



IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Răzvan-Andrei Mihăescu
18th April 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. 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.

- Problems you want to find answers

- 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 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 various methods
 - Data collection was done using get request to the SpaceX API.
 - Next, we decoded the response content as a Json using `.json()` function call and turn it into a pandas dataframe using `.json_normalize()`.
 - We then cleaned the data, checked for missing values and fill in missing values where necessary.
 - In addition, we performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.
 - 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

- We used the get request to the SpaceX API to collect data, clean the requested data and did some basic data wrangling and formatting.
- The link to the notebook is <https://github.com/mihaescurazvan/IBM-Data-Science-Capstone-Project-SpaceX/blob/master/Data%20Collection%20API%20Lab.ipynb>

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 successful with the 200 status response code

```
In [10]: response.status_code
Out[10]: 200
```

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

```
In [11]: # Use json_normalize method to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

Using the dataframe `data` print the first 5 rows

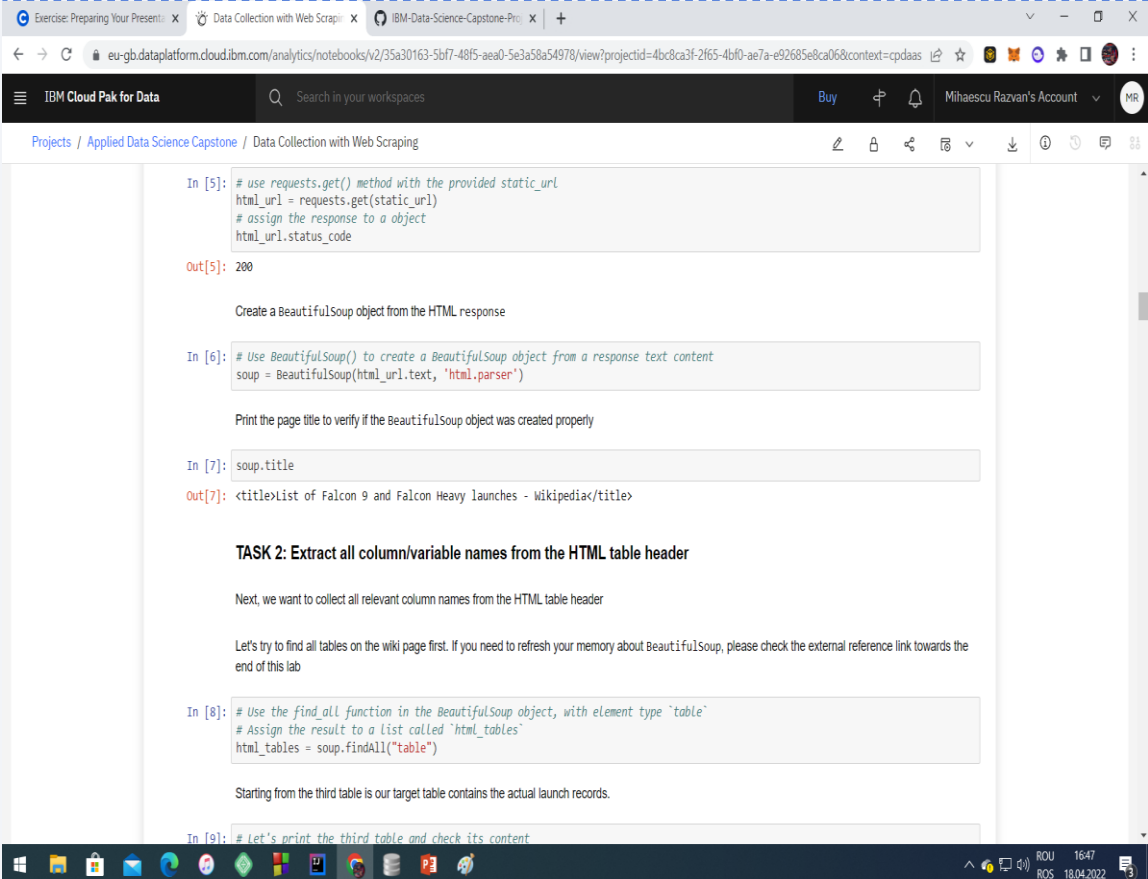
```
In [12]: # Get the head of the dataframe
data.head()
```

```
Out[12]:
```

| | static_fire_date_utc | static_fire_date_unix | net | window | rocket | success | failures | details | crew | ships | capsules |
|---|--------------------------|-----------------------|-------|--------|-------------------------|---------|---|---------|------|-------|----------|
| 0 | 2006-03-17T00:00:00.000Z | 1.142554e+09 | False | 0.0 | 5e9d0d95eda69955709d1eb | False | ["time": 33, "altitude": None, "reason": "merlin engine failure at 33 seconds and loss of vehicle"] | | | | |

Data Collection - Scraping

- We applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup
- We parsed the table and converted it into a pandas dataframe.
- The link to the notebook is <https://github.com/mihaescu-razvan/IBM-Data-Science-Capstone-Project-SpaceX/blob/master/Data%20Collection%20with%20Web%20Scraping%20.ipynb>



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

Out[5]: 200

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_url.text, 'html.parser')

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

In [7]: soup.title

Out[7]: <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

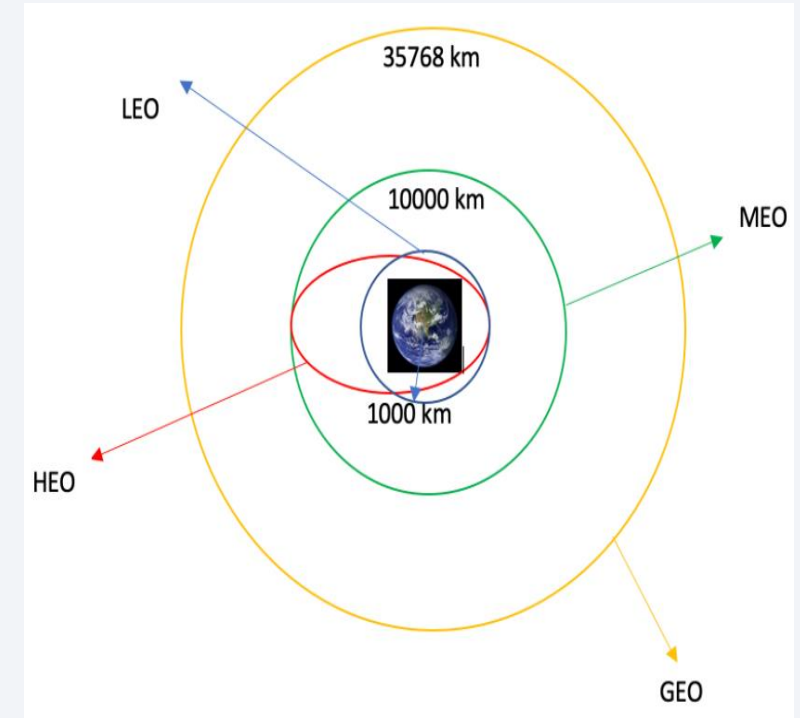
In [8]: # Use the find_all function in the BeautifulSoup object, with element type 'table'
# Assign the result to a list called 'html_tables'
html_tables = soup.findAll("table")

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

In [9]: # let's print the third table and check its content
```

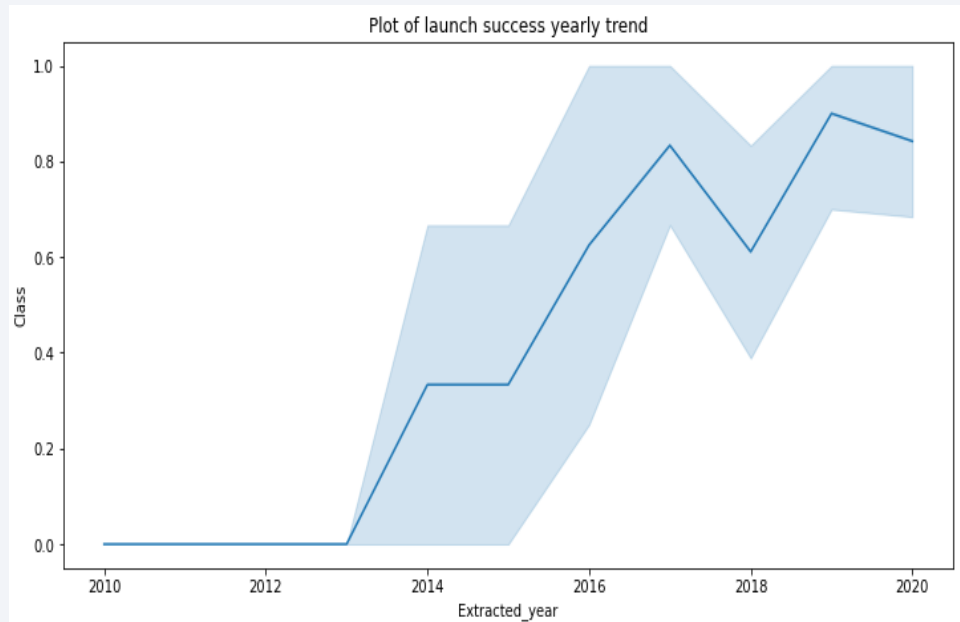
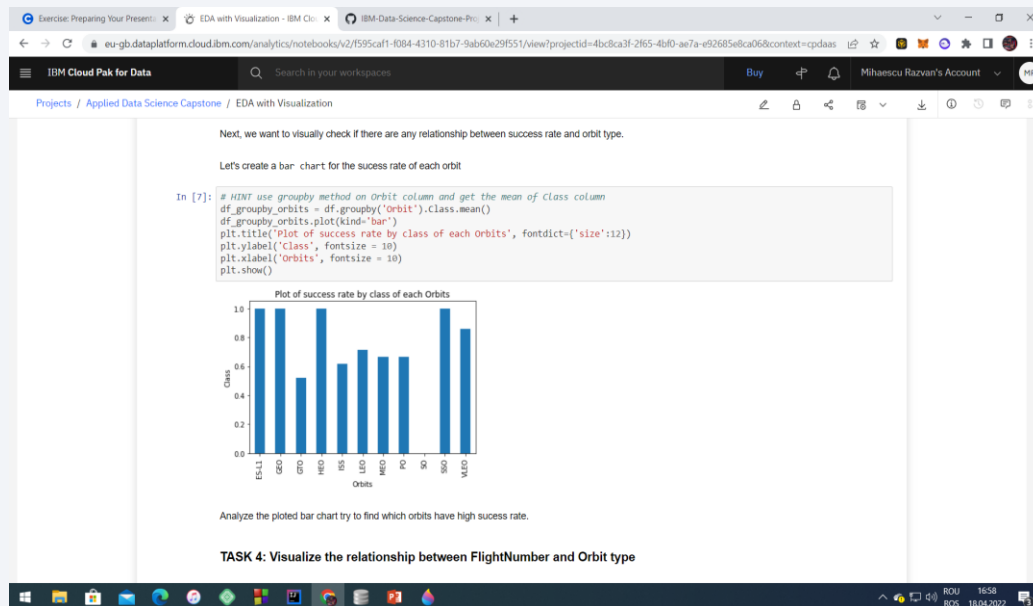
Data Wrangling

- 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 <https://github.com/mihaescurazvan/IBM-Data-Science-Capstone-Project-SpaceX/blob/master/Data%20Wrangling.ipynb>.



EDA with Data Visualization

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



The link to the notebook is
<https://github.com/mihaescurazvan/IBM-Data-Science-Capstone-Project-SpaceX/blob/master/EDA%20with%20Visualization.ipynb>

EDA with SQL

- We loaded the SpaceX dataset into a PostgreSQL database without leaving the jupyter notebook.
- We applied EDA with SQL to get insight from the data. We wrote queries to find out for instance:
 - The names of unique launch sites in the space mission.
 - The total payload mass carried by boosters launched by NASA (CRS)
 - The average payload mass carried by booster version F9 v1.1
 - The total number of successful and failure mission outcomes
 - The failed landing outcomes in drone ship, their booster version and launch site names.
- The link to the notebook is <https://github.com/mihaescurazvan/IBM-Data-Science-Capstone-Project-SpaceX/blob/master/EDA%20with%20SQL.ipynb>

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.

Predictive Analysis (Classification)

- We loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- We built different machine learning models and tune different hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model.
- The link to the notebook is <https://github.com/mihaescurazvan/IBM-Data-Science-Capstone-Project-SpaceX/blob/master/Machine%20Learning%20Prediction.ipynb>

Results

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

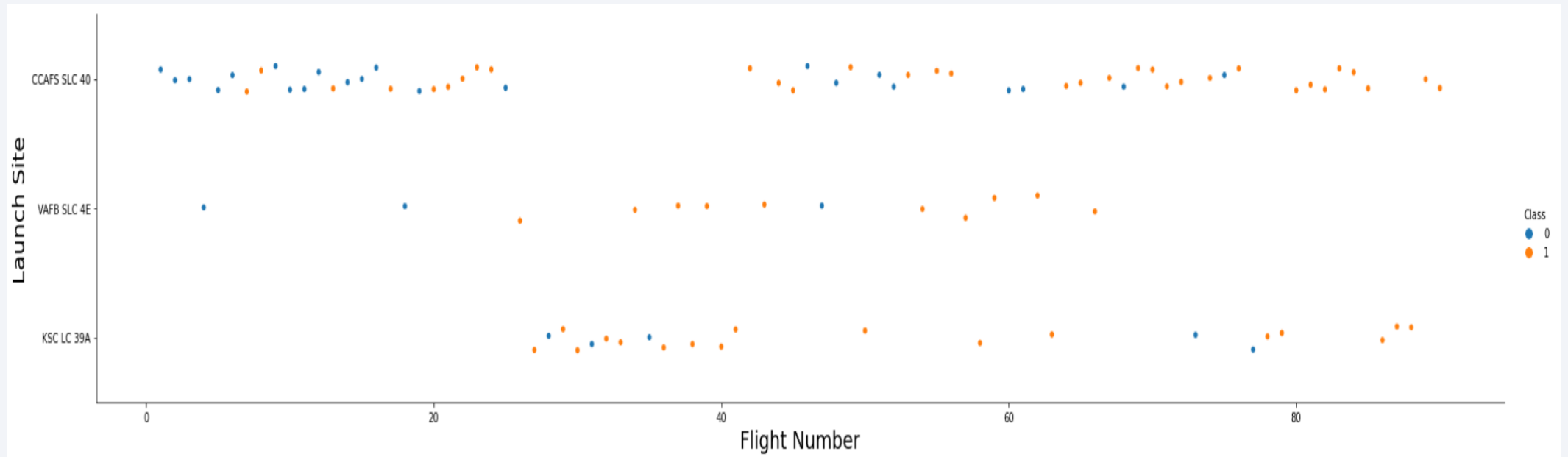
The background of the slide is an abstract composition. It features a dark blue base color. Overlaid on this are numerous diagonal streaks in shades of red and cyan. A faint, light blue grid pattern is also visible, particularly in the lower-left quadrant. The overall effect is dynamic and technological.

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

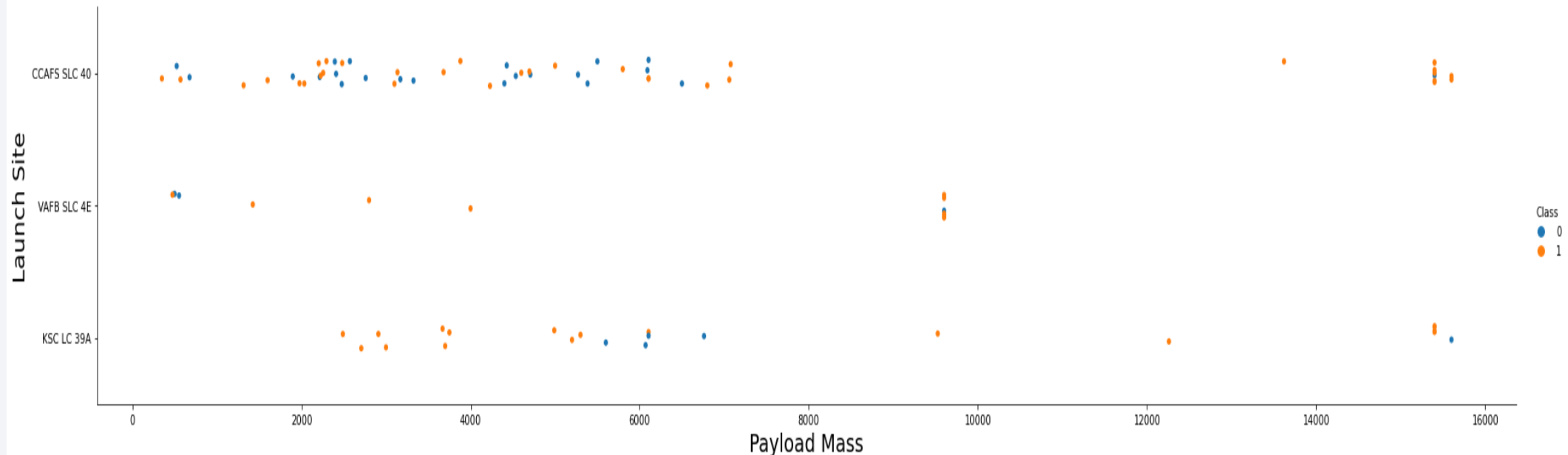
- From the plot, we found that the larger the flight amount at a launch site, the greater the success rate at a launch site.



Payload vs. Launch Site

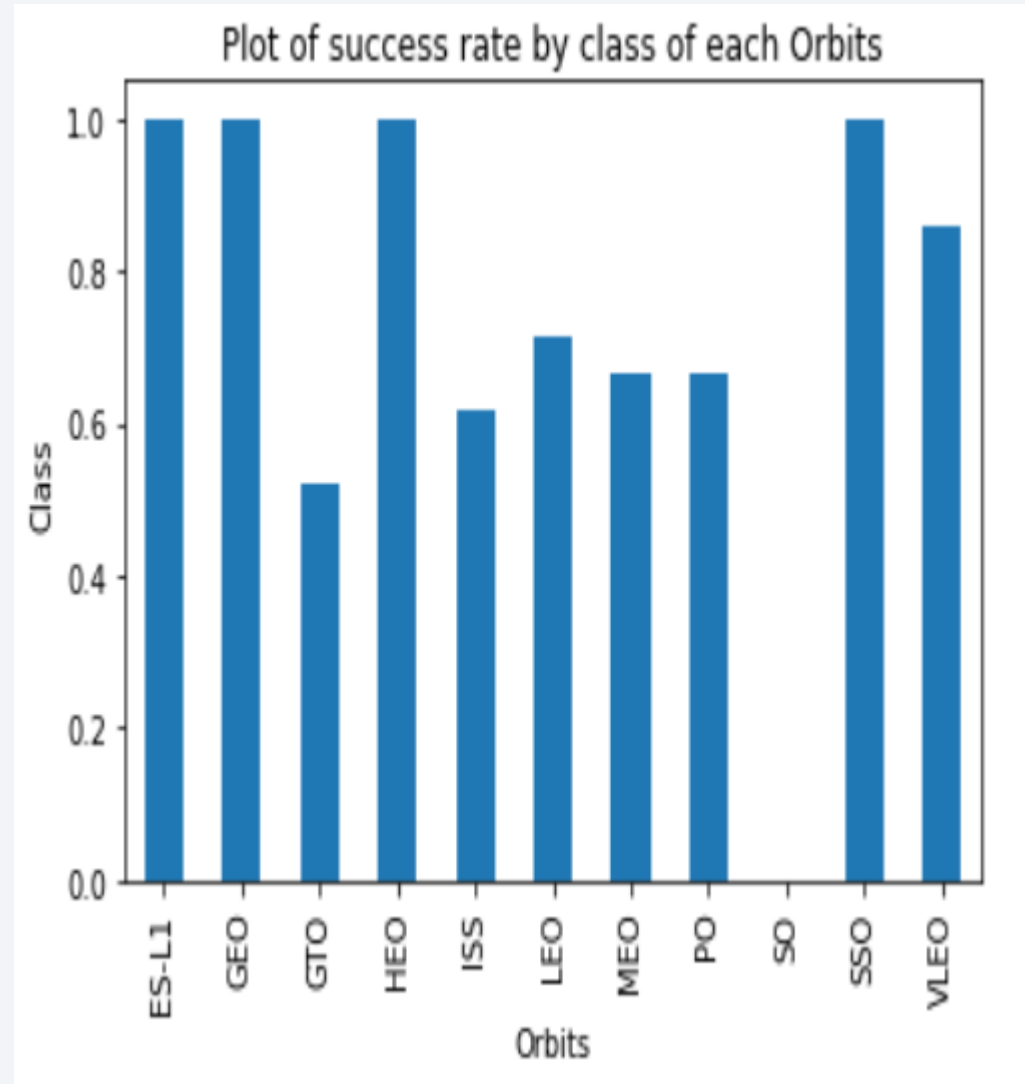


The greater the payload mass for launch site CCAFS SLC 40 the higher the success rate for the rocket.



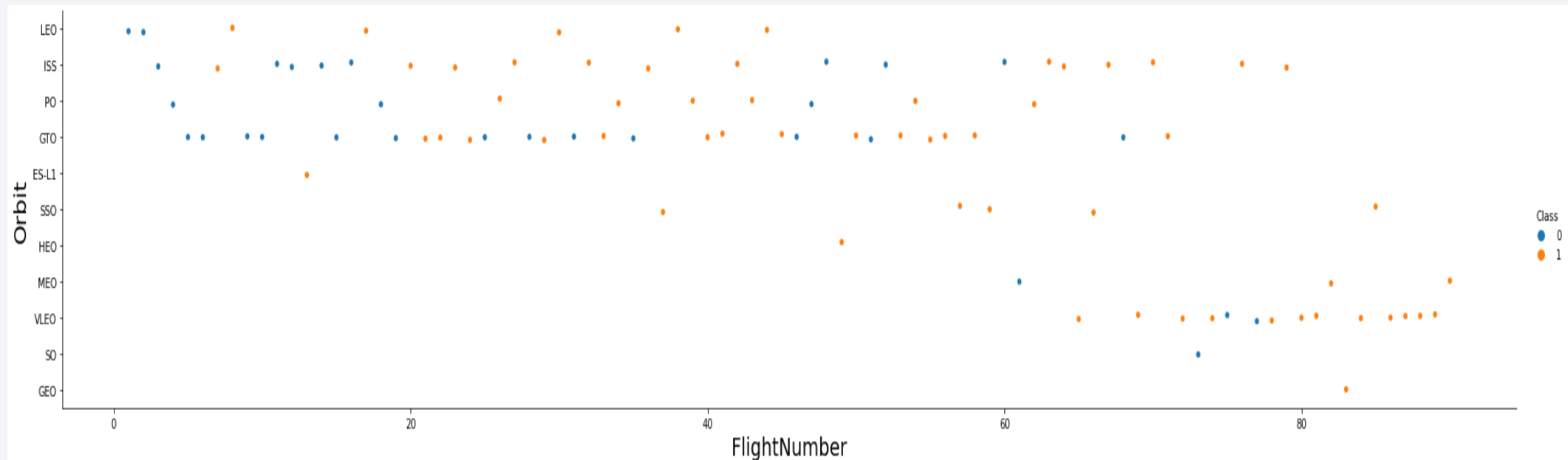
Success Rate vs. Orbit Type

- From the plot, we can see that ES-L1, GEO, HEO, SSO, VLEO had the most success rate.



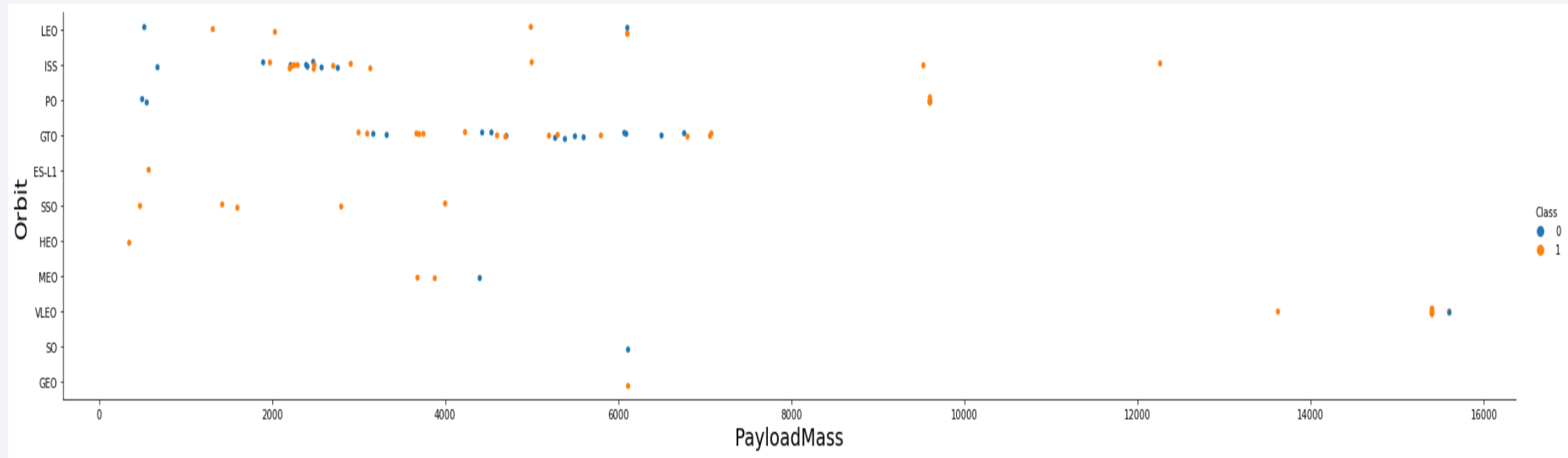
Flight Number vs. Orbit Type

- The plot below shows the Flight Number vs. Orbit type. We observe that in the LEO orbit, success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit.



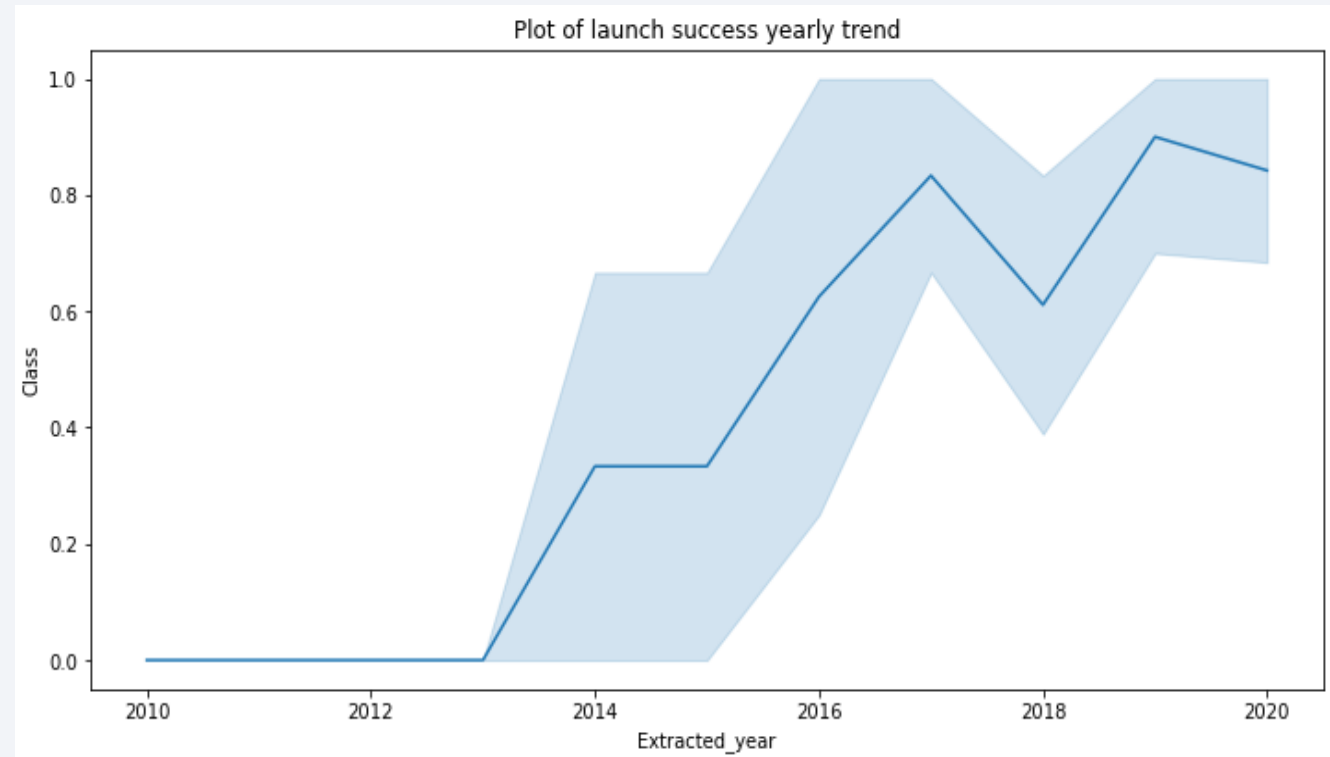
Payload vs. Orbit Type

- We can observe that with heavy payloads, the successful landing are more for PO, LEO and ISS orbits.



Launch Success Yearly Trend

- From the plot, we can observe that success rate since 2013 kept on increasing till 2020.



All Launch Site Names

- We used the key word **DISTINCT** to show only unique launch sites from the SpaceX data.

Task 1

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

In [5]:

%%sql

```
select DISTINCT LAUNCH_SITE FROM SPACEXTBL;
```

* ibm_db_sa://prb06299:***@824dfd4d-99de-440d-9991-629c01b3832d.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30119/bludb
Done.

Out[5]:

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

Launch Site Names Begin with 'CCA'

Task 2

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

```
In [7]: %%sql
select * from SPACEXTBL where launch_site like 'CCA%' limit 5;

* ibm_db_sa://prb06299:***@824dfd4d-99de-440d-9991-629c01b3832d.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30119/blddb
Done.
```

Out[7]:

| 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 |

- We used the query above to display 5 records where launch sites begin with `CCA`

Total Payload Mass

- We calculated the total payload carried by boosters from NASA as 45596 using the query below

Task 3

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

```
In [18]: %%sql
select SUM(payload_mass__kg_) AS Total_PayloadMass from SPACEXTBL where Customer LIKE 'NASA (CRS)';

* ibm_db_sa://prb06299:***@824dfd4d-99de-440d-9991-629c01b3832d.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30119/bludb
Done.
```

```
Out[18]:
```

| total_payloadmass |
|-------------------|
| 45596 |

Average Payload Mass by F9 v1.1

- We calculated the average payload mass carried by booster version F9 v1.1 as 2928.

Task 4

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

In [20]:

```
%%sql
select avg(payload_mass__kg_) AS Average_PayloadMass from SPACEXTBL where booster_version = 'F9 v1.1';

* ibm_db_sa://prb06299:***@824dfd4d-99de-440d-9991-629c01b3832d.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30119/bludb
Done.
```

Out[20]:

| average_payloadmass |
|---------------------|
| 2928 |

First Successful Ground Landing Date

- We observed that the dates of the first successful landing outcome on ground pad was 22nd December 2015

Task 5

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

Hint: Use min function

```
In [55]: %sql select min(DATE) from SPACEXTBL where ("Landing _Outcome") LIKE 'Success (ground pad)';
```

```
* ibm_db_sa://prb06299:***@824dfd4d-99de-440d-9991-629c01b3832d.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30119/bludb
Done.
```

```
Out[55]:
```

| |
|------------|
| 1 |
| 2015-12-22 |

Successful Drone Ship Landing with Payload between 4000 and 6000

- We used the **WHERE** clause to filter for boosters which have successfully landed on drone ship and applied the **AND** condition to determine successful landing with 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

```
In [56]: %sql select booster_version from SPACEXTBL where ("Landing _Outcome") = 'Success (drone ship)' AND payload_mass__kg_ between 4000 and 6000;
```

```
* ibm_db_sa://prb06299:***@824dfd4d-99de-440d-9991-629c01b3832d.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30119/bludb
Done.
```

```
Out[56]:
```

| booster_version |
|-----------------|
| F9 FT B1022 |
| F9 FT B1026 |
| F9 FT B1021.2 |
| F9 FT B1031.2 |

Total Number of Successful and Failure Mission Outcomes

- We used wildcard like '%' to filter for **WHERE** MissionOutcome was a success or a failure.

Task 7

List the total number of successful and failure mission outcomes

```
In [58]: %sql select count(mission_outcome) as successful_mission_outcomes from SPACEXTBL where ("Landing _Outcome") LIKE 'Succes%'
* ibm_db_sa://prb06299:***@824dfd4d-99de-440d-9991-629c01b3832d.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30119/bludb
Done.
```

```
Out[58]:
```

| successful_mission_outcomes |
|-----------------------------|
| 61 |

```
In [59]: %sql select count(mission_outcome) as failure_mission_outcomes from SPACEXTBL where ("Landing _Outcome") LIKE 'Failure%'
* ibm_db_sa://prb06299:***@824dfd4d-99de-440d-9991-629c01b3832d.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30119/bludb
Done.
```

```
Out[59]:
```

| failure_mission_outcomes |
|--------------------------|
| 10 |

Boosters Carried Maximum Payload

- We determined the booster that have carried the maximum payload using a subquery in the **WHERE** clause and the **MAX()** function.

Task 8

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

```
In [60]: %%sql
select booster_version from SPACEXTBL where payload_mass__kg_ = (select max(payload_mass__kg_) from SPACEXTBL)

* ibm_db_sa://prb06299:***@824dfd4d-99de-440d-9991-629c01b3832d.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30119/bludb
Done.
```

```
Out[60]:
```

| booster_version |
|-----------------|
| F9 B5 B1048.4 |
| F9 B5 B1049.4 |
| F9 B5 B1051.3 |
| F9 B5 B1056.4 |
| F9 B5 B1048.5 |
| F9 B5 B1051.4 |
| F9 B5 B1049.5 |
| F9 B5 B1060.2 |
| F9 B5 B1058.3 |
| F9 B5 B1051.6 |
| F9 B5 B1060.3 |
| F9 B5 B1049.7 |

2015 Launch Records

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

Task 9

List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

```
In [61]: %%sql
SELECT booster_version,launch_site, ("Landing _Outcome") FROM SPACEXTBL WHERE ("Landing _Outcome") LIKE 'Failure (drone ship)' A
ND DATE BETWEEN '2015-01-01' AND '2015-12-31'

* ibm_db_sa://prb06299:***@824dfd4d-99de-440d-9991-629c01b3832d.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30119/bludb
Done.
```

```
Out[61]:
```

| booster_version | launch_site | Landing _Outcome |
|-----------------|-------------|----------------------|
| F9 v1.1 B1012 | CCAFS LC-40 | Failure (drone ship) |
| F9 v1.1 B1015 | CCAFS LC-40 | Failure (drone ship) |

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

- We 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 2017-03-20.
- We 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 landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

```
In [62]: %%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://prb06299:***@824dfd4d-99de-440d-9991-629c01b3832d.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30119/bludb
Done.

Out[62]:

| Landing_Outcome | 2 |
|------------------------|----|
| 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 |

A satellite view of Earth from space, showing the curvature of the planet and city lights at night. The background is a deep blue gradient.

Section 3

Launch Sites Proximities Analysis

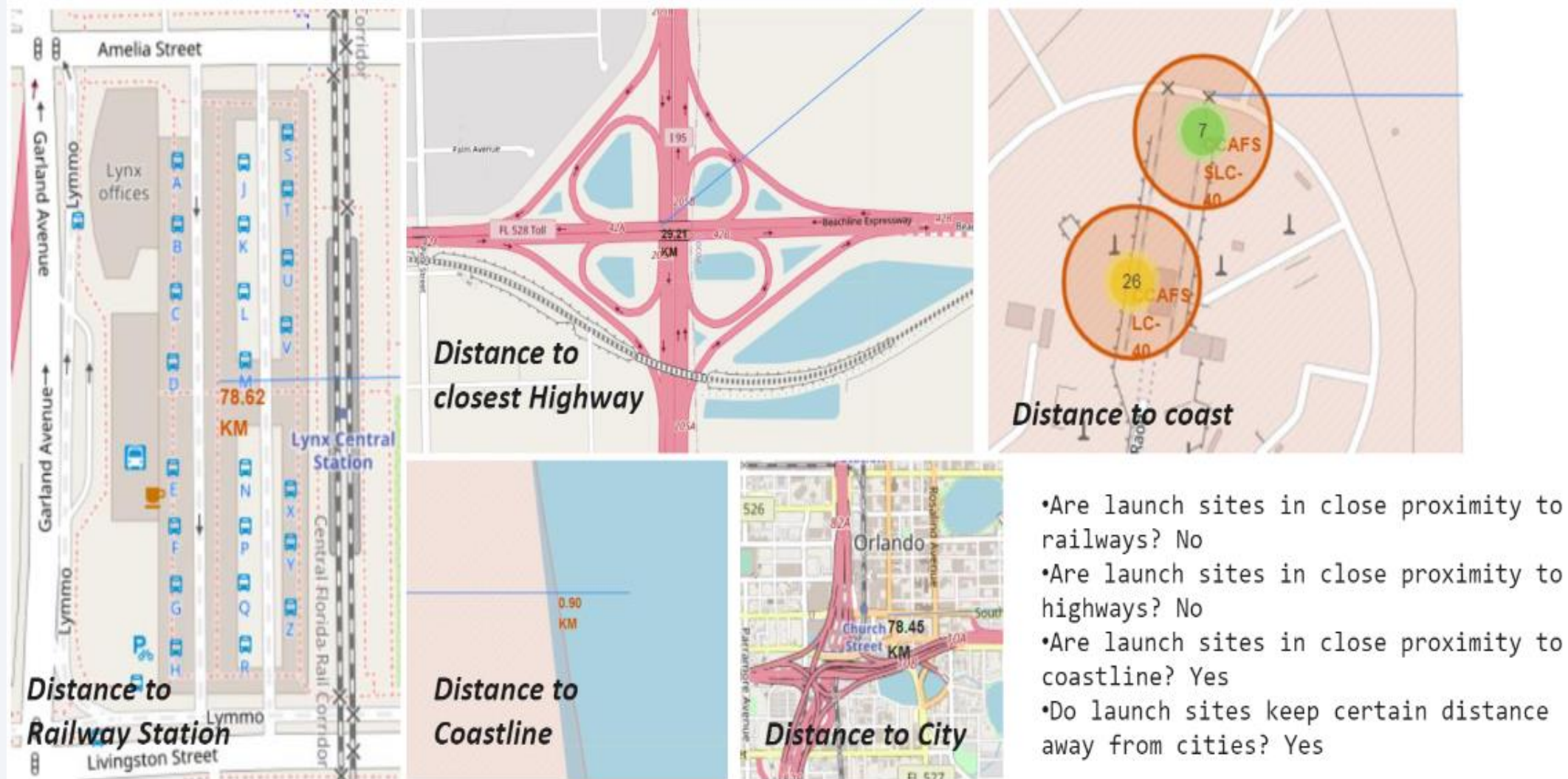
All launch sites global map markers



Markers showing launch sites with color labels



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

Predictive Analysis (Classification)

Classification Accuracy

- The decision tree classifier is the model with the highest classification accuracy

```
In [24]: 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()
```

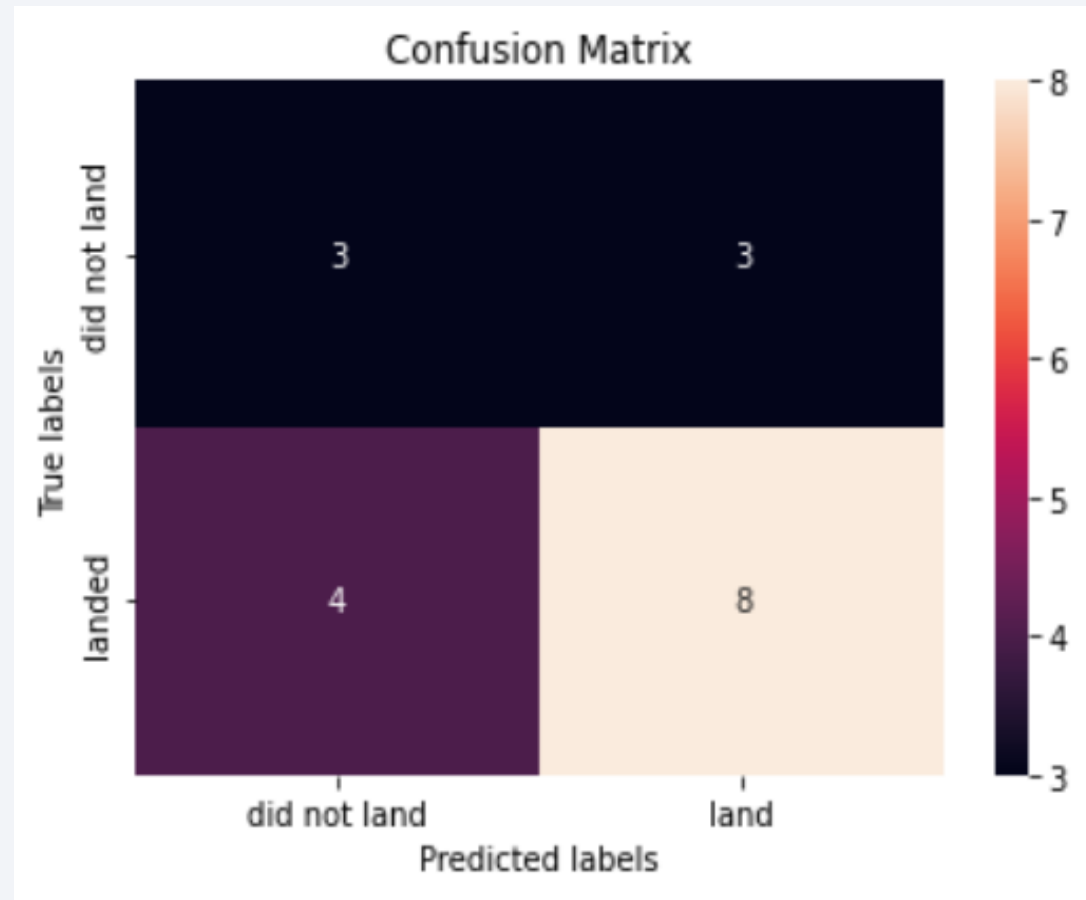
```
In [25]: grid_search_tree = GridSearchCV(
        estimator = tree,
        param_grid = parameters,
        scoring = 'accuracy',
        cv = 10
    )
tree_cv = grid_search_tree.fit(X_train,Y_train)
```

```
In [26]: print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
        print("accuracy :",tree_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters) {'criterion': 'entropy', 'max_depth': 10, 'max_features': 'sqrt', 'min samples leaf':
2, 'min_samples_split': 2, 'splitter': 'best'}
accuracy : 0.8875
```


Confusion Matrix

- The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The major problem is the false positives .i.e., unsuccessful landing marked as successful landing by the classifier.



Conclusions

We can conclude that:

- 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 this task.

Thank you!

