IBM Developer
SKILLS NETWORK

My Applied Data Science Capstone Project

<Lena Munad>
<03-11-2024>

🚀Table of Contents🚀

Executive Summary

Welcome to the culmination of my journey through the IBM Data Science Professional Certificate on Coursera. This project aimed to predict the successful first-stage landing of SpaceX's Falcon 9 rockets.

Methodology

- Data Collection & Wrangling
- EDA & Visual Analytics
- Predictive Analysis

- My findings identified key factors influencing landing success and established a machine learning model for future predictions. This capability can empower stakeholders to estimate launch costs and potentially inform competitive bidding strategies.

Introduction

Project Background and Context:

- This project focuses on predicting successful landings of the Falcon 9 rocket's first stage, a key factor in SpaceX's cost-effective launches. By predicting this, we can estimate launch costs and empower potential competitors with valuable information for bidding against SpaceX. This predictive capability is crucial as successful first-stage reuse significantly reduces launch costs compared to traditional methods.

Section
1

Methodology

Methodology

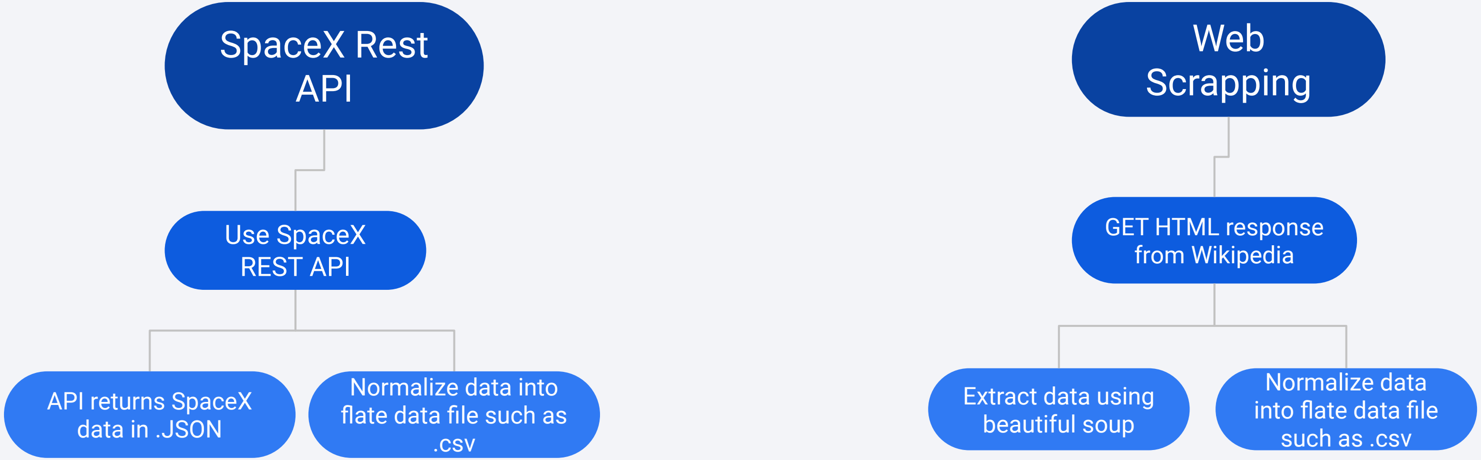
Executive Summary

- Data collection methodology:
    - Using SpaceX Rest API
    - Using Web Scrapping from Wikipedia
- Perform data wrangling
    - Preprocessed the data to ensure accuracy and consistency.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
    - Implemented classification algorithms to build a model for predicting landing success. Fine-tuned the model and evaluating its performance to ensure reliable predictions.

Data Collection

Data collection methodology:

- Using SpaceX Rest API
- Using Web Scrapping from Wikipedia



Data Collection – SpaceX API

1.Getting Responses from API

2.Converting Response to .JSON file

3.Apply custom functions to clean data

4. Assign list to dictionary then dataframe

5. Filter dataframe and then export to flat file (.csv)

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

```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

```
# Call getBoosterVersion
getBoosterVersion(data)
# Call getPayloadData
getPayloadData(data)
# Call getLaunchSite
getLaunchSite(data)
# Call getCoreData
getCoreData(data)
```

```
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_falcon9 = df.loc[df['BoosterVersion']!="Falcon 1"]
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

[LenaMunad-IBM-Data-Science-Capstone-Project.git]

Data Collection - Scraping

1. GET Responses from HTML
2. Create BeautifulSoup Object
3. Find tables
4. GET column names
5. Creation of dictionary
6. Append data to keys
7. Convert dictionary to dataframe
8. Convert dataframe to flat file (.csv)

```
# use requests.get() method with the provided static_url
# assign the response to a object
page=requests.get(static_url)
```

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(page.text, 'html.parser')
```

```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables=soup.find_all('table')
```

```
column_names = []
# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
for i in first_launch_table.find_all('th'):
    if extract_column_from_header(i)!=None:
        if len(extract_column_from_header(i))>0:
            column_names.append(extract_column_from_header(i))
```

```
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']=[]
```

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
```

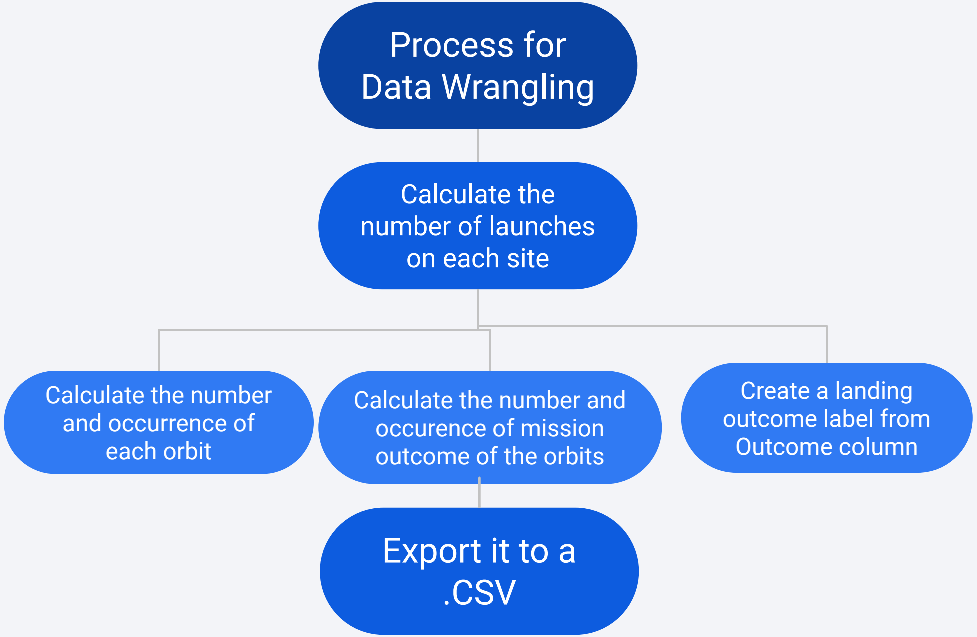
```
df=pd.DataFrame(launch_dict)
```

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

LenaMunad-IBM-Data-Science-Capstone-Project.git

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; for example, 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 on a drone ship False ASDS means the mission outcome was unsuccessfully landed on a drone ship.



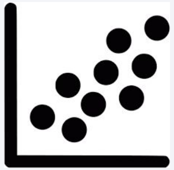
LenaMunad-IBM-Data-Science-Capstone-Project.git

EDA with Data Visualization

Scatter plot

Scatter plot is a great tool to depict correlation.

- **FlightNumber vs PayloadMass**
- **FlightNumber vs LaunchSite**
- **FlightNumber vs Orbit type**
- **Payload vs Launch Site**
- **Payload vs Orbit type**



Bar Chart

Bar chart compares the measure of categorical dimension.

- **Success rate vs Orbit type**

Line Chart

Line chart displays the trends and developments of numeric data over time.

- **Launch success yearly trend**

[LenaMunad-IBM-Data-Science-Capstone-Project.git]

EDA with SQL

Summary of SQL queries performed:

- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'CCA'
- Display the total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first successful landing outcome in ground pad was achieved
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- List the total number of successful and failure mission outcomes
- List the names of the booster versions which have carried the maximum payload mass
- Display the records which will display the month names, failure landing_outcomes in drone ship, booster versions, launch_site for the months in year 2015
- Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

Build an Interactive Map with Folium

- **Mark all Launch sites on an interactive map:**
  - The query uses latitude and longitude coordinates to add circle markers around each launch site with a label for the site name.

- **Mark successful launches for each site on the map:**
  - The query assigns success and failure launch outcomes to classes 0 and 1. It then displays them on the map with green markers for successful launches and red markers for failures, using a MarkerCluster() function.

- **Calculate distances between a Launch site and its surroundings:**
  - The query calculates the distance from the launch site to features like highways, roadways, and coastlines using the Haversine formula. Lines are then drawn on the map to show these distances.

Examples of some trends in which Launch site is situated in:

- Are launch sites in close proximity to railways? **No**
- Are launch sites in close proximity to highways? **No**
- Are launch sites in close proximity to coastline? **Yes**
- Do launch sites keep certain distance away from cities? **Yes**

LenaMunad-IBM-Data-Science-Capstone-Project.git

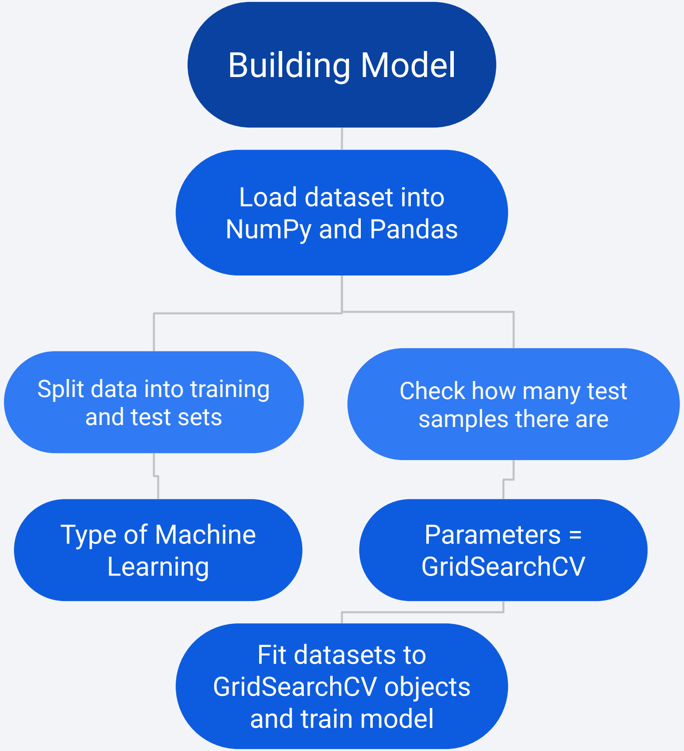
Build a Dashboard with Plotly Dash

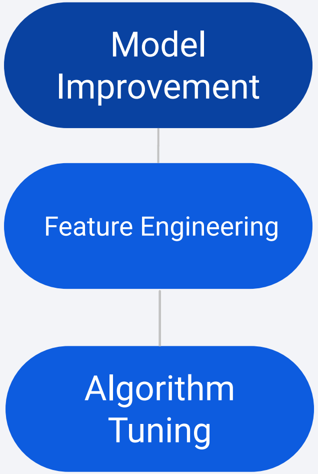
- A Plotly Dash application for users to perform interactive visual analytics on SpaceX launch data in real-time
- An interactive dashboard allows users to play around and view filtered output.

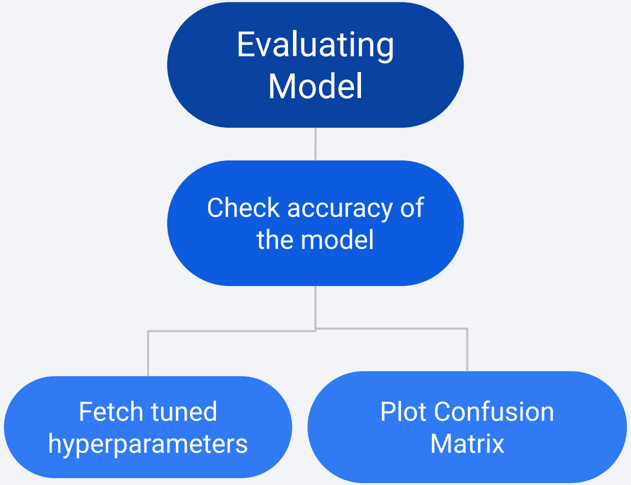
Graphs displayed:

- **Pie Chart** showing the total launches by a specific site/all sites.
  - This chart displays the relative proportions of launches for each launch site. The size of the circle corresponds to the total number of launches from that site.
- **Scatter plot** displaying the relationship Payload Mass (Kg) vs Launch Outcome for the different Booster Versions.
  - This scatter plot allows users to see if there is a correlation between the payload mass and launch outcome (successful or failed) for different booster versions. Users can filter the range of data displayed by launch outcome.

Predictive Analysis (Classification)







How to find the best performing classification model?

- The model with the best accuracy score is chosen.

[LenaMunad-IBM-Data-Science-Capstone-Project.git]

Results

- Exploratory data analysis results

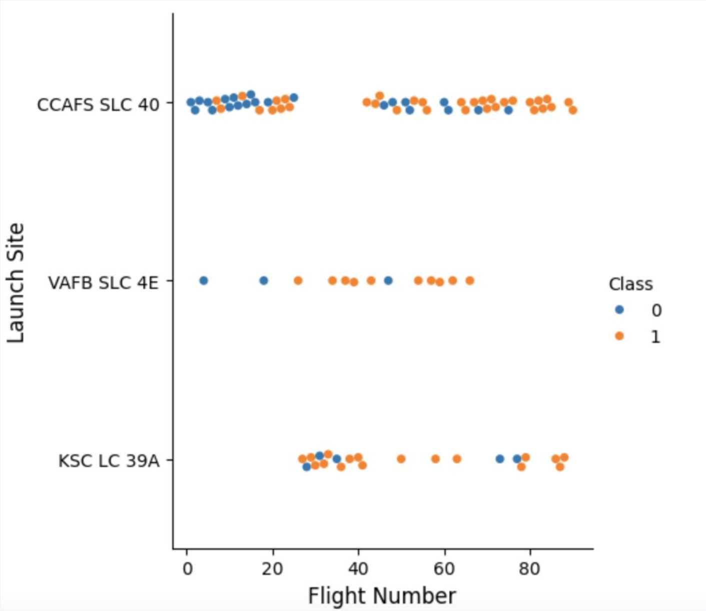
- Predictive analysis results

- Interactive analytics demo in screenshots

Section
2

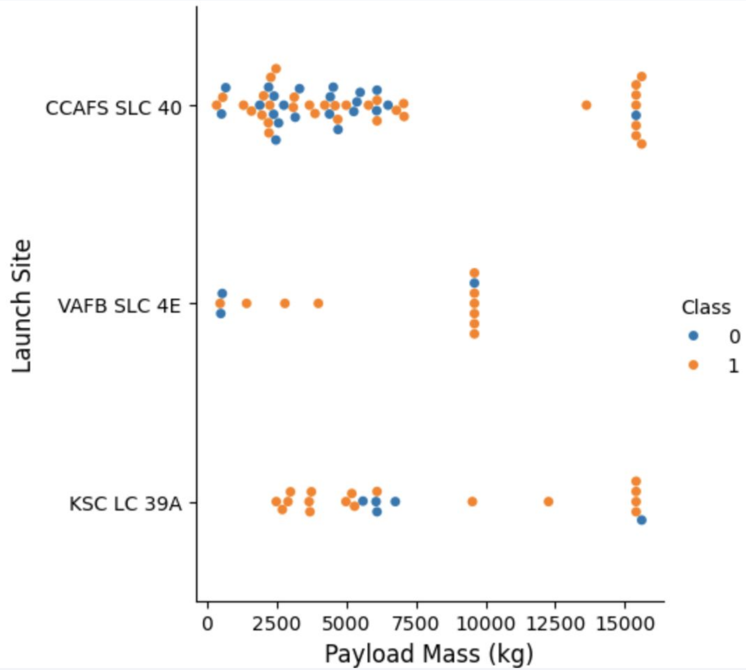
Insights drawn from EDA

Flight Number vs. Launch Site



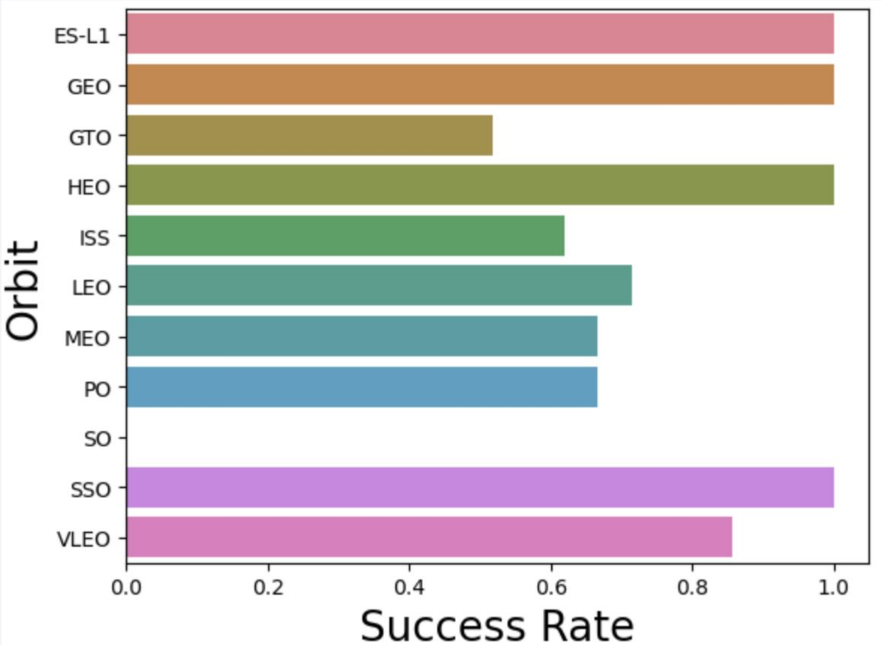
The more flights at a launch site the greater the success rate at the launch site.

Payload vs. Launch Site



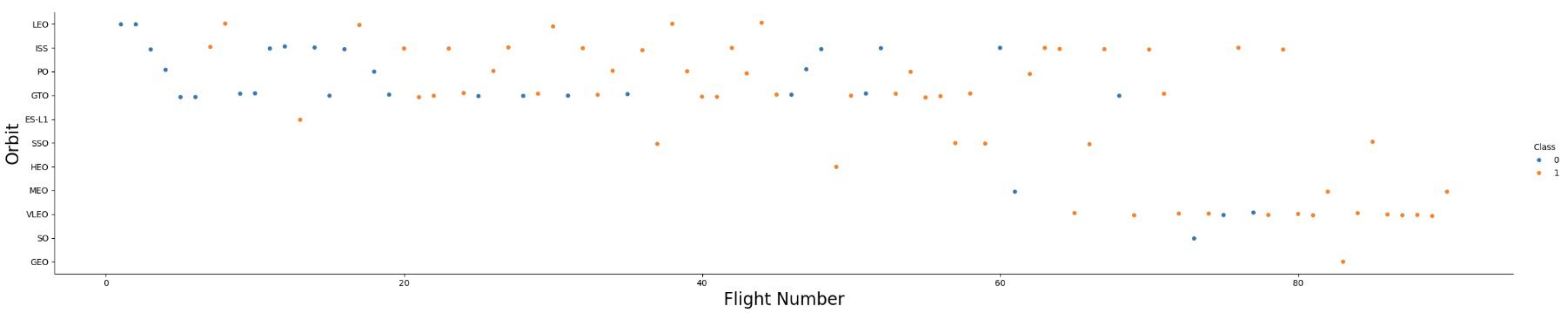
In the Payload vs Launch Site scatter plot, you will find for the VAFB-SLC 4E launch site there are no rockets launched for heavy payload mass (greater than 10000).

Success Rate vs. Orbit Type



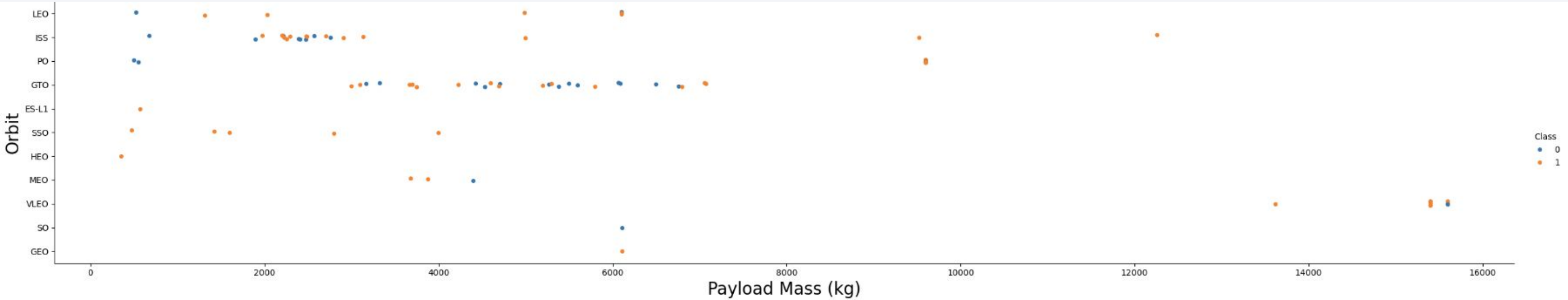
Orbit type GEO, HEO, SSO, ES-L1 have the highest Success Rate.

Flight Number vs. Orbit Type



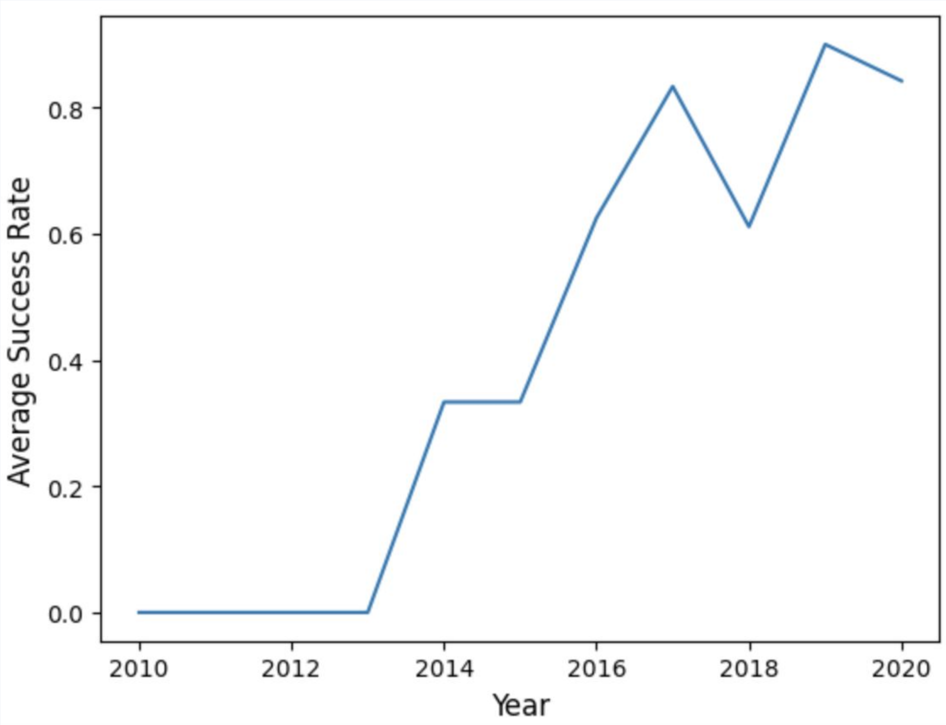
LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

Payload vs. Orbit Type



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 (unsuccessful mission that’s failed), which can be seen in this chart.

Launch Success Yearly Trend



We can observe that the success rate since 2013 kept increasing until 2020.

All Launch Site Names

- **SQL QUERY**
  - SELECT DISTINCT Launch_Site From SPACEXTBL

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

Query Explanation:

- Using the word ***DISTINCT*** in the query means that it will only show unique values in the ***Launch_Site*** column from ***SPACEXTBL***

Launch Site Names Begin with 'CCA'

- **SQL QUERY**

SELECT * from SPACEXTBL where (LAUNCH_SITE) LIKE ‘CCA%’ LIMIT 5

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

Query Explanation:

- Using the word ***limit5*** in the query means that it will only show 5 records from ***SPACEXTBL*** and ***LIKE*** keyword has a wild card to suggest that the Launch_Site name must start with OCA.

Total Payload Mass

- SQL Query

SELECT SUM(PAYLOAD_MASS_KG) FROM SPACEXTBL WHERE Customer = ‘NASA (CRS)’

1
45596

Query Explanation:

Calculate the total payload carried by boosters from NASA, using the function ***SUM arrive the*** total in the column ***PAYLOAD_MASS_KG.*** The ***WHERE*** clause filters the dataset to only perform calculations on ***Customer NASA (CRS)***.

Average Payload Mass by F9 v1.1

- SQL Query

SELECT AVG(PAYLOAD_MASS_KG_) FROM SPACEXTBL WHERE Booster_Version = ‘F9 v1.1’

1
2928

Query Explanation:

Using the function ***AVG*** computes the average in the column ***PAYLOAD_MASS_KG_***. The ***WHERE*** clause filters the dataset to only perform calculations on ***Booster_version F9 v1.1.***

First Successful Ground Landing Date

- SQL Query

SELECT MIN(Date) FROM SPACEXTBL WHERE Landing_Outcome = ‘Success (ground pad)’

1
2015-12-22

Query Explanation:

Using the function ***MIN()*** computes the minimum date in the column Date. The ***WHERE*** clause filters the dataset to only perform calculations on ***Landing_Outcome Success (ground pad)***.

Successful Drone Ship Landing with Payload between 4000 and 6000

- SQL Query

SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE “LANDING_OUTCOME” = ‘Success (drone ship)’ AND 4000 < PAYLOAD_MASS_ < 6000

booster_version
F9 FT B1021.1
F9 FT B1023.1
F9 FT B1029.2
F9 FT B1038.1
F9 B4 B1042.1
F9 B4 B1045.1
F9 B5 B1046.1

Query Explanation:

Selecting Booster_Version column

The ***WHERE*** clause filters the dataset to Landing_Outcome = Success (drone ship), and the ***AND*** clause specifies additional filter conditions: Payload_Mass_KG_ > 4000 AND Payload_Mass_KG < 6000.

Total Number of Successful and Failure Mission Outcomes

- SQL Query

SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TOTAL_NUMBER FROM SPACEXTBL GROUP BY MISSION_OUTCOME

mission_outcome	total_number
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

Query Explanation:

Select Mission_Outcome column, Count() function is used to get the total number of each mission outcome, and Group by to summarize the results.

Boosters Carried Maximum Payload

- SQL Query

SELECT DISTINCT BOOSTER_VERSION

FROM SPACEXTBL

WHERE PAYLOAD_MASS_KG_ = (SELECT MAX (PAYLOAD_MASS_KG_) FROM SPACEXTBL

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

Query Explanation:

Using the word ***DISTINCT*** in the query means that it will only show unique values in the Booster_version column from ***SPACEXTBL SUBQUERY*** in which the ***WHERE*** clause is used to filter the result to a certain condition maximum payload.

2015 Launch Records

- SQL Query

SELECT LANDING_OUTCOME, BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE Landing_Outcome = ‘Failure (drone ship)’ AND YEAR(DATE) = 2015

landing__outcome	booster_version	launch_site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

Query Explanation:

The ***WHERE*** and ***AND*** clause filter the result to 2 conditions: “Landing_Outcome = ‘Failure (drone ship)’” and YEAR in 2015.

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

- SQL Query

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

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

Query Explanation:

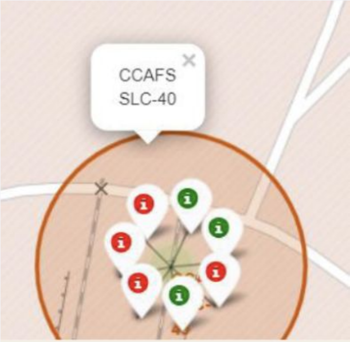
Count() function counts the total “***landing_outcome***” column. The ***WHERE*** and ***AND*** clauses filter the result by 2 conditions. ***Group by*** to summarize the data with landing_outcome, and ***Order by*** ranking the data.

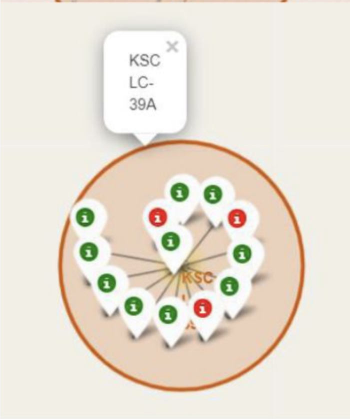
Section
3

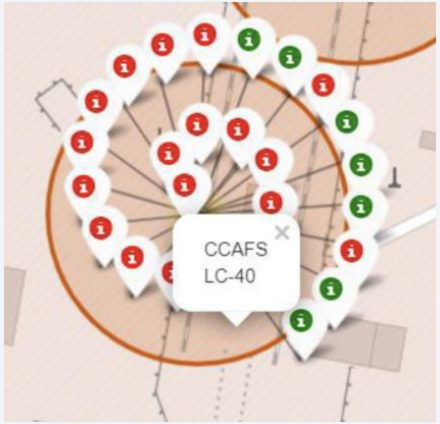
Launch Sites Proximities Analysis

All launch sites on a map

Color Labelled Markers







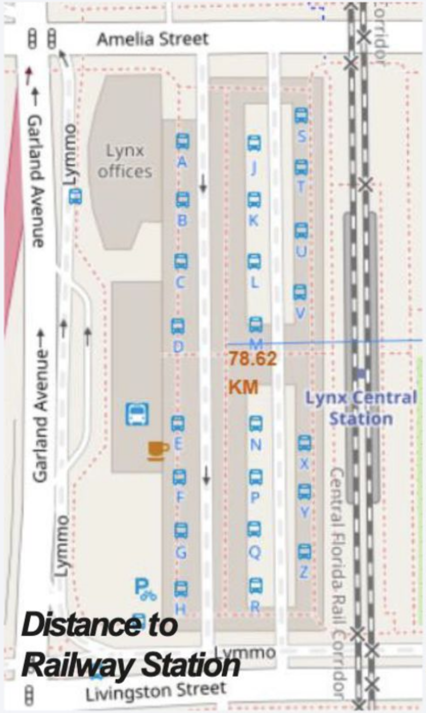
Florida Launch Sites

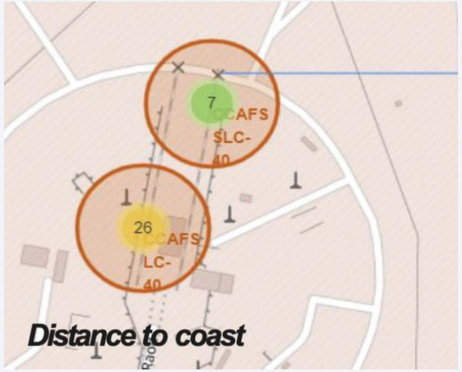
- Green Marker shows successful launches and Red Marker shows failures.

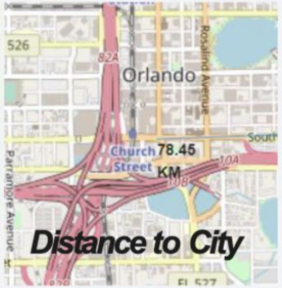
California Launch Sites

- Green Marker shows successful launches and Red Marker shows failure

Launch sites distance to landmark its proximities







- Are launch sites in close proximity to railways? **No**
- Are launch sites in close proximity to highways? **No**
- Are launch sites in close proximity to coastline? **Yes**
- Do launch sites keep certain distance away from cities? **Yes**

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

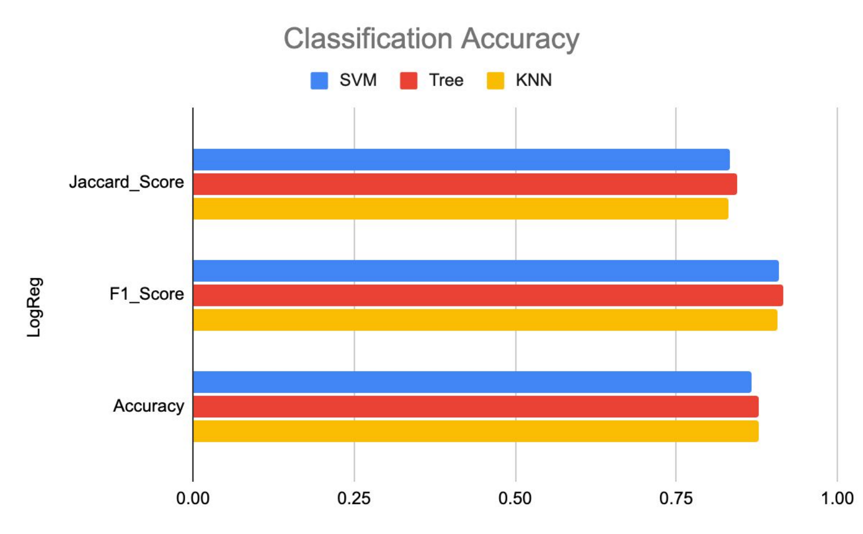
- As you can see, the accuracy is very close and Tree Aligorithm has slightly higher accuracy than the others.

Test Dataset

	LogReg	SVM	Tree	KNN
Jaccard_Score	0.800000	0.800000	0.800000	0.800000
F1_Score	0.888889	0.888889	0.888889	0.888889
Accuracy	0.833333	0.833333	0.888889	0.833333

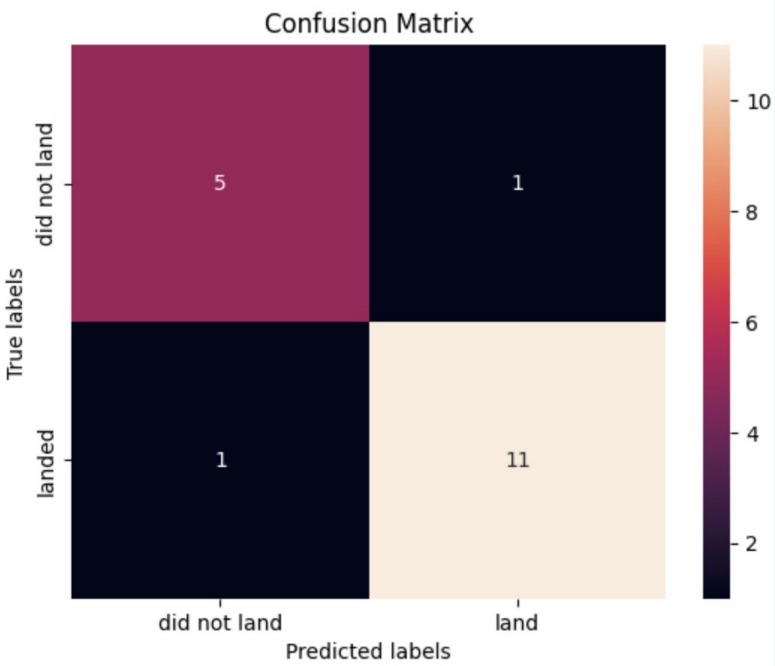
Whole Dataset

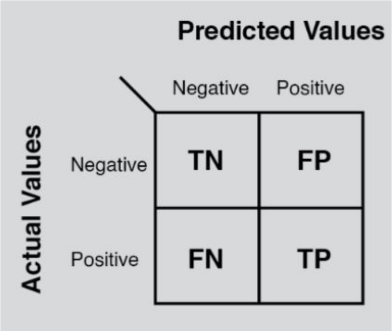
	LogReg	SVM	Tree	KNN
Jaccard_Score	0.833333	0.845070	0.830769	0.819444
F1_Score	0.909091	0.916031	0.907563	0.900763
Accuracy	0.866667	0.877778	0.877778	0.855556



Confusion Matrix

- Examining the confusion matrix, we see that Tree can distinguish between different classes. In this matrix, there is a major issue of false positives.





🚀Conclusions🚀

- The Tree Classifier Algorithm is the slightly better for Machine Learning for training dataset, all model perform 83% accuracy on test data.
- Low weighted payloads perform better than the heavier payloads.
- The success rates for SpaceX launches is directly proportional time in years they will eventually perfect the launches.
- We can see that KSCLC-39A had the most successful launches from all the sites.
- Orbit type GEO,HEO,SSO,ESL1 has the best Success Rate.

🚀Appendix🚀

- All Juypter Notebooks can be found in my Git repo: [LenaMunad-IBM-Data-Science-Capstone-Project.git]

Haversine formula is used in Folium map calculating distances

Introduction

❖ The haversine formula determines the great-circle distance between two points on a sphere given their longitudes and latitudes. Important in navigation, it is a special case of a more general formula in spherical trigonometry, the law of haversines, that relates the sides and angles of spherical triangles.

Usage?

❖ Why did I use this formula? For integrating my ADGGoogle Maps API with a Python function to calculate the distance using two distinct sets of (longitudinal, latitudinal) list sets. Haversine was the trigonometric solution to solve this.

Formula

$$a = \sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos\varphi 1 \cdot \cos\varphi 2 \cdot \sin^2\left(\frac{\Delta\lambda}{2}\right)$$

$$c = 2 \cdot \operatorname{atan2}(\sqrt{a}, \sqrt{(1-a)})$$

$$d = R \cdot c$$

Thank You!

Come again soon! :)