IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Mary Gutierrez
April 18, 2022

Table of Contents

Executive Summary

- Methodologies:

  Data was obtained from SpaceX REST API and Wikipedia. Relevant information was merged, independent categorical variables were changed to binary variables with one hot encoding, and a binary target variable was created (successful vs. unsuccessful landing). Feature relationships were explored with SQL queries and visualizations, both static and interactive. Finally, four different classification models were tested, Logistic Regression, SVM, Decision Tree, and K Nearest Neighbors.

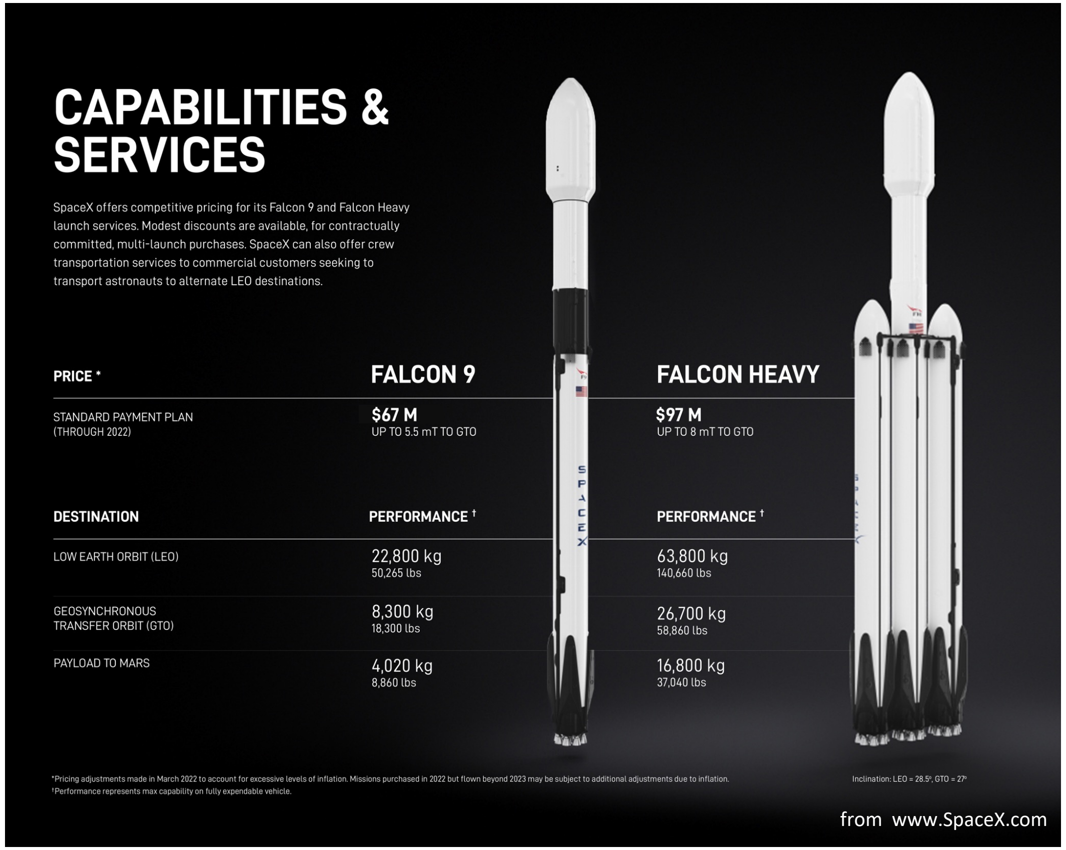
- Results:

  Some interesting relationships were demonstrated during exploratory data analysis, especially the relationship between flight number and successful landing. Predictive analysis with optimal parameters for the Decision Tree achieved an accuracy score of 0.9444. This is a very good score, but should be further verified with more data as it is available.

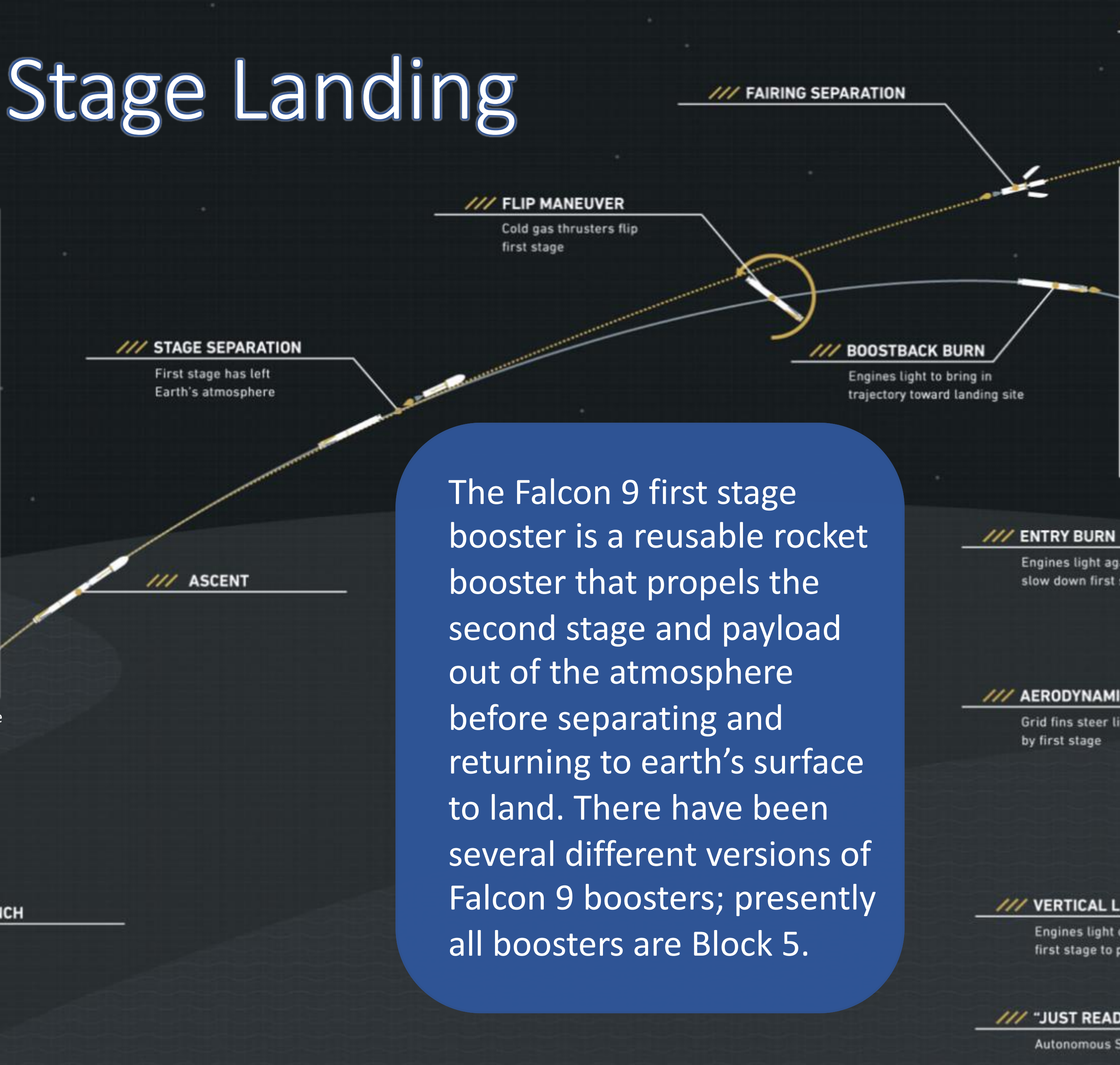
Introduction

In 2018 SpaceX became the leading global commercial launch provider due to the company's ability to provide launch services at a fraction of the cost of its competitors. Reusability of the first stage in their Falcon 9 and Falcon Heavy rockets is a major factor in their ability to reduce costs.

We want to determine if prediction of successful landing of the first stage, and therefore prediction of the cost of the launch, is possible given publicly available data.



First Stage Landing



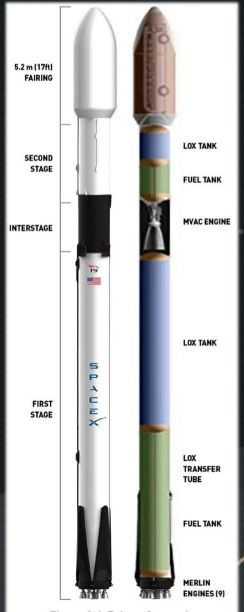
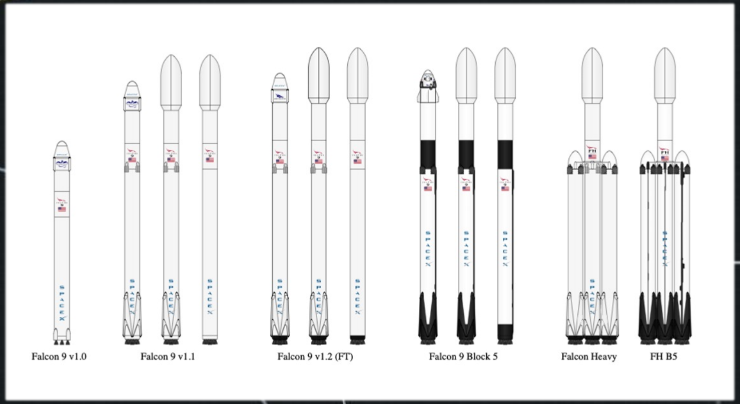


Figure 2-1: Falcon 9 overview

from the Falcon User Guide



By Lucabon (based on work of Markus Säynevirta and Craigboy and Rressi) - Falcon rocket family4.svg, CC BY-SA 4.0,

The Falcon 9 first stage booster is a reusable rocket booster that propels the second stage and payload out of the atmosphere before separating and returning to earth's surface to land. There have been several different versions of Falcon 9 boosters; presently all boosters are Block 5.

from the Falcon User Guide

Section 1

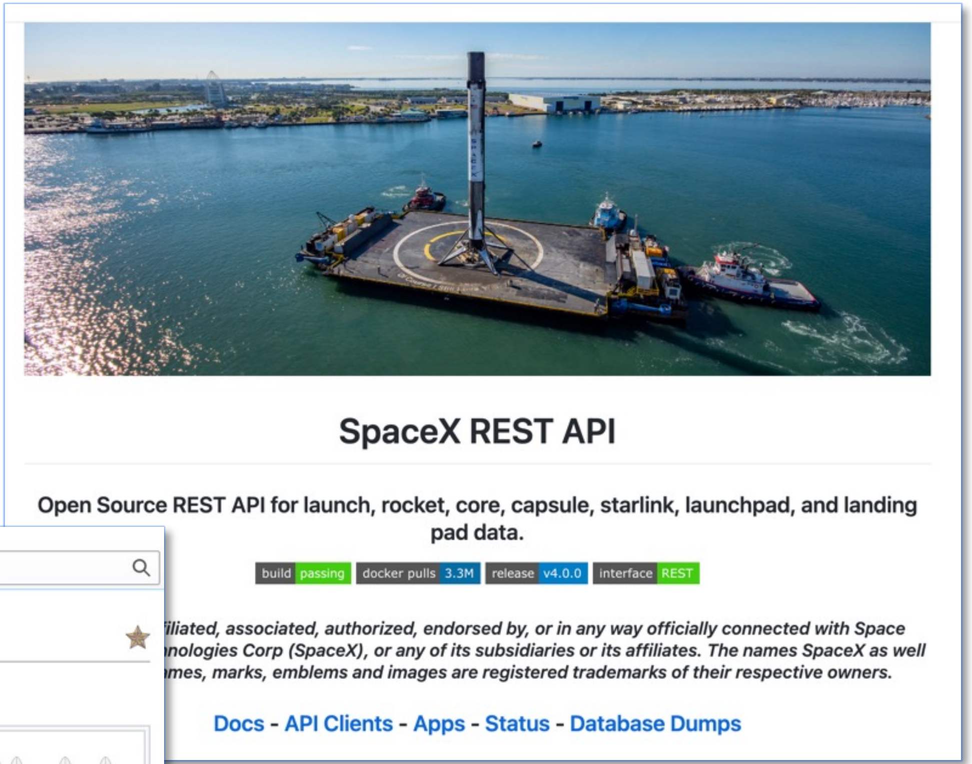
Methodology

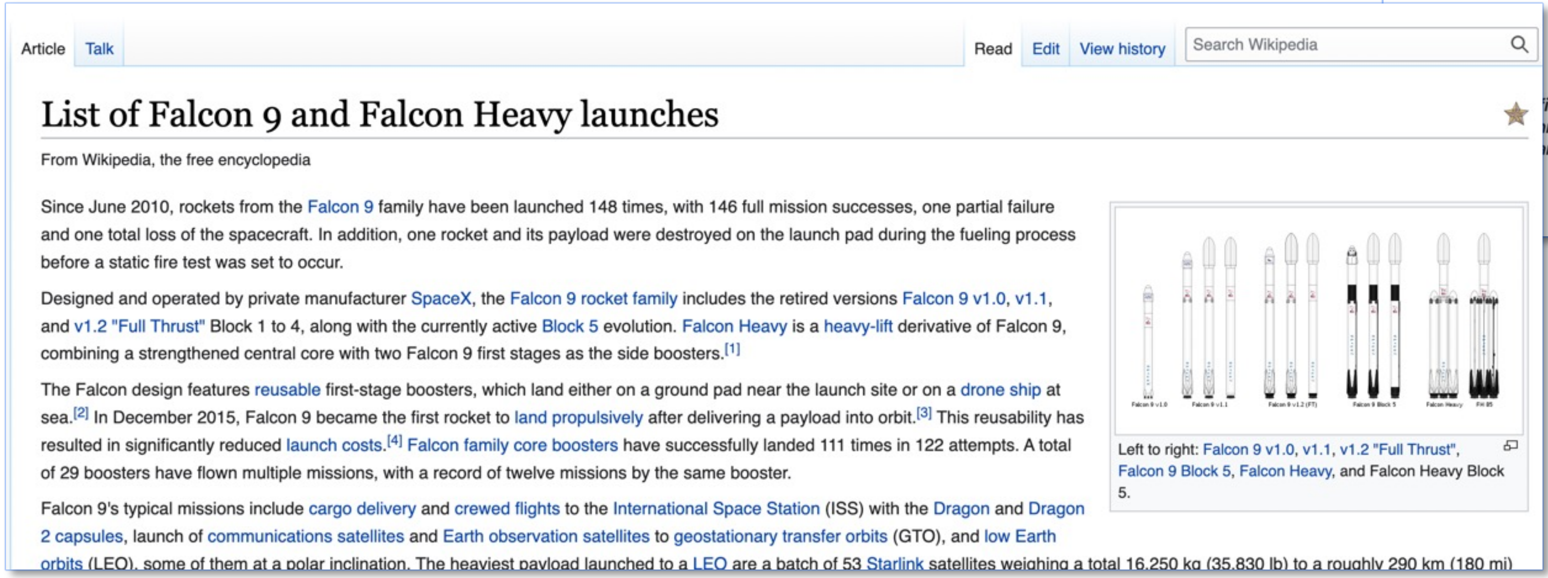
Methodology

- Data collection methodology:
  - Data collected from both the SpaceX REST API and Wikipedia
- Perform data wrangling
  - Training label ‘Class’ created by categorizing launches as successful or unsuccessful.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Try four different classification models using GridSearchCV to choose the best parameters

Data Collection

The data for this analysis was collected from SpaceX REST API and from tables in the Wikipedia article “List of Falcon 9 and Falcon Heavy Launches.”





Article Talk Read Edit View history Search Wikipedia

List of Falcon 9 and Falcon Heavy launches

From Wikipedia, the free encyclopedia

Since June 2010, rockets from the Falcon 9 family have been launched 148 times, with 146 full mission successes, one partial failure and one total loss of the spacecraft. In addition, one rocket and its payload were destroyed on the launch pad during the fueling process before a static fire test was set to occur.

Designed and operated by private manufacturer SpaceX, the Falcon 9 rocket family includes the retired versions Falcon 9 v1.0, v1.1, and v1.2 "Full Thrust" Block 1 to 4, along with the currently active Block 5 evolution. Falcon Heavy is a heavy-lift derivative of Falcon 9, combining a strengthened central core with two Falcon 9 first stages as the side boosters.[1]

The Falcon design features reusable first-stage boosters, which land either on a ground pad near the launch site or on a drone ship at sea.[2] In December 2015, Falcon 9 became the first rocket to land propulsively after delivering a payload into orbit.[3] This reusability has resulted in significantly reduced launch costs.[4] Falcon family core boosters have successfully landed 111 times in 122 attempts. A total of 29 boosters have flown multiple missions, with a record of twelve missions by the same booster.

Falcon 9's typical missions include cargo delivery and crewed flights to the International Space Station (ISS) with the Dragon and Dragon 2 capsules, launch of communications satellites and Earth observation satellites to geostationary transfer orbits (GTO), and low Earth orbits (LEO), some of them at a polar inclination. The heaviest payload launched to a LEO are a batch of 53 Starlink satellites weighing a total 16,250 kg (35,830 lb) to a roughly 290 km (180 mi)

Left to right: Falcon 9 v1.0, v1.1, v1.2 "Full Thrust", Falcon 9 Block 5, Falcon Heavy, and Falcon Heavy Block 5.

Data Collection – SpaceX API

A request was sent to https://api.spacexdata.com/v4/launches/past to get rocket launch data. Custom functions were applied to retrieve and append information about the booster version, launch site location, payload mass, and orbit for each launch.

GitHub URL: https://github.com/maryprimer/IBMCapstone/blob/main/Data Collection API Lab.ipynb

1. Get response from API

```
[ ]: spacex_url="https://api.spacexdata.com/v4/launches/past"
     response = requests.get(spacex_url)
```

2. Convert response to a JSON file and then a Pandas dataframe

```
[ ]: jFile=response.json()
     data=pd.json_normalize(jFile)
```

3. Apply custom functions to append data

```
[ ]: getBoosterVersion(data)
     getLaunchSite(data)
     getLaunchSite(data)
     getPayloadData(data)
```

4. Assign list to dictionary and 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 flat file

```
[ ]: data_falcon9=data[data['BoosterVersion']!='Falcon 1']
     data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data Collection - Scraping

Launch data was extracted from a Wikipedia page, and then BeautifulSoup was used to parse it into a dictionary while accounting for noise in the table. The dictionary was then converted to a dataframe and exported as a csv file.

GitHub URL:
https://github.com/maryprimer/IBMCapstone/blob/main/Data%20Collection%20with%20Web%20Scraping.ipynb

1. Get HTML text from Wikipedia page and create BeautifulSoup object

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_H"
data=requests.get(static_url).text
soup=BeautifulSoup(data, "html5lib")
```

2. Find the tables in the HTML text using the find_all method, assign the first launch table to a variable

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

3. Use find_all method to loop through table headers and extract column names using a custom function and assign them to dictionary keys

```
for head in first_launch_table.find_all(name='th'):
    name=extract_column_from_header(head)
    if name is not None and len(name)>0:
        column_names.append(name)
launch_dict= dict.fromkeys(column_names)
```

4. Parse table information into the dictionary

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable pl
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading has a 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
            launch_dict['Flight No.'].append(flight_number)
            datatimelist=date_time(row[0])
            date = datatimelist[0].strip(',')
            launch_dict['Date'].append(date)
            time = datatimelist[1]
```

5. Convert dictionary to a dataframe and export to a CSV file

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

Data Wrangling

The number of launches at each site and orbit and the occurrence of each landing outcome were calculated. Then the training label ‘Class’ was determined by categorizing the values for landing outcome as successful (1) or not successful (0).

GitHub URL:
https://github.com/maryprimer/IBMCapstone/blob/main/Data%20Wrangling%20Lab.ipynb

1. Calculate the number of launches at each site

```
df['LaunchSite'].value_counts()
```

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

2. Calculate the number of launches into each orbit

```
df['Orbit'].value_counts()
```

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

3. Calculate the occurence of each landing outcome

```
landing_outcomes=df['Outcome'].value_counts()
landing_outcomes
```

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

4. Create a set of unsuccessful outcomes

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

```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS',
'None None'}
```

5. Create label 'Class' from 'Outcome' assiging 0 or 1 to unsuccessful or successful outcomes

```
df['Class'] = df['Outcome'].apply(lambda x: 0 if (x in bad_outcomes)  else 1)
df.head()
```

lights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.561857	0
1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561857	0
1	False	False	False	NaN	1.0	0	B0007	-80.577366	28.561857	0

EDA with Data Visualization

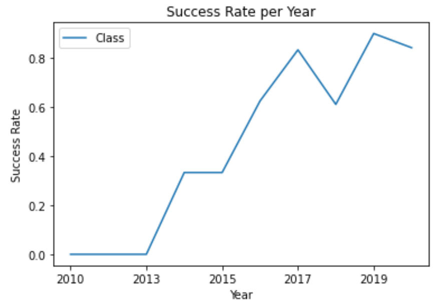
Exploratory data analysis was performed to look at the relationships of different variables.

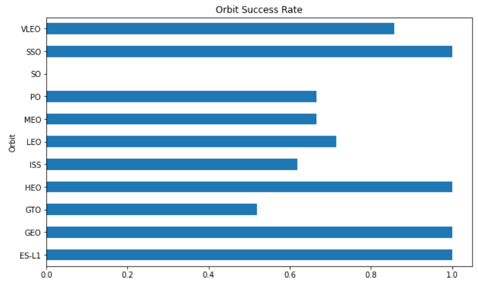
The visualizations of these relationships helped to illustrate how these variables correlate to landing success as well as to each other.

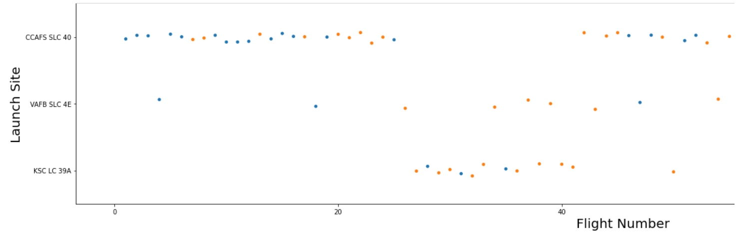
Relationships explored include:

- Landing Success, Flight Number, and Launch Site
- Landing Success, Payload, and Launch Site
- Success rate with respect to different orbit types
- Landing Success, Flight Number, and Orbit Type
- Landing Success, Payload, and Orbit Type
- Yearly landing success rates

All can be viewed in the Section 2.







GitHub URL:
https://github.com/maryprimer/IBMCapstone/blob/main/Exploratory%20Data%20Analysis%20with%20Visualization.ipynb

EDA with SQL

Exploratory data analyses was also performed using SQL queries on a dataset uploaded to a Db2 database.

The following queries can be found in Section 2:

- Unique launch site names
- Records with launch site names that begin with 'CCA'
- Total payload mass carried by boosters launched by NASA
- Average payload mass carried by booster version F9 v1.1
- Date of first successful landing outcome on a ground pad
- Names of boosters that have landing success on a drone ship and carried a payload mass between 4000kg and 6000kg
- Total number of failure and success mission outcomes
- Names of the booster versions that have carried the maximum payload mass
- Date, booster version, and launch site names for failed drone ship landings in 2015
- Ranked count of landing outcomes between 6/4/2010 and 03/20/2017

TIME__UTC_	BOOSTER_VERSION	LAUNCH_SITE	PAYLOAD	PAYLOAD_MASS__KG_	ORBIT	CUSTOMER	MISSION_OUTCOME	LANDING__C
18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (para
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 (para
07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX	677	LEO (ISS)	NASA (CRS)	Success	No attempt

GitHub URL:
https://github.com/maryprimer/IBMCapstone/blob/main/Exploratory%20Data%20Analysis%20with%20SQL.ipynb

Build an Interactive Map with Folium

Folium was used to create interactive maps of the SpaceX launch sites in order to explore what factors contribute to an optimal launch site location and to further the exploration of the relationship between launch site and successful landing.

- Circles with pop up labels were used to mark the different launch sites.
- Marker clusters at the different launch sites were color coded to denote successful and unsuccessful outcomes.
- Distance markers were plotted at various points, with the help of a custom function to calculate distance, and connected to the launch sites with lines to illustrate launch site proximity to certain objects.

Examples of these maps can be found in the Section 3.

GitHub URL:

https://github.com/maryprimer/IBMCapstone/blob/main/Data%20visualization%20with%20Folium2.ipynb

Build a Dashboard with Plotly Dash

- The dashboard has a dropdown menu for choosing all or only one of the four launch sites.
- The pie chart shows either the percentages of the total successful launches for all sites, or the ratio of successful to unsuccessful launches for a chosen site.
- The range slider is for choosing the payload range for the scatterplot.
- The scatterplot shows either the correlation between payload and success over all of the sites, or the correlation between payload and success for a chosen site.

This dashboard allows for quick investigation into the various relationships between success, launch site, and payload which can help in building our predictive models and understanding their results. Screen shots of the dashboard can be found in Section 4.

GitHub URL: https://github.com/maryprimer/IBMCapstone/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

Four different classification models were fitted with the best parameters to achieve their highest accuracy scores.

Dictionaries of parameters to be tested were input to the GridSearchCV method, which used ten-fold cross validation and the 'accuracy' scoring method to find which parameters to use for the best fit.

Finally, the **score** method was used to score each model on the test set, and confusion matrices were constructed.

Logistic Regression

SVM

Decision Tree

K Nearest Neighbors

GitHub URL:
https://github.com/maryprimer/IBMCapstone/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Predictive Analysis: Logistic Regression

Three different values for C were tried when fitting the logistic regression model.

The best value for C was .01 and the test set score was 0.83333.

Logistic Regression

1. Create logistic regression object

```
lr=LogisticRegression()
```

2. Create parameter variable

```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}
```

3. Fit model using GridSearchCV with parameters variable

```
logreg_cv= GridSearchCV(lr,
                        param_grid=parameters,
                        scoring='accuracy',
                        cv=10)
logreg_cv.fit(X_train, Y_train)
```

4. Get best parameters and test set score

```
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
print("Test score: ", logreg_cv.score(X_test, Y_test))
```

```
tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver':
'lbfgs'}
Test score:  0.8333333333333334
```

Predictive Analysis: Support Vector Machine

Five different values between -3 and 3 were tried for **C** and **gamma**, along with five different **kernels** when fitting the support vector model.

The best value for **C** was 1.0, for **gamma** was 0.0316, and the best **kernel** was ‘sigmoid’. The test set score was 0.83333.

Support Vector Machine

```
svm = SVC()
```

1. Create SVM object

2. Create parameter variable

```
parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
              'C': np.logspace(-3, 3, 5),
              'gamma':np.logspace(-3, 3, 5)}
```

```
svm_cv= GridSearchCV(svm,
                     param_grid=parameters,
                     scoring='accuracy',
                     cv=10)
svm_cv.fit(X_train, Y_train)
```

3. Fit model using GridSearchCV with parameters variable

4. Get best parameters and test set score

```
print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
print("Test score: ", svm_cv.score(X_test, Y_test))
```

```
tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379,
'kernel': 'sigmoid'}
Test score:  0.8333333333333334
```

Predictive Analysis: Decision Tree

Two different **criterion**, **splitter**, and **maximum features** parameters were tried along with three different **minimum leaf** and **split samples** parameters, and ten **maximum range** parameters.

The best value for **criterion** was 'gini', for **splitter** was 'best', for **maximum features** was 'sqrt', for **minimum leaf samples** was 4, **minimum split samples** was 5, and the best **maximum depth** was 8. Test set score was 0.94444.

Decision Tree

1. Create decision tree object

```
tree = DecisionTreeClassifier()
```

2. Create parameter variable

```
parameters = {'criterion': ['gini', 'entropy'],
     'splitter': ['best', 'random'],
     'max_depth': [2*n for n in range(1,10)],
     'max_features': ['auto', 'sqrt'],
     'min_samples_leaf': [1, 2, 4],
     'min_samples_split': [2, 5, 10]}
```

3. Fit model using GridSearchCV with parameters variable

```
tree_cv= GridSearchCV(tree,
                      param_grid=parameters,
                      scoring='accuracy',
                      cv=10)
tree_cv.fit(X_train, Y_train)
```

4. Get best parameters and test set score

```
print("tuned hyperparameters :(best parameters) ",tree_cv.best_params_)
print(tree_cv.score(X_test, Y_test))
```

```
tuned hyperparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 8, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 5, 'splitte
r': 'best'}
0.9444444444444444
```

Predictive Analysis: K Nearest Neighbors

Four different **algorithms**, **p** equal to one and two, and **number of neighbors** equal to one through ten were tried when fitting the K Nearest Neighbors model.

The best value for **algorithm** was 'auto', for **number of neighbors was** 10, and for **p** was 1. The test set score was 0.83333.

K Nearest Neighbors

```
KNN = KNeighborsClassifier()
```

1. Create KNN object

2. Create parameter variable

```
parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
              'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
              'p': [1,2]}
```

```
KNN_cv= GridSearchCV(KNN,
                     param_grid=parameters,
                     scoring='accuracy',
                     cv=10)
KNN_cv.fit(X_train, Y_train)
```

3. Fit model using GridSearchCV with parameters variable

4. Get best parameters and test set score

```
print("tuned hpyerparameters :(best parameters) ",KNN_cv.best_params_)
print("Test set score: ", KNN_cv.score(X_test, Y_test))
```

```
tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors':
10, 'p': 1}
Test set score:  0.8333333333333334
```

Results: Exploratory Data Analysis

Exploratory data analysis revealed a number of interesting trends and relationships. The most obvious is the dependence of success on flight number. The first twenty flights were during the development phase of the landing process and had a very small percentage of successful landings.

Some of the other relationships explored were influenced by this. For instance, launches to some orbits were not attempted until after the initial development phase, and therefore the overall success rates for flights to those orbits were higher than the rates for flights to the orbits that were being launched to from the beginning.

Results: Interactive Analytics

Several interactive visualizations were used to explore differences between the four launch sites. The influences of geographical location and pay load mass on success rates were demonstrated with interactive Folium maps, scatter plots, and pie charts. It can be seen from the Folium maps that the success rates for the launch sites varied.

The dashboard scatter plots depicting the relationship between payload mass and success along with booster version reveal that not all versions were used at all of the launch sites and that the sites using the more recent booster versions had better overall success.

Results: Predictive Analysis

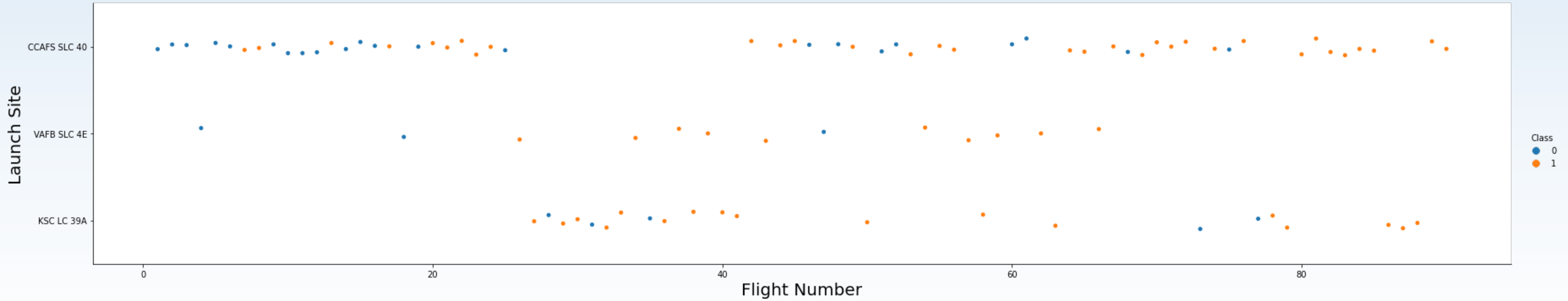
Four different classification models were used to predict landing success based on the given data. Features were chosen and the best parameters for the different models were determined with the help of GridSearchCV.

The results were similar, but the Decision Tree Model had the greatest accuracy predicting the test set. A more robust comparison with more data should be performed when possible, as the test set had only eighteen observations.

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site



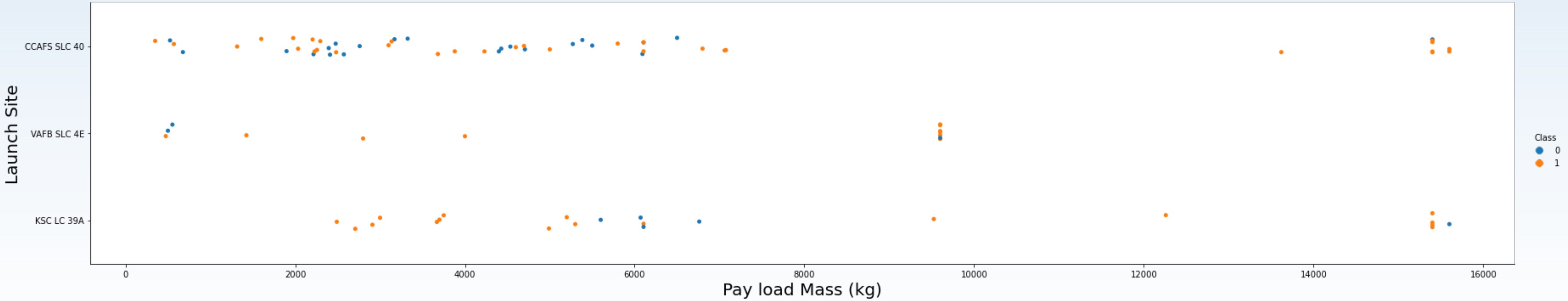
The above scatter plot shows how Flight Number, which is chronological, and Launch Site are related. The colors red and blue indicate successful and unsuccessful landings.

Two basic observations that can be drawn from this plot are:

- The launch site VAFB SLC 4E has a significantly smaller number of flights than the other two sites
- The percentage of successful landings increases with time.

When looking at the relationship between Launch Site and success, it is important to figure in chronology. Site CCAFS SLC 40 has a lower percentage of success since it was the location of nearly all of the earlier attempts at landing the first stage.

Payload vs. Launch Site



The above scatter plot shows the relationship between Payload Mass and Launch Site. The colors red and blue indicate successful and unsuccessful landings.

The site with the largest range of payload mass is CCAFS SLC 40, from 350kg - 15,600kg, and the site with the smallest range is VAFB SLC 4E, from 475kg – 9,600kg.

The percentage of success is higher for the heavier payloads, but this may be attributed to the fact that SpaceX used relatively smaller payloads in their initial trials and did not start launching the more heavy payloads until after the overall success rate had increased.

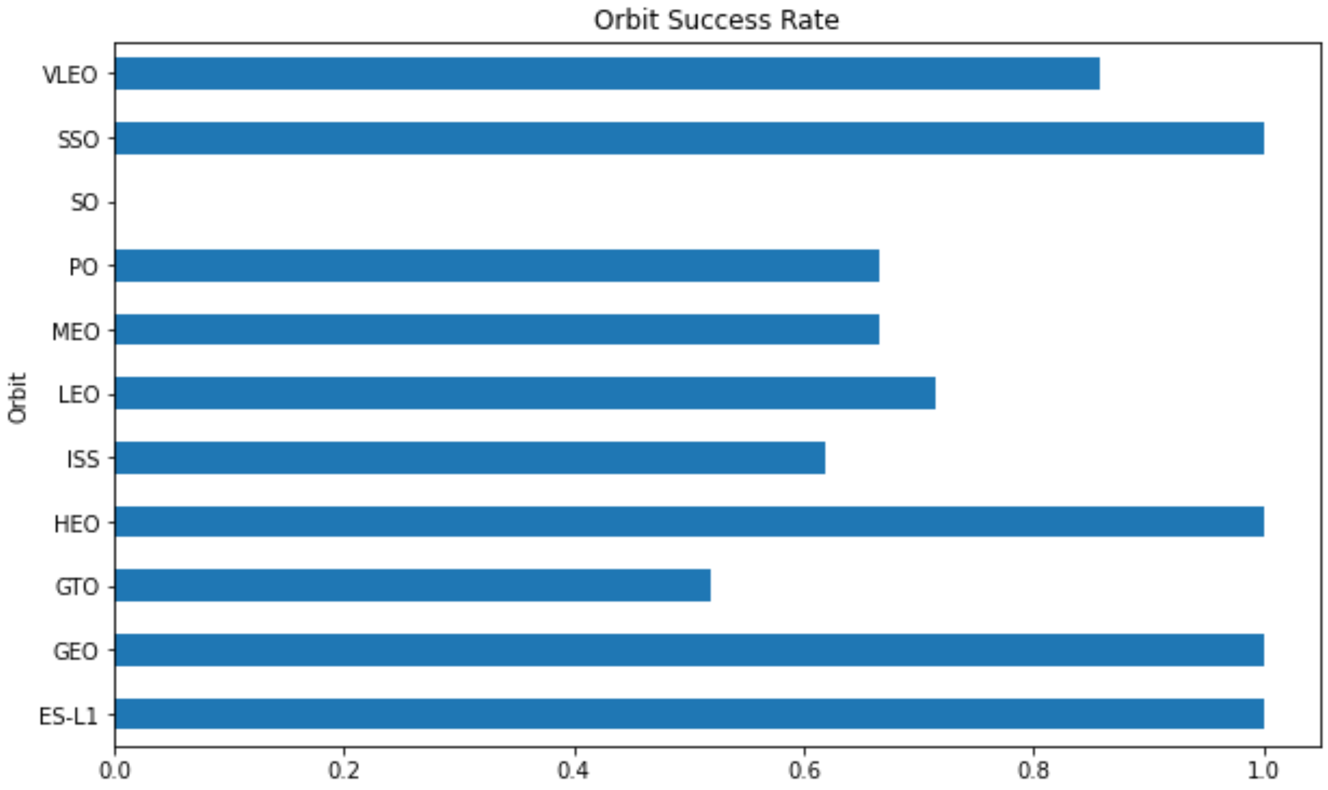
Success Rate vs. Orbit Type

This bar graph shows the average success rate for launches into different orbits.

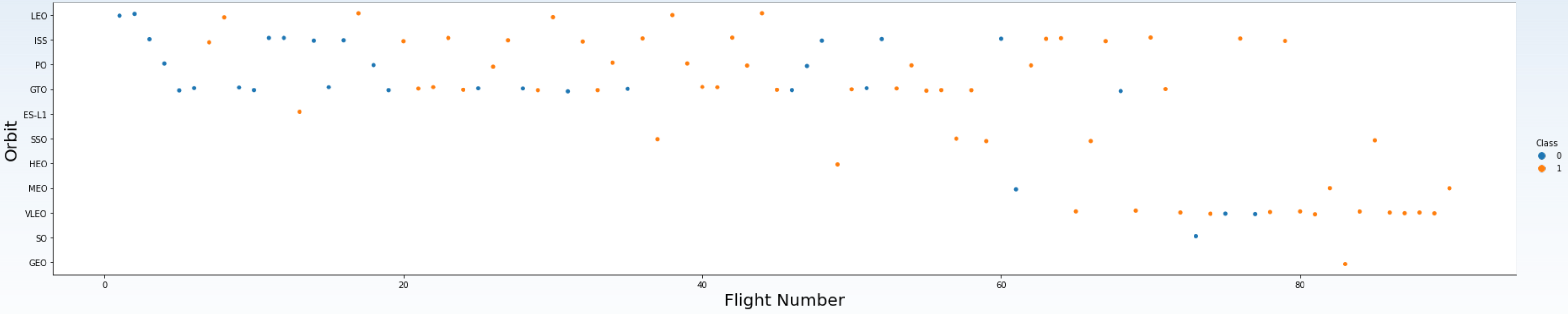
Flights into some of these orbits were not attempted until later, and that may be a factor in their higher success rates.

For instance, there is an 86% success rate for launches into VLEO orbits, which did not start until after the 60th flight. GEO orbits have a success rate of 100%, but there was only one flight and it was the 83rd out of 90. Conversely, the first attempted flights were to the PO, LEO, ISS, and GTO orbits, which have success rates of 67%, 71%, 62%, and 52%.

The relationship between Flight Number and Orbit Type can be seen in the following plot.



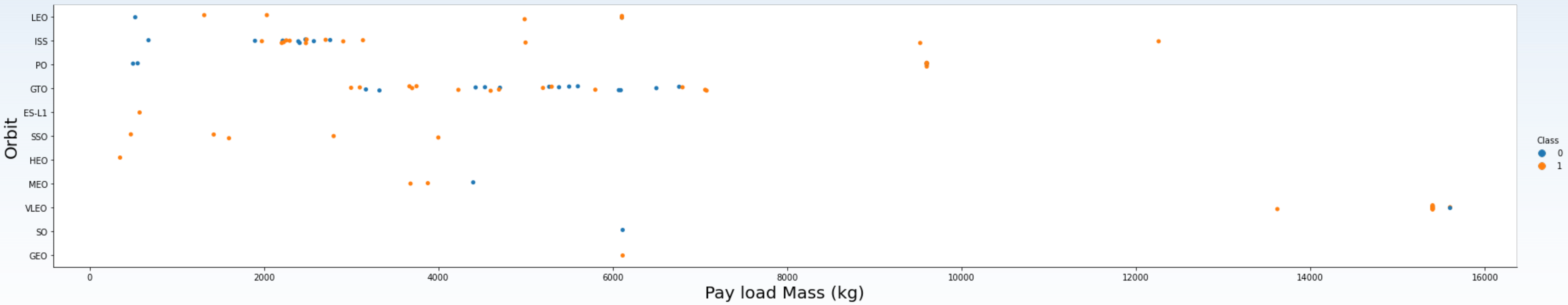
Flight Number vs. Orbit Type



The scatter plot above shows the relationship between Flight Number and Orbit Type. The colors red and blue indicate successful and unsuccessful landings.

It can be observed here that flights into some of the orbits were not attempted until later. This positively impacts their success rate as the overall success rate is positively correlated with time (see 'Launch Success Yearly Trend').

Payload vs. Orbit Type



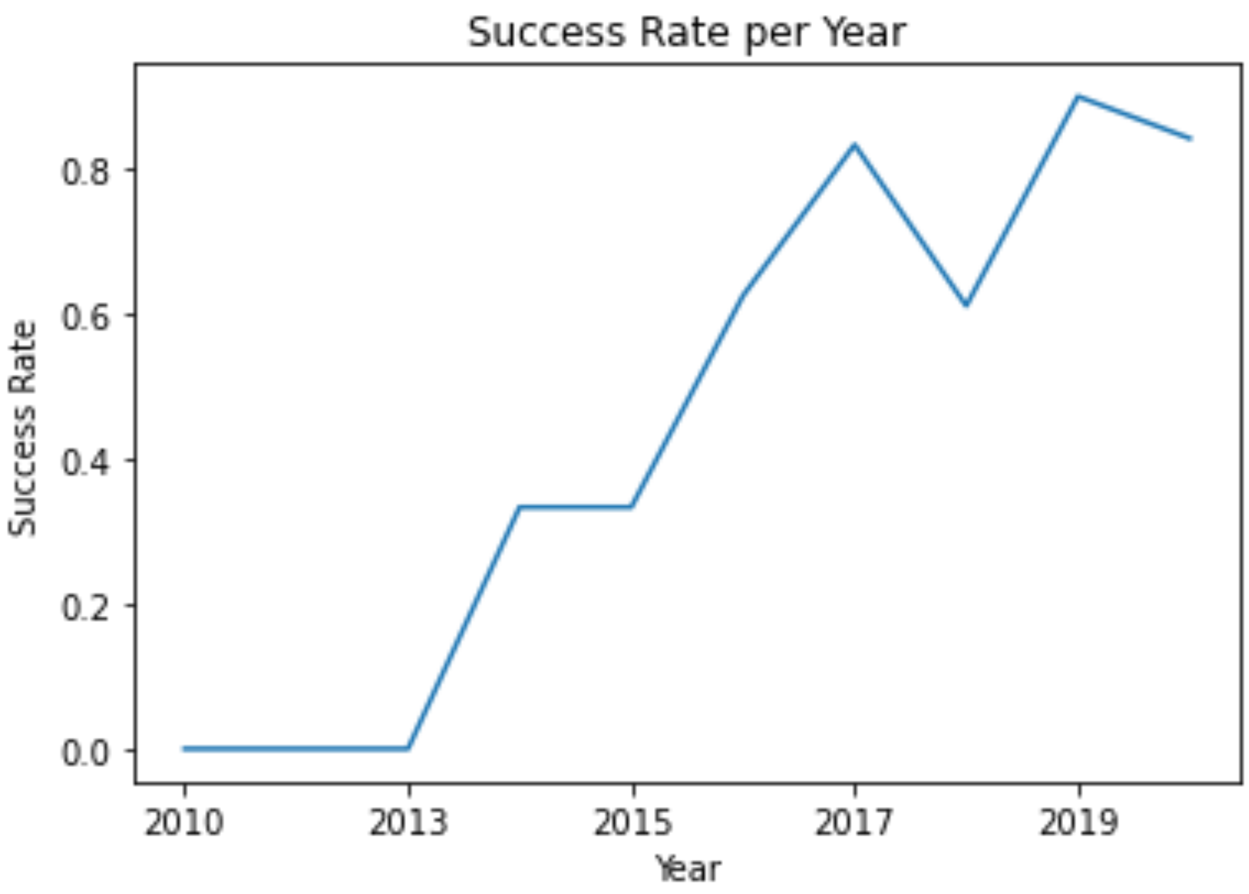
The scatter plot above shows the relationship between Payload Mass and Orbit Type. The colors red and blue indicate successful and unsuccessful landings.

It can be seen here that the heaviest payloads were launched into VLEO orbits.

Launch Success Yearly Trend

The adjacent line graph shows the launch success rate per year.

It can be seen here that the first successful landing was in 2013 and the success rate gradually increases over time.



All Launch Site Names

This query was run using the DISTINCT statement to display the unique launch site names from the dataset.

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

```
%sql SELECT DISTINCT(launch_site) FROM spacextbl;
```

```
 * ibm_db_sa://vzc47433:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb
Done.
```

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

Launch Site Names Begin with 'CCA'

This query was run using a WHERE clause with and a LIKE operator to display the first five records with launch sites that begin with 'CCA.'

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

```
%sql SELECT * FROM spacextbl WHERE launch_site LIKE 'CCA%' LIMIT 5;
```

```
 * ibm_db_sa://vzc47433:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/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

Total Payload Mass

This query was run using the SUM function with the GROUP BY statement and HAVING clause to display the total payload mass carried by boosters launched by NASA.

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

```
%sql SELECT SUM(payload_mass__kg_) AS nasa_crs FROM spacextbl
GROUP BY customer HAVING customer='NASA (CRS)'
```

```
 * ibm_db_sa://vzc47433:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb
Done.
```

nasa_crs
45596

Average Payload Mass by F9 v1.1

This query was run using a WHERE clause and the AVG function to display the average payload mass carried by booster version F9 v1.1.

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

```
%sql SELECT AVG(payload_mass__kg_) AS AVG_F9V11 FROM spacextbl
WHERE booster_version='F9 v1.1'
```

```
 * ibm_db_sa://vzc47433:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n
41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb
Done.
```

avg_f9v11
2928

First Successful Ground Landing Date

This query was run using a WHERE clause and the ORDER BY key word to display the first successful landing on a ground pad.

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

```
%sql SELECT date AS first_success FROM spacextbl
WHERE landing__outcome = 'Success (ground pad)'
ORDER BY date ASC LIMIT 1
```

```
 * ibm_db_sa://vzc47433:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb
Done.
```

first_success
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

This query was run using a WHERE clause, the BETWEEN operator, and the = operator to display the names of the boosters which had successful landings on drone ships while carrying a payload mass between 4000kg and 6000kg.

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

```
%sql SELECT booster_version FROM spacextbl
WHERE payload_mass__kg_ BETWEEN 4000 AND 6000
AND landing__outcome ='Success (drone ship)'
```

```
 * ibm_db_sa://vzc47433:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb
Done.
```

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

Total Number of Successful and Failure Mission Outcomes

This query was run using the COUNT function with the GROUP BY statement to calculate and display the number of successful and unsuccessful mission outcomes.

List the total number of successful and failure mission outcomes

```
%sql SELECT mission_outcome, COUNT(*) AS count
FROM spacextbl GROUP BY mission_outcome
```

```
 * ibm_db_sa://vzc47433:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb
Done.
```

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

Boosters Carried Maximum Payload

This query was run using the DISTINCT statement, a WHERE clause, and a subquery which uses the MAX function to display the names of the booster versions that have carried the maximum payload mass.

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

```
%sql SELECT DISTINCT(booster_version) FROM spacextbl
WHERE payload_mass__kg_ = (SELECT MAX(payload_mass__kg_)
                          FROM spacextbl)
```

```
 * ibm_db_sa://vzc47433:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb
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

This query was run using a WHERE clause to display the landing outcome, date, booster version, and launch site from failed drone ship landings in 2015.

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

```
%sql SELECT landing__outcome, date, booster_version, launch_site
FROM spacextbl WHERE landing__outcome ='Failure (drone ship)'
AND YEAR(date)='2015'
```

```
 * ibm_db_sa://vzc47433:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c
3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb
Done.
```

landing__outcome	DATE	booster_version	launch_site
Failure (drone ship)	2015-01-10	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	2015-04-14	F9 v1.1 B1015	CCAFS LC-40

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

This query was run using the COUNT function and a subquery using a WHERE clause with the BETWEEN operator, the GROUP BY statement, and the ORDER BY key word.

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

```
%sql SELECT landing__outcome, COUNT(*) AS count FROM (
    SELECT landing__outcome FROM spacextbl
    WHERE date BETWEEN '2010-06-04' AND '2017-03-20')
    GROUP BY landing__outcome ORDER BY count DESC
```

```
 * ibm_db_sa://vzc47433:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb
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

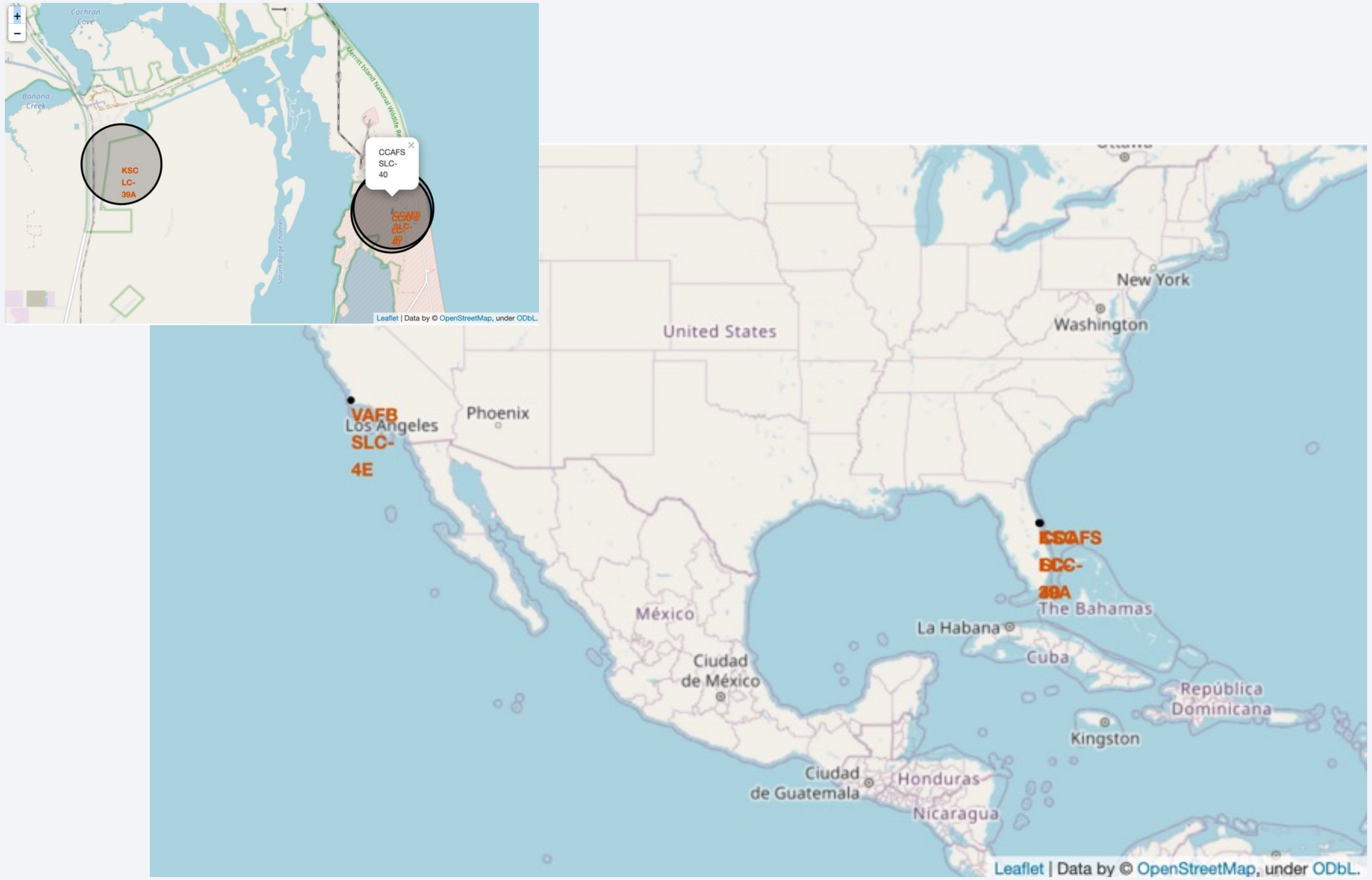
Section 3

Launch Sites Proximities Analysis

Folium Map of Launch Sites

A folium map was created with the four SpaceX launch sites plotted - one in California and three in Florida.

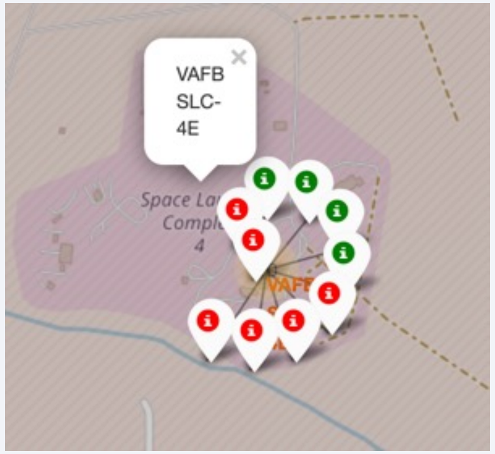
You can see that the three launch sites in Florida are close to each other; the different sites can be viewed by expanding the map.

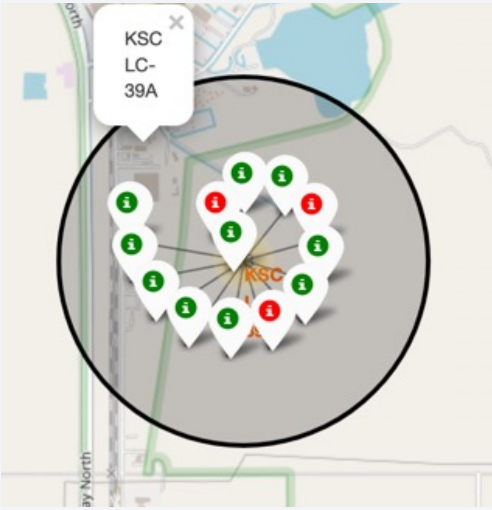


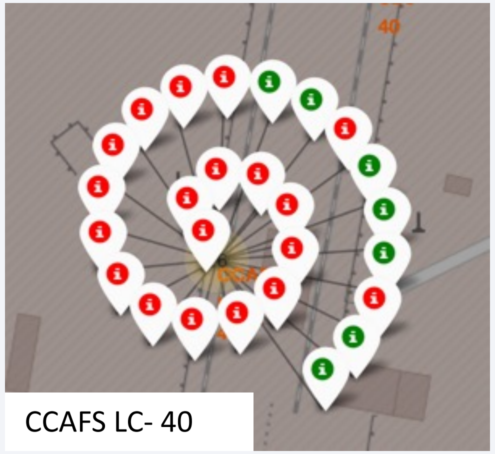
Folium Map with Launch Outcomes

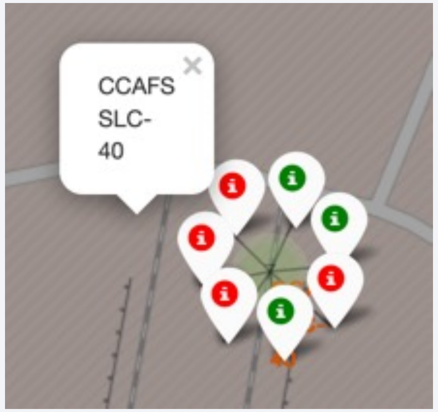
Marker clusters were used to show landing outcomes, green for successful and red for unsuccessful, at each launch site.

The dataset that these outcomes were pulled from contains only 56 observations, from the first flight in 2010 to the 56th flight in June 2018. Since launch outcomes improve over time, there is a large percentage of unsuccessful flights at all of the sites except for KSC LC-40, whose 7 flights start at flight number 45.









Folium Map of Distances to Proximities

This map shows the distance from launch site CCAFS SLC- 40 to three different land marks:

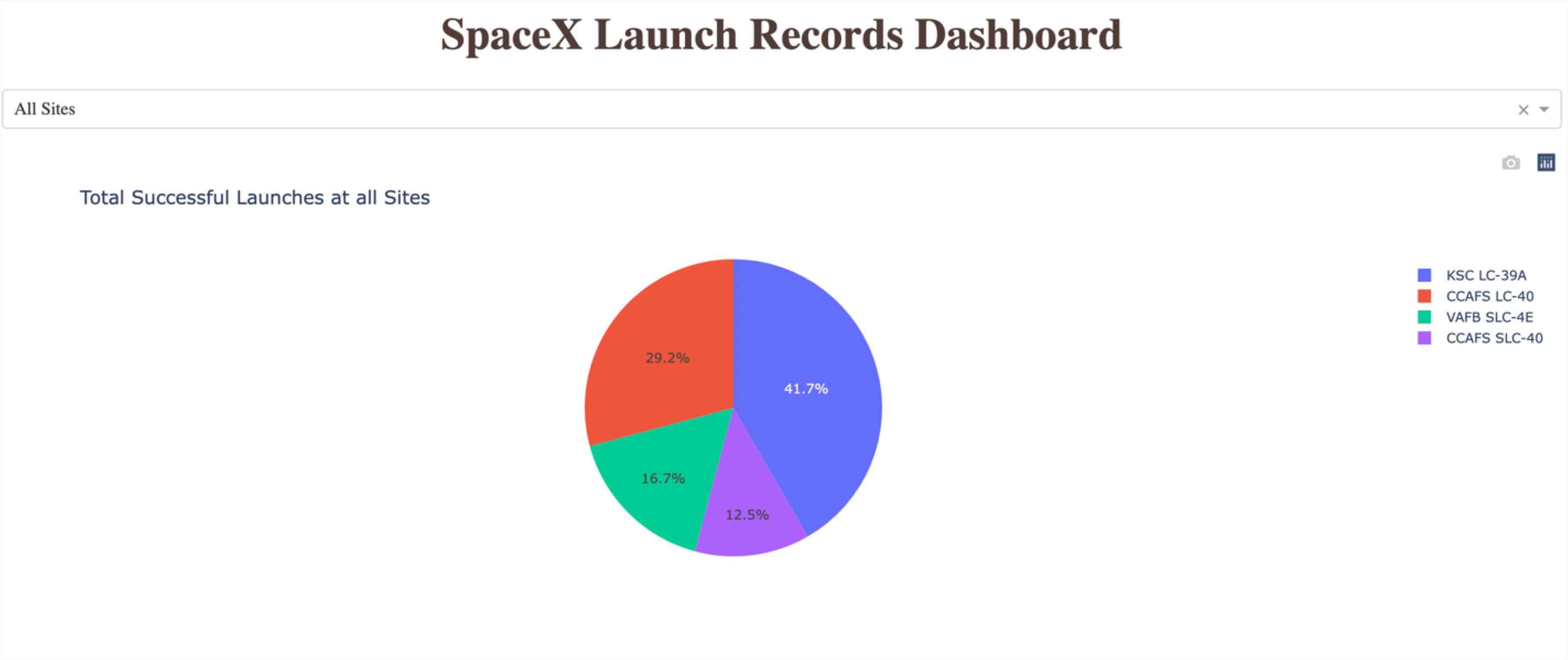
- NASA Railroad (1.29km)
- Coastline (.87km)
- Highway (.59km)

This site, like the others, is in close proximity to a railroad, a highway, and a coastline. The railroad and highway offer viable routs for transportation of materials and manufactured rockets (see appendix), while the ocean offers a safe place to land the first stage.

Section 4

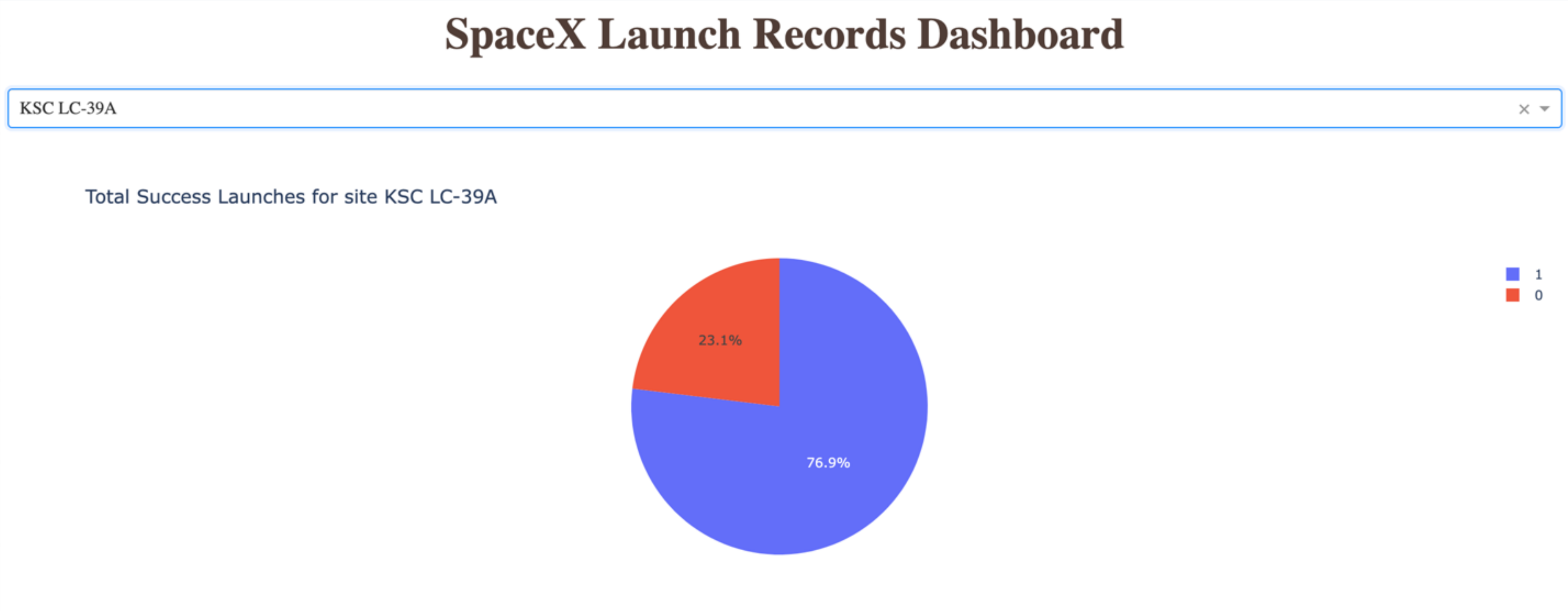
Build a Dashboard with Plotly Dash

Dashboard: Launch Success at all Sites



This pie chart shows the percentage of total successful flights for each launch site. Site KSC LC-39A has the greatest percentage of successful launches at 41.7%.

Dashboard: Highest Success Rate



This pie chart shows the ratio of successful to unsuccessful launch outcomes for the launch site with the greatest percentage of successful launches, KSC LC-39A. This site has a positive launch rate of 76.9%.

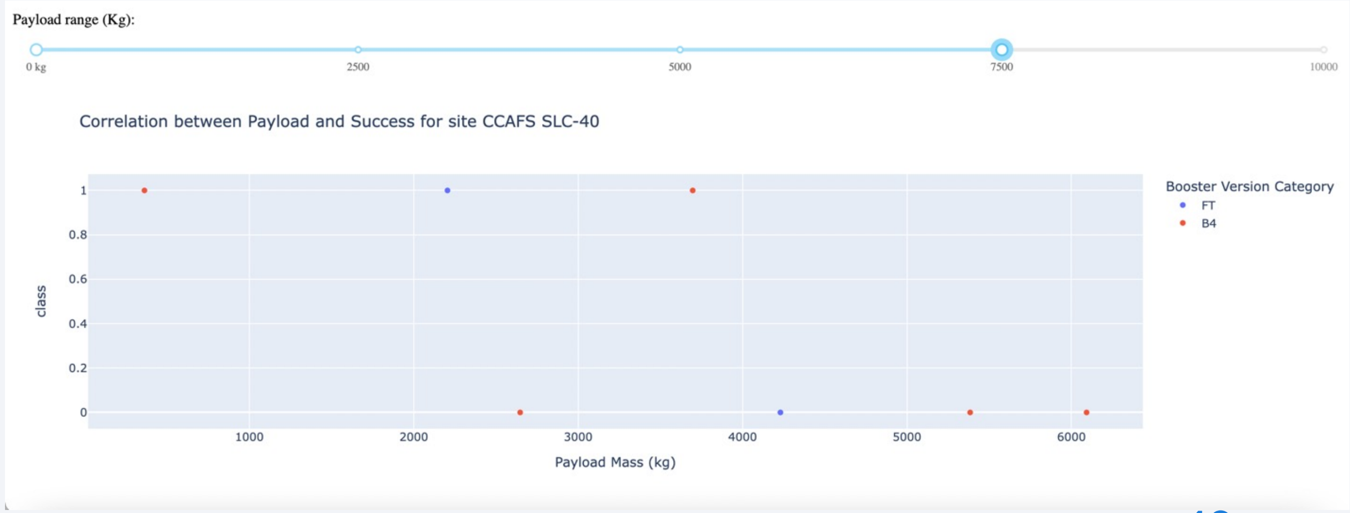
Dashboard: Payload vs. Launch Outcome

This scatterplot show the relationship between Payload and Success (Class) for all sites. The key on the right shows the colors associated with different booster versions, where v1.0 is the oldest and B5 is the most recent. You can see that the more recent booster versions have a greater rate of success and that heavier payloads are mostly associated with the more recent booster versions.

Dashboard: Payload vs. Launch Outcome

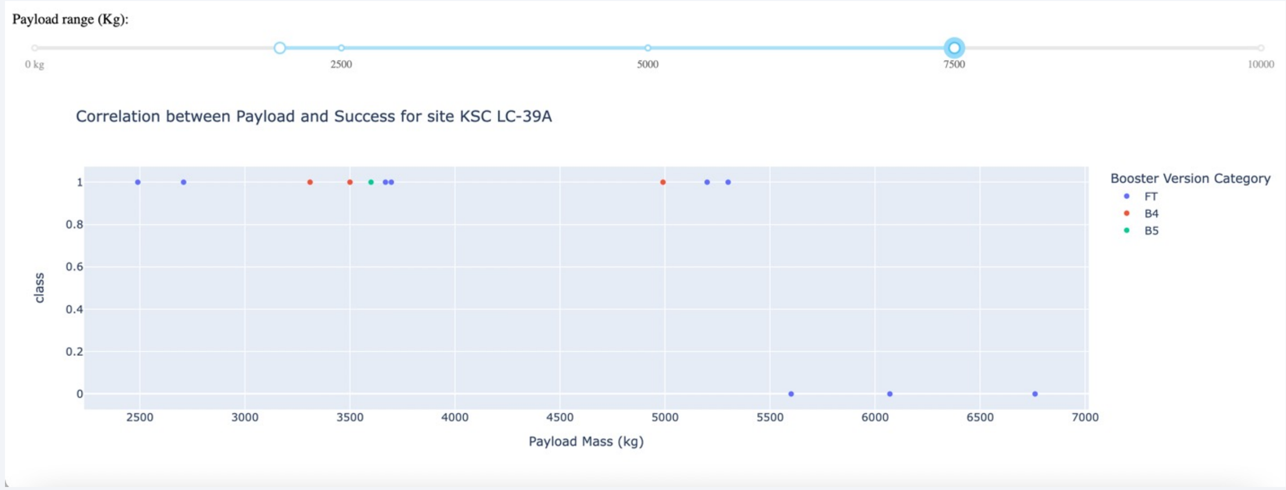
This plot is of flights from the CCAFS LC-40 launch site. The slider is restricting the payload mass to the range 0kg – 5500kg, which does include all observations. It can be observed that there are significantly less successful launches than unsuccessful and that all of the launches were with the three first booster versions.

This plot is of flights from the CCAFS SLC-40 launch site. The slider is restricting the payload mass to the range 0kg – 6500kg, which does include all observations. It can be observed there are only seven launches from this site and that the booster versions are the third and second most recent. The ratio of successful to unsuccessful flights is 3 to 4. The heavier payloads here are associated with unsuccessful landings.



Dashboard: Payload vs. Launch Outcome

This plot is of flights from the KSC LC-39A launch site. The slider is restricting the payload mass to the range 2000kg – 7000kg, which does include all observations. It can be observed that there are significantly more successful launches than unsuccessful and that all of the launches were with the three most recent booster versions. Here, again, the unsuccessful launches are with heavier payloads.



This plot is of flights from the VAFB SLC-4E launch site. The slider is allowing for the full range, 0kg-10,000kg, in order to include all observations. It can be observed that there are only seven launches from this site and all are unsuccessful except for two. One success was with the second booster version and a very light payload and the other was with the second most recent booster and the heaviest payload.

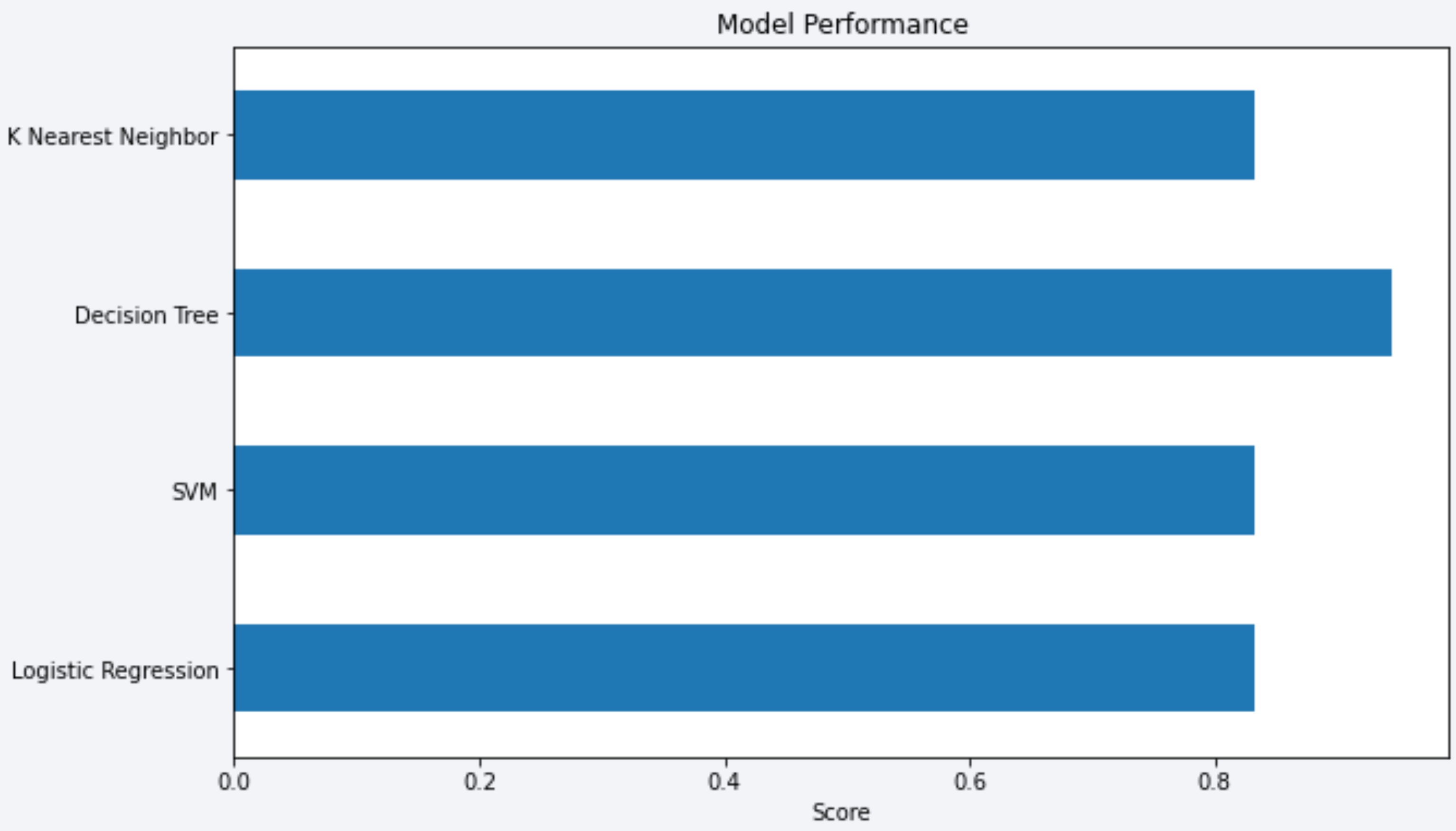
Section 5

Predictive Analysis (Classification)

Classification Accuracy

The Decision Tree model had the highest accuracy score out of the four.

The other three had the same accuracy scores. The test set had only 18 observations; a more robust comparison could be achieved with more data.



Confusion Matrix

This is the confusion matrix for the Decision Tree model predicting the test set.

The test set contained only 18 observations. The Decision Tree model correctly predicted 5 unsuccessful landings and12 successful landings while incorrectly predicting 1 successful landing.

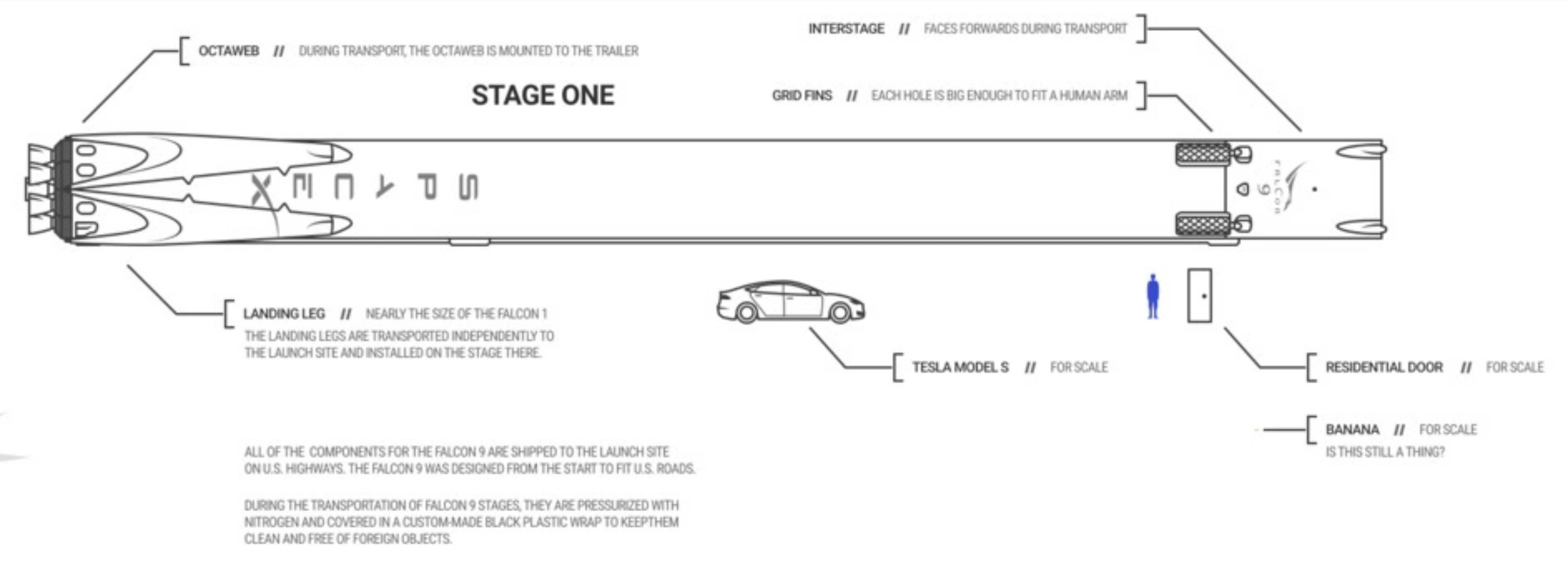
Decision Tree

Conclusions

- Exploratory data analysis shows a solid relationship between flight number and landing success, particularly in the first half of the data.
- With the data and features chosen, the Decision Tree Model using parameters **criterion =** ‘gini’, **splitter** = ‘best’, **maximum features =** ‘sqrt’, **minimum leaf samples** = 4, **minimum split samples =** 5, and **maximum depth =** 8 gave the best predictions on the test set with an **accuracy** score of 0.9444.
- Prediction of landing success is be possible, but will be more accurate as more data is accumulated. The classification models should be re-evaluated once more data is obtained.

Appendix

- This schematic diagram of the first stage shows how it was designed to be transportable on highways.



Thank you!