IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

NISCHALA.K
21 September 2022

Outline

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

Executive Summary

- **<u>Summary of methodologies</u>**

✓ Data Wrangling

✓ Exploratory Data Analysis with SQL

✓ Exploratory Data Analysis with Data Visualization

✓ interactive Visual Analytics through Folium

✓ Machine Learning Prediction

- **<u>Summary of all results</u>**

✓ Exploratory Data Analysis result

✓ Interactive analytics in screenshots

✓ Predictive Analytics result

Introduction

- **<u>Project background and context</u>**

Space x advertises falcon9 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. 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 space X for a rocket launch. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.

- **<u>Problems you want to find answers</u>**

➢ What factors determine if the rocket will land successfully?

➢ The interaction amongst various features that determine the success rate of a successful landing?

➢ What operating conditions needs to be in place to ensure a successful landing program?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:

  Data was collected using request and get operations on SpaceX API and web scraping using beautiful soup from Wikipedia

- Perform data wrangling

  Organize data into dataframe and clean null values and understand the relations

- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models

  Build, tune, evaluate classification models using sklearn,pandas libraries

Data Collection

The data was collected using various methods

- Data was collected using request & get operations on the SpaceX API.
- Next, decoded the response content as a Json using .json() function call and organized it to a pandas dataframe using .json_normalize() method.
- Filtered the dataframe for falcon9 records
- Then checked for the missing values in the data using isnull() , replaced 5 missing values of payload mass with its mean using np.nan, mean() & replace() methods .
- Also performed web scraping from Wikipedia for Falcon 9 launch records using BeautifulSoup-find_all() attribute.
- The objective was to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for future analysis.

Data Collection – SpaceX API

- data collected using get request on SpaceX API and got json response which normalized to pandas dataframe. After basic data wrangling, the result - dataset1.csv

- GitHub link of the completed SpaceX API calls notebook

  data collection using spaceX API notebook

1. Get request for rocket launch data using API

```
In [6]: spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
In [7]: response = requests.get(spacex_url)
```

2. Use json_normalize method to convert json result to dataframe

```
In [12]: # Use json_normalize method to convert the json result into a dataframe

         # decode response content as json
         static_json_df = res.json()
```

```
In [13]: # apply json_normalize
         data = pd.json_normalize(static_json_df)
```

3. We then performed data cleaning and filling in the missing values

```
In [30]: rows = data_falcon9['PayloadMass'].values.tolist()[0]

         df_rows = pd.DataFrame(rows)
         df_rows = df_rows.replace(np.nan, PayloadMass)

         data_falcon9['PayloadMass'][0] = df_rows.values
         data_falcon9
```

Data Collection - Scraping

- web scraped falcon9 records from Wikipedia using BeautifulSoup & its attributes. used html parser, dictionary and pandas to obtain the dataset- webscraped dataset.csv

- GitHub link of the web scraping notebook webscraping notebook

1. Apply HTTP Get method to request the Falcon 9 rocket launch page

In [4]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

In [5]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
html_data = requests.get(static_url)
html_data.status_code
```

Out[5]: 200

2. Create a BeautifulSoup object from the HTML response

In [6]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(html_data.text, 'html.parser')
```

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

In [7]:
```
# Use soup.title attribute
soup.title
```

Out[7]:
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

3. Extract all column names from the HTML table header

In [10]:
```
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

element = soup.find_all('th')
for row in range(len(element)):
    try:
        name = extract_column_from_header(element[row])
        if (name is not None and len(name) > 0):
            column_names.append(name)
    except:
        pass
```

4. Create a dataframe by parsing the launch HTML tables
5. Export data to csv

Data Wrangling

- First understand various numerical and categorical features of data using dtypes and calculated number of launch sites , orbits , outcomes by using .value_counts(). Then, looped df['outcome'] to create a class column representing the landing outcome result and got the success rate as class mean using mean().
- Dataset obtained after data wranging- dataset2.csv
- GitHub link of data wrangling notebook data wrangling notebook

Calculated number of launch sites, orbits, outcomes using .value_counts()

↓

Derive bad_outcomes and loop outcomes & conditioned with bad_outcomes

↓

Create class column with success=1 else 0 and save data into a dataframe

EDA with SQL

- Download the spacex.csv file and load it to the database. Now, create a connection to the database and query the data in jupyter notebook using sqlite3 & magicSql.
- Some of the questions, to which we executed queries and got insights-

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

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

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

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

- GitHub link of EDA with SQL notebook- EDA with SQL notebook

EDA with Data Visualization

- Various charts were plotted like scatter plots between flight number, payload mass, launch site, orbit types to visualize the relationship between one another. A bar graph and a line plot were also plotted to get the insights related to success rate vs orbit types and yearly success trend respectively.
- The plots were listed in the following section 2- 'Insights drawn from EDA' .
- GitHub link of EDA with data visualization notebook- EDA with visualizations notebook

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. We answered some question for instance:
  - ✓ -Are launch sites near railways, highways and coastlines.
  - ✓ Do launch sites keep certain distance away from cities
- GitHub link of interactive map with Folium map- folium map notebook

Build a Dashboard with Plotly Dash

- Built an interactive dashboard which consists of a dropdown, a pie-chart, a slider and a scatter plot
- The dropdown listed various launch sites to select and display respective launch site's success rate pie-chart with default value as all,which renders a pie-chart with all launch sites success rate keeping faillures aside.
- Then, followed a slider with payload mass range of 0-10,000kg. At last plotted a bubble plot, where success rate associated with pie-chart on y-axis and payload mass fixed on the slider on X-axis for various booster versions.
- GitHub link of Plotly Dash lab- Dashboard with plotly dash lab

Predictive Analysis (Classification)

- Firstly, loaded the data using pandas by reading csv file to dataframe and created an array using df['class'] & to_numpy() , standardize the data using StandardScaler() method and fit_transform() to tranform the data , then split the data into train and test data using train_test_split() method with test_size of 20%,random_state as 2.
- Built logistic regression, decision tree, SVM, KNN algorithms and tuned different hyperparameters using GridSearchCV with cv=10.then found best parameters for each algorithm
- Used score() to calculate the accuracy of the model and plotted confusion matrix to get insights.
- The best performing classification model is founded by getting algorithm model with max(accuracy score)
- GitHub link of predictive analysis lab- machine learning prediction analysis

Results

Exploratory data analysis results

Interactive analytics demo in screenshots

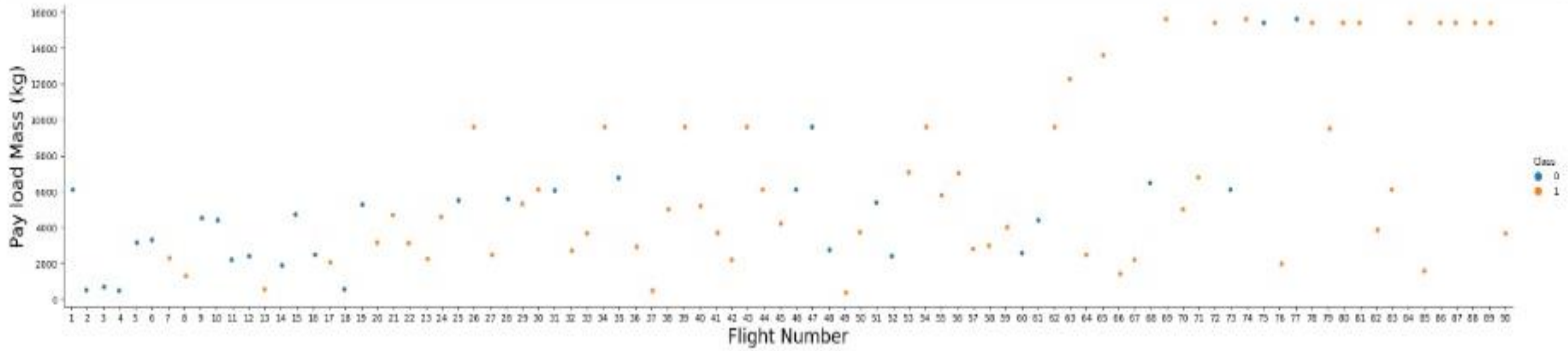
Predictive analysis results

Section 2

Insights drawn from EDA

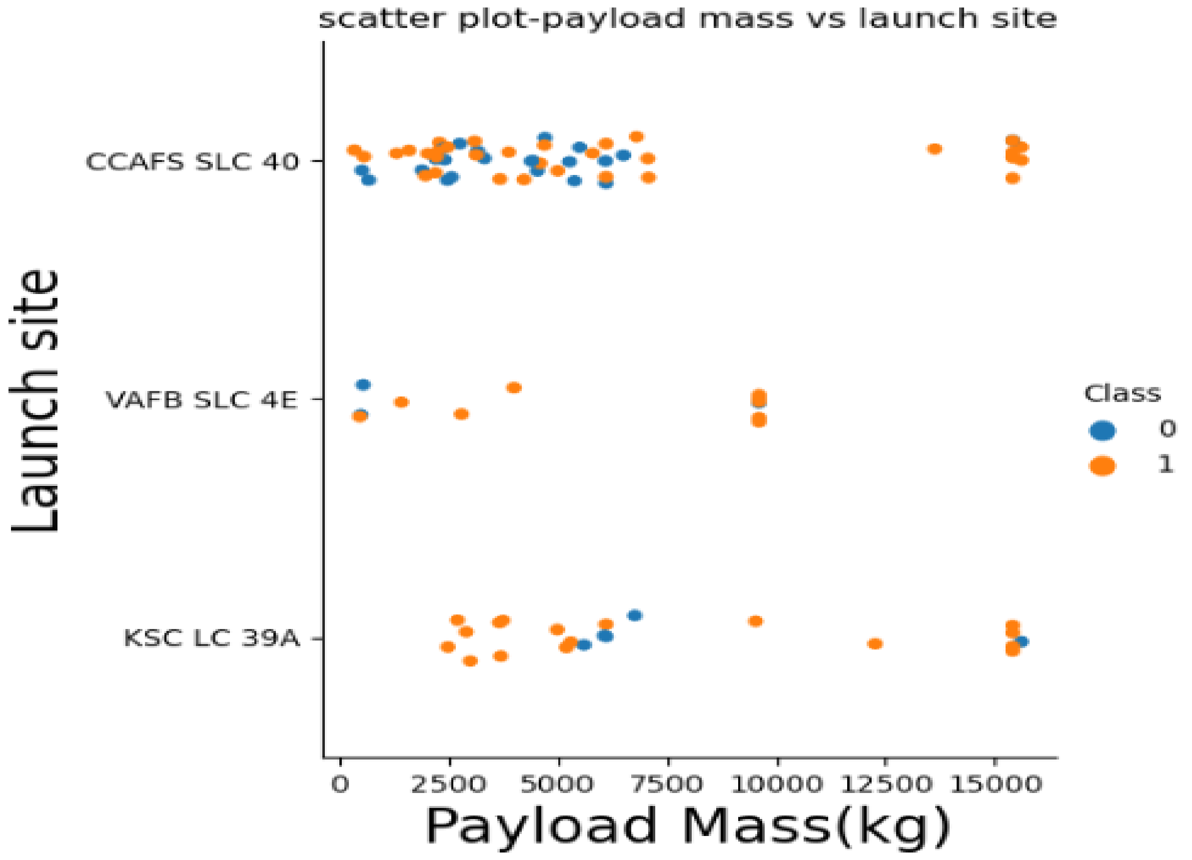
Flight Number vs. Launch Site

[3]:
```
sns.catplot(y="PayloadMass", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```



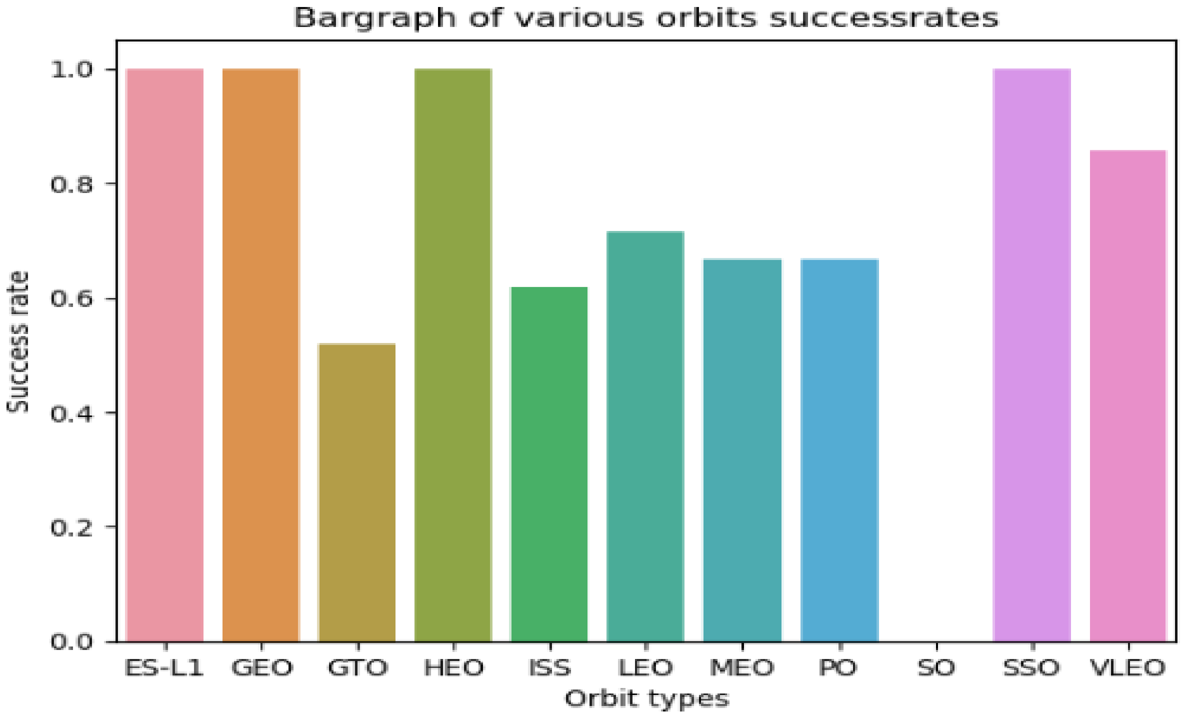
We see that different launch sites have different success rates. `CCAFS LC-40`, has a success rate of 60 %, while `KSC LC-39A` and `VAFB SLC 4E` has a success rate of 77%.

Payload vs. Launch Site



Now if you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

Success Rate vs. Orbit Type

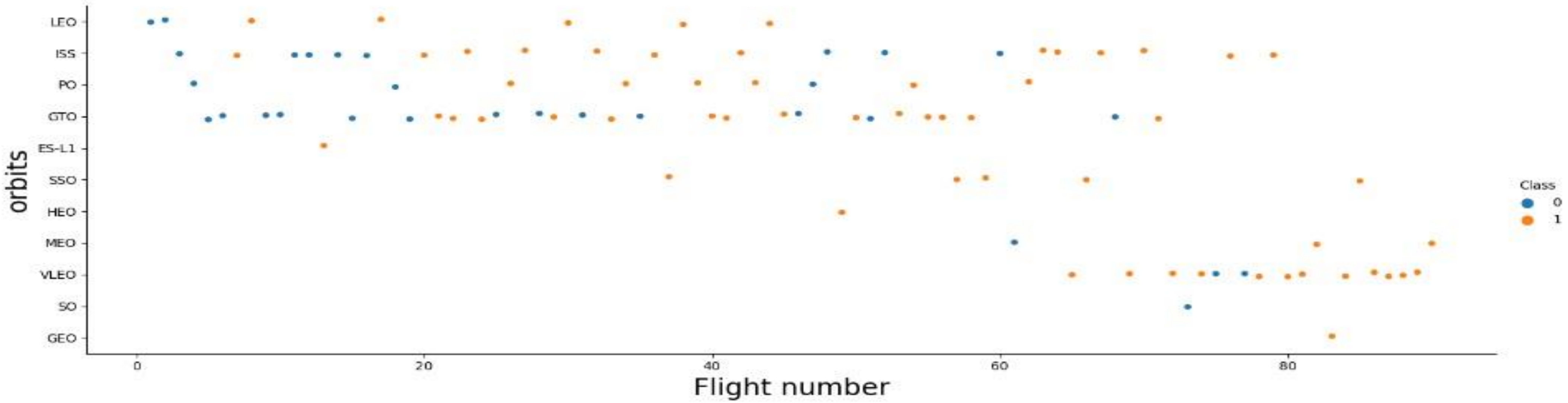


Let's create a `bar chart` for the sucess rate of each orbit

Analyze the ploted bar chart try to find which orbits have high success rate. **orbits with high success rate are-** ES-L1, GEO, HEO, SSO

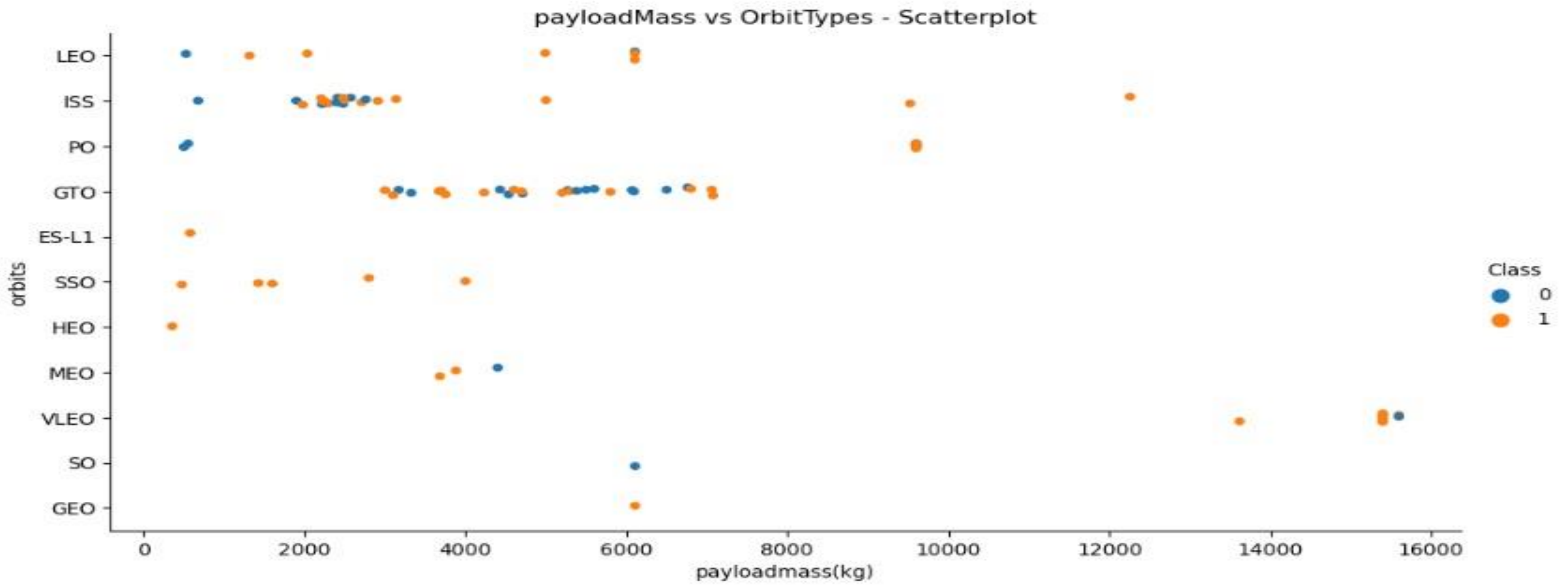
Flight Number vs. Orbit Type

```
[7]: # Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
sns.catplot(x="FlightNumber",y="Orbit",data=df,hue="Class",aspect=3)
plt.xlabel("Flight number",fontsize=20)
plt.ylabel("orbits",fontsize=20)
plt.show()
```



You should 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.

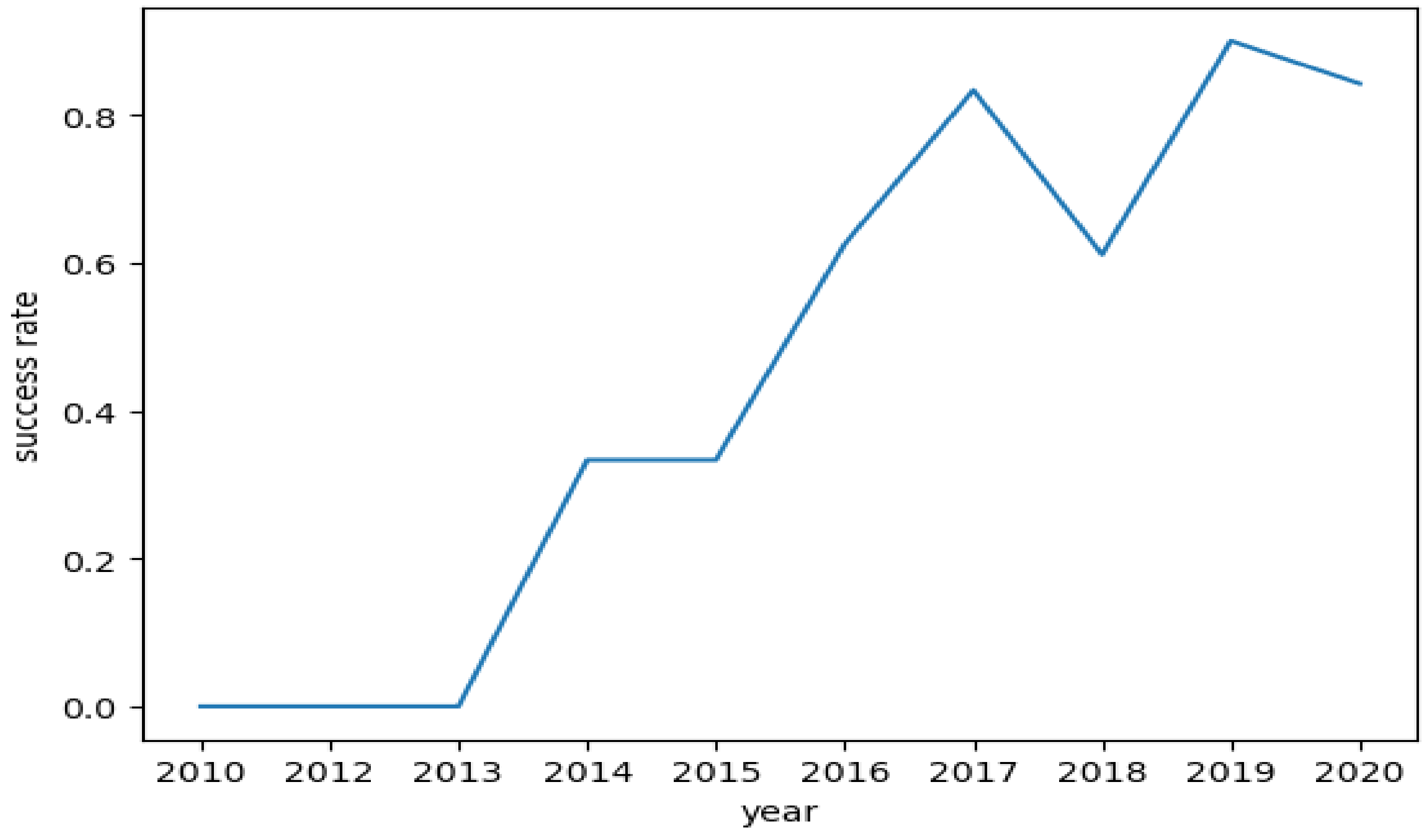
Payload vs. Orbit Type



With heavy payloads the successful landing or positive landing rate are more for Polar,VLEO and ISS.

However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both present.

Launch Success Yearly Trend



you can observe that the sucess rate since 2013 kept increasing till 2020

All Launch Site Names

Task 1

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

[7]:
```
%%sql
select distinct(LAUNCH_SITE) from SPACEXTBL;
```

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

[7]:

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

- Used distinct keyword to get unique launch sites

Launch Site Names Begin with 'CCA'

- Used where with like to get launch site with `CCA`

Task 2

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

[8]:
```
%%sql
select * from SPACEXTBL where LAUNCH_SITE like 'CCA%%' limit 5
```

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

[8]:

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

- Used sum to calculate the total payload carried by boosters from NASA

Task 3

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

```
[9]: %%sql
select sum(PAYLOAD_MASS__KG_) AS total_payload_mass from SPACEXTBL where CUSTOMER='NASA (CRS)'
```

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

[9]:

total_payload_mass
45596

Average Payload Mass by F9 v1.1

- Used avg to calculate the average payload mass carried by booster version F9 v1.1

Task 4

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

[10]:
```
%%sql
select avg(PAYLOAD_MASS__KG_) AS avg_payload_mass from SPACEXTBL where Booster_Version like 'F9 v1.1%%'
```

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

[10]:

avg_payload_mass
2534.6666666666665

First Successful Ground Landing Date

- Used min on date to get the date of the first successful landing outcome on ground pad

Task 5

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

Hint:Use min function

[11]:
```
%%sql
select Min(Date) as first_successful_landingoutcome_groundpad from SPACEXTBL where landing_outcome like 'Success (ground pad)'
```

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

[11]:

first_successful_landingoutcome_groundpad
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- used the **WHERE** clause to filter for boosters which have successfully landed on drone ship and applied between to get the payload mass greater than 4000 but less than 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

[12]:
```
%%sql select Booster_Version,PAYLOAD_MASS__KG_ from SPACEXTBL
where landing_outcome='Success (drone ship)' and PAYLOAD_MASS__KG_ between 4000 and 6000
```

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

[12]:

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

- Used **group by on mission outcome** to calculate the total number of successful and failure mission outcomes

Task 7

List the total number of successful and failure mission outcomes

[13]:
```
%sql select Mission_Outcome,count(Mission_Outcome) from SPACEXTBL group by(Mission_Outcome)
```

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

[13]:

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

Boosters Carried Maximum Payload

- Used max on payload mass sub query to list the names of the booster which have carried the maximum payload mass

Task 8

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

[14]:
```
%%sql
select Booster_version,PAYLOAD_MASS__KG_ from SPACEXTBL
where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from SPACEXTBL)
```

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

[14]:

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

- list the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015- found 2 such 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.

[15]:
```
%%sql
select substr(Date,6,2) as month,substr(Date,1,4) as year,landing_outcome,Booster_Version,Launch_Site from SPACEXTBL
where year='2015'and landing_outcome='Failure (drone ship)'
```

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

[15]:

month	year	landing_outcome	Booster_Version	Launch_Site
01	2015	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	2015	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

- selected Landing outcomes and the **COUNT** of landing outcomes from the data and used the **WHERE** clause to filter for landing outcomes **BETWEEN** 2010-06-04 to 2010-03-20.
- Then applied the **GROUP BY** clause to group the landing outcomes and the **ORDER BY** clause to order the grouped landing outcome in descending order

Task 10

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

[7]:
```
%%sql
select landing_outcome,count(landing_outcome) as outcomes from SPACEXTBL
where Date between '2010-06-04'and '2017-03-20'
group by landing_outcome
order by outcomes DESC
```

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

[7]:

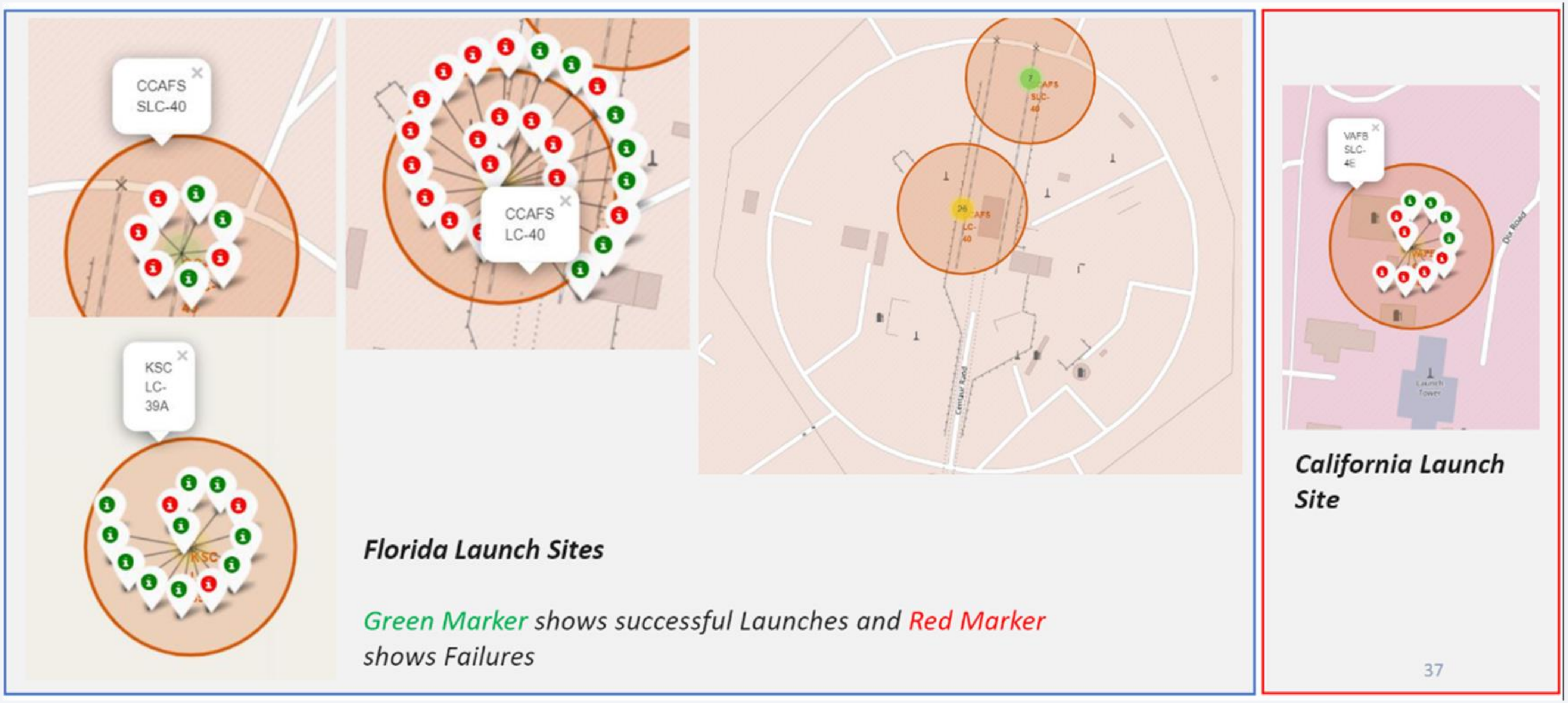
landing_outcome	outcomes
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

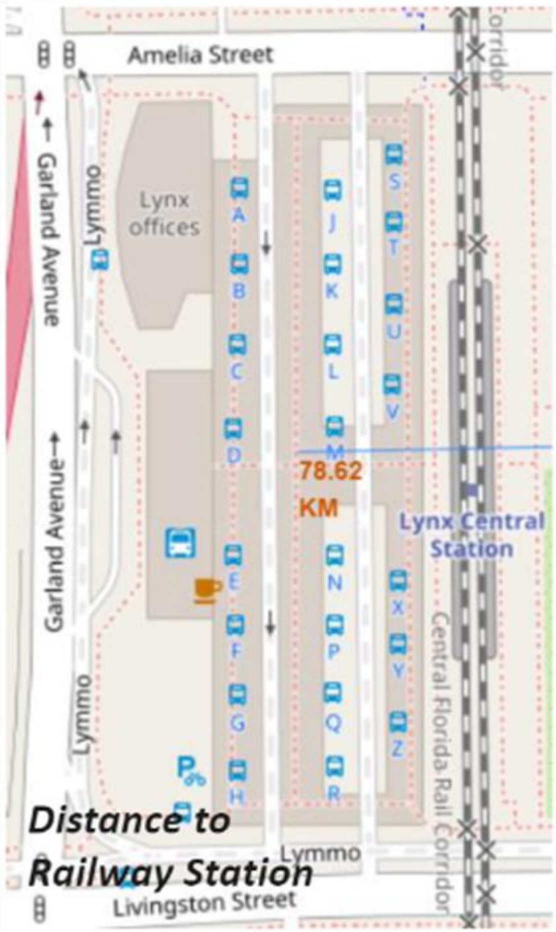
Launch Sites Proximities Analysis

All launch sites global map markers

Markers with color labels in clusters



launch site distance to landmarks



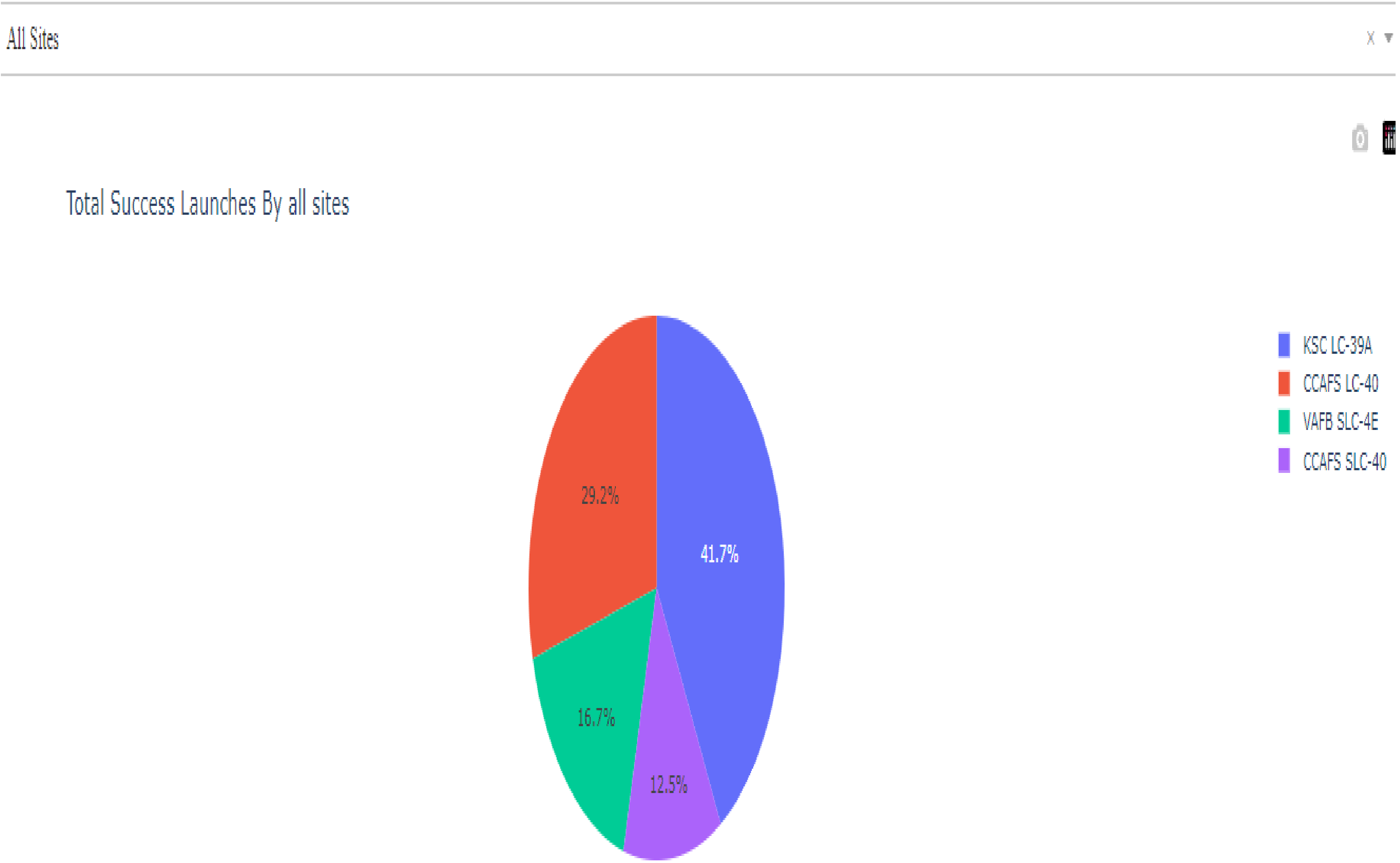
- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

Section 4

Build a Dashboard with Plotly Dash

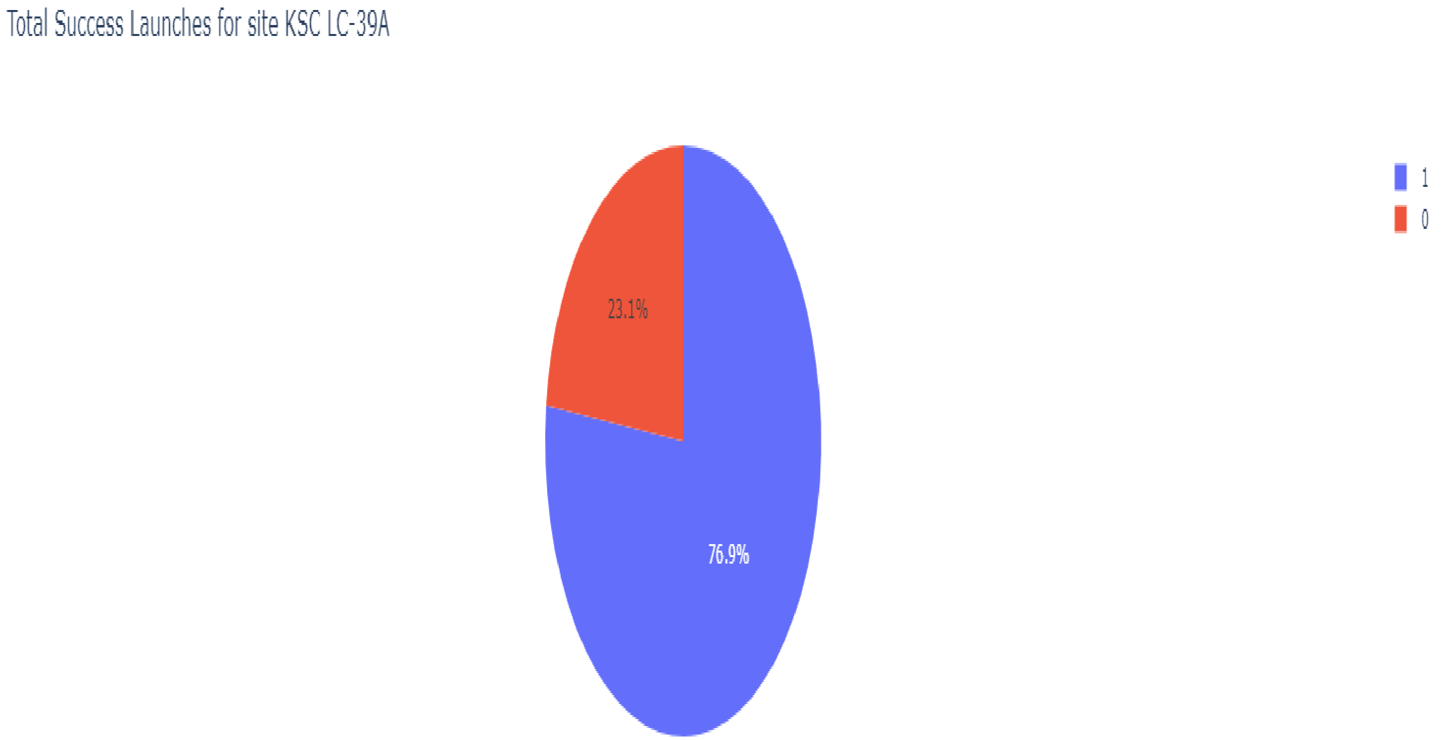
Launch success count for all sites in pie-chart

- Maximum success rate is seen in KSC LS-39A



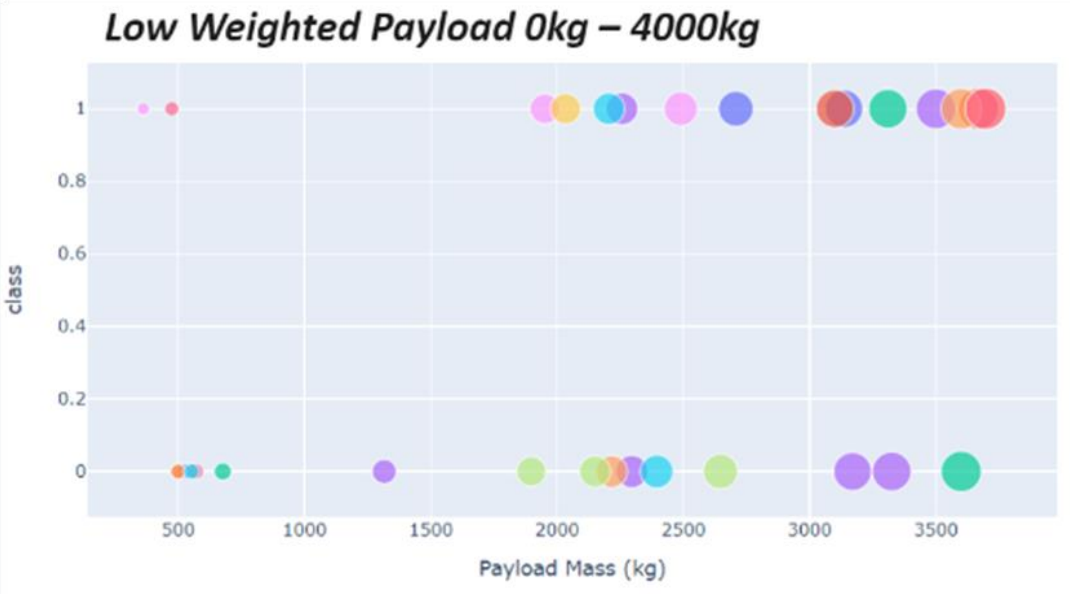
Pie-chart of highest success ratio launch site

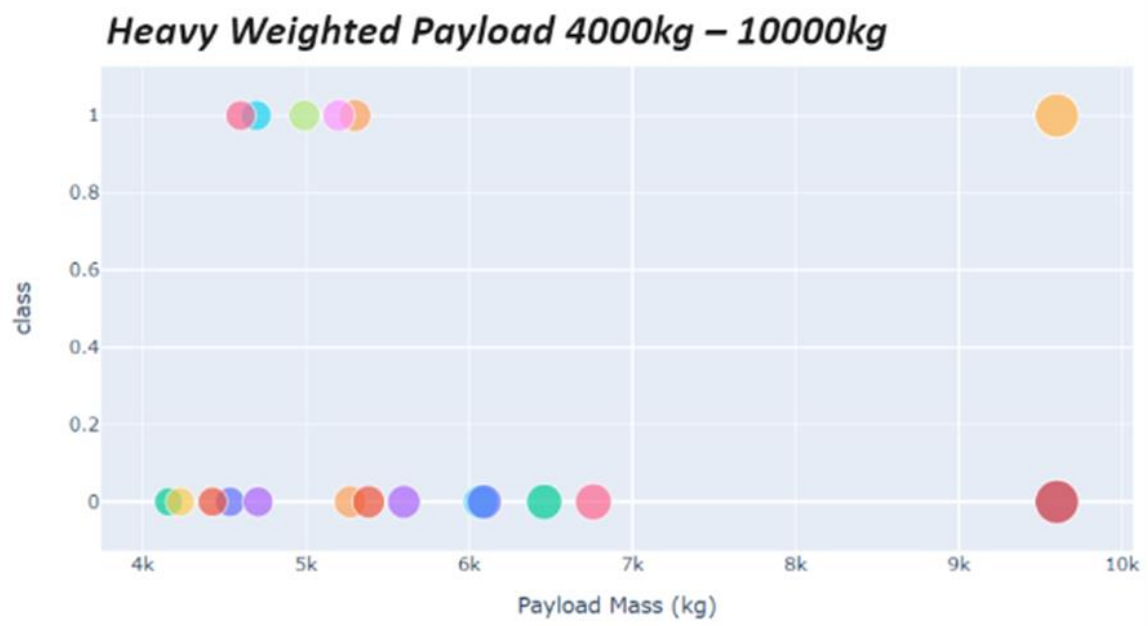
- Highest success rate is seen in KSC LS_39A with 76.9% success



Payload vs success rate bubble plot

- payload range- 2000 to 4000 has highest success rate and booster version B1014 have the largest success rate.





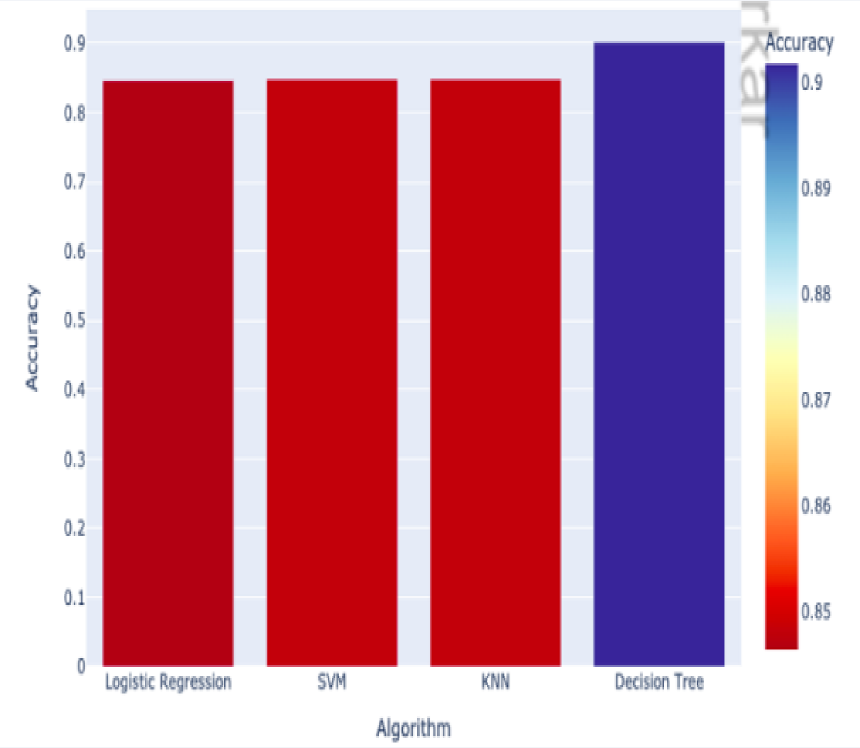
We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

Section 5

Predictive Analysis (Classification)

Classification Accuracy

- The best model with highest classification accuracy of 87% is **decision tree classifier**



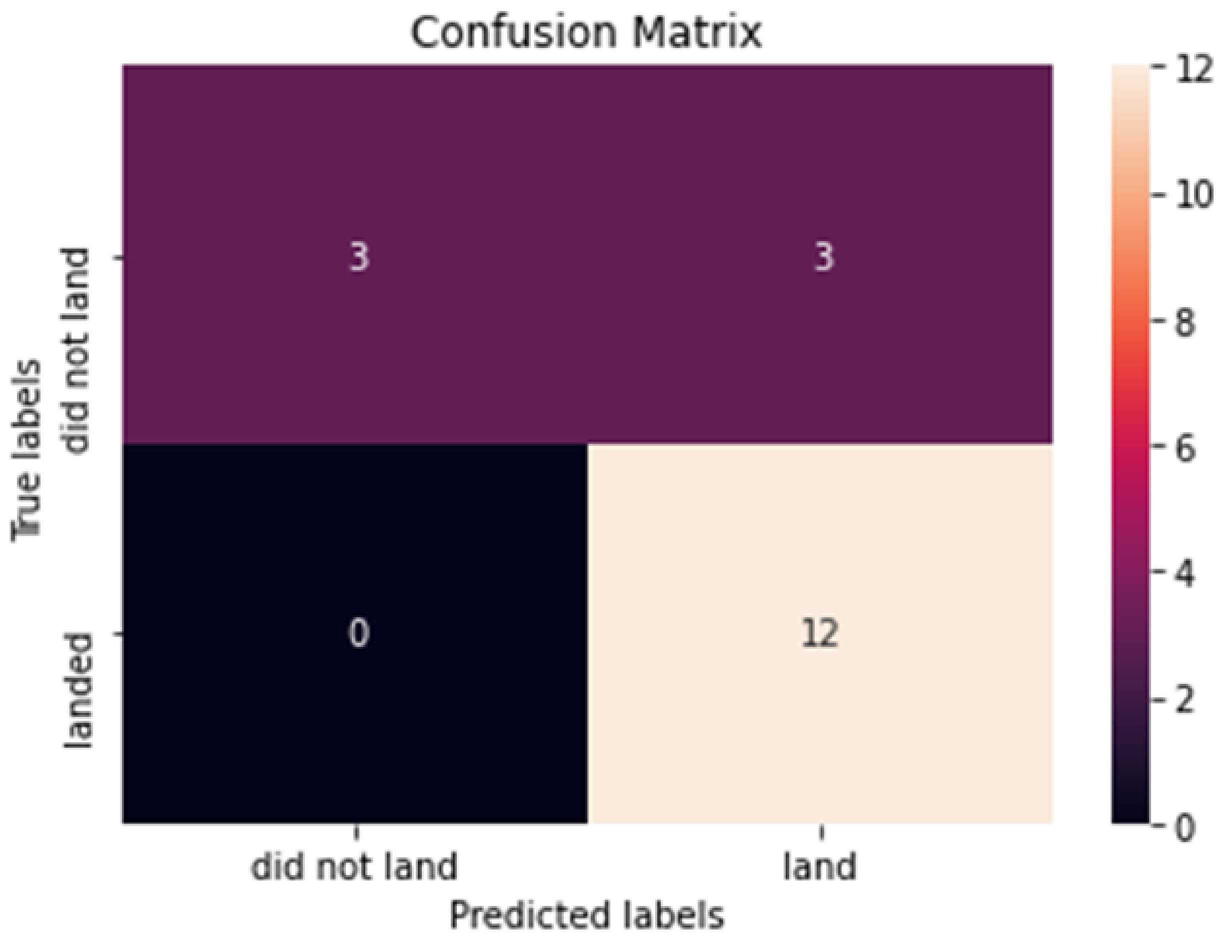

```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

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 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

Confusion Matrix

- The problem is with false positives .i.e., unsuccessful landing marked as successful landing by the classifier.



Conclusions

- 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.
- The Decision tree classifier is the best machine learning algorithm for the given dataset.

Appendix

- Plotly dashboard full view pdf
- spacex_geo_dataset.csv

Thank you!