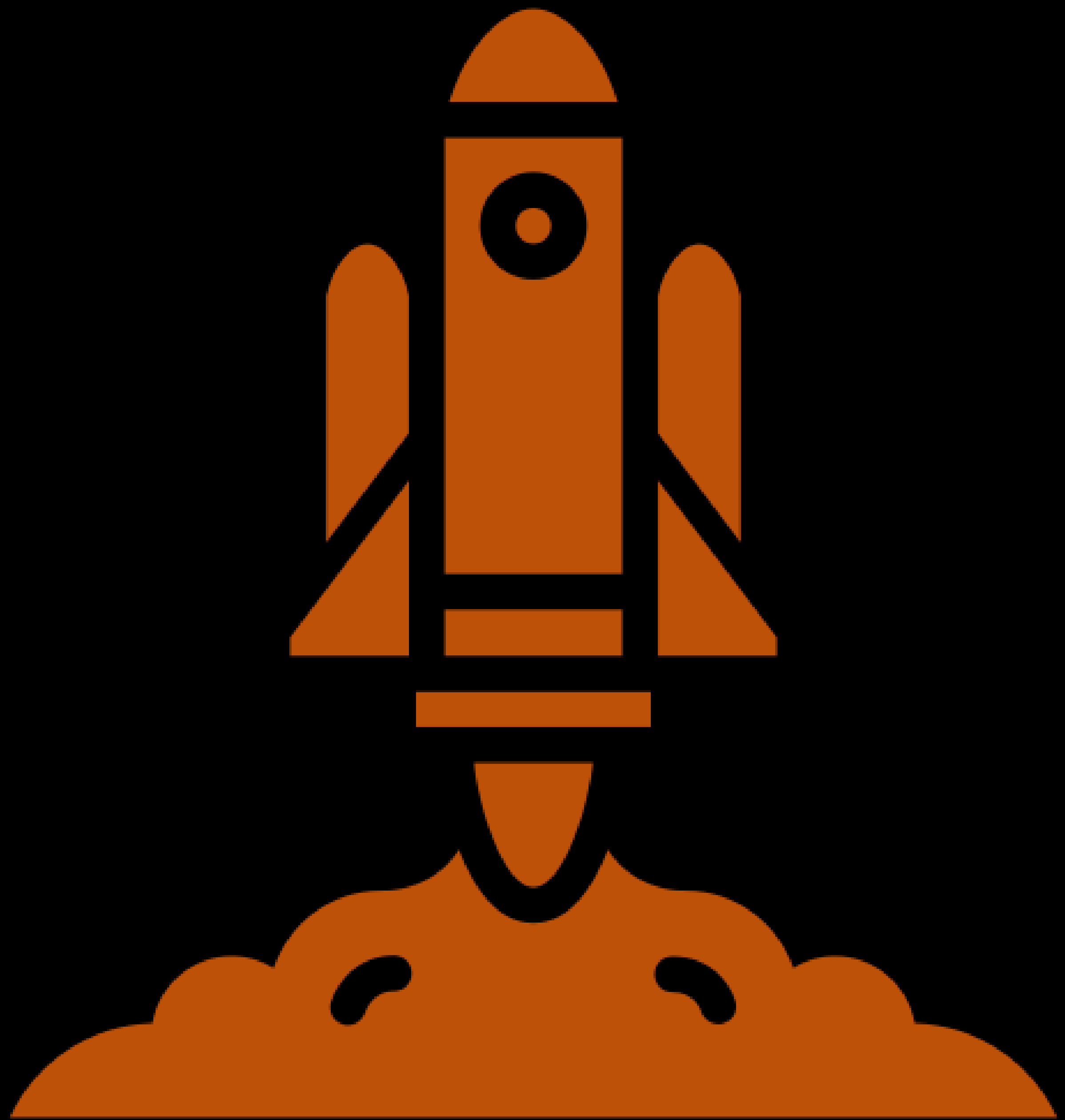
Winning Space Race with Data Science



Yi-Hua PAN

Nov. 30th, 2023

OUTLINE

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

EXECUTIVE SUMMARY

- Data Collection: from API and Web Page
- Data Wrangling
- Exploratory Data Analysis
  - Using SQL
  - Using Pandas and Matplotlib
- Interactive Visual Analytics and Dashboard
- Predictive Analysis

INTRODUCTION

In the competitive Space Launch Market, SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars whereas other providers cost upward of 165 million dollars each, much of the savings is due to the reuse of the first stage.

Determine if the First Stage will land

Determine the Cost of a Launch

In this capstone, we are going to take the role of a data scientist working for a new rocket company. Our objective is to determine the price of each launch using SpaceX Falcon9 Database. This information will be provided to the company in bidding against SpaceX for a rocket launch.

METHODOLOGY

METHODOLOGY
- Data Collection and Data Wrangling

Data Collection

- Request rocket launch data from SpaceX API with the URL
- Extract a Falcon 9 launch records HTML table from Wikipedia with *BeautifulSoup()*, parsing and converting it into a Pandas data frame

Data Wrangling

- Dealing with missing values by using *.mean()* and *.replace()* functions to replace *np.nan* value
- Perform some Exploratory Data Analysis (EDA), such as *.value_counts()* to find some patterns in the dataset.

METHODOLOGY
- EDA and Interactive Visual Analytics

Exploratory Data Analysis (EDA)

- Load the SQL extension and establish a connection with the SQLite database
- Execute SQL queries with the SQL magic, *%sql*, commands in Python
- Perform EDA using Pandas, Matplotlib and Seaborn to visualize the relationship between variables and
- Preparing Data Featuring Engineering with *get_dummies()*

Interactive Visual Analytics

- Perform interactive visual analytics using Folium with *folium.Circle()*, *folium.Maker()*, *MarkerCluster()*, *MousePosition()*
- Build a Plotly Dash application with a dropdown list and a range slider to interact with a pie chart and a scatter chart

METHODOLOGY
- Predictive Analysis

Machine Learning Prediction

- Create a machine learning pipeline to predict if the first stage will land given the data from the preceding labs
- Standardize data with *preprocessing.StandardScaler()* and *fit_transform()*
- Split train and test data with *train_test_split()*
- Find hyperparameter with *GridSearchCV()* for Logistic Regression, SVM, DecisionTree, and k nearest neighnors

RESULTS
- EDA with SQL

- There were four launch sites in the space mission (Table 1)
- The total payload mass launched by NASA (CRS) was 45596
- The average payload mass carried by booster version F9 v1.1 was 2534.67
- The date of the first successful landing outcome in ground pad was 2015-12-22
- Four booster versions have success in drone ship with payload mass greater than 4000 and less than 6000 (Table 2)

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

Table 1

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

Table 2

RESULTS
- EDA with SQL

- Total number of successful and failure mission outcomes was shown in Table 3.
- The records with failure landing outcomes in 2015 was shown in Table 4.
- The count of landing outcomes between 2010-06-064 and 2017-03-20 was shown in Table 5.

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

Table 3

Year	Month	Booster_Version	Launch_Site	Landing_Outcome
2015	01	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
2015	04	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)
2016	01	F9 v1.1 B1017	VAFB SLC-4E	Failure (drone ship)
2016	03	F9 FT B1020	CCAFS LC-40	Failure (drone ship)
2016	06	F9 FT B1024	CCAFS LC-40	Failure (drone ship)

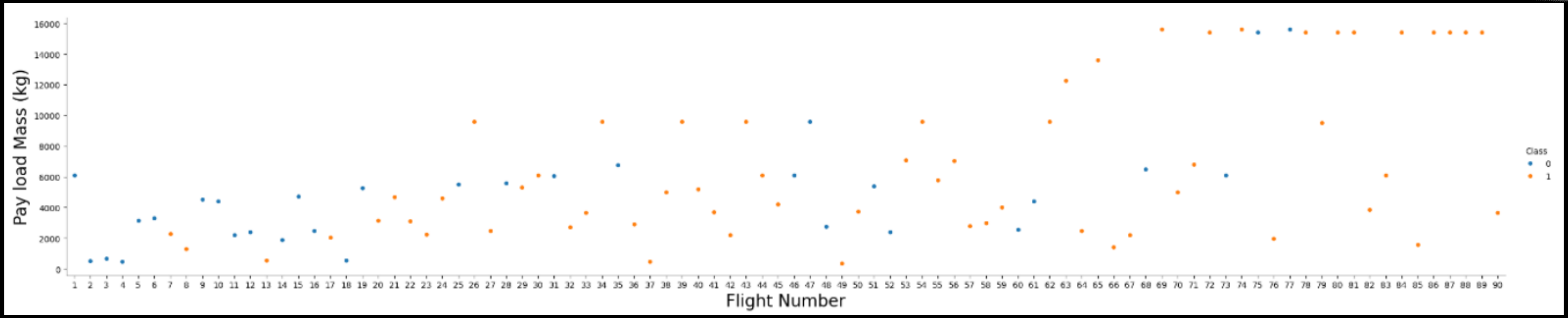
Table 4

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

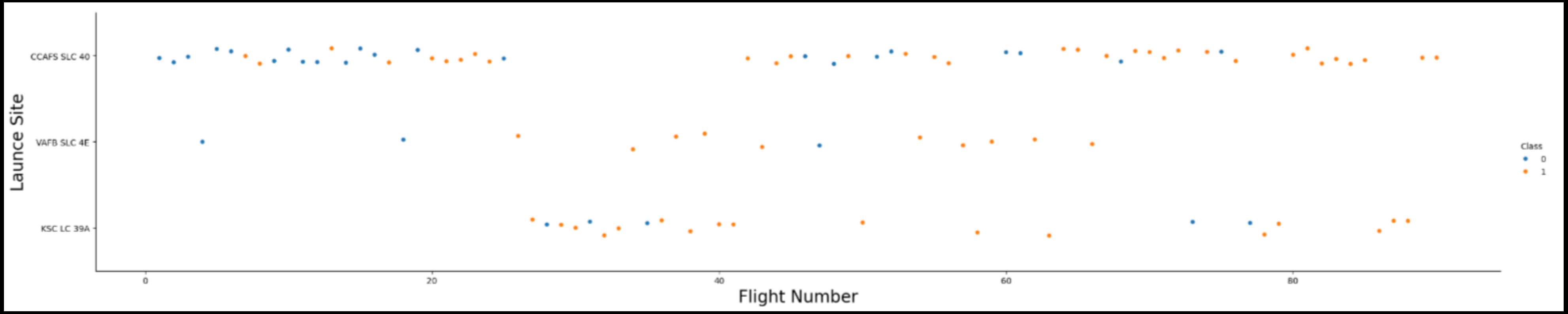
Table 5

RESULTS
- EDA with visualization

Relationship between Payload Mass and Flight Number

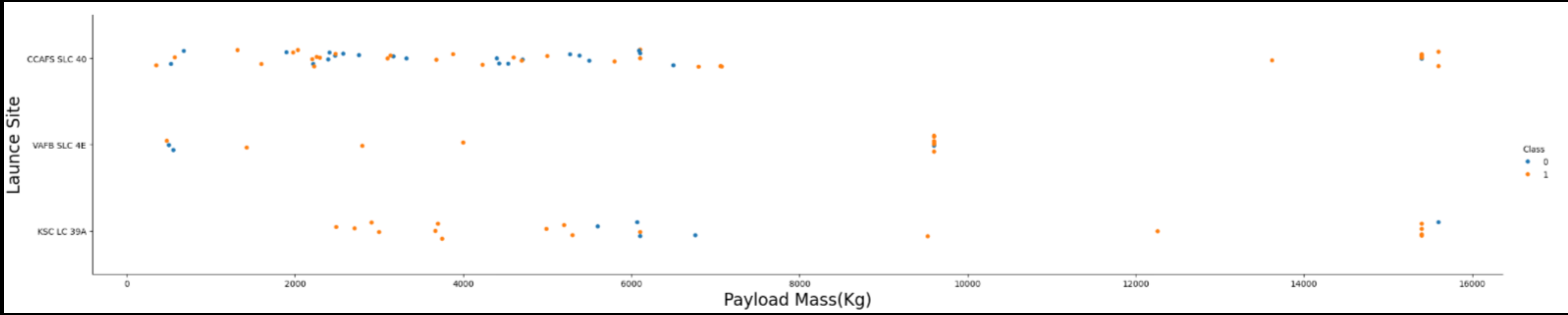


Relationship between Launch Sites and Flight Number

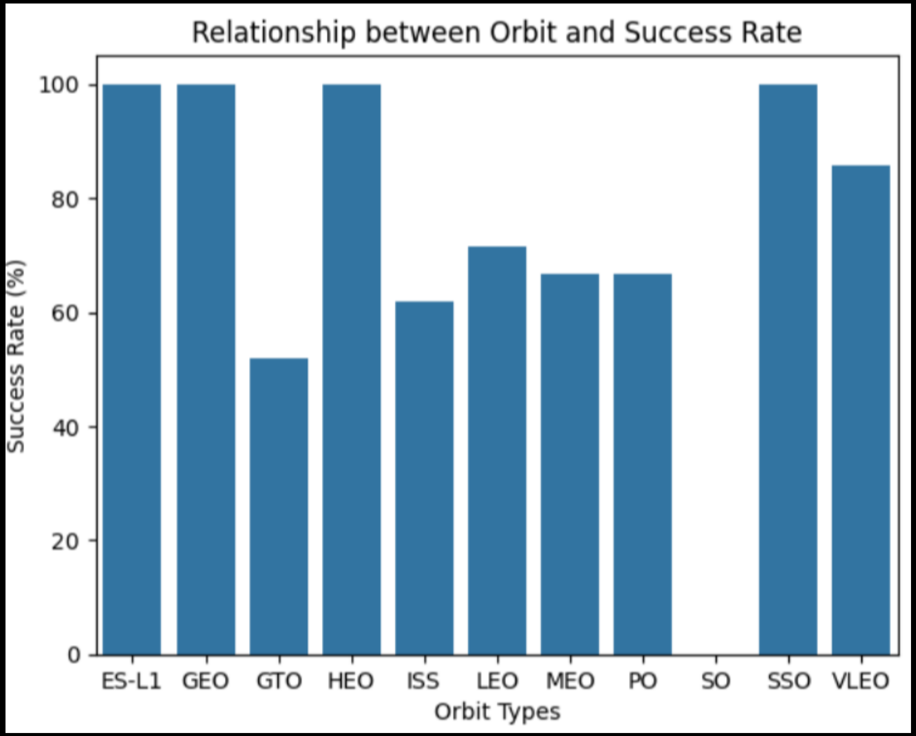


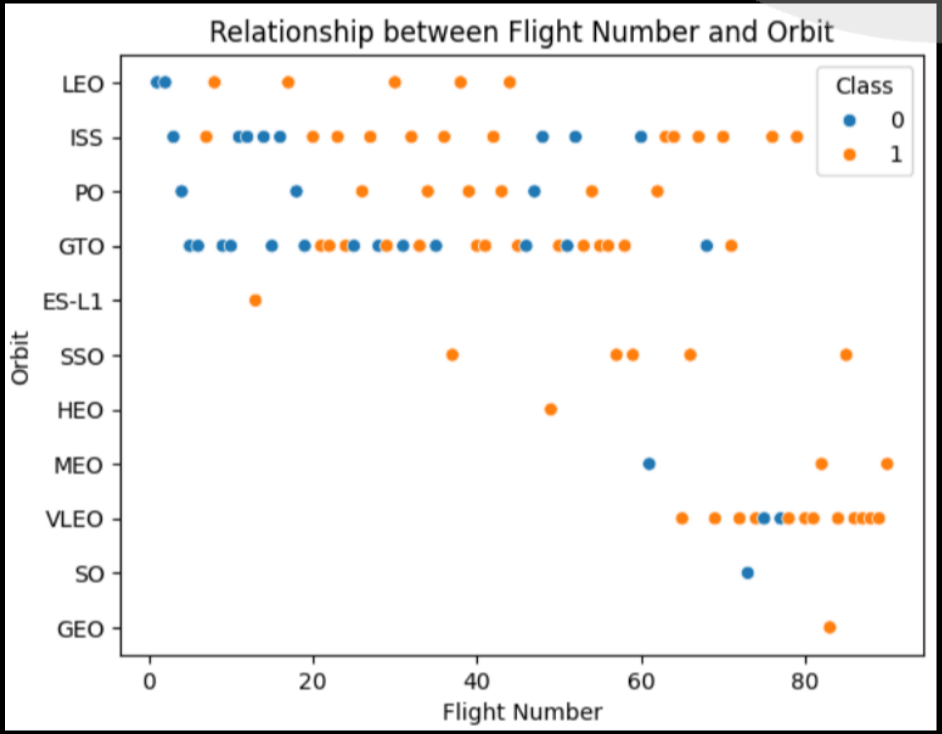
RESULTS
- EDA with visualization

Relationship between Payload Mass and Launch Sites

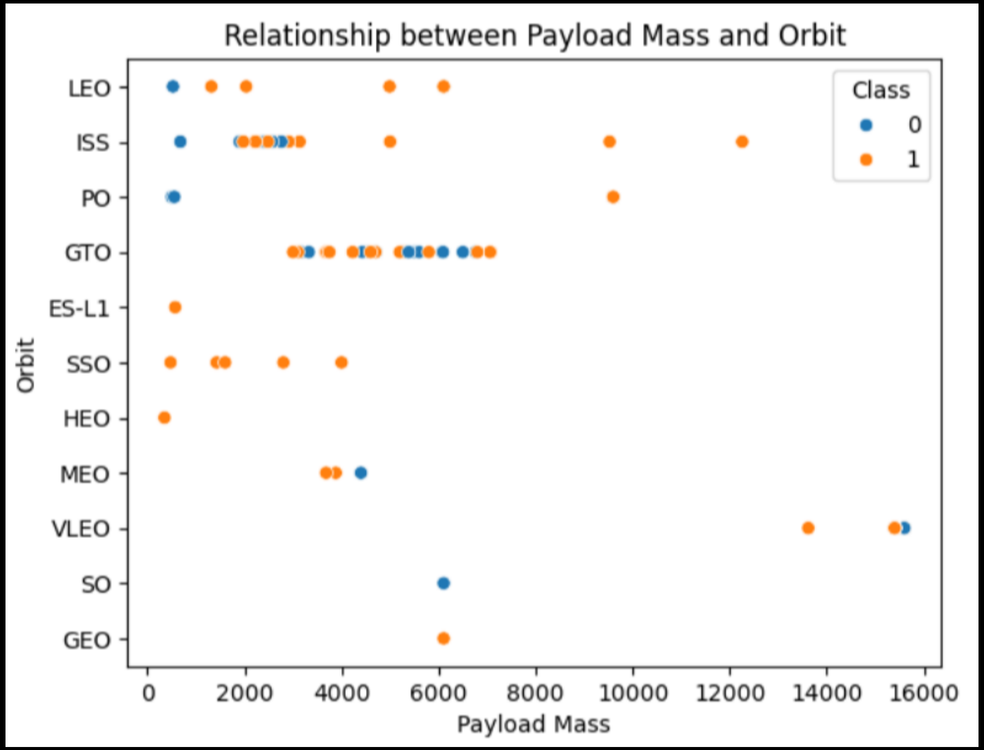


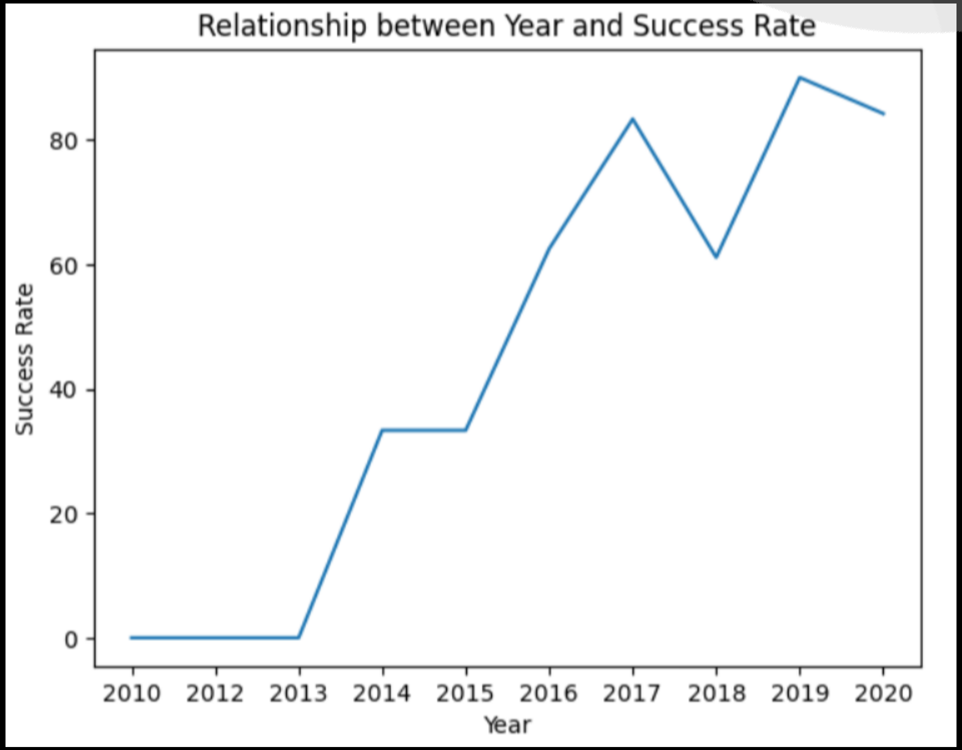
RESULTS
- EDA with visualization





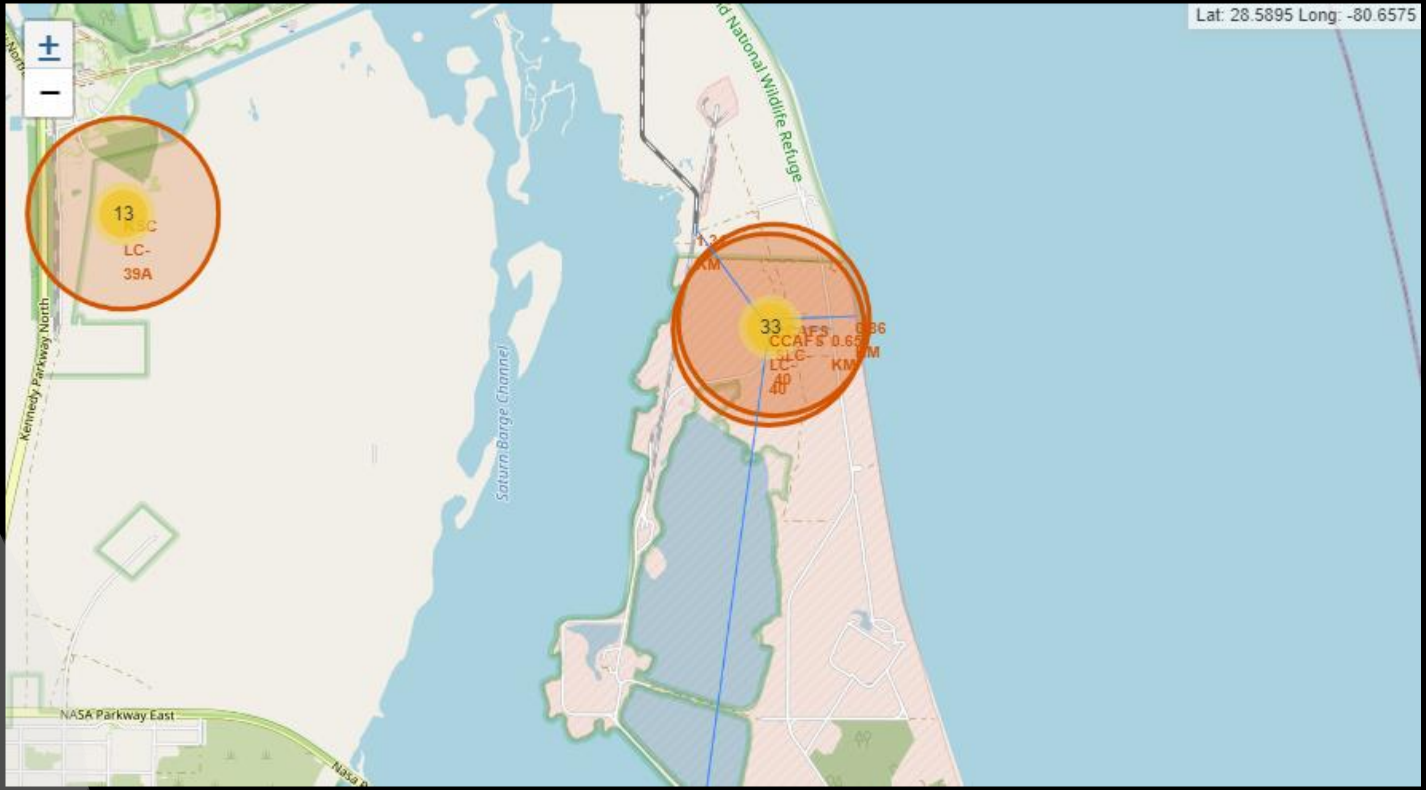
RESULTS
- EDA with visualization





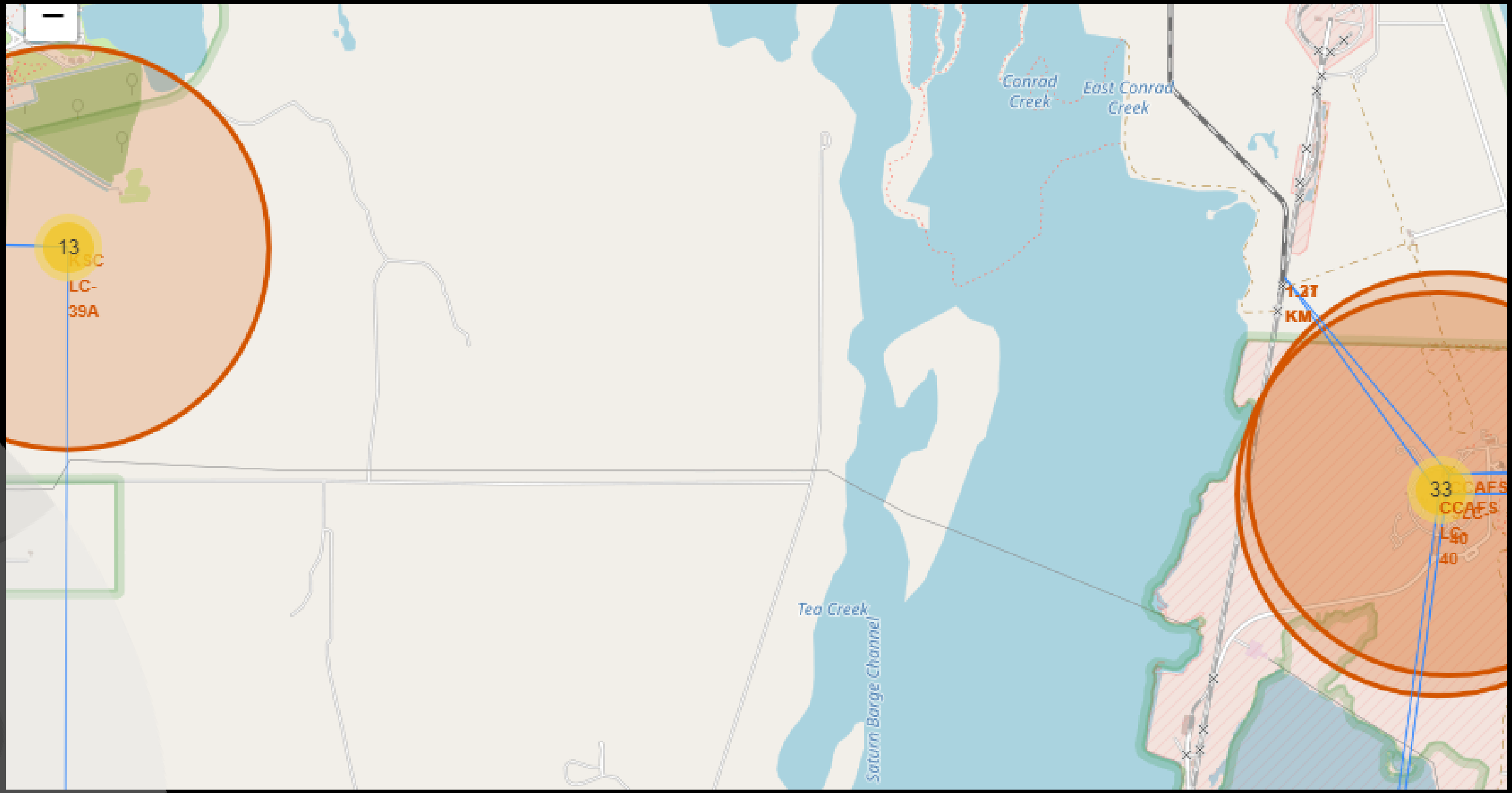
RESULTS
- Interactive Map with Folium

Distance between CCAFS_SLC_40 and the nearest city, railway and highway



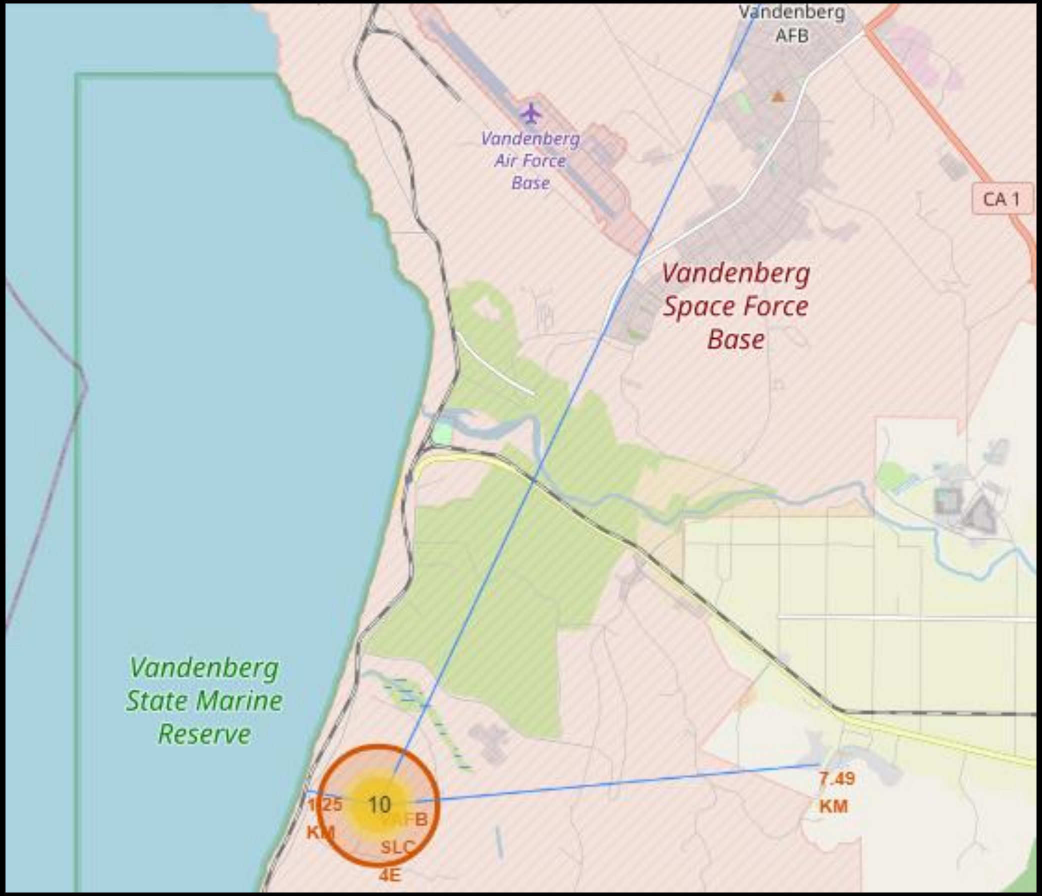
RESULTS
- Interactive Map with Folium

Distance between KSC_LC_39A and the nearest city, railway and highway



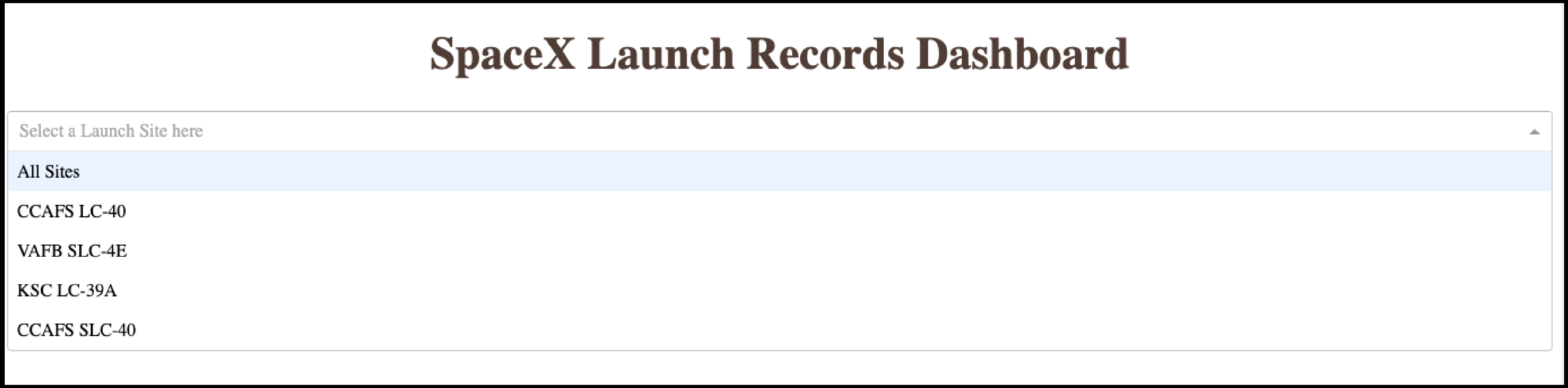
RESULTS
- Interactive Map with Folium

Distance between VAFB_SLC_4E and the nearest city, railway and highway



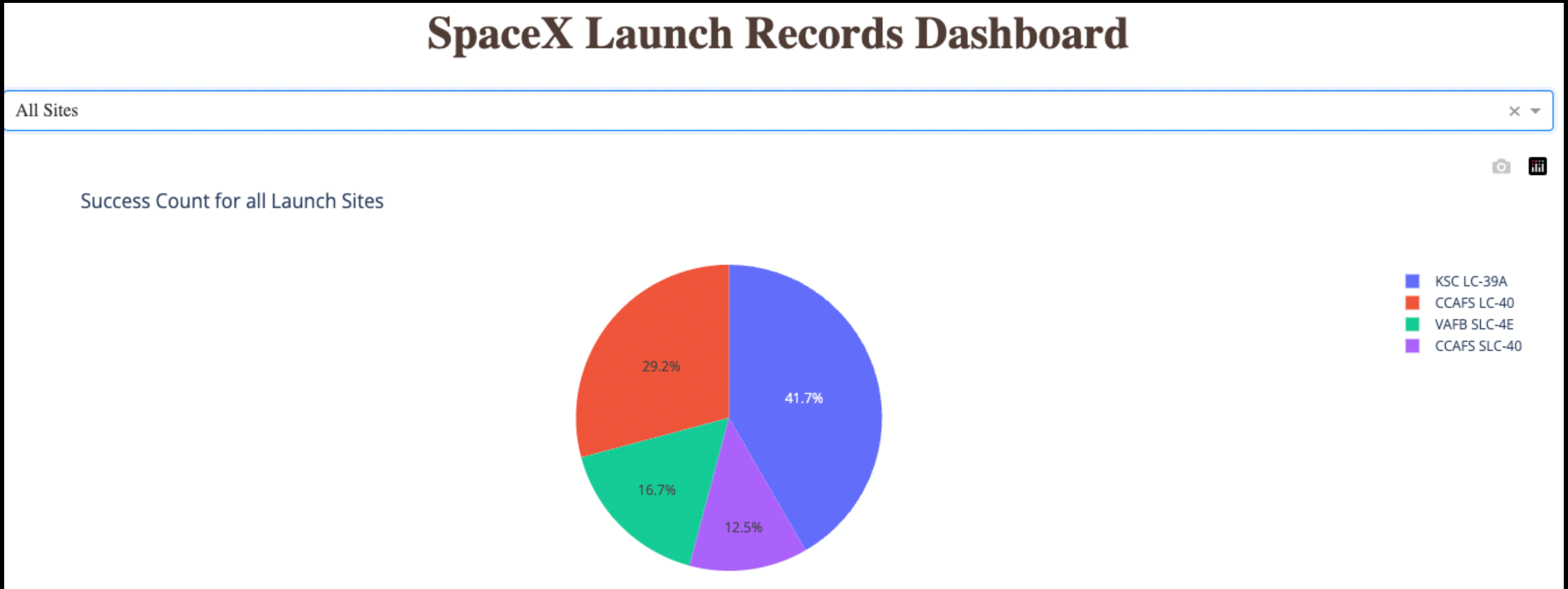
RESULTS
- Plotly Dash Dashboard

Dropdown menu



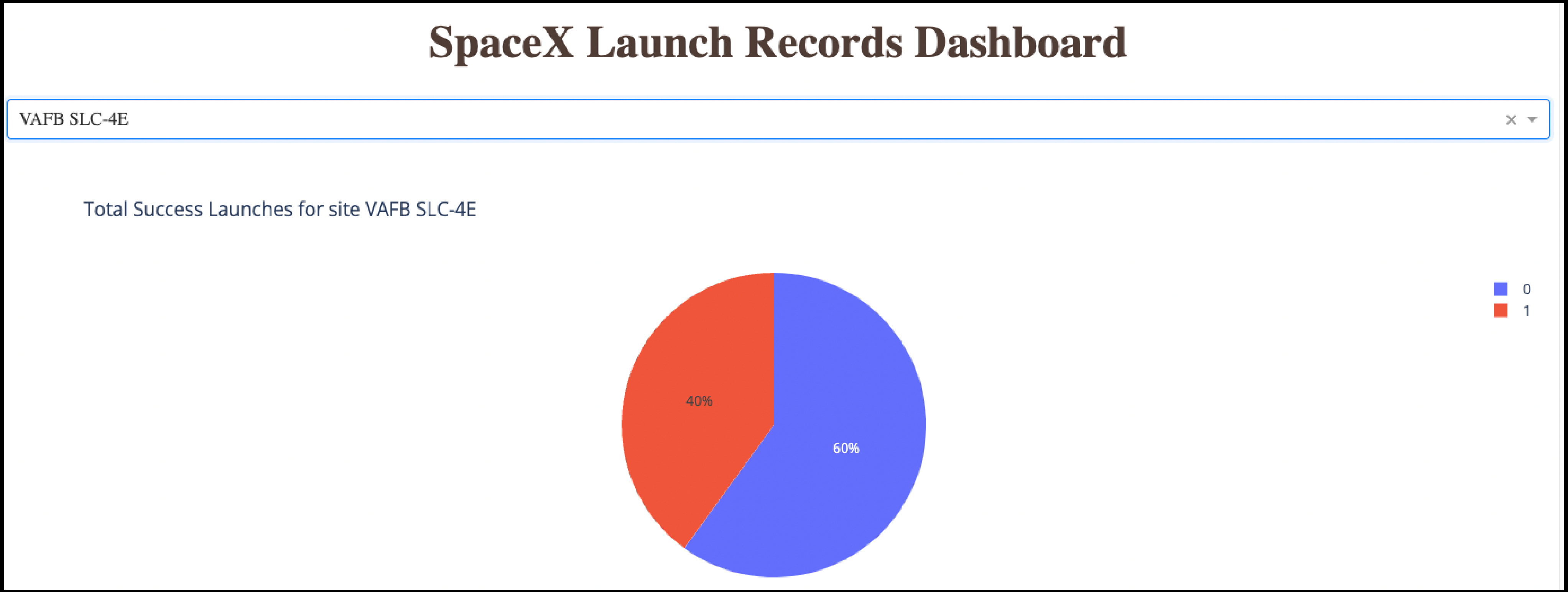
RESULTS
- Plotly Dash Dashboard

Pie chart for all sites are selected



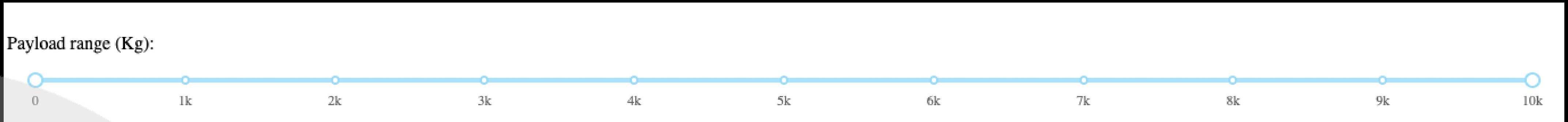
RESULTS
- Plotly Dash Dashboard

Pie chart for VAFB SLC-4E is selected



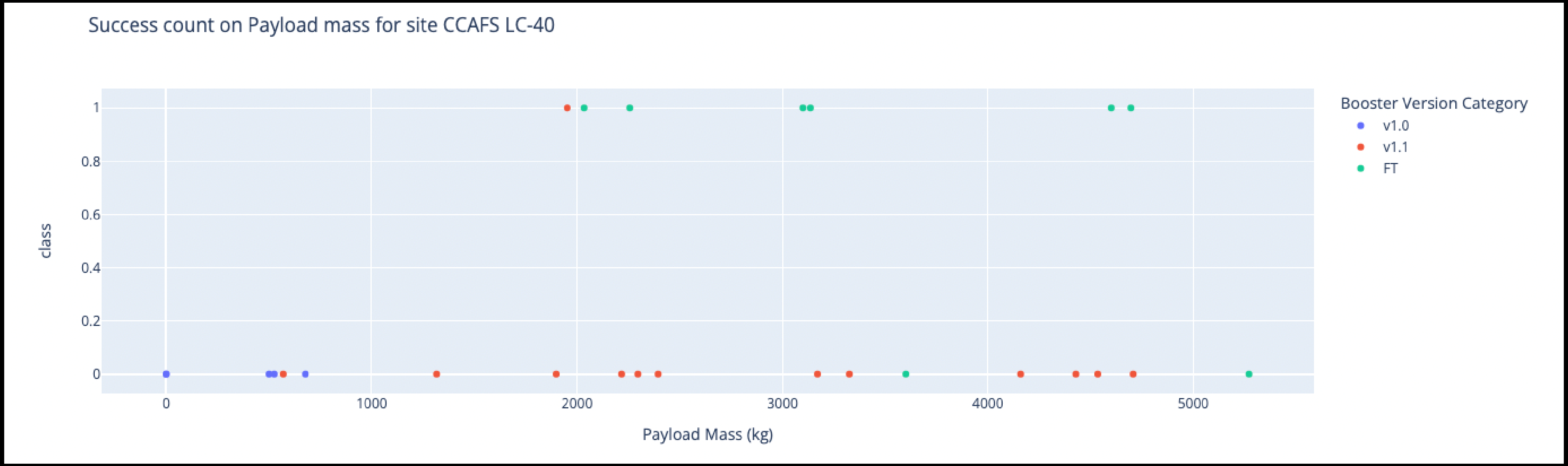
RESULTS
- Plotly Dash Dashboard

Payload Range Slider



RESULTS
- Plotly Dash Dashboard

Scatter plot with the x axis to be the payload and the y axis to be the launch outcome



RESULTS
- Predictive Analysis (Classification)

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']).

```
Y = data['Class'].to_numpy()
Y
```

```
array([0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1,
       1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1], dtype=int64)
```

TASK 2

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

```
# students get this
transform = preprocessing.StandardScaler()
X = transform.fit_transform(X)
X
```

```
array([[-1.71291154e+00, -1.94814463e-16, -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 (Classification)

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`

```
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.

```
Y_test.shape
```

```
(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`.

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

```
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)
```

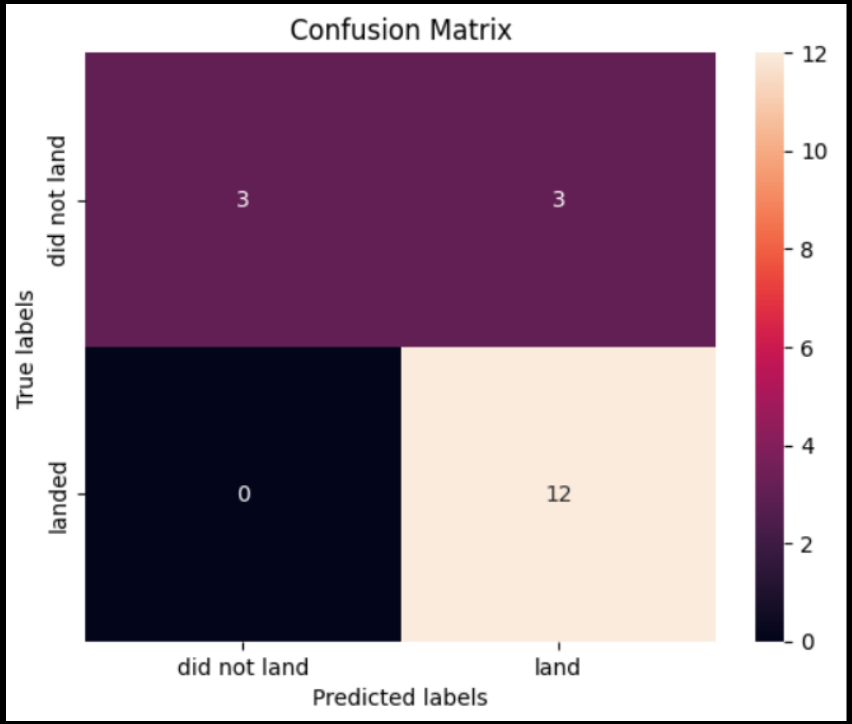
```
GridSearchCV(cv=10, estimator=LogisticRegression(),
             param_grid={'C': [0.01, 0.1, 1], 'penalty': ['l2'],
                         'solver': ['lbfgs']})
```

RESULTS
- Predictive Analysis (Classification)

Logistic Regression

Tuned hpyerparameters :(best parameters) {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs' }

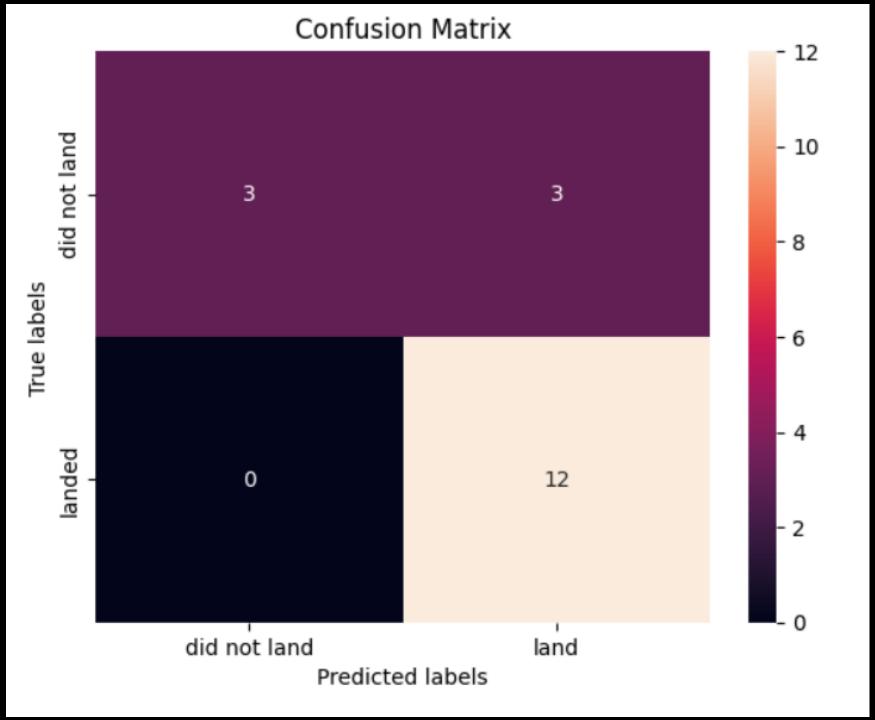
Accuracy : 0.8464285714285713



SVM

Tuned hpyerparameters :(best parameters) {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}

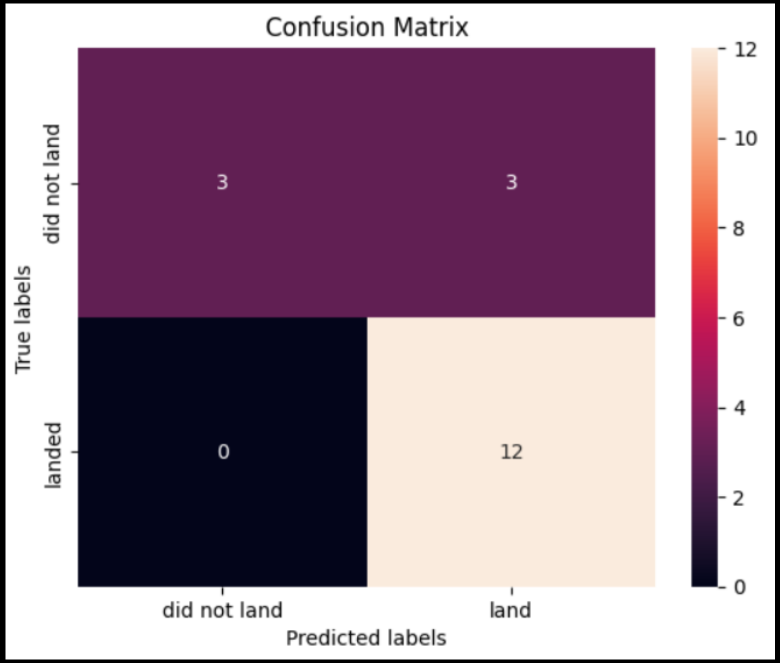
Accuracy : 0.8482142857142856



RESULTS
- Predictive Analysis (Classification)

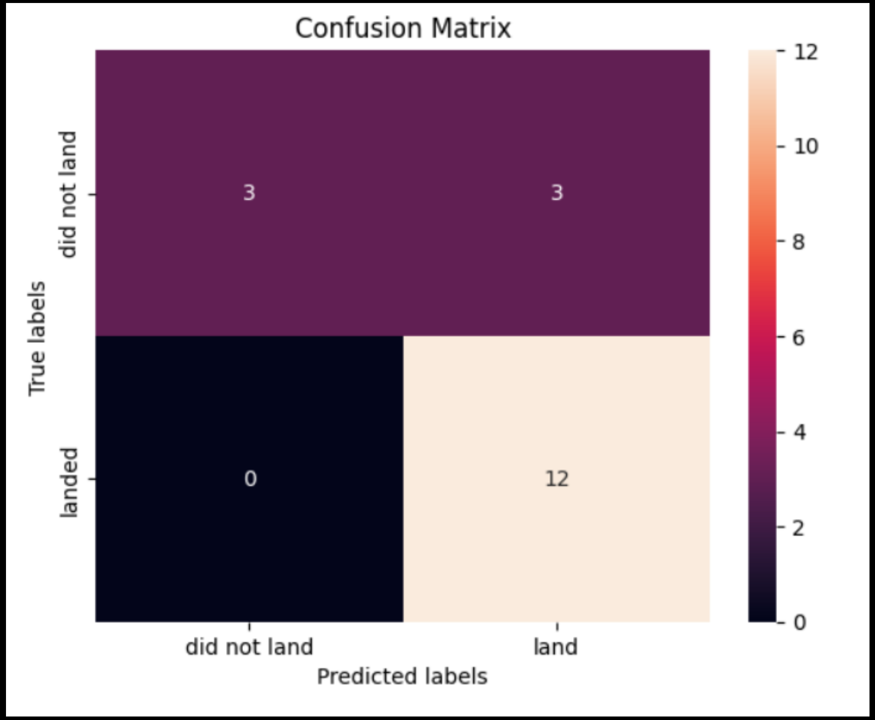
Decision Tree Classifier

Tuned hpyerparameters :(best parameters)
{'criterion': 'entropy', 'max_depth': 4, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
Accuracy : 0.875



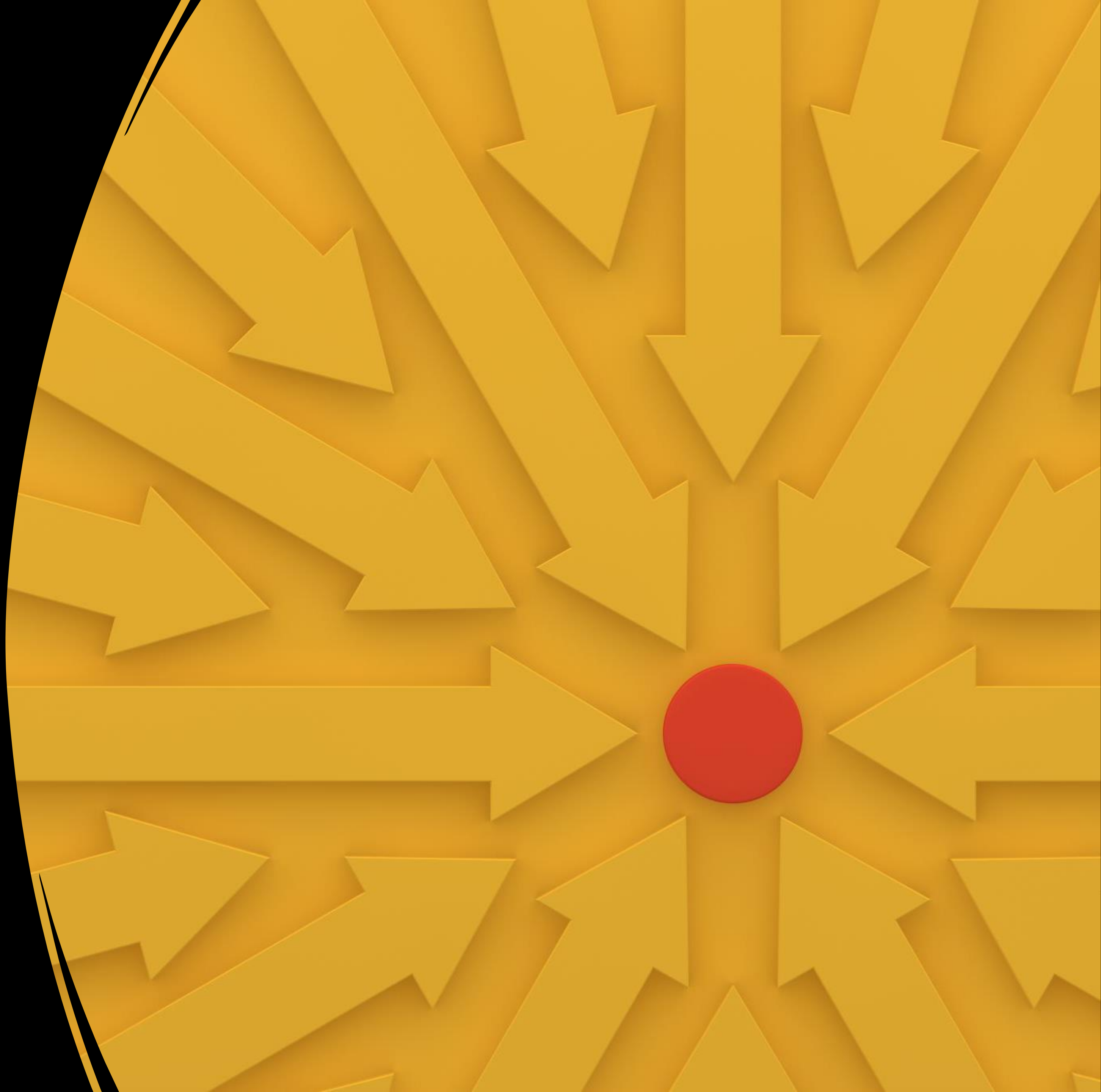
K Nearest Neighbors

Tuned hpyerparameters :(best parameters)
{'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
Accuracy : 0.8482142857142858



CONCLUSION

- Different launch sites have different success rates
- The success rate of launches increases over years
- Most of launch sites are in proximity to the coast
- Decision Tree Model is the best algorithm.



APPENDIX

Special Thanks to:
Coursera and IBM

Detailed Information for this presentation:
[IBM_Applied-Data-Science-Capstone]