IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Janaka Dabare
28th October 2025

Outline

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

Executive Summary

- For the data Analysis, Data collected via by calling SPACEX API and collected data were validated and many plots were made and machine leaning models were applied
- Recommends which model should be implemented get better success in future launches

Introduction

Starlink, a satellite internet constellation providing satellite Internet access. Sending manned missions to Space. One reason SpaceX can do this is the rocket launches are relatively inexpensive. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upwards of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage.

Purpose of this Analysis is to find dependent entities which effect success of future launches will be. I.E. for example which launch pad, which mass should be used, if sent to which orbit will be successful, etc.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
- Perform data wrangling
- 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

- Request to the SpaceX API
- Clean Request Data

Data Collection – SpaceX API

- SpaceX REST calls
    - url="https://api.spacexdata.com/v4/launches/past"
    - response = requests.get(url)

- https://github.com/ajkd/IBM-Data-Science-Capston/blob/main/dataset_part_1.csv

Data Collection - Scraping

- Web scrap Falcon 9 launch records with BeautifulSoup
- Extract a Falcon 9 launch records HTML table from Wikipedia

static_url =
https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922

response = requests.get(static_url,headers=headers)

- Parse the table and convert it into a Pandas data frame

soup = BeautifulSoup(response.content, 'html.parser')

- https://github.com/ajkd/IBM-Data-Science-Capston/blob/main/spacex_web_scraped.csv

Data Wrangling

- In the data set, there are several different cases where the booster did not land successfully. Sometimes a landing was attempted but failed due to an accident
- we will mainly convert those outcomes into Training Labels with 1 means the booster successfully landed 0 means it was unsuccessful.
- Perform exploratory Data Analysis and determine Training Labels

  - Exploratory Data Analysis

  - Determine Training Labels

- Read csv file from https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_1.csv

- Identify and calculate the percentage of the missing values in each attribute
- Identify which columns are numerical and categorical
- Calculate the number of launches on each site
- Calculate number and occurrence of each orbit,
- Calculate the number and occurence of mission outcome of the orbits¶
- Create a landing outcome label from Outcome column
- Found success rate (np.float64(0.6666666666666666))

- https://github.com/ajkd/IBM-Data-Science-Capston/blob/main/dataset_part_2.csv

EDA with Data Visualization

- Summarize what charts were plotted and why you used those charts

    - Visualize the relationship between Flight Number and Launch Site

    - Visualize the relationship between Payload and Launch Site

    - Visualize the relationship between success rate of each orbit type

    - Visualize the relationship between FlightNumber and Orbit type

    - Visualize the relationship between Payload and Orbit type

    - Visualize the launch success yearly trend

- https://github.com/ajkd/IBM-Data-Science-Capston/blob/main/dataset_part_3.csv

EDA with SQL

- SQL queries performed

  - SELECT * FROM SPACEXTBL

  - SELECT DISTINCT launch_site FROM SPACEXTBL

  - SELECT * FROM SPACEXTBL where launch_Site like 'CCA%' limit 200

  - SELECT sum(PAYLOAD_MASS__KG_) FROM SPACEXTBL where Customer like 'NASA%‘

  - SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL where Booster_Version like 'F9 v1.1%‘

  - SELECT Date FROM SPACEXTBL where Landing_Outcome = 'Success (ground pad)' order by Date Asc limit 1

  - SELECT Date, Booster_Version FROM SPACEXTBL where Landing_Outcome = 'Success (drone ship)' and PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000“

  - SELECT count(*) FROM SPACEXTBL where Mission_Outcome like 'Succ%‘

  - SELECT count(*) FROM SPACEXTBL where Mission_Outcome like 'Fail%'

- SELECT Booster_Version FROM SPACEXTBL

WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL

- SELECT CASE substr(Date, 6, 2) WHEN '01' THEN 'January' WHEN '02' THEN 'February' WHEN '03' THEN 'March' WHEN '04' THEN 'April' WHEN '05' THEN 'May' WHEN '06' THEN 'June' WHEN '07' THEN 'July' WHEN '08' THEN 'August' WHEN '09' THEN 'September' WHEN '10' THEN 'October' WHEN '11' THEN 'November' WHEN '12' THEN 'December' END as month_name, Booster_Version, Launch_Site, Landing_Outcome FROM SPACEXTBL WHERE substr(Date, 1, 4) = '2015' AND Landing_Outcome LIKE '%drone ship%' AND Landing_Outcome LIKE '%Failure%'

- SELECT Landing_Outcome, COUNT(*) as outcome_count FROM SPACEXTBL WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' AND Landing_Outcome IS NOT NULL GROUP BY Landing_Outcome ORDER BY outcome_count DESC

- https://github.com/ajkd/IBM-Data-Science-Capston/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Predictive Analysis (Classification)

- Summarize how you built, evaluated, improved, and found the best performing classification model

- Independent variable ata set was obtained from 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_3.csv' and target data was obtained from https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_2.csv

- Standardize the variable data

- Split the data with size=0.2

- With parameters ={'C':[0.01,0.1,1], 'penalty':['l2'], 'solver':['lbfgs']} Create a logistic regression object then create a GridSearchCV object and got result as tuned hpyerparameters :(best parameters) {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'} accuracy : 0.8464285714285713 and Test Set Accuracy: 0.8333

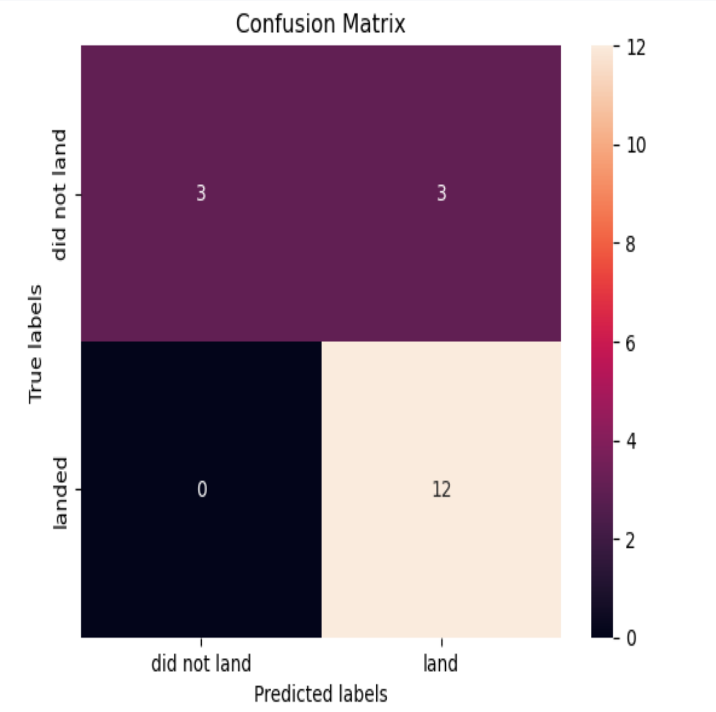
- Create a support vector machine object then create a GridSearchCV object svm_cv with cv = 10 with parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'), 'C': np.logspace(-3, 3, 5), 'gamma':np.logspace(-3, 3, 5)} and got result => tuned hpyerparameters :(best parameters) {'C': np.float64(1.0), 'gamma': np.float64(0.03162277660168379), 'kernel': 'sigmoid'} accuracy : 0.8482142857142856 and Test Set Accuracy: 0.8333

- Create a decision tree classifier object then create a GridSearchCV object tree_cv with cv = 10 with parameters = {'criterion': ['gini', 'entropy'], 'splitter': ['best', 'random'], 'max_depth': [2*n for n in range(1,10)], 'max_features': ['auto', 'sqrt'], 'min_samples_leaf': [1, 2, 4], 'min_samples_split': [2, 5, 10]} result ➔ tuned hpyerparameters :(best parameters) {'criterion': 'entropy', 'max_depth': 18, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'splitter': 'random'} accuracy : 0.8875 and Test Set Accuracy: 0.8333
- Create a k nearest neighbors object then create a GridSearchCV object knn_cv with cv = 10 with parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], 'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'], 'p': [1,2]} result ➔ tuned hpyerparameters :(best parameters) {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1} accuracy : 0.8482142857142858 and Test Set Accuracy: 0.8333

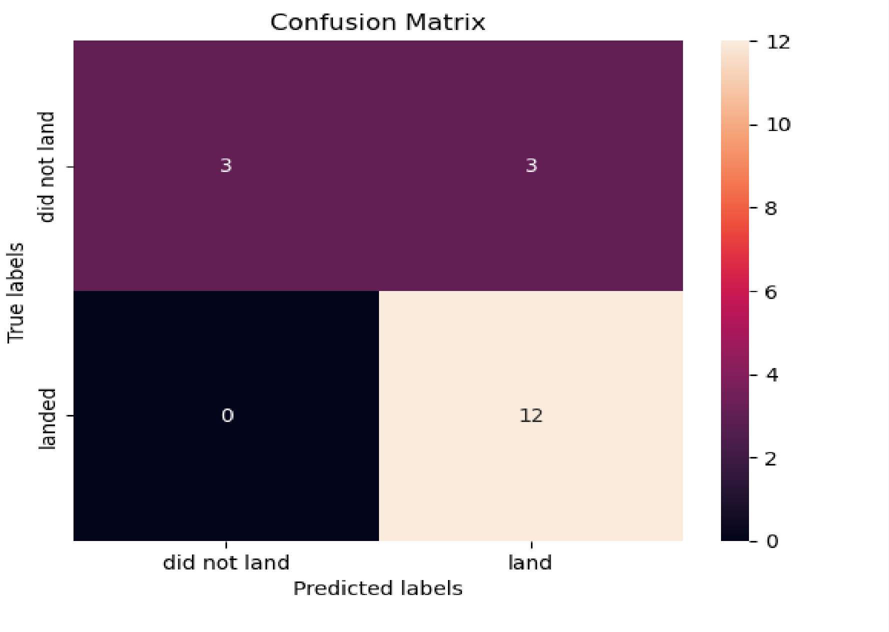
- https://github.com/ajkd/IBM-Data-Science-Capston/blob/main/SpaceX-Machine-Learning-Prediction-Part-5-v1.ipynb

Results

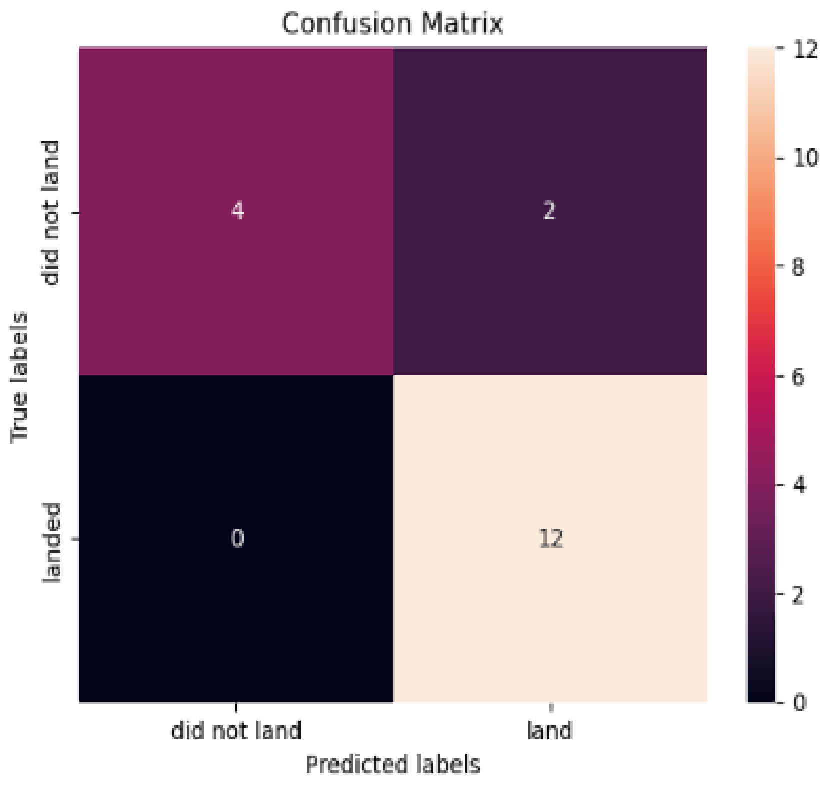
Logistc reg model



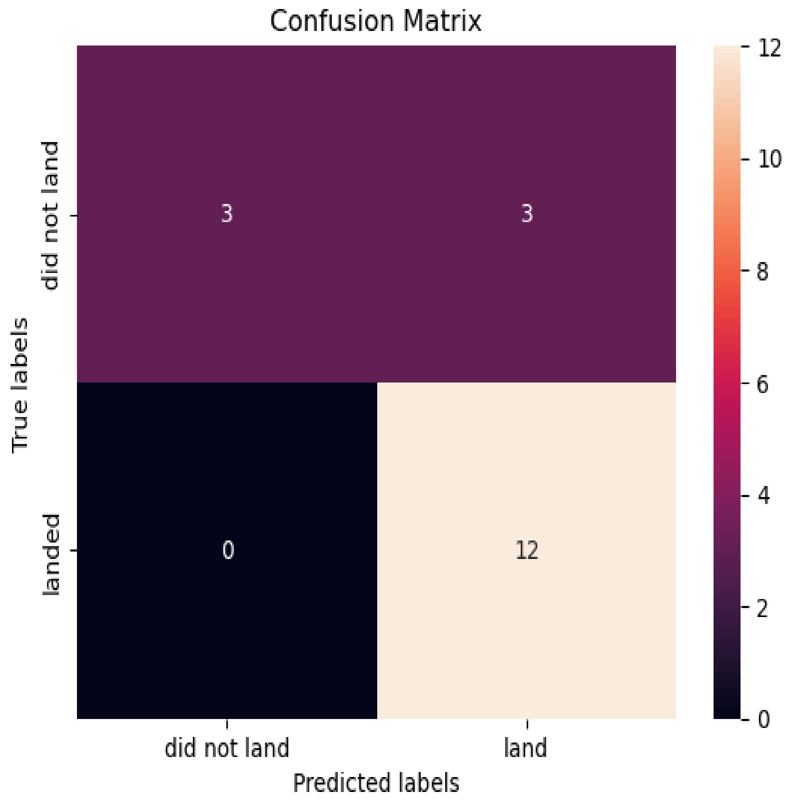
support vector machine model



Decision tree classifier



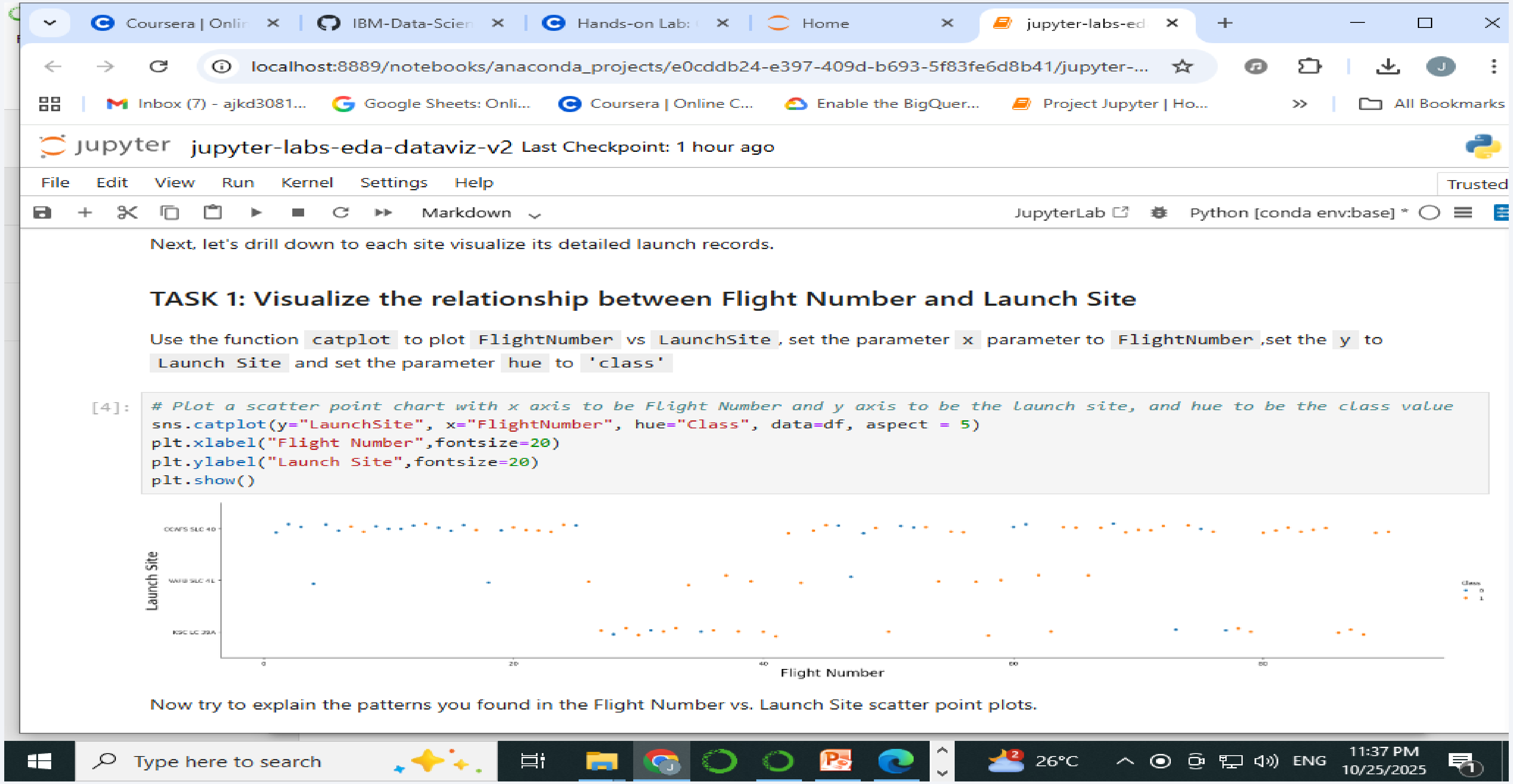
KNN Model



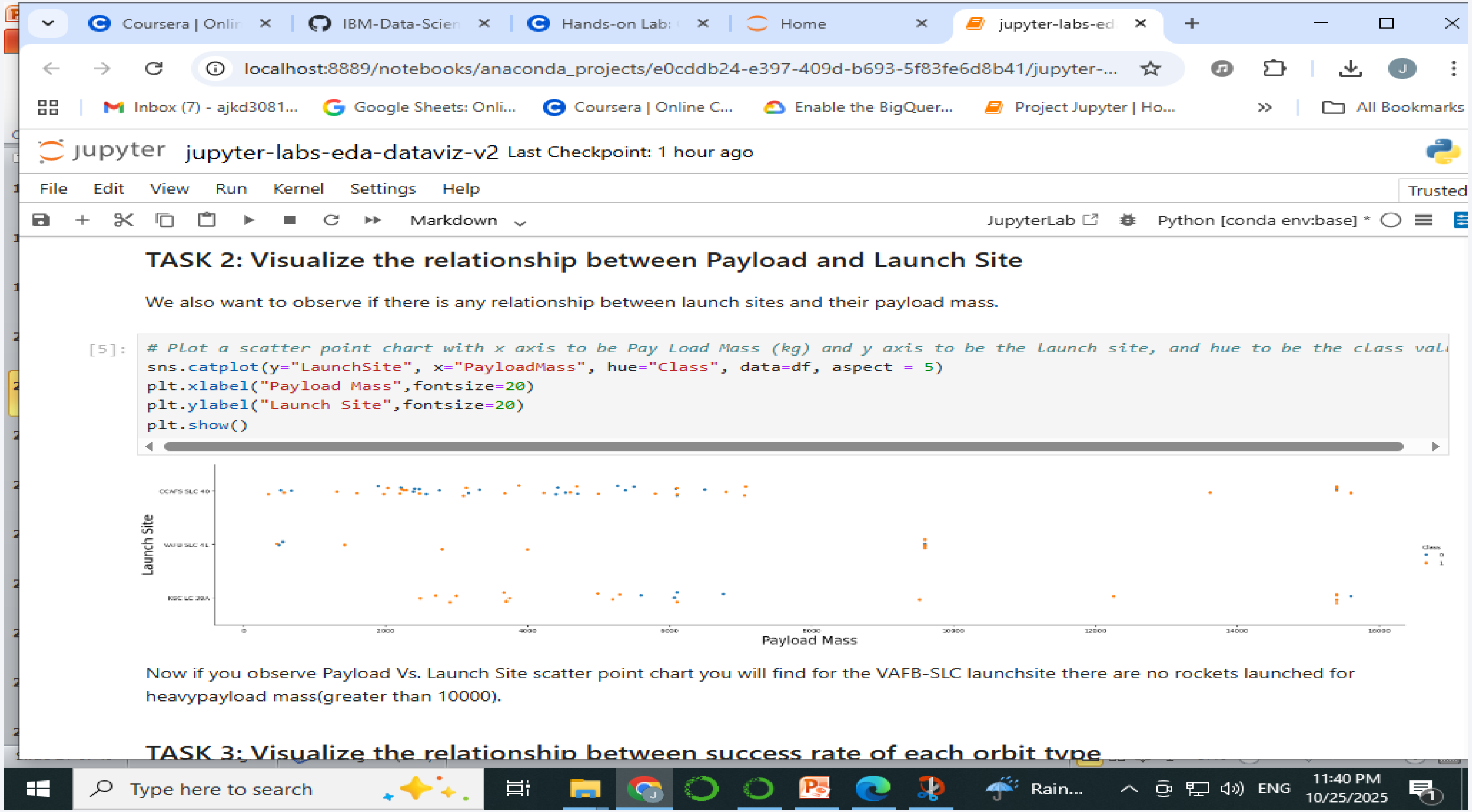
Section 2

Insights drawn from EDA

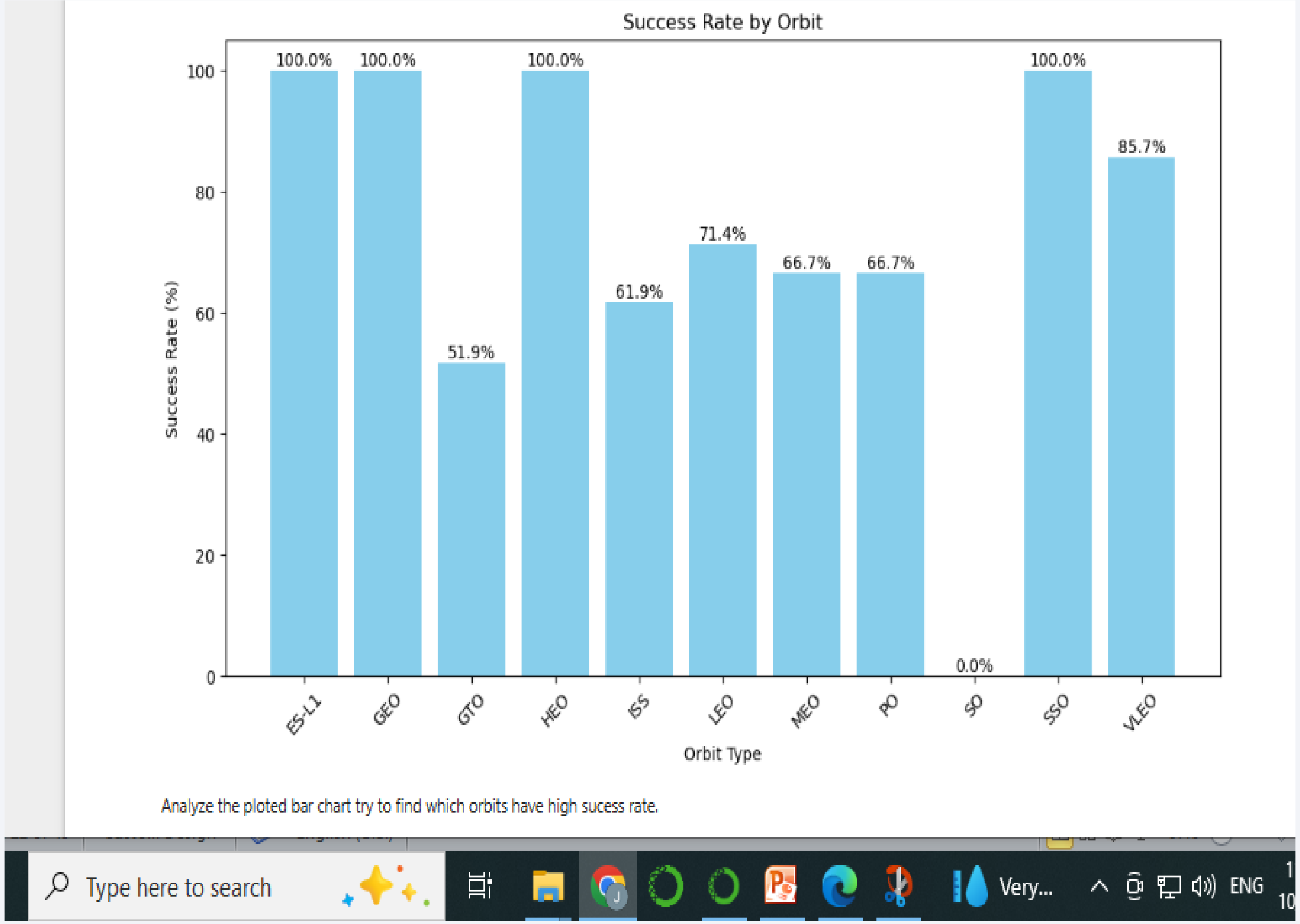
Flight Number vs. Launch Site



Payload vs. Launch Site

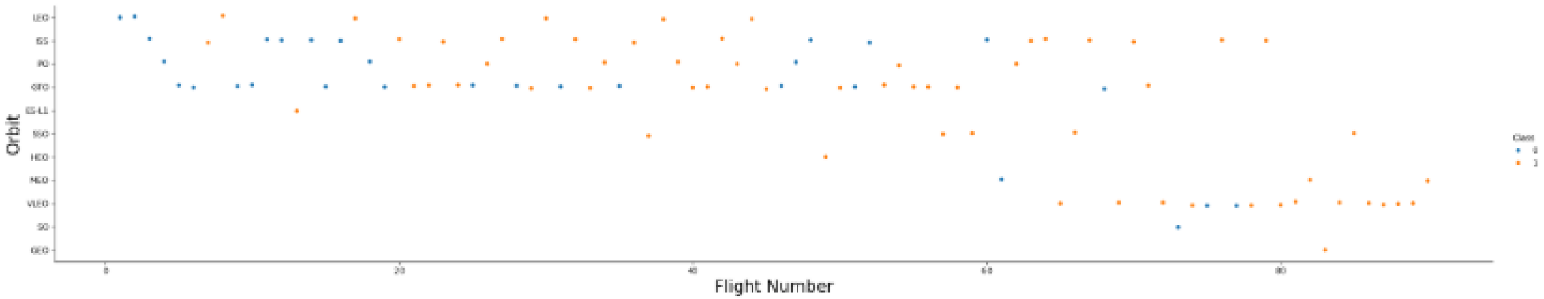


Success Rate vs. Orbit Type



Flight Number vs. Orbit Type

```
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()
```



You should see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

TASK 5: Visualize the relationship between Payload and Orbit type

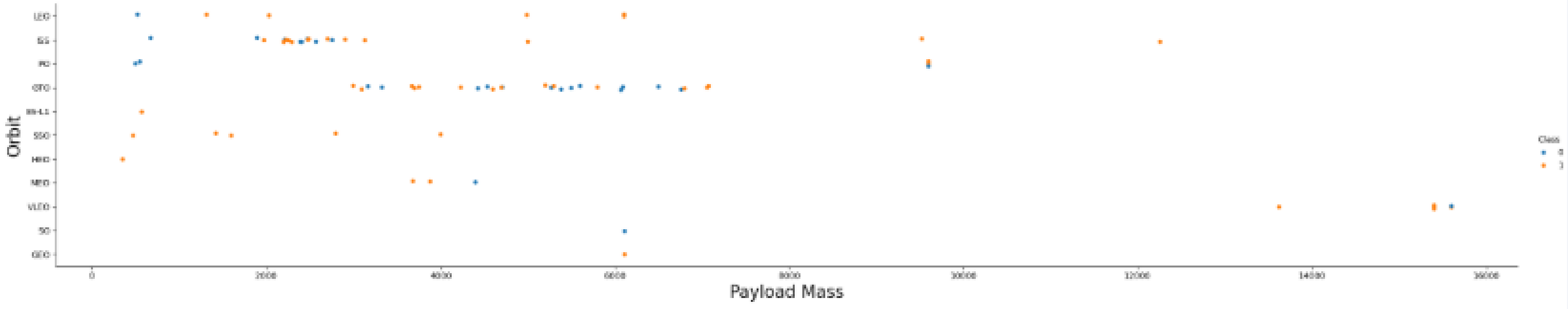
Similarly, we can plot the Payload vs. Orbit scatter point charts to reveal the relationship between Payload and Orbit type

Payload vs. Orbit Type

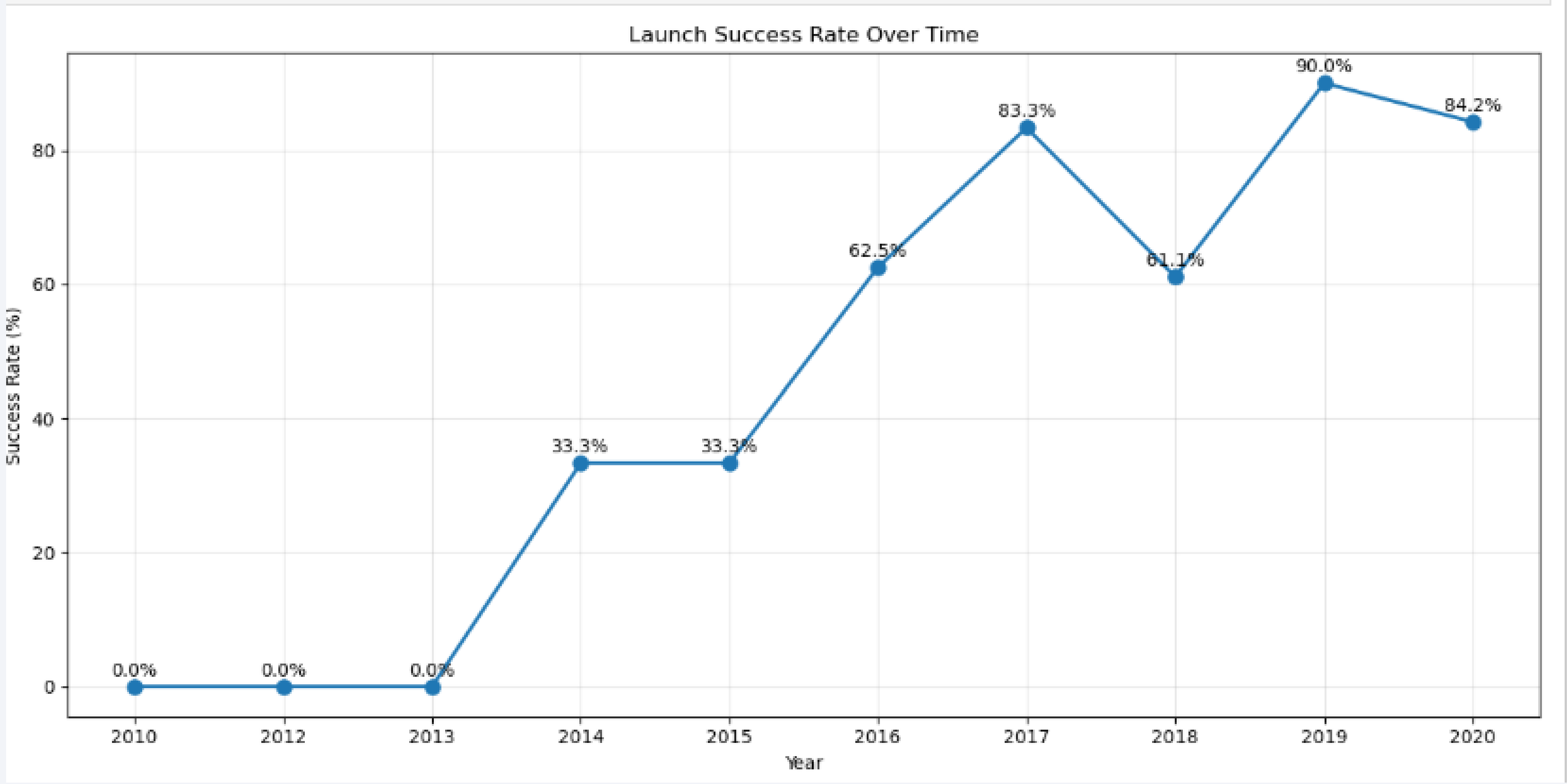
TASK 5: Visualize the relationship between Payload and Orbit type

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



Launch Success Yearly Trend



All Launch Site Names

- Unique launch sites:
- CCAFS LC-40
- VAFB SLC-4E
- KSC LC-39A
- CCAFS SLC-40

Launch Site Names Begin with 'CCA'

- 2010-06-04 18:45:00 F9 v1.0 B0003 CCAFS LC-40
- 2010-12-08 15:43:00 F9 v1.0 B0004 CCAFS LC-40
- 2012-05-22 7:44:00 F9 v1.0 B0005 CCAFS LC-40
- 2012-10-08 0:35:00 F9 v1.0 B0006 CCAFS LC-40
- 2013-03-01 15:10:00 F9 v1.0 B0007 CCAFS LC-40

Total Payload Mass

Task 3

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

```
0]: total_payload_mass = pd.read_sql("SELECT sum(PAYLOAD_MASS__KG_) FROM SPACEXTBL where Customer like 'NASA%'", con)
print( "total payload mass:")
print( total_payload_mass)
```

```
total payload mass:
   sum(PAYLOAD_MASS__KG_)
0                   99980
```

Average Payload Mass by F9 v1.1

Task 4

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

```
[11]: average_payload_mass = pd.read_sql("SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL where Booster_Version like 'F9 v1.1%'", con)
print( "average payload mass:")
print( average_payload_mass)
```

```
average payload mass:
   AVG(PAYLOAD_MASS__KG_)
0             2534.666667
```

First Successful Ground Landing Date

Task 5

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

Hint:Use min function

```
[12]: sdate = pd.read_sql("SELECT Date FROM SPACEXTBL where Landing_Outcome = 'Success (ground pad)' order by Date Asc limit 1 ", con)
print( "first succesful landing:")
print( sdate)
```

```
first succesful landing:
         Date
0  2015-12-22
```

Successful Drone Ship Landing with Payload between 4000 and 6000

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

```
[13]: names_of_the_boosters = pd.read_sql("SELECT Date, Booster_Version FROM SPACEXTBL where Landing_Outcome = 'Success (drone ship)' and PAYLOAD_MASS__KG_ > 4
print( "names of the boosters:")
print( names_of_the_boosters)
```

```
names of the boosters:
         Date Booster_Version
0  2016-05-06     F9 FT B1022
1  2016-08-14     F9 FT B1026
2  2017-03-30  F9 FT  B1021.2
3  2017-10-11  F9 FT  B1031.2
```

Total Number of Successful and Failure Mission Outcomes

Task 7

List the total number of successful and failure mission outcomes

[14]:
```
totals = pd.read_sql("SELECT count(*) FROM SPACEXTBL where Mission_Outcome like 'Succ%'", con)

print( "total success:")
print( totals)

totalf = pd.read_sql("SELECT count(*) FROM SPACEXTBL where Mission_Outcome like 'Fail%'", con)

print( "total falure:")
print( totalf)
```

```
total success:
   count(*)
0       100
total falure:
   count(*)
0         1
```

Boosters Carried Maximum Payload

Task 8

List all the booster_versions that have carried the maximum payload mass, using a subquery with a suitable aggregate function.

```
15]: max_payload_boosters = pd.read_sql("""
    SELECT Booster_Version
    FROM SPACEXTBL
    WHERE PAYLOAD_MASS__KG_ = (
        SELECT MAX(PAYLOAD_MASS__KG_)
        FROM SPACEXTBL
    )
""", con)

print("Boosters that carried maximum payload mass:")
print(max_payload_boosters)
```

```
Boosters that carried maximum payload mass:
   Booster_Version
0    F9 B5 B1048.4
1    F9 B5 B1049.4
2    F9 B5 B1051.3
3    F9 B5 B1056.4
4    F9 B5 B1048.5
5    F9 B5 B1051.4
6    F9 B5 B1049.5
7   F9 B5 B1060.2
8   F9 B5 B1058.3
9    F9 B5 B1051.6
10   F9 B5 B1060.3
11  F9 B5 B1049.7
```

2015 Launch Records

Task 9

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

Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

[16]:

```
result = pd.read_sql("""
    SELECT
        CASE substr(Date, 6, 2)
            WHEN '01' THEN 'January' WHEN '02' THEN 'February' WHEN '03' THEN 'March'
            WHEN '04' THEN 'April' WHEN '05' THEN 'May' WHEN '06' THEN 'June'
            WHEN '07' THEN 'July' WHEN '08' THEN 'August' WHEN '09' THEN 'September'
            WHEN '10' THEN 'October' WHEN '11' THEN 'November' WHEN '12' THEN 'December'
        END as month_name,
        Booster_Version, Launch_Site, Landing_Outcome
    FROM SPACEXTBL
    WHERE substr(Date, 1, 4) = '2015'
    AND Landing_Outcome LIKE '%drone ship%'
    AND Landing_Outcome LIKE '%Failure%'
""", con)
print(result)
```

```
  month_name Booster_Version  Launch_Site       Landing_Outcome
0    January   F9 v1.1 B1012  CCAFS LC-40  Failure (drone ship)
1      April   F9 v1.1 B1015  CCAFS LC-40  Failure (drone ship)
```

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

Task 10

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.

17]:
```
landing_rank = pd.read_sql("""
    SELECT
        Landing_Outcome,
        COUNT(*) as outcome_count
    FROM SPACEXTBL
    WHERE Date BETWEEN '2010-06-04' AND '2017-03-20'
    AND Landing_Outcome IS NOT NULL
    GROUP BY Landing_Outcome
    ORDER BY outcome_count DESC
""", con)

print("Landing Outcomes Ranked by Count (2010-06-04 to 2017-03-20):")
print(landing_rank)
```

```
Landing Outcomes Ranked by Count (2010-06-04 to 2017-03-20):
          Landing_Outcome  outcome_count
0              No attempt             10
1     Success (drone ship)             5
2     Failure (drone ship)             5
3     Success (ground pad)             3
4       Controlled (ocean)             3
5     Uncontrolled (ocean)             2
6      Failure (parachute)             2
7   Precluded (drone ship)             1
```

Section 3

Launch Sites Proximities Analysis

<Folium Map Screenshot 1>

- Replace <Folium map screenshot 1> title with an appropriate title
- Explore the generated folium map and make a proper screenshot to include all launch sites’ location markers on a global map
- Explain the important elements and findings on the screenshot

<Folium Map Screenshot 2>

- Replace <Folium map screenshot 2> title with an appropriate title
- Explore the folium map and make a proper screenshot to show the color-labeled launch outcomes on the map
- Explain the important elements and findings on the screenshot

<Folium Map Screenshot 3>

- Replace <Folium map screenshot 3> title with an appropriate title

- Explore the generated folium map and show the screenshot of a selected launch site to its proximities such as railway, highway, coastline, with distance calculated and displayed

- Explain the important elements and findings on the screenshot

Section 4

Build a Dashboard with Plotly Dash

<Dashboard Screenshot 1>

- Replace <Dashboard screenshot 1> title with an appropriate title
- Show the screenshot of launch success count for all sites, in a piechart
- Explain the important elements and findings on the screenshot

<Dashboard Screenshot 2>

- Replace <Dashboard screenshot 2> title with an appropriate title

- Show the screenshot of the piechart for the launch site with highest launch success ratio

- Explain the important elements and findings on the screenshot

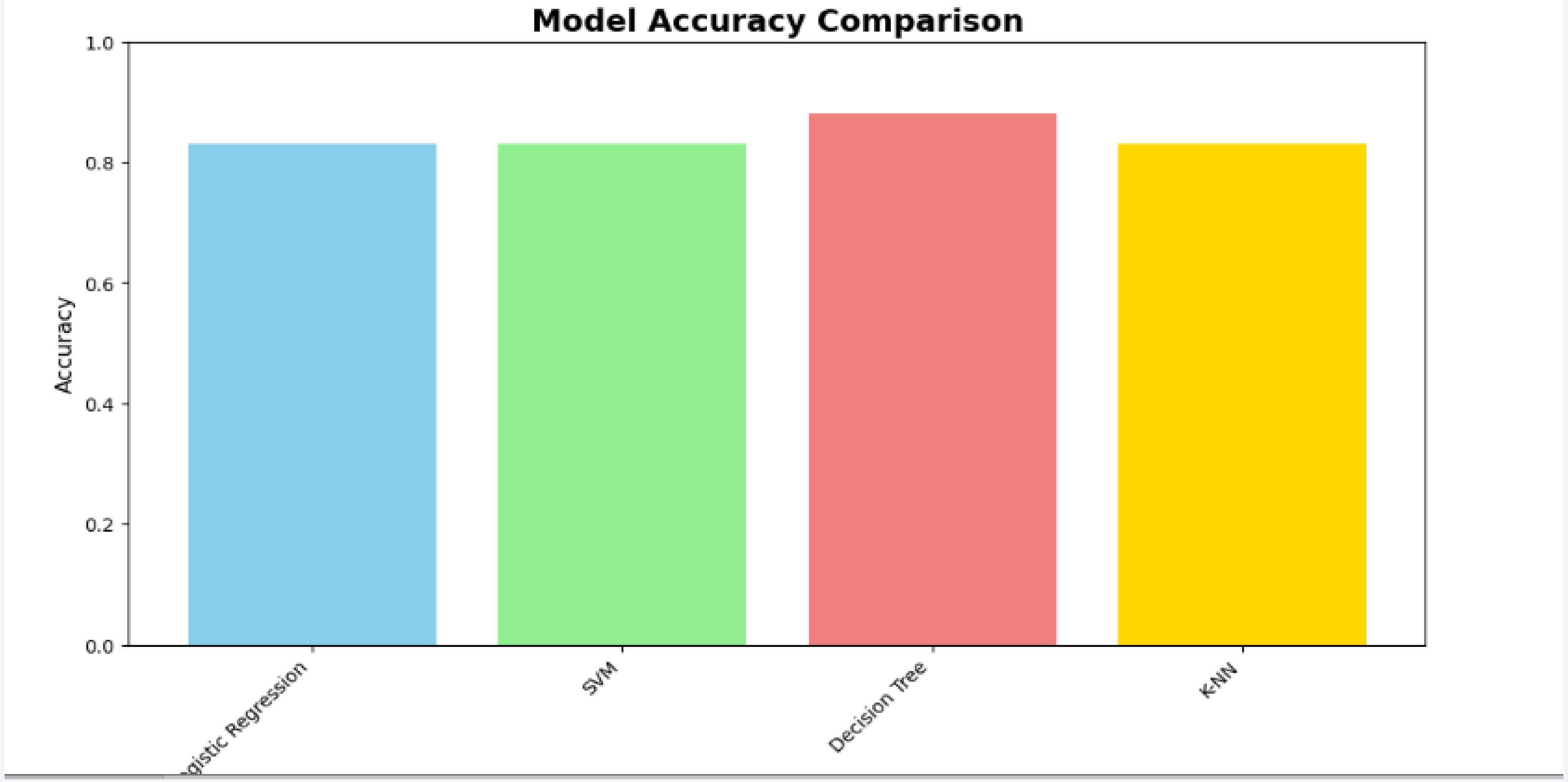
<Dashboard Screenshot 3>

- Replace <Dashboard screenshot 3> title with an appropriate title
- Show screenshots of Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider
- Explain the important elements and findings on the screenshot, such as which payload range or booster version have the largest success rate, etc.

Section 5

Predictive Analysis (Classification)

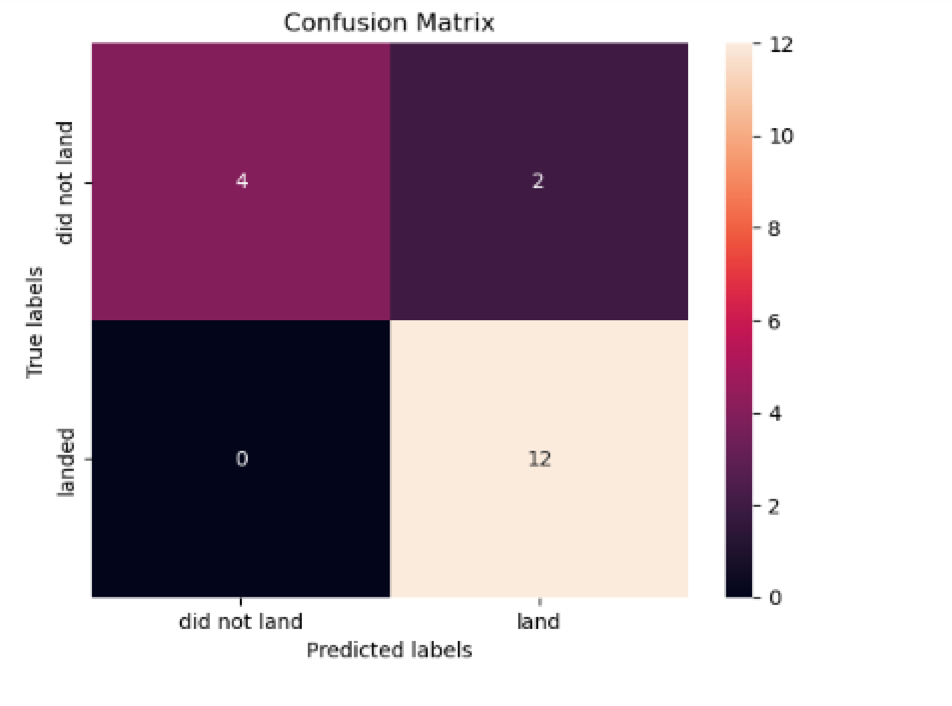
Classification Accuracy



- according to bar chart accurate model Is Decision Tree model (test Accuracy 0.88 where all other models gives 0.83). This also confirm with the confusion matrix.

Confusion Matrix

- Confusion matrix of the decision tree model



Conclusions

- We have accessed the SPACEX API and validated
- Many Plots were made
- Machine learning models logistic regression, SVM, decision tree, KNN methods were applied
- Learnt decision tree model gives better result

…

Appendix

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

Thank you!