IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

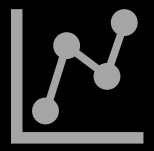
SAICHARAN K V
28-02-2022

Outline

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

Executive Summary

Followed methodologies like Web-Scrapping , Structured Query Language , Data Wrangling ,Also Applied Machine learning Algorithms.



Gone through many algorithms which could best fit and predict most accurate decision.

We were successfully able to predict the Falcon_9 stage 1 part could land on earth properly or not based on the data which we collected through Web Scrapping.

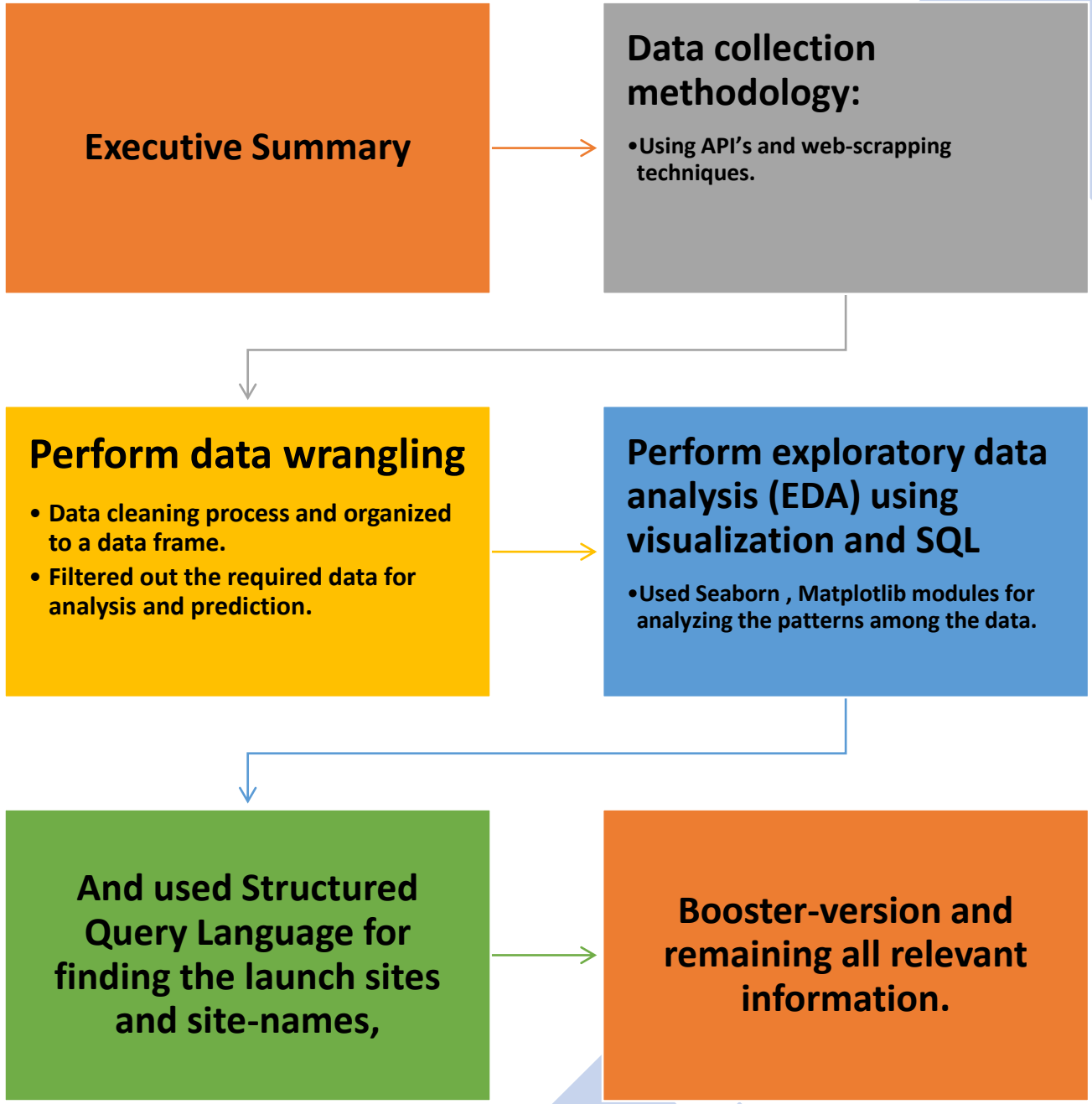
Introduction

- **Space-X** is a well-known company for its space works such as sending satellites , Manned missions, Missions for searching life on mars. It also manufactures and launches parts that could return to Earth. Company launches many spacecrafts to different needs to mankind .like Falcon-9 spacecraft that could return to earth after its work is done.
- Now the SpaceX company wants to know whether the Falcon-9 stage 1 part after separation in space could return to earth and land at their coordinated locations accurately or not.
- At which location we could predict high accuracy of success of landing space craft.

Section 1

Methodology

Methodology

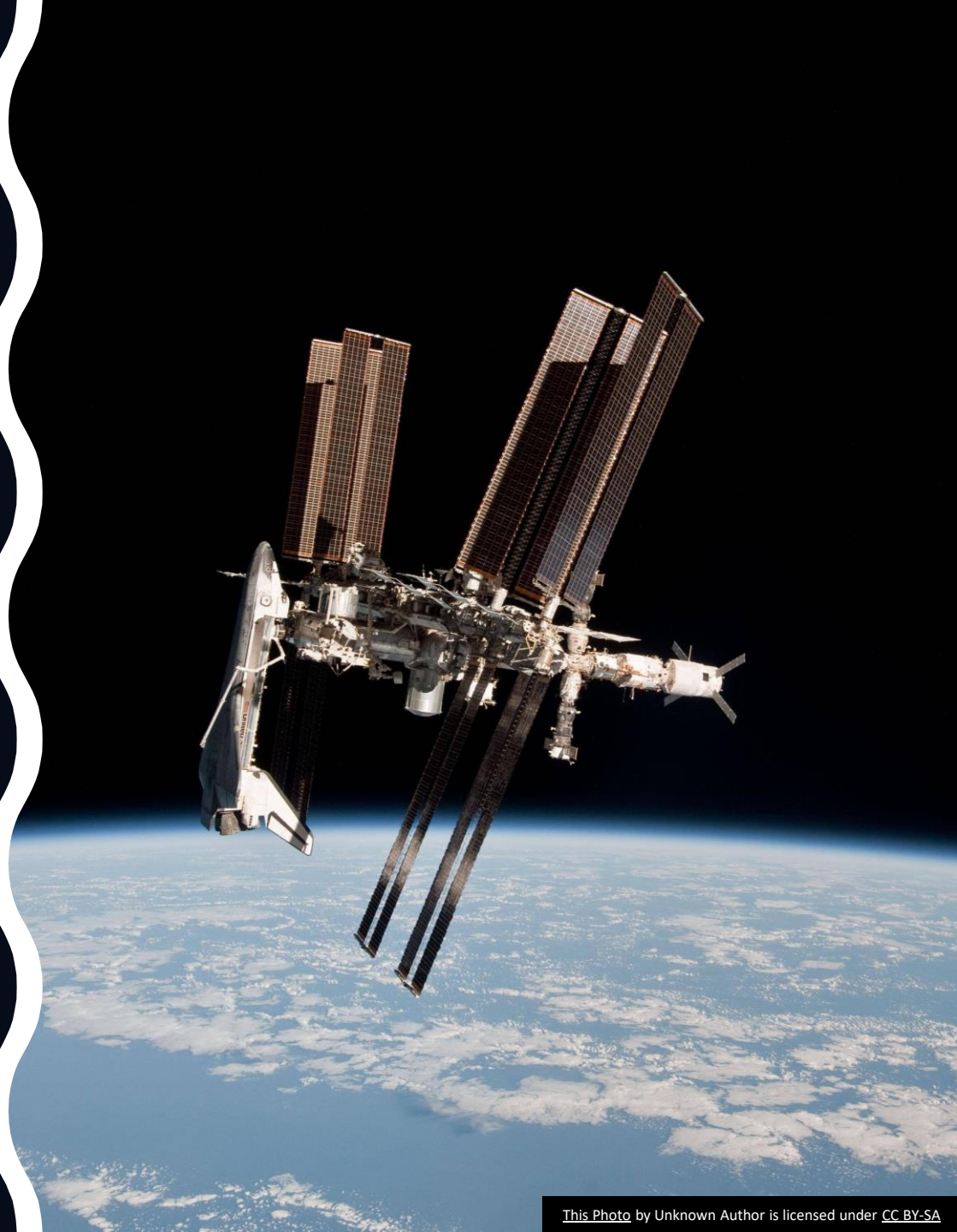


Perform interactive visual analytics using Folium and Plotly Dash

Used folium and Plotly Dash modules for identification of locations of Falcon-9 landing on a geographical map. And plotly dash for creating Dashboard of all sites landing locations.

Perform predictive analysis using classification models

1. Selecting Features and Target variables.
2. Standardize the data.
3.One-hot encoding.
3.Training and Splitting the data .
5.Comparing all classification algorithms with accuracy
6.Following with best algorithm.



Data Collection

- **Using API’s and user defined functions like booster-version and web scrapping using Beautiful soup .**
- **Data collected and stored to a data frame and then converted to Excel file.**

Data Collection – SpaceX API

GitHub Link:https://github.com/SAICHARANKV/IBM_CAPSTONE_PROJECT/blob/f716dabc3f6f2726cd8f543ef42cac1daaf966aa/IBM_API_CAPSTONE%20COLLECTION.ipynb

Flow chart of process followed

- Collecting the data with API call
- Converting to JSON
- Updating columns and rows (pre-processing)
- Filtering the data to keep only Falcon 9 launches
- Convert the data to csv file with name ‘Datas.et_part_1.csv’

Data Collection – SpaceX API

1. Collecting the data with API call

```
In [2]: # Takes the dataset and uses the rocket column to call the API and append the data to the list
def getBoosterVersion(data):
    for x in data['rocket']:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])
```

From the launchpad we would like to know the name of the launch site being used, the logitude, and the latitude.

```
In [3]: # Takes the dataset and uses the launchpad column to call the API and append the data to the list
def getLaunchSite(data):
    for x in data['launchpad']:
        response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
        Longitude.append(response['longitude'])
        Latitude.append(response['latitude'])
        LaunchSite.append(response['name'])
```

2. Converting to JSON

```
In [2]: # Takes the dataset and uses the rocket column to call the API and append the data to the list
def getBoosterVersion(data):
    for x in data['rocket']:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])
```

3. Updating columns and rows (pre-processing)

Then, we need to create a Pandas data frame from the dictionary launch_dict.

```
In [40]: # Create a data from launch_dict
df=pd.DataFrame(launch_dict)
```

Show the summary of the dataframe

```
In [41]: # Show the head of the dataframe
df.head()
```

Out[41]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount
0	1	2006-03-24	Falcon 1	20.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0
1	2	2007-03-21	Falcon 1	NaN	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0
2	4	2008-09-28	Falcon 1	165.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0
3	5	2009-07-13	Falcon 1	200.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0

4. Filtering the data to keep only Falcon 9 launches

launches. Save the filtered data to a new dataframe called "data_falcon9".

```
In [46]: # Hint data['BoosterVersion']!='Falcon 1'
data_falcon9=df[df['BoosterVersion']!='Falcon 1']
```

Now that we have removed some values we should reset the FlgihtNumber column

```
In [47]: data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

5. Convert the data to csv file with name 'Datas.et_part_1.csv'

```
data_falcon9.to_csv('dataset_part\_1.csv', index=False)
```

Data Collection - Scraping

GitHub Repo:https://github.com/SAICHARANKV/IBM_CAPSTONE_PROJECT/blob/df7fcadaea8921814c192a72783beb59cd2157e9/jupyter-labs-webscraping%20(1).ipynb

1. Importing Beautiful Soup

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

Create a BeautifulSoup object from the HTML response

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

2.Getting column names

```
In [9]: # Use the find_all function in the BeautifulSoup object, with element type `table`
html_tables=[]
for li in soup.find_all("table"):
    print(li.text, end=" ")
    # html_tables.append(li.text)
# Assign the result to a list called `html_tables`
#html_tables
```

Starting from the third table is our target table contains the actual launch

```
In [10]: # Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

3.Creating Launch_dict

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

4.Data frame creation

```
5]: df=pd.DataFrame(launch_dict)
```

5.Creating CSV file

```
df.to_csv('spacex_web_scraped.csv', index=False)
```

Data Wrangling

GitHub Repo:
https://github.com/SAICHARANKV/IBM_CAPSTONE_PROJECT/blob/0f0b3d13208514f6d10d17fbce087be023df27c3/spacex-Data%20wrangling.ipynb

1.Loading Data Set

```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_1.csv")
df.head(10)
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	Lar
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	Nal
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	Nal
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	Nal
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	Nal
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	Nal
5	6	2014-01-06	Falcon 9	3325.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	Nal
6	7	2014-04-18	Falcon 9	2296.000000	ISS	CCAFS SLC 40	True Ocean	1	False	False	True	Nal

2.Used value_count() for determining no of outcomes and occurrences in orbit

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

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

3.Creating landing outcome variable from outcome and assigning to data frame

```
In [23]: # landing_class = 0 if bad_outcome
         # landing_class = 1 otherwise
         landing_class=[]
         for i,outcome in enumerate(landing_outcomes.keys()):
             if outcome in bad_outcomes:
                 landing_class.append(0)
             else:
                 landing_class.append(1)

         df['Class']=pd.DataFrame(landing_class)
```

This variable will represent the classification variable that rep
stage did not land successfully; one means the first stage lar

```
In [25]: #df['Class']=landing_class
         df[['Class']].head(8)
```

Out[25]:

	Class
0	1.0
1	0.0
2	1.0
3	0.0

4.Converting modified DF to CSV file

```
]: df["Class"].mean()
```

```
]: 0.375
```

We can now export it to a CSV for the next se
pre-selected date range.

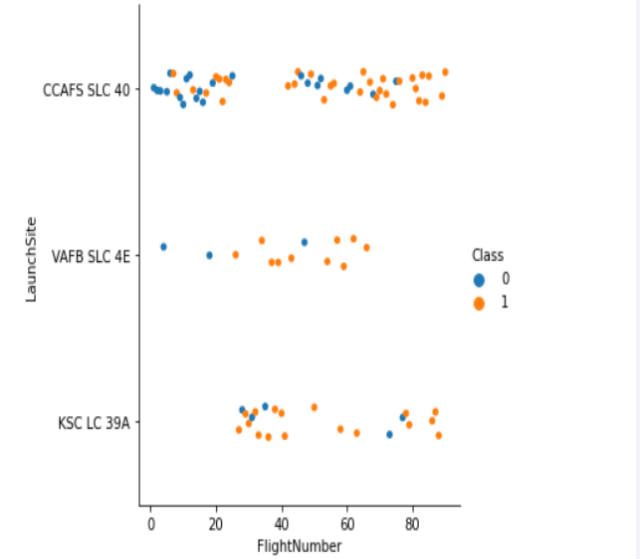
```
df.to_csv("dataset_part\_2.csv", index=False)
```

EDA with Data Visualization

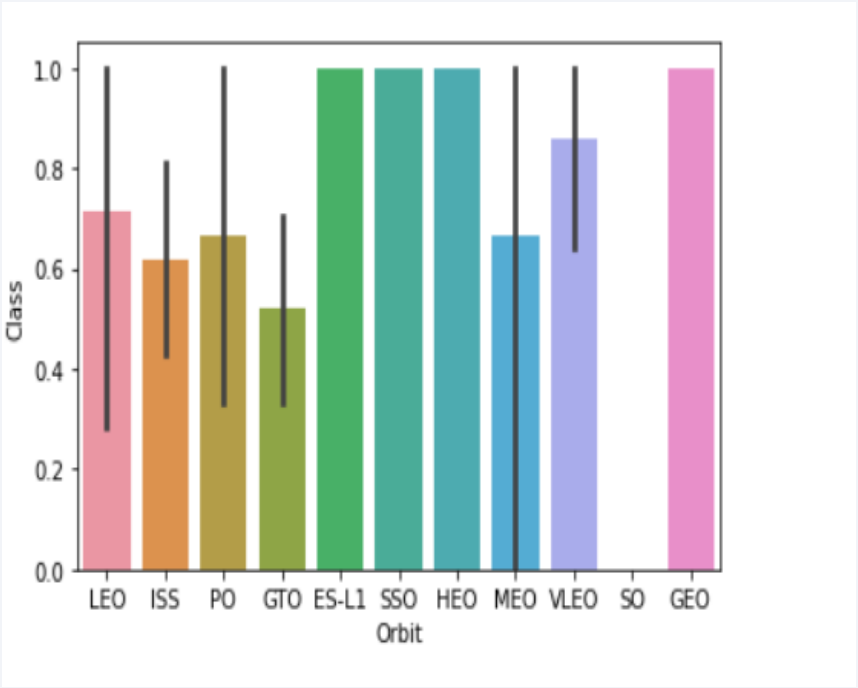
1.Scatter plot between Launchsite vs Flightnumber

```
sns.catplot(y="LaunchSite",x="FlightNumber",hue="Class",data=df)
```

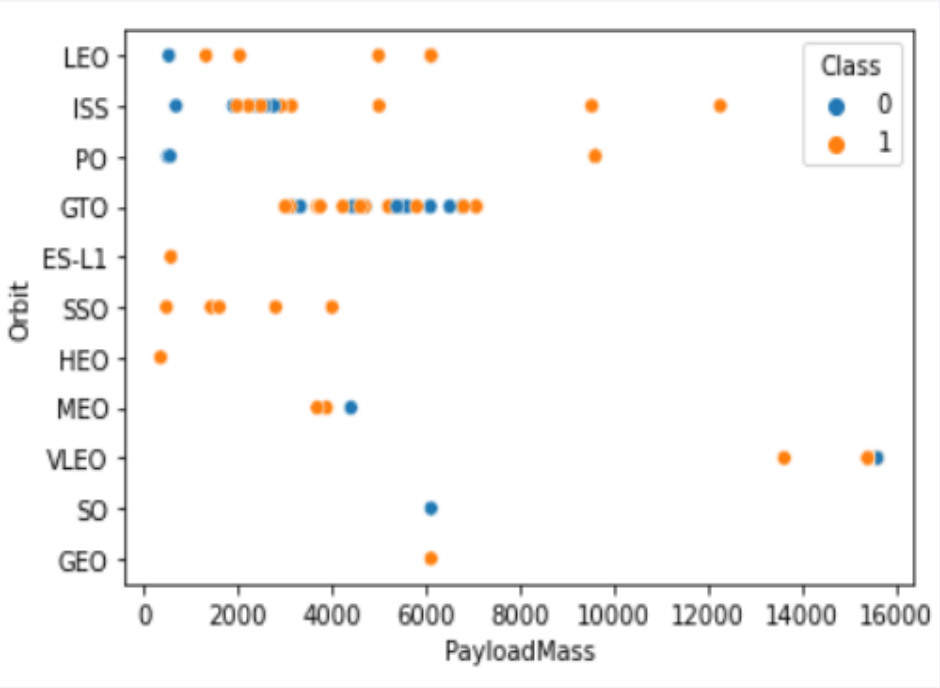
```
<seaborn.axisgrid.FacetGrid at 0x7fa56c35a490>
```



2.Relationship between Orbits vs Class parameters



3.Scatter plot between Payloadmass vs orbit



4.ONE_HOT_ENCODING ON FEATURES

```
features_one_hot=pd.get_dummies(features)
features_one_hot.head()
```

	FlightNumber	PayloadMass	Flights	GridFins	Reused	Legs	Block	ReusedCount	Orbit_ES-L1	Orbit_GEO	...	Serial_E
0	1	6104.959412	1	False	False	False	1.0	0	0	0	...	0
1	2	525.000000	1	False	False	False	1.0	0	0	0	...	0
2	3	677.000000	1	False	False	False	1.0	0	0	0	...	0
3	4	500.000000	1	False	False	False	1.0	0	0	0	...	0

5.Converting all numerical categories to Float Data type.

```
features_one_hot.astype('float64')
```

	FlightNumber	PayloadMass	Flights	GridFins	Reused	Legs	Block	ReusedCount	Orbit_ES-L1	Orbit_GEO	...	Serial_
0	1.0	6104.959412	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0
1	2.0	525.000000	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0
2	3.0	677.000000	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0
3	4.0	500.000000	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0
4	5.0	3170.000000	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...

6.Converting Data Frame to CSV

```
features_one_hot.to_csv('dataset_part\_3.csv', index=False)
```

EDA with SQL

GitHub Repo:
https://github.com/SAICHARANKV/IBM_CAPSTONE_PROJECT/blob/ffc42028934fa929064dfc760dd8356d77e43160/EDA_With_SQL.ipynb

I used SQL queries to answer the following questions:

- 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 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 the names of the booster_versions which have carried the maximum payload mass. Use a subquery
- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for the 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

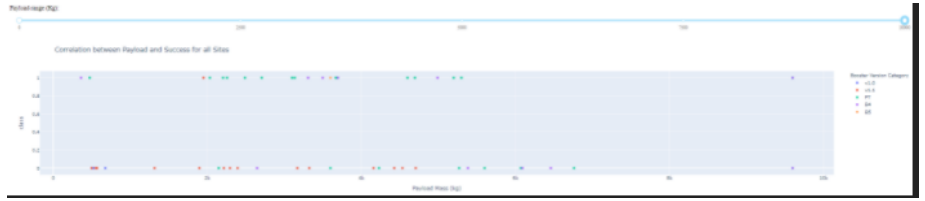
Build an Interactive Map with Folium

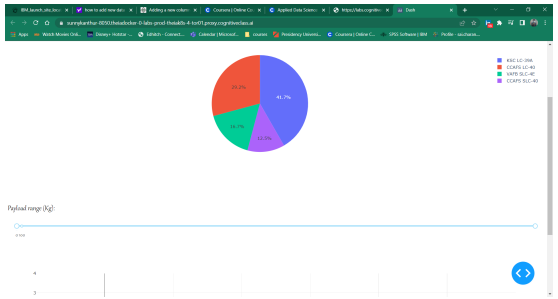
- **folium.Marker() was used to create marks on the maps.**
- **folium.Circle() was used to create a circles above markers on the map.**
- **folium.Icon() was used to create an icon on the map.**
- **folium.PolyLine() was used to create polynomial line between the points.**
- **folium.plugins.AntPath() was used to create animated line between the points.**
- **markerCluster() was used to simplify the maps which contain several markers with identical coordination.**

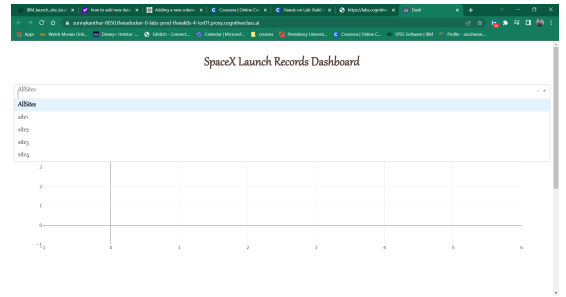
GitHub Repo: - https://github.com/SAICHARANKV/IBM_CAPSTONE_PROJECT/blob/84ceb2c5861cfa2ec563a1d05396d0d340e6f4c1/IBM_launch_site_location_week3.ipynb

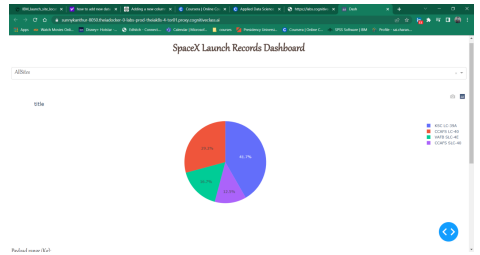
Build a Dashboard with Plotly Dash

- • Dash and html components were used as they are the most important thing and almost everything depends on them, such as graphs, tables, dropdowns, etc.
- • Pandas was used to simplifying the work by creating dataframe.
- • Plotly was used to plot the graphs.
- • Pie chart and scatter chart were used to for plotting purposes.
- • Rangeslider was used for payload mass range selection.
- • Dropdown was used for launch sites.









GitHub repo:
https://github.com/SAICHARANKV/IBM_CAPSTONE_PROJECT/blob/4469f38c1dde3d08a7dbe8ae341e479ac7c02d81/Ploty_%20DashBoard/spacex_dash_app.py

Predictive Analysis (Classification)

1. Building the model

- Create column for the class
- Standardize the data
- Split the data info train and test sets
- Build GridSearchCV model and fit the data

GitHub repo:
https://github.com/SAICHARANKV/IBM_CAPSTONE_PROJECT/blob/c4268764be3dc65b69b759d10ab050bafed9547b/SpaceX_Machine%20Learning%20Prediction.ipynb

2. Evaluating the model

- Calculating the accuracies
- Calculating the confusion matrixes
- Plot the results

3. Finding the optimal model

- Find the best hyperparameters for the models
- Find the best model with highest accuracy
- Confirm the optimal model

Results

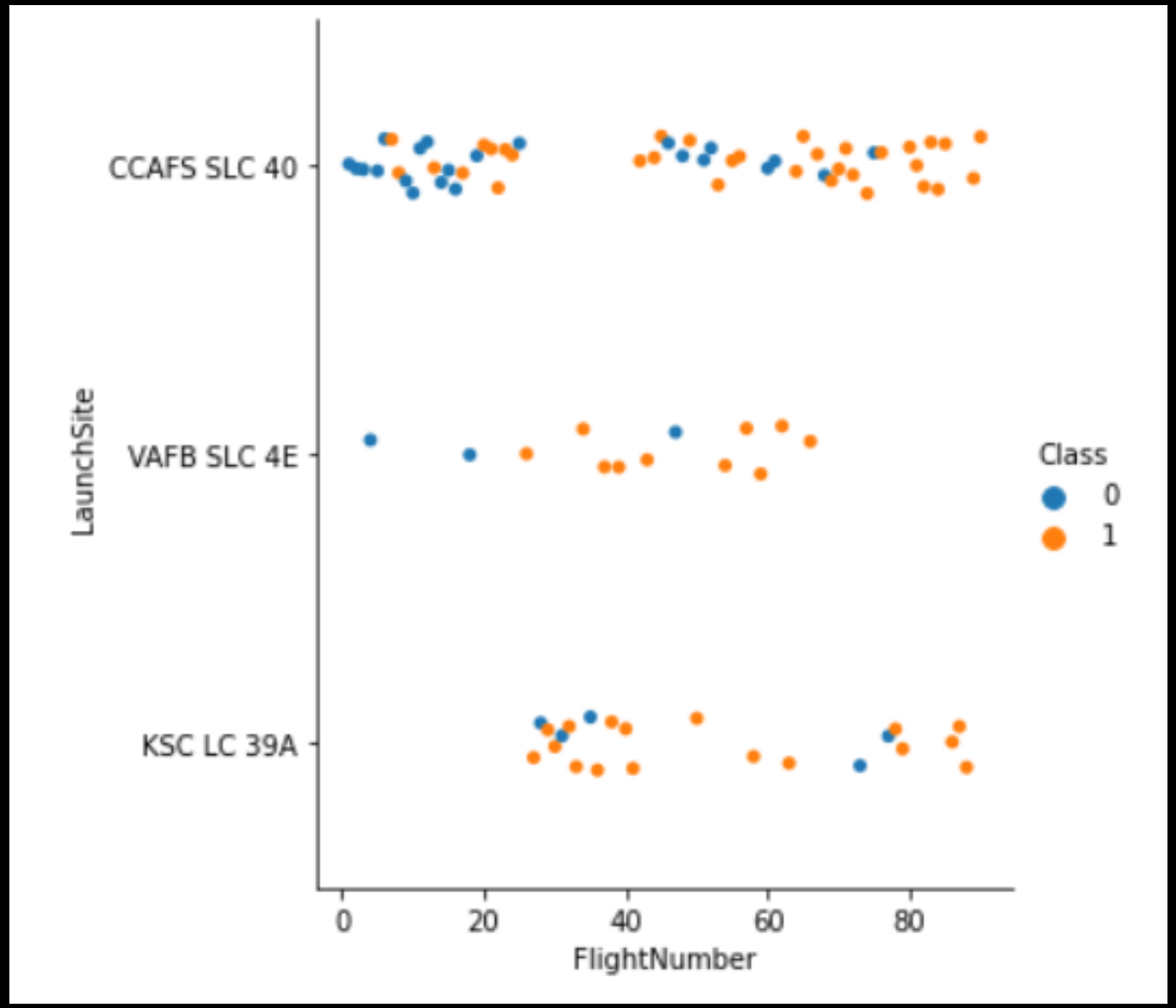
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

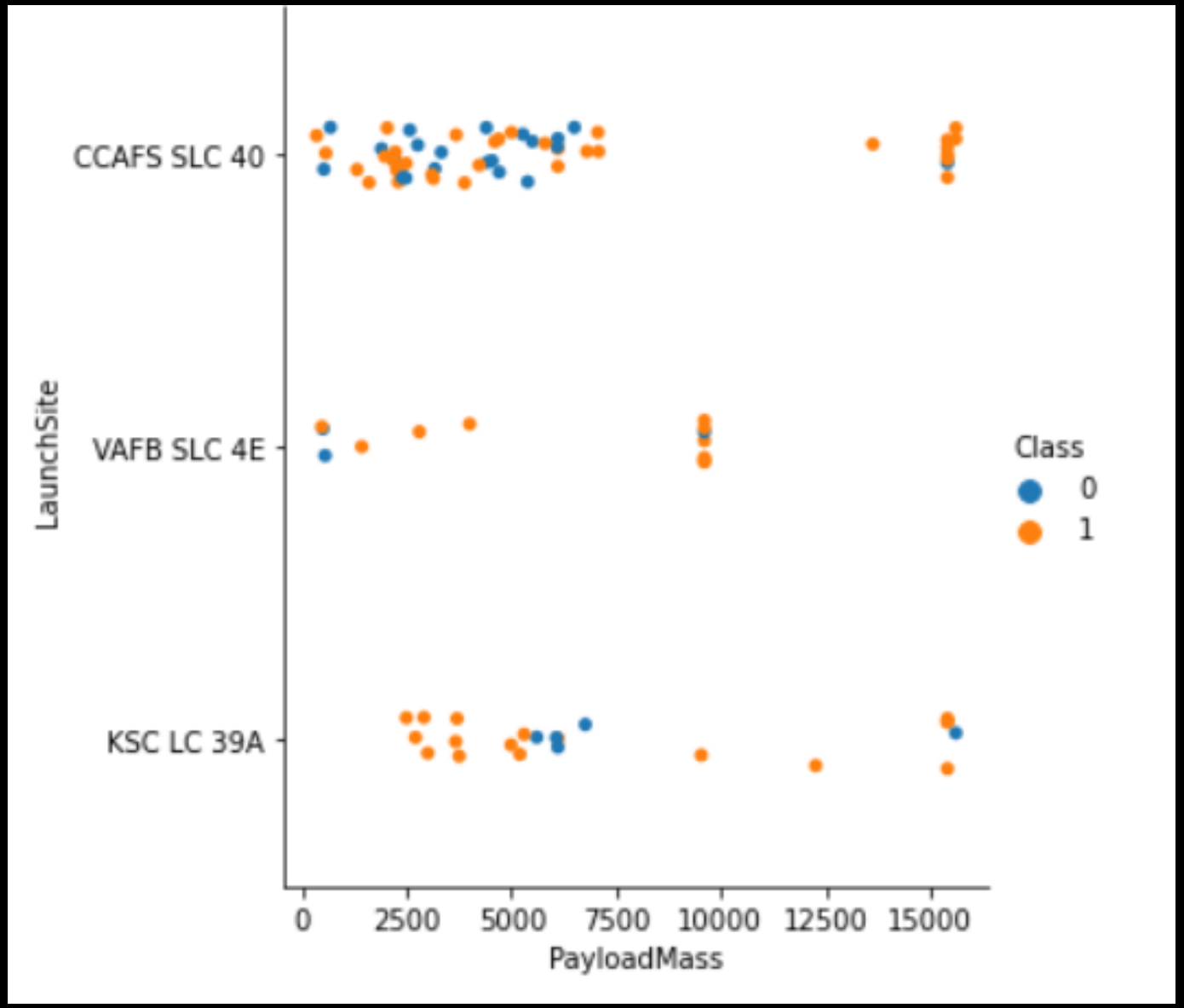
Flight Number vs. Launch Site

With the increase of flight number, the success rate is increasing as well in the launch site

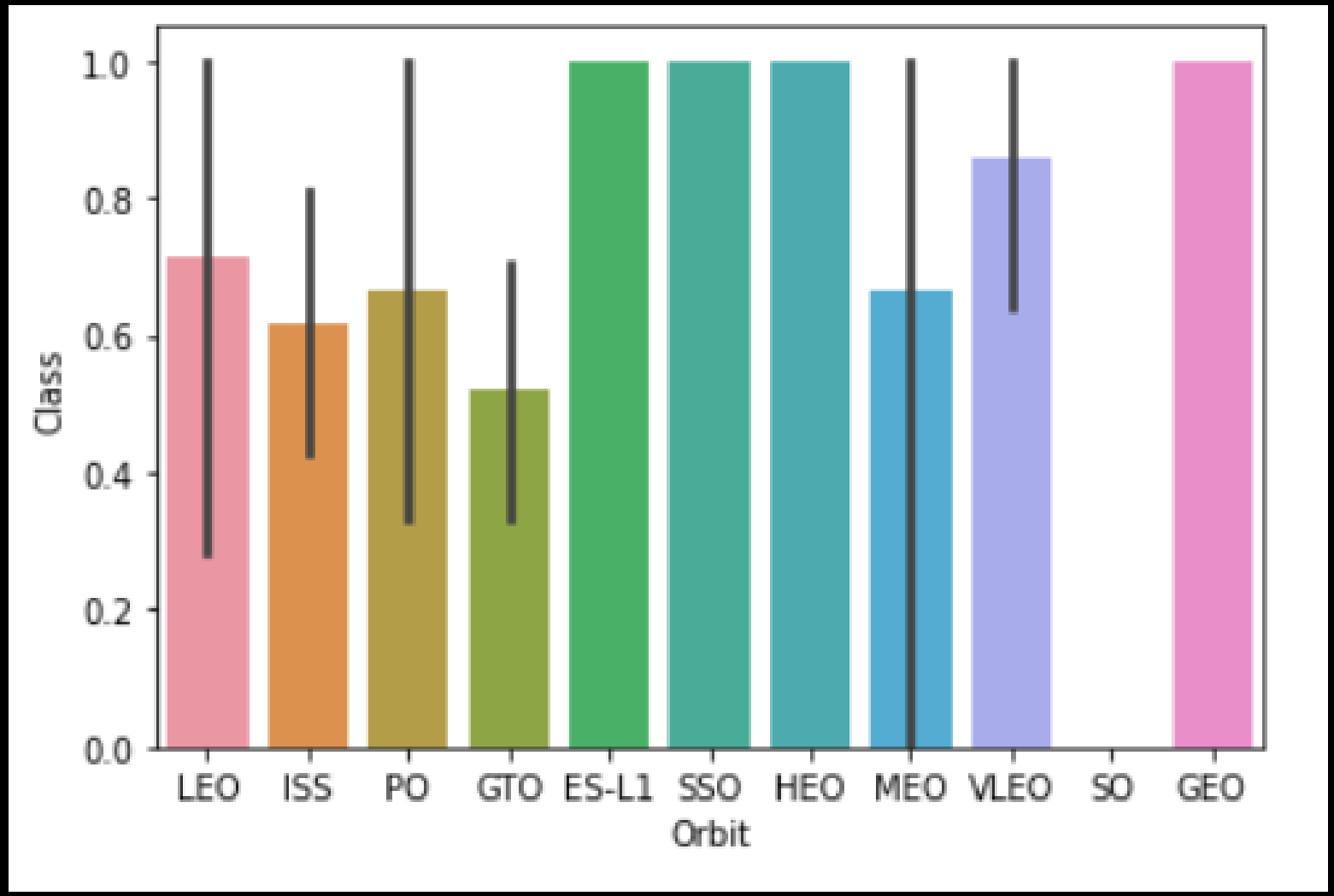


Payload vs. Launch Site

With the increase of Pay load Mass, the success rate is increasing as well in the launch sites



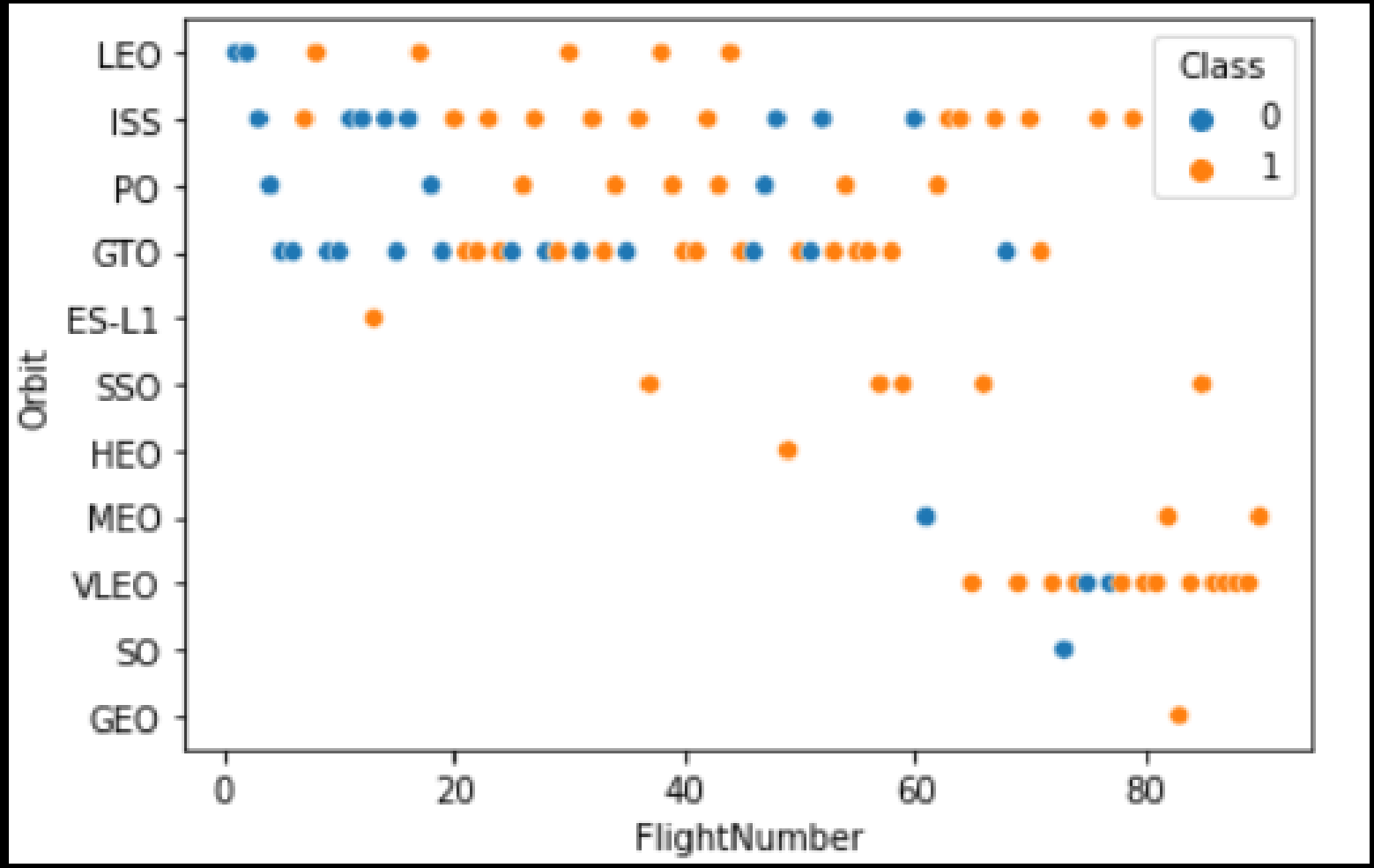
Success Rate vs. Orbit Type



ES-L1, GEO, HEO, and SSO have a success rate of 100%

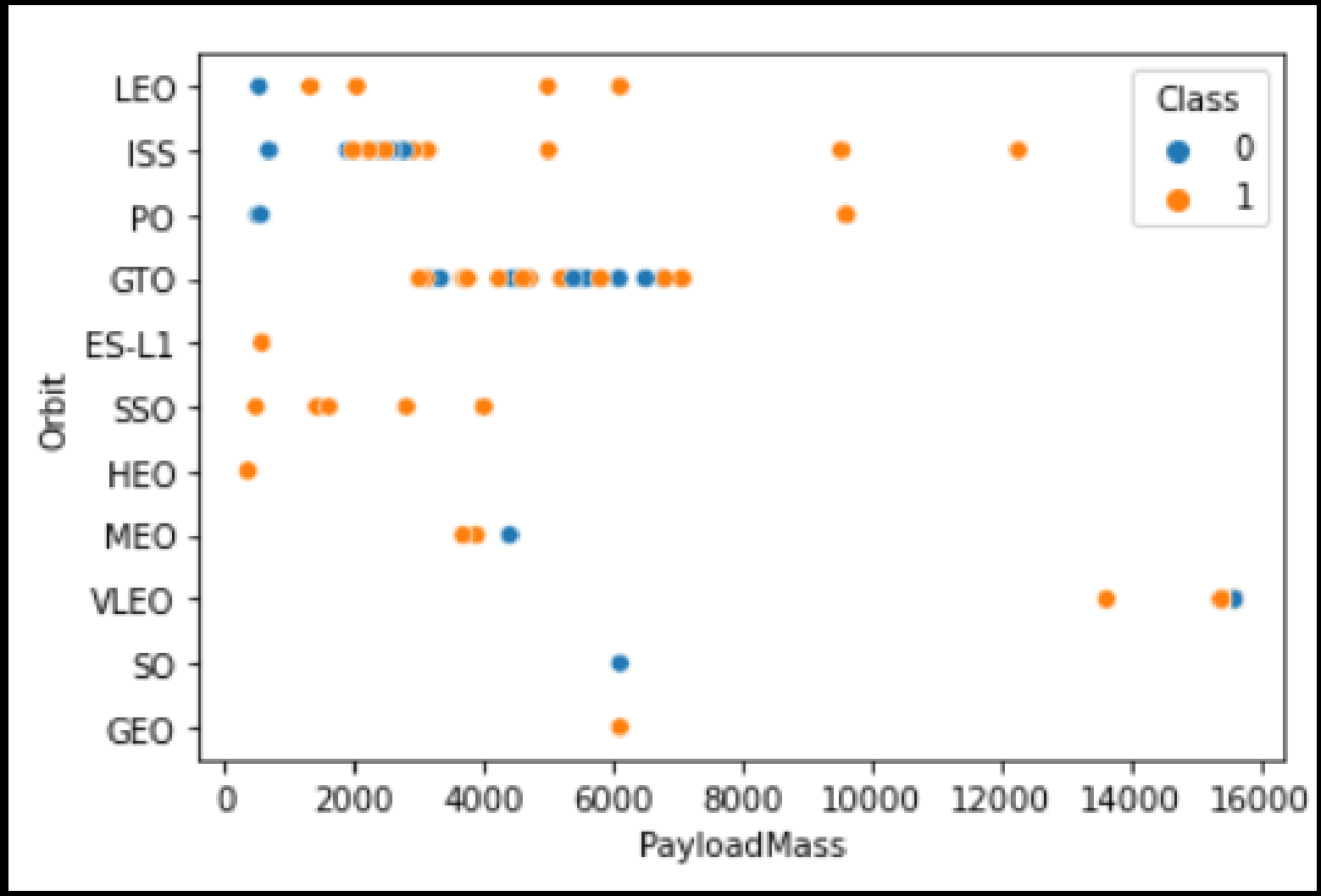
SO has a success rate of 0%

Flight Number vs. Orbit Type



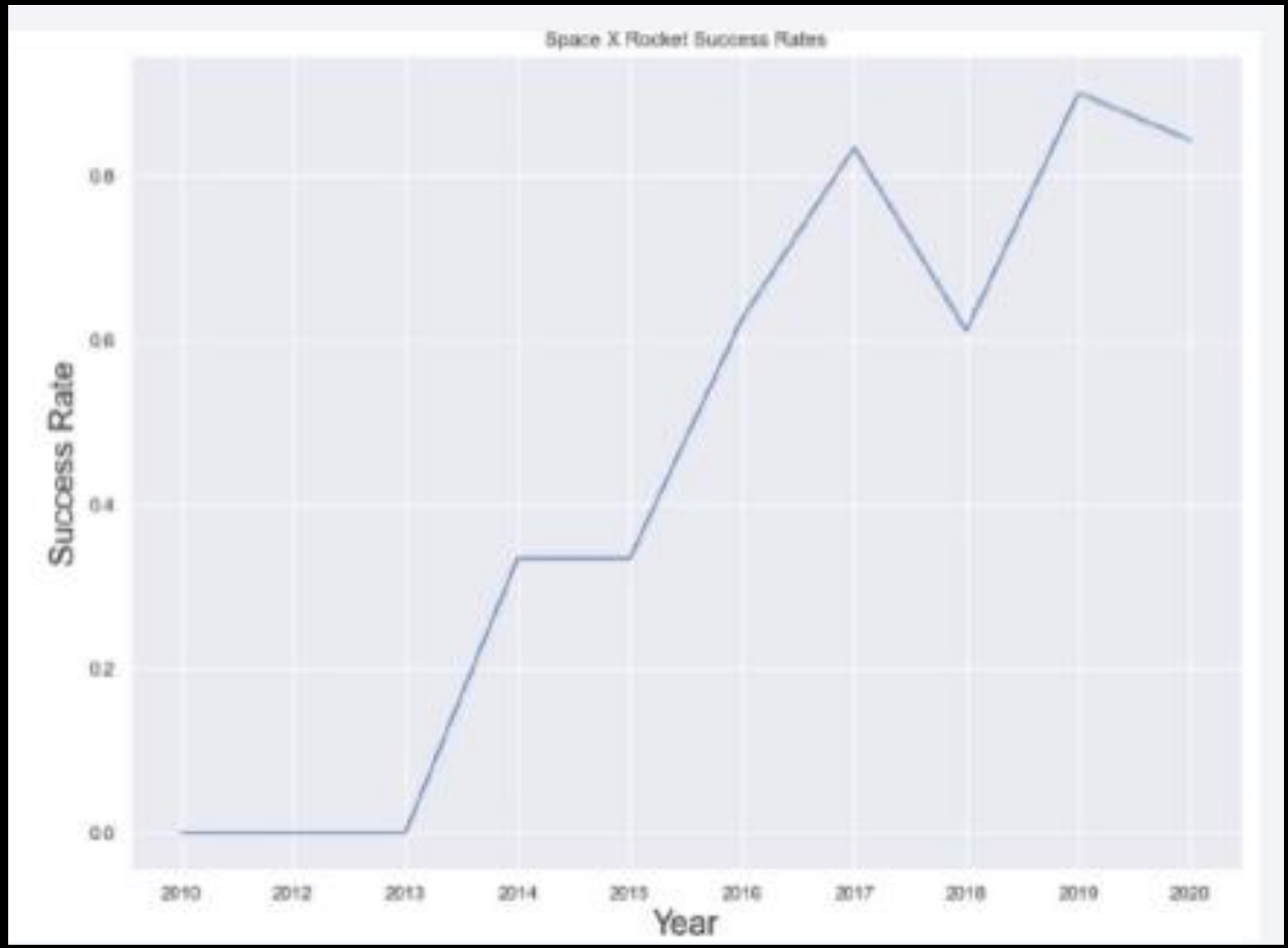
We can’t determine any relationship between Flighnumber and Orbit name

Payload vs. Orbit Type



More than 10000kg mass of Rocket and if they are projected to orbits ISS and VLEO only they can return successfully.

Launch Success Yearly Trend



Year wise success rate in improving due to increase in technology and errors rectification.

```
%sql select Unique(LAUNCH_SITE) from SPACEXTBL_2;
```

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

All Launch Site Names

Launch Site Names Begin with 'CCA'

```
%sql SELECT Launch_site from SPACEXTBL_2 where (LAUNCH_SITE) LIKE 'CCA%' LIMIT 5
```

```
 * ibm_db_sa://dsc14704:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb
Done.
```

launch_site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Total Payload Mass

```
%sql select sum(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL_2;
```

```
* ibm_db_sa://dsc14704:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1
ases.appdomain.cloud:31505/bludb
Done.
```

payloadmass
619967

Average Payload Mass by F9 v1.1

```
%sql select avg(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL_2;
```

```
* ibm_db_sa://dsc14704:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l
ases.appdomain.cloud:31505/bludb
Done.
```

payloadmass
6138

First Successful Ground Landing Date

```
%sql select min(DATE) from SPACEXTBL_2;
```

```
 * ibm_db_sa://dsc14704:***@ea286ace-86c7-4d5b-
ases.appdomain.cloud:31505/bludb
Done.
```

1
01-03-2013

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql select BOOSTER_VERSION from SPACEXTBL_2 where LANDING__OUTCOME='Success (drone ship)' and PAYLOAD_MASS__KG_ BETWEEN 4000
```

```
 * ibm_db_sa://dsc14704:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb
Done.
```

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

Total Number of Successful and Failure Mission Outcomes

```
%sql select count(MISSION_OUTCOME) as missionoutcomes from SPACEXTBL_2 GROUP BY MISSION_OUTCOME;
```

```
 * ibm_db_sa://dsc14704:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:
Done.
```

missionoutcomes
1
99
1

Boosters Carried Maximum Payload

```
%sql select BOOSTER_VERSION as boosterversion from SPACEXTBL_2 where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from SPACE
```

```
 * ibm_db_sa://dsc14704:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb
Done.
```

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

2015 Launch Records

```
%sql SELECT month(DATE) as Month, BOOSTER_VERSION, LAUNCH_SITE FROM SPACEX WHERE year(DATE) = '2015' AND \
LANDING__OUTCOME = 'Failure (drone ship)';
```

MONTH	booster_version	launch_site
1	F9 v1.1 B1012	CCAFS LC-40
4	F9 v1.1 B1015	CCAFS LC-40

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

```
%sql SELECT LANDING_OUTCOME as "Landing Outcome", COUNT(LANDING_OUTCOME) AS "Total Count" FROM SPACEX \
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' \
GROUP BY  LANDING_OUTCOME \
ORDER BY COUNT(LANDING_OUTCOME) DESC ;
```

Landing Outcome	Total Count
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

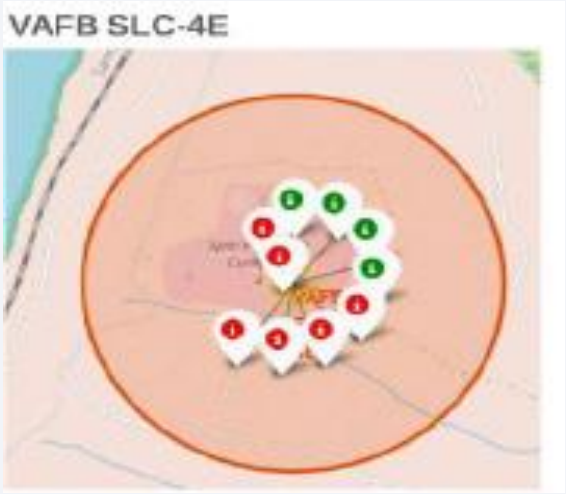
By using “ORDER” we can order the values in descending order, and with “COUNT” we can count all numbers as we did previously

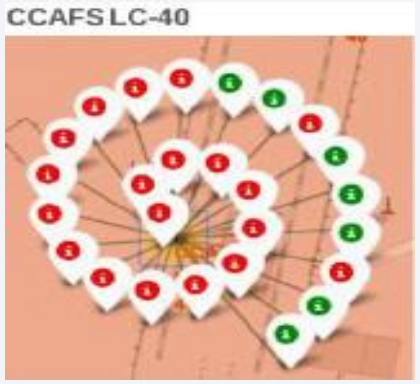

Section 3
Launch Sites
Proximities Analysis

All Launch Sites’ Location Markers

All the launches are near USA, Florida, and California

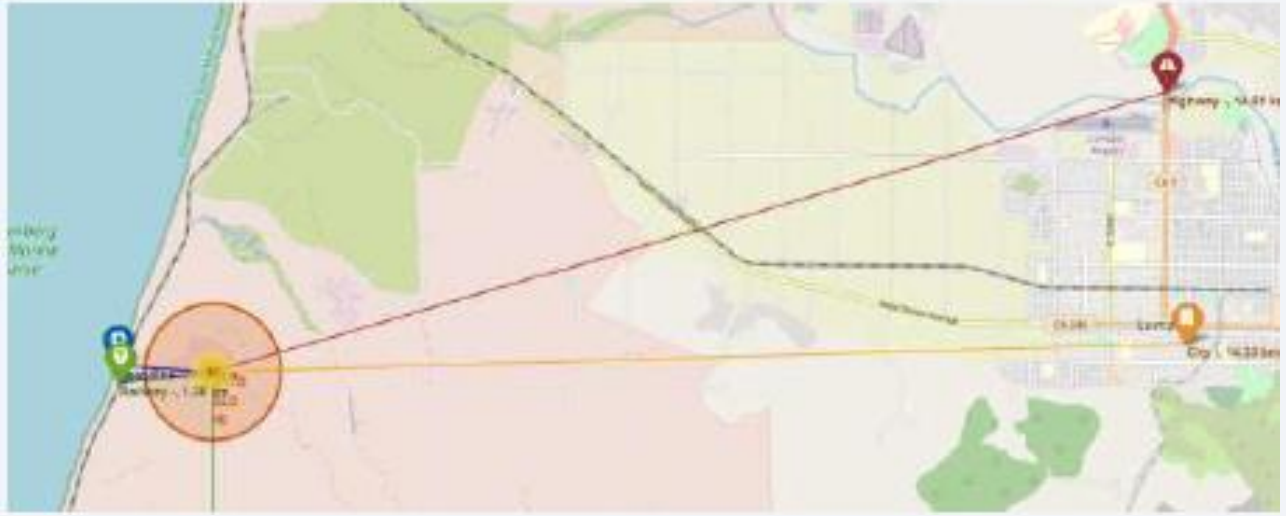
Color-labeled Launch Outcomes



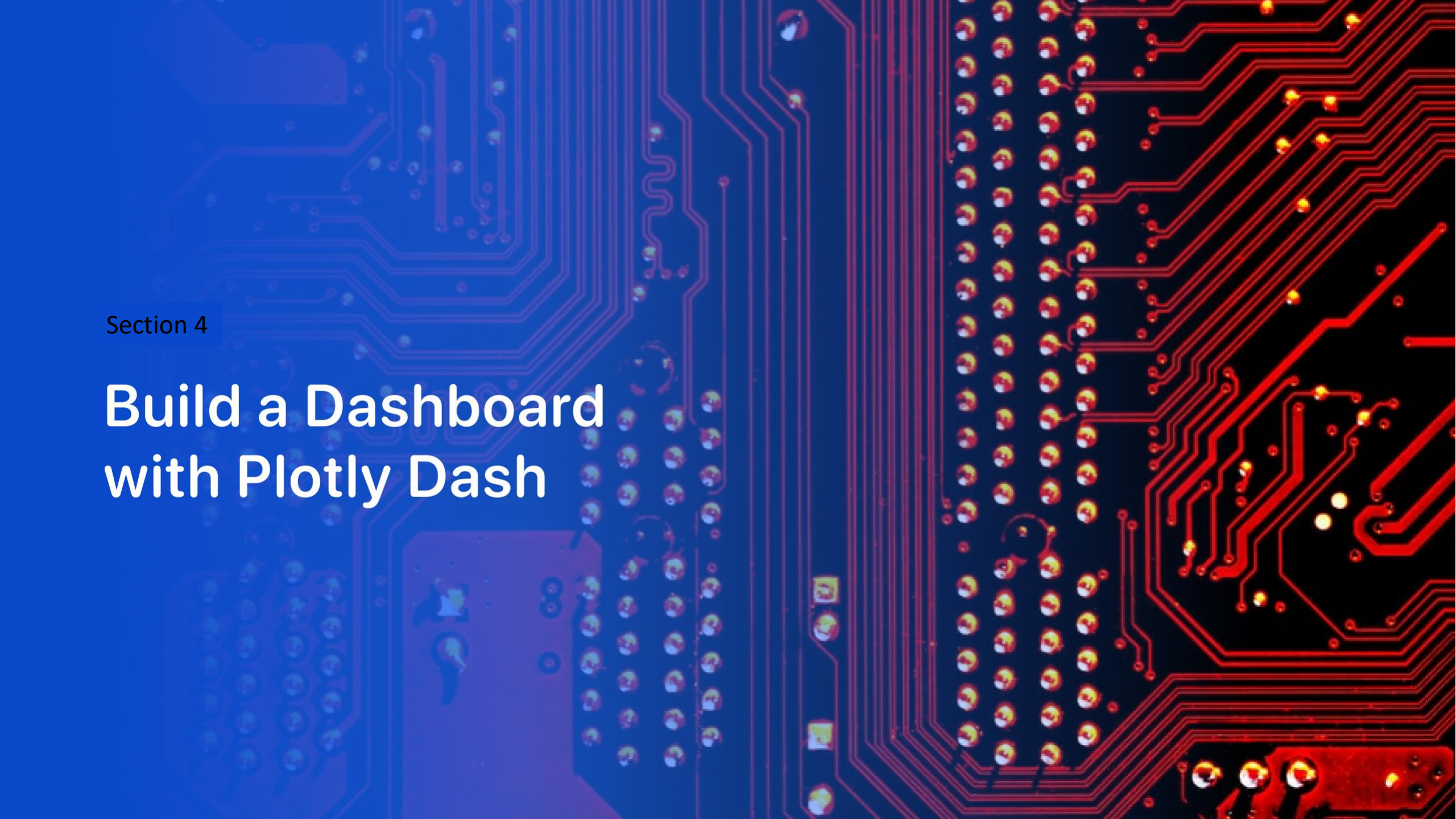


Green for Success
Red for Failure

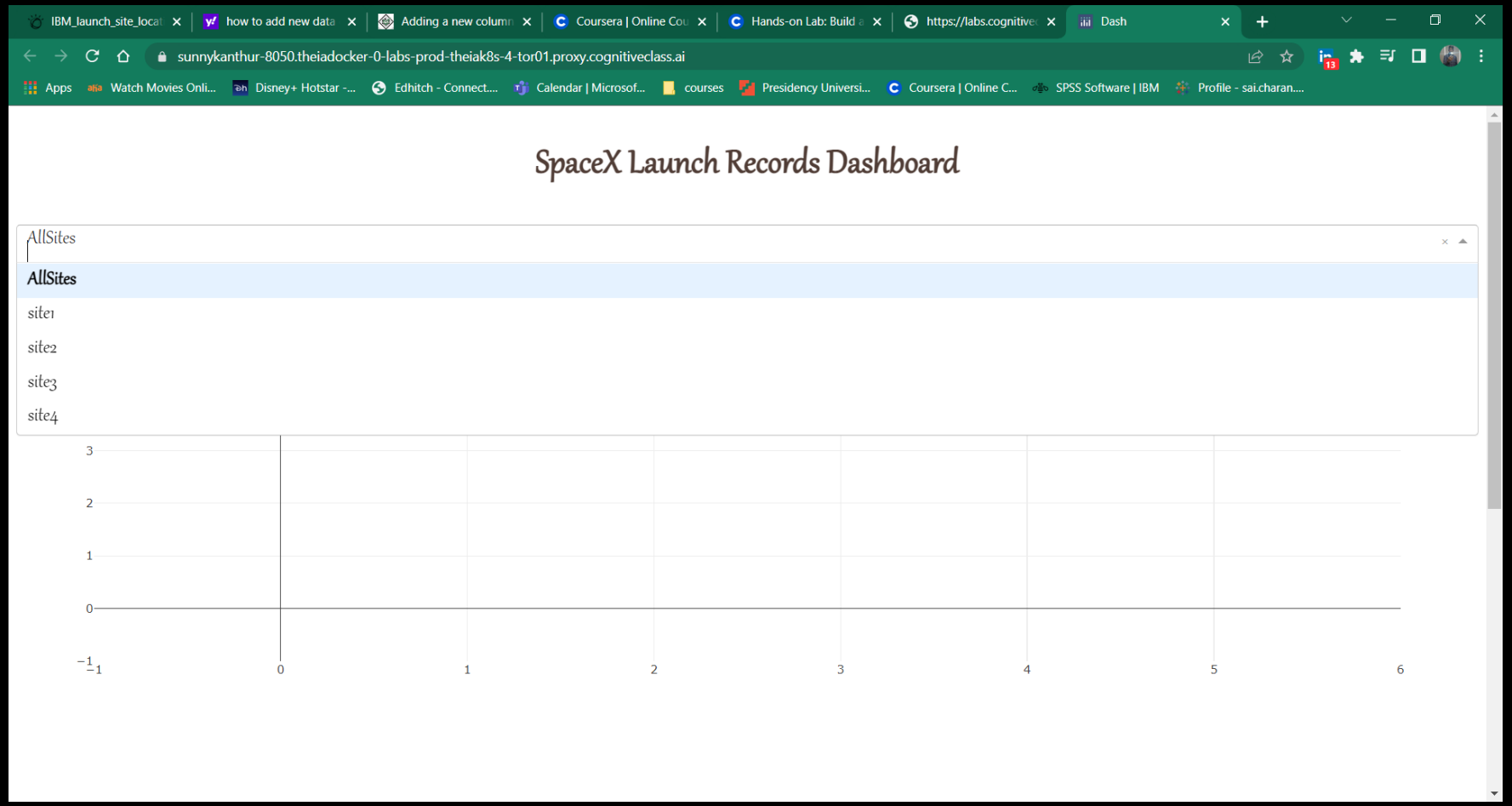
Launch Sites to its Proximities



All distances from launch sites to its proximities, they weren’t far from railway tracks.

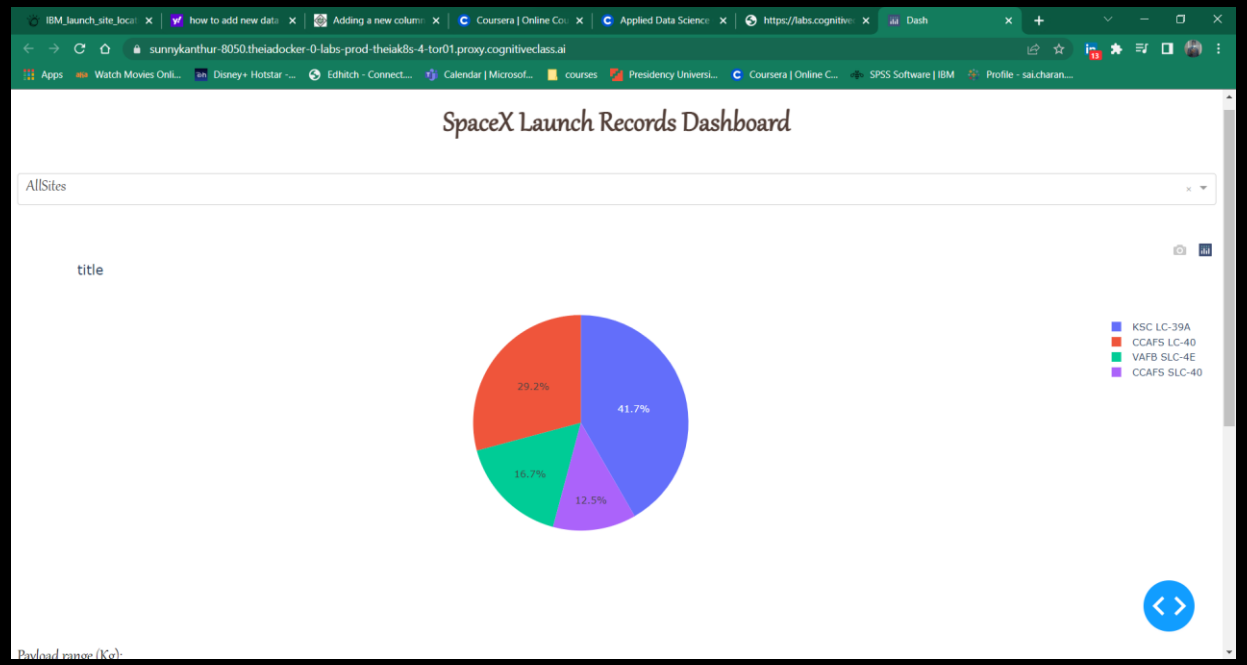

Section 4
Build a Dashboard with Plotly Dash

Dropdown menu for Launchsites

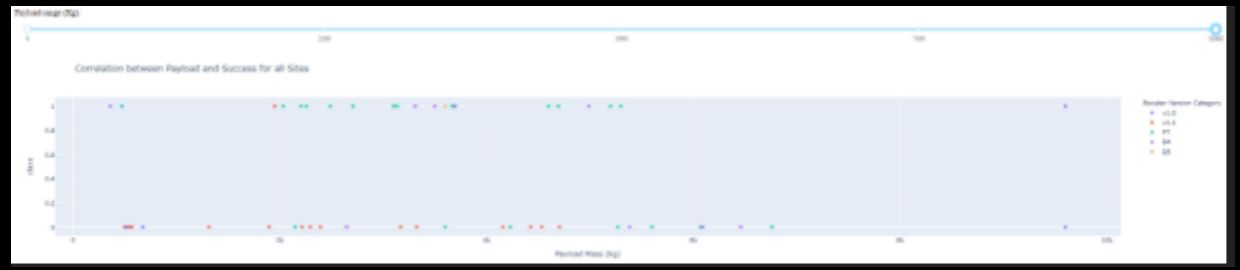


Shows the details of each site when selected.

Pie And Scatter Chart In Dashboard

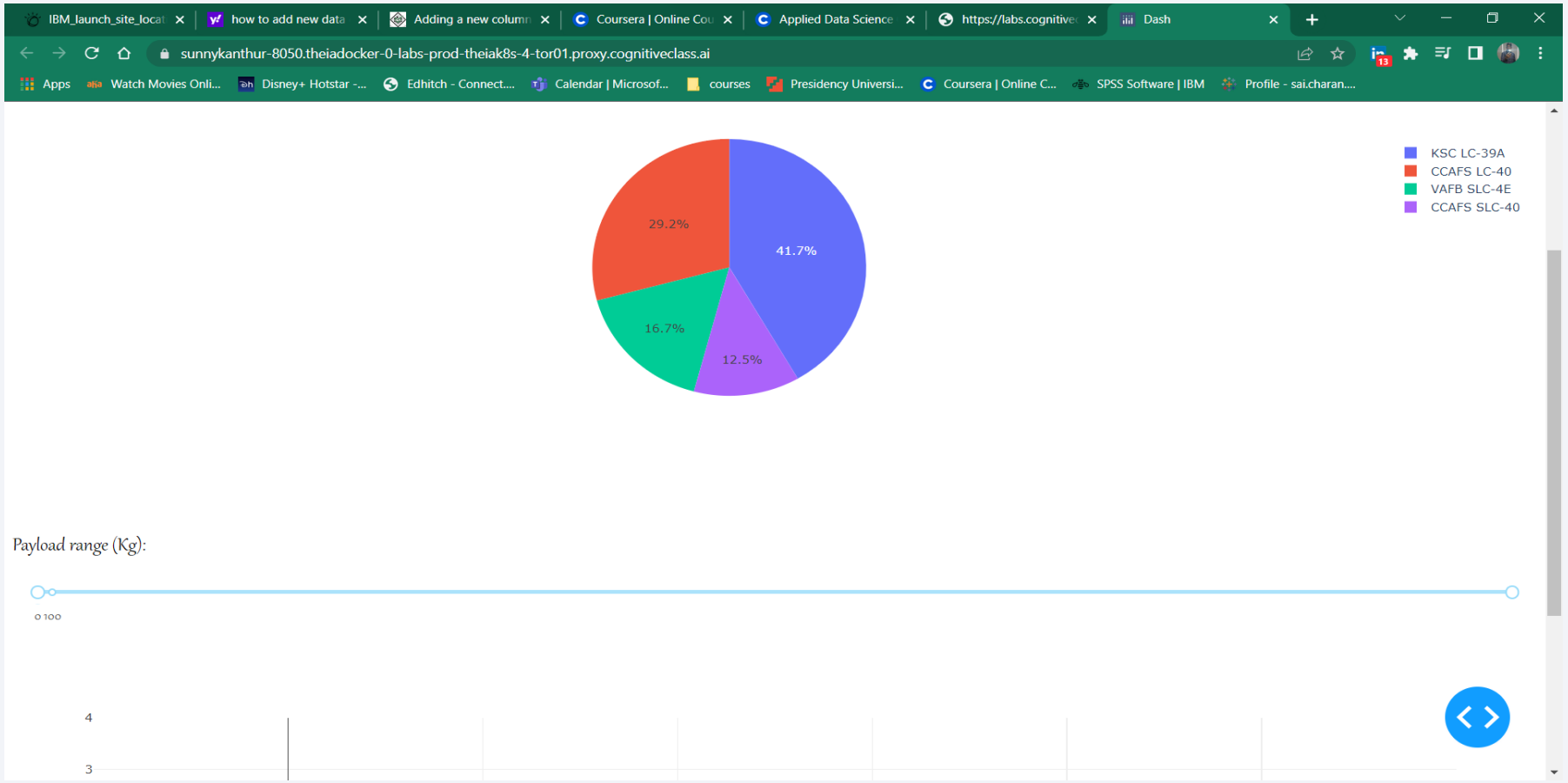


Shows success rate of each Location after launch and return



Scatter plot for all flight numbers with respective payloadmass(kg)

Range Slider

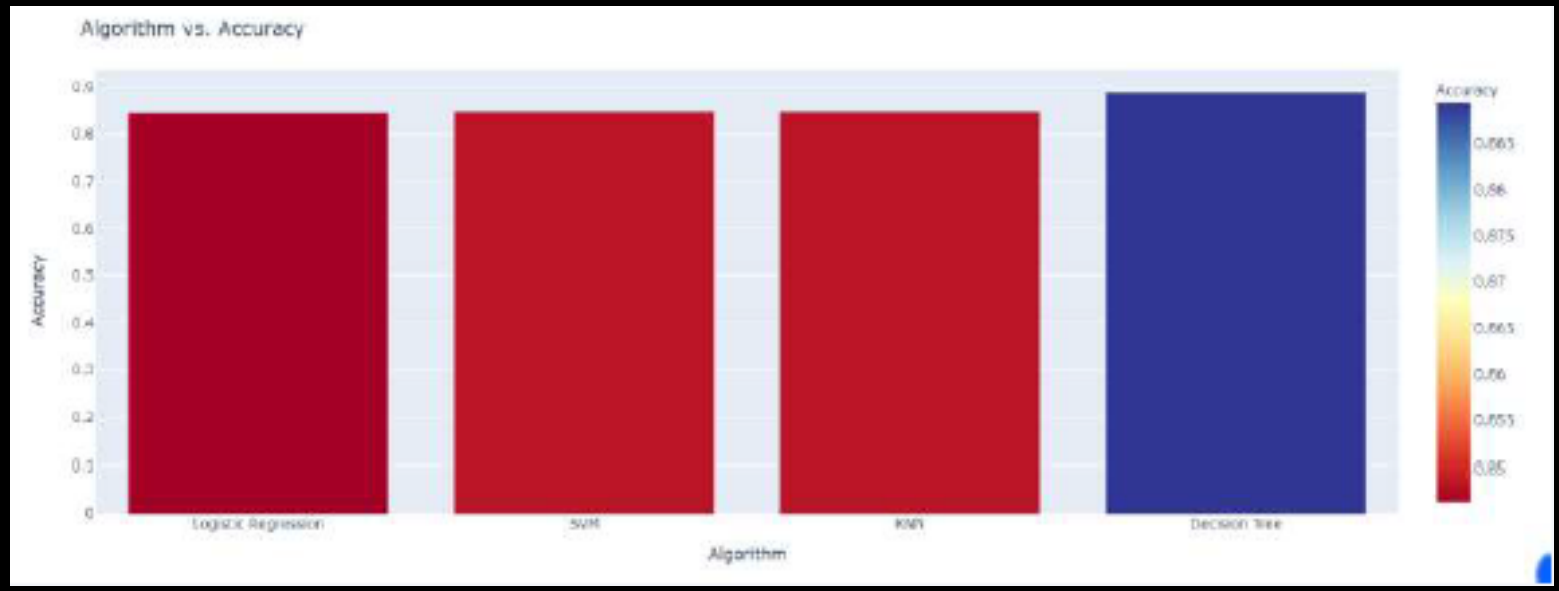


Gives a adjustable line that predicts success rate based on range of Payload

Section 5

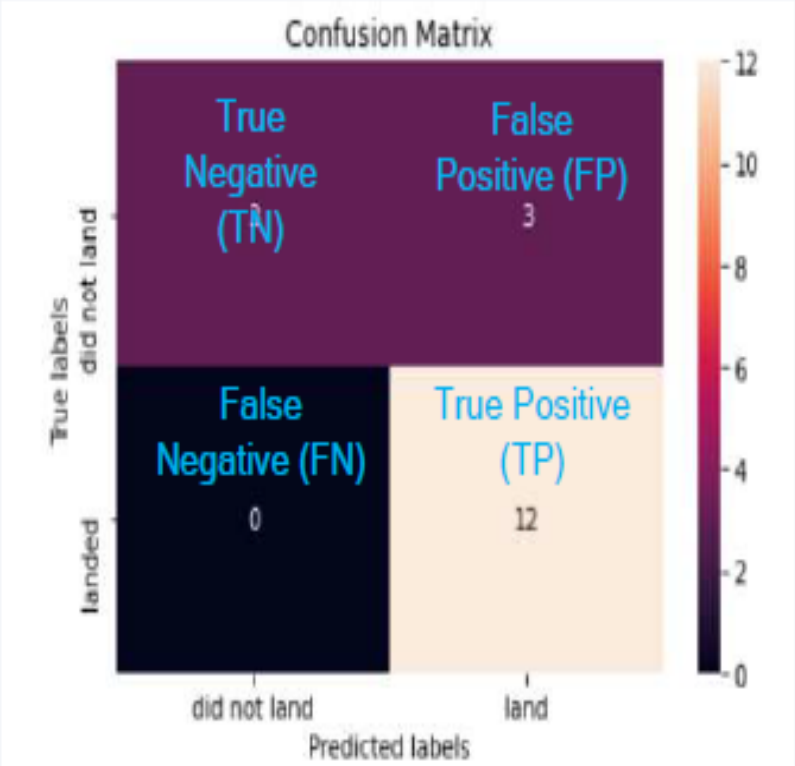
Predictive Analysis (Classification)

Classification Accuracy



Decision tree gives the best output and predicts more accurately.

Confusion Matrix



Sensitivity = 1.00, formula: TPR = TP / (TP + FN)

Specificity = 0.50, formula: SPC = TN / (FP + TN)

Precision = 0.80, formula: PPV = TP / (TP + FP)

Accuracy = 0.83, formula: ACC = (TP + TN) / (P + N)

F1 Score = 0.89, formula: F1 = 2TP / (2TP + FP + FN)

False Positive Rate = 0.50, formula: FPR = FP / (FP + TN)

False Discovery Rate = 0.20, formula: FDR = FP / (FP + TP)

Conclusions

- We found the site with highest score which was KSC LC-39A
- The payload of 0 kg to 5000 kg was more diverse than 6000 kg to 10000 kg
- Decision Tree was the optimal model with accuracy of almost 0.89
- We calculated the launch sites distance to its proximities

Appendix

- All codes can be found in my Github Repository.

https://github.com/SAICHARANKV/IBM_CAPSTONE_PROJECT

Thank you!