IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Trung Nguyen
2023-07-03

Outline

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

Executive Summary

- This research crawled data from web then applied EDA to get overview of data. Finally, using several ML methods (logistic regression, KNN, Decision Tree, SVM) to predict the probability of a SpaceX landed or not
- Predict correct ~84%

Introduction

- 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

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Request to the SpaceX API to get data
- Perform data wrangling
  - Cleaning data (correct data type, keep needed column…)
  - Apply onehot
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models

Data Collection

- Request SpaceX API to get data.
- Github: https://github.com/TrungNguyenDA/Ds_course/blob/main/Get%20Data%20form%20API.ipynb

Data Wrangling

- Cleaning data (correct data type, keep needed column…)
- Link github: https://github.com/TrungNguyenDA/Ds_course/blob/main/jupyter-labs-webscraping.ipynb

EDA with Data Visualization

- Link github: https://github.com/TrungNguyenDA/Ds_course/blob/main/jupyter-labs-eda-sql-coursera.ipynb

EDA with SQL

- Link github: https://github.com/TrungNguyenDA/Ds_course/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

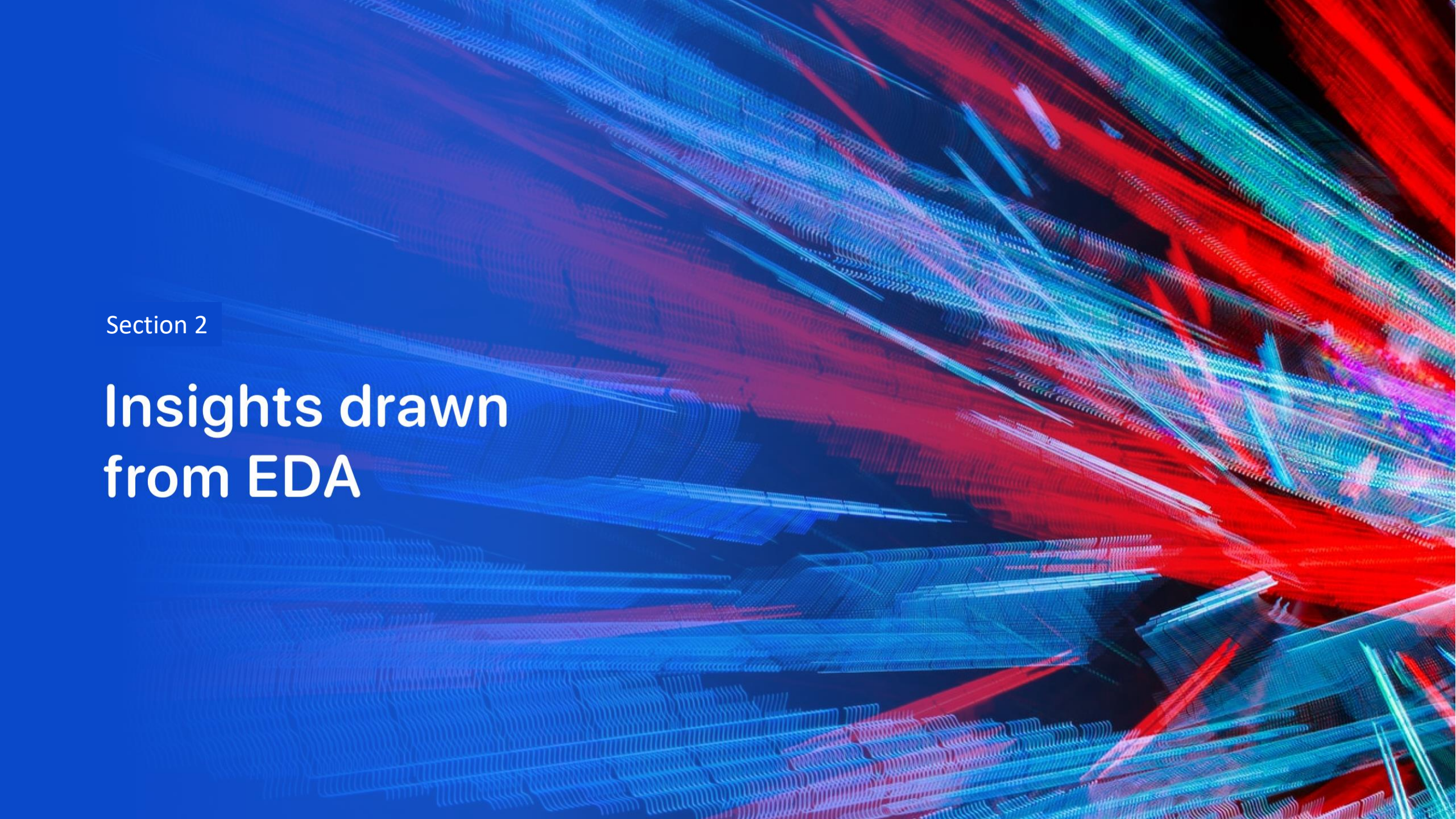
- Link github: https://github.com/TrungNguyenDA/Ds_course/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

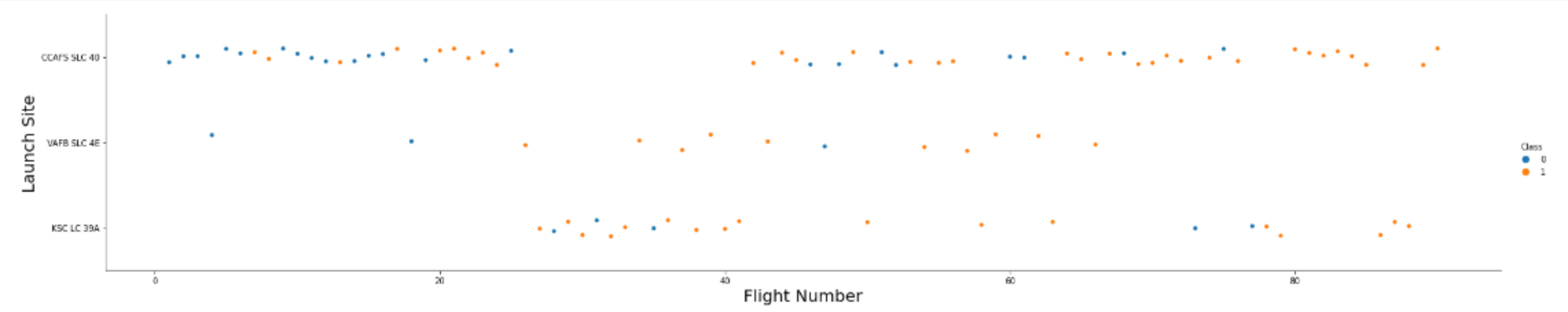
- Link github: https://github.com/TrungNguyenDA/Ds_course/blob/main/lab_jupyter_launch_site_location.ipynb

Predictive Analysis (Classification)

- Model Score (logistic regression, SVM, Decision Tree) = 84%
- Link github: https://github.com/TrungNguyenDA/Ds_course/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

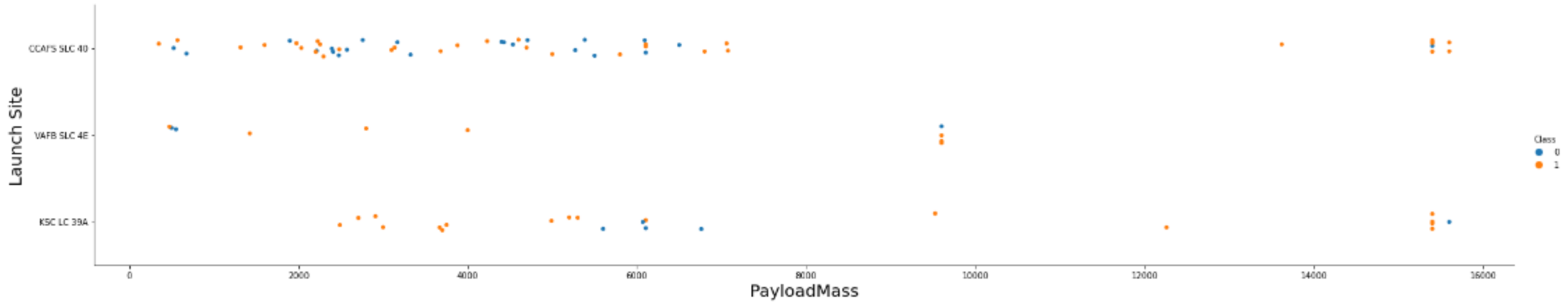

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



Payload vs. Launch Site

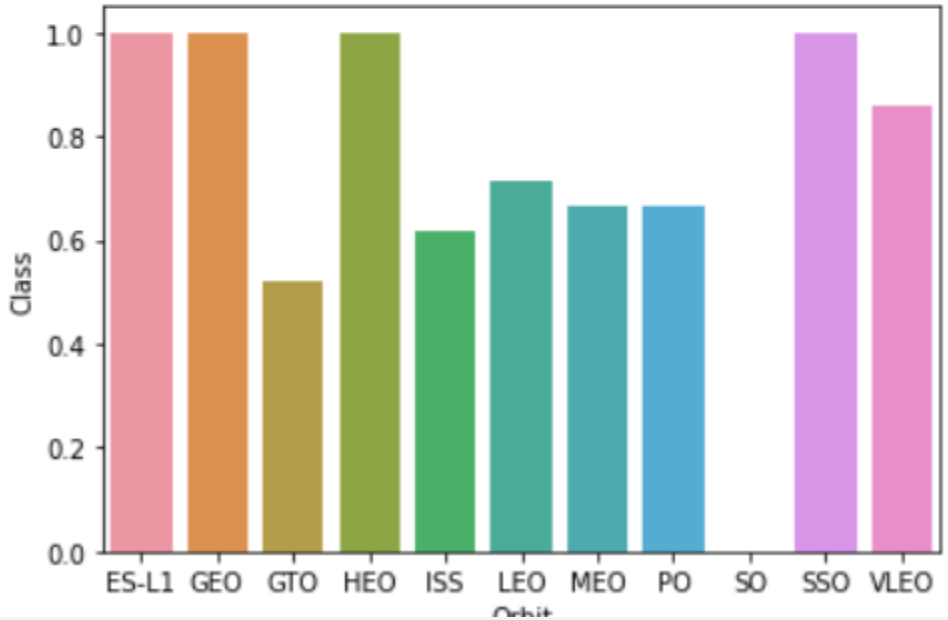
```
[5]: # Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the class v
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("PayloadMass",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



Success Rate vs. Orbit Type

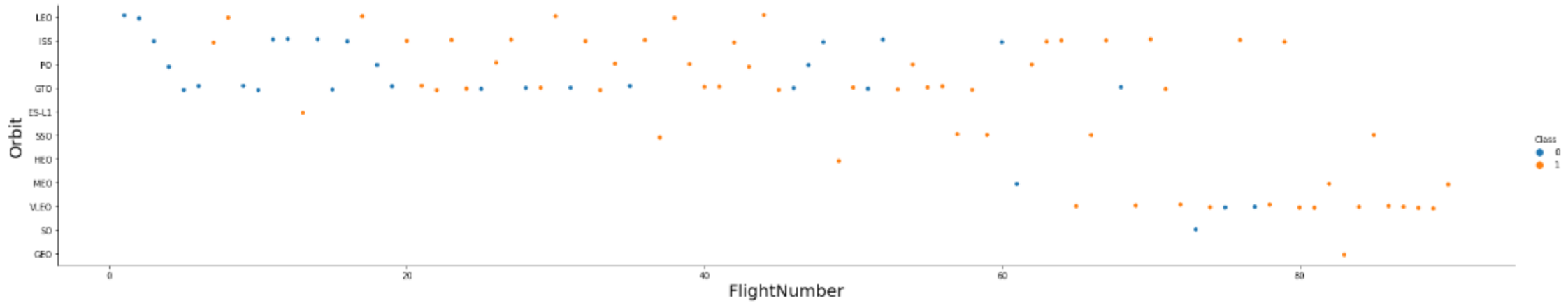
[16]:
```
# HINT use groupby method on Orbit column and get the mean of Class column
# df_success_rate =
sns.barplot(x = 'Orbit',
            y = 'Class',
            data = pd.DataFrame(df.groupby("Orbit")["Class"].mean()).reset_index())

# Show the plot
plt.show()
```



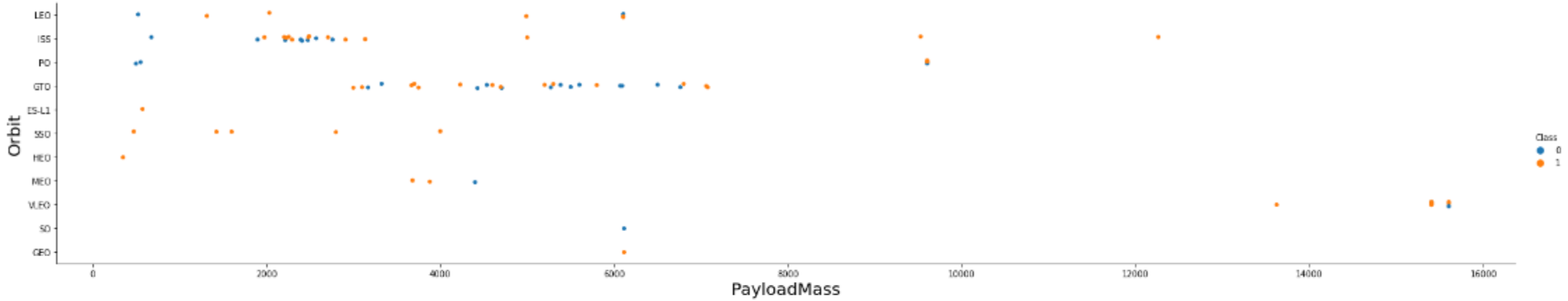
Flight Number vs. Orbit Type

[17]:
```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("FlightNumber",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



Payload vs. Orbit Type

[19]:
```
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("PayloadMass",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



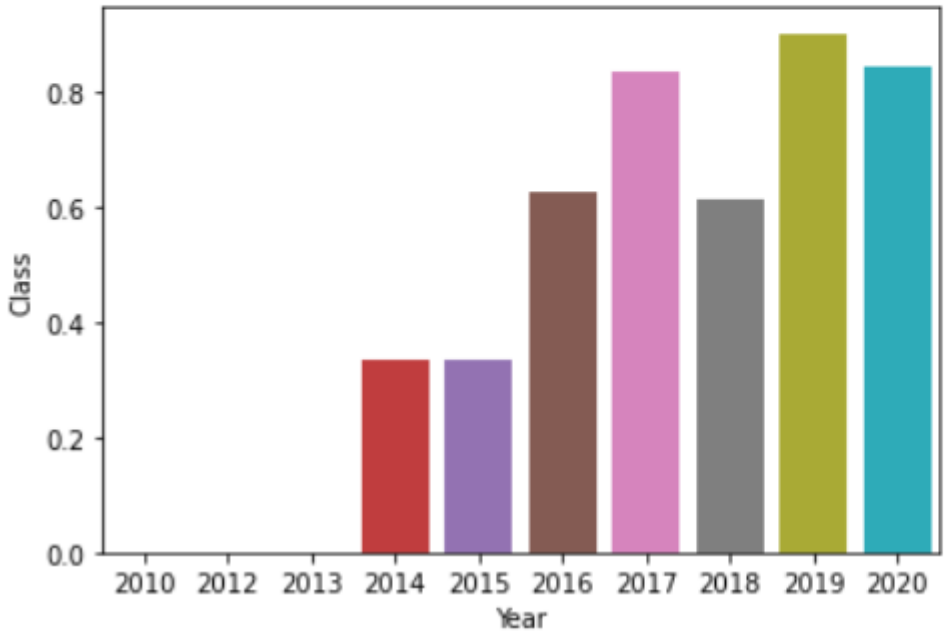
With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.

Launch Success Yearly Trend

[30]:
```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
sns.barplot(x = 'Year',
            y = 'Class',
            data = pd.DataFrame(df.groupby("Year")["Class"].mean()).reset_index())

# Show the plot
plt.show()
```



All Launch Site Names

```
df["LaunchSite"].value_counts()
```

```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
```

Launch Site Names Begin with 'CCA'

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

[20]:
```
sql_sentence = """
select *
from SPACEXTBL
where upper(Launch_Site) like 'CCA%'
limit 5"""
cur.execute(sql_sentence).fetchall()
```

[20]:
```
[('06/04/2010',
  '18:45:00',
  'F9 v1.0  B0003',
  'CCAFS LC-40',
  'Dragon Spacecraft Qualification Unit',
  0.0,
  'LEO',
  'SpaceX',
  'Success',
  'Failure (parachute)'),
 ('12/08/2010',
  '15:43:00',
  'F9 v1.0  B0004',
  'CCAFS LC-40',
  'Dragon demo flight C1, two CubeSats, barrel of Brouere cheese',
  0.0,
```

Total Payload Mass

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

```
sql_sentence = """
select sum(PAYLOAD_MASS__KG_) as total_payload_mass
from SPACEXTBL
where customer= 'NASA (CRS)'
"""
cur.execute(sql_sentence).fetchall()
```

```
[(45596.0,)]
```

Average Payload Mass by F9 v1.1

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

```
sql_sentence = """
select avg(PAYLOAD_MASS__KG_) as avg_payload_mass
from SPACEXTBL
where Booster_Version = 'F9 v1.1'
"""
cur.execute(sql_sentence).fetchall()
```

```
[(2928.4,)]
```

First Successful Ground Landing Date

List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

```
sql_sentence = """
select min(Date)
from SPACEXTBL
where Landing_Outcome = 'Success (ground pad)'
"""
cur.execute(sql_sentence).fetchall()
```

```
[('01/08/2018',)]
```

Successful Drone Ship Landing with Payload between 4000 and 6000

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

```
sql_sentence = """
select *
from SPACEXTBL
where Landing_Outcome = 'Success (drone ship)'
      and PAYLOAD_MASS__KG_ > 4000
       and PAYLOAD_MASS__KG_ < 6000
"""
cur.execute(sql_sentence).fetchall()
```

```
[('05/06/2016',
  '5:21:00',
  'F9 FT B1022',
  'CCAFS LC-40',
  'JCSAT-14',
  4696.0,
  'GTO',
  'SKY Perfect JSAT Group',
  'Success',
  'Success (drone ship)'),
 ('14/08/2016',
  '5:26:00',
```

Total Number of Successful and Failure Mission Outcomes

List the total number of successful and failure mission outcomes

```
sql_sentence = """
select Landing_Outcome
        , count(Payload) as total
from SPACEXTBL
group by Landing_Outcome
order by total desc
"""
cur.execute(sql_sentence).fetchall()
```

```
[('Success', 38),
 ('No attempt', 21),
 ('Success (drone ship)', 14),
 ('Success (ground pad)', 9),
 ('Failure (drone ship)', 5),
 ('Controlled (ocean)', 5),
 ('Failure', 3),
 ('Uncontrolled (ocean)', 2),
 ('Failure (parachute)', 2),
 ('Precluded (drone ship)', 1),
 ('No attempt ', 1),
 (None, 0)]
```

Boosters Carried Maximum Payload

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

```
sql_sentence = """
select Booster_Version
from SPACEXTBL
where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_)
                           from SPACEXTBL)
"""
cur.execute(sql_sentence).fetchall()
```

```
[('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

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.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.

```
sql_sentence = """
select substr(Date, 4, 2) as month,Landing_Outcome,Booster_Version,Launch_Site
from SPACEXTBL
where Landing_Outcome = 'Failure (drone ship)'
    and substr(Date,7,4)='2015'
"""
cur.execute(sql_sentence).fetchall()
```

```
[('10', 'Failure (drone ship)', 'F9 v1.1 B1012', 'CCAFS LC-40'),
 ('04', 'Failure (drone ship)', 'F9 v1.1 B1015', 'CCAFS LC-40')]
```

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

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.

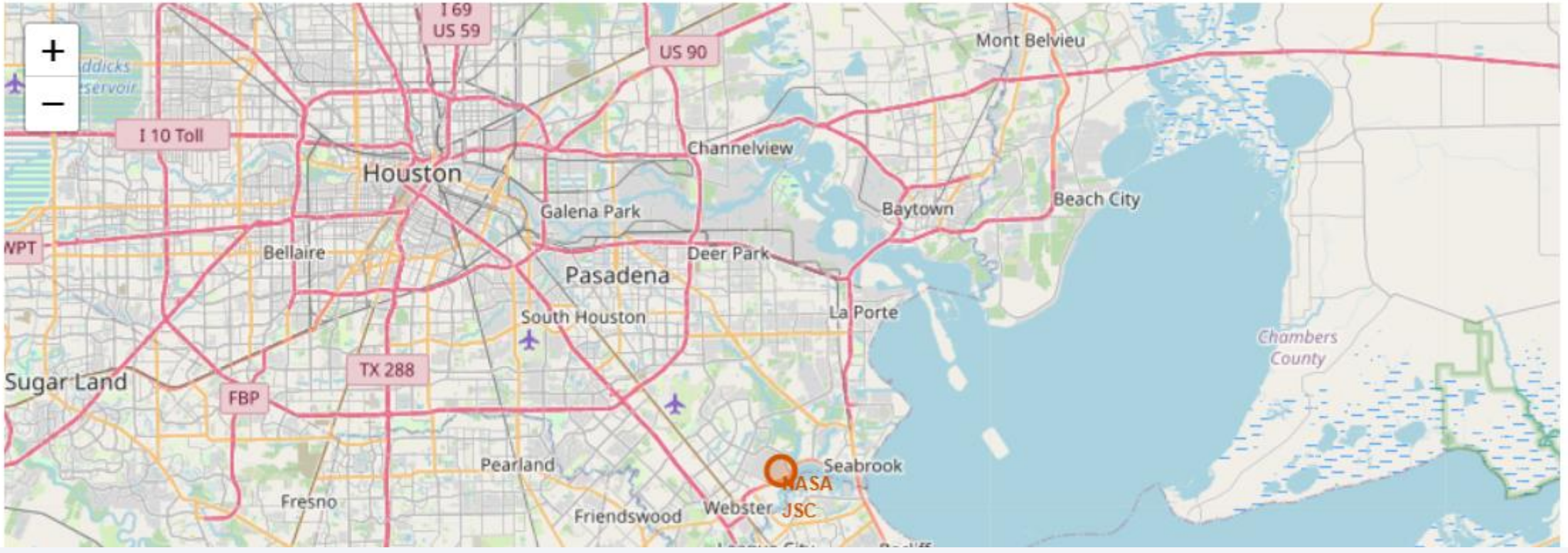
```
sql_sentence = """
select Landing_Outcome
        , count(Payload) as total
from SPACEXTBL
where Date between '04/06/2010' and '20/03/2017'
group by Landing_Outcome
order by total desc
"""
cur.execute(sql_sentence).fetchall()
```

```
[('Success', 20),
 ('No attempt', 9),
 ('Success (drone ship)', 8),
 ('Success (ground pad)', 7),
 ('Failure (drone ship)', 3),
 ('Failure', 3),
 ('Failure (parachute)', 2),
 ('Controlled (ocean)', 2),
 ('No attempt ', 1)]
```

Section 3

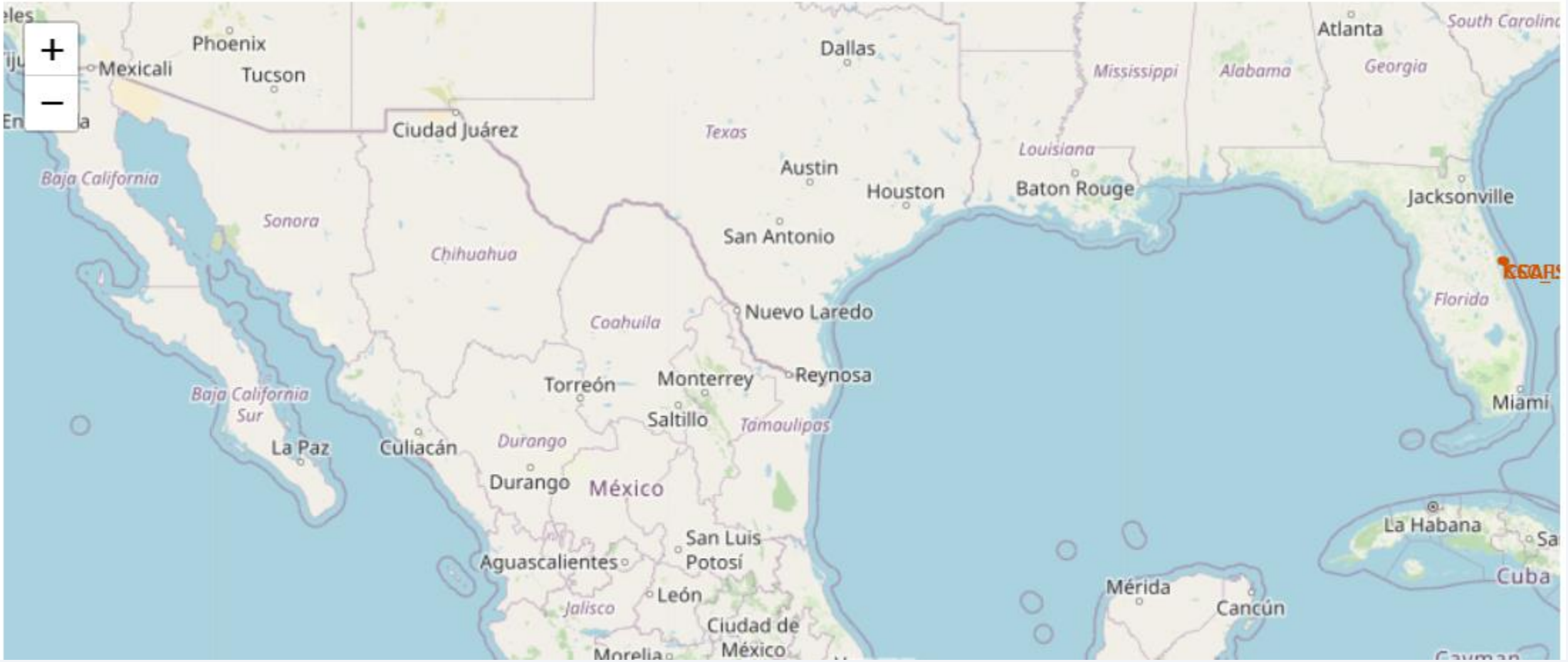
Launch Sites Proximities Analysis

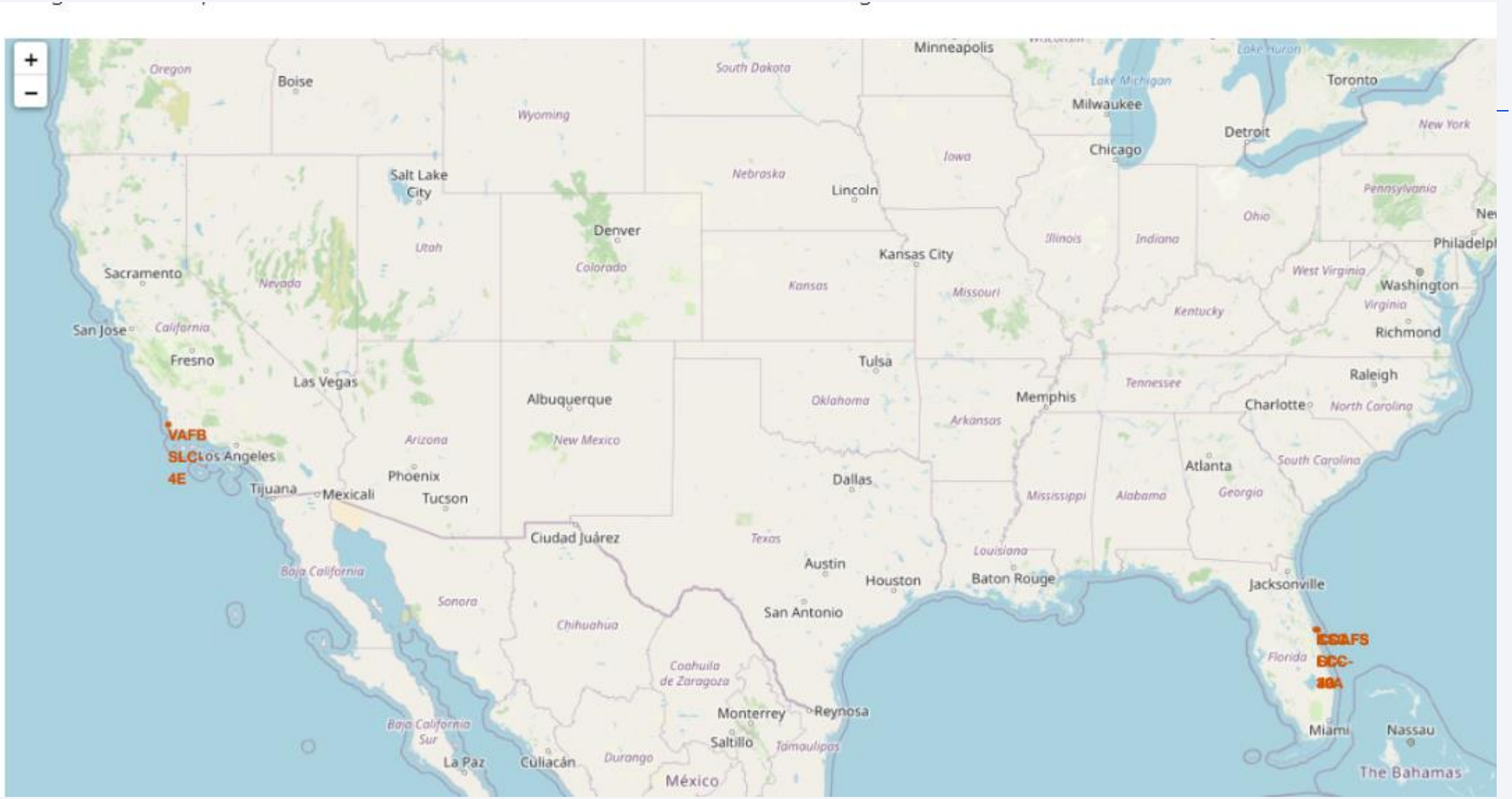
```
# Create a blue circle at NASA Johnson Space Center's coordinate with a popup label showing its name
circle = folium.Circle(nasa_coordinate, radius=1000, color='#d35400', fill=True).add_child(folium.Popup('NASA Johnson Space
# Create a blue circle at NASA Johnson Space Center's coordinate with a icon showing its name
marker = folium.map.Marker(
    nasa_coordinate,
    # Create an icon as a text label
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % 'NASA JSC',
        )
    )
site_map.add_child(circle)
site_map.add_child(marker)
```



```
site_map.add_child(KSC_LC_39A_circle)
site_map.add_child(KSC_LC_39A_marker)

# KSC_LC_39A
VAFB_SLC_4E_circle,VAFB_SLC_4E_marker = make_circle_marker(input_cordinate = VAFB_SLC_4E_coordinate,input_label = 'VAFB_SL
site_map.add_child(VAFB_SLC_4E_circle)
site_map.add_child(VAFB_SLC_4E_marker)
```




Minneapolis
Toronto
Milwaukee
Detroit
Chicago
Boise
Salt Lake City
Denver
Lincoln
Kansas City
Sacramento
San Jose
Fresno
Las Vegas
Albuquerque
Tulsa
Memphis
Raleigh
Charlotte
Richmond
Washington
Atlanta
Dallas
Phoenix
Tucson
Tijuana
Mexicali
Los Angeles
VAFB
SLC
4E
Ciudad Juárez
Austin
Houston
Baton Rouge
San Antonio
Jacksonville
ESAFS
SCC-
39A
Monterrey
Reynosa
Saltillo
La Paz
Culiacán
Miami
Nassau
The Bahamas
México
Oregon
South Dakota
Wyoming
Nebraska
Iowa
Lake Michigan
Lake Huron
New York
Pennsylvania
Ohio
Illinois
Indiana
West Virginia
Virginia
Kentucky
Missouri
Kansas
Colorado
Utah
Nevada
California
Oklahoma
Arkansas
Tennessee
North Carolina
South Carolina
Georgia
Alabama
Mississippi
Louisiana
Texas
Arizona
New Mexico
Baja California
Sonora
Chihuahua
Coahuila de Zaragoza
Durango
Tamaulipas
Baja California Sur
Florida

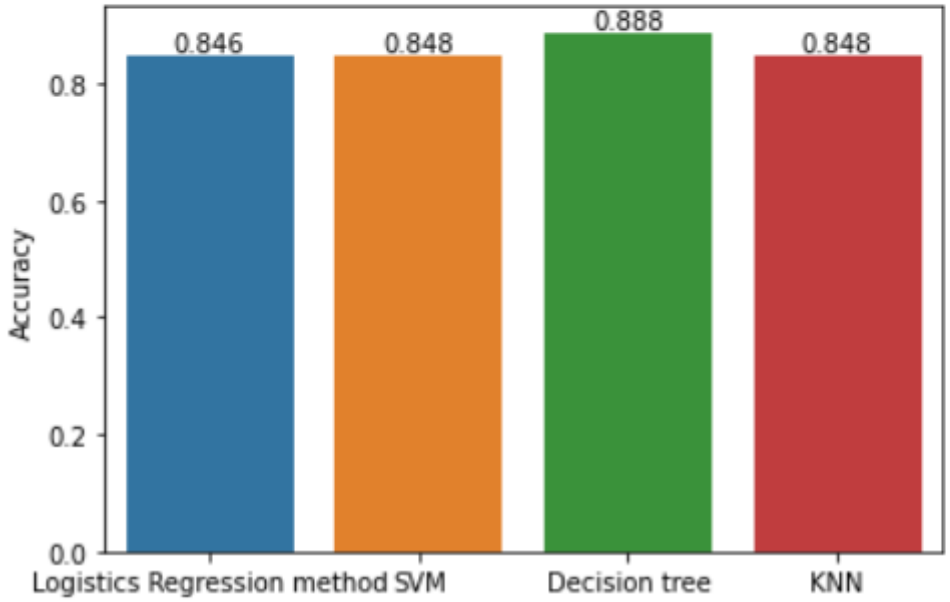
Section 5

Predictive Analysis (Classification)

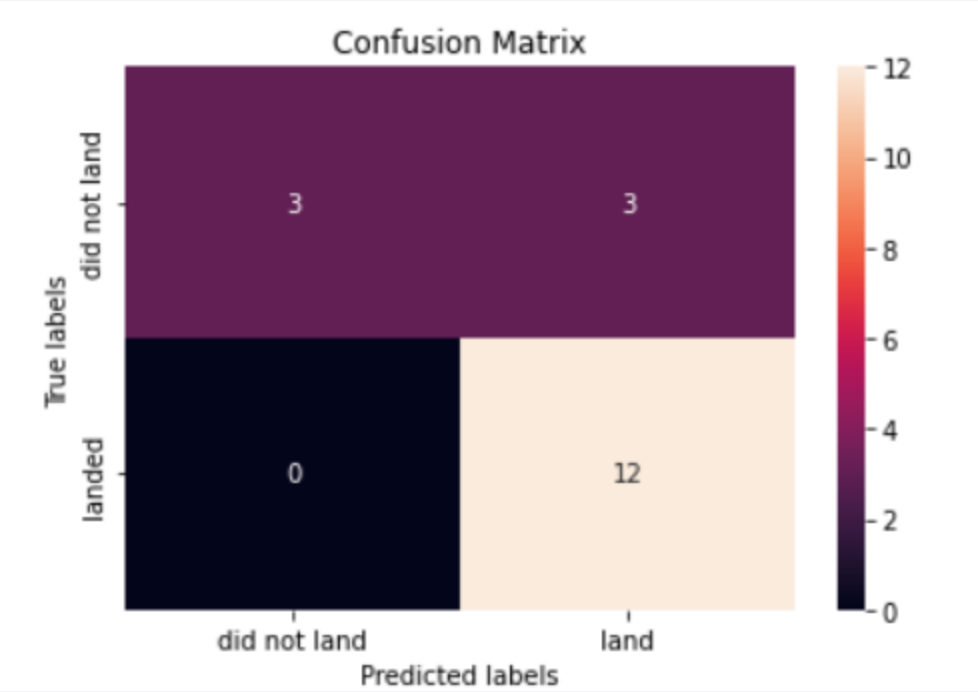
Classification Accuracy

```
import seaborn as sns
import matplotlib.pyplot as plt
ax = sns.barplot(x = 'Model',
                 y = 'Accuracy',
                 data = df_final)
ax.bar_label(ax.containers[0], fmt='%.3f')

# Show the plot
plt.show()
```



Confusion Matrix



Thank you!