IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Jonathan Scott
11/24/2024

Outline

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

Executive Summary

- **Summary of methodologies**
- Data collection
- Web Scraping
- Data cleaning
- Data wrangling
- EDA with SQL
- EDA Data Visualization
- Launch Sites Location Analysis
- Machine Learning Predictions
- Dash Bord app

- **Summary of all results**
- Lauch Success Rates
- Optimal Launch sites
- Predictive models

Introduction

- **Project background and context**
- SpaceX's accomplishments include: Sending spacecraft to the International Space Station. Starlink, (a satellite internet constellation providing satellite Internet access) Sending manned missions to Space, etc.
- One reason SpaceX can do this is the rocket launches are relatively inexpensive. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; compared with other provider at 165 million dollars each, much of the savings is because SpaceX can reuse the first stage.
- **Problems you want to find answers**
- Determine if the first stage will land.
- Determine the final cost of a launch.
- Determine if the success of a launch depend on the location and proximities of a launch site.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - SpaceX API to extract information
  - Web scraping from an online webpage
- Perform data wrangling
  - Performed some exploratory data analysis (EDA) to find some patterns in the data and determine what would be the label for training supervised models.
- Data Cleaning:
  - Handel missing values.
  - Remove duplicates.
  - Correct errors.
- Data Transformation:
  - Data normalization.
  - One-Hot Encoding.
  - Feature Engineering.

Data Collection

- Describe how data sets were collected.
- The data sets for the analysis were obtained by requesting data from the SpaceX API
- And also by performing web scraping on the Wikipedia webpage referent to the List of Falcon 9 Launches

Data Collection – SpaceX API

- **Summary:** Make a get request to the SpaceX API
- [GitHub URL]

Flowchart:

Request the rocket launch data from SpaceX API

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

Convert the Json result into a data frame

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

Data Parsing

```
data                = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]
data                = data[data['cores'].map(len)==1]
data                = data[data['payloads'].map(len)==1]
data['cores']       = data['cores'].map(lambda x : x[0])
data['payloads']    = data['payloads'].map(lambda x : x[0])
data['date']        = pd.to_datetime(data['date_utc']).dt.date
data                = data[data['date'] <= datetime.date(2020, 11, 13)]
```

Create new columns to store the data

```
BoosterVersion = [], PayloadMass  = [], Orbit       = []
LaunchSite     = [], Outcome      = [], Flights     = []
GridFins       = [], Reused       = [], Legs        = []
LandingPad     = [], Block        = [], ReusedCount = []
Serial         = [] ,Longitude    = [], Latitude    = []
```

Construct a dataset using the obtained data.

```
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}
```

Create new Data Frame and exported to a CSV file

```
df = pd.DataFrame(launch_dict)
data_falcon9 = df.drop(df[df['BoosterVersion'] == 'Falcon 1'].index)
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data Collection - Scraping

- Web scraping from the Wikipedia webpage ["List of Falcon 9 and Falcon Heavy launches"]
- [GitHub URL]

Flowchart:

Request the HTML page from the URL, and get a Beautiful-soup object.

```python
static_url = "https://en.wikipedia.org/List_of_Falcon_9_and_Falcon_Heavy_launches"
response = requests.get(static_url).text
soup = BeautifulSoup(response, 'html.parser')
```

Extract all column/variable names from the HTML table header.

```python
html_tables = soup.find_all('table')
all_headers = []
for table in html_tables:
    headers = [header.text.strip() for header in table.find_all('th')]
    all_headers.append(headers)
```

Parsing the data to find all the elements in the table.

```python
header_elements = first_launch_table.find_all('th')
column_names = []
for header in header_elements:
    name = extract_column_from_header(header)
    if name is not None and len(name) >0:
        column_names.append(name)
```

Create a data frame by parsing the launch HTML tables.

```python
# Initialize Dictionary
launch_dict= dict.fromkeys(column_names)

# Remove an irrelevant 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'           ] = []

# Initialize variables
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 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 dictionary
        if flag:
            extracted_row += 1

            datatimelist = date_time(row[0])

            # Append the flight Number into launch_dict
            launch_dict['Flight No.'].append(flight_number)

            # Append the date into launch_dict
            date = datatimelist[0].strip(',')
            launch_dict['Date'].append(date)

            # Append the time into launch_dict
            time = datatimelist[1]
            launch_dict['Time'].append(time)

            # Append the bv into launch_dict
            bv = booster_version(row[1])
            if not(bv):
                bv = row[1].a.string
            launch_dict['Version Booster'].append(bv)

            # Extract and append the launch site into launch_dict
            launch_site = row[2].a.string
            launch_dict['Launch site'].append(launch_site)

            # Append the payload into launch_dict
            payload = row[3].a.string
            launch_dict['Payload'].append(payload)

            # Append the payload_mass into launch_dict
            payload_mass = get_mass(row[4])
            launch_dict['Payload mass'].append(payload_mass)

            # Append the orbit into launch_dict
            orbit = row[5].a.string
            launch_dict['Orbit'].append(orbit)

            # Append the customer into launch_dict
            if len(row) > 6 and row[6].a is not None:
                customer = row[6].a.string
            else:
                customer = None
            launch_dict['Customer'].append(customer)

            # Append the launch_outcome into launch_dict
            if len(row) > 7 and row[7].a is not None:
                launch_outcome = row[7].a.string
            else:
                launch_outcome = None
            launch_dict['Launch outcome'].append(launch_outcome)

            # Append the booster_landing into launch_dict
            if len(row) > 8 and row[8].a is not None:
                booster_landing = row[8].a.string
            else:
                booster_landing = None
            launch_dict['Booster landing'].append(booster_landing)
```

Create a Pandas Data Frame from the 'launch_dict' and export it to a CSV file.

```python
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
df.to_csv('spacex_web_scraped.csv', index=False)
```

Data Wrangling

- Exploratory Data Analysis to determine Training Labels.
- GitHub URL

Flowchart:

Load SpaceX Data Set and calculating the percentage of missing values of each attribute

```
df=pd.read_csv("https://space_x/datasets/dataset_part_1.csv")
df.isnull().sum()/len(df)*100
```

Determine the number of launches on each site

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

Determine the number and occurrence of each orbit in the column 'Orbit'

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

Determine the number of landing outcomes

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

Determine the number of successes, failures and pre-landings

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

Create a set of outcomes where the second stage did not land successfully

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

Apply One Hot Encoding to 'landing_class' and add the new column to the data fame

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

Determine the success rate of the launches

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

Calculate the mean of the new 'Class' column

```
df["Class"].mean()
```

Export the data frame to a new CSV file

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

EDA with Data Visualization

- **SpaceX Falcon 9 First Stage Landing Prediction**
- [GitHub URL]

Summary of charts:

	Chart Title:	Chart Type:	Description:
1.	Flight Number vs. Payload Mass.	Scatter Plot.	To understand how the different variables will affect the launch outcome.
2.	Flight Number vs. Launch Site.	Scatter Plot.	To understand the relationship between landing success and launch site.
3.	Payload Mass vs. Launch Site.	Scatter Plot.	To understand the relationship between Payload Mass and Launch site.
4.	Success Rate vs. Orbit Type.	Bar Plot.	To understand the relationship between Success Rate and Orbit Type.
5.	Orbit Type vs. Outcome.	Bar Plot.	To understand the relationship between Orbit Type and Landing Success.
6.	Flight Number vs. Orbit Type.	Scatter Plot.	To understand the relationship between Flight Number and Orbit Type.
7.	Payload Mass vs. Orbit Type.	Scatter Plot.	To understand the relationship between Payload Mass and Orbit Type.
8.	Year vs. Success Rate.	Line Plot.	To get obtain the average launch success trend.

EDA with SQL

- **Exploratory Data Analysis with SQL**
- GitHub URL

Summary:

- Connect with the database to create a SQL table, and Fetch the data for unique values.
- Identify the different launch sites and filter the data frame.
- Execute magic commands to remove blanks and create table SPACEXTABLE.
- Display the names of the unique launch sites in the space mission.
- Fetch the date when the first successful landing in ground pad was achieved.
- Average Payload Mass by buster version F9 v1.1.
- Fetching data to get total payload mass and unique values for NASA-launched Boosters.
- Querying the database to obtain 5 records where launch sites start with 'CCA'.
- Fetch boosters that succeed in drone ship with payload between 4000 and 6000.
- Quering the database to obtain the landing outcomes in different landing pads.

Build an Interactive Map with Folium

- Launch Site Location Analysis.
- GitHub URL

Summary:

- Create a folium Circle and folium Marker for each launch site on the site map to identify launch locations.
- Create a Marker clusters object of success/failed launches for each side on the map for visualization .
- Add Mouse Position to get the coordinate (Lat, Long) for a mouse over on the map
- Calculate the distance from the launch site to the surroundings to understand the relationship between the launch side and the proximities.

Build a Dashboard with Plotly Dash

- Building a Plotly Dash application for users to perform interactive visual analytics on SpaceX launch data in real-time.
- GitHub URL

Summary:

1. Dropdown list to enable launch site selection for interaction with Pie Chart and Scatter Plot to select a specific launch side for analysis.

2. Interactive Pie Chart to showcase the successful launches count for the selected site to compare the success rate of each site.

3. A Slider to select payload range that interact with the Scatter Plot to narrow down the payload range for analysis.

4. Scatter plot that interacts with the payload range Slider and the drop down list to compare the buster version for each side for analysis.

Summarize how you built, evaluated, improved, and found the best performing classification model

Predictive Analysis (Classification)

- **Space X Falcon 9 First Stage Landing Prediction**
- [GitHub URL]

Flowchart:

Load the data in two data frames

```
data = pd.read_csv("https:///IBM-SkillsNetwork/datasets_part_2.csv")
X = pd.read_csv('https://IBMSkillsNetwork/dataset_part_3.csv')
```

Create a NumPy array from 'Class' on 'data' and standardize data on 'X'

```
Y = data['Class'].to_numpy()
transform = preprocessing.StandardScaler()
X = transform.fit_transform(X)
```

Split the data into training and testing

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state = 2)
print ('Train set:', X_train.shape,  Y_train.shape)
print ('Test set:', X_test.shape,  Y_test.shape)
```

Create a logistic regression and a GridSearchCV objects

```
parameters ={"C":[0.01,0.1,1], 'penalty':['l2'], 'solver':['lbfgs']}
lr=LogisticRegression()
logreg_cv = GridSearchCV(lr, parameters, cv = 10)
logreg_cv.fit(X_train, Y_train)
```

Create a Support vector machine, and a GridSearchCV objects

```
parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
              'C': np.logspace(-3, 3, 5),
              'gamma':np.logspace(-3, 3, 5)}
svm = SVC()
svm_cv = GridSearchCV(svm, parameters, cv = 10)
svm_cv.fit(X_train, Y_train)
```

Create a Decision Tree and a GridSearchCV objects

```
parameters = {'criterion': ['gini', 'entropy'],'splitter': ['best', 'random'],
     'max_depth': [2*n for n in range(1,10)],'max_features': [None, 'sqrt'],
     'min_samples_leaf': [1, 2, 4],'min_samples_split': [2, 5, 10]}
tree = DecisionTreeClassifier()
tree_cv = GridSearchCV(tree, parameters, cv =10, error_score='raise')
tree_cv.fit(X_train, Y_train)
```

Create a K nearest neighbors and a GridSearchCV objects

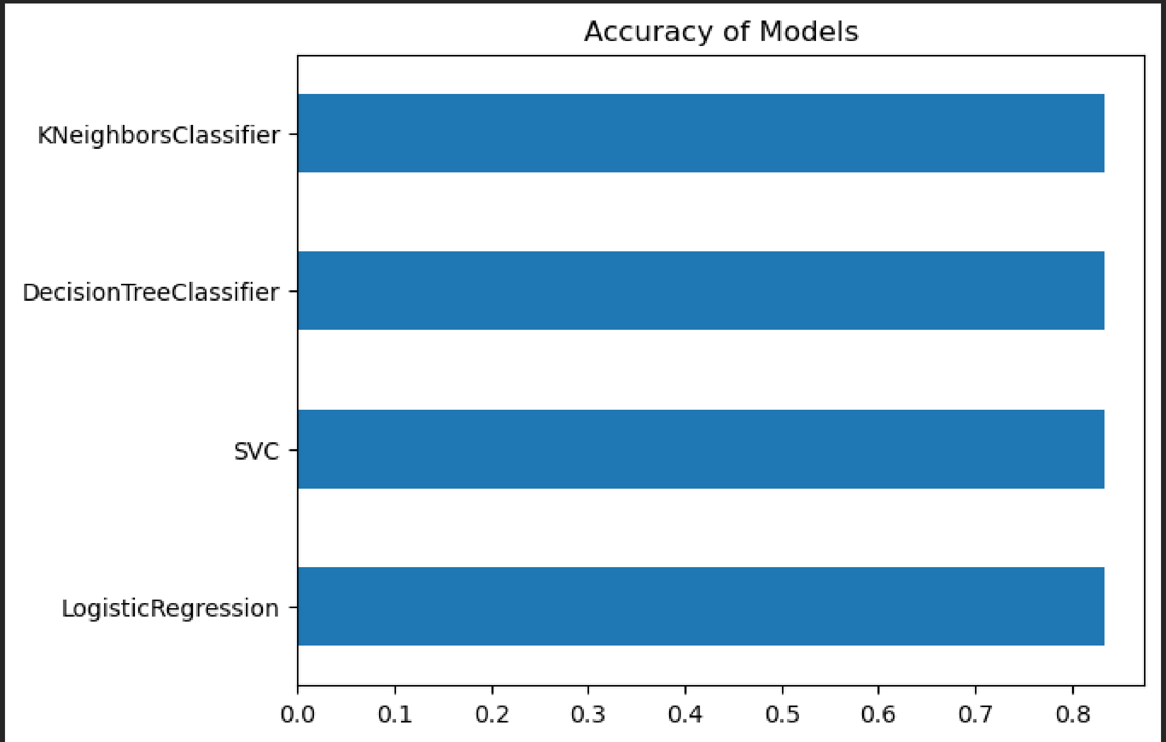
```
parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
              'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
              'p': [1,2]}
KNN = KNeighborsClassifier()
knn_cv = GridSearchCV(KNN, parameters, cv = 10)
knn_cv.fit(X_train, Y_train)
```

Find the method that performs best

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

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

    'DecisionTreeClassifier': {'criterion'        : ['gini', 'entropy'],
        'splitter'         : ['best', 'random'],
        'max_depth'        : [2*n for n in range(1, 10)],
        'max_features'     : [None, 'sqrt'],
        'min_samples_leaf' : [1, 2, 4],
        'min_samples_split': [2, 5, 10]},

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

models = {'LogisticRegression'    : LogisticRegression(),
          'SVC'                   : SVC(),
          'DecisionTreeClassifier': DecisionTreeClassifier(),
          'KNeighborsClassifier'  : KNeighborsClassifier()}

best_estimators = {}
for model_name in models:
    grid_search = GridSearchCV(estimator = models[model_name],
                               param_grid=param_grids[model_name], cv=10)
    grid_search.fit(X_train, Y_train)
    best_estimators[model_name] = grid_search.best_estimator_
for model_name, estimator in best_estimators.items():
    print(f"Best {model_name}: {estimator}")
```

Results

- We found a correlation ship between the geographic location, the orbit type, and the payload.
- The machine learning analysis indicated that all the prediction models train for this project, have an accuracy of 0.8333.

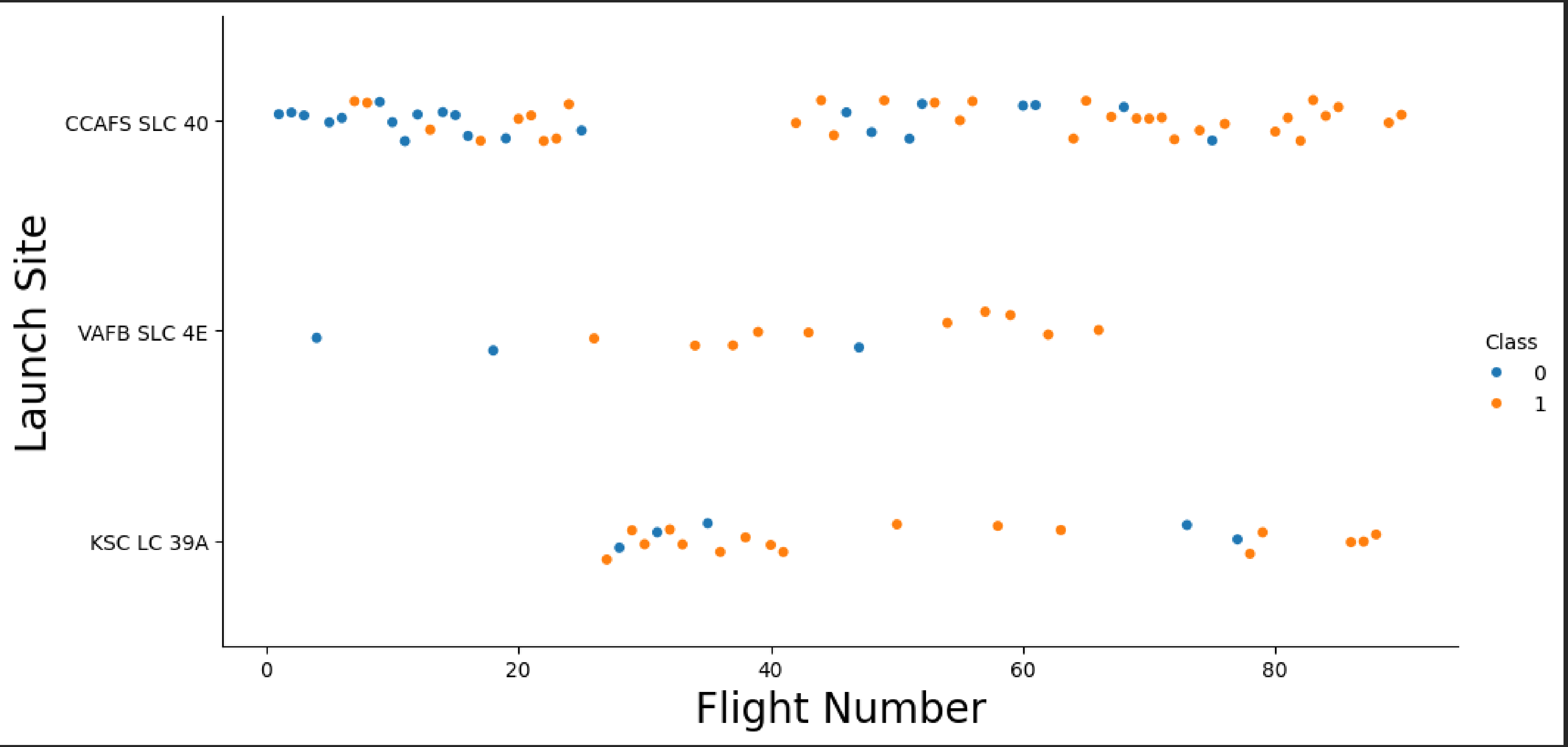


Section 2

Insights drawn from EDA

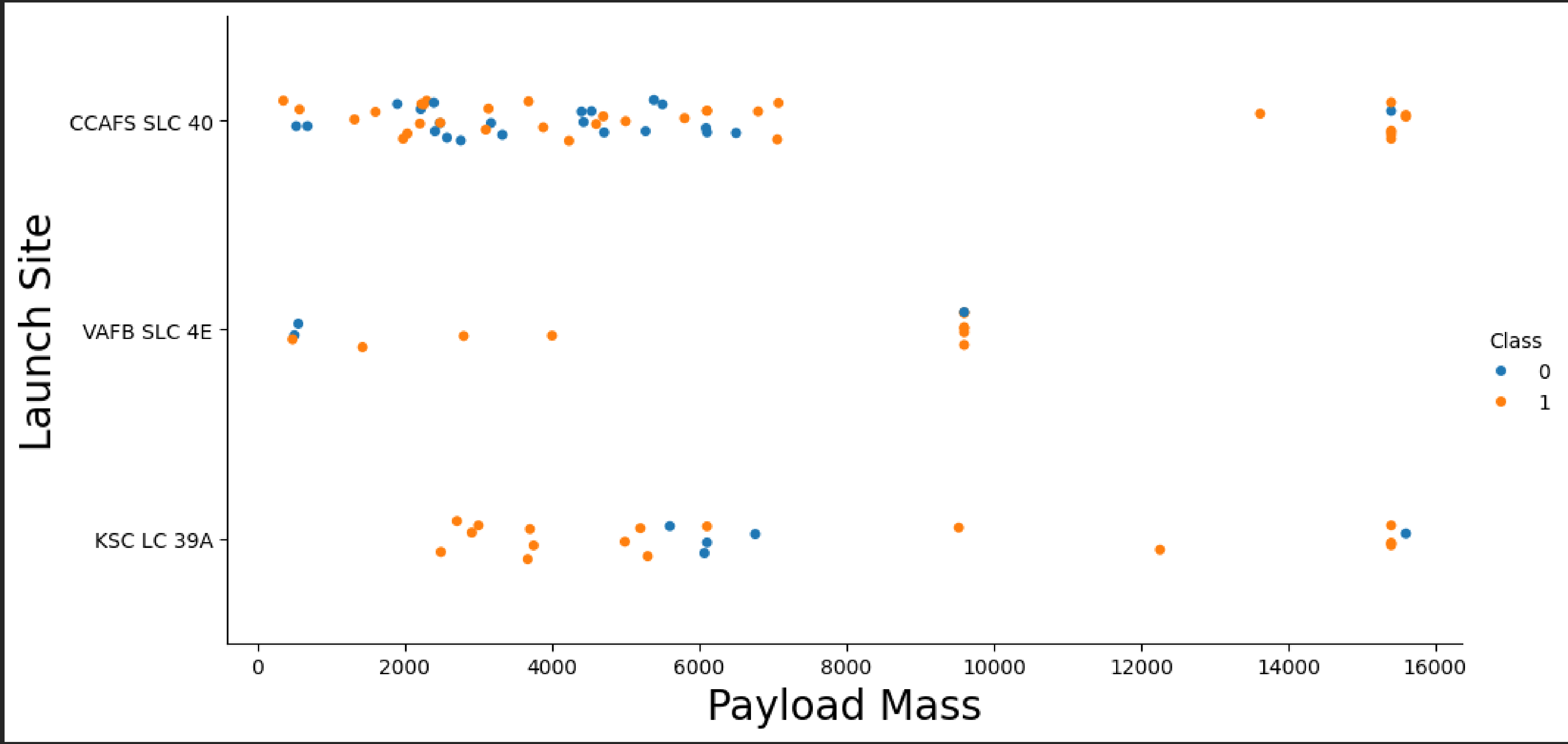
Flight Number vs. Launch Site

'KSC LC 39A' and 'VAFB SLC 4E' have higher success rates for landings compared to 'CCAFS SLC 40', which has only a 60% success rate.



Payload vs. Launch Site

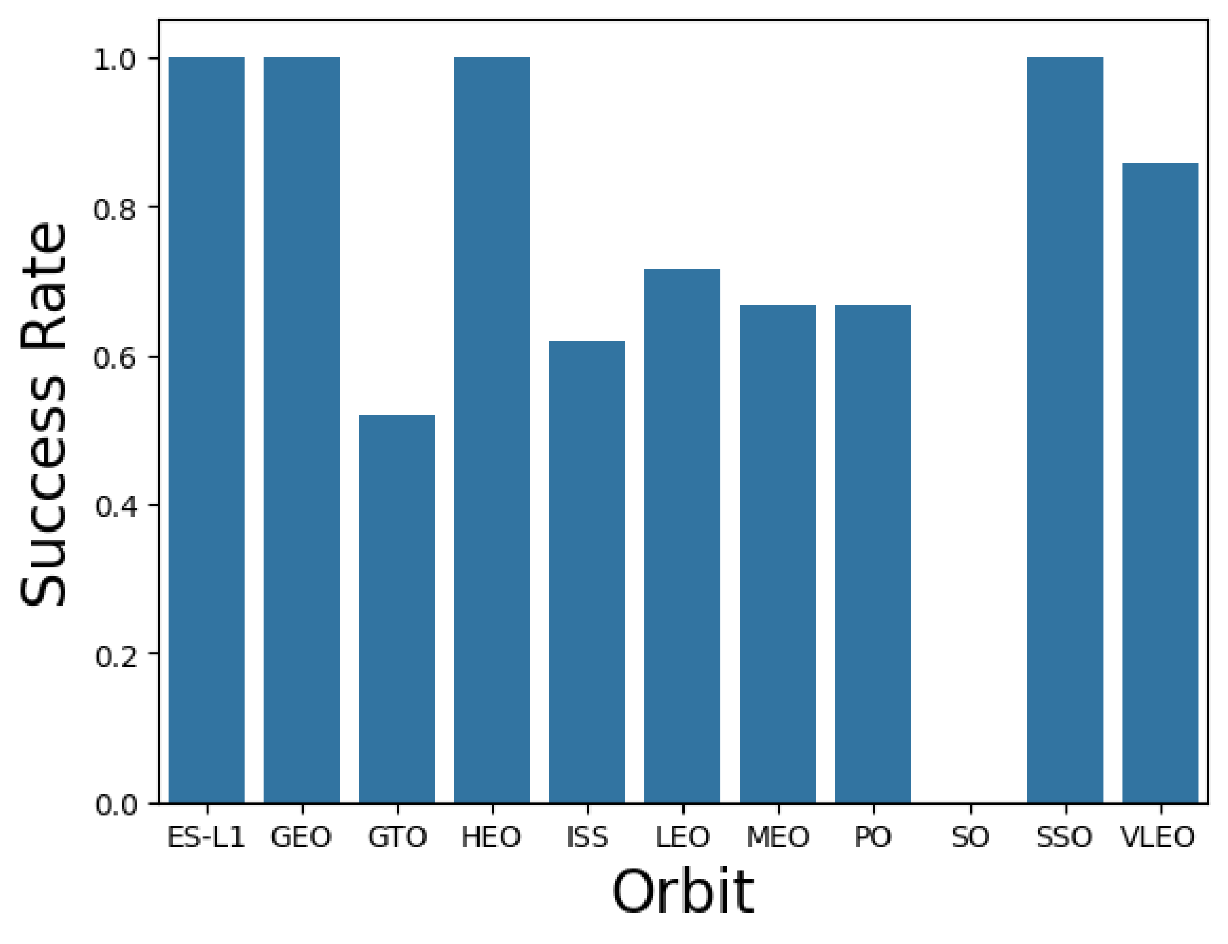
- Most failed landings occur with a payload mass between 500 and 7000.



Success Rate vs. Orbit Type

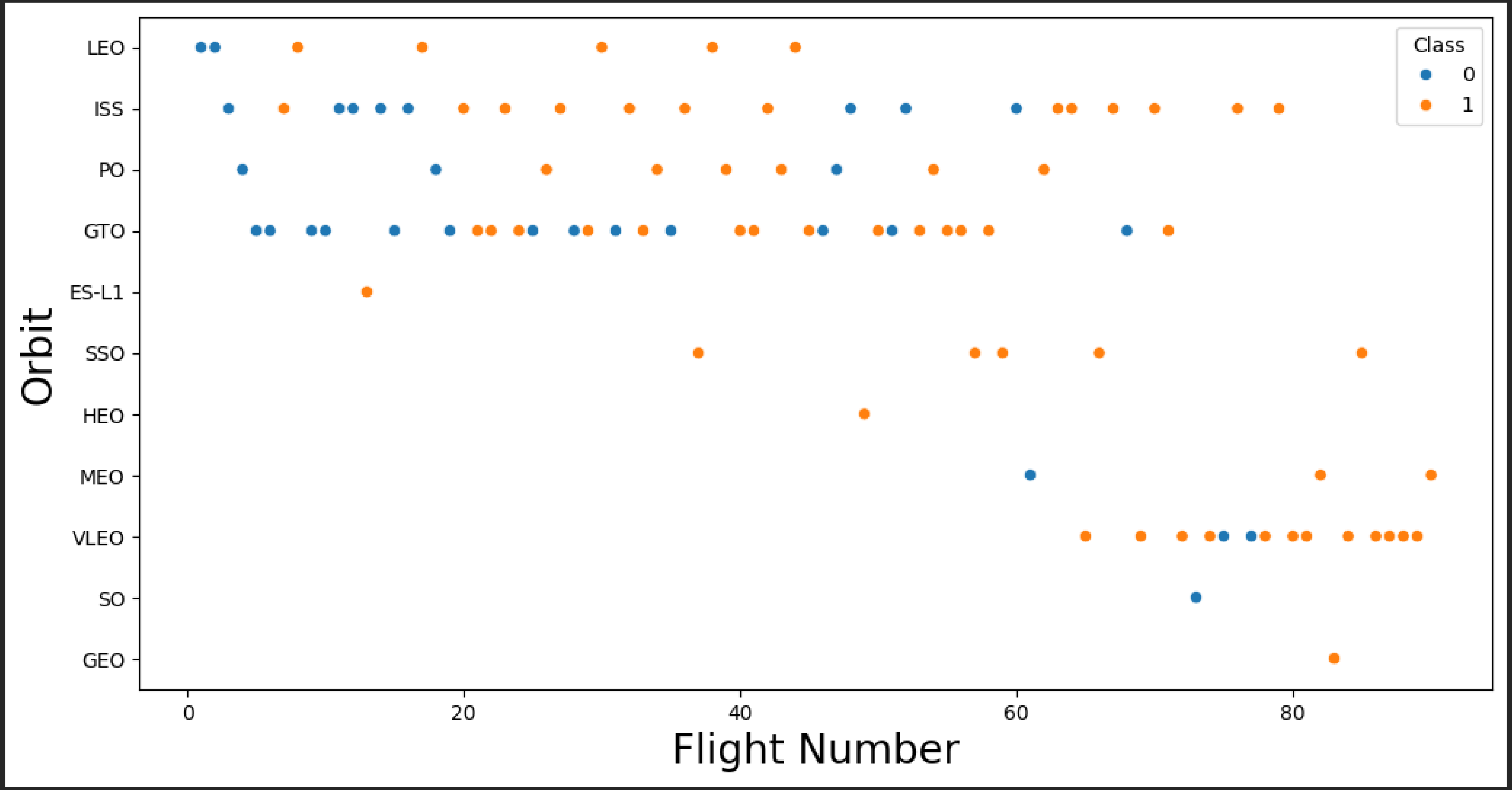
We can see a clear relationship between the Success Rate and the Orbit Type.

Orbit	Success Percentage
ES-L1	100%
GEO	100%
GTO	51.85%
HEO	100%
ISS	61.19%
LEO	71.42%
MEO	66.66%
PO	66.66%
SO	N/A
SSO	100%
VLEO	85.57%



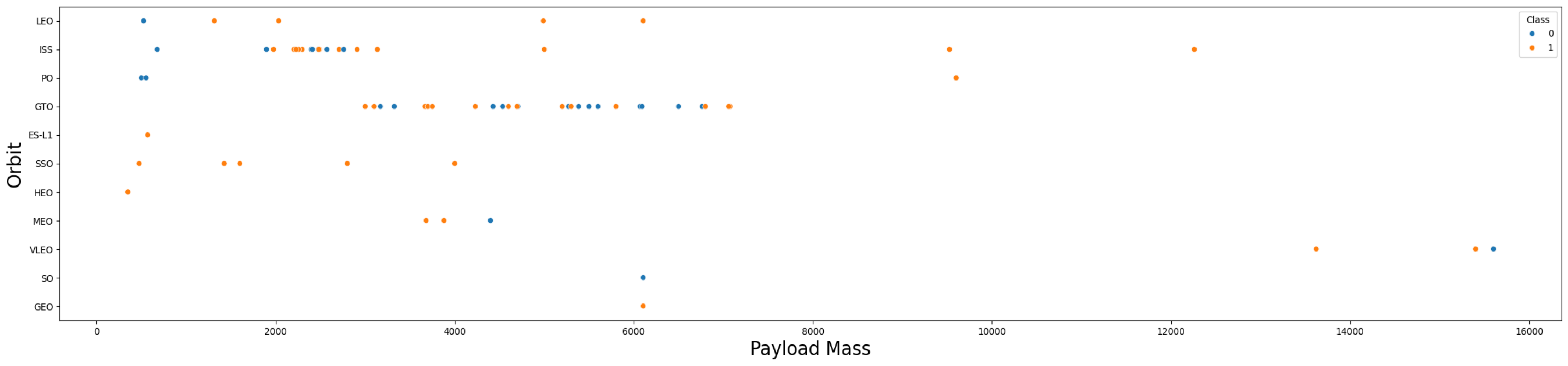
Flight Number vs. Orbit Type

As the number of flights increased, the success of landings also improved.



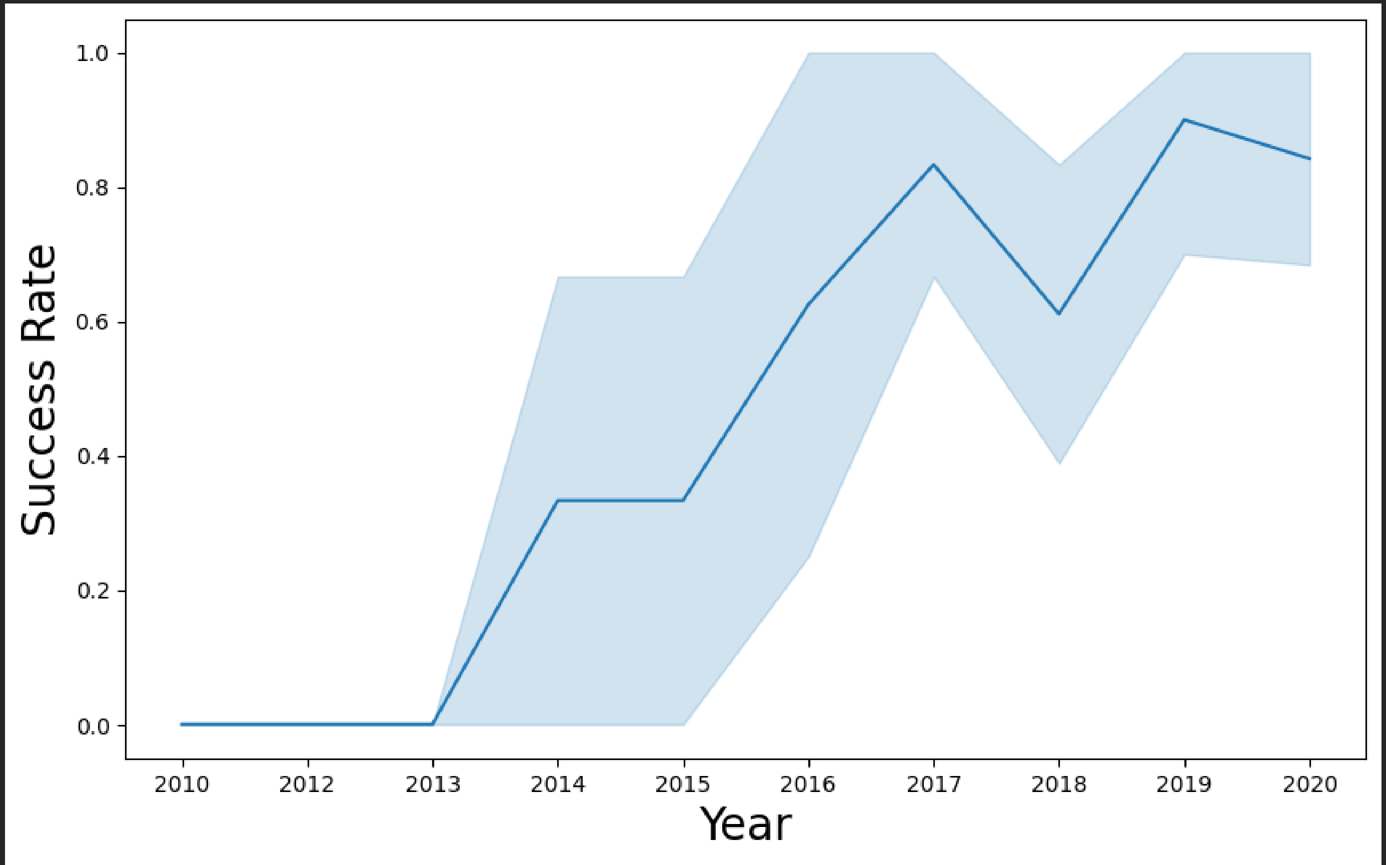
Payload vs. Orbit Type

Successful landing rates for heavy payloads are higher for Polar, LEO, and ISS. However, for lower orbits, we cannot differentiate clearly as both positive and negative landings are present in that payload mass range.



Launch Success Yearly Trend

There is a clear relationship between the success rate and the number of launches, which increased considerably from 2013 until 2017. After a short period of decrease, it started increasing again in 2018.



All Launch Site Names

Names of the launch sites:

	Launch Site
0	CCAFS LC-40
1	VAFB SLC-4E
2	KSC LC-39A
3	CCAFS SLC-40

The four locations are distributed between California and Florida, with VAFB SLC-4E being the only location in California.

I used SQL to obtain the data, and Pandas to create a Data Frame for readability.

Launch Site Names Begin with 'CCA'

First 5 records where launch sites begin with `CCA`

	Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
0	2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
3	2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

I used SQL to obtain the data, and Pandas to create a Data Frame for readability.

Total Payload Mass

Total payload carried by boosters from NASA

Total payload carried by boosters from NASA
99980

I used SQL to obtain the data, and Pandas to create a Data Frame for readability.

Average Payload Mass by F9 v1.1

Average payload mass carried by booster version F9 v1.1

Average Payload Mass by F9 v1.1
2928.4

I used SQL to obtain the data, and Pandas to create a Data Frame for readability.

First Successful Ground Landing Date

Date of the first successful landing outcome on ground pad

First Successful Ground Landing Date
2015-12-22

I used SQL to obtain the data, and Pandas to create a Data Frame for readability.

Successful Drone Ship Landing with Payload between 4000 and 6000

Booster names that have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

	Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
0	2016-05-06	5:21:00	F9 FT B1022	CCAFS LC-40	JCSAT-14	4696	GTO	SKY Perfect JSAT Group	Success	Success (drone ship)
1	2016-08-14	5:26:00	F9 FT B1026	CCAFS LC-40	JCSAT-16	4600	GTO	SKY Perfect JSAT Group	Success	Success (drone ship)
2	2017-03-30	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300	GTO	SES	Success	Success (drone ship)
3	2017-10-11	22:53:00	F9 FT B1031.2	KSC LC-39A	SES-11 / EchoStar 105	5200	GTO	SES EchoStar	Success	Success (drone ship)

I used SQL to obtain the data, and Pandas to create a Data Frame for readability.

Total Number of Successful and Failure Mission Outcomes

Total number of successful and failure mission outcomes.

	Landing Outcome	Total
0	Controlled (ocean)	5
1	Failure	3
2	Failure (drone ship)	5
3	Failure (parachute)	2
4	No attempt	21
5	No attempt	1
6	Precluded (drone ship)	1
7	Success	38
8	Success (drone ship)	14
9	Success (ground pad)	9
10	Uncontrolled (ocean)	2

I used SQL to obtain the data, and Pandas to create a Data Frame for readability.

Boosters Carried Maximum Payload

Names of the booster which have carried the maximum payload mass.

	Booster Version	Payload Mass
0	F9 B5 B1048.4	15600
1	F9 B5 B1049.4	15600
2	F9 B5 B1051.3	15600
3	F9 B5 B1056.4	15600
4	F9 B5 B1048.5	15600
5	F9 B5 B1051.4	15600
6	F9 B5 B1049.5	15600
7	F9 B5 B1060.2	15600
8	F9 B5 B1058.3	15600
9	F9 B5 B1051.6	15600
10	F9 B5 B1060.3	15600
11	F9 B5 B1049.7	15600

I used SQL to obtain the data, and Pandas to create a Data Frame for readability.

2015 Launch Records

Failed 'landing_outcomes' in drone ship, their booster versions,

and launch site names for in year 2015

	Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
0	2015-01-10	9:47:00	F9 v1.1 B1012	CCAFS LC-40	SpaceX CRS-5	2395	LEO (ISS)	NASA (CRS)	Success	Failure (drone ship)
1	2015-04-14	20:10:00	F9 v1.1 B1015	CCAFS LC-40	SpaceX CRS-6	1898	LEO (ISS)	NASA (CRS)	Success	Failure (drone ship)

I used SQL to obtain the data, and Pandas to create a Data Frame for readability.

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

Count of landing outcomes between the date 2010-06-04 and 2017-03-20, in descending order.

	Landing_Outcome	outcome_count
0	No attempt	10
1	Success (drone ship)	6
2	Failure (drone ship)	5
3	Success (ground pad)	3
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Failure (parachute)	2
7	Precluded (drone ship)	1

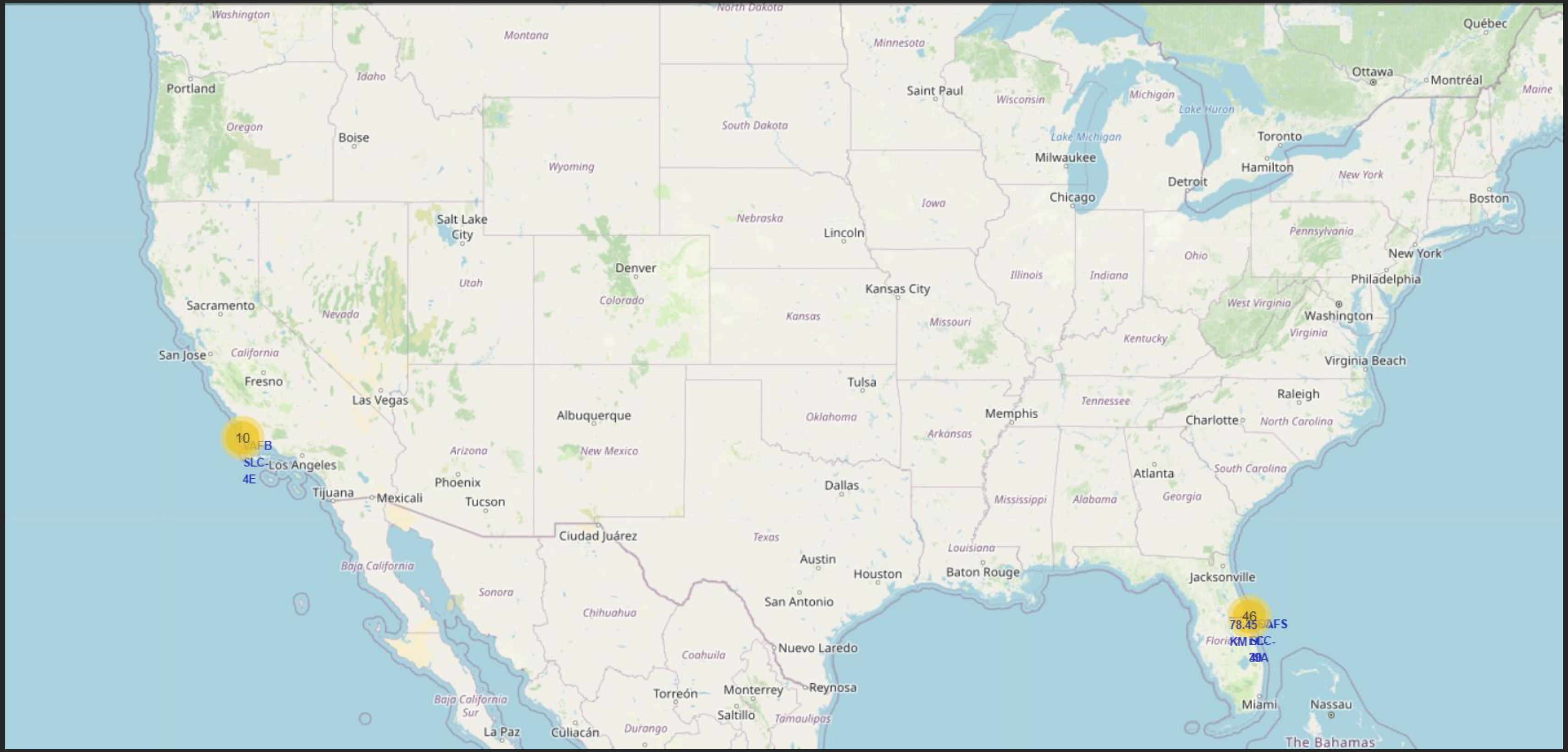
I used SQL to obtain the data, and Pandas to create a Data Frame for readability.

Section 3

Launch Sites Proximities Analysis

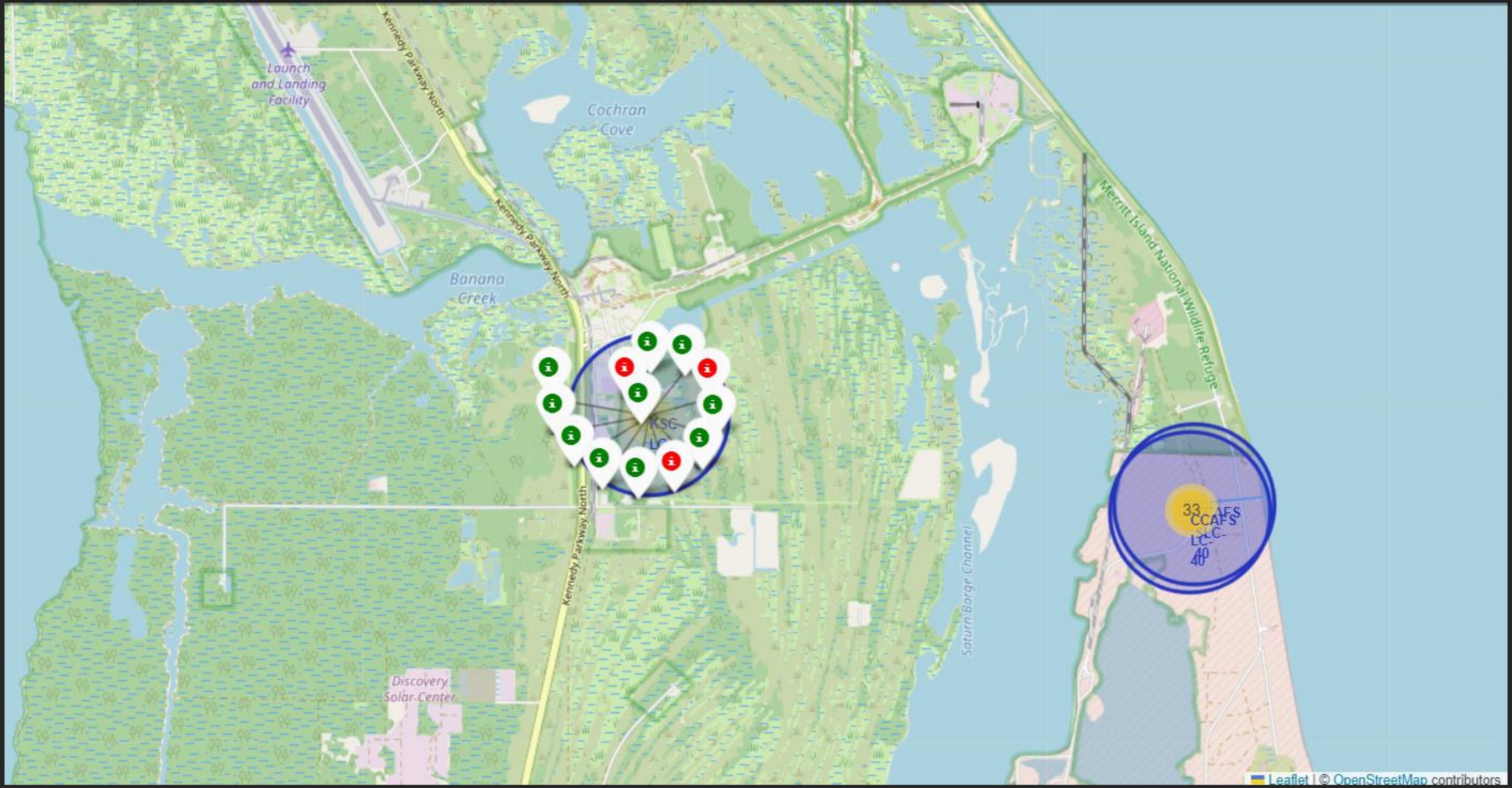
Map of Launch Locations

Different locations used for SpaceX for the launches



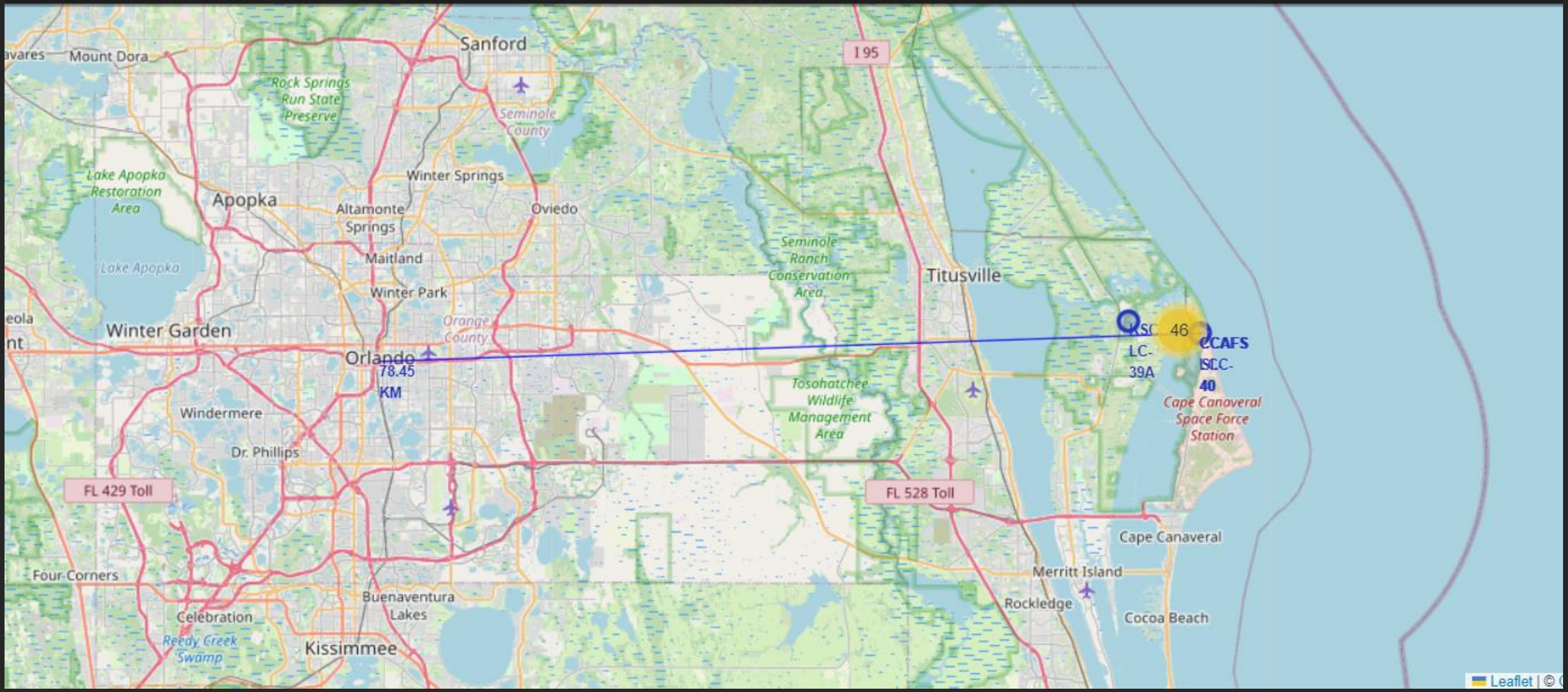
Map of Successful and Failed launches

Comparison between successful and failed launches at KSC LC-39A.



Distance from the launch site to the nearest city

The distance from CCAFS SLC-40 to Orlando is approximately 78.45 kilometers.

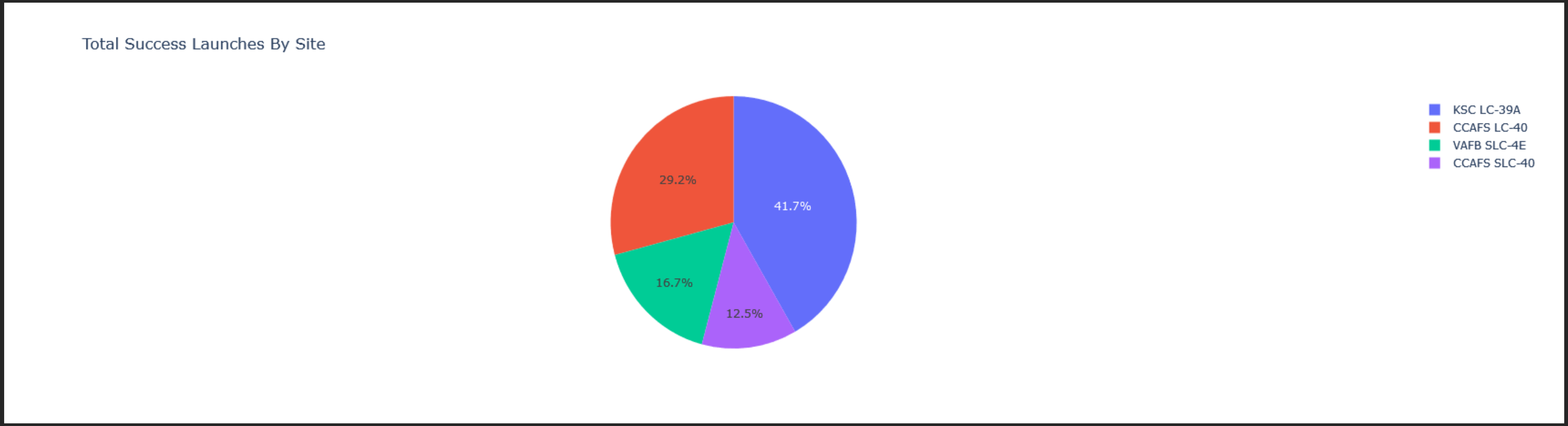


Section 4

Build a Dashboard with Plotly Dash

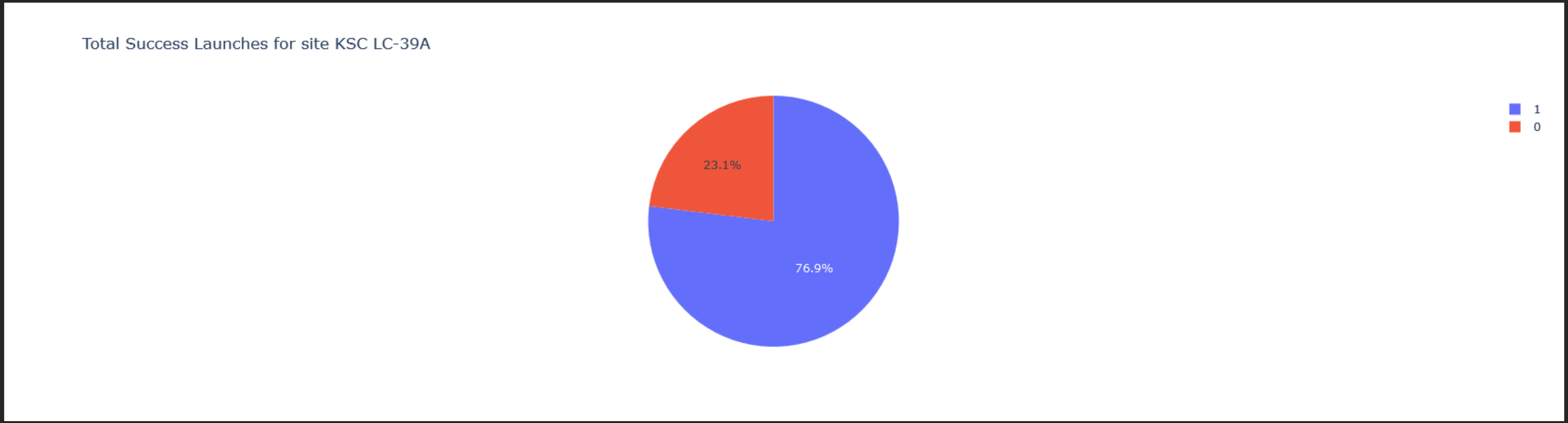
Launch Success Count for all Sites

A pie chart comparing the success rates of launches at all launch sites.



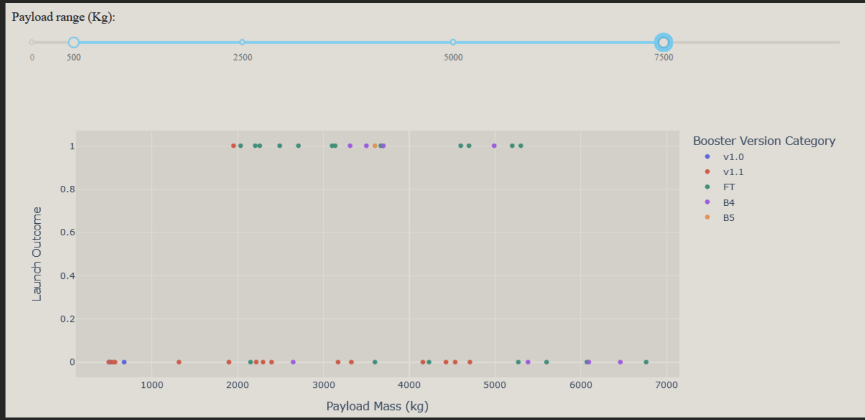
Launch Site with the Highest Success Ratio

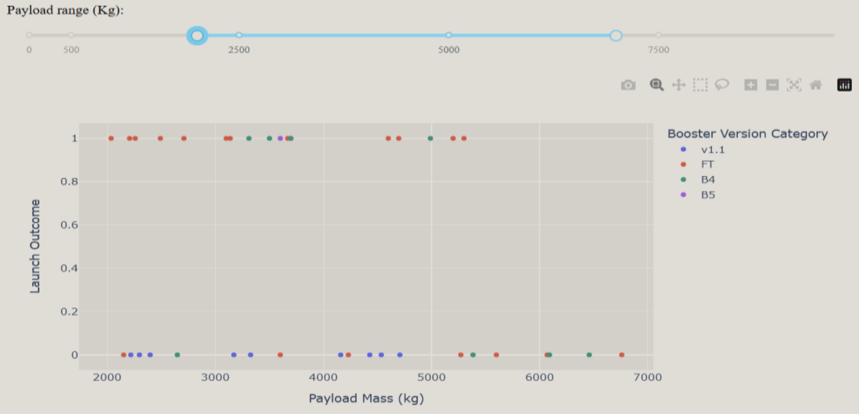
Pie chart of the launch site KSC LC-39A with the highest launch success ratio

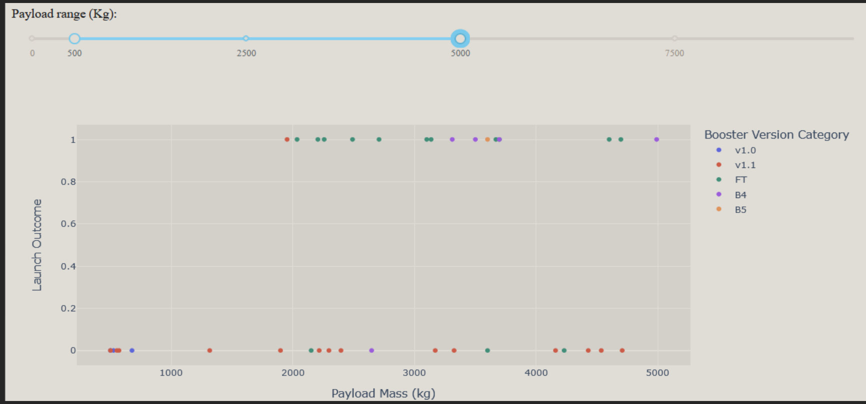


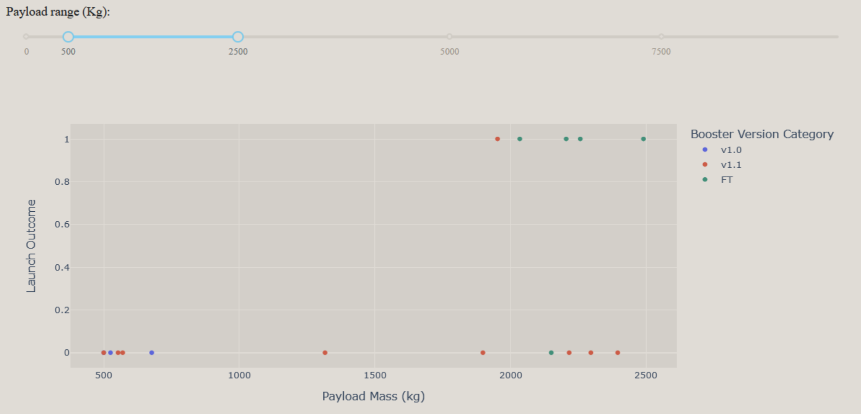
Payload Mass vs. Launch Outcome Scatter plot

The Buster versions FT and B4 have the highest success rate with a payload mass between 1000 to 6000 kilograms.







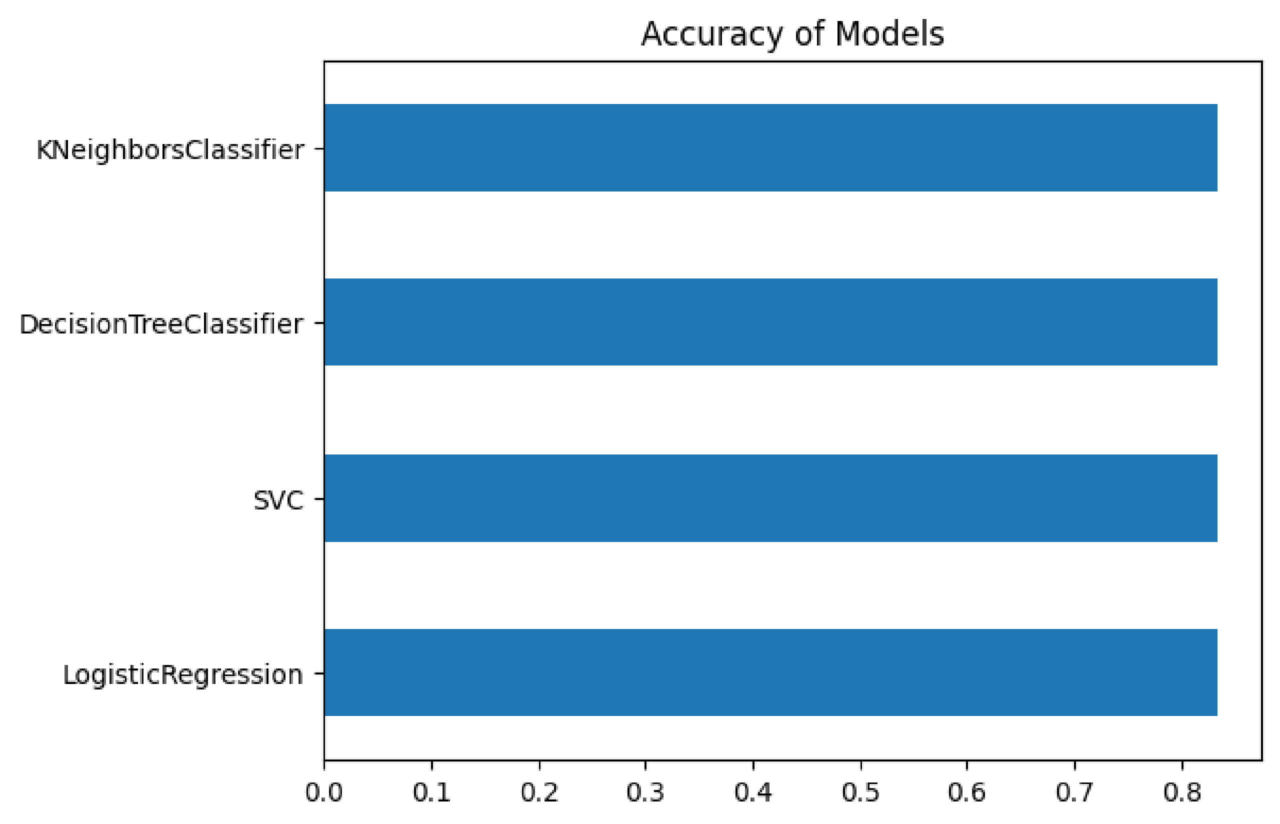


Section 5

Predictive Analysis (Classification)

Classification Accuracy

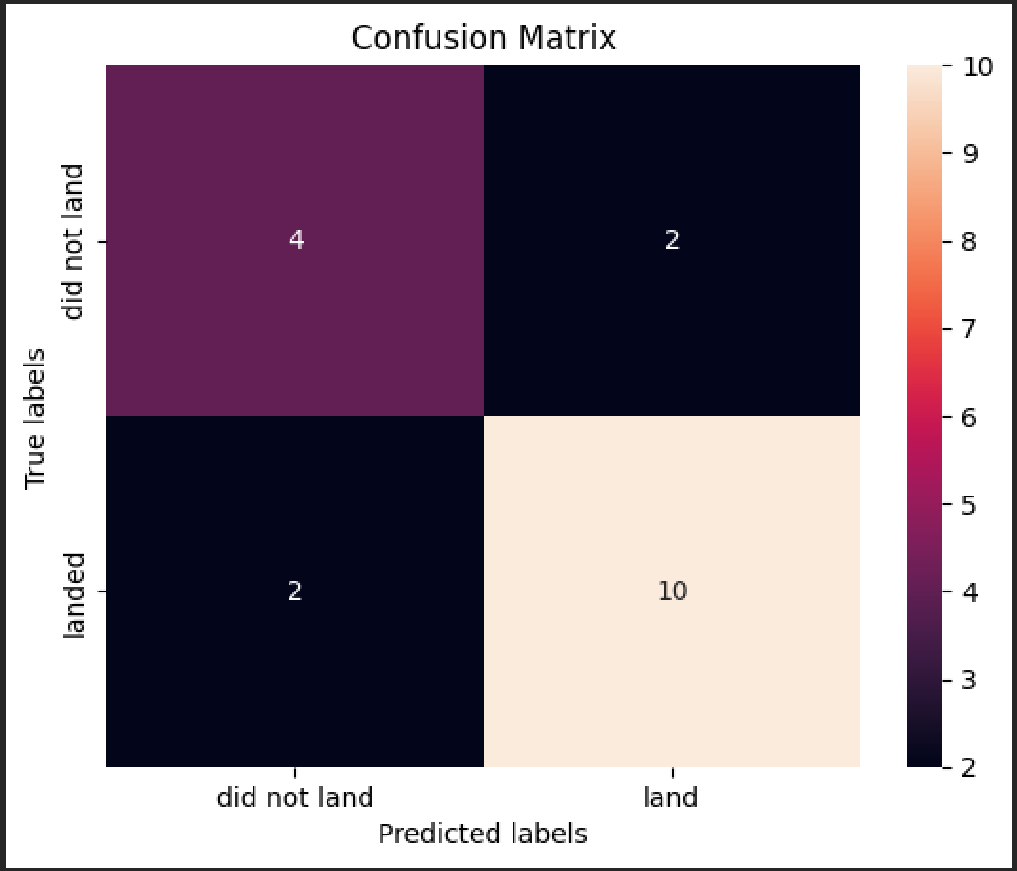
The machine learning analysis indicated that all the prediction models train for this project, have an accuracy of 0.8333.



	LogisticRegression	SVC	DecisionTreeClassifier	KNeighborsClassifier
Accuracy	0.833333	0.833333	0.833333	0.833333

Confusion Matrix

Although all the models have the same accuracy, the confusion matrix from the Decision Tree class model seems to be the most accurate.

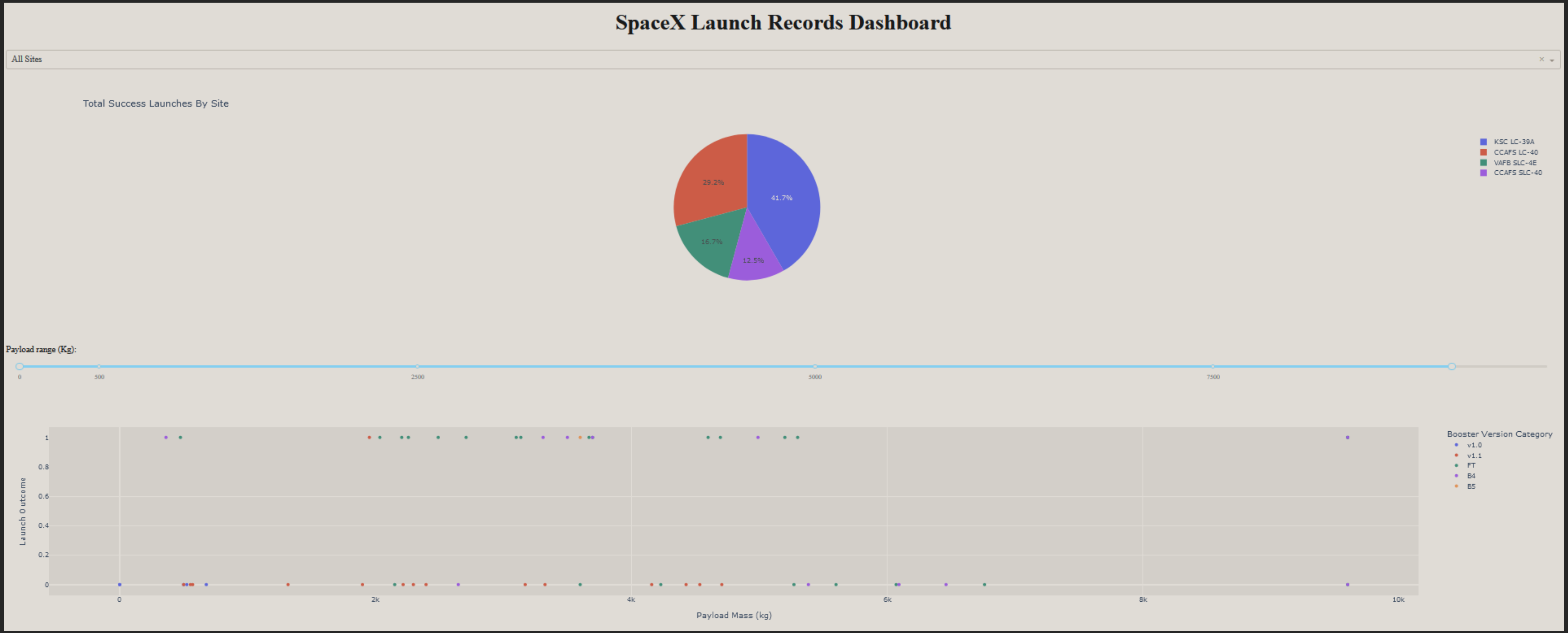


Conclusions

- 'KSC LC 39A' and 'VAFB SLC 4E' have higher success rates for landings compared to 'CCAFS SLC 40', which has only a 60% success rate.
- Most failed landings occur with a payload mass between 500 and 7000.
- A clear relationship between the Success Rate and the Orbit Type.
- Successful landing rates for heavy payloads are higher for Polar, LEO, and ISS.
- There is a clear relationship between the success rate and the number of launches.
- The Buster versions FT and B4 have the highest success rate with a payload mass between 1000 to 6000 kilograms.

Appendix

Interactive dashboard application for visualizing and comparing data.



Thank you!