IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Daniel Rodela
12/12/2023

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection via API
  - Data collection via Web Scraping
  - Data Wrangling
  - Exploratory analysis using SQL
  - Exploratory analysis with Visualization
  - Interactive Visual Analytics with Folium
  - Machine Learning with predictive analysis
- Summary of results
  - Interactive Analytics with print screen shots
  - Predictive analytics from Machine Learning lab
  - Exploratory Data Analysis

Introduction

- Project background and context

  Commercial suborbital space flights are being offered at the cost of $62 million per launch by space flight pioneer SpaceX. This low-cost fee is absolutely dependent on the ability to reuse the extremely expensive first stage component of their rockets. However, re-landing of this component is unpredictable due to a crash, payload, orbit or other issues. Monitoring SpaceX launches, new rocket company SpaceY, is entering the market. In order to offer lower space flights, they are building a newly created Machine Learning (ML) prediction model and its learning pipeline. With the goal of utilizing this model's ability to determine SpaceX launch costs and predicting successful stage 1 landings using public access data, SpaceY will achieve ownership of the suborbital space flight market share.

- Problems requiring answers:
  - Identify key factors that determine a successful stage 1 landing.
  - Study, summarize, and explain relationships between features that determine positive landings
  - Recognize operational conditions required for successful landing

Section 1

Methodology

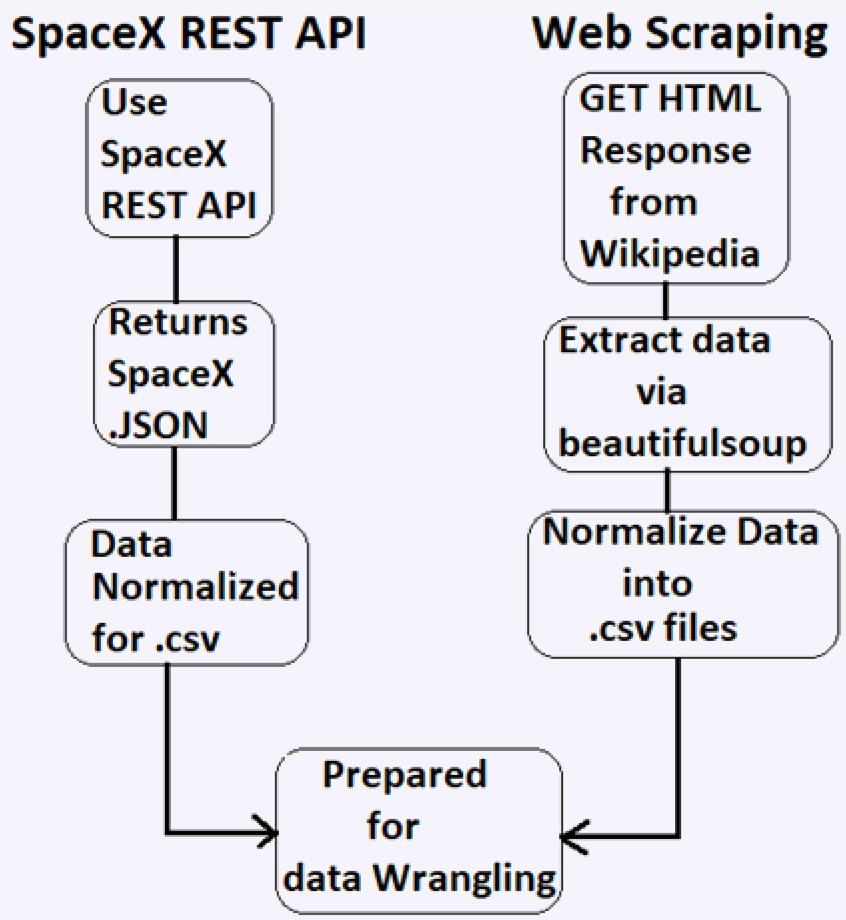
Methodology

Executive Summary

- Data collection methodology:
  - Web-scraping from Wikipedia
  - SpaceX REST API
- Perform data wrangling
  - One-hot encoding was used for categorical features, null values removal and irrelevant columns
- 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

- Datasets included in the analysis:
  - SpaceX launch data extracted using the SpaceX REST API.
  - Coding included returning HTTP request content as a JSON using .json() function call
  - Converted returned JSON objects into a pandas dataframe using .json_normalize().
  - Explored public access data returned for details related to rockets used, payload, launch details, landing specs and landing results
  - Data prepared for Wrangling stage
- Applied web scraping to extract Falcon 9 launch records with BeautifulSoup
- Tables parsed and converted pandas dataframe and exported to .csv format.



Data Collection – SpaceX API

- Used the get request to the SpaceX API to collect data, converted the returned data and performed basic data wrangling and cleaning.

- GitHub link: https://github.com/dmrodela/Applied-Data-Science-Capstone-Week-1/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

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 scraping was used on the Falcon 9 Wikipedia page to webscrape Falcon 9 launch records using BeautifulSoup
- Tables were parsed and converted it into pandas dataframe.
- GitHub link: https://github.com/dmrodela/Web-scraping-Falcon-9-and-Falcon-Heavy-Launches-Records-from-Wikipedia

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

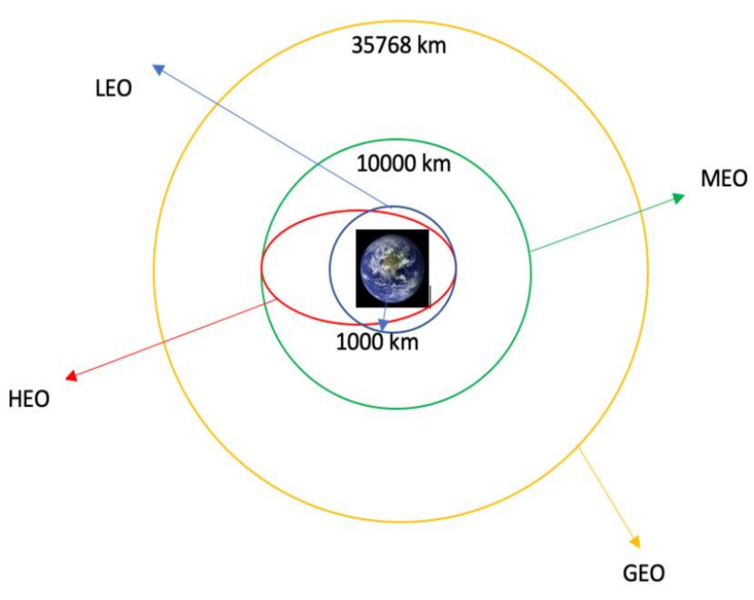
In [10]:
```
column_names = []

# 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

Use the method .value_counts() on the column Outcome to determine the number of landing_outcomes

Data Wrangling

- Performed exploratory data analysis and determined the training labels.
- Calculated the number of launches on each site including the number and occurrence of each orbits
- Used the method .value_counts() on the column Outcome to determine the number of landing_outcomes
- GitHub link: https://github.com/dmrodela/Week-1-Lab-2-Data-wrangling




```
[7]: # landing_outcomes = values on Outcome column
     landing_outcomes = df['Outcome'].value_counts()
     landing_outcomes

[7]: True ASDS      41
     None None      19
     True RTLS      14
     False ASDS      6
     True Ocean      5
     False Ocean     2
     None ASDS       2
     False RTLS      1
     Name: Outcome, dtype: int64
```

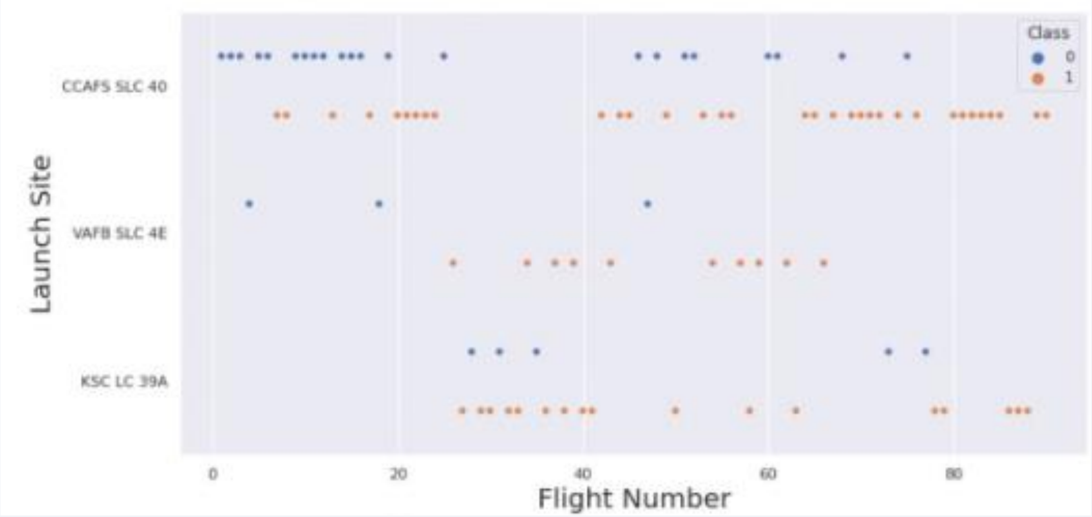
```
[5]: # Apply value_counts() on column LaunchSite
     launches = df['LaunchSite'].value_counts()
     launches

[5]: CCAFS SLC 40    55
     KSC LC 39A      22
     VAFB SLC 4E     13
     Name: LaunchSite, dtype: int64
```

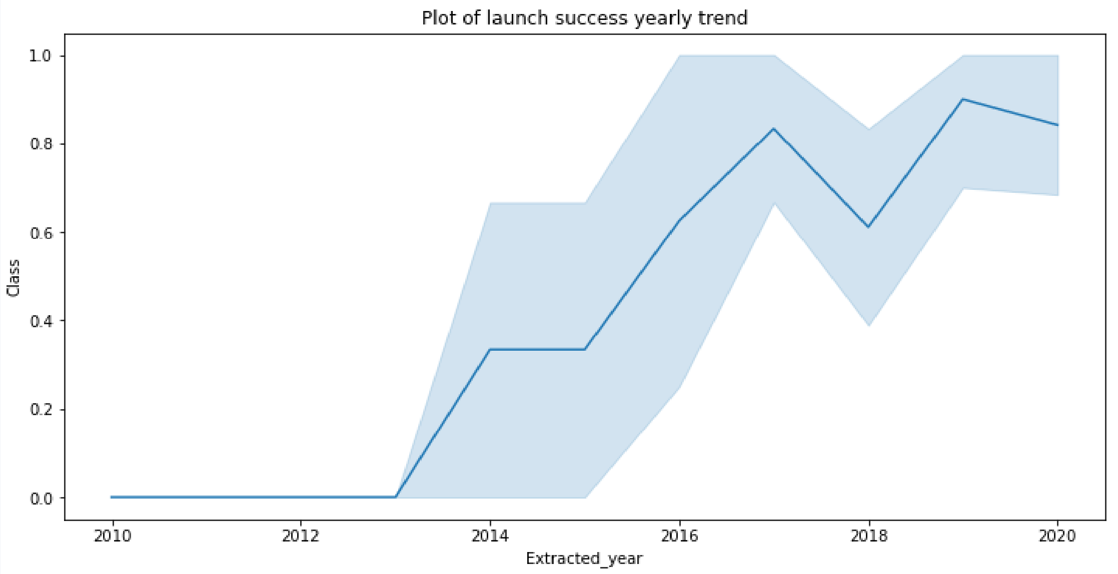
EDA with Data Visualization

- Data exploration began by using various graphs to find relationships between attributes. In particular, Payload vs Flight Number and Payload vs Launch Site.
- Additionally, Flight Number vs Launch Site, Flight Number vs Orbit Type and Yearly launch success rates.

- These relationships are included in the graphs to the right.

- Github link: https://github.com/dmrodela/Assignment-Exploring-and-Preparing-Data/blob/main/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb





EDA with SQL

- EDA was used with SQL to gain understanding of the data. SQL queries were coded to:

  - Return the names of the launch sites.

  - Return 5 records where launch sites begin with the string ‘CCA’.

  - Return the total payload mass carried by booster launched by NASA (CRS).

  - Return the average payload mass carried by booster version F9 v1.1.

  - Listing the date when the first successful landing outcome in ground pad was achieved.

  - Listing the names of the boosters which have success in drone ship and have payload mass launch sites reference and peer-review purpose

  GitHub link: https://github.com/dmrodela/Week-2-Hands-on-Lab-Complete-the-EDA-with-SQL-Assignment-SQL-Notebook-for-Peer-Assignment

Build an Interactive Map with Folium

Folium and Plotly Dash coding was used to build an interactive map and dashboard to perform interactive visual analytics where:

- Launch sites were marked, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- Feature launch outcomes (failure or success) to class 0 = failure and 1 = success were assigned.
- Using the color-labeled marker clusters, launch sites were identified as having a relatively high success rate.
- Distances were calculated between a launch site to its proximities. Questions related to the following were answered:
- Are launch sites near railways, highways and coastlines.
- Do launch sites keep certain distance away from cities.
- GitHub link: https://github.com/dmrodela/Week-3-Launch-Sites-Locations-Analysis-with-Folium/blob/main/lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

- The ML model provided data resulting in the design of interactive dashboards coded with Plotly Dash
- Dashboards included plotted pie charts displaying total launches by key rocket launch sites
- Scatter graphs revealed correlations between Outcome and Payload Mass (Kg) for various booster versions.
- GitHub link: https://github.com/dmrodela/Week-3-Build-a-Dashboard-Application-with-Plotly-Dash/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

Build the ML Model

- Data frames were loaded using numpy and pandas
- Data transformation performed including splitting the data into train and test partitions using train_test_split()
- Parameter classification algorithms were tested and the best one selected using GridSearchCV

Model Evaluation

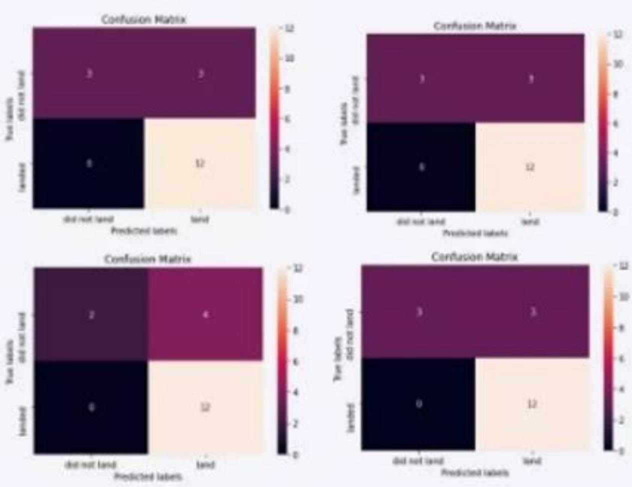
- Calculated the accuracy on the test data using the method score()
- Created a logistic regression object then created a GridSearchCV object logreg_cv with cv = 10.
- Fitted the object to find the best parameters using dictionary parameters
- Output generated a GridSearchCV object for logistic regression
- Included a confusion matrix to examine how logistic regression can distinguish between the different classes

ML Model Improvement

Employed Feature engineering and Algorithm Tuning

Best Model

The SVM, and Logistic Regression Model(LR) performed best where the LR achieved a 83.3% accuracy with the SVM at .958 in relation to Area Under the Curve.



GitHub link:https://github.com/dmrodela/Week-4-Assignment-Machine-Learning-Prediction/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Results

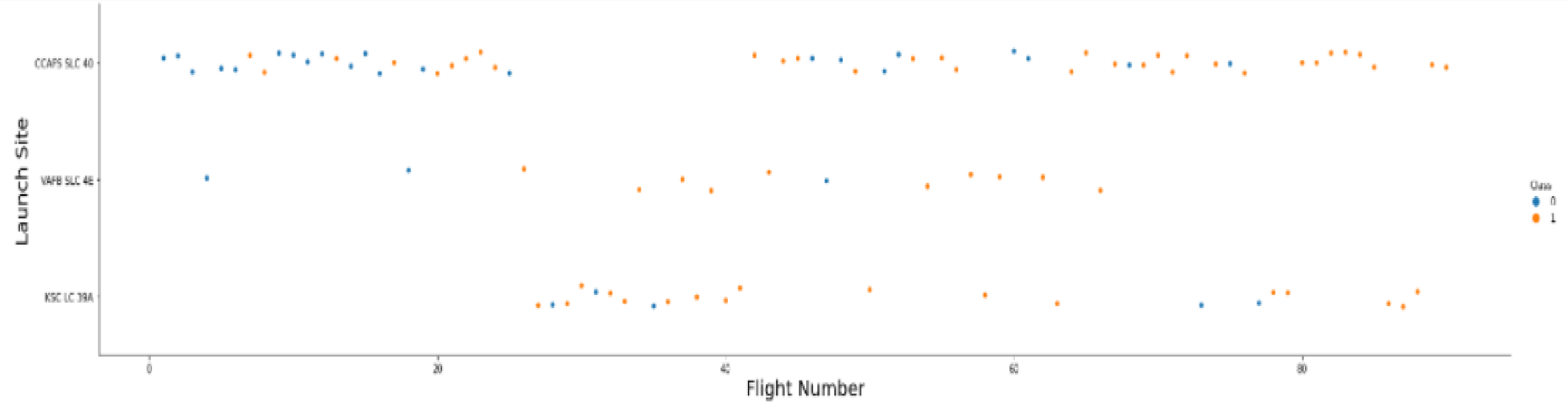
- The SVM, and Logistic Regression Model(LRM) performed best
- Moderate-weighted pay lodes were critical and extremely noticeable in performance
- KSC LC 39A achieved the highest amount of successful launches of all launch sites
- Orbit GEO, HEO, SSO, ES L1 achieved the best success rates

Section 2

Insights drawn from EDA

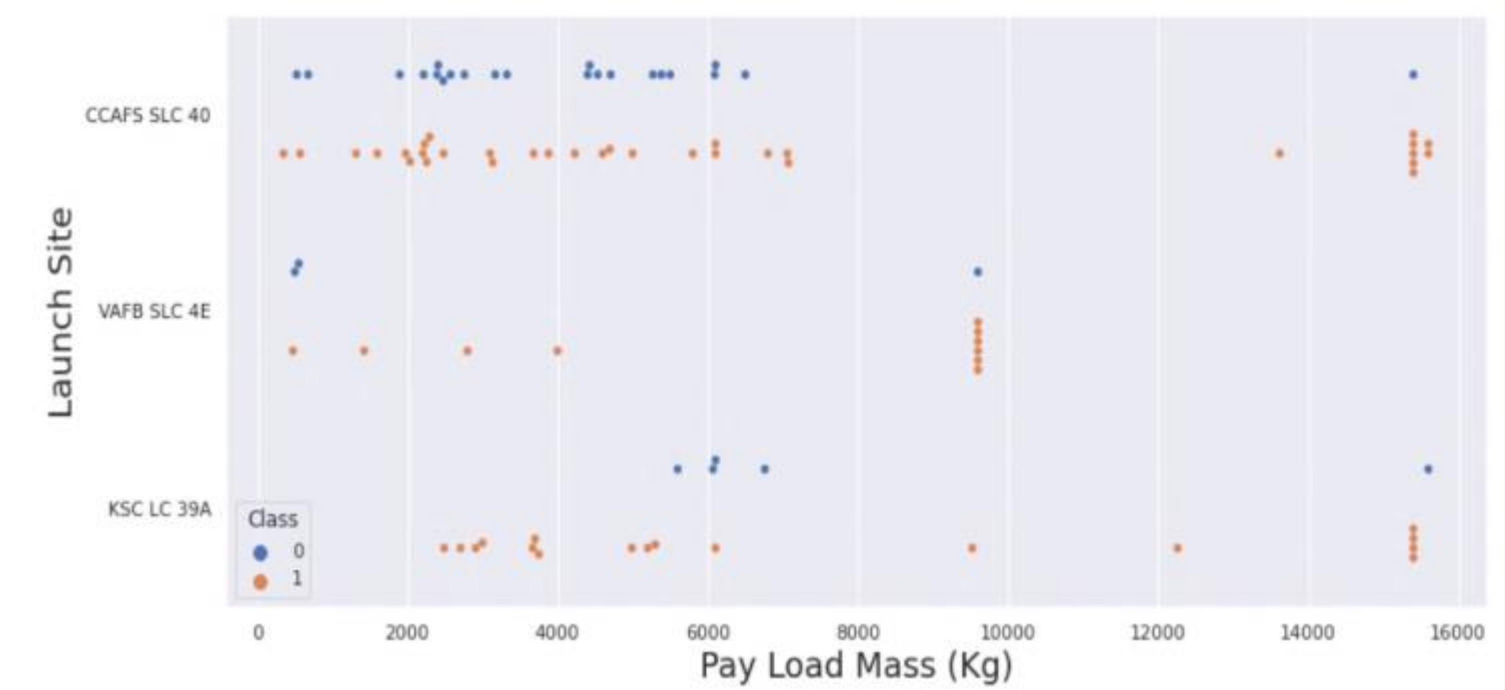
Flight Number vs. Launch Site

- From the plot, the team found that the larger the flight amount at a launch site, the greater the success rate at a launch site.



- However, launch site CCAFS SLC40 does not follow this trend.

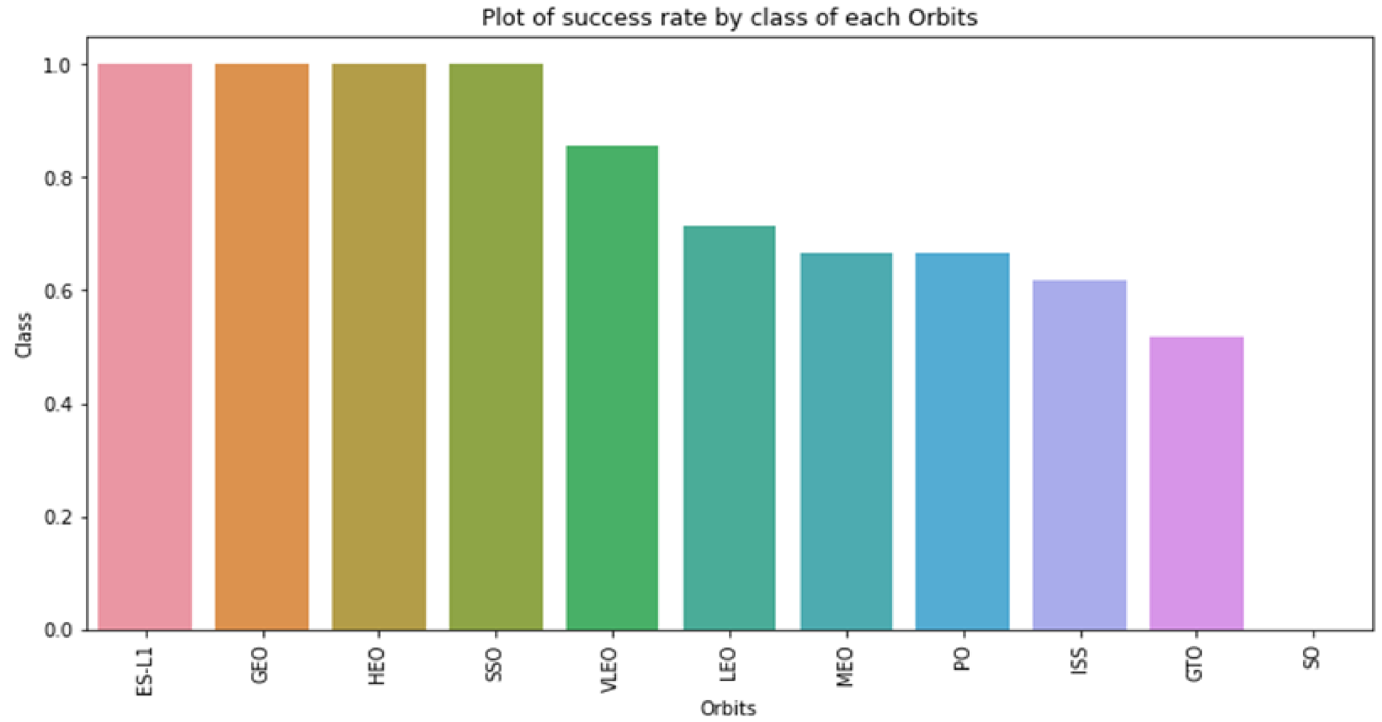
Payload vs. Launch Site



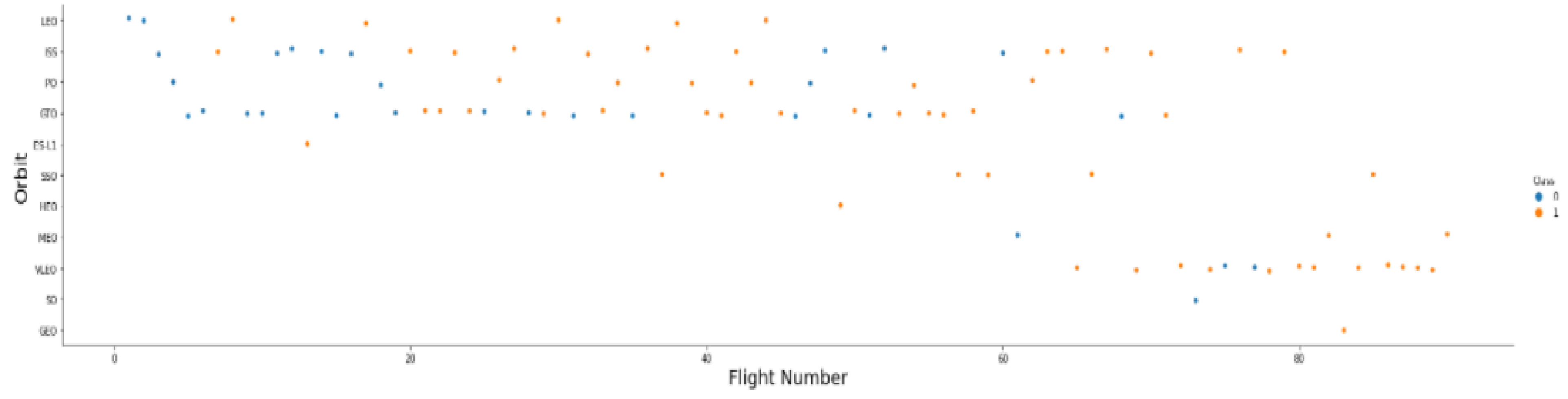
The Payload vs Launch Site scatter plot clearly indicates when the pay load mass is less than 7000kg, the probability success rate increases significantly. However, launch site CCFAS SLC 40 was consistently more successful with lighter payloads

Success Rate vs. Orbit Type

- From the bar chart on the right we can view the data comparisons of Success Rate vs. Orbit type
- Each rectangular bar represents the value of plotted data
- It is easily recognizable that ES-L1, GEO, HEO, SSO, VLEO had the highest success rates



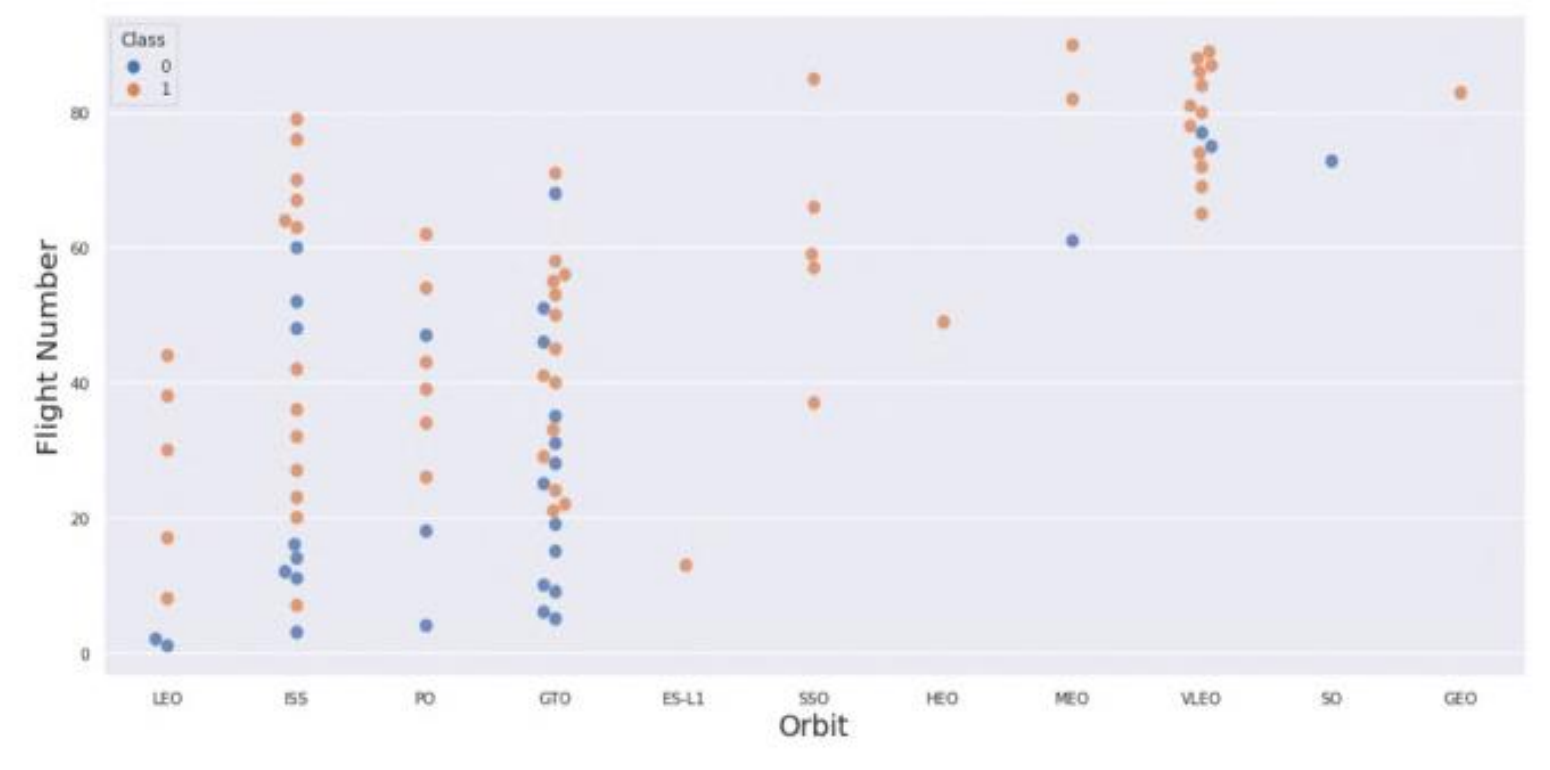
Flight Number vs. Orbit Type



The scatter plot above allows us to observe the Flight Number vs. Orbit type. As such we can visibly identify the LEO orbit and how success is related to the number of flights. GTO visibly indicates there is no correlation between flight number and the orbit attribute.

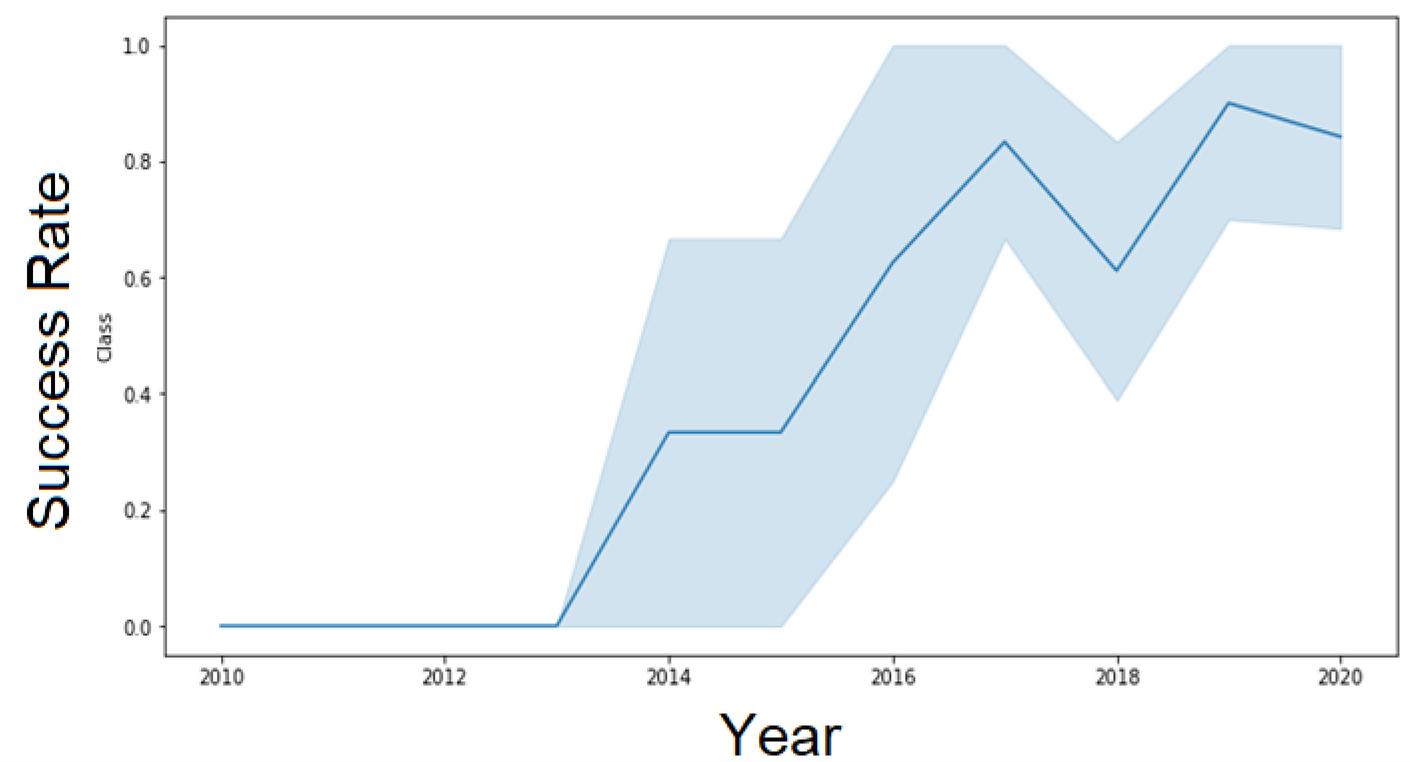
Payload vs. Orbit Type

- Per the scatter point chart to the right it can be evidenced that heavier payload had positive impact on orbits LEO, ISS and P0
- In contrast the chart displays negative impacts on the MEO and VLEO orbits
- GTO orbit data displays no relationship between the attributes.
- Data related to orbits SO, GEO and HEO require further analysis to identify any patterns or trends.



Launch Success Yearly Trend

It is observed from the scatter plot that success rates since 2013 continued increasing thru 2020.



All Launch Site Names

In [5]:
```
%sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEX;
```

Out[5]:

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

The above SQL query required using the Jupyter notebook line Magic command simplifying working with Python and useful for data analysis. The results retrieved are individual launch sites stored in the solution's database and were originally extracted from the SpaceX public data access website.

Launch Site Names Begin with 'CCA'

In [11]:

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

Out[11]:

	date	time	boosterversion	launchsite	payload	payloadmasskg	orbit	customer	missionoutcome	landingoutcome
0	2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	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
3	2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Python SQL provides:

- Cell magics: start with a double %% sign and apply to the entire cell
- Line magics: start with a single % (percent) sign and apply to a particular line in a cell
- Their usage simplifies executing queries and in the above example the code was written to retrieve launch sites with names beginning with ‘CCA’. It also restricted the number of rows to 5 and filtered columns by date, time, booster version, launchsite, payload, payloadmasskg, orbit, customer, missionoutcome and landingoutcome.

Total Payload Mass

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS "Total Payload Mass by NASA (CRS)
```

Total Payload Mass by NASA (CRS)
45596

The above query was coded to return the Total Payload Mass for SpaceX. According to suborbital space industry standards for a rocket to reach orbit, its payload should be limited to 6% of the rocket’s total mass. The engines, fuel tanks and other should be 3% with the fuel at 91%. Any reductions of mass can markedly reduce the amount of propellant needed and this feature should be monitored as a key cost factor.

Average Payload Mass by F9 v1.1

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS "Average Payload Mass by Booster
WHERE BOOSTER_VERSION = 'F9 v1.1';
```

Average Payload Mass by Booster Version F9 v1.1
2928

The above SQL code uses the SQL Avg() function to calculate the average payload mass carried by booster version F9 v1.1. The result was 2928.4 kg.

First Successful Ground Landing Date

Below, the data analyst uses the SQL min() function on column Date. Min() is an aggregate function that returns a numeric value from a set of data and used here to find the dates of the first successful landing outcome on ground pad, December 22, 2015.

```
In [12]: task_5 = '''
                 SELECT MIN(Date) AS FirstSuccessfull_landing_date
                 FROM SpaceX
                 WHERE LandingOutcome LIKE 'Success (ground pad)'
                 '''
         create_pandas_df(task_5, database=conn)
```

Out[12]:

	firstsuccessfull_landing_date
0	2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

Our team easily calculated the successful drone ship landings where payloads were between 4,000 and 6,000 kg using the Python's built-in SQL environment and the SQL language.

In [12]:
```
task_6 = '''
        SELECT BoosterVersion
        FROM SpaceX
        WHERE LandingOutcome = 'Success (drone ship)'
            AND PayloadMassKG > 4000
            AND PayloadMassKG < 6000
        '''
create_pandas_df(task_6, database=conn)
```

Out[12]:

	boosterversion
0	F9 FT B1022
1	F9 FT B1026
2	F9 FT B1021.2
3	F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

```
%sql SELECT COUNT(MISSION_OUTCOME) AS "Successful Mission" FROM SPACEX WHERE MISSION_OUTCOME LIKE 'Success%';
```

Successful Mission
100

```
%sql SELECT COUNT(MISSION_OUTCOME) AS "Failure Mission" FROM SPACEX WHERE MISSION_OUTCOME LIKE 'Failure%';
```

Failure Mission
1

The above SQL code contains a wildcard character where it is used to substitute one or more characters in a string. Wildcard characters are used with the LIKE operator. Here, the LIKE operator is used in a WHERE clause to search for the specified patterns 'Success' and 'Failure' in the MISSION_OUTCOME column.
There were 100 successful missions and 1 failure per the results.

Boosters Carried Maximum Payload

```
%sql SELECT DISTINCT BOOSTER_VERSION AS "Booster Versions which carried the Maximum Payload Mass" FROM SPACEX
WHERE PAYLOAD_MASS__KG_ =(SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEX);
```

Booster Versions which carried the Maximum Payload Mass
F9 B5 B1048.4
F9 B5 B1048.5
F9 B5 B1049.4
F9 B5 B1049.5
F9 B5 B1049.7
F9 B5 B1051.3
F9 B5 B1051.4
F9 B5 B1051.6
F9 B5 B1056.4
F9 B5 B1058.3
F9 B5 B1060.2
F9 B5 B1060.3

To determine the booster having carried the maximum payload a SQL subquery was used. The code on the left displays the WHERE clause and the MAX() function usage and all results.

2015 Launch Records

Our analyst used a combination of the WHERE clause, LIKE, AND, and BETWEEN conditions to filter for failed landing outcomes in a drone ship. It also included their booster versions and launch site names for the year 2015. See results below:

In [18]:
```
task_9 = '''
        SELECT BoosterVersion, LaunchSite, LandingOutcome
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Failure (drone ship)'
            AND Date BETWEEN '2015-01-01' AND '2015-12-31'
        '''
create_pandas_df(task_9, database=conn)
```

Out[18]:

	boosterversion	launchsite	landingoutcome
0	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
1	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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

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

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

- The client requested the team rank landing outcomes by a specific date range. The above SQL code begins by selecting Landing outcomes and then uses the COUNT() function to calculate the number of landing outcomes from the result set. Additionally, the WHERE clause further filters for landing outcomes BETWEEN 2010-06-04 to 2010-03-20.
- The GROUP BY aggregate function had to be used to group the landing outcomes and the ORDER BY clause to order the grouped landing outcome in descending order

Section 3

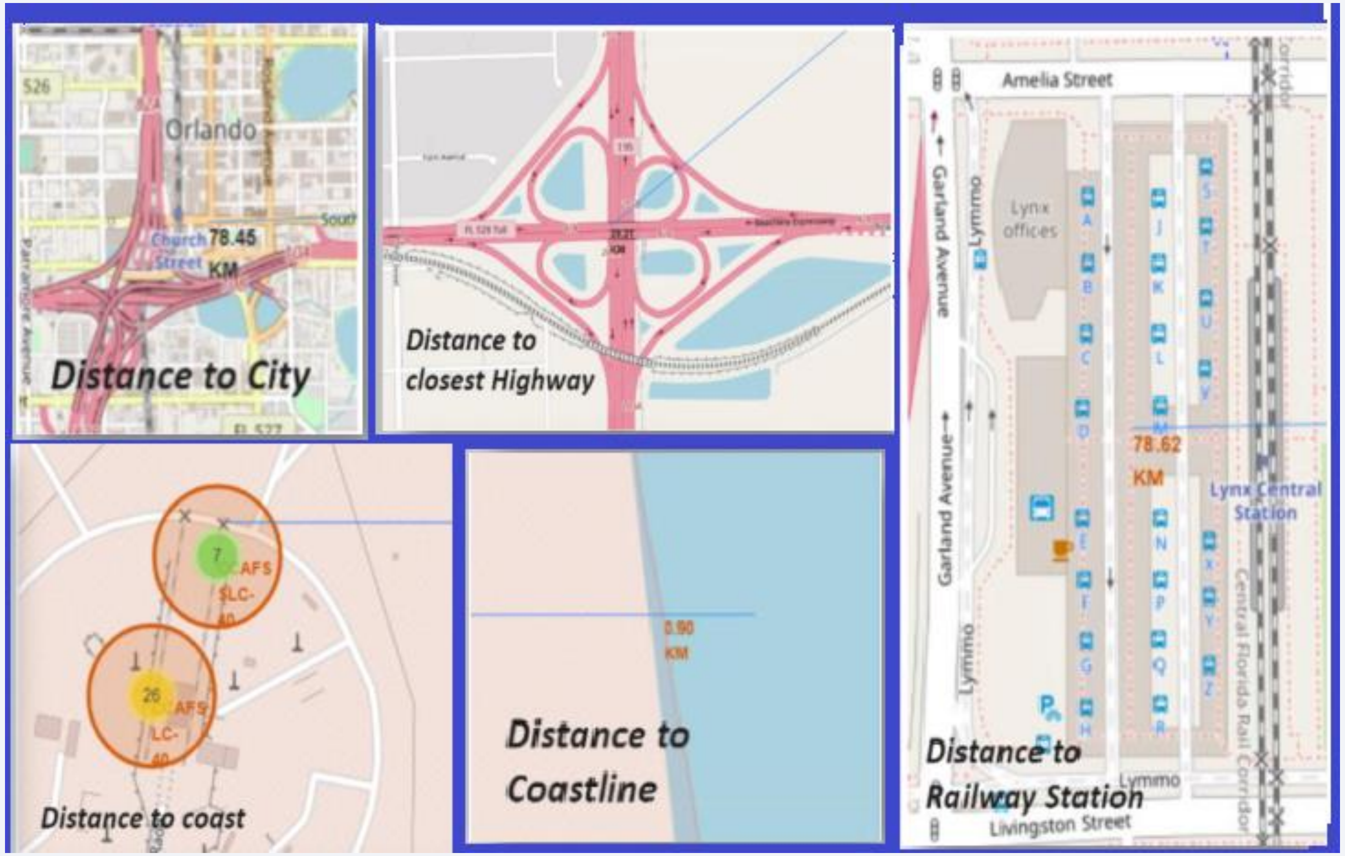
Launch Sites Proximities Analysis

Launch Site Locations

- All launch sites are located within the USA.
- An East Coast location is preferred because any rockets leaving Earth's surface and traveling eastward get a boost from the Earth's west-to-east spin.
- Evidently, Vandenberg's location and capabilities make it a valuable launch site for certain types of missions, particularly those that require polar or near-polar orbits

Launch sites with Marker color labels Showing Success or Failures

Launch Sites vs. 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

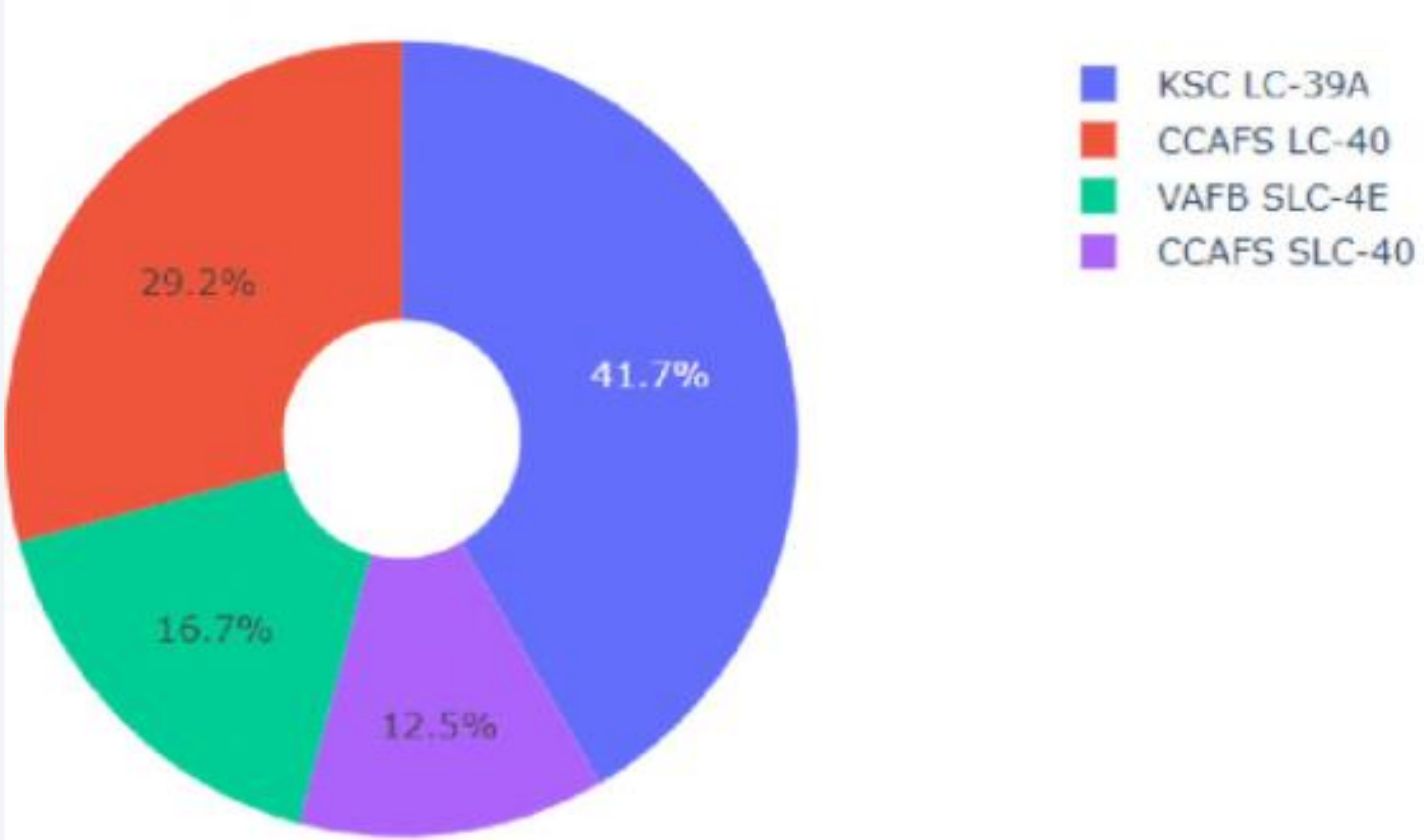
Per the data analysis in this project, rockets often crash (approx.5%). So, their path must be over uninhabited areas and the oceans are considered a large safety zone.

Section 4

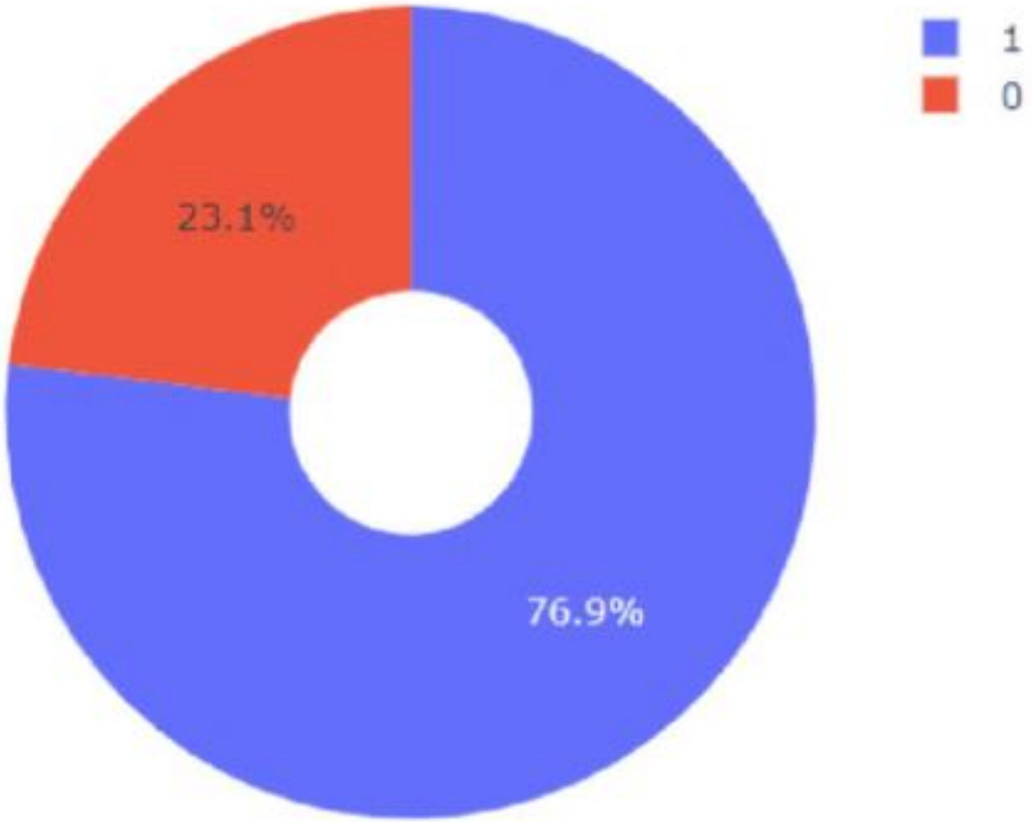
Build a Dashboard with Plotly Dash

Pie Chart Launch Sites Featuring Success Rates

- Launch site KSC holds the largest percentage of successful launches followed by:
  CCAFS – East Coast
  VAFB – West Coast
  CCAFS – East Coast



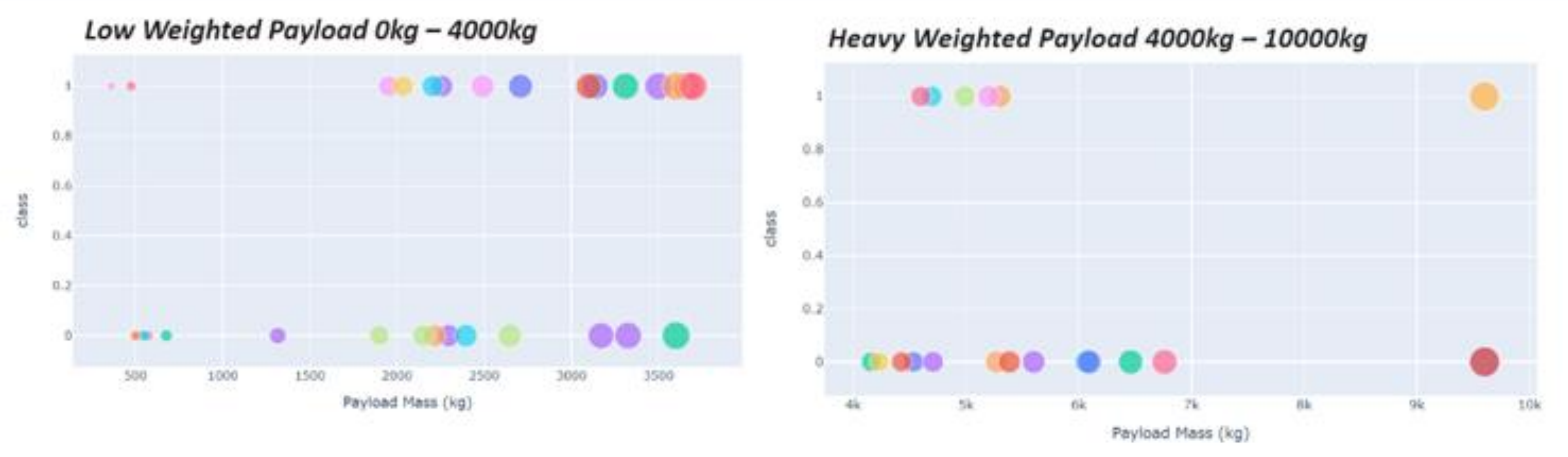
Highest Launch-Success Ratio



The highest launch-success ratio: KSC LC-39A - 76.9 %

Failures - 23.1 %

Scatter Plot with Payload vs Launch Outcome All sites using payload selected in Range Slider



Note: Success rates for low payload weights are greater than heavy loads

Section 5

Predictive Analysis (Classification)

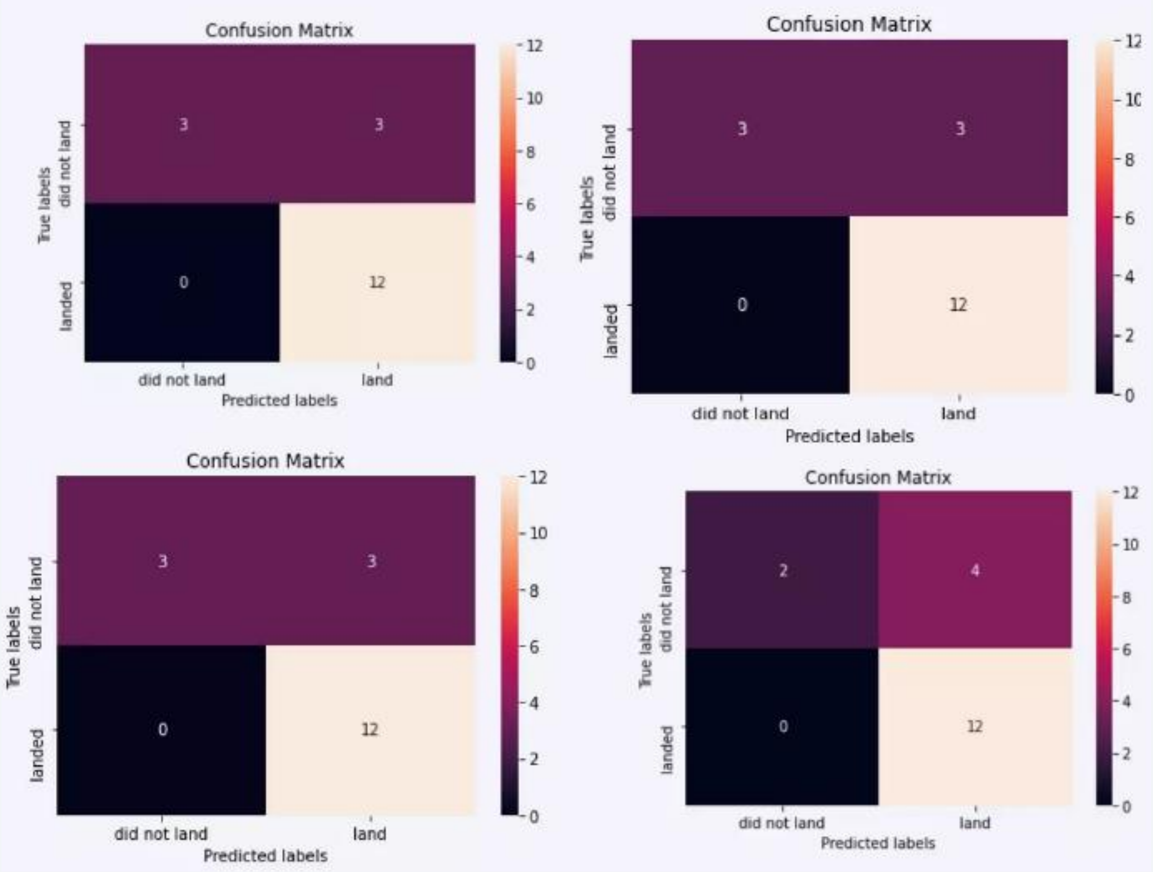
Classification Accuracy

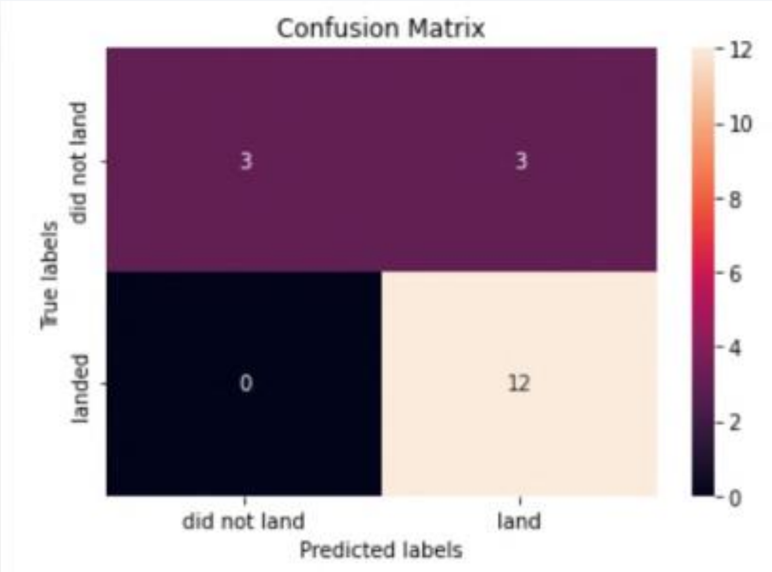
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

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

```
Best model is DecisionTree with a score of 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

Per the code above, the results using this algorithm identified the Decision Tree achieved the highest classification accuracy with a score of 0.873

Confusion Matrix





As shown by the Confusion Matrix diagram, the decision tree classifier distinguishes the different classes. However, false positives are a big issue and must be resolved with further analysis.

Conclusions

- Beginning 2013, SpaceX launch rate success increased. This linear observation has competitive significance. At this progression SpaceX will eventually perfect their launches.
- The Decision Tree Algorithm used by the Machine Learning model was the best.
- Low payload weights, less than 4000kg, performed better than heavier loads.
- KSC LC-39A should be monitored closely as they had the most successful launches.
- Due to time restrictions the reports in this solution are minimal. With further analysis far more data can be analyzed providing SpaceY with a ML model capable of offering strategic recommendations.

Thank you!