Winning Space Race with Data Science

Ajogwu Salifu
13/01/2024
sjogwu@gmail.com

Outline

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

Executive Summary

The objective of this study is to examine SpaceX Falcon 9 data gathered from diverse sources and leverage Machine Learning models to forecast the success of the first stage landing. This insight is crucial for other space agencies contemplating whether to compete with SpaceX.

Summary of methodologies

Various approaches and techniques were applied throughout the research process, encompassing data collection through API and Web scraping, data transformation via effective wrangling, exploratory data analysis conducted using SQL and data visualizations. An interactive map was constructed using Folium to assess launch site proximity, and a dynamic dashboard was developed with Plotly Dash. The culmination involved building a predictive model to determine the likelihood of a successful landing for the Falcon 9's first stage.

Summary of all results

The findings will be presented through diverse channels, including:
-Data analysis outcomes
-Visual representations and interactive dashboards.
- Analytical results derived from predictive model analysis.

This comprehensive report aims to provide insights into SpaceX Falcon 9 data, facilitating a nuanced understanding of launch success predictors and offering a range of visualization tools for a more interactive exploration of the results.

Introduction

- Project background and context

The commercial space industry is experiencing a transformative era, marked by companies making space travel more accessible. Key players, such as Virgin Galactic, Rocket Lab, Blue Origin, and notably SpaceX, have been instrumental in advancing space exploration. SpaceX, in particular, has achieved remarkable milestones, including sending spacecraft to the International Space Station (ISS), launching Starlink satellite constellations for global internet access, and conducting manned space missions. SpaceX's unique cost advantage lies in its ability to reuse the first stage of its Falcon 9 rocket.

- Problems you want to find answers

- The cost-effectiveness of SpaceX's launches is closely tied to the successful landing and reuse of the first stage. The problem is to predict whether the first stage will land successfully, enabling accurate cost estimates for each launch.
- Analyzing the influence of various parameters and variables on the landing outcomes of SpaceX's Falcon 9 like launch site locations, payload mass, booster versions, and other variables contribute to the success or failure of first stage landings.
- Investigating relationship between specific launch sites and the success rates of Falcon 9 launches. Understanding whether certain launch sites exhibit higher success rates and identifying any geographical or operational factors that contribute to these trends.

Section 1

Methodology

Methodology

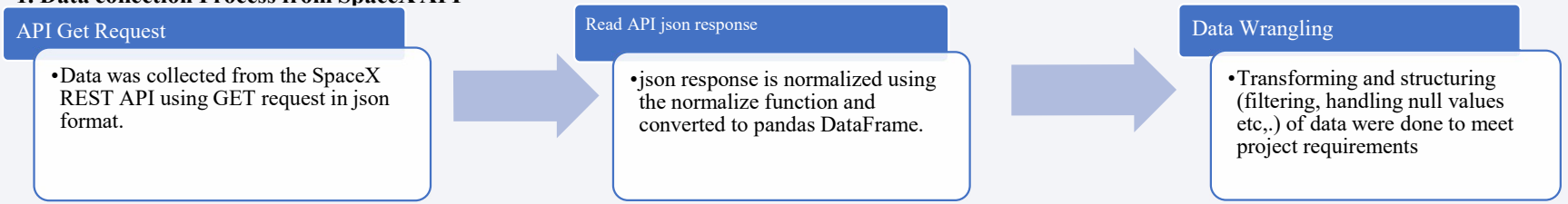
Executive Summary

- Data collection methodology:
  - Data was sourced from SpaceX API.
  - Web scraping was performed on falcon9 launch records on Wikipedia, parsed and converted into a pandas data frame. (https://en.wikipedia.org/wiki/ListofFalcon9andFalconHeavylaunches)
  - Perform data wrangling
  - Exploratory Data Analysis (EDA) was carried out to find some patterns in the data and determine what would be the label for training supervised models; where 1 means the booster landed successfully and 0 unsuccessfully.
- Exploratory data analysis (EDA) was performed on the dataset using visualization and SQL
- Interactive visual analytics using Folium and Plotly Dash was performed
- Predictive analysis using classification models done:
  - A machine learning pipeline engineered (using Logistic Regression, SVM and Decision Tree) to predict if the first stage is successful or not using standardized and transformed data while incorporating train/test split to determine the most suitable classification algorithm on test data.

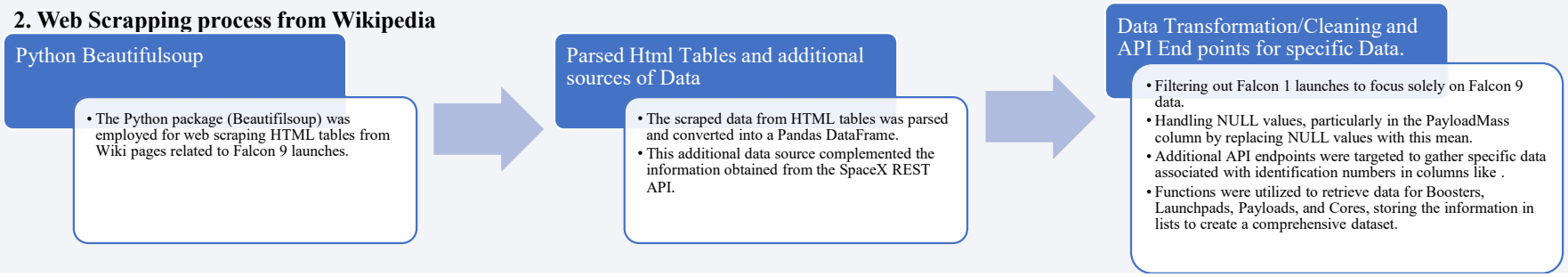
Data Collection

Data was collected from the SpaceX REST API endpoint `api.spacexdata.com/v4/launches/past`. A GET request was made to this endpoint using the `requests` library, obtaining launch data in JSON format. The JSON response, representing past launches, was normalized into a flat table using the `json_normalize` function. The resulting table-form JSON was converted into a Pandas DataFrame for further analysis.
The Python `BeautifulSoup` package was also employed for web scraping HTML tables from Wiki pages related to Falcon 9 launches.The scraped data from HTML tables was parsed and converted into a Pandas DataFrame. This additional data source complemented the information obtained from the SpaceX REST API.
Data wrangling was performed to filter out Falcon 1 launches to focus solely on Falcon 9 data, handle NULL values, particularly in the `PayloadMass` column and issue related to the `LandingPad` column
Additional API endpoints (`api.spacexdata.com/v4/booster`, `api.spacexdata.com/v4/launchpad`, `api.spacexdata.com/v4/payload`, `api.spacexdata.com/v4/core`) were targeted to gather specific data associated with identification numbers in columns like `rocket`.
Functions were utilized to retrieve data for Boosters, Launchpads, Payloads, and Cores, storing the information in lists to create a comprehensive dataset

1. Data collection Process from SpaceX API



2. Web Scrapping process from Wikipedia



Data Collection – SpaceX API

1. **Make a Get request using 'request library', normalize data using 'json normalize', read response as a DataFrame:**

- spacex_url=https://api.spacexdata.com/v4/launches/past
- response = requests.get(spacex_url)
-data = pd.json_normalize(response.json())

2. Reuse the API again to get information about the launches using the IDs given for each launch Specifically (rocket, payloads, launchpad, and cores) store in list and create anew DataFrame representing Global Variables:

```
#Global variables
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

3. Apply getBoosterVersion function method to get the booster version for relevant data:

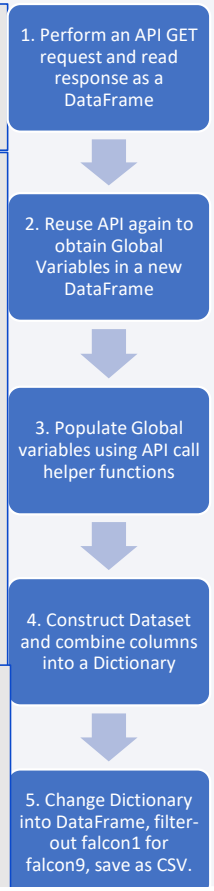
-getBoosterVersion(data)
-getLaunchSite(data)
-getPayloadData(data)
-getCoreData(data)

4. Construct our dataset using the data we have obtained and combine the columns into a dictionary:

```
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}
```

5. Create a data frame from the dictionary and filter using BoosterVersion column to keep falcon9 launches only.

-data_launch = pd.DataFrame(launch_dict)
-data_falcon9 = data_launch[data_launch['BoosterVersion']!= 'Falcon 1']
-data_falcon9.to_csv('dataset_part_1.csv', index=False)



Data Collection - Scraping

1. Conduct a GET request to retrieve falcon9 launch data from HTML:

```
-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)
```

2. Create a BeautifulSoup object from the HTML response:

```
-soup = BeautifulSoup(response.text, 'html.parser')
-page_title = soup.title.text
print(f"Page Title: {page_title}")
```

3. Collect all relevant column names from the HTML table header:

```
- html_tables = soup.find_all('table')
- column_names = []
- th_elements = first_launch_table.find_all('th')
for th in th_elements:
        name = th.get_text(strip=True)
        if name is not None and len(name) > 0:
            column_names.append(name)
print("Column Names:", column_names)
```

4. Construct an empty dictionary with keys from the extracted column names:

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
launch_dict.pop('Date and time ( )', None)

# 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']=[]
```

5. Utilize code snippet to help you to fill up the launch_dict with records:

```
- time = datatimelist[1]
          launch_dict['Time'].append(time)
          print(time)
-  bv=booster_version(row[1])
          if not(bv):
              bv=row[1].a.string
          print(bv)
-  payload_mass = get_mass(row[4])
          launch_dict['Payload mass'].append(payload_mass)
-  booster_landing = landing_status(row[8])
          launch_dict['Booster landing'].append(booster_landing)
          print(booster_landing)
```

6. Create DataFrame from Parsed Launch records:

```
- df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
- df.to_csv('spacex_web_scraped.csv', index=False)
```

1. Perform a HTTP GET method to request the Falcon9 Launch HTML page.
2. Create a BeautifulSoup object from the HTML response
3. Collect all relevant column names from the HTML table header
4. Create a data frame by parsing the launch HTML tables
5. Use of helper function to fill up Dictionary and save as a DataFrame.
6. Convert Launch Dictionary into a DataFrame and save as CSV

Data Wrangling

Exploratory Data Analysis (EDA) was performed to find some patterns in the data and determine what would be the label for training supervised models. 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, where;

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

We eventually converted those outcomes into Training Labels with 1 means the booster successfully landed 0 means it was unsuccessful.

Data Wrangling Flow Chart

Load SpaceX dataset, from last section → Perform wrangling to discover attributes of Dataset → Create a landing outcome label and determine success rate

1. Load SpaceX Dataset:

- df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_1.csv")

2. Discover patterns and attributes in Dataset:

a. Calculate the number of launches on each site;

- launch_site_counts = df['LaunchSite'].value_counts()

Number of launches on each site:

CCAFS SLC 40 55
KSC LC 39A 22
VAFB SLC 4E 13

b. Calculate the number and occurrence of each orbit:

orbit_counts = df['Orbit'].value_counts()

Number and occurrence of each orbit:

GTO 27
ISS 21
VLEO 14
PO 9
LEO 7
SSO 5
MEO 3
ES-L1 1
HEO 1
SO 1
GEO 1

c. Calculate the number and occurence of mission outcome of the orbits:

landing_outcomes = df['Outcome'].value_counts()

3. Create a landing outcome label from Outcome column:

a. landing_class = [0 if outcome in bad_outcomes else 1 for outcome in df['Outcome']]

b. df['Class']=landing_class
df[['Class']].head(8);

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

c. Determine success rate:

- df['Class']=landing_class;

0.6666666666666666

EDA with Data Visualization

To gain further insights from the SpaceX dataset, Explorative Data Analysis (EDA) was carried out plotting the following charts:

- Scatter Plot:

A scatter plot is used to visually represent the relationship between two continuous variables, revealing patterns, trends, and potential correlations in the data. Scatter plots where used to determine the following:

- Correlation between Flight Number and Launch Site
- Correlation between Payload and Launch Site
- Correlation between Flight Number and Orbit Type
- Correlation between Payload and Orbit Type

- Bar Chart:

A bar chart is used to visually represent and compare the quantities or values of different categories or groups. It consists of rectangular bars, where the length of each bar corresponds to the value it represents, allowing for easy interpretation and comparison of data across categories.

- Bar Plot was constructed to examine the relationship between orbit and success rate.

Line Chart:

A line chart is a graphical representation that displays data points or values over a continuous interval using connected data points with straight lines. It is commonly used to illustrate trends and patterns in data, showing how a variable changes over time or across different conditions. Line charts are effective for visualizing the relationships and fluctuations in data points, making them valuable for time-series analysis and identifying trends or correlations.

- A line chart was plotted to get the average launch success trend.

EDA with SQL

To gain more insights into the SpaceX dataset, we performed SQL queries of the Dataset loaded into an IBM Db2 Database. The following is an overview of the operations carried out:

- **We first loaded the SQL extension and established a connection with the database.**
- **Display the names of the unique launch sites in the space mission.**
- **Show 5 entries where the launch sites start with 'CCA'.**
- **Show the total payload mass transported by NASA (CRS) boosters.**
- **Display the average payload mass transported by the booster version F9 v1.1.**
- **Provide the dates of the first successful ground pad landing outcomes.**
- **List the names of boosters with success in drone ship landings and payload mass greater than 4000 but less than 6000.**
- **Display the total number of successful and failed mission outcomes.**
- **List the names of booster versions that carried the maximum payload mass, using a subquery.**
- **Show records indicating month names, failure landing outcomes in drone ships, booster versions, and launch sites for the months in the year 2015.**
- **Rank the count of landing outcomes [such as Failure (drone ship) or Success (ground pad)] between the dates 2010-06-04 and 2017-03-20 in descending order.**

Build an Interactive Map with Folium

Folium is a Python library used for creating interactive maps, allowing users to visualize geographical data in an interactive and dynamic manner. It leverages the Leaflet.js library and enables the incorporation of various map elements, such as markers, circles, and polygons, providing a versatile tool for geographical data exploration and analysis.

The following map objects with its accompanying functions were added to the Map:

- Markers were attached (using folium circles and markers) to a launch sites to help highlight and locate launch sites on the map.
- . Mark_cluster objects were created to classify launch outcome where green is equal to Success and Red failure for all launch sites

We calculated the distances between a launch sites to its proximities(Railways, coastal lines, highways and cities):

-MousePosition was added on the map to get coordinate for a mouse over a point on the map making it easily find the coordinates of any points of interests (such as railway)
- Folium.marker was used to calculate launch site distance to its proximities.
- A folium PolyLine between a launch site to the selected coastline point was drawn.
- Create a marker with distance to a closest city, railway, highway, etc. and draw a line between the marker to the launch site.

The folium Map enabled us to answer whether launch site locations are in close proximity to railways, coastal lines, highways and cities.

Build a Dashboard with Plotly Dash

A Plotly Dash Dashboard Application was built to perform interactive visual analysis in real-time. This dashboard application contains input components such as a dropdown list and a range slider to interact with a pie chart and a scatter point chart.

- We added the plots and interactions to meet the following objectives:

- We added a Launch Site Drop-down Input Component to see which one has the largest success count and be able to select different launch sites.

- We added a callback function to render success-pie-chart based on selected site dropdown. This callback function is to get the selected launch site from site-dropdown and render a pie chart visualizing launch success counts.

- A Range Slider was added to Select Payload and find if the variables payload is correlated to mission outcome. From a dashboard point of view, we were able to select different payload range and see if we can identify some visual patterns.

- A callback function was incorporated to render the success-payload-scatter-chart scatter plot to plot a scatter plot with the x axis to be the payload and the y axis to be the launch outcome (i.e., class column). As such, we are able to visually observe how payload may be correlated with mission outcomes for selected site(s).

- With the dashboard completed, we are be able to use it to analyze SpaceX launch data, and answer the following questions:

- Which site has the largest successful launches?

- Which site has the highest launch success rate?

- Which payload range(s) has the highest launch success rate?

- Which payload range(s) has the lowest launch success rate?

- Which F9 Booster version (v1.0, v1.1, FT, B4, B5, etc.) has the highest

- launch success rate?

Predictive Analysis (Classification)

Exploratory Data Analysis was performed and determined Training Labels by first creating a column for the class, standardizing the data and splitting dataset into training data and test data

We then found best Hyperparameter for SVM, Classification Trees and Logistic Regression by finding the method that performs best using test data

Load Dataset and Create a NumPy array from the column Class → Standardize the data in X then reassign → Split the data X and Y into training and test data → Finetune model parameters → Best performing model

1. After loading the dataset Create a NumPy array from the column Class in data, by applying the method to_numpy() then assign it to the variable Y, make sure the output is a Pandas series (only one bracket df['name of column']).

```
- Y = data['Class'].to_numpy()
# Verify the type of Y
print(type(Y))
```

2. Standardize the data in X then reassign it to the variable X using the transform method:

```
- # Standardize the data in X
transform = preprocessing.StandardScaler()
X = transform.fit_transform(X)
# Split the data into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
```

3. Use the function train_test_split to split the data X and Y into training and test data:

```
- X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
```

4. Create a logistic regression object then create a GridSearchCV object logreg_cv and fit to best parmeter:

```
- parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge.
- logreg_cv = GridSearchCV(logreg, parameters, cv=10)
- logreg_cv.fit(X_train, Y_train)
```

Best parameters displays using best_params_ and accuracy validation using best_score_.

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

Calculate the accuracy on the test data using the method score

```
- accuracy_lr = logreg_cv.score(X_test, Y_test)
```

Incorporate confusion matrixs.

```
- yhat=logreg_cv.predict(X_test)
```

plot_confusion_matrix(Y_test,yhat)

Iterate for the other rarameters (SVM, Decision tree and KNN)

5. Find the method performs best:

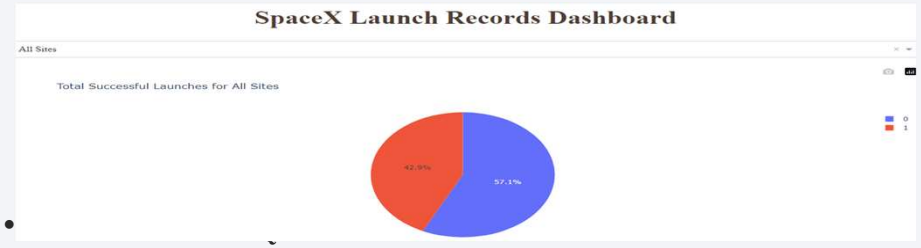
```
# Identify the best-performing model
best_model = max([accuracy_lr, accuracy_svm, accuracy_tree])
# Print the best-performing model
if best_model == accuracy_lr:
    print("The best-performing model is Logistic Regression.")
elif best_model == accuracy_svm:
    print("The best-performing model is Support Vector Machine.")
else:
    print("The best-performing model is Decision Tree.")
```

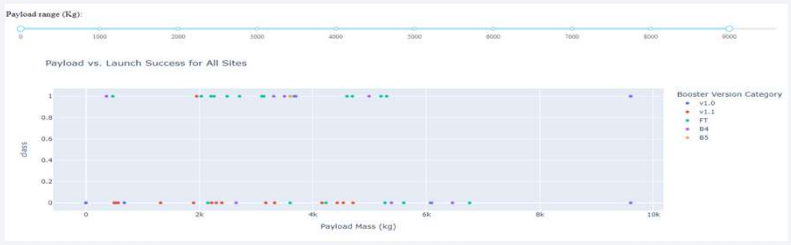
Results

- **Exploratory data analysis results**

After plotting the FlightNumber vs. PayloadMass and overlay the outcome of the launch. Is visible, that as the flight number increases, the first stage is more likely to land successfully. It seems the more massive the payload, the less likely the first stage will return.
We see that different launch sites have different success rates. CCAFS LC-40, has a success rate of 60 %, while KSC LC-39A and VAFB SLC 4E has a success rate of 77%.
After analyzing the Bar chart we see orbit ES-L1, GEO and HEO have a higher success rate than any orbit type.

- **Interactive analytics demo in screenshots**





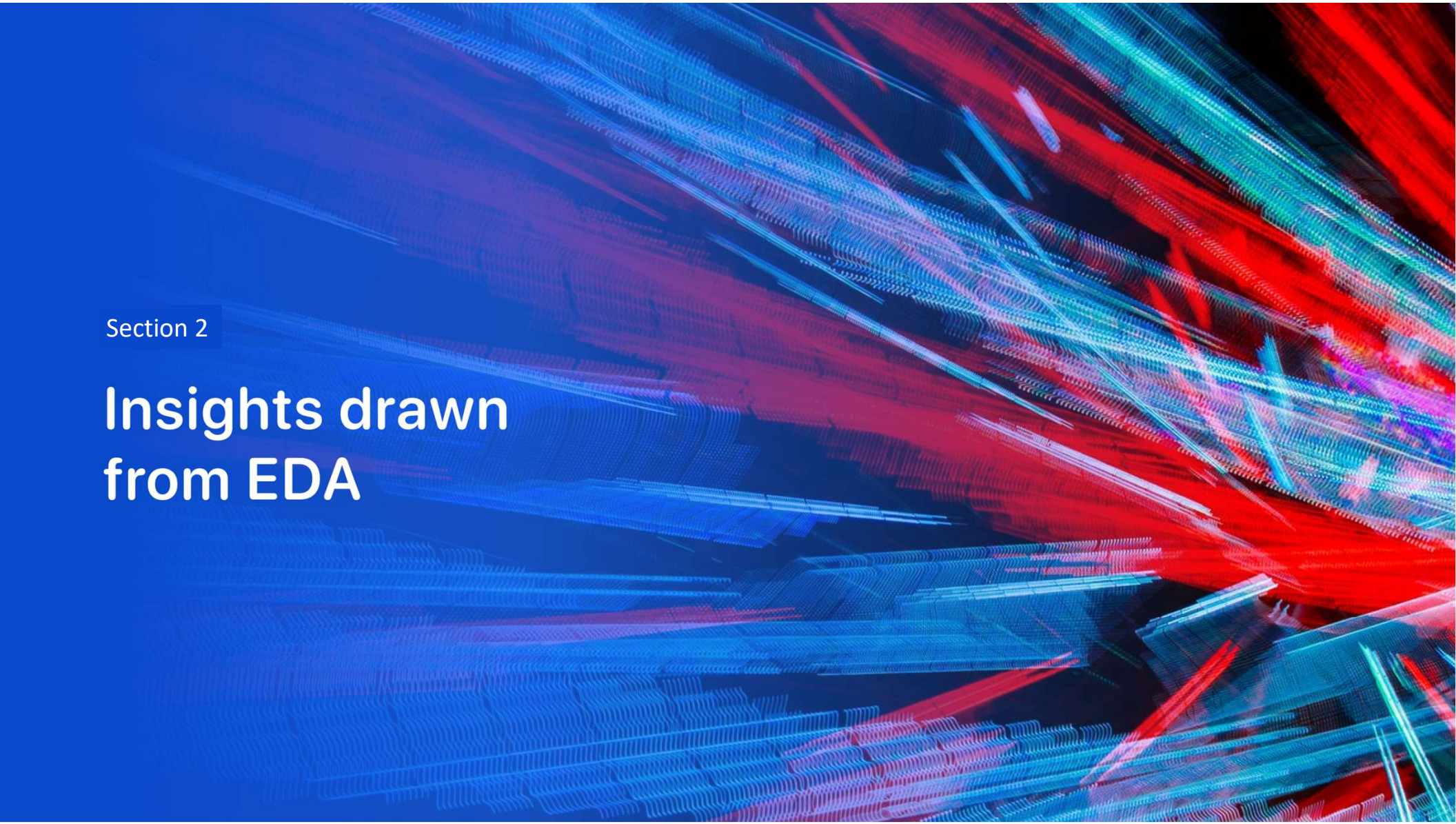
After using the score method to calculate the accuracy launches the following results where gotten:

Accuracy on Test Data (Logistic Regression): 83.33%

Accuracy on Test Data (Support Vector Machine): 83.33%

Accuracy on Test Data (Decision Tree): 72.22%

The best-performing model is Logistic Regression.



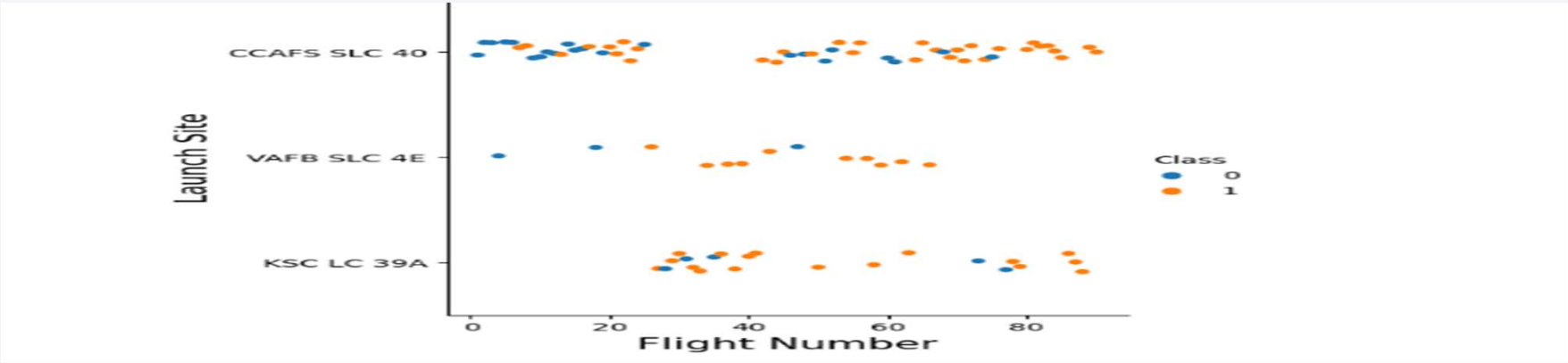
Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

Scatterplot of Flight Number vs. Launch Site

- sns.catplot(y="LaunchSite",x="FlightNumber",hue="Class", data=df, aspect = 1)
- plt.ylabel("Launch Site",fontsize=15)
- plt.xlabel("Flight Number",fontsize=15)
- plt.show()



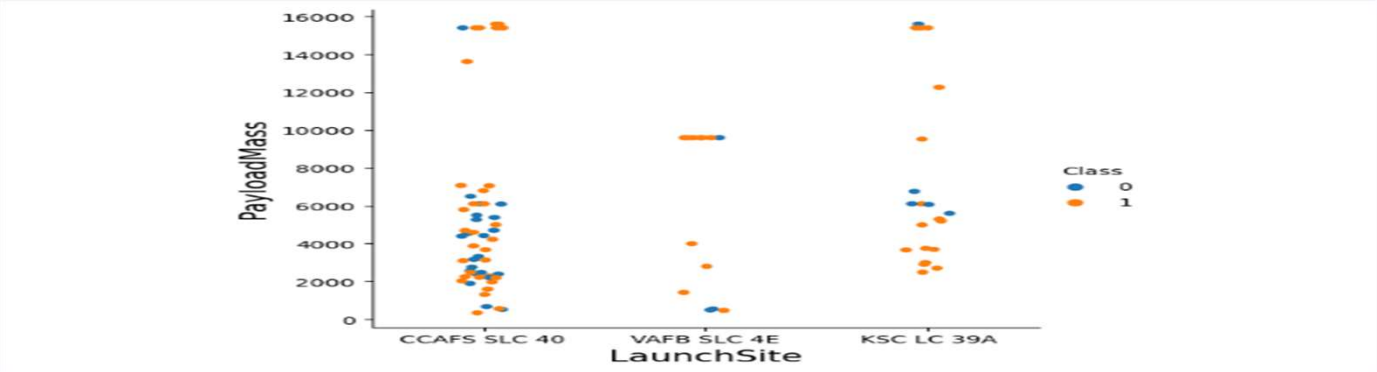
- From the scatterplot it can be inferred that class 1 (representing success rate) increases as Flight Number increases.

- The Scatterplot shows that in the overall flights taken for all the launch sites, there is higher success than failure rate.

Payload vs. Launch Site

Scatterplot showing Payload vs. Launch Site

- sns.catplot(y="PayloadMass",x="LaunchSite",hue="Class", data=df, aspect = 1)
- plt.ylabel("PayloadMass",fontsize=15)
- plt.xlabel("LaunchSite",fontsize=15)
- plt.show()

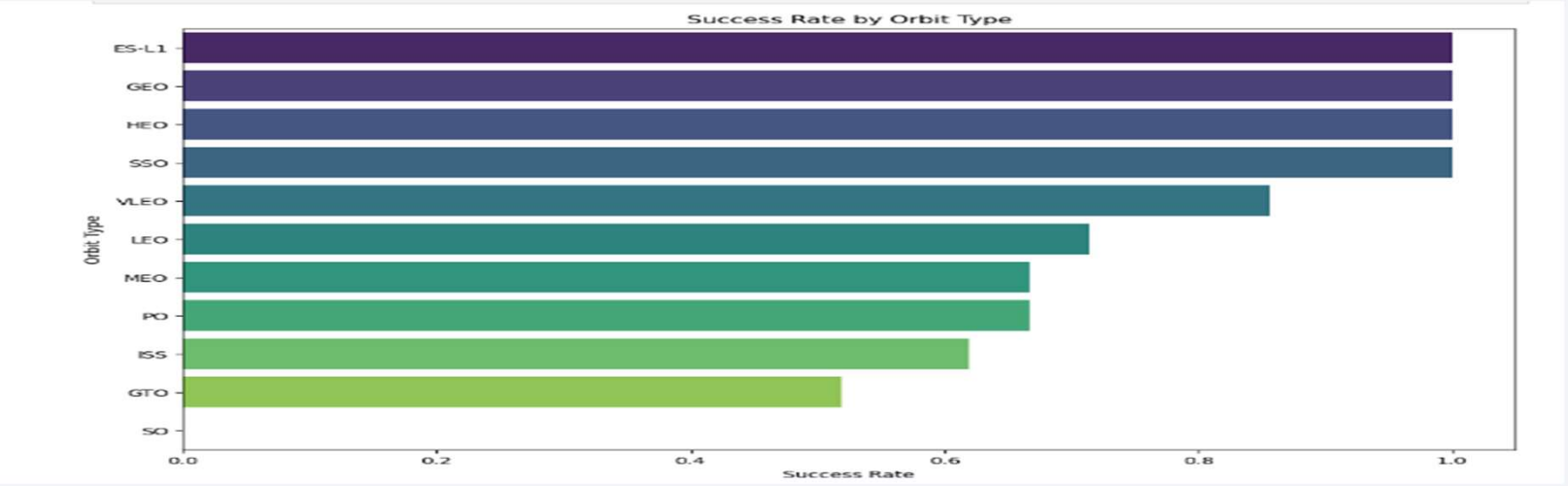


- From the scatterplot we can see that from payload mass greater than 10,000kg success rate increases launch sites KSC LC-39A and CCAFS SLC 40.
- The plot also shows that for VAFB-SLC 4E launch site success rate increases as payload mass increases up to 10,000kg where no rocket is launched.
- Hence there is a clear correlation between size of payload mass and success rate (as payload mass increases success rate increases.

Success Rate vs. Orbit Type

Bar chart showing the success rate of each orbit type:

```
plt.figure(figsize=(12, 8))
success_rate_by_orbit = df.groupby('Orbit')['Class'].mean().sort_values(ascending=False)
sns.barplot(x=success_rate_by_orbit.values, y=success_rate_by_orbit.index, palette='viridis')
plt.title('Success Rate by Orbit Type')
plt.xlabel('Success Rate')
plt.ylabel('Orbit Type')
plt.show()
```

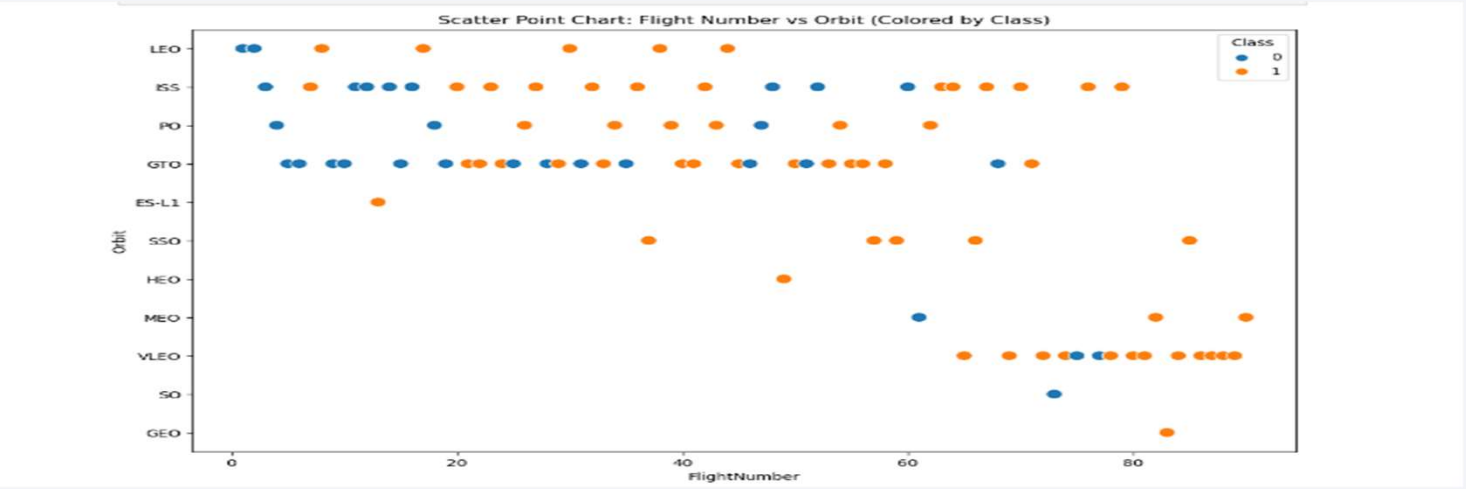


- The bar chart shows that orbits ES-L1, GEO, HEO and SSO has the highest success rate.
- The orbit GTO has the lowest success rate for launches.

Flight Number vs. Orbit Type

Scatterplot of Flight number vs. Orbit type

```
plt.figure(figsize=(12, 8))
sns.scatterplot(x='FlightNumber', y='Orbit', hue='Class', data=df, s=100)
plt.title('Scatter Point Chart: Flight Number vs Orbit (Colored by Class)')
plt.show()
```

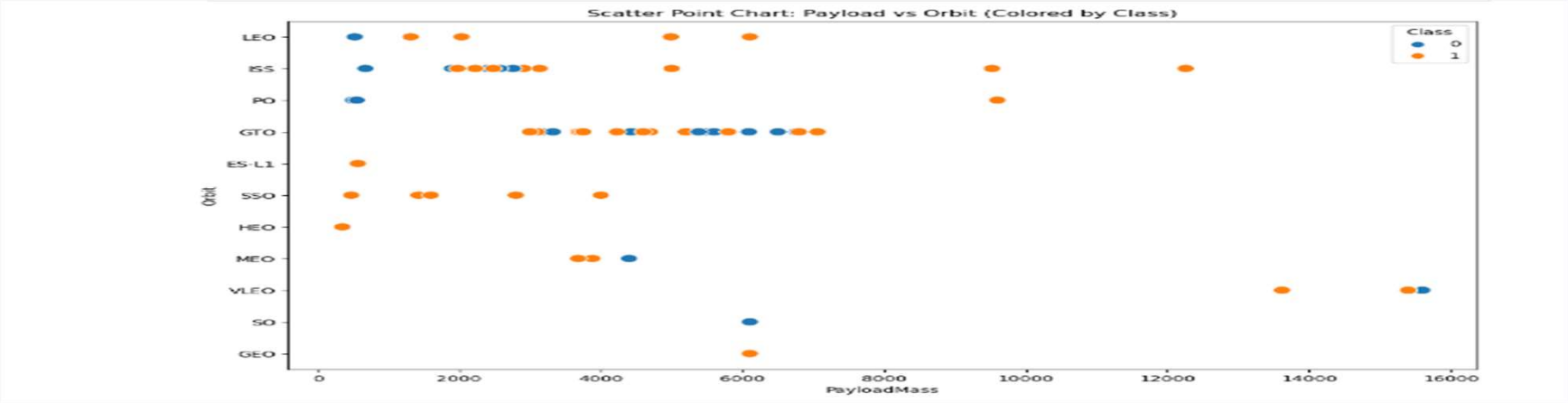


- From the plot we see that VLEO launch outcomes start to occur from above 60 flights.
- Apart for GTO orbit (which exhibits no relationship) there seems to be a direct correlation between successful landing and high flight numbers in most orbits (LEO, ISS, PO, SSO, MEO,VLEO)

Payload vs. Orbit Type

Scatterplot of payload vs. orbit type

```
plt.figure(figsize=(12, 8))
sns.scatterplot(x='PayloadMass', y='Orbit', hue='Class', data=df, s=100)
plt.title('Scatter Point Chart: Payload vs Orbit (Colored by Class)')
plt.show()
```

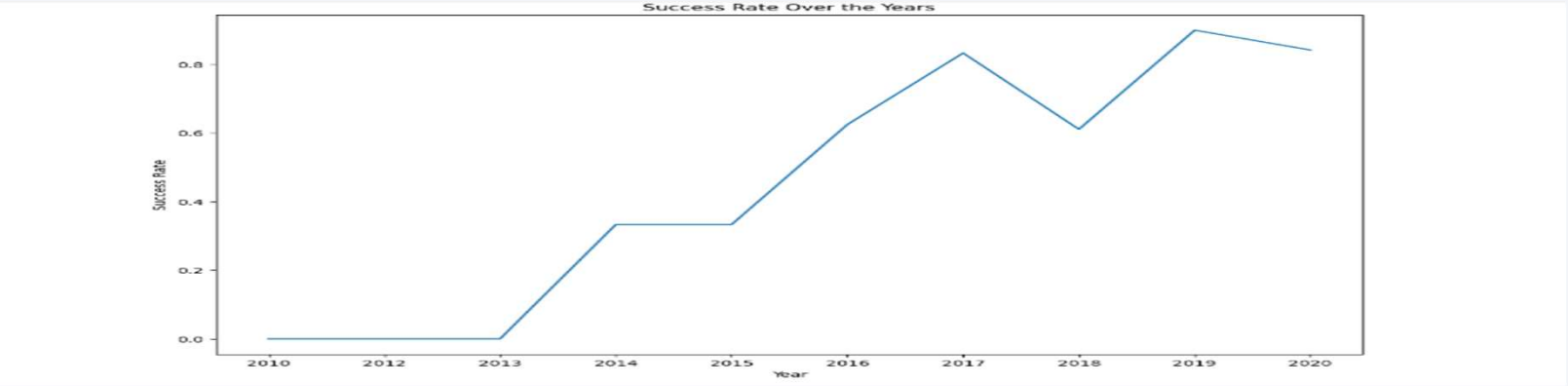


- From the plot there is a positive relationship between heavy payload and successful landing outcomes for orbits LEO, ISS, PO, SSO and VLEO.
- The nature of relationship that exists between payloads successful landings and GTO orbit is obscure and not immediately visible.

Launch Success Yearly Trend

Line chart of yearly average success rate

```
df['Year'] = Extract_year(df['Date'])
# Calculate success rate by year
success_rate_by_year = df.groupby('Year')['Class'].mean()
# Plot a line chart
plt.figure(figsize=(12, 8))
sns.lineplot(x=success_rate_by_year.index, y=success_rate_by_year.values)
plt.title('Success Rate Over the Years')
plt.xlabel('Year')
```



- Success rate was zero from year 2010 to 2013 and started increasing after year 2013 and peaked at 0.4 in year 2014.
- Success rate remained the same from year 2014 to 2015 then peaked again to 0.8 in year 2017 and then decline to 0.6 in the year 2018.
- Success rate peaked to over 0.8 in 2019 however it started to decline up till 2020.

All Launch Site Names

Names of the unique launch sites:

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

SQL Query:

```
%sql SELECT DISTINCT "Launch_Site" FROM SPACEXTABLE;
 * sqlite:///my_data1.db
```

Explanation:

The above query displayed the names of the unique launch sites in the space mission which show 4 launch sites.

Launch Site Names Begin with 'CCA'

Five records where launch sites begin with `CCA`

]:

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

SQL Query

```
%sql SELECT * FROM SPACEXTABLE WHERE "Launch_Site" LIKE 'CCA%' LIMIT 5;
 * sqlite:///my_data1.db
```

Explanation

- Using the Keywords “LIKE” and “CCA%” displays record columns where launch site begins with CCA.
- The Keyword “LIMIT 5” limits the records display to 5.

Total Payload Mass

Total payload carried by boosters from NASA:

Total_Payload_Mass_kg
48213

SQL Query:

```
%sql SELECT SUM("Payload_Mass__kg_") as "Total_Payload_Mass_kg" FROM SPACEXTABLE WHERE "Customer" LIKE 'NASA (CRS)%';
 * sqlite:///my_data1.db
```

Explanation:

The keyword “SUM("Payload_Mass__kg_")” sums up all the payload mass of booster launched by NASA and saves it with the keyword Total_Payload_Mass_kg.

Average Payload Mass by F9 v1.1

Average payload mass carried by booster version F9 v1.1:

Average_Payload_Mass_kg
2928.4

SQL Query:

```
%sql SELECT AVG("Payload_Mass__kg_") as "Average_Payload_Mass_kg" FROM SPACEXTABLE WHERE "Booster_Version" = 'F9 v1.1';
 * sqlite:///my_data1.db
```

Explanation:
The keyword AVG("Payload_Mass__kg_") computes the average payload mass for booster version F9 v1.1 and saves it as Average_Payload_Mass_kg.

First Successful Ground Landing Date

First successful landing outcome on ground pad:

First_Successful_Landing_Date
2015-12-22

SQL Query:

```
%sql SELECT MIN("Date") as "First_Successful_Landing_Date" FROM SPACEXTABLE WHERE "Landing_Outcome" = 'Success (ground pad)';
 * sqlite:///my_data1.db
```

Explanation:

- The Keyword MIN("Date") selects the first successful date for landing outcome from SPACEXTABLE.
- The WHERE "Landing_Outcome" = 'Success (ground pad)' limit limits the success date to that of ground pads.

Successful Drone Ship Landing with Payload between 4000 and 6000

Names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000:

Booster_Version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

SQL Query:

```
%sql SELECT "Booster_Version" FROM SPACEXTABLE WHERE "Landing_Outcome" = 'Success (drone ship)' AND "Payload_Mass__kg_" BETWEEN 4000 AND 6000;
 * sqlite:///my_data1.db
```

Explanation:

- The above query displays Booster_Version from SPACEXTABLE showing the success rate for drone ship according to payload mass range between 4000 and 6000 kg.
- The ' operator in the where clause retrieves booster versions where both conditions specified in the where clause are satisfied.

Total Number of Successful and Failure Mission Outcomes

Total number of successful and failure mission outcomes:

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

SQL Query:

```
%sql SELECT "Mission_Outcome", COUNT(*) as "Total_Missions" FROM SPACEXTABLE GROUP BY "Mission_Outcome";
 * sqlite:///my_data1.db
```

Explanation:

- Mission outcomes are displayed from Total Missions querying SPACEXTABLE activating the GROUP BY method to arrange mission outcome groups (Failure and Success).
- In the table we see a 99 success count, 1 Failure count and a success count with a payload status that is unclear.

Boosters Carried Maximum Payload

Names of the booster which have carried the maximum payload mass:

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:

```
%sql SELECT DISTINCT "Booster_Version" FROM SPACEXTABLE WHERE "Payload_Mass__kg_" = (SELECT MAX("Payload_Mass__kg_") FROM SPACEXTABLE);
 * sqlite:///my_data1.db
```

Explanations:

- The subquery retrieves the maximum payload mass by utilizing the 'MAX' keyword on the payload mass column.
- The main query provides booster versions and their corresponding payload mass, specifically where the payload mass is at its maximum with a value of 15600.

2015 Launch Records

List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015:

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

SQL Query:

```
%sql SELECT SUBSTR("Date", 6, 2) as "Month", "Landing_Outcome", "Booster_Version", "Launch_Site" FROM SPACEXTABLE \
WHERE SUBSTR("Date", 0, 5) = '2015' AND "Landing_Outcome" LIKE '%Failure (drone ship)%';
 * sqlite:///my_data1.db
```

Explanation:

- The SQL query displays the landing outcome, booster version, and launch site where the landing outcome is a failure on a drone ship, in the year 2015.
- The 'AND' operator in the WHERE clause filters booster versions where both conditions in the WHERE clause are met.
- The keyword “YEAR” spotlights the year from the 'Date' column.
- The outcomes reveal the launch site as 'CCAFS LC 40' and booster versions as F9 v1.1 B1012 and B1015, both experiencing failed landing outcomes on a drone ship in the year 2015.

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

Rankings of 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:

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

SQL Query:

```
%sql SELECT "Landing_Outcome", COUNT(*) as "Count" FROM SPACEXTABLE \
WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20' \
GROUP BY "Landing_Outcome" \
ORDER BY "Count" DESC;
```

Explanations:

- This SQL query retrieves the count of different landing outcomes from the "SPACEXTABLE" table where the date is between '2010-06-04' and '2017-03-20'.
- It groups the results by the "Landing_Outcome" column and orders them in descending order based on the count. The result shows the number of occurrences for each landing outcome during the specified date range.

Section 3

Launch Sites Proximities Analysis

Launch Sites Map for Spacex Falcon9

Fig 1: NASA Johnson Space Center

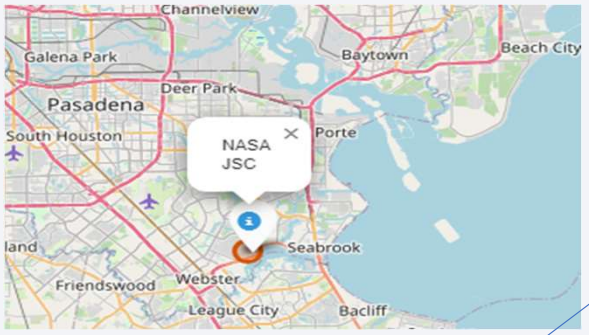


Fig 2: Global map

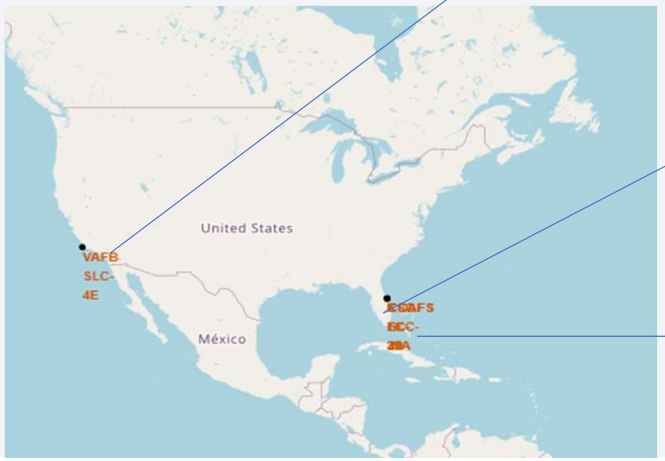


Fig 3: Zoom A

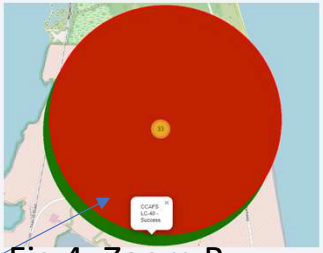


Fig 4: Zoom B

Fig 5: Zoom C

Fig 1 shows an initial center location to be NASA Johnson Space Center at Houston, Texas.

Fig 2 showcases a world map featuring Falcon 9 launch sites situated in the United States, specifically in California and Florida. Each launch site is represented by a circle, accompanied by a label and a popup providing information about the location and name of the launch site. Notably, all the launch sites are positioned in close proximity to the coast.

Figures 3, 4 and 5 focus on the launch sites, showcasing four specific locations:

- VAFB SLC 4E(CL)
- CCAFS SLC 40(FL)
- KSC LC 39A(FL)
- CCAFS SLC 40(FL)

SpaceX Falcon9 launch sites and outcomes

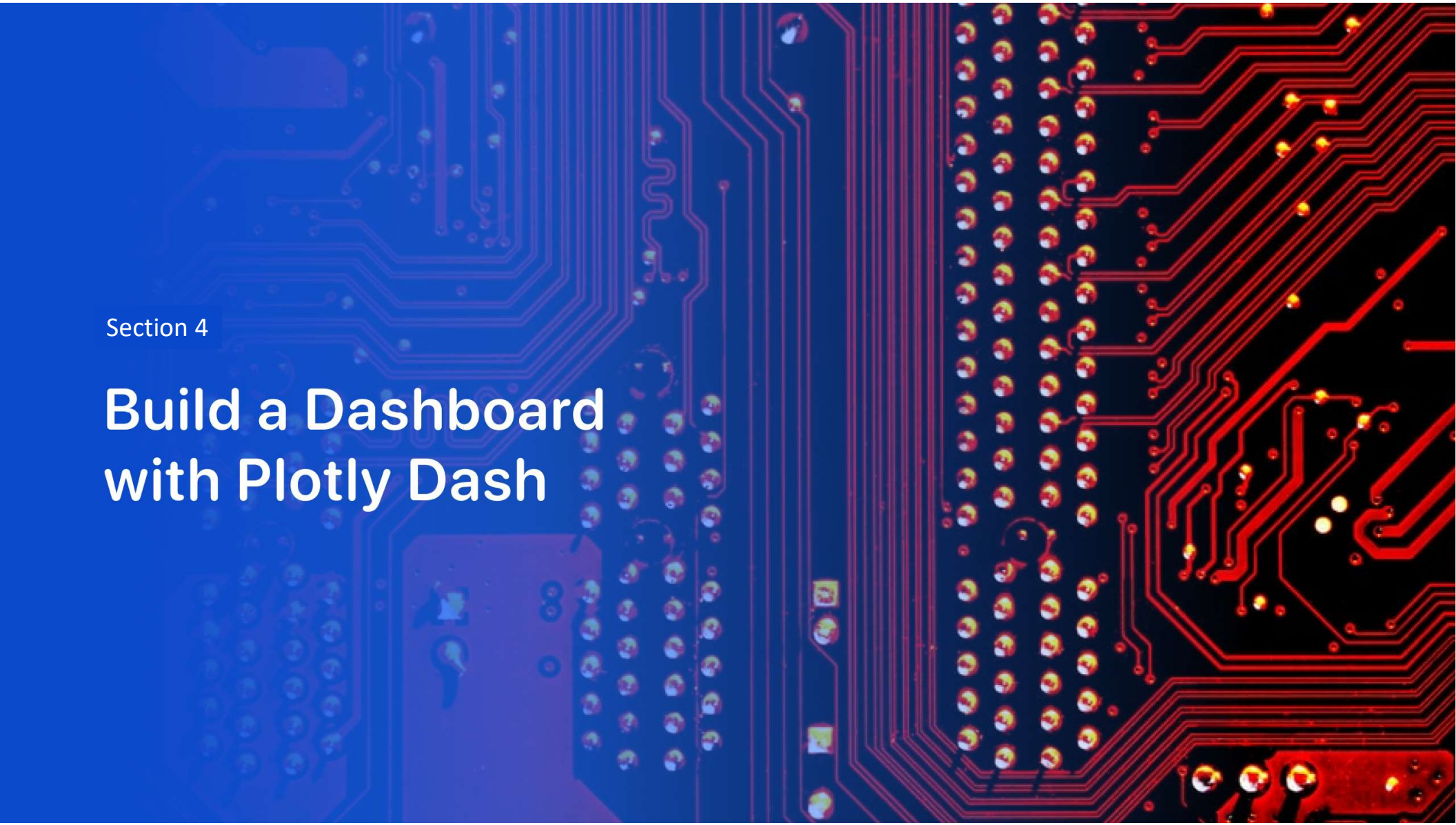
Fig 1 shows VAFB sites with success/failure markers

Fig 2 displays – KSC with success/failure markers

Fig 3 shows – CCAFS with success/failure marker

Fig 3 shows – CCAFS with success/failure marker

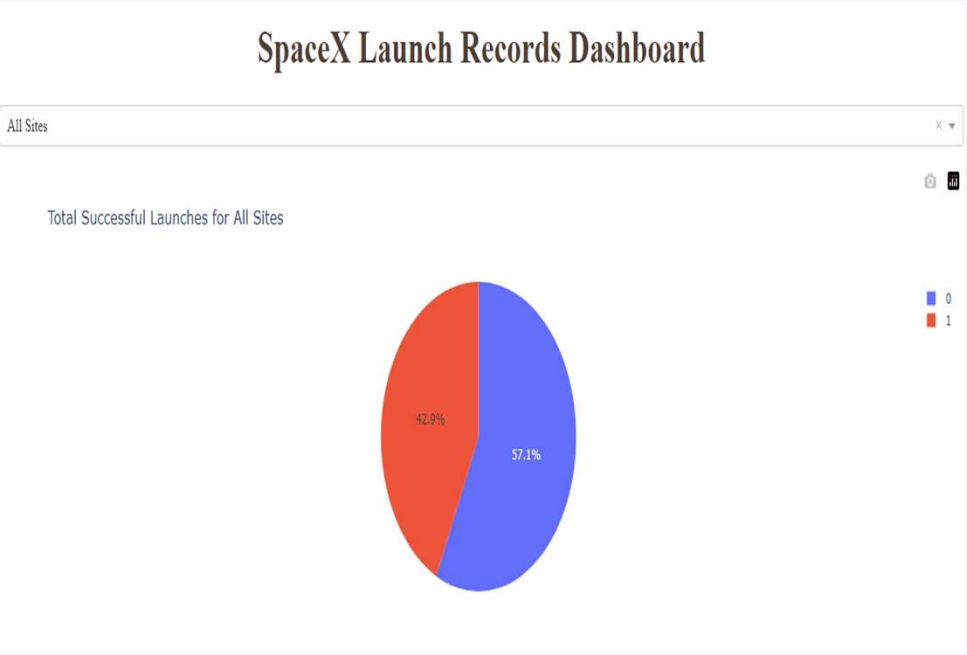
Figures 1, 2, 3, and 4 provide detailed views of each site, indicating success with green markers and failure with red markers. Upon close inspection, the KSC LC 39A Launch Site stands out with the highest number of successful launches.



Section 4

Build a Dashboard with Plotly Dash

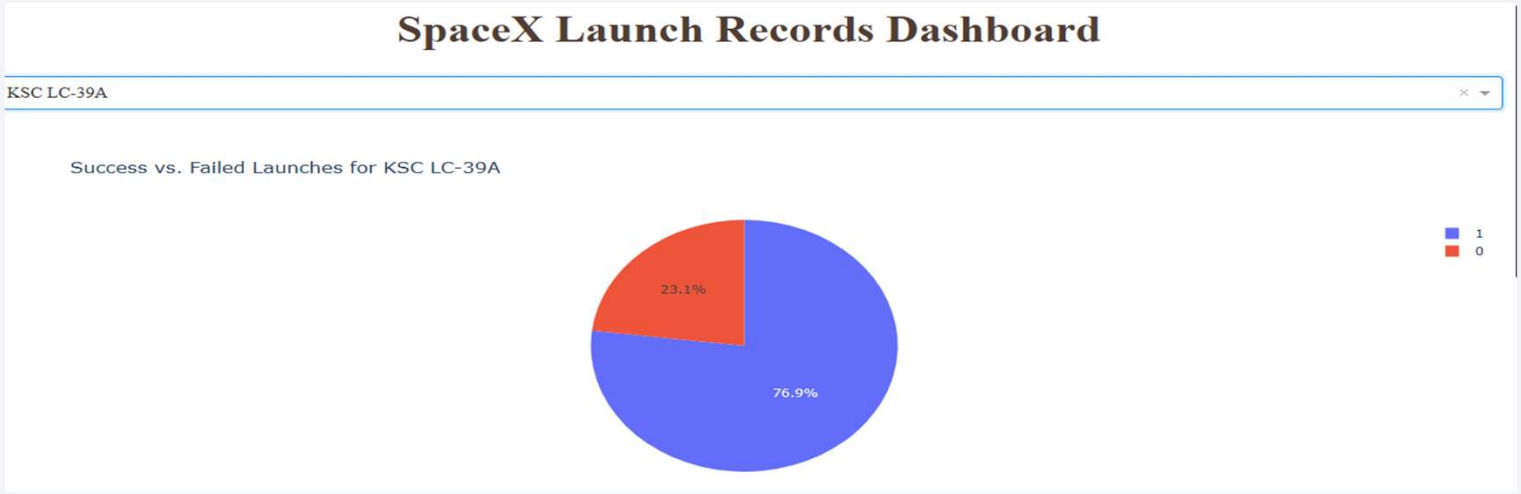
Launch outcomes for all sites (Success/Failure)



The above Figure is an interactive dashboard showing showing SpaceX launch records all launch sites where Blue(0) represents failure and Orange(1) represents success.

The Figure above displays all the launch site for all SpaceX falcon9 launches.

Highest Launch success rate from a launch site



- KSC LC-39A Launch Site has the highest launch success rate and count.
- Launch success to failure rate is 76.9%.
- Launch failure to success rate is 23.1%.

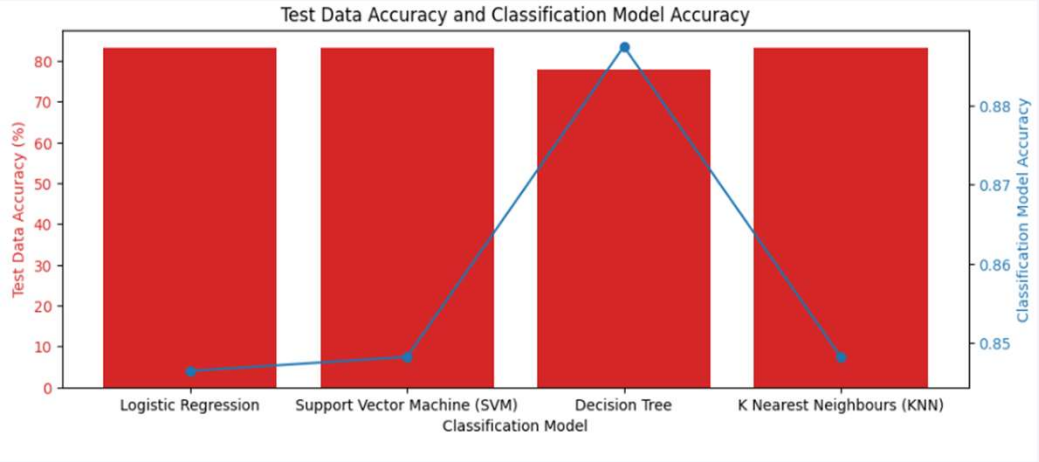
Payload mass vs. launch outcomes for all sites

- The Dashboard shows that as payload mass(kg) increases e.g. (2034kg to 4990kg) the success rate increases.
- The Dashboard also shows that the booster version FT has the highest success rate compared to all other booster versions (v1.0, v1.1, B4 and B5).
- Booster version v1.1 shows the highest incidence of failure rate.

Section 5

Predictive Analysis (Classification)

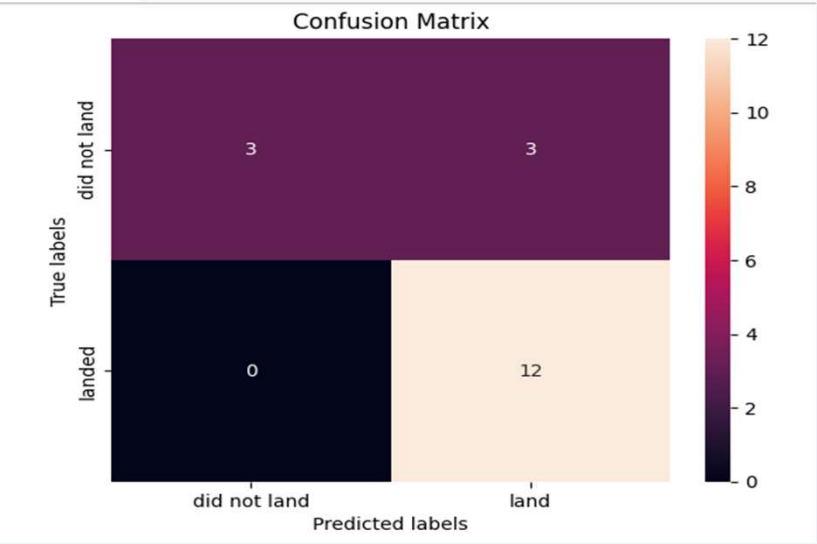
Classification Accuracy



Classification Model	Test Data Accuracy	Classification Model Accuracy
Logistic Regression	83.33%	84.64%
SVM	83.33%	84.82%
Decision Tree	77.78%	88.75%
KNN	83.33%	84.82%

- "Decision Tree" has the lowest test data accuracy (77.78%) but achieves the highest overall classification accuracy (88.75%).
- -"Logistic Regression," "Support Vector Machine (SVM)," and "K Nearest Neighbours (KNN)" share a higher test data accuracy of 83.33%.
- "Logistic Regression" and "SVM" show similar and competitive overall classification accuracies at 84.64% and 84.82%, respectively.
- -"KNN" also performs well with a classification accuracy of 84.82%.
- -"Decision Tree" exhibits a potential issue of overfitting, as indicated by the lower test data accuracy.
- -"KNN" and "SVM" consistently perform well in both test data accuracy and overall classification accuracy.
- "Logistic Regression" provides balanced performance across both metrics.

Confusion Matrix



- From the confusion matrix, we can derive the following insights:
- Accuracy: The overall accuracy of the model is calculated as (TP + TN) / (TP + TN + FP + FN). It represents the proportion of correctly predicted instances out of the total.
- Precision: Precision is calculated as TP / (TP + FP). It measures the accuracy of positive predictions. A higher precision indicates fewer false positives.
- Recall (Sensitivity): Recall is calculated as TP / (TP + FN). It measures the model's ability to capture all the positive instances without missing any (lower false negatives).
- F1-Score: The F1-Score is the harmonic mean of precision and recall, calculated as 2 * (Precision * Recall) / (Precision + Recall). It provides a balance between precision and recall.
- In general, the classifier achieves an accuracy of approximately 83%, correctly predicting instances as either TPs or TNs, out of the total instances. The misclassification or error rate, which includes FN and FN, is around 16.5%.

Conclusions

Flight Success Factors:

- Positive correlation observed between success rates and higher flight numbers.

- Payload mass above 10,000kg associated with increased success rates, varying across launch sites.

Orbit and Launch Site Impact:

- Orbits ES-L1, GEO, HEO, and SSO consistently show high success rates whileGTO experiences consistently lower success rates.

- VLEO launch outcomes become prominent after 60 flights, indicating a correlation between high flight numbers and successful landings.

Payload and Landing Outcomes:

- Positive relationships identified between heavy payload masses and successful landings for specific orbits but an unclear relationship between payloads, successful landings, and GTO orbit.

Temporal Trends:

- Success rates display temporal variations, with peaks in 2014, 2017, and 2019.

- Dashboard visualization indicates increasing payload mass correlates with higher success rates.

Booster Performance:

- Booster version FT consistently outperforms others while v1.1 shows the highest incidence of failure.

Machine Learning Model Evaluation:

- "Decision Tree" indicates potential overfitting, while "KNN" and "SVM" consistently perform well in both metrics. "Logistic Regression" provides balanced performance across both metrics.

These insights emphasize the importance of flight numbers, payload characteristics, and launch site considerations for mission success. The machine learning models exhibit varying strengths, with the "Decision Tree" model standing out for overall accuracy.

Appendix

All information use can be gotten from the following link:

https://github.com/ajogwusalifu/Applied-Data-Science-Capstonestone

Thank you!