IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Chaitanya Jindal
August 2023

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion

Executive Summary

- Summary of methodologies
  - Data Collection via API, Web Scraping
  - Exploratory Data Analysis (EDA) with Data Visualization
  - EDA with SQL
  - Interactive Map with Folium
  - Predictive Analysis
- Summary of all results
  - Exploratory Data Analysis results
  - Interactive maps and dashboard
  - Predictive results

Introduction

- Project background and context
  - The objective of this project is to forecast the successful landing of the Falcon 9 first stage. According to SpaceX's website, the Falcon 9 rocket launch costs 62 million dollars, whereas other providers charge over 165 million dollars per launch because they don't reuse the first stage. By assessing whether the stage will land successfully, we can ascertain the launch cost, which may prove valuable to other companies seeking to compete with SpaceX in the rocket launch market.
- Problems you want to find answers
  - The main characteristics of a successful or failed landing involve distinct attributes that set them apart. The effects of the relationships among rocket variables play a significant role in determining the outcome, with each variable contributing to either success or failure. To achieve the best landing success rate, SpaceX must meet specific conditions, including thorough testing and validation, utilization of advanced technology, continuous improvement through data analysis, a skilled team, and learning from past failures to implement necessary improvements in their landing procedures.

Section 1

Methodology

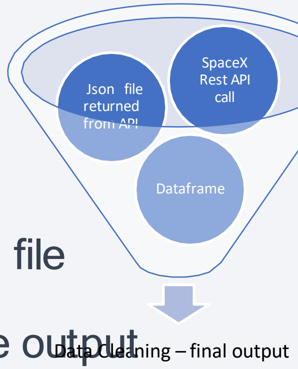
Methodology

Executive Summary

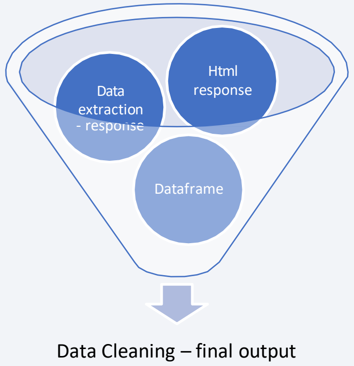
- Data collection methodology:
    - The data was collected via API (SpaceX Rest API) and via Web Scraping from Wikipedia
- Perform data wrangling
    - Dropping unnecessary columns
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
    - How to build, tune, evaluate classification models

Data Collection

- Describe how data sets were collected.
  - The data was collected via Rest API and Web Scraping, details below:
    - Rest API
      - 1st SpaceX Rest API call
      - 2nd API returns a json file
      - 3rd Construction of a dataframe from the json file
      - 4th Performing a data cleaning and export the output



    - Web Scraping
      - 1st From html response from Wikipedia
      - 2nd Data extraction using beasutifulSoup python lib
      - 3rd Construction of a dataframe
      - 4th Export the output



Data Collection - SpaceX API

SpaceX Rest API call

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

API response

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

Dataframe

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

```
data = pd.DataFrame.from_dict(launch_dict)
```

Export the output - file

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

- GitHub URL of the completed SpaceX API calls notebook [here]

Data Collection - Scraping

html response

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

Data extraction using BeasutifulSoup

```
soup = BeautifulSoup(response.text, "html5lib")
```

Dataframe

```
launch_dict= dict.fromkeys(column_names)

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

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

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

Export the output - file

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

- GitHub URL of the completed web scraping notebook here

Data Wrangling

The dataset contains instances of both successful and unsuccessful booster landings. A successful mission is denoted by True Ocean, True RTLS, and True ASDS values, while a failed mission is indicated by False Ocean, False RTLS, and False ASDS values. To convert the string variables into categorical variables, we assign the value of 1 to indicate a successful mission and 0 to represent a failed mission.

- **Calculate the number of launches on each site**

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

- **Calculate the number and occurrence of each orbit**

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

- **Calculate the number and occurrence of mission outcome per orbit type**

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

- **Final output – stored**

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

GitHub URL of your completed

data wrangling related notebooks [here]

EDA with Data Visualization

- We conducted an exploratory data analysis and feature engineering using Pandas and Matplotlib, presenting the results in charts.
  - Scatter plot charts
    - Flight Number vs Payload Max
    - Flight Number vs Launch Site
    - Payload vs Launch Site
    - Orbit vs Flight Number
    - Playload vs Orbit Type
    - Orbit vs Payload Mass
  - Bar charts
    - Success rate vs Orbit

GitHub URL of your completed
data wrangling related notebooks, [here]

EDA with SQL

- We conducted SQL queries to collect and analyze data from the dataset, and the results are as follows:

1. Displaying the names of the unique launch sites in the space mission.
2. Display 5 records where launch sites begin with the string 'CCA'.
3. Display the total payload mass carried by boosters launched by NASA (CRS).
4. Display the average payload mass carried by the booster version F9 v1.1.
5. List the date when the first successful landing outcome on a ground pad was achieved.
6. List the names of the boosters that achieved success on a drone ship and carried a payload mass greater than 4000 but less than 6000.
7. List the total number of successful and failure mission outcomes.
8. List the names of the booster_versions that carried the maximum payload mass.
9. List the records displaying the month names, failure landing outcomes on a drone ship, booster versions, and launch site for the months in the year 2015.
10. Rank the count of successful landing outcomes between the dates 04-06-2010 and 20-03-2017 in descending order.

GitHub URL of your completed
data wrangling related notebooks here

Build an Interactive Map with Folium

The map objects used:

1. A red circle placed at the coordinates of the NASA Johnson Space Center, labeled with its name (folium.Circle, folium.map).
2. Red circles at the coordinates of each launch site, labeled with the launch site names (folium.Circle, folium.map.Marker, folium.features.DivIcon).
3. Grouping of points in clusters to display multiple and different information for the same coordinates (folium.plugins).
4. Markers indicating successful landings in green and unsuccessful landings in red (folium.map.Marker, folium).
5. Markers showing the distance between launch sites and key locations such as railways, highways, coastways, and cities, with lines drawn between them (folium.map.Marker, folium.PolyLine, folium.features.DivIcon).
6. These objects have been created to gain a better understanding of the problem and the data, providing an easy way to visualize all launch sites, their surroundings, and the number of successful and unsuccessful landings.

GitHub URL of your completed
data wrangling related notebooks here

Predictive Analysis (Classification)

In the data preparation phase, the dataset is loaded and then normalized to ensure consistency and better performance during modeling. The data is then split into training and test sets for model training and evaluation. For model preparation, various machine learning algorithms are selected, and their parameters are set using GridSearchCV. The selected models are trained with the training dataset, and their hyperparameters are evaluated to determine the best configuration for each model. The accuracy of each model is computed using the test dataset, and Confusion Matrix plots are generated to visualize the model's performance. Model comparison is done based on their accuracy, and the model with the highest accuracy is chosen as the final selection. For detailed results, refer to the Notebook.

GitHub URL of your completed
data wrangling related notebooks here

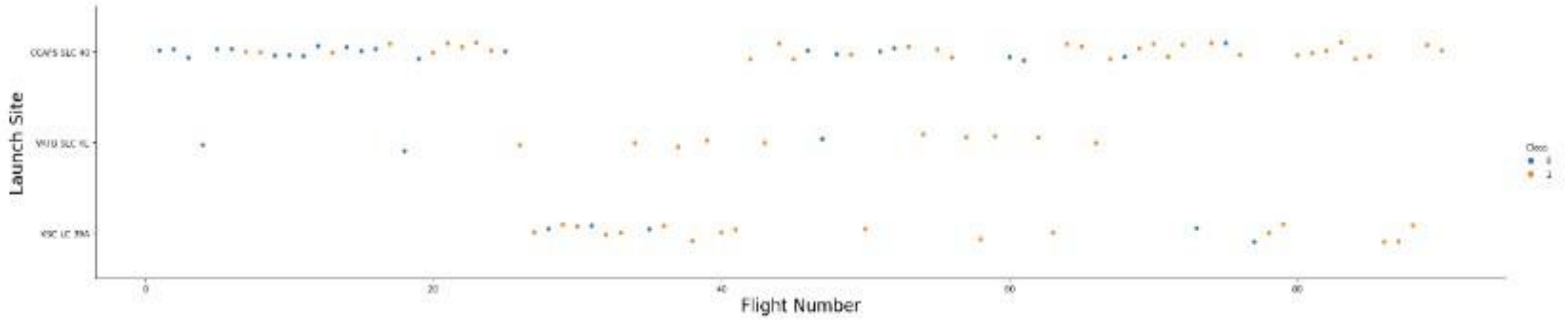
Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

In [5]:

```
# 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 valu
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()
```

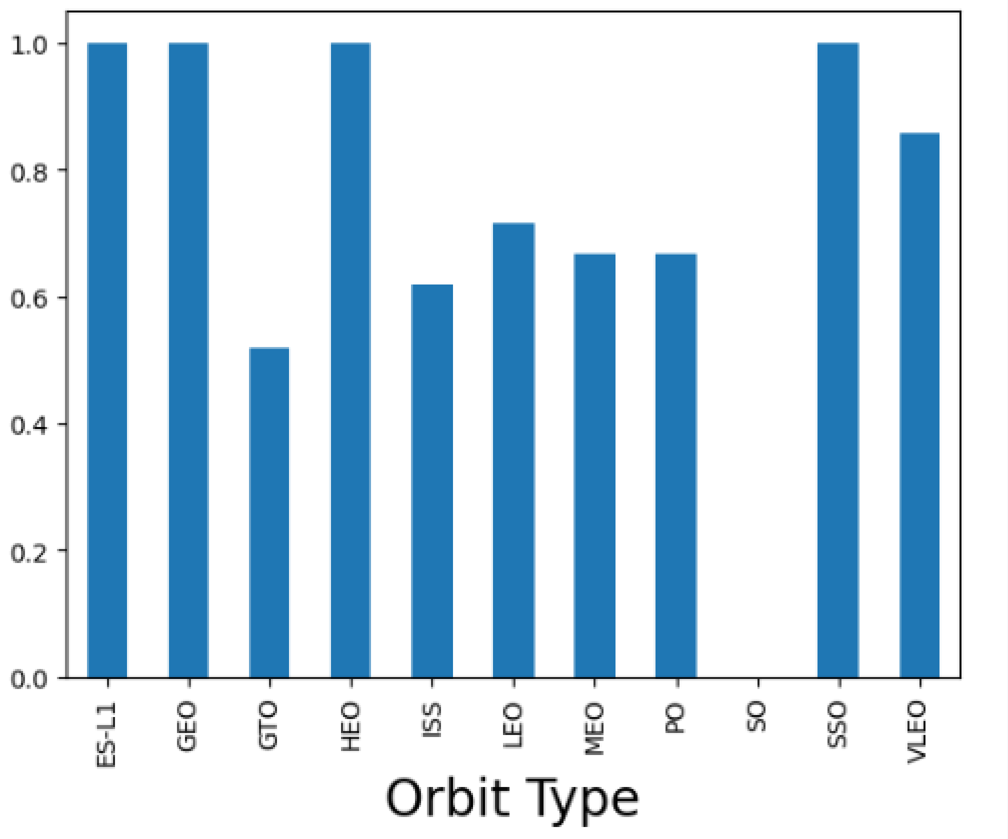


Payload vs. Launch Site

```
### TASK 2: Visualize the relationship between Payload and Launch Site
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()
```

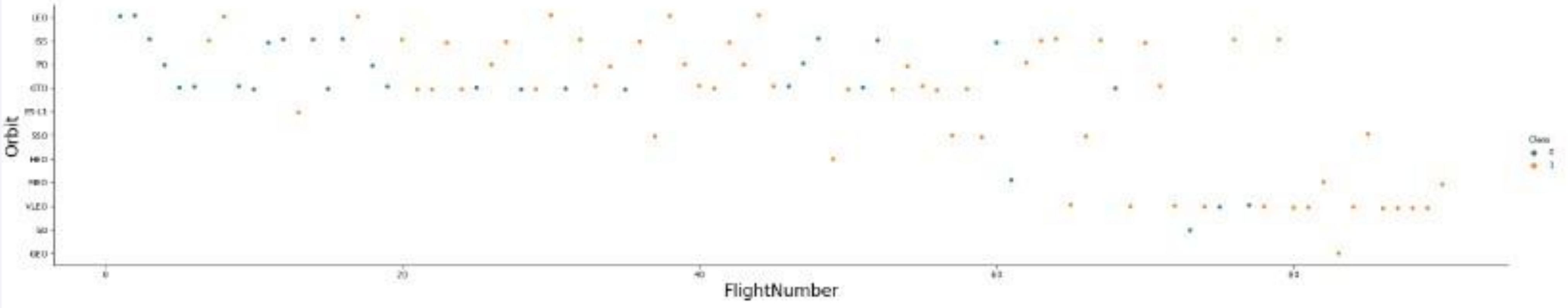
Success Rate vs. Orbit Type

```
# HINT use groupby method on Orbit column and get the mean of Class column
df.groupby("Orbit").mean()['Class'].plot(kind='bar')
plt.xlabel("Orbit Type",fontsize=20)
plt.ylabel("Success Rate",fontsize=20)
plt.show()
```



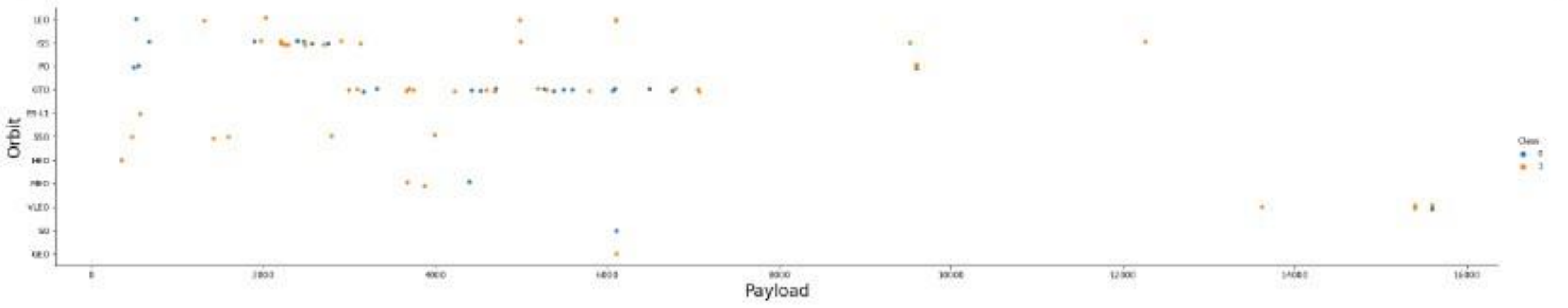
Flight Number vs. Orbit Type

```
# Plot a scatter point chart with x axis to be FlightNumber 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("FlightNumber",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



Payload vs. 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",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



All Launch Site Names

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

In [12]:
```
%%sql
SELECT DISTINCT LAUNCH_SITE
FROM SPACEXTBL;
```

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

Out[12]:

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'

In [13]:

```
%%sql
SELECT LAUNCH_SITE
FROM SPACEXTBL
WHERE LAUNCH_SITE LIKE 'CCA%'
LIMIT 5;
```

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

Out[13]:

Launch_Site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Total Payload Mass

Task 3

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

In [14]:

```
%%sql
SELECT SUM(PAYLOAD_MASS__KG_)
FROM SPACEXTBL
WHERE Customer = 'NASA (CRS)';
```

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

Out[14]:

SUM(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

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

In [15]:

```
%%sql
SELECT AVG(PAYLOAD_MASS__KG_)
FROM SPACEXTBL
WHERE Booster_Version LIKE 'F9 v1.0%';
```

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

Out[15]:

AVG(PAYLOAD_MASS__KG_)
340.4

First Successful Ground Landing Date

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

Hint:Use min function

In [19]:
```
%%sql
SELECT MIN(Date)
FROM SPACEXTBL
WHERE Landing_Outcome = 'Success (ground pad)';
```

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

Out[19]:

MIN(Date)
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

In [22]:

```
%%sql
SELECT BOOSTER_VERSION
FROM SPACEXTBL
WHERE LANDING_OUTCOME = 'Success (drone ship)'
    AND 4000 < PAYLOAD_MASS__KG_ < 6000;
```

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

Out[22]:

Booster_Version
F9 FT B1021.1
F9 FT B1022
F9 FT B1023.1
F9 FT B1026
F9 FT B1029.1
F9 FT B1021.2
F9 FT B1029.2
F9 FT B1036.1
F9 FT B1038.1
F9 B4 B1041.1
F9 FT B1031.2
F9 B4 B1042.1
F9 B4 B1045.1
F9 B5 B1046.1

Total Number of Successful and Failure Mission Outcomes

List the total number of successful and failure mission outcomes

In [23]:
```
%%sql
SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TOTAL_NUMBER
FROM SPACEXTBL
GROUP BY MISSION_OUTCOME;
```

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

Out[23]:

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

Boosters Carried Maximum Payload

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

In [24]:

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

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

Out[24]:

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

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.

```
%%sql
SELECT LANDING_OUTCOME, BOOSTER_VERSION, LAUNCH_SITE, DATE
FROM SPACEXTBL
WHERE Landing_Outcome = 'Failure (drone ship)'
AND DATE like '2015%';
```

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

Landing_Outcome	Booster_Version	Launch_Site	Date
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40	2015-10-01
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40	2015-04-14

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(LANDING_OUTCOME) AS TOTAL_NUMBER
FROM SPACEXTBL
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY LANDING_OUTCOME
ORDER BY TOTAL_NUMBER DESC
```

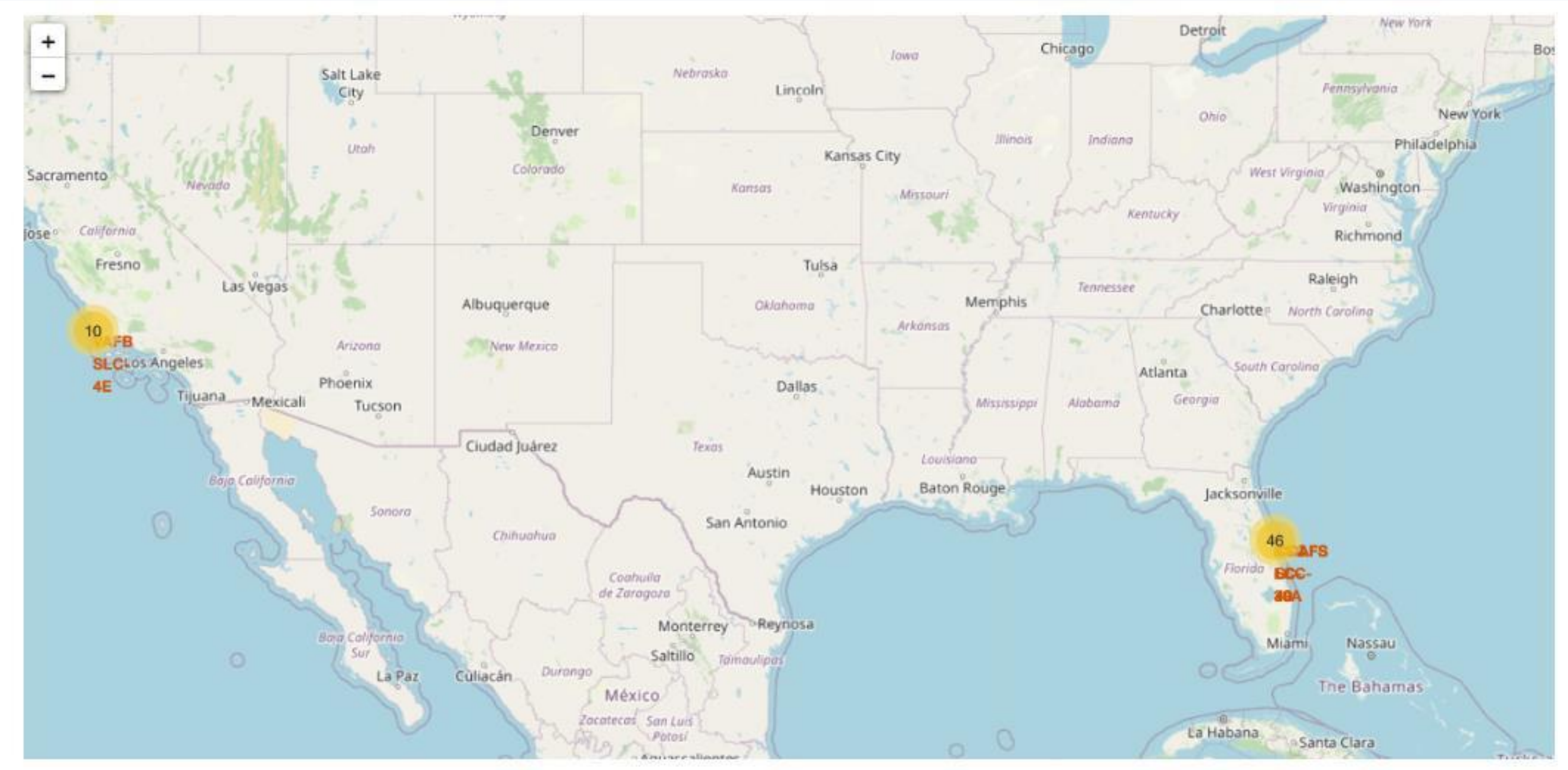
```
 * sqlite:///my_data1.db
Done.
```

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

Section 3

Launch Sites Proximities Analysis

Folium map- launch sites location



We can see they are located along the coast.

Folium map- launch sites location - markers

Folium Map - Distances from railways

Section 5

Predictive Analysis (Classification)

Classification Accuracy

```
▸ GridSearchCV
▸ estimator: LogisticRegression
  ▸ LogisticRegression
```

We output the `GridSearchCV` object for logistic regression. We display th

```
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_
print("accuracy :",logreg_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty':
accuracy : 0.8464285714285713
```

TASK 5

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

```
print("test set accuracy :",logreg_cv.score(X_test, Y_test))
```

```
test set accuracy : 0.8333333333333334
```

```
▸ GridSearchCV
▸ estimator: SVC
  ▸ SVC
```

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

```
tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277
accuracy : 0.8482142857142856
```

TASK 7

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

```
print("test set accuracy :",svm_cv.score(X_test, Y_test))
```

```
test set accuracy : 0.8333333333333334
```

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

```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_dep
accuracy : 0.8892857142857142
```

TASK 9

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

```
print("test set accuracy :",tree_cv.score(X_test, Y_test))
```

```
test set accuracy : 0.8333333333333334
```

```
▸ GridSearchCV
▸ estimator: KNeighborsClassifier
  ▸ KNeighborsClassifier
```

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

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

TASK 11

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

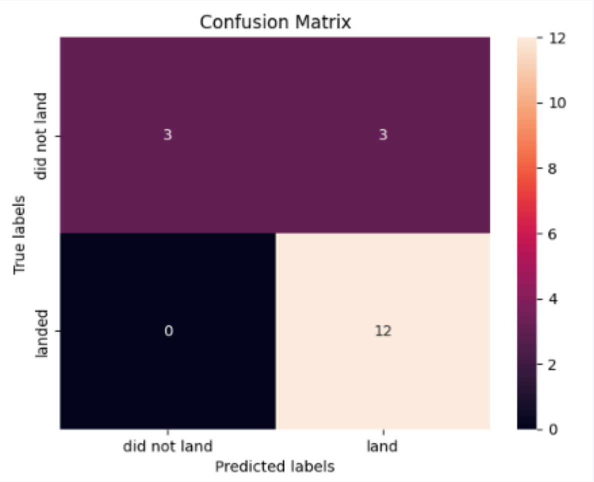
```
print("test set accuracy :",knn_cv.score(X_test, Y_test))
```

```
test set accuracy : 0.8333333333333334
```

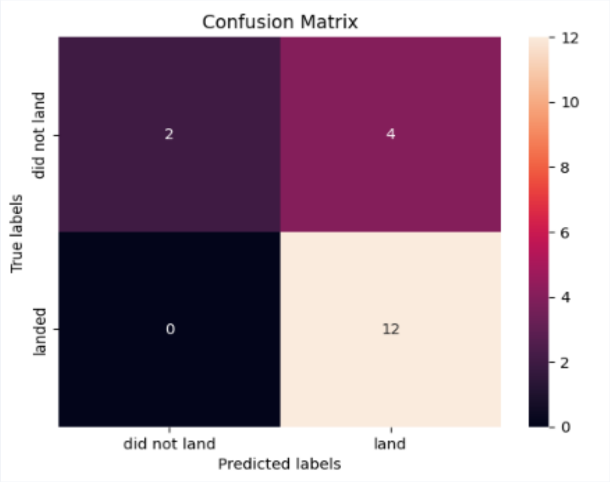
For accuracy test, all methods performed similar. We could get more test data to decide between them. But if we really need to choose one right now, we would take the decision tree.

Confusion Matrix

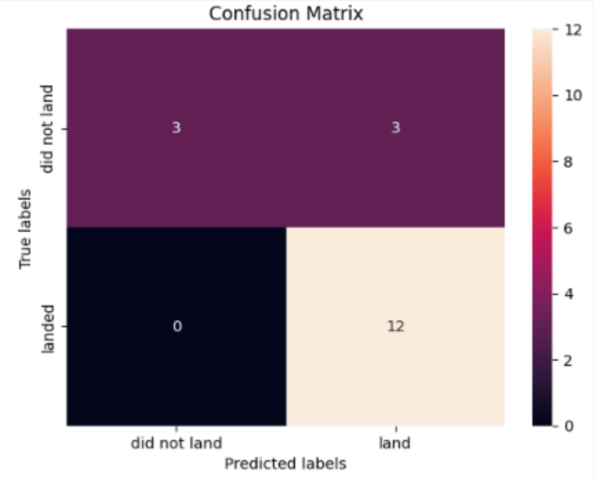
Logistic regression



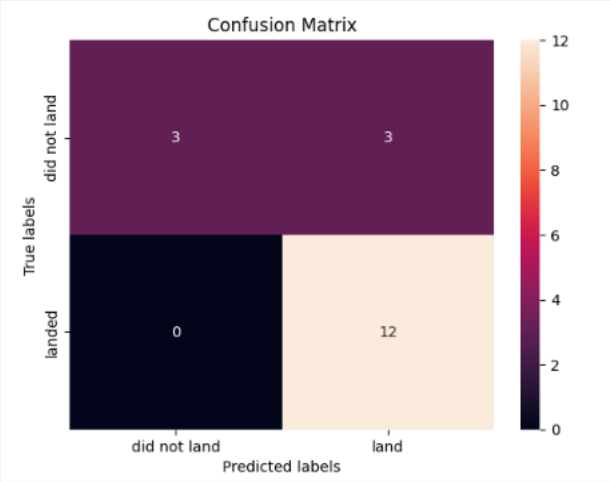
Decision Tree



SVM



KNN



As the test accuracy are all equal, the confusion matrices are also identical.

Conclusions

- The success of a space mission can be attributed to various factors, including the launch site, the specific orbit chosen, and notably, the number of previous launches. Accumulated knowledge and experience gained from previous launches likely contribute to the transition from launch failures to successful missions.
- Certain orbits, such as GEO, HEO, SSO, and ES L1, exhibit better success rates compared to others. Payload mass also plays a role in mission success, with some orbits requiring light or heavy payloads. Generally, missions with lower payload masses tend to perform better than those with heavier payloads.
- The reasons behind the varying performance of different launch sites, with KSC LC 39A identified as the best site, cannot be fully explained with the current data. To gain more insights, additional data, including atmospheric or other relevant information, may be necessary.
- For the given dataset, the Decision Tree Algorithm is chosen as the best model, even though the test accuracy is identical among all the models used. This decision is based on the better train accuracy achieved by the Decision Tree Algorithm.

Thank you!