IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Vivasvat Kaul
6/9/2022

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection
  - Webscraping
  - Data Wrangling
  - EDA using SQL
  - EDA using Data Visualization
  - Launch Site Location Analysis using Folium
  - Creating a Dashboard with Plotly Dash
  - Comparison of Machine Learning Methods for Predicting Success of First Stage
- Summary of all results
  - Exploratory Data Analysis results
  - Screenshots of Dashboard from Plotly Dash
  - Results of Machine Learning Models
    - Determination of which was best

The objective of this project was to analyze SpaceX data and then use that data to train and test a machine learning model to predict the success of the first stage of the rocket's launch.

In the end the best model for predicting the outcome of the first stage was found to be a Decision Tree model, though the other models tested all exhibited similar accuracies.

Introduction

- Project background and context
  - The aim of this project was to create a model that would predict if the first stage of the Falcon 9 SpaceX rocket would successfully land. There is a tremendous financial incentive to do this considering that much of the savings for the rocket comes form being able to reuse the first stage. Doing this would allow a competitor to refine the methods pioneered by SpaceX to better compete.

- Problems to Solve
  - What influences the success of the first stage of the Falcon 9 rocket?
  - What other interesting interactions might be worth considering for future data explorations and model building?

Section 1

Methodology

Methodology

Methodology Executive Summary

- Data collection methodology:
  - Data was collected using the SpaceX REST API
  - Additional information was also obtained via web scraping the Wikipedia article "List of Falcon 9 and Falcon Heavy launches"
- Perform data wrangling
  - Landing data containing the launch outcomes was one-hot encoded to create a field called 'Class', where 1 signified that the booster had landed successfully and 0 meant that the landing was unsuccessful.
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Used scatter and bar plots to show the relationships between variables to look for patterns in the data.
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models (NOTE: The following was all done using sklearn)
  - Split the data into test and train data sets.
  - Normalize the data
  - Create four models: Logistic Regression, SVM, Decision Tree, and KNN.
  - The models were then ranked against each other using the score function from the sklearn package.

Data Collection

- The bulk of the data was obtained via the SpaceX REST API. Information from the API included the following information:
  - Rocket used
  - Payload mass
  - Launch location (through latitude and longitude)
  - Landing outcome (the variable that was to be predicted)
- Additional information was also obtained via web scraping the Wikipedia article "List of Falcon 9 and Falcon Heavy launches". Information within the web scrape included:
  - Flight Number
  - Version of Booster
  - Customer.

Data Collection – SpaceX API

- Flowchart of SpaceX REST API call, conversion into DataFrame, and output to .csv.

- GitHub Notebook

1. Getting Response from API

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

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

2. Converting Response to a .json file

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

3. Apply custom functions to clean data

```
# Call getLaunchSite
getLaunchSite(data)
```

```
# Call getPayloadData
getPayloadData(data)
```

```
# Call getCoreData
getCoreData(data)
```

4. Assign list to dictionary then DataFrame

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

5. Filter DataFrame and export to .csv

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = launch_data.loc[launch_data.BoosterVersion != 'Falcon 1'].reset_index(drop=True)
```

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

Data Collection - Scraping

- Flowchart of Wikipedia web scrape, conversion to DataFrame, and output to .csv.
- [GitHub Notebook]

1. Getting Response from HTML

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

2. Creating BeautifulSoup object

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

3. Finding tables

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

4. Getting column names

```
column_names = []

# Apply find_all() function with `th` element on
th_list = first_launch_table.find_all('th')

# Iterate each th element and apply the provided
for item in th_list:
    name = extract_column_from_header(item)
    # Append the Non-empty column name (`if name
    if name is not None and len(name)>0:
        column_names.append(name)
```

5. Creation of dictionary

```
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']=[]
```

6. Appending data to keys

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        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 key `Flight No.`

            #print(flight_number)
            datatimelist=date_time(row[0])
            launch_dict['Flight No.'].append(flight_number)
```

7. Converting dictionary to DataFrame

```
# df=pd.DataFrame(launch_dict)
# replaced the above with the following from the forums
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
```

8. DataFrame to .csv

```
df.to_csv('spacex_web_scraped.csv', index=False)
```

Data Wrangling

In the data set, there are several different cases where the booster did not land successfully. Sometimes a landing was attempted but failed due to an accident; for example, **True Ocean** means the mission outcome was successfully landed to a specific region of the ocean while **False Ocean** means the mission outcome was unsuccessfully landed to a specific region of the ocean. **True RTLS** means the mission outcome was successfully landed to a ground pad **False RTLS** means the mission outcome was unsuccessfully landed to a ground pad. **True ASDS** means the mission outcome was successfully landed on a drone ship **False ASDS** means the mission outcome was unsuccessfully landed on a drone ship.

We convert those outcomes into Training Labels with 1 means the booster successfully landed 0 means it was unsuccessful.

- Flowchart of counts for various fields, one-hot encoding of Outcome column, and output to .csv
- GitHub Notebook

1. Calculate the number of launches on each site

```
# Apply value_counts() on column LaunchSite
df.LaunchSite.value_counts()
```

2. Calculate the number and occurrence of each orbit

```
# Apply value_counts on Orbit column
df.Orbit.value_counts()
```

3. Calculate the number and occurrence of mission outcome per orbit type

```
landing_outcomes = df.Outcome.value_counts()
```

4. Create a landing outcome label from Outcome column

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

5. Output to .csv

```
df.to_csv("dataset_part_2.csv", index=False)
```

EDA with Data Visualization

- Scatter Plots Drawn:
  - 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
- Bar Plots Drawn:
  - Mean Success Rate for each Orbit Type
- Scatter Plots Drawn:
  - Landing Success Rate from 2010-2019
- GitHub Notebook

EDA with SQL

- Here are the lists of queries that were run:
  - Displaying the names of the unique launch sites in the space mission
  - Displaying five records where launch sites begin with the string 'CCA'
  - Displaying the total payload mass carried by boosters launched by NASA (CRS)
  - Displaying average payload mass carried by booster version F9 v1.1
  - Listing the date when the first successful landing outcome in on a ground pad was achieved
  - Listing the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
  - Listing the total number of successful and failure mission outcomes
  - Listing the names of the booster versions which have carried the maximum payload mass using a subsquery
  - Listing the failed landing outcomes on the drone ship, their booster versions, and launch site names in the year 2015.
  - Ranking the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the dates 2010-06-04 and 2017-03-20, in descending order.

GitHub Notebook

Build an Interactive Map with Folium

- The objects that were added to the map using Folium included the following:
  - Launch sites
  - Successful and unsuccessful landings
  - Proximity example to key locations: railway, highway, coast, and city.

- This allows us to understand why launch sites may be located where they are. Also visualizes successful landings relative to location.

- GitHub Notebook

Build a Dashboard with Plotly Dash

- Dashboard includes a pie plot and scatter plot
  - The pie plot shows the percentage of successful launches across all launch sites. It can also be altered to show the percentages of successful and unsuccessful launches at any given launch site.
    - This provides an easy-to-read visual that displays the success rate for any given launch site.
  - The scatter plot shows successful and unsuccessful launches as a function of payload mass. There is a slider that goes from 0 kg – 10000 kg. The legend also details what booster version is represented by each point.
    - This allows the user to see how the success of the launch is dependent on launch sites, payload mass, and booster version category.
- GitHub Plotly Dashboard

Predictive Analysis (Classification)

GitHub Notebook

Build the Model

- Load and convert the data into numpy arrays to perform mathematical operations on.
- Normalize the data
- Perform a test-train split
- Create the classification objects of KNN, SVM, Decision Tree, and Logistic Regression.

Evaluate the Model

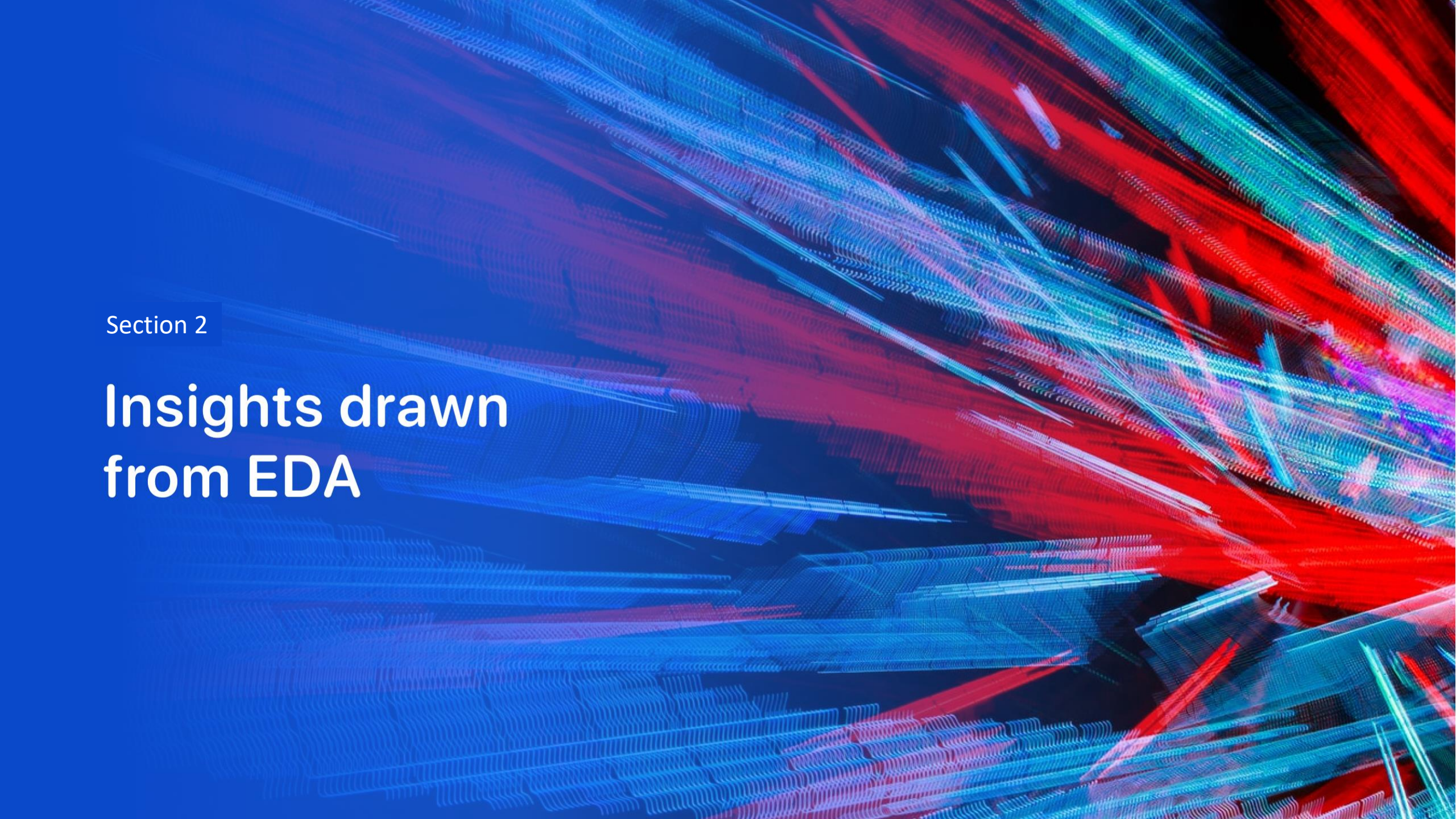
- Create the GridSearchCV object for all of the classification models
- Fit all the model objects by passing training data.
- Output the GridSearchCV object and analyze the values.

Find the Best Performing Model

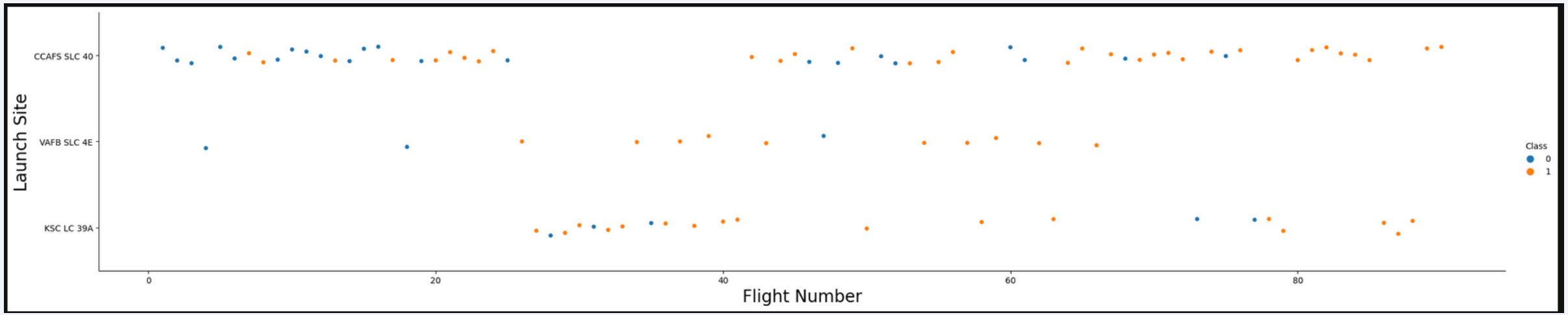
- Find the tuned hyperparameters using best_params_
- Find the accuracy on the validation data using best_score_
- Calculate the accuracy on the test data using score method
- Plot the confusion matrices for comparison.

Results

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

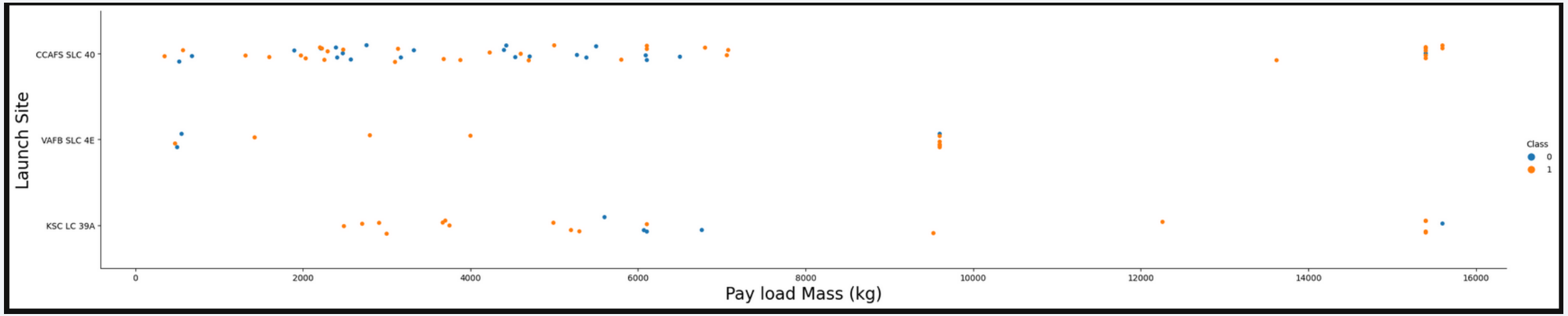

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



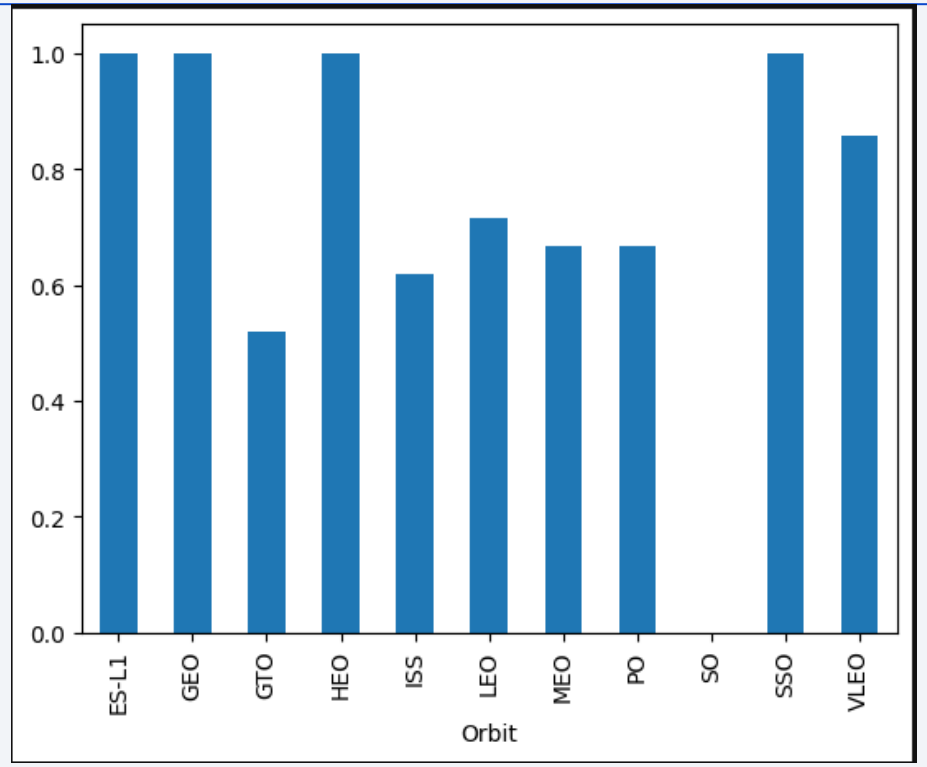
- Scatter plot of Flight Number vs. Launch Site. Blue dots indicate an unsuccessful launch while orange dots indicate a successful launch. CCAFS appears to be the main launch site just based on the volume of flights. We can also see that after flight number 20, the proportion of successful launches seems to increase.

Payload vs. Launch Site



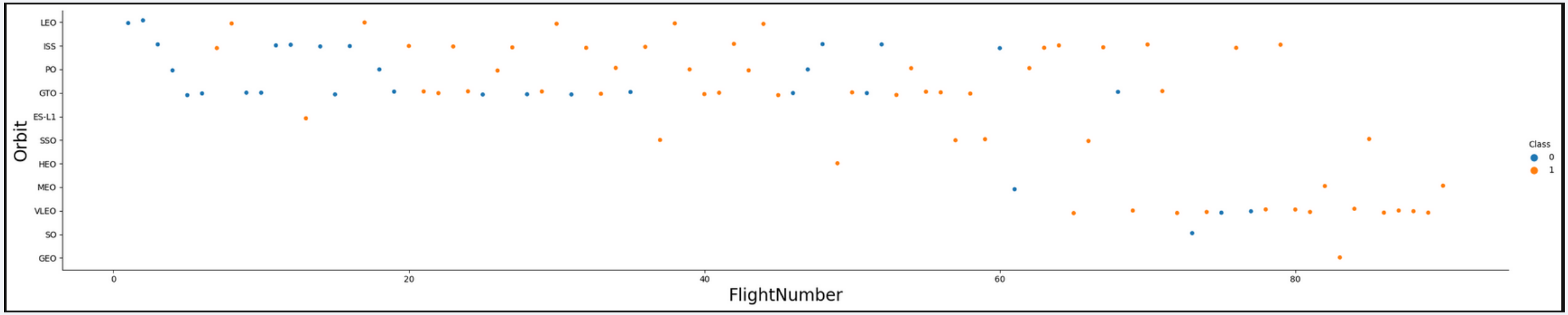
- Scatter plot of Payload Mass vs Launch Site. Most of the payloads appear to be distributed within a range of 0 to 6000 kg. VAFB does not seem to support payloads exceeding 10000 kg. Interestingly, we see that there appear to be fewer unsuccessful launches at heavy payloads proportionally.

Success Rate vs. Orbit Type



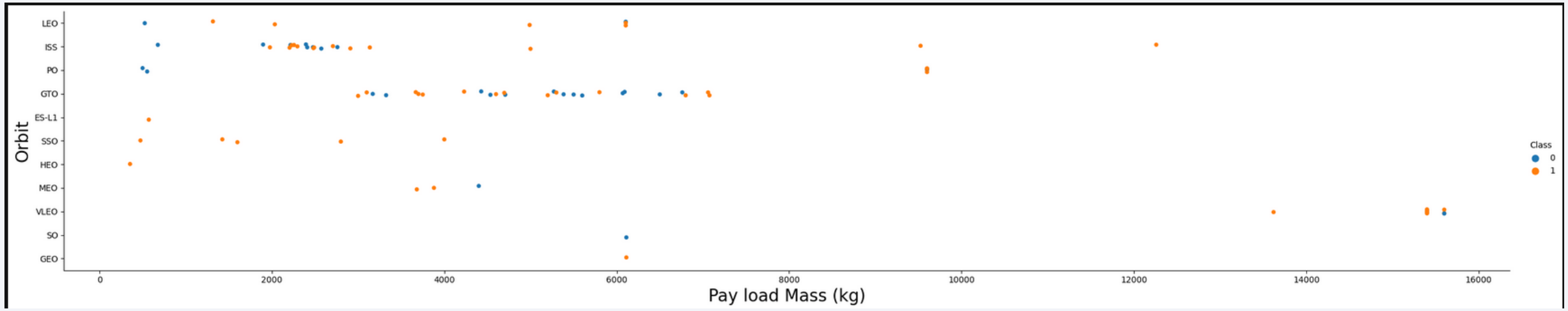
- Bar plot of Mean Success Rate vs Orbit Type. We can see that ES-1, GEO, HEO, and SSO have the highest success rates from the array of orbit types. Given how many types there are, it would be reasonable to conclude that orbit type could have an impact on launch success.

Flight Number vs. Orbit Type



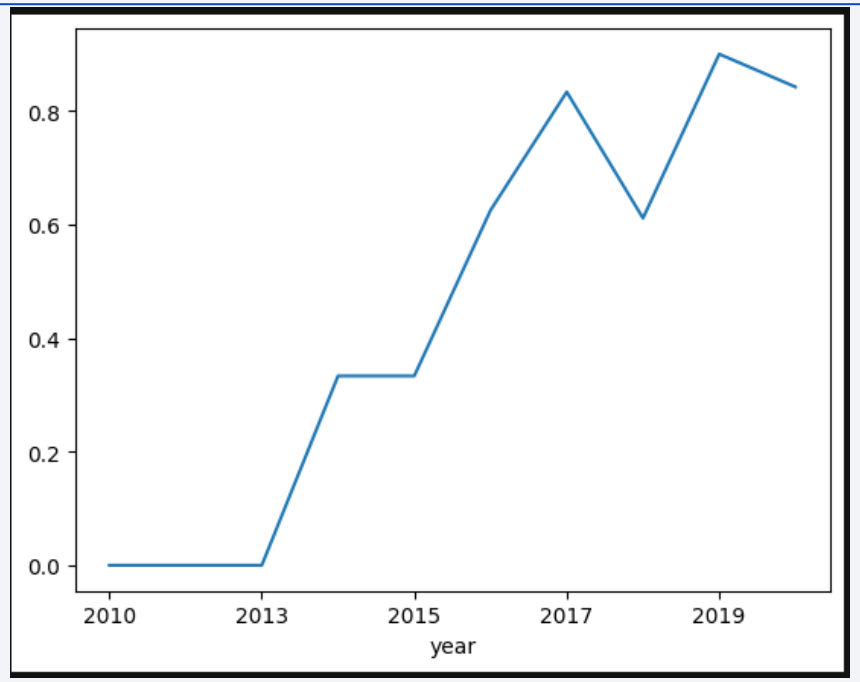
- Scatter plot depicting Flight Number vs. Orbit Type. The most common orbit types by the number of flights appear to be LEO, ISS, and GTO. The addition of VLEO appears to be more recent given the plot. The launches to ISS and VLEO appear to be the most successful of the orbit types based on flight number.

Payload vs. Orbit Type



- Scatter plot of Payload Mass vs Orbit Type. It's difficult to discern a correlation between payload mass and orbit type. The payload mass is not uniformly distributed across all orbit types. Therefore, it's difficult to say if there's a significant correlation based on the relationship of payload mass and orbit type.

Launch Success Yearly Trend



- Line graph showing the Average Launch Success from 2010-2020. We can see that launch success has trended up over time until there was a dip in 2018. Logically, this makes sense since we would expect that has SpaceX refined its methodology, the success rate for launches has generally gone up.

All Launch Site Names

- Query to obtain distinct Launch Site names.
- Using a SELECT DISTINCT on the launch_site column, we can see that there are four unique sites.

```
%%sql
select distinct launch_site
from spacextbl
```

```
 * ibm_db_sa://jzs26123:***@1
Done.
```

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

Launch Site Names Begin with 'CCA'

```
%%sql
select *
from spacextbl
where launch_site like 'CCA%'
limit 5
```

```
 * ibm_db_sa://jzs26123:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30699/bludb
Done.
```

DATE	time_utc_	booster_version	launch_site	payload	payload_mass_kg_	orbit	customer	mission_outcome	landing_outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	07:44:00	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	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

- Query to obtain the first five records with Launch Site where the names began with ‘CCA’
- This query is useful since it can help to pull out any data that is related to either of the CCAFS Launch Sites. This is useful for filtering the data.

Total Payload Mass

- Query to calculate the total Payload Mass carried by boosters by NASA (CRS)
- Using a SUM command, we can easily get the total payload that has been sent by a given customer. This can be useful to help determine which customers may require launches that can hold larger payloads.

```
%%sql
select SUM(payload_mass__kg_)
from spacextbl
where customer = 'NASA (CRS)'
```

```
 * ibm_db_sa://jzs26123:***@19a
Done.
    1
45596
```

Average Payload Mass by F9 v1.1

- Query to calculate the average payload mass carried by booster version F9 v1.1
- Using the AVG function and a WHERE clause, the average payload can be calculated for a given Booster Version. This is useful in order to help determine if there is a link between the load that a booster carries and its chance of success.

```
%%sql
select AVG(payload_mass__kg_)
from spacextbl
where booster_version = 'F9 v1.1'
```

```
 * ibm_db_sa://jzs26123:***@19af644
Done.
```

1
2928

First Successful Ground Landing Date

- Query to find the date of the first successful landing outcome on ground pad
- Using the MIN function with a WHERE clause for the Landing Outcome, the historical records can be parsed to find important dates. This might be necessary to help track if the timing of specific events correlates to possible launch success.

```
: %%sql
  select MIN(date)
  from spacextbl
  where landing__outcome = 'Success (ground pad)'

   * ibm_db_sa://jzs26123:***@19af6446-6171-4641-8a
  Done.

:          1

  2015-12-22
```

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%%sql
select distinct booster_version
from spacextbl
where landing__outcome = 'Success (drone ship)' and payload_mass__kg_>4000 and payload_mass__kg_<6000
```

```
 * ibm_db_sa://jzs26123:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain
Done.
```

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

- Query to list the names of boosters which have successfully landed on the drone ship and had a payload mass greater than 4000 but less than 6000
- Using a SELECT DISTINCT clause and a compound WHERE clause, the data can be filtered on two fields simultaneously. We can look for a specific kind of success while also seeing which Booster Versions and Payload Masses worked.

Total Number of Successful and Failure Mission Outcomes

- Query to calculate the total number of successful and failed mission outcomes
- Using a COUNT function, we can create a new column called total that displays the total number of successes and failed outcomes. Then with a GROUP BY clause we can aggregate the counts.

```
%%sql
select mission_outcome, COUNT(mission_outcome) as total
from spacextbl
group by mission_outcome
```

```
 * ibm_db_sa://jzs26123:***@19af6446-6171-4641-8aba-9dcff8
Done.
```

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

Boosters Carried Maximum Payload

- Query to list the names of the booster which have carried the maximum payload mass
- Using a subquery with a MAX function, we can find the maximum Payload Mass and then use that to filter the data. From here we select the Booster Version. This is useful for seeing which boosters have been tested at the maximum weight used in launches.

```
%%sql
select distinct booster_version
from spacextbl
where payload_mass__kg_ = (select MAX(payload_mass__kg_) from spacextbl)
```

```
 * ibm_db_sa://jzs26123:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd
Done.
```

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

2015 Launch Records

```
%%sql
select landing__outcome, booster_version, launch_site
from spacextbl
where landing__outcome = 'Failure (drone ship)' and date between '2014-12-31' and '2016-01-01'
```

```
 * ibm_db_sa://jzs26123:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.a
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

- Query to list the failed Landing Outcomes in drone ship, their Booster Versions, and Launch Site names for the year 2015.
- Using a compound WHERE clause, we can select the three columns of interest. Here we see that two of the boosters failed within the time period. Interestingly, they were also launched from the same Launch Site.

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

- Query to rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the dates 2010-06-04 and 2017-03-20, in descending order.
- Using an ORDER BY clause and the DESC parameter, we can order the results. Here we can see that of all the possible Landing Outcomes, 'No attempt' is the highest at 10. This would seem to indicate that there may have been ground conditions that prevented the launch from occurring. We can also see that drone ship landings were attempted more than other types within the time period specified.

```
%%sql
select landing__outcome, count(landing__outcome) as count
from spacextbl
where date between '2010-06-04' and '2017-03-20'
group by landing__outcome
order by count desc
```

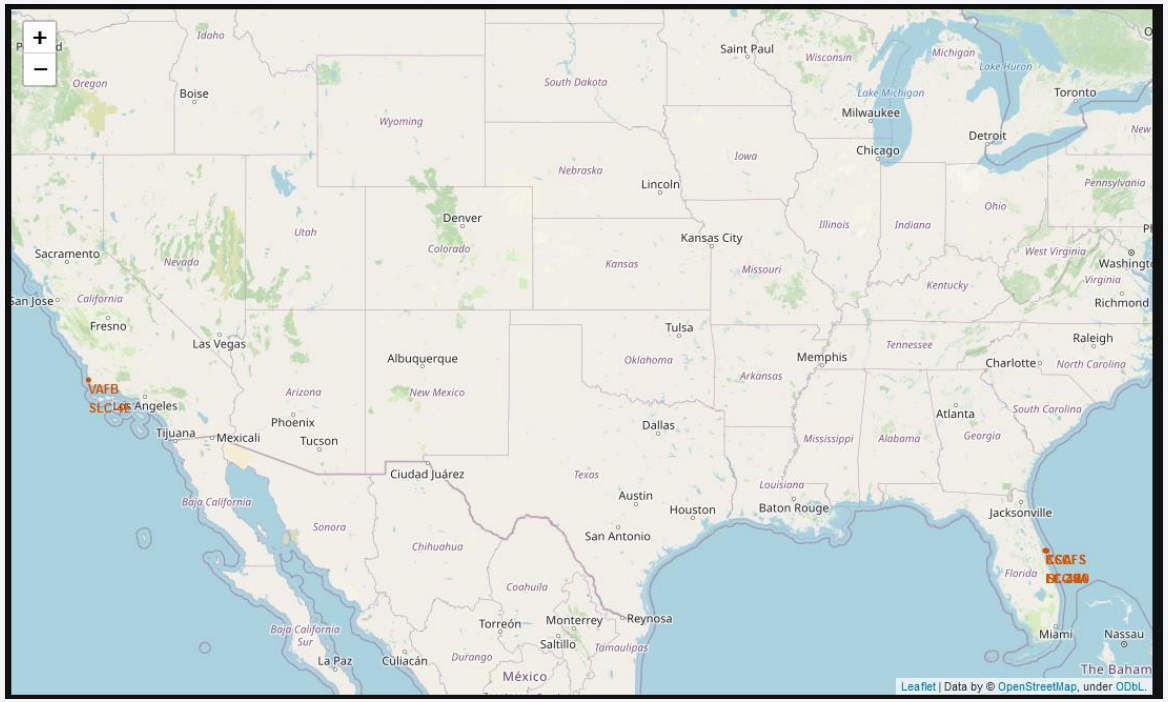
```
 * ibm_db_sa://jzs26123:***@19af6446-6171-4641-8aba-9dcff8e
Done.
```

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

Section 3

Launch Sites Proximities Analysis

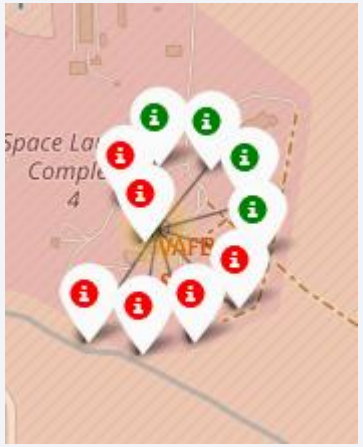
All Launch Sites – Global Map



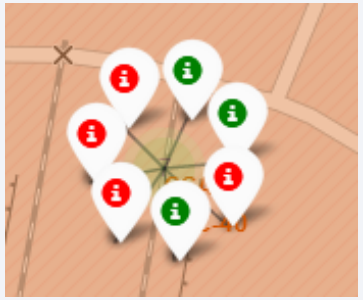
- All Launch Sites – Global Folium Map
- We can see that the launch sites are concentrated on the coasts. VAFB is located in southern California and KSC and both CCAFS sites located in Florida.

Launch Results for each Launch Site

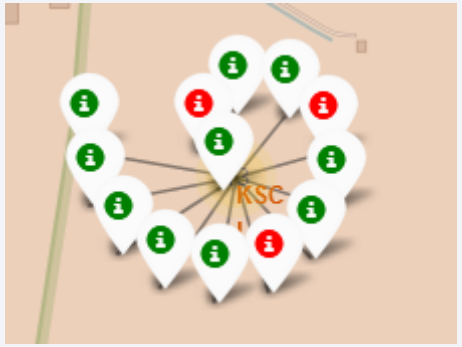
- Looking at the four launch sites we can map how many of the launches from that site were actually successful on the Folium Map. Successes are marked in green, while failures are marked in red. We can see that, proportionally, KSC LC-39 A clearly has the highest proportion of successful launches. CCAFS LC-40 has the highest number of launches, but more than half of them failed.



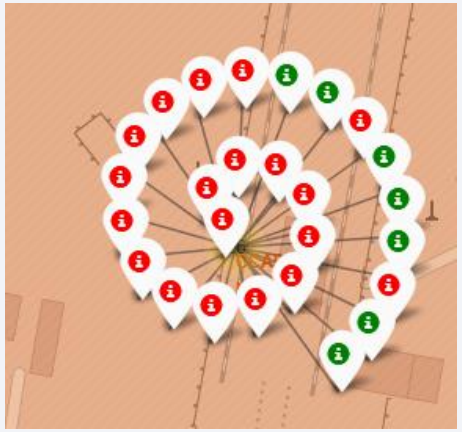
VAFB SLC-4E



CCAFS SLC-40

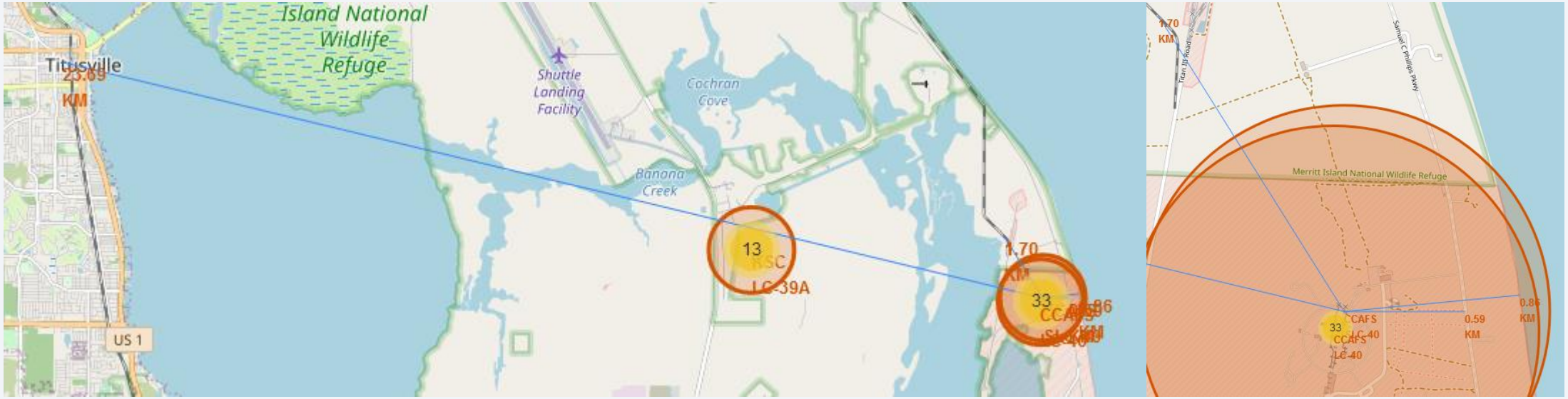


KSC LC-39 A

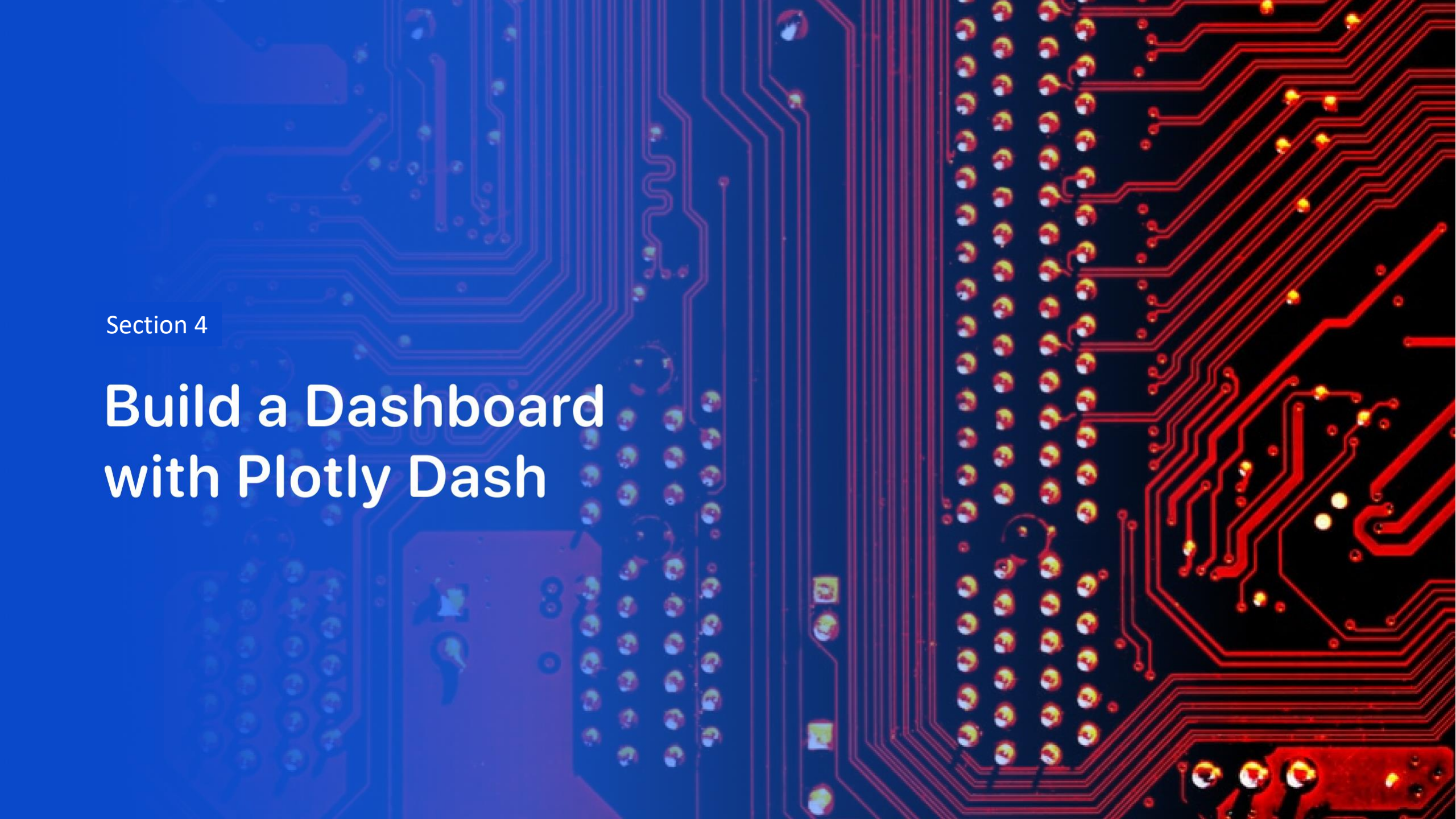


CCAFS LC-40

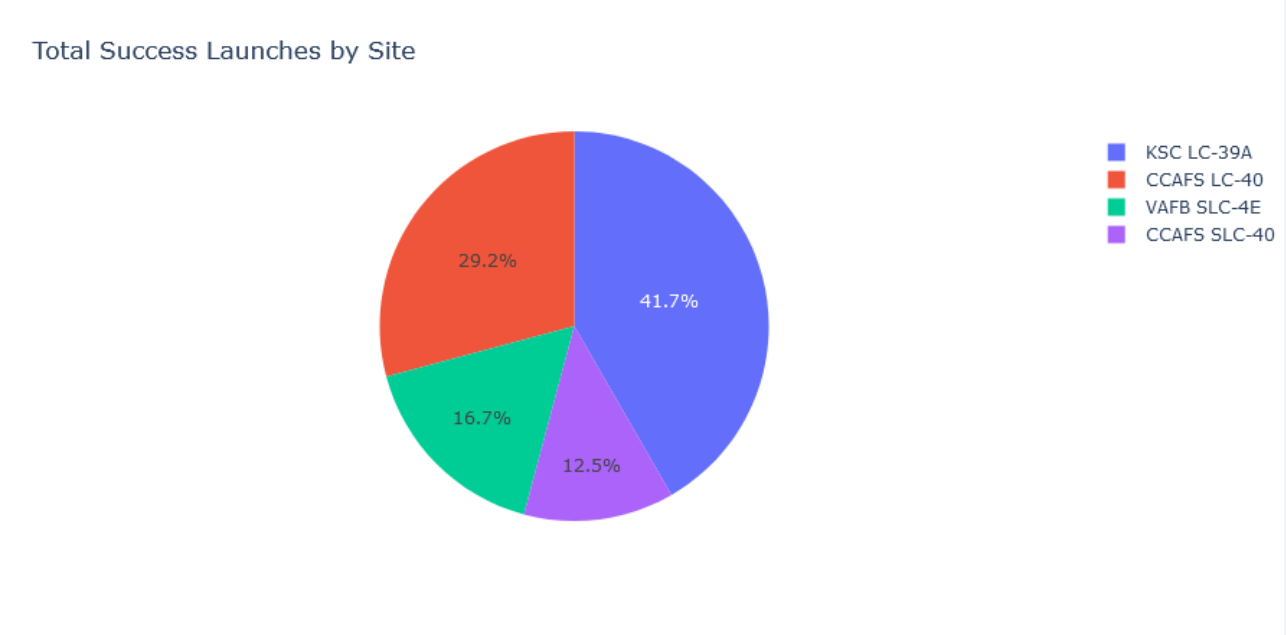
Landmark Proximity to CCAFS SLC-40



- The screenshot on the left shows the Folium Map depicting the closest city to CCAFS SLC-40 which is Titusville which is approximately 23.69 km away. The right screenshot shows a more zoomed in view to show the closest railroad (1.70 km), closest highway (0.59 km), and closest coast (0.86 km). It follows logically that given that something could go wrong during a launch that the site would be located far from a populated area. But getting material to the site via rail or road would still be necessary. Finally, since the rocket will attempt to land on a drone pad in the ocean, the site is in close proximity to the coast.

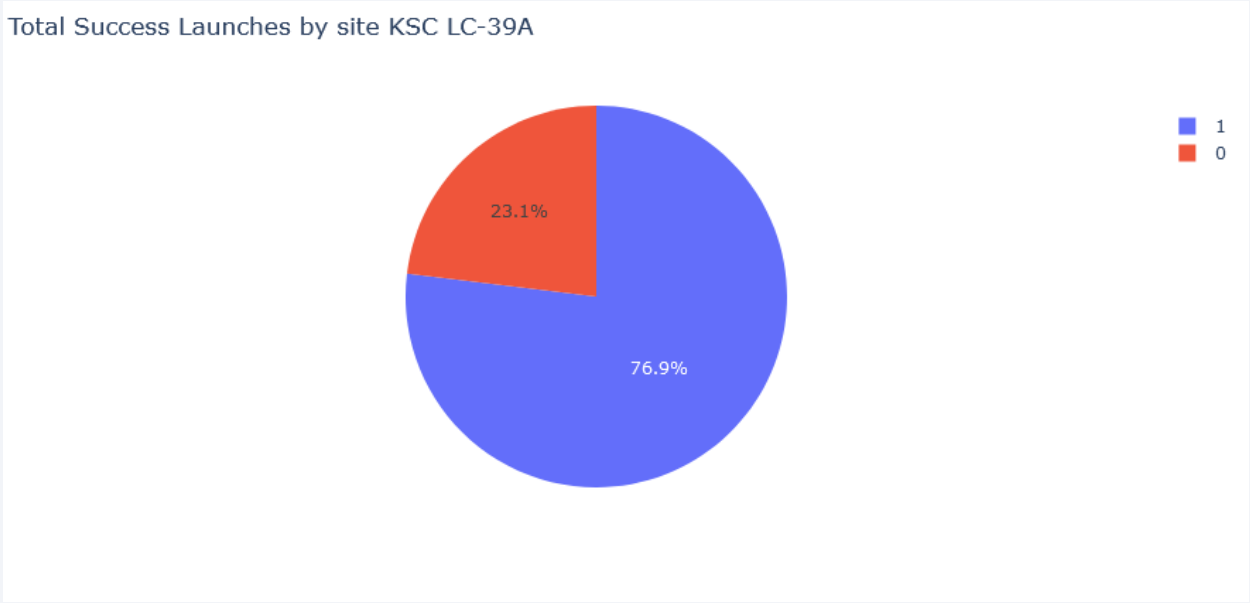

Section 4
Build a Dashboard with Plotly Dash

Total Successful Lauches by Site



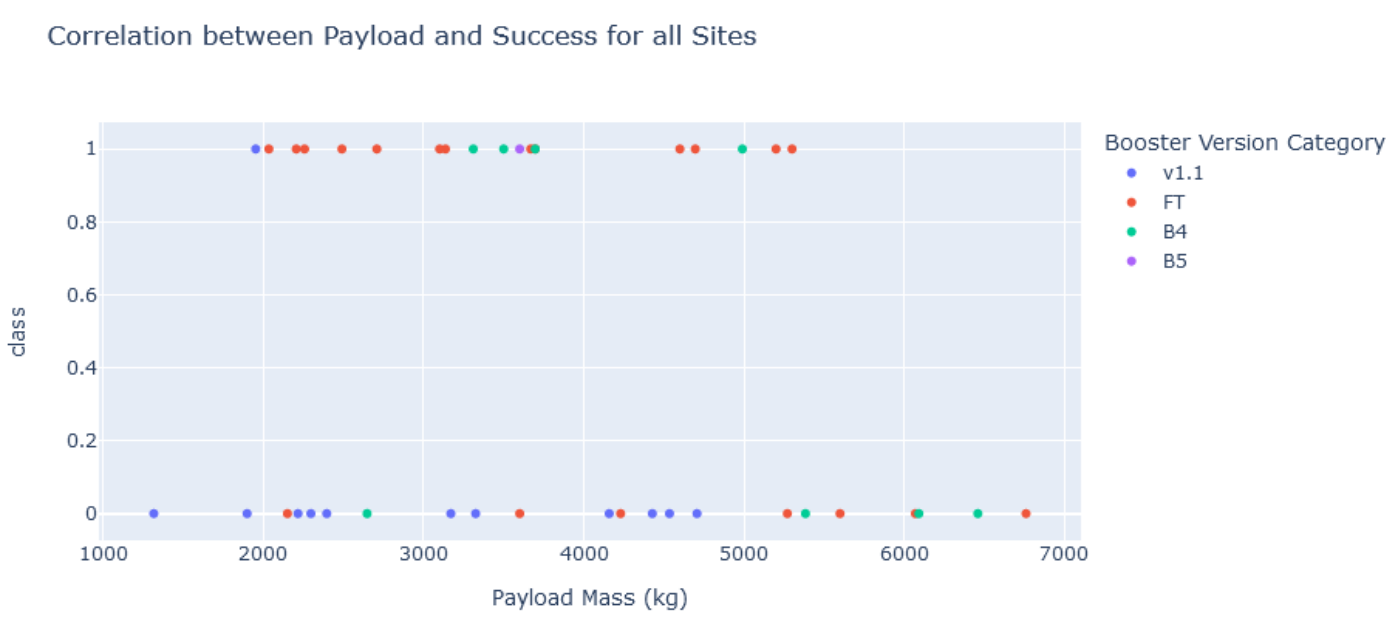
- The pie chart above shows the percentage of successful launches for each Launch Site as a function of the total number of launches. We can see that KSC LC-39A represents the highest proportion of successes with 41.7% of the total.

Total Successful Lauches at KSC LC-39A



- The above screenshot shows the percentage split of successful and unsuccessful launches from KSC LC-39A. We can see that the overwhelming majority of launches (76.9%) were successful from this site, with only 23.1% failing.

Payload Mass vs. Launch Outcome – All Sites

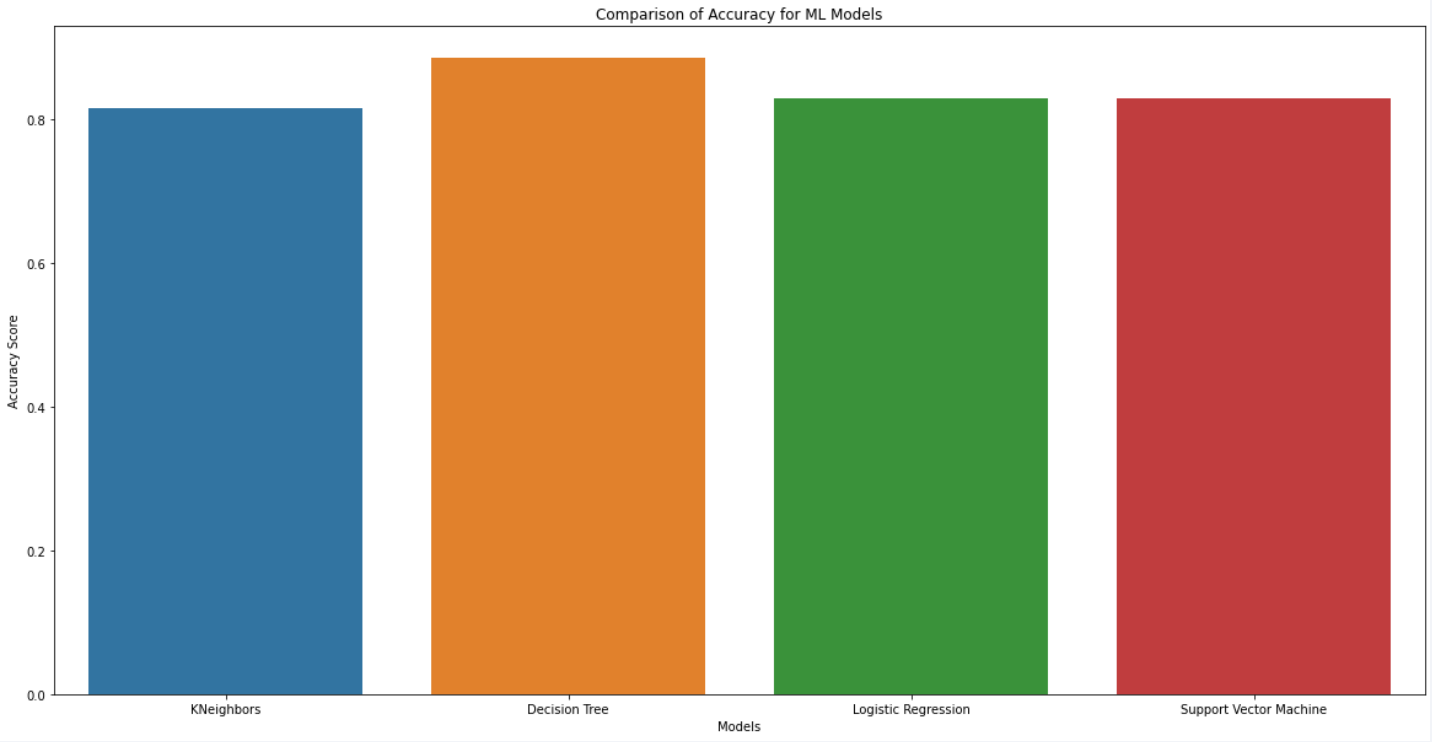


- The above scatter plot depicts the Payload Mass vs. Launch Outcome for all launch sites. From the scatter we can see that most of the points are concentrated within the range of 1000-7000 kg. Within this range, we can see that the v1.1 booster had the highest number of failures. Conversely, within this range the FT booster had the highest number of successful launches.

Section 5

Predictive Analysis (Classification)

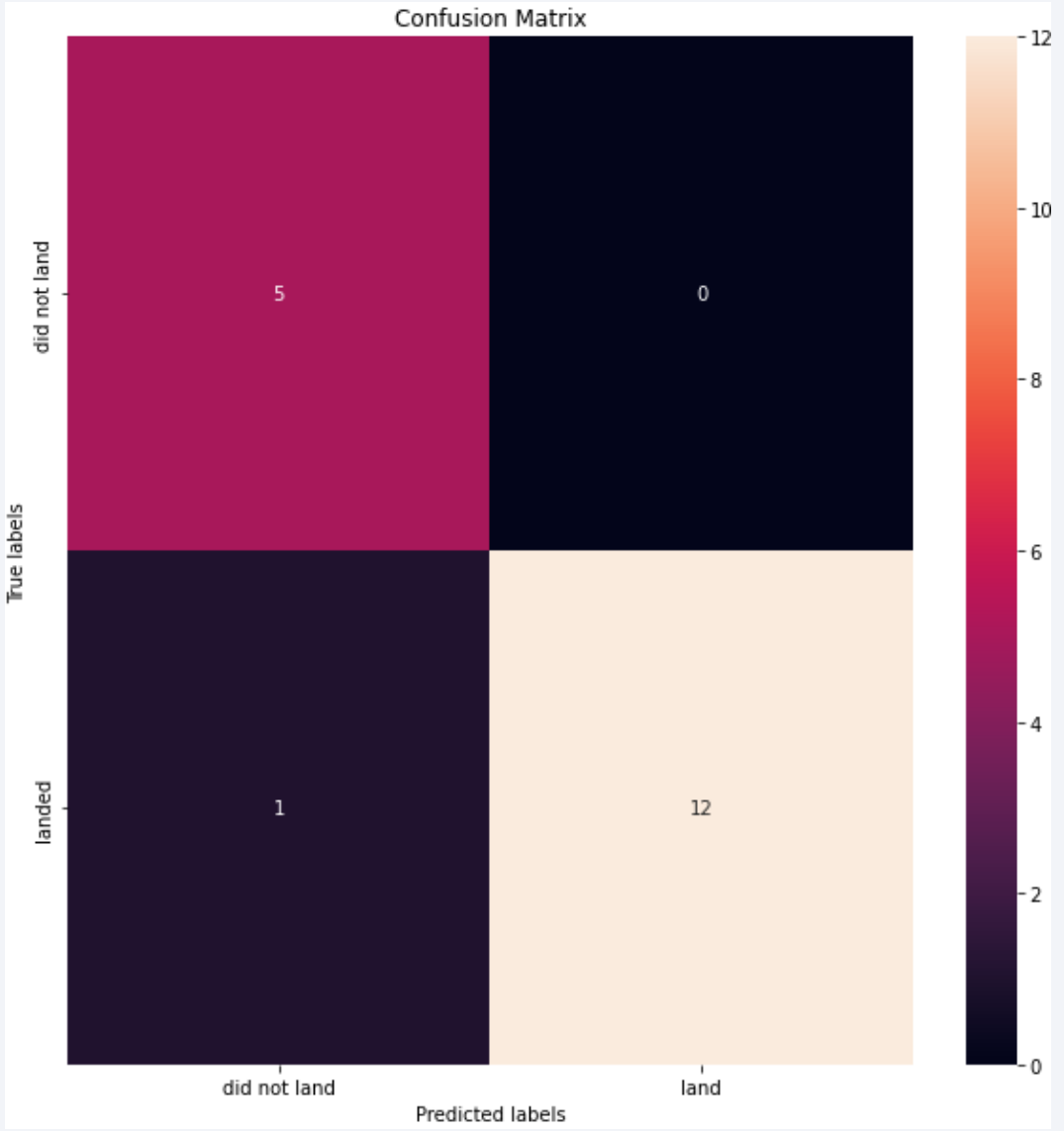
Classification Accuracy



- The bar plot above shows the classification accuracy for each of the models tested. We can see that the Decision Tree classifier has the highest accuracy with a score of 0.9000. The KNN model had the lowest accuracy with 0.8161 and the Logistic Regression and SVM both had 0.8286.

Confusion Matrix

- To the right is the confusion matrix for the Decision Tree classifier. From the top left and moving clockwise, the boxes represent the True Negative (TN), False Positive (FP), True Positive (TP), and False Negative (FN). We can see that the model correctly classified all of the TN samples and misclassified only one TP result from the test set. This result coupled to the high accuracy score show that this is a very good model for the data.



Conclusions

- In conclusion, working through this project yielded several findings for a company that would be interested in following in the footsteps of SpaceX.
  - Generally, the structure of the data is very useful itself and should be replicated for any company planning on moving forward. Having it be available via REST API is very convenient for analysis. Moreover, opening this API up to the public for individual analysis would be very handy.
  - Additional data should be added to the API so that a web scrape from an alternate site is not necessary as all of the data can be obtained from a single source.
  - The data included within the REST API can be cleaned very easily using Pandas and modeled using Sklearn to create a robust set of predictive models from which the Decision Tree classifier exhibited the highest accuracy and a very high specificity and sensitivity. These can be further improved with the incorporation of new data.
  - Furthermore, additional data could also be collected to improve the predictive accuracy, such as weather conditions at the time of the launch, fuel type, and man hours spent building the rocket.

Appendix

- Modifications to plot_confusion_matrix
  - The function for plot_confusion_matrix was altered to enlarge the output figure size to make it easier for reporting. The figure to the right shows that the figsize has been included to be 10x10.
- Breaking out the TN, TP, FN and FP from the confusion matrix.
  - The ravel function was used to understand the layout of the confusion matrix from sklearn. The function shown at the right comes form the sklearn documentation and can be found here: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

```
def plot_confusion_matrix(y, y_predict):
    "this function plots the confusion matrix"
    from sklearn.metrics import confusion_matrix

    cm = confusion_matrix(y, y_predict)
    plt.figure(figsize=(10, 10))
    ax = plt.subplot()
    sns.heatmap(cm, annot=True, ax=ax)
    # annot=True to annotate cells
    ax.set_xlabel("Predicted labels")
    ax.set_ylabel("True labels")
    ax.set_title("Confusion Matrix")
    ax.xaxis.set_ticklabels(["did not land", "land"])
    ax.yaxis.set_ticklabels(["did not land", "landed"])
```

```
]: from sklearn.metrics import confusion_matrix

tn, fp, fn, tp = confusion_matrix(Y_test, yhat).ravel()
print("TN:", tn)
print("TP:", tp)
print("FN:", fn)
print("FP:", fp)
```

```
TN: 5
TP: 12
FN: 1
FP: 0
```

Thank you!