IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Frankie Sanjana
Feb 2024

Outline

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

Executive Summary

- Methodologies used in the project include:
  - Data collection from the SpaceX API
  - Web scraping using the Python BeautifulSoup library
  - Conversion of the JSON file to a Pandas dataframe
  - Exploratory data analysis using graphs and SQL queries
  - Creation of interactive visual analytics using Folium and Plotly Dash
  - Predictive analysis using classification models
- Summary of all results
  - Launches were much more likely to be successful in later years
  - Landing success rate differs by orbit type
  - Further analysis could investigate the statistical significance of our results. Although the classifier models appear to predict launch outcome well, sample size is (necessarily) small

Introduction

- SpaceX advertises Falcon 9 rocket launches on its website, with a cost of 62 million USD
  - Other providers charge over 165 million USD for the same service
- A large part of the cost differential is because SpaceX is able to reuse the “first stage” [a part of the rocket]
- Therefore if we can determine if the first stage will land, we can determine the cost of a launch
  - If we can identify what features affect whether or not the first stage will land, we can use this information to maximise the chance of future launches landing the first stage successfully
  - This information can be used for a company that wants to launch its own rockets with a reusable first stage

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected from the SpaceX API and returned in a JSON file; web scraping was also performed from Wikipedia data using the Python BeautifulSoup library
- Perform data wrangling
  - The JSON file was converted into a Pandas dataframe
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Suitable classification models were selected and evaluated: logistic regression, decision tree classifier, K-nearest neighbor and support vector machine
  - For each model, multiple hyperparameters were tested to find the best performance

Data Collection

- Data sets were collected by making a GET request to the SpaceX REST API
- We used the SpaceX REST API endpoints, or URL, to target a specific endpoint of the API to get past launch data.
- We performed a GET request using the requests library to obtain the launch data
- The result of this can be viewed by calling the .json() method
- The output of the request was a list of JSON objects that was then readily convertible to a Pandas dataframe using the .json_normalize() method, i.e., we ‘normalized’ the structured JSON data into a flat table
- The different columns of the API were used as follows:
    - From the rocket column we extracted the booster name
    - From the launchpad column we took the name of the launch site being used and its coordinates
    - The mass of the payload and the orbit it was travelling to came from the payload column
    - The type and outcome of the landing, and other technical details, were taken from the cores column

Data Collection – SpaceX API

1. We make a GET request to the SpaceX API

Now let's start requesting rocket launch data from SpaceX API with the following URL:

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

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

2. The API returns a series of JSON objects…

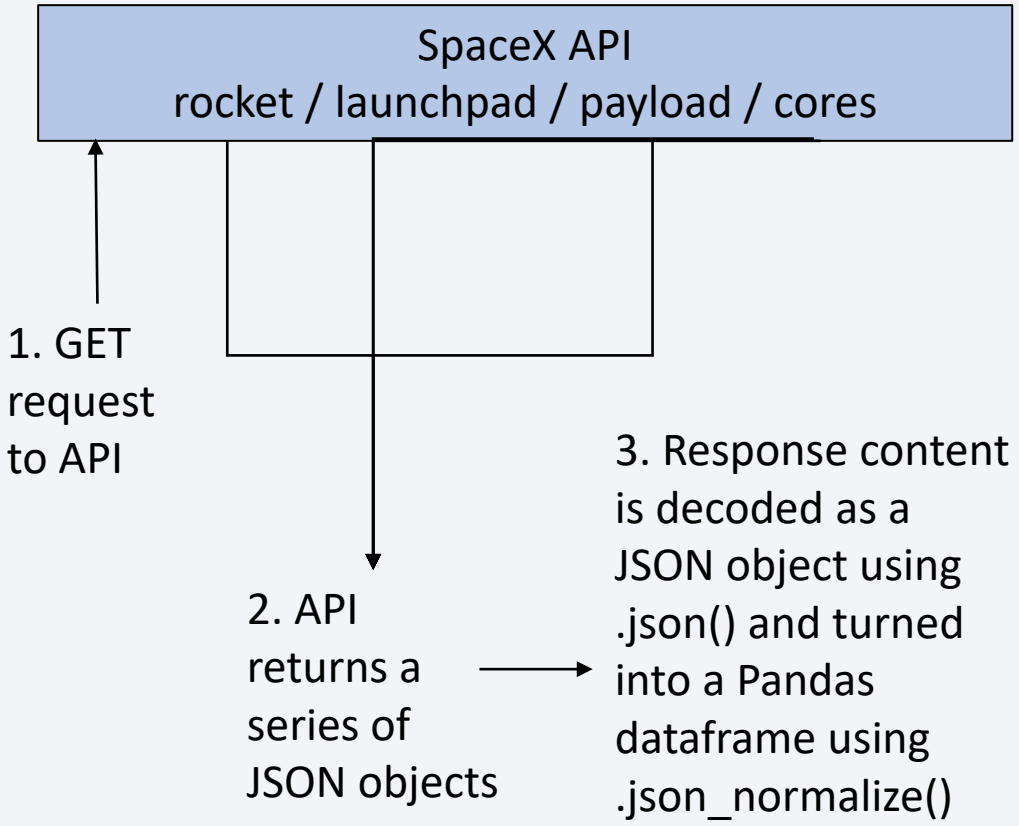

```
4,"gridfins":true,"legs":true,"reused":true,"landing_attempt":true,"landing_success":true,"landing_type":"ASDS","landpad":"5e9
e3033383ecbb9e534e7cc"}],"auto_update":true,"tbd":false,"launch_library_id":"1c903b65-6667-4fd5-944d-296c5f13e01f","id":"63161
339ffc78f3b8567070c"},{"fairings":null,"links":{"patch":{"small":"https://images2.imgbox.com/eb/d8/D1Yywp0w_o.png","large":"ht
tps://images2.imgbox.com/33/2e/k6VE4iYl_o.png"},"reddit":{"campaign":null,"launch":"https://www.reddit.com/r/spacex/comments/x
vm76j/rspacex_crew5_launchcoast_docking_discussion_and/","media":null,"recovery":null},"flickr":{"small":[],"original":[]},"pr
```


…but we can't work with the data in this format! So…

3. We use Python code to turn this data into a flat table known as a Pandas dataframe, so that it is both more easily readable by humans and can be manipulated to derive insights

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

```
15]: # Use json_normalize meethod to convert the json result into a dataframe
     json_data = response.json()

     # Use json_normalize on the list
     data = pd.json_normalize(json_data)
```



GitHub URL: https://github.com/frankiesanjana/ibm-data-science-capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

Data Collection - Scraping

1. We make a GET request to the Wikipedia URL

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

2. Convert the HTML response to a BeautifulSoup object

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

3. Extract the column names

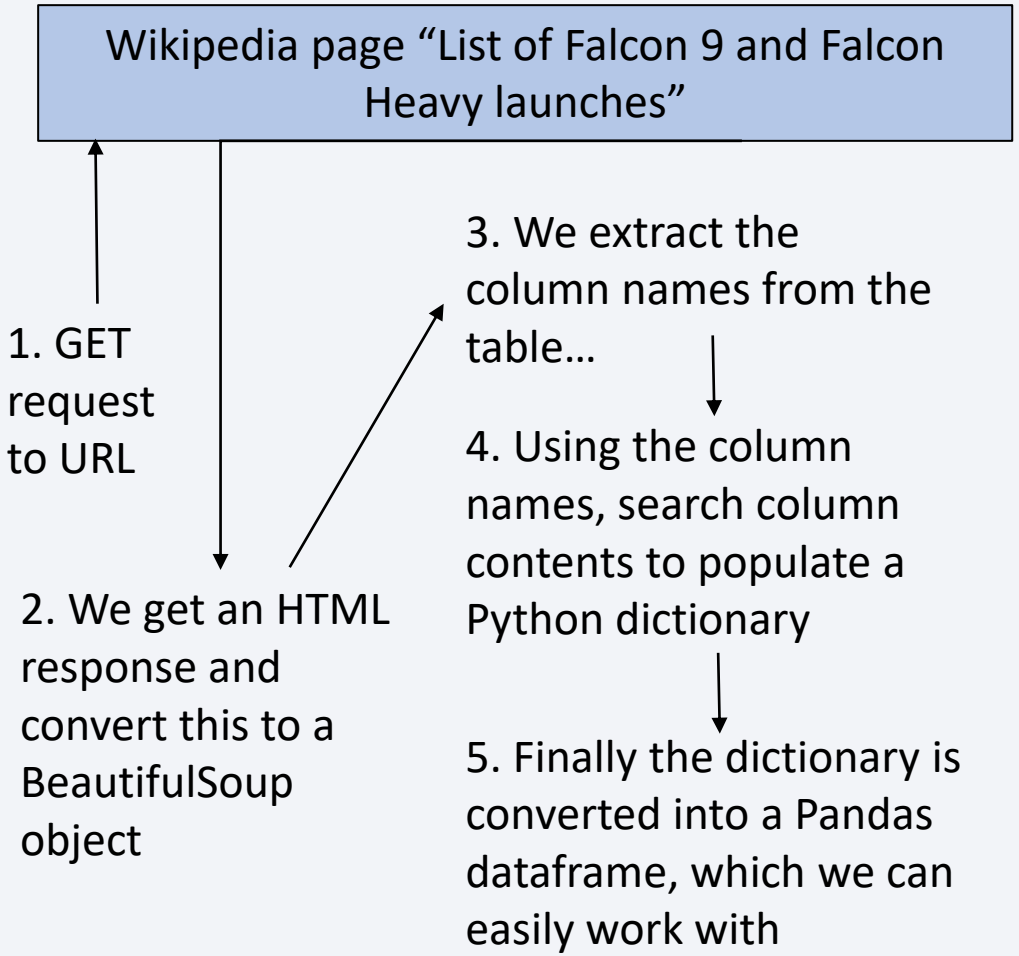
```
column_names = []

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

4. Search and populate the launch_dict dictionary

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

5. Convert to a Pandas dataframe

```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
```



GitHub URL: https://github.com/frankiesanjana/ibm-data-science-capstone/blob/main/jupyter-labs-webscraping.ipynb

Data Wrangling (1 of 2)

- The data obtained and loaded into the Pandas dataframe included various landing outcomes
  - We wanted to adjust these so that the landing outcome was simply shown as 1 (i.e. the landing completed successfully) or 0 (i.e. the landing failed)
  - The landing outcomes in the raw data are shown here in the first picture
  - This was addressed by first creating a set of “bad” outcomes that should be coded as 0; see the second picture (ctd ->)

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

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

```
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
```

We create a set of outcomes where the second stage did not land successfully:

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


GitHub URL: https://github.com/frankiesanjana/ibm-data-science-capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

Data Wrangling (2 of 2)

- It was then straightforward to assign outcomes to the correct class using a ‘for’ loop:

Using the `Outcome`, create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome`; otherwise, it's one. Then assign it to the variable `landing_class`:

]:
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for outcome in df['Outcome']:
    if outcome in bad_outcomes:
        outcome = 0
    else:
        outcome = 1
    landing_class.append(outcome)

len(landing_class)
```

]: 90

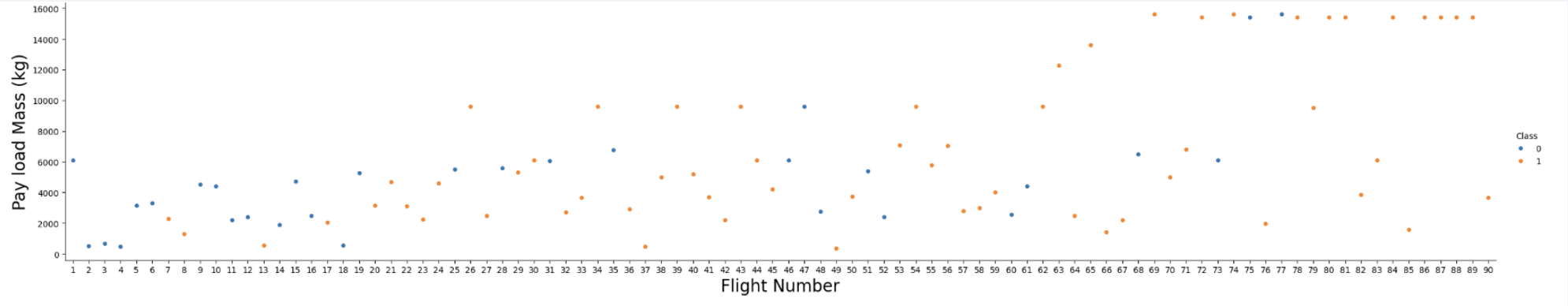
This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

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

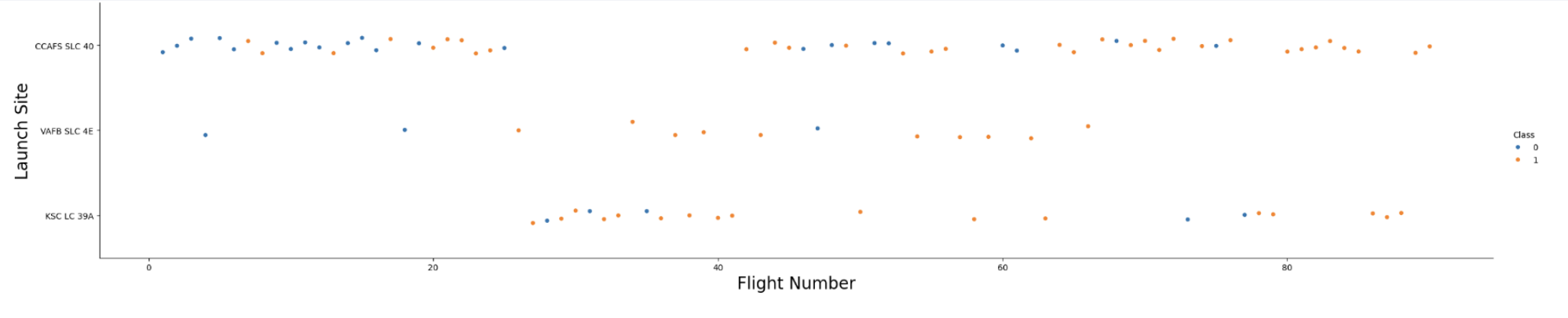

GitHub URL: https://github.com/frankiesanjana/ibm-data-science-capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

- We plotted the Flight Number vs. Payload Mass and also showed the launch outcome



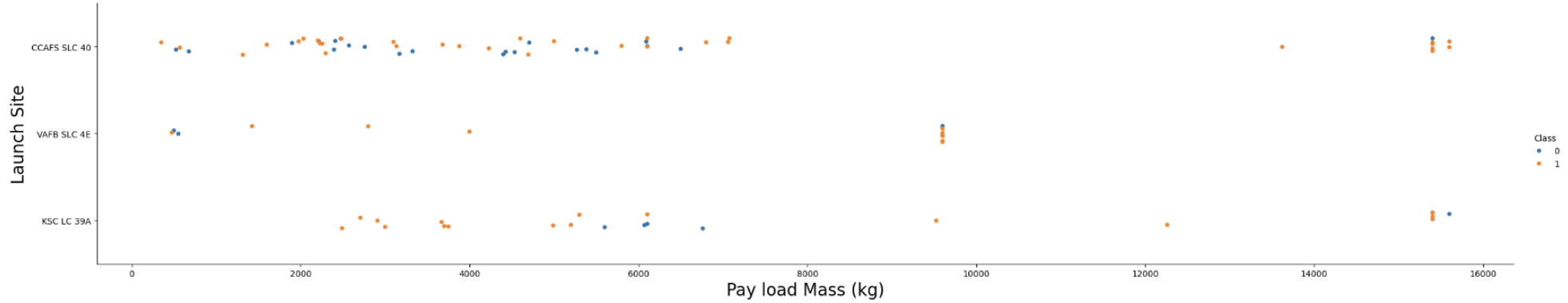
- We then plotted the Flight Number vs. Launch Site and also showed the launch outcome



GitHub URL: https://github.com/frankiesanjana/ibm-data-science-capstone/blob/main/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA with Data Visualization

- We checked for an association between launch sites and their payload mass

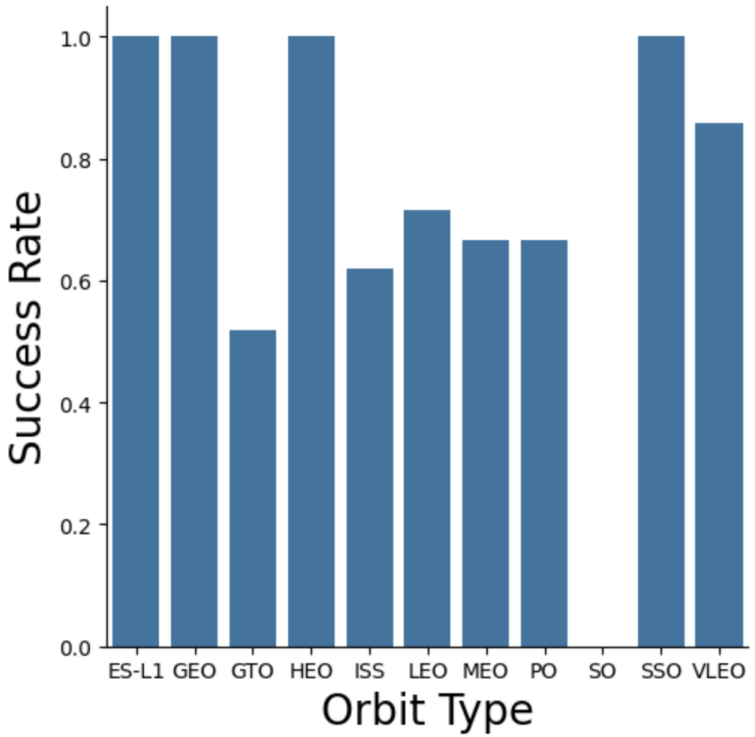


- From these scatter graphs, we observe that
  - Higher numbered flights tend to be more likely to land successfully
  - Different launch sites have slightly different payload mass profiles
  - There does not appear to be a clear association between launch site and likelihood of a successful landing
  - There does not appear to be a clear association between payload mass and likelihood of a successful landing

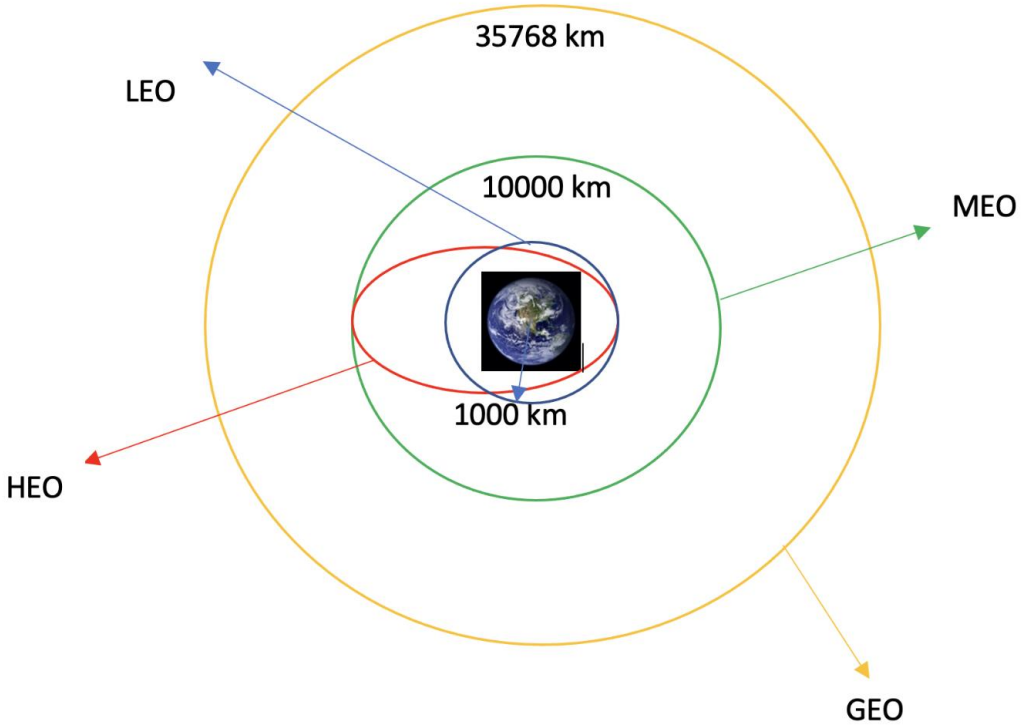
GitHub URL: https://github.com/frankiesanjana/ibm-data-science-capstone/blob/main/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

A bar chart showed differences in landing success rate by orbit type

Orbit type and success rate



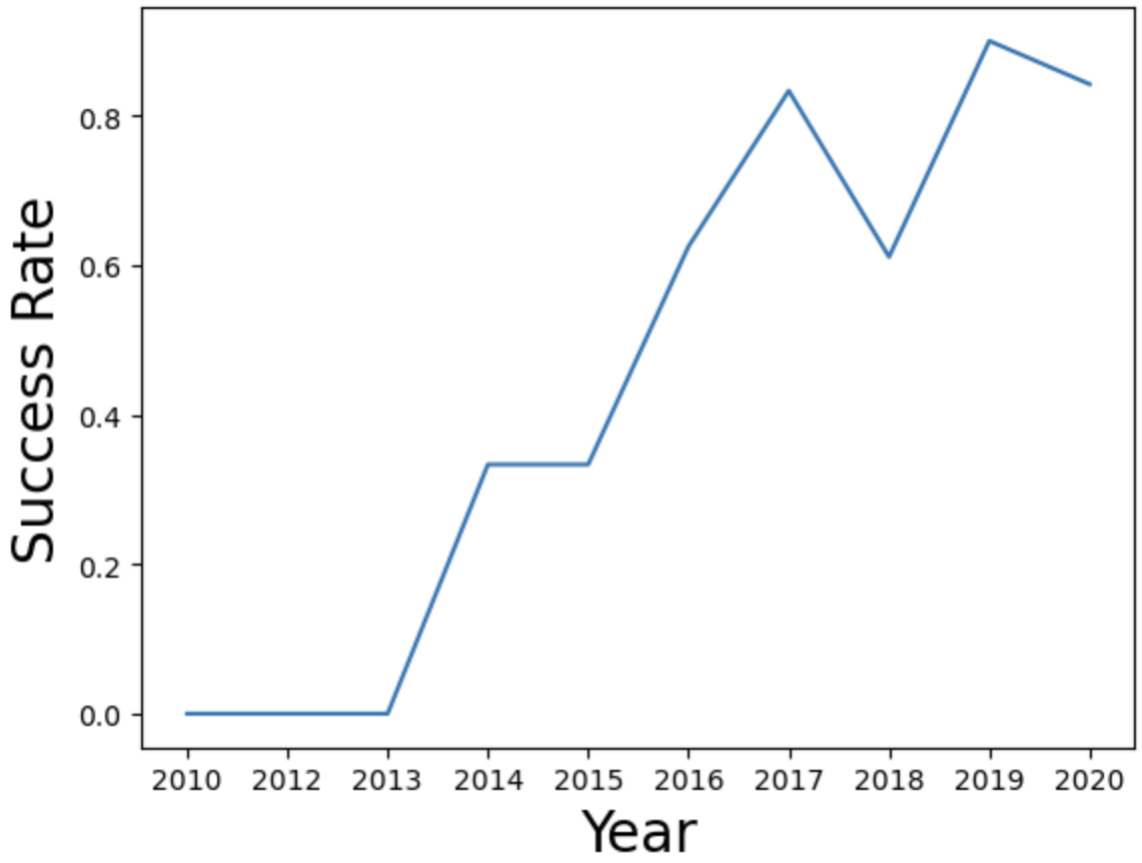
Visualisation illustrating principal types of orbit



GitHub URL: https://github.com/frankiesanjana/ibm-data-science-capstone/blob/main/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

We see that landing success rate increased sharply over time

Landing success rate, 2010-20



- The strongest correlations with landing success rate that were observed in the EDA relate to year and flight number
- This is likely to suggest
  - (i) learnings from initial flights being applied to later flights; and / or
  - (ii) technological, operational or other improvements being made over time that increase the chances of a successful landing

GitHub URL: https://github.com/frankiesanjana/ibm-data-science-capstone/blob/main/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA with SQL

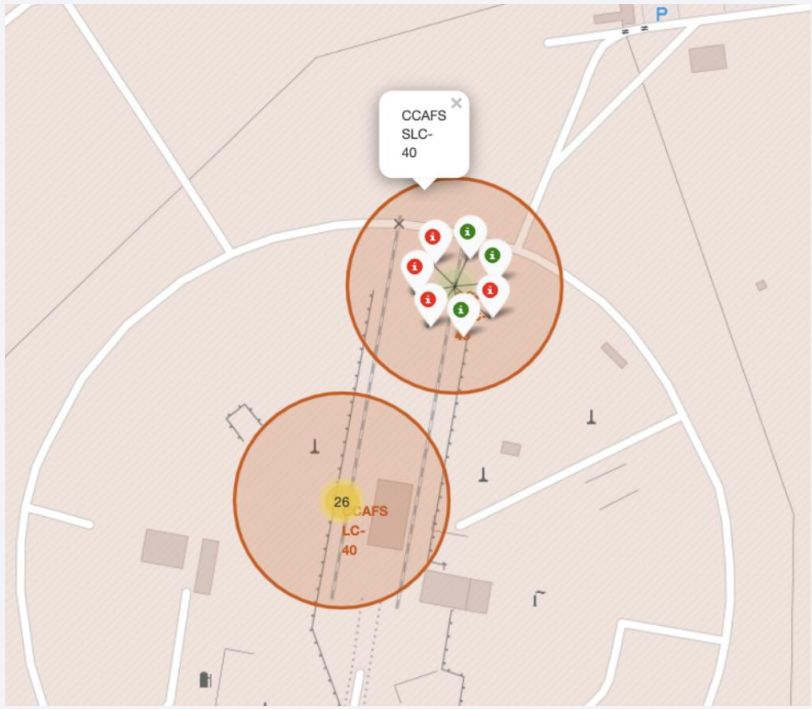
- SQL queries were executed to show the following:
  - Names of the unique launch sites in the space mission
  - 5 records where launch sites begin with the string 'CCA'
  - Total payload mass carried by boosters launched by NASA (CRS)
  - Average payload mass carried by booster version F9 v1.1
  - Date of the first succesful landing outcome in ground pad
  - Names of boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
  - Total number of success and failure mission outcomes
  - Names of the booster_versions that have carried the maximum payload mass
  - 2015 month names, failure landing_outcomes in drone ship, booster versions, launch_site
  - Count of landing outcomes by type between 4 June 2010 and 20 March 2017

GitHub URL: https://github.com/frankiesanjana/ibm-data-science-capstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- The following features were visualised on an interactive map using Folium:
  - The location of each launch site
  - Successful and failed launches for each site
  - Distances from a launch site to its proximities
- This allows users to see at a glance where the launch sites are located and a clear visual indication of the successful and failed launches from each site
- Users can also explore the proximities of each launch site

Example of a map showing successful and failed launches at a given site



GitHub URL: https://github.com/frankiesanjana/ibm-data-science-capstone/blob/main/lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

- The first feature added to the dashboard was a dropdown list, to allow launch site selection
- A pie chart clearly illustrates the split between successful and failed launches for all sites or a specific site, if selected
- A slider allows the user to select the payload range to be shown, so that the user can customise the range of payload mass according to what they find most relevant
- Finally, a scatter graph shows the correlation between payload mass and launch success

GitHub URL: https://github.com/frankiesanjana/ibm-data-science-capstone/blob/main/spacex_dash_app.py

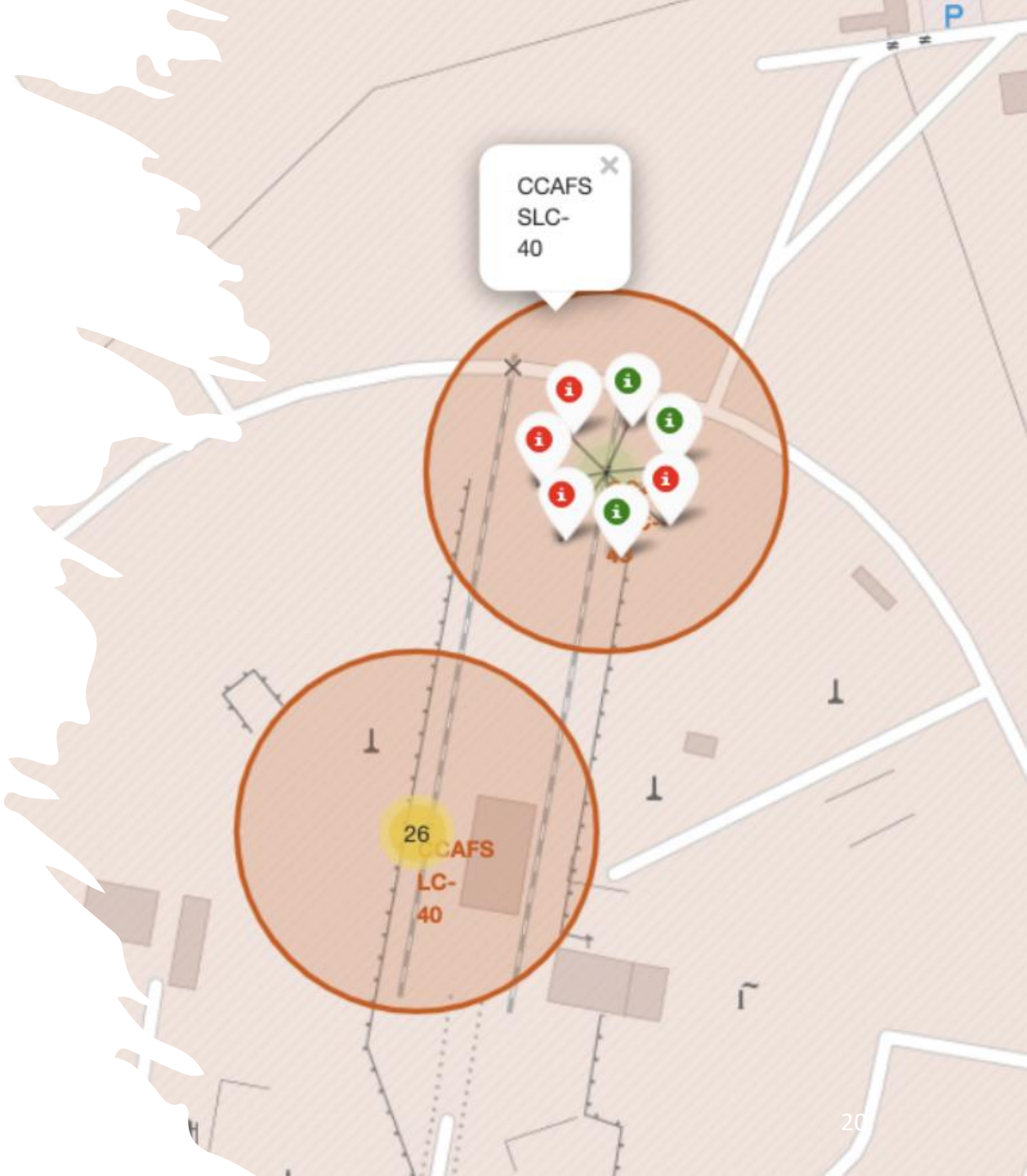
Predictive Analysis (Classification)

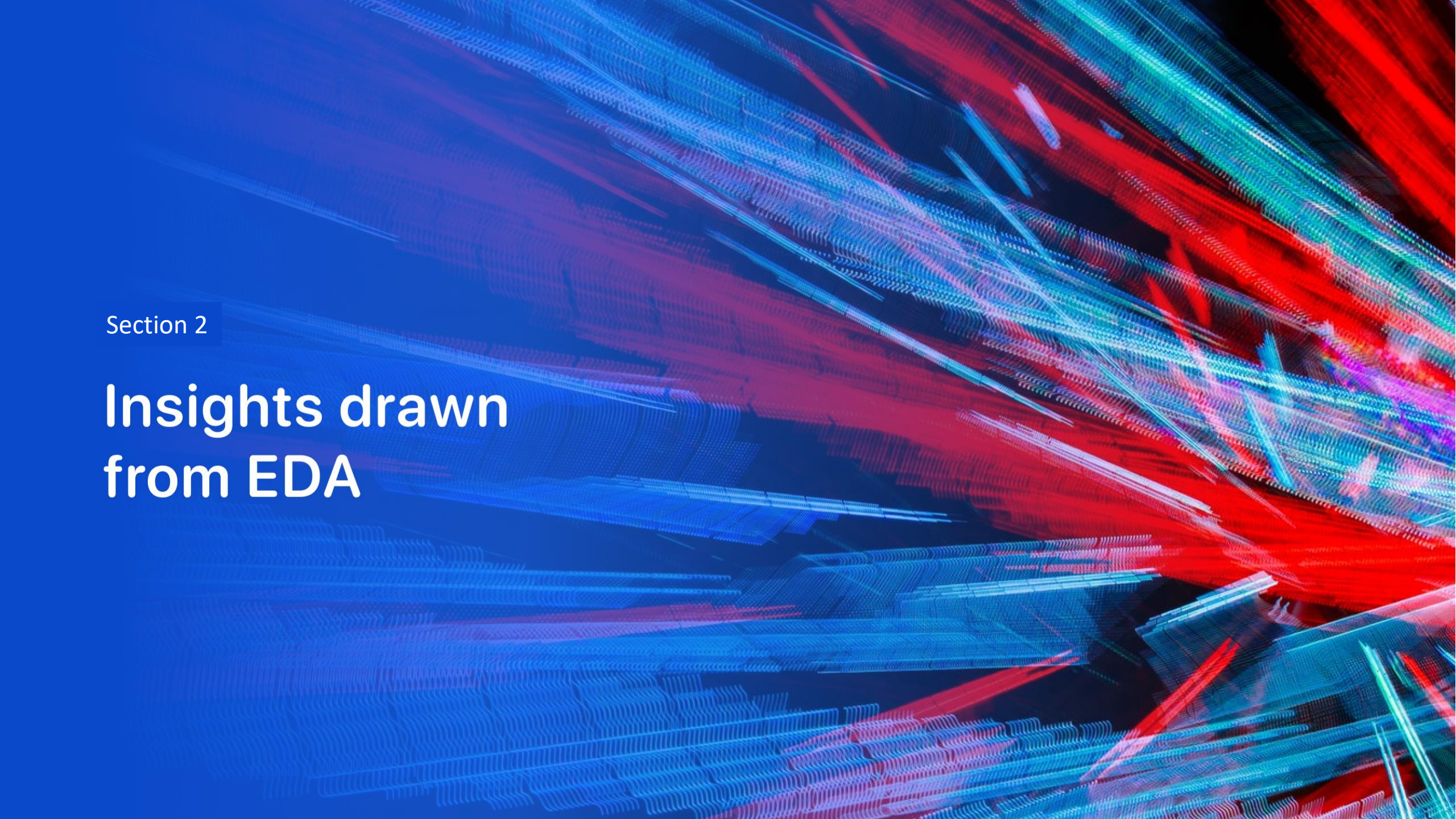
- Various ML classification models were tested to find which one was best at predicting success or failure of a launch on new data that the model had not been trained on
- To do this, the data was split into a training set to train the model, and a test set that was kept back until after model training in order to evaluate its performance
- Each model was then trained using the training set, including setting different hyperparameter values for the model to allow the most optimal hyperparameter settings for the given data to be selected
- Once the best hyperparameters had been evaluated, the model used these hyperparameters to predict the outcome of the launches in the test set
- The model's prediction was then compared with the actual values, and accuracy scores and a confusion matrix were computed to find the best-performing model

GitHub URL: https://github.com/frankiesanjana/ibm-data-science-capstone/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Results

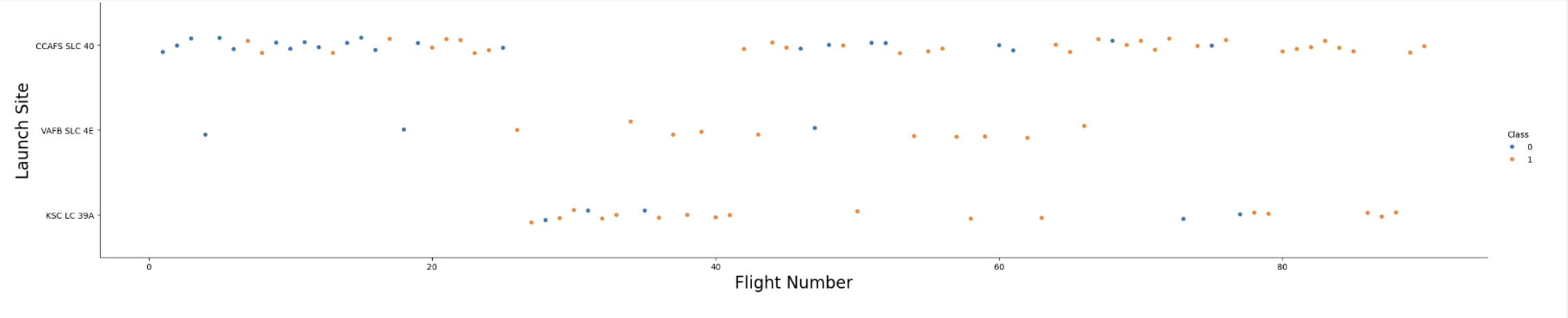
- The strongest correlations with landing success rate that were observed in the EDA relate to year and flight number
- Interactive analytics were successfully created, allowing users to gain their desired insights from the data
- The predictive analysis showed strong performances from all models; however, the best prediction came from the Decision Tree classifier, with an accuracy of 94% on the test data




Section 2
Insights drawn from EDA

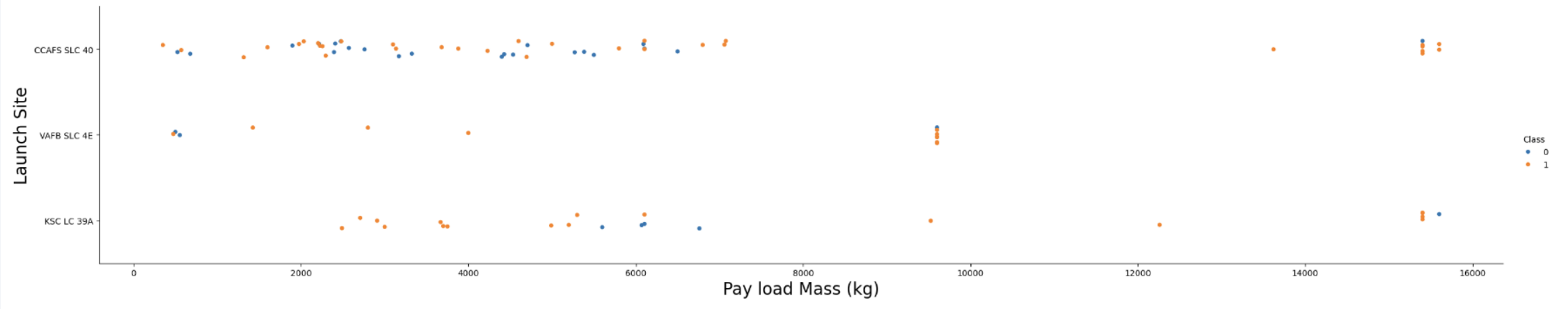
Flight Number vs. Launch Site

- We plotted the Flight Number vs. Launch Site and also showed the launch outcome
- This shows that flights tend to launch in batches from each launch site, especially for lower flight numbers
- There appears to be little correlation between launch site and likelihood of a successful landing



Payload vs. Launch Site

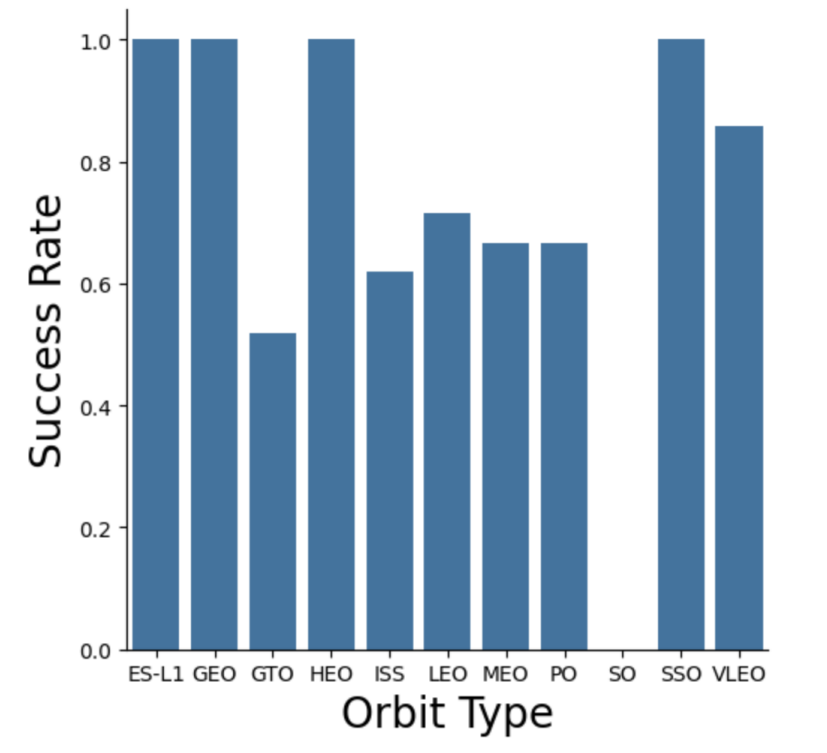
- We checked for an association between launch sites and their payload mass



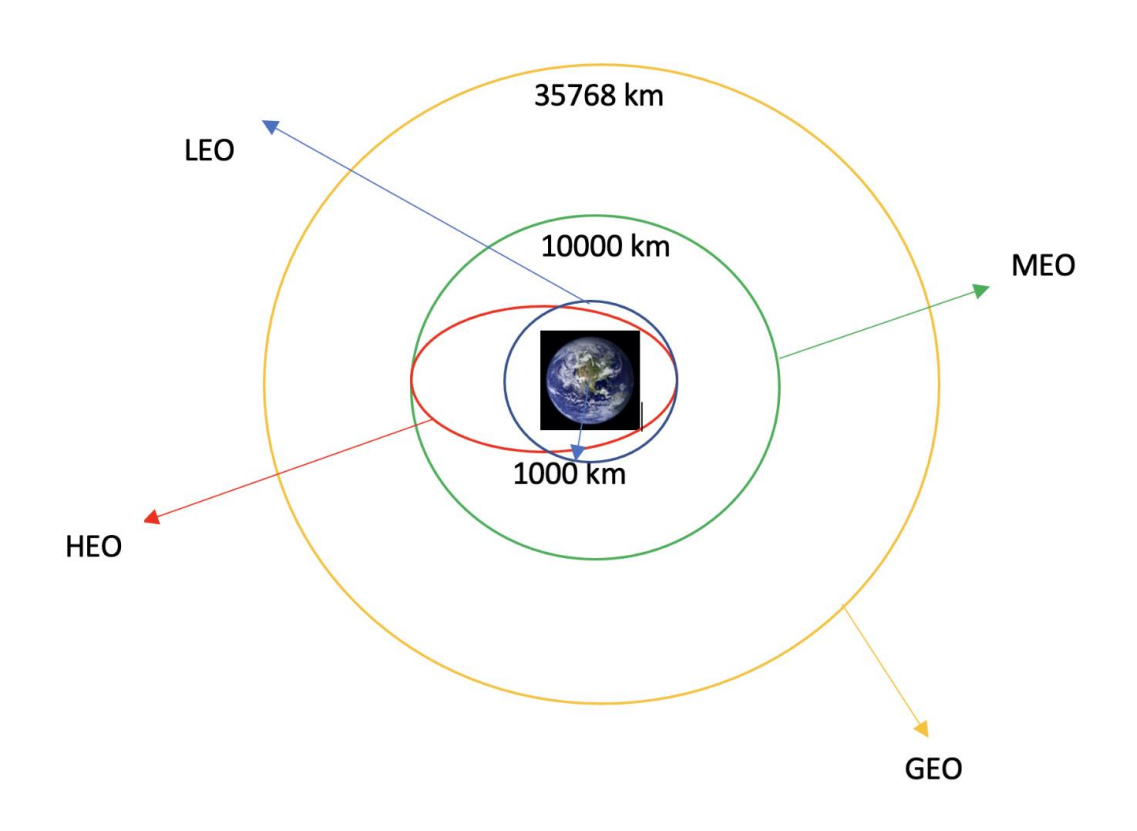
- Different launch sites have slightly different payload mass profiles
- There does not appear to be a clear association between payload mass and likelihood of a successful landing

Success Rate vs. Orbit Type

Orbit type and success rate

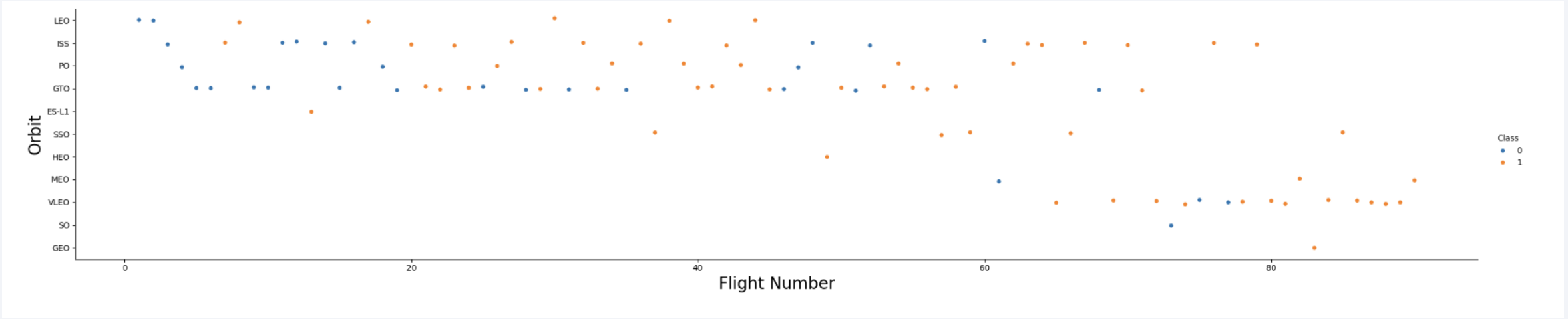


Visualisation illustrating principal types of orbit



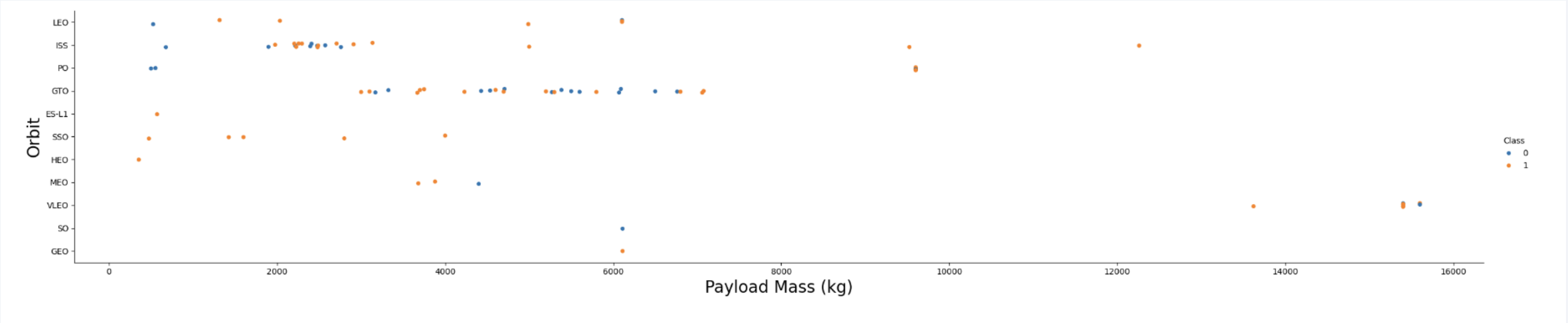
Flight Number vs. Orbit Type

- We plotted the Flight Number vs. Orbit Type and also showed the launch outcome
- No obvious relationship is apparent between flight number and orbit type, with the most noticeable trend again being that of higher-numbered flights being the most likely to have a successful outcome



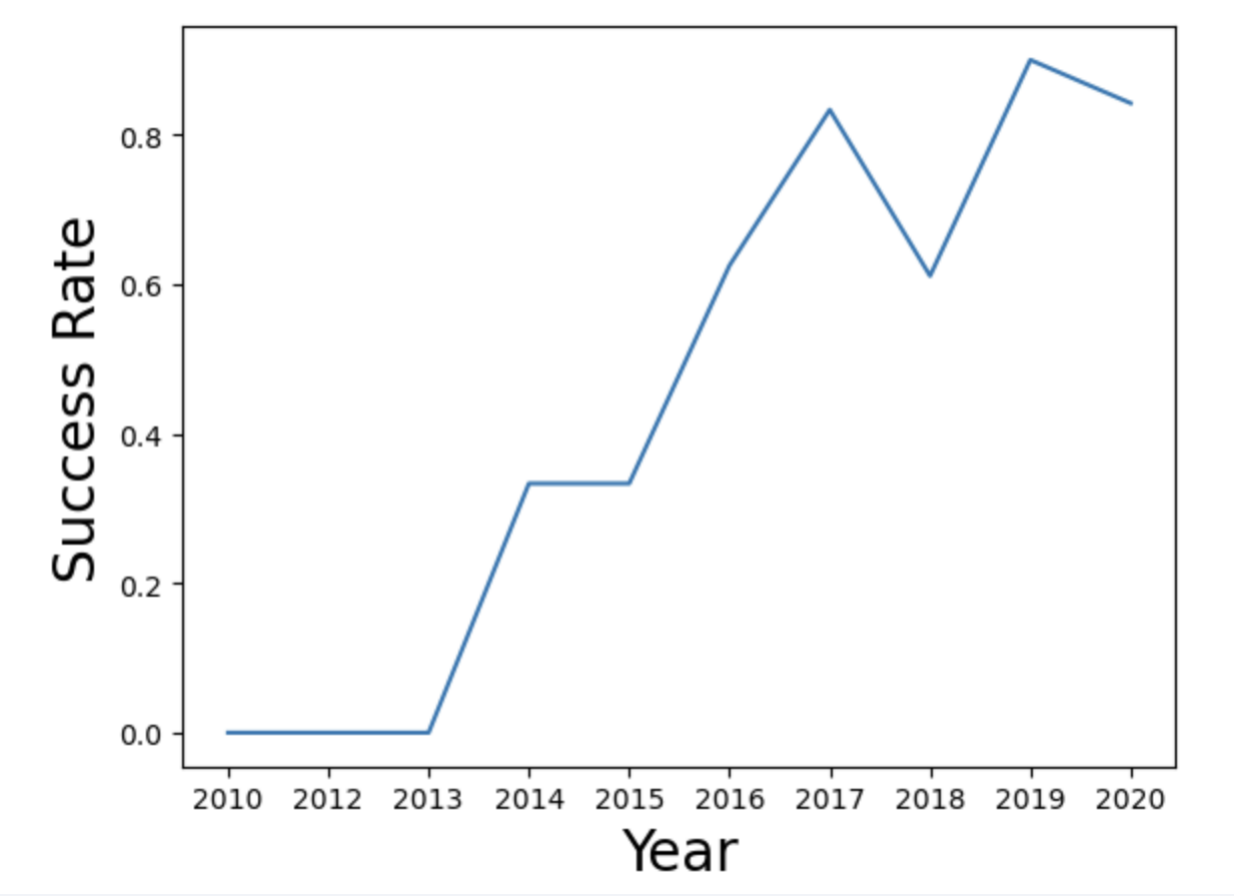
Payload vs. Orbit Type

- We plotted the Payload Mass vs. Orbit Type and also showed the launch outcome
- We see that the ISS orbit tends to have lower payload mass values; GTO has moderate values and VLEO orbit has the highest values for payload mass, with other orbit types having low or moderate payload mass
- Again, we do not see a clear correlation between these metrics and success of launch outcome



Launch Success Yearly Trend

Landing success rate, 2010-20



- The strongest correlations with landing success rate that were observed in the EDA relate to year and flight number
- This is likely to suggest
  - (i) learnings from initial flights being applied to later flights; and / or
  - (ii) technological, operational or other improvements being made over time that increase the chances of a successful landing

All Launch Site Names

- In SQL, the SELECT DISTINCT command returns unique values in the specified column
- The below command can therefore be used to return all unique launch site names

Display the names of the unique launch sites in the space mission

[8]:
```
%sql select distinct "Launch_Site" from SPACEXTABLE
```

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

[8]:

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

Launch Site Names Beginning with 'CCA'

- The WHERE clause allows us to specify additional conditions
- LIKE allows us to search for a given text string, and the % sign indicates that additional text may be present after the searched text
- LIMIT 5 truncates the search after the first 5 results

Display 5 records where launch sites begin with the string 'CCA'

```
%sql select * from SPACEXTABLE where "Launch_Site" like "CCA%" limit 5
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX
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
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)

Total Payload Mass

- The SUM operation allows us to sum all values where the given criteria are met

Task 3

Display the total payload mass carried by boosters launched by NASA (CRS)

```
%sql select sum("PAYLOAD_MASS__KG_") from SPACEXTABLE where Customer = "NASA (CRS)"
```

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

sum("PAYLOAD_MASS__KG_")
45596

Average Payload Mass by F9 v1.1

- The AVG method allows us to average over the values meeting the given criterion
- This could be useful in assessing whether the average is a tipping point beyond which there is an increased or decreased likelihood of launch success

Task 4

Display average payload mass carried by booster version F9 v1.1

```
[12]: %sql select avg("PAYLOAD_MASS__KG_") from SPACEXTABLE where "Booster_Version" = "F9 v1.1"
 * sqlite:///my_data1.db
Done.
[12]: avg("PAYLOAD_MASS__KG_")
      2928.4
```

First Successful Ground Landing Date

- The MIN function returns the lowest value in the specified range
- This query allows us to see the first successful landing outcome of this type

List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

```
%sql select min(date) from SPACEXTABLE where "Landing_Outcome" = "Success (ground pad)"
```

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

min(date)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- To find values in a range, use “BETWEEN x AND y”
- Note also that with WHERE we can specify multiple conditions simply by using AND before adding the next condition

Task 6

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

```
%sql select "Booster_Version" from SPACEXTABLE where "Landing_Outcome" = "Success (drone ship)" and "PAYLOAD_MASS__KG_" betw
```

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

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

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes
- GROUP BY will group the results by category. Once categories have been defined they can be referred to by number, i.e. Mission Outcome in the below query is 1 and count(*) would be 2

List the total number of successful and failure mission outcomes

```
%sql select Mission_Outcome, count (*) from SPACEXTABLE group by 1
```

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

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

Boosters Carried Maximum Payload

- The subquery in SQL is a query that is nested inside another query. This allows data to be retrieved from multiple tables, or, as in this case, the use of the result of one query as a condition in another
- Here we obtain a list of all the booster versions that have carried the maximum payload mass

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

```
%sql select Booster_Version from SPACEXTABLE where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTABLE)
```

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

Booster_Version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3

2015 Launch Records

- We are using SQLLite, which does not support month names
- We therefore use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

```
%sql select substr(Date, 6,2) as month, Date, Booster_Version, Launch_Site from SPACEXTABLE \
where substr(Date,0,5)='2015' and Landing_Outcome = 'Failure (drone ship)'
```

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

month	Date	Booster_Version	Launch_Site
01	2015-01-10	F9 v1.1 B1012	CCAFS LC-40
04	2015-04-14	F9 v1.1 B1015	CCAFS LC-40

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

- We again see the use of the BETWEEN … AND syntax, and here we also see ORDER BY
- By default, or if ASC is specified, the results of ORDER BY are sorted in ascending order. However, in this case we want the highest number to appear first, so we use DESC to achieve this

```
%sql select Landing_Outcome, count(*) as count_outcome from SPACEXTABLE \
where Date between '2010-06-04' and '2017-03-20' \
group by Landing_Outcome \
order by count_outcome desc;
```

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

Landing_Outcome	count_outcome
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

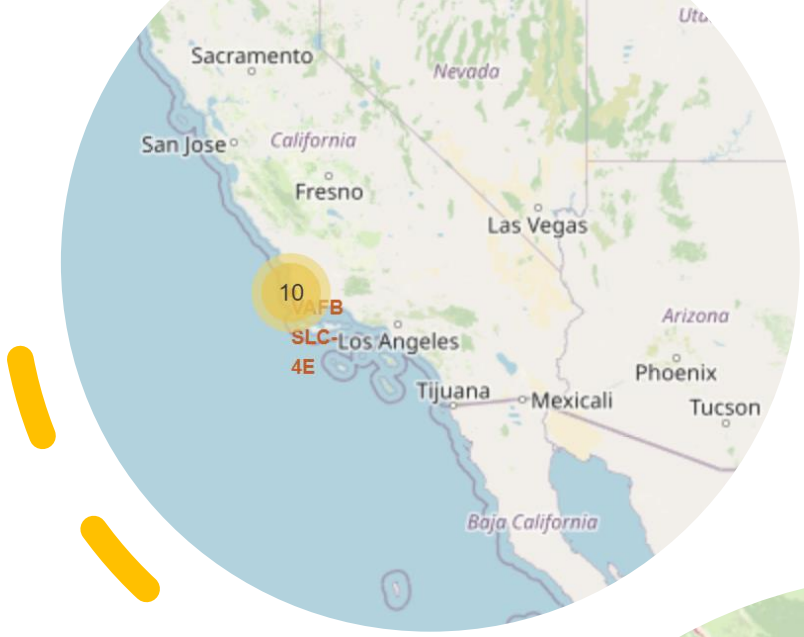
Launch Sites on a Folium Map

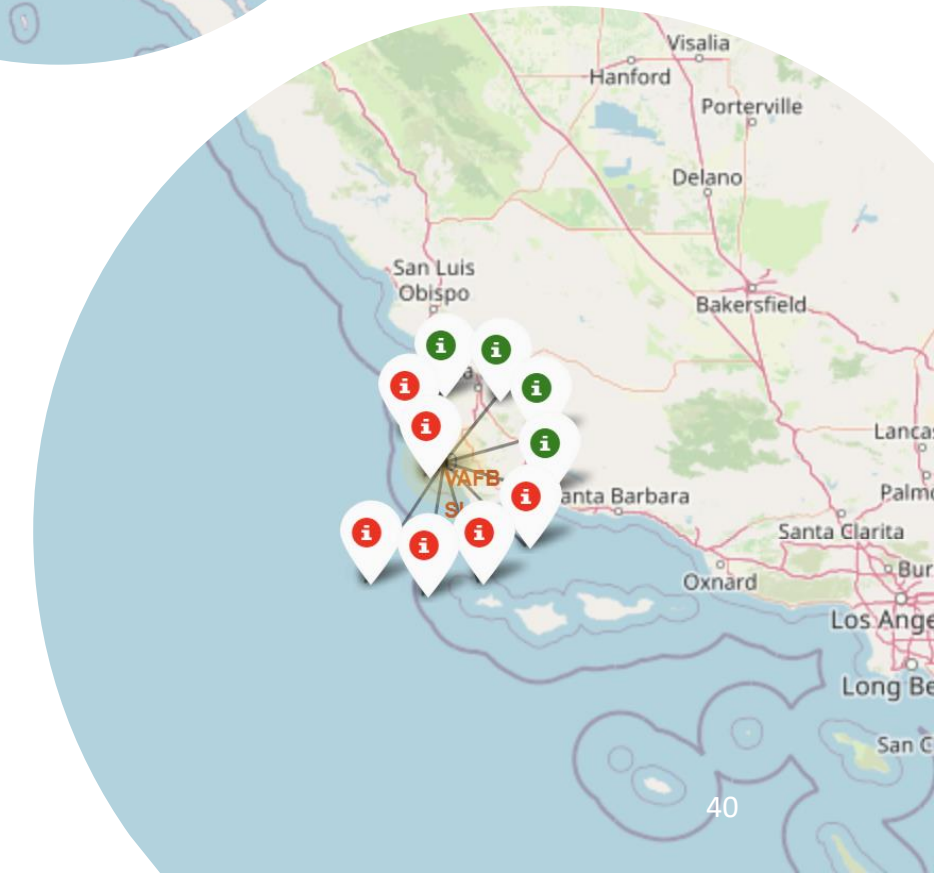
- The map allows users to gain an at-a-glance overview of where launch sites are located

Launch Site Details

- A launch only happens in one of the four launch sites, which means many launch records have the exact same coordinates. Marker clusters are used initially to simplify the map since many markers have the same location

- When the user clicks on the launch site, the details showing successful and failed launches are exploded, as seen in the second graphic

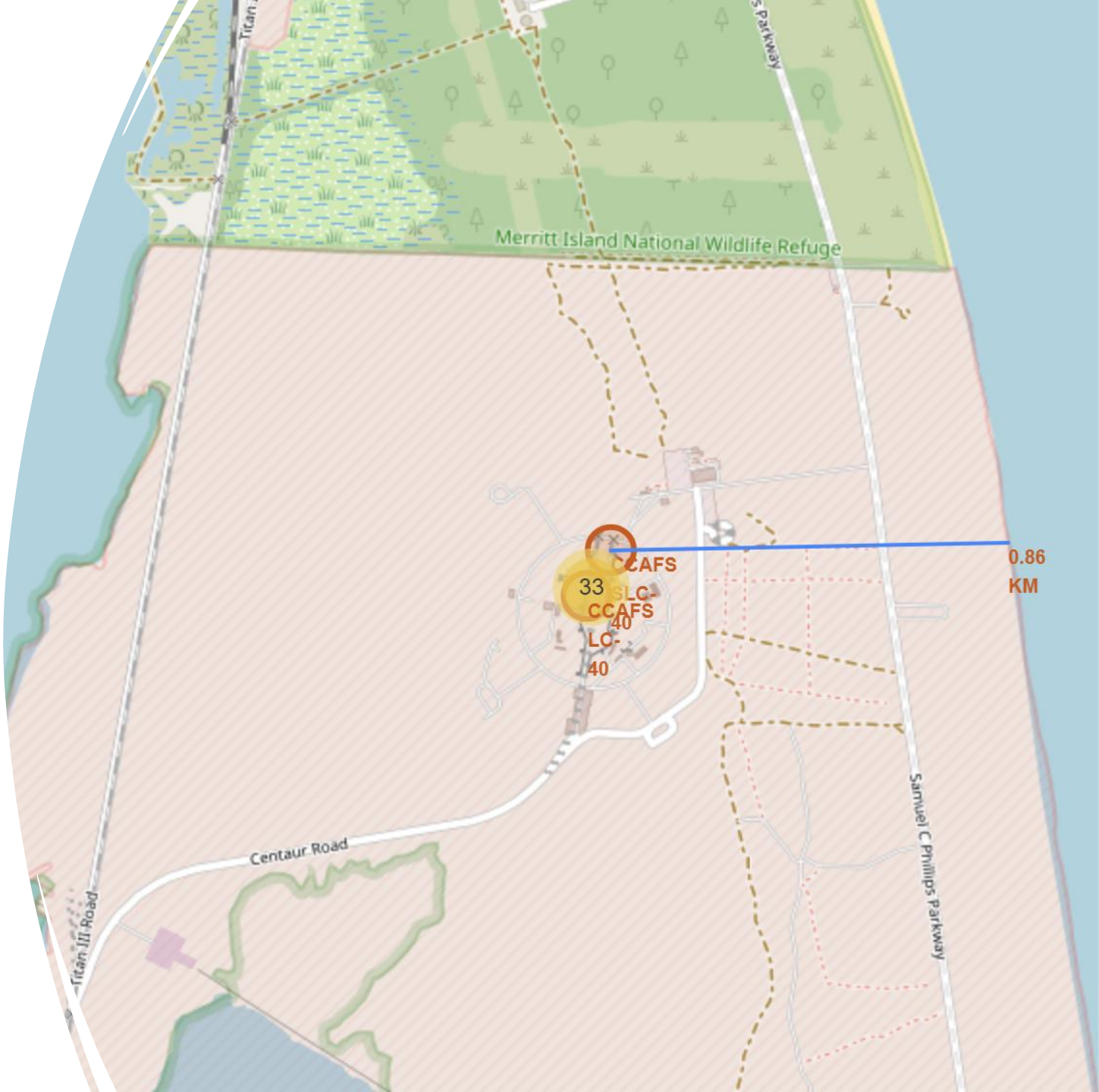


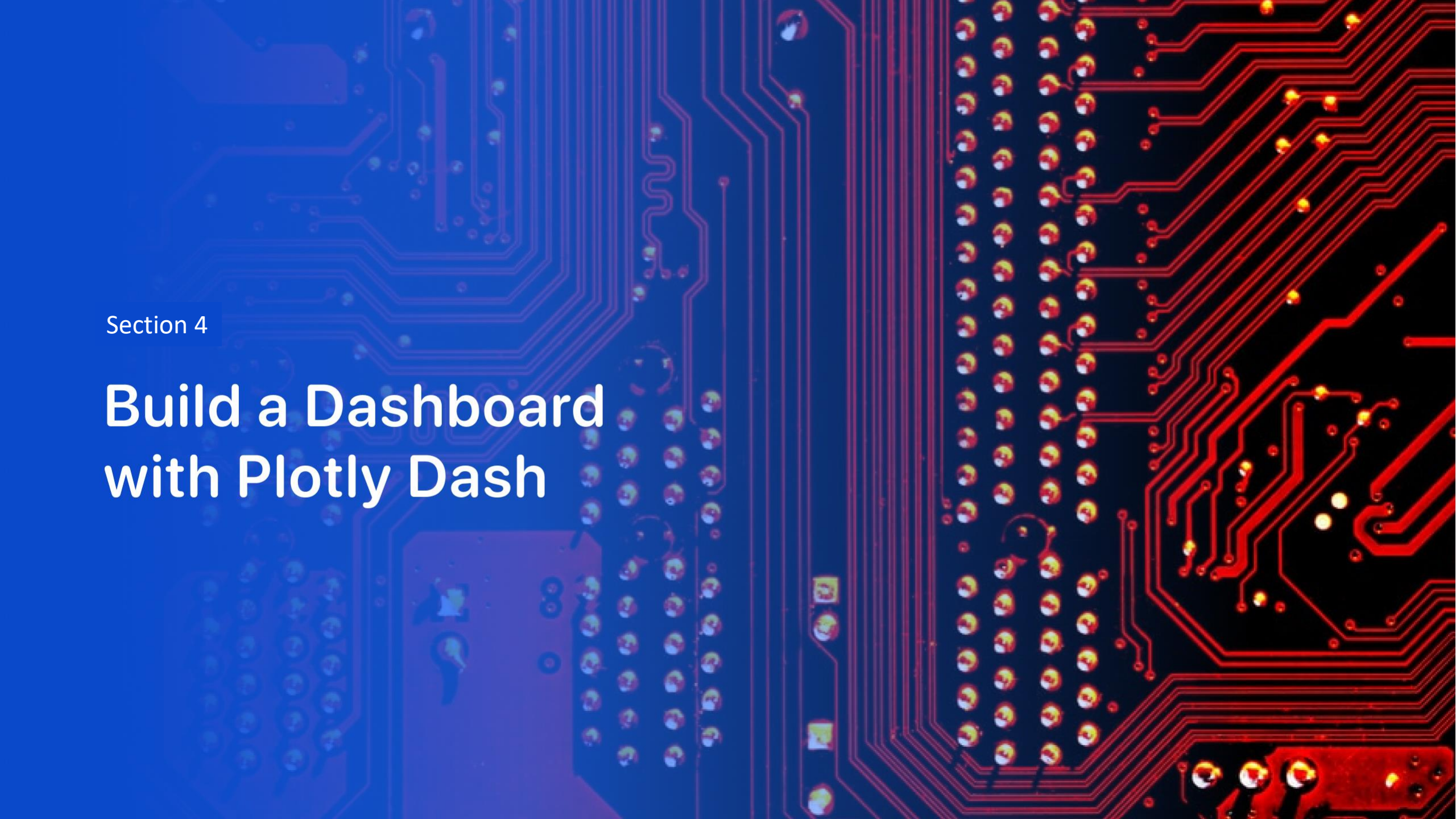


Distance to Nearby Features of Interest

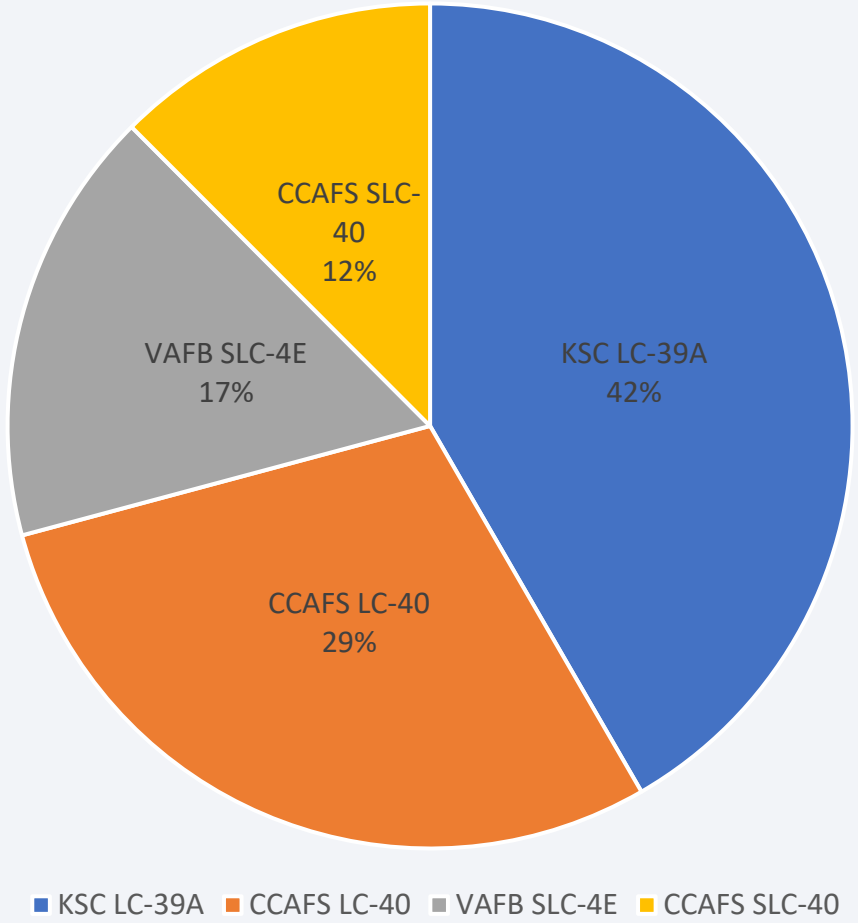
- We can also display the distance between of selected launch site to its proximities such as railway, highway, coastline, with distance calculated and displayed on the map

- This allows the user to see details of nearby landmarks that may be relevant to operational use of the launch site



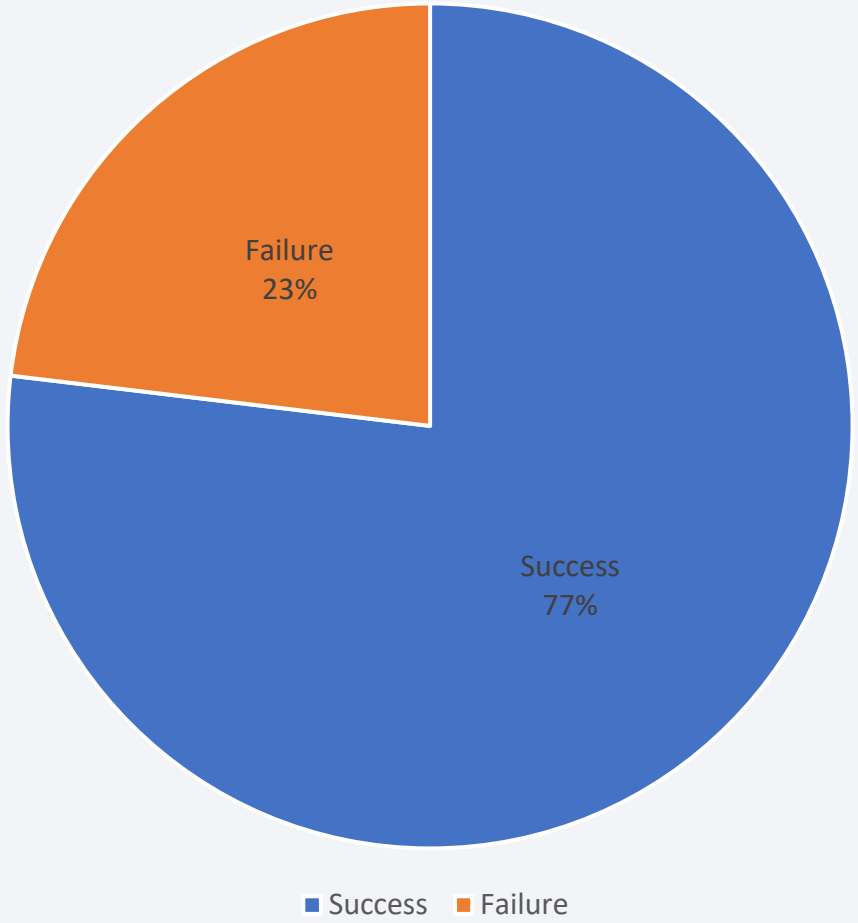

Section 4
Build a Dashboard with Plotly Dash

Total Successful Launches by Site



- The total number of successful launches by site shows that the highest number of successful launches have come from the KSC LC-39A with 42%
- Following this, CCAFS LC-40 has 29%, VAFB SLC-4E 17% and finally CCAFS SLC-40 with 12%

Highest-scoring launch site success ratio



- The KSC LC-39A launch site has the highest success ratio
- We see here that the large majority of its launches were successful, with a 77% success rate for this site

Section 5

Predictive Analysis (Classification)

Classification Accuracy

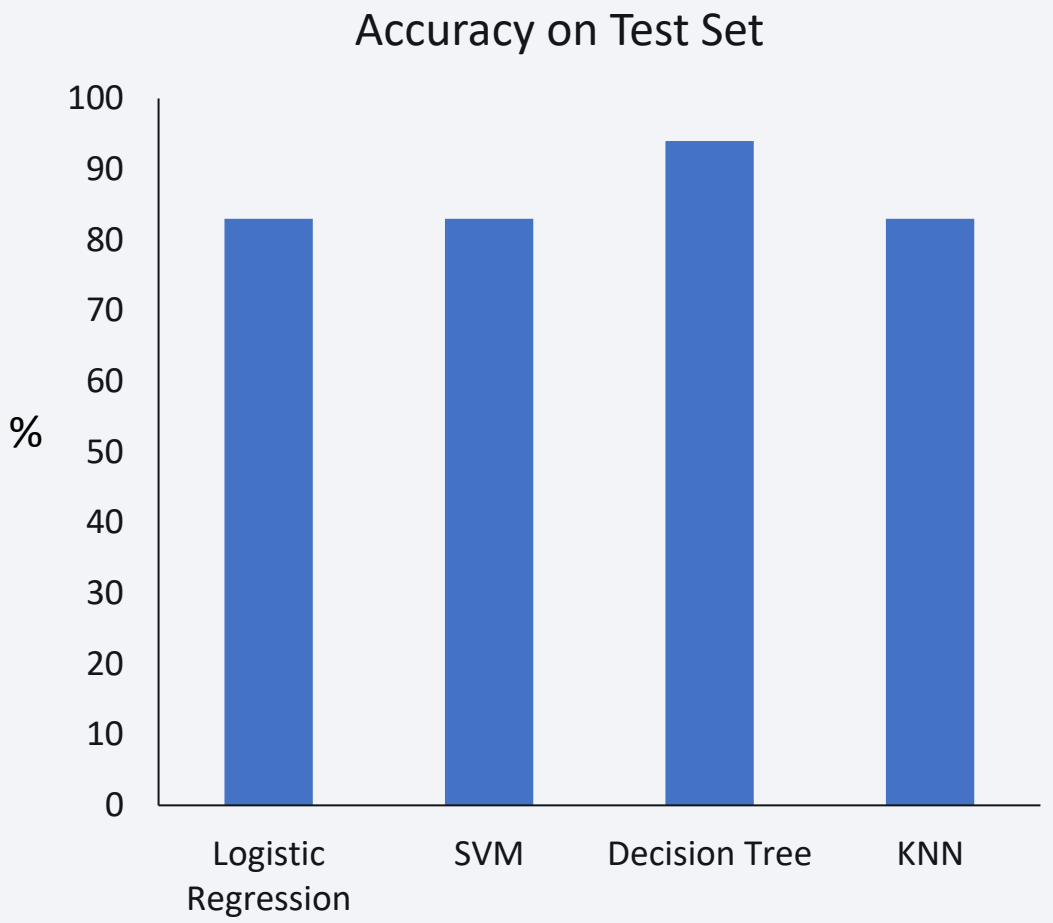
- The following code cell was run at the end of the modelling notebook to provide a side-by-side comparison of model accuracy

Find the method performs best:

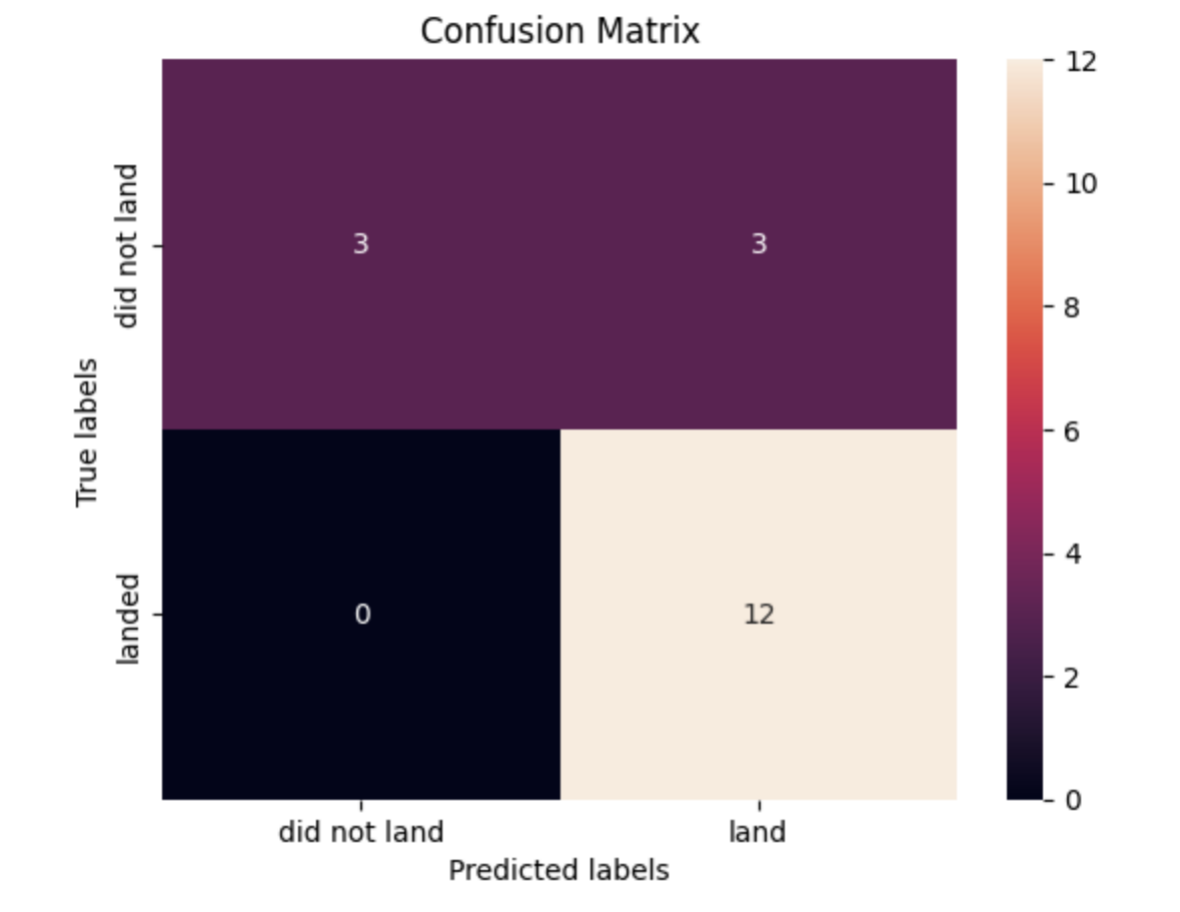
```
methods_considered = {"Logistic Regression": logreg_cv, "SVM": svm_cv, "Decision Tree": tree_cv, "KNN": knn_cv}

results = []

for key, value in methods_considered.items():
    results.append((key, value.score(X_test, Y_test)))

print(results)
```

```
[('Logistic Regression', 0.8333333333333334), ('SVM', 0.8333333333333334), ('Decision Tree', 0.9444444444444444), ('KNN', 0.83
33333333333334)]
```

- From the data we can clearly see that although all models performed well on unseen data, Decision Tree Classifier had the best performance at 94% accuracy



Confusion Matrix: Decision Tree Classifier



- The confusion matrix gives a visual representation of the model's predictions vs actual results
- We see that the model had a good performance, with no false negatives. However, there were some false positives (predictions that a launch did land, when in fact it did not)
- We should be careful of reading too much into these results, since the sample size is extremely small

Conclusions

- Launches were much more likely to be successful in later years
- Landing success rate differs by orbit type
- The classifier models appear to predict launch outcome well, and the best performer is Decision Tree Classifier, with 94% accuracy and no false negatives
- However, the available dataset is small, since there have only been a limited number of launches to date
    - Further work could investigate the statistical significance of our results
    - Analysis could also be re-run with new data as the dataset grows; this would allow a higher level of confidence in the results

Thank you!