APPLIED DATA SCIENCE CAPSTONE

BY MARIANA SANTOS

EXECUTIVE SUMMARY

- SpaceX revolutionizes the space industry by offering Falcon 9 rocket launches at a significantly lower cost of 62 million dollars compared to other providers' prices exceeding 165 million dollars per launch. The key to this cost efficiency lies in SpaceX's ability to reuse the first stage of the rocket. By accurately predicting the successful landing of the first stage, it is possible to estimate the total cost of a launch. This crucial information becomes valuable for potential competitors looking to challenge SpaceX in the rocket launch market.

INTRODUCTION

- SpaceX, a trailblazer in the aerospace industry, has disrupted traditional norms by offering Falcon 9 rocket launches at a fraction of the cost compared to other providers. Priced at 62 million dollars per launch, SpaceX's competitive advantage stems from its innovative approach of reusing the first stage of the Falcon 9 rocket. This cost-saving strategy not only drives down expenses but also sets a new standard for efficiency in space exploration. The ability to predict the successful landing of the first stage plays a pivotal role in determining the overall cost of a launch, giving SpaceX a strategic edge over competitors.
- the stage for exploring how data insights can empower informed decision-making and strategic planning in the realm of space technology and exploration.

METHODOLOGY

- Data Collection
- Data Collection with Web Scraping
- Data Wrangling
- Data Analysis using SQL
- EDA with Visualization
- Interactive Visual Analytics and Dashboard
- Predictive Analysis

DATA COLLECTION AND DATA WRANGLING

62]:

	Column1	Column2	Column3	FlightNumber
0	1	A	10.1	1
1	2	B	20.2	2
2	3	C	30.3	3
3	4	D	40.4	4
4	5	E	50.5	5

DATA COLLECTION AND DATA WRANGLING

```
print(df)
```

```
   Column1 Column2  Column3
0        1       A     10.1
1        2       B     20.2
2        3       C     30.3
3        4       D     40.4
4        5       E     50.5
```

Show the summary of the dataframe

```
# Display the head of the DataFrame (the first few rows)
dataframe_head = df.head()

# Print the head of the DataFrame
print(dataframe_head)
```

```
   Column1 Column2  Column3
0        1       A     10.1
1        2       B     20.2
2        3       C     30.3
3        4       D     40.4
4        5       E     50.5
```

DATA COLLECTION AND DATA WRANGLING

Task 2: Filter the dataframe to only include `Falcon 9` launches

Finally we will remove the Falcon 1 launches keeping only the Falcon 9 launches. Filter the data dataframe using the `BoosterVersion` column to only keep the Falcon 9 launches. Save the filtered data to a new called `data_falcon9`.

```
[5]: # Hint data['BoosterVersion']!='Falcon 1'
# Assuming 'df' is your Pandas DataFrame
# Display summary statistics for numerical columns
summary_statistics = df.describe()

# Display concise summary of the DataFrame
dataframe_info = df.info()

# Print the summary statistics and DataFrame info
print("Summary Statistics:")
print(summary_statistics)
print("\nDataFrame Info:")
print(dataframe_info)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Column1  5 non-null      int64
 1   Column2  5 non-null      object
 2   Column3  5 non-null      float64
dtypes: float64(1), int64(1), object(1)
memory usage: 248.0+ bytes
Summary Statistics:
        Column1    Column3
count  5.000000   5.000000
mean   3.000000  30.300000
std    1.581139  15.969502
min    1.000000  10.100000
25%    2.000000  20.200000
50%    3.000000  30.300000
75%    4.000000  40.400000
max    5.000000  50.500000

DataFrame Info:
None
```

Now that we have removed some values we should reset the FlgihtNumber column

DATA COLLECTION AND DATA WRANGLING

[47]:
```python
# Assuming 'df' is your Pandas DataFrame
# Resetting the 'FlightNumber' column
df['FlightNumber'] = range(1, len(df) + 1)

# Display summary statistics for numerical columns
summary_statistics = df.describe()

# Display concise summary of the DataFrame
dataframe_info = df.info()

# Print the summary statistics and DataFrame info
print("Summary Statistics:")
print(summary_statistics)
print("\nDataFrame Info:")
print(dataframe_info)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Column1       5 non-null      int64
 1   Column2       5 non-null      object
 2   Column3       5 non-null      float64
 3   FlightNumber  5 non-null      int64
dtypes: float64(1), int64(2), object(1)
memory usage: 288.0+ bytes
Summary Statistics:
        Column1    Column3  FlightNumber
count  5.000000   5.000000      5.000000
mean   3.000000  30.300000      3.000000
std    1.581139  15.969502      1.581139
min    1.000000  10.100000      1.000000
25%    2.000000  20.200000      2.000000
50%    3.000000  30.300000      3.000000
75%    4.000000  40.400000      4.000000
max    5.000000  50.500000      5.000000

DataFrame Info:
None
```

DATA WRANGLING

Data Wrangling

We can see below that some of the rows are missing values in our dataset.

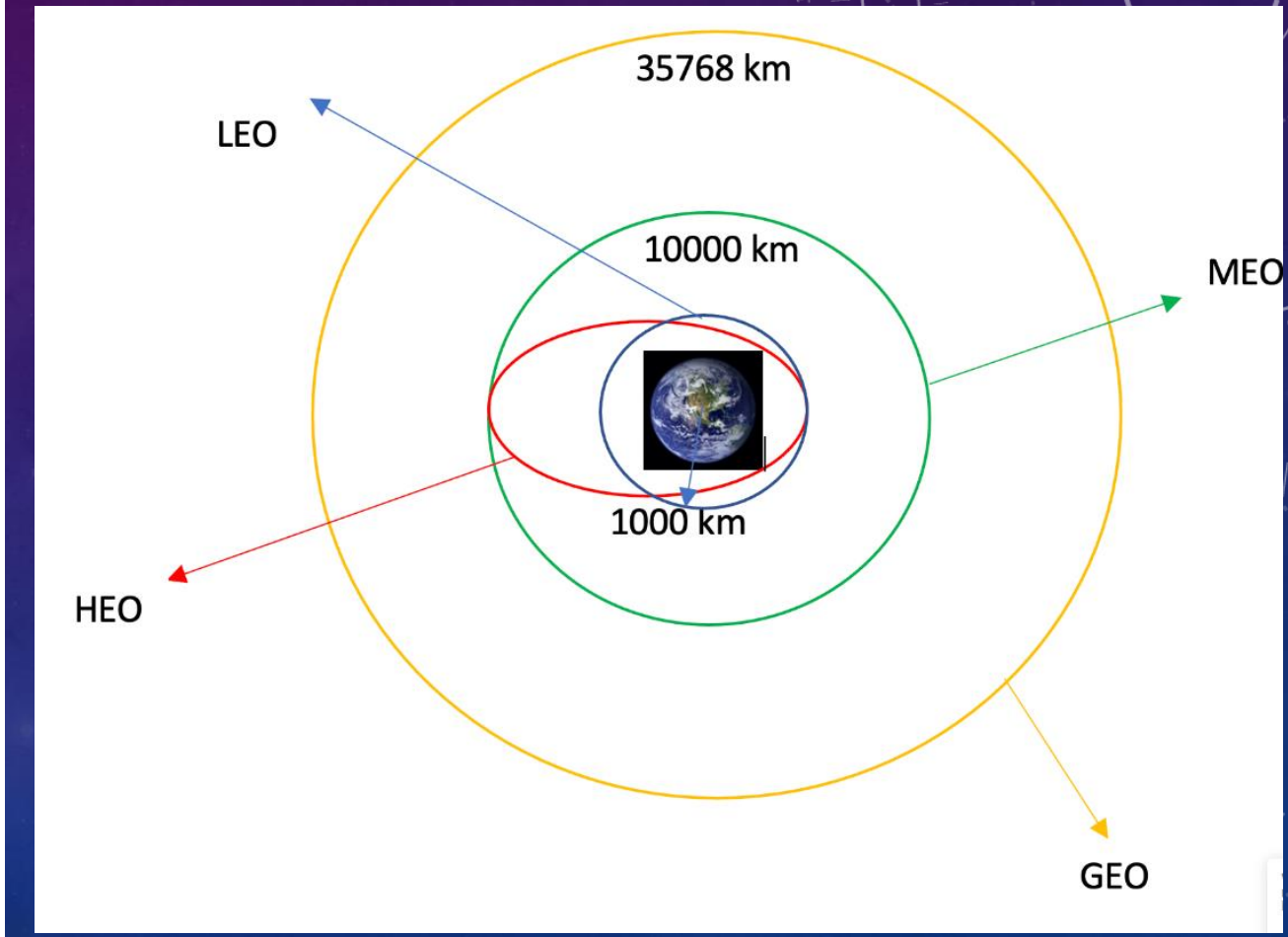
```
48]: # Fill missing values with zeros
df_filled = df.fillna(0)

# Display concise summary of the DataFrame after filling missing values
filled_dataframe_info = df_filled.info()

# Print the DataFrame info after filling missing values
print("DataFrame Info after filling missing values with zeros:")
print(filled_dataframe_info)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Column1       5 non-null      int64
 1   Column2       5 non-null      object
 2   Column3       5 non-null      float64
 3   FlightNumber  5 non-null      int64
dtypes: float64(1), int64(2), object(1)
memory usage: 288.0+ bytes
DataFrame Info after filling missing values with zeros:
None
```

Before we can continue we must deal with these missing values. The `LandingPad` column will retain None values to represent when landing pads were not used.

DATA WRANGLING

```
   FlightNumber        Date BoosterVersion  PayloadMass Orbit    LaunchSite  \
0             1  2010-06-04       Falcon 9  6104.959412   LEO  CCAFS SLC 40
1             2  2012-05-22       Falcon 9   525.000000   LEO  CCAFS SLC 40
2             3  2013-03-01       Falcon 9   677.000000   ISS  CCAFS SLC 40
3             4  2013-09-29       Falcon 9   500.000000    PO   VAFB SLC 4E
4             5  2013-12-03       Falcon 9  3170.000000   GTO  CCAFS SLC 40
5             6  2014-01-06       Falcon 9  3325.000000   GTO  CCAFS SLC 40
6             7  2014-04-18       Falcon 9  2296.000000   ISS  CCAFS SLC 40
7             8  2014-07-14       Falcon 9  1316.000000   LEO  CCAFS SLC 40
8             9  2014-08-05       Falcon 9  4535.000000   GTO  CCAFS SLC 40
9            10  2014-09-07       Falcon 9  4428.000000   GTO  CCAFS SLC 40

       Outcome  Flights  GridFins  Reused   Legs LandingPad  Block  \
0    None None        1     False   False  False        NaN    1.0
1    None None        1     False   False  False        NaN    1.0
2    None None        1     False   False  False        NaN    1.0
3   False Ocean       1     False   False  False        NaN    1.0
4    None None        1     False   False  False        NaN    1.0
5    None None        1     False   False  False        NaN    1.0
6    True Ocean       1     False   False   True        NaN    1.0
7    True Ocean       1     False   False   True        NaN    1.0
8    None None        1     False   False  False        NaN    1.0
9    None None        1     False   False  False        NaN    1.0

   ReusedCount Serial   Longitude   Latitude
0            0  B0003  -80.577366  28.561857
1            0  B0005  -80.577366  28.561857
2            0  B0007  -80.577366  28.561857
3            0  B1003 -120.610829  34.632093
4            0  B1004  -80.577366  28.561857
5            0  B1005  -80.577366  28.561857
6            0  B1006  -80.577366  28.561857
7            0  B1007  -80.577366  28.561857
8            0  B1008  -80.577366  28.561857
9            0  B1011  -80.577366  28.561857
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```



DATA WRANGLING

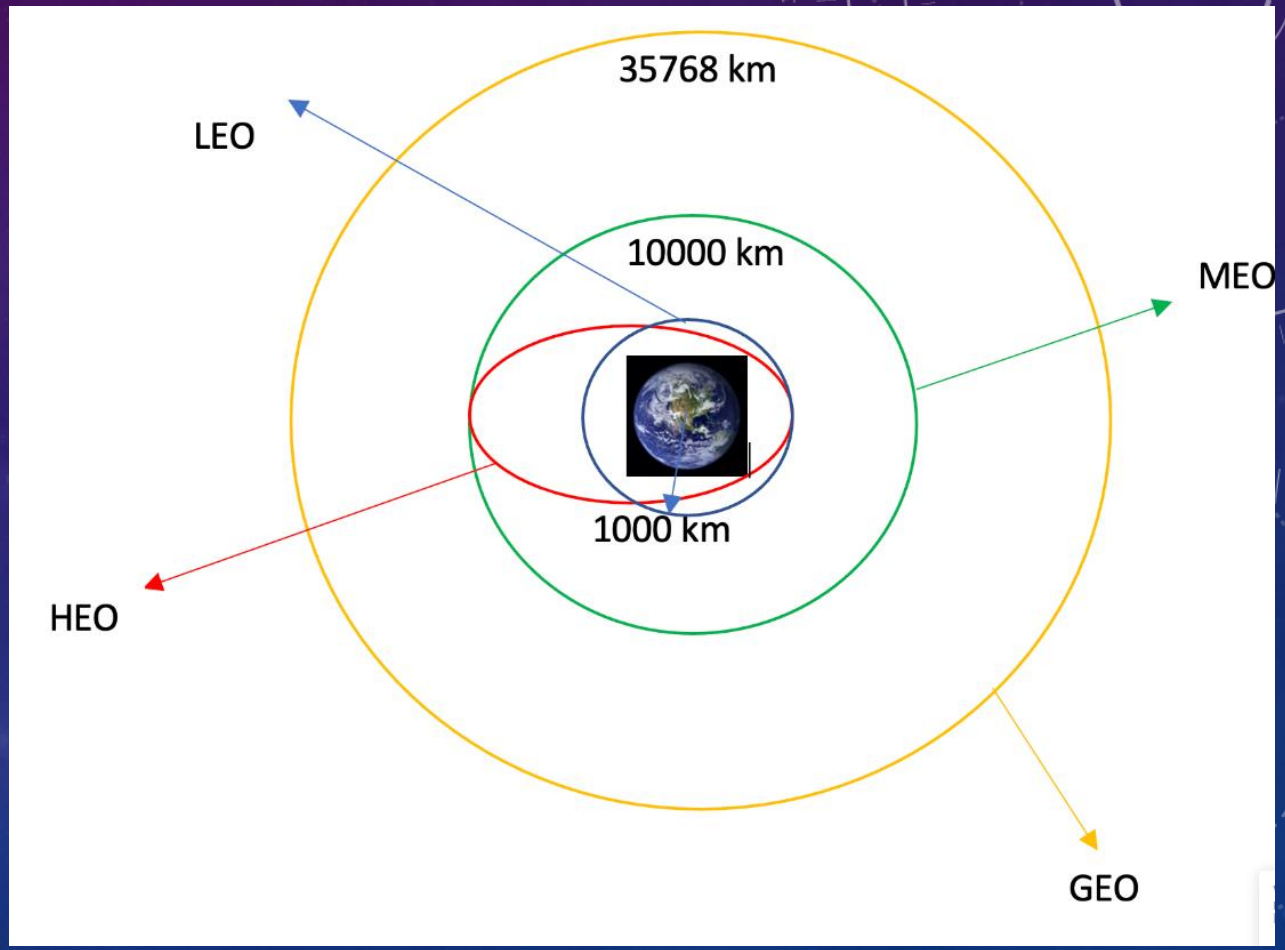
```
GTO     27
ISS     21
VLEO    14
PO       9
LEO      7
SSO      5
MEO      3
ES-L1    1
HEO      1
SO       1
GEO      1
Name: Orbit, dtype: int64
```

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.

```
# Landing_outcomes = values on Outcome column
import pandas as pd

# Read the CSV file into a DataFrame
df = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_1.csv")

# Use value_counts() on the Outcome column to determine the number of landing outcomes
landing_outcomes = df['Outcome'].value_counts()

# Print the landing outcomes
print(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
```



DATA WRANGLING

```
# Print the landing_class list
print(landing_class)
```

```
[1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

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

[11]:
```
df['Class']=landing_class
df[['Class']].head(8)
```

[11]:

	Class
0	1
1	1
2	1
3	0
4	1
5	1
6	1
7	1

[12]:
```
df.head(5)
```

[12]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.561857	1
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561857	1
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577366	28.561857	1
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003	-120.610829	34.632093	0
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004	-80.577366	28.561857	1

We can use the following line of code to determine the success rate:

[13]:
```
df["Class"].mean()
```

[13]: 0.9666666666666667

We can now export it to a CSV for the next section,but to make the answers consistent, in the next lab we will provide data in a pre-selected date range.

```
df.to_csv("dataset_part_2.csv", index=False)
```

EDA AND INTERACTIVE VISUAL ANALYTICS

Exploratory Data Analysis

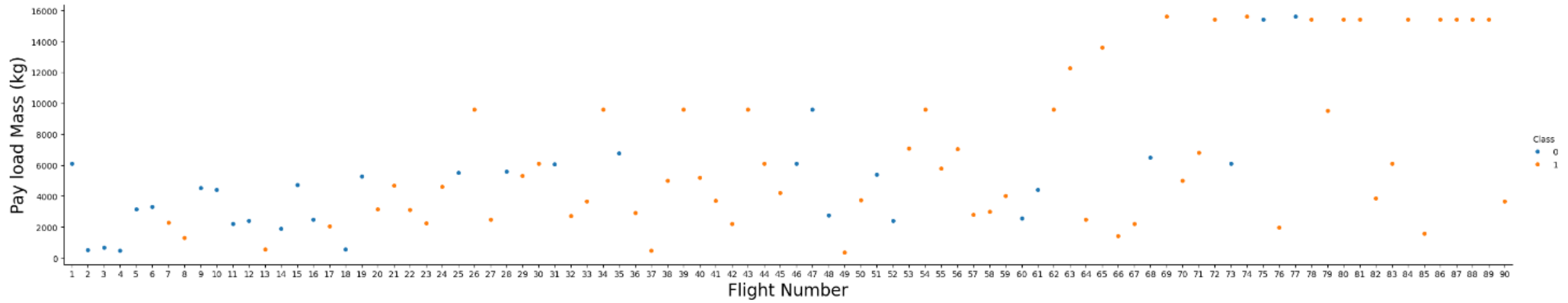
First, let's read the SpaceX dataset into a Pandas dataframe and print its summary

[5]:
```
from js import fetch
import io

URL = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_2.csv"
resp = await fetch(URL)
dataset_part_2_csv = io.BytesIO((await resp.arrayBuffer()).to_py())
df=pd.read_csv(dataset_part_2_csv)
df.head(5)
```

[5]:

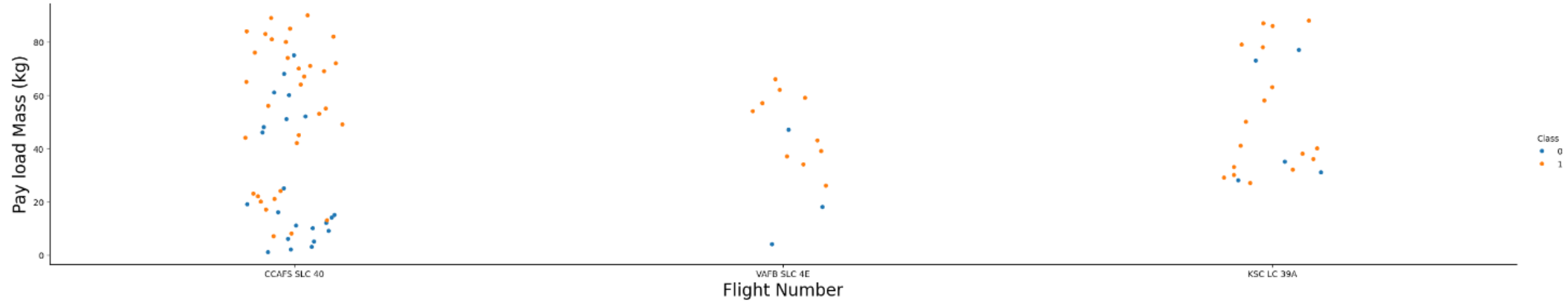
	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.561857	0
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561857	0
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577366	28.561857	0
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003	-120.610829	34.632093	0
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004	-80.577366	28.561857	0

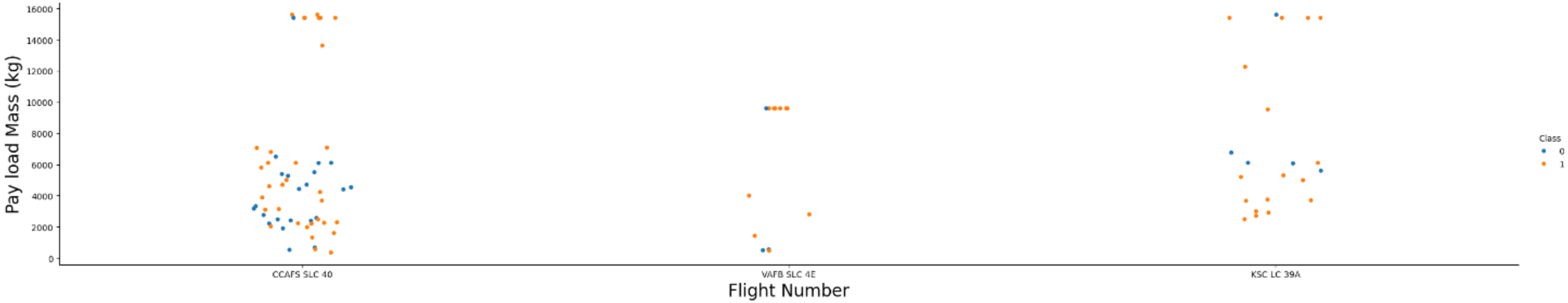
[6]:
```
sns.catplot(y="PayloadMass", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```



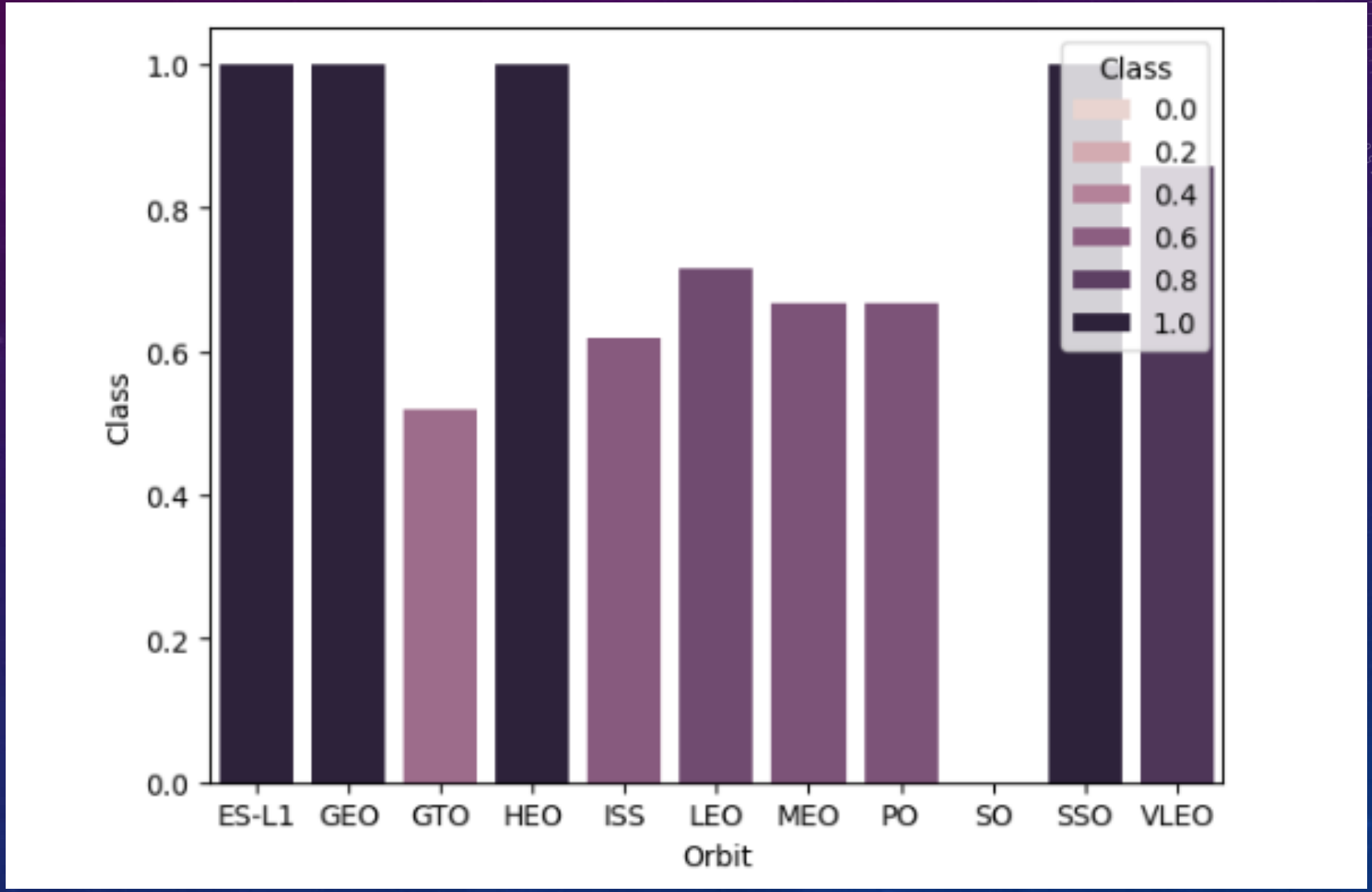
EDA AND INTERACTIVE VISUAL ANALYTICS

```
[8]: # 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 value
sns.catplot(y="FlightNumber",x="LaunchSite",hue='Class',data=df, aspect=5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```

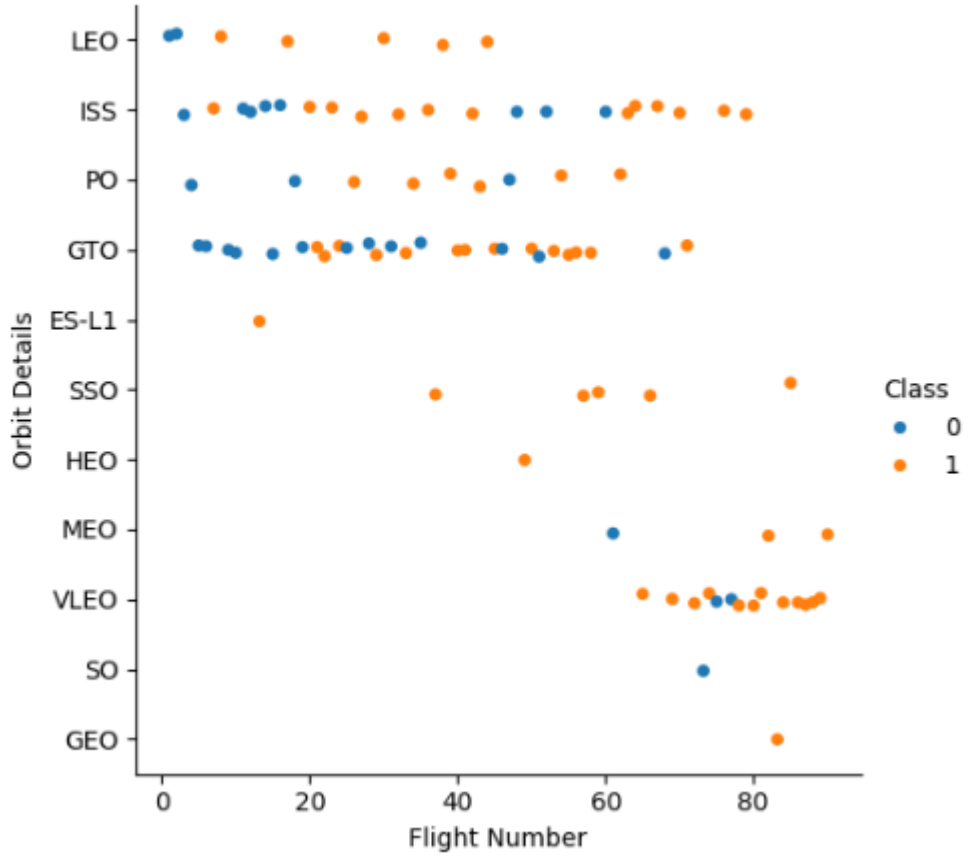




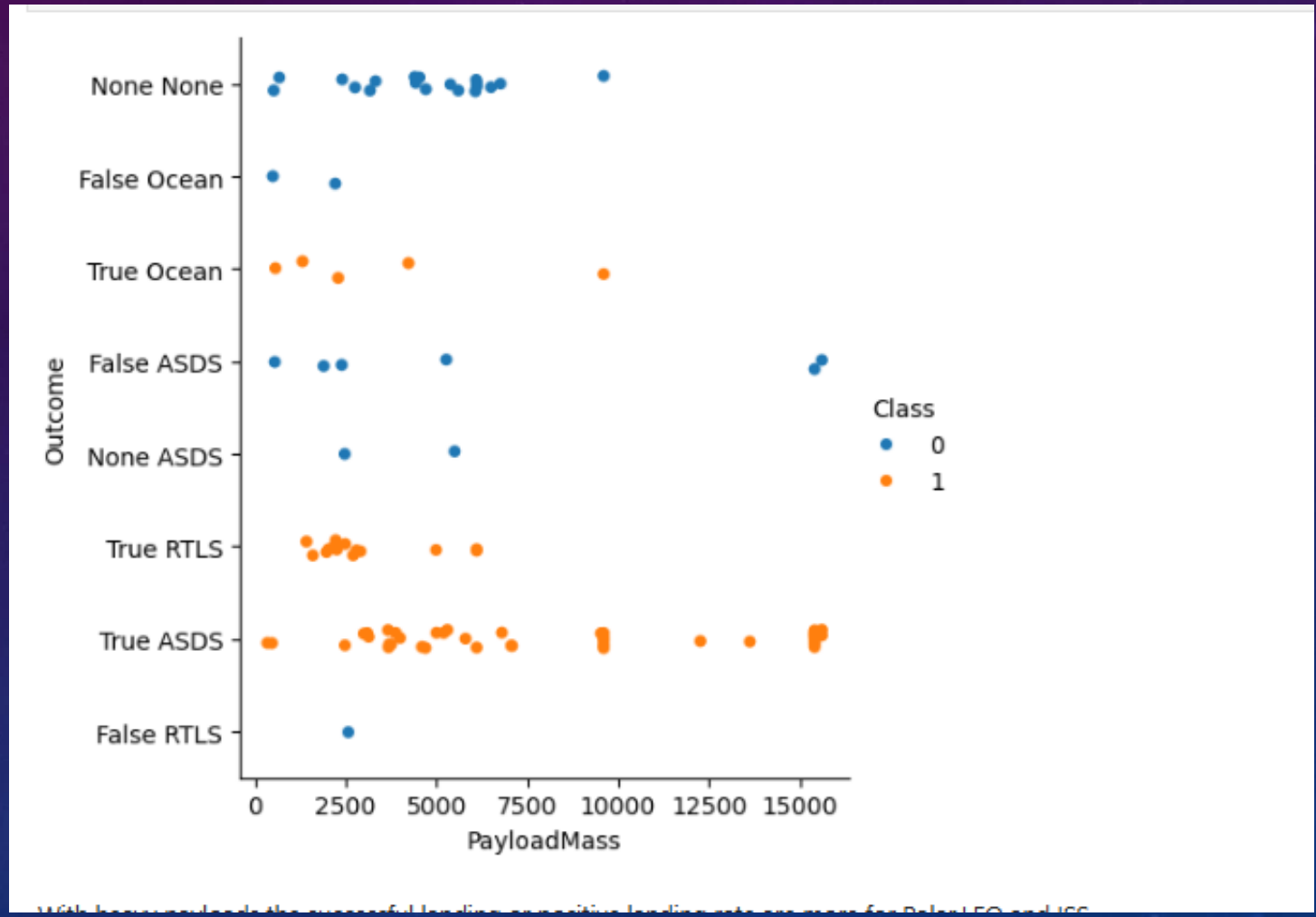
EDA AND INTERACTIVE VISUAL ANALYTICS



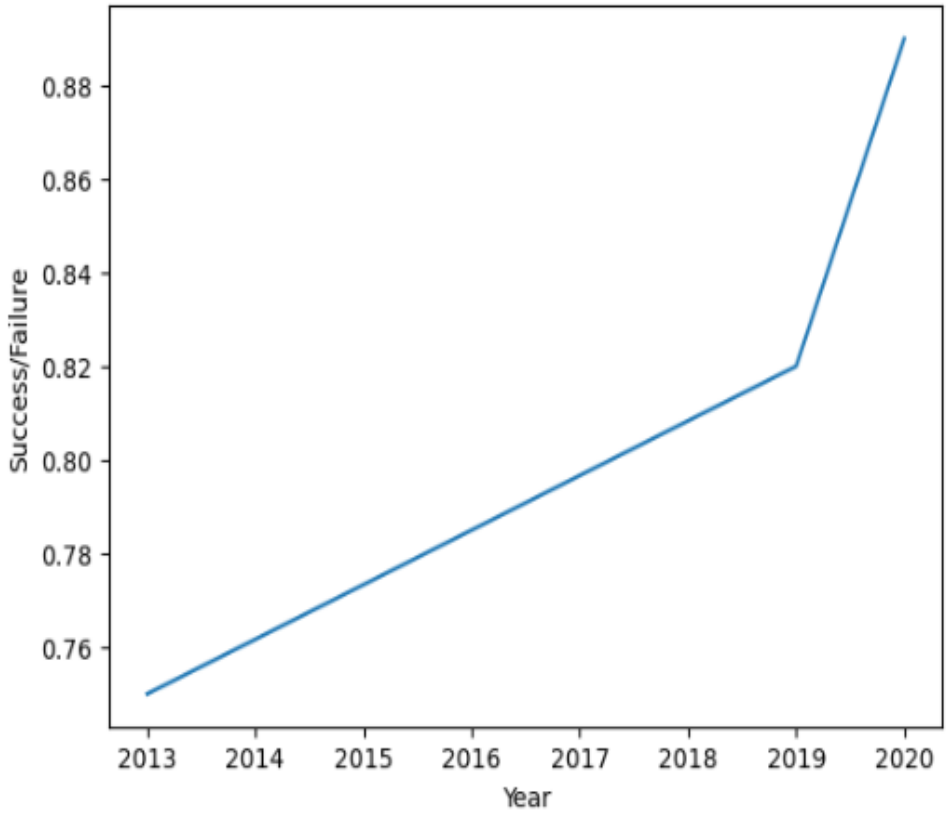
EDA AND INTERACTIVE VISUAL ANALYTICS



EDA AND INTERACTIVE VISUAL ANALYTICS



EDA AND INTERACTIVE VISUAL ANALYTICS



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

By now, you should obtain some preliminary insights about how each important variable would affect the success rate, we will select the features that will be used in success prediction in the future module.

```
features = df[['FlightNumber', 'PayloadMass', 'Orbit', 'LaunchSite', 'Flights', 'GridFins', 'Reused', 'Legs', 'LandingPad', 'Block', 'ReusedCount', 'Serial']]
features.head()
```

	FlightNumber	PayloadMass	Orbit	LaunchSite	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial
0	1	6104.959412	LEO	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B0003
1	2	525.000000	LEO	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B0005
2	3	677.000000	ISS	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B0007
3	4	500.000000	PO	VAFB SLC 4E	1	False	False	False	NaN	1.0	0	B1003
4	5	3170.000000	GTO	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B1004

EDA AND INTERACTIVE VISUAL ANALYTICS

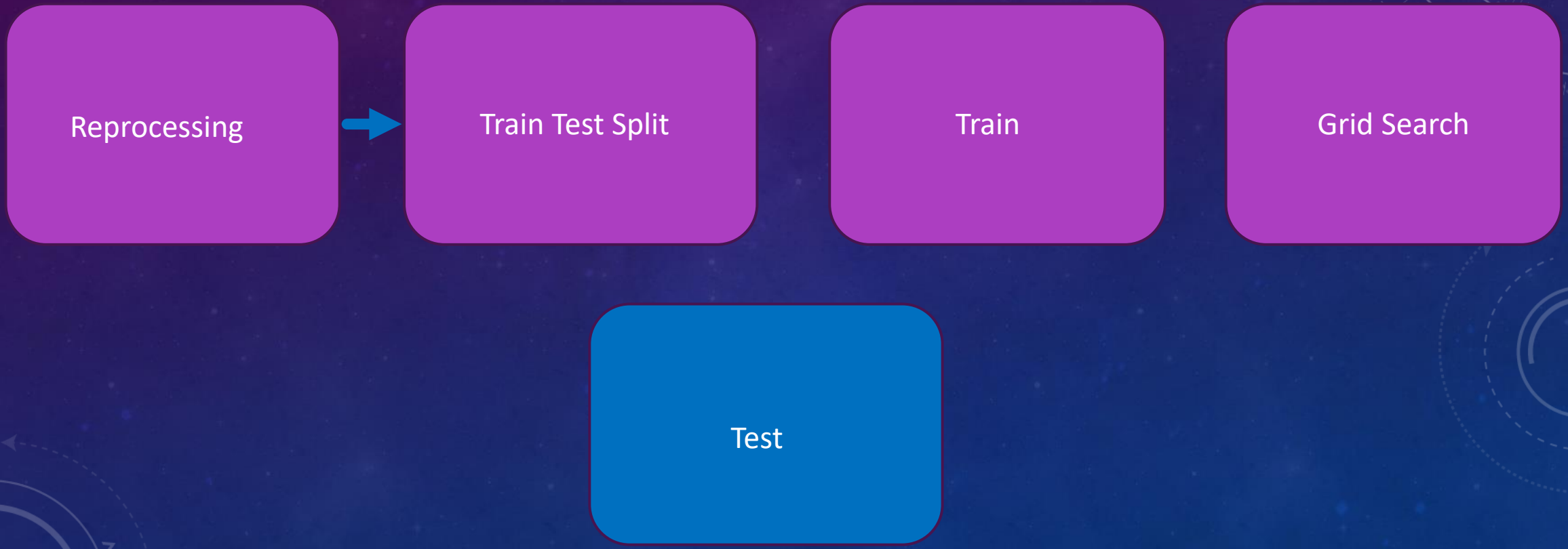
	FlightNumber	PayloadMass	Flights	GridFins	Reused	Legs	Block	ReusedCount	Orbit_ES-L1	Orbit_GEO	...	Serial_B1048	Serial_B1049	Serial_B1050	Serial_B1051	Serial_B1054	Serial_B1056	Serial_B1058	Serial_B1059	Serial_B1060	Serial_B
0	1	6104.959412	1	False	False	False	1.0	0	False	False	...	False	False	False	False	False	False	False	False	False	
1	2	525.000000	1	False	False	False	1.0	0	False	False	...	False	False	False	False	False	False	False	False	False	
2	3	677.000000	1	False	False	False	1.0	0	False	False	...	False	False	False	False	False	False	False	False	False	
3	4	500.000000	1	False	False	False	1.0	0	False	False	...	False	False	False	False	False	False	False	False	False	
4	5	3170.000000	1	False	False	False	1.0	0	False	False	...	False	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
85	86	15400.000000	2	True	True	True	5.0	2	False	False	...	False	False	False	False	False	False	False	False	True	
86	87	15400.000000	3	True	True	True	5.0	2	False	False	...	False	False	False	False	False	False	True	False	False	
87	88	15400.000000	6	True	True	True	5.0	5	False	False	...	False	False	False	True	False	False	False	False	False	
88	89	15400.000000	3	True	True	True	5.0	2	False	False	...	False	False	False	False	False	False	False	False	True	
89	90	3681.000000	1	True	False	True	5.0	0	False	False	...	False	False	False	False	False	False	False	False	False	

90 rows × 80 columns

PREDICTIVE ANALYSIS METHODOLOGY

Build a machine-learning pipeline

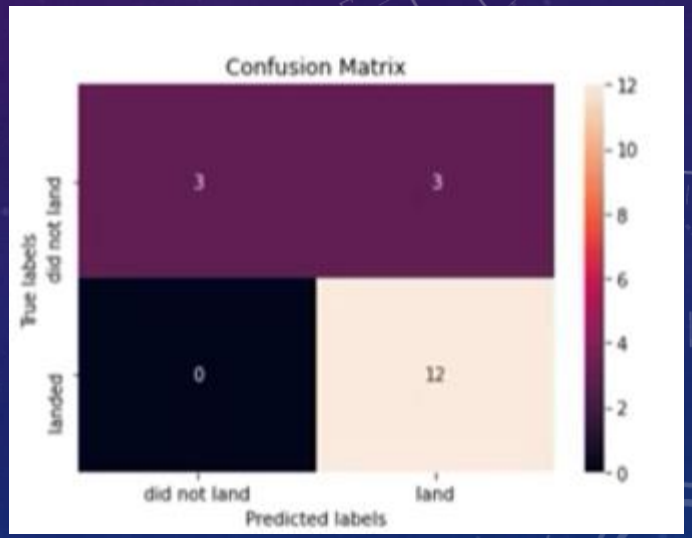
- Predict whether the first stage of Falcon 9 will land successfully



DETERMINE MODEL WITH BEST ACCURACY

Test

Model
Logistic Regression
Support Vector Machine
Decision Tree Classifier
K-nearest Neighbors



EDA WITH VISUALIZATION SQL

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

```
%sql SELECT * \
    FROM SPACEXTBL \
    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

EDA WITH SQL RESULTS

Task 3

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

[15]:
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) \
    FROM SPACEXTBL \
    WHERE CUSTOMER = 'NASA (CRS)';
```

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

[15]:

SUM(PAYLOAD_MASS__KG_)
45596

Task 4

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

[16]:
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) \
    FROM SPACEXTBL \
    WHERE BOOSTER_VERSION = 'F9 v1.1';
```

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

[16]:

AVG(PAYLOAD_MASS__KG_)
2928.4

EDA WITH SQL RESULTS

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

Hint:Use min function

```
%sql SELECT MIN(DATE) \
FROM SPACEXTBL \
WHERE LANDING__OUTCOME = 'Success (ground pad)'
```

```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) no such column: LANDING__OUTCOME
[SQL: SELECT MIN(DATE) FROM SPACEXTBL WHERE LANDING__OUTCOME = 'Success (ground pad)']
(Background on this error at: https://sqlalche.me/e/20/e3q8)
```

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 PAYLOAD \
FROM SPACEXTBL \
WHERE LANDING__OUTCOME = 'Success (drone ship)' \
AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000;
```

```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) no such column: LANDING__OUTCOME
[SQL: SELECT PAYLOAD FROM SPACEXTBL WHERE LANDING__OUTCOME = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000;]
(Background on this error at: https://sqlalche.me/e/20/e3q8)
```

EDA WITH SQL RESULTS

Task 7

List the total number of successful and failure mission outcomes

```
%sql SELECT MISSION_OUTCOME, COUNT(*) as total_number \
FROM SPACEXTBL \
GROUP BY MISSION_OUTCOME;
```

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

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

EDA WITH SQL RESULTS

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

```
%sql SELECT BOOSTER_VERSION \
FROM SPACEXTBL \
WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

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

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

EDA WITH SQL RESULTS

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.

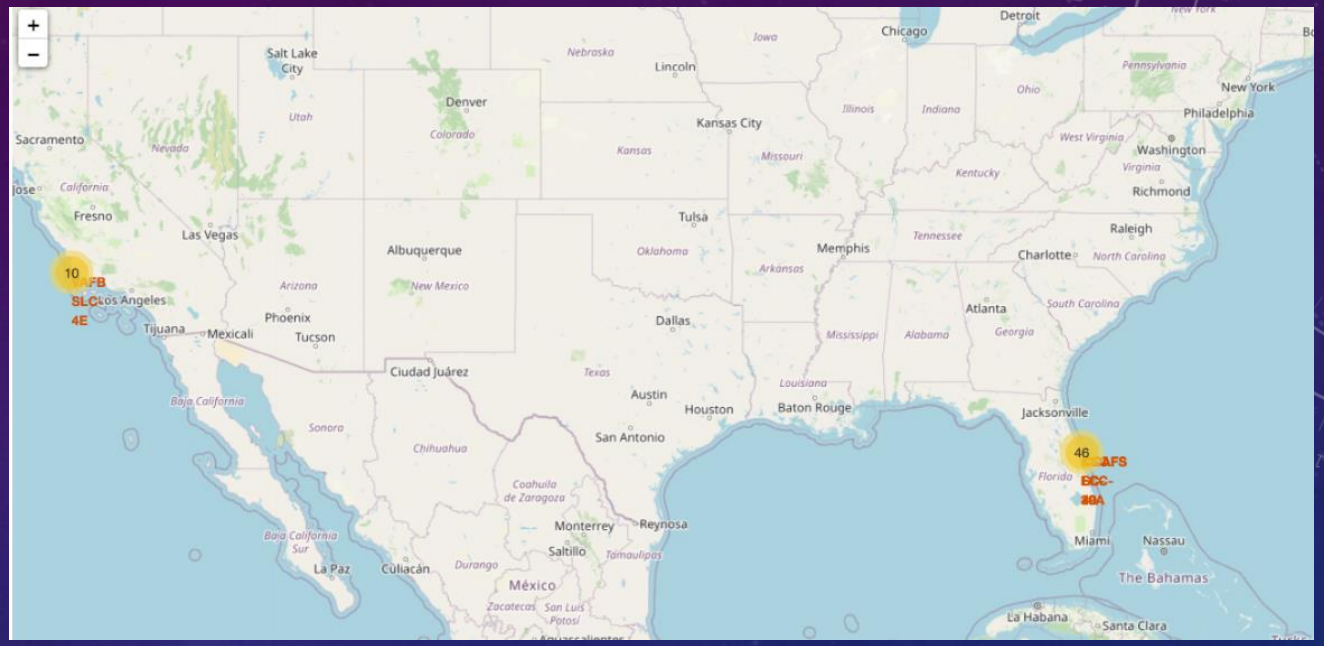
```
%%sql

SELECT "Landing _Outcome", COUNT(*) as count
FROM SPACEXTBL
WHERE Date BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY "Landing _Outcome", "Success", "Failure"
ORDER BY count DESC;
```

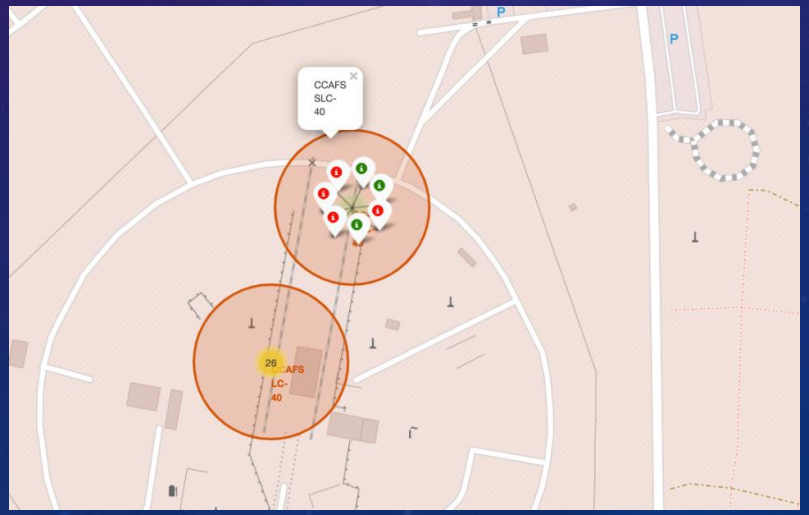
```
 * sqlite:///my_data1.db
Done.
```

"Landing _Outcome"	count
Landing _Outcome	31

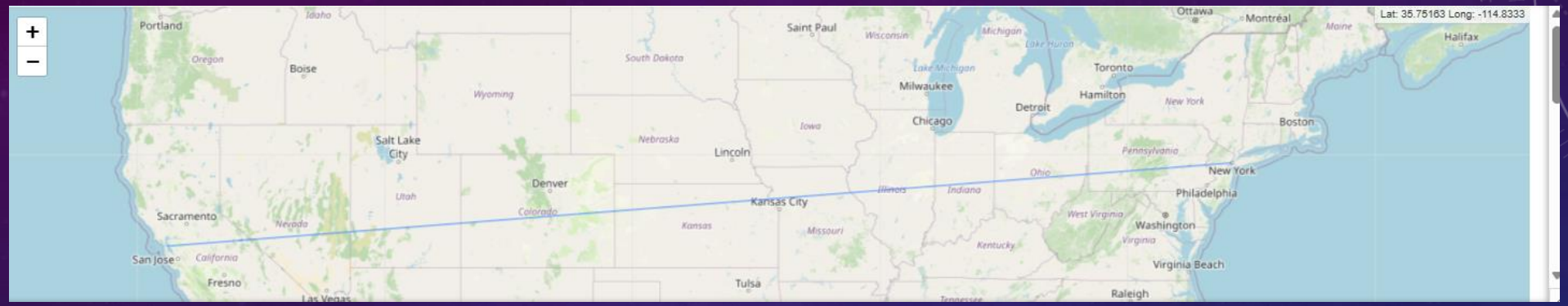
REQUIRED INTERACTIVE MAP WITH FOLIUM RESULTS

	Launch Site	Lat	Long	class
46	KSC LC-39A	28.573255	-80.646895	1
47	KSC LC-39A	28.573255	-80.646895	1
48	KSC LC-39A	28.573255	-80.646895	1
49	CCAFS SLC-40	28.563197	-80.576820	1
50	CCAFS SLC-40	28.563197	-80.576820	1
51	CCAFS SLC-40	28.563197	-80.576820	0
52	CCAFS SLC-40	28.563197	-80.576820	0
53	CCAFS SLC-40	28.563197	-80.576820	0
54	CCAFS SLC-40	28.563197	-80.576820	1
55	CCAFS SLC-40	28.563197	-80.576820	0





REQUIRED INTERACTIVE MAP WITH FOLIUM RESULTS



Your updated map with distance line should look like the following screenshot:

PREDICTIVE ANALYSIS (CLASSIFICATION)

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Clas
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.561857	
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561857	
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577366	28.561857	
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003	-120.610829	34.632093	
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004	-80.577366	28.561857	

PREDICTIVE ANALYSIS (CLASSIFICATION)

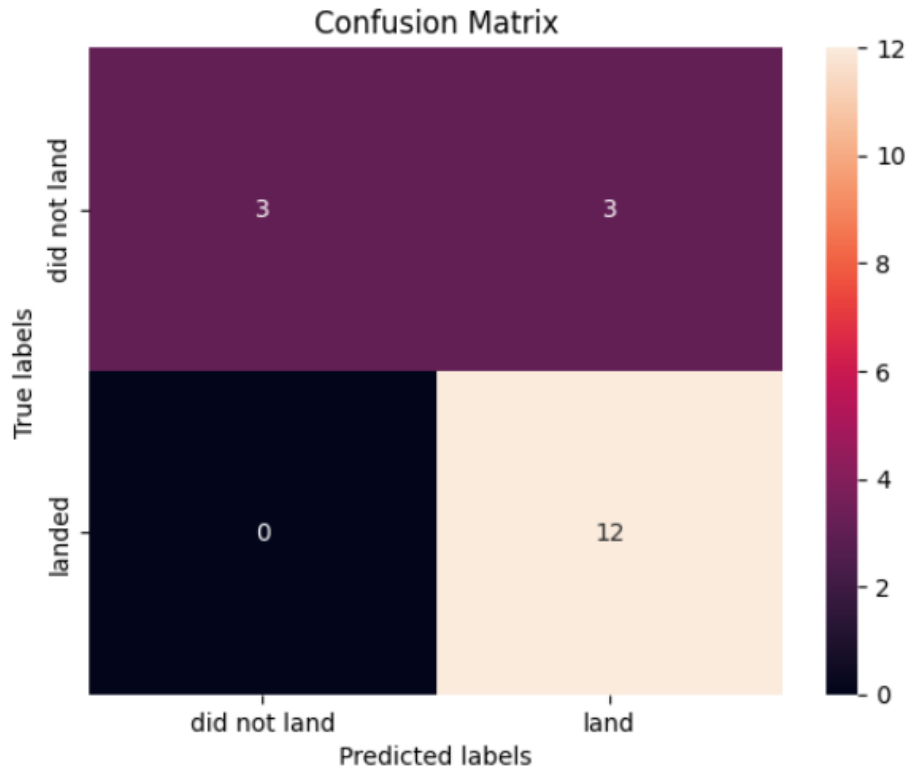
Calculate the accuracy on the test data using the method `score`:

[63]:
```
svm_cv.score(X_test, Y_test)
```

[63]:
```
0.8333333333333334
```

We can plot the confusion matrix

[64]:
```
yhat=svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



PREDICTIVE ANALYSIS (CLASSIFICATION)

Create a support vector machine object then create a `GridSearchCV` object `svm_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

[60]:
```
parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
              'C': np.logspace(-3, 3, 5),
              'gamma':np.logspace(-3, 3, 5)}
svm = SVC()
```

[61]:
```
svm_cv = GridSearchCV(svm, param_grid=parameters,scoring='accuracy', cv=10)
svm_cv.fit(X_train, Y_train)
svm_cv.best_params_
```

[61]:
```
{'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
```

[62]:
```
print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
print("accuracy :",svm_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
accuracy : 0.8482142857142856
```

PREDICTIVE ANALYSIS (CLASSIFICATION)

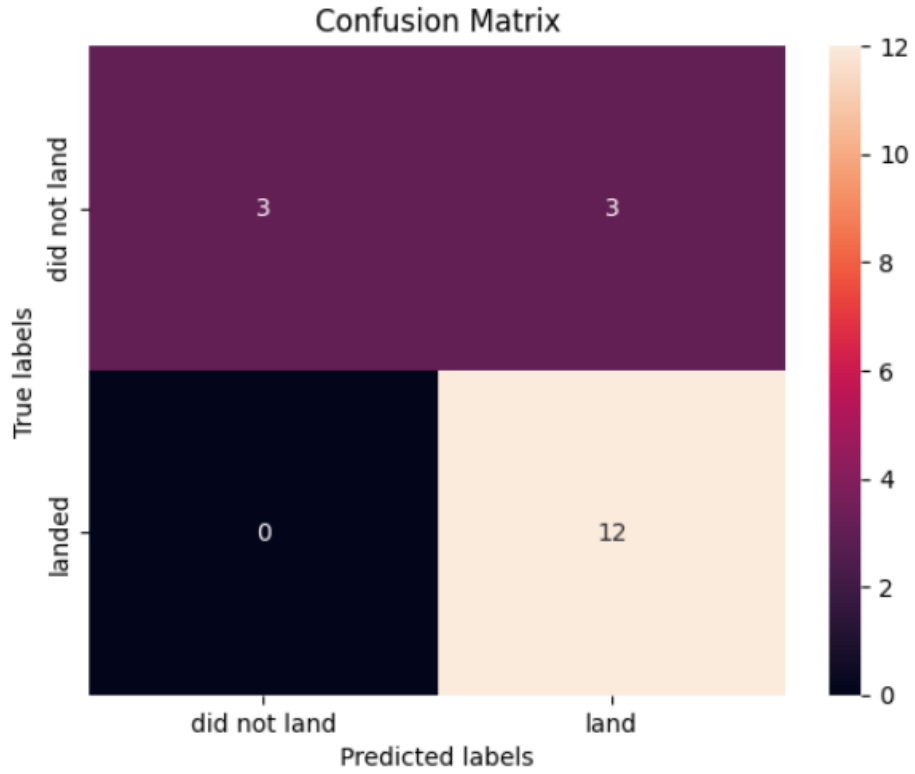
Calculate the accuracy on the test data using the method `score` :

[63]:
```
svm_cv.score(X_test, Y_test)
```

[63]:
```
0.8333333333333334
```

We can plot the confusion matrix

[64]:
```
yhat=svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



PREDICTIVE ANALYSIS (CLASSIFICATION)

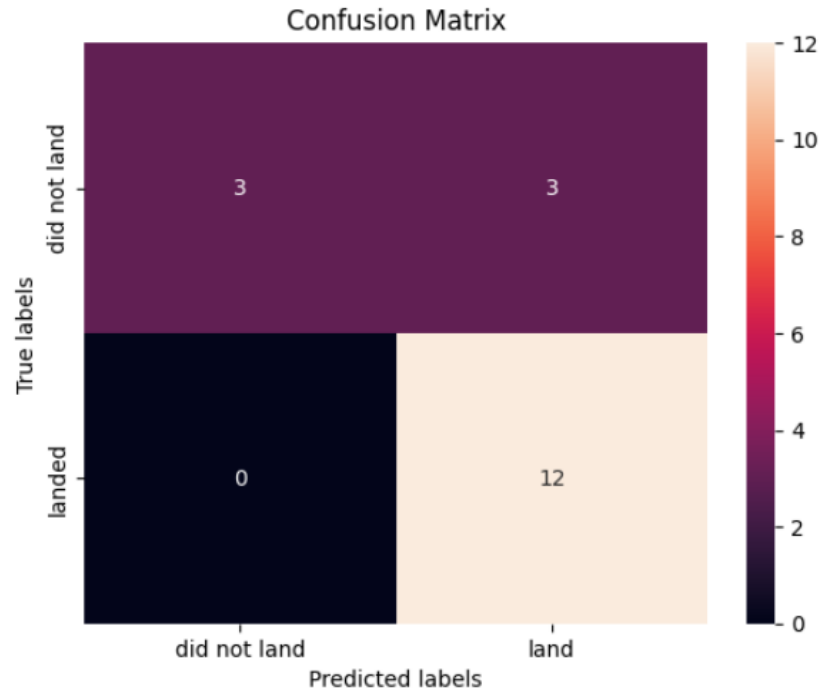
Calculate the accuracy of tree_cv on the test data using the method `score` :

[68]:
```
tree_cv.score(X_test, Y_test)
```

[68]:
```
0.8333333333333334
```

We can plot the confusion matrix

[69]:
```
yhat = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



PREDICTIVE ANALYSIS (CLASSIFICATION)

Create a k nearest neighbors object then create a `GridSearchCV` object `knn_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

```
parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
              'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
              'p': [1,2]}

KNN = KNeighborsClassifier()
```

```
knn_cv = GridSearchCV(KNN, param_grid=parameters,scoring='accuracy', cv=10)
knn_cv.fit(X_train, Y_train)
knn_cv.best_params_
```

```
{'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
```

```
print("tuned hpyerparameters :(best parameters) ",knn_cv.best_params_)
print("accuracy :",knn_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
accuracy : 0.8482142857142858
```

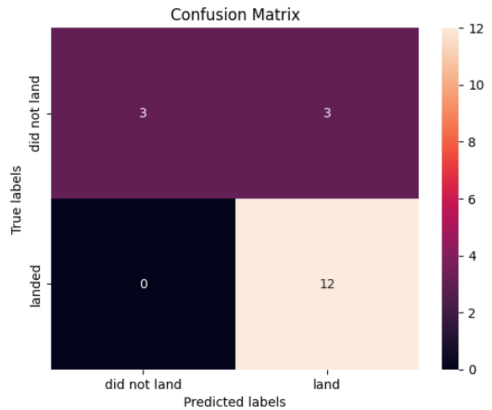
Calculate the accuracy of knn_cv on the test data using the method `score`:

```
knn_cv.score(X_test, Y_test)
```

```
0.8333333333333334
```

We can plot the confusion matrix

```
yhat = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



CONCLUSION

- Every method used is around the 0.83 number however, some numbers were 0.84
- Comparing models consistently across various studies proves difficult, as there is currently no tool available to assess models using identical performance metrics.

INNOVATIVE INSIGHTS

Performance Evaluation: Accuracy serves as a primary performance evaluation metric, indicating the percentage of correctly predicted instances by a model. It helps gauge the model's effectiveness in making precise decisions based on the available data.

Model Optimization: Improving accuracy involves optimizing the model through techniques like feature selection, hyperparameter tuning, and data preprocessing. Enhancing accuracy leads to more reliable predictions and better outcomes in real-world scenarios.