Project: First stage of SpaceX falcon 9 rocket will land successfully with Data Science

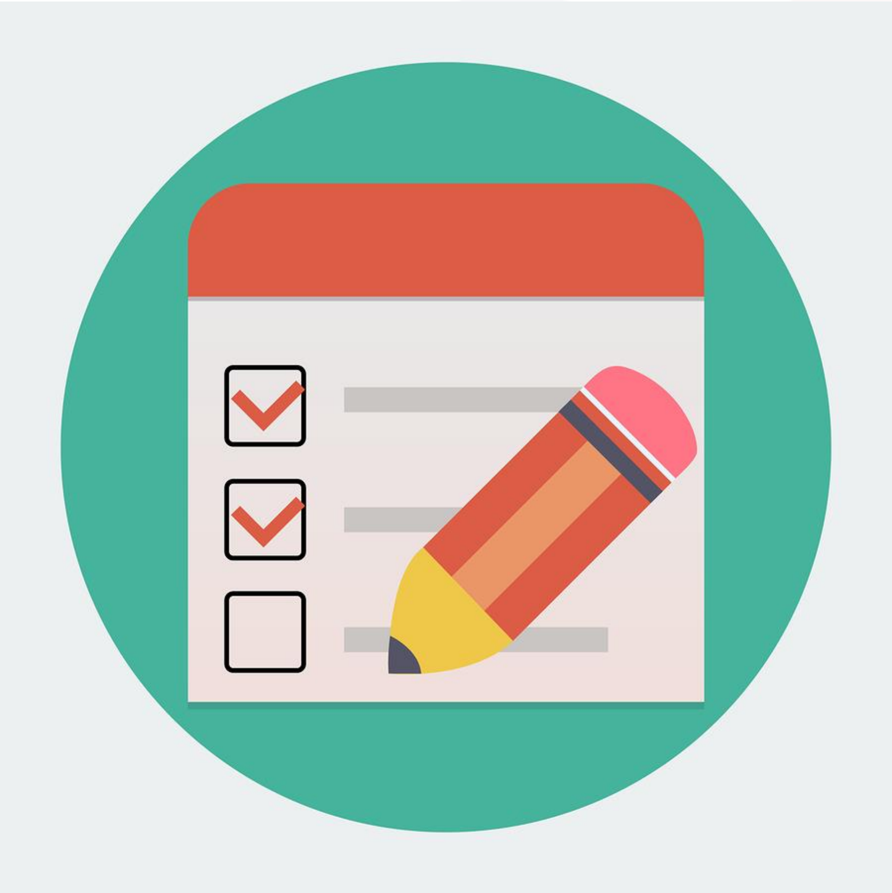
Submit by-Khushi Yadav

Date-27th July, 2022

OUTLINE

- Executive Summary
- Introduction
- Methodology
- Results
  - Visualization Charts
  - Visualization Maps
  - Dashboard
- Discussion
- Conclusion
- Appendix

EXECUTIVE SUMMARY



- Data collection
- Data wrangling
- EDA with visualization
- EDA with SQL
- Build interactive map with Folium
- Plotly Dash dashboards
- Classification
- Result
  - EDA results
  - Interactive analysis

INTRODUCTION

Project:

Space X advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because Space X can reuse the first stage. Therefore if we can determine if the first stage will land, we can determine the cost of a launch.

Task:

Task is to predict the first stage of SpaceX falcon 9 rocket will land successfully

METHODOLOGY SUMMARY

- Data collection methodology
  - SpaceX REST API
  - Web scrapping from Wikipedia
- Data wrangling
  - Data cleaning of null values and irrelevant column
- Perform exploratory data analysis using SQL
- Perform exploratory data analysis using pandas and matplotlib
- Perform Interactive visual analytics and dashboards
  - Using Folium and Plotly Dash
- Perform predictive analysis using classification
  - Logistic Regression, Support Vector Machines, K-nearest neighbours and Decision Tree model have built and analysis which one is best.

Data Collection-SpaceX API

Task 1: Request and parse the SpaceX launch data using the GET request

In [9]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

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

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

Out[10]:
```
200
```

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

In [11]:
```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

Using the dataframe `data` print the first 5 rows

In [12]:
```
# Get the head of the dataframe
data.head()
```


Task 2: Filter the data-frame to only include Falcon 9 launches

In [24]:
```
# 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 [25]:
```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

Data Collection-SpaceX API

- **Task 3: Dealing with Missing Values**

In [27]:

```
# Calculate the mean value of PayloadMass column
payloadmassavg = data_falcon9['PayloadMass'].mean()

# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].replace(np.nan, payloadmassavg, inplace=True)
```

```
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/pandas/core/generic.py:6619: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  return self._update_inplace(result)
```

You should see the number of missing values of the `PayLoadMass` change to zero.

In [28]:

```
data_falcon9.isnull().sum()
```

Out[28]:

```
FlightNumber       0
Date               0
BoosterVersion     0
PayloadMass        0
Orbit              0
LaunchSite         0
Outcome            0
Flights            0
GridFins           0
Reused             0
Legs               0
LandingPad        26
Block              0
ReusedCount        0
Serial             0
Longitude          0
Latitude           0
dtype: int64
```

Data Collection-Web Scraping

Web scrap Falcon 9 launch records with BeautifulSoup:

Task1: Extract a Falcon 9 launch records HTML table from Wikipedia and found title

In [6]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
data  = requests.get(static_url).text
```

Create a `BeautifulSoup` object from the HTML `response`

In [10]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
# soup = BeautifulSoup(data, 'html5lib')
soup = BeautifulSoup(data,  "html.parser")
```

Print the page title to verify if the `BeautifulSoup` object was created properly

In [11]:
```
# Use soup.title attribute
print(soup.title)
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

Task 2: Extract all column/variable names from the HTML table header

In [12]:
```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

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

In [13]:
```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

Data Collection-Web Scraping

Task3: create an empty dictionary with keys from the extracted column names

In [16]:

```
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']=[]
```

Task 4: Create a data frame by parsing the launch HTML tables

In [19]:

```
df=pd.DataFrame(launch_dict)
df.head()
```

Out[19]:

Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time

Data Wrangling

Perform exploratory Data Analysis and determine Training Labels:

Task1:Calculate the number of launches on each site

In [5]:
```
# Apply value_counts() on column LaunchSite
df.LaunchSite.value_counts()
```

Out[5]:
```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

Task 2: Calculate the number and occurrence of each orbit

In [6]:
```
# Apply value_counts on Orbit column
df.Orbit.value_counts()
```

Out[6]:
```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
ES-L1     1
HEO       1
SO        1
GEO       1
Name: Orbit, dtype: int64
```

Data Wrangling

Task3: Calculate the number and occurence of mission outcome per orbit type

In [7]:
```
# landing_outcomes = values on Outcome column
landing_outcomes = df.Outcome.value_counts()
landing_outcomes
```

Out[7]:
```
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
Name: Outcome, dtype: int64
```

Task 4: Create a landing outcome label from Outcome column and success rate

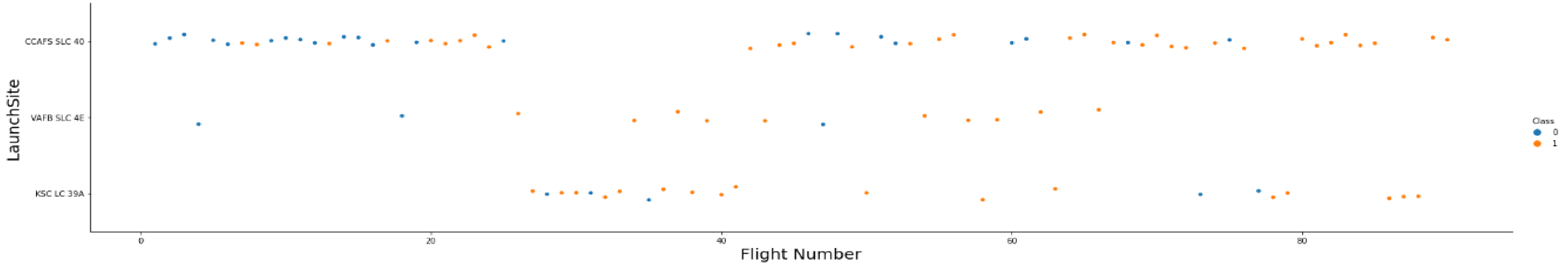
In [11]:
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for outcome in df['Outcome']:
    if outcome in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

In [14]:
```
df["Class"].mean()
```

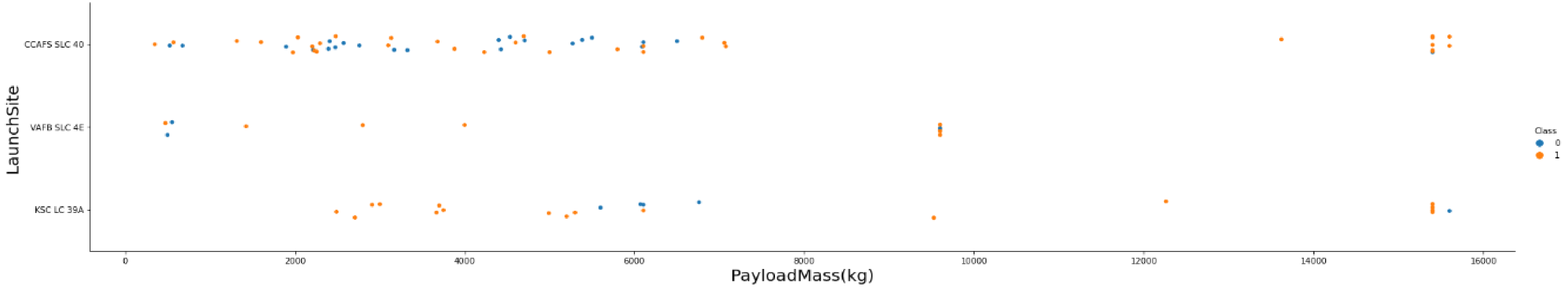
Out[14]:
```
0.6666666666666666
```

EDA with Data visualization

Perform exploratory Data Analysis and Feature Engineering using Pandas and Matplotlib:

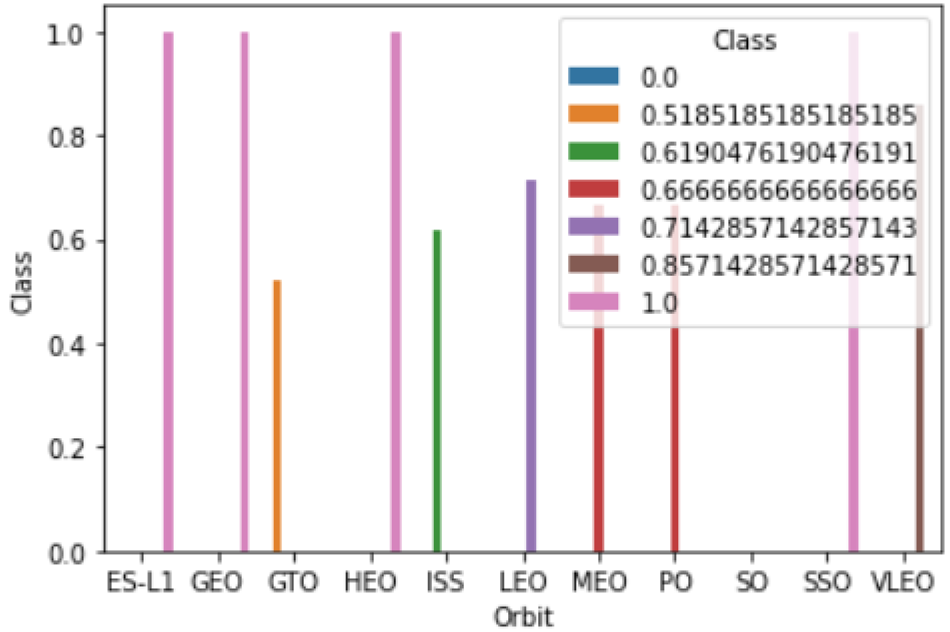


Visualize the relationship between Flight Number and Launch Site

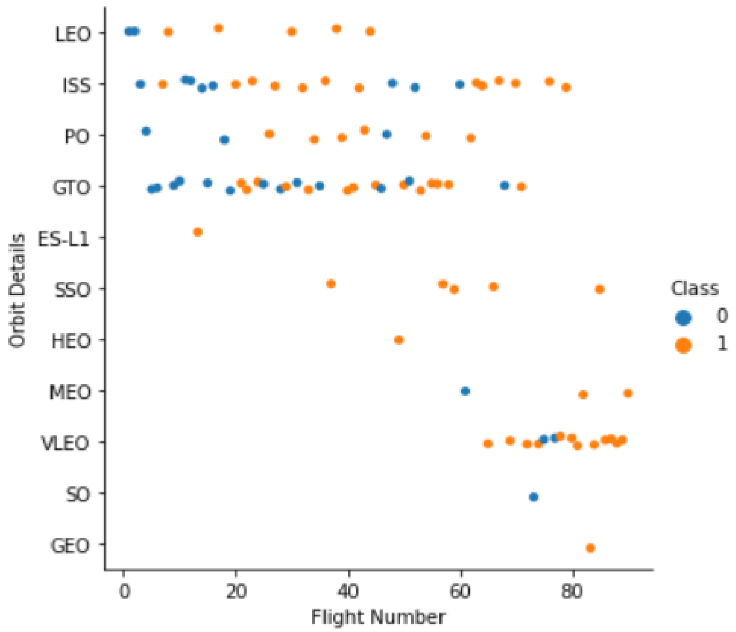


Visualize the relationship between Payload and Launch Site

EDA with Data visualization

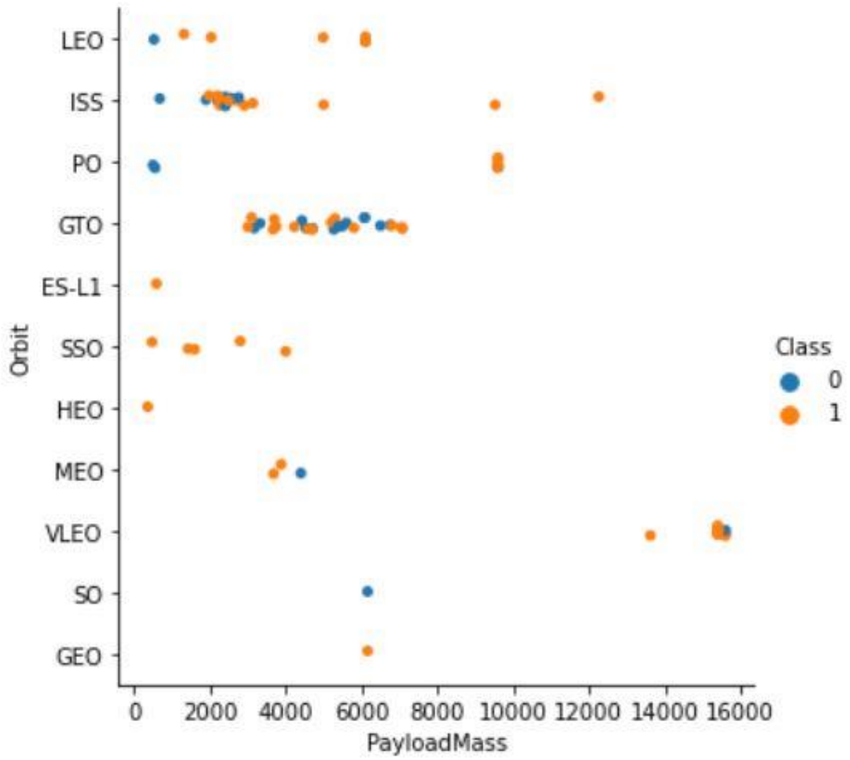


Visualize the relationship between success rate of each orbit type

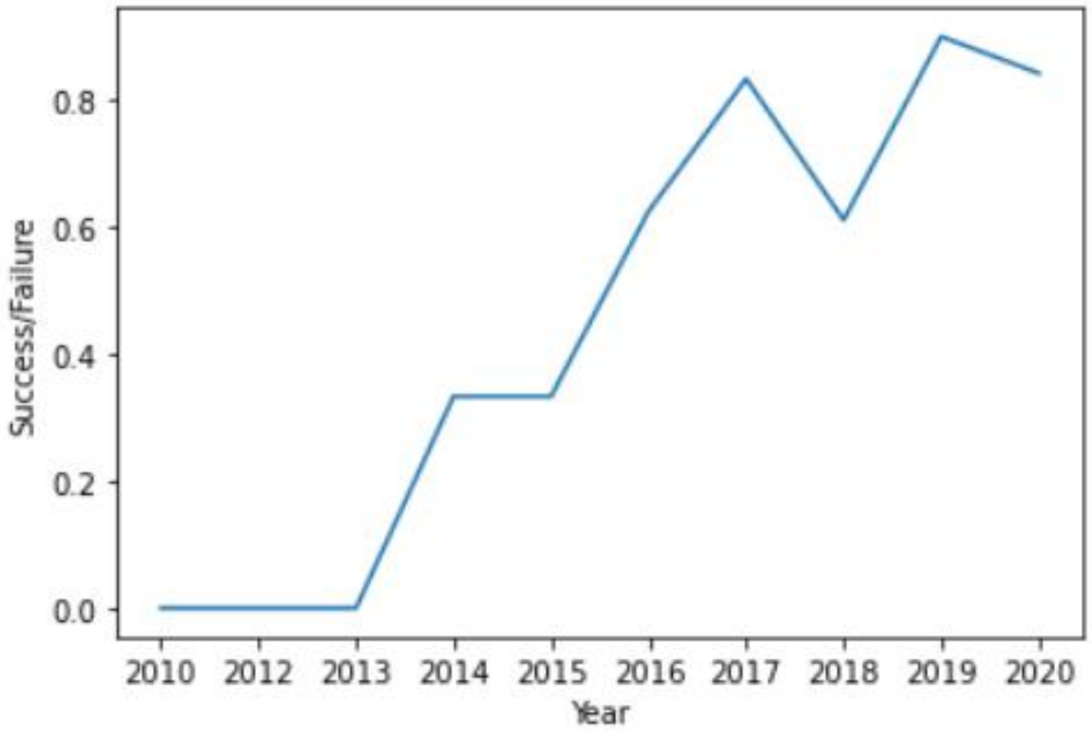


Visualize the relationship between FlightNumber and Orbit type

EDA with Data visualization



Visualize the relationship between payloadMass and orbit

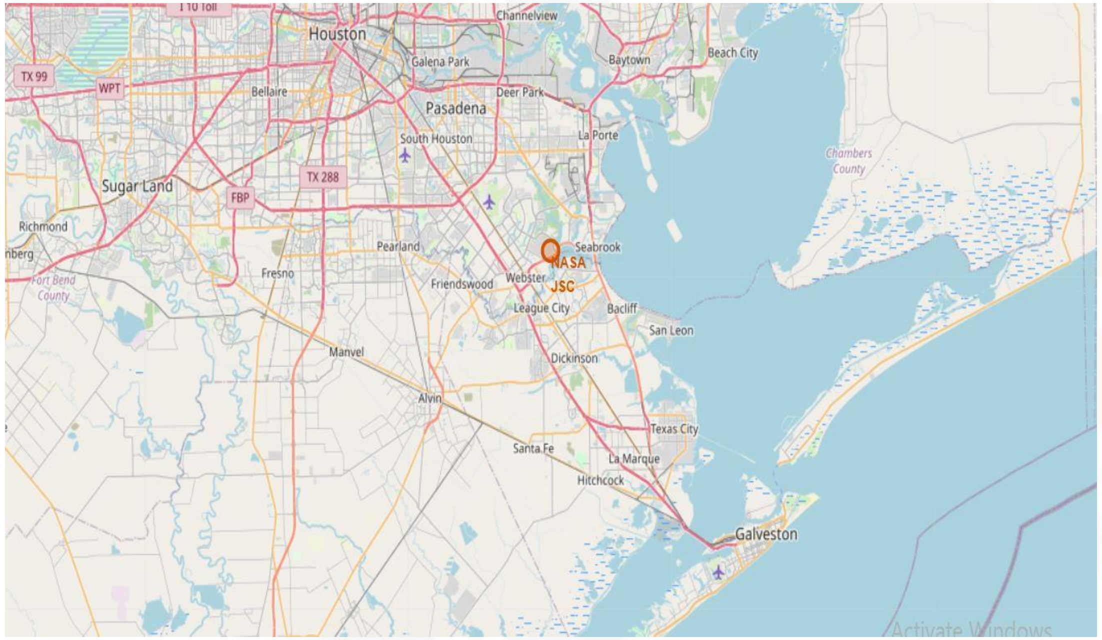


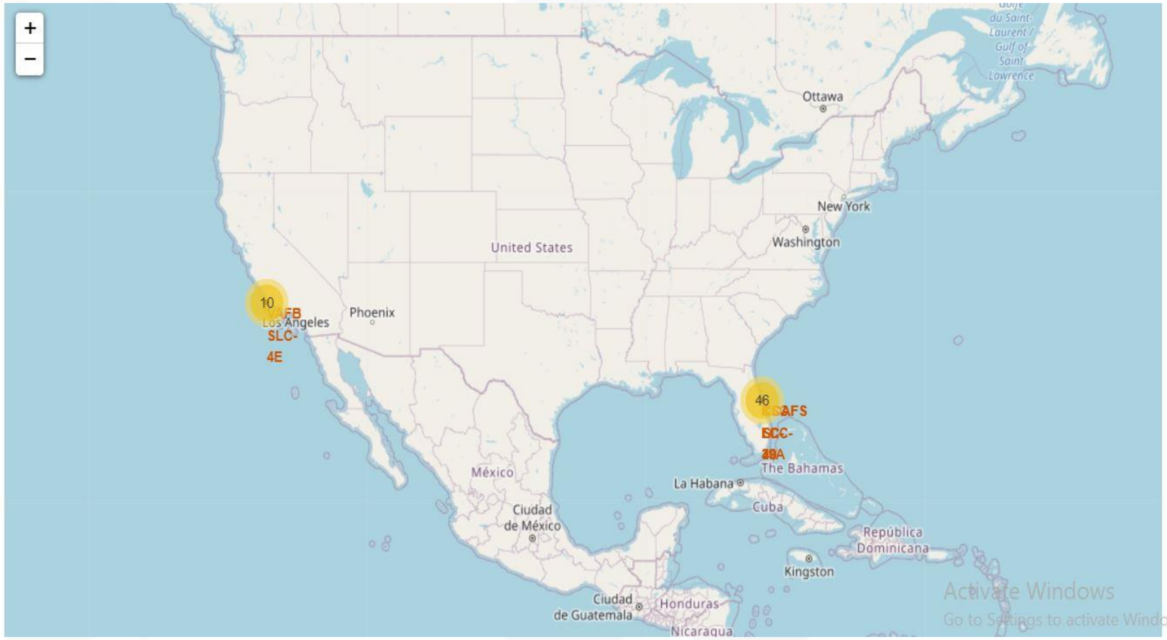
Visualize the launch success yearly rate

EDA with SQL

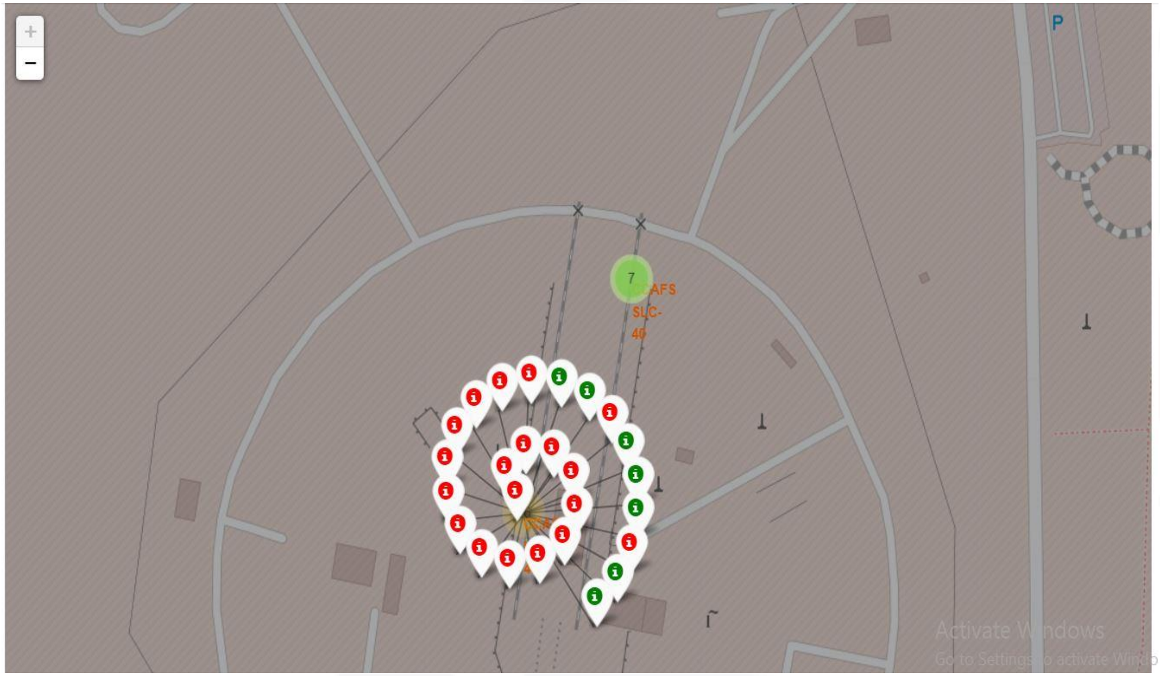
- Task1: Display the names of the unique launch sites in the space mission
- Task2: Display 5 records where launch sites begin with the string 'CCA'
- Task3: Display the total payload mass carried by boosters launched by NASA (CRS)
- Task4: Display average payload mass carried by booster version F9 v1.1
- Task5: List the date when the first succesful landing outcome in ground pad was acheived.
- Task6: List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- Task7: List the total number of successful and failure mission outcomes
- Task8: List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
- Task9: Records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.
- Task10: Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

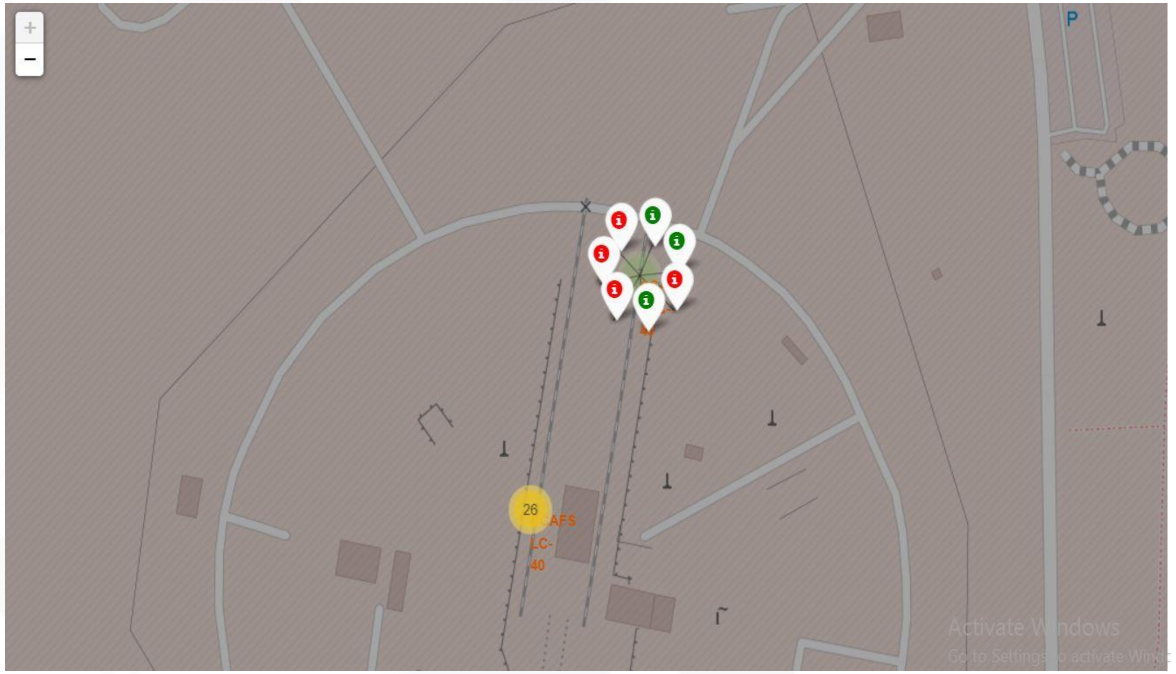
Launch Sites Locations Analysis with Folium





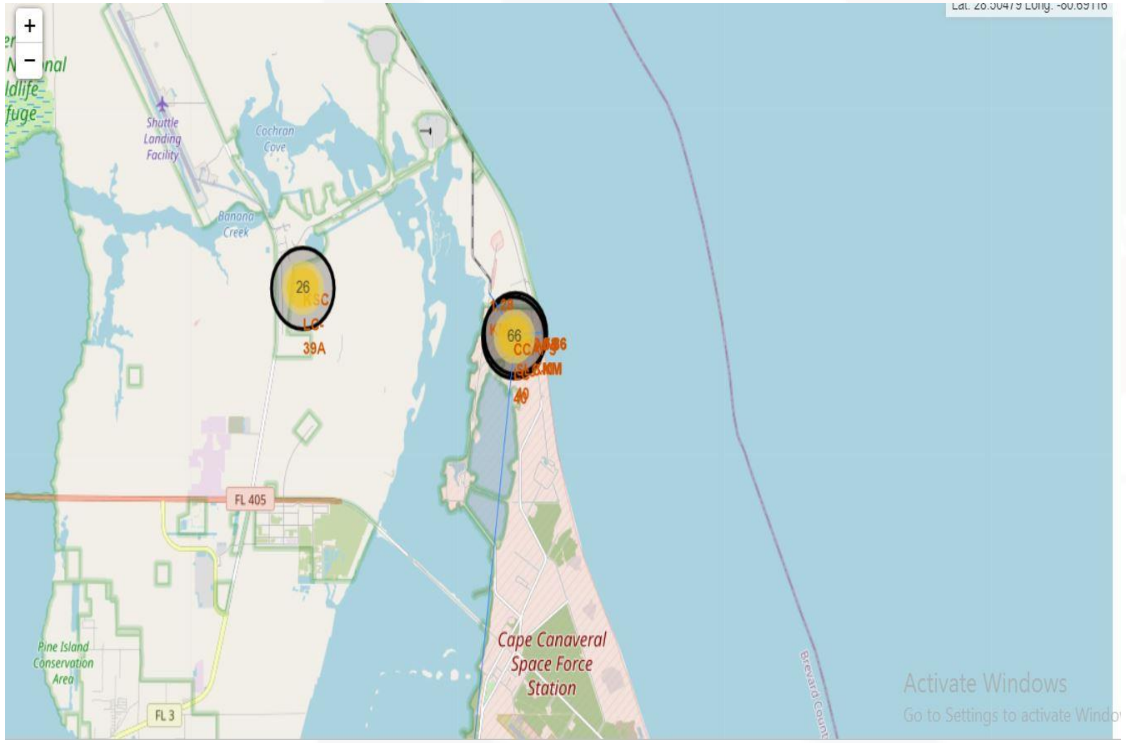
Launch Sites Locations Analysis with Folium

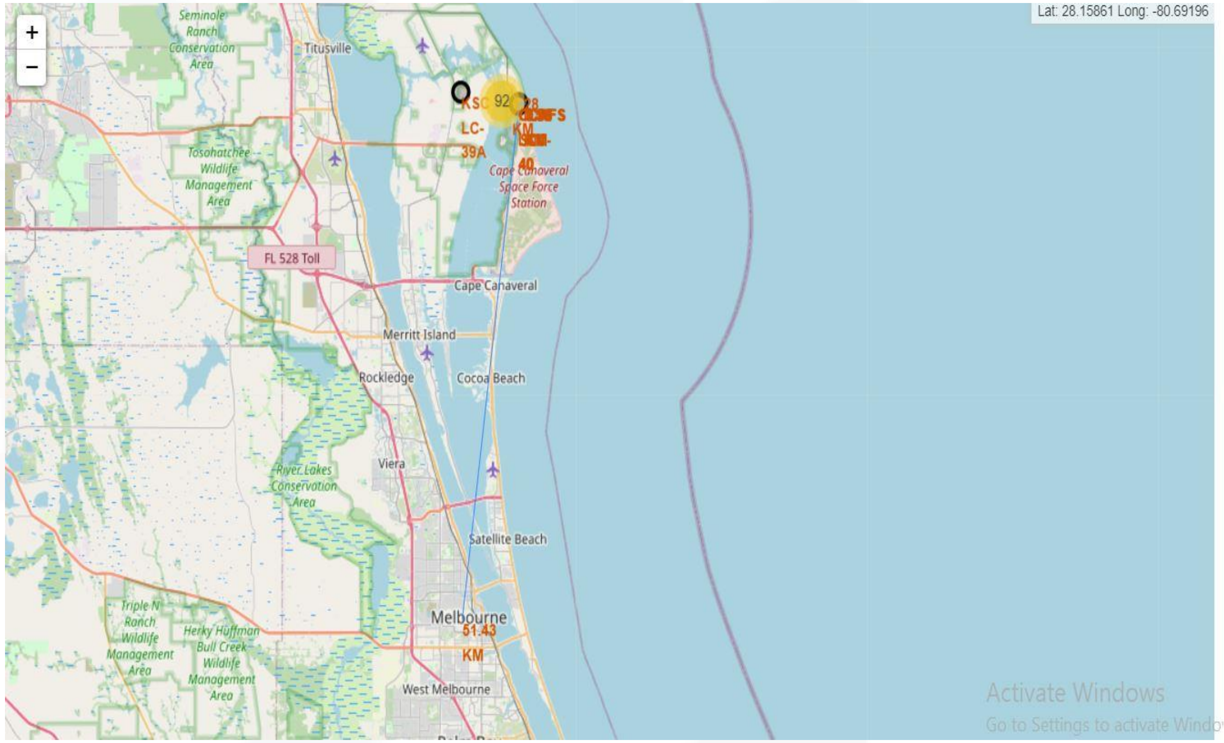




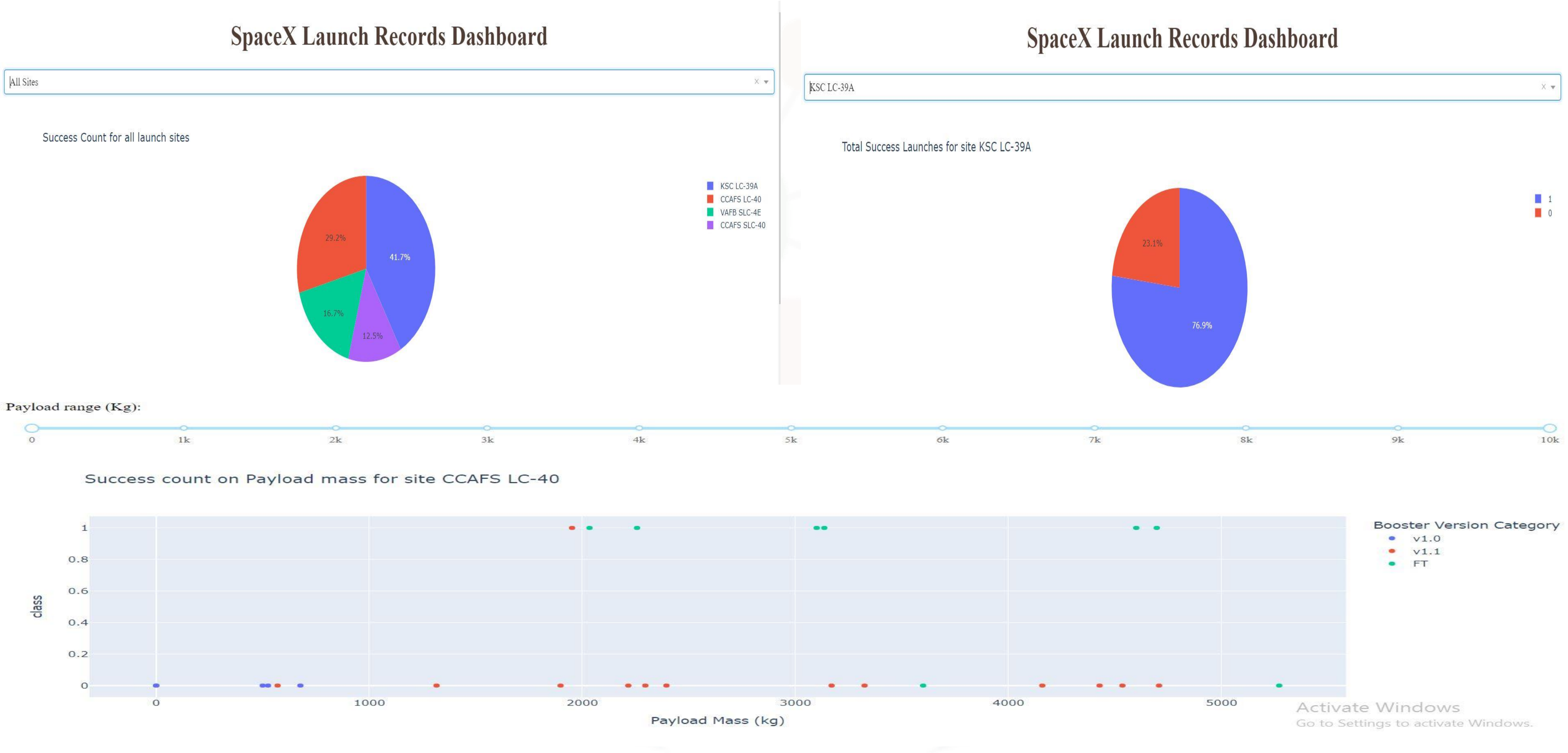
Unsuccessful launches
Successful launches

Calculate the distances between a launch site to its proximities

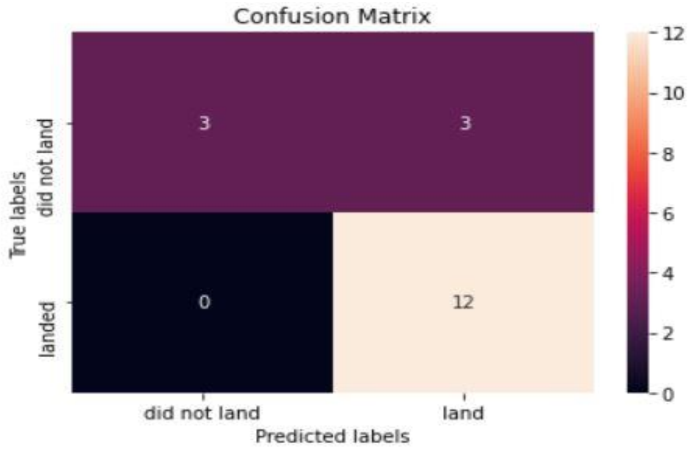




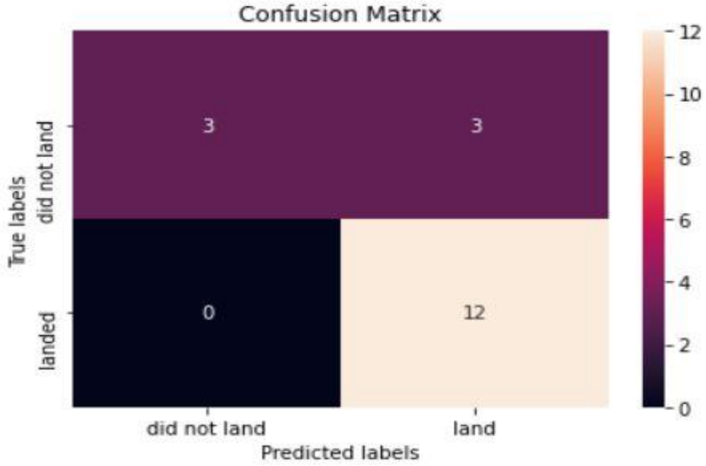
Dashboards


SpaceX Launch Records Dashboard
All Sites
Success Count for all launch sites
KSC LC-39A
CCAFS LC-40
VAFB SLC-4E
CCAFS SLC-40
41.7%
29.2%
16.7%
12.5%
SpaceX Launch Records Dashboard
KSC LC-39A
Total Success Launches for site KSC LC-39A
1
0
23.1%
76.9%
Payload range (Kg):
0
1k
2k
3k
4k
5k
6k
7k
8k
9k
10k
Success count on Payload mass for site CCAFS LC-40
class
1
0.8
0.6
0.4
0.2
0
0
1000
2000
3000
4000
5000
Payload Mass (kg)
Booster Version Category
v1.0
v1.1
FT
Activate Windows
Go to Settings to activate Windows.

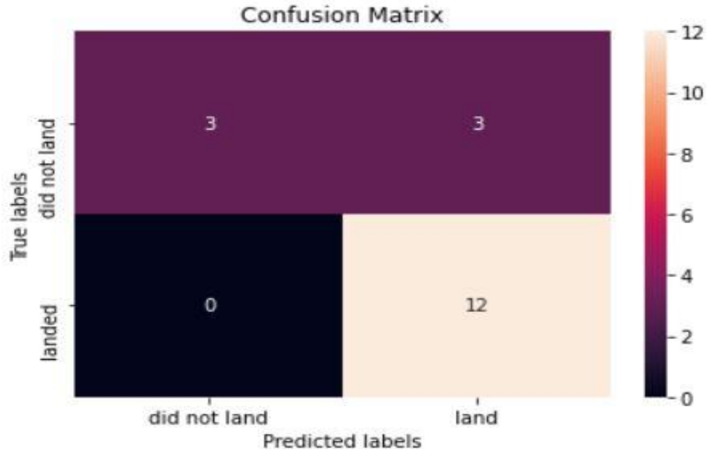
Predictive analysis(Classification)



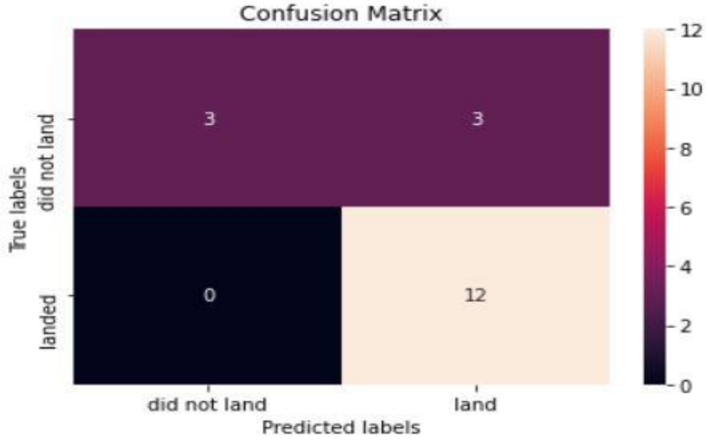
KNN Confusion Matrix



LR Confusion Matrix



SVM Confusion Matrix

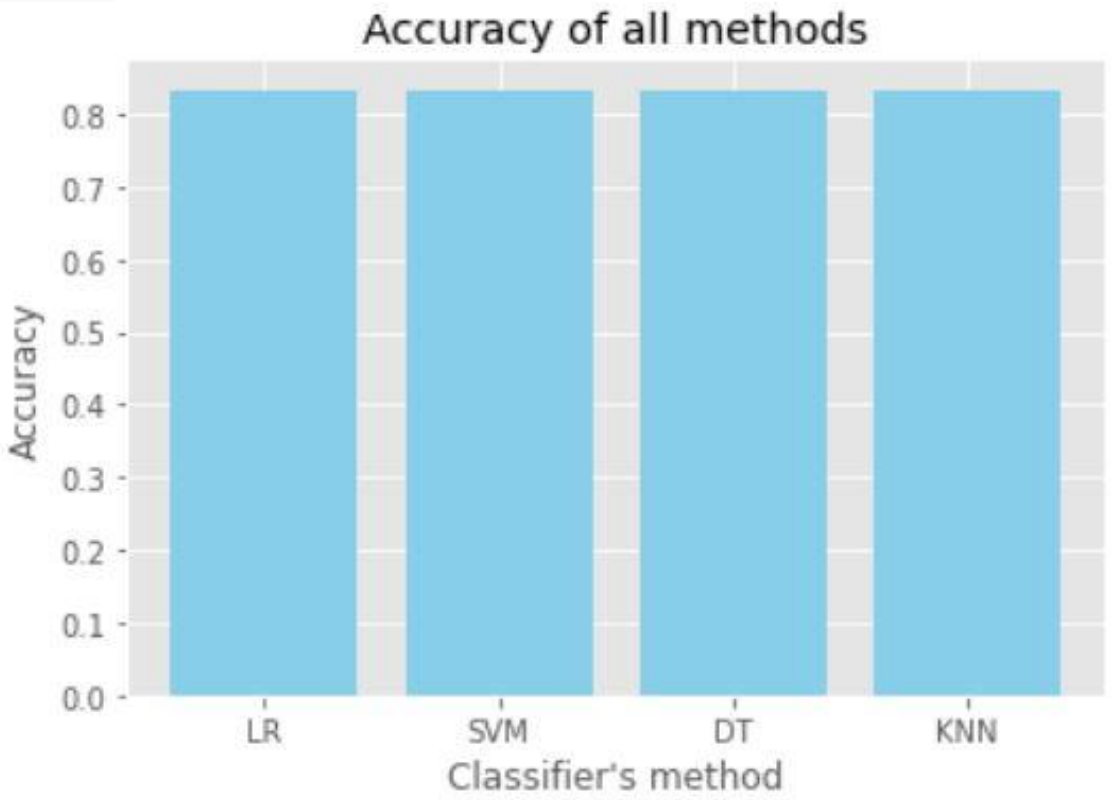


DT Confusion Matrix

Predictive analysis(Classification)

- Logistic Regression, Support Vector Machines, K-nearest neighbours and Decision Tree have the same accuracy 83.3%

- All Models give us same accuracy.

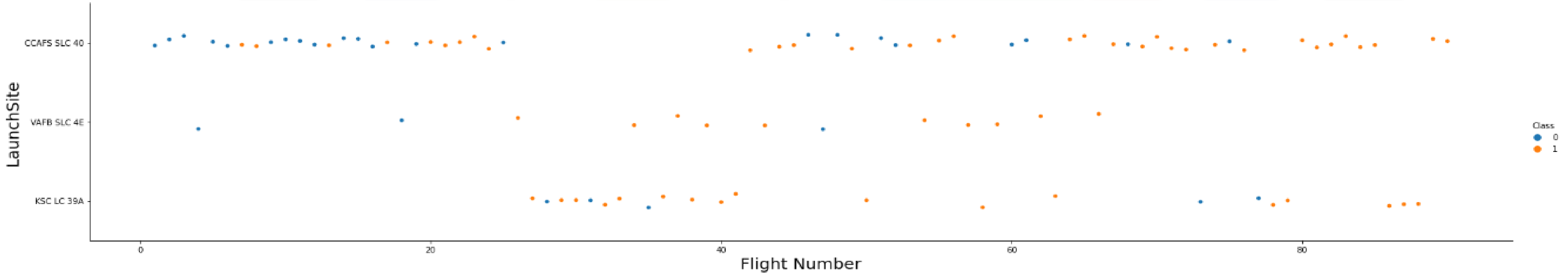


RESULTS

- Logistic Regression, Support Vector Machines, K-nearest neighbours and Decision Tree have the same accuracy 83.3%
- KSC LC-39A have high success count around 41.7% among all.
- The orbit type ES-L1,GEO,HEO,SSO have highest success rate
- CCAFS SLC 40 have higher Launch Site
- CCAFS SLC 40 have higher Launch Site when Payload Mass(kg)<7000
- Flight Number between 60-80, orbit type is VLEO
- Strong relationship between ISS orbit v/s Payload Mass range(2000-3000) and between GTO orbit v/s Payload Mass range(4000-8000)
- Launch success rate has increased from 2013 to 2020
- Low weighted payload performs better then higher payload
- Minimum distance is with highway=0.583 km and maximum distance is with rail road= 1.284 km from launch site.

EDA with visualization

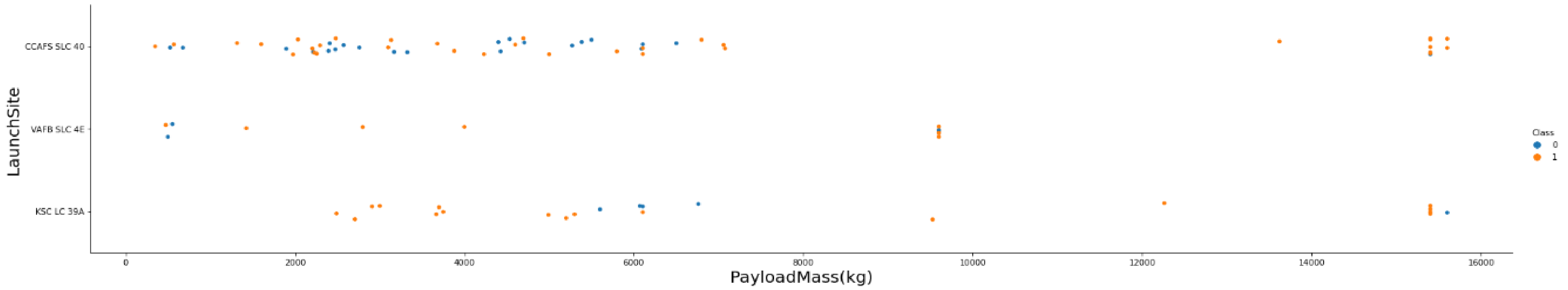
✓ Visualize the relationship between Flight Number and Launch Site



Result: CCAFS SLC 40 have higher Launch Site

EDA with visualization

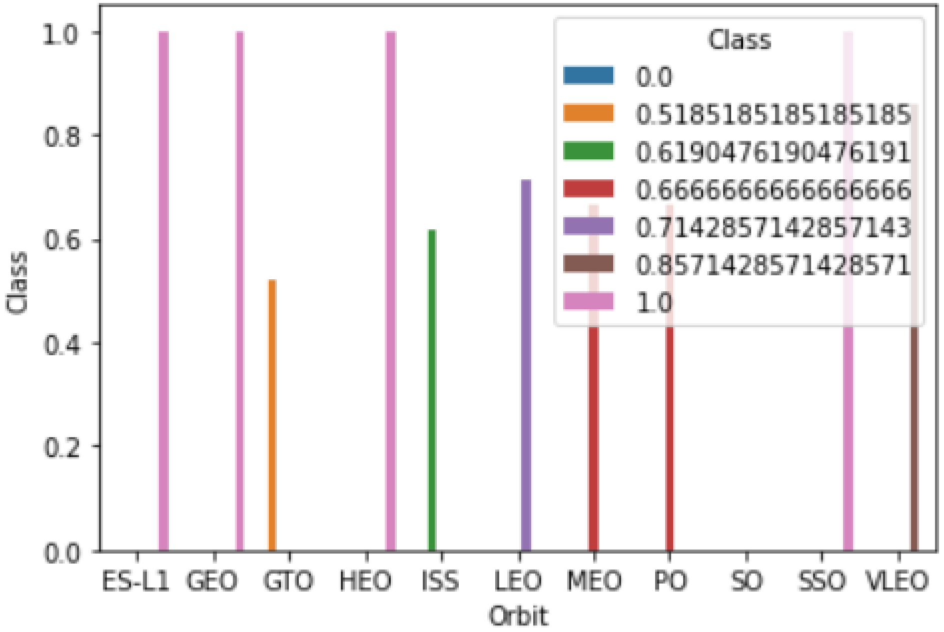
✓ Visualize the relationship between Payload and Launch Site



Result: CCAFS SLC 40 have higher Launch Site when Payload Mass(kg)<7000

EDA with visualization

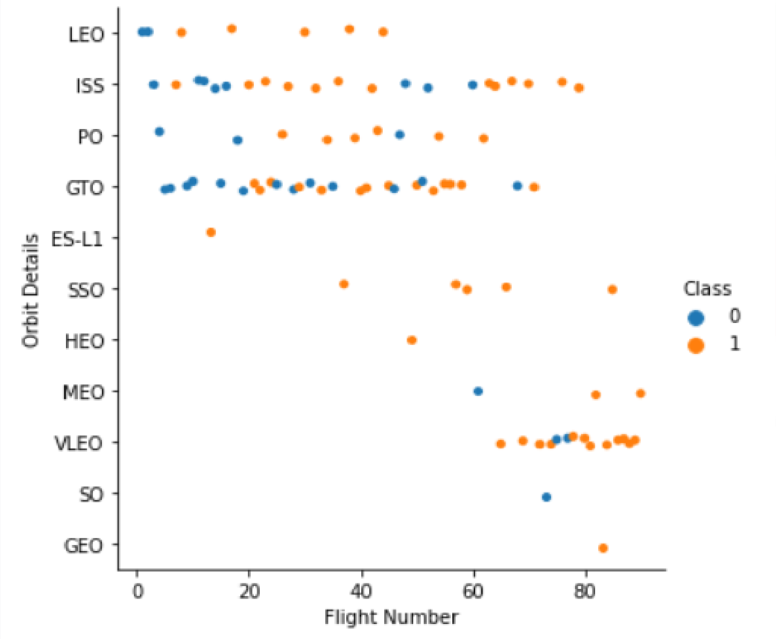
- ✓ Visualize the relationship between <u>success rate of each orbit type</u>



Result: The orbit type ES-L1,GEO,HEO,SSO have highest success rate

EDA with visualization

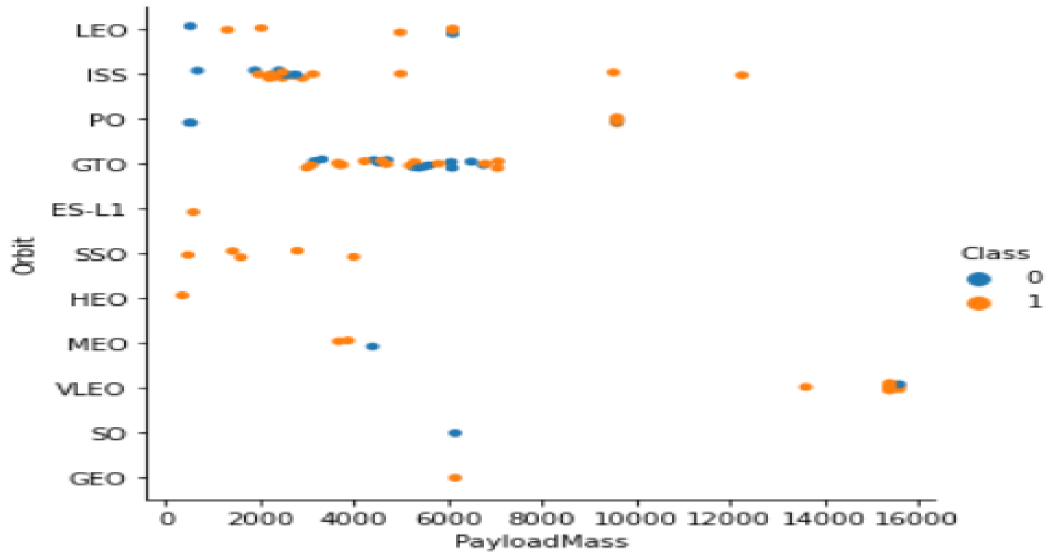
✓ Visualize the relationship between Flight Number and Orbit type



Result: Flight Number between 60-80, orbit type is VLEO

EDA with visualization

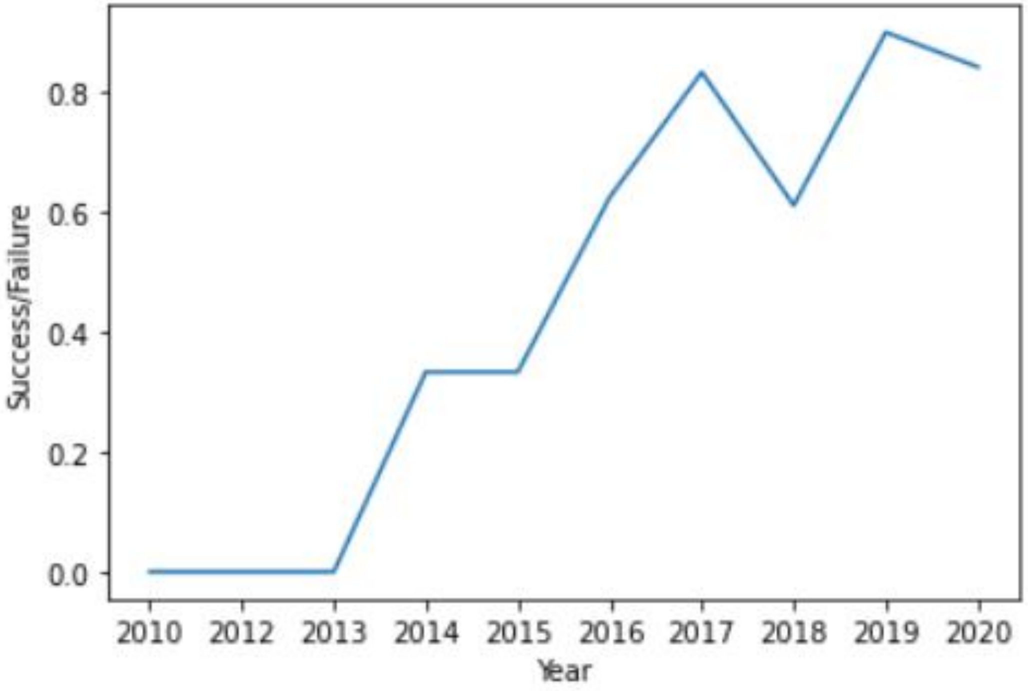
✓ Visualize the relationship between Payload Mass and Orbit type



Result: Strong relationship between ISS orbit v/s Payload Mass range(2000-3000) and between GTO orbit v/s Payload Mass range(4000-8000)

EDA with visualization

✓ Visualize the launch success yearly trend



Result: Launch success rate has increased from 2013 to 2020

EDA with SQL

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

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

```
[10]: %sql select distinct(LAUNCH_SITE) from SPACEXTBL;
```

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

[10]:

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

EDA with SQL

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

Task 2

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

[19]:
```
%sql SELECT * from SPACEXTBL where (LAUNCH_SITE) LIKE 'CCA%' LIMIT 5;
```

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

[19]:

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

EDA with SQL

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

```
[14]: %sql select sum(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL where Customer='NASA (CRS)';

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

[14]:

payloadmass
45596

EDA with SQL

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

```
[15]: %sql select avg(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL where Booster_Version= 'F9 v1.1';

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

[15]:

payloadmass
2928.4

EDA with SQL

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

```
[62]: %sql select min(DATE) from SPACEXTBL where Landing _Outcome='Success(ground pad)';

 * sqlite:///my_data1.db
(sqlite3.OperationalError) near "_Outcome": syntax error
[SQL: select min(DATE) from SPACEXTBL where Landing _Outcome='Success(ground pad)';]
(Background on this error at: http://sqlalche.me/e/e3q8)
```

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

```
[46]: %sql select Booster_Version from SPACEXTBL where Landing _Outcome ='Success(drone ship)' and PAYLOAD_MASS__KG_>4000 and PAYLOAD_MASS__KG_<60
```

```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) near "_Outcome": syntax error
[SQL: select Booster_Version from SPACEXTBL where Landing _Outcome ='Success(drone ship)' and PAYLOAD_MASS__KG_>4000 and PAYLOAD_MASS__KG_<6
000;]
(Background on this error at: http://sqlalche.me/e/e3q8)
```

EDA with SQL

✓ List the total number of successful and failure mission outcomes.

```
[38]: %sql select count(MISSION_OUTCOME) as missionoutcomes from SPACEXTBL GROUP BY MISSION_OUTCOME='Success'or MISSION_OUTCOME='Failure(in flight
```

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

[38]:

missionoutcomes
3
98

EDA with SQL

✓ List the names of the booster_versions which have carried the maximum payload mass.

In [21]:
```
%sql select BOOSTER_VERSION as boosterversion from SPACEXTBL where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from SPACEXTBL);
```

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

Out[21]:

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

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.

```
[52]: %sql SELECT MONTH(NAME),Landing_Outcome,BOOSTER_VERSION,LAUNCH_SITE FROM SPACEXTBL where Landing__Outcome like 'Success%' and(DATE between ':
```

```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) no such column: NAME
[SQL: SELECT MONTH(NAME),Landing_Outcome,BOOSTER_VERSION,LAUNCH_SITE FROM SPACEXTBL where Landing__Outcome like 'Success%' and(DATE between
'2015-01-01' and '2015-12-31') order by data desc;]
(Background on this error at: http://sqlalche.me/e/e3q8)
```

EDA with SQL

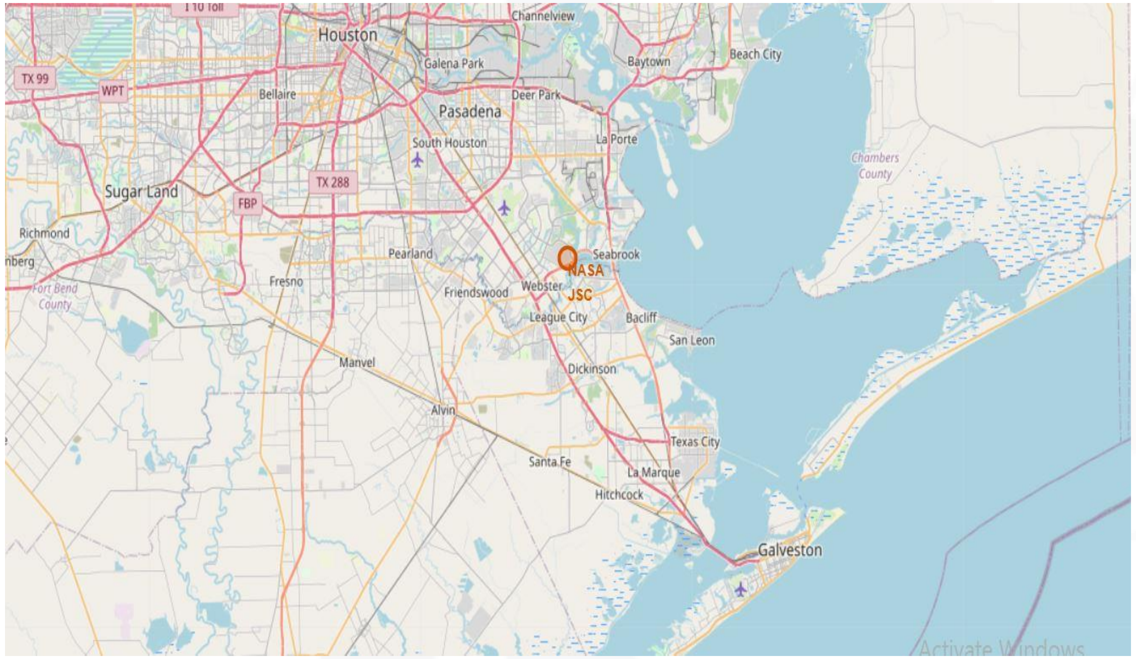
✓ Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

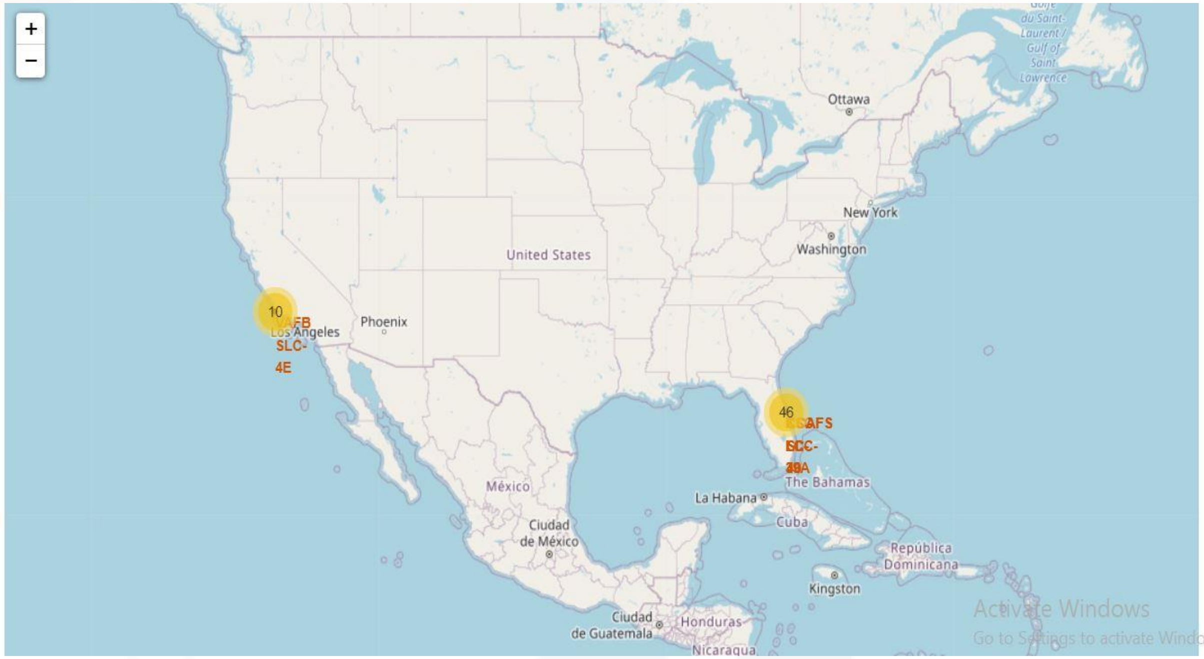
```
[55]: %sql SELECT * FROM SPACEXTBL WHERE Landing_Outcome like'Success%'and (DATE BETWEEN '2010-06-04' AND '2017-03-20') ORDER BY DATE DESC;
```

```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) no such column: Landing_Outcome
[SQL: SELECT * FROM SPACEXTBL WHERE Landing_Outcome like'Success%'and (DATE BETWEEN '2010-06-04' AND '2017-03-20') ORDER BY DATE DESC;]
(Background on this error at: http://sqlalche.me/e/e3q8)
```

Launch Sites on Map

✓ Task 1: Mark all launch sites on a map

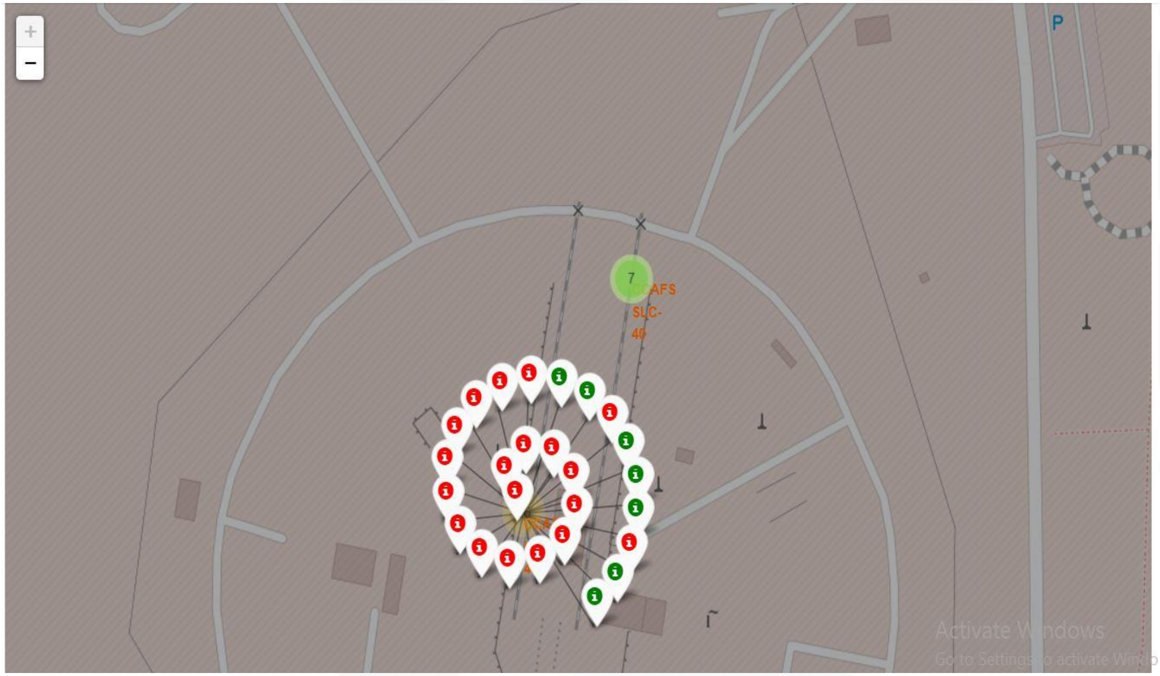


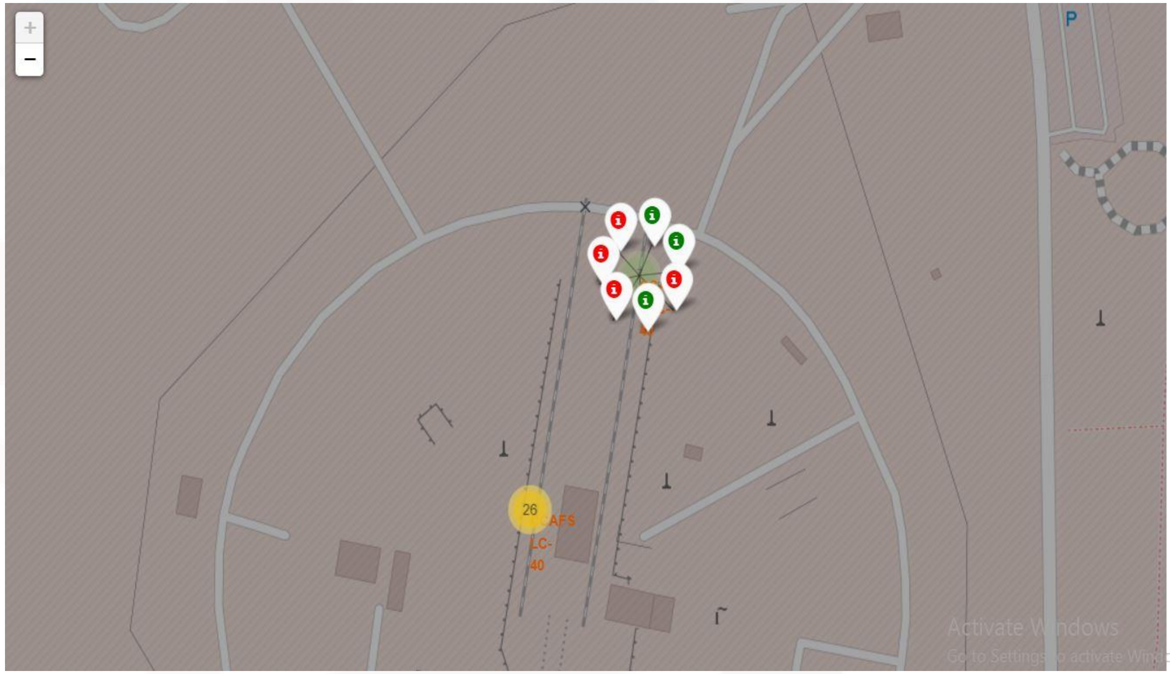


Found: launch sites are in proximity to the equator and the coast. This makes sense as it takes less fuel to get into space from the equator due to the physics of Earth's rotation. The launch sites in close proximity to the coast are also logical for safety reasons.

Launch Sites on Map

✓ Task 2: Mark the success/failed launches for each site on the map



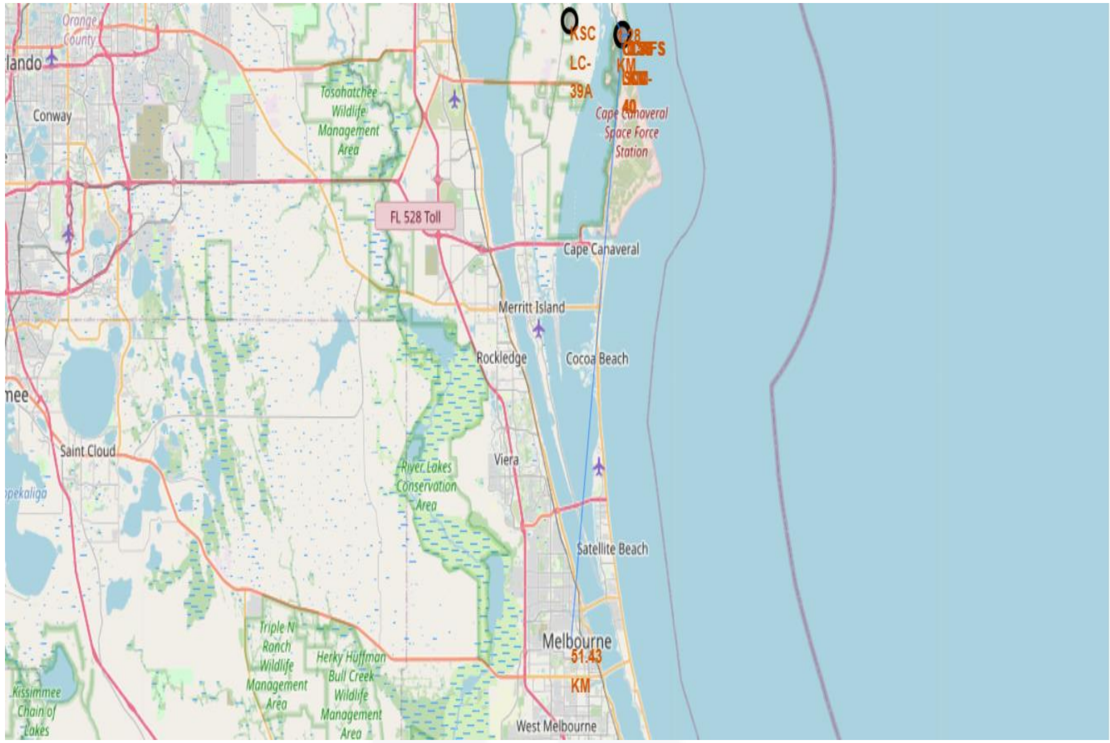


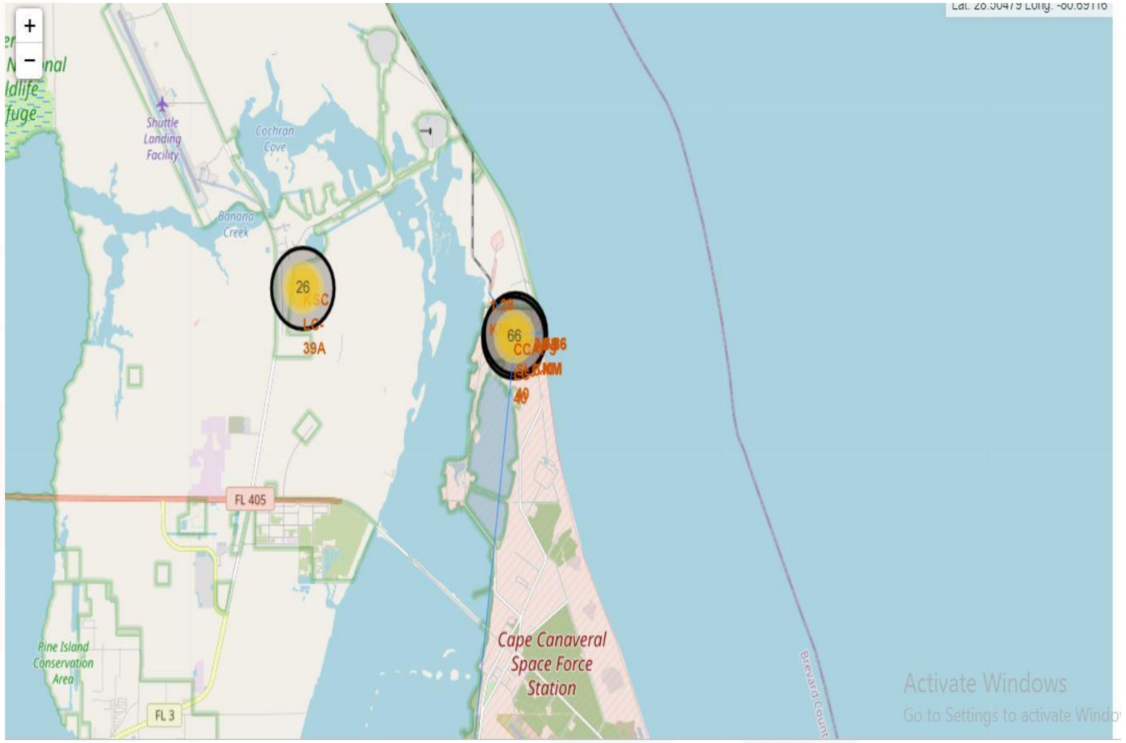
Found: From colour label markers we identify which launch have high success rate where-

- Unsuccessful launch
- Successful launch

Launch Sites on Map

- ✓ TASK 3: Calculate the distances between a launch site to its proximities



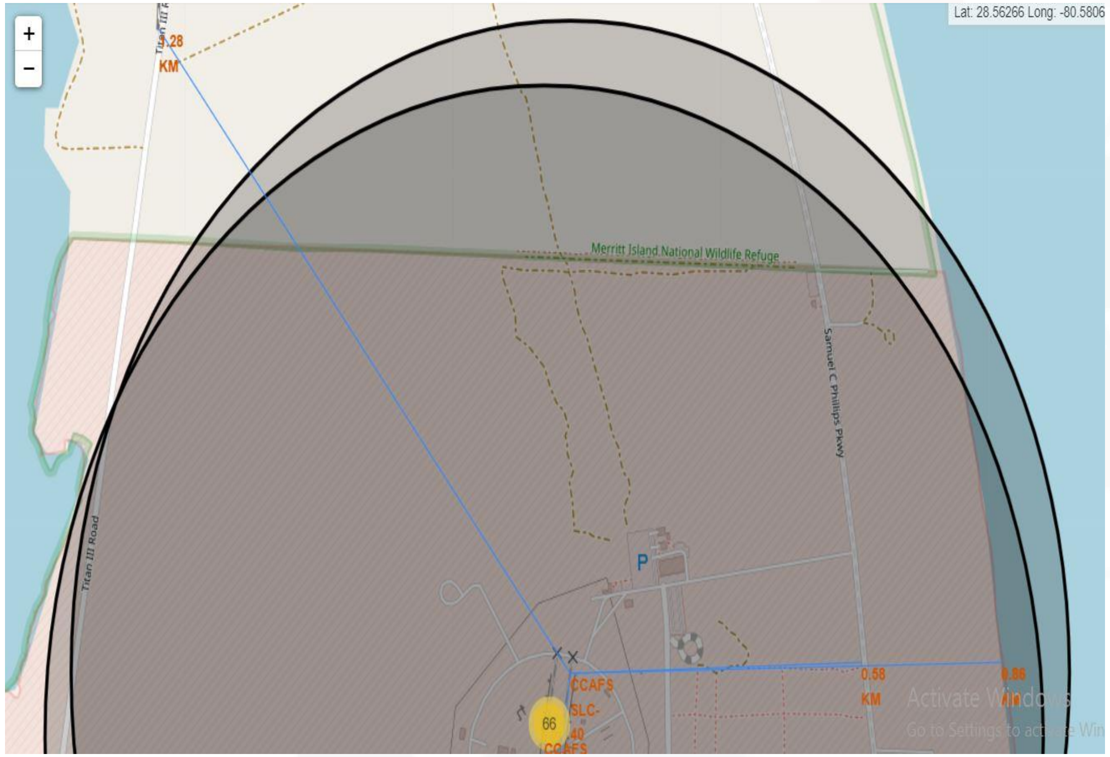


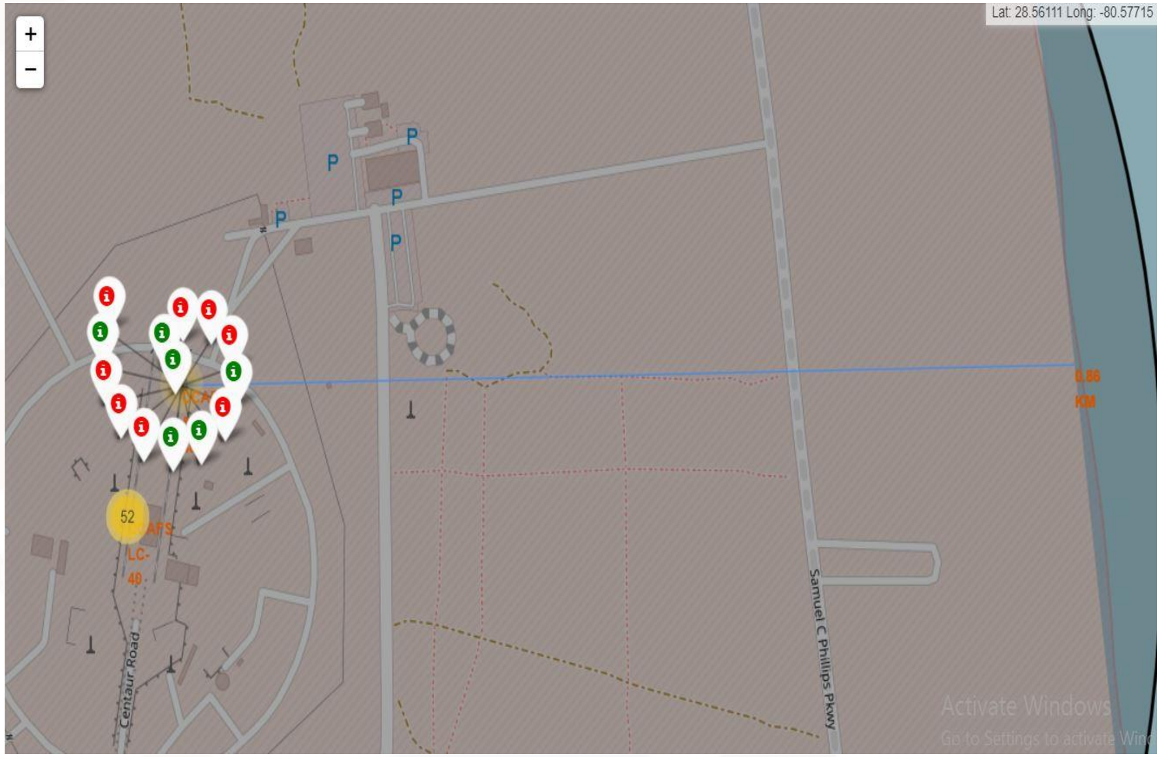

```
In [24]:
distance_highway = calculate_distance(launch_site_lat, launch_site_lon, closest_highway[0], closest_highway[1])
print('distance_highway =',distance_highway, ' km')
distance_railroad = calculate_distance(launch_site_lat, launch_site_lon, closest_railroad[0], closest_railroad[1])
print('distance_railroad =',distance_railroad, ' km')
distance_city = calculate_distance(launch_site_lat, launch_site_lon, closest_city[0], closest_city[1])
print('distance_city =',distance_city, ' km')
```

```
distance_highway = 0.5834695366934144  km
distance_railroad = 1.2845344718142522  km
distance_city = 51.43416999517233  km
```

Launch Sites on Map

✓ TASK 3: Calculate the distances between a launch site to its proximities





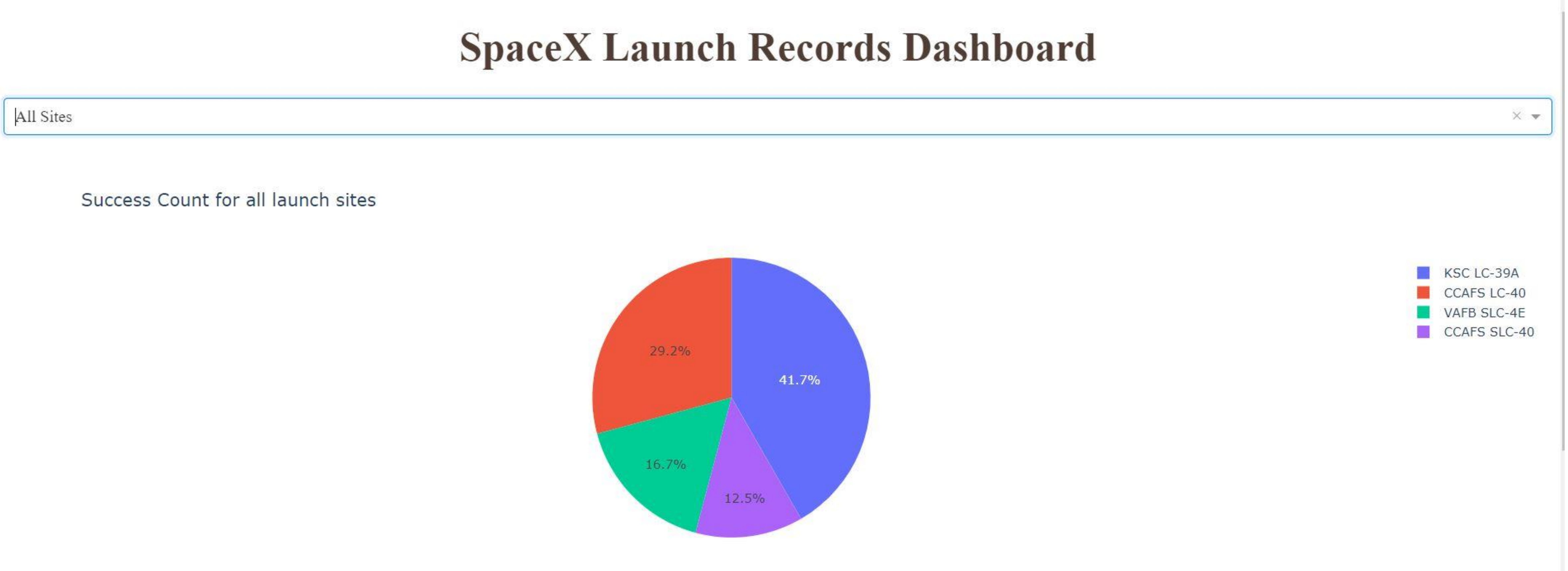
Result: Minimum distance with highway=0.583 km and maximum distance with rail road= 1.284 km from launch site

DASHBOARD

https://khushi7050-8050.theiadocker-1-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/

Dashboard Tab1

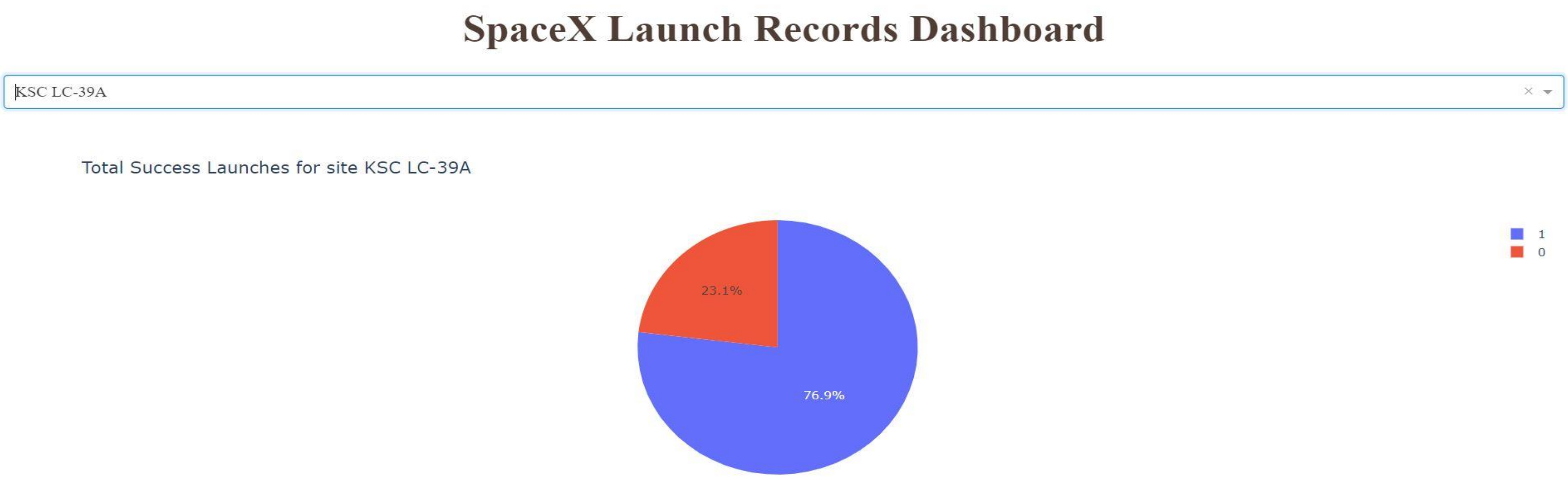
✓ TASK 1: Success count for all launch sites



Result: KSC LC-39A have high success count around 41.7%

Dashboard Tab2

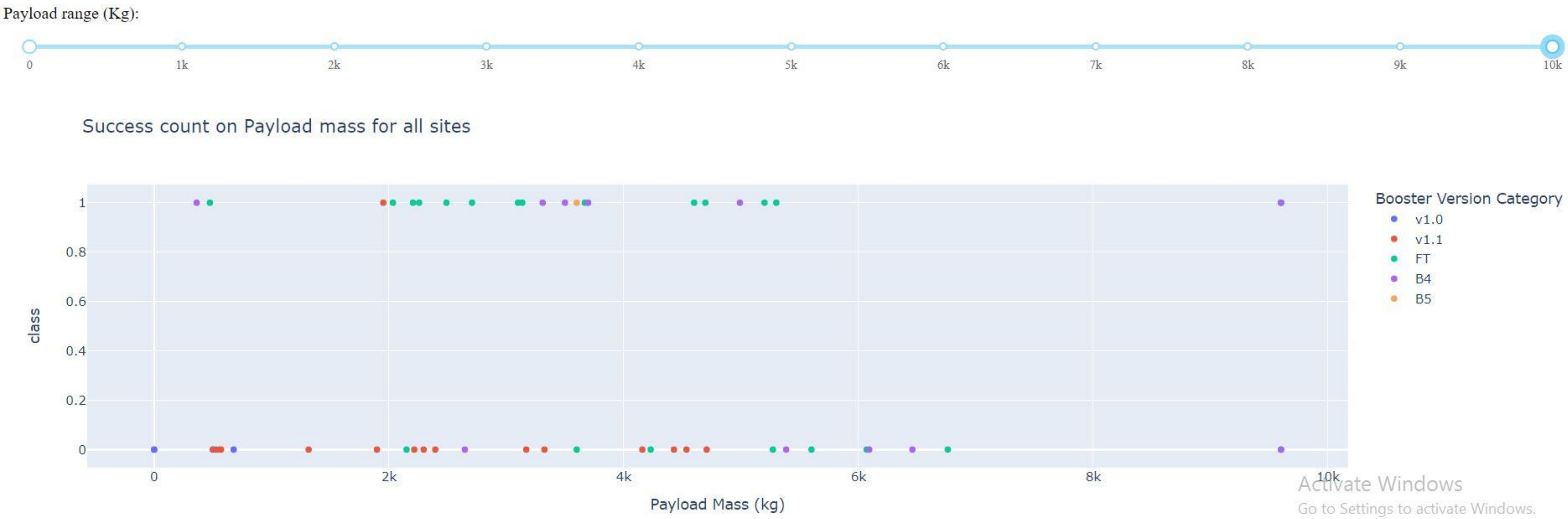
- ✓ TASK 2: Total Success Launches for site KSC LC-39A



Result: KSC LC-39A have 76.9% success rate and 23.1% failure rate

Dashboard Tab3

- ✓ TASK 3: Success count on Payload mass range(0-10,000 kg) for all sites

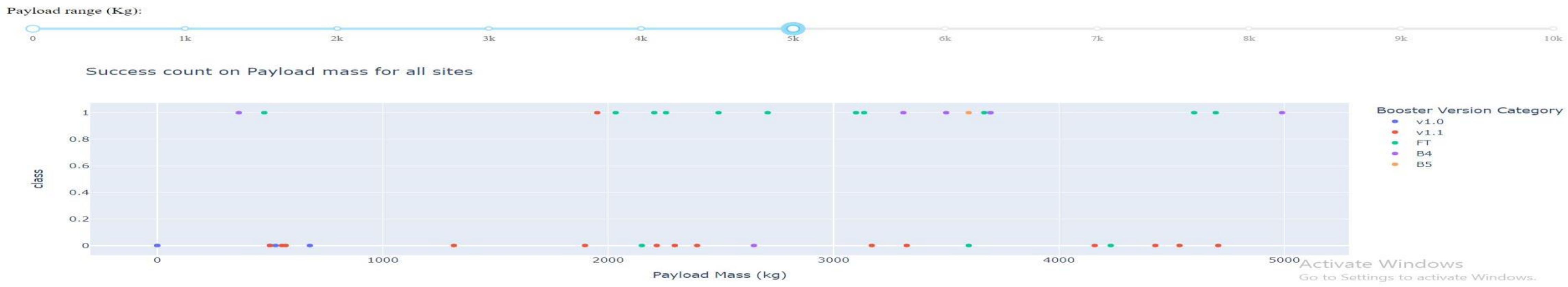


Result: Lower Payload Mass success count is more then higher Payload Mass

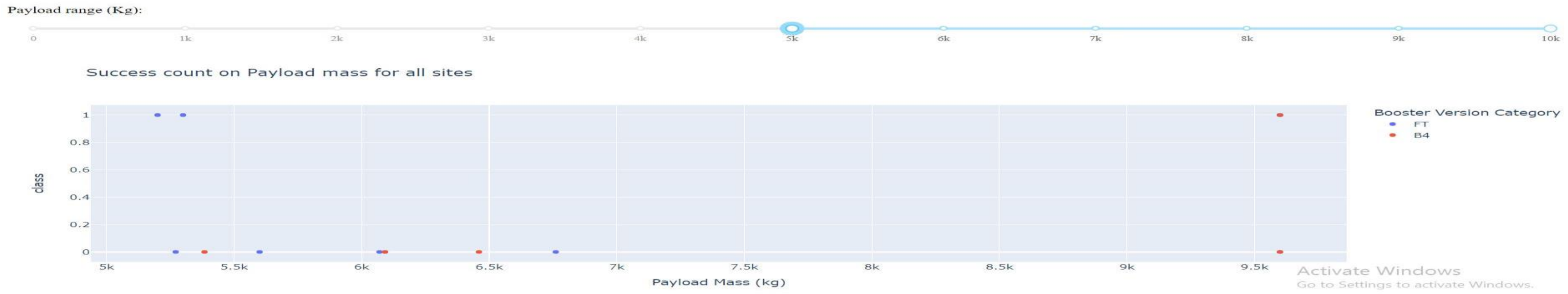
IBM Developer SKILLS NETWORK

Dashboard Tab4

✓ TASK 4: Success count on Lower Payload range and Higher Payload range



Lower payload range 0 to 5000kg



Higher payload range 5000kg to 10,000kg

Classification:

✓ Create a NumPy array from the column Class in data, by applying the method to_numpy() then assign it to the variable Y,make sure the output is a Pandas series (only one bracket df['name of column']).

```
[5]: Y=data['Class'].to_numpy()
     Y
```

```
[5]: array([0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1,
            1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
            1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1,
            1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
            1, 1])
```

✓ Use the function train_test_split to split the data X and Y into training and test data. Set the parameter test_size to 0.2 and random_state to 2. The training data and test data should be assigned to the following labels.

```
X_train, X_test, Y_train, Y_test
```

```
In [8]:
X_train, X_test, Y_train, Y_test = train_test_split( X, Y, test_size=0.2, random_state=2)
print ('Train set:', X_train.shape,  Y_train.shape)
print ('Test set:', X_test.shape,  Y_test.shape)
```

```
Train set: (72, 83) (72,)
Test set: (18, 83) (18,)
```

Classification:

- ✓ We output the GridSearchCV object for logistic regression. We display the best parameters using the data attribute best_params_ and the accuracy on the validation data using the data attribute best_score_.

In [12]:
```
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
```

- ✓ Calculate the accuracy on the test data using the method score:

In [13]:
```
logreg_cv.score(X_test, Y_test)
```

Out[13]:
```
0.8333333333333334
```

Classification:

- ✓ We output the GridSearchCV object for logistic regression. We display the best parameters using the data attribute best_params_ and the accuracy on the validation data using the data attribute best_score_.

In [12]:
```
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
```

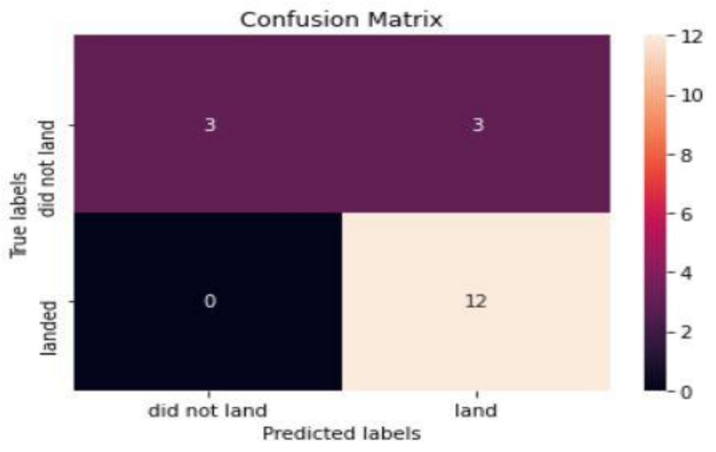
- ✓ Calculate the accuracy on the test data using the method score:

In [13]:
```
logreg_cv.score(X_test, Y_test)
```

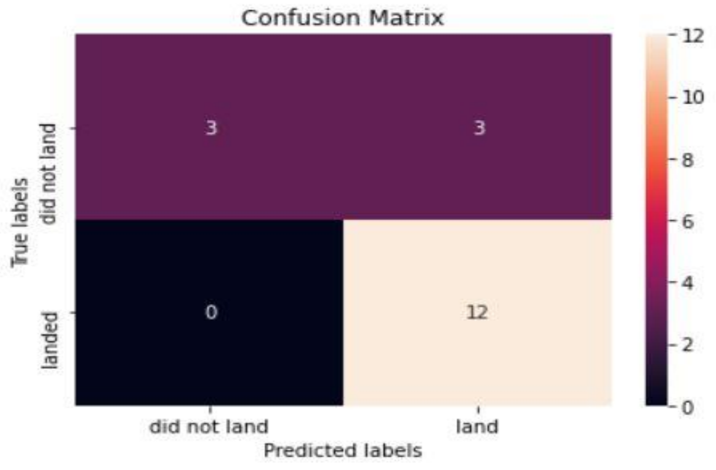
Out[13]:
```
0.8333333333333334
```

Classification:

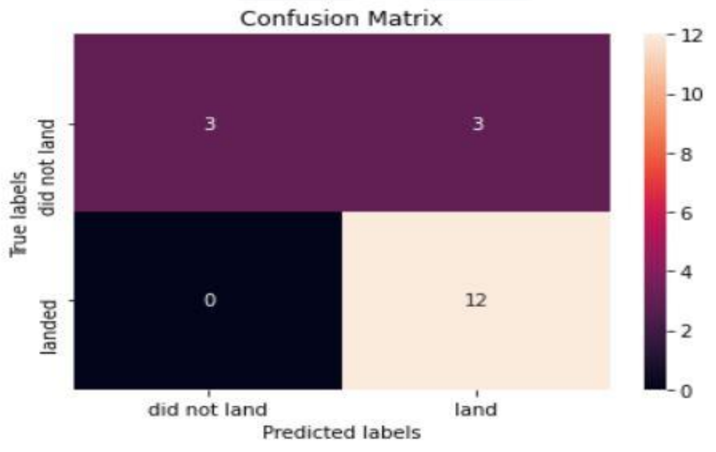
✓ Confusion matrix of Decisions Tree



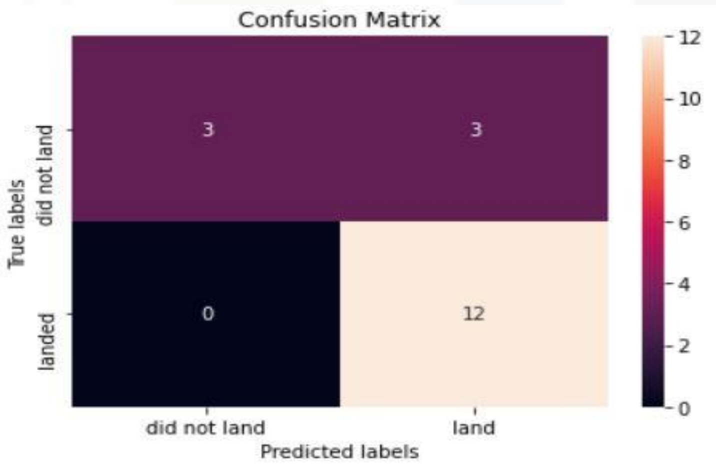
✓ Confusion matrix of Logical Regression



✓ Confusion matrix of SVN



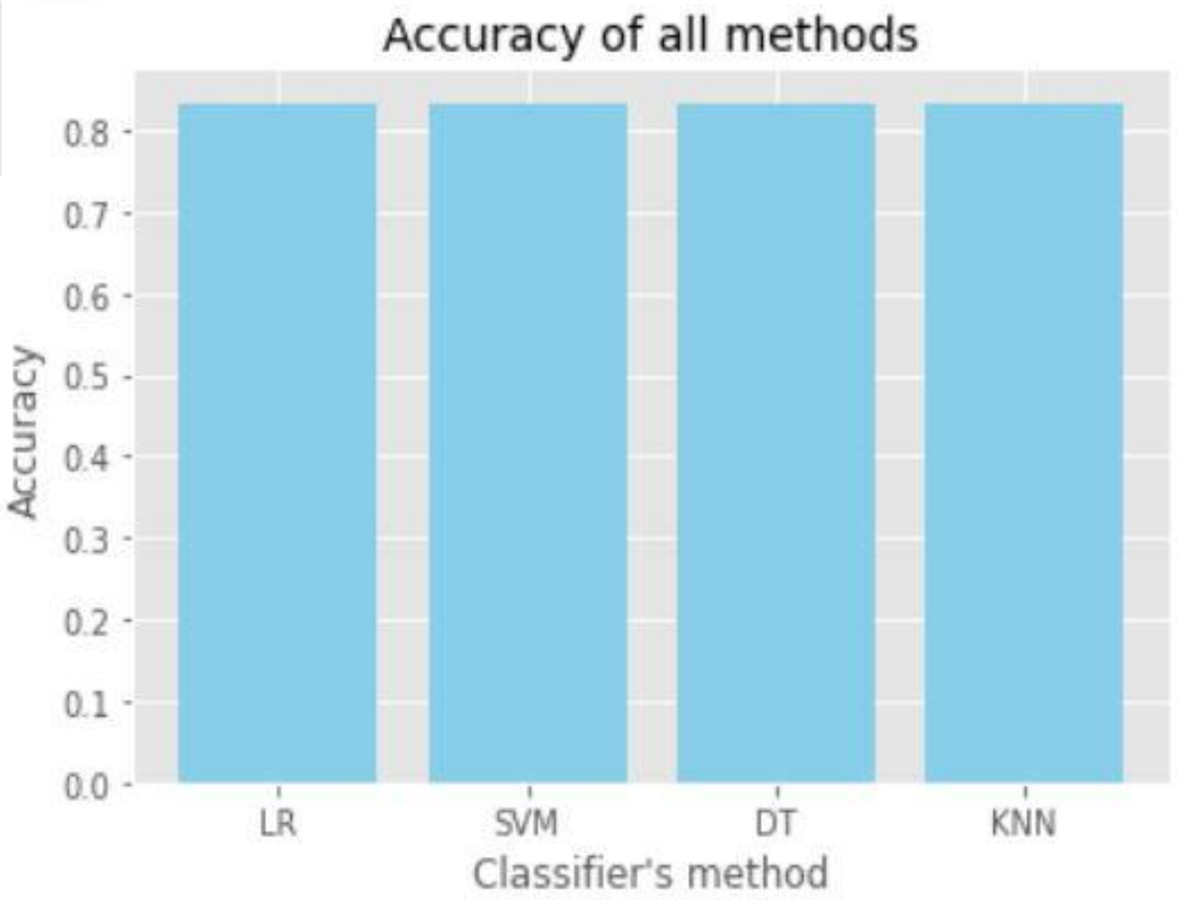
✓ Confusion matrix of KNN



Classification:

```
[29]: print('Accuracy for Logistics Regression method:', logreg_cv.score(X_test, Y_test))
print('Accuracy for Support Vector Machine method:', svm_cv.score(X_test, Y_test))
print('Accuracy for Decision tree method:', tree_cv.score(X_test, Y_test))
print('Accuracy for K nearsdt neighbors method:', knn_cv.score(X_test, Y_test))
```

```
Accuracy for Logistics Regression method: 0.8333333333333334
Accuracy for Support Vector Machine method: 0.8333333333333334
Accuracy for Decision tree method: 0.8333333333333334
Accuracy for K nearsdt neighbors method: 0.8333333333333334
```



RESULT-

- Logistic Regression, Support Vector Machines, K-nearest neighbours and Decision Tree have the same accuracy 83.3%

- All Models give us same accuracy.

DISCUSSION

→ CCAFS SLC 40 had higher Launch Site among all and higher Launch Site when Payload Mass(kg)<7000.

→ Launch success rate had increased from 2013 to 2020.

→ launch sites are in proximity to the equator and the coast. This makes sense as it takes less fuel to get into space from the equator due to the physics of Earth's rotation. The launch sites in close proximity to the coast are also logical for safety reasons.

→The distances between a launch site to its proximities is as follows:

- Highway distance=0.5834
- Railroad distance=1.2845
- City distance=51.4341

→ KSC LC-39A had high success count around 41.7% among all and that had 76.9% success rate and 23.1% failure rate

→ Low weighted payload like 0 to 5000kg performs better then higher payload like 5000kg to 10,000 kg.

→Logistic Regression, Support Vector Machines, K-nearest neighbours and Decision Tree have the same accuracy 83.3% .

CONCLUSION

VISUALIZATION CONCLUSION-

- CCAFS SLC 40 have higher Launch Site among all.
- CCAFS SLC 40 have higher Launch Site when Payload Mass(kg)<7000.
- Flight Number between 60-80, orbit type is VLEO.
- Strong relationship between ISS orbit v/s Payload Mass range(2000-3000) and between GTO orbit v/s Payload Mass range(4000-8000).
- Launch success rate has increased from 2013 to 2020.

LAUNCH SITE CONCLUSION-

- *launch sites are in proximity to the equator and the coast.* The launch sites in close proximity to the coast for safety reasons.
- the distances between a launch site to its proximities is as follows:
    - Highway distance=0.5834
    - Railroad distance=1.2845
    - City distance=51.4341

DASHBOARD CONCLUSION

- KSC LC-39A have high success count around 41.7% among all.
- KSC LC-39A have 76.9% success rate and 23.1% failure rate.
- Low weighted payload performs better then higher payload.

CLASSIFICATION CONCLUSION

- Logistic Regression, Support Vector Machines, K-nearest neighbours and Decision Tree have the same accuracy 83.3%

APPENDIX

- DATA COLLECTION API- https://github.com/khushiyadav2022/capstone-project/blob/b1951e520e8caa1965963ed6931ada1d90a44aa7/jupyter-labs-spacex-data-collection-api.ipynb
- WEB SCRAPPING- https://github.com/khushiyadav2022/capstone-project/blob/b1951e520e8caa1965963ed6931ada1d90a44aa7/jupyter-labs-webscraping.ipynb
- DATA WRANGLING- https://github.com/khushiyadav2022/capstone-project/blob/b1951e520e8caa1965963ed6931ada1d90a44aa7/jupyter-labs-webscraping.ipynb
- EDA with SQL- https://github.com/khushiyadav2022/capstone-project/blob/b1951e520e8caa1965963ed6931ada1d90a44aa7/jupyter-labs-eda-sql-coursera_sqllite%20(1).ipynb
- DATA VISUALIZATION- https://github.com/khushiyadav2022/capstone-project/blob/b1951e520e8caa1965963ed6931ada1d90a44aa7/jupyter-labs-eda-sql-coursera_sqllite%20(1).ipynb
- SITE LAUNCHING- https://github.com/khushiyadav2022/capstone-project/blob/b1951e520e8caa1965963ed6931ada1d90a44aa7/lab_jupyter_launch_site_location.ipynb
- DASHBOARD- https://github.com/khushiyadav2022/capstone-project/blob/b1951e520e8caa1965963ed6931ada1d90a44aa7/spacex_dash_app.py
- CLASSSIFICATION MODELS- https://github.com/khushiyadav2022/capstone-project/blob/ac7e6e71ab8d900533a0a77d628a479a785524de/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb


THANK YOU :)