IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Rami Lahoud
5 January 2023

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection via API, Web Scraping
  - Exploratory Data Analysis (EDA) with Data Visualization
  - EDA with SQL
  - Interactive Map with Folium
  - Dashboards with Plotly Dash
  - Predictive Analysis
- Summary of all results
  - Exploratory Data Analysis results
  - Interactive maps and dashboard
  - Predictive results

Introduction

- Project background and context
  - The objective of this project is to predict if Falcon 9 first stage will successfully land. SpaceX indicates on its website that the Falcon 9 rocket launch costs 62M dollars. Other providers cost upward of 165M dollars each. The price difference is explained by the fact that SpaceX can reuse the first stage. By determining if the stage will land, we can determine the cost of a launch. This information is helpful for another company, if it wants to compete with SpaceX for a rocket launch.
- Problems you want to find answers
  - What are the main characteristics of a successful or failed landing?
  - What are the effects of each relationship of the rocket variables on the success or failure of a landing?
  - What are the conditions which will allow SpaceX to achieve the best landing success rate?

Section 1

Methodology

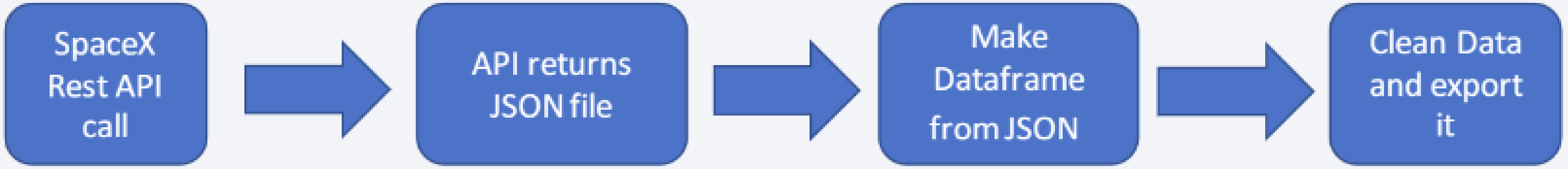
Methodology

Executive Summary

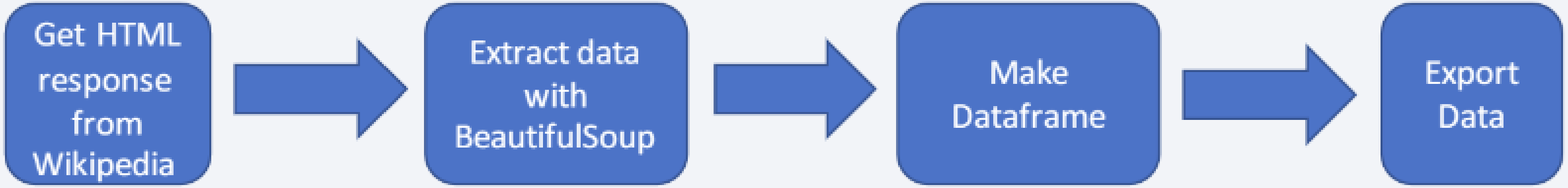
- Data collection methodology:
  - SpaceX REST API
  - Web Scrapping from Wikipedia
- Perform data wrangling
  - Dropping unnecessary columns
  - One Hot Encoding for classification models.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

- Datasets are collected from REST SpaceX API and Web Scraping Wikipedia .
  - The information obtained by the API are rocket, launches, and payload information.
    - The SpaceX REST API URL is https://api.spacexdata.com/v4/



- The information obtained by Web Scraping of Wikipedia are launches, landing, and payload information.
  - URL is https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922



Data Collection – SpaceX API

1. Getting Response from API

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

2. Convert Response to JSON File

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

3. Transform data

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

4. Create dictionary with 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}
```

5. Create dataframe

```
data = pd.DataFrame.from_dict(launch_dict)
```

6. Filter dataframe

```
data_falcon9 = data[data.BoosterVersion == 'Falcon 9']
```

7. Export to file

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

https://github.com/RLAH84/IBM-Data-Science-Capstone-Project/blob/2ebc38b442287efa3b1ae920f0e7a390fb924106/Data%20Collection%20API%20Lab.ipynb

Data Collection - Scraping

1. Getting Response from HTML

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

2. Create BeautifulSoup Object

```
soup = BeautifulSoup(data,"html.parser")
```

3. Find all tables

```
html_tables = soup.findAll('table')
```

4. Get column names

```
column_names = []
table_headers = first_launch_table.find_all('th')
for j, table_header in enumerate(table_headers):
    name = extract_column_from_header(table_header)
    if name is not None and len(name) > 0:
        column_names.append(name)
```

5. Create dictionary

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

6. Add data to keys

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is a
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.stri
                flag=flight_number.isdigit()
```

See notebook for the rest of code

7. Create dataframe from dictionary

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

8. Export to file

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

https://github.com/RLAH84/IBM-Data-Science-Capstone-Project/blob/2ebc38b442287efa3b1ae920f0e7a390fb924106/Data%20Collection%20with%20Web%20Scraping.ipynb

Data Wrangling

- In the dataset, there are several cases where the booster did not land successfully
  - True Ocean, True RTLS, and True ASDS means the mission was successful.
  - False Ocean, False RTLS, and False ASDS means the mission was a failure.
- We need to transform String variables into Categorical variables where 1 means the mission was successful and 0 means the mission was a failure.

1. Calculate launches number for 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 and occurence of each orbit

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

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

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

```
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 landing outcome label from Outcome column

```
landing_class = [0 if x in bad_outcomes
                 else 1 for x in df['Outcome']]
df['Class']=landing_class
print(df[['Class']].head(8))
print(df["Class"].mean())
print(df.head(5))
```

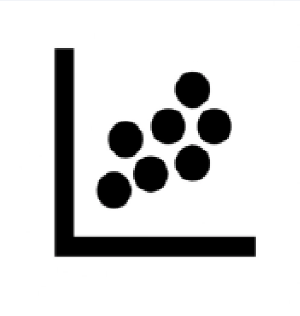
5. Export to file

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

https://github.com/RLAH84/IBM-Data-Science-Capstone-Project/blob/4e8f04bbaecec2d06a9868c3744707ed485a41f7/EDA.ipynb

EDA with Data Visualization

- Scatter Graphs
  - Flight Number vs. Payload Mass
  - Flight Number vs. Launch Site
  - Payload vs. Launch Site
  - Orbit vs. Flight Number
  - Payload vs. Orbit Type
  - Orbit vs. Payload Mass



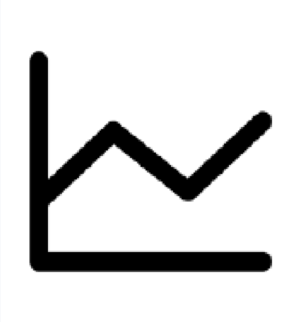
Scatter plots show relationship between variables. This relationship is called the correlation.

- Bar Graph
  - Success rate vs. Orbit

Bar graphs show the relationship between numeric and categoric variables.

- Line Graph
  - Success rate vs. Year

Line graphs show data variables and their trends. Line graphs can help to show global behavior and make prediction for unseen data.



https://github.com/RLAH84/IBM-Data-Science-Capstone-Project/blob/4e8f04bbaecec2d06a9868c3744707ed485a41f7/EDA%20with%20Data%20Visualization.ipynb

EDA with SQL

- We performed SQL queries to gather and understand data from dataset:
  - Displaying the names of the unique launch sites in the space mission.
  - Display 5 records where launch sites begin with the string 'CCA'
  - Display the total payload mass carried by boosters launched by NASA (CRS).
  - Display average payload mass carried by booster version F9 v1.1.
  - List the date when the first successful landing outcome in ground pad was achieved.
  - List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000.
  - List the total number of successful and failure mission outcomes.
  - List the names of the booster versions which have carried the maximum payload mass.
  - List the records which will display the month names, failure landing outcomes in drone ship, booster versions, launch site for the months in year 2015.
  - Rank the count of successful landing outcomes between the date 2010-06-04 and 2017-03-20 in descending order.

https://github.com/RLAH84/IBM-Data-Science-Capstone-Project/blob/4e8f04bbaecec2d06a9868c3744707ed485a41f7/EDA%20with%20SQL.ipynb

Build an Interactive Map with Folium

- Folium map object is a map centered on NASA Johnson Space Center at Houston, Texas
  - Red circle at NASA Johnson Space Center's coordinate with label showing its name (folium.Circle, folium.map.Marker).
  - Red circles at each launch site coordinates with label showing launch site name (folium.Circle, folium.map.Marker, folium.features.DivIcon).
  - The grouping of points in a cluster to display multiple and different information for the same coordinates (folium.plugins.MarkerCluster).
  - Markers to show successful and unsuccessful landings. Green for successful landing and Red for unsuccessful landing. (folium.map.Marker, folium.Icon).
  - Markers to show distance between launch site to key locations (railway, highway, coastway, city) and plot a line between them. (folium.map.Marker, folium.PolyLine, folium.features.DivIcon*)*

- These objects are created in order to understand better the problem and the data. We can show easily all launch sites, their surroundings and the number of successful and unsuccessful landings.

https://github.com/RLAH84/IBM-Data-Science-Capstone-Project/blob/4e8f04bbaecec2d06a9868c3744707ed485a41f7/Interactive%20Visual%20Analytics%20with%20Folium%20Lab.ipynb

Build a Dashboard with Plotly Dash

- Dashboard has dropdown, pie chart, rangeslider and scatter plot components
  - Dropdown allows a user to choose the launch site or all launch sites (dash_core_components.Dropdown).
  - Pie chart shows the total success and the total failure for the launch site chosen with the dropdown component(plotly.express.pie).
  - Rangeslider allows a user to select a payload mass in a fixed range (dash_core_components.RangeSlider).
  - Scatter chart shows the relationship between two variables, in particular Success vs Payload Mass (plotly.express.scatter).

https://github.com/RLAH84/IBM-Data-Science-Capstone-Project/blob/a2df370f19dba1564a7f9842fee9d39f9ce99dc1/spacex_dash_app.py

Predictive Analysis (Classification)

- Data preparation
  - Load dataset
  - Normalize data
  - Split data into training and test sets.
- Model preparation
  - Selection of machine learning algorithms
  - Set parameters for each algorithm to GridSearchCV
  - Training GridSearchModel models with training dataset
- Model evaluation
  - Get best hyperparameters for each type of model
  - Compute accuracy for each model with test dataset
  - Plot Confusion Matrix
- Model comparison
  - Comparison of models according to their accuracy
  - The model with the best accuracy will be chosen (see Notebook for result)

https://github.com/RLAH84/IBM-Data-Science-Capstone-Project/blob/a2df370f19dba1564a7f9842fee9d39f9ce99dc1/Machine%20Learning%20Prediction.ipynb

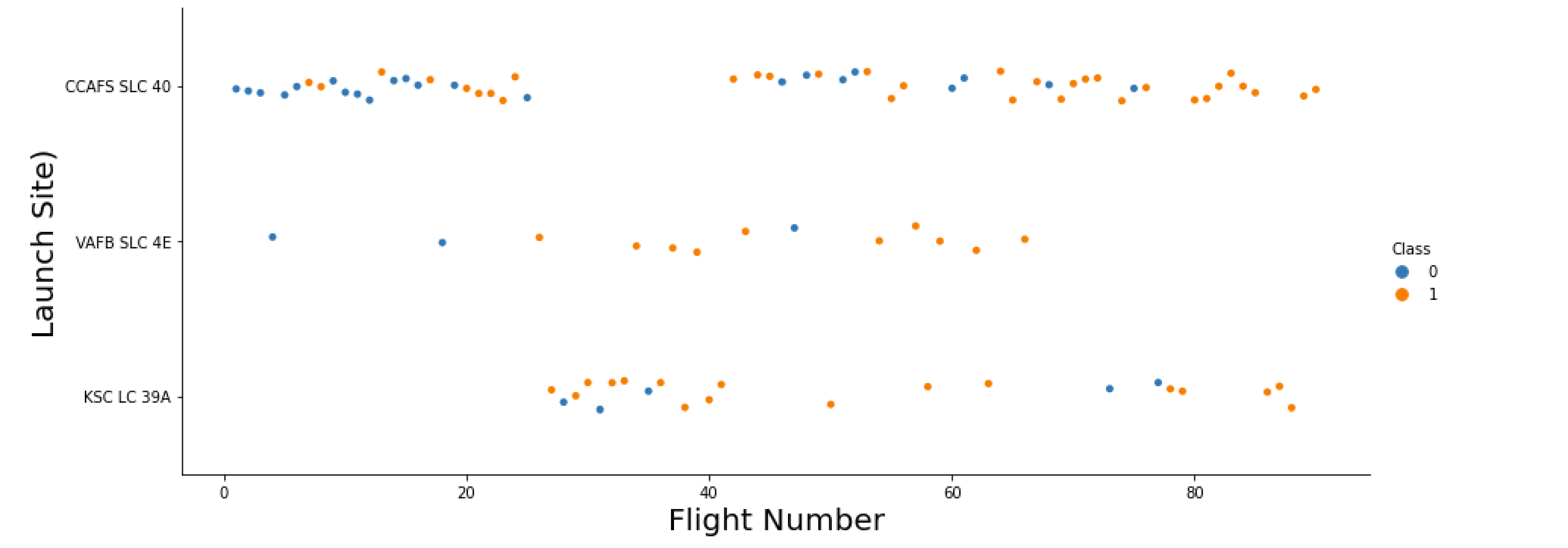
Results

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

Section 2

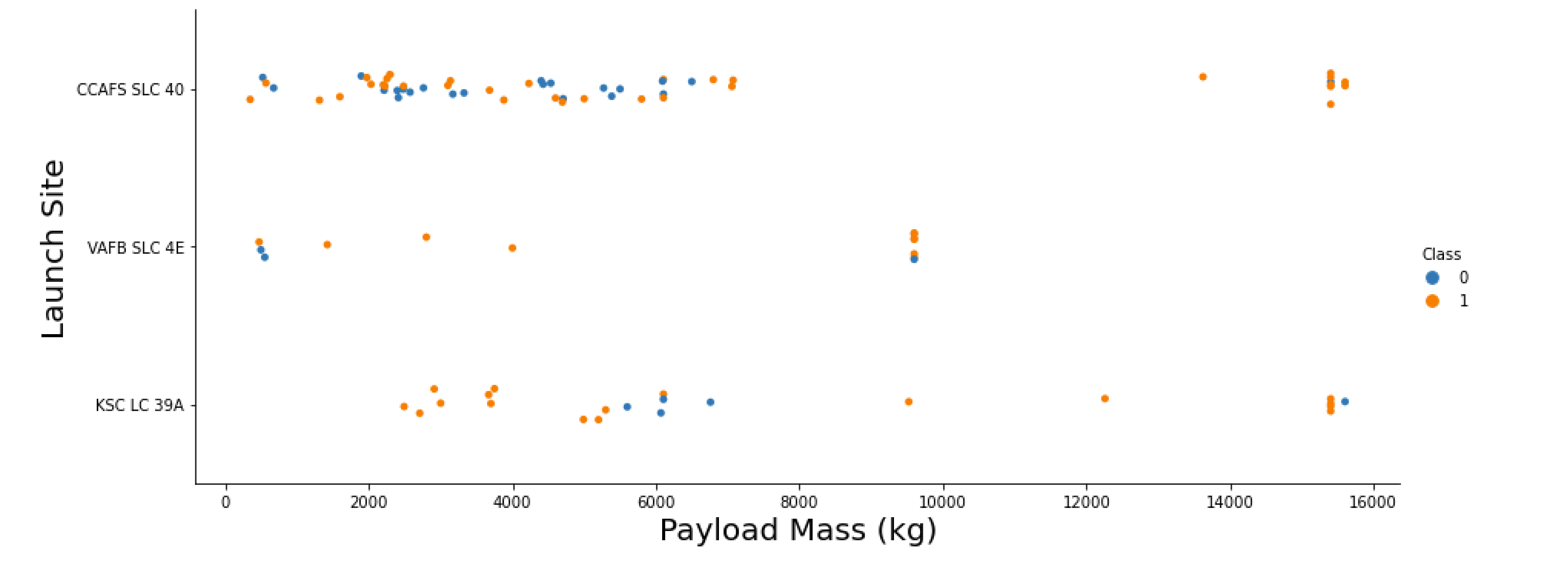
Insights drawn from EDA

Flight Number vs. Launch Site



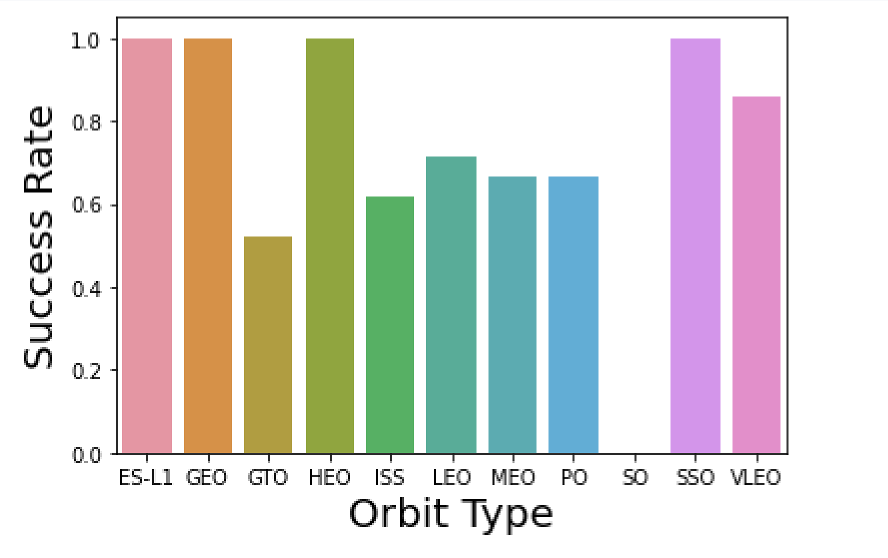
We observe that, for each site, the success rate is increasing.

Payload vs. Launch Site



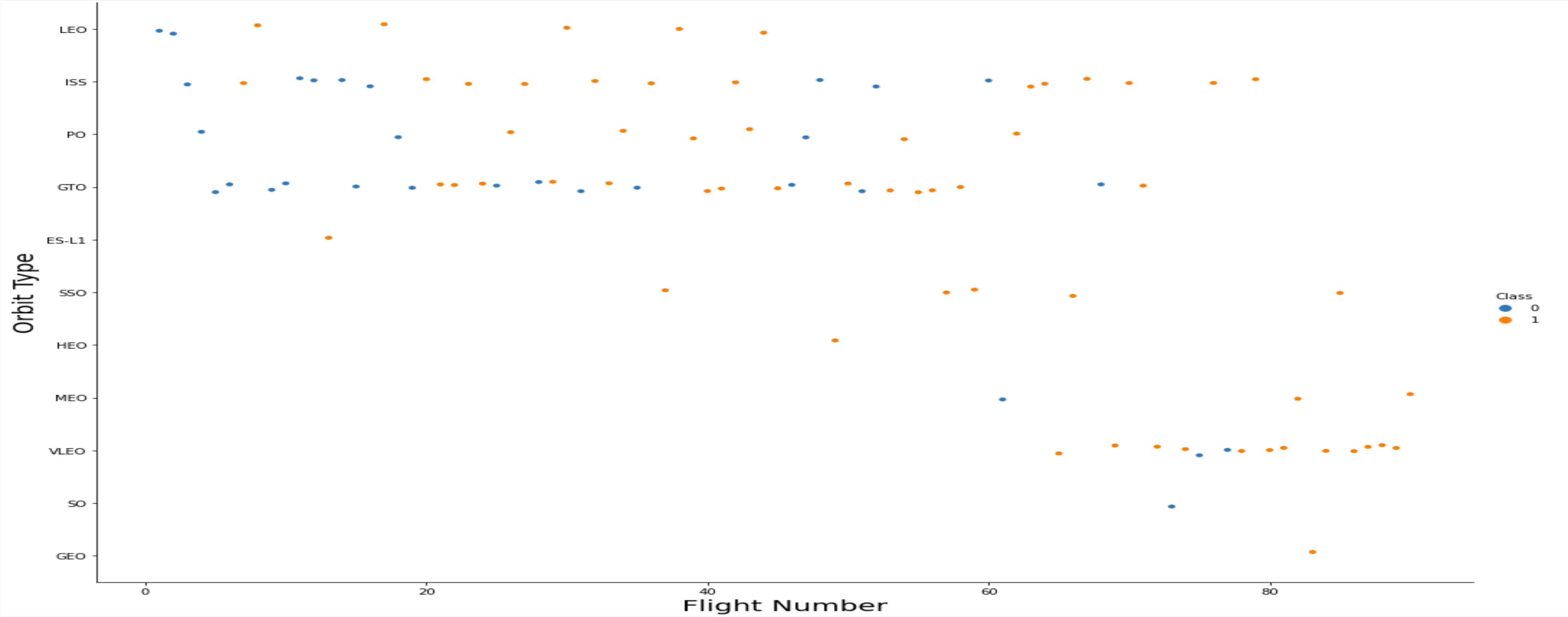
Depending on the launch site, a heavier payload may be a consideration for a successful landing. On the other hand, a too heavy payload can make a landing fail.

Success Rate vs. Orbit Type



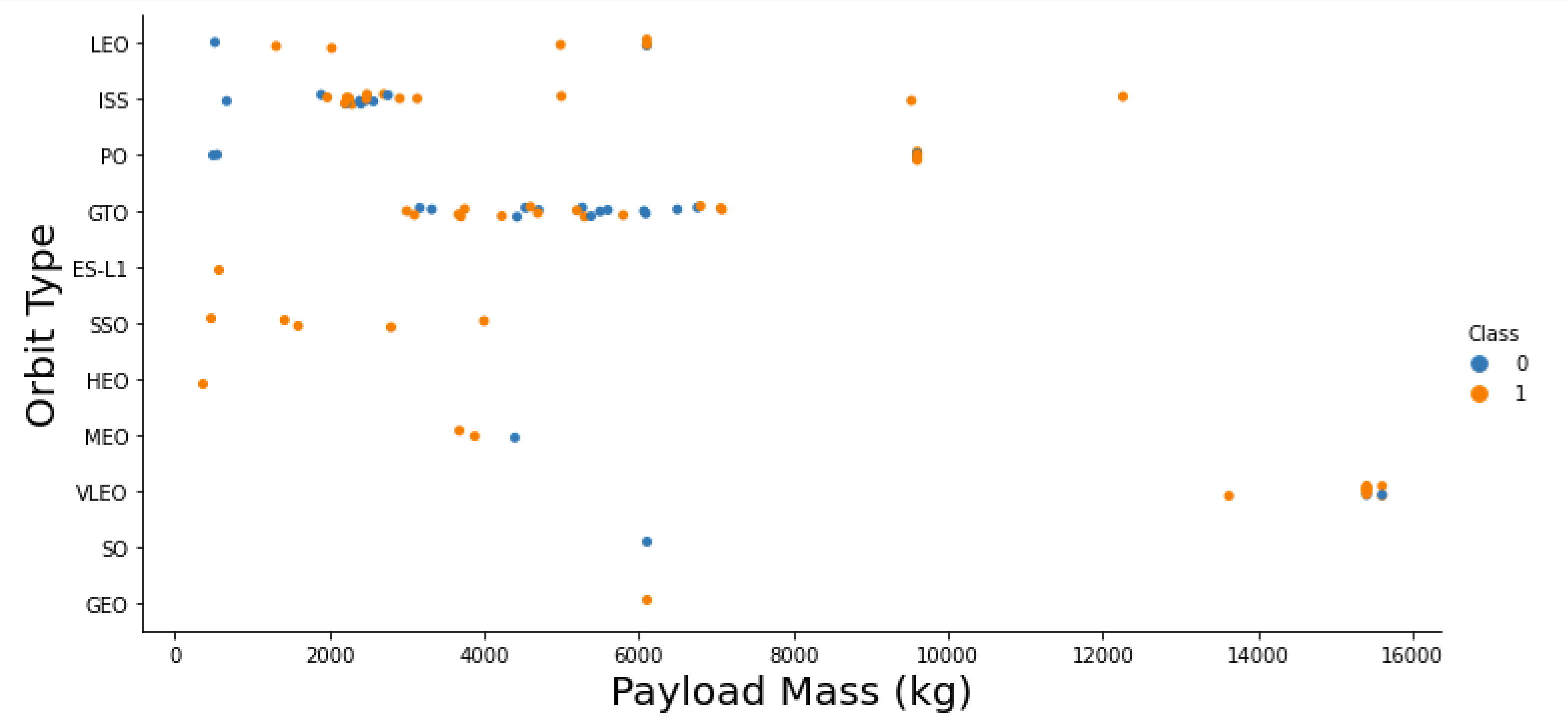
With this plot, we can see success rate for different orbit types. We note that ES-L1, GEO, HEO, SSO have the best success rate.

Flight Number vs. Orbit Type



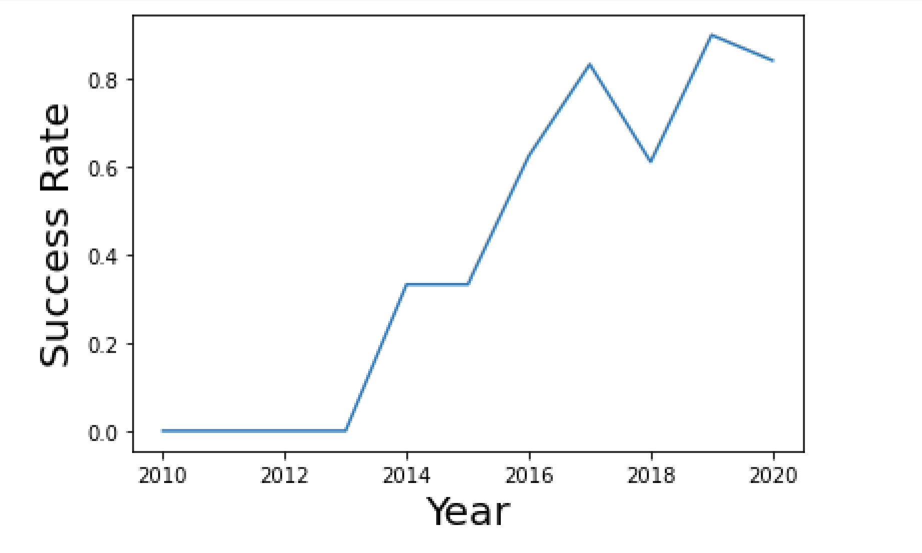
We notice that the success rate increases with the number of flights for the LEO orbit. For some orbits like GTO, there is no relation between the success rate and the number of flights. But we can suppose that the high success rate of some orbits like SSO or HEO is due to the knowledge learned during former launches for other orbits.

Payload vs. Orbit Type



The weight of the payloads can have a great influence on the success rate of the launches in certain orbits. For example, heavier payloads improve the success rate for the LEO orbit. Another finding is that decreasing the payload weight for a GTO orbit improves the success of a launch.

Launch Success Yearly Trend



Since 2013, we can see an increase in the Space X Rocket success rate.

All Launch Site Names

SQL Query

```
conn = sqlite3.connect(':memory:')  # in memory database
df.to_sql(name="spacexdata", con=conn, if_exists="replace")

q = pd.read_sql('select distinct Launch_Site from spacexdata', conn)
q
```

Results

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

Explanation

The use of DISTINCT in the query allows to remove duplicate LAUNCH_SITE.

Launch Site Names Begin with 'CCA'

SQL Query

```
q = pd.read_sql("select * from spacexdata where Launch_Site like 'CCA%' limit 5", conn)
q
```

Results

	Date	Time_(UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS_KG_	Orbit	Customer
	2010-06-04 00:00:00	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX
	2010-12-08 00:00:00	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO
	2012-05-22 00:00:00	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)
	2012-10-08 00:00:00	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)
	2013-03-01 00:00:00	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)

Explanation

The WHERE clause followed by LIKE clause filters launch sites that contain the substring CCA. LIMIT 5 shows 5 records from filtering.

Total Payload Mass

SQL Query

```
q = pd.read_sql("select sum(PAYLOAD_MASS__KG_) from spacexdata where Customer='NASA (CRS)'", conn)
q
```

Results

sum(PAYLOAD_MASS__KG_)
45596

Explanation

This query returns the sum of all payload masses where the customer is NASA (CRS).

Average Payload Mass by F9 v1.1

SQL Query

```
q = pd.read_sql("select avg(PAYLOAD_MASS__KG_) from spacexdata where Booster_Version='F9 v1.1'", conn)
q
```

Results

avg(PAYLOAD_MASS__KG_)
2928.4

Explanation

This query returns the average of all payload masses where the booster version contains the substring F9 v1.1.

First Successful Ground Landing Date

SQL Query

```
q = pd.read_sql("select min(Date) from spacexdata where Landing__Outcome='Success (ground pad)'", conn)
q
```

Results

min(Date)
2015-12-22 00:00:00

Explanation

With this query, we select the oldest successful landing.
The WHERE clause filters dataset in order to keep only records where landing was successful. With the MIN function, we select the record with the oldest date.

Successful Drone Ship Landing with Payload between 4000 and 6000

SQL Query

```
q = pd.read_sql("select distinct Booster_Version from spacexdata where Landing__Outcome='Success (drone ship)' and PAYLOAD_MASS__KG_ between 4000 and 6000", conn)
q
```

Explanation

This query returns the booster version where landing was successful and payload mass is between 4000 and 6000 kg. The WHERE and AND clauses filter the dataset.

Results

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

Total Number of Successful and Failure Mission Outcomes

SQL Query

```
q = pd.read_sql("select substr(Mission_Outcome,1,7) as Mission_Outcome, count(*) from spacexdata  group by 1", conn)
q
```

Explanation

With the first SELECT, we show the subqueries that return results. The first subquery counts the successful mission. The second subquery counts the unsuccessful mission. The WHERE clause followed by LIKE clause filters mission outcome. The COUNT function counts records filtered.

Results

Mission_Outcome	count(*)
Failure	1
Success	100

Boosters Carried Maximum Payload

SQL Query

```
q = pd.read_sql("select distinct Booster_Version from spacexdata where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from spacexdata)", conn)
q
```

Explanation

We used a subquery to filter data by returning only the heaviest payload mass with MAX function. The main query uses subquery results and returns unique booster version (SELECT DISTINCT) with the heaviest payload mass.

Results

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

2015 Launch Records

SQL Query

```
q = pd.read_sql("select distinct Landing__Outcome, Booster_Version, Launch_Site from spacexdata where Landing__Outcome='Failure (drone ship)'", conn)
q
```

Explanation

This query returns the failed landing outcomes in drone ship, their booster versions, and launch site names for, in year 2015

Results

Landing__Outcome	Booster_Version	Launch_Site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1017	VAFB SLC-4E
Failure (drone ship)	F9 FT B1020	CCAFS LC-40
Failure (drone ship)	F9 FT B1024	CCAFS LC-40

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

SQL Query

```
q = pd.read_sql("select Landing__Outcome, count(*) from spacexdata where Date between '2011-06-04' and '2017-03-20' group by Landing__Outcome order by 2 desc", conn)
q
```

Results

Landing__Outcome	count(*)
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Precluded (drone ship)	1

Explanation

This query returns landing outcomes and their Count where the mission was successful and date is between 2010-06-04 and 2017-03-20. The Group By clause groups results by landing outcome and Order By Count Desc shows results in decreasing order

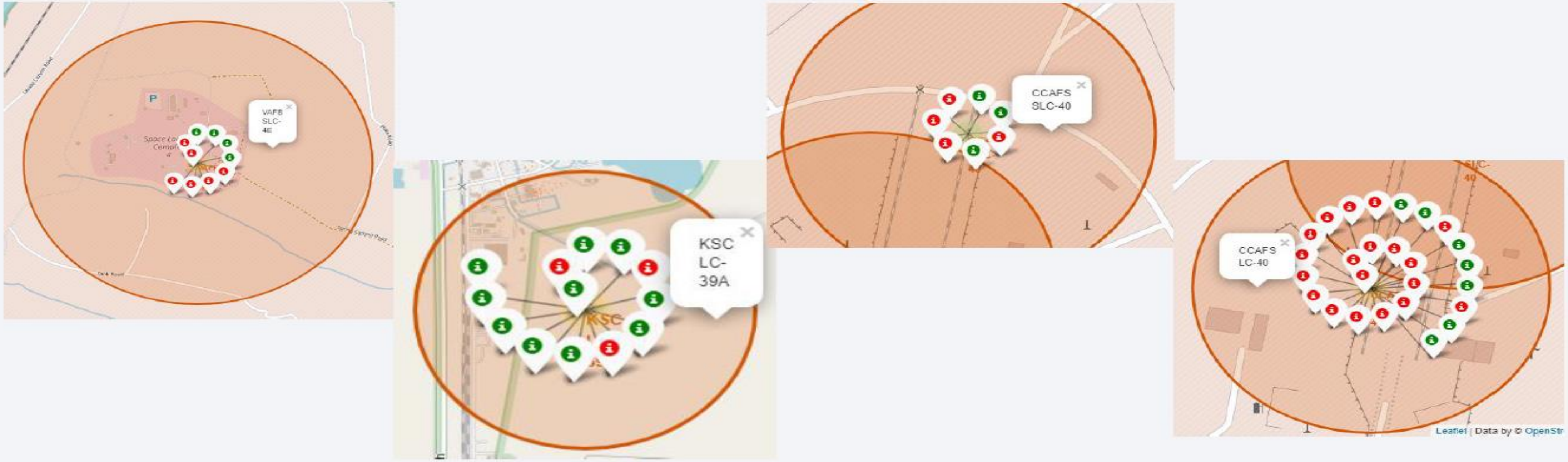
Section 3

Launch Sites Proximities Analysis

Folium map – Ground Stations

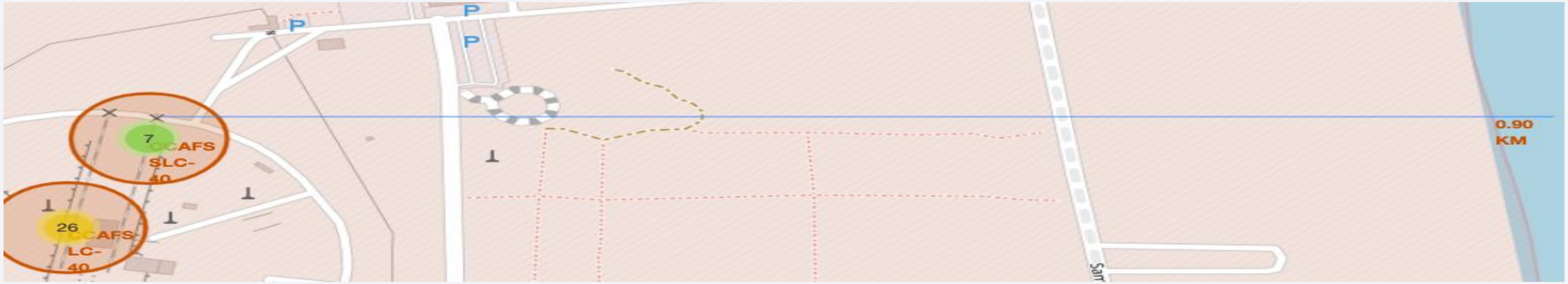
We see that Space X launch sites are located on the coast of the United States

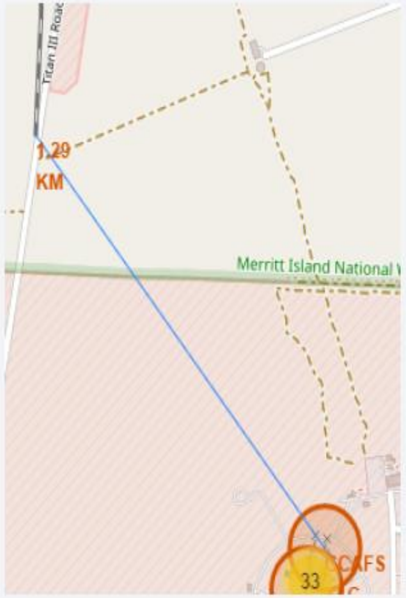
Folium map – Color Labeled Markers



Green marker represents successful launches. Red marker represents unsuccessful launches. We note that KSC LC-39A has a higher launch success rate.

Folium Map – Distances between CCAFS SLC-40 and its proximities



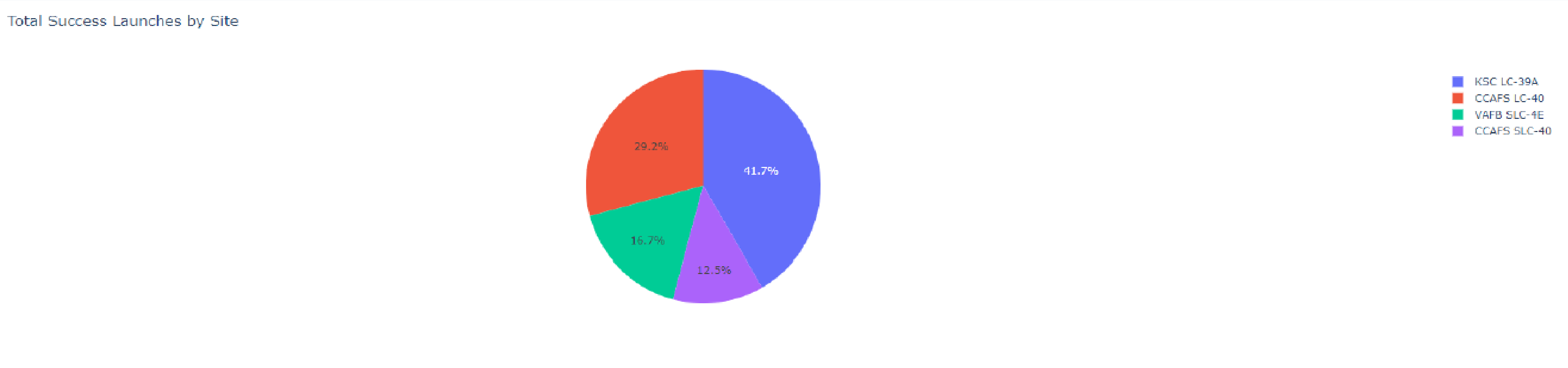


Is CCAFS SLC-40 in close proximity to railways ? Yes
Is CCAFS SLC-40 in close proximity to highways ? Yes
Is CCAFS SLC-40 in close proximity to coastline ? Yes
Do CCAFS SLC-40 keeps certain distance away from cities ? No

Section 4

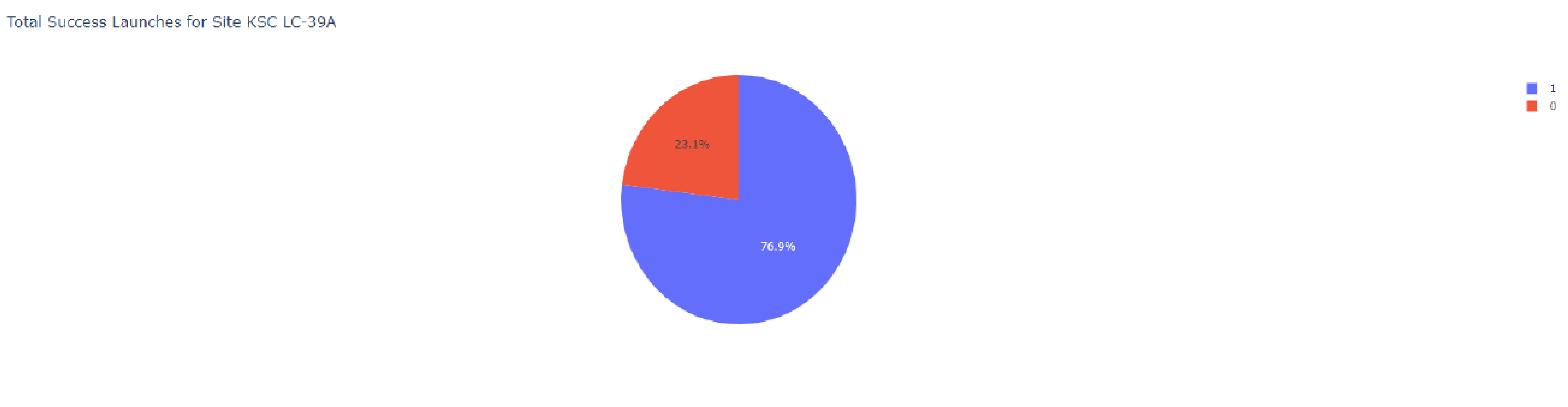
Build a Dashboard with Plotly Dash

Dashboard – Total Success by Site



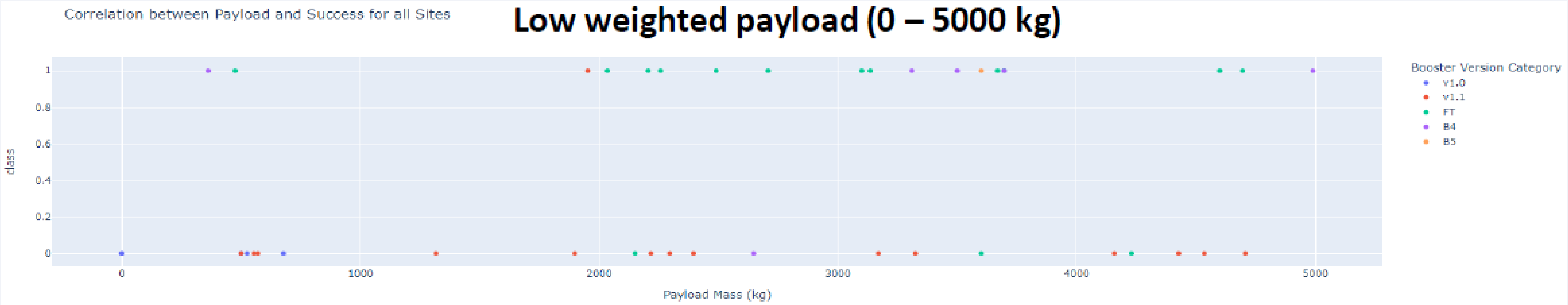
We see that KSC LC-39A has the best success rate of launches.

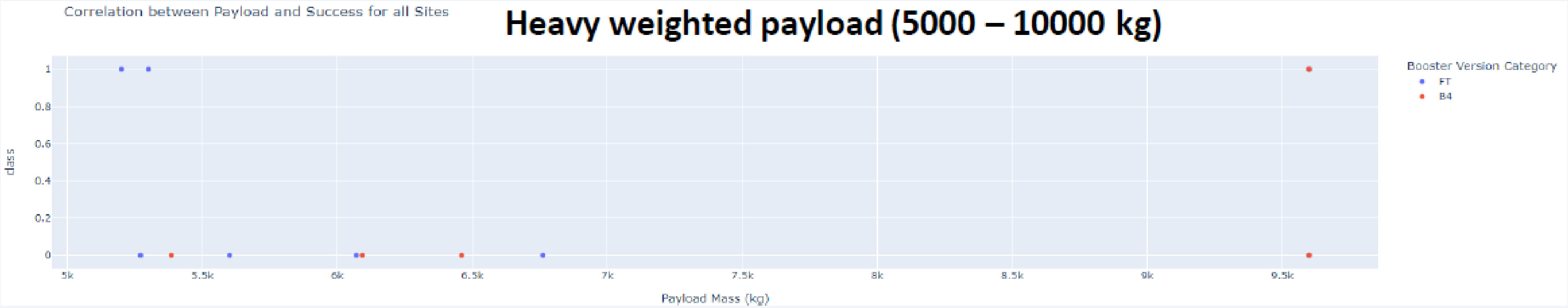
Dashboard – Total Success Launches for Site KSC LC-39A



We see that KSC LC-39A has achieved a 76.9% success rate while getting a 23.1% failure rate.

Dashboard – Payload Mass vs Outcome for all Sites with Different Payload Mass Selected





Low weighted payloads have a better success rate than the heavy weighted payloads.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

```
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
```

```
logreg_cv.score(X_test, Y_test)

0.8333333333333334
```

```
print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
print("accuracy :",svm_cv.best_score_)

tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
accuracy : 0.8482142857142856
```

```
svm_cv.score(X_test, Y_test)

0.8333333333333334
```

```
print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)

tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 16, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_spli
t': 10, 'splitter': 'best'}
accuracy : 0.875
```

```
tree_cv.score(X_test, Y_test)

0.7222222222222222
```

```
print("tuned hpyerparameters :(best parameters) ",knn_cv.best_params_)
print("accuracy :",knn_cv.best_score_)

tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
accuracy : 0.8482142857142858
```

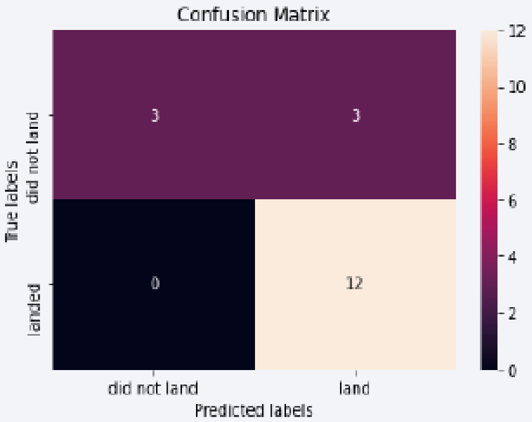
```
knn_cv.score(X_test, Y_test)

0.8333333333333334
```

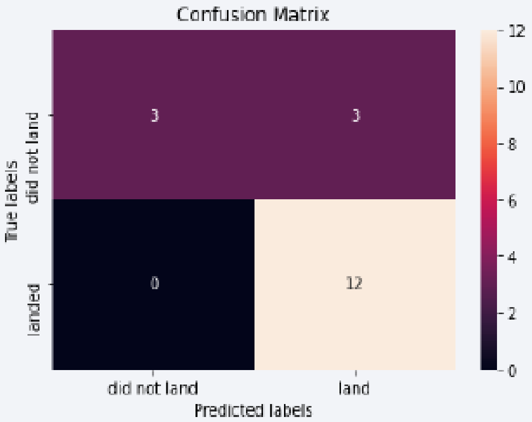
- For accuracy test, all methods performed similar. We can test additional data to decide between each but for this outcome, we would take the Decision Tree. Decision Tree Method is considered to be the best parameter to be used.

Confusion Matrix

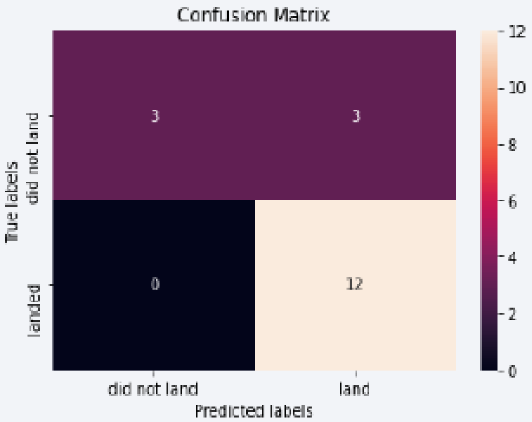
Logistic regression



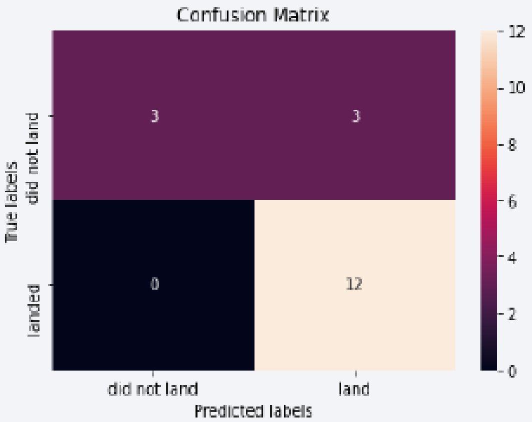
Decision Tree



kNN



SVM



As the test accuracy are all equal, the confusion matrices are also identical. The main problem of these models are false positives.

		Actual values	
		1	0
Predicted values	1	TP	FP
	0	FN	TN

Conclusions

- The success of a mission can be explained by several factors such as the launch site, the Orbit and a number of previous launches. We can have a better prediction with the assumption that there has been additional information regarding launches that allowed to go from a launch failure to a success.
- GEO, HEO, SSO and ES-L1 are the Orbits with the best success rated.
- The Payload mass is a criteria to take into consideration for the success of the mission. Some Orbits require a heavy or a light payload mass, however, it was noted that low weighted payloads perform better than the heavy payloads.
- Decision Tree Method has been considered as the best model, even if the test accuracy between all the models used were similar. The Decision Tree Method has proven to have a better train accuracy.

Appendix

- https://github.com/RLAH84/IBM-Data-Science-Capstone-Project.git

Thank you!