Data Science - The New Space Race

Sepulveda, Eduardo

2023-01-02

IBM Developer

SKILLS NETWORK

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion
- Acknowledgment

Executive Summary

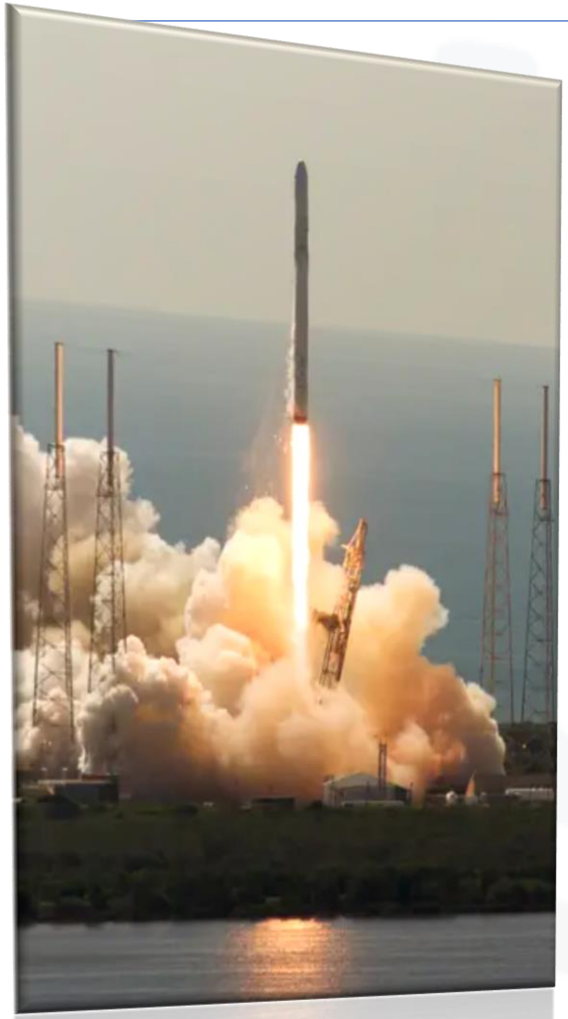
✓Summary of methodologies

- Data collection
- Data wrangling
- EDA with data visualization
- EDA with SQL
- Building an interactive map with Folium
- Building a Dashboard with Plotly Dash
- Predictive analysis (Classification)

✓Summary of all results

- EDA results
- Interactive analytics
- Predictive analysis

Introduction

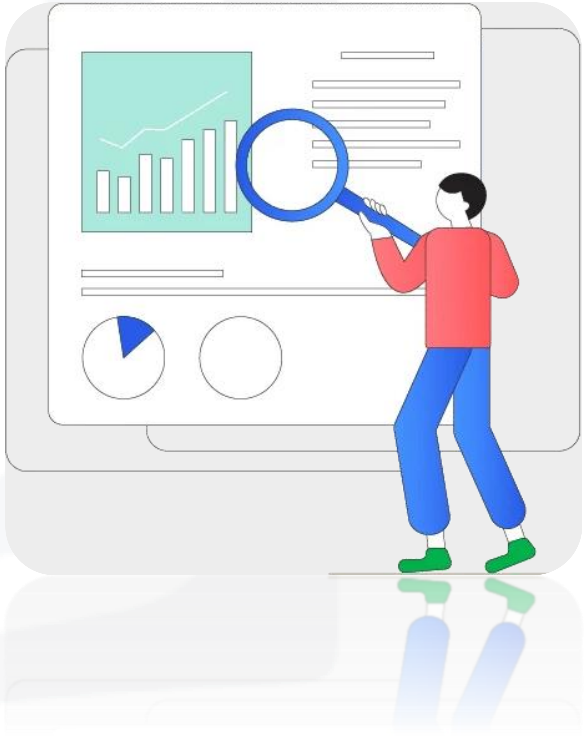


SpaceX is trying to change the space exploration game. The Falcon 9 is the world's first orbital-class reusable rocket. Reuse allows SpaceX to reflect the most expensive parts of the rocket, which in turn reduces the cost of accessing space. Part of that will be improving data analysis and integration processes and tools to make the business faster and better. This is where Data Science comes in.

Methodology

Executive Summary

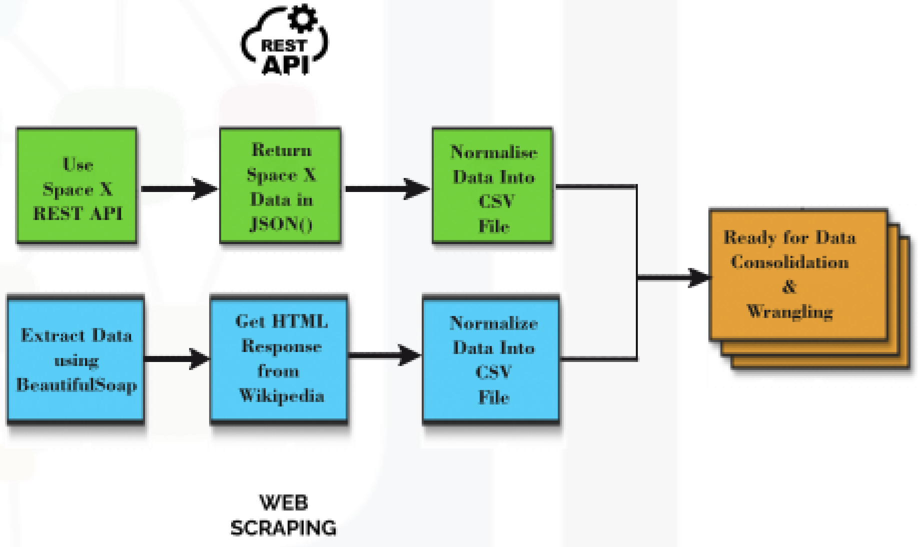
- Data collection methodology:
  - SpaceX Rest API
  - Web Scrapping from Wikipedia
- Perform data wrangling
  - One Hot Ecoding data fields for Machine Learning and data cleaning of null values and irrelevant columns
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interative visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - LR, KNN, SVM, DT models have been built and evaluated for the best classifier



Data Collection

✓ The following datasets was collected:

- Space launch data that gathered from the Space X Reset API.
- We will be working with SpaceX launch data collected from an API, SPECEX REST API. It will provide us with data on launches, including information on the rocket used, payload delivered, launch specifications, landing specifications and landing result.
- The SpaceX Rest API endpoints, or URL, starts with **api.spacexdata.com/v4/**
- Wikipedia another popular data source for obtaining Falcon 9 BeautifulSoup



Data Collection SpaceX API

In [9]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-c
```

Request and parse the SpaceX launch data using the GET request

In [10]:
```
response.status_code
```

Out[10]: 200

We should see that the request was successfull with the 200 status response code

In [14]:
```
# Use json_normalize meethod to convert the jso

data_df = pd.json_normalize(response.json())
```

Now we decode the response content as a Json using .json() and turn it into a Pandas dataframe using .json_normalize()

In [16]:
```
# Lets take a subset of our dataframe keeping only the features we want and the flight
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extr
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value i
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.

We can now export it to a CSV

In [49]:
```
#Saving the file
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

https://github.com/ENSS1971/Lab1_Collecting/blob/master/jupyter-labs-spacex-data-collection-api.ipynb

Data Collection SpaceX WebScraping

```
In [4]:
static_url = "https://en.wikipedia.org/w/index.php?t
```

Request the Falcon9 Launch Wiki page from its URL

```
In [11]:
# use requests.get() method with the provided static_url
response = requests.get(static_url)
# assign the response to a object
#print(response.content)
data_html = response.text
```

Create a BeautifulSoup object from the HTML response

```
In [12]:
# Use BeautifulSoup() to create a BeautifulSoup c
soup = BeautifulSoup(data_html,"html.parser")
```

Create a data frame by parsing the launch HTML tables

```
In [14]:
# Use the find_all function in the Beautifu
# Assign the result to a list called `html_
html_tables = soup.find_all('table')
```

Extract all column/variable names from the HTML table header

```
In [17]:
column_names = []

# Apply find_all() function with `th` element on first_
# Iterate each th element and apply the provided extrac
# Append the Non-empty column name (`if name is not Nor

table_headers = first_launch_table.find_all('th')

# print(table_headers)
for j, table_header in enumerate(table_headers):
    name = extract_column_from_header(table_header)
    if name is not None and len(name) > 0:
        column_names.append(name)
```

Extract column name one by one

https://github.com/ENSS1971/Lab1_Collecting/blob/master/jupyter-labs-webscraping.ipynb

Data Collection SpaceX WebScraping

In [25]:
```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheader
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a
        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:
```

Formatting dataframe data

Finally, conveting dictionary to dataframe.

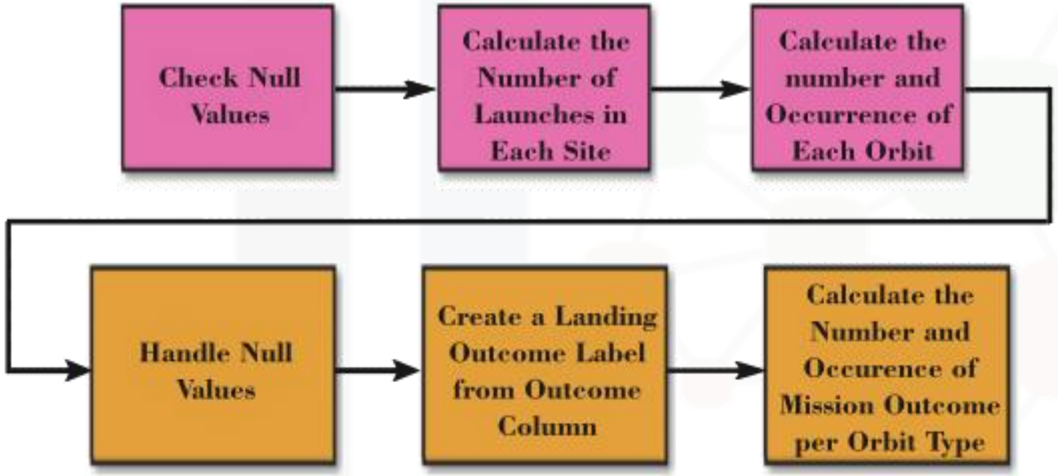
In [28]:
```
# First Stage Landing Prediction Dataframe
fslp_df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
```

In [29]:
```
#Saving the file
fslp_df.to_csv('spacex_web_scraped.csv', index=False)
```

We can now export it to a CSV

https://github.com/ENSS1971/Lab1_Collecting/blob/master/jupyter-labs-webscraping.ipynb

Data Collection Data Wrangling

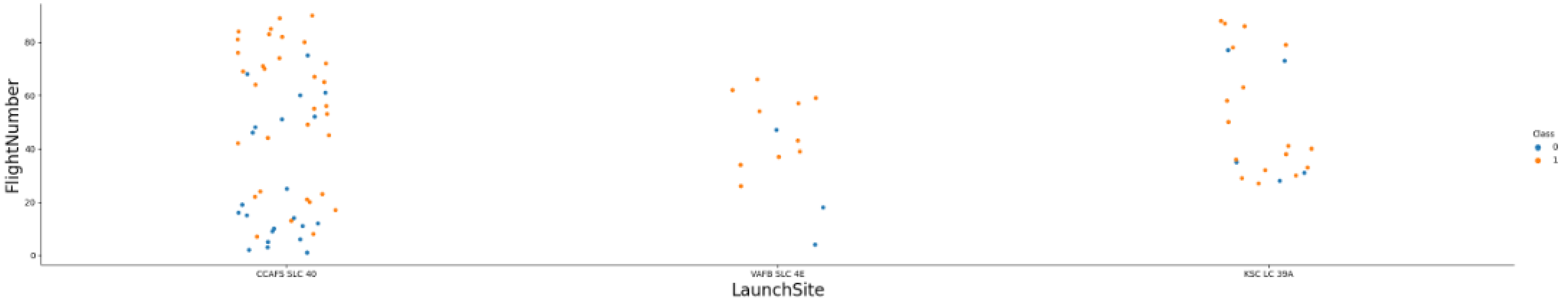


Exploratory Data Analysis (EDA)

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns,to spot anomalies,to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

It is a good practice to understand the data first and try to gather as many insights from it. EDA is all about making sense of data in hand, before getting them dirty with it.

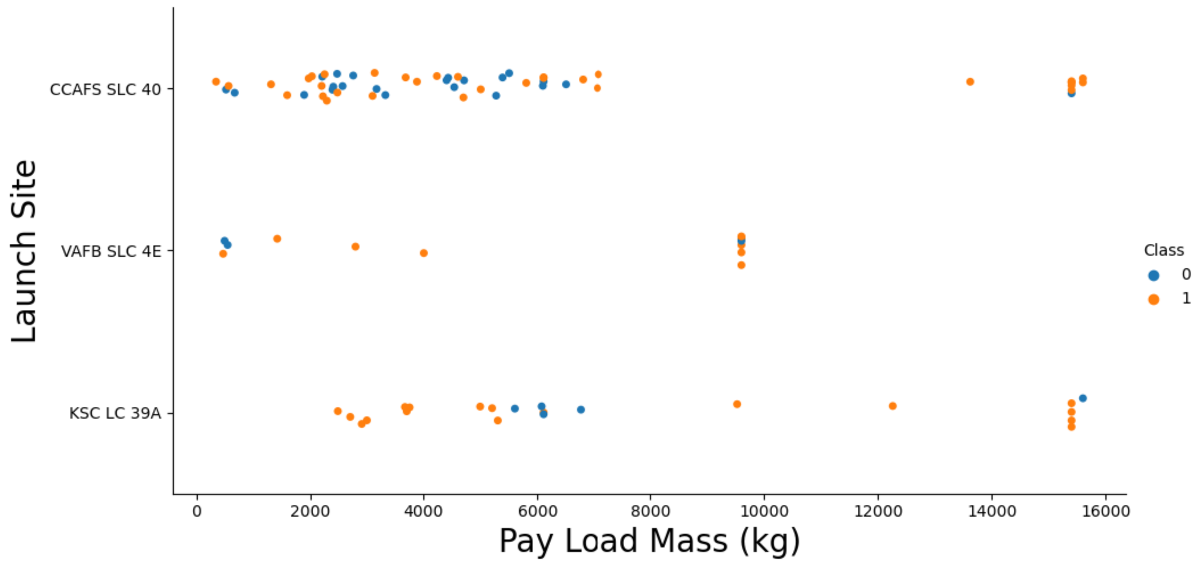
Exploratory Data Analysis



CCAFS SLC 40 website releases are significantly larger than other releases.

https://github.com/ENSS1971/Lab1_Collecting/blob/master/IBM-DS0321EN-SkillsNetwork_labs_module_2_jupyter-labs-eda-dataviz.ipynb.jupyterlite%20(1).ipynb

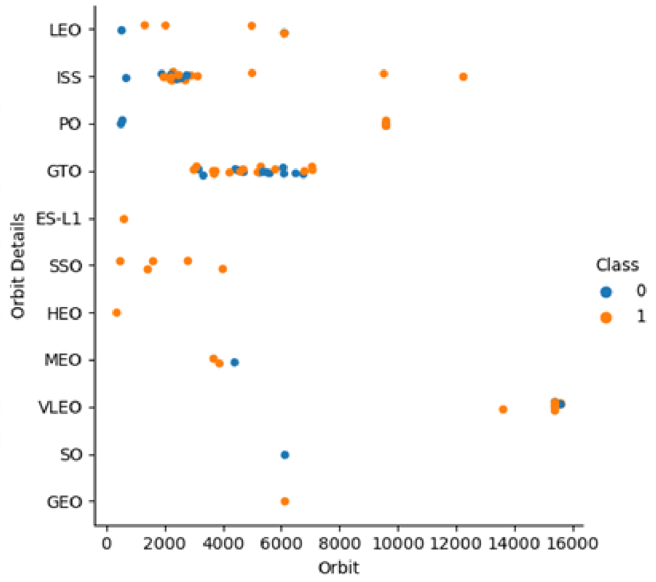
Exploratory Data Analysis



Most launches with lower mass were launched from CCAFS SLC 40.

https://github.com/ENSS1971/Lab1_Collecting/blob/master/IBM-DS0321EN-SkillsNetwork_labs_module_2_jupyter-labs-eda-dataviz.ipynb.jupyterlite%20(1).ipynb

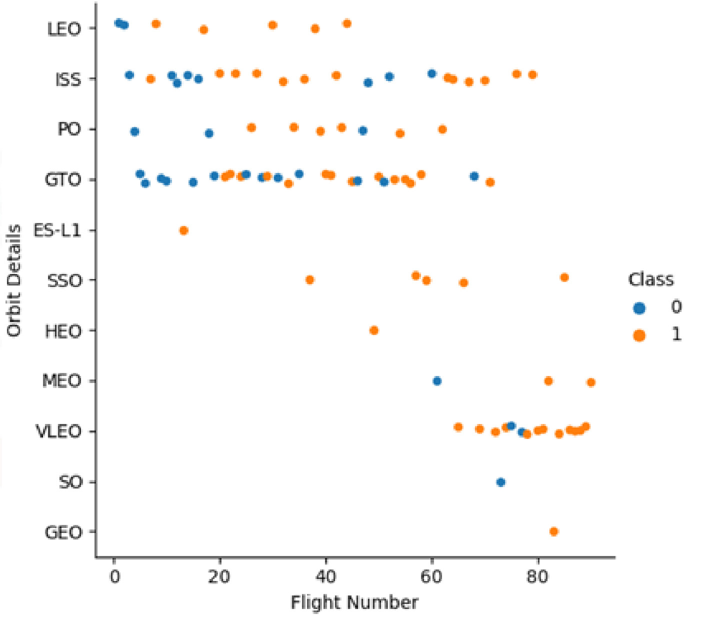
Exploratory Data Analysis



Most satellite launches went to the GTO orbit in the range between 4000km and 8000km altitude

https://github.com/ENSS1971/Lab1_Collecting/blob/master/IBM-DS0321EN-SkillsNetwork_labs_module_2_jupyter-labs-eda-dataviz.ipynb.jupyterlite%20(1).ipynb

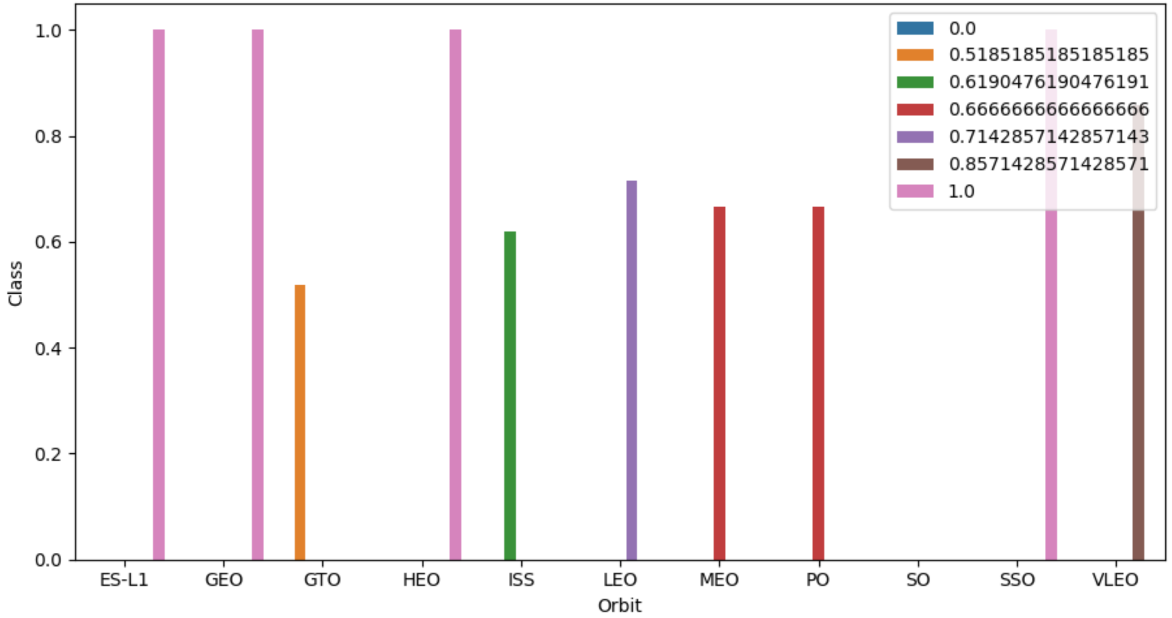
Exploratory Data Analysis



You should see that in LEO orbit Success appears related to the number of flights; on the other hand, in recent years there has been an increase in launches in the VLEO orbit

https://github.com/ENSS1971/Lab1_Collecting/blob/master/IBM-DS0321EN-SkillsNetwork_labs_module_2_jupyter-labs-eda-dataviz.ipynb.jupyterlite%20(1).ipynb

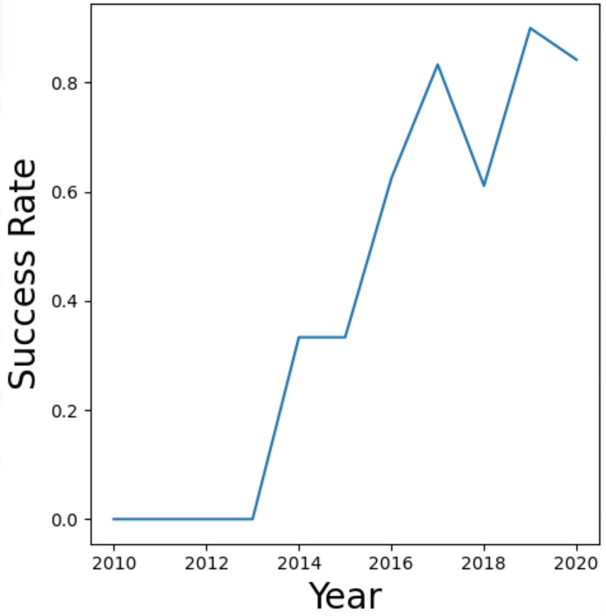
Exploratory Data Analysis



The orbit types of ES-L1, GEO, HEO and SSO are among the successful launches

https://github.com/ENSS1971/Lab1_Collecting/blob/master/IBM-DS0321EN-SkillsNetwork_labs_module_2_jupyter-labs-eda-dataviz.ipynb.jupyterlite%20(1).ipynb

Exploratory Data Analysis



The orbit types of ES-L1, GEO, HEO and
SSO are among the successful launches

https://github.com/ENSS1971/Lab1_Collecting/blob/master/IBM-DS0321EN-SkillsNetwork_labs_module_2_jupyter-labs-eda-dataviz.ipynb.jupyterlite%20(1).ipynb

EDA with SQL

✓ SQL queries performed include:

- Displaying the names of the unique launch sites in the space mission
- Displaying 5 records where launch sites begin with the string ‘
- Displaying the total payload mass carried by boosters launched by NASA (CRS)
- Displaying average payload mass carried by booster version F9 v1.1
- Listing the date where the successful landing outcome in drone ship was achieved.
- Listing the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000
- Listing the names of the booster_versions which have carried the maximum payload mass.
- Listing the records which will display the month names, successful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2015

https://github.com/ENSS1971/Lab1_Collecting/blob/master/jupyter-labs-eda-sql-coursera_sqllite.ipynb

EDA with SQL

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

In [30]:
```
%sql select distinct launch_site from spacextbl
```

```
 * ibm_db_sa://qwz86117:***@fbd88901-ebdb-4a4f-a32e-9
Done.
```

Out[30]:

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

https://github.com/ENSS1971/Lab1_Collecting/blob/master/jupyter-labs-eda-sql-coursera_sqllite.ipynb

EDA with SQL

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

In [36]:
```
%sql select * from spacextbl where launch_site like 'CCA%' fetch first 5 row only
```

```
 * ibm_db_sa://qwz86117:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Out[36]:

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	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	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

https://github.com/ENSS1971/Lab1_Collecting/blob/master/jupyter-labs-eda-sql-coursera_sqllite.ipynb

EDA with SQL

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

```
In [39]:  1  %sql select sum(payload_mass__kg_) from spacextbl where customer = 'NASA (CRS)'

 * ibm_db_sa://qwz86117:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde0(
Done.

Out[39]:
```

1
45596

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

```
In [40]:  1  %sql select avg(payload_mass__kg_) from spacextbl where booster_version = 'F9 v1.1'

 * ibm_db_sa://qwz86117:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.da
Done.

Out[40]:
```

1
2928

https://github.com/ENSS1971/Lab1_Collecting/blob/master/jupyter-labs-eda-sql-coursera_sqllite.ipynb

EDA with SQL

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

```
In [43]:  1  %sql select min(DATE) from spacextbl where landing__outcome = 'Success (ground pad)'

 * ibm_db_sa://qwz86117:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.dat
Done.

Out[43]:
```

1
2015-12-22

https://github.com/ENSS1971/Lab1_Collecting/blob/master/jupyter-labs-eda-sql-coursera_sqllite.ipynb

EDA with SQL

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

In [45]:

```
%sql select booster_version from spacextbl where landing__outcome ='Success (drone ship)'  and payload_mass__kg_ between 4000 and 6000
```

```
 * ibm_db_sa://qwz86117:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.databases.appdomain.cloud:32731/bludb
Done.
```

Out[45]:

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

https://github.com/ENSS1971/Lab1_Collecting/blob/master/jupyter-labs-eda-sql-coursera_sqllite.ipynb

EDA with SQL

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

In [48]:
```
%sql select distinct booster_version from spacextbl where payload_mass__kg_ = (select max(payload_mass__kg_) FROM spacextbl)
```

```
 * ibm_db_sa://qwz86117:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Out[48]:

booster_version
F9 B5 B1048.4
F9 B5 B1048.5
F9 B5 B1049.4
F9 B5 B1049.5
F9 B5 B1049.7
F9 B5 B1051.3
F9 B5 B1051.4
F9 B5 B1051.6
F9 B5 B1056.4
F9 B5 B1058.3
F9 B5 B1060.2
F9 B5 B1060.3

https://github.com/ENSS1971/Lab1_Collecting/blob/master/jupyter-labs-eda-sql-coursera_sqllite.ipynb

EDA with SQL

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.

In [54]:
```
%sql select month(DATE),mission_outcome, booster_version, launch_site from spacextbl where extract(year from date)='2015';
```

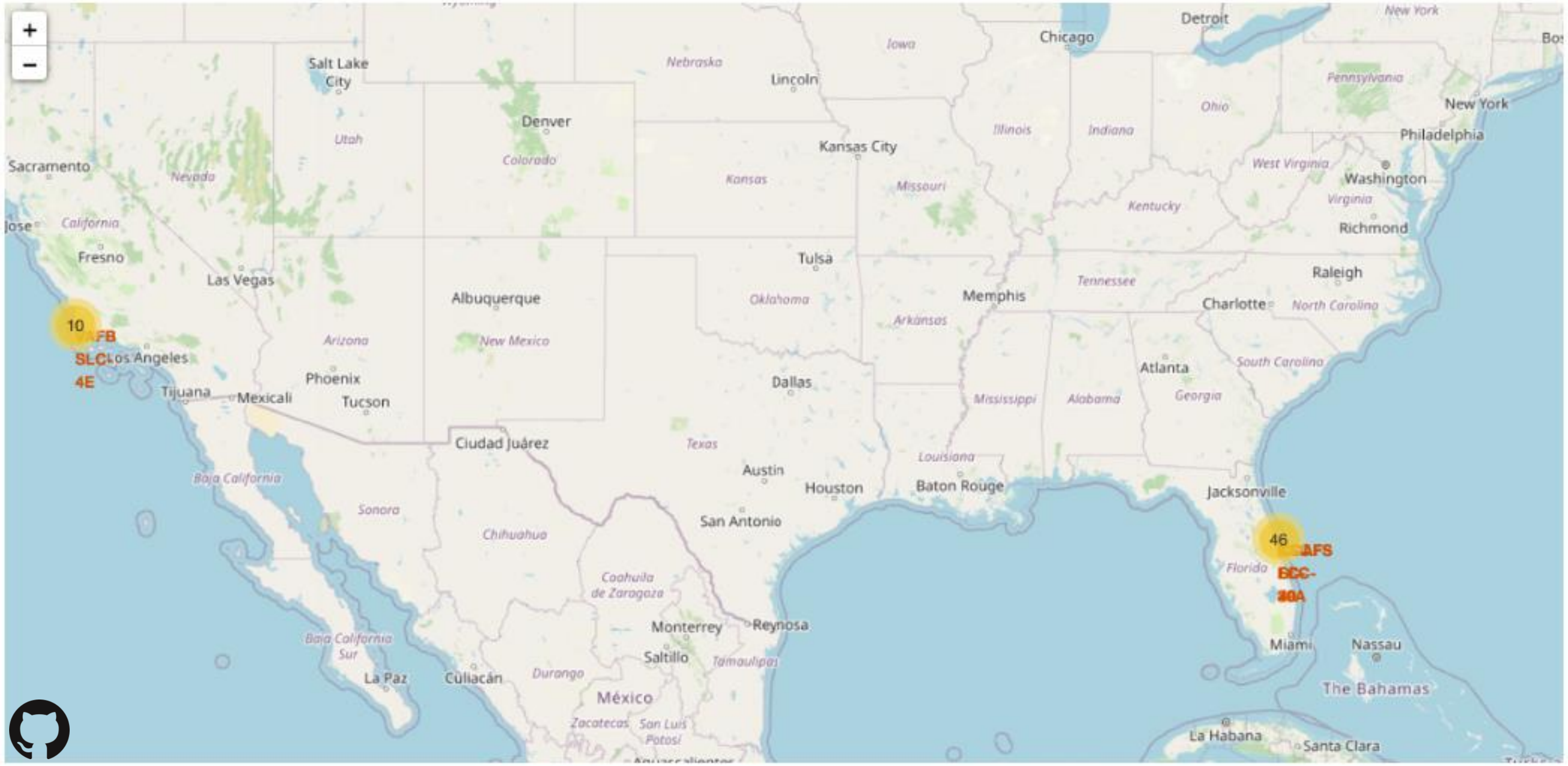
```
 * ibm_db_sa://qwz86117:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu01qde00.databases.appdomain.cloud:32731/bludb
Done.
```

Out[54]:

1	mission_outcome	booster_version	launch_site
1	Success	F9 v1.1 B1012	CCAFS LC-40
2	Success	F9 v1.1 B1013	CCAFS LC-40
3	Success	F9 v1.1 B1014	CCAFS LC-40
4	Success	F9 v1.1 B1015	CCAFS LC-40
4	Success	F9 v1.1 B1016	CCAFS LC-40
6	Failure (in flight)	F9 v1.1 B1018	CCAFS LC-40
12	Success	F9 FT B1019	CCAFS LC-40

https://github.com/ENSS1971/Lab1_Collecting/blob/master/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium



Map with location of successful/failed launches for each site on the map

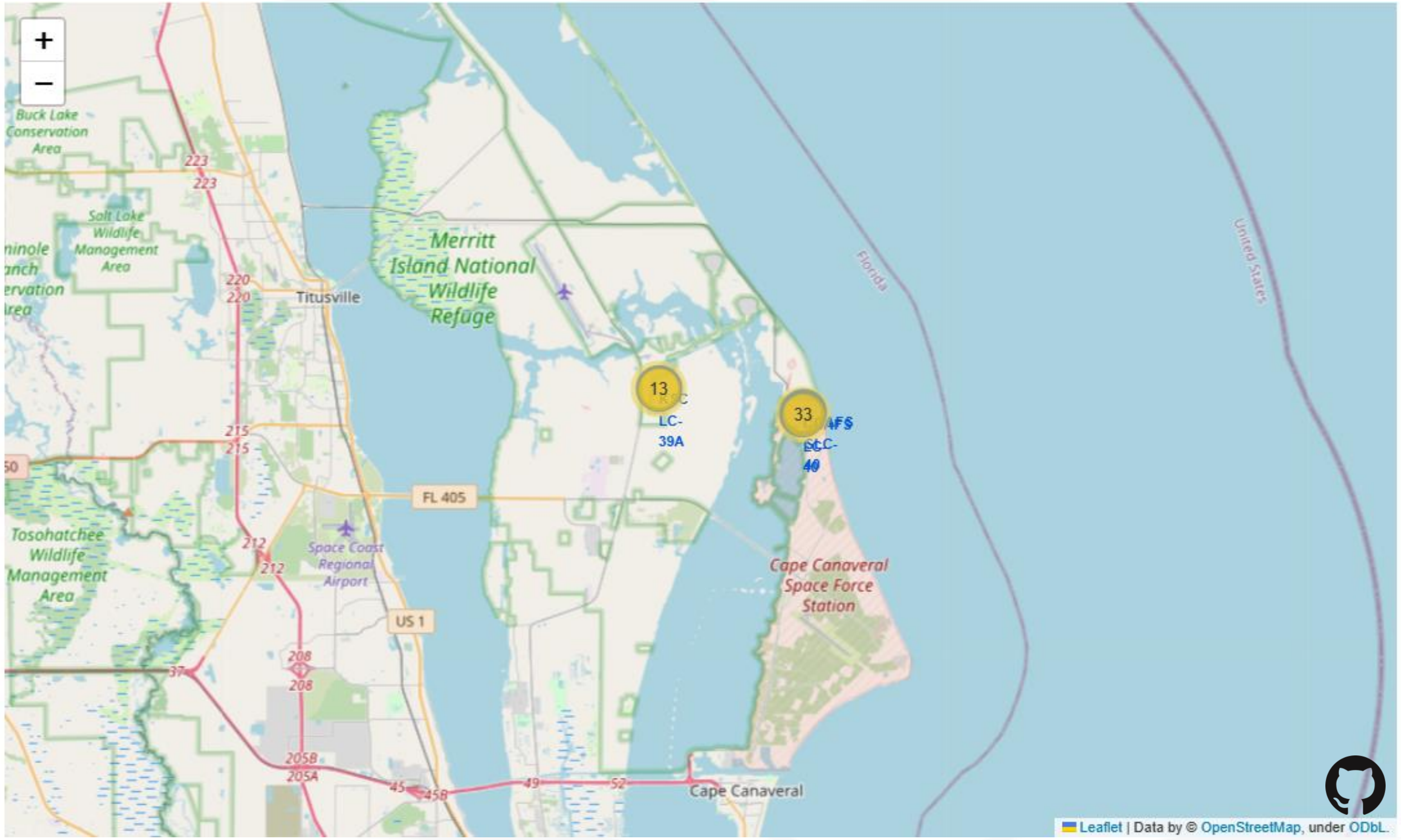
https://github.com/ENSS1971/Lab1_Collecting/blob/master/lab_jupyter_launch_site_location.ipynb

IBM Developer

SKILLS NETWORK

Build an Interactive Map with Folium

Out[13]:

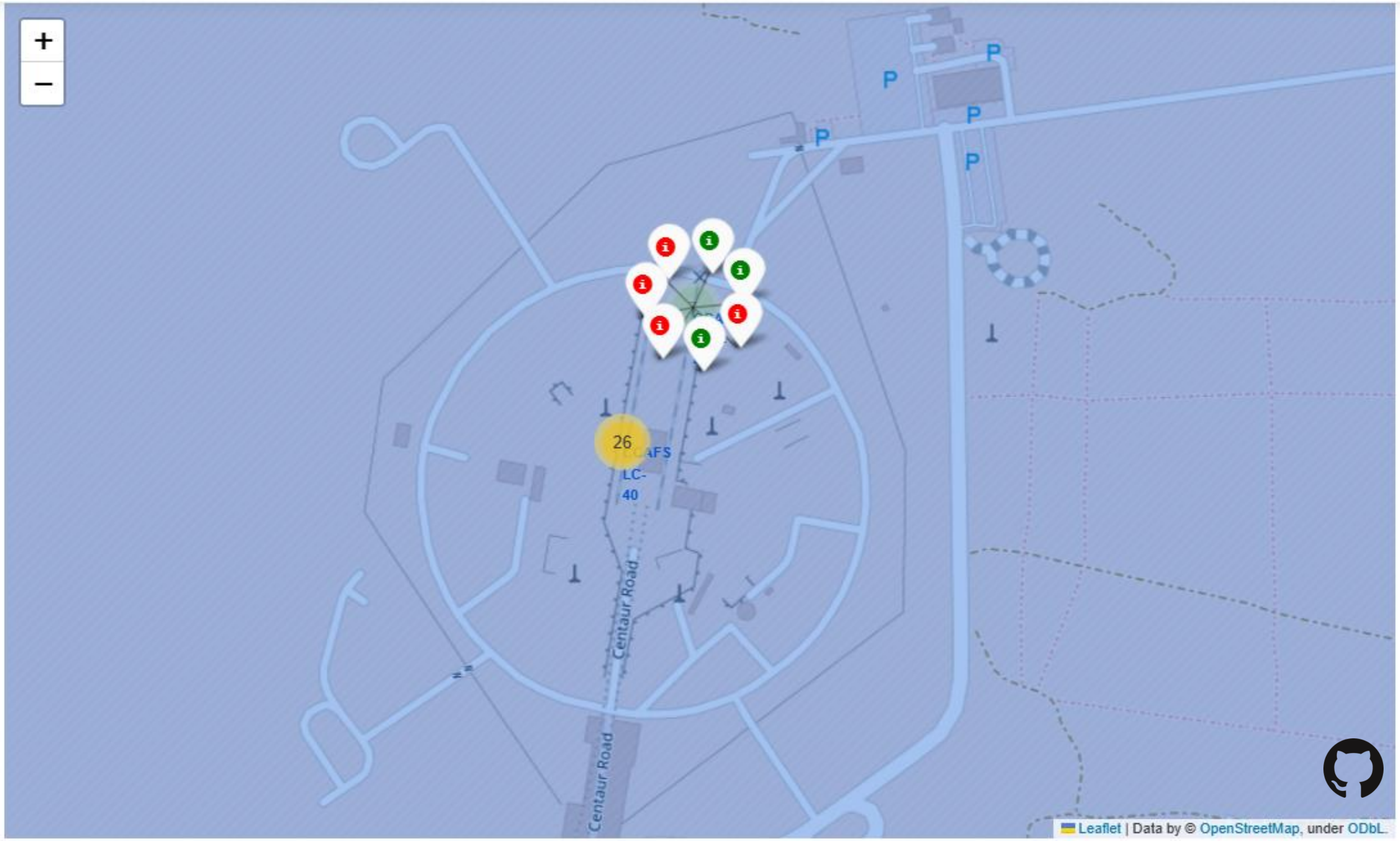


Zooming in on Florida, we will identify the two launch pads CCAFS LC-40 and CCAFS SLC-40

https://github.com/ENSS1971/Lab1_Collecting/blob/master/lab_jupyter_launch_site_location.ipynb

Build an Interactive Map with Folium

Out[13]:



By clicking on the CCAFS SLC-40 launch base, we can check the successful and failed launches

https://github.com/ENSS1971/Lab1_Collecting/blob/master/lab_jupyter_launch_site_location.ipynb

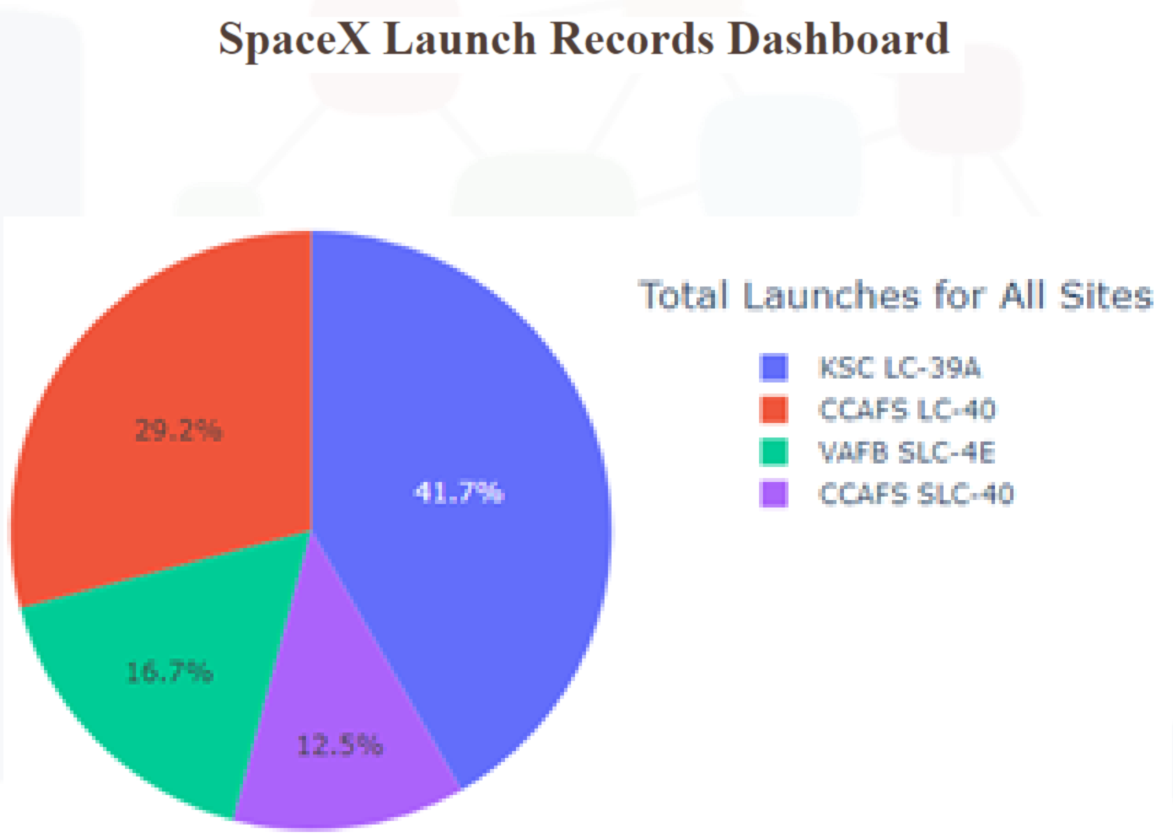
Build an Interactive Map with Folium

Out[22]:

The distance between a launch site and the selected point on the coast

https://github.com/ENSS1971/Lab1_Collecting/blob/master/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

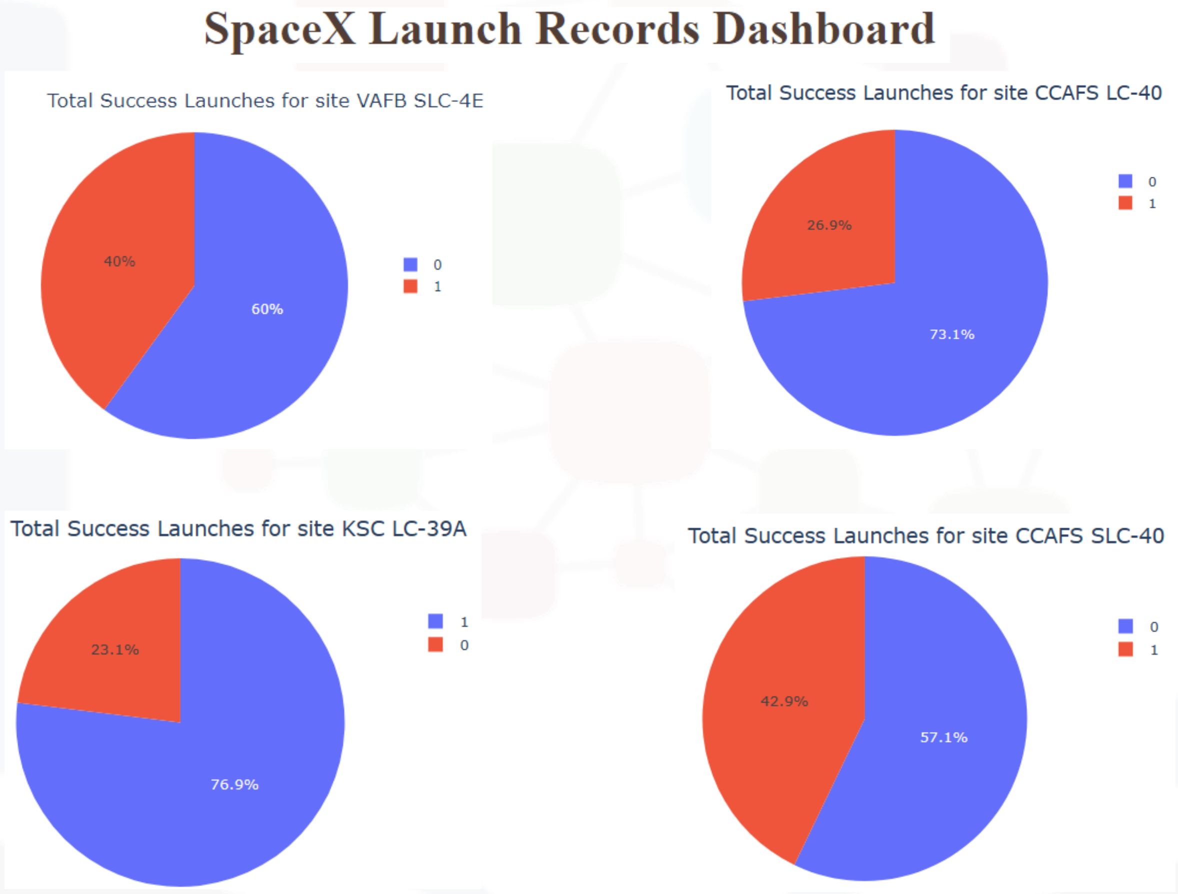


https://github.com/ENSS1971/Lab1_Collecting/blob/master/Build_a_Dashboard_with_Plotly_%20Dash_JupyterLab.ipynb

IBM Developer

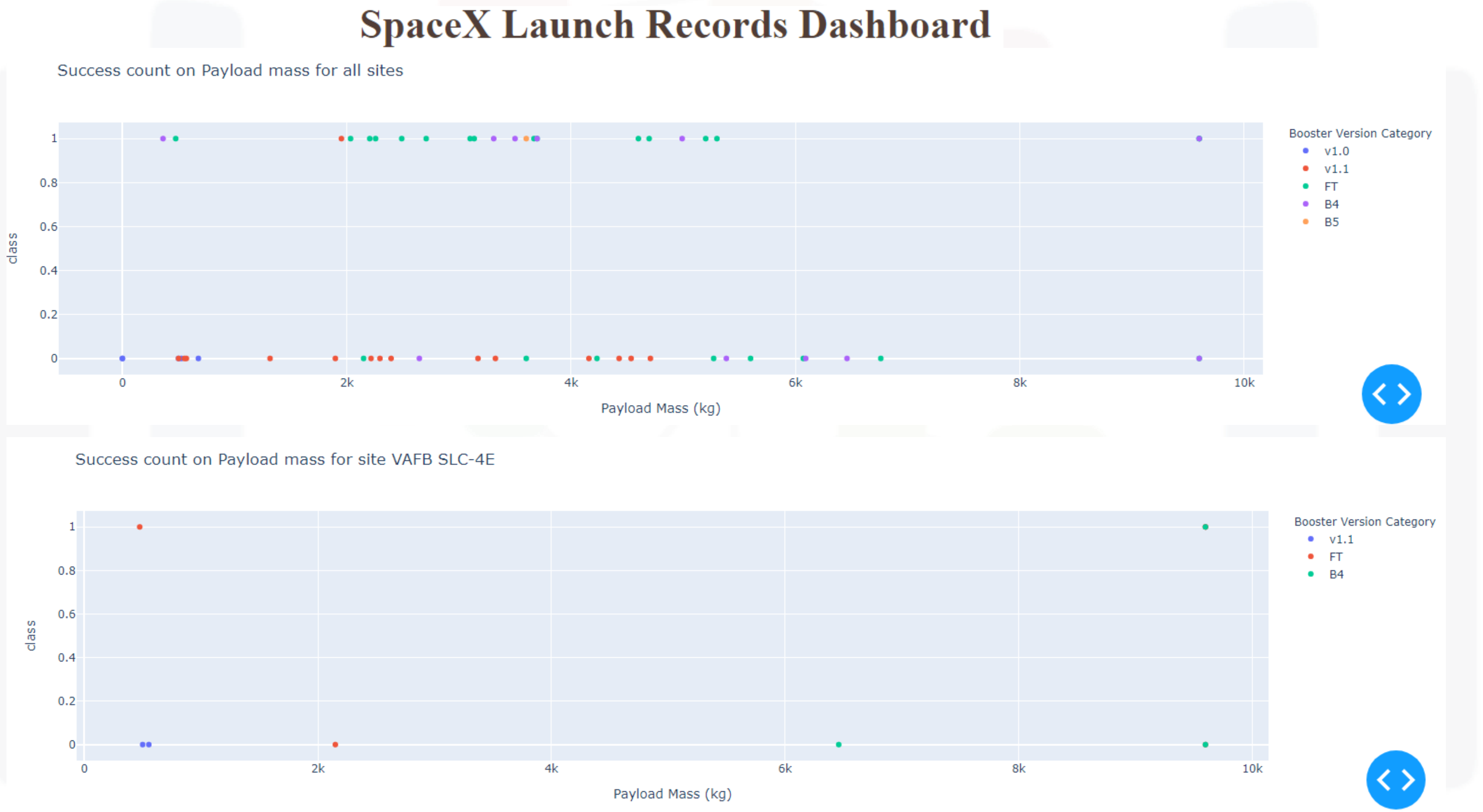
SKILLS NETWORK

Build a Dashboard with Plotly Dash



https://github.com/ENSS1971/Lab1_Collecting/blob/master/Build_a_Dashboard_with_Plotly_%20Dash_JupyterLab.ipynb

Build a Dashboard with Plotly Dash



https://github.com/ENSS1971/Lab1_Collecting/blob/master/Build_a_Dashboard_with_Plotly_%20Dash_JupyterLab.ipynb

Predictive Analysis (Classification)

Create a logistic regression

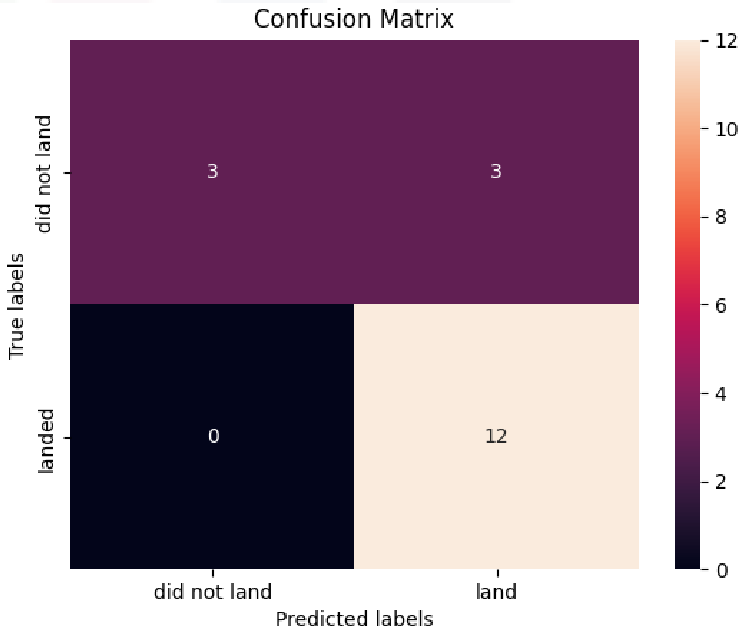
In [51]:
```
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
print("accuracy :",logreg_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8464285714285713
```

In [79]:
```
print('Accuracy on test data is: {:.10f}'.format(logreg_cv.score(X_test, Y_test
```

```
Accuracy on test data is: 0.8333333333
```

In [65]:
```
#Building confusion-matrix
plot_confusion_matrix(Y_test,yhat)
plt.show()
```



https://github.com/ENSS1971/Lab1_Collecting/blob/master/IBM-DS0321EN-SkillsNetwork_labs_module_4_SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite%20(2).ipynb

Predictive Analysis (Classification)

Create a support vector machine

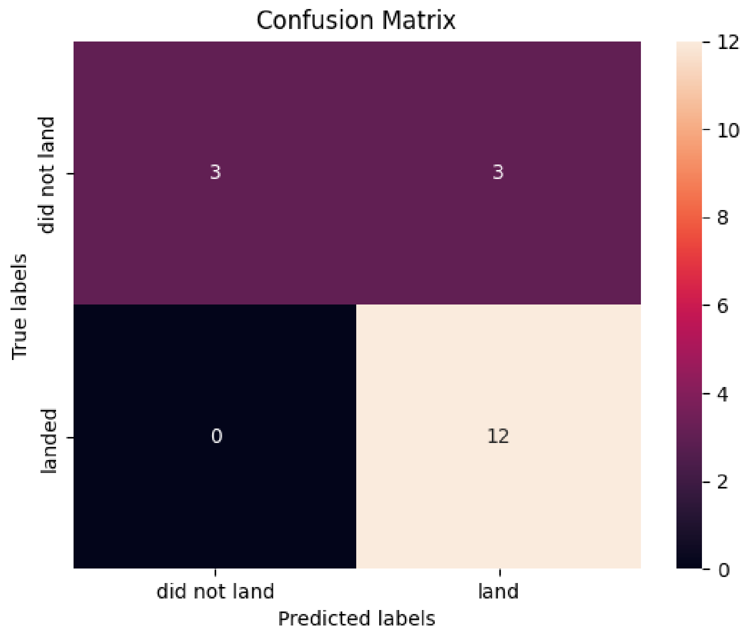
In [68]:
```
print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
print("accuracy :",svm_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
accuracy : 0.8482142857142856
```

In [78]:
```
print('Accuracy on test data is: {:.10f}'.format(svm_cv.score(X_test, Y_test)))
```

```
Accuracy on test data is: 0.8333333333
```

In [70]:
```
#Building confusion matrix
yhat=svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
plt.show()
```



https://github.com/ENSS1971/Lab1_Collecting/blob/master/IBM-DS0321EN-SkillsNetwork_labs_module_4_SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite%20(2).ipynb

Predictive Analysis (Classification)

Create a decision tree classifier

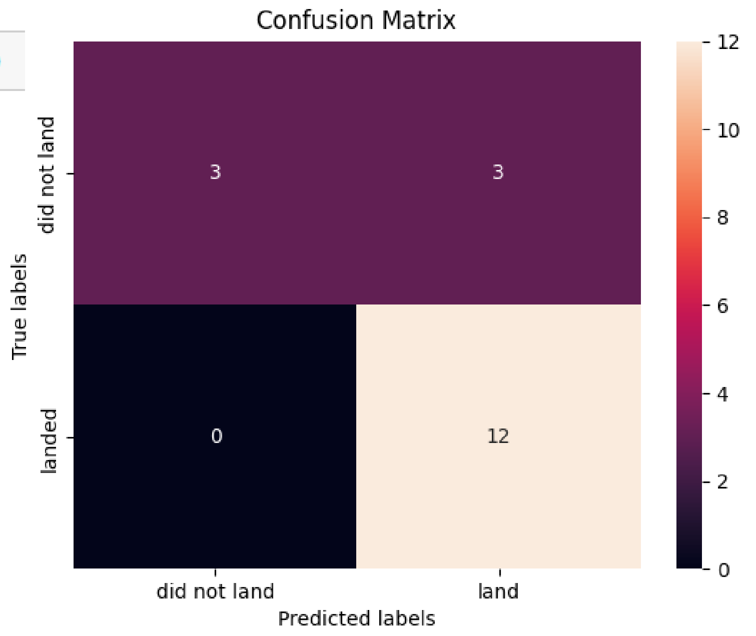
In [73]:
```
print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 10, 'splitter': 'random'}
accuracy : 0.8910714285714286
```

In [77]:
```
print('Accuracy on test data is: {:.10f}'.format(tree_cv.score(X_test, Y_test)))
```

```
Accuracy on test data is: 0.8333333333
```

In [ ]:
```
yhat = svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
plt.show()
```



https://github.com/ENSS1971/Lab1_Collecting/blob/master/IBM-DS0321EN-SkillsNetwork_labs_module_4_SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite%20(2).ipynb

Predictive Analysis (Classification)

Create a k nearest neighbors

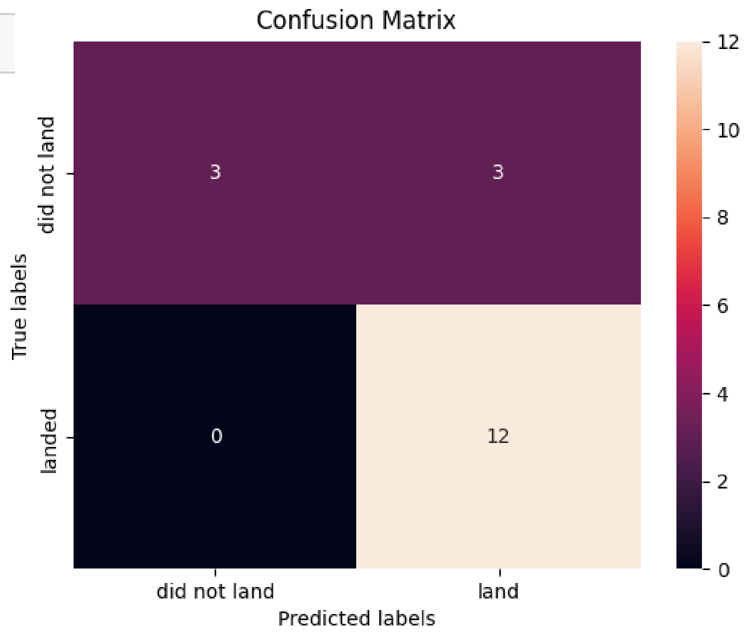
In [83]:
```
print("tuned hpyerparameters :(best parameters) ",KNN_cv.best_params_)
print("accuracy :",KNN_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
accuracy : 0.8482142857142858
```

In [84]:
```
print('Accuracy on test data is: {:.10f}'.format(tree_cv.score(X_test, Y_test)))
```

```
Accuracy on test data is: 0.8333333333
```

In [86]:
```
yhat = KNN_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
plt.show()
```



https://github.com/ENSS1971/Lab1_Collecting/blob/master/IBM-DS0321EN-SkillsNetwork_labs_module_4_SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite%20(2).ipynb

Accuracy Comparison

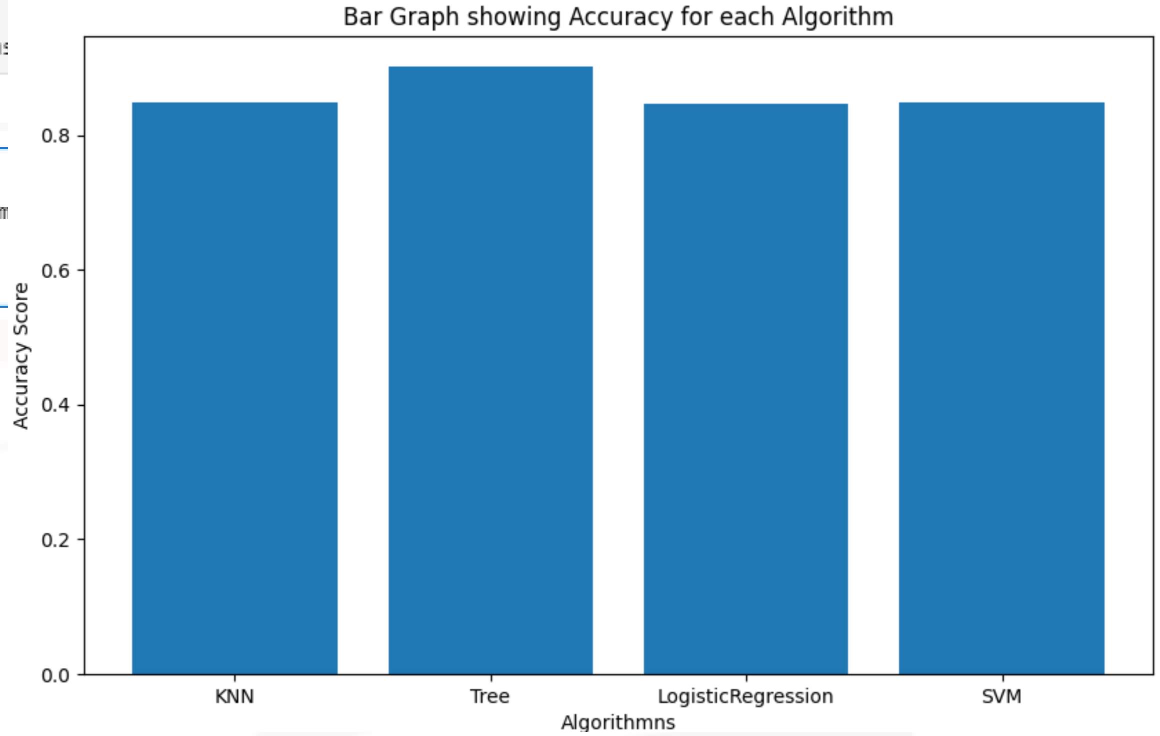
[38]:
```
#Accuracy Comparison of different algorithms on training data
algorithms = {'KNN':KNN_cv.best_score_,'Tree':tree_cv.best_score_,'L
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms
```

```
Best Algorithm is Tree with a score of 0.9017857142857142
```

[40]:
```
#Building algorithm dataframe
score_df = pd.DataFrame.from_dict(algorithms, orient='index', colum
score_df.sort_values(['Train Data Accuracy'], inplace=True)
score_df.head(6)
```

[40]:

	Train Data Accuracy
LogisticRegression	0.846429
SVM	0.848214
KNN	0.848214
Tree	0.901786



https://github.com/ENSS1971/Lab1_Collecting/blob/master/IBM-DS0321EN-SkillsNetwork_labs_module_4_SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite%20(2).ipynb

Conclusion

The four best methods SVM Hyperparameters, Classification Tree, Logistic Regression showed exactly the same confusion matrix. All models produced an Accuracy on test data equal to 0.8333333. This is because the dataset is small producing small values. But I can nominate the classification tree as the best model. Because we have an accuracy of 0.8910714285714286.

We can also highlight three important points in our analysis:

- Low weighted payloads perform better than the heavier payloads.
- The success rates for SpaceX launches is directly proportional time in years they will eventually perfect the launches.
- KSC LC 39A had the most successful launches from all the sites.
- Orbit GEO,HEO,SSO,ES L1 has the best Success Rate.

Acknowledgment

“I am very grateful to Coursera for having such competent and dedicated professionals. Thanks for the partnership, team!” “Motivation, partnership, dedication and teamwork is what led us to achieve our goal. Thank you for everyone's effort!”

IBM Developer

SKILLS NETWORK

My Presentation

Eduardo Nascimento de Souza Sepulveda

I have 10 years of experience in administrative/financial routines. Over the years I have achieved great results focusing on optimizing processes, contract management, cost reduction and contact with suppliers without compromising quality. I have a Bachelor's Degree in Business Administration with an emphasis on Management Information Systems and Marketing. I recently started studies in the area of Data Science with the aim of working on projects of this nature, developing predictive models especially for the financial area.

in linkedin.com/in/eduardo-sepulveda-6ba88135