IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Jungkee Song
March 10, 2023

Outline

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

Executive Summary

- Methodologies
  - Collected Falcon 9 launches data from SpaceX and Wikipedia via REST API and web scraping
  - Carried out EDA by examining the variables (Pandas, SQL), drawing plots, setting up an interactive dashboard
  - Cleaned up important features and determined the label (launch outcomes)
  - Selected important features that show correlations
  - Evaluated 4 ML classification models with the features and label, and identify the best model
- Results
  - Launch sites, orbits, payload mass, and flight numbers are identified as important features.
  - Logistic Regression, SVM, Decision Tree, and KNN trained with the selected features showed high level of accuracy. F1 scores of >=0.81.
  - The Decision Tree Classifier is the best model among the evaluated models, resulting in the F1 score of 0.94.

Introduction

- Background
  - We live in commercial space age: many companies explore space travel and space missions.
  - Space exploration demands enormous costs. SpaceX has achieved good milestones because they reduced costs by reusing the first stage of the rockets.
  - SpaceX's Falcon 9 launch cost is 62M, much less compared to 165M by other providers.
  - This means we could better understand the cost expectation of business if we could predict if launches can land the first stages successfully.
- Problem and Hypothesis
  - Determining the price of each launch is difficult because the first stage landing in a rocket launch is sometimes successful but sometime fails or is sacrificed for mission parameters.
  - Hypothesis: The public data about SpaceX's Falcon 9 past launches can provide relevant features that can predict the outcome of future launch attempts.

Section 1

Methodology

Methodology

Executive Summary

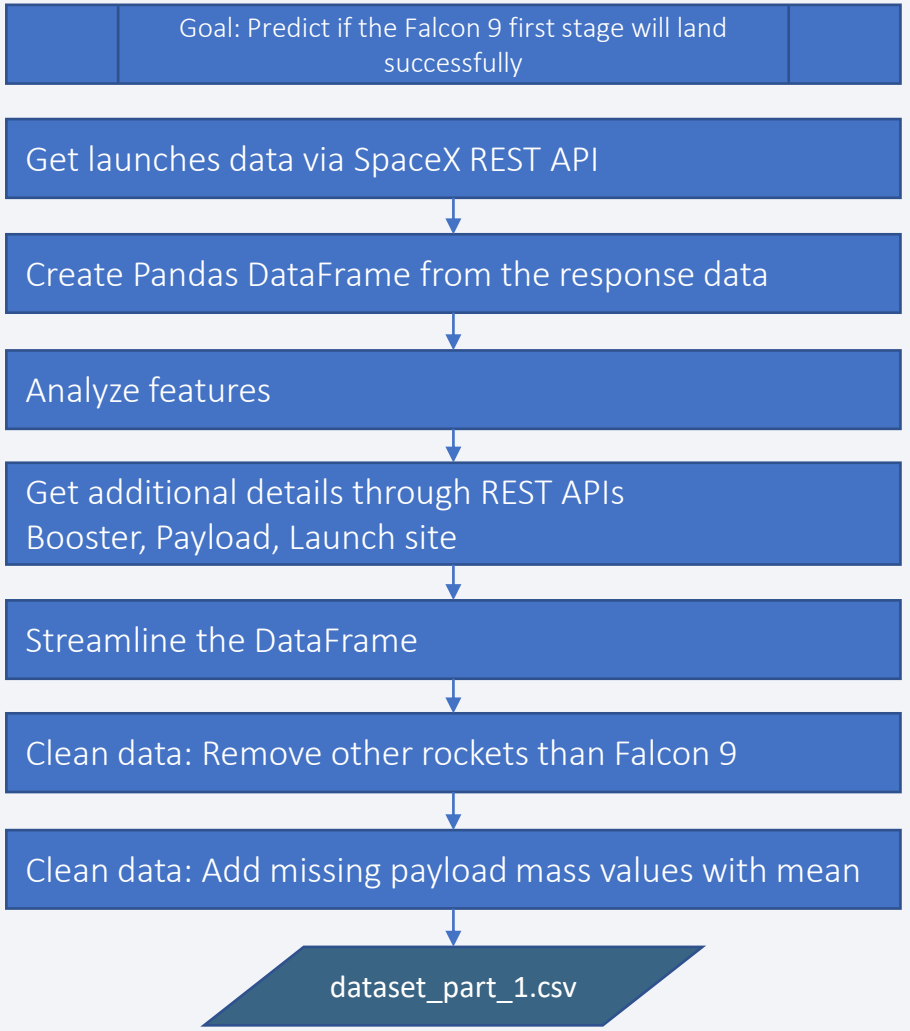
- Performed data collection
  - Used SpaceX's REST API and web scraping to get launch data from SpaceX and Wikipedia
- Performed data wrangling
  - Explored attributes, added missing values, and determined the label
- Performed exploratory data analysis (EDA) using visualization and SQL
- Performed interactive visual analytics using Folium and Plotly Dash
- Performed predictive analysis using classification models
  - Cross evaluated the following models: Logistic Regression, Decision Tree Classifier, KNN, and SVM
  - Streamlined features and label, normalized the data, split train and test sets, used GridSearchCV to get the optimal hyperparameter values, trained the models with training data, predicted the outcome with the testing data, evaluated/compared the performance using confusion matrix and accuracy scores

Data Collection

- Collected datasets through the following methodologies
  - Fetched data about SpaceX launches through a REST API: requests.get('https://api.spacexdata.com/v4/launches/past')
  - Additionally fetched BoosterVersion, LaunchSite, PayLoadData, and CoreData through REST APIs
  - Took subset of the dataframe as features to explore
  - Cleaned the data: dropping other rockets than the Falcon 9, adding missing values in PayloadMass feature (mean)
  - Web scraped Falcon 9 launch data from the Wiki page and created a DataFrame from the table

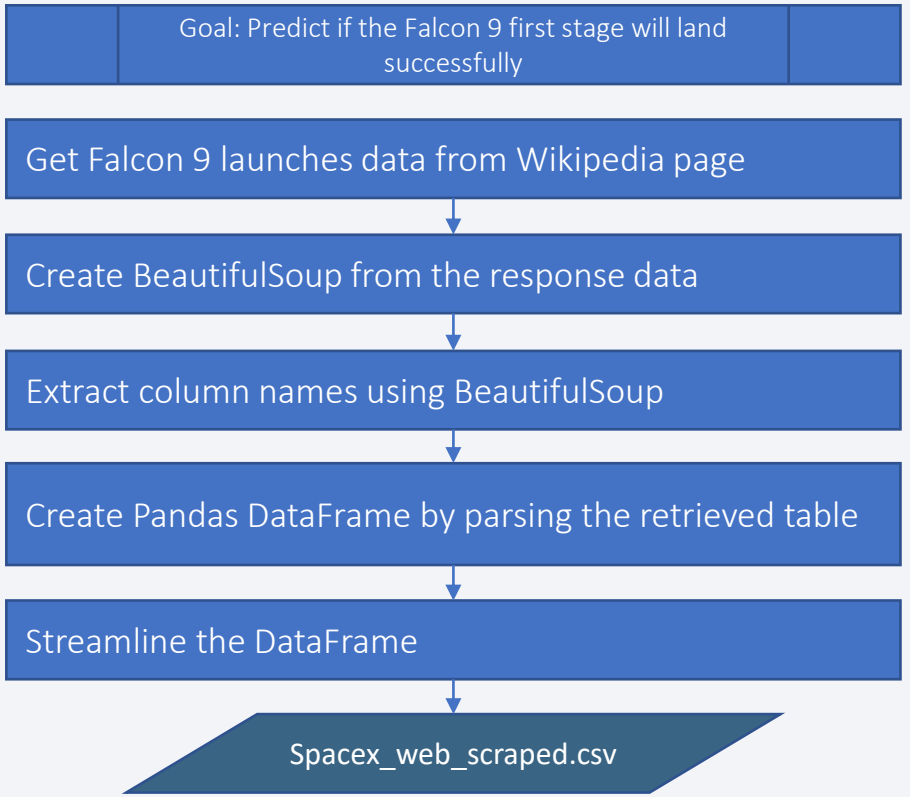
Data Collection – SpaceX API

- Collected data about rocket launches via SpaceX REST calls
- Inspected features and cleaned data
- Completed SpaceX API calls notebook: [IBM-data-science-capstone/Notebook1_Data_Collection_Using_API.ipynb at main · jungkees/IBM-data-science-capstone · GitHub]



Data Collection - Scraping

- Collected data about Falcon 9 launches via web scraping
- Pulled data from the Wikipedia page and extracted information using BeautifulSoup API
- Completed web scraping notebook: IBM-data-science-capstone/Notebook2_Web_Scraping.ipynb at main · jungkees/IBM-data-science-capstone · GitHub



Data Wrangling

- Performed data wrangling
  - Checked missing values and data types
  - Explored Launch Sites, Orbits, and Outcome variables
  - Parsed the meaning of Outcome variable and classified the values into success or failure class
  - Coded the Outcome as 0 and 1
  - Created a column called Class as a determined label

- Completed data wrangling notebook: IBM-data-science-capstone/Notebook3_Data_Wrangling.ipynb at main · jungkees/IBM-data-science-capstone · GitHub

EDA with Data Visualization

- Plotted the following charts to understand patterns and relations of variables
  - Flight number vs Payload Mass with color coded Class (Scatter)
  - Flight number vs Launch site with color coded Class (Scatter)
  - Payload mass vs Launch site with color coded Class (Scatter)
  - Success rates of Orbits (Bar)
  - Flight number vs Orbit with color coded Class (Scatter)
  - Payload mass vs Orbit with color coded Class (Scatter)
  - Yearly success rates (Line)
- Completed EDA with data visualization notebook: IBM-data-science-capstone/Notebook5_EDA_DataViz.ipynb at main · jungkees/IBM-data-science-capstone · GitHub

EDA with SQL

- Using bullet point format, summarize the SQL queries you performed
  - Identified unique launch sites
  - Retrieved launch attempts carried out in the sites starting with 'CCA'
  - Get the total payload mass launched for the customer, NASA (CRS)
  - Get the average payload mass carried by F9 v1.1
  - Identified the date of the first successful landing in ground pad
  - Identified the Booster versions that succeeded landing in drone ship with the payload mass of >4,000 and <6,000
  - Get the counts of successful and failure mission outcomes
  - Identified the Booster names that have carried the maximum payload mass using a sub-query
  - Identified launches that failed in drone ship in 2015
  - Ranked the count of successful landing outcomes between 04/06/2010 and 20/03/2017
- Completed EDA with SQL notebook: IBM-data-science-capstone/Notebook4_EDA_SQL.ipynb at main · jungkees/IBM-data-science-capstone · GitHub

Build an Interactive Map with Folium

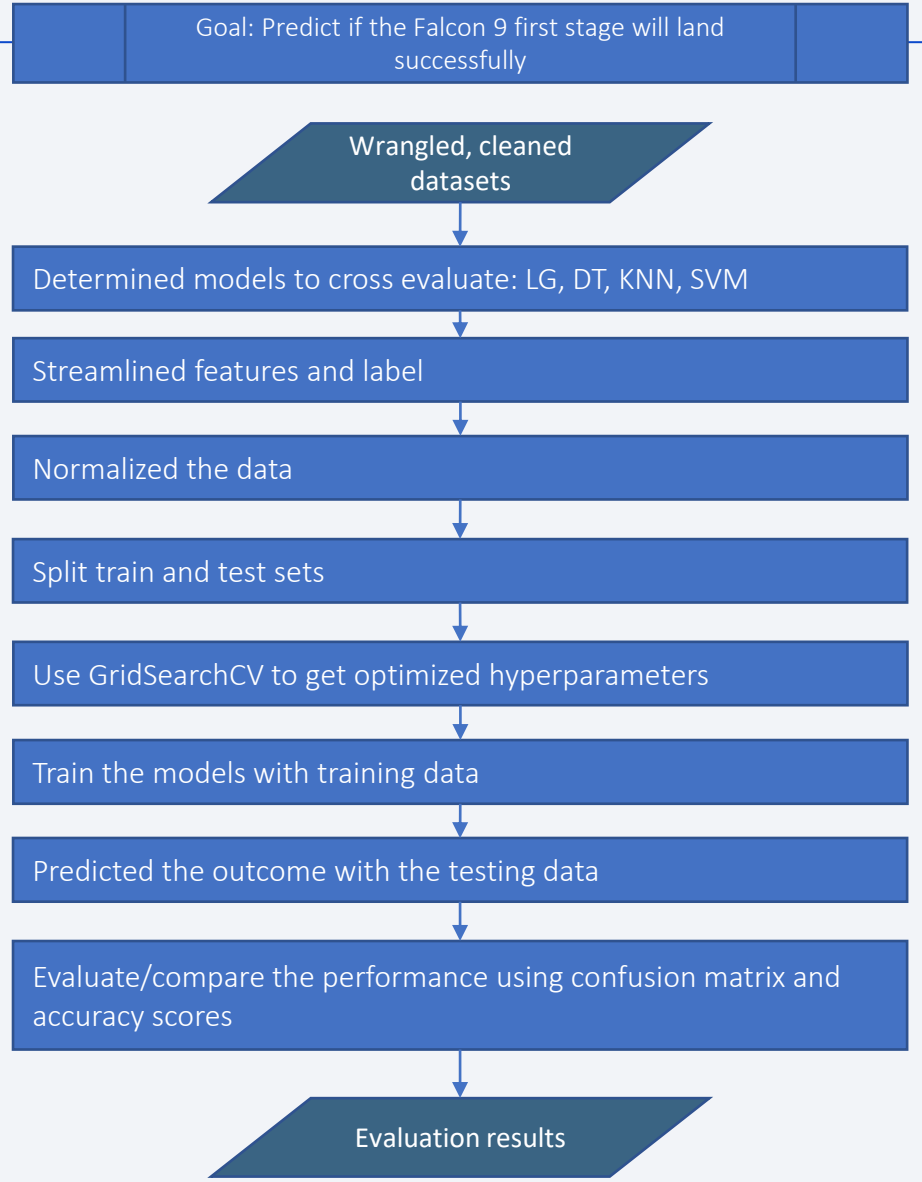
- Added launch sites to a folium map using their geo coordinates
- Created the following map objects and added them to the folium map
  - Circles – to add a highlighted circle area to each launch site with a pop-up
  - Markers – to add a text label to each launch site
  - MarkerCluster – to indicate multiple success/failed launches at the same coordinate of each launch site
  - Lines – to intuitively show a distance between a launch site and its proximity like a coastline, railway, and highway.
- Completed interactive map with Folium map: IBM-data-science-capstone/Notebook6_Launch_Site_Locations_Analysis.ipynb at main · jungkees/IBM-data-science-capstone · GitHub

Build a Dashboard with Plotly Dash

- Added the following chart/plot and interactions to the dashboard
  - Pie chart that shows success counts and rates for Launch Sites
    - Made this chart configurable by a dropdown with Launch Sites as value options
  - Scatter plot for Payload Mass vs Class with color coded Booster information
    - Made this chart configurable by a dropdown with Launch Sites as value options, and a slider for Payload Mass range
- Added this interactive dashboard to encourage stakeholders to try themselves and get the insights on the impact of Launch Site and Payload Mass on the launch outcome
- Completed Plotly Dash lab
  - Python code: IBM-data-science-capstone/dash_code.py at main · jungkees/IBM-data-science-capstone · GitHub
  - Dashboard snapshots (Link)

Predictive Analysis (Classification)

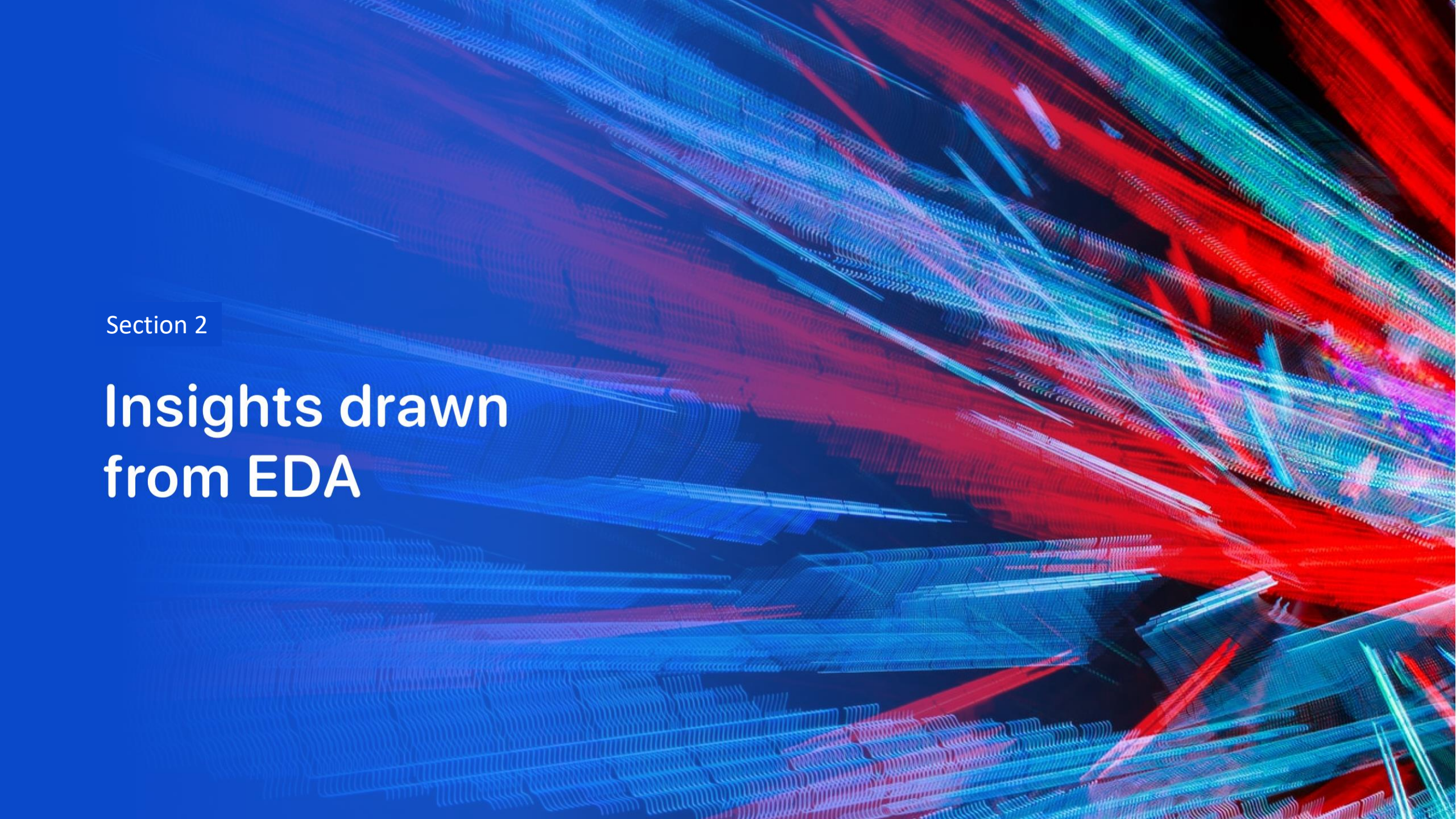
- Started with the data wrangled and cleaned from the previous stages
- Determined to evaluate multiple classification algorithms: Linear Regression, Decision Tree, KNN, and SVM
- Cross validated the models with the optimized hyperparameters
- Evaluated the accuracy scores to decide the best performing model
- Completed predictive analysis lab: [IBM-data-science-capstone/Notebook7_Machine_Learning_Prediction.ipynb at main · jungkees/IBM-data-science-capstone · GitHub]



Results

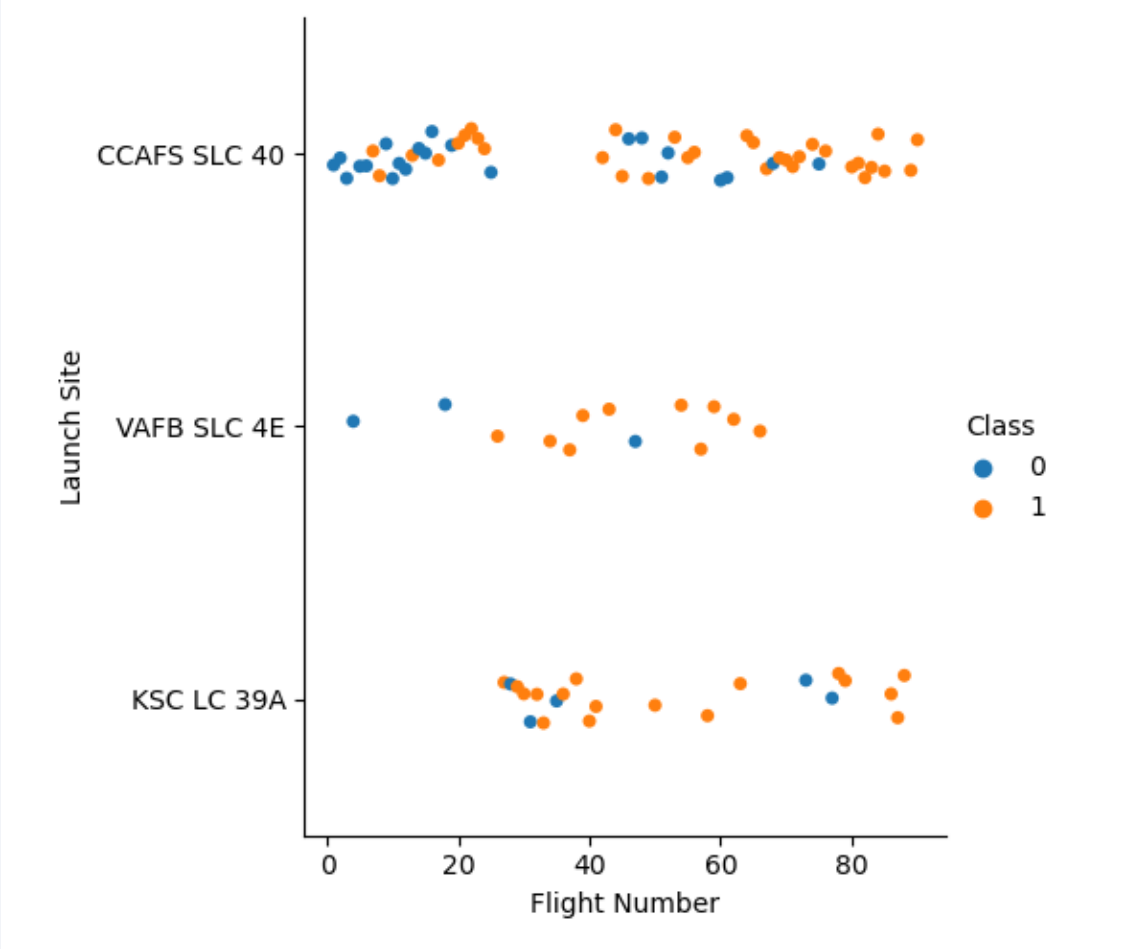
- Exploratory data analysis results
  - Launch Sites, Payload Mass, Orbits, and Flight Numbers show meaningful correlations with success rates, and also with one another themselves
- Interactive analytics demo in screenshots
  - Explored the above variables with an interactive dashboard
- Predictive analysis results
  - Evaluated 4 classification models: Logistic Regression, SVM, Decision Tree, and KNN based on cross validation method
  - Trained and tested the models with the features cleaned and selected from the previous stages
  - Decision Tree performs best with the highest accuracy score and F1 score.

More screenshots in the later slides


Section 2
Insights drawn from EDA

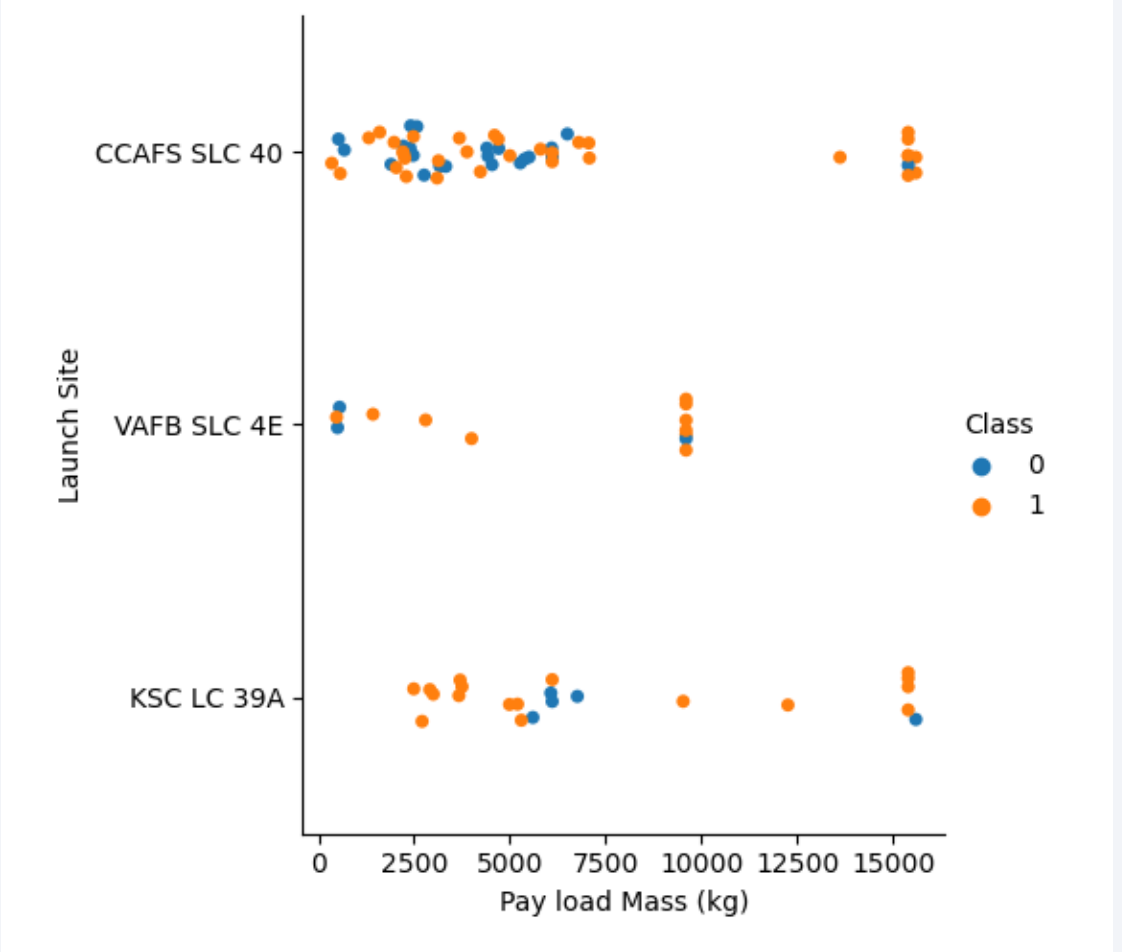
Flight Number vs. Launch Site

- Flight Number vs. Launch Site
- Observation
  - CCAFS SLC 40 has the highest failure rate among other sites.
  - Success rates have improved overall as the flight number increased.
  - CCAFS SLC 40 and VAFB SLC 4E have high failure rates from their early attempts.



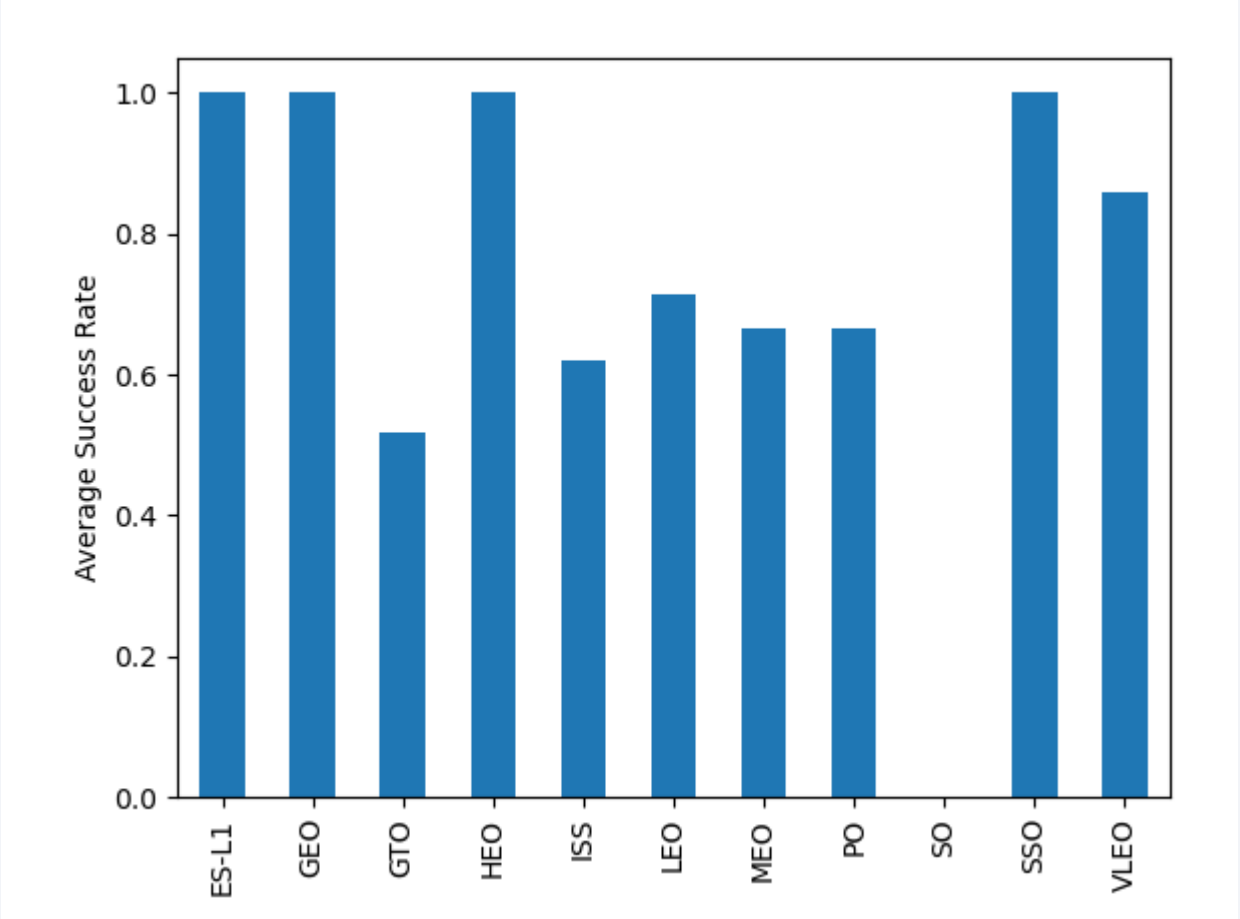
Payload vs. Launch Site

- Payload Mass vs. Launch Site
- Observation
  - VAFB SLC 4E has no rockets launched for heavy payload mass >10,000
  - CCAFS SLC 40 has high failure rate especially for the payload mass range of <7,500



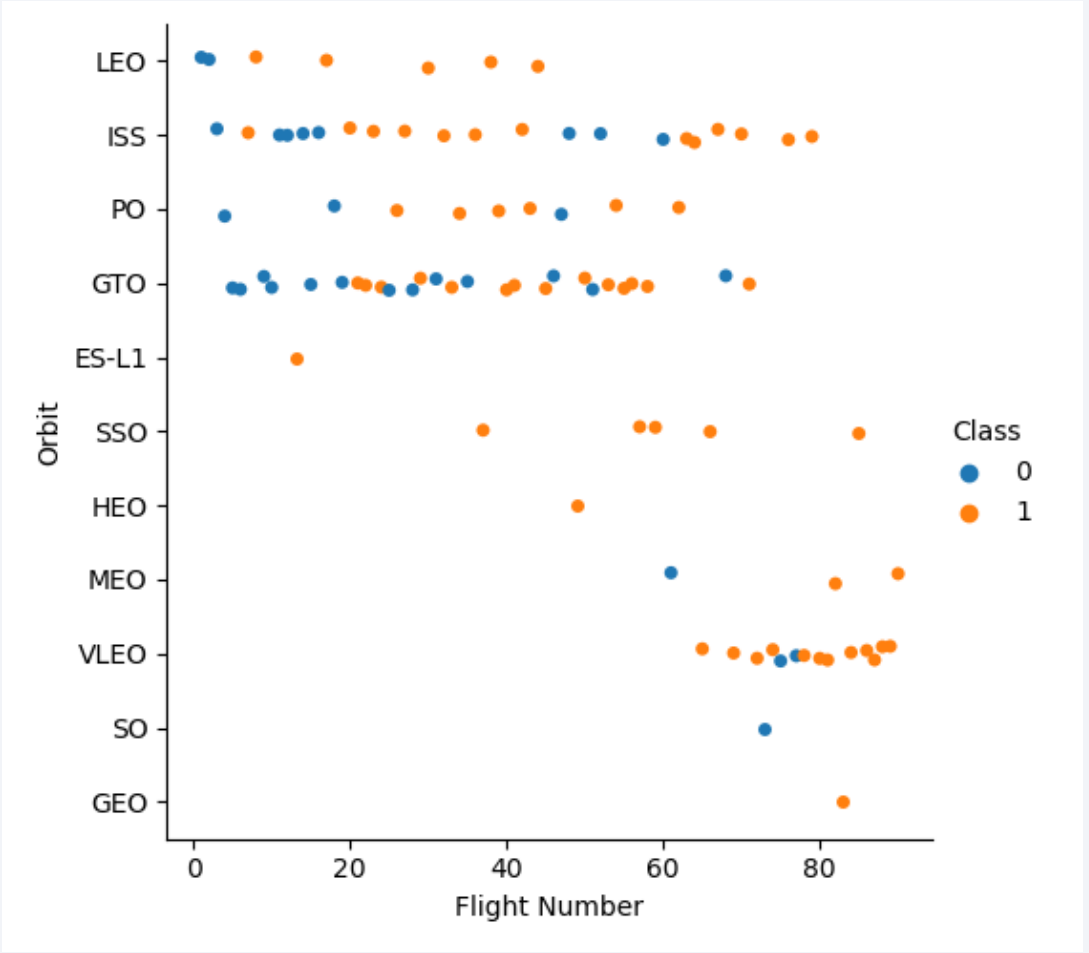
Success Rate vs. Orbit Type

- Success rate of each orbit type
- Observation
  - The orbits, ES-L1, GEO, HEO, and SSO, have high success rates.
  - GTO has the lowest success rate of around 50%.
  - The rest of the orbits have the success rates between 60 to 70%.



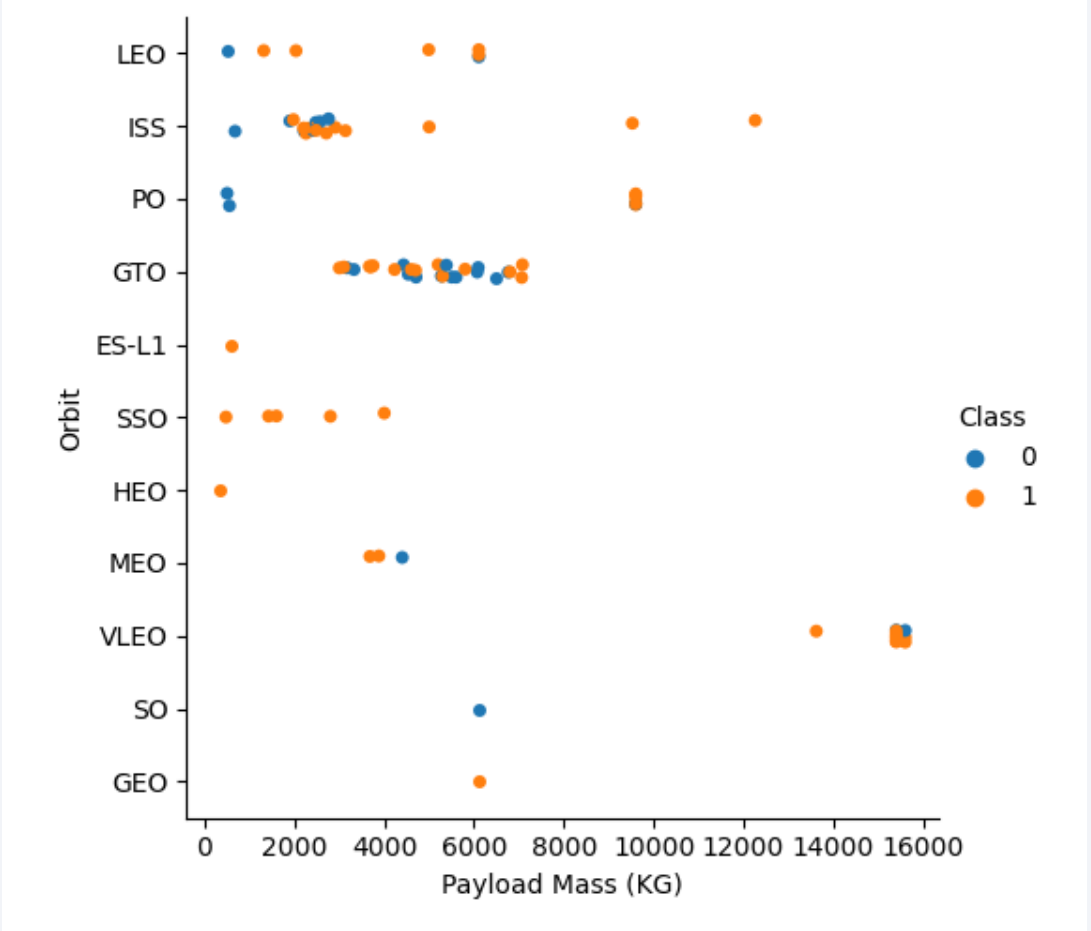
Flight Number vs. Orbit Type

- Flight number vs. Orbit type
- Observation
  - This plot reveals ES-L1, GEO, HEO have high success rates but from a limited number of flights whereas SSO has a high success rate from multiple flights.
  - The LEO orbit has a few failures from its early attempts but has consecutive success after that.
  - LEO, ISS, PO, GTO are the orbits that SpaceX experimented the flights from the outset. SpaceX expanded the exploration to other orbits as they increased the number of flights.



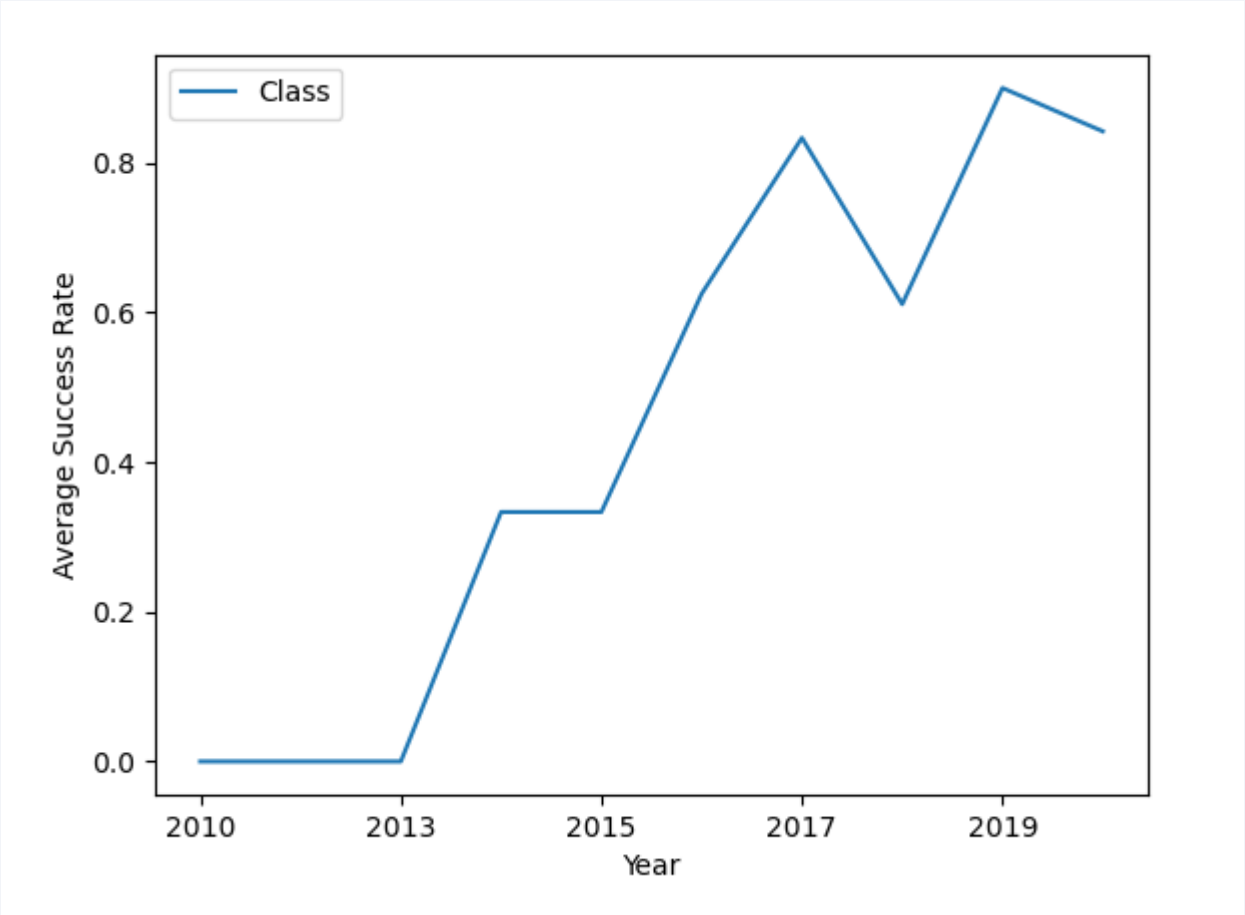
Payload vs. Orbit Type

- Payload Mass vs. Orbit type
- Observation
  - For PO, LEO, and ISS, success landing rates are higher with heavy payloads.
  - GTO does not show a clear pattern between success for failure against Payload Mass.



Launch Success Yearly Trend

- Yearly average success rate
- Observation
  - Since 2013, the success rates have generally been improved as time went on.
  - There was slight drops in 2018 and 2020.



All Launch Site Names

- Find the names of the unique launch sites

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

In [7]:
```
%%sql

SELECT DISTINCT("Launch_Site")
FROM SPACEXTBL;
```

```
 * sqlite:///my_data1.db
Done.
```

Out[7]:

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

- There are four launch sites from the SPACEXTBL table.

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`

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

In [8]:
```
%%sql

SELECT *
FROM SPACEXTBL
WHERE "Launch_Site" LIKE 'CCA%'
LIMIT 5;
```

```
 * sqlite:///my_data1.db
Done.
```

Out[8]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	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)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

- Queried launch sites staring with ‘CCA’ using LIKE clause

Total Payload Mass

- Calculate the total payload carried by boosters from NASA (CRS)

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

In [9]:
```
%%sql

SELECT Customer, SUM("PAYLOAD_MASS__KG_") AS Total_Payload_Mass
FROM SPACEXTBL
GROUP BY Customer
Having Customer = 'NASA (CRS)';
```

```
 * sqlite:///my_data1.db
Done.
```

Out[9]:

Customer	Total_Payload_Mass
NASA (CRS)	45596

- Used GROUP BY and HAVING clauses and the aggregate function SUM to get the data

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1

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

In [10]:
```
%%sql

SELECT AVG("PAYLOAD_MASS__KG_")
FROM SPACEXTBL
WHERE "Booster_Version" LIKE 'F9 v1.1%';
```

```
 * sqlite:///my_data1.db
Done.
```

Out[10]:

AVG("PAYLOAD_MASS__KG_")
2534.6666666666665

- Used LIKE clause and the aggregate function AVG to get the data

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad

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

Hint:Use min function

In [25]:
```
%%sql

SELECT Date
FROM SPACEXTBL
WHERE "Landing _Outcome" = 'Success (ground pad)'
ORDER BY SUBSTR(Date,7,4), SUBSTR(Date,4,2), SUBSTR(Date,1,2)
LIMIT 1;
```

```
 * sqlite:///my_data1.db
Done.
```

Out[25]:

Date
22-12-2015

- Used ORDER BY and LIMIT clauses to get the earliest date record from the result set.
- Used SUBSTR to parse the date for ordering by year, month, and date

Successful Drone Ship Landing with Payload between 4000 and 6000

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

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

In [15]:
```
%%sql

SELECT Booster_Version
FROM SPACEXTBL
WHERE "Landing _Outcome" = 'Success (drone ship)'
    AND ("PAYLOAD_MASS__KG_" > 4000 AND "PAYLOAD_MASS__KG_" < 6000);
```

```
 * sqlite:///my_data1.db
Done.
```

Out[15]:

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

- Used two conditions, Landing Outcome and Payload Mass, in WHERE clause to get the Booster names. There are four Booster Versions that meet the conditions

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes

List the total number of successful and failure mission outcomes

In [23]:
```
%%sql

SELECT SUBSTR("Mission_Outcome", 1, 7) AS Mission_Outcome, COUNT(*) AS Count
FROM SPACEXTBL
GROUP BY SUBSTR("Mission_Outcome", 1, 7);
```

```
 * sqlite:///my_data1.db
Done.
```

Out[23]:

Mission_Outcome	Count
Failure	1
Success	100

- Grouped the records by (SUBSTR of) Mission Outcome values and used COUNT function to get the number of success and failures

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass

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

In [24]:
```
%%sql

SELECT "Booster_Version"
FROM SPACEXTBL
WHERE "PAYLOAD_MASS__KG_" = (
    SELECT MAX("PAYLOAD_MASS__KG_") FROM SPACEXTBL);
```

```
 * sqlite:///my_data1.db
Done.
```

Out[24]:

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

- Used a sub-query to retrieve records that carried the maximum payload mass

2015 Launch Records

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

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.

In [26]:
```
%%sql

SELECT SUBSTR(Date, 4, 2) as Month, "Landing _Outcome", "Booster_Version", "Launch_Site"
FROM SPACEXTBL
WHERE SUBSTR(Date,7,4) = '2015' AND "Landing _Outcome" = 'Failure (drone ship)';
```

```
 * sqlite:///my_data1.db
Done.
```

Out[26]:

Month	Landing _Outcome	Booster_Version	Launch_Site
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

- Queried records that meet the conditions given for Date and Landing Outcome. Used SUBSTR to compare year part of Date column.

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

- Rank the count of successful landing outcomes between the date 2010-06-04 and 2017-03-20, in descending order

Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

In [27]:
```
%%sql

SELECT "Landing _Outcome", COUNT(*) as Success_Count
FROM SPACEXTBL
WHERE DATE BETWEEN '04-06-2010' AND '20-03-2017'
GROUP BY "Landing _Outcome"
HAVING "Landing _Outcome" LIKE 'Success%'
ORDER BY Success_Count DESC;
```

```
 * sqlite:///my_data1.db
Done.
```

Out[27]:

Landing _Outcome	Success_Count
Success	20
Success (drone ship)	8
Success (ground pad)	6

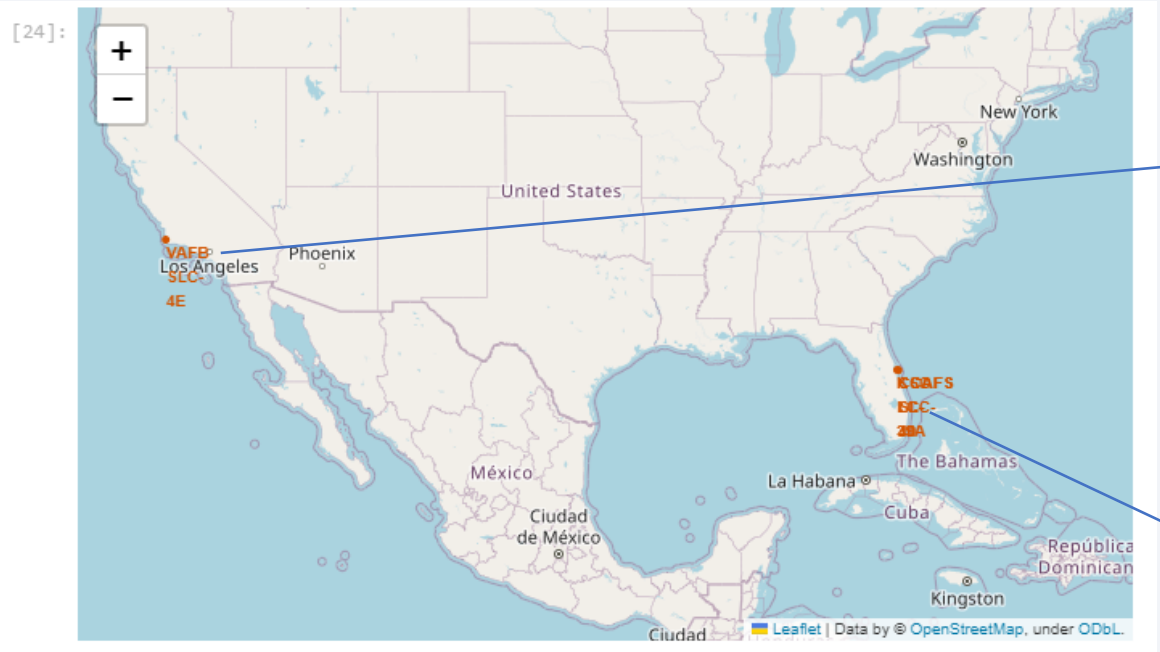
- Used GROUP BY and HAVING clauses to group the data
- Used BETWEEN clause to confine the range of dates
- Used DESC clause to order the results in descending order by success counts

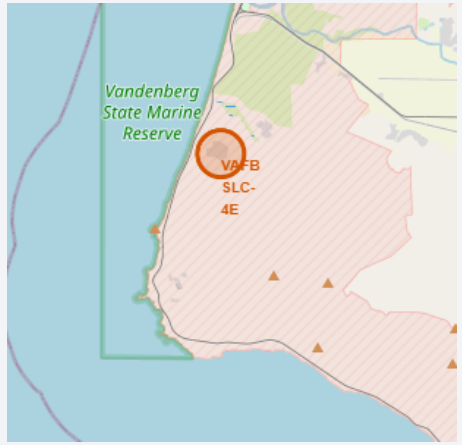
Section 3

Launch Sites Proximities Analysis

Launch Sites Locations

- Explore the generated folium map and make a proper screenshot to include all launch sites' location markers on a global map

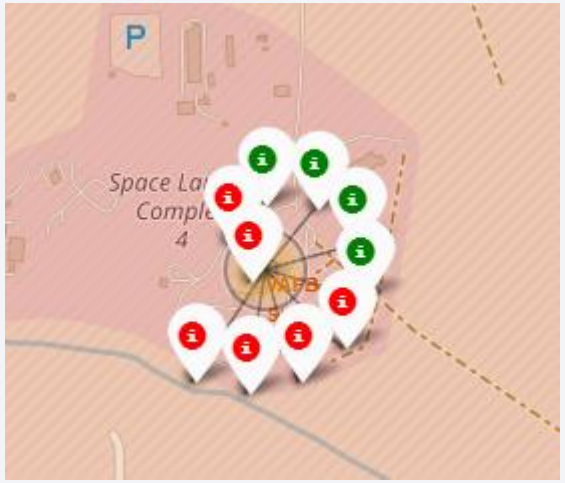




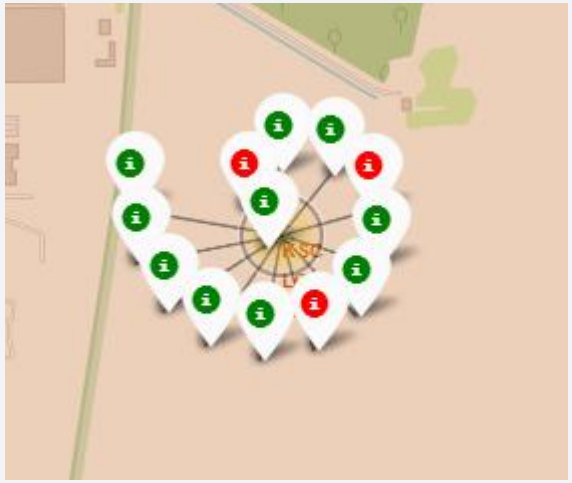
- There are 4 launch sites: one in west coast, three in east coast.
- They are all close to coastlines.
- CCAFS SLC-40 and CCAFS LC-40 are close each other.

Launch Outcomes at Launch Sites

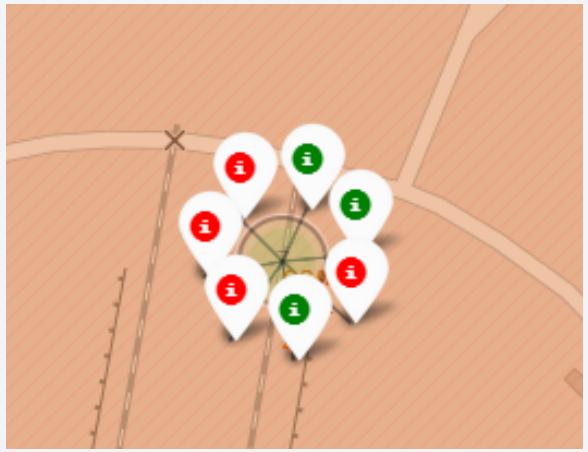
- Explore the folium map and make a proper screenshot to show the color-labeled launch outcomes on the map



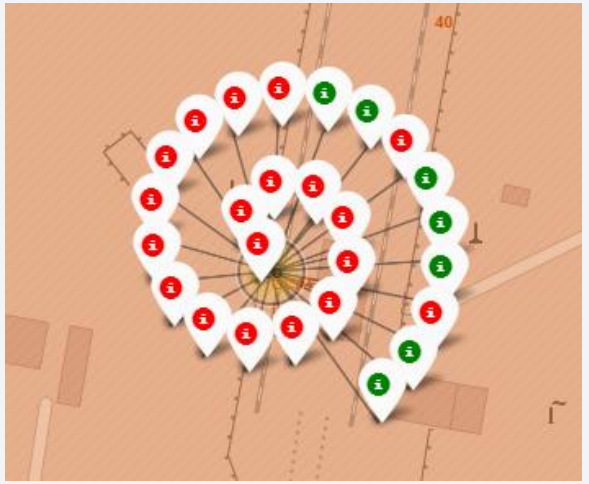
VAFB SLC-4E



KSC LC-39A



CCAFS SLC-40

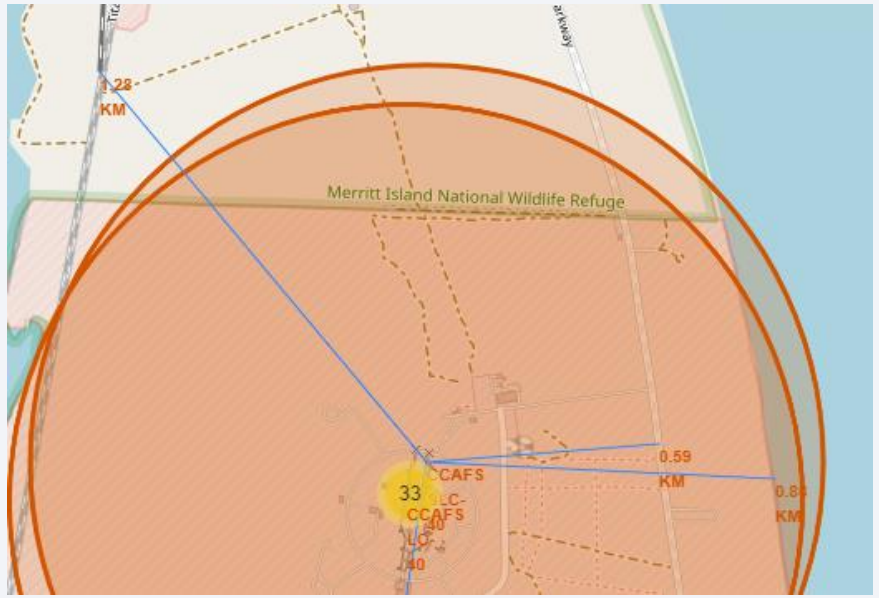


CCAFS LC-40

- CCAFS LC-40 has the largest number of launches.
- KSC LC-39A has the highest success rate.

Distance to proximities

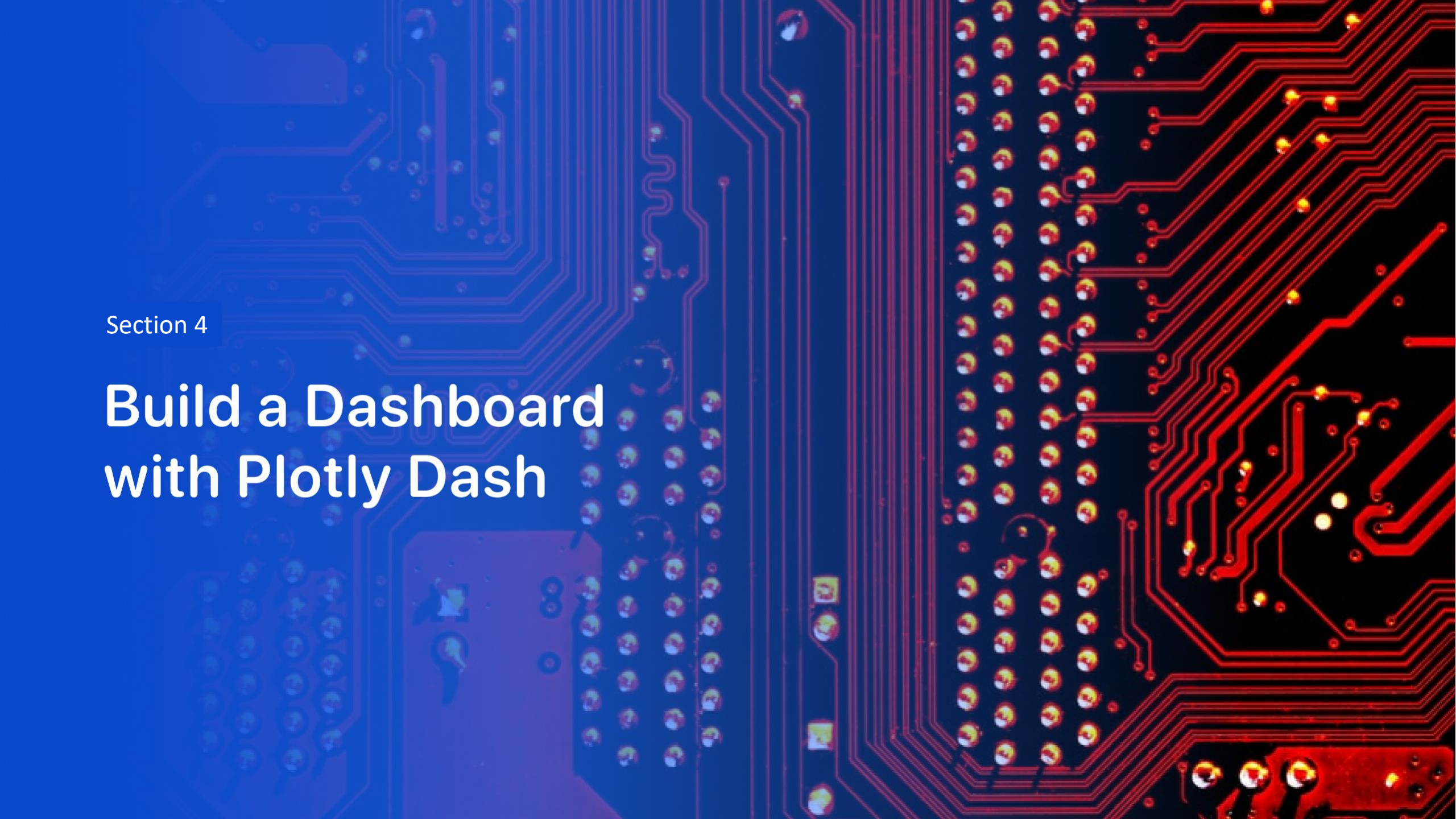
- Explore the generated folium map and show the screenshot of a selected launch site to its proximities such as railway, highway, coastline, with distance calculated and displayed



CCAFS SLC-40 is close to railway, highway, and coastline with the distances, 1.28Km, 0.59Km, and 0.88Km, respectively.

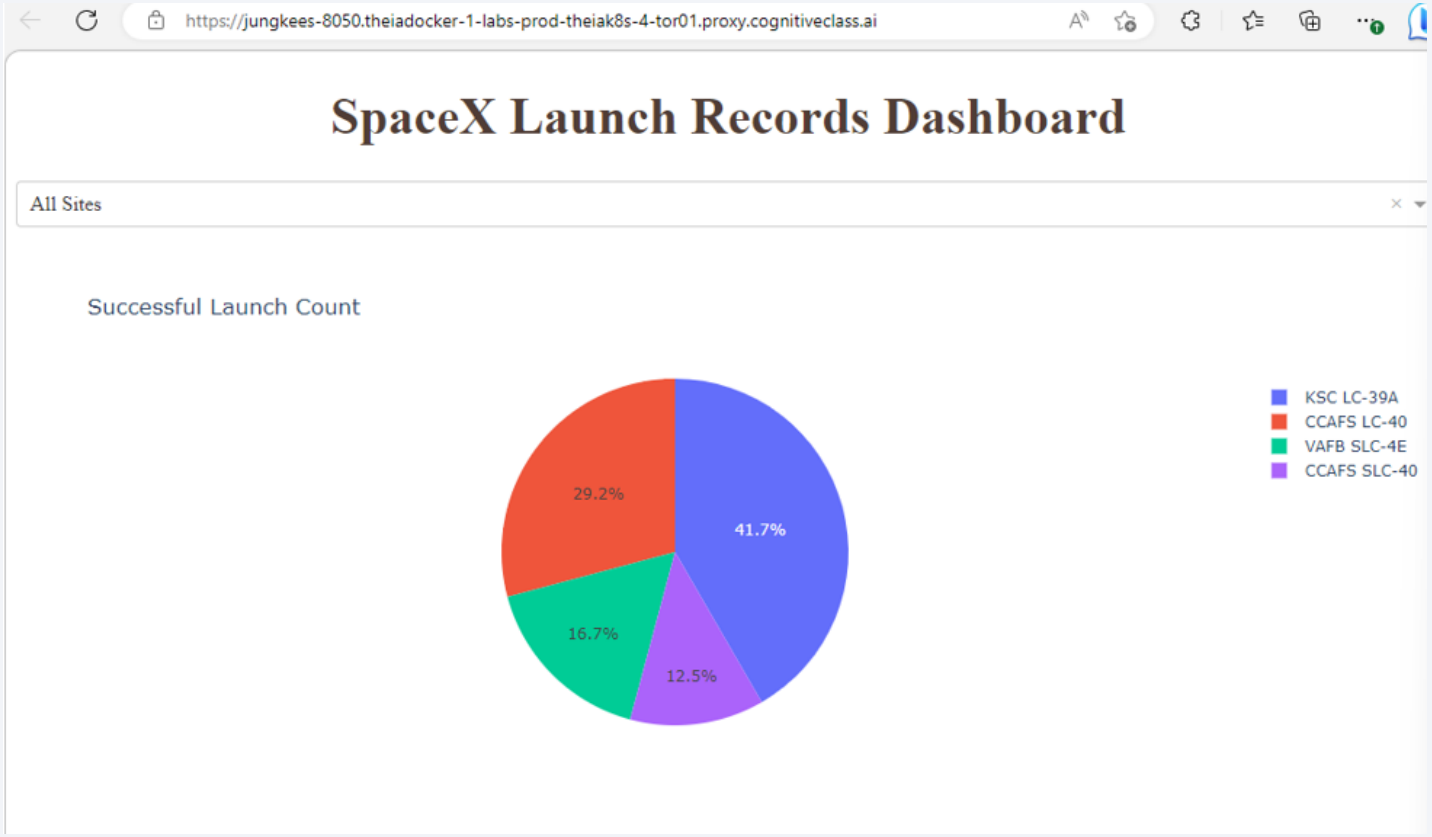
CCAFS SLC-40 is not so much close to a big city. Cape Canaveral is the nearest city, which is 18.14 Km away.

This pattern holds for all the launch sites.


Section 4
Build a Dashboard with Plotly Dash

Successful Launches for all Sites

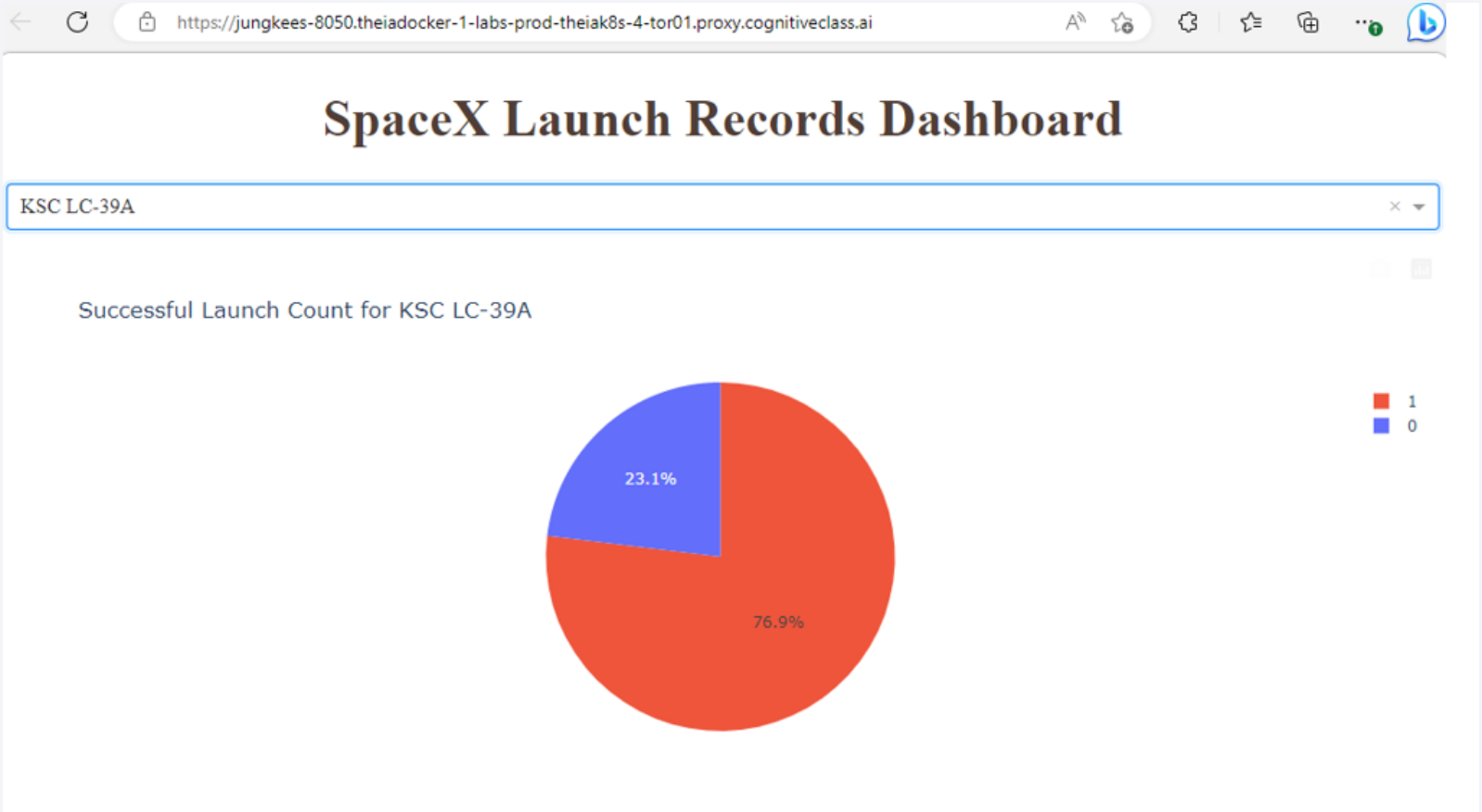
- Show the screenshot of launch success count for all sites, in a piechart



- KSC LC-39A has the most success count which accounts for 41.7% of success counts from all sites.

Highest Launch Success Rates

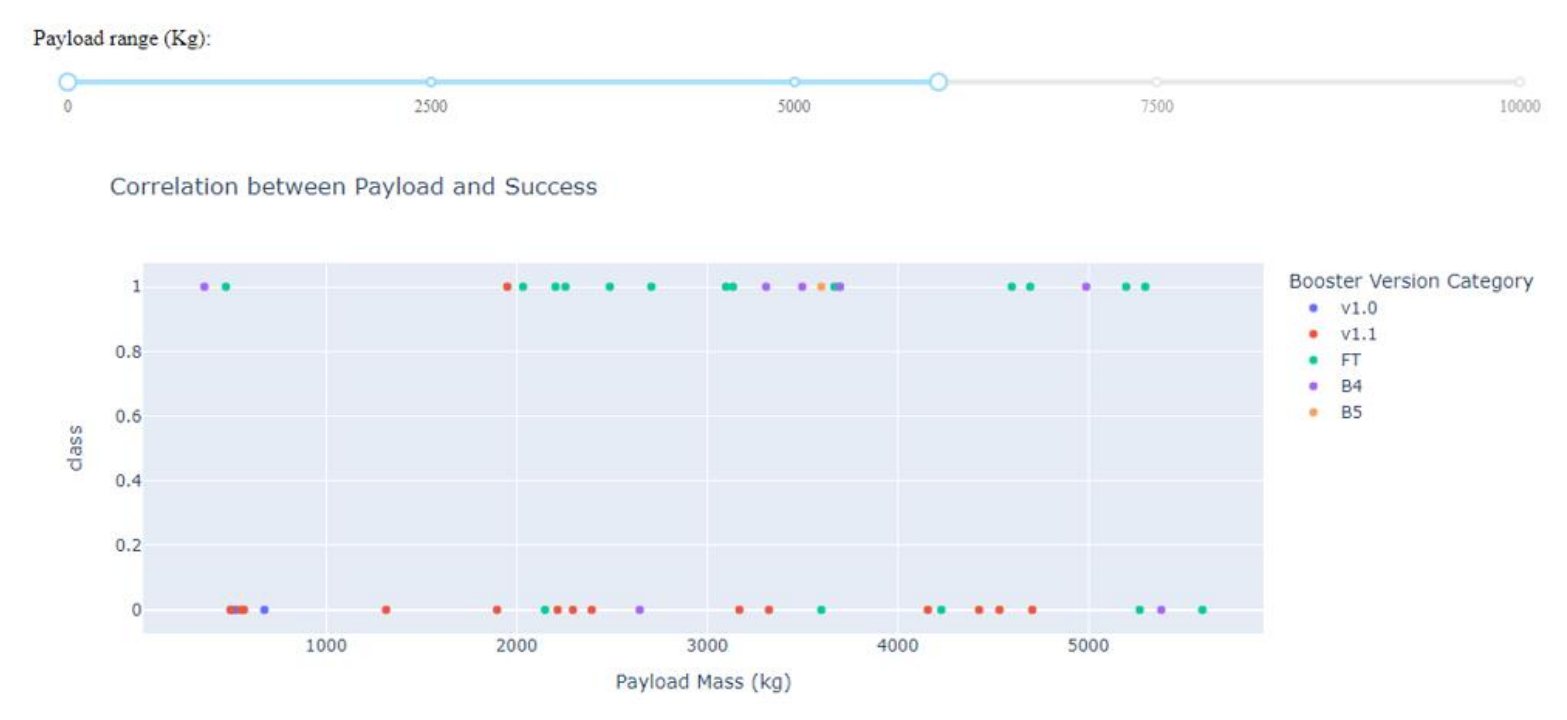
- Show the screenshot of the piechart for the launch site with highest launch success ratio



- KSC LC-39A has the highest launch success rates of 76.9%.

Correlation between Payload Mass and Success

- Show screenshots of Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider



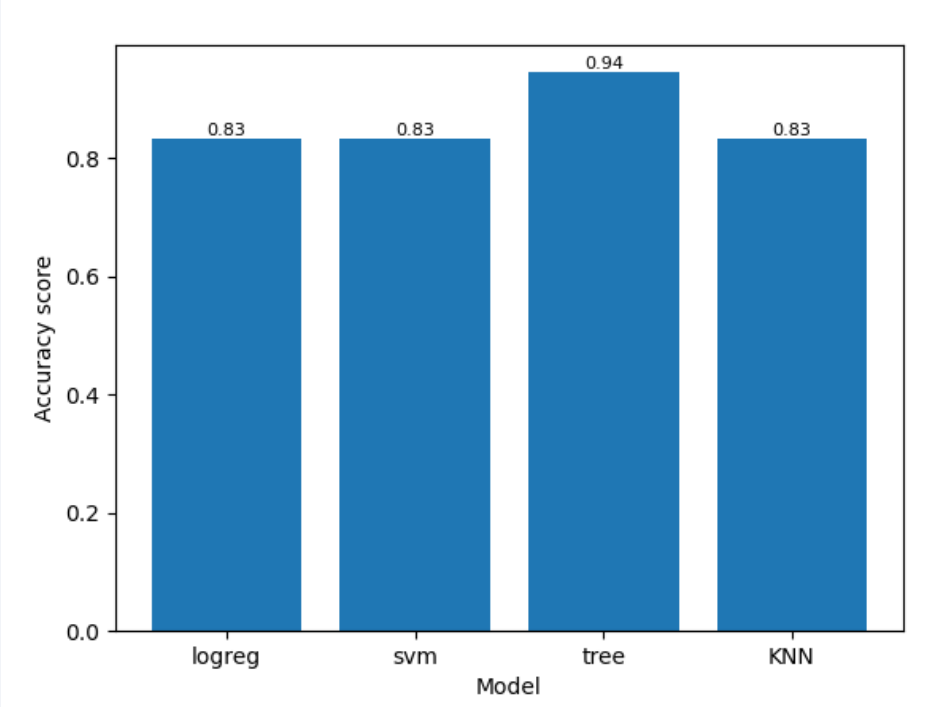
- Launches with the payload mass of 2,000 to 5,500 Kg range have the highest success rates. Especially the launches with the FT booster show the best success rate.

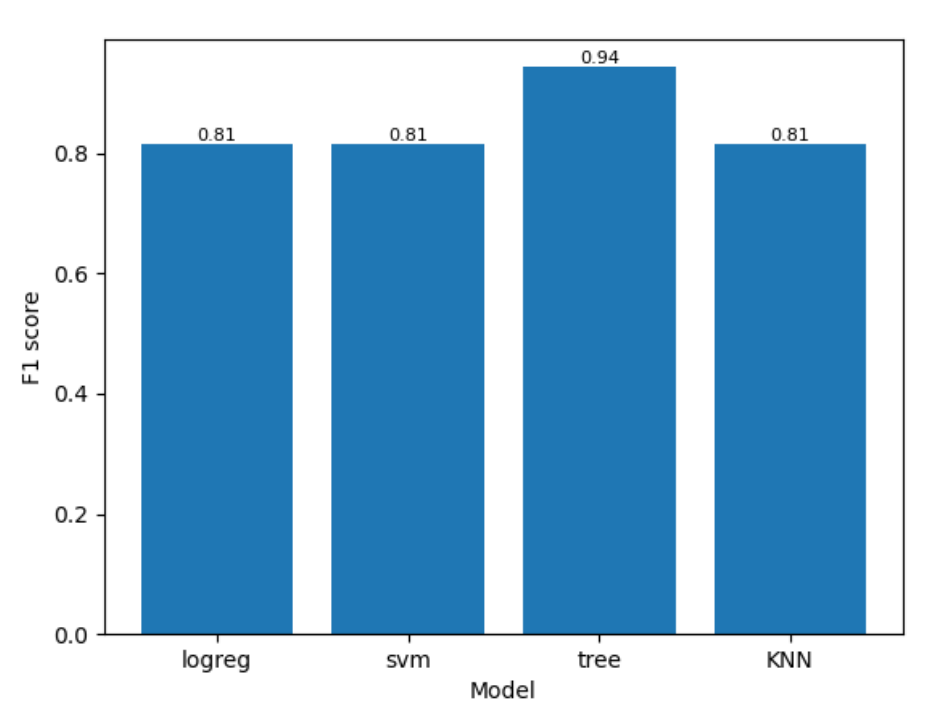
Section 5

Predictive Analysis (Classification)

Classification Accuracy

- Visualize the built model accuracy for all built classification models, in a bar chart

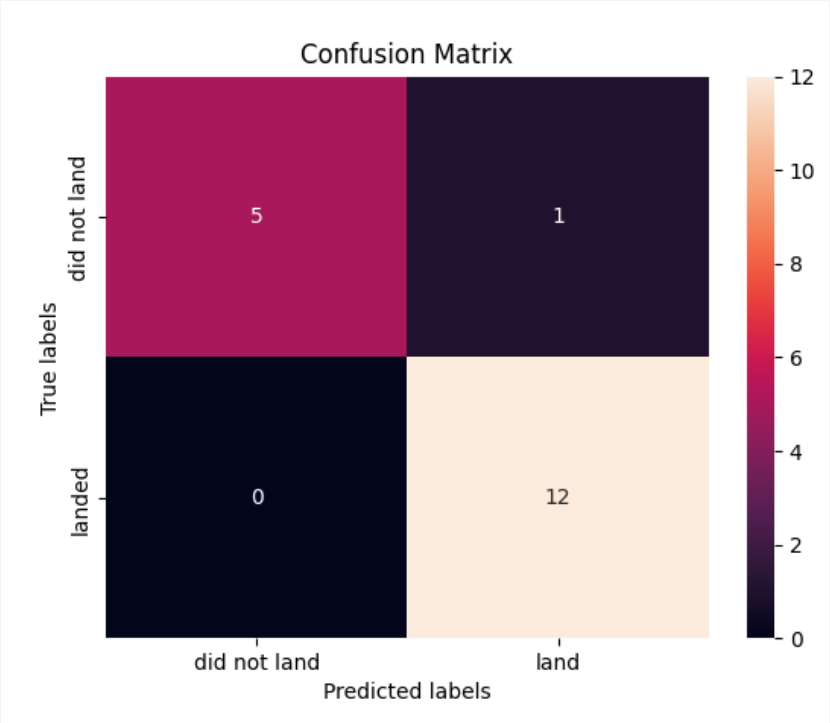




- The Decision Tree Classifier has the best accuracy score and the F1 score among all the evaluated models

Confusion Matrix

- Show the confusion matrix of the best performing model (Decision Tree) with an explanation



- For failure cases: Precision (100%), Recall (83.33%)
- For success cases: Precision (92.31%), Recall (100%)
- This matrix for Decision Tree resulted in the F1 score of 0.94, which is the best among evaluated models.

Conclusions

- The Decision Tree model trained with the selected features has a strong predictive performance. F1 Score: 0.94 from the test datasets (out of sample data).
- We can predict the outcomes of new launches based on the model, which can help determine the cost of a launch.
- The prediction results can help stakeholders make grounded decisions.
- We have future work.
  - Expand the train/evaluation scope to the most recent datasets
  - Decision Tree resulted in some varying accuracy scores across training cycles. We should investigate what caused the variation and whether that's normal with Decision Tree.
  - Try to train some Deep Learning models if we can achieve better performance.
  - Track the performance with out of sample data and iterate the process.

Appendix

- Folium code that adds Circles and Markers for launch sites

[25]:
```
# Initial the map
site_map = folium.Map(location=nasa_coordinate, zoom_start=4)
# For each launch site, add a Circle object based on its coordinate (Lat, Long) values.
# In addition, add Launch site name as a popup label
sites = []
for i, row in launch_sites_df.iterrows():
    location = [row[launch_sites_df.columns[1]], row[launch_sites_df.columns[2]]]
    site_name = row[launch_sites_df.columns[0]]
    # print('location: {}, site_name: {}'.format(location, site_name))
    circle = folium.Circle(location, radius=1000, color='#d35400', fill=True) \
        .add_child(folium.Popup(site_name))
    marker = folium.map.Marker(location, icon=DivIcon(icon_size=(20,20),icon_anchor=(0,0), \
             html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % site_name, ))
    sites.append((circle, marker))

#Launch Site 1
site_map.add_child(sites[0][0])
site_map.add_child(sites[0][1])
#Launch Site 2
site_map.add_child(sites[1][0])
site_map.add_child(sites[1][1])
#Launch Site 3
site_map.add_child(sites[2][0])
site_map.add_child(sites[2][1])
#Launch Site 4
site_map.add_child(sites[3][0])
site_map.add_child(sites[3][1])
```

[25]:

Appendix

- Folium code that adds Markers to indicate success/failures for a launch site

[12]:
```
# Add marker_cluster to current site_map
site_map.add_child(marker_cluster)

# for each row in spacex_df data frame
# create a Marker object with its coordinate
# and customize the Marker's icon property to indicate if this launch was successed or failed,
# e.g., icon=folium.Icon(color='white', icon_color=row['marker_color']
for index, record in spacex_df.iterrows():
    # TODO: Create and add a Marker cluster to the site map
    marker = folium.Marker(
        location=[record['Lat'], record['Long']],
        icon=folium.Icon(color='white', icon_color=record['marker_color'])
        )
    marker_cluster.add_child(marker)

site_map
```

[12]:

Thank you!