IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Umm E Abiha>
<15/06/2025>

Outline

Executive Summary

- We done data collection from spaceX API and webscraping from wikipedia.
- We done EDA by using visualization tools like seaborne , matplotlib and folium.
- Then we use plotly dash to build interactive charts for non – technical persons like stackholders
- In the end we build optimal model after trying different models and applying gridsearchcv on all models to find optimal parameters for models
- By doing all of these we found more information about data , we build dashboard, and throught gridsearchcv we choose best model with best parameters

Introduction

- Main purpose of this project is to build Machine learning model which can predict whether falcon 9 rocket first stage will successfully landed back on earth or not.
- Through this model we will find which feature is contribute to success landing of first stage, so we can focus on that more in future launches to increase our success rate of landing back of first stage.

Section 1

Methodology

Methodology

- Data collection methodology:
  - Data is collected from two sources one from spaceX API and second using webscrapping from wikipedia.
- Perform data wrangling:
  - We want only falcon 9 launchers so we filtered data.
  - Landing pad columns has some nan values we did'nt change that and other column which nan value we replaced that with mean value of that column .

Methodology

Perform exploratory data analysis (EDA) using visualization and SQL:

- In EDA we check if Orbit feature affect the success or not.
- We found there is relation between flight number and class.
- We found out that orbit geo heo and sso has 100% success rate through bar chart.
- And we also found out that higher the payload mass higher the chances of success.
- Success rate are increases day by day because we are improving after every lauches.

Methodology

- Perform interactive visual analytics using Folium and Plotly Dash:
- Using folium we mark launches site in map from this we found that all launches site is build near costline.
- Then we make cluster of marker for each site to see number of success and failure.
- From plotly we build dashboard for interactive charts and graph , we build pie chart and scatter plot which take two input site name and payload mass ranges.

Methodology

- Perform predictive analysis using classification models:
  - We use four classification models Linear regression, Support vector classifier, Decision trees and K nearest neighbors.
  - Then we perform gridsearchcv on all models to find optimal parameters for them.
  - In the end we choose that model which returns highest roc_auc score.

-

Data Collection

Data is collected from two sources one from spaceX API and second using webscrapping from wikipedia. For webscrapping we used beautyfull soup.

Data Collection – SpaceX API

- GitHub URL of the completed SpaceX API calls notebook:
- https://github.com/Abiha-source/Final-Capston-Project/blob/main/SpaceX_API.ipynb

➔ Import neccesary libraries.
➔ Creating function to make data collection easy from API.
➔ Then use request module to send request to API.
➔ Convert response data to pandas dataframe.
➔ We initiate empty lists and append this lists by making api calls using helper function.
➔ Then we replace nan values of column payload mass with its mean value.
➔ We also had nan value in LandingPad but did'nt change that.
➔ We filtered data to only have launcher with falcon 7 .

Data Collection - Scraping

- GitHub URL of the completed web scraping notebook:
- https://github.com/Abiha-source/Final-Capston-Project/blob/main/webscraping.ipynb

➔ Import neccesary libraries.
➔ Understaning helper function, this function helps to clean our response data.
➔ Then use request module to send request to wikipedia.
➔ Used BeautifulSoup library parse html content of wikipedia.
➔ We use find_all method of BeautifulSoup and use loops and helper function to extract columns names
➔ Then we initiate empty lists then again we use find_all method and loops and helper function to extract rows from wikipedia html content and append this rows in lists.
➔ And in last we create dataframe from this lists.

Data Wrangling

The GitHub URL of completed data wrangling related notebooks: https://github.com/Abiha-source/Final-Capston-Project/blob/main/Data_wrangling.ipynb

- Import neccesary libraries.
- Then we check dtypes of all columns
- Then we check how many rockets are launch by each site.
- Then we change landing outcomes to categories of 0's and 1's and stored it to class column.
- Then in last we check how many rockets are launch to each orbit.

EDA with Data Visualization

the GitHub URL of completed EDA with data visualization notebook:

https://github.com/Abiha-source/Final-Capston-Project/blob/main/EDA_VIZ.ipynb

- Import neccesary libraries.
- For vizualization we used matplotlib and seaborne
- First we plot cat plot of flight number and payload mass
- Then we plot cat plot of flight number and launch site
- Then payload mass and launch site
- Then we plot bar chart of orbit to check which orbit has higher chance of success
- In task 4 we plot cat plot of orbit and launch site for task 5 we plot payload mass and orbit type.
- Then we plot line plot over a time with success rate , we noticed that success is increasing over a time.
- In last we use get_duumies function to one hot code our categorical column and in end we change type of this column to float64 using astype method.

EDA with SQL

the GitHub URL of completed EDA with SQL notebook:

https://github.com/Abiha-source/Final-Capston-Project/blob/main/EDA_SQL.ipynb

- Import necessary libraries.
- Load necessary extentions like sql.
- SELECT DISTINCT(Landing_Outcome) FROM SPACEXTBL for unique landing outcome.
- sql SELECT * FROM SPACEXTBL WHERE "Launch_Site" LIKE "CCA%" to select only those rows where launch site starts with CCA.
- SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE "Customer" = "NASA (CRS)" for total payload mass where customer is NASA (CRS.)
- SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE "Booster_Version" = "F9 v1.1" for average of payload mass where booster version is F9 v1.1
- SELECT MIN(Date) FROM SPACEXTBL WHERE "Landing_Outcome" = "Success (ground pad)" selecting first date when first stage successfully landed on ground pad.
- SELECT DISTINCT(Booster_Version) FROM SPACEXTBL WHERE "Landing_Outcome" = "Success (drone ship)" AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000 selecting unique booster version landing outcome is success on drone ship and payload mass is between 4000 to 6000
- SELECT COUNT(*) FROM SPACEXTBL WHERE "Landing_Outcome" LIKE "Success%" counting how many landing is successed
- SELECT DISTINCT(Booster_Version) FROM SPACEXTBL WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL) selecting only those booster version where payload mass is maximum.

The GitHub URL of completed interactive map with Folium map:

https://github.com/Abiha-source/Final-Capston-Project/blob/main/Folium.ipynb

- ➔ Import necessary libraries.
- ➔ We use folium library to plot map.
- ➔ Then we select important column and done groupby on launch site.
- ➔ Then we initiate folium marker and circle object then add this object to our map. To see where our lauch site are in map.
- ➔ Then we build marker cluster to see number of success and failure in each site.
- ➔ In last we helper function and mouseposition object to draw poly line to see how far is launch site to its nearest coastline.

the GitHub URL of completed Plotly Dash lab:

https://github.com/Abiha-source/Final-Capston-Project/blob/main/my_dash.py

- Import necessary libraries
- First we add dropdown with all unique option of launch site.
- Then I add piechart with callback function by using this dropdown and I also filters data as per selected value in dropdown.
- Then I add range slider for user to choose range of payload mass for scatter plot and then I use another callback function for this scatter plot with two input first input is dropdown and second input is range slider and in last I filtered data as per both input

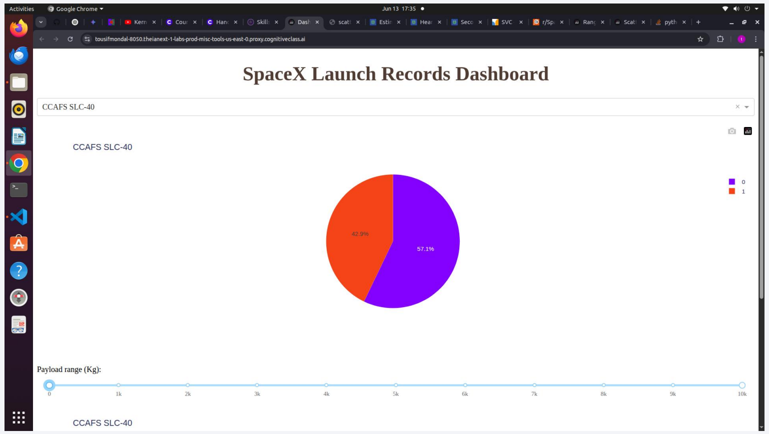
the GitHub URL of completed predictive analysis lab:

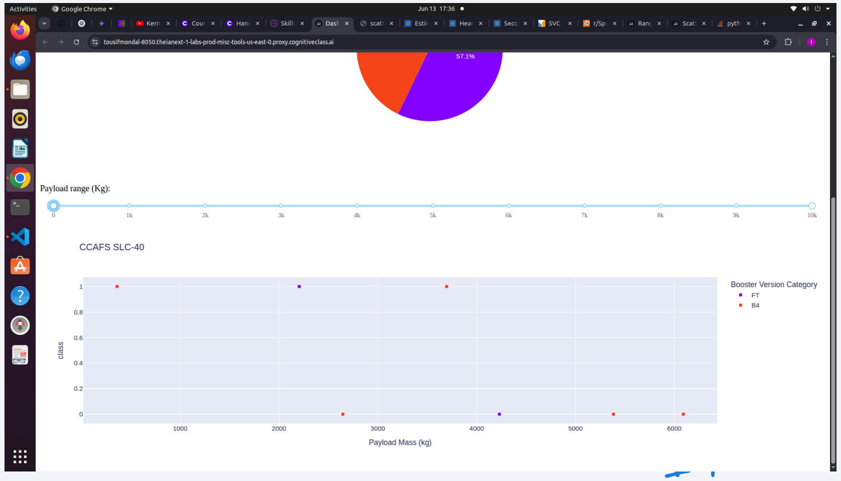
https://github.com/Abiha-source/Final-Capston-Project/blob/main/SpaceX_Machine%20Learning%20Prediction.ipynb

- We use four classification models Linear regression, Support vector classifier, Decision trees and K nearest neighbors.
- Then we perform gridsearchcv on all models to find optimal parameters for them.
- In the end we choose that model which returns highest roc_auc score.

Results

- Exploratory data analysis results
- through EDA we get to know about relation between two features.
- We also done one hot encodeing and create a new column named class which has 0's and 1's which tell about success and failure.
- Interactive analytics demo in screenshots
- In right side we have screenshot of dash application.
- Predictive analysis results
- we found out that almost all model gives same accuracy so we choose that model which have maximum roc_auc score.





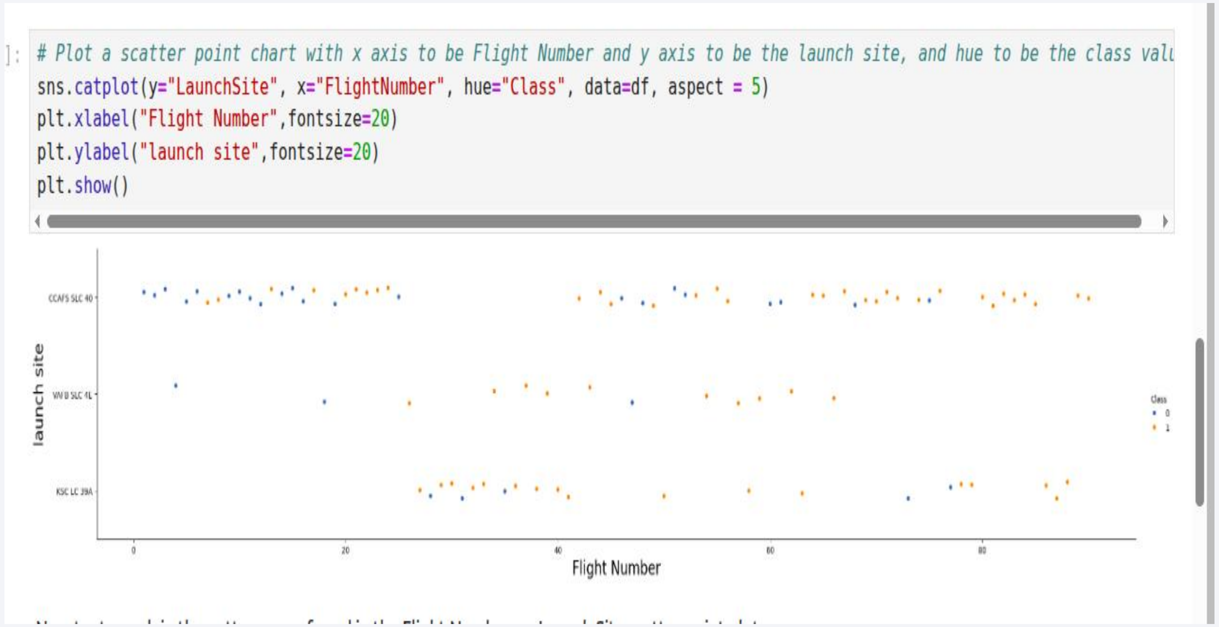
Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

- Show a scatter plot of Flight Number vs. Launch Site

- From this we can say that different site has different number of launches some more or some has site has very less.

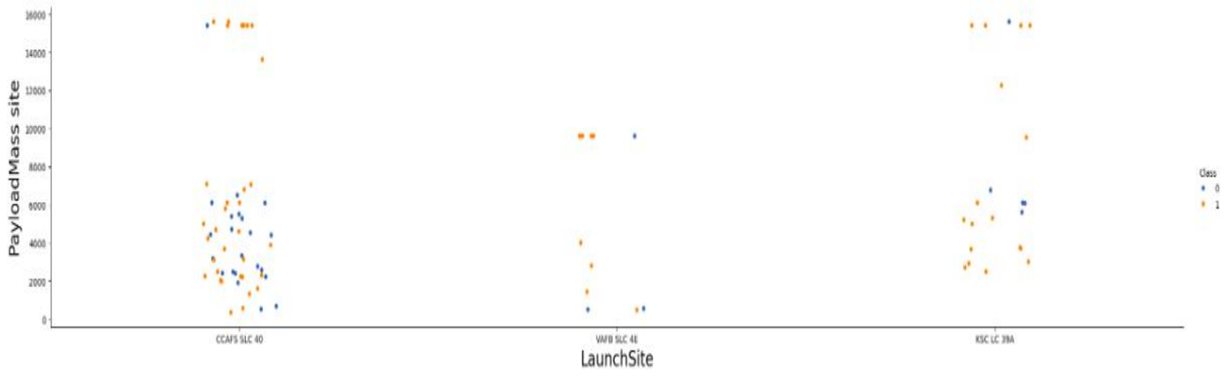


```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class valu
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("launch site",fontsize=20)
plt.show()
```

Payload vs. Launch Site

- Show a scatter plot of Payload vs. Launch Site
- From this we can say that higher payload mass launches has higher chances of success despite over Which Site launches rocket.

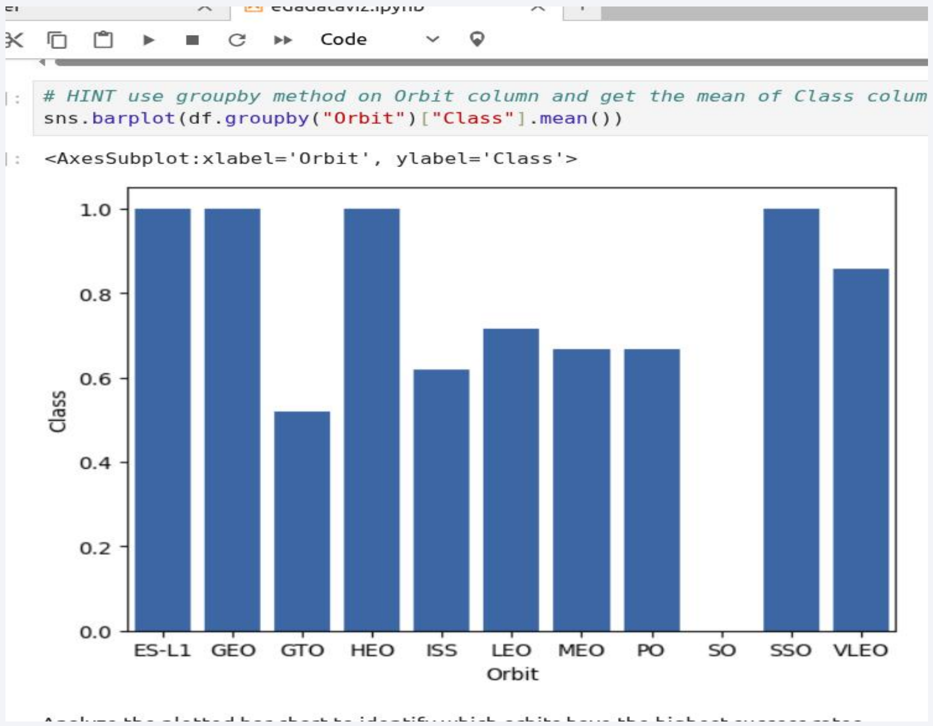
```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the class
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class valu
sns.catplot(y="PayloadMass", x="LaunchSite", hue="Class",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



Success Rate vs. Orbit Type

- Show a bar chart for the success rate of each orbit type

- From this we can say that orbit significantly increase the chances of success for example ES-L1 , GEO, HEO and SSO has 100 % of success.



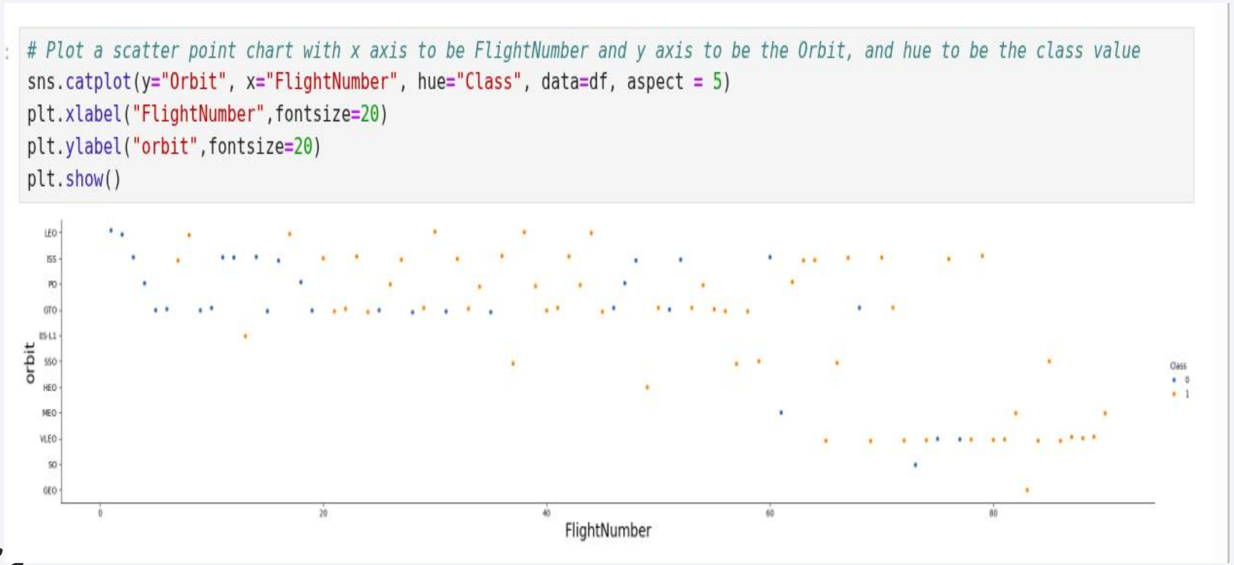
```
# HINT use groupby method on Orbit column and get the mean of Class colum
sns.barplot(df.groupby("Orbit")["Class"].mean())
```

```
<AxesSubplot:xlabel='Orbit', ylabel='Class'>
```

Flight Number vs. Orbit Type

- Show a scatter point of Flight number vs. Orbit type

- We can say that chances of success is increases for all orbit level except GTO.



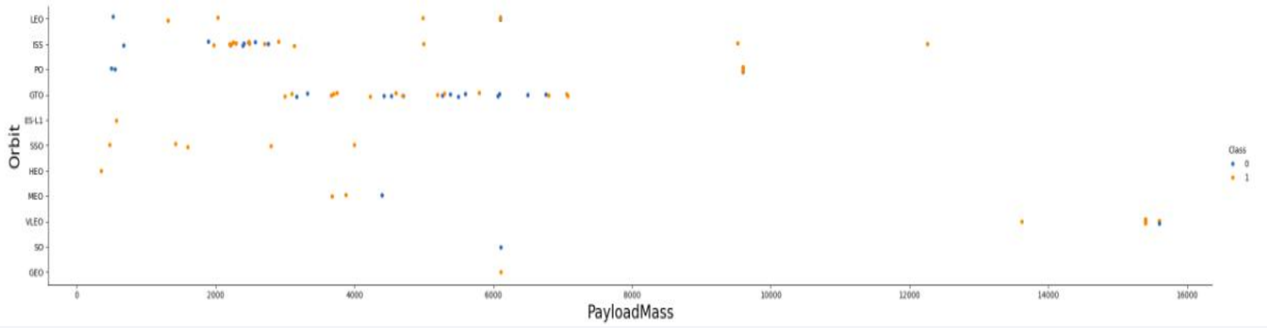
```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("FlightNumber",fontsize=20)
plt.ylabel("orbit",fontsize=20)
plt.show()
```

Payload vs. Orbit Type

- Show a scatter point of payload vs. orbit type

- From this we can say that higher payload mass launches has higher chances of success despite over Which orbit rocket goes to.

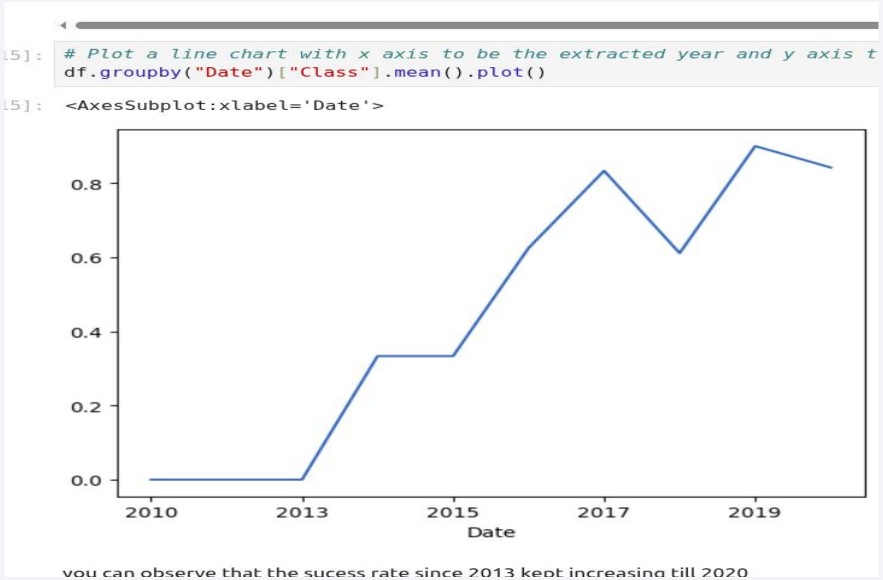
```
# Plot a scatter point chart with x axis to be Payload Mass and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("PayloadMass",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



Launch Success Yearly Trend

- Show a line chart of yearly average success rate

- After seeing this plot we can that spaceX keep learning from its previous mistakes.



```
15]: # Plot a line chart with x axis to be the extracted year and y axis t
     df.groupby("Date")["Class"].mean().plot()

15]: <AxesSubplot:xlabel='Date'>
```

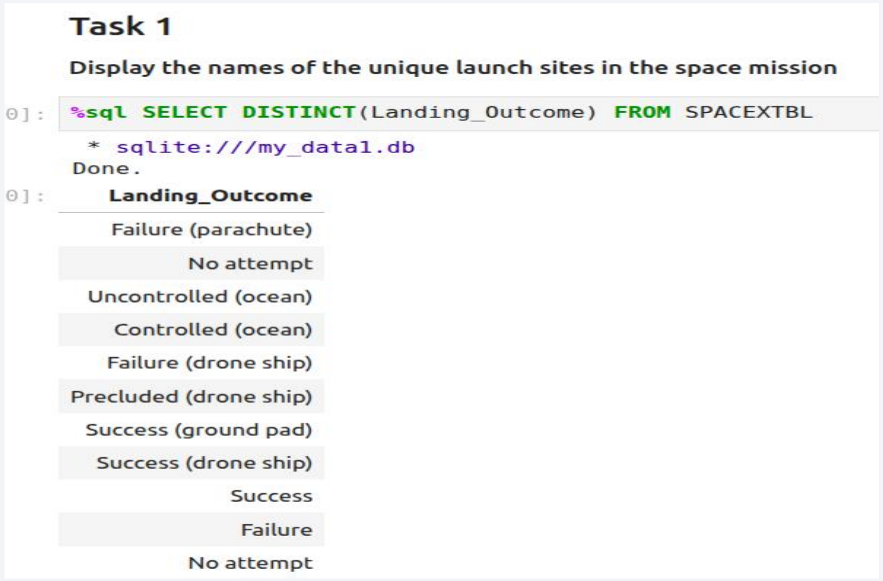
you can observe that the sucess rate since 2013 kept increasing till 2020

All Launch Site Names

- Find the names of the unique launch sites.

- SELECT DISTINCT(Landing_Outcome) FROM SPACEXTBL.

  This query returns unique landing outcome.



Task 1

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

```
0]: %sql SELECT DISTINCT(Landing_Outcome) FROM SPACEXTBL
 * sqlite:///my_data1.db
Done.
0]:
```

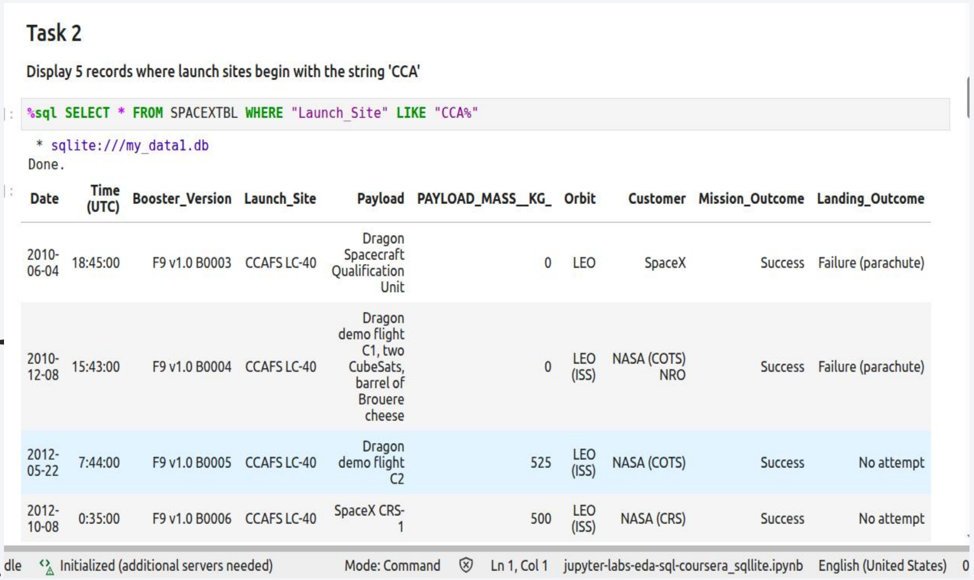
Landing_Outcome
Failure (parachute)
No attempt
Uncontrolled (ocean)
Controlled (ocean)
Failure (drone ship)
Precluded (drone ship)
Success (ground pad)
Success (drone ship)
Success
Failure
No attempt

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`.

- SELECT * FROM SPACEXTBL WHERE "Launch_Site" LIKE "CCA%".

  This query returns all records where launch site starts with CAA.



Task 2

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

```
%sql SELECT * FROM SPACEXTBL WHERE "Launch_Site" LIKE "CCA%"
```

```
 * 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

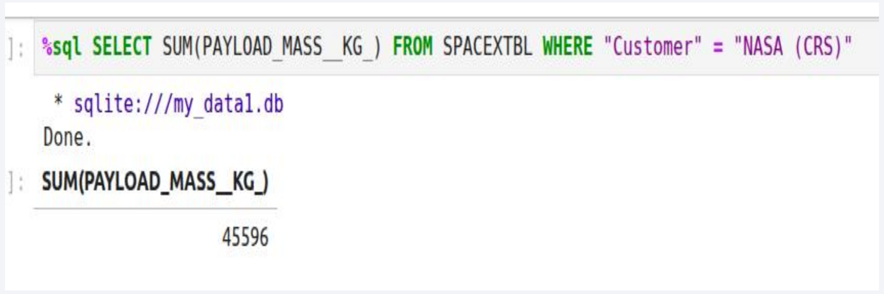
dle
Initialized (additional servers needed)
Mode: Command
Ln 1, Col 1
jupyter-labs-eda-sql-coursera_sqllite.ipynb
English (United States)
0

Total Payload Mass

- Calculate the total payload carried by boosters from NASA.

- SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE "Customer" = "NASA (CRS)".

  This query returns total payload mass where customer is NASA (CRS).



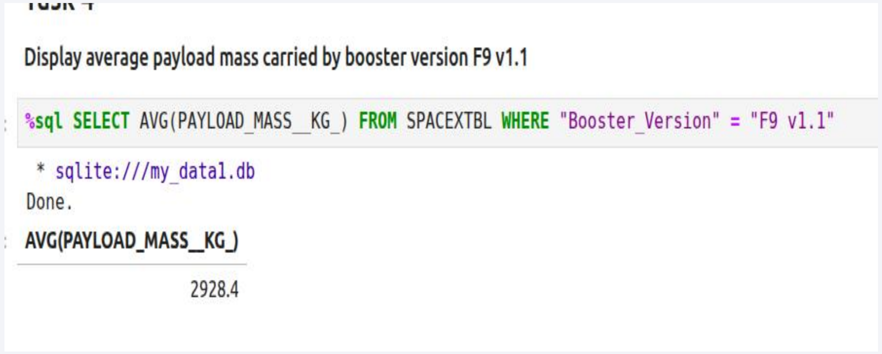
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE "Customer" = "NASA (CRS)"

 * sqlite:///my_data1.db
Done.

SUM(PAYLOAD_MASS__KG_)
                 45596
```

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1.

- SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE "Booster_Version" = "F9 v1.1".

  This query returns average payload mass where booster version is F9 v1.1.



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

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE "Booster_Version" = "F9 v1.1"
```

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

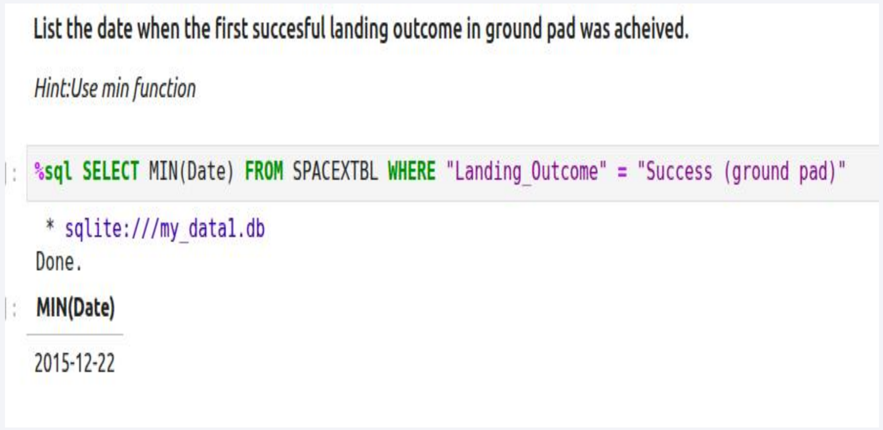
AVG(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad.

- SELECT MIN(Date) FROM SPACEXTBL WHERE "Landing_Outcome" = "Success (ground pad)".

  This query returns first date where landing outcome is Success (ground pad).



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

Hint:Use min function

```
%sql SELECT MIN(Date) FROM SPACEXTBL WHERE "Landing_Outcome" = "Success (ground pad)"
```

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

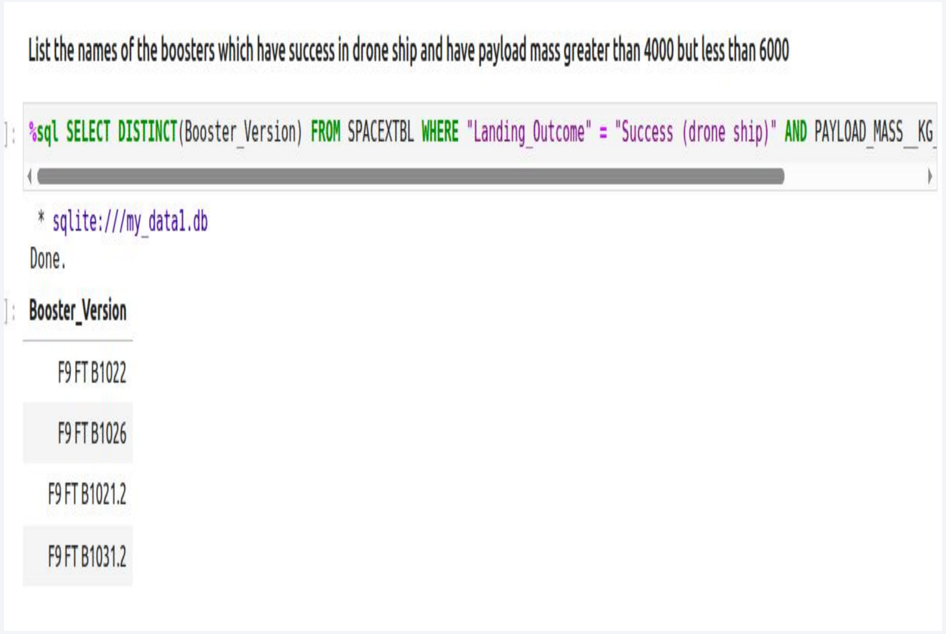
MIN(Date)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000.

- SELECT DISTINCT(Booster_Version) FROM SPACEXTBL WHERE "Landing_Outcome" = "Success (drone ship)" AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000.

  This query returns unique name of booster version where landing outcome is success (drone ship) and payload mass is between 4000 to 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 SELECT DISTINCT(Booster_Version) FROM SPACEXTBL WHERE "Landing_Outcome" = "Success (drone ship)" AND PAYLOAD_MASS__KG
```

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

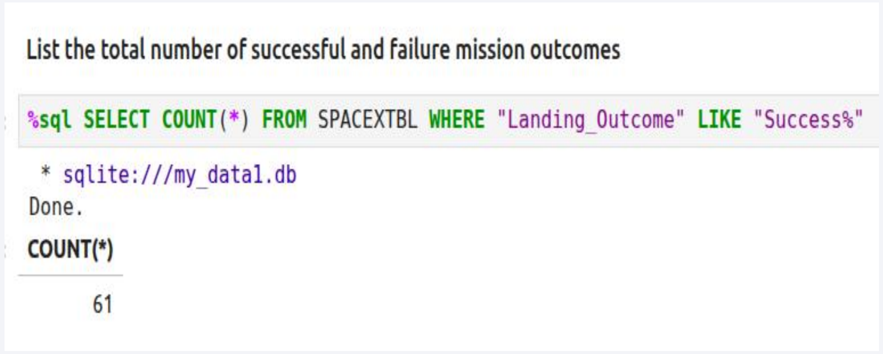
Booster_Version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes.

- SELECT COUNT(*) FROM SPACEXTBL WHERE "Landing_Outcome" LIKE "Success%".

  This query number of records where landing outcome is Success.



List the total number of successful and failure mission outcomes

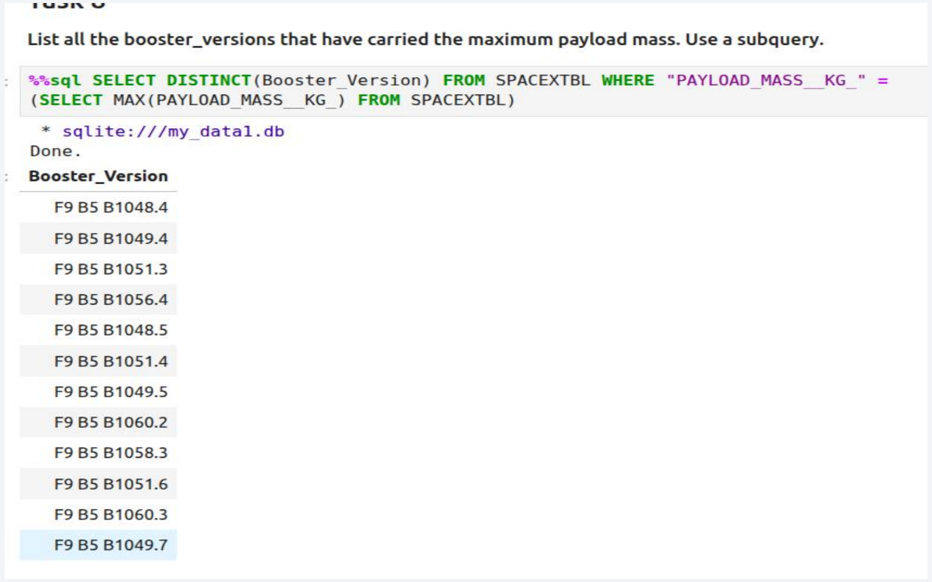
```
%sql SELECT COUNT(*) FROM SPACEXTBL WHERE "Landing_Outcome" LIKE "Success%"
```

```
 * sqlite:///my_data1.db
Done.
COUNT(*)
      61
```

- List the names of the booster which have carried the maximum payload mass

- SELECT DISTINCT(Booster_Version) FROM SPACEXTBL WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL).

  This query returns unique booster version name where payload mass is maximum.



List all the booster_versions that have carried the maximum payload mass. Use a subquery.

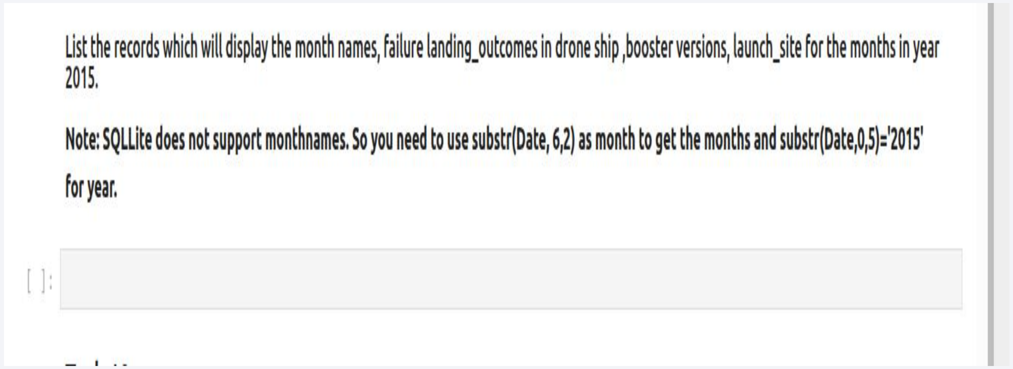
```
%%sql SELECT DISTINCT(Booster_Version) FROM SPACEXTBL WHERE "PAYLOAD_MASS__KG_" =
(SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)
```

```
 * 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

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

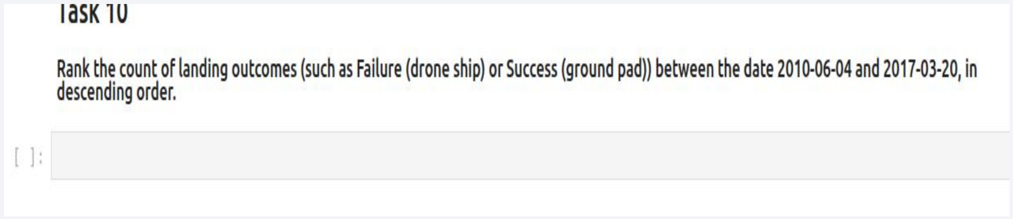


List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

```
[ ]:
```

- 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.



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.

```
[ ]:
```

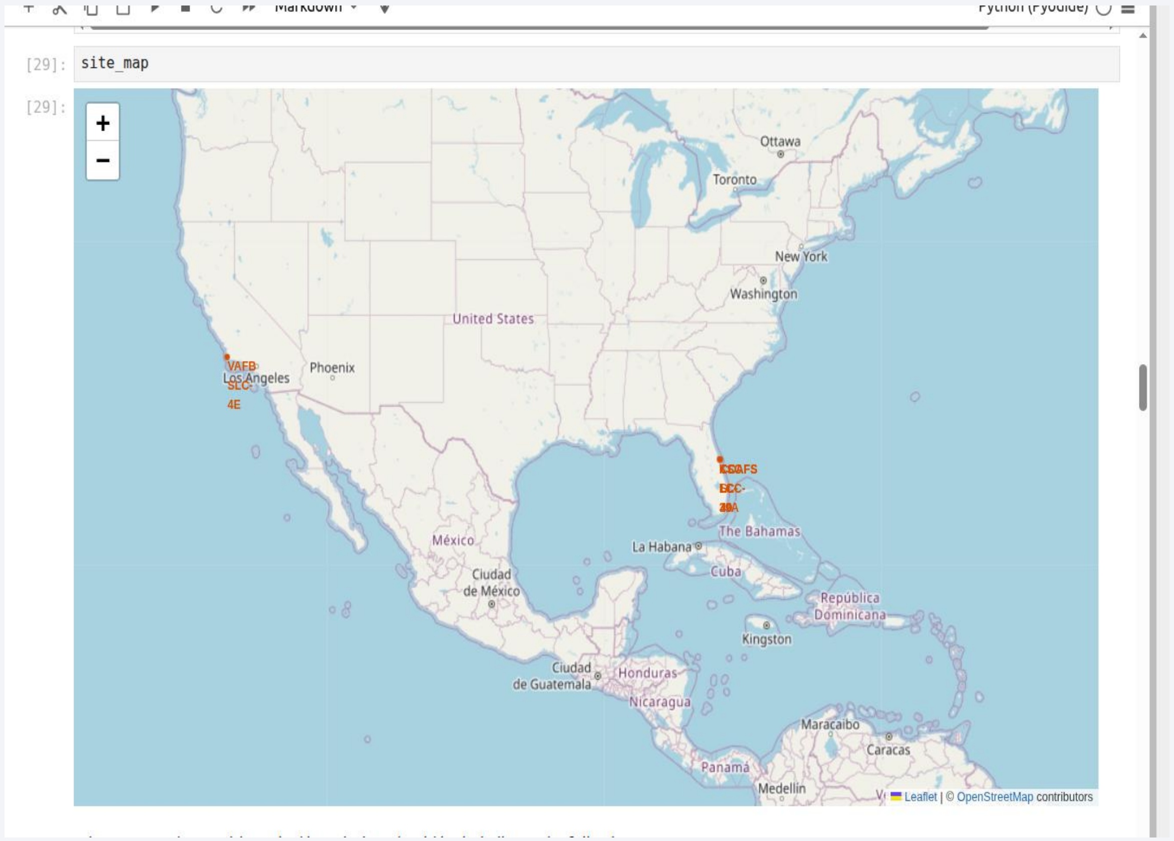
Section 3

Launch Sites Proximities Analysis

<Folium Map Screenshot 1>

Launch vizualization in world map:

We use folium library to plot map.
Then we select important column and done groupby on launch site.
Then we initiate folium marker and circle object then add this object to our map.
To see where our lauch site are in map.

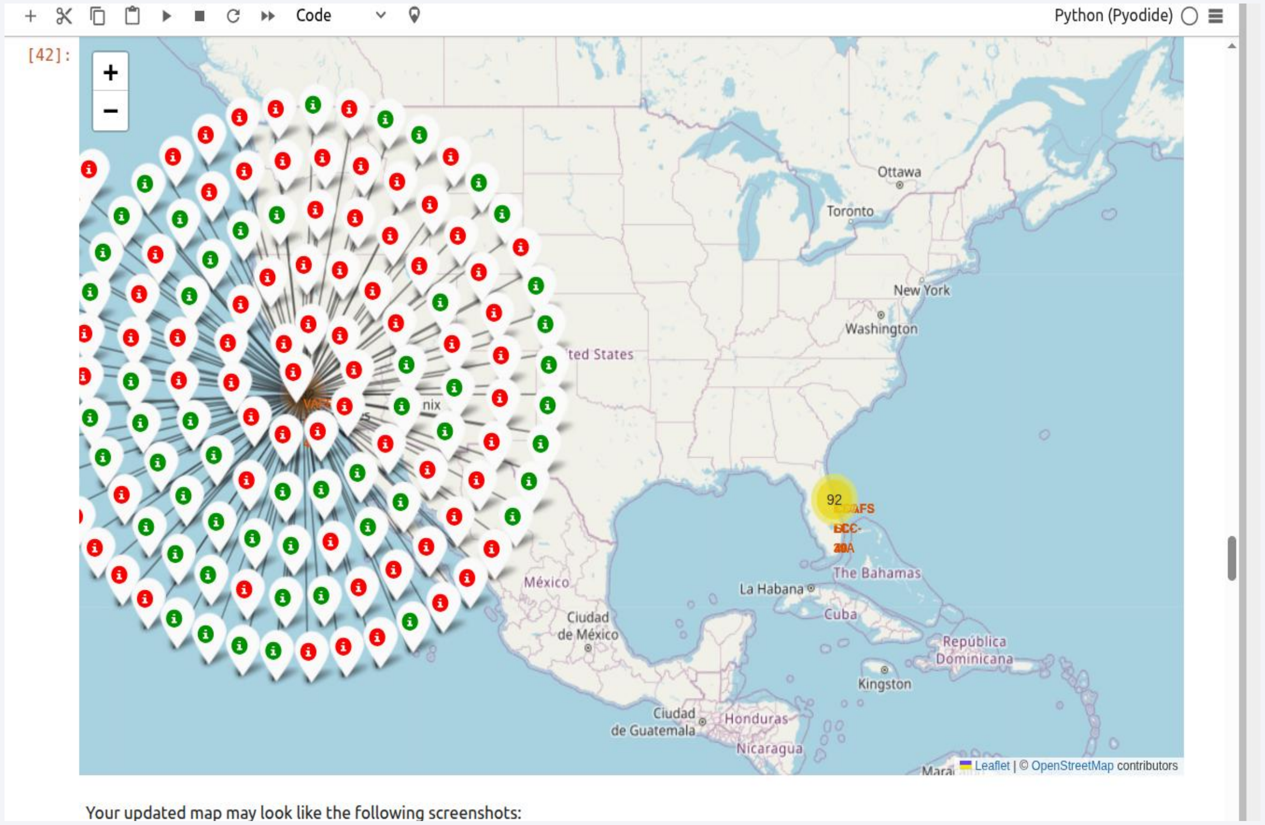


```
[29]: site_map
```

<Folium Map Screenshot 2>

Vizualization of each Launch Site success rate:

Then we build marker cluster to see number of success and failure in each site.through this we see that some site launches very less rockets as compare to other sites

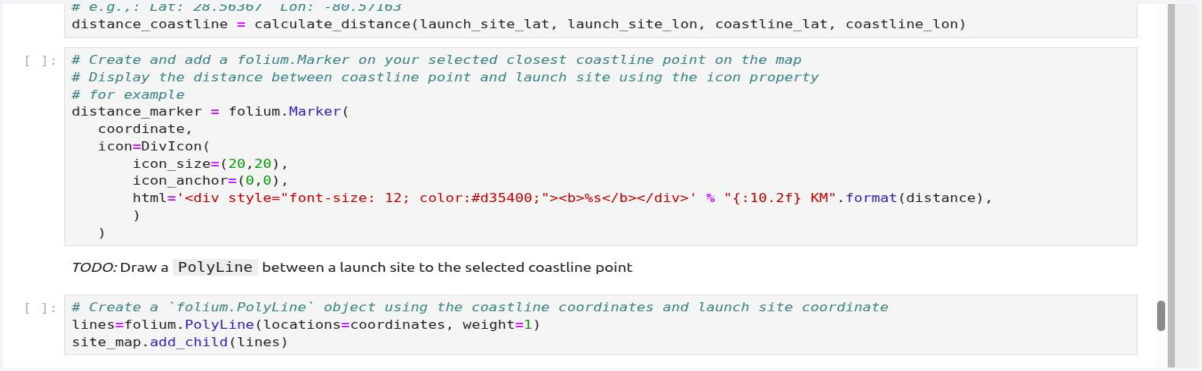


Your updated map may look like the following screenshots:

<Folium Map Screenshot 3>

Vizualization of coastline from Launch Site:

In last we helper function and mouseposition object to draw poly line to see how far is launch site to its nearest coastline.

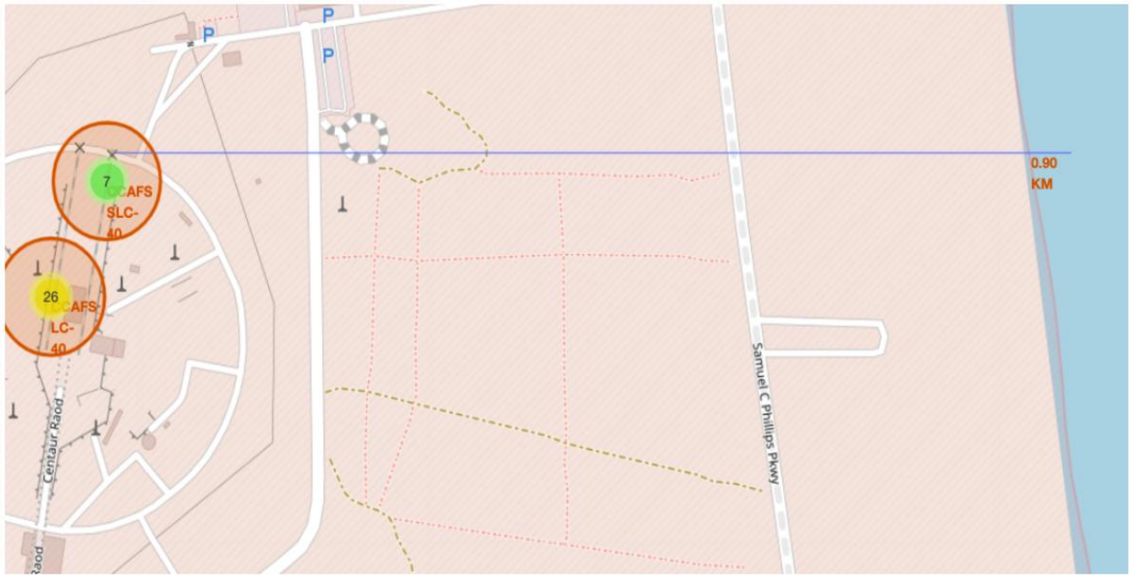


```
# e.g.,: Lat: 28.56367  Lon: -80.57163
distance_coastline = calculate_distance(launch_site_lat, launch_site_lon, coastline_lat, coastline_lon)
```

```
[ ]: # Create and add a folium.Marker on your selected closest coastline point on the map
     # Display the distance between coastline point and launch site using the icon property
     # for example
     distance_marker = folium.Marker(
        coordinate,
        icon=DivIcon(
            icon_size=(20,20),
            icon_anchor=(0,0),
            html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % "{:10.2f} KM".format(distance),
            )
        )
```

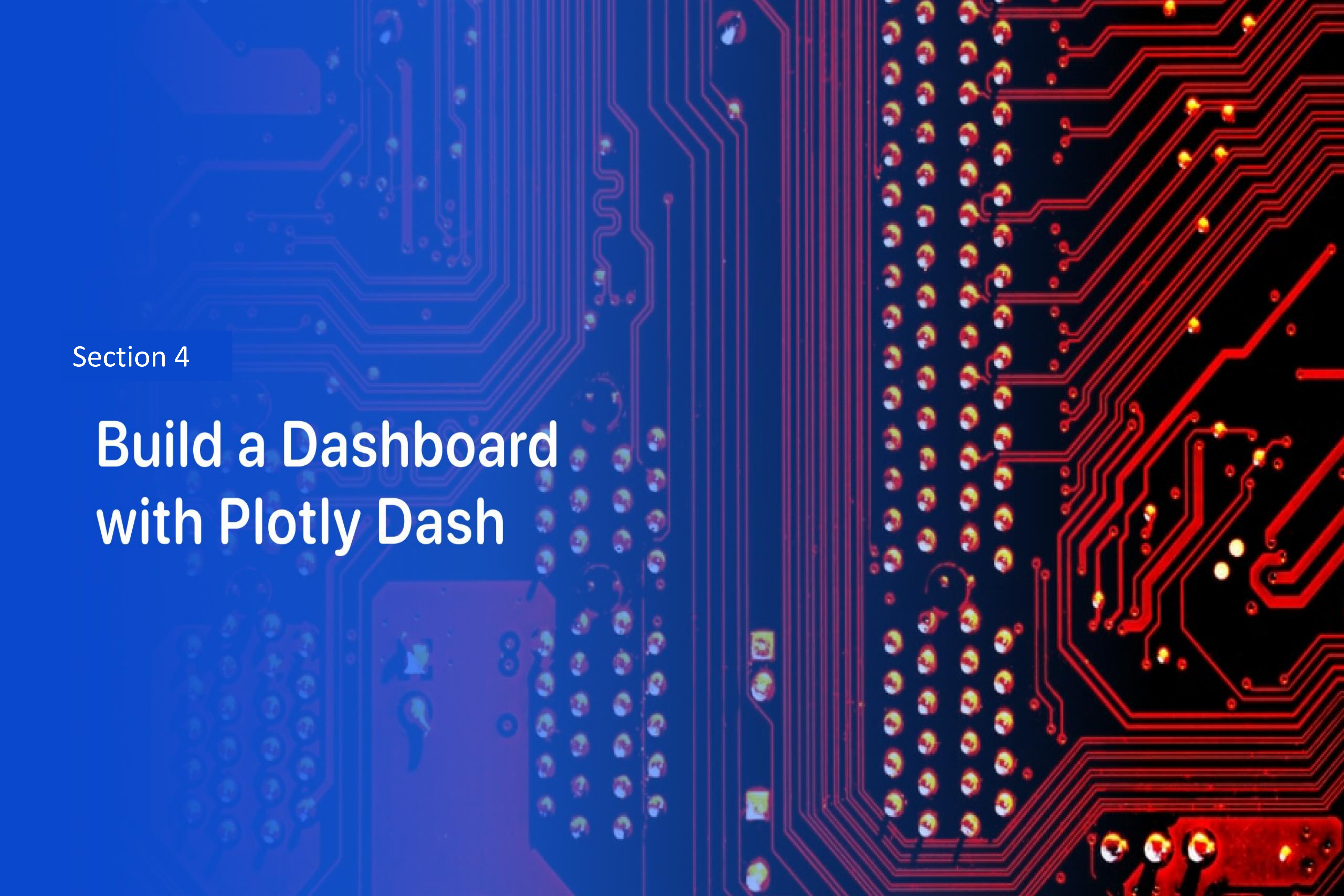
TODO: Draw a `PolyLine` between a launch site to the selected coastline point

```
[ ]: # Create a `folium.PolyLine` object using the coastline coordinates and launch site coordinate
     lines=folium.PolyLine(locations=coordinates, weight=1)
     site_map.add_child(lines)
```



Section 4

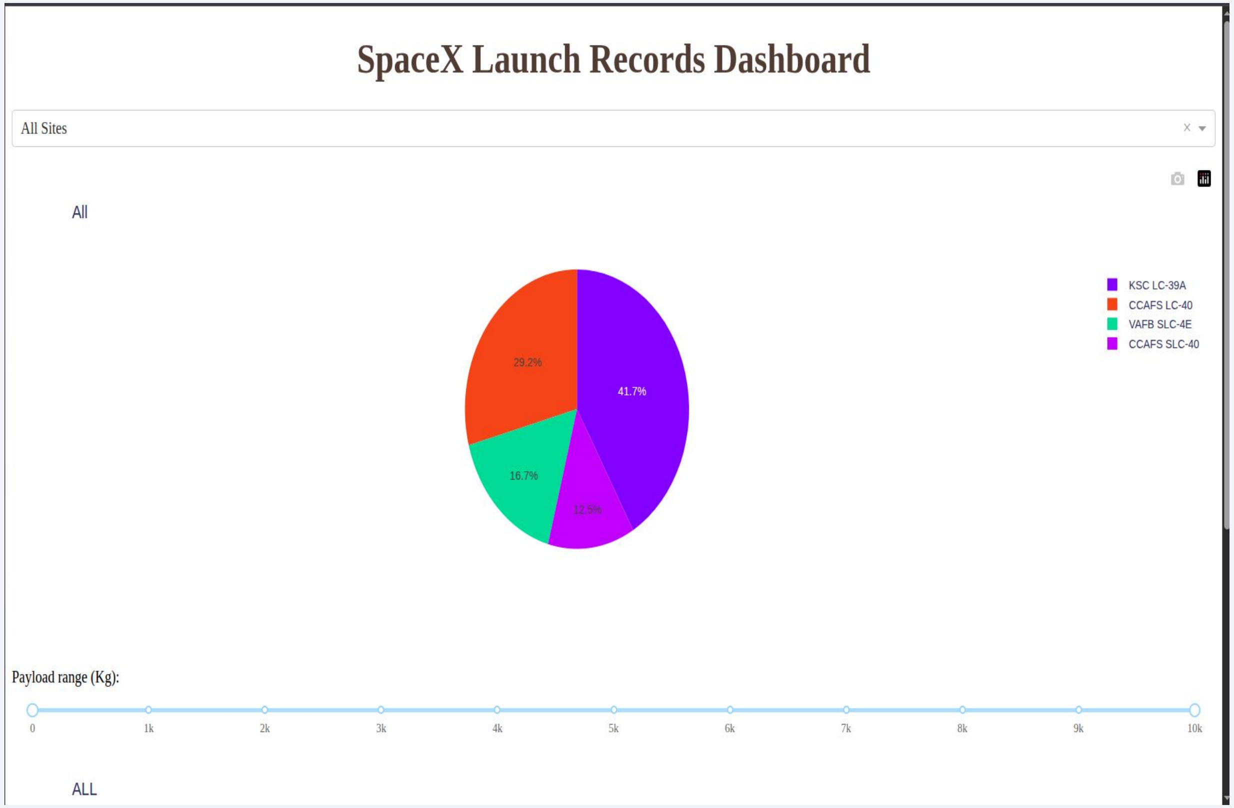
Build a Dashboard with Plotly Dash



<Dashboard Screenshot 1>

Piechart of all sites:

First we add dropdown with all unique option of launch site.
Then I add piechart with callback function by using this dropdown and I also filters data as per selected value in dropdown.

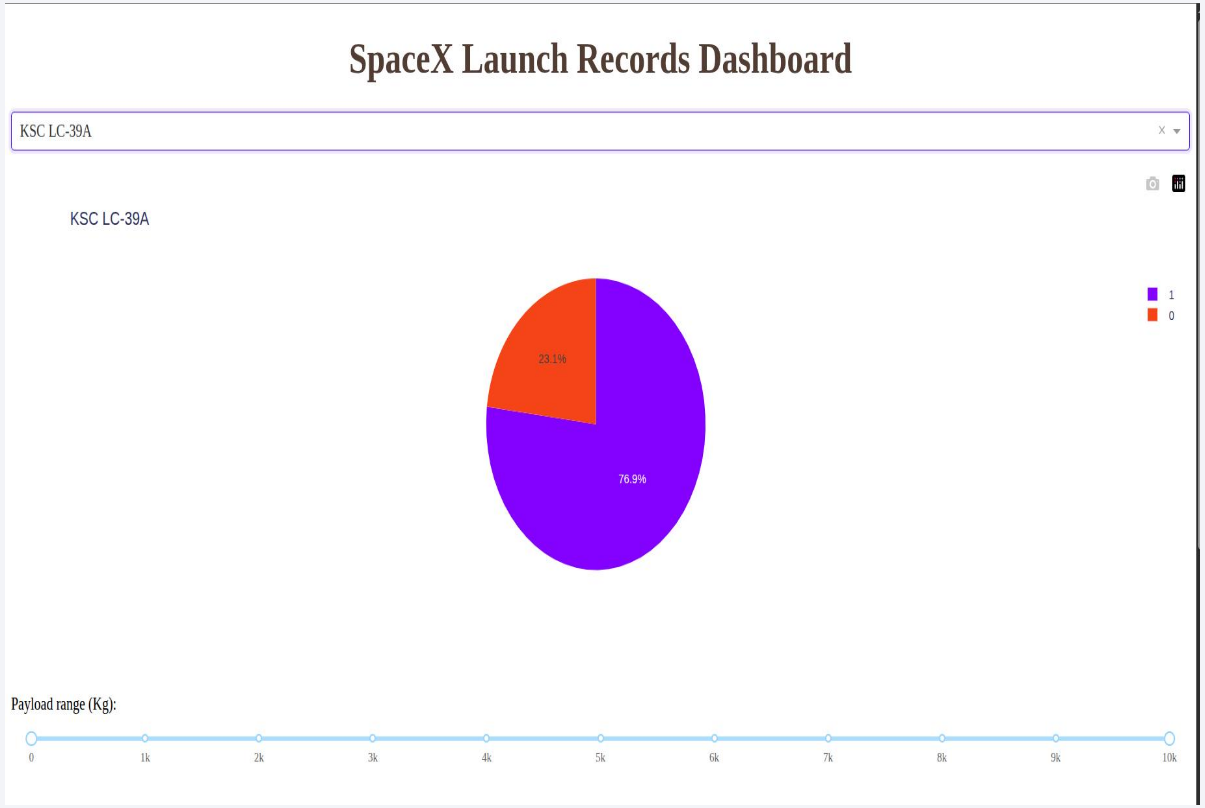

SpaceX Launch Records Dashboard

All Sites

<Dashboard Screenshot 2>

Vizualizing piechart of that site which has highest rate of success:

Here we can see that KSC LC-39A has highest rate of success, it maybe because this Site lauched very less rockets.

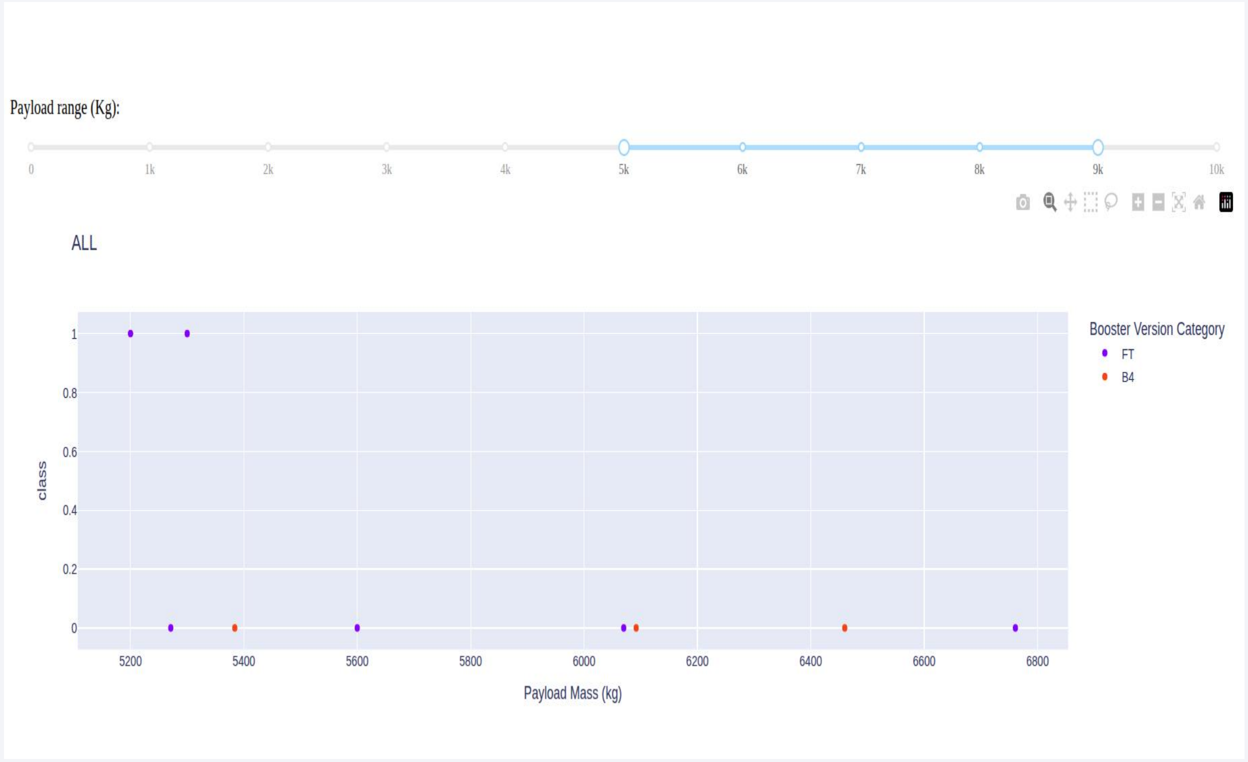

SpaceX Launch Records Dashboard

KSC LC-39A

<Dashboard Screenshot 3>

Vizualization of scatter plot of payload mass and success column:

Then I add range slider for user to choose range of payload mass for scatter plot and then I use another callback function for this scatter plot with two input first input is dropdown and second input is range slider and in last I filtered data as per both input



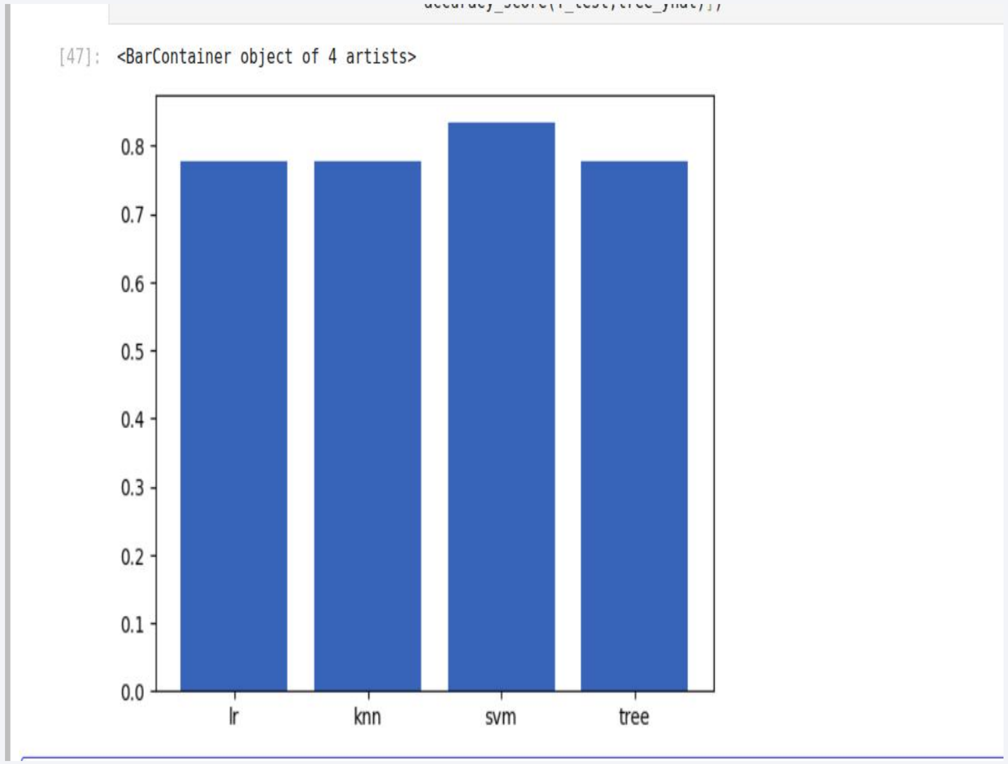
Section 5

Predictive Analysis (Classification)

Classification Accuracy

- Visualize the built model accuracy for all built classification models, in a bar chart

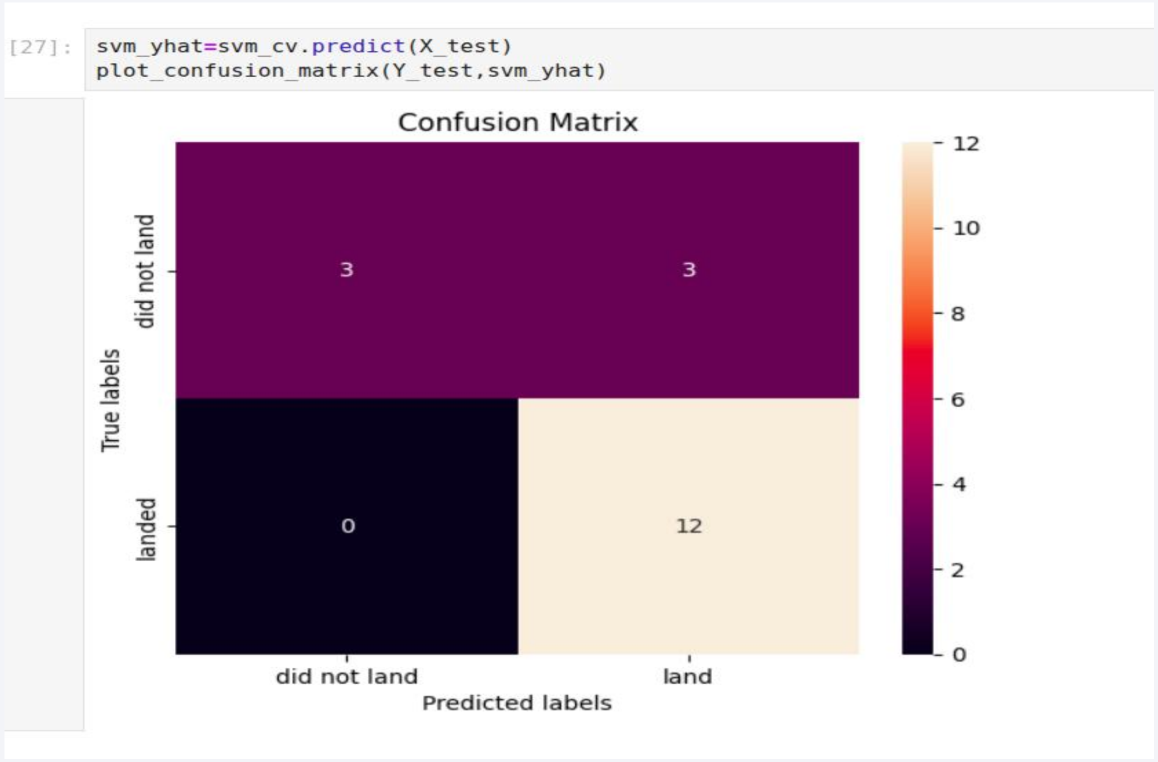
- AS we can see that our SVM has best accuracy rate as compare other model with above 80% accuracy.



```
[47]: <BarContainer object of 4 artists>
```

Confusion Matrix

- Show the confusion matrix of the best performing model SVM.
- As we can see that there is not a one case where our model say's that it will not land and it landed but there is 3 case where model say's it will land but it did'nt .

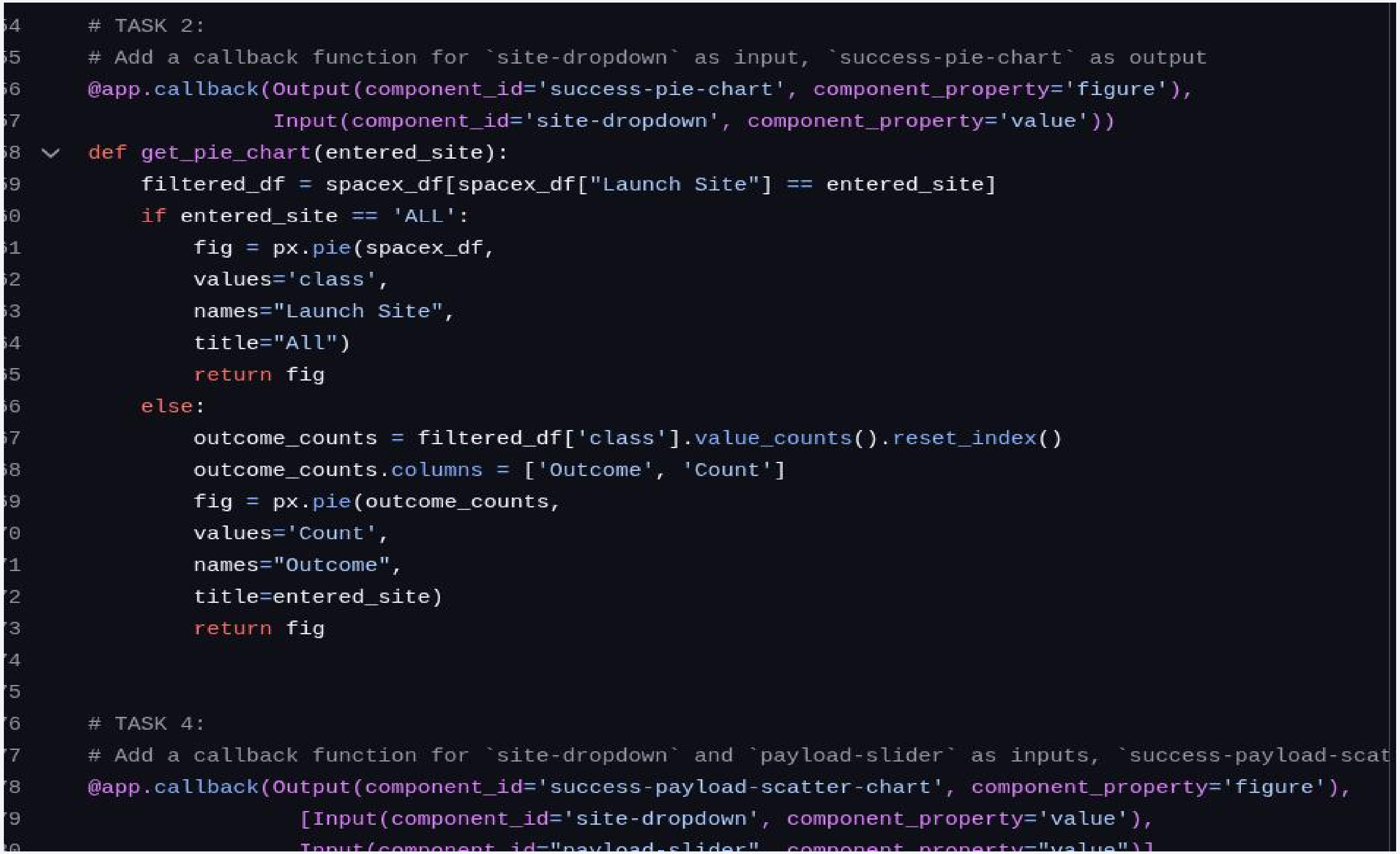


```
[27]: svm_yhat=svm_cv.predict(X_test)
      plot_confusion_matrix(Y_test,svm_yhat)
```

Conclusions

- There is no type 2 error.
- There is 3 case of type 1 error.
- We choose svm because it has high accuracy and has maximum roc_auc score.

Appendix



```
# TASK 2:
# Add a callback function for `site-dropdown` as input, `success-pie-chart` as 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):
    filtered_df = spacex_df[spacex_df["Launch Site"] == entered_site]
    if entered_site == 'ALL':
        fig = px.pie(spacex_df,
        values='class',
        names="Launch Site",
        title="All")
        return fig
    else:
        outcome_counts = filtered_df['class'].value_counts().reset_index()
        outcome_counts.columns = ['Outcome', 'Count']
        fig = px.pie(outcome_counts,
        values='Count',
        names="Outcome",
        title=entered_site)
        return fig


# TASK 4:
# Add a callback function for `site-dropdown` and `payload-slider` as inputs, `success-payload-scat
@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")]
```

Thank you!