IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Tahmid S
2022-07-06

Outline

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

Executive Summary

- Summary of methodologies

- Data collection and web scraping

- EDA involving data wrangling, data visualization, SQL and interactive visual analytics

- Location site visualization

- Machine learning predictions

- Summary of all results

- Real time data collected from SpaceX API

- Machine learning predictions displayed best model in terms of feature engineering and feature selection

Introduction

- Project background and context

- The purpose of this initiative was to evaluate how SpaceY would compete with SpaceX

- Problems you want to find answers

- Best places to make SpaceY launches
- Estimating total cost for launches
- Predicting landing sites and visualizations

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - SpaceX API
  - Web Scraping in python
- Perform data wrangling
  - Data collected was combined with landing outcomes and additional features
- 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

- SpaceX API
- Web scraping

Data Collection – SpaceX API

Github URL of SpaceX API calls notebook: https://github.com/tahmid-s99/Applied-DataScience-Project/blob/main/Data_Collection_API.ipynb

SpaceX API request for launch datasets

Filtering data for only Falcon 9 launches

Preprocessing and missing values

Data Collection - Scraping

- Github URL of web scraping for launch data: https://github.com/tahmid-s99/Applied-DataScience-Project/blob/main/Data_Collection_Web_Scraping.ipynb

Request for Falcon 9 launch wiki page

Extracting columns/variables from HTML tags

DataFrame using launch HTML tables

Data Wrangling

Github URL for data wrangling: https://github.com/tahmid-s99/Applied-DataScience-Project/blob/main/Data_Wrangling.ipynb

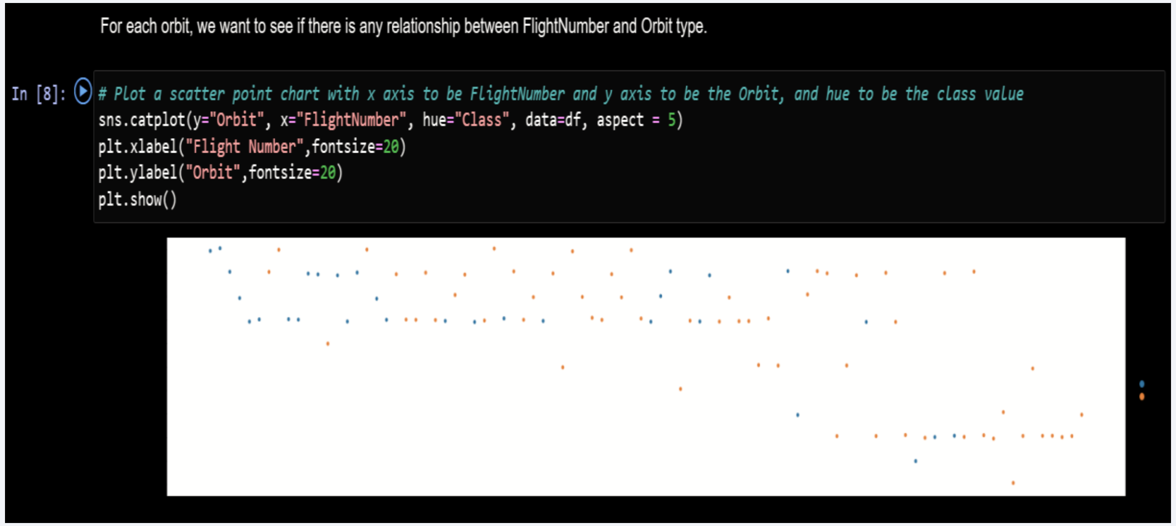
EDA

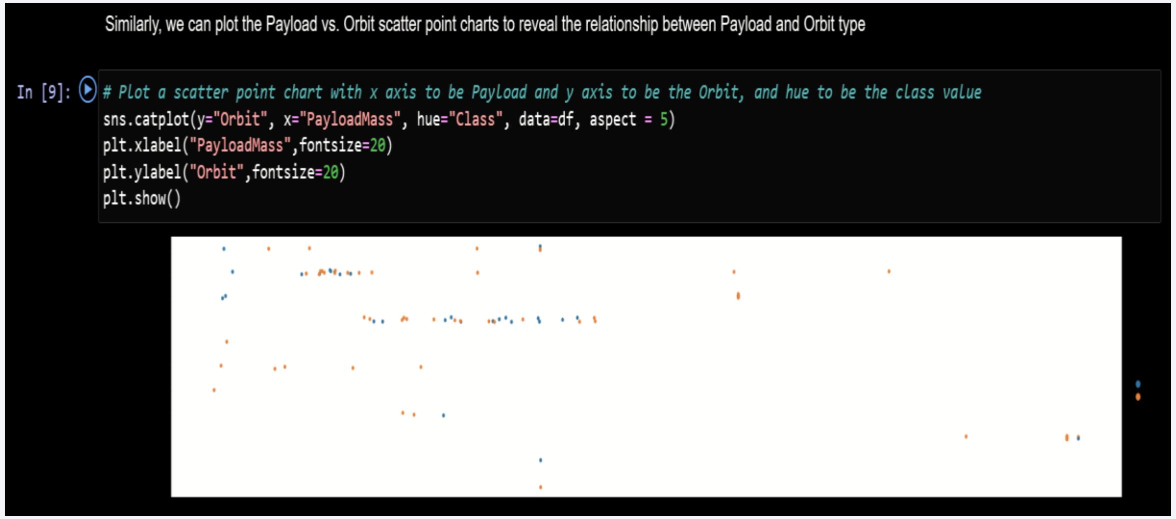
Summary of data findings

Landmarks outcome labels

EDA with Data Visualization

- Visualization of different pairs of features were done
- Github URL of EDA for data visualization: https://github.com/tahmid-s99/Applied-DataScience-Project/blob/main/EDA_with_Visualization.ipynb





EDA with SQL

- Following queries were performed:
- Name of unique launch sites in mission
- Top 5 launch sites with ‘CCA’ in their names
- Average payload mass carried by boosters
- First successful landing outcome
- Names of boosters having success in drone ship and payload mass between 4000 and 6000 kg
- Total successful and failure missions
- Name of boosters having carried maximum payload mass
- Failed landing outcomes
- Rank of count of landing outcomes
- Github URL of EDA with SQL: https://github.com/tahmid-s99/Applied-DataScience-Project/blob/main/EDA_with_SQL.ipynb

Build an Interactive Map with Folium

- Markers, circles marker clusters and lines were used in Folium maps:

- Markers indicated launch sites
- Marker clusters indicated groups of events in different coordinates
- Lines were used in indicating distance between coordinates
- Circles indicated highlighted areas around coordinates

Github URL of Folium visualization: https://github.com/tahmid-s99/Applied-DataScience-Project/blob/main/Folium_Visualization.ipynb

Build a Dashboard with Plotly Dash

- Percentage of launches by site and payload range was usd in Plotly Dashboard
- We can quickly find relationships between payloads and launch sites
- Github URL of Plotly Dash Dashboard: https://github.com/tahmid-s99/Applied-DataScience-Project/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- Different classification models were performed: Logistic regression, SVM, Decision Tree, KNN
- Github URL of ML Predictive Analysis: https://github.com/tahmid-s99/Applied-DataScience-Project/blob/main/SpaceX_ML_Prediction.ipynb

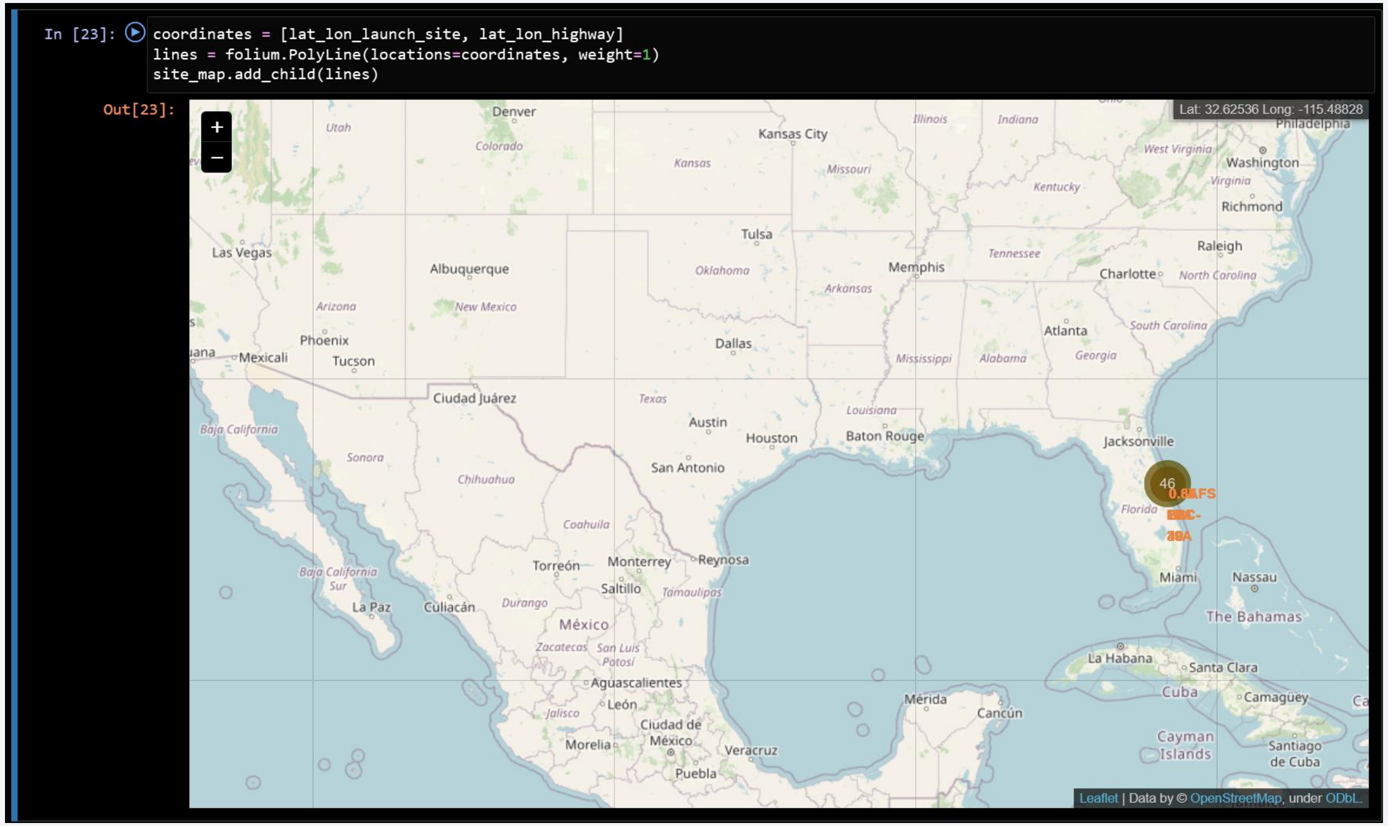
Data preprocessing and standardization

Testing of model combinations and hyperparameters

Results and findings

Results

- Utilizing interactive analytics made it possible to identify launch sites used for satefy, near sea and other locations



Section 2

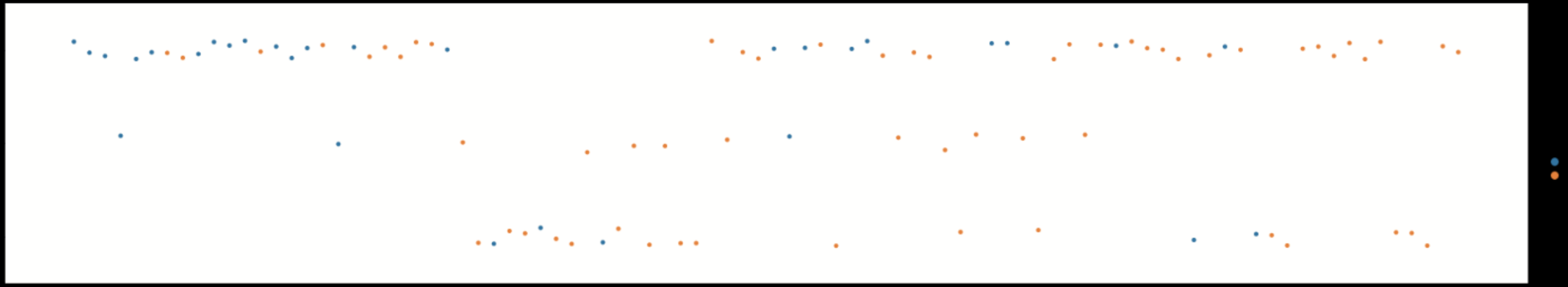
Insights drawn from EDA

Flight Number vs. Launch Site

- The best launch site seems to be CCAFS SLC 40
- The general success rate of launches improved over time

Use the function `catplot` to plot `FlightNumber` vs `LaunchSite`, set the parameter `x` parameter to `FlightNumber`,set the `y` to `Launch Site` and set the parameter `hue` to `'class'`

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

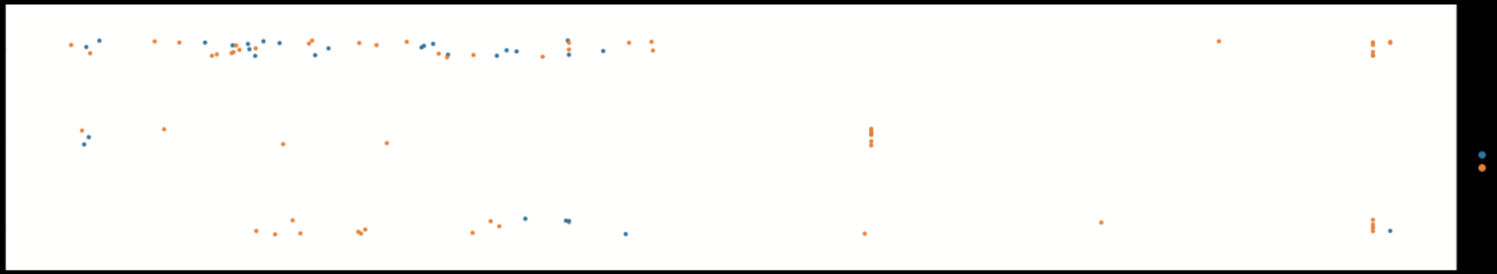


Payload vs. Launch Site

- Payloads over 9000 kg have good success rates whereas payloads over 12000 kg are only possible on the launch sites CCAFS SLC 40 and KSC LC 39A

We also want to observe if there is any relationship between launch sites and their payload mass.

In [6]:
```
# 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 value
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("Pay load Mass (kg)",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



Now if you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

Success Rate vs. Orbit Type

- The biggest success rates occurs for:

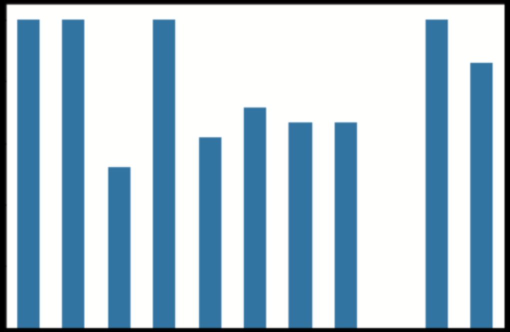
- ES-L1
- GEO
- HEO
- SSO

Next, we want to visually check if there are any relationship between success rate and orbit type.

Let's create a `bar chart` for the sucess rate of each orbit

In [7]:
```
# HINT use groupby method on Orbit column and get the mean of Class column
pl = df.groupby('Orbit')['Class'].mean()
ax = pl.plot(kind='bar')
ax.set_xlabel("Orbit")
ax.set_ylabel("Sucess Rate of each Orbit")
```

Out[7]:
```
Text(0, 0.5, 'Sucess Rate of each Orbit')
```



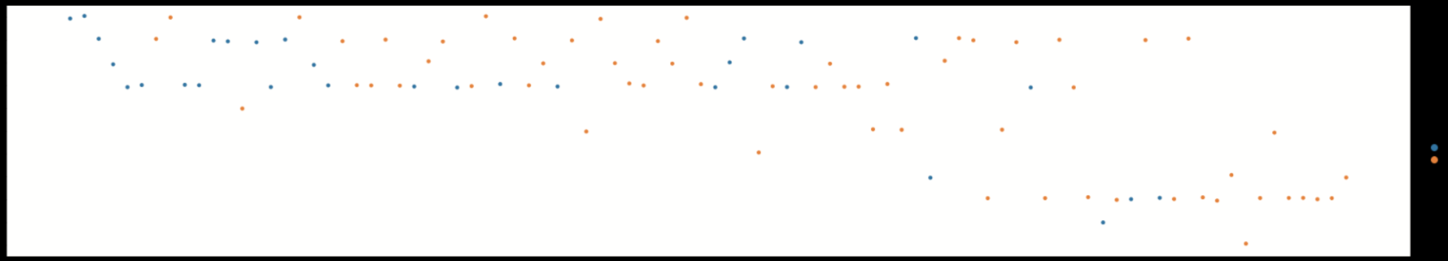
Analyze the ploted bar chart try to find which orbits have high sucess rate.

Flight Number vs. Orbit Type

- Success rate improves over time for all orbits
- VLEO orbit increases in its frequency for successes

For each orbit, we want to see if there is any relationship between FlightNumber and Orbit type.

In [8]:
```
# 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("Flight Number",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



You should see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

Payload vs. Orbit Type

- No defined relationship between payload and orbit type

Similarly, we can plot the Payload vs. Orbit scatter point charts to reveal the relationship between Payload and Orbit type

In [9]:
```
# 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()
```

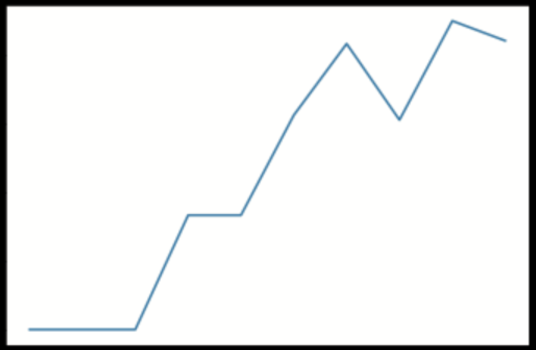
With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

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

- Success rate over time shows an increasing trend from 2013 until 2020

```
In [12]: # Plot a line chart with x axis to be the extracted year and y axis to be the success rate
df['Year'] = Extract_year(df["Date"])
df_groupby_year = df.groupby("Year",as_index=False)["Class"].mean()
sns.lineplot(data = df_groupby_year, x="Year", y="Class")
plt.xlabel("Year")
plt.title('Space X Rocket Success Rate')
plt.ylabel("Success Rate")
plt.show()
```



you can observe that the sucess rate since 2013 kept increasing till 2020

All Launch Site Names

- There are mainly 4 below launch sites

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

In [10]:
```
%sql select distinct "LAUNCH_SITE" from SPACEXTABLE
```

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

Out[10]:

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

Launch Site Names Begin with 'CCA'

- Below 5 launch sites start with ‘CCA’

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

In [11]:
```
%sql select * from SPACEXTABLE where "LAUNCH_SITE" like '%CCA%' limit 5
```

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

Out[11]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	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)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- The total sum of payload is 111.268 kg.

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

```
In [33]: %sql select sum(PAYLOAD_MASS__KG_) AS sum_payload from SPACEXTABLE where PAYLOAD like '%CRS%'

 * sqlite:///applied_ds_data.db
Done.

Out[33]:
```

sum_payload
111268

Average Payload Mass by F9 v1.1

- The average payload mass carried by booster version F9 v1.1 is below.

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

```
In [13]: %sql select avg("PAYLOAD_MASS__KG_") from SPACEXTABLE where "BOOSTER_VERSION" like '%F9 v1.1%'

 * sqlite:///applied_ds_data.db
Done.

Out[13]:
```

avg("PAYLOAD_MASS__KG_")
2534.6666666666665

First Successful Ground Landing Date

- The first successful ground landing date is shown below:

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

Hint:Use min function

```
In [17]: %sql select min("DATE") as min_date from SPACEXTABLE where "Landing _Outcome" like '%Success%'

 * sqlite:///applied_ds_data.db
Done.

Out[17]:
```

min_date
01-05-2017

Successful Drone Ship Landing with Payload between 4000 and 6000

- All names of boosters which have successfully landed on drone ship and had 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 [22]: SION" from SPACEXTABLE where "LANDING _OUTCOME" = 'Success (drone ship)' and "PAYLOAD_MASS__KG_" > 4000 and "PAYLOAD_MASS__KG_" < 6000;
```

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

Out[22]:

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

Total Number of Successful and Failure Mission Outcomes

- Total number of successful and failure mission outcomes

List the total number of successful and failure mission outcomes

```
In [34]: %sql select (select count("MISSION_OUTCOME") from SPACEXTABLE where "MISSION_OUTCOME" like "%Success%") as success, (select count("MISS
```

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

Out[34]:

success	failure
100	1

Boosters Carried Maximum Payload

- Boosters carrying the maximum payload mass:

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

In [24]:
```
%sql select distinct "BOOSTER_VERSION" from SPACEXTABLE where "PAYLOAD_MASS__KG_" = (select max("PAYLOAD_MASS__KG_") from SPACEXTABLE)
```

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

Out[24]:

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

- Failed landing outcomes in drone ship in the year 2015 (there are two occurrences):

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

```
In [26]: h, "BOOSTER_VERSION", "LAUNCH_SITE" from SPACEXTABLE where "LANDING _OUTCOME" = 'Failure (drone ship)' and substr("DATE", 7, 4) = "2015'
```

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

Out[26]:

month	Booster_Version	Launch_Site
01	F9 v1.1 B1012	CCAFS LC-40
04	F9 v1.1 B1015	CCAFS LC-40

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

- Ranking the count of successes and 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

In [27]:
```
and "DATE" <= "20-03-2017" and "LANDING _OUTCOME" like "%Success%" group by "LANDING _OUTCOME" order by count("LANDING _OUTCOME") desc
```

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

Out[27]:

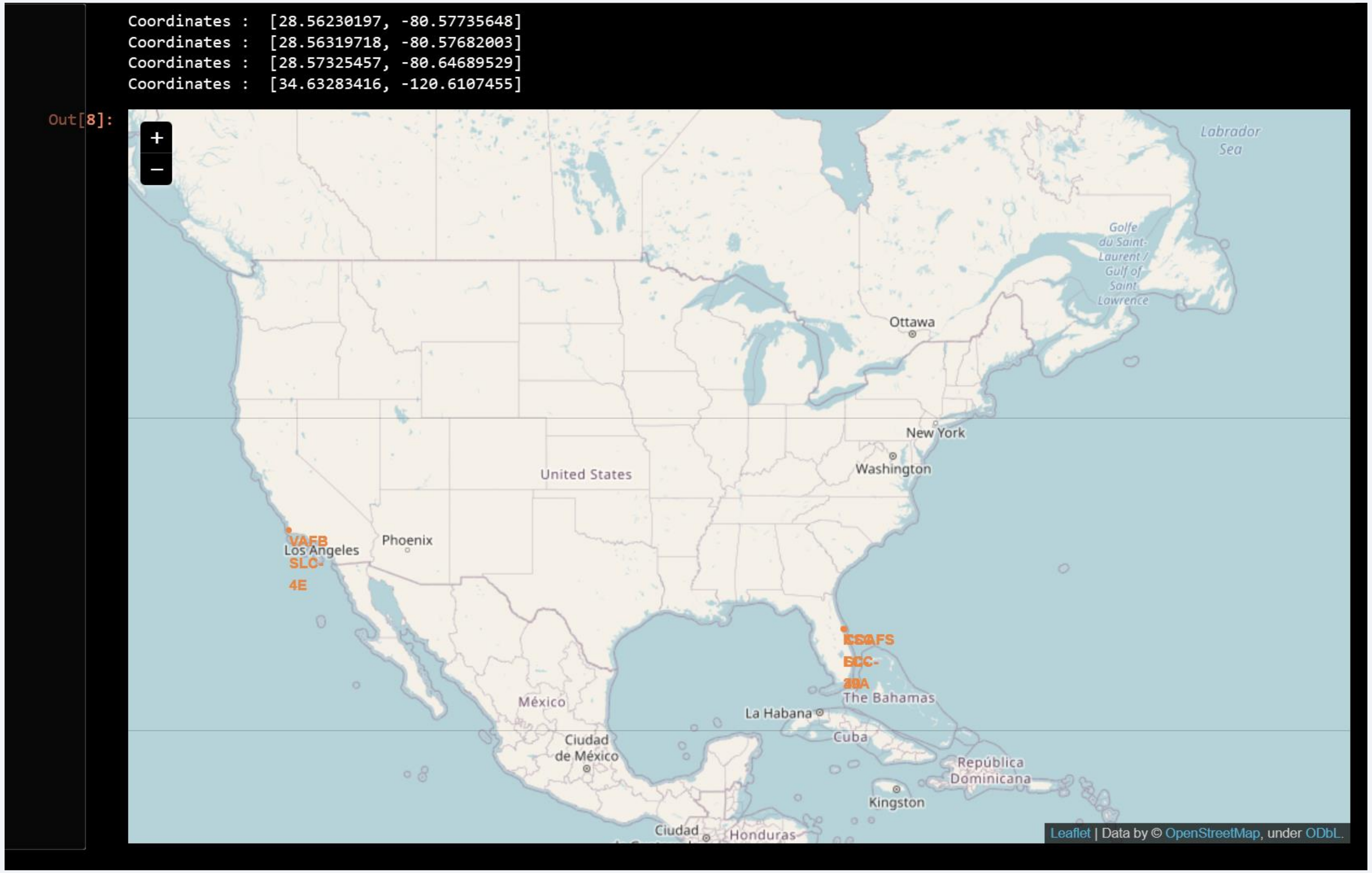
Landing _Outcome	count("LANDING _OUTCOME")
Success	20
Success (drone ship)	8
Success (ground pad)	6

Section 3

Launch Sites Proximities Analysis

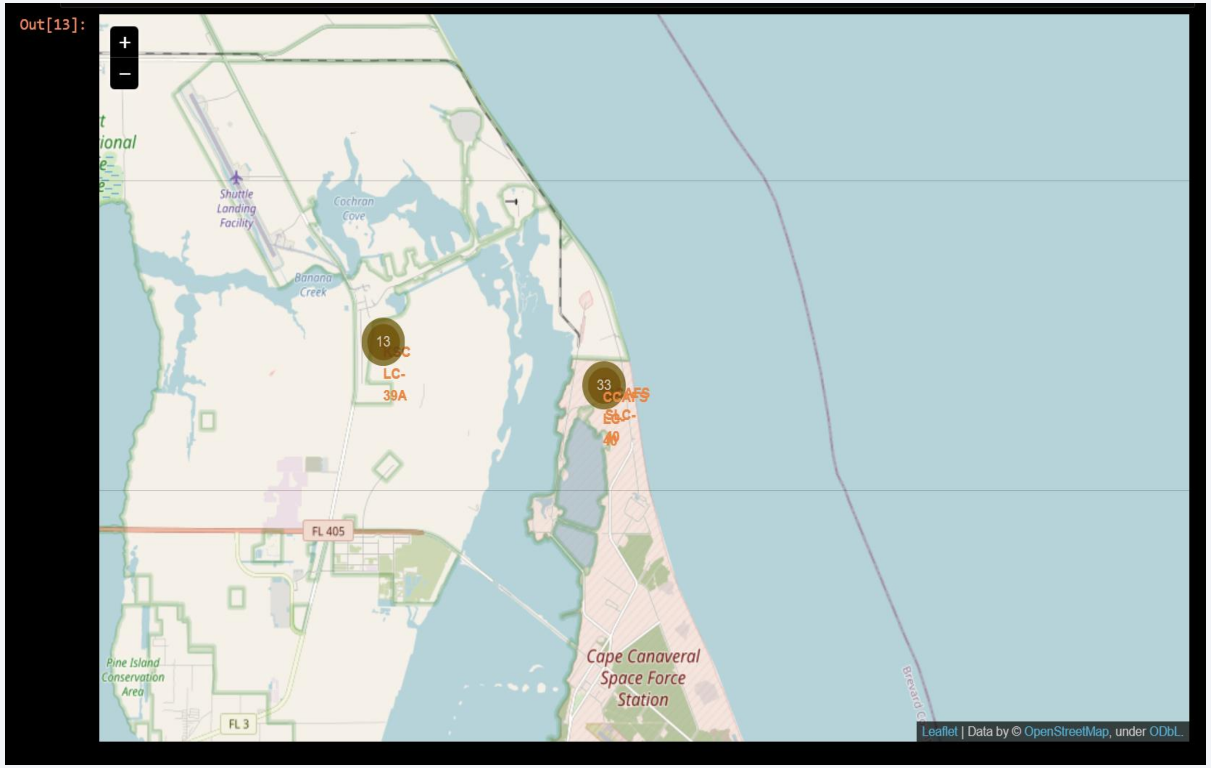
Map of all launch sites

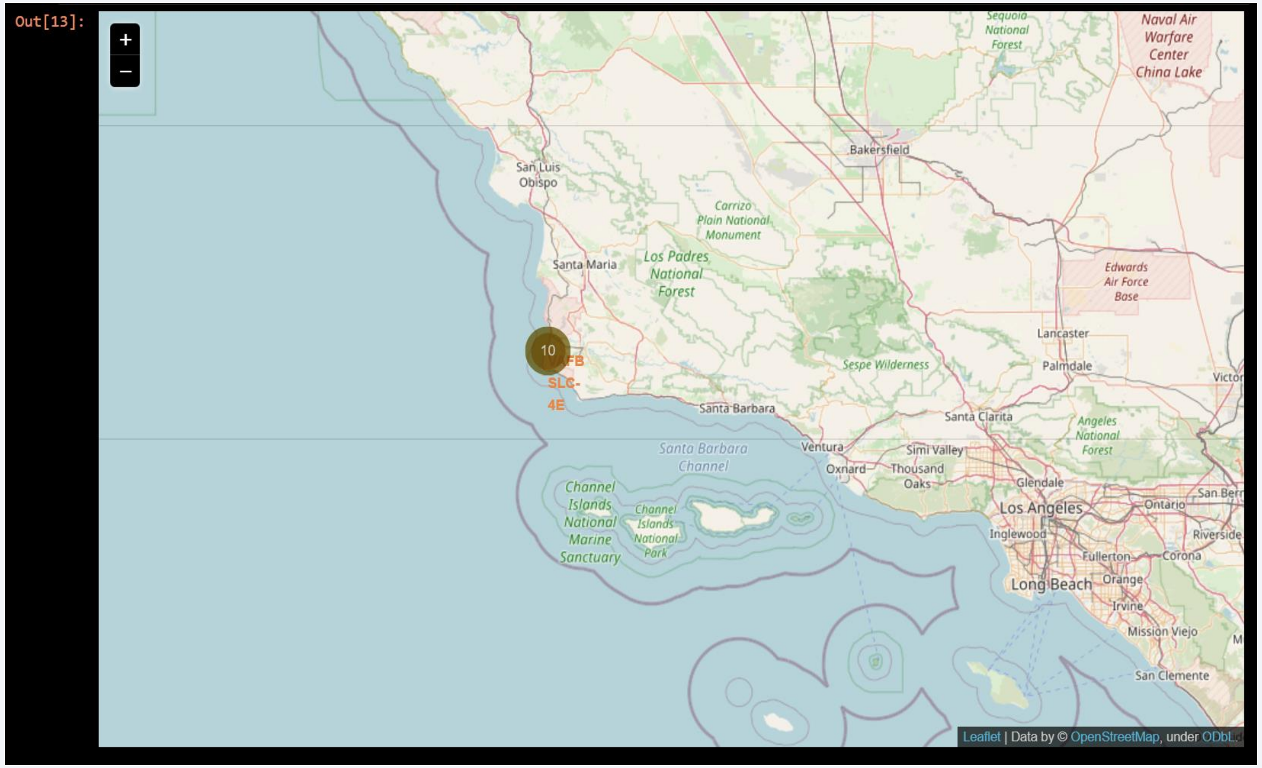
- All launch sites near sea, away from main land areas:



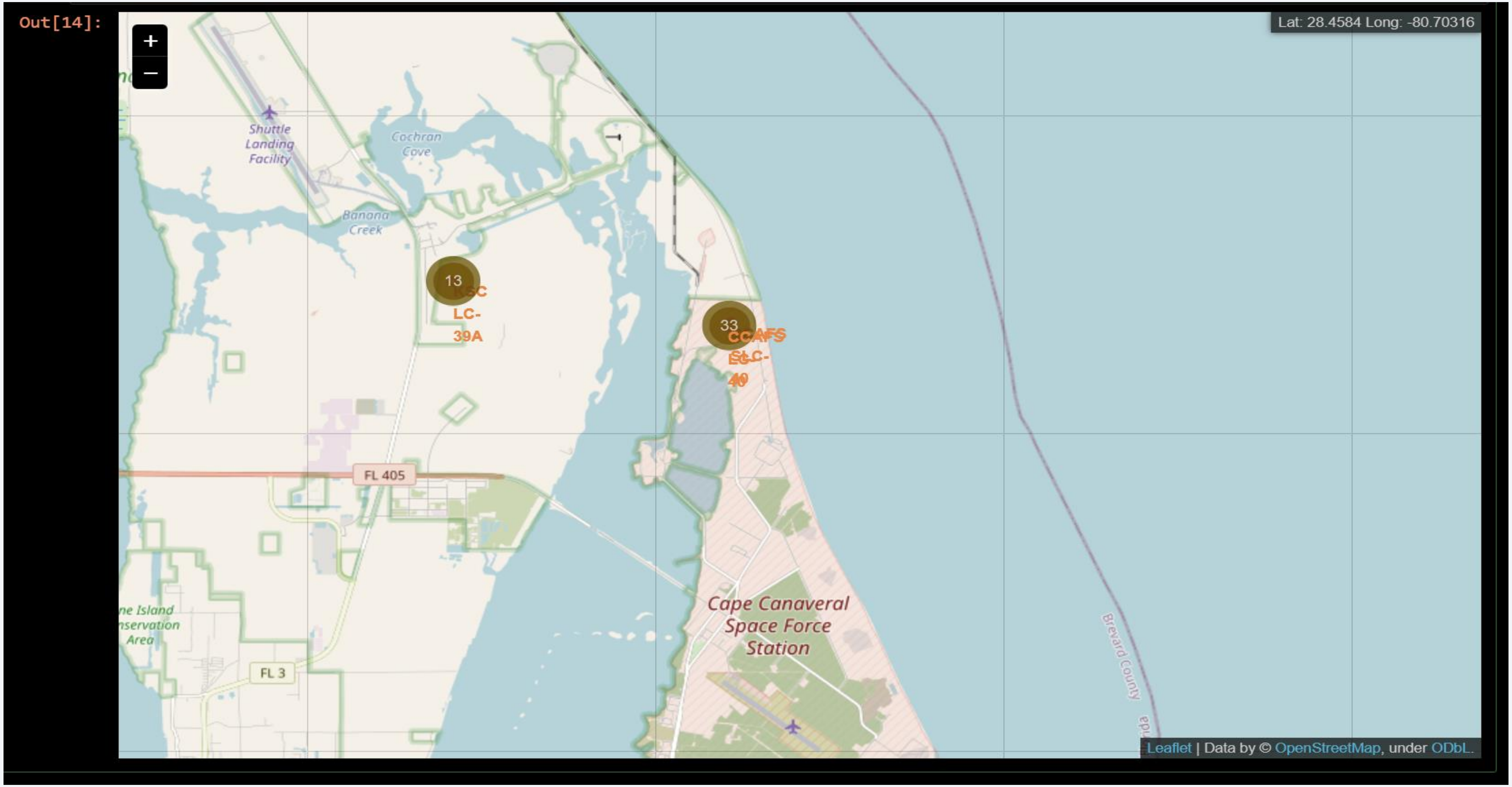
Site launch outcomes

- Displaying site launch outcomes:





Proximity of launch sites



Section 4

Build a Dashboard with Plotly Dash

Dashboard results

- Unable to generate dashboard results, please find code in Appendix below

Dashboard results

- Unable to generate dashboard results, please find code in Appendix below

Dashboard results

- Unable to generate dashboard results, please find code in Appendix below

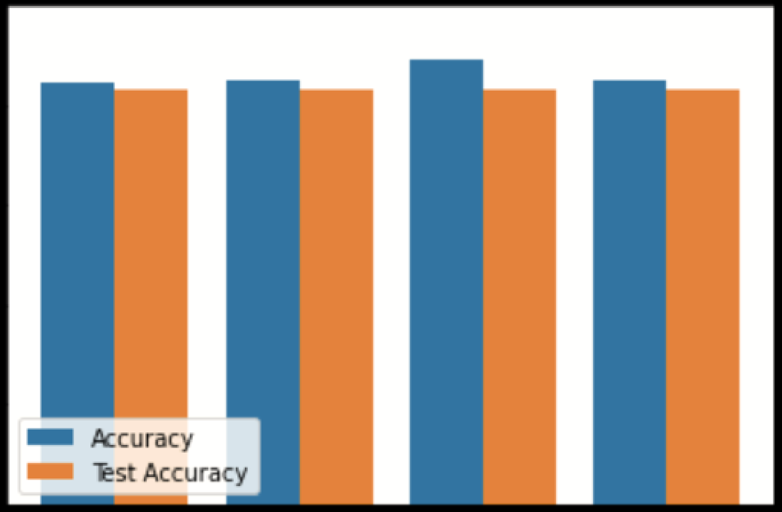
Section 5

Predictive Analysis (Classification)

Classification Accuracy

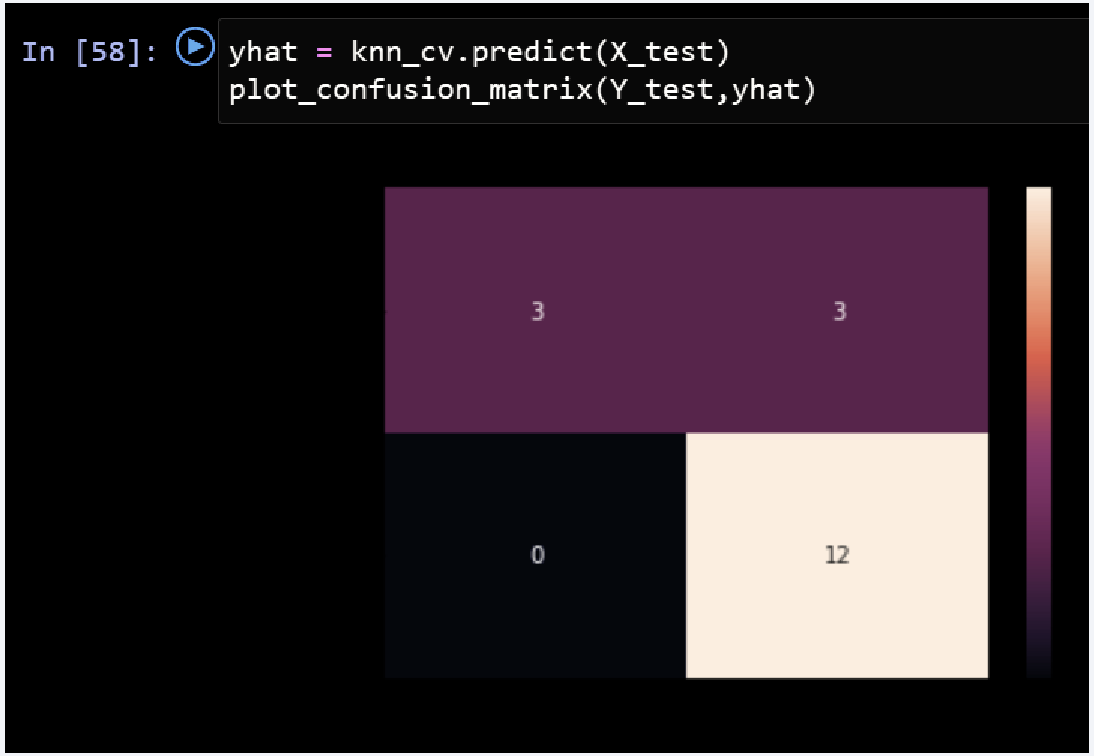
- Different models were evaluated, and their accuracies are shown on the right. The accuracy of a decision tree performs the best.

Model	Accuracy	TestAccuracy
LogReg	0.84643	0.83333
SVM	0.84821	0.83333
Tree	0.88929	0.83333
KNN	0.84821	0.83333



Confusion Matrix

- Confusion matrix of decision tree



Conclusions

- Different sources of data were analyzed, which helped support model develop and discovering conclusions about the solution and problem.
- The decision tree classifier predicted successful landings and can be used to increase revenue
- Successful landing improves over time, along with the technology of launches and rockets

Appendix

```python
# Import required libraries
import pandas as pd
import dash
from dash import html
from dash import dcc
from dash.dependencies import Input, Output
import plotly.express as px

# Read the airline data into pandas dataframe
spacex_df = pd.read_csv("spacex_dash.csv")
max_payload = spacex_df['Payload Mass (kg)'].max()
min_payload = spacex_df['Payload Mass (kg)'].min()

#print(spacex_df['Launch Site'].unique())

# Create a dash application
app = dash.Dash(__name__)

# Create an app layout
app.layout = html.Div(children=[html.H1('SpaceX Launch Records Dashboard',
                                        style={'textAlign': 'center', 'color': '#503D36',
                                               'font-size': 40}),
                                # TASK 1: Add a dropdown list to enable Launch Site selection
                                # The default select value is for ALL sites
                                dcc.Dropdown(id='site-dropdown',
                                             options=[
                                                 {'label': 'All Sites', 'value': 'ALL'},
                                                 {'label': 'CCAFS LC-40', 'value': 'CCAFS LC-40'},
                                                 {'label': 'VAFB SLC-4E', 'value': 'VAFB SLC-4E'},
                                                 {'label': 'KSC LC-39A', 'value': 'KSC LC-39A'},
                                                 {'label': 'CCAFS SLC-40', 'value': 'CCAFS SLC-40'}],
                                             value='ALL',
                                             placeholder="State",
                                             searchable=True
                                            ),
                                html.Br(),

                                # TASK 2: Add a pie chart to show the total successful launches count for all sites
                                # If a specific launch site was selected, show the Success vs. Failed counts for the site
                                html.Div(dcc.Graph(id='success-pie-chart')),
                                html.Br(),

                                html.P("Payload range (Kg):"),
                                # TASK 3: Add a slider to select payload range
                                dcc.RangeSlider(id='payload-slider',
                                                min=0, max=10000, step=1000,
                                                marks={0: '0', 100: '100'},
                                                value=[min_payload, max_payload]),

                                # TASK 4: Add a scatter chart to show the correlation between payload and launch success
                                html.Div(dcc.Graph(id='success-payload-scatter-chart')),
                                ])

# TASK 2:
# Add a callback function for `site-dropdown` as input, `success-pie-chart` as output
# Function decorator to specify function input and output
@app.callback(Output(component_id='success-pie-chart', component_property='figure'),
              Input(component_id='site-dropdown', component_property='value'))
def get_pie_chart(entered_site):
    if entered_site == 'ALL':
        fig = px.pie(spacex_df,
                     values='class',
                     names='Launch Site',
                     title='Total Success Launches By Site')
    else:
        filtered_df = spacex_df[spacex_df['Launch Site'] == entered_site]
        filtered_df = filtered_df.groupby('class').count().reset_index()
        fig = px.pie(filtered_df,
                     values='Unnamed: 0',
                     names='class',
                     title='Total Launches for site {}'.format(entered_site))
        # return the outcomes piechart for a selected site
    return fig

# TASK 4:
# Add a callback function for `site-dropdown` and `payload-slider` as inputs, `success-payload-scatter-chart` as output
@app.callback(Output(component_id='success-payload-scatter-chart', component_property='figure'),
              Input(component_id='site-dropdown', component_property='value'),
              Input(component_id="payload-slider", component_property="value"))
def get_scatter_chart(entered_site, payload_range):
    print('Params: {} {}'.format(entered_site, payload_range))
    if entered_site == 'ALL':
        filtered_df = spacex_df[(spacex_df['Payload Mass (kg)'] >= int(payload_range[0])) &
                                (spacex_df['Payload Mass (kg)'] <= int(payload_range[1]))
                                ]
        fig = px.scatter(filtered_df, x='Payload Mass (kg)', y='class', color='Booster Version Category', title='All sites - payload mass between {:8,d}kg and {:8,d}kg'.format(int(payload_range[0]),int(payload_range[1])))
    else:
        filtered_df = spacex_df[(spacex_df['Launch Site'] == entered_site) &
                                (spacex_df['Payload Mass (kg)'] >= int(payload_range[0])) &
                                (spacex_df['Payload Mass (kg)'] <= int(payload_range[1]))
                                ]
        fig = px.scatter(filtered_df, x='Payload Mass (kg)', y='class', color='Booster Version Category', title='Site {} - payload mass between {:8,d}kg and {:8,d}kg'.format(entered_site,int(payload_range[0]),int(payload_range[1])))

    return fig
```

Thank you!