IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Pradyumn Tiwari
25-08-2022

Outline

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

Executive Summary

Summary of methodologies

- Data Collection through API
- Data Collection with Web Scraping
- Data Wrangling
- Exploratory Data Analysis with SQL
- Exploratory Data Analysis with Data Visualization
- Interactive Visual Analytics with Folium
- Machine Learning Prediction

Summary of all results

- Exploratory Data Analysis result
- Interactive analytics in screenshots
- Predictive Analytics result from Machine Learning Lab

Introduction

SpaceX is a revolutionary company who has disrupt the space industry by offering a rocket launches specifically Falcon 9 as low as 62 million dollars; while other providers cost upward of 165 million dollar each. Most of this saving thanks to SpaceX astounding idea to reuse the first stage of the launch by re-land the rocket to be used on the next mission. Repeating this process will make the price down even further. As a data scientist of a startup rivaling SpaceX, the goal of this project is to create the machine learning pipeline to predict the landing outcome of the first stage in the future. This project is crucial in identifying the right price to bid against SpaceX for a rocket launch.

The problems included:

- Identifying all factors that influence the landing outcome.
- The relationship between each variables and how it is affecting the outcome.
- The best condition needed to increase the probability of successful landing.

Methodology

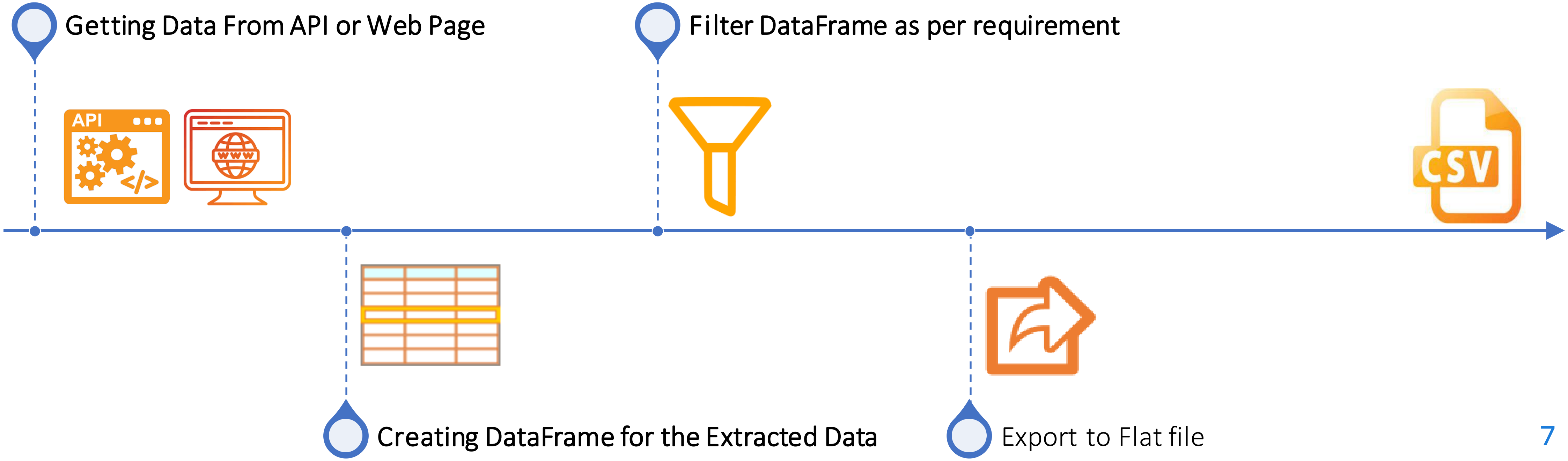
Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX REST API and Web scraping from Falcon 9 Wikipedia page.
- Perform data wrangling
  - Data was processed using one-hot encoding for categorical features.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Classification models were tuned using Cross-validation and evaluated through Confusion Matrix.

Data Collection

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



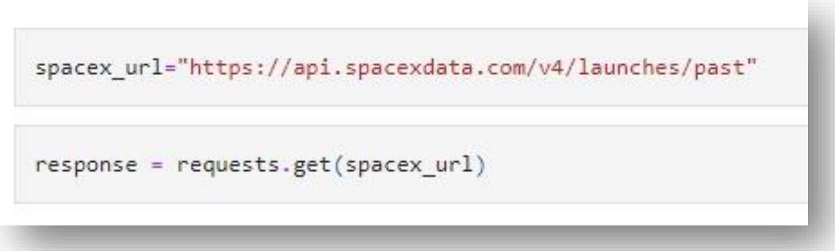
Getting Data From API or Web Page

Creating DataFrame for the Extracted Data

Filter DataFrame as per requirement

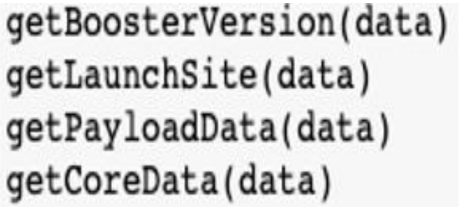
Export to Flat file

Data Collection – SpaceX API

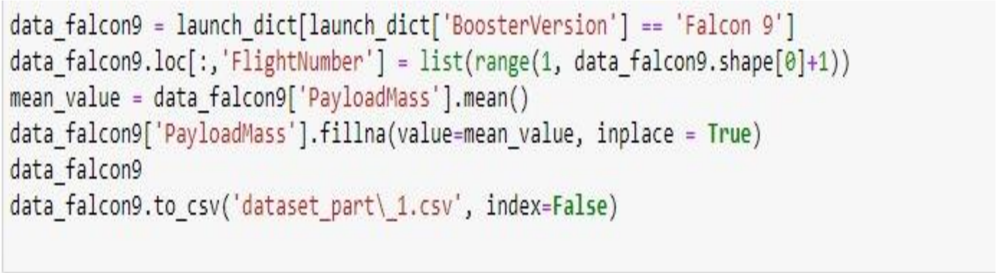


```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

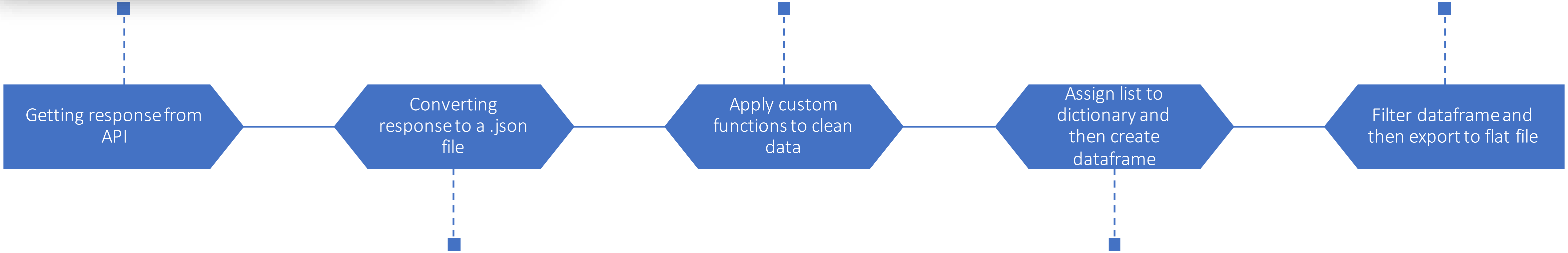
```
response = requests.get(spacex_url)
```



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



```
data_falcon9 = launch_dict[launch_dict['BoosterVersion'] == 'Falcon 9']
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
mean_value = data_falcon9['PayloadMass'].mean()
data_falcon9['PayloadMass'].fillna(value=mean_value, inplace = True)
data_falcon9
data_falcon9.to_csv('dataset_part\_1.csv', index=False)
```



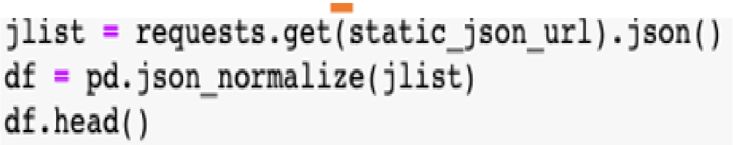
Getting response from API

Converting response to a .json file

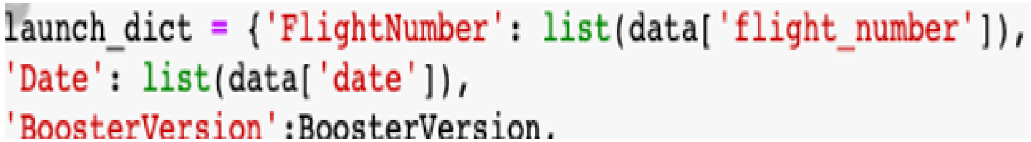
Apply custom functions to clean data

Assign list to dictionary and then create dataframe

Filter dataframe and then export to flat file



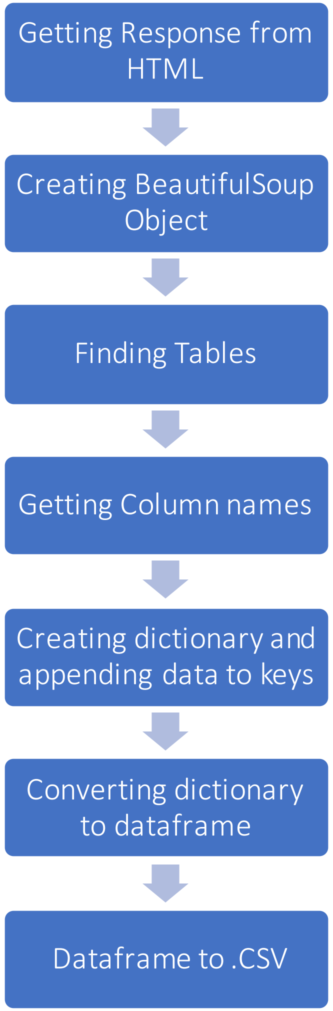
```
jlist = requests.get(static_json_url).json()
df = pd.json_normalize(jlist)
df.head()
```



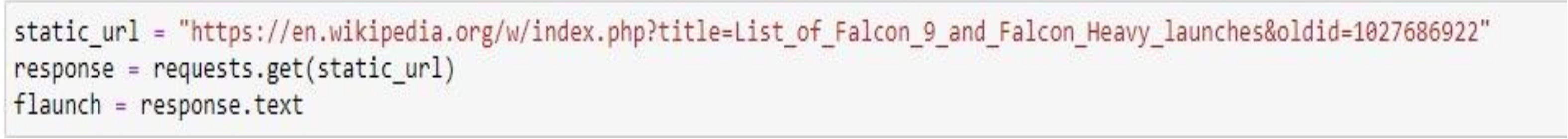
```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
```

Data Collection via SpaceX API Notebook

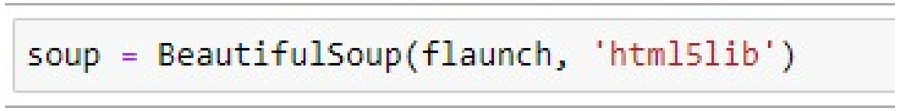
Data Collection - Scraping



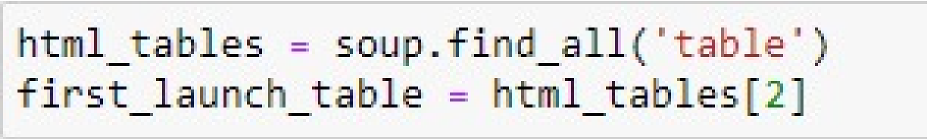
Getting Response from HTML
Creating BeautifulSoup Object
Finding Tables
Getting Column names
Creating dictionary and appending data to keys
Converting dictionary to dataframe
Dataframe to .CSV



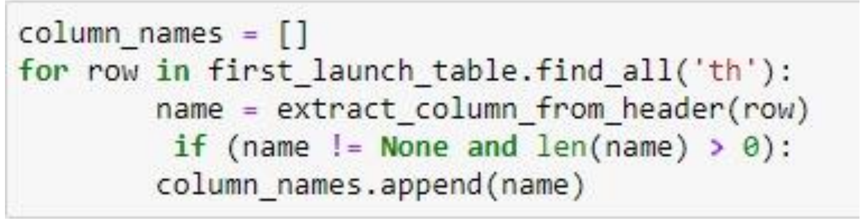
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
response = requests.get(static_url)
flaunch = response.text
```



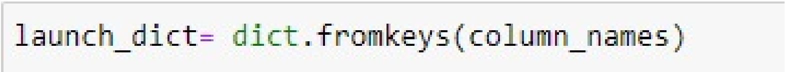
```
soup = BeautifulSoup(flaunch, 'html5lib')
```



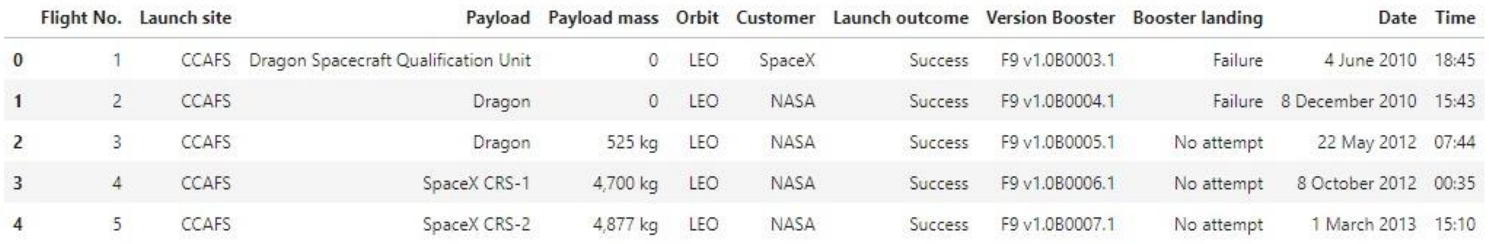
```
html_tables = soup.find_all('table')
first_launch_table = html_tables[2]
```



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



```
launch_dict= dict.fromkeys(column_names)
```

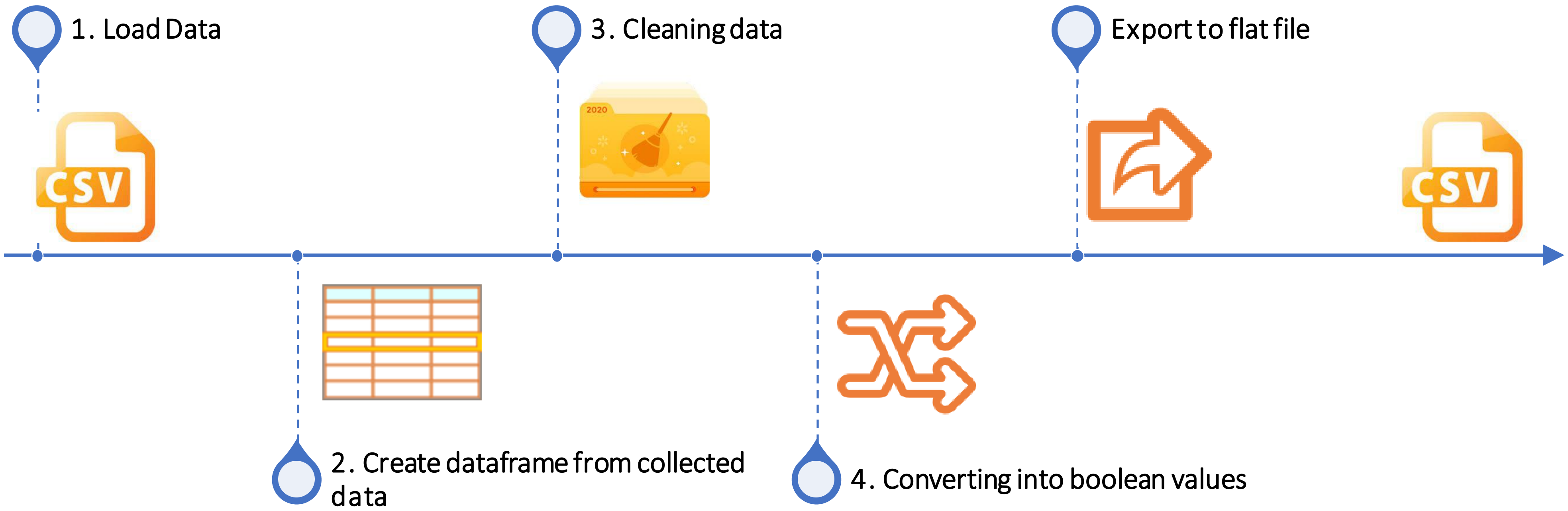


	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	F9 v1.0B0003.1	Failure	4 June 2010	18:45
1	2	CCAFS	Dragon	0	LEO	NASA	Success	F9 v1.0B0004.1	Failure	8 December 2010	15:43
2	3	CCAFS	Dragon	525 kg	LEO	NASA	Success	F9 v1.0B0005.1	No attempt	22 May 2012	07:44
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA	Success	F9 v1.0B0006.1	No attempt	8 October 2012	00:35
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA	Success	F9 v1.0B0007.1	No attempt	1 March 2013	15:10

- Data Collection via Web Scraping Notebook

Data Wrangling

Data wrangling is the process of removing errors and combining complex data sets to make them more accessible and easier to analyze. In this project, we use **One-Hot Encoding** to convert training labels with 1 as landing success and 0 as landing failure.


1. Load Data
3. Cleaning data
Export to flat file
CSV
2020
CSV
2. Create dataframe from collected data
4. Converting into boolean values

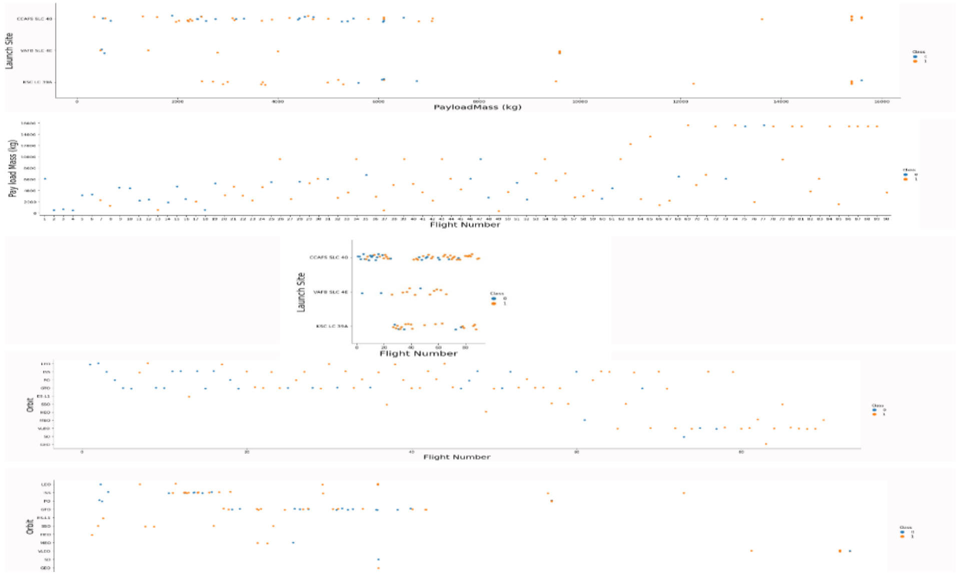
[Data Wrangling Notebook]

EDA with Data Visualization

Scatter Graphs :

- Payload vs Flight Number
- Flight Number vs Launch Site
- Flight Number vs Orbit Type
- Payload vs Launch Site
- Payload vs Orbit Type

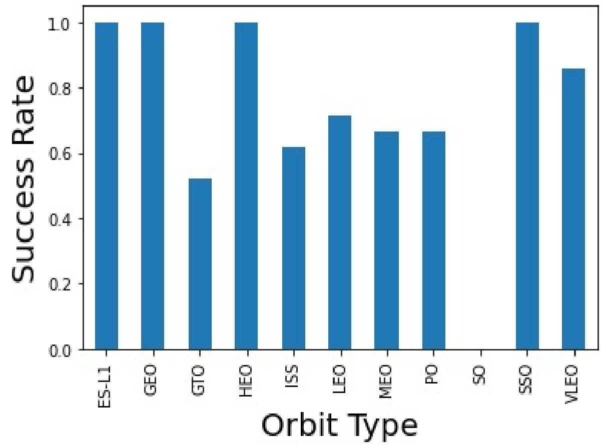
Scatter plot is used here to determine correlation or pattern between different set of variables in order to determine which factors will lead to maximum probability of success in landing outcome.



Bar Graphs :

Success Rate vs Orbit Type

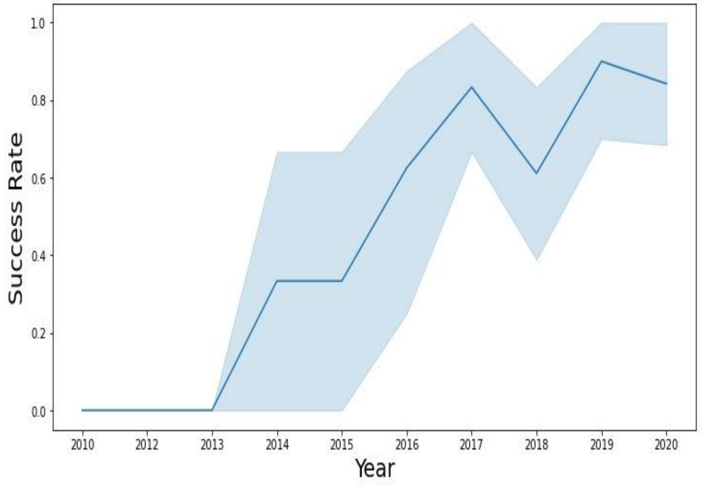
Bar graphs are best suited to represent relation between two categorical variables. In this project it is used to find relation between Success rate and Orbit type.



Line Chart:

Launch Success Yearly Trend

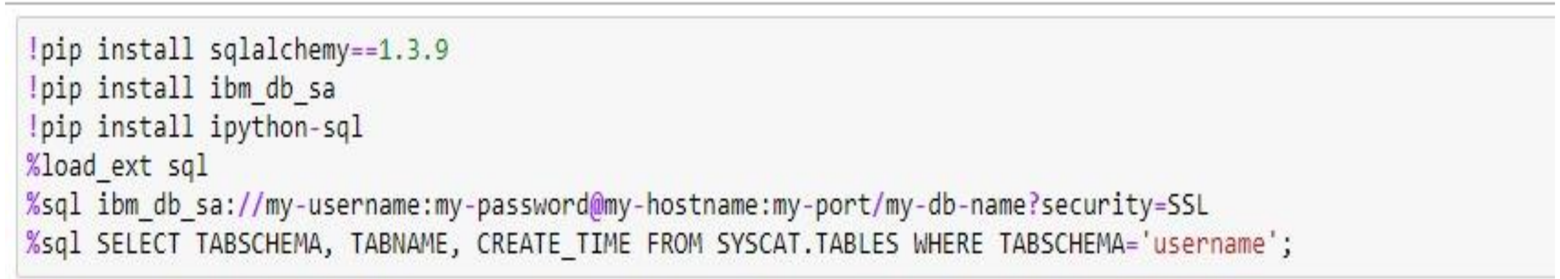
Line chart is used in this project to plot the average launch success trend against previous years which helps in prediction of future launch outcomes.



- EDA with Data Visualization Notebook

EDA with SQL

SQL is designed for a specific purpose to query data contained in a relational database. Due to this it is an indispensable tool for data scientist to deal with real world data driven problems. In this prokect, we are using IBM's DB2 for Cloud as database which is a fully managed SQL service.



```
!pip install sqlalchemy==1.3.9
!pip install ibm_db_sa
!pip install ipython-sql
%load_ext sql
%sql ibm_db_sa://my-username:my-password@my-hostname:my-port/my-db-name?security=SSL
%sql SELECT TABSCHEMA, TABNAME, CREATE_TIME FROM SYSCAT.TABLES WHERE TABSCHEMA='username';
```

SQL queries performed in the project :

- Displaying the names of 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 failed landing_outcomes in drone ship, their booster versions, and launch site names for 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*

- EDA with SQL Notebook

Build an Interactive Map with Folium

Folium makes it easy to visualize geo-spatial data in Python using interactive leaflet map. In this project, we use latitudes and longitudes of each launch site with a Circle Marker, Name Label and Cluster Markers for Successful and failed launches on each site.

Map Objects	Code	Result
Map Marker	folium.Marker(	Map object to create a marker on map.
Icon Marker	folium.Icon(	Create an icon on map.
Circle Marker	folium.Circle(	Create a circle at marker position.
PolyLine	folium.PolyLine(	Creating a line between two points on map.
Marker Cluster Object	MarkerCluster()	Creating a cluster of multiple markers at same position on map.

- Interactive Map with Folium Notebook

Build a Dashboard with Plotly Dash

Dash is a python framework created by plotly for creating interactive web applications written on the top of Flask, Plotly.Js and React.Js . In this project, we used IBM's Theia IDE platform to create interactive dashboard with Pie chart and Scatter plots.

Components	Code	Function
DropDown	dcc.Dropdown(	Creates a dropdown list to select from different launch sites.
RangeSlider	dcc.RangeSlider(	Creates a rangeslider for Payload Mass range selection.
Pie Chart	px.pie(	Displays success percentage of each launch site.
Scatter Plot	px.scatter(	Displays correlation between Payload Mass and Launch outcome.

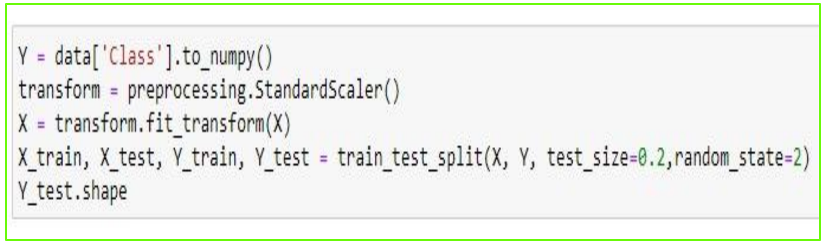
- SpaceX Plotly Dash Lab Notebook

Predictive Analysis (Classification)

ML Predictive Analysis SpaceX Notebook

Building Model

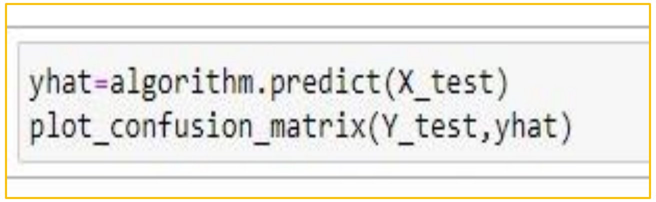
- Load data into data frame and transform into NumPy arrays.
- Standardize and transform data
- Split data into training and test data sets
- Checking number of test samples
- Setting parameters to GridSearchCV and train our model



```
Y = data['Class'].to_numpy()
transform = preprocessing.StandardScaler()
X = transform.fit_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
```

Evaluating Model

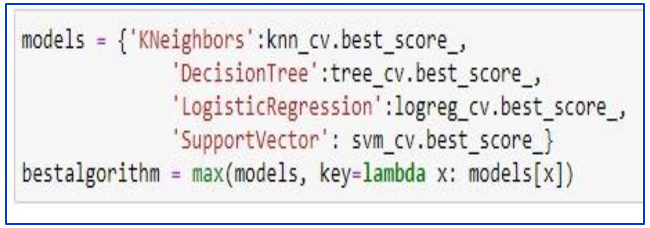
- Check accuracy for each model
- Get best hyperparameters for each algorithm
- Plot confusion matrix for evaluation



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

Finding Best Performing Classification Model

The model accuracy is checked for each model and the model with maximum score is selected.



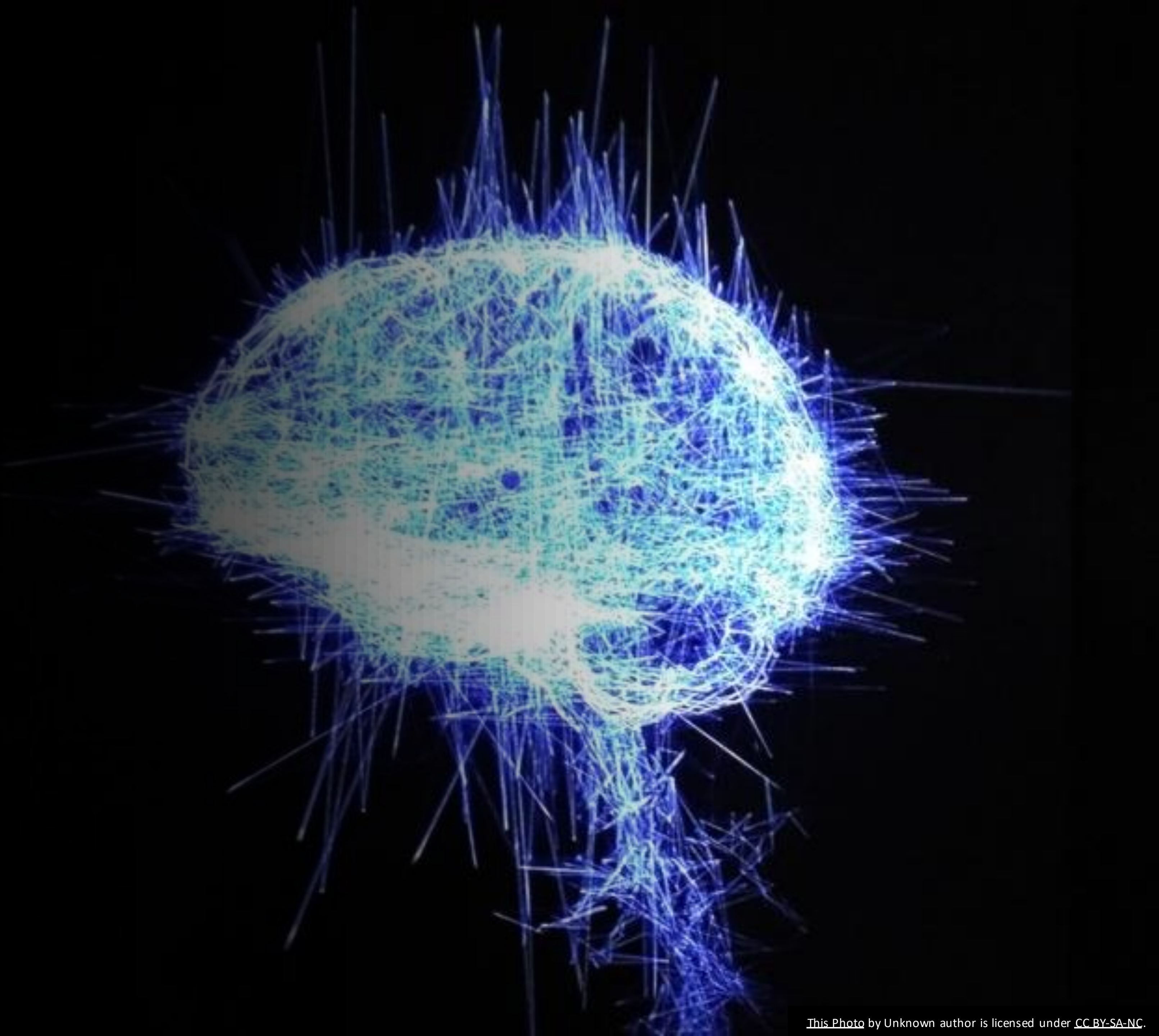
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}
bestalgorithm = max(models, key=lambda x: models[x])
```

Best Model

Results

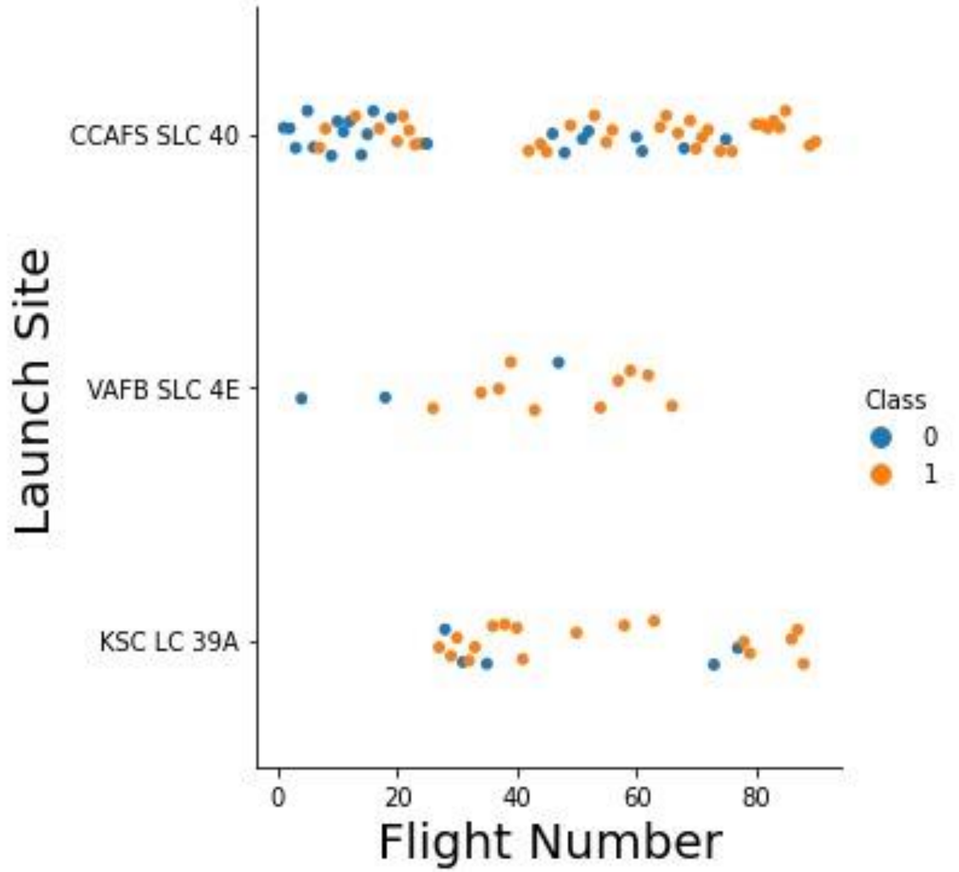
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Insights drawn from EDA



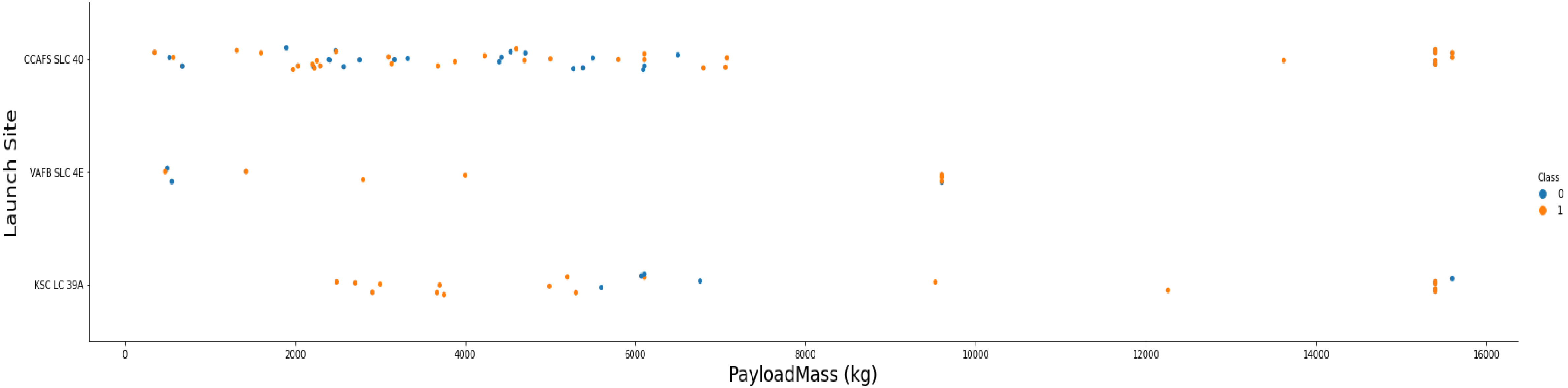
Flight Number vs. Launch Site

- As the flight number increases, so does the success rate for the rocket on each Launch Site.



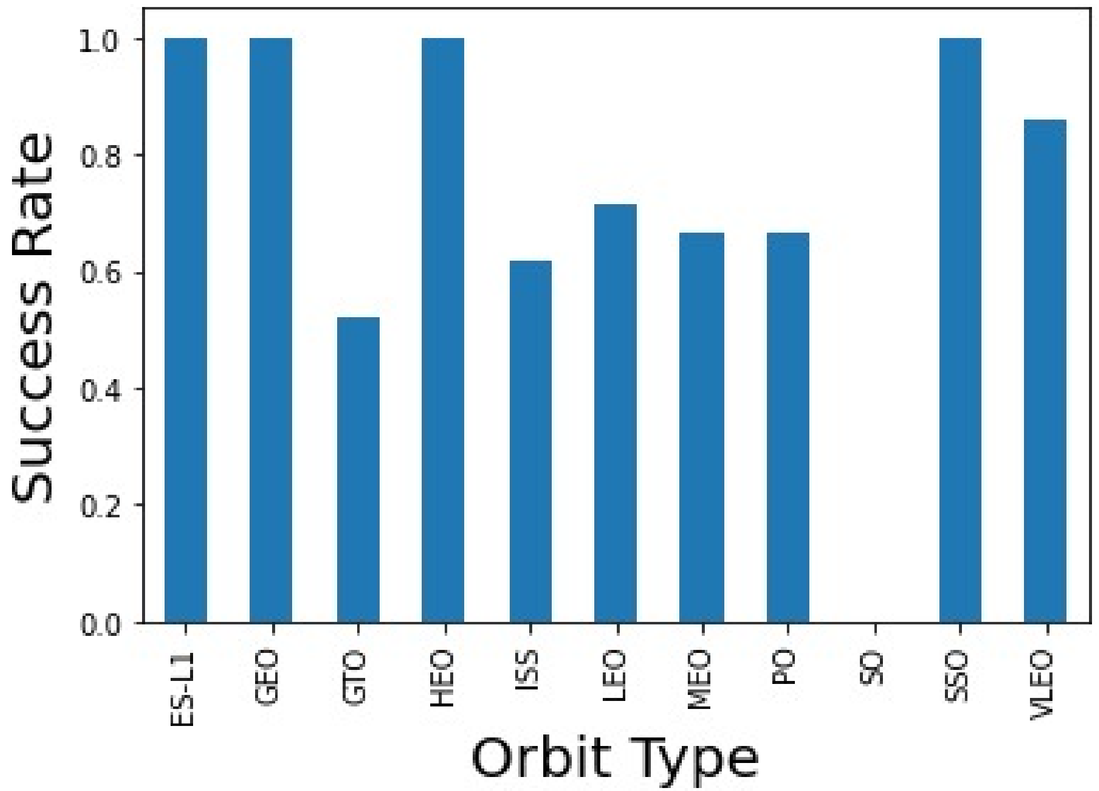
Payload vs. Launch Site

- The greater the payload mass (more than 7000 Kg), higher the success rate. However there's no clear pattern to determine if launch site is dependent on payload mass for success.



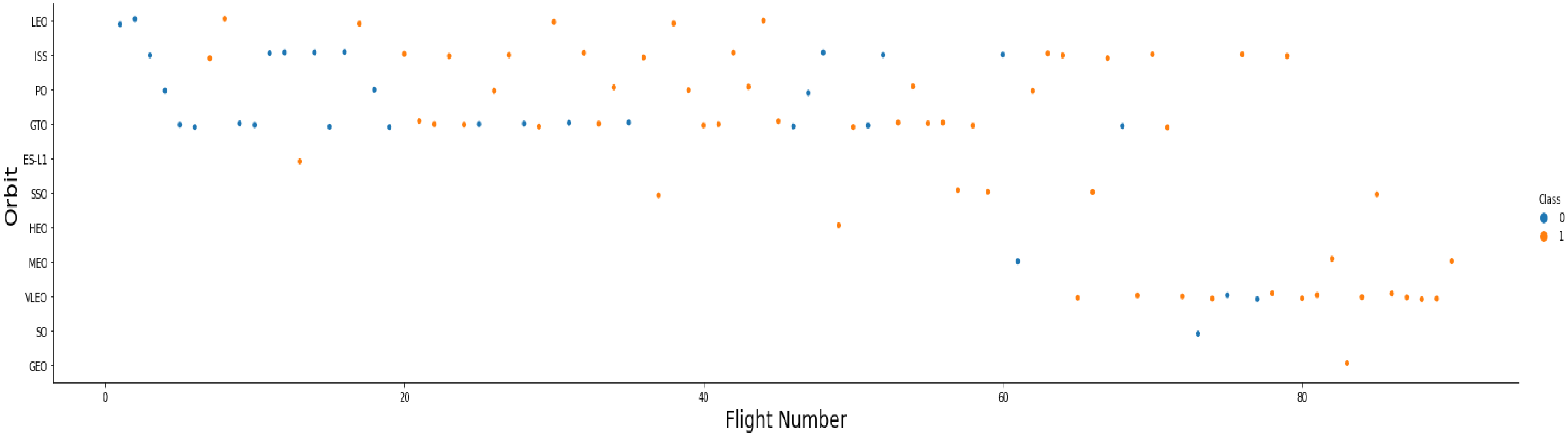
Success Rate vs. Orbit Type

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



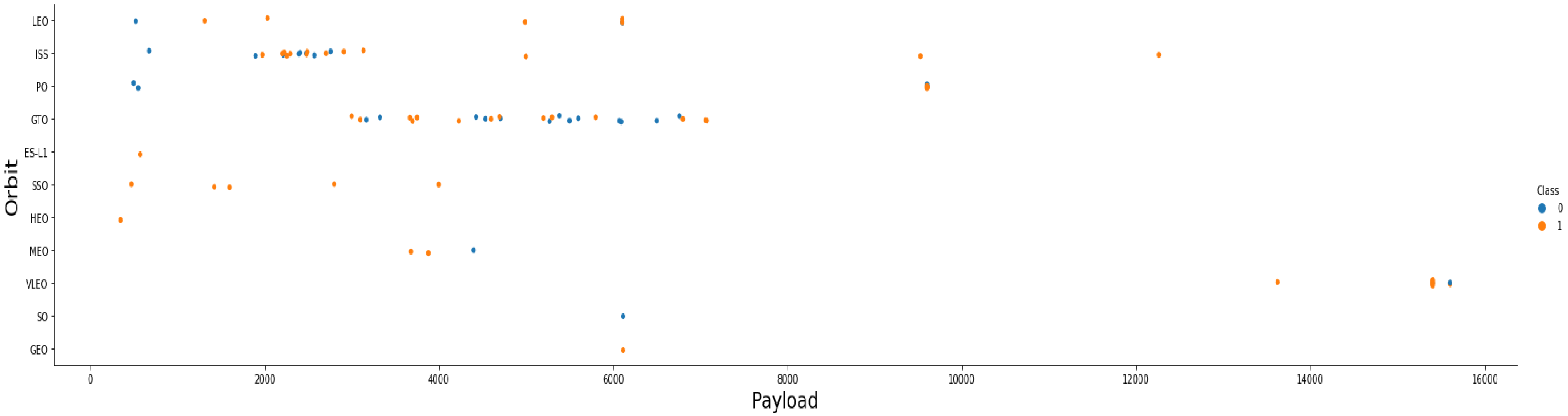
Flight Number vs. Orbit Type

We can see the for LEO success rate increases with number of flights, however there seems no relation between flight number and GTO orbit.



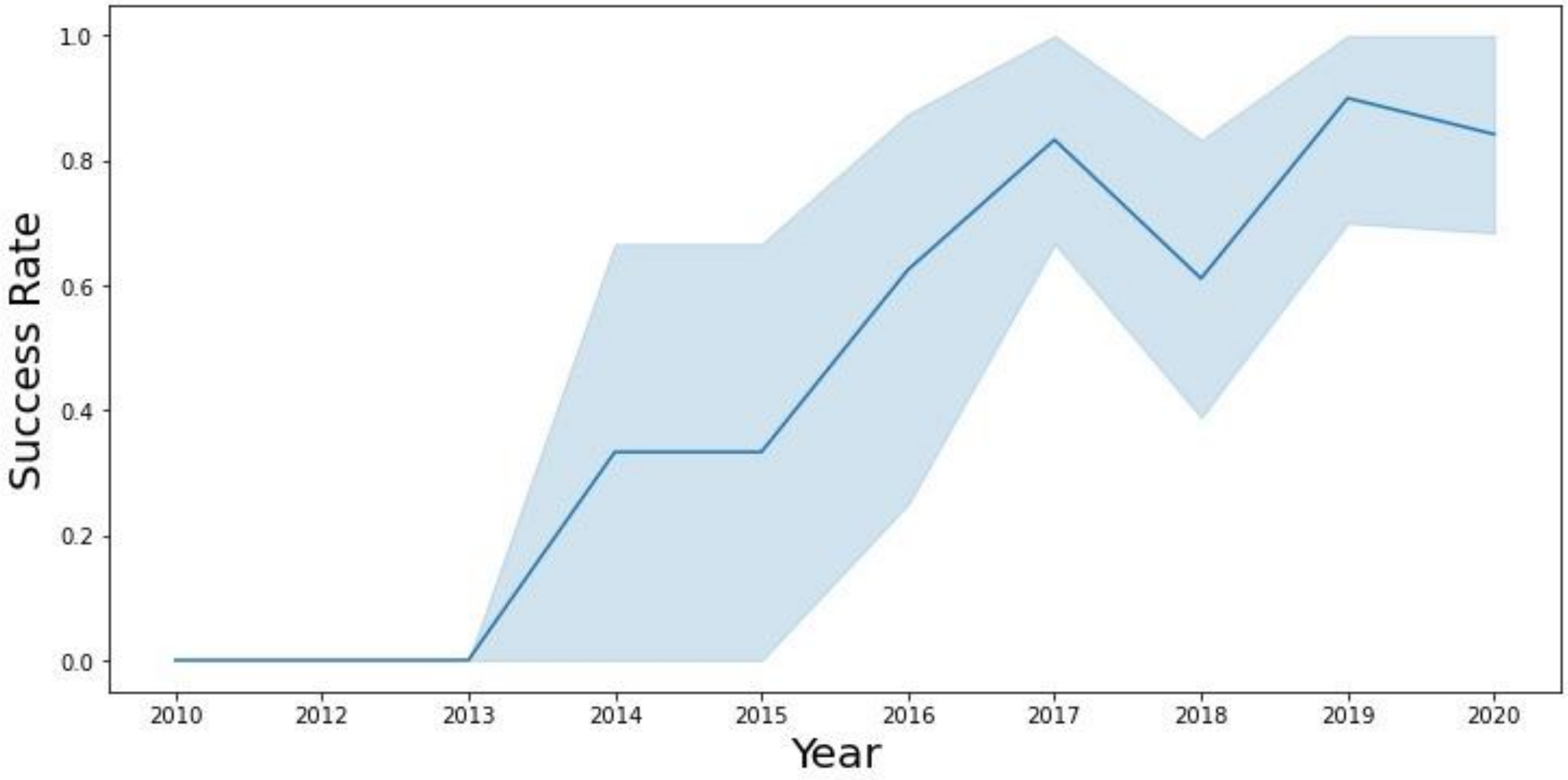
Payload vs. Orbit Type

- We can observe that as heavy payloads has negative influence on MEO, GTO, VLEO orbits.
- LEO, ISS orbits are positively influenced by heavy payloads.



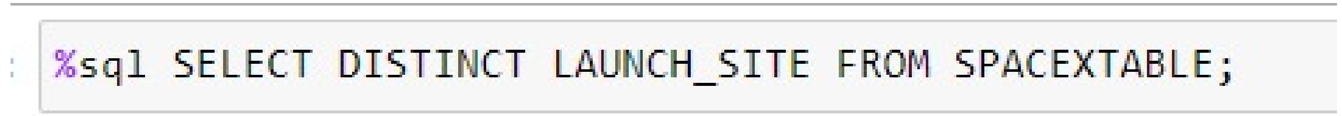
Launch Success Yearly Trend

- We can observer that success rate has been increasing relatively since 2013 though there is a slight dip during 2018.



All Launch Site Names

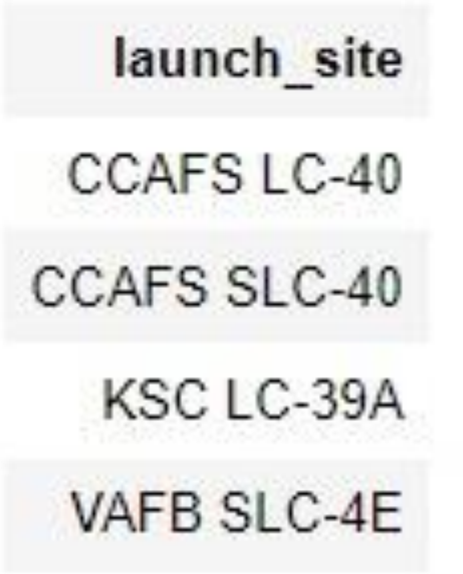
SQL Query



```
%sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTABLE;
```

Description

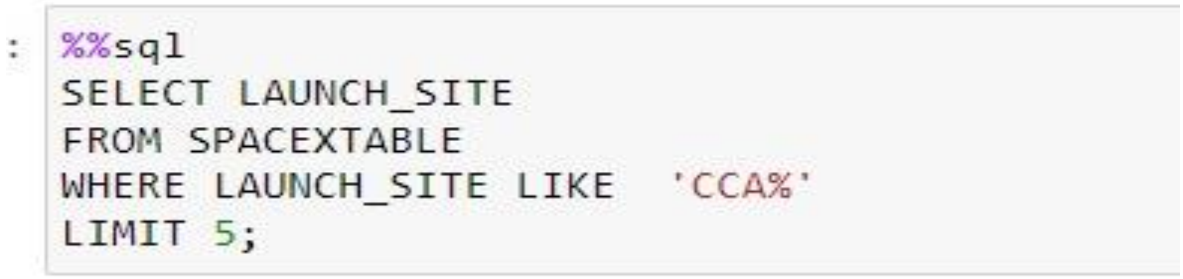
We use keyword DISTINCT in the query to pull unique values for the column "launch_site" from table SPACEX.



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

Launch Site Names Begin with 'CCA'

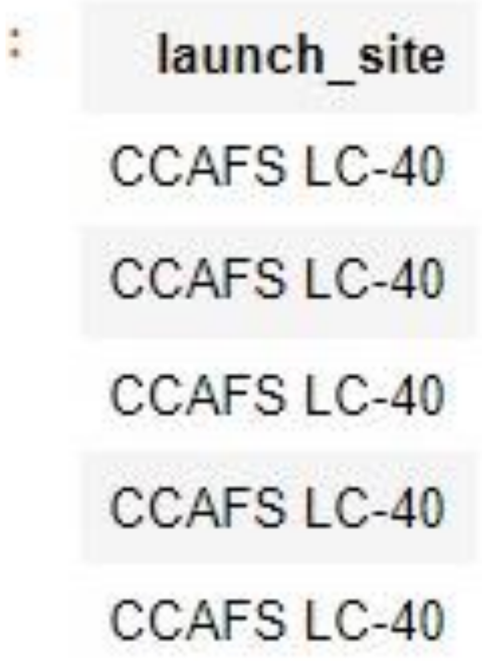
SQL Query



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

Description

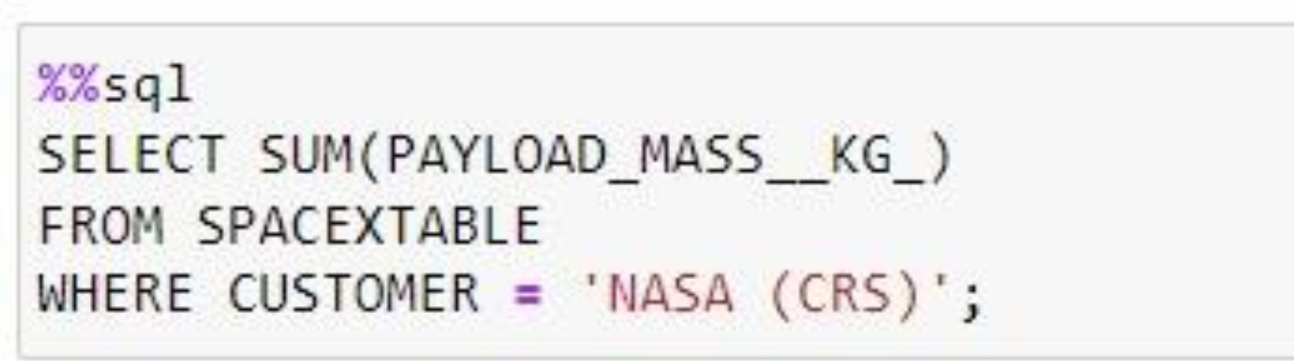
We use keyword 'LIMIT 5' in the query to fetch only 5 records from table SPACEX. The 'LIKE' keyword with wildcard 'CCA%' suggests that names must start with CCA .



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

Total Payload Mass

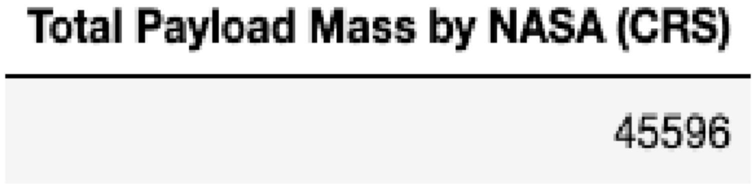
SQL Query



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

Description

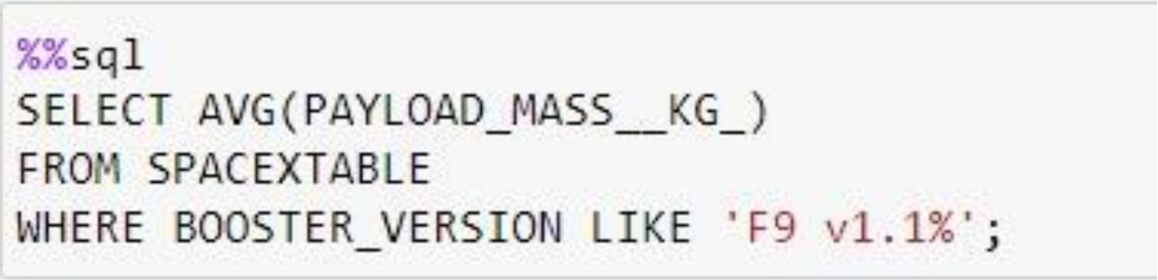
Using the function SUM calculates total in PAYLOAD_MASS__KG_ column and WHERE clause filters the data to fetch customer with name 'NASA (CRS)'.



Total Payload Mass by NASA (CRS)
45596

Average Payload Mass by F9 v1.1

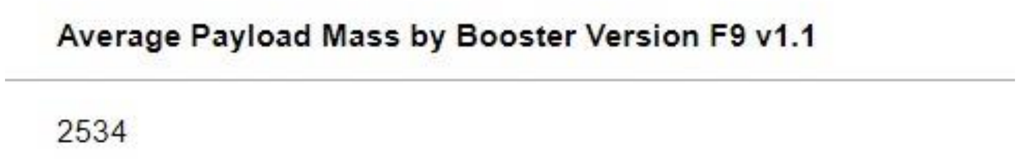
SQL Query



```
%%sql
SELECT AVG(PAYLOAD_MASS__KG_)
FROM SPACEXTABLE
WHERE BOOSTER_VERSION LIKE 'F9 v1.1%';
```

Description

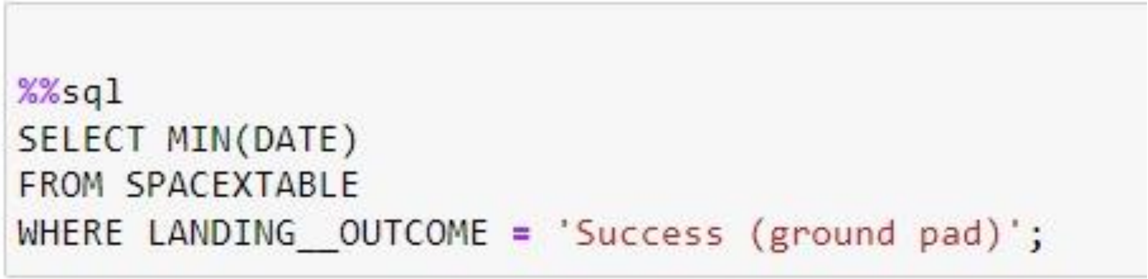
Using the function AVG calculates the average in PAYLOAD_MASS__KG_ column and WHERE clause filters the data to perform calculations only on Booster Version F9 v1.1 .



Average Payload Mass by Booster Version F9 v1.1
2534

First Successful Ground Landing Date

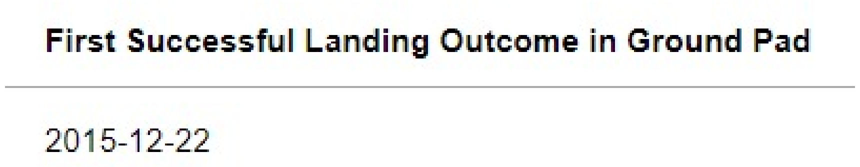
SQL Query



```
%%sql
SELECT MIN(DATE)
FROM SPACEXTABLE
WHERE LANDING__OUTCOME = 'Success (ground pad)';
```

Description

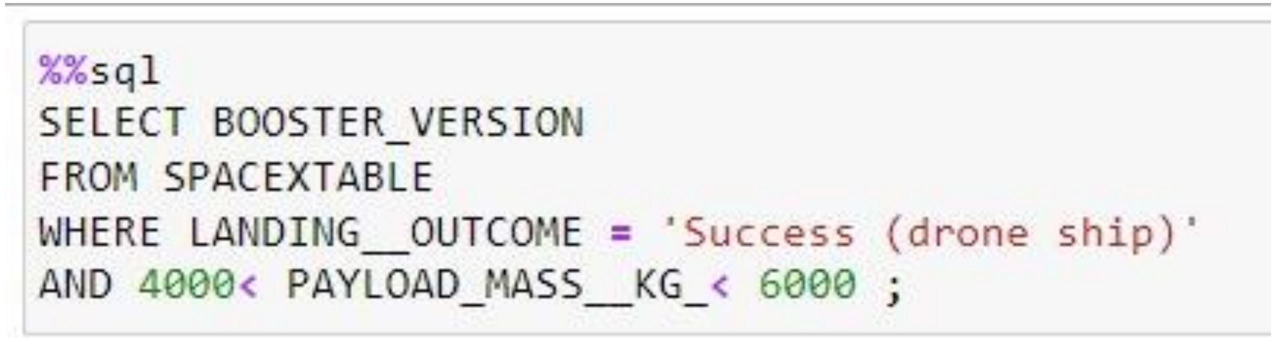
Using the function MIN finds the minimum date in DATE column and WHERE clause filters the data to only perform calculations on Landing_Outcomes with Values "Success (ground pad)".



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

Successful Drone Ship Landing with Payload between 4000 and 6000

SQL Query



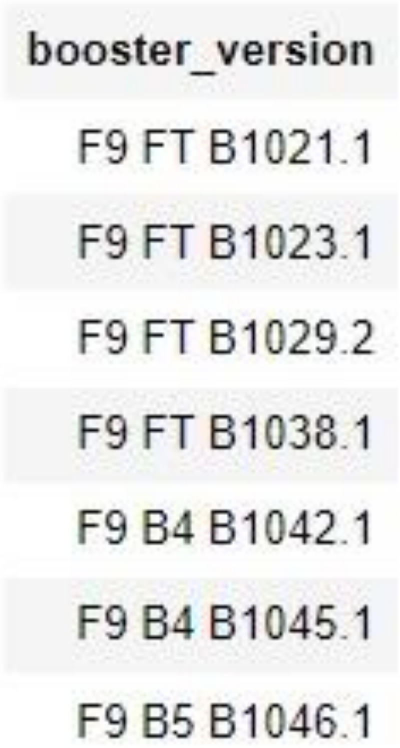
```
%%sql
SELECT BOOSTER_VERSION
FROM SPACEXTABLE
WHERE LANDING__OUTCOME = 'Success (drone ship)'
AND 4000< PAYLOAD_MASS__KG_< 6000 ;
```

Description

Selecting only Booster_Version,

and **WHERE** clause filters the data to Landing_Outcomes with Values "Success (ground pad)"

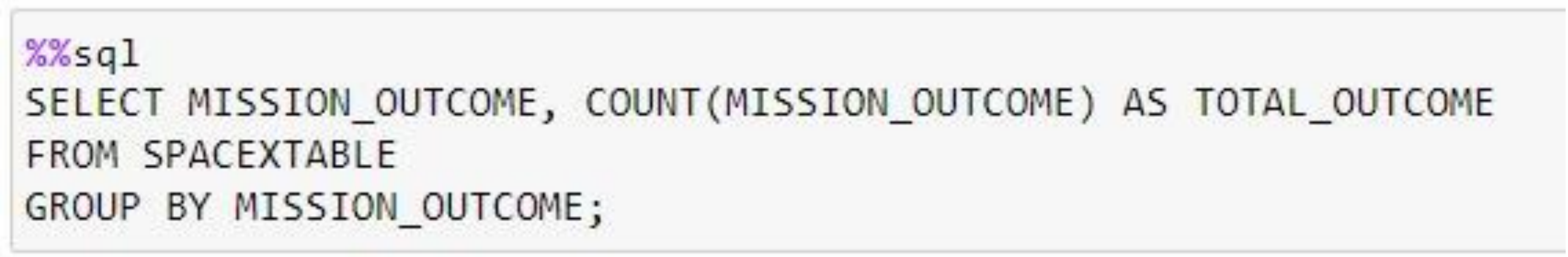
AND clause provides additional filter conditions to select data with Payload mass range between 4000 and 6000.



booster_version
F9 FT B1021.1
F9 FT B1023.1
F9 FT B1029.2
F9 FT B1038.1
F9 B4 B1042.1
F9 B4 B1045.1
F9 B5 B1046.1

Total Number of Successful and Failure Mission Outcomes

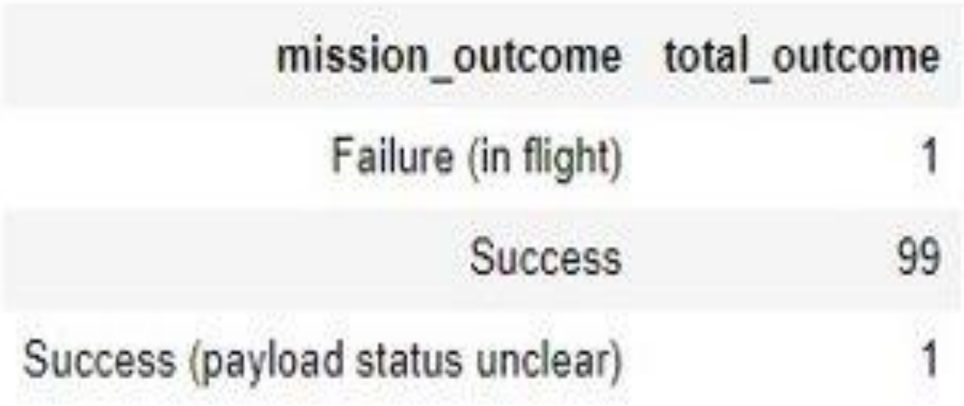
SQL Query



```
%%sql
SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TOTAL_OUTCOME
FROM SPACEXTABLE
GROUP BY MISSION_OUTCOME;
```

Description

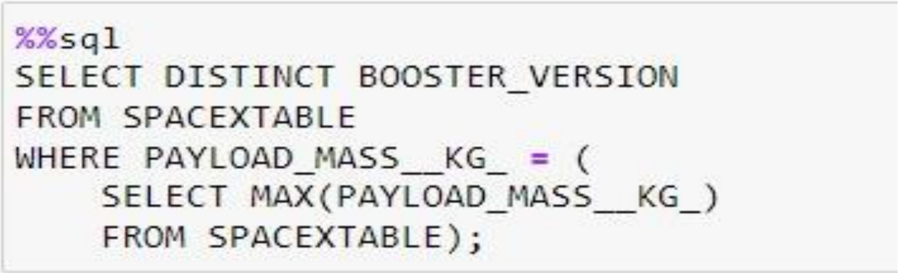
Selecting only MISSION_OUTCOME from SPACEX table, COUNT keyword is used to count total number of successful and failure mission outcomes while grouping them using GROUP BY clause an then results are displayed in a new column with name TOTAL_OUTCOME.



mission_outcome	total_outcome
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

Boosters Carried Maximum Payload

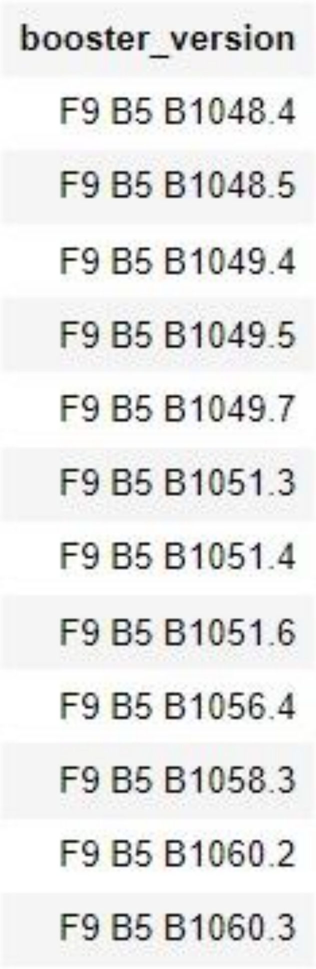
SQL Query



```
%%sql
SELECT DISTINCT BOOSTER_VERSION
FROM SPACEXTABLE
WHERE PAYLOAD_MASS__KG_ = (
    SELECT MAX(PAYLOAD_MASS__KG_)
    FROM SPACEXTABLE);
```

Description

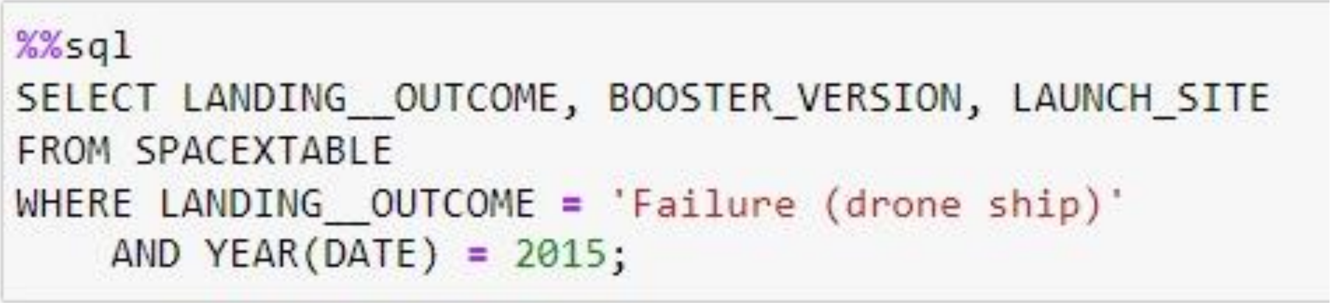
MAX keyword finds the maximum value in the column PAYLOAD_MASS__KG_ and WHERE clause filters the booster versions.



booster_version
F9 B5 B1048.4
F9 B5 B1048.5
F9 B5 B1049.4
F9 B5 B1049.5
F9 B5 B1049.7
F9 B5 B1051.3
F9 B5 B1051.4
F9 B5 B1051.6
F9 B5 B1056.4
F9 B5 B1058.3
F9 B5 B1060.2
F9 B5 B1060.3

2015 Launch Records

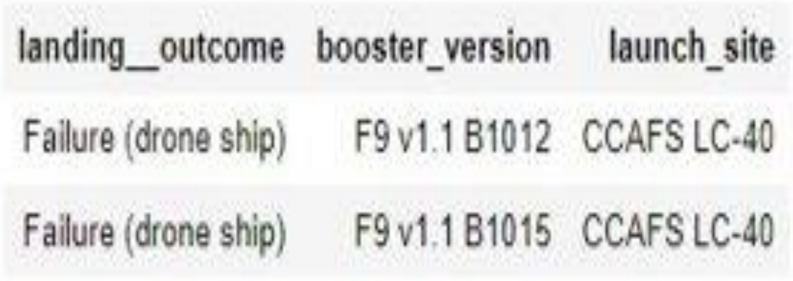
SQL Query



```
%%sql
SELECT LANDING__OUTCOME, BOOSTER_VERSION, LAUNCH_SITE
FROM SPACEXTABLE
WHERE LANDING__OUTCOME = 'Failure (drone ship)'
    AND YEAR(DATE) = 2015;
```

Description

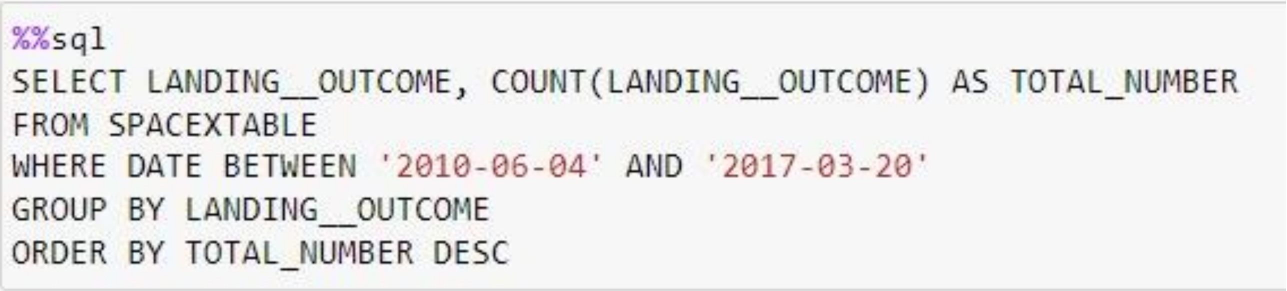
First, we use **Year** function to extract year from the DATE column and then we use **WHERE** clause to filter out the records that has an outcome value of 'Failure (drone ship)' and are from Year 2015.



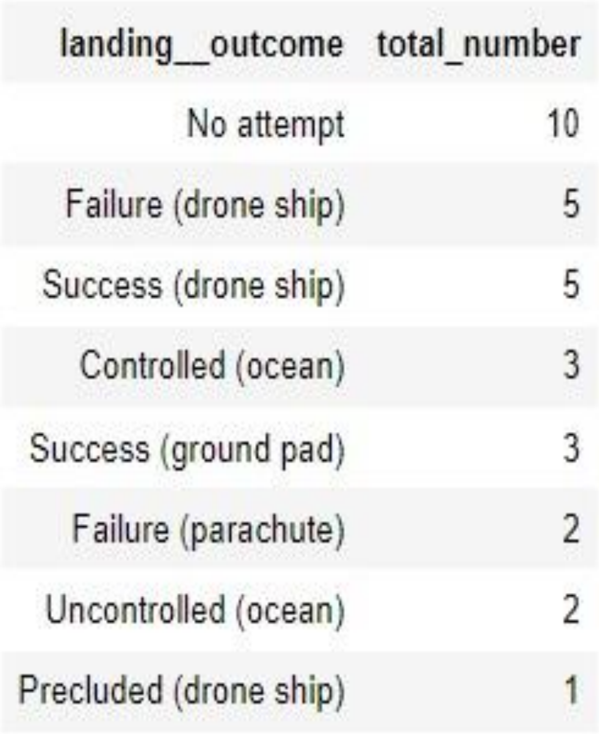
landing__outcome	booster_version	launch_site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

SQL Query



```
%%sql
SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) AS TOTAL_NUMBER
FROM SPACEXTABLE
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY LANDING__OUTCOME
ORDER BY TOTAL_NUMBER DESC
```



landing__outcome	total_number
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

Description

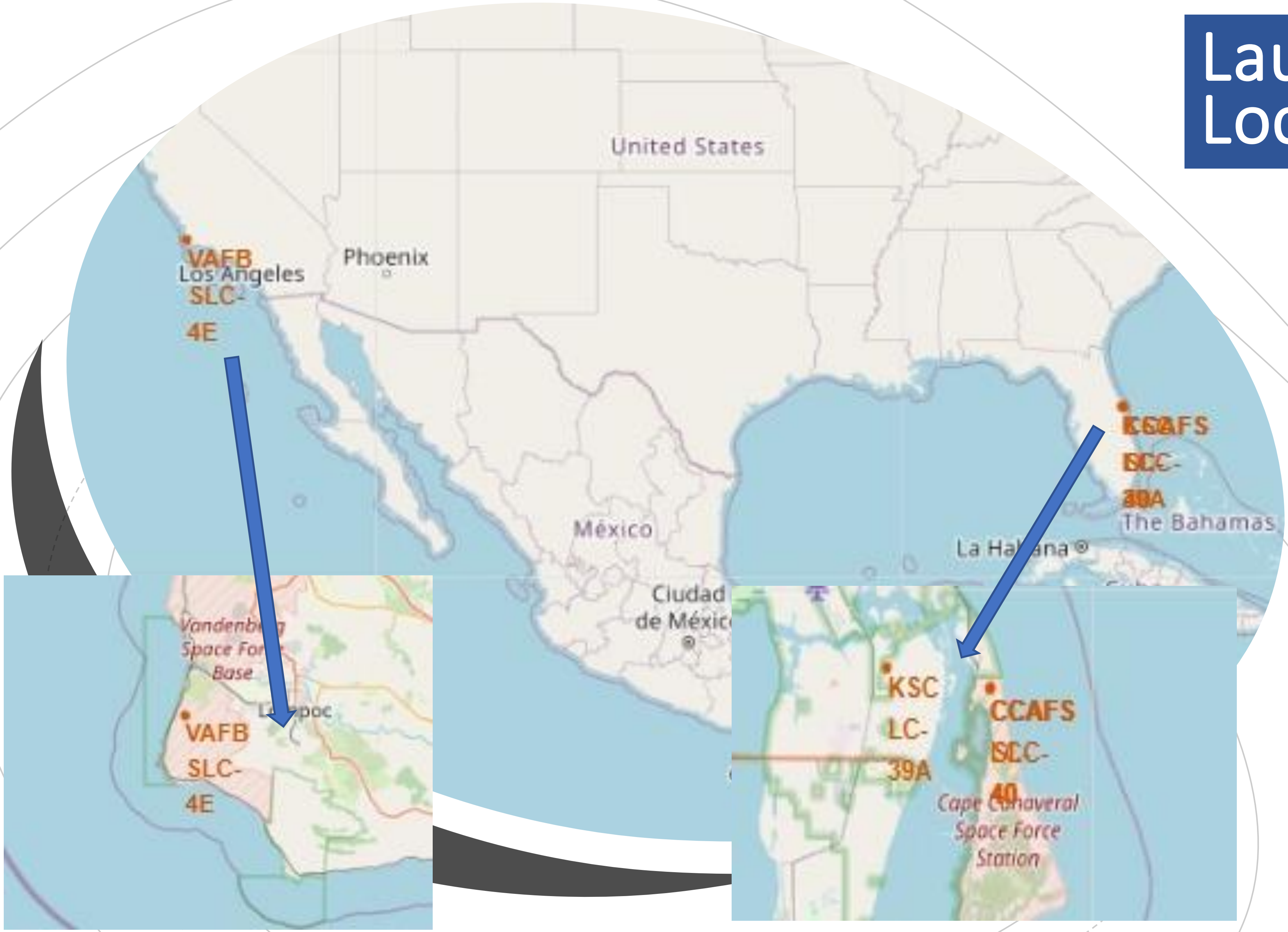
Selecting only Landing_Outcome,

and **WHERE** clause filters the data between '2010-06-04' and '2017-03-20'

GROUP BY is used to group result by landing_outcomes and **ORDER BY** is used to arrange the results in descending order using keyword **DESC**.

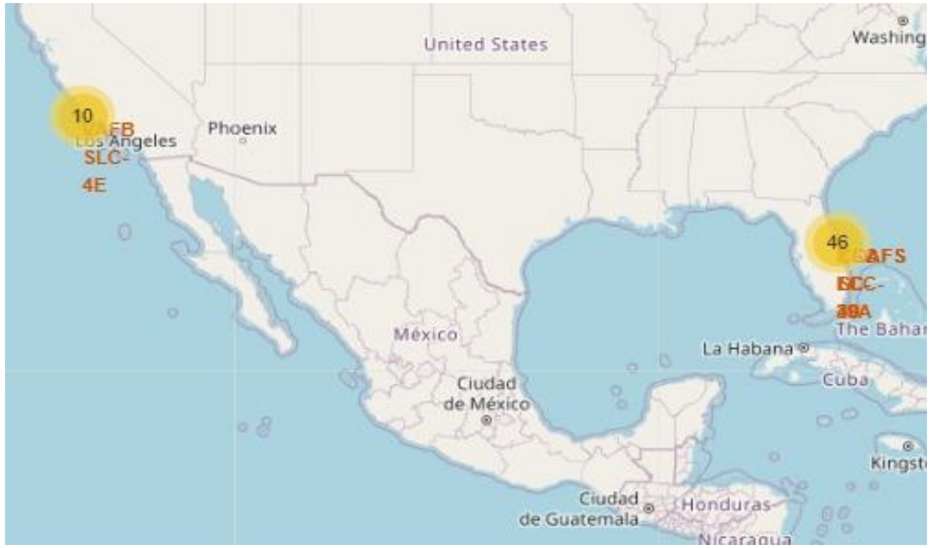
Launch Sites Proximities Analysis

Launch Site Locations



We can see that the SpaceX launch sites are close to the United States of America coasts i.e., Florida and California region.

Launch Outcomes

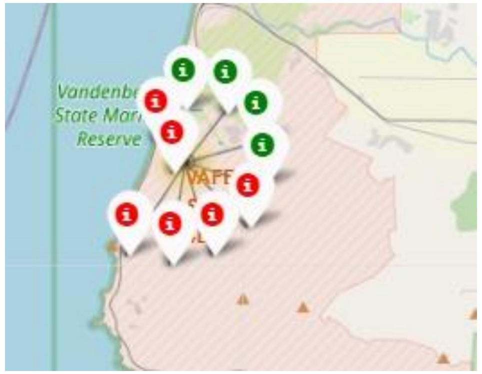


Green Marker ● shows successful launches and Red Marker ● shows failures.

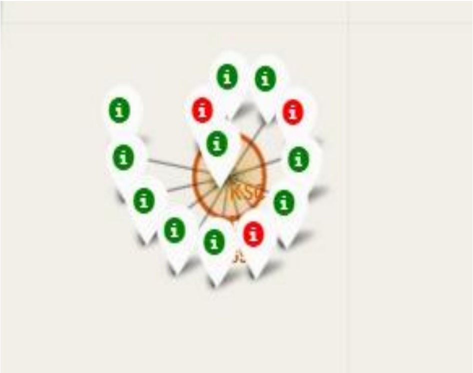
From these screenshots, **it can be easily sighted that KSC LC-39A has the maximum success rate.**

7
AFS
SLC-
40
26
AFS
LC-
40

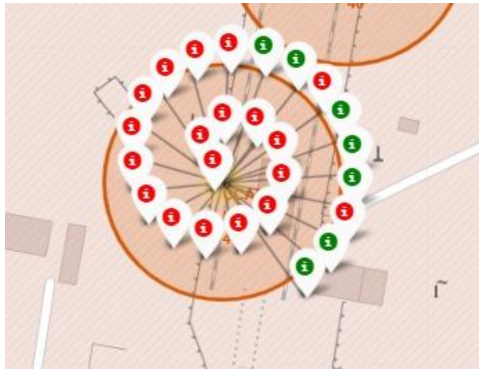
VAFB SLC-4E



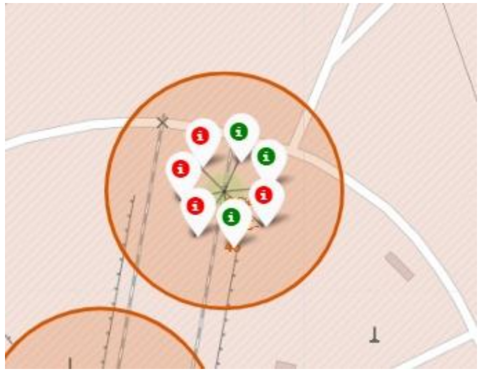
KSC LC-39A



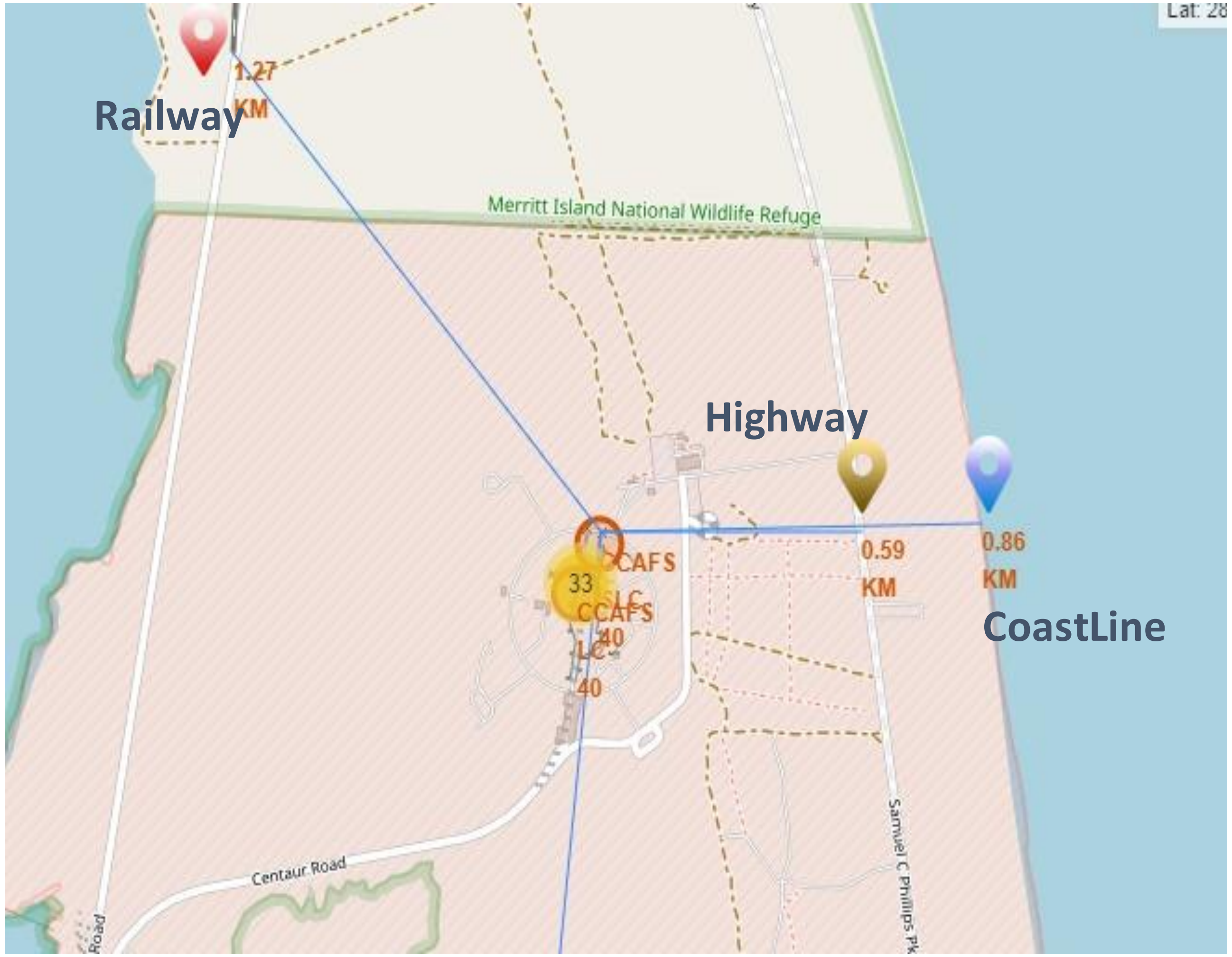
CCAFS LC-40



CCAFS SLC-40



Launch Site Proximities



Conclusion :

- Are launch sites in close proximity to railways?

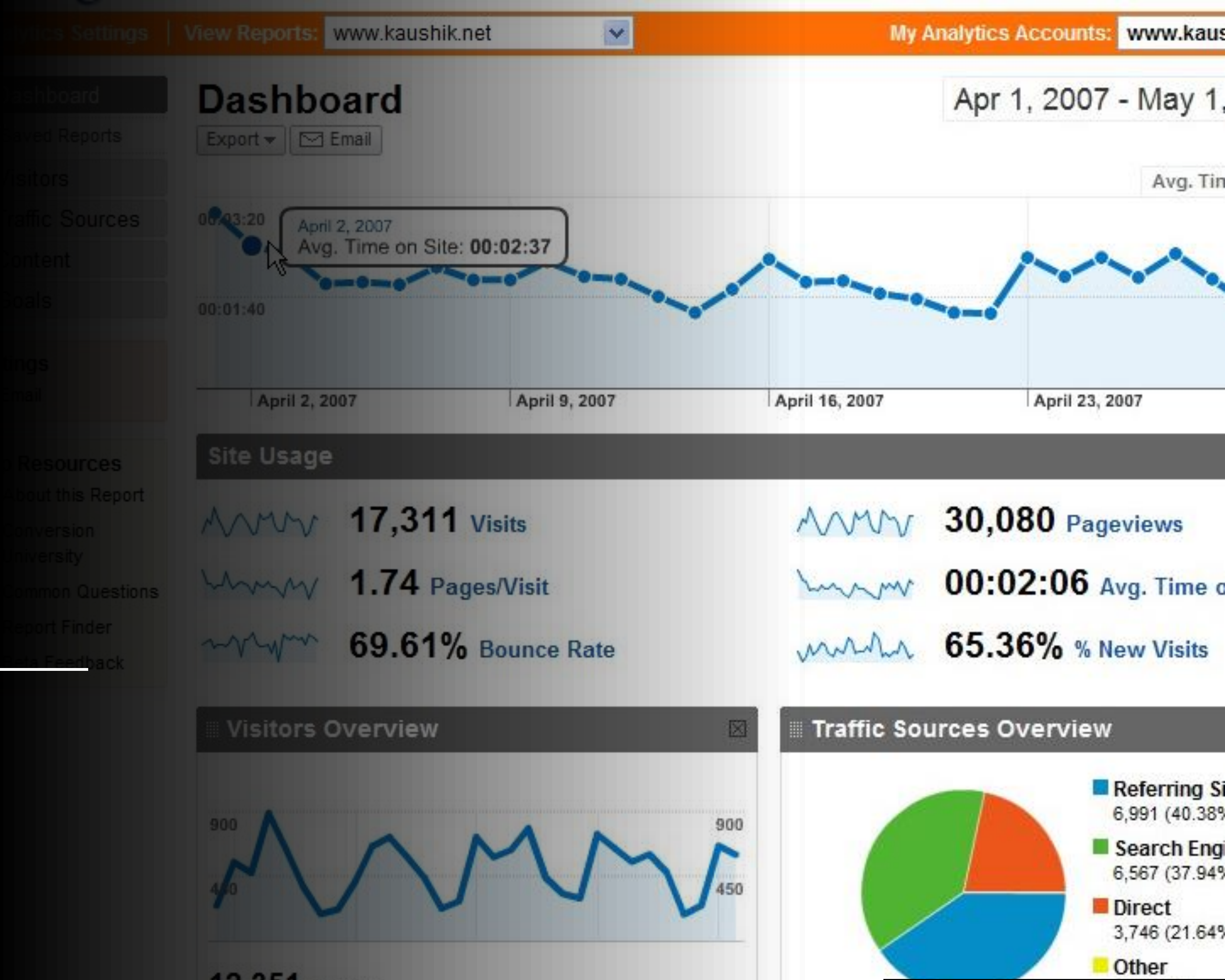
  Yes (Less than 2km)
- Are launch sites in close proximity to highways?

  Yes (Less than 2 Km)
- Are launch sites in close proximity to coastline?

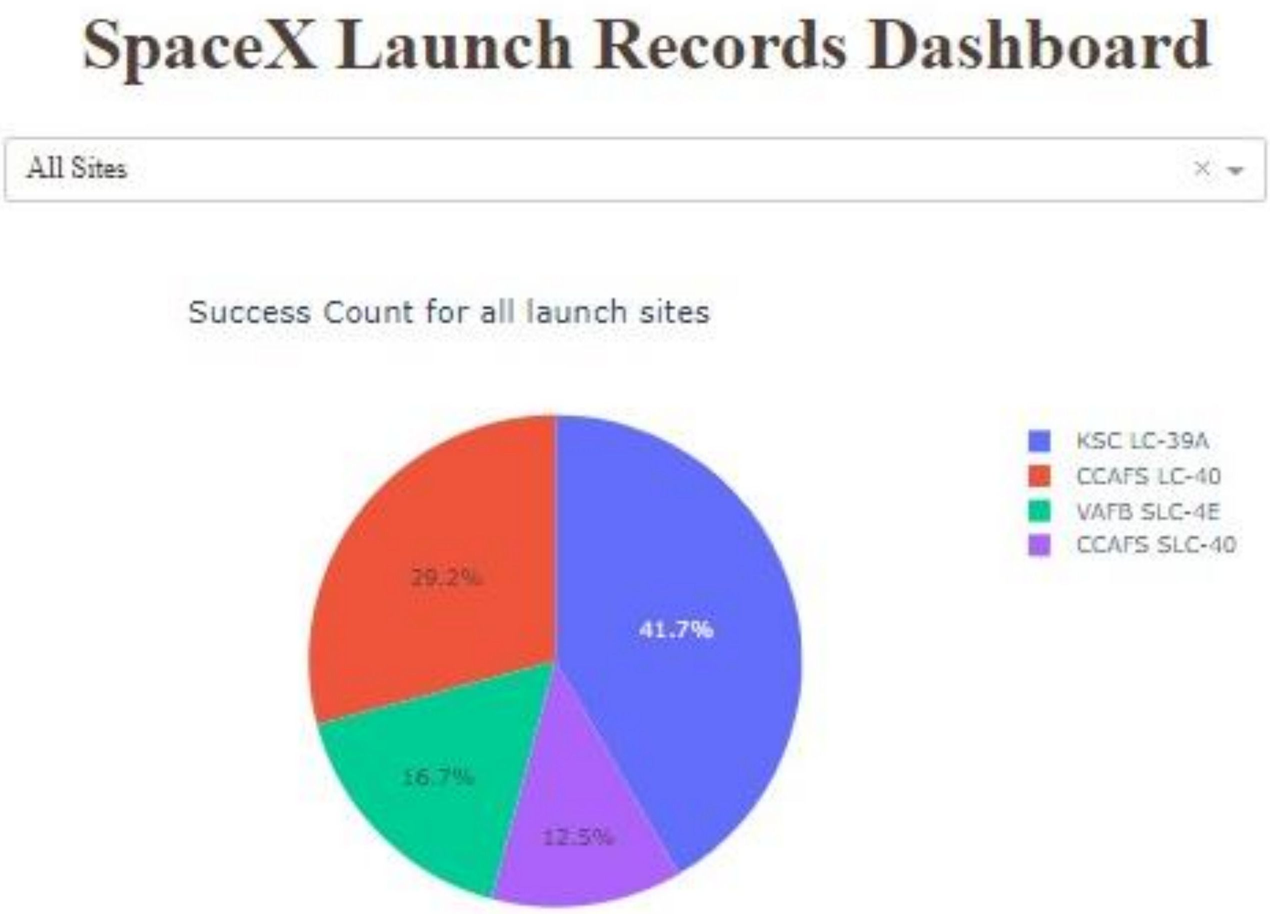
  Yes (Less than 5 Km)
- Do launch sites keep certain distance away from cities?

  Yes (More than 15 Km)

Build a Dashboard with Plotly Dash



Launch Success Count for All Sites

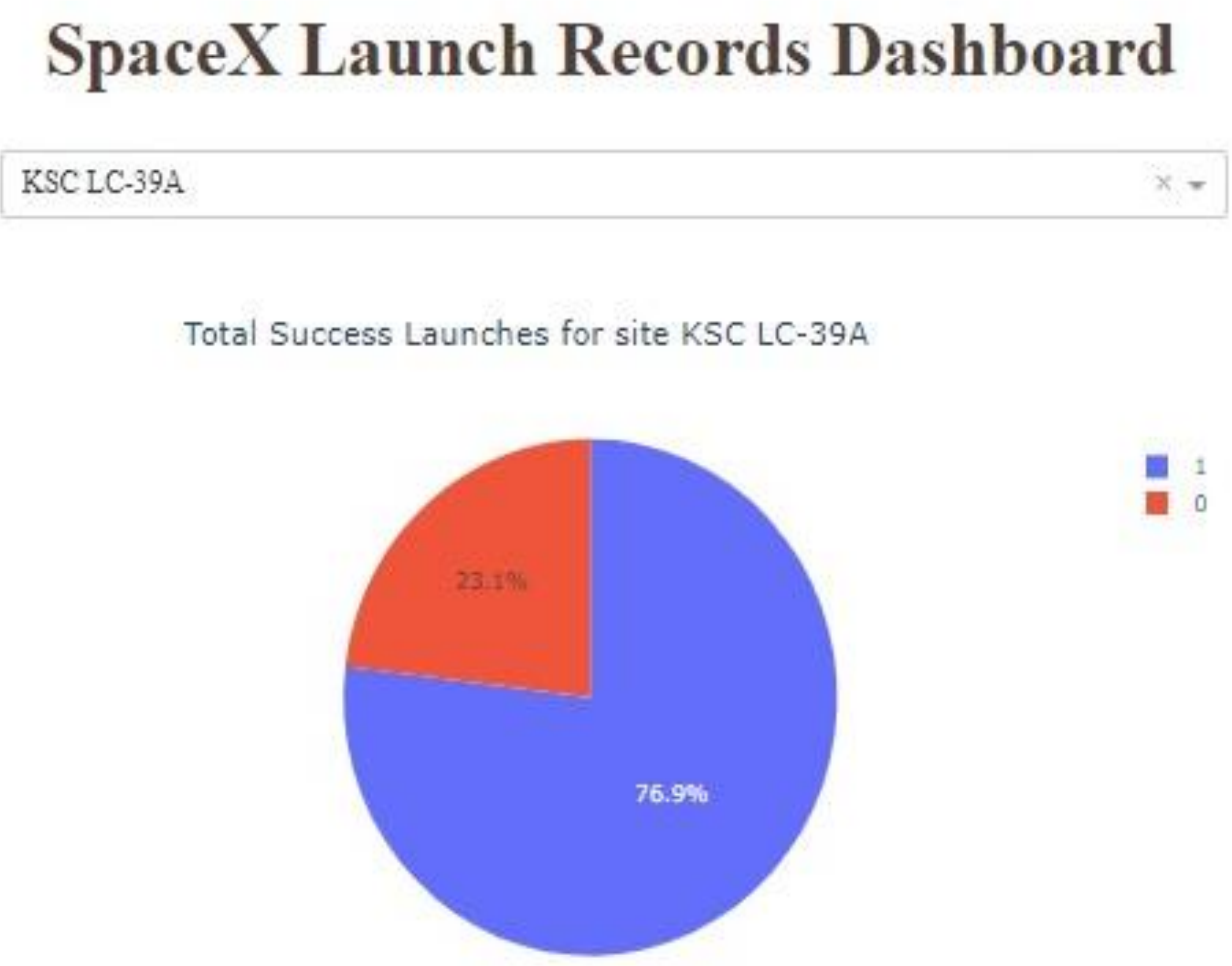


SpaceX Launch Records Dashboard

All Sites

This can be sighted from the image that **KSC LC-39A** has the most number of successful launches among all the sites.

Launch Site with Highest Launch Success Ratio



SpaceX Launch Records Dashboard

KSC LC-39A

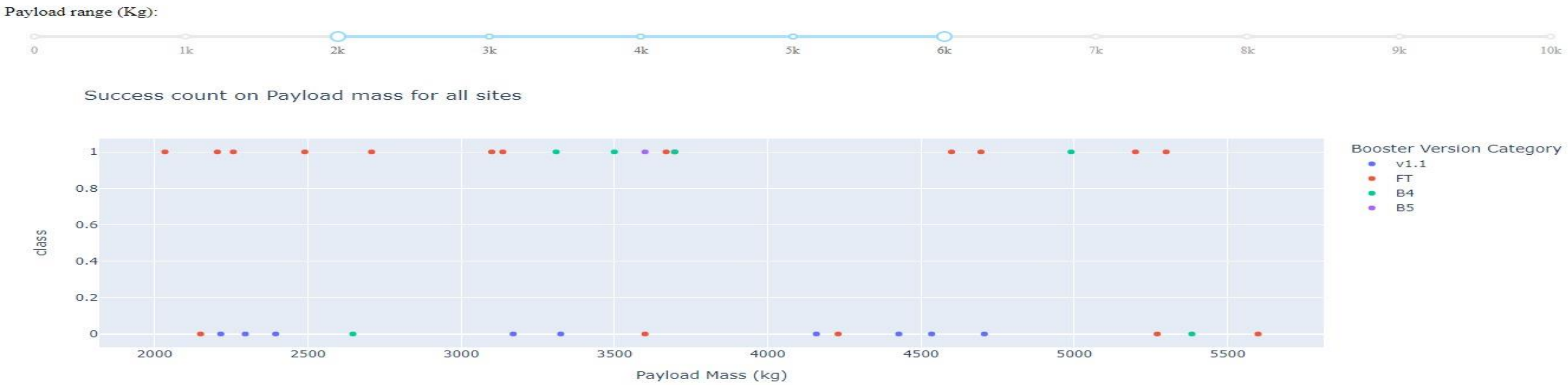
KSC LC-39A achieved a **76.9%** success rate while getting **23.1%** failure rate.

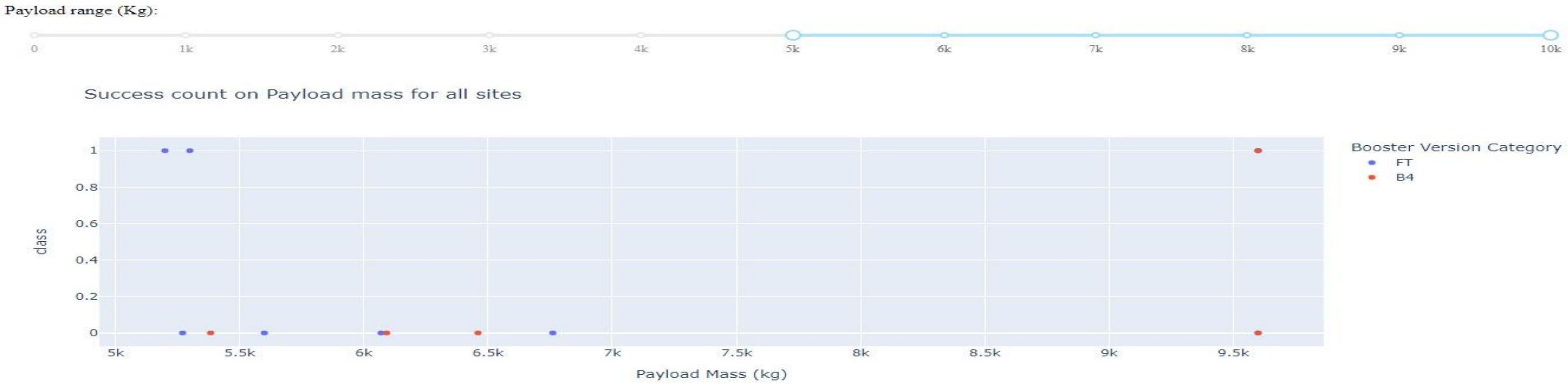
Further Insights obtained from visual analysis :

- **KSC LC-39A** has the highest launch success rate.
- Highest launch success rate is between **2000 Kg - 10000 Kg** payload range.
- Lowest launch success rate is between **0 Kg - 1000 Kg** payload range
- F9 Booster Version **FT** has the highest success rate among all (v1.0, v1.1, FT, B4, B5, etc.)

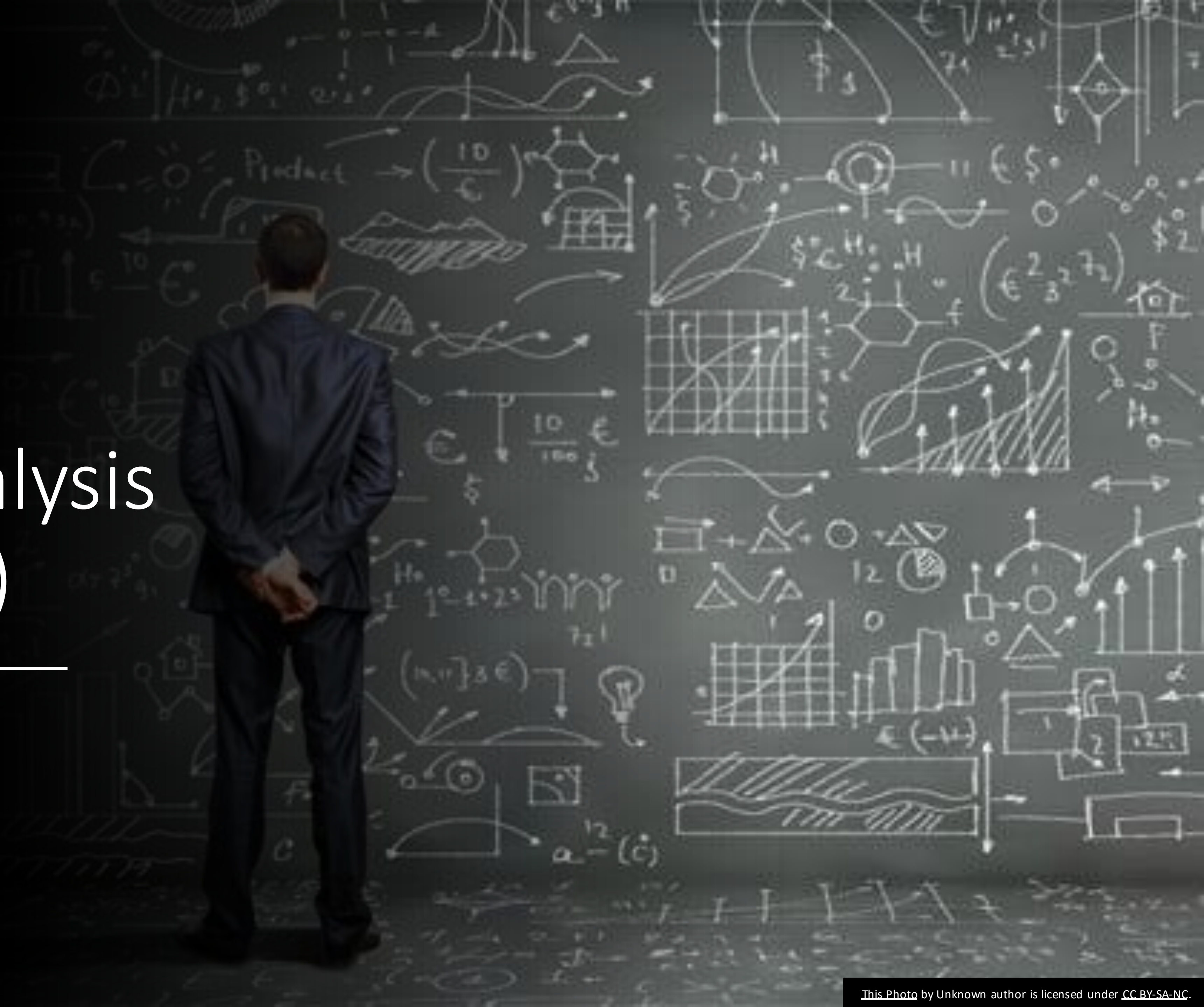
Payload vs. Launch Outcome Scatter Plot for All Sites

We can see the success rate for low weighted payloads is higher than that of high weighted payloads.



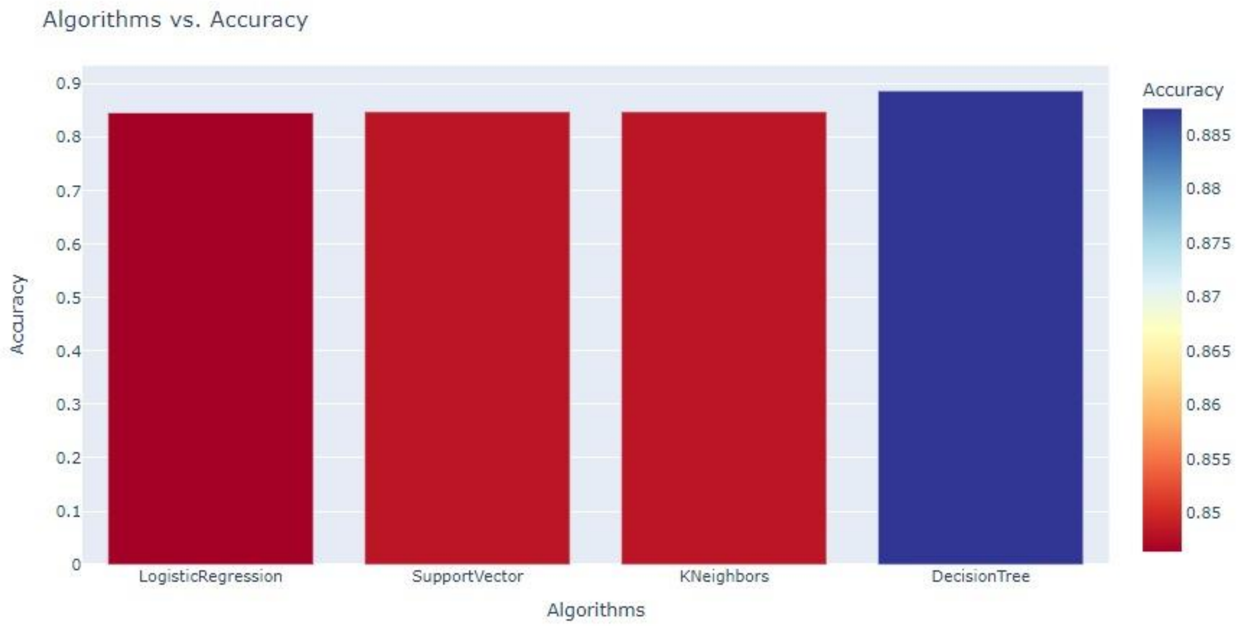


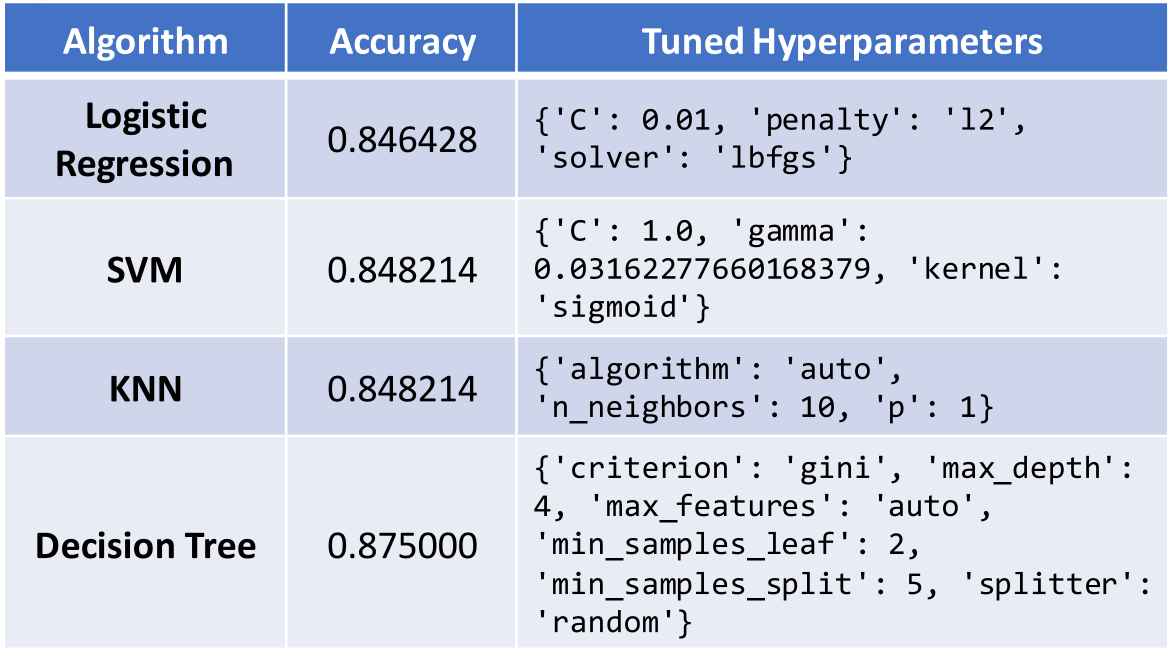
Predictive Analysis (Classification)



Classification Accuracy

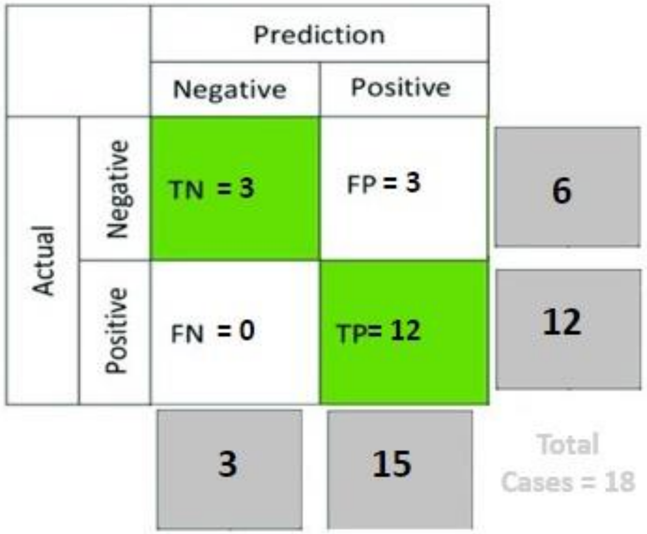
All four models had same accuracy of 83% on the test data. However as we can see below Decision Tree has performed best with maximum accuracy of 87.5% in training.



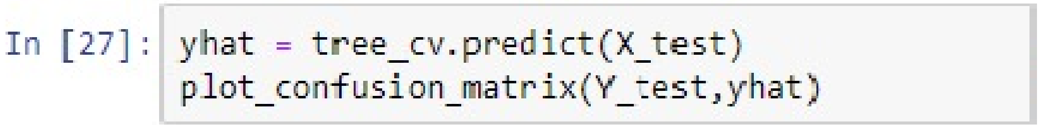


Algorithm	Accuracy	Tuned Hyperparameters
Logistic Regression	0.846428	`{'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}`
SVM	0.848214	`{'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}`
KNN	0.848214	`{'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}`
Decision Tree	0.875000	`{'criterion': 'gini', 'max_depth': 4, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}`

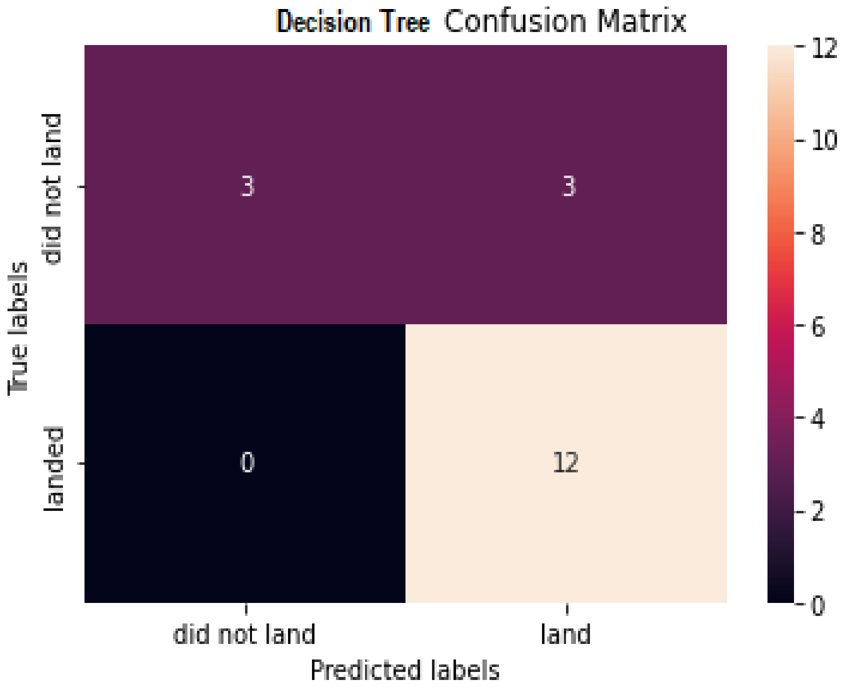
Confusion Matrix



		Prediction		
		Negative	Positive	
Actual	Negative	TN = 3	FP = 3	6
	Positive	FN = 0	TP= 12	12
		3	15	Total Cases = 18



```
In [27]: yhat = tree_cv.predict(X_test)
         plot_confusion_matrix(Y_test,yhat)
```



- **Accuracy:** (TP+TN)/Total = (12+3)/18 = 0.8333
- **Misclassification Rate:** (FP+FN)/Total = (3+0)/18 = 0.1667
- **True Positive Rate:** TP/Actual Positive =12/12 = 1
- **False Positive Rate:** FP/Actual Negative = 3/6 = 2
- **True Negative Rate** : TN/Actual Negative = 3/6 = 2
- **Precision:** TP/Predicted Positive = 12/15 = 0.8
- **Prevalence:** Actual Positive/Total = 12/18 = 0.6667

Conclusions

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

Success rates for SpaceX launches has been increasing relatively with time.

KSC LC-39A had the most successful launches however increasing the payload mass impacts negatively on success rate.

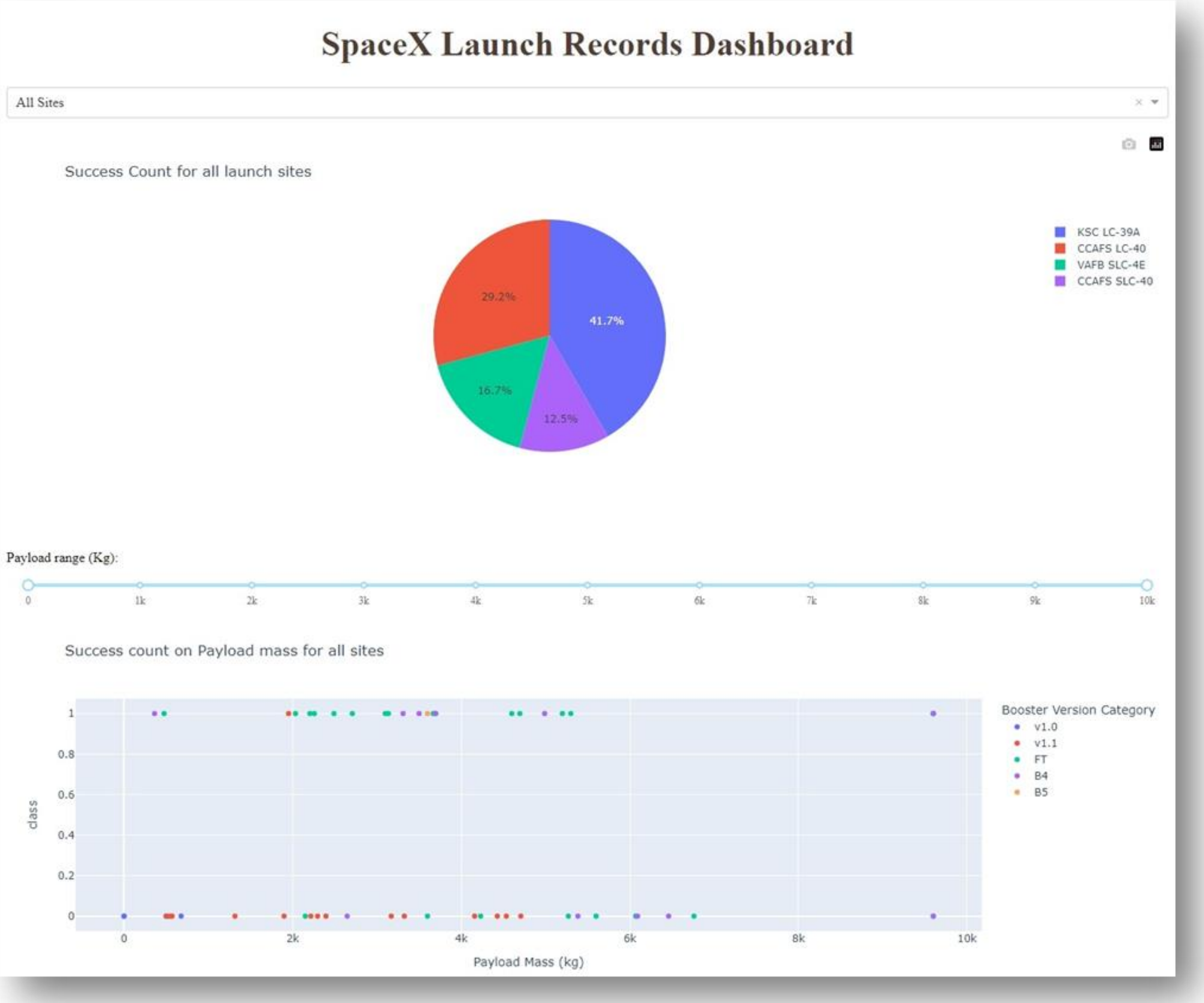
Decision Tree Classification Algorithm is best suited Machine Learning Model for the given data set.

Appendix

Live Plotly Dashboard

Microsoft VS Code Tool

Live Dashboard on “PythonAnywhere”



SpaceX Launch Records Dashboard

All Sites

Success Count for all launch sites

Payload range (Kg):

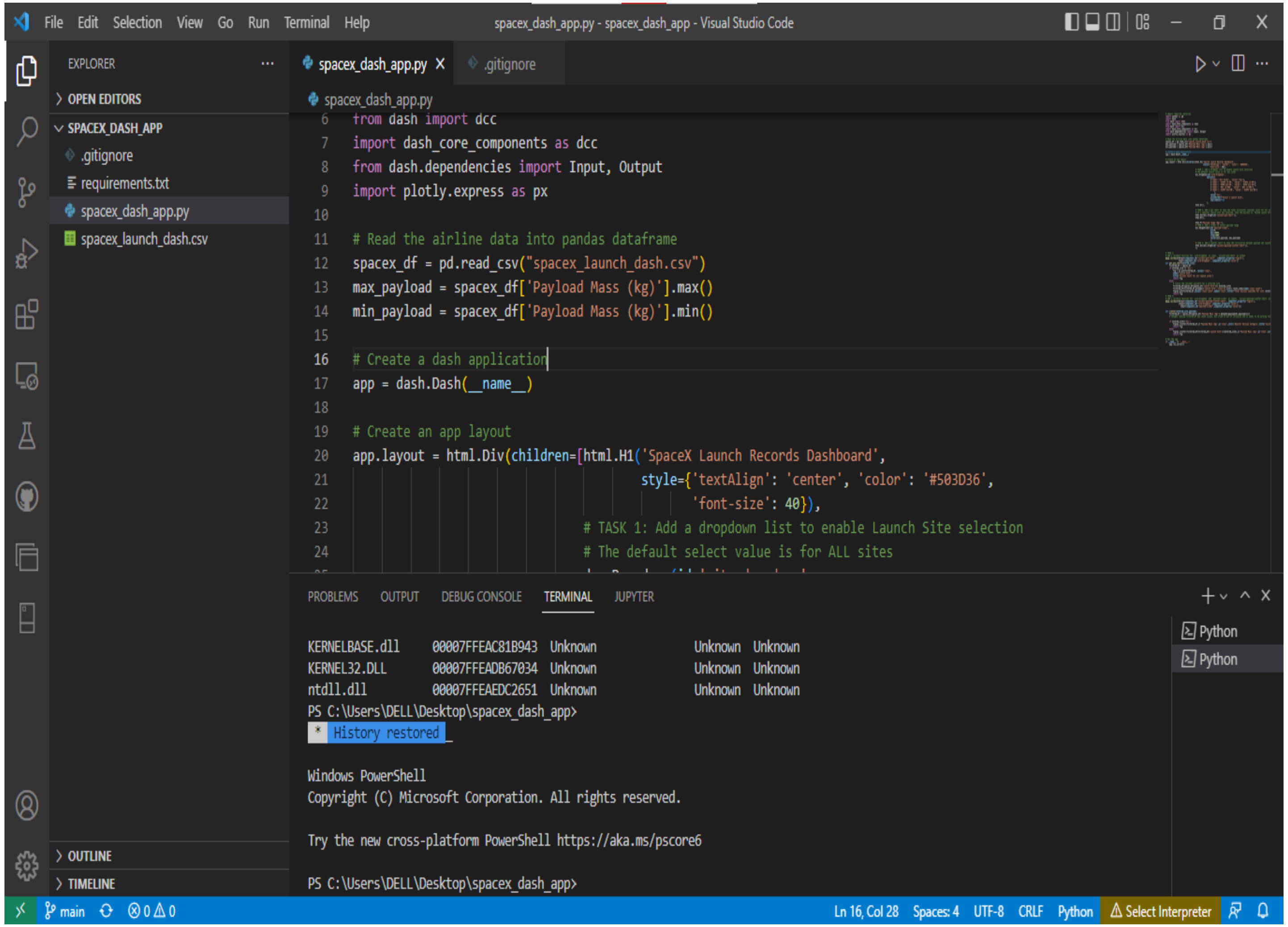
Success count on Payload mass for all sites

The live dashboard is deployed and hosted on PythonAnywhere using Flask and Dash.

Furthermore, as the dashboard was developed in skill labs virtual environment. Anaconda and VS code is used for deployment.

SpaceX Dashboard

Microsoft VS Code Tool



```
from dash import dcc
import dash_core_components as dcc
from dash.dependencies import Input, Output
import plotly.express as px

# Read the airline data into pandas dataframe
spacex_df = pd.read_csv("spacex_launch_dash.csv")
max_payload = spacex_df['Payload Mass (kg)'].max()
min_payload = spacex_df['Payload Mass (kg)'].min()

# Create a dash application
app = dash.Dash(__name__)

# Create an app layout
app.layout = html.Div(children=[html.H1('SpaceX Launch Records Dashboard',
                                        style={'textAlign': 'center', 'color': '#503D36',
                                               'font-size': 40}),
                                # TASK 1: Add a dropdown list to enable Launch Site selection
                                # The default select value is for ALL sites
```

```
KERNELBASE.dll     00007FFEAC81B943  Unknown               Unknown  Unknown
KERNEL32.DLL       00007FFEADB67034  Unknown               Unknown  Unknown
ntdll.dll          00007FFEAEDC2651  Unknown               Unknown  Unknown
PS C:\Users\DELL\Desktop\spacex_dash_app>
 *  History restored 

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\DELL\Desktop\spacex_dash_app>
```

Microsoft Visual Studio Code is used with Anaconda
virtual environment for creating Dashboard.

Python File Link

THANK YOU