IBM Developer
SKILLS NETWORK

# Winning Space Race with Data Science

Daniel Carvalho Pedrini
05/01/2023

# Outline

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

# Executive Summary

- Summary of methodologies
  - Define and collect the data collection within different ways
  - Define, apply and compare Machine Learning models
  - Create Data Visualization to best view the data
- Summary of all results
  - Define the best model applied
  - Create the Visualizations to have best decision making

# Introduction

- Project background and context

For this project, we will work with the SpaceX company to predict the Falcon 9 to check the success of the first step landing. It is important for the matter of costs.

- Problems you want to find answers

1. What is the factors behind the failure of landing?
2. Will rockets land successfully?
3. What is the accuracy of successful landing?

Section 1

# Methodology

# Methodology

## Executive Summary

- Data collection methodology:
  - We used two methodology for data collection: API and Web Scrapping
- Perform data wrangling
  - We used One-hot-encoding
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - We applied different models to compere and find the best model

# Data Collection – SpaceX API

```
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

```
response = requests.get(static_json_url)
response.json()
```

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

```
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}
```

en, we need to create a Pandas data frame from the dictionary launch_di

```
# Create a data from launch_dict
launch_df = pd.DataFrame(launch_dict)
```

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = launch_df.query('BoosterVersion == "Falcon 9"')
```

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = launch_df.query('BoosterVersion == "Falcon 9"')
```

```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

- Call the API to get the JSON
- Convert JSON to DataFrame
- Update Columns and Rows
- Filter the data to keep Falcon 9
- Dealing with miss values of Payload Mass
- Save the data as .csv file

https://github.com/danielpedrini/capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

# Data Collection – SpaceX API

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

Next, request the HTML page from the above URL and get a response object

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.

```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url)
```

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.text,'html.parser')
```

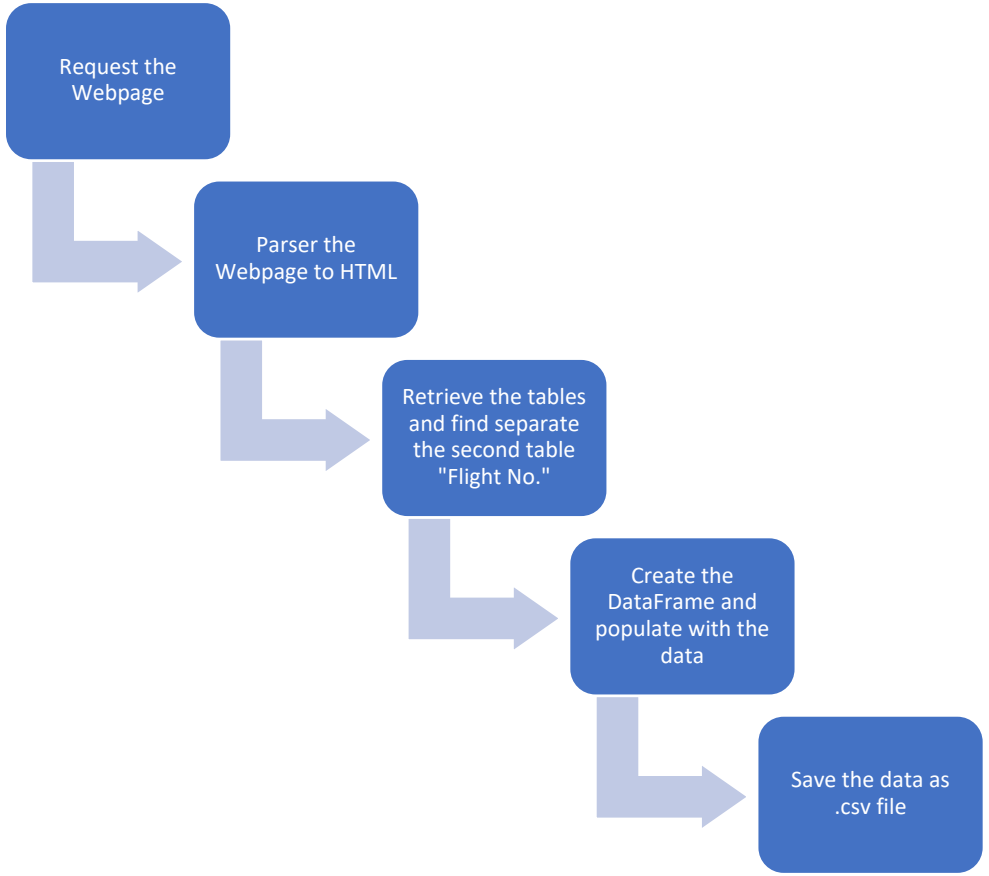
```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

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

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

```
extracted_row = 0
launch_dict = []
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
    for rows in table.find_all("tr"):
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        row=rows.find_all('td')
        if flag:
            extracted_row += 1
            datatimelist=date_time(row[0])
            date = datatimelist[0].strip(',')
            time = datatimelist[1]
            bv=booster_version(row[1])
            if not(bv):
                bv=row[1].a.string
            print(bv)
            launch_site = row[2].a.string
            payload = row[3].a.string
            payload_mass = get_mass(row[4])
            orbit = row[5].a.string
            if row[6].a:
                customer = row[6].a.string
            else:
                customer = None
            launch_outcome = list(row[7].strings)[0]
            booster_landing = landing_status(row[8])
            launch_outcome = {
                'Flight No.': flight_number,
                'Date': date,
                'Time': time,
                'Version Booster': bv,
                'Launch Site': launch_site,
                'Payload': payload,
                'Payload mass': payload_mass,
                'Orbit': orbit,
                'Customer': customer,
                'Launch outcome': launch_outcome,
                'Booster landing': booster_landing
            }
            launch_dict.append(launch_outcome)
```

```
df=pd.DataFrame(launch_dict)
```




https://github.com/danielpedrini/capstone/blob/main/jupyter-labs-webscraping.ipynb

# Data Wrangling

- Identify missing values and the percentage of missing values
- Identify the data type of each columns
- Check the unique values of Launch Site, Orbit and Outcomes and the frequency of each values
- Create columns Class with failure outcome.
- Save the file to format .csv

https://github.com/danielpedrini/capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

# EDA with Data Visualization

- Charts used in this project.
  - Seaborn catplot: we used this type of graphic to plot how the data behavior between different categories in each class
  - Seaborn barplot: We used this plot to check the mean values of class in each orbit
  - Seaborn lineplot: We used this plot to see the evolution of mean value of the class peer year

https://github.com/danielpedrini/capstone/blob/main/jupyter-labs-eda-dataviz.ipynb

# EDA with SQL

- Query Used to performe EDA:
  - `select distinct(Launch_Site) from SPACEXTBL`
  - `select * from SPACEXTBL where Launch_Site like 'CCA%' limit 5`
  - `select sum(PAYLOAD_MASS__KG_) from SPACEXTBL`
  - `select avg(PAYLOAD_MASS__KG_) from SPACEXTBL where Booster_Version like 'F9 v1.1%'`
  - `select min(date) from SPACEXTBL where "Landing _Outcome" = 'Success'`
  - `select distinct(Booster_Version) from SPACEXTBL where "Landing _Outcome" = 'Success (drone ship)' and PAYLOAD_MASS__KG_ between 4000 and 6000`
  - `select count(Mission_Outcome) from SPACEXTBL where Mission_Outcome like '%Success'`
  - `select count(Mission_Outcome) from SPACEXTBL where Mission_Outcome not like '%Success'`
  - `select Booster_Version from SPACEXTBL where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTBL)`
  - `elect substr(Date, 4,2) as month, "Landing _Outcome", Booster_Version, launch_site, substr(Date, 7,4) from SPACEXTBL where "Landing _Outcome" = 'Failure (drone ship)' and substr(Date, 7,4) = '2015'`

https://github.com/danielpedrini/capstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

# Build an Interactive Map with Folium

- folium.Marker() used to create marks on map
- folium.Circle() used to create a circles above ther markers
- folium.Icon() used to create an icon on the map
- folion.PolyLine() used to create polynomial line between points
- folium.pluguins.AntPath() used to create animated line between points
- markerCluster used to simplifiy the map

https://github.com/danielpedrini/capstone/blob/main/lab_jupyter_launch_site_location.ipynb

# Build a Dashboard with Plotly Dash

- Dash and html components we were used and almost everything depends on them like graphs, tables, dropbox
- Ploty used to plot graphs
- Pandas used to manipulate the data
- Piechart and scatterchart to present the chart data
- Slider to select the mass range
- Dropdown to select the sites

https://github.com/danielpedrini/capstone/blob/main/dashboard.py

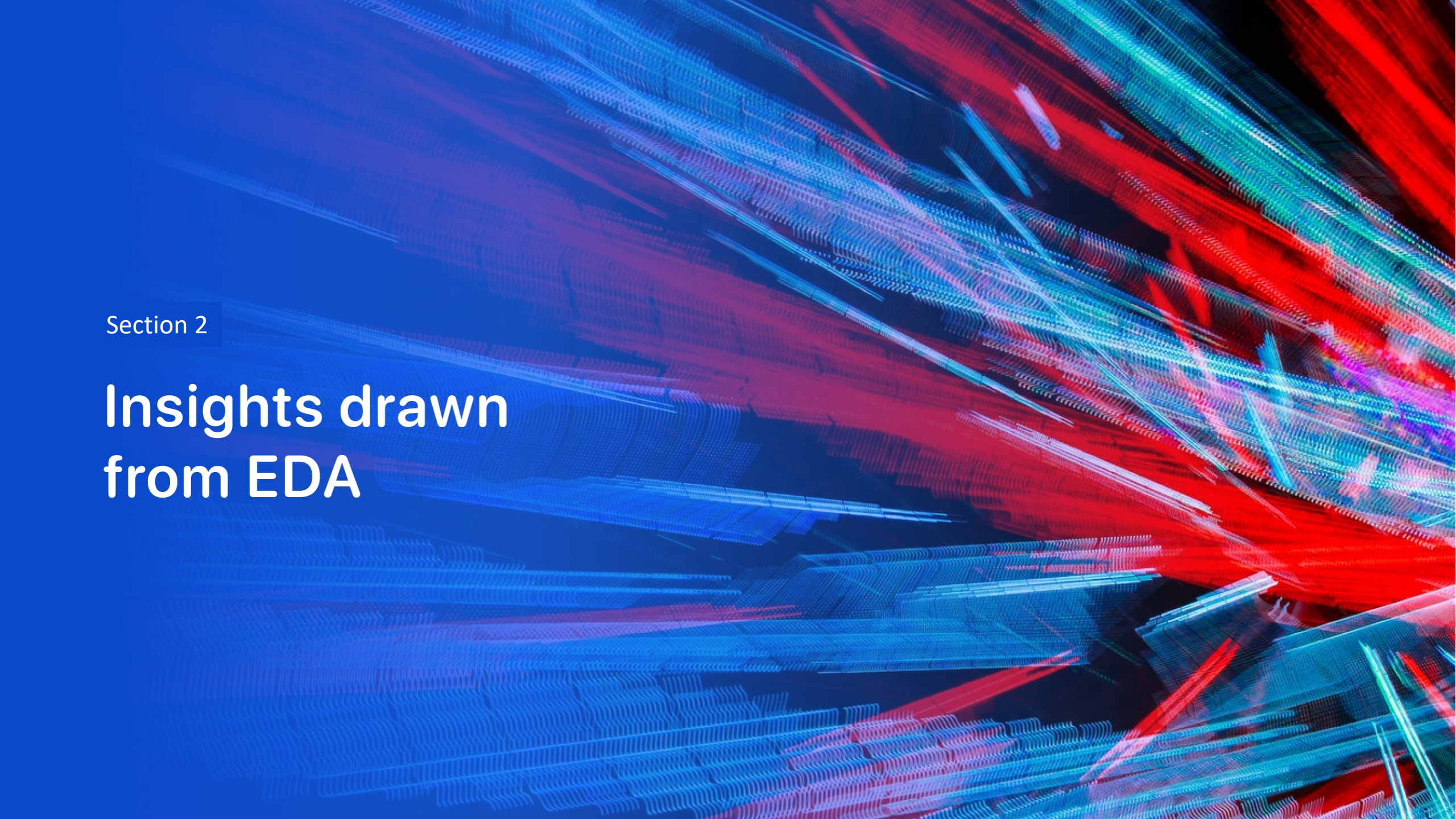
# Predictive Analysis (Classification)

- Separate the Features and the Target in two DataFrame
- Transform the Feature to best predict the model using StandardScaler
- Split the data into X_train, X_test, Y_train and Y_test
- Select four models to analyze: Logistic Regression, SVM, Decision Tree and KNN
- To find the best tunning, we used GridSearch to set the parameters and test all the parameters defined
- Tested all the models defined with some parameters
- Check the results with: Accuracy, R2_score for train and test Values
- Check the Confusion Matrix for each best parameters result on the test
- Compare best parameters result between the models

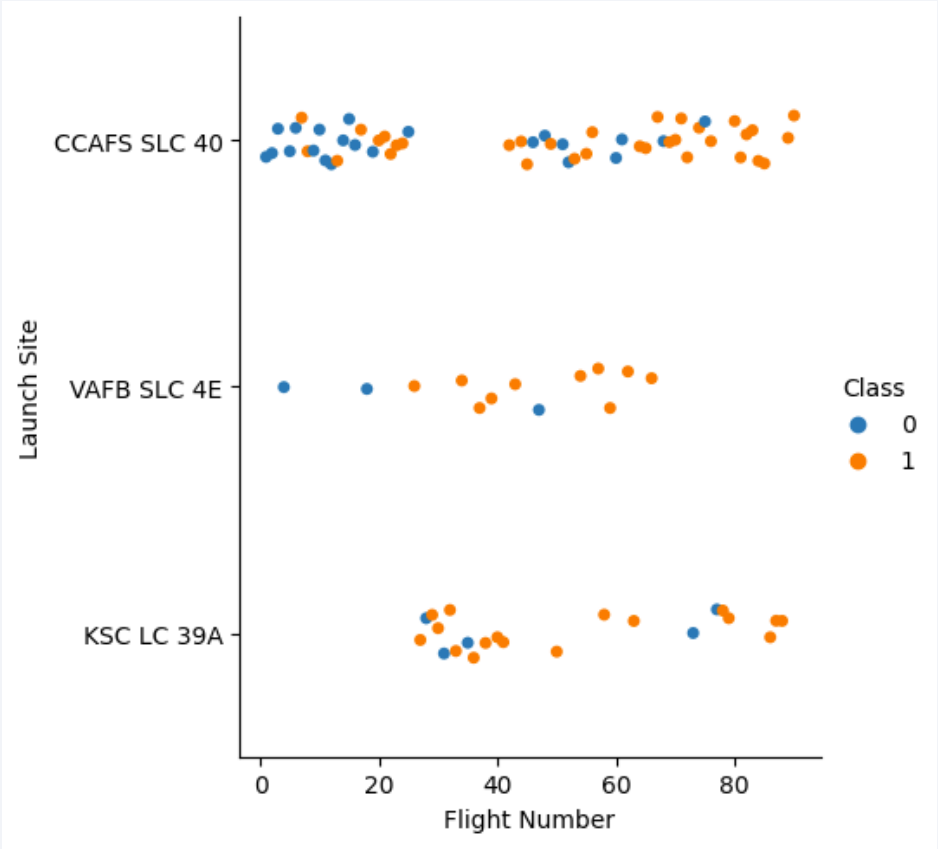
https://github.com/danielpedrini/capstone/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

# Results

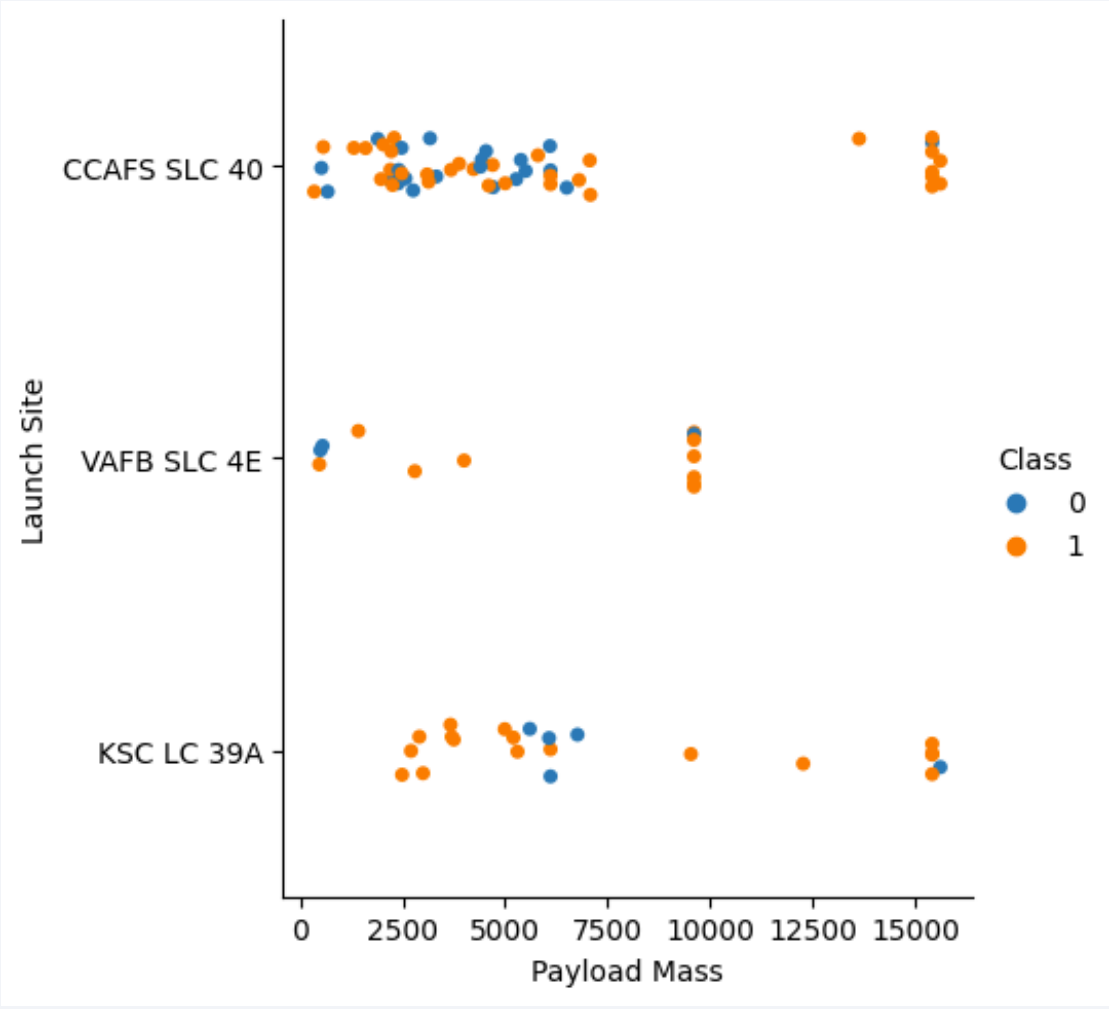
- The site with highest score was KSC LC-39A
- The Payload of 0 to 5000kg was more diverse then 6000kg to 10000kg
- Decision Tree was the best optimal model with the accuracy of 0.8732


Section 2
Insights drawn from EDA

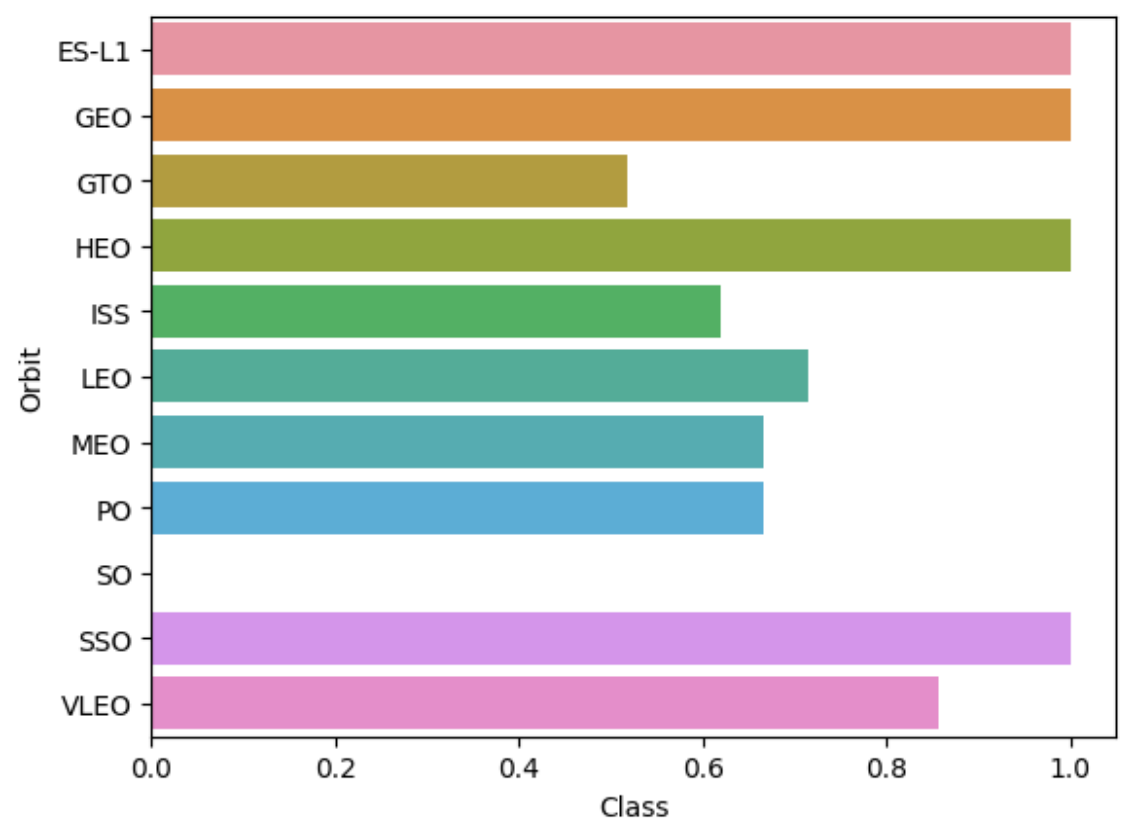
# Flight Number vs. Launch Site



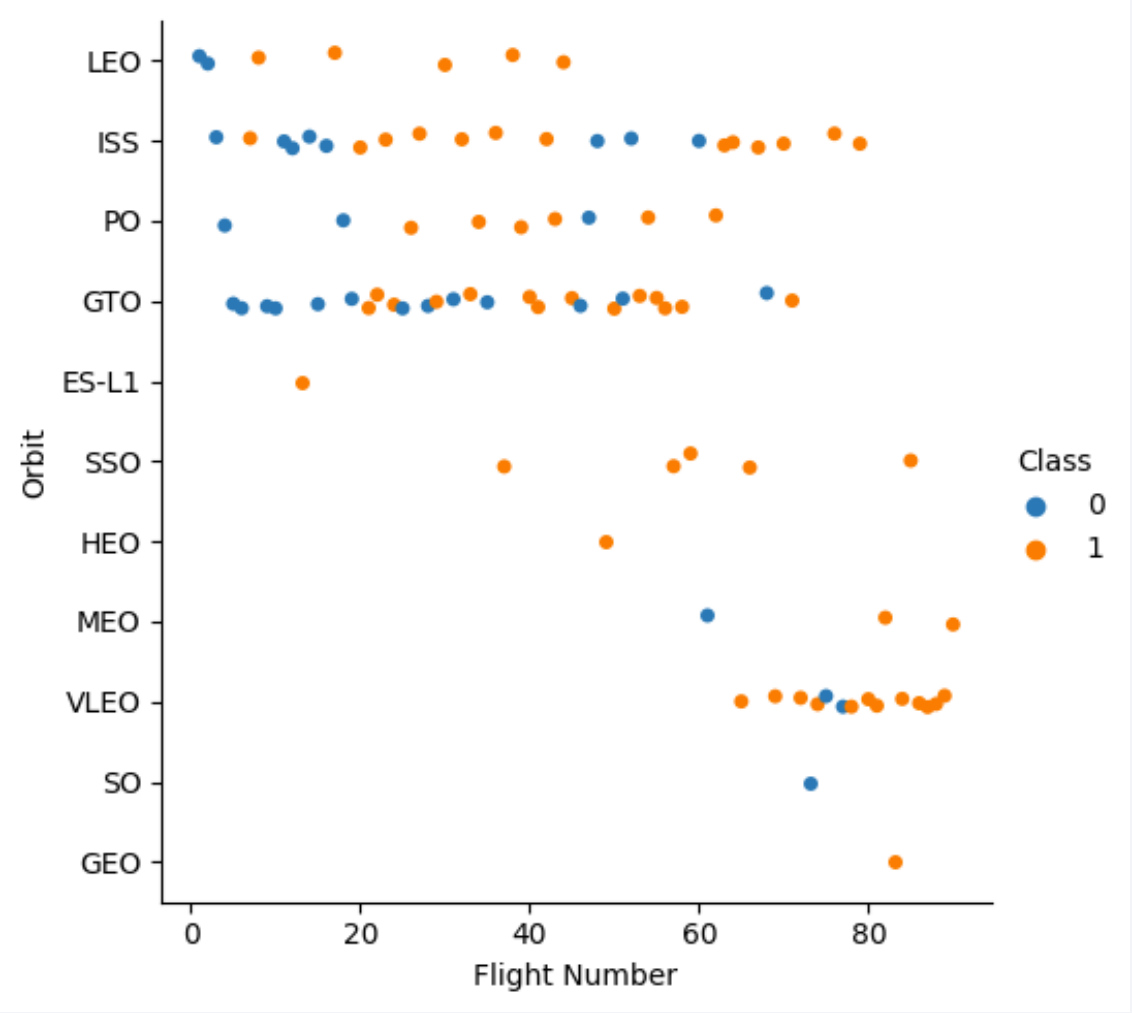
# Payload vs. Launch Site



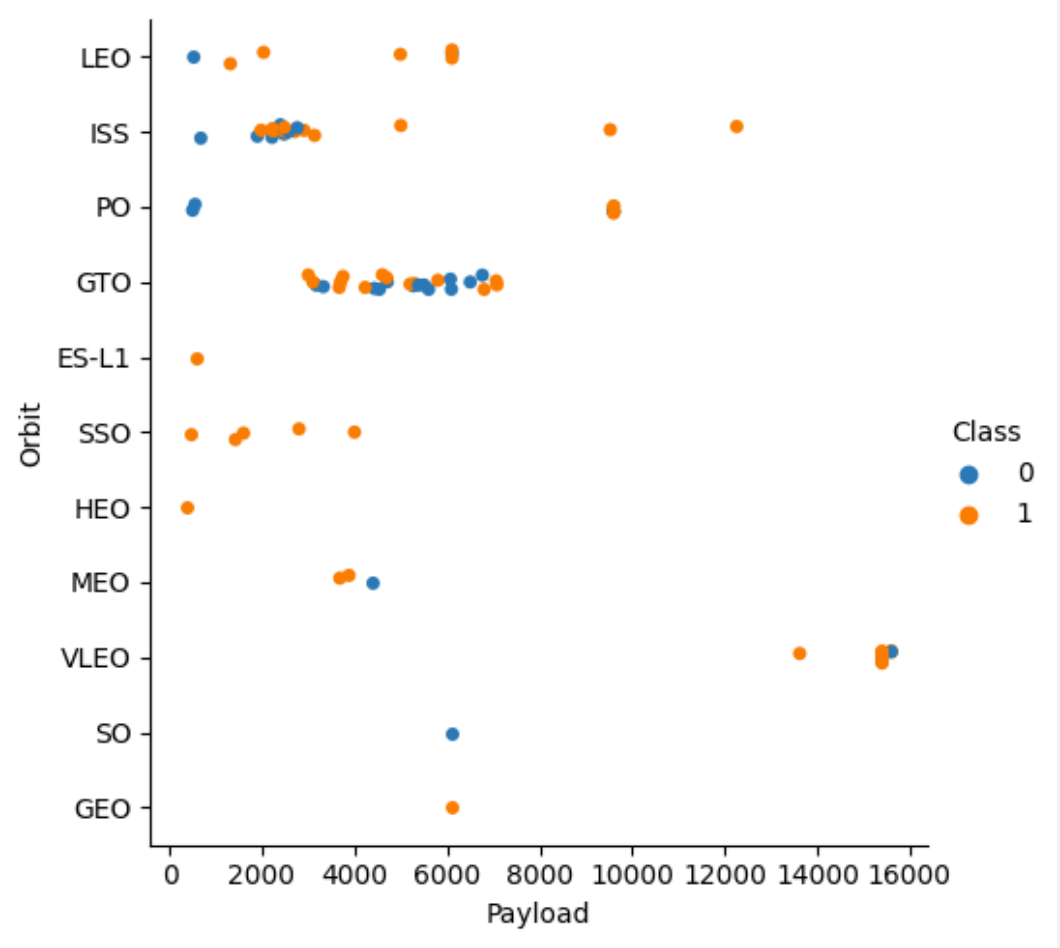
# Success Rate vs. Orbit Type



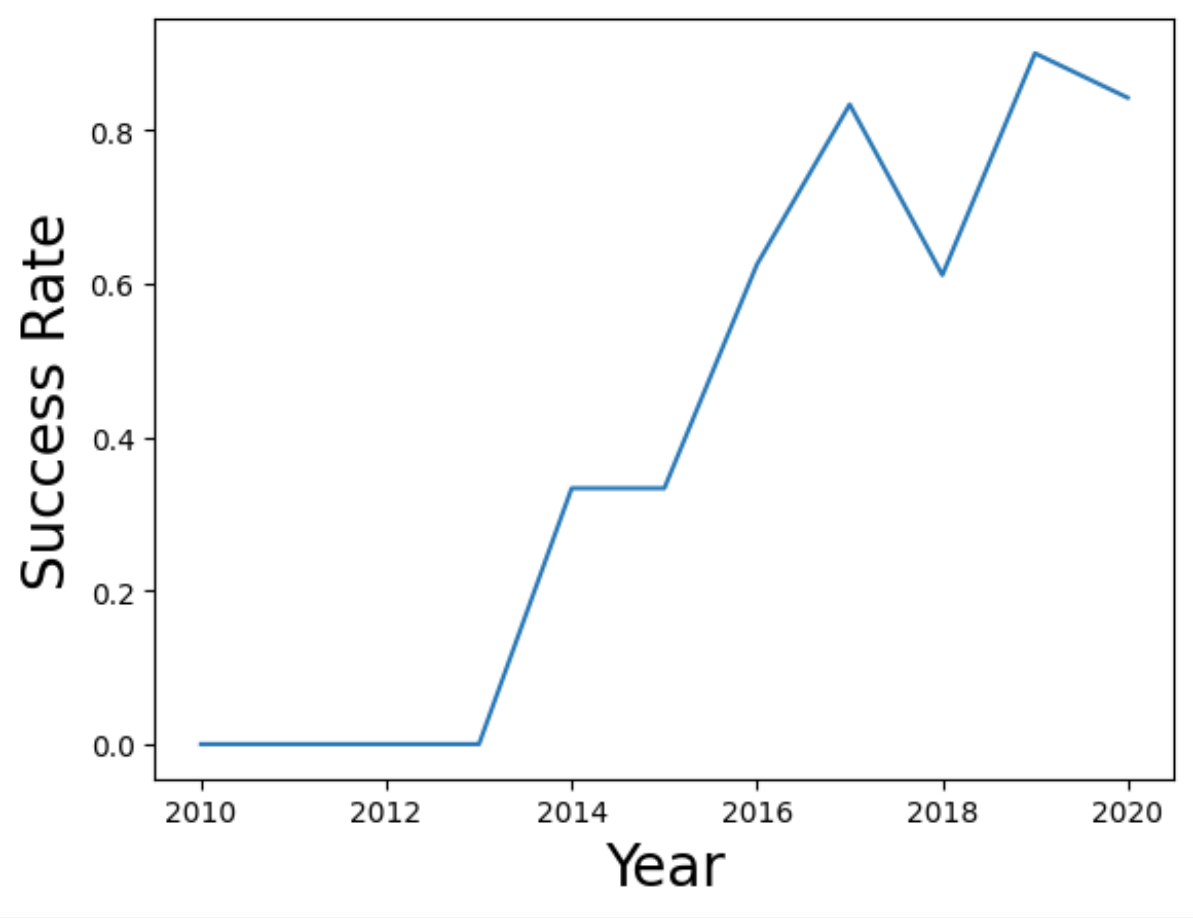
# Flight Number vs. Orbit Type



# Payload vs. Orbit Type



# Launch Success Yearly Trend



# All Launch Site Names

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

✓ 0.0s

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

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

# Launch Site Names Begin with 'CCA'

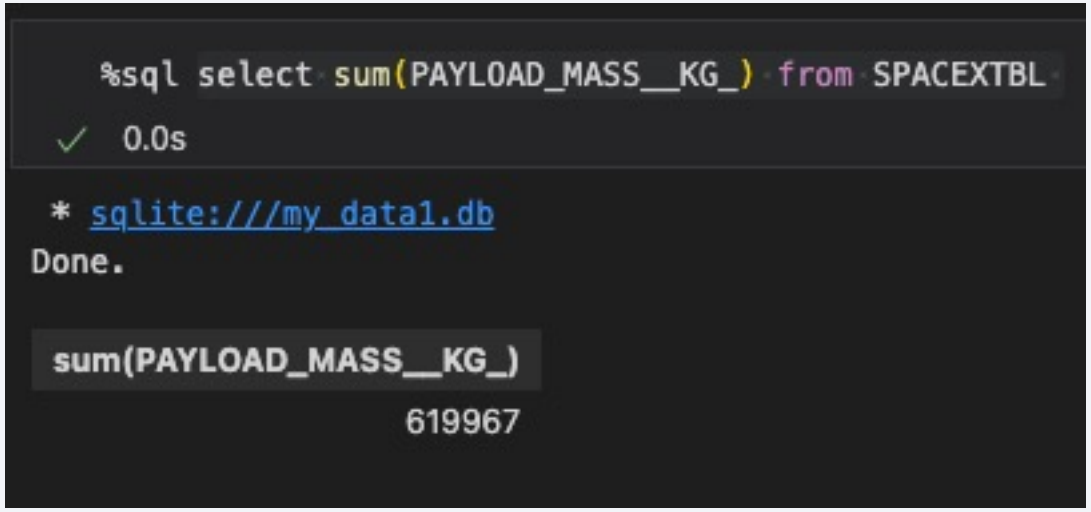
```
%%sql sqlite:///my_data1.db
select launch_site from SPACEXTBL where Launch_Site like 'CCA%' limit 5
```

✓ 0.0s

Done.

| Launch_Site |
| --- |
| CCAFS LC-40 |
| CCAFS LC-40 |
| CCAFS LC-40 |
| CCAFS LC-40 |
| CCAFS LC-40 |

# Total Payload Mass



# Average Payload Mass by F9 v1.1

```
%sql select avg(PAYLOAD_MASS__KG_) from SPACEXTBL where Booster_Version like 'F9 v1.1%'
✓ 0.0s

 * sqlite:///my_data1.db
Done.

avg(PAYLOAD_MASS__KG_)
2534.6666666666665
```

# First Successful Ground Landing Date

```
%sql select min(date) from SPACEXTBL where "Landing _Outcome" = 'Success'
```

✓ 0.0s

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

| min(date) |
| --- |
| 02-03-2019 |

# Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql select distinct(Booster_Version) from SPACEXTBL where "Landing _Outcome" = 'Success (drone ship)' and PAYLOAD_MASS__KG_ between 4000 and 6000
```

✓ 0.0s

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

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

# Total Number of Successful and Failure Mission Outcomes

```
%sql select count(Mission_Outcome) from SPACEXTBL where Mission_Outcome like '%Success';
```

✓ 0.0s

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

| count(Mission_Outcome) |
|---|
| 98 |

```
%sql select count(Mission_Outcome) from SPACEXTBL where Mission_Outcome not like '%Success';
```

✓ 0.0s

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

| count(Mission_Outcome) |
|---|
| 3 |

# Boosters Carried Maximum Payload

```
%sql select Booster_Version from SPACEXTBL where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTBL)
```

✓ 0.0s

```
 * 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 |

# 2015 Launch Records

```
_Outcome", Booster_Version, launch_site, substr(Date, 7,4) from SPACEXTBL where "Landing _Outcome" = 'Failure (drone ship)' and substr(Date, 7,4) = '2015'
```

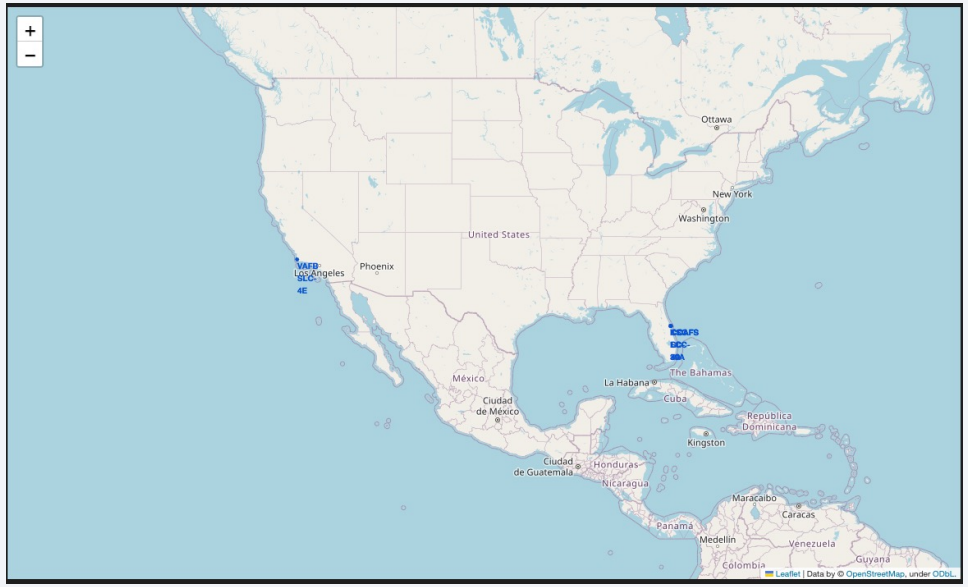
4] ✓ 0.0s Python

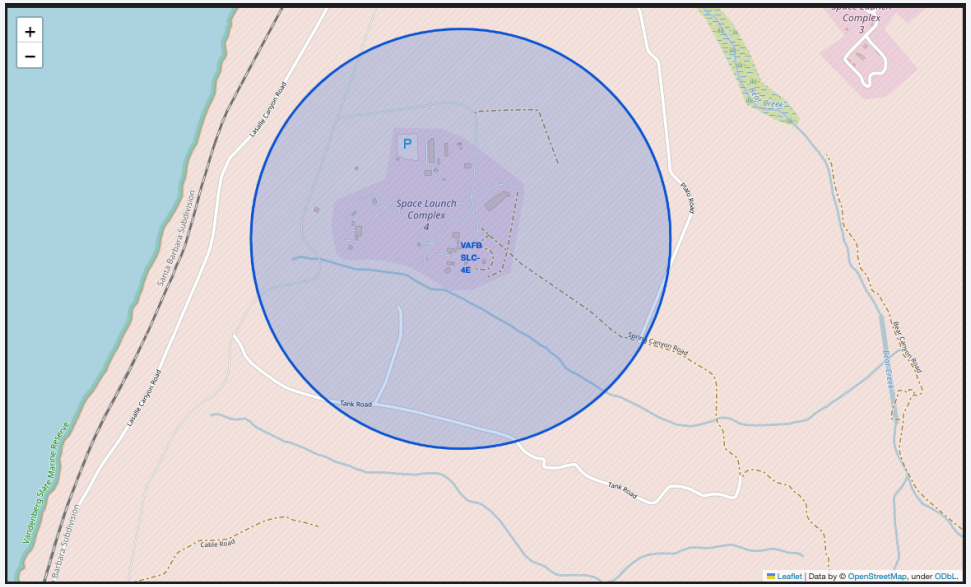
```
 * sqlite:///my_data1.db
Done.
```

| month | Landing _Outcome | Booster_Version | Launch_Site | substr(Date, 7,4) |
|---|---|---|---|---|
| 01 | Failure (drone ship) | F9 v1.1 B1012 | CCAFS LC-40 | 2015 |
| 04 | Failure (drone ship) | F9 v1.1 B1015 | CCAFS LC-40 | 2015 |

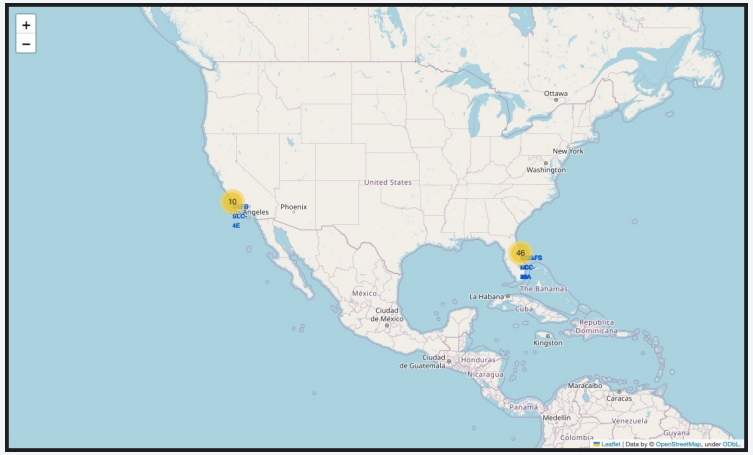

Section 3
Launch Sites
Proximities Analysis

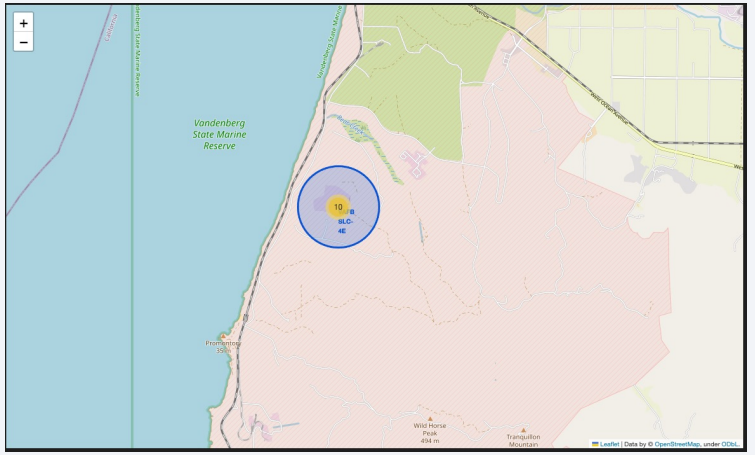
# Map One – Marks and Circle



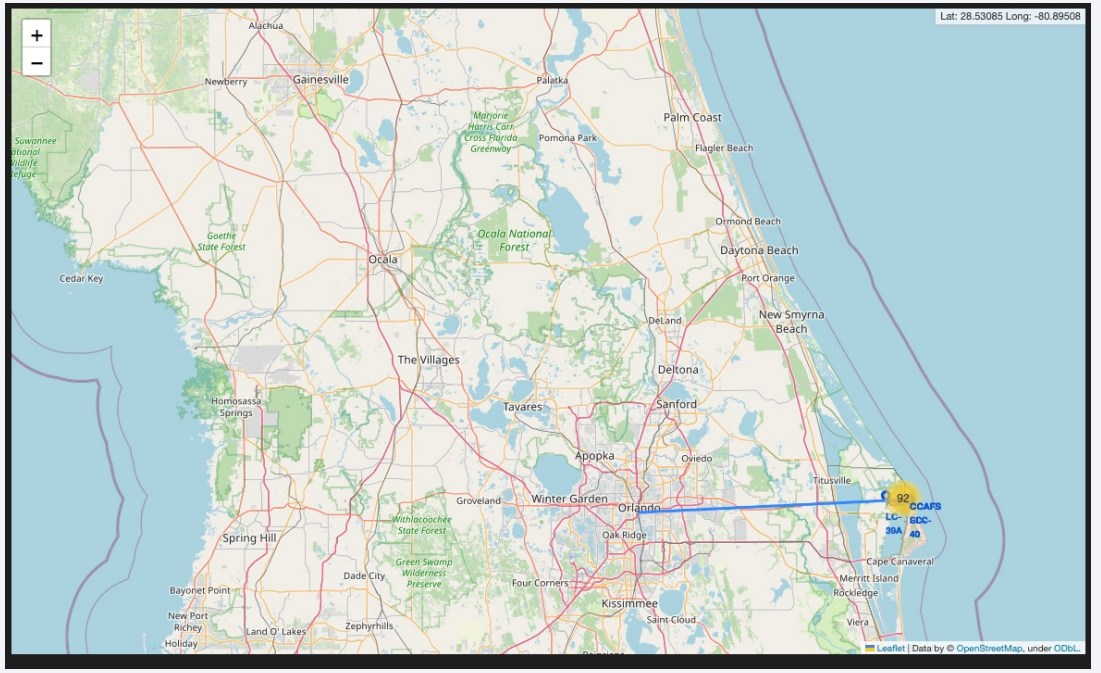
# Map 2 – Cluster and Success /Failure

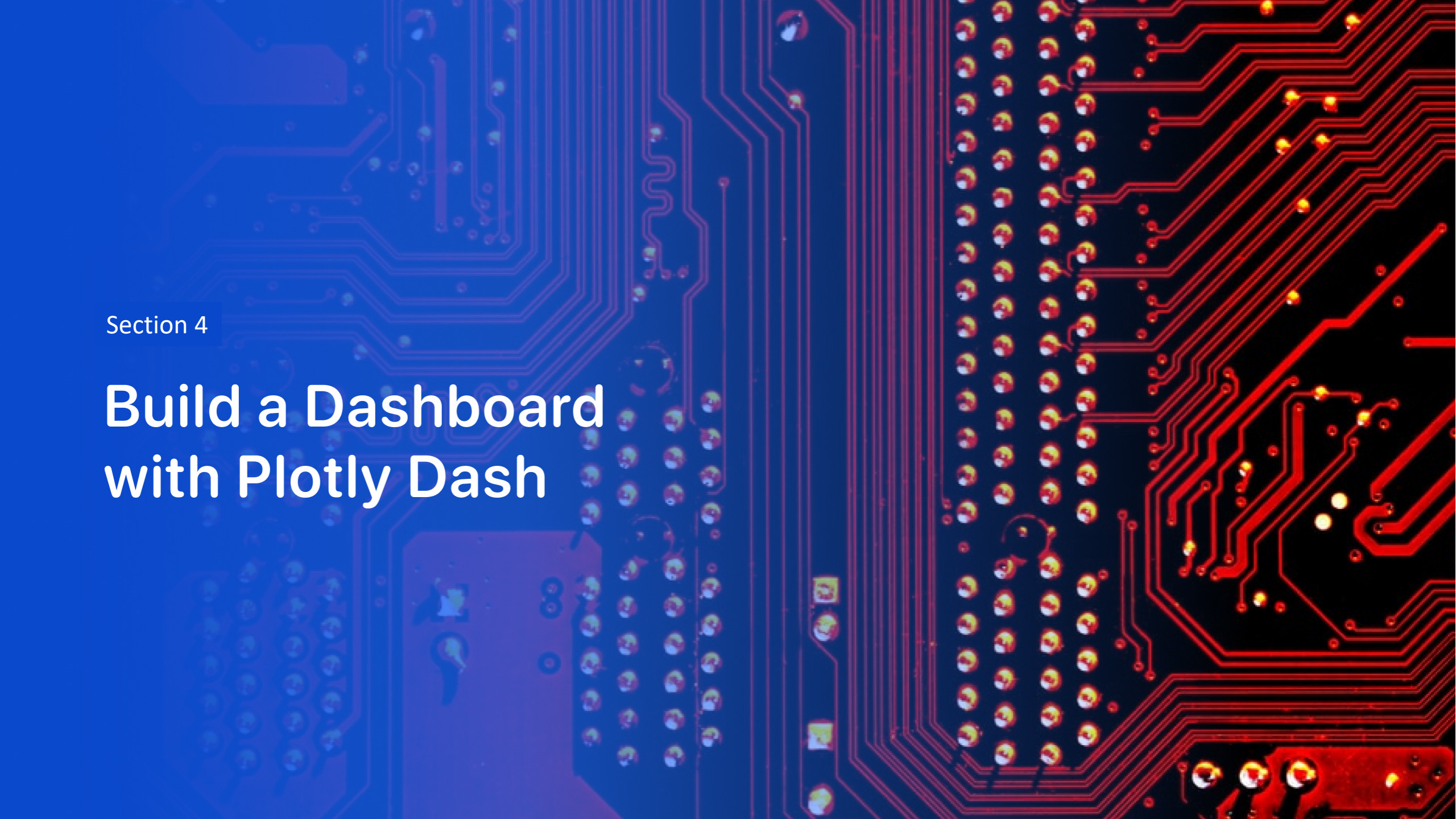


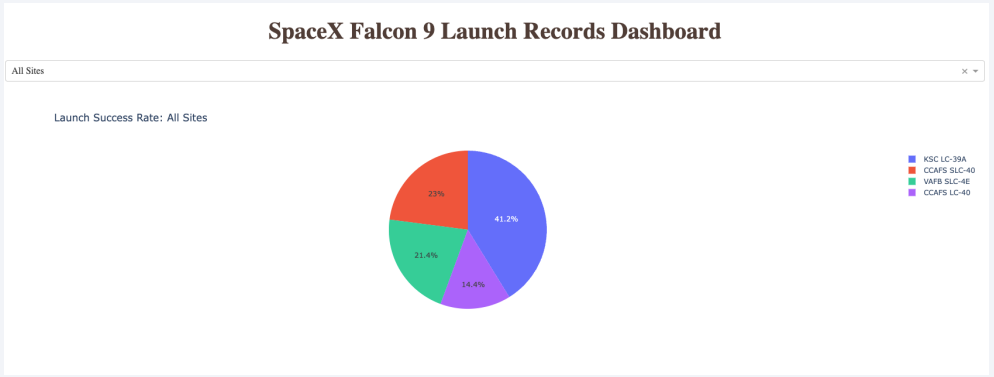

# Map 3 - Line

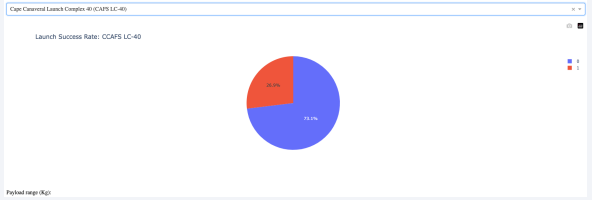

Lat: 28.56324 Long: -80.57927
14
CCAFS SLC-40
52
Leaflet | Data by © OpenStreetMap, under ODbL.

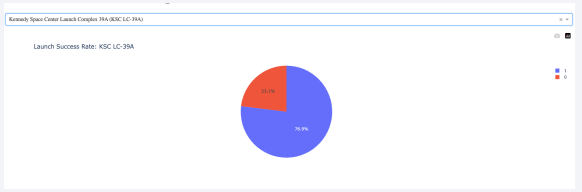

Lat: 28.53085 Long: -80.89508
Gainesville
Ocala
The Villages
Tavares
Sanford
Orlando
Winter Garden
Kissimmee
Daytona Beach
Palm Coast
92
CCAFS SLC-40
Leaflet | Data by © OpenStreetMap, under ODbL.

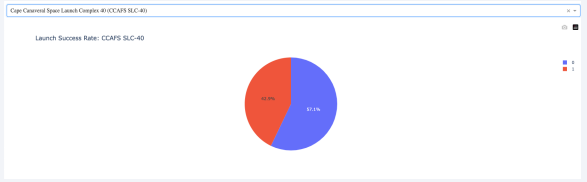

Section 4
Build a Dashboard with Plotly Dash

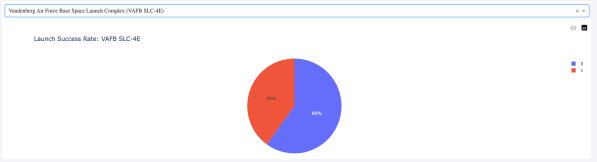
# Dashbord


SpaceX Falcon 9 Launch Records Dashboard
All Sites
Launch Success Rate: All Sites
41.2%
23%
21.4%
14.4%
KSC LC-39A
CCAFS SLC-40
VAFB SLC-4E
CCAFS LC-40

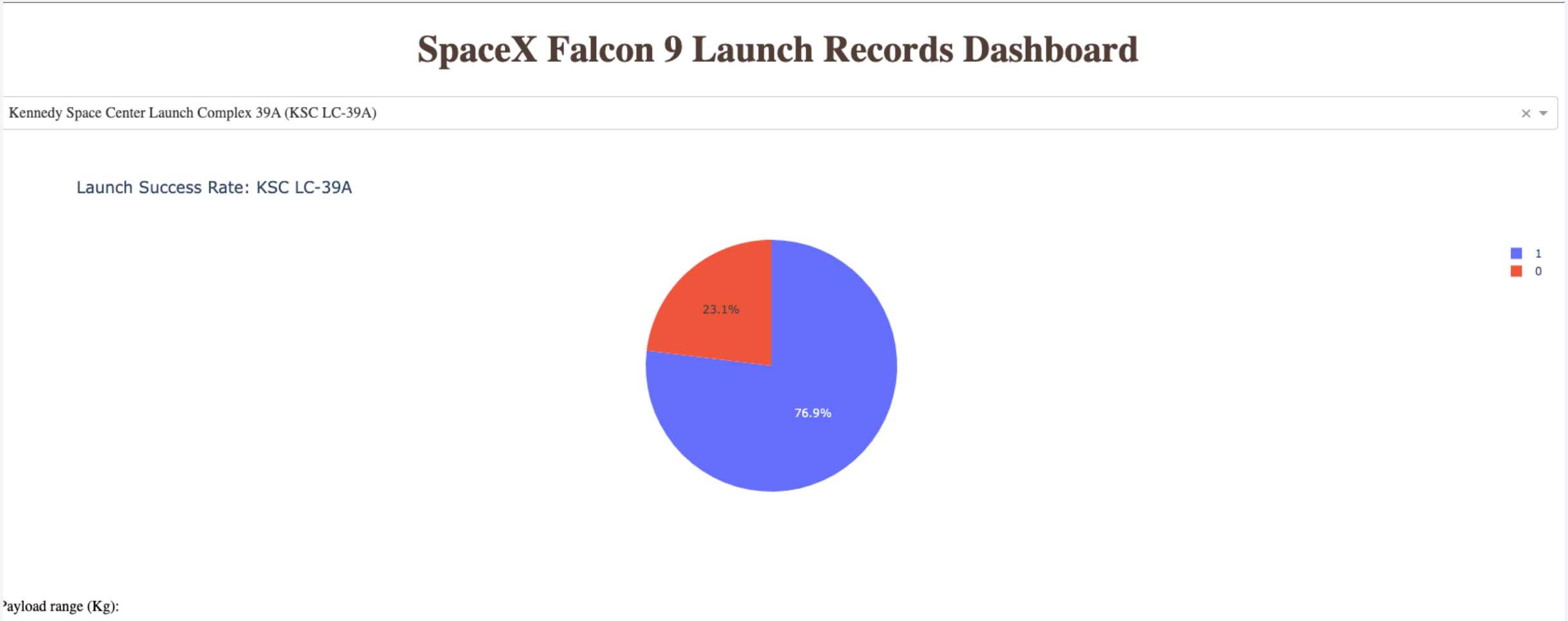

Launch Success Rate: CCAFS LC-40
Payload range (Kg):


Launch Success Rate: KSC LC-39A

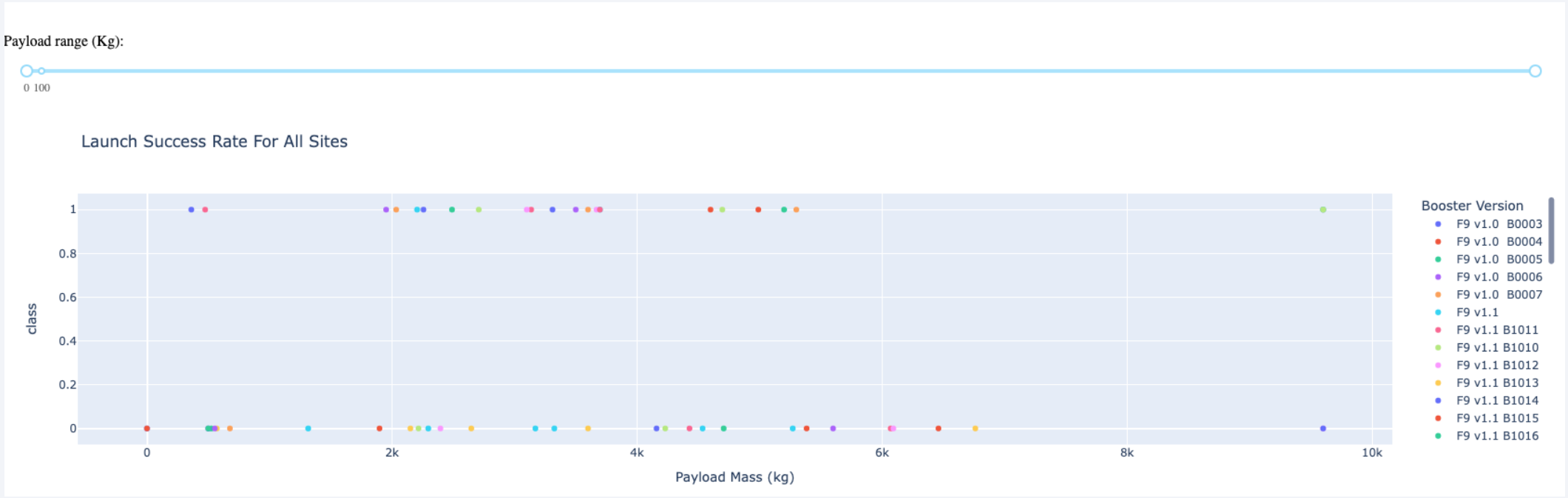

Launch Success Rate: CCAFS SLC-40


Launch Success Rate: VAFB SLC-4E

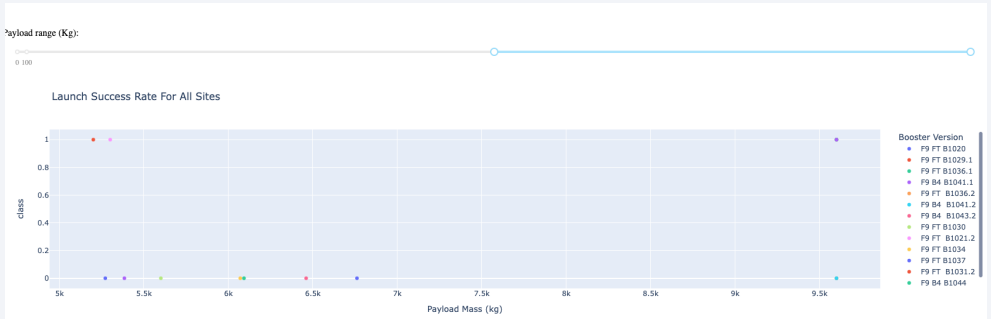
# Dashboard highest success



# <Dashboard Screenshot 3>



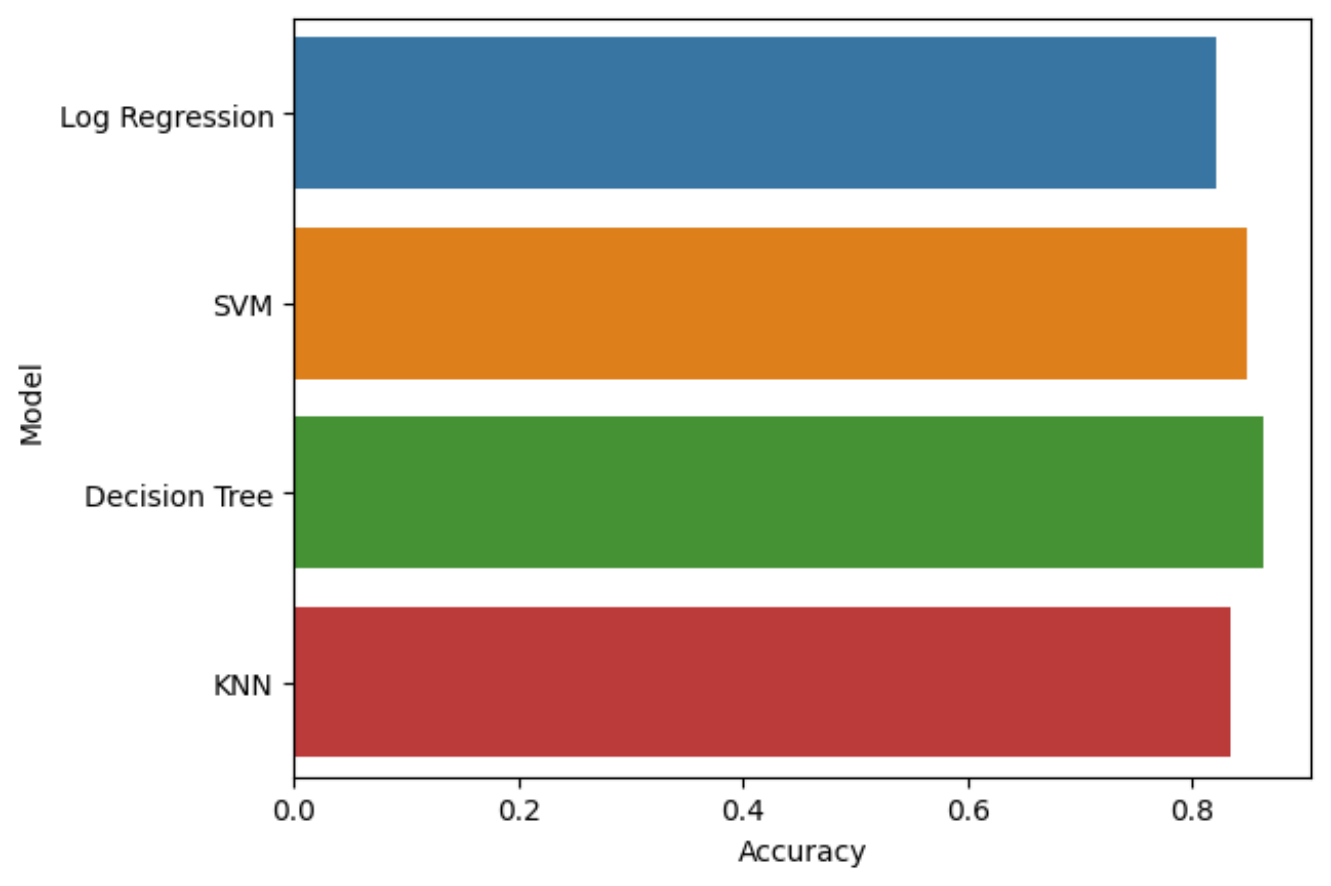




Section 5

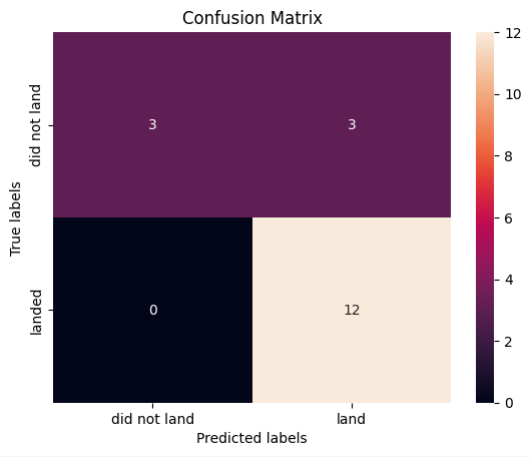
# Predictive Analysis (Classification)

# Classification Accuracy

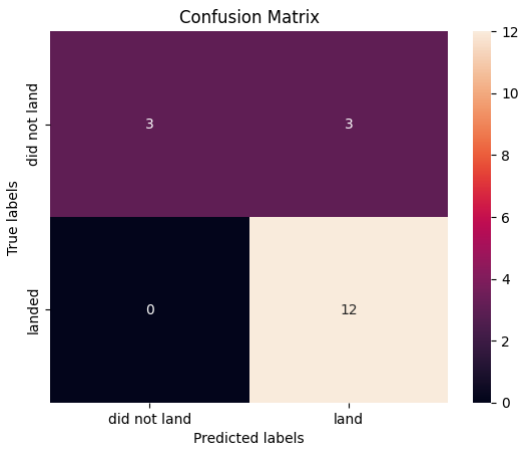


# Confusion Matrix

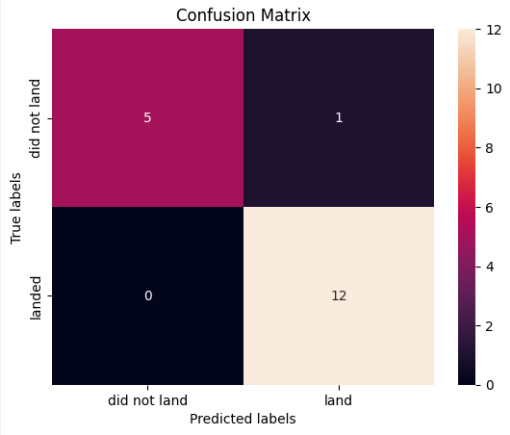
## Logistic Regression



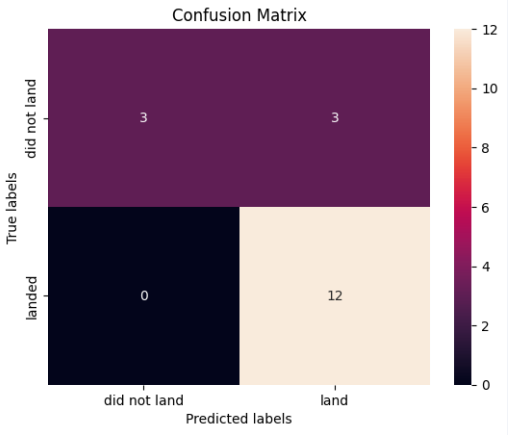

## SVM




## Decision Tree




## KNN



# Conclusions

- Decision Tree has the best Accuracy
- Comparing the R2_score in Test data, Decision tree has the best result to, with 0.94
- Using the GridSearch, we could find the best hyperparameter for each model, and with the results compere each one to decide the best model

# Appendix

- Access to the github project

https://github.com/danielpedrini/capstone

# Thank you!