IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Indre Balyte-Zyke
2024-03-04

Outline

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

Executive Summary

- Summary of methodologies
  - Collecting the Data with an API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis using SQL
  - Exploratory Data Analysis for Data Visualization
  - Interactive Visual Analytics and Dashboards
  - Machine Learning for Predictive Analysis
- Summary of all results
  - Data analysis result
  - Data visualization
  - Predictive analysis

Introduction

- Project background and context

The commercial space age is here, companies are making space travel affordable for everyone. 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. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch

By using machine learning models we could determine if the first stage will land successfully and we could determine the cost of a launch.

- Problems you want to find answers
    - What variables have impact for the rocket landing successfully?
    - What relationships are between variables defining success rate
    - What conditions has to be in landing program to be successful

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data is collected by using SpaceX API
- Perform data wrangling
  - Response was decode using json and data was normalised into dataframe with featrures we will analyse
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

- Describe how data sets were collected:
  - The SpaceX REST API is used to get data
  - json_normalize function is used to convert JSON to a dataframe
  - Missing (null) values are replace with mean value
  - Falcon 9 Launch data is web scraping from Wikipedia
  - The Python BeautifulSoup package is used to web scrape HTML tables into a pandas dataframe

Data Collection – SpaceX API

- Use requests.get to launch data from SpaceX API
- decode the response content as a Json using .json() and turn it into a Pandas dataframe using .json_normalize()
- Clean the requested data

- GitHub URL of the completed SpaceX API calls notebook: https://github.com/IndreBZ/Capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

```
[6]: spacex_url="https://api.spacexdata.com/v4/launches/past"

[7]: response = requests.get(spacex_url)
     response
```

To make the requested JSON results more consistent, we will use the following stat

```
]: static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.clou

[87]: # Use json_normalize method to convert the json result into a dataframe
      data = pd.json_normalize(response.json())

[77]: # Calculate the mean value of PayloadMass column
      mean = data_falcon9['PayloadMass'].mean()

      # Replace the np.nan values with its mean value
      data_falcon9['PayloadMass'].fillna(value=mean, inplace=True)
      data_falcon9.isnull().sum()
```

Data Collection - Scraping

- Extract a Falcon 9 launch records HTML table from Wikipedia
- Parse the table and convert it into a Pandas data frame
- GitHub URL of the completed web scraping notebook: https://github.com/IndreBZ/Capstone/blob/main/jupyter-labs-webscraping.ipynb

```
[18]: # use requests.get() method with the provided static_url
# assign the response to a object
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falc
data  = requests.get(static_url).text
```

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(data, 'lxml')
```

```
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 row in first_launch_table.find_all('th'):
    name = extract_column_from_header(row)
    if (name != None and len(name) > 0):
        column_names.append(name)
```

```
#df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
#df.head()
df=pd.DataFrame(launch_dict)
df.head()
```

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

Data Wrangling

- Do Exploratory Data Analysis:
  - Identify and calculate the percentage of the missing values in each attribute using df.isnull().sum()/len(df)*100
  - Identify which columns are numerical and categorical by using df.dtypes
  - Calculate number of launches on each site, the number and occurrence of each orbit by using .value_counts()
- Determine Training Labels:
  - Creating outcome *Class* which represent the classification variable: 0 – failed landing, 1 – success landing

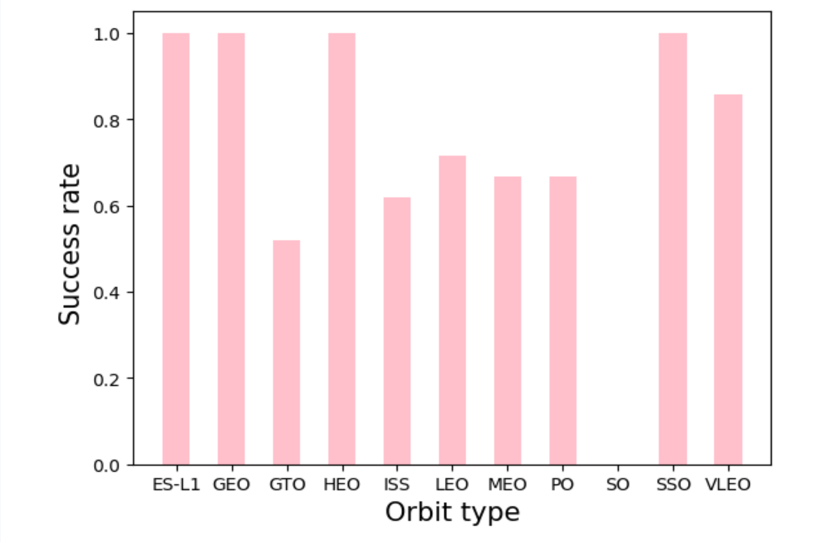
- GitHubURL of data wrangling related notebook: https://github.com/IndreBZ/Capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

Bar chart is created to visualise relationship between success rate and orbit type

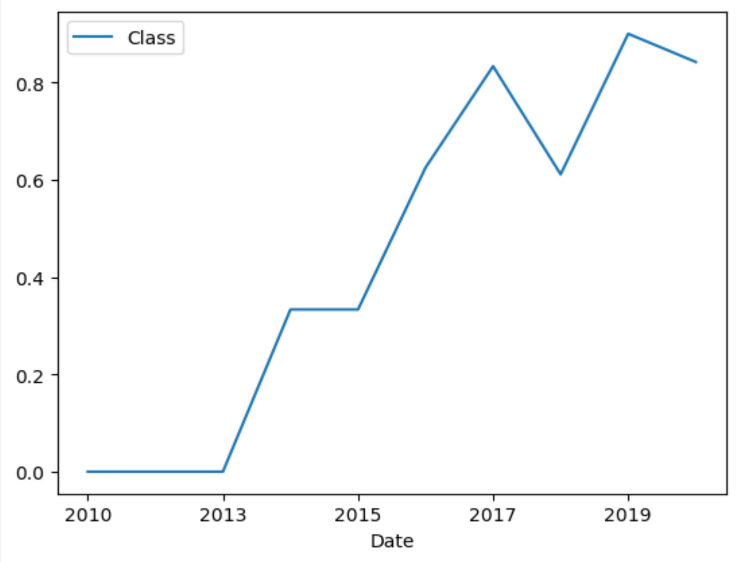
```
# HINT use groupby method on Orbit column and get the mean of Class column
orbit=df.groupby(['Orbit'],as_index=False)[['Class']].mean()
orbit

# Create bars
#plt.bar(y_pos, height)
plt.bar(orbit['Orbit'], orbit['Class'], color='pink', width=0.5)
plt.xlabel("Orbit type",fontsize=15)
plt.ylabel("Success rate ",fontsize=15)
plt.show()
```



Line chart with x axis = Year and y axis = average success rate shows the average launch success trend

```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
year_success=df.groupby(['Date'],as_index=False)[['Class']].mean()
year_success.plot(x='Date',y='Class', kind='line')
year_success
```



- GitHub URL of completed EDA with data visualization notebook:https://github.com/IndreBZ/Capstone/blob/main/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA with SQL

- Dataset was loaded into the corresponding table (SPACEXTBL) in a Db2 database
- Queries were written to solve these tasks:
  - 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 succesful landing outcome in ground pad was acheived
  - 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 mas
  - 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
- GitHub URL of completed EDA with SQL notebook: https://github.com/IndreBZ/Capstone/blob/main/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

Build an Interactive Map with Folium

- Following objects were created and added to a folium map:
  - Circle and marker at NASA Johnson Space Center's
  - Circle nea the the city of Houston
  - Markers of success/failed launches
  - Lines of distance between launch site to its proximities
  - Lines to the nearest coastlines, city, highway and railway
- Using markers with different colours help identify which launch site have better success rate
- Using lines – distance between launch site railway and highway helps to plan transportation
- Distance from city is used to keep save distance in case of failure outcome
- GitHub URL of interactive map with Folium map:https://github.com/IndreBZ/Capstone/blob/main/lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

- Summarize what plots/graphs and interactions you have added to a dashboard
- Explain why you added those plots and interactions
- Add the GitHub URL of your completed Plotly Dash lab, as an external reference and peer-review purpose

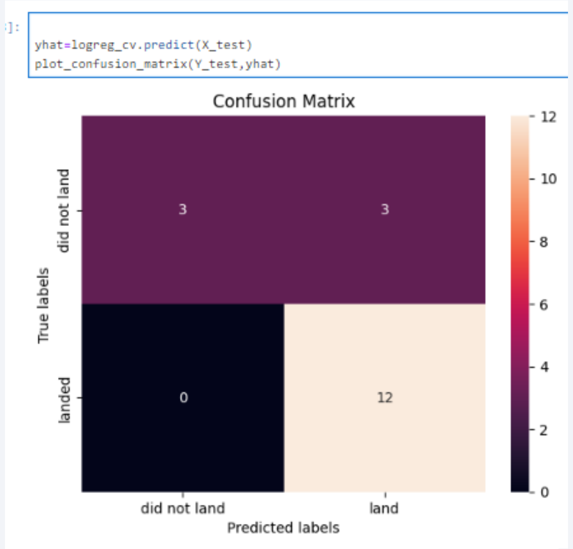
Predictive Analysis (Classification)

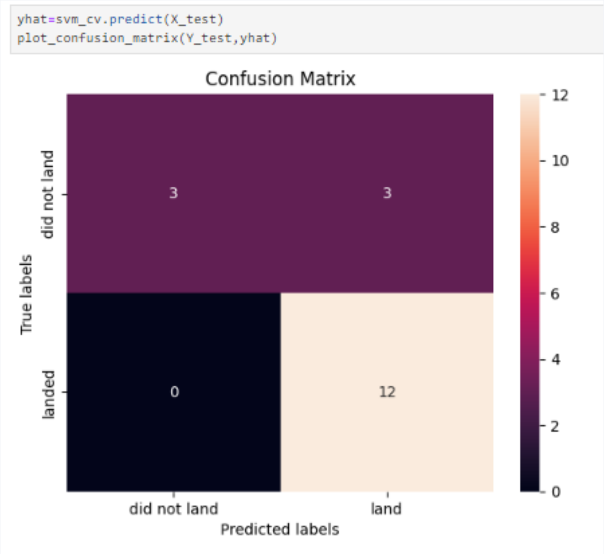
- Perform Exploratory data Analysis
    - create a column for the class - `Y = data['Class'].to_numpy()`
    - Standardize the data - `transform = preprocessing.StandardScaler() X = transform.fit_transform(X)`
    - Split into training data and test data - `X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)`
- Create different models for dataset:
    - Logistic Regression - `lr=LogisticRegression()logreg_cv = GridSearchCV(lr,parameters,cv=10)logreg_cv.fit(X_train, Y_train)`
    - Support Vector Machine - `svm = SVC() svm_cv = GridSearchCV(svm,parameters,cv=10)svm_cv.fit(X_train, Y_train)`
    - Decision Tree Classifier - `tree = DecisionTreeClassifier() tree_cv = GridSearchCV(tree,parameters,cv=10)tree_cv.fit(X_train, Y_train)`
    - K Neighbours Classifier - `KNN = KNeighborsClassifier() knn_cv = GridSearchCV(KNN,parameters,cv=10)knn_cv.fit(X_train, Y_train)`

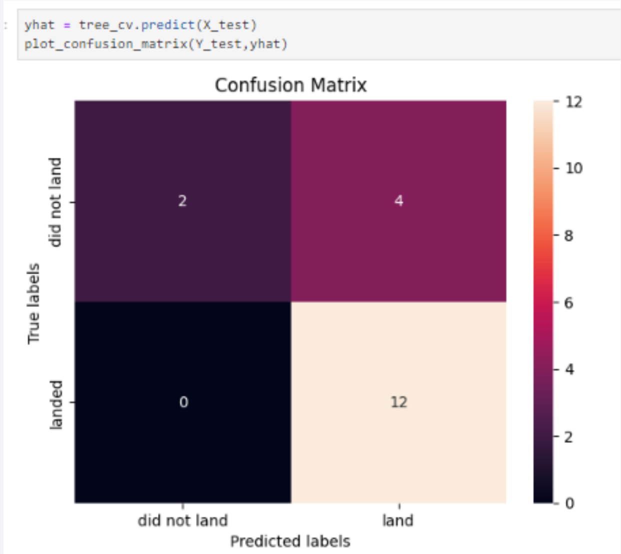
```
LogReg accuracy : 0.8333333333333334
SVM accuracy : 0.8333333333333334
Tree accuracy : 0.7777777777777778
KNN accuracy : 0.8333333333333334
```

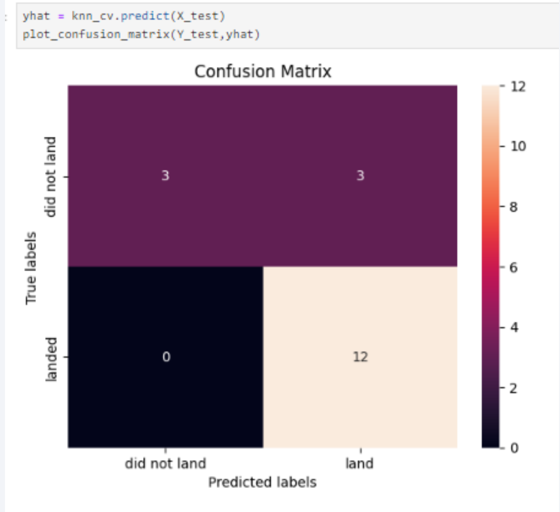
- GitHub URL of completed predictive analysis lab:https://github.com/IndreBZ/Capstone/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Results









Cofusiosion Matrix for all developed models

Predictive analysis results

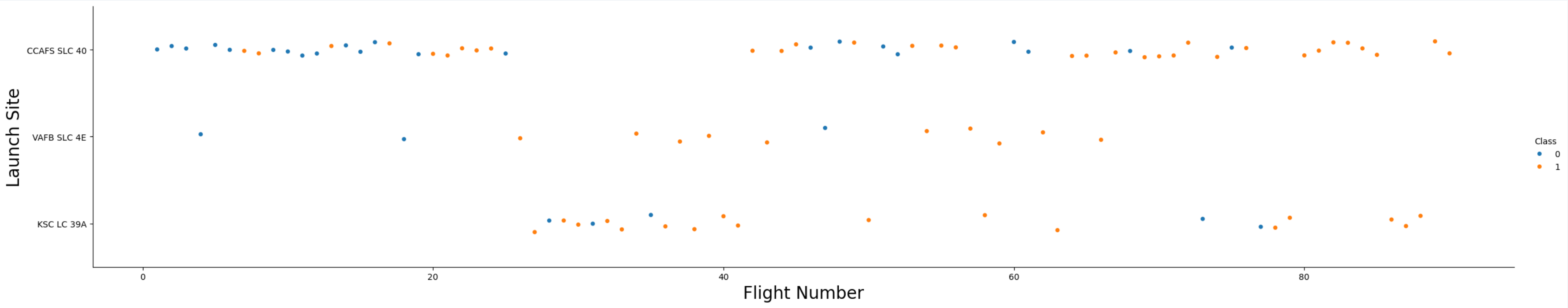
```
LogReg accuracy : 0.8333333333333334
SVM accuracy : 0.8333333333333334
Tree accuracy : 0.7777777777777778
KNN accuracy : 0.8333333333333334
```

Section 2

Insights drawn from EDA

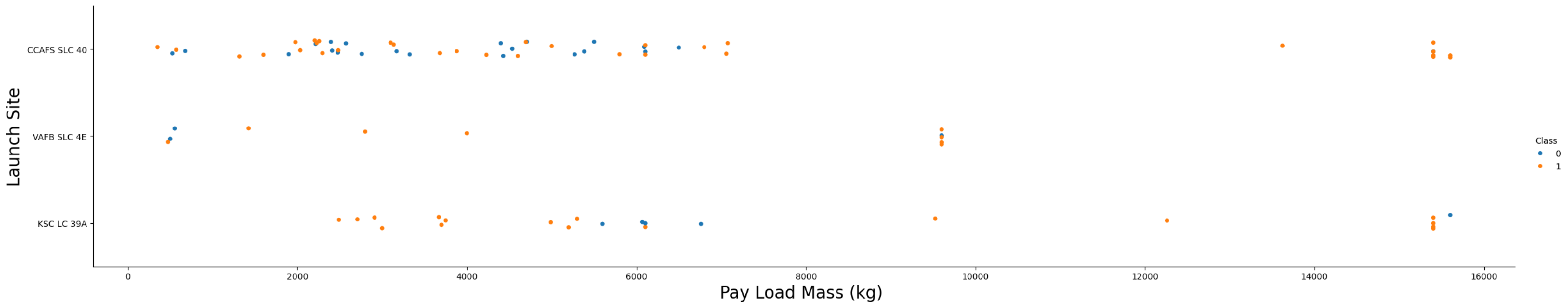
Flight Number vs. Launch Site

- From this plot we could see that the larger flight number we have the better success rate at a launch site



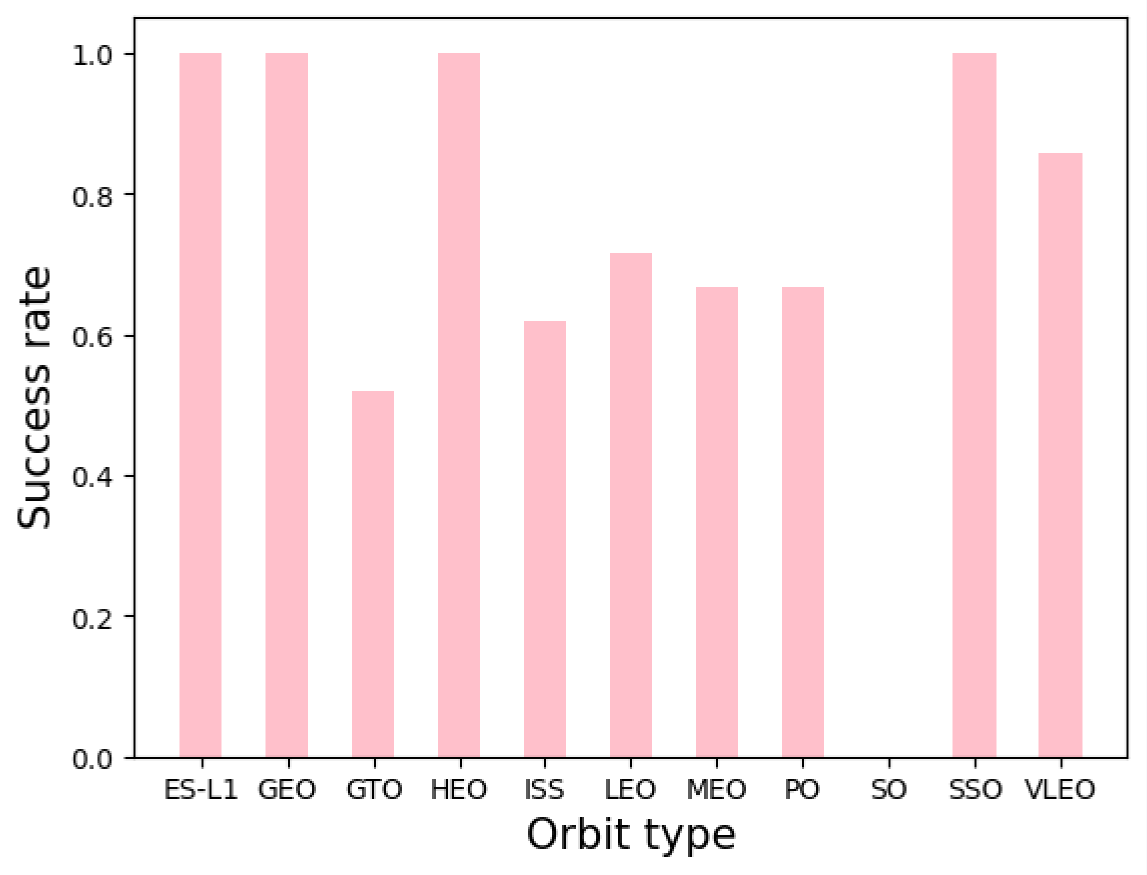
Payload vs. Launch Site

- From this plot we could see that the LargerPay Load mass the better success rate we have



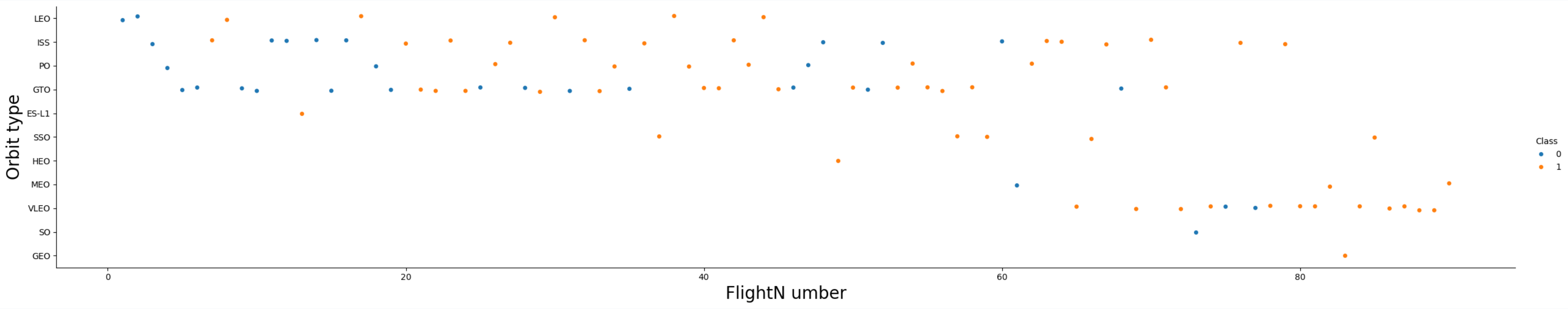
Success Rate vs. Orbit Type

- From this plot we could see that orbits ES-L1, GEO, HEO, SSO have the best success rate



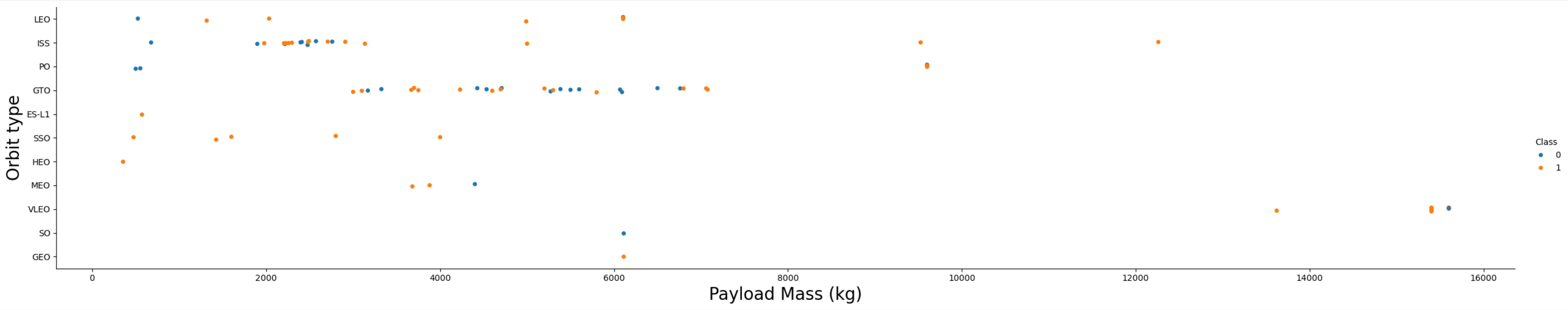
Flight Number vs. Orbit Type

- From this plot we could see that LEO, VLEO have better succes rate when flight number increase. Gto orbit doesn‘t have this dependency



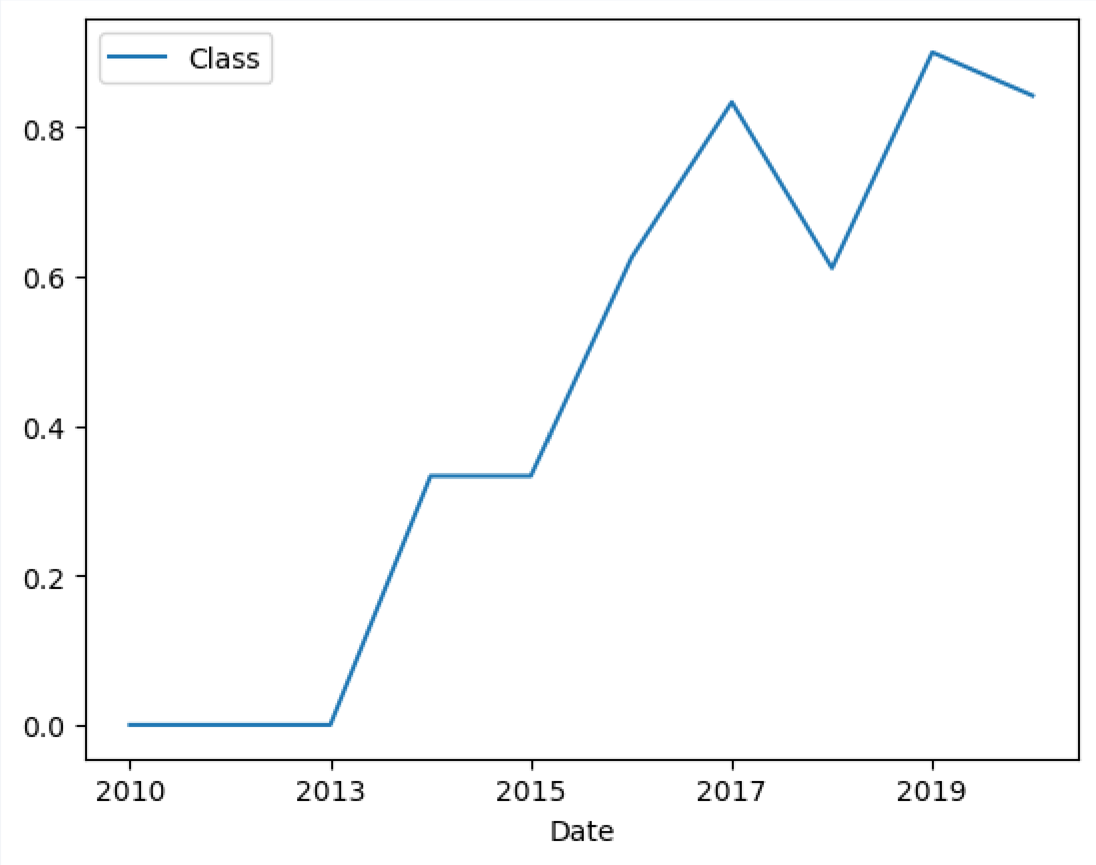
Payload vs. Orbit Type

- From this plot we could see that orbits LEO, ISS, PO orbits have better suucees rate when Payload Mass is bigger



Launch Success Yearly Trend

- From this plot we could see that success rate increases from 2023, except 2018



All Launch Site Names

- Find the names of the unique launch sites
- Using DISTINCT to have different names in table

```
%sql select distinct Launch_Site from SPACEXTBL
```

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

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

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`
- Using LIMIT to get exact number of records and LIKE to get specific name

Display 5 records where launch sites begin with the string 'CCA'

```
%sql select * from SPACEXTBL where Launch_Site like 'CCA%' limit 5
```

```
 * 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	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

Total Payload Mass

- Calculate the total payload carried by boosters from NASA
- Using SUM to calculate total amount

Task 3

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

```
%sql select sum(PAYLOAD_MASS__KG_) from SPACEXTBL where Customer = 'NASA (CRS)'
```

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

sum(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1
- Using AVG to calculate average

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

```
%sql select avg(PAYLOAD_MASS__KG_) from SPACEXTBL where Booster_Version = 'F9 v1.1'
```

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

avg(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad
- Using MIN to get the first date

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

Hint:Use min function

```
%sql select min(date) from SPACEXTBL where    Landing_Outcome='Success (ground pad)'
```

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

min(date)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

- Using BETWEEN to get specific mass

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

```
%sql select Booster_Version from SPACEXTBL where  Landing_Outcome = 'Success (drone ship)' and PAYLOAD_MASS__KG_ between 4000 and 6000
```

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

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

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes
- Define success rate with CASE and using GROUP BY

List the total number of successful and failure mission outcomes

```
%sql select case when Landing_Outcome like '%Success%'then 'Success' when Landing_Outcome like '%Failure%'then 'Failure' End as Outcome, count(*) as Total_number from SPACEXTBL group by case when Landing_Outcome like '%Success%'then 'Success' when Landing_Outcome like '%Failure%'then 'Failure' End
```

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

Outcome	Total_number
None	30
Failure	10
Success	61

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass
- Using subquery to calculate max payload mass

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

```
%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

2015 Launch Records

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

- Using SUBSTRING to get month from date

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.

```
%sql select substr(Date, 6,2),Booster_Version,Landing_Outcome,Launch_Site from SPACEXTBL where substr(Date,0,5)='2015' and Landing_Outcome = 'Failure (drone ship)'
```

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

substr(Date, 6,2)	Booster_Version	Landing_Outcome	Launch_Site
01	F9 v1.1 B1012	Failure (drone ship)	CCAFS LC-40
04	F9 v1.1 B1015	Failure (drone ship)	CCAFS LC-40

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

- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order
- Using GROUP BY to get rates and DESC in ordering to get rank

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

```
%sql select Landing_Outcome, count(*) from SPACEXTBL where Date between '2010-06-04' and '2017-03-20'group by Landing_Outcome order by 2 desc
```

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

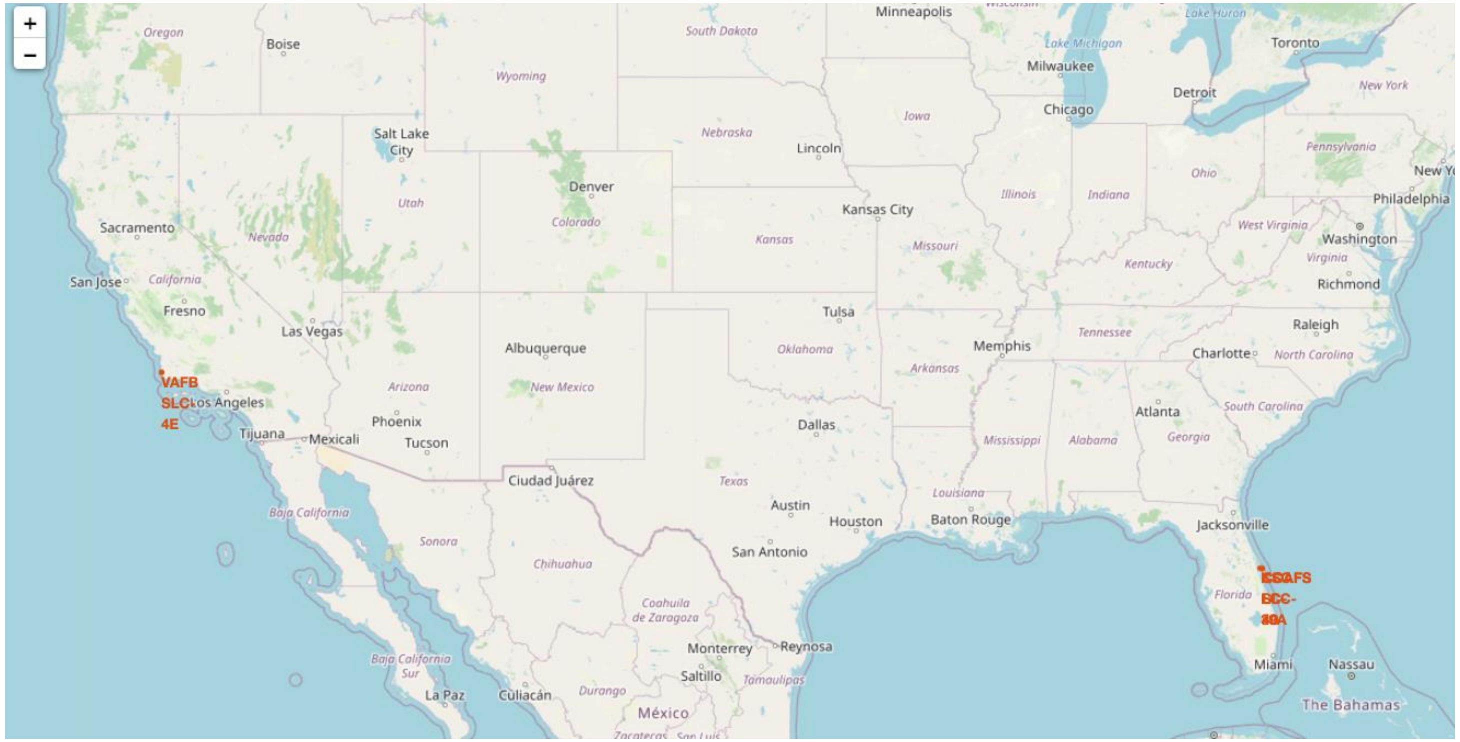
Landing_Outcome	count(*)
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

All launch sites global map markers

- Space X launch sites are in the USA coasts: Californiai and Florida
- Launch sites are near coasts and close to the Equator line



Launch sites with color labels

- Green markers show successful launched, red - failures

Florida

California

Launch Site distance to landmarks

- Distance to the coast

Calculate distance the nearest cous, highway, railway and the city

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)

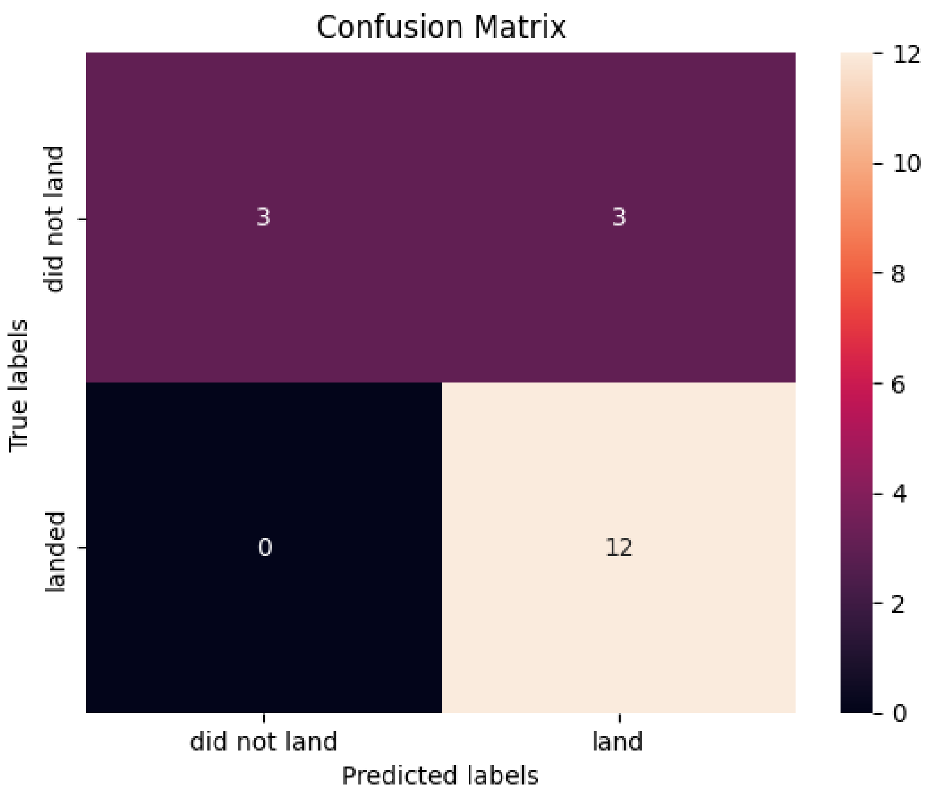
Classification Accuracy

- Visualize the built model accuracy for all built classification models, in a bar chart

- The three models Logistic regression, support vector maschine and k nearest neighbors have the same score: 0.83 and decision tree has the lowest score 0.78

```
print("LogReg accuracy :",logreg_cv.score(X_test, Y_test))
print("SVM accuracy :",svm_cv.score(X_test, Y_test))
print("Tree accuracy :",tree_cv.score(X_test, Y_test))
print("KNN accuracy :",knn_cv.score(X_test, Y_test))
```

```
LogReg accuracy : 0.8333333333333334
SVM accuracy : 0.8333333333333334
Tree accuracy : 0.7777777777777778
KNN accuracy : 0.8333333333333334
```



Confusion Matrix

- Since we have 3 confusion matrix the same, this is one of them
- We see that landing fact is predicted very well but for not landing this is 50% not very accurate

Conclusions

- The larger flight number we have the better success rate at a launch site
- Orbits ES-L1, GEO, HEO, SSO have the best success rate
- Ther models (log reg, svm, knn) have the same accuracy score

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!