IBM Applied Data Science Capstone – Space Y

Ankit Roy
11 November, 2021

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection
  - Data wrangling
  - EDA with data visualization
  - EDA with SQL
  - Building an interactive map with Folium
  - Building a Dashboard with Plotly
  - Predictive analysis
- Summary of all results
  - EDA results
  - Interactive analysis screenshot
  - Predictive analysis results

Introduction

- Project background and context
  - We are to predict if the Falcon 9 first stage will land successfully. SpaceX advertises Falcon 9 rocket launches on its website, with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. 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 SpaceX for a rocket launch
- Problems you want to find answers
  - What factors influences the successful landing of the rocket
  - The effect each factors have on the outcome of the launch
  - The combination of factors to get the best landing success rate

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Making request to SpaceX API
  - Web scraping (Wikipedia)
- Data wrangling methodology
  - One hot encoding of data fields using Machine Learning were done.
- Exploratory data analysis (EDA) using visualization and SQL
  - More insights were gathered about the data by virtue of querying the data using SQL.
  - Using different graphs : Scatter plot, Bar plot, Line plot more information were gathered regarding the relationship between different variables

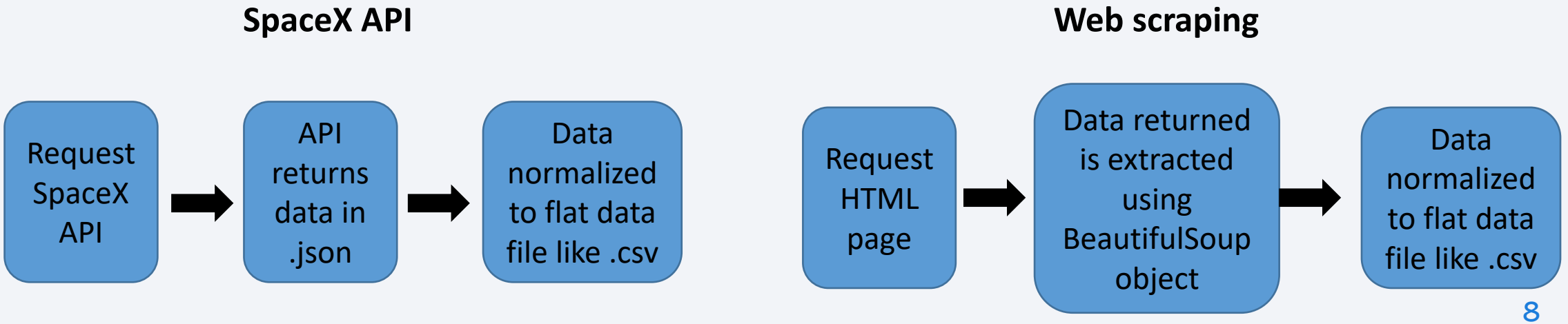
Methodology

Executive Summary

- Interactive visual analytics using Folium and Plotly Dash
  - More insights were gathered regarding the dataset by virtue of analyzing the data using folium and Plotly dashboard
- Predictive analysis using classification models
  - After splitting the data set into test and train data, we determine which classification model is the most accurate estimate of the data.

Data Collection

- In order to make an accurate estimation of the successful landing outcome of the launch data we collected date from two sources.
  1. The SpaceX launch data available in the SpaceX REST API
  2. Web scraping the launch data stored in Wikipedia



Data Collection – SpaceX API

1. Requesting and storing the response from API

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
response = requests.get(spacex_url)
```

2. Normalizing the json response to flat file format

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

3. Applying custom function for data cleaning

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

4. Assigning the list to dictionary then dataframe

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}

df = pd.DataFrame.from_dict(launch_dict)
```

5. Filtering the dataframe

```
data_falcon9 = df[df['BoosterVersion']!='Falcon 1']
data_falcon9.reset_index(inplace= True)
```

6. Data wrangling for missing values

```
mean_payload = data_falcon9["PayloadMass"].astype("float").mean(axis=0)

data_falcon9["PayloadMass"].replace(np.nan, mean_payload, inplace = True)
```

7. Exporting to flat file

```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

[Github URL to notebook]

Data Collection - Scraping

1. Requesting HTML page

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

2. Creating the BeautifulSoup object

```
space_soup = BeautifulSoup(data, 'html5lib')
```

3. Finding all tables

```
html_tables = space_soup.find_all("table")
```

4. Finding all column names

```
x = first_launch_table.find_all('th')
for i in range(len(x)):
    try:
        name = extract_column_from_header(x[i])
        if (name is not None and len(name) > 0):
            column_names.append(name)
    except:
        pass
```

5. Creating a dictionary to parse the HTML table

```
launch_dict= dict.fromkeys(column_names)
```

6. Adding columns with empty values to the dictionary

```
del launch_dict['Date and time ( )']
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

7. Appending data to keys (*Block 13 in notebook)*
8. Converting dictionary to dataframe

```
df=pd.DataFrame(launch_dict)
```

9. Exporting dataframe to flat file(.csv)

```
df.to_csv('spacex_web_scraped.csv', index=False)
```

Github URL to notebook

Data Wrangling

- In the data set, there are several different cases where the booster did not land successfully. Sometimes a landing was attempted but failed due to an accident. We mainly converted those outcomes into Training Labels with 1 means the booster successfully landed 0 means it was unsuccessful.

Calculate the number of launches on each site

```
df["LaunchSite"].value_counts()
```

```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

Calculate the number and occurrence of each orbit

```
df["Orbit"].value_counts()
```

```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
SO        1
GEO       1
HEO       1
ES-L1     1
Name: Orbit, dtype: int64
```

Calculate the number and occurence of mission outcome per orbit type

```
landing_outcomes = df["Outcome"].value_counts()
landing_outcomes
```

```
True ASDS       41
None None       19
True RTLS       14
False ASDS       6
True Ocean       5
None ASDS        2
False Ocean      2
False RTLS       1
Name: Outcome, dtype: int64
```

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

```
0 True ASDS
1 None None
2 True RTLS
3 False ASDS
4 True Ocean
5 None ASDS
6 False Ocean
7 False RTLS
```

Create a landing outcome label from Outcome column

```
landing_class = []
for key,value in df["Outcome"].items():
     if value in bad_outcomes:
        landing_class.append(0)
     else:
        landing_class.append(1)
landing_class
```

```
df['Class']=landing_class
```

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

```
0.6666666666666666
```

Github URL to notebook

EDA with Data Visualization

Scatter Plots

They show the relationship between two variable. How one variable is related to another is determined by the correlation. The following variable combinations were considered for EDA of the data set:

1. Flight Number Vs Payload Mass
2. Flight Number Vs Launch Site
3. Payload Mass Vs Launch Site
4. Flight Number Vs Orbit
5. Payload Mass Vs Orbit

Github URL to notebook

Bar Plot

They shows comparisons among discrete categories. One axis of the chart shows the specific categories being compared, and the other axis represents a measured value. The following variable combinations were considered for EDA of the data set:

1. Orbit Vs Class Mean

Line Plot

They displays information as a series of data points called 'markers' connected by straight line segments. It is usually drawn to visualize a trend in data over a period of time. The following variable combinations were considered for EDA of the data set:

1. Success rate vs Year

EDA with SQL

SQL queries were performed on the dataset to gather further information. Some of the inference that we gathered are as follows:

1. Display names of different launch sites in the data
2. Display the record of 5 launch sites which has their name starting with “CCA”
3. Display the total payload mass carried by boosters launched by NASA (CRS)
4. Display average payload mass carried by booster version F9 v1.1
5. List the date when the first successful landing outcome in ground pad was acheived
6. List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
7. List the total number of successful and failure mission outcomes
8. List the names of the booster_versions which have carried the maximum payload mass.
9. List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
10. 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

Github URL to notebook

Build an Interactive Map with Folium

1. The latitude and longitude of each launch site were considered and a **Circle marker** was added to each site on the folium map with a label to help in identifying each location.

2. The launch_outcomes (failure or success) in the dataframe were assigned variables 0 or 1 and marked by Green or Red colour in the folium map using **MarketCluster**.

3. Using a user defined function calculate_distance, the various distances to different landmarks from the launch site were measured along with **Lines** on the map showing the exact position.

Some trends that were observed from the Folium map analysis are :

- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

Github URL to notebook

Build a Dashboard with Plotly Dash

To get a better insight of how the variables interact amongst themselves we have added the following:

- Pie chart to show the total successful launches of all site, individual site.
- Scatter plot to show the relationship between payload and outcome of launches from all sites

Predictive Analysis (Classification)

To make a predictive analysis, we need to make different classification models based on the data obtained. The process is as follows:

1. The dataset needs to be loaded into NumPy array then store it as a Pandas series
2. The data needs to be standardized and transformed
3. The data needs to be split into training and test data sets
4. The type of machine learning algorithm needs to be decided
5. Parameters need to be set and GridSearchCV object needs to be created
6. The GridSearchCV object needs to be trained with the train data

Model Evaluation

1. The accuracy of each model needs to be ascertained against the test data
2. Best parameters for each type of algorithm are to be determined
3. Confusion matrix needs to be plotted

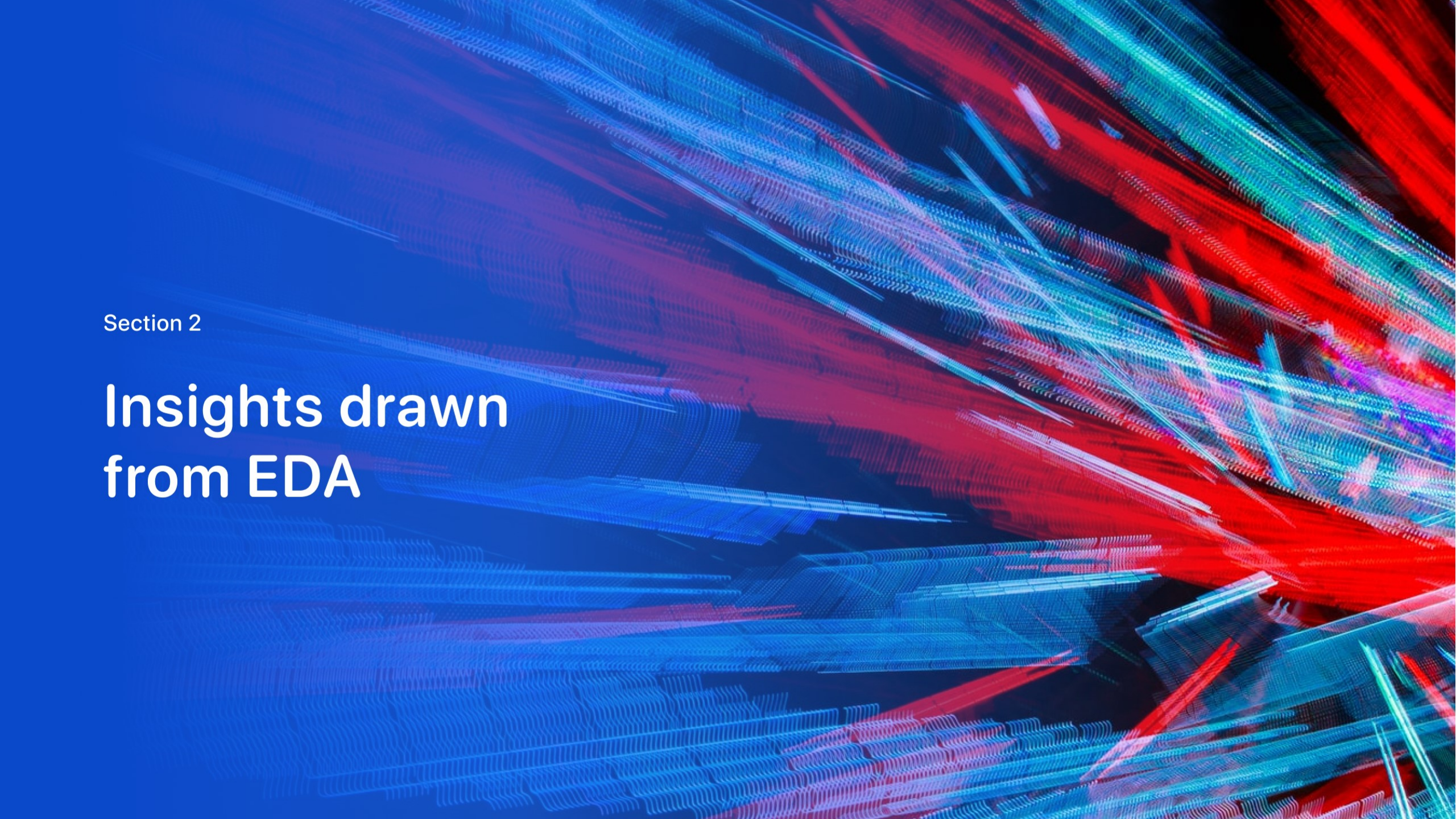
Best Classification Model

1. The model with the best accuracy score is the best suited model for the dataset

Github URL to notebook

Results

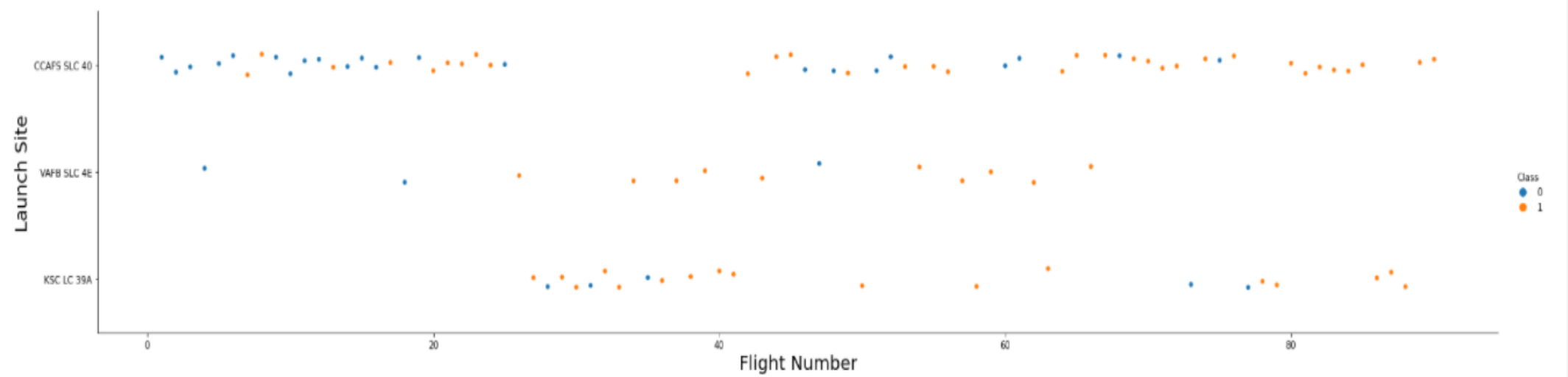
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results



Section 2

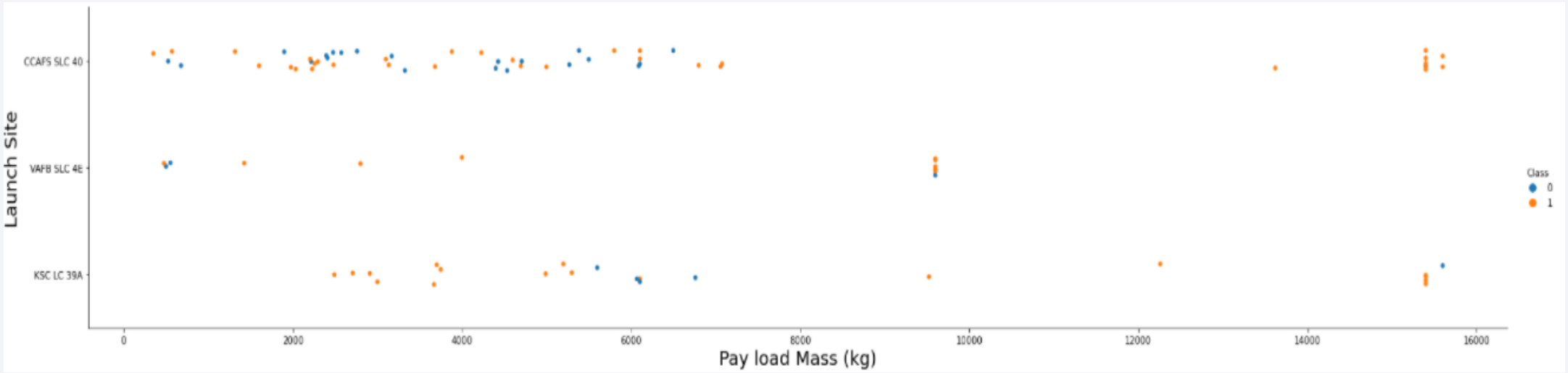
Insights drawn from EDA

Flight Number vs. Launch Site



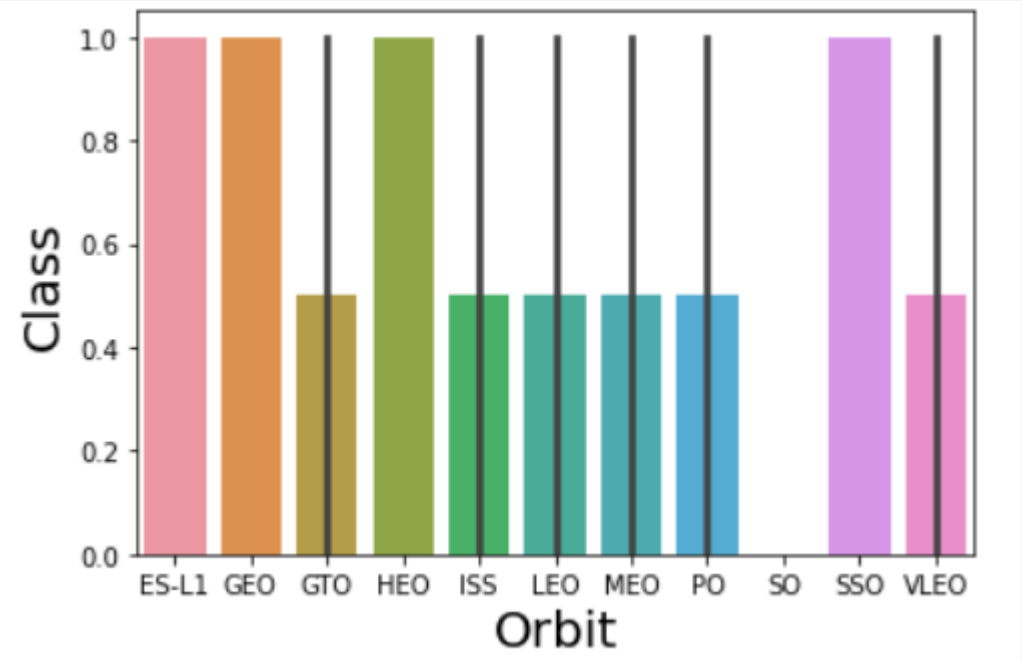
The number of flights at a launch site does not have any relation with the outcome of the landing.

Payload vs. Launch Site



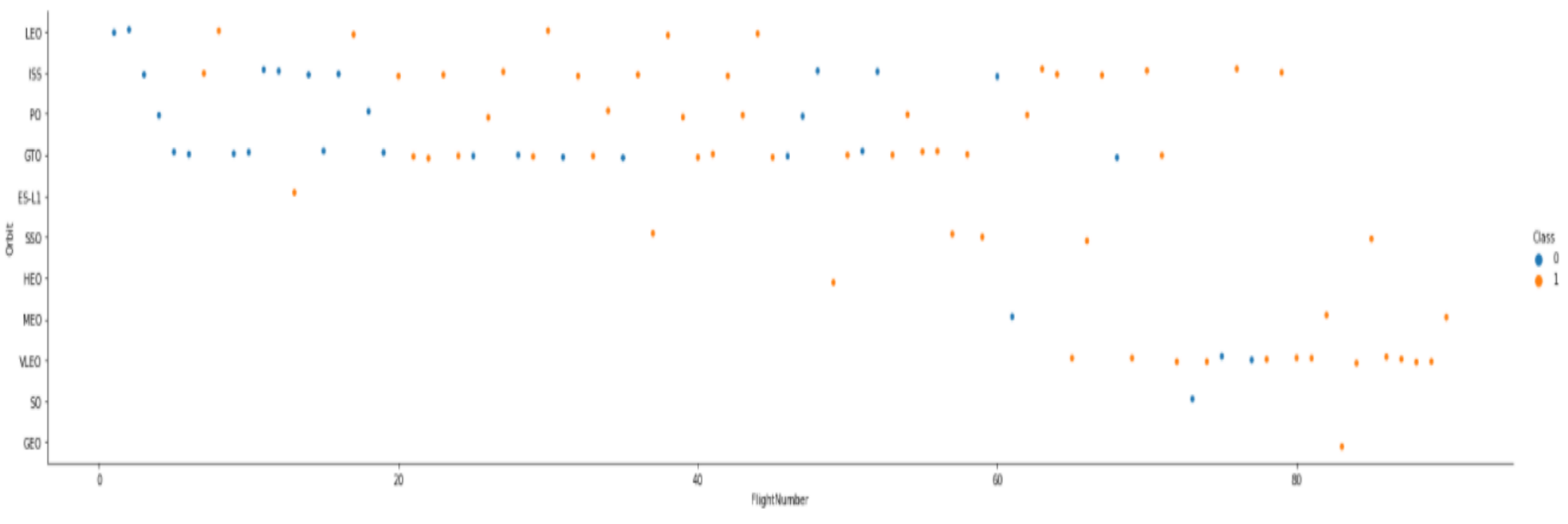
A clear pattern is noticeable that with higher payload mass, the success rate is higher at each individual launch site

Success Rate vs. Orbit Type



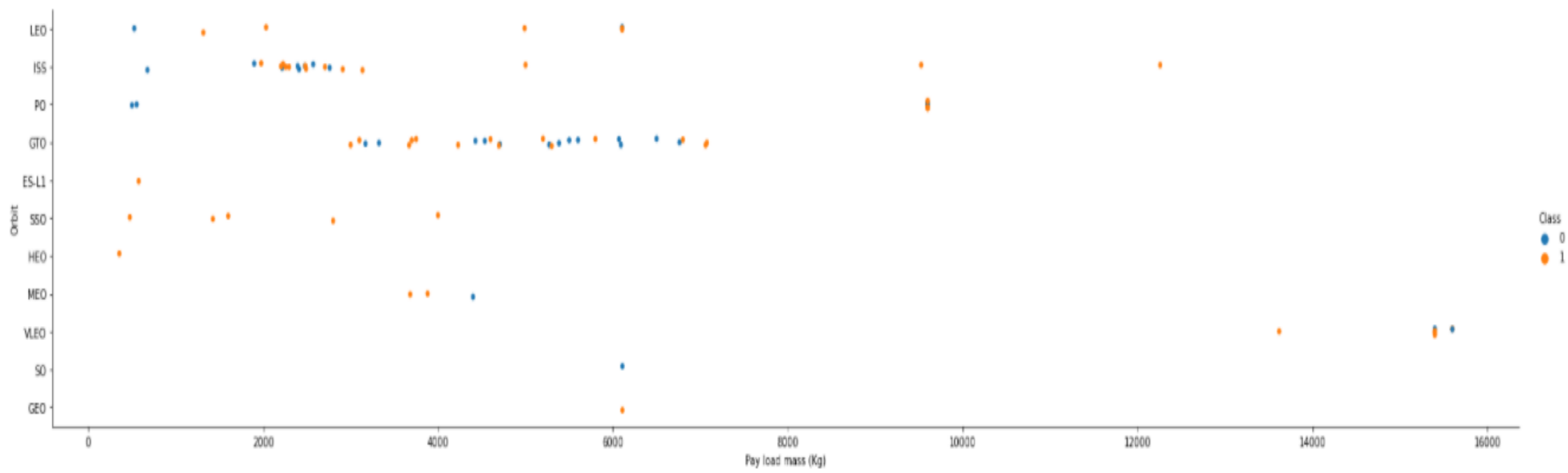
Orbits ES-L1, GEO, HEO, SSO have the highest success rate

Flight Number vs. Orbit Type



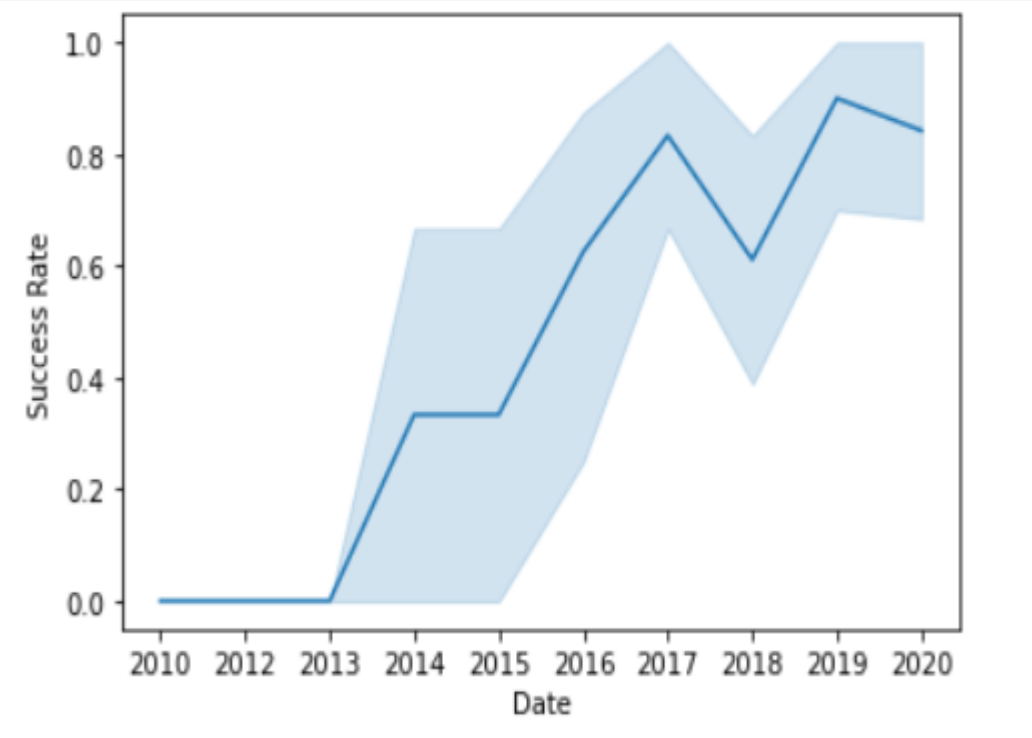
The outcome of a landing does not have any relationship with the orbit type. Even though we have noticed in the previous slide that success rate is higher for certain orbit type, it should be noted that except SSO most of the other orbit types had only one single flight number so it is not a significant breakthrough.

Payload vs. Orbit Type



Except for Orbit type GTO it has been observed that there is a higher success rate of landing when the payload is significantly higher.

Launch Success Yearly Trend



It is observed that the launch success rate has an upward trajectory since 2013 till 2020

All Launch Site Names

```
%sql select distinct(LAUNCH_SITE) from SPACEXDATASET
```

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

The DISTINCT function is used in order to query all the different labels in the “LAUNCH_SITE” column of the table SPACEXDATASET

Launch Site Names Begin with 'CCA'

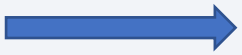
```
%sql SELECT * FROM SPACEXDATASET WHERE launch_site like 'CCA%' LIMIT 5
```

The LIKE keyword uses the wildcard “CCA%” where the % at the end signifies launch_site names that start with “CCA” and may or may not have trailing letters. The limit 5 restricts the output to 5 records

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	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	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)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

```
%sql  select sum(payload_mass__kg_) from SPACEXDATASET where customer = 'NASA (CRS)'
```



1
45596

We have done the SUM function on the column payload_mass__kg for customers who are ‘NASA (CRS)’ only.

Average Payload Mass by F9 v1.1

```
%sql select avg(payload_mass__kg_) from SPACEXDATASET where booster_version = 'F9 v1.1'
```

1
2928

We have done the AVG function on the column payload_mass__kg for booster_version ‘F9 v1.1’ only.

First Successful Ground Landing Date

```
%sql select min(date) from SPACEXDATASET where landing__outcome = 'Success (ground pad)'
```

1
2015-12-22

To find the first successful ground landing date we use the MIN function on the ‘date’ column where landing__outcome is ‘Success (ground pad)’

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql select booster_version from SPACEXDATASET where landing__outcome = 'Success (drone ship)' and payload_mass__kg_>4000 and payload_mas
s__kg_<6000
```

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

In order to find the booster_version with the given conditions we need to use them in the WHERE clause using AND

Total Number of Successful and Failure Mission Outcomes

```
%sql select mission_outcome, count(*) as numbers from SPACEXDATASET group by mission_outcome
```

mission_outcome	numbers
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

In order to find the total number of successful and failure outcomes we need to count the number of occurrences of each type after we have grouped the data using mission_outcome.

Boosters Carried Maximum Payload

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

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

In order to find the booster_version carrying the maximum payload, we first need to find the maximum payload using MAX function in a sub-query. Once we have determined the maximum payload in the sub-query we can easily find the booster_version in the outer query by matching the payload_mass__kg to the maximum payload as obtained in the sub-query.

2015 Launch Records

```
%sql select booster_version, launch_site from SPACEXDATASET where landing__outcome = 'Failure (drone ship)' and DATE like '2015%'
```

booster_version	launch_site
F9 v1.1 B1012	CCAFS LC-40
F9 v1.1 B1015	CCAFS LC-40

We need to give conditions of the landing_outcome and DATE in the WHERE clause using AND in order to find the required outcome. It should be noted that since DATE is in full date format, so in order to find the ones in 2015 only, we are using a LIKE ‘2015%’

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

```
%sql select landing__outcome, count(*) from SPACEXDATASET where DATE between '2010-06-04' and '2017-03-20' group by landing__outcome orde
r by count(*) desc
```

landing__outcome	2
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

In order to find the landing outcomes count between a specified dates we need to use group by landing_outcome. The desired result needs to be ranked, so we need an order by clause on the count function.

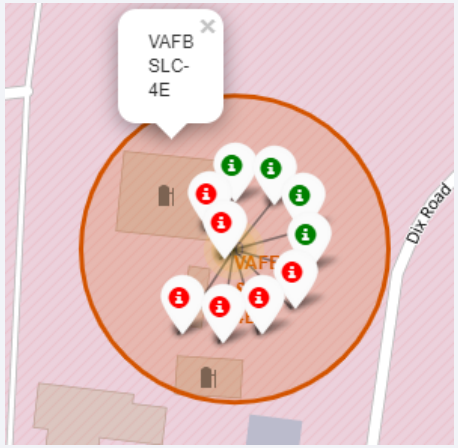
Section 4

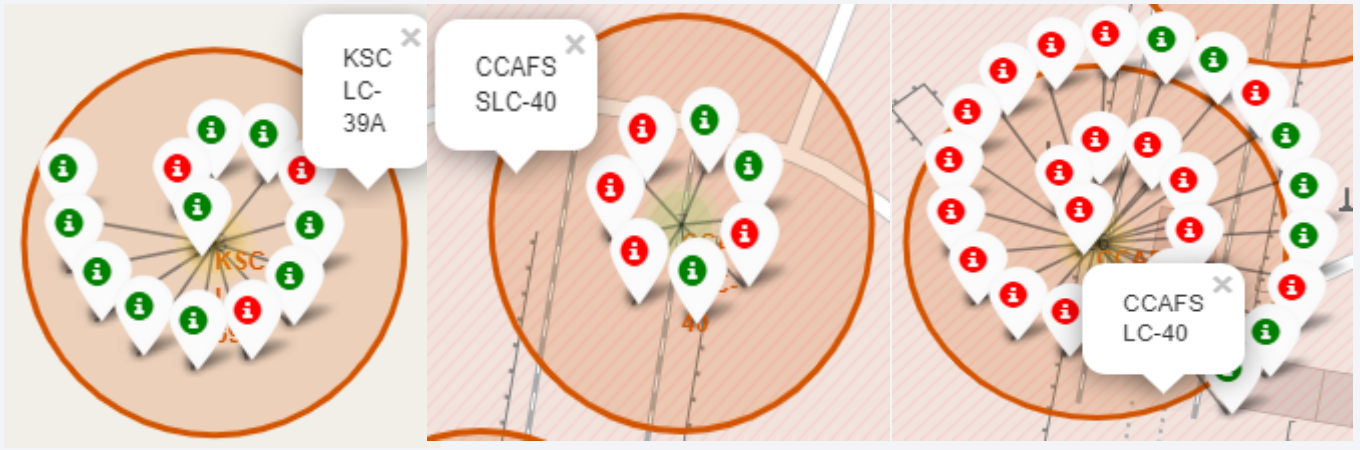
Launch Sites Proximities Analysis

All Launch Sites

The SpaceX launch sites are located in the eastern and western coast of USA

Colour labelled launches outcome

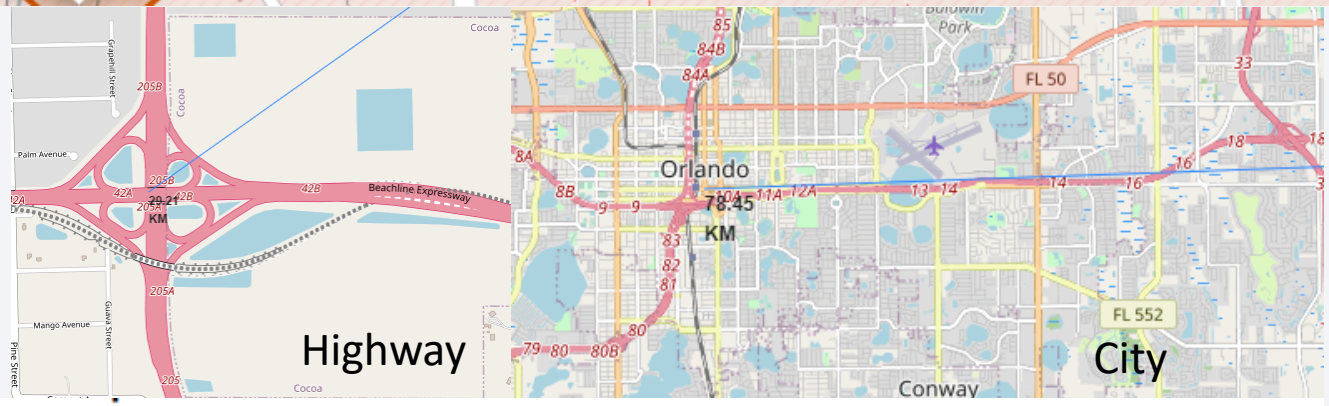




The Green and Red marker in each individual launch sites show the

Launch site distance to Rail, Road, Coast and City

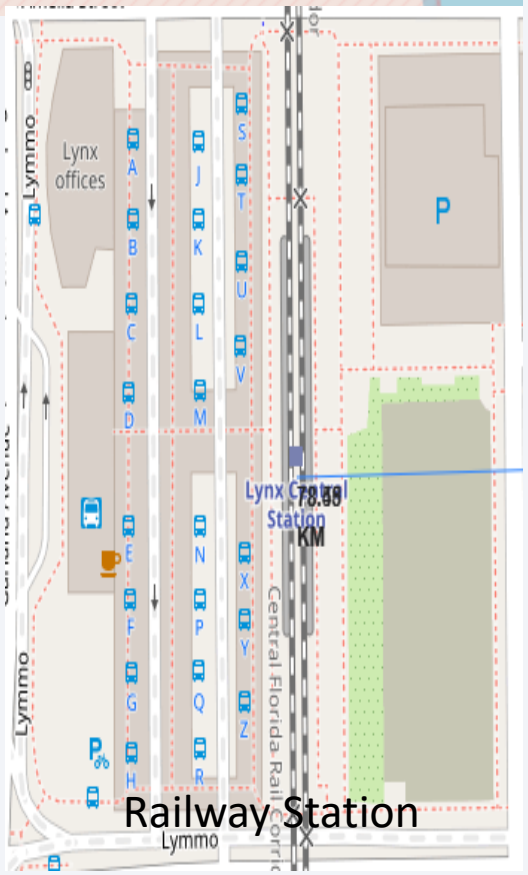
Coast line



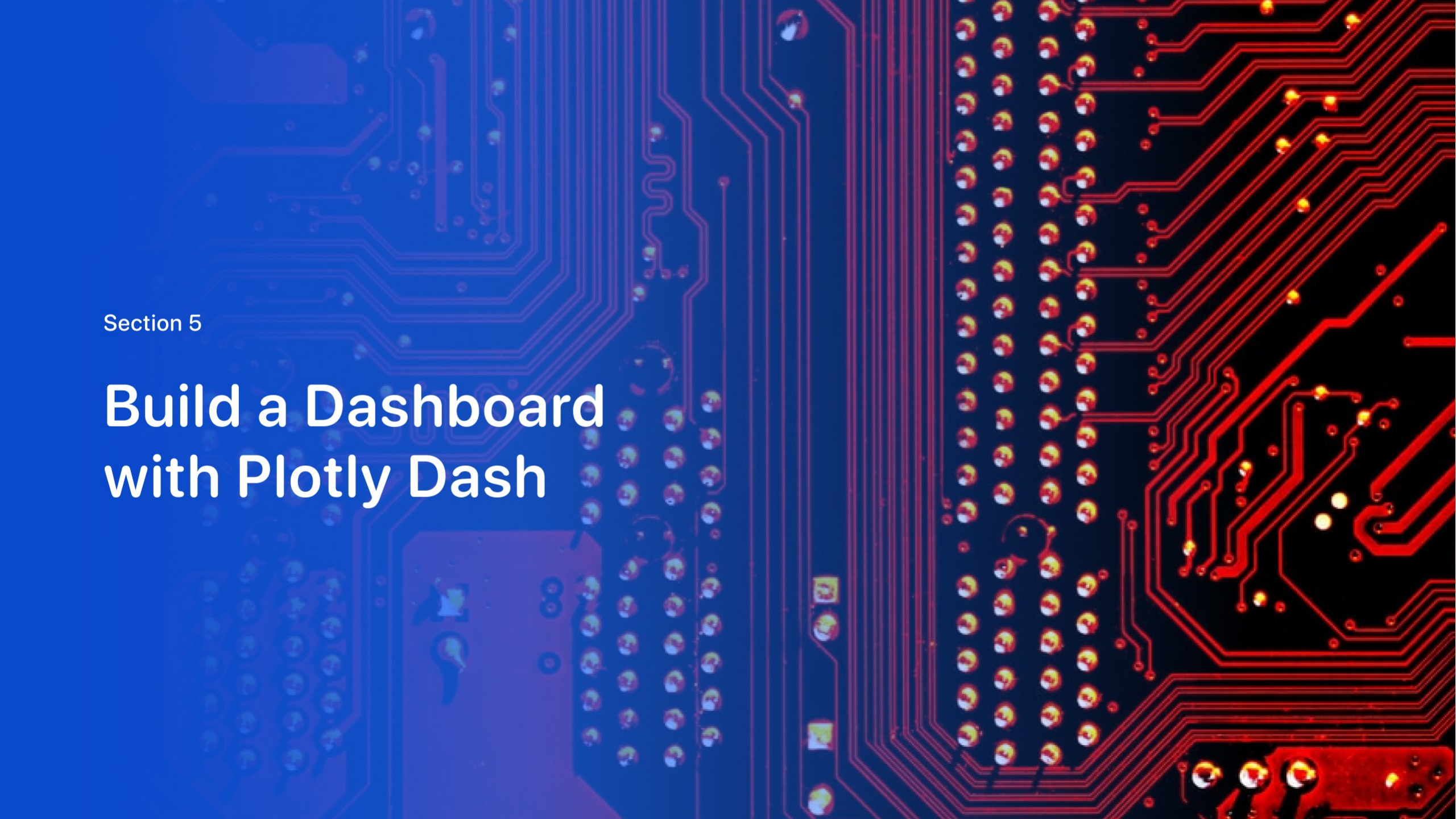
Highway

City

- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes



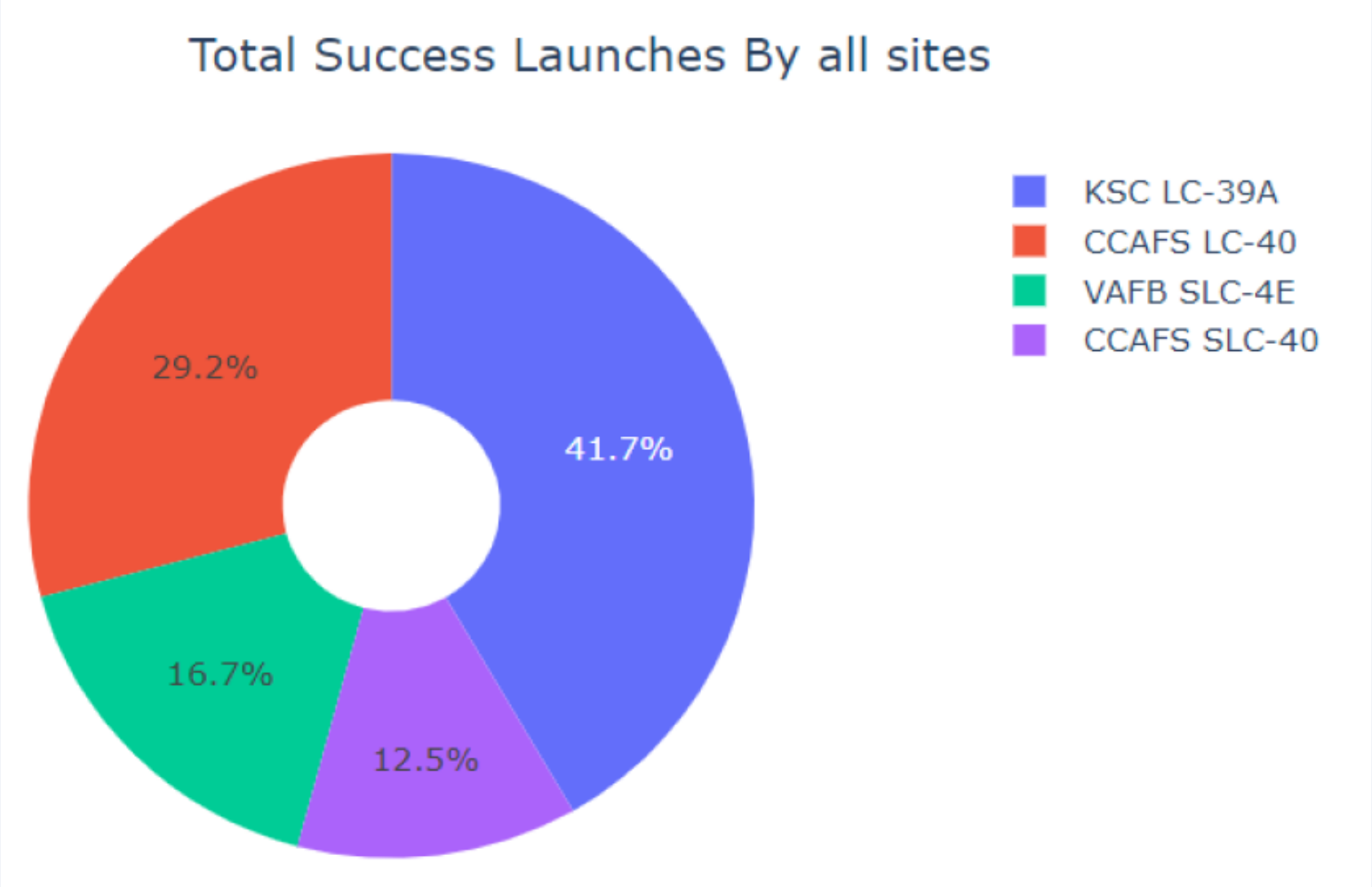
Railway Station



Section 5

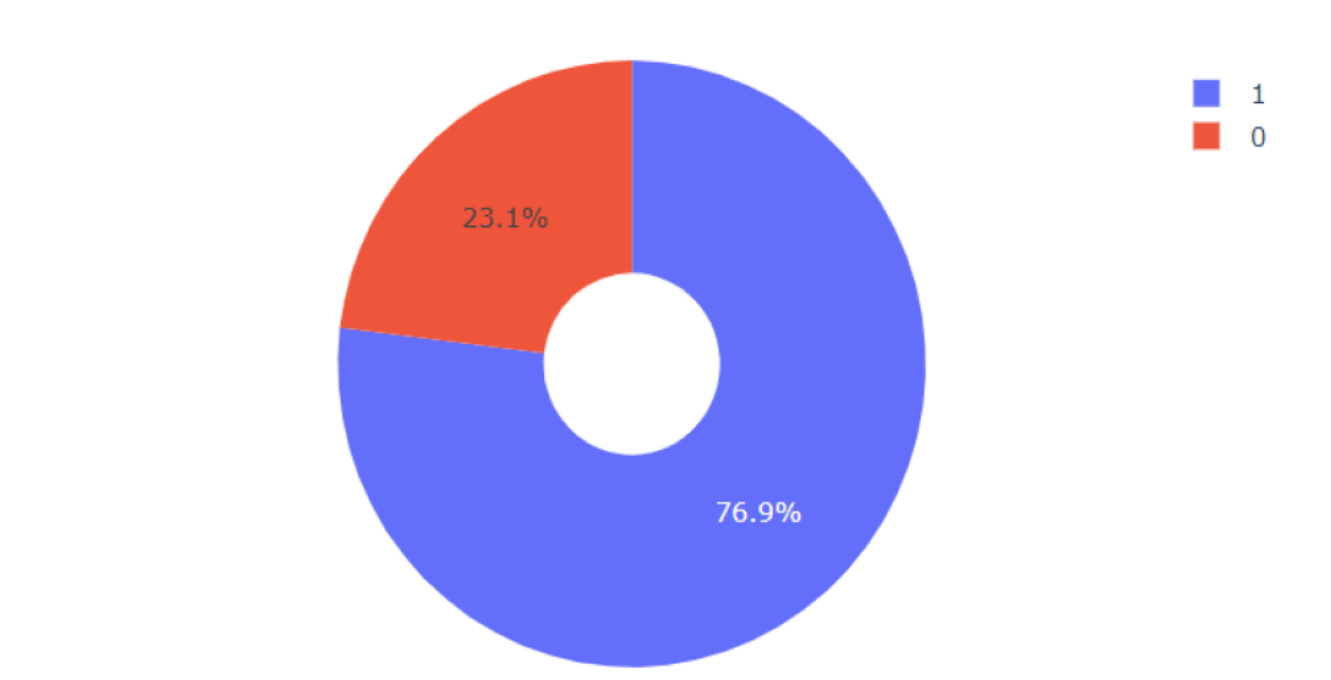
Build a Dashboard with Plotly Dash

Pie chart of the success percentage of each launch site



KSC LC-39A has the best success percentage amongst all the launch sites

Pie chart showing the ratio of the most successful launch site



KSC LC-39A has 76.9 percentage of success out of all its launches.

Section 6

Predictive Analysis (Classification)

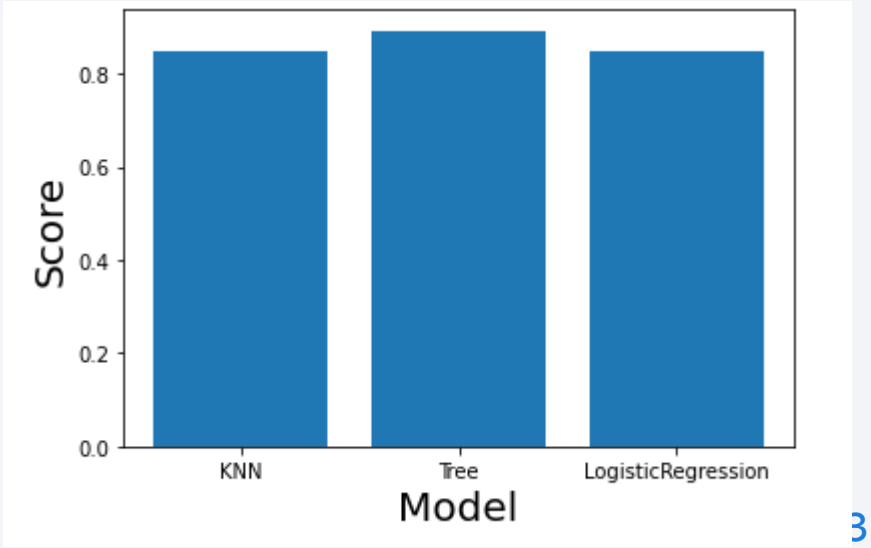
Classification Accuracy

```
algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}

bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Tree':
    print('Best Params is :',tree_cv.best_params_)
if bestalgorithm == 'KNN':
    print('Best Params is :',knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best Params is :',logreg_cv.best_params_)
```

```
Best Algorithm is Tree with a score of 0.8928571428571427
Best Params is : {'criterion': 'gini', 'max_depth': 8, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 2, 'splitter': 'best'}
```

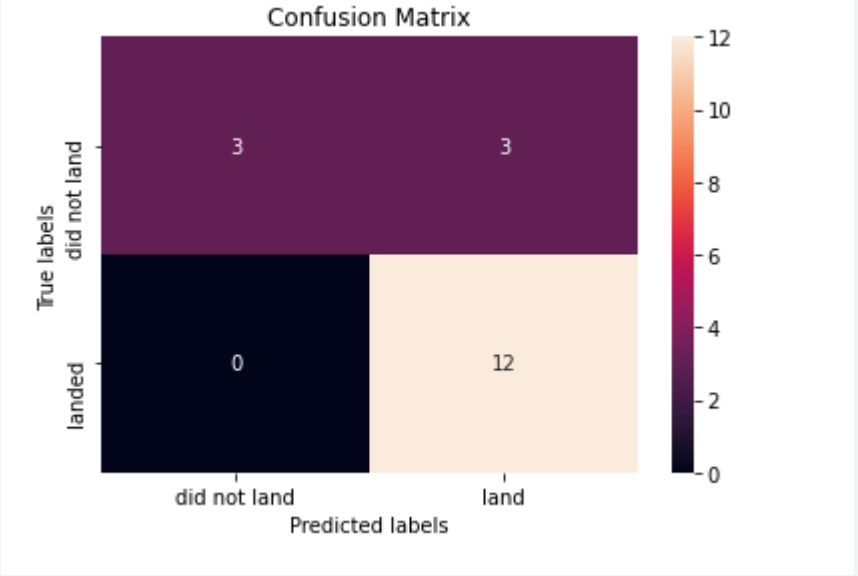
```
{'KNN': 0.8482142857142858,
 'Tree': 0.8928571428571427,
 'LogisticRegression': 0.8464285714285713}
```



As evident from the code as well as the bar plot that Tree is the best classification model

Confusion Matrix

We can see there are 12 occurrences of events where the predicted outcome and the actual outcome actually matches.



Conclusions

- Orbit SSO has the highest success rate of launches
- KSC LC-39A had the most successful launches amongst all the launch sites
- Launch sites have close proximity to coastline
- The success rate of launches is on an upward trajectory since 2013
- The Tree classifier algorithm is the best suited for machine learning for this dataset

Thank you!