Winning Space Race with Data Science

Chew Kar Heng
16/06/2022

Outline

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

Executive Summary

Summary of methodologies:

- In this capstone project, the methodologies will used are Data Collection, Data Wrangling, Data Analysis, Visual Analytic and Predictive Analysis to explore the data from SpaceX and determine the best prediction method to achieve a higher accuracy of the first stage will land and the cost of a launch.

Summary of all results :

- In this capstone project, the result for **Data Collection** is to show that the data is collected as per given instruction and collected data is in the correct format.
- The result in **Data Wrangling**, should showed the Exploratory Data Analysis (EDA) is performed and from the patterns of the data and he data to determine what would be the label for training supervised models.
- The result in **Data analysis**, should showed that the data is followed as per instruction where the csv is upload to the IBM server and using SQL to query and explore the data.
- The result in **Visual Analytic,** should showed geographical patterns about launch sites.
- The result in **Data Prediction,** will allow the us to know which is the most suitable hyperparameters and model to be used on the SpaceX data.

Introduction

In this Capstone Project, we will predict if the Falcon 9 first stage will land successfully. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.

Problems you want to find answers:

- To apply data science toolkit and machine learning in order to accurately predict the likelihood of the first stage rocket landing successfully, and thus determine the cost of a launch.
- Explore the data in order to obtain more insight from the data.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - The data was collected using SpaceX API URL and Web Scraping from the Wikipedia Website.
- Perform data wrangling
  - The data has gone through the process of Data Acquisition, Joining Data and Data Cleaning.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - The data in import from the given database. Once the data is retrieved, we need to standardize the data and assign the data into training and test data sets. The dataset will undergo few models to determine the most suitable model

Data Collection

- The data sets were collected as below:

1. The data is collected by using the get request call the data in SpaceX API.
2. Then decode the response content as a Json using .json() and turn it into a Pandas dataframe using .json_normalize()
3. We then cleaned the data, that has no information, for example the rocket column has no information about the rocket.
4. Next, we checked for missing values and fill in missing values and replace with the mean value.
5. In addition, we performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.
6. The objective was to extract the launch records as HTML table, parse the table and convert it to a pandas data frame for future analysis.

Data Collection – SpaceX API

- We used the get request to the SpaceX API to collect data, clean the requested data and did some basic data wrangling and formatting.
- The link to the notebook is https://github.com/ACJB/Data-Science-Capstone-Project/blob/907f208add310eba57bbeadc3cedc7ce8ba0f841/jupyter-labs-spacex-data-collection-api.ipynb

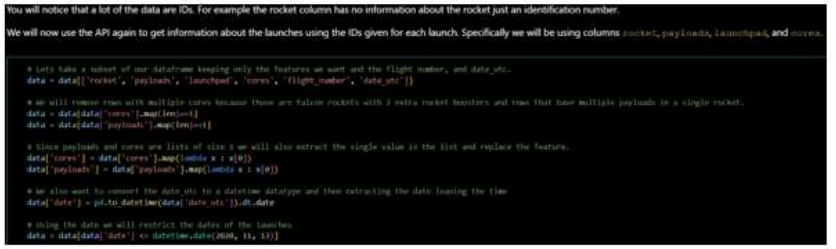
Step 1: Using "get()" to retrieved data from SpaceX API

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

Step 2: Using Json_normalize to covert the result into dataframe

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

Step 3: Cleaned the data, that has no information, for example the rocket column has no information about the rocket.

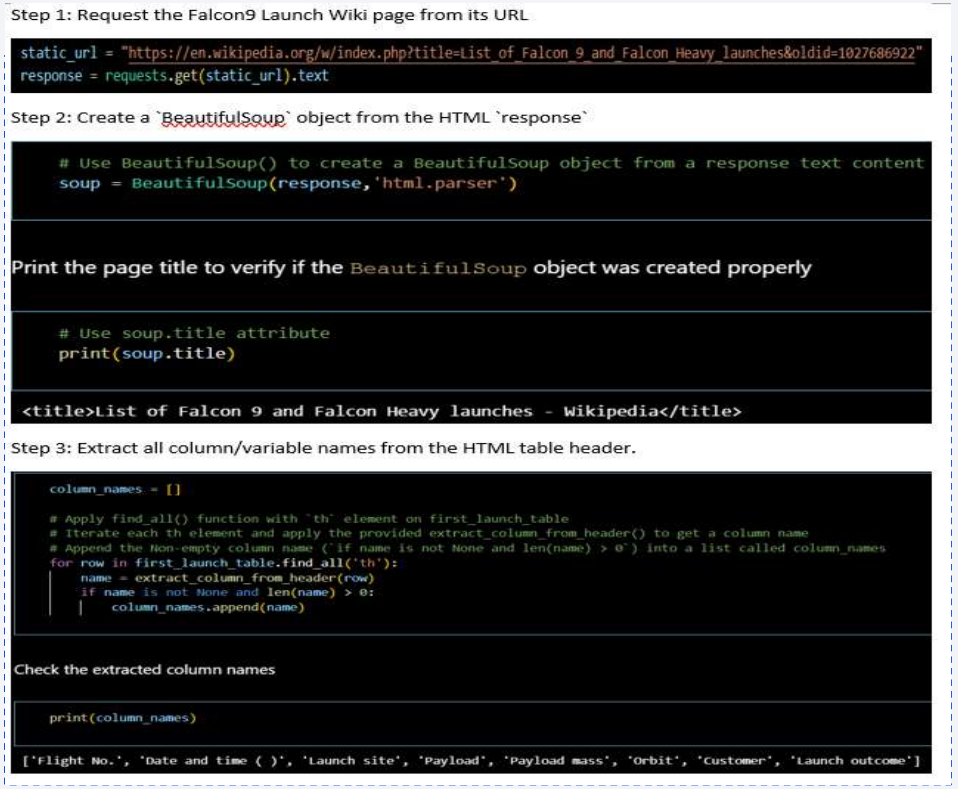


Step 4: Dealing with Missing Values

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

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

Data Collection - Scraping

- We applied BeautifulSoup to webscrap Falcon 9 launch records.
- The link to the notebook is https://github.com/ACJB/Data-Science-Capstone-Project/blob/907f208add310eba57bbeadc3cedc7ce8ba0f841/jupyter-labs-webscraping.ipynb



Data Wrangling

- In Data Wrangling, we will perform exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the results to csv.
- The link to the notebook is https://github.com/ACJB/Data-Science-Capstone-Project/blob/907f208add310eba57bbeadc3cedc7ce8ba0f841/labs-jupyter-spacex-Data%20wrangling.ipynb

Step 1: Load Space X dataset, from last section.

```
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)
```

Step 2: Calculate the number of launches on each site

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

✓ 0.9s

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

Step 3: Calculate the number and occurrence of each orbit

```
# Apply value_counts on Orbit column
df.Orbit.value_counts()
```

✓ 0.9s

```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

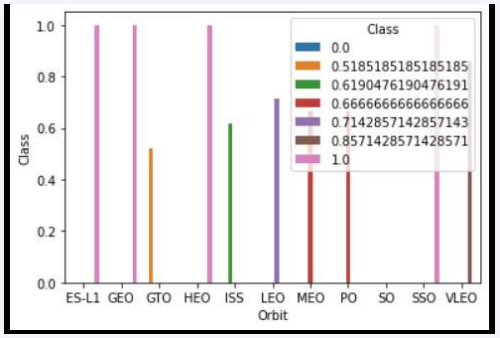
```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

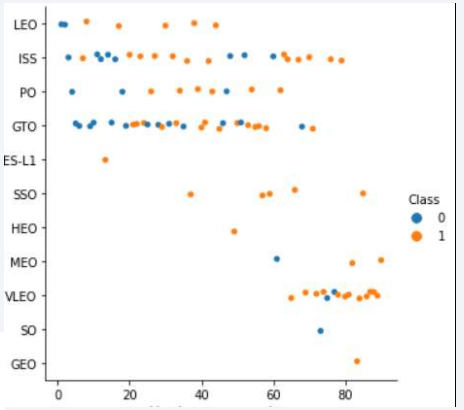
Step 4: Create a landing outcome label from Outcome column

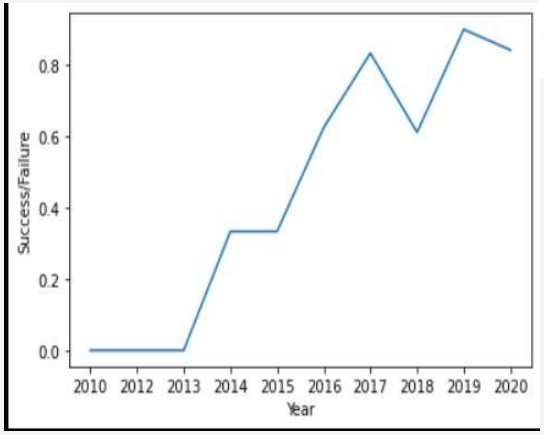
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = df['Outcome'].apply(lambda x: 0 if x == bad_outcomes else 1)

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

df.head(5)
```

EDA with Data Visualization

- In this Capstone project, we have explored Scatter plot, Bar Chart and Line Chart.
- Scatter plot feature displaying values for typically two values for a set of data, thus illustrating the degree of correlation between two variables, in this project **Scatter plot** is used to visualize the relation between payload, flight number and launch site.
- Line Chart are used to track changes over short and long periods of time and in the capstone **line chart** is used to visualize the launch success yearly trend.
- Bar Chart are used to compare things between different groups or to track changes over time and in the project **bar chart** is used to visualize the relationship between success rate of each orbit type
- The link to the notebook is https://github.com/ACJB/Data-Science-Capstone-Project/blob/907f208add310eba57bbeadc3cedc7ce8ba0f841/jupyter-labs-eda-dataviz.ipynb







EDA with SQL

- We loaded the SpaceX dataset into a PostgreSQL database without leaving the jupyter notebook.
- We applied EDA with SQL to get insight from the data. We wrote queries to find out for instance:
  - The names of unique launch sites in the space mission.
  - The total payload mass carried by boosters launched by NASA (CRS)
  - The average payload mass carried by booster version F9 v1.1
  - The total number of successful and failure mission outcomes
  - The failed landing outcomes in drone ship, their booster version and launch site names.
- The link to the notebook is https://github.com/ACJB/Data-Science-Capstone-Project/blob/1cc2ca5edce077c4da26ca230cba3fac4cca4a47/jupyter-labs-eda-sql-coursera.ipynb

Build an Interactive Map with Folium

- The objective is to marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- We assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- We calculated the distances between a launch site to its proximities. We answered some question for instance:
  - Are launch sites near railways, highways and coastlines.
  - Do launch sites keep certain distance away from cities.
- The link to the notebook is https://github.com/ACJB/Data-Science-Capstone-Project/blob/907f208add310eba57bbeadc3cedc7ce8ba0f841/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash
- We plotted pie charts showing the total launches by a certain sites
- We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- The link to the notebook is https://github.com/ACJB/Data-Science-Capstone-Project/blob/907f208add310eba57bbeadc3cedc7ce8ba0f841/Dashboard.py

Predictive Analysis (Classification)

- Using numpy and pandas to load the data and transformed the data, then split our data into training and testing data set.
- We built different machine learning models and use GridSearchCV to test on the different hyperparameters.
- Then we check the accuracy for each model and improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model.
- The link to the notebook is https://github.com/ACJB/Data-Science-Capstone-Project/blob/907f208add310eba57bbeadc3cedc7ce8ba0f841/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Step 1: Load the data.

```
data = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_2.csv")

# If you were unable to complete the previous lab correctly you can uncomment and load this csv

# data = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DS0701EN-SkillsNetwork/api/dataset_part_2.csv')

data.head()
```

Step 2: Create a NumPy array from the column Class in data, by applying the method to_numpy() then assign it to the variable Y.

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

Step 3: Standardize and transform he data and assign the data into training and test data sets

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

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
print('The Train set for X and Y are :', X_train.shape, Y_train.shape)
print('The Test set for X and Y are :', X_test.shape, Y_test.shape)
```

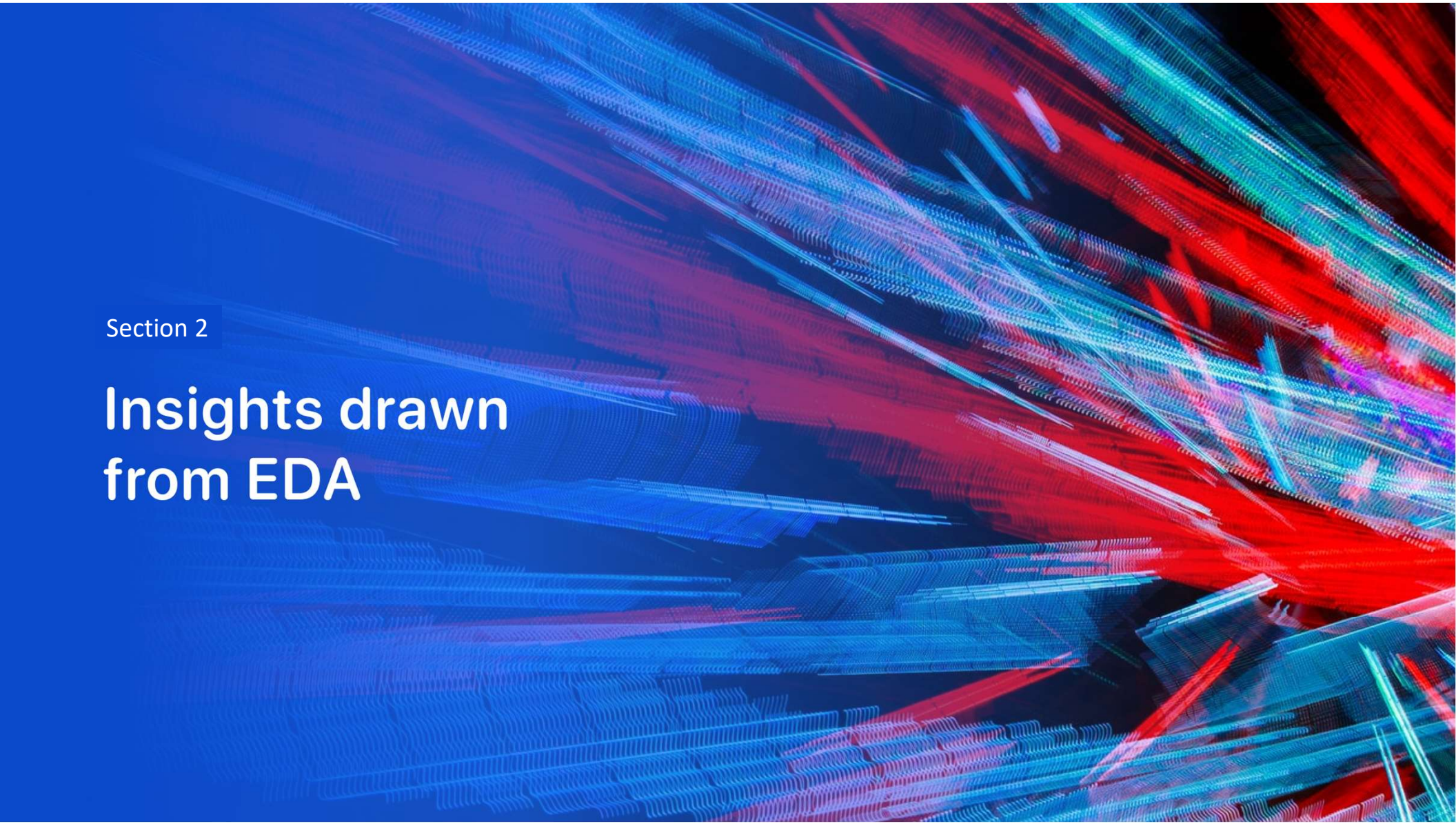
Step 4: Create a logistic regression object then create a GridSearchCV object logreg_cv with cv = 10. Fit the object to find the best parameters from the dictionary parameters

```
parameters ={'C':[0.01,0.1,1],
             'penalty':['l2'],
             'solver':['lbfgs']}

lr=LogisticRegression()
grid_search = GridSearchCV(lr,parameters, cv=10)
logreg_cv = grid_search.fit(X_train, Y_train)

parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
lr=LogisticRegression()

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

logreg_cv.score(X_test, Y_test)

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

Step 5: Then repeat step 4 with different model : SVM, Decision Tree and KNN

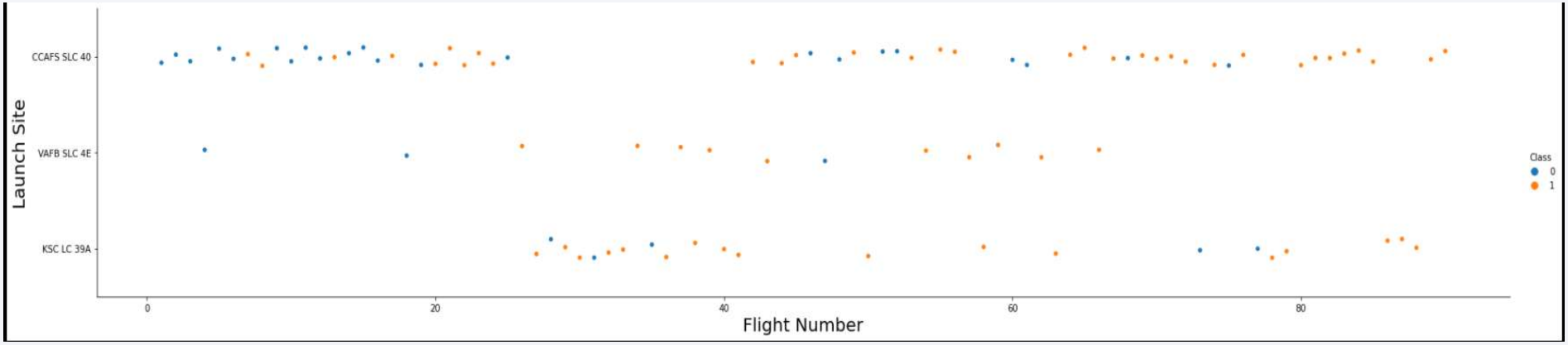
Results

- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results


Section 2
Insights drawn
from EDA

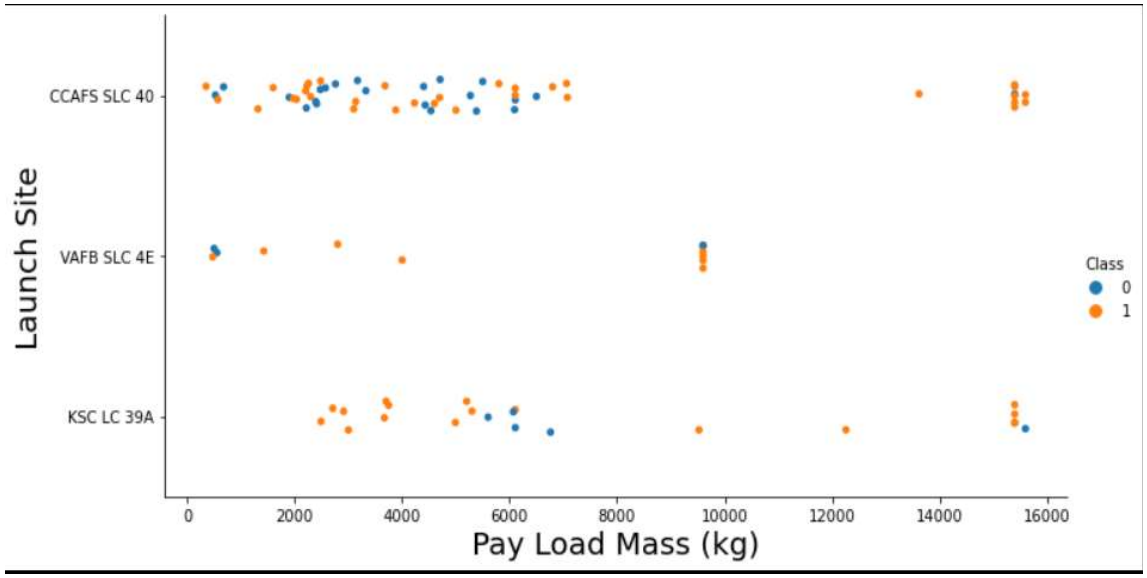
Flight Number vs. Launch Site

- Scatter plot of Flight Number vs. Launch Site



- From the plot, we can observe that the larger the flight amount at a launch site, the greater the success rate at a launch site.

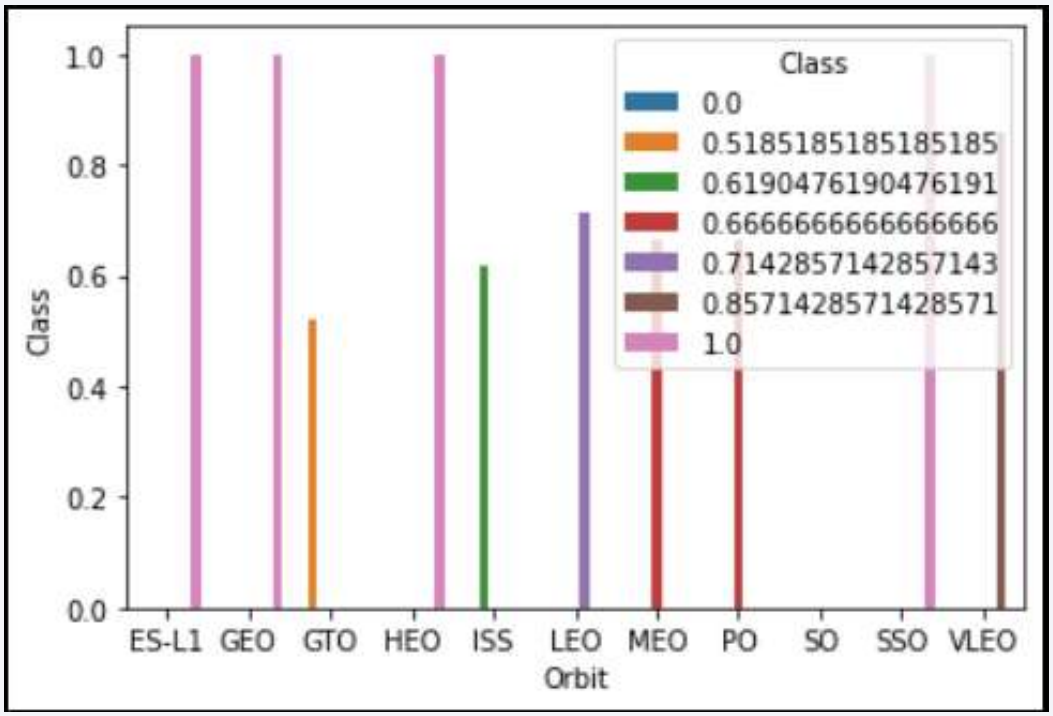
Payload vs. Launch Site



From the plot, we can observe that the greater the payload mass for launch site CCAFS SLC 40, the higher the success rate for the rocket.

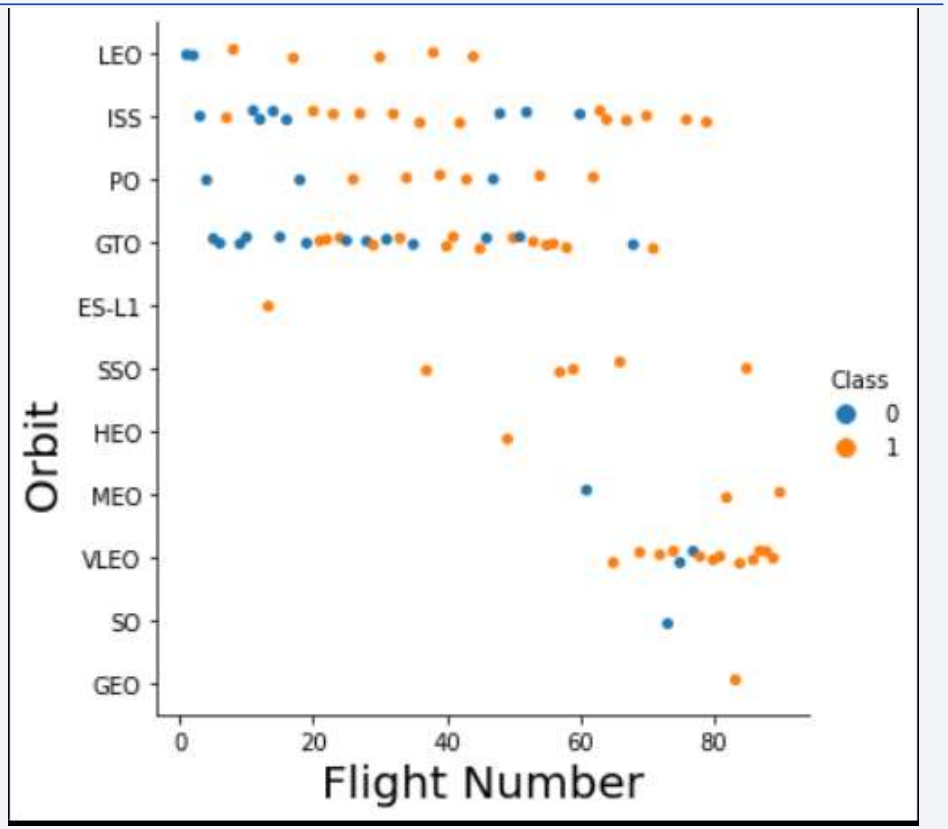
Success Rate vs. Orbit Type

- From the plot, we can observe that ES-L1, GEO, HEO, SSO, VLEO had the most success rate.



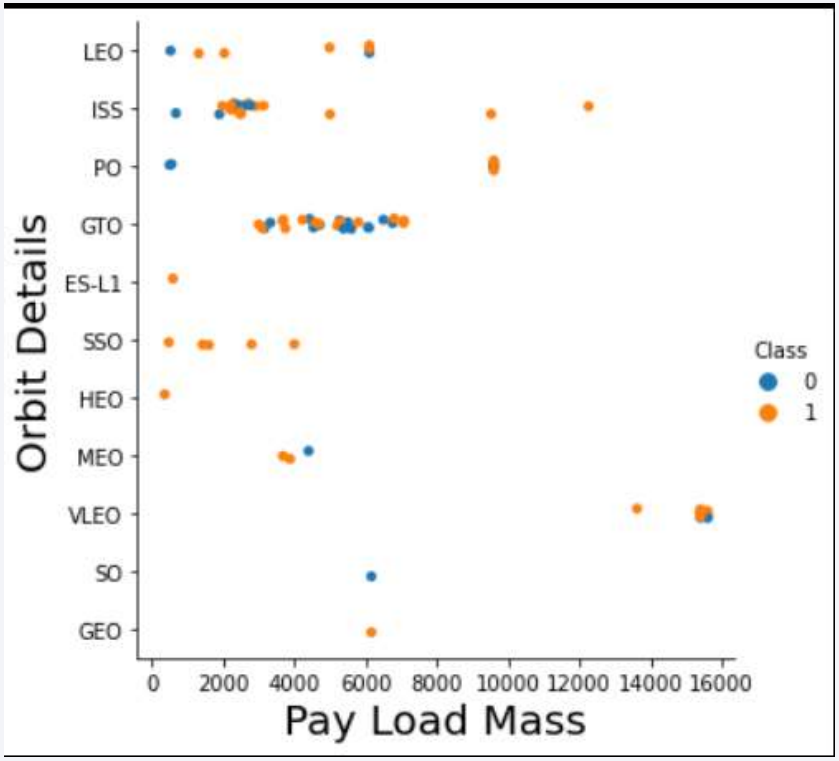
Flight Number vs. Orbit Type

- From the plot we can observe that in the LEO orbit, success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit.



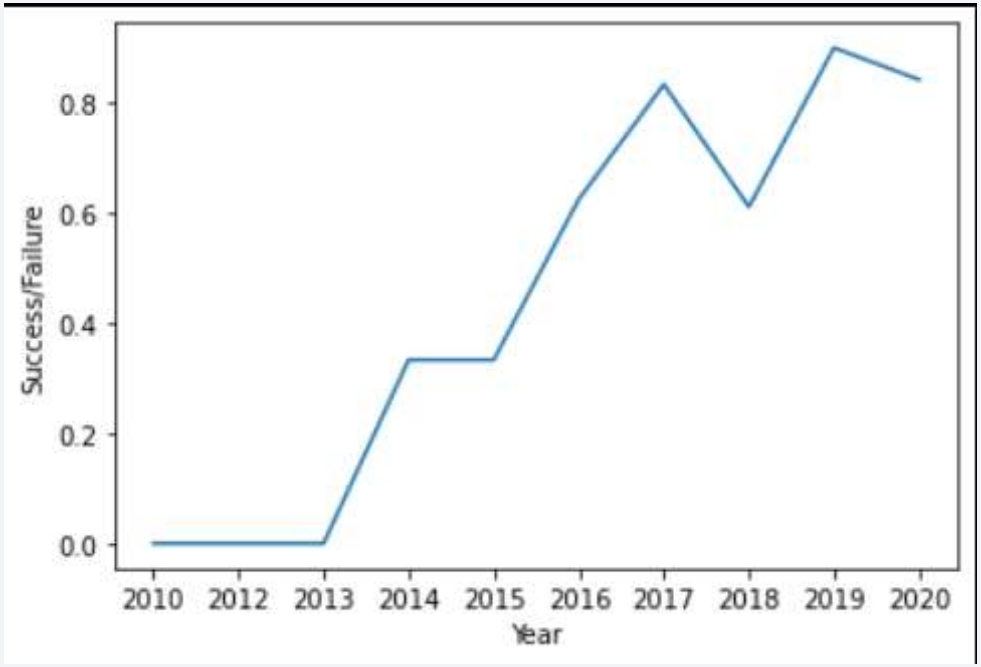
Payload vs. Orbit Type

- From the plot we can observe that with heavy payloads, the successful landing are more for PO, LEO and ISS orbits.



Launch Success Yearly Trend

- From the plot, we can observe that success rate since 2013 kept on increasing till 2017.
- The success rate has slightly drop in between 2017 to 2018 and increase in 2019.



All Launch Site Names

- By using the key word **Unique()** the program will show the unique launch site name without overlapping sites from the SpaceX data.

```
%sql select Unique(LAUNCH_SITE) from SPACEXTBL;
```

```
 * ibm_db_sa://wqr24617:***@9938aec0-8105-433e-8bf9-
Done.
```

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

Launch Site Names Begin with 'CCA'

- We used the query above to display 5 records where launch sites begin with `CCA`

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://wqr24617:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32459/bludb
Done.
```

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

Total Payload Mass

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

```
%sql SELECT sum(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL where Customer LIKE 'NASA (CRS)';
```

✓ 0.7s

```
 * ibm_db_sa://wqr24617:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32459/bludb
Done.
```

payloadmass
45596

- From the code above, we calculated the total payload mass carried by boosters from NASA (CRS) is 45596.

Average Payload Mass by F9 v1.1

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

```
%sql SELECT avg(PAYLOAD_MASS__KG_) as avgpayloadmass from SPACEXTBL WHERE BOOSTER_VERSION = 'F9 v1.1';
```

✓ 0.7s

```
* ibm_db_sa://wqr24617:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32459/bludb
Done.
```

avgpayloadmass
2928

- From the code above, we calculated average payload mass carried by booster version F9 v1.1 is 2928.

First Successful Ground Landing Date

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

Hint:Use min function

```
%sql SELECT min(DATE) from SPACEXTBL WHERE LANDING__OUTCOME like 'Success (ground pad)' ;
```

✓ 0.7s

```
 * ibm_db_sa://wqr24617:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32459/bludb
Done.
```

1
2015-12-22

- The dates of the first successful landing outcome on ground pad was 22/12/2015

Successful Drone Ship Landing with Payload between 4000 and 6000

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;
```

✓ 0.7s

```
 * ibm_db_sa://wqr24617:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32459/bludb
Done.
```

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

- We used the **WHERE** clause to filter the landing outcome which have successfully landed on drone ship and applied the **AND** condition to determine successful landing with payload mass greater than 4000 but less than 6000.

Total Number of Successful and Failure Mission Outcomes

List the total number of successful and failure mission outcomes

```
%sql SELECT count(MISSION_OUTCOME) as SuccessOutcomes from SPACEXTBL WHERE MISSION_OUTCOME like 'Success%' ;
```

✓ 0.7s

```
 * ibm_db_sa://wqr24617:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32459/bludb
Done.
```

successoutcomes
100

List the total number of successful and failure mission outcomes

```
%sql SELECT count(MISSION_OUTCOME) as FailureOutcomes from SPACEXTBL WHERE MISSION_OUTCOME like 'Failed%' ;
```

✓ 0.7s

```
 * ibm_db_sa://wqr24617:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32459/bludb
Done.
```

failureoutcomes
0

- From the code above we can see that the Success outcome rate is 100, therefore the Failure outcome rate is 0.

Boosters Carried Maximum Payload

- We determined the booster that have carried the maximum payload using a subquery in the **WHERE** clause and the **MAX()** function.

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

```
%sql select BOOSTER_VERSION as boosterversion from SPACEXTBL where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from SPACEXTBL);
```

✓ 0.7s

```
* ibm_db_sa://wqr24617:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32459/bludb
Done.
```

boosterversion
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

```
%sql select MONTH(DATE),MISSION_OUTCOME,BOOSTER_VERSION,LAUNCH_SITE from SPACEXTBL where DATE between '2015-01-01' and '2015-12-31' and LANDING__OUTCOME like 'Failure (drone ship)';
```

✓ 0.7s Python

```
 * ibm_db_sa://wqr24617:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32459/bludb
Done.
```

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

- By using the **WHERE** clause, **LIKE**, **AND**, and **BETWEEN** conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015

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

```
%sql select LANDING__OUTCOME, COUNT(LANDING__OUTCOME) from SPACEXTBL where DATE BETWEEN '2010-06-04' and '2017-03-20' GROUP BY LANDING__OUTCOME ORDER BY COUNT(LANDING__OUTCOME) DESC
```

✓ 0.7s Python

```
 * ibm_db_sa://wqr24617:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32459/bludb
Done.
```

landing__outcome	2
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

- We selected Landing outcomes and the **COUNT** of landing outcomes from the data and used the **WHERE** clause to filter for landing outcomes **BETWEEN** 2010-06-04 to 2017-03-20.
- We applied the **GROUP BY** clause to group the landing outcomes and the **ORDER BY** clause to order the grouped landing outcome in descending order.

Section 3

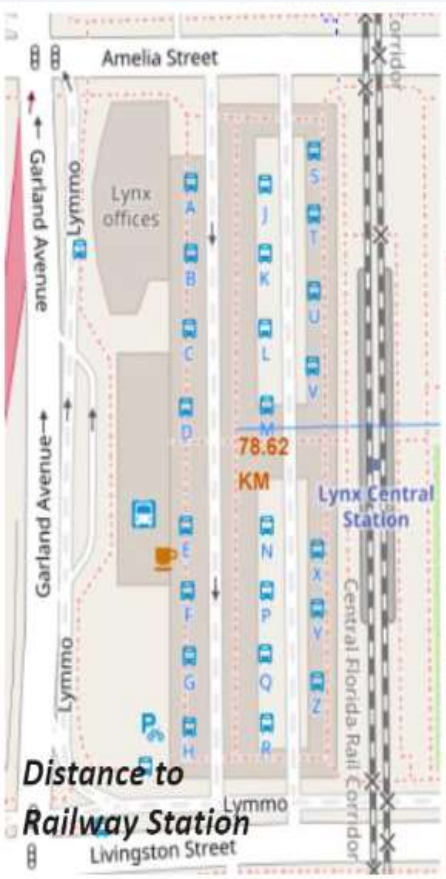
Launch Sites Proximities Analysis

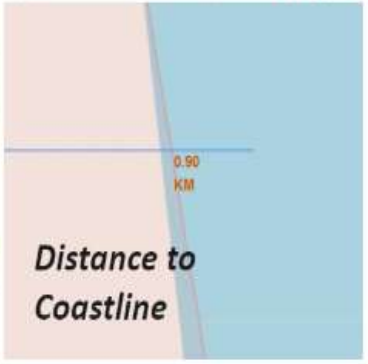
All launch sites global map markers

- Based on the Observation we can see that the launches site is located at Florida and California

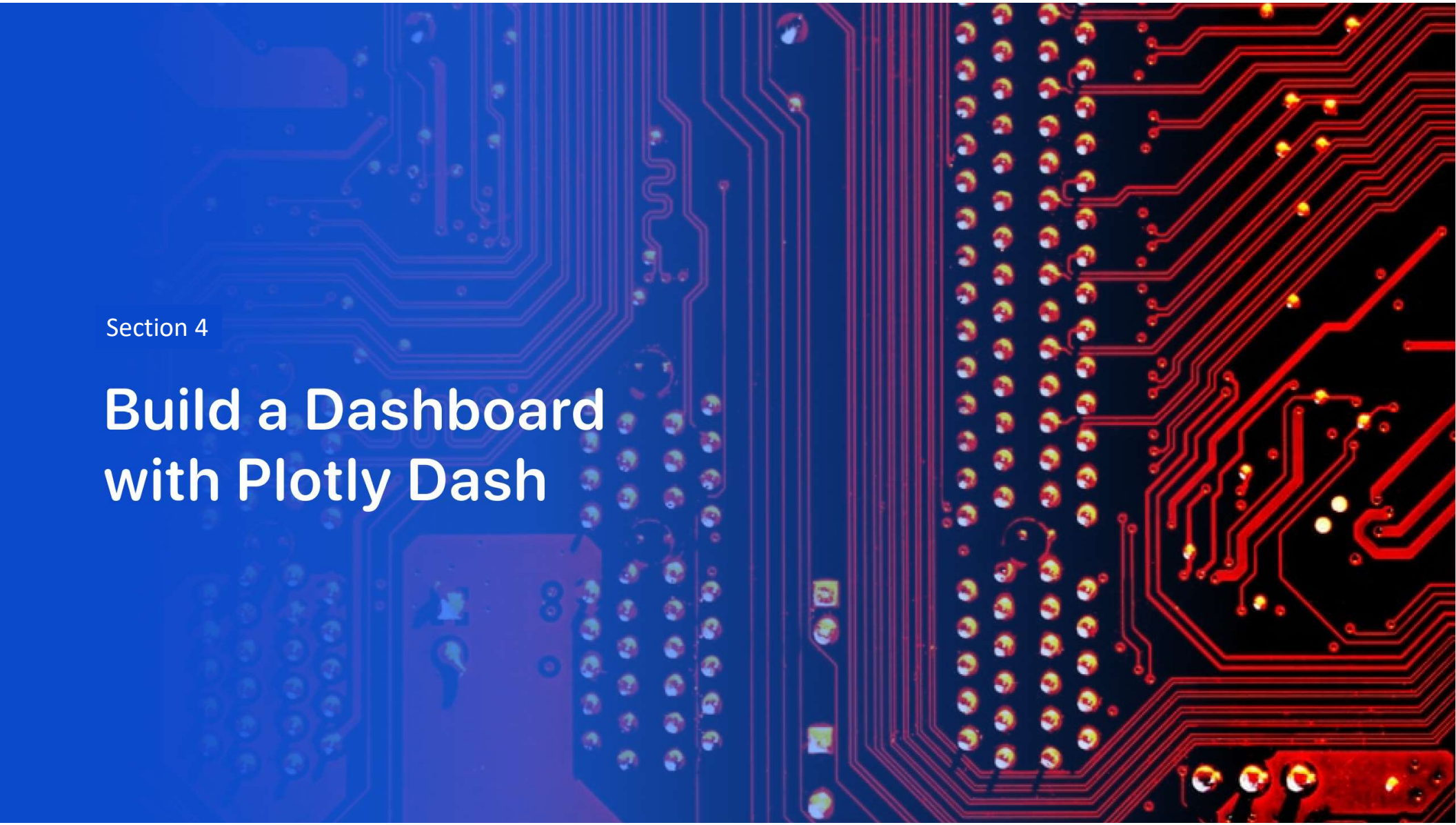
Markers showing launch sites with color labels

Launch Site distance to landmarks





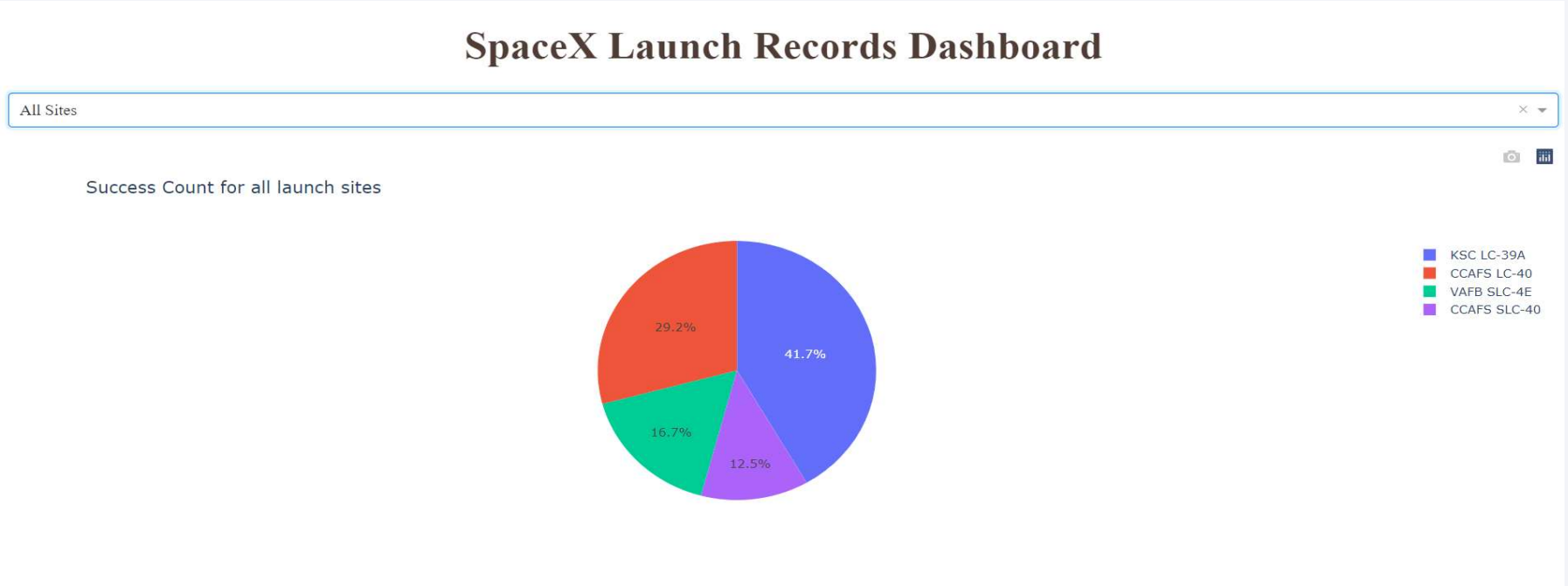
- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes



Section 4

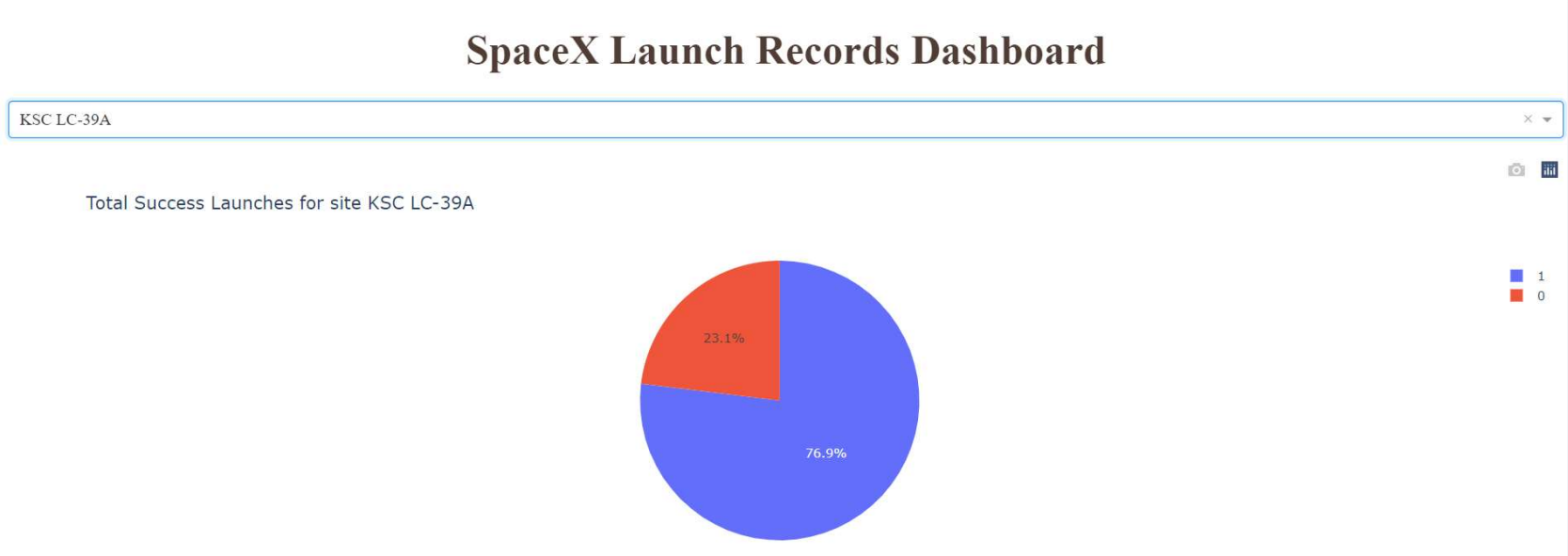
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site



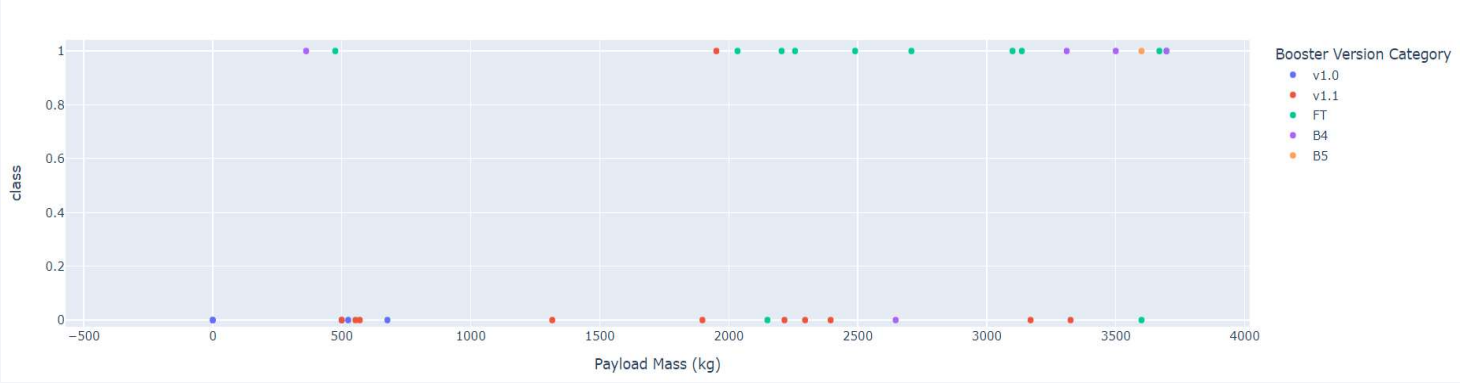
- Based on the Pie Chart we can observe that KSC LC-39A holds the highest success launch rate from all sites.

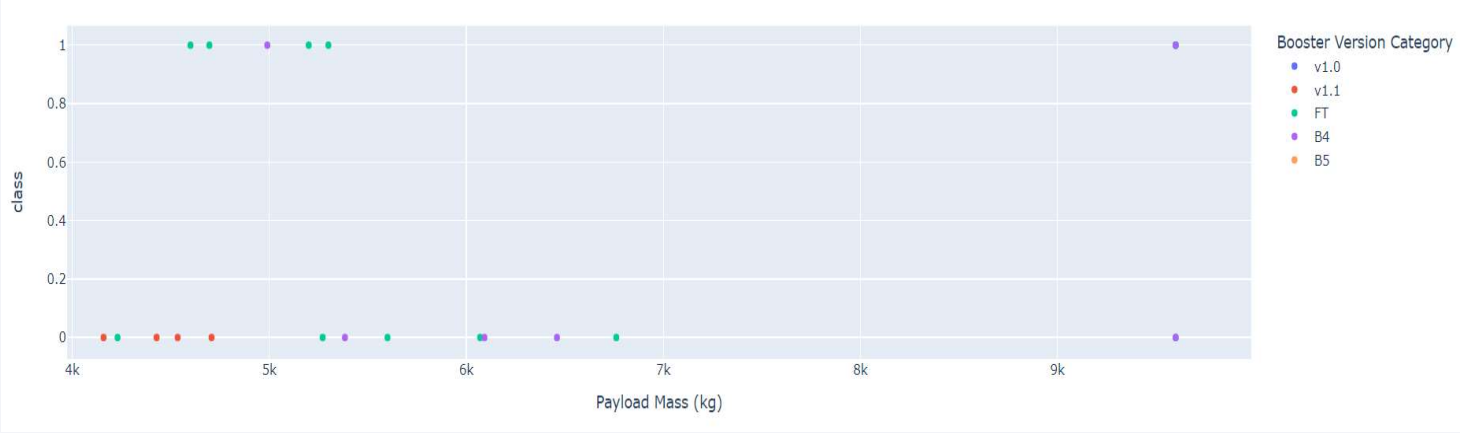
Pie chart showing the Launch site with the highest launch success ratio



Based on the Pie Chart we can observe that KSC LC-39A achieved a success rate of 76.9% with a 23.1% failure rate.

Scatter plot of Payload vs Launch Outcome for all sites, with different payload selected in the range slider





- Based on the observation we can observe that the success rate is higher in the range 0 kg to 4000 kg than the range 4000 kg to 10000 kg

Section 5

Predictive Analysis (Classification)

Classification Accuracy

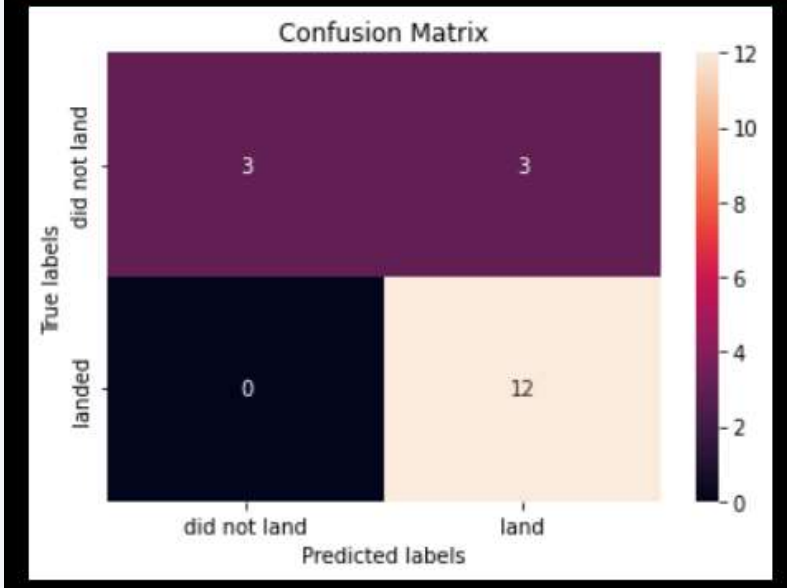
```
print('Accuracy for Logistics Regression method:', logreg_cv.score(X_test, Y_test))
print( 'Accuracy for Support Vector Machine method:', svm_cv.score(X_test, Y_test))
print('Accuracy for Decision tree method:', tree_cv.score(X_test, Y_test))
print('Accuracy for K nearsdt neighbors method:', knn_cv.score(X_test, Y_test))
```

- By showing the score of all model, we can see that the model that having the Highest accuracy of 87% is the Decision Tree Model.

Confusion Matrix

- The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The major problem is the false positives .i.e., unsuccessful landing marked as successful landing by the classifier.

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



Conclusions

- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- Launch success rate started to increase in 2013 till 2017 and drop on 2018 then increasing till 2020.
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- KSC LC-39A had the most successful launches of any sites.
- The Decision tree classifier is the best machine learning algorithm for this task.

Appendix

- The outcome of the Dashboard can be found as per link: https://github.com/ACJB/Data-Science-Capstone-Project/blob/1cc2ca5edce077c4da26ca230cba3fac4cca4a47/Dashboard.py

Thank you!