IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Petker, Andreas

17.02.2023

Outline

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

Executive Summary

- Methodology
  - The methodology for this report involves data collection using the SpaceX API and web scraping, followed by data wrangling and cleaning.
  - The data is then analyzed through exploratory data analysis using visualization and SQL, interactive visual analytics using Folium and Plotly Dash, and predictive analysis using classification models.
- Summary Results
  - This report presents the results of an Exploratory Data Analysis of SpaceX launch data (landing of SpaceX Falcon 9 first stage).
  - The analysis suggests that Launch Site CCAFS SLC 40 has a higher variability in launch outcomes compared to other launch sites with a lower proportion of unsuccessful launches.
  - The success rate varies for different orbits, with GEO, SO, and VLEO having the highest success rate and ES-L1, HEO, LEO, and MEO having the lowest.
  - The success rate in LEO orbit appears to be related to the number of flights.
  - The successful landing or positive landing rate is higher for Polar, LEO, and ISS when heavy payloads are involved.
  - The launch success rate has been increasing steadily since 2013 until 2020.
  - The report also describes the use of various tools, including a SpaceX database table, a global map of rocket launch sites, a Plotly Dash dashboard, and a model accuracy analysis using bar plots and a confusion matrix.
  - The LOGREG and SVM models performed well in classifying landing occurrence.
  - Overall, the report provides valuable insights into SpaceX launch data that can be used for future research and decision-making

Introduction

The space industry has seen significant developments in recent years, with companies such as SpaceX revolutionizing the way we approach space exploration. SpaceX has become a major player in the market by offering rocket launches at a lower cost than other providers, in part because they can reuse the first stage of their Falcon 9 rocket. The cost of a launch depends on the success of the first stage landing, which can provide valuable information for companies looking to bid against SpaceX. In this report, we present the results of an in-depth analysis of SpaceX launch data, including landing outcomes of the first stage of Falcon 9 rockets. The report follows a methodology that involves data collection using the SpaceX API and web scraping, followed by data wrangling, cleaning, and exploratory analysis using various tools, including visualization and SQL. The report provides insights into launch success rates, the variability of launch outcomes across different launch sites, and the impact of payload mass on landing success. Additionally, the report evaluates the accuracy of different classification models for landing occurrence. The findings of this report can be used for future research and decision-making in the space industry.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - For this report, the data collection methodology involved using the SpaceX API [1] and web scraping. The SpaceX API provided detailed information on SpaceX's launch history, rocket specifications, and mission outcomes, which were collected using Python scripts. In addition, data was extracted from a Wikipedia article [2] related to the topic of the report.
- Perform data wrangling
  - The report will involve data wrangling and cleaning, including the use of one-hot encoding, to prepare the data for further analysis. This process will enable machine learning algorithms to process the data, which typically require numerical input.

Methodology

Executive Summary

- Perform exploratory data analysis (EDA) using visualization and SQL
  - Visualization is a powerful tool that enables us to understand complex data in a simple and intuitive way. SQL is an essential tool for data analysis, allowing us to extract, manipulate, and summarize data quickly and efficiently.
- Perform interactive visual analytics using Folium and Plotly Dash
  - The report will present an interactive dashboard using Plotly Dash and Folium for visualizing launch site locations on a map, analyzing launch outcomes, and exploring the proximity of launch sites to various features such as railways, highways, coastlines, and cities.
- Perform predictive analysis using classification models
  - The process of training a classification model involves standardizing the data, splitting it into training and test sets, finding the best hyperparameters for different Machine Learning Models and then determining which method performs best by rating the prediction evaluating the model's accuracy.

Data Collection

- To answer the central question, relevant information on SpaceX launches was collected using the mentioned API [1]. The data was downloaded in json format.
- The data was decoded using the json() method (which is part of the Request interface). The data (a list with embedded json objects) was finally converted into a flat pandas.DataFrame using pandas.json_normalize() method.
- Missing information was reconstructed via API using identification number and various features. Different auxiliary functions were implemented to parse the data. The reconstructed data sets were combined into a dataset and written into a corresponding pandas.DataFrame for further processing/analysis.
- The data was filtered and completed for the Falcon 9 rocket (using the 'FlightNumber' column), and all other records were excluded. The dataset consists of 90 records.
- Missing values were replaced with the mean value.
- A useful source for this was the mentioned Wikipedia page [2], which contains a detailed table with information on Launch Site, Payload, Launch outcome, etc. This Data was acquired through web scraping and used for analysis. The obtained Information was parsed using the BeautifulSoup library and transferred into pandas.DataFrame. The dataset consists of 121 records.

Data Collection – SpaceX API

Requesting records using SpaceX-API and transferring data into flat pandas.Dataframe

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
[7] ✓ 0.5s

response = requests.get(spacex_url)
[8] ✓ 0.7s

# Use json_normalize method to convert the json result into a dataframe
data = pd.json_normalize(response.json())
[14] ✓ 0.8s
```

```
# Lets take a subset of our dataframe keeping only the features we want and the flight number,
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatyp
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]

data.head()
[16] ✓ 0.1s
```

Cleaning, parsing, reconstructing missing information

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

```
# Call getBoosterVersion
getBoosterVersion(data)
[19] ✓

# Call getLaunchSite
getLaunchSite(data)
[21] ✓

# Call getPayloadData
getPayloadData(data)
[22] ✓ 33.7s

# Call getCoreData
getCoreData(data)
[23] ✓ 33.7s

# Create a data from launch_dict
launch_df = pd.DataFrame.from_dict(launch_dict)
[28] ✓ 0.5s
```

Filtering for Falcon 9 records

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = launch_df[launch_df['BoosterVersion'] != 'Falcon 1']
[30] ✓ 0.7s
```

Handling missing values

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

# Replace the np.nan values with its mean value
data_falcon9.fillna(value=payloadMassMean, inplace=True)
[33] ✓ 0.2s
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude
4	1	2010-06-04	Falcon 9	6123.547647	LEO	CCSFS SLC 40	None None	1	False	False	False	6123.547647	1.0	0	B0003	-80.577366	28.561857
5	2	2012-05-22	Falcon 9	525.000000	LEO	CCSFS SLC 40	None None	1	False	False	False	6123.547647	1.0	0	B0005	-80.577366	28.561857
6	3	2013-03-01	Falcon 9	677.000000	ISS	CCSFS SLC 40	None None	1	False	False	False	6123.547647	1.0	0	B0007	-80.577366	28.561857
7	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	6123.547647	1.0	0	B1003	-120.610829	34.632093
8	5	2013-12-03	Falcon 9	3170.000000	GTO	CCSFS SLC 40	None None	1	False	False	False	6123.547647	1.0	0	B1004	-80.577366	28.561857

GitHub URL: https://github.com/IBMCourse-Petker/Applied-Data-Science-Capstone/blob/6df53fdb5bce82e428c65f6b0c806473d110d7b2/01_Data_Collection_API.ipynb

Data Collection - Scraping

GitHub URL: https://github.com/IBMCourse-Petker/Applied-Data-Science-Capstone/blob/6df53fdb5bce82e428c65f6b0c806473d110d7b2/02_Data_Collection_with_Web_Scraping.ipynb

```
static_url = "https://en.wikipedia.org/w/index.php?tit
[3] ✓ 0.7s
```

```
# use requests.get() method with the p
# assign the response to a object
response = requests.get(static_url)
response.status_code
✓ 0.3s
```

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.content, "html.parser")
✓ 1.7s
```

Requesting Falcon 9 Launch Wiki page from its URL using BeautifulSoup

```
# Use the find_all function in the BeautifulSoup obj
# Assign the result to a list called `html_tables`
html_tables = []
for table in soup.find_all('table'):
    html_tables.append(table)
[9]
```

```
# Let's print the third table and check
first_launch_table = html_tables[2]
print(first_launch_table)
[10] ✓ 0.5s
```

```
# Apply find_all() function with `th` element on firs
# Iterate each th element and apply the provided extr
# Append the Non-empty column name (`if name is not N

column_names = []

for name in first_launch_table.find_all('th'):
    name = extract_column_from_header(name)
    if name is not None:
        if len(name)>0:
            column_names.append(name)
[19] ✓ 0.9s
```

Column/variable names are extracted from an HTML table header. Third table is selected as the target table. Column names are extracted from its header using the find_all() function with th element, and the provided extract_column_from_header() function is applied to extract column names. Non-empty column names are appended into a list called 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
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
```

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('tab
    # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as num
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with k
            launch_dict['Flight No.'].append(flight_number)
            #print(len(flight_number))

            # Date value
            # TODO: Append the date into launch_dict with key `Date`
            datatimelist=date_time(row[0])
            date = datatimelist[0].strip(',')
            launch_dict['Date'].append(date)
            #print(len(date))

            # Time value
            # TODO: Append the time into launch_dict with key `Time`
```

```
df=pd.DataFrame(launch_dict)
df.head()
for key, val in launch_dict.items():
    print(f'{key}: {len(val)}')
df.head()
[15] ✓ 0.1s
```

Empty dictionary is created and initialized with empty lists for each key; new keys are added.

Dictionary launch_dict is filled with launch records that have been extracted from table rows. Tables and rows are iterated; specific elements are checked; values are extracted and then appended to the dictionary.

Finally dataframe is created from the launch_dict.

	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing		e
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success\n	F9 v1.0B0003.1	Failure	4 June 201	18:45
1	2	CCAFS	Dragon	0	LEO	NASA (COTS)\nNRO	Success	F9 v1.0B0004.1	Failure	8 December 2010	15:43
2	3	CCAFS	Dragon	525 kg	LEO	NASA (COTS)	Success	F9 v1.0B0005.1	No attempt\n	22 May 2012	07:44
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA (CRS)	Success\n	F9 v1.0B0006.1	No attempt	8 October 2012	00:35
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA (CRS)	Success\n	F9 v1.0B0007.1	No attempt\n	1 March 2013	15:10

+ Code + Markdown

Data Wrangling

GitHub URL: https://github.com/IBMCourse-Petker/Applied-Data-Science-Capstone/blob/6df53fdb5bce82e428c65f6b0c806473d110d7b2/03_EDA.ipynb

```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwor
df.head(10)
```

```
[3]
df.isnull().sum()/df.count()*100
```

```
[4]
df.dtypes
```

```
FlightNumber
Date
BoosterVersion
PayloadMass
Orbit
```

```
[8]
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()
```

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

```
[7]
# Apply value_counts on Orbit column
df['Orbit'].value_counts()
✓ 0.8s
```

```
[8]
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
✓ 0.1s
```

```
[12]
for i,outcome in enumerate(landing_outcomes.keys()):
```

```
[15]
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
type(bad_outcomes)
✓ 0.1s
```

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = len(df['Outcome'])*[1]
for no_i, i in enumerate(list(df['Outcome'])):
    for k in list(bad_outcomes):
        if i in k:
            landing_class[no_i] = 0
✓ 0.6s
```

```
[14]
df["Class"].mean()
✓ 0.6s
0.6666666666666666
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.561857	0
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561857	0
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577366	28.561857	0
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003	-120.610829	34.632093	0
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004	-80.577366	28.561857	0

Reading in data set. Calculating the percentage of missing values in each attribute. Checking the data types of the columns and counting the number of launches on each launch site using the pandas method value_counts().

Counting the number and occurrence of each orbit type.

Counting the number and occurrence of mission outcomes per orbit type. The result is a series object containing the counts of each unique value.

Create a landing outcome label from the outcome column, where the value is 0 if the corresponding row in the outcome is in a set of bad outcomes, otherwise it's 1. Use this label to represent the classification variable that indicates the outcome of each launch. Calculate the success rate by computing the mean of this variable.

EDA with Data Visualization

- A series of visualizations and data manipulations were performed to analyze the relationship between launch success rates and various features, such as flight number, launch site, payload mass, and orbit type.
- Scatter plots, bar plots, and line charts were used to examine these relationships and identify patterns and correlations in the data.
- To make the data suitable for machine learning algorithms, categorical features were converted into a numerical format using one-hot encoding, and all resulting columns were cast to the float64 data type.
- These steps were taken to enable accurate predictions of future launch success rates based on the identified patterns and correlations in the data.
- GitHub URL: https://github.com/IBMCourse-Petker/Applied-Data-Science-Capstone/blob/6df53fdb5bce82e428c65f6b0c806473d110d7b2/04_EDA_with_Visualization.ipynb

EDA with SQL

- The given SQL queries involve extracting information from a SpaceX dataset.
- The queries involve selecting specific columns and filtering rows based on criteria such as launch site, payload mass, mission outcome, landing outcome, and dates.
- The results include unique launch sites, specific records, total payload mass, average payload mass, first successful landing outcome, names of boosters, total number of successful and failed missions, booster versions with maximum payload mass, failed landing outcomes in drone ship, and a ranked count of landing outcomes within a specified date range.
- The results provide insights into booster launches, successful landings, and mission outcomes over time.
- GitHub URL: https://github.com/IBMCourse-Petker/Applied-Data-Science-Capstone/blob/6df53fdb5bce82e428c65f6b0c806473d110d7b2/05_EDA_with_SQL.ipynb

Build an Interactive Map with Folium

- To visualize SpaceX launch sites, marking success/failed launches, and calculating distances to proximities Folium was used.
- This includes tasks such as marking all launch sites on a map, adding success/failed launches for each site on the map, and calculating distances between a launch site and its proximities.
- The aim is to find geographical patterns about launch sites and their location in relation to the Equator line, coastlines, railways, highways, cities, and success rates.
- The findings can be used to build a dashboard using Plotly Dash on detailed launch records.
- GitHub URL: https://github.com/IBMCourse-Petker/Applied-Data-Science-Capstone/blob/6df53fdb5bce82e428c65f6b0c806473d110d7b2/06_Interactive_Visual_Analytics_with_Folium.ipynb
- Rendered Jupyter Notebook on nbviewer.org: LINK

Build a Dashboard with Plotly Dash

- A dashboard includes two plots (pie chart and scatter chart) and interactions (dropdown list, range slider + callbacks) was created to explore the SpaceX launch data.
- The dashboard includes a dropdown list to select launch sites, a pie chart to show the total number of successful launches, a range slider to select payload range, and a scatter chart to show the correlation between payload and launch success.
- The pie chart and scatter chart are interactive and update based on the user's selection from the dropdown list and payload range using callbacks.
- The pie chart displays either the total successful launches count for all sites or the success vs. failed counts for a specific site.
- The scatter chart displays either the success launches for all sites within the payload range or the success launches for a specific site within the payload range. The scatter chart also includes the booster version category as a color-coded variable.
- GitHub URL: https://github.com/IBMCourse-Petker/Applied-Data-Science-Capstone/blob/bee2df9c377d449bc09a45b9648728bbf89df1d6/07_spacex_dash_app.py

Predictive Analysis (Classification)

- The predivtice Analysis includes the use of Python libraries such as pandas, numpy, matplotlib, seaborn, and scikit-learn to perform classification tasks on a given dataset.
- The dataset is split into training and testing data using the train_test_split function.
- The classification algorithms used in this report are logistic regression, support vector machines, decision trees, and k-nearest neighbors.
- The GridSearchCV function is used to fine-tune the hyperparameters for each algorithm, and the accuracy of each algorithm is evaluated on the test data.
- The best-performing method is determined based on the highest accuracy score.
- A confusion matrix is used to visualize the results of each classification algorithm.
- GitHub URL: https://github.com/IBMCourse-Petker/Applied-Data-Science-Capstone/blob/main/08_SpaceX_Machine_Learning_Prediction.ipynb

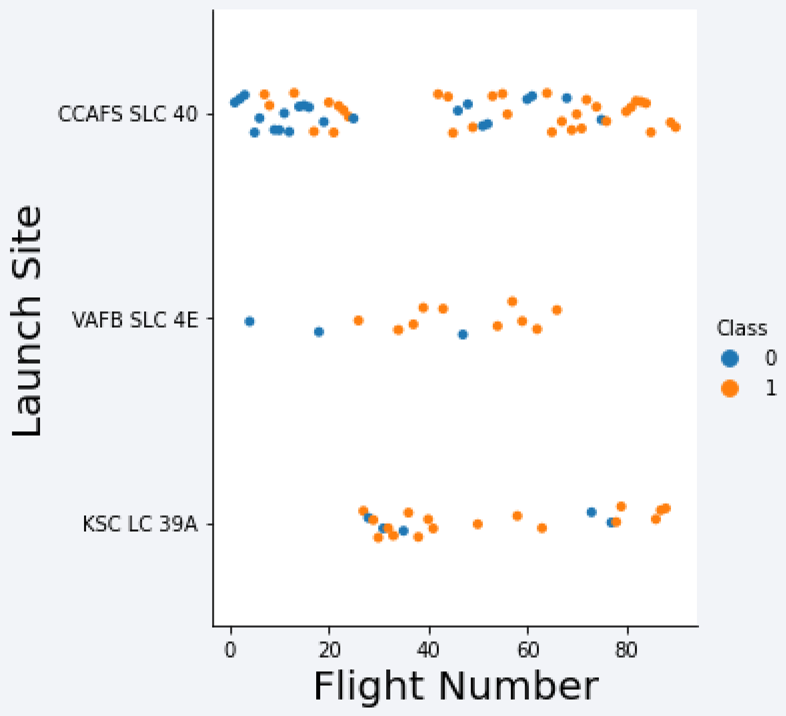
Results

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

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

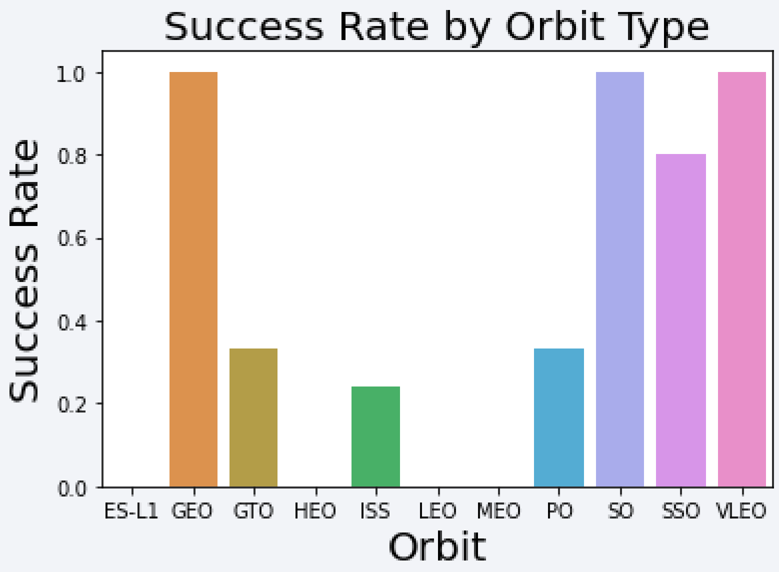


The pattern observed in the generated diagram suggests that for the Launch Site CCAFS SLC 40, both successful and unsuccessful launch outcomes occur with approximately equal frequency, whereas for the other Launch Sites, there are fewer instances of unsuccessful launches (Class 0). This could indicate that the Launch Site with an equal distribution of successful and unsuccessful launches is associated with higher variability in launch outcomes, while the Launch Sites with a lower proportion of unsuccessful launches may have better control over factors that could lead to mission failure.
Furthermore, the relationship between success and Flight Number appears to be non-random, with higher success rates observed as Flight Number increases. This trend is particularly evident for Launch Site CCAFS SLC 40.

Payload vs. Launch Site

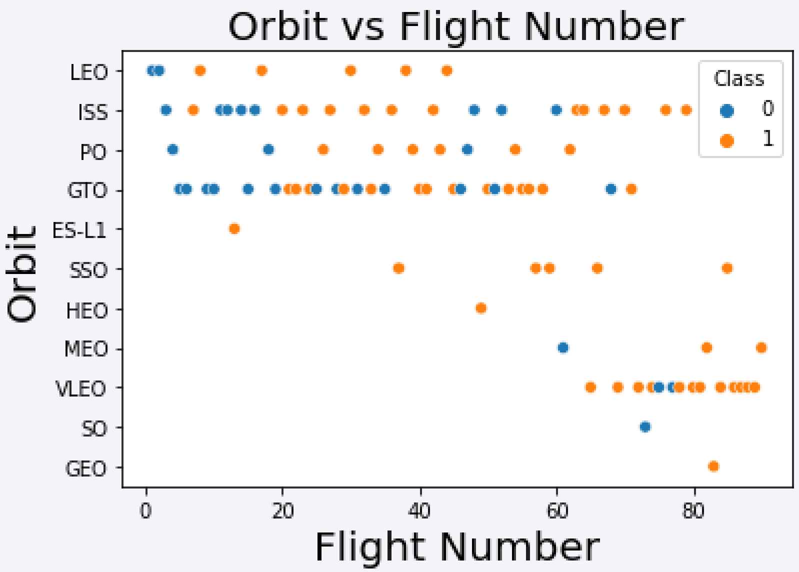
The plot shows that there were no rockets launched for heavy payload masses (greater than 10,000) at the VAFB-SLC launch site.

Success Rate vs. Orbit Type



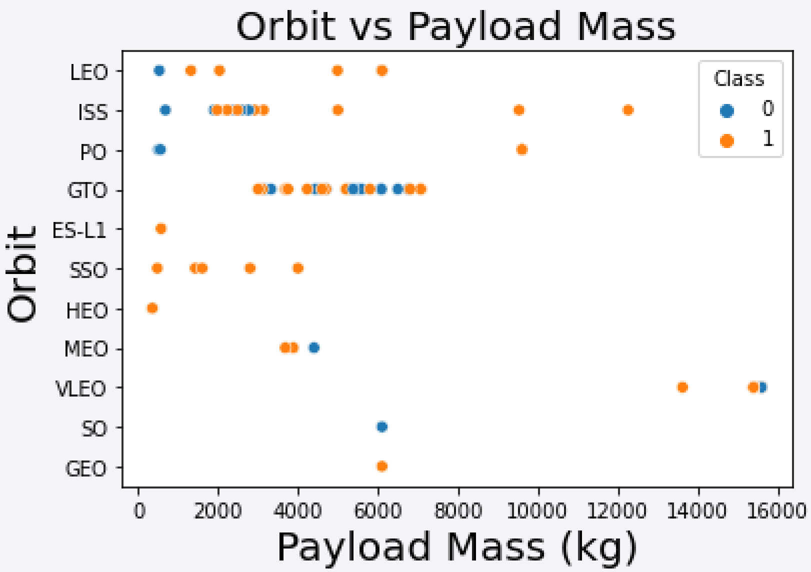
Orbits with highest success rate are GEO, SO and VLEO.
Orbits with lowest success rate are ES-L1, HEO, LEO, MEO.
Other orbits' success rate are between ~0.2-0.8

Flight Number vs. Orbit Type



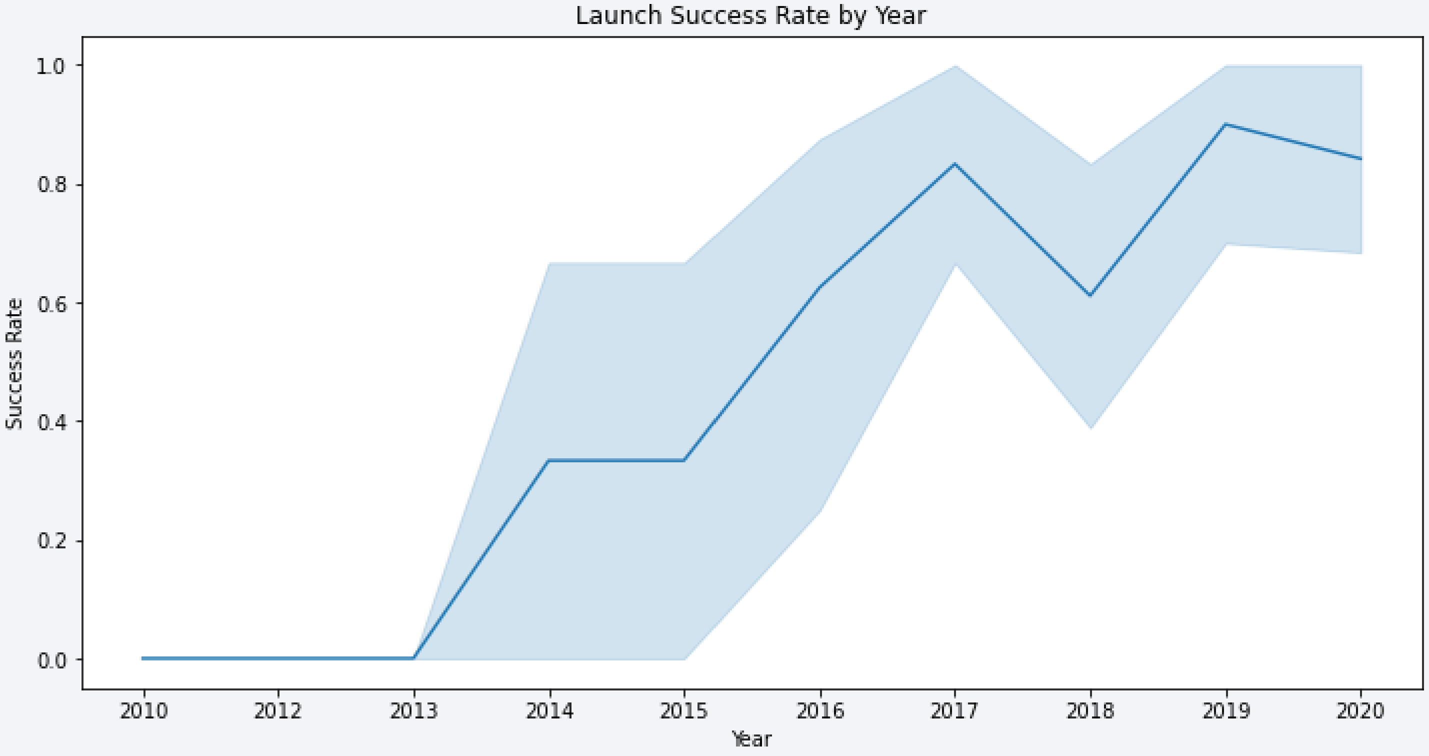
It can be observed that in the LEO orbit, success appears to be related to the number of flights, whereas no relationship between flight number and success is evident when in GTO orbit.

Payload vs. Orbit Type



It is observed that the successful landing or positive landing rate is higher for Polar, LEO, and ISS when heavy payloads are involved. However, in GTO, it is difficult to distinguish between successful and unsuccessful missions as both positive and negative landing rates are present.

Launch Success Yearly Trend



The resulting plot shows that the launch success rate has been increasing steadily since 2013 until 2020, indicating an improvement in the overall success rate of launches during this period.

All Launch Site Names

```
result1 = %sql SELECT DISTINCT LAUNCH_SITE FROM SPACEX
result1
```

[48] ✓ 0.3s

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

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

Retrieving a list of unique launch sites from a SpaceX database

Launch Site Names Begin with 'CCA'

```
result2 = %sql SELECT * FROM SPACEX WHERE LAUNCH_SITE LIKE '%CCA%' LIMIT 5
result2
```

[89] ✓ 0.5s Python

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

DATE	TIME	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

Retrieving information about the first 5 SpaceX launches that occurred at a launch site with the substring "CCA" in its name. The results will include all columns of the SPACEX table for those launches. The results is limited to the first 5 records that match the previous conditions.

Total Payload Mass

```
result3 = %sql SELECT SUM(PAYLOAD_MASS_KG) FROM SPACEX WHERE CUSTOMER = 'NASA (CRS)'
result3
```

[91] ✓ 0.3s

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

SUM(PAYLOAD_MASS_KG)
45596

Total sum of payload mass is retrieved by summing the "PAYLOAD_MASS_KG" column of the "SPACEX" table for records where the "CUSTOMER" column equals "NASA (CRS)".

Average Payload Mass by F9 v1.1

```
result4 = %sql SELECT AVG(PAYLOAD_MASS_KG) FROM SPACEX WHERE BOOSTER_VERSION LIKE '%F9 v1.1%'
result4
```

[92] ✓ 0.9s

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

AVG(PAYLOAD_MASS_KG)
2534.6666666666665

Retrieving the average value of the "PAYLOAD_MASS_KG" column from the "SPACEX" table, where the "BOOSTER_VERSION" column contains the string "F9 v1.1".

First Successful Ground Landing Date

```
result5 = %sql SELECT MIN(DATE) FROM SPACEX WHERE LANDING_OUTCOME LIKE 'Success (ground pad)'
result5
```

[94] ✓ 0.5s

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

MIN(DATE)
01-05-2017

Retrieving the earliest date from "DATE" column of the "SPACEX" table, where the “LANDING_OUTCOME" column is equal to the string "Success".

Successful Drone Ship Landing with Payload between 4000 and 6000

```
result6 = %sql SELECT BOOSTER_VERSION, PAYLOAD_MASS_KG FROM SPACEX WHERE LANDING_OUTCOME = 'Success (drone ship)' AND PAYLOAD_MASS_KG > 4000 AND PAYLOAD_MASS_KG < 6000
result6
```

[96] ✓ 0.1s

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

BOOSTER_VERSION	PAYLOAD_MASS_KG
F9 FT B1022	4696
F9 FT B1026	4600
F9 FT B1021.2	5300
F9 FT B1031.2	5200

Retrieving data from a table named "SPACEX" Selection of "BOOSTER_VERSION" and (in addition!) "PAYLOAD_MASS_KG" from the table. The data returned by the query satisfies three conditions:

1) LANDING_OUTCOME was equals to “Success (drone ship)“,
2) "PAYLOAD_MASS_KG" > 4000 and
3) "PAYLOAD_MASS_KG" is <6000.

Total Number of Successful and Failure Mission Outcomes

```
sucress = %sql SELECT COUNT(*) FROM SPACEX WHERE MISSION_OUTCOME LIKE '%Success%'
failure = %sql SELECT COUNT(*) FROM SPACEX WHERE MISSION_OUTCOME LIKE '%Failure%'
print(f'\nTask 7\n-------\nSuccessful Mission Outcome: {sucress[0][0]}\nUnsuccessful Mission Outcome: {failure[0][0]}')
```

[113] ✓ 0.4s

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

Task 7
-------
Successful Mission Outcome: 100
Unsuccessful Mission Outcome: 1
```

Queries to count the number of rows in the table where the "MISSION_OUTCOME" column contains the word "Success" and “Failure" in any part of the column value.

Boosters Carried Maximum Payload

```
result8 = %sql SELECT BOOSTER_VERSION, PAYLOAD_MASS_KG FROM SPACEX WHERE PAYLOAD_MASS_KG = (SELECT MAX(PAYLOAD_MASS_KG) FROM SPACEX WHERE BOOSTER_VERSION = SPACEX.BOOSTER_VERSION)
result8
```

✓ 0.5s Python

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

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

Retrieving the "BOOSTER_VERSION" (in addition the "PAYLOAD_MASS_KG") of each SpaceX mission that had the maximum payload mass for its corresponding booster version. The query uses a subquery that finds the maximum "PAYLOAD_MASS_KG" value for each unique "BOOSTER_VERSION"

2015 Launch Records

```
result9 = %sql SELECT LANDING_OUTCOME, BOOSTER_VERSION, LAUNCH_SITE FROM SPACEX WHERE DATE LIKE '%2015%' AND LANDING_OUTCOME = 'Failure (drone ship)'
result9
```

✓ 0.5s

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

LANDING_OUTCOME	BOOSTER_VERSION	LAUNCH_SITE
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

Retrieving data from table "SPACEX“. Selection of three columns, "LANDING_OUTCOME" "BOOSTER_VERSION" and "LAUNCH_SITE" from the table. The data returned by this query satisfies the following conditions:

1) The "DATE" column in the table must contain the year "2015“
2) The "LANDING_OUTCOME" column in the table must have the value "Failure (drone ship)"

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

```
success = %sql SELECT LANDING_OUTCOME, COUNT(LANDING_OUTCOME) FROM SPACEX\
    WHERE DATE BETWEEN '04-06-2010' AND '20-03-2017' \
        AND LANDING_OUTCOME LIKE '%Success%' \
    GROUP BY LANDING_OUTCOME ORDER BY DATE
success
```

✓ 0.1s

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

LANDING_OUTCOME	COUNT(LANDING_OUTCOME)
Success	20
Success (drone ship)	8
Success (ground pad)	6

```
failure = %sql SELECT LANDING_OUTCOME, COUNT(LANDING_OUTCOME) FROM SPACEX\
    WHERE DATE BETWEEN '04-06-2010' AND '20-03-2017' \
        AND LANDING_OUTCOME LIKE '%Failure%' \
    GROUP BY LANDING_OUTCOME ORDER BY DATE
failure
```

✓ 0.7s

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

LANDING_OUTCOME	COUNT(LANDING_OUTCOME)
Failure (parachute)	2
Failure	3
Failure (drone ship)	4

Retrieving data from a table "SPACEX". Seletion of the columns, "LANDING_OUTCOME" and the count of the "LANDING_OUTCOME", and groups them based on the "LANDING_OUTCOME" column. The data returned by the query satisfies the following conditions:

1) The "DATE" column in the table must be between the dates '04-06-2010' and '20-03-2017',
2) The "LANDING_OUTCOME" column in the table must contain the words "Success" or "Failure" in any part of the column value.

Section 3

Launch Sites Proximities Analysis

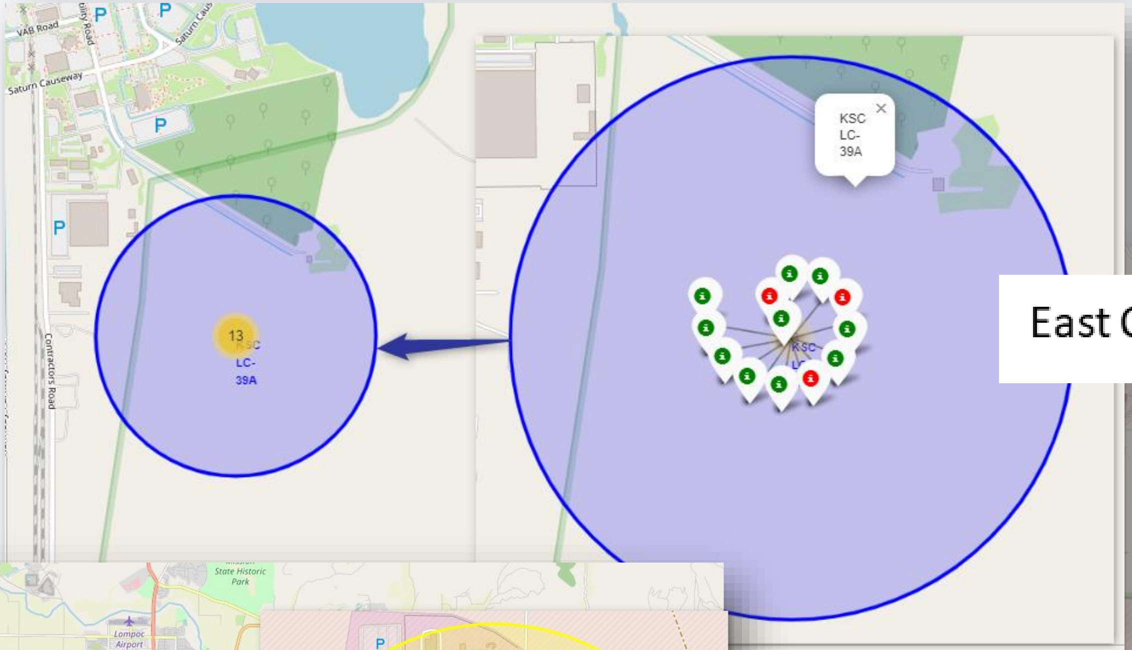
Global Map of Rocket Launch Sites

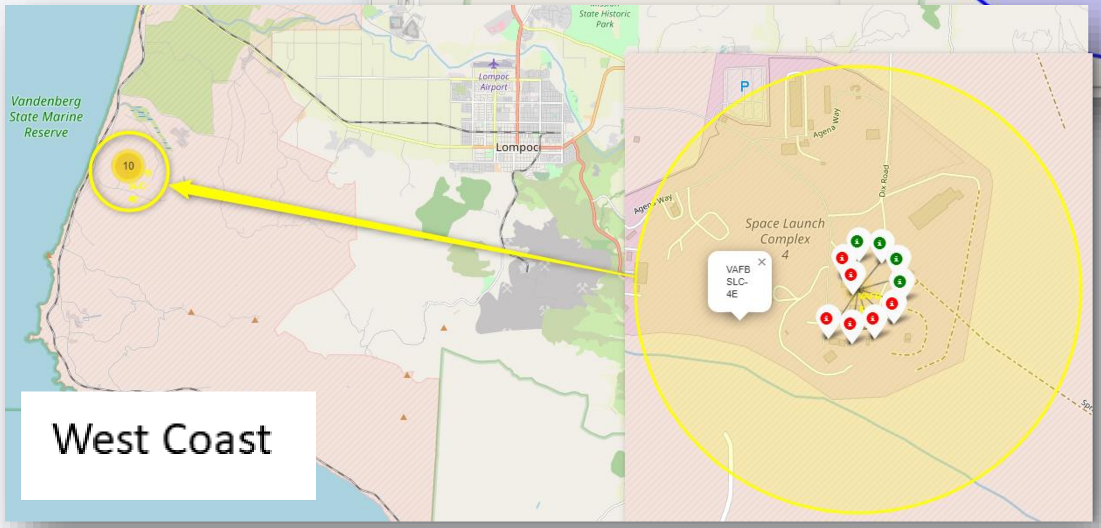
	Launch Site	Lat	Long
0	CCAFS LC-40	28.562302	-80.577356
1	CCAFS SLC-40	28.563197	-80.576820
2	KSC LC-39A	28.573255	-80.646895
3	VAFB SLC-4E	34.632834	-120.610745

The Launch Sites 'CCAFS LC-40', 'CCAFS SLC-40' and 'KSC LC-39A' are located on the East Coast in Florida (at Cape Canaveral and Kennedy Space Center), while 'VAFB SLC-4E' is located on the West Coast in California (at Vandenberg Air Force Base).

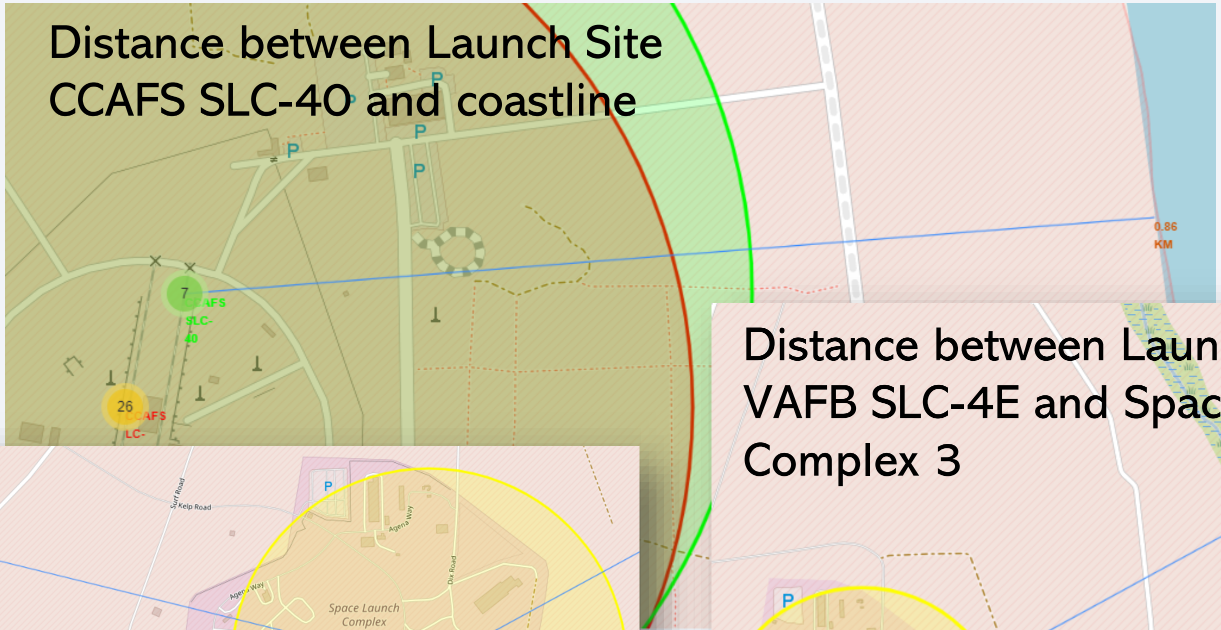
Visualizing Launch Success

Green Marker: Successful Launches
Red Marker: Failed Launches

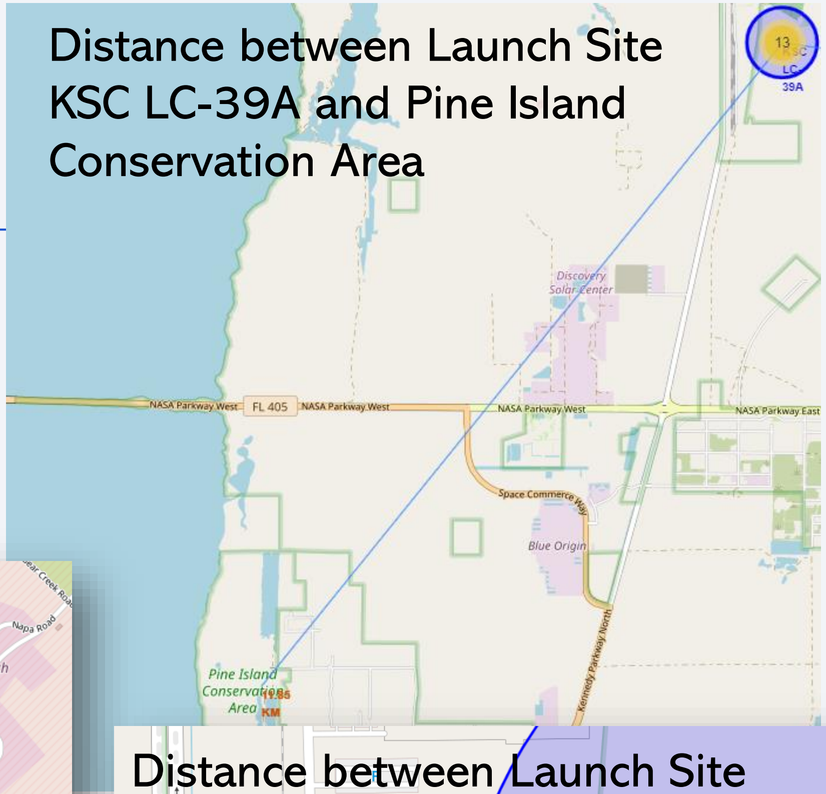




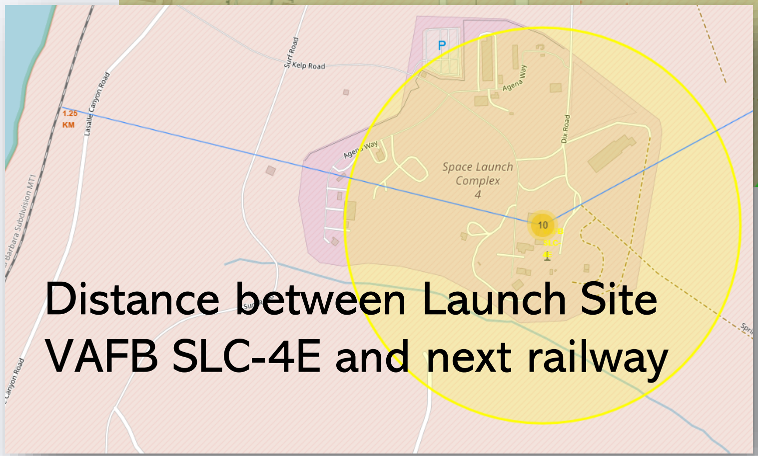
Exploring Proximity



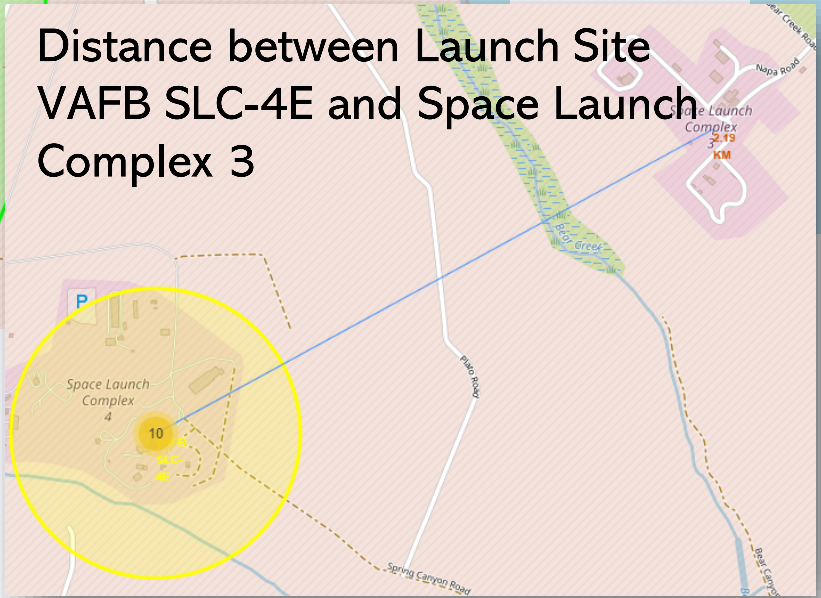
Distance between Launch Site CCAFS SLC-40 and coastline



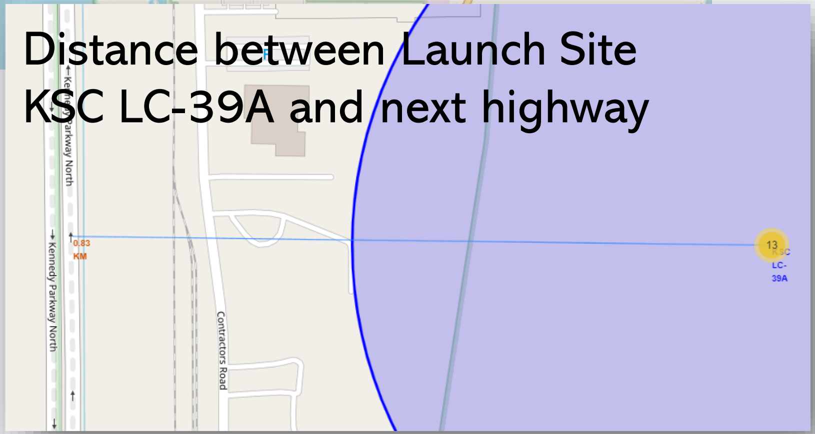
Distance between Launch Site KSC LC-39A and Pine Island Conservation Area



Distance between Launch Site VAFB SLC-4E and next railway



Distance between Launch Site VAFB SLC-4E and Space Launch Complex 3



Distance between Launch Site KSC LC-39A and next highway

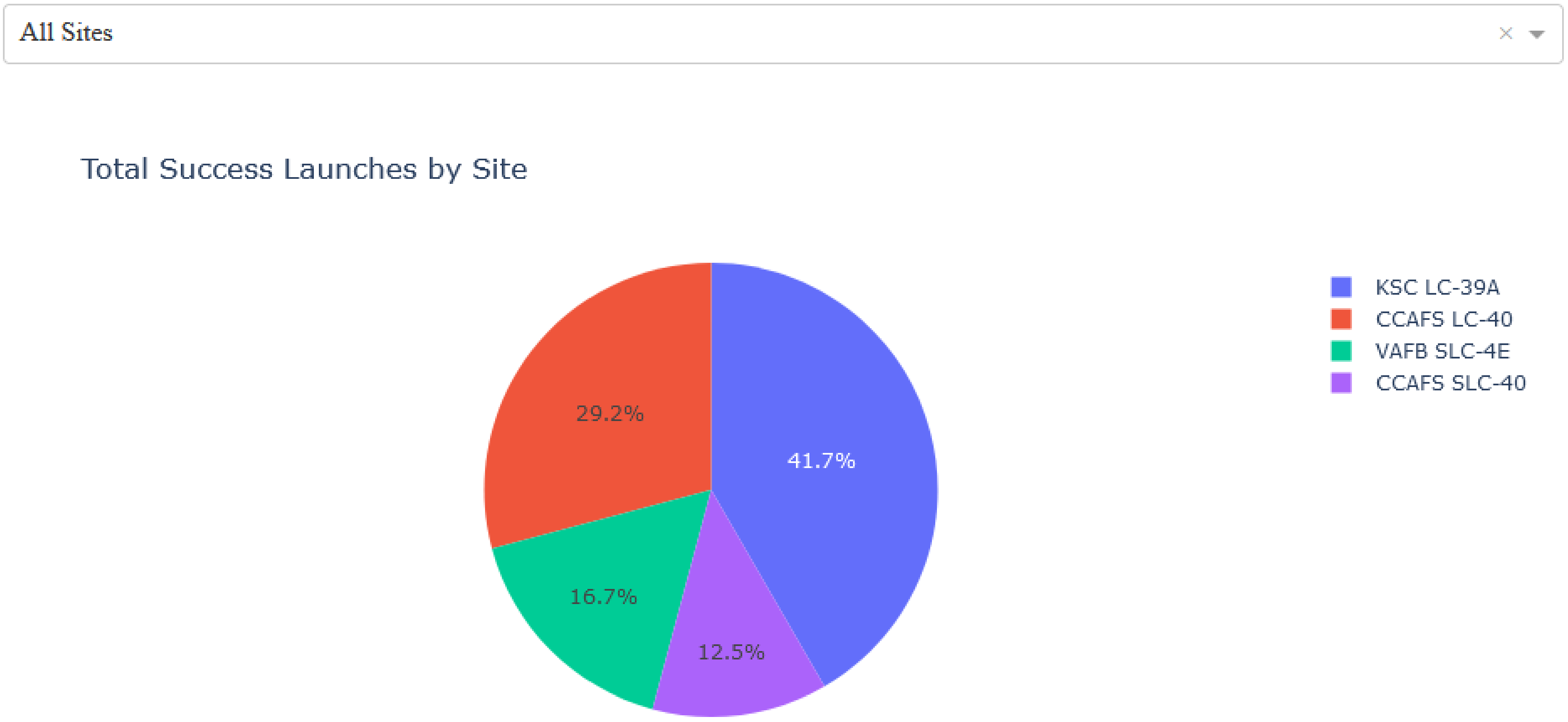
Obtaining coordinates using `MousePosition`. Mark down the coordinates of points of interest and calculate the distance to other points based on their latitude and longitude values.

[Jupyter Notebook Viewer (nbviewer.org)](https://nbviewer.org)

Section 4

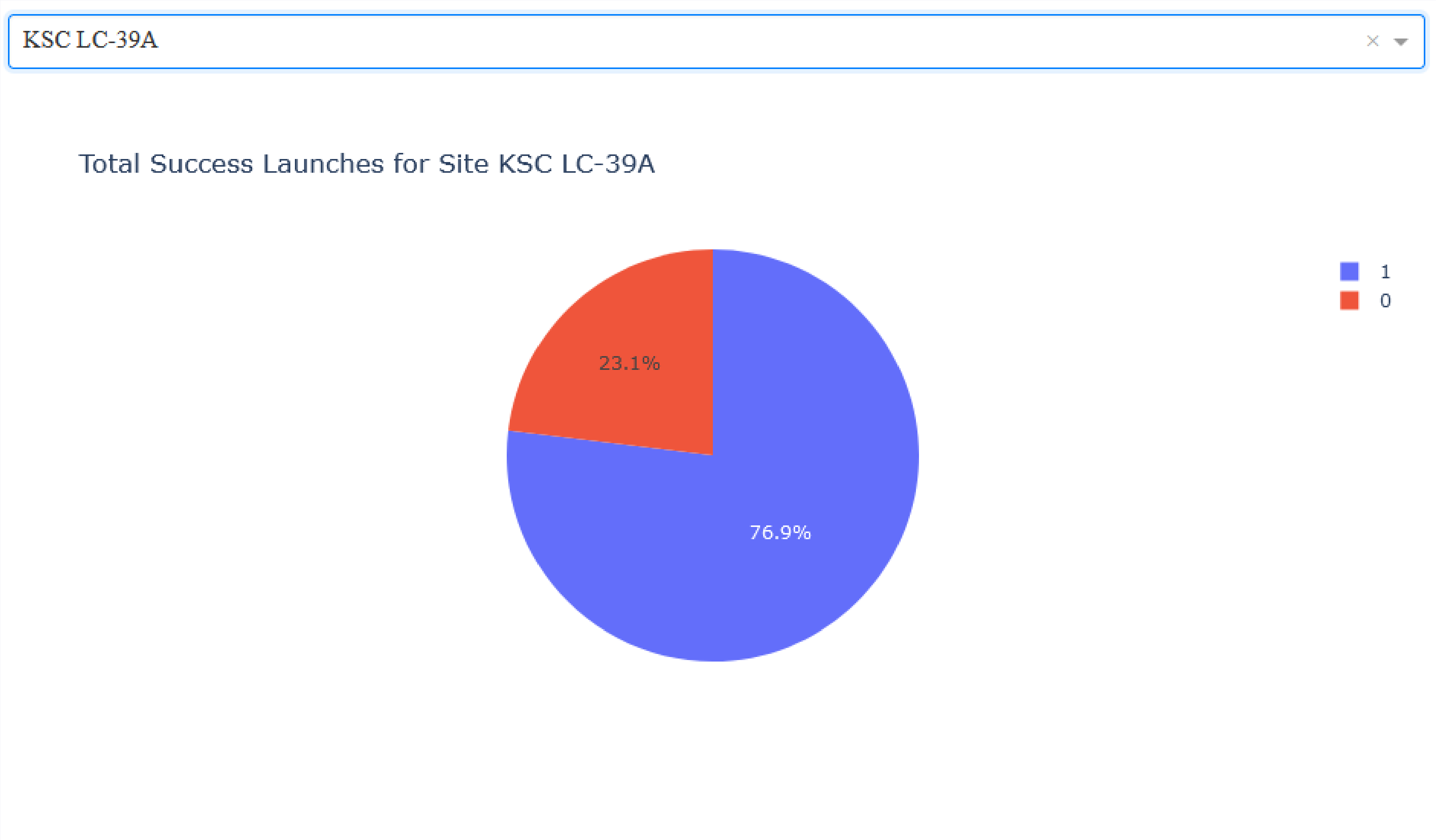
Build a Dashboard with Plotly Dash

Total Success Rates for SpaceX Launch Sites

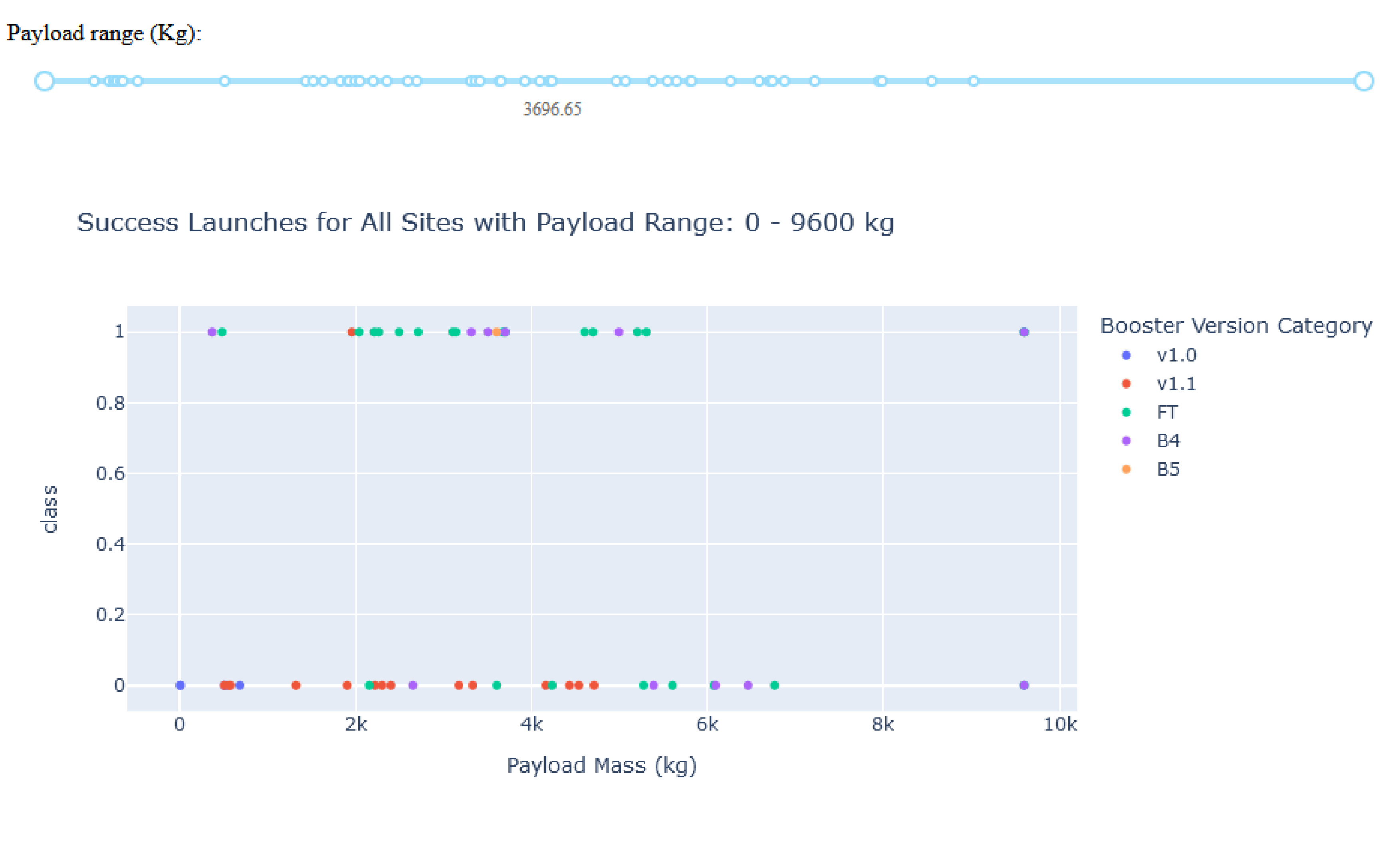


Based on the pie chart data, we can see that KSC LC-39A had the highest percentage of successful launches at 41.7%, followed by CCAFS LC-40 with 29.2%, VAFB SLC-4E with 16.7%, and CCAFS SLC-40 with the lowest percentage of successful launches at 12.5%.

Launch Site with Highest Success Rate



Payload and Launch Success Rates

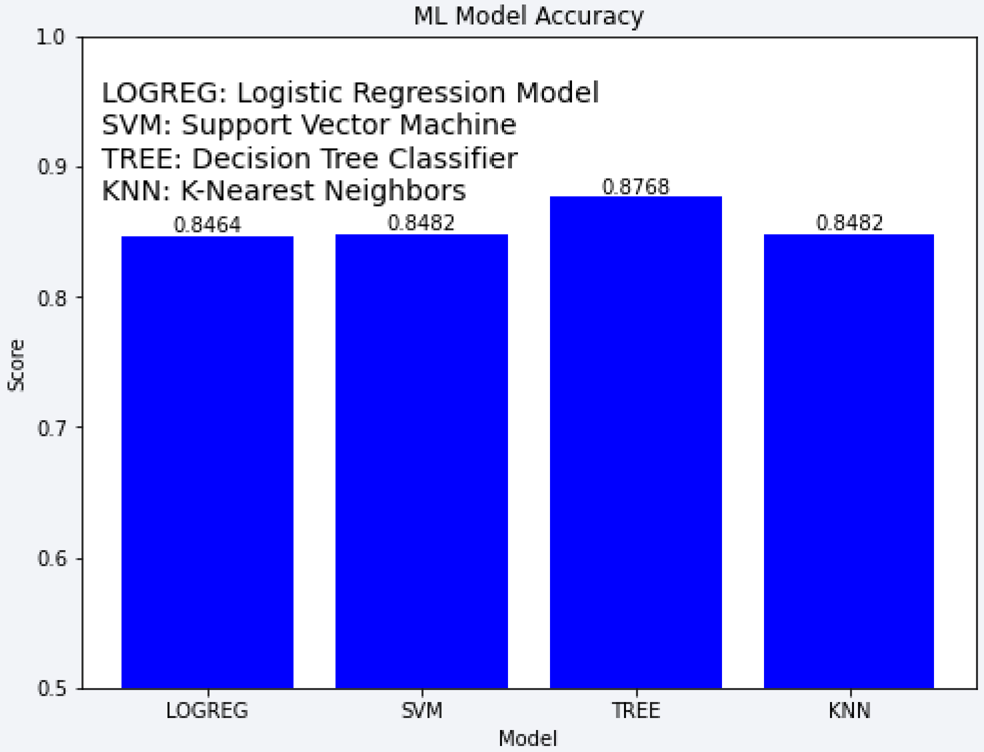


Section 5

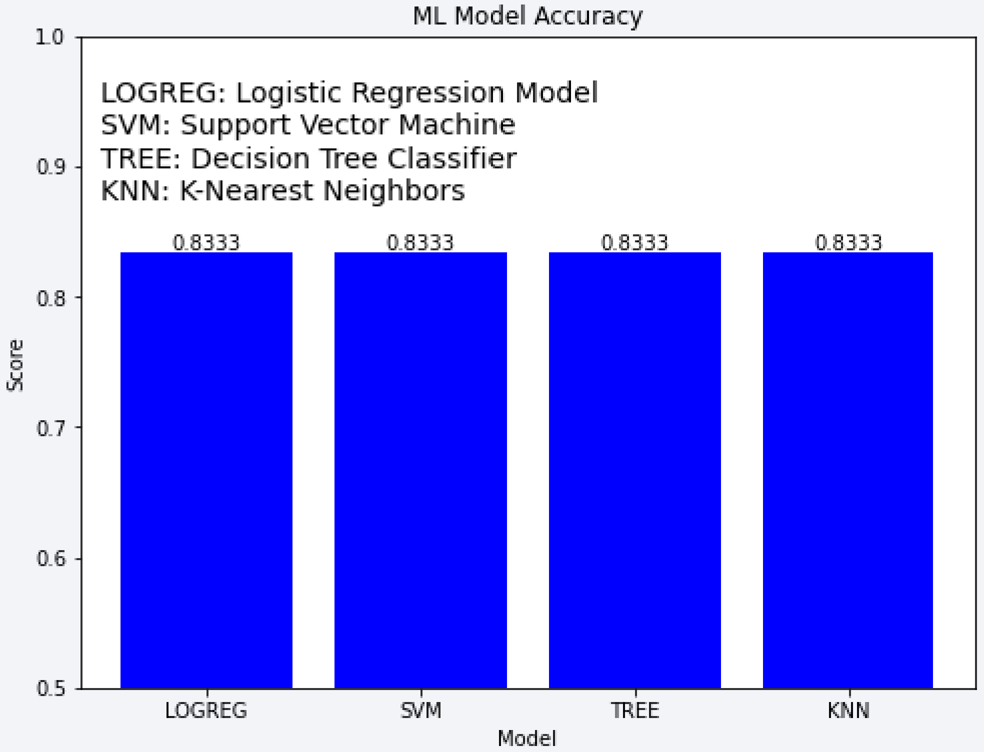
Predictive Analysis (Classification)

Classification Accuracy

Model Accuracy on the Training Data



Model Accuracy on the Test Data



Based on the calculated accuracy scores, all of the models seem to perform similarly on the test data with an accuracy of 0.83, although the Decision Tree Classifier has good performance on the training data, but overfitting is apparent on the test data. Overall, the given results may indicate that LOGREG and SVM could be a good choice to apply the model to new data. See also Additional Results.

Confusion Matrix

LOGREG/SVM

The results of the confusion matrix (LOGREG/SVM) are as follows:

- True negatives (TN): the classifier correctly predicted that a landing would not occur 3 times.
- False positives (FP): the classifier predicted a landing would occur 3 times when it did not.
- False negatives (FN): the classifier correctly predicted a landing would occur 12 times.
- True positives (TP): the classifier correctly predicted that a landing would occur 12 times.

Accuracy = (number of correctly classified instances) / (total number of instances) = 15/18 ≡ 83.33%

Conclusions

- The provided insights from the exploratory data analysis (EDA) suggest that the launch site CCAFS SLC 40 has a higher variability in launch outcomes, while launch sites with a lower proportion of unsuccessful launches may have better control over factors that could lead to mission failure. The Kennedy Space Center Launch Complex 39A (launch site KSC LC-39A) had the highest percentage of successful launches.
- The success rate for different orbits varies, with GEO, SO, and VLEO having the highest success rate and ES-L1, HEO, LEO, and MEO having the lowest success rate. The success rate in LEO orbit appears to be related to the number of flights, whereas no relationship between flight number and success is evident in GTO orbit.
- The successful landing or positive landing rate is higher for Polar, LEO, and ISS when heavy payloads are involved, while in GTO, it is difficult to distinguish between successful and unsuccessful missions.
- The launch success rate has been steadily increasing since 2013 until 2020, indicating an improvement in the overall success rate of launches during this period
- These queries extract specific data from a SpaceX database table called "SPACEX" based on various conditions, such as selecting unique launch sites, filtering by launch dates and landing outcomes, and calculating the total payload mass and average payload mass for specific missions.
- This project uses a global map of launch sites, created with folium, to explore the proximity of launch sites to various points of interest. It uses markers, circles, and polylines to represent launch sites and calculates distances between launch sites and other points on the map. The project focuses on four launch sites and uses mouse position and distance calculation methods to explore distances. Successful and failed launches are represented with green and red markers, respectively.
- This is a dashboard built with Plotly Dash for analyzing SpaceX launch data. It includes a pie chart of success rates for launch sites, a scatter plot showing payload of different booster versions and launch success rates, and a pie chart identifying the launch site with the highest success rate.
- This report analyzes the accuracy of different models for landing occurrence classification using bar plots and a confusion matrix. The Logistic Regression (LOGREG) and Support Vector Machine (SVM) models performed similarly well, suggesting they may be good choices for future use. The overall accuracy was 83.33%, with 12 true positives, 3 true negatives, 3 false positives, and 12 false negatives.

Reference

[1] SpaceX API: https://api.spacexdata.com/v4/launches/past (https://github.com/r-spacex/SpaceX-API), Accessed: 14.02.2023

[2] List of Falcon 9 and Falcon Heavy launches (Wikipedia)
URL: https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches, Accessed: 14.02.2023

Thank you!