IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

RIYANSH SACHDEV
16-09-2024

Outline

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

Executive Summary

The following methodologies were employed to analyze the data:

- Data Collection: Data was gathered through web scraping and the SpaceX API.
- Exploratory Data Analysis (EDA): This included data wrangling, visualization, and interactive visual analytics.
- Machine Learning Prediction: Models were developed to predict outcomes.

Summary of Results

- Valuable data was successfully collected from public sources.
- EDA helped identify the most significant features for predicting launch success.
- Machine learning predictions revealed the best model for determining which characteristics most effectively drive launch success, utilizing all collected data.

Introduction

- The objective is to evaluate the viability of the new company Space Y to compete with Space X.
- Desirable answers:
  - The best way to estimate the total cost for launches, by predicting successful landings of the first stage of rockets;
  - Where is the best place to make launches.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data from Space X was obtained from 2 sources:

  Space X API (https://api.spacexdata.com/v4/rockets/)

  - WebScraping

  (https://en.wikipedia.org/wiki/List_of_Falcon/_9/_and_Falcon_Heavy_launches)
- Perform data wrangling
  - Collected data was enriched by creating a landing outcome label based on outcome data after summarizing and analyzing features
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Data that was collected until this step were normalized, divided in training and test data sets and evaluated by four different classification models, being the accuracy of each model evaluated using different combinations of parameters.

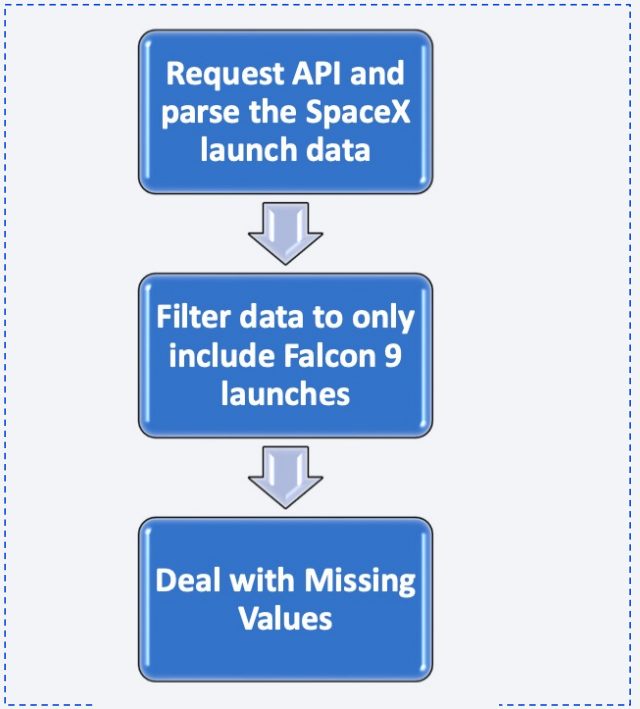
Data Collection

- Data sets were collected from Space X API (https://api.spacexdata.com/v4/rockets/) and via Wikipedia.
- We also used web scraping techniques.

Data Collection – SpaceX API

- SpaceX offers a public API from where data can be obtained and used.
- This API was used according to the flowchart attached.

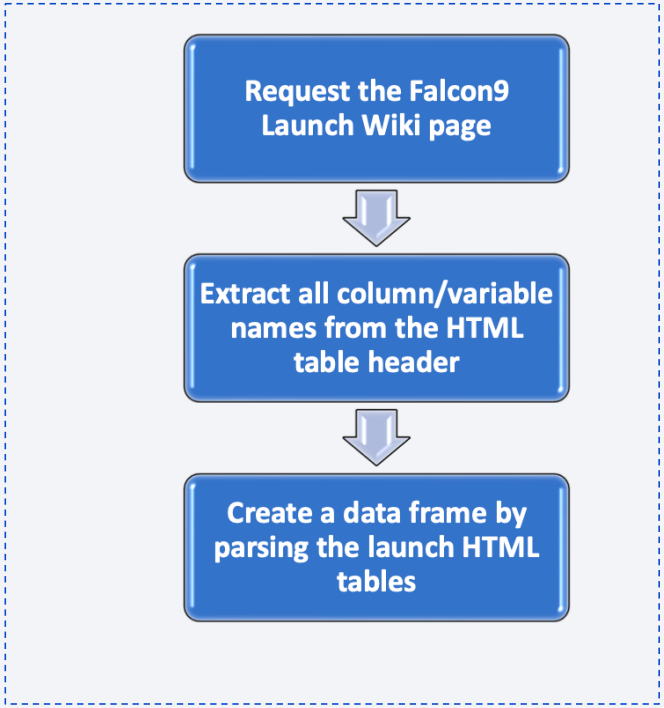
- https://github.com/riyansh100/Applied-Data-Science-Capstone/blob/main/Data%20Collection%20API.ipynb



Data Collection - Scraping

• Data was collected from SpaceX launches And from Wikipedia.

https://github.com/riyansh100/Applied-Data-Science-Capstone/blob/main/Data%20Collection%20with%20Web%20Scraping.ipynb



Data Wrangling

- Initially some Exploratory Data Analysis (EDA) was performed on the dataset.
- Then the summaries launches per site, occurrences of each orbit and occurrences of mission outcome per orbit type were calculated.
- Finally, the landing outcome label was created from Outcome column.

https://github.com/riyansh100/Applied-Data-Science-Capstone/blob/main/Data%20Wrangling.ipynb

EDA with Data Visualization

• To explore data, scatterplots, barplots, etc. were used to visualize the relationship between various pair of features.

https://github.com/riyansh100/Applied-Data-Science-Capstone/blob/main/EDA%20with%20Visualization.ipynb

EDA with SQL

SQL queries performed were :

- Names of the unique launch sites in the space mission;
- Top 5 launch sites whose name begin with the string 'CCA';
- Total payload mass carried by boosters launched by NASA (CRS);
- Average payload mass carried by booster version F9 v1.1;
- Date when the first successful landing outcome in ground pad was achieved;
- Names of the boosters which have success in drone ship and have payload mass between 4000 and 6000 kg;
- Total number of successful and failure mission outcomes;
- Names of the booster versions which have carried the maximum payload mass;
- Failed landing outcomes in drone ship, their booster versions, and launch site names for in year 2015; and
- Rank of the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20.

https://github.com/riyansh100/Applied-Data-Science-Capstone/blob/main/EDA%20with%20SQL.ipynb

Build an Interactive Map with Folium

- Markers indicate points like launch sites;
- Circles indicate highlighted areas around specific coordinates, like NASA Johnson Space

Center;

- Marker clusters indicates groups of events in each coordinate, like launches in a launch site;

and

- Lines are used to indicate distances between two coordinates.

https://github.com/riyansh100/Applied-Data-Science-Capstone/blob/main/Interactive%20Visual%20Analytics%20with%20Folium.ipynb

Build a Dashboard with Plotly Dash

The following graphs and plots were used to visualize data

- Percentage of launches by site
- Payload range

This combination allowed to quickly analyze the relation between payloads and launch sites, helping to identify where is best place to launch according to payloads.

https://github.com/riyansh100/Applied-Data-Science-Capstone/blob/main/Dashboard%20Plotly.py

Predictive Analysis (Classification)

• Four classification models were compared: logistic regression, support vector machine, decision tree and k nearest neighbours.

https://github.com/riyansh100/Applied-Data-Science Capstone/blob/main/Machine%20Learning%20Prediction.ipynb

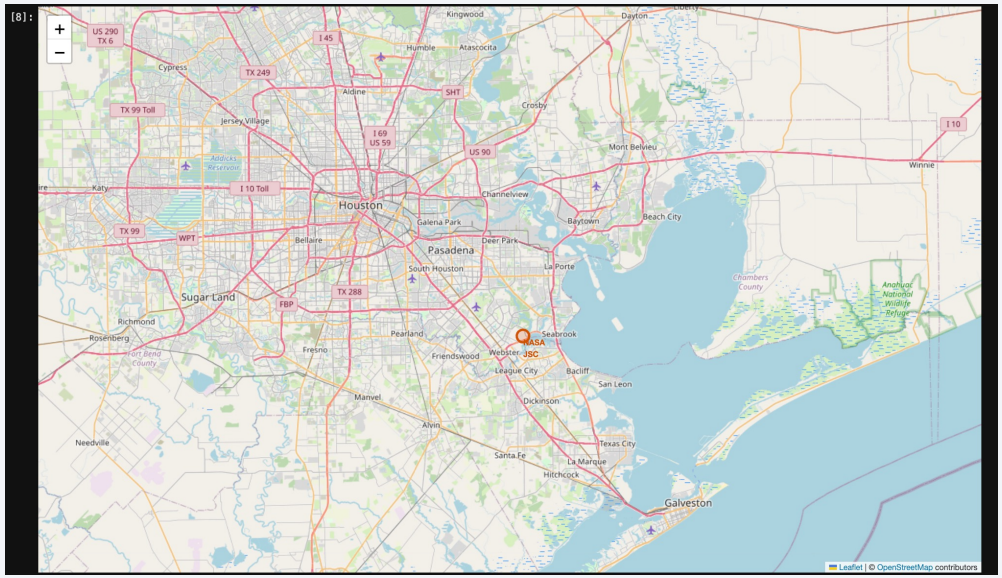
Results

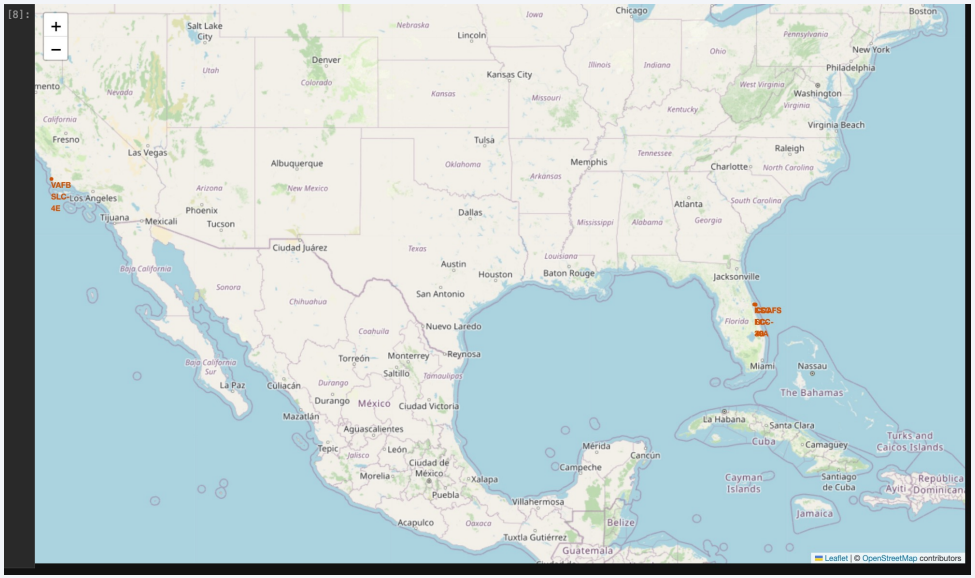
Exploratory data analysis results

- Space X uses 4 different launch sites;
- The first launches were done to Space X itself and NASA;
- The average payload of F9 v1.1 booster is 2,928 kg;
- The first success landing outcome happened in 2015 fiver year after the first launch;
- Many Falcon 9 booster versions were successful at landing in drone ships having payload above the average;
- Almost 100% of mission outcomes were successful;
- Two booster versions failed at landing in drone ships in 2015: F9 v1.1 B1012 and F9 v1.1 B1015;
- The number of landing outcomes became as better as years passed.

Results

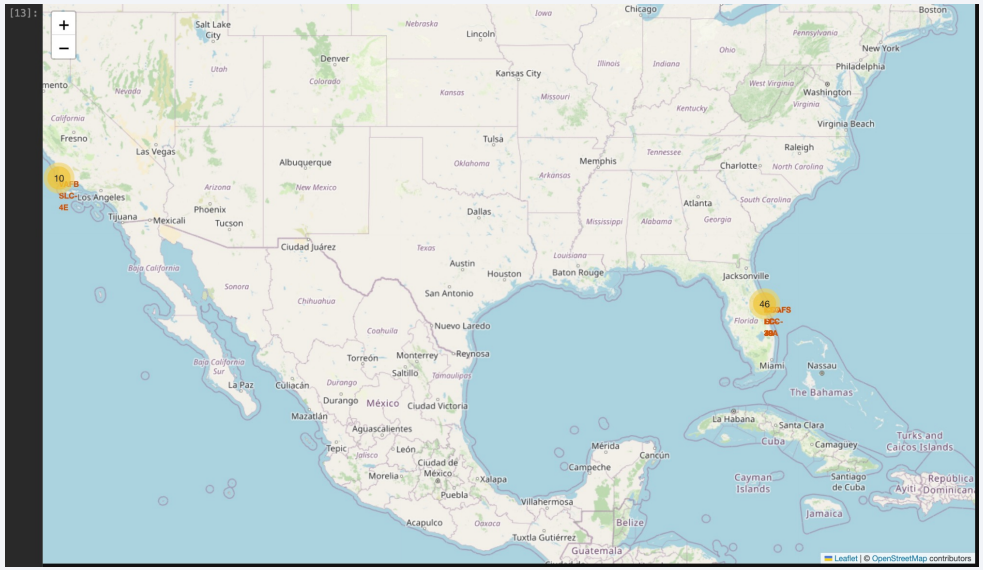
- Interactive analytics demo in screenshots

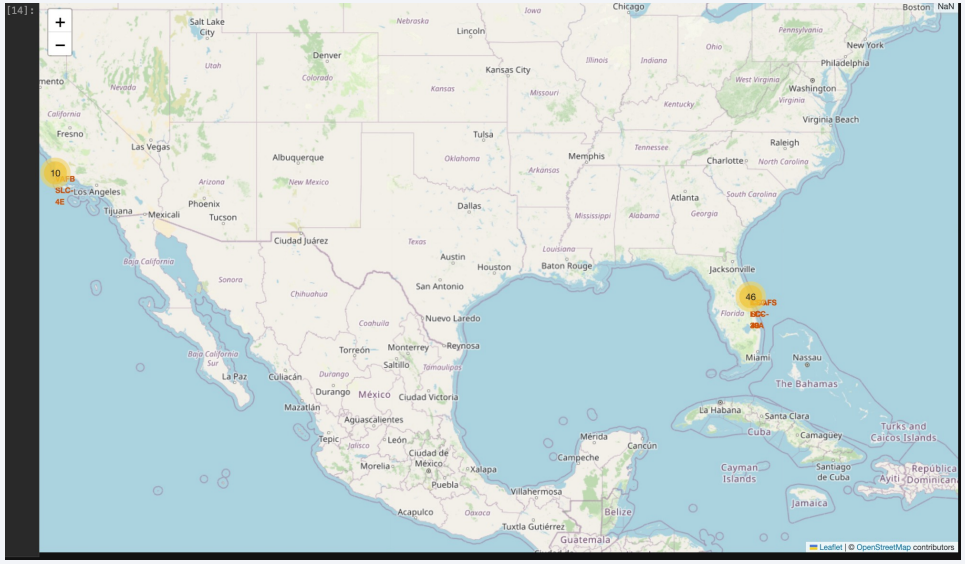




Results

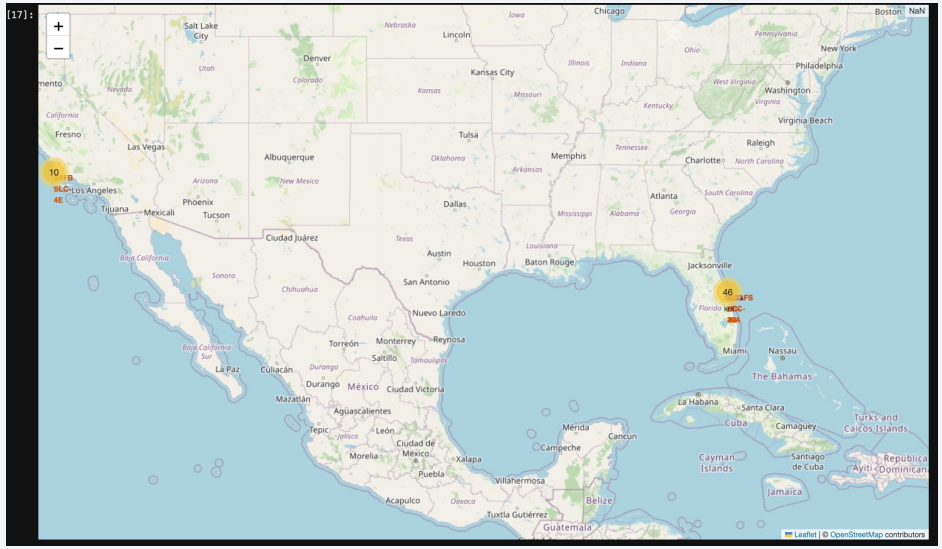
- Interactive analytics demo in screenshots

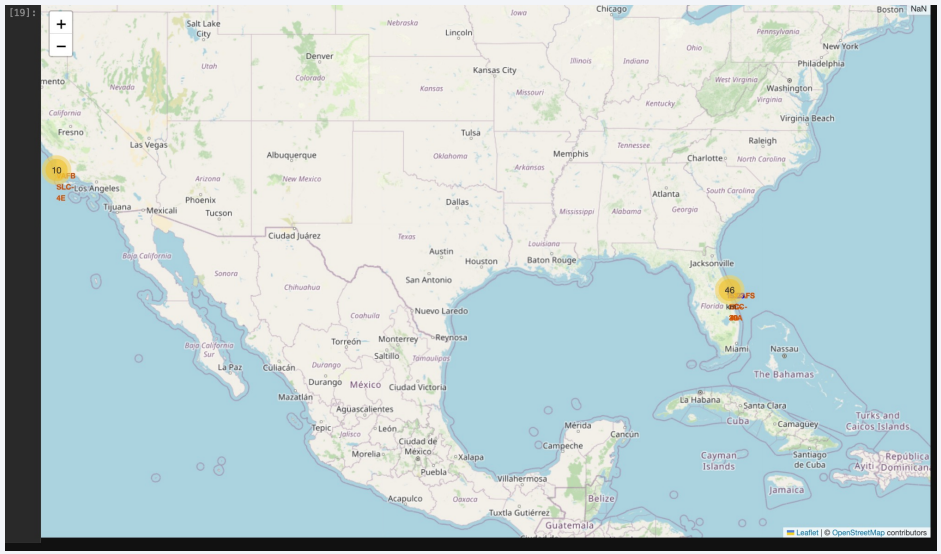




Results

- Interactive analytics demo in screenshots





Results

- Predictive analysis results

TASK 1

Create a NumPy array from the column `Class` in `data`, by applying the method `to_numpy()` then assign it to the variable `Y`,make sure the output is a Pandas series (only one bracket df['name of column']).

In [9]:
```
Y = pd.Series(data['Class'].to_numpy())
Y.head()
```

Out[9]:
```
0    0
1    0
2    0
3    0
4    0
dtype: int64
```

TASK 2

Standardize the data in `X` then reassign it to the variable `X` using the transform provided below.

In [11]:
```
# students get this
transform = preprocessing.StandardScaler()
X = transform.fit(X).transform(X)
X
```

Out[11]:
```
array([[-1.71291154e+00, -5.29526321e-17, -6.53912840e-01, ...,
        -8.35531692e-01,  1.93309133e+00, -1.93309133e+00],
       [-1.67441914e+00, -1.19523159e+00, -6.53912840e-01, ...,
        -8.35531692e-01,  1.93309133e+00, -1.93309133e+00],
       [-1.63592675e+00, -1.16267307e+00, -6.53912840e-01, ...,
        -8.35531692e-01,  1.93309133e+00, -1.93309133e+00],
       ...,
       [ 1.63592675e+00,  1.99100483e+00,  3.49060516e+00, ...,
         1.19684269e+00, -5.17306132e-01,  5.17306132e-01],
       [ 1.67441914e+00,  1.99100483e+00,  1.00389436e+00, ...,
         1.19684269e+00, -5.17306132e-01,  5.17306132e-01],
       [ 1.71291154e+00, -5.19213966e-01, -6.53912840e-01, ...,
        -8.35531692e-01, -5.17306132e-01,  5.17306132e-01]])
```

Results

- Predictive analysis results

TASK 3

Use the function train_test_split to split the data X and Y into training and test data. Set the parameter test_size to 0.2 and random_state to 2. The training data and test data should be assigned to the following labels.

`X_train, X_test, Y_train, Y_test`

In [13]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
```

we can see we only have 18 test samples.

In [14]:
```
Y_test.shape
```

Out[14]:
```
(18,)
```

TASK 4

Create a logistic regression object then create a GridSearchCV object `logreg_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

In [15]:
```
parameters ={'C':[0.01,0.1,1],
             'penalty':['l2'],
             'solver':['lbfgs']}
```

In [17]:
```
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)
```

Out[17]:
```
GridSearchCV(cv=10, estimator=LogisticRegression(),
             param_grid={'C': [0.01, 0.1, 1], 'penalty': ['l2'],
                         'solver': ['lbfgs']})
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

We output the `GridSearchCV` object for logistic regression. We display the best parameters using the data attribute `best_params_` and the accuracy on the validation data using the data attribute `best_score_`.

In [18]:
```
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
```

Results

- Predictive analysis results

TASK 5

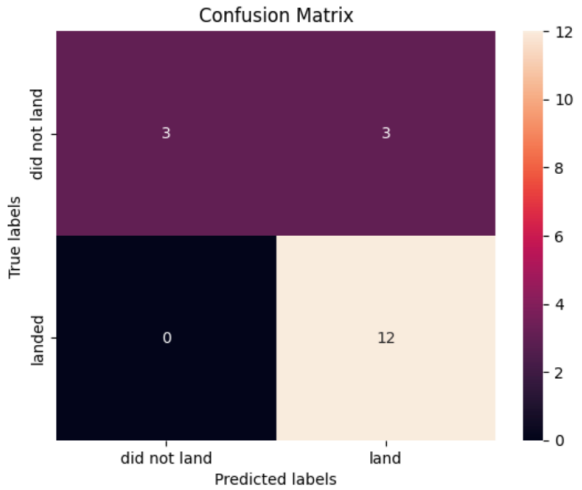
Calculate the accuracy on the test data using the method `score`:

In [19]:
```
logreg_accuracy = logreg_cv.score(X_test, Y_test)
logreg_accuracy
```

Out[19]:
```
0.8333333333333334
```

Lets look at the confusion matrix:

In [20]:
```
yhat=logreg_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



TASK 6

Create a support vector machine object then create a `GridSearchCV` object `svm_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

In [21]:
```
parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
              'C': np.logspace(-3, 3, 5),
              'gamma':np.logspace(-3, 3, 5)}
svm = SVC()
```

In [22]:
```
svm_cv = GridSearchCV(svm, parameters, cv=10)
svm_cv.fit(X_train, Y_train)
```

Out[22]:
```
GridSearchCV(cv=10, estimator=SVC(),
             param_grid={'C': array([1.00000000e-03, 3.16227766e-02, 1.00000000e+00, 3.16227766e+01,
       1.00000000e+03]),
                         'gamma': array([1.00000000e-03, 3.16227766e-02, 1.00000000e+00, 3.16227766e+0
1,
       1.00000000e+03]),
                         'kernel': ('linear', 'rbf', 'poly', 'rbf', 'sigmoid')})
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [23]:
```
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
```

Results

- Predictive analysis results

TASK 7

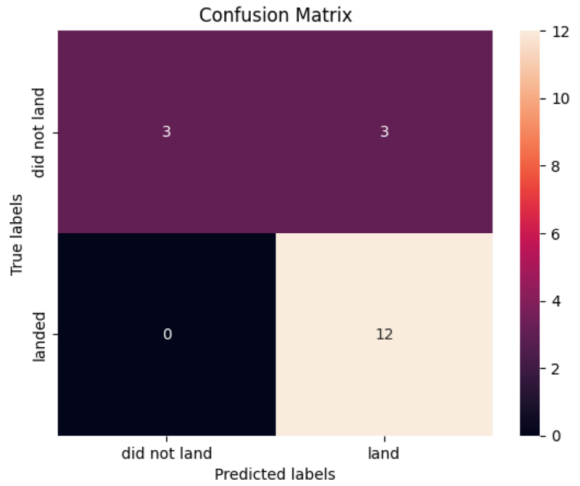
Calculate the accuracy on the test data using the method `score`:

In [24]:
```
svm_accuracy = svm_cv.score(X_test, Y_test)
svm_accuracy
```

Out[24]: 0.8333333333333334

We can plot the confusion matrix

In [25]:
```
yhat=svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Results

- Predictive analysis results

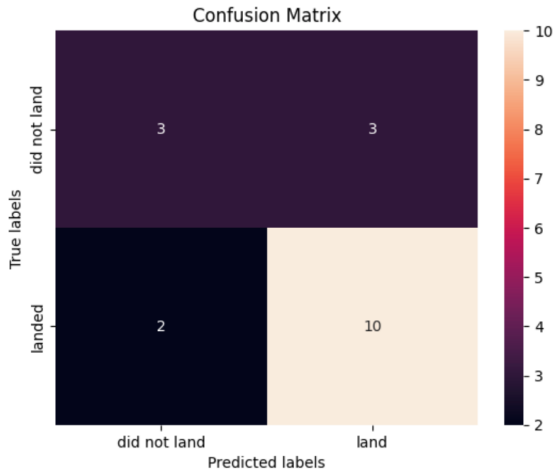
TASK 9

Calculate the accuracy of tree_cv on the test data using the method `score`:

In [ ]:
```
tree_accuracy = tree_cv.score(X_test, Y_test)
tree_accuracy
```

We can plot the confusion matrix

In [29]:
```
yhat = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



TASK 10

Create a k nearest neighbors object then create a `GridSearchCV` object `knn_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

In [30]:
```
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()
```

In [31]:
```
knn_cv = GridSearchCV(KNN, parameters, cv=10)
knn_cv.fit(X_train, Y_train)
```

Out[31]:
```
GridSearchCV(cv=10, estimator=KNeighborsClassifier(),
             param_grid={'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
                         'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                         'p': [1, 2]})
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [33]:
```
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
```

Results

- Predictive analysis results

TASK 11

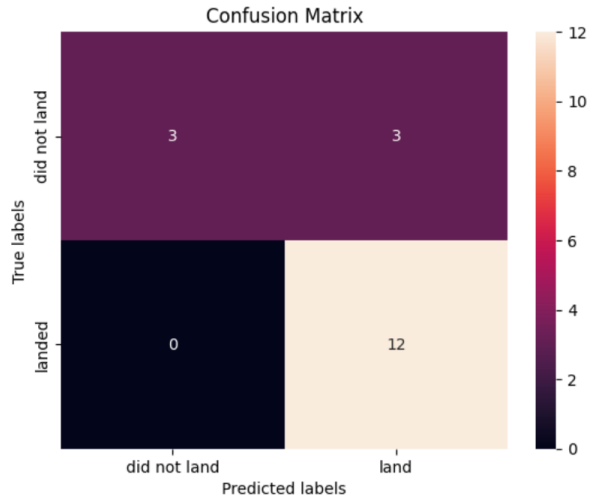
Calculate the accuracy of knn_cv on the test data using the method `score`:

In [34]:
```
knn_accuracy = knn_cv.score(X_test, Y_test)
knn_accuracy
```

Out[34]:
```
0.8333333333333334
```

We can plot the confusion matrix

In [35]:
```
yhat = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



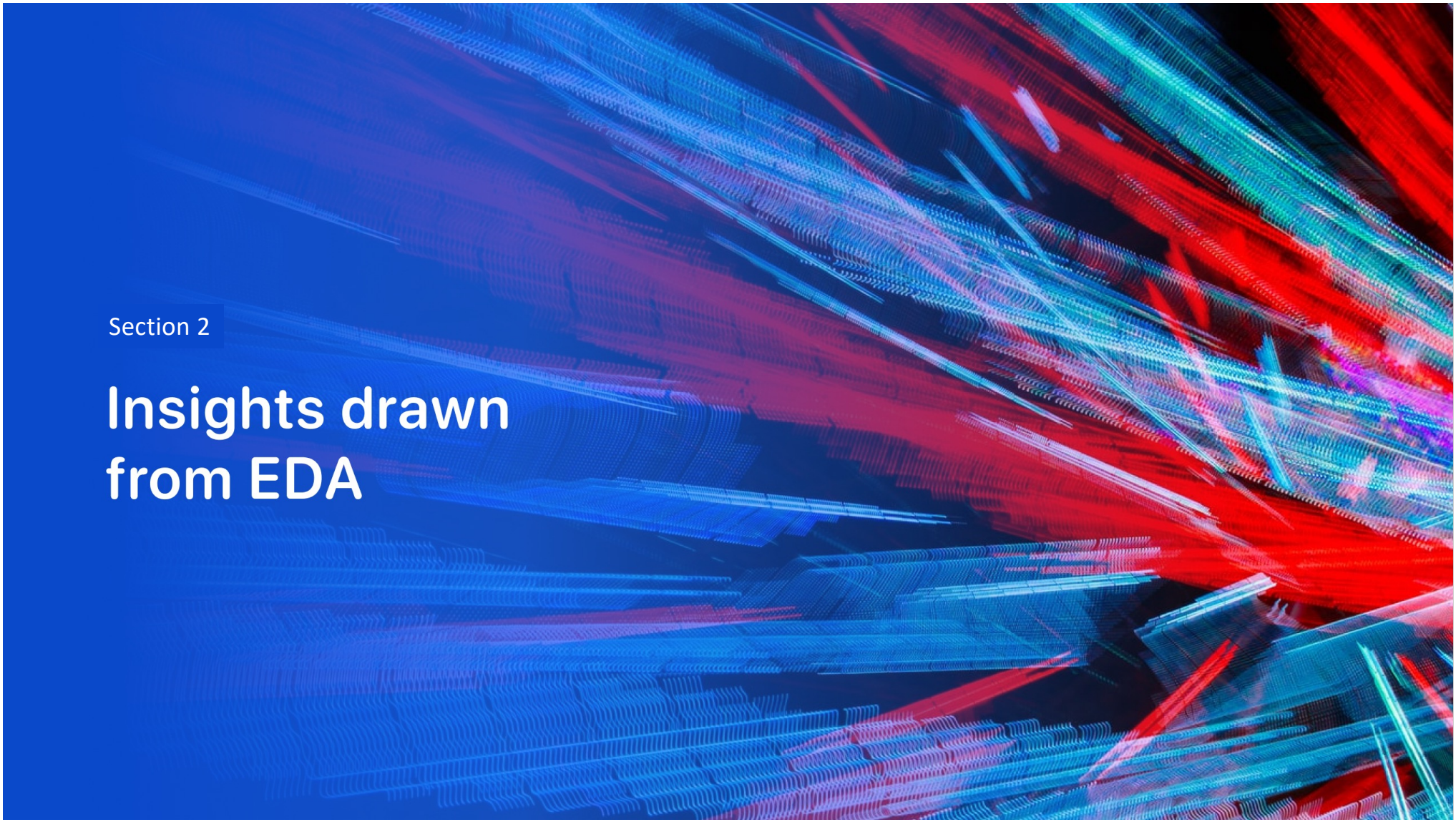
TASK 12

Find the method performs best:

In [36]:
```
from sklearn.metrics import jaccard_score, f1_score

# Examining the scores from Test sets
jaccard_scores = [
                  jaccard_score(Y_test, logreg_yhat, average='binary'),
                  jaccard_score(Y_test, svm_yhat, average='binary'),
                  jaccard_score(Y_test, tree_yhat, average='binary'),
                  jaccard_score(Y_test, knn_yhat, average='binary'),
                 ]

f1_scores = [
             f1_score(Y_test, logreg_yhat, average='binary'),
             f1_score(Y_test, svm_yhat, average='binary'),
             f1_score(Y_test, tree_yhat, average='binary'),
             f1_score(Y_test, knn_yhat, average='binary'),
            ]

accuracy = [logreg_accuracy, svm_accuracy, tree_accuracy, knn_accuracy]

scores = pd.DataFrame(np.array([jaccard_scores, f1_scores, accuracy]), index=['Jaccard_Score', 'F1_Score', 'Accur
scores
```

Section 2

Insights drawn from EDA



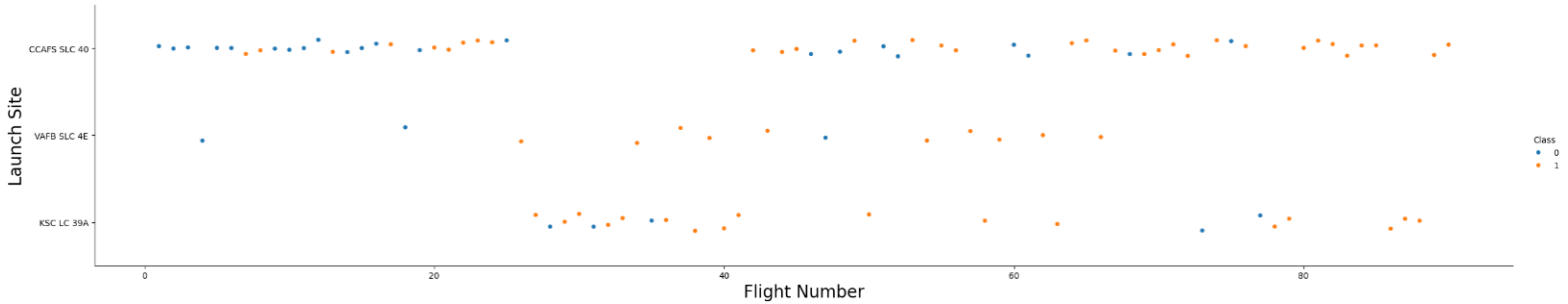
Flight Number vs. Launch Site

TASK 1: Visualize the relationship between Flight Number and Launch Site

Use the function `catplot` to plot `FlightNumber` vs `LaunchSite`, set the parameter `x` parameter to `FlightNumber`,set the `y` to `Launch Site` and set the parameter `hue` to `'class'`

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
sns.catplot(x='FlightNumber', y='LaunchSite', hue='Class', data=df, aspect=5)
plt.xlabel('Flight Number', fontsize=20)
plt.ylabel('Launch Site', fontsize=20)
plt.show()
```

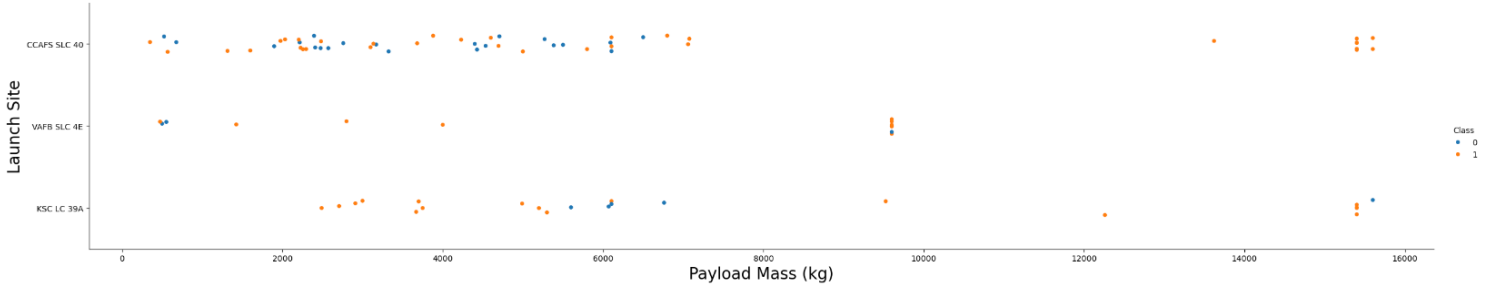


Payload vs. Launch Site

TASK 2: Visualize the relationship between Payload Mass and Launch Site

We also want to observe if there is any relationship between launch sites and their payload mass.

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
sns.catplot(x='PayloadMass', y='LaunchSite', hue='Class', data=df, aspect = 5)
plt.xlabel('Payload Mass (kg)',fontsize=20)
plt.ylabel('Launch Site',fontsize=20)
plt.show()
```



Now if you observe Payload Mass Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

Success Rate vs. Orbit Type

- Show a bar chart for the success rate of each orbit type
- Show the screenshot of the scatter plot with explanations

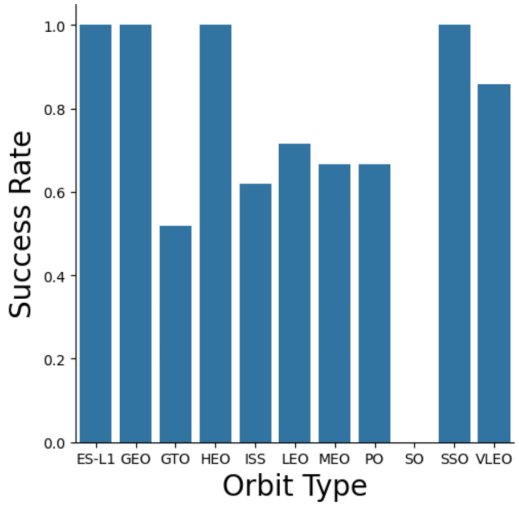
TASK 3: Visualize the relationship between success rate of each orbit type

Next, we want to visually check if there are any relationship between success rate and orbit type.

Let's create a `bar chart` for the sucess rate of each orbit

In [7]:

```
# HINT use groupby method on Orbit column and get the mean of Class column
sns.catplot(x= 'Orbit', y = 'Class', data = df.groupby('Orbit')['Class'].mean().reset_index(), kind = 'bar')
plt.xlabel('Orbit Type',fontsize=20)
plt.ylabel('Success Rate',fontsize=20)
plt.show()
```

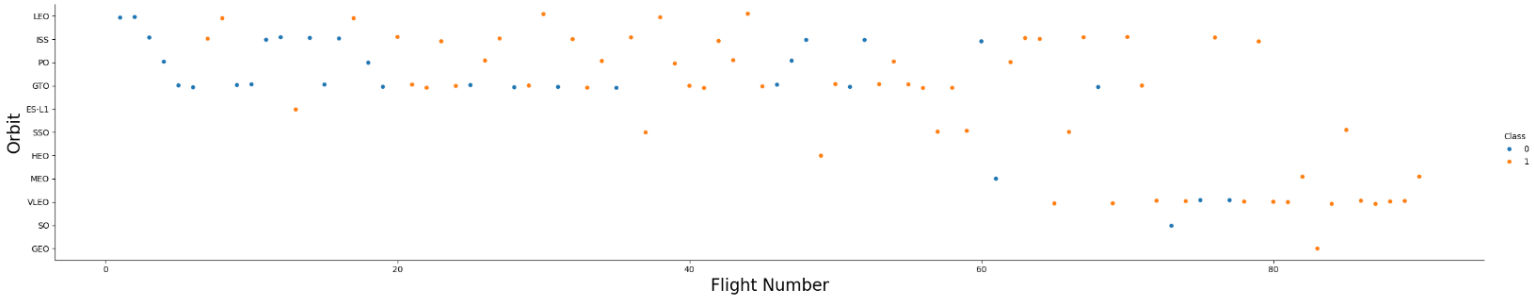


Flight Number vs. Orbit Type

TASK 4: Visualize the relationship between FlightNumber and Orbit type

For each orbit, we want to see if there is any relationship between FlightNumber and Orbit type.

In [8]:
```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class v
sns.catplot(x = 'FlightNumber', y = 'Orbit', hue = 'Class', data = df, aspect = 5)
plt.xlabel('Flight Number', fontsize = 20)
plt.ylabel('Orbit', fontsize = 20)
plt.show()
```



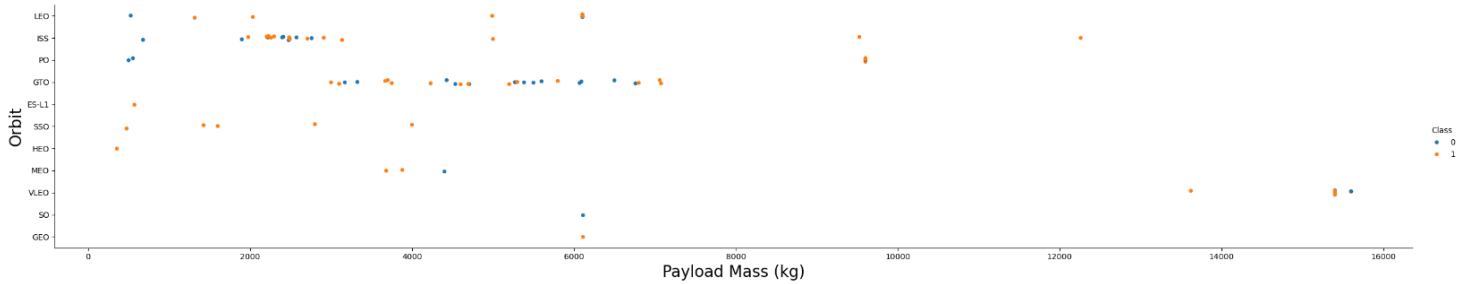
You can observe that in the LEO orbit, success seems to be related to the number of flights. Conversely, in the GTO orbit, there appears to be no relationship between flight number and success.

Payload vs. Orbit Type

TASK 5: Visualize the relationship between Payload Mass and Orbit type

Similarly, we can plot the Payload Mass vs. Orbit scatter point charts to reveal the relationship between Payload Mass and Orbit type

In [9]:
```
# 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 v
sns.catplot(x = 'PayloadMass', y = 'Orbit', hue = 'Class', data = df, aspect = 5)
plt.xlabel('Payload Mass (kg)', fontsize = 20)
plt.ylabel('Orbit', fontsize = 20)
plt.show()
```

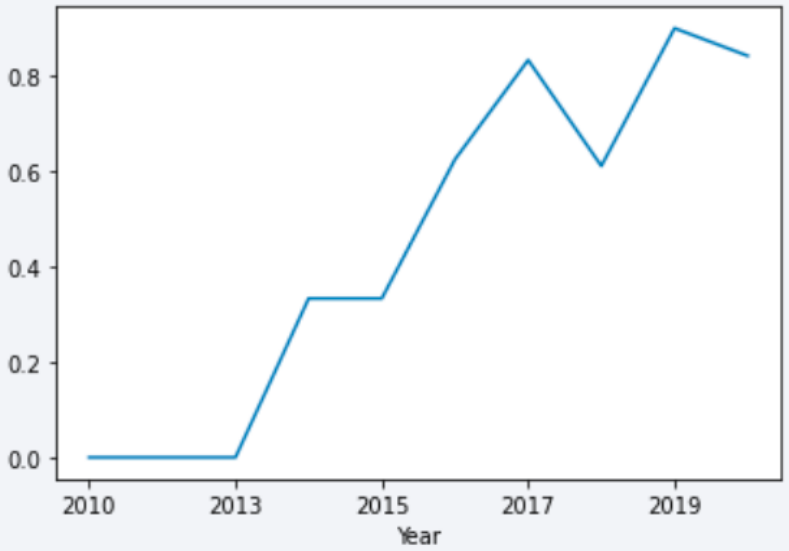


With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

However, for GTO, it's difficult to distinguish between successful and unsuccessful landings as both outcomes are present.

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

Find the names of the unique launch sites

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

Launch Site Names Begin with 'CCA'

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

Date	Time UTC	Booster Version	Launch Site	Payload	Payload Mass kg	Orbit	Customer	Mission Outcome	Landing Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	**CCA**FS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	**CCA**FS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	07:44:00	F9 v1.0 B0005	**CCA**FS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	F9 v1.0 B0006	**CCA**FS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	**CCA**FS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attemp

Total Payload Mass

- Calculate the total payload carried by boosters from NASA

111.268 kg

Average Payload Mass by F9 v1.1

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

2.928kg

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad

2015-12-22

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

- F9 FT B1021.2
- F9 FT B1031.2
- F9 FT B1022
- F9 FT B1026

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes

Mission Outcome	Occurrences
Success	99
Success (payload status unclear)	1
Failure (in flight)	1

Boosters Carried Maximum Payload

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

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

2015 Launch Records

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

Booster Version	Launch Site
F9 v1.1 B1012	CCAFS LC-40
F9 v1.1 B1015	CCAFS LC-40

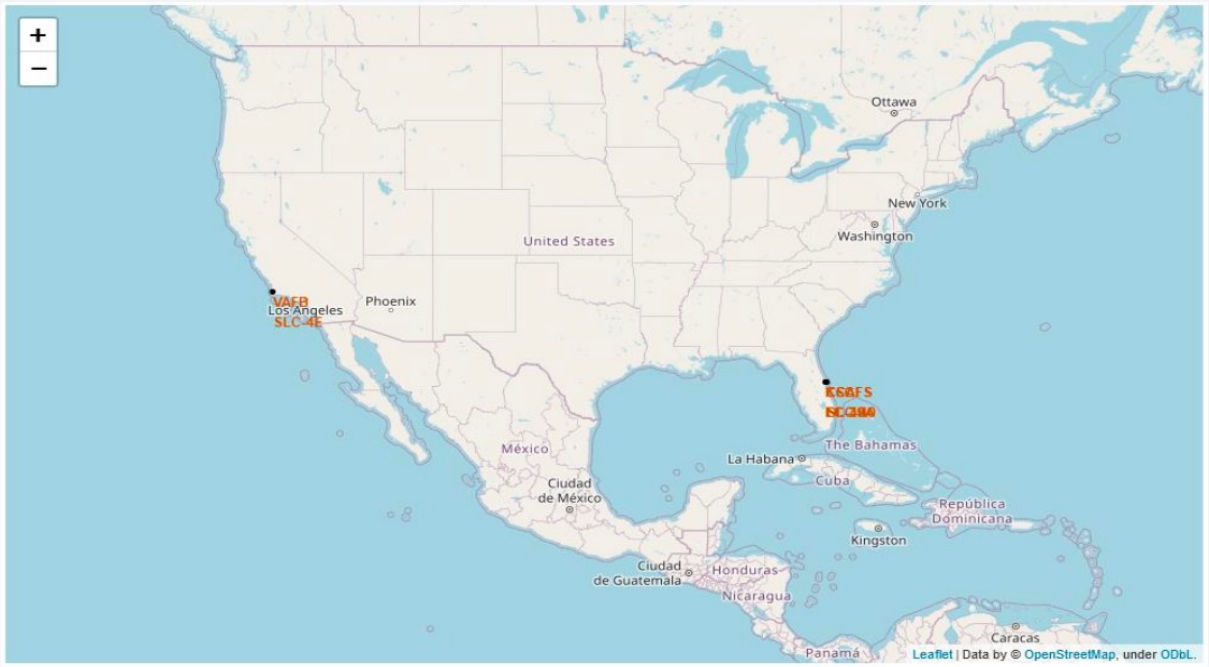
Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

- 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

Landing Outcome	Occurrences
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

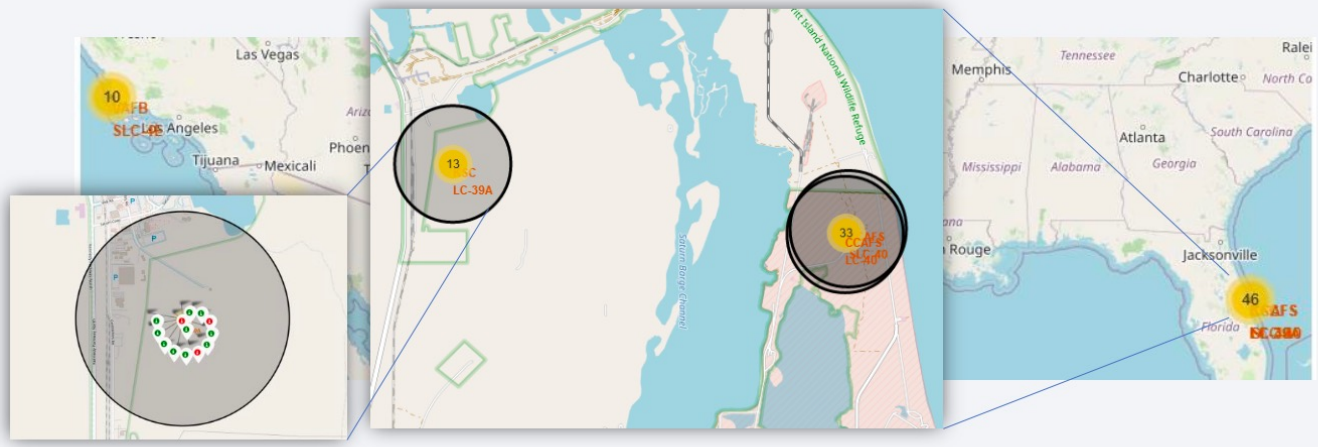
<Folium Map Screenshot 1>



- Launch sites are near sea, probably by safety, but not too far from roads and railroads.

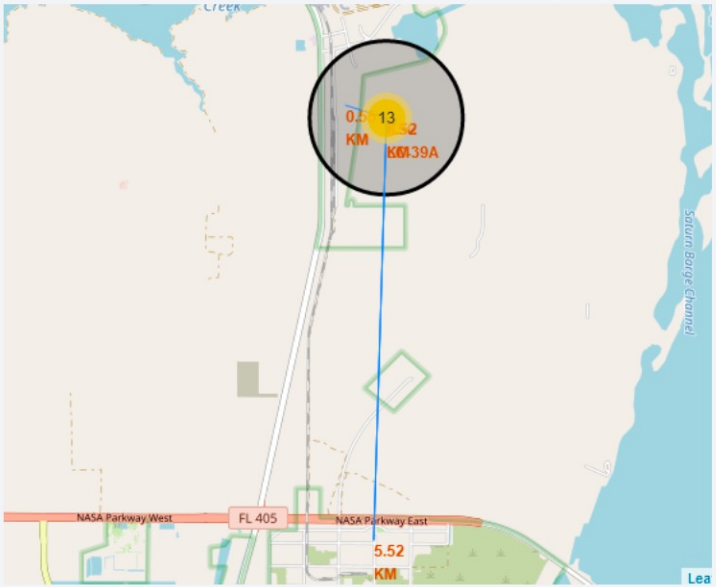
<Folium Map Screenshot 2>

- Example of KSC LC-39A launch site launch outcomes

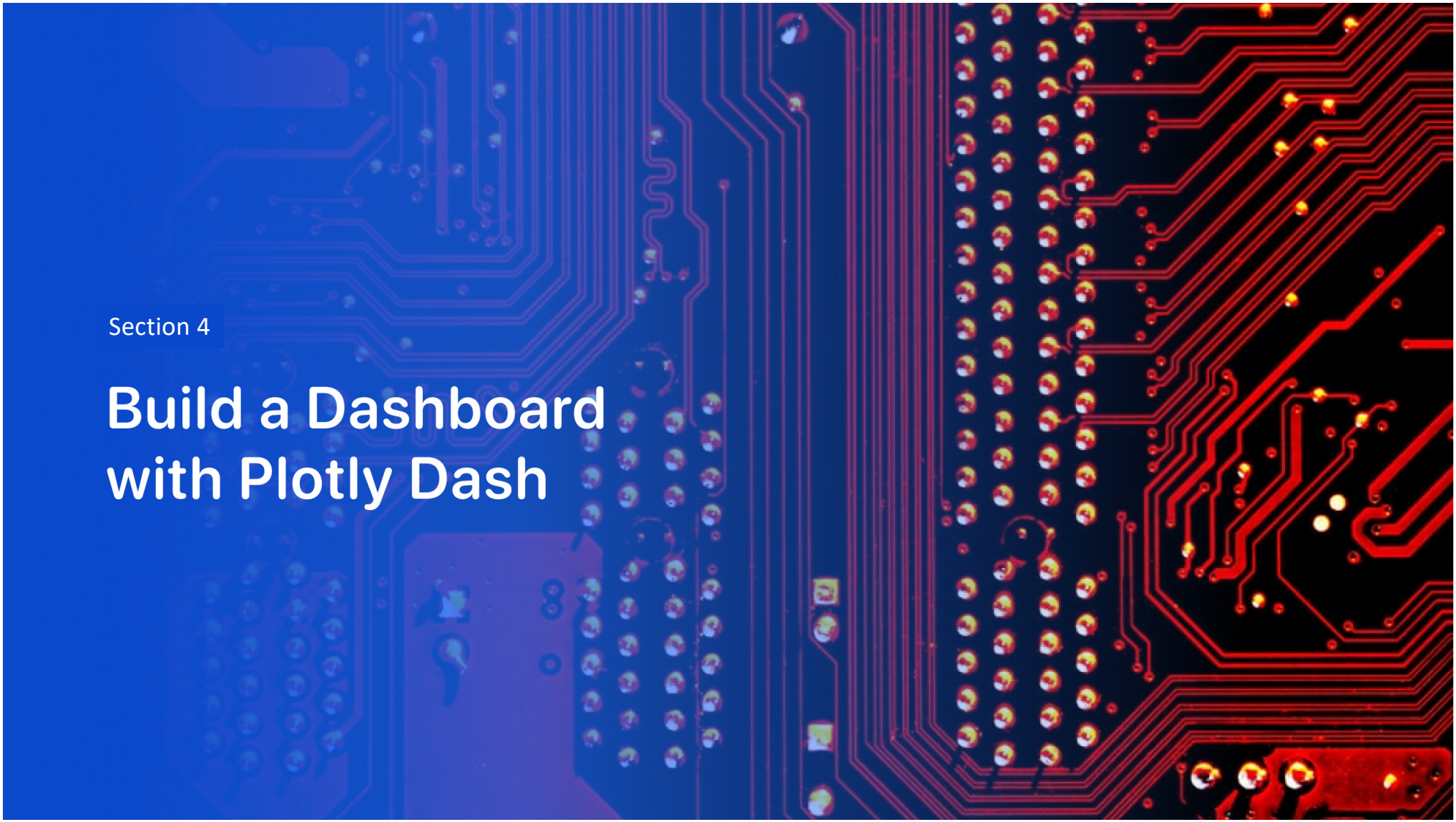


- Green markers indicate successful and red ones indicate failure.

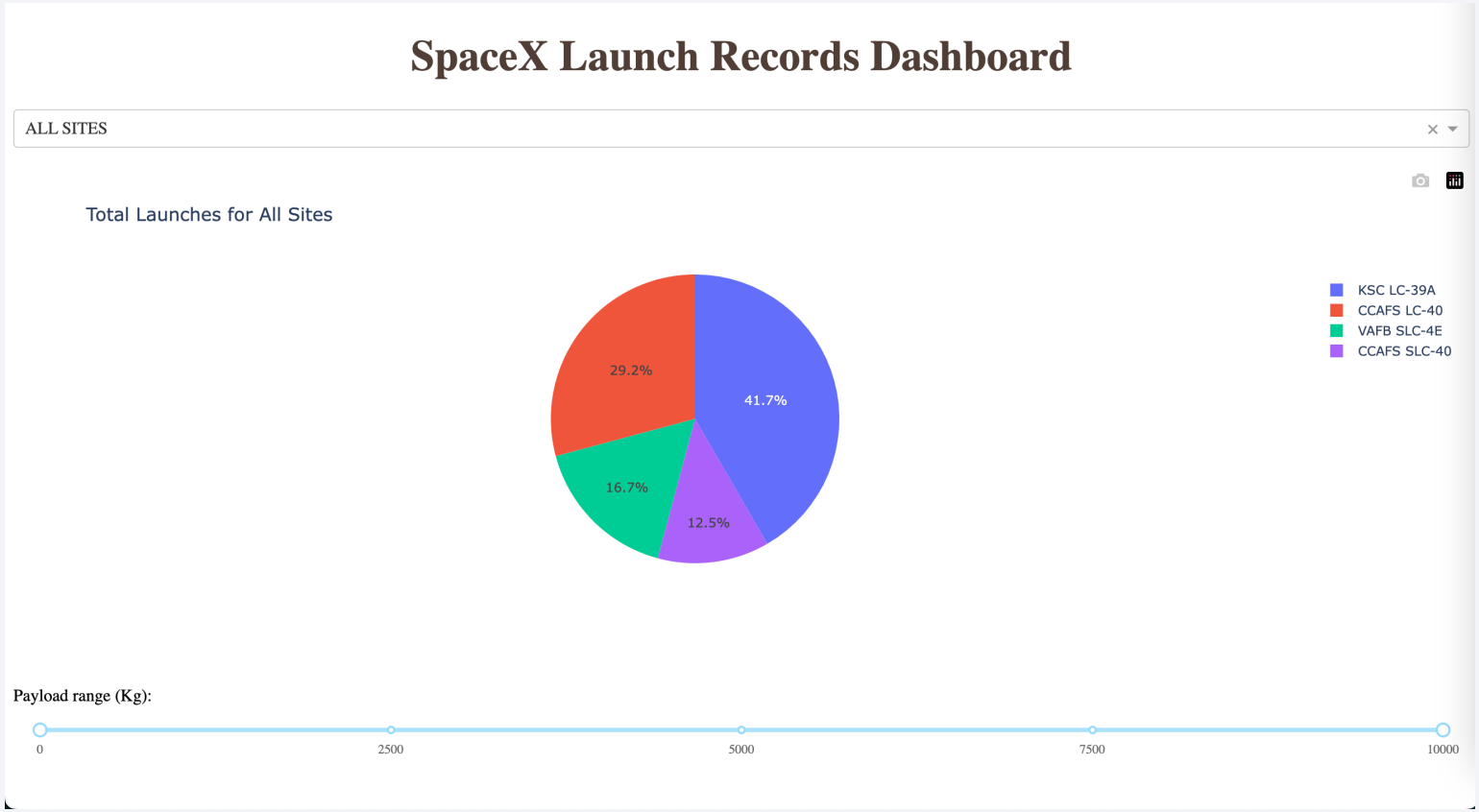
<Folium Map Screenshot 3>



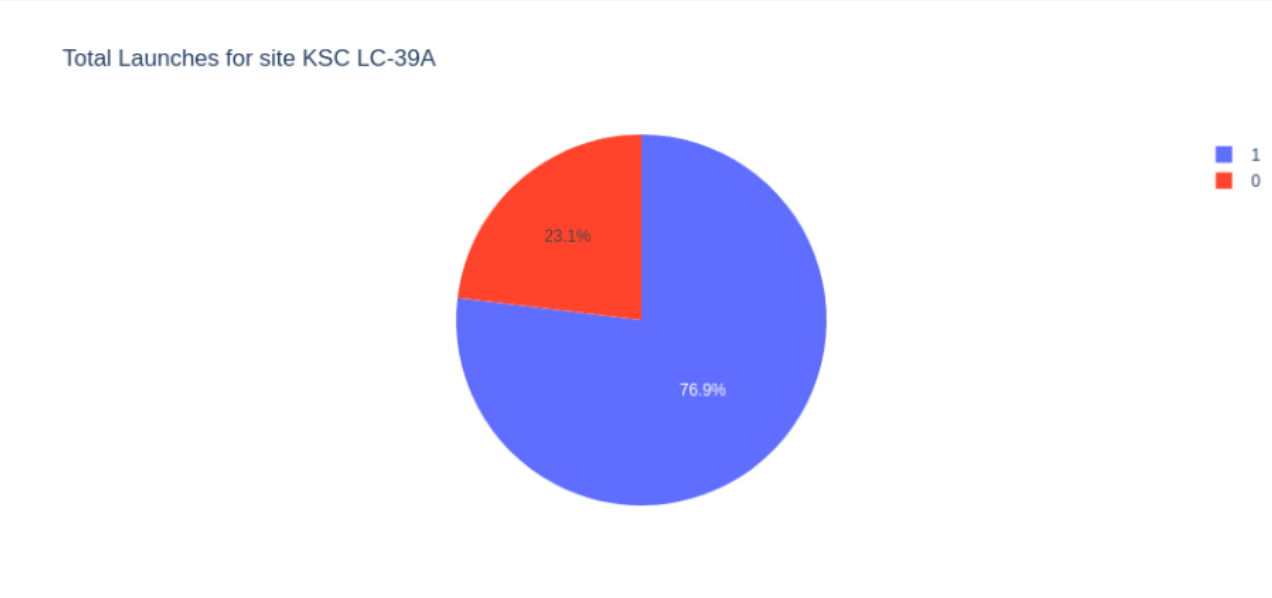
- Launch site KSC LC-39A has good logistics aspects, being near railroad and road and relatively far from inhabited areas.


Section 4
Build a Dashboard with Plotly Dash

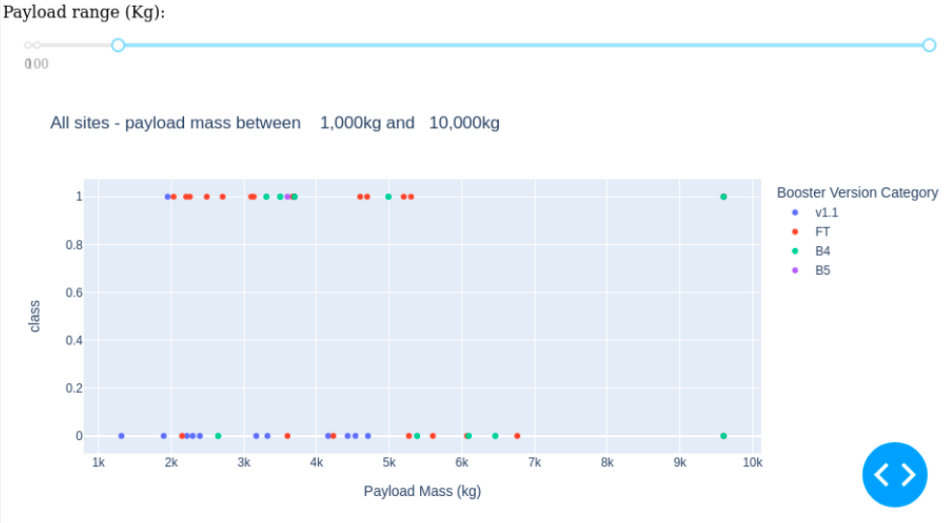
<Dashboard Screenshot 1>



<Dashboard Screenshot 2>



<Dashboard Screenshot 3>



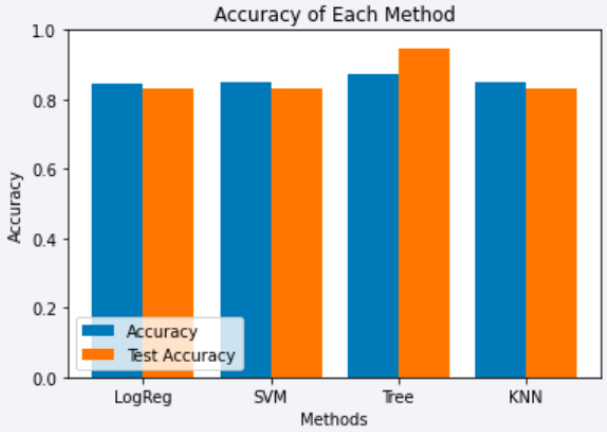
Section 5

Predictive Analysis (Classification)

Classification Accuracy

- Visualize the built model accuracy for all built classification models, in a bar chart

- Decision Tree Classifier model has the highest classification accuracy



Confusion Matrix

- Show the confusion matrix of the Decision Tree model

Conclusions

- Various data sources were analysed, leading to refined conclusions throughout the process.
- The optimal launch site identified is KSC LC-39A.
- A Decision Tree Classifier can be utilized to predict successful landings, enhancing profitability.
- Launches exceeding 7,000 kg are associated with lower risk.
- While most missions are successful, the rate of successful landings appears to improve over time due to advancements in processes and rocket technology.

Appendix

- All the required Python code snippets, SQL queries, charts, Notebook outputs, or data sets have been uploaded on the GitHub Link provided here.

https://github.com/riyansh100/Applied-Data-Science-Capstone/tree/main

Thank you!