IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Shehla Kulsum Baig
24-4-2024

Outline

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

Executive Summary

- Summary of methodologies:
  - Data Collection
  - Data Wrangling
  - EDA with SQL
  - EDA with data visualization
  - Building interactive map with Folium
  - Building Dashboard with plotly Dash
  - Predictive analysis using Classification
- Summary of all results:
  - Exploratory Data Analysis (EDA) results
  - Interactive analytics results
  - Predictive analytics results

Introduction

- Project background and context:

  Here we use data analytics to predict whether first stage of SpaceX's Falcon9 will land successfully. This will help us to determine total cost in case of successful launce. It is also important to determine cost of failure in case of unsuccessful launch.

- Problems you want to find answers:
  - What factors are responsible for success or failure of rocket launch.
  - Find relationship between various parameters that will impact rocket launch
  - What is the accuracy for successful landing.
  - Which conditions will achieve higher success rate

Section 1

Methodology

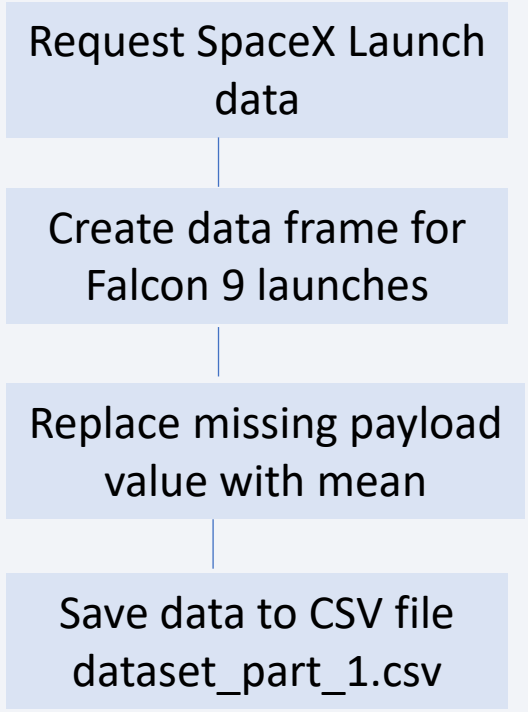
Methodology

Executive Summary

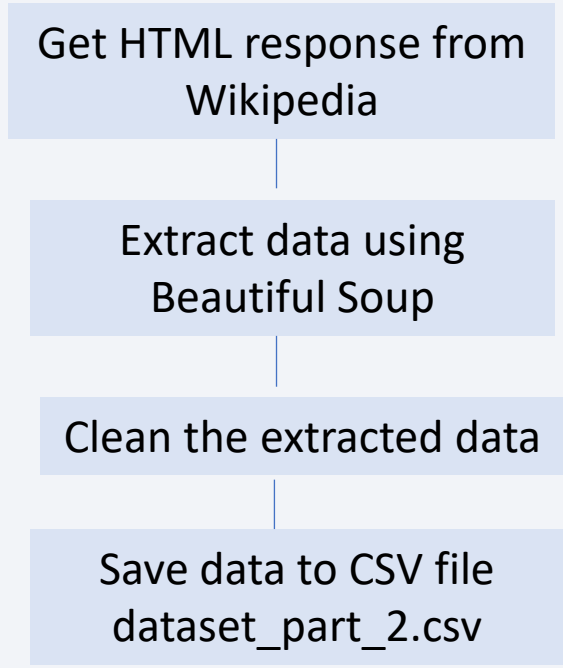
- Data collection methodology:
  - SpaceX Rest API
  - Web scraping using Wikipidia
- Perform data wrangling
  - Transform data and apply One Hot Encoding for ML feature selection
- Perform exploratory data analysis (EDA) using visualization and SQL
  - EDA for feature engineering and Visualization techniques like Scatter plot, Bar graph to discover patterns in data.
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Machine Learning algorithms for classification (LR, K-NN, SVM, Decission Tree) were used

Data Collection

1) Data was collected from open source SpaceX REST API. This data include info on rocket, Launchpad, payload, cores, etc.



2) Data was collected using web scraping Wikipedia using Beautiful Soup for Falcon 9 launch data



Data Collection – SpaceX API

GitHub Link : https://github.com/shehla1739/Applied-Data-Science-Capstone/blob/main/Week1%20-%20Lab1%20DataCollectWithAPI%20(1).ipynb

1 Getting Response from APT

```
In [9]: static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork
                        /datasets/API_call_spacex_api.json'
```

2. Converting Response to .json file

```
In [11]: # Use json_normalize meethod to convert the json result into a dataframe
         data = pd.json_normalize(response.json())
```

3. Apply custom function to clean data

```
In [13]: # Lets take a subset of our dataframe keeping only the features we want and the flight r
         data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

         # We will remove rows with multiple cores because those are falcon rockets with 2 extra
         data = data[data['cores'].map(len)==1]
         data = data[data['payloads'].map(len)==1]

         # Since payloads and cores are lists of size 1 we will also extract the single value in
         data['cores'] = data['cores'].map(lambda x : x[0])
         data['payloads'] = data['payloads'].map(lambda x : x[0])

         # We also want to convert the date_utc to a datetime datatype and then extracting the da
         data['date'] = pd.to_datetime(data['date_utc']).dt.date

         # Using the date we will restrict the dates of the launches
         data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

4. Assign list to dictionary and then to dataframe

```
In [21]: 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. Filter dataframe and export to .csv file

```
In [26]: # Hint data['BoosterVersion']!='Falcon 1'
         data_falcon9 = data.loc[data['BoosterVersion']!="Falcon 1"]
```

```
In [34]: data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data Collection - Scraping

GitHub: https://github.com/shehla1739/Applied-Data-Science-Capstone/blob/main/Week1%20-%20Data%20Collection%20With%20WebScrapingFalcon9.ipynb

1 Creating BeautifulSoup object

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

2. Getting column names

```
column_names = []

# Apply find_all() function with `th` element on first
# Iterate each th element and apply the provided extra
# Append the Non-empty column name (`if name is not No

temp = soup.find_all('th')
for x in range(len(temp)):
    try:
     name = extract_column_from_header(temp[x])
     if (name is not None and len(name) > 0):
        column_names.append(name)
    except:
     pass
```

3. Creating the launch_dict

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each
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']=[]
```

4. Converting to dataframe

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

5. Filter dataframe and export to .csv file

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

Data Wrangling

GitHub URL: https://github.com/shehla1739/Applied-Data-Science-Capstone/blob/main/Week%201%20-%20Lab2%20Data%20wrangling.ipynb

1 Load the dataset

```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
               IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_1.csv")
df.head(10)
```

2. Create landing outcome

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for key,value in df["Outcome"].items():
     if value in bad_outcomes:
        landing_class.append(0)
     else:
        landing_class.append(1)
```

3. Find the bad outcomes

```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

4. Present outcomes as 0's and 1's

```
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

5. Determine the success outcome

```
df["Class"].mean()
```

```
0.6666666666666666
```

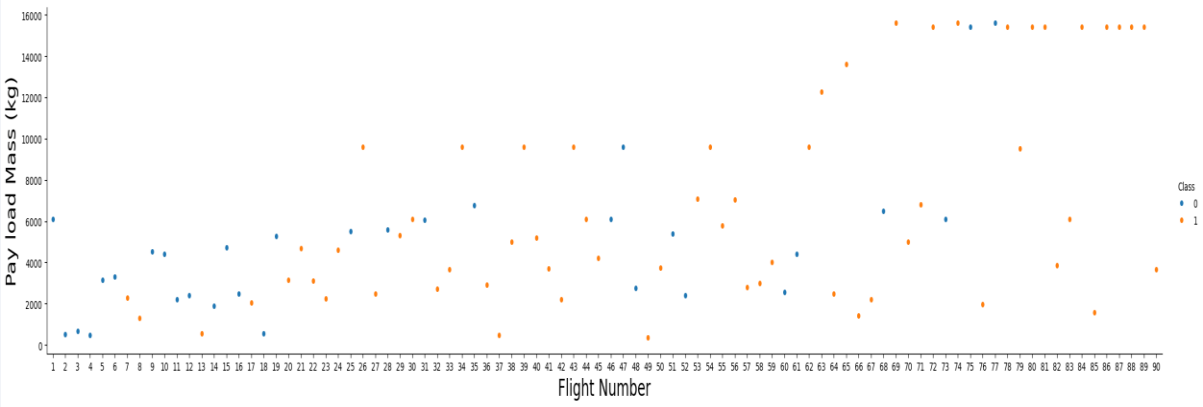
6. export to .csv file

```
df.to_csv("dataset_part_2.csv", index=False)
```

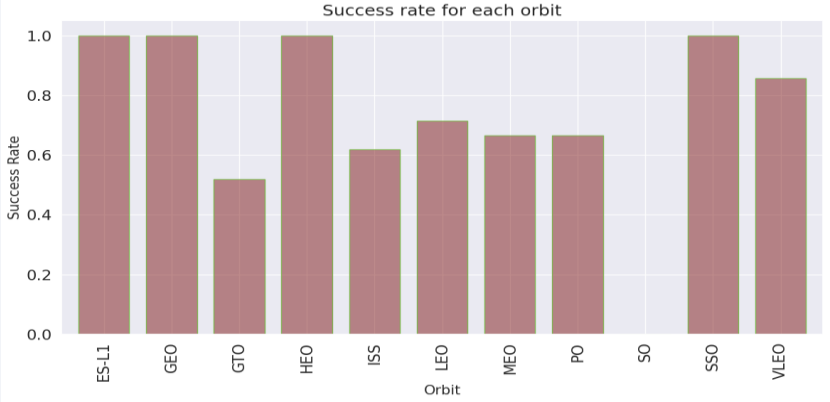
EDA with Data Visualization

GitHub URL: https://github.com/shehla1739/Applied-Data-Science-Capstone/blob/main/Week2%20-%20Explore%20and%20Prepare%20data%20-%20edadataviz.ipynb

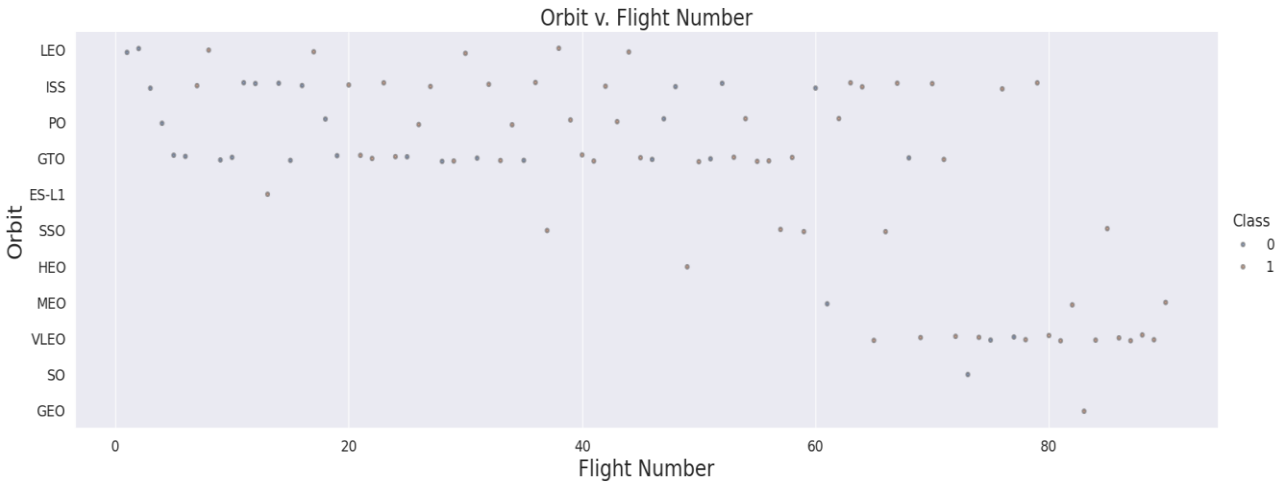
Category plot between Flight number and Payload mass (Kg)



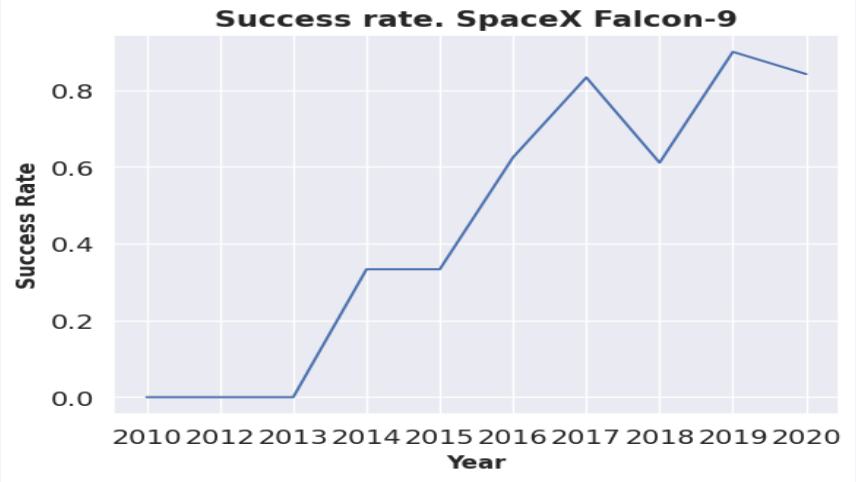
Bar chart between Orbit and success rate of each orbit



Scatter plot between Orbit and Flight number



Line chart between Year and Success rate



EDA with SQL

GitHub URL: https://github.com/shehla1739/Applied-Data-Science-Capstone/blob/main/Week2%20-%20labs%20SQL%20Notebook%20eda-sql-coursera_sqllite.ipynb

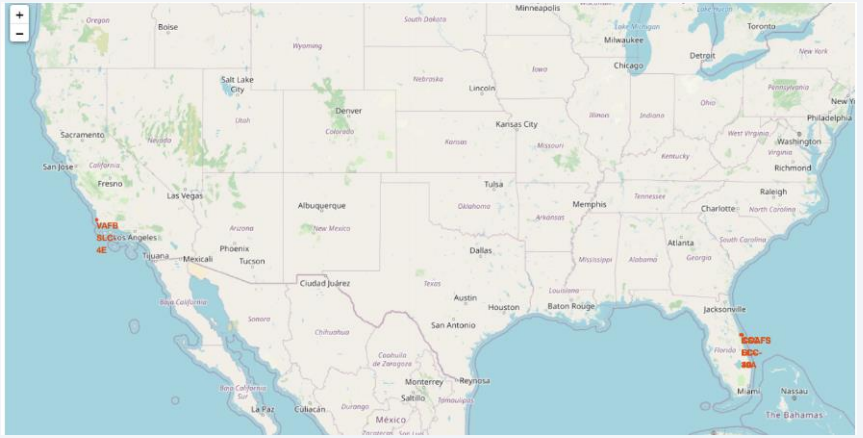
Following are SQL queries performed to gather information of dataset:

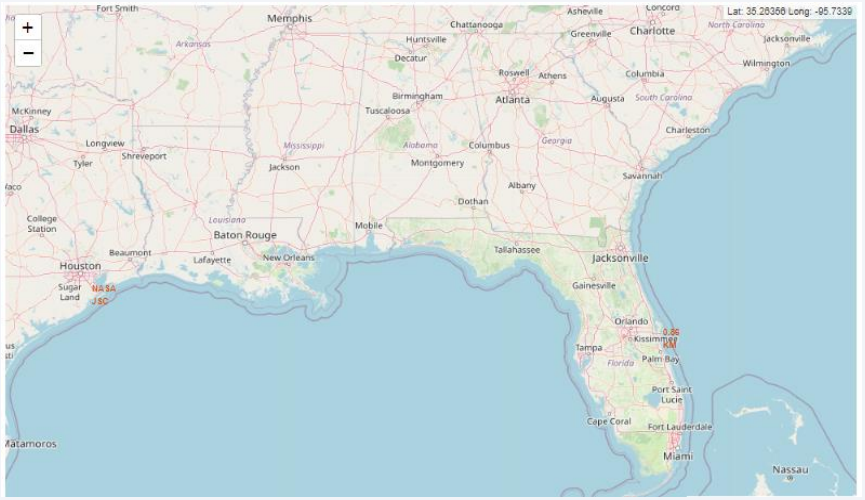
- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'CCA'
- Display the total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the 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. Use a subquery
- 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.
- 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.

Build an Interactive Map with Folium

GitHub URL: https://github.com/shehla1739/Applied-Data-Science-Capstone/blob/main/Week3%20-%20lab_launch_site_location.ipynb

- Folium Interactive Map was used to visualize and analyze SpaceX Launch Sites. The distance between a launch site was claculated by using longitude and latitude coordinate information.
- To create markers on map, folium.Marker() was used. To create circles folium.Circles() was used. To create icons on map folium.icon was used. Folium.polyLine() was used to create polynomial lines between the points.
- The successful and failed launches for each site on map were identified by assigning launch_outcome(failures, successes) to class 0 / 1 with red and green marhers on the map in a MarkerCluster()





Build a Dashboard with Plotly Dash

- The interactive dashboard were build using Plotly and Dash.
- The Dropdown menu is provided for selecting launch sites. The default select value is for ALL sites
- The pie chart is used to show the total successful launches count for all sites. If a specific launch site was selected, then it shows the Success vs. Failed counts for the site
- Scatter chart were used for displaying launch site, payload mass, success/failure.
- A scatter chart is used to show the correlation between payload and launch success. A callback function is used for `site-dropdown` as input, `success-pie-chart` as output. Another callback function is added for `site-dropdown` and `payload-slider` as inputs, `success-payload-scatter-chart` as output.

GitHub URL:
https://github.com/shehla1739/Applied-Data-Science-Capstone/blob/main/Week3%20-%20spacex_dash_app.py

Predictive Analysis (Classification)

GitHub URL: https://github.com/shehla1739/Applied-Data-Science-Capstone/blob/main/Week4%20-%20SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Model Building:

- Load dataset into Numpy and Pandas
- Transform data
- Split data into training and testing datasets
- Select machine learning algo to use and set parameter
- Train the ML model using training dataset

Model Evaluation:

- Use testing dataset to check accuracy of ML Model
- For each ML model use tuned hyper-parameter
- Plot Confusion Matrix for ML model being used

Model Improving

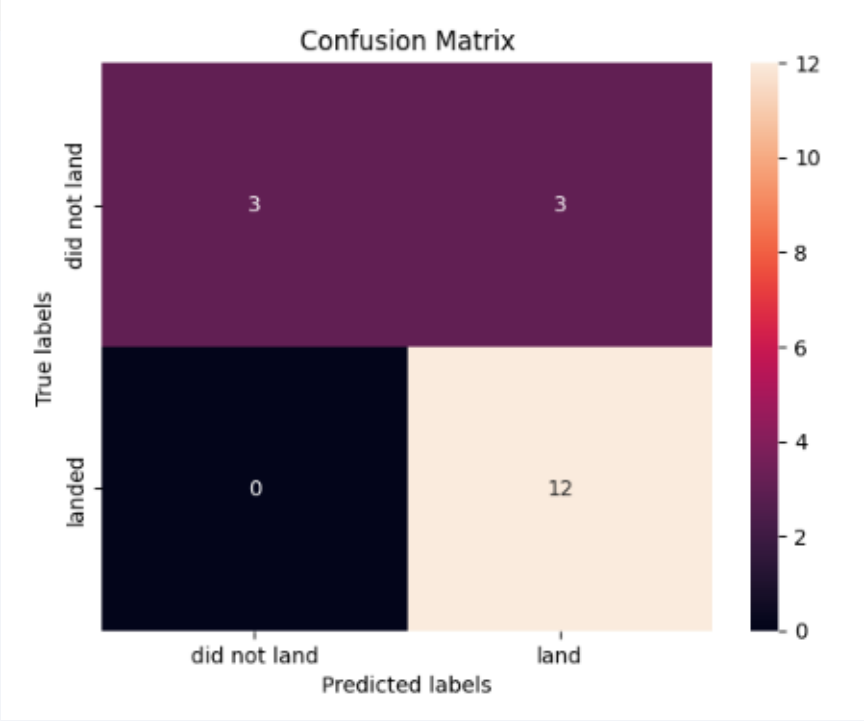
- Feature Engineering is used to improve model performance
- Algorithm tuning is also used for model improvement

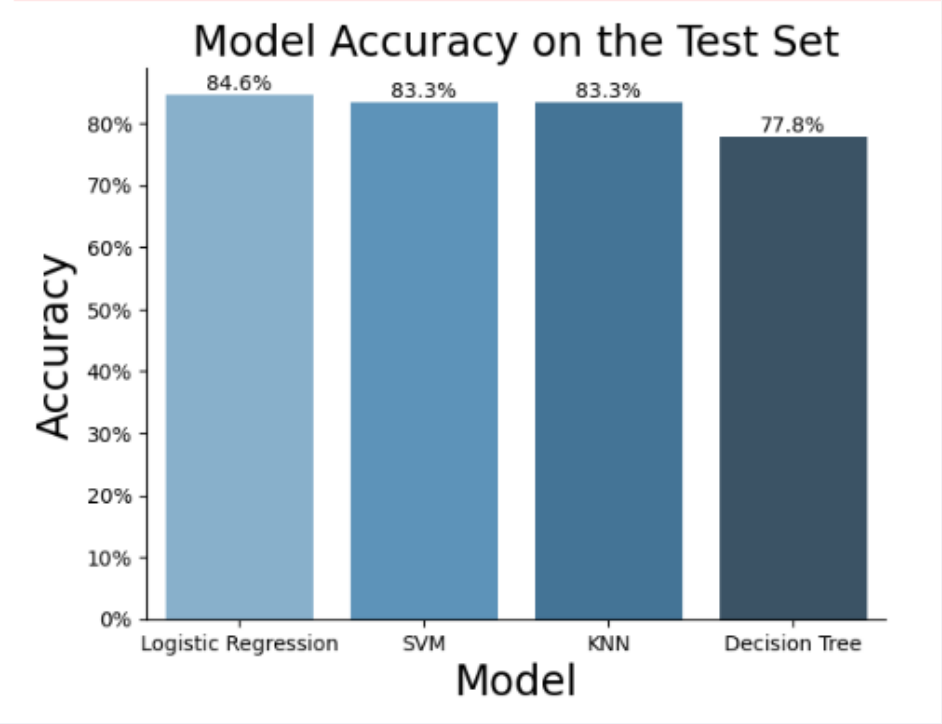
Results

GitHub URL: https://github.com/shehla1739/Applied-Data-Science-Capstone/blob/main/Week4%20-%20SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

The best Classification Model is the one with highest accuracy score. The models hyper-parameters were optimized using Scikit-learn GridSearchCV .

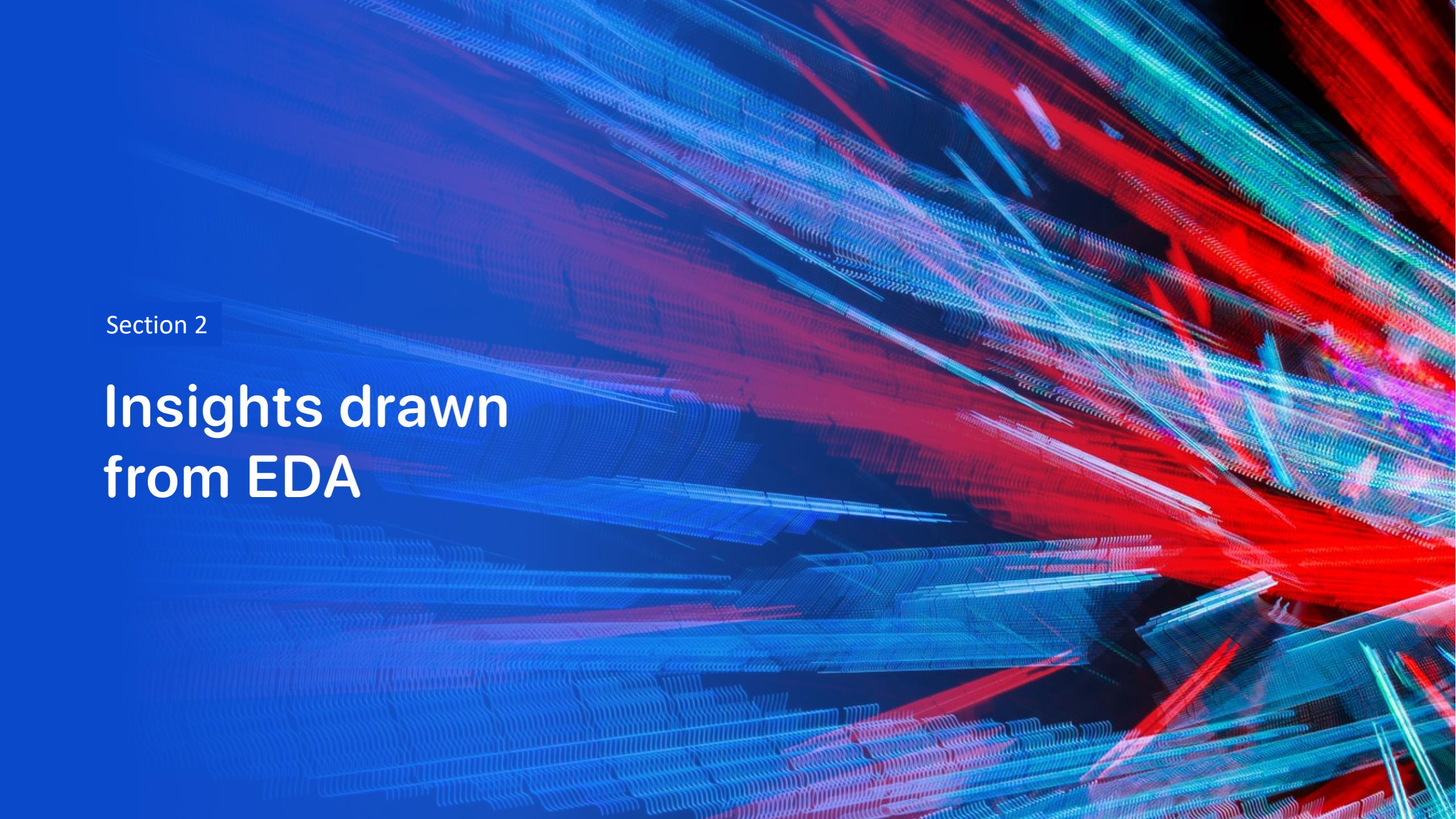
The Logistic Regression, SVM, KNN, model achieve accuracy of 83.3%, where as the accuracy of Decision Tree is 77.8%
For the given dataset, SVM, KNN, Logistic Regression model achieve best in terms of predictive accuracy





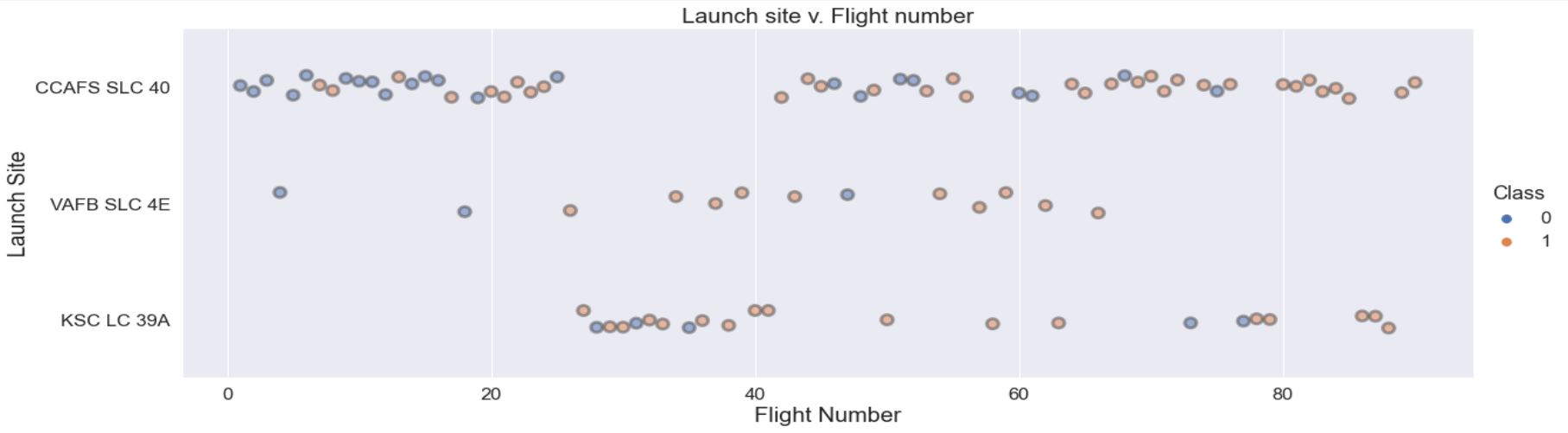
Section 2

Insights drawn from EDA



Flight Number vs. Launch Site

- CCAFS SLC-40 is the most used launch site.
- Launches from the site of CCAFS SLC-40 are significantly higher than launches form other sites
- CCAFS SLC-40 has most of failures specially in the early stage

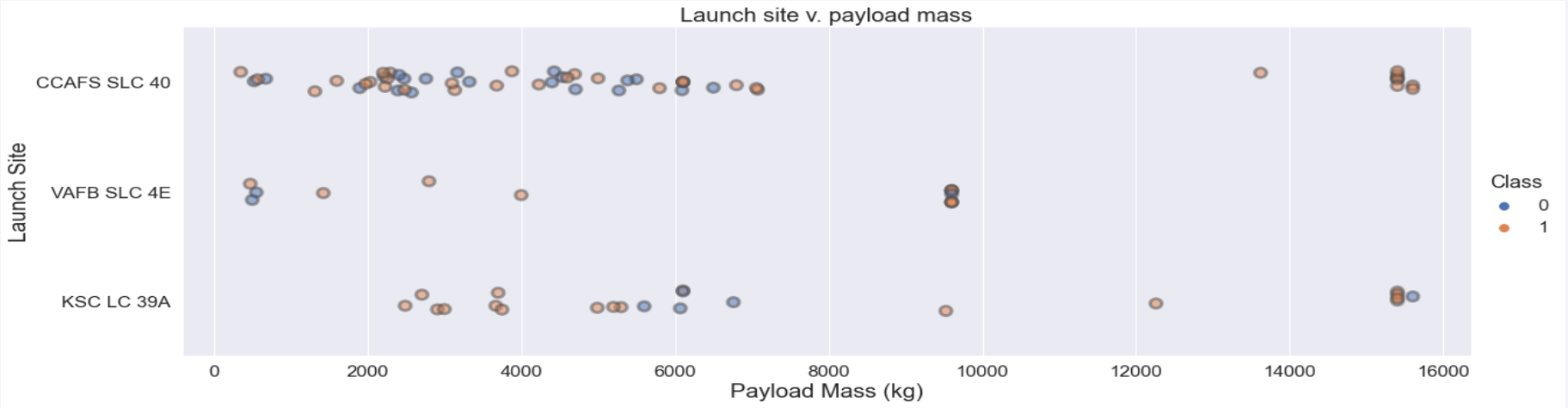


Payload vs. Launch Site

The percentage of failures is lower for heavy payload

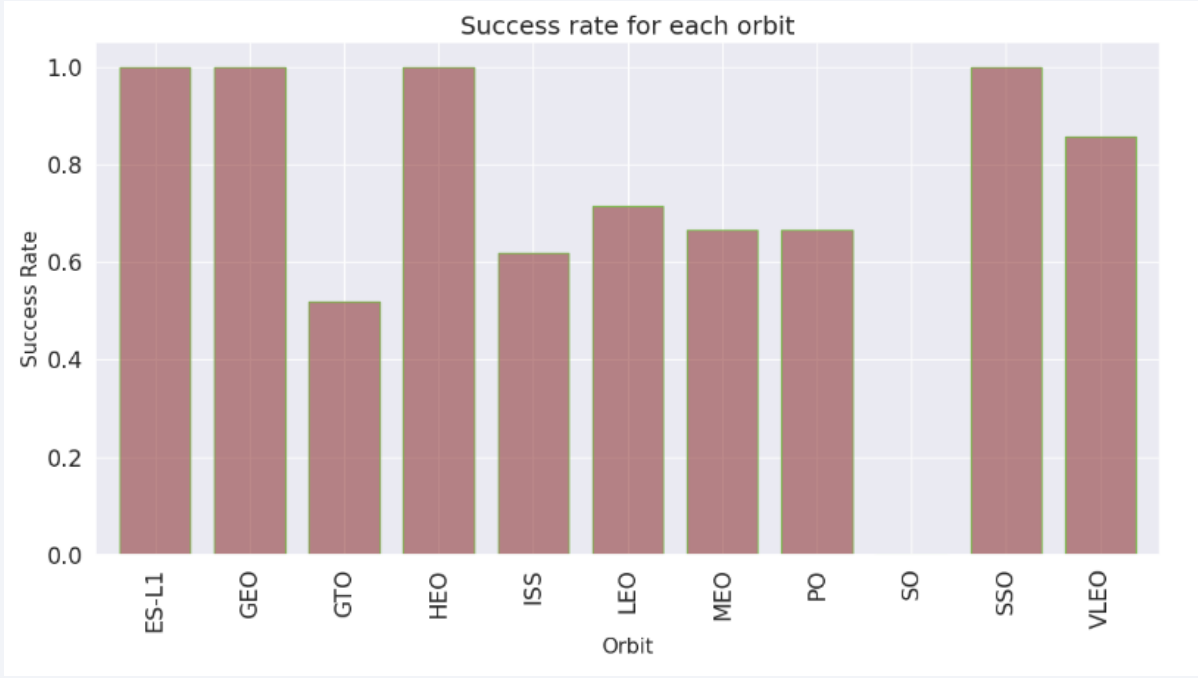
The majority of payload with lower mass has been launched from CCAFS SLC 40

heavy payloads > 10000 kg are sent to low/medium orbits LEO/MEO only



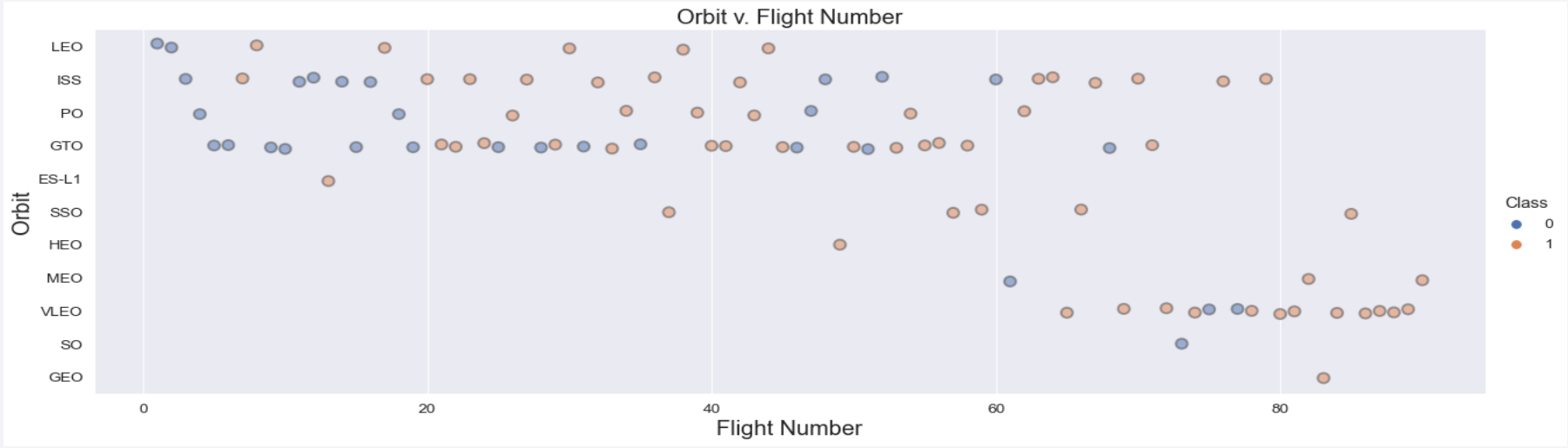
Success Rate vs. Orbit Type

- Success rate may strongly depend payload mass and orbit.
- Orbit GEO, HEO, SSO, ES-L1 has the highest success rate
- GTO has the lowest success rate where as SSO (polar low orbit) the highest one.
- ES-L1, GEO, HEO, SSO has success rate of 100% where as SO has success rate of 0%



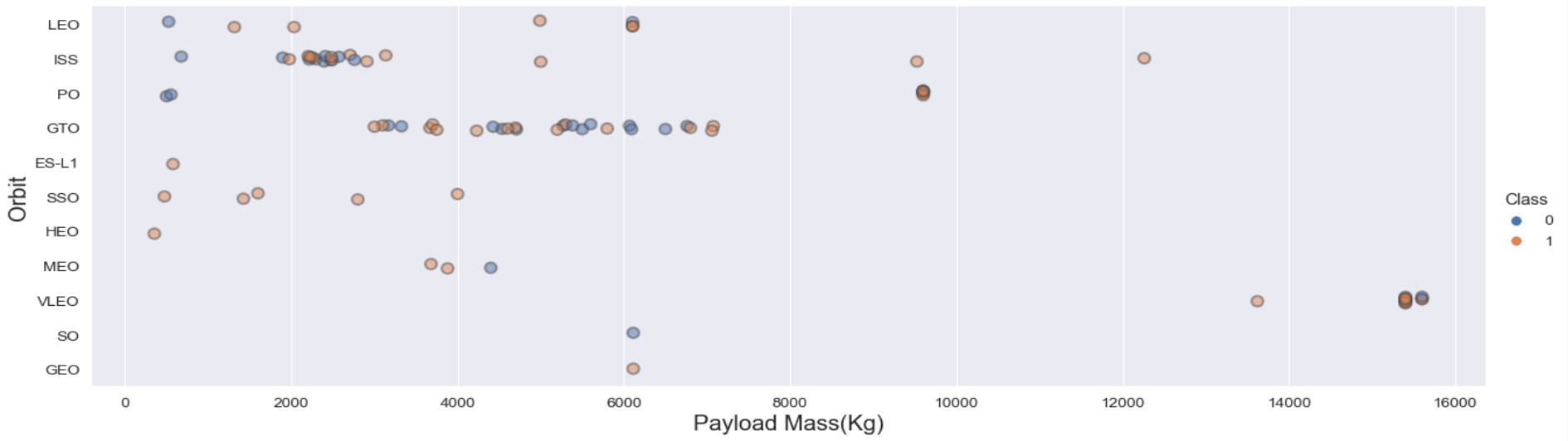
Flight Number vs. Orbit Type

- PO, SSO, ISS, VLEO are low orbits
- GTO is a transfer orbit to GEO.
- The number of flights for: GEO, SO, HEO, ESL-1, MEO is not significant for concluding about success rate



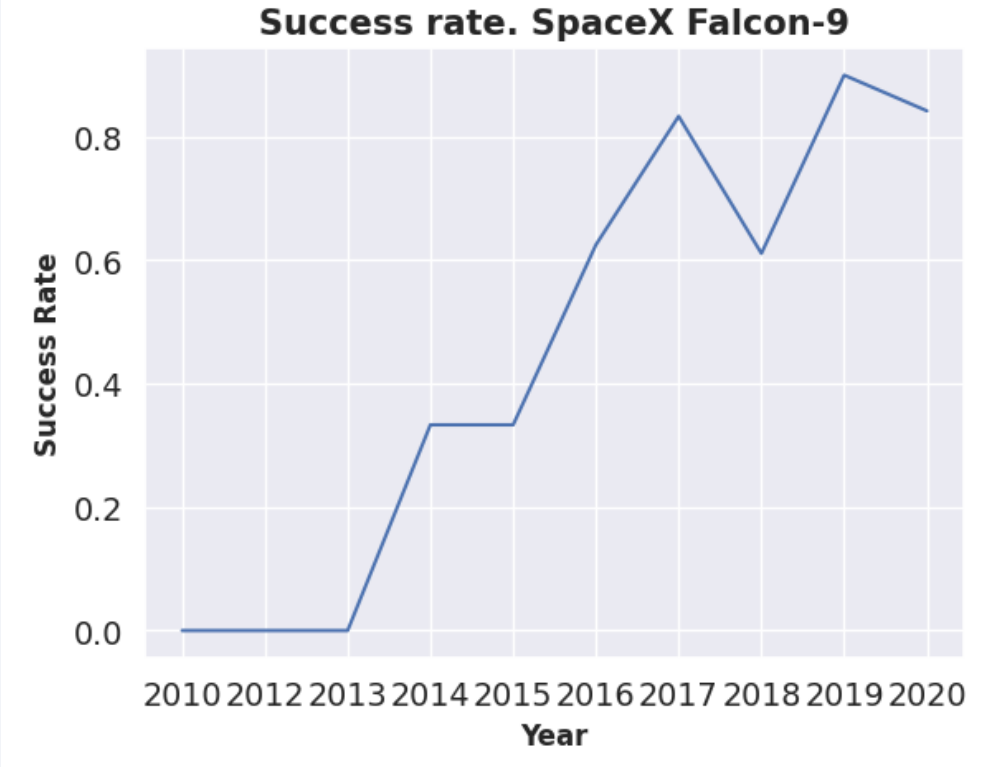
Payload vs. Orbit Type

- Maximum success rate with: low orbit except (ISS) and low payload mass
- Between 2000 and 7500 kg, success rate seems to be evenly distributed for GTO
- Payload mass between 2000 and 3000 is affecting ISS
- Payload mass between 3000 and 7000 is affecting GTO



Launch Success Yearly Trend

From the line chart we observe that success rate significantly improved over time. There is increase in success rate from 2013 till 2017 then there is drop for year 2018 then again increase in success rate.



All Launch Site Names

```
%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

Explanation:

DISTINCT keyword in SQL eliminates all duplicate records from the result returned by the SQL query. Therefore above SQL statement will return unique launch sites from tblSpaceX

Launch Site Names Begin with '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

Explanation:

LIKE is a logical operator in SQL that allows you to match on similar values. Here we use CCA% to match all words beginning with CCA

The LIMIT clause is used to specify the number of records to return. Here we are retrieving 5 records.

Total Payload Mass

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

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

SUM(PAYLOAD_MASS__KG_)
45596

Explanation:

The SUM() function returns the total sum of a numeric column. Here sum summates Total_Mass_Kg_ column

The WHERE clause specifies criteria that field values must meet for the records that contain the values to be included in the query results. Here where filters only those records where Customer is equal to NASA(CRS)

Average Payload Mass by 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

Explanation:

The AVG() function returns the average value of a numeric column. Here we use avg() function to compute the average value on Payload_Mass_Kg_ column

The WHERE clause is used to filter records. Here Where clause is used to filter records where Booster_Version is equal to ‘F9v1.1’

First Successful Ground Landing Date

```
%sql SELECT min(DATE) FROM SPACEXTBL WHERE LANDING_OUTCOME='Success (ground pad)'
```

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

min(DATE)
2015-12-22

Explanation:

The MIN() function returns the smallest value of the selected column. Here min() function is used on date column.

The WHERE clause is used to filter records. Here Where clause is used to filter records where Landing_Outcome is equal to Success(drone ship)

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ between 4000 and 6000 AND LANDING_OUTCOME='Success (drone ship
```

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

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

Explanation:

In above query select clause is used for selecting Booster Version

The where clause is used to filter records where Landing_Outcome is equal to Success(drone ship)

The AND operator displays a record if all the conditions separated by AND are TRUE. Here we use ‘and’ to filter condition Payload_Mass_Kg_> 4000 and Payload_Mass_Kg_< 6000

Total Number of Successful and Failure Mission Outcomes

```
%sql SELECT COUNT(*) FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE '%Success%' OR MISSION_OUTCOME LIKE '%Failure%'
```

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

COUNT(*)
101

Explanation:

In the above query The COUNT() function returns the number of records returned by a select query. The LIKE is a logical operator in SQL that allows you to match on similar values rather than exact ones. The above query is used to count total number of successful and failure missions

Boosters Carried Maximum Payload

```
%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

Explanation:

A subquery is a query within another query. The outer query is called as main query and inner query is called as subquery.

A subquery in a WHERE clause can be used to qualify a column against a set of rows.

Here we made used of subquery in main query to list Boosters carrying max payload

2015 Launch Records

```
%sql SELECT substr(Date,6,2) as month, DATE,BOOSTER_VERSION, LAUNCH_SITE, LANDING_OUTCOME \
FROM SPACEXTBL \
where LANDING_OUTCOME = 'Failure (drone ship)' and substr(Date,0,5)='2015';
```

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

month	Date	Booster_Version	Launch_Site	Landing_Outcome
01	2015-01-10	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
04	2015-04-14	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

Explanation:

The SUBSTRING() function extracts a substring from a string (starting at any position).

Here, we selected the failed landing_outcomes in drone ship, their booster versions, and launch site names for in

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

```
%sql SELECT LANDING_OUTCOME, count(*) as count_outcomes FROM SPACEXTBL \
WHERE "DATE" BETWEEN '2010-06-04' and '2017-03-20' \
    GROUP BY "DATE" \
    ORDER BY count_outcomes DESC
```

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

Landing_Outcome	count_outcomes
No attempt	1
Success (ground pad)	1
Success (drone ship)	1
Success (drone ship)	1
Success (ground pad)	1
Failure (drone ship)	1
Success (drone ship)	1
Success (drone ship)	1
Success (drone ship)	1
Failure (drone ship)	1

Explanation:

The COUNT() function returns the number of records returned by a select query.

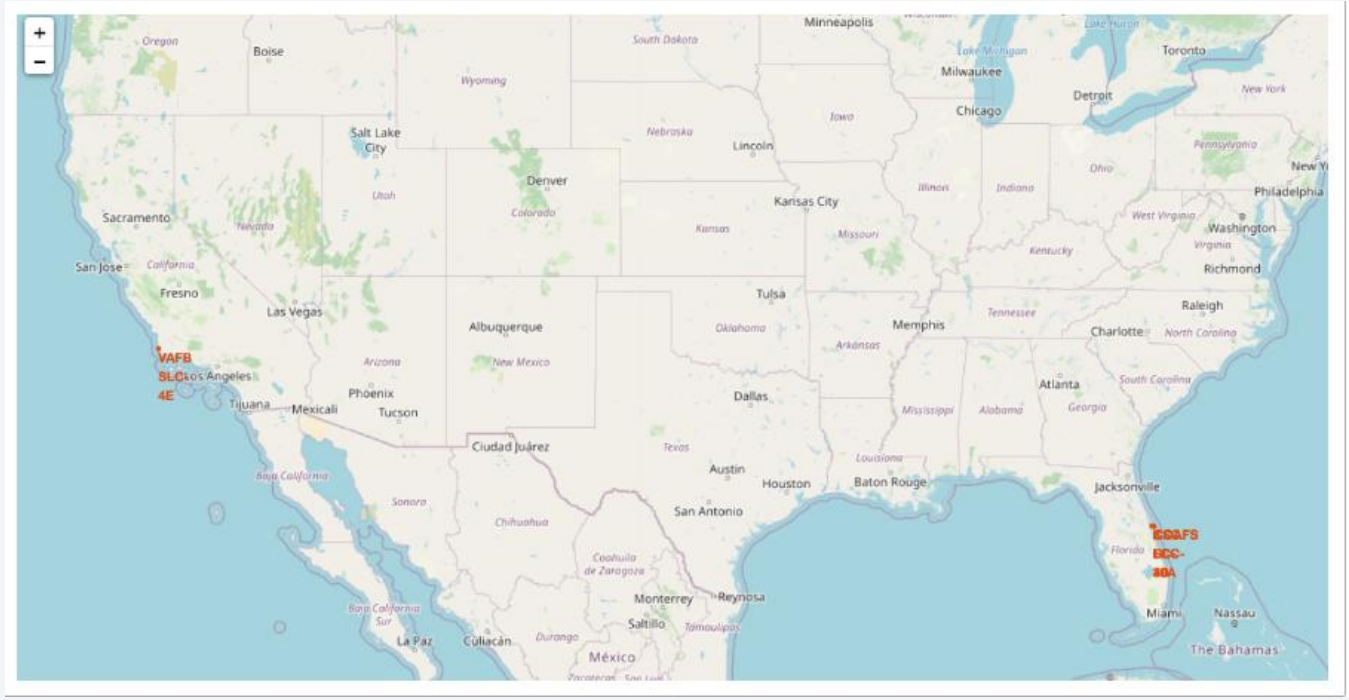
The BETWEEN operator is used to test whether an expression is within a range of values.

The above query is used to rank the Landing outcome between 2010-06-04 and 2017-03-20


Section 3
Launch Sites
Proximities Analysis

Global map for launch site location marker

All Launch Sites Location Marker for launch sites near USA Florida and California

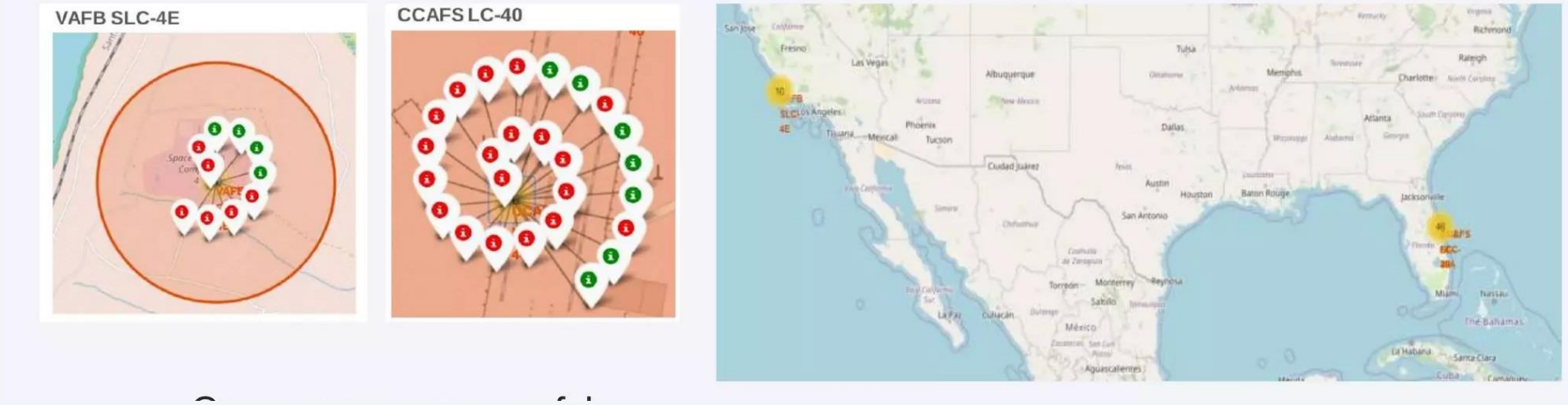


We observe that SpaceX Launch sites are located in coastal areas of United States of America, Florida and California

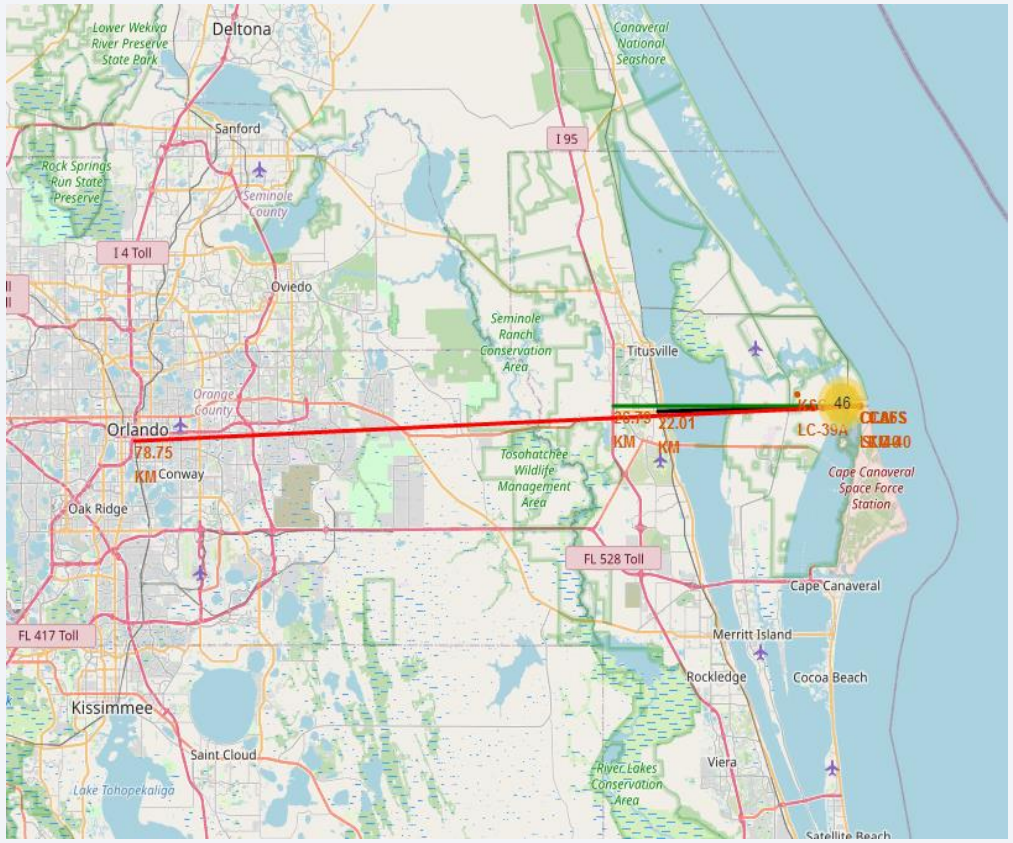
Color-labeled launch outcomes

Florida Launch Site:

Green marker shows successful launch and red marker shows failure

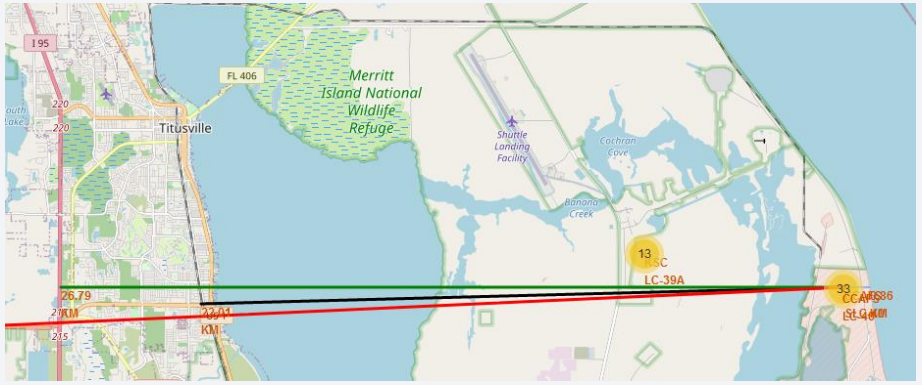


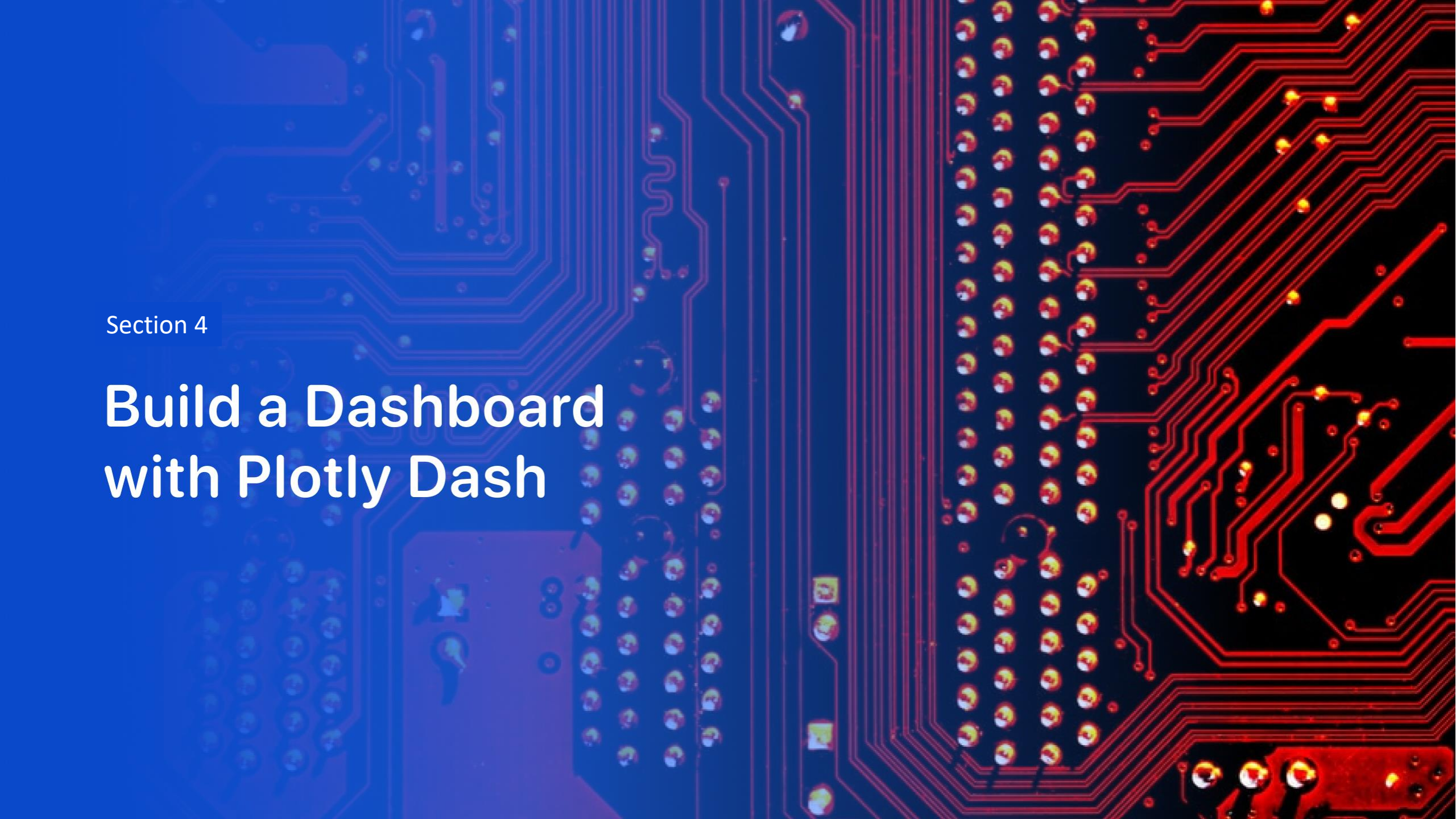
Distance of launch site to its proximities



Distance from CCAFS_SLC40 to:

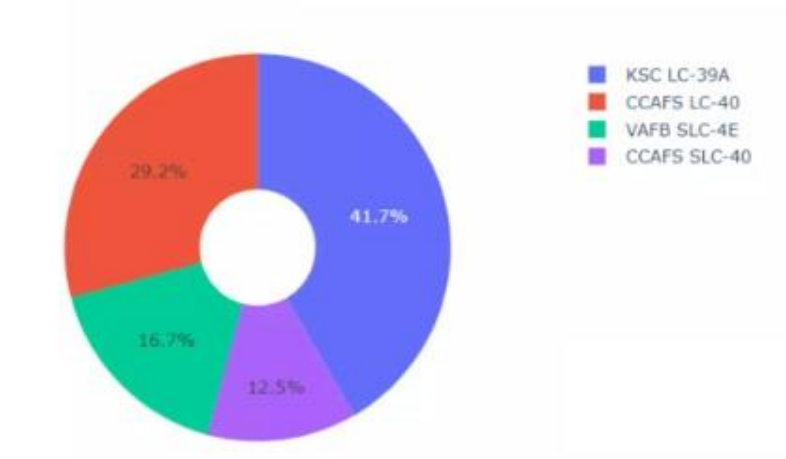
- Closest coast: ~900 m
- Florida East Coast Railway: 22.0 km
- Highway I 95: 26.8 km
- Orlando: 78.75 km




Section 4
Build a Dashboard with Plotly Dash

Count of successful launches for all sites

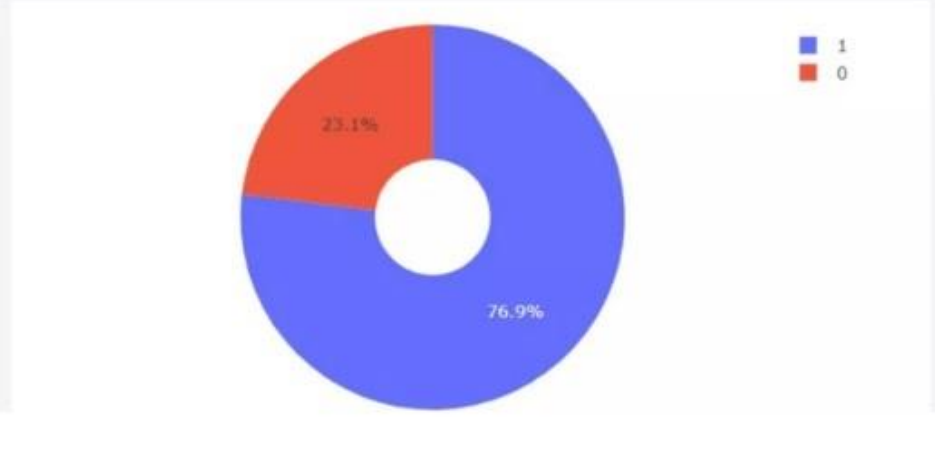
Pie chart showing success count for all sites



From the above chart we observe that KSC LC-39A as the successful launches of 41.7%, followed by CCAFF LC-40 which has successful launches of 29.2%, CCAFS SLC-40 has the lowest successful launches of 12.5%

Launch site with highest launch success ratio

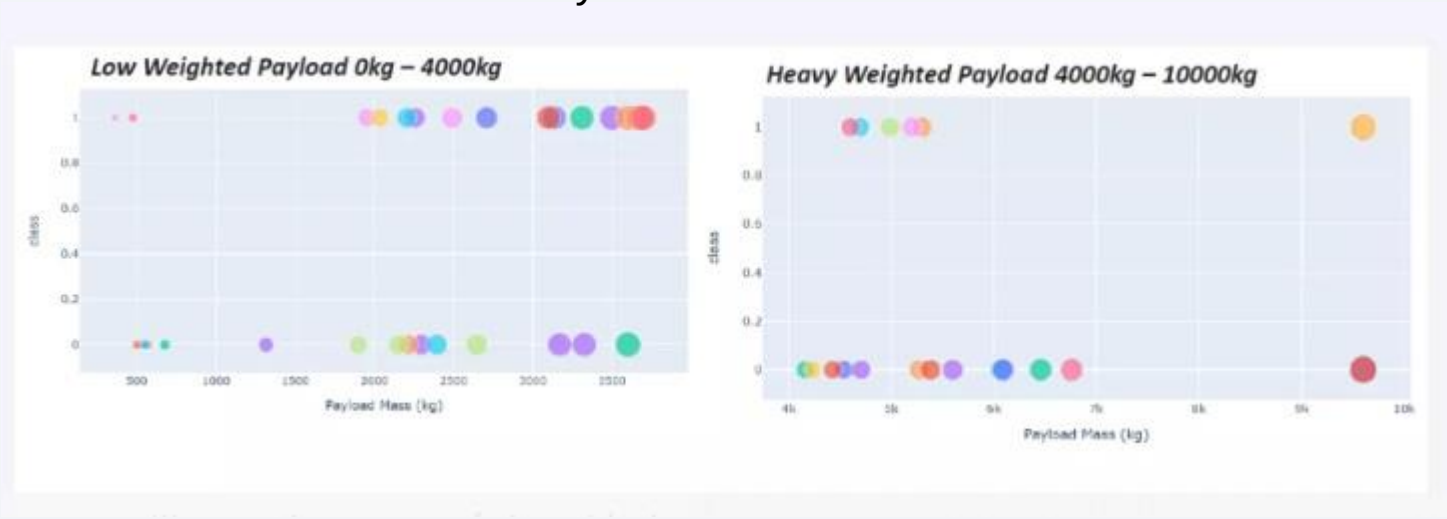
Pie chart showing launch site with highest launch success ratio



From the above chart, we observe that KSC LC-39A has the highest launch success rate of 76.9% where as its the failure rate is 23.1%

Payload vs. Launch Outcome

Scatter Plot of Payload vs. Launch Outcome

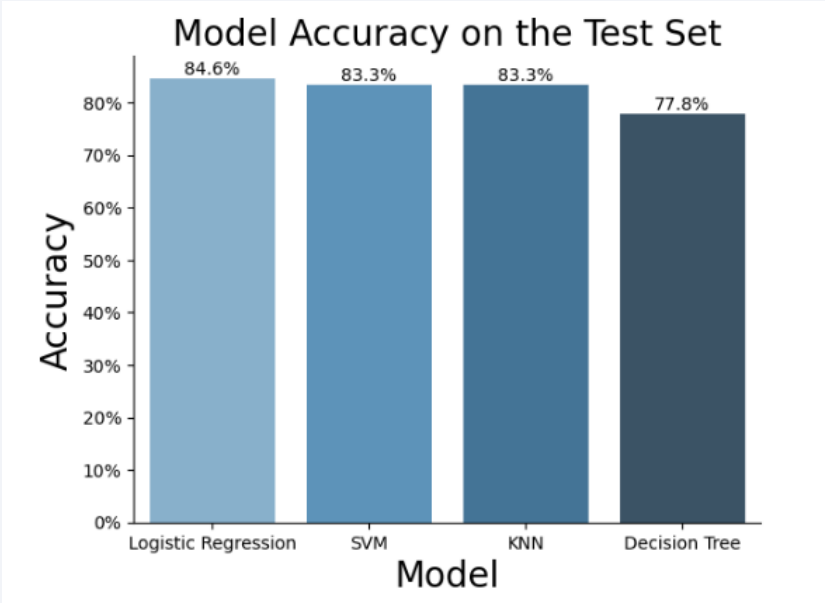


The above Scatter Plot shows the Payload vs. Launch Outcome for all sites, with different payload selected in the range slider. We can observe that success rate for low weighted payloads is higher than the heavy weighted payload.

Section 5

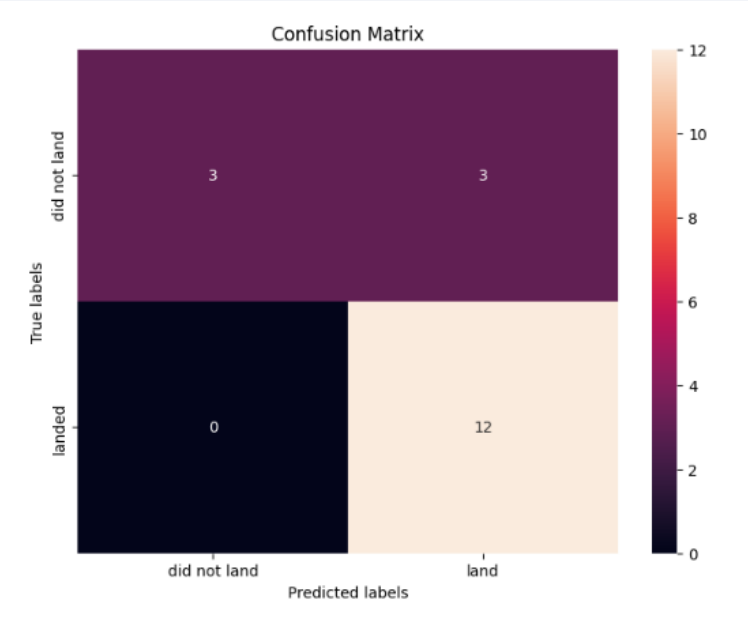
Predictive Analysis (Classification)

Classification Accuracy



From the bar chart we observe that Logistic Regression has the heighest classification accuracy of 84.6%, followed by SVM and KNN each having classification accuray of 83.3% respectively. Finally the classification accuracy of Decission Tree is 77.8% which is lowest among all.

Confusion Matrix



	F1-Score	Precision	Recall	Accuracy
Logistic Regression	0.889	0.8	1.00	0.846
SVM	0.889	0.8	1.00	0.833
KNN	0.889	0.8	1.00	0.833
Decision Tree	0.818	0.9	0.75	0.778

Explanation:

The best results are achieved using Logistic Regression, with accuracy of 84%.

Examining the confusion matrix, we see that logistic regression can distinguish between the different classes. We see that the major problem is false positives

Conclusions

- Low weighted payloads perform better than high weighted payloads
- Success rate is directly proportional to time (in years)
- Success rate may strongly depend payload mass and orbit.
- CCAFS-SLC 40 is the most used launch site.
- Launches from the site of CCAFS SLC 40 are significantly higher than launches form other sites
- CCAFS-SLC 40 has most of failures specially in the early stage
- Orbit GEO, HEO, SSO, ES-L1 has the highest success rate
- GTO has the lowest success rate where as SSO (polar low orbit) the highest one.
- We can conclude that Logistic Regression classification algorithm has the best performance with accuracy of 84.6%.
- The classification algorithms SVM and KNN has prediction accuracy of 83.3% and the classification accuracy of Decision Tree is 77.8% .
- For the given dataset, SVM, KNN, Logistic Regression model achieve best in terms of predictive accuracy

Appendix

GitHub Repository Link for IBM Data Science Capstone project :

https://github.com/shehla1739/Applied-Data-Science-Capstone

Thank you!