IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Simon Weber
10.08.2022

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection through API and web scraping
  - Data wrangling
  - Exploratory Data Analysis (EDA) with SQL
  - EDA with data visualization
  - Interactive Visual Analytics with Folium and Dash
  - Machin Learning prediction with different classifiers
- Summary of all results
  - EDA results
  - Interactive Visual Analytics results
  - Machine Learning prediction results

Introduction

- Project background and context
  - SpaceX advertises Falcon 9 rocket launches with a cost of 62 million dollars, while other providers charge 165 million dollars or more. Much of the savings are due to the ability of SpaceX to land and reuse the first stage of the rocket. Thus, if we can determine whether the first stage will land successfully, we can determine the cost of a launch more precisely. The goal of this project is to build a machine learning pipeline to predict if the first stage will land successfully, to use this information for a potential rival company that wants to bid against SpaceX.
- Problems you want to find answers
  - How do we find the best data to build our model on?
  - What factors determine if a rocket will land successfully?
  - What is the best model to predict the landing success?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - We collected data from the SpaceX API and used web scraping on Wikipedia tables
- Perform data wrangling
  - We filtered data for "Falcon 9" entries, replaced missing PayloadMass values with the mean of that feature and added the outcome-variable "class"
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Data was one-hot encoded, standardized and split into train and tests sets
  - We used grid search to build optimal Logistic Regression, SVM, decision tree and k-nearest neighbor models

Data Collection

- We used "get request" to collect data from the SpaceX API
- We decoded the response content as a .json file and turned it into a pandas dataframe
- We removed non-relevant data and replaced missing values whereever necessary

- We performed web scraping for Falcon 9 launch data from Wikipedia using "BeautifulSoup"
- We extracted data from HTML tables and converted it to a pandas dataframe

Data Collection – SpaceX API

- We used get request to the SpaceX API to collect data, turned it into a pandas dataframe using json_normalize(), filtered for "Falcon 9" launches and replaced missing values in the PayloadMass column with its mean

- See https://github.com/visuelcortes/IBM_DataScienceCapstone/blob/main/Data%20Collection%20API.ipynb

```
In [6]:  spacex_url="https://api.spacexdata.com/v4/launches/past"

In [16]: response = requests.get(spacex_url)

In [20]: # Use json_normalize meethod to convert the json result into a dataframe

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

In [48]: # Hint data['BoosterVersion']!='Falcon 1'

         data_falcon9 = launch_data[launch_data['BoosterVersion'] == 'Falcon 9']
         data_falcon9.head()

In [59]: # Calculate the mean value of PayloadMass column

         payload_mean = data_falcon9['PayloadMass'].mean()

         # Replace the np.nan values with its mean value

         data_falcon9['PayloadMass'].replace(np.nan, payload_mean, inplace = True)
         data_falcon9
```

Data Collection - Scraping

- We performed a get request to collect data from a Wikipedia entry, parsed the html to a BeautifulSoup object, extracted the tables embedded in the website and appended the column names and table content to a pandas dataframe

- See https://github.com/visuelcortes/IBM_DataScienceCapstone/blob/main/Data%20Collection%20with%20Web%20Scraping.ipynb

```
In [4]: # use requests.get() method with the provided static_url
        # assign the response to a object

        response = requests.get(static_url)

In [5]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content

        soup = BeautifulSoup(response.content, "html.parser")

In [7]: # Use the find_all function in the BeautifulSoup object, with element type `table`
        # Assign the result to a list called `html_tables`

        html_tables = soup.find_all('table')

In [9]: column_names = []

        # Apply find_all() function with `th` element on first_launch_table
        # Iterate each th element and apply the provided extract_column_from_header() to get a colum
        # Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list cal

        for row in first_launch_table.find_all('th'):
            name = extract_column_from_header(row)
            if name != None and len(name) > 0:
                column_names.append(name)

In [24]: extracted_row = 0
         #Extract each table
         for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collaps
            # get table row
             for rows in table.find_all("tr"):
                 #check to see if first table heading is as number corresponding to launch a number
                 if rows.th:
                     if rows.th.string:
                         flight_number=rows.th.string.strip()
                         flag=flight_number.isdigit()
                 else:
                     flag=False
                 #get table element
                 row=rows.find_all('td')
                 #if it is number save cells in a dictonary
                 if flag:
                     extracted_row += 1
                     # Flight Number value
                     # TODO: Append the flight_number into launch_dict with key `Flight No.`
                     print(flight_number)
                     launch_dict['Flight No.'].append(flight_number)
```

Data Wrangling

- We calculated the number of launches per launch site
- We calculated the number of launches to each orbit
- We counted the occurrences of each landing outcome
- We created a "Class" column, coding for successful und failed landings
- See https://github.com/visuelcortes/IBM_DataScienceCapstone/blob/main/EDA.ipynb

```
In [5]: # Apply value_counts() on column LaunchSite

        df['LaunchSite'].value_counts()

Out[5]: CCAFS SLC 40    55
        KSC LC 39A      22
        VAFB SLC 4E     13
        Name: LaunchSite, dtype: int64
```

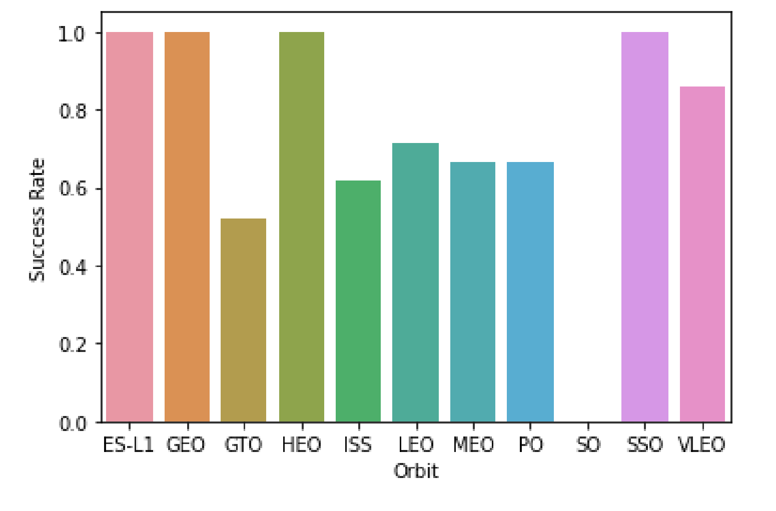
```
In [6]: # Apply value_counts on Orbit column

        df['Orbit'].value_counts()

Out[6]: GTO      27
        ISS      21
        VLEO     14
        PO        9
        LEO       7
        SSO       5
        MEO       3
        ES-L1     1
        HEO       1
        SO        1
        GEO       1
        Name: Orbit, dtype: int64
```

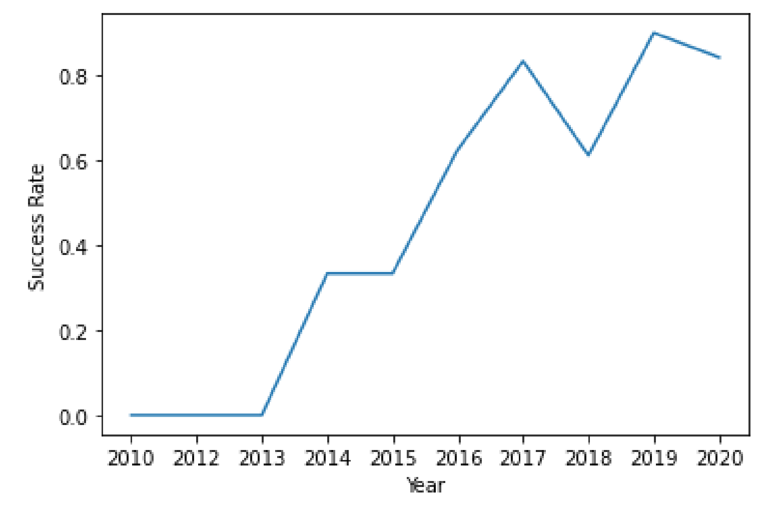
```
In [7]: # landing_outcomes = values on Outcome column

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

Out[7]: True ASDS      41
        None None      19
        True RTLS      14
        False ASDS      6
        True Ocean      5
        False Ocean     2
        None ASDS       2
        False RTLS      1
        Name: Outcome, dtype: int64
```

```
In [13]: # landing_class = 0 if bad_outcome
         # landing_class = 1 otherwise

         landing_class = []

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

EDA with Data Visualization

- We plotted several scatter plots to investigate the relationship of various variable combinations
- We plotted the success rate of each orbit type (top graph)
- We plotted the success rate across time (bottom graph)
- See https://github.com/visuelcortes/IBM_DataScienceCapstone/blob/main/EDA%20with%20Data%20Visualization.ipynb





EDA with SQL

- We loaded the data into a Db2 database and used SQL queries to explore the following aspects of the data:
  - The distinct names of the launch sites
  - The total payload mass carried by boosters launched by NASA (CRS)
  - Average payload mass carried by booster version F9 v1.1
  - The date when the first successful landing outcome in ground pad was achieved
  - The names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
  - The total number of successful and failure mission outcomes
  - The names of the booster versions which have carried the maximum payload mass
  - The failed landing outcomes in drone ship, their booster versions, and launch site names for in year 2015
  - The count of landing outcomes between the date 2010-06-04 and 2017-03-20
- See https://github.com/visuelcortes/IBM_DataScienceCapstone/blob/main/EDA%20with%20SQL.ipynb

Build an Interactive Map with Folium

- We added markers and circles for all launch sites
- We added colored marker clusters with labels to indicate success/failed launches for each site
- We calculated the minimum distance of launch sites to its proximities (such as coast, railways, highways and cities) and added lines and text-markers to indicate them on the map
- See https://github.com/visuelcortes/IBM_DataScienceCapstone/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- We created an interactive dashboard with Plotly dash
- We plotted pie charts to show total launches and their success rate for the different sites, with a dropdown menu to select individual sites
- We plotted scatter plots to show the relationship between Outcome and Payload Mass (Kg) for the different booster version, with a slider to adjust the payload range included in the plot
- See https://github.com/visuelcortes/IBM_DataScienceCapstone/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- We loaded the data into numpy arrays, standardized it and split it into training and testing sets
- Using GridSearchCV, we selected the best parameters for our prediction models based on the accuracy score
- We built logistic regression (see example on right), SVM, decision tree and k-nearest-neighbor models and compared their performance based on accuracy scores and confusion matrices
- See https://github.com/visuelcortes/IBM_DataScienceCapstone/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

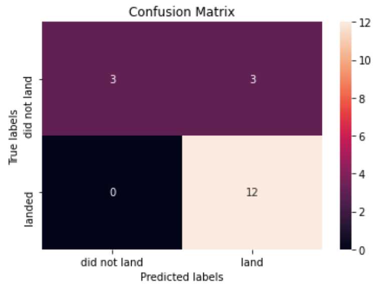
```
In [4]: Y = data['Class'].to_numpy()
        Y

In [6]: # students get this
        transform = preprocessing.StandardScaler().fit(X).transform(X)

In [8]: X_train, X_test, Y_train, Y_test = train_test_split(transform, Y, test_size = 0.2, random_st

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

In [11]: parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
         lr=LogisticRegression()
         logreg_cv = GridSearchCV(lr, parameters, cv=10)
         logreg_cv.fit(X_train, Y_train)

Out[11]: GridSearchCV(cv=10, estimator=LogisticRegression(),
                      param_grid={'C': [0.01, 0.1, 1], 'penalty': ['l2'],
                                  'solver': ['lbfgs']})
```

We output the `GridSearchCV` object for logistic regression. We display the best parameters using the data attribute `best_params_` and the accuracy on the validation data using the data attribute `best_score_`.

```
In [12]: print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
         print("accuracy :",logreg_cv.best_score_)

         tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
         accuracy : 0.8464285714285713

In [13]: logreg_cv.score(X_test, Y_test)

Out[13]: 0.8333333333333334
```

Lets look at the confusion matrix:

```
In [17]: yhat=logreg_cv.predict(X_test)
         plot_confusion_matrix(Y_test,yhat)
```



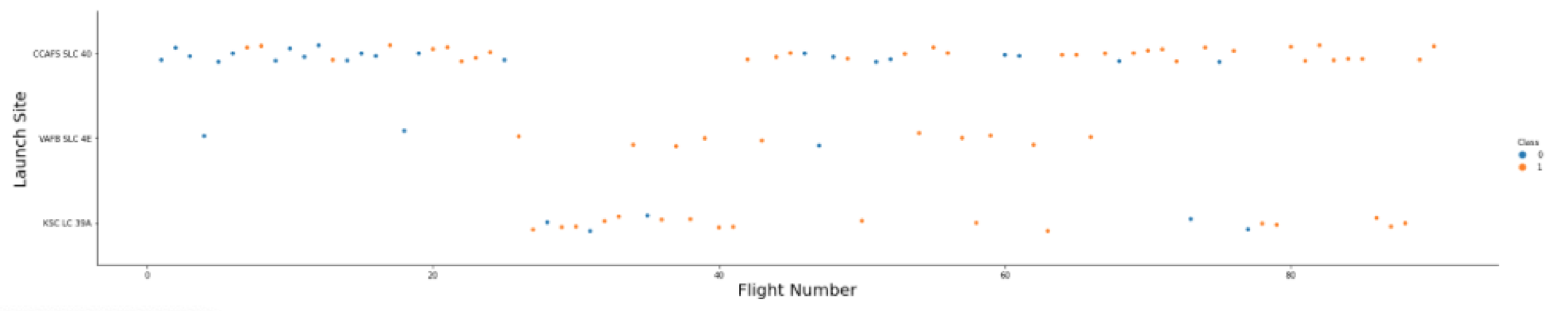
Results

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

Section 2

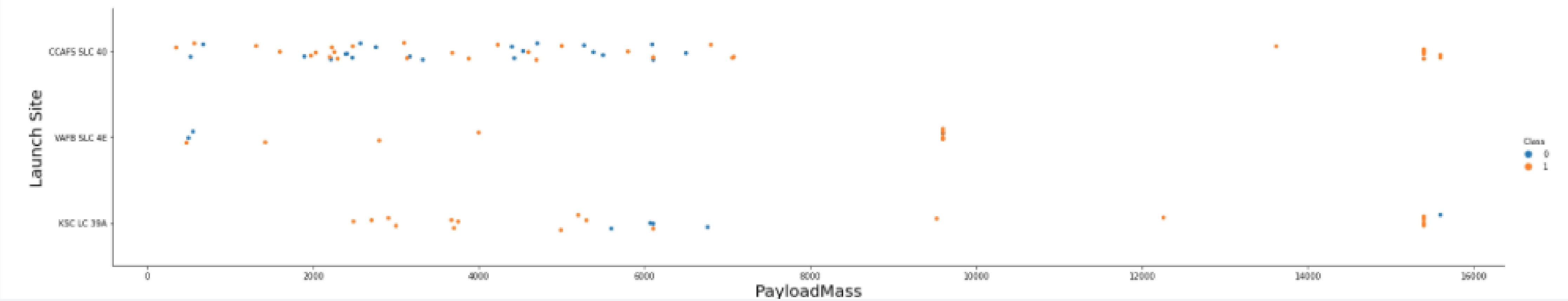
Insights drawn from EDA

Flight Number vs. Launch Site



- We can show that the number of successful launches increases with the number of launches at a site

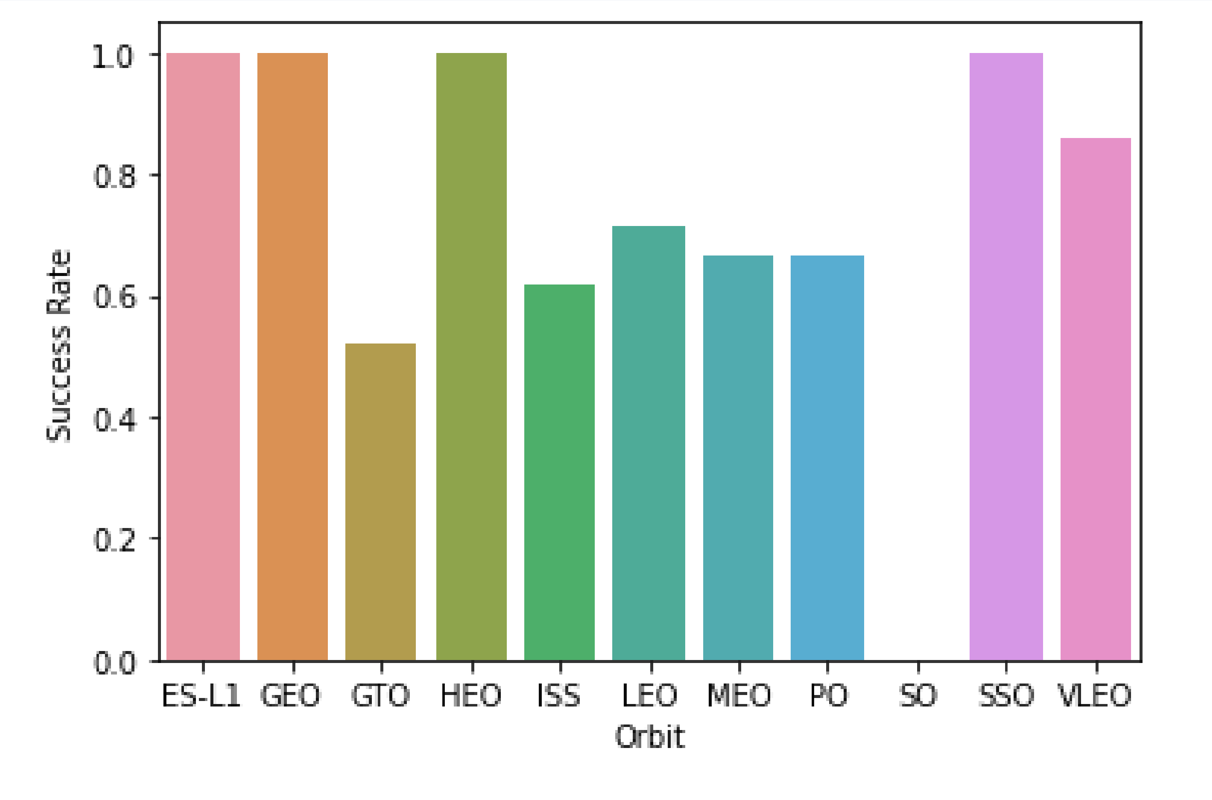
Payload vs. Launch Site



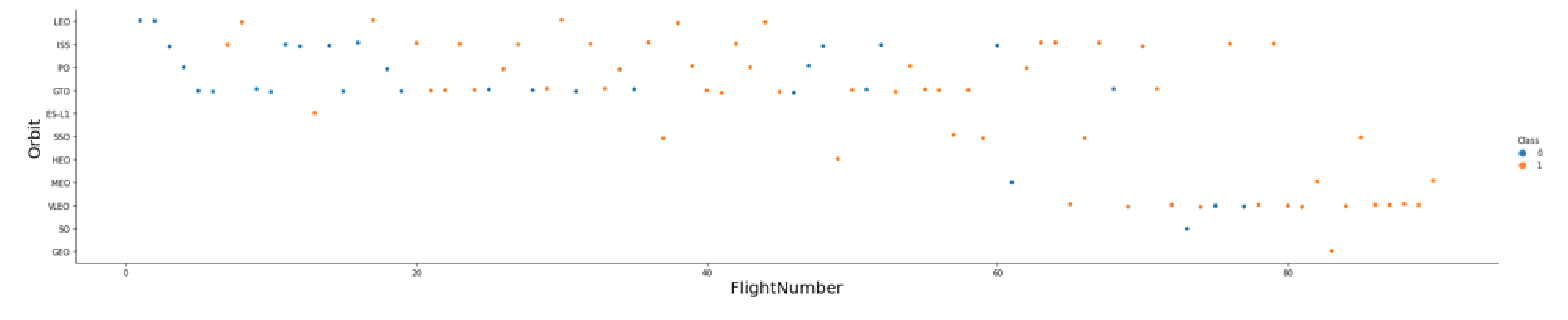
- We can show that for the VAFB-SLC launch site there are no rockets launched for heavy payload mass (greater than 10000)

Success Rate vs. Orbit Type

- We can show that orbits ES-L1, GEO, HEO, SSO and VLEO have the highest success rate while SO has the lowest success rate

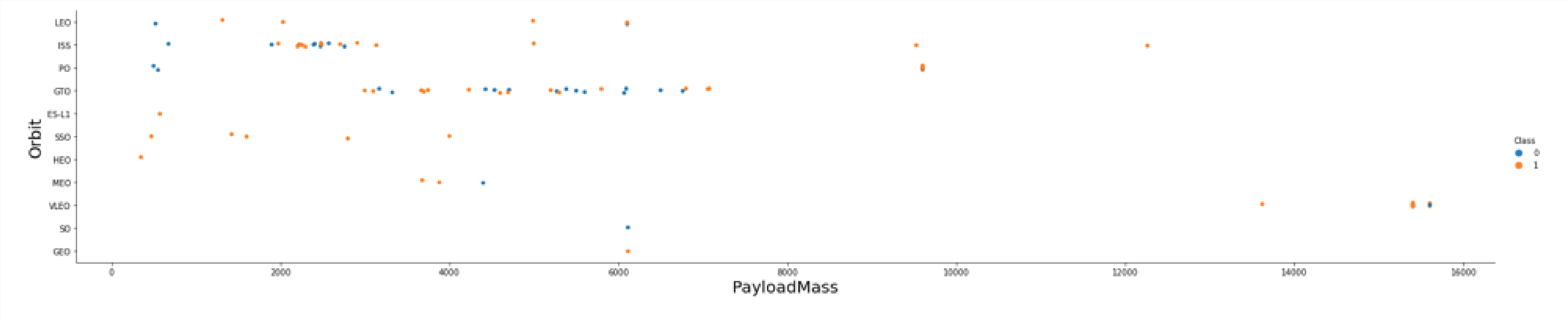


Flight Number vs. Orbit Type



We can show that for the LEO orbit the success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number for the GTO orbit

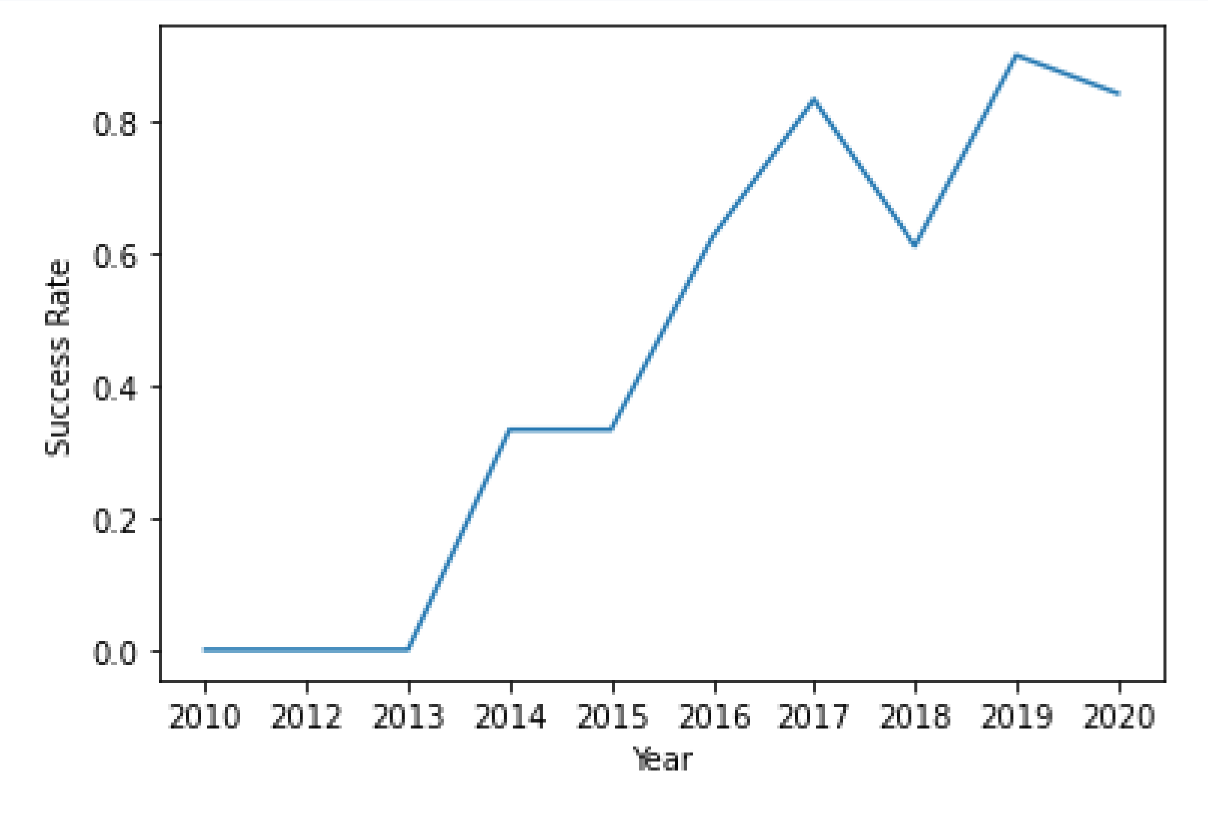
Payload vs. Orbit Type



We can show that heavy payloads the number of successful landings is greater for Polar, LEO and ISS, however for GTO there does not seem to be a comparable trend

Launch Success Yearly Trend

- We can show that the success rate increases across time



All Launch Site Names

```
In [7]: %sql SELECT DISTINCT launch_site FROM SPACEXDATASET;

 * ibm_db_sa://dyy13801:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:324
59/bludb
Done.

Out[7]:
```

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

- We used DISTINCT to show the unique (without repetition) launch site names

Launch Site Names Begin with 'CCA'

```
In [13]: %sql SELECT * FROM SPACEXDATASET \
                 WHERE (launch_site LIKE 'CCA%') \
                 LIMIT 5;

 * ibm_db_sa://dyy13801:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:324
59/bludb
Done.
```

Out[13]:

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

- We used a WHERE condition to select entries with "CCA" and LIMIT to restrict the output to 5 entries

Total Payload Mass

```
In [14]: %sql SELECT SUM(payload_mass__kg_) FROM SPACEXDATASET \
             WHERE (customer LIKE '%NASA%');

 * ibm_db_sa://dyy13801:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32459/bludb
Done.
```

Out[14]:

1
107010

- We used SUM to calculate the total payload mass for NASA launches, which is 107010 kg

Average Payload Mass by F9 v1.1

In [15]:
```
%sql SELECT AVG(payload_mass__kg_) FROM SPACEXDATASET \
    WHERE (booster_version LIKE '%F9 v1.1%');
```

```
 * ibm_db_sa://dyy13801:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32459/bludb
Done.
```

Out[15]:

1
2534

- We used AVG to calculate the average payload mass for F9 v1.1 rockets

First Successful Ground Landing Date

```
In [18]: %sql SELECT MIN(DATE) FROM SPACEXDATASET \
            WHERE landing__outcome LIKE 'Success (ground pad)';

 * ibm_db_sa://dyy13801:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32459/bludb
Done.

Out[18]:
```

1
2015-12-22

- We used MIN to find the smallest data that corresponded to a successful ground pad landing

Successful Drone Ship Landing with Payload between 4000 and 6000

```
In [20]: %sql SELECT DISTINCT booster_version FROM SPACEXDATASET \
             WHERE landing__outcome LIKE 'Success (drone ship)' \
             AND payload_mass__kg_ BETWEEN 4000 AND 6000;

 * ibm_db_sa://dyy13801:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:324
59/bludb
Done.
```

Out[20]:

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

- We used DISTINCT to list the unique booster versions that successfully landed on a drone ship with a payload between 4000 and 6000 kg

Total Number of Successful and Failure Mission Outcomes

In [30]:
```
%sql SELECT DISTINCT(mission_outcome), COUNT(mission_outcome) FROM SPACEXDATASET \
    GROUP BY mission_outcome;
```

```
 * ibm_db_sa://dyy13801:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:324
59/bludb
Done.
```

Out[30]:

mission_outcome	2
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

- We listed and counted each distinct mission outcome

Boosters Carried Maximum Payload

```
In [39]: %sql SELECT DISTINCT(booster_version), payload_mass__kg_ FROM SPACEXDATASET \
             WHERE payload_mass__kg_ = (SELECT MAX(payload_mass__kg_) FROM SPACEXDATASET);

 * ibm_db_sa://dyy13801:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:324
59/bludb
Done.
```

Out[39]:

booster_version	payload_mass__kg_
F9 B5 B1048.4	15600
F9 B5 B1048.5	15600
F9 B5 B1049.4	15600
F9 B5 B1049.5	15600
F9 B5 B1049.7	15600
F9 B5 B1051.3	15600
F9 B5 B1051.4	15600
F9 B5 B1051.6	15600
F9 B5 B1056.4	15600
F9 B5 B1058.3	15600
F9 B5 B1060.2	15600
F9 B5 B1060.3	15600

- We listed the booster versions that carried the maximum payload mass

2015 Launch Records

```
In [41]: %sql SELECT DATE, landing__outcome, booster_version, launch_site FROM SPACEXDATASET \
             WHERE landing__outcome LIKE '%Failure (drone ship)%' \
             AND YEAR(DATE) = 2015;

 * ibm_db_sa://dyy13801:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:324
59/bludb
Done.
```

Out[41]:

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

- We listed the instances of failed drone ship landings in 2015 with their booster version

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

```
In [54]: %sql SELECT landing__outcome, COUNT(landing__outcome) AS outcome_count FROM SPACEXDATASET \
             WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' \
             GROUP BY landing__outcome \
             ORDER BY outcome_count DESC;

 * ibm_db_sa://dyy13801:***@9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:324
59/bludb
Done.
```

Out[54]:

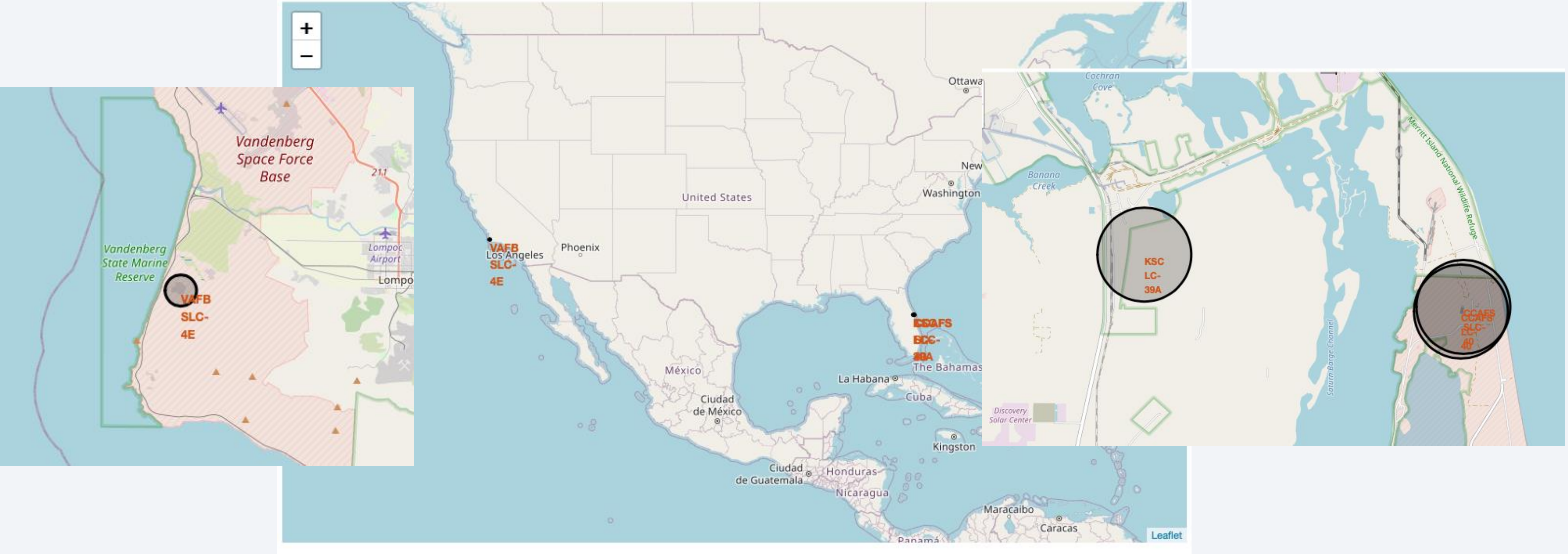
landing__outcome	outcome_count
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

- We ranked all landing outcomes between 2010-06-04 and 2017-03-20 according to the number of their occurrences

Section 3

Launch Sites Proximities Analysis

Location of all Launch sites

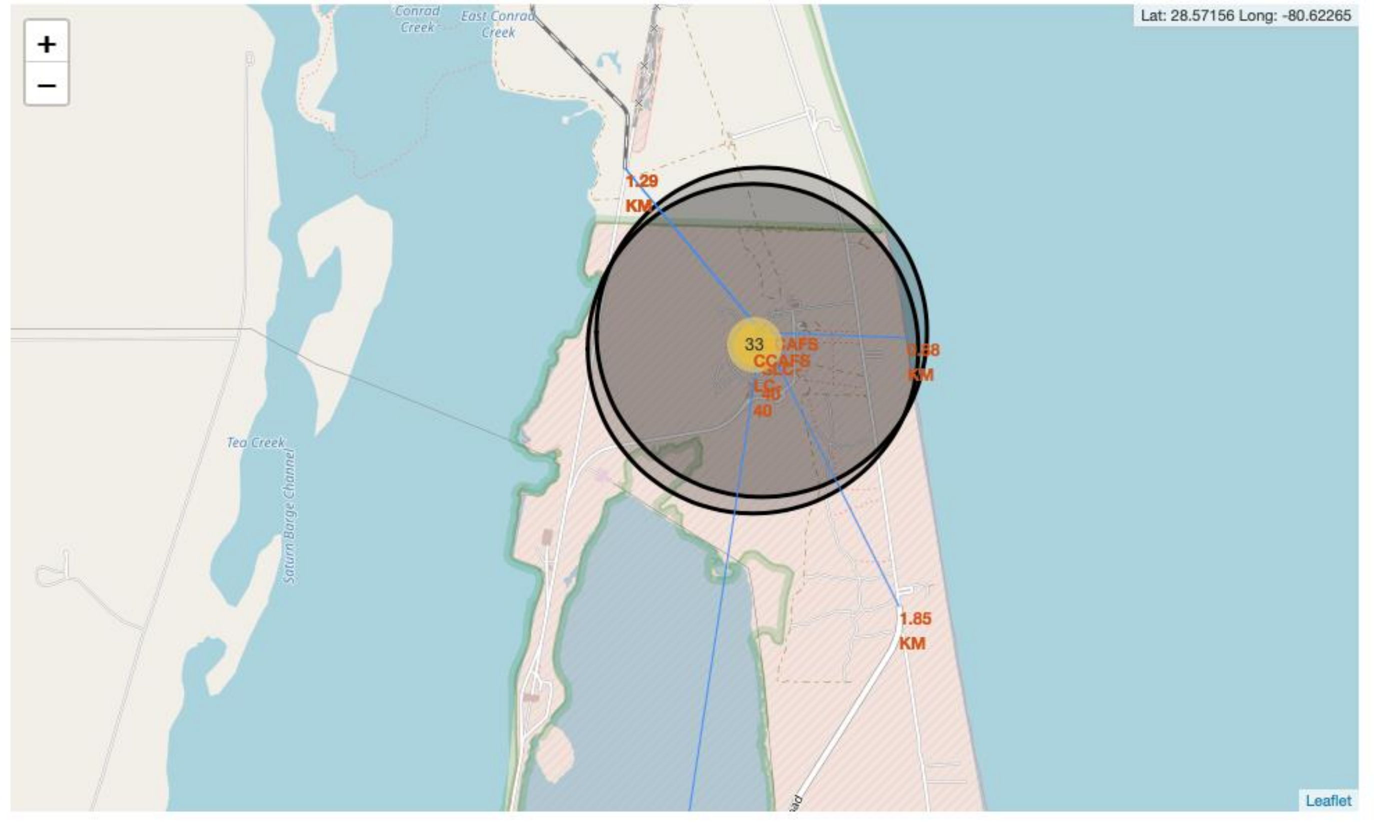


- Folium Map and markers for all launch sites on the east and west coast of the USA

Success-Markers for launches at CCAFS LC-40 and CCAFS SLC-40

- We show one marker for each launch at CCAFS LC-40 (left) and CCAFS SLC-40 (right), with green markers indicating successful launches and red markers indicating failed launches

Proximity of CCAFS SLC-40 to infrastructure

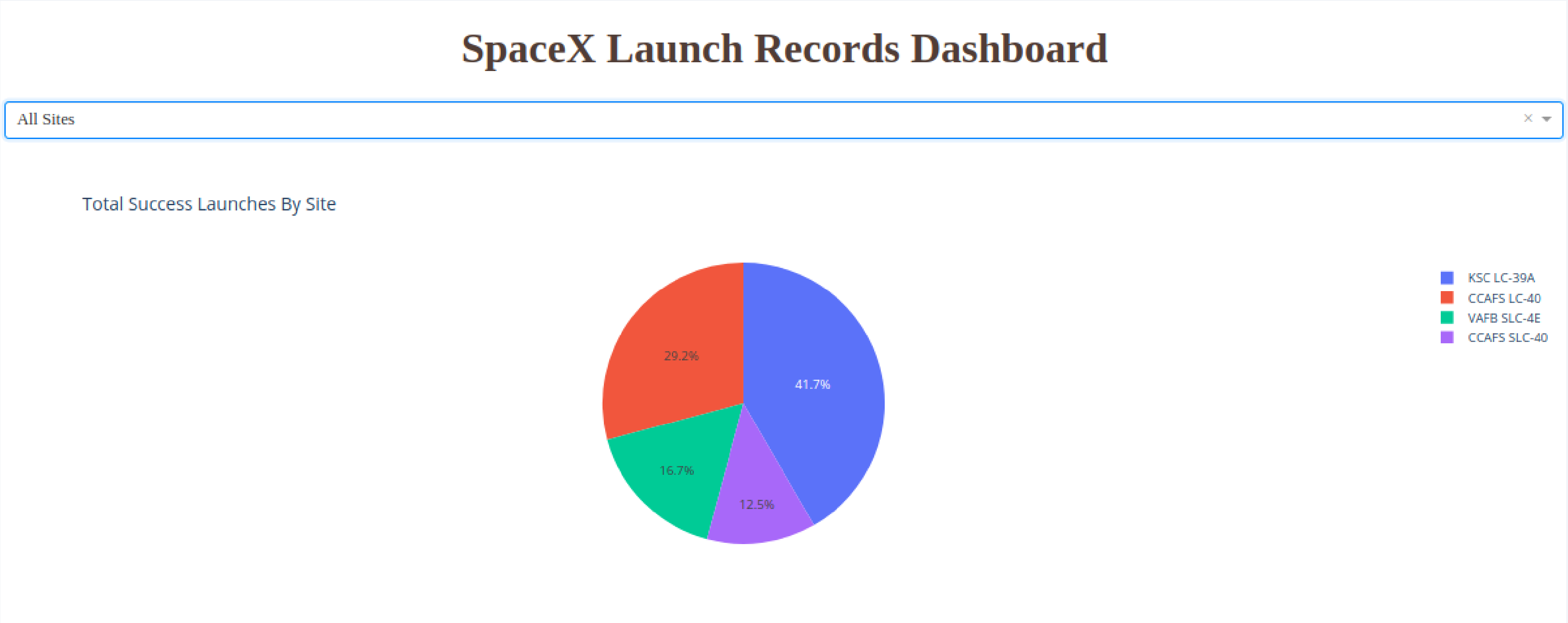


- The blue lines indicate the distance of CCAFS SLC-40 to the closest coast (0.88 km), railway (1.25 km), highway (1.85 km) and city (Cape Canaveral, 18.08 km, not shown)

Section 4

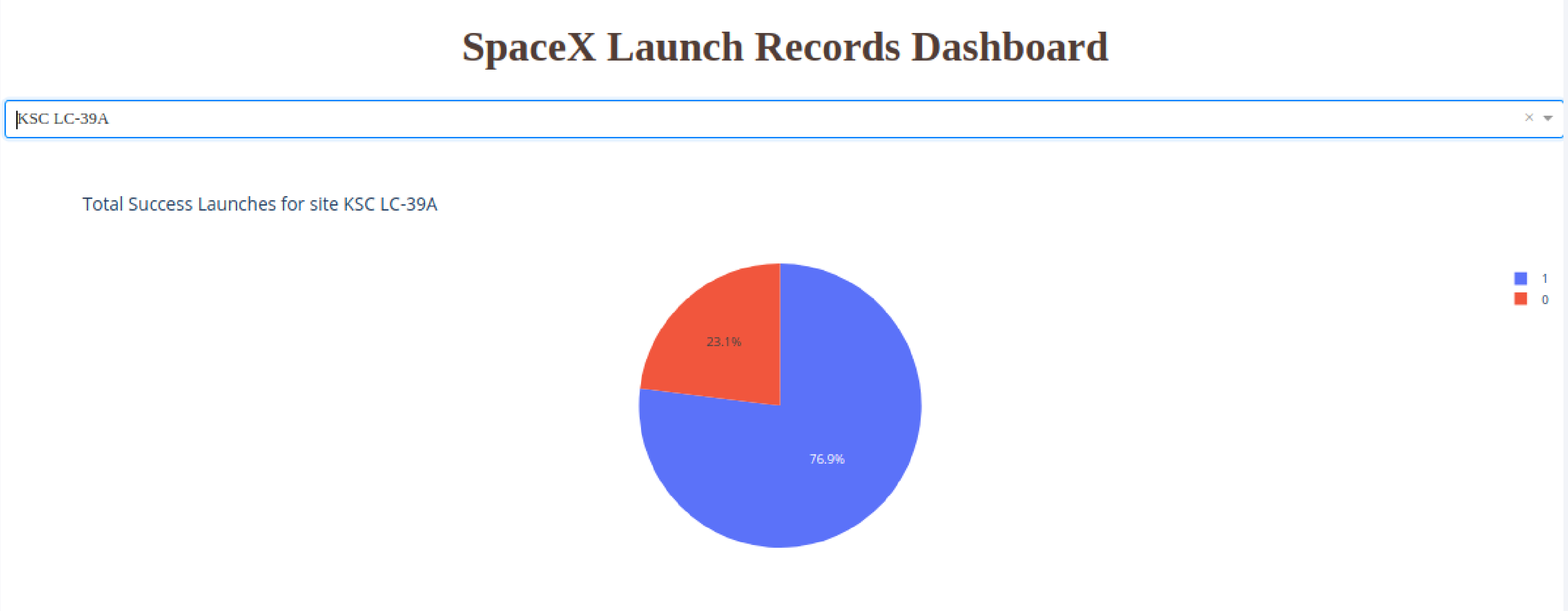
Build a Dashboard with Plotly Dash

Total Success Launches By Site



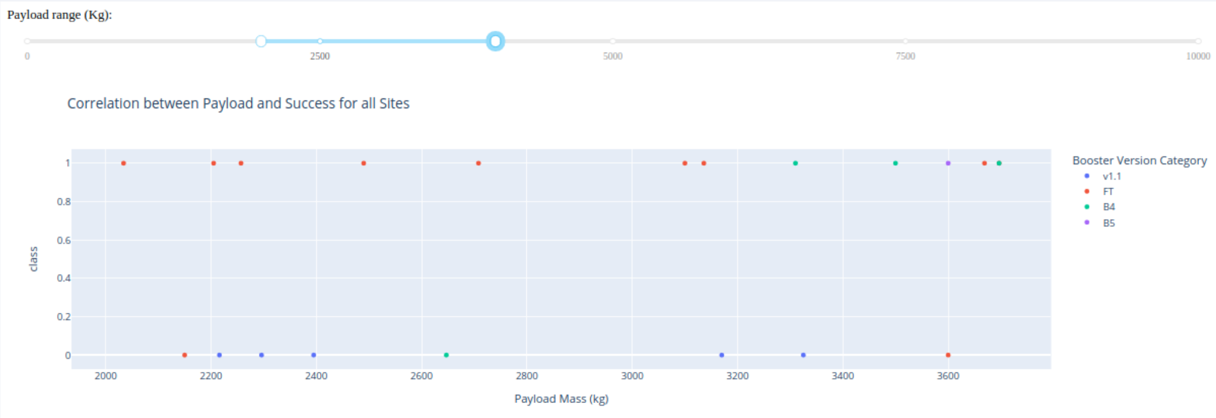
- We show in a pie chart that the site KSC LC-39A has the highest launch success rate, and CCAFS SLC-40 has the lowest success rate

Total Success Launches for site KSC LC-39A



- We show in a pie cahrt that the site KSC LC-39A has a 76.9% success rate

Correlation between Payload and Success for all Sites



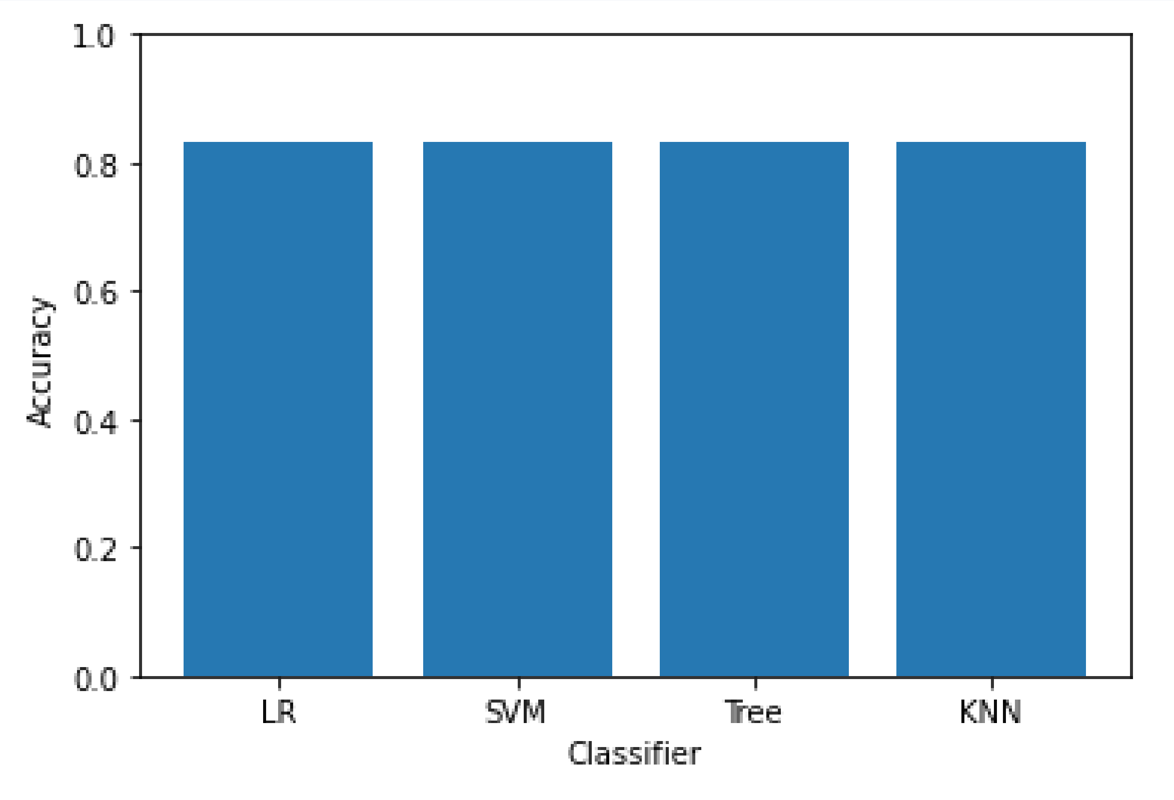
- We show in two scatter plots that Payloads between 2000 and 4000 kg have a particularly high success rate, while Payloads of less than 200 kg have a particularly low success rate

Section 5

Predictive Analysis (Classification)

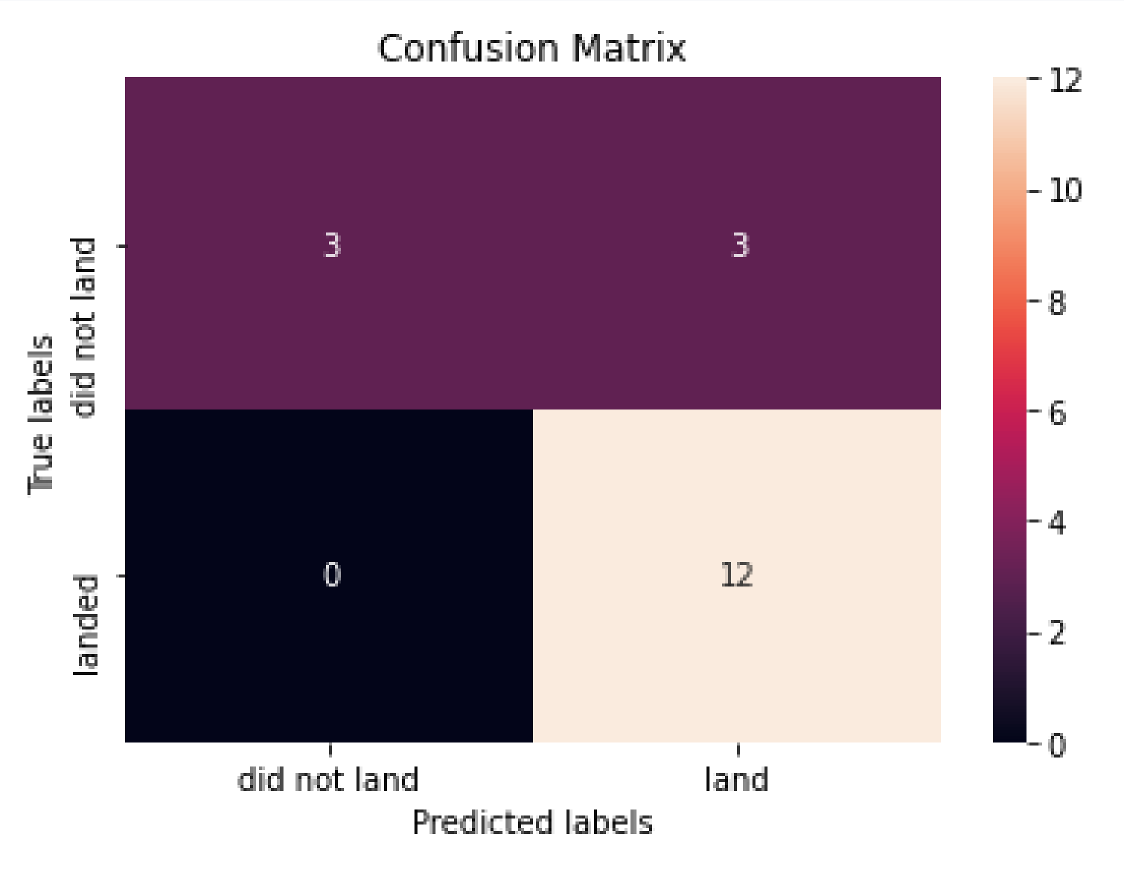
Classification Accuracy

- We show that all classification models have the same accuracy of 0.833334



Confusion Matrix

- The confusion matrices for all models were identical. Each classifier correctly identified 12 landings and 3 failures. However, 3 failures were falsely classified as landings. No failures were misclassified as successes.



Conclusions

We can conclude that:

- The more rockets are launched at a site, the greater is the success rate at that site
- Launch success rate increased substantially between 2013 until 2020
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the highest success rate
- KSC LC-39A had the most successful launches of any sites
- All classifiers predict launch success equally well

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

- Link to GitHub repo: https://github.com/visuelcortes/IBM_DataScienceCapstone

Thank you!