Winning Space Race with Data Science

Valentina Ocloo
27.01.2024

Outline

- Executive Summary
- Introduction
- Methodology
- Results
  - Exploratory Analysis using SQL
  - Exploratory Data Analysis((EDA) with Visualization
  - Interactive Visual Analytics with Folium
  - Interactive Dashboard with Ploty Dash
  - Machine Learning Prediction
- Conclusion
- Appendix

Executive Summary

Summary of methodologies

- Collect data using SpaceX Rest API and web scraping techniques
- Wrangle data to obtain success or fail outcome variable
- Explore data with visualization techniques
- Analyze the data with SQL
- Explore and visualize launch site success rate and proximity to geographical markers
- Build models to predict the landing outcomes

Summary of all results

- EDA results, Interactive/Visualization analytics, Predictive analysis

Introduction

- Project background

SpaceX is a commercial space leader in providing space travel affordable for everyone and they advertises Falcon 9 rockets launches on its website, with the lowest cost of 62 million dollars and reuse of it in the first stage compared to other provides. Since we can determine if the first stage will land then we can estimate the cost of a launch so as bid against other competitors. The objective of this project is to use existing data to create machine learning pipeline to predict whether SpaceX can land successfully in their first stage.

- Problem Statement
  - To determine the what factors leads to a first-stage rocket landing success
  - To determine the rate of successful landings over time
  - To determine the best predictive model for successful landing

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data collected through SpaceX Rest API and web scraping from Wikipedia
- Perform data wrangling
  - Filtered data, handled miss values and apply one-hot encoding to category 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
  - Tune and evaluate classification models to find the best classifier and parameters

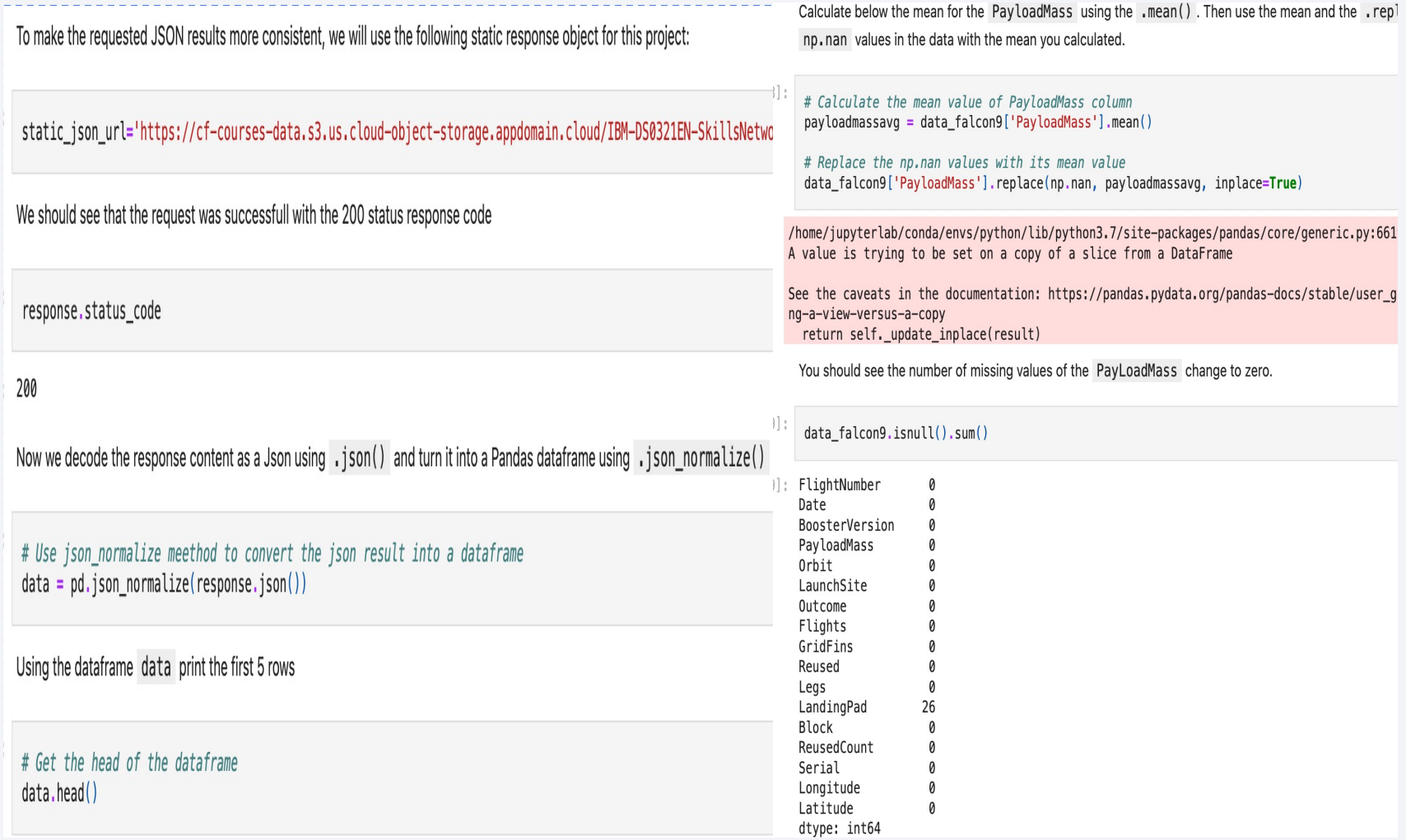
Data Collection

- Several collecting data technique
  - Firstly, data collected using get request from the SpaceX Rest API
  - Next, we decode the response content using .json() function to call and turn it into a pandas dataframe using json_normalize()
  - Clean data through filtering, checked missing values and fill in necessary missing values (Payload Mass) by replacing by its .mean()
  - Export data to csv file
  - Furthermore, we performed web scraping from Wikipedia for Falcon 9 records using BeautifulSoup
  - Extracted the launch records as HTML table, parse the table and convert it to panda dataframe for subsequent analysis

Data Collection – SpaceX API

- Requests to get SpaceX API and decode response content as json and then normalize it to turn into dataframe. Finally perform data wrangling and formatting.

- Click here https://github.com/valentina650/Data-Science-Capstone/blob/main/SpaceX-data-collection-api.ipynb



To make the requested JSON results more consistent, we will use the following static response object for this project:

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

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

Using the dataframe `data` print the first 5 rows

```
# Get the head of the dataframe
data.head()
```

Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the mean and the `.rep` `np.nan` values in the data with the mean you calculated.

```
# Calculate the mean value of PayloadMass column
payloadmassavg = data_falcon9['PayloadMass'].mean()

# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].replace(np.nan, payloadmassavg, inplace=True)
```

```
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/pandas/core/generic.py:661
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
ng-a-view-versus-a-copy
  return self._update_inplace(result)
```

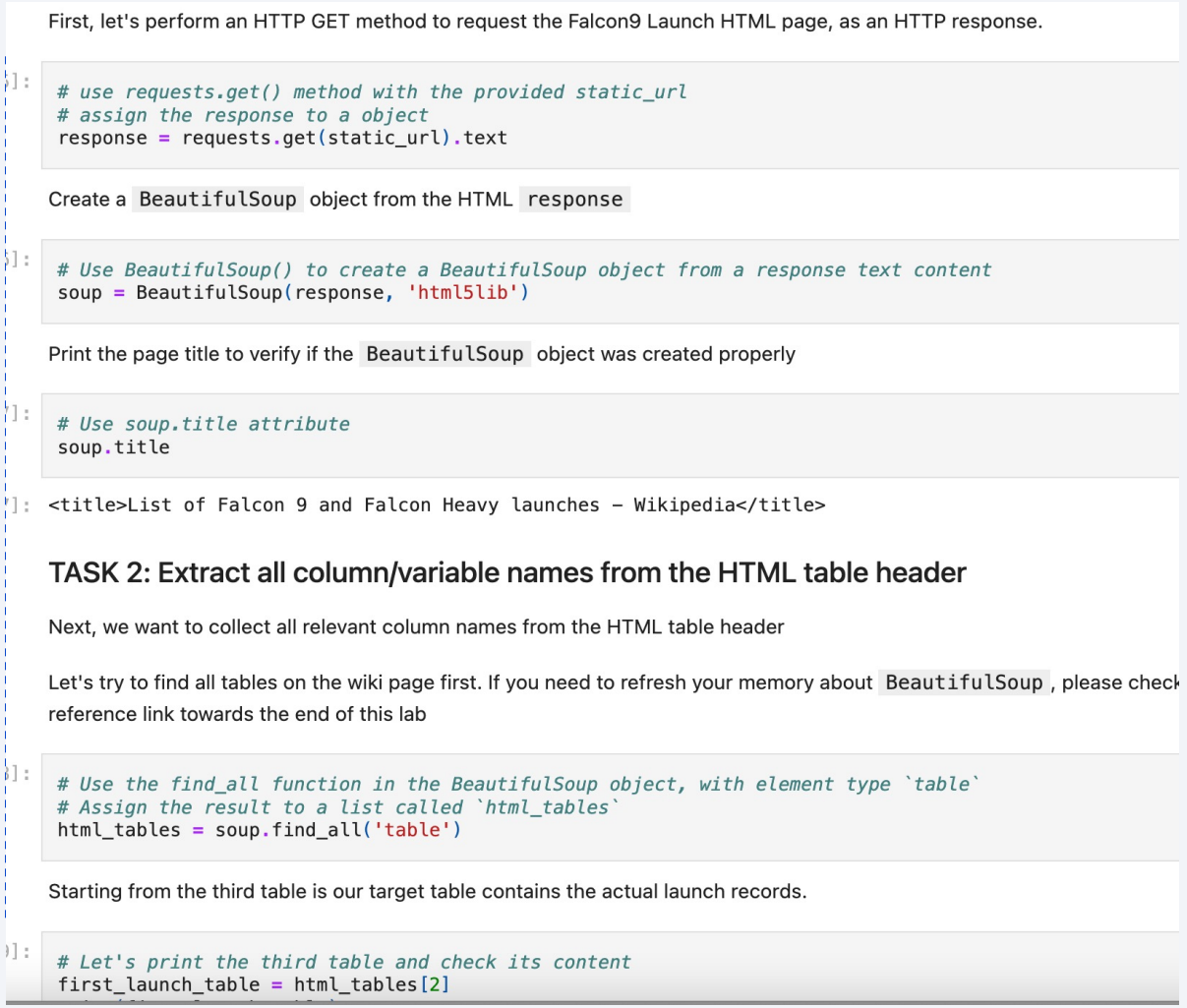
You should see the number of missing values of the `PayLoadMass` change to zero.

```
data_falcon9.isnull().sum()
```

```
FlightNumber      0
Date              0
BoosterVersion    0
PayloadMass       0
Orbit             0
LaunchSite        0
Outcome           0
Flights           0
GridFins          0
Reused            0
Legs              0
LandingPad       26
Block             0
ReusedCount       0
Serial            0
Longitude         0
Latitude          0
dtype: int64
```

Data Collection - Scraping

- Performed webscrap on Falcon 9 launch records using BeautifulSoup
- Pared the table and converted it into dataframe
- Click here https://github.com/valentina650/Data-Science-Capstone/blob/main/SpaceX_Webscraping.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
response = requests.get(static_url).text
```

Create a `BeautifulSoup` object from the HTML `response`

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response, 'html5lib')
```

Print the page title to verify if the `BeautifulSoup` object was created properly

```
# Use soup.title attribute
soup.title
```

```
<title>List of Falcon 9 and Falcon Heavy launches – Wikipedia</title>
```

TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check reference link towards the end of this lab

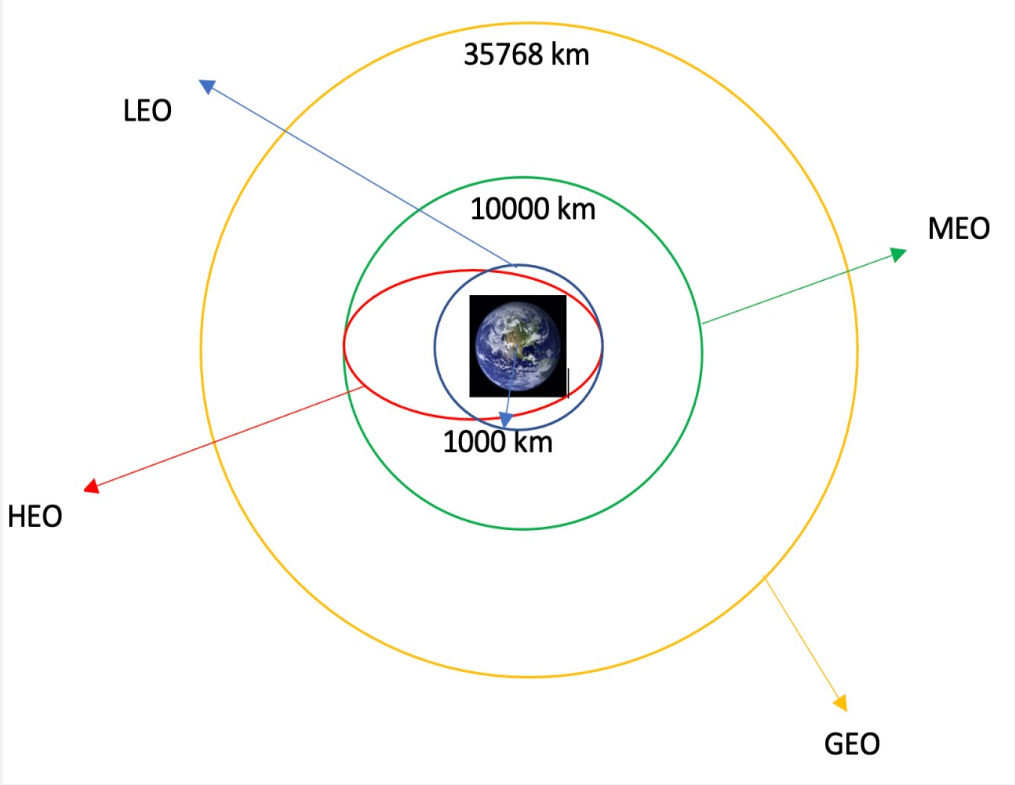
```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

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

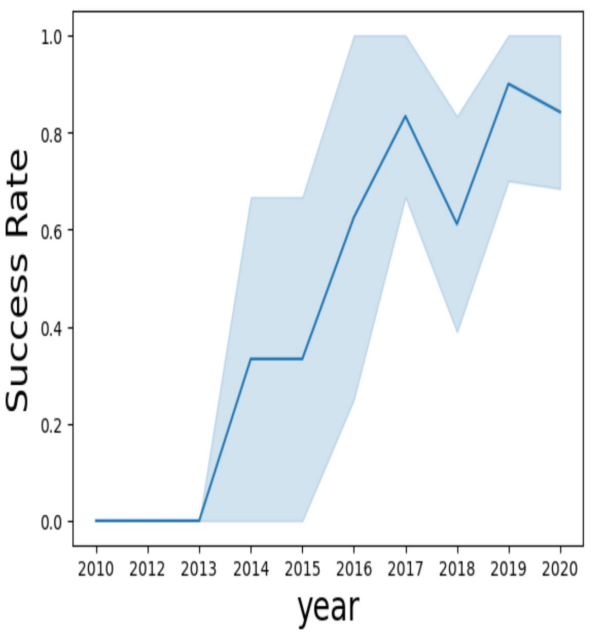
```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
```

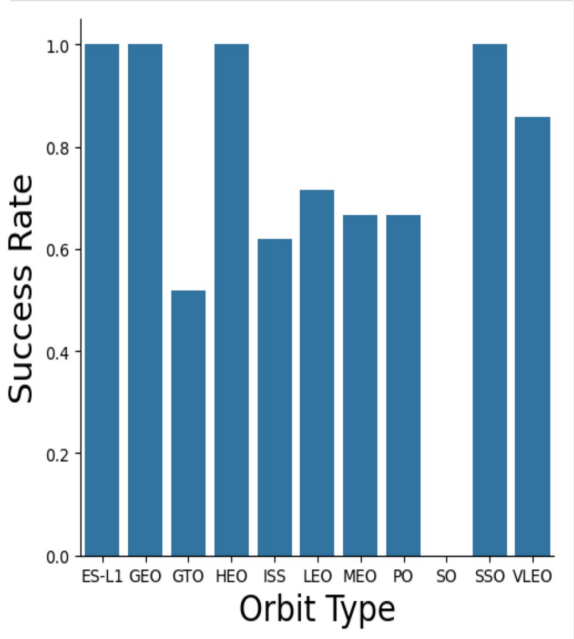
Data Wrangling

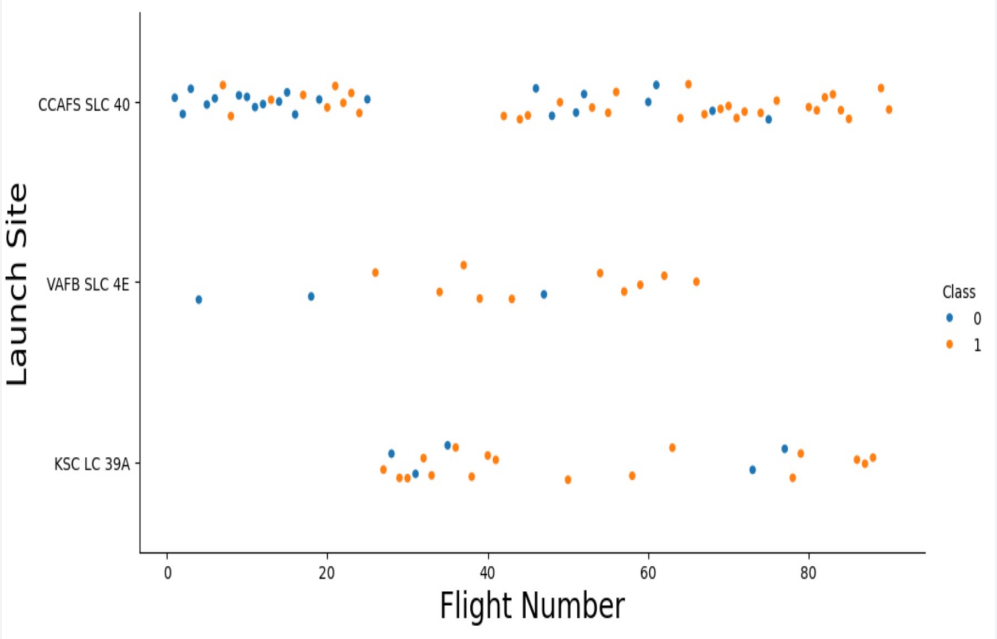
- Performed exploratory data analysis and determined the training labels
- Calculated the number of launches at each site, and the number of occurrence of each orbit and mission outcomes
- Create landing outcome label (landing class) to help determine the success rate and saved as csv file
- Click here https://github.com/valentina650/Data-Science-Capstone/blob/main/SpaceX-Data wrangling.ipynb



EDA with Data Visualization







Click here https://github.com/valentina650/Data-Science-Capstone/blob/main/SpaceX_EDA_Data_Visualization.ipynb

- Explored the data by visualizing the relationship between flight number and launch site, payload and launch site, success rate of each orbit type, flight number and orbit type and then launch success yearly trend.

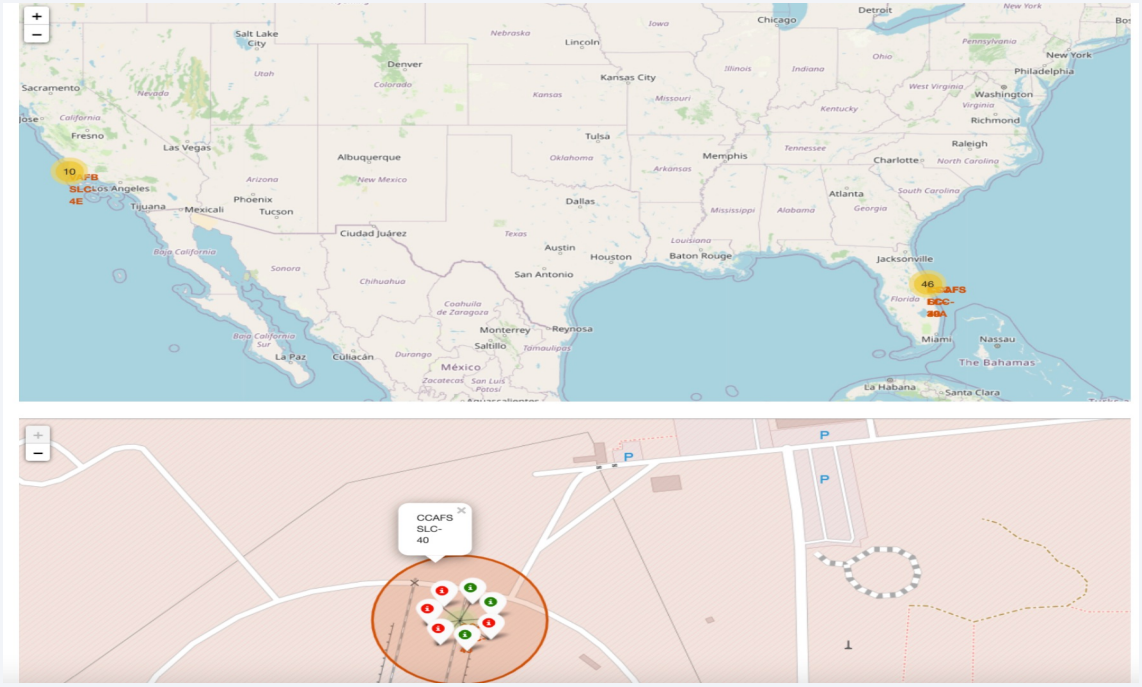
EDA with SQL

- Load the SpaceX dataset into sql database on the jupyter notebook and applied EDA with SQLL to get insight from the data such as:
  - The names of the unique launch sites in the space mission
  - The total payload mass carried by booster launch by NASA (CRS)
  - The average payload mass carried by booster version F9 v1.1
  - The date when first successful landing outcome in ground pad was achieved
  - The names of boosters which have success in drone ship and have payload mass > 400 but < 6000
  - The total number of success and failure mission outcomes

- Click here https://github.com/valentina650/Data-Science-Capstone/blob/main/SpaceX_EDA_SQL.ipynb

Build an Interactive Map with Folium

- Marked all launch sites and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map. This helps to finding an optimal location for building launch site
- Assigned the feature launch outcomes(failure- 0 or success-1)
- Used the color-labeled marker clusters helps to identify the success rate
- Calculated the distances between a launch site to its proximities



Click here https://github.com/valentina650/Data-Science-Capstone/blob/main/SpaceX_Interactive_Visual_Analytics_Folium.ipynb

Build a Dashboard with Plotly Dash

- Built an interactive dashboard with Ploty dash
- Plotted pie charts showing the total launches by a certain sites
- Plotted scatter graph showing the relationship with Outcome and Payload

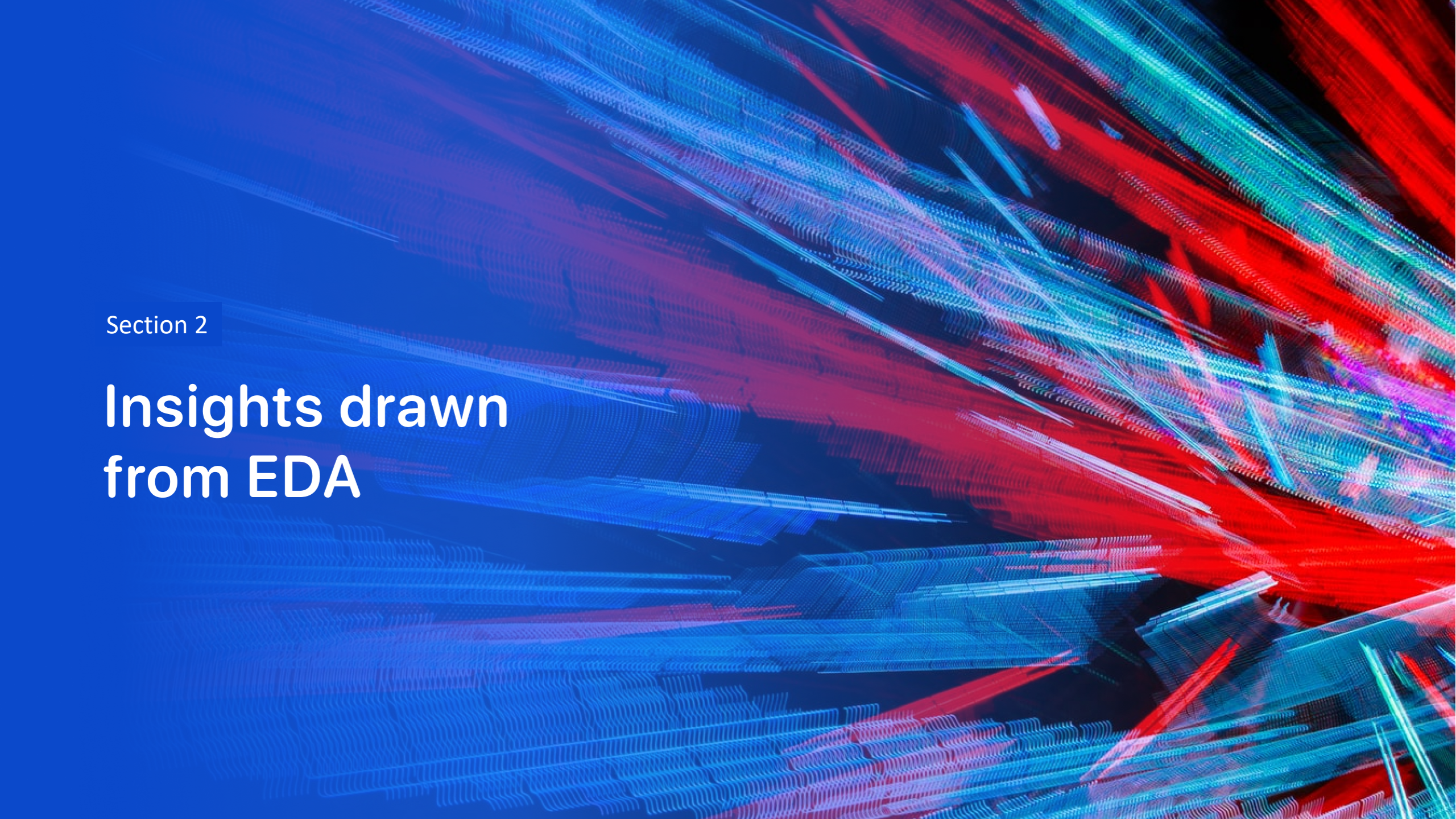
  Mass(kg) for the different booster version
- Click here https://github.com/valentina650/Data-Science-Capstone/blob/main/SpaceX_Visual_Analytics_Plotly.py

Predictive Analysis (Classification)

- Loaded the data using numpy and pandas, transformed the data, and split the data into training and testing
- Built different machine learning models (logistic regression, support vector machine(SVM), decision tree, and K-Nearest Neighbor(KNN)) and tune different hyperparameters using GridSearchCV
- Used accuracy as the metric for our model and improved the model using algorithm tuning
- Asses the confusion matrix for all models
- Best classification model was obtained using Jaccard score, F1 score and accuracy
- Click here https://github.com/valentina650/Data-Science-Capstone/blob/main/SpaceX_Machine_Learning_Prediction.ipynb

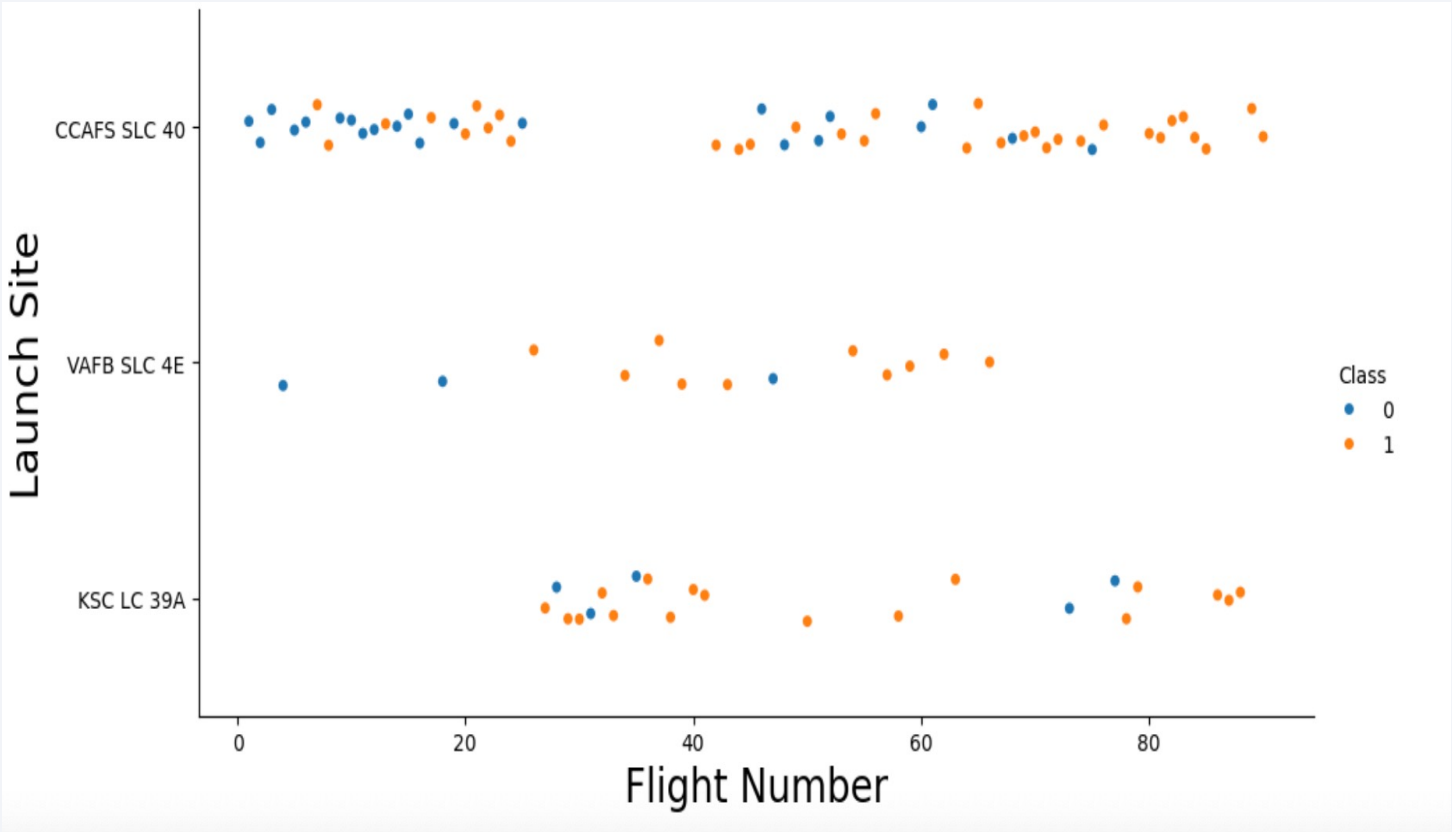
Results

- Exploratory data analysis
  - Launch success improved over time
  - KSC LC-39A has the highest success rate among landing sites
  - ES-L1, GEO, HEO and SSO orbits recorded 100% success rate
- Interactive analytics
  - Most launch sites are near the equator and all are close to the coast
  - Launch sites are far enough away from anything a failed launch can damage (city, highway, railways), while still close enough to bring people and material to support launch activities
- Predictive analysis
  - Decision Tree model is the best predictive model with score as 0.875 for the dataset


Section 2
Insights drawn from EDA

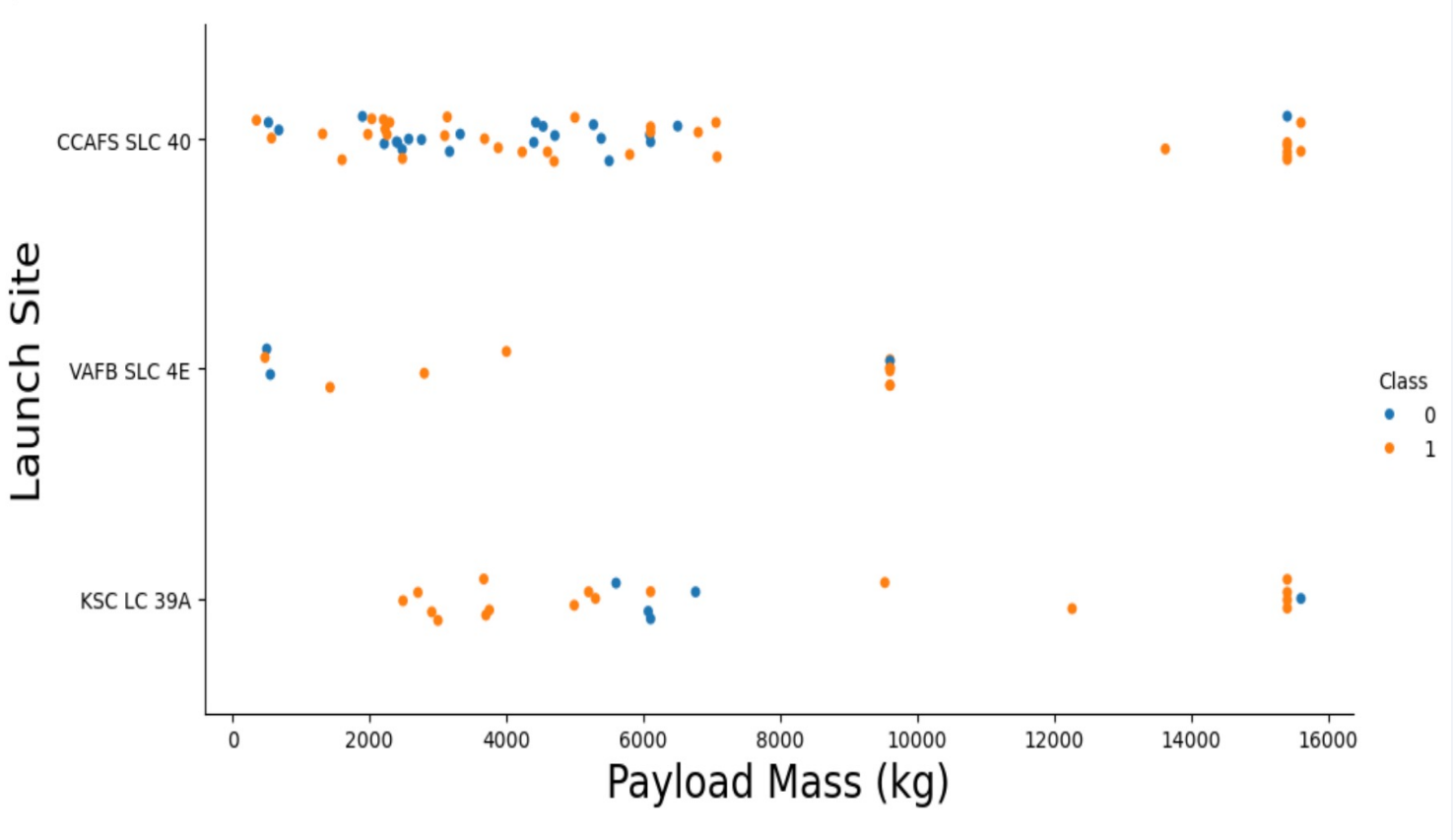
Flight Number vs. Launch Site

- The (blue=fail) and (orange = success)
- Earlier flights had more fails (lower success rate)
- Later flights had more success rate
- New launches have higher success rate
- VAFB SLC 4E have less launches but higher success rate



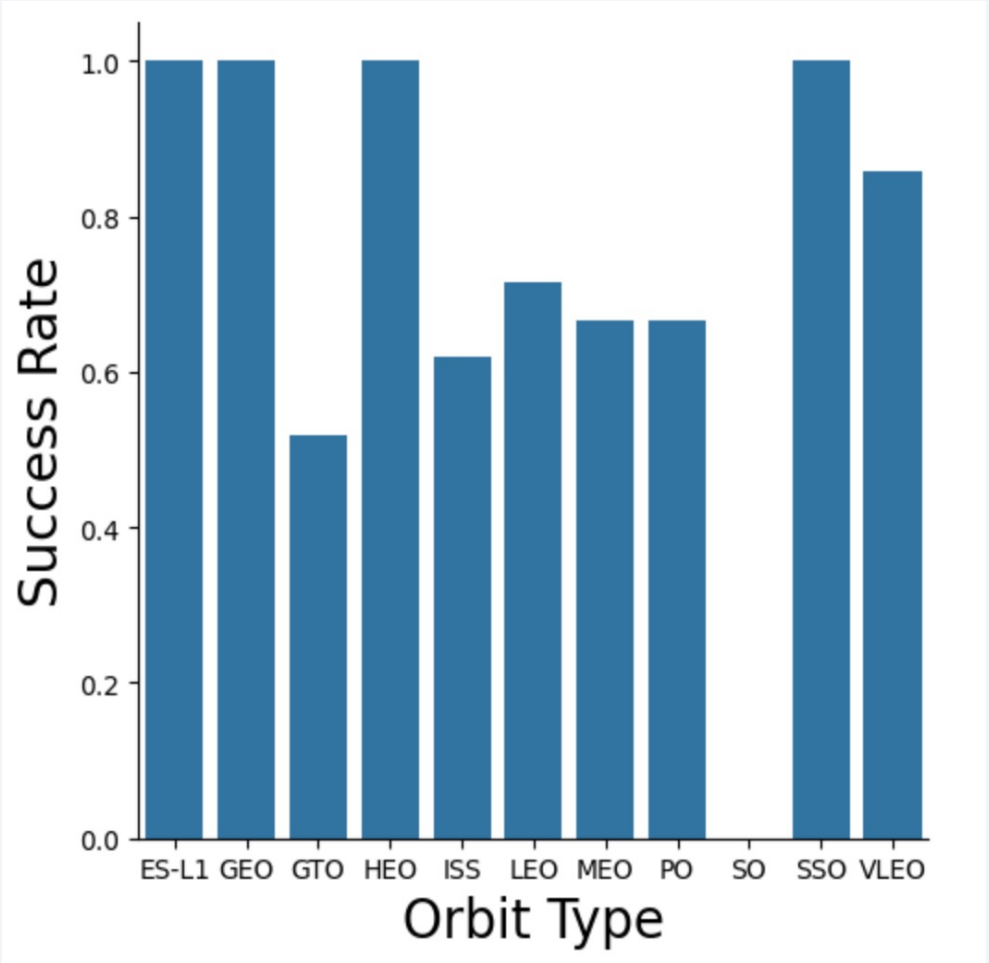
Payload vs. Launch Site

- The most payload with lower mass were launched from CCAF SLC 40 sites
- VAFB SLC 4E has never launched anything greater than 10,000 kg



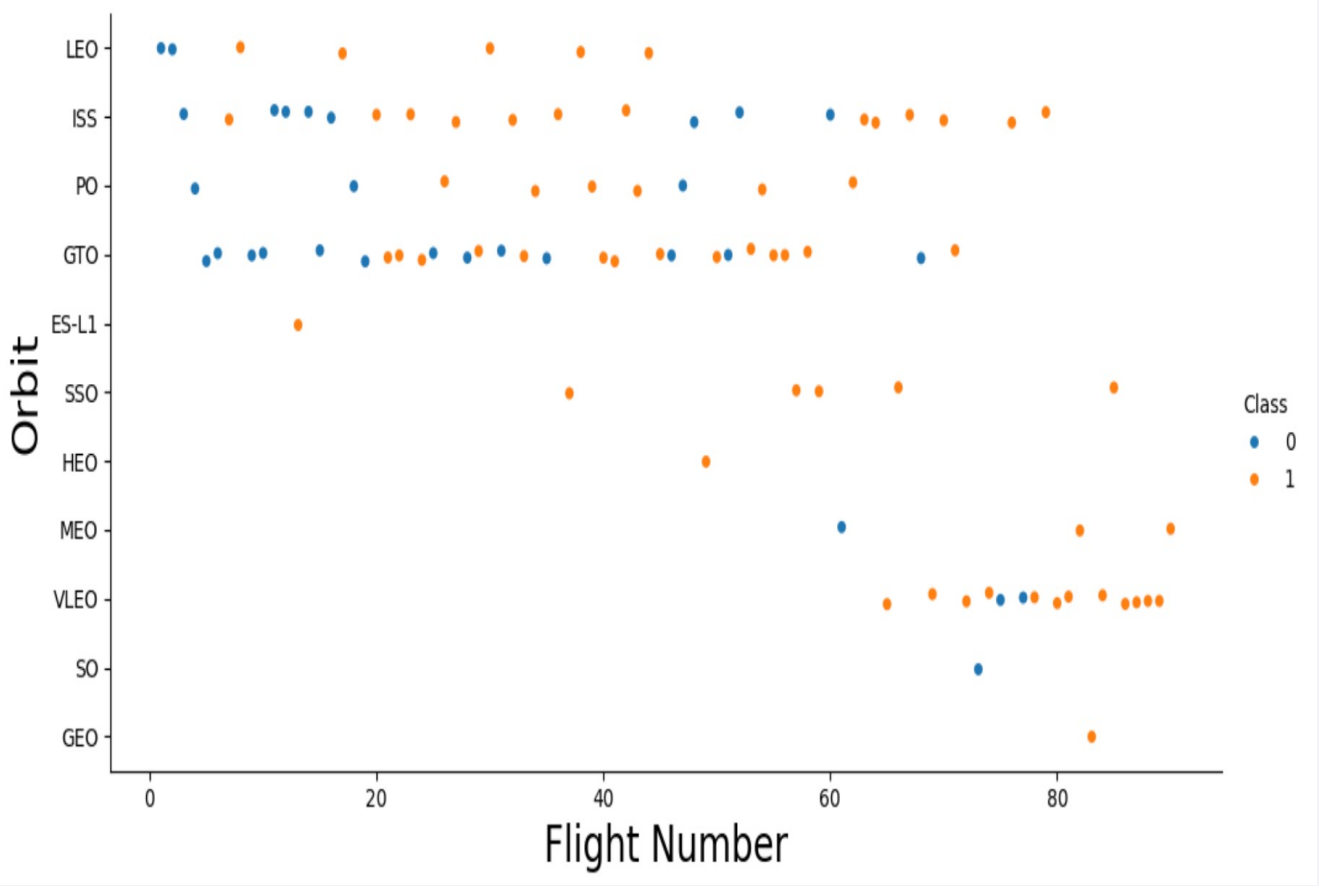
Success Rate vs. Orbit Type

- ES-L1,GEO,HEO and SSO have 100% success rate
- VLEO has approx. 84% success rate
- GTO, ISS, LEO, MEO and PO have 50-80% success rate
- Only SO has 0% success rate



Flight Number vs. Orbit Type

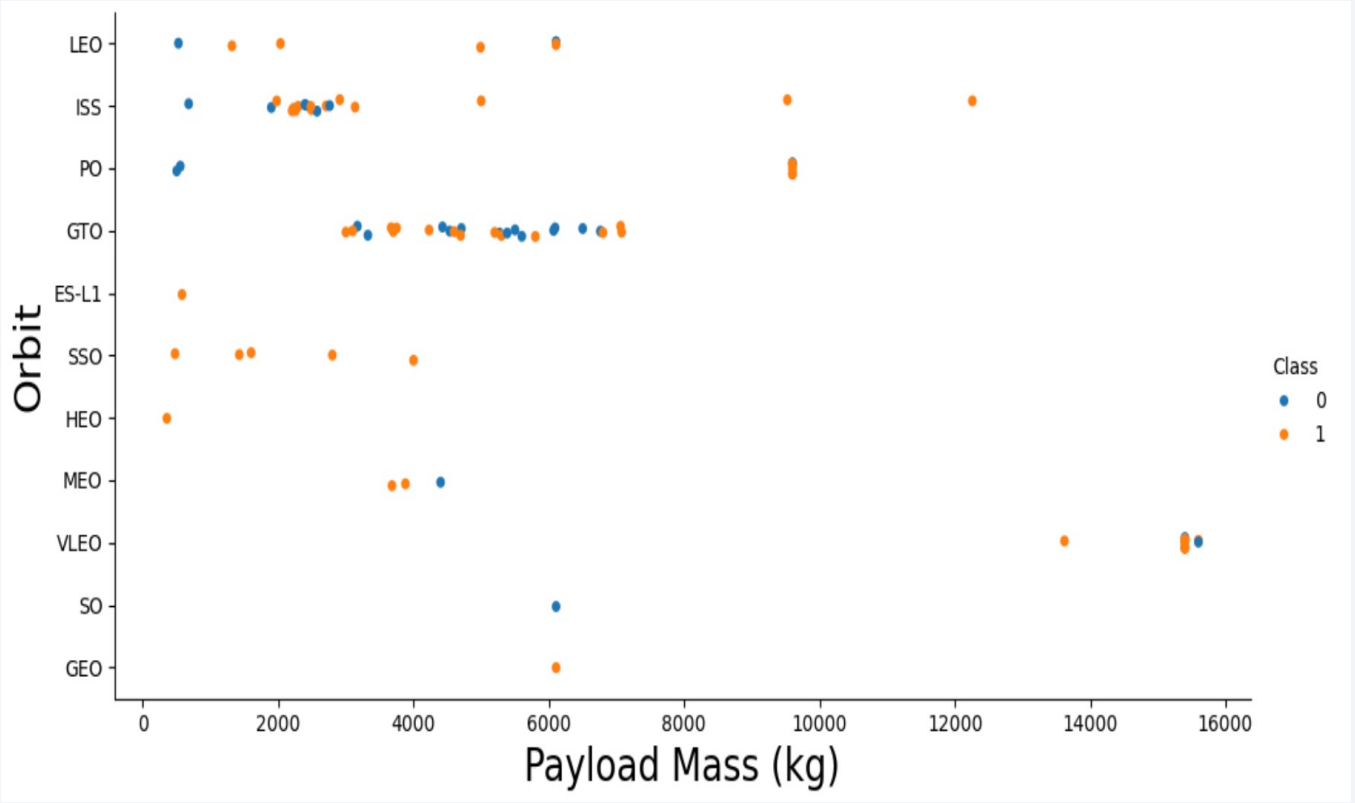
- LEO success is related to number flights whereas GTO has no relationship between flight number and orbit.



Payload vs. Orbit Type

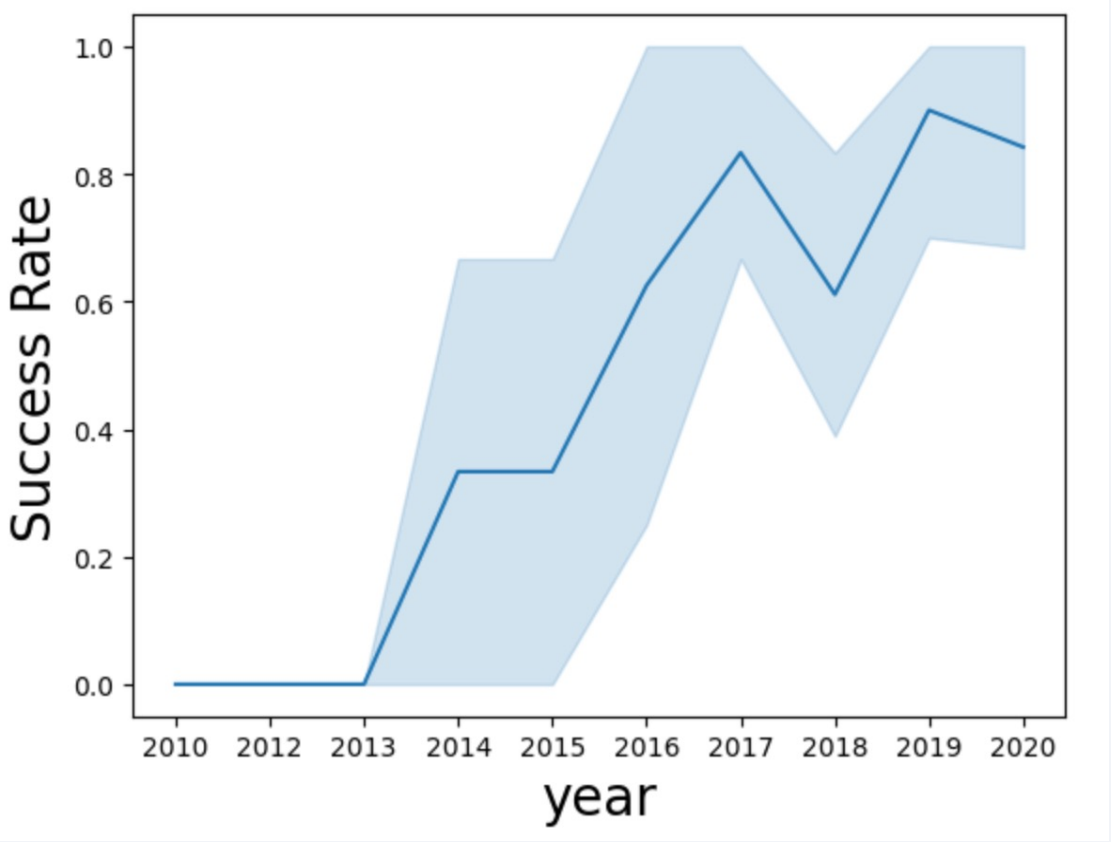
- Higher payloads are better with LEO, ISS and PO orbits

- GTO orbit has mixed success with heavier payloads



Launch Success Yearly Trend

- From 2013-2017 success rate has improved
- A declined of success rate from 2017-2018 and 2019-2020 was observed
- Overall the success rate has improved since 2013



All Launch Site Names

- DISTINCT keyword in SQL was used to show unique launch sites from SpaceX as seen below

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

Launch Site Names Begin with 'CCA'

- We used wild card LIKE query to obtain ‘CCA’ and as seen

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
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	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	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

- The SUM calculate the total payload mass and then WHERE clause query out ‘NASA (CRS)’ as 45596

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

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) \
    FROM SPACEXTBL \
    WHERE CUSTOMER = 'NASA (CRS)';
```

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

SUM(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

- We calculated the average payload mass carried WHERE booster version ‘F9 v1.1’ and obtained 2928.4

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

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) \
    FROM SPACEXTBL \
    WHERE BOOSTER_VERSION = 'F9 v1.1';
```

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

AVG(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

- We observed that first successful ground landing was 2015-12-22

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

Hint:Use min function

```
%sql SELECT MIN(DATE) AS "First Successful Landing" FROM SPACEXTBL WHERE landing_outcome = 'Success (ground pad)'
```

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

First Successful Landing
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- The WHERE clause to filter for boosters which have success in drone ship and used the AND condition to determine successful landing with payload mass greater than 4000 but less than 6000

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 SPACEX WHERE landing_outcome = 'Success (drone ship)' AND payload_mass_kg_ BETWE
```

```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) no such table: SPACEX
[SQL: SELECT booster_version FROM SPACEX WHERE landing_outcome = 'Success (drone ship)' AND payload_mass_kg_ BETWEE
N 4000 AND 6000;]
(Background on this error at: http://sqlalche.me/e/e3q8)
```

Total Number of Successful and Failure Mission Outcomes

- We use COUNT and GROUP BY the Mission Outcomes was success or a failure.

```
%sql SELECT MISSION_OUTCOME, COUNT(*) as total_number \
FROM SPACEXTBL \
GROUP BY MISSION_OUTCOME;
```

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

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

Boosters Carried Maximum Payload

- We obtained the booster that have carried the maximum payload using a subquery in the WHERE clause and the MAX()

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

```
%sql SELECT BOOSTER_VERSION \
FROM SPACEXTBL \
WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

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

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

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

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, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

```
%sql SELECT booster_version, launch_site FROM SPACEXTBL WHERE DATE LIKE '2015-%' AND \
landing_outcome = 'Failure (drone ship)';
```

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

Booster_Version	Launch_Site
F9 v1.1 B1012	CCAFS LC-40
F9 v1.1 B1015	CCAFS LC-40

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

- We selected Landing outcomes and the COUNT it and used WHERE clause to filter the landing outcomes BETWEEN the date 2010-06-04 and 2017-03-20 by GROUP and COUNT in descending order

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.

```
%sql SELECT landing_outcome as "Landing Outcome", COUNT(landing_outcome) AS "Total Count" FROM SPACEXTBL \
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' \
GROUP BY  landing_outcome \
ORDER BY COUNT(landing_outcome) DESC ;
```

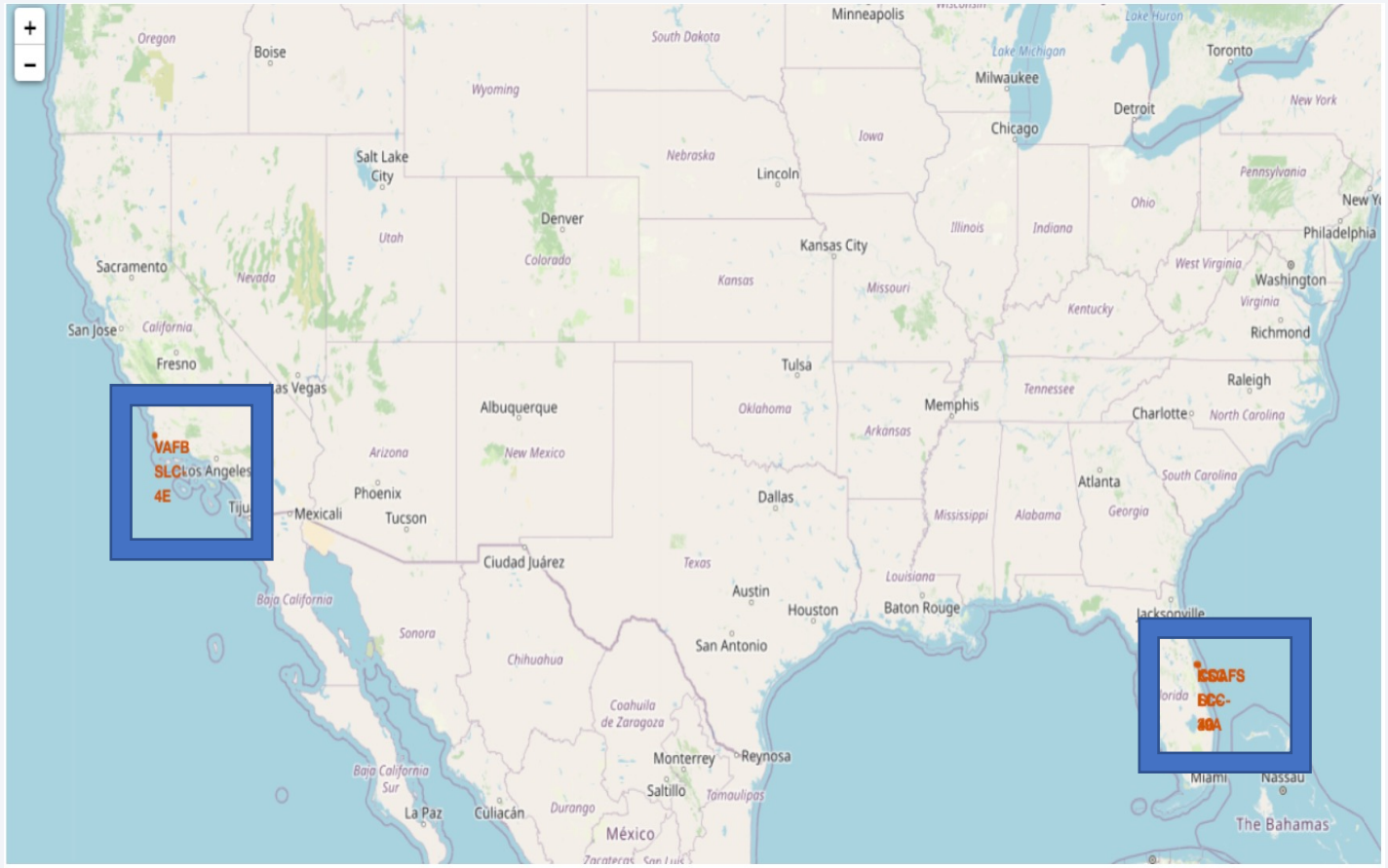
```
 * sqlite:///my_data1.db
Done.
```

Landing Outcome	Total Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1


Section 3
Launch Sites
Proximities Analysis

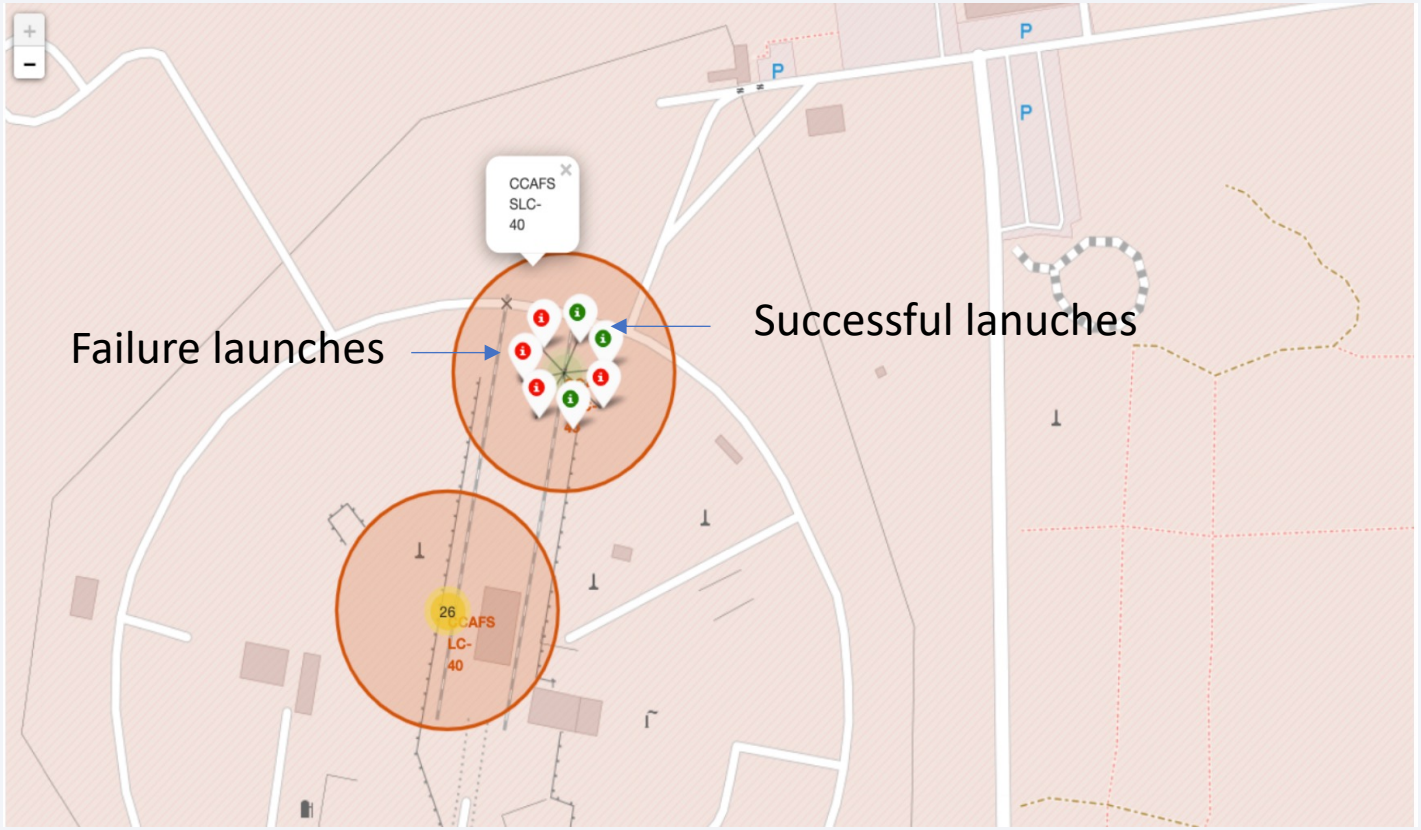
Launch Sites on Global Map

- The launch sites are closer to the equator (USA), which easier to launch to the equatorial orbit
- The rockets launched sites near the equator get an additional natural boost
- It helps to reduce cost of putting in extra fuel and boosters
- Reduce accidents at residential areas



Launch Outcomes

- Launch sites CCAFS SLC-40 has 3/7 success rate (42.9%)



Distance to Proximities

```
City Distance 23.234752126023245
Railway Distance 21.961465676043673
Highway Distance 26.88038569681492
Coastline Distance 0.8627671182499878
```

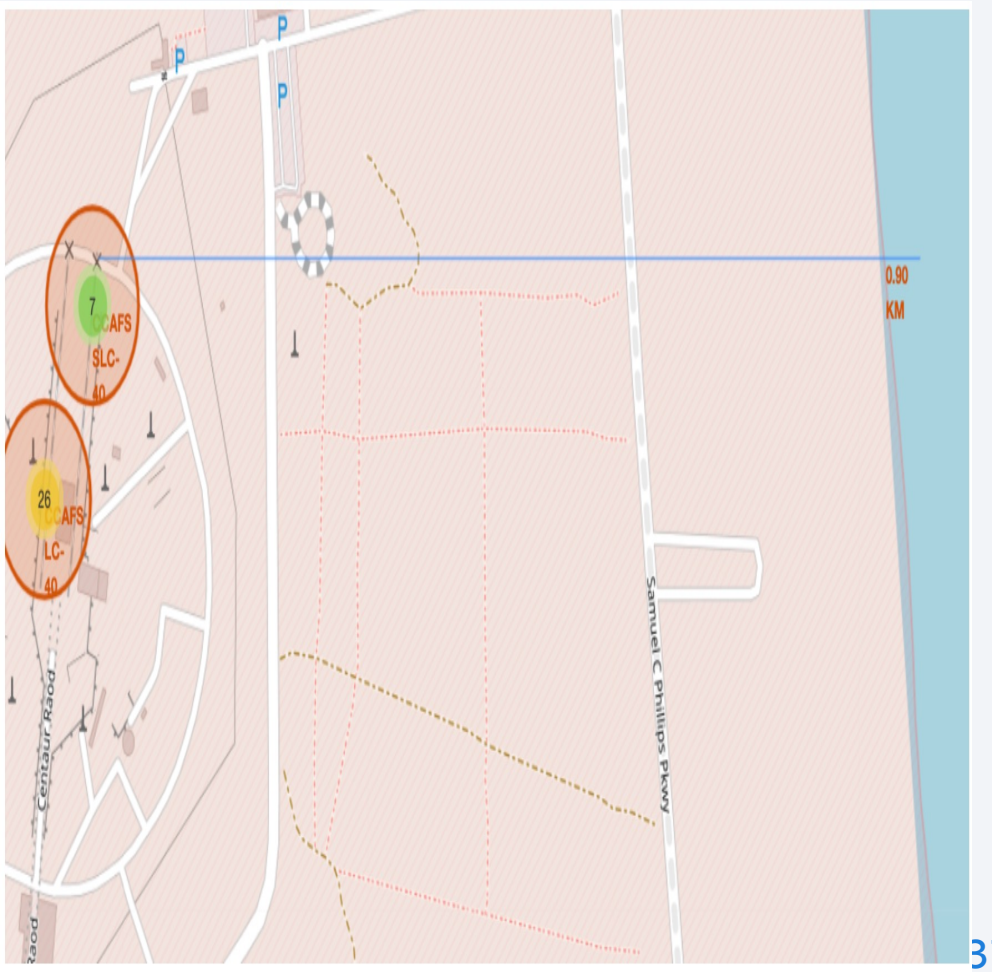
After you plot distance lines to the proximities, you can answer

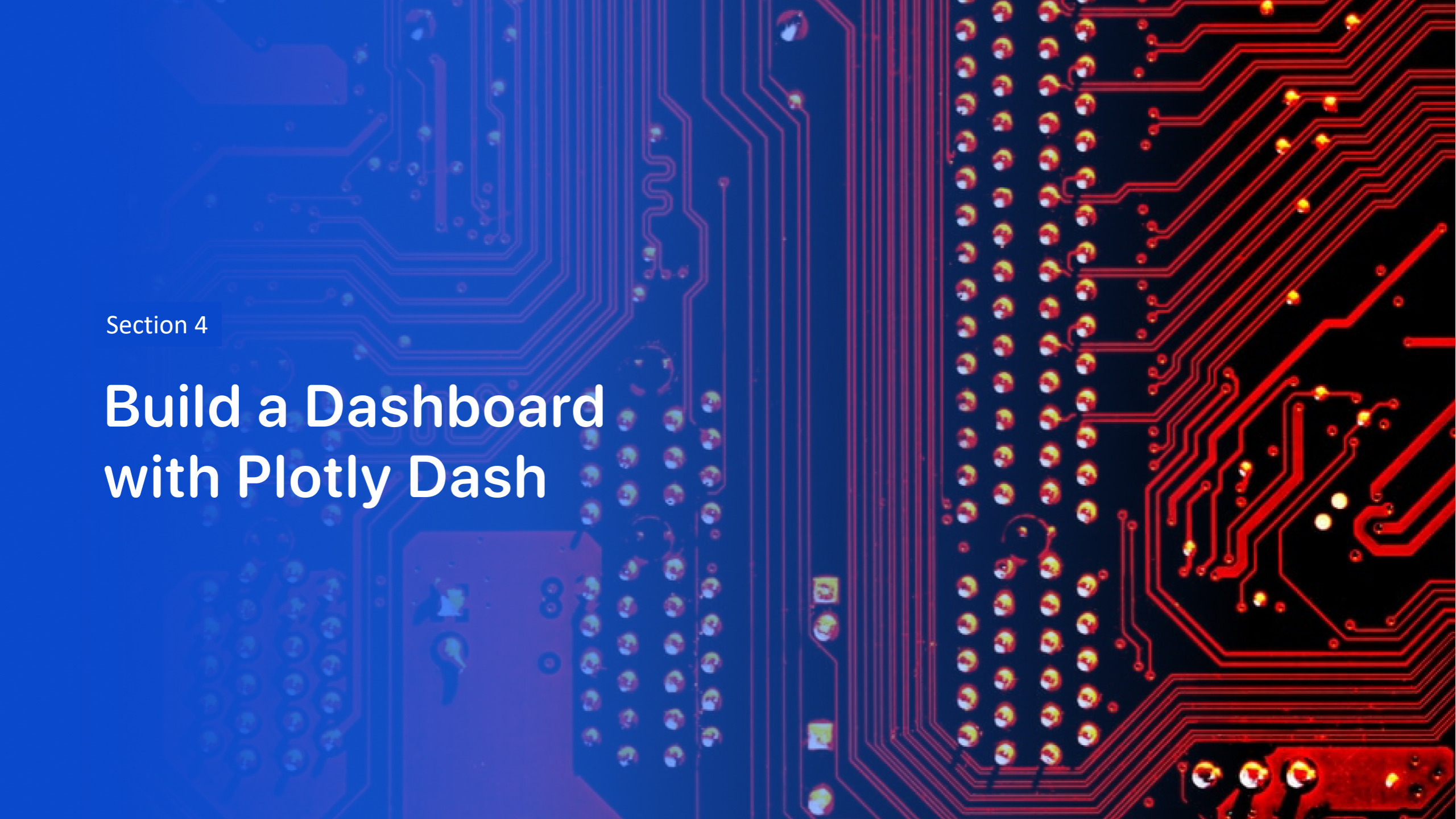
- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? No
- Do launch sites keep certain distance away from cities? yes

A railway map symbol may look like this:

A highway map symbol may look like this:

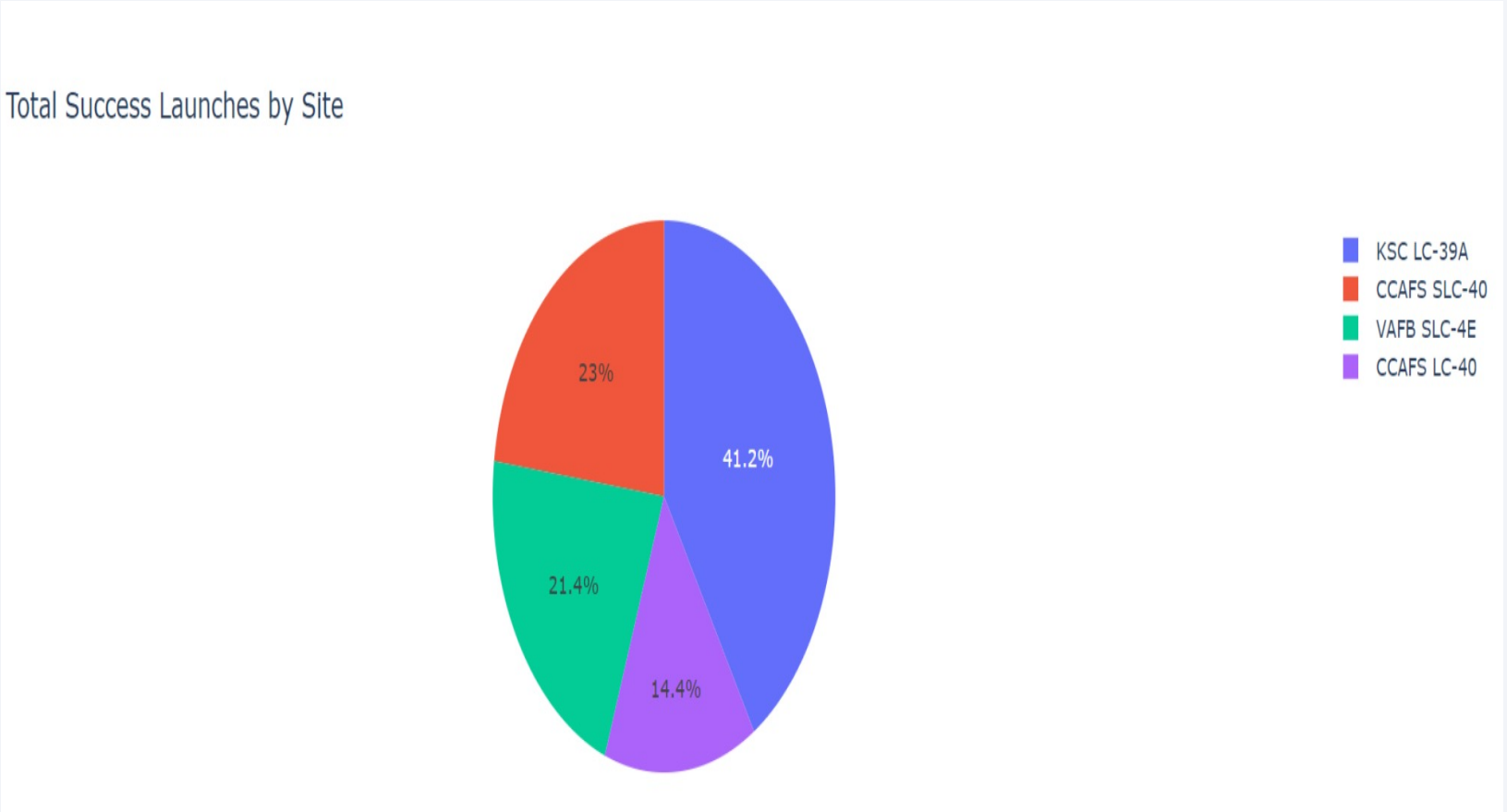
A city map symbol may look like this:




Section 4
Build a Dashboard with Plotly Dash

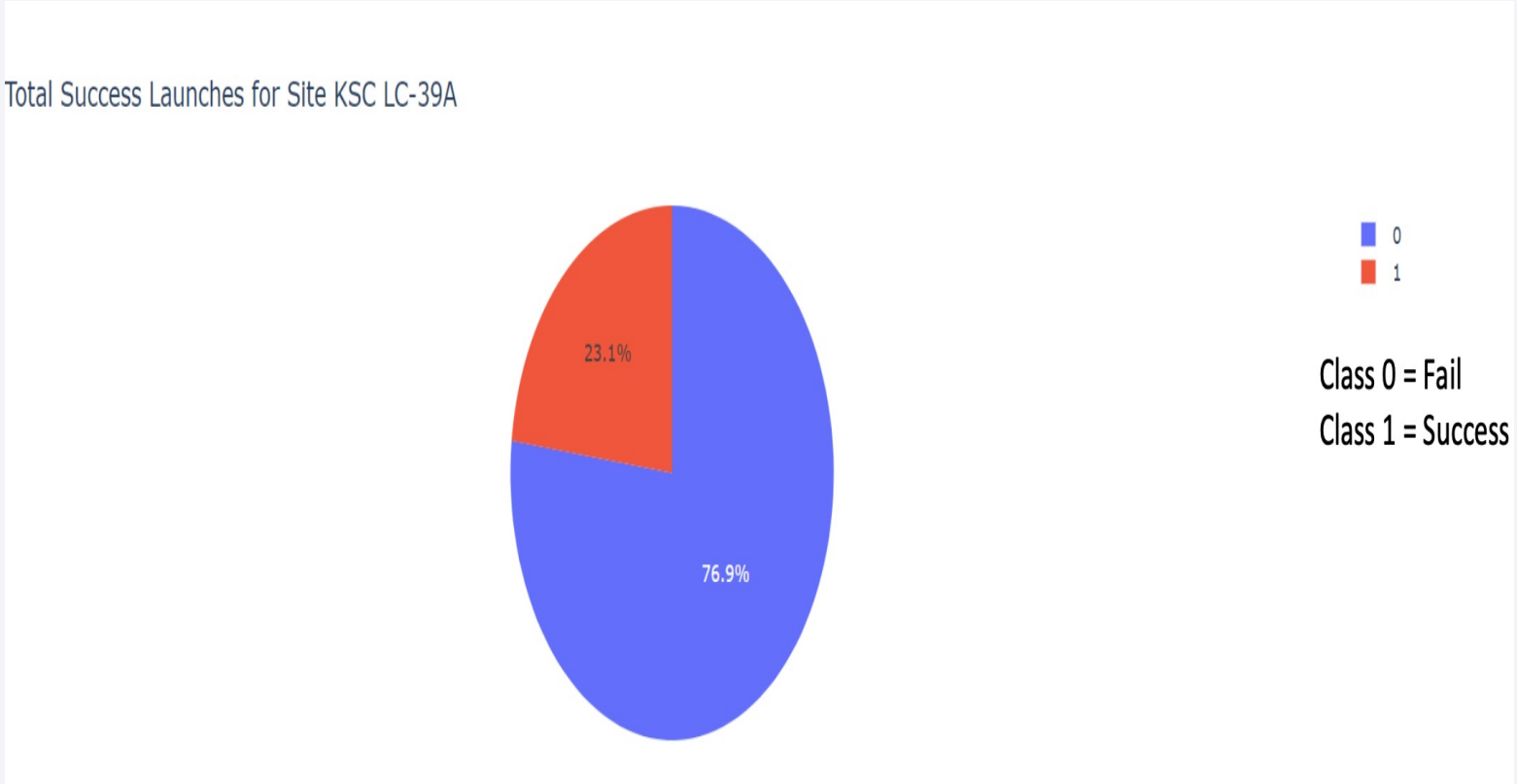
Pie chart of the Successful launches of Each Sites

- KSC-LC-39A had the most successful launches from all the sites



Ratio Launch Success for KSC LC-39A

- The piechart shows the launch sites KSC LC-39A has 76.9% launch success rate and 23.1% fail rate launch



Scatter Plot of Payload vs Launch Outcome

- We observe that success rate for lower mass payloads is higher than the heavy mass payloads

Section 5

Predictive Analysis (Classification)

Classification Accuracy

- Among KNN, Decision tree, Logistic Regression and SVM: the best classifier for the dataset is the Decision tree model with the highest accuracy score of 0.891071

```
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_)
```

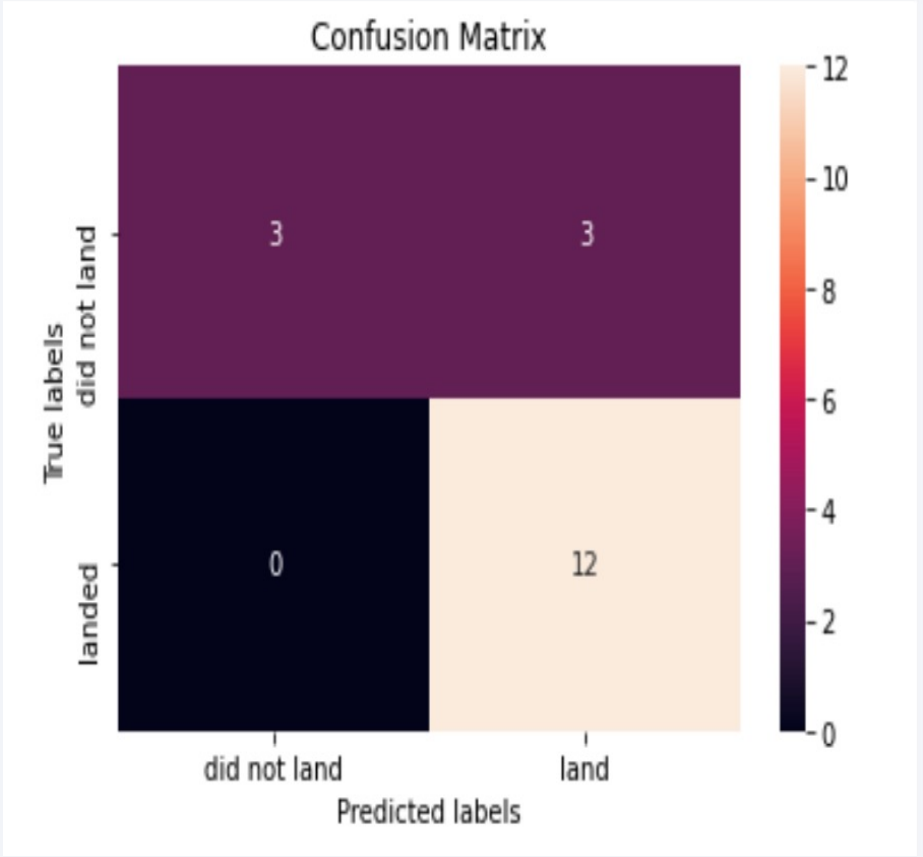
```
Best model is DecisionTree with a score of 0.8910714285714286
Best params is : {'criterion': 'entropy', 'max_depth': 8, 'max_features': 'sqrt', 'min_samp
```

```
score_df1 = pd.DataFrame.from_dict(models, orient='index', columns=['Test Data Accuracy'])
score_df1.sort_values(['Test Data Accuracy'], inplace=True)
score_df1 = score_df1.reset_index()
score_df1.rename(columns = {'index': 'Algorithm'}, inplace = True)
score_df1.head(4)
```

	Algorithm	Test Data Accuracy
0	LogisticRegression	0.846429
1	SupportVector	0.848214
2	KNeighbors	0.848214
3	DecisionTree	0.891071

Confusion Matrix

- The confusion matrix summarizes the performance of a classification algorithm
- All the confusion matrices were identical of all the classifiers
- There are false positives (Type 1 error) which is not good
  - 12 True positive
  - 3 True negative
  - 3 False positive
  - 0 False Negative



Conclusions

- All models performed similarly on the test set with the decision tree model slightly outperformed
- All the launch sites are close to the coast
- Launch success increases over time from 2013
- KSC LC-39A has the highest success rate among launch sites and 100% success rate for launches less than 5500kg
- At the launch sites, the higher the payload mass(kg), the higher success rate
- ES-L1, GEO, HEO, and SSO have a 100% success rate

Appendix

- The python code of all steps followed can be found here on github link given in the slides

Thank you!