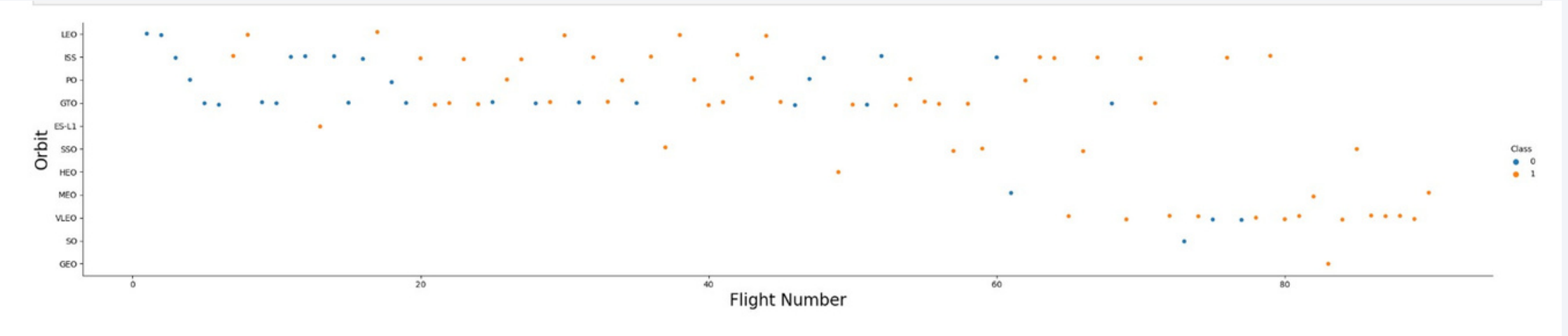
Payload vs. Launch Site



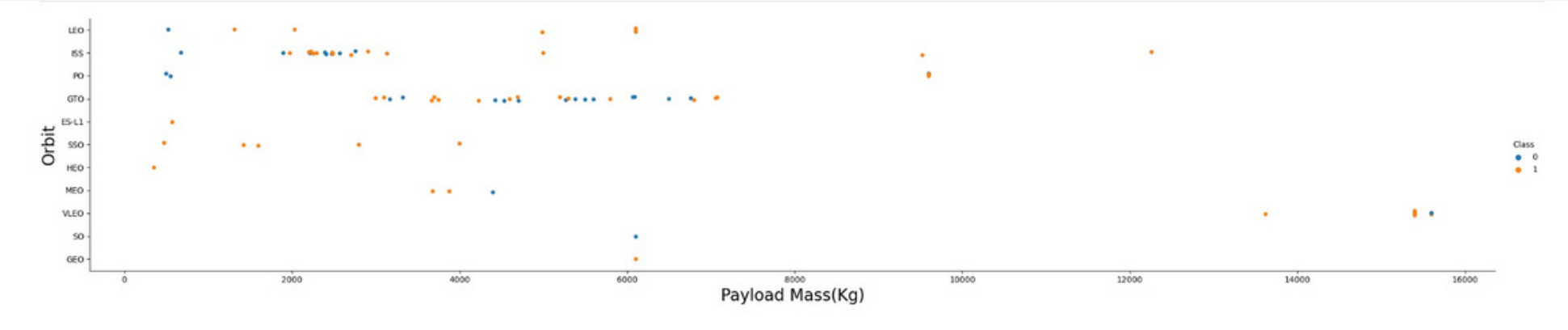
Success Rate vs. Orbit Type



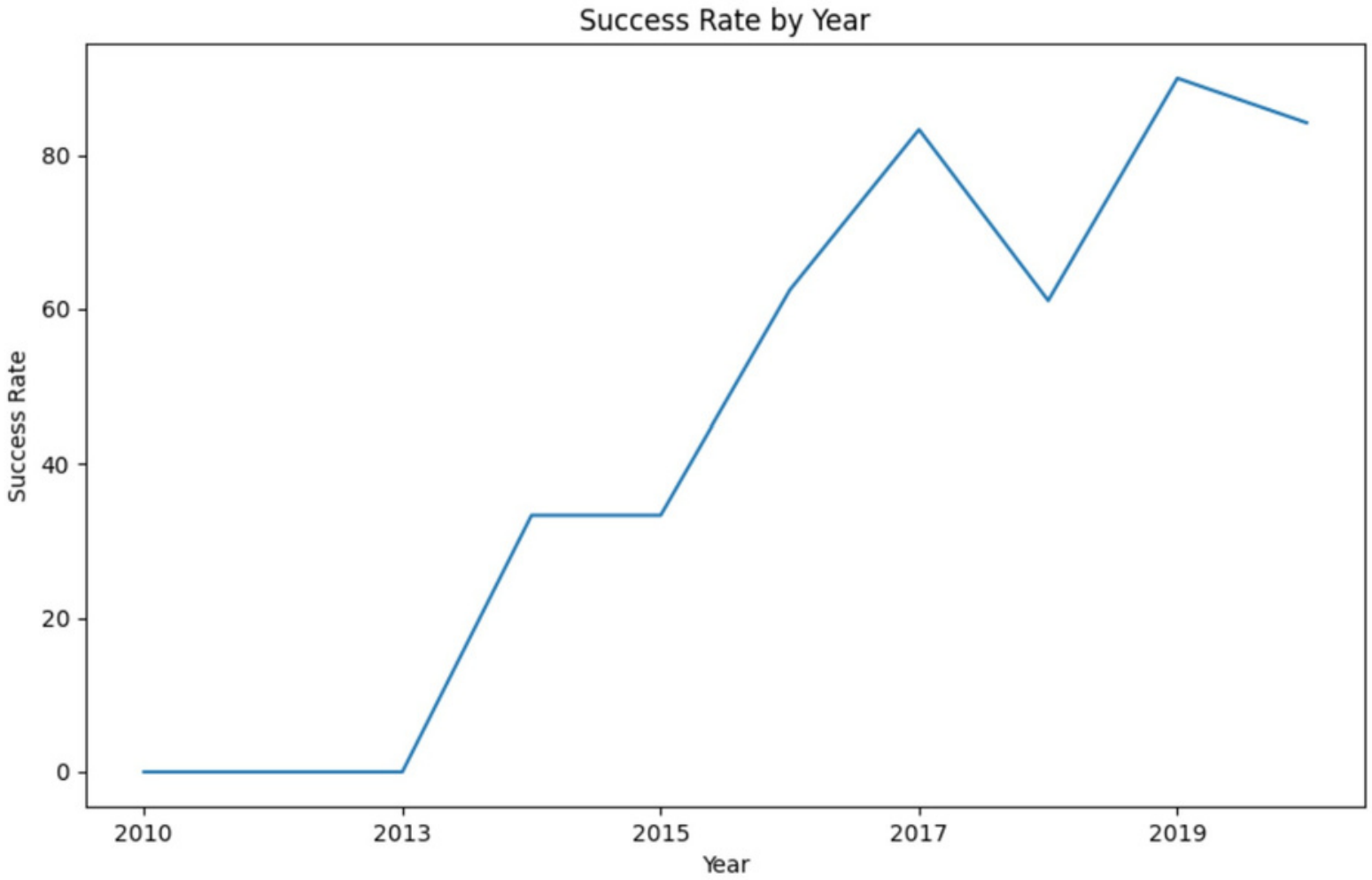
Flight Number vs. Orbit Type



Payload vs. OrbitType



Launch Success Yearly Trend



All Launch Site Names

- Find the names of the unique launch sites
- Present your query result with a short explanation here

```
cur.execute('''select distinct Launch_Site from SPACEXTBL;''')
print (cur.fetchall())
```

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

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`
- Present your query result with a short explanation here

```
cur.execute('''select * from SPACEXTBL where substr(Launch_Site,1,3) = 'CCA' Limit 5;''')
print (cur.fetchall())
```

```
[('04-06-2010', '18:45:00', 'F9 v1.0  B0003', 'CCAFS LC-40', 'Dragon Spacecraft Qualification Unit', 0, 'LEO', 'SpaceX', 'Succes
s', 'Failure (parachute)'), ('08-12-2010', '15:43:00', 'F9 v1.0  B0004', 'CCAFS LC-40', 'Dragon demo flight C1, two CubeSats, bar
rel 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', '1
5:10:00', 'F9 v1.0  B0007', 'CCAFS LC-40', 'SpaceX CRS-2', 677, 'LEO (ISS)', 'NASA (CRS)', 'Success', 'No attempt')]
```

Total Payload Mass

- Calculate the total payload carried by boosters from NASA
- Present your query result with a short explanation here

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

```
cur.execute('''select sum(PAYLOAD_MASS__KG_) from SPACEXTBL where Customer = 'NASA (CRS)';''')
print (cur.fetchall())
```

```
[(45596,)]
```

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1
- Present your query result with a short explanation here

```
cur.execute('''select avg(PAYLOAD_MASS__KG_) from SPACEXTBL where Booster_Version = 'F9 v1.1';''')
print (cur.fetchall())
```

```
[(2928.4,)]
```

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad
- Present your query result with a short explanation here

```
cur.execute('''select min(Date) from SPACEXTBL where [Landing _Outcome] = 'Success (ground pad)';''')
print (cur.fetchall())
```

```
[('01-05-2017',)]
```

Successful Drone Ship Landing with Payload between 4000 and 6000

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

- Present your query result with a short explanation here

```
cur.execute('''select distinct Booster_Version from SPACEXTBL where PAYLOAD_MASS__KG_ > 4000 and PAYLOAD_MASS__KG_ < 6000;''')
print (cur.fetchall())
```

```
[('F9 v1.1',), ('F9 v1.1 B1011',), ('F9 v1.1 B1014',), ('F9 v1.1 B1016',), ('F9 FT B1020',), ('F9 FT B1022',), ('F9 FT B1026',),
('F9 FT B1030',), ('F9 FT  B1021.2',), ('F9 FT B1032.1',), ('F9 B4 B1040.1',), ('F9 FT  B1031.2',), ('F9 B4 B1043.1',), ('F9 FT
B1032.2',), ('F9 B4  B1040.2',), ('F9 B5 B1046.2',), ('F9 B5 B1047.2',), ('F9 B5B1054',), ('F9 B5 B1048.3',), ('F9 B5 B1051.2
',), ('F9 B5B1060.1',), ('F9 B5 B1058.2 ',), ('F9 B5B1062.1',)]
```

Total Number of Successful and FailureMission Outcomes

- Calculate the total number of successful and failure mission outcomes
- Present your query result with a short explanation here

```
cur.execute('''select Mission_Outcome, count(*) from SPACEXTBL Group by Mission_Outcome;''')
print (cur.fetchall())
```

```
[('Failure (in flight)', 1), ('Success', 98), ('Success ', 1), ('Success (payload status unclear)', 1)]
```

Boosters Carried MaximumPayload

- List the names of the booster which have carried the maximum payload mass
- Present your query result with a short explanation here

```
cur.execute('''select distinct Booster_Version from SPACEXTBL where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACE
print (cur.fetchall())
```

```
[('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 failedlanding_outcomesin drone ship, their booster versions, andlaunch site names for in year 2015

- Present your query result with a short explanation here

```
cur.execute('''select substr(Date,4,2) as month,[Landing _Outcome], Booster_Version, Launch_Site from SPACEXTBL where substr(Date
print (cur.fetchall())
```

```
[('01', 'Failure (drone ship)', 'F9 v1.1 B1012', 'CCAFS LC-40'), ('04', '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

- Present your query result with a short explanation here

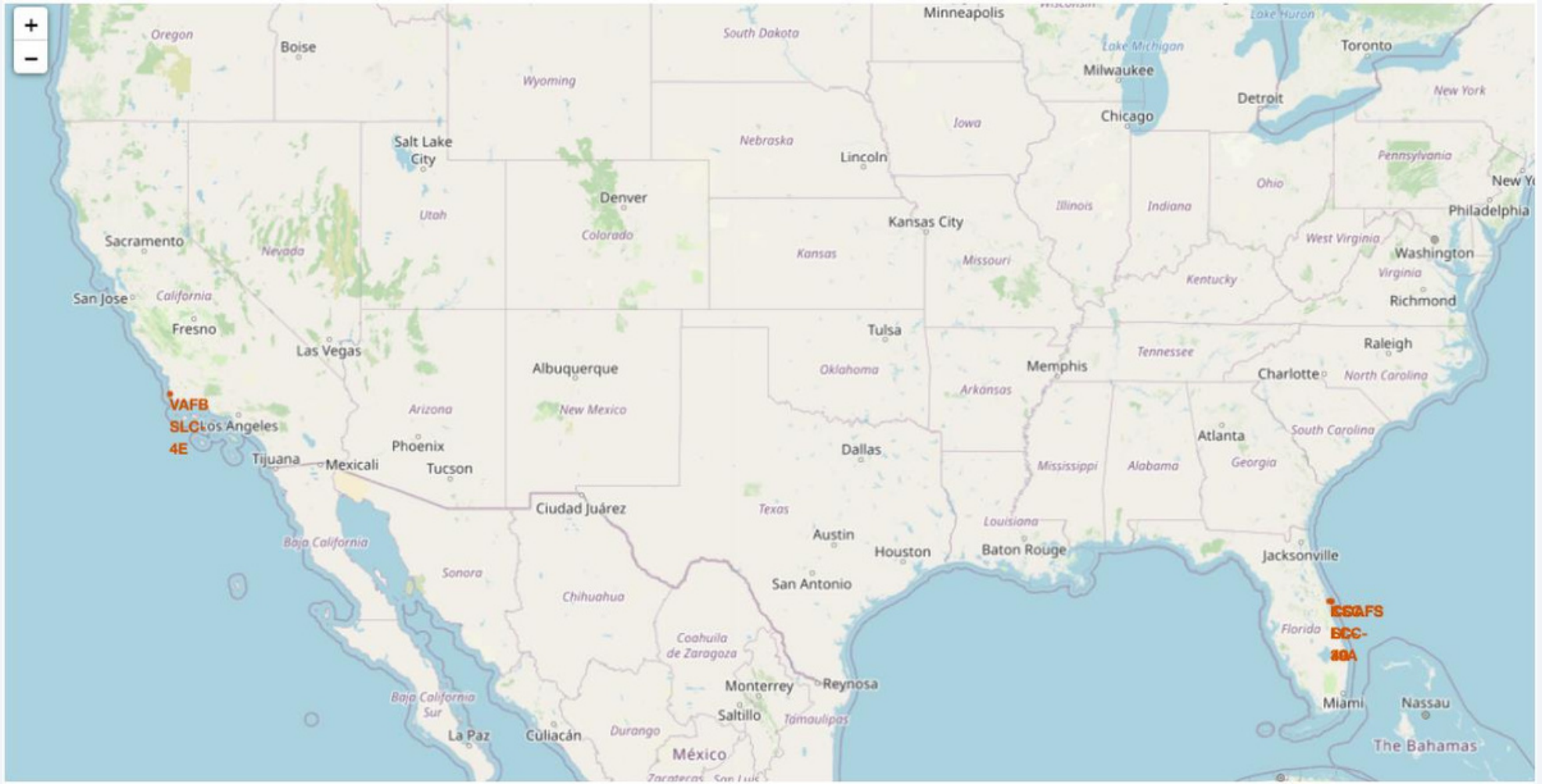
```
cur.execute('''select date, count(*) as count from SPACEXTBL where date between '04-06-2020' and '20-03-2017' Group by date order
print (cur.fetchall())
```

```
[('19-02-2017', 1), ('19-01-2020', 1), ('18-10-2020', 1), ('18-08-2020', 1), ('18-07-2016', 1), ('18-04-2018', 1), ('18-04-2014',
1), ('18-03-2020', 1), ('17-12-2019', 1), ('17-02-2020', 1), ('17-01-2016', 1), ('16-11-2020', 1), ('16-03-2017', 1), ('15-12-201
7', 1), ('15-11-2018', 1), ('15-06-2016', 1), ('15-05-2017', 1), ('14-08-2017', 1), ('14-08-2016', 1), ('14-07-2014', 1), ('14-04
-2015', 1), ('14-01-2017', 1), ('13-06-2020', 1), ('12-06-2019', 1), ('11-11-2019', 1), ('11-10-2017', 1), ('11-05-2018', 1), ('1
1-02-2015', 1), ('11-01-2019', 1), ('10-09-2018', 1), ('10-01-2015', 1), ('09-10-2017', 1), ('08-12-2010', 1), ('08-10-2018', 1),
('08-10-2012', 1), ('08-04-2016', 1), ('08-01-2018', 1), ('07-09-2017', 1), ('07-09-2014', 1), ('07-08-2020', 1), ('07-08-2018',
1), ('07-03-2020', 1), ('07-01-2020', 1), ('06-12-2020', 1), ('06-10-2020', 1), ('06-08-2019', 1), ('06-05-2016', 1), ('06-03-201
8', 1), ('06-01-2014', 1), ('05-12-2019', 1), ('05-12-2018', 1), ('05-11-2020', 1), ('05-08-2014', 1), ('05-07-2017', 1), ('04-06
-2020', 1)]
```

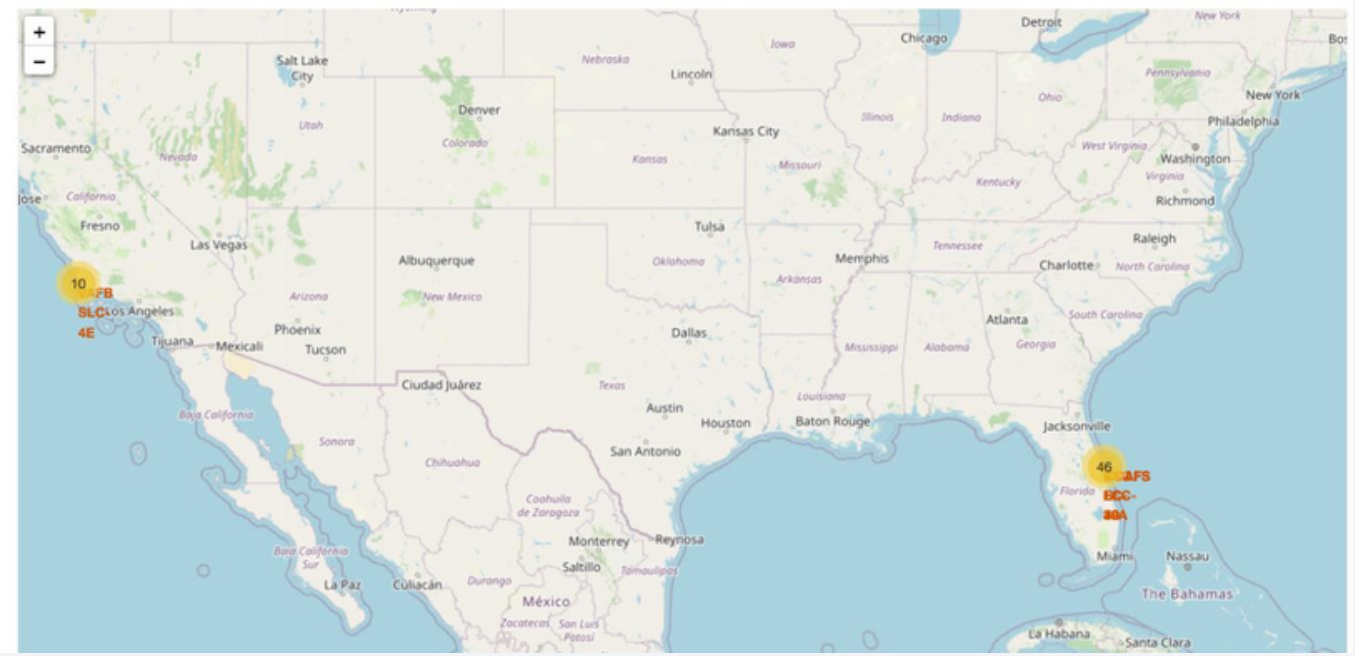
Section 3

Launch Sites Proximities Analysis

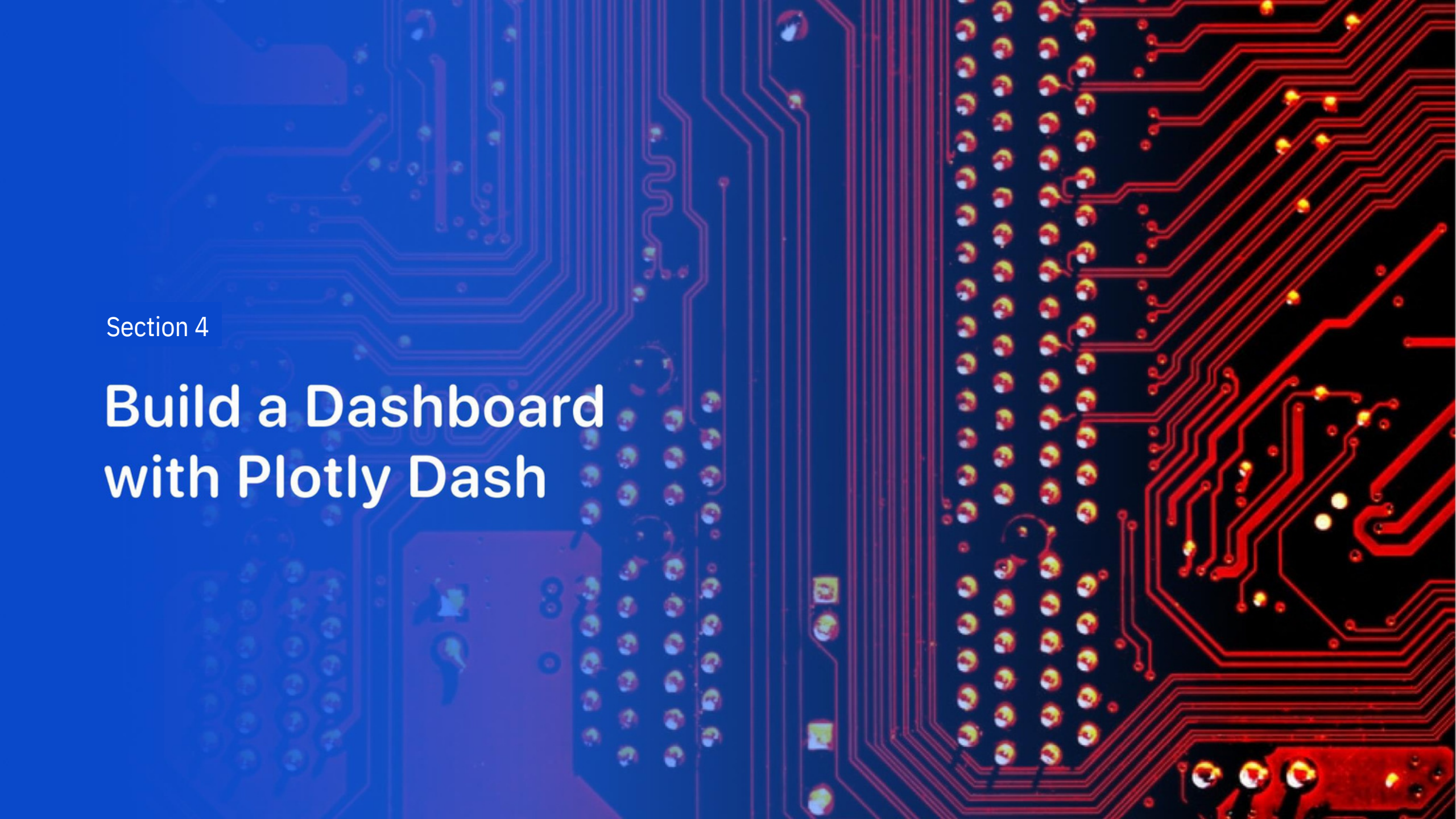
All Launch Sites



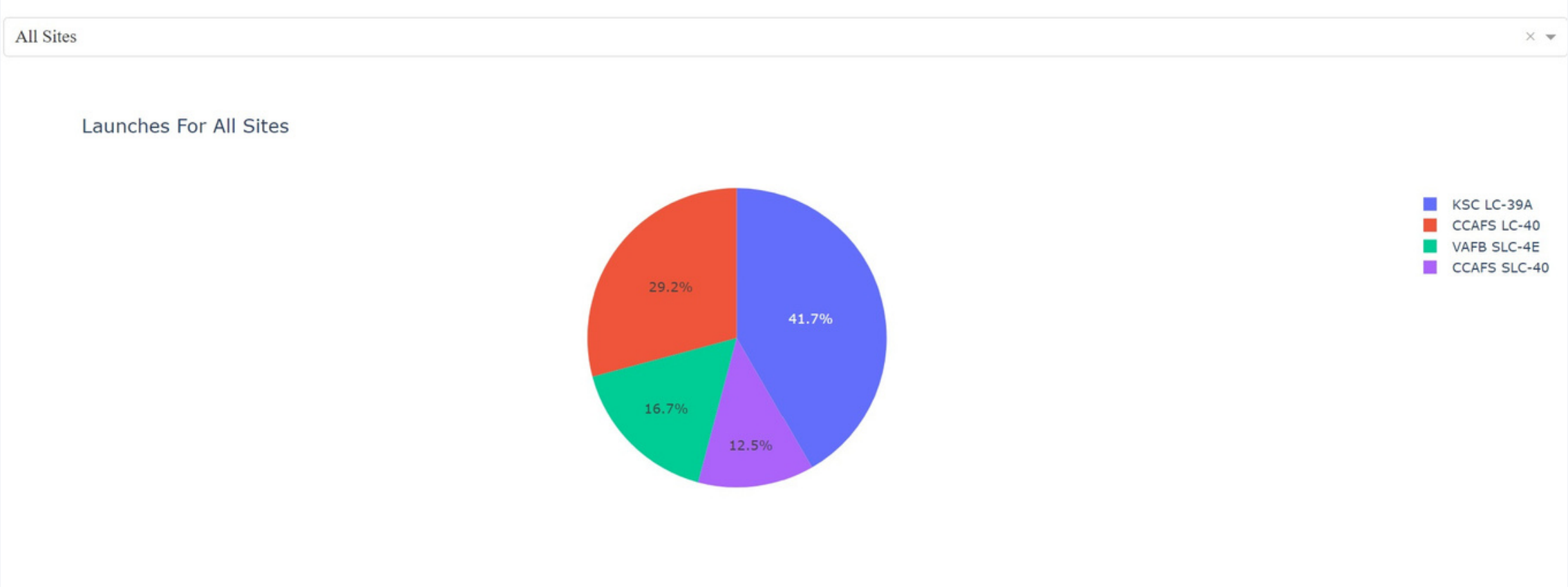
Launches by Launch Sites



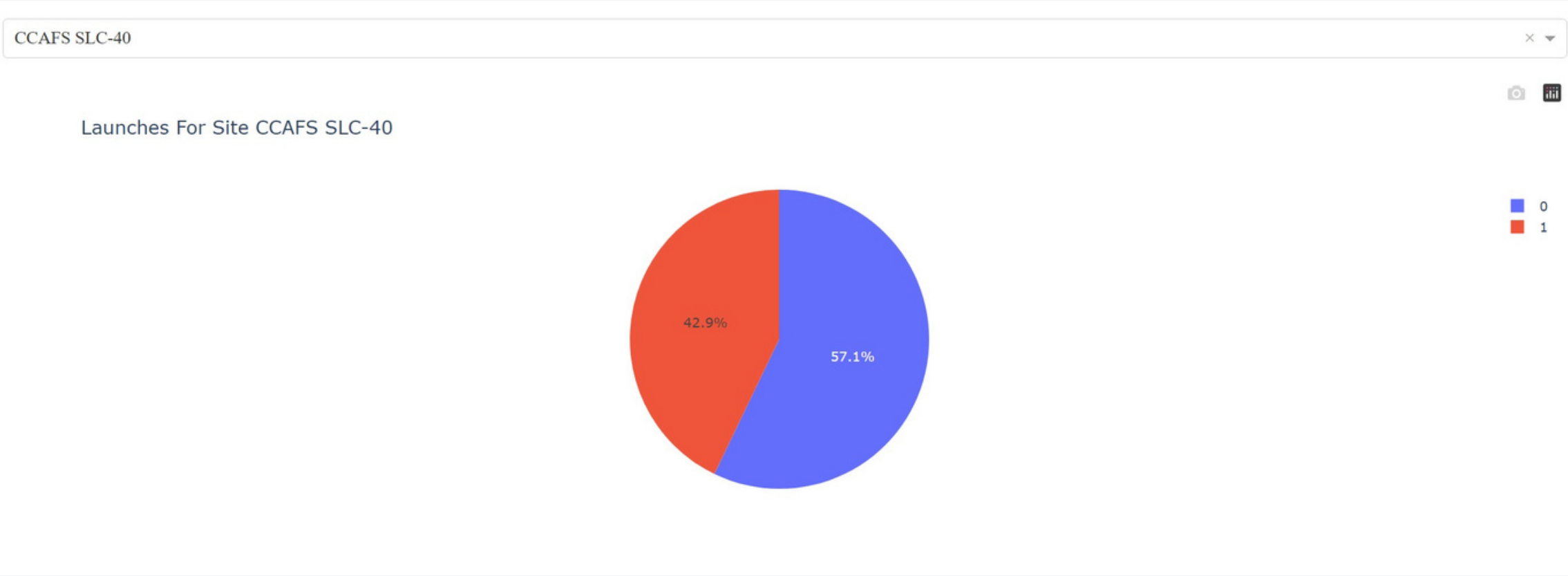
Site Proximity to Coast, Railway, Highway and City


Section 4
Build a Dashboard with Plotly Dash

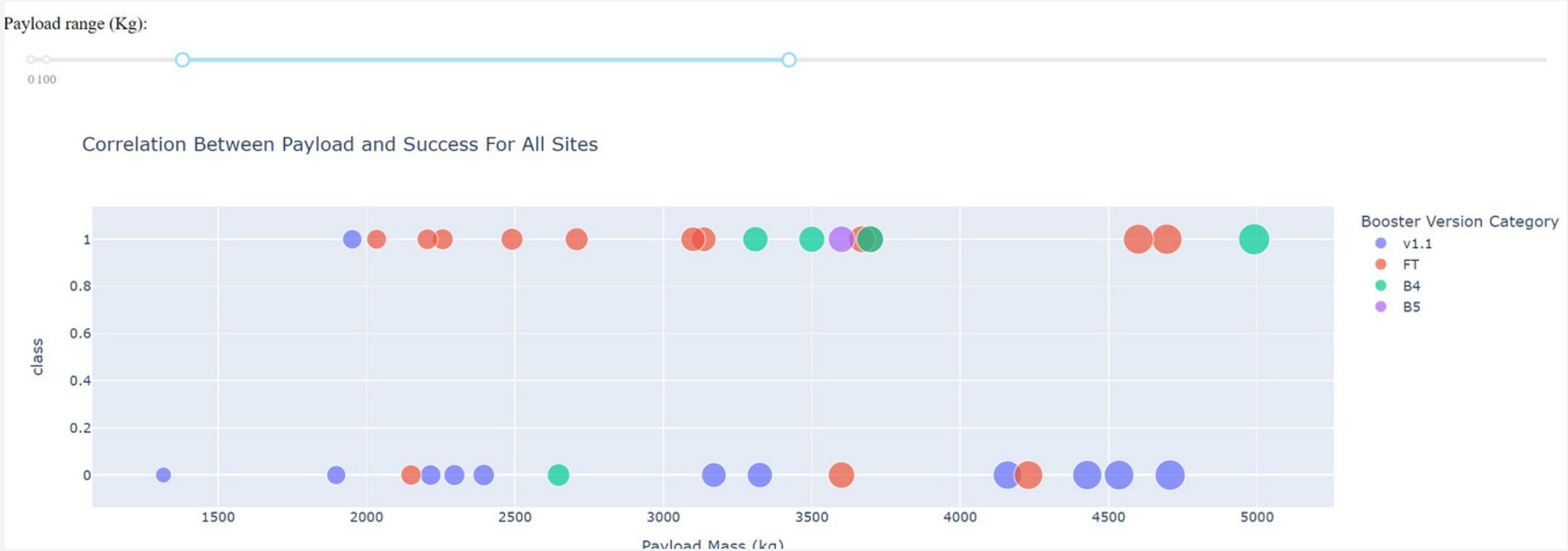
Pie Chart on Success Launches by Launch Sites



Site with Highest Launch Success Ratio



Payload Range and Success For All Sites



Section 5

Predictive Analysis (Classification)

Classification Accuracy

TASK 8

Create a decision tree classifier object then create a `GridSearchCV` object `tree_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

```
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()
```

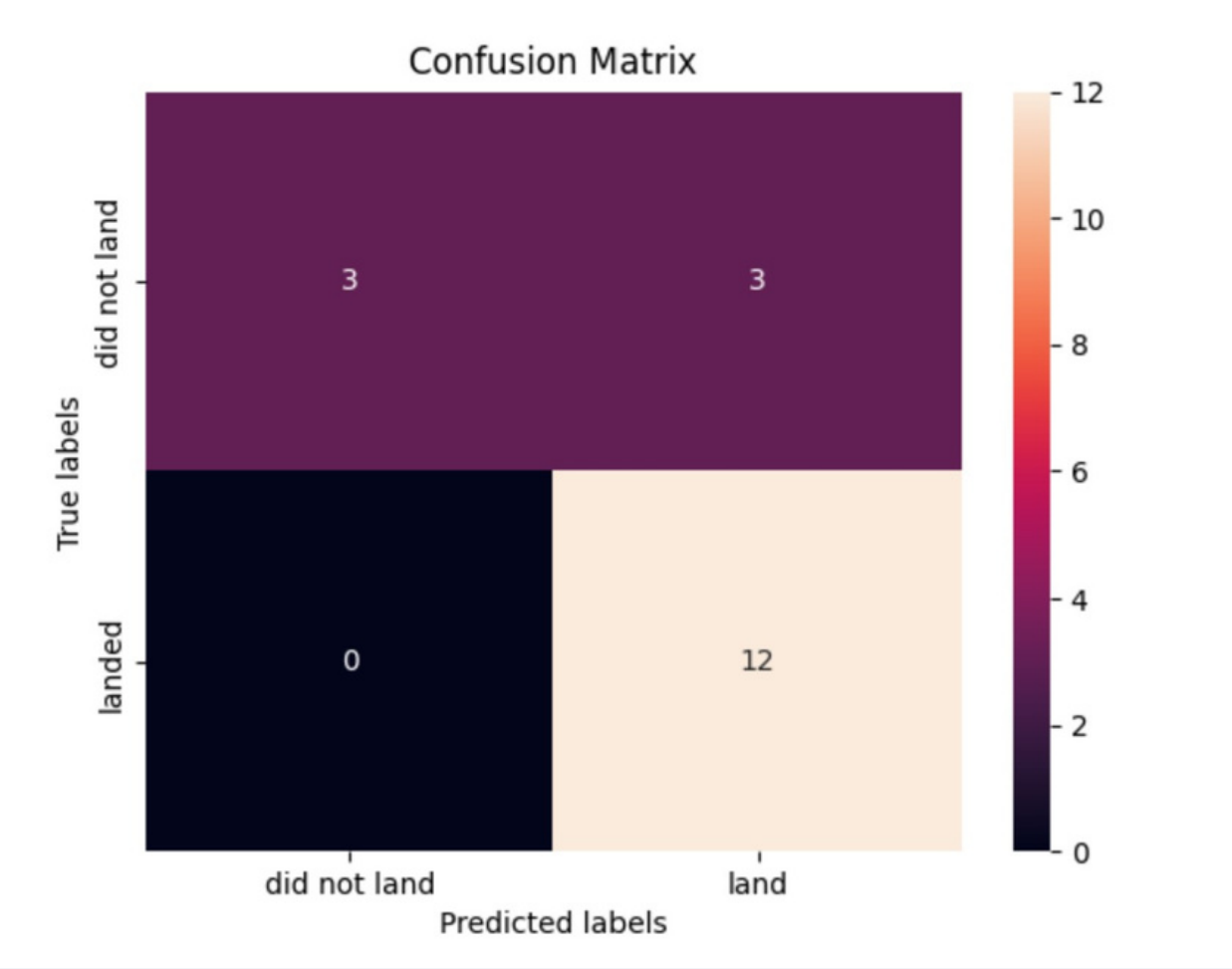
```
tree_cv=GridSearchCV(tree, parameters, cv=10, scoring='accuracy')
tree_cv.fit(X_train,Y_train)
```

```
GridSearchCV(cv=10, estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 4, 6, 8, 10, 12, 14, 16, 18],
                         'max_features': ['auto', 'sqrt'],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [2, 5, 10],
                         'splitter': ['best', 'random']},
             scoring='accuracy')
```

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

```
tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 6, 'max_features': 'sqrt', 'min_samples_leaf': 4,
'min_samples_split': 10, 'splitter': 'best'}
accuracy : 0.8928571428571429
```

Confusion Matrix



Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

- https://github.com/YawningFold/IBM

Thank you!