IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Jahangir Khan
April 26, 2023

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection
    - SpaceX API (Application Programming Interface)
    - Web-Scraping
    - Data Wrangling
  - EDA (Exploratory Data Analysis)
    - Pandas
    - SQL (Sequential Query Language)
    - Interactive Visual Analytics (Maps)
      - Folium
      - Plotly Dash
  - ML-based Predictive Analysis for Classification models

- Summary of results
  - EDA-based Insights
  - Visualization and analysis
  - Interactive visualization-based Results
  - Choosing the best model for ML-based Predictive Analysis

Introduction

- Project background and context
  - **Goal:** To predict whether Falcon 9 first stage will land successfully
  - **Background:** SpaceX advertises Falcon 9 rocket launches with a cost of 62 million dollars;

    Other providers cost upward of 165 million dollars each

    SpaceX can reuse the first stage to cut down the cost for saving purpose
  - **Benefit:** To determine the cost of a launch if the first stage lands successfully

    This information can also be used by an alternate company to bid against SpaceX for a rocket launch
- Research Questions
  - What are the determining factors involving in rocket's successful launch?
  - What is the correlation amongst various features for determining the successful launches?
  - What are the operating conditions necessary for a successful launch?

Section 1

Methodology

Methodology

Data Collection	• Via SpaceX REST API • Web Scrapping via Wikipedia
Data Wrangling	• The values of the features, found discrete in nature, are converted into numerical values. • One hot encoding data fields for ML and dropping irrelevant columns
Exploratory Data Analysis	• Using Visualization and SQL • Via Scatter and Bar Graphs
Interactive Visual Analytics	• Using Folium and Plotly Dash
Predictive Analysis	• Build, tune, and evaluate classification models using: • Logistic Regression, SVM, Decision Tree, KNN Models • Grid Search Cross-Validation for tuning Hyperparameters • Score Method and Confusion Matrix for determining accuracy on Test Data

Data Collection via SpaceX REST API

Get Request to API → Data Decoding → DataFrame Creation → Data Cleaning → Data Filtering → Flat File Formatting

Step 1: Get Request to and Get Response from SpaceX API

Step 2: Decode the data

Step 3: Convert data into a DataFrame

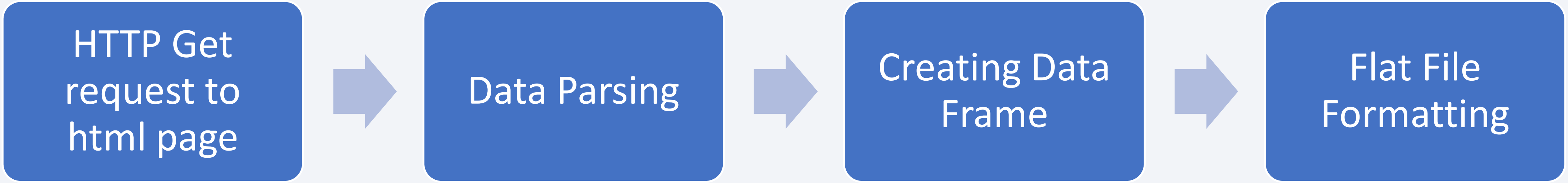
Step 4: Perform Data cleaning via custom functions

Step 5: Filter Data-Frame

Step 6: Export to flat file (.csv format)

- GitHub URL

Data Collection via Scraping



Step 1: Get request to and get response from relevant Wikipedia webpage

Step 2: Perform Data parsing of Launch records using Beautiful Soup

Step 3: Transform Data into DataFrame

Step 4: Export to Flat File (.csv format)

- GitHub URL

Data Wrangling

Data Loading → DataFrame Creation → Data Cleaning → Data Simplification with Boolean values

Data Wrangling Process

Step 1. Load the data

Step 2. Create DataFrame

Step 3. Perform Data Cleaning

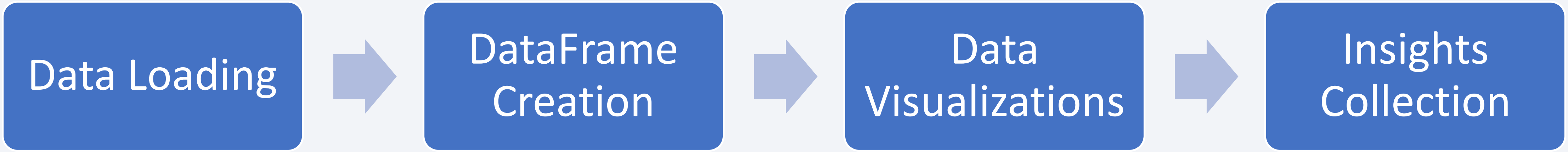
Step 4. Perform Data Simplifying by transforming values into Boolean Values

Step 5. Flat File Formatting

Calculate Launches per site → Calculate Occurrences per orbit type → Calculate Mission Outcomes per orbit type → Create Landing Outcome label from Outcome column

- GitHub URL

Exploratory Data Analysis via Data Visualization



EDA with Data Visualization

Using Pandas and Matplotlib, the following graphs were used for visualizing our data:

- **Scatter Plot:**

  Scatter plots are helpful in depicting dependencies among various attributes. In our case, these are helpful in identifying and predicting the success rate lying in rocket landing and its resulting outcome.

- **Bar Graph:**

  In our case, Bar graph is helpful in clearly visualizing the highest success rate among various orbit types.

- **Line Graph:**

  Line graphs are helpful in visualizing trends and getting insights into.

Selected Attributes for Data Visualization

- **Scatter plots** are drawn among the following attributes:

  1. Payload and Flight Number
  2. Flight Number and Launch Site
  3. Payload and Launch Site
  4. Flight Number and Orbit Type
  5. Payload and Orbit Type

- **Bar graph** is drawn among the following attributes:

  1. Success Rate and Orbit Type

- **Line graph** is drawn for showing:

  1. Launching Success Rates over years

- GitHub URL

EDA with SQL

List of SQL queries performed to retrieve information from the Dataset:

- All launching sites
- Launching sites beginning with 'CCA' (5 records)
- Total Payload Mass carried by Boosters, as launched by NASA (CRS)
- Average Payload Mass carried by Booster version F9v1.1
- Enlisting the dates for successful landing outcome on drone ship
- Enlisting the boosters with payload mass > 4000 and < 6000 and which were found successful on ground
- Total number of success and failure for mission Outcomes
- Names of the booster versions carrying to its maximum payload mass
- For the year 2015, enlist the failed landing outcomes on drone ship
- To rank the count of landing outcomes between the dates 2010-06-04 and 2017-03-20

- **GitHub URL**

Build an Interactive Map with Folium

1. **Map Marker:**
   To make a mark on map
2. **Icon Marker:**
   To create an icon on map
3. **Circle Marker:**
   To create a circle on map
4. **Poly Line:**
   To draw a line between points
5. **Marker Cluster:**
   To place bunch of markers on map having same coordinates
6. **Ant Path:**
   To draw an animated line between points

- GitHub URL

Build a Dashboard with Plotly Dash

The following chart/plot are used for visualization over the Dashboard:

1. **Pie Chart:**
    - Total success for all sites or a specific launch site
    - Success rate (in percentage) for each launching site
2. **Scatter Plot:**
    - Relationship of Payload to its Success Rate for all sites or a specific site
    - Relationship of Booster Version category to its Success Rate

- GitHub URL

Predictive Analysis (Classification)

Build Model → Evaluate Model → Choose the Classification Model having its best performance

Procedure for Predictive Analysis (Classification)

1. **Building Model:**

   The feature engineered data is loaded into DataFrame and is transformed into arrays. The data is standardized and transformed. It is then split into Training and Test Datasets. The parameters and algorithms to GridSearchCV. Finally, the Datasets were set as fit into GridSearchCV objects and the model been put to train.

2. **Model Evaluation:**

   Accuracy is confirmed for each model and got best hyperparameters for each type of algorithm. Finally, the confusion matrix is plotted.

3. **Choosing the best Classification Model:**

   The best model is selected based upon its best accuracy score.

- GitHub URL

Results

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

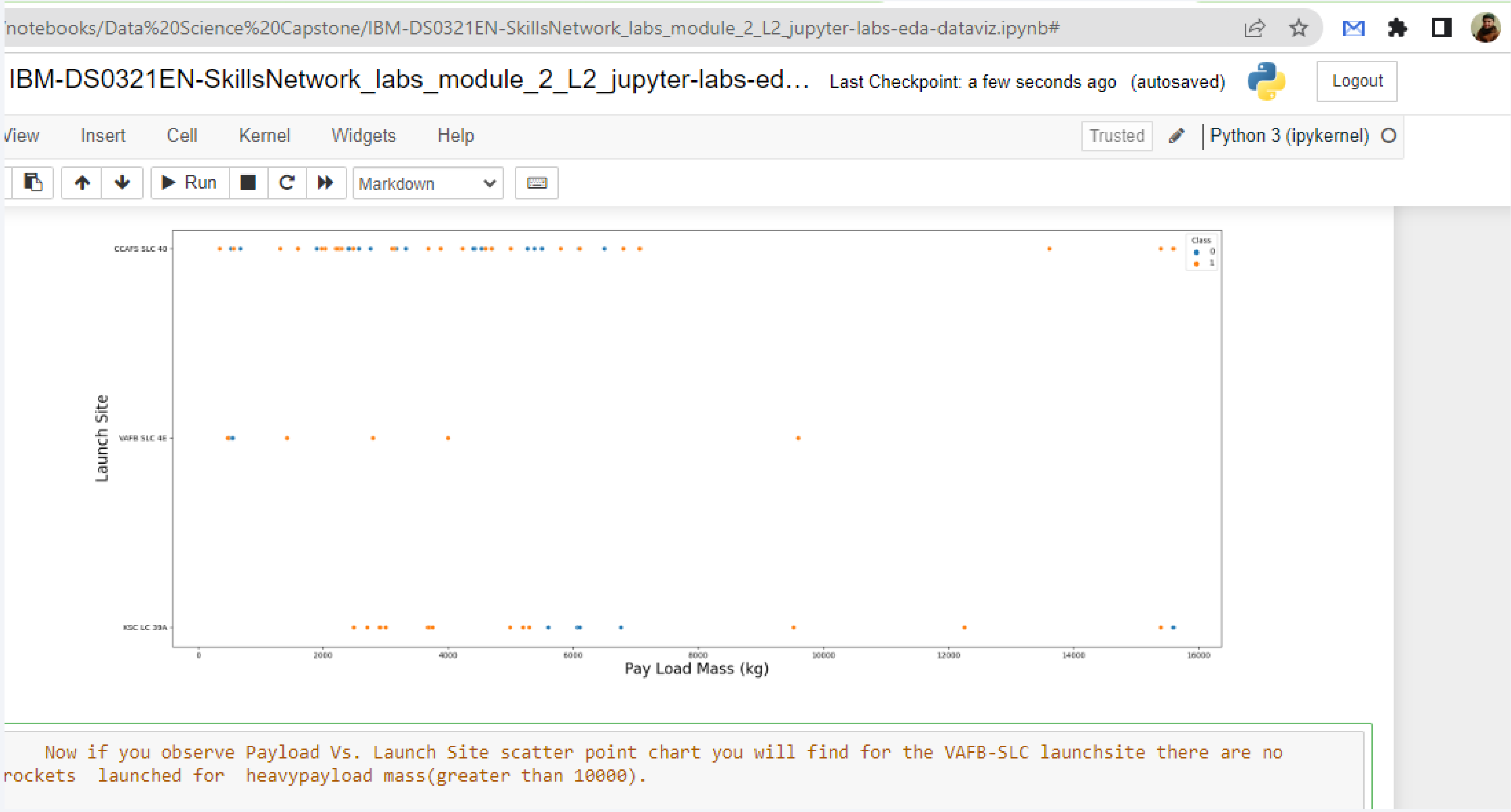
Section 2

Insights drawn from EDA

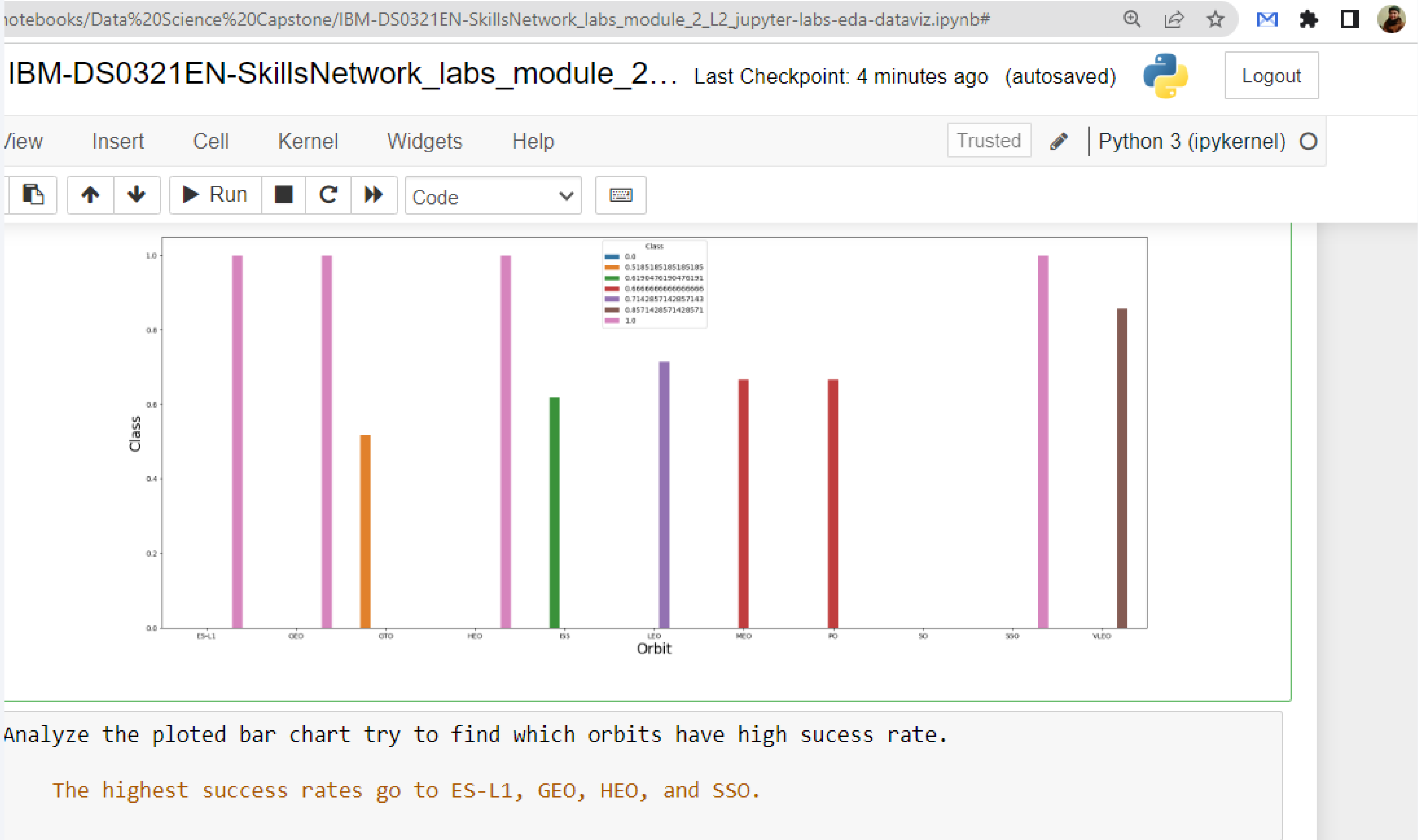
Flight Number vs. Launch Site

- GitHub URL

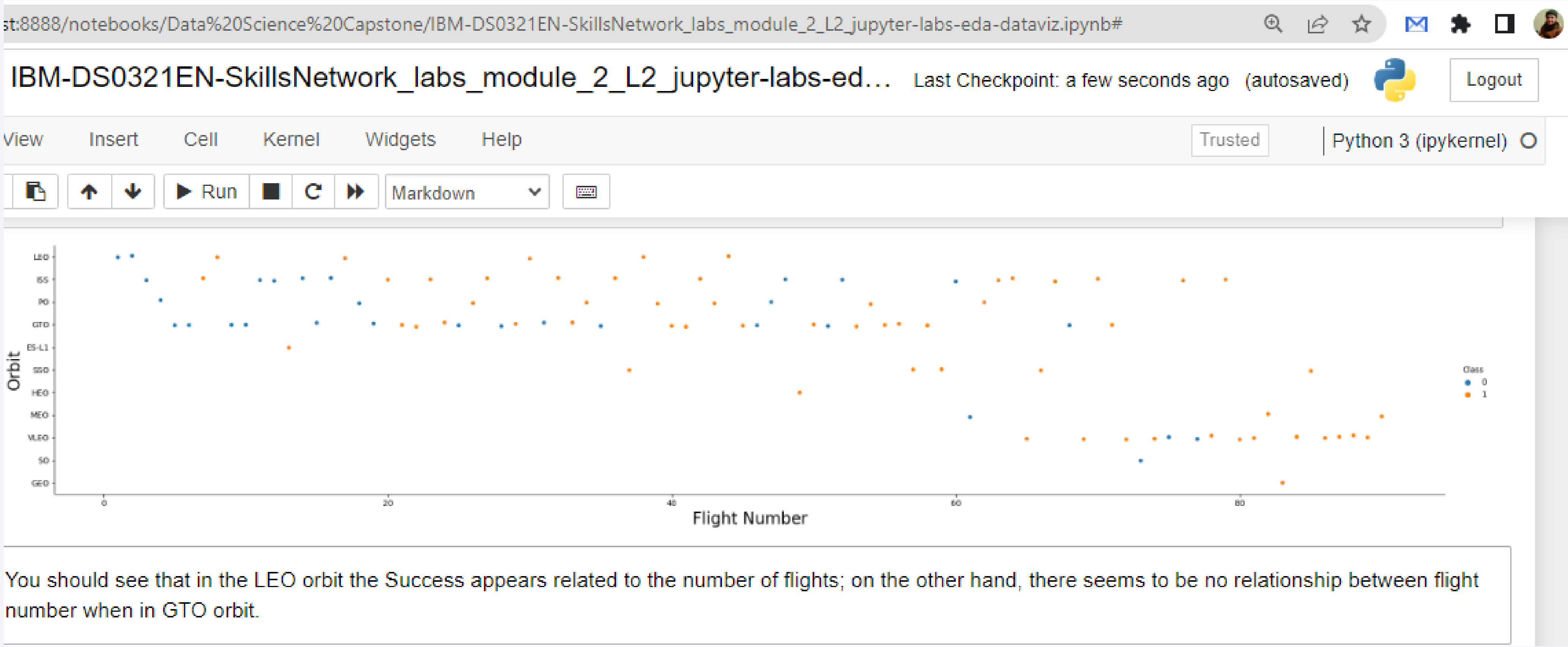
Payload vs. Launch Site



Success Rate vs. Orbit Type



Flight Number vs. Orbit Type



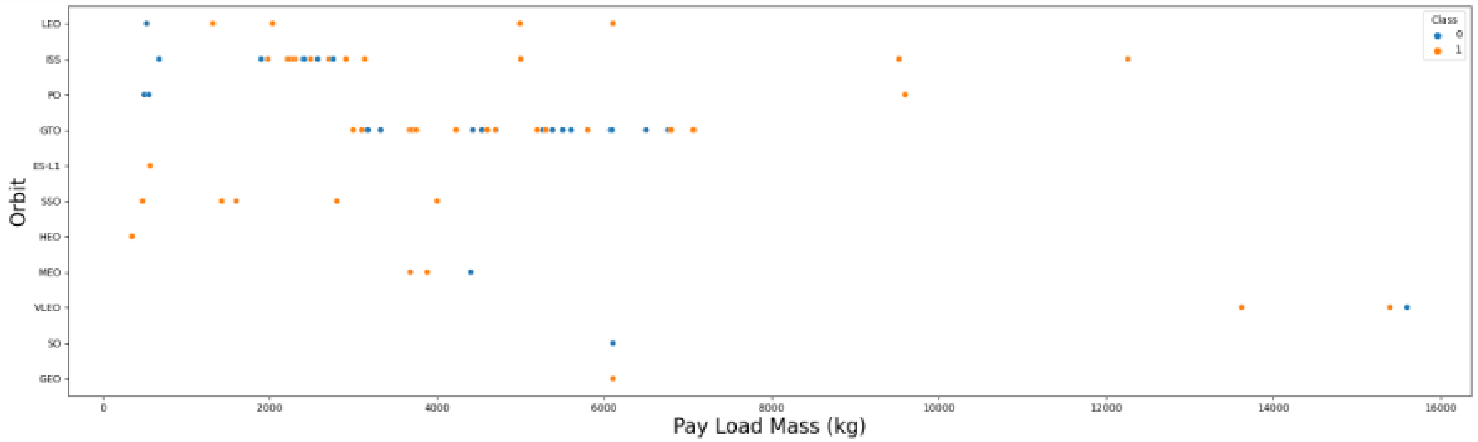
Payload vs. Orbit Type

ost:8888/notebooks/Data%20Science%20Capstone/IBM-DS0321EN-SkillsNetwork_labs_module_2_L2_jupyter-labs-eda-dataviz.ipynb#

IBM-DS0321EN-SkillsNetwork_labs_module_2_L2_jupyter-labs-eda-d… Last Checkpoint: 2 minutes ago (autosaved) Logout

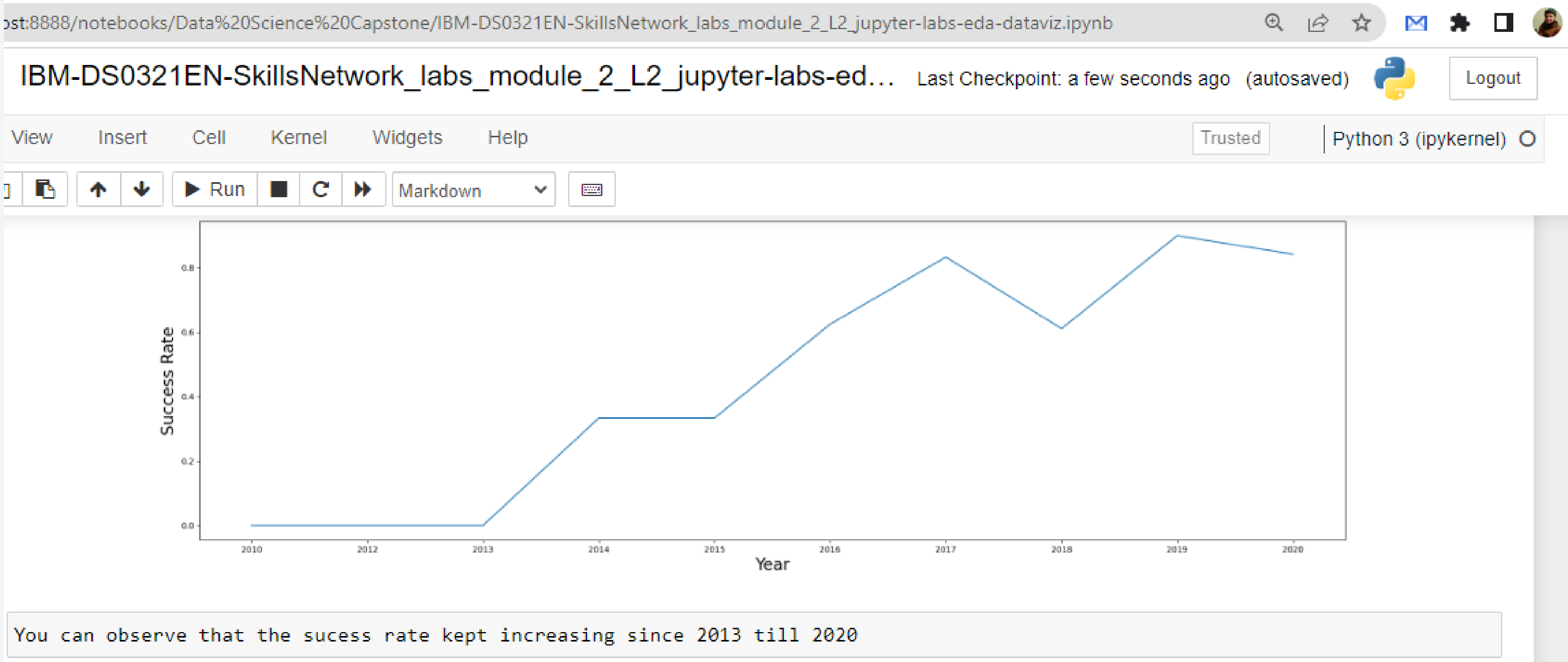
View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Markdown



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) both exist.

Launch Success Yearly Trend



EDA with SQL

All Launch Site Names

localhost:8888/notebooks/Data%20Science%20

jupyter IBM-DS0321EN-SkillsNetw

File Edit View Insert Cell Kernel

Run

In [228]:

```
%%sql
SELECT DISTINCT(Launch_Site)
FROM spacextbl;
```

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

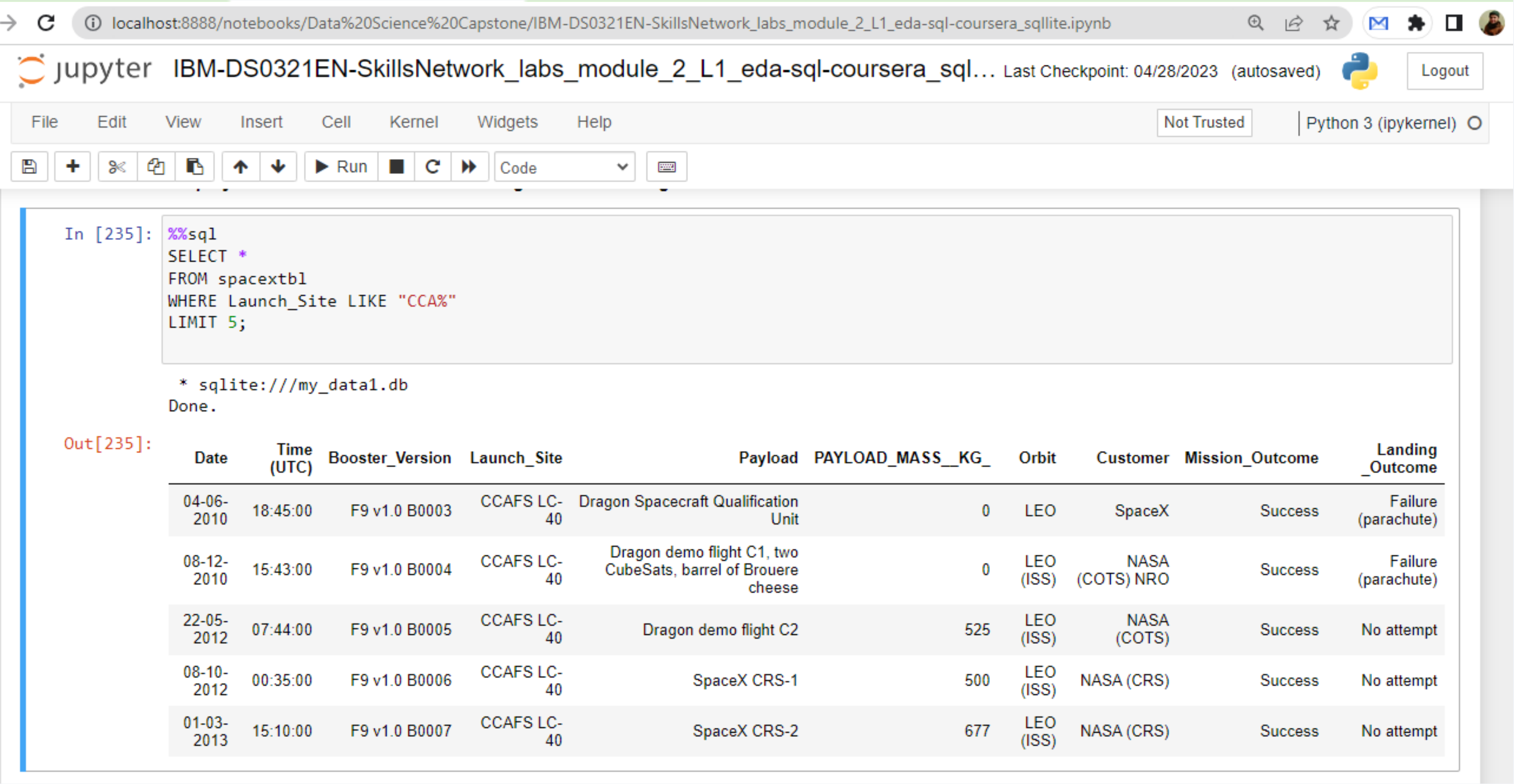
Out[228]:

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

The query is to retrieve all unique Site Names from the particular column of the table using DISTINCT keyword.

- GitHub URL

Launch Site Names Beginning with 'CCA'




```
%%sql
SELECT *
FROM spacextbl
WHERE Launch_Site LIKE "CCA%"
LIMIT 5;
```

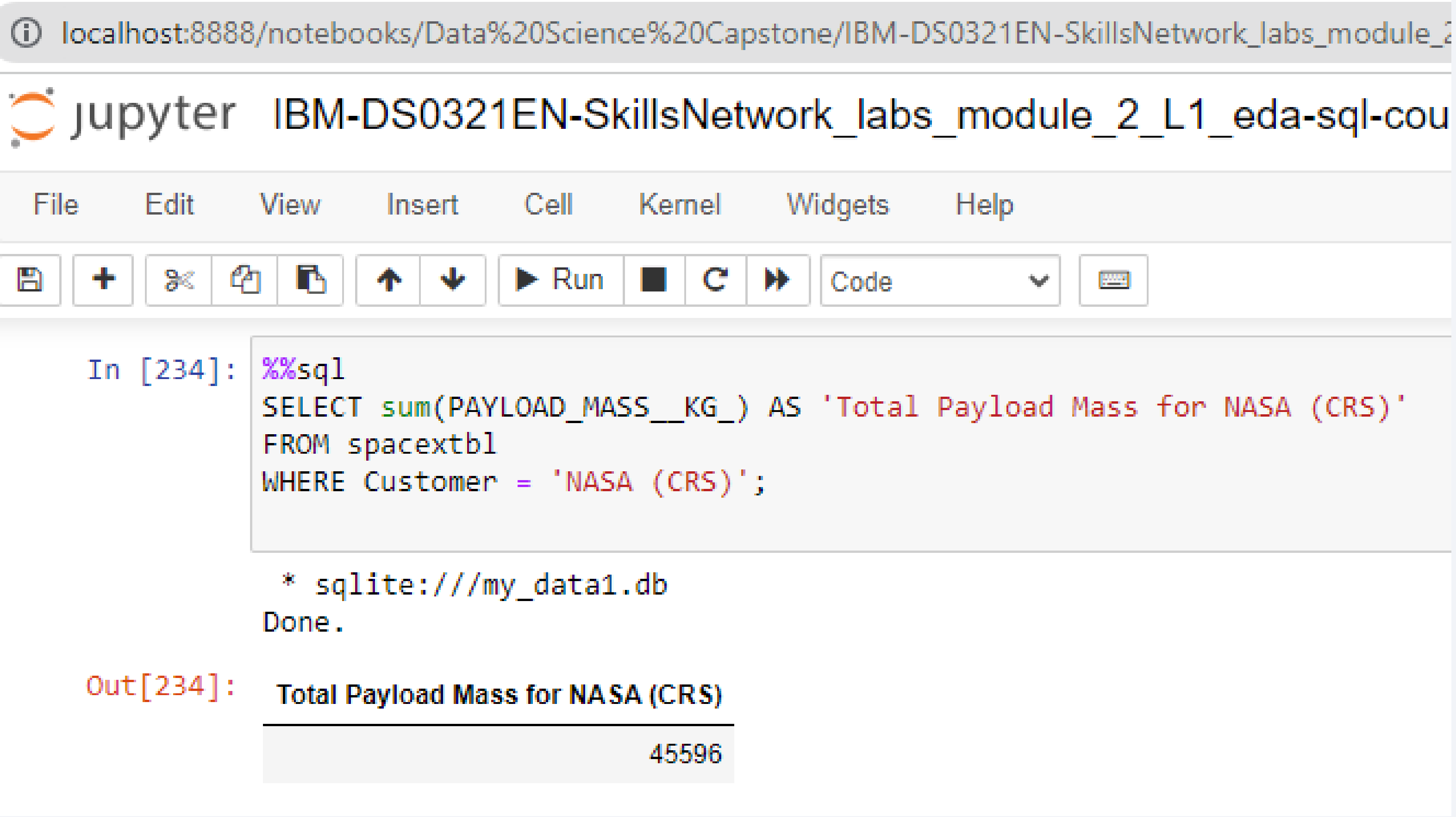
```
 * sqlite:///my_data1.db
Done.
```

Out[235]:

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

The query is to retrieve all the names of the Launch sites which start with ‘CCA’ using LIKE keyword to resemble and wild card ‘%’ is used to ignore what come after ‘CCA’ in a name.

Total Payload Mass



The query is to calculate and display the sum of Payload Mass for NASA, the values of which are retrieved from the specific column of the table based upon the value in the 'Customer' column indicating 'NASA (CRS)' only, using 'WHERE' clause. Moreover for clarity purpose an alias upon column name is used for clarity using 'AS' keyword.

Average Payload Mass by F9 v1.1

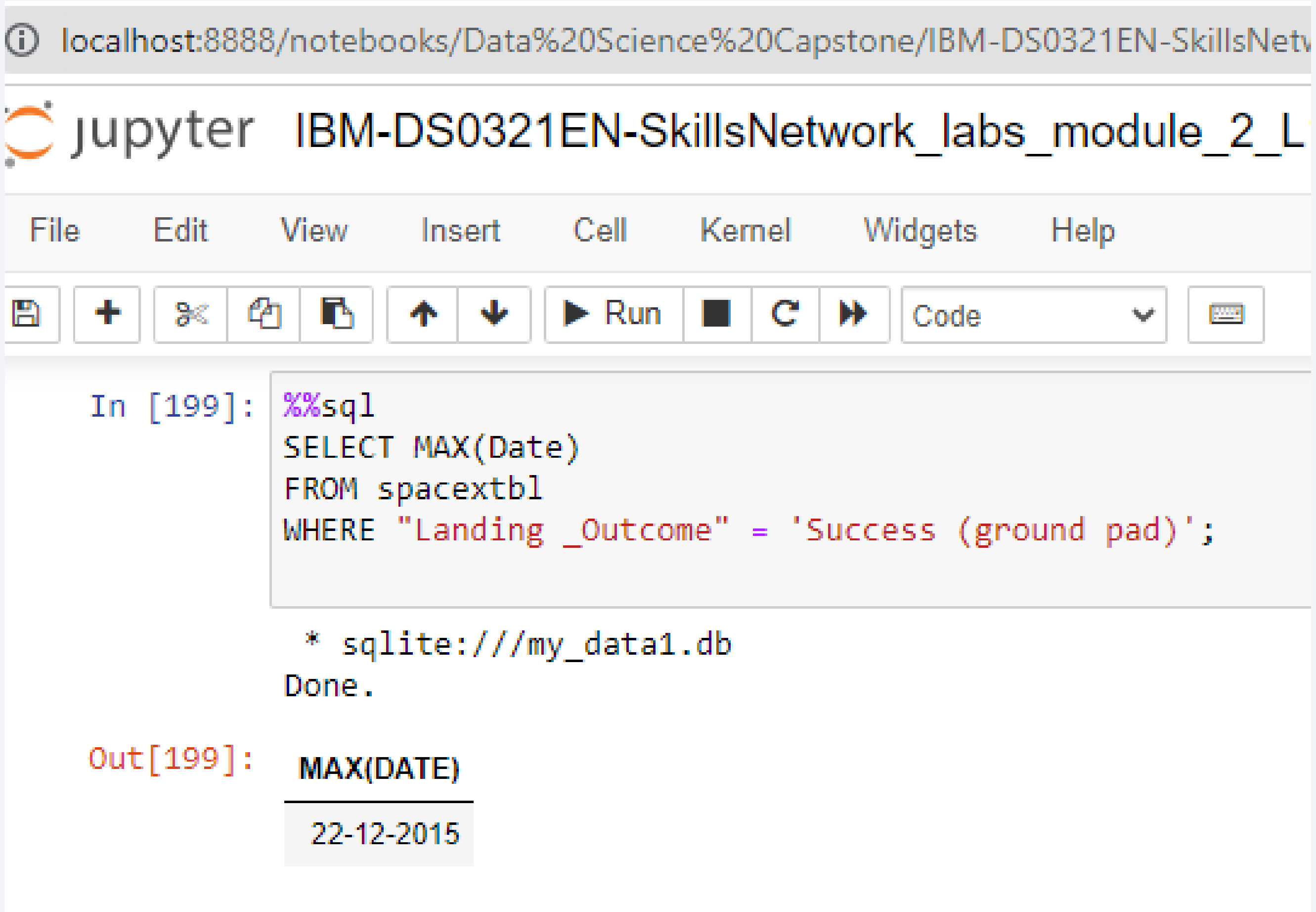
```
In [233]: %%sql
SELECT AVG(PAYLOAD_MASS__KG_) as "AVG Payload (F9 v1.1)"
FROM spacextbl
WHERE Booster_Version LIKE 'F9 v1.1%';

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

Out[233]:

AVG Payload (F9 v1.1)
2534.6666666666665

The query instructs to calculate and display the average of Payload Mass by limiting the selection of those records for which Booster Version starts with the name 'F9 v1.1' and wild card '%' sign indicates ignoring whatever comes after the specified starting characters

First Successful Ground Landing Date



The query retrieves the value by fulfilling the condition upon the Landing outcome column indicating 'Success (ground pad)' using WHERE clause and by using 'MAX' keyword for choosing and displaying the first successful ground landing date only.

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%%sql
SELECT Booster_Version
FROM spacextbl
WHERE (PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000) AND "Landing _Outcome" = "Success (drone ship)";
```

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

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

The query enforces the condition of mentioning the specified range of Payload mass using BETWEEN keyword in the WHERE clause and enforces selecting only those values from Landing outcome which contains ‘Success (drone ship)’. It displays the list of Booster versions upon fulfilling the above conditions.

Total Number of Successful and Failure Mission Outcomes

In [236]:
```
%%sql
SELECT COUNT(Mission_Outcome) AS "Successful Mission Outcomes"
FROM spacextbl
WHERE Mission_Outcome LIKE "Success%";
```

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

Out[236]:

Successful Mission Outcomes
100

In [237]:
```
%%sql
SELECT COUNT(Mission_Outcome) as "Failure Mission Outcomes"
FROM spacextbl
WHERE Mission_Outcome LIKE "Failure%";
```

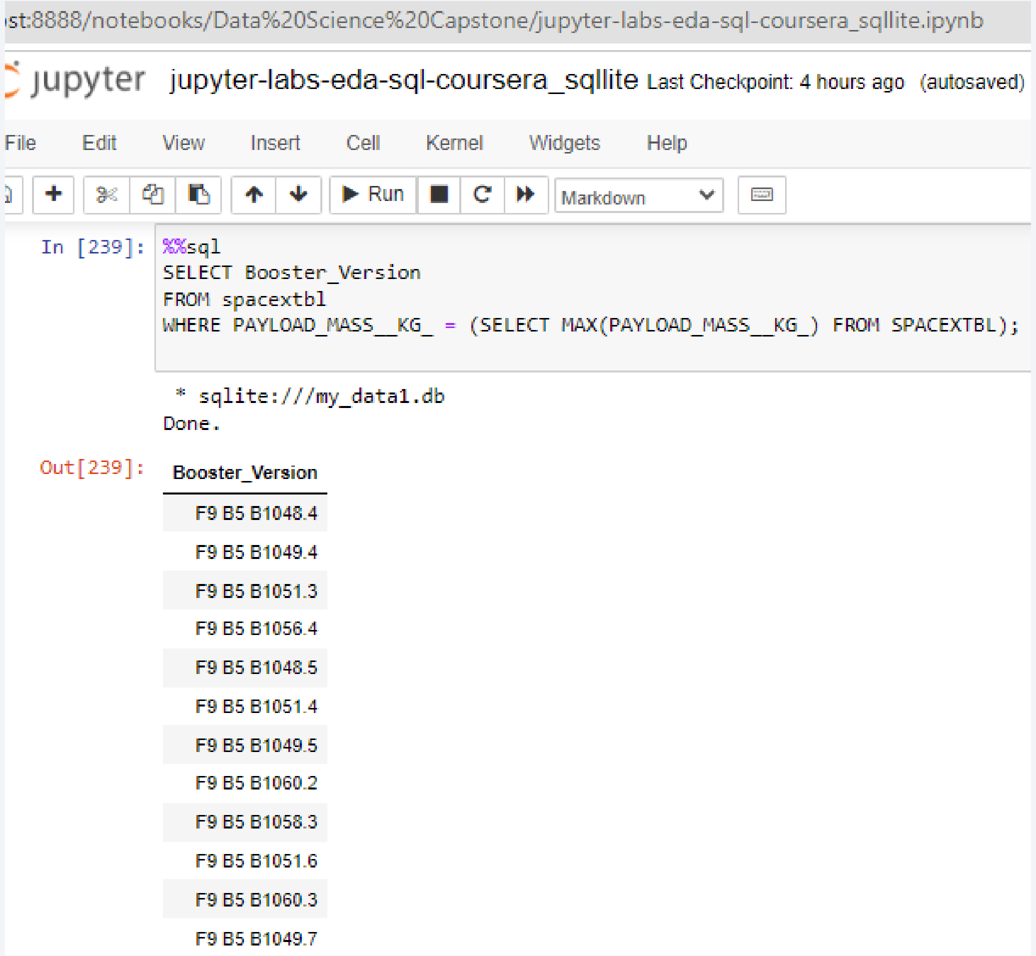
```
 * sqlite:///my_data1.db
Done.
```

Out[237]:

Successful Mission Outcomes
1

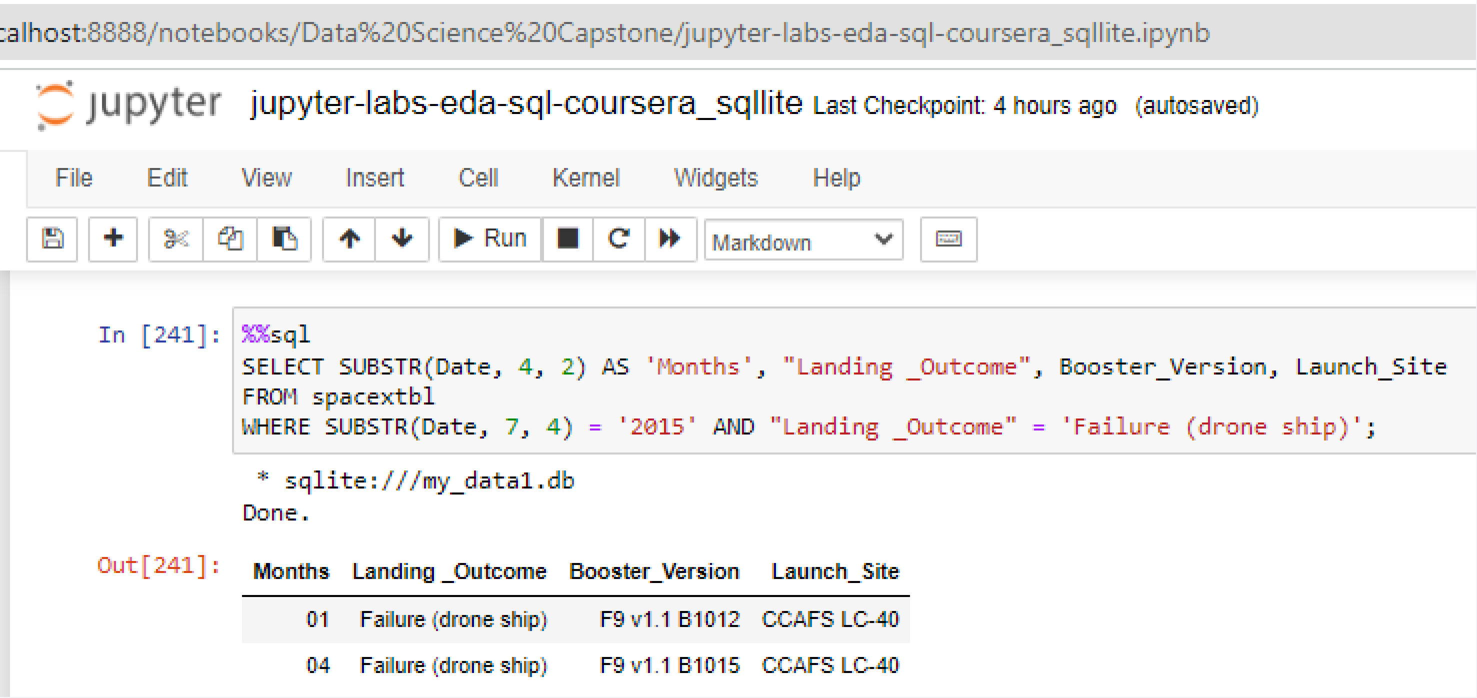
Queries 1 & 2 are applying conditions upon Mission Outcome to get value resembling to 'Success%' and 'Failure%' using LIKE keyword and using wild card '%' for ignoring the characters afterwards. Lastly, applying COUNT upon the retrieved records display summation of the Successful and Unsuccessful Mission Outcomes.

Boosters Carrying Maximum Payload



This query is having a subquery within which selects maximum value of Payload mass using MAX function and compares it with the Payload mass in a condition using WHERE clause which displays each booster version which carried to its maximum payload mass.

Launch Records 2015




```
%%sql
SELECT SUBSTR(Date, 4, 2) AS 'Months', "Landing _Outcome", Booster_Version, Launch_Site
FROM spacextbl
WHERE SUBSTR(Date, 7, 4) = '2015' AND "Landing _Outcome" = 'Failure (drone ship)';
```

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

Months	Landing _Outcome	Booster_Version	Launch_Site
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

This query makes use of SUBSTR function to locate month and year within date values. Firstly, imposing the condition of the year 2015 upon the date column along with searching for ‘Failure (drone ship)’ value in the Landing Outcome column. Secondly, SUBSTR is again applied to Date column for locating and displaying all months for the year 2015 along with the other required column values of the selected records.

Landing Outcomes Ranking for the Period 2010-06-04 and 2017-03-20

```
In [242]: %%sql
SELECT "Landing _Outcome", COUNT("Landing _Outcome") AS 'Successful Landings (04-06-2010 between 20-03-2017)'
FROM spacextbl
WHERE ("Landing _Outcome" LIKE 'Success%') AND Date BETWEEN '04-06-2010' AND '20-03-2017'
GROUP BY "Landing _Outcome"
ORDER BY 'Successful Landings (04-06-2010 and 20-03-2017)' DESC;

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

Out[242]:

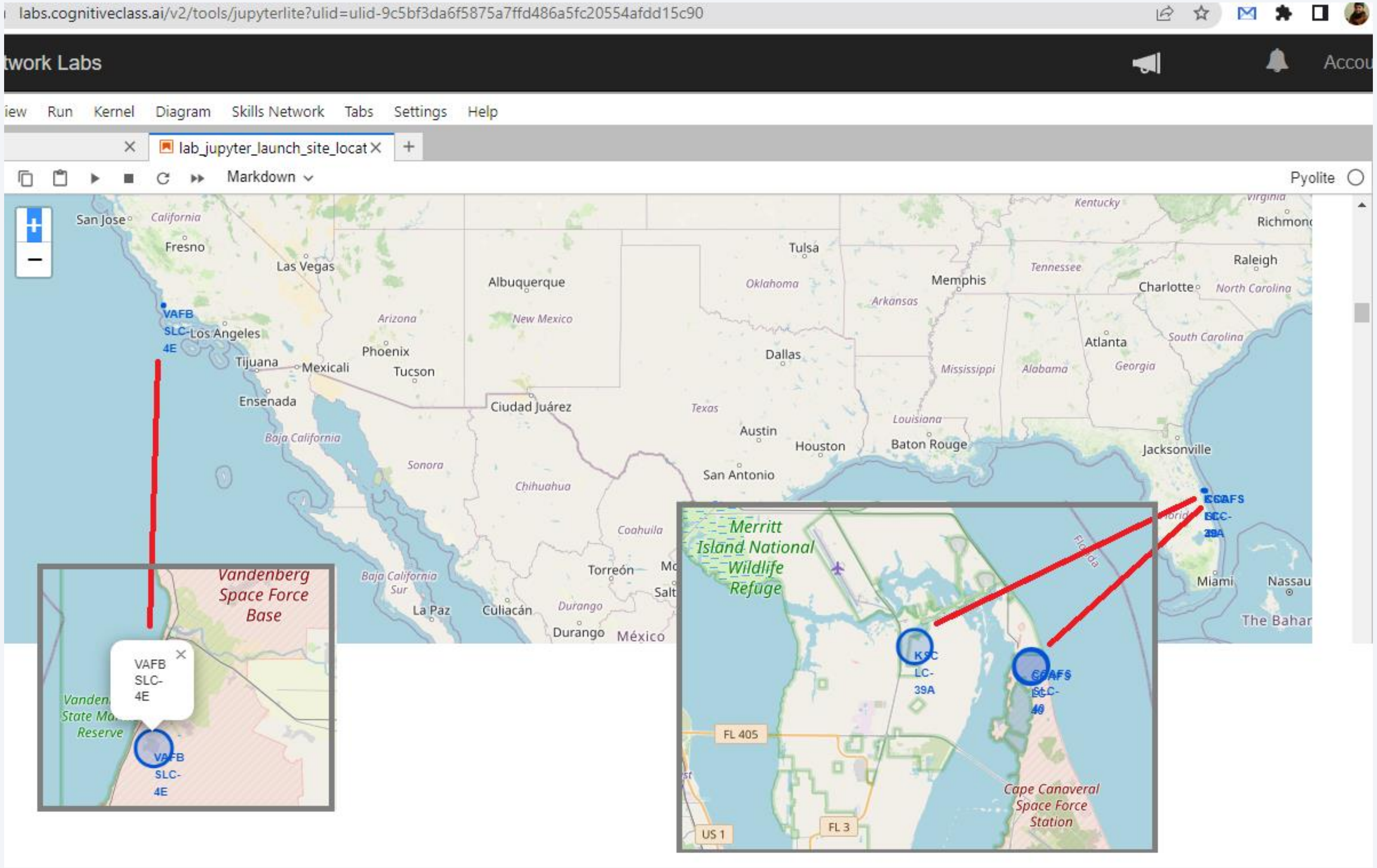
Landing _Outcome	Successful Landings (04-06-2010 between 20-03-2017)
Success (ground pad)	6
Success (drone ship)	8
Success	20

This query displays unique values of the column ‘Landing Outcome‘ using GROUP BY clause. The condition is applied with WHERE clause which resembles values of column ‘Landing Outcome’ as ‘Success’ using LIKE keyword and wildcard ‘%’ to ignore the characters after Success, if any. Also the WHERE clause puts up the condition stating range of dates using BETWEEN keyword. The query then displays output in groups while counting up the number of records found under each group using COUNT function upon ‘Landing Outcome’.

Section 3

Launch Sites Proximities Analysis

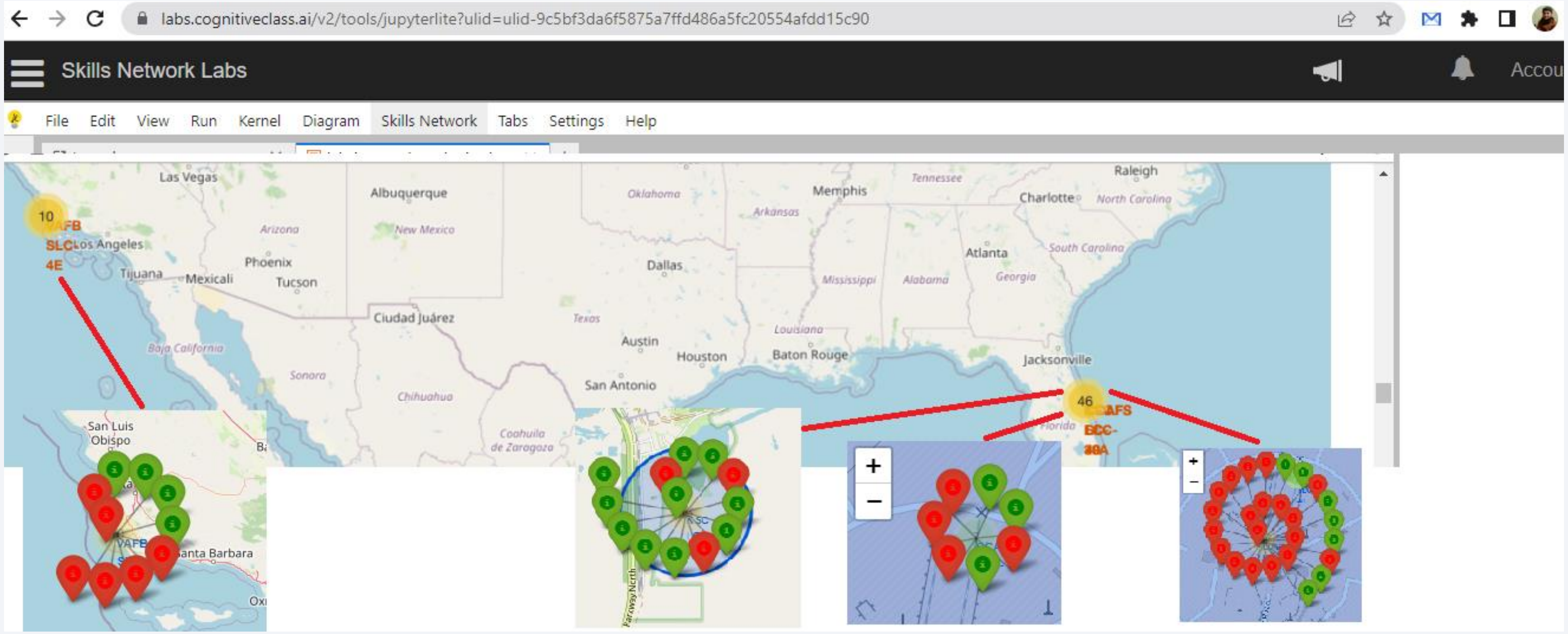
Folium Map – All Launch Sites



- It has been observed that all Launch Sites are close to the coastlines of California and Florida in United States.
- The reason is to manage risk factors involved during launches.

- [GitHub URL]

Launch Outcomes for each Site

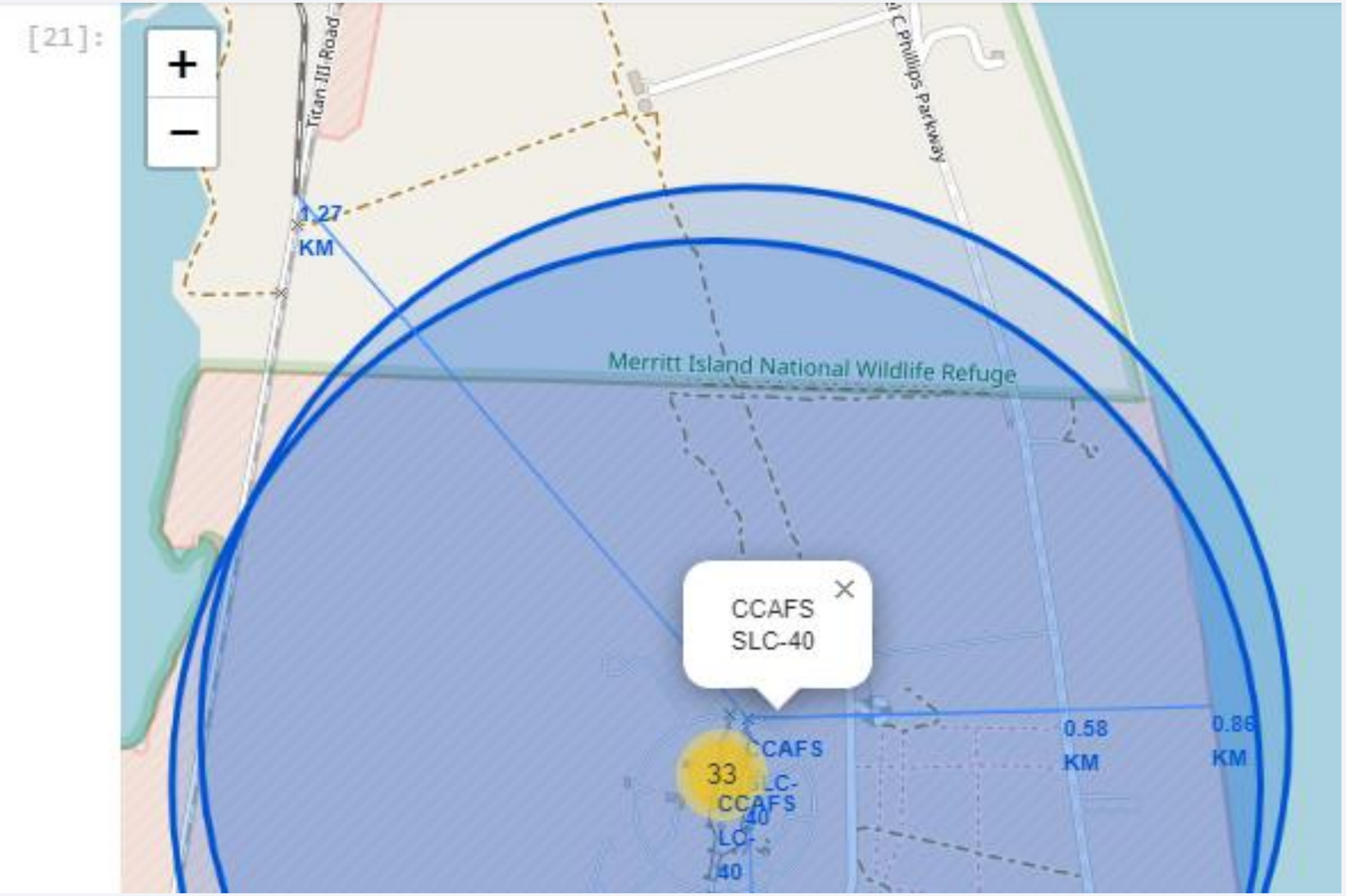


VAFB SLC-4E KSC LC-39A CCAFS SLC-40 CCAFS LC-40

- Launch Outcomes
- **Green Marker** showing successful launches
- **Red Marker** showing failed launches

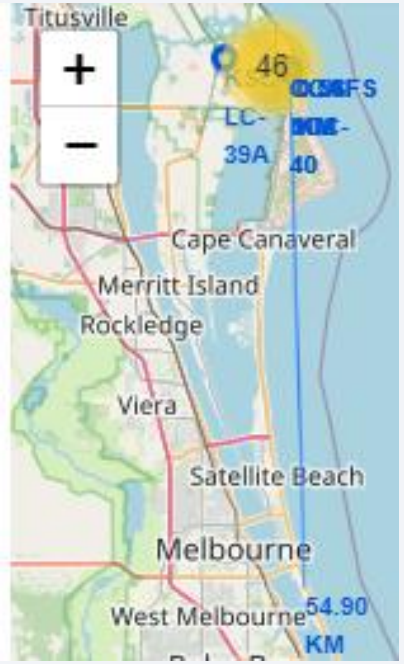
After exploring each site on the map and at our first glance shows that KSC LC-39A has the most success rate comparatively.

On-Map Distances Calculation – Launch Site to its Proximities



Conclusively, on the map, we can see that

- The Launch site CCAFS SLC-40 is in its closest proximity to Railways, i.e., 1.27 km.
- The same site is in its closest proximity to highway, i.e., 0.58 km.
- The same site is in its closest proximity to coastline, i.e., 0.86 km.

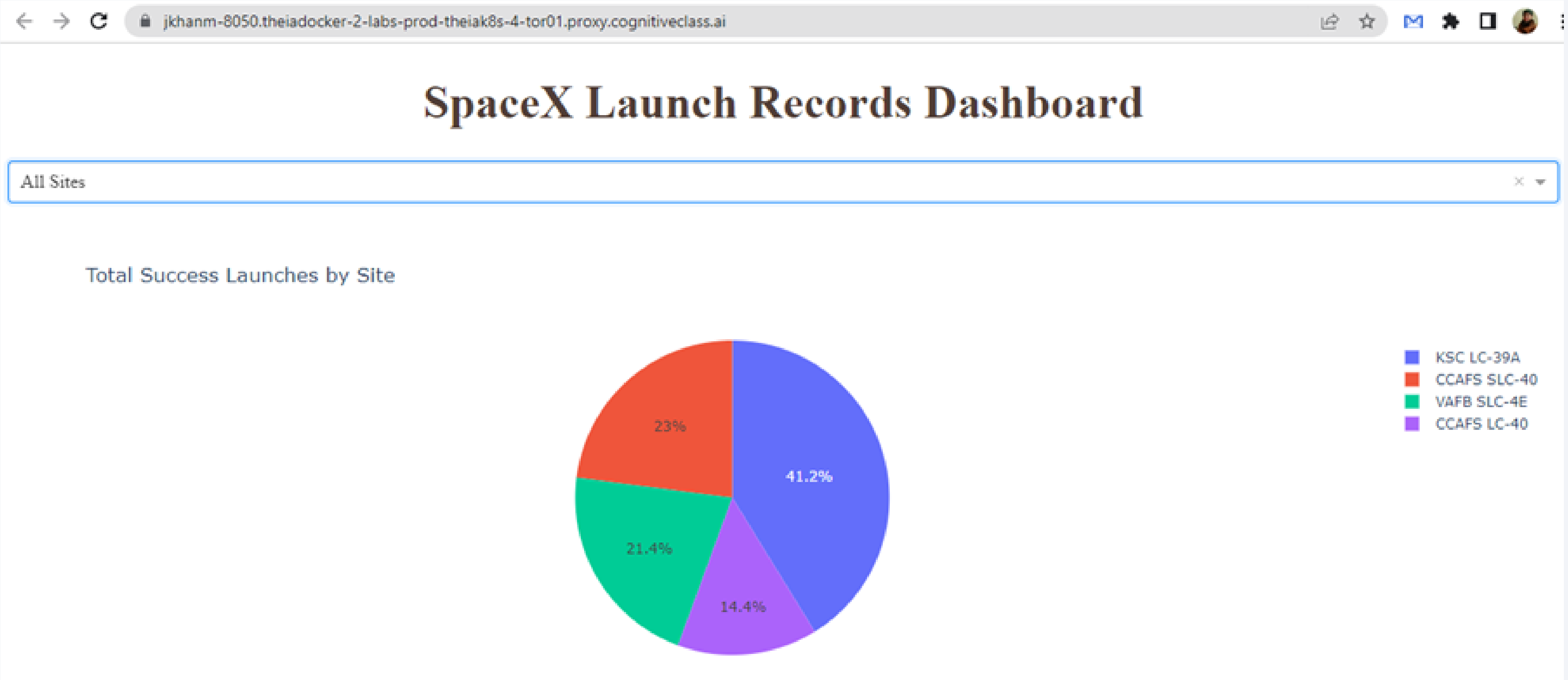


- The Launch site CCAFS SLC-40 is at a distance quite away from the Melbourne city, i.e., 54.90 km.

Section 4

Build a Dashboard with Plotly Dash

Success Rates for all Launch Sites

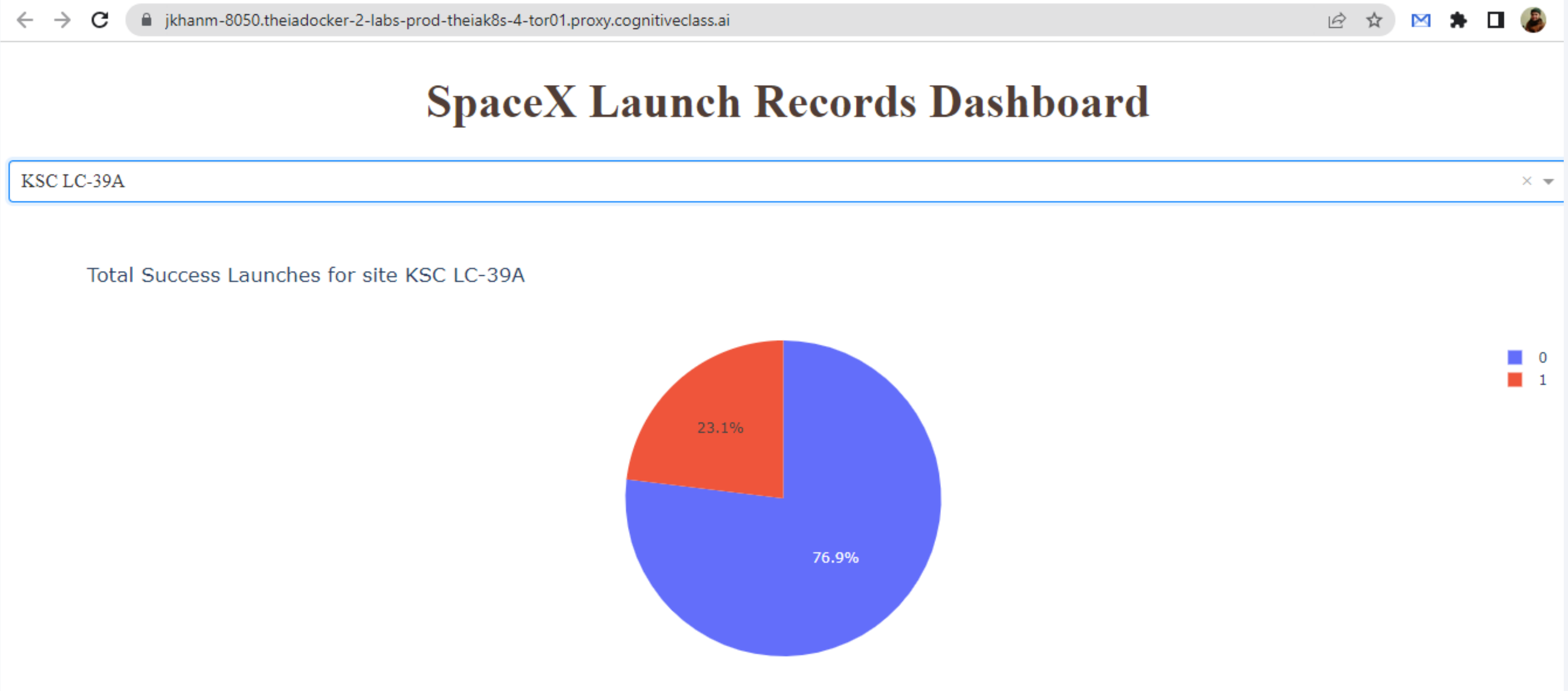


It can be easily visualized from the Pie chart that KSC LC 39-A is having the winning rate in its successful launches as compared to other launch sites.

- [GitHub URL]

See Appendix-A for complete programming code of this Dashboard Application

Success Rates for the Launch Site with Highest Success Rate



- The Pie Chart shows its Success to Failure ratio for KSC LC-39A, which is 76.9% success to 23.1% failure rate.

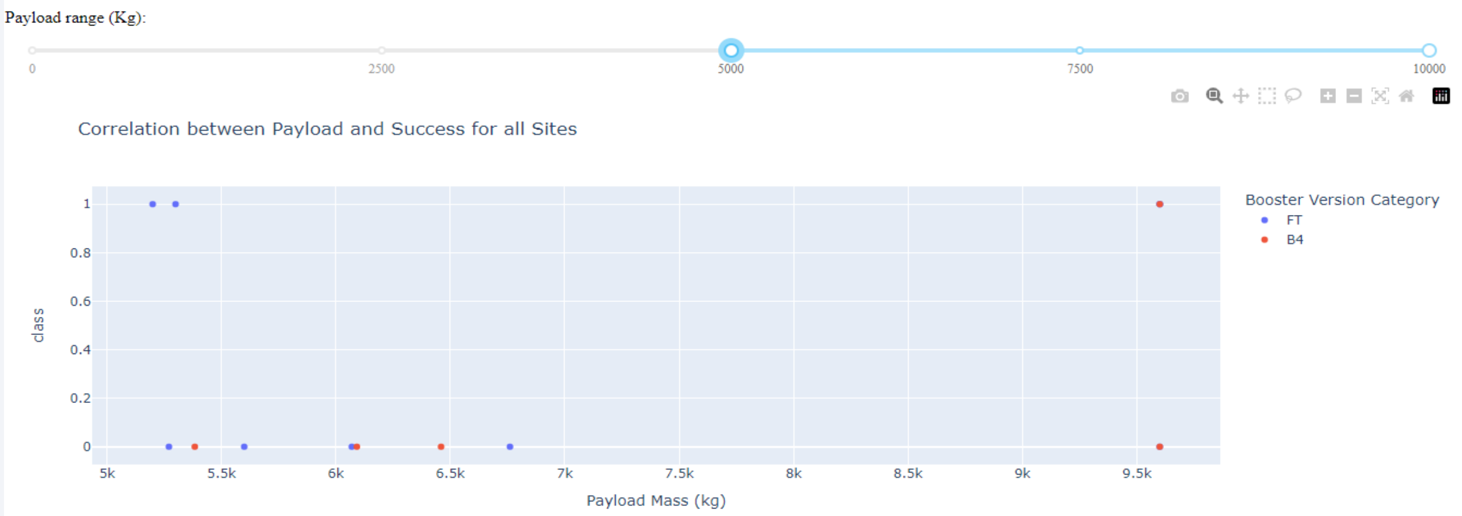
Insights Obtained:

1. The Launch Site KSC LC-39A is having the highest rate in its successful launches.
2. It has also been observed while comparing Payload Mass to Outcomes that Range of Payloads 2000 kg to 10000 kg is having the highest launch success rate while,
3. Range of Payloads 1000 kg to 1000 kg has the lowest success rate.
4. Booster Version FT has the highest launch success rate.

Payload Mass to Outcome Correlation for Booster Versions

Lower Payload Range (0 – 5000 kg)

Higher Payload Range (5000 – 10000 kg)



Here, it is observed that lower the Payload Mass, higher the rate of successful launches which makes it inversely proportional to each other.

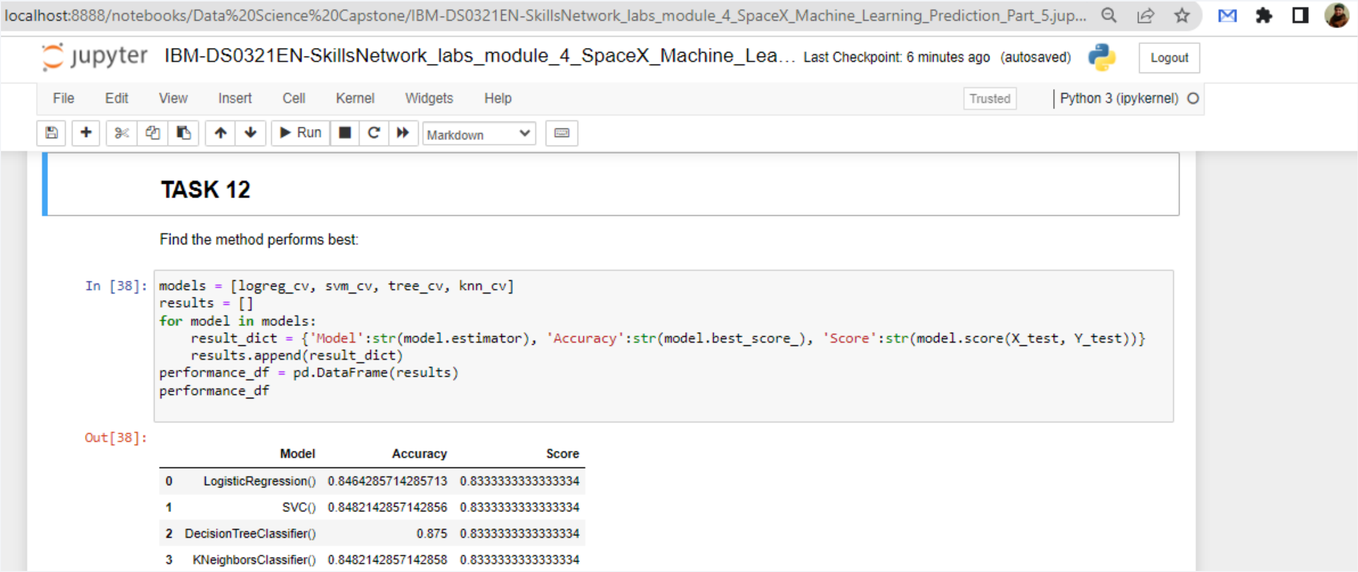
Insights Obtained:

1. The Launch Site KSC LC-39A is having the highest rate in its successful launches.
2. It has also been observed while comparing Payload Mass to Outcomes that Range of Payloads 2000 kg to 10000 kg is having the highest launch success rate while,
3. Range of Payloads 1000 kg to 1000 kg has the lowest success rate.
4. Booster Version FT has the highest launch success rate.

Section 5

Predictive Analysis (Classification)

Classification Accuracy



TASK 12

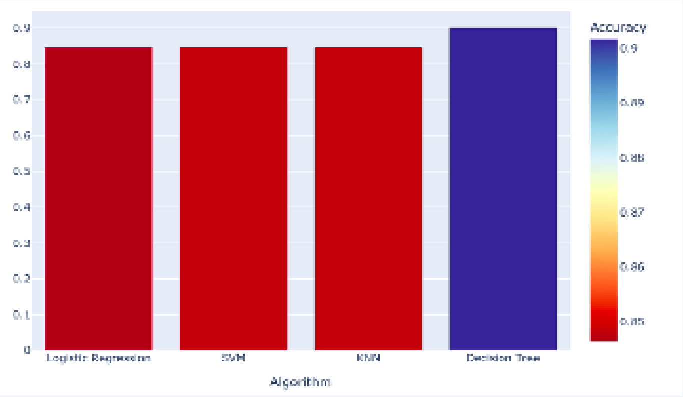
Find the method performs best:

In [38]:

```
models = [logreg_cv, svm_cv, tree_cv, knn_cv]
results = []
for model in models:
    result_dict = {'Model':str(model.estimator), 'Accuracy':str(model.best_score_), 'Score':str(model.score(X_test, Y_test))}
    results.append(result_dict)
performance_df = pd.DataFrame(results)
performance_df
```

Out[38]:

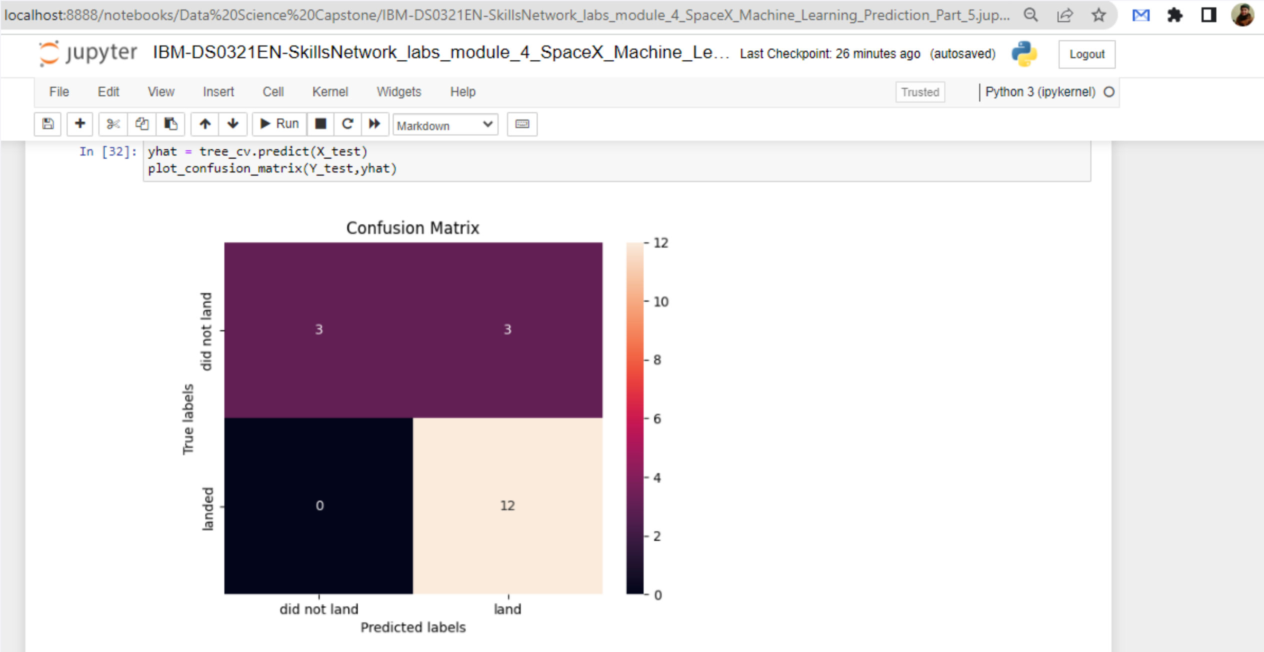
	Model	Accuracy	Score
0	LogisticRegression()	0.8464285714285713	0.8333333333333334
1	SVC()	0.8482142857142856	0.8333333333333334
2	DecisionTreeClassifier()	0.875	0.8333333333333334
3	KNeighborsClassifier()	0.8482142857142858	0.8333333333333334



- Four models have been trained and compared by retrieving its accuracy values.
- The output shows that Decision Tree with its applied algorithm is having the highest Accuracy of 0.875.

- GitHub URL

Confusion Matrix



- Confusion Matrix is used for Performance Measurement for Machine Learning Classification.
- It has been found that there is no dissimilarity among the four algorithmic models applied for performance measurement.
- But in terms of accuracy, the winning score goes to Decision Tree Model which is 0.875 as highest.

Conclusions

1. The Orbits with highest success rates are ES-L1, GEO, HEO, SSO.
2. The Launch success rate is found increased over time which tends to meet the required success goal.
3. KSC LC-39A launch site got most successful launches in past which could be a better choice for future launch. The factor of Payload Mass is to be considered as its decreasing leads to success.
4. For the given Dataset, the Algorithm “Decision Tree Classifier” is found to be the best ML Model.

Appendix A – Code Snippet of SpaceX Dash Application

```
# Import required libraries
import pandas as pd
import dash
import dash_html_components as html
import dash_core_components as 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_launch_dash.csv")
max_payload = spacex_df['Payload Mass (kg)'].max()
min_payload = spacex_df['Payload Mass (kg)'].min()
# 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',...)
                                dcc.Dropdown(id='site-dropdown',
                                options=[
                                    {'label': 'All Sites', 'value': 'All Sites'},
                                    {'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'}
                                ],
                                placeholder='Select a Launch Site Here',
                                value='All Sites',
                                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=25000,
                                marks={i: '{}'.format(i) for i in range(0, 10001, 2500)},
                                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
@app.callback( Output(component_id='success-pie-chart', component_property='figure'),
              Input(component_id='site-dropdown', component_property='value'))
def get_pie_chart(launch_site):
    if launch_site == 'All Sites':
        fig = px.pie(values=spacex_df.groupby('Launch Site')['class'].mean(),
                     names=spacex_df.groupby('Launch Site')['Launch Site'].first(),
                     title='Total Success Launches by Site')
    else:
        fig = px.pie(values=spacex_df[spacex_df['Launch Site']==str(launch_site)]['class'].value_counts(normalize=True),
                     names=spacex_df['class'].unique(),
                     title='Total Success Launches for site {}'.format(launch_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_payload_chart(launch_site, payload_mass):
    if launch_site == 'All Sites':
        fig = px.scatter(spacex_df[spacex_df['Payload Mass (kg)'].between(payload_mass[0], payload_mass[1])],
                x="Payload Mass (kg)",
                y="class",
                color="Booster Version Category",
                hover_data=['Launch Site'],
                title='Correlation between Payload and Success for all Sites')
    else:
        df = spacex_df[spacex_df['Launch Site']==str(launch_site)]
        fig = px.scatter(df[df['Payload Mass (kg)'].between(payload_mass[0], payload_mass[1])],
                x="Payload Mass (kg)",
                y="class",
                color="Booster Version Category",
                hover_data=['Launch Site'],
                title='Correlation between Payload and Success for Site {}'.format(launch_site))
    return(fig)
# Run the app
if __name__ == '__main__':
    app.run_server()
```

Appendix B – Other Downloadable Resources

1. spacex.csv
2. spacex_launch_dash.csv
3. spacex_dash_app.py
4. Spacex_dash_app.ipynb

Thank you!