IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Tiago Oliveira
28 january 2024

Outline

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

Executive Summary

- **Summary of methodologies**
  - Data Collection API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Visualization
  - Interactive Visual Analytics with Folium
  - Interactive Dashboard with Plotly Dash
  - Machine Learning Prediction
- **Summary of all results**
  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - Predictive Analytics results

Introduction

SpaceX is an innovative company that works on the space industry, providing a groundbreaking option of space travel that is much lower in cost when in comparison to others.

SpaceX advertises Falcon 9 rocket launches on its website with a cost of 63 million dollars, while other providers cost upwards of 165 million dollars each. Much of the savings are accomplished by the reuse of the first stage of the rockets. This is conquered by making the first stage land back on the ground.

As a data science on a startup called SpaceY, the goal of the project is to create a machine learning pipeline to predict the landing outcome of the first stage, in order to calculate the price of each launch.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data collected through SpaceX API and web scraping from Wikipedia
- Perform data wrangling
  - Categorical data was processed using one-hot encoding
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

Data collection was executed using two different types of methods, through the SpaceX REST API and Web Scraping from Wikipedia.

For the REST API, it is used a GET request in order to obtain the json response object and then transform it to a dataframe.

For the Web Scraping, it is used Beautiful Soup method to obtain the data from table HTML elements. Then it is parsed and, after, turned into a dataframe.

Data Collection – SpaceX API

- GET request for rocket data using API
- Convert json object into dataframe
- Some data cleaning and filling missing values

https://github.com/TiagoOliveira98/AppliedDataScienceCapstone/blob/main/Data_Collection_API_Lab.ipynb

```
[6]
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
[7]
response = requests.get(spacex_url)
```

```
[13]
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

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

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows that have multiple payloads in a single rocket.
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 in the list and replace the feature.
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 date leaving the time
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)]
```

Data Collection - Scraping

- GET request to get the HTML object of the Falcon 9 Wiki page
- Create a BeautifulSoup from the HTML response object
- Extract all columns names from HTML header
- Scrap all the data

https://github.com/TiagoOliveira98/AppliedDataScienceCapstone/blob/main/Data_Collection_with_Web_Scraping_lab.ipynb

```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url)
```

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.content)
```

```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# extend the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
for th in first_launch_table.find_all('th'):
    name = extract_column_from_header(th)
    if name is not None and len(name) > 0:
        column_names.append(name)
```

```
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:
            print(row)
            print(row[6].text)
```

Data Wrangling

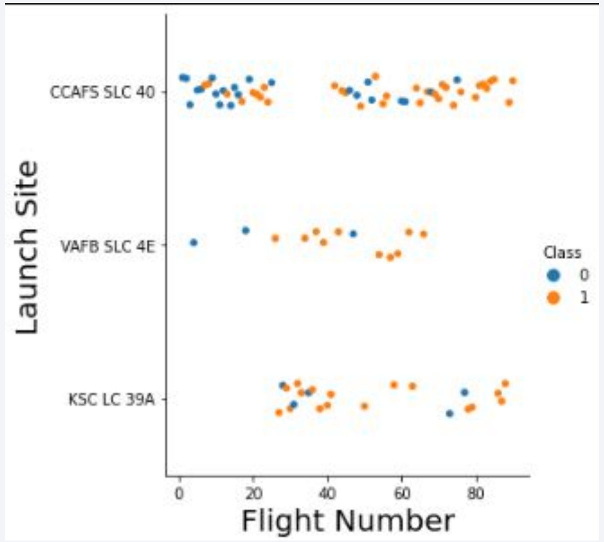
- Data wrangling corresponds to the process of cleaning data sets in order for easy access and Exploratory Data Analysis (EDA)
- Calculation of number of launches on each site and number of mission outcomes per orbit type
- Creation of landing outcomes label from outcome columns

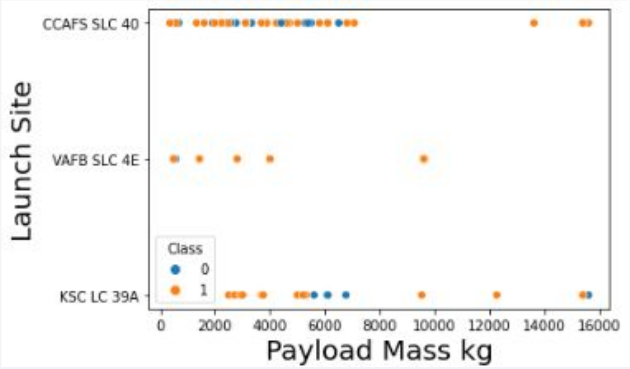
https://github.com/TiagoOliveira98/AppliedDataScienceCapstone/blob/main/Data_Wrangling.ipynb

EDA with Data Visualization

- Verified the relationship between attributes such as:
  - Payload and Flight Number
  - Flight Number and Launch Site
  - Payload and Launch Site
  - Flight Number and Orbit Type
  - Payload and Orbit Type
- Verification using scatter plots

https://github.com/TiagoOliveira98/AppliedDataScienceCapstone/blob/main/EDA_Dataviz.ipynb





EDA with SQL

- Queries executed on the database:
    - 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
    - Display average payload mass carried by booster version F9 v1.1
    - List the date when the first successful landing outcome in ground pad was achieved
    - 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 names of the booster_versions which have carried the maximum payload mass
    - 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 between the date 2010-06-04 and 2017-03-20, in descending order

https://github.com/TiagoOliveira98/AppliedDataScienceCapstone/blob/main/EDA_With_Python.ipynb

Build an Interactive Map with Folium

- For each launch site it was added a circle marker with the label of the launch site
- It was assigned the color Red to the class 0 of ‘launch_outcomes’ and the color Green to the class 1 of ‘launch_outcomes’. Using the MarkerCluster method the markers were added to each of the launch sites.
- Then it were created straight lines to visualize the distances between the launch sites and various landmarks such as railways, highways, cities and coastlines

https://github.com/TiagoOliveira98/AppliedDataScienceCapstone/blob/main/Launch_Site_Location.ipynb

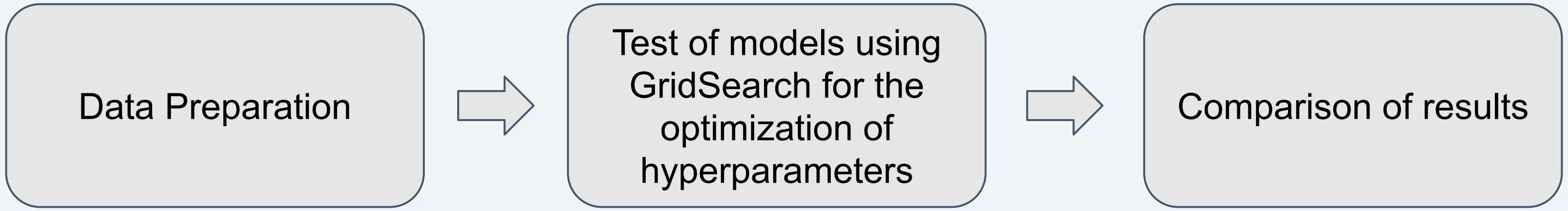
Build a Dashboard with Plotly Dash

- Following graphs were used to visualize the data:
  - Percentage of launches by site
  - Payload Range
- The graphs allowed for the analysis of the relation between payload and site, figuring out what would be the best sites for launch

- https://github.com/TiagoOliveira98/AppliedDataScienceCapstone/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- Four different models were compared to select the best performing one: Logistic Regression, Support Vector Machine, Decision Tree and K-nearest neighbors



https://github.com/TiagoOliveira98/AppliedDataScienceCapstone/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.ipynb

Results

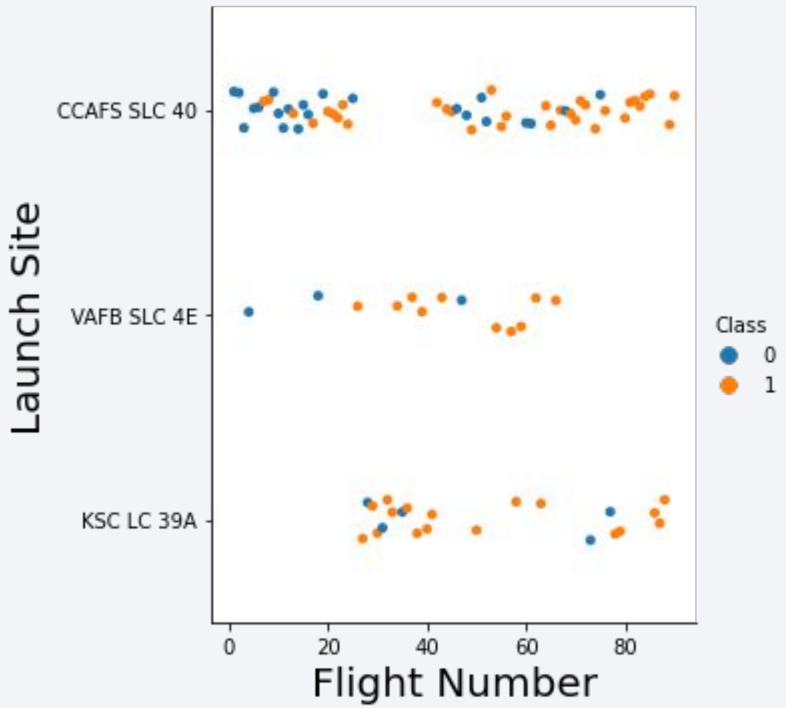
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

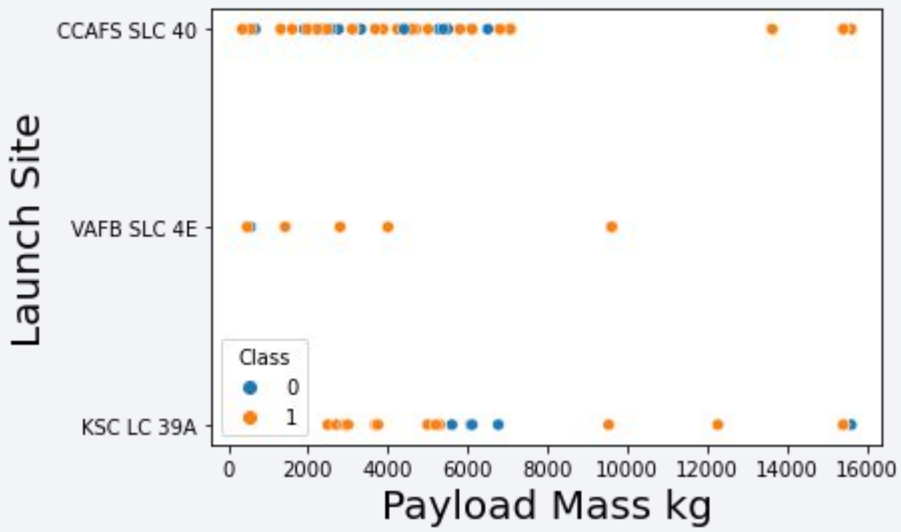
Flight Number vs. Launch Site

- Observed that VAFB SLC 4E launch site is maybe not used anymore since there are no high flight numbers



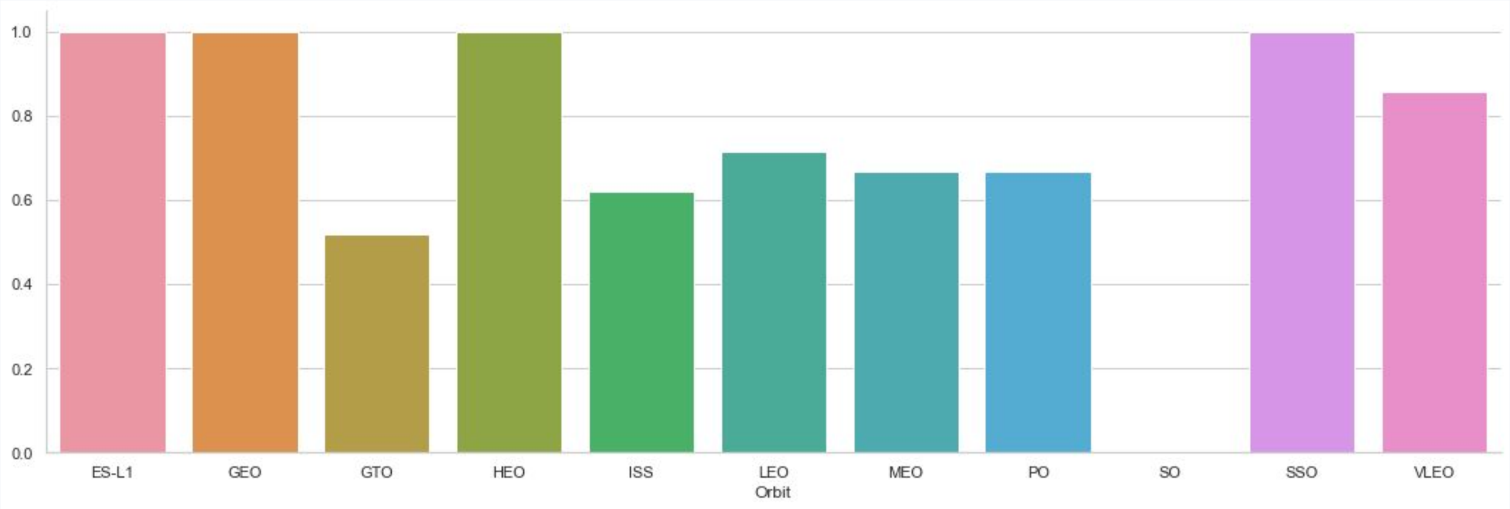
Payload vs. Launch Site

- The higher the payload may lead to a more probable successful landing.



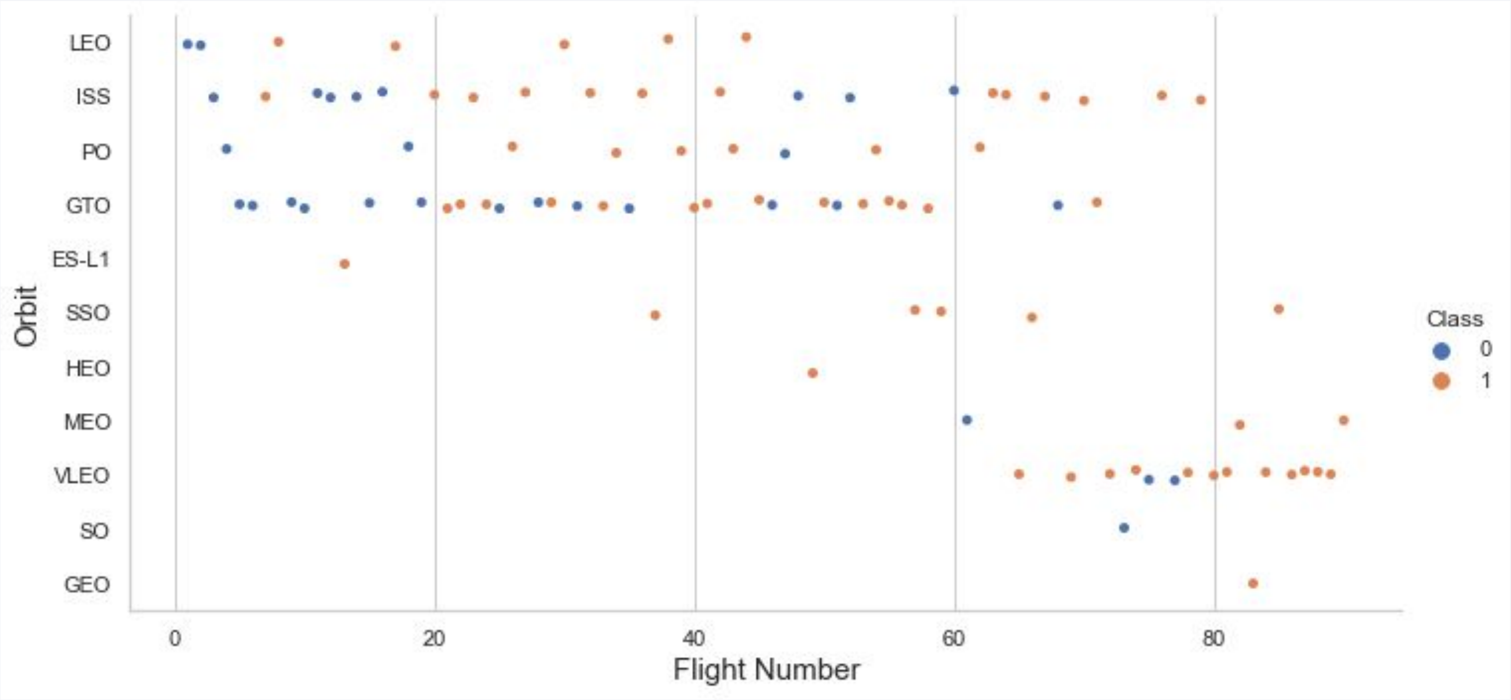
Success Rate vs. Orbit Type

- It is possible to see that 4 of the orbits have 100% success rate



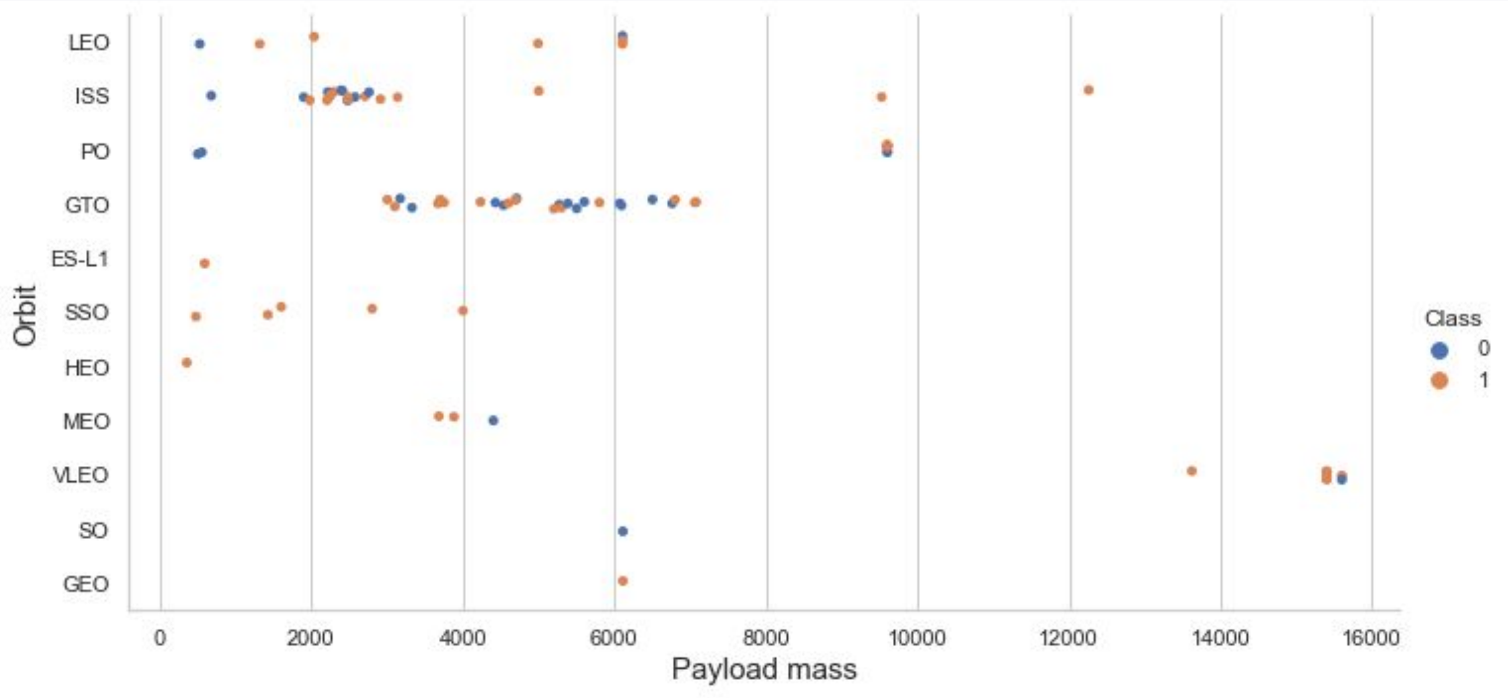
Flight Number vs. Orbit Type

- In general, the success rate increase with the number of flights.



Payload vs. Orbit Type

- In certain orbits the payload may have a great influence on the success rate



Launch Success Yearly Trend

- Show a line chart of yearly average success rate
- Show the screenshot of the scatter plot with explanations

All Launch Site Names

```
%sql select distinct("Launch_Site") from SPACEXTABLE
```

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

Explanation

The use of DISTINCT enables to clean the repeating values of Launch_Site

Launch Site Names Begin with 'CCA'

```
%sql select * from SPACEXTABLE where "Launch_Site" like "CCA%" limit 5
```

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

Explanation

The LIKE statement in this case enables the search for strings that start with ‘’CCA’

Total Payload Mass

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

sum("PAYLOAD_MASS__KG_")
99980

Explanation

Using the SUM statement we get the sum of all the entries in the column selected.

Average Payload Mass by F9 v1.1

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

avg("PAYLOAD_MASS__KG_")
2534.6666666666665

Explanation

After the execution of the Lab I saw that it was intended to give the mean for each type of F9 v1.1. So what I am showing is missing a group by statement

First Successful Ground Landing Date

```
%sql select min("Date") from (select * from SPACEXTABLE where "Mission_Outcome" = "Success")
# INSTEAD OF 'MISSION_OUTCOME' IT SHOULD BE 'LANDING_OUTCOME'
```

min("Date")
2010-06-04

Explanation

After the execution of the Lab I saw that it was intended to give the first observation that had the Landing outcome successful, not the Mission outcome.

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql select "Booster_Version" from (select * from SPACEXTABLE where "PAYLOAD_MASS__KG_" between 4000 and 6000)
```

Explanation

Usage of BETWEEN statement to refer to an interval

Booster_Version
F9 v1.1
F9 v1.1 B1011
F9 v1.1 B1014
F9 v1.1 B1016
F9 FT B1020
F9 FT B1022
F9 FT B1026
F9 FT B1030
F9 FT B1021.2
F9 FT B1032.1
F9 B4 B1040.1
F9 FT B1031.2
F9 B4 B1043.1
F9 FT B1032.2
F9 B4 B1040.2
F9 B5 B1046.2
F9 B5 B1047.2
F9 B5 B1046.3
F9 B5B1054
F9 B5 B1048.3
F9 B5 B1051.2
F9 B5B1060.1
F9 B5 B1058.2
F9 B5B1062.1

Total Number of Successful and Failure Mission Outcomes

```
%sql select "Mission_Outcome",count(*) as "Count" from SPACEXTABLE group by "Mission_Outcome"
```

Mission_Outcome	Count
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Explanation

Usage of GROUP BY statement for grouping the different Mission Outcomes

Boosters Carried Maximum Payload

```
%sql select "Booster_Version" from SPACEXTABLE where "PAYLOAD_MASS__KG_" = ( select max("PAYLOAD_MASS__KG_") from SPACEXTABLE )
```

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

Explanation

Usage of subquery to use the maximum value as a filter

2015 Launch Records

```
%sql select substr(Date, 6,2), "Landing_Outcome", "Booster_Version", "Launch_Site" from (select * from SPACEXTABLE where substr(Date,0,5)='2015' and "Mission_Outcome" like "Failure%")
```

Explanation

Usage of subquery to restrict the focus table to the entries that coincide with the intervals we want

substr(Date, 6,2)	Landing_Outcome	Booster_Version	Launch_Site
06	Precluded (drone ship)	F9 v1.1 B1018	CCAFS LC-40

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

```
%sql select "Landing_Outcome", count("Landing_Outcome") from SPACEXTABLE group by "Landing_Outcome" having "Date" between "2010-06-04" and "2017-03-20"
```

Landing_Outcome	count("Landing_Outcome")
Controlled (ocean)	5
Failure (drone ship)	5
Failure (parachute)	2
No attempt	21
Precluded (drone ship)	1
Success (drone ship)	14
Success (ground pad)	9
Uncontrolled (ocean)	2

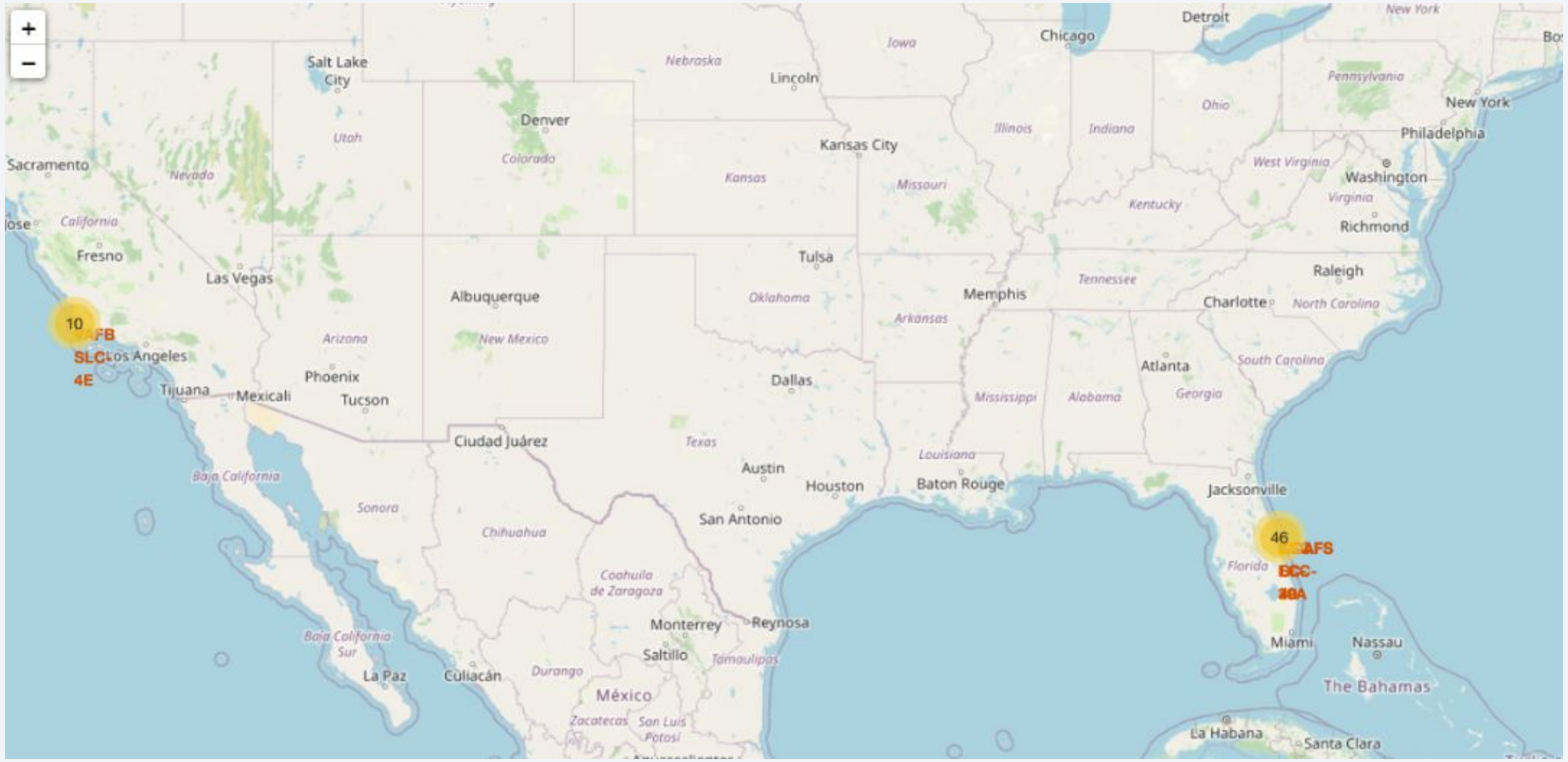
Explanation

Usage of GROUP BY statement to get count from each different Landing Outcome and HAVING statement to include within an interval

Section 3

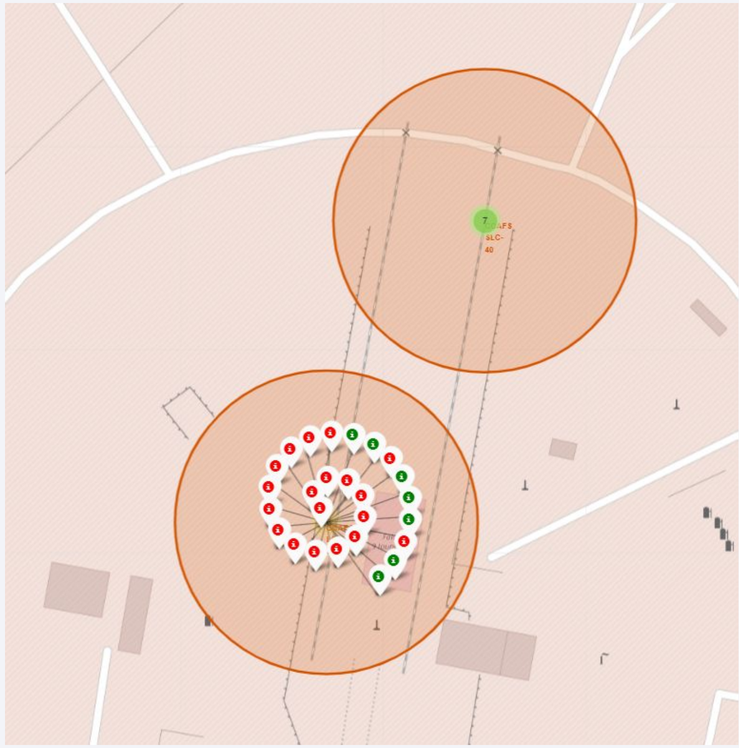
Launch Sites Proximities Analysis

Folium Map - Ground Stations



Stations are located near coastline

Folium Map - Color Labeled Markers



Green markers represent successful launches. Red markers represent unsuccessful launches

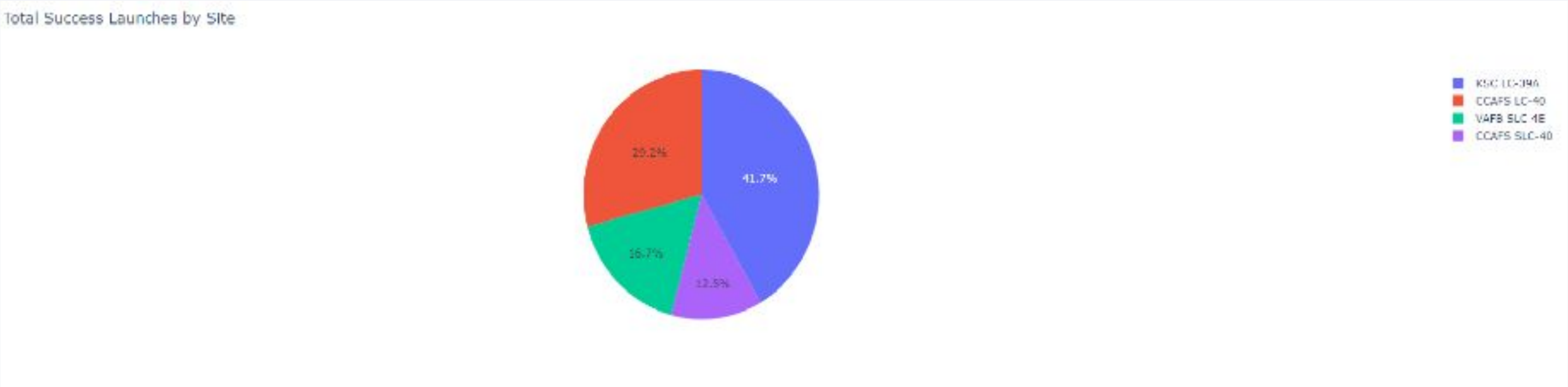
Folium Map - Distances between CCAFS SLC-40 and its proximities

In the image it is possible to verify that it is close to coastline.
All the other maps created show that the stations are far away from railways, highways and cities

Section 4

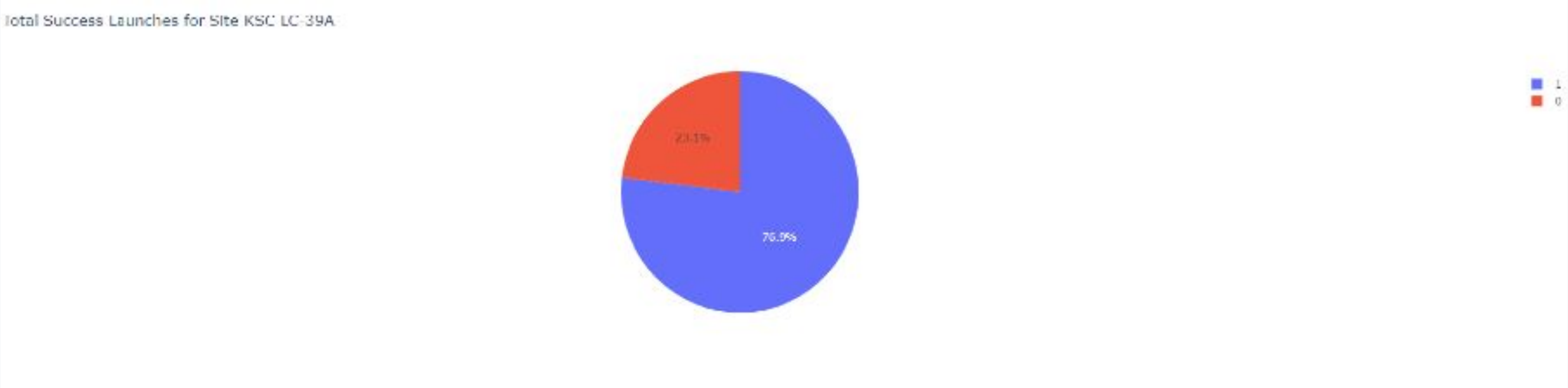
Build a Dashboard with Plotly Dash

Dashboard - Total success by Site



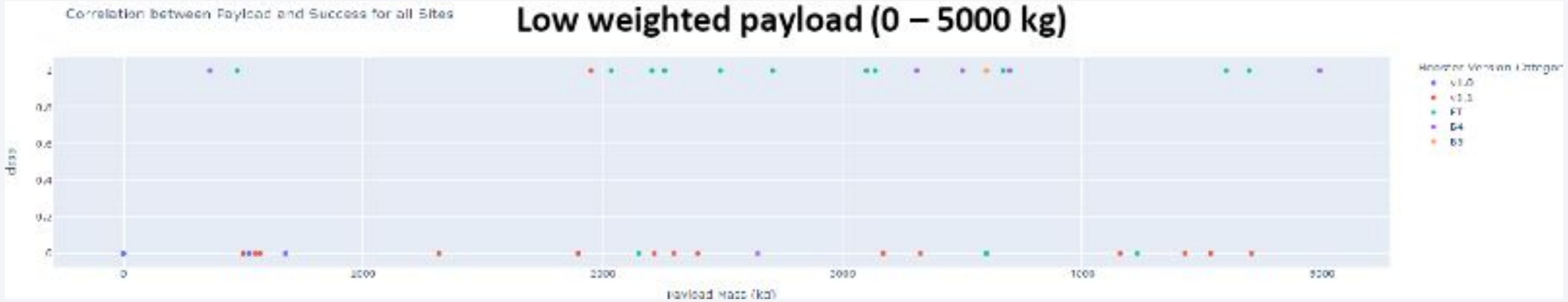
KSC LC-39A has the best success rate.

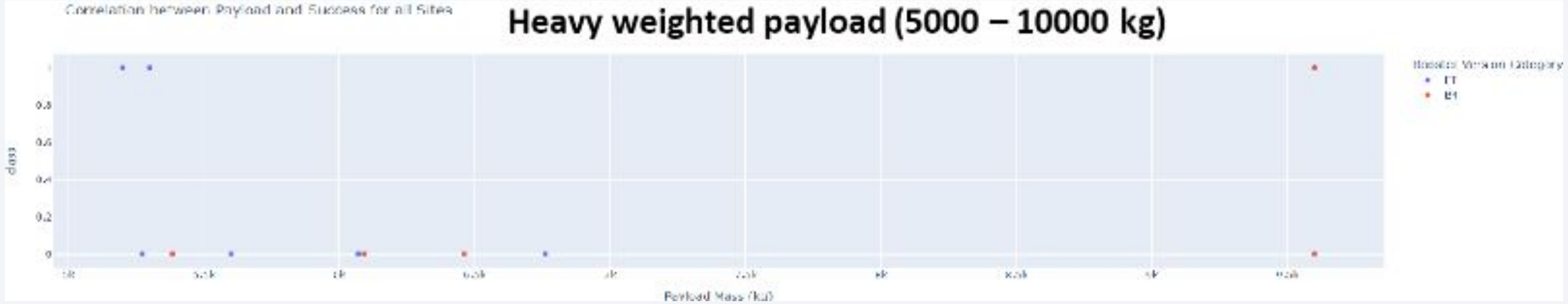
Dashboard - Total success launches for Site KSC LC-39A



KSC LC-39A has achieved a 76.9% success rate.

Dashboard - Payload mass vs. Outcome for all sites with different payload mass selected





Low payloads have a better success rate than heavy payloads

Section 5

Predictive Analysis (Classification)

Classification Accuracy

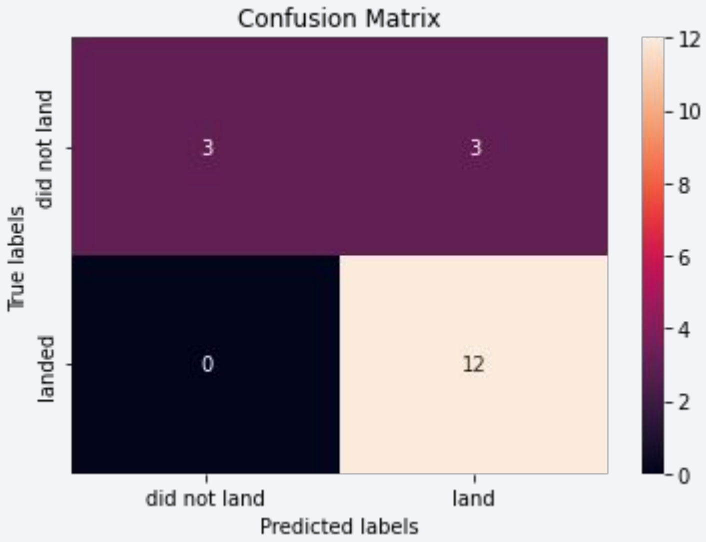
Model	Train Accuracy	Test Accuracy
Tree	0.8911	0.7222
KNN	0.8482	0.8333
SVM	0.8482	0.8333
LogReg	0.8464	0.8333

```
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
```

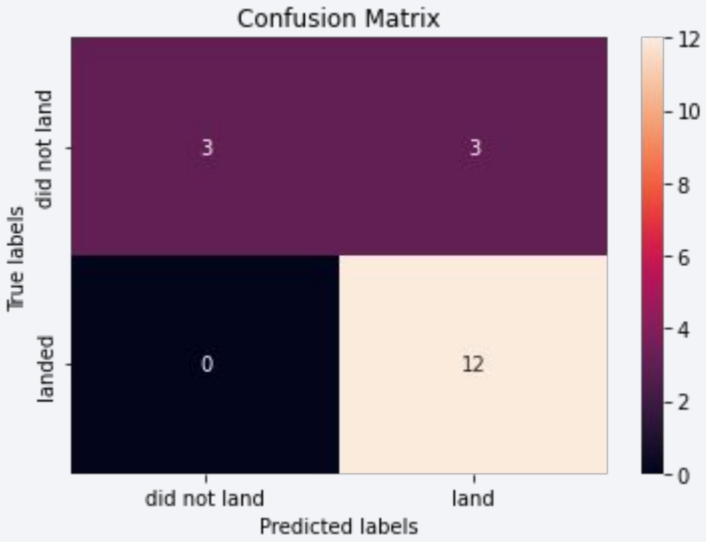
```
print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
print("accuracy :",svm_cv.best_score_)

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

Confusion Matrix

LogReg



SVM



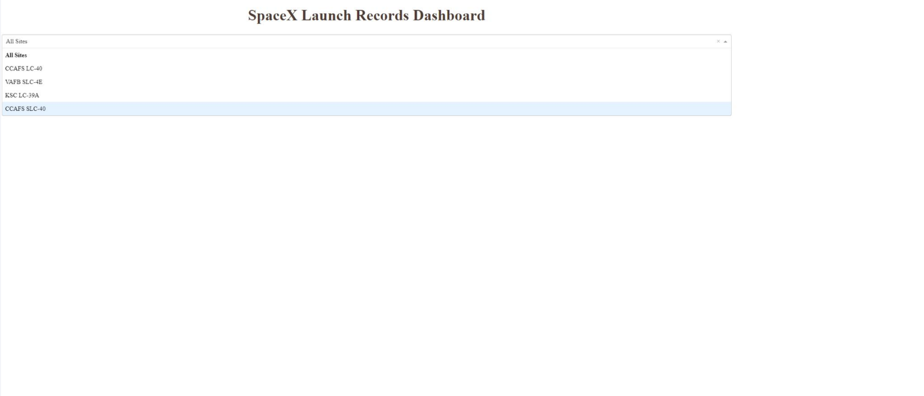
Tree

KNN

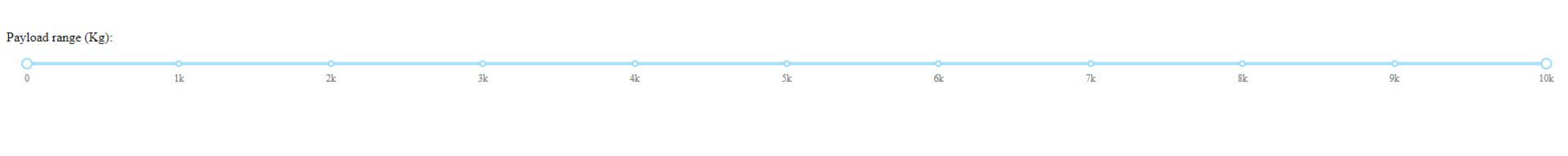
Conclusions

- The success of a mission can be explained by several factors like Launch site, orbit, number of previous launches.
- Some of the orbits have perfect or near-perfect success-rates
- Depending on orbits, the payloads may take a very important weight into the success of the mission.
- For this case, for the prediction of the success of the mission, it can be chosen both SVM or KNN for the model, since both have exactly the same training and testing accuracy. Just like equal confusion matrices

Appendix (Dashboard Images)




```python
# 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):
    filter_df = spacex_df
    if entered_site == 'ALL':
        fig = px.pie(filter_df, values='class',
        names='Launch Site',
        title='Success Count for All launch sites')
        return fig
    else:
        # return the outcomes piechart for a selected site
        filter_df=spacex_df[spacex_df['Launch Site']== entered_site]
        filter_df=filter_df.groupby(['Launch Site','class']).size().reset_index(name='class count')
        fig=px.pie(filter_df,values='class count',names='class',title=f"Total Success Launches for site {entered_site}")
        return fig
```




```python
# 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 scatter(entered_site,payload):
    filter_df = spacex_df[spacex_df['Payload Mass (kg)'].between(payload[0],payload[1])]
    # thought reusing filtered_df may cause issues, but tried it out of curiosity and it seems to be working fine

    if entered_site=='ALL':
        fig=px.scatter(filter_df,x='Payload Mass (kg)',y='class',color='Booster Version Category',title='Success count on Payload mass for All sites')
        return fig
    else:
        fig=px.scatter(filter_df[filter_df['Launch Site']==entered_site],x='Payload Mass (kg)',y='class',color='Booster Version Category',title=f"Success count on Payload mass for site {entered_site}")
        return fig
```

Thank you!