IBM Developer
SKILLS NETWORK

IBM

Data Science Capstone Project

Muhammad Alfiandri Adin

22nd January 2022

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection through API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Data Visualization
  - Interactive Visual Analytics with Folium and Dash
  - Machine Learning Prediction
- Summary of all results
  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - Predictive Analytics result

Executive Summary

- SpaceX have a successful recoveries that generally have the following properties:
  - Having light payload (lesser than 8000kg)
  - Launched from site KSC LC-39A
  - Most successful recovery used drone ship
  - Launch date in the year 2016 or later (having success rate above 0.6)
  - Used orbit GEO, HEO, SSO, ES-L1
- Using machine learning algorithm we predict the outcome of a given recovery with a reasonable degree of accuracy, 83.33%
- The machine learning algorithm best used for prediction model is **Decision Tree**

Introduction

- Project background and context

  Most space exploration companies may spend up to $165 million to launch a single rocket. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars. Much of the savings is because Space X can reuse the first stage. Therefore, if we can determine if the first stage will land successfully, this information can be used if a competing company wants to bid against SpaceX for a rocket launch.

- Problems you want to find answers:

  - What conditions which will aid SpaceX to achieve the best result in launching rocket?
  - What type of machine learning algorithm suited best for predicting the success of future rocket launch by SpaceX?

Section 1

Methodology

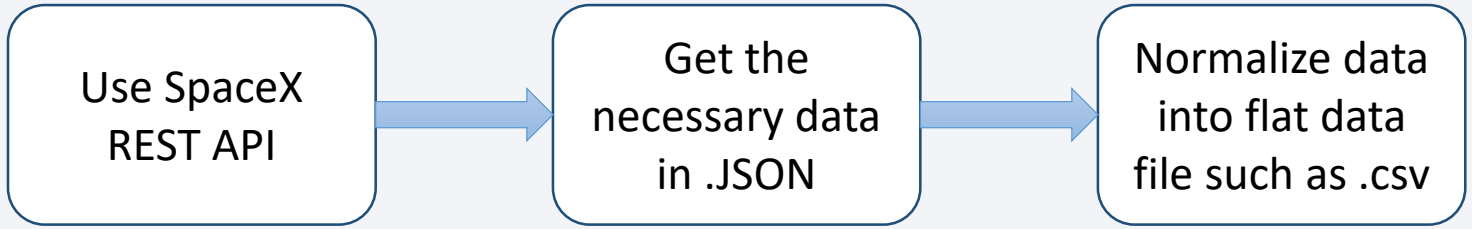
Methodology

- Data collection
  - Data was collected using SpaceX API and web scraping from Wikipedia.
- Perform data wrangling
  - One-hot encoding was applied to categorical features
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Scatter and bar graphs to show patterns between data
- Perform interactive visual analytics
  - Using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Split the dataset into train and test data; build, train, and evaluate various classification model

Data Collection

- The data collection process involved with performing request using SpaceX API to get the necessary .json file and collecting data by using Web Scrapping method from website such as Wikipedia.

SpaceX API



Web Scrapping

Data Collection – SpaceX API

- The following steps in performing data collection using SpaceX API:
  1. Getting response from API
  2. Converting response to a .JSON file
  3. Clean and filter data uasing custom function
  4. Assign dictionary then create the dataframe
  5. Filter dataframe and export to a flat file (.csv)
- The link to the notebook is Here

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

```
# Call getBoosterVersion
getBoosterVersion(data)
```

```
# Call getLaunchSite
getLaunchSite(data)
```

```
# Call getPayloadData
getPayloadData(data)
```

```
# Call getCoreData
getCoreData(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}
```

```
# Create a data from launch_dict
falcon9 = pd.DataFrame(launch_dict)
```

```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9.to_csv('dataset_part_1_version2.csv', index=False)
```

Data Collection - Scraping

- The following steps in performing data collection using Web Scrapping:
  1. Getting response from HTML
  2. Create BeautifulSoup object
  3. Find all the tables
  4. Get the column names
  5. Create the dictionary
  6. Append the data into the keys
  7. Convert dictionary into dataframe
  8. Export dataframe into a flat file (.csv)
- The link to the notebook is Here

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

```
# 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')
```

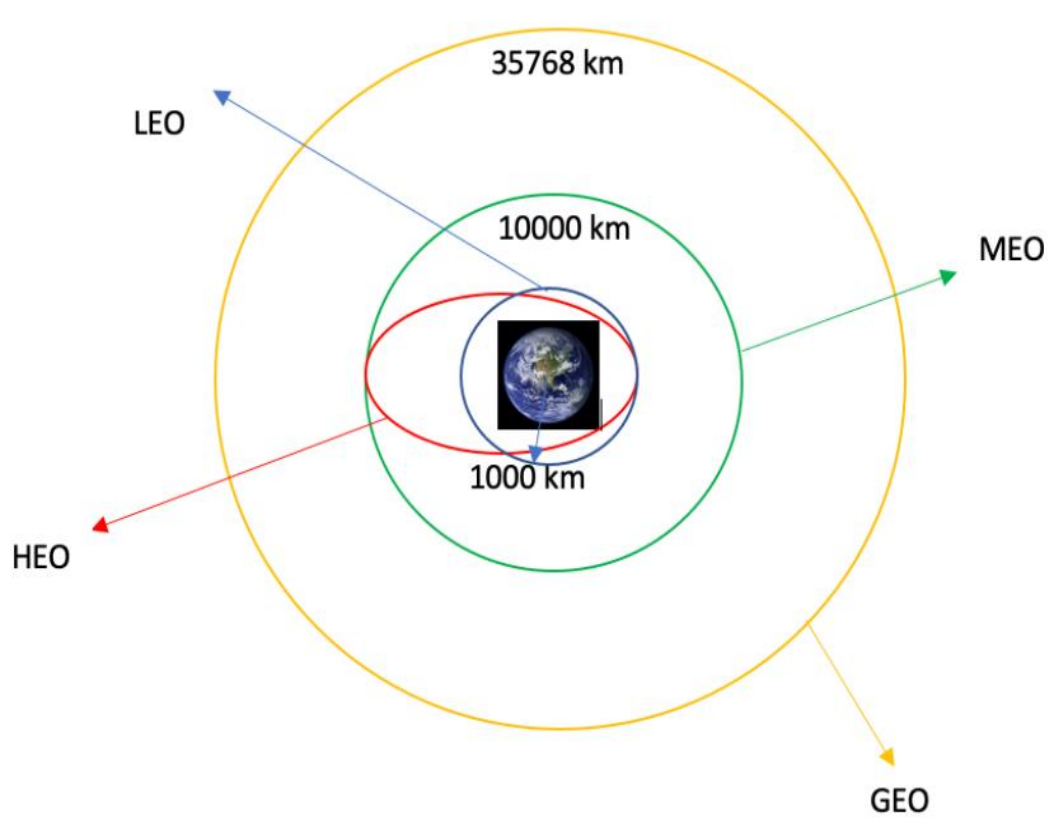
```
column_names = []
test_i = []
for j,i in enumerate(first_launch_table.find_all('th', scope='col')):
    if i != None:
        test_i.append(extract_column_from_header(i))
    if test_i[j] != None and len(test_i[j]) > 0:
        column_names.append(test_i[j])
```

```
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']=[]
```

```
df = pd.DataFrame.from_dict(launch_dict, orient='index')
```

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

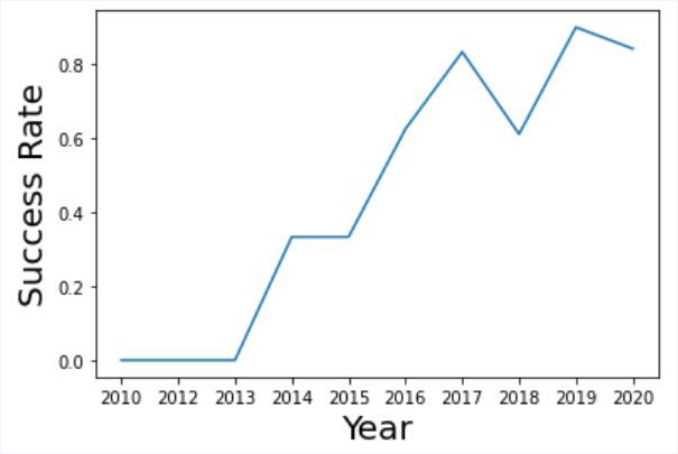
Data Wrangling

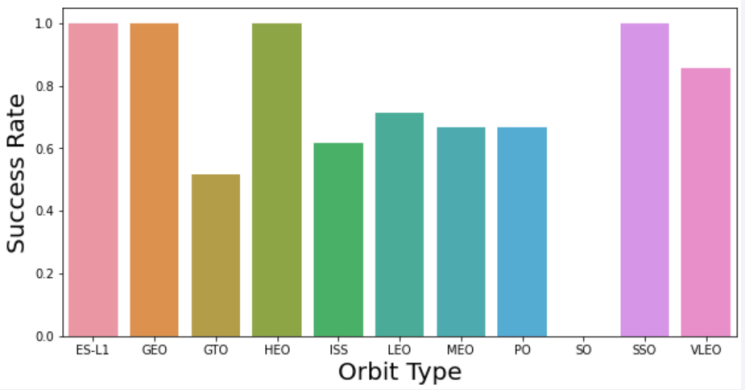


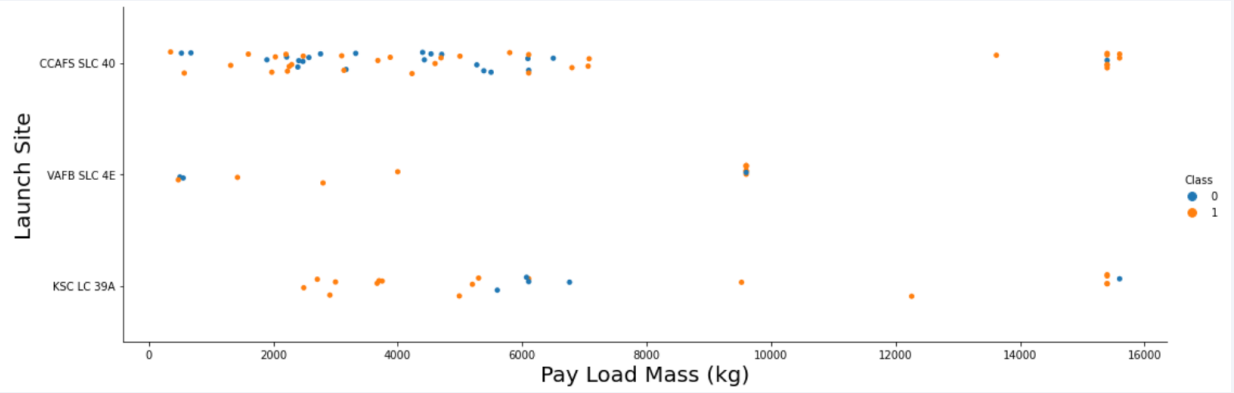
- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the results to csv.
- The link to the notebook is Here

EDA with Data Visualization

- We explored the data by visualizing the relationship between different variables:
  - Flight Number and Launch Site
  - Payload and Launch Site
  - Success Rate of each orbit ype
  - Flight Number and Orbit type
  - Payload and Orbit type
- We use line plot, scatter plot, cat plot, and bar plot to visualize the relationships between variables
- The link to the notebook is Here







EDA with SQL

- EDA with SQL uses open-source **PostgreSQL** because the trial period for IBM db2 is already over and the provided link by EdX to get a trial extension code does not work.
- We applied EDA with SQL to get insight from the data. We wrote queries to:
  - Display the names of the unique launch sites in the space mission
  - Display 5 records where launch sites begin with the string 'KSC‘
  - 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 where the first succesful landing outcome in drone ship was achieved
  - List the names of the boosters which have success in ground pad 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, succesful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017
  - Rank the count of successful landing_outcomes between the date 2010-06-04 and 2017-03-20 in descending order
- The link to the notebook is Here

Build an Interactive Map with Folium

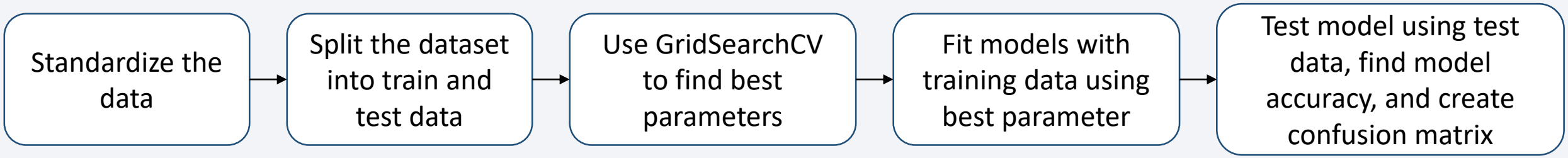
- We marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- We assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- We calculated the distances between a launch site to its proximities.
- The link to the notebook is Here

Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash and run it live with a Flask app using www.pythonanywhere.com
- We plotted pie charts showing the total launches by a certain sites
- We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- The link to the dash app github repo is Here
- The link to the live interactive site using www.pythonanywhere.com is Here

Predictive Analysis (Classification)

- The machine learning algorithm tested are:
    1. Logarithmic Regression
    2. Support Vector Machine
    3. Decision Tree
    4. K-Nearest Neighbor
- The link to the notebook is Here

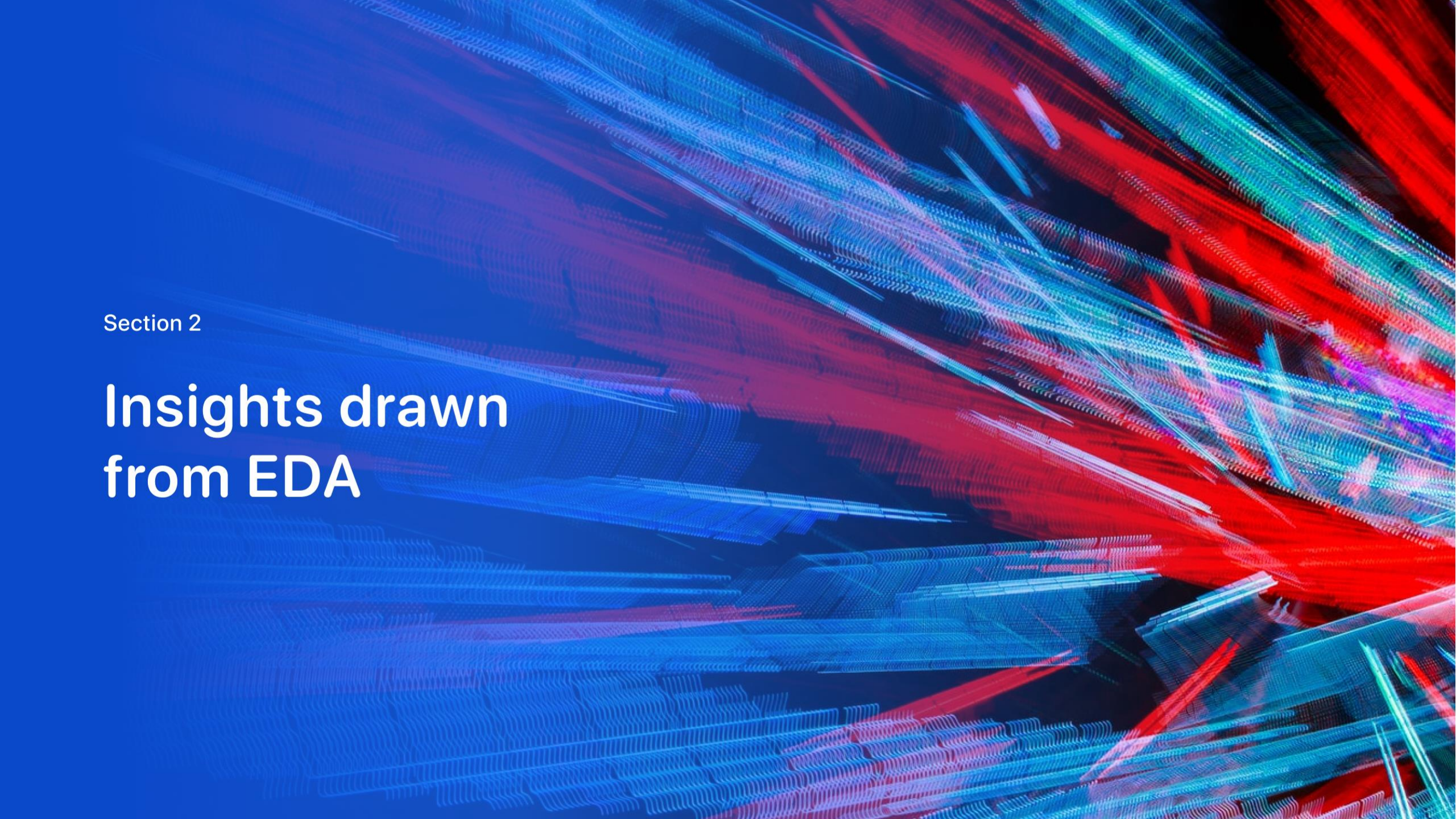


Results

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

Section 2

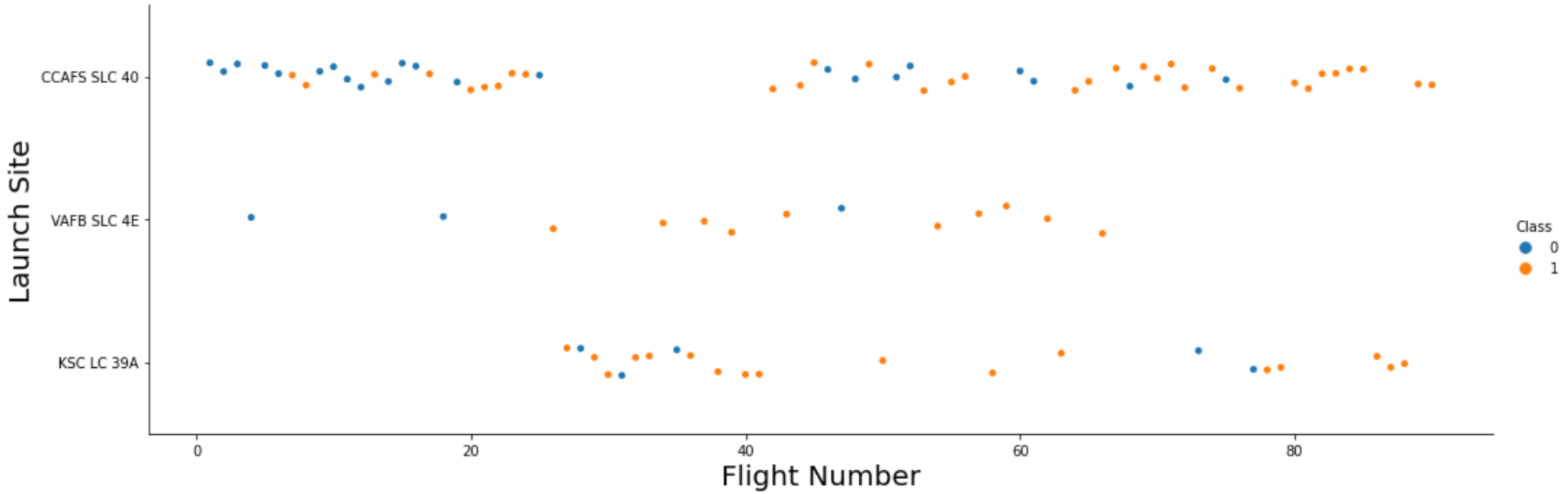
Insights drawn from EDA



EDA with Visualization

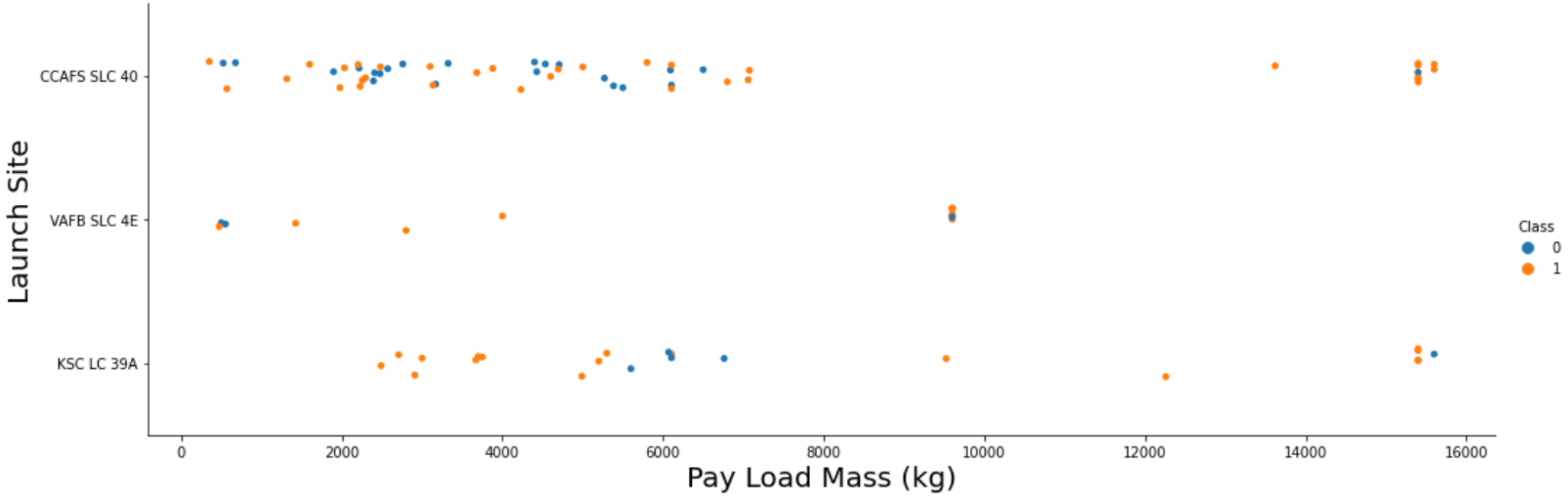
Flight Number vs. Launch Site

- From the plot, we found that the more flights performed at a launch site the greater the success rate for that launch site



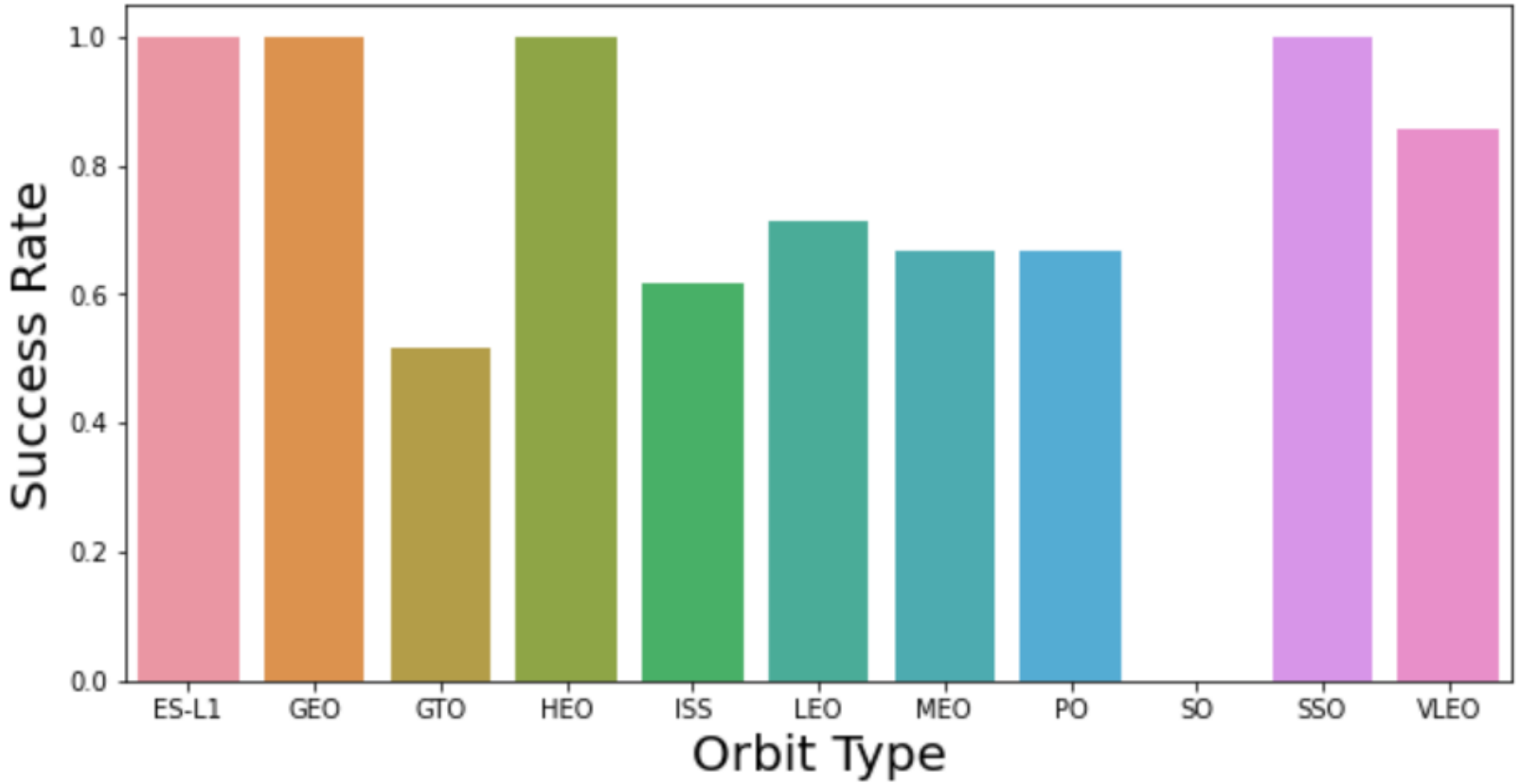
Payload vs. Launch Site

- From the plot, we found that flights with payload lesser than 8000kg have higher success rate. At VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).



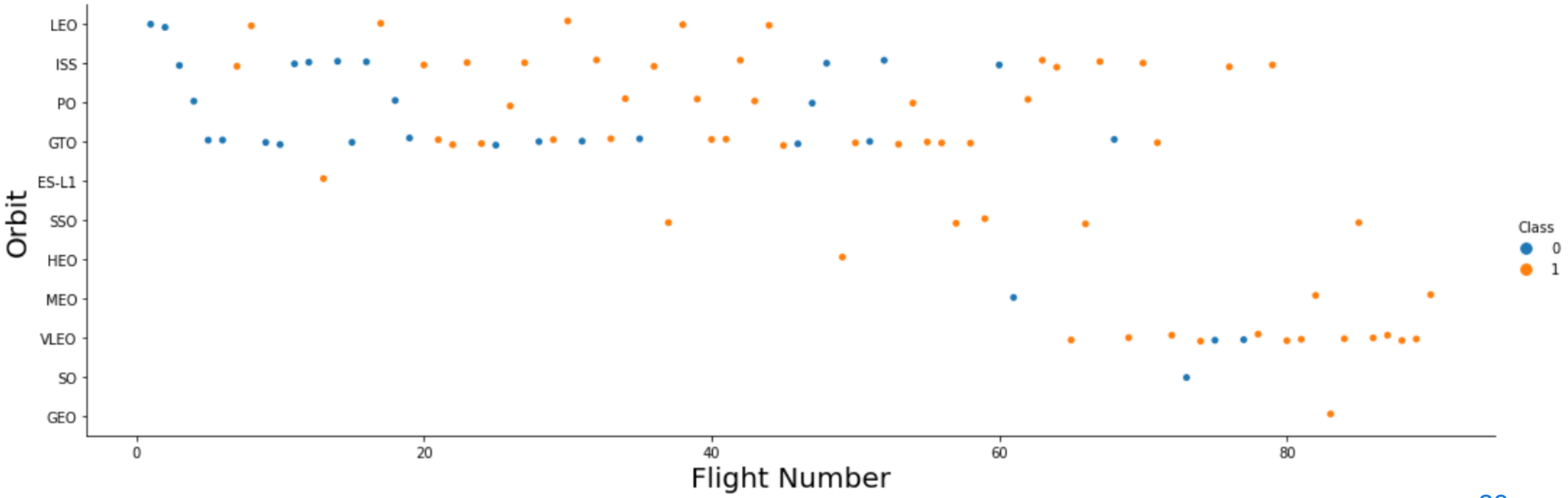
Success Rate vs. Orbit type

- From the plot, we found that orbit ES-L1, GEO, HEO, and SSO has the best success rate compared to the other orbit



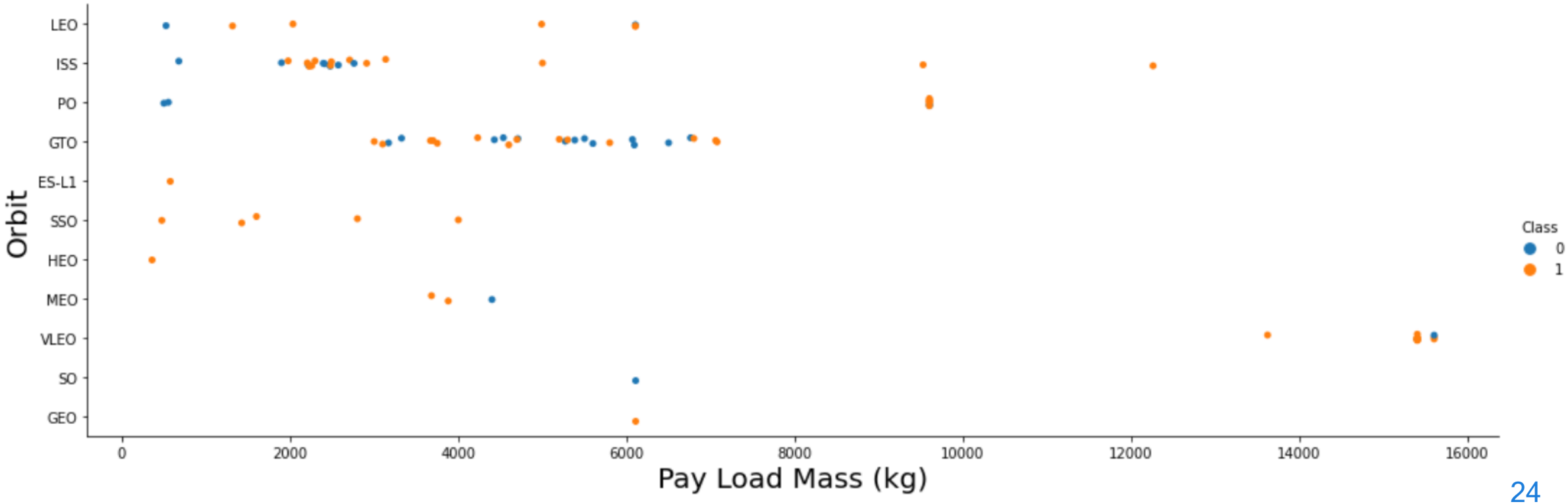
Flight Number vs. Orbit Type

- From the plot below, we see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.



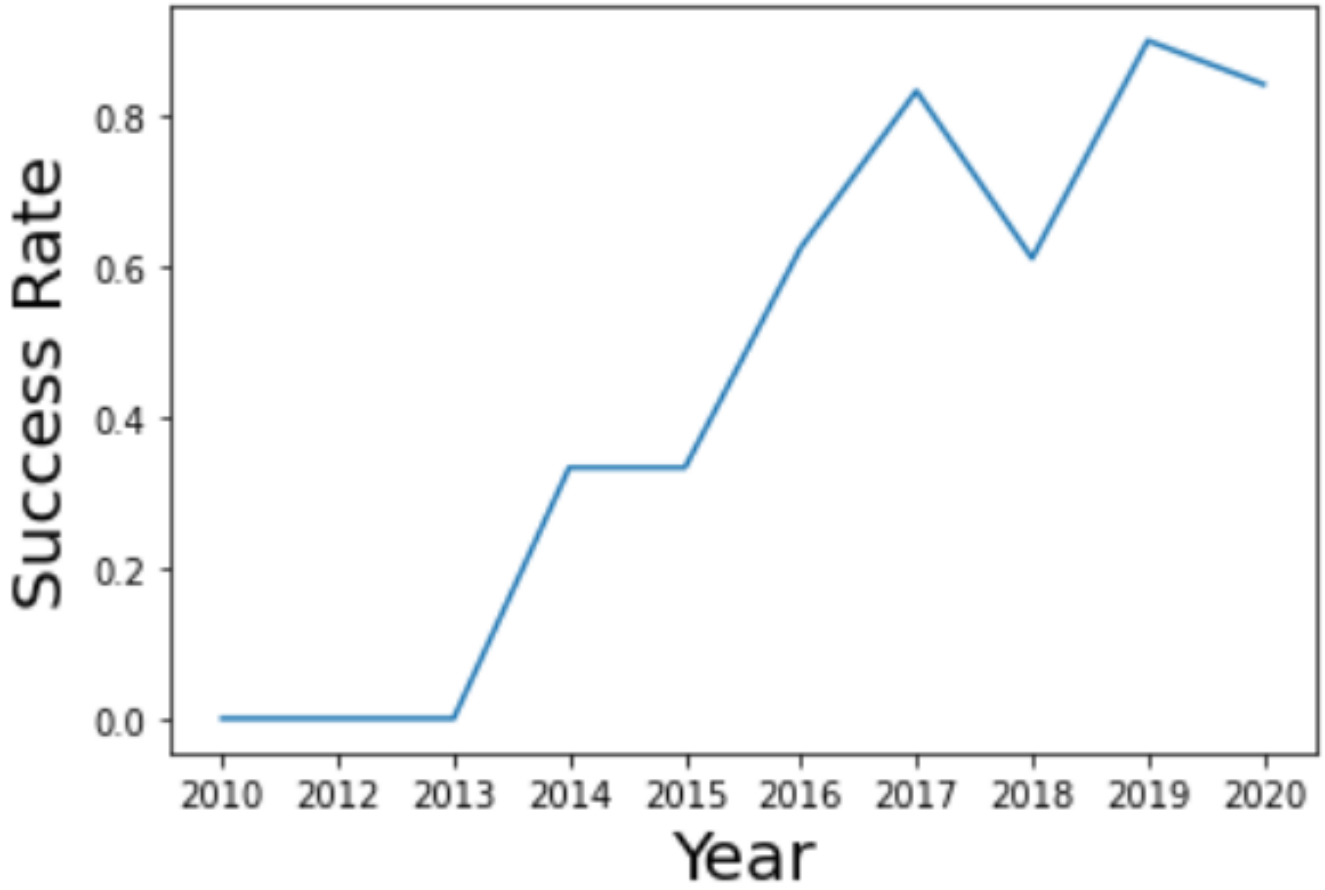
Payload vs. Orbit Type

- From the plot below, we see that With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS.
- However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.



Launch Success Yearly Trend

- From the plot below, we see that the success rate since 2013 kept increasing till 2020. We can also see that since after 2016, the yearly launch success rate has been consistently above 0.5.



EDA with SQL

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

SQL Query

```
task_1 = '''
        SELECT DISTINCT LaunchSite
        FROM SpaceX
'''
create_pandas_df(task_1, database=conn)
```

	launchsite
0	CCAFS SLC-40
1	KSC LC-39A
2	CCAFS LC-40
3	VAFB SLC-4E

Description

Using the **DISTINCT** word in the query, we pull unique values for **LaunchSite** from table **SpaceX**

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

SQL Query

```
task_2 = '''
        SELECT *
        FROM SpaceX
        WHERE LaunchSite LIKE 'KSC%'
        LIMIT 5
        '''
create_pandas_df(task_2, database=conn)
```

	date	time	boosterversion	launchsite	payload	payloadmasskg	orbit	customer	missionoutcome	landingoutcome
0	2017-02-19	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
1	2017-03-16	06:00:00	F9 FT B1030	KSC LC-39A	EchoStar 23	5600	GTO	EchoStar	Success	No attempt
2	2017-03-30	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300	GTO	SES	Success	Success (drone ship)
3	2017-05-01	11:15:00	F9 FT B1032.1	KSC LC-39A	NROL-76	5300	LEO	NRO	Success	Success (ground pad)
4	2017-05-15	23:21:00	F9 FT B1034	KSC LC-39A	Inmarsat-5 F4	6070	GTO	Inmarsat	Success	No attempt

Description

Using keyword **LIMIT** 5 in the query, we fetch only 5 records from table **SpaceX** with the condition **LIKE** and with wild card **'KSC%'**. The percentage symbol at the end suggests that the **LaunchSite** name must start with KSC.

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

SQL Query

```
task_3 = '''
        SELECT SUM(PayloadMassKG) AS Total_PayloadMass
        FROM SpaceX
        WHERE Customer LIKE 'NASA (CRS)'
        '''
create_pandas_df(task_3, database=conn)
```

	total_payloadmass
0	45596

Description

Using the function **SUM** calculates the total in column **PayloadMassKG** and the **WHERE** clause filters the data so that it only fetch **Customer** by name **'NASA (CRS)'**

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

SQL Query

```
task_4 = '''
        SELECT AVG(PayloadMassKG) AS Avg_PayloadMass
        FROM SpaceX
        WHERE BoosterVersion = 'F9 v1.1'
        '''
create_pandas_df(task_4, database=conn)
```

	avg_payloadmass
0	2928.4

Description

Using the function **AVG** works out the average in the column **PayloadMassKG**. The **WHERE** clause filters the dataset to only perform calculations **on BoosterVersion 'F9 v1.1'**

List the date where the first succesful landing outcome in drone ship was acheived

SQL Query

```
task_5 = '''
        SELECT MIN(Date) AS FirstSuccessfull_landing_date
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Success (drone ship)'
        '''
create_pandas_df(task_5, database=conn)
```

	firstsuccessfull_landing_date
0	2016-04-08

Description

Using the function **MIN** works out the minimum date in the column **Date**. The **WHERE** clause filters the dataset to only perform calculations on **LandingOutcome** with conditions **LIKE 'Success (drone ship)'**

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

SQL Query

```
task_6 = '''
        SELECT BoosterVersion
        FROM SpaceX
        WHERE LandingOutcome = 'Success (ground pad)'
            AND PayloadMassKG > 4000
            AND PayloadMassKG < 6000
        '''
create_pandas_df(task_6, database=conn)
```

	boosterversion
0	F9 FT B1032.1
1	F9 B4 B1040.1
2	F9 B4 B1043.1

Description

Selecting only **Booster_Version** . The **WHERE** clause filters the dataset to **Landing_Outcome = 'Success (ground pad)'** . The **AND** clause specifies additional filter conditions **PayloadMassKG**>4000 and **PayloadMassKG<6000**

List the total number of successful and failure mission outcomes

SQL Query

```
task_7a = '''
        SELECT COUNT(MissionOutcome) AS SuccessOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Success%'
        '''

task_7b = '''
        SELECT COUNT(MissionOutcome) AS FailureOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Failure%'
        '''
print('The total number of successful mission outcome is:')
display(create_pandas_df(task_7a, database=conn))
print()
print('The total number of failed mission outcome is:')
create_pandas_df(task_7b, database=conn)
```

```
The total number of successful mission outcome is:
```

	successoutcome
0	100

```
The total number of failed mission outcome is:
```

	failureoutcome
0	1

Description

Create two **SELECT** queries, one to fetch the total number of success and the other for the total number of failure. Use the function **COUNT** to get the number of **MissionOutcome** with **WHERE** clause and **LIKE** condition **'Success%'** and **'Failure%'**

List the total number of successful and failure mission outcomes

SQL Query

```
task_8 = '''
        SELECT BoosterVersion, PayloadMassKG
        FROM SpaceX
        WHERE PayloadMassKG = (
                                SELECT MAX(PayloadMassKG)
                                FROM SpaceX
                                )
        ORDER BY BoosterVersion
        '''
create_pandas_df(task_8, database=conn)
```

	boosterversion	payloadmasskg
0	F9 B5 B1048.4	15600
1	F9 B5 B1048.5	15600
2	F9 B5 B1049.4	15600
3	F9 B5 B1049.5	15600
4	F9 B5 B1049.7	15600
5	F9 B5 B1051.3	15600
6	F9 B5 B1051.4	15600
7	F9 B5 B1051.6	15600
8	F9 B5 B1056.4	15600
9	F9 B5 B1058.3	15600
10	F9 B5 B1060.2	15600
11	F9 B5 B1060.3	15600

Description

Using the word **DISTINCT** in the query means that it will only show Unique values in the **BoosterVersion** column from table **SpaceX**. **ORDER BY** puts the list in order set to a certain condition.

List the records which will display the month names, succesful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017

SQL Query

```
task_9 = '''
        SELECT TO_CHAR(Date, 'Month') AS "Month", LandingOutcome, BoosterVersion, LaunchSite
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Success (ground pad)'
            AND Date BETWEEN '2017-01-01' AND '2017-12-31'
        '''
create_pandas_df(task_9, database=conn)
```

	Month	landingoutcome	boosterversion	launchsite
0	February	Success (ground pad)	F9 FT B1031.1	KSC LC-39A
1	May	Success (ground pad)	F9 FT B1032.1	KSC LC-39A
2	June	Success (ground pad)	F9 FT B1035.1	KSC LC-39A
3	August	Success (ground pad)	F9 B4 B1039.1	KSC LC-39A
4	September	Success (ground pad)	F9 B4 B1040.1	KSC LC-39A
5	December	Success (ground pad)	F9 FT B1035.2	CCAFS SLC-40

Description

The PostgreSQL function **TO_CHAR(expression, format)** returns month name from column **Date**. The **WHERE** clause filters the date so its in the year 2017 with the condition **LIKE 'Success (ground pad)'**

Rank the count of successful landing_outcomes between the date 2010-06-04 and 2017-03-20 in descending order

SQL Query

```
task_10 = '''
        SELECT LandingOutcome, COUNT(LandingOutcome)
        FROM SpaceX
        WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
        GROUP BY LandingOutcome
        ORDER BY COUNT(LandingOutcome) DESC
        '''
create_pandas_df(task_10, database=conn)
```

	landingoutcome	count
0	No attempt	10
1	Failure (drone ship)	5
2	Success (drone ship)	5
3	Success (ground pad)	3
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Failure (parachute)	2
7	Precluded (drone ship)	1

Description

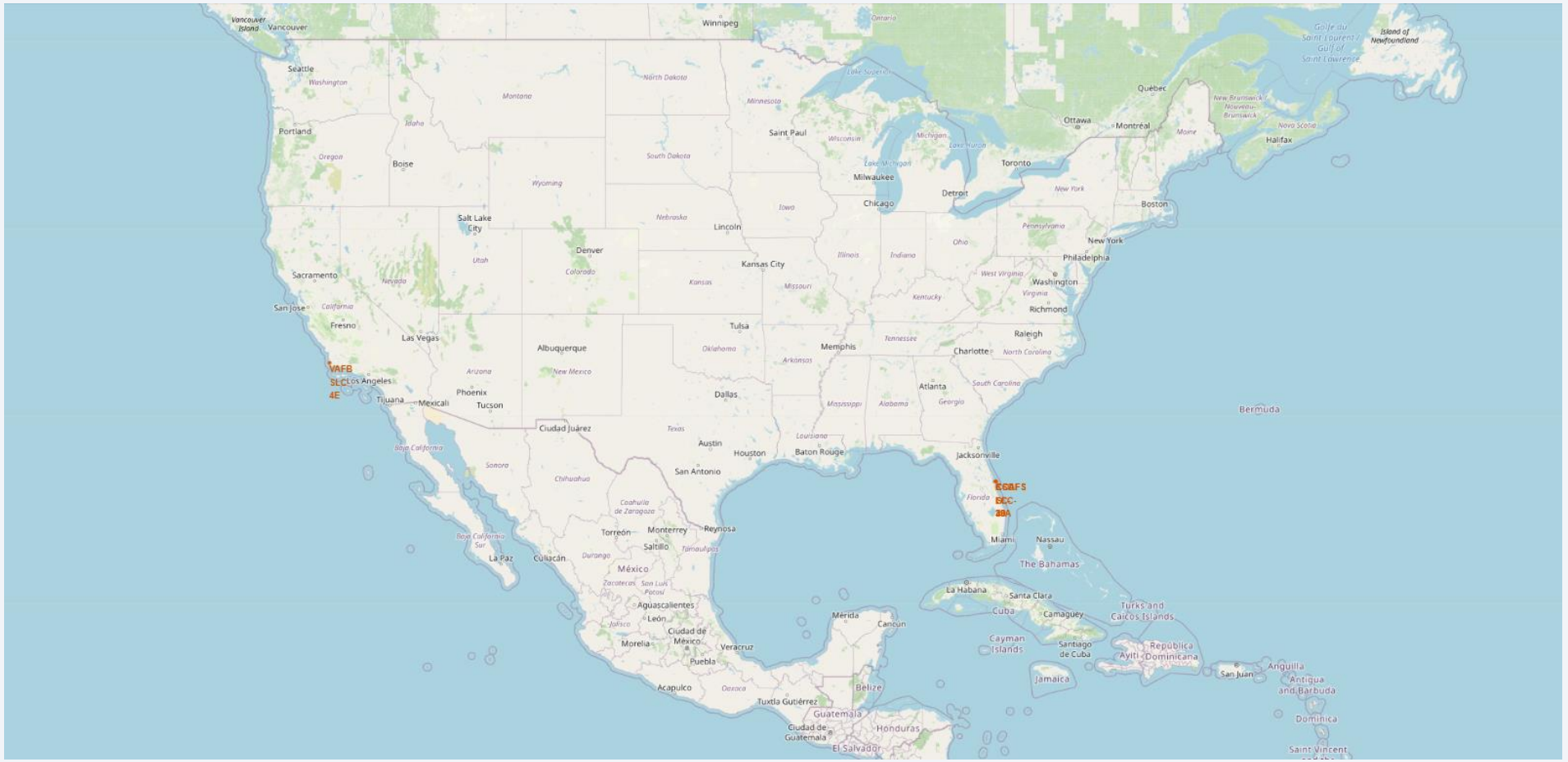
Function **COUNT** counts data in column, clause **WHERE** filters the data with defined conditions

Section 4

Launch Sites Proximities Analysis

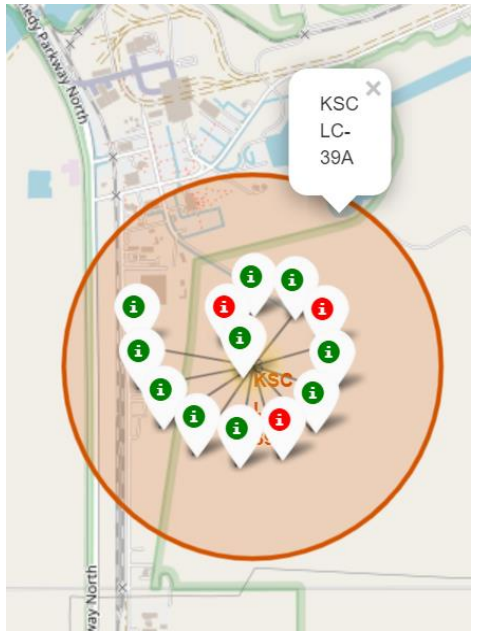
Launch sites global map markers

All of the launch sites are located in US coastlines, specifically in Florida and California.

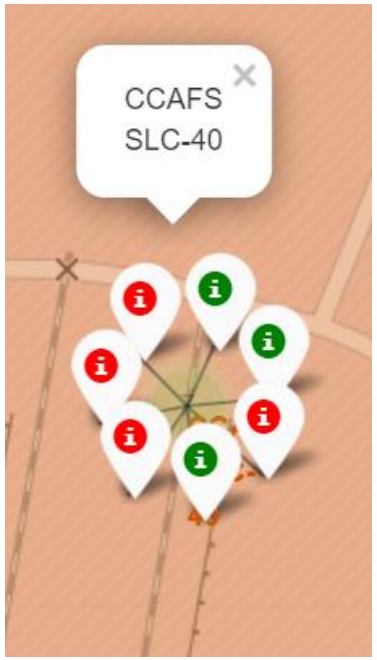


Markers showing launch sites with color labels

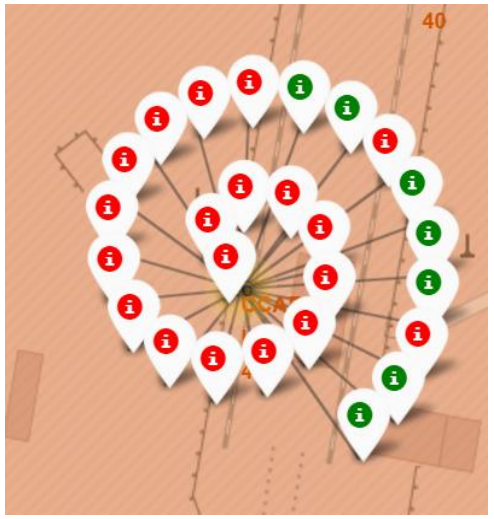
Florida based launch sites



KSC LC-39A

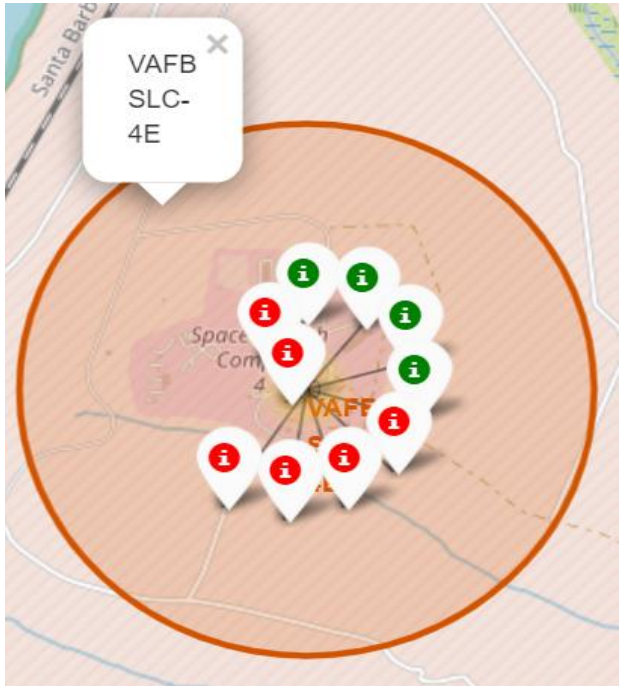


CCAFS SLC-40



CCAFS LC-40

California based launch site



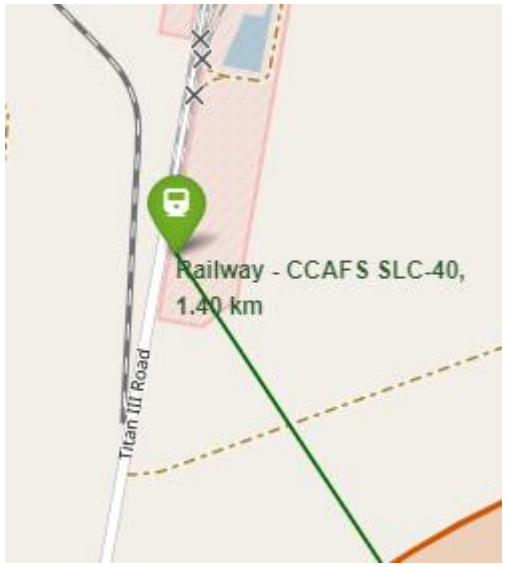
VAFB SLC-4E

- Green marker indicates a successful rocket launch
- Red marker indicates a failed rocket launch

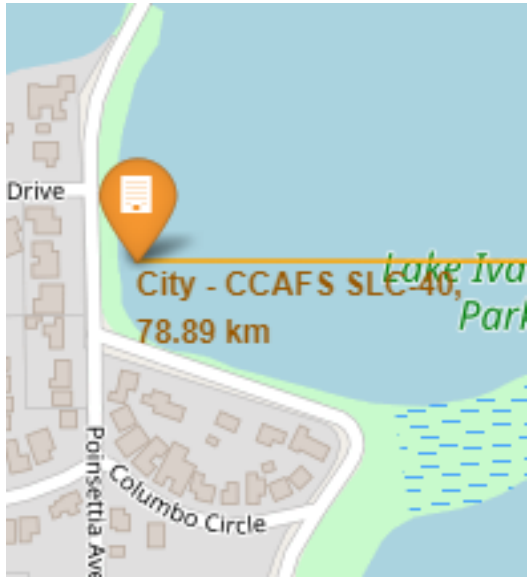
Launch Site distance to landmarks (CCAFS LC-40 and CCAFS SLC-40)

- Both CCAFS LC-40 and CCAFS SLC-40 are located close to each other, therefore their distance to surrounding landmarks can be taken as the same to simplify the analysis.

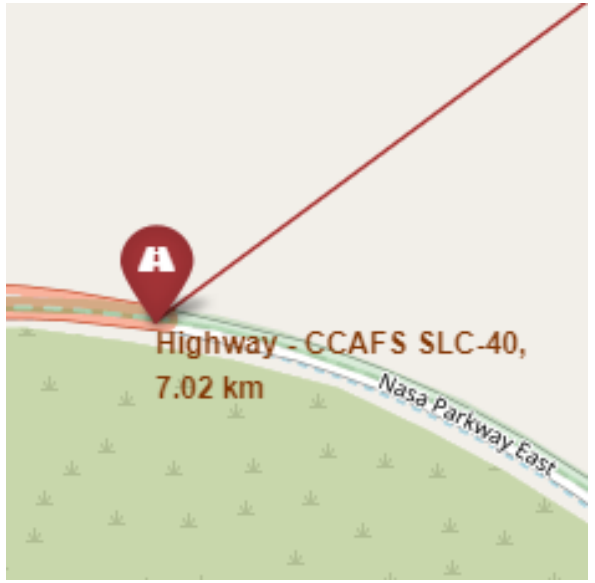
Distance to Railway



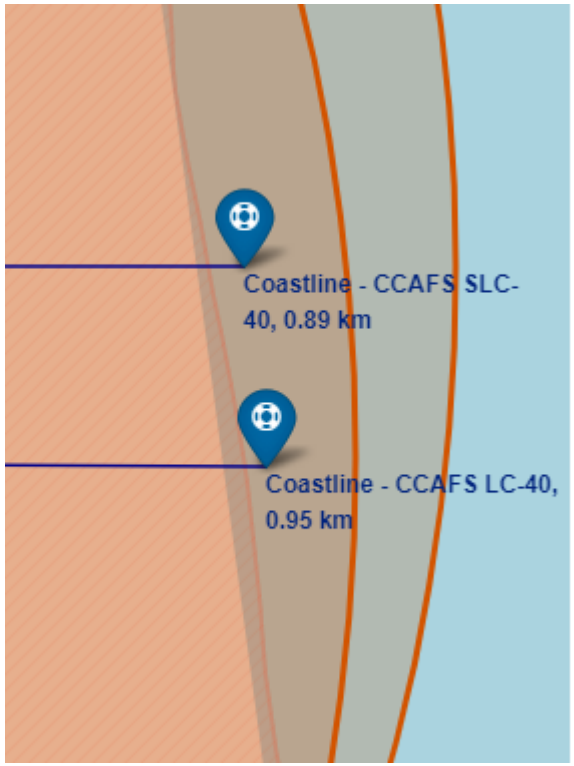
Distance to City



Distance Highway



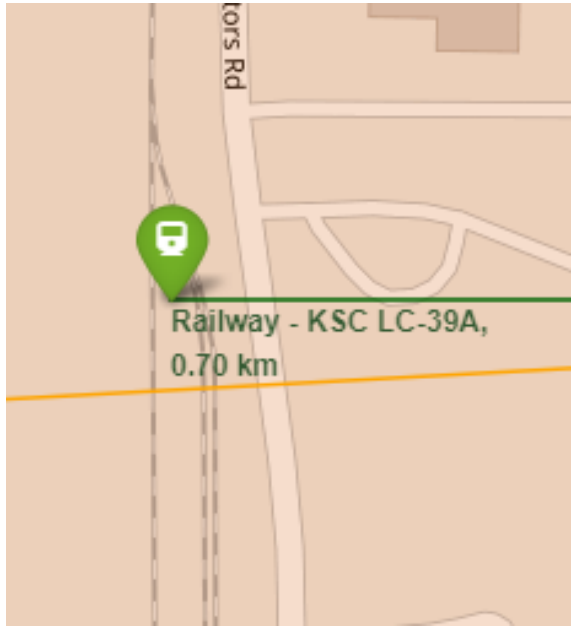
Distance to Coastline



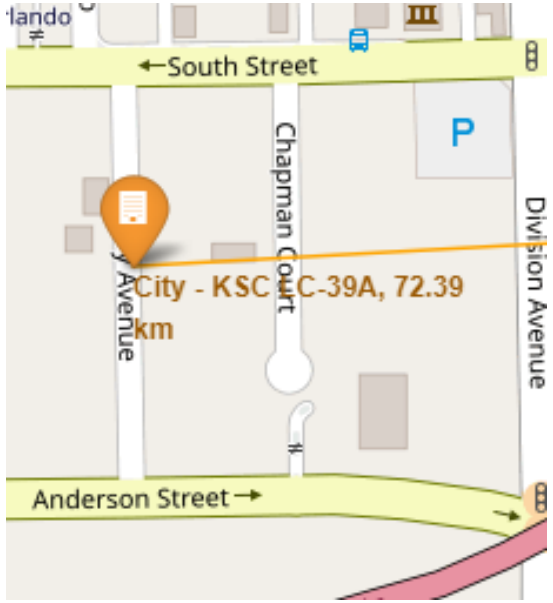
- Distance to nearest railway is 1,4 km
- Distance to nearest city is 78.89 km
- Distance to nearest highway is 7.02 km
- Distance to nearest coastline is 0.89 km

Launch Site distance to landmarks (KSC LC-39A)

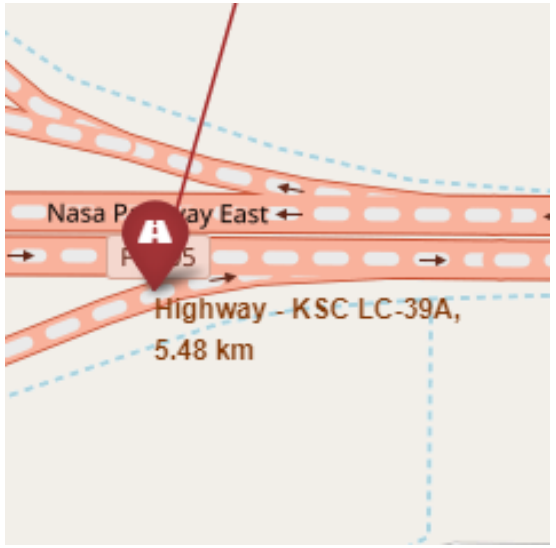
Distance to Railway



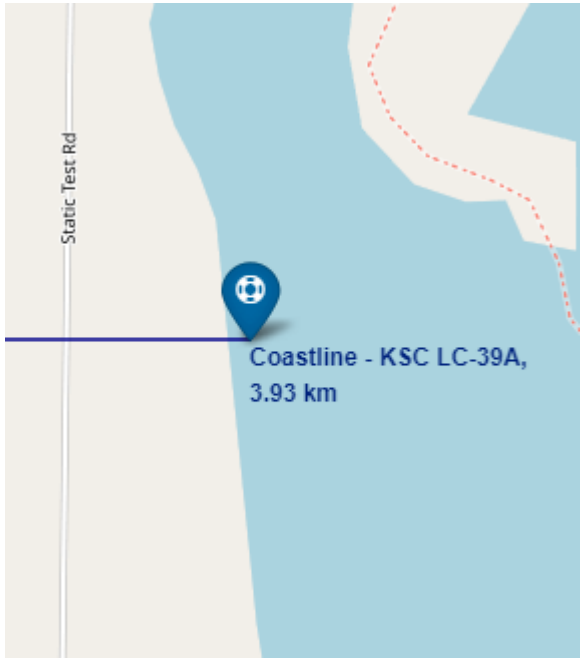
Distance to City



Distance Highway



Distance to Coastline

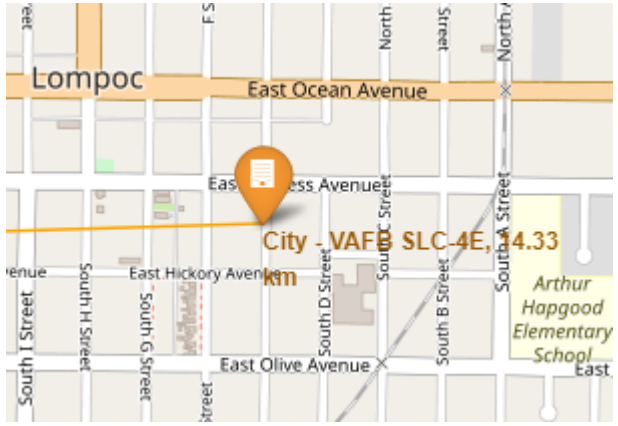


- Distance to nearest railway is 0.7 km
- Distance to nearest city is 72.39 km
- Distance to nearest highway is 5.48 km
- Distance to nearest coastline is 3.93 km

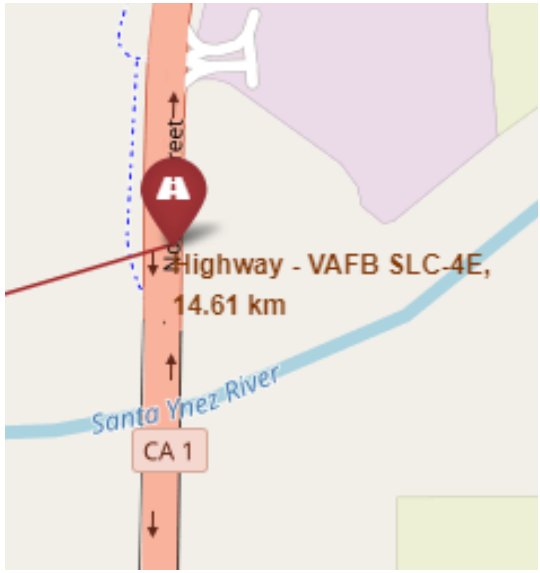
Launch Site distance to landmarks (VAFB SLC-4E)

Distance to Railway

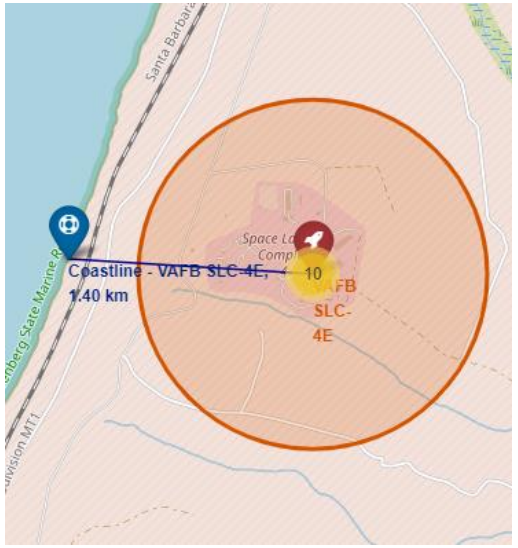
Distance to City



Distance Highway



Distance to Coastline



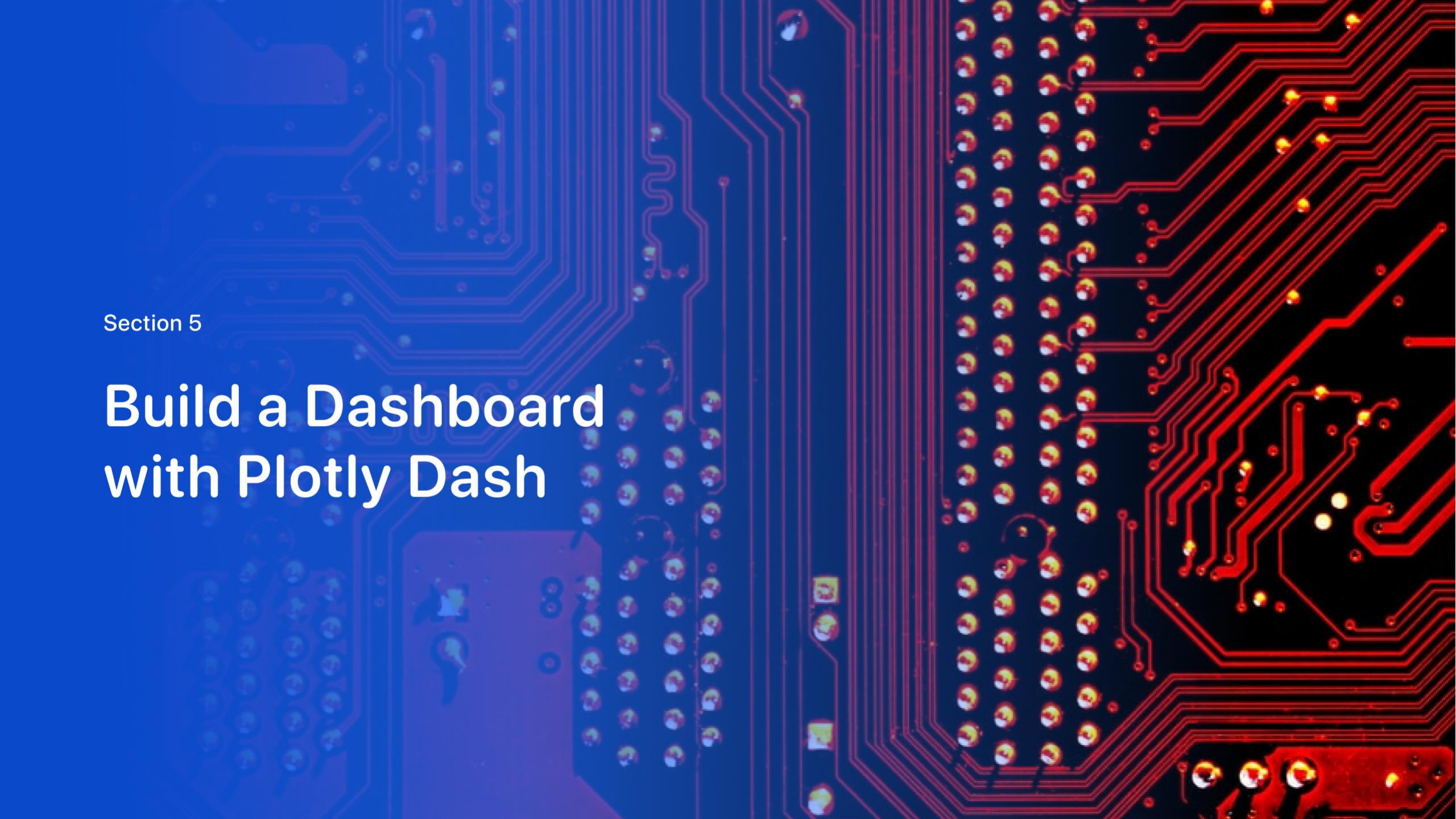
- Distance to nearest railway is 1.38 km
- Distance to nearest city is 14.33 km
- Distance to nearest highway is 14.61 km
- Distance to nearest coastline is 1.4 km

Launch Site distance to landmarks Key Findings

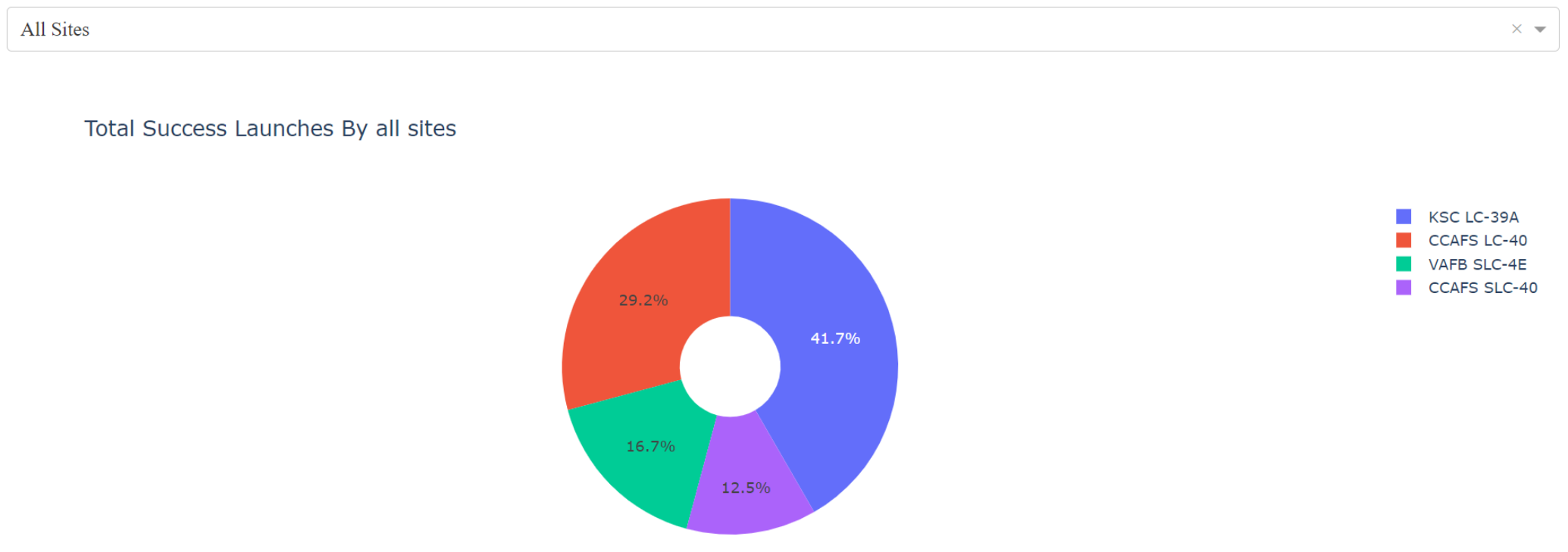
- Are launch sites in close proximity to railways? Yes, because all launch sites have distance to the nearest railway less than 2 km
- Are launch sites in close proximity to highways? No, because the nearest distance of a highway to one of the launch site is greater than 5 km
- Are launch sites in close proximity to coastline? Yes, because all launch sites have distance to the nearest coastline less than 2 km
- Are launch sites keep certain distance away from cities? Yes, because the distance of the nearest city to one of the launch site is greater than 10 km

Section 5

Build a Dashboard with Plotly Dash



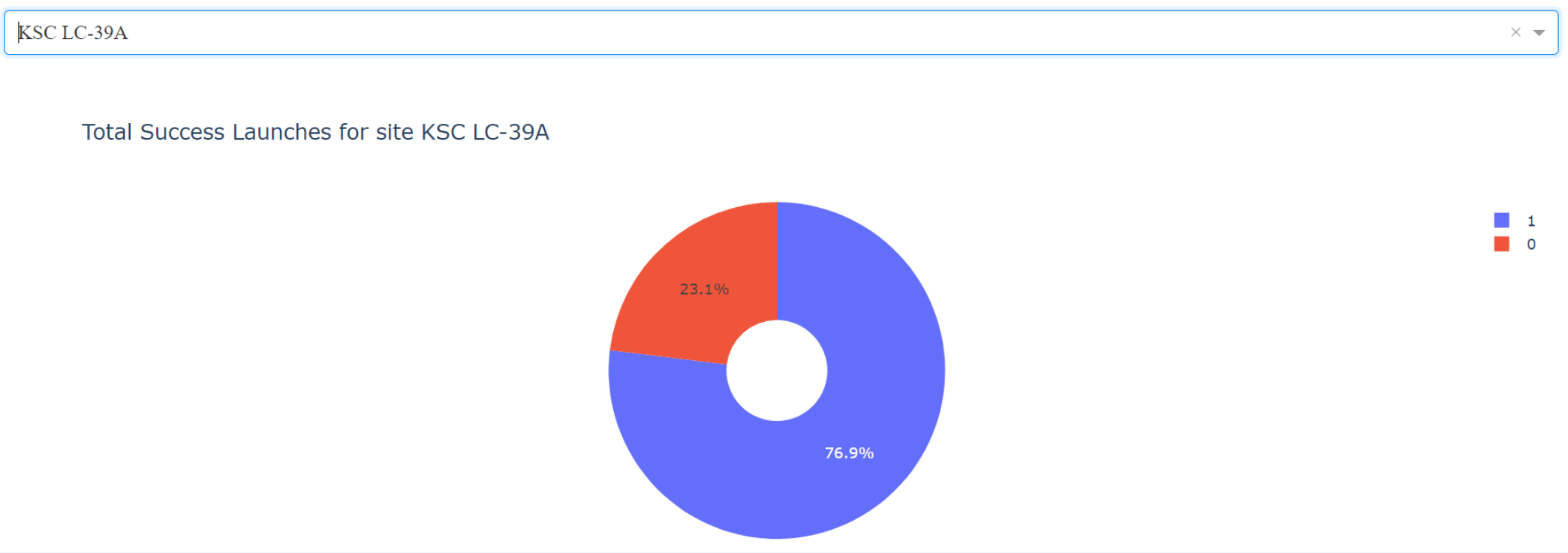
Pie Chart - Total Success Launches By All Sites



- We see that KSC LC-395 is the launch site with highest success rate compared to other launch site

Live Site Code

Pie Chart – Total Success Launches for Site KSC LC-39A

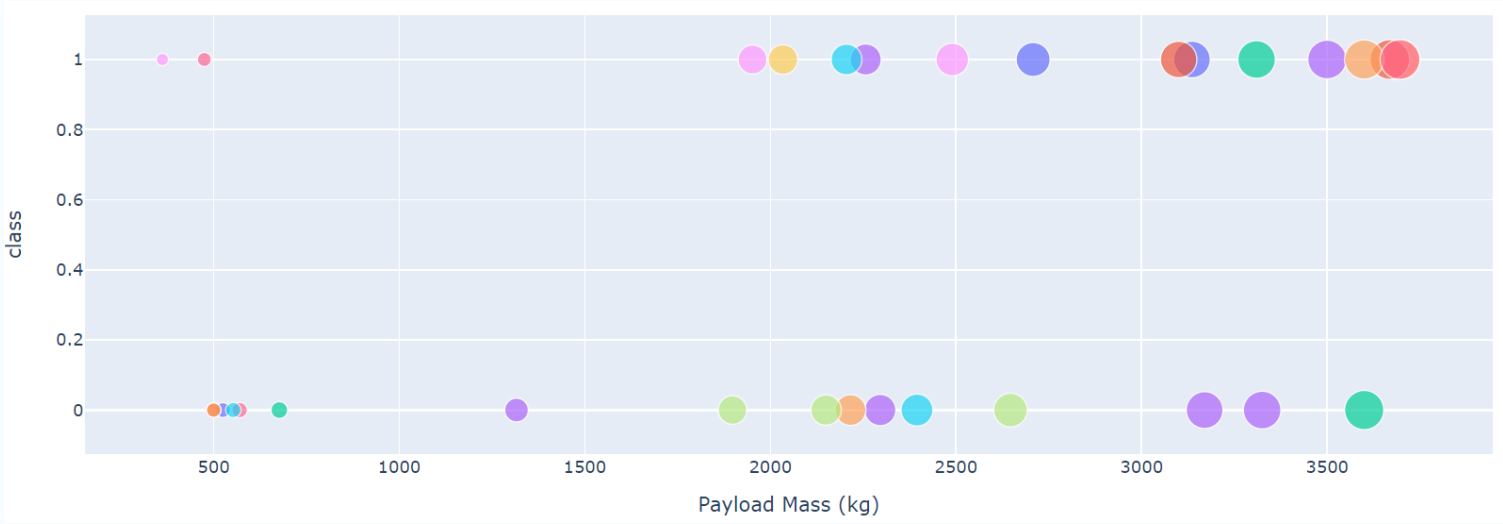


- We see that the launch site KSC LC-395 has a success rate of 76.9% and failure rate of 23.1%

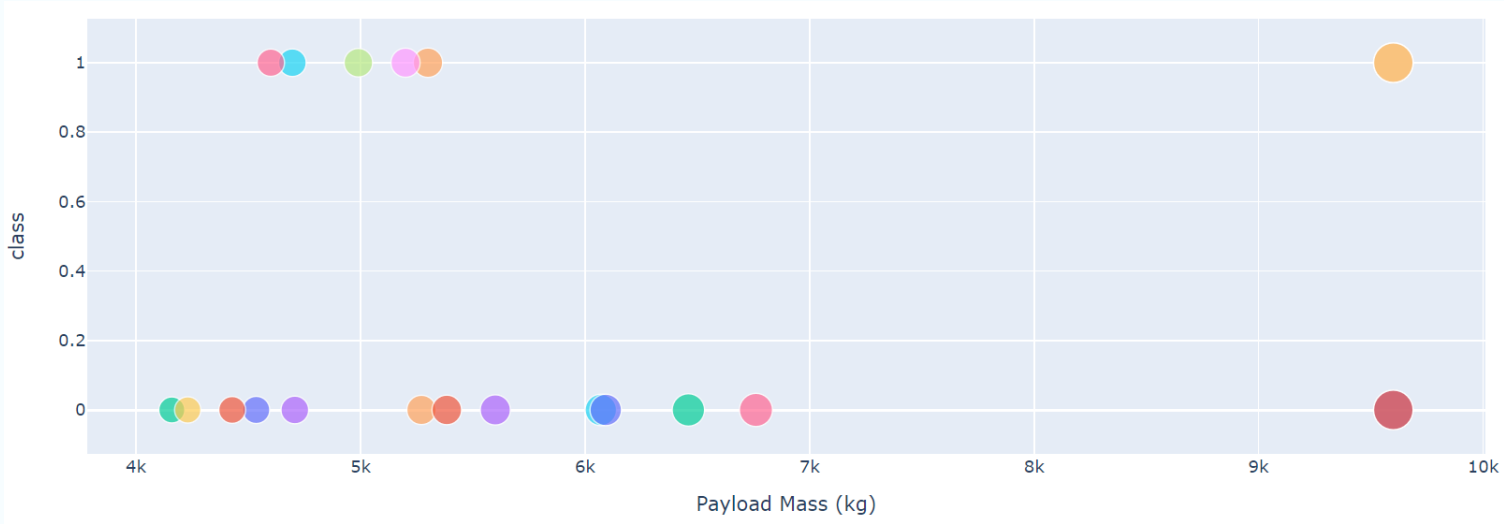
Live Site Code

Scatter Plot – Payload vs. Launch Outcome for Various Booster

Low Weighted Payloads (0kg - 4000kg)



Heavy Weighted Payloads (4000kg – 10000kg)



Live Site Code

We see that the success rate for low weighted payloads (0kg – 4000kg) is higher than heavy weighted payloads (4000kg – 10000kg)

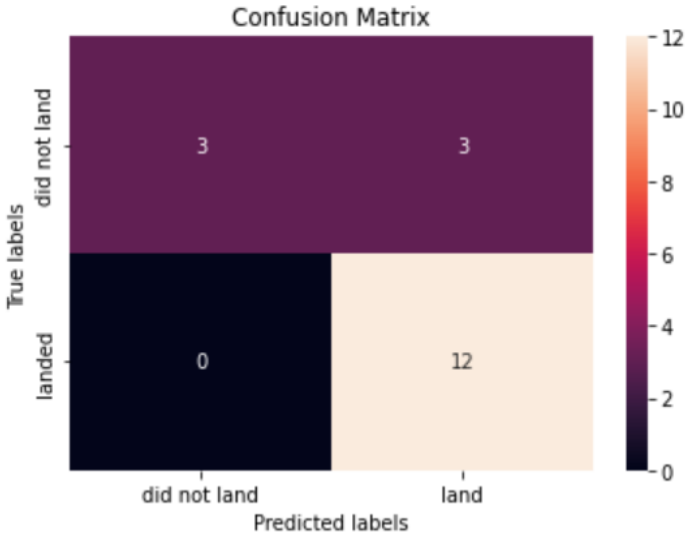
Section 6

Predictive Analysis (Classification)

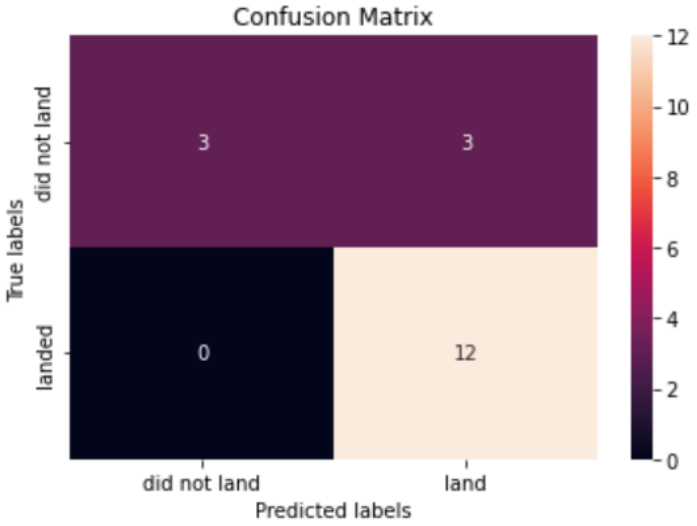
Confusion Matrix

The confusion matrix for each classification model shows the same result, as such the accuracy score using test data gives the same result regardless of model type, **83.33%**. To determine which model is best suited for prediction, we must evaluate the accuracy using train data.

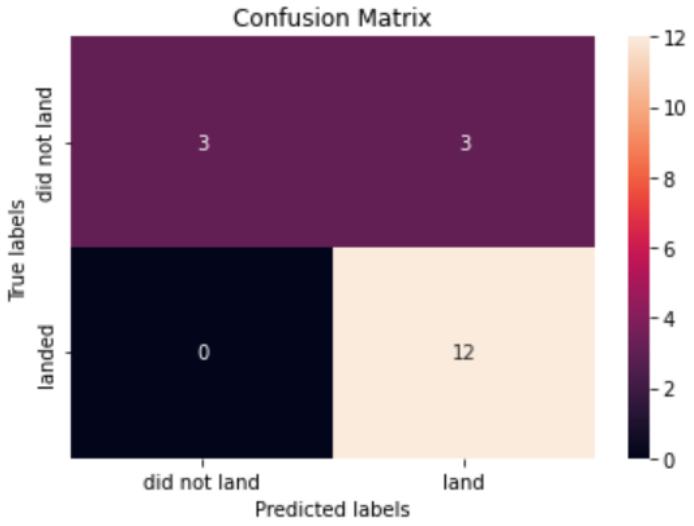
Decision Tree



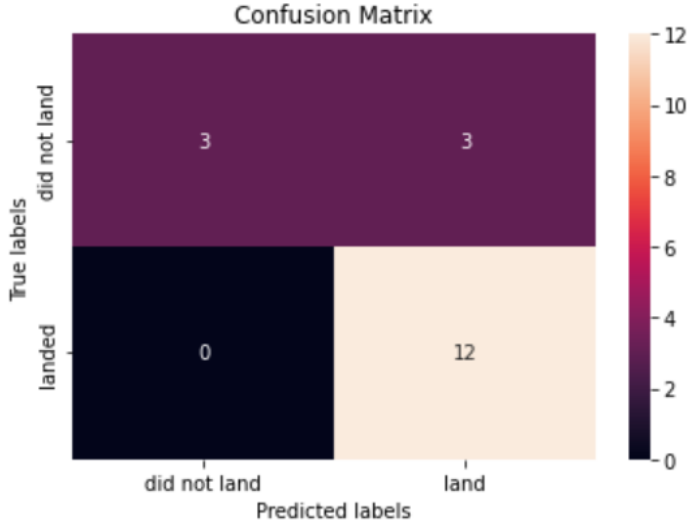
Logistic Regression



Support Vector Machine



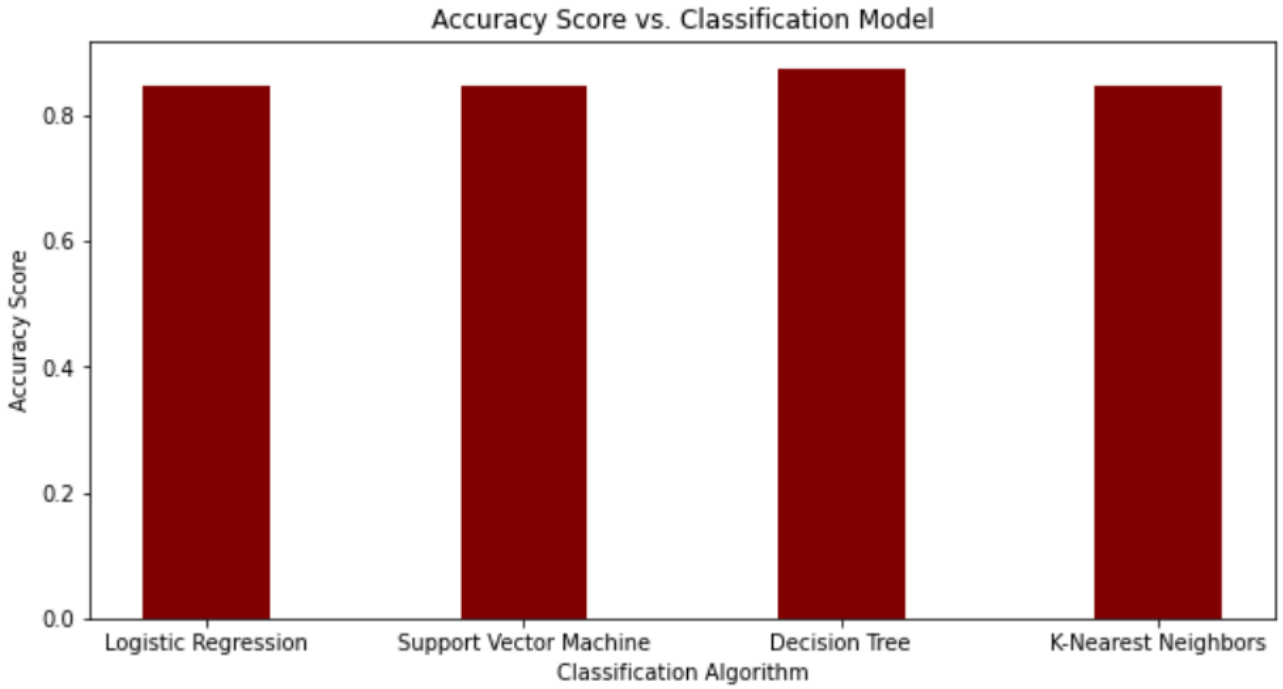
K-Nearest Neighbors



Classification Accuracy

- The **Decision Tree** classifier is the model with the highest classification accuracy with a value of 87.5%

Logistic Regression	Support Vector Machine	Decision Tree	K-Nearest Neighbors
0.846429	0.848214	0.875	0.848214




```
The best classification method for this case:
Decision Tree
tuned hpyerparameters :(best parameters)  {'cri
terion': 'gini', 'max_depth': 4, 'max_feature
s': 'auto', 'min_samples_leaf': 1, 'min_samples
_split': 2, 'splitter': 'random'}
accuracy :  0.875
```

Conclusions

- SpaceX have a successful recoveries that generally have the following properties:
  - Having light payload (lesser than 8000kg)
  - Launched from site KSC LC-39A
  - Most successful recovery used drone ship
  - Launch date in the year 2016 or later (having success rate above 0.6)
  - Used orbit GEO, HEO, SSO, ES-L1
- Using machine learning algorithm we predict the outcome of a given recovery with a reasonable degree of accuracy, 83.33%
- The machine learning algorithm best used for prediction model is **Decision Tree**

Apendix

Python Anywhere - Live Site for Plotly Dashboard

- To run the interactive plotly dashboard without interruptions 24/7, we used the service of 'Python Anywhere' (www.pythonanywhere.com) because its free to use with the limitation that a site run using its service will only operable for 3 months (we have to hit the renew button every 3 months).
- The implementation of the dashboard using Flask, we have two files flask_app.py and wasgi.py in order to run the site

Python Anywhere link

Plotly Dashboard running using Python Anywhere

Github link for dash app code

Thank you!