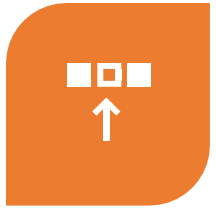
Winning Space Race with Data Science

Chetan Gowda
2025

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 Dash

-Predictive analysis (Classification)

- Summary of all results

-Exploratory data analysis results

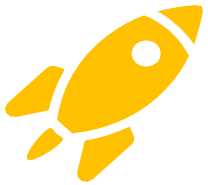
-Interactive analytics demo in screenshots

-Predictive analysis results

Introduction

Project background and context:

We will 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 successfully, 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 a successful rocket landing?

The correlation of rocket variables with the success rate of a rocket landing.

What conditions does SpaceX have to achieve to ensure the highest success rocket landing rate.

Section 1

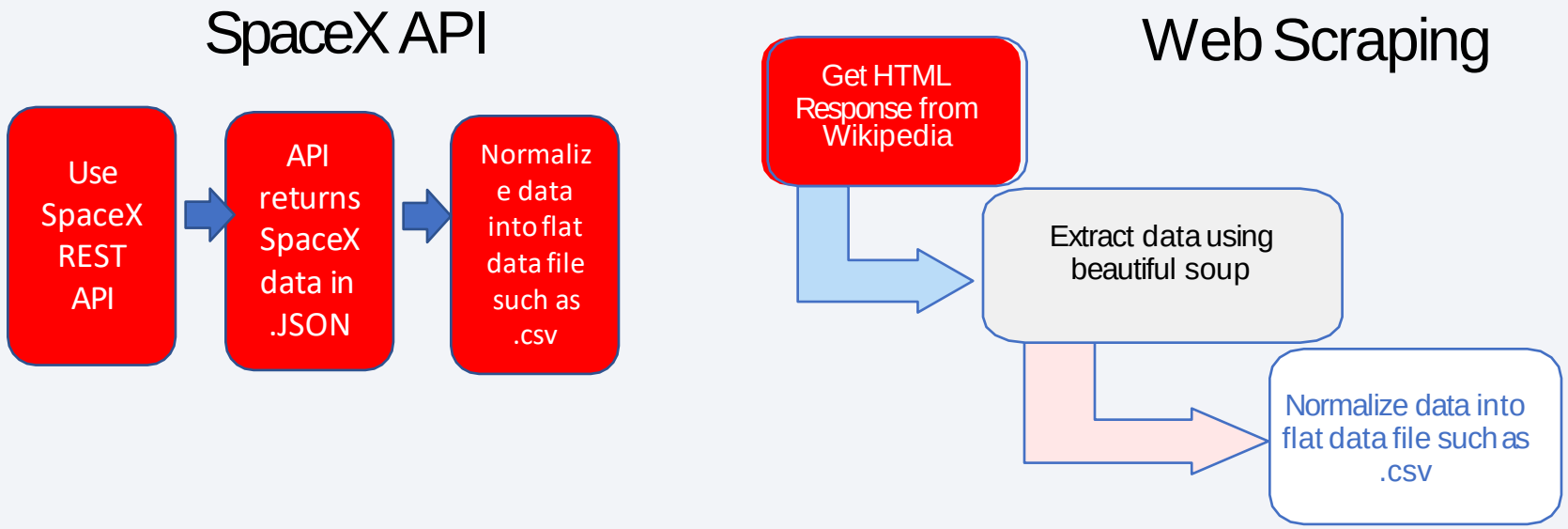
Methodology

Methodology

- Data collection methodology:
  - SpaceX Rest API
  - (Web Scrapping) from Wikipedia
- Perform data wrangling
  - One Hot Encoding data fields for Machine Learning and dropping irrelevant columns
- Perform exploratory data analysis (EDA) using visualization and SQL
- Plotting : Scatter Graphs, Bar Graphs to examine the relationships between variables and show trends.
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models and find the best model

Data Collection

- Gather data from SpaceX Rest API
- Web Scraping Falcon 9 Launch Data from Wikipedia



Data Collection – SpaceX API

simplified flow chart

1 .Getting Response from API

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

2. Converting Response to a .json file

```
response = requests.get(static_json_url).json()
data = pd.json_normalize(response)
```

3. Apply custom functions to clean data

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

```
getBoosterVersion(data)
```

4. Assign 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. Filter dataframe and export to flat file (.csv)

```
data_falcon9 = df.loc[df['BoosterVersion']!="Falcon 1"]
```

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

8

Data Collection - Scraping

simplified flow chart

1 .Getting Response from HTML

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

2. Creating BeautifulSoup Object

```
soup = BeautifulSoup(page.text, 'html.parser')
```

3. Finding tables

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

4. Getting column names

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

5. Creation of dictionary

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
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'] = []
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

6. Appending data to keys (refer) to notebook block 12

```
In [12]: extracted_row = 0
         #Extract each table
         for table_number,table in enumerate(
            # get table row
             for rows in table.find_all("tr")
                 #check to see if first table
```

7. Converting dictionary to dataframe

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

8. Dataframe to .CSV

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


9

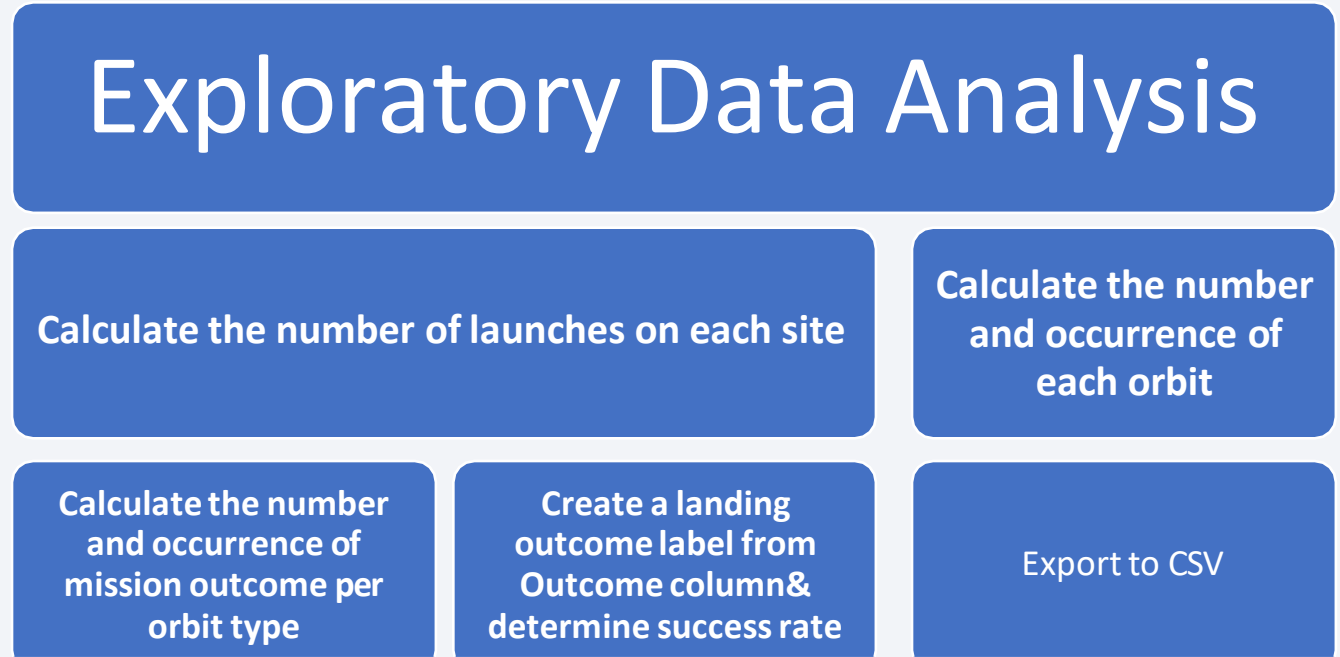
Data Wrangling

Introduction

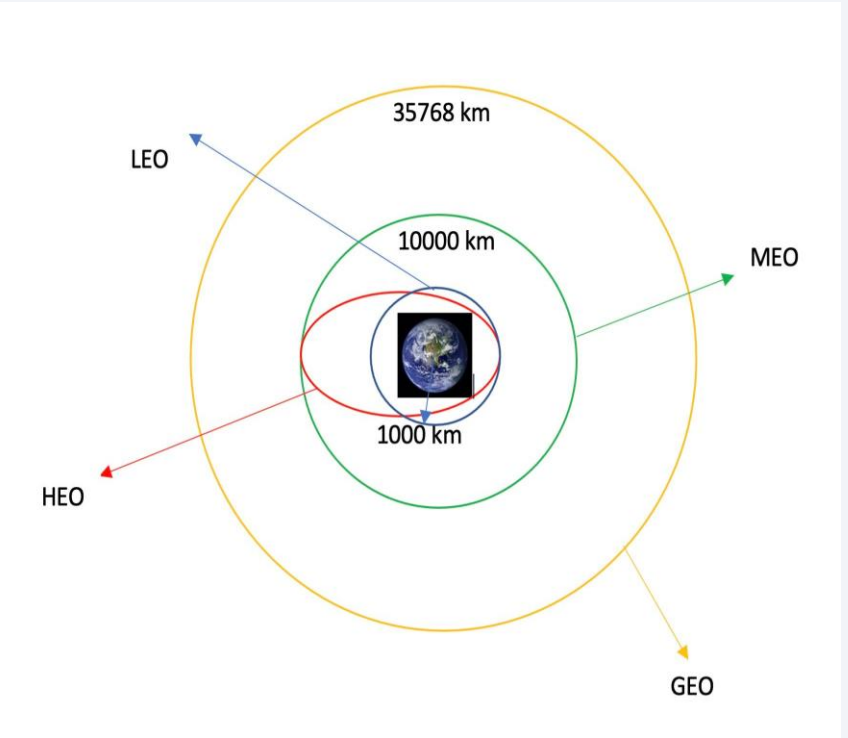
In the data set, there are several different cases where the booster did not land successfully. Sometimes a landing was attempted but failed due to an accident; for example, True Ocean means the mission outcome was successfully landed to a specific region of the ocean while False Ocean means the mission outcome was unsuccessfully landed to a specific region of the ocean. True RTLS means the mission outcome was successfully landed to a ground pad False RTLS means the mission outcome was unsuccessfully landed to a ground pad.True ASDS means the mission outcome was successfully landed on a drone ship False ASDS means the mission outcome was unsuccessfully landed on a drone ship.

We mainly convert those outcomes into Training Labels with 1 means the booster successfully landed 0 means it was unsuccessful.

- Process flowcharts



Each launch aims to an dedicated orbit, and here are some common orbit types



EDA with Data Visualization

Scatterplot:

Scatter plot is a great tool to depict correlation.

- **FlightNumber vs. PayloadMass**
- **FlightNumber vs LaunchSite**
- **FlightNumber vs Orbit type**
- **Payload vs Launch Site**
- **Payload vs Orbit type**

Bar Chart:

Bar chart compares the measure of categorical dimension

- **Success rate vs 0orbit type**

Line Chart:

display trends and developments of numeric data over time.

- **Launch success yearly trend**

EDA with SQL

Summary of the SQL queries performed

- **Display the names of the unique launch sites in the space mission**
- **Display 5 records where launch sites begin with the string 'CCA'**
- **Display the total payload mass carried by boosters launched by NASA (CRS)**
- **Display average payload mass carried by booster version F9 v1.1**
- **List the date when the first succesful landing outcome in ground pad was acheived.**
- **List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000**
- **List the total number of successful and failure mission outcomes**
- **List the names of the booster versions which have carried the maximum payload mass**
- **List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.**
- **Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order**

Build an Interactive Map with Folium

- **Mark all launch sites on an interactive map.**

Used latitude and longitude Coordinates of each launch site and added a *Circle Marker around each site and a label of the name of the site.*

- **Mark the success/failed launches for each site on the map**

assigned the dataframe launch_outcomes(failures, successes) to *classes 0 and 1*

with Green and Red markers on the map in a MarkerCluster()

- **Calculate the distances between a launch site to its proximities**

Used Haversine's formula to calculate the distance from the Launch Site to its proximities such as highway, roadway, coastlines. Lines are drawn on the map to measure distance to these proximities.

- **Example of some trends in which the Launch Site is situated in.**

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

Build a Dashboard with Plotly Dash

A Plotly Dash application for users to perform interactive visual analytics on SpaceX launch data in real-time

A interactive dashboard allow users to play around and view filtered output.

Graphs

- **Pie Chart showing the total launches by a specific site/all sites**
  - *display relative proportions of multiple classes of data.*
  - *size of the circle can be made proportional to the total quantity it represents.*

- **Scatter plot displaying the relationship PayloadMass (Kg) vs Launch Outcome for the different Booster Versions**

  It shows the correlation.

  The range of data i.e. maximum and minimum value, can be filtered

Predictive Analysis (Classification)

- **BUILDING MODEL**
- Load our dataset into NumPy and Pandas
- Transform Data
- Split the dataset into training and test data sets
- Check how many test samples we have
- Decide which type of machine learning algorithms we want to use
- Set our parameters and algorithms to GridSearchCV
- Fit our datasets into the GridSearchCV objects and train model with dataset.

- **EVALUATING MODEL**
- Check accuracy for each model
- Get tuned hyperparameters for each model
- Plot Confusion Matrix

- **IMPROVING MODEL**
- Feature Engineering
- Algorithm Tuning

FINDING THE BEST PERFORMING CLASSIFICATION MODEL

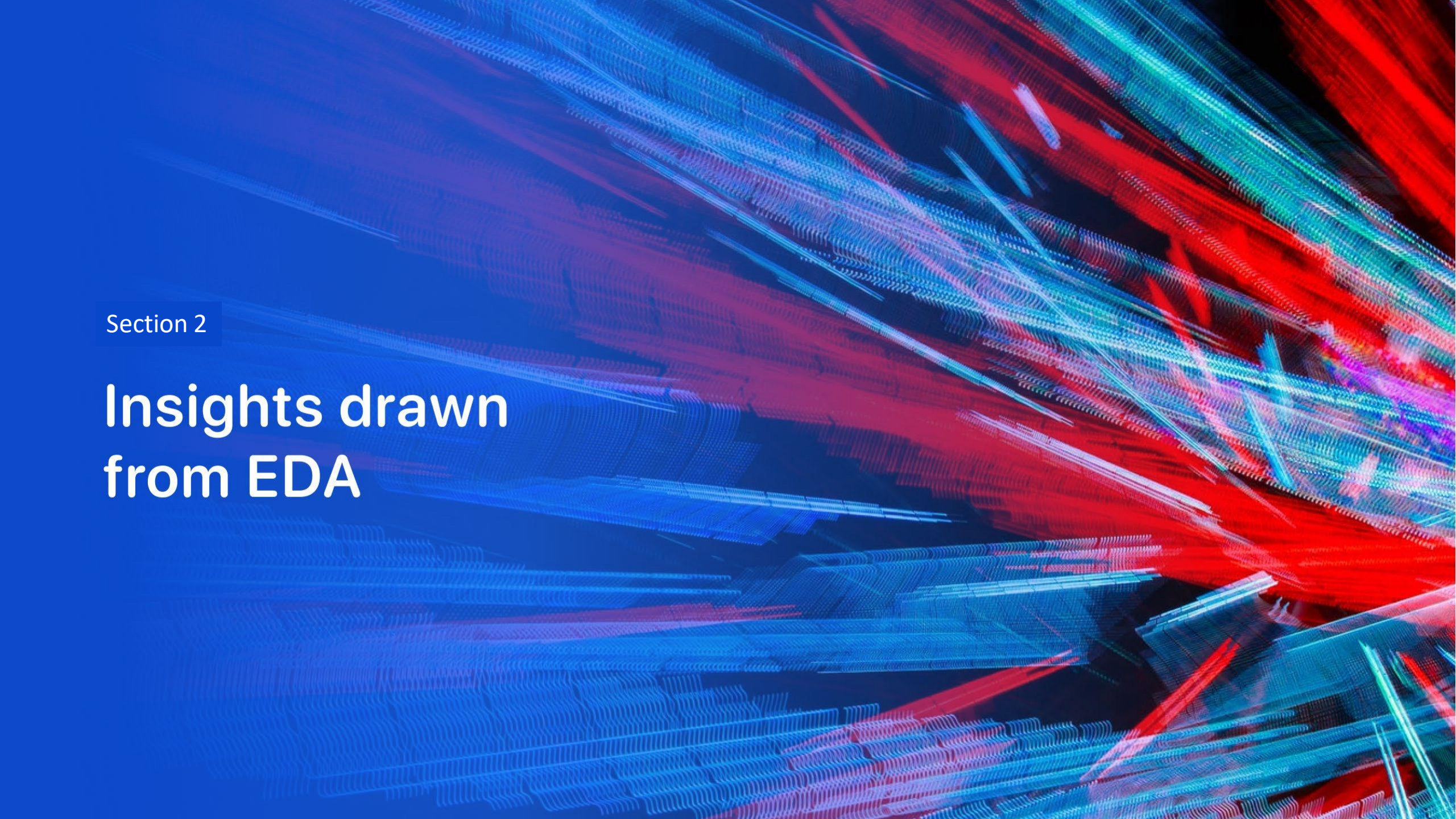
The model with the best accuracy score wins the best performing model

Results

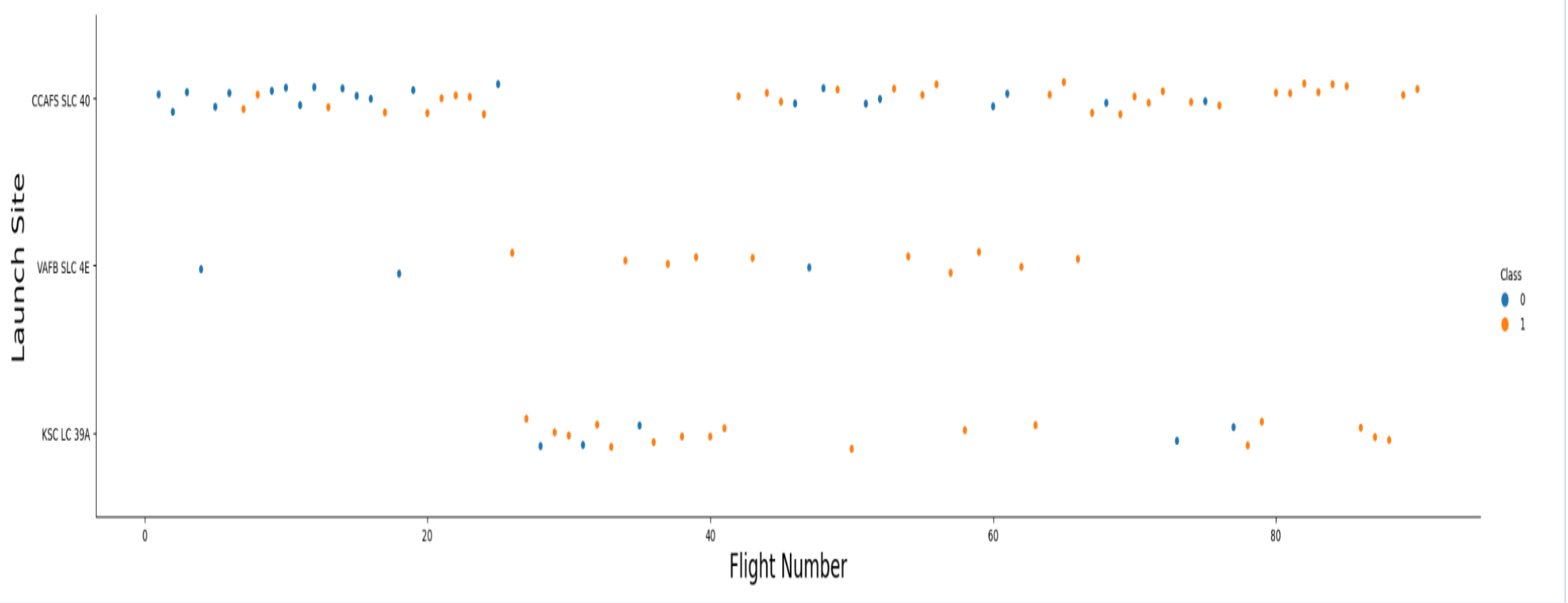
Exploratory data analysis results

Interactive analytics demo in screenshots

Predictive analysis results

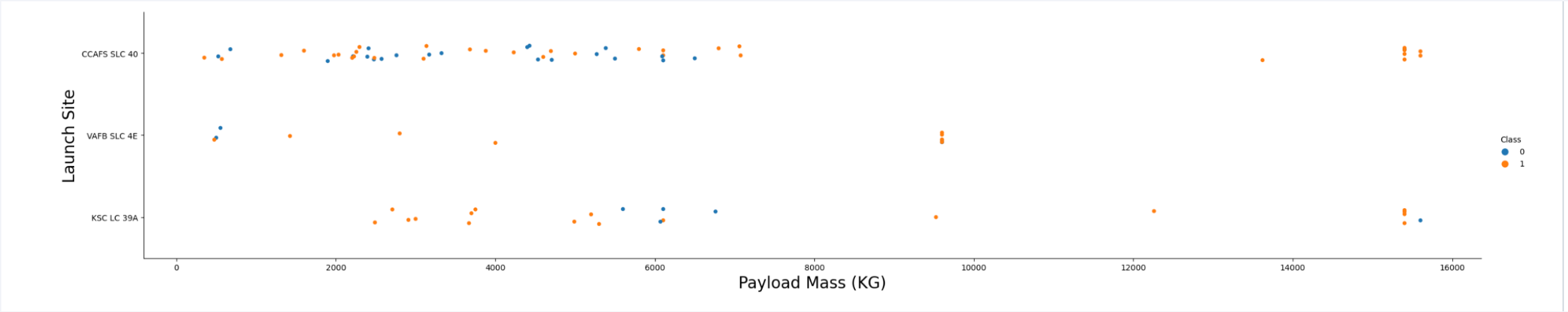

Section 2
Insights drawn
from EDA

Flight Number vs. Launch Site



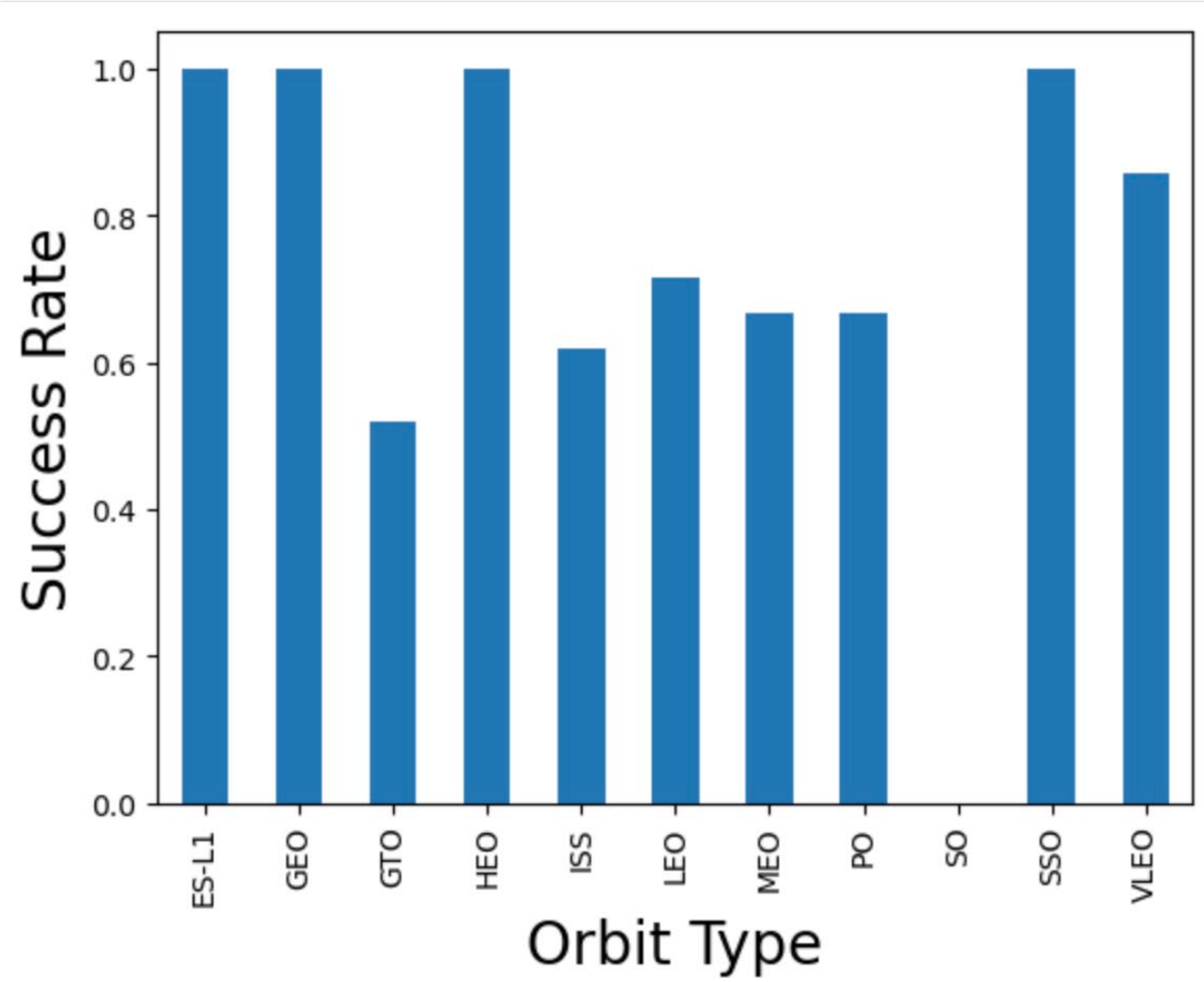
- The more flights at a launch site the greater the success rate at a launch site.

Payload vs. Launch Site



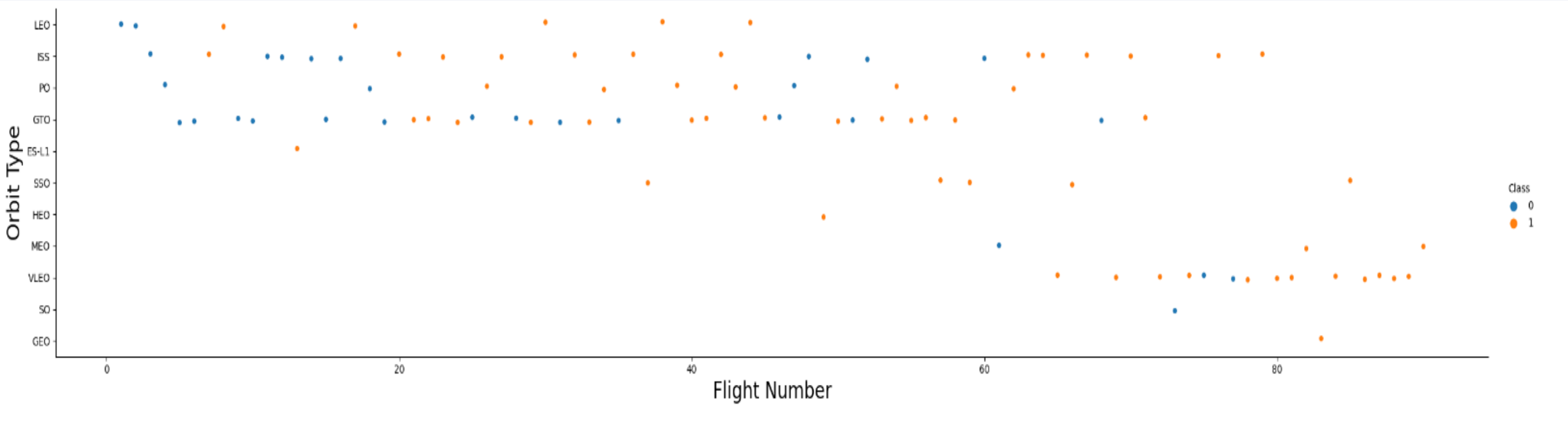
Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launch site there are no rockets launched for heavy payload mass(greater than 10000).

Success Rate vs. Orbit Type



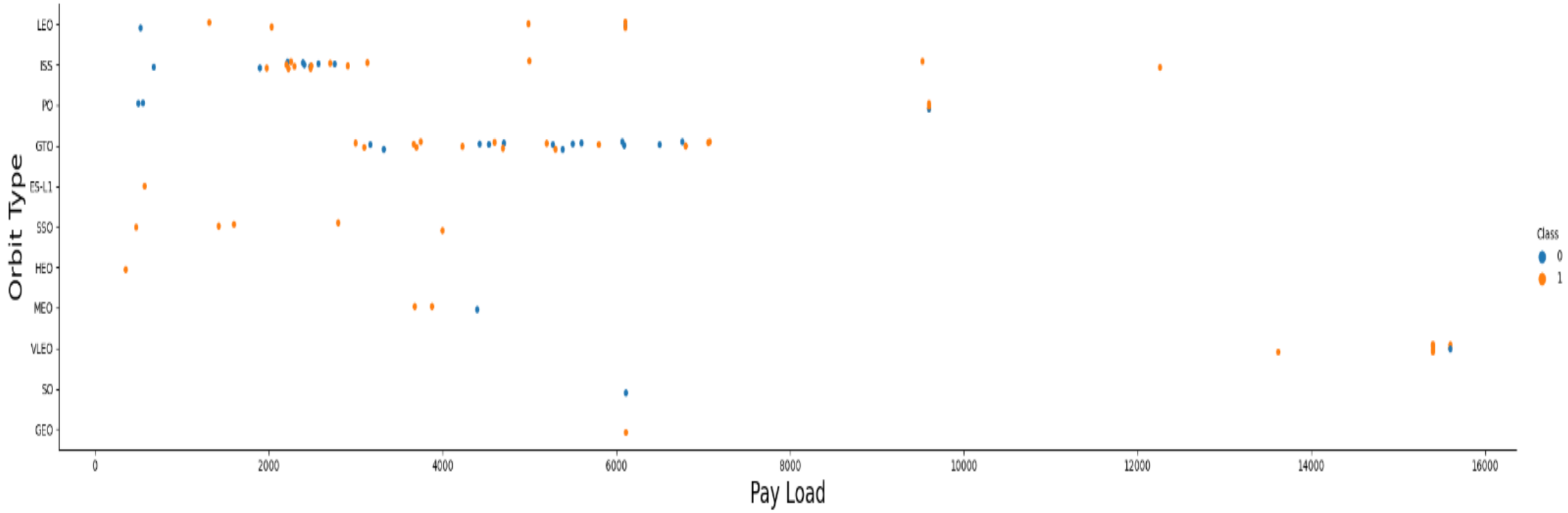
Orbit type GEO,HEO,SSO,ES-L1 have the highest Success Rate

Flight Number vs. Orbit Type



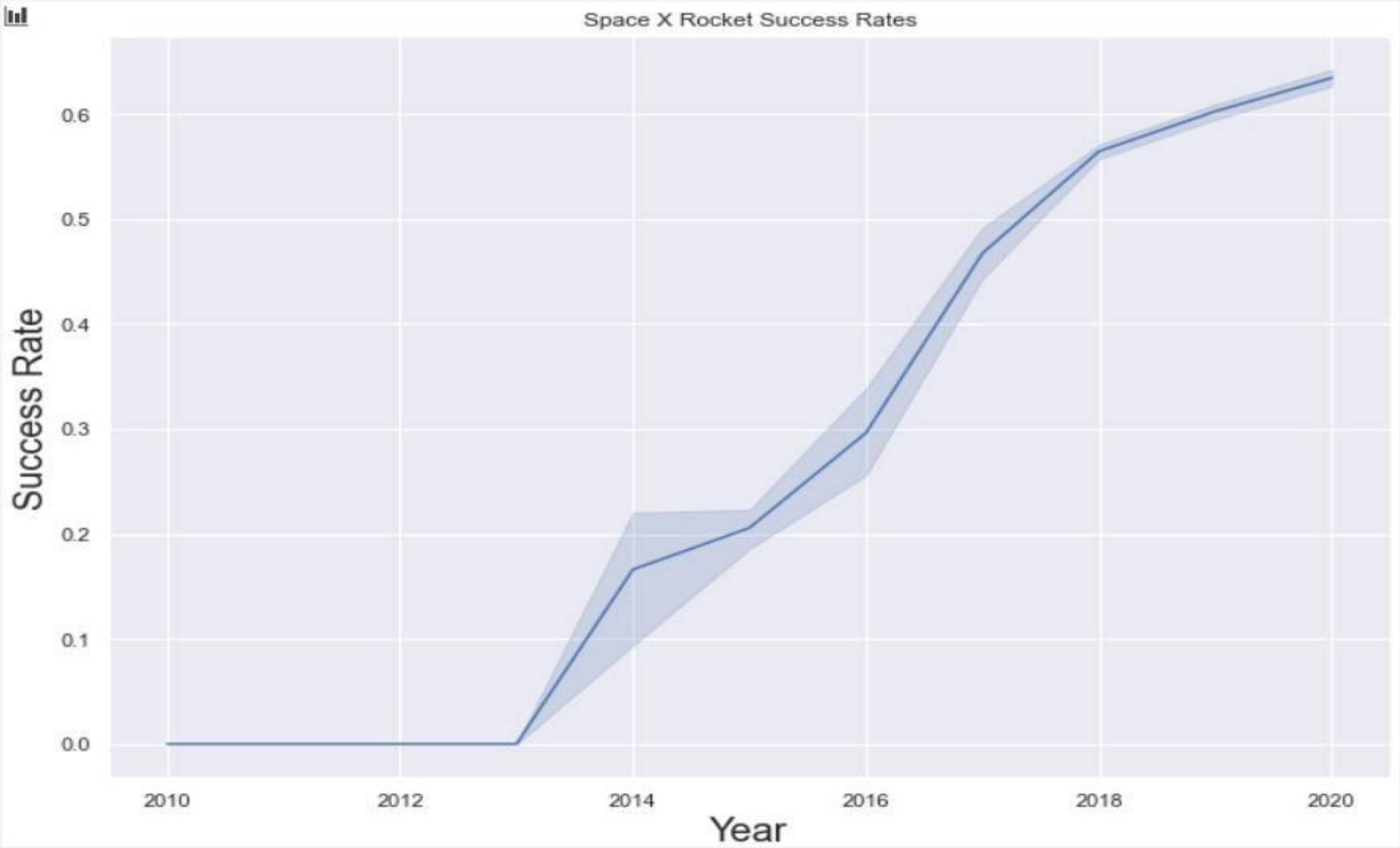
LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit

Payload vs. Orbit Type



With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS. However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here

Launch Success Yearly Trend



We can observe that the success rate since 2013 kept increasing till 2020

All Launch Site Names

- **SQL QUERY**

```
SELECT DISTINCT Launch_Site From SPACEXTBL
```

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

QUERYEXPLAINATION

Using the word ***DISTINCT*** in the query means that it will only show Unique values in the ***Launch_Site*** column from ***SpaceXTBL***

Launch Site Names Begin with 'CCA'

- **SQL QUERY**

```
SELECT * from SPACEXTBL where (LAUNCH_SITE) LIKE 'CCA%' LIMIT 5
```

]:

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

QUERYEXPLAINATION

Using the word ***limit5*** in the query means that it will only show 5 records from ***SpaceXTBL*** and ***LIKE*** keyword has a wild card to suggest that the Launch_Site name must start with CCA

Total Payload Mass

- SQL QUERY

```
SELECT SUM(PAYLOAD_MASS_KG_) FROM SPACEXTBL WHERE Customer =
'NASA (CRS)'
```

1
45596

QUERYEXPLAINATION

Calculate the total payload carried by boosters from NASA
Using the function ***SUM arrive the*** the total in the column
PAYLOAD_MASS_KG_

The ***WHERE*** clause filters the dataset to only perform calculations on ***CustomerNASA(CRS)***

Average Payload Mass by F9 v1.1

- SQL QUERY

```
SELECT AVG(PAYLOAD_MASS_KG_) FROM SPACEXTBL WHERE
Booster_Version= 'F9 v1.1'
```

1
2928

QUERYEXPLAINATION

Using the function ***AVG*** works out the average in the column ***PAYLOAD_MASS_KG_***

The ***WHERE*** clause filters the dataset to only perform calculations on ***Booster_version F9 v1.1***

First Successful Ground Landing Date

SQL QUERY

```
SELECT MIN(Date) FROM SPACEXTBL WHERE Landing_Outcome = 'Success (ground pad)'
```

1
2015-12-22

QUERYEXPLAINATION

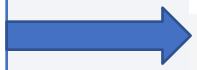
Using the function ***MIN()*** works out the minimum date in the column ***Date***

The ***WHERE*** clause filters the dataset to only perform calculations on ***Landing_Outcome Success (ground pad)***

Successful Drone Ship Landing with Payload between 4000 and 6000

SQL QUERY

```
SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE
"LANDING__OUTCOME" = 'Success (drone ship)' AND 4000 <
PAYLOAD_MASS__<6000
```



booster_version
F9 FT B1021.1
F9 FT B1023.1
F9 FT B1029.2
F9 FT B1038.1
F9 B4 B1042.1
F9 B4 B1045.1
F9 B5 B1046.1

QUERY EXPLAINATION

Selecting Booster_Version Column

The WHERE clause filters the dataset to Landing_Outcome = Success (drone ship)

The AND clause specifies additional filter conditions

Payload_MASS_KG_ > 4000 AND Payload_MASS_KG_ < 6000

Total Number of Successful and Failure Mission Outcomes

SQL QUERY

```
SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TOTAL_NUMBER  FROM
SPACEXTBL GROUP BY MISSION_OUTCOME
```

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

Query Explanation:
Select Mission_Outcome column,
Count() to get the total number of each mission outcome
Group by to summarize the results.

Boosters Carried Maximum Payload

- SQL QUERY

```
SELECT DISTINCT BOOSTER_VERSION

FROM SPACEXTBL

WHERE PAYLOAD_MASS__KG_ =

   (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL
```

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

QUERY EXPLAINATION

Using the word ***DISTINCT*** in the query means that it will only show Unique values in the ***Booster_Version*** column from ***SPACEXTBL***

SUBQUERY in th e **where** Clause filtered the result to a certain condition maximum payload.

2015 Launch Records

SQL QUERY

```
SELECT LANDING__OUTCOME, BOOSTER_VERSION, LAUNCH_SITE
FROM SPACEXTBL WHERE Landing_Outcome = 'Failure (drone ship)'
AND YEAR(DATE) = 2015
```

landing__outcome	booster_version	launch_site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

QUERY EXPLAINATION

WHERE and AND clause filtered the result to 2 conditions

“Landing_Outcome = 'Failure (drone ship)” and in 2015.

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

SQL QUERY

```
SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) AS TOTAL_NUMBER FROM SPACEXTBL WHERE DATE
BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY LANDING__OUTCOME ORDER BY TOTAL_NUMBER DESC
```

landing__outcome	total_number
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

QUERY EXPLAINATION

Count() count the total “landing_outcome” column

WHERE and AND clauses filtered the result by 2 conditions. Group by to summarize the data with landing_outcome. Order by rank the data.

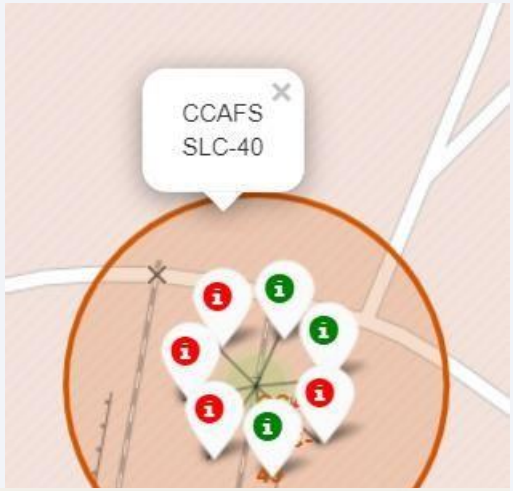
Section 3

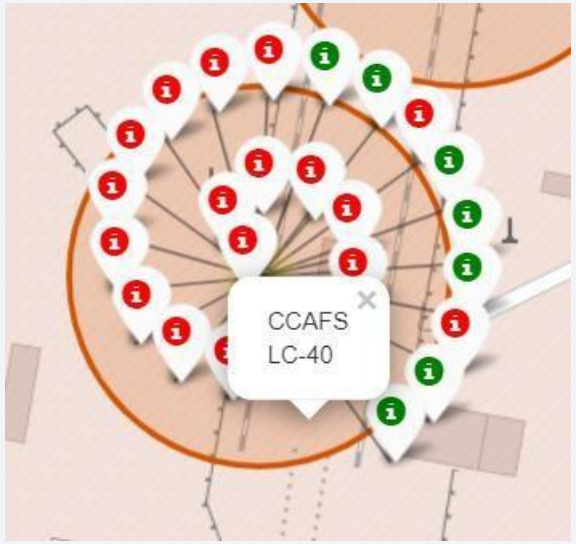
Launch Sites Proximities Analysis

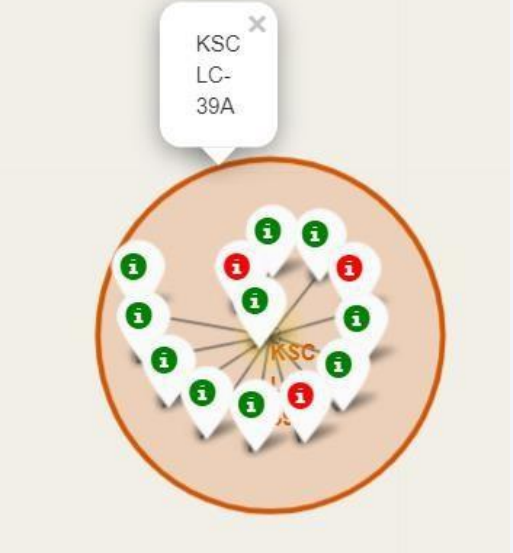
All launch sites global map markers

We can see that the SpaceX launch sites are in the United States of America coasts. Florida and California

Colour Labelled Markers





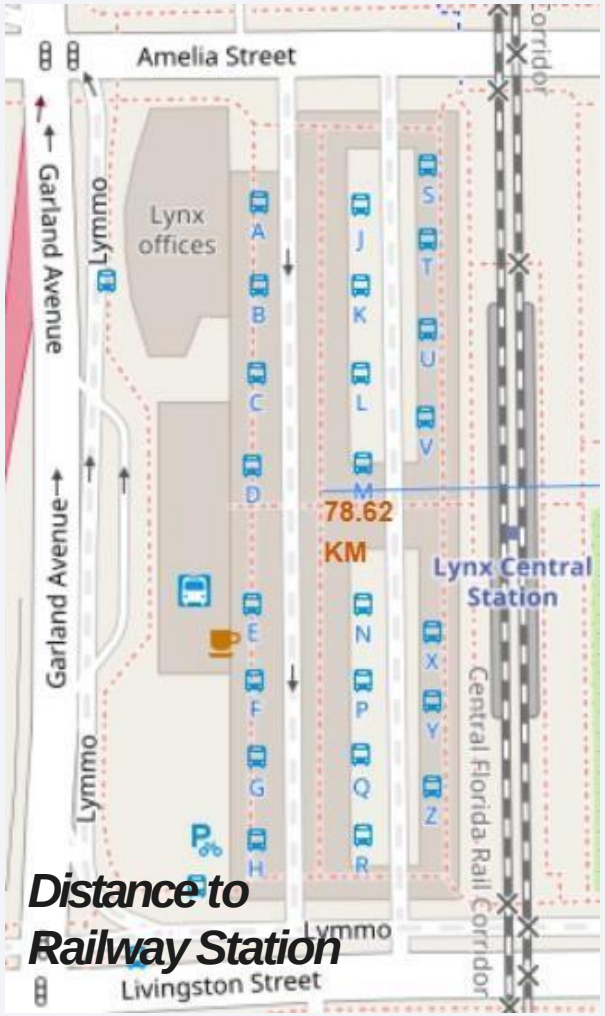


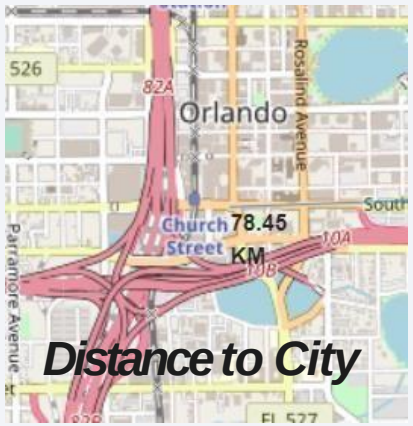
Florida Launch Sites

Green Marker shows successful Launches and Red Marker shows Failures

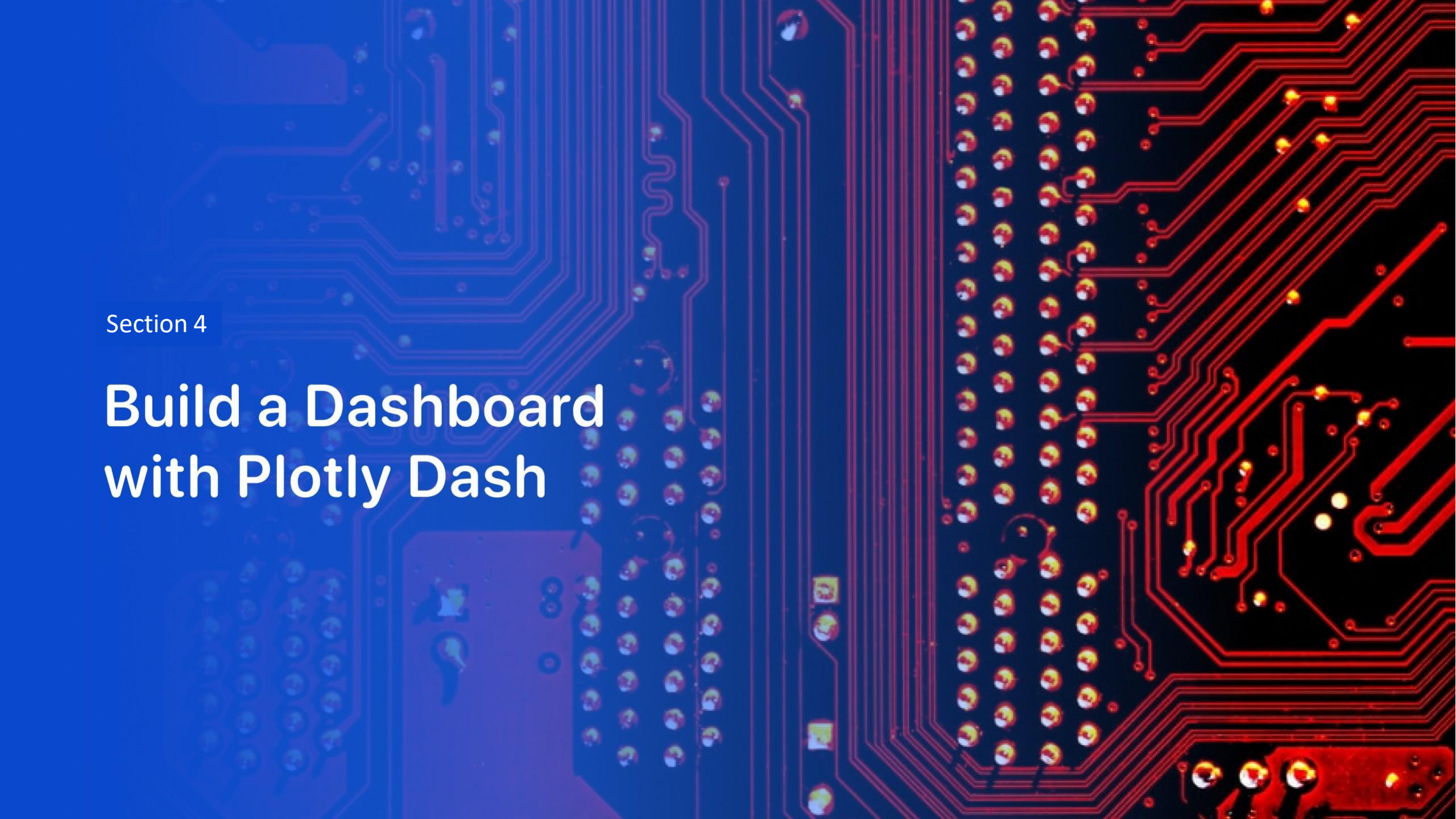
California Launch Site

Launch Sites distance to landmarks its proximities

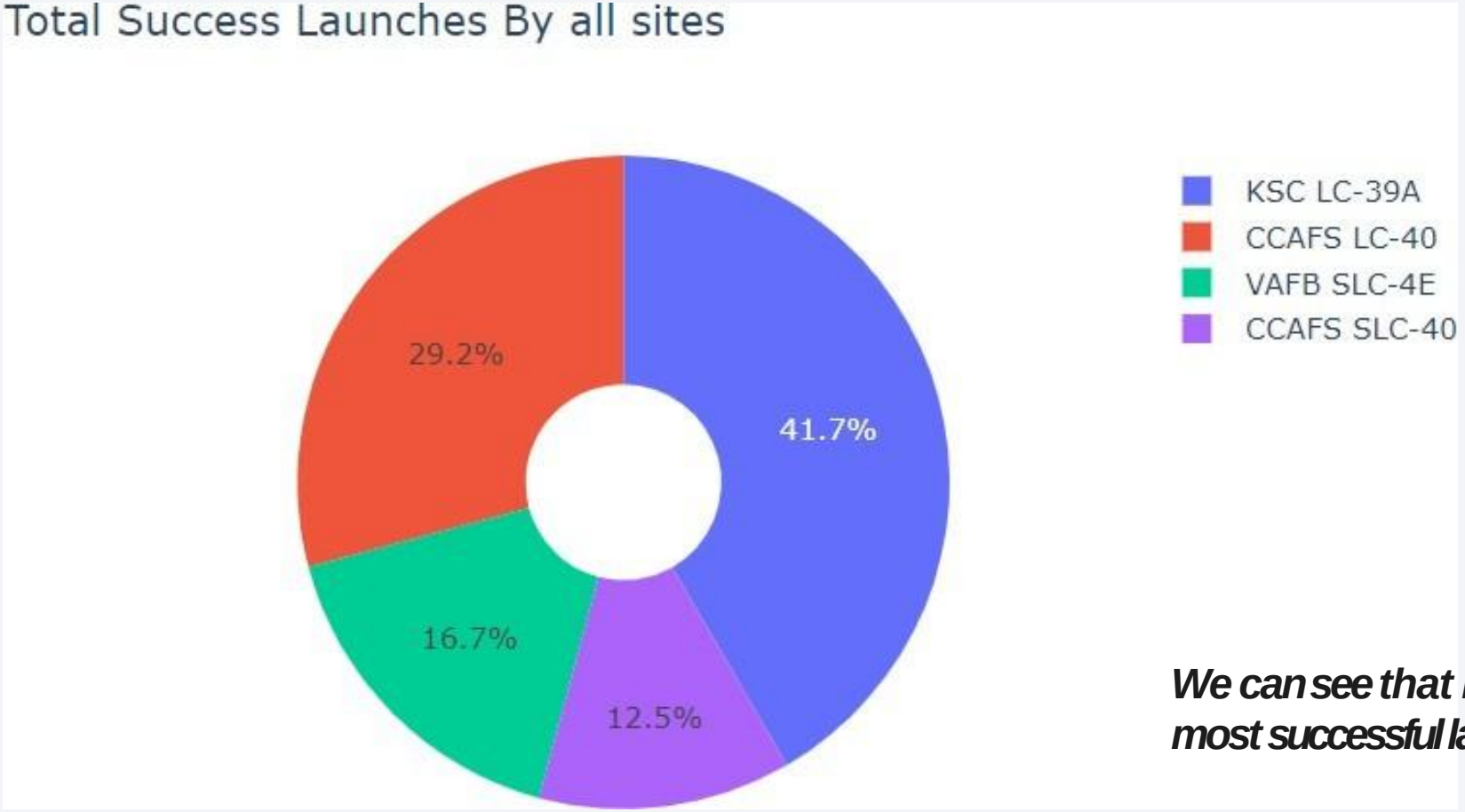




- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

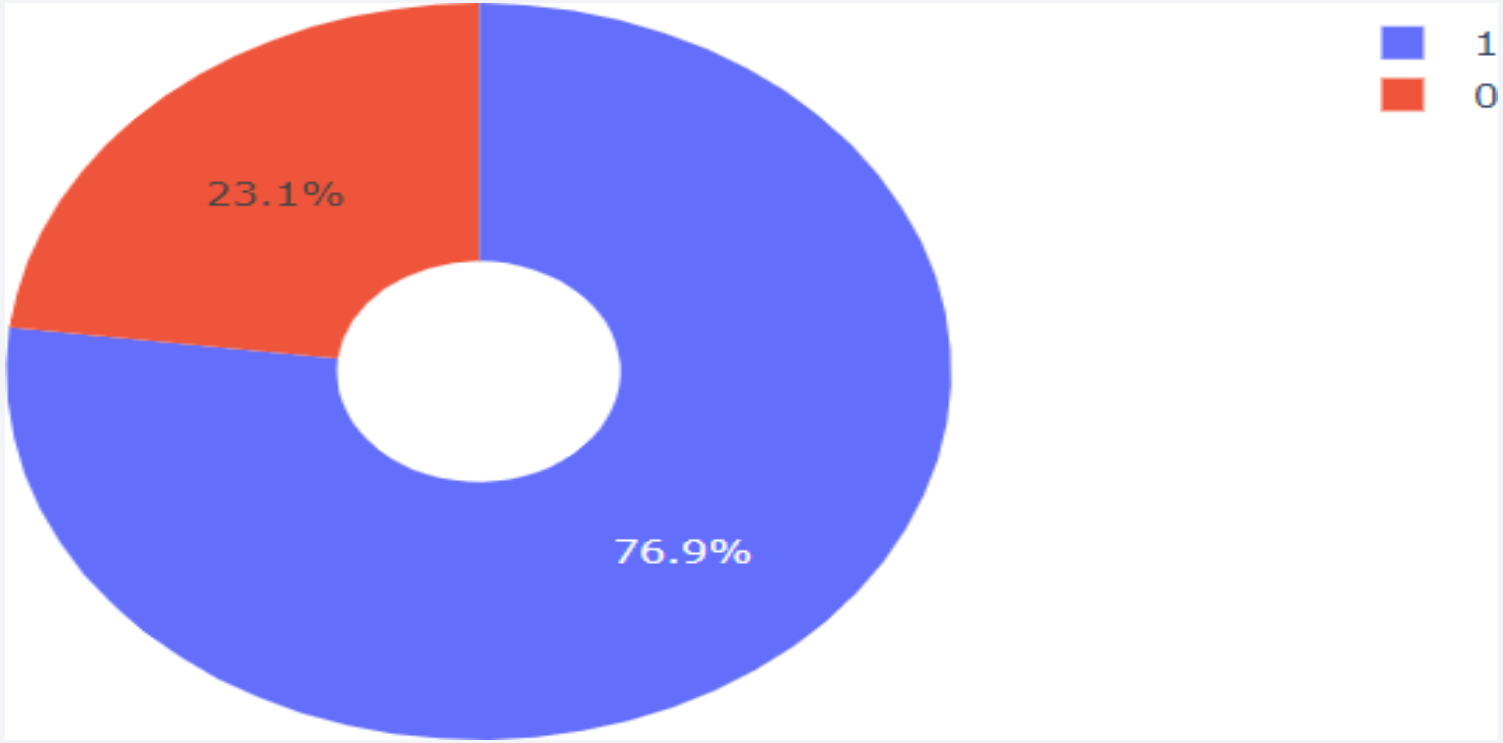

Section 4
Build a Dashboard with Plotly Dash

DASHBOARD–Piechart Total Success by each launch site



We can see that KSCLC-39A had the most successful launches from all the sites

DASHBOARD – Pie chart for launch site with the highest launch success ratio



KSCLC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

DASHBOARD – Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider

Low Weighted Payload 0kg – 5000kg

Heavy Weighted Payload 5000kg – 10000kg

We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

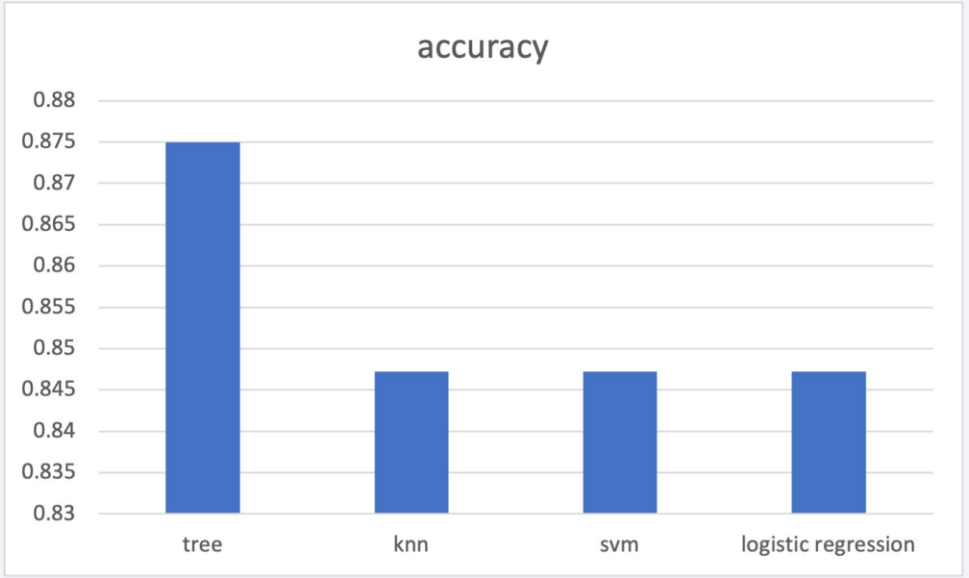
Section 5

Predictive Analysis (Classification)

Classification Accuracy

As you can see our accuracy is extremely close and Tree Algorithm slightly higher accuracy

algorithem	accuracy
tree	0.875
knn	0.847222222
svm	0.847222222
logistic regre	0.847222222



Find the method performs best:

```
]: print("Accuracy for Logistic Regression:",logreg_cv.score(X_test,Y_test))
   print( 'Accuracy for Support Vector Machine:', svm_cv.score(X_test, Y_test))
   print('Accuracy for Decision tree :', tree_cv.score(X_test, Y_test))
   print('Accuracy for K nearest neighbors :', knn_cv.score(X_test, Y_test))
```

```
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/sklearn/linear_model/base.py:283: DeprecationWarning: `np.int` is a deprecated alias for
builtin `int`. To silence this warning, use `int` by itself. Doing this will not modify any behavior and is safe. When replacing `np.int`, you may wish
se e.g. `np.int64` or `np.int32` to specify the precision. If you wish to review your current use, check the release note link for additional informati
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  indices = (scores > 0).astype(np.int)
Accuracy for Logistic Regression: 0.8333333333333334
Accuracy for Support Vector Machine: 0.8333333333333334
Accuracy for Decision tree : 0.8333333333333334
Accuracy for K nearest neighbors : 0.8333333333333334
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/sklearn/neighbors/base.py:442: DeprecationWarning: distutils Version classes are depreca
Use packaging.version instead.
  old_joblib = LooseVersion(joblib_version) < LooseVersion('0.12')
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/sklearn/neighbors/base.py:442: DeprecationWarning: distutils Version classes are depreca
Use packaging.version instead.
  old_joblib = LooseVersion(joblib_version) < LooseVersion('0.12')
```

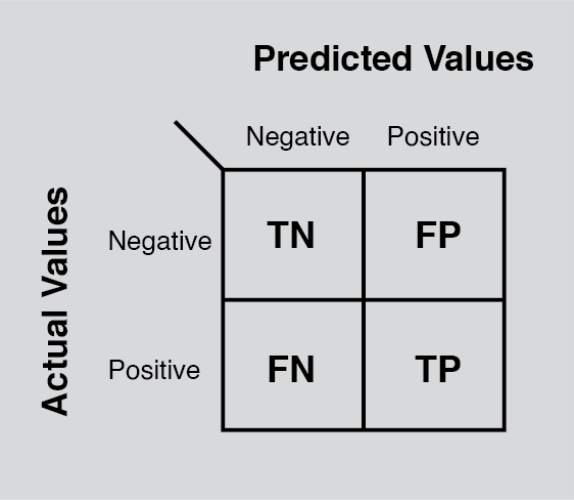
```
1]: print("Practically all these algorithms give the same result")
```

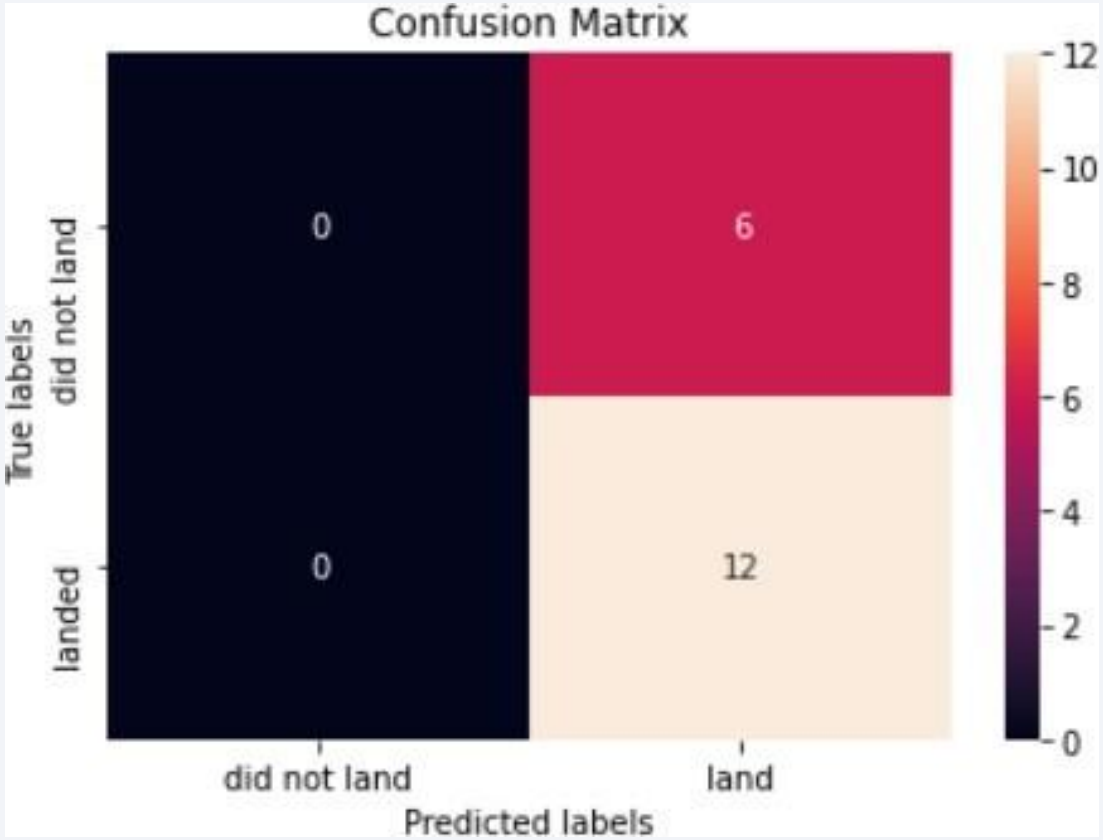
```
Practically all these algorithms give the same result
```

we achieved 83.33% accuracy on the test data.

Confusion Matrix

Examining the confusion matrix, we see that Tree can distinguish between the different classes. We see that the major problem is false positives.





Conclusions

- The Tree Classifier Algorithm is the slightly better for Machine Learning for training dataset, all model perform 83% accuracy on test data.
- Low weighted payloads perform better than the heavier payloads
- The success rates for SpaceX launches is directly proportional time in years they will eventually perfect the launches
- We can see that KSC LC-39A had the most successful launches from all the sites
- Orbit type GEO, HEO, SSO, ES-L1 has the best Success Rate

Appendix

Haversine formula is used in Folium map calculating distances

Introduction

The haversine formula determines the great-circle distance between two points on a sphere given their longitudes and latitudes. Important in navigation, it is a special case of a more general formula in spherical trigonometry, the law of haversines, that relates the sides and angles of spherical triangles.

Usage

Why did I use this formula? First of all, I believe the Earth is round/elliptical. I am not a Flat Earth Believer! Jokes aside when doing Google research for integrating my ADG Google Maps API with a Python function to calculate the distance using two distinct sets of {longitudinal, latitudinal} list sets. Haversine was the trigonometric solution to solve my requirements above.

Formula

$$a = \sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos\varphi 1 \cdot \cos\varphi 2 \cdot \sin^2\left(\frac{\Delta\lambda}{2}\right)$$

$$c = 2 \cdot \operatorname{atan2}\left(\sqrt{a}, \sqrt{(1-a)}\right)$$

$$d = R \cdot c$$

Thank you!