Data Science Capstone

Luke Ganalon

11/15/2023

OUTLINE

- Executive Summary
- Introduction
- Methodology
- Results
  - Visualization – Charts
  - Dashboard
- Discussion
  - Findings & Implications
- Conclusion
- Appendix

EXECUTIVE SUMMARY

- Summary of methodologies
  - SpaceX Data Collection using SpaceX API
  - SpaceX Data Collection with Web Scraping
  - SpaceX Data Wrangling
  - SpaceX Exploratory Data Analysis using SQL
  - SpaceX EDA DataViz
  - SpaceX Launch Sites Analysis
  - SpaceX Machine Learning Landing Prediction
- Summary of all results
  - EDA Results
  - Interactive Visual Analytics and Dashboards
  - Predictive Analysis

INTRODUCTION

- Project background
  - SpaceX Falcon 9 rocket launches save money due to the reuse of their first stage.
- Problem to solve
  - Predict if the Falcon 9 first stage will land successfully

METHODOLOGY

- Data Collection methodology
  - Describes how data was collected
- Data Wrangling
  - Processing the data
- EDA
- Folium and Plotly Dash
- Predictive Analysis
  - Building, tuning, and evaluating models

Data Collection

Collecting Data on the SpaceX Falcon9

- Data was collected using the SpaceX API. A series of helper functions were defined to extract information using identification numbers. Then the data was requested from the SpaceX API url.
- The SpaceX launch data was requested and parsed using the GET request and then decoded as a JSON file to be converted into a Pandas data frame.
- Web scraping was also performed to collect the historical data of Falcon9 launches from the given wikipedia website. Using BeautifulSoup and request libraries, the Falcon9 HTML table records were extracted. Finally, the table was parsed and converted into a Pandas data frame.
- Here is the relative Github URLs: https://github.com/lganalon/IBM-Data-Science-Capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb & https://github.com/lganalon/IBM-Data-Science-Capstone/blob/main/jupyter-labs-webscraping.ipynb

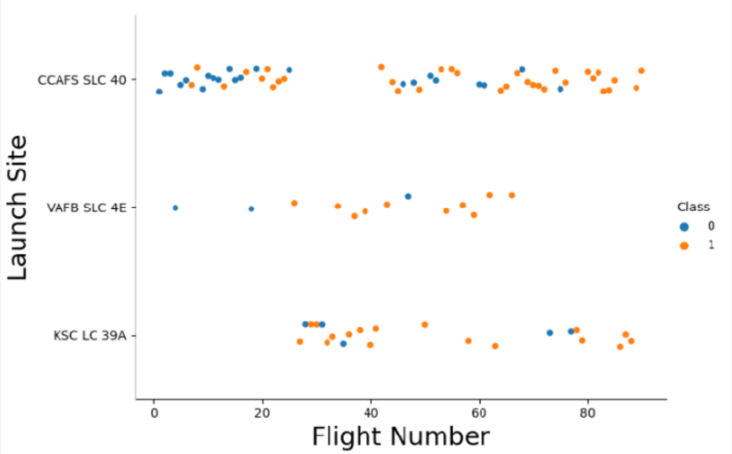
Data Wrangling

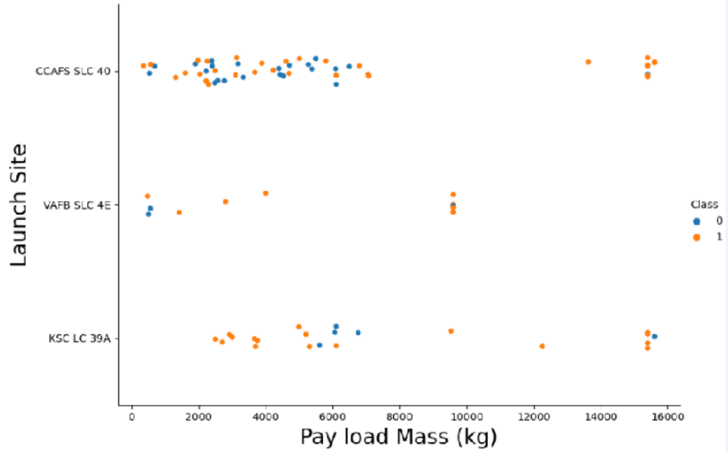
- After the creation of the Pandas DF, the data was filtered to only keep the launches of the Falcon9. The missing values were then dealt with.
- Exploratory Data Analysis (EDA) was then performed to find some patterns in the data.
- Here is the relative Github URL: https://github.com/lganalon/IBM-Data-Science-Capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

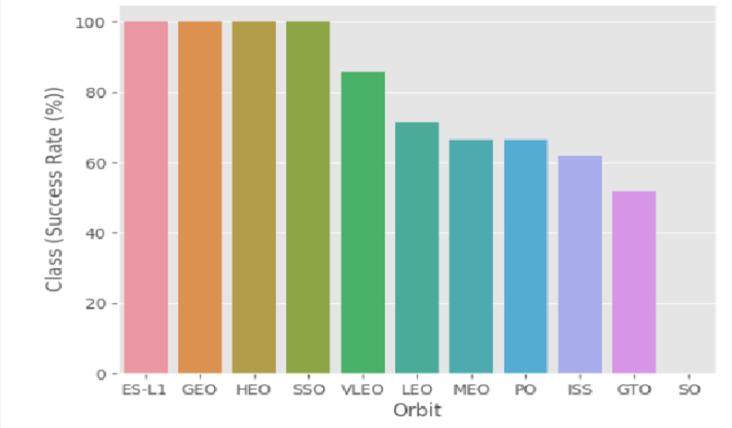
EDA with Data Visualization

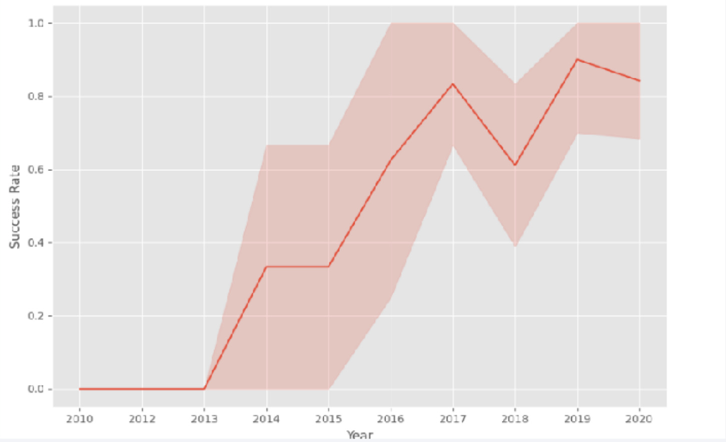
- After the creation of the Pandas DF, the data was filtered to only keep the launches of the Falcon9. The missing values were then dealt with.
- Exploratory Data Analysis (EDA) was then performed to find some patterns in the data.
- Here is the relative Github URL: https://github.com/lganalon/IBM-Data-Science-Capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization









EDA with SQL

- SQL queries:
  - Display the names of the unique launch sites
  - Display 5 records where launch sites began
  - Display the total payload mass carried by boosters
  - Display average payload mass carried by booster version F9 v1.1
  - List the date when the first successful landing outcome in the ground pad was achieved
  - List the names of boosters which have success in drone ship and have a payload between 4000 and 6000 kg
  - List the total number of successful and failed mission outcomes
  - Github URL: https://github.com/lganalon/IBM-Data-Science-Capstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

EDA with SQL

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

```
 * ibm_db_sa://yyy33800:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/BLUDB
   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	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	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

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

```
 * ibm_db_sa://yyy33800:***@1bbf73c5-d84a-4
   sqlite:///my_data1.db
Done.
```

1
45596

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

```
 * ibm_db_sa://yyy33800:***@1bbf73c5-d84a-4
   sqlite:///my_data1.db
Done.
```

1
2928

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

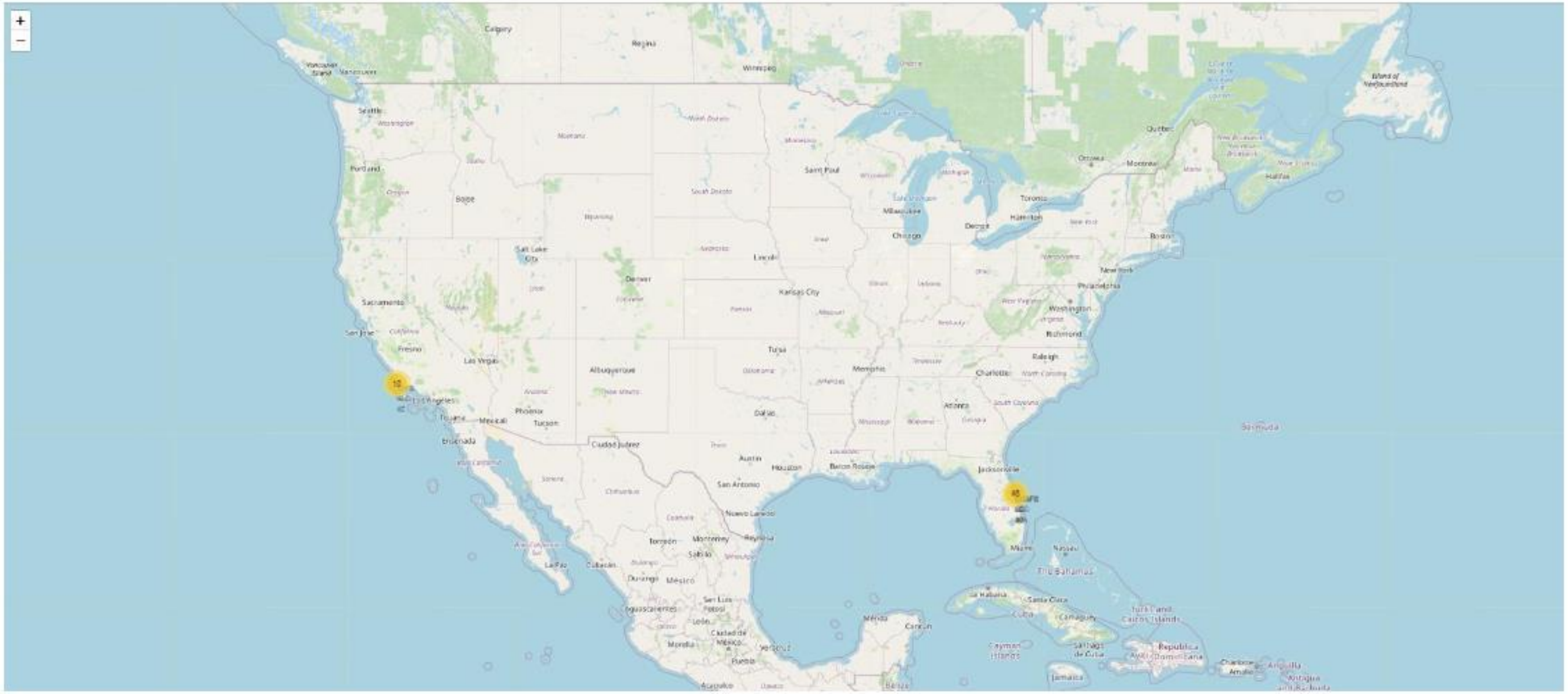
```
 * sqlite:///my_data1.db
Done.
```

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

Building an Interactive Map w/ Folium

- Built a folium map to mark the launch sites
- Added map objects such as markers, circles, and lines to mark the success or failure of launches for each site
- Created a launch set outcomes (failure=0 and success=1)
- Github URL: https://github.com/lganalon/IBM-Data-Science-Capstone/blob/main/lab_jupyter_launch_site_location.jupyterlite.ipynb

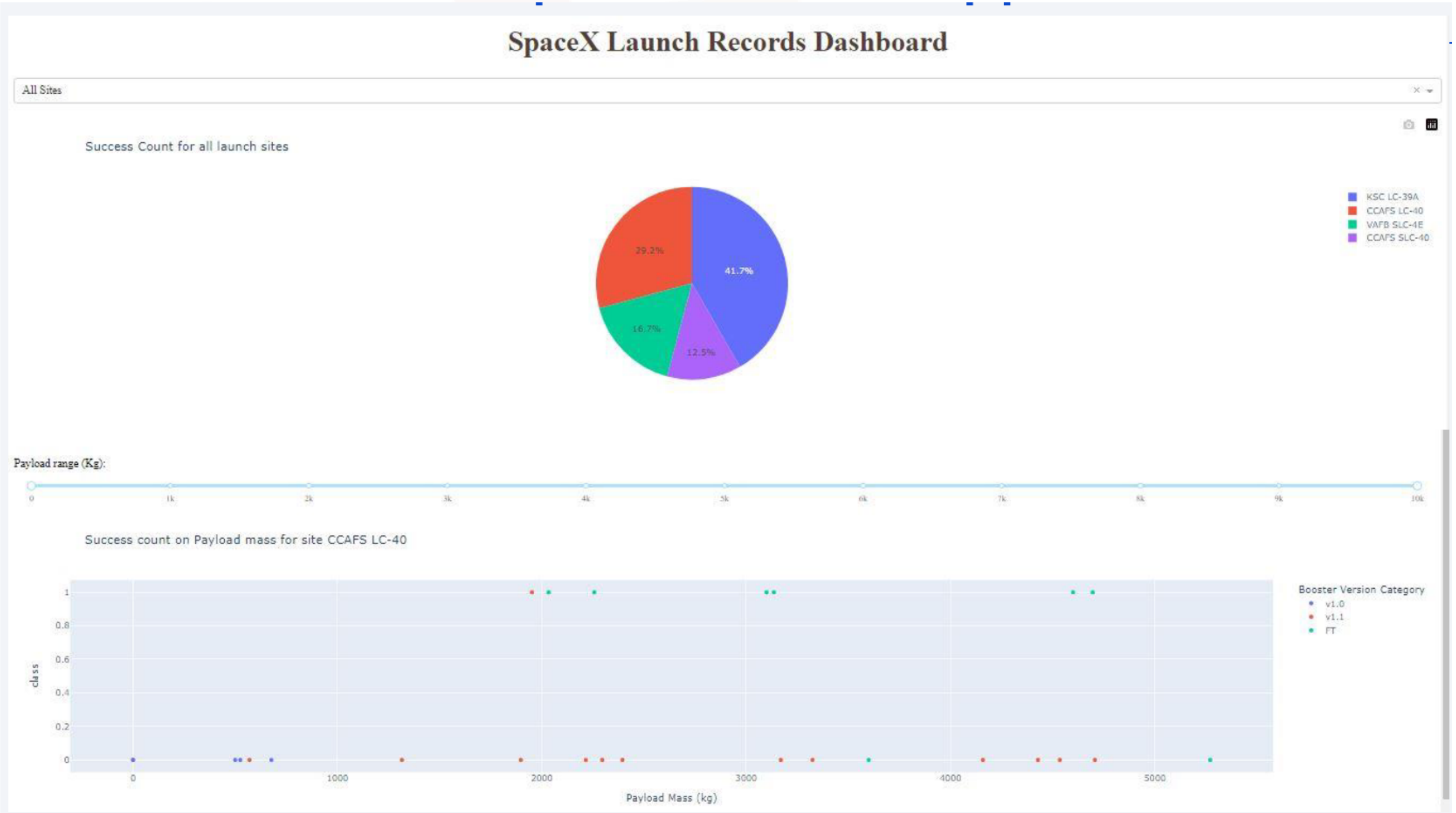
Building an Interactive Map w/ Folium



Build a Dashboard w/ Plotly Dash

- Built an interactive dashboard application with Plotly Dash
  - Add launch sites drop-down input component
  - Add callback functions to render success pie-chart and render the success-payload scatter plot
  - Add a range slider to Select Payload kilograms
- Github URL: https://github.com/lganalon/IBM-Data-Science-Capstone/blob/main/spacex_dash_app.py

SpaceX Dash App



Predictive Analysis (Classification)

- Summary of the classification model
- First, EDA was performed and training labels were determined by:
  - Creating a NumPy array from the column Class, by applying the method of to_numpy() then assign it to the variable 'Y' as the outcome variable
  - Standardize the feature dataset by transforming it using the StandardScaler() function from Sklearn
  - Split the data into training and test sets and apply parameters

Predictive Analysis (Classification) cont.

- Run various models to see which performs best on the test set
  - SVM
  - Classification Tree
  - K-Nearest-Neighbors
  - Logistic Regression

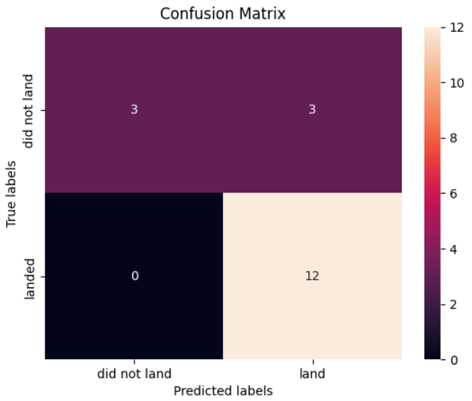
Out[68]:

	0
Method	Test Data Accuracy
Logistic_Reg	0.833333
SVM	0.833333
Decision Tree	0.833333
KNN	0.833333

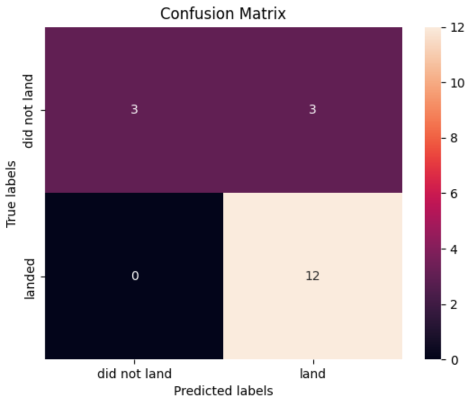
Github URL: https://github.com/lganalon/IBM-Data-Science-Capstone/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Predictive Analysis (Classification) cont.

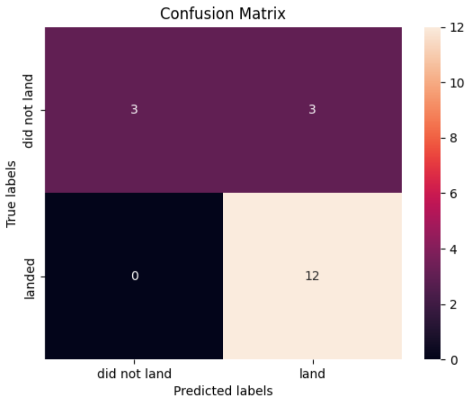
Logistic Regression



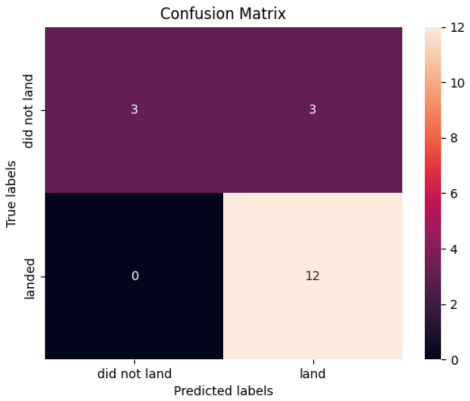
Decision Tree



SVM



KNN



CONCLUSION

- Model Performance: The models performed similarly on the test set with the decision tree model slightly outperforming
- Equator: Most of the launch sites are near the equator for an additional natural boost - due to the rotational speed of earth - which helps save the cost of putting in extra fuel and boosters
- Coast: All the launch sites are close to the coast
- Launch Success: Increases over time
- KSC LC-39A: Has the highest success rate among launch sites. Has a 100% success rate for launches less than 5,500 kg
- Orbits: ES-L1, GEO, HEO, and SSO have a 100% success rate
- Payload Mass: Across all launch sites, the higher the payload mass (kg), the higher the success rate