IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Click to add text

Tony Zhang
02-02-2022

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection Using API
  - Data Collection with Web Scarping
  - Data Wrangling
  - Exploratory Data Analysis(EDA) using SQL
  - EDA using Pandas and Matplotlib
  - Interactive Dashboard with Folium
  - Interactive Dashboard with Ploty Dash
  - Predictive Analysis (Classification)
- Summary of all results
  - EDA result
  - Screenshots of Interactive Dashboards
  - Predictive Analytics Result

Introduction

- Project background and context
    - SpaceX advertises Falcon 9 rocket launches on its website, with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. Therefore if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.
- Problems you want to find answers
    - Factors that determine the outcomes of whether or not a rocket will land successfully
    - Through the interactive dashboard, we want to determine the successful rate of each site.
    - What are the optimal condition for a successful landing?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using the SpaceX API
- Perform data wrangling
  - Convert categorical features to numerical
- 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

- Describe how data sets were collected.
    - Data collection was complete with a get request from SpaceX API
    - Next, we parsed the response to a pandas dataframes using .json() and json_normalize()
    - We use the .mean() and .replace() functions to replace np.nan values in the data with the mean value
    - We also uses beafutifulsoup4 to extract Falcon 9 Launch records from Wikipedia
    - The goal was to extract Falcon 9 launch record for future analysis

Data Collection – SpaceX API

- We used get request to collect data from SpaceX API, clean the data by replace np.nan with mean value
- GitHub link: https://github.com/zhang1995/coursera_applied_data_science_capstone/blob/main/data%20import%20api.ipynb

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

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

```
# Use json_normalize meethod to convert the json result into a dataframe
data=response.json()
data=pd.json_normalize(data)
```

```
# Calculate the mean value of PayloadMass column
mean = data_falcon9[['PayloadMass']].mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass']=data_falcon9['PayloadMass'].replace(np.nan, mean[0])
data_falcon9
data_falcon9.isnull().sum()
```

Data Collection - Scraping

- Creating BeautifulSoup object from Falcon 9 Wikipedia page
- Creating pandas dataframe object from HTML table element
- GitHub link: https://github.com/zhang1995/coursera_applied_data_science_capstone/blob/main/web%20scarping.ipynb

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

```
# use requests.get() method with the provided static_url
# assign the response to a object
falcon_9=requests.get(static_url)
```

Create a BeautifulSoup object from the HTML response

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(falcon_9.content, 'html.parser')
```

```
# 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')
```

Starting from the third table is our target table contains the actual launch records.

```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
```

```
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 column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
column_names = []
th_elements = first_launch_table.find_all('th')
for element in th_elements:
    name = extract_column_from_header(element)
    if name is not None and len(name) > 0:
        column_names.append(name)
```

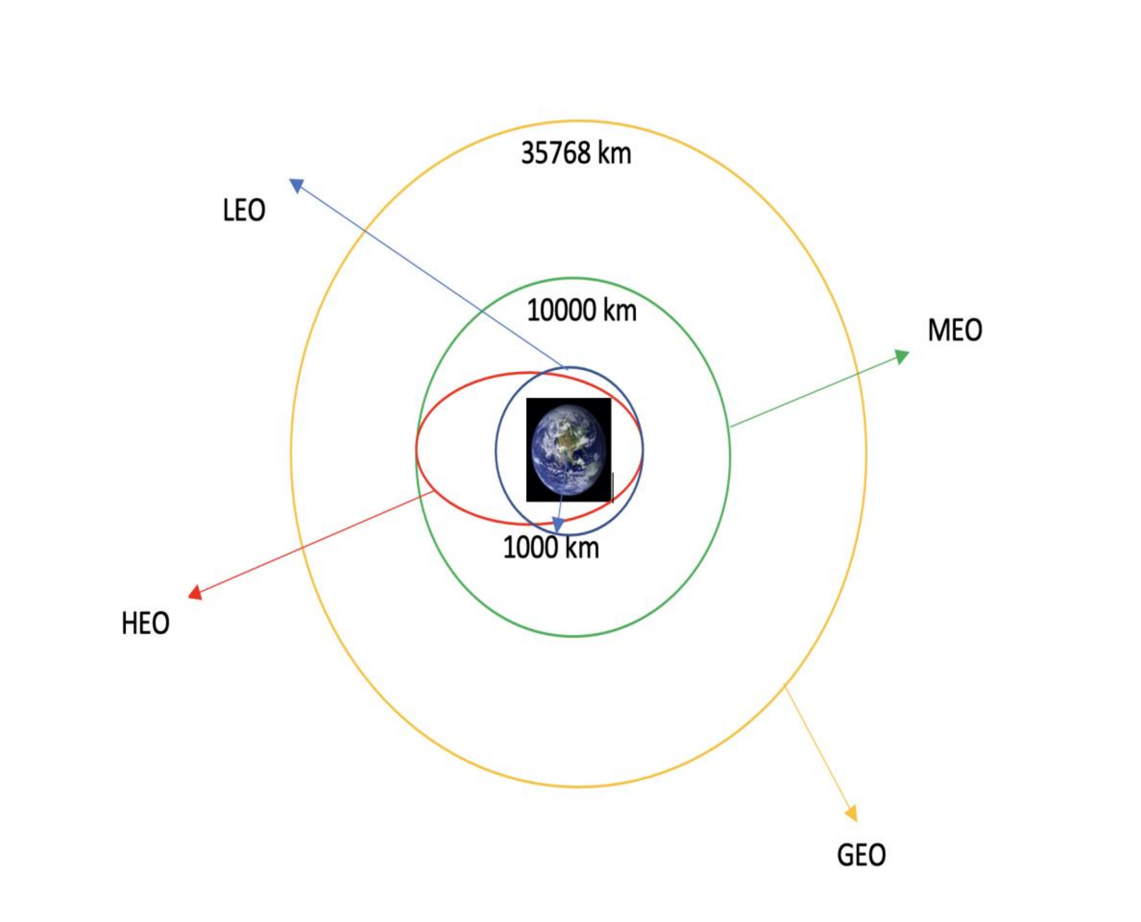
Check the extracted column names

```
print(column_names)
```

```
['Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome']
```

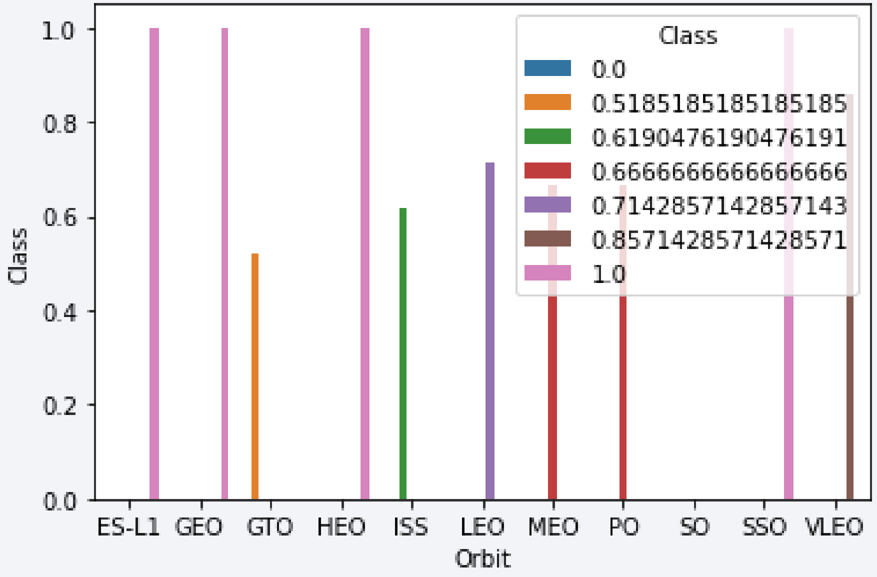
Data Wrangling

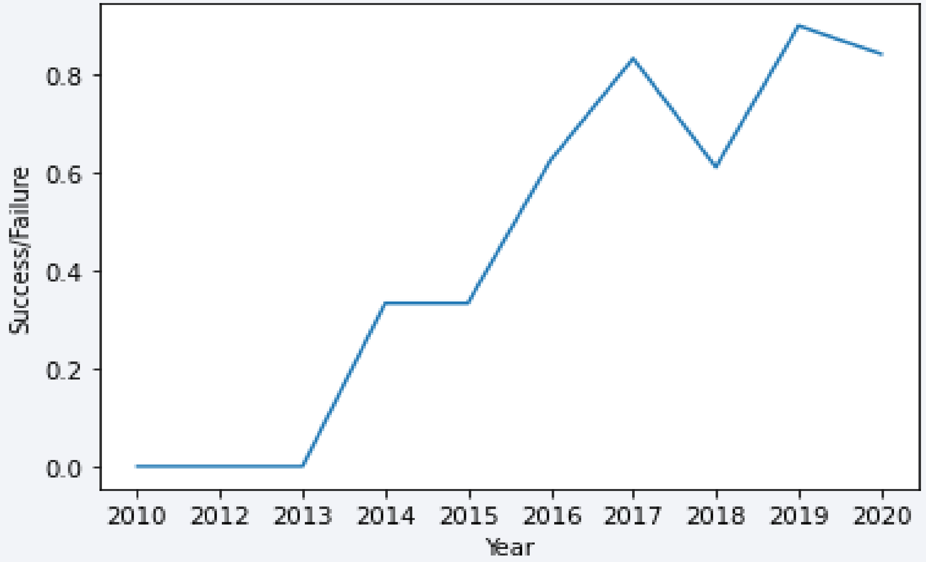
- We performs Exploratory Data Analysis to find some patterns in the data and determine the label for training supervised models
- We calculate number of launches on each site, number and occurrence of each orbit, number and occurrence of mission outcome per orbit type.
- We create a landing outcome label from outcome column
- GitHub link: https://github.com/zhang1995/coursera_applied_data_science_capstone/blob/main/Data%20Wrangling.ipynb



EDA with Data Visualization

- We visualizing the relationship between flight number and launch site, payload and launch site, success rate of each orbit type, flight number and orbit type, the launch success yearly trend
- GitHub link: https://github.com/zhang1995/coursera_applied_data_science_capstone/blob/main/EDA%20data%20visualization%20.ipynb





EDA with SQL

- we wrote SQL queries to gain insights on following topics:
  - Names of the unique launch sites in the space mission
  - Total payload carried by boosters launched by NASA(CRS)
  - Average payload mass carried by booster version F9 v1.1
  - Total number of successful and failure mission outcomes
  - Failed landing outcome in drone ship, booster version and launch site names.
  - Rank the count of landing outcomes in descending order
- GitHub link: https://github.com/zhang1995/coursera_applied_data_science_capstone/blob/main/EDA%20with%20SQL.ipynb

Build an Interactive Map with Folium

- We marked all launch sites on the map
  - Are all launch sites in very close proximity to the coast
  - Are all launch sites in proximity to the Equator line
- We marked the success/failed launches for each site
  - which sites have high success rates
- We marked distance between a launch site to its proximities
  - Are launch sites near railways, highways and coastlines.
  - Do launch sites keep certain distance away from cities.
- GitHub link: https://github.com/zhang1995/coursera_applied_data_science_capstone/blob/main/Interactive%20Visual%20Analytics%20with%20Folium.ipynb

Build a Dashboard with Plotly Dash

- We built pie charts showing the total launches by sits
- We plotted scatter graph showing the relationship with Outcome and payload Mass(Kg) for the different booster version.
- GitHub
  Link: https://github.com/zhang1995/coursera_applied_data_science_capstone/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- We loaded the data and standardize the data, then we split the data into train and test set
- Next, we use GridSearchCV on different machine learning models to find the best parameter for each model.
- We use accuracy to determine that Decision tree is the best model for the task
- GitHub Link: https://github.com/zhang1995/coursera_applied_data_science_capstone/blob/main/Prediction.ipynb

Results

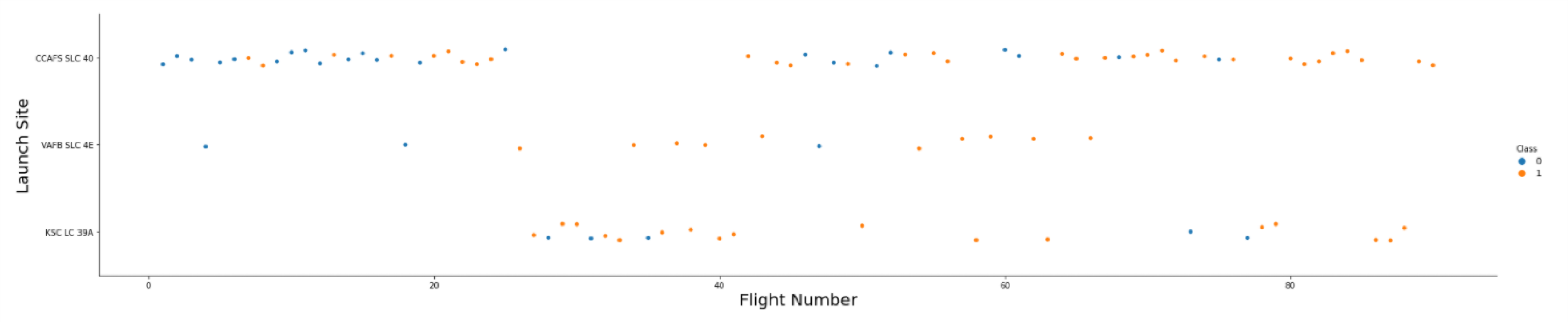
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

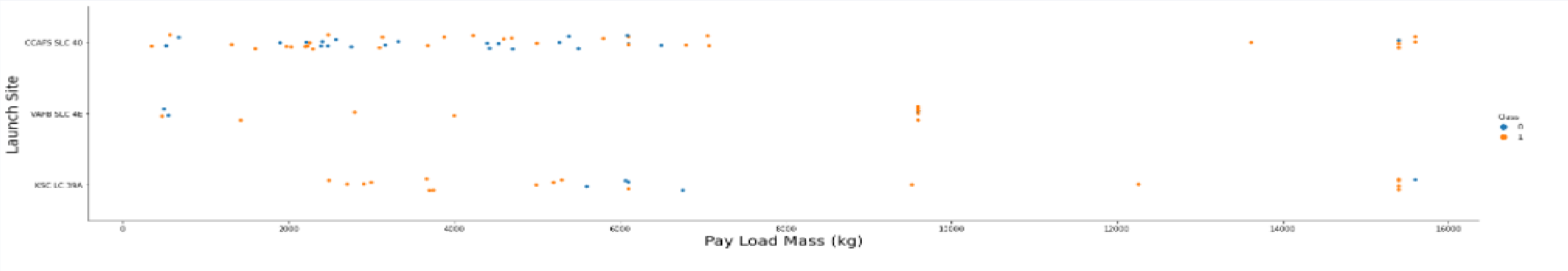
Flight Number vs. Launch Site

- We see most of the rocket launched at site: CCAFS SLC 40



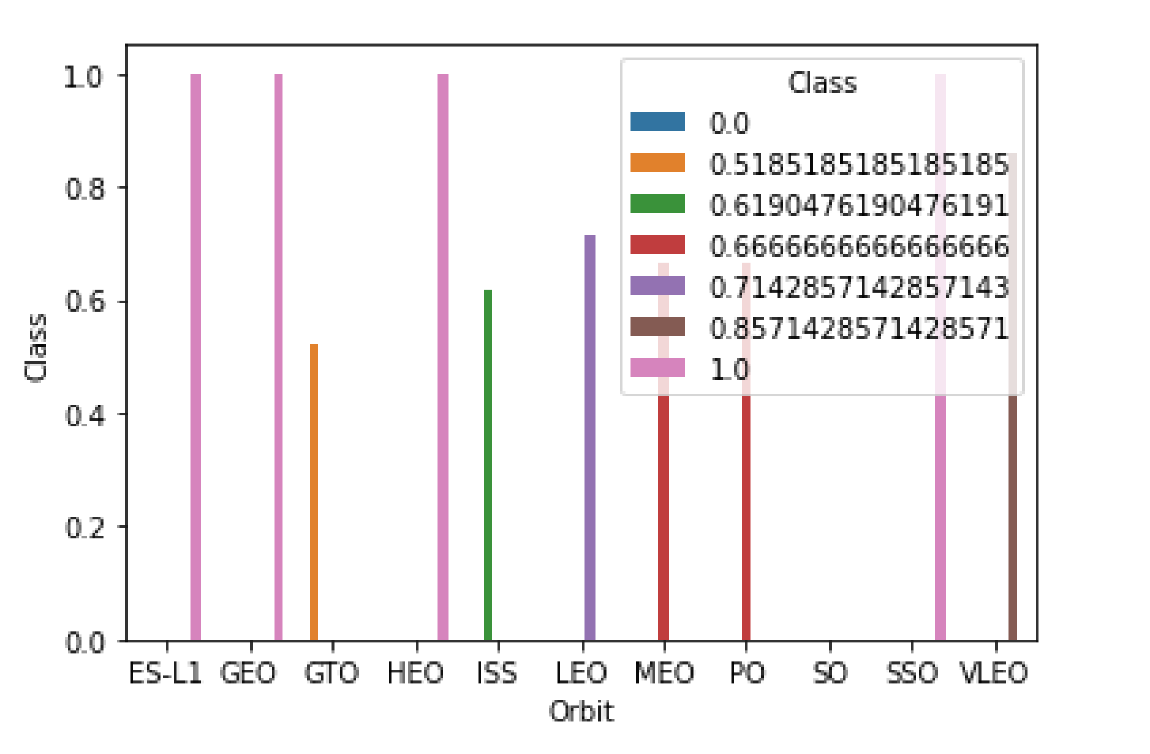
Payload vs. Launch Site

- For VAFB-SLC launch site there are no rockets launched for heavy payload mass(>10000)



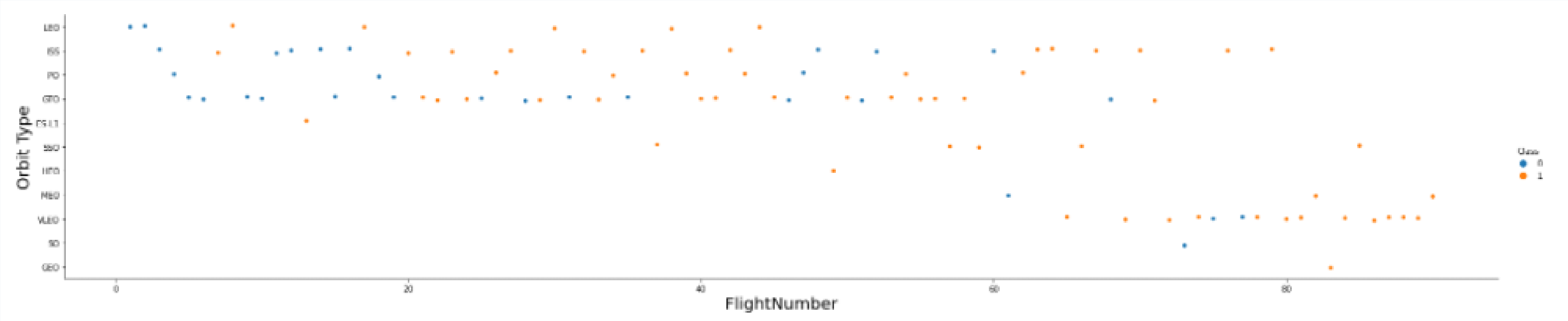
Success Rate vs. Orbit Type

- Base on the plot, ES-L1, GEO, HEO, SSO and VLEO has the most success rate



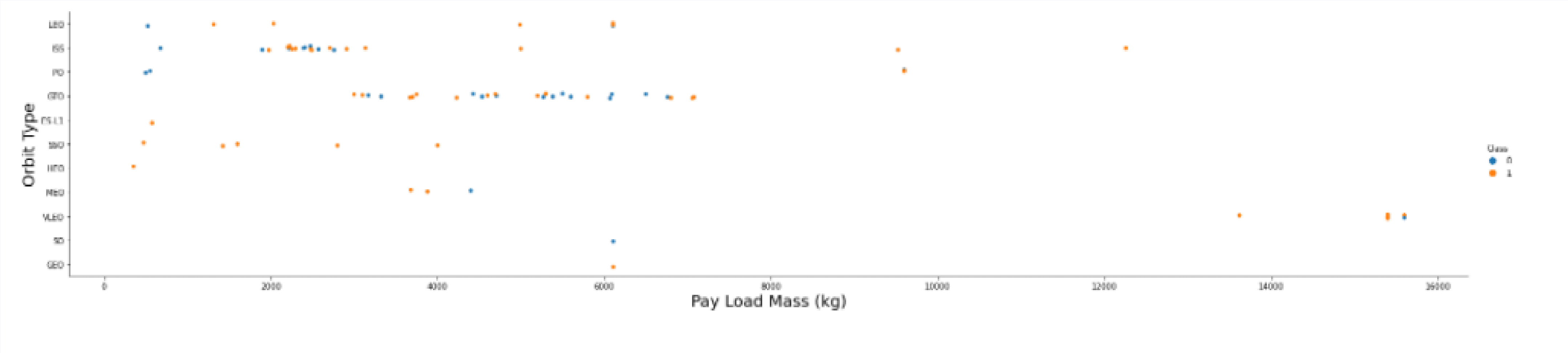
Flight Number vs. Orbit Type

- For LEO orbit, success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit



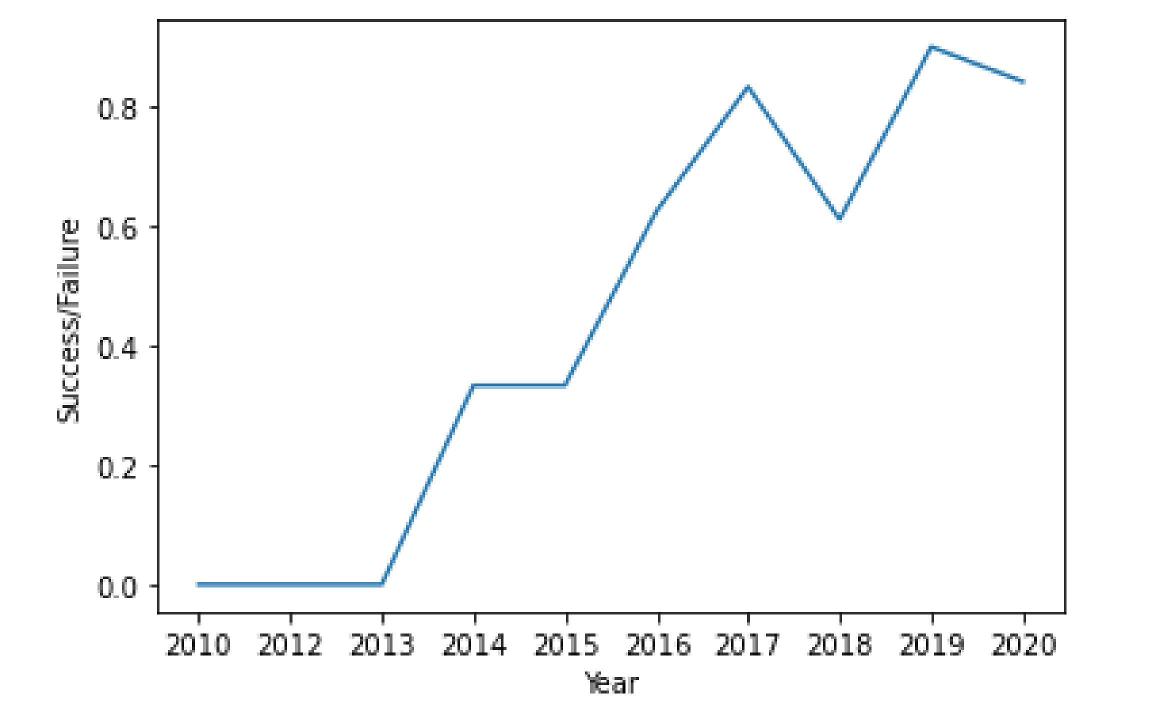
Payload vs. Orbit Type

- with heavy payloads, the successful landing are more for PO, LEO and ISS orbits.



Launch Success Yearly Trend

- This plot shows success rate kept increasing since 2013



All Launch Site Names

- Using the DISTINCT to show unique launch sites

```
In [6]: %sql select DISTINCT launch_site from SPACEXTBL;

 * ibm_db_sa://ccm41386:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb
Done.

Out[6]:
```

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

Launch Site Names Begin with 'CCA'

- We use WHERE and LIMIT clauses to display launch sites begin with CCA

```
%sql select * from SPACEXTBL where launch_site like 'CCA%' limit 5
```

```
 * ibm_db_sa://ccm41386:***@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

- We calculate the total mass carried for NASA using below query

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

```
%sql Select sum(payload_mass__kg_) from SPACEXTBL where customer = 'NASA (CRS)'
```

```
 * ibm_db_sa://ccm41386:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb
Done.
```

1
45596

Average Payload Mass by F9 v1.1

- We calculate the average payload mass carried by booster version F9 v1.1 with AVG

```
%sql select AVG(payload_mass__kg_) from SPACEXTBL where booster_version like 'F9 v1.1%'
```

```
 * ibm_db_sa://ccm41386:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb
Done.
```

1
2534

First Successful Ground Landing Date

- We find the dates of the first successful landing outcome on ground pad using "min", and the first successful landing on ground pad was 2015-12-22

```
%sql select min(DATE) from SPACEXTBL where landing__outcome ='Success (ground pad)'
```

```
 * ibm_db_sa://ccm41386:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb
Done.
```

1
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- We used below query to find boosters that successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

```
%sql select booster_version, payload_mass__kg_ from SPACEXTBL where landing__outcome ='Success (drone ship)' and payload_mass__k
g_ between 4000 and 6000
```

```
 * ibm_db_sa://ccm41386:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb
Done.
```

booster_version	payload_mass__kg_
F9 FT B1022	4696
F9 FT B1026	4600
F9 FT B1021.2	5300
F9 FT B1031.2	5200

Total Number of Successful and Failure Mission Outcomes

- We used GROUP BY and COUNT to calculate the total number of successful and failure mission outcomes

```
%sql select mission_outcome, count(*) from SPACEXTBL group by mission_outcome
```

```
 * ibm_db_sa://ccm41386:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb
Done.
```

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

Boosters Carried Maximum Payload

- We use a subquery to determine the maximum payload mass

```
%sql select booster_version, payload_mass__kg_ from SPACEXTBL where payload_mass__kg_ = (select max(payload_mass__kg_) from SPAC
EXTBL)
```

```
 * ibm_db_sa://ccm41386:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb
Done.
```

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

2015 Launch Records

- We used a combinations of the WHERE clause, LIKE, AND, and BETWEEN conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015

```
%sql select booster_version from SPACEXTBL where landing__outcome = 'Failure (drone ship)' and DATE between '2015-01-01' and '20
15-12-31'
```

```
 * ibm_db_sa://ccm41386:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb
Done.
```

booster_version
F9 v1.1 B1012
F9 v1.1 B1015

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

- We use WHERE DATE to filter landing outcomes BETWEEN 2010-06-04 and 2017-03-20. We then uses Group BY, COUNT and ORDER BY to rank the number.

```
%sql select landing__outcome, count(*) as count
from SPACEXTBL
where DATE between '2010-06-04' and '2017-03-20'
group by landing__outcome
order by count DESC
```

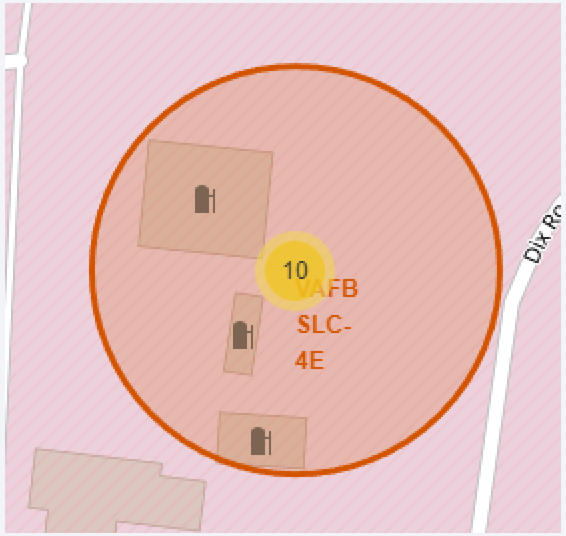
landing__outcome	COUNT
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

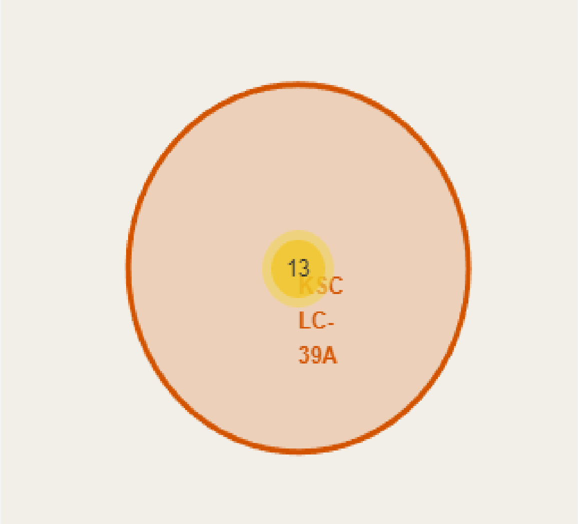
Section 3

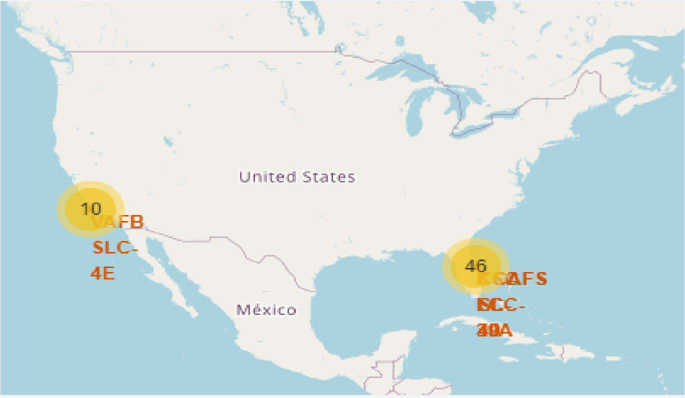
Launch Sites Proximities Analysis

All Launch Sites on a Map

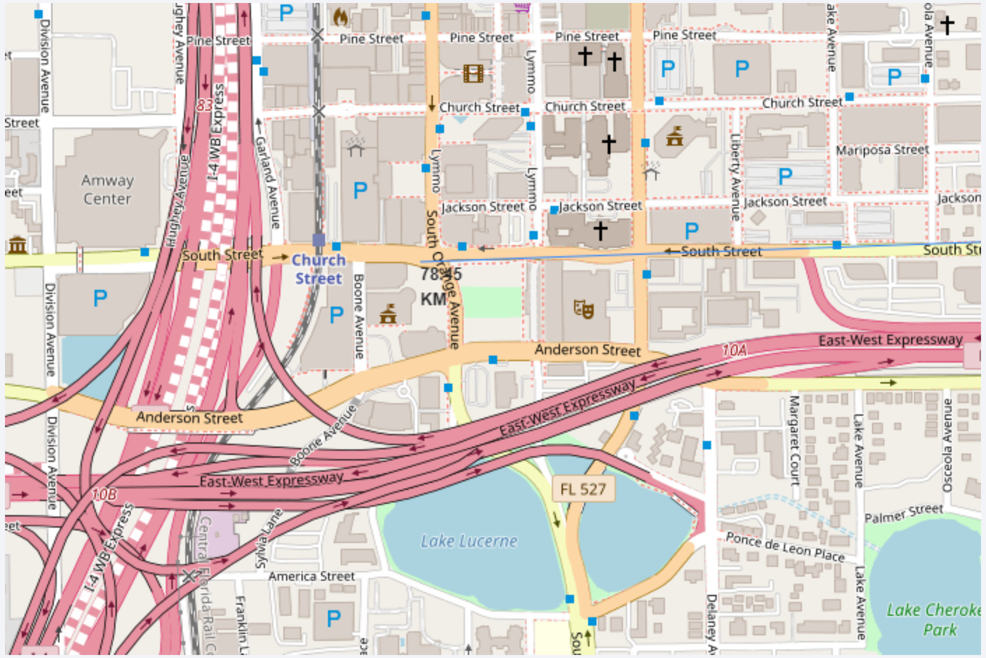
Success/Failed Launches for Each Site







Launch Site Distance to Landmarks

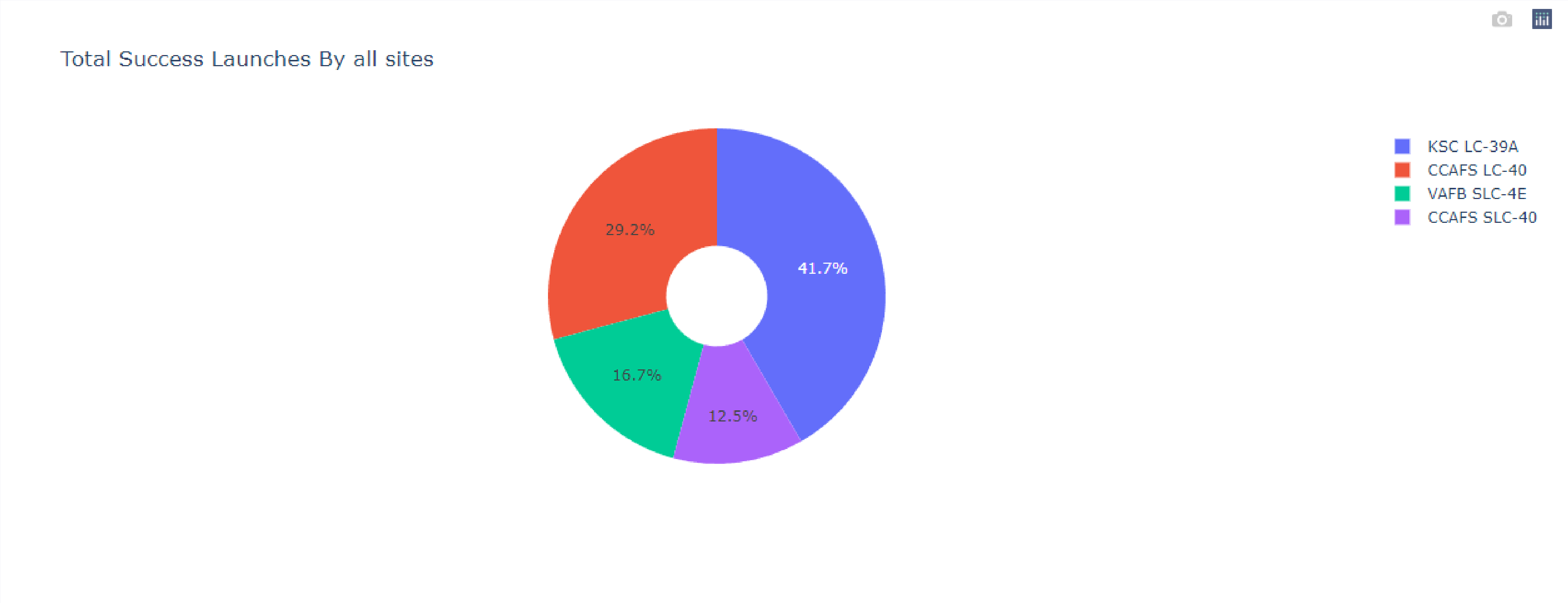


Section 4

Build a Dashboard with Plotly Dash

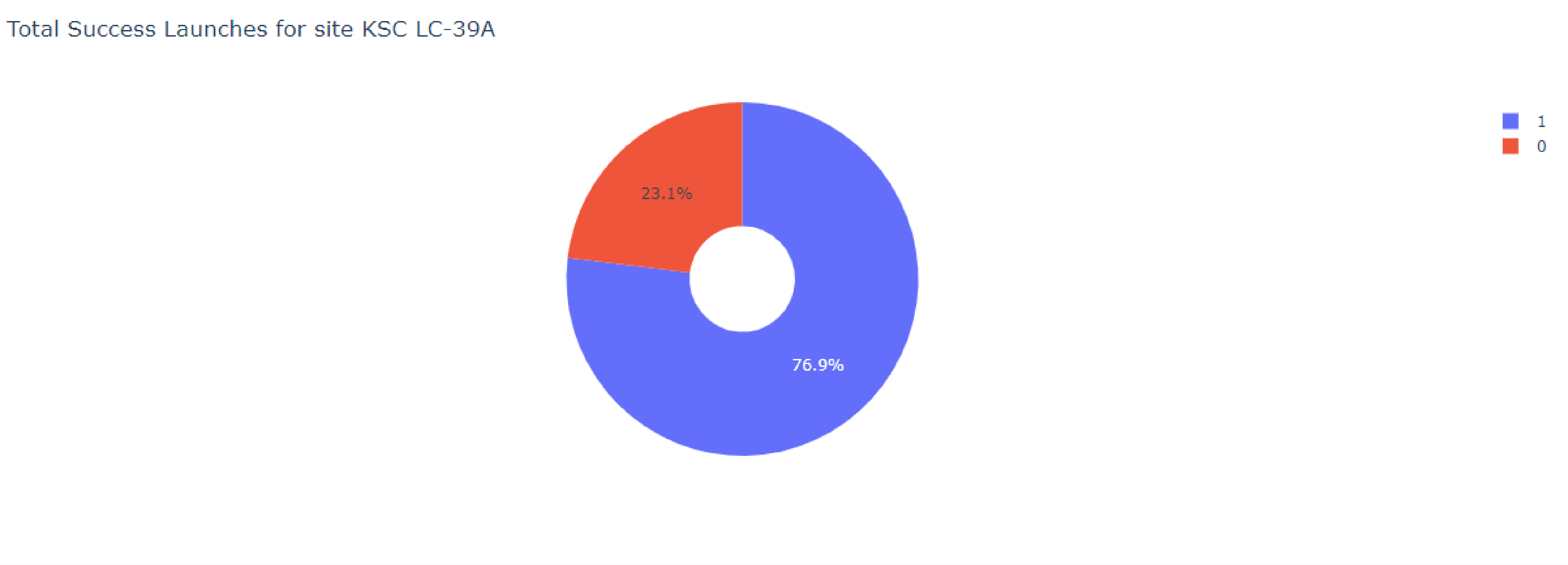
Success percentage by each launch site

- This pie chart shows KSC LC-39A had the most successful launches

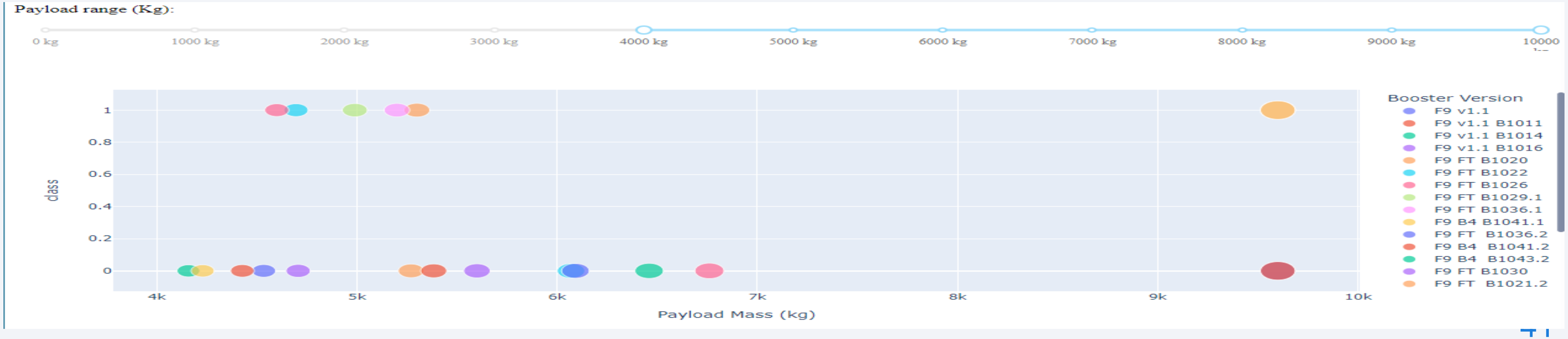


<Dashboard Screenshot 2>

KSC LC-39A had a 76.9 Success rate and 23.1 failure rate



Payload vs Launch Outcome for all sites



Section 5

Predictive Analysis (Classification)

Classification Accuracy

- DecisionTree has the highest classification accuracy

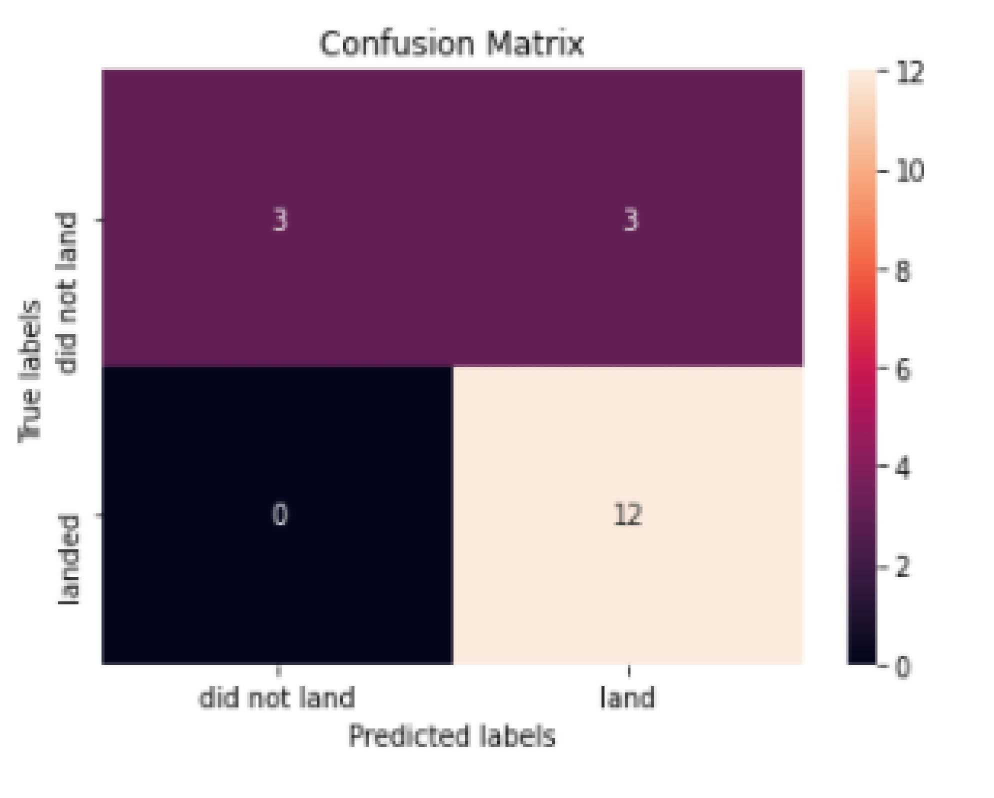
```
models = {'KNeighbors':knn_cv.best_score_,
          'DecisionTree':tree_cv.best_score_,
          'LogisticRegression':logreg_cv.best_score_,
          'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

```
Best model is DecisionTree with a score of 0.875
Best params is : {'criterion': 'gini', 'max_depth': 2, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 10, 'splitter': 'random'}
```

Confusion Matrix

- The confusion matrix for the decision tree models shows it can distinguish between different outcomes
- There is a problem with this models because of the false positives



Conclusions

- We can conclude that:
  - The larger the flight amount at a launch site, the greater the success rate at a launch site.
  - Launch success rate started to increase in 2013 till 2020. • Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
  - KSC LC-39A had the most successful launches of any sites.
  - Decision tree classifier is the best model for this task

Thank you!