Data Science Capstone Project

Siddhartha Gadiparthi

23rd Aug 2022

Outline

- EXECUTIVE SUMMARY
- INTRODUCTION
- METHODOLOGY
- RESULTS
- CONCLUSION
- APPENDIX

Executive Summary

- ## Summary of methodologies

1. Data Collection Via SQL, API, Web Scraping

2. Data Wrangling and Analysis

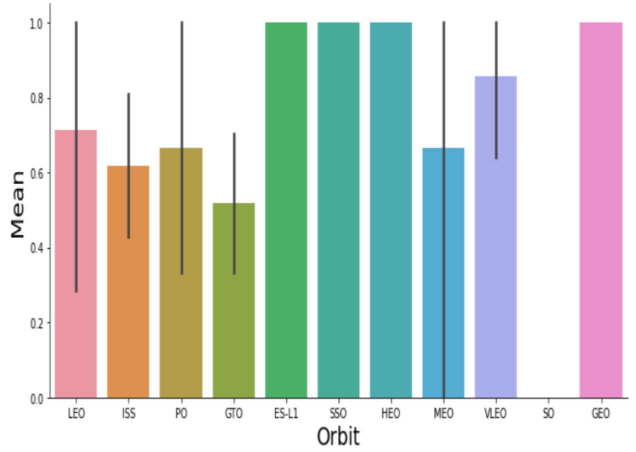
3. Interactive Maps with Folium

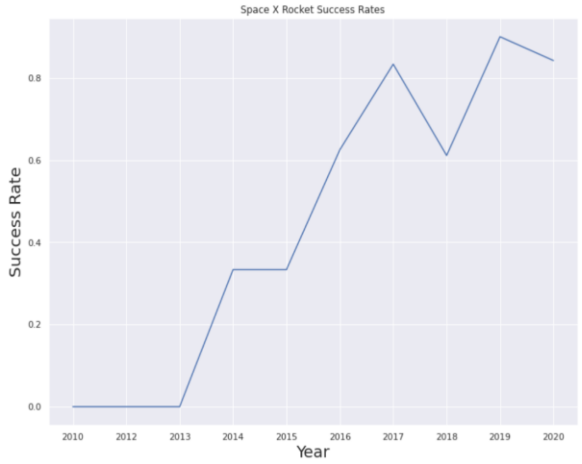
4.Predictive Analysis for each Classification Model

- ## Summary of all results

1. Data Analysis along with Interactive Visualization

2. Best Model for Predictive Analysis





Introduction

- **Project background and context**

SpaceX has gained worldwide attention for a series of historic milestones. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars whereas other providers cost upward of 165 million dollars each, much of the savings is because Space X can reuse the first stage. Therefore if we can determine if the first stage will land, we can determine the cost of a launch.

- **Problems you want to find answers**

1. With what factors, the rocket will land successfully?

2. The effect of each relationship of rocket variables on outcome.

3. Conditions which will aid SpaceX have to achieve the best results.

Methodology

Data collection methodology:(Describe how data was collected?)
- Via SpaceX RESTAPI
- Web Scrapping From WIKIPEDIA

Perform data wrangling (Describe how data was processed?)
- One hot encoding data fields for ML and dropping irrelevant columns (Tranforming data for ML)

Perform exploratory data analysis (EDA) using visualization and SQL
- Scatter and bar graphs to show patterns between data.

Perform interactive visual analytics
- Using Folium and Plotly Dash Data Visualizations

Perform predictive analysis using classification models
- Build, tune, evaluate classification models

Data Collection

It is a process of gathering and measuring information on targeted variables in an established system, which then enables one to answer relevant questions and evaluate outcomes.

DATA COLLECTION VIA SPACEX API :

1. Getting Response from API

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

```
response = requests.get(spacex_url)
```

2. Converting Response to a .json file

```
# Use json_normalize meethod to convert the json result into a dataframe
# Reading json file

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

3. Apply custom functions to clean data

```
# Call getBoosterVersion
getBoosterVersion(data)
```

the list has now been update

```
BoosterVersion[0:5]
```

we can apply the rest of the functions here:

```
# Call getLaunchSite
getLaunchSite(data)
```

```
# Call getPayloadData
getPayloadData(data)
```

```
# Call getCoreData
getCoreData(data)
```

4. Assign list to dictionary to create dataframe.

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
```

5. Filter dataframe and export to flat file

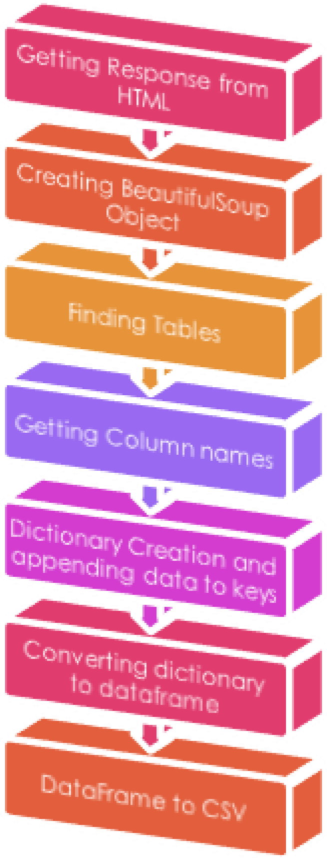
Github_Collection

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = data.drop(data[data['BoosterVersion']!='Falcon 9'].index)
data_falcon9
```

Now that we have removed some values we should reset the FlgihtNumber column

```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data Collection – Web Scraping




```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
data = requests.get(static_url).text
```

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(data, 'html5lib')
```

Print the page title to verify if the `BeautifulSoup` object was created properly

```
# Use soup.title attribute
soup.title.string
```

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

Starting from the third table is our target table contains the actual launch records.

```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

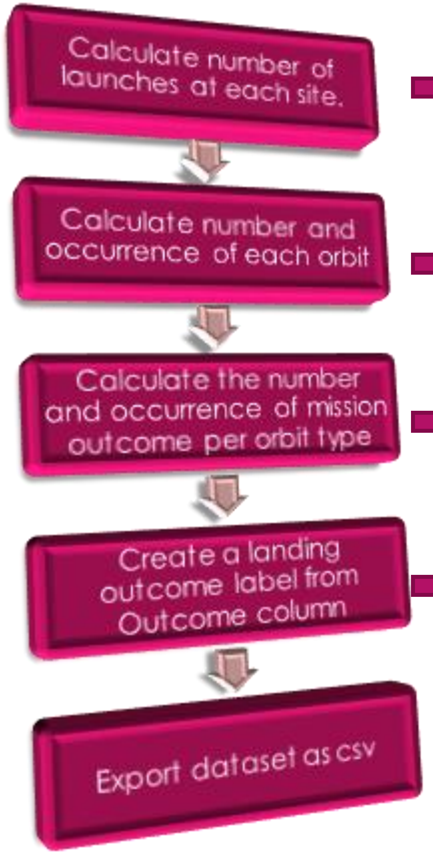
```
column_names = []

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

Github_Collection

Data Wrangling

It is the process of cleaning and unifying messing and complex data sets for easy access and analysis.



Github_Collection

```
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()
```

Use the method `.value_counts()` to determine the number and occurrence of each orbit in the column `Orbit`

```
# Apply value_counts on Orbit column
df['Orbit'].value_counts()
```

Use the method `.value_counts()` on the column `Outcome` to determine the number of `landing_outcomes` .Then assign it to a variable landing_outcomes.

```
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
landing_outcomes
```

Using the `Outcome` , create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome` ; otherwise, it's one. Then assign it to the variable `landing_class` :

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise

df['Class'] = df['Outcome'].apply(lambda landing_class: 0 if landing_class in bad_outcomes else 1)
df[['Class']].head(8)
```

Here we mainly convert those outcomes into Training Labels with `1` means the booster successfully landed `0` means it was unsuccessful.

EDA with Data Visualization

Exploratory data analysis is an approach of analyzing data sets to summarize their main characteristics using statistical graphics and other data visualization methods.

Scatter Graphs Drawn :

1. Flight Number vs PayloadMass

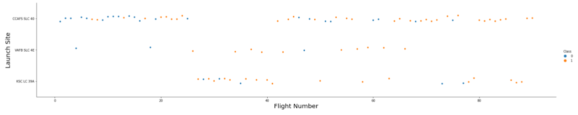
2. Flight Number and Launch Site

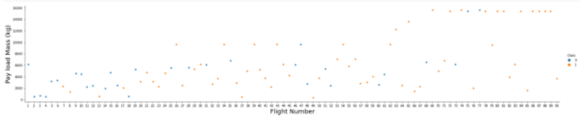
3. Payload and Launch Site

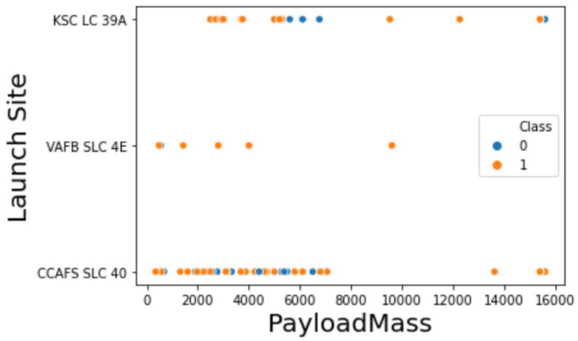
4. Flight Number and Orbit type

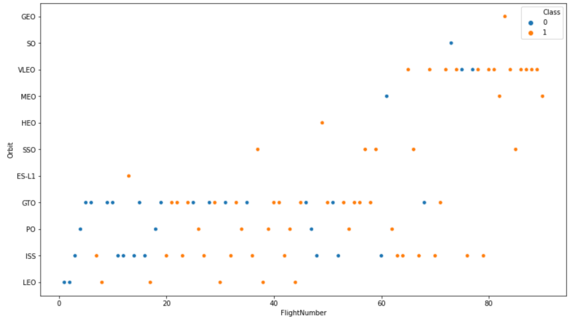
5. Payload and Orbit type

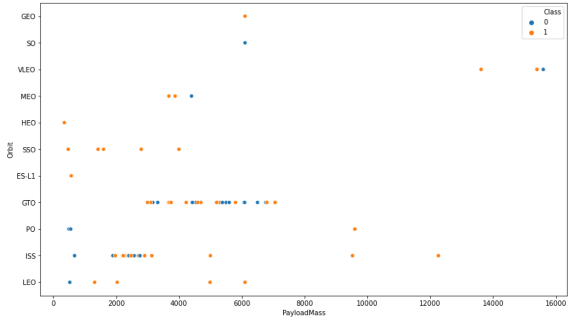
Scatter plot shows dependency of attributes on each other. Once a pattern is determined it is very easy to Predict which factors will lead to maximum probability of success in both outcome and landing.







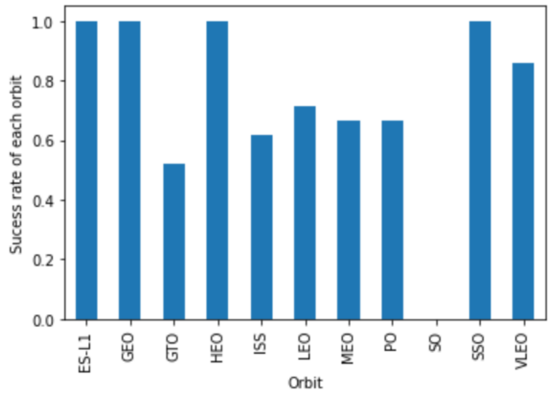




Bar Graph Drawn :

Bar graphs are easiest to interpret a relationship between attributes. Via this bar graph we can easily determine which orbits have the highest probability of success.

1. Success rate of each orbit type



[Github_Collection]

Line Graph Drawn :

Line graphs shows the trend clearly and Can aid the predictions for the future.

1. Launch success yearly trend

EDA with SQL

SQL is an indispensable tool for Data Scientists and Analyst as most of the real-world data is stored in datasets. It's not only the Standard Language for Relational database operations, but also an incredible tool for analyzing data and drawing useful insights from it. Here we use IBM Db2 for cloud, which is fully managed SQL database provided as a service.

We performed SQL Queries to gather information from the given dataset.

1. Display the names of the unique launch sites in the space mission
2. Display 5 records where launch sites begin with the string 'CCA'
3. Display the total payload mass carried by boosters launched by NASA (CRS)
4. Display average payload mass carried by booster version F9 v1.1
5. List the date when the first successful landing outcome in ground pad was achieved.
6. List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
7. List the total number of successful and failure mission outcomes
8. List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
9. List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
10. 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 Collection

Build an Interactive Map with Folium

Folium makes it easy to visualize the data that's been manipulated in Python on an interactive leaflet map. We use latitude and longitude coordinates for each launch site and added a circle marker around each launch site with a label of the name of the each site. It is also easy to visualize the number of success and failure for each launch site with Green and Red markers on the map.

Map Objects	Code	Results
Map marker	folium.Marker(	**Map object to make a mark on the map.**
Icon marker	folium.Icon(	**Create an icon on map.**
Circle marker	folium.Circle(	**Create a circle where marker is being placed.**
Polyline	folium.PolyLine(	**Create a line between points.**
Marker Cluster Object	MarkerCluster()	**To simplify a map which contains many markers having same coordinates.**
Antpath	Folium.plugins.Antpath(	**Create an animated line between points.**

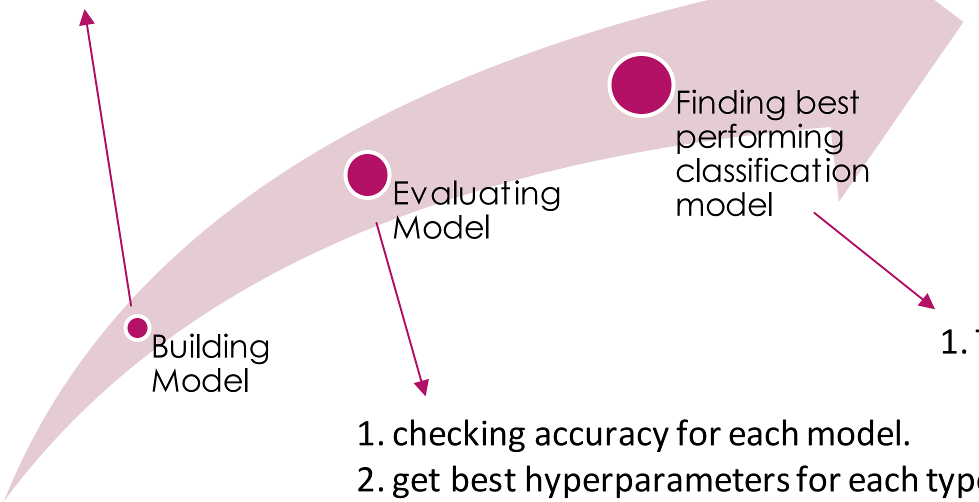
Github_Collection

Predictive Analysis (Classification)

Building Model

- Load our feature engineered data into dataframe.
- Transfer it into Numpy arrays
- Standardize and Transform data
- Split data into training and testing data sets
- Check how many test samples has been created
- List down ML algorithms we want to use.
- Set our parameters and algorithms to GridSearchCV
- Fit our datasets into the GridSearchCV objects and train our model.

```
y = data['Class'].to_numpy()
transform = preprocessing.StandardScaler()
X = transform.fit(X).transform(X)
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size = 0.2, random_state = 2)
Y_test.shape
```




```
algorithms = {'KNN':knn_cv.best_score_,'Decision Tree':tree_cv.best_score_,'Logistic Regression':logreg_cv.best_score_,'SVM':svm_cv.best_score_}
best_algorithm = max(algorithms, key= lambda x: algorithms[x])
```

1. The model with best accuracy score wins the best performing model.

1. checking accuracy for each model.
2. get best hyperparameters for each type of algorithms.
3. plot confusion matrix.

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

RESULTS

EXPLORATORY DATA ANALYSIS RESULTS

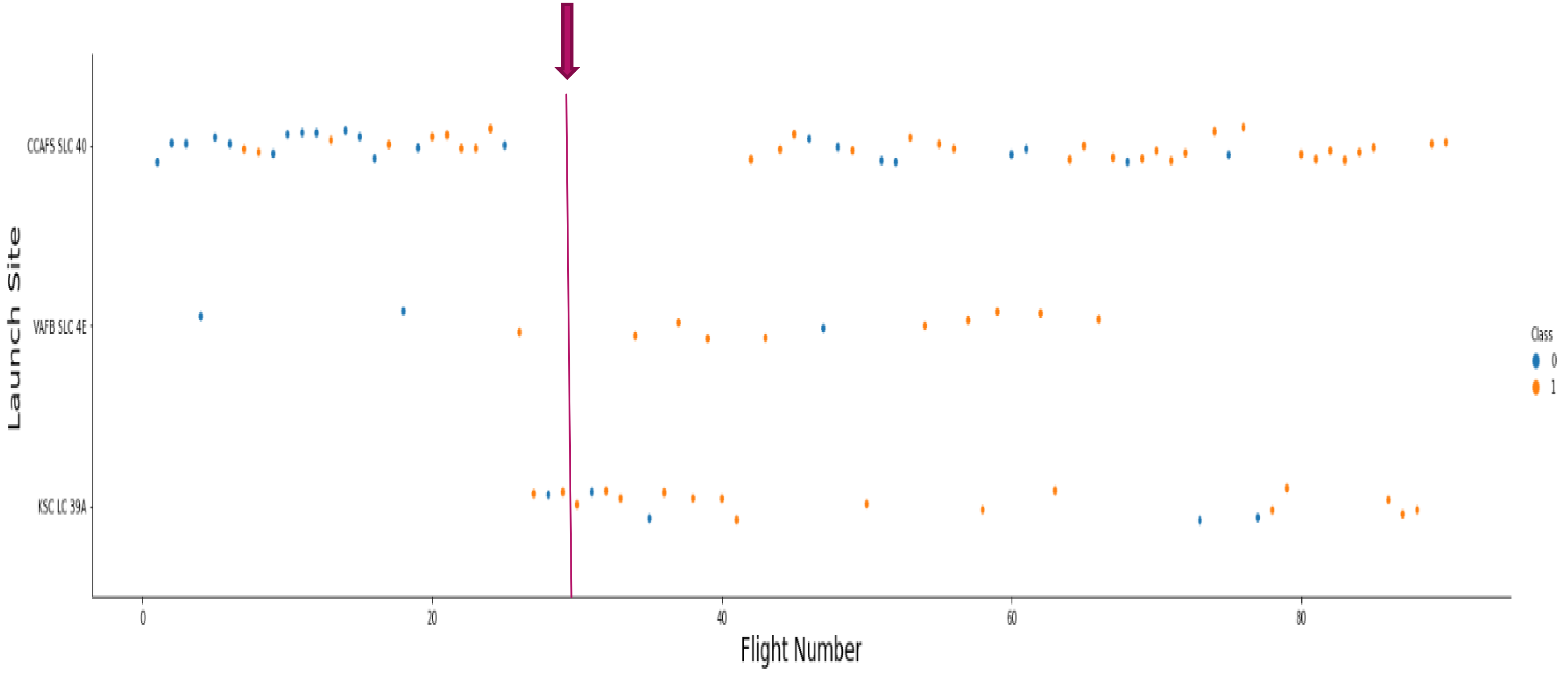
INTERACTIVE ANALYTICS DEMO IN SCREENSHOTS

PREDICTIVE ANALYSIS RESULTS

EDA with VISUALIZATION

Flight Number vs. Launch Site

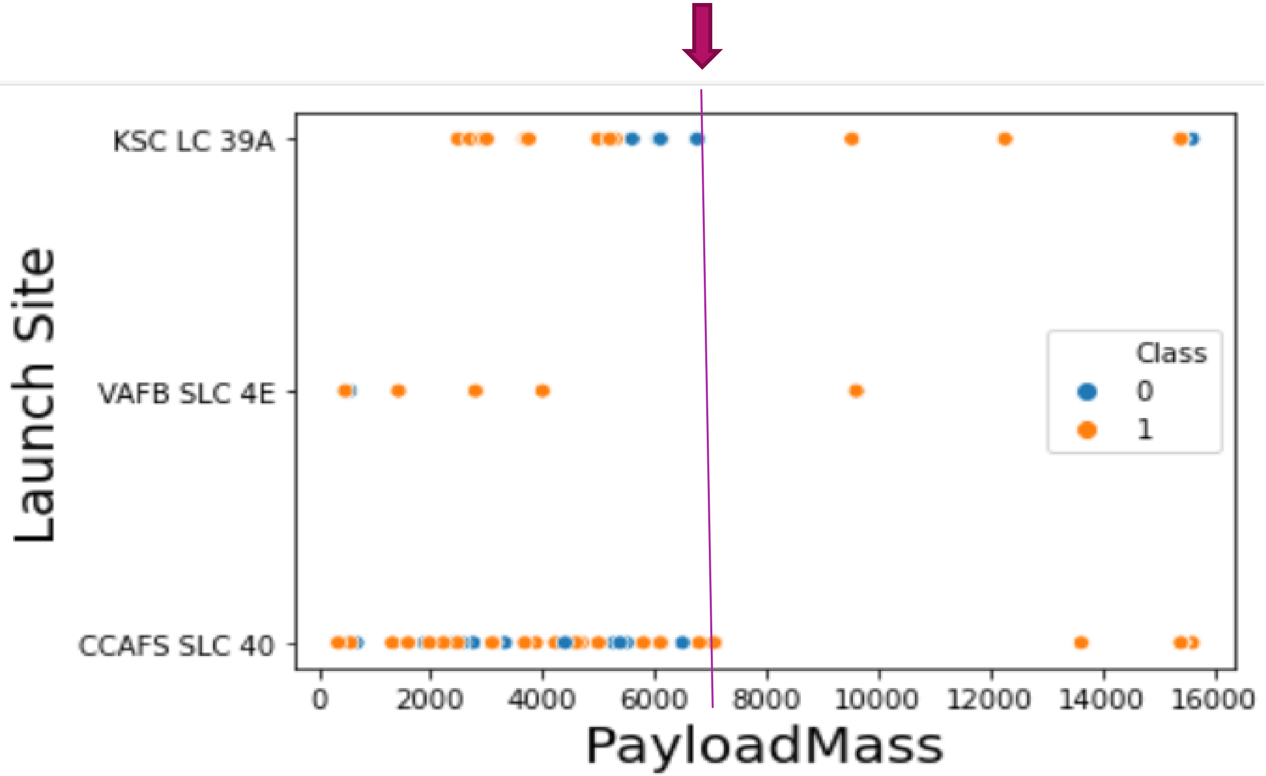
- With higher flight numbers (greater than 30) the success rate for the rocket is increasing.



Github_Collection

Payload vs. Launch Site

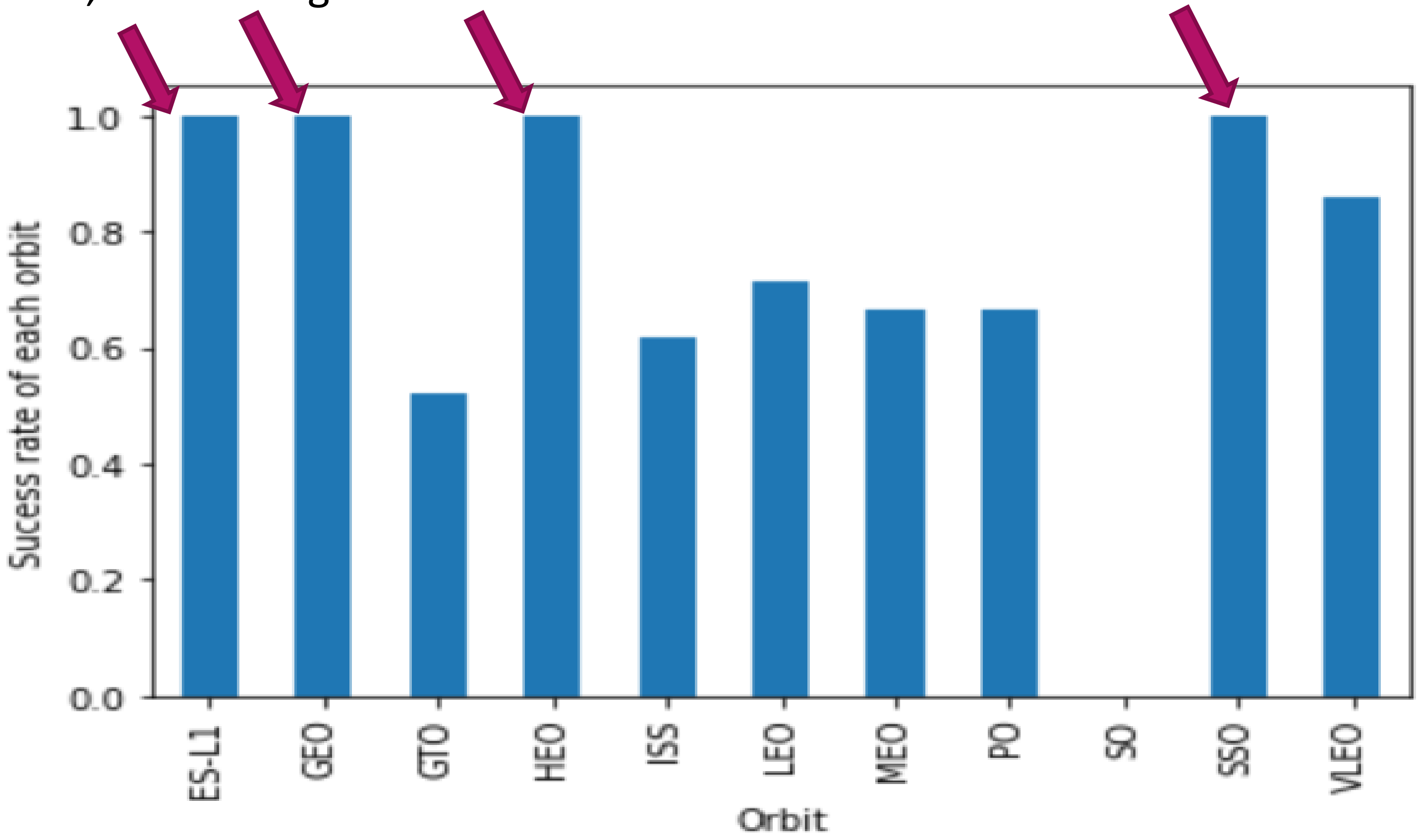
- The greater the payload mass (greater than 7000 kg) higher the success rate for the rocket. But there is no clear pattern to take the decision, if the launch site is dependent on PayLoad Mass for a success launch.



Github_Collection

Success Rate vs. Orbit Type

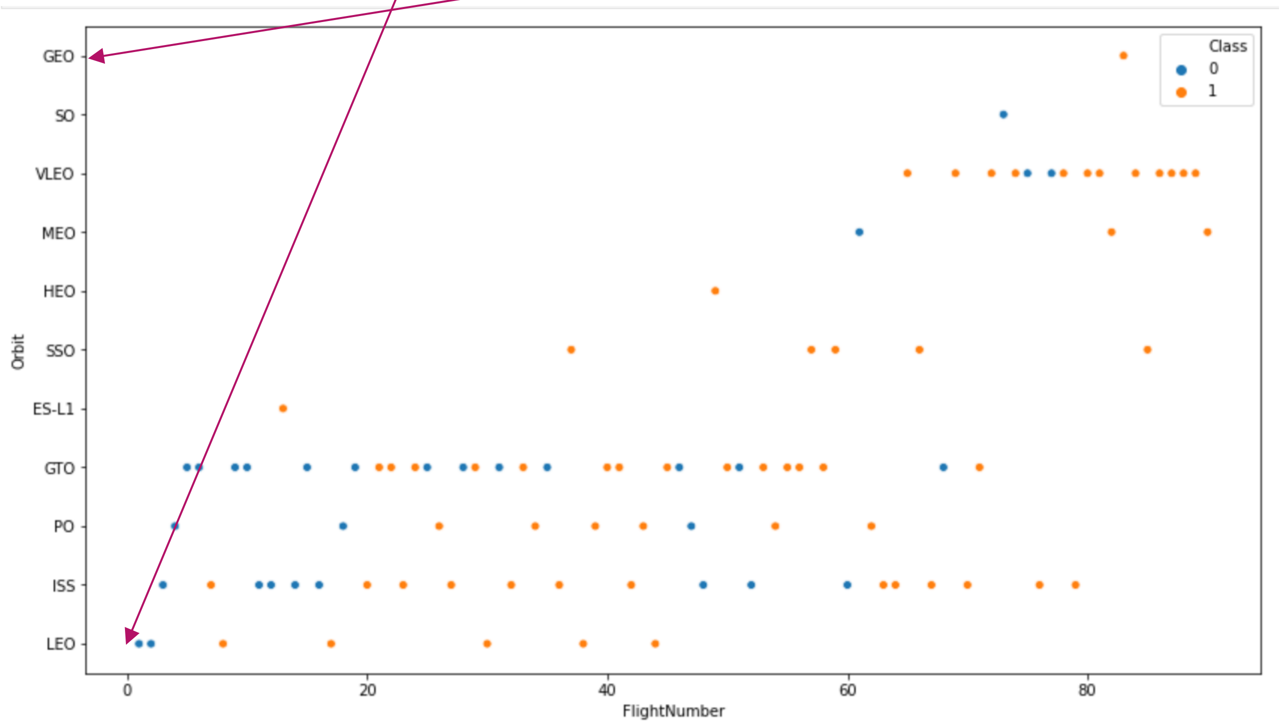
- ES-L1, GEO, HEO, SSO has highest success rate.



Github_Collection

Flight Number vs. Orbit Type

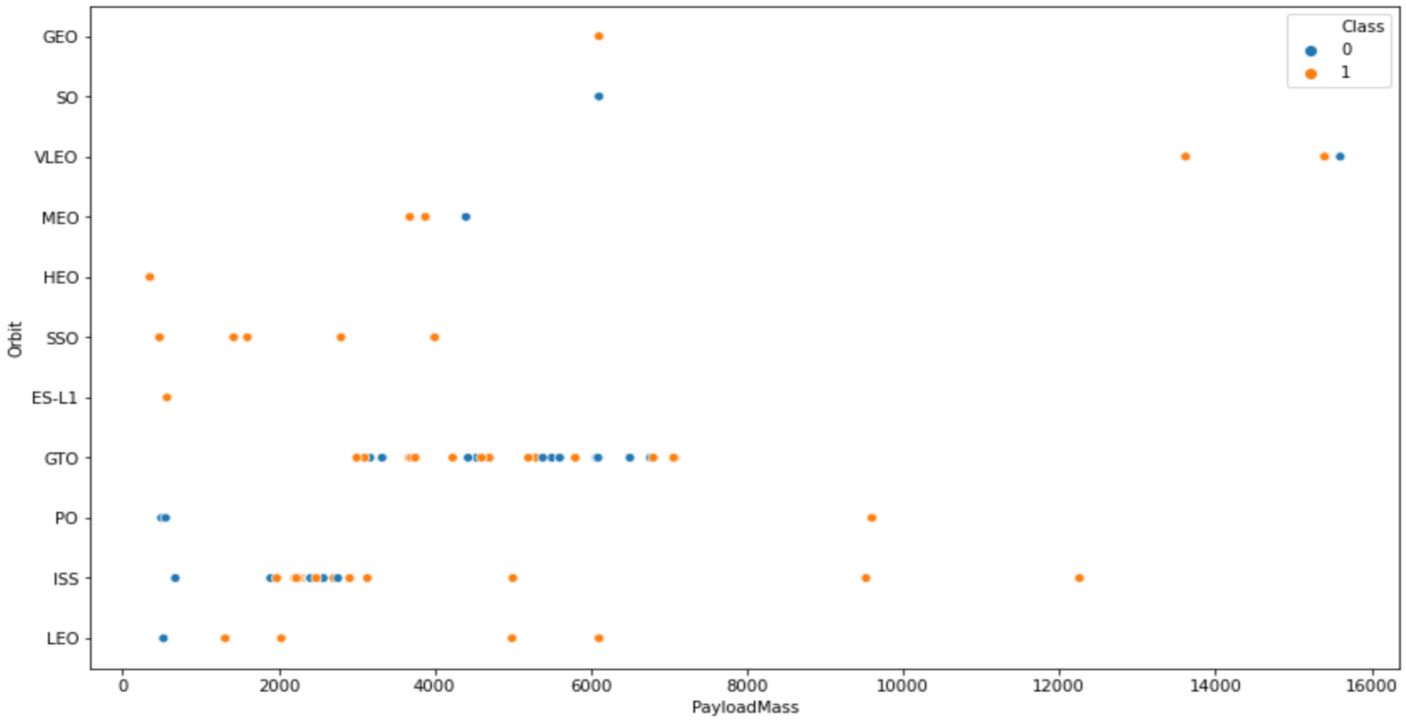
- We see that for LEO orbit the success increases with number of flights.
- On the other hand, there seems to be no relationship between flight number and the GTO orbit.



Github Collection

Payload vs. Orbit Type

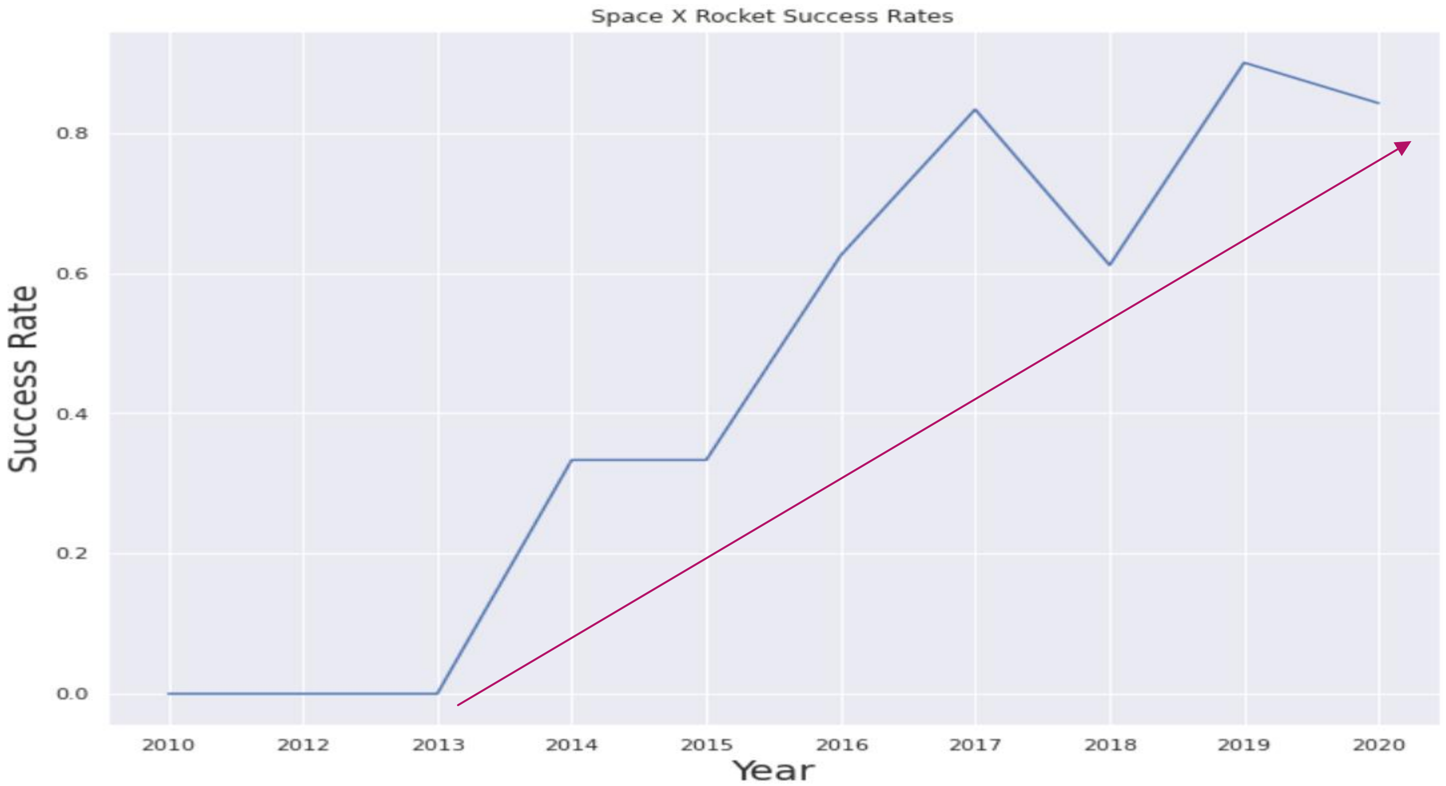
- We observe that heavy payloads have a negative influence on MEO, GTO, VLEO orbits.
- Positive on LEO, ISS orbits.



Github_Collection

Launch Success Yearly Trend

- We can observe that the success rate since 2013 kept increasing relatively though there is slight dip after 2019.



Github_Collection

EDA with SQL

ALL LAUNCH SITE NAMES

SQL Query

Display the names of the unique launch sites in the space mission

```
%sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;
```

```
 * ibm_db_sa://qmk00434:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

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

Using the word DISTINCT in the query we pull unique values for Launch site column from the table SPACEXTBL.

Github_Collection

Launch Site Names Begin with 'CCA'

SQL Query

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

```
%sql SELECT LAUNCH_SITE FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5
```

```
 * ibm_db_sa://qmk00434:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

launch_site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Github_Collection

- We can fetch from the table SPACEXTBL using keyword 'LIMIT 5' and with condition 'LIKE' keyword with wild card 'CCA%'.
- The percentage in the end suggests that the launch_site name must with CCA.

Total Payload Mass

SQL Query

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

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS "Total Payload Mass by NASA (CRS)" FROM SPACEXTBL WHERE CUSTOMER = 'NASA (CRS)';
```

```
 * ibm_db_sa://qmk00434:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

Total Payload Mass by NASA (CRS)
45596

- The function SUM calculates the total in the column PAYLOAD_MASS_KG_ and WHERE clause filters the data to fetch Customer's by name "NASA(CRS)".

Github_Collection

Average Payload Mass by F9 v1.1

SQL Query

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

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS "AVERAGE PAYLOAD MASS BY BOOSTER F9 v1.1" FROM SPACEXTBL \
WHERE BOOSTER_VERSION = 'F9 v1.1';
```

```
 * ibm_db_sa://qmk00434:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

AVERAGE PAYLOAD MASS BY BOOSTER F9 v1.1
2928

- The function AVG works out the average in the column "PALOAD_MASS_KG_" and the WHERE clause filters the dataset to only perform calculations on Booster_Version "F9 v1.1".

Github Collection

First Successful Ground Landing Date

SQL Query

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

Hint:Use min function

```
%sql SELECT MIN(DATE) AS "First Successful Landing Outcome in Ground Pad" FROM SPACEXTBL \
WHERE LANDING__OUTCOME = 'Success (ground pad)';
```

```
 * ibm_db_sa://qmk00434:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

First Successful Landing Outcome in Ground Pad
2015-12-22

Github Collection

- The function MIN works out the minimum date in the column "DATE" and the WHERE clause filters the data to only perform calculations on Landing_Outcome with values "Success (ground Pad)".

Successful Drone Ship Landing with Payload between 4000 and 6000

SQL Query

```
%sql SELECT * FROM SPACEXTBL
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL \
WHERE PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000 AND landing__outcome = 'Success (drone ship)';
```

```
 * ibm_db_sa://qmk00434:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
 * ibm_db_sa://qmk00434:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

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

Github_Collection

- Selecting Booster Version column and WHERE clause filters the dataset to Landing_Outcome = "Success (ground Pad)".
- 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

```
%sql SELECT COUNT(MISSION_OUTCOME) AS "Total_Number_of_Success_Mission_Outcomes" FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE 'Success%';
```

```
 * ibm_db_sa://qmk00434:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

Total_Number_of_Success_Mission_Outcomes
100

```
%sql SELECT COUNT(MISSION_OUTCOME) AS "Total_Number_of_FAILURE_Mission_Outcomes" FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE 'Failure%';
```

```
 * ibm_db_sa://qmk00434:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

Total_Number_of_FAILURE_Mission_Outcomes
1

Github_Collection

- Selecting multiple count is a complex query. Counting the mission outcome with WHERE clause using keyword LIKE.
- The total number of success mission outcomes are 100 and failure mission outcomes are 1.

Boosters Carried Maximum Payload

SQL Query

```
%sql SELECT BOOSTER_VERSION AS "Booster_versions which have carried the maximum payload mass" FROM SPACEXTBL \
WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

```
 * ibm_db_sa://qmk00434:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

Booster_versions which have carried the maximum payload mass
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

- The function MAX works out the maximum payload in the column PAYLOAD_MASS_KG_ in sub query.
- WHERE clause filters Booster Version which had that Maximum Payload.

Github_Collection

2015 Launch Records

SQL Query

```
%sql SELECT BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL  \
WHERE LANDING__OUTCOME LIKE 'Failure (drone ship)' AND DATE LIKE '2015%';
```

```
 * ibm_db_sa://qmk00434:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

booster_version	launch_site
F9 v1.1 B1012	CCAFS LC-40
F9 v1.1 B1015	CCAFS LC-40

- We need to list the records which will display the failure landing outcomes in drone ship, booster versions, launch site for the months in year 2015.

Github_Collection

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

SQL Query

```
%sql SELECT LANDING__OUTCOME as "Landing Outcome", COUNT(LANDING__OUTCOME) AS "Total Count" FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-0
```

```
 * ibm_db_sa://qmk00434:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

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

- Selecting only Landing_Outcome WHERE clause filters the data with DATE BETWEEN '2010-06-04' AND '2017-03-20'.

Github_Collection

Classification Accuracy

As you can see our accuracy is very close, but we do have a clear winner which performs best - "Decision Tree" with a score of 0.90178.

Algorithm	Accuracy	Accuracy on Test Data	Tuned Hyperparameters
Logistic Regression	0.846429	0.833334	{'C':0.01,'penalty':'l2, 'solver':'ibfgs'}
SVM	0.848214	0.833334	{'c':1.0,'gamma':0.03162277,'kernel':'sigmoid'}
KNN	0.848214	0.833334	{'algorithm':'auto','n_neighbors':10,''p':1}
Decision Tree	0.901786	0.833334	{'criterion':'gini','max_depth':10,'max_features': 'sqrt','splitter':'best'}

Github_Collection

Confusion Matrix

Out here for all the models unfortunately, we have same confusion matrix.

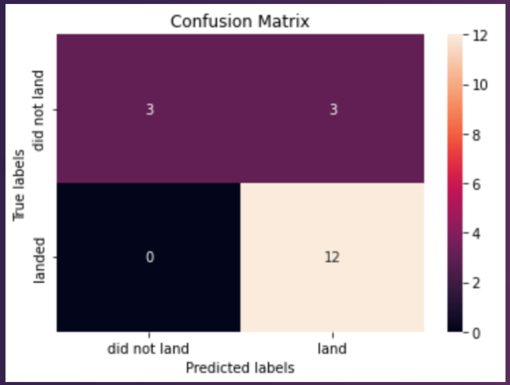
Predicted Values

		Predicted No	Predicted Yes	
Actual values	Actual No	**True Negative** **TN = 3**	**False Positive** **FP = 3**	**6**
	Actual Yes	**False Negative** **FN = 0**	**True Positive** **TP = 12**	**12**
		3	**15**	Total Cases = 18

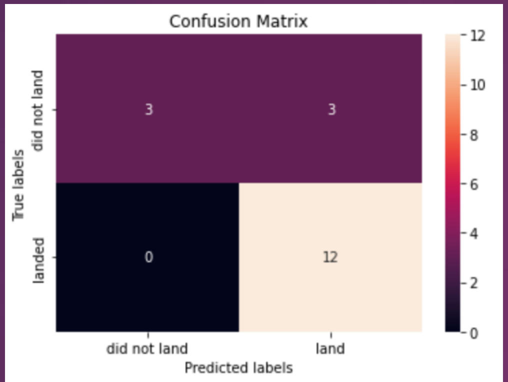
Accuracy : (TP + TN)/Total = (12 + 3)/18 = 0.83333
Misclassification Rate : (FP + FN)/Total = (3 + 0)/18 = 0.1667
True Positive Rate : TP/Actual Yes = 12/12 = 1
False Positive Rate : FP/Actual No = 3/6 = 0.5
True Negative Rate : TN/Actual No = 3/6 = 0.5
Precision : TP/Predicted Yes = 12/15 = 0.8
Prevalence = Actual Yes/Total = 12/18 = 0.667

Confusion Matrix

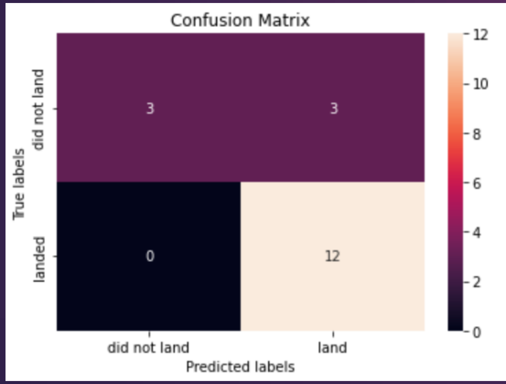
Logistic Regression



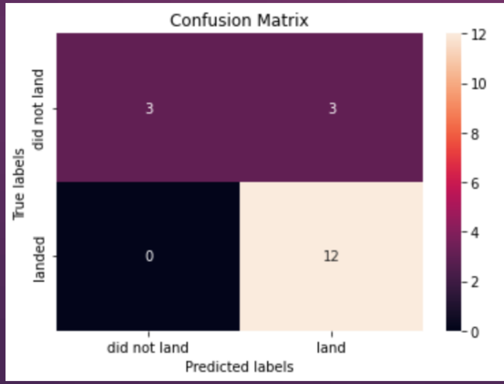
SVM



Decision Tree



KNN



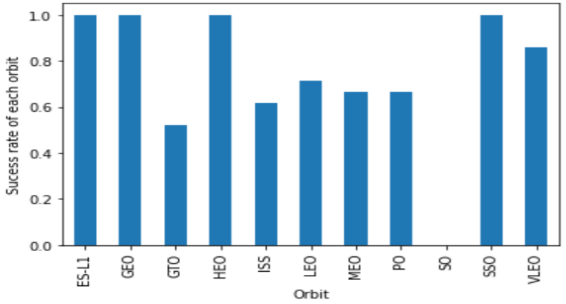
Conclusions

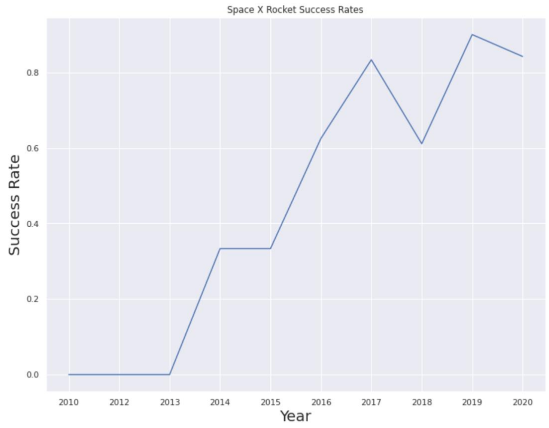
1 Orbits ES-L1, GEO, HEO, SSO has highest success rates.

2 Success rates for SpaceX launches has been increasing relatively with time and it looks like soon they will reach the required target.

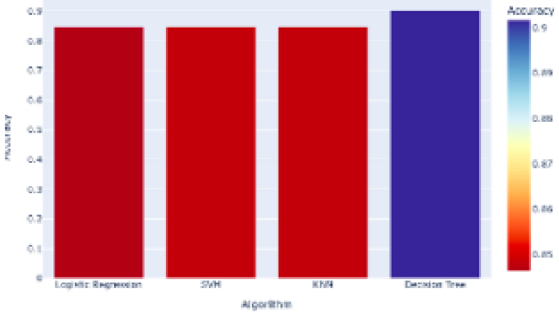
3 KSC LC-39A had the most successful launches but increasing payload mass seems to have negative impact on success.

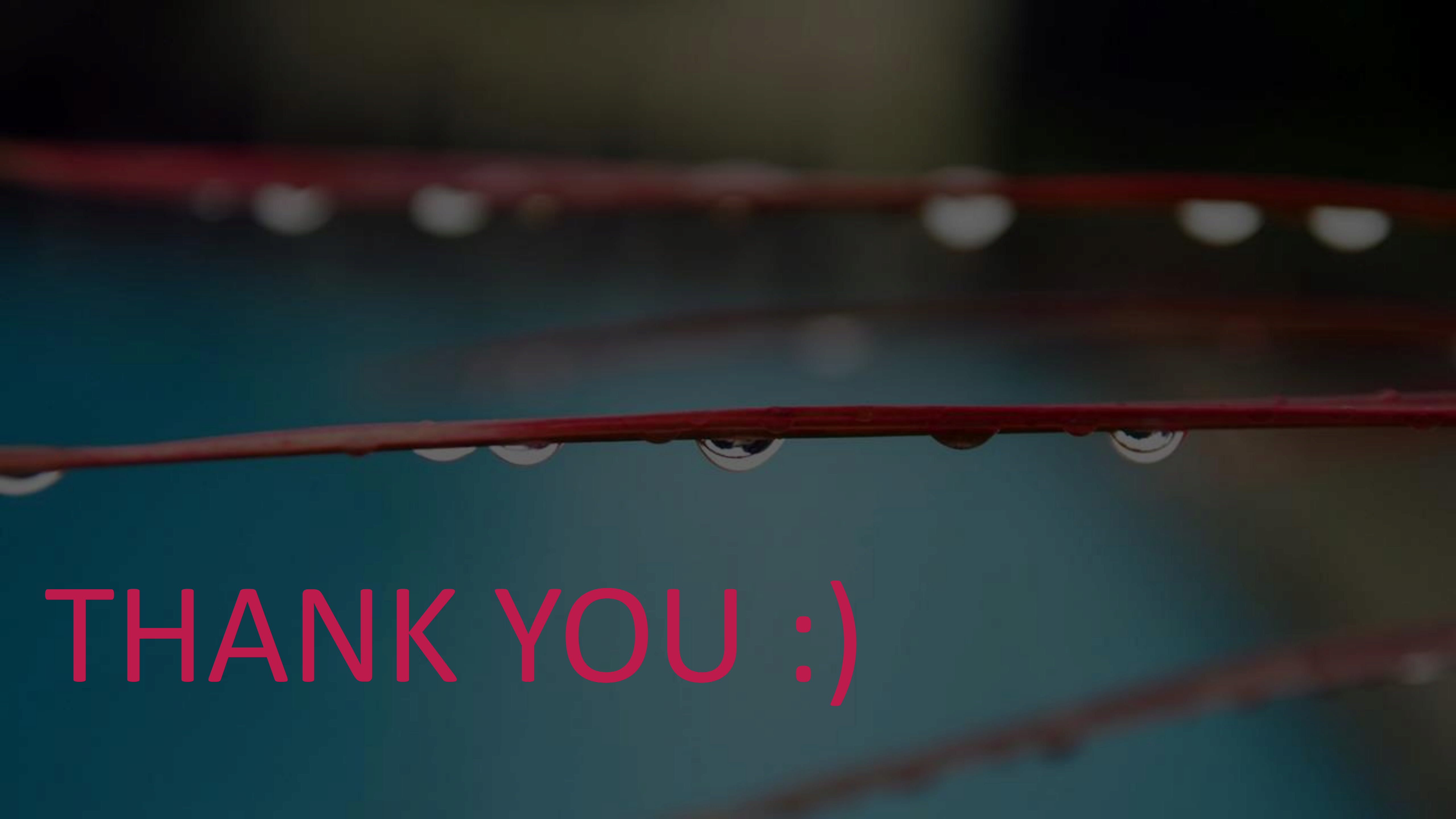
4 Decision Tree classifier algorithm is the best for ML model for provided dataset.





	Accuracy
Logistic Regression	0.846429
SVM	0.848214
KNN	0.848214
Decision Tree	0.900000





THANK YOU :)