IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

L. Brown
06 July 2025

Outline

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

Executive Summary

Methodology Summary:

- Data Wrangling to find patterns and determine the label for the prediction model
- Exploratory Data Analysis (EDA) using visualizations and SQL to
- Interactive visual analytics using Folium and Plotly Dash to identify the optimal launch site
- Predictive analysis to develop the most accurate predictive model

- Results Summary:
- The optimal label for data mining is the success or failure of a rocket launch, modeled in the data with a 1 (success) or 0 (failure)
- Exploratory Data Analysis (EDA) using visualizations and SQL to query the data and create visualizations to uncover interesting patterns in the data
- Interactive visual analytics using Folium and Plotly Dash to identify the optimal launch site
- best performing predictor model is expected to correctly identify the success or failure of a launch approximately 83.3% of time

Introduction

Project background and context

The commercial space race is in progress and currently the clear leader is SpaceX. The company charges about $62 million per rocket launch as compared to $165 for competitors. The most significant cost of a launch is the first stage rocket, which if the stage lands, SpaceX can reuse the booster, thus making it able to charge less.

The purpose of the project to analyze launch data and develop a predictive model on whether a given launch will be successful. The model will give other companies insight on whether a bid against SpaceX is advisable.

Questions to answer:

- What is the optimal launch site?
- What combination of booster, payload, and orbit type is the most and least successful?
- What is the most accurate classification model, based on the available data?

Section 1

Methodology

Methodology

Data collection methodology:

- Data was collected from web scraping the Wikipedia of Falcon9 and Falcon Heavy Launch data (Wikipedia Link) and launch data at the: SpaceX API Link

- Perform data wrangling
  - Data was processed by calculating number and percentage of missing values, determine numerical and categorical columns, number of launches per site, the count of each obit type, and the launch success rate
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash

Methodology

- Perform predictive analysis using classification models
  - Build pipeline and develop Support Vector Machine, Classification Trees, and Logistic Regression models
  - Find the best hyperparameters for each model
  - Evaluate and determine which model performs best on test data

Data Collection

Launch data was collected through the SpaceX API in JSON format and loaded into a pandas data frame for further processing. SpaceX API

Additional launch information was collected by web-scraping the data from a Wikipedia page using the python package BeautifulSoup to collect historical launch information on the Falcon-9 launches.(Wilipedia Launch Page)

Data Collection - SpaceX API

1. API Call and Response:

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
response=requests.get(static_json_url)
response.status_code
200
```

2. Convert the Response JSON to a pandas Data Frame:

```
# Use json_normalize method to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

3. Define and use function for data cleaning:

```
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
```

4. Create subset of desired features in dictionary and create new Data Frame

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
```

5. Create final Data Frame:

```
# Create a data from launch_dict
launch_df = pd.DataFrame(launch_dict)
```

6. GitHub URL

GitHub Link: Data Collection - SpaceX API

Data Collection - Scraping

1. Get Request Object from Wiki Page:

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

0.0s

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

2. Create BeautifulSoup Object to extract data:

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

3. Extract column names:

```
for row in launch_headers:
    name = extract_column_from_header(row)
    if (name is not None) :
        if len(name) > 0:
            #print(len(name))
            column_names.append(name)
```

4. Create Empty Dictionary (partial):

```
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'] = []
```

5. Create final Data Frame from Dictionary:

```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
```

✓ 0.0s

6. GitHub URL

GitHub Link: Data Collection - Scraping

Data Wrangling

Data was processed to find patterns and determine the appropriate label for training supervised models and perform exploratory data analysis.

Exploratory Data Analysis:

Identify and calculate percentage of missing values

```
df.isnull().sum()/len(df)*100
```

Identify numerical and categorical columns

```
df.dtypes
```

Calculate the number of launches on each site

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

Calculate number and occurrence of each orbit

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

Calculate number and occurrence of mission outcome

```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

Create landing outcome label

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise

landing_class = []

for value in df['Outcome']:
    if value in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

Calculate the launch success rate

```
df["Class"].mean()
```

```
0.6666666666666666
```

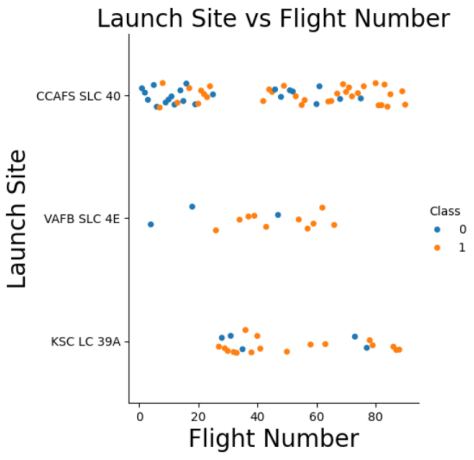
GitHub Link: Data Wrangling

EDA with Data Visualization

Various charts were created to visualize relationships in the data and what features are important in predicting the success or failure of rocket launches.

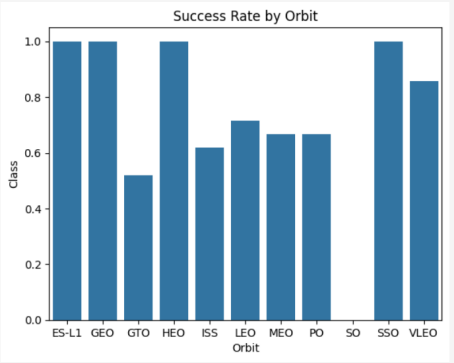
Scatterplots:

- Flight Number vs Payload Mass
- Flight Number vs Launch Site
- Payload Mass vs. Launch Site
- Flight Number vs Orbit Type
- Payload Mass vs Orbit Type



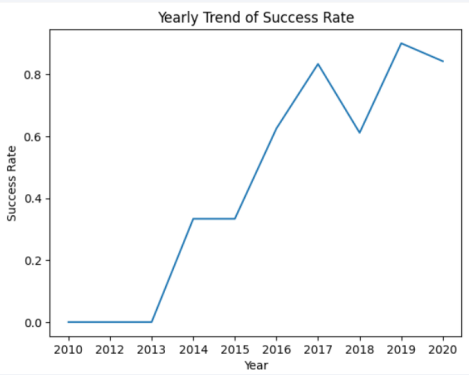
Bar Chart:

- Success Rate of Orbit Types



Line Chart

- Launch success yearly Trend



[GitHub Link: EDA with Data Visualization]

EDA with SQL

Various SQL queries were developed and executed to explore and further enhance understanding of the data. A summarized list of queries :

- Display names of unique launch sites
- Display payload mass values
- Display 5 records of CCA launch sites
- Display the total of all payload masses
- Display the average payload mass
- Display the first successful ground pad landing date
- Display names of booster for successful drone ship landings with payloads between 4,000 and 6,000 kilograms
- Display count of successful and failure mission outcomes
- Display booster versions carrying the maximum payload
- Display drone ship landing failure dates for 2015
- Display the count of landing outcomes between June 4, 2010, and March 20, 2017

GitHub Link: EDA with SQL

Build an Interactive Map with Folium

A Folium map was created with various markers to develop insights around the launch locations, whether the launch was successful or not, how close the sites are to infrastructure (city, road, railway, etc.) and the distance to the nearest coastline.

Summary of map objects:

- A circle map marker for each launch site, with popup for the site name
- Marker clusters for all launch records at each site, with green for successful launches and red for failures
- Polylines to explore and identify proximities of the launch sites, such as roads, railways, and nearest coastline.

The map revealed that launch sites are in more remote areas and rockets are normally launched over the ocean to minimize the risk of damage and injury.

GitHub Link: Interactive Map with Folium

Build a Dashboard with Plotly Dash

An interactive dashboard was built to perform interactive visual analytics on the SpaceX launch data. The goal of the dashboard was to attempt to answer these questions:

- Which site had the most successful launches?
- Which site had the highest success rate?
- What payload weight range has the highest and lowest success rates:
- Which F9 Booster Version has the highest success rate:

An interactive pie chart was created to show the overall success rates for all launch sites and to dynamically select each launch site and see the specific success rate for that site.

An interactive scatterplot was created to show the correlation between payload mass (kilograms) and success/failure, allowing for dynamic selection of different payloads.

GitHub Link: Dashboard with Plotly Dash

Predictive Analysis (Classification)

A predictive model was created to predict if the first stage of a rocket launch will land, based on data from previous launches. The following steps were completed to develop several models (logistic regression, support vector machine, decision tree classifier, and k-nearest neighbors) and then evaluate and determine the best performing model.

1. Read the data

```
URL1 = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_2.csv"
# resp1 = await fetch(URL1)
# text1 = io.BytesIO((await resp1.arrayBuffer()).to_py())
data = pd.read_csv(URL1)
```

2. Create NumPy array for the Class variable

```
#Create numpy arrays for features and target variable
Y = (data['Class']).to_numpy()
Y
```

3. Standardize the array using StandardScaler() and use the fit_transform() the data

```
# Standardize the data in X and then transform and assign to X
transform = preprocessing.StandardScaler()
X = transform.fit_transform(X)
```

4. Split the data into training and test splits using train_test_split()

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
```

5. Create logistic regression model using GridSearchCV() and check the accuracy with a confusion matrix

```
logreg_cv = GridSearchCV(lr, parameters, cv=10)
logreg_cv.fit(X_train, Y_train)
```

```
yhat=logreg_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

Predictive Analysis (Classification) - continued

6. Create support vector machine object model using GridSearchCV() and check the accuracy with a confusion matrix

```
# Instantiate the GridSearchCV object: svm_cv
svm_cv = GridSearchCV(svm, parameters, cv=10)
# Fit it to the data
svm_cv.fit(X_train, Y_train)
```

```
yhat=svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

7. Create decision tree classifier model using GridSearchCV() and check the accuracy with a confusion matrix

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

```
yhat = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

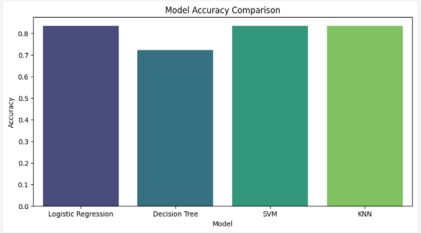
8. Create k-nearest neighbors model using GridSearchCV() and check the accuracy with a confusion matrix

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

```
yhat = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

9. Chart and compare each model's accuracy to select the best performer

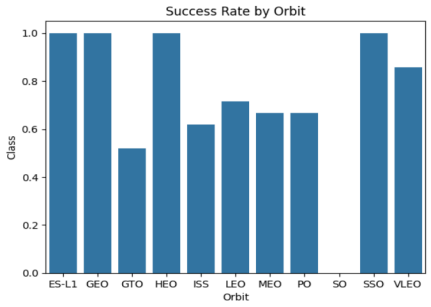
```
# bar chart of model accuracy
plt.figure(figsize=(10, 5))
sns.barplot(x='Model', y='Accuracy', data=summary, palette='viridis')
plt.title('Model Accuracy Comparison')
plt.xlabel('Model')
plt.ylabel('Accuracy')
plt.show()
```



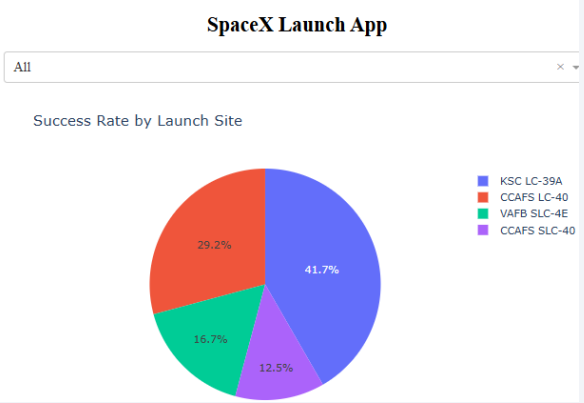
GitHub Link: Machine Learning Prediction

Results

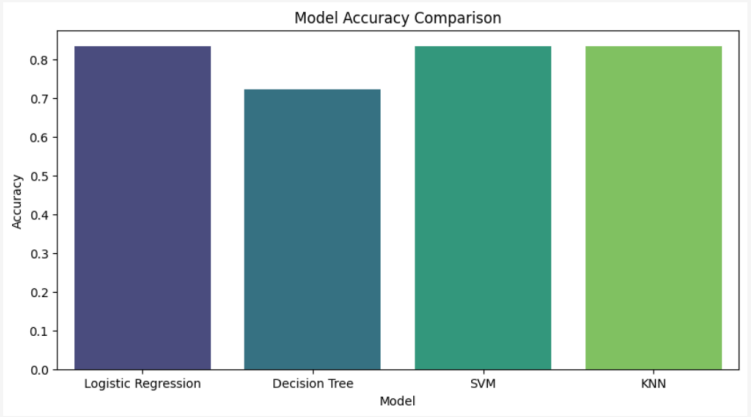
Exploratory data analysis clearly shows some Orbit Types had higher success rates than others:



Interactive analytics demo show the overall success rate for all launch sites and the correlation between booster versions and payloads.



Predictive analysis reveals that the three of four models have similar accuracy, but the decision tree model lagged the other models.

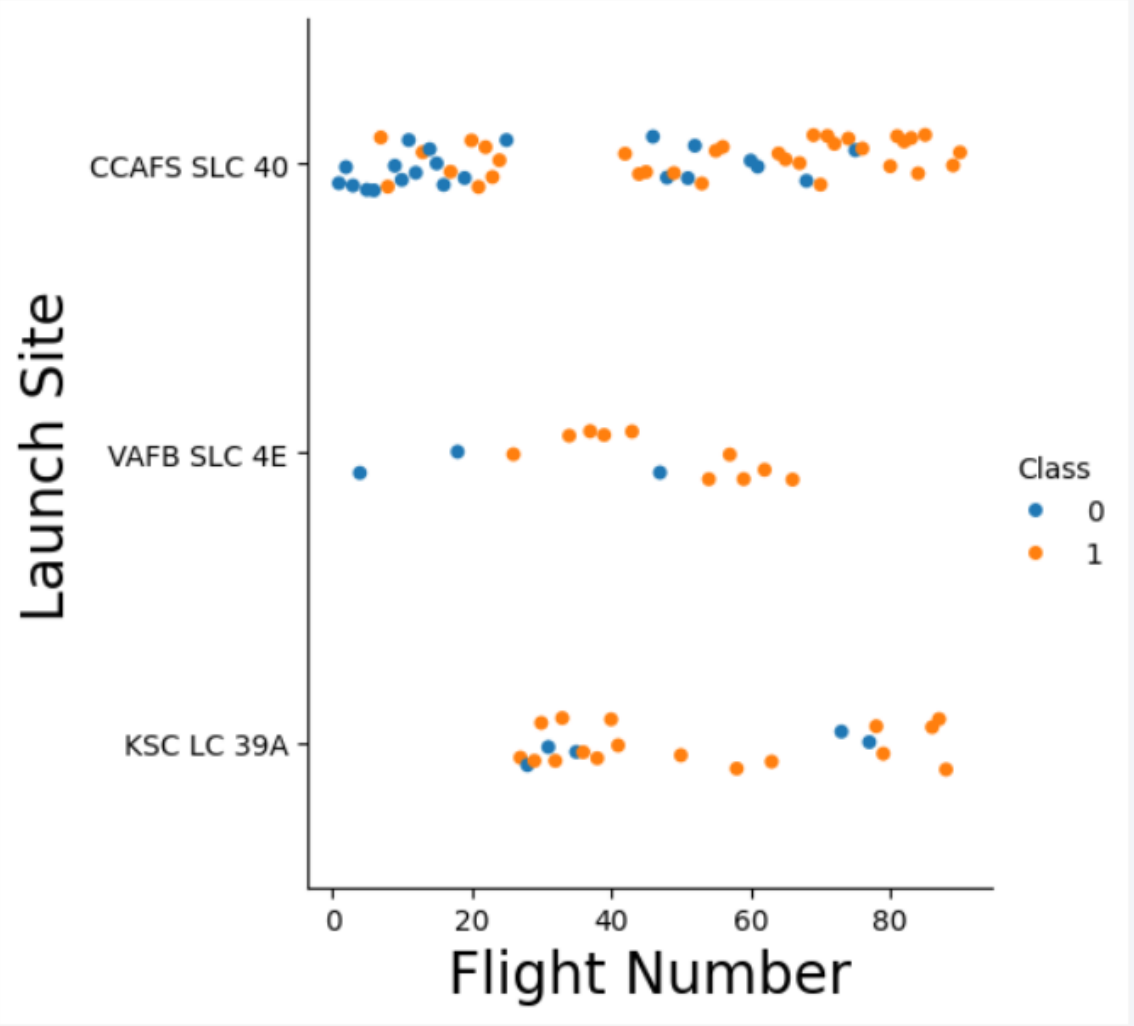


Section 2

Insights drawn from EDA

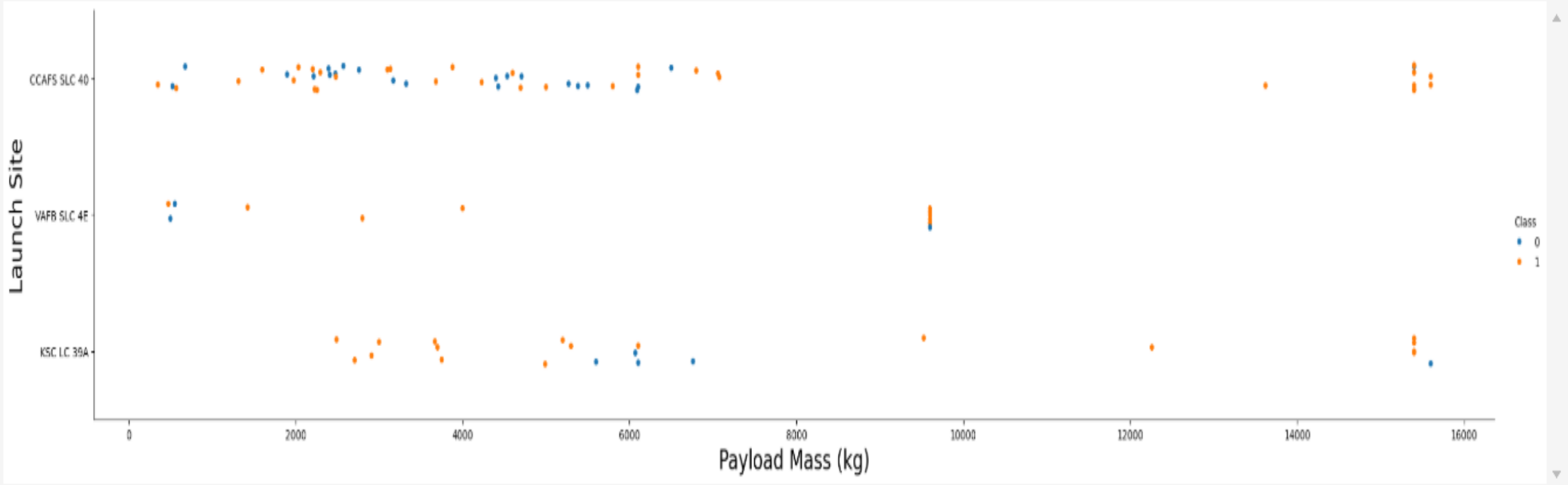
Flight Number vs. Launch Site

The scatter plot illustrates the relationship between Flight Number and Launch Site. Class 1 observations are successful launches and Class 0 were not successful.



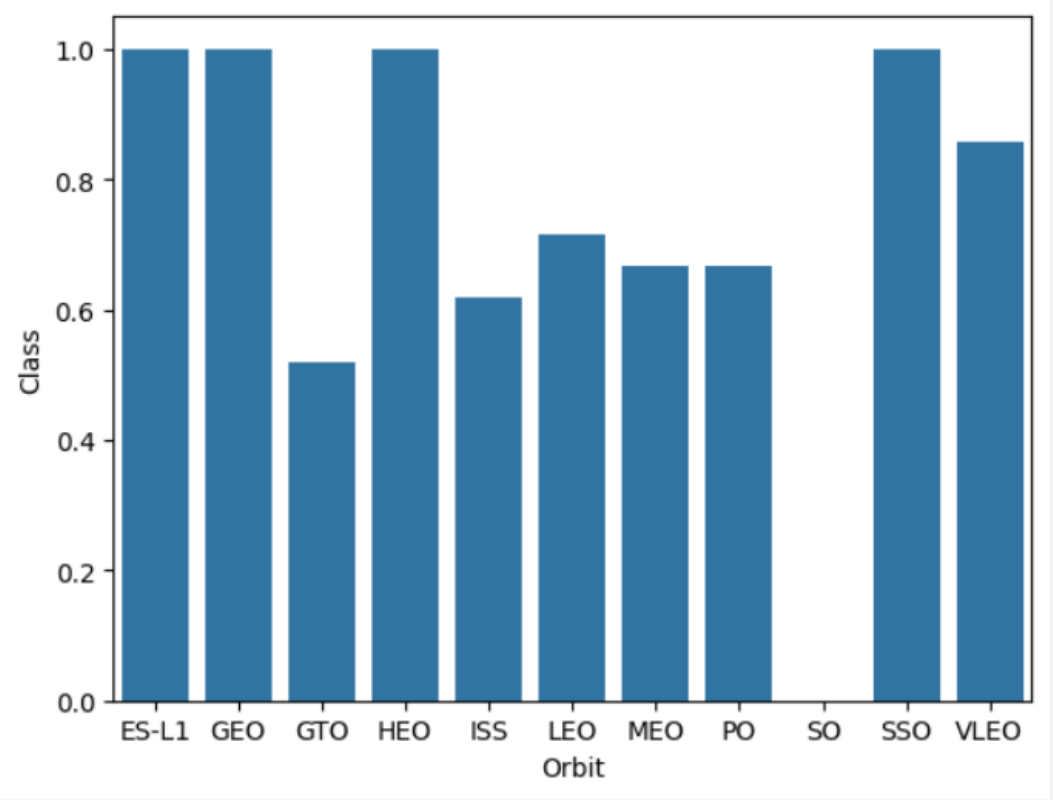
Payload vs. Launch Site

The scatter plot illustrates the relationship between the Payload Mass (kg) and Launch Site. Class 1 observations are successful launches and Class 0 were not successful.



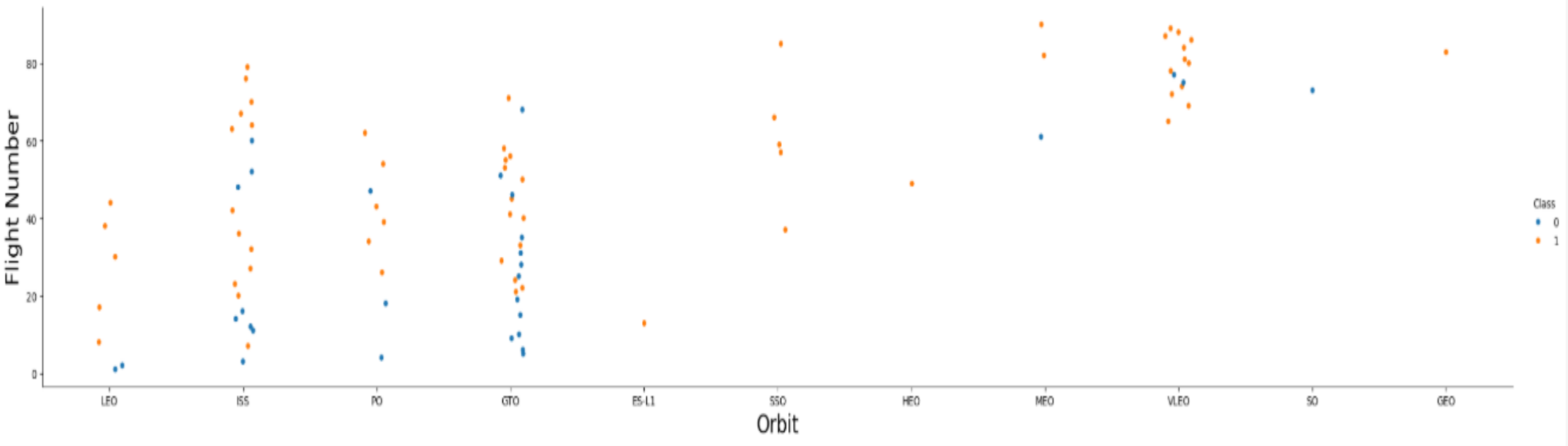
Success Rate vs. Orbit Type

The chart shows that ES-L1, GEO, HEO, and SSO have 100% rates of success. The SO orbit type has no successful launches. Other orbit types (GTO, ISS, LEO, MEO, PO, and VLEO) have varying rates of successful launches, ranging from about 50% to about 83%.



Flight Number vs. Orbit Type

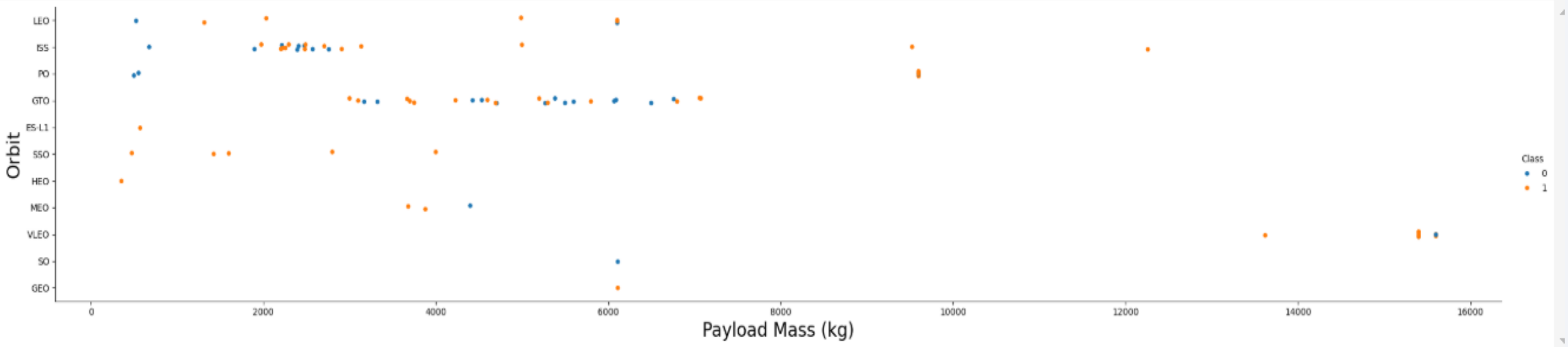
The scatterplot shows the relationship between the flight numbers and the Orbit type. Class 1 are successful launches and Class 0 are non-successful launches.



Payload vs. Orbit Type

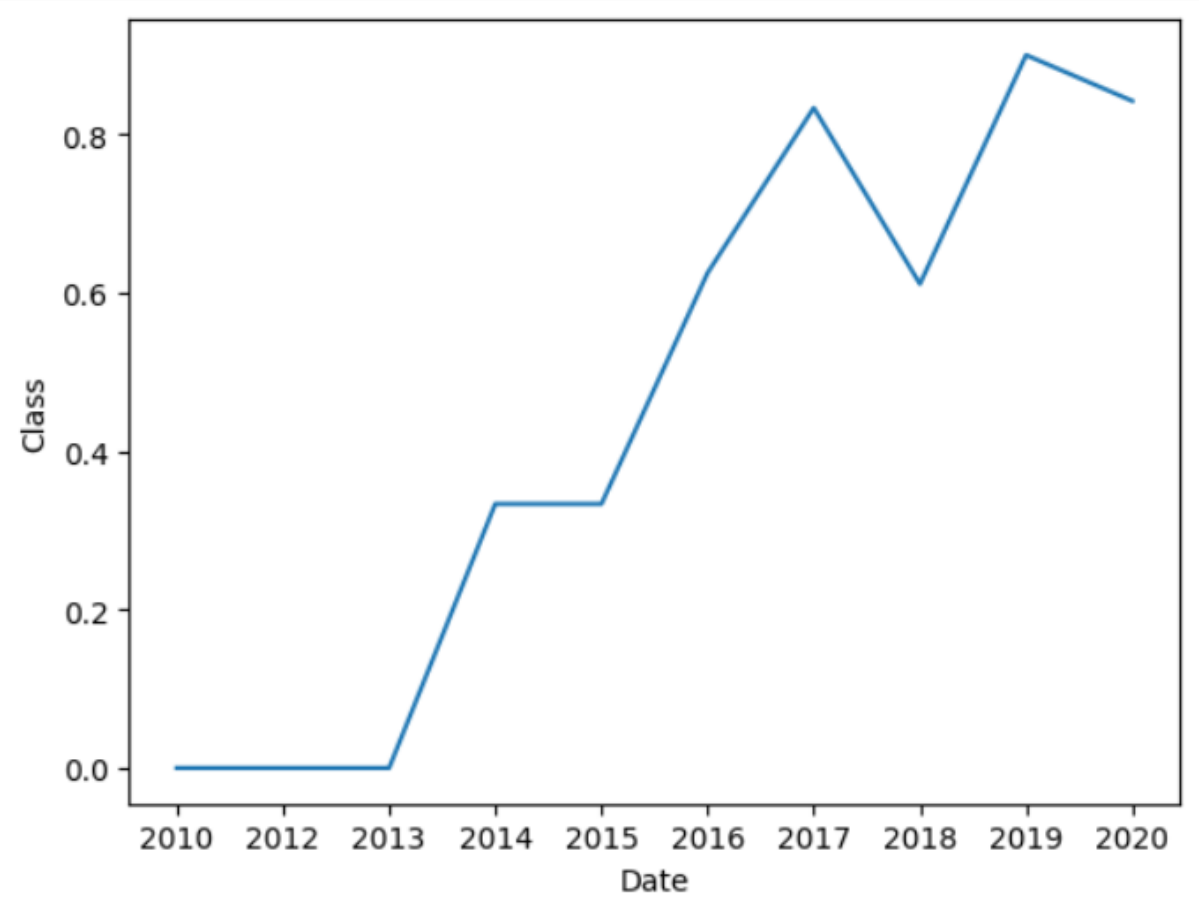
- Show a scatter point of payload vs. orbit type

- Show the screenshot of the scatter plot with explanations



Launch Success Yearly Trend

- Show a line chart of yearly average success rate
- Show the screenshot of the scatter plot with explanations



All Launch Site Names

The query selects unique sites by using the “distinct” keyword and returning the values from the spacetable database

```
1  %sql select distinct launch_site from spacextable order by launch_site
```

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

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

Launch Site Names Begin with 'CCA'

The query selects sites that have the letters “CCA” in the name and returns five (5) the values from the spacetable database

```
1  %sql select launch_site from spacextable where launch_site like '%CCA%' limit 5
```

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

Launch_Site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Total Payload Mass

The query sums the entire payload_mass_kg column and returns a scalar value for the total number of kilograms of payload in the spacetable database

```
1  %sql select sum(PAYLOAD_MASS__KG_) from spacextable
```

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

sum(PAYLOAD_MASS__KG_)
619967

Average Payload Mass by F9 v1.1

The query calculates the average of the PAYLAOD_MASS_KG column where the booster version is F9 v1.1 and returns a scalar value from the spacetable database

```
1  %sql select Booster_version, avg(PAYLOAD_MASS__KG_) from spacextable where Booster_version like '%F9 v1.1%'
```

✓ 0.0s

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

Booster_Version	avg(PAYLOAD_MASS__KG_)
F9 v1.1 B1003	2534.6666666666665

First Successful Ground Landing Date

The query returns the earlier date of a successful landing on a ground pad and returns the date from the spacetable database

```
1  %sql select min(Date) from spacextable where Landing_outcome like '%ground%'
```

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

min(Date)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

The query return the names of the booster that successfully land on a drone shop and had payloads between 4000 and 6000 kilograms. The conditions in the where clause limit the returns to only those booster matching all the criteria.

```
%sql select Booster_Version, Landing_Outcome, PAYLOAD_MASS__KG_
from spacextable
where Landing_Outcome like '%Success (drone ship)%'
and PAYLOAD_MASS__KG_ >4000  and PAYLOAD_MASS__KG_ < 6000 order by PAYLOAD_MASS__KG_
```

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

Booster_Version	Landing_Outcome	PAYLOAD_MASS__KG_
F9 FT B1026	Success (drone ship)	4600
F9 FT B1022	Success (drone ship)	4696
F9 FT B1031.2	Success (drone ship)	5200
F9 FT B1021.2	Success (drone ship)	5300

Total Number of Successful and Failure Mission Outcomes

The query counts each type of mission outcome and returns the data based on the on the mission outcome from the spacetable database.

```
1  %sql select Mission_Outcome, count(Mission_Outcome) from spacextable group by Mission_Outcome
```

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

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

Boosters Carried Maximum Payload

The query lists the distinct booster versions that were launched with the maximum payload weight and returns values from the spacetable database

```
%%sql select distinct Booster_Version
from spacextable
where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from spacextable)
```

✓ 0.0s

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

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

The query returns the all the failed outcomes with drone shops, the booster version, and the launch site and returns the values from the spacetable database

```
%sql select Date,Booster_Version, Launch_Site, Landing_Outcome
from spacextable
where Date > '2014-12-31' and Date < '2016-01-01' and Landing_Outcome like '%Failure (drone ship)%'
```

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

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

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

The query ranks the landing outcomes between 2010-06-04 and 2017-03-20, in descending order from the spacetable database. The where clause limits the range of dates to those of interest.

```
%sql select Landing_Outcome, count(Landing_Outcome)
from spacextable
where Date > '2010-06-04' and Date < '2017-03-20' group by Landing_outcome
```

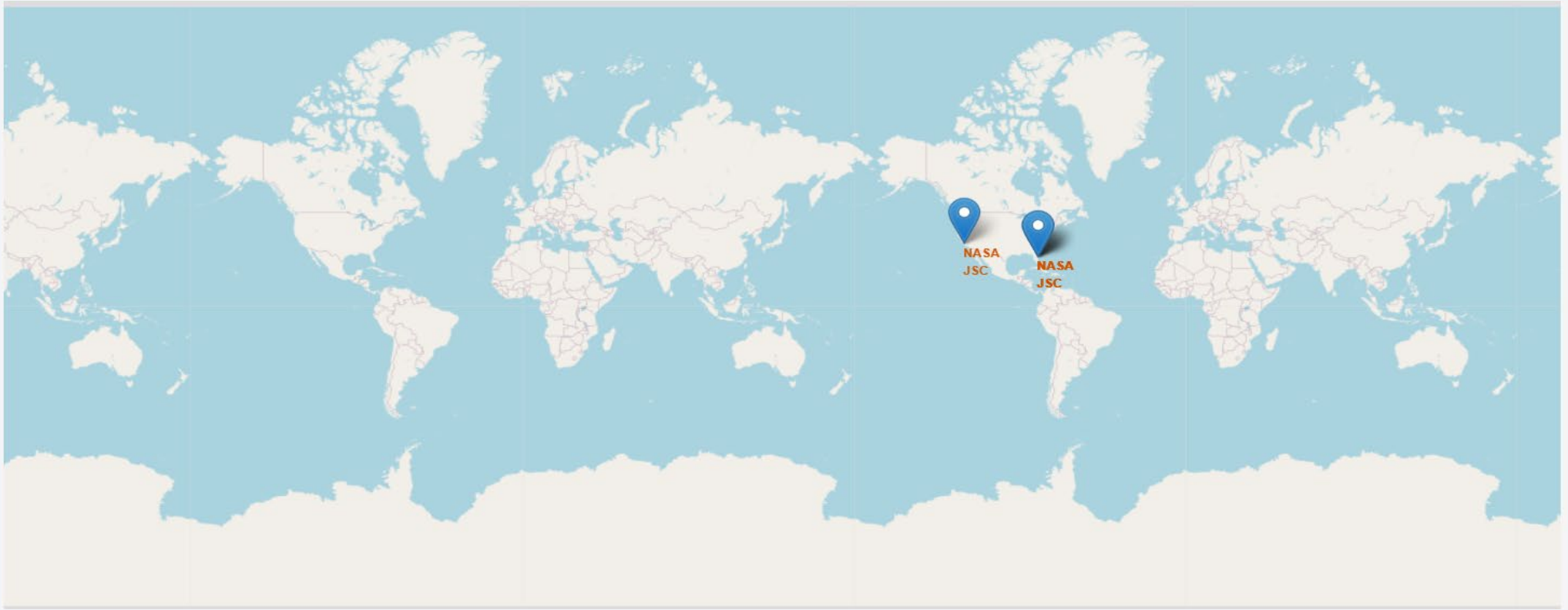
```
 * sqlite:///my_data1.db
Done.
```

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


Section 3
Launch Sites
Proximities Analysis

Global View of SpaceX Launch Sites

A global view of the SpaceX launch site. All in the United States and include sites on both east and west coasts.

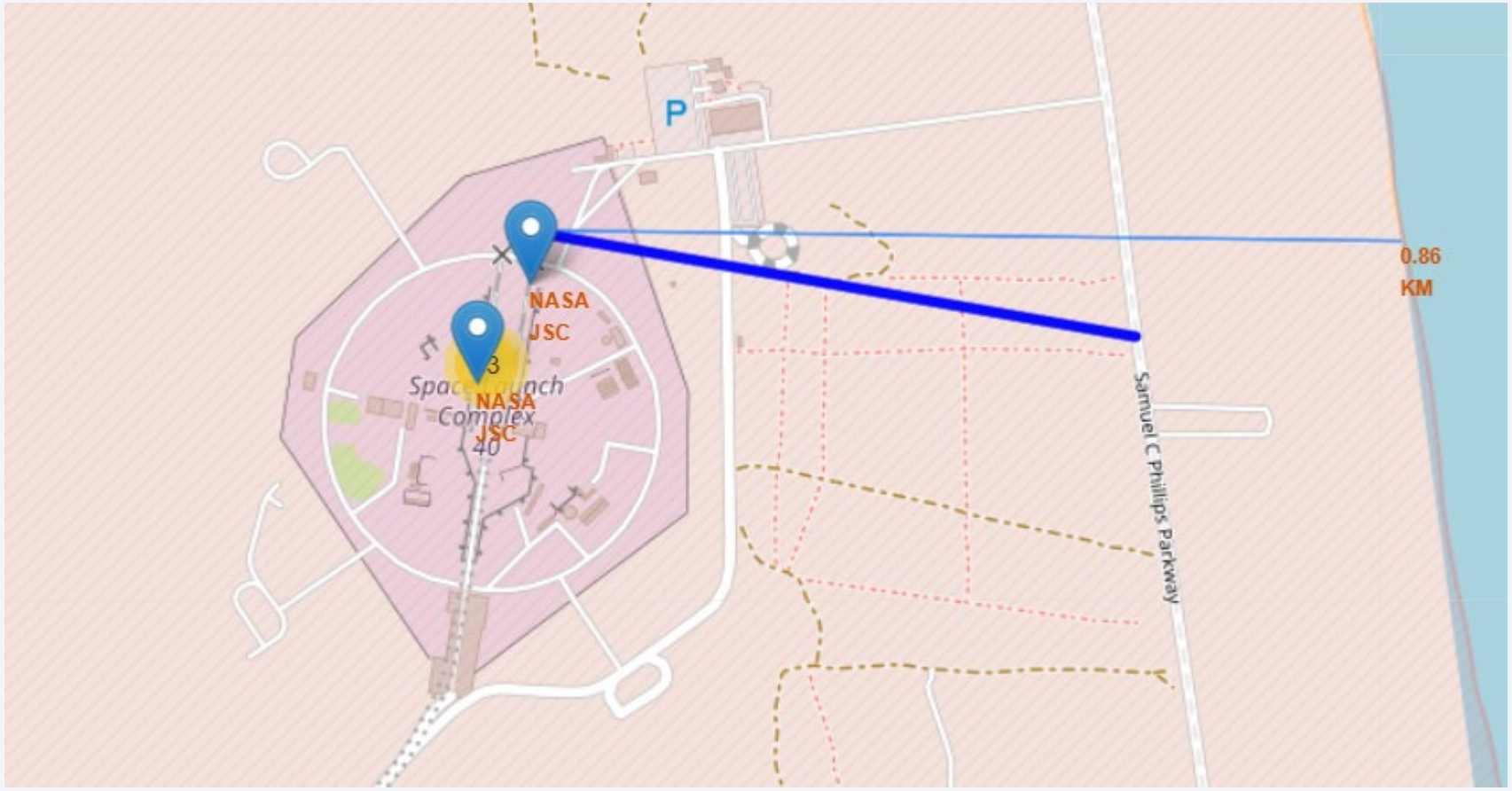


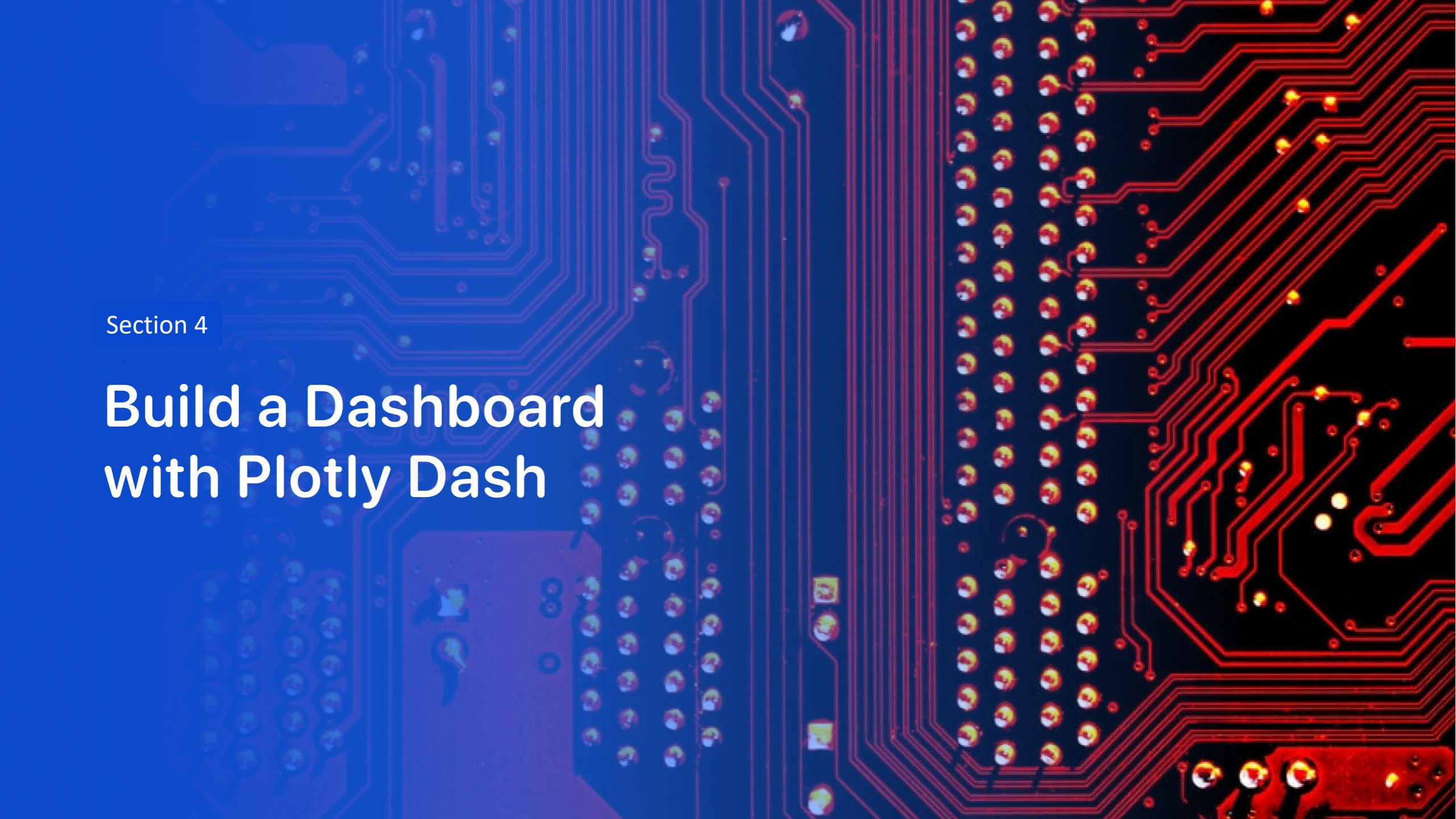
Success/Failed Launches for Site CCAFS LC-40

The red and green markers indicated successful (green) and failure (red) outcomes of rocket launches.
The frequency of markers indicated how successful the launch site has been.

Map View of Distance to Coast and Highway

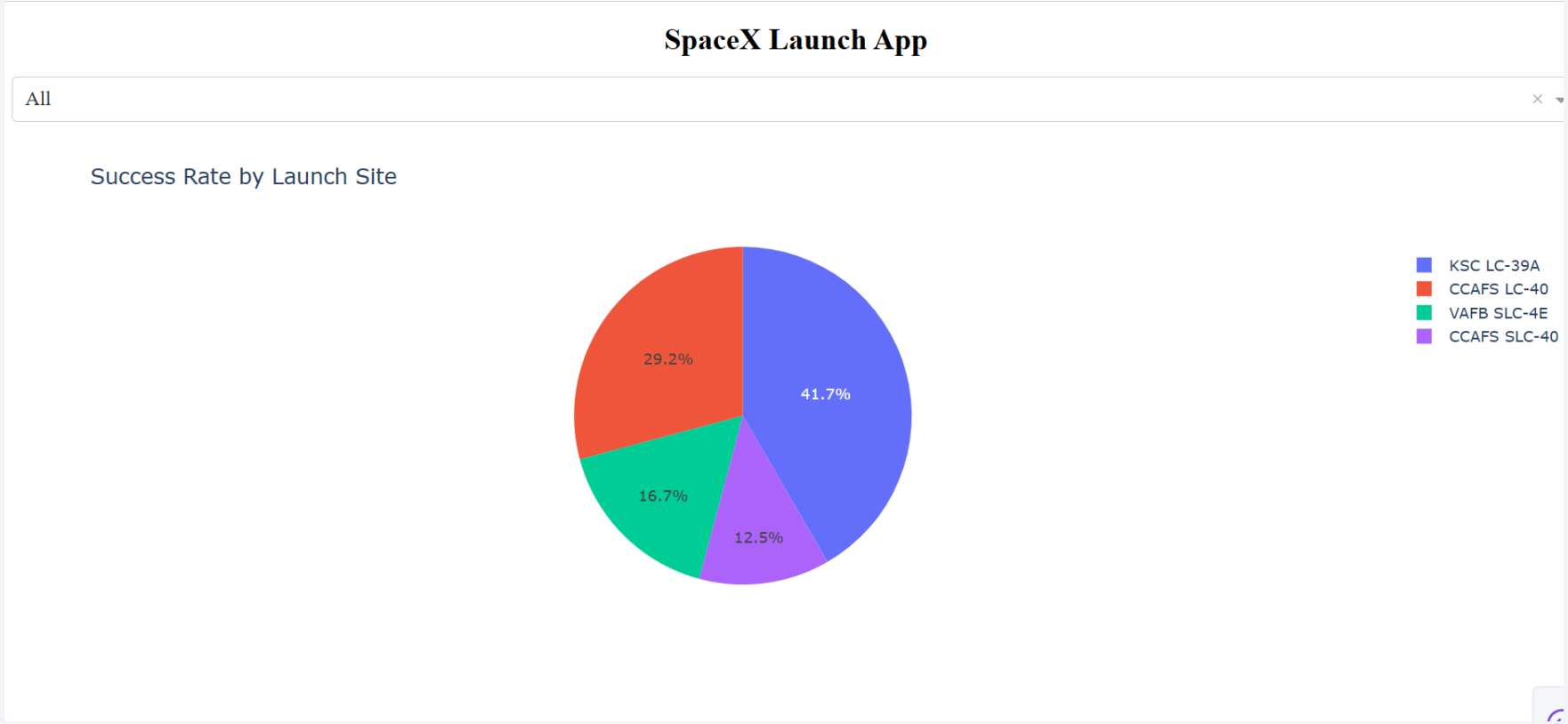
The lines give a visual indicator of proximities are to the launch sites.




Section 4
Build a Dashboard with Plotly Dash

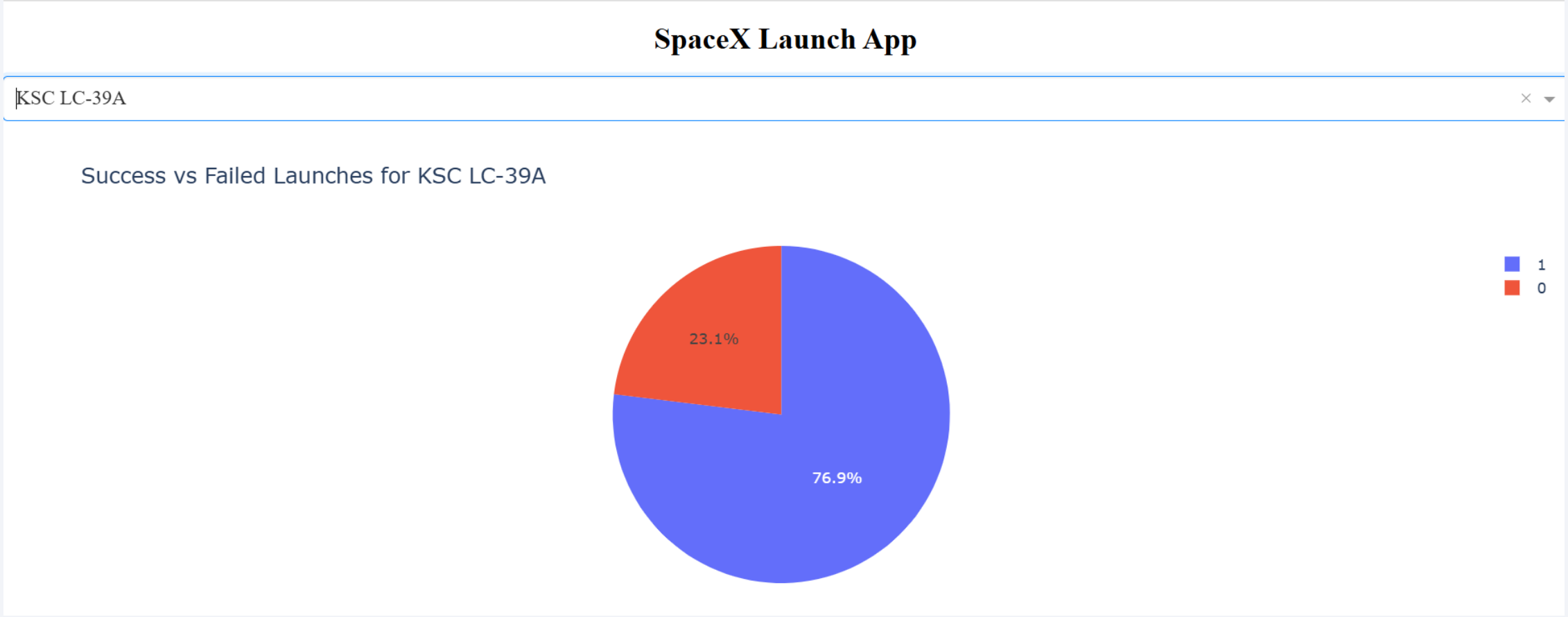
Success Rate for All Launch Sites

The chart displayed shows the success rate by all launch sites. The chart is interactive and each individual site can be selected and the chart will display the success rate for that site.



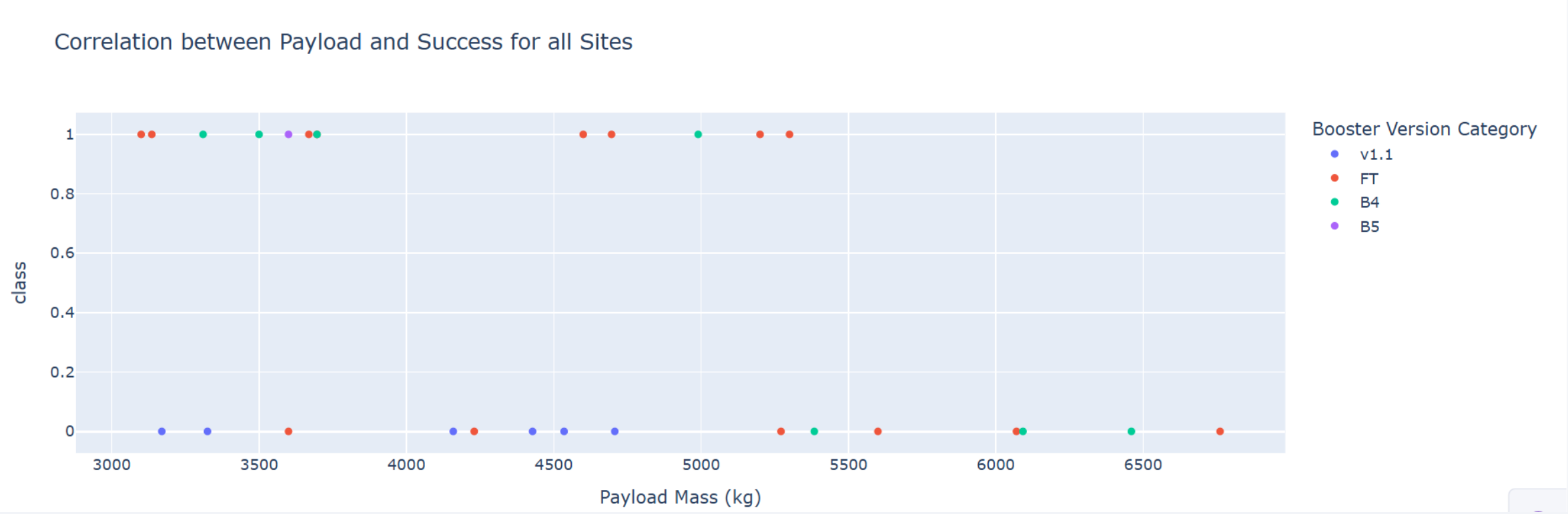
Success vs Failed Launches for KSC LC-39A

The chart displayed shows the interactivity of the launch app, indicating that site KSC LC-39A has a success rate of 76.9% and failure rate of 23.1%.



Correlation between Payload and Success for all Sites

The scatterplot shows the correlation between the payload amounts and success rates for all sites by booster version. Booster version FT shows better success than other versions.

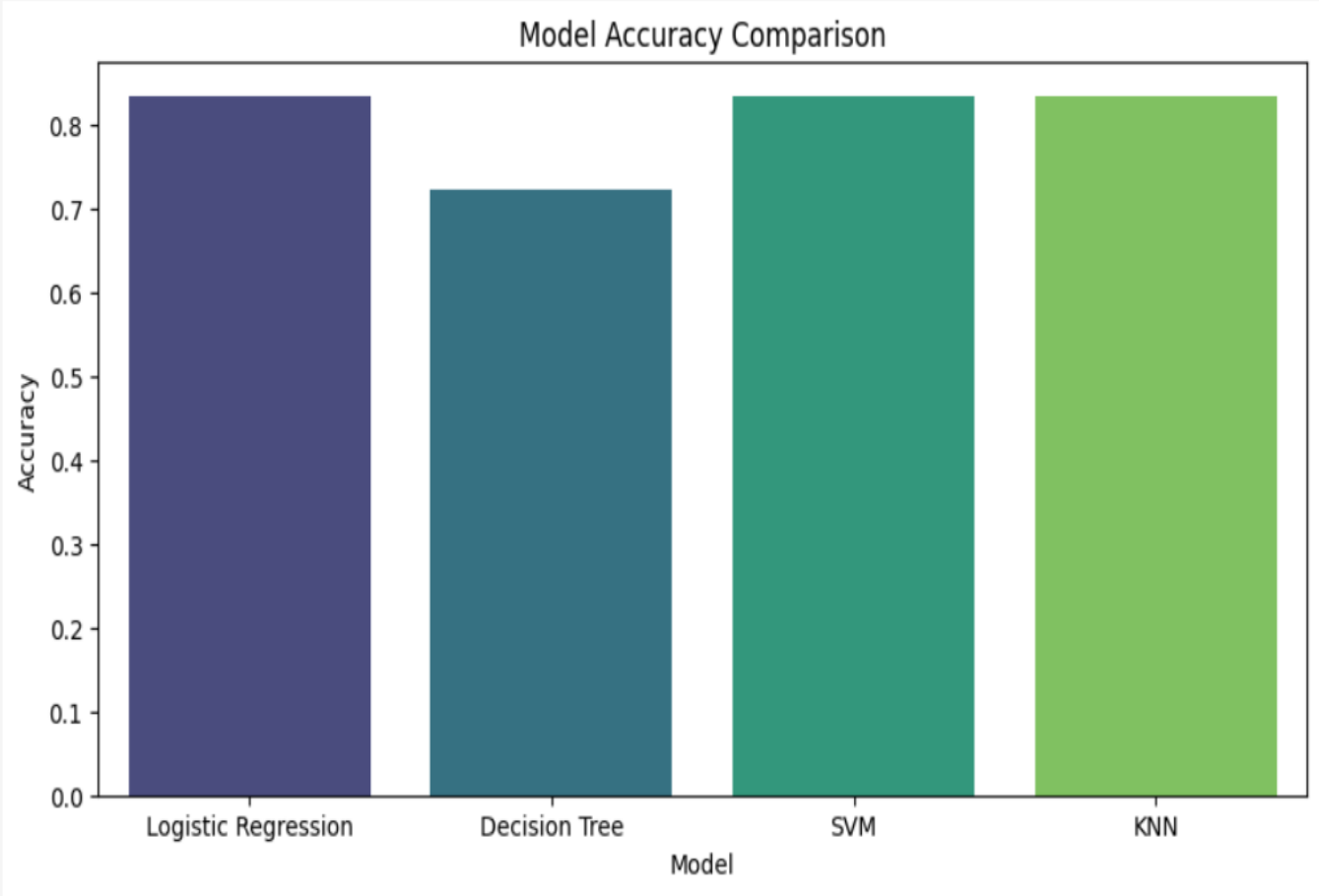


Section 5

Predictive Analysis (Classification)

Classification Accuracy

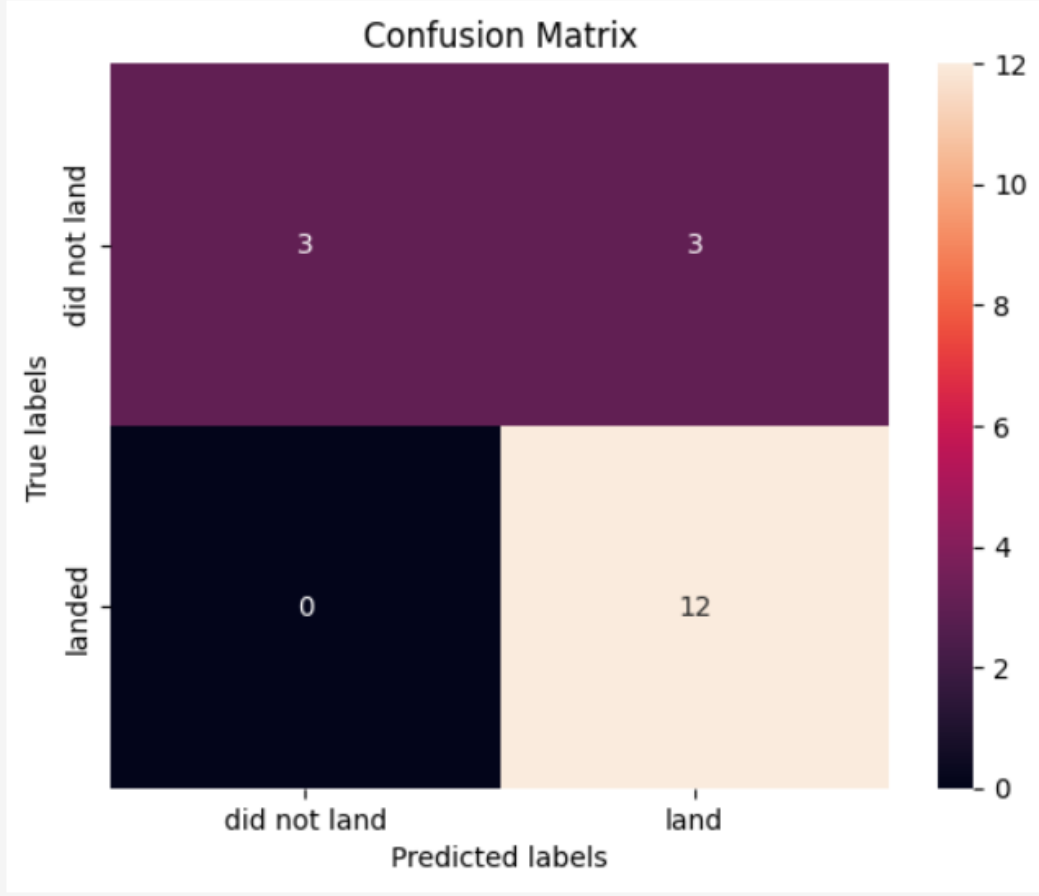
- The accuracy of the logistic regression, support vector machine (SVM) and the KNN models are very similar. The decision tree model lagged other models in accuracy.
- The Logistic Regression model by a few hundredths had a slightly better accuracy rate than the other models.



Confusion Matrix

- The confusion matrix for the logistic regression model shows the correct model prediction in the top left and bottom right. The top right and bottom left values show where the model made an incorrect prediction.

```
1  yhat=logreg_cv.predict(X_test)
2  plot_confusion_matrix(Y_test,yhat)
```



Conclusions

- Based on the data and visual line chart, SpaceX has increased the number of successful launches over time periods in the data.
- The success rate for some Orbit Types (ES-L1, GEO, HEO, and SSO have much higher success rates than other obit types.
- Launch site KSC LC-39A has the largest launch success rate of all the sites and an overall success rate of 76.9%.
- The best performing predictor model is expected to correctly identify the success or failure of a launch approximately 83.3% of time.

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

All code, SQL queries, charts, notebooks, and the launch data database is included the GitHub repository and is available at this link:

GitHub Link to Main Page

Thank you!