IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

David Zinda
21/01/2023

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection via API, Web Scraping
  - Exploratory Data Analytics (EDA) with Data Visualization
  - EDA with SQL
  - Interactive Map with Folium
  - Dashboards with Plotly Dash
  - Predictive Analysis
- Summary of all results
  - Exploratory Data Analysis results
  - Interactive maps and dashboard
  - Predictive results

Introduction

- Background of the project:

The aim of the project is to determine the price of each Falcon 9 launch, by determining whether the first stage will land successfully, and thus can be reused. SpaceX states on its site that the Falcon 9 costs 65 million dollars, compared to other providers whose rockets cost upwards of 165 million. The price difference is driven by the fact SpaceX reuses the first stage.

- Problems we are looking to answer:
  - What are the main characteristics of a successful or failed landing?
  - What factors affect the success or failure of a landing?
  - What other considerations does SpaceX take into account to achieve a successful landing?

Section 1

Methodology

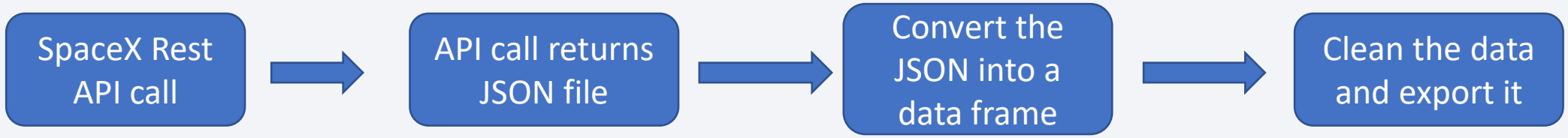
Methodology

Executive Summary

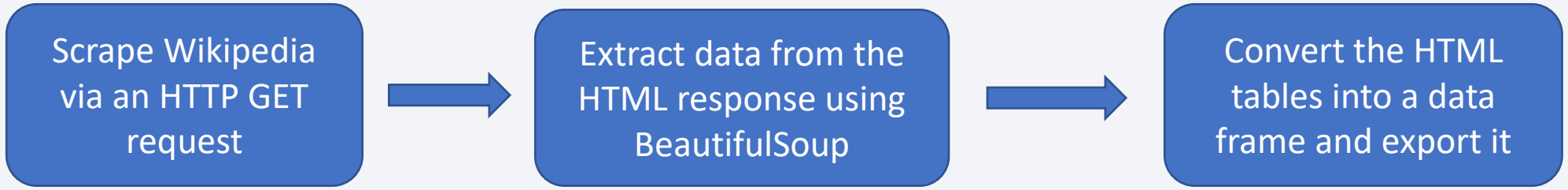
- Data collection methodology:
  - SpaceX launch data gathered from SpaceX REST API
  - Web Scraping from Wikipedia
- Perform data wrangling
  - Converting the data into training labels – 1 for successful landing and 0 for a failure
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

- Data was collected from SpaceX site using a SpaceX Rest API and also through web scrapping Wikipedia
  - The information obtained using the API was rocket/booster name, launching site, mass of the payload etc



  - The information collected through web scrapping were the launch records including launches



Data Collection – SpaceX API

1. Get response from API

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

[7]: response = requests.get(spacex_url)
```

2. Normalize Json and convert the API call result into a dataframe

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

3. Construct dataset using dictionary

```
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}
```

Link to code - [SpaceX API]

4. Create a dataframe from dictionary

```
# Create a data from launch_dict
data = pd.DataFrame({key:pd.Series(value) for key, value in launch_dict.items()})
```

5. Filter dataframe for just Falcon 9 data

```
data_falcon9 = data[data['BoosterVersion']!='Falcon 1']
```

6. Export the data to a csv

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

Data Collection - Scraping

1. Request the Wikipedia page as an HTTP response

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

2. Create BeautifulSoup object

```
soup = BeautifulSoup(response.content, "html.parser")
```

3. Find table that contains the launch records

```
html_tables = soup.findAll('table')
```

4. Extract all column names from the table

```
column_names = []

for th in first_launch_table.find_all('th'):
    name = extract_column_from_header(th)
    if name is not None and len(name) > 0 :
        column_names.append(name)
```

5. Create an empty dictionary with keys as extracted columns

```
launch_dict= dict.fromkeys(column_names)

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']=[]
```

6. Add data to the keys

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowh
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to lau
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
```

7. Create dataframe from dictionary and export data to csv file

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

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

Link to code - [Webscraping]

Data Wrangling

- The dataset includes records of both successful and failed launches. This is shown as string variables, true and false. We need to transform the string variables into categorical variables where 1 implies success and 0 failure

1. 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
```

2. 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
ES-L1     1
HEO       1
SO        1
GEO       1
Name: Orbit, dtype: int64
```

3. Determine number of landing outcomes

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

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

4. Creating a list with o as bad outcome, or 1 otherwise

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

5. Assign the list to a variable

```
df['Class']=landing_class
df[['Class']].head(8)
```

	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

6. Calculate the success rate

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

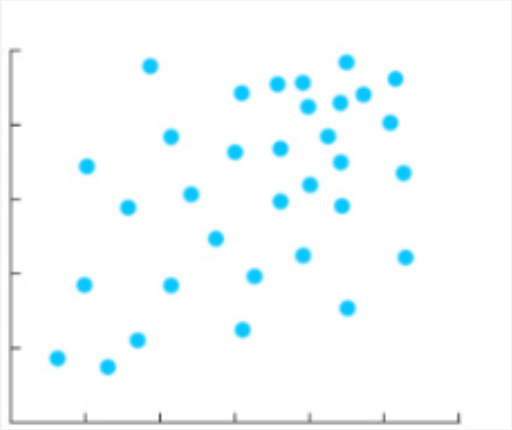
```
0.6666666666666666
```

Link to code – [Data Wrangling]

EDA with Data Visualization

Scatter graphs:

- Flight Number v Payload Mass
- Flight Number v Launch Site
- Payload v Launch Site
- Orbit v Flight Number
- Payload v Orbit
- Orbit v Payload Mass



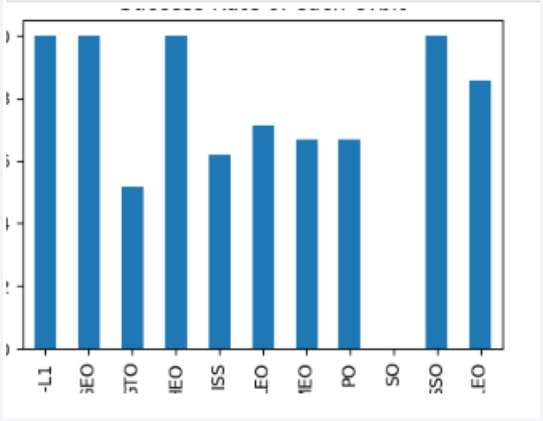
Scatter plots show relationship between variables, i.e. correlation

Link to code – **EDA Data Visualization**

Bar Graph:

- Success rate v Orbit

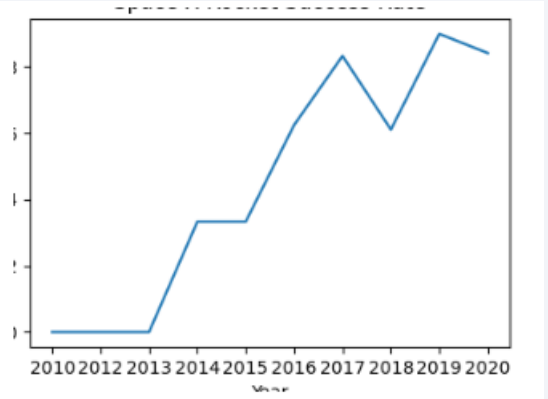
The bar graph shows the relationship between numeric and categoric variables



Line graph:

- Average launch success trend

Line graphs show how data varies over time. A line graph can help to show trend behaviour and make future predictions



EDA with SQL

We performed SQL queries to gather and understand data from the dataset

Link to code – **EDA with SQL**

- Displaying the names of the unique launch sites in the space mission
- Displaying the total payload mass carried by booster launched at different sites
- Displaying average payload mass carried by booster version F9 v1.1
- Listing the date when the first successful landing in ground pad was achieved
- Listing the names of booster which have success in drone ship and have payload mass greater than 4000 but less than 6000
- Listing the number of successful and failed mission outcomes
- Listing the names of boosters that have carried the maximum payload mass
- Listing the records which will display the month names, failed landing outcomes, booster versions by launch site for 2015
- Rank the count of successful landing outcomes between 04/06/2010 and 20/03/2017

Build an Interactive Map with Folium

- The folium map created was centered around NASA Johnson Space Center at Houston, Texas:
  - Read circle at NASA Johnson Space Center, showing the site name.
  - Red circles at each launch site coordinates with labels for each launch site
  - Clustering points to display different information at each launch site coordinates
  - Markers to show successful and failed landings. Green for a successful landing and red for a failed landing.
  - Markers to show distance between launch site to key locations (railway, highway, coastway, city).
- The objects were added to create a better visualization of the different launch sites, and their surroundings, as well as the number of successful and unsuccessful landings at each site. This makes the data much easier to read for the user.

Link to code – Launch site locations

Build a Dashboard with Plotly Dash

- The dropdown dash has a pie chart, rang slider, and different scatter plots.
  - The dropdown allows a user to select a launch site to view related charts
  - The rang slider allows a user to choose the range of payload mass whose charts they wish to visualize/view
  - The pie chart shows the total successes and total failures at each launch site chosen using the dropdown
  - The scatter plots show relationship between different variables, for instance success rate vs payload mass, for whichever site has been selected at the dropdown.

- Link to code – Plotly Dash

Predictive Analysis (Classification)

Loaded and normalized the data

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

```
X = transform.fit_transform(X)
X
```

Split the data into training and testing dataset

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

Create objects for different models, and fit the models using GridSearchCV to find the best 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()

#Insatiate the GridSeachCV object: logreg_cv
logreg_cv = GridSearchCV(lr, parameters, cv=10)

# Fit it to the data
logreg_cv.fit(X_train, Y_train)
```

Models used:

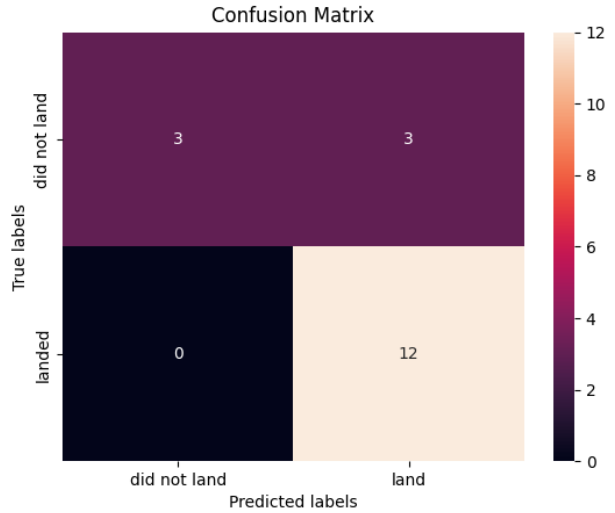
- Logistic regression
- Support Vector Machine
- Decision Tree Classifier
- K Nearest Neighbours (KNN)

Test the model, using the test dataset. I used the accuracy (score method) and confusion matrix

```
knn_cv.score(X_test, Y_test)
```

```
0.8333333333333334
```

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



Find the model that performs the best

Find the method performs best:

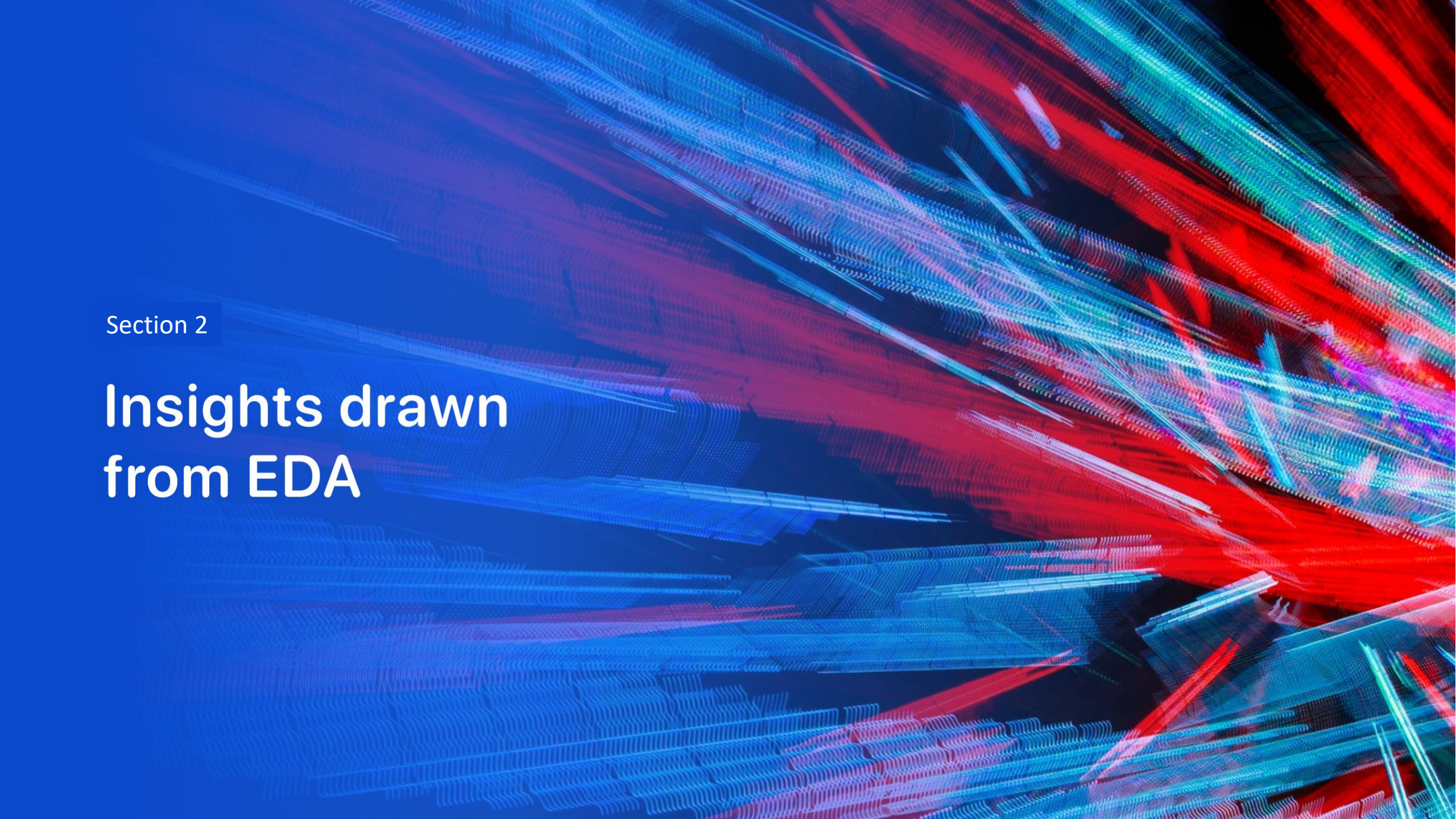
```
predictors = [knn_cv, svm_cv, logreg_cv, tree_cv]
best_predictor = ""
best_result = 0
for predictor in predictors:

    predictor.score(X_test, Y_test)
```

Link to code – [Predictive Analysis]

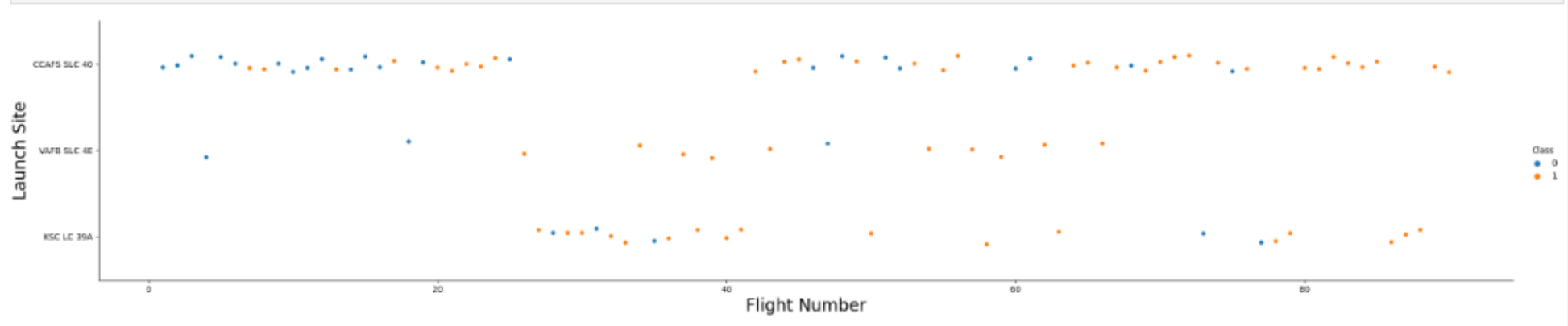
Results

On the following slides is a presentation of the results from the analysis, including:

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

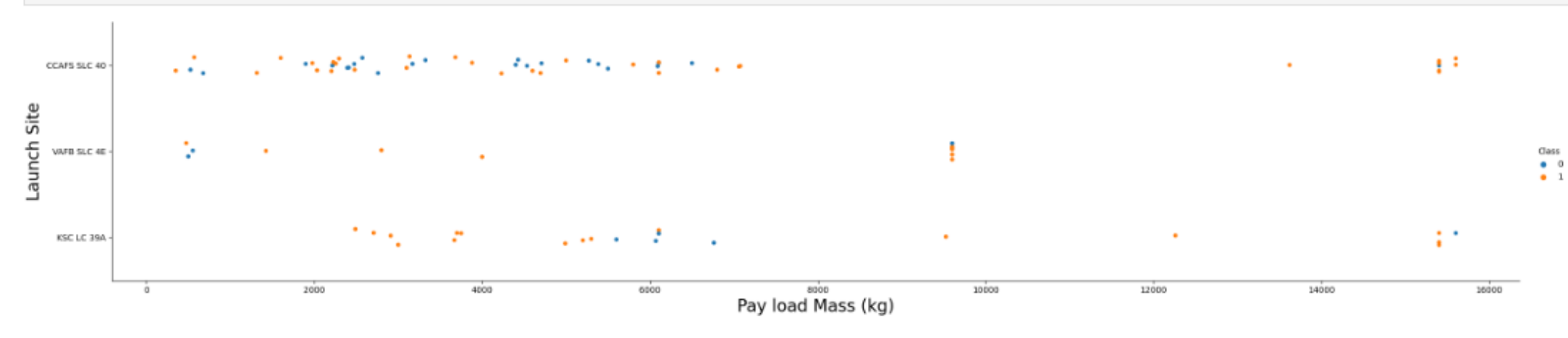

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



For each site, the number of successful launches increases with increase in number of flights

Payload vs. Launch Site

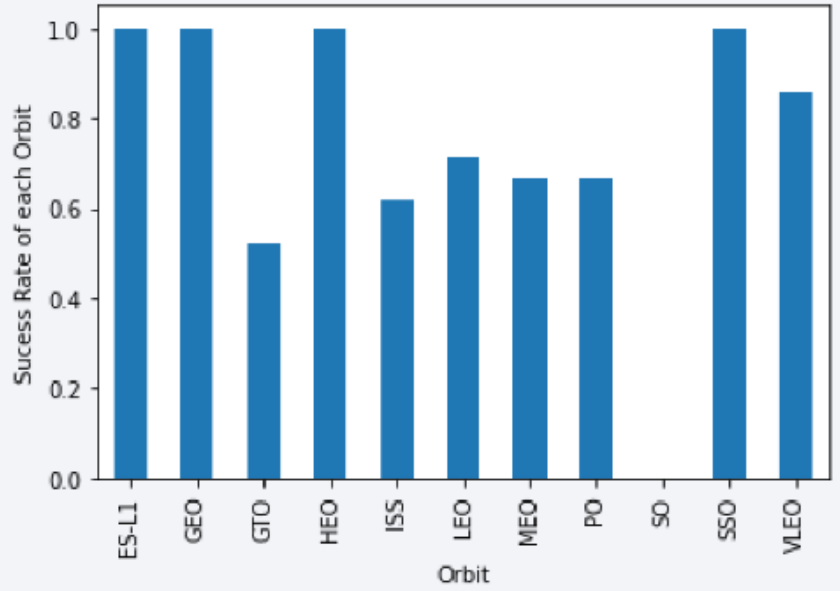


From the plot, we note that one of the sites. VAFB-SLC, there are no rockets launched for heavy payload mass i.e. greater than 10,000.

We also note that the higher the payload, the higher the success rate at each launch site for VAFB SLC-4E and CCAFS LC-40.

KSC LC-39A has a high success rate at both low payload mass and high payload mass, as seen in the plot.

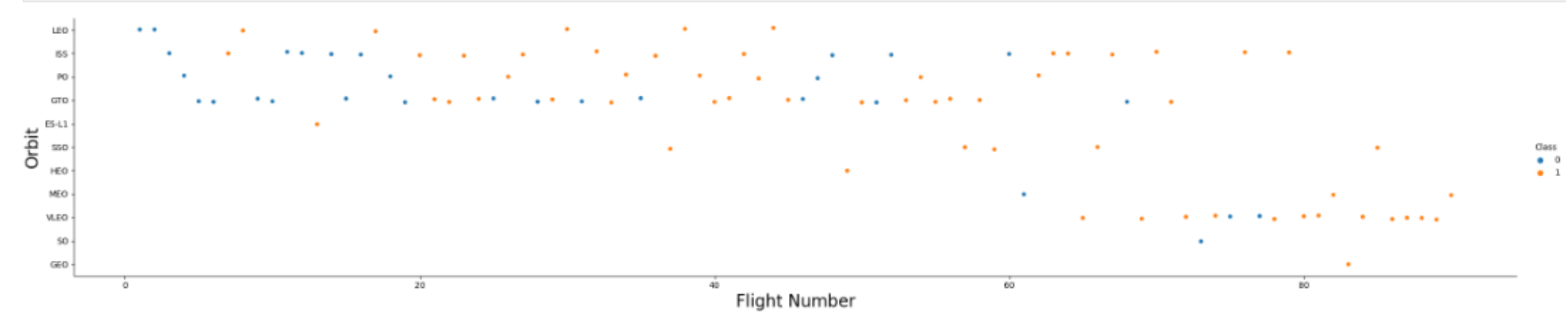
Success Rate vs. Orbit Type



The bar chart shows the success rate of each orbit. The chart clearly shows ES-L1, GEO, HEO and SSO with the highest success rate.

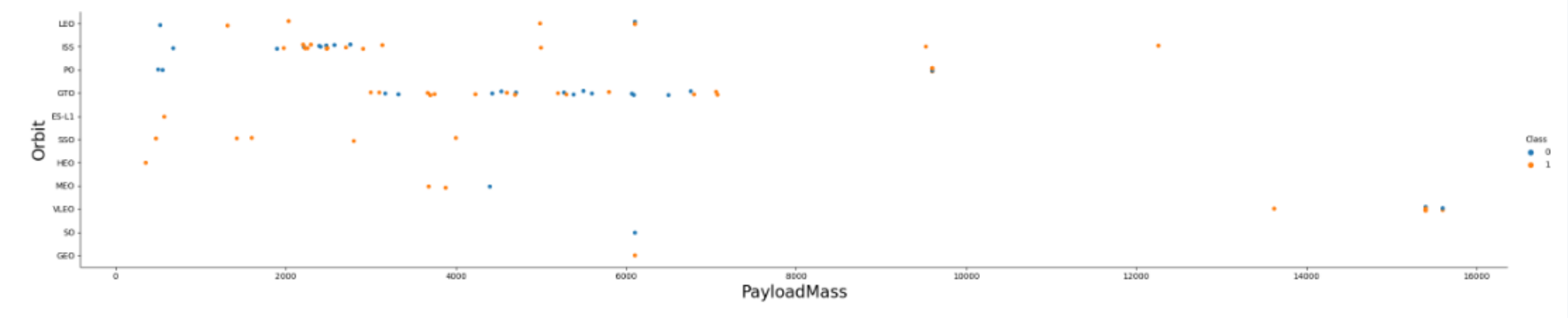
GTO has the lowest success rate.

Flight Number vs. Orbit Type



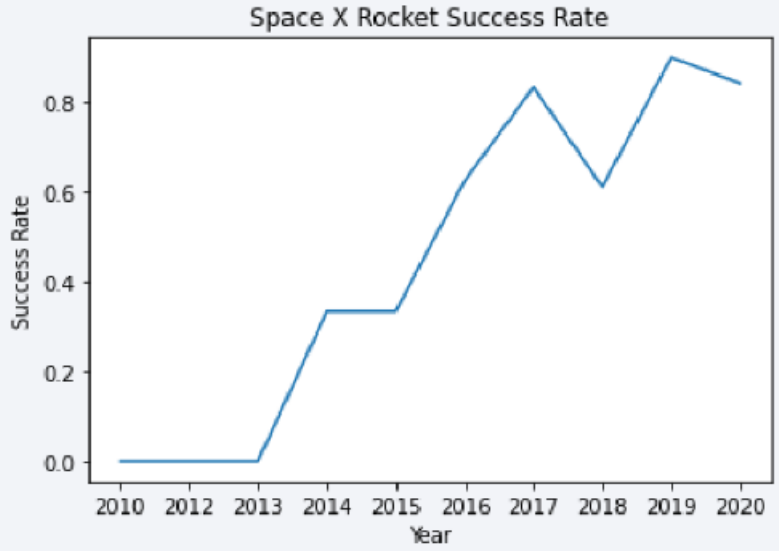
We note that the success rate increases with the number of flights for LEO. For some orbits like GTO, there is no relation between the success rate the number of flights. We can assume that the high success rate of some orbits like SSO and HEO, is due to knowledge learned from previous launches in other orbits.

Payload vs. Orbit Type



With heavy payloads successful landings or positive landing rates are more for LEO and ISS. We also note that lower payload increases the success rate for SSO.. The payload mass can thus have great influence on the success rate of landing in a given orbit

Launch Success Yearly Trend



We can see an increase in launch success since 2013 for SpaceX rocket. This implies the knowledge from previous launches could be helping SpaceX improve the success rate of future launches.

All Launch Site Names

```
%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

The use of DISTINCT in the query allows to remove duplicates from the result. Specifying LAUNCH_SITE in the query allows to return on records from the launch site colum.

Launch Site Names Begin with 'CCA'

```
%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

Using “where” and “LIKE” filters all launch sites in the specified “Luanch_Site” column that contain the substring CCA. “LIMIT 5” limits the returned results to 5 rows.

Total Payload Mass

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Customer = 'NASA (CRS)'
```

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

SUM(PAYLOAD_MASS__KG_)
45596

The query returns the total payload mass of all sites where the customer is NASA (CRS)

Average Payload Mass by F9 v1.1

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Booster_Version LIKE 'F9 v1.0%'
```

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

AVG(PAYLOAD_MASS__KG_)
340.4

The query returns the average payload mass where the booster version contains the substring F9 v1.1

First Successful Ground Landing Date

```
%sql SELECT MIN(Date) FROM SPACEXTBL WHERE "Landing _Outcome" = 'Success (ground pad)'
```

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

MIN(Date)
01-05-2017

The query returns the first successful landing on a ground pad. The “WHERE” clause filters for the landing outcome column for records containing the specified landing outcome, and the MIN(DATE) function returns the first date matching the criteria.

Successful Drone Ship Landing with Payload between 4000 and 6000

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

```
%sql SELECT Booster_Version FROM SPACEXTBL WHERE "LANDING _OUTCOME" = 'Success (drone ship)' AND 4000 < PAYLOAD_MASS__KG_ < 6000
```

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

Booster_Version
F9 FT B1021.1
F9 FT B1022
F9 FT B1023.1
F9 FT B1026
F9 FT B1029.1
F9 FT B1021.2
F9 FT B1029.2
F9 FT B1036.1
F9 FT B1038.1
F9 B4 B1041.1
F9 FT B1031.2
F9 B4 B1042.1
F9 B4 B1045.1
F9 B5 B1046.1

The query returns the booster version where the landing was successful, and the payload mass lies in the range 4000 to 6000 kg. The WHERE and AND clauses are used to filter the data set.

Total Number of Successful and Failure Mission Outcomes

```
%sql SELECT (SELECT COUNT("MISSION_OUTCOME") FROM SPACEXTBL WHERE "MISSION_OUTCOME" LIKE '%Success%') AS SUCCESS, \
(SELECT COUNT("MISSION_OUTCOME") FROM SPACEXTBL WHERE "MISSION_OUTCOME" LIKE '%Failure%') AS FAILURE
```

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

SUCCESS	FAILURE
100	1

This query contains two subqueries. The first sub query returns the number of successful mission outcomes in the dataset. The second subquery returns the count of failed mission outcomes within the dataset.

The Where and Like clauses, as we have encountered before, filter the data based on the specified parameters. i.e. success and failure.

Boosters Carried Maximum Payload

```
%sql SELECT DISTINCT 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

We use a subquery to filter data by returning only the heaviest payload mass with the MAX function. The main query uses the sub query results and returns unique booster versions (DISTINCT) that fit the criteria.

2015 Launch Records

```
%sql SELECT substr("DATE", 4, 2) AS MONTH, "BOOSTER_VERSION", "LAUNCH_SITE" FROM SPACEXTBL\
WHERE "LANDING _OUTCOME" = 'Failure (drone ship)' and substr("DATE",7,4) = '2015'
```

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

MONTH	Booster_Version	Launch_Site
01	F9 v1.1 B1012	CCAFS LC-40
04	F9 v1.1 B1015	CCAFS LC-40

Returns the month, booster version, and the launch site where the landing was unsuccessful for the year 2015. SQLLite does not support monthnames. So we use substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.

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

```
%sql SELECT "LANDING _OUTCOME", COUNT("LANDING _OUTCOME") FROM SPACEXTBL\
WHERE "DATE" >= '04-06-2010' and "DATE" <= '20-03-2017' and "LANDING _OUTCOME" LIKE '%Success%'\
GROUP BY "LANDING _OUTCOME" \
ORDER BY COUNT("LANDING _OUTCOME") DESC ;
```

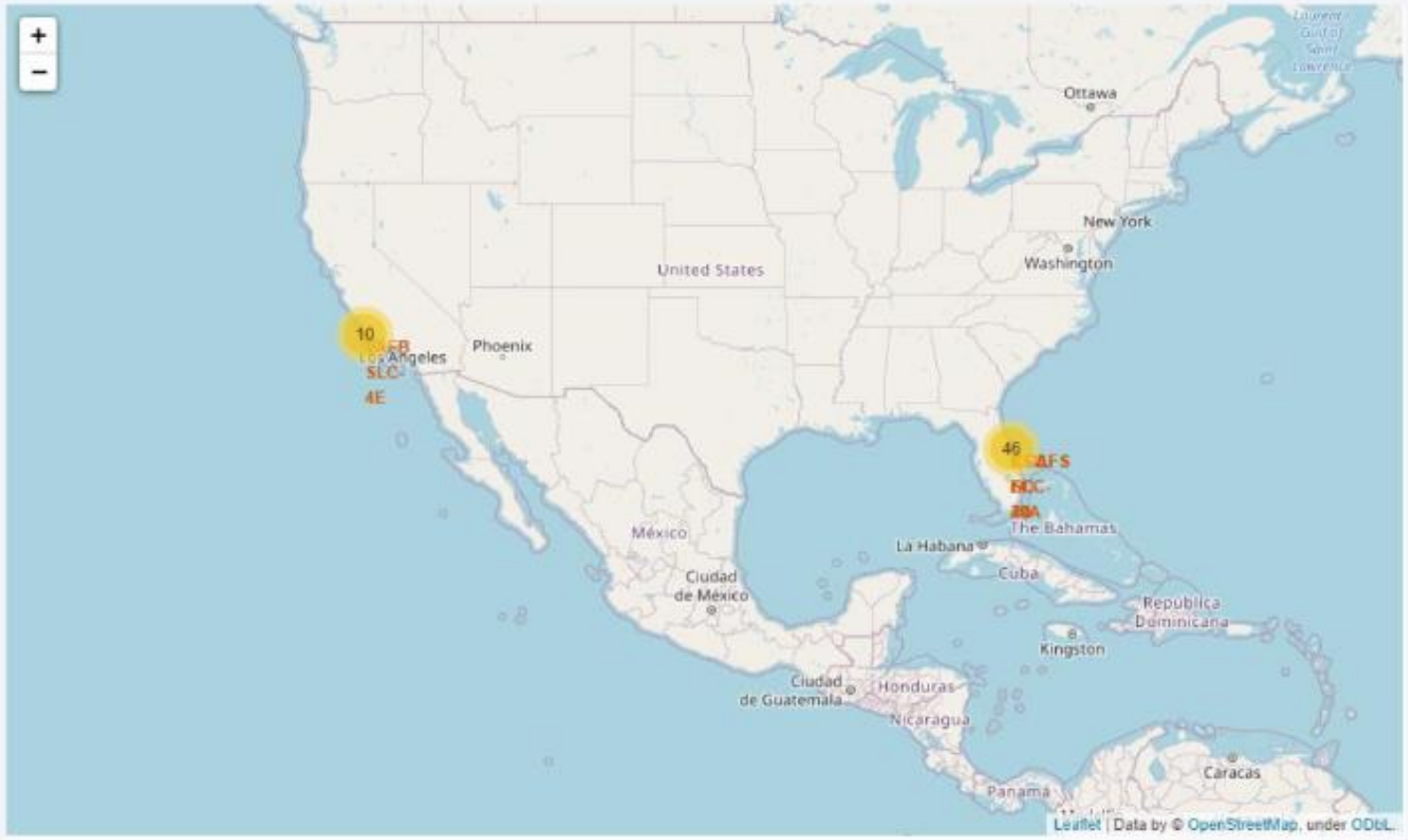
```
 * sqlite:///my_data1.db
Done.
```

Landing _Outcome	COUNT("LANDING _OUTCOME")
Success	20
Success (drone ship)	8
Success (ground pad)	6

The query returns the number of landing outcomes for each successful mission between the specified dates. The GROUP BY function groups the results by landing outcome and the DESC clause shows the results in descending order by number of outcomes.

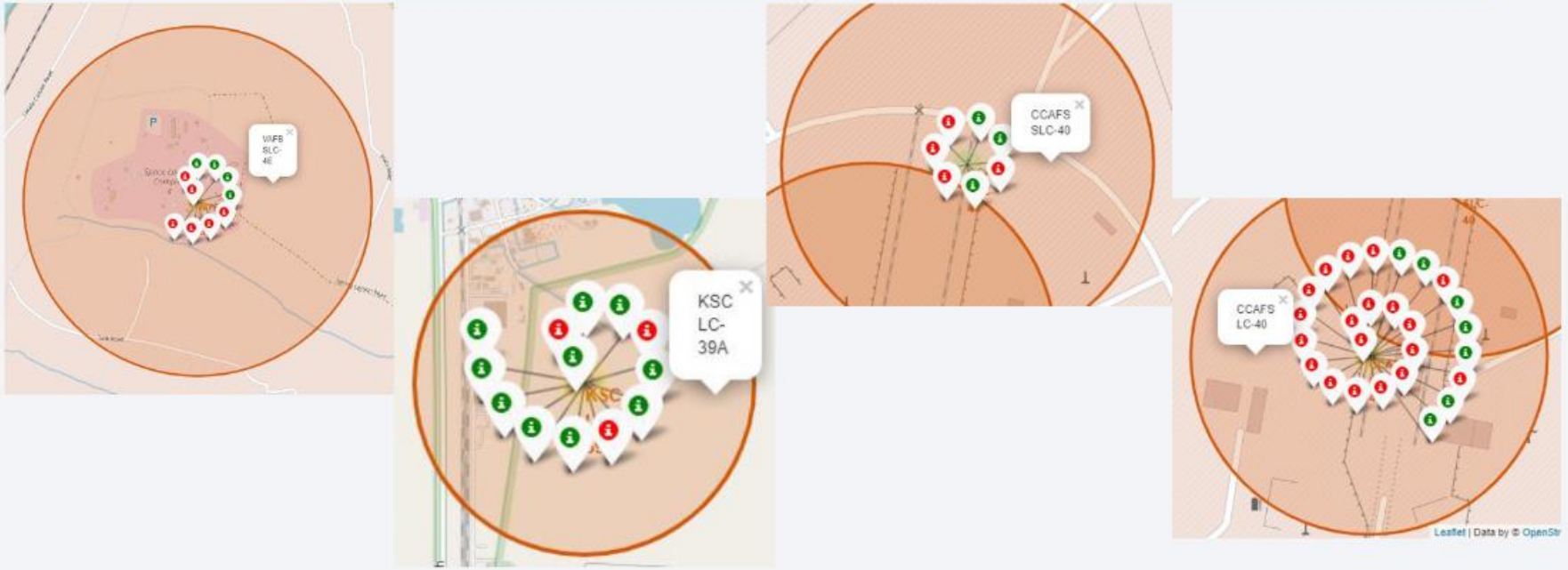

Section 3
Launch Sites
Proximities Analysis

Folium Map – Ground Stations



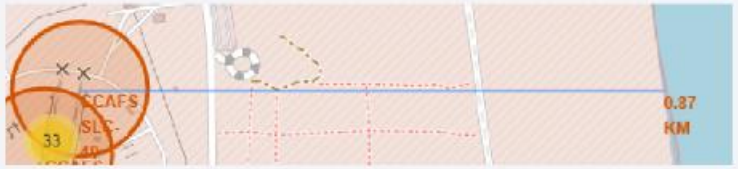
- The SpaceX launch sites are located on the cost of the United States, in Florida and Los Angeles

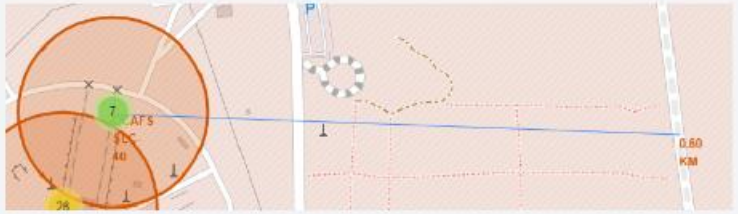
Folium map – Launch Sites Magnified



The green markers represent successful launches and the red markers represent unsuccessful launches. We note that KSC LC-39A has a higher launch success rate.

Folium map – Distance between CCAFS SLC-40 and its prximities



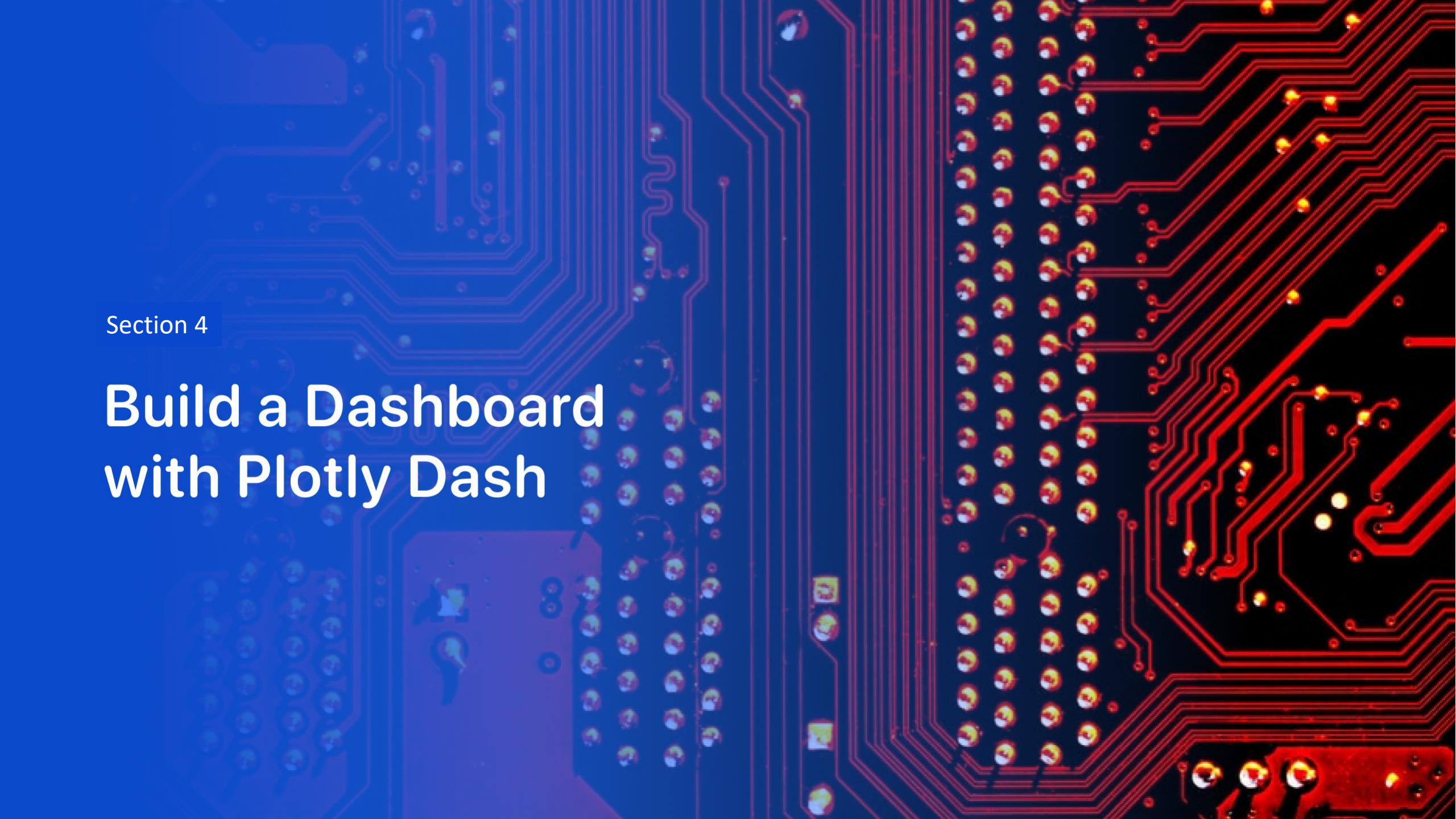


Is CCAFS SLC-40 in close proximity to railways? – Yes

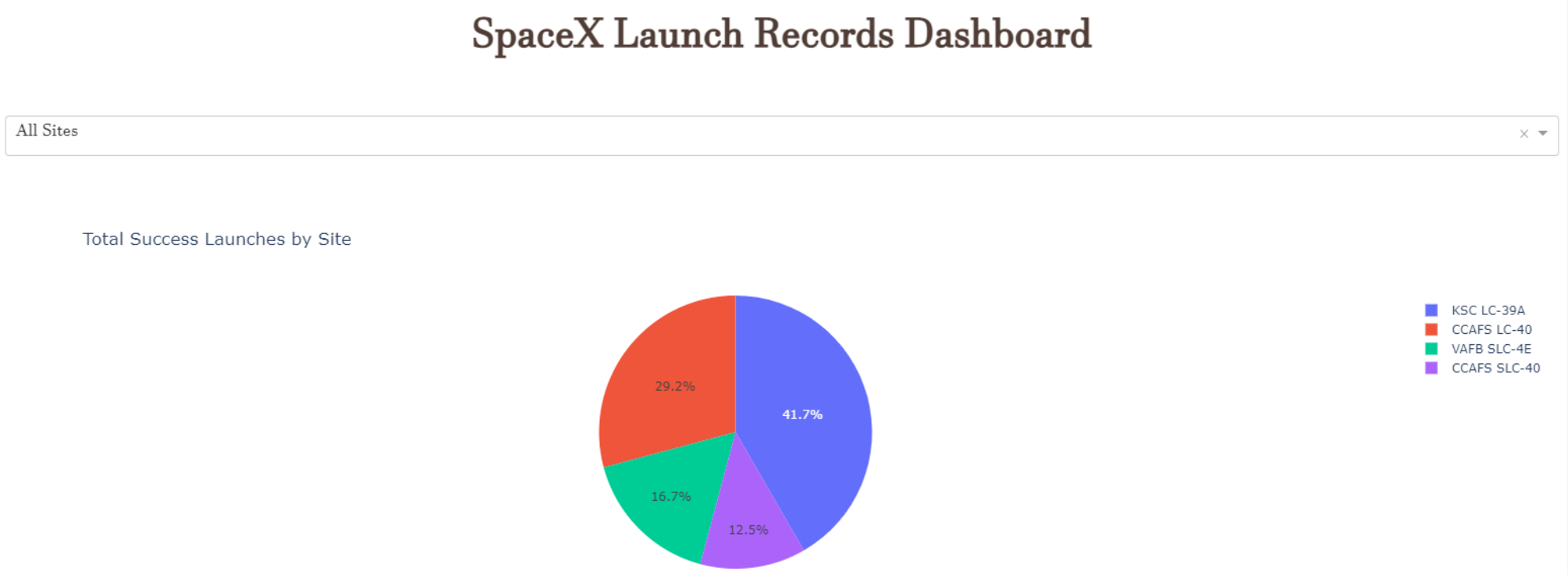
Is CCAFS SLC-40 in close proximity to highways? – Yes

Is CCAFS SLC-40 in close proximity to the coastline? – Yes

Is CCAFS SLC-40 in close proximity to cities? - No

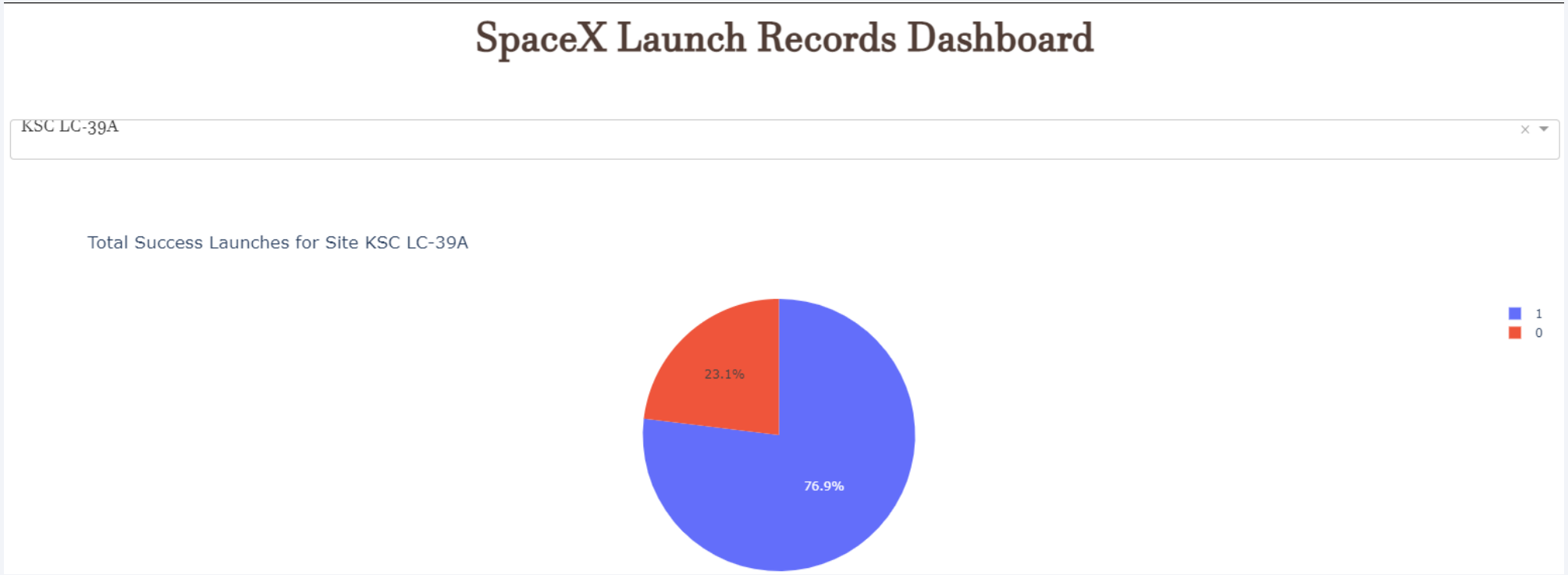

Section 4
Build a Dashboard with Plotly Dash

Success Rate for All Sites



We can see from the pie chart that KSC LC-39A has the highest success rate of all launch sites, and CCAFS SLC-40 has the lowest.

Total success rate for KSC LC-39A



The chart shows that KSC LC-39A had 76.9% of all it's launches have successful outcomes.

Comparing success rate for all sites with different payloads

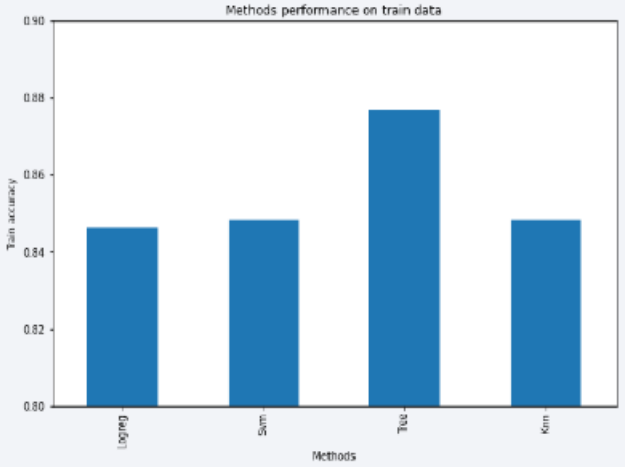
We note that rockers with lower payloads have better success rates compared to rockets with higher payloads

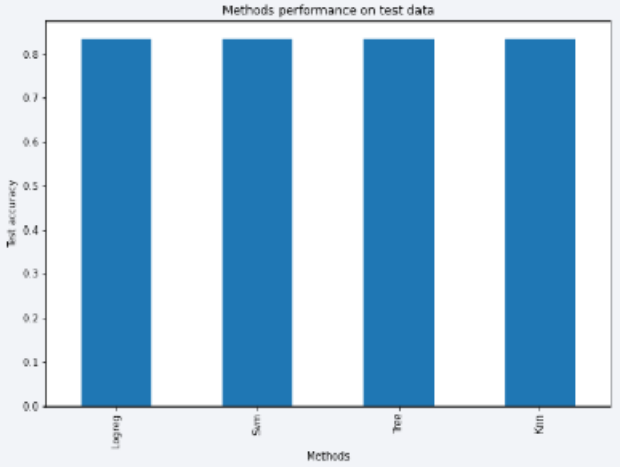
Section 5

Predictive Analysis (Classification)

Classification Accuracy

	Accuracy Train	Accuracy Test
Tree	0.876786	0.833333
Knn	0.848214	0.833333
Svm	0.848214	0.833333
Logreg	0.846429	0.833333





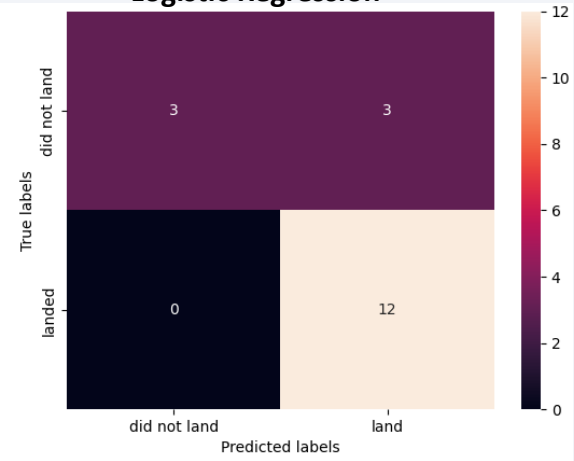
For accuracy, all models performed the same. We could get more data to train or test the models, but we have chosen decision tree as our preferred mode.

Decision tree best parameters

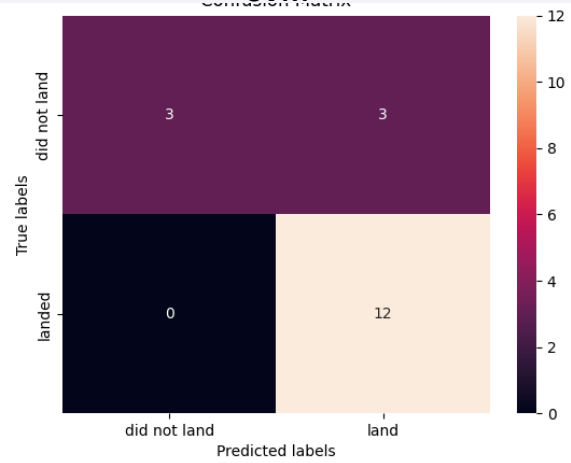
```
tuned hyperparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 12, 'max_features': 'sqrt', 'min_samples_leaf':
4, 'min_samples_split': 2, 'splitter': 'random'}
```

Confusion Matrix

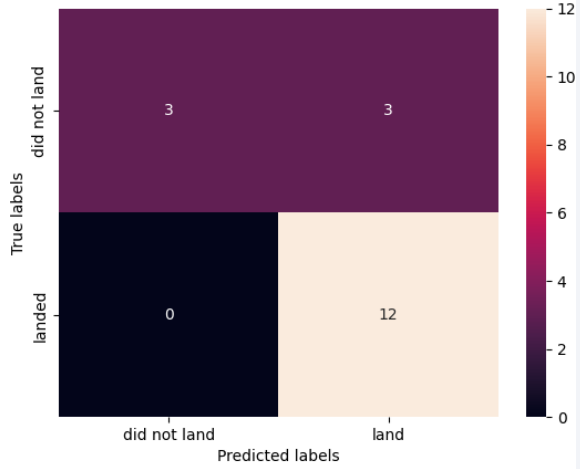
Logistic Regression



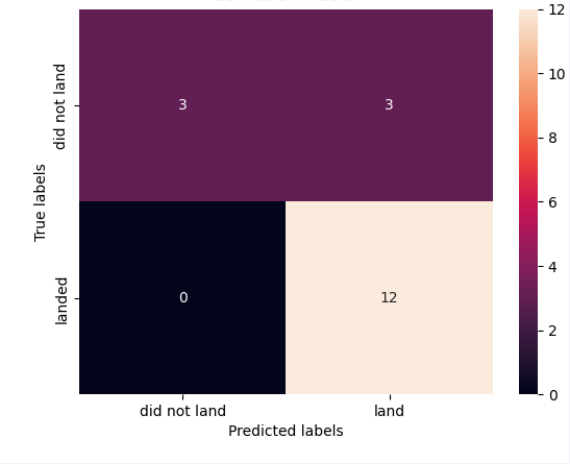
SVM



Decision Tree



KNN



As the test accuracy is the same across all the models, the confusion matrix is identical as can be seen in the screenshots.

The main issue with the models are the false positives.

Conclusions

- The success of a mission depends on a number of factors, including, the launch site, the orbit, number of previous launches, and the payload mass for each launch.
- It is clear that SpaceX has used lessons learned from previous launches, factoring in different factors that led to a successful launch, to improve future launches. This is clear from the yearly increase in the success rate graph.
- KSC LC-39A has the highest success rate among all the launching sites, and this should be used as a model site to check what factors influence the success or failure of launches at the site
- The orbits with the best success rates are GEO, HEO, SSO and ES-L1. This should inform the user where to launch the rockets.
- Depending on the orbit, and launch site, the payload mass can be a criterion to take into account for the success of a mission. Some orbits require a lighter load, while other a heavier load. But generally, low weighted payloads perform better than the heavier loads.
- We note, however, that different sites behave differently with the different factors. This could imply that we need more data to better understand what influences the success or failure of each launch.

Thank you!