IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Farhan Zafar Khan
10 October, 2022

Outline

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

Executive Summary

- Summary of methodologies
    - Data collection using SpaceX APIs
    - Data wrangling using python
    - Exploratory data analysis (EDA) using visualization and SQL
    - Perform interactive visual analytics using Folium and Plotly Dash
    - Perform predictive analysis using classification models
- Summary of all results
    - EDA results
    - Interactive analytics
    - Predictive analytic results

Introduction

- Project background and context
  - Space X 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 Space X can reuse the first stage. Using machine learning and data analysis we can predict whether a stage one rocket will land successfully. The goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.
- Problems you want to find answers
  - What factors determine if the rocket will land successfully?
  - The different features that determine the success of a landing.
  - The accuracy of a prediction for successful landing

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using web scraping and SPACEX API
- Perform data wrangling
  - One-hot encoding was applied to categorical features
- 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

- Data was collected using GET request from python 'requests' library, to the SpaceX API.
- The response content was converted to Json using .json(), and then into a pandas dataframe using .json_normalize().
- The data was cleaned, checked for missing values and fill in missing values where necessary.
- Web scraping from Wikipedia using BeautifulSoup for Falcon 9 launch records was used to extract the launch records as HTML table, then converted to a pandas dataframe for future analysis
- The Dataframe was saved as csv for future use

Data Collection – SpaceX API

- Present your data collection with SpaceX REST calls using key phrases and flowcharts

- [IBMDSProfessionalCertificate_Capstone/jupyter-labs-spacex-data-collection-api.ipynb at main · farhanzafark/IBMDSProfessionalCertificate_Capstone (github.com)]

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

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

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api
```

We should see that the request was successfull with the 200 status response code

```
response.status_code
```

```
200
```

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

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

data = pd.json_normalize(responseJson)
```

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = dfLaunch[dfLaunch['BoosterVersion']!='Falcon 1']
```

Now that we have removed some values we should reset the FlgihtNumber column

```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

Data Collection - Scraping

- Web scraping of the SPACEX Falcon 9 launch wikipedia page was done using BeautifulSoup.
- HTML tags <table>, <tr> were used to find the relevant data
- The scraped data was converted to data frame
- [IBMDSProfessionalCertificate_Capstone/jupyter-labs-webscraping.ipynb at main · farhanzafark/IBMDSProfessionalCertificate_Capstone (github.com)]

```
# use requests.get() method with the provided static_url
# assign the response to a object
response = 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(response.text,'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]
print(first_launch_table)
```

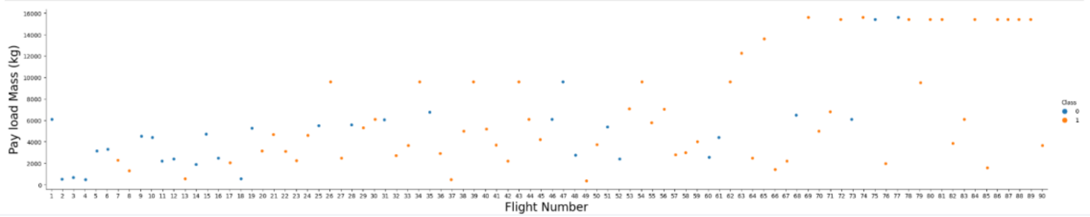
```
launch_dict= dict.fromkeys(column_names)

# 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']=[]
```

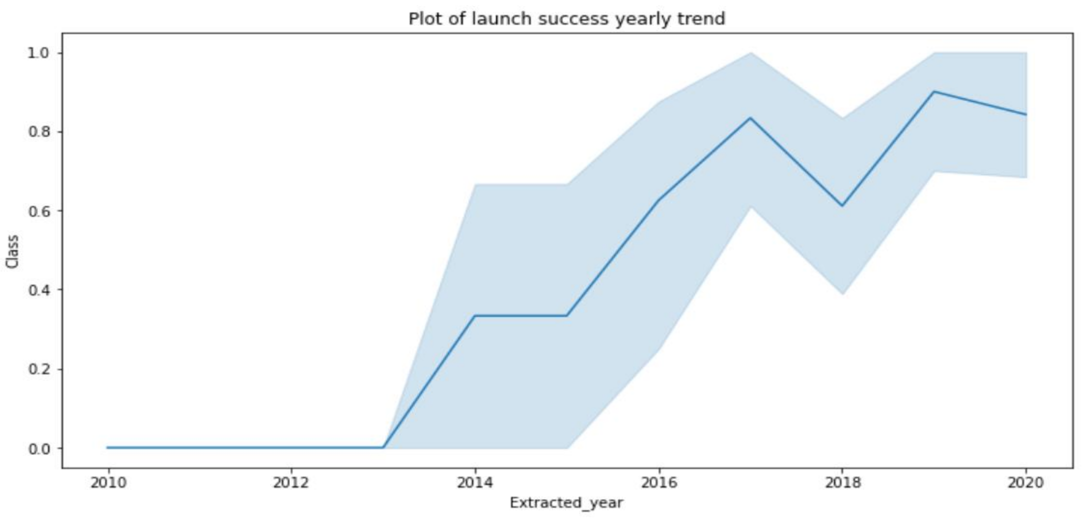
Data Wrangling

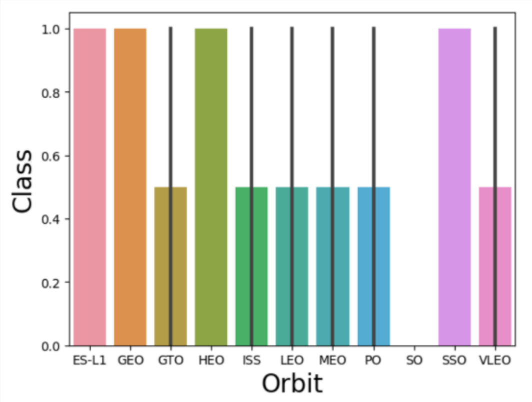
- The data set was loaded into a data frame and null values were counted for each column
- The main columns to focus were the launch site, type of launch and launch outcome
- An output data column classifying the landing outcomes as good or bad was created and values were assigned to it
- IBMDSProfessionalCertificate_Capstone/labs-jupyter-spacex-Data wrangling.ipynb at main · farhanzafark/IBMDSProfessionalCertificate_Capstone (github.com)

EDA with Data Visualization

- Visualization of different parameters (payload/lauch site, flight number/launch site) was carried out
- Success rate of launch sites and number of successful launches were also displayed
- One hot encoding was used to categorize different variables
- [IBMDSProfessionalCertificate_Capstone/jupyter_labs_eda_dataviz.ipynb at main · farhanzafark/IBMDSProfessionalCertificate_Capstone (github.com)]







EDA with SQL

- The dataframe was loaded into an SQL table
- Distinct launch sites were selected using a query. Further, records for a certain launch site were also fetched
- Sum and average of payload mass were calculated. Booster version was selected for payloads of a certain range of mass
- Records were selected from the table based on a range of provided dates
- IBMDSProfessionalCertificate_Capstone/jupyter-labs-eda-sql-coursera_sqllite (1).ipynb at main · farhanzafark/IBMDSProfessionalCertificate_Capstone (github.com)

Build an Interactive Map with Folium

- The launch sites were marked on the Folium map with circles, and a marker was added with additional information for each site
- Success of launches at each site was marked using a marker cluster, with red for failed launch and green for successful launch
- WThe distances between a launch site to its proximities was calculated using mouse cursor position and a custom function
- IBMDSProfessionalCertificate_Capstone/lab_jupyter_launch_site_location (1).ipynb at main · farhanzafark/IBMDSProfessionalCertificate_Capstone (github.com)

Build a Dashboard with Plotly Dash

- Plotly dash was used to build an interactive dashboard
- Pie charts were used to showing the succesful and failed launches by a certain site
- Scatter graph was used to show the relationship with Outcome and Payload Mass (Kg) for the different booster version.

- IBMDSProfessionalCertificate_Capstone/app.py at main · farhanzafark/IBMDSProfessionalCertificate_Capstone (github.com)

Predictive Analysis (Classification)

- We loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- We built different machine learning models and tune different hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- The best performing ML model was determined using
- IBMDSProfessionalCertificate_Capstone/SpaceX_Machine_Learning Prediction_Part_5 (1).ipynb at main · farhanzafark/IBMDSProfessionalCertificate_Capstone (github.com)

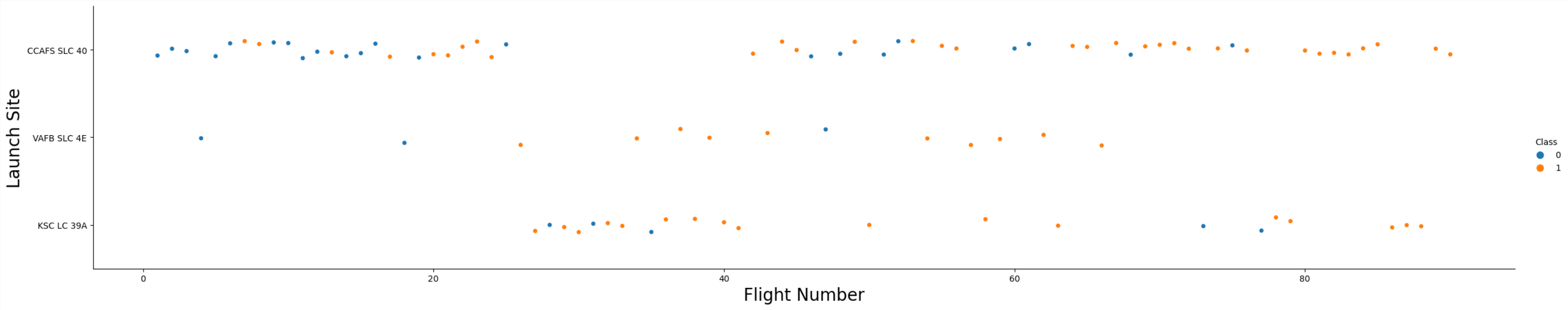
Results

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

Section 2

Insights drawn from EDA

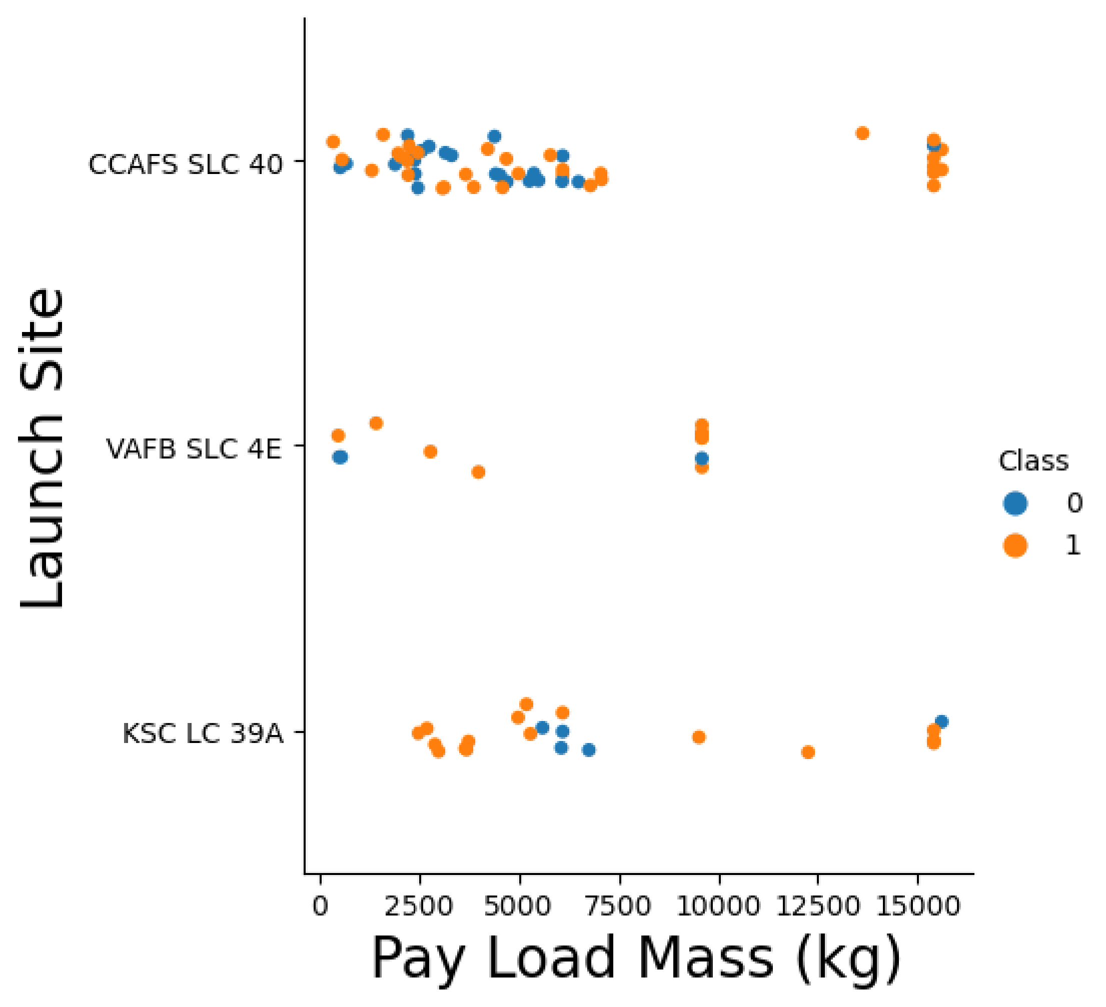
Flight Number vs. Launch Site



- The largest number of flights was carried out at CCAFS SLC 40
- KSC LC 39A seems to have the best flight record

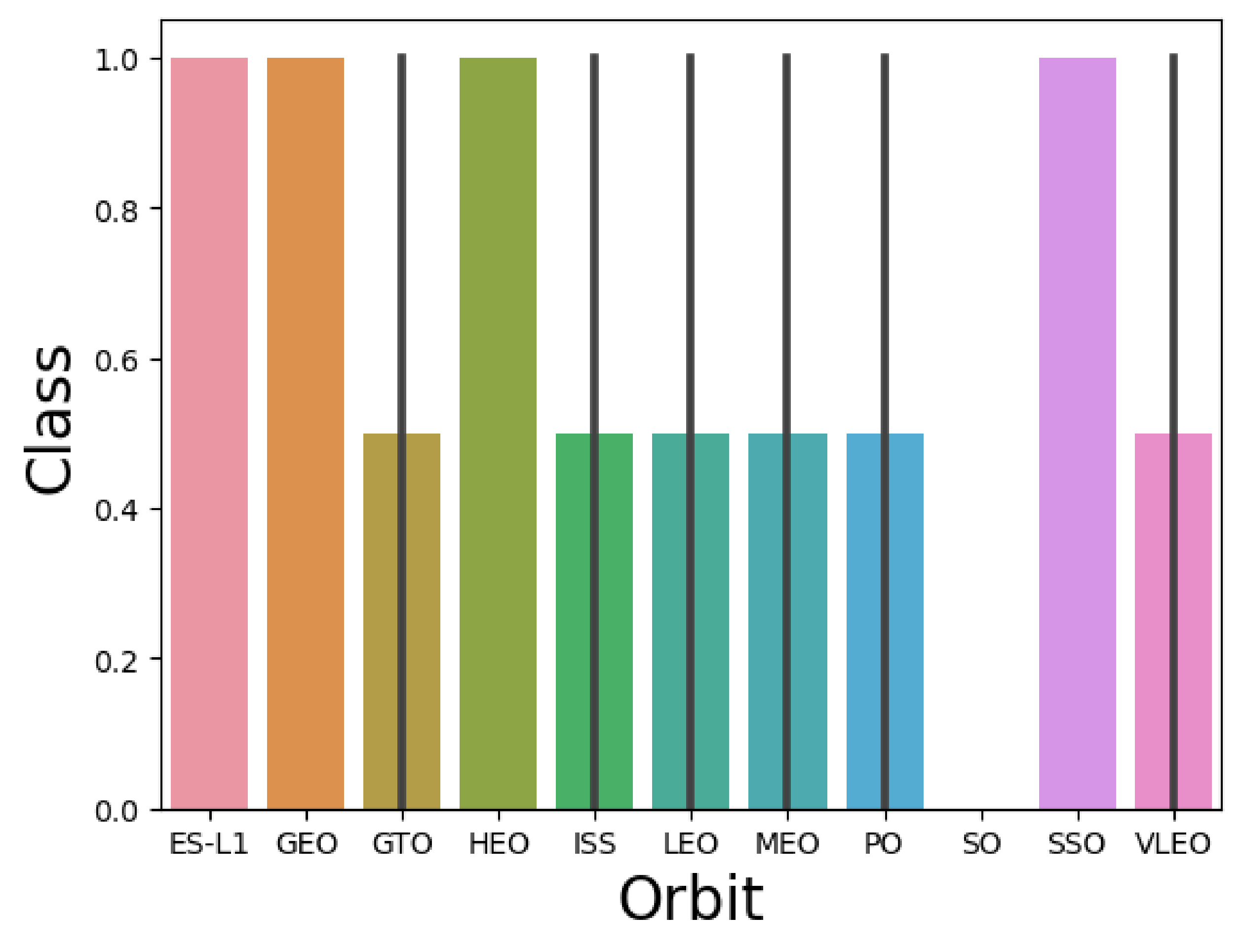
Payload vs. Launch Site

- CCAFS SLC 40 has carried out the most flights with the heaviest payloads with a high success rate
- Payloads of greater than 10000 kg were not lauched form VAFB SLC 4E



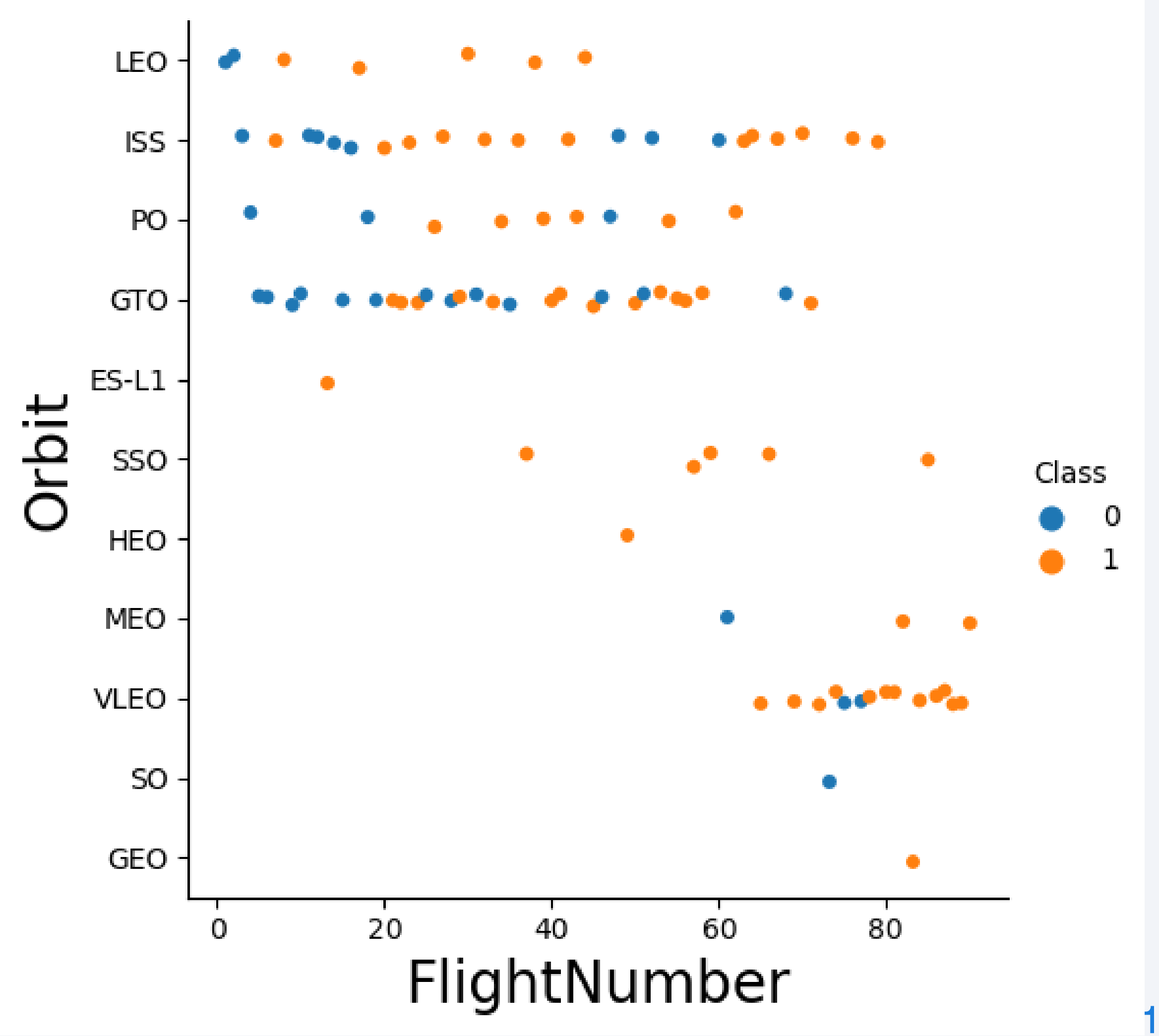
Success Rate vs. Orbit Type

- ES-L2, GEO, HEO and SSO have the most success among launches
- No payloads were launched into the SO orbit



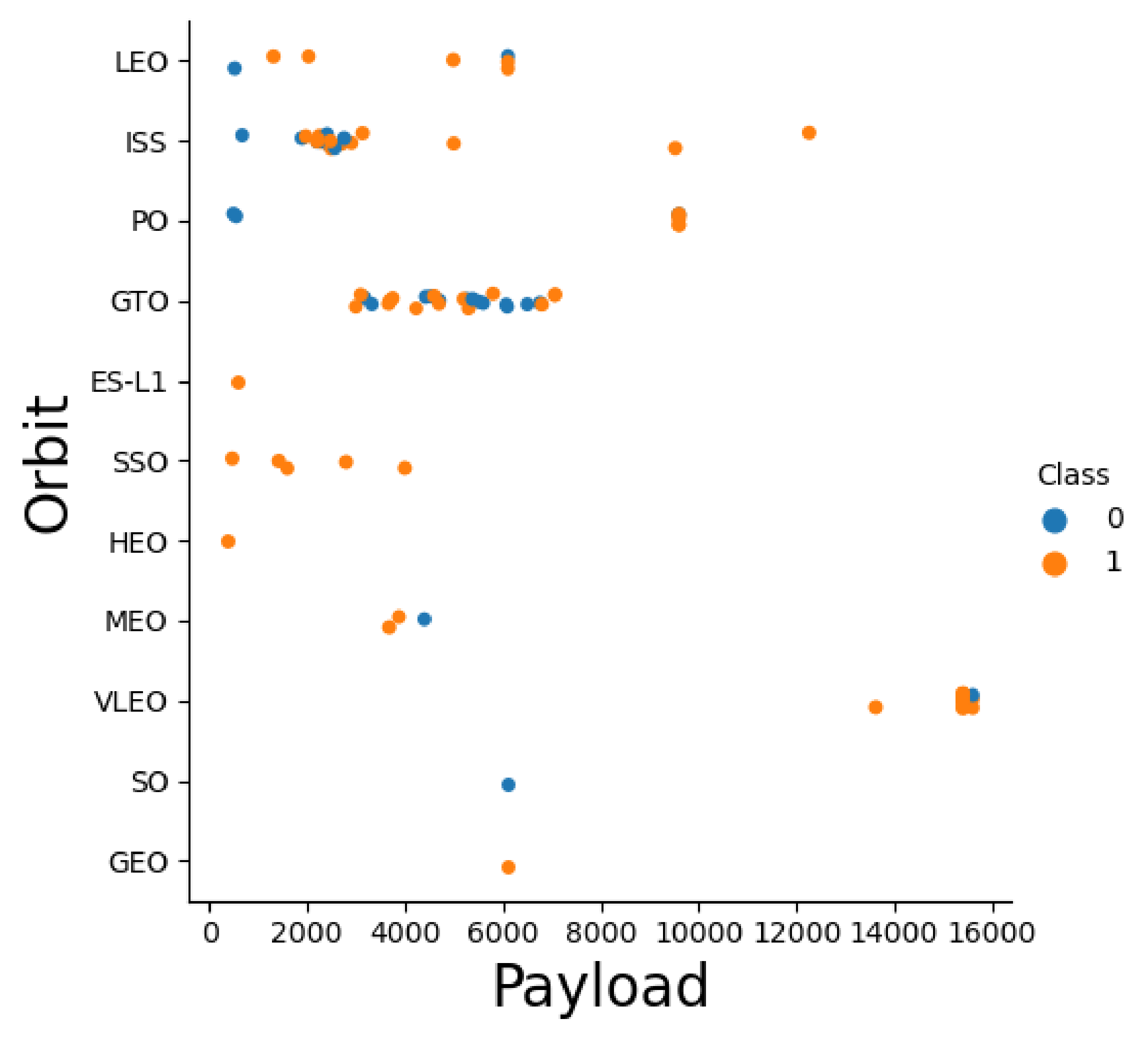
Flight Number vs. Orbit Type

- Most number of flights were launched into the GTO and ISS orbits
- Highest rate of success was seen in VLEO orbit
- As the flight number increases, the rate of success also increases



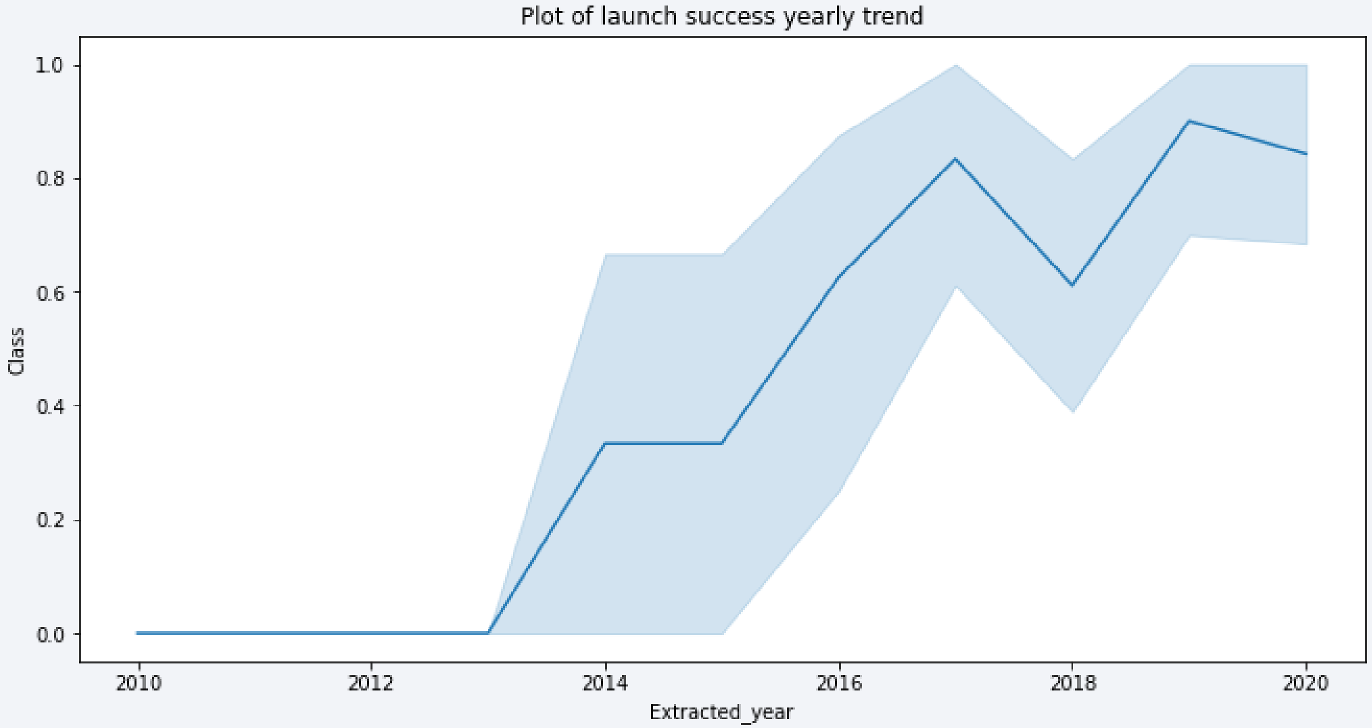
Payload vs. Orbit Type

- Highest mass of payloads were sent into VLEO orbit
- A very specific weight of payload (2000 to 4000 kg) were sent into ISS
- Payloads ranging from 3000 to 7000 kg were sent into GTO orbit



Launch Success Yearly Trend

- As time increases the rate of success increases
- However from 2017 to 2018 a negative trend was seen in terms of launch success
- A similar downward trend is also seen from 2019 to present



All Launch Site Names

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

```
%sql select distinct Launch_Site from SPACEXTBL
```

```
 * sqlite:///my_data1.db
Done.
```

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

- Distinct keyword is used alongwith the column Launch_Site to get the unique values from the column

Launch Site Names Begin with 'CCA'

Task 2

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

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

```
 * sqlite:///my_data1.db
Done.
```

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	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)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

- All keywords containing CCA within the launch_site column are selected.
- Limit keyword is used to limit the displayed result to 5

Total Payload Mass

Task 3

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

```
%sql select sum(PAYLOAD_MASS__KG_) as totalMass from SPACEXTBL;
```

```
 * sqlite:///my_data1.db
Done.
```

totalMass
619967

- Sum was applied to the payload mass in kg and assigned to totalMass

Average Payload Mass by F9 v1.1

Task 4

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

```
%sql select avg(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL;
```

```
 * sqlite:///my_data1.db
Done.
```

payloadmass
6138.287128712871

- Avg was applied to the payload mass in kg and assigned to payloadMass

First Successful Ground Landing Date

Task 5

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

Hint:Use min function

```
%sql select min(DATE) from SPACEXTBL;
```

```
 * sqlite:///my_data1.db
Done.
```

min(DATE)
01-03-2013

- The date (01-03-2013) of first successful landing was found using minimum on Date column

Successful Drone Ship Landing with Payload between 4000 and 6000

Task 6

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

```
%sql select BOOSTER_VERSION from SPACEXTBL where "Landing _Outcome" ='Success (drone ship)' and PAYLOAD_MASS__KG_ BETWEEN 4000 and 6000;
```

```
 * sqlite:///my_data1.db
Done.
```

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

- Multiple queries were used to filter the results
- Landing outcome was Success on drone ship and payload mass was between 4000 and 6000 kg

Total Number of Successful and Failure Mission Outcomes

Task 7

List the total number of successful and failure mission outcomes

```
%sql select Mission_Outcome,count(Mission_Outcome) as MissionOutcomes from SPACEXTBL Group by Mission_Outcome
```

```
 * sqlite:///my_data1.db
Done.
```

Mission_Outcome	MissionOutcomes
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

- Count on mission outcome was used to find the number of successful and failed mission outcomes

Boosters Carried Maximum Payload

Task 8

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

```
%sql select BOOSTER_VERSION as boosterversion from SPACEXTBL where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from SPACEXTBL);
```

```
 * sqlite:///my_data1.db
Done.
```

boosterversion
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

- A subquery was used to calculate the maximum payload mass and corresponding booster version was queried and returned

2015 Launch Records

Task 9

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.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.

```
%sql SELECT substr(DATE,4,2),MISSION_OUTCOME,BOOSTER_VERSION,LAUNCH_SITE,"Landing _Outcome" FROM SPACEXTBL where substr(Date,7,4)='2015' AND
```

```
 * sqlite:///my_data1.db
Done.
```

substr(DATE,4,2)	Mission_Outcome	Booster_Version	Launch_Site	Landing _Outcome
01	Success	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
04	Success	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

- Both failed landing outcomes in 2015 are from the lauch site CCAFS LC-40

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

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

- Present your query result with a short explanation here

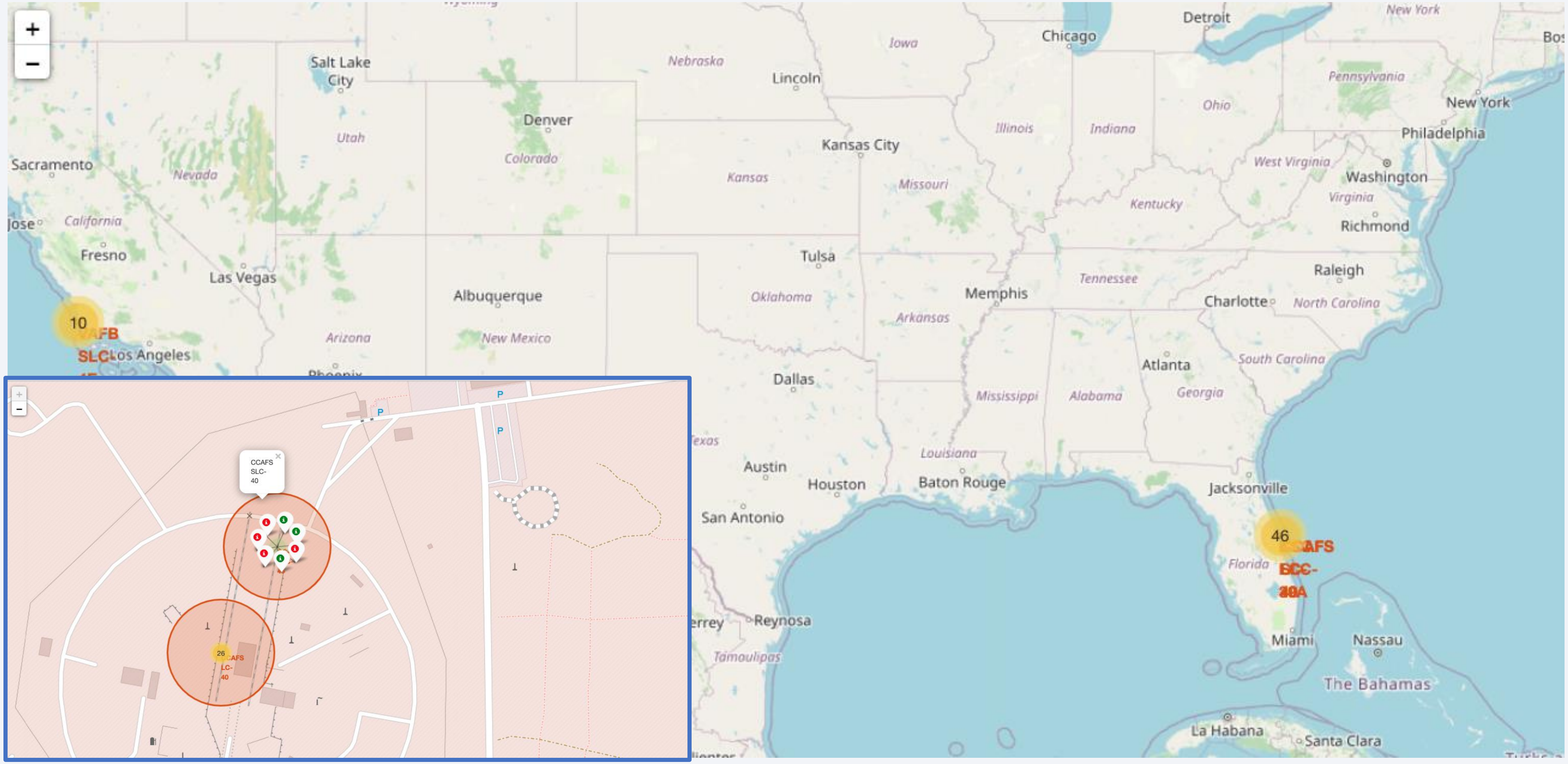
Section 3

Launch Sites Proximities Analysis

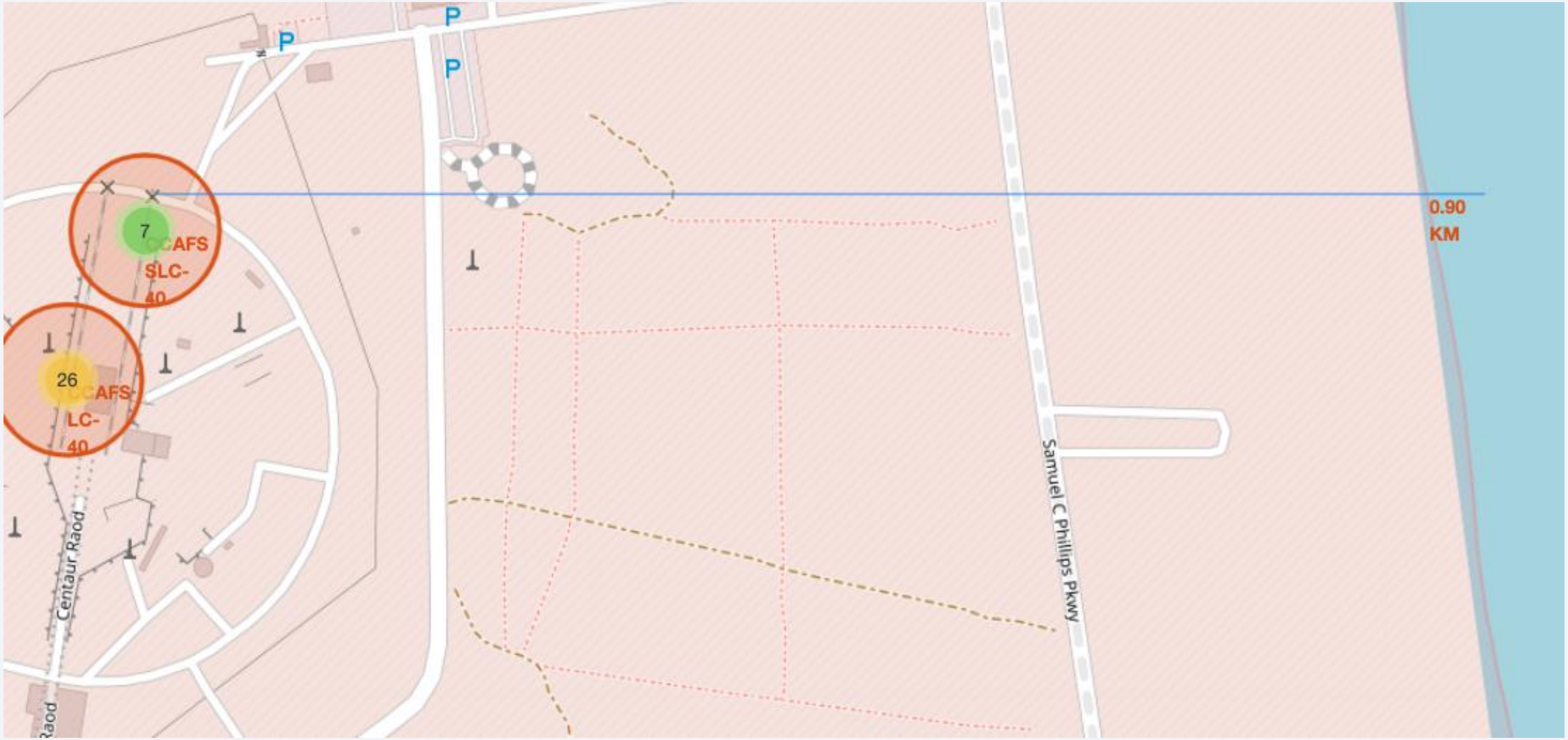
All Launch Sites

- Folium module was used to display the total number of sites

Mark the success/failed launches for each site on the map



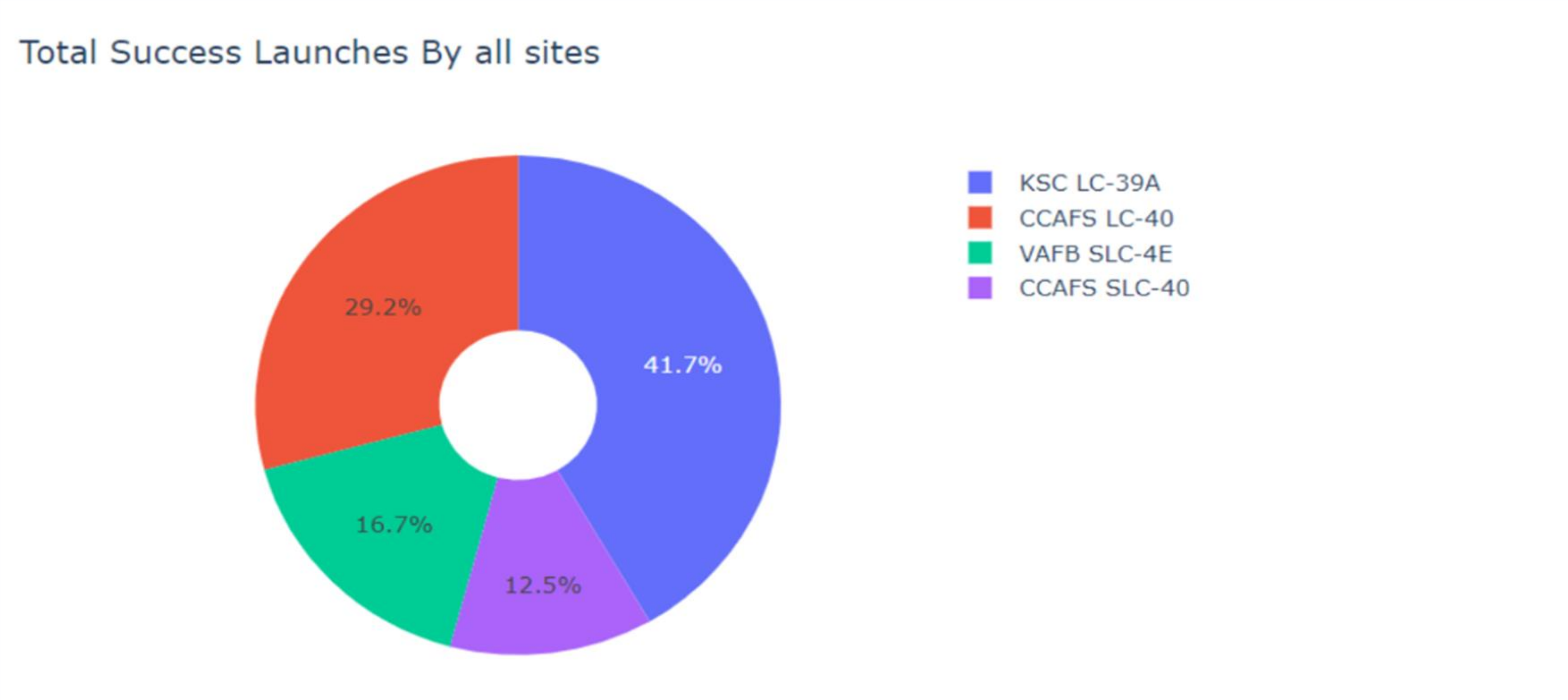
Calculate the distances between a launch site to its proximities



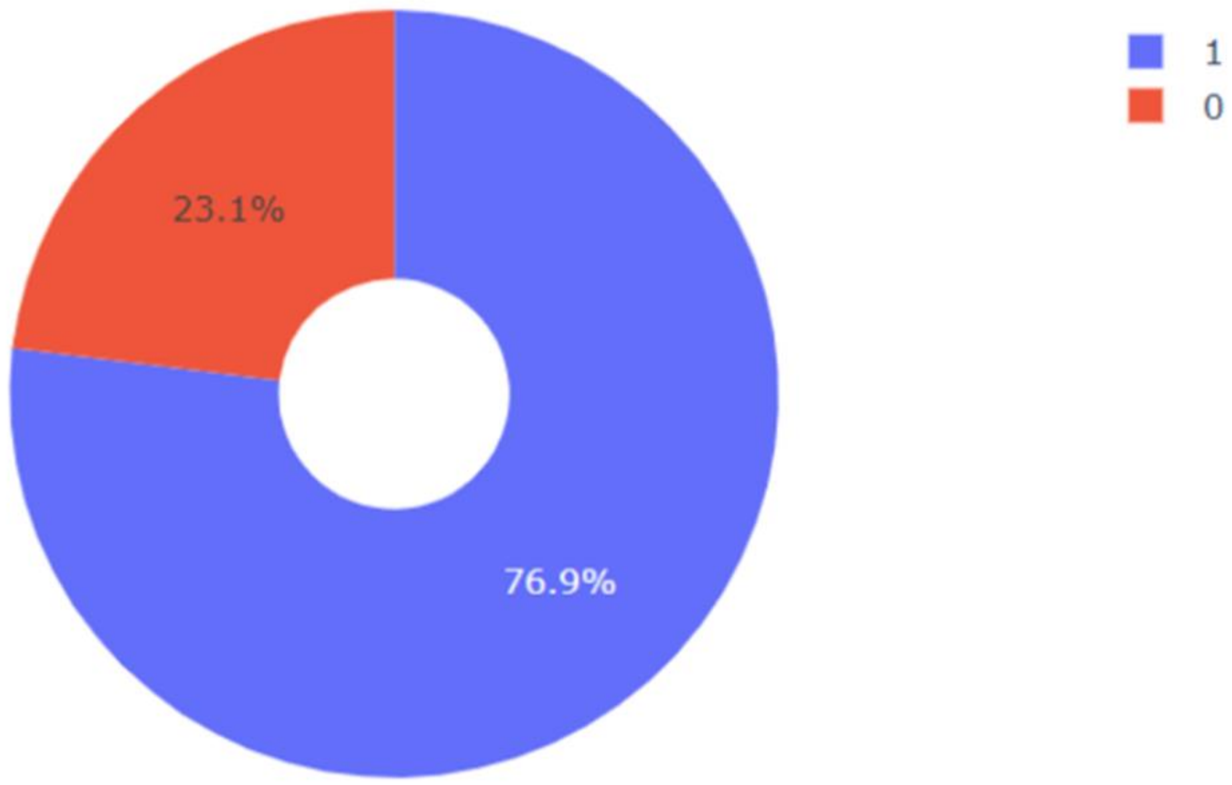
Section 4

Build a Dashboard with Plotly Dash

Success Rates for All Sites

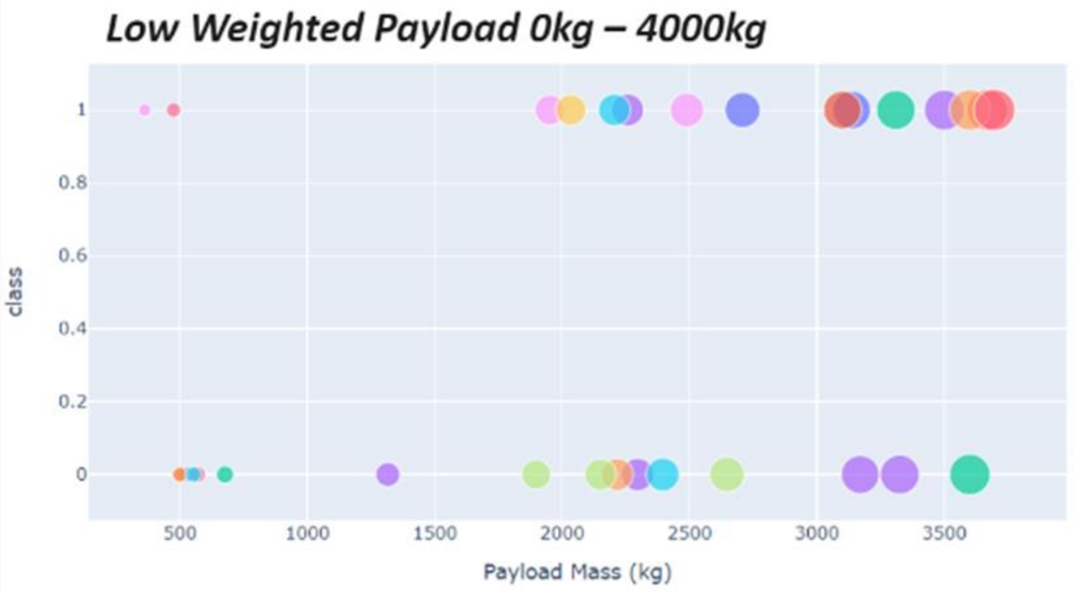


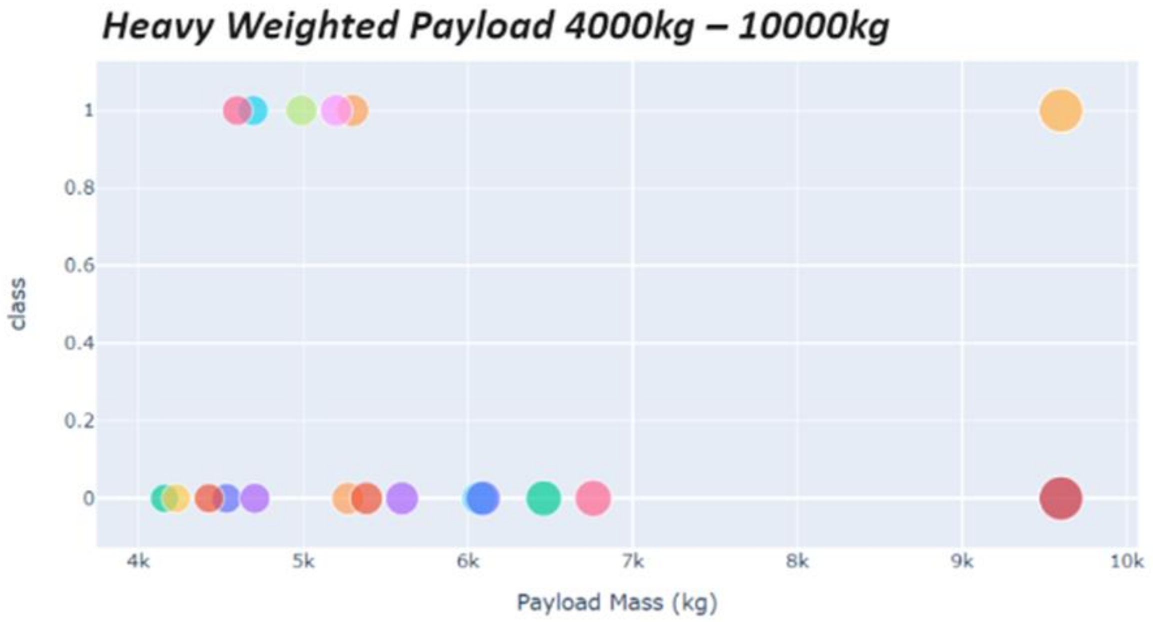
Pie Chart for KSC LC-39A



KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

Payload Range Slider Selection





Highest rate of success was observed in payloads ranging from 2000 to 5000 kg

Section 5

Predictive Analysis (Classification)

Classification Accuracy

TASK 12

Find the method performs best:

```python
algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Tree':
    print('Best Params is :',tree_cv.best_params_)
if bestalgorithm == 'KNN':
    print('Best Params is :',knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best Params is :',logreg_cv.best_params_)
```

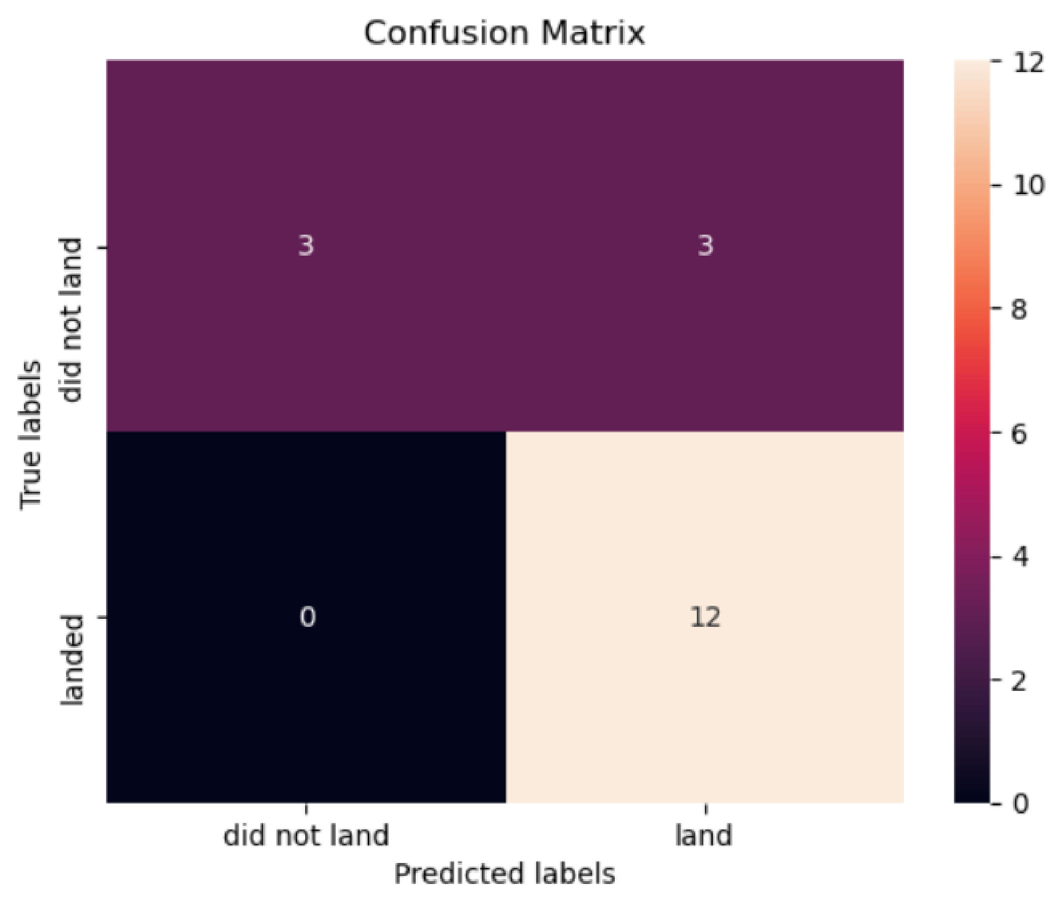
Python

```
Best Algorithm is Tree with a score of 0.8888888888888888
Best Params is : {'criterion': 'entropy', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 10, 'splitter': 'random'}
```

- The best performing algorithm is Tree with a score of 0.888 or 88% accuracy

Confusion Matrix

```
yhat = svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



- Confusion matrix of tree is shown in the figure
- The algorithm is very good at predicting failed landings
- Some false positives are reported for successful landing

Conclusions

- The larger the flight amount at a launch site, the greater the success rate at a launch site
- Launch success rate saw an overall increase however there were some negative trends between some years
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate
- KSC LC-39A had the most successful launches of any sites
- The Decision tree classifier has the best accuracy out of the different machine learning algorithms

Thank you!