IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Alamsyah Jeremy Hasudungan
March 19, 2022

Outline

Executive Summary

1. Summary of methodologies
   a) Data Collection
   b) Data Wrangling
   c) Exploratory Data Analysis
   d) Interactive Visual Analytics and Dashboard
   e) Machine Learning Prediction
2. Summary of all results
   a) Exploratory Data Analysis Results
   b) Interactive Visual Analytics and Dashboard Screenshots
   c) Machine Learning Prediction Results

Introduction – Background and Context

https://www.spacex.com/media/Capabilities&Services.pdf

1. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 67 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage.
2. SpaceX's Falcon 9 Can recover the first stage. Sometimes the first stage does not land. Sometimes it will crash. Other times, Space X will sacrifice the first stage due to the mission parameters like payload, orbit, and customer.
3. SpaceY, a new commercial rocket launch provider founded by Billionaire industrialist Allon Musk, would like to bid against SpaceX for a rocket launch.

Introduction – Objective

The objectives of this project is to predict if the first stage of SpaceX's Falcon 9 will land successfully. If we can determine if the first stage will land, we can determine the cost of a launch. This information can be used by SpaceY to bid against SpaceX for a rocket launch.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data Collection using SpaceX API
  - Web Scraping from Wikipedia
- Perform data wrangling
  - Exploratory Data Analysis
  - Determine Training Labels
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Build, tune, evaluate classification models using GridSearchCV and Confusion Matrix

Data Collection

Data sets were collected using API and Web Scraping.

1. Data Collection using API
    a) Request and parse the data using the GET request
    b) Convert the requested JSON to dataframe
    c) Filter the dataframe to only include necessary informations
    d) Perform data wrangling to deal with missing values
2. Data Collection using Web Scraping
    a) Request the data using BeautifulSoup
    b) Extract relevant column/names from the HTML table header
    c) Parse the HTML tables to dataframe

Data Collection – SpaceX API

- Use the GET request to the SpaceX API to collect SpaceX launch data. Then, convert the JSON to dataframe and perform data wrangling to clean the data and deal with missing values.
- GitHub URL of the completed SpaceX API calls notebook: [IBM-Data-Science-Professional-Certificate/SpaceX Data Collection using API.ipynb at main · ajeremy15/IBM-Data-Science-Professional-Certificate (github.com)]

Now let's start requesting rocket launch data from SpaceX API with the following URL:

In [6]:
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

In [7]:
```
response = requests.get(spacex_url)
```

To make the requested JSON results more consistent, we will use the following static response object for this project:

In [9]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

We should see that the request was successfull with the 200 status response code

In [10]:
```
response.status_code
```

Out[10]: 200

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

In [14]:
```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
data
```

Finally we will remove the Falcon 1 launches keeping only the Falcon 9 launches. Filter the data dataframe using the `BoosterVersion` column to only keep the Falcon 9 launches. Save the filtered data to a new dataframe called `data_falcon9`.

In [40]:
```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = launch_df[launch_df.BoosterVersion != 'Falcon 1']
data_falcon9
```

Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the mean and the `.replace()` function to replace `np.nan` values in the data with the mean you calculated.

In [45]:
```
# Calculate the mean value of PayloadMass column
payload_mean = data_falcon9.PayloadMass.mean()
# Replace the np.nan values with its mean value
data_falcon9[['PayloadMass']] = data_falcon9[['PayloadMass']].replace(np.nan,payload_mean)
data_falcon9.head()
```

Data Collection – Web Scraping

- Use BeautifulSoup to request SpaceX's Falcon 9 launch HTML page from Wikipedia. Then, extract column/variable names from the HTML header and parse them into dataframe.
- GitHub URL of the completed web scraping notebook: [IBM-Data-Science-Professional-Certificate/Web scraping Falcon 9 and Falcon Heavy Launches Records from Wikipedia.ipynb at main · ajeremy15/IBM-Data-Science-Professional-Certificate (github.com)]

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

In [13]:
```
# use requests.get() method with the provided static_url
page = requests.get(static_url).text
# assign the response to a object
```

Create a `BeautifulSoup` object from the HTML `response`

In [14]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(page, 'html.parser')
```

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the external reference link towards the end of this lab

In [16]:
```
# Use the find_all function in the BeautifulSoup object, with element type `table`
html_tables = soup.find_all('table')
# Assign the result to a list called `html_tables`
print(html_tables)
```

We will create an empty dictionary with keys from the extracted column names in the previous task. Later, this dictionary will be converted into a Pandas dataframe

In [20]:
```
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']=[]
```

After you have fill in the parsed launch record values into `launch_dict`, you can create a dataframe from it.

In [22]:
```
df=pd.DataFrame(launch_dict)
```

Data Wrangling – Exploratory Data Analysis

- Exploratory Data Analysis were performed to find patterns in the data. This process includes calculation of the number of launches on each site, the number and occurrence of each orbit, and the number and occurrence of mission outcome per orbit type.
- GitHub URL of the completed data wrangling notebooks: [IBM-Data-Science-Professional-Certificate/SpaceX Data Wrangling.ipynb at main · ajeremy15/IBM-Data-Science-Professional-Certificate (github.com)]

Use the method `value_counts()` on the column `LaunchSite` to determine the number of launches on each site:

```
In [5]:
# Apply value_counts() on column LaunchSite
df["LaunchSite"].value_counts()

Out[5]:
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

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

```
In [6]:
# Apply value_counts on Orbit column
df["Orbit"].value_counts()

Out[6]:
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
ES-L1     1
HEO       1
SO        1
GEO       1
Name: Orbit, dtype: int64
```

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

```
In [7]:
# landing_outcomes = values on Outcome column
landing_outcomes = df["Outcome"].value_counts()
landing_outcomes

Out[7]:
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
Name: Outcome, dtype: int64
```

Data Wrangling – Determine Training Labels

- Training labels were created by creating a landing outcome label from outcome column.

In [8]:
```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

```
0 True ASDS
1 None None
2 True RTLS
3 False ASDS
4 True Ocean
5 False Ocean
6 None ASDS
7 False RTLS
```

We create a set of outcomes where the second stage did not land successfully:

In [9]:
```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

Out[9]:
```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

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

In [10]:
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise

def outcome_one_hot_encoding(value):
    if value in bad_outcomes:
        return 0
    else:
        return 1

landing_class = df["Outcome"].apply(outcome_one_hot_encoding)
landing_class
```

In [11]:
```
df['Class']=landing_class
df[['Class']].head(8)
```

Out[11]:

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

EDA with Data Visualization

- Explore the data by visualizing it using Matplotlib and Seaborn. Plots and Charts that will be used:

  1. Scatter Plot: to observe the relationship between Flight Number vs Payload Mass, Flight Number vs Launch Site, Payload Mass vs Launch Site, Flight Number vs Orbit Type, Payload Mass vs Orbit Type.

  2. Bar Chart: to observe the relationship between Orbit vs Success Rate.

  3. Line Plot: to observe the relationship between Year vs Success Rate.

- GitHub URL of the completed EDA with data visualization notebook: [IBM-Data-Science-Professional-Certificate/Exploratory Data Analysis using Pandas and Matplotlib.ipynb at main · ajeremy15/IBM-Data-Science-Professional-Certificate (github.com)]

EDA with SQL

- Explore the data using SQL by connecting Jupyter Notebook to a database in Microsoft SQL Server. Load the SpaceX launch data and perform some queries to find out:
  - the names of the unique launch sites in the space mission
  - 5 records where launch sites begin with the string 'CCA'
  - the total payload mass carried by boosters launched by NASA (CRS)
  - average payload mass carried by booster version F9 v1.1
  - the date when the first successful landing outcome in ground pad was acheived.
- GitHub URL of the completed EDA with SQL notebook: IBM-Data-Science-Professional-Certificate/Exploratory Data Analysis using SQL with SpaceX Data.ipynb at main · ajeremy15/IBM-Data-Science-Professional-Certificate (github.com)

EDA with SQL

- Some queries are also performed to find out:
  - the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
  - the total number of successful and failure mission outcomes
  - the names of the booster_versions which have carried the maximum payload mass
  - the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
  - the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20
- GitHub URL of the completed EDA with SQL notebook: IBM-Data-Science-Professional-Certificate/Exploratory Data Analysis using SQL with SpaceX Data.ipynb at main · ajeremy15/IBM-Data-Science-Professional-Certificate (github.com)

Build an Interactive Map with Folium

- Generate an interactive map with Folium to perform launch site locations analysis. Map objects that will be used:
    1. Circles: to mark all launch sites on map.
    2. MarkerCluster: to mark the success/failed launches for each site on the map.
    3. PolyLine: to calculate the distances between a launch site to its proximities (coast, railway, highway, city).
- GitHub URL of the completed interactive map with Folium map: Jupyter Notebook Viewer (nbviewer.org)

Build a Dashboard with Plotly Dash

- Create an interactive dashboard application using Plotly Dash to perform interactive visual analytics on the launch data in real-time. Plot/graphs and interactions that will be used:

    1. Dropdown Interaction: to enable launch site selections, including ‘all sites’.
    2. Pie Chart: to show the total successful launches count for selected site, including ‘all sites’.
    3. RangeSlider Interaction: to enable payload range selection.
    4. Scatter Plot: to show the correlation between payload and launch success.
    5. Callback Function: to render Pie Chart and Scatter Plot selection based on selected site dropdown.

- GitHub URL of the completed Plotly Dash lab: IBM-Data-Science-Professional-Certificate/Applied Data Science Capstone/Week 3 - Interactive Visual Analytics and Dashboard/Interactive Visual Dashboard using Dash and Plotly at main · ajeremy15/IBM-Data-Science-Professional-Certificate (github.com)

Predictive Analysis (Classification)

- Develop and evaluate machine learning models that can predict the landing outcome. The algorithm is as shown below:
    1. Load the dataset using Pandas and NumPy.
    2. Create a column for the class using NumPy Array.
    3. Standardized the data using StandardScaler.
    4. Split the data into training and test data using train_test_split.
    5. Fit the data into different machine learning models (Logistic Regression, Support Vector Machine, Decision Tree, and K-Nearest Neighbor.)
    6. Find the best hyperparameter for each model using GridSearchCV.
    7. Evaluate the accuracy using Score and visualize the result using Bar Chart to find the model that performs best.
- GitHub URL of the completed predictive analysis lab: IBM-Data-Science-Professional-Certificate/SpaceX_Machine_Learning_Prediction.ipynb at main · ajeremy15/IBM-Data-Science-Professional-Certificate (github.com)

Results

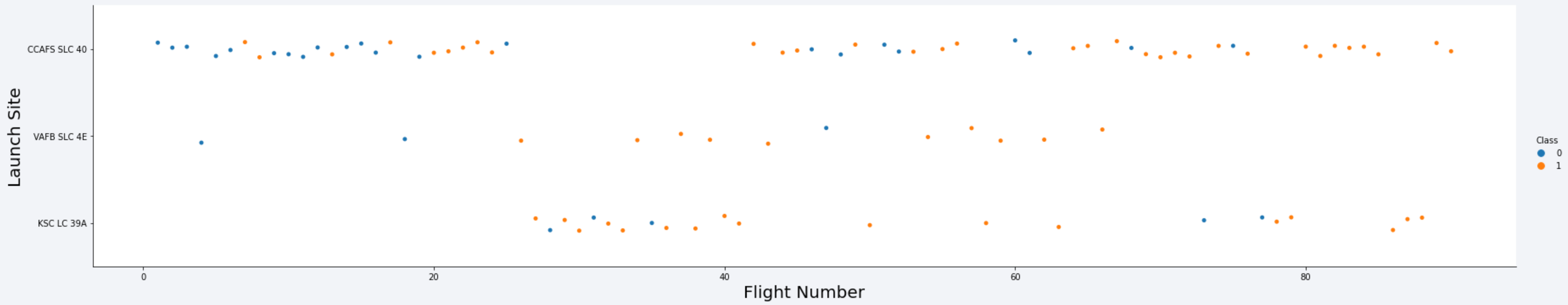
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

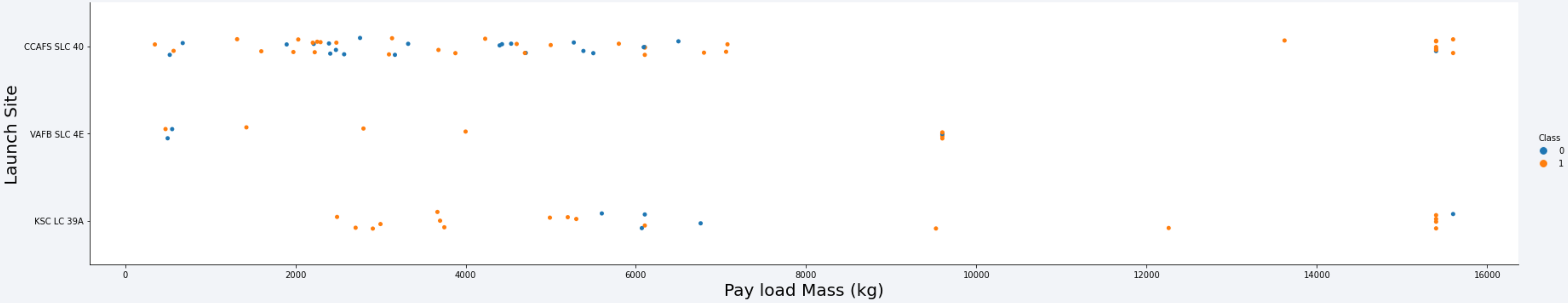
Flight Number vs. Launch Site

From the scatter plot, we can observe that increase in flight number leads to a relatively higher success rate in landing outcome for launch site CCAFS SLC 40 and KSC LC 39A.



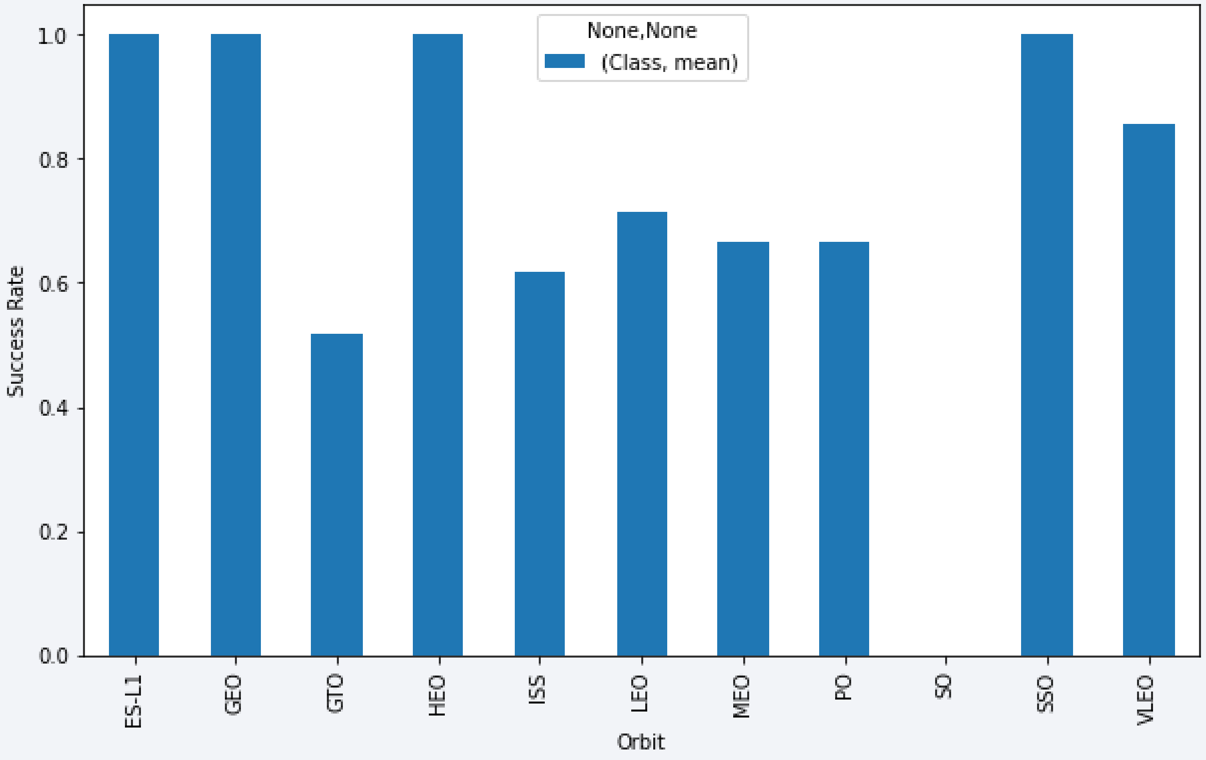
Payload vs. Launch Site

From the scatter plot, we can observe that payload mass between 2000 and 4000 kg have a very high success rate in landing outcome for launch site KSC LC 39A. Also, payload mass above 8000 kg has a very high success rate for launch site KSC LC39A and CCAFS SLC 40.



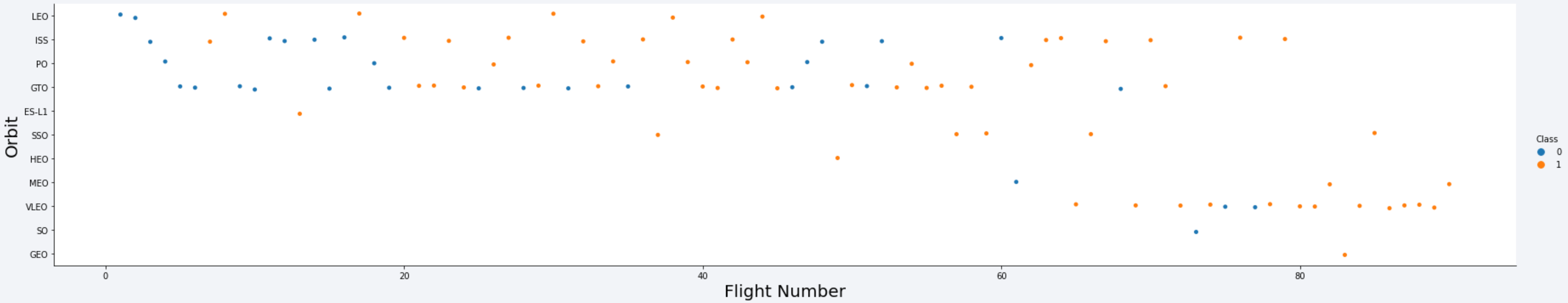
Success Rate vs. Orbit Type

From the bar chart, we can observe that orbit ES-L1, GEO, HEO, and SSO have the highest success rate.



Flight Number vs. Orbit Type

From the scatter plot, we can observe that the success rate of orbit LEO is related to the number of flights while in orbit GTO there's seem to be no relation between the success rate and the number of flights.

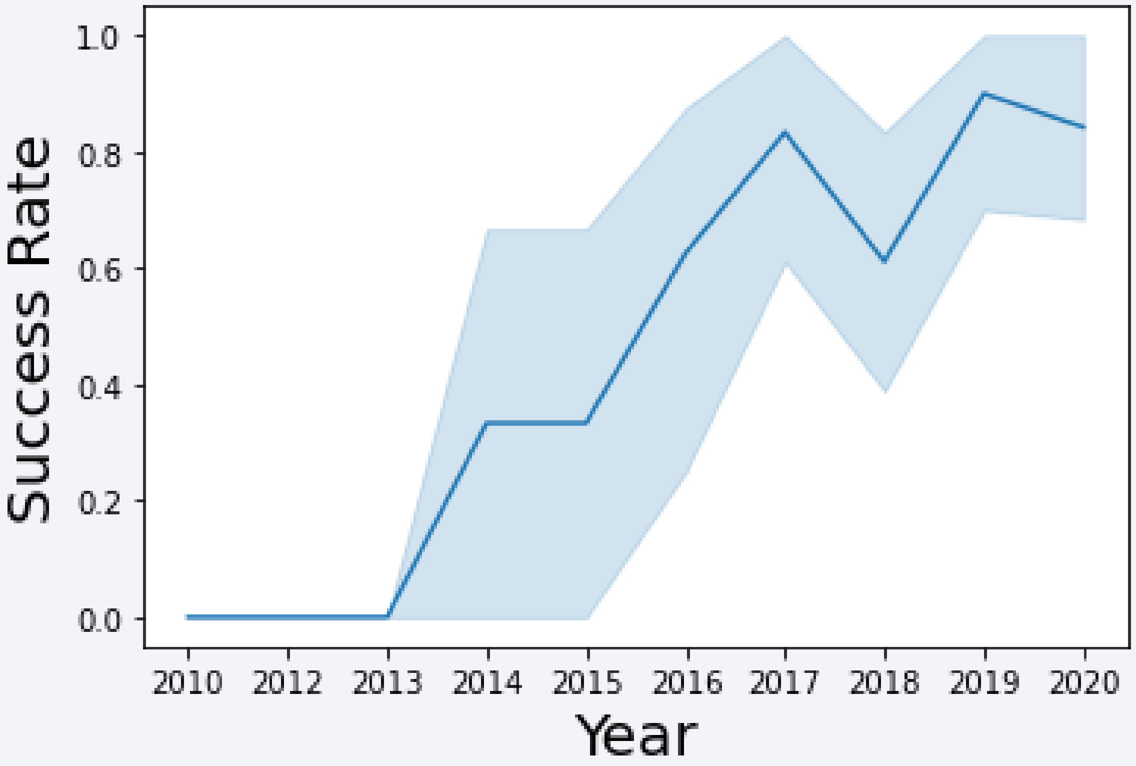


Payload vs. Orbit Type

From the scatter plot, we can observe that orbit PO, LEO, and ISS have a high success rate with heavy payloads. While on the other hand, we cannot distinguish the correlation between payload mass and success rate in orbit GTO.

Launch Success Yearly Trend

From the bar chart, we can observe that there is a positive trend in the success rate from 2013 to 2020.



All Launch Site Names

- We can use DISTINCT to return unique value from Launch_Site.
- As we can see, there are four launch site names, namely:
  - CCAFS LC-40
  - CCAFS SLC-40
  - KSC LC-39A
  - VAFB SLC-4E

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

In [4]:
```
%%sql task_1 <<

SELECT DISTINCT Launch_Site
FROM IBMDataScienceCourseSQLDatabase.dbo.Spacex
```
```
 * mssql+pyodbc://sa:***@sqlsrv
Done.
Returning data to local variable task_1
```

In [5]:
```
task_1
```

Out[5]:

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

Launch Site Names Begin with 'CCA'

- We can use WHERE and LIKE to query only Launch_Site that begins with ‘CCA’ and we can use TOP 5 to only return 5 records.

Task 2

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

In [6]:

```
%%sql task_2 <<

SELECT TOP 5 *
FROM IBMDataScienceCourseSQLDatabase.dbo.Spacex
WHERE Launch_Site LIKE 'CCA%'
```

```
 * mssql+pyodbc://sa:***@sqlsrv
Done.
Returning data to local variable task_2
```

In [7]:

```
task_2
```

Out[7]:

Date	Time_UTC	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS_KG	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00.0000000	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00.0000000	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	07:44:00.0000000	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00.0000000	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00.0000000	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- We can use SUM to calculate PAYLOAD_MASS_KG and we can use WHERE to only select customers from NASA.
- We can see that the total payload mass carried by boosters from NASA is 45596 kg.

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

In [8]:
```
%%sql task_3 <<

SELECT SUM(PAYLOAD_MASS_KG) as Total_Payload_Mass_NASA_CRS_KG
FROM IBMDataScienceCourseSQLDatabase.dbo.Spacex
WHERE Customer = 'NASA (CRS)'
```

```
 * mssql+pyodbc://sa:***@sqlsrv
Done.
Returning data to local variable task_3
```

In [9]:
```
task_3
```

Out[9]:

Total_Payload_Mass_NASA_CRS_KG
45596

Average Payload Mass by F9 v1.1

- We can use AVG to calculate the average PAYLOAD_MASS_KG and we can use WHERE and LIKE to only query booster version F9 v1.1.
- We can see that the average payload mass carried by booster version F9 v1.1 is 2534 kg.

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

```
In [10]:  %%sql task_4 <<

          SELECT AVG(PAYLOAD_MASS_KG) as Average_Payload_Mass_Booster_version_F9v1_1_KG
          FROM IBMDataScienceCourseSQLDatabase.dbo.Spacex
          WHERE Booster_Version LIKE 'F9 v1.1%'

           * mssql+pyodbc://sa:***@sqlsrv
          Done.
          Returning data to local variable task_4

In [11]:  task_4

Out[11]:
```

Average_Payload_Mass_Booster_version_F9v1_1_KG
2534

First Successful Ground Landing Date

- We can use MIN to find the dates of the first successful and we can use WHERE to only include successful landing outcome on ground pad.
- We can see that the first successful landing in ground pad was on December 22nd, 2015.

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

Hint:Use min function

In [12]:

```
%%sql task_5 <<

SELECT MIN(Date) AS First_Success_Landing_Ground_Pad
FROM IBMDataScienceCourseSQLDatabase.dbo.Spacex
WHERE Landing_Outcome = 'Success (ground pad)'
```

```
 * mssql+pyodbc://sa:***@sqlsrv
Done.
Returning data to local variable task_5
```

In [13]:

```
task_5
```

Out[13]:

First_Success_Landing_Ground_Pad
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- We can use WHERE to only list the names of boosters which have successfully landed on drone ship and we can set the PAYLOAD_MASS_KG to be greater than 4000 but less than 6000.
- We can see that there are four booster version, namely: F9 FT B1022, F9 FT B1026, F9 FT B1021.2, and F9 FT B1031.2.

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

In [14]:

```
%%sql task_6 <<

SELECT Booster_Version
FROM IBMDataScienceCourseSQLDatabase.dbo.Spacex
WHERE Landing_Outcome = 'Success (drone ship)' AND PAYLOAD_MASS_KG > 4000 AND PAYLOAD_MASS_KG < 6000
```

```
 * mssql+pyodbc://sa:***@sqlsrv
Done.
Returning data to local variable task_6
```

In [15]:

```
task_6
```

Out[15]:

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

Total Number of Successful and Failure Mission Outcomes

- We can use COUNT and CASE WHEN to Calculate the total number of successful and failure mission outcomes.
- The total of success mission outcome is 100 while the total of failed mission outcome is 1. This indicates that the mission outcome has a very high success rate.

List the total number of successful and failure mission outcomes

In [16]:
```
%%sql task_7 <<

SELECT
    COUNT(CASE WHEN Mission_Outcome LIKE 'Success%' THEN 1 END) AS Total_Success,
    COUNT(CASE WHEN Mission_Outcome LIKE 'Failure%' THEN 1 END) AS Total_Failure
FROM IBMDataScienceCourseSQLDatabase.dbo.Spacex
```

```
 * mssql+pyodbc://sa:***@sqlsrv
Done.
Returning data to local variable task_7
```

In [17]:
```
task_7
```

Out[17]:

Total_Success	Total_Failure
100	1

Boosters Carried Maximum Payload

- We can use Subquery to list the names of the booster which have carried the maximum payload mass.
- We can see that booster version that begins with F9 B5 has carried the maximum payload mass.

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

In [18]:

```
%%sql task_8 <<

SELECT Booster_Version
FROM IBMDataScienceCourseSQLDatabase.dbo.Spacex
WHERE PAYLOAD_MASS_KG IN (SELECT MAX(PAYLOAD_MASS_KG) FROM IBMDataScienceCourseSQLDatabase.dbo.Spacex)
```

```
 * mssql+pyodbc://sa:***@sqlsrv
Done.
Returning data to local variable task_8
```

In [19]:

```
task_8
```

Out[19]:

Booster_Version
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

- We can use WHERE to only query the failed landing_outcomes in drone ship and we can use LIKE to only include date that has '2015' in them.
- We can see that in 2015 the failed landing outcomes in drone ship were all from Launch Site CCAFS LC-40.

List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

In [20]:
```
%%sql task_9 <<

SELECT Booster_Version, Launch_Site, Landing_Outcome
FROM IBMDataScienceCourseSQLDatabase.dbo.Spacex
WHERE Landing_Outcome = 'Failure (drone ship)' AND Date LIKE '2015%'
```

```
 * mssql+pyodbc://sa:***@sqlsrv
Done.
Returning data to local variable task_9
```

In [21]:
```
task_9
```

Out[21]:

Booster_Version	Launch_Site	Landing_Outcome
F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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

- We can use COUNT to calculate the total landing outcome, WHERE and BETWEEN to filter the date, GROUP BY to group the landing outcome and ORDER BY DESC to sort the result in descending order.
- We can see that ‘No attempt’ is the highest landing outcome between 22 June 2010 and 20 March 2017.

In [22]:

```
%%sql task_10 <<

SELECT Landing_Outcome, COUNT(*) AS Total
FROM IBMDataScienceCourseSQLDatabase.dbo.Spacex
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY Landing_Outcome
ORDER BY Total DESC
```

```
 * mssql+pyodbc://sa:***@sqlsrv
Done.
Returning data to local variable task_10
```

In [23]:

```
task_10
```

Out[23]:

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

Section 3

Launch Sites Proximities Analysis

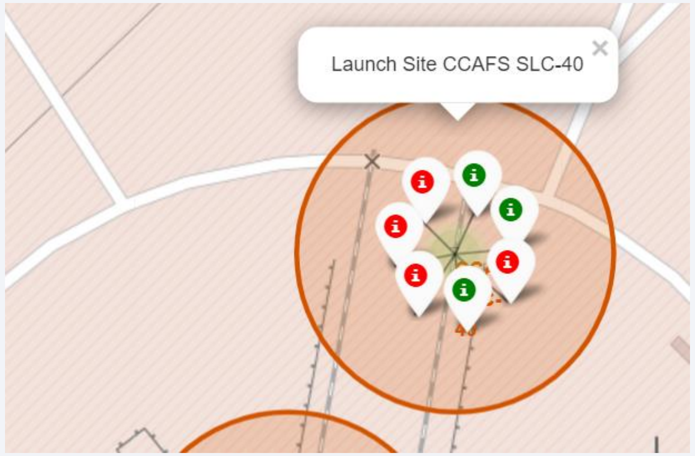
Launch Site Locations on the Map

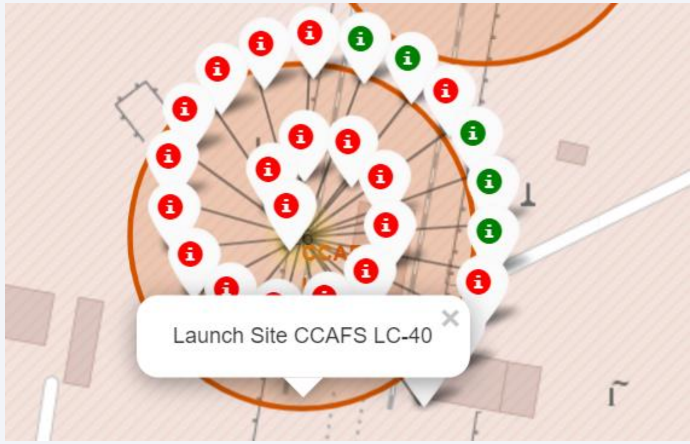
Launch Site Locations Analysis

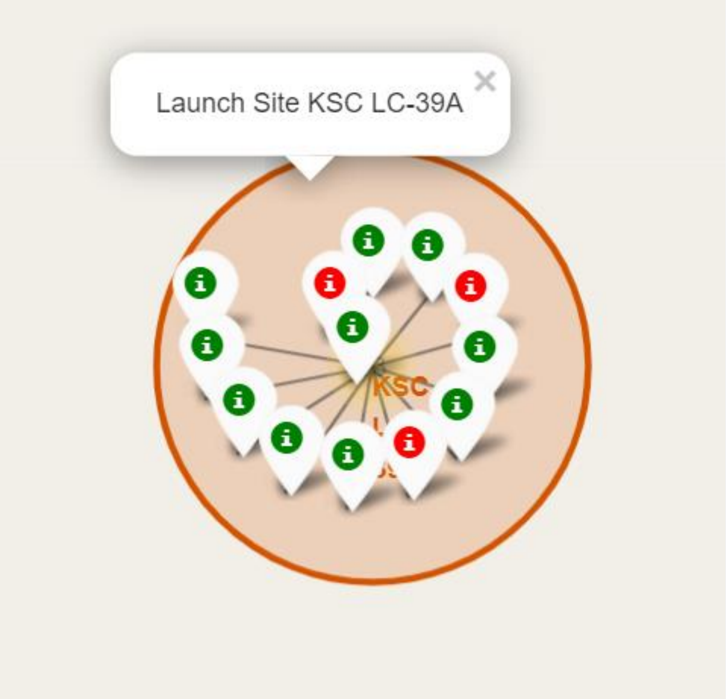
The map shows the marked locations of all the launch sites. From the map, we can see that:

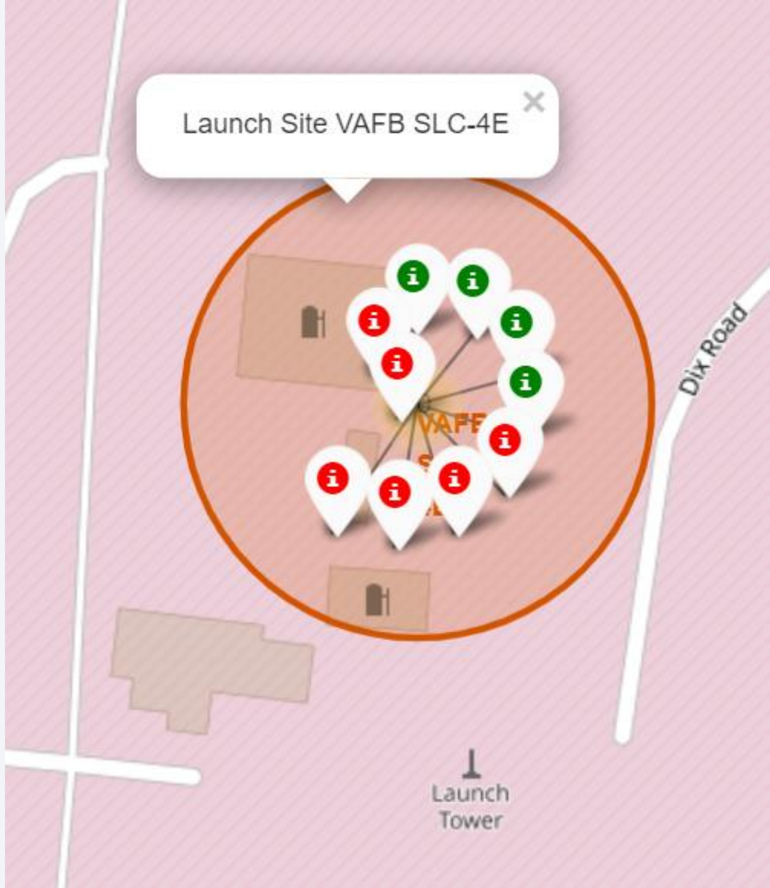
- All launch sites are in proximity to the equator, so that the spaceships can take advantage of the Earth's Substantial rotational speed and get additional boosts.
- All launch sites are in proximity to the coast, so that if something goes wrong, the debris is more likely to fall into an ocean, which is far away from populated areas like city.

Success/Failed Launches Marker on the Map







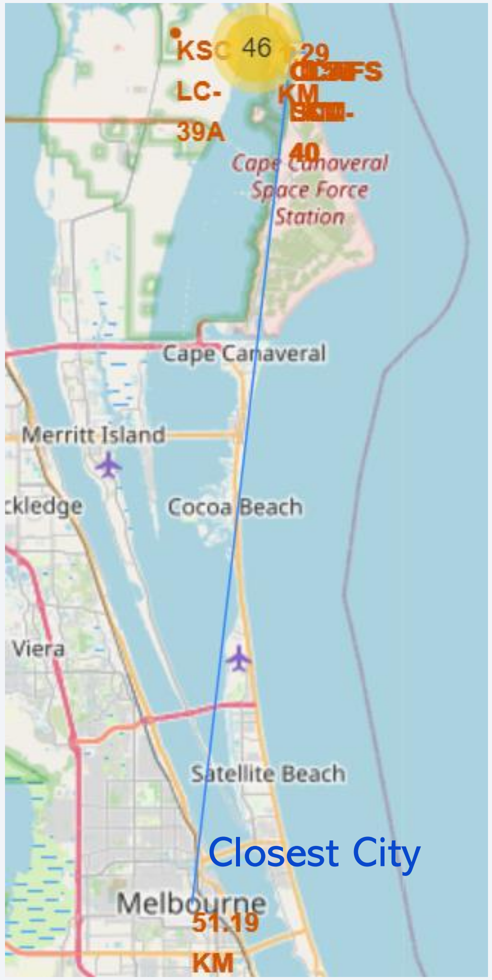


Success/Failed Launches Marker Analysis

The map shows the marker of all the success and failed launches on each launch sites on the map, where green indicates success launches and red represents failed launches. From the launches marker on each launch sites, we can see that:

- Launch Site KSC LC-39A has the highest launches success rate.
- Launch Site CCAFS LC-40 has attempted the most launches.

Distance between a Launch Site to its Proximities on the Map



Distance between a Launch Site to its Proximities Analysis

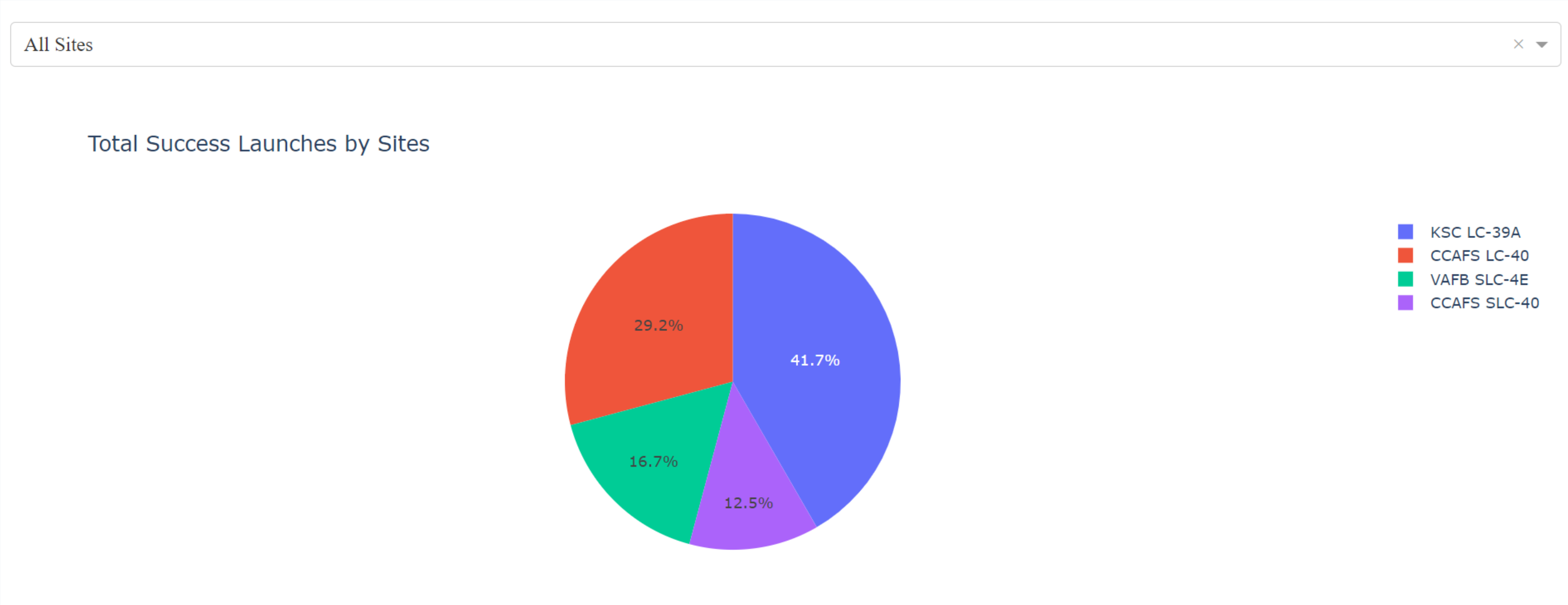
The map shows the distance between a launch site to its proximities. We can see that:

- The distance to closest railway is 1.29 km, which is close. This is because railway is necessary for transporting heavy cargo.
- The distance to closest highway is 0.59 km, which is very close. This is because highway is necessary for transporting personnel and equipment.
- The distance to closest coastline is 0.86 km, which is close. So that, If something goes wrong, the debris is more likely to fall into an ocean.
- Distance to closest city is 51.19 km, which is very far. Launch sites keep certain distances away from cities for safety reasons.

Section 4

Build a Dashboard with Plotly Dash

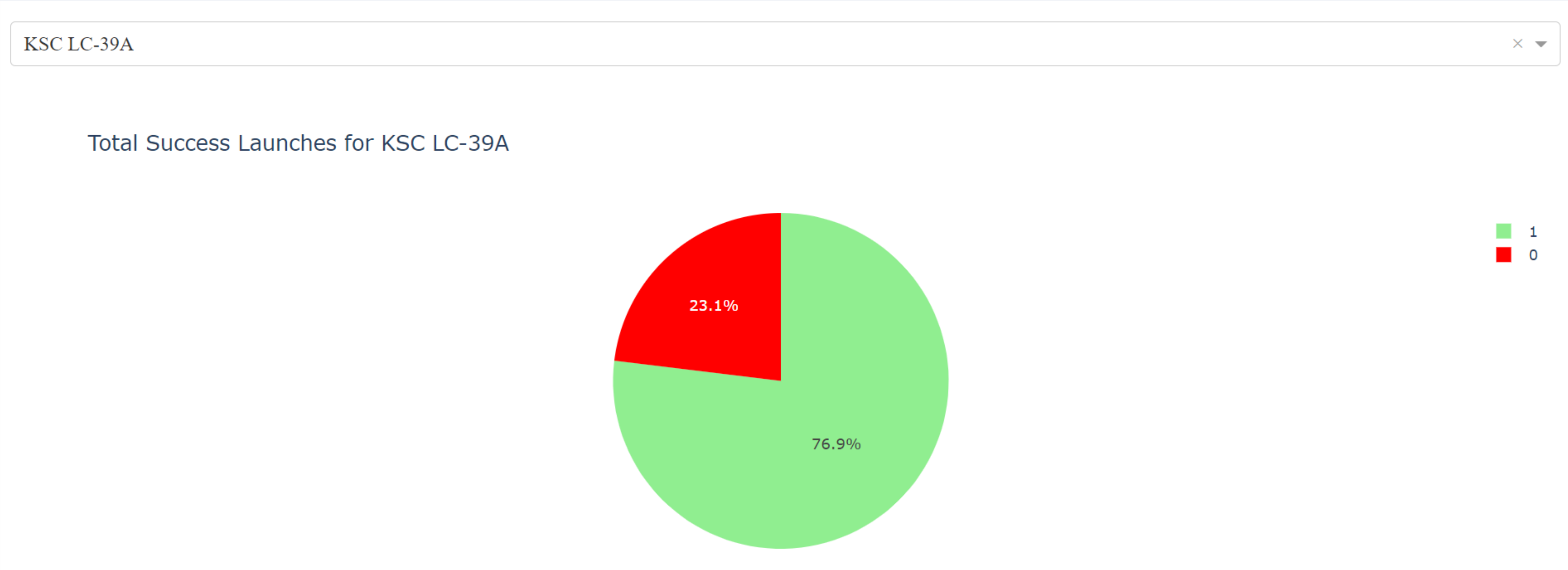
Total Success Launches by Sites



Total Success Launches by Sites Analysis

- The pie chart shows the percentage of success launches from all launch sites.
- From the pie chart, we can see that launch site KSC LC-39A has the highest percentage of success launches, which is 41.7%.

Total Success Launches for Launch Site KSC LC-39A



Total Success Launches for Launch Site KSC LC-39A Analysis

- The pie chart shows the percentage of success and failed launches for Launch Site KSC LC-39A, where 1 represents the success launches rate and 0 represents the failed launches rate.
- From the pie chart, we can see that launch site KSC LC-39A has a success launches rate of 76.9% and a failed launches rate of 23.1%.

Correlation between Payload and Success Launches for All Sites

Correlation between Payload and Success Launches for All Sites Analysis

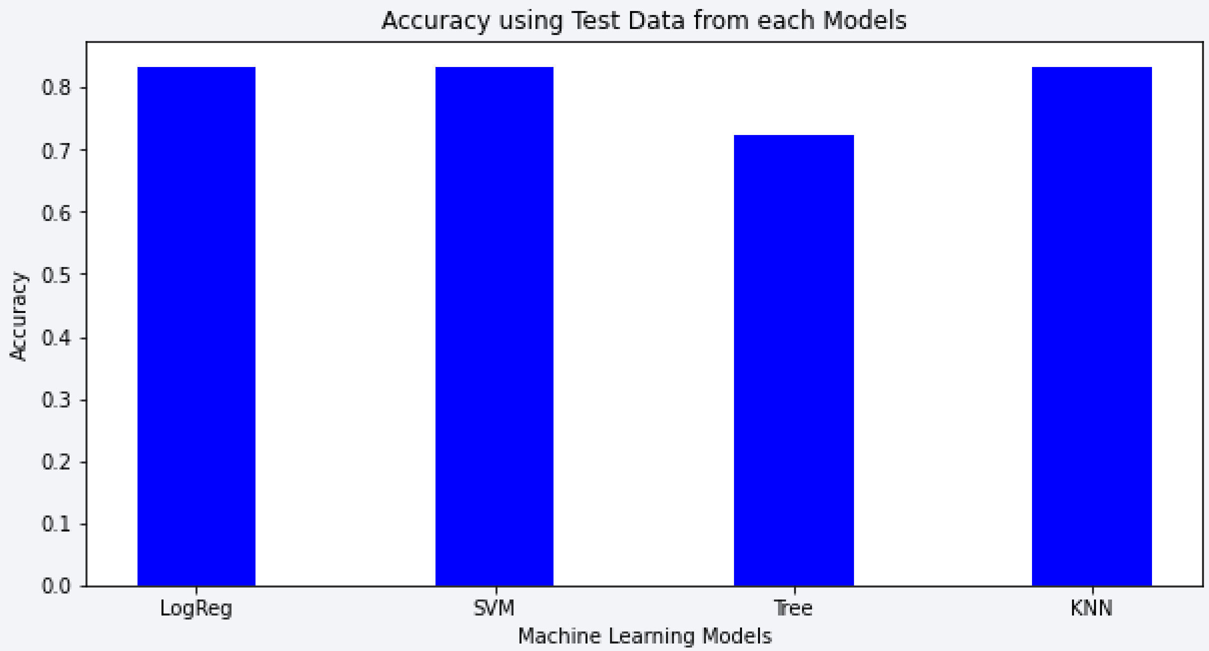
- The pie chart shows the correlation between payload and success launches for all sites using different booster versions.
- From the pie chart, we can see that payloads below 4000 kg has a higher success rate than payloads above 4000 kg.
- We can also observe that booster version FT has the highest success rate.

Section 5

Predictive Analysis (Classification)

Classification Accuracy using Test Data

- This bar chart shows the classification accuracy of each models, where:
  - LogReg is Logistic Regression
  - SVM is Support Vector Machine
  - Tree is Decision Tree
  - KNN is K-Nearest Neighbor
- From the bar chart, we can see that LogReg, SVM, and KNN has the highest classification accuracy.



Confusion Matrix of LogReg, SVM, and KNN

- The confusion matrix shows different classes which represent the number of correct and incorrect predictions from the best performing models (LogReg, SVM, and KNN).
- We can see that the major problem is the false positives where the models labeled 3 failed landing as successful landing.

Conclusions

- Increase in numbers of flight leads to a higher success rate.
- There is a positive trend of success rate from 2013 to 2020.
- Launches that are aim to Orbit ES-L1, GEO, HEO, and SSO have the highest success rate.
- Launches from Launch Site KSC LC-39A have the highest success rate.
- Launches with payload mass below 4000 kg have a higher success rate.
- Launches using booster version FT have the highest success rate.
- Logistic Regression, Support Vector Machine, and K-Nearest Neighbor are the best machine learning algorithms to predict if the first stage of SpaceX's Falcon 9 will land successfully.

Appendix

- Link to the complete notebook used for this project: [IBM-Data-Science-Professional-Certificate/Applied Data Science Capstone at main · ajeremy15/IBM-Data-Science-Professional-Certificate (github.com)]

- Acknowledgements:

  - Thanks to Joseph Santarcangelo and Yan Luo for creating the course materials and lab sessions in the Applied Data Science Capstone.

  - Thanks to Rav Ahuja, Alex Aklson, Aije Egwaikhide, Svetlana Levitan, Romeo Kienzler, Polong Lin, Azim Hijrani, Hima Vasudevan, Saishruti Swaminathan, and Saeed Aghabozorgi for creating the courses materials and lab sessions in the IBM Data Science Professional Certificate.

- References:

  - https://api.spacexdata.com/v4/launches/past

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

  - https://www.spacex.com/media/Capabilities&Services.pdf

Thank you!