IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Anderson McCord

6/9/24

Outline

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

Executive Summary

Methodologies

The analysis began with loading the SpaceX dataset using the pandas library. An initial exploration displayed the first 10 rows to understand the dataset's structure. The percentage of missing values in each column was calculated to assess data completeness, revealing that some columns had missing values. Column types were identified to distinguish between numerical and categorical data.

To analyze the distribution of launches across different SpaceX sites, the LaunchSite column was examined using the value_counts() method. The occurrence and distribution of various orbit types were determined similarly by applying the value_counts() method to the Orbit column.

The outcomes of the missions were analyzed by categorizing the mission outcomes and calculating their occurrences. A set of outcomes indicating unsuccessful landings was identified. A new column, Class, was created to label the success (1) or failure (0) of the first stage landing for each mission. The success rate of the missions was calculated by taking the mean of the Class column.

Finally, the processed dataset was exported to a new CSV file for subsequent analysis phases.

Results

The dataset contained missing values in several columns, but the extent varied. It included both numerical and categorical data types, providing a comprehensive view of launch details. The analysis revealed the distribution of launches across different SpaceX sites, with certain sites having higher launch frequencies. Various orbit types were identified, including Low Earth Orbit (LEO) and Geosynchronous Transfer Orbit (GTO), each serving different mission purposes.

The mission outcomes varied, with several missions successfully landing the first stage on land or ocean platforms, while others were unsuccessful. The overall success rate of first stage landings was calculated, providing a quantitative measure of SpaceX's landing success over the analyzed period.

The processed dataset was saved as "dataset_part_2.csv" for further detailed analysis, ensuring consistency and readiness for subsequent research steps.

Introduction

Background

Space Exploration Technologies Corp. (SpaceX) is a private American aerospace manufacturer and space transportation company founded by Elon Musk in 2002. SpaceX has developed the Falcon 1, Falcon 9, and Falcon Heavy rockets, as well as the Dragon spacecraft. These innovations have significantly impacted the space industry by reducing the cost of space travel and enabling reusable rocket technology. The company has conducted numerous launches, placing satellites into orbit, resupplying the International Space Station (ISS), and testing the feasibility of Mars colonization.

The dataset under analysis includes information about various SpaceX launches, detailing aspects such as launch dates, booster versions, payload masses, orbits, launch sites, and mission outcomes. This dataset provides a comprehensive overview of SpaceX's launch activities, offering valuable insights into the company's performance and operational patterns.

Context

The analysis of the SpaceX launch dataset aims to understand the distribution and success rate of SpaceX launches over time. By examining the data, we can identify patterns and trends that contribute to successful missions, and understand the factors influencing the different outcomes of launches. This analysis not only highlights SpaceX's achievements but also sheds light on areas that require improvement, guiding future strategies and decisions.

Problems:

- **Launch Site Utilization**:
  - What are the most frequently used launch sites by SpaceX?
  - How is the distribution of launches across different SpaceX launch facilities?
- **Orbit Distribution**:
  - Which types of orbits are most commonly targeted by SpaceX launches?
  - How do the different types of orbits (e.g., Low Earth Orbit, Geosynchronous Transfer Orbit) correlate with the mission objectives?
- **Mission Outcomes**:
  - What are the common outcomes of SpaceX missions, and how often do these outcomes occur?
  - What factors contribute to the success or failure of the first stage landing in SpaceX missions?
- **Landing Success Rate**:
  - What is the overall success rate of the first stage landings in SpaceX missions?
  - How has the success rate of landings evolved over time?
- **Impact of Reusability**:
  - How does the reusability of boosters impact the success rate and cost-efficiency of SpaceX launches?
  - What trends can be observed in the reuse of boosters over time?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected via an await fetch
  - This was placed into a pandas dataframe
- Performed data wrangling:
  - Utilizing NumPy and Pandas we split and standardized the data
- Performed exploratory data analysis (EDA) using visualization and SQL
- Performed interactive visual analytics using Folium and Plotly Dash
- Performed predictive analysis using classification models
  - Data is first split into training / test sets. We utilize GridSearchCV We output the GridSearchCV object for logistic regression. We display the best parameters using the data attribute best_params_ and the accuracy on the validation data using the data attribute best_score_.

Data Collection – SpaceX API

From the `rocket` column we would like to learn the booster name.

In [2]:
```python
# Takes the dataset and uses the rocket column to call the API and append the data to the list
def getBoosterVersion(data):
    for x in data['rocket']:
       if x:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])
```

From the `launchpad` we would like to know the name of the launch site being used, the logitude, and the latitude.

In [3]:
```python
# Takes the dataset and uses the launchpad column to call the API and append the data to the list
def getLaunchSite(data):
    for x in data['launchpad']:
       if x:
         response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
         Longitude.append(response['longitude'])
         Latitude.append(response['latitude'])
         LaunchSite.append(response['name'])
```

From the `payload` we would like to learn the mass of the payload and the orbit that it is going to.

In [4]:
```python
# Takes the dataset and uses the payloads column to call the API and append the data to the lists
def getPayloadData(data):
    for load in data['payloads']:
       if load:
        response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
        PayloadMass.append(response['mass_kg'])
        Orbit.append(response['orbit'])
```

version of cores, the number of times this specific core has been reused, and the serial of the core.

In [5]:
```python
# Takes the dataset and uses the cores column to call the API and append the data to the lists
def getCoreData(data):
    for core in data['cores']:
            if core['core'] != None:
                response = requests.get("https://api.spacexdata.com/v4/cores/"+core['core']).json()
                Block.append(response['block'])
                ReusedCount.append(response['reuse_count'])
                Serial.append(response['serial'])
            else:
                Block.append(None)
                ReusedCount.append(None)
                Serial.append(None)
            Outcome.append(str(core['landing_success'])+' '+str(core['landing_type']))
            Flights.append(core['flight'])
            GridFins.append(core['gridfins'])
            Reused.append(core['reused'])
            Legs.append(core['legs'])
            LandingPad.append(core['landpad'])
```

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

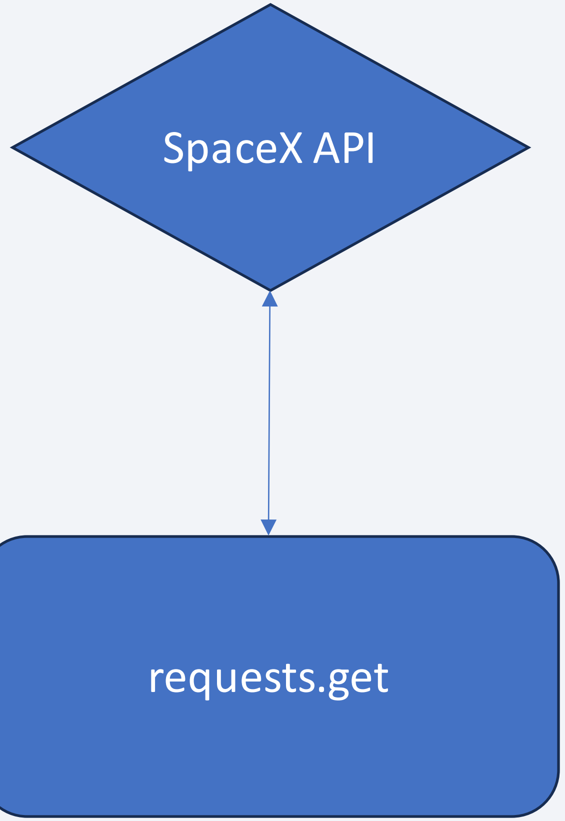
In [6]:
```python
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

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

Check the content of the response

In [8]:
```python
print(response.content)
```

b'[{"fairings":{"reused":false,"recovery_attempt":false,"recovered":false,"ships":[]},"links":{"patch":{"small":"https://images2.imgbox.com/94/f2/NN6Ph45r_o.png","large":"https://images2.imgbox.com/5b/02/QcxHUb5V_o.png"},"reddit":{"campaign":null,"launch":null,"media":null,"recovery":null},"flickr":{"small":[],"original":[]},"presskit":null,"webcast":"https://www.youtube.com/watch?v=0a_00nJ_Y88","youtube_id":"0a_00nJ_Y88","article":"https://www.space.com/2196-spacex-inaugural-falcon-1-rocket-lost-launch.html","wikipedia":"https://en.wikipedia.org/wiki/DemoSat"},"static_fire_date_utc":"2006-03-17T00:00:00.000Z","static_fire_date_unix":1142553600,"net":false,"window":0,"rocket":"5e9d0d95eda69955f709d1eb","success":false,"failures":[{"time":33,"altitude":null,"reason":"merlin engine failure"}],"details":"Engine failure at 33 seconds and loss of vehicle","crew":[],"ships":[],"capsules":[],"payloads":["5eb0e4b5b6c3bb0006eeb1e1"],"launchpad":"5e9e4502f5090995de566f86","flight_number":



Data Collection - Scraping

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

```
from pandas import json_normalize

df = json_normalize(json_data)
```

Using the dataframe `data` print the first 5 rows

```
# Get the head of the dataframe
print(df.head())
```

```
      static_fire_date_utc  static_fire_date_unix    net  window  \
0  2006-03-17T00:00:00.000Z           1.142554e+09  False     0.0
1                      None                    NaN  False     0.0
2                      None                    NaN  False     0.0
3  2008-09-20T00:00:00.000Z           1.221869e+09  False     0.0
4                      None                    NaN  False     0.0

                     rocket  success  \
0  5e9d0d95eda69955f709d1eb    False
1  5e9d0d95eda69955f709d1eb    False
2  5e9d0d95eda69955f709d1eb    False
3  5e9d0d95eda69955f709d1eb     True
4  5e9d0d95eda69955f709d1eb     True

                                                                                              failures  \
0                                      [{'time': 33, 'altitude': None, 'reason': 'merlin engine failure'}]
1                [{'time': 301, 'altitude': 289, 'reason': 'harmonic oscillation leading to premature engine shutdown'}]
2  [{'time': 140, 'altitude': 35, 'reason': 'residual stage-1 thrust led to collision between stage 1 and stage 2'}]
3                                                                                                       []
4                                                                                                       []
```

My GitHub https://github.com/thebigduck/IBMCapstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

Data Wrangling

TASK 1: Calculate the number of launches on each site

The data contains several Space X launch facilities: Cape Canaveral Space Launch Complex 40 **VAFB SLC 4E** , Vandenberg Air Force Base Space Launch Complex 4E **(SLC-4E)**, Kennedy Space Center Launch Complex 39A **KSC LC 39A** .The location of each Launch Is placed in the column `LaunchSite`

Next, let's see the number of launches for each site.

Use the method `value_counts()` on the column `LaunchSite` to determine the number of launches on each site:

[5]:
```
# Apply value_counts() on column LaunchSite
launches_per_site = df['LaunchSite'].value_counts()
launches_per_site
```

TASK 2: Calculate the number and occurrence of each orbit

Use the method `.value_counts()` to determine the number and occurrence of each orbit in the column `Orbit`

In [6]:
```
# Apply value_counts on Orbit column
orbits_occurrence = df['Orbit'].value_counts()
orbits_occurrence
```

Out[6]:
```
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
```

TASK 3: Calculate the number and occurence of mission outcome of the orbits

Use the method `.value_counts()` on the column `Outcome` to determine the number of `landing_outcomes` .Then assign it to a variable landing_outcomes.

In [7]:
```
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
landing_outcomes
```

Out[7]:
```
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
```

`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 to a drone ship `False ASDS` means the mission outcome was unsuccessfully landed to a drone ship. `None ASDS` and `None None` these represent a failure to land.

In [8]:
```
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
```

- My GitHub = https://github.com/thebigduck/IBMCapstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

- The dataset was imported via Pandas:

- GitHub= https://github.com/thebigduck/IBMCapstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

EDA with SQL pt1

- **Create a New Table**: `CREATE TABLE SPACEXTABLE AS SELECT * FROM SPACEXTBL WHERE Date IS NOT NULL`
  - Creates a new table called SPACEXTABLE by copying all rows from SPACEXTBL where the Date is not null.
- `SELECT DISTINCT "Launch_Site" FROM SPACEXTABLE`
  - Retrieves a list of unique launch sites from the SPACEXTABLE.
- `SELECT * FROM SPACEXTABLE WHERE "Launch_Site" LIKE 'CCA%' LIMIT 5`
  - Displays the first 5 records from SPACEXTABLE where the launch site name starts with 'CCA'.
- `SELECT SUM("PAYLOAD_MASS__KG_") AS Total_Payload_Mass FROM SPACEXTABLE WHERE "Customer" = 'NASA (CRS)'`
  - Calculates the total payload mass carried by boosters for NASA (CRS) missions.
- `SELECT AVG("PAYLOAD_MASS__KG_") AS Average_Payload_Mass FROM SPACEXTABLE WHERE "Booster_Version" = 'F9 v1.1'`
  - Calculates the average payload mass carried by the booster version 'F9 v1.1'.
- `SELECT MIN("Date") AS First_Successful_Landing FROM SPACEXTABLE WHERE "Landing_Outcome" = 'Success (ground pad)'`
  - Identifies the date of the first successful landing on a ground pad.

EDA with SQL pt2

- `SELECT "Booster_Version" FROM SPACEXTABLE WHERE "Landing_Outcome" = 'Success (drone ship)' AND "PAYLOAD_MASS__KG_" > 4000 AND "PAYLOAD_MASS__KG_" < 6000`
    - Lists booster versions that successfully landed on a drone ship and carried a payload mass between 4000 and 6000 kg.
- `SELECT "Mission_Outcome", COUNT(*) AS Total_Count FROM SPACEXTABLE GROUP BY "Mission_Outcome"`
    - Counts the total number of missions with each outcome (success or failure).
- `SELECT "Booster_Version" FROM SPACEXTABLE WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX("PAYLOAD_MASS__KG_") FROM SPACEXTABLE)`
    - Lists the booster versions that carried the maximum payload mass.
- `SELECT strftime('%m', "Date") AS Month, "Landing_Outcome", "Booster_Version", "Launch_Site" FROM SPACEXTABLE WHERE "Landing_Outcome" LIKE 'Failure (drone ship)' AND strftime('%Y', "Date") = '2015'`
    - Lists the records of failed drone ship landings in 2015, including the month, landing outcome, booster version, and launch site.
- `SELECT "Landing_Outcome", COUNT(*) AS Outcome_Count FROM SPACEXTABLE WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY "Landing_Outcome" ORDER BY Outcome_Count DESC`
    - Ranks the count of different landing outcomes (such as 'Failure (drone ship)' or 'Success (ground pad)') between June 4, 2010, and March 20, 2017, in descending order.

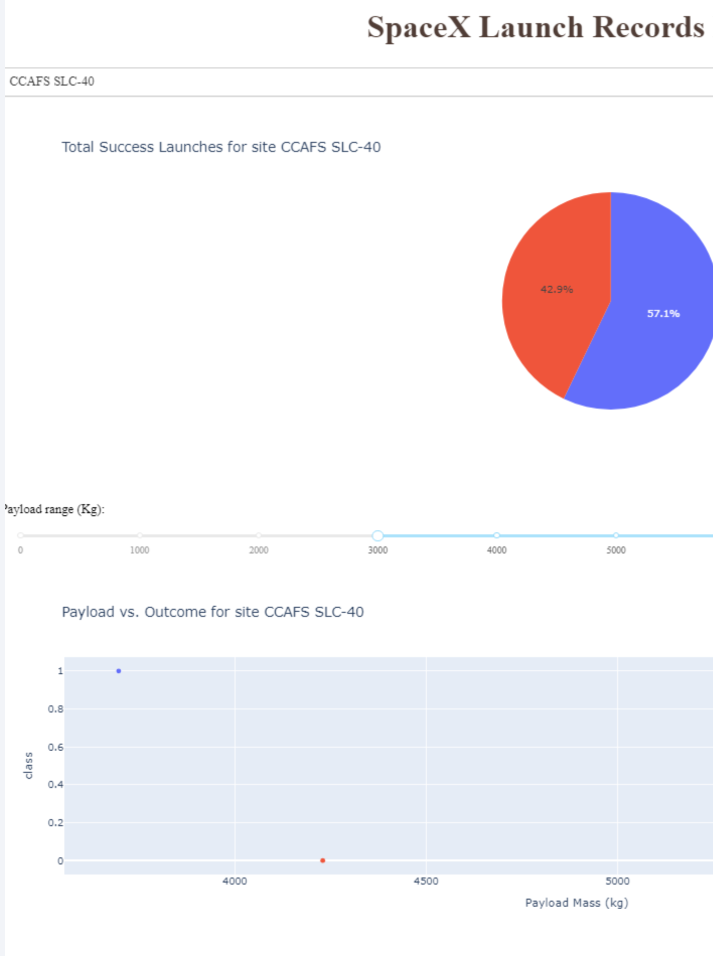
Build an Interactive Map with Folium

- Mark down a point on the closest coastline using MousePosition and calculated the distance
- Similarly, I draw a line between a launch site to its closest city, railway, highway
- These were marked to assist in visually assessing the location of the launch sites to their surroundings
- We utilized this as well as the success rate of the locations to gather additonal insights

- Github= https://github.com/thebigduck/IBMCapstone/blob/main/lab_jupyter_launch_site_location.ipynb

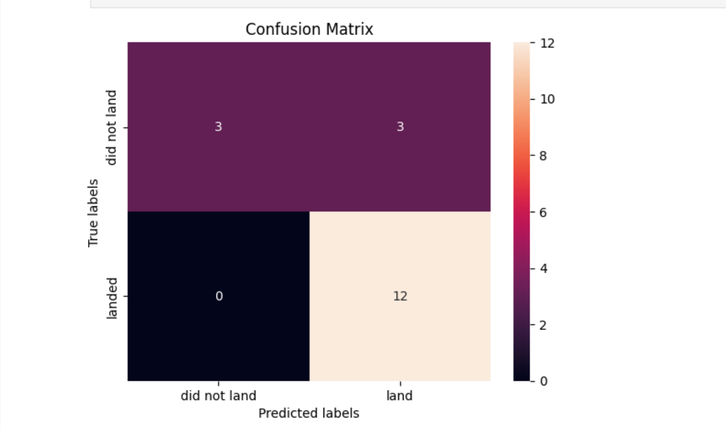
Build a Dashboard with Plotly Dash

- Created a launch records dashboard, which includes a dropdown for the launch site, and payload adjuster for filtering purposes
- Success rate for the sites are displayed via Pie chart. The outcome based on payload mass is displayed at the bottom on a separate plot
- Add the GitHub URL of your completed Plotly Dash lab, as an external reference and peer-review purpose



Predictive Analysis (Classification)

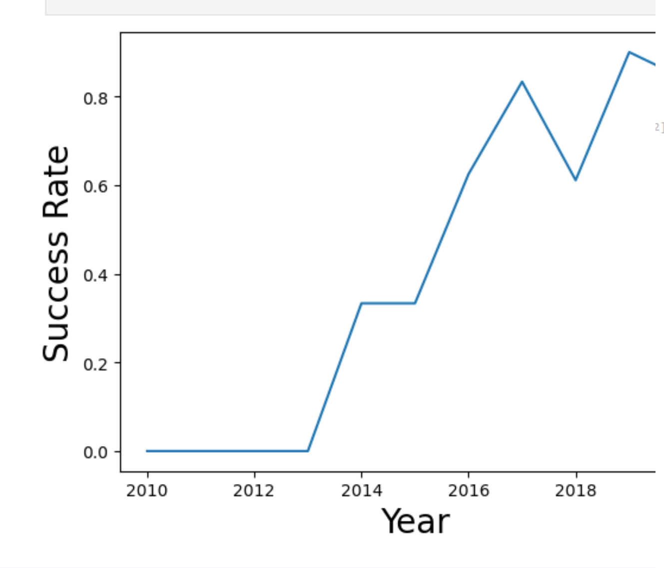
- Data was split the into training and testing sets to evaluate the various Models
- Performed extensive hyperparameter tuning using grid search and random search to find the optimal parameters for each model.

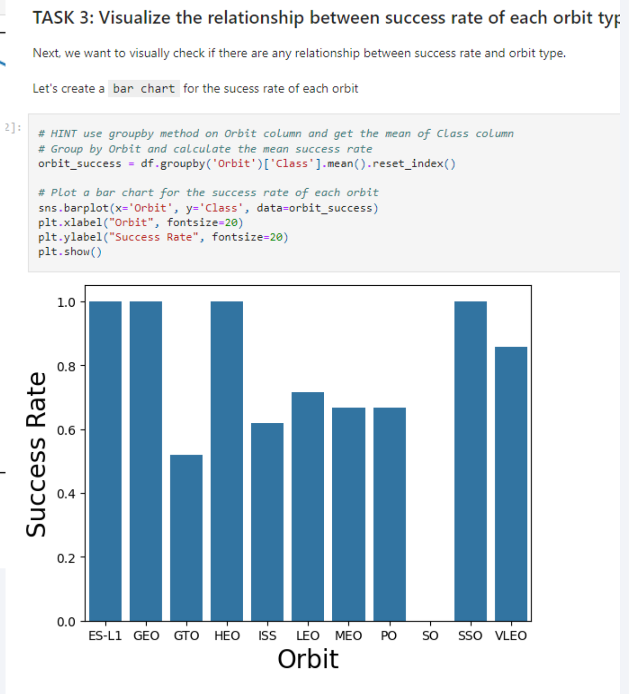


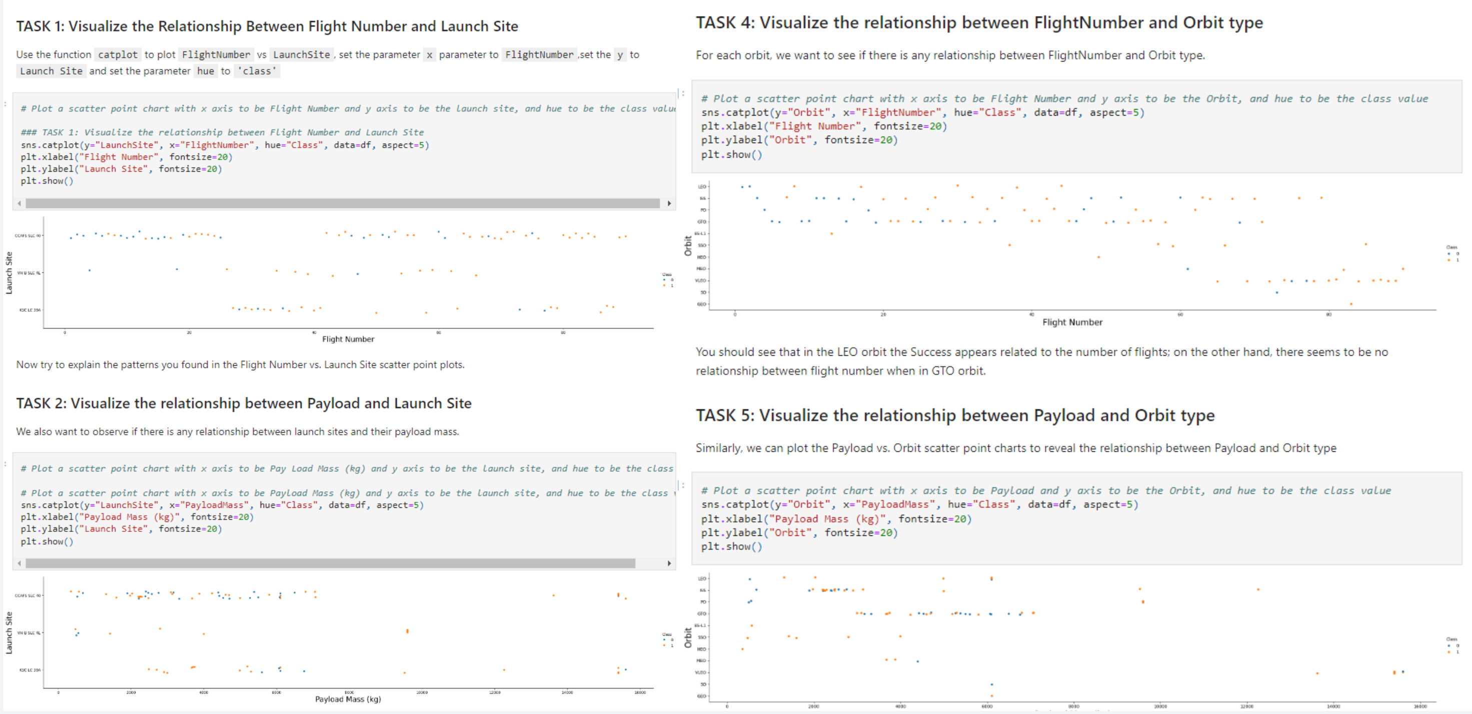
- https://github.com/thebigduck/IBMCapstone/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

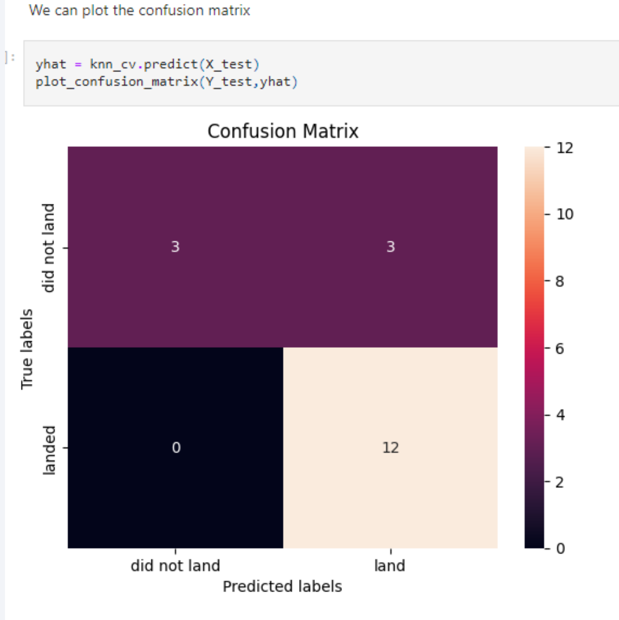
Results

- Predictive analysis results:
  - The best performing method is K-Nearest Neighbors with an accuracy of 0.85









Section 2

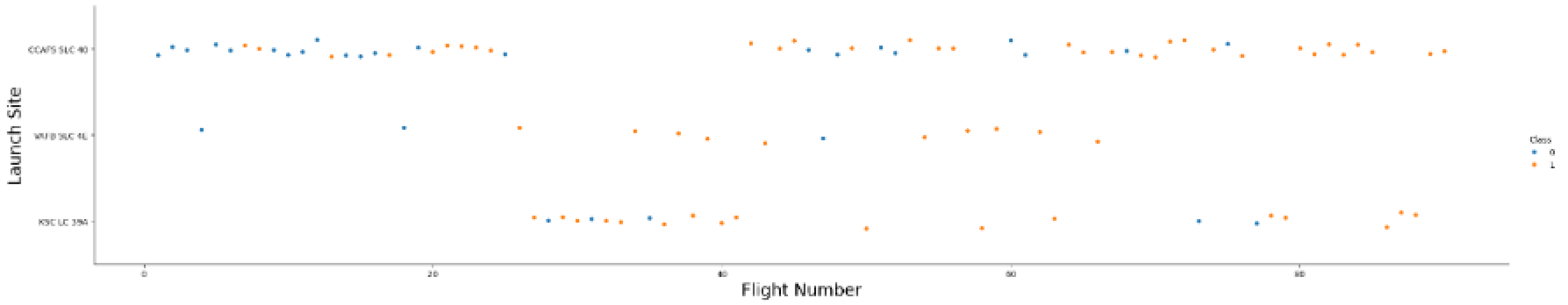
Insights drawn from EDA

Flight Number vs. Launch Site

TASK 1: Visualize the Relationship Between Flight Number and Launch Site

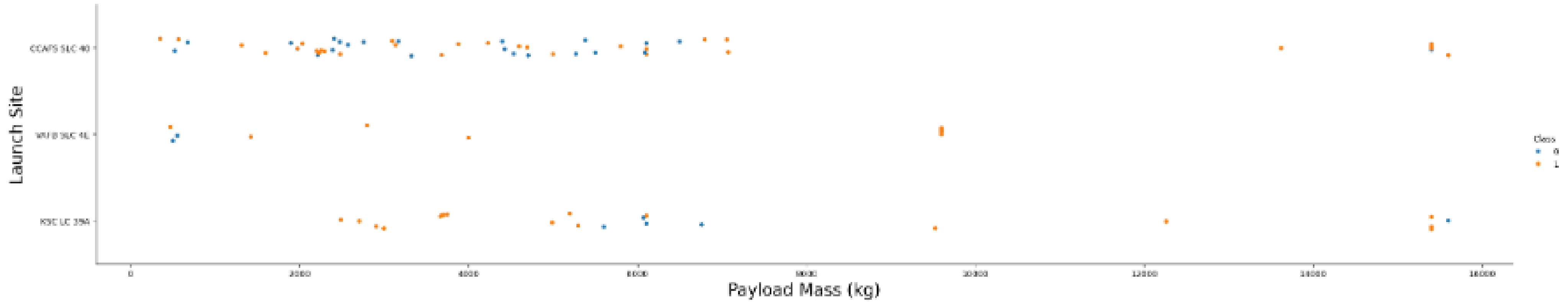
Use the function `catplot` to plot `FlightNumber` vs `LaunchSite`, set the parameter `x` parameter to `FlightNumber`,set the `y` to `Launch Site` and set the parameter `hue` to `'class'`

```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class va

### TASK 1: Visualize the relationship between Flight Number and Launch Site
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect=5)
plt.xlabel("Flight Number", fontsize=20)
plt.ylabel("Launch Site", fontsize=20)
plt.show()
```

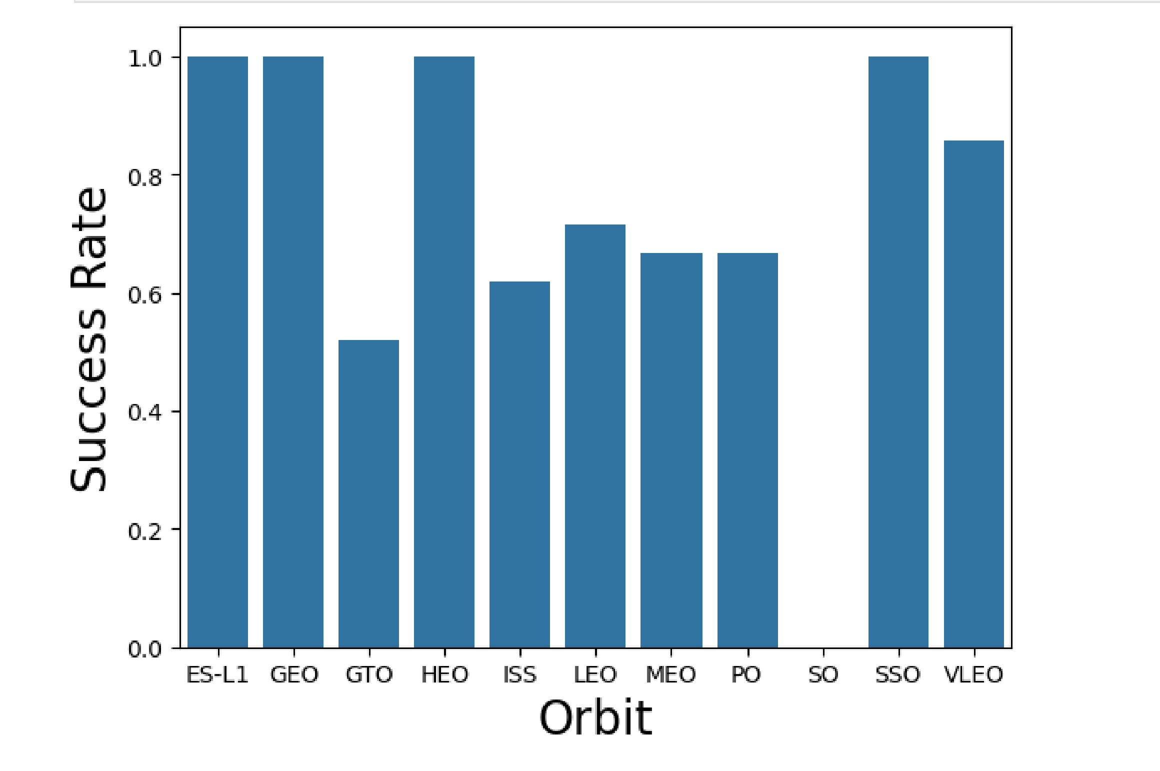


Payload vs. Launch Site

```
]:
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the cla

# Plot a scatter point chart with x axis to be Payload Mass (kg) and y axis to be the launch site, and hue to be the clas
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect=5)
plt.xlabel("Payload Mass (kg)", fontsize=20)
plt.ylabel("Launch Site", fontsize=20)
plt.show()
```



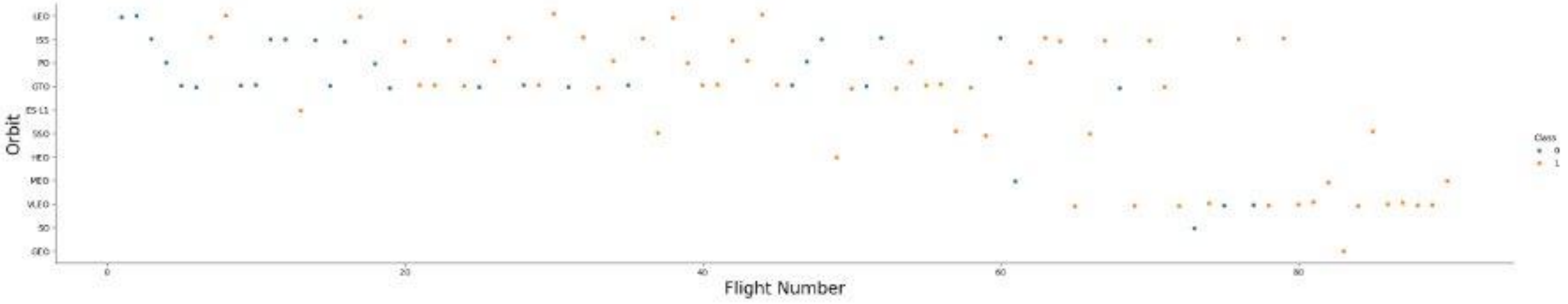
Success Rate vs. Orbit Type



Flight Number vs. Orbit Type

For each orbit, we want to see if there is any relationship between FlightNumber and Orbit type.

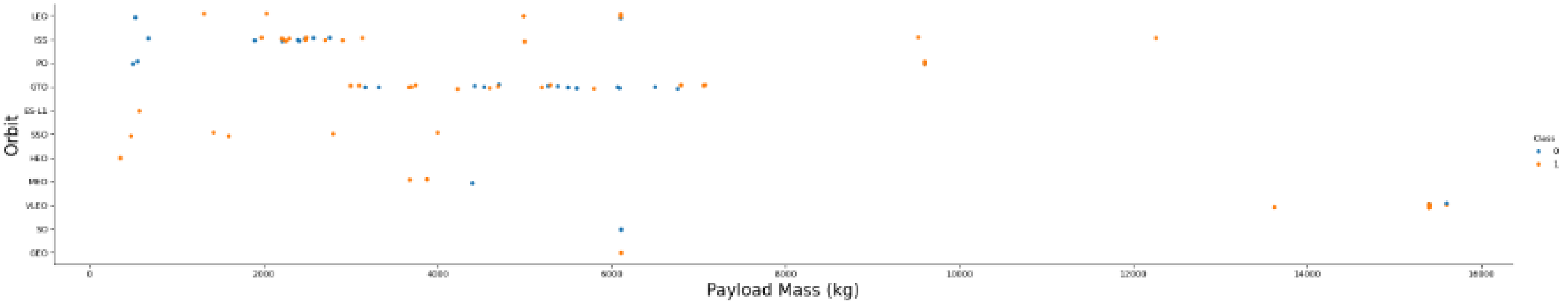
```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="FlightNumber", hue="Class", data=df, aspect=5)
plt.xlabel("Flight Number", fontsize=20)
plt.ylabel("Orbit", fontsize=20)
plt.show()
```



Payload vs. Orbit Type

Similarly, we can plot the Payload vs. Orbit scatter point charts to reveal the relationship between Payload and Orbit type

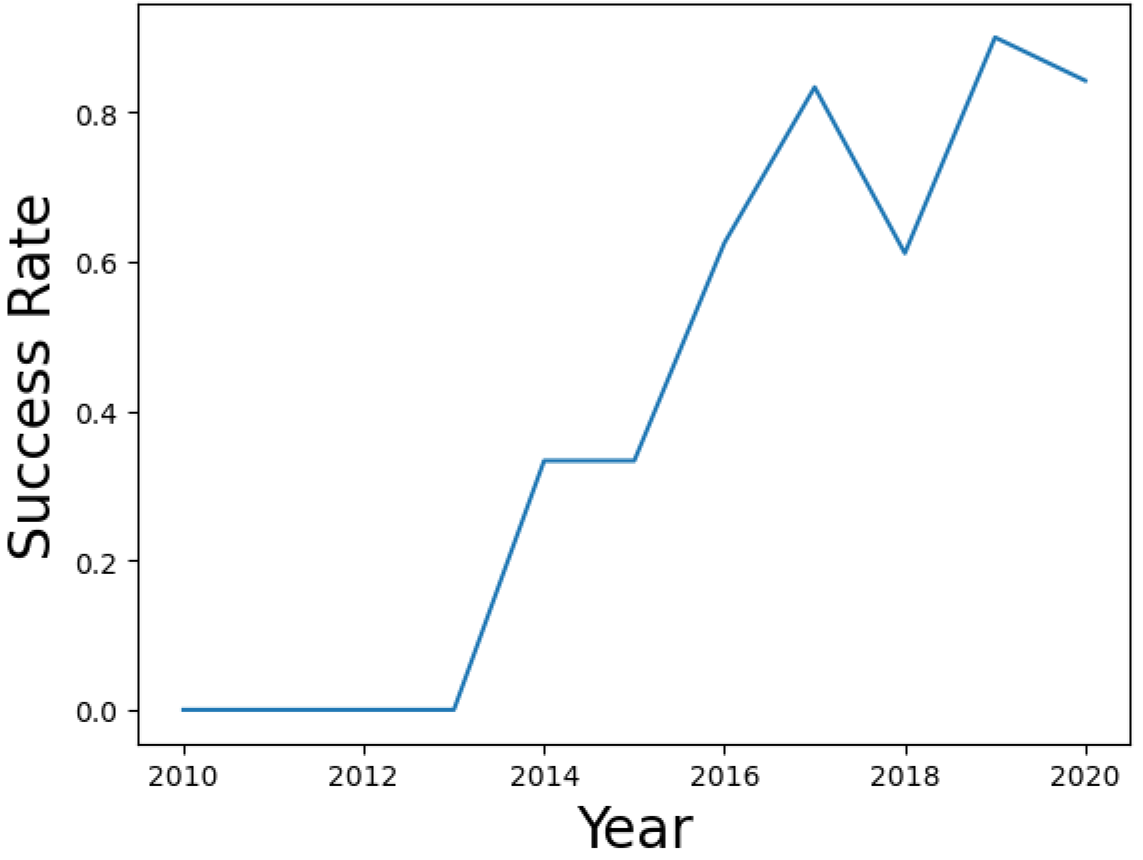
```
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df, aspect=5)
plt.xlabel("Payload Mass (kg)", fontsize=20)
plt.ylabel("Orbit", fontsize=20)
plt.show()
```



With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.

Launch Success Yearly Trend



you can observe that the sucess rate since 2013 kept increasing till 2020

All Launch Site Names

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

[14]:
```
%%sql
SELECT DISTINCT "Launch_Site"
FROM SPACEXTABLE;
```

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

[14]:

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

Launch Site Names Begin with 'CCA'

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	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	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

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

```
%%sql
SELECT SUM("PAYLOAD_MASS__KG_") AS Total_Payload_Mass
FROM SPACEXTABLE
WHERE "Customer" = 'NASA (CRS)';
```

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

Total_Payload_Mass
45596

Average Payload Mass by F9 v1.1

Task 4

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

```
%%sql
SELECT AVG("PAYLOAD_MASS__KG_") AS Average_Payload_Mass
FROM SPACEXTABLE
WHERE "Booster_Version" = 'F9 v1.1';
```

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

Average_Payload_Mass
2928.4

First Successful Ground Landing Date

```
[18]: %%sql
SELECT MIN("Date") AS First_Successful_Landing
FROM SPACEXTABLE
WHERE "Landing_Outcome" = 'Success (ground pad)';

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

[18]:

First_Successful_Landing
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

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

```
%%sql
SELECT "Booster_Version"
FROM SPACEXTABLE
WHERE "Landing_Outcome" = 'Success (drone ship)'
  AND "PAYLOAD_MASS__KG_" > 4000
  AND "PAYLOAD_MASS__KG_" < 6000;
```

```
 * 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

Task 7

List the total number of successful and failure mission outcomes

```
%%sql
SELECT "Mission_Outcome", COUNT(*) AS Total_Count
FROM SPACEXTABLE
GROUP BY "Mission_Outcome";
```

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

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

Boosters Carried Maximum Payload

Task 8

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

[21]:
```
%%sql
SELECT "Booster_Version"
FROM SPACEXTABLE
WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX("PAYLOAD_MASS__KG_") FROM SPACEXTABLE);
```

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

[21]:

Booster_Version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

2015 Launch Records

```
%%sql
SELECT strftime('%m', "Date") AS Month, "Landing_Outcome", "Booster_Version", "Launch_Site"
FROM SPACEXTABLE
WHERE "Landing_Outcome" LIKE 'Failure (drone ship)'
  AND strftime('%Y', "Date") = '2015';
```

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

Month	Landing_Outcome	Booster_Version	Launch_Site
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

```
%%sql
SELECT "Landing_Outcome", COUNT(*) AS Outcome_Count
FROM SPACEXTABLE
WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY "Landing_Outcome"
ORDER BY Outcome_Count DESC;
```

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

Landing_Outcome	Outcome_Count
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

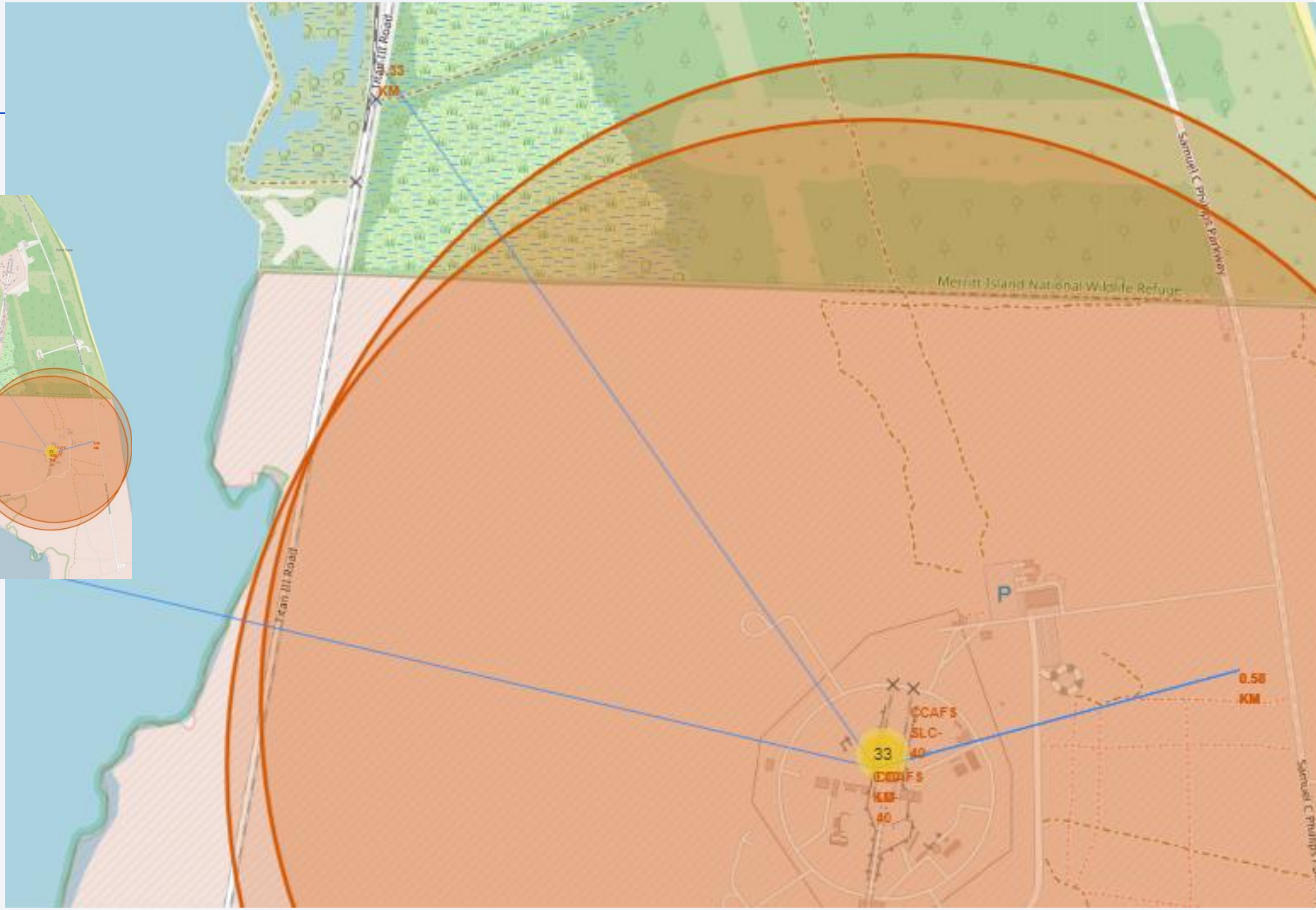
Section 3

Launch Sites Proximities Analysis

Launch Site Screenshots

Launches by success or failure

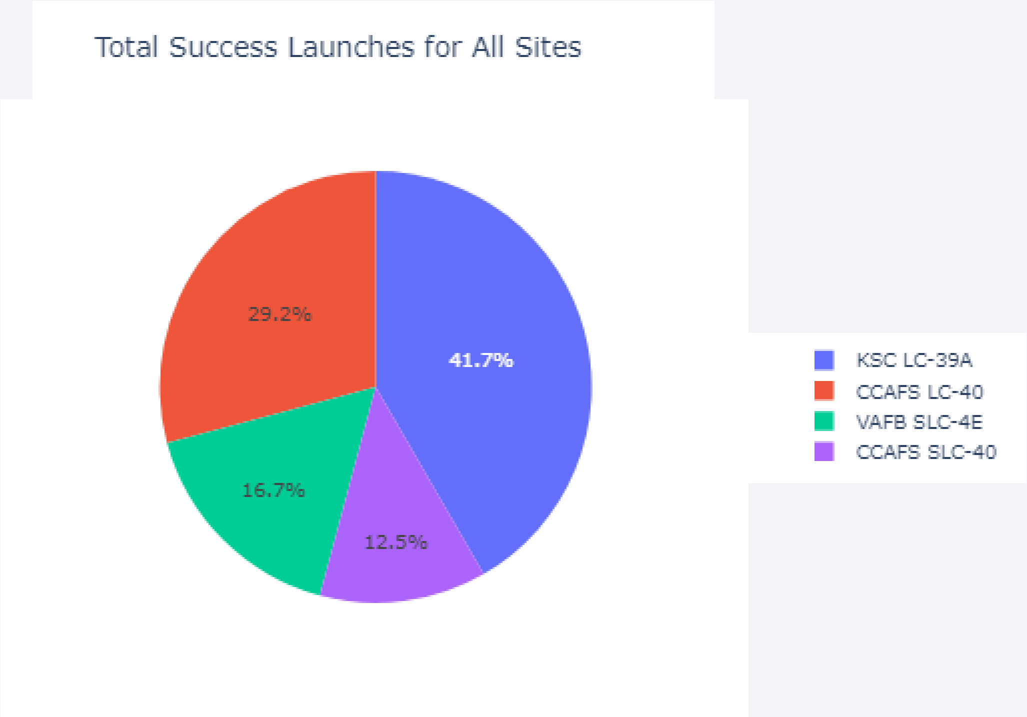
Distances


Merritt Island National Wildlife Refuge
Samuel C Phillips Parkway
0.58 KM
33
CCAFS SLC-40
P

Section 4

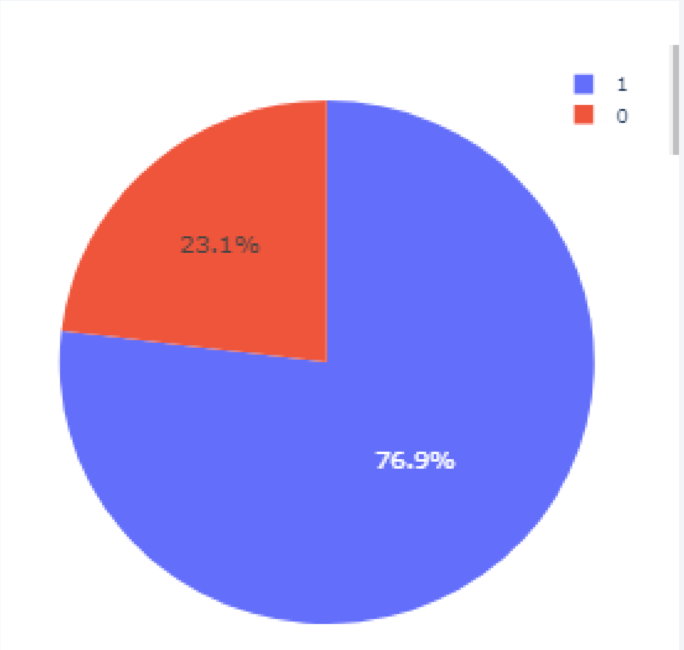
Build a Dashboard with Plotly Dash

Launch success count for all sites



Launch site with highest launch success

Total Success Launches for site KSC LC-39A



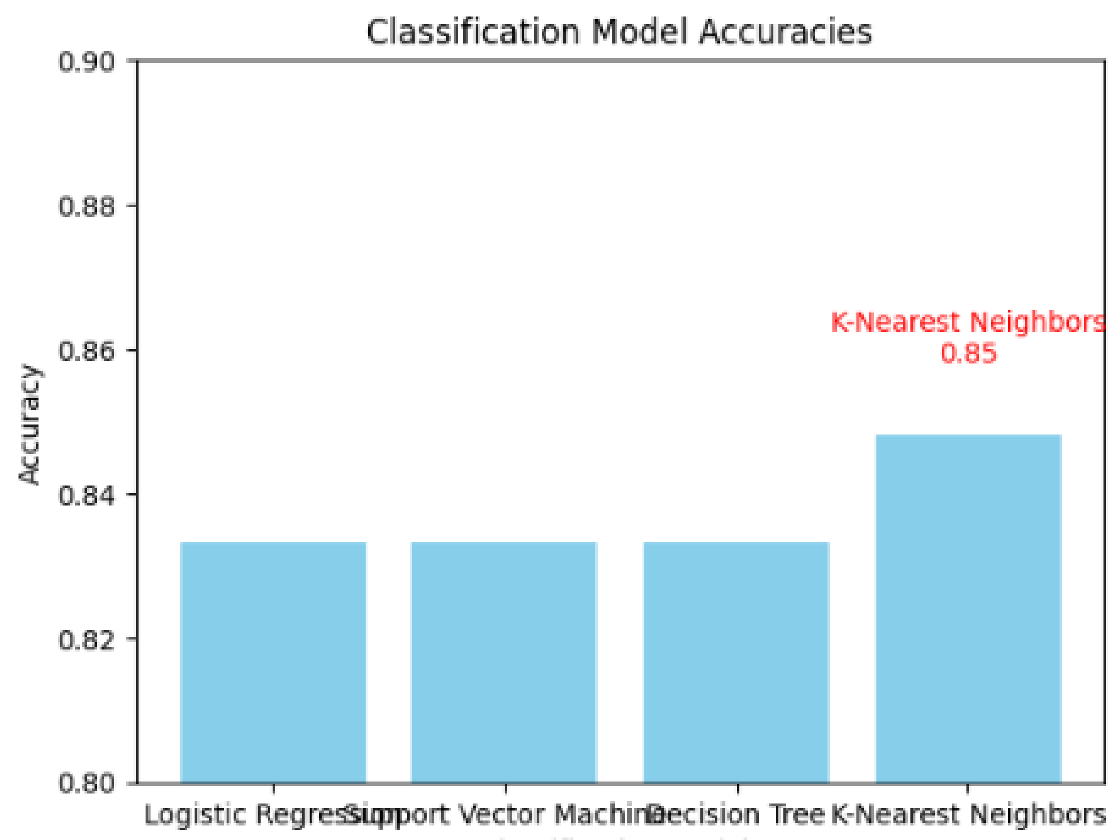
Payload vs. Launch Outcome

Section 5

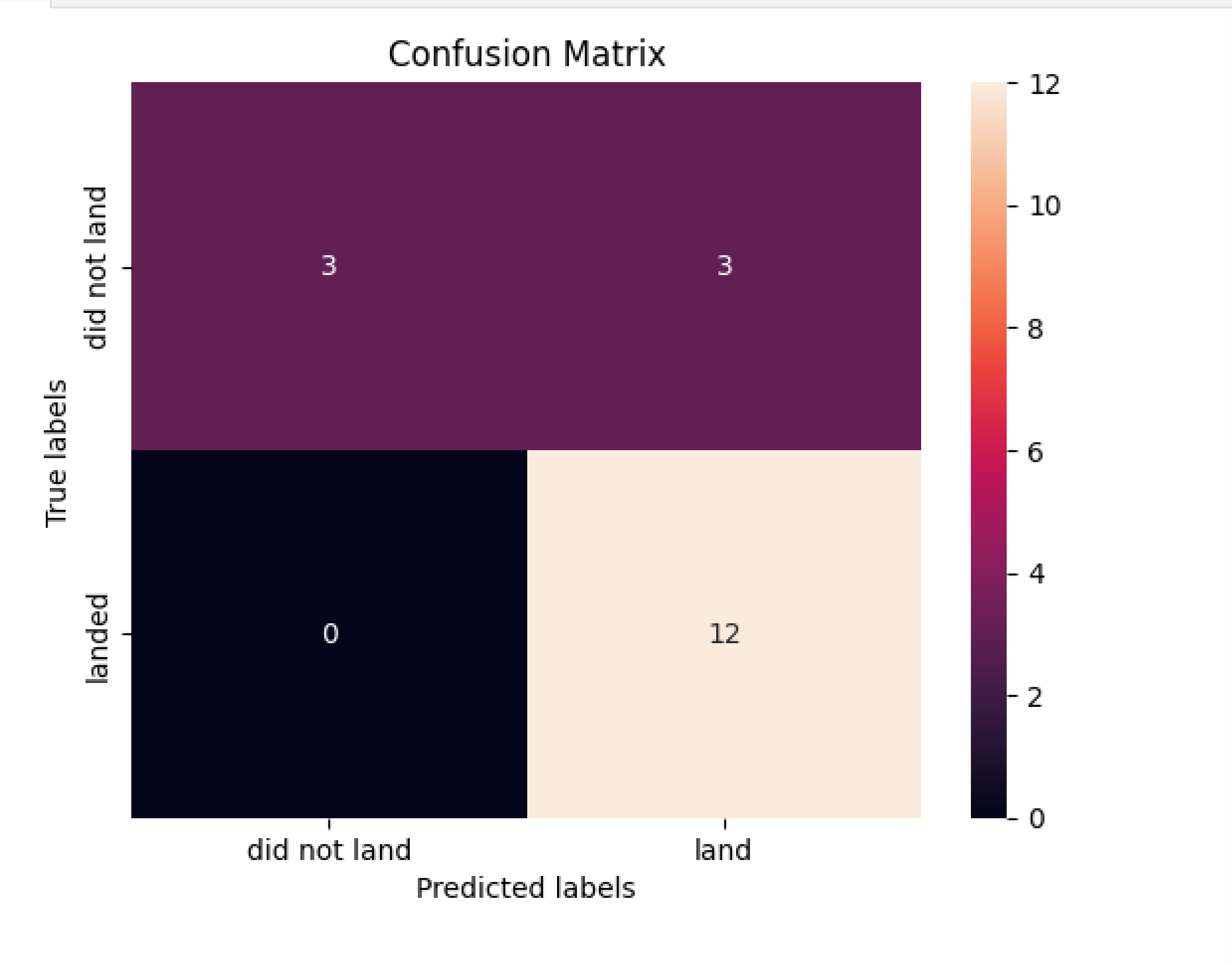
Predictive Analysis (Classification)

Classification Accuracy

```
plt.show()
```



Confusion Matrix



Find the method performs best:

[41]:
```
accuracies = {
    "Logistic Regression": logreg_test_accuracy,
    "Support Vector Machine": svm_test_accuracy,
    "Decision Tree": tree_test_accuracy,
    "K-Nearest Neighbors": knn_test_accuracy
}

best_model = max(accuracies, key=accuracies.get)
print(f"The best performing method is {best_model} with an accuracy of {accuracies[best_model]:.2f}")
```

```
The best performing method is K-Nearest Neighbors with an accuracy of 0.85
```

Conclusions

- The best performing method is K-Nearest Neighbors with an accuracy of 0.85

Thank you!