IBM DATA SCIENCE CAPSTONE PROJECT SPACEX

presented by JIMMY MPAWENIMANA

JANNURY 2023

Outline

Executive Summary

Summary of methodologies

-Data collection

- Data wrangling
- EDA with data visualization
- EDA with SQL
- Building an interactive map with Folium
- Building a Dashboard with Plotly Dash
- Predictive analysis (Classification00 x 717 px

Summary of all results

-Exploratory data analysis results

- Interactive analytics demo in screenshots
- Predictive analysis results


AM
12 43
METHODOLOGY

METHODOLOGY

Data collection methodology:

. SpaceX Rest API

- (Web Scrapping) from Wikipedia

Data wrangling:

One Hot Encoding data fields for Machine Learning and dropping irrelevant columns

Performed (EDA) using visualization and SQL

Plotting : Scatter Graphs, Bar Graphs to show relationships between variables to show patterns of data.

Performed interactive and predictive visual analytics

. using Folium and Plotly Dash

. classification models

METHODOLOGY

The following datasets was collected by :

- We worked with SpaceX launch data that is gathered from the SpaceX REST API.
- This API will give us data about launches, including information about the rocket used, payload delivered, launch specifications, landing specifications, and landing outcome.
- The goal is to use this data to predict whether SpaceX will attempt to land a rocket or not.
- The SpaceX REST API endpoints, or URL, starts with api.spacexdata.com/v4/.
- Another popular data source for obtaining Falcon 9 Launch data is web scraping Wikipedia using BeautifulSoup.

Data collection – SpaceX API

- Use SpaceX REST API
- API returns SpaceX data in .JSON
- Normalize data into flat data file such as .csv

simplified flow chart

1 .Getting Response from API

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

2. Converting Response to a .json file

```
response = requests.get(static_json_url).json()
data = pd.json_normalize(response)
```

3. Apply custom functions to clean data

```
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
```

```
getBoosterVersion(data)
```

4. Assign list to dictionary then dataframe

```
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}
```

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

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

```
data_falcon9 = df.loc[df['BoosterVersion']!="Falcon 1"]
```

```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data collection - Web Scrapping

- Use SpaceX REST API
- API returns SpaceX data in .JSON
- Parse HTML table into a list--> dictionary
- Normalize data into flat data file such as .csv

simplified flow chart

1 .Getting Response from HTML

```
page = requests.get(static_url)
```

2. Creating BeautifulSoup Object

```
soup = BeautifulSoup(page.text, 'html.parser')
```

3. Finding tables

```
html_tables = soup.find_all('table')
```

4. Getting column names

```
column_names = []
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
```

5. Creation of dictionary

```
launch_dict= dict.fromkeys(column_names)

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


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'] = []
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

6. Appending data to keys (refer) to notebook block 12

```
In [12]: extracted_row = 0
         #Extract each table
         for table_number,table in enumerate(
            # get table row
             for rows in table.find_all("tr")
                 #check to see if first table
```

7. Converting dictionary to dataframe

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

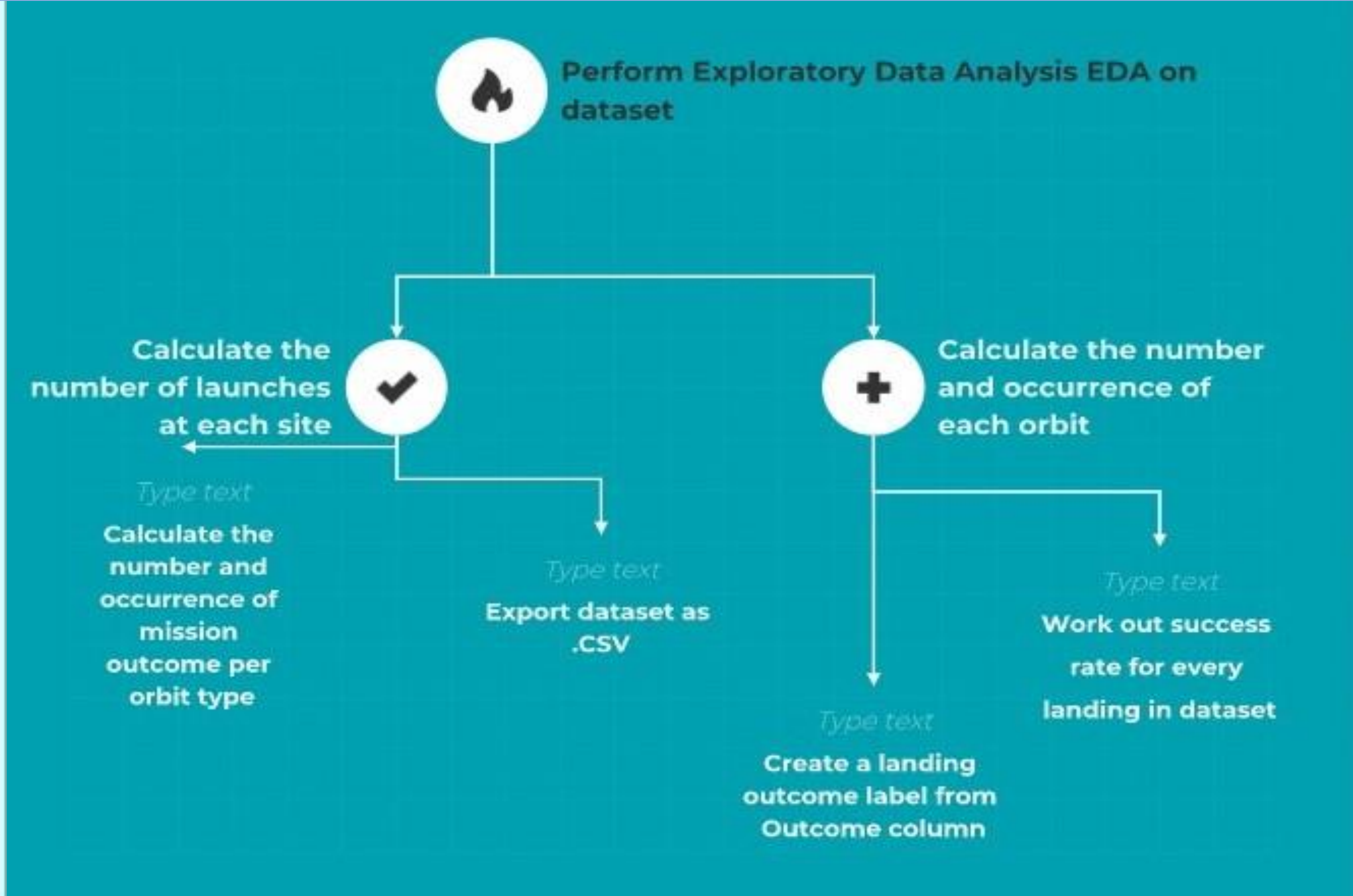
8. Dataframe to .CSV

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

Data Wrangling

Introduction

In the dataset, there are several different cases where the booster did not land successfully. Sometimes a landing was attempted but failed due to an accident; for example, True Ocean means the mission outcome was successfully landed to a specific region of the ocean while False Ocean means the mission outcome was unsuccessfully landed to a specific region of the ocean. True RTLS means the mission outcome was successfully landed to a ground pad False RTLS means the mission outcome was unsuccessfully landed to a ground pad. True ASDS means the mission outcome was successfully landed on a drone ship False ASDS means the mission outcome was unsuccessfully landed on a drone ship.We mainly convert those outcomes into Training Labels with 1 means the booster successfully landed 0 means it was unsuccessful.



EDA with data visualization

- Scatter Graphs being drawn:
  Scatter plots show how much one variable is affected by another. The relationship between two variables is called their correlation . Scatter plots usually consist of a large body of data
- Bar Graph being drawn:
  A bar diagram makes it easy to compare sets of data between different groups at a glance. The graph represents categories on one axis and a discrete value in the other. The goal is to show the relationship between the two axes. Bar charts can also show big changes in data over time.
- Line Graph being drawn:
  Line graphs are useful in that they show data variables and trends very clearly and can help to make predictions about the results of data not yet recorded

Scatter Graphs being drawn:

1. *Flight Number VS. Payload Mass*
2. *Flight Number VS. Launch Site*
3. *Payload VS. Launch Site*
4. *Orbit VS. Flight Number*
5. *Payload VS. Orbit Type*
6. *Orbit VS. Payload Mass*

Bar Graph being drawn:

Mean VS. Orbit

Line Graph being drawn:

Success Rate VS. Year

EDA with SQL

Performed SQL queries to gather information about the dataset.

For example of some questions we were asked about the data we needed information about. Which we are using SQL queries to get the answers in the dataset :

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

Building An Interactive Map With Folium

Visualizing the Launch Data into an interactive map. We took the Latitude and Longitude Coordinates at each launch site and added a Circle Marker around each launch site with a label of the name of the launch site.

Assigned the dataframe launch_outcomes(failures, successes) to classes 0 and 1 with Green and Red markers on the map in a MarkerCluster()

Using Haversine's formula we calculated the distance from the Launch Site to various landmarks to find various trends about what is around the Launch Site to measure patterns. **Lines** are drawn on the map to measure distance to landmarks

Example of some trends in which the Launch Site is situated in.

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

Build a Dashboard with Plotly Dash

1. Built a Plotly Dash web application to perform interactive visual analytics on SpaceX launch data in real-time. Added Launch Site Drop-down, Pie Chart, Payload range slide, and a Scatter chart to the Dashboard.

- ❖Added a Launch Site Drop-down Input component to the dashboard to provide an ability to filter Dashboard visual by all launch sites or a particular launch site
- ❖ Added a Pie Chart to the Dashboard to show total success launches when ‘All Sites’ is selected and show success and failed counts when a particular site is selected
- ❖ Added a Payload range slider to the Dashboard to easily select different payload ranges to identify visual patterns
- ❖Added a Scatter chart to observe how payload may be correlated with mission outcomes for selected site(s). The color-label Booster version on each scatter point provided missions outcomes with different boosters

2. Dashboard helped answer following questions:

   1. Which site has the largest successful launches? **KSC LC-39A with 10**
   2. Which site has the highest launch success rate? **KSC LC-39A with 76.9% success**
   3. Which payload range(s) has the highest launch success rate? **2000 – 5000 kg**
   4. Which payload range(s) has the lowest launch success rate? **0-2000 and 5500 - 7000**
   5. Which F9 Booster version (v1.0, v1.1, FT, B4, B5, etc.) has the highest launch success rate? **FT**

Predictive Analysis (Classification)

1. Read dataset into Dataframe and create a ‘Class’ array
2. Standardize the data
3. Train/Test/Split data in to training and test data sets
4. Create and Refine Models
5. Find the best performing Model

1. Load SpaceX dataset (csv) in to a Dataframe and create NumPy array from the column class in data

```
data = pd.read_csv("https://cf-courses-data.s3.us.cloud-object
et_part_2.csv")
```

```
Y = data['Class'].to_numpy()
```

2. Standardize data in X then reassign to variable X using transform

```
X= preprocessing.StandardScaler().fit(X).transform(X)
```

3. Train/test/split X and Y in to training and test data sets.

```
# Split data for training and testing data sets
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split
( X, Y, test_size=0.2, random_state=2)
print ('Train set:', X_train.shape,  Y_train.shape)
print ('Test set:', X_test.shape,  Y_test.shape)
```

4. Create and refine Models based on following classification Algorithms: (below is LR example)
   i. Create Logistic Regression object and then create a GridSearchCV object
   ii. Fit train data set in to the GridSearchCV object and train the Model

```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}
LR = LogisticRegression()
logreg_cv = GridSearchCV(LR, parameters,cv=10)
logreg_cv.fit(X_train, Y_train)
```

   iii. Find and display best hyperparameters and accuracy score

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

   iv. Check the accuracy on the test data by creating a confusion matrix

```
yhat=logreg_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

   v. Repeat above steps for Decision Tree, KNN, and SVM algorithms

5. Find the method performs best:

```
Model_Performance_df = pd.DataFrame({'Algo Type': ['Logistic Regression', 'SVM','Decision Tree','KNN'],
'Accuracy Score': [logreg_cv.best_score_, svm_cv.best_score_, tree_cv.best_score_, knn_cv.best_score_],
'Test Data Accuracy Score': [logreg_cv.score(X_test, Y_test), svm_cv.score(X_test, Y_test),
tree_cv.score(X_test, Y_test), knn_cv.score(X_test, Y_test)]})
```

```
Model_Performance_df.sort_values(['Accuracy Score'], ascending = False, inplace=True)
```

```
Model_Performance_df
```

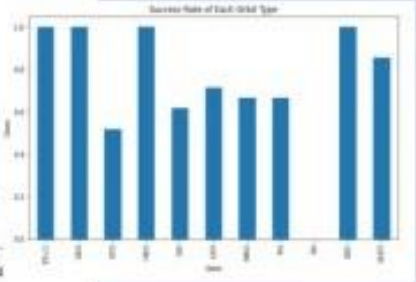
]:

	Algo Type	Accuracy Score	Test Data Accuracy Score
2	Decision Tree	0.889286	0.888889
3	KNN	0.848214	0.833333
1	SVM	0.848214	0.833333
0	Logistic Regression	0.846429	0.833333

Results

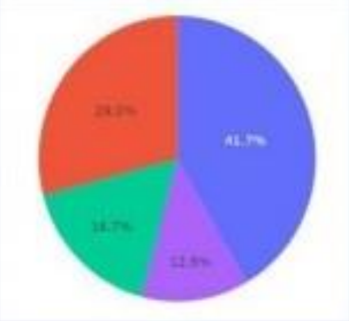
Exploratory data analysis results

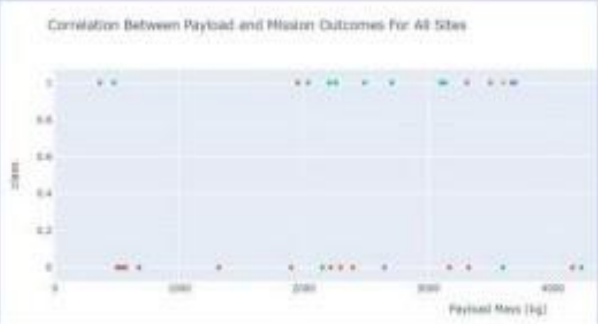
- Samples:



Interactive analytics demo in screenshots

- Samples





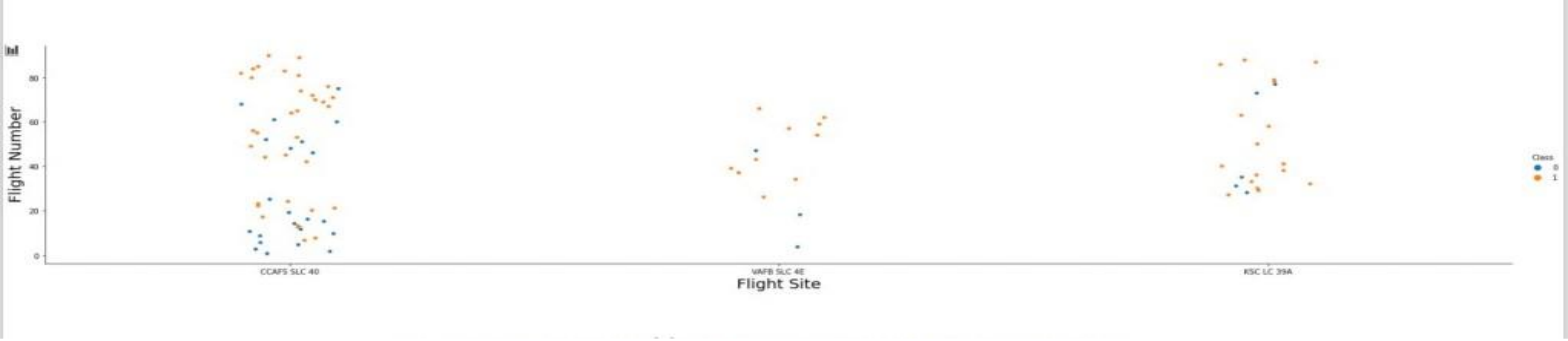
Predictive analysis results

- Samples

	Algo Type	Accuracy Score	Test Data Accuracy Score
2	Decision Tree	0.876786	0.722222
3	KNN	0.848214	0.833333
1	SVM	0.848214	0.833333
0	Logistic Regression	0.846429	0.833333

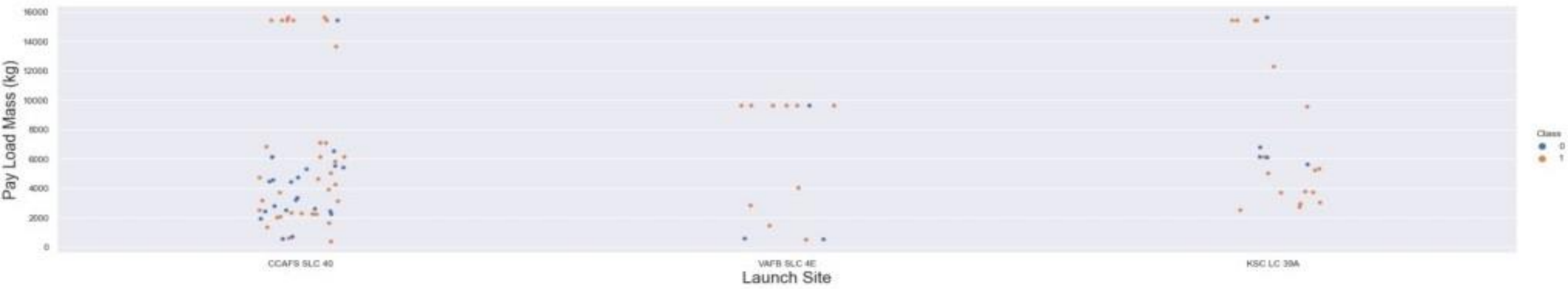

AM
12 43
EDA with visualization

Flight Number vs. Flight Site



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

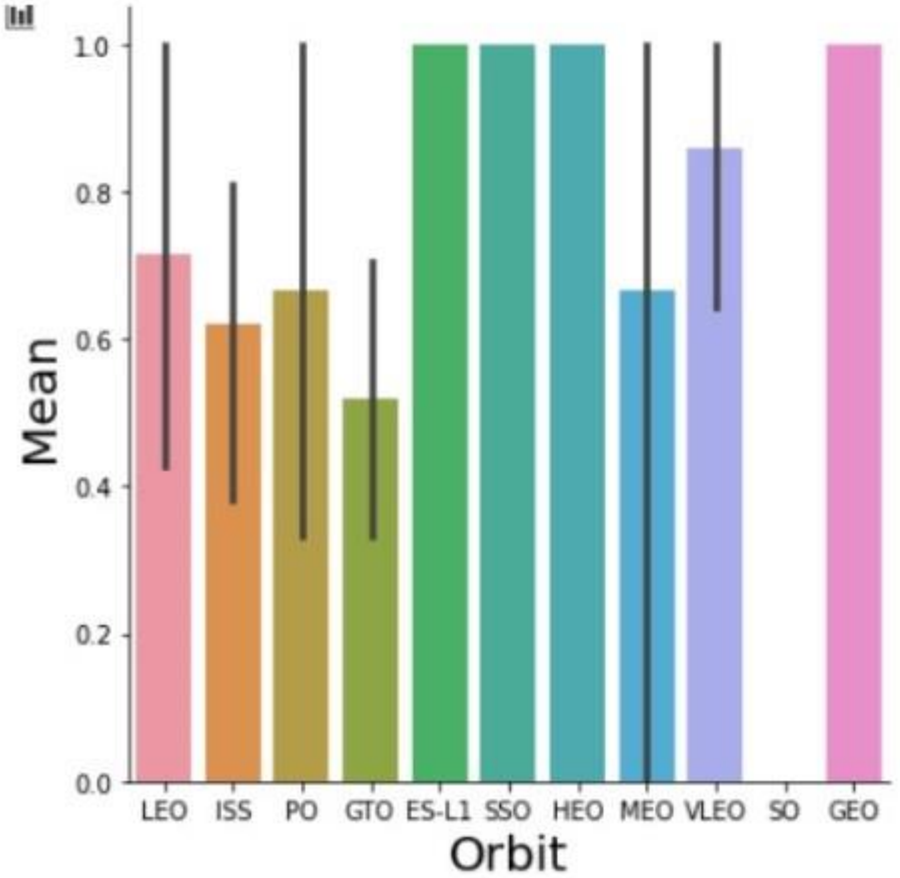
Payload Mass vs. Launch Site



The greater the payload mass for Launch Site CCAFS SLC 40 the higher the success rate for the Rocket. There is not quite a clear pattern to be found using this visualization to make a decision if the Launch Site is dependent on Pay Load Mass for a success launch

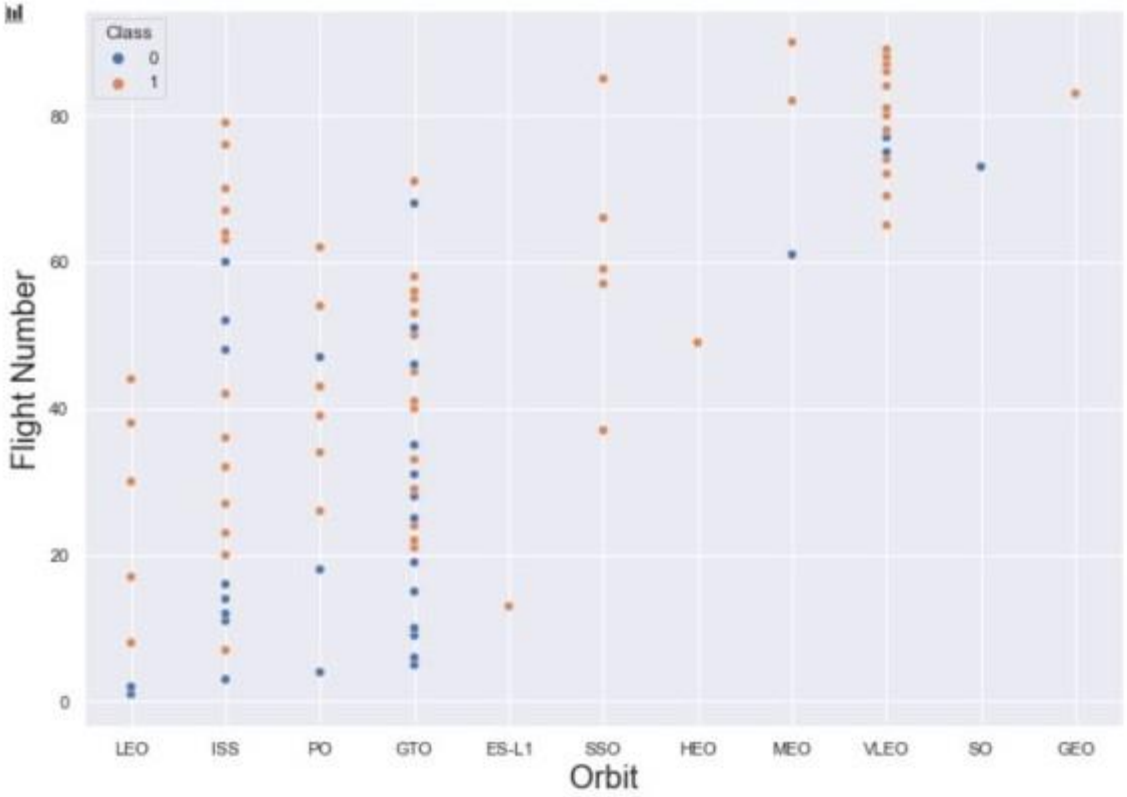
Success rate vs. Orbit type

Orbit
GEO,HEO,SSO,ES-L1
has the best Success Rate



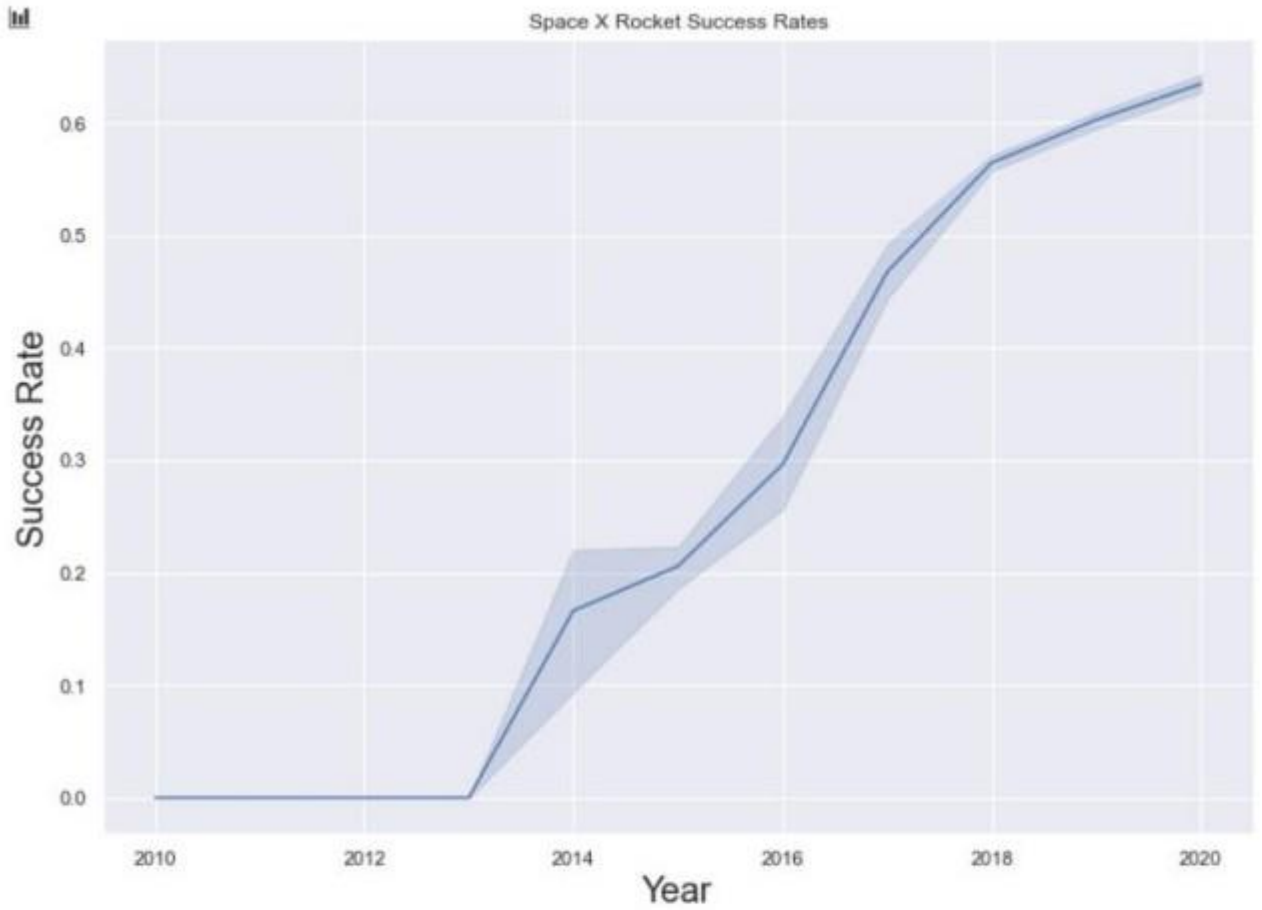
Flight Number vs. Orbit type

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



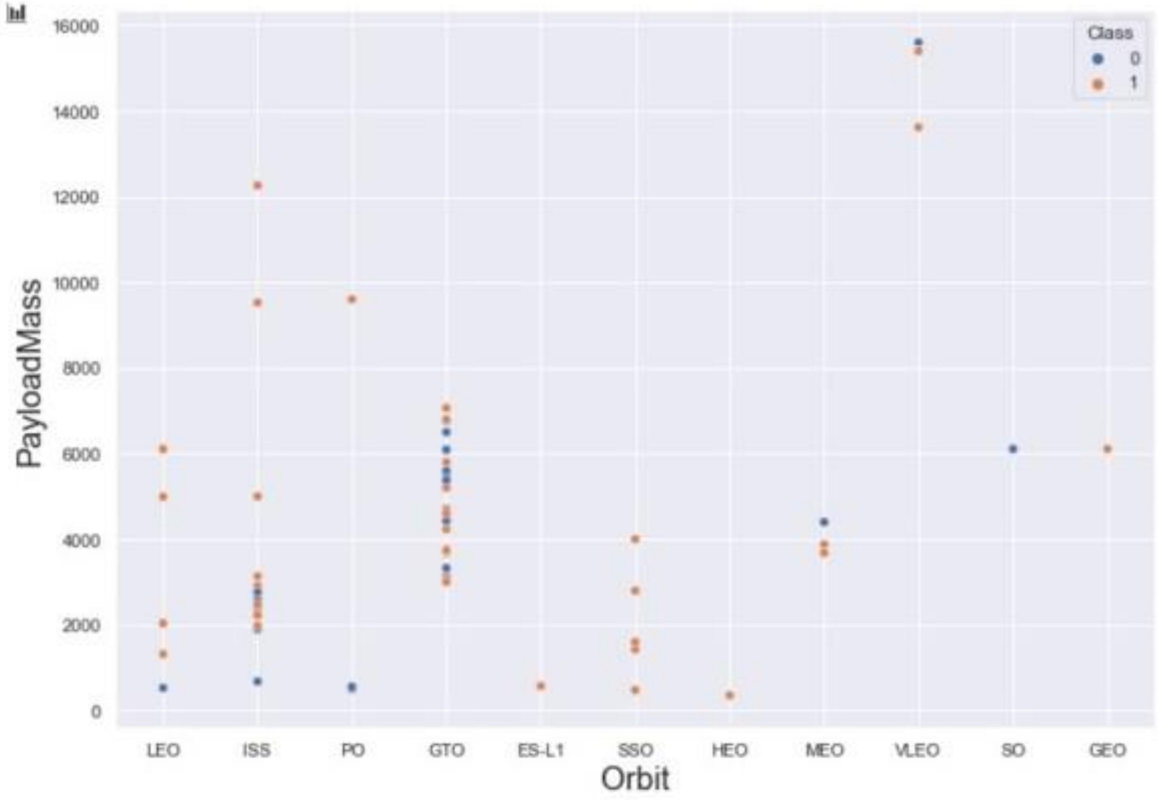
Payload vs. Orbit type

you can observe that the success rate since 2013 kept increasing till 2020



Launch success yearly trend

You should observe that Heavy payloads have a negative influence on GTO orbits and positive on GTO and Polar LEO (ISS) orbits.




EDA with SQL

Unique Launch Site

SQL QUERY: select DISTINCT Launch_Site from tblSpaceX

Unique Launch Sites
CCAFS LC-40
CCAFS SLC-40
CCAFS SLC-40
KSC LC-39A
VAFB SLC-4E

QUERY EXPLAINATION: Using the word ***DISTINCT*** in the query means that it will only show Unique values in the ***Launch_Site*** column from ***tblSpaceX***

Launch site names begin with `CCA`

SQL QUERY: select TOP 5 * from tblSpaceX WHERE Launch_Site LIKE 'KSC%'

QUERY EXPLAINATION: Using the word **TOP 5** in the query means that it will only show 5 records from ***tblSpaceX*** and ***LIKE*** keyword has a wild card with the words ***'KSC%'*** the percentage in the end suggests that the Launch_Site name must start with KSC.

	Date	Time_UTC	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS_KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
0	19-02-2017	2021-07-02 14:39:00.0000000	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
1	16-03-2017	2021-07-02 06:00:00.0000000	F9 FT B1030	KSC LC-39A	EchoStar 23	5600	GTO	EchoStar	Success	No attempt
2	30-03-2017	2021-07-02 22:27:00.0000000	F9 FT B1021.2	KSC LC-39A	SES-10	5300	GTO	SES	Success	Success (drone ship)
3	01-05-2017	2021-07-02 11:15:00.0000000	F9 FT B1032.1	KSC LC-39A	NROL-76	5300	LEO	NRO	Success	Success (ground pad)
4	15-05-2017	2021-07-02 23:21:00.0000000	F9 FT B1034	KSC LC-39A	Inmarsat-5 F4	6070	GTO	Inmarsat	Success	No attempt

Total Payload Mass by Customer NASA (CRS)

SQL QUERY: select SUM(PAYLOAD_MASS_KG_) TotalPayloadMass from tblSpaceX where Customer = 'NASA (CRS)'",'TotalPayloadMass

	Total Payload Mass
0	45596

QUERY EXPLAINATION: Using the function ***SUM*** summates the total in the column ***PAYLOAD_MASS_KG_***

The ***WHERE*** clause filters the dataset to only perform calculations on ***Customer NASA (CRS)***

Average Payload Mass carried by booster version F9 v1.1

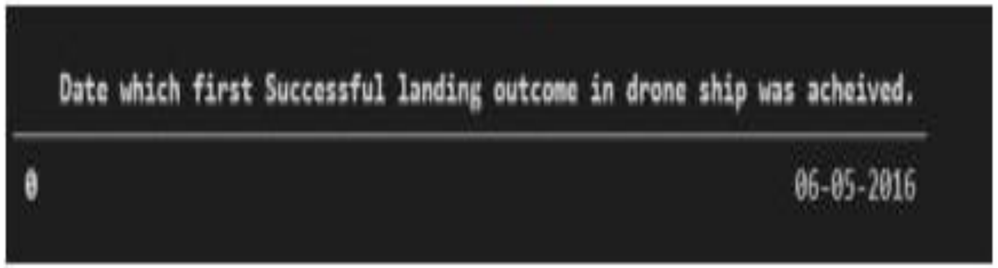
SQL QUERY: select AVG(PAYLOAD_MASS_KG_) AveragePayloadMass from tblSpaceX where Booster_Version = 'F9 v1.1'

	Average Payload Mass
0	2928

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

The date where the successful ground landing date 29 The date where the successful landing outcome in drone ship was achieved

SQL QUERY: select MIN(Date) SLO from tblSpaceX where Landing_Outcome = ”Success (drone ship)”



	Date which first Successful landing outcome in drone ship was acheived.
0	06-05-2016

QUERY EXPLAINATION: Using the function ***MIN*** works out the minimum date in the column ***Date***

The ***WHERE*** clause filters the dataset to only perform calculations on ***Landing_Outcome Success (drone ship)***

Successful drone ship landing with payload between 4000 and 6000

SQL QUERY: select Booster_Version from tblSpaceX where Landing_Outcome = 'Success (ground pad)' AND Payload_MASS_KG_ > 4000 AND Payload_MASS_KG_ < 6000

	Date which first Successful landing outcome in drone ship was acheived.
0	F9 FT B1032.1
1	F9 B4 B1040.1
2	F9 B4 B1043.1

QUERY EXPLAINATION: Selecting only ***Booster_Version***
The ***WHERE*** clause filters the dataset to ***Landing_Outcome = Success (drone ship)***
The ***AND*** clause specifies additional filter conditions ***Payload_MASS_KG_*** > 4000 AND ***Payload_MASS_KG_ < 6000***

Total Number of Successful and Failure Mission Outcomes

SQL QUERY: SELECT(SELECT Count(Mission_Outcome) from tblSpaceX where Mission_Outcome LIKE '%Success%’) as Successful_Mission_Outcomes, (SELECT Count(Mission_Outcome) from tblSpaceX where Mission_Outcome LIKE '%Failure%’) as Failure_Mission_Coutcomes

	Successful_Mission_Outcomes	Failure_Mission_Outcomes
0	100	1

QUERY EXPLAINATION: a much harder query I must say, we used subqueries here to produce the results. The LIKE ***‘%foo%’*** wildcard shows that in the record the ***foo*** phrase is in any part of the string in the records for example.

PHRASE “(Drone Ship was a Success)” LIKE ‘%Success%’ Word ‘Success’ is in the phrase the filter will include it in the dataset

Boosters carried maximum payload

SQL QUERY: SELECT DISTINCT Booster_Version, MAX(PAYLOAD_MASS _KG_) AS [Maximum Payload Mass] FROM tblSpaceX GROUP BY Booster_Version ORDER BY [Maximum Payload Mass] DESC

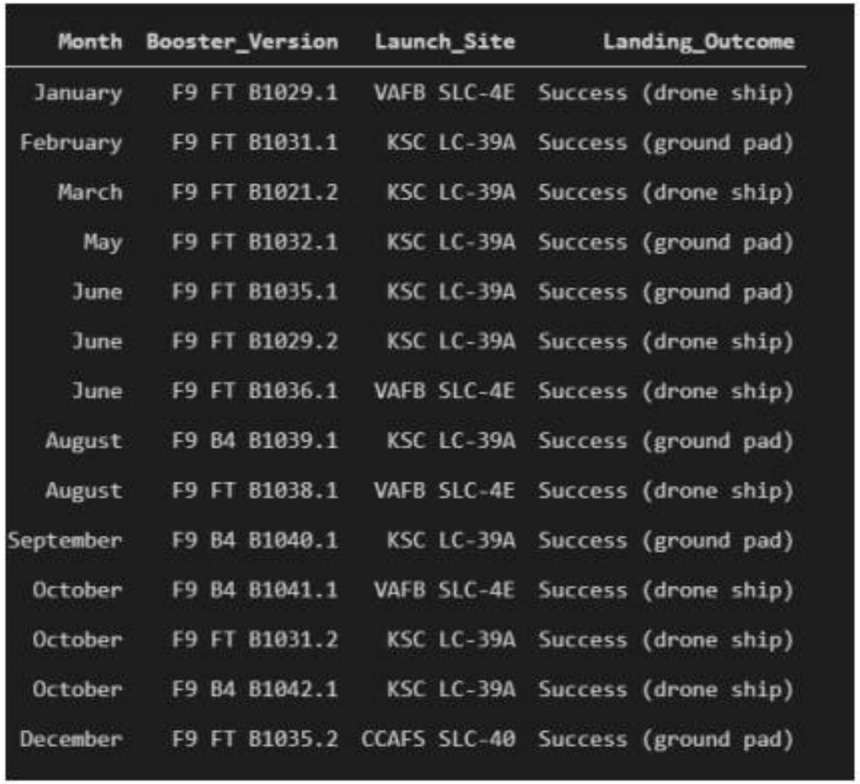
	Booster_Version	Maximum Payload Mass
0	F9 B5 B1048.4	15600
1	F9 B5 B1048.5	15600
2	F9 B5 B1049.4	15600
3	F9 B5 B1049.5	15600
4	F9 B5 B1049.7	15600
...	...	...
92	F9 v1.1 B1003	500
93	F9 FT B1038.1	475
94	F9 B4 B1045.1	362
95	F9 v1.0 B0003	0
96	F9 v1.0 B0004	0

97 rows × 2 columns

QUERY EXPLAINATION:
Using the word ***DISTINCT*** in the query means that it will only show Unique values in the ***Booster_Version*** column from ***tblSpaceX***
GROUP BY puts the list in order set to a certain condition.
DESC means its arranging the dataset into descending order

2017 Launch Records

SQL QUERY: SELECT DATENAME(month, DATEADD(month, MONTH(CONVERT(date, Date, 105)), 0) - 1) AS Month, Booster_Version, Launch_Site, Landing_Outcome FROM tblSpaceX WHERE (Landing_Outcome LIKE N'%Success%') AND (YEAR(CONVERT(date, Date, 105)) = '2017')

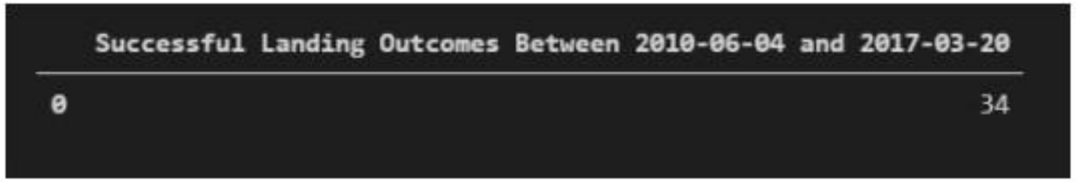


Month	Booster_Version	Launch_Site	Landing_Outcome
January	F9 FT B1029.1	VAFB SLC-4E	Success (drone ship)
February	F9 FT B1031.1	KSC LC-39A	Success (ground pad)
March	F9 FT B1021.2	KSC LC-39A	Success (drone ship)
May	F9 FT B1032.1	KSC LC-39A	Success (ground pad)
June	F9 FT B1035.1	KSC LC-39A	Success (ground pad)
June	F9 FT B1029.2	KSC LC-39A	Success (drone ship)
June	F9 FT B1036.1	VAFB SLC-4E	Success (drone ship)
August	F9 B4 B1039.1	KSC LC-39A	Success (ground pad)
August	F9 FT B1038.1	VAFB SLC-4E	Success (drone ship)
September	F9 B4 B1040.1	KSC LC-39A	Success (ground pad)
October	F9 B4 B1041.1	VAFB SLC-4E	Success (drone ship)
October	F9 FT B1031.2	KSC LC-39A	Success (drone ship)
October	F9 B4 B1042.1	KSC LC-39A	Success (drone ship)
December	F9 FT B1035.2	CCAFS SLC-40	Success (ground pad)

QUERY EXPLAINATION:
a much more complex query as I had my ***Date*** fields in SQL Server stored as ***NVARCHAR*** the ***MONTH*** function returns name month. The function ***CONVERT*** converts ***NVARCHAR*** to Date.
WHERE clause filters ***Year*** to be 2017

Rank success count between 2010-06-04 and 2017-03-20

SQL QUERY: SELECT COUNT(Landing_Outcome) FROM tblSpaceX WHERE (Landing_Outcome LIKE '%Success%’) AND (Date > '04-06-2010’) AND (Date < '20-03-2017')



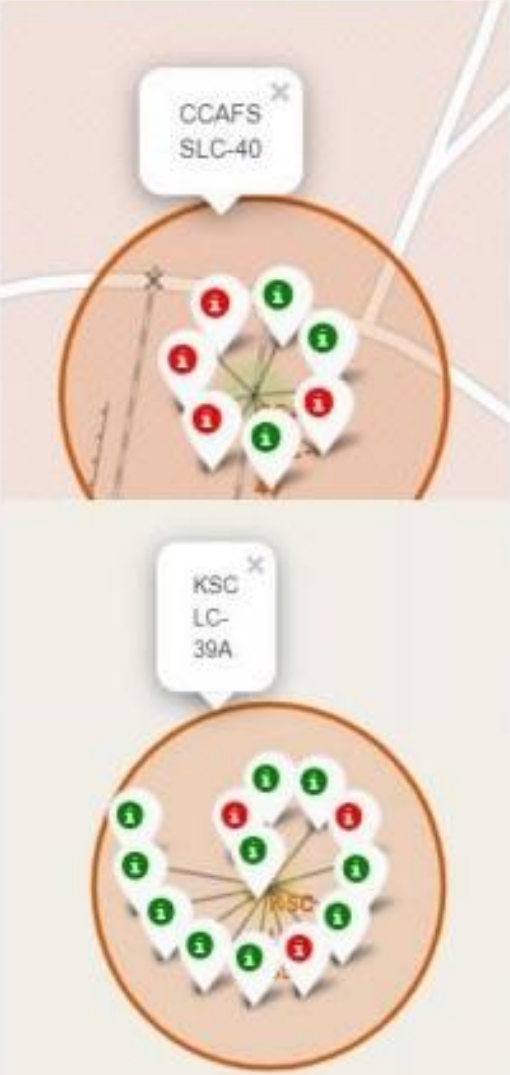
	Successful Landing Outcomes Between 2010-06-04 and 2017-03-20
0	34

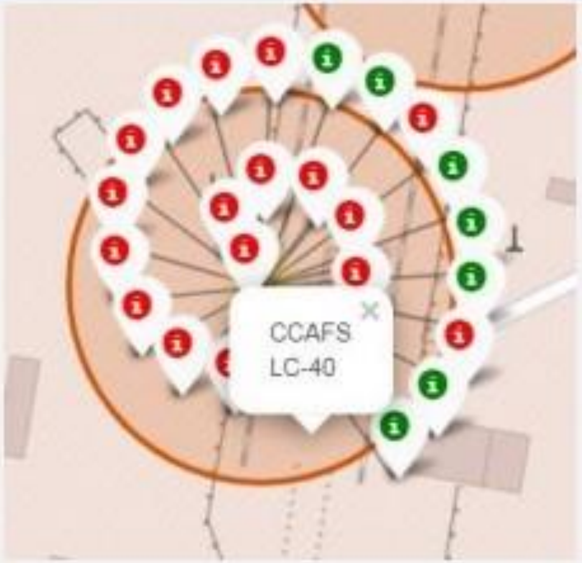
QUERY EXPLAINATION:
Function ***COUNT*** counts records in column
WHERE filters data
LIKE (wildcard)
AND (conditions)
AND (conditions)

Interactive map with Folium

All launch sites global map markers

Color Labelled Markers



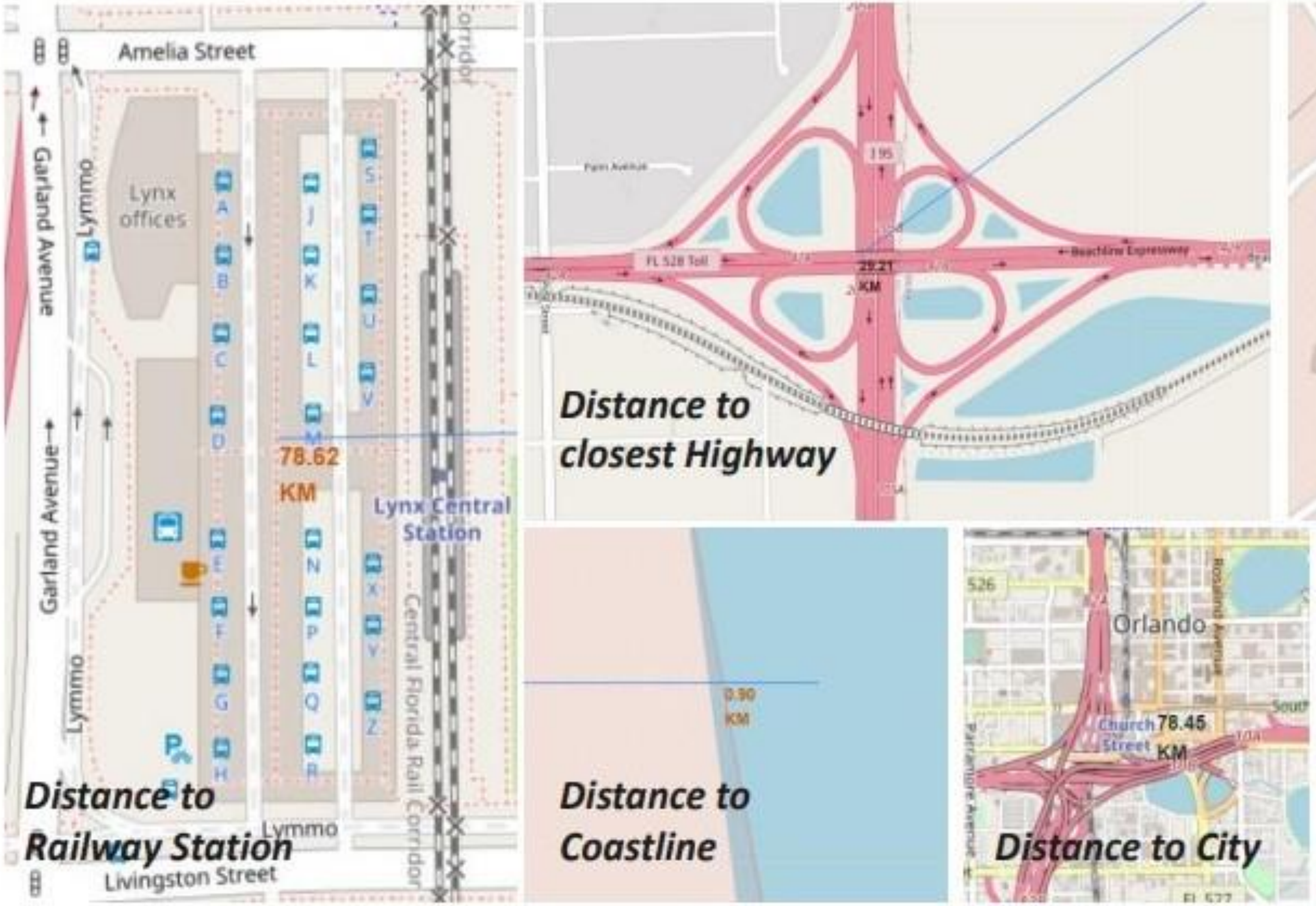


California Launch Site

Florida Launch Sites

Green Marker shows successful Launches and Red Marker shows Failures

Working out Launch Sites distance to landmarks to find trends with Haversine formula using CCAFS-SLC-40 as a reference

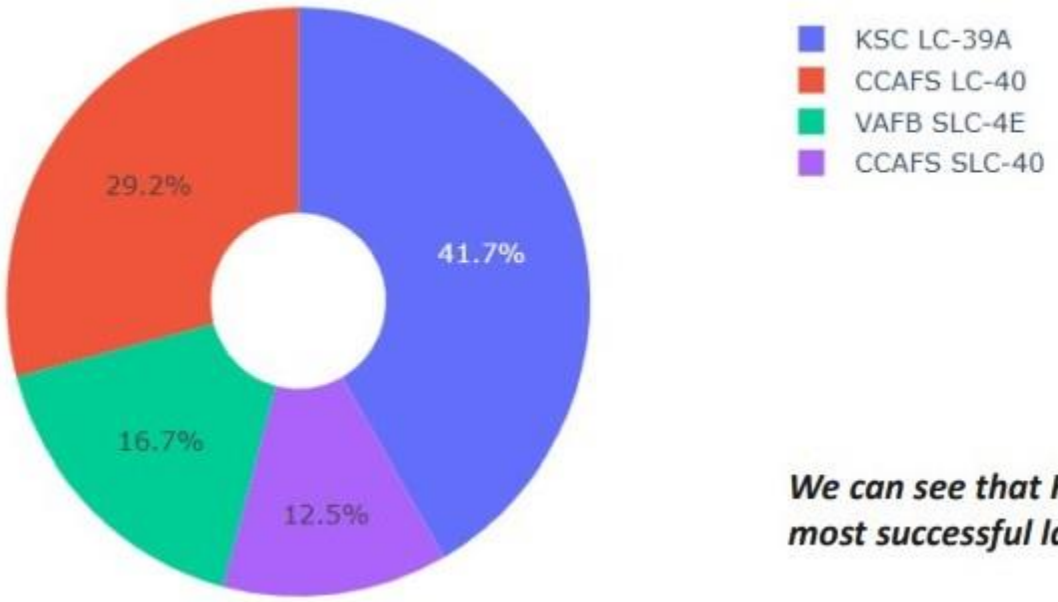


- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

Dashboard with Plotly Dash

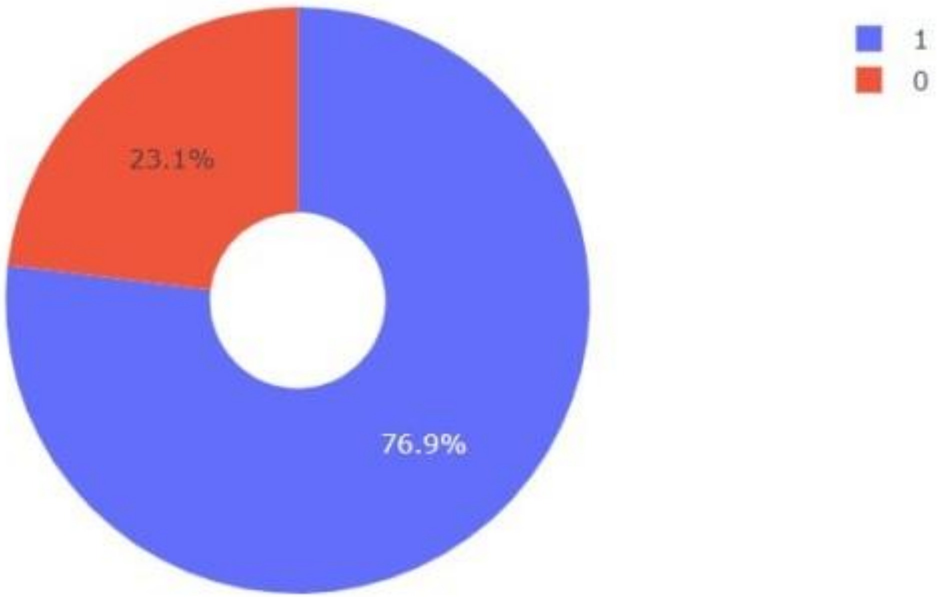
DASHBOARD – Pie chart showing the success percentage achieved by each launch site

Total success launches by all sites



We can see that KSC LC-39A had the most successful launches from all the sites

DASHBOARD – Pie chart for the launch site with highest launch success ratio

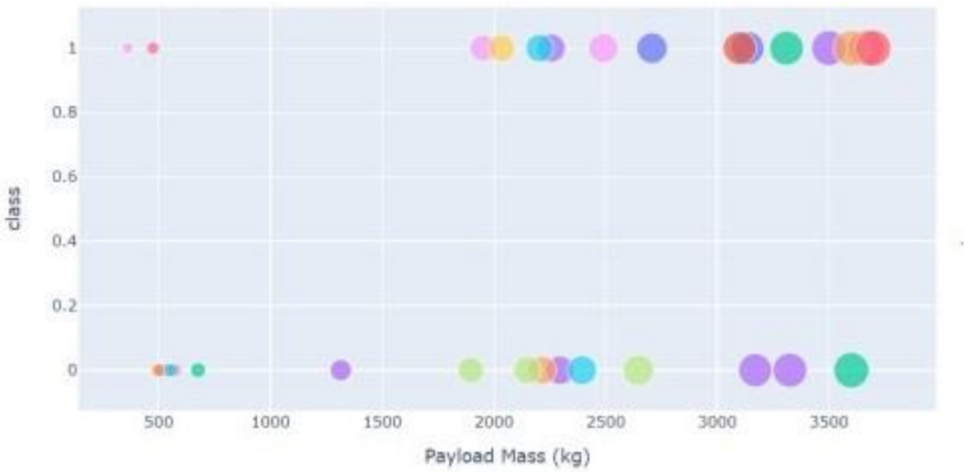


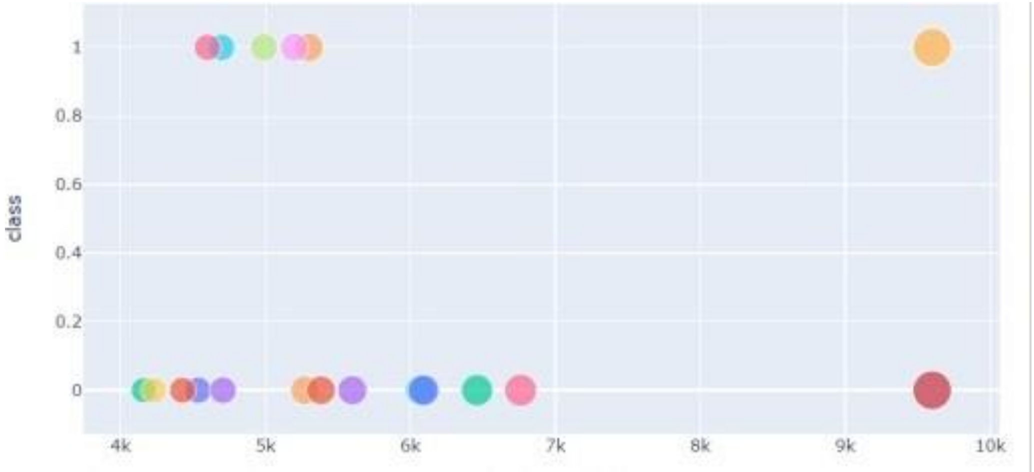
KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

DASHBOARD – Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider

Low Weighted Payload 0kg – 4000kg

Heavy Weighted Payload 4000kg – 10000kg

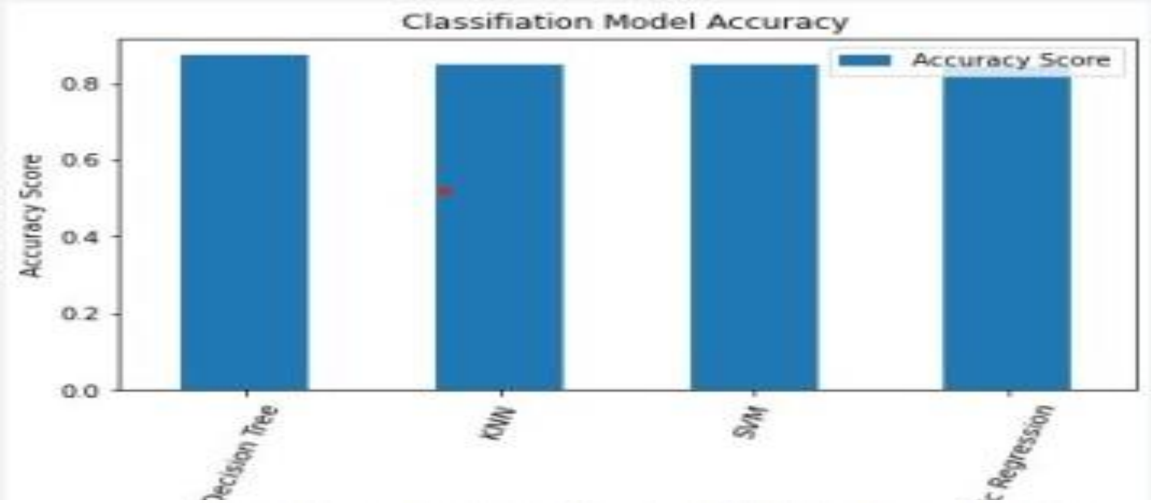




We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

Predictive analysis (Classification)

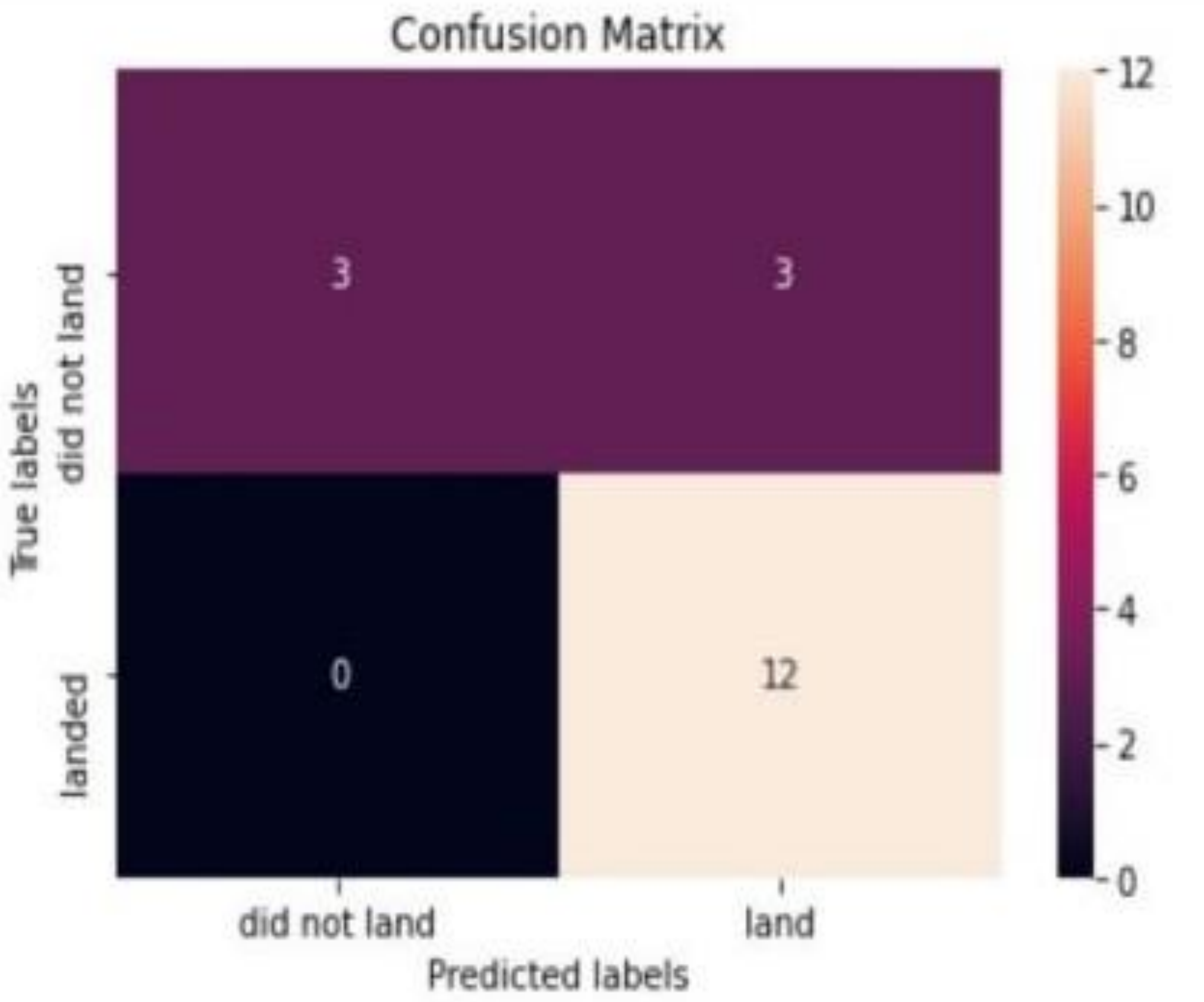
Classification Accuracy using training data



	Algo Type	Accuracy Score	Test Data Accuracy Score
2	Decision Tree	0.889286	0.888889
3	KNN	0.848214	0.833333
1	SVM	0.848214	0.833333
0	Logistic Regression	0.846429	0.833333

- Based on the Accuracy scores and as also evident from the bar chart, Decision Tree algorithm has the highest classification score with a value of 0.88
- Accuracy Score on the test data is almost same for all the classification algorithms based on the data set with a value of 0.8333
- Given that the Accuracy scores for Classification algorithms are very close and the test scores are the same, we may need a broader data set to further tune the models

Confusion Matrix



- The confusion matrix is same for all the models (LR, SVM, Decision Tree, KNN)
- Per the confusion matrix, the classifier made 18 predictions
- 12 scenarios were predicted Yes for landing, and they did land successfully (True positive)
- 3 scenarios (top left) were predicted No for landing, and they did not land (True negative)
- 3 scenarios (top right) were predicted Yes for landing, but they did not land successfully (False positive)
- Overall, the classifier is correct about 83% of the time ((TP + TN) / Total) with a misclassification or error rate ((FP + FN) / Total) of about 16.5% Confusion Matrix

CONCLUSION

- The Tree Classifier Algorithm is the best for Machine Learning for this dataset
- Low weighted payloads perform better than the heavier payloads
- The success rates for SpaceX launches is directly proportional time in years they will eventually perfect the launches
- We can see that KSC LC-39A had the most successful launches from all the sites
- Orbit GEO,HEO,SSO,ES-L1 has the best Success Rate