Winning Space Race with Data Science

Parthiban S.M.
08-Feb-2024

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection with SpaceX API
  - Data collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis (EDA) using SQL
  - EDA using Data Visualization
  - Interactive Visual Analytics with Folium and Dashboard
  - Predictive Analysis with Machine Learning
- Summary of all results
  - Exploratory data analysis results
  - Interactive analytics demo in screenshots
  - Predictive analysis results

Introduction

- Project background and context

  Space X 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 Space X can reuse the first stage. Therefore 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 space X for a rocket launch. The objective of this project is to create a machine learning pipeline to predict if the first stage will land successfully.

- Problems you want to find answers
  - What factors determine if the rocket will land successfully?
  - The interaction amongst various features that determine the success rate of a successful landing.
  - What operating conditions needs to be in place to ensure a successful landing program.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was obtained using SpaceX API and web-scraping from Wikipedia.
- Perform data wrangling
  - One-hot encoding was done on categorical 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
  - Logistic Regression, SVM, Decision Tree, and KNN models were used
  - Hyperparameters were tuned using Grid Search cross-validation
  - Accuracy on test data was calculated using score method and confusion matrix was plotted.

Data Collection

- Following steps were involved in Data Collection:
  - SpaceX API:
    - Data was collected by sending a GET request to SpaceX API
    - The data was decoded as json using .json() and turned into a DataFrame using. json_normalize()
    - The data was then cleaned, missing values were identified and filled where necessary
- Web-Scraping:
  - Falcon 9 launch records were also collected by web-scraping from Wikipedia
  - A HTTP GET request was sent to Falcon 9 launch HTML page
  - The data was parsed and stored as a DataFrame using BeautifulSoup.

Data Collection – SpaceX API

- A GET request was sent to the SpaceX API.
- The data was decoded using .json() and converted to DataFrame using .json_normalize()

https://github.com/partthy/DataScienceCapstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

1. GET Request to SpaceX API

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
response = requests.get(spacex_url)
```

2. Decoded using .json() and converted to DataFrame using .json_normalize()

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call
```

```
data = pd.json_normalize(response.json())
```

3. Data Cleaning and Dealing With Missing Values

```
# Calculate the mean value of PayloadMass column
mean_PayloadMass = data_falcon9['PayloadMass'].mean()

# Replace the np.nan values with its mean value

data_falcon9['PayloadMass'].replace(np.nan, mean_PayloadMass, inplace = True)
```

Data Collection - Scraping

- HTTP GET request was sent to Falcon 9 Launch HTML page
- Data was parsed as stored as DataFrame using BeautifulSoup.
- https://github.com/partthy/DataScienceCapstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

1. HTTP GET Request to SpaceX Falcon 9 Launch HTML Page

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

```
bsoup = requests.get(static_url).text
```

2. BeatufilSoup Object

```
bsoup_obj = BeautifulSoup(bsoup, "html.parser")
```

```
# Use soup.title attribute
print(bsoup_obj.title)
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

3. Extract all column/variable names from the HTML table header

```
html_tables = bsoup_obj.find_all('table')
first_launch_table = html_tables[2]
column_names = []
th_elements = first_launch_table.find_all('th')
for i in th_elements:
    name = extract_column_from_header(i)
    if name is not None and len(name) > 0:
        column_names.append(name)
```

4. Parsed Falcon 9 Launch Records to create a DataFrame

5. Exported DataFrame to CSV

Data Wrangling

- Calculated the number of launches on each site, the number and occurrence of each orbit, and types and respective number of landing outcomes.
- Created a landing outcome label form the Outcome column
- https://github.com/partthy/DataScienceCapstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

1. Number of Launches on Each Site

```
In [11]: df['LaunchSite'].value_counts()
Out[11]: CCSFS SLC 40     55
         KSC LC 39A       22
         VAFB SLC 4E      13
         Name: LaunchSite, dtype: int64
```

2. Number and Occurrence of Each Orbit

```
In [13]: df['Orbit'].value_counts()
Out[13]: GTO      27
         ISS      21
         VLEO     14
         PO        9
         LEO       7
         SSO       5
         MEO       3
         ES-L1     1
         HEO       1
         SO        1
         GEO       1
         Name: Orbit, dtype: int64
```

3. Types and Number of Landing Outcomes

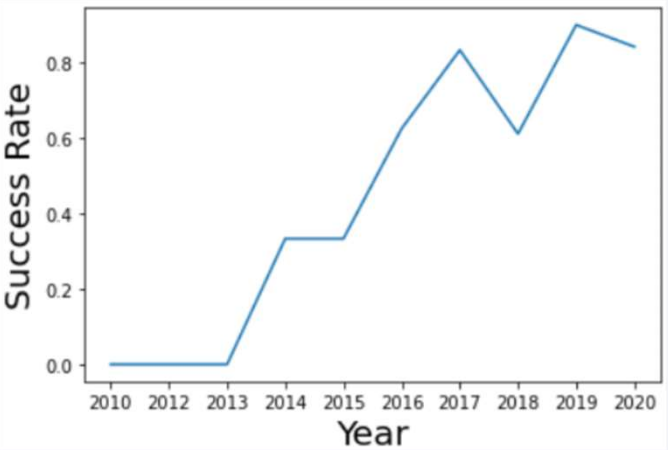
```
In [ ]: landing_outcomes = df['Outcome'].value_counts()
```

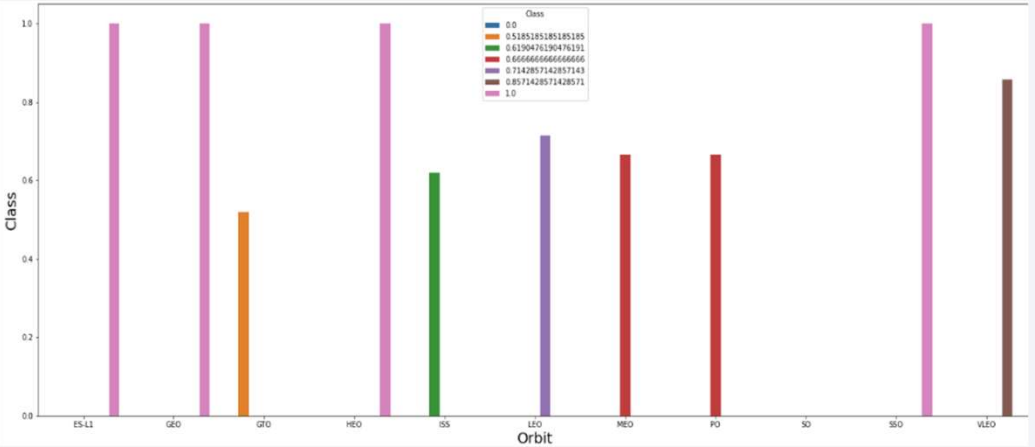
4. Landing Column Label from Outcome Column

```
In [ ]: # landing_class = 0 if bad_outcome
        # landing_class = 1 otherwise
        landing_class = []
        for i in range(len(df['Outcome'])):
            if df['Outcome'][i] in bad_outcomes:
                landing_class.append(0)
            else:
                landing_class.append(1)
```

EDA with Data Visualization

- We visualized the relationship between flight number and launch site, payload and launch site, success rate and orbit, number of flights and orbit, payload mass and orbit, and launch success yearly trend
- https://github.com/partthy/DataScienceCapstone/blob/main/jupyter-labs-eda-dataviz.ipynb





EDA with SQL

- SQL table was loaded in Jupyter and following SQL queries were performed:
  - The names of unique launch sites in the space mission.
  - The total payload mass carried by boosters launched by NASA (CRS)
  - The average payload mass carried by booster version F9 v1.1
  - The total number of successful and failure mission outcomes
  - The failed landing outcomes in drone ship, their booster version and launch site names
- https://github.com/partthy/DataScienceCapstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite%20(1).ipynb

Build an Interactive Map with Folium

- We marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- We assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- We calculated the distances between the launch site and its proximities like railways, highways, and cities
- https://github.com/partthy/DataScienceCapstone/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

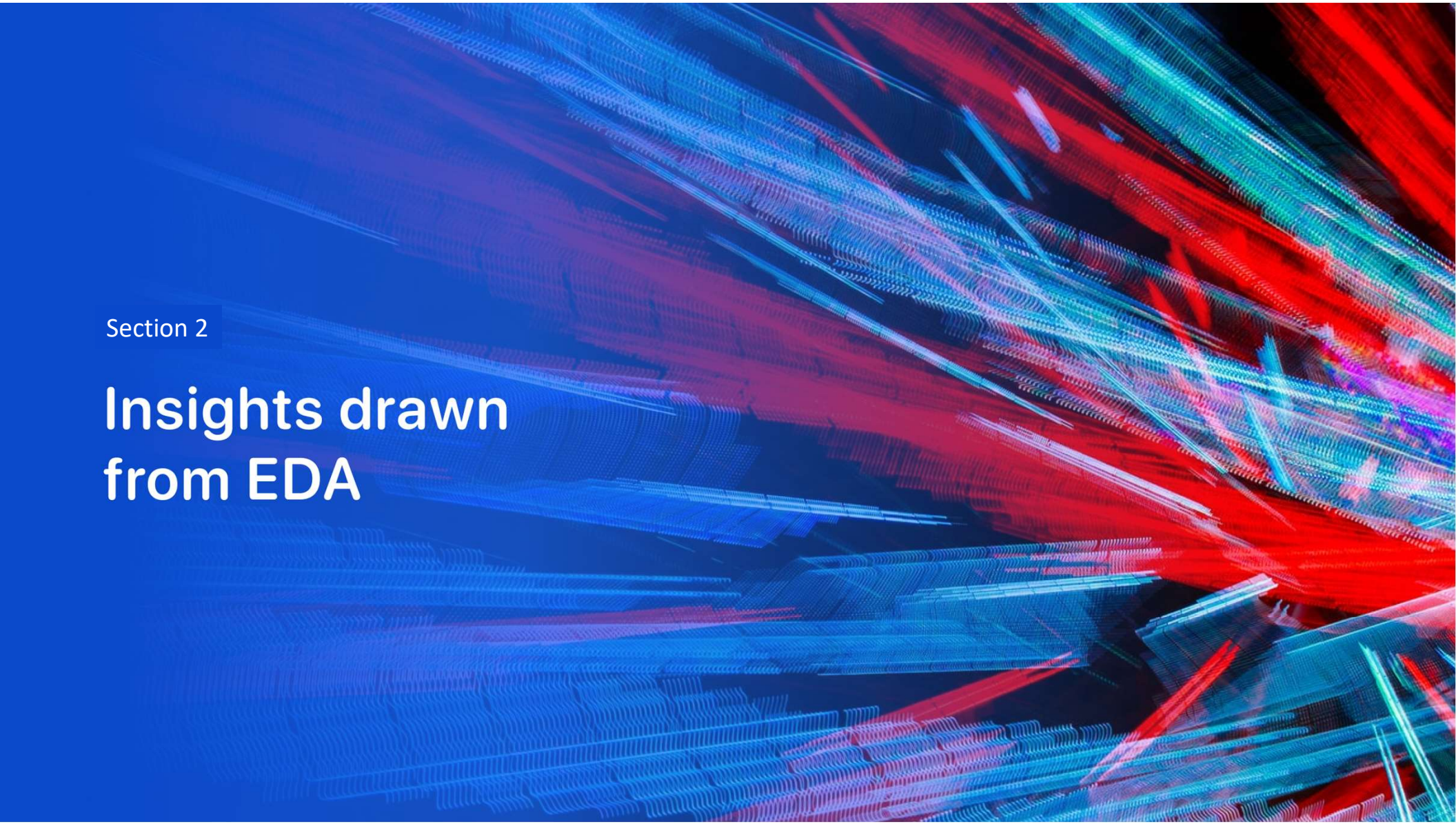
- We built an interactive dashboard with Plotly dash
- We plotted pie charts showing the total launches by a certain sites
- We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- https://github.com/partthy/DataScienceCapstone/blob/main/Dashboard.ipynb

Predictive Analysis (Classification)

- We split the dataset into training and testing set.
- We built different machine learning models and tuned different hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, and improved the model using feature engineering and hyperparameter tuning.
- We found the best performing classification model
- https://github.com/partthy/DataScienceCapstone/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

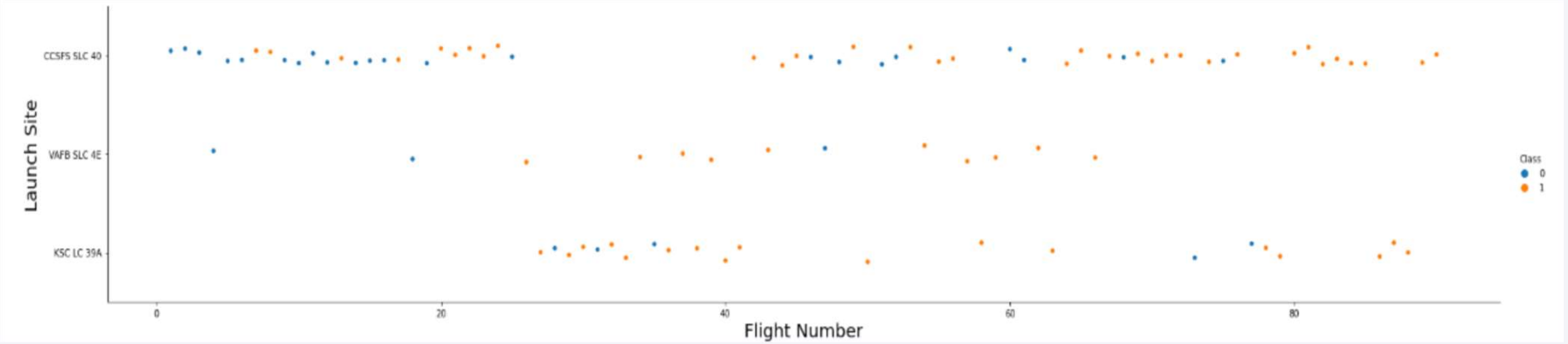
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results



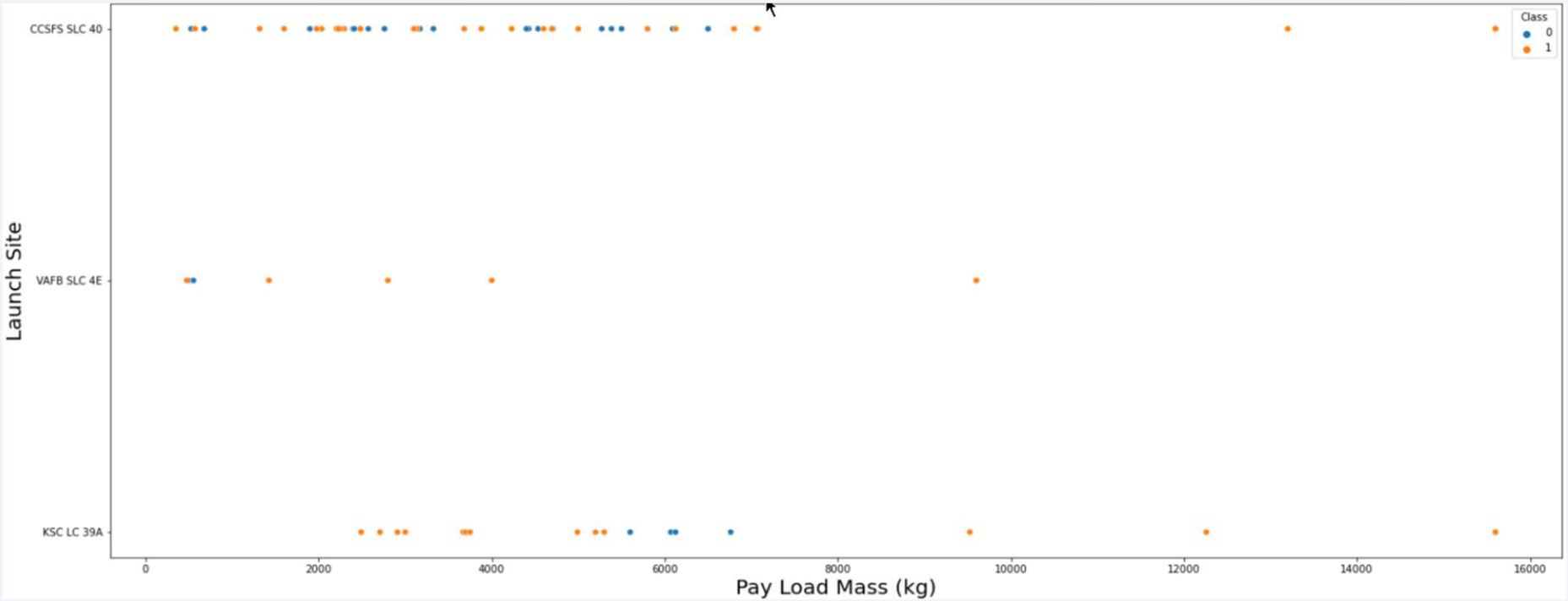
Section 2

Insights drawn from EDA

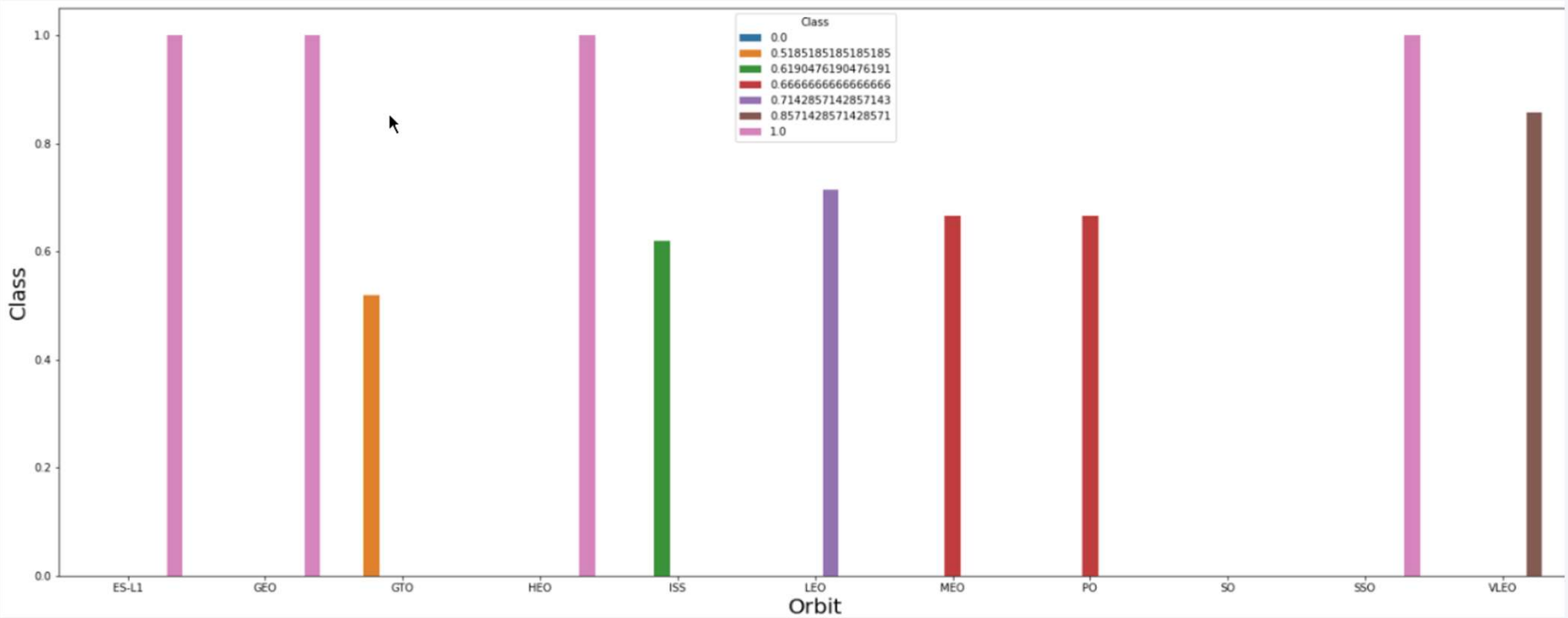
Flight Number vs. Launch Site



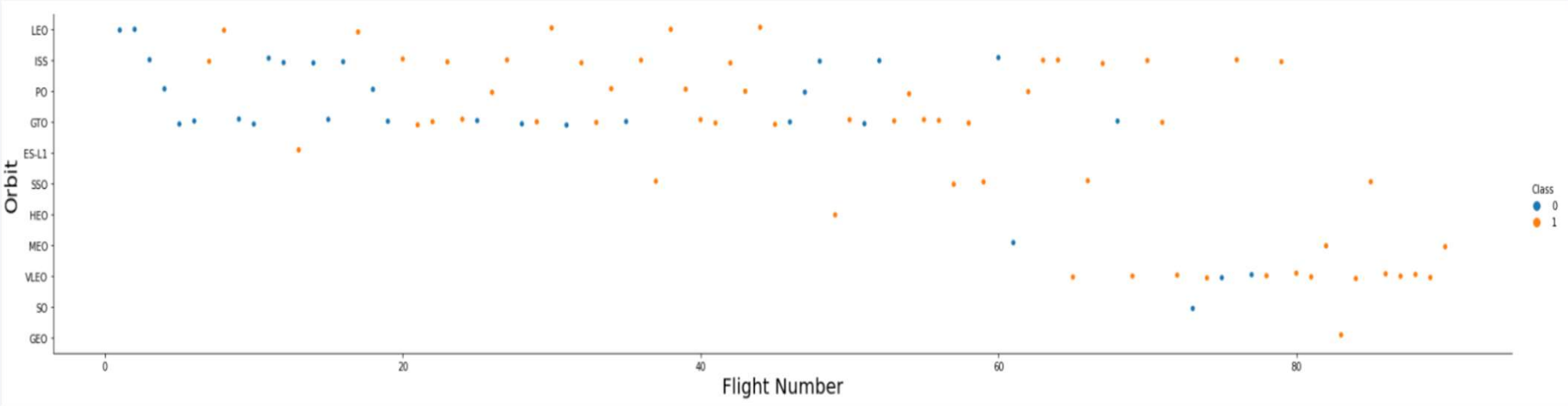
Payload vs. Launch Site



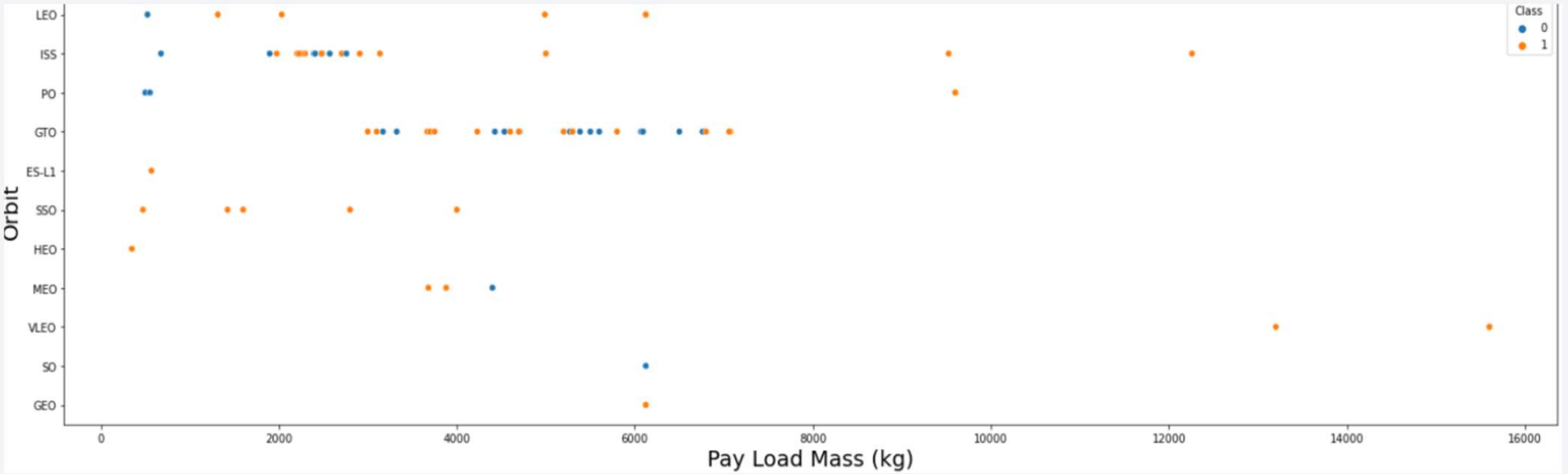
Success Rate vs. Orbit Type



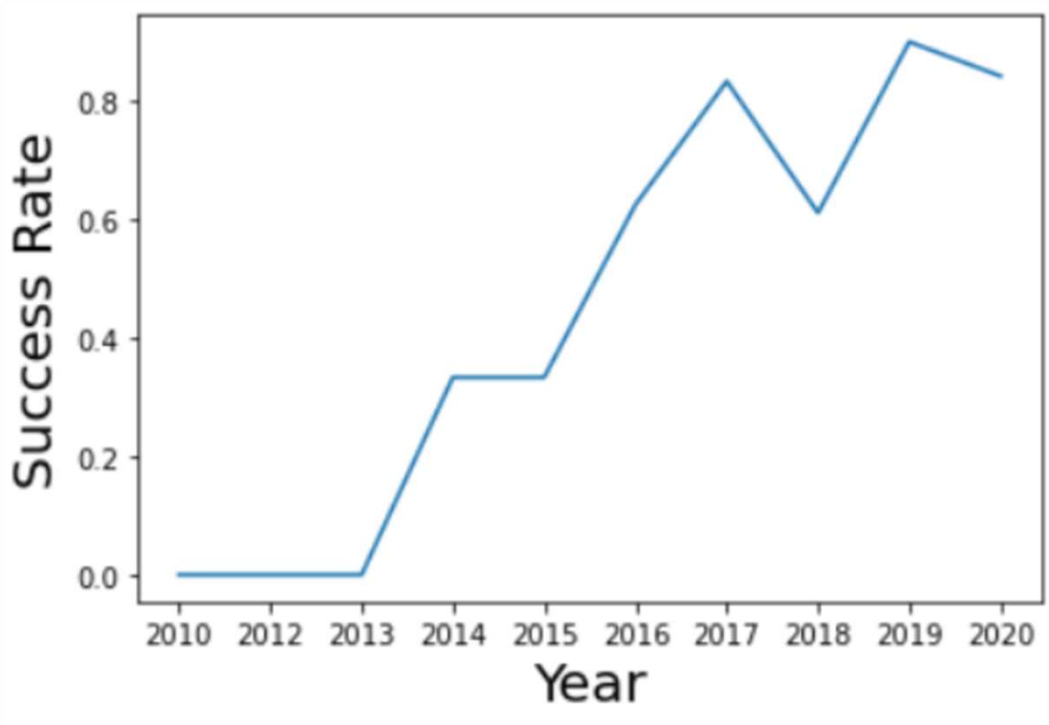
Flight Number vs. Orbit Type



Payload vs. Orbit Type



Launch Success Yearly Trend



All Launch Site Names

- We use the keyword DISTINCT to
- select unique launch sites from
- the table.

```
%sql select distinct(launch_site) from SPACEXTBL
```

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

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

Launch Site Names Begin with 'CCA'

- The keyword LIKE is used to specify that the launch site name begins with CCA.
- The limit keyword is used to query only 5 records

```
%%sql
select * from SPACEXTBL
where launch_site like "CCA%"
limit 5
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel 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	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- The total payload mass for customer NASA (CSR) is 45,596 kg.
- The function SUM is used to obtain the sum of payload mass.
- The WHERE clause is used to specify that the customer should be NASA (CRS)

```
%sql select sum(payload_mass__kg_) as 'total payload mass for NASA (CRS)' from SPACEXTBL where customer='NASA (CRS)'
```

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

total payload mass for NASA (CRS)
45596

Average Payload Mass by F9 v1.1

- The average payload mass for rockets with booster version F9 v1.1 is 2928.4 kg.
- The function AVG is used to calculate the average of payload mass.
- The WHERE clause is used to specify that the booster version should be F9 v1.1

```
%sql select avg(payload_mass__kg_) as 'average payload mass for booster F9 v1.1' from SPACEXTBL where booster_version='F9 v1.1'
 * sqlite:///my_data1.db
Done.
```

average payload mass for booster F9 v1.1
2928.4

First Successful Ground Landing Date

- The first successful ground landing occurred on 22 December 2015.
- The function MIN is used to find the earliest date.
- The WHERE clause is used to specify that the landing outcome is a successful ground pad landing.

```
%%sql
select max(`Date`) from SPACEXTBL
where `landing _outcome`='Success (ground pad)'
```

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

max(`Date`)
22-12-2015

Successful Drone Ship Landing with Payload between 4000 and 6000

- The names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000 – F9 FT B1022, F9 FT B1026, F9 FT B1021.2, F9 FT B1031.2
- The WHERE clause specifies that the landing was a success drone ship landing.
- The AND clause adds another condition that the payload mass is between 4000 and 6000

```
%%sql
select booster_version from SPACEXTBL
where `landing _outcome`='Success (drone ship)' and `payload_mass__kg_` between 4001 and 5999
```

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

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

Total Number of Successful and Failure Mission Outcomes

- Total Number of Successful Mission Outcomes – 61
- Total Number of Failure Mission Outcomes - 10
- The COUNT function gives the number of landings.
- The LIKE clause is used to specify that the outcome is a success or a failure.

```
%%sql
select count(`landing _outcome`) as 'Number of successful Outcomes'
from SPACEXTBL where `landing _outcome` like 'Success%'
```

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

Number of successful Outcomes
61

```
%%sql
select count(`landing _outcome`) as 'Number of failure Outcomes'
from SPACEXTBL where `landing _outcome` like 'Failure%'
```

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

Number of failure Outcomes
10

Boosters Carried Maximum Payload

- The WHERE clause is used to specify that the booster version with the maximum payload mass is to be selected
- The MAX function gives the maximum payload mass.

```
%%sql
select `booster_version` from SPACEXTBL
where `payload_mass__kg_`= (select max(`payload_mass__kg_`) from SPACEXTBL)
```

```
 * 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 SUBSTR function selects the year out of the date.
- The WHERE clause specifies that the date should be 2015 and the landing outcome should be a drone ship failure.

```
%%sql
select substr(Date, 4, 2) as 'month names', `landing _outcome`, booster_version, launch_site from SPACEXTBL
where substr(Date,7,4)='2015' and `landing _outcome`='Failure (drone ship)'
```

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

month names	Landing _Outcome	Booster_Version	Launch_Site
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
- The GROUP BY clause was used to group the landing outcomes and the order by clause was used to sort the grouped landing outcome in descending order

```
%%sql
select `landing _outcome`, count(`landing _outcome`) as 'No. of successful landing between 04-06-2010 and 20-03-2017'
from SPACEXTBL
where (`landing _outcome` like 'Success%') and `Date` between '04-06-2010' and '20-03-2017'
group by `landing _outcome`
order by 'No. of successful landing between 04-06-2010 and 20-03-2017' desc
```

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

Landing _Outcome	No. of successful landing between 04-06-2010 and 20-03-2017
Success (ground pad)	6
Success (drone ship)	8
Success	20

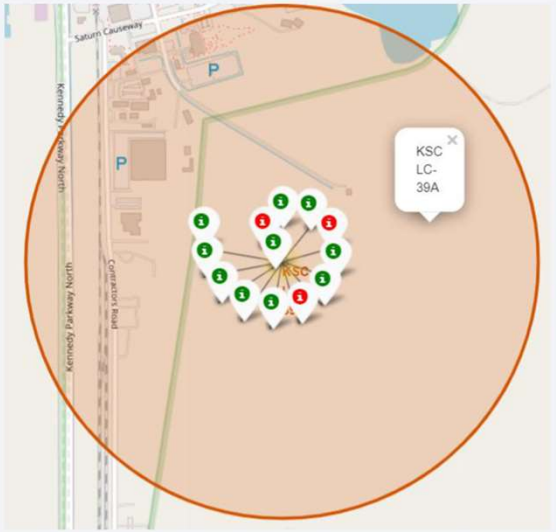
Section 3

Launch Sites Proximities Analysis

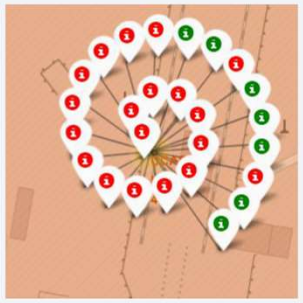
SpaceX Launch Sites on the Global Map

- The SpaceX Launch Sites are on the coasts Florida and California in America
- Only the launch site VAFB SLC – 4E is in California; the rest are in Florida

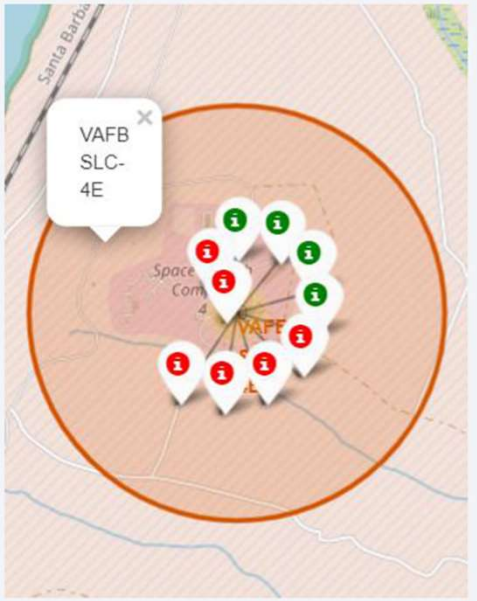
Success/Failed Launches for Each Launch Site



Florida Launch Sites



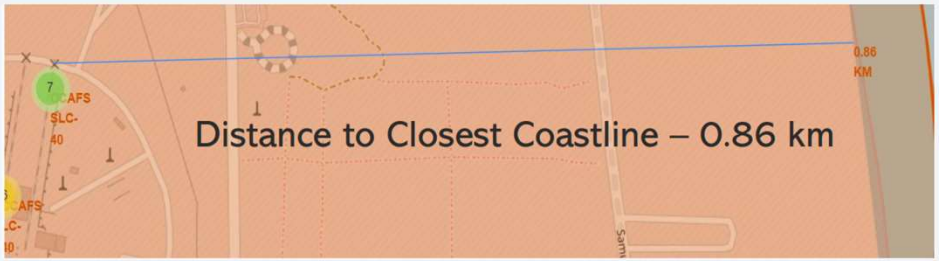
CCAFS LC-40

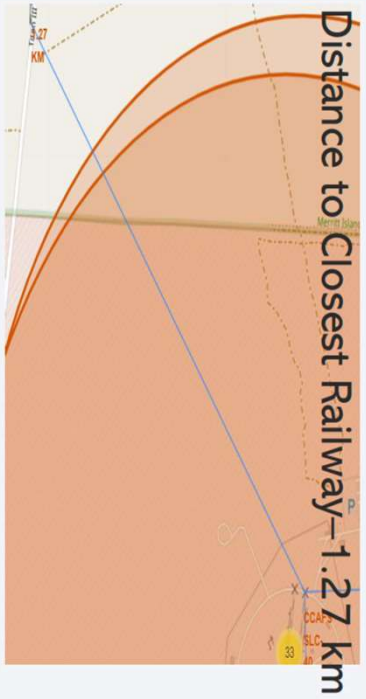


California Launch Site

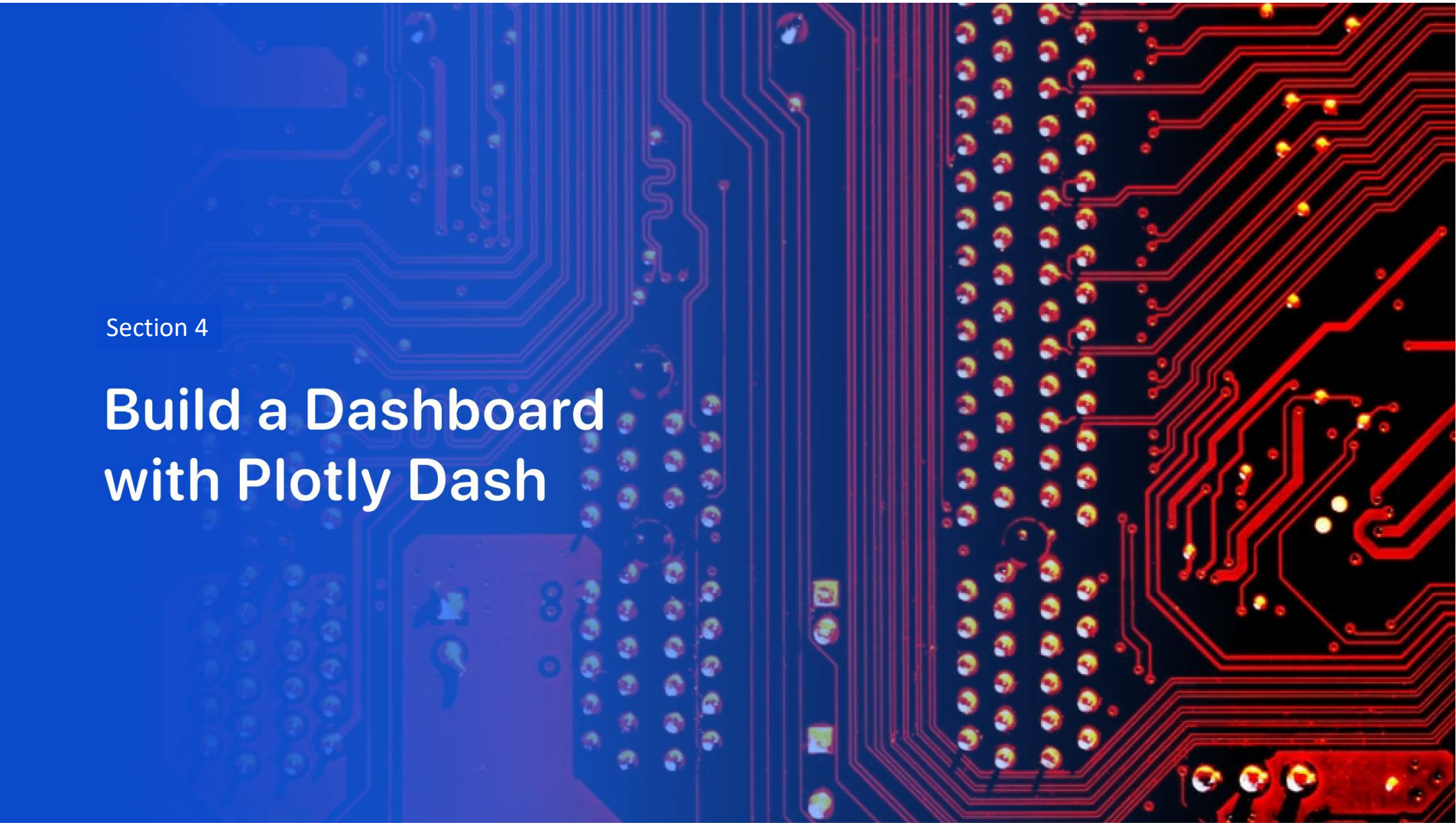
Red Markers – Failure; Green Markers - Success

Launch Site Proximity From Landmarks





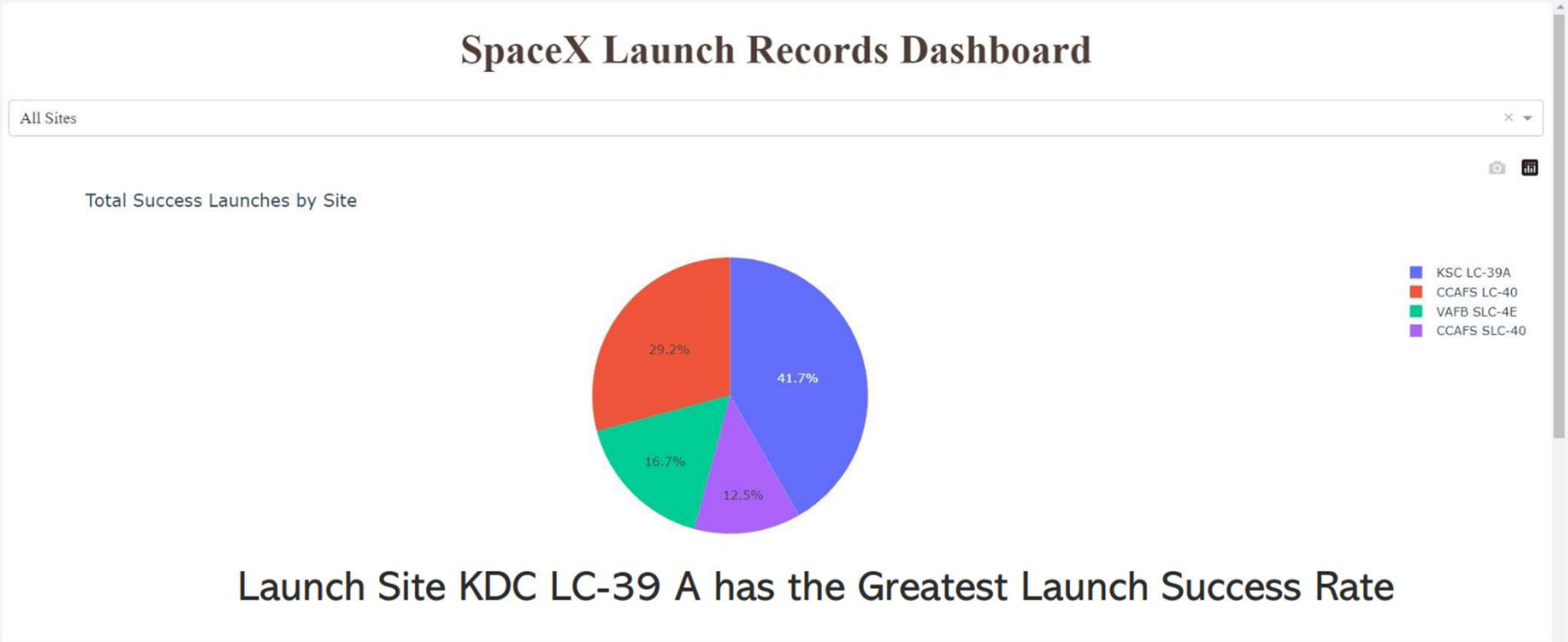
Are launch sites in close proximity to railways? – Yes (1.27 km)
Are launch sites in close proximity to highways? Yes (0.58 km)
Are launch sites in close proximity to coastline? – Yes (0.86 km)
Do launch sites keep certain distance away from cities? – Yes (54.9 km from Melbourne Beach)



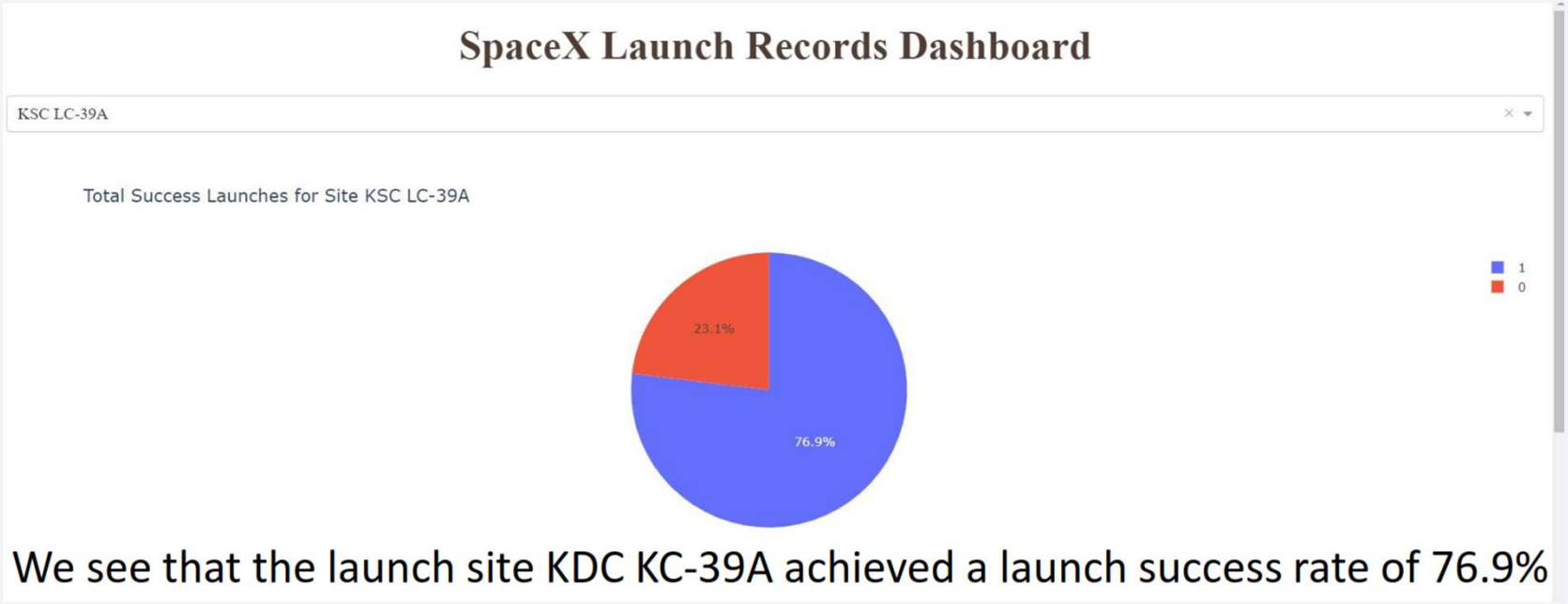
Section 4

Build a Dashboard with Plotly Dash

Pie Chart of Total Success Launches by all Sites



Pie Chart of Success/Failure Launch Rate of KDC LC-39 A



We see that the launch site KDC KC-39A achieved a launch success rate of 76.9%

Payload vs. Launch Outcome

Low Payload Weight

0 – 4000 kg

Heavy Payload Weight

4000 – 10,000 kg

Success Rate for Heavier Payload is lesser than that for Lighter Payload 41

Section 5

Predictive Analysis (Classification)

Classification Accuracy

```
models = [logreg_cv, svm_cv, tree_cv, knn_cv]
results = []
for model in models:
    result_dict = {'Model':str(model.estimator), 'Accuracy':str(model.best_score_), 'Score':str(model.score(X_test, Y_test))}
    results.append(result_dict)
performance_df = pd.DataFrame(results)
performance_df
```

	Model	Accuracy	Score
0	LogisticRegression()	0.8464285714285713	0.8333333333333334
1	SVC()	0.8482142857142856	0.8333333333333334
2	DecisionTreeClassifier()	0.875	0.7222222222222222
3	KNeighborsClassifier()	0.8482142857142858	0.8333333333333334

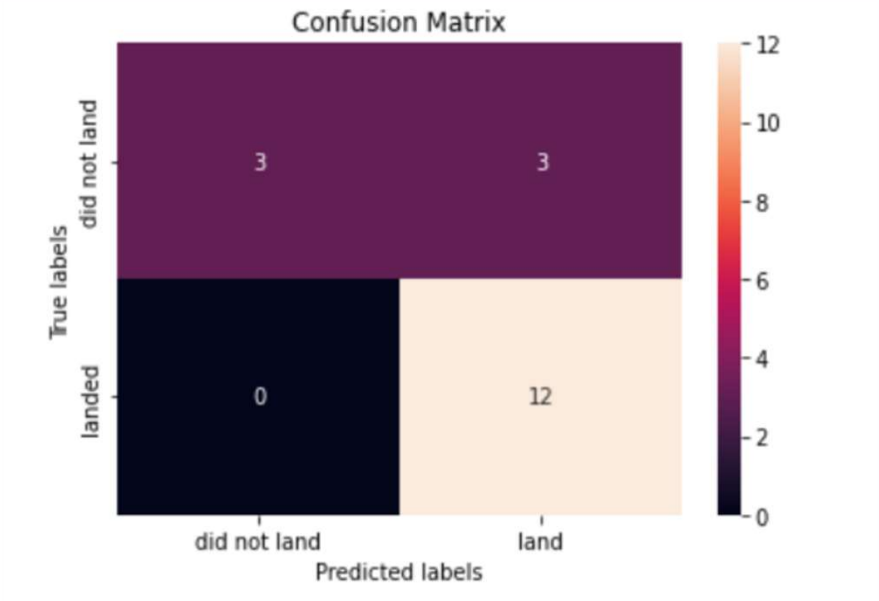
```
print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 6, 'max_features': 'sqrt', 'min_samples_leaf': 1,
'min_samples_split': 10, 'splitter': 'random'}
accuracy : 0.875
```

We can see that the decision tree classifier has the highest classification accuracy of 0.875

Confusion Matrix

- Best Model – Decision Tree Classifier
- The confusion matrix for the Decision Tree Classifier shows that the classifiers gives a high number of false positives, i.e., the rocket did not actually land successfully but the classifiers predicts that it landed successfully.



Conclusions

- The greater the number of flights at a launch site, greater is the success rate at the launch site.
- ES-L1, GEO, HEO, and SSO orbits have the highest success rate.
- For heavier payloads, success rate increases for LEO, ISS and Polar orbits.
- Launch success rate increased from 2013 to 2020 with minor fluctuations throughout the period.
- Launch Site KDC LC-39 A has the Greatest Launch Success Rate
- The Decision Tree Classifier is the best machine learning algorithm to predict SpaceX Falcon 9 rocket's first stage will land successfully.

Thank you!