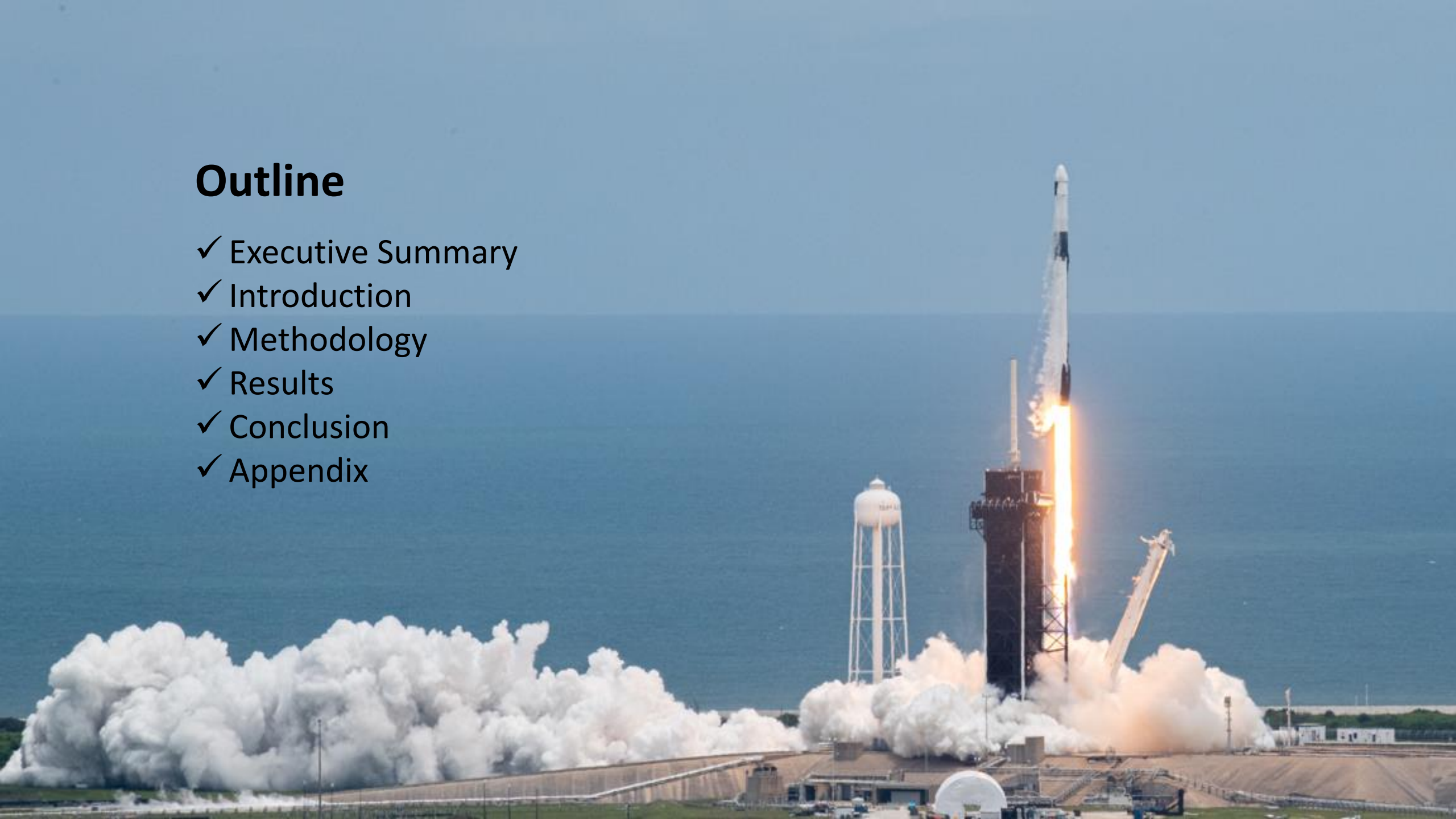
IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Veer Pal Singh
Data Scientist
28/11/2021

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 Dash
- ✓ Predictive Analysis(Classification)

✓Summary of all results

- ✓ Exploratory data analysis
- ✓ Interactive analysis demo in screenshots
- ✓ Predictive Analysis results

Introduction

Project background and context

- ✓ We predicted if the Falcon 9 first stage will land successfully
- ✓ SpaceX advertises Falcon 9 rocket launches on its website, with a cost of 65millon dollars; other providers cost upward of 165 million dollar, much of savings is because SpaceX can reuse the first stage will land,
- ✓ Therefor, 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 alternative company wants to bid against SpaceX for a rocket launch

Problems that need to be solved?

- ✓ What influences if the rocket will land successfully?
- ✓ The effect each relationship with certain rocket variables will impact in determining the success rate of a successful landing
- ✓ What conditions does SpaceX have to achieve to get the best results and ensure the best rocket success landing rate.

Section 1

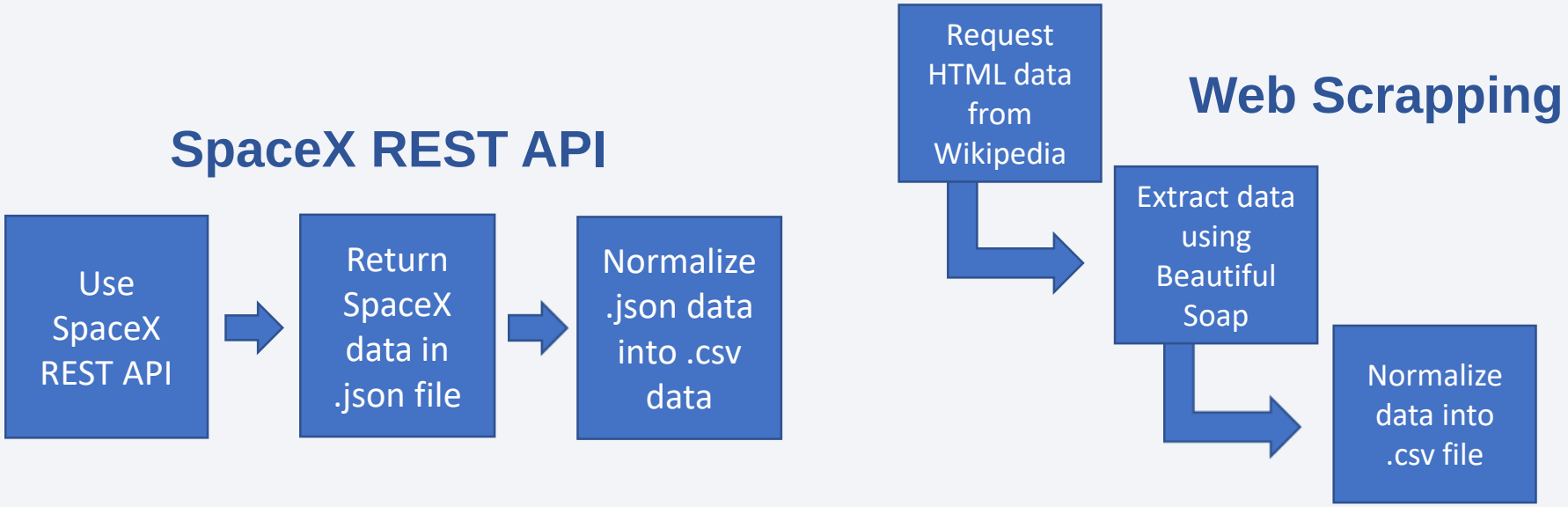
Methodology

Methodology

- ✓ Data collection methodology:
  - ✓ SpaceX REST API
  - ✓ Web Scraping from Wikipedia
- ✓ Perform data wrangling
  - ✓ One Hot Encoding data fields for Machine Learning and dropping out irrelevant columns
- ✓ Perform exploratory data analysis (EDA) using visualization and SQL
  - ✓ Plotting: scattered Graphs and Bar Graphs to show relationship between variables
- ✓ Perform interactive visual analytics using Folium and Plotly Dash
- ✓ Perform predictive analysis using classification models
  - ✓ How to build, how tuned, how evaluated the classification models

Data Collection

- ✓ We worked with SpaceX launch data that is gathered from the SpaceX REST API.
- ✓ This API will give us data about launches, including information about the rocket used, payload delivered, launch specifications, landing specifications, and landing outcome.
- ✓ Our goal is to use this data to predict whether SpaceX will attempt to land a rocket or not.
- ✓ The SpaceX REST API endpoints, or URL, starts with api.spacexdata.com/v4/.
- ✓ Another popular data source for obtaining Falcon 9 Launch data is web scraping Wikipedia using BeautifulSoup.



Data Collection – SpaceX API

- ✓ **Data collection with SpaceX REST API**
  - ✓ Use SpaceX REST API
  - ✓ API return SpaceX data in .json
  - ✓ Filter .json data into dataframe
  - ✓ Clean the data
  - ✓ Export data to flat .csv file

- ✓ Add the GitHub URL: https://github.com/veer2701/github_projects/blob/main/IBM-Data-Science---SpaceX-Capstone-Project/1.1%20%20jupyter-labs-spacex-data-collection-api-OK-Final.ipynb

1. Getting response from API

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

2. Converting Response to a .json file

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

3. Apply custom functions to clean data

```
getBoosterVersion(data)
getLaunchSite(data)
getPayloadData(data)
getCoreData(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}
data2 = pd.DataFrame(launch_dict)
```

5. Filter dataframe and export to flat file in .csv

```
data_falcon9 = data2[data2['BoosterVersion']=='Falcon 9']
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data Collection – Web Scraping

Add the GitHub URL:
https://github.com/veer2701/github_projects/blob/main/IBM-Data-Science---SpaceX-Capstone-Project/1.2%20%20jupyter-labs-webscraping-OK.ipynb

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

```
for i in first_launch_table.find_all('th'):
    if extract_column_from_header(i)!=None:
        if len(extract_column_from_header(i))>0:
            column_names.append(extract_column_from_header(i)
print(column_names)
```

5. Creation of dictionary

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each
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. Appending data to keys

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"w
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number cc
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
```

7. Converting dictionary to dataframe

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

8. Dataframe to .csv file

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

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.

Perform Exploratory Data Analysis EDA on dataset

Calculate the number of launches at each site

Calculate the number of occurrence of each orbit

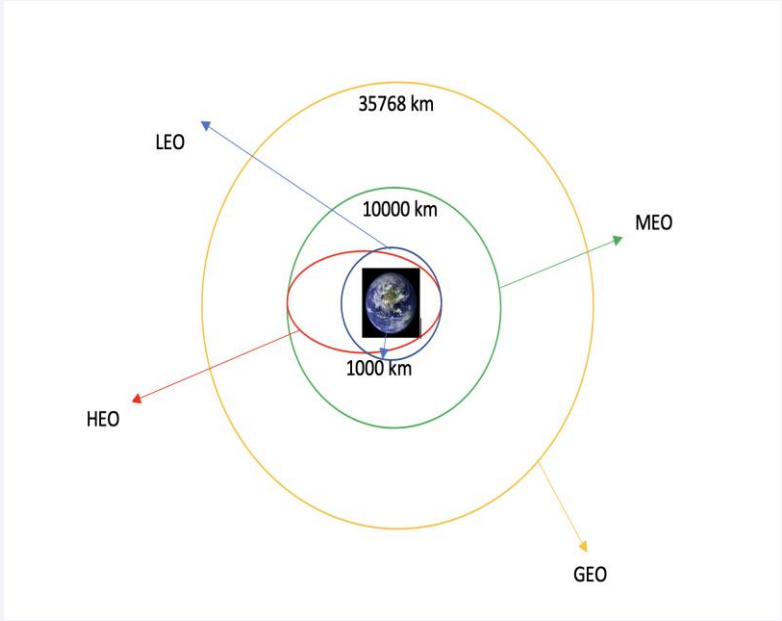
Calculate the number and occurrence of mission outcome per orbit type

Export dataset as a .csv file

Create a landing outcome label from Outcome column

Work out success rate for every landing in dataset

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

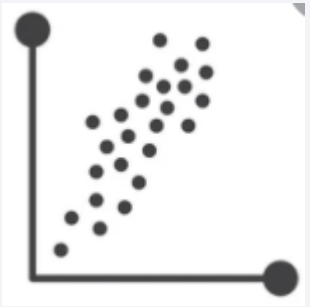


GitHub URL:
https://github.com/veer2701/github_projects/blob/main/IBM-Data-Science---SpaceX-Capstone-Project/1.3%20%20labs-jupyter-spacex-Data%20wrangling-OK.ipynb

EDA with Data Visualization

Scatter Graphs being drawn:

- ✓ Flight Number vs Payload Mass
- ✓ Flight Number vs Launch Site
- ✓ Payload Mass vs Launch Site
- ✓ Orbit vs. Flight Number
- ✓ Payload vs. Orbit Type
- ✓ Orbit vs. Payload Mass



Scatter plots show how much one variable is affected by another. The relationship between two variables is called their correlation . Scatter plots usually consist of a large body of data.

Add the GitHub URL:
https://github.com/veer2701/github_projects/blob/main/IBM-Data-Science---SpaceX-Capstone-Project/2.2%20%20jupyter-labs-eda-dataviz-OK.ipynb

Bar Graph being drawn:

Mean vs. Orbit

A bar diagram makes it easy to compare sets of data between different groups at a glance. The graph represents categories on one axis and a discrete value in the other. The goal is to show the relationship between the two axes. Bar charts can also show big changes in data over time.

Line Graph being drawn:

Success rate vs. Year

Line graphs are useful in that they show data variables and trends very clearly and can help to make predictions about the results of data not yet recorded

EDA with SQL

Performed SQL queries to gather information about the dataset

We needed the information about some questions that we were asked about data. For that we used SQL queries to get the answers in the dataset:

- ✓ Displaying the names of the unique launch sites in the space mission
- ✓ Displaying 5 records where launch sites begin with the string 'KSC’
- ✓ Displaying the total payload mass carried by boosters launched by NASA (CRS)
- ✓ Displaying average payload mass carried by booster version F9 v1.1
- ✓ Listing the date where the successful landing outcome in drone ship was achieved.
- ✓ Listing the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000
- ✓ Listing the total number of successful and failure mission outcomes
- ✓ Listing the names of the booster versions which have carried the maximum payload mass.
- ✓ Listing the records which will display the month names, successful landing outcomes in ground pad ,booster versions, launch site for the months in year 2017
- ✓ Ranking the count of successful landing outcomes between the date 2010-06-04 and 2017-03-20 in descending order.

GitHub url: https://github.com/veer2701/github_projects/blob/main/IBM-Data-Science---SpaceX-Capstone-Project/2.1%20%20jupyter-labs-eda-sql-coursera_OK.ipynb

Build an Interactive Map with Folium

To visualize the Launch Data into an interactive map. We took the Latitude and Longitude Coordinates at each launch site and added a Circle Marker around each launch site with a label of the name of the launch site.

We assigned the dataframe **launch_outcomes(failures, successes) to classes 0 and 1** with **Green** and **Red** markers on the map in a MarkerCluster()

Using Haversine's formula we calculated the distance from the Launch Site to various landmarks to find various trends about what is around the Launch Site to measure patterns. **Lines** are drawn on the map to measure distance to landmarks

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

Add the GitHub URL https://github.com/veer2701/github_projects/blob/main/IBM-Data-Science---SpaceX-Capstone-Project/3.1%20%20%20lab_jupyter_launch_site_location-OK.ipynb

Build a Dashboard with Plotly Dash

Used Python to host the website so your can play around with the data and view the data

✓ **The dashboard is built with Plotly and Dash web framework.**

Graphs

- Pie Chart showing the total launches by a certain 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 Graph showing the relationship with Outcome and Payload Mass (Kg) for the different Booster Versions**

- ✓ It shows the relationship between two variables.
- ✓ It is the best method to show you a non-linear pattern.
- ✓ The range of data flow, i.e. maximum and minimum value, can be determined.
- ✓ Observation and reading are straightforward.

Add the GitHub URL:

https://github.com/veer2701/github_projects/blob/main/IBM-Data-Science---SpaceX-Capstone-Project/3.2%20%20Data%20Visualization%20with%20Python.py

Predictive Analysis (Classification)

✓Building Model

- ✓ Load our dataset into NumPy and Pandas
- ✓ Transform Data
- ✓ Split our data 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 our dataset.

✓Evaluating Model

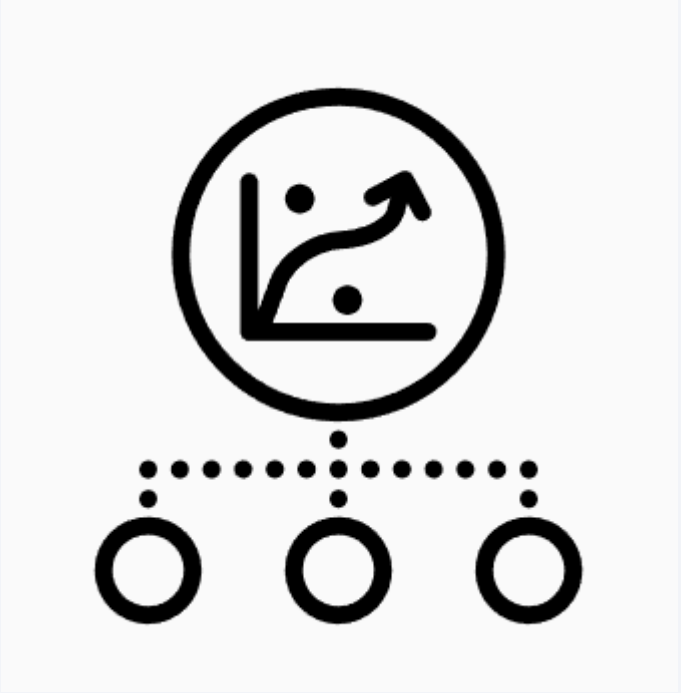
- ✓ Check accuracy for each model
- ✓ Get tuned hyperparameters for each type of algorithms
- ✓ Plot Confusion Matrix

✓Improving Model

- ✓ Check accuracy for each model
- ✓ Get tuned hyperparameters for each type of algorithms
- ✓ Plot Confusion Matrix

✓Finding the best performing Classification model

- ✓ Check accuracy for each model
- ✓ Get tuned hyperparameters for each type of algorithms
- ✓ Plot Confusion Matrix



Add the GitHub URL

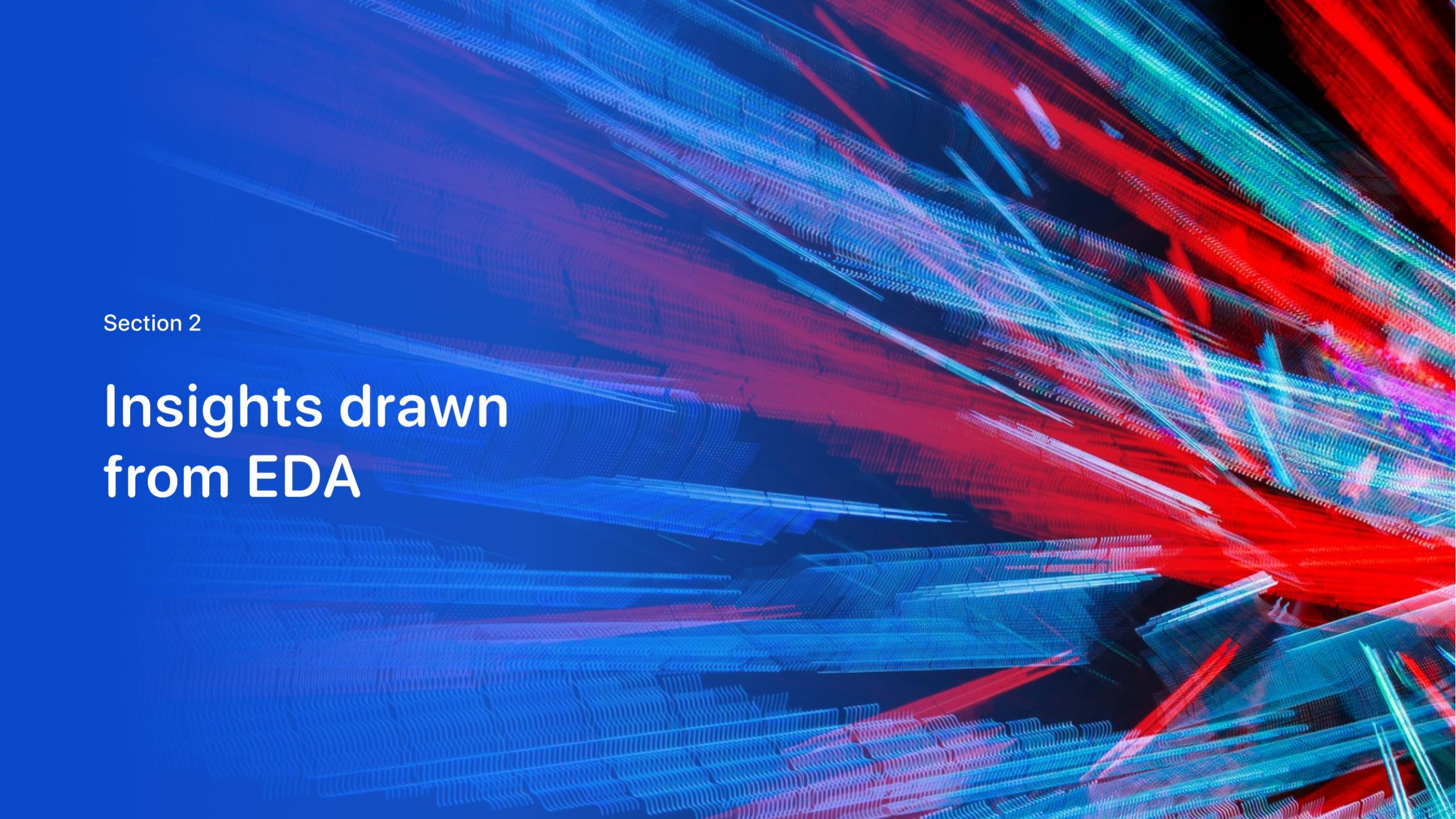
https://github.com/veer2701/github_projects/blob/main/IBM-Data-Science---SpaceX-Capstone-Project/4%20%20SpaceX_Machine%20Learning%20Prediction_Part_5-OK-Final.ipynb

Results

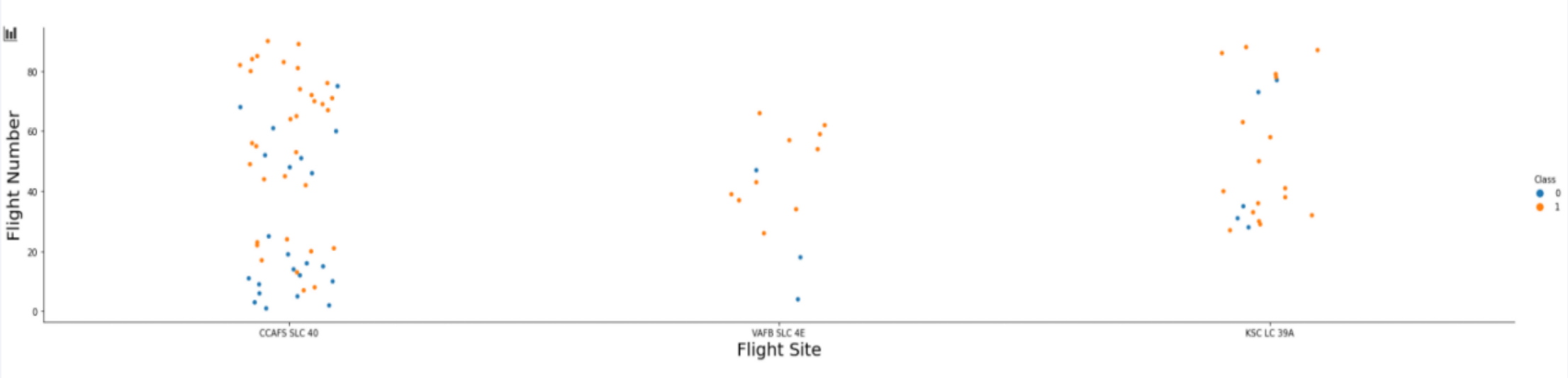
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

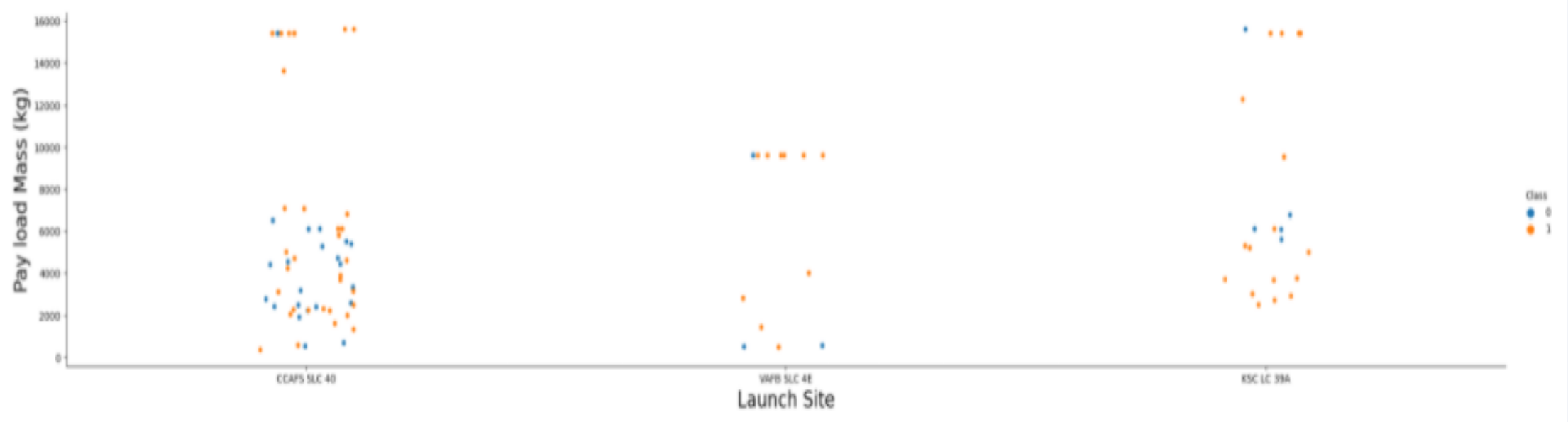


Flight Number vs. Launch Site



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

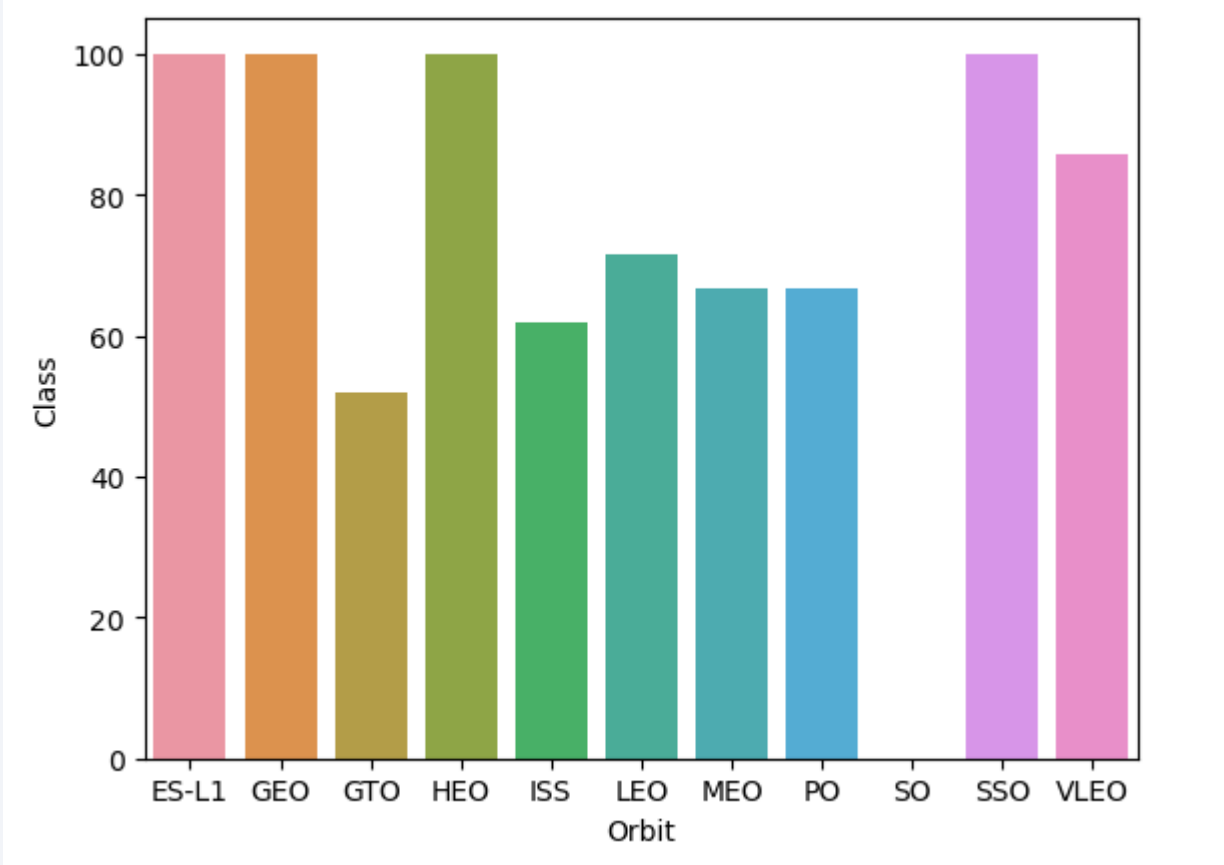
Payload vs. Launch Site



The greater the payload mass for Launch Site CCAFS SLC 40 the higher the success rate for the Rocket. There is not quite a clear pattern to be found using this visualization to make a decision if the Launch Site is dependent on Pay Load Mass for a success launch

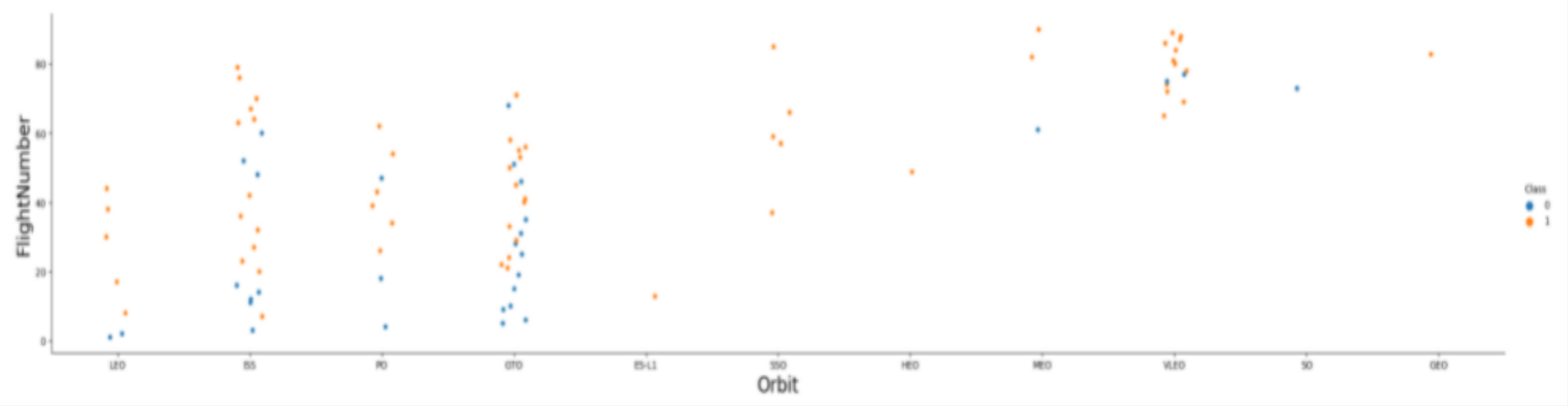
Success Rate vs. Orbit Type

Bar chart for the success rate of each orbit type



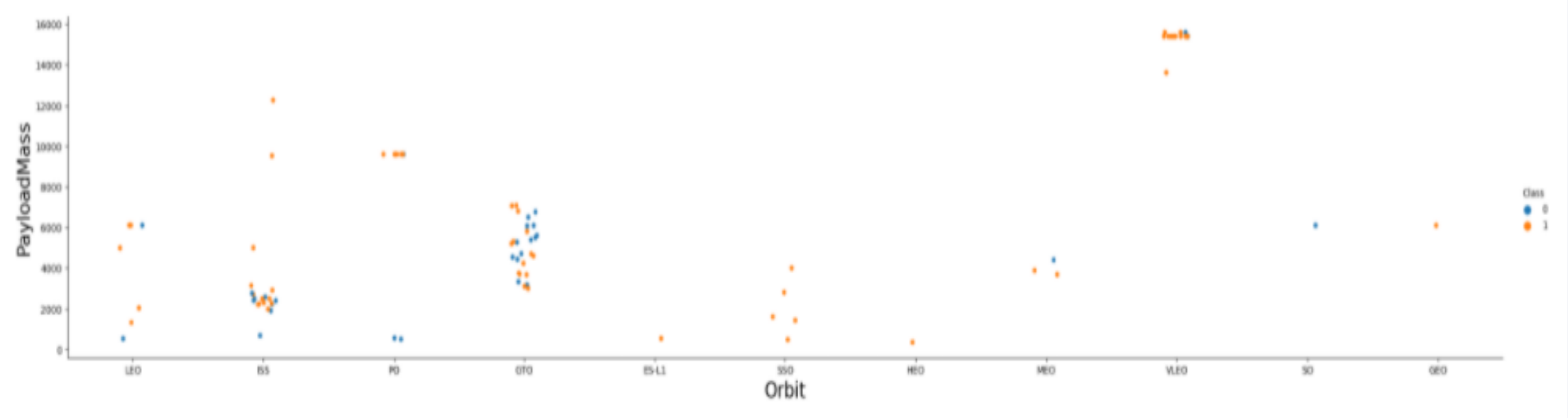
Orbit GEO, HEO, SSO, ES-L1 has the best Success Rate

Flight Number vs. Orbit Type



You should see that in the 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

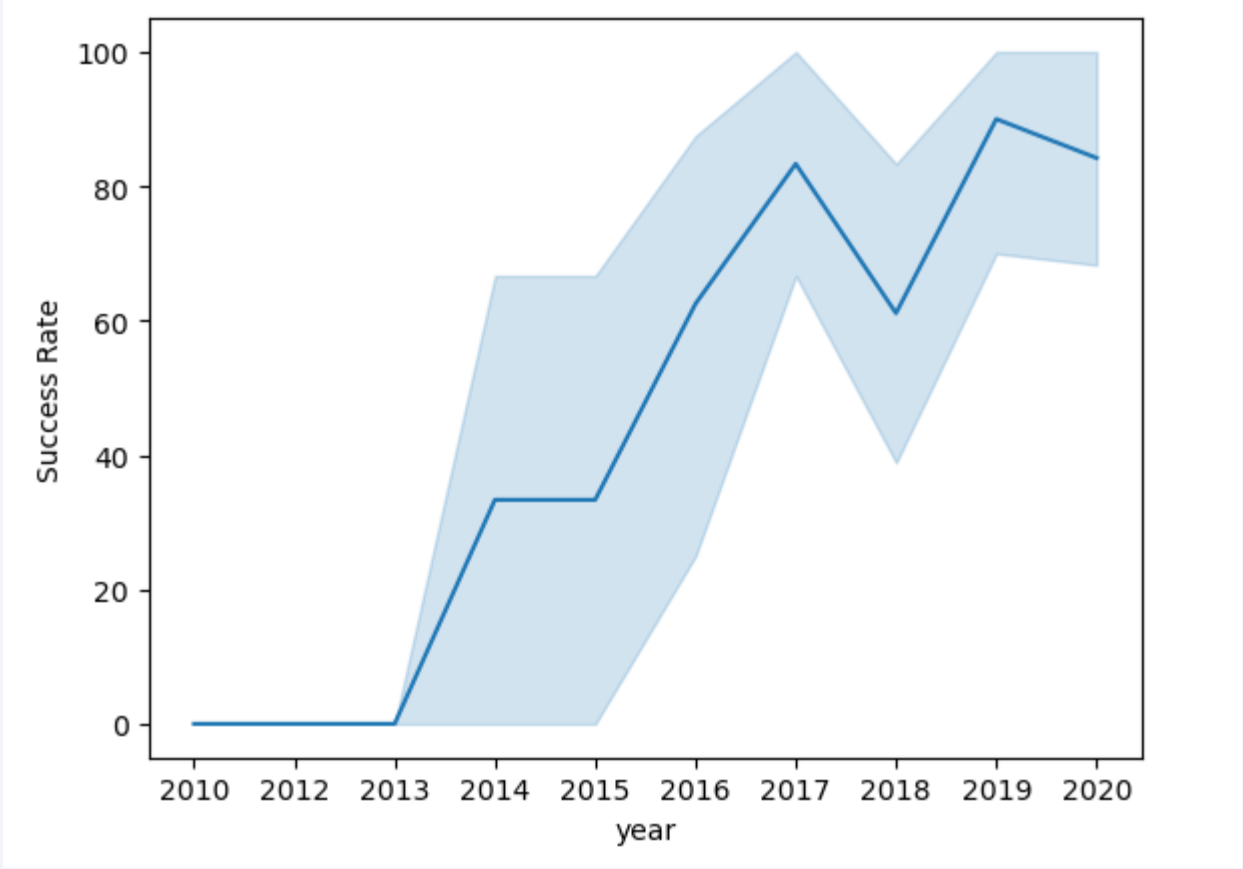


You can observe that Heavy payloads have a negative influence on GTO orbits and positive on GTO and Polar LEO (ISS) orbits.

Launch Success Yearly Trend

Line chart of yearly average success rate

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



All Launch Site Names

SQL Query:

%sql select DISTINCT LAUNCH_SITE from SPACEX

```
1 %sql select DISTINCT LAUNCH_SITE from SPACEX
 * ibm_db_sa://yjp99633:***@55fbc997-9266-4331-afd3-8
Done.
```

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

Query Explanation:

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

Launch Site Names Begin with 'CCA'

SQL Query:

%sql select * from SPACEX where launch_site like 'CCA%' limit 5

Query Explanation:

First five entries in database with Launch Site name beginning with CCA.

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 Brouère cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2+	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	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 Query:

%sql select sum(payload_mass__kg_) as sum from SPACEX where customer like 'NASA (CRS)'

```
1 %sql select sum(payload_mass__kg_) as sum
 * ibm_db_sa://yjp99633:***@55fbc997-9266-4331-
Done.
 SUM
45596
```

Query Explanation:

This query sums the total payload mass in kg where NASA was the customer.

CRS stands for Commercial Resupply Services which indicates that these payloads were sent to the International Space Station (ISS).

Average Payload Mass by F9 v1.1

SQL Query:

%sql select avg(payload_mass__kg_) as Average from SPACEX
where booster_version like 'F9 v1.1%'

```
1 %sql select avg(payload_mass__kg_) as Av
 * ibm_db_sa://yjp99633:***@55fbc997-9266-43
Done.
average
2534
```

Query Explanation:

This query calculates the average payload mass or launches which used booster version F9 v1.1

Average payload mass of F9 1.1 is on the low end of our payload mass range

First Successful Ground Landing Date

SQL Query:

%sql select min(date) as Date from SPACEX where mission_outcome like 'Success'

```
1 %sql select min(date) as Date from SPACEX where landing_outcome like 'Success
 * ibm_db_sa://yjp99633:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1od8l
Done.
```

DATE
2016-05-06

Query Explanation:

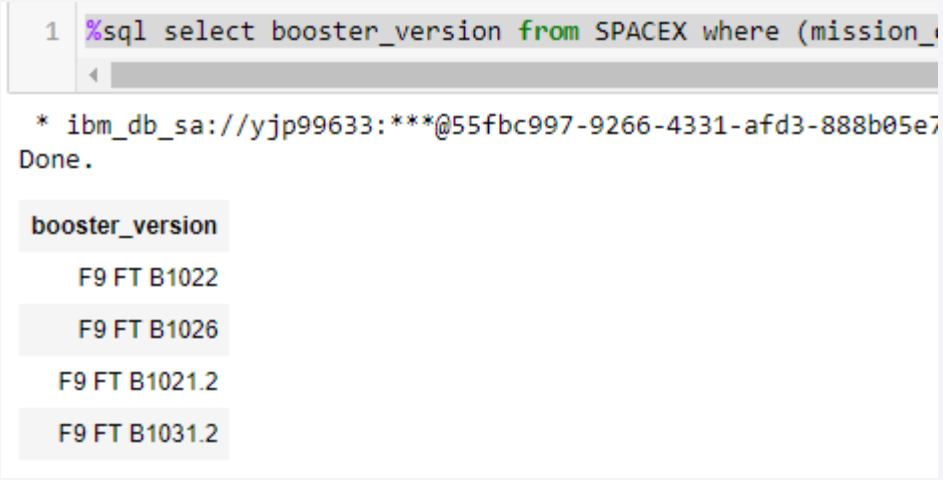
Using the function MIN works out the minimum date in the column Date

The WHERE clause filters the dataset to only perform calculations on Landing_Outcome Success (drone ship)

Successful Drone Ship Landing with Payload between 4000 and 6000

SQL Query:

%sql select booster_version from SPACEX where (mission_outcome like 'Success') AND (payload_mass__kg_ BETWEEN 4000 AND 6000) AND (landing_outcome like 'Success (drone ship)')



Query Explanation:

Selecting only Booster_Version

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:

%sql SELECT mission_outcome, count(*) as Count FROM SPACEX GROUP by mission_outcome ORDER BY mission_outcome

```
1 %sql SELECT mission_outcome, count(*) as Count FROM SPACEX GROUP by
 * ibm_db_sa://yjp99633:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io9
Done.
```

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

Query Explanation:

This query returns a count of each mission outcome. SpaceX appears to achieve its mission outcome nearly 99% of the time.

This means that most of the landing failures are intended.

Interestingly, one launch has an unclear payload status and unfortunately one failed in flight.

Boosters Carried Maximum Payload

SQL Query:

%sql SELECT DISTINCT booster_version, MAX(payload_mass__kg_) as MaximumPayloadMass FROM SPACEX GROUP BY booster_version

Query Explanation:

Using the word DISTINCT in the query means that it will only show Unique values in the Booster_Version column from SpaceX
GROUP BY puts the list in order set to a certain condition.
DESC means its arranging the dataset into descending order

```
1 %%sql
2 SELECT booster_version, payload_mass__kg_
3 FROM SPACEX
4 where payload_mass__kg_ = (select MAX(payload_mass__kg_)
```

```
 * ibm_db_sa://yjp99633:***@55fbc997-9266-4331-afd3-888b05e7
Done.
```

booster_version	payload_mass__kg_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

2016 Launch Records

SQL Query:

%sql SELECT MONTH(DATE), DATE, LANDING_OUTCOME, BOOSTER_VERSION, LAUNCH_SITE FROM SPACEX where LANDING_OUTCOME like 'Failure (drone ship)';

```
1 %sql SELECT MONTH(DATE), DATE, LANDING_OUTCOME, BOOSTER_VERSION, payload_mass__kg_
```

```
 * ibm_db_sa://yjp99633:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1od8lcg.
Done.
```

1	DATE	landing_outcome	booster_version	payload_mass__kg_	launch_site
1	2016-01-17	Failure (drone ship)	F9 v1.1 B1017	553	VAFB SLC-4E
3	2016-03-04	Failure (drone ship)	F9 FT B1020	5271	CCAFS LC-40

Query Explanation:

This query returns the Month, Landing Outcome, Booster Version, Payload Mass (kg), and Launch site of 2016 launches where stage 1 failed to land on a drone ship.

There were two such occurrences.

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

SQL Query:

%%sql select landing_outcome, COUNT(*) AS no_outcome
from SPACEX where landing_outcome like 'Success%' AND DATE BETWEEN '2010-06-04' AND '2017-03-20'
GROUP by landing_outcome
ORDER BY no_outcome Desc;

Query Explanation:

This query returns a list of successful landings and between 2010-06-04 and 2017-03-20 inclusively.

There are two types of successful landing outcomes: drone ship and ground pad landings.

There were 8 successful landings in total during this time

```
1 %%sql select landing_outcome, COUNT(*) AS no_outcome
2 from SPACEX where landing_outcome like 'Success%' AND DATE BETWEEN '2010-06-04' AND '2017-03-20'
3 GROUP by landing_outcome
4 ORDER BY no_outcome Desc;
```

```
 * ibm_db_sa://yjp99633:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1od8lcg.databases.appdo
Done.
```

landing_outcome	no_outcome
Success (drone ship)	3
Success (drone ship)	2
Success (ground pad)	2
Success (ground pad)	1

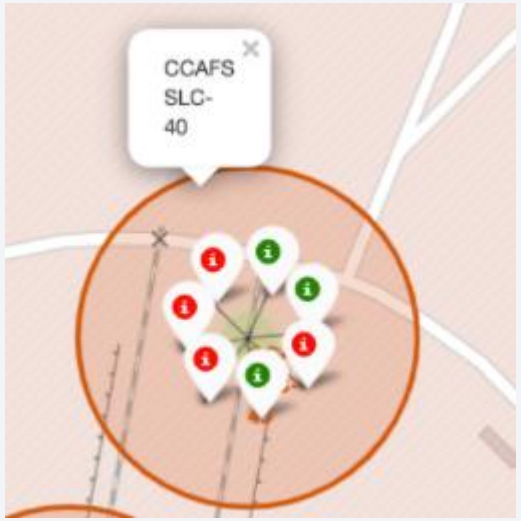
Section 4

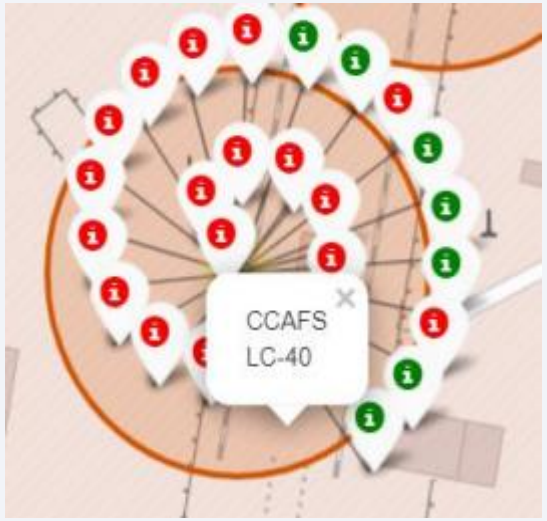
Launch Sites Proximities Analysis

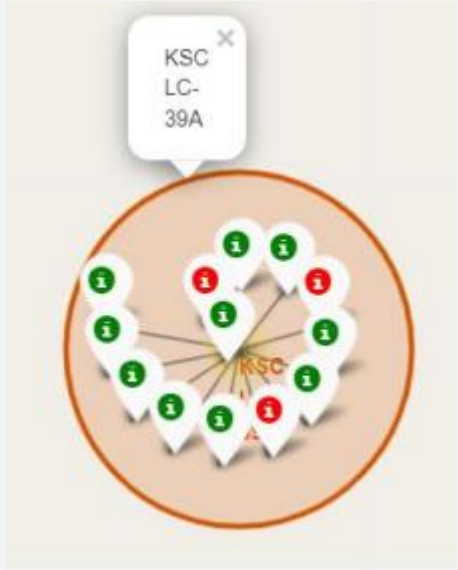
All launch sites map markers

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

Color Labelled Markers



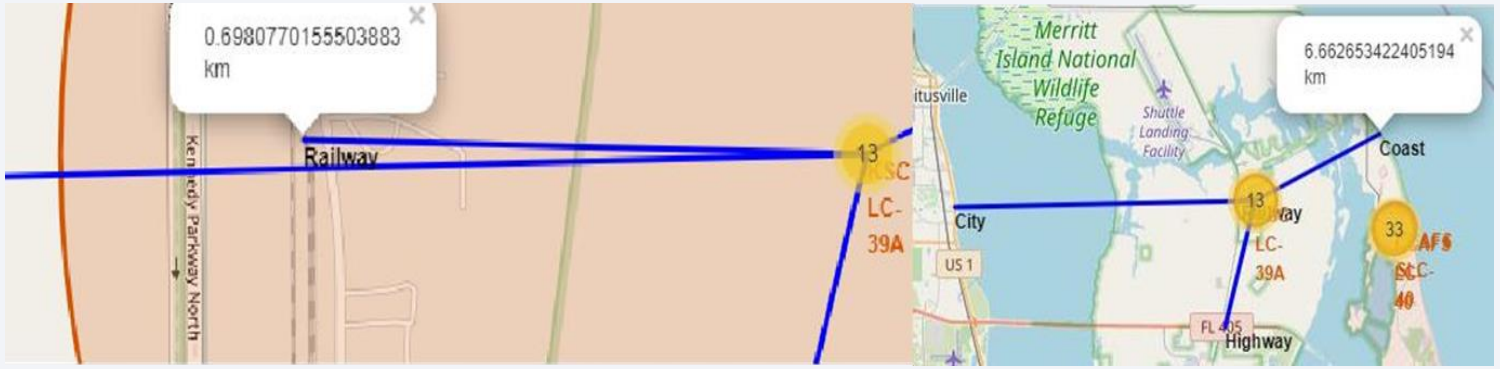




Florida Launch Sites

- ✓ Green Marker shows successful launches
- ✓ Red Marker shows unsuccessful launches

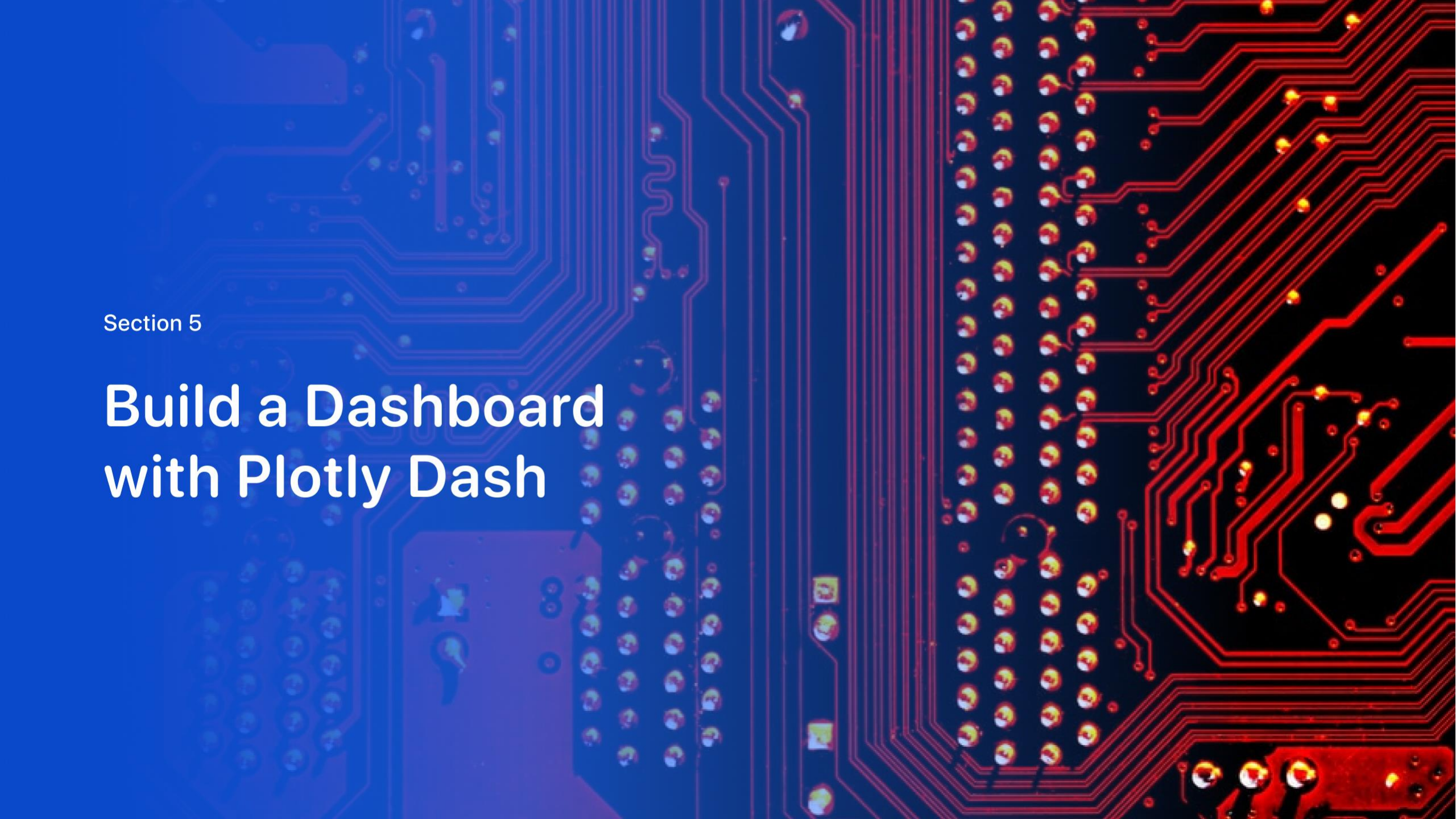
Working out Launch sites distances to landmarks to find the trends with Haversine formula using CCAFS-SLC-40 as a reference



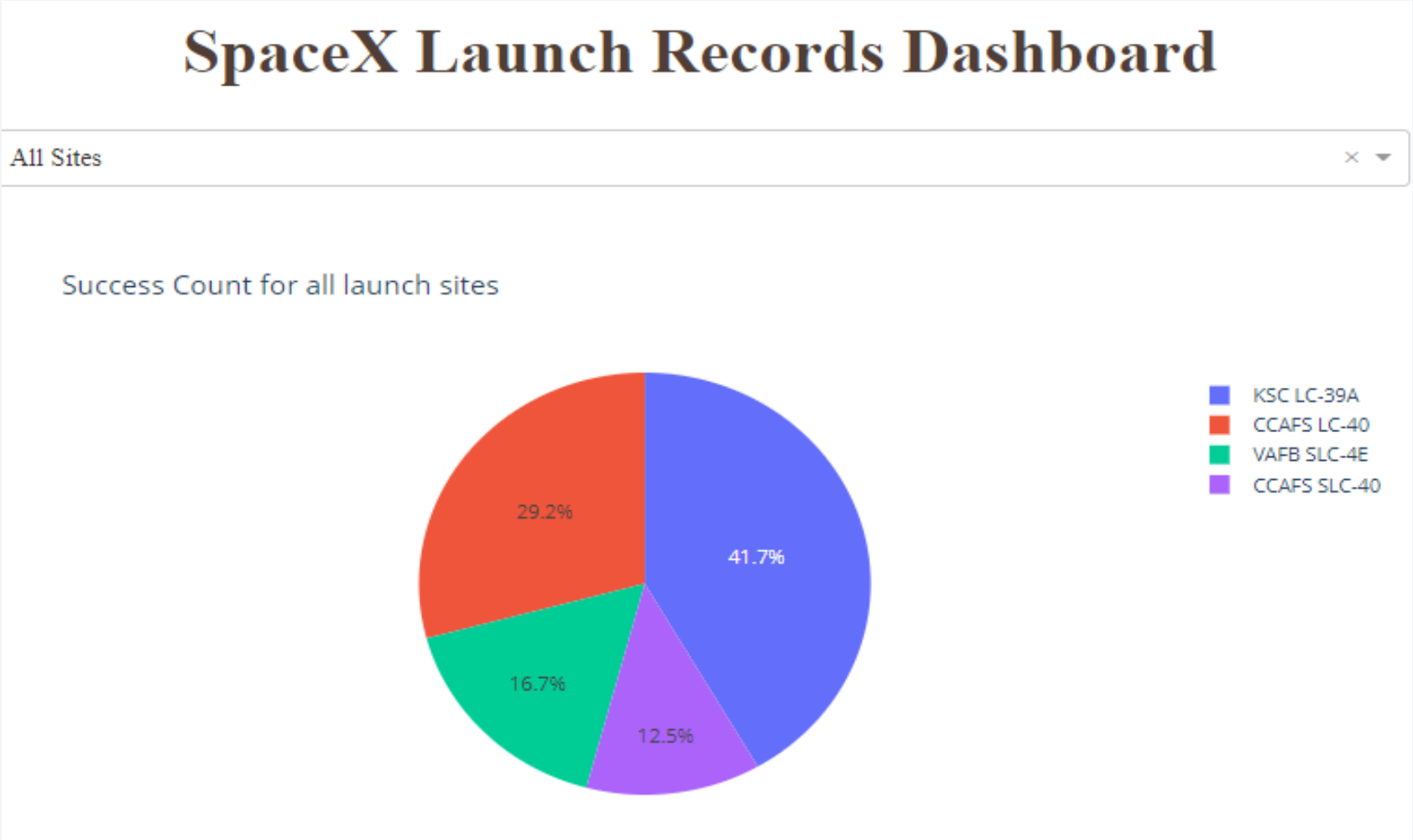
- ✓ Are launch sites in close proximity to railways? No
- ✓
- ✓ Are launch sites in close proximity to highways? No
- ✓ Are launch sites in close proximity to coastline? Yes
- ✓ Do launch sites keep certain distance away from cities? Yes

Section 5

Build a Dashboard with Plotly Dash

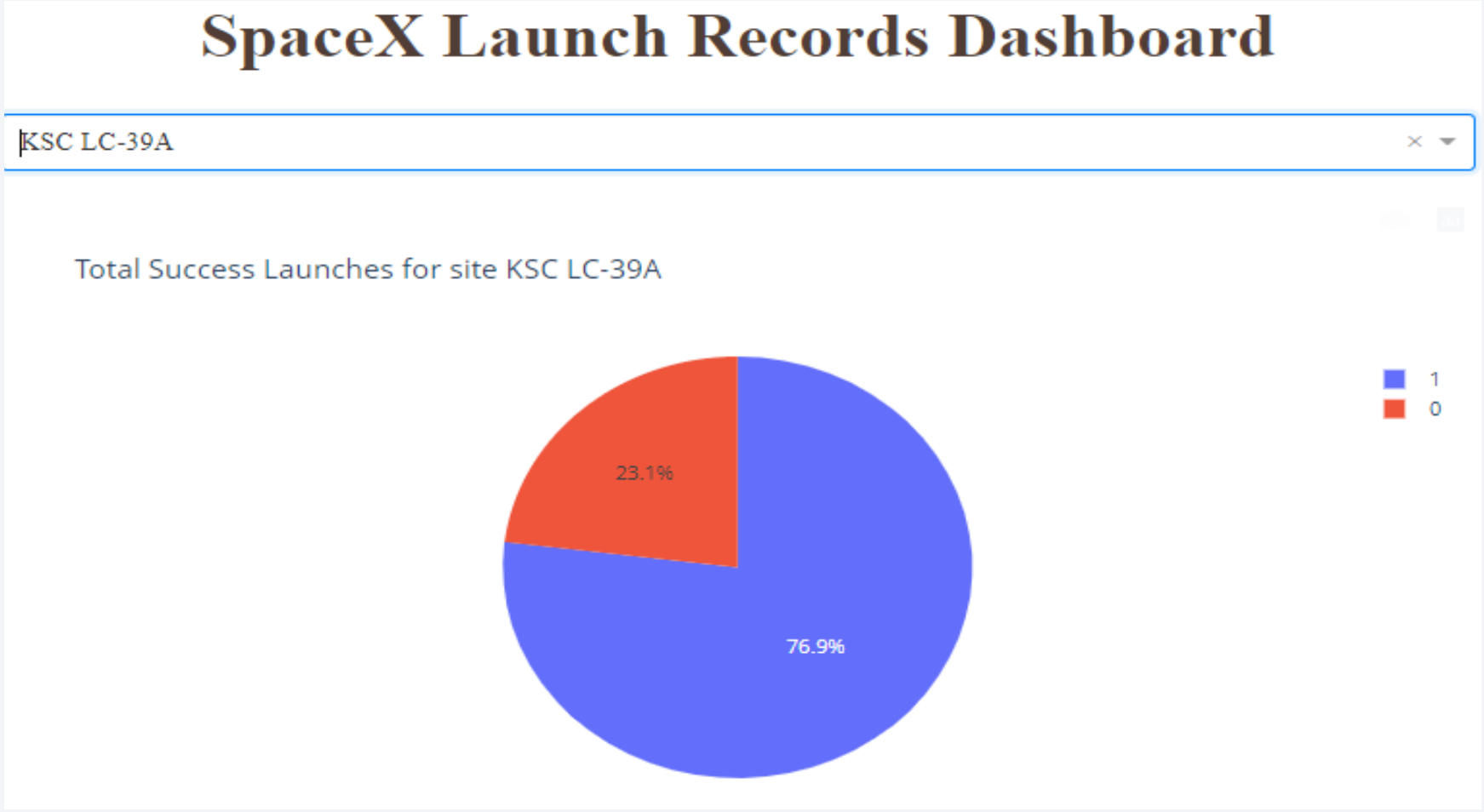


Dashboard – Pie Chart showing the success percentage achieved by each launch site



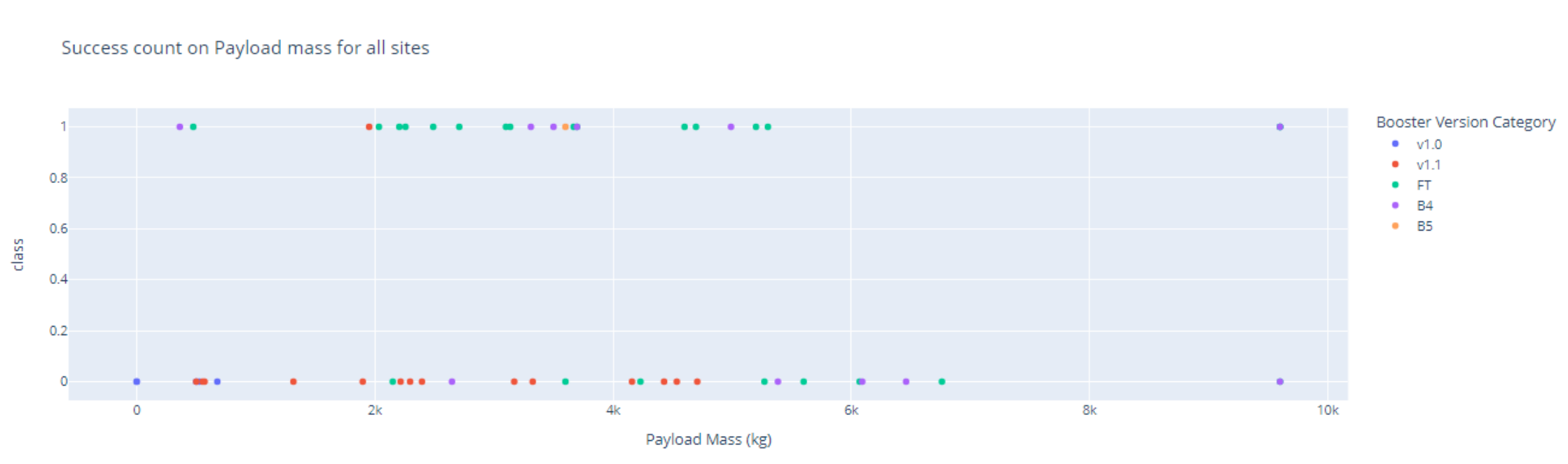
We can see that KSC LC-39A had the most successful launches from all the sites

Dashboard – Pie Chart for the launch site with heist launch success ratio



KSC LC-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



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

Section 6

Predictive Analysis (Classification)

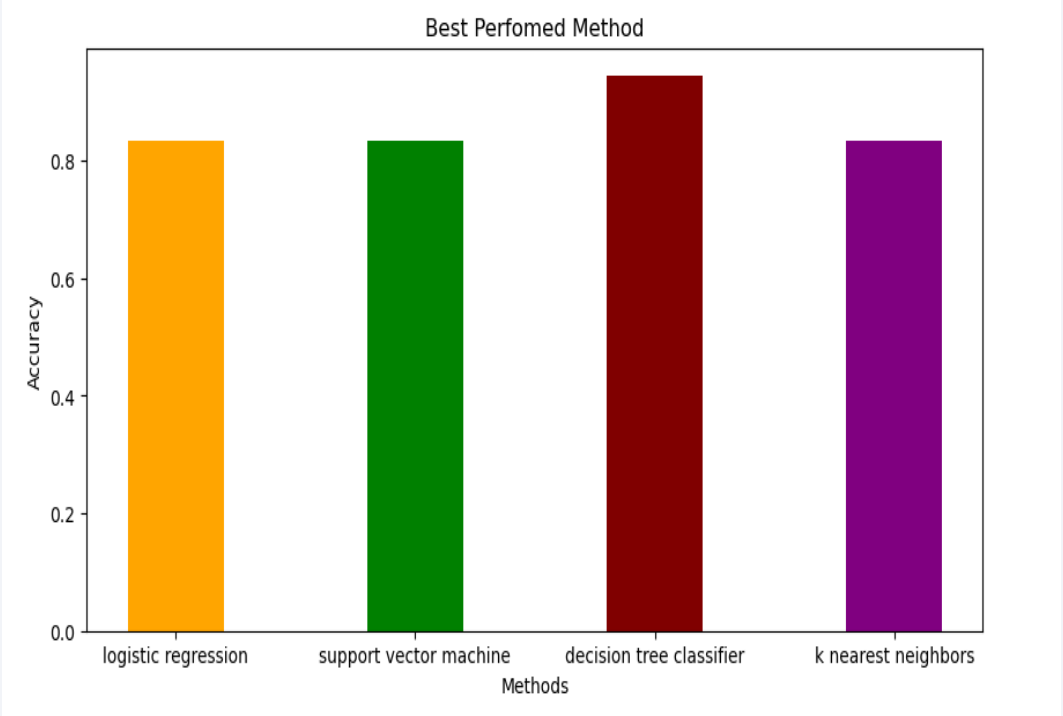
Classification Accuracy

Classification Accuracy using training data

As you can see our accuracy is extremely close but we do have a winner its down to decimal places! using this function

```
1 print(methods)
2 print(accu)
```

```
['logistic regression', 'support vector machine', 'decision tree classifier', 'k nearest neighbors']
[0.8333333333333334, 0.8333333333333334, 0.9444444444444444, 0.8333333333333334]
```



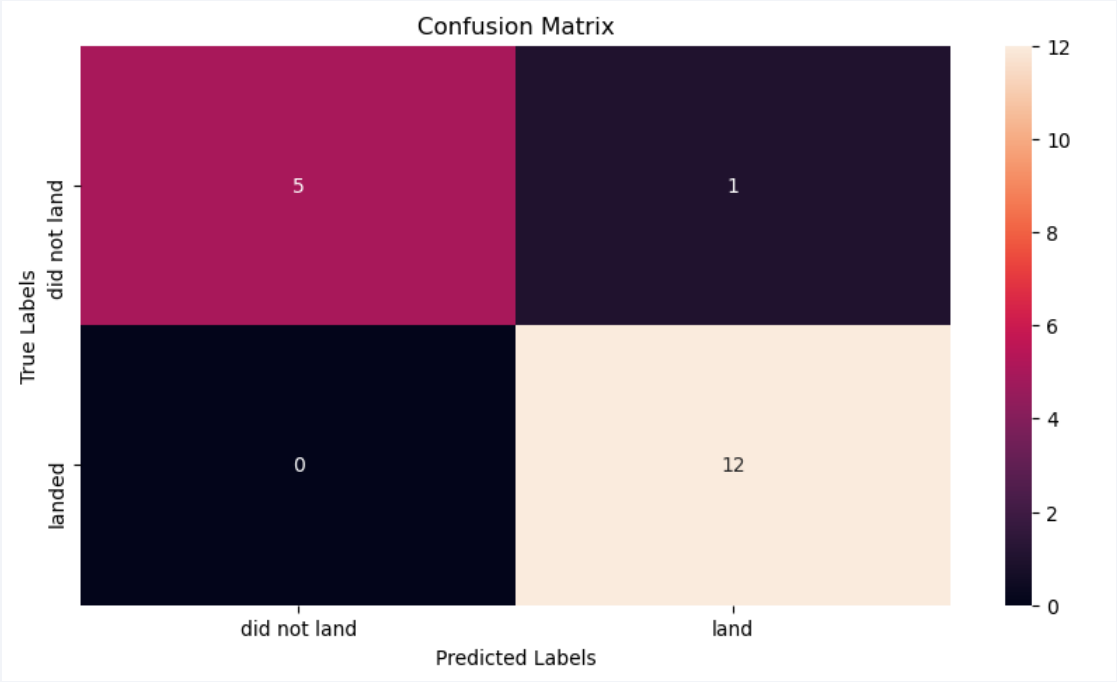
The Tree algorithm wins!!

After selecting the best hyperparameters for the decision tree classifier using the validation data, we achieved 94.44% 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.

	Predicted: NO	Predicted: YES
Actual: NO	TN = ??	FP = ??
Actual: YES	FN = ??	TP = ??



Confusion matrix for Tree Algorithm

Conclusions

- ✓Tree Classifier Algorithm is the best for Machine Learning Model
- ✓Low weighted payloads perform better than the heavier payloads
- ✓The success rates for SpaceX launches is directly proportional to time in years they will eventually perfect the launches
- ✓KSC LC-39A has the most successful launches from all Launch Sites
- ✓Orbit GEO, HEO, SSO, ES-L1 has best success rate

Appendix

1. Python Anywhere
2. Haversine Formula
3. Folium Maps
4. Coursera
5. IBM Watson Server
6. IBM Db2 Server
7. All Instructors from IBM
8. Github

Appendix

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 ADGGoogleMaps 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!