IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Pradeep Aryal
12/15/2024

Outline

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

Executive Summary

Extracted data from the public SpaceX API and the SpaceX Wikipedia page.
Analyzed and visualized data using SQL queries, graphical visualizations, Folium maps, and dashboards. Selected relevant columns to serve as features for modeling.

Utilized GridSearchCV to optimize hyperparameters for model performance.
Developed and evaluated four different machine learning models:

All models achieved similar accuracy rates of approximately 83.33%, but consistently overpredicted successful landings.
The limited dataset impacted the ability to accurately distinguish between successful and unsuccessful landings. Additional data is needed to improve model reliability and prediction accuracy.

Introduction

- SpaceX launches Falcon 9 rockets at an estimated cost of $62 million, significantly lower than the industry average of $165 million or more. This cost efficiency is largely attributed to SpaceX's ability to recover and reuse the first stage of the rocket, a groundbreaking innovation in aerospace technology.

- By predicting whether the first stage of a Falcon 9 rocket will successfully land, we can estimate the overall launch cost. This information can be leveraged to evaluate whether an alternate company should compete with SpaceX for a rocket launch contract.

Section 1

Methodology

Methodology

Executive Summary

- **Data collection methodology:**
  - Get request to SpaceX API and web scraping SpaceX Wikipedia
- **Perform data wrangling**
  - Handling missing values
  - Creating a landing outcome label that shows 0 when the booster did not land successful and1 when the booster did land successfully
- **Perform exploratory data analysis (EDA) using visualization and SQL**
- **Perform interactive visual analytics using Folium and Plotly Dash**
- **Perform predictive analysis using classification models**
  - Along with simpler machine learning algorithm from scikit-learn, GridSearchCV was to build, tune, evaluate classification models

Data Collection

- Make a GET response to SpaceX API
- Normalized the response and create a data frame
- Create new data frame from constructed dictionary that followed series of cleaning steps.
- Remove data consisting of Falcon 1 launches.
- Handle missing values using mean imputation.

Data Collection – SpaceX API

GitHub url:

https://github.com/waributwal/IBM-Data-Science-Professional-Certificate/blob/main/Applied%20Data%20Science%20Capstone/Module%201/jupyter-labs-spacex-data-collection-api-v2.ipynb

1
```
spacex_url = https://api.spacexdata.com/v4/launches/past
response = requests.get(spacex_url)
```

2
```
data = pd.json_normalize(response.json())
```

3
```
data_launch = pd.DataFrame.from_dict(launch_dict)
```

4
```
data_falcon9 = data_launch[data_launch['BoosterVersion'] != 'Falcon 1']
```

5
```
payload_mass_mean = data_falcon9['PayloadMass'].mean()
valuedata_falcon9['PayloadMass'].replace(np.nan, payload_mass_mean, inplace = True)
```

Data Collection - Scraping

- Request the HTML page from the static URL and assign to an object
- Create a Beautiful Soup object from the HTML response object
- Find all tables within the HTML page
- Collect all column header names from the tables
- Fill the dictionary with several functions

GitHub url:

https://github.com/waributwal/IBM-Data-Science-Professional-Certificate/blob/main/Applied%20Data%20Science%20Capstone/Module%201/jupyter-labs-webscraping.ipynb

```
url=https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922

html_data = requests.get(url)
```

```
soup = BeautifulSoup(html_data.text)
```

```
html_tables = soup.find_all('table')
```

```
launch_dict= dict.fromkeys(column_names)
```

```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
```

Data Wrangling

- Using the .value_counts() method to determine the following:
  - Number of launches on each site
  - Number and occurrence of each orbit
  - Number and occurrence of landing outcome per orbit type

- To determine whether a booster will successfully land, the value of 1 or 0 was used. This was achieved using
  - Defining a set of unsuccessful (bad) outcomes, bad_outcome
  - Creating a list, landing_class, where the element is 0 if the corresponding row in Outcome is in the set bad_outcome, otherwise, it's 1.

GitHub Url:

https://github.com/waributwal/IBM-Data-Science-Professional-Certificate/blob/main/Applied%20Data%20Science%20Capstone/Module%201/labs-jupyter-spacex-Data%20wrangling-v2.ipynb

EDA with Data Visualization

Line charts were produced to depict relationships between:

Success Rate and Year

Scatter charts were used to depict relationships between:

- Flight Number and Launch Site
- Payload and Launch Site
- Orbit Type and Flight Number
- Payload and Orbit Type

A bar chart was produced to depict the relationship between:

Success Rate and Orbit Type

GitHub Url:
https://github.com/waributwal/IBM-Data-Science-Professional-Certificate/blob/main/Applied%20Data%20Science%20Capstone/Module%202/jupyter-labs-eda-dataviz-v2.ipynb

EDA with SQL

The SQL queries performed were:

- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'CCA'
- Display the total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first successful landing outcome in ground pad was achieved.
- 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 total number of successful and failure mission outcomes
- List the names of the booster_versions which have carried the maximum payload mass.
- List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.
- 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.

GitHub Url:

https://github.com/waributwal/IBM-Data-Science-Professional-Certificate/blob/main/Applied%20Data%20Science%20Capstone/Module%202/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- Mark all launch sites on a map

This was achieved by using Folium Map object and using folium.Circle and folium.Marker for each launch site

- Mark the success/failed launches for each site on a map

This was done by putting launches into clusters adding folium.Marker to the MarkerCluster() object.

- Calculate the distances between a launch site to its proximities

Calculations of distances between points can be made using the Lat and Long values and using the Lat and Long values, create a folium.Marker object to show the distance.

Build a Dashboard with Plotly Dash

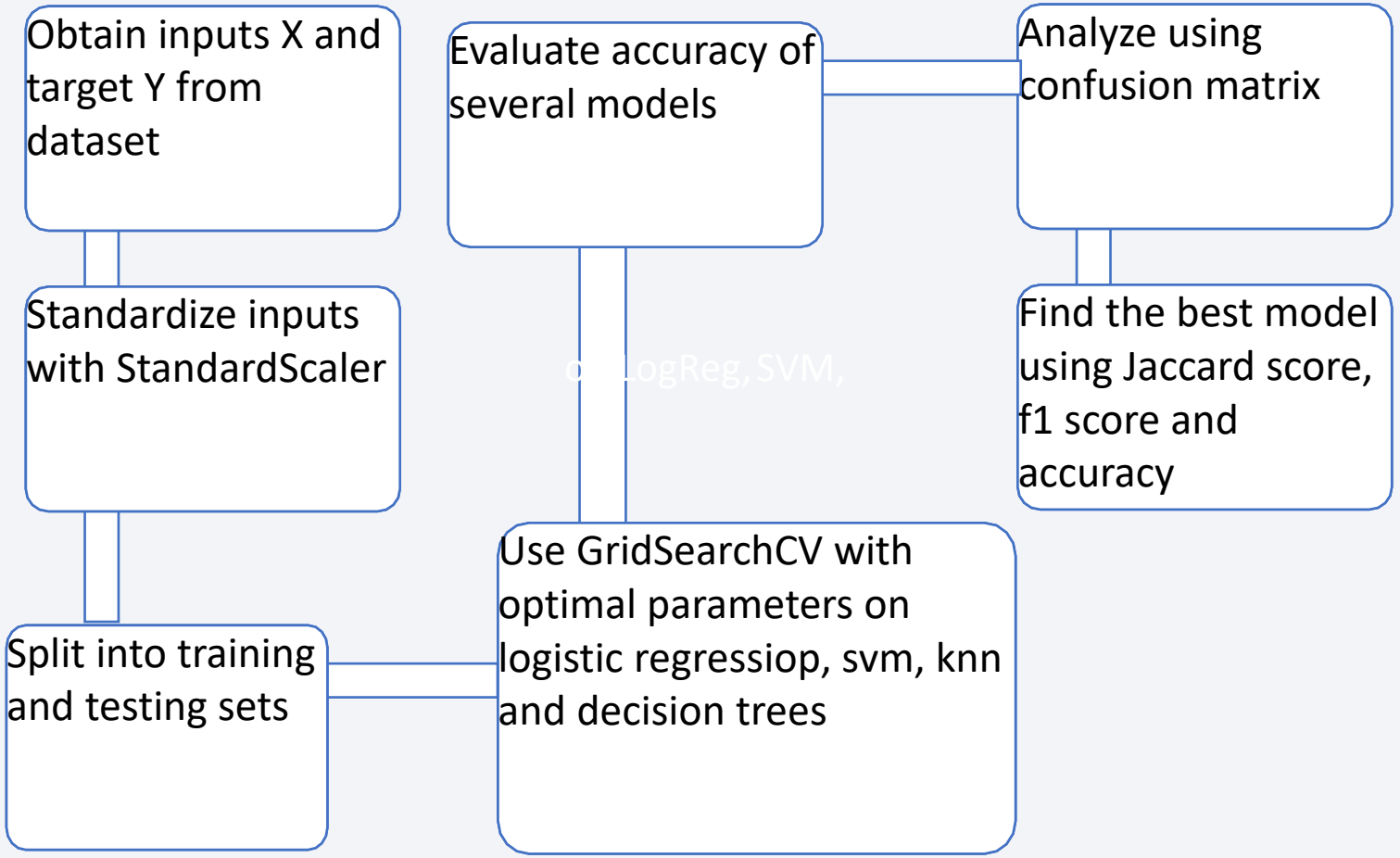
The dashboard includes a pie chart and a scatter plot.

Pie Chart(**px.pie()**): Displays the distribution of successful landings across all launch sites. Users can also select individual launch sites to view their specific success rates.

Scatter Plot(**px.scatter()**): Illustrates how success varies based on launch sites, payload mass, and booster version category.

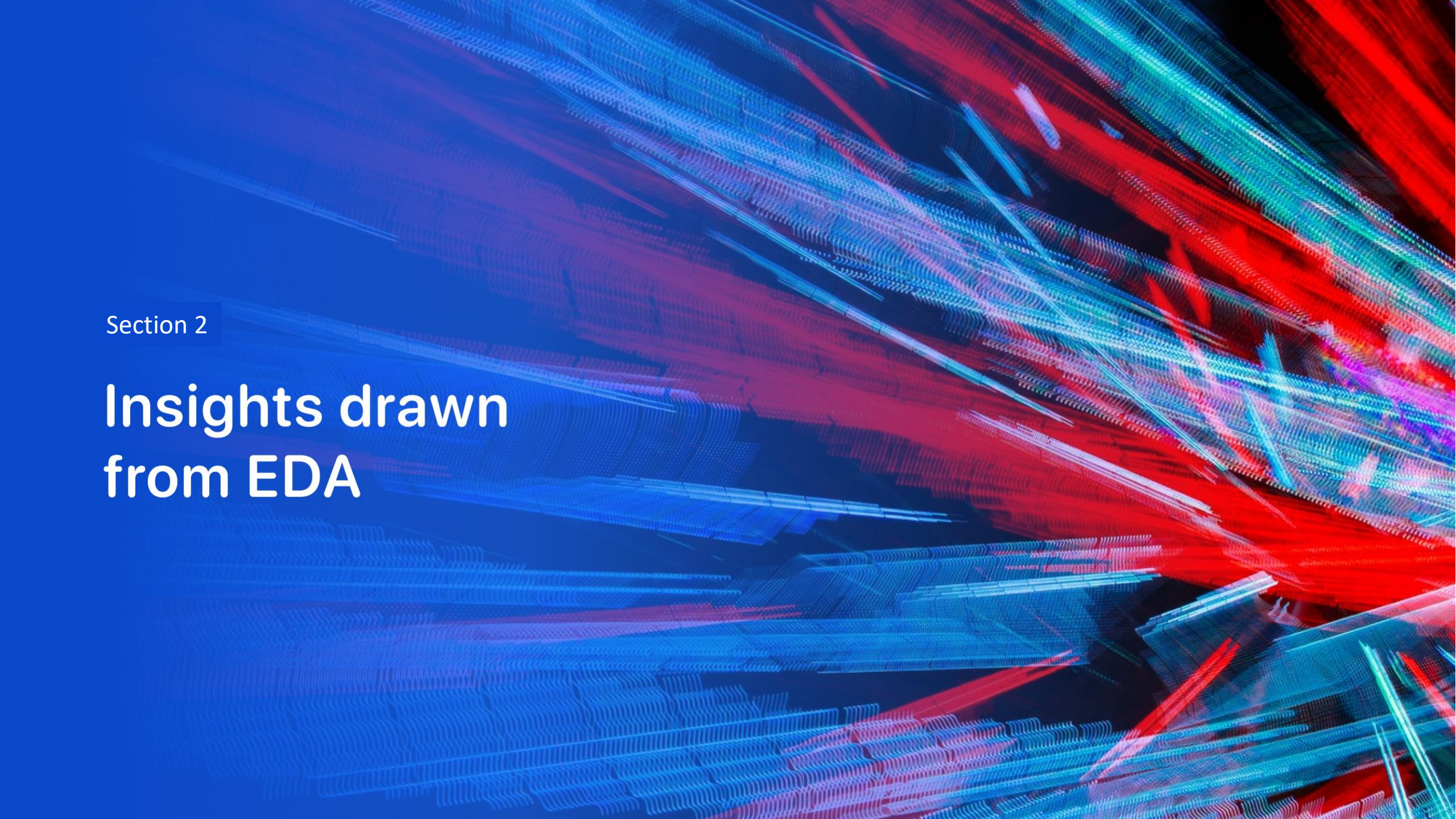
Predictive Analysis (Classification)

https://github.com/waributwal/IBM-Data-Science-Professional-Certificate/blob/main/Applied%20Data%20Science%20Capstone/Module%204/SpaceX-Machine-Learning-Prediction-Part-5-v1.ipynb

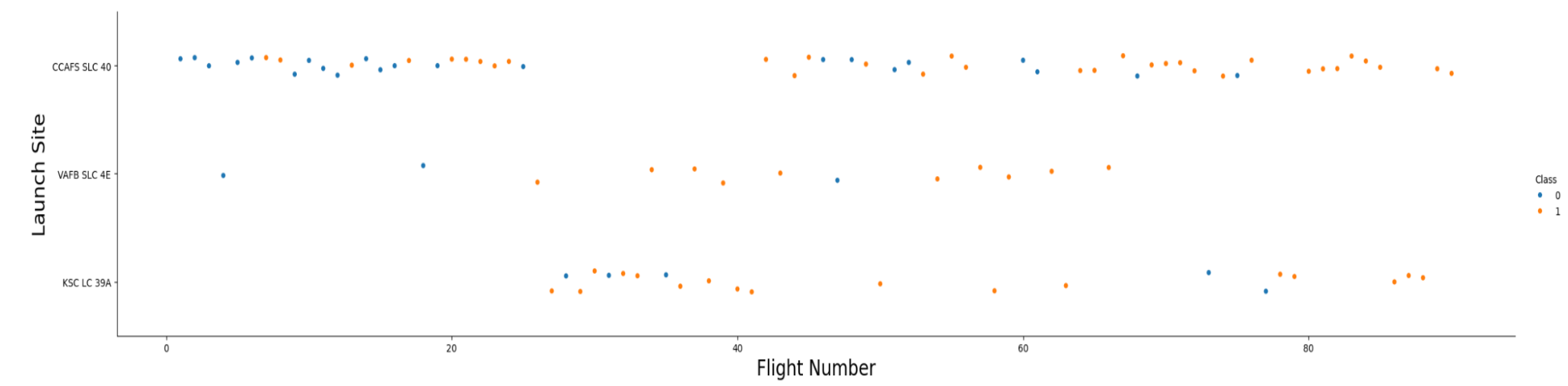


Results

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

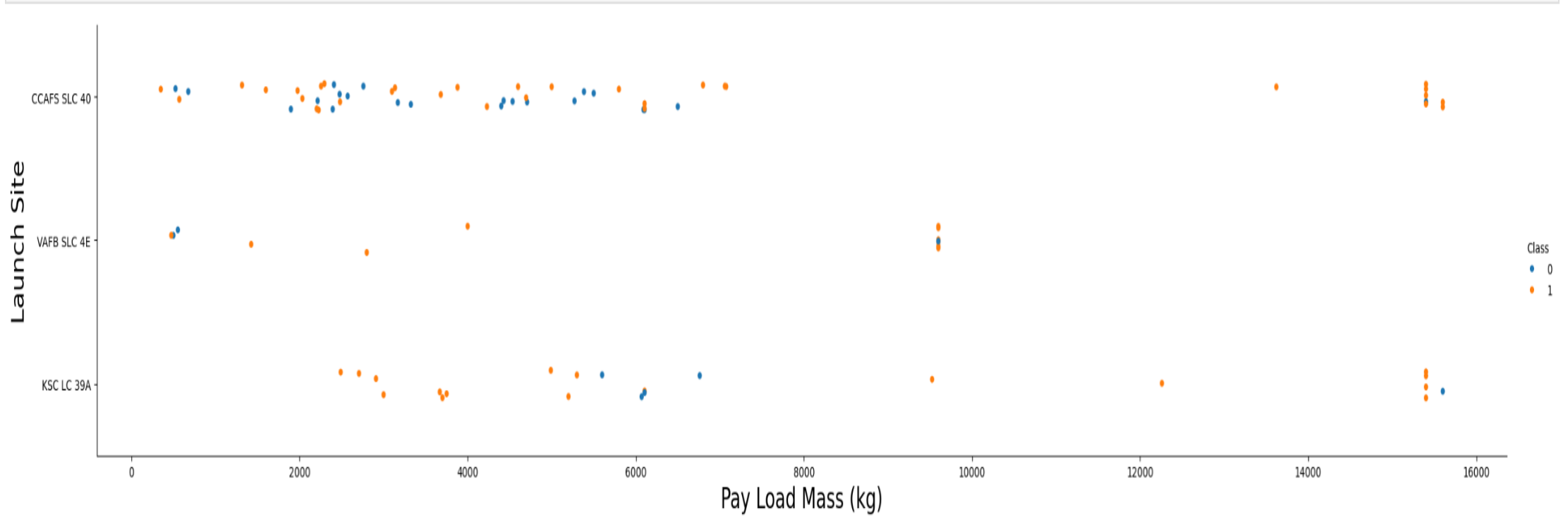

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



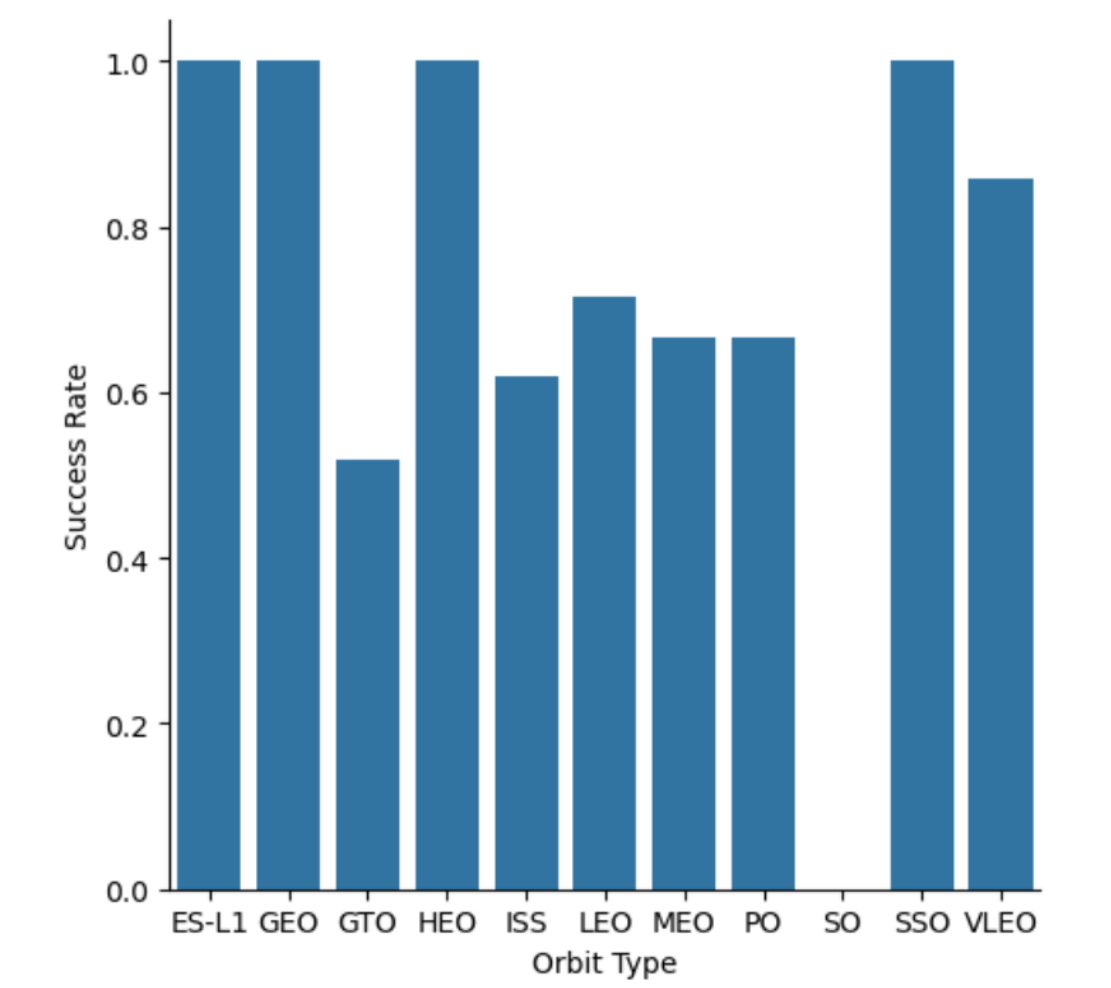
The insights drwan is as the number of flight increases the rate of success at launch site increases. From 40 most class falls under 1 which is success.

Payload vs. Launch Site



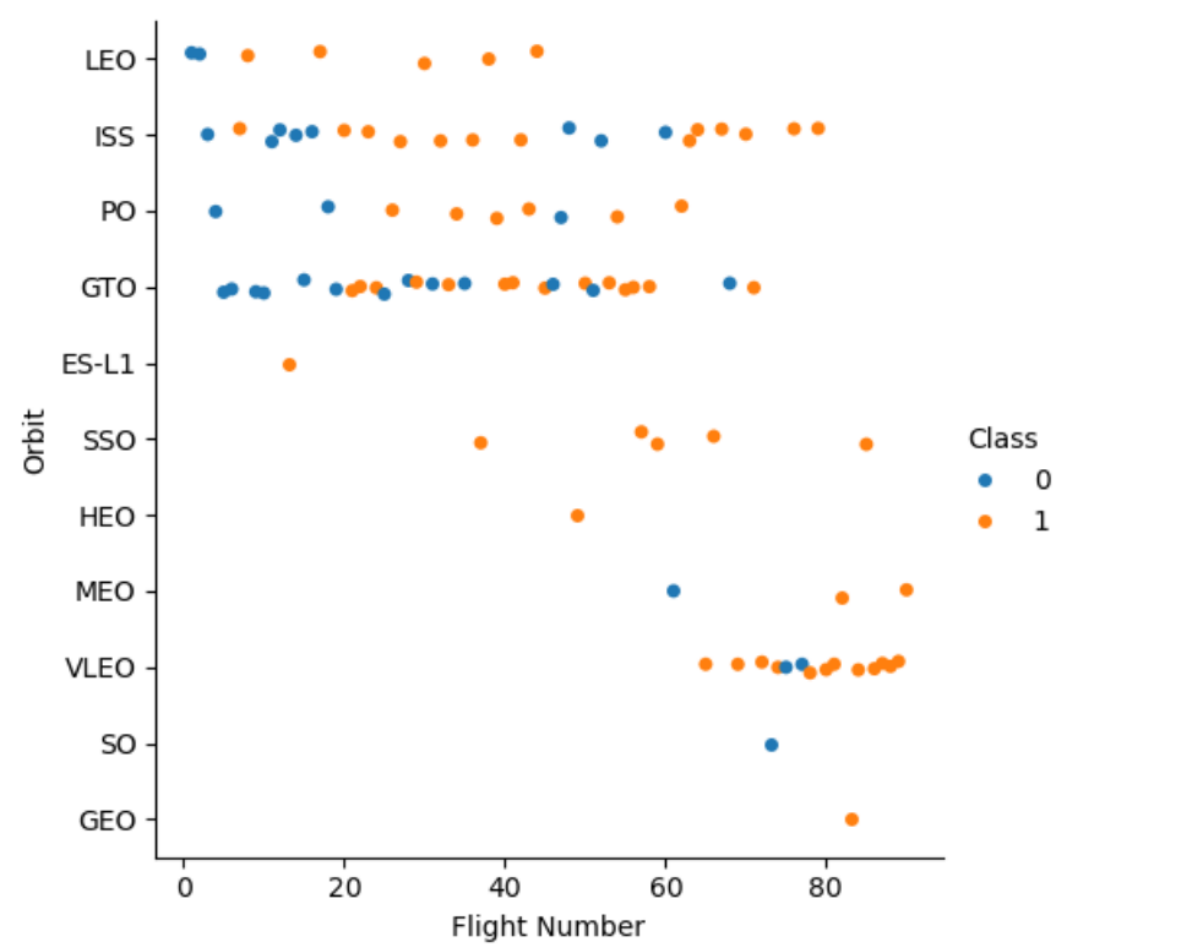
There is no clear relationship between pay load and success rate. Also, after around 8000 kg there are very less data but most are of success class.

Success Rate vs. Orbit Type



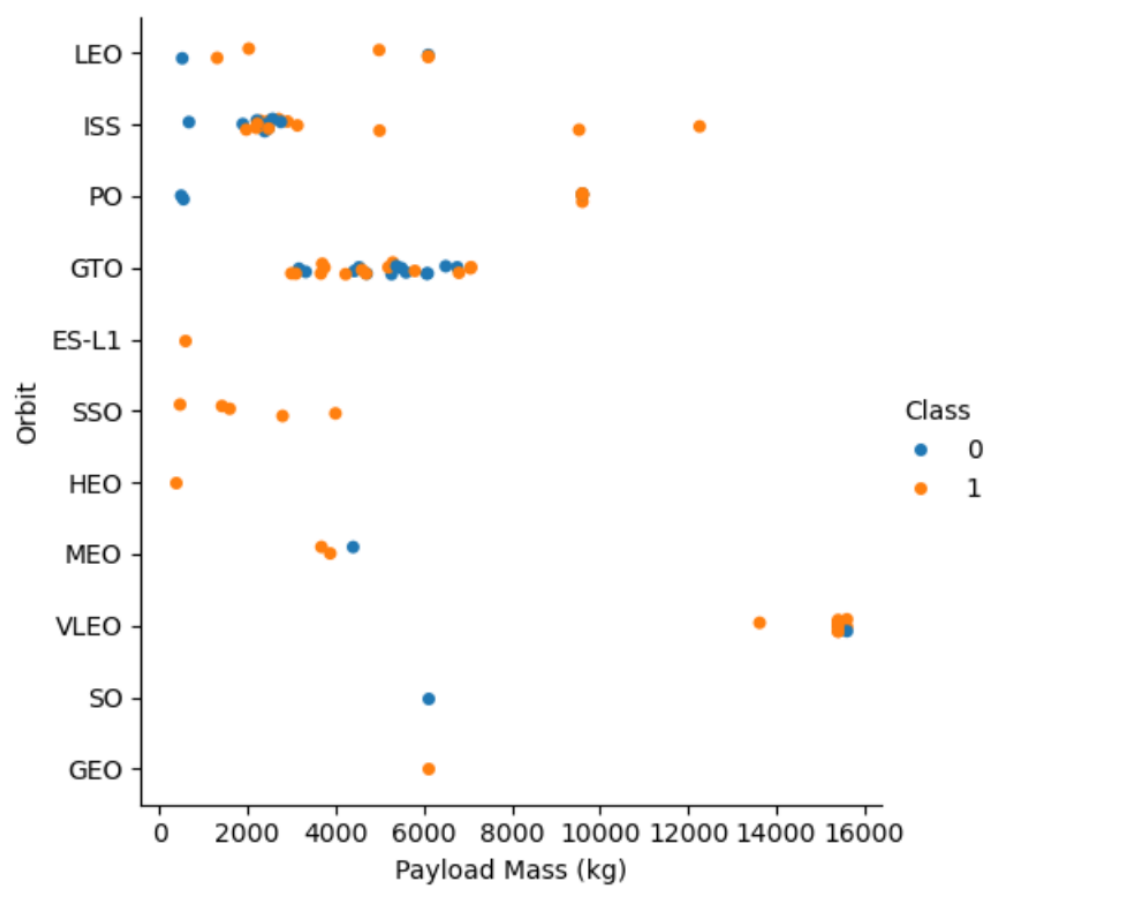
ES-L1, GEO, HEO, SSO have 100% success rate, while S0 have complete failure and other orbit have decent success rate.

Flight Number vs. Orbit Type



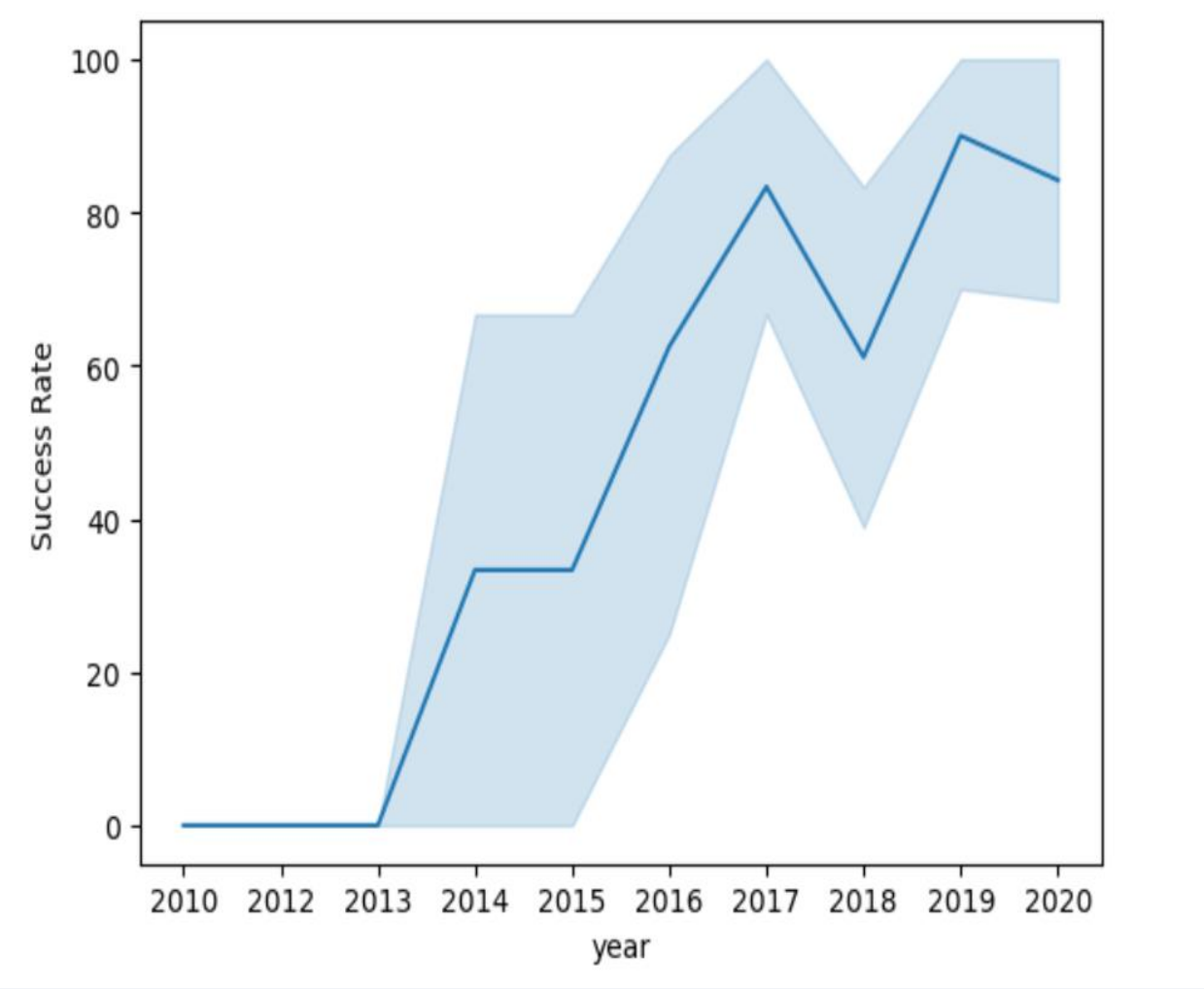
- The orbits having 100% success rate have fewer flights.
- There is no relationship between Flight Number and Orbits for GTO.
- General relation is if flight number increases success rate is increased too.

Payload vs. Orbit Type



- There is not clear relationship between payload and success for GTO.
- ES-L1, SSO, HEO have 100% success rate.

Launch Success Yearly Trend



- Strong increase in success rate from 2013 to 2018 with no change from 2014 to 2015.
- Success rate is 0 from 2012 to 2013
- From 2018 to 2019 there is increase in success rate.

All Launch Site Names

```
%sql SELECT DISTINCT(Launch_Site) FROM SPACEXTBL
```

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

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

DISTINCT(Launch_Site) function allows to generate unique Launch Sites.

Launch Site Names Begin with 'CCA'

```
%sql SELECT * FROM SPACEXTBL where Launch_Site like 'CCA%' LIMIT 5
```

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

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	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	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

Like CCA% finds launch_site starting only with CCA and limit 5 shows only 5 launch_sites.

Total Payload Mass

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Customer = 'NASA (CRS)'
```

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

SUM(PAYLOAD_MASS__KG_)
45596

SUM calcutes the total payload mass and where clause with customer = Nasa(CRS) shows the total payload of NASA.

Average Payload Mass by F9 v1.1

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE BOOSTER_VERSION LIKE '%F9 v1.1%'
```

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

AVG(PAYLOAD_MASS__KG_)
2534.6666666666665

AVG calcutes the average payload mass and WHERE BOOSTER_VERSION LIKE '%F9 v1.1%’ displays average payload mass only for F9 v1.1

First Successful Ground Landing Date

```
%sql SELECT MIN(Date) FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (ground pad)'
```

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

MIN(Date)
2015-12-22

Min(Date) displays first date and where clause with landing_outcome gives first successful ground landing date.

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%%sql
  SELECT Booster_Version FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (drone ship)'
  AND PAYLOAD_MASS__KG_ BETWEEN 4000 and 6000
```

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

Booster_Version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

BETWEEN 4000 and 6000 limits the payload mass between this range and use of where clause gives successful drone ship landing with payload between 4000 and 6000.

Total Number of Successful and Failure Mission Outcomes

```
%%sql
    SELECT Mission_Outcome, COUNT(*) AS Total_Number
    FROM SPACEXTABLE GROUP BY Mission_Outcome
```

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

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

GROUP BY Mission_Outcome limits the column to Mission_Outcome and Count(*) displays total number of mission outcomes

Boosters Carried Maximum Payload

```
%%sql
    SELECT Booster_Version FROM SPACEXTABLE
    WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE)
```

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

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

Use of subquery inside bracket along with where clause for payload resulted in boosters carrying maximum payload

2015 Launch Records

```
%%sql
    SELECT substr(Date,6,2) as month, DATE,BOOSTER_VERSION, LAUNCH_SITE, Landing_Outcome
    FROM SPACEXTABLE
    WHERE Landing_Outcome = 'Failure (drone ship)' and substr(Date,0,5)='2015';
```

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

month	Date	Booster_Version	Launch_Site	Landing_Outcome
01	2015-01-10	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
04	2015-04-14	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

substr(Date, 6, 2) helps to get the months and substr(Date, 0, 5) resulted in year 2015. Also, where clause is used for finding landing with failures.

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

```
%%sql
    SELECT Landing_Outcome, COUNT(*) as count_outcomes
    FROM SPACEXTABLE
    WHERE DATE BETWEEN '2010-06-04' and '2017-03-20'
    GROUP BY Landing_Outcome ORDER BY count_outcomes DESC;
```

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

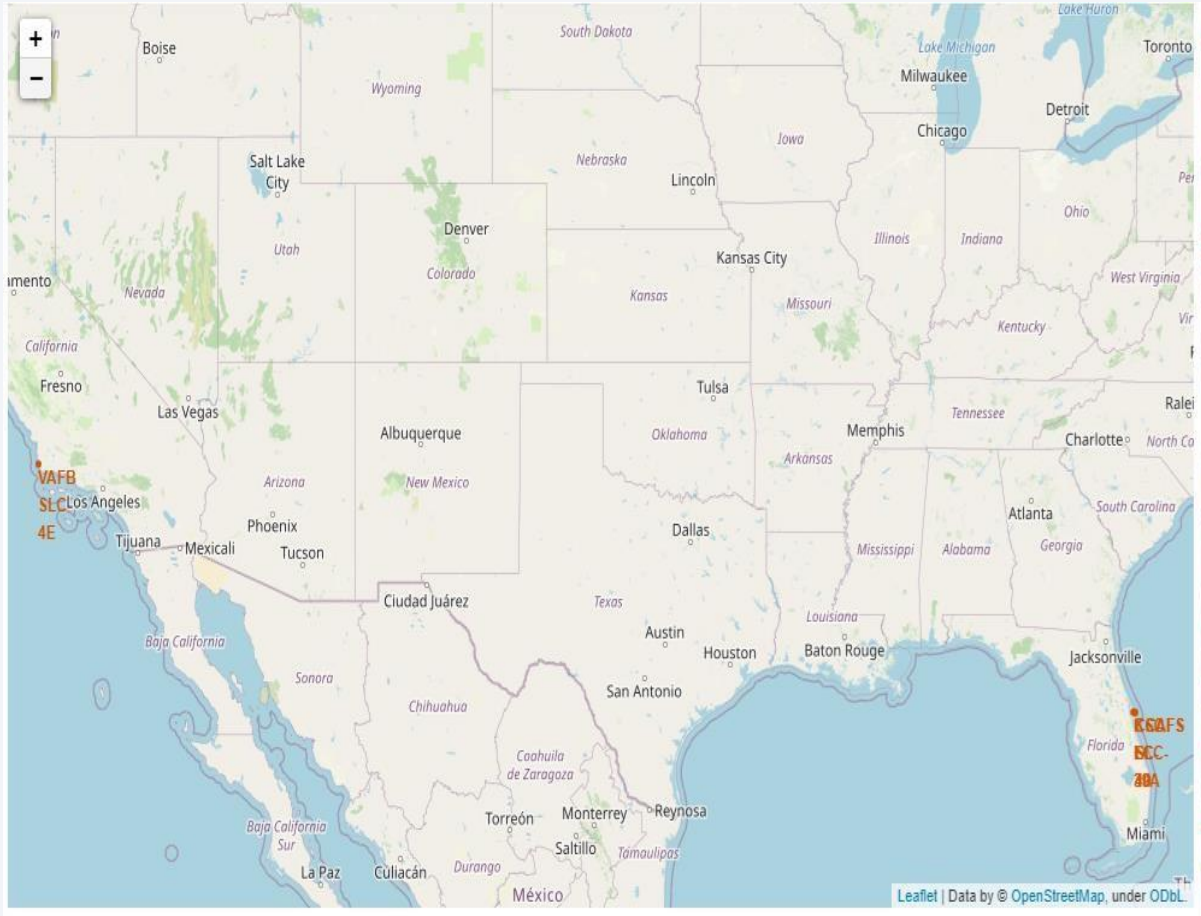
Landing_Outcome	count_outcomes
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

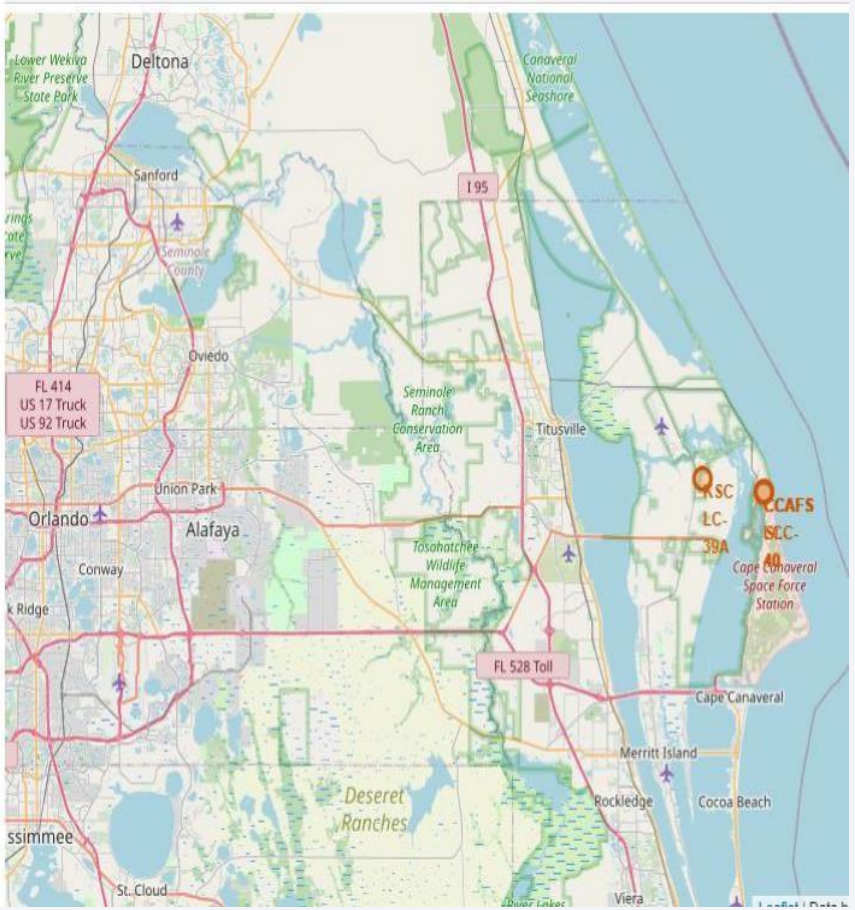
DESC is used for results in descending order. Group By is used for selecting column of our interest adn date between filters selected date range.

Section 3

Launch Sites Proximities Analysis

Displaying Launch Sites on Map





Section 5

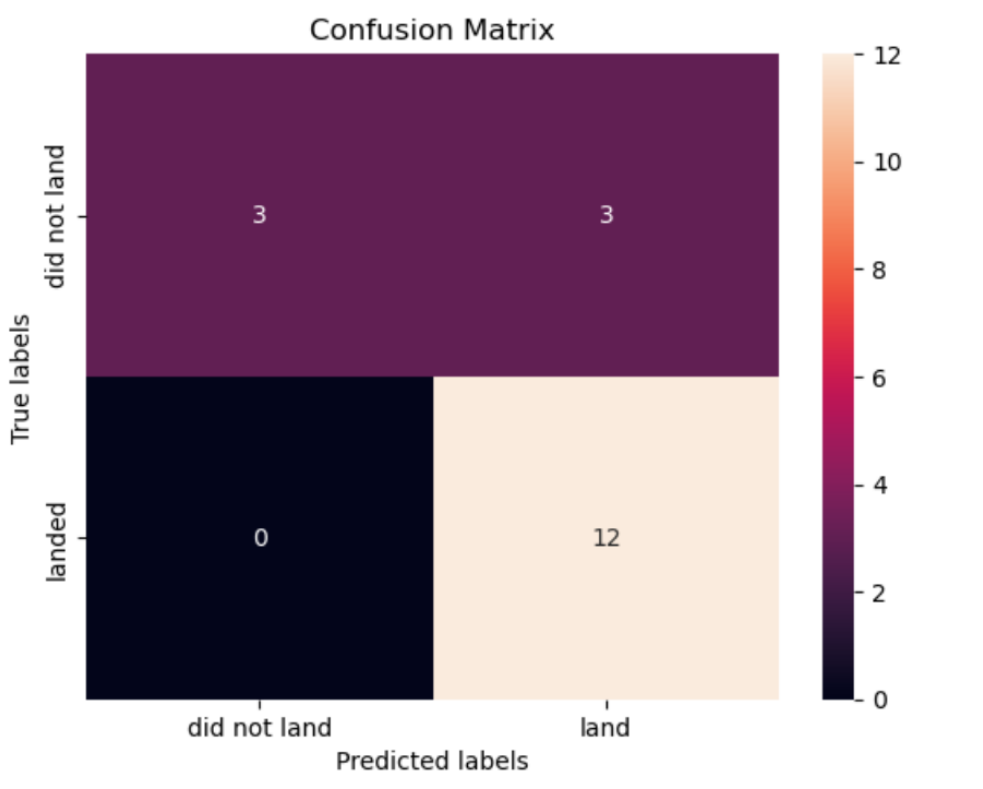
Predictive Analysis (Classification)

Classification Accuracy

	LogReg	SVM	Tree	KNN
Jaccard_Score	0.833333	0.845070	0.845070	0.819444
F1_Score	0.909091	0.916031	0.916031	0.900763
Accuracy	0.866667	0.877778	0.877778	0.855556

All model had similar accuracy on test set of 83.33%. But on whole date set Decision Tree performs with overall better result.

Confusion Matrix



All model had similar accuracy on test set of 83.33%. So this is the confusion matrix of tree classifier with 15 correct prediction and 3 incorrect for did not land.

Conclusions

- Extracted data from SpaceX's public API.
- Supplemented the dataset via web scraping the SpaceX Wikipedia page.
- Created labels for training, ensuring meaningful insights.
- Stored the processed data in a SQL database
- Designed a user-friendly dashboard.
- Built a machine learning model with an 83% accuracy on test data set.
- Need to collect additional data points to enhance model training and validation.

Appendix

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

```
# Takes the dataset and uses the rocket column to call the API and append the data to the list
def getBoosterVersion(data):
    for x in data['rocket']:
       if x:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])
```

```
# Takes the dataset and uses the launchpad column to call the API and append the data to the list
def getLaunchSite(data):
    for x in data['launchpad']:
       if x:
         response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
         Longitude.append(response['longitude'])
         Latitude.append(response['latitude'])
         LaunchSite.append(response['name'])
```

Code snippets for SpaceX data collection api

Thank you!