IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Fotis Tasioulis
06/12/2022

Outline

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

Executive Summary

Summary of methodologies

Data Collection through API

Data Collection with Web Scraping

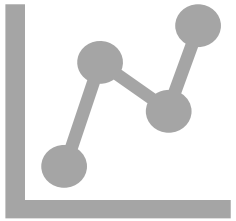
Data Wrangling

Exploratory Data Analysis with SQL

Exploratory Data Analysis with Data Visualization

Interactive Visual Analytics with Folium

Machine Learning Prediction



Summary of all results

Exploratory Data Analysis result

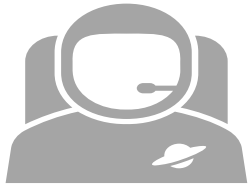
Interactive analytics in screenshots

Predictive Analytics result

Introduction

Project background and context

Space X advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars - other providers cost upward of 165 million dollars each, much of the savings is because Space X can reuse the first stage. In this project, I will help my company Space Y which is a competitor to determine the price of each launch by gathering information and creating dashboards for my team. The target is to determine if Space X will re-use the first stage using a machine learning model and public information. This will help my company to bid against Space X for a rocket launch.



Problems you want to find answers

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

What factors determine if the rocket will land successfully?

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

Section 1

Methodology

Methodology

- Executive Summary
- Data collection methodology:
  - 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
- Perform interactive visual analytics using Folium
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

- The data was collected using the following methods:
  - Used the get request to the SpaceX API.
  - Decoded the response content as a JSON using .json() function call and turned it into a pandas dataframe using .json_normalize().
  - Cleaned the data, checked for missing values, and filled missing values where required.
  - Performed web-scraping from Wikipedia for Falcon 9 launch records using BeautifulSoup.
  - The objective was to extract the launch records as an HTML table, parse the table and convert it to a pandas data frame for future analysis.

Data Collection – SpaceX API

- Collection of data with the get request, data formatting and cleaning.

- Notebook link: https://github.com/ftasioulis/Data-Science-Capstone-Project/blob/f4e94ce5eb9a681a4d8271f156e3f2d6f05b6ee9/SpaceX%20-%20Data%20Collection.ipynb.

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

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

Check the content of the response

Task 1: Request and parse the SpaceX launch data using the GET request

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

In [9]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

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

In [ ]:
```
response.status_code
```

Out[ ]:
```
200
```

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

In [13]:
```
# Use json_normalize meethod to convert the json result into a dataframe
data=pd.json_normalize(response.json())
```

Task 3: Dealing with Missing Values

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

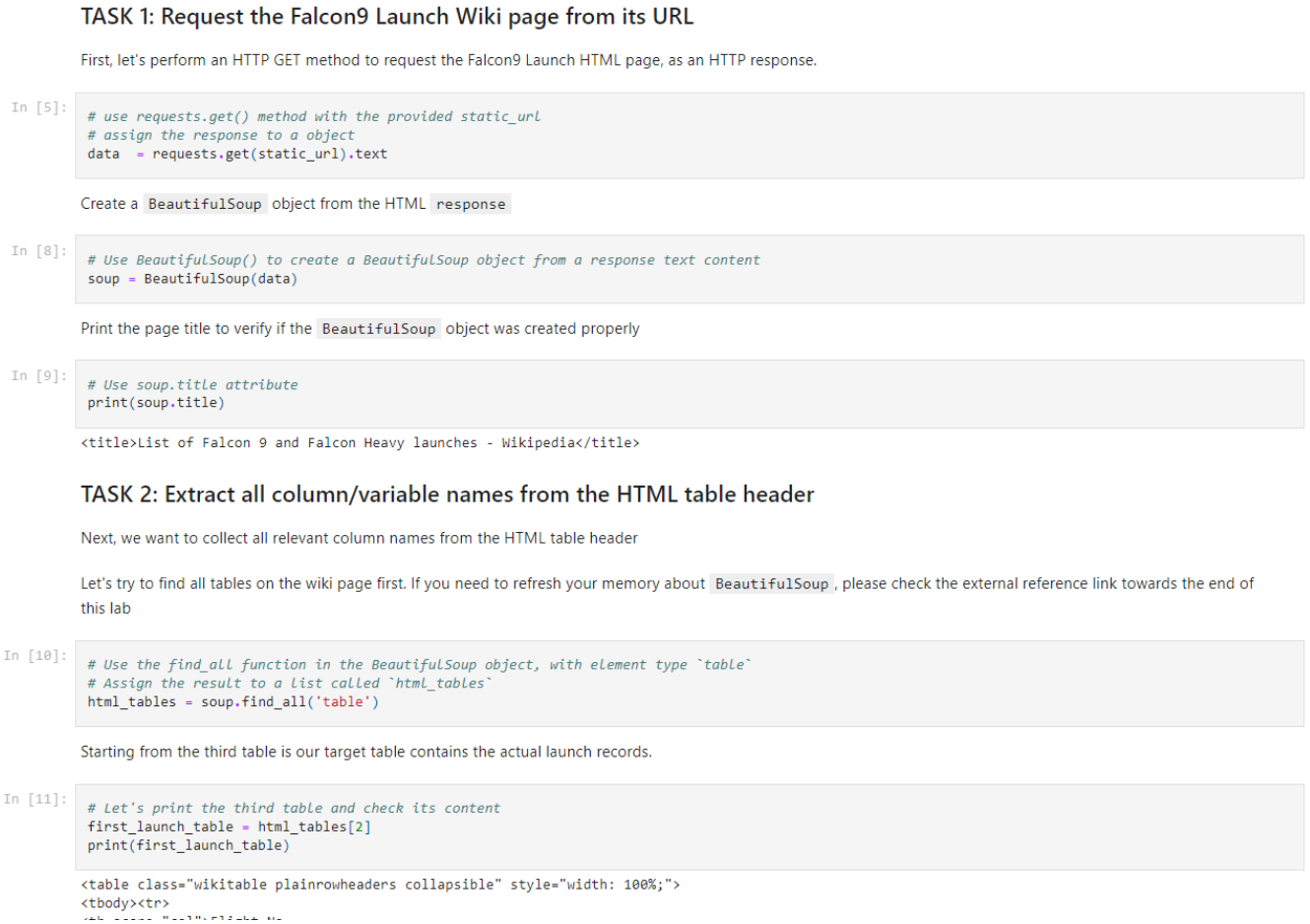
3]:
```
# Calculate the mean value of PayloadMass column
avg=data_falcon9['PayloadMass'].mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].replace(np.nan, avg, inplace=True)
```

```
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/pandas/core/generic.py:6619: SettingWithCopyWarning:
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_guide/indexing.html#returning-a-view-versus-a-copy
  return self._update_inplace(result)
```

Data Collection - Scraping

- Applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup.
- Parsed the table and converted it to a pandas dataframe.
- Notebook link: https://github.com/ftasioulis/Data-Science-Capstone-Project/blob/f4e94ce5eb9a681a4d8271f156e3f2d6f05b6ee9/SpaceX-webscraping.ipynb.



TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

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

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

In [8]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(data)
```

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

In [9]:
```
# Use soup.title attribute
print(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 the external reference link towards the end of this lab

In [10]:
```
# 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.

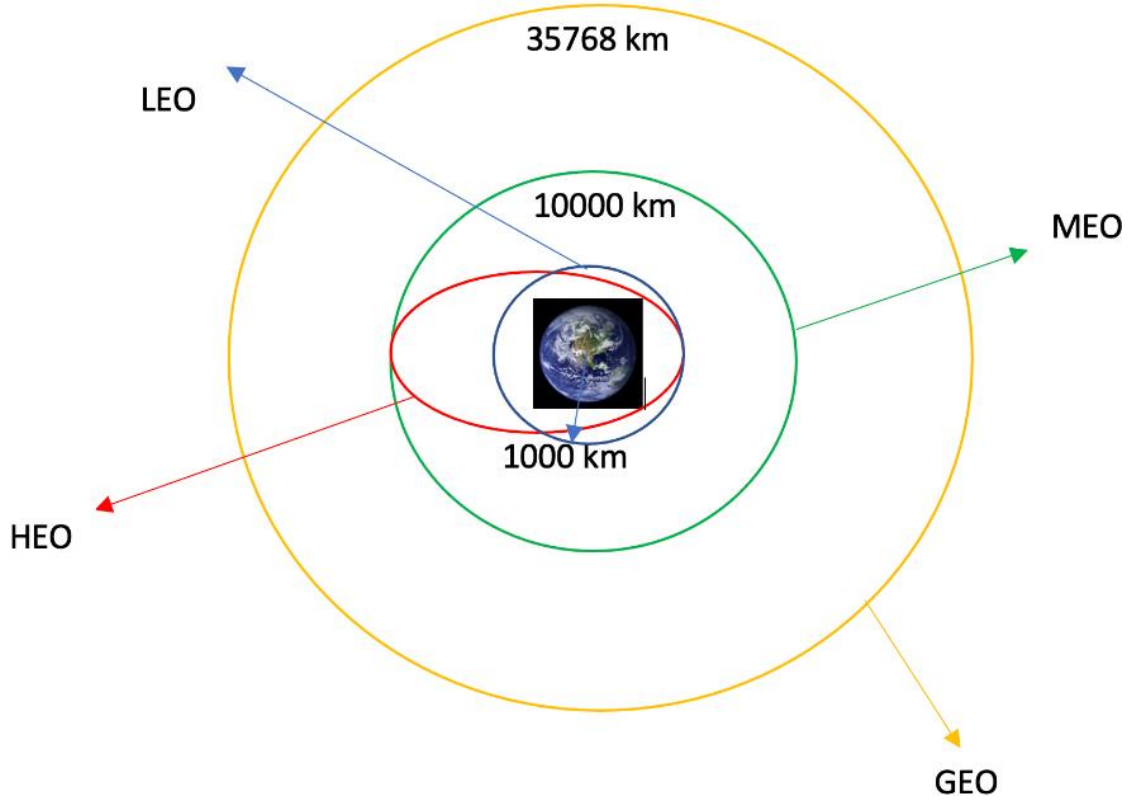
In [11]:
```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

```
<table class="wikitable plainrowheaders collapsible" style="width: 100%;">
<tbody><tr>
```

Data Wrangling

- Performed exploratory data analysis and determined the training labels.
- Calculated the number of launches at each site - number and occurrence of each orbit.
- Created a landing outcome label from the Outcome column.

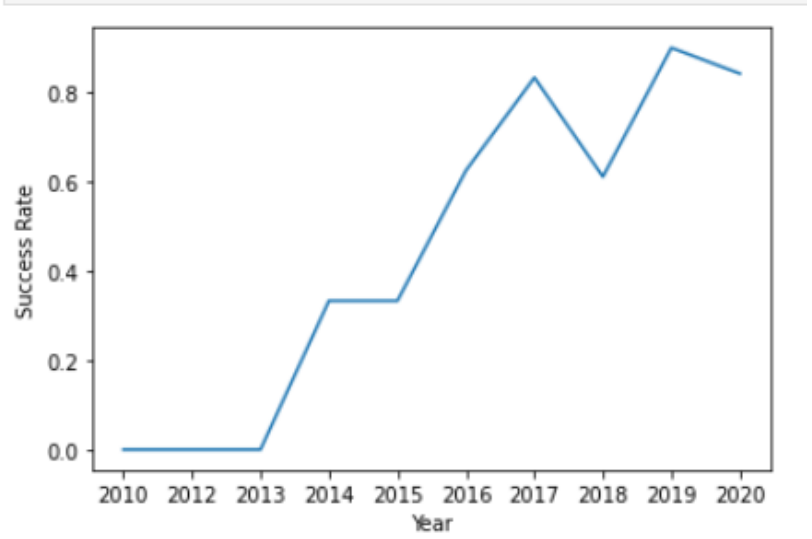
- Notebook link: https://github.com/ftasioulis/Data-Science-Capstone-Project/blob/f4e94ce5eb9a681a4d8271f156e3f2d6f05b6ee9/SpaceX-Data%20wrangling.ipynb.

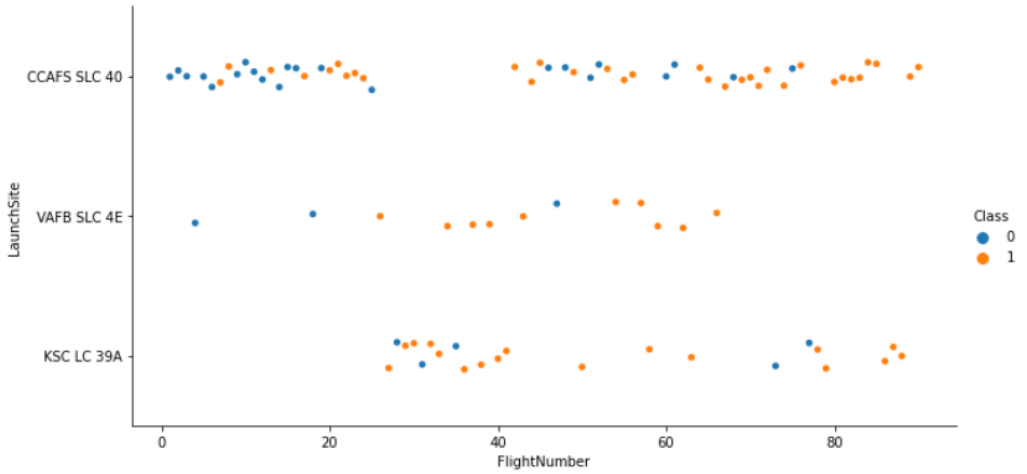


EDA with Data Visualization

- Performed exploratory data analysis and feature engineering using Pandas and Matplotlib.
- Visualized relationships between Flight Number, Launch Site, Payload, the success rate of each orbit type, and launch success yearly trend.

- Notebook link: https://github.com/ftasioulis/Data-Science-Capstone-Project/blob/439136a95fe460e97393aec299ca073e9ad795e2/SpaceX-EDA%20Data%20Visualization.ipynb.





EDA with SQL

- Loaded the SpaceX dataset into a PostgreSQL database.
- Executed the following queries:
  - Displayed the names of the unique launch sites in the space mission.
  - Displayed 5 records where launch sites begin with the string ‘CCA’.
  - Displayed the total payload mass carried by boosters launched by NASA.
  - Displayed average payload mass carried by booster version F9 v1.1.
  - Listed the date when the first successful landing outcome in the ground pad was achieved.
  - Listed the names of the boosters which have success in drone ships and have payload mass greater than 4000 but less than 6000 kg.
  - Listed the total number of successful and failed mission outcomes.
  - Listed the names of the booster versions which have carried the maximum payload mass.
  - Listed the records which will display the month names, failures, booster versions, and launch sites for the months in year 2015.
  - Ranked the count of successful landing outcomes between the date 04-06-2010 and 20-03-2017 in descending order.
- Notebook link: https://github.com/ftasioulis/Data-Science-Capstone-Project/blob/439136a95fe460e97393aec299ca073e9ad795e2/SpaceX%20EDA%20with%20SQL.ipynb.

Build an Interactive Map with Folium

Added map objects such as markers, circles, and lines.

Target was to find an optimal location for building a launch site by analyzing the data from the existing launch site locations.

Objectives were the following:

Mark all launch sites on the folium map.

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

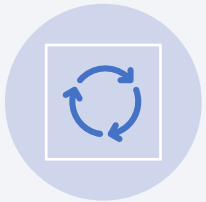
Calculate the distances between a launch site to its proximities.

Notebook link:
https://github.com/ftasioulis/Data-Science-Capstone-Project/blob/439136a95fe460e97393aec299ca073e9ad795e2/SpaceX-Launch%20Sites%20Locations%20Analysis%20with%20Folium.ipynb.

Predictive Analysis (Classification)

Created a machine learning model pipeline to predict if the first stage will land.

Loaded the data using NumPy and Pandas libraries, transformed the data, and split them into training and testing data to improve the accuracy of the models.



Used different machine learning models (SVM, Classification trees, and Logistic Regression) and tuned different hyperparameters.

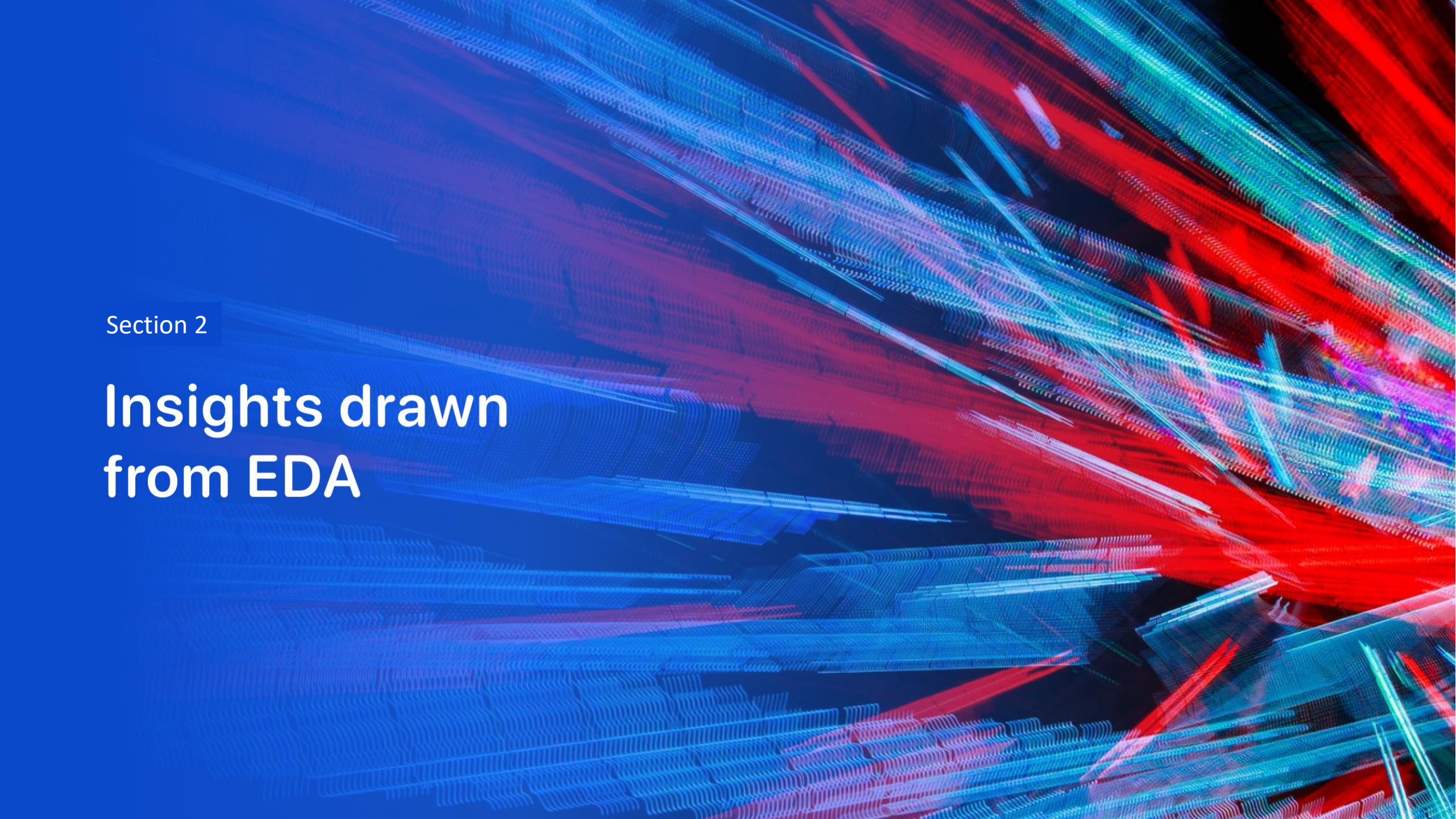
Found which method performs better by considering accuracy.

Best model was the Decision Tree with an accuracy of approximately 89%.

Notebook link: https://github.com/ftasioulis/Data-Science-Capstone-Project/blob/0c6d6166af66883d484970fc7f97f834988f5e2f/SpaxeX-Predictive%20Analysis%20(Classification).ipynb.

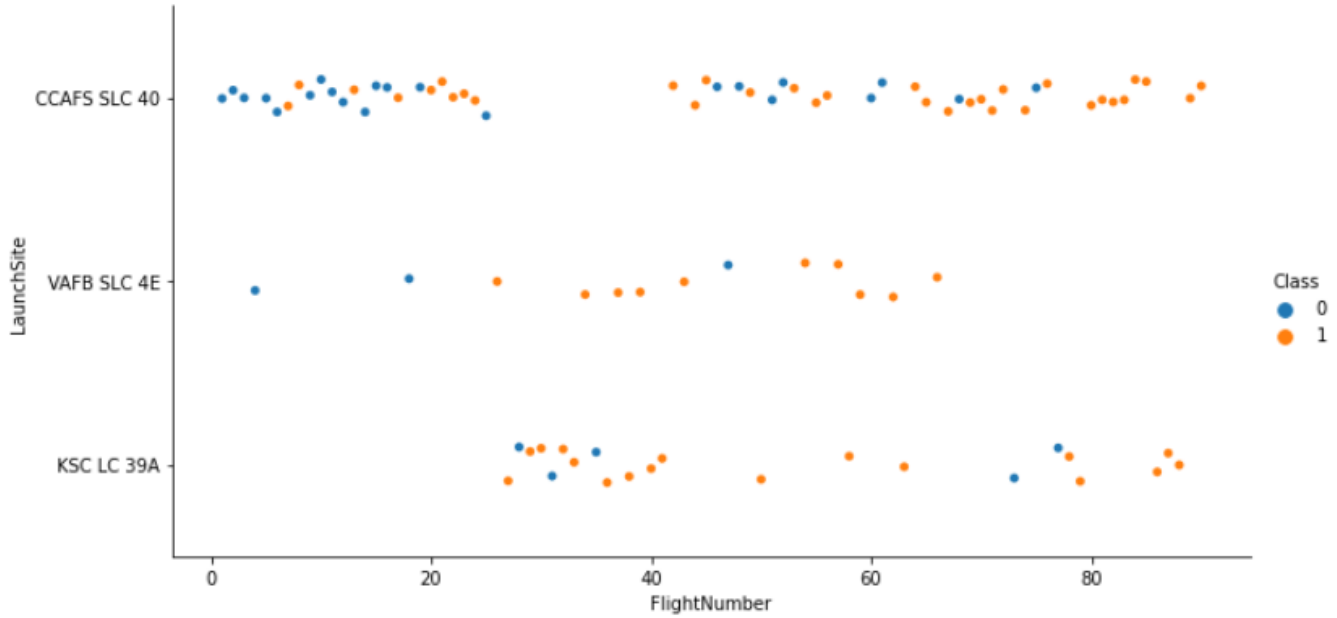
Results

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


Section 2
Insights drawn from EDA

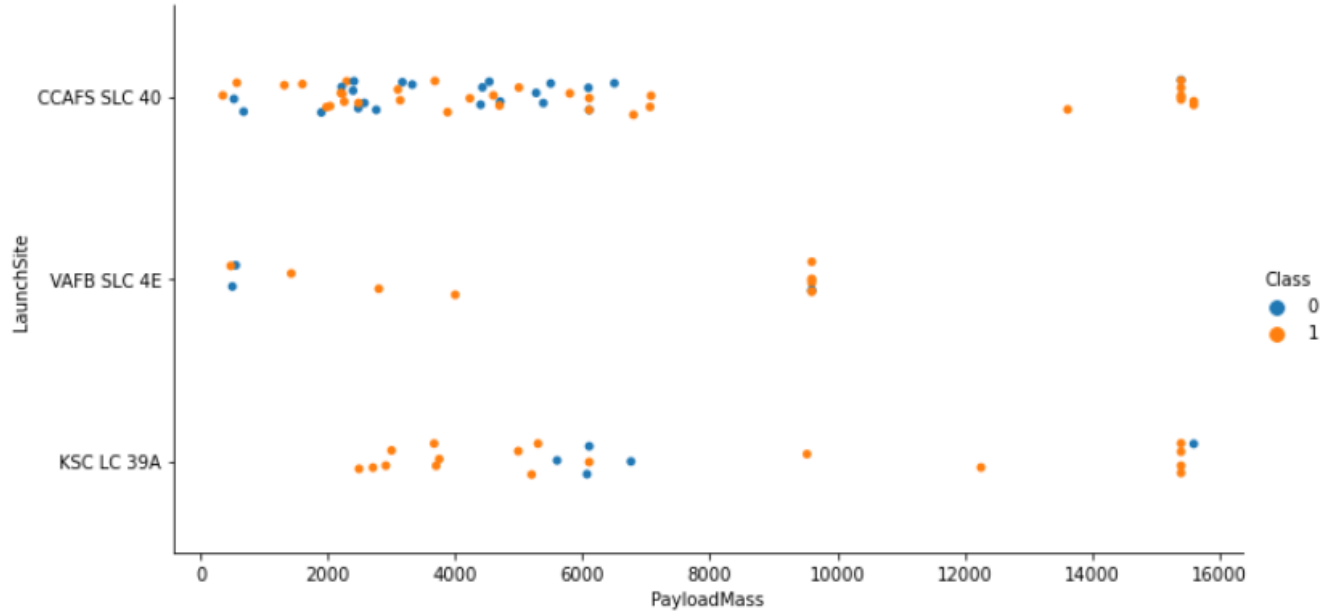
Flight Number vs. Launch Site

- It can be noticed that as the flight number from a launch site increases the success rate from a launch site improves as well.



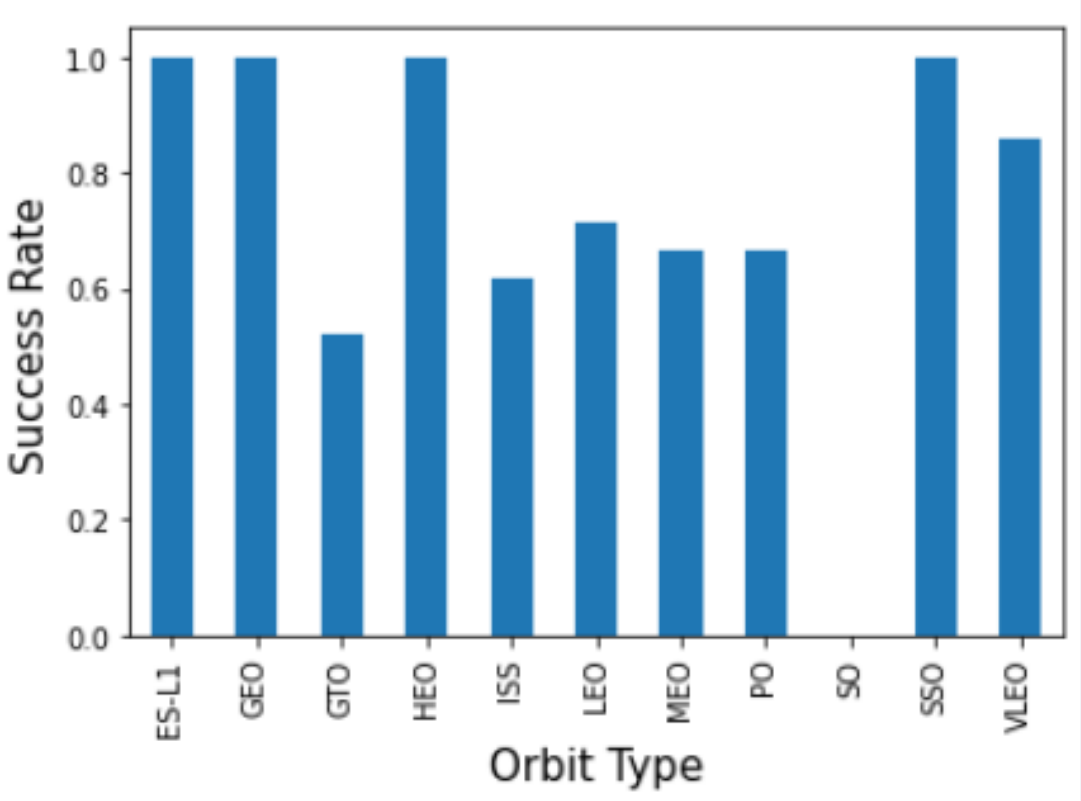
Payload vs. Launch Site

- It can be noticed that for the VAFB-SLC launch site there are no rockets launched for heavy upload mass greater than 10000.



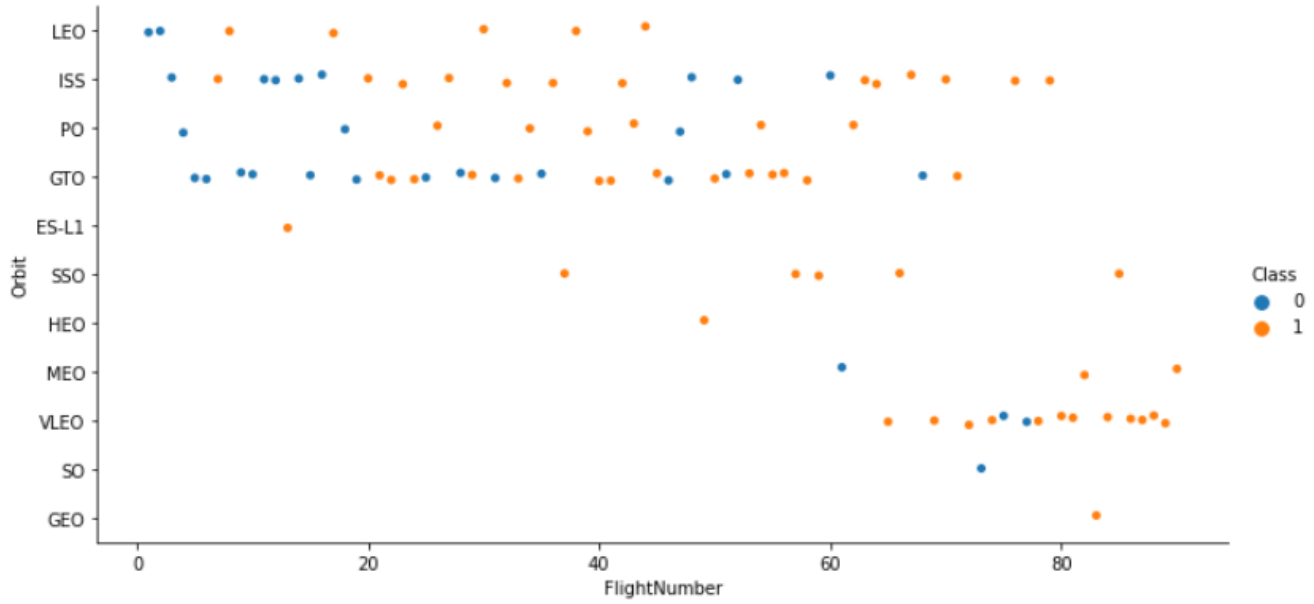
Success Rate vs. Orbit Type

- It can be noticed that ES-L1, GFO, HEO, and SSO - Orbit Types have the best success rates.



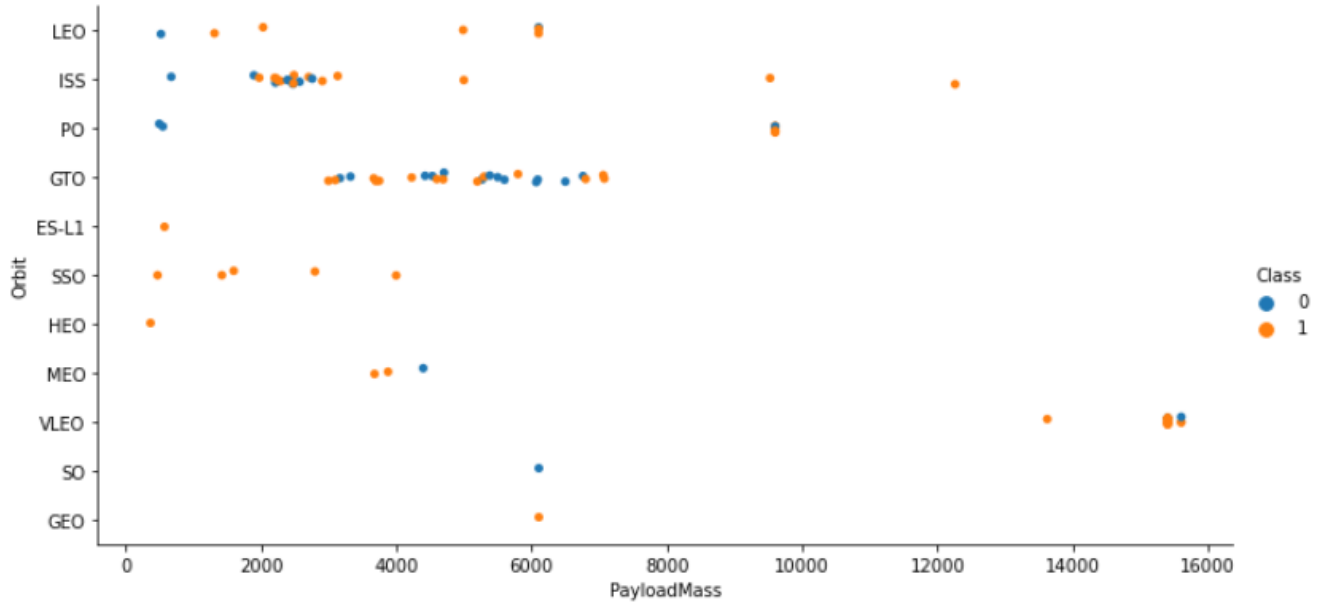
Flight Number vs. Orbit Type

- It can be noticed that in LEO Orbit the success appears related to the number of flights whereas when in GTO orbit there seems to be no relationship between the flight number.



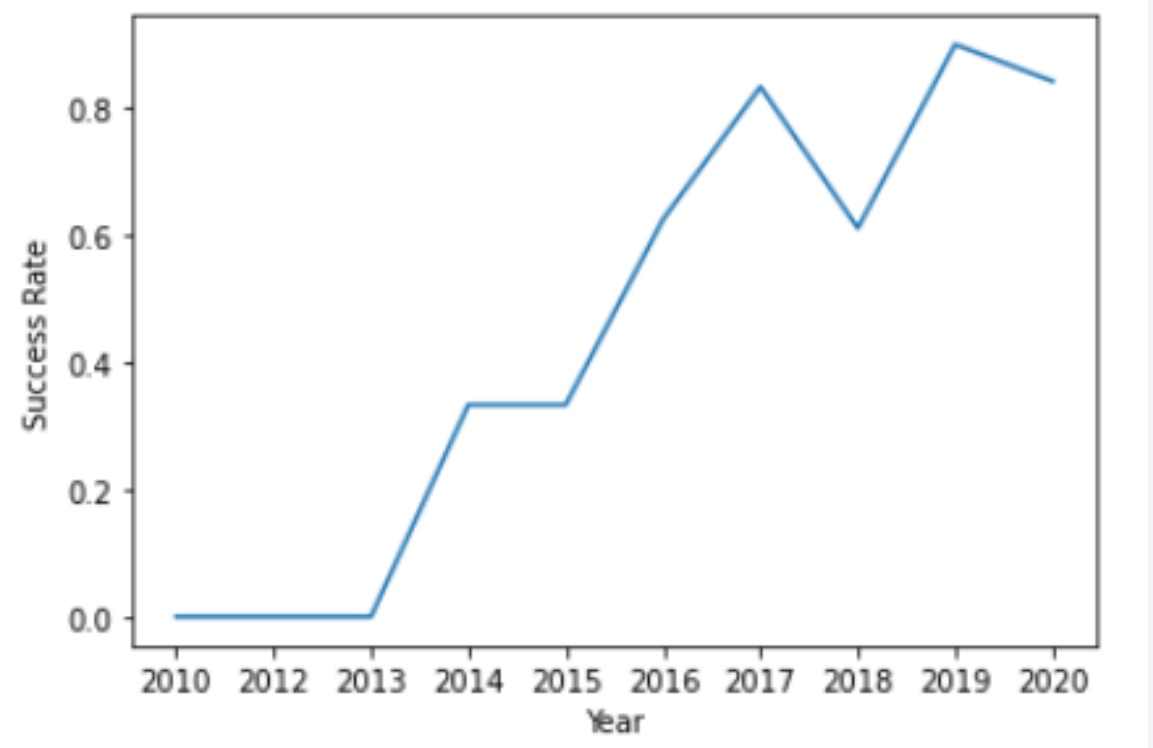
Payload vs. Orbit Type

- It can be noticed that with heavy uploads the successful landing is more for POLAR, LEO, and ISS.
- For GTO it cannot be distinguished well as positive and negative landing are both together.



Launch Success Yearly Trend

- It can be noticed that the success rate since 2013 kept on increasing till 2020.



All Launch Site Names

- Used the keyword unique to find the different launch sites from the Space X table.
- Displayed the names of the unique launch sites in the space mission.

```
%sql select unique(launch_site) from spacextbl
```

```
 * ibm_db_sa://lxt60102:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

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

Launch Site Names Begin with 'CCA'

- Used the query to display 5 records where launch sites begin with the string ‘CCA’.

```
%sql select * from spacextbl where launch_site like 'CCA%' limit 5;
```

```
 * ibm_db_sa://lxt60102:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

DATE	time_utc_	booster_version	launch_site	payload	payload_mass_kg_	orbit	customer	mission_outcome	landing_outcome
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-08-12	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)
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-12	22:41:00	F9 v1.1	CCAFS LC-40	SES-8	3170	GTO	SES	Success	No attempt

Total Payload Mass

- Displayed the total payload mass carried by boosters launched by NASA (CRS).

```
%sql select sum(payload_mass__kg_) as total_payload_mass from spacextbl where customer ='NASA (CRS)';
```

```
 * ibm_db_sa://lxt60102:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

total_payload_mass
22007

Average Payload Mass by F9 v1.1

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

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

```
%sql select avg(payload_mass__kg_) as avg_pay_load_mass from spacextbl where booster_version='F9 v1.1'
```

```
 * ibm_db_sa://lxt60102:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

avg_pay_load_mass
3676

First Successful Ground Landing Date

- Displayed the date when the first successful landing outcome on the ground pad was achieved.

```
%sql select distinct(landing__outcome) from spacextbl;
```

```
 * ibm_db_sa://lxt60102:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

landing__outcome
Controlled (ocean)
Failure
Failure (drone ship)
Failure (parachute)
No attempt
Success
Success (drone ship)
Success (ground pad)

```
%sql select min(Date) as first_succesful_landing from spacextbl where landing__outcome='Success (ground pad)';
```

```
 * ibm_db_sa://lxt60102:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

first_succesful_landing
2017-01-05

Successful Drone Ship Landing with Payload between 4000 and 6000

- Displayed 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 as name_of_boosters from spacextbl where landing__outcome='Success (drone ship)' and payload_mass__kg_>4000 and payload_mass__kg_<6000;
 * ibm_db_sa://lxt60102:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
]:
```

name_of_boosters
F9 FT B1022
F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

- Displayed the total number of successful and failed mission outcomes.

```
%sql select distinct(mission_outcome) from spacextbl;
```

```
 * ibm_db_sa://lxt60102:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

mission_outcome
Success
Success (payload status unclear)

```
%sql select count(mission_outcome) as success_outcome from spacextbl where mission_outcome='Success' or mission_outcome='Success (payload status unclear)';
```

```
 * ibm_db_sa://lxt60102:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

success_outcome
45

```
%sql select count(mission_outcome) as failure_outcome from spacextbl where mission_outcome='Failure';
```

```
 * ibm_db_sa://lxt60102:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

failure_outcome
0

Boosters Carried Maximum Payload

- Displayed the names of the booster versions which have carried the maximum payload mass.

```
%sql select booster_version, payload_mass__kg_ from spacextbl where payload_mass__kg_=(select max(payload_mass__kg_) from spacextbl) order by booster_version;
```

```
 * ibm_db_sa://lxt60102:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

booster_version	payload_mass__kg_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1049.5	15600
F9 B5 B1058.3	15600
F9 B5 B1060.2	15600

2015 Launch Records

- Listed the records which will display the month names, failure landing outcomes in drone ship, booster versions, and launch site for the months in the year 2015.

```
%sql select booster_version, launch_site, landing__outcome from spacextbl where landing__outcome='Failure (drone ship)' and DATE between '2015-01-01' and '2015-12-31';
```

```
 * ibm_db_sa://lxt60102:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

booster_version	launch_site	landing__outcome
F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)

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

- Ranked the count of successful landing outcomes between the above dates in descending order.

```
%sql select landing__outcome, count(landing__outcome) from spacextbl where date between '2010-06-04' and '2017-03-20' group by landing__outcome order by count(landing__outcome) desc
```

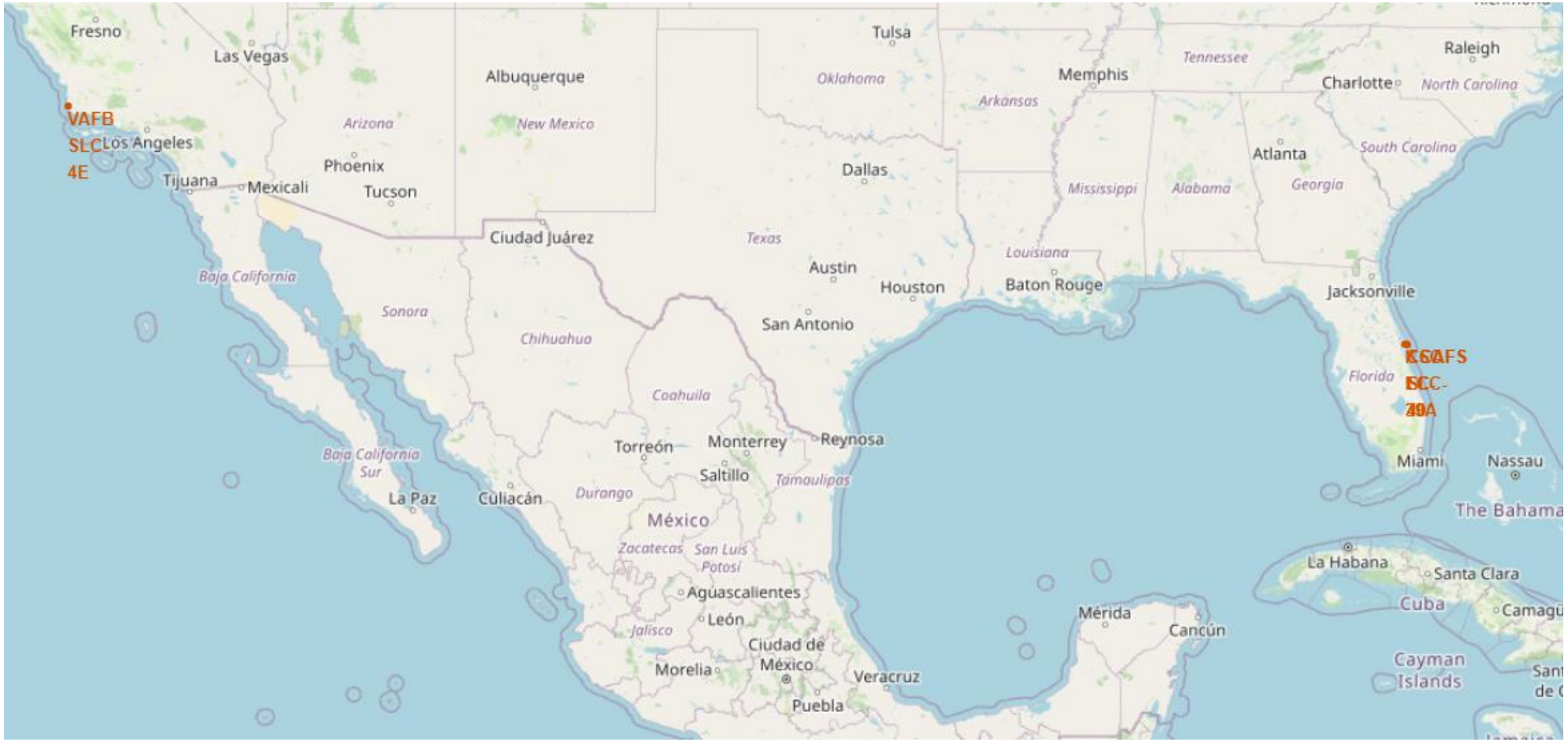
```
 * ibm_db_sa://lxt60102:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.
```

landing__outcome	2
No attempt	7
Failure (drone ship)	2
Success (drone ship)	2
Success (ground pad)	2
Controlled (ocean)	1
Failure (parachute)	1

Section 3

Launch Sites Proximities Analysis

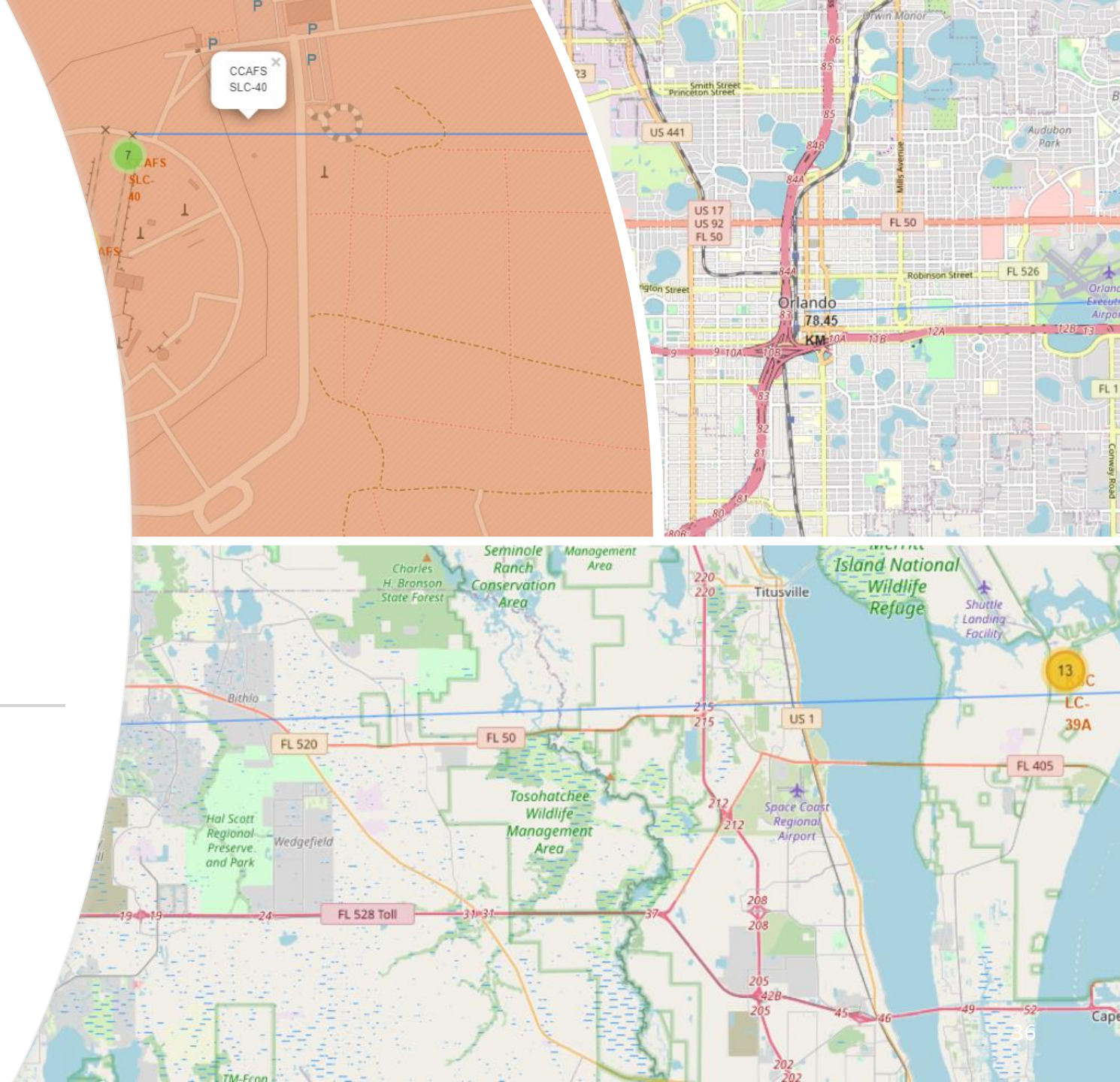
Space X Launch Sites



Markers Indicating Successful and Failed Launches

- Red Colour = Fail
- Green = Success

Launch Site Distance to Landmarks



Section 5

Predictive Analysis (Classification)

Classification Accuracy

- The Decision Tree classification model had the best accuracy.

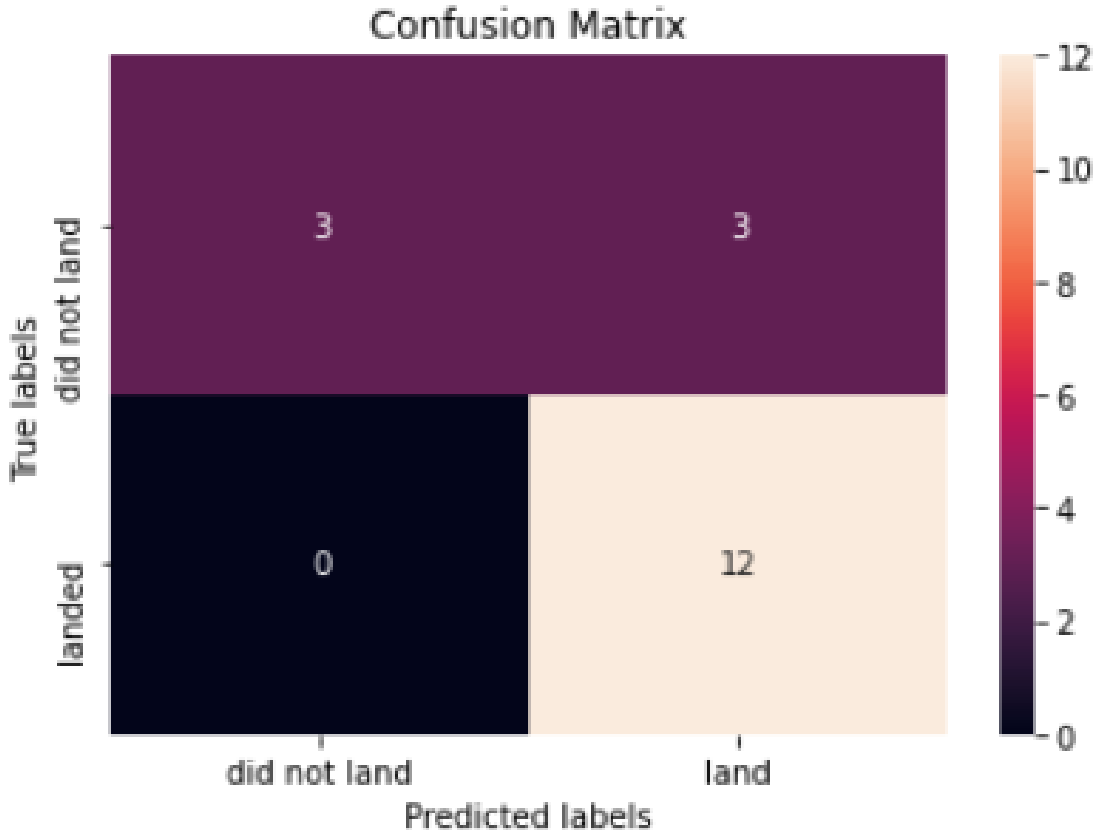
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

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 0.8888888888888888
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 2, 'splitter': 'random'}
```

Confusion Matrix

- The confusion matrix for the model shows that the classifier is able to separate the different classes.



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 from 2013 till 2020.
- Orbits ES-L1, GEO, HEO, and SSO had the best success rate.
- KSC LC-39A had the most successful launches of any launch site.
- The Decision Tree classifier had the highest accuracy for predictions between the machine algorithms that were tested.

Appendix

```
# Pandas is a software library written for the Python programming language for
import pandas as pd
# NumPy is a library for the Python programming language, adding support for l
import numpy as np
# Matplotlib is a plotting library for python and pyplot gives us a MatLab lik
import matplotlib.pyplot as plt
#Seaborn is a Python data visualization library based on matplotlib. It provia
import seaborn as sns
# Preprocessing allows us to standarsize our data
from sklearn import preprocessing
# Allows us to split our data into training and testing data
from sklearn.model_selection import train_test_split
# Allows us to test parameters of classification algorithms and find the best
from sklearn.model_selection import GridSearchCV
# Logistic Regression classification algorithm
from sklearn.linear_model import LogisticRegression
# Support Vector Machine classification algorithm
from sklearn.svm import SVC
# Decision Tree classification algorithm
from sklearn.tree import DecisionTreeClassifier
# K Nearest Neighbors classification algorithm
from sklearn.neighbors import KNeighborsClassifier
```

```
parameters = {'criterion': ['gini', 'entropy'],
     'splitter': ['best', 'random'],
     'max_depth': [2*n for n in range(1,10)],
     'max_features': ['auto', 'sqrt'],
     'min_samples_leaf': [1, 2, 4],
     'min_samples_split': [2, 5, 10]}

tree = DecisionTreeClassifier()
```

```
tree_cv=GridSearchCV(tree,parameters,cv=10)
tree_cv.fit(X_train,Y_train)
```

•••

```
print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_dept
accuracy : 0.8888888888888888
```

Thank you!