Winning Space Race with Data Science

Rhesa Browning
12/1/2021

Outline

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

Executive Summary

The goal of this project was to analyze Space X Falcon 9 launch data in order to understand what factors promote the safe landing of the stage 1 rocket. Space X is able to charge less than their competitors ($62M vs $165M per launch) because they are able to reuse the stage 1 rockets which are successfully landed.

Data was provided by the Space X Rest API and Wikipedia page. In order to prepare it for analyze, the data sets had to be collected, scraped, wrangled, standardized, and reformatted into Pandas dataframes. The data was visually analyzed by various plots and charts, SQL queries, Folium maps, and an interactive dashboard made with Plotly Dash software. The analysis determined that success rates improved as more launches were performed and with lighter payloads. The launch site tht had the highest rate of success was KSC LC 39A. Also, launches into orbits ES-L1, GEO, HEO, SSO, and VLEO were successful more than 80% of the time.

Last, four classification models were built to predict if an individual launch would be successful. They were each tuned with GridSearchCV() and evaluated with accuracy calculations and confusion plots. The most accurate model was the Decision Tree Classifier (DTC). However, I recommend using the Support Machine Vector Classifier (SVMC) because the DTC Confusion Plot produced three times more False Negatives than the SVMC Confusion Plot.

Introduction

- Space transportation can now be provided by private businesses. Each company competes to deliver their services at the most competitive prices. Currently, Space X states they can charge $62 million to launch one of their Falcon 9 rockets. Other rocket launch providers typically charge around $165 million for the same service. Space X can charge less because they are able to successfully land the first stage of the rocket sometimes and reuse it thereby saving cost. In order to accurately price each launch and respond to competitive bids, Space X needs to determine if the first stage will land.
- The goal of this study was to analyze launch data to understand what factors influence a successful first stage landing, create a dashboard to help illustrate the results, and train a machine learning model to predict whether a specific landing will be successful or not.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Requested the Space X API and cleaned the data to get it in the correct format for use
- Perform data wrangling
  - Converted Outcome descriptions into labels: “1” for successful land and “0” for unsuccessful land
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Performed GridCV() to determine the optimum hyperparameters of Logistic Regression, Support Vector Machine, Decision Tree, and K-nearest Neighbors classification models
  - Split dataset into train and test subsets and created models of each type with training set
  - Compared the accuracy and Confusion matrix of each model for evaluation

Data Collection

- Used Space X REST API to collect launch data: api.spacexdata.com/v4/launches/past
- The API includes information about dates, launch specifications, landing specifications, rocket type, payload mass, launch site, site location, landing outcomes, orbit, etc.
- BeautifulSoup used to web scrape Falcon 9 historical records from Wikipedia page
- Custom functions were applied to the API and Wikipedia page to extract the relevant information
- The datasets were formatted and converted into a Pandas Dataframes.

Data Collection – SpaceX API

Request Space X Launch Data

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

response = requests.get(spacex_url)
```

Convert Dataset to JSON File

```
static_json_url='https://cf-courses-data.s3.us.

response.json()

data=pd.json_normalize(response.json())
```

- Get Launch Functions from IDs
- Apply Custom Functions
- Construct Dictionary with Columns

```
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
```

```
launch_dict = {'FlightNumber':
(data['flight_number']), 'Date':
(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}
```

```
launch_dict_df=pd.DataFrame(launch_dict)
```

Convert To Pandas Dataframe and filter for Falcon 9 launches

```
data_falcon9=launch_dict_df.loc[launch_dict_df['BoosterVersion']!='Falcon 1']
```

Export to Csv File

```
data_falcon9.to_csv('dataset_part\_1.csv', index=False)
```

https://github.com/rhesabowning/IBM-Data-Science-Program/blob/master/Data%20Collection%20API.ipynb

Data Collection - Scraping

Use HTTP GET to request Falcon 9 Wiki Page

```
static_url = "https://en.wikipedia.org/w/index.
response=requests.get(static_url)
```

Convert HTML to BeautifulSoup Object

```
soup=BeautifulSoup(response.content)
```

Collect Column And Variable Names from HTML table header

```
html_tables=soup.find_all('table')
first_launch_table = html_tables[2]
first_launch_table_columns=first_launc

for th in first_launch_table_columns:
    extract_column_from_header(th)
    names.append(th)

for name in names:
    if name in names is not None and l
        column_names.append(name)
```

Create Empty Dictionary

```
# Let's initial the launch_dict
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']=[]
```

Extract Launch Records & Add To Rows

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(
rowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr")
        #check to see if first table
        if rows.th:
            if rows.th.string:
                flight_number=rows.t
                flag=flight_number.i
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells
        if flag:
```

Convert To Dataframe

df=pd.DataFrame(launch_dict)

Export to CSV File

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

https://github.com/rhesabowning/IBM-Data-Science-Program/blob/master/Data%20Collection%20With%20Web%20Scraping.ipynb

Data Wrangling

- The objectives were to perform Explanatory Data Analysis (EDA) on the Pandas dataframe and using the Outcome variable, to generate labels for training a supervised, classification model

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

Locate/Calculate % Missing Values In Attributes

```
df.isnull().sum()/df.count()*100
```

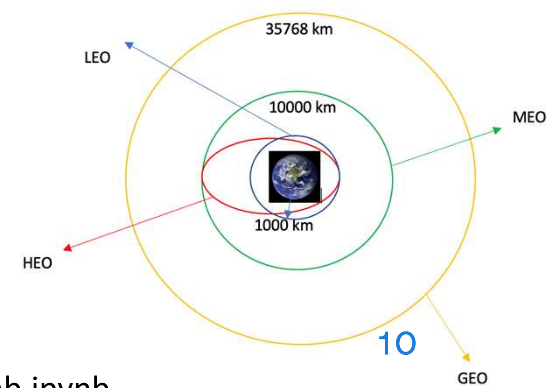
Identify Data Types of Attributes

```
FlightNumber      int64
Date             object
BoosterVersion   object
PayloadMass     float64
Orbit            object
LaunchSite       object
Outcome          object
Flights           int64
GridFins           bool
Reused             bool
Legs               bool
LandingPad       object
Block           float64
ReusedCount       int64
Serial           object
Longitude       float64
Latitude        float64
dtype: object
```

Calculate Descriptive Statistics: launches, orbits, outcomes

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

https://github.com/rhesabowning/IBM-Data-Science-Program/blob/master/EDA%20Lab.ipynb



Create Column 'Class' with outcome labels:
'1' = successful
'0' = unsuccesful

Export To CSV File

```
df.to_csv("dataset_part\_2.csv", index=False)
```

https://github.com/rhesabowning/IBM-Data-Science-Program/blob/master/EDA%20Lab.ipynb

EDA with Data Visualization

- Utilized Seaborn and Matplotlib libraries to produce charts for data visualization
  - Scatter plots were made to understand relationships between features
  - Bar charts were used to display success rate of each orbit, compare performances of categorical features
  - A line chart, time series plot, shows how the landing success rate changed over time
- Chose features to use for success prediction in the project
- Applied OneHotEncoding on categorical features, made dummy variables to contain the results

https://github.com/rhesabowning/IBM-Data-Science-Program/blob/master/EDA%20Lab%20With%20Visualization.ipynb

EDA with SQL

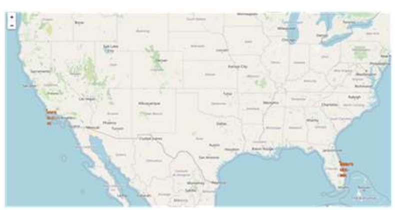
- Performed all tasks using sqlite3 library in Jupyter Notebook (include link for Notebook and IBM lab)
- Converted .csv file to database table & Opened a connection to SQL
- Performed multiple SQL queries on dataset
  - The names of unique launch sites in the space missions
  - 5 records with the launch site that starts with string “KSC”
  - Total payload mass carried by boosters launched by NASA (CRS)
  - Average payload mass carried by booster version F9 v1.1
  - Date of first successful landing outcome
  - Names of the booster_versions which have successfully landed on a ground pad with payload mass in the range 4000 – 6000 kg
  - Total number of successful and failed missions
  - Names of booster versions which have carried the maximum payload mass
  - List of records showing month names, successful landing_outcomes in ground pad, booster versions, launch_site for each month in 2017
  - Rank in descending order the count of successful landing_outcomes between 2010-06-04 and 2017-03-20

https://github.com/rhesabowning/IBM-Data-Science-Program/blob/master/EDA_Lab_with_SQL.ipynb

https://github.com/rhesabowning/IBM-Data-Science-Program/blob/master/EDA%20With%20SQL%20Lab.ipynb

Build an Interactive Map with Folium

- Folium was used to create several interactive maps to analyze launch sites:
  - Marking the location of each of the launch sites in the dataset
  - Marking the map with the successful and unsuccessful landings at each site
  - Marking the map with lines and calculating the proximity of launch sites to different map elements
    - Coastlines, highways, rail roads, cities
- Map objects were made to display information about the launch sites
  - Markers – used to identify the name of each launch site, landing outcome, number of launches at a site, and the label distances of elements from the launch site
  - Circles – used to display the exact location of the launch sites
  - Lines – used to visually indicate the distance being calculated between a site and a map element

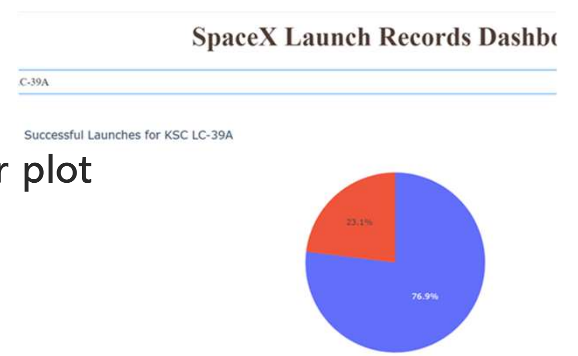


https://github.com/rhesabowning/IBM-Data-Science-Program/blob/master/Interactive%20Visual%20Analytics%20Lab.ipynb

Build a Dashboard with Plotly Dash

SPACEX

- Created a dashboard with 2 interactive plots using Plotly Dash
- Pie chart compares successful landing outcomes within and between launching sites
  - Added a dropdown menu to select all sites or one individual site
    - Selecting "All" shows the number of successful landings at each site
    - Selecting a site shows the number of successful and unsuccessful landings at that site
  - Pie chart with dropdown menu allows you to determine where most successful landings occur and which sites have the highest% and lowest% of successful landings
- Scatter plot shows relationship between Payload Mass and Successful Landing
  - Added a slide bar to change the Payload Mass range displayed in the scatter plot
  - Booster version is indicated on scatter plot by color and legend
  - Determine if payload mass makes a landing more/less likely to be successful
  - Determine which launch sites are better at landing smaller or larger payloads



https://github.com/rhesabrowning/IBM-Data-Science-Program/blob/master/spacex_dash_app.py

Predictive Analysis (Classification)

Build Classification Models

- Load cvs files into dataframes
- Create Numpy array from 'Class' column
- Standardize X data
- Perform test_train_split on data
- Models: Logistic Regression, Support Vector Machine, Decision Tree, K-nearest Neighbors

Evaluate

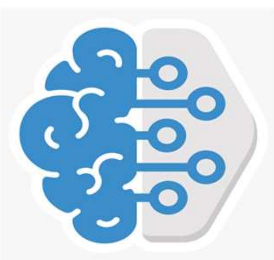
- Calculate accuracy of models fit to X, Y

Improve

- Determine best performing model parameters with GridSearchCV()

Determine Best Performing

- After Improve: calculate test accuracy, generate confusion matrix



https://github.com/rhesabowning/IBM-Data-Science-Program/blob/master/Machine%20Learning%20Predictive%20lab.ipynb

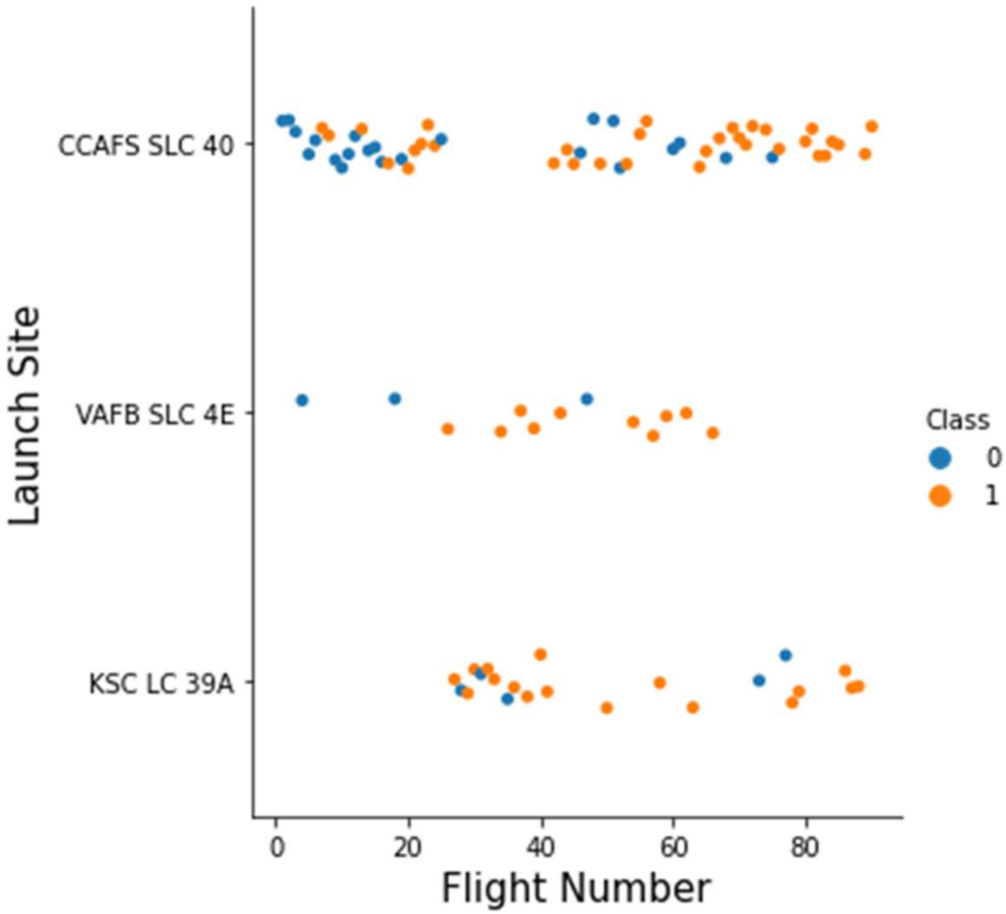
Results

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

Section 2

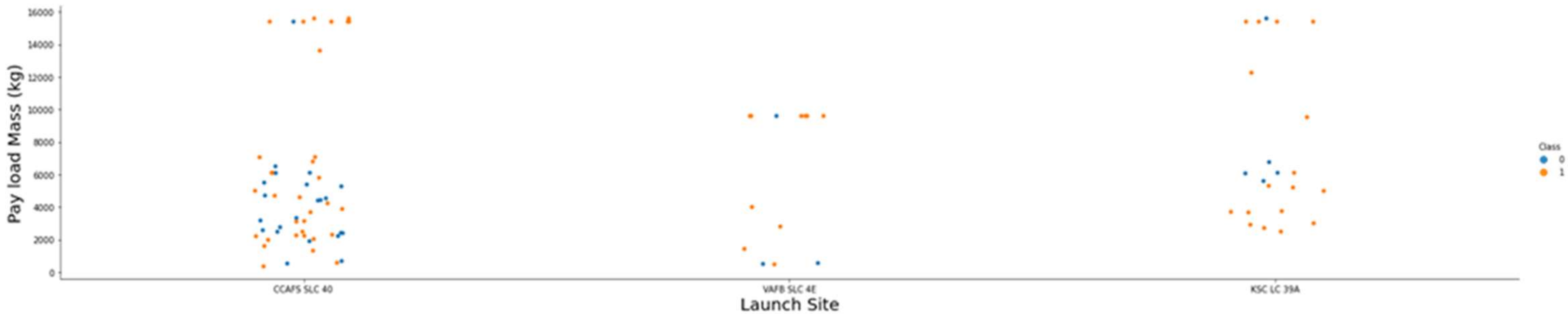
Insights drawn from EDA

Flight Number vs. Launch Site



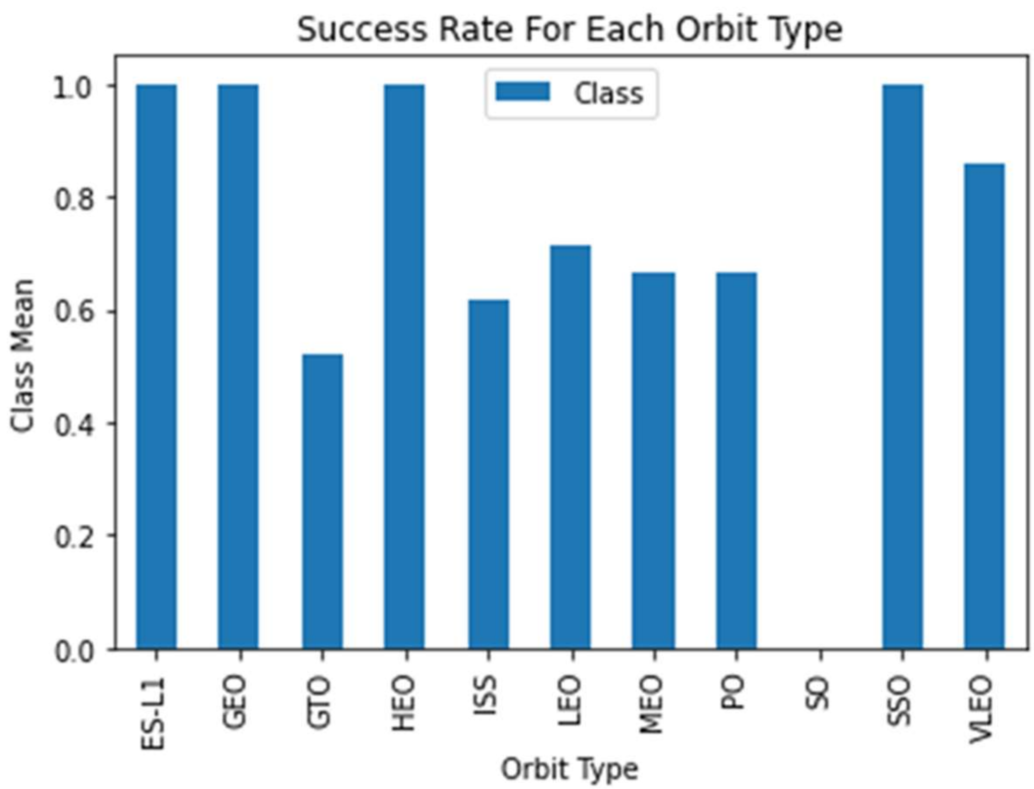
- CCAFS SLC 40 performed the most launches
- CCAFS SLC 40 site launched across the entire flight number range, but with one gap coinciding when the KSC LC 39A site started launches
- VAFB SLC 4E performed the least launches
- Each site became more successful at landing rockets as it launched more rockets

Payload vs. Launch Site



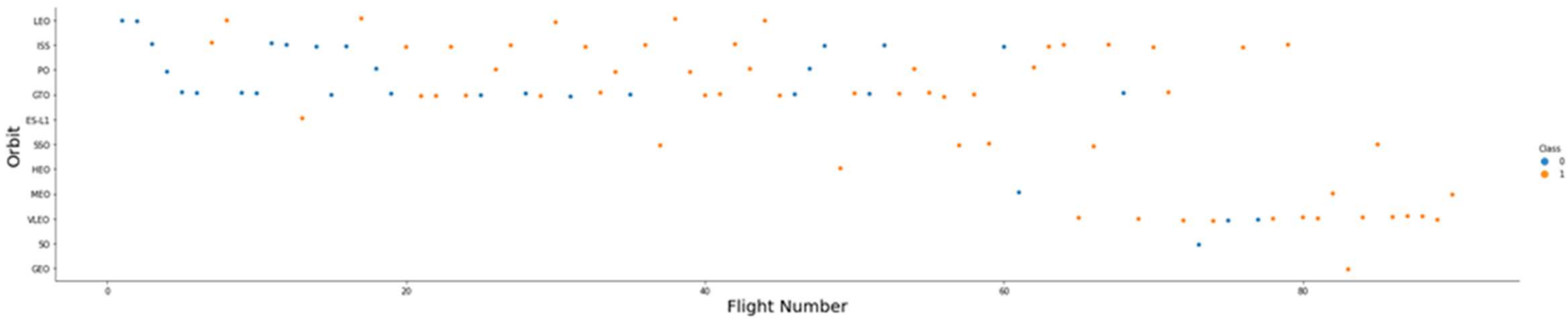
- Most payloads were <10,000 kg
- Launches with higher payloads had a higher landing success rate than lighter payloads
- CCAFS had the most launches overall, the most launches with payloads >10,000 kg, but the lowest overall success rate
- KSC had highest success rate for lighter payloads, but didn’t launch payloads >10,000 kg

Success Rate vs. Orbit Type



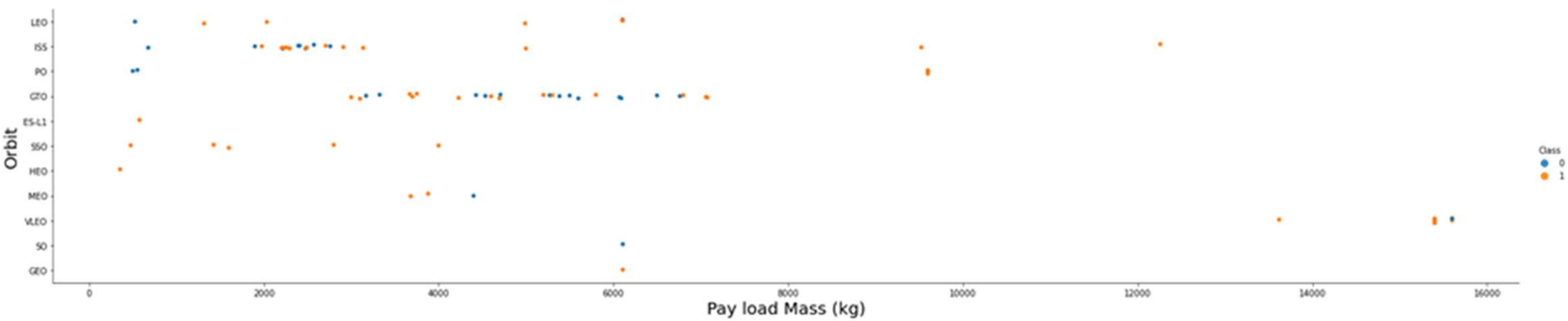
- There is 100% landing success rate for 4 orbit types: ES-L1, GEO, HEO, and SSO
- The orbit with lowest success rate is GTO at close to 50%
- VLEO has a success rate just above 80%

Flight Number vs. Orbit Type



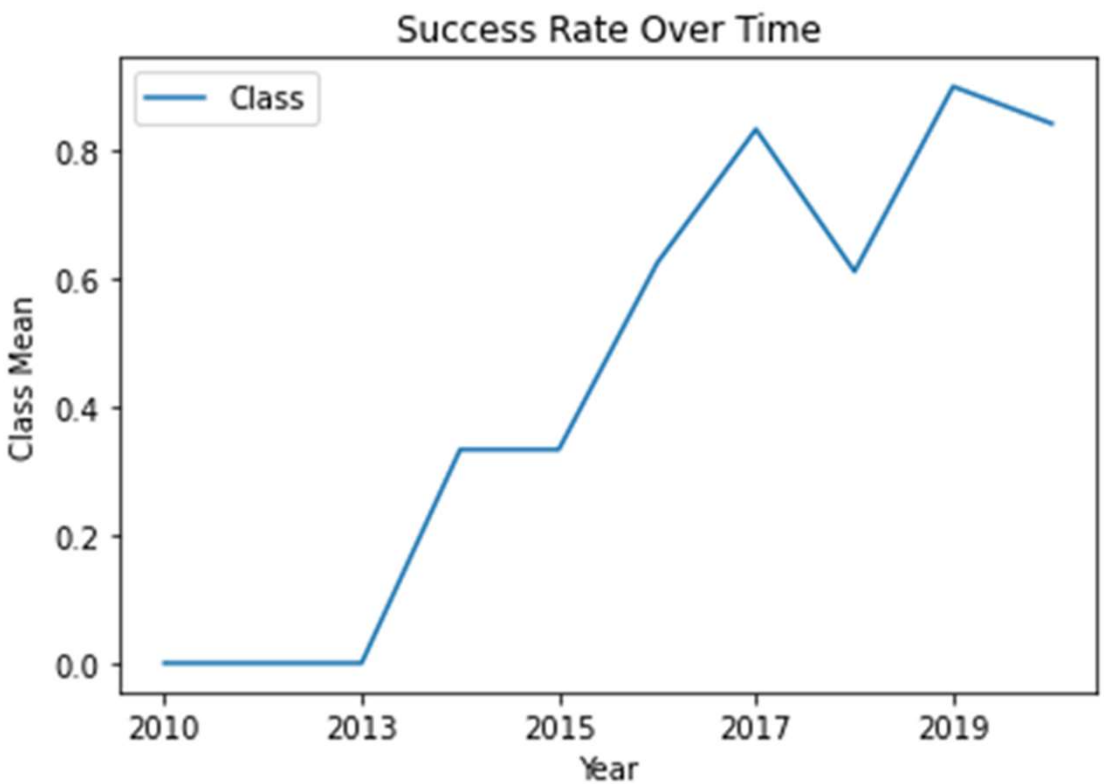
- Success rate goes up for each orbit as flight number increases
- Orbits ISS, PO, GTO all had landing failures in later launches
- Orbits LEO, SSO, HEO, VLEO, GEO all did not have failures after
- Orbits ES-L1, SO, GEO, HEO were only used 1 time
- SSO is only orbit with 100% landing success rate and more than one launch

Payload vs. Orbit Type



- The orbits used for payloads >8000 kg were ISS, PO and VLEO orbits
- Success rate increased with payload mass for LEO, ISS, and PO orbits
- Success rate did not improve as payload mass increased in GTO orbit
- PO was bimodal with successful heavy payloads, and unsuccessful lighter payloads

Launch Success Yearly Trend



- Success rate increased over time
- It took around 3 years before a successful launch was achieved
- The two biggest increases in success rate occurred in 2014 and 2016
- The success rate plateaued at around 80%

All Launch Site Names

```
In [75]: # Task 1 - Display the names of the unique launch sites in the space mission
         for row in cursor.execute('SELECT DISTINCT Launch_Site FROM Launches'):
             print(row)

('CCAFS LC-40',)
('VAFB SLC-4E',)
('KSC LC-39A',)
('CCAFS SLC-40',)
```

- Carried out all SQL queries in Jupyter Notebook using sqlite3 library
- In query called column name from the database ‘Launches’ and limited results to only unique values.

Launch Site Names Begin with 'KSC'

```
In [73]: # Task 2 - Display 5 records where launch sites begin with the string 'KSC'
         for row in cursor.execute('SELECT * FROM Launches WHERE Launch_Site Like "KSC%" LIMIT 5'):
             print(row)

('19-02-2017', '14:39:00', 'F9 FT B1031.1', 'KSC LC-39A', 'SpaceX CRS-10', 2490, 'LEO (ISS)', 'NASA (CRS)', 'Success', 'Success (ground pad)')
('16-03-2017', '06:00:00', 'F9 FT B1030', 'KSC LC-39A', 'EchoStar 23', 5600, 'GTO', 'EchoStar', 'Success', 'No attempt')
('30-03-2017', '22:27:00', 'F9 FT  B1021.2', 'KSC LC-39A', 'SES-10', 5300, 'GTO', 'SES', 'Success', 'Success (drone ship)')
('01-05-2017', '11:15:00', 'F9 FT B1032.1', 'KSC LC-39A', 'NROL-76', 5300, 'LEO', 'NRO', 'Success', 'Success (ground pad)')
('15-05-2017', '23:21:00', 'F9 FT B1034', 'KSC LC-39A', 'Inmarsat-5 F4', 6070, 'GTO', 'Inmarsat', 'Success', 'No attempt')
```

- My EDA with SQL Lab asked for Launch sites starting with KSC
- Called all columns that met the Launch Site search criteria
- Used LIMIT function to call 5 records

Total Payload Mass

```
In [79]: #Task 3 Display the total payload mass carried by boosters launched by NASA (CRS)
         for row in cursor.execute('SELECT SUM(PAYLOAD_MASS__KG_) FROM Launches WHERE Customer = "NASA (CRS)"'):
             print(row)

(45596,)
```

- Used SUM functions and WHERE functions to add up payloads that NASA (CRS) launched
- The total was 45,596 kg

Average Payload Mass by F9 v1.1

```
In [82]: #Task 4 Display average payload mass carried by booster version F9 v1.1
         for row in cursor.execute('SELECT AVG(PAYLOAD_MASS__KG_) FROM Launches WHERE Booster_Version = "F9 v1.1"'):
             print(row)

(2928.4,)
```

- Used the AVG function to average payload mass in query
- Used WHERE function and = condition to specify booster version F9 v1.1
- The average payload of the F9 v1.1 was 2928.4 kg

First Successful Ground Landing Date

```
In [98]: # Task 5 List the date where the first succesful landing outcome in drone ship was acheived. Hint:Use min function
         for row in cursor.execute('SELECT MIN(Date) FROM Launches WHERE Landing_Outcome = "Success (drone ship)"'):
             print(row)

('06-05-2016',)
```

- Use MIN() function to find first occurrence of Success (drone ship) landing outcome
- The date returned was 06-05-2016, in dd-mm-yyyy format

Successful Drone Ship Landing with Payload between 4000 and 6000

```
In [99]: #Task 6 List the names of the boosters which have success in ground pad and have payload mass greater
         #than 4000 but less than 6000
         for row in cursor.execute('SELECT DISTINCT Booster_Version FROM Launches WHERE
                                  Landing_Outcome = "Success (ground pad)" AND PAYLOAD_MASS__KG_ >4000 AND PAYLOAD_MASS__KG_ <6000'):
             print(row)

         ('F9 FT B1032.1',)
         ('F9 B4 B1040.1',)
         ('F9 B4 B1043.1',)
```

- Used DISTINCT function to call only unique booster version outputs
- Limited results to Landing Outcomes of Success (ground pad)
- Add conditional statements t limit results to rows where payload mass was in the range 4000-6000 kg range

Total Number of Successful and Failure Mission Outcomes

```
In [111]: #Task 7 List the total number of successful and failure mission outcomes
for row in cursor.execute('SELECT Mission_Outcome, COUNT(Mission_Outcome) FROM Launches WHERE Mission_Outcome LIKE "Success%"'):
    print(row)
for row in cursor.execute('SELECT Mission_Outcome, COUNT(Mission_Outcome) FROM Launches WHERE Mission_Outcome LIKE "Failure%"'):
    print(row)

('Success', 100)
('Failure (in flight)', 1)
```

- Used two queries to generate the results for Mission_Outcome column
- Included all Successful outcomes with “Success%” WHERE condition
- Included all Failing outcomes with “Failure%” WHERE condition
- Results show 100 Successful missions and 1 Failing mission

Boosters Carried Maximum Payload

```
In [121]: #Task 8 List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
          for row in cursor.execute('SELECT DISTINCT Booster_Version, PAYLOAD_MASS__KG_ FROM Launches
                                    WHERE PAYLOAD_MASS__KG_=(SELECT MAX(PAYLOAD_MASS__KG_) FROM Launches)'):
              print(row)

          ('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)
```

- Used DISTINCT function to simplify list to only unique values
- Used subquery in the WHERE function so that list only included boosters which carried the maximum payload mass of 15,600 kg
- List includes 12 different booster versions

2017 Launch Records

```
In [130]: #Task 9 List the records which will display the month names, succesful landing_outcomes in ground pad ,
          # booster versions, launch_site for the months in year 2017
          for row in cursor.execute('SELECT Date, Booster_Version, Launch_Site, Landing_Outcome FROM Launches
                                     WHERE Landing_Outcome="Success (ground pad)" AND Date LIKE "%2017"'):
              print(row)

          ('19-02-2017', 'F9 FT B1031.1', 'KSC LC-39A', 'Success (ground pad)')
          ('01-05-2017', 'F9 FT B1032.1', 'KSC LC-39A', 'Success (ground pad)')
          ('03-06-2017', 'F9 FT B1035.1', 'KSC LC-39A', 'Success (ground pad)')
          ('14-08-2017', 'F9 B4 B1039.1', 'KSC LC-39A', 'Success (ground pad)')
          ('07-09-2017', 'F9 B4 B1040.1', 'KSC LC-39A', 'Success (ground pad)')
          ('15-12-2017', 'F9 FT  B1035.2', 'CCAFS SLC-40', 'Success (ground pad)')
```

- Created list with launches which had outcome of Success (ground pad) and occurred within 2017
- List displays the date, landing outcome, booster versions, and launch site for each result

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

```
In [148]: #Task 10 Rank the count of successful landing_outcomes between the date 2010-06-04 and 2017-03-20 in descending order.
          for row in cursor.execute('SELECT Landing_Outcome, COUNT(Landing_Outcome) FROM Launches
                                    WHERE Landing_Outcome LIKE "Success%" AND (Date>"2010-06-04") AND (Date<"2017-03-20")'):
              print(row)

          ('Success (ground pad)', 10)
```

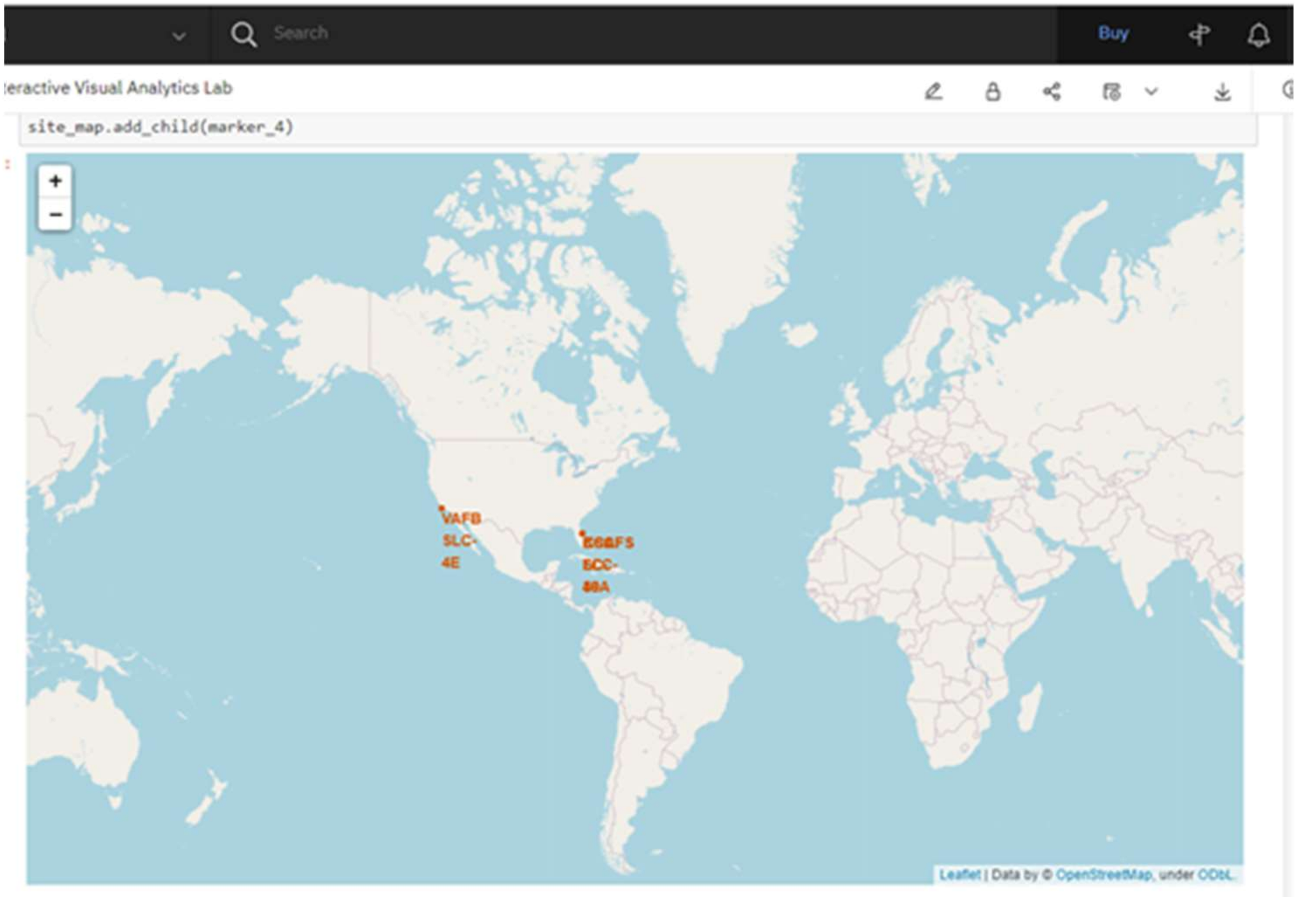
```
In [147]: #Task 10 Rank the count of successful landing_outcomes between the date 2010-06-04 and 2017-03-20 in descending order.
          for row in cursor.execute('SELECT Date, Landing_Outcome FROM Launches WHERE Landing_Outcome LIKE "Success%"
                                    AND (Date>"2010-06-04") AND (Date<"2017-03-20") ORDER BY Date DESC'):
              print(row)

          ('2017-03-06 00:00:00', 'Success (ground pad)')
          ('2017-02-19 00:00:00', 'Success (ground pad)')
          ('2017-01-14 00:00:00', 'Success (drone ship)')
          ('2017-01-05 00:00:00', 'Success (ground pad)')
          ('2016-08-14 00:00:00', 'Success (drone ship)')
          ('2016-08-04 00:00:00', 'Success (drone ship)')
          ('2016-07-18 00:00:00', 'Success (ground pad)')
          ('2016-06-05 00:00:00', 'Success (drone ship)')
          ('2016-05-27 00:00:00', 'Success (drone ship)')
          ('2015-12-22 00:00:00', 'Success (ground pad)')
```

- The instructions in the lab were unclear so I ran 2 different queries.
- The first query counted 10 successful landings occurred in the date range of June 4, 2010 – March 20, 2017
- The second query displayed the successful launches in the date range with their occurrence dates in descending order

Section 4

Launch Sites Proximities Analysis

Mark Launch Sites On Folium Map

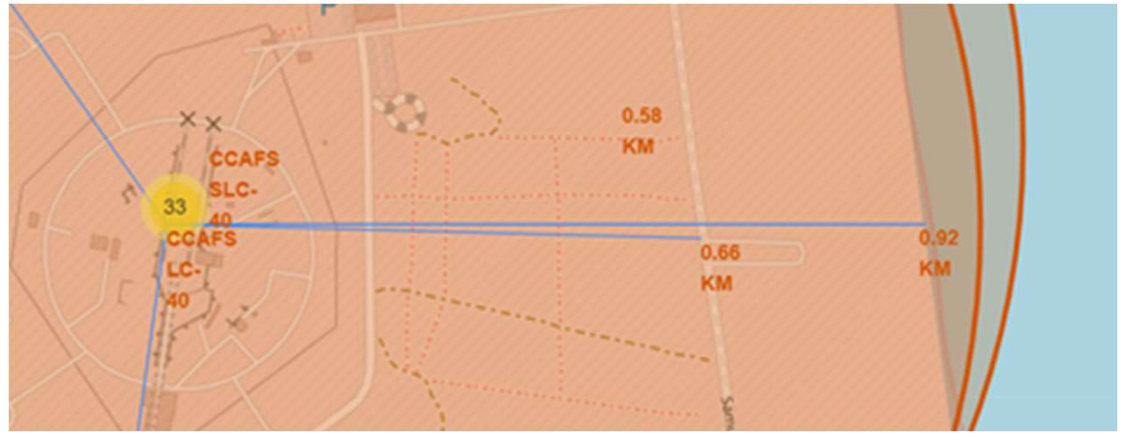


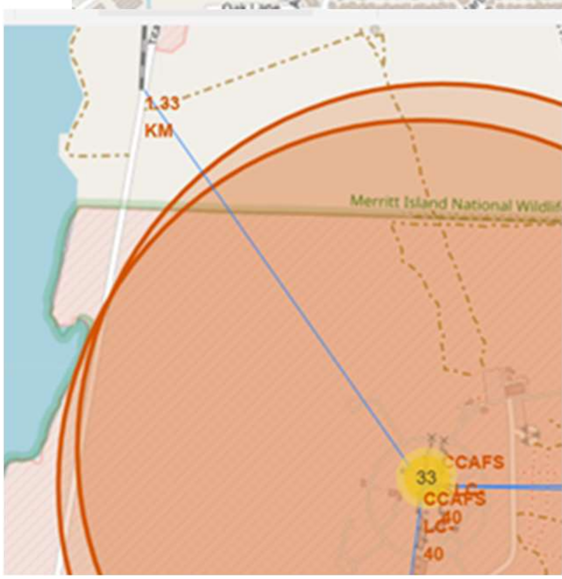
- The Space X launch sites are in the US, near the Pacific and Atlantic coasts, specifically in California and Florida

Mark Outcomes With Colored Labels

- The Green markers show successful launches
- The Red markers show unsuccessful launches
- Most of the rockets were launched in Florida (46 vs. 10)
- The CCAFS LC 40 site launched the most rockets (26 vs. 13, 7)
- The KSC LC-39A site launched the most rockets successfully (10)

Distances From Launch Site To Landmarks

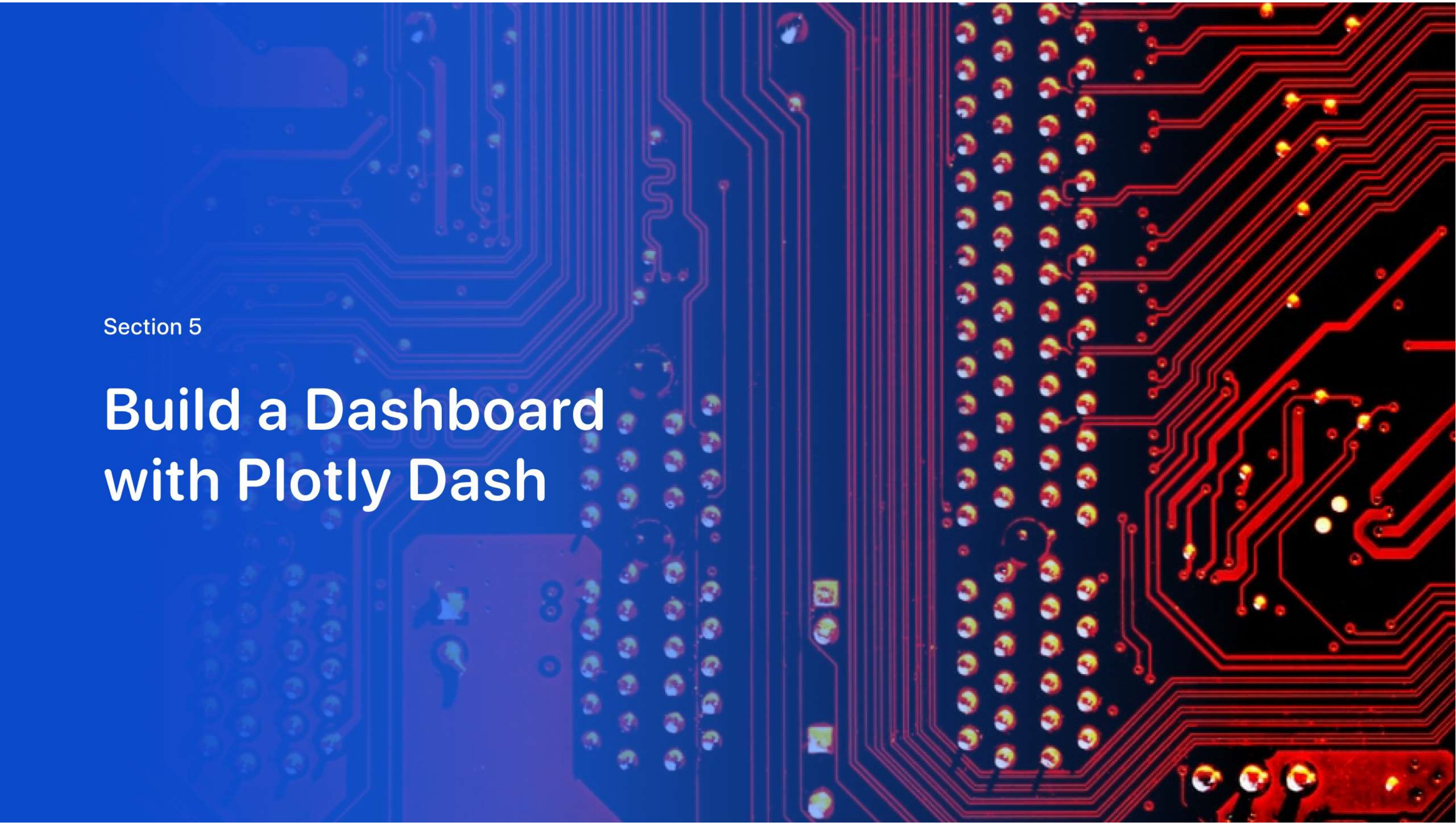




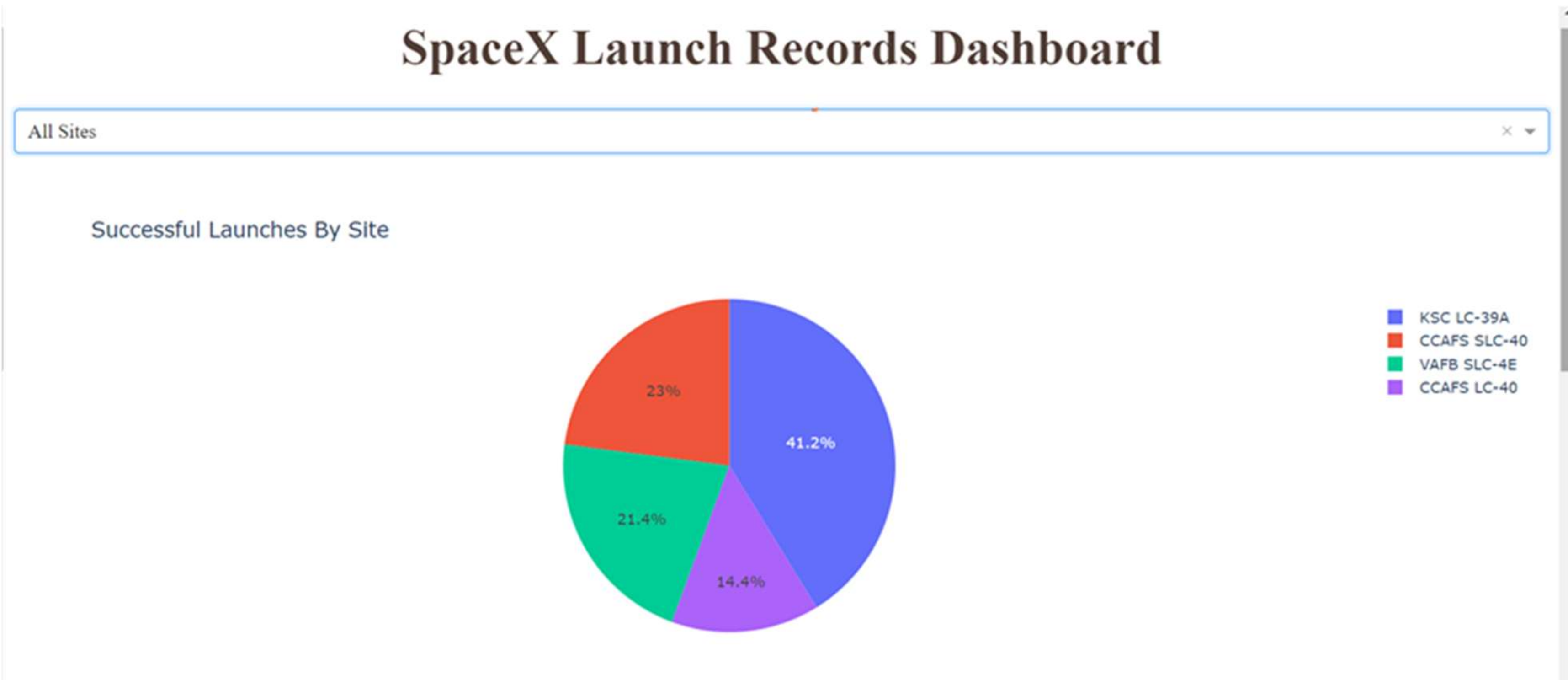
- Are launch sites in close proximity to railways? Yes, need to transport equipment and materials efficiently to launch site
- Are launch sites in close proximity to highways? Yes, need to transport equipment and materials efficiently to launch site
- Are launch sites in close proximity to coastline? Yes, provides options for landing surface if needed
- Do launch sites keep certain distance away from cities? Yes, for safety

Section 5

Build a Dashboard with Plotly Dash

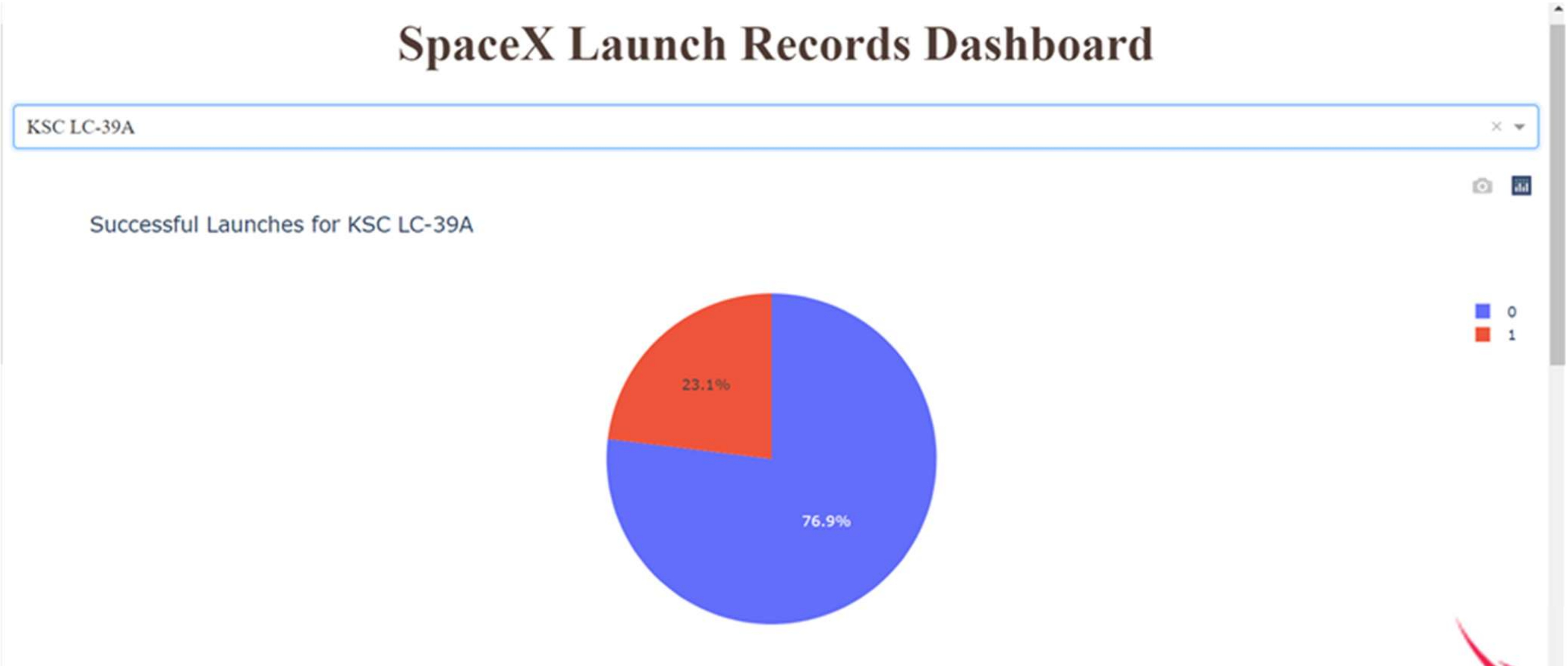


Dashboard Pie Chart For All Sites Successful Launches



- Shows that KSC LC-39A had the most successful launches for All sites
- Easy to understand comparison for successful launches at All sites

Dashboard Pie Chart Of KSC LC 39-A Success Rate



- Looking deeper into KSC LC 39-A performance since it was the site with the most successful launches
- This site's launches were successful 76.9% of the time.

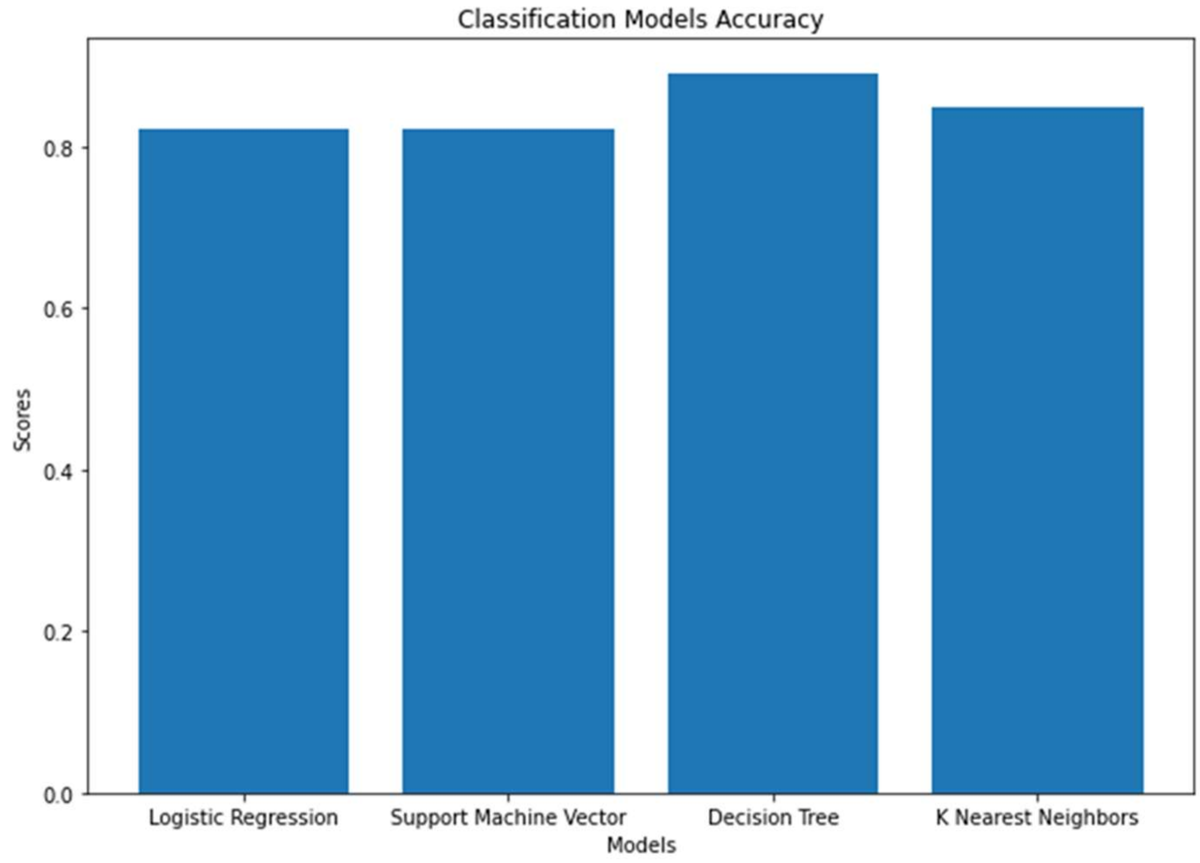
Dashboard Scatter Plot of Relationship Between Payload Mass and Launch Outcome For All Sites

- There were more successful launches in the 2500 – 5000 kg payload range than the 5000 – 7500 kg range
- The success rate in the 2500 – 5000 kg range was >50%
- The success rate in the 5000 – 7500 kg range was <50%

Section 6

Predictive Analysis (Classification)

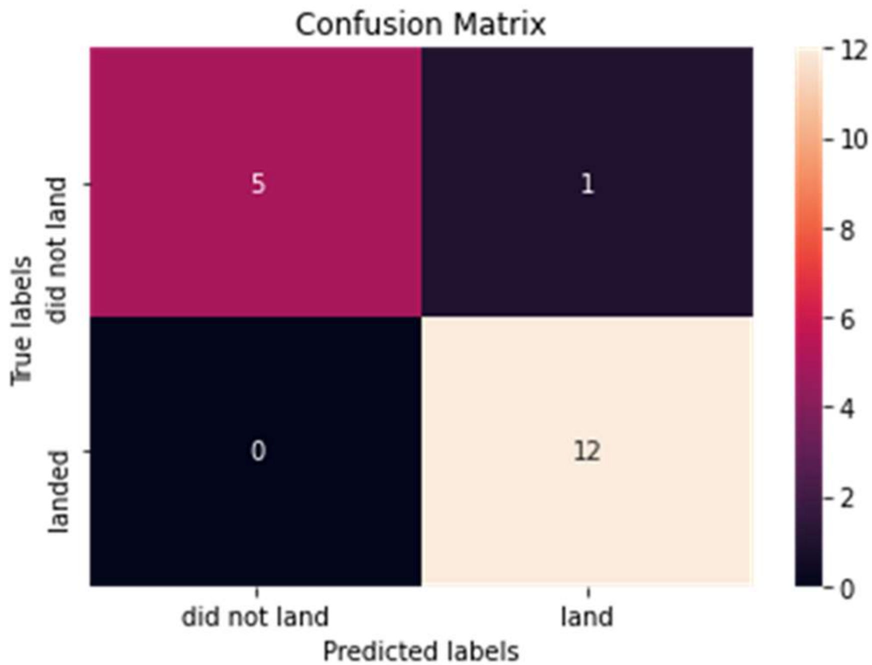
Classification Accuracy



- The Decision Tree Classifier model has the highest training accuracy at 88.9%

Confusion Matrix

Logistic Regression & SVM



Decision Tree & K Nearest Neighbors

- The confusion matrix of the model with the highest accuracy score, Decision Tree, shows that 16.7% of its predictions will be false positives (predicts land, but does not land), which is a concern

Conclusions

- Success rate improved in general over time and as more launches were attempted per site, payload, and orbit
- KSC LC 39A site launched the most rockets successfully and achieved the highest success rate
- CCAFS SLC 40 launched the most rockets and was the first site used by Space X
- Success rate was higher for 2500-5000 kg payloads than for 5000-7500 payloads
- Orbits ES-L1, GEO, HEO, SSO, and VLEO all have high success rates, >80%
- However, orbits ES-L1, GEO, and HEO were all only used 1 time
- The Decision Tree Classifier Model was the most accurate with a score of 88.9%

<u>2nd Order Conclusions</u>

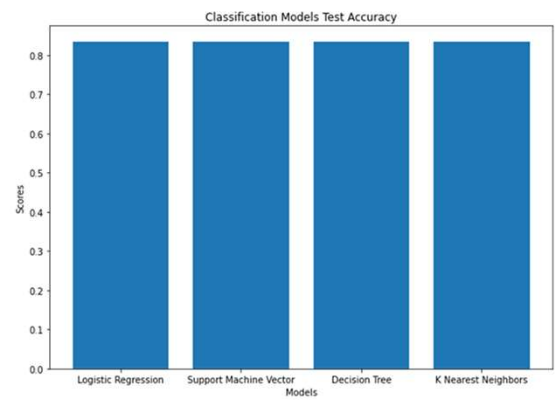
- Higher successful levels may be possible by finding the optimum combination of Site, Orbit, and Payload to use for a specific launch
- Perform a study using predictions to determine what that combination could be
- Recommend using the Support Vector Machine model instead of the Decision Tree model. It has a lower false negative rate (5.6% vs. 16.7%).
- The goal of the model is to use it as a tool to increase stage 1 rocket reuse. False negative predictions are the greatest barrier to that

Appendix

- Haversine Formula – calculate distances of earth's surface

```python
from math import sin, cos, sqrt, atan2, radians

def calculate_distance(lat1, lon1, lat2, lon2):
    # approximate radius of earth in km
    R = 6373.0

    lat1 = radians(lat1)
    lon1 = radians(lon1)
    lat2 = radians(lat2)
    lon2 = radians(lon2)

    dlon = lon2 - lon1
    dlat = lat2 - lat1

    a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
    c = 2 * atan2(sqrt(a), sqrt(1 - a))

    distance = R * c
    return distance
```

Bar chart of test accuracy



Additional Dashboard Screenshot


Thank you!