IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Fernand De La Selle
February 17, 2023

Outline

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

Executive Summary

- For this capstone project, the goal is to determine if the SpaceX Falcon 9 first stage will successfully land using machine learning algorithms to predict based off previously collected data.
- In turn, this will allow stakeholders to decide whether to invest in SpaceX for their next payload delivery into space.
- The goal has been achieved by following the included steps:
  - Data collection through webscraping and API queries
  - Exploratory data analysis
  - Data visualization
  - Machine learning predictions
- Our data shows:
  - That heavier payloads results in more successful retrieval of the first stage for orbit types polar, LEO and ISS
  - Over the past years SpaceX successful launches have increased.

Introduction

- Project background and context
  - With the recent successes of private firms launching spacecrafts, this creates an opportunity for competitors to join this industry, however, the cost of launches remains a barrier to entry.
  - SpaceX advertises that their Falcon 9 rocket launches for only 62 million dollars, while other competitors cost upwards of 165 million dollars. The reason for SpaceX to launch so cheaply is due to the reusability of the first stage. Therefore, if we can predict the success of the first stage landing, then determining the cost launch of the next rocket.
- Problems you want to find answers
  - What factors influence the successful landing of the first stage of a Falcon 9 rocket?
  - Therefore, what conditions should be met to permit the successful recovery for a given rocket?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Webscraping on SpaceX Wikipedia page
  - SpaceX API
- Perform data wrangling
  - Successful and failed launches were converted to numerical values
  - Dropped data related to rockets other than Falcon 9
  - Used one hot encoding to process categorical variables into numerical ones
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Trained various models: logistic regression, support vector machine (SVM), decisions tree, K-nearest neighbors

Data Collection

SpaceX API

- API request
  - Request launch data from SpaceX API
- Read API response
  - Read .json response and convert to a dataframe
- Preliminary data wrangling
  - Extract relevant data fields to match the requirements
- Convert to CSV
  - Convert new dataframe to CSV format for the next phase

Webscrapping data from Wiki

- Perform HTTP request
  - Request Falcon 9 html page through the GET method
- Create a Beautiful Soup object
  - Create a Beautiful Soup object from the HTML response
- Extract data to dataframe
  - Collect all relevant headers/extract rows to dataframe
- Convert to CSV
  - Convert dataframe to CSV format for the next phase

Data Collection – SpaceX API

API request

Read API response

Preliminary data wrangling

Convert to CSV

1. Create API get request and normalize data into a pandas dataframe

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

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

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

2. Obtained data from relevant API calls

```
# Call getBoosterVersion
getBoosterVersion(data)
```

```
# Call getLaunchSite
getLaunchSite(data)
```

```
# Call getPayloadData
getPayloadData(data)
```

```
# Call getCoreData
getCoreData(data)
```

3. Declared global variables and assign data from API

```
#Global variables
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

```
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}
```

4. From the newly made dataframe, data was organized for Falcon 9 launches and cleaned for null launches

```
# Create a data from launch_dict
df = pd.DataFrame(data=launch_dict)
```

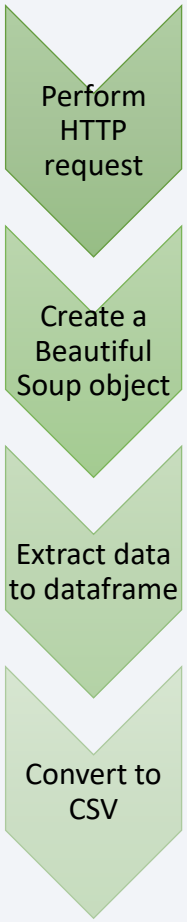
```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = df[df['BoosterVersion']!='Falcon 1']
```

```
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'][data_falcon9['PayloadMass'].isnull()] = data_falcon9['PayloadMass'].mean()
```

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

Github to file: https://github.com/AceYeti/IBM-Data-Science-Capstone-project/blob/main/SpaceX%20data%20collection%20API.ipynb

Data Collection - Scraping



1. HTTP request

```
# use requests.get() method with the provided static_url
request = requests.get(static_url)
# assign the response to a object
response = request.text
```

2. Beautiful soup object

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response)
```

```
# Use soup.title attribute
soup.title
```

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

3. Data extraction/manipulation

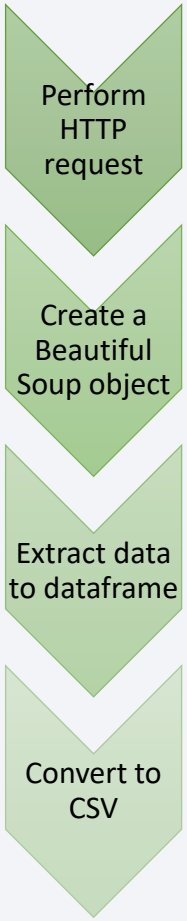
```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
data_html = first_launch_table.find_all('th')
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
for n in range(1,len(data_html)):
    column_name = extract_column_from_header(first_launch_table.find_all('th')[n])
    if column_name != None and len(column_name)>0:
        column_names.append(column_name)
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
```

```
print(column_names)
```

```
['Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome']
```

Github to file: https://github.com/AceYeti/IBM-Data-Science-Capstone-project/blob/main/SpaceX%20data%20webscraping.ipynb

Data Collection - Scraping



4. Assign to dataframe and convert to .csv

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

	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Flight No.	Version Booster	Booster landing	Date	Time
0	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	[[SpaceX], \n]	Success\n	1	F9 v1.0B0003.1	Failure	4 June 2010	18:45
1	CCAFS	Dragon	0	LEO	[[.mw-parser-output .plainlist ol,.mw-parser-o...	Success	2	F9 v1.0B0004.1	Failure	8 December 2010	15:43
2	CCAFS	Dragon	525 kg	LEO	[[NASA], (, [COTS],)\n]	Success	3	F9 v1.0B0005.1	No attempt\n	22 May 2012	07:44
3	CCAFS	SpaceX CRS-1	4,700 kg	LEO	[[NASA], (, [CRS],)\n]	Success\n	4	F9 v1.0B0006.1	No attempt	8 October 2012	00:35
4	CCAFS	SpaceX CRS-2	4,877 kg	LEO	[[NASA], (, [CRS],)\n]	Success\n	5	F9 v1.0B0007.1	No attempt\n	1 March 2013	15:10

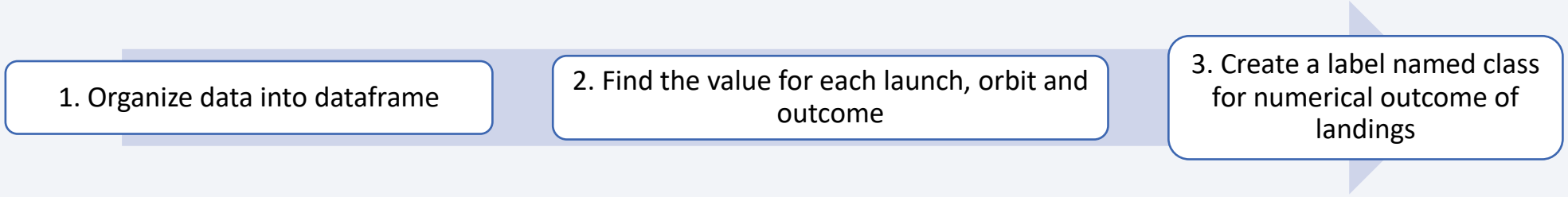
```
df.to_csv('spacex_web_scraped.csv', index=False)
```


Github to file: https://github.com/AceYeti/IBM-Data-Science-Capstone-project/blob/main/SpaceX%20data%20webscraping.ipynb

Data Wrangling

- We used exploratory data analysis (EDA) to find some pattern in the data which later will be used for training supervised models
- We have created a label called “class” which summarizes the outcome of each flight as 1 or 0 from a landing in the ocean, on the ground (RTLS) or on a drone ship (ASDS)

- The process went as followed:



Data Wrangling

1. Organize data into dataframe

2. Find the value for each launch, orbit and outcome

3. Create a label named class for numerical outcome of landings

1.

```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/datas
df.head(10)
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Se
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0(
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0(

2.1

```
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()
```

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

```
# Apply value_counts on Orbit column
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
```

```
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
```

2.2

```
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 False Ocean
6 None ASDS
7 False RTLS
```

2.3

```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

3

```
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

Github link to file: https://github.com/AceYeti/IBM-Data-Science-Capstone-project/blob/main/SpaceX%20data%20wrangling.ipynb

EDA with Data Visualization

- To further understand the data we have collected, we developed graphs (scatter, bar, and line).

- Scattered plot
  - Used to identify the relationship between two variables
  - Ex. Payload and orbit type

- Bar graph
  - Used to compare values from different groups. The magnitude of the difference between variables is easily observable with the height of each bar.
  - Ex. Success rate and orbit type

- Line graph
  - Used to track the changes of a variable over time thereby depicting trends or patterns
  - Ex. Average launch success over time

Github link to file: https://github.com/AceYeti/IBM-Data-Science-Capstone-project/blob/main/SpaceX%20data%20visualization.ipynb

EDA with SQL

- SQL queries were used to gain insight on the SpaceX dataset using **SQLite**. Here is a summary of the queries used

1. Display the names of the unique launch sites in the space mission
2. Display 5 records where launch sites begin with the string '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 succesful landing outcome in ground pad was achieved
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. Use a subquery
9. 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
10. Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order

Github link to file: https://github.com/AceYeti/IBM-Data-Science-Capstone-project/blob/main/SpaceX%20EDA%20SQL.ipynb

Build an Interactive Map with Folium

- Folium is a python library that enables the user to use maps to visualize geographical locations and analyze geospatial data to better represent distances between points of interests.
- All launch sites were marked with circles to visualize their location compared to each other on a map
- Within markers, marker clusters were added to keep track of the number of flights per launch sites and they were color coded green and red to indicate if the launch was successful
- Calculated the distance between launch sites and proximities (ie. coastline, railroad, city, highway) to show the accessibility of the pad or dangers of launches.

Github to file: https://github.com/AceYeti/IBM-Data-Science-Capstone-project/blob/main/SpaceX%20launch%20site%20location.ipynb

Build a Dashboard with Plotly Dash

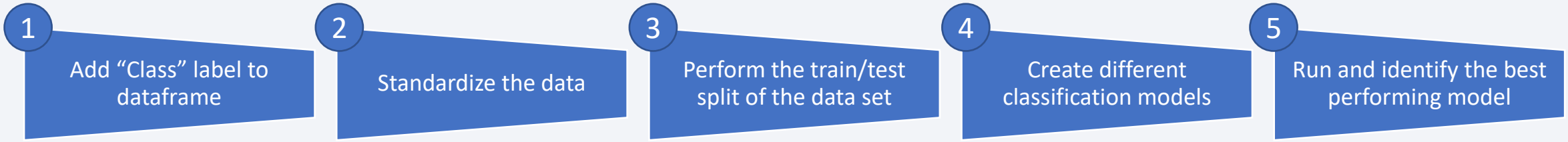
- With plotly dash an interactive dashboard was generated to perform analytics using the SpaceX data.

- A dropdown menu to enable launch site selection
- A pie chart showing the total successful launches
- A slider to select payload range
- A scatter plot of payload and launch success

- These interactions and charts allows for a better understanding of the data in real time and makes visualization easier

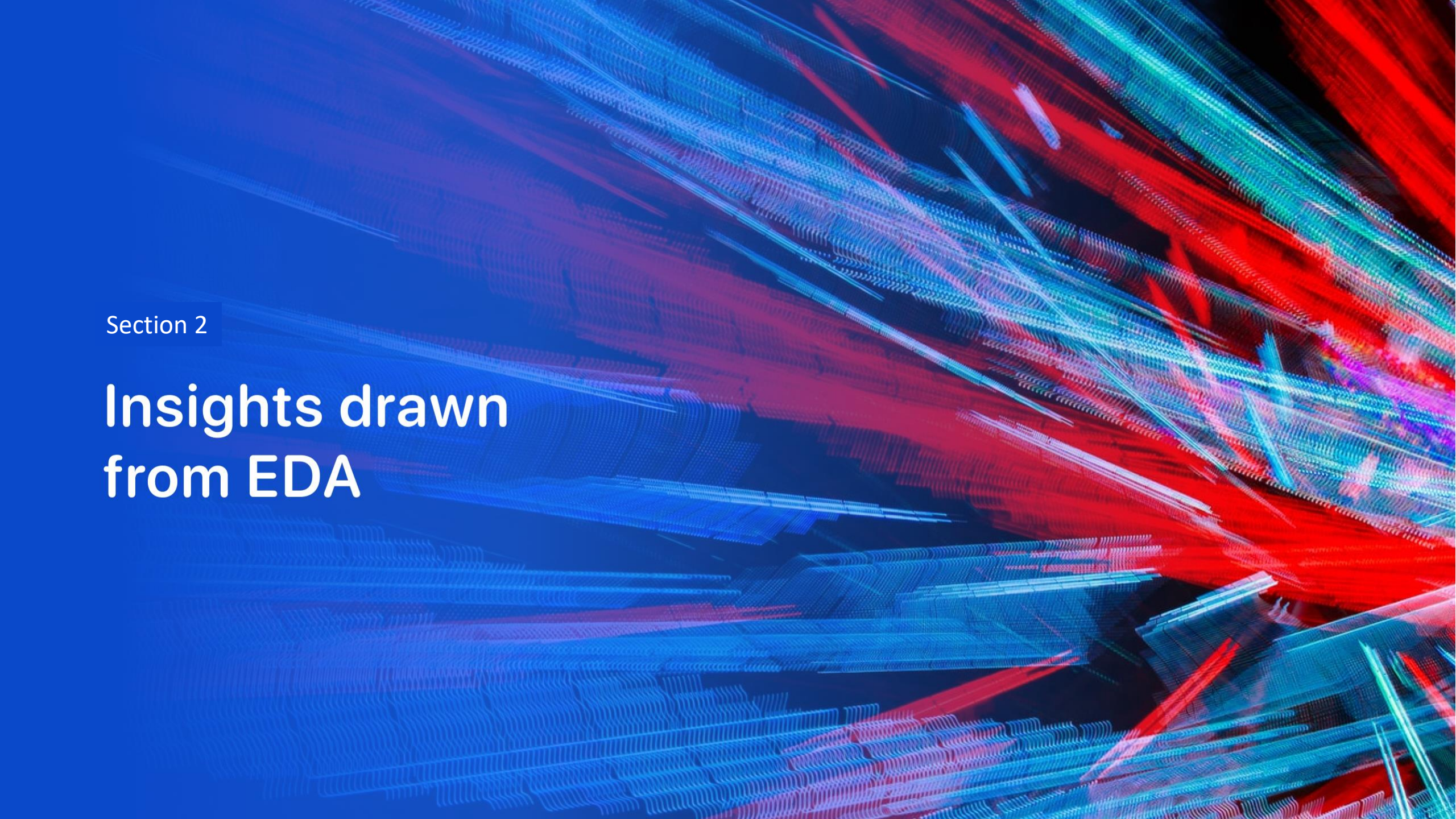
Github to file: https://github.com/AceYeti/IBM-Data-Science-Capstone-project/blob/main/SpaceX_dash_app.py

Predictive Analysis (Classification)

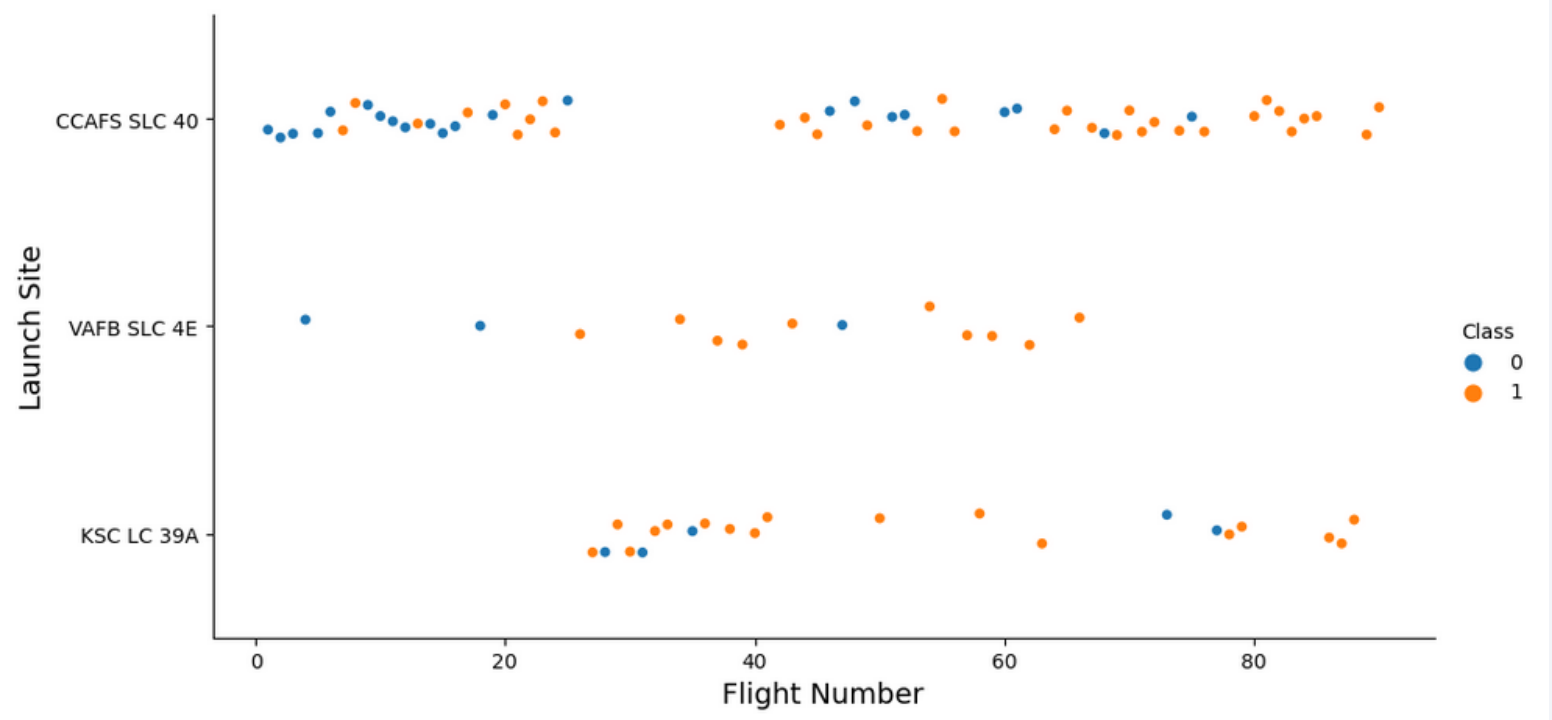


1. Added the class label to the dataframe and converted the class data into a numpy array
2. For preprocessing we used the standard scaler to standardize the X values
3. Train/test split performed with the train_test_split function where the test size was equal to 20% of the data set
4. The different models used were:
    1. Logistic regression
    2. Support vector machine
    3. Decision tree
    4. K-nearest neighbors
5. The different models were run, and the accuracy scores were calculated

Github link to file: https://github.com/AceYeti/IBM-Data-Science-Capstone-project/blob/main/SpaceX%20Machine%20Learning%20Prediction.ipynb

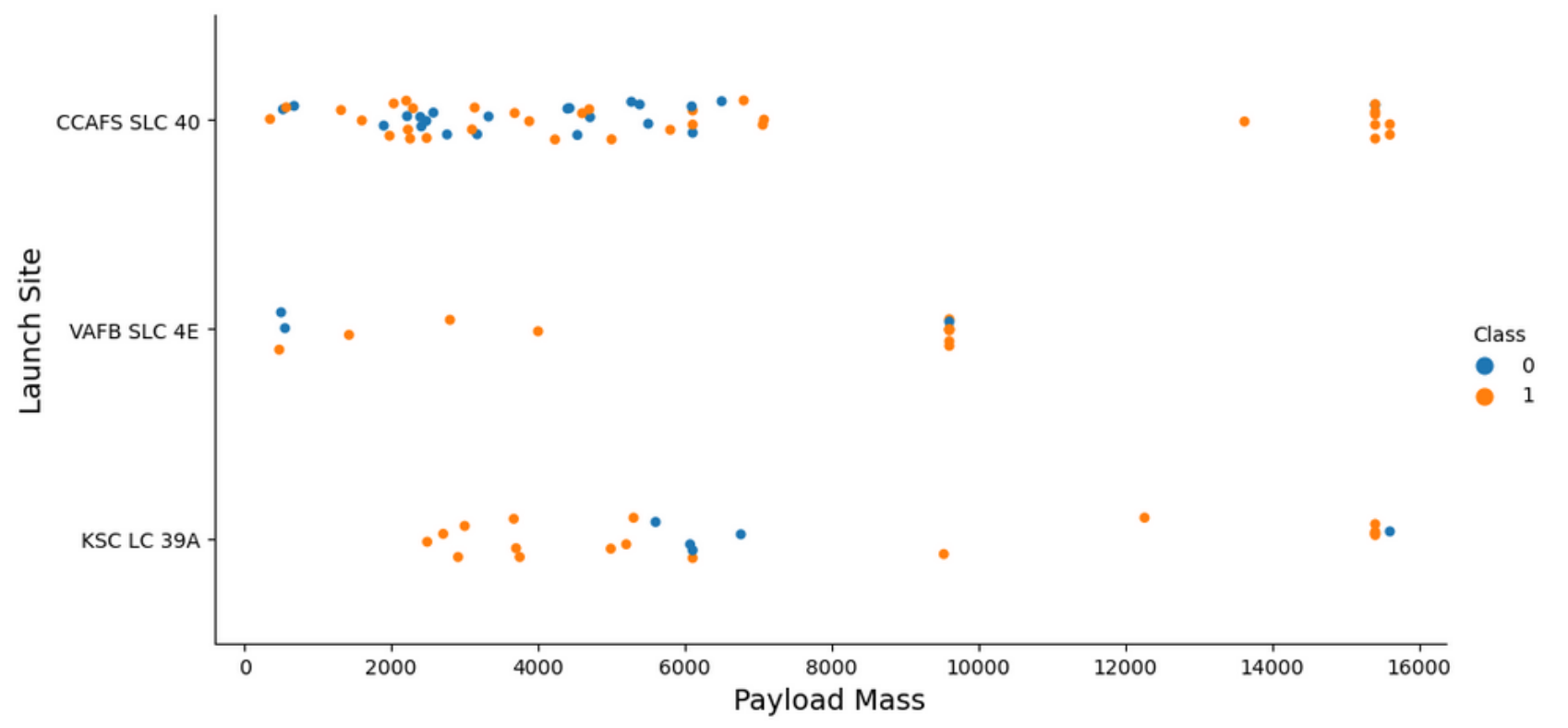

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



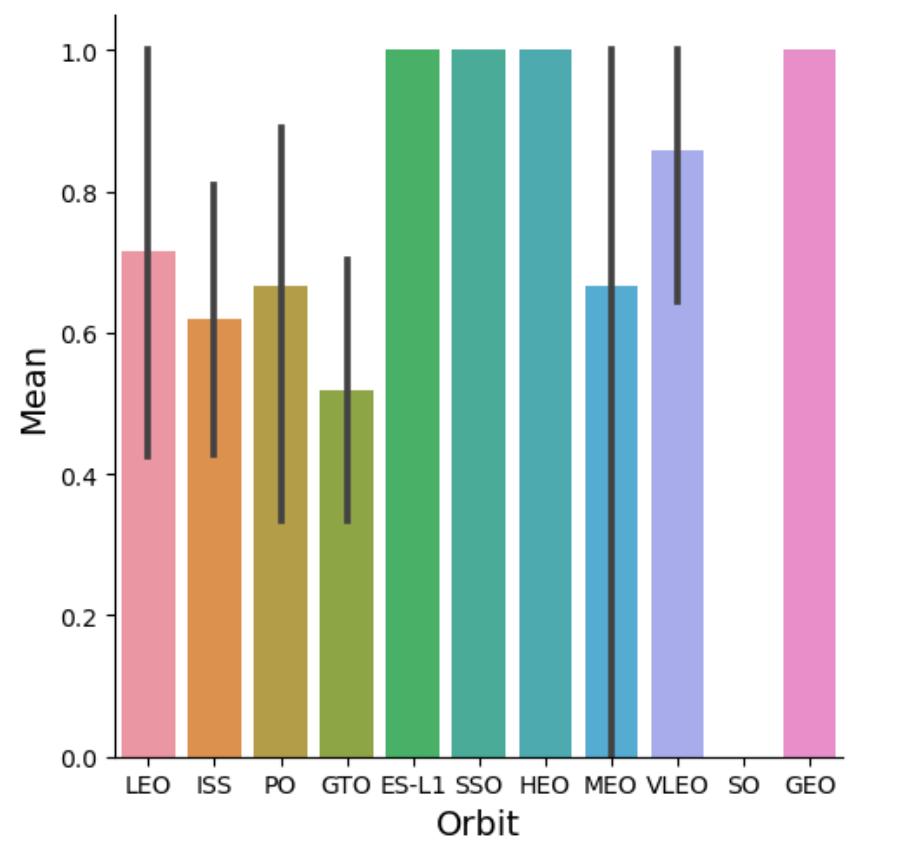
- Class represents the success (class=1) of each flight. As flight number increases the success of flights also increases for the 3 launch pads
- KSC LC 39A shows consistent successful launches from flight number 33

Payload vs. Launch Site



- CCAFS SLC 40 shows a relationship with heavier payload and success of the launch
- VAFB SLC 4E launched payloads no larger than 10,000kg
- KSC LC 39A shows more successful launches with payload lower than 6000kg

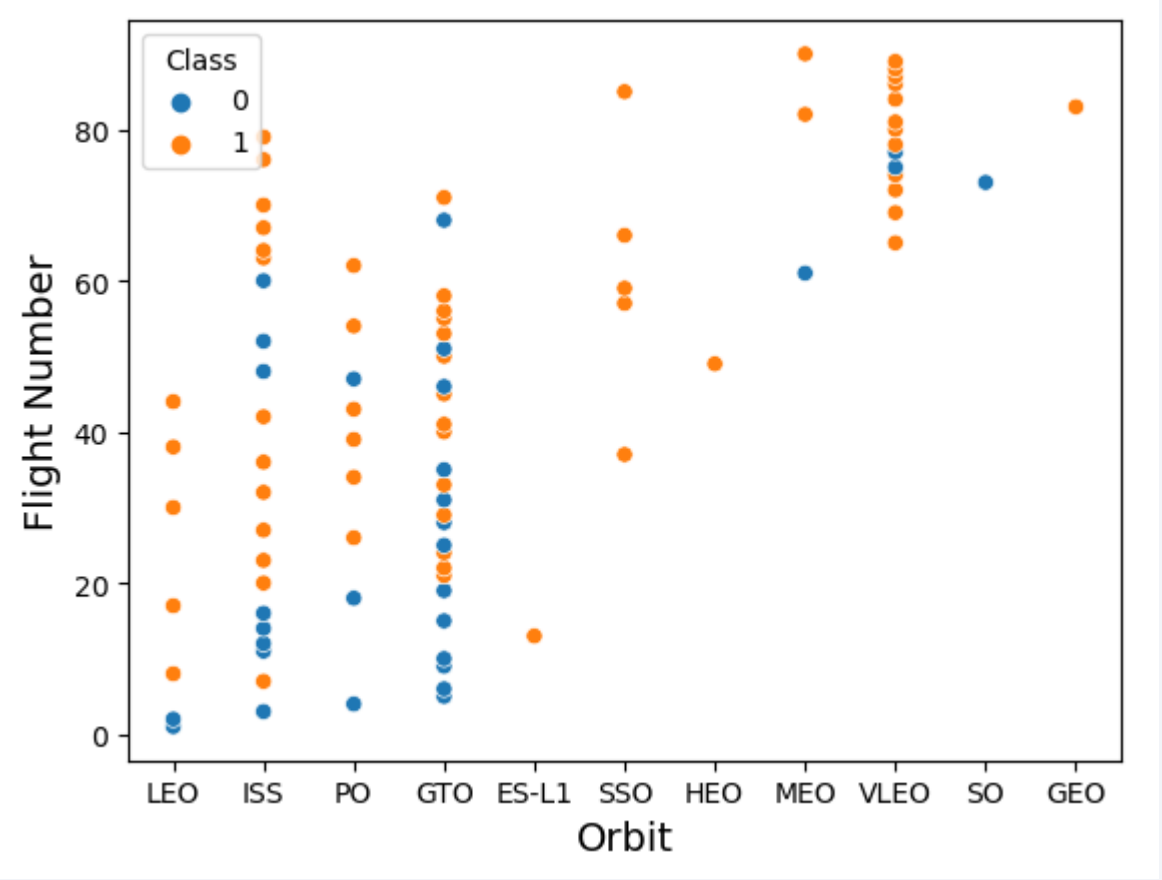
Success Rate vs. Orbit Type



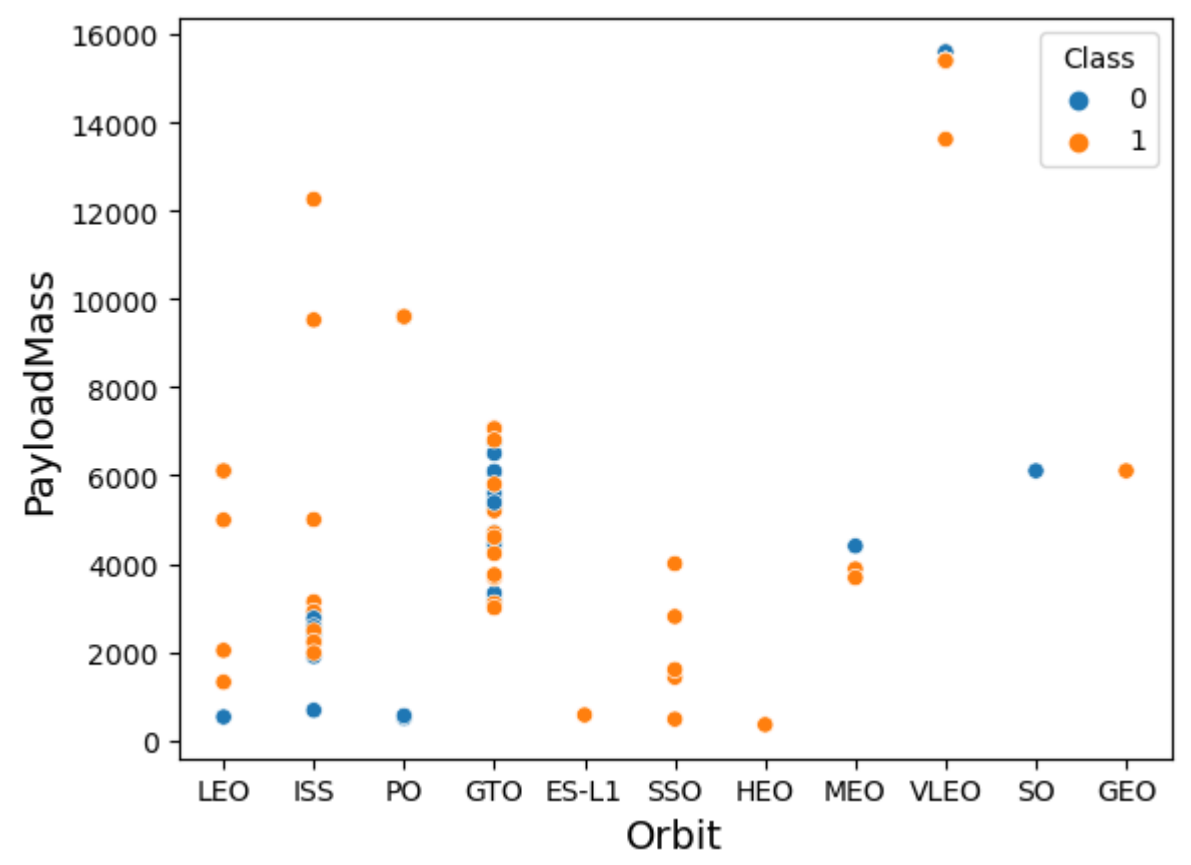
- ES-L1, SSO, HEO, and GEO orbits show the highest success rate compared to the other orbits
- GTO orbit has the lowest success rate
- SO orbit obtained no success
- MEO orbit show the most variability in success

Flight Number vs. Orbit Type

- Generally it appears that as flight number increases the different orbit launches become more successful (ie. LEO, ISS, PO, SSO, MEO, VLEO)
- The GTO orbit does not seem to follow the same trend



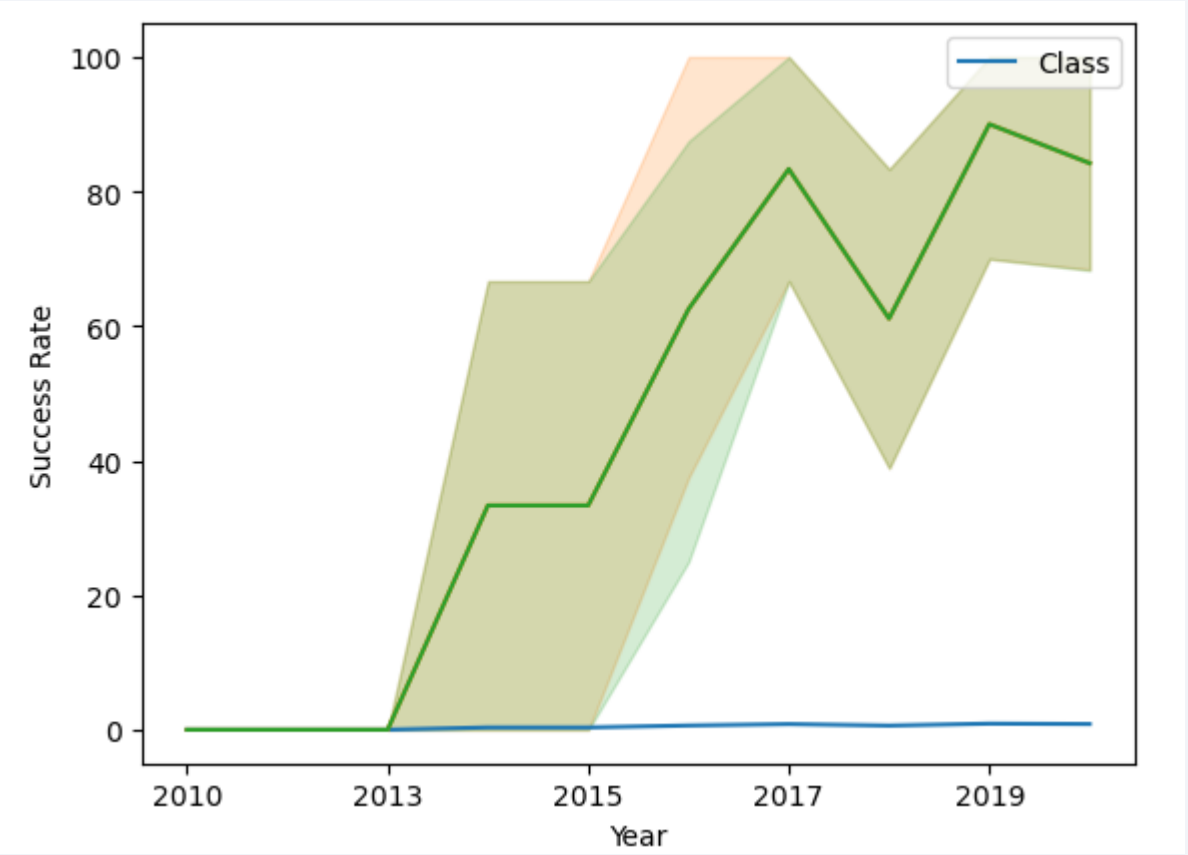
Payload vs. Orbit Type



- Generally, success rate increases with payload mass (except for VELO, MEO, SO, GTO)
- VELO is the orbit that successfully launched the highest payload

Launch Success Yearly Trend

- There is a clear improvements between time and success rate between 2013 and 2017
- After experiencing a downfall in 2018, 2019 and 2020 are the most successful recorded years



All Launch Site Names

- Query

```
%sql select distinct Launch_Site from SPACEXTBL
```

- Result

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

- Explanation: the distinct function returns the unique values found in Launch_Site

Launch Site Names Begin with 'CCA'

- Query

```
%sql select * from SPACEXTBL WHERE launch_site LIKE'CCA%' LIMIT 5
```

- Results

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

- Explanation: the sql command “like” and the formatting command ‘CCA%’ will return records that begin with CCA in the launch site column

Total Payload Mass

- Query

```
%sql select sum(payload_mass__kg_) from SPACEXTBL where customer LIKE 'NASA (CRS)'
```

- Result

sum(payload_mass__kg_)
45596

- Explanation: the sum function adds all the values for payload mass into a sum when the condition customer = ‘NASA (CRS)’ is met

Average Payload Mass by F9 v1.1

- Query

```
%sql select avg(payload_mass__kg_) FROM SPACEXTBL WHERE booster_version LIKE 'F9 v1.1'
```

- Result

avg(payload_mass__kg_)
2928.4

- Explanation: payload mass values are averaged for the cells containing booster version= F9 v1.1

First Successful Ground Landing Date

- Query

```
%sql Select MIN(DATE) as "FIRST_SUCCESS" FROM SPACEXTBL where Landing_Outcome= 'Success (ground pad)'
```

- Result

FIRST_SUCCESS
01-05-2017

- Explanation: the min function on date will retrieve the earliest date when landing outcome = Success (ground pad)

Successful Drone Ship Landing with Payload between 4000 and 6000

- Query

```
%%sql SELECT BOOSTER_VERSION, payload_mass__kg_, Landing_Outcome  FROM SPACEXTBL
where 4000 < payload_mass__kg_ and payload_mass__kg_ < 6000 and Landing_Outcome = 'Success (drone ship)'
```

- Result

Booster_Version	PAYLOAD_MASS__KG_	Landing_Outcome
F9 FT B1022	4696	Success (drone ship)
F9 FT B1026	4600	Success (drone ship)
F9 FT B1021.2	5300	Success (drone ship)
F9 FT B1031.2	5200	Success (drone ship)

- Explanation: we are looking to output booster version, payload mass and landing outcome when 3 conditions are met:
  1. Payload mass is greater than 4000
  2. Payload mass is less than 6000
  3. Landing outcome is ‘Success (drone ship)’

Total Number of Successful and Failure Mission Outcomes

- Query

```
%sql select Mission_Outcome, count(Mission_Outcome) as "Total" FROM SPACEXTBL Group by Mission_Outcome
```

- Result

Mission_Outcome	Total
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

- Explanation: the function ‘group by’ arranges data in a column into a group and the function count enumerates the different outputs according to the number of times they occur

Boosters Carried Maximum Payload

- Query

```
%%sql SELECT booster_version,payload_mass__kg_ from SPACEXTBL
WHERE payload_mass__kg_ = (SELECT max(payload_mass__kg_) FROM SPACEXTBL)
```

- Result

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

- Explanation: sub query return the maximum payload mass. So the output shows the booster version with the highest payload mass in the output

2015 Launch Records

- Query

```
%%sql select landing_outcome, booster_version, launch_site FROM SPACEXTBL
WHERE landing_outcome LIKE 'Failure (drone ship)' and year(Date)= '2015'
```

- Result

```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) no such function: year
[SQL: select landing_outcome, booster_version, launch_site FROM SPACEXTBL
WHERE landing_outcome LIKE 'Failure (drone ship)' and year(Date)= '2015']
(Background on this error at: http://sqlalche.me/e/e3q8)
```

- Explanation:
  - SQLite doesn't seem to understand the year function despite being taught this function in the course. I would have to investigate this further.

- Ideally:
  - The output lists landing outcome, booster version, and launch site when the landing outcome is set to 'Failure (drone ship)' and the year is set to 2015
  - The year function extracts the year value from the date

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

- Query

```
%%sql select count(landing_outcome) as COUNT, landing_outcome from SPACEXTBL
WHERE DATE > '2010-06-04' and DATE<'2017-03-20' GROUP BY landing_outcome ORDER BY COUNT DESC
```

- Result

COUNT	Landing_Outcome

- Explanation:
  - No error is shown, but no data is shown either only the titles, meaning the query was processed correctly. It seems the dates have different formatting in the data which may render them unreadable. I would have to look more into it.

- Ideally:
  - As mentioned previously, the group by function arranges the output from column landing outcome
  - Then a date condition is set between 2010-06-04 and 2017-03-20
  - Finally, the data is ranked in descending order, with the highest occurrence as the first output

Section 3

Launch Sites Proximities Analysis

Launch sites for SpaceX Falcon9 rockets

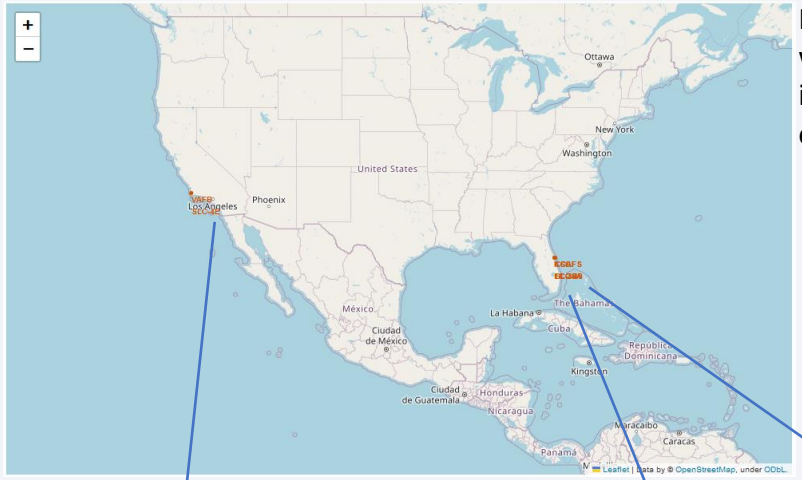


Figure 1: Global map with markers indicating location of space centers

- This map displays the location where Falcon 9 rockets are launched. These locations are located in the southern united states close near the coast.
- Florida:
  - KSC LC 39A
  - CCAFS SLC 40
  - CCAFS LC 40
- California:
  - VAFB SLC 4E

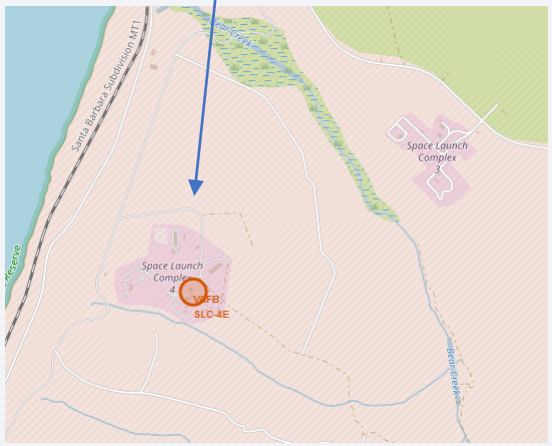


Figure 2: VAFB SLC 4E

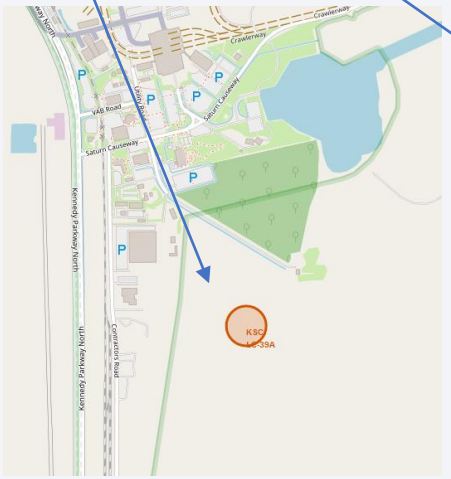


Figure 3: KSC LC 39A

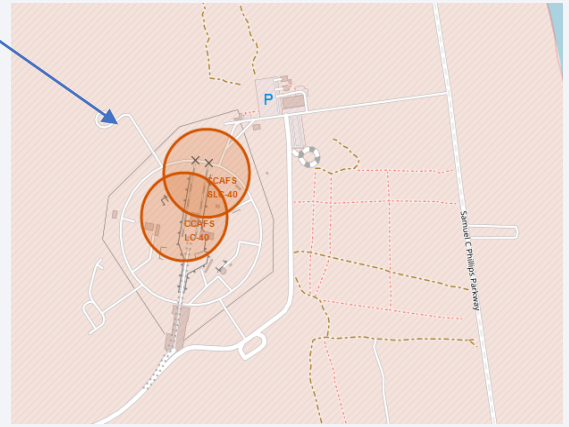


Figure 4: CCAFS SLC 40 and CCAFS LC 40

Launch Sites success of SpaceX Falcon9 rockets

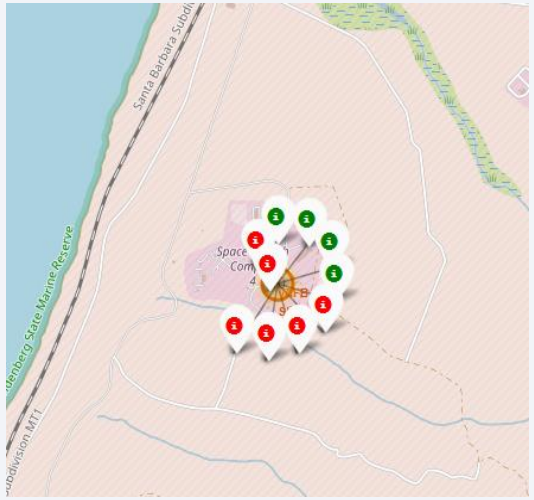


Figure 1: VAFB SLC 4E (California)

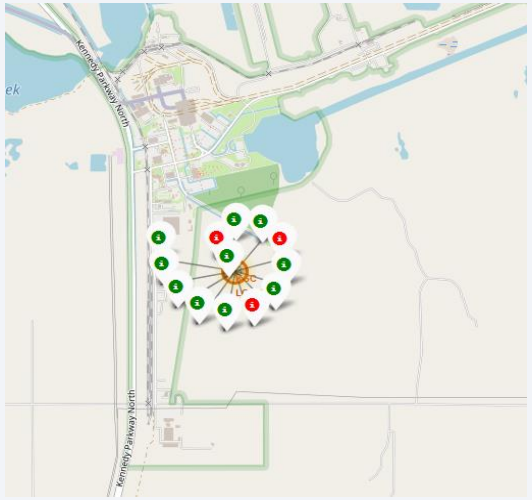


Figure 2: KSC LC 39A (Florida)

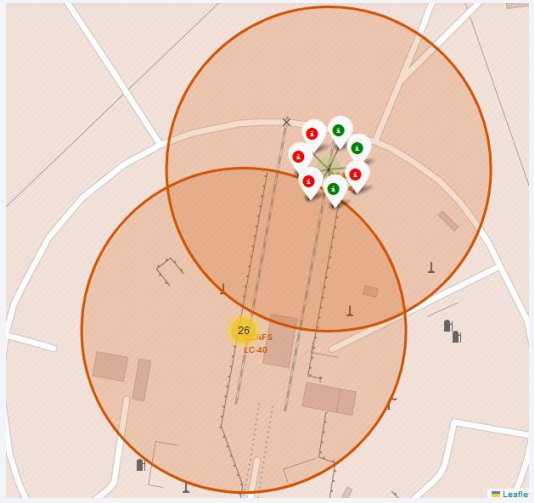


Figure 3: CCAFS SLC 40 (Florida)

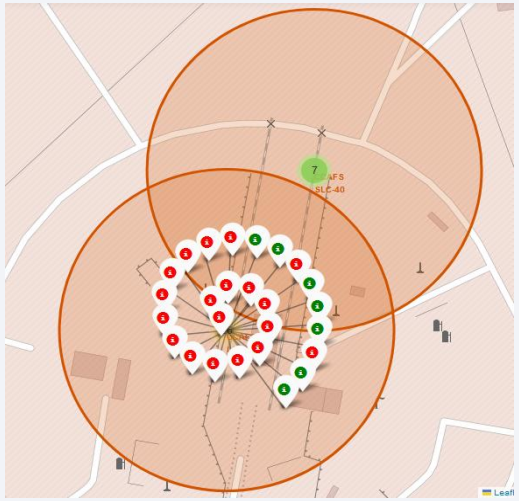


Figure 4: CCAFS LC 40 (Florida)

- Markers indicate success of launches where the center are the earlier launches versus the end of the spiral representing the latest launches
- KSC LC 39A shows the most number of successful launches
- CCAFS LC 40 shows the greatest improvement in successful launches

Proximities to Falcon9 launch sites

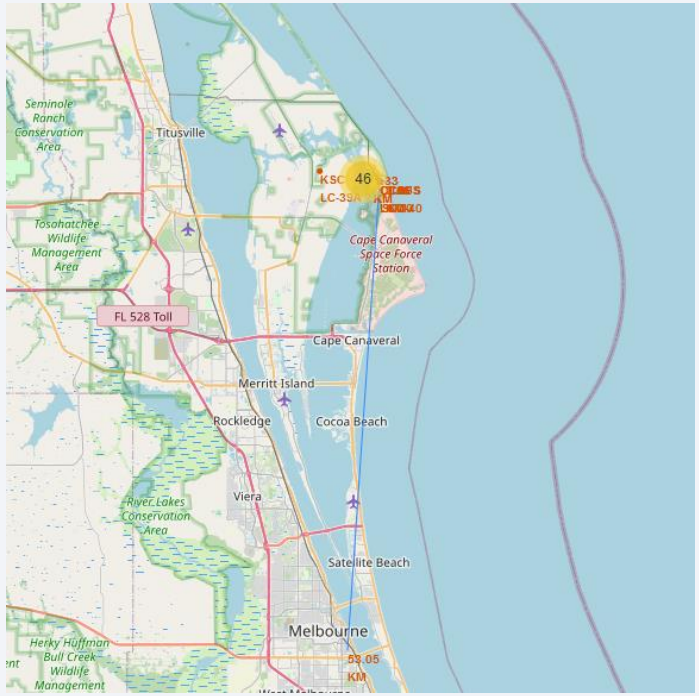


Figure 1: Distance from CCAFS LC 40 to Melbourne (53km)

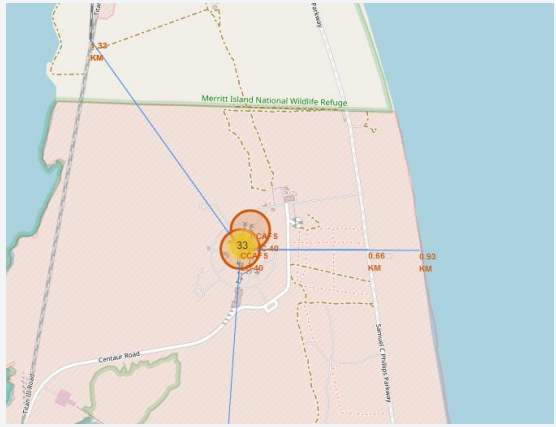
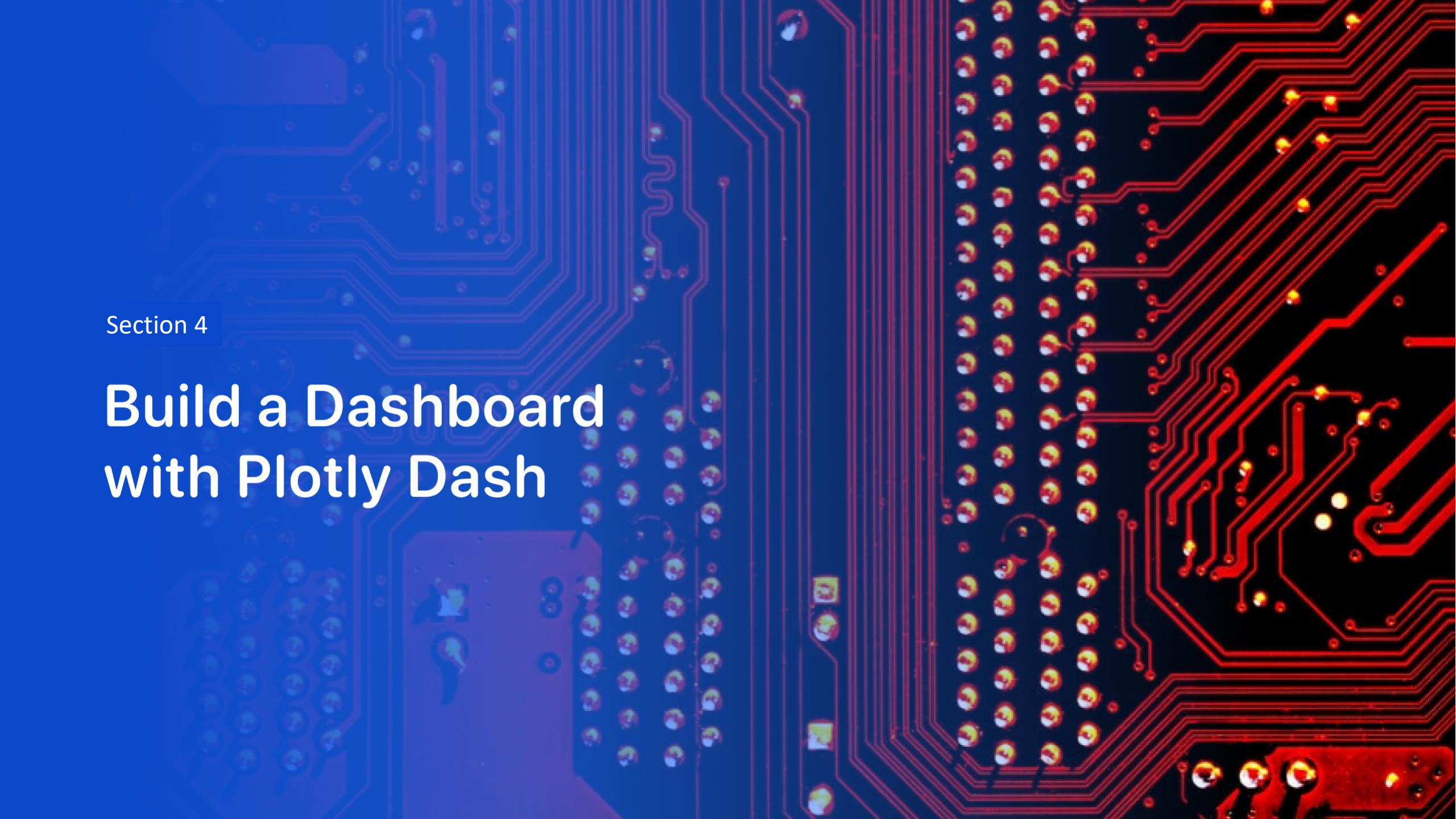
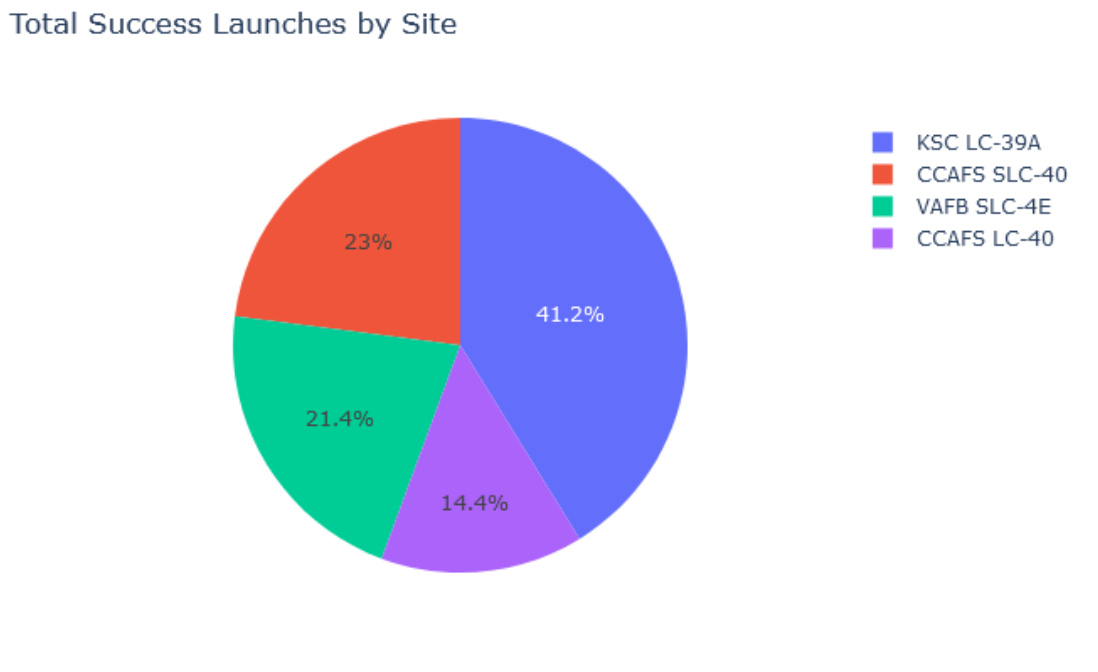


Figure 2: Distance from CCAFS LC 40 to the coast (0.93km) and a railway (1.33km)

- The CCAFS LC 40 is
  - 53km away from Melbourne
  - 1.33km away from a railroad
  - 0.93km away from the coast
- Launch sites need
  - To be away from cities due to the hazard of rockets
  - To be close to railways and coastlines for transport and accessibility of resources

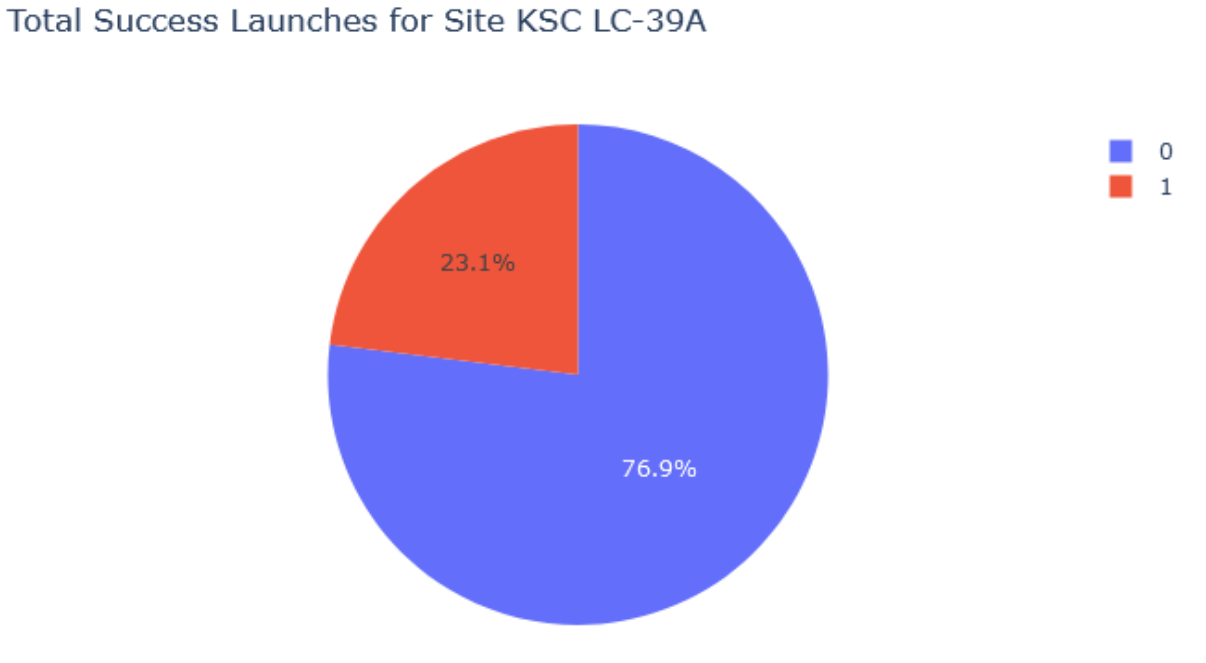

Section 4
Build a Dashboard with Plotly Dash

Launch Success Count According to Sites



- KSC LC 39A has the highest launch success rate
- CCAFS LC 40 has the lowest launch success rate

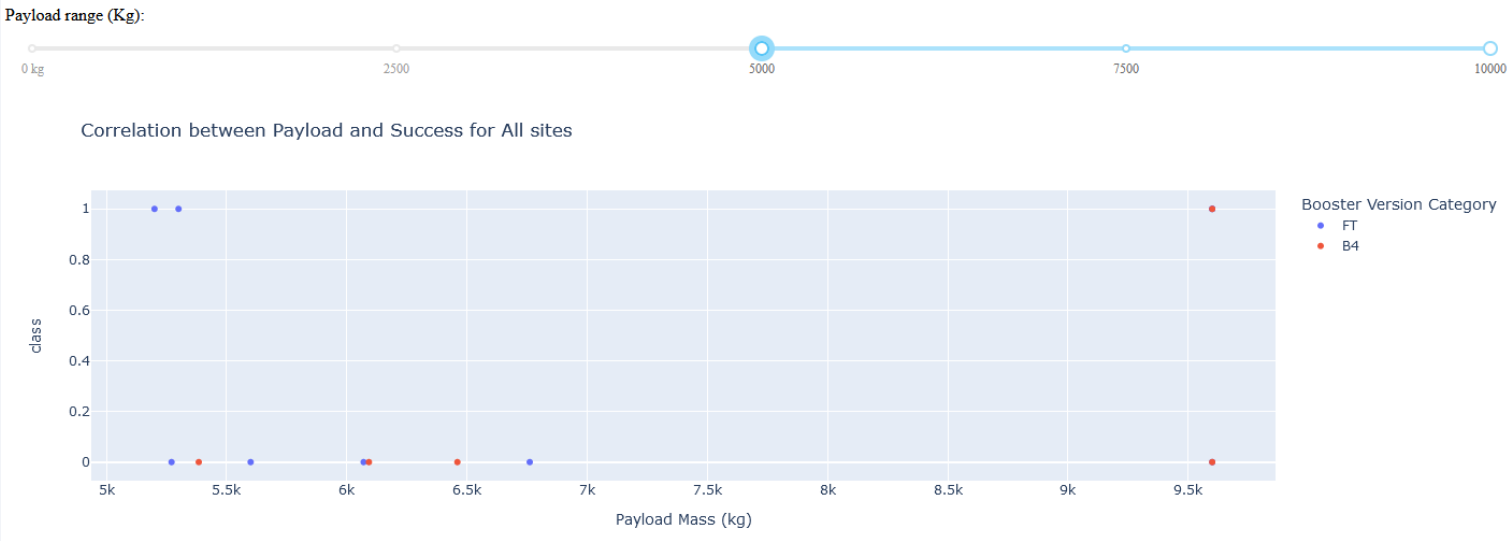
KSC LC 39A has the highest probability of success



The launch success/failure breakdown is as followed

- 76.9% success
- 23.1% failure

Comparison of launch outcome with payload mass and booster version

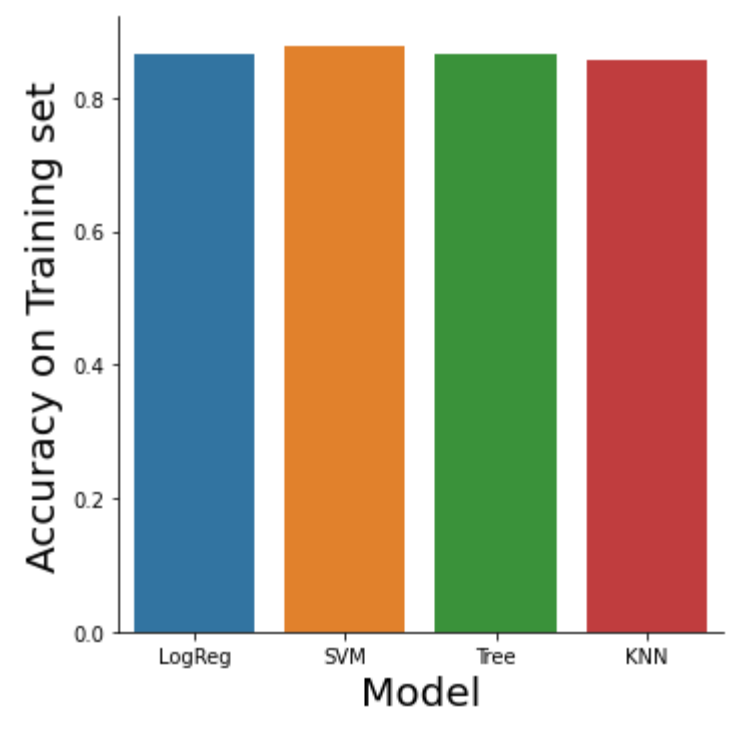


- Booster version B4 brought the highest payload successfully
- Booster version FT were the most successful
- Booster version 1.1 was the least successful
- The most successful payload mass range is between 2000-5000kg

Section 5

Predictive Analysis (Classification)

Classification Accuracy

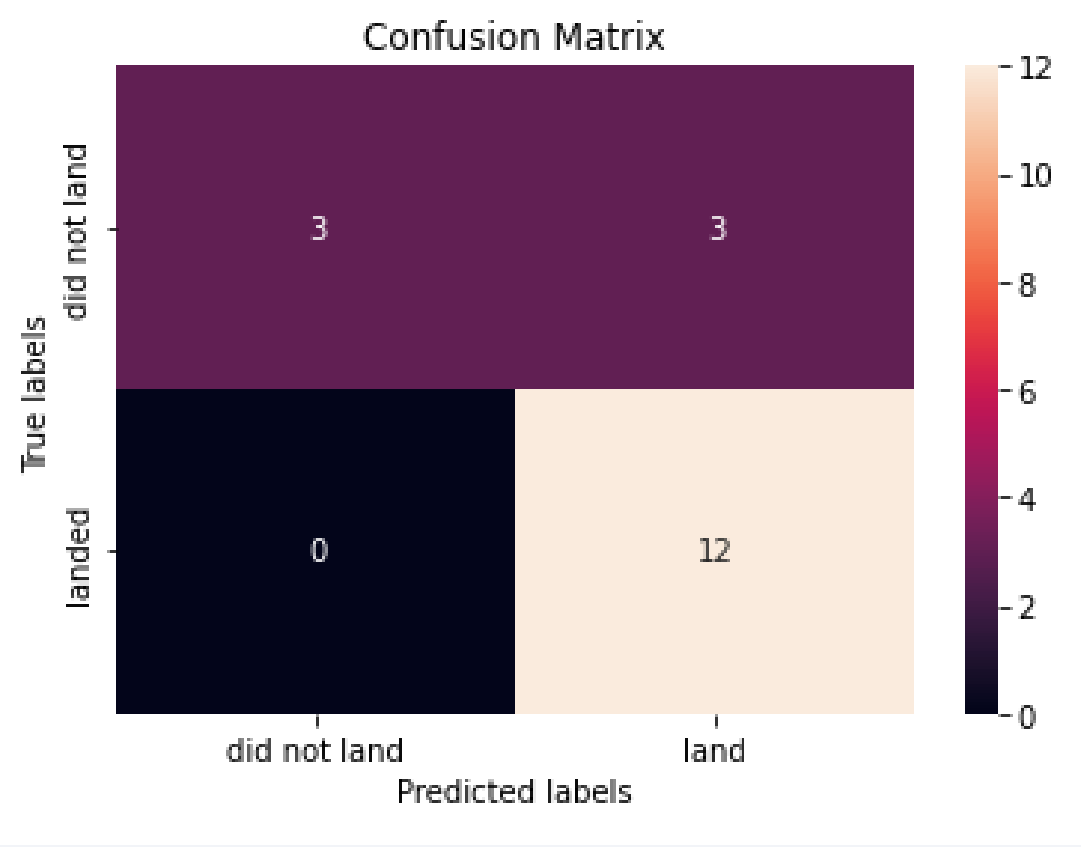


	LogReg	SVM	Tree	KNN
Jaccard_Score	0.833333	0.845070	0.819444	0.819444
F1_Score	0.909091	0.916031	0.900763	0.900763
Accuracy	0.866667	0.877778	0.855556	0.855556

- Overall, the models showed high accuracy
- The SVM model showed the best accuracy performance
- The decision tree and K-nearest neighbors were the least accurate

Confusion Matrix

- The support vector machine confusion matrix which correctly identifies landed rockets
- The rockets that ‘did not land’ were predicted with 50% accuracy (false positives)



Conclusions

- As SpaceX launches more rockets, the probability of successful launches increases
- As payload mass increases successful launches increases
- ES-L1, SSO, HEO, and GEO orbits show the highest success rate compared to the other orbits
- 2019 displayed the most number of successful launches
- KSC LC 39A shows the most number of successful launches
- Launch sites need to be located near the coast and railroads and away from cities for accessibility to resources and safety purposes
- Booster version FT were the most successful
- The most successful payload mass range is between 2000-5000kg
- The best performing classification algorithm was the support vector machine (87.7% accurate)

For stakeholders:

- The successful parameters for a future rocket would be to incorporate a booster version FT with payload between 2000-5000kg to be launched in ES-L1, SSO, HEO or GEO orbit at KSC LC 39A. Before the next flight of a future rocket, it should be tested by a SVM algorithm.

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!