IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Nikolai Duncan
October 4th, 2024

Outline

Executive Summary

The intended goal of this research project was to analyze SpaceX Falcon 9 data collected through various sources such as web scraping. After this, Machine Learning models were tested and trained to predict the first-stage landing's success rate, allowing other space agencies to decide on bidding against SpaceX.

- Summary of methodologies

The concepts and methodologies listed below were used to collect and analyze data, model and evaluate machine learning algorithms, and make calculated predictions:

- Data collection through API calls and Web Scraping
- Data transformation via Data Wrangling
- Perform Exploratory Data Analysis (EDA) with SQL and create Data Visualizations
- Determine launch site proximity through an interactive map using Folium
- Build, test, and evaluate a predictive Machine-Learning model to forecast Flacon 9 first-stage landing success rates

- Summary of all results

The results will be summarized in varying forms:

- Data Analysis Results
- Data Visualizations
- Interactive Web-Based Dashboards
- Predictive Analysis Results

Introduction

With recent successes in private space travel, the space industry is becoming a main attraction geared towards providing accessibility to the general population. However, the cost of launching rockets remains the main barrier for aspiring competitors to enter the space race.

SpaceX has proven with its first-stage reuse functionality, that they are ahead of its competitors. With the added benefit of the first-stage reuse, SpaceX although spending 62 Million per launch, has saved significantly as other companies have spent upwards of 165 Million per launch.

Throughout the document, several questions will be answered:

- Can we determine if the first stage of SpaceX Falcon 9 launches will land successfully?

- What impact do parameters such as launch site, payload mass, booster version, etc. have on the landing outcome?

- What are the possible correlations between the launch site and its success rates?

Section 1

Methodology

Methodology

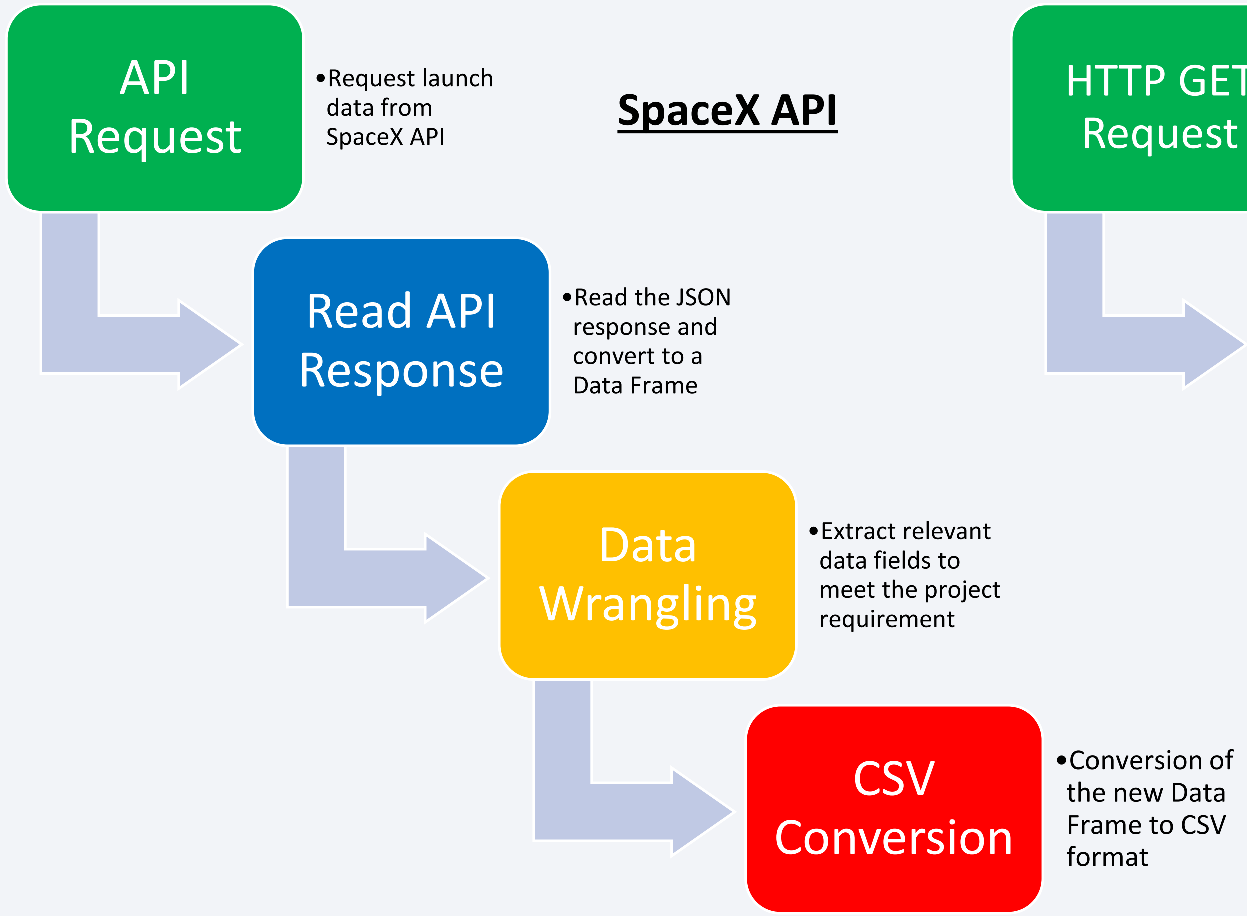
Executive Summary

- Data collection methodology:
  - Via the SpaceX API
  - By Web Scraping the Falcon 9 and Falcon Heavy Launch records from Wikipedia
- Perform data wrangling
  - Labels were determined for training the supervised models by converting landing outcomes using training labels such as 0 – unsuccessful, and 1- successful.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - A class column was created, data was transformed and normalized, test/train data was split, and the best classification algorithm was determined using the test data.

Data Collection

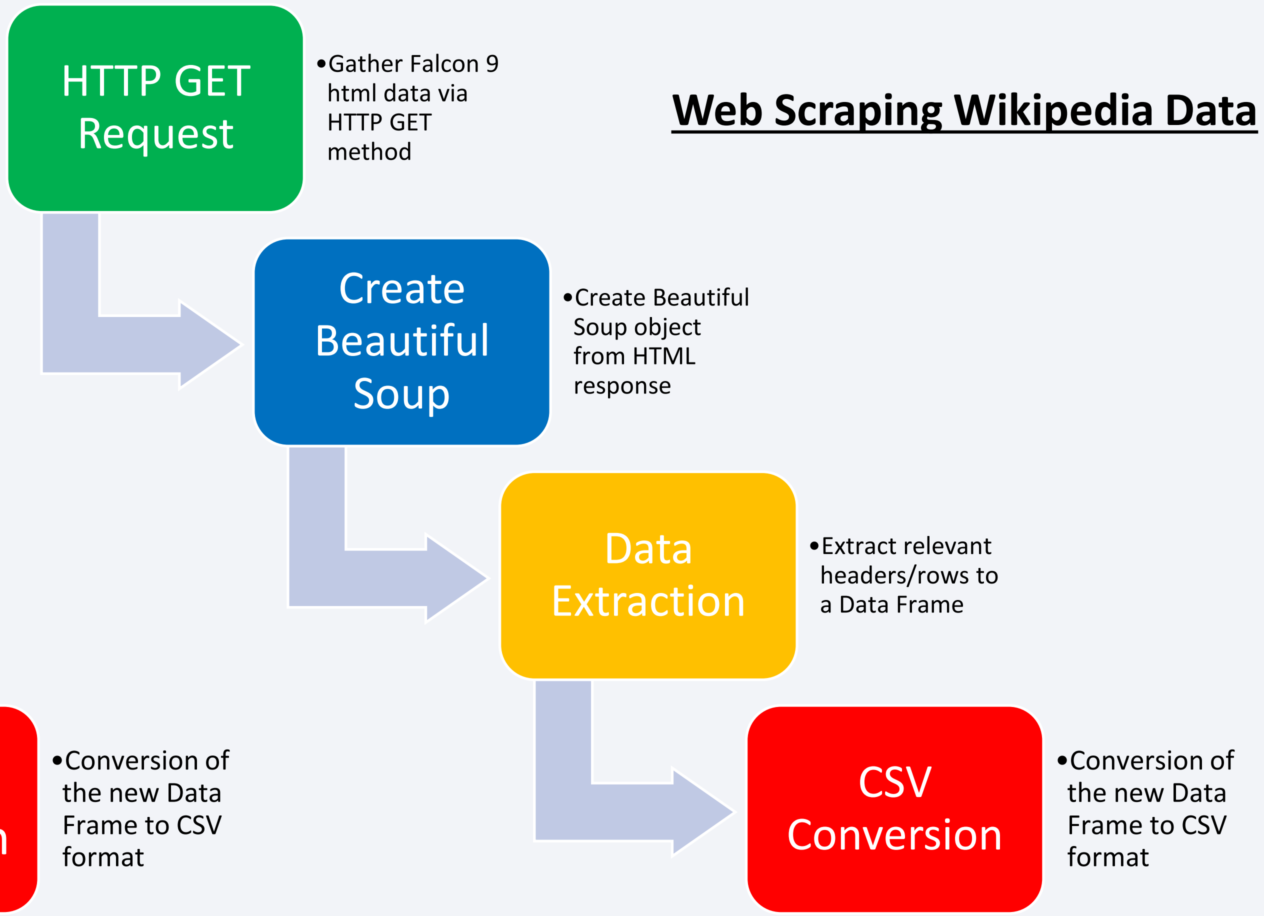
Data collection is the systematic process of gathering observations or measurements. This process is crucial for research, business operations, and decision-making across various fields. Data collection was performed through the SpaceX API and Web Scraping Wikipedia data.

SpaceX API



- Request launch data from SpaceX API
- Read the JSON response and convert to a Data Frame
- Extract relevant data fields to meet the project requirement
- Conversion of the new Data Frame to CSV format

Web Scraping Wikipedia Data



- Gather Falcon 9 html data via HTTP GET method
- Create Beautiful Soup object from HTML response
- Extract relevant headers/rows to a Data Frame
- Conversion of the new Data Frame to CSV format

Data Collection – SpaceX API

GitHub Link

1. API Request and read response into Data Frame
2. Declare Global Variables
3. Call helper functions with API calls to populate Global Variables
4. Construct data using Dictionary
5. Convert Dict to Data Frame, filter for Falcon 9 launches, convert to CSV

1. Create API GET method to request Falcon 9 Launch HTML page

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

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

2. Declare global variables list to store data returned by helper functions with additional API calls for relevant data

```
#Global variables
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

3. Call helper functions to retrieve data where columns have IDs

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

4. Construct a dataset from the received data and combine columns into a dictionary

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

5. Create a Data Frame from a dictionary and filter to keep only the Falcon 9 launches

```
# Create a data from launch_dict
df_launch = pd.DataFrame(launch_dict)

# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 =
df_launch.loc[df_launch['BoosterVersion']!='Falcon 1’]

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

Data Collection - Scraping

[GitHub Link]

1. Perform HTTP GET to request HTML page
2. Create Beautiful Soup Object
3. Extraction of column names from HTML table headers
4. Create a Dictionary with keys from extracted column names
5. Call helper functions to fill up the dict with launch records
6. Convert the Dictionary to a Data Frame

1. Create API GET method to request Falcon 9 launch HTML page

```
static_url = https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922

# use requests.get() method with the provided static_url
# assign the response to an object
html_data = requests.get(static_url).text
```

2. Create a Beautiful Soup Object

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(html_data,"html.parser")
```

3. Find all the tables from the Wikipedia page and extract the required column names from the HTML table headers

```
# 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')

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
colnames = soup.find_all('th')
for x in range (len(colnames)):
    name2 = extract_column_from_header(colnames[x])
    if (name2 is not None and len(name2) > 3):
        column_names.append(name2)
```

4. Create an empty Dictionary with keys from extracted column names

```
launch_dict= dict.fromkeys(column_names)

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

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

5. Fill up the launch_dict with launch records extracted from table rows
- helper functions to process web-scraped HTML table data

```
def date_time(table_cells)
def booster_version(table_cells)
def landing_status(table_cells)
def get_mass(table_cells)
def extract_column_from_header(row)
```

6. Convert launch_dict to a Data Frame

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

Data Wrangling

GitHub Link

- To determine the patterns in the data and define labels for supervised training models, Exploratory Data Analysis (EDA) was conducted.
- The dataset comprised varying missions' outcomes converted into Training Labels where 1 indicated a successfully landed booster and 0 a failed booster landing.
- The following landing scenarios were considered when creating the labels:
  - ❖ True Ocean – the mission outcome successfully landed in a specific region of the ocean
  - ❖ False Ocean – the mission outcome unsuccessfully landed in a specific region of the ocean
  - ❖ RTLS – the mission outcome successfully landed on a ground pad
  - ❖ False RTLS – the mission outcome unsuccessfully landed on a ground pad
  - ❖ True ASDS – the mission outcome successfully landed on a drone ship
  - ❖ False ASDS – the mission outcome unsuccessfully landed on a drone ship

Data Wrangling – cont'd

GitHub Link

1. Load dataset into the Data Frame

1. Load SpaceX dataset (CSV file) into a Data Frame

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

2. Establish data patterns

- Calculate the number of launches at each site

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

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

2. Establish data patterns

- Calculate the number and occurrence of each Orbit

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

```
Orbit
GTO     27
ISS     21
VLEO    14
PO       9
LEO      7
SSO      5
MEO      3
HEO      1
ES-L1    1
SO       1
GEO      1
```

- Calculate the number and occurrence of mission outcomes per Orbit Type

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

3. Create a landing outcome label

3. Create a landing outcome label from the Outcome column in the Data Frame

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for i in df['Outcome']:
    if i in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)

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

```
     Class
0    0
1    0
2    0
3    0
4    0
5    0
6    1
7    1
```

EDA with Data Visualization

GitHub Link

The following charts were plotted to gain further insights about the dataset, during the Exploratory Data Analysis (EDA) phase:

Scatter Plot

Identifies the relationship or correlation between two variables making patterns easy to observe. The following charts were visualized:

- Flight Number and Launch Site relationships
- Payload and Launch Site relationships
- Flight number and Orbit Type relationships
- Payload and Orbit Type relationships

Bar Chart

A bar chart is commonly used to compare variable values at a given point and makes it easy to see which groups are common, and how other groups compare against each other. The length of each bar is proportional to the value of the item it represents. The following charts were visualized:

- Success Rate relationships between each Orbit Type

Line Chart

Line charts are commonly used to track changes over a period and help to depict trends. The following charts were visualized:

- Average yearly trend for launch successes

EDA with SQL

GitHub Link

The following SQL queries were completed to further the understanding of the SpaceX Dataset:

- ✓ Display the names of the unique launch sites for the space mission
- ✓ Display five (5) records where launch sites begin with the string “CCA”
- ✓ Display the total payload mass carried by boosters launched by NASA (CRS)
- ✓ Display the average payload mass carried by booster version F9 v1.1
- ✓ List the date when the first successful landing outcome on a ground pad was achieved
- ✓ List the names of the boosters which have had success in drone ships and have a payload mass greater than 4000 but less than 6000
- ✓ List the total number of successful and failed mission outcomes
- ✓ List the names of the booster versions which have carried the maximum payload mass using a subquery
- ✓ List the failed landing outcomes in drone ships, their booster versions, and the launch site names for the year 2015
- ✓ Rank the count of landing outcomes between the date 2010-06-04 and 2017-03-20, in descending order

Build an Interactive Map with Folium

GitHub Link

Folium interactive map helps to analyze geospatial data to perform more interactive visual analytics and better understand factors such as location and proximity of launch sites that impact launch success rate.

The following map objects were created and added to the map:

- Mark all launch sites on the map. This allowed us to visualize launch sites on the map
- Added ‘folium.circle’ and ‘folium.marker’ to highlight the circle area with a text label over each launch site
- Added a ‘MarkerCluster()’ to show launch success (green) and failure (red) markers for each launch site
- Calculated distances between a launch site to its proximities (e.g., coastline, railroad, highway, city)
- Added ‘MousePosition() to get coordinate for a mouse position over a point on the map
- Added ‘folium.Marker()’ to display distance (in KM) on the point on the map (e.g., coastline, railroad, highway, city)
- Added ‘folium.Polyline()’ to draw a line between the point on the map and the launch site
- Repeated steps above to add markers and draw lines between launch sites and proximities – coastline, railroad, highway, city)

Building the Interactive Map with Folium helped answer the following questions:

- Are launch sites in close proximity to railways? YES
- Are launch sites in close proximity to highways? YES
- Are launch sites in close proximity to coastline? YES
- Do launch sites keep a certain distance away from cities? YES

Build a Dashboard with Plotly Dash

GitHub Link

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 the ability to filter Dashboard visuals 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 select different payload ranges to identify visual patterns

Added a Scatter chart to observe how payload may be correlated with mission outcomes for the selected site(s). The color-label Booster version on each scatter point provided mission outcomes with different boosters

Dashboard helped answer the 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)

GitHub Link

1. Read the dataset into the Data Frame and create a Class array
2. Standardize and transform the data
3. Train/Test/Split the data into training and test data sets
4. Create and Refine Data Models
5. Find the best performing Model

1. Load the SpaceX dataset into a Data Frame and create a NumPy array from the column class in the data

```
from js import fetch
import io

URL1 = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_2.csv"
resp1 = await fetch(URL1)
text1 = io.BytesIO((await resp1.arrayBuffer()).to_py())
data = pd.read_csv(text1)

URL2 = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_3.csv'
resp2 = await fetch(URL2)
text2 = io.BytesIO((await resp2.arrayBuffer()).to_py())
X = pd.read_csv(text2)

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

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

```
# Students get this
X= preprocessing.StandardScaler().fit(X).transform(X)
```

3. Train/test/split X and Y into 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 the Models based on the following Logistic Regression classification algorithm

- Create a Logistic Regression object, then create a GridSearchCV object

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

- Fit train data set into the GridSearchCV object and train the Model

```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
lr = LogisticRegression()
logreg_cv = GridSearchCV(lr, parameters,cv=10)
logreg_cv.fit(X_train, Y_train)
```

- Find and display the best hyperparameters and accuracy score

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

- Check the accuracy of the test data by creating a confusion matrix

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

- Repeat the steps above for the Decision Tree, KNN, and SVM algorithms

5. Find the best-performing Model

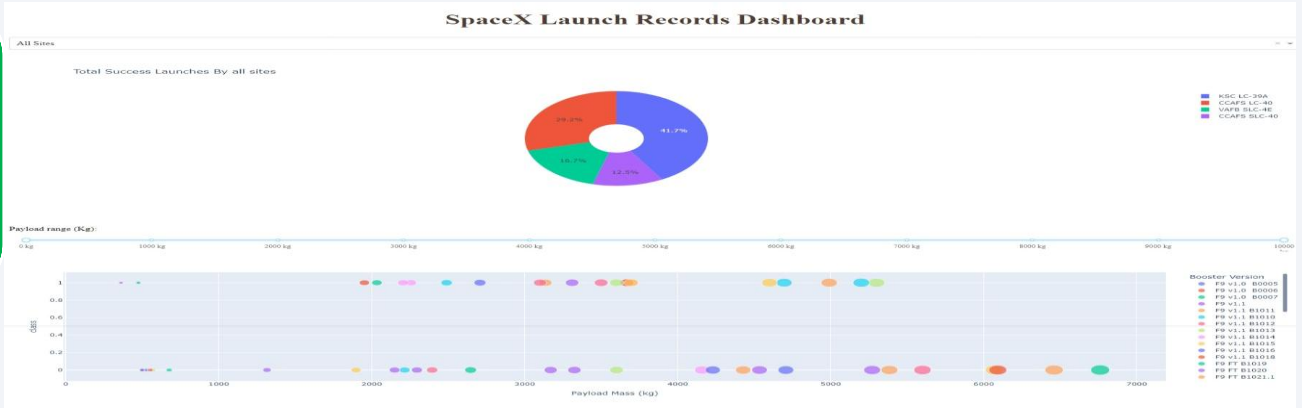
```
# Create a DF for algorithm type and respective best scores
Model_Performance_df = pd.DataFrame({'Algo Type': ['Logistic Regression', 'SVM','Decision Tree','KNN'],
'Accuracy Score': [logreg_cv.best_score_, svm_cv.best_score_, tree_cv.best_score_, knn_cv.best_score_],
'Test Data Accuracy Score': [logreg_cv.score(X_test, 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
```

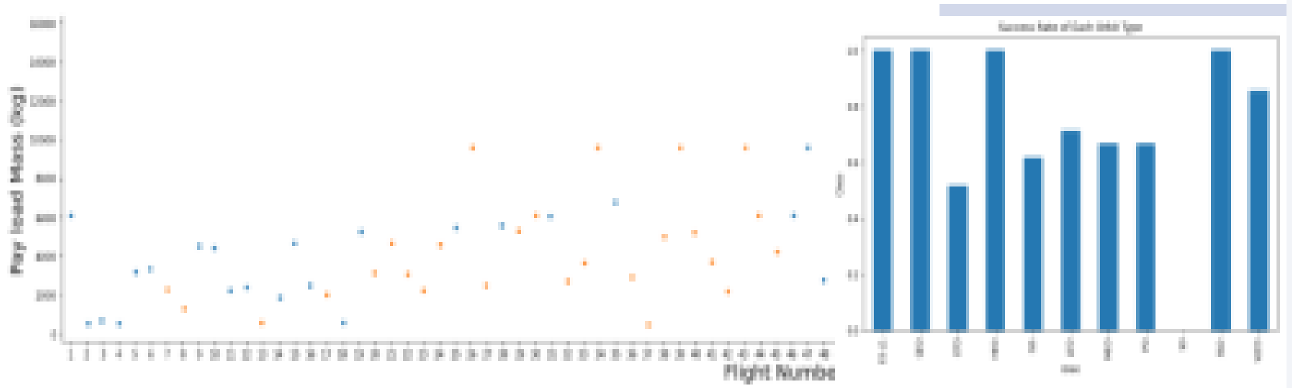
Results

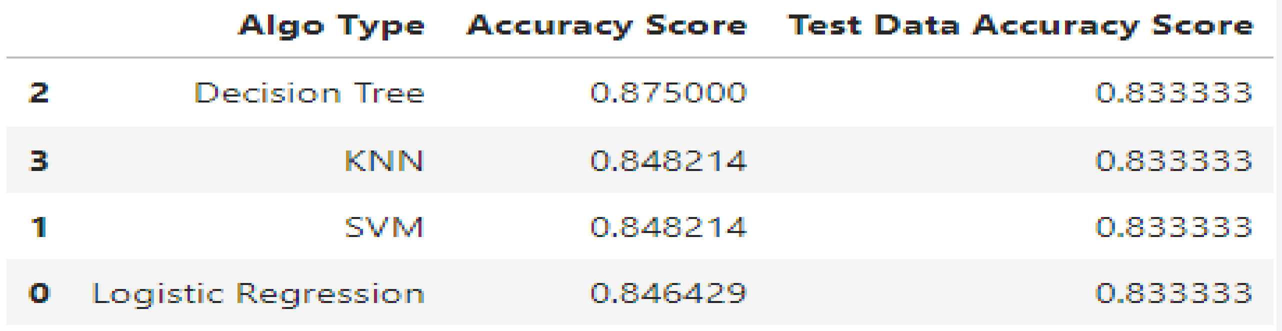
Interactive Dashboard

Exploratory Data Analysis (EDA)

Predictive Analysis Results





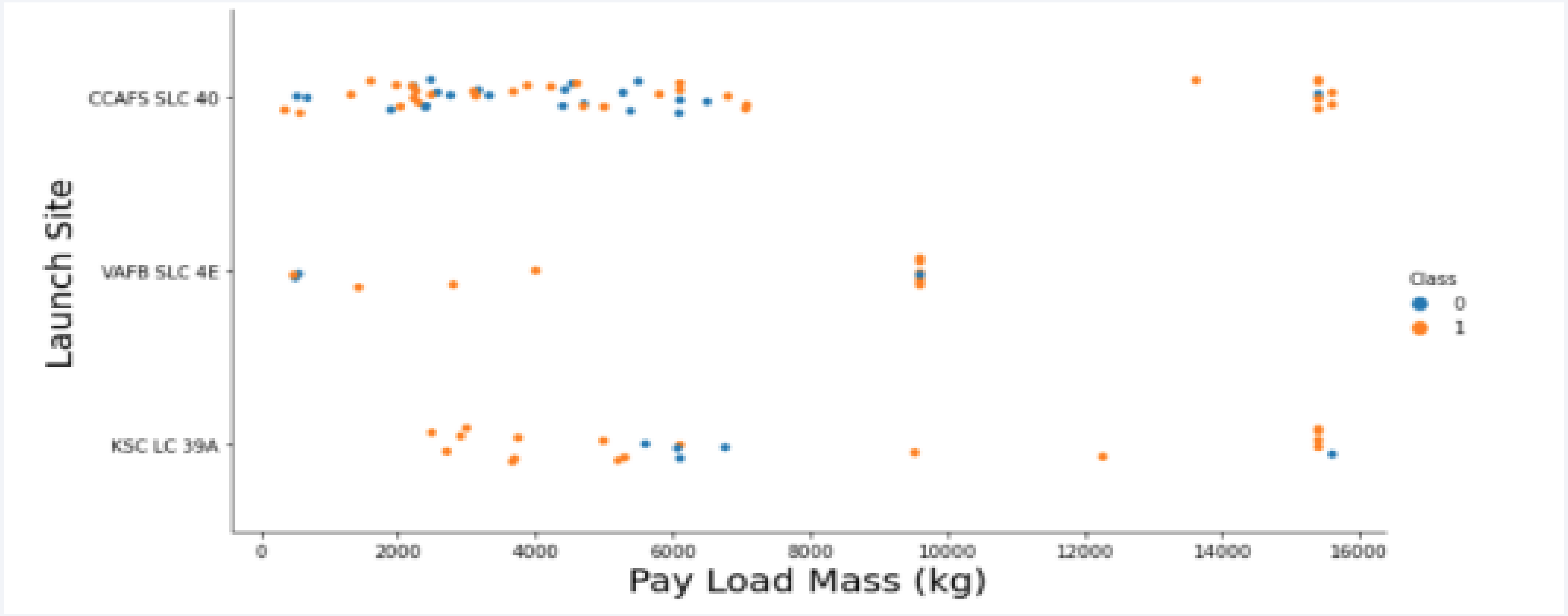


	Algo Type	Accuracy Score	Test Data Accuracy Score
2	Decision Tree	0.875000	0.833333
3	KNN	0.848214	0.833333
1	SVM	0.848214	0.833333
0	Logistic Regression	0.846429	0.833333

Section 2

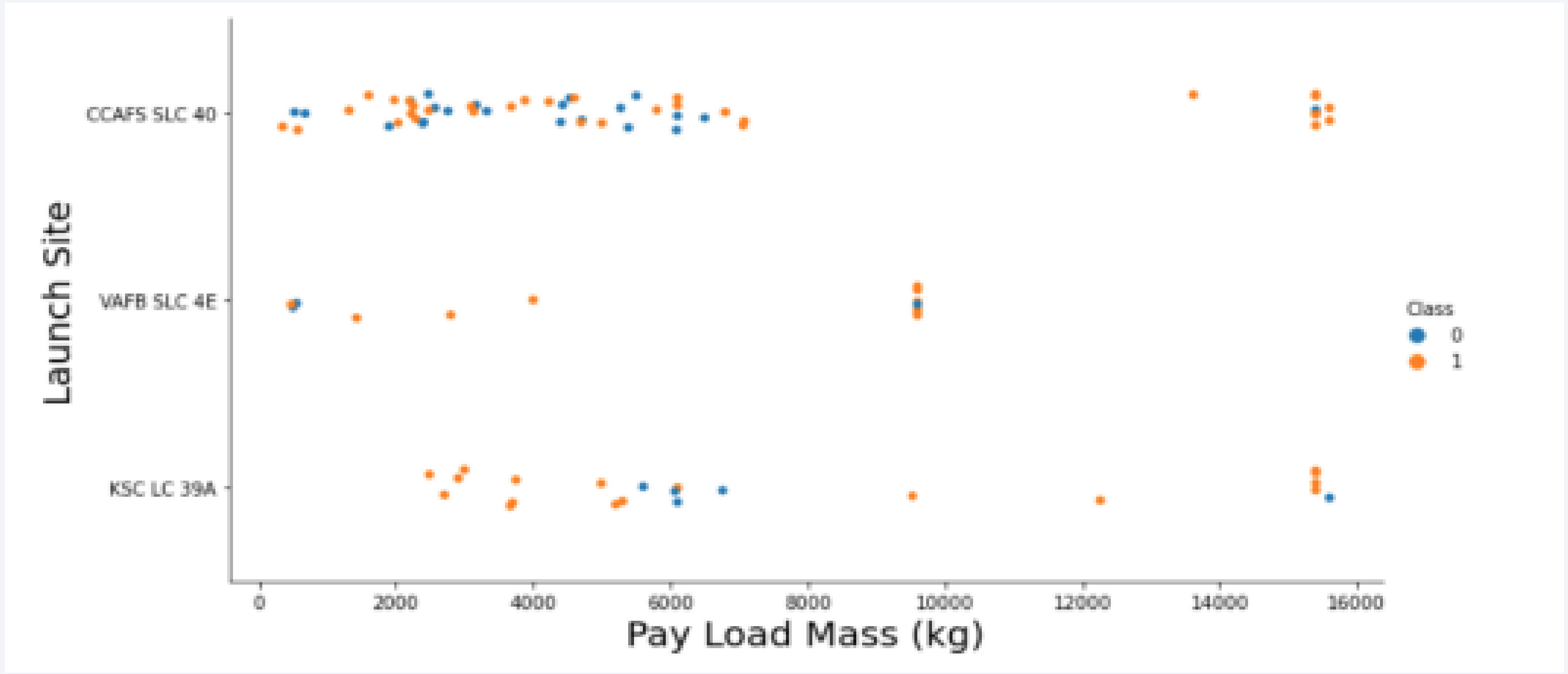
Insights drawn from EDA

Flight Number vs. Launch Site



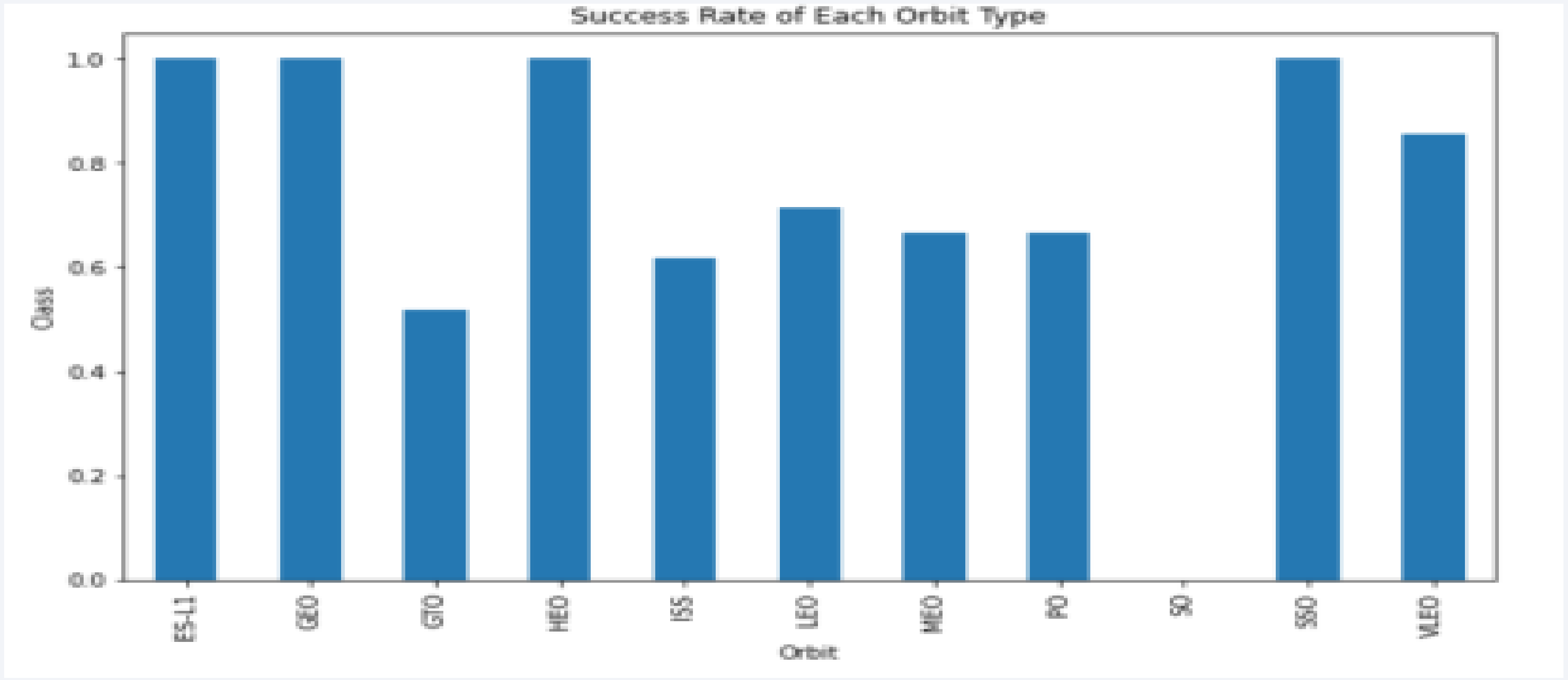
- Success rates (Class=1) increase as the number of flights increase
- For launch site 'KSC LC 39A', it takes at least around 25 launches before its first successful launch

Payload vs. Launch Site



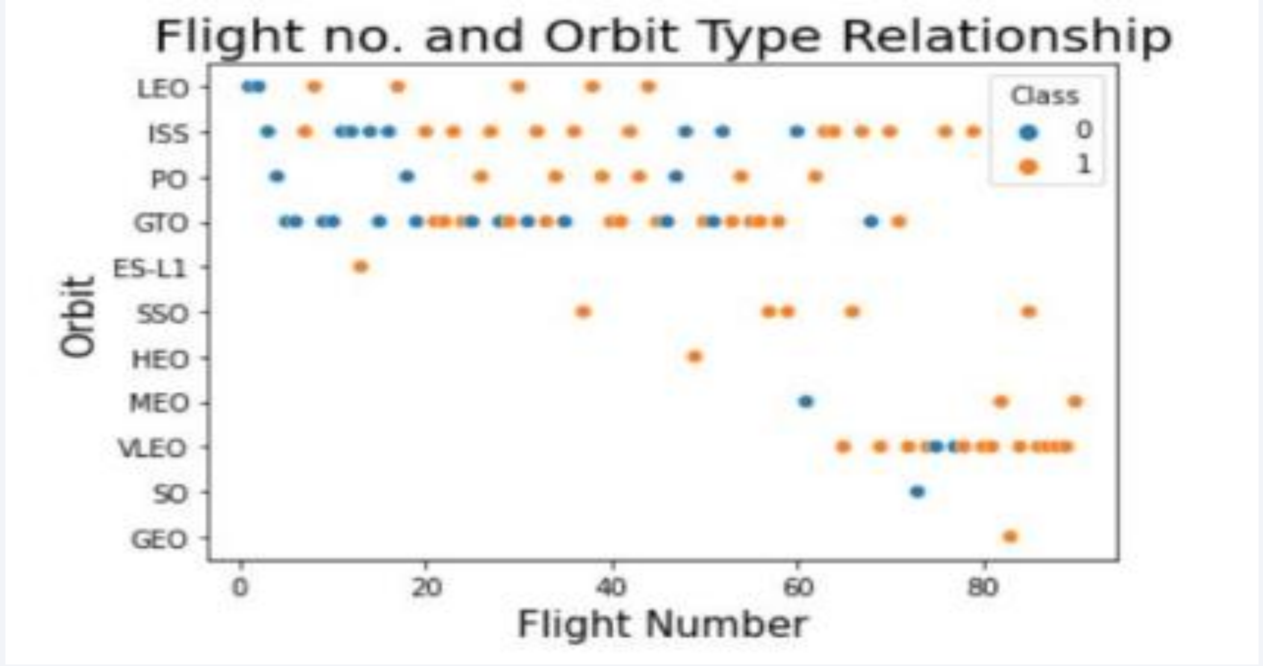
- For launch site ‘VAFB SLC 4E’, there are no rockets launched for payload greater than 10,000 kg
- Percentage of successful launch (Class=1) increases for launch site ‘VAFB SLC 4E’ as the payload mass increases
- There is no clear correlation or pattern between launch site and payload mass

Success Rate vs. Orbit Type



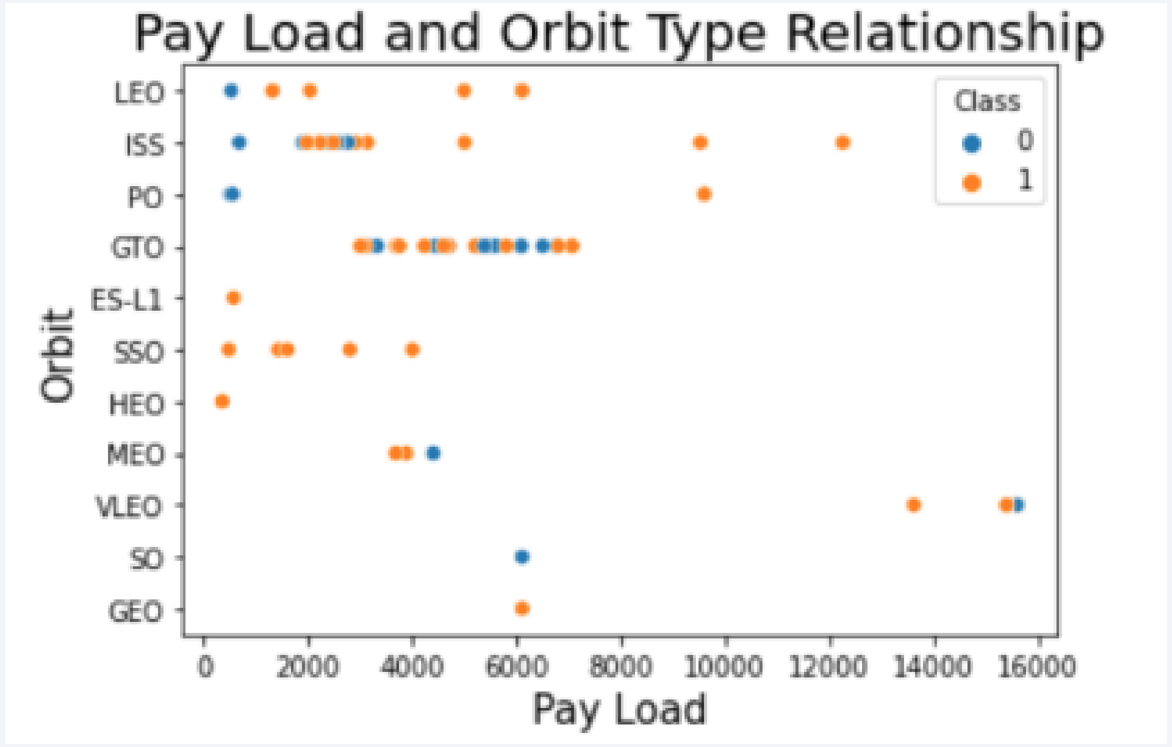
- Orbits ES-LI, GEO, HEO, and SSO have the highest success rates
- GTO orbit has the lowest success rate

Flight Number vs. Orbit Type



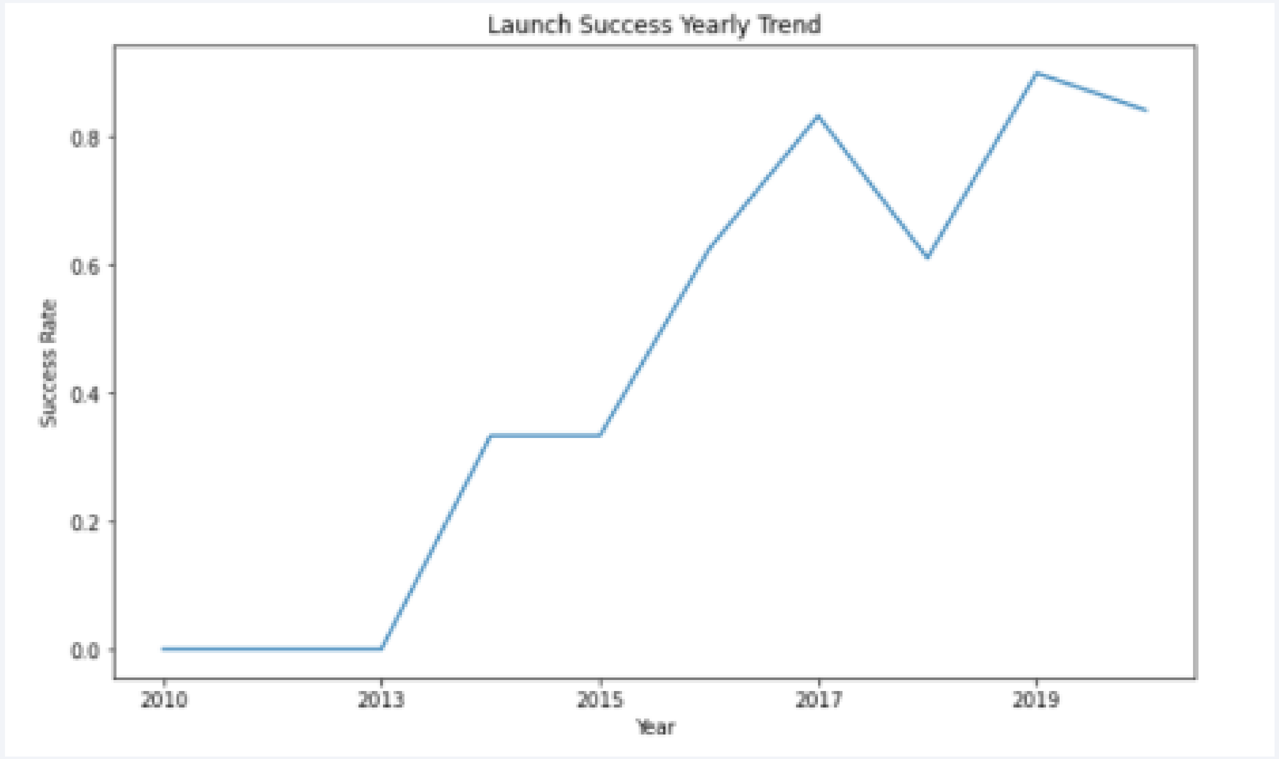
- For orbit VLEO, the first successful landing (class=1) doesn’t occur until 60+ number of flights
- For most orbits (LEO, ISS, PO, SSO, MEO, VLEO) successful landing rates appear to increase with flight numbers
- There is no relationship between flight number and orbit for GTO

Payload vs. Orbit Type



- Successful landing rates (Class=1) appear to increase with payload for orbits LEO, ISS, PO, and SSO
- For GEO orbit, there is no clear pattern between payload and orbit for successful or unsuccessful landing

Launch Success Yearly Trend



- Success rate (Class=1) increased by about 80% between 2013 and 2020
- Success rates remained the same between 2010 and 2013 and between 2014 and 2015
- Success rates decreased between 2017 and 2018 and between 2019 and 2020

All Launch Site Names

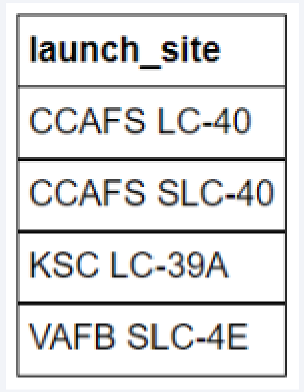
- ***Query:***

```
%%sql
select distinct Launch_Site from spacextbl;
```

- ***Description:***

"Distinct" returns unique values from the queried column, Launch_Site yielding only four (4) results.

- ***Result:***



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

Launch Site Names Begin with 'CCA'

- ***Query:***

%%sql

select * from spacextbl where Launch_Site LIKE 'CCA%' limit 5;

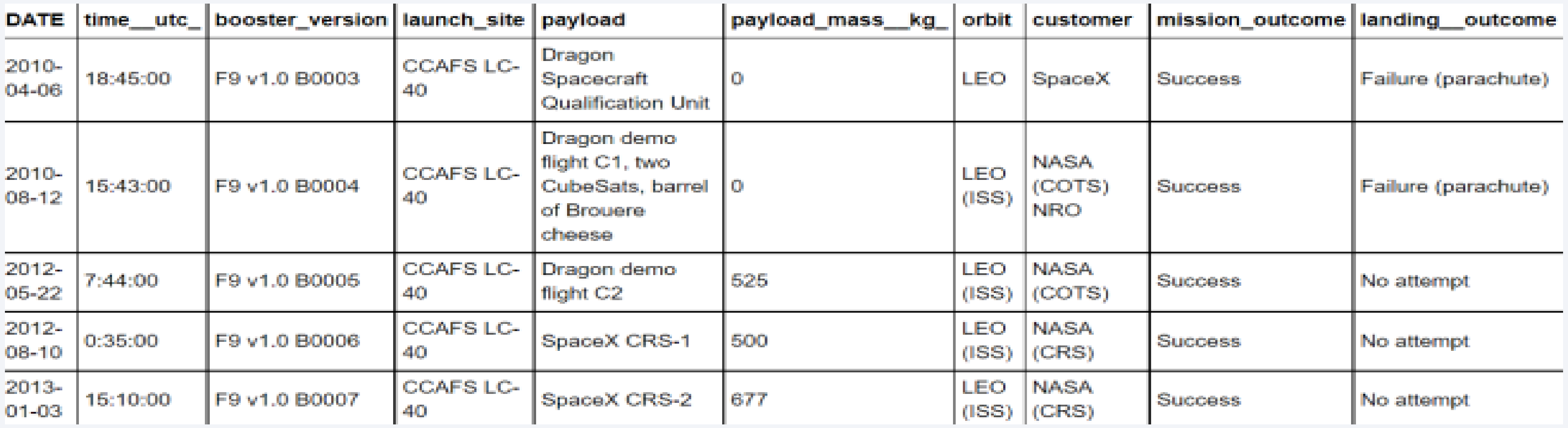
- ***Description:***

Using the keyword ***'Like'*** and format ***'CCA%'***, returns records where the ***'Launch_Site'*** column

starts with ***"CCA"***.

Limit 5, limits the number of returned records to five (5)

- ***Result:***



DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-08-10	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- ***Query:***

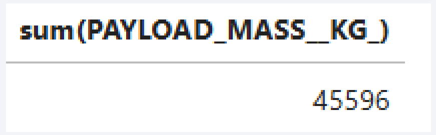
%%sql

select sum(PAYLOAD_MASS__KG_) from spacextbl where Customer = 'NASA (CRS)';

- ***Description:***

'sum' adds column ***'PAYLOAD_MASS_KG'*** and returns the total payload mass for customers named ***'NASA (CRS)'***

- ***Result:***



sum(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

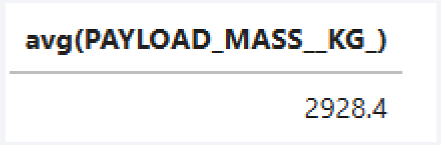
- ***Query:***

```
%%sql

select avg(PAYLOAD_MASS__KG_) from spacextbl where Booster_Version LIKE 'F9 v1.1';
```

- ***Description:***

The *'avg'* keyword returns the average payload mass in the *'PAYLOAD_MASS_KG'* column where the booster version is *'F9 v1.1'*

- ***Result:***



avg(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

- ***Query:***

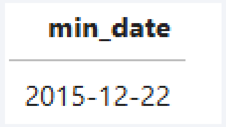
```
%%sql

select min(Date) as min_date from spacextbl where Landing_Outcome = 'Success (ground pad)';
```

- ***Description:***

'min(Date)' selects the first or the oldest date from the ***'Date'*** column where the first successful landing on the group pad was achieved

The Where clause defines the criteria to return the date for scenarios where ***'Landing_Outcome'*** value is equal to ***'Success (ground pad)'***

- ***Result:***



min_date
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- ***Query:***

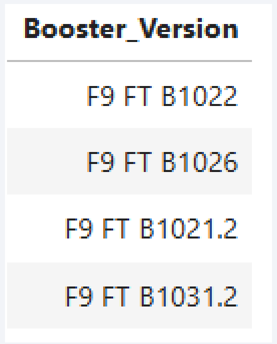
```
%%sql

select Booster_Version from spacextbl where (PAYLOAD_MASS__KG_> 4000 and PAYLOAD_MASS__KG_ < 6000)

and (Landing_Outcome = 'Success (drone ship)’);
```

- ***Description:***

The query finds the booster version where payload mass is greater than 4000 but less than 6000 and the landing outcome is success in the drone ship

The *‘and’* operator in the where clause returns booster versions where both conditions in the where clause are true

- ***Result:***



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

Total Number of Successful and Failure Mission Outcomes

- ***Query:***

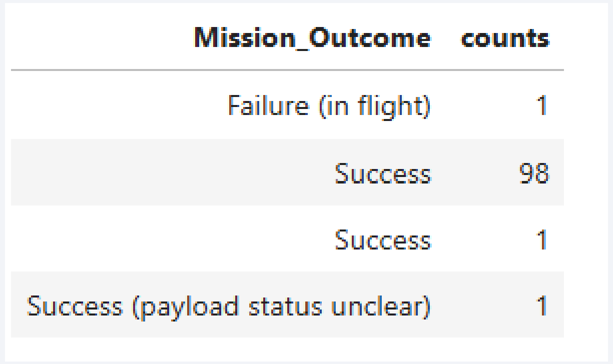
```
%%sql
select Mission_Outcome, count(Mission_Outcome) as counts from spacextbl group by Mission_Outcome;
```

- ***Description:***

The ***'group by'*** keyword arranges identical data in a column into a group

In this case, the number of mission outcomes by types of outcomes are grouped in column 'counts'

- ***Result:***



Mission_Outcome	counts
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Boosters Carried Maximum Payload

- ***Query:***

%%sql

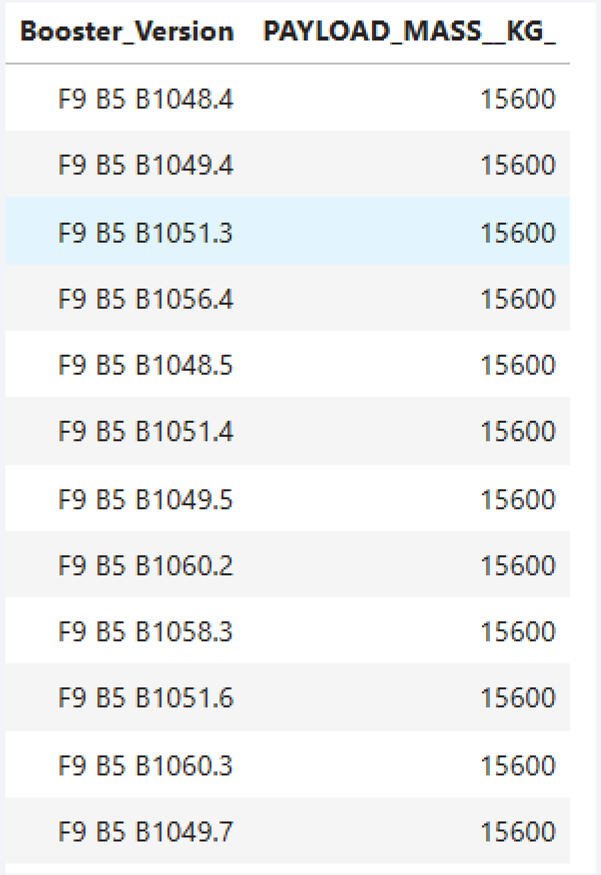
select Booster_Version, PAYLOAD_MASS__KG_ from spacextbl where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from spacextbl);

- ***Description:***

The sub-query returns the maximum payload mass by using the keyword ***'max'*** on the payload mass column

The main query returns booster versions and respective payload mass where payload mass is the maximum with value of 15600

- ***Result:***



Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

2015 Launch Records

- ***Query:***

```
%%sql
select Landing_Outcome, Booster_Version, Launch_Site from spacextbl where Landing_Outcome = 'Failure (drone ship)' and year(Date) = '2015'
```

- ***Description:***

The query lists the landing outcome, booster version, and the launch site where the landing outcome is failed in drone ship and the year is 2015

The ***'and'*** operator in the where clause returns booster versions where both conditions in the where clause are true

The ***'year'*** keyword extracts the year from column ***'Date'***

The results identify the launch site as 'CCAFS LC-40' and the booster version as F9 v1.1 B1012 and B1015 had failed landing outcomes in drop ship in the year 2015

- ***Result:***

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

- Present your query result with a short explanation here

Section 3

Launch Sites Proximities Analysis

<Folium Map Screenshot 1>

- Replace <Folium map screenshot 1> title with an appropriate title
- Explore the generated folium map and make a proper screenshot to include all launch sites' location markers on a global map
- Explain the important elements and findings on the screenshot

<Folium Map Screenshot 2>

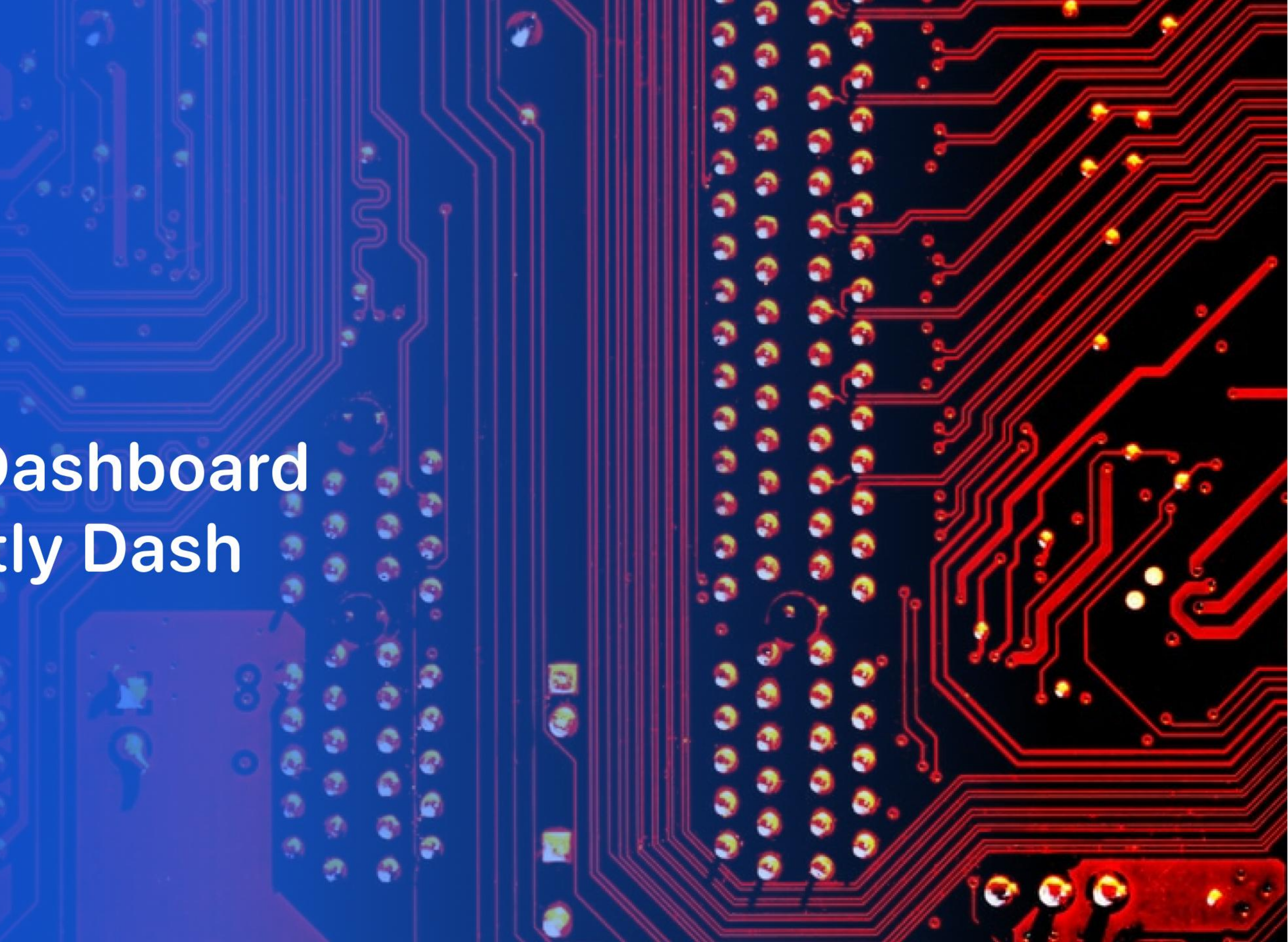
- Replace <Folium map screenshot 2> title with an appropriate title
- Explore the folium map and make a proper screenshot to show the color-labeled launch outcomes on the map
- Explain the important elements and findings on the screenshot

<Folium Map Screenshot 3>

- Replace <Folium map screenshot 3> title with an appropriate title
- Explore the generated folium map and show the screenshot of a selected launch site to its proximities such as railway, highway, coastline, with distance calculated and displayed
- Explain the important elements and findings on the screenshot

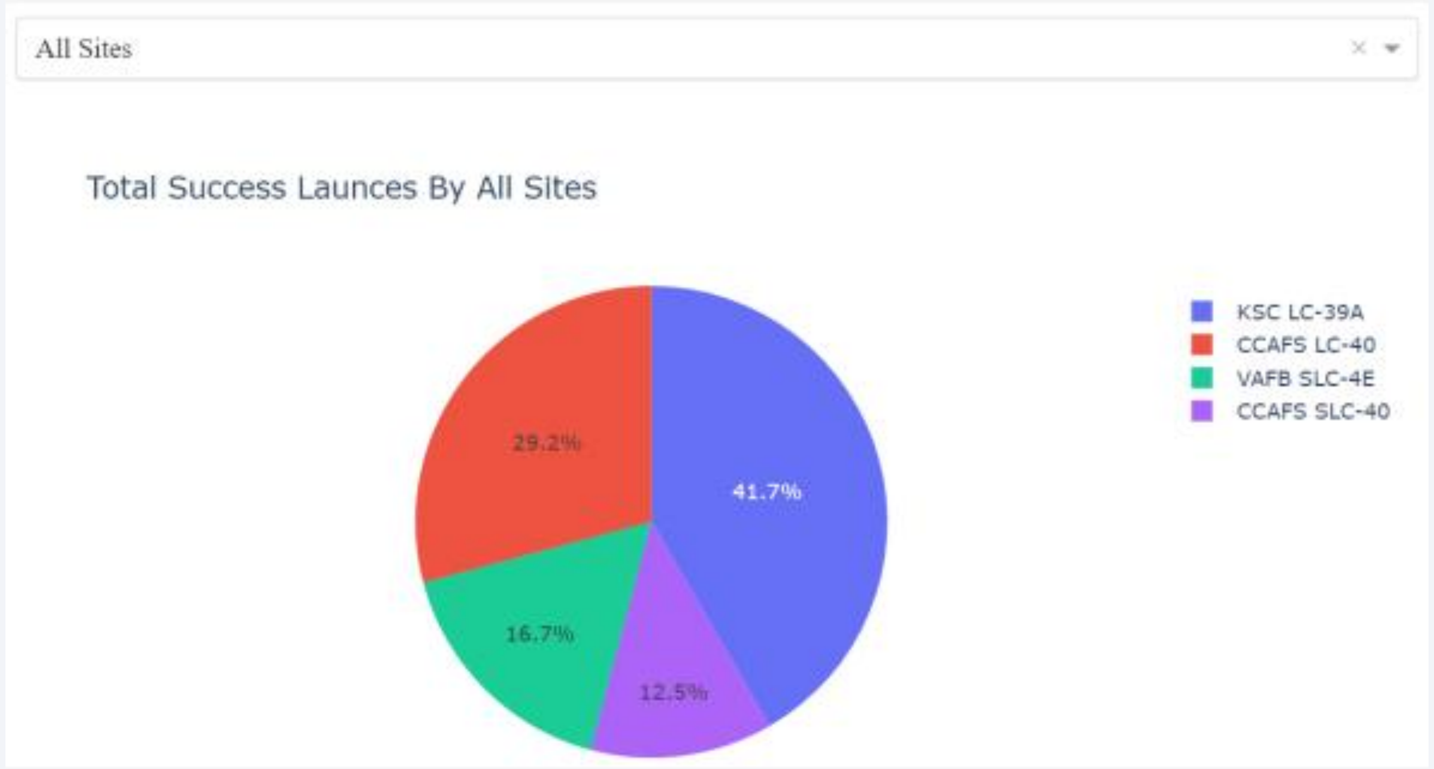
Section 4

Build a Dashboard with Plotly Dash



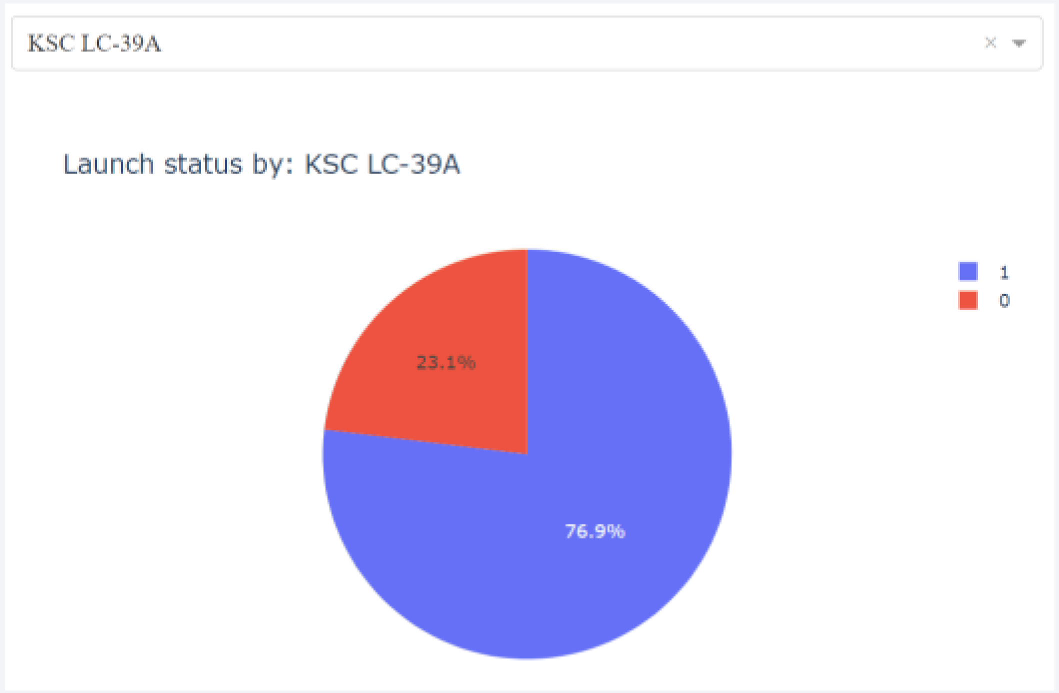
Launch Success Counts For All Sites

- Launch Site ***'KSC LC-39A'*** has the highest launch success rate
- Launch Site ***'CCAFS SLC-40'*** has the lowest launch success rate



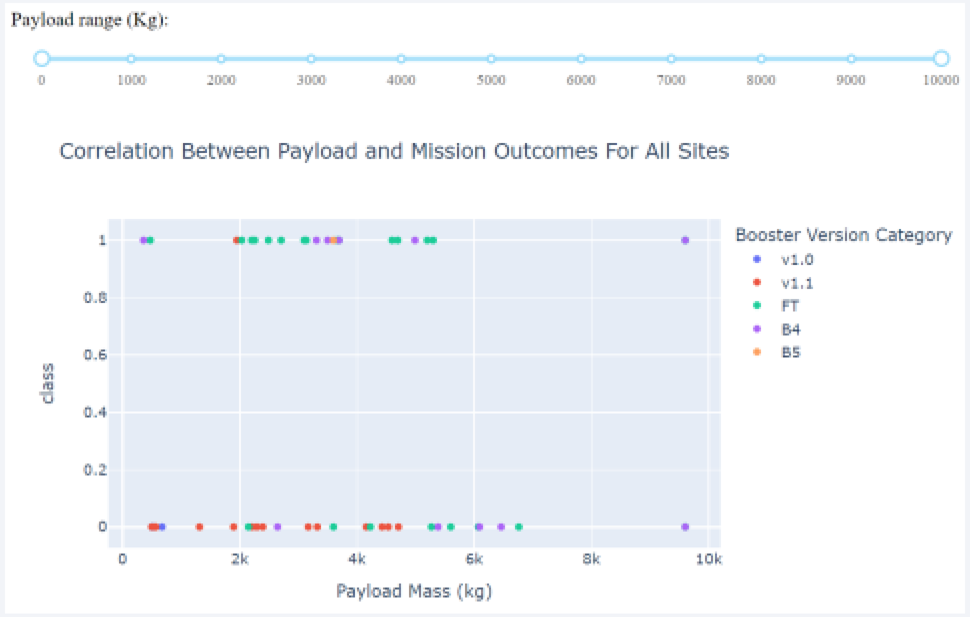
Launch Site With The Highest Launch Success Ratio

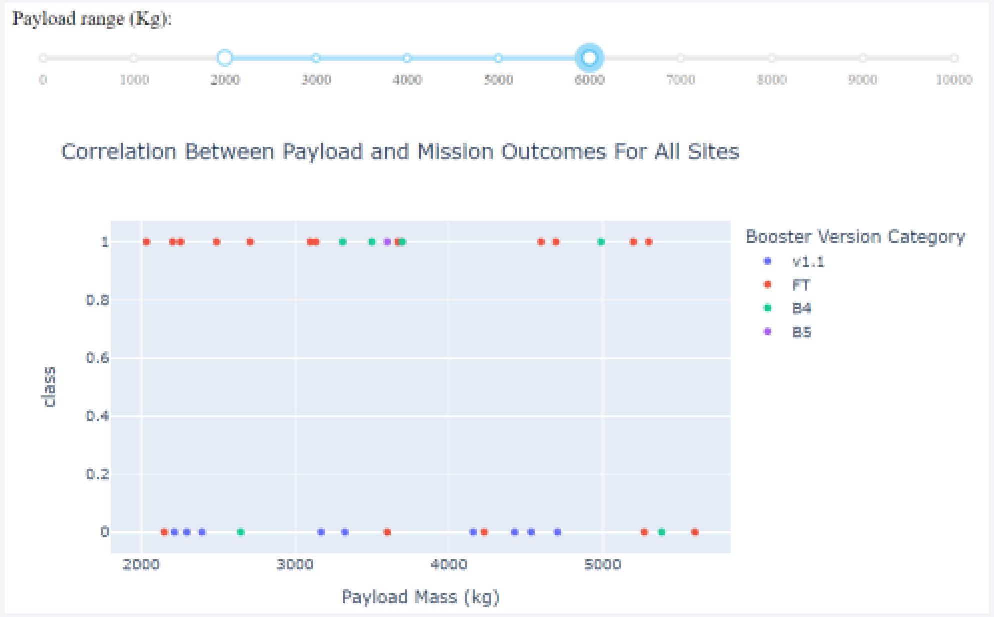
- KSC LC-39A Launch Site has the highest launch success rate and count
- Launch success rate is 76.9%
- Launch success-failure rate is 23.1%



Payload Versus Launch Outcome Scatter Plot (All Sites)

- Most successful launches are in the payload range from 2000 to about 5500
- Booster version category *'FT'* has the most successful launches
- Only booster with a successful launch when the payload is greater than 6k is *'B4'*



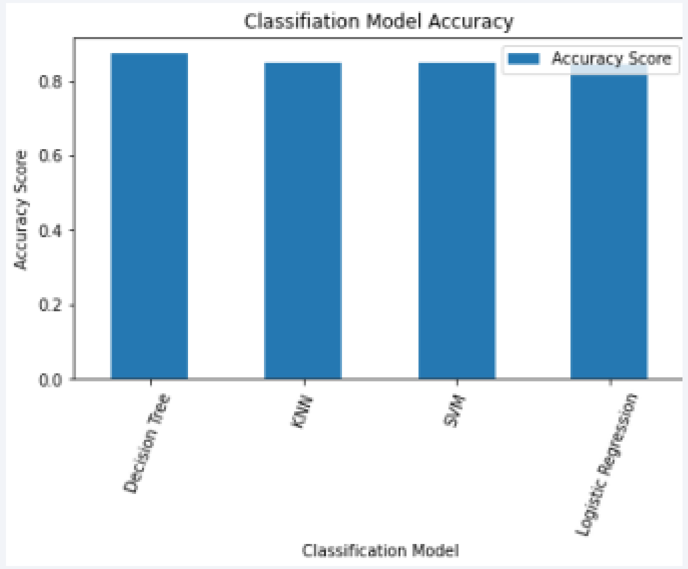


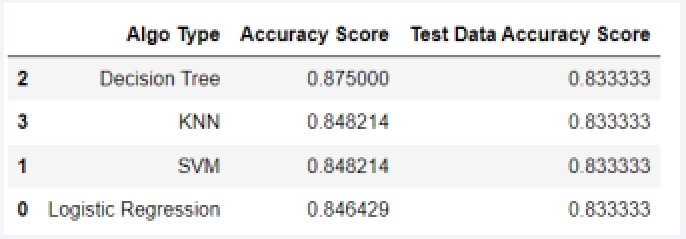
Section 5

Predictive Analysis (Classification)

Classification Accuracy

- Based on the Accuracy scores and evidence from the bar chart, the Decision Tree algorithm has the highest classification score with a value of .8750
- The Accuracy Score on the test data is the same for all the classification algorithms based on the data set with a value of .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 tune the models

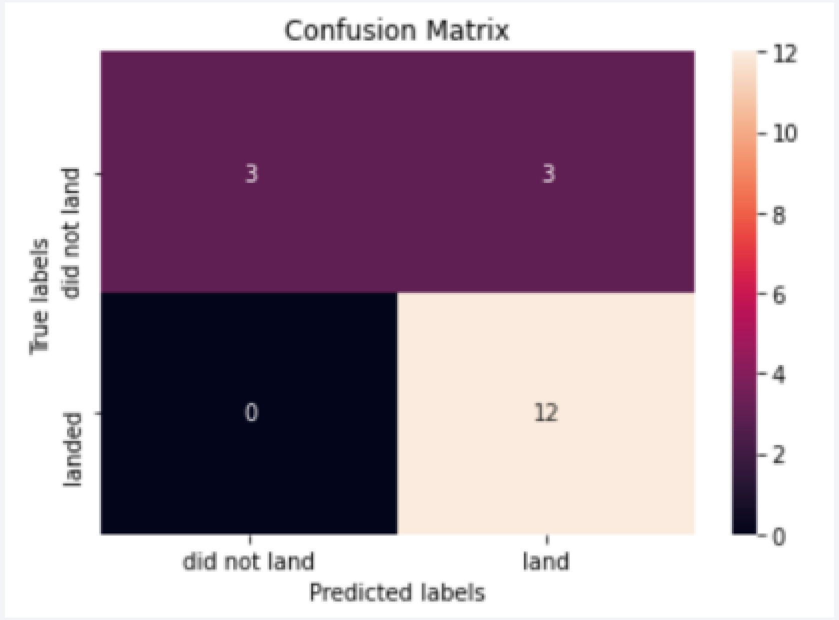




	Algo Type	Accuracy Score	Test Data Accuracy Score
2	Decision Tree	0.875000	0.833333
3	KNN	0.848214	0.833333
1	SVM	0.848214	0.833333
0	Logistic Regression	0.846429	0.833333

Confusion Matrix

- The confusion matrix is the 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%



Conclusions

- As the number of flights increases, the first stage is more likely to land successfully
- Success rates appear to go up as Payload increases but there is no clear correlation between Payload mass and success rates
- Launch success rate increased by about ***80%*** from ***2013 to 2020***
- Launch Site ***'KSC LC-39A'*** has the ***highest launch success rate*** and Launch Site ***'CCAFS SLC 40'*** has the ***lowest launch success rate***
- Orbits ***ES-L1, GEO, HEO***, and ***SSO*** have the ***highest launch success rates*** and orbit ***GTO*** has the ***lowest launch success rate***
- Launch sites are located strategically away from the cities and closer to the coastline, railroads, and highways
- The best-performing Machine Learning Classification Model is the Decision Tree with an accuracy of about 87.5%. When the models were scored on the test data, the accuracy score was about 83% for all models. More data may be needed to tune the models and find a potential better fit.

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!