IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Mazen al hatem>
<02-02-2024>

Outline

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

Executive Summary

- Summary of methodologies

Data collection : such as API , web Scraping

Data Wrangling

Machine Learning Classification : KNN, Logestic Regression, SVM Regression ,Decision Tree

- Summary of all results

KSC LC 39A has the best success Launching Ration among other Sites

Introduction

- The main Goal of this Project is to apply the main concepts of Data science by Solving a Problem of better understanding the Space missions by studding the History of Launching Process and how it is affected by many features and circumstances .

- Applying different concepts of Data science and Visualize the Data in an Attractive way is our second goal in this Project.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology: Different ways to collect the Data
- Perform data wrangling
  - The missing values have been replaced with the mean of the relevant values
- Perform exploratory data analysis (EDA) using visualization and SQL

  How to inquery the Data using sql light
- Perform interactive visual analytics using Folium and Plotly Dash

  How can Dash library be used to develop an interactive Dashboard
- Perform predictive analysis using classification models
  - Split the Data to Train and Test Parts then create a model and train the data, Predict the Values and Evaluate the model Results using Machine Learning Techniques .

Data Collection

- The data SpaceX Falcon 9 was collected by calling a get Request to the SpaceX API.

- in The Following Slides more about Data Collection

Data Collection - SpaceX API

- Get the Data

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

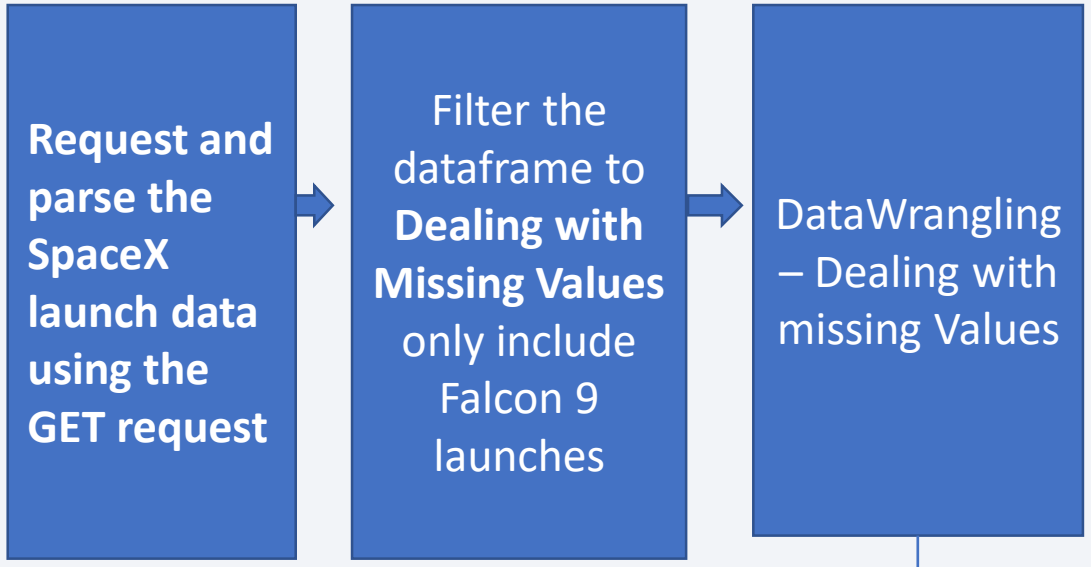
- Normalize the data

```
# Use json_normalize meethod to convert the json result into a dataframe
response=requests.get(static_json_url)
df=pd.json_normalize(response.json())
```

- Filter the Data

```
data_falcon9=dic_df.loc[dic_df.BoosterVersion=='Falcon 9']
data_falcon9.head(5)
```

- Data Wrangling




```
# Calculate the mean value of PayloadMass column
mean_df=data_falcon9['PayloadMass'].mean()
mean_df
data_falcon9.PayloadMass.replace(np.nan,mean_df, inplace=True)
data_falcon9.isnull().sum()
# Replace the np.nan values with its mean value
```

Reference

[jupyter-labs-spacex-data-collection-api.ipynb]

https://github.com/mazenhat/Data_Science.git

Data Collection - Scraping

- Extract a Falcon 9 launch records HTML table from Wikipedia

- Parse the table and convert it into a Pandas data frame

Request the Falcon9 Launch Wiki page from its URL

```
import requests
r = requests.get(static_url)
```

```
column_names = []

for th in first_launch_table.find_all('th'):
    column_name=extract_column_from_header(th)
    if column_name!='' and (column_name is not None):
        column_names.append(column_name)
    else:
        pass
```

Check the extracted column names

```
print(column_names)
```

```
['Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome']
```

Extract all column/variable names from the HTML table header

Create a data frame by parsing the launch HTML tables

Reference

[jupyter-labs-webscraping.ipynb]

https://github.com/mazenhat/Data_Science.git

parsing the launch HTML tables

- Parse the table and convert it into a Pandas data frame

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            #print(row[extracted_row])
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            launch_dict['Flight No.'].append(flight_number)
            datatimelist=date_time(row[0])
            # Date value
            # TODO: Append the date into launch_dict with key `Date`
            #launch_dict['Date'].append(row.th.string)
            date = datatimelist[0].strip(',')
            launch_dict['Date'].append(date)
            #print(date)
            # Time value
            # TODO: Append the time into launch_dict with key `Time`
            time = datatimelist[1]
            launch_dict['Time'].append(time)
            #print(time)
            # Booster version
            # TODO: Append the bv into launch_dict with key `Version Booster`
            bv=booster_version(row[1])
            if not(bv):
                bv=row[1].a.string
                launch_dict['Version Booster'].append(bv)
```

Reference

[jupyter-labs-webscraping.ipynb]

https://github.com/mazenhat/Data_Science.git

parsing the launch HTML tables

- .csv is included in the GIT repository .

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

```
dd = df.reset_index(drop=True)
dd
```

	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	customer	Success\n	F9 v1.1	Failure	4 June 2010	18:45
1	2	CCAFS	Dragon	0	LEO	customer	Success	F9 v1.1	Failure	8 December 2010	15:43
2	3	CCAFS	Dragon	525 kg	LEO	customer	Success	F9 v1.1	No attempt\n	22 May 2012	07:44
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	customer	Success\n	F9 v1.1	No attempt	8 October 2012	00:35
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	customer	Success\n	F9 v1.1	No attempt\n	1 March 2013	15:10
...	...	...	...	...	...	...	...	...	...	...	...
116	117	CCSFS	Starlink	15,600 kg	LEO	customer	Success\n	NaN	Success	9 May 2021	06:42
117	118	KSC	Starlink	~14,000 kg	LEO	customer	Success\n	NaN	Success	15 May 2021	22:56
118	119	CCSFS	Starlink	15,600 kg	LEO	customer	Success\n	NaN	Success	26 May 2021	18:59
119	120	KSC	SpaceX CRS-22	3,328 kg	LEO	customer	Success\n	NaN	Success	3 June 2021	17:29
120	121	CCSFS	SXM-8	7,000 kg	GTO	customer	Success\n	NaN	Success	6 June 2021	04:26

121 rows × 11 columns

```
dd.to_csv('spacex_web_scraped.csv', index=False, encoding="utf-8-sig")
```

Reference

[jupyter-labs-webscraping.ipynb]

https://github.com/mazenhat/Data_Science.git

Data Wrangling

Calculate the number of launches on each site

```
# Apply value_counts() on column LaunchSite
df.value_counts('LaunchSite')
```

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

Calculate the number and occurrence of each orbit

```
# Apply value_counts on Orbit column
df.value_counts('Orbit')
```

```
Orbit
GTO     27
ISS     21
VLEO    14
PO       9
LEO      7
SSO      5
MEO      3
ES-L1    1
GEO      1
HEO      1
SO       1
```

Calculate the number and occurence of mission outcome of the orbits

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for out in df['Outcome']:
  if out in bad_outcomes: #print ('bad outcome')
      landing_class.append(0)
  else:
       landing_class.append(1)
    # print('ok')
```

```
# landing_outcomes = values on Outcome column
landing_outcomes = df.value_counts('Outcome')
landing_outcomes
```

```
Outcome
True ASDS       41
None None       19
True RTLS       14
False ASDS       6
True Ocean       5
False Ocean      2
None ASDS        2
False RTLS       1
```

Create a landing outcome label from Outcome column

Reference

[labs-jupyter-spacex-Data wrangling.ipynb]

https://github.com/mazenhat/Data_Science.git

Data Wrangling

Calculate the number and occurrence of mission outcome of the orbits

```
# landing_outcomes = values on Outcome column
landing_outcomes = df.value_counts('Outcome')
landing_outcomes
```

```
Outcome
True ASDS       41
None None       19
True RTLS       14
False ASDS       6
True Ocean       5
False Ocean      2
None ASDS        2
False RTLS       1
```

Create a landing outcome label from Outcome column

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for out in df['Outcome']:
  if out in bad_outcomes: #print ('bad outcome')
      landing_class.append(0)
  else:
       landing_class.append(1)
    # print('ok')
```

Reference

[labs-jupyter-spacex-Data wrangling.ipynb]

https://github.com/mazenhat/Data_Science.git

EDA with Data Visualization

- Visualize the relationship between Flight Number and Launch Site
- Visualize the relationship between Payload and Launch Site
- Visualize the relationship between success rate of each Orbit type
- Visualize the relationship between FlightNumber and Orbit type
- Visualize the relationship between Payload and Orbit type
- Visualize the launch success yearly trend

Reference

jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

https://github.com/mazenhat/Data_Science.git

EDA with SQL

- 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 succesful landing outcome in ground pad was acheived.
- 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. Use a subquery
- 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.

Reference

jupyter-labs-eda-sql-coursera_sqllite.ipynb

https://github.com/mazenhat/Data_Science.git

All sql queries are in the Repository Git

Build an Interactive Map with Folium

- Mark all launch sites on a map
- Mark the success/failed launches for each site on the map
- Calculate the distances between a launch site to its proximities

Markers and Circles on map make it easy to show the Info in the Geographical point of view and will enrich the Presentation with more Details and Ideas.

Reference

lab_jupyter_launch_site_location.jupyterlite.ipynb

https://github.com/mazenhat/Data_Science.git

Build a Dashboard with Plotly Dash

- Add a Launch Site Drop-down Input Component
- Add a callback function to render success-pie-chart based on selected site dropdown
- Add a Range Slider to Select Payload
- Add a callback function to render the success-payload-scatter-chart scatter plot

Designing an Interactive Dashboard is very creative and let the End user better understand the Figures

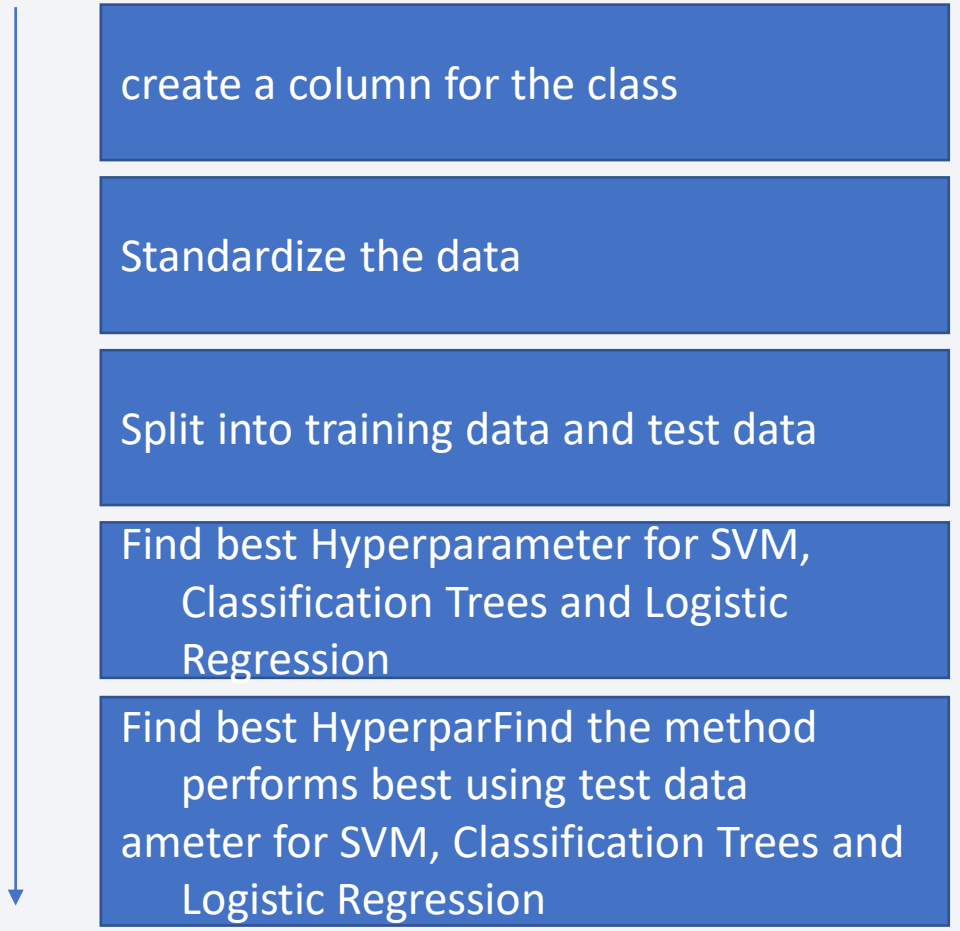
Reference

spacex_dash_app.py

https://github.com/mazenhat/Data_Science.git

Predictive Analysis (Classification)

- Prepare the Data
- Train the Data with Different Classification Algorithms
- Evaluation the Results



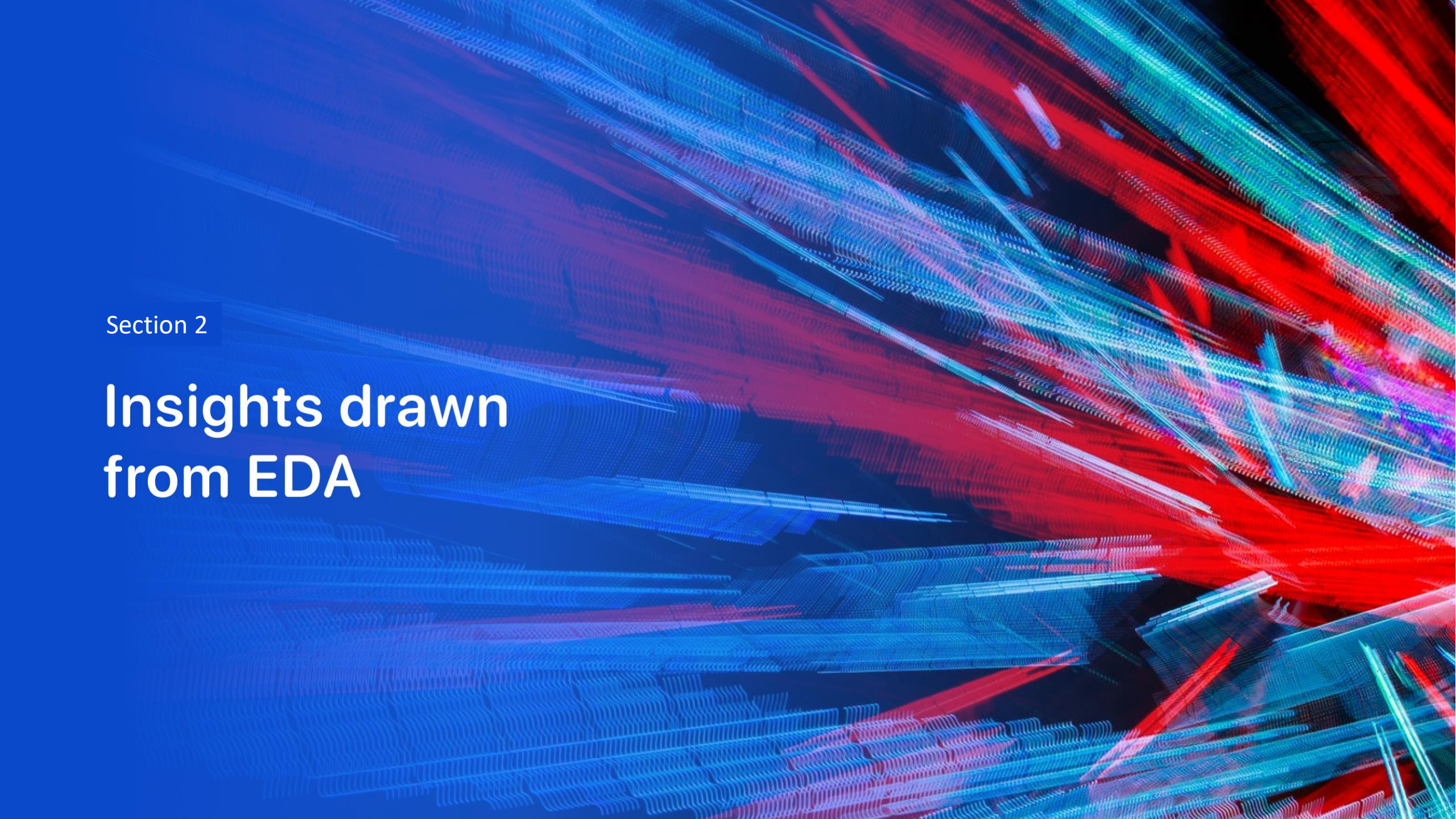
Reference

SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

https://github.com/mazenhat/Data_Science.git

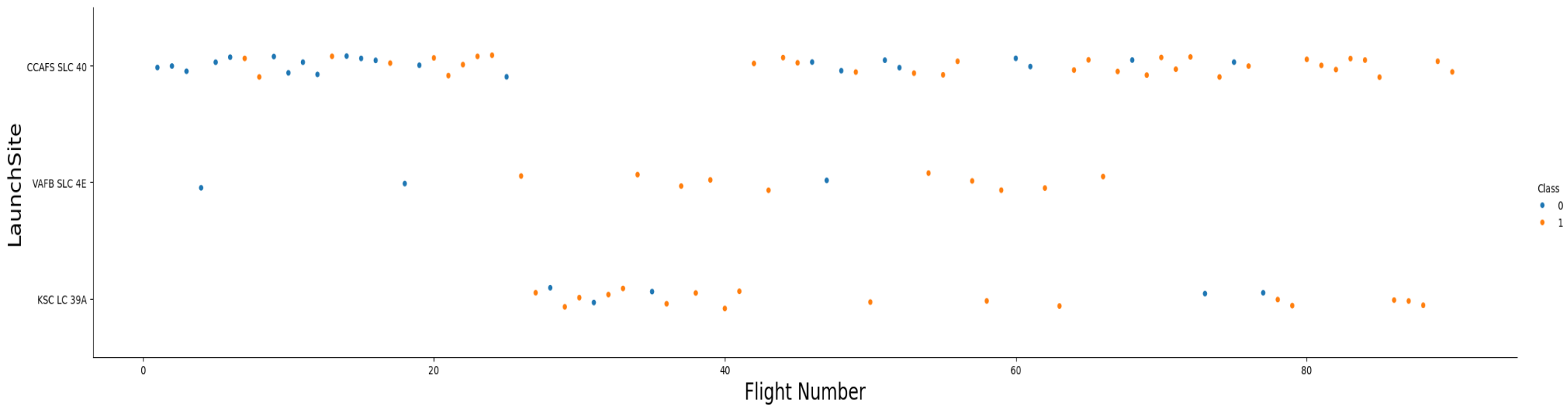
Results

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


Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

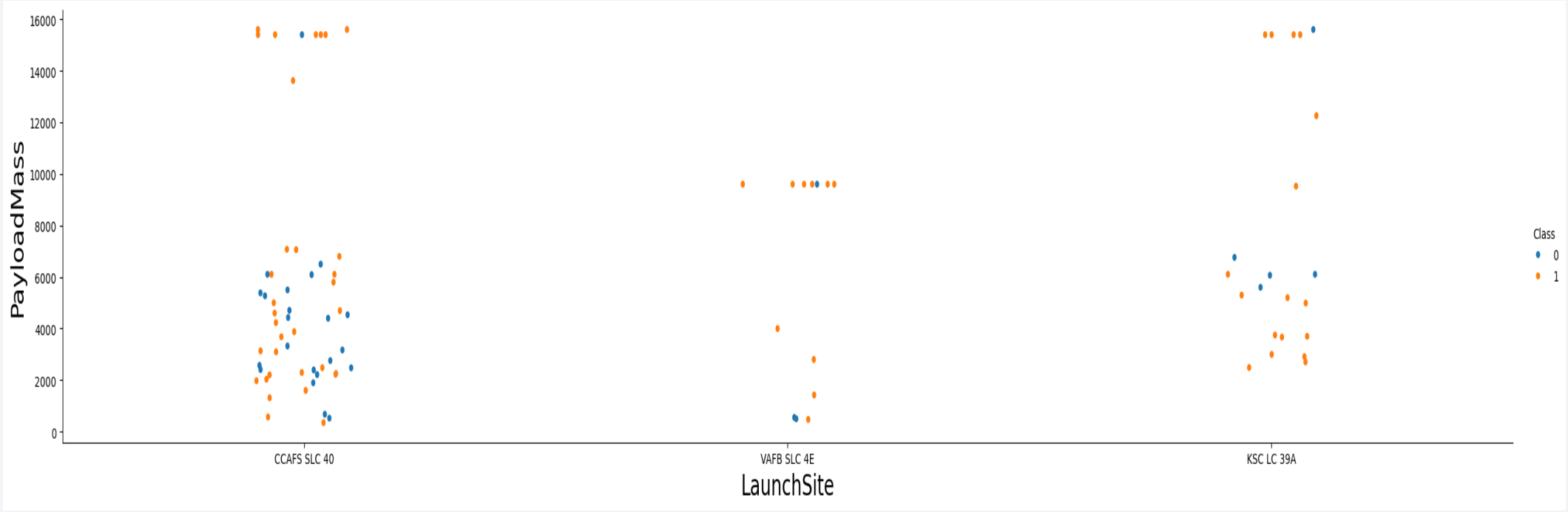
```
### TASK 1: Visualize the relationship between Flight Number and Launch Site
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("LaunchSite",fontsize=20)
plt.show()
```



The chance of successful Launch increases with more Flights number for All Launchsites

Payload vs. Launch Site

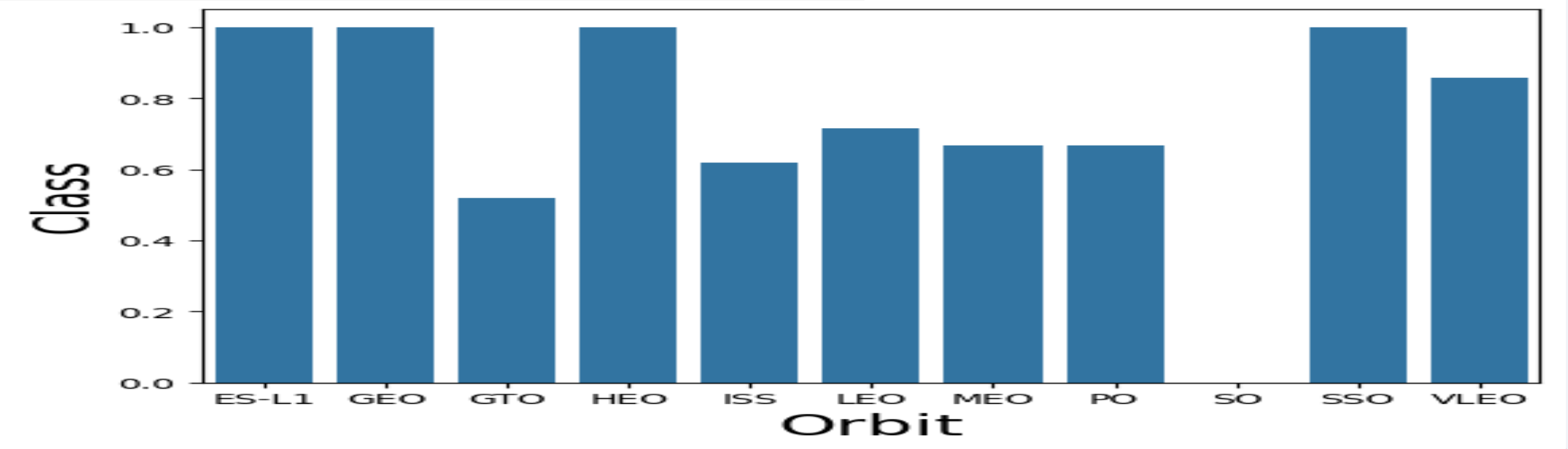
```
### TASK 2: Visualize the relationship between Payload and Launch Site
sns.catplot(y="PayloadMass", x="LaunchSite", hue="Class", data=df, aspect = 5)
plt.xlabel("LaunchSite",fontsize=20)
plt.ylabel("PayloadMass",fontsize=20)
plt.show()
```



for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

Success Rate vs. Orbit Type

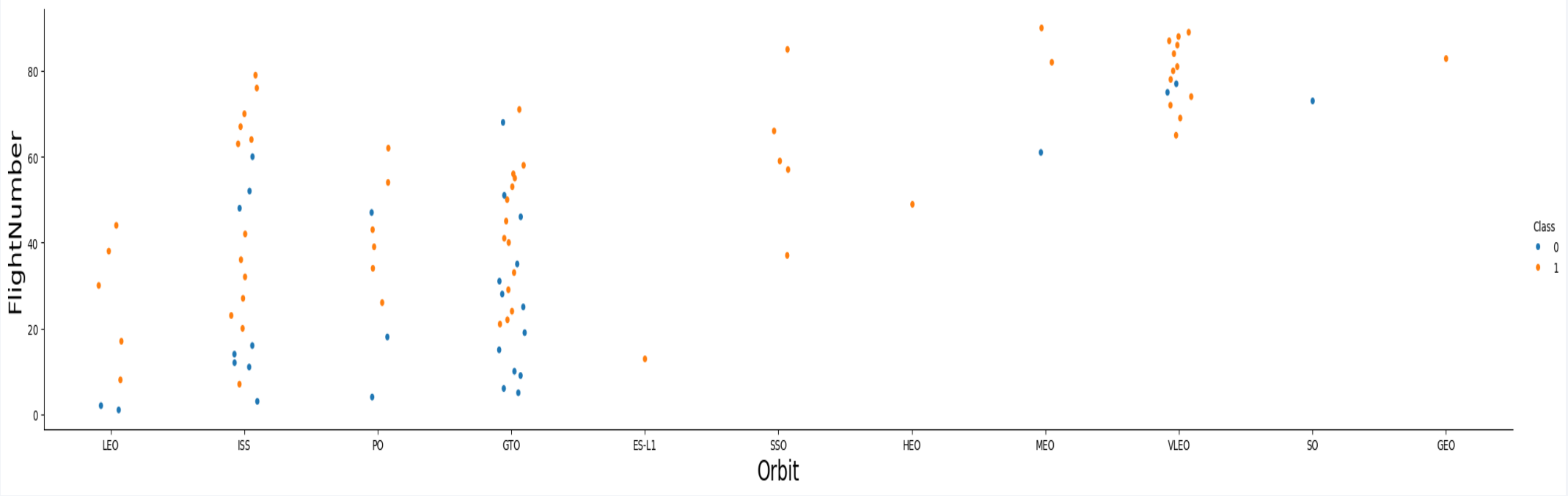
```
### TASK  3: Visualize the relationship between success rate of each orbit type

df_grouped=df.groupby(['Orbit'])['Class'].mean().to_frame()

sns.barplot(y="Class", x="Orbit", data=df_grouped)
plt.xlabel("Orbit",fontsize=20)
plt.ylabel("Class",fontsize=20)
plt.show()
```



Orbit SO has no Success Launch , success Rate for all other Orbits are similar with more advantage for SSO-HEO-ESL1-GEO

Flight Number vs. Orbit Type

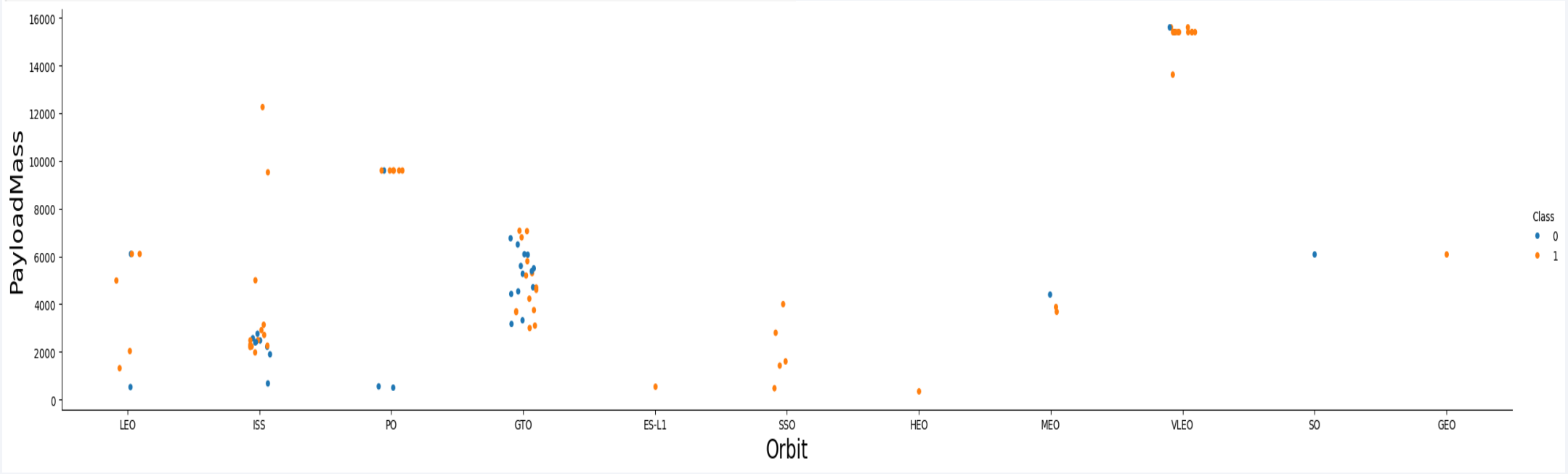
```
### TASK  4: Visualize the relationship between FlightNumber and Orbit type
sns.catplot(y="FlightNumber", x="Orbit", hue="Class", data=df, aspect = 5)
plt.xlabel("Orbit",fontsize=20)
plt.ylabel("FlightNumber",fontsize=20)
plt.show()
```



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

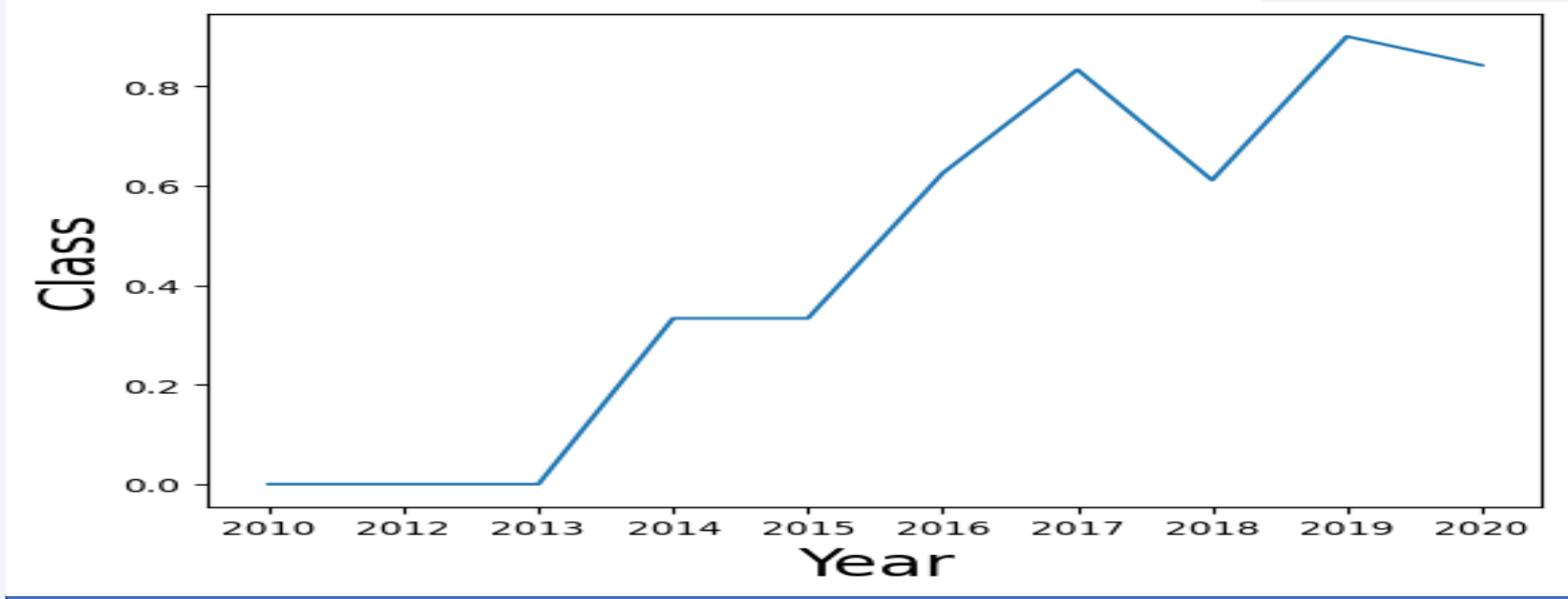
```
### TASK  5: Visualize the relationship between Payload and Orbit type
sns.catplot(y="PayloadMass", x="Orbit", hue="Class", data=df, aspect = 5)
plt.xlabel("Orbit",fontsize=20)
plt.ylabel("PayloadMass",fontsize=20)
plt.show()
```



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

```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
df_grouped_year=df.groupby(['Date'])['Class'].mean().to_frame()
sns.lineplot(y="Class", x="Date",  data=df_grouped_year)
plt.xlabel("Year",fontsize=20)
plt.ylabel("Class",fontsize=20)
plt.show()
```



the sucess rate since 2013 kept increasing till 2020

All Launch Site Names

```
%sql select distinct Launch_Site  from  SPACEXTABLE
```

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

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

Using DISTINCT to see just the unique sites

The Notebook is included in the Repository Data_Science Git

jupyter-labs-eda-sql-coursera_sqllite.ipynb

Launch Site Names Begin with 'CCA'

Task 2

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

```
%sql select * from  SPACEXTABLE  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

Using Like and Limit

The Notebook is included in the Repository Data_Science Git

jupyter-labs-eda-sql-coursera_sqllite.ipynb

Total Payload Mass

Task 3

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

```
%sql  select sum(PAYLOAD_MASS__KG_) from SPACEXTABLE where Customer = "NASA (CRS)"
```

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

sum(PAYLOAD_MASS__KG_)
45596

The Notebook is included in the Repository Data_Science Git

jupyter-labs-eda-sql-coursera_sqllite.ipynb

Using Sum on sql

Average Payload Mass by F9 v1.1

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

```
%sql  select avg(PAYLOAD_MASS__KG_) from SPACEXTABLE where Booster_Version = "F9 v1.1"
```

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

avg(PAYLOAD_MASS__KG_)
2928.4

Using AVG function

The Notebook is included in the Repository Data_Science Git

jupyter-labs-eda-sql-coursera_sqllite.ipynb

First Successful Ground Landing Date

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

Hint:Use min function

```
%sql  select min(Date) from SPACEXTABLE where Landing_Outcome="Success (ground pad)"
```

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

min(Date)
2015-12-22

Use Min Function

The Notebook is included in the Repository Data_Science Git

jupyter-labs-eda-sql-coursera_sqllite.ipynb

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  select distinct 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

The Notebook is included in the Repository Data_Science Git

jupyter-labs-eda-sql-coursera_sqllite.ipynb

Using Between in where clause

Total Number of Successful and Failure Mission Outcomes

List the total number of successful and failure mission outcomes ¶

```
%sql  select count(*),  Landing_Outcome from SPACEXTABLE where Landing_Outcome in ("Success","Failure") group by Landing_Outcome
```

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

count(*)	Landing_Outcome
3	Failure
38	Success

The Notebook is included in the Repository Data_Science Git

jupyter-labs-eda-sql-coursera_sqllite.ipynb

Three Failure vs 38 success was found - Group by Technique is used

Boosters Carried Maximum Payload

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

```
%sql select distinct booster_version,PAYLOAD_MASS__KG_ from SPACEXTABLE where PAYLOAD_MASS__KG_ = ( select max(PAYLOAD_MASS__KG_) from SPACEXTABLE)
```

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

Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

The Notebook is included in the Repository Data_Science Git

jupyter-labs-eda-sql-coursera_sqllite.ipynb

Using Subquery

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, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

```
%sql select substr(Date, 6,2) month ,Landing_Outcome from SPACEXTABLE  where Landing_Outcome="Failure (drone ship)" and substr(Date,0,5)='2015'
```

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

month	Landing_Outcome
01	Failure (drone ship)
04	Failure (drone ship)

The Notebook is included in the Repository Data_Science Git

jupyter-labs-eda-sql-coursera_sqllite.ipynb

In Jan and April are the most Failure in 2015

Dealing With string , using substr function

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 select Landing_Outcome , count(*) from SPACEXTABLE a WHERE a.[Date] between "2010-06-04 " and "2017-03-20" group by Landing_Outcome order by count(*) desc
```

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

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

The Notebook is included in the Repository Data_Science Git

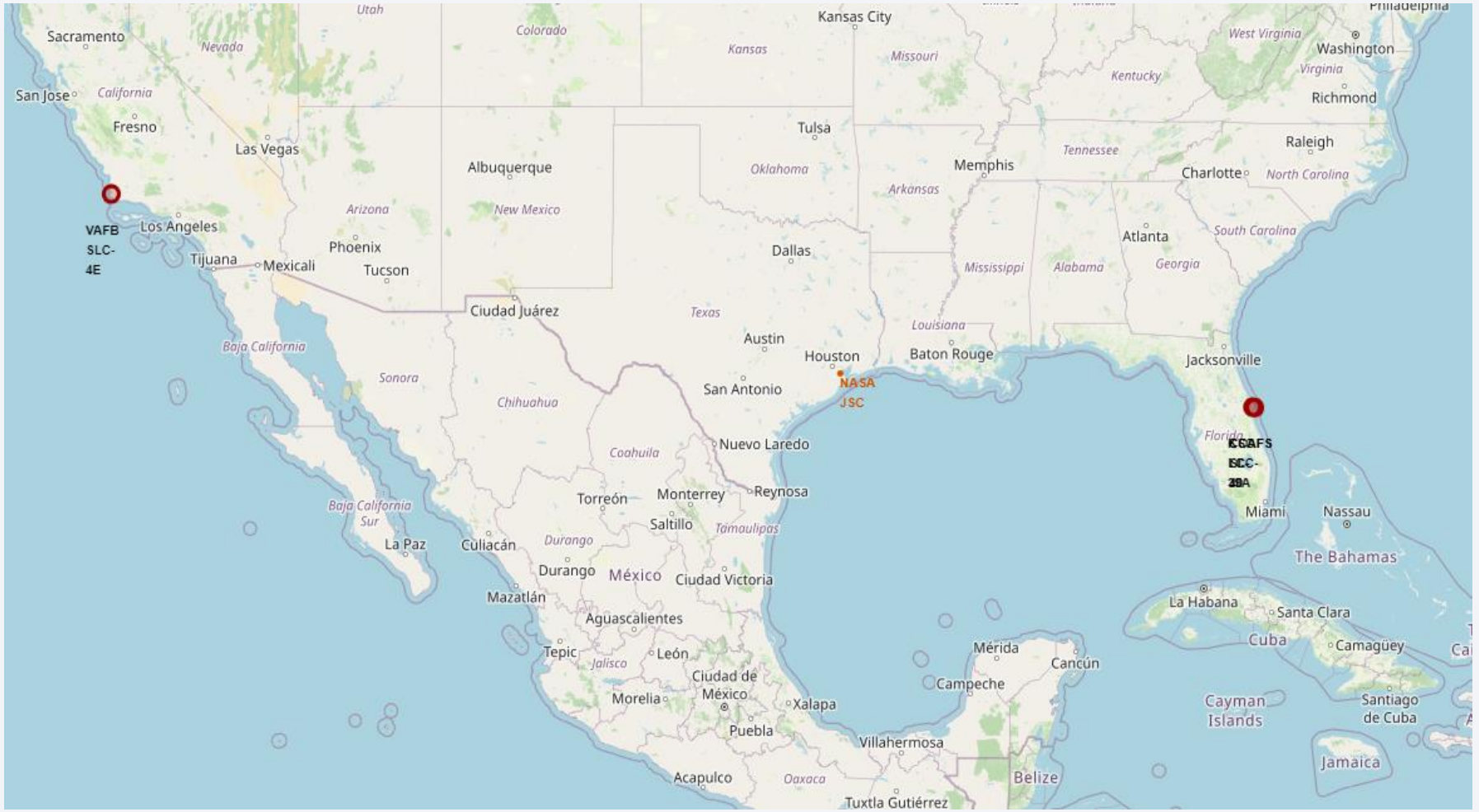
jupyter-labs-eda-sql-coursera_sqllite.ipynb

Count of all Landing outcomes Types

Using sql group by and order by statement


Section 3
Launch Sites
Proximities Analysis

Mark all launch sites on a Map

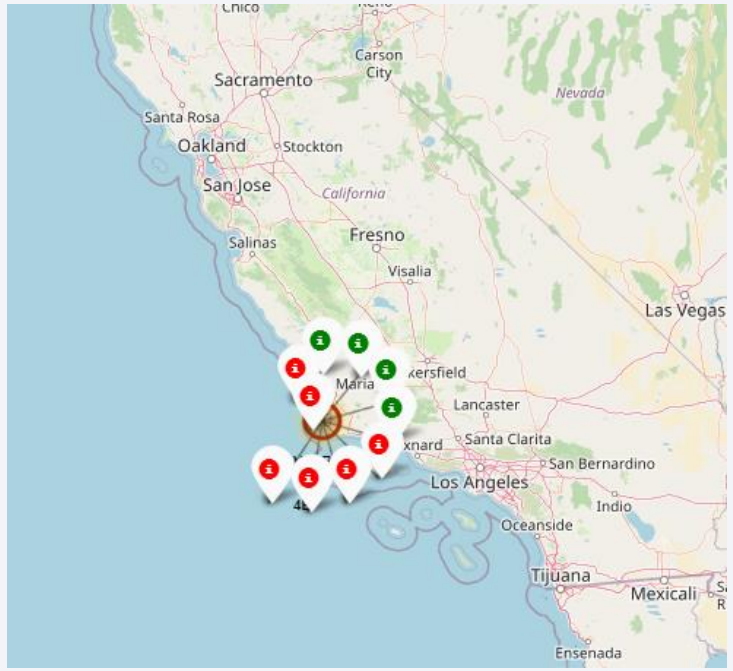


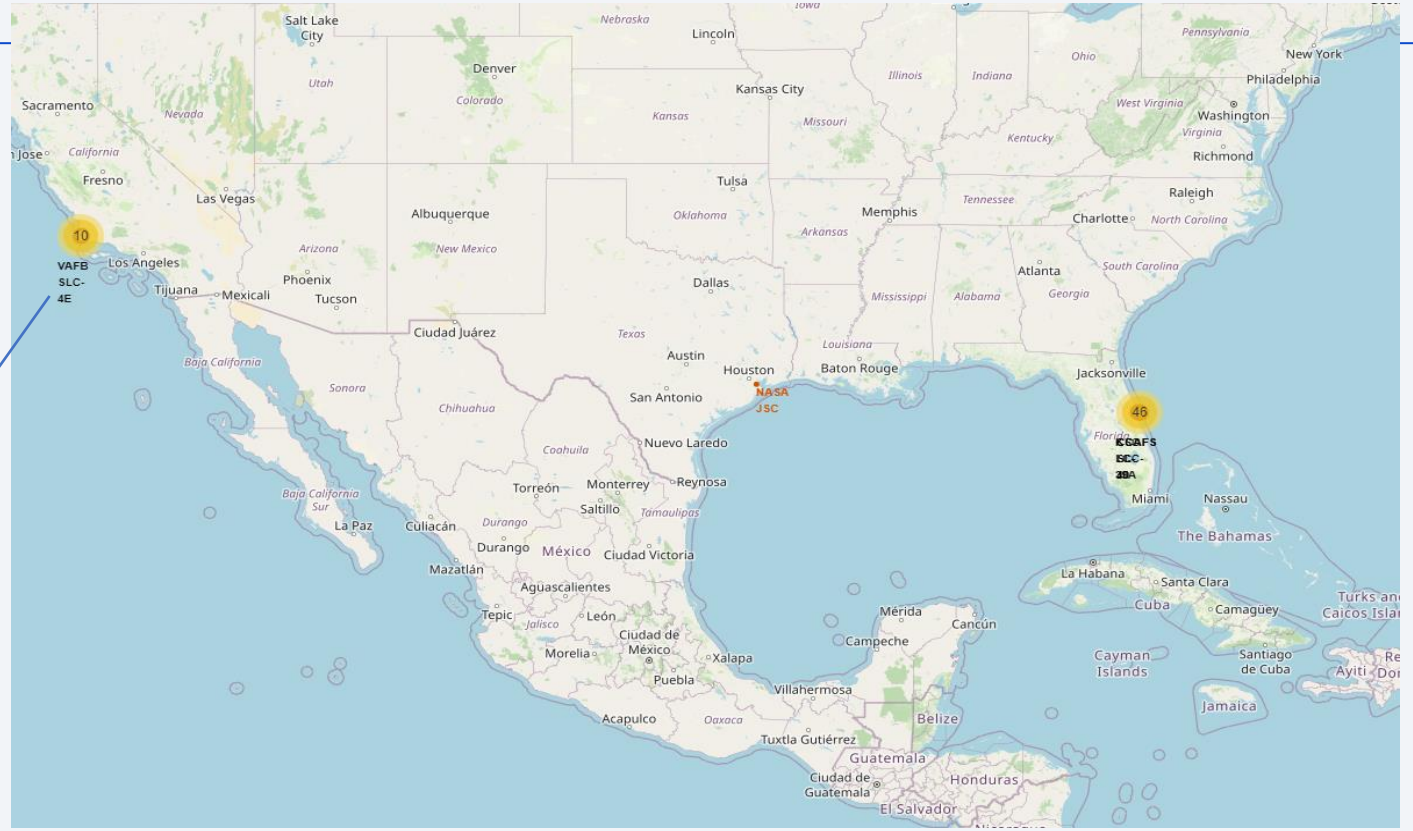
The Notebook with all scripts is included in the Repository Data_Science Git

lab_jupyter_launch_site_location.jupyterlite.ipynb

Total successful Launches in the Launch sites

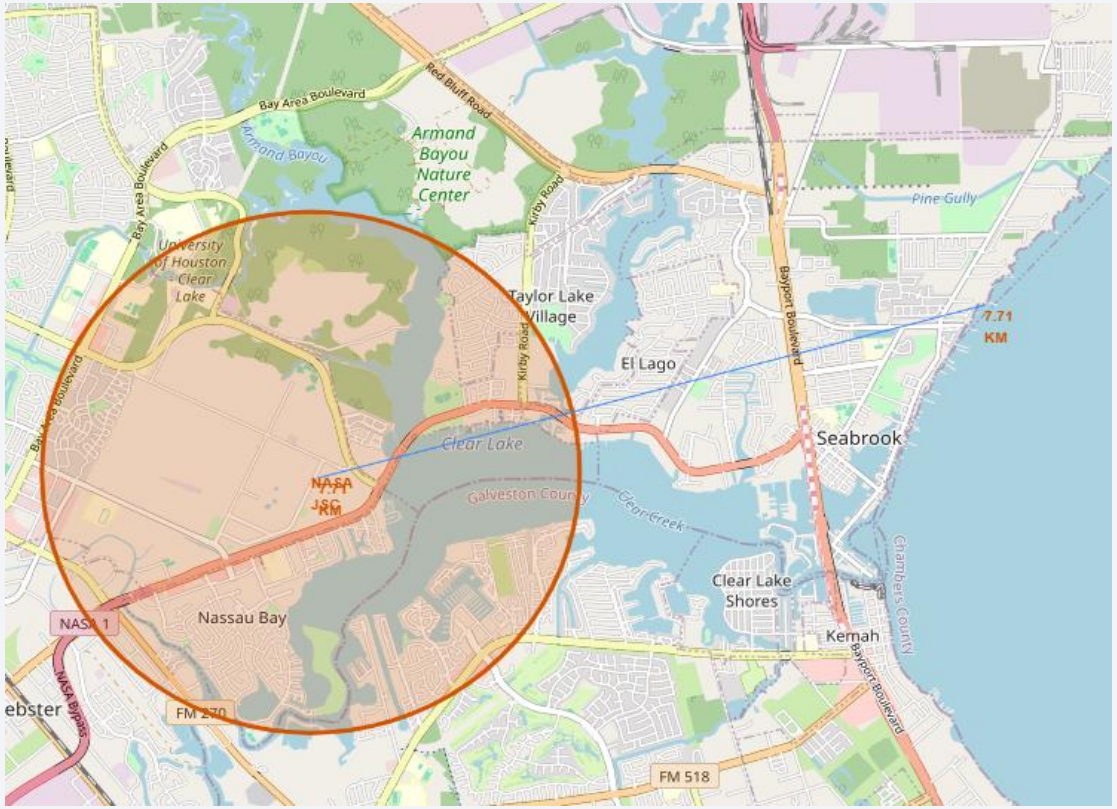
4 Successufl vs 6 Failaure in the west cost Launch site





The Notebook with all scripts is included in the Repository Data_Science Git lab_jupyter_launch_site_location.jupyterlite.ipynb

Nasa Juhanson Space Center Launch Site- Location Geo Infos-2

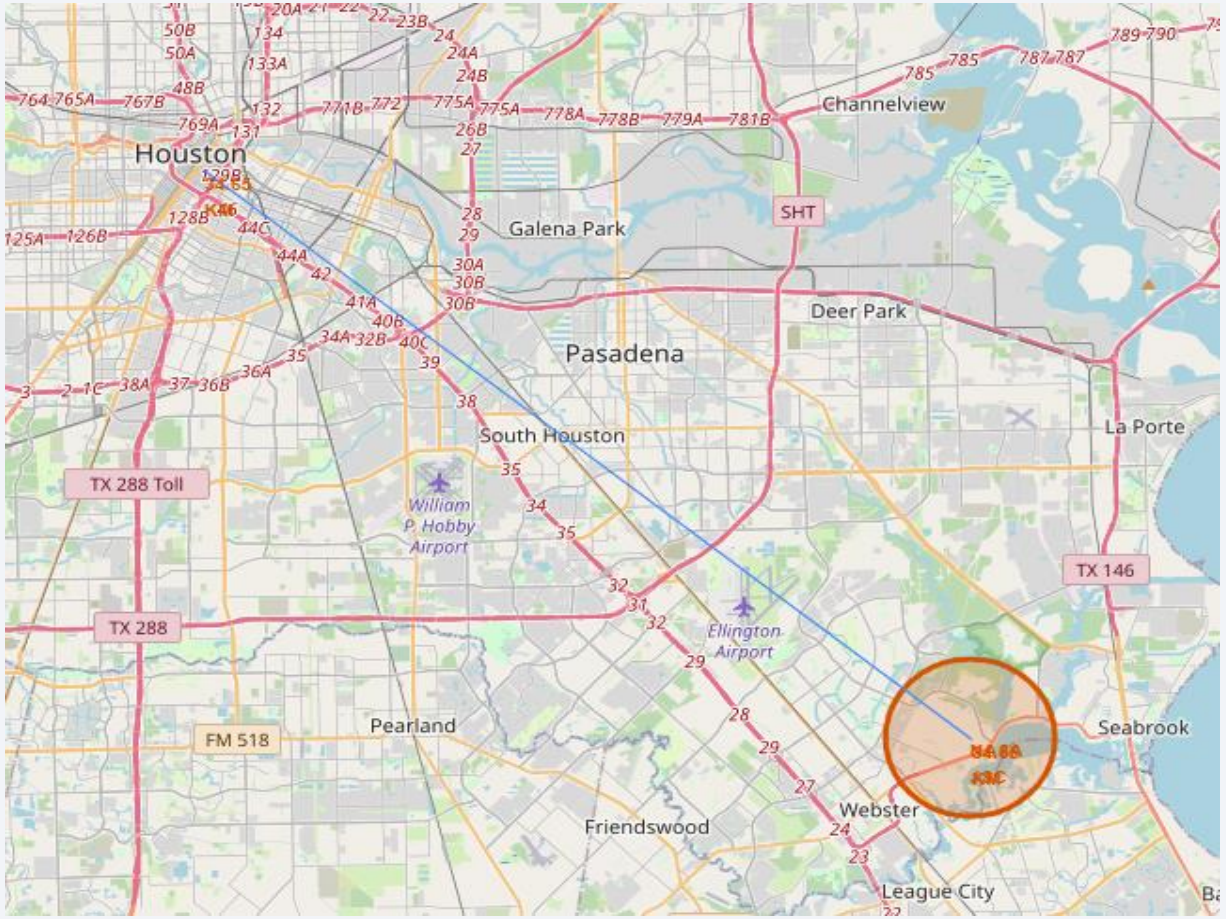


The Notebook with all scripts is included in the Repository Data_Science Git lab_jupyter_launch_site_location.jupyterlite.ipynb

Is the launch sites in close proximity to coastline?
Yes (7.71 km)

5.51 km the Distance between Nasa JSC and the closest Highway

Nasa Juhanson Space Center Launch Site - Location Geo Infos-2

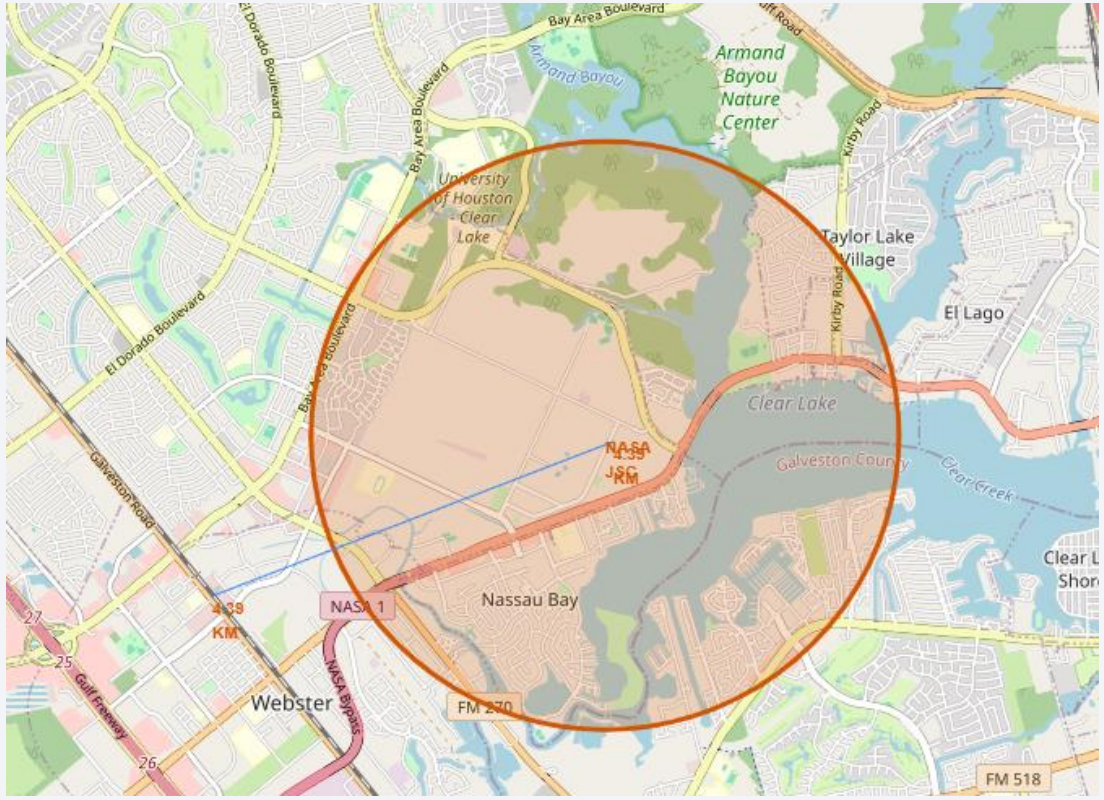


The Notebook with all scripts is included in the Repository Data_Science Git lab_jupyter_launch_site_location.jupyterlite.ipynb

Do launch site keep certain distance away from cities? No Housten center just 34.6 km

5.51 km the Distance between Nasa JSC and the closest Highway

Nasa Juhanson Space Center Launch Site- Location Geo Infos-2

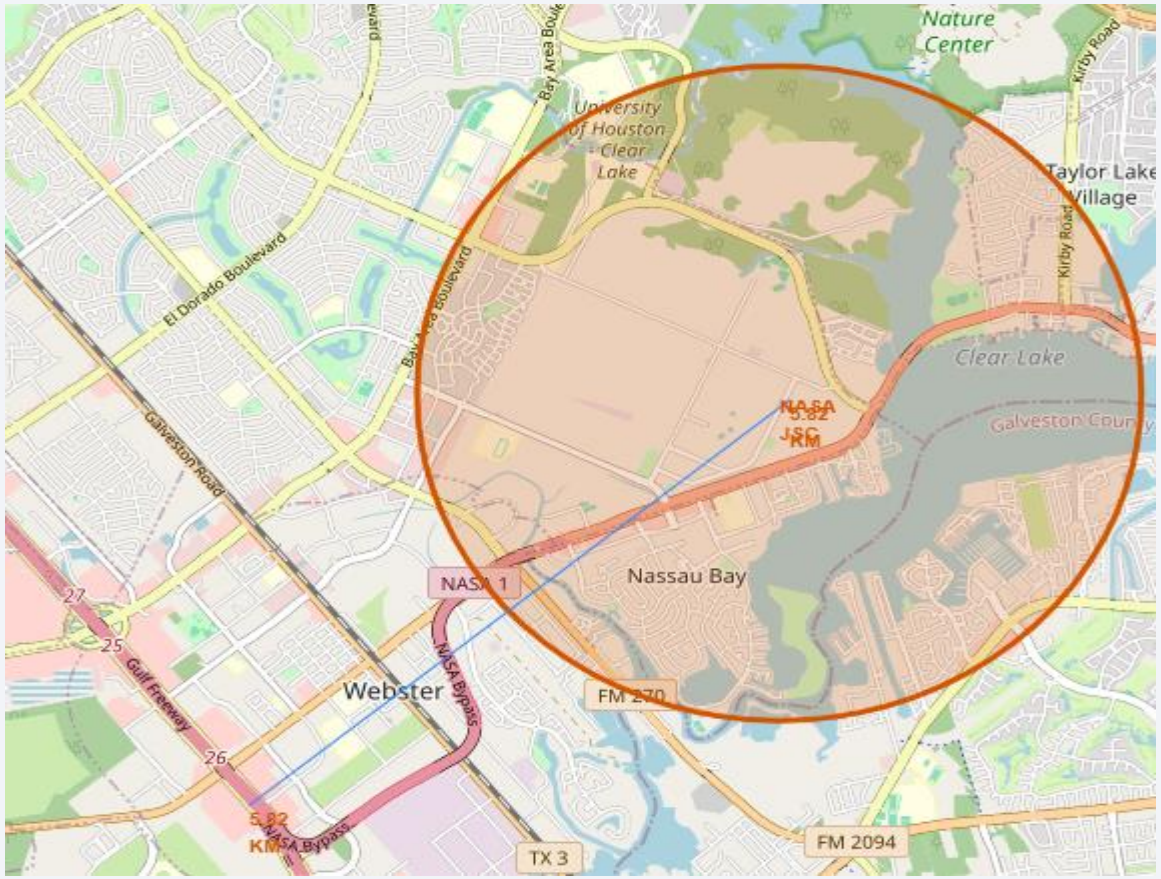


The Notebook with all scripts is included in the Repository Data_Science Git lab_jupyter_launch_site_location.jupyterlite.ipynb

•Is the launch sites in close proximity to railways?
Yes – just 4,39 km

5.51 km the Distance between Nasa JSC and the closest Highway

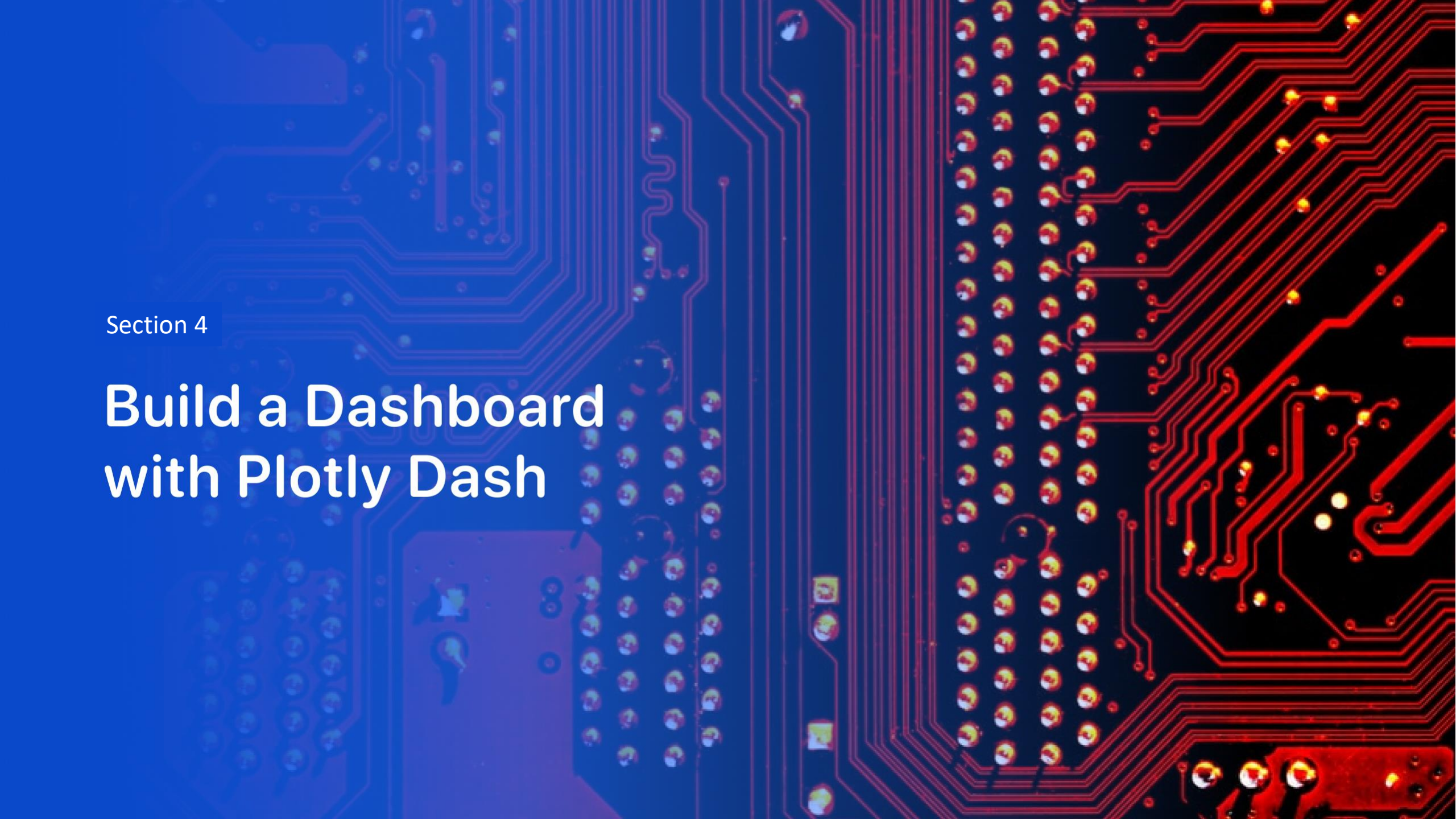
Nasa Juhanson Space Center Launch Site- Location Geo Infos-2



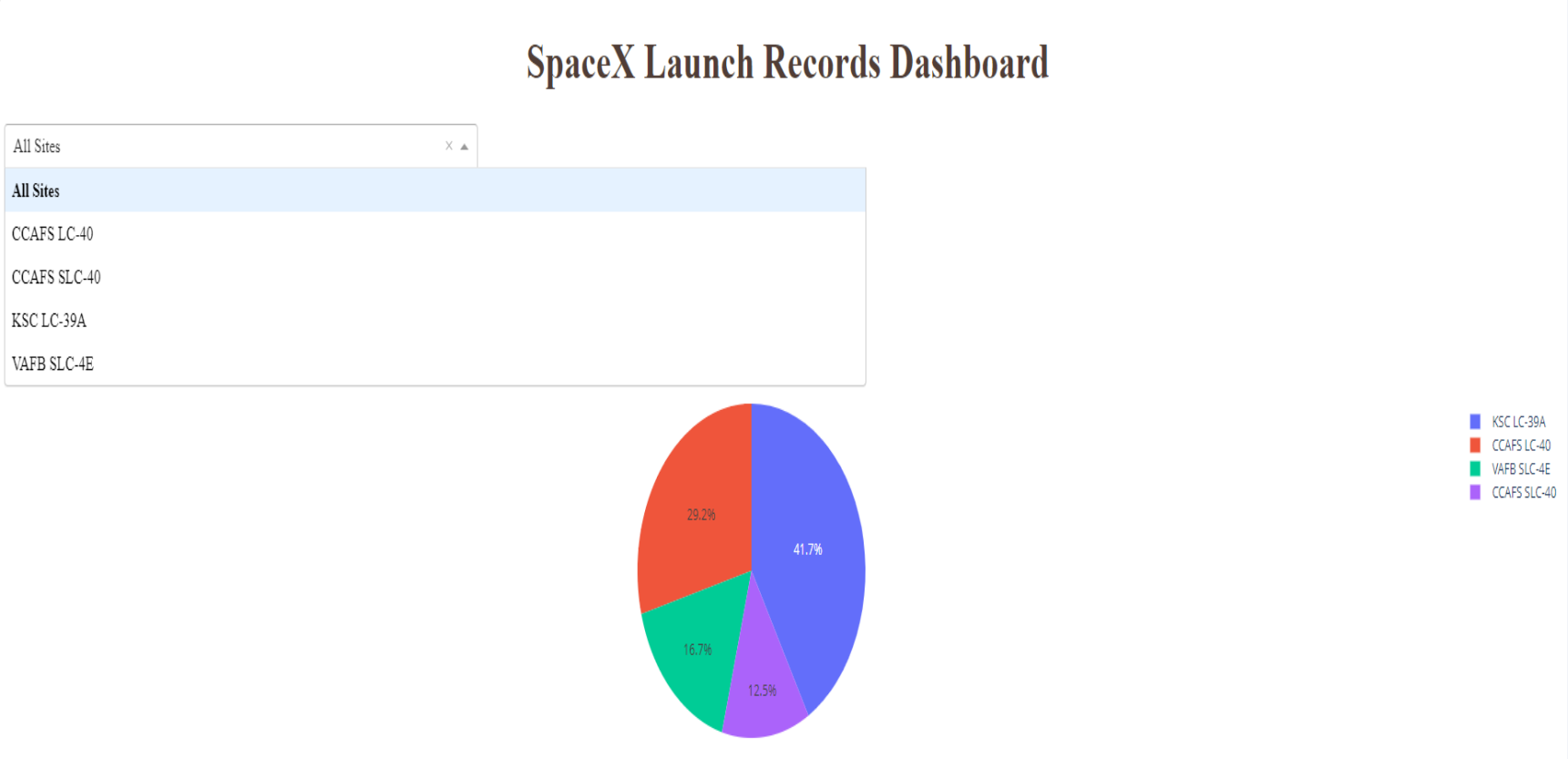
The Notebook with all scripts is included in the Repository Data_Science Git lab_jupyter_launch_site_location.jupyterlite.ipynb

•Is the launch site in close proximity to highways?
Yes – just 5,82 km

5.51 km the Distance between Nasa JSC and the closest Highway


Section 4
Build a Dashboard with Plotly Dash

SpaceX Launch Records Dashboard

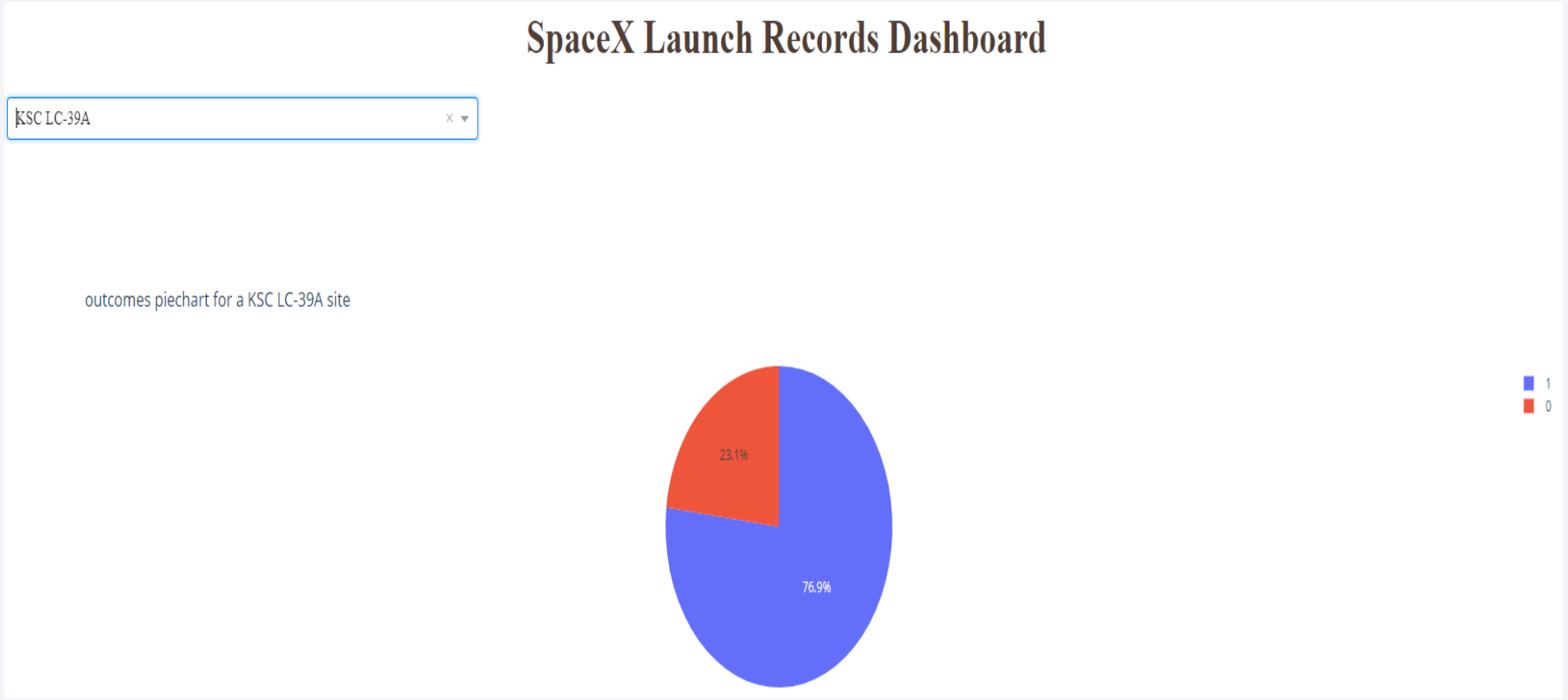


The Notebook and scripts are included in the Repository Data_Science Git

[spacex_dash_app.py]

Successs Rates Pro Site – the Best Success Launch is in KSC LC-39A

Which Site has the highest launch success Ratio

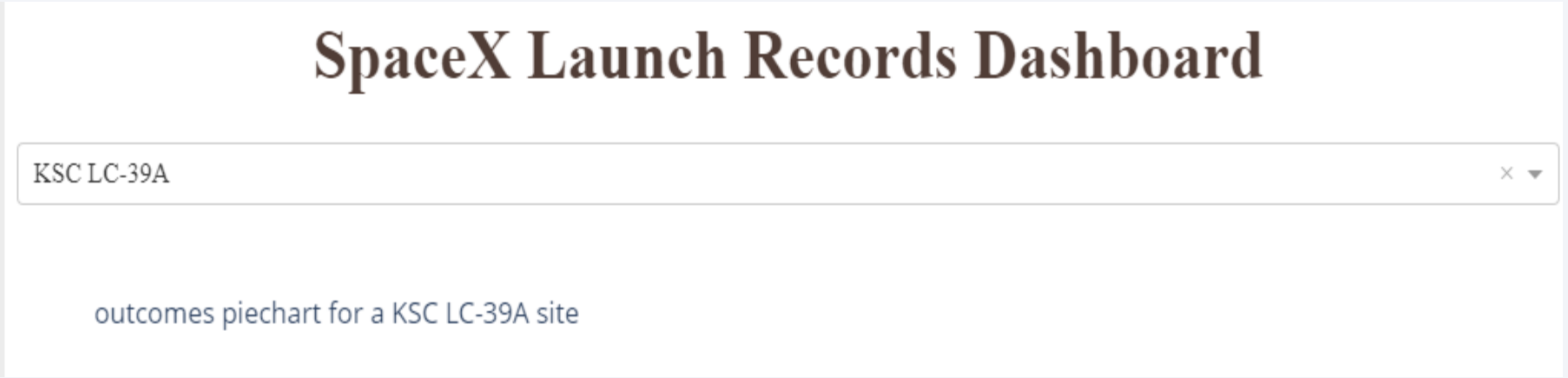


The Notebook and scripts are included in the Repository Data_Science Git

[spacex_dash_app.py]

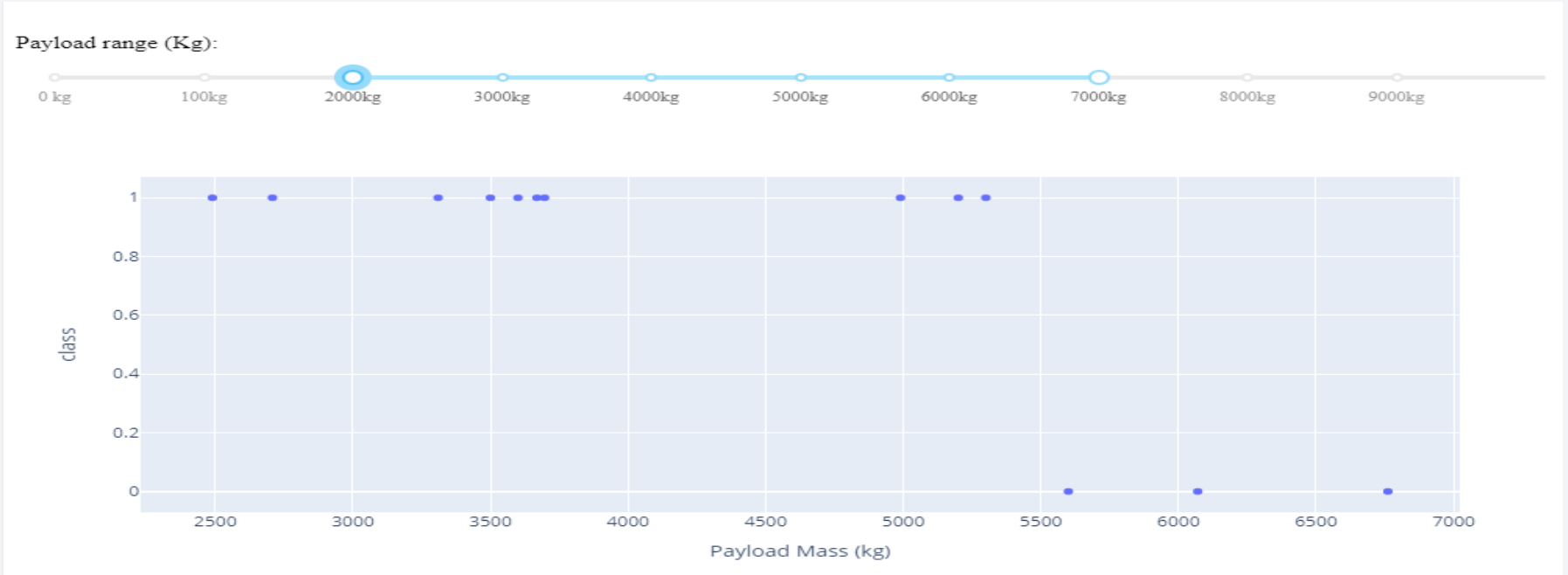
The Site KSC-LC-39A hast the most success Rate with about 77%

The Relationship between successful rate and payload mass



The Notebook and scripts are included in the Repository Data_Science Git

[spacex_dash_app.py]



For this Site All Heavy Payloads Launches above 5500 kg has been failed .

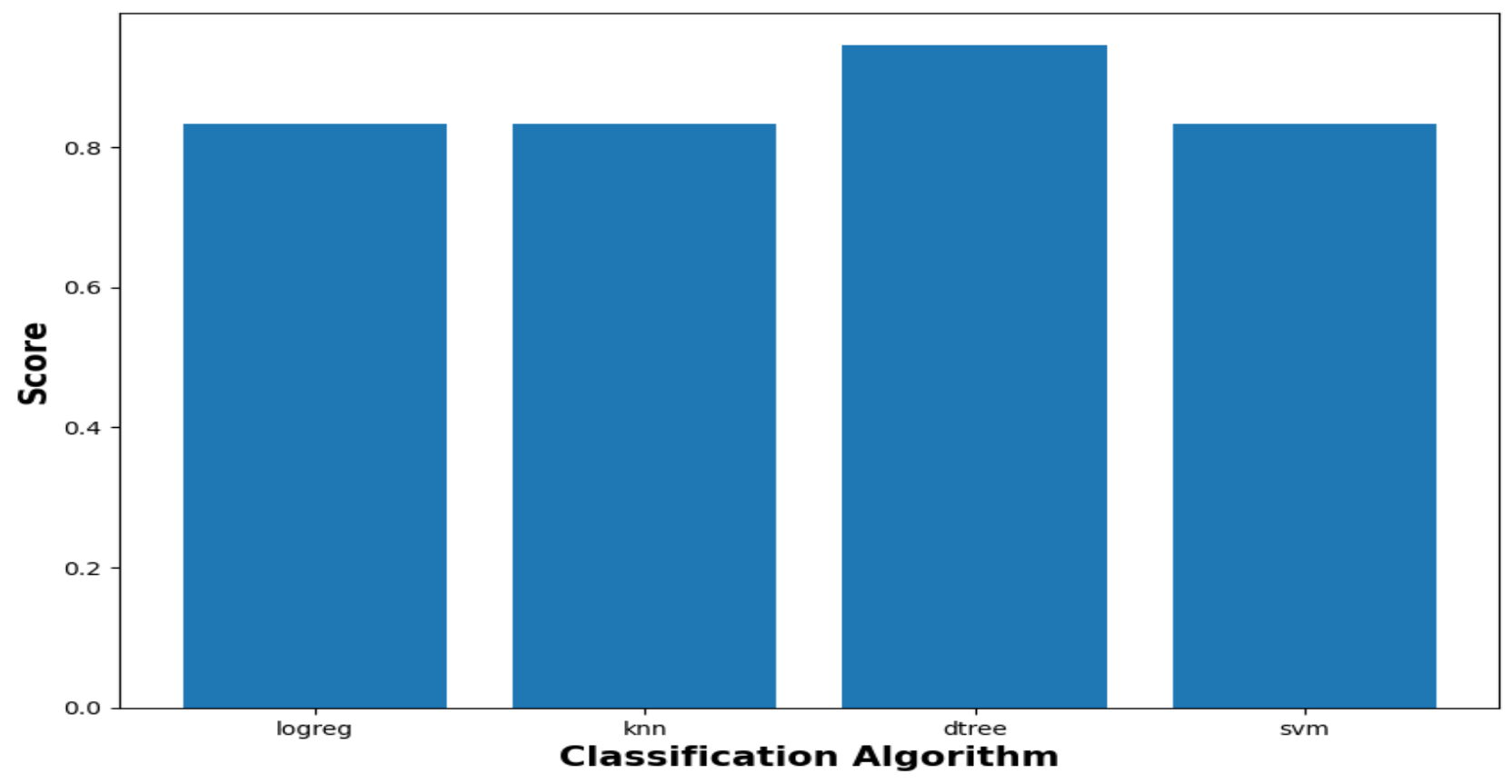
Section 5

Predictive Analysis (Classification)

Classification Purpose

- create a machine learning pipeline to predict if the first stage will land given the data from the preceding labs.
- Perform exploratory Data Analysis and determine Training Labels
- create a column for the class
- Standardize the data
- Split into training data and test data
- -Find best Hyperparameter for SVM, Classification Trees and Logistic Regression
- Find the method performs best using test data

Classification Accuracy



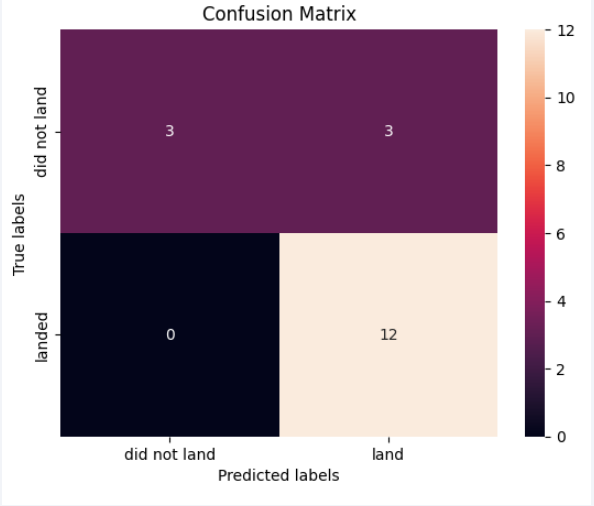
The Notebook and scripts are included in the Repository Data_Science Git

SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Decision Tree has the best performance with about 94% accuracy vs about 83-84% accuracy for other Classification Algorithms

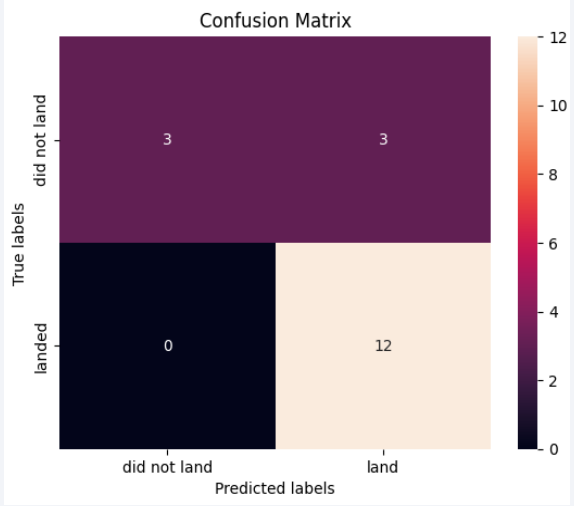
Confusion Matrix

```
yhat = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



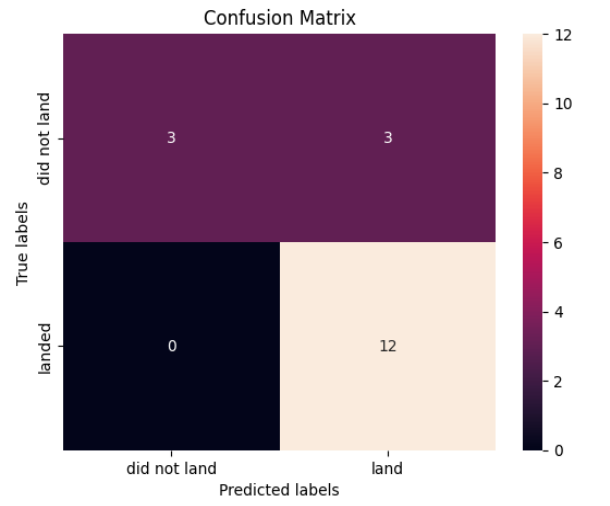
Knn

```
yhat=logreg_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



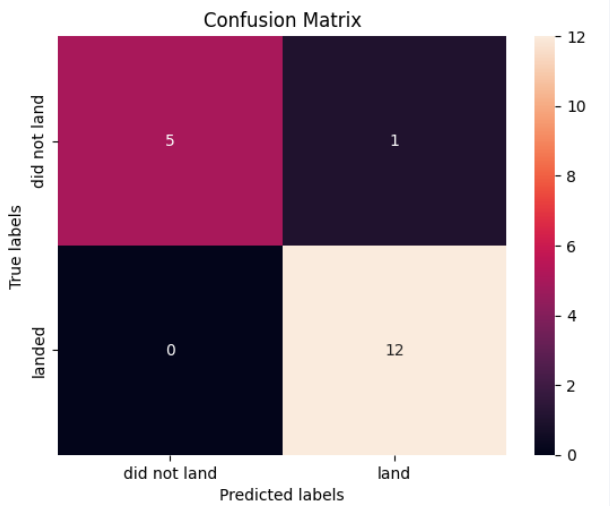
Logestic Regression

```
yhat=svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



SVM Regression

```
yhat = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



D-Tree Regression

The Notebook and scripts are included in the Repository Data_Science Git

SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Decision Tree matrix shows the best contrast in the confution matrix

Conclusions

The Accuracy of All used Classification Types was ok

Choosing the best parameters can change the Results

Decision Tree has the best performance with about 94% accuracy vs about 83-84% accuracy for other Classification Algorithms

Decision Tree matrix shows the best contrast in the confution matrix

The Notebook and scripts are included in the Repository Data_Science Git

SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Appendix

All scripts and exported csv Files are included in the Repository DataScience .

Reference

https://github.com/mazenhat/Data_Science.git

Thank you!