IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Hitesh Tolani

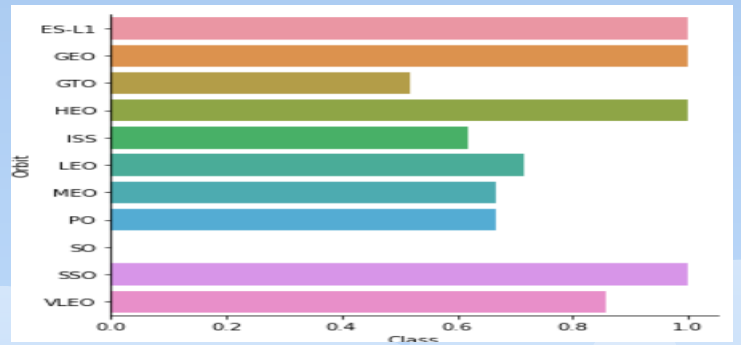
3-2-2022

Table of Contents

Topic	Slide Number
Executive Summary	3 & 4
Introduction	5
Methodology	6 to 22
Results	52 & 53
Conclusion	54
Appendix	55

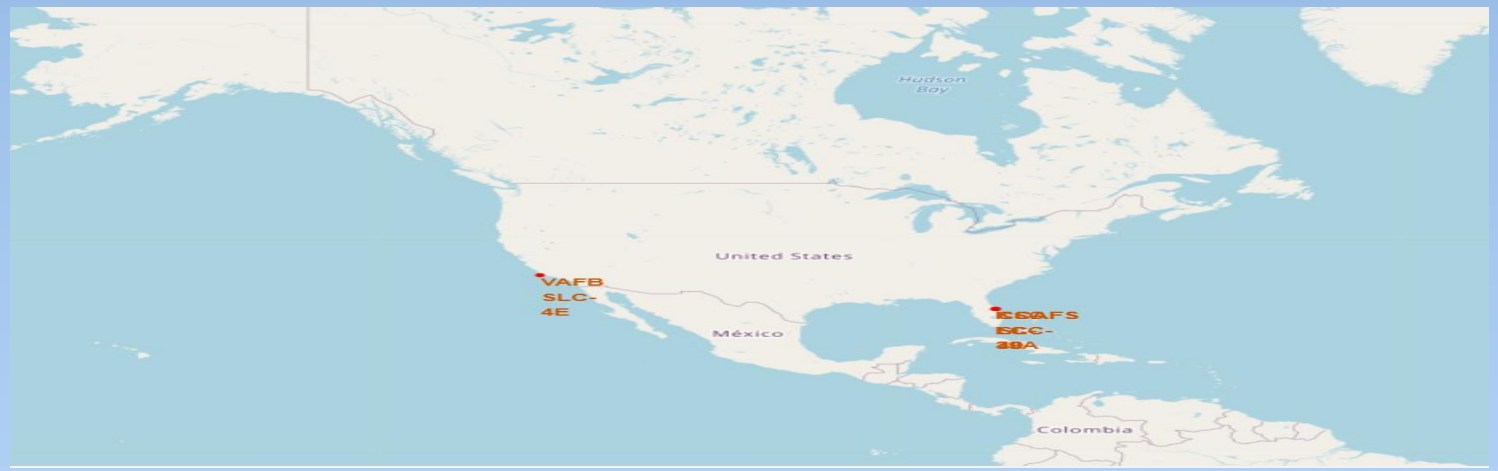
Executive Summary

- Data Collection
  - Data is mainly collected using SpaceX Rest API . The endpoints of SpaceX REST API start with api.spacexdata.com/v4/
- Data Wrangling
  - In the dataframe , the column outcome has 8 possible values (True Ocean , False Ocean , True RTLS, False RTLS , True ASDS , False ASDS , None ASDS and None None).
- EDA using Visualization and SQL



Executive Summary

- Interactive visual analytics using Folium and Plotly Dash



- Predictive analysis using classification models
  - We trained a KNN model with ~84%,this was found to be the performing model which was then used to predict the success rate of the launches

Introduction

- We will try to determine if the first stage will land successfully.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. This information can be used if an alternate company wants to bid against space X for a rocket launch.
- Problems that need to be solved.
  - What is the best booster version?
  - What is the most suitable orbit for launches?
  - What is the most suitable launch site?

Section 1

Methodology

Data Collection Methodology

- Data is mainly collected using SpaceX Rest API.
- The API will provide us data regarding rockets used, payload delivered , launch specifications , landing specifications and landing outcome of launches
- The endpoints of SpaceX REST API start with api.spacexdata.com/v4/

Data Collection Methodology

- After Data is collected using SpaceX REST API ,the data is normalized and then convert into pandas dataframe for further analysis.

- Another source used for data collection is wikipages for extracting Falcon 9 launch data .

Data Collection SpaceX API

Notebook

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

⬇

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

⬇

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

⬇

```
# 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')
```

⬇

```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
```

⬇

```
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']=[]
```

⬇

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

Data Collection Scraping

Notebook

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

⬇

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

⬇

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

⬇

```
# 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')
```

⬇

```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
```

⬇

```
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']=[]
```

⬇

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

Data Wrangling

- In the dataframe , the column outcome has 8 possible values (True Ocean , False Ocean , True RTLS, False RTLS , True ASDS , False ASDS , None ASDS and None None).

- True Ocean means the rocket was landed successfully to a specific part of ocean. False Ocean means the rocket was landed unsuccessfully to a specific part of ocean.

Data Wrangling

- True RTLS means the rocket was landed successfully to a ground pad. False RTLS means the rocket was landed unsuccessfully to a ground pad .

- True ASDS means the rocket was landed successfully to a drone ship. False ASDS means the rocket was landed unsuccessfully to drone ship.

- None ASDS and None None means the rocket was landed unsuccessfully.

Data Wrangling

[Notebook]

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

```
# Apply value_counts() on column LaunchSite
df["LaunchSite"].value_counts()
```

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

```
# landing_outcomes = values on Outcome column

landing_outcomes = df["Outcome"].value_counts()
```

```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

```
# 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)
```

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

EDA with Data Visualization

Scatter Plot :

It is used to represent the correlation between the variables in the dataset .

Bar Plot :

It is used to presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent.

Line Plot:

It displays data as marks above a number line , showing the frequency of each value.

Scatter Plot plotted:

- Flight Number VS Launch Site
- Payload VS Launch Site
- Flight Number VS Orbit type
- Payload and Orbit type

Bar Plot Plotted :

- Orbit vs Success Rate

Line Plot Plotted:

- Success Rate VS Year

Notebook

EDA with SQL

The following SQL queries where used to perform EDA on the dataset:

- 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 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.

Notebook

Build an Interactive Map with Folium

- All launch sites where marked on the map.
- Circles were added to highlight a circle area and add label to it.
- Marker Cluster was used to mark successful and failed launches on the map.
- Polyline was used to represent the distance between the launch site and coastline.

- Are launch sites in close proximity to railways? Yes.
- Are launch sites in close proximity to highways? Yes.
- Are launch sites in close proximity to coastline? Yes.
- Do launch sites keep certain distance away from cities? Yes.

Notebook

Build a Dashboard with Plotly Dash.

Pie Chart :

A pie chart is a circular statistical graphic, which is divided into slices to illustrate numerical proportion. In a pie chart, the arc length of each slice, is proportional to the quantity it represents.

A pie chart was plotted to examine the number of successful launches per site.

Scatter Plot :

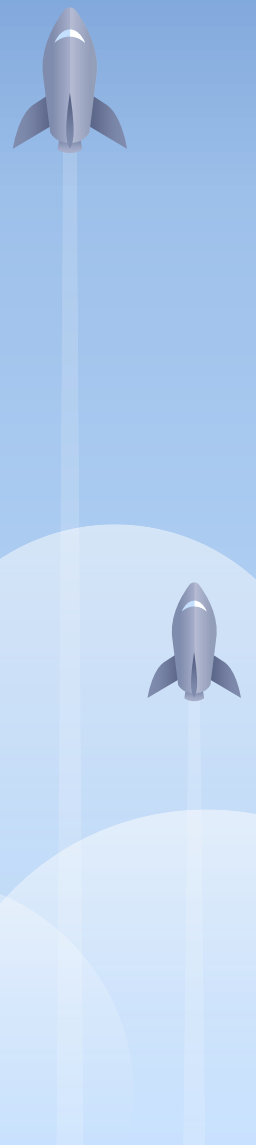
It is used to represent the correlation between the variables in the dataset .

A scatter plot was plotted to represent the Success rate VS payload mass
Of each booster version category.

Notebook

Predictive Analysis (Classification)

- The model was built using python's sci-kit learn library and with the help of pandas and numpy library.
- The dataset was splitted into train and test parts with the test dataset containing 20% of the original dataset.
- Cross-validation and Hyper parameter tuning was done with help of GridSearchCV with cv being equal to 10.
- The best model was picked with help of best_params_ which was found to be KNN with accuracy of 83%.



Predictive Analysis (Classification)

Notebook

```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size =0.2 ,random_state = 2)
```

⬇

```
parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
              'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
              'p': [1,2]}

KNN = KNeighborsClassifier()
knn_cv = GridSearchCV(KNN,parameters, cv = 10)
```

⬇

```
knn_cv.fit(X_train,Y_train)
```

⬇

```
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
```

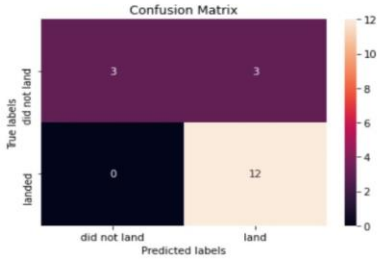
⬇

```
knn_cv.score(X_test,Y_test)
```

```
0.8333333333333334
```

⬇

```
yhat = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

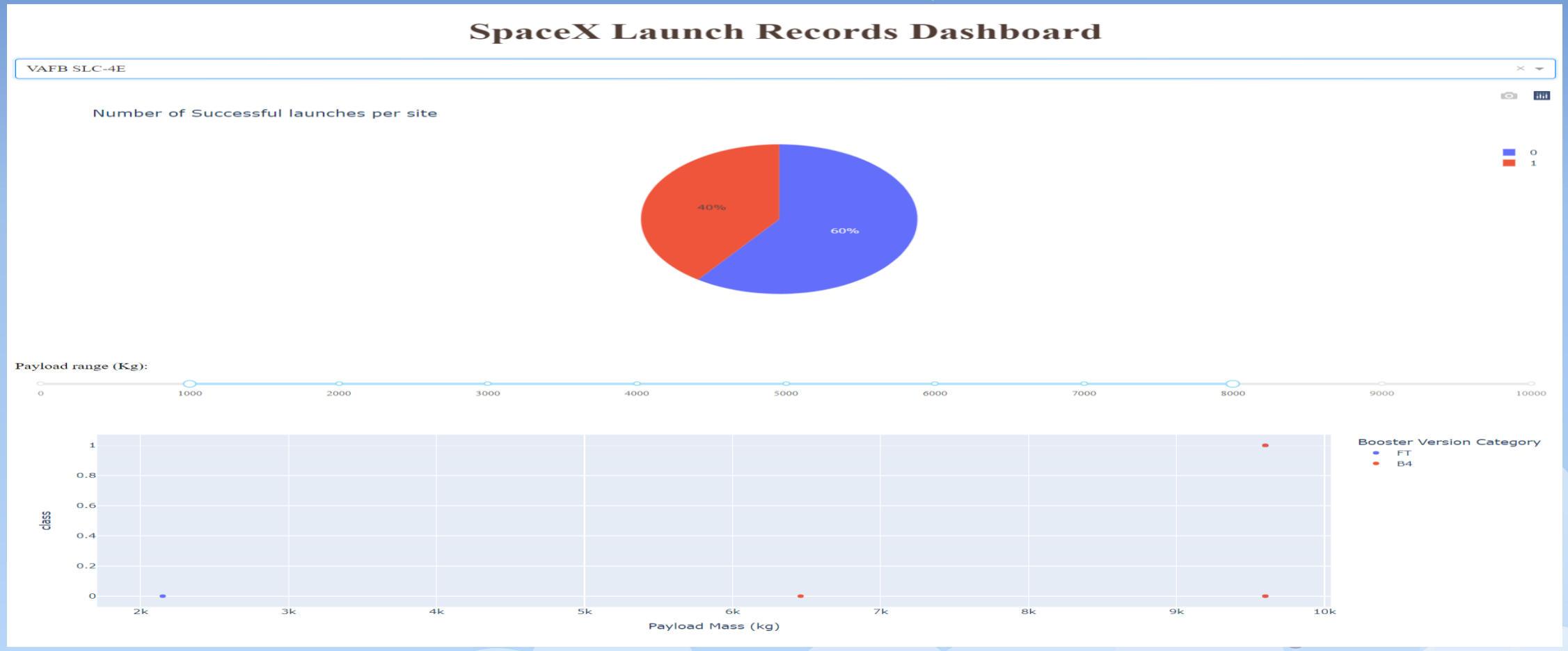


Results

	FlightNumber	PayloadMass	Flights	GridFins	Reused	Legs	Block	ReusedCount	Orbit_ES-L1	Orbit_GEO	...	Serial_B1048	Serial_B1049	Serial_B1050	Serial_B1051	Serial_B1054	Serial_B1056	Serial_B1058	Serial_B1059	Serial_B1060	Serial_B1062
0	1.0	6104.959412	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	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	0.0	0.0	0.0	0.0	0.0	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	0.0	0.0	0.0	0.0	0.0	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	0.0	0.0	0.0	0.0	0.0	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	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
85	86.0	15400.000000	2.0	1.0	1.0	1.0	5.0	2.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
86	87.0	15400.000000	3.0	1.0	1.0	1.0	5.0	2.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
87	88.0	15400.000000	6.0	1.0	1.0	1.0	5.0	5.0	0.0	0.0	...	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
88	89.0	15400.000000	3.0	1.0	1.0	1.0	5.0	2.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
89	90.0	3681.000000	1.0	1.0	0.0	1.0	5.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0

Exploratory Data Analysis

Results



Interactive analytics demo in screenshots

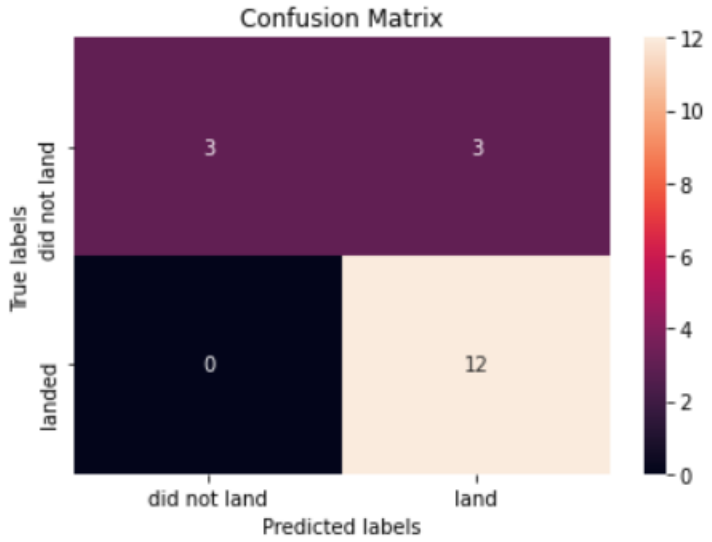
Results

```
knn_cv.score(X_test,Y_test)
```

```
0.8333333333333334
```

We can plot the confusion matrix

```
yhat = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

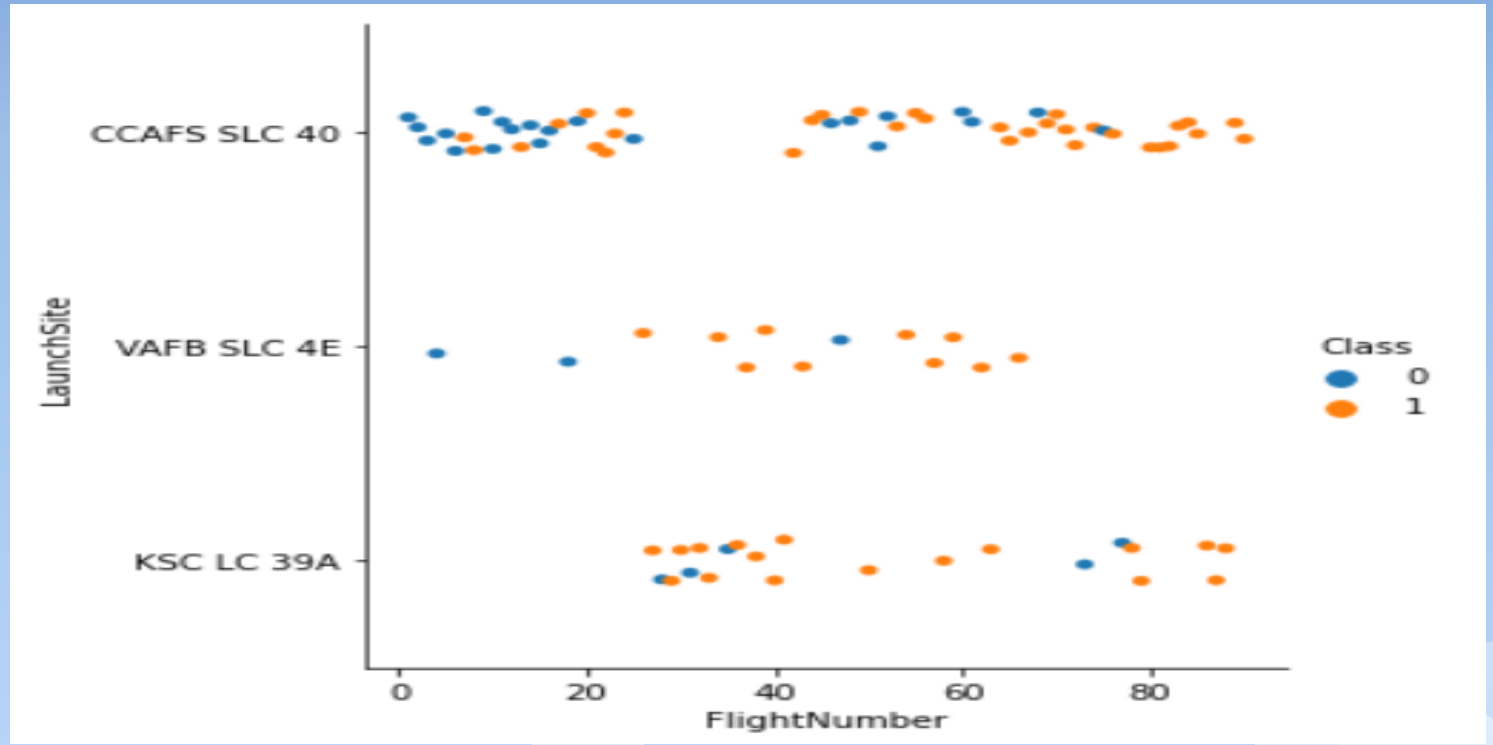


Predictive analysis Results

Section 2

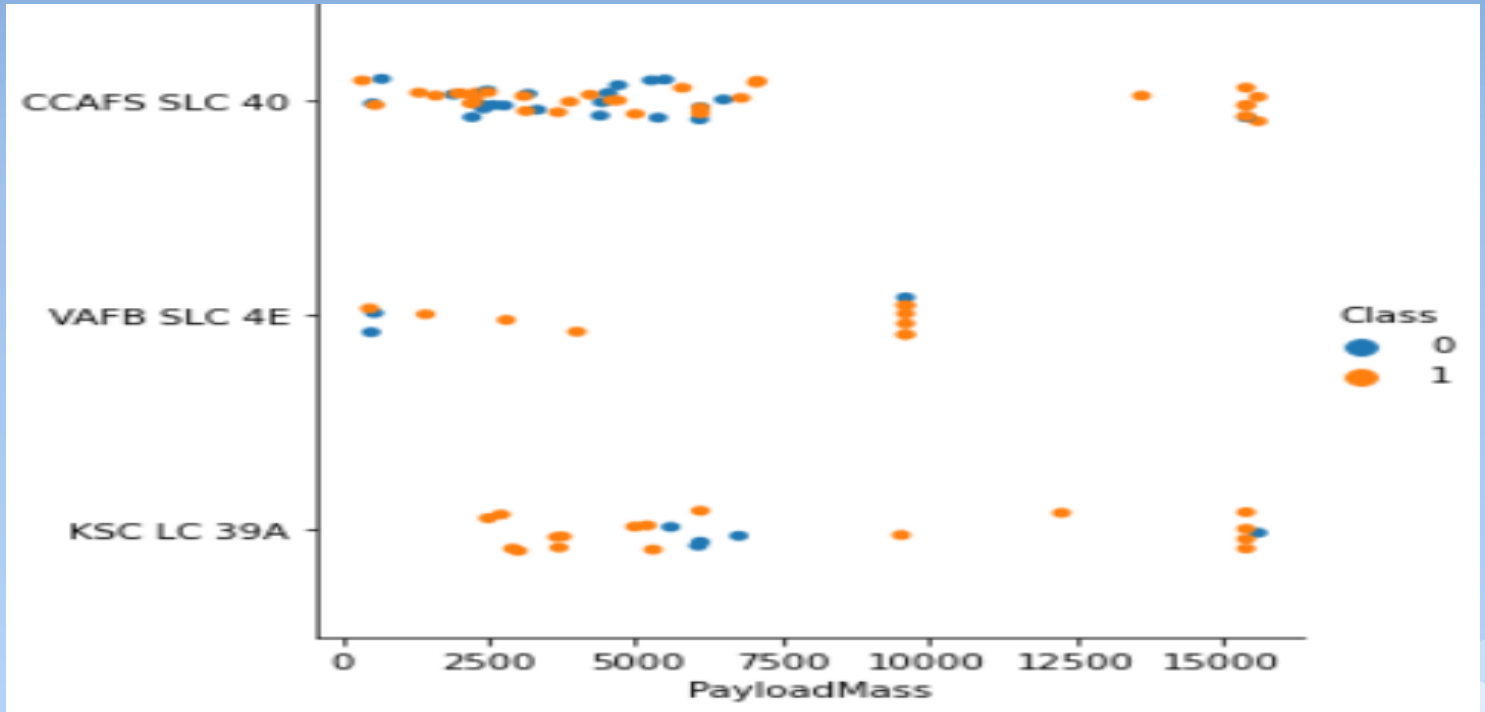
Insights drawn from EDA

Flight Number vs Launch Site



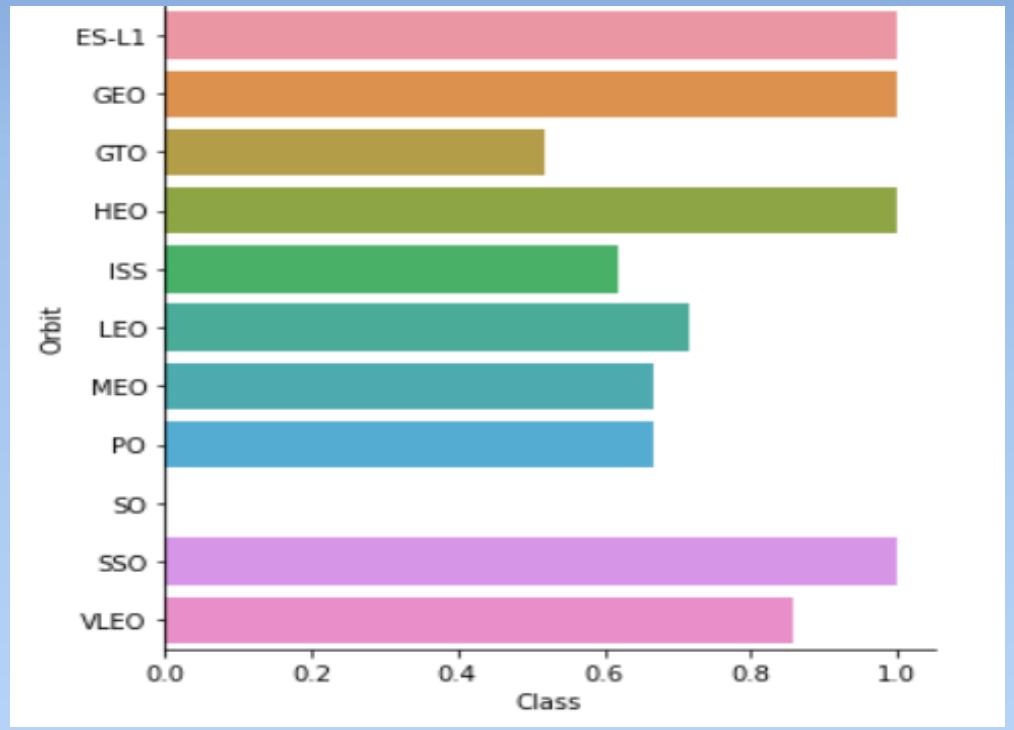
- The success rate of KSC LC 39 A and VAFB SLC 4E appears to be significantly more than CCAFS SLC 40 .

Payload vs Launch Site



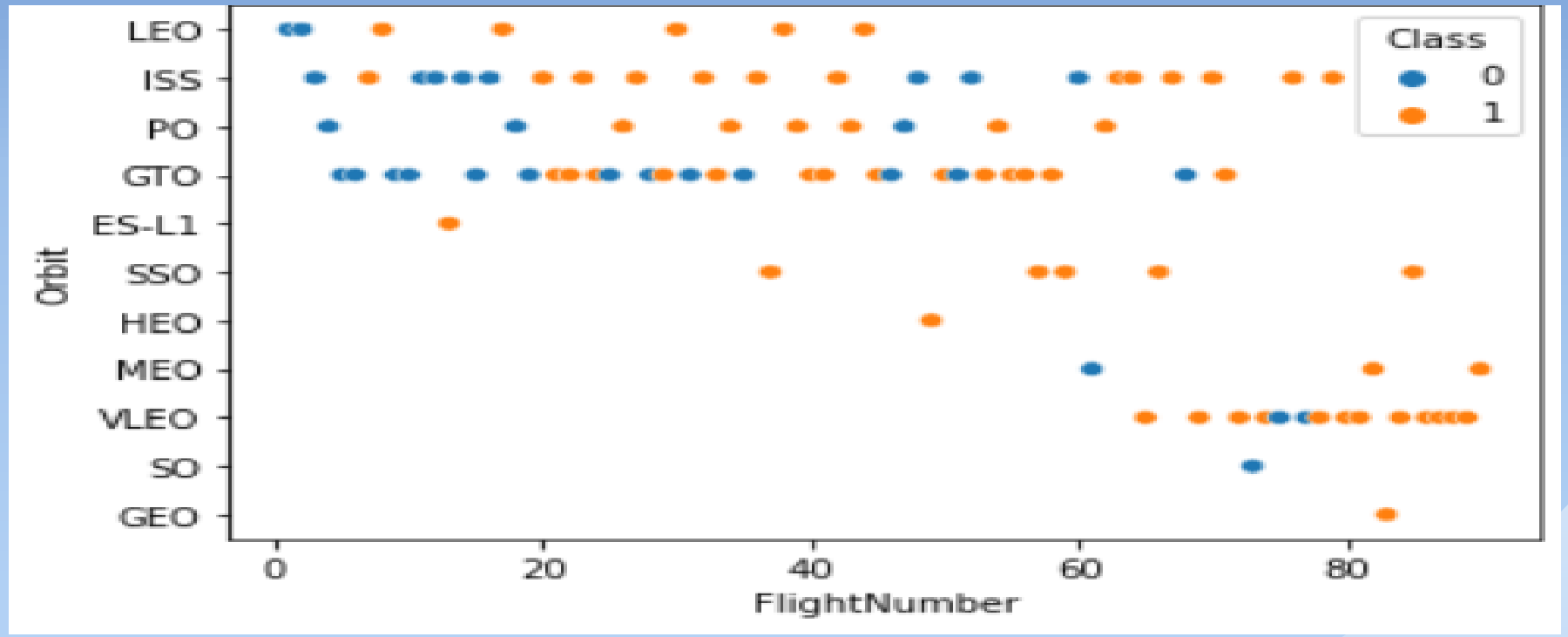
- The success rate of KSC LC 39 A and CCAFS SLC 40 appears to be significantly more than VAFB SLC 4E for payload mass indicating that they are more suitable for heavy payload.

Success rate vs Orbit Type



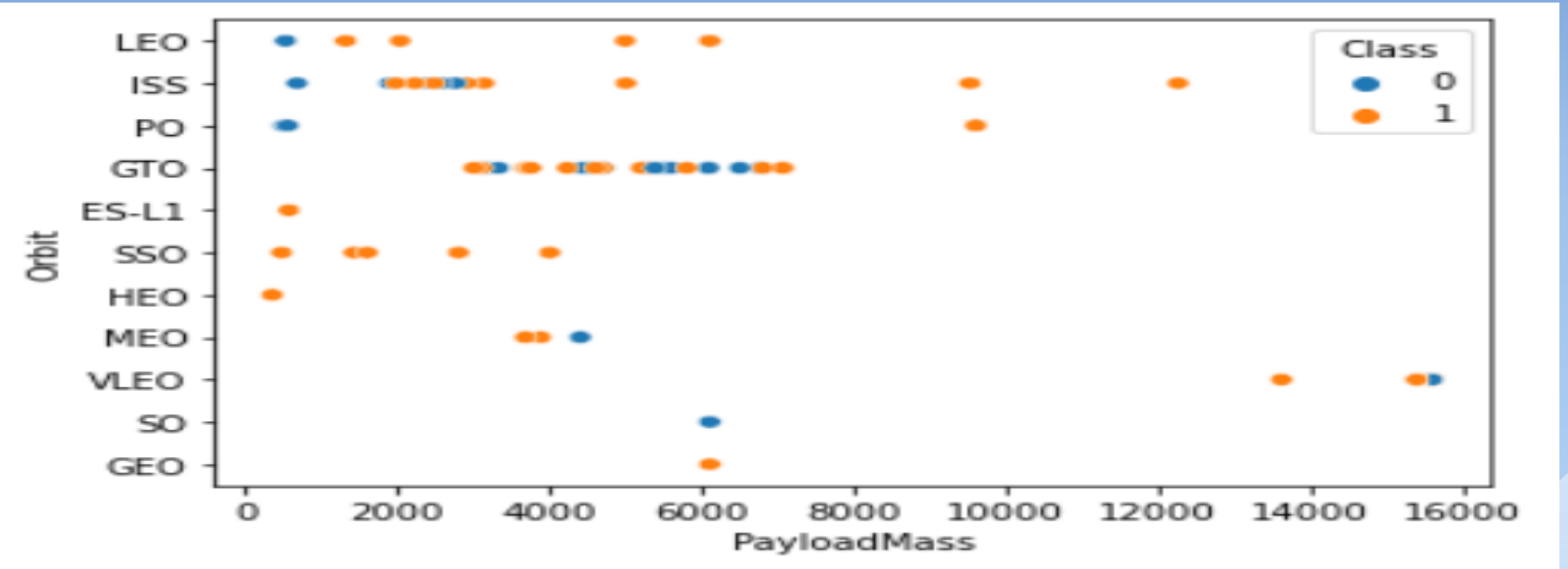
- The success rate of ES-L1,GEO ,HEO ,SSO and VLEO appears to be significantly more than others indicating that they are more suitable for launches. SO turns out to be most unsuitable orbit for rocket launches but it can be also due to lack of data as SO has had only one launch and that too had failed

Flight number vs Orbit type



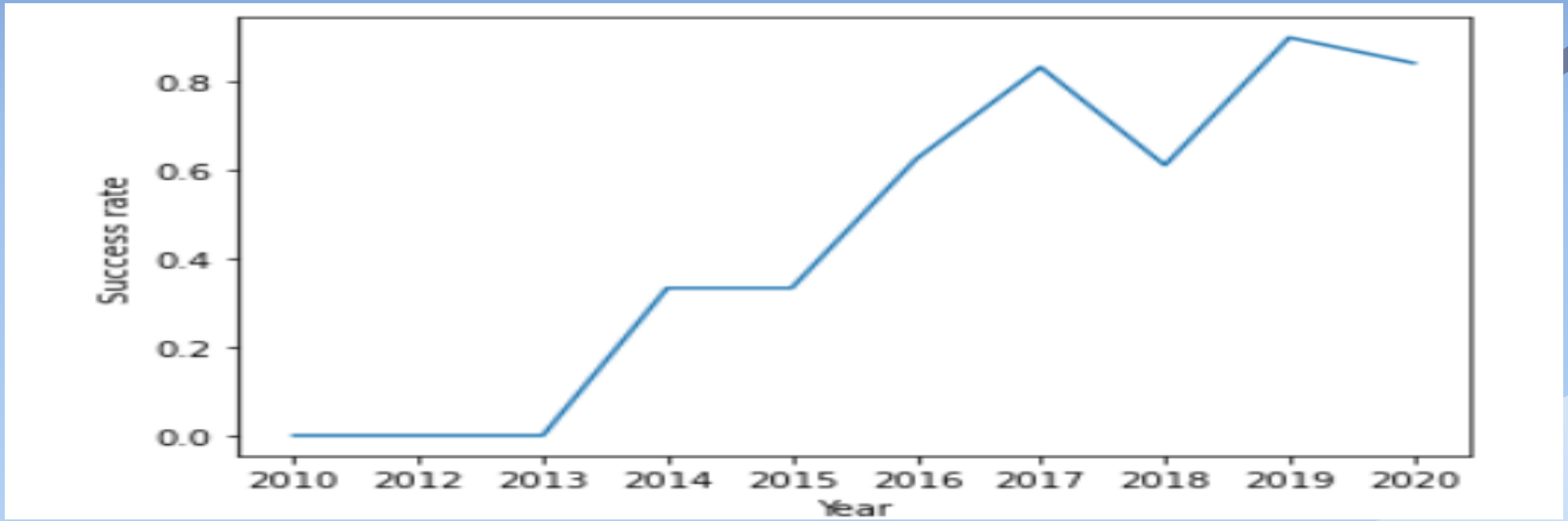
- We can see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

Payload vs Orbit type



- SSO appears to be the best choice for payloads in range 0 to 4000kg while for higher payloads ISS and VLEO appear to be a good choice for orbit.

Payload vs Orbit type



- The success rate appears to drastically change in the from the year 2013 and had taken a significant dip in the year 2018.

All launch site names

```
%sql SELECT DISTINCT(launch_site) FROM SPACEXTBL
```

```
 * ibm_db_sa://qjm40322:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/bludb
Done.
```

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

- DISTINCT clause is used to show all the unique values in the column launch_site.

Launch Site Names Begin with ‘CCA’

```
%sql SELECT * FROM SPACEXTBL WHERE launch_site LIKE 'CCA%' LIMIT 5
```

```
 * ibm_db_sa://qjm40322:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/bludb
Done.
```

DATE	time_utc_	booster_version	launch_site	payload	payload_mass_kg_	orbit	customer	mission_outcome	landing_outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

- LIMIT clause is used to restrict the number of rows/records to be displayed in this case to 5 .

Launch Site Names Begin with ‘CCA’

```
%sql SELECT * FROM SPACEXTBL WHERE launch_site LIKE 'CCA%' LIMIT 5
```

```
 * ibm_db_sa://qjm40322:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/bludb
Done.
```

DATE	time_utc_	booster_version	launch_site	payload	payload_mass_kg_	orbit	customer	mission_outcome	landing_outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

- LIMIT clause is used to restrict the number of rows/records to be displayed in this case to 5 .

Total Payload Mass

```
%sql SELECT SUM(payload_mass__kg_) as Total FROM SPACEXTBL WHERE customer LIKE 'NASA (CRS)'
```

```
 * ibm_db_sa://qjm40322:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/bludb
Done.
```

total
45596

- SUM is used to sum the column payload_mass_kg_ , LIKE ‘NASA (CRS)’ is used to allow only those rows where customer column value is ‘NASA (CRS)’.

Average Payload Mass by F9 v1.1

```
%sql SELECT AVG(payload_mass__kg_) as avg_payload FROM SPACEXTBL WHERE booster_version LIKE 'F9 v1.1%'
```

```
 * ibm_db_sa://qjm40322:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/bludb
Done.
```

avg_payload
2534

- AVG is used to average the column payload_mass_kg_ , LIKE ‘F9 v1.1%’ is used to allow only those rows where booster_version column value is ‘F9 v1.1%’.

First Successful Ground Landing Date

```
%sql SELECT MIN(DATE) as DATE FROM SPACEXTBL WHERE landing__outcome LIKE 'Success (ground pad)'
```

```
 * ibm_db_sa://qjm40322:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/bludb
Done.
```

DATE
2015-12-22

- MIN is used to find the minimum of the column DATE , LIKE ‘Success (ground pad)’ is used to allow only those rows where landing_outcome column value is ‘Success (ground pad)’.

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql SELECT Booster_Version as Booster FROM SPACEXTBL WHERE landing__outcome LIKE 'Success (drone ship)' and payload_mass__kg_ > 4000 and payload_mass_
```

```
 * ibm_db_sa://qjm40322:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/bludb
Done.
```

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

- LIKE ‘Success (drone ship)’ is used to allow only those rows where landing_outcome column value is ‘Success (drone ship)’.

Total Number of Successful and Failure Mission Outcomes

```
%sql SELECT COUNT(mission_outcome) AS NUM_OF_MISSION_OUTCOMES FROM SPACEXTBL
```

```
 * ibm_db_sa://qjm40322:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/bludb
Done.
```

num_of_mission_outcomes
101

- COUNT is used find the number of values in a column in this case mission_outcome being the column.

Boosters Carried Maximum Payload

```
%sql SELECT Booster_Version as Booster FROM SPACEXTBL WHERE payload_mass__kg_ = (select MAX(payload_mass__kg_) FROM SPACEXTBL)
```

```
 * ibm_db_sa://qjm40322:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/bludb
Done.
```

booster
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

- MAX is used find the maximum value in a column in this case payload_mass_kg_ being the column and this max value is then compared with all the values of payload_mass_kg_ to find the booster version with maximum payload capacity.

2015 Launch Records

```
%sql SELECT booster_version,launch_site,landing__outcome FROM SPACEXTBL WHERE landing__outcome = 'Failure (drone ship)' and YEAR(DATE) = 2015
```

```
 * ibm_db_sa://qjm40322:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/bludb
Done.
```

booster_version	launch_site	landing__outcome
F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

- YEAR is used to extract the year from the column DATE and to compare it against 2015 to find the booster version,launch_site and landing_outcome.

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

```
%sql SELECT DATE,COUNT(landing__outcome) AS COUNT FROM SPACEXTBL GROUP BY DATE having DATE BETWEEN '2010-06-04' and '2017-03-20' ORDER BY COUNT DESC
```

```
 * ibm_db_sa://qjm40322:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/bludb
Done.
```

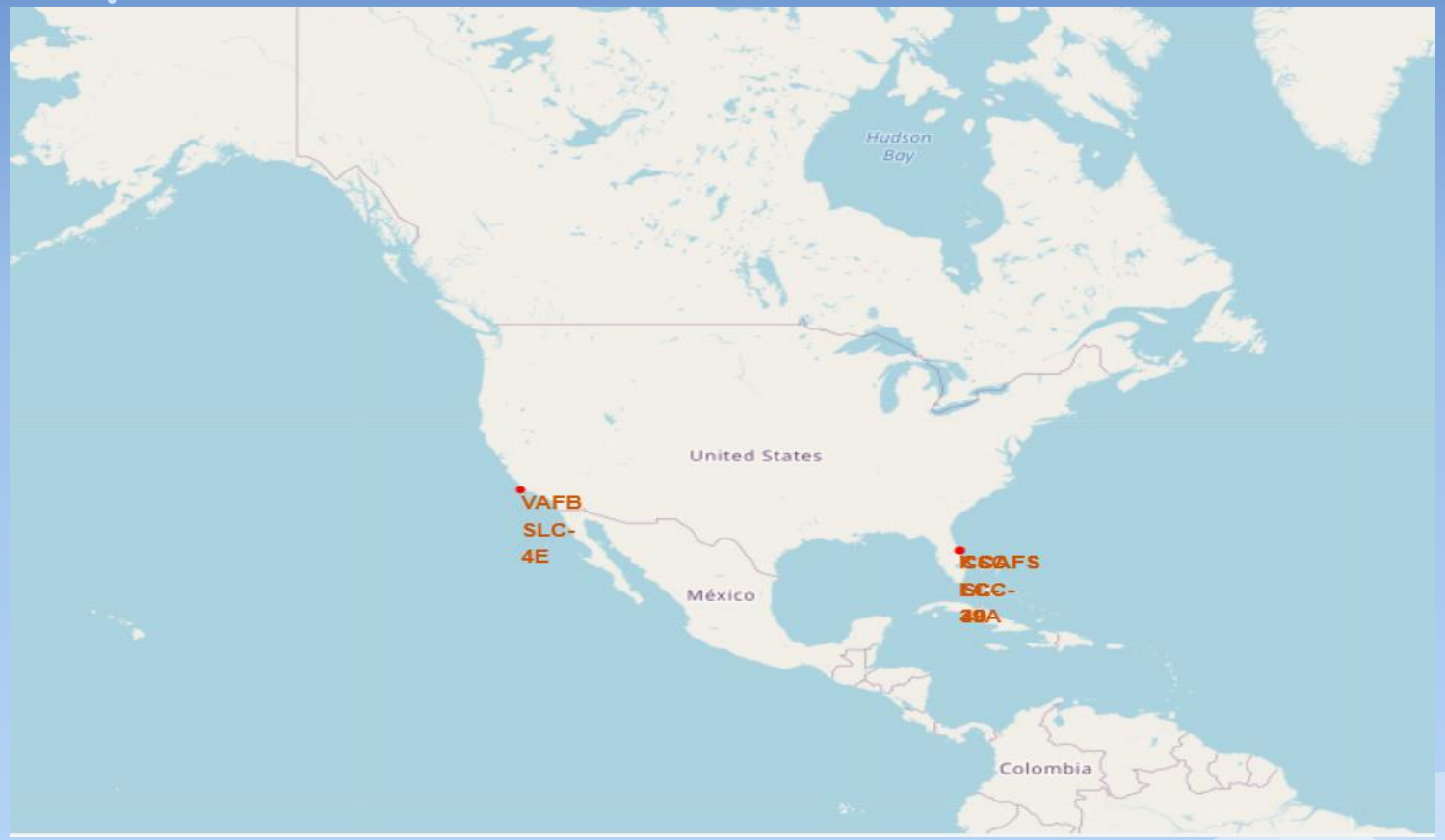
DATE	COUNT
2010-06-04	1
2010-12-08	1
2012-05-22	1
2012-10-08	1
2013-03-01	1
2013-09-29	1
2013-12-03	1
2014-01-06	1
2014-04-18	1
2014-07-14	1
2014-08-05	1
2014-09-07	1
2014-09-21	1
2015-01-10	1
2015-02-11	1
2015-03-02	1
2015-04-14	1
2015-04-27	1
2015-06-28	1
2015-12-22	1
2016-01-17	1
2016-03-04	1
2016-04-08	1
2016-05-06	1
2016-05-27	1
2016-06-15	1
2016-07-18	1
2016-08-14	1
2017-01-14	1
2017-02-19	1
2017-03-16	1

- COUNT is used find the number of values in a column in this case landing_outcome being the column. GROUP BY is used to group by column DATE ,Having is used to add the condition in GROUP BY clause and DESC is used to arrange the data in descending order using count as column.

Section 3

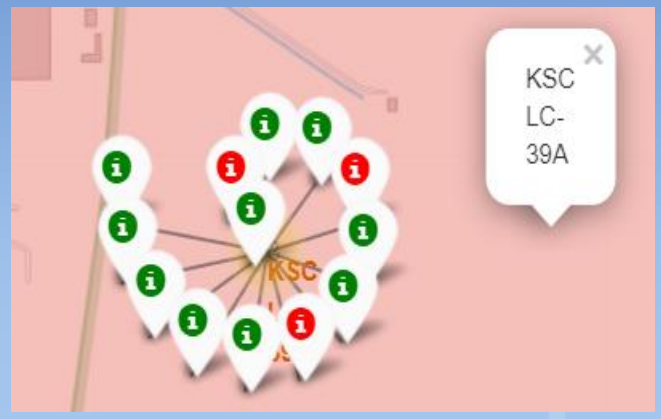
Launch Sites Proximities Analysis

Map Showing All the Launch Sites



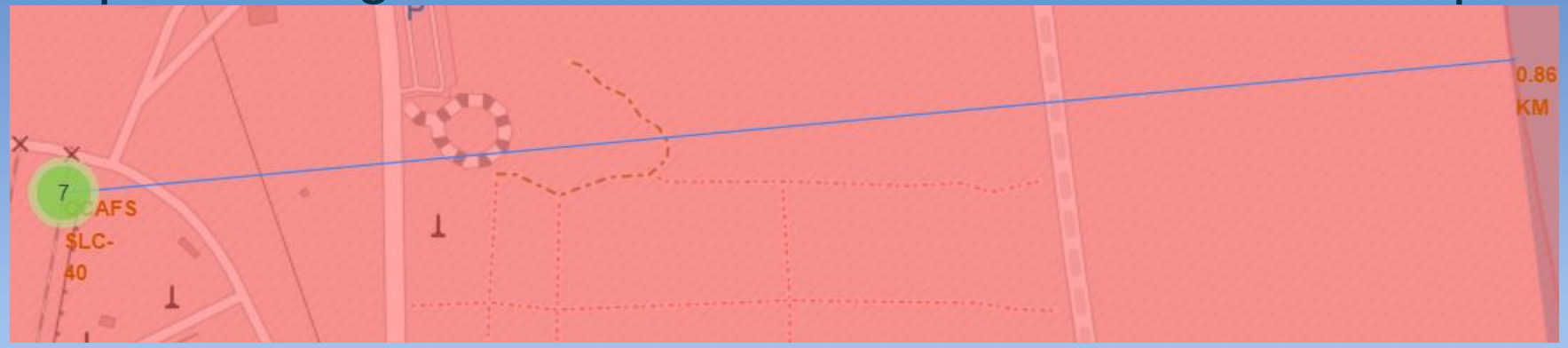
- The folium map shows 3 launch sites (CCAFS LC-40, CCAFS SLC-40, KSC LC-39A) on the east coast and one (VAFB SLC- 4E) on the west coast .

Maps Showing Successful and Failed Launches for Each Site



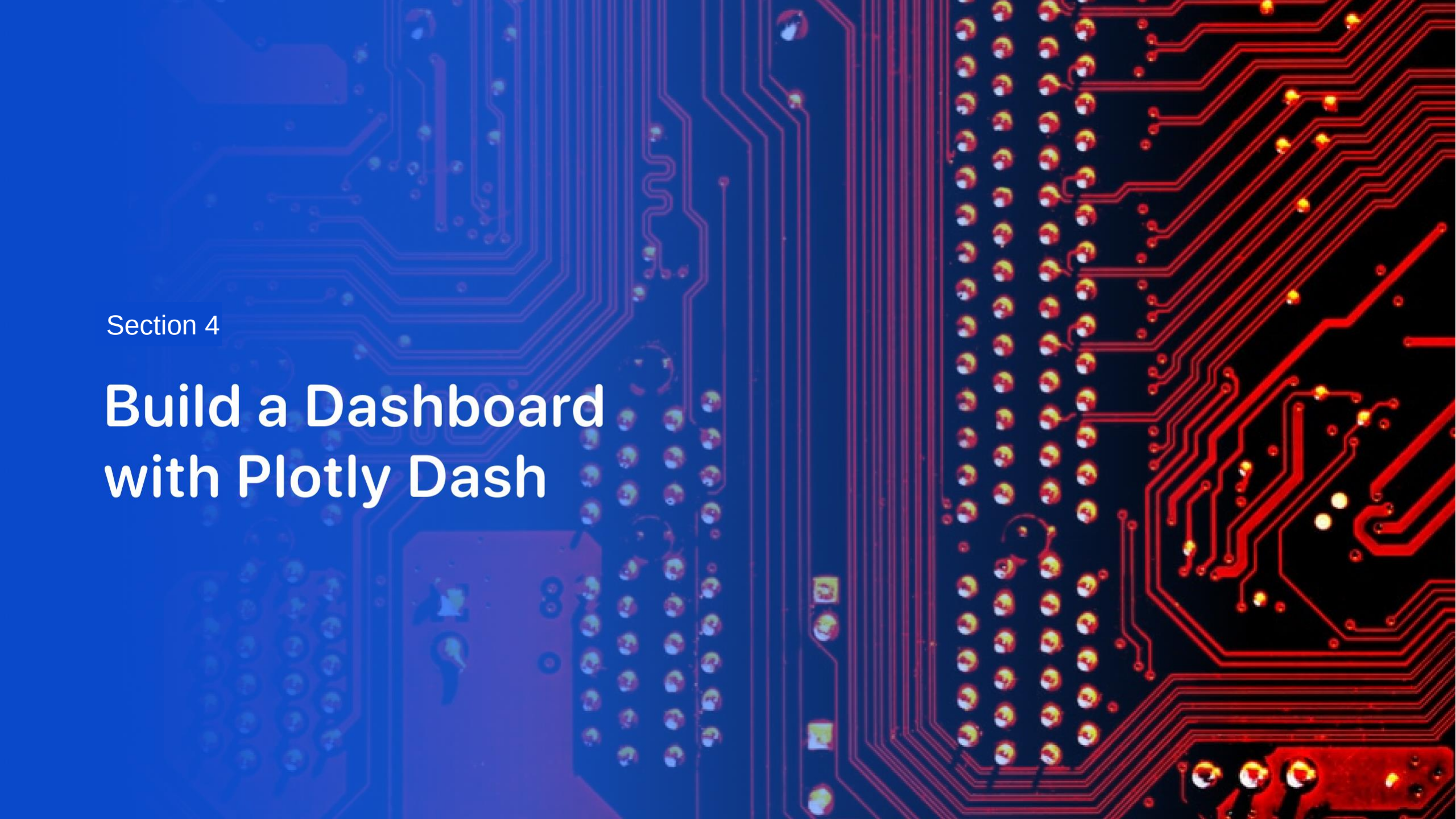
- The above maps tell us the number of launches like VAFB SLC 4E has 10 launches and 4 out of 10 were successful
- CCAFS LC has 26 launches and CCAFS SLC-40 has 7 launches
- KSC LC-39 A has 13 launches and 10 out of 13 were successful.

Map Showing Distance between launch site and its proximities

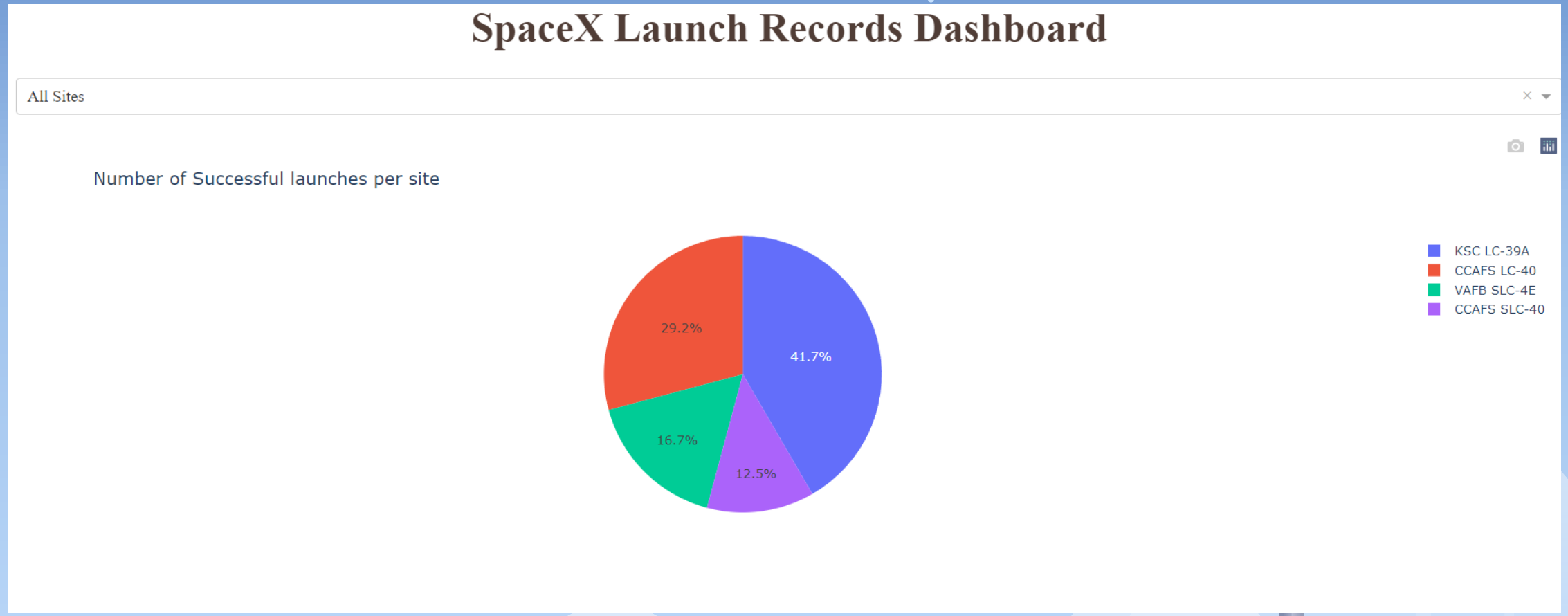


Map Showing Distance between launch site and its proximities

- The first folium map shows the distance between CCAFS SLC-40 launch site and its nearest coastline to be 0.86 KM.
- The second folium map shows the distance between VAFB SLC- 4E launch site and its nearest coastline to be 1.35 KM and to nearest railroad to be 1.27 KM .
- The third folium map shows the distance between VAFB SLC- 4E launch site and its nearest city (Lompoc) to be 14.00 KM.

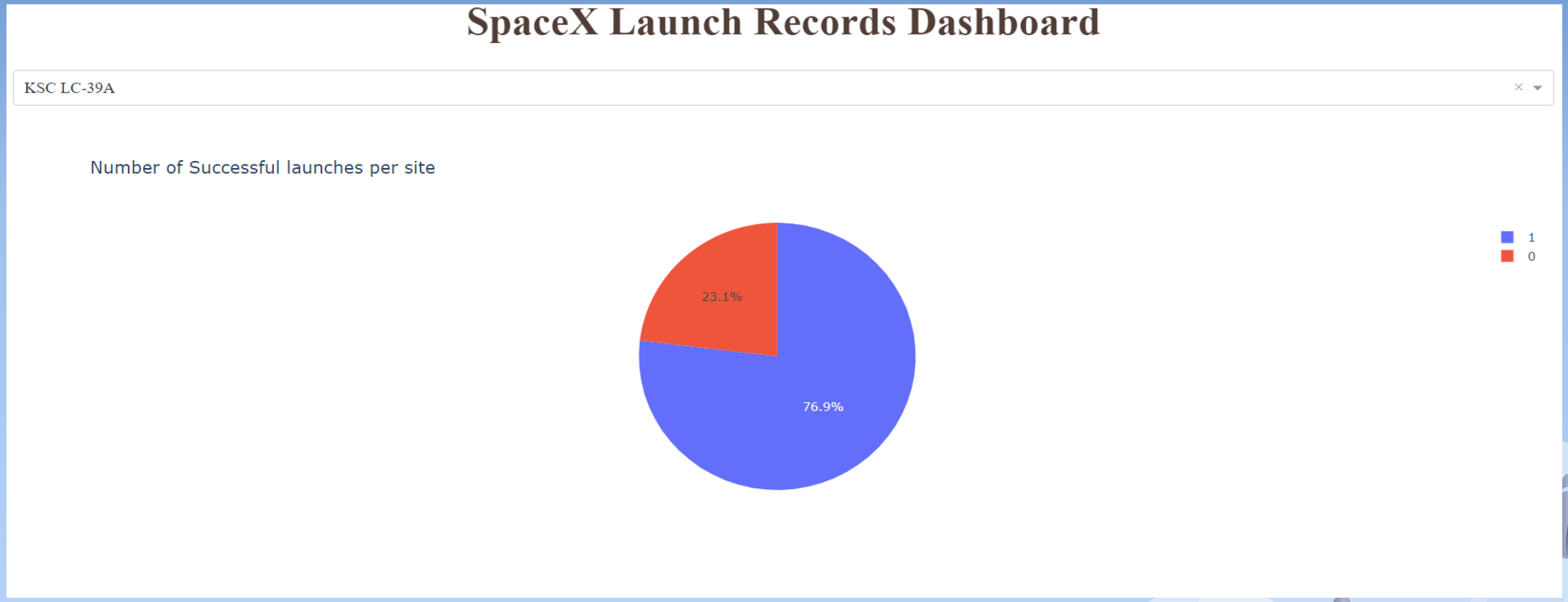

Section 4
Build a Dashboard with Plotly Dash

Pie Chart Showing Successful Launches per Site



- The above pie chart shows that KSC LC-39A and CCAFS LC-40 account for the most number of successful launches i.e they are the most suitable for launches.

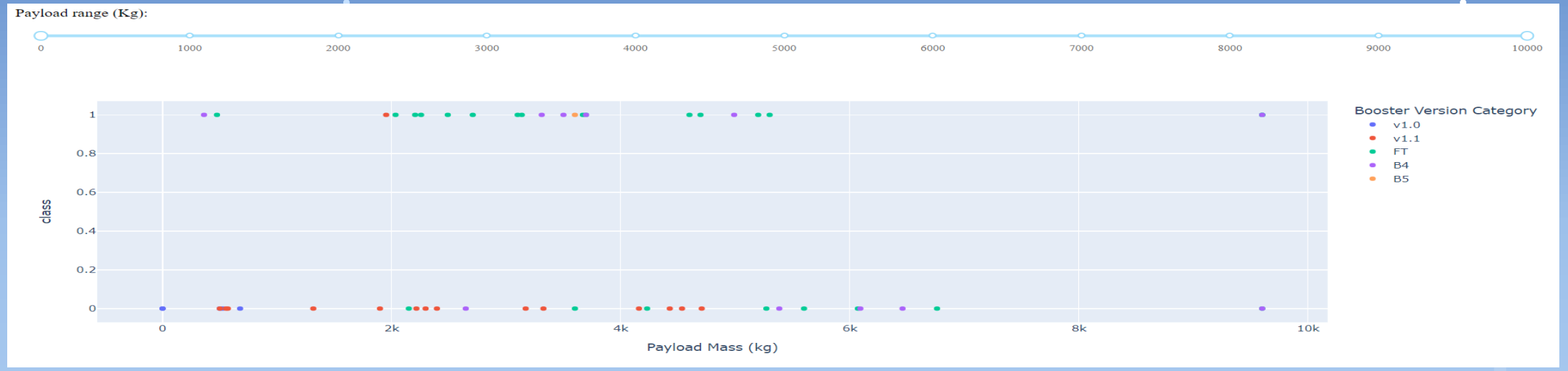
Pie Chart Showing Successful Launches at KSC LC-39A



- The above pie chart shows that ~77% of launches at KSC LC-39A were successful.

.

Scatter plot showing Successful launches vs Payload Mass



Scatter plot showing Successful launches vs Payload Mass

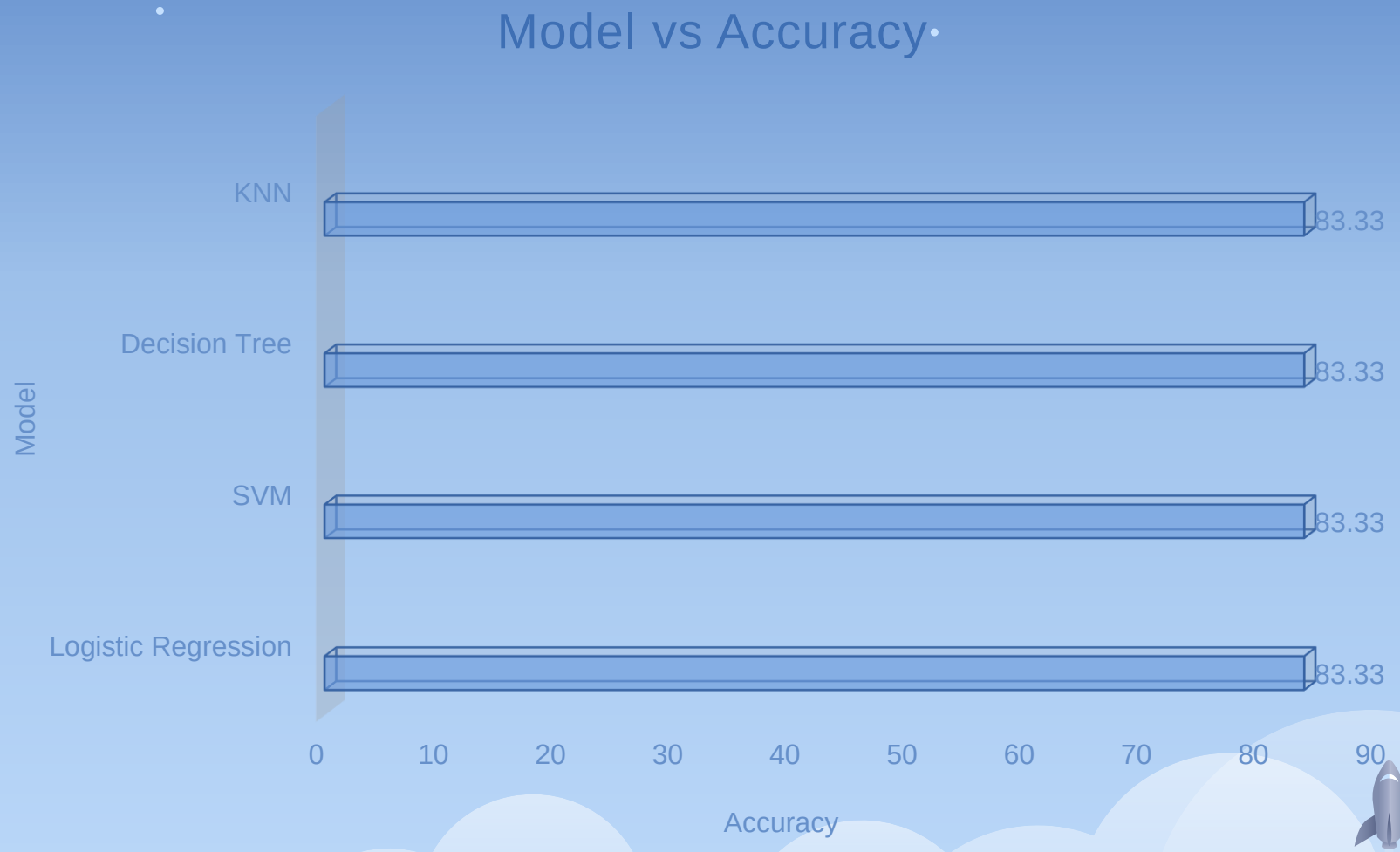
- The first scatter plot shows that in the range 0 to 10000 ,the FT booster version is the best choice while v1.1 tends to perform the worst.

- The second scatter plot shows that in the range 3000 to 4000 ,the B5 booster version is the best choice with no failed launches while v1.1 tends to perform the worst.

Section 5

Predictive Analysis (Classification)

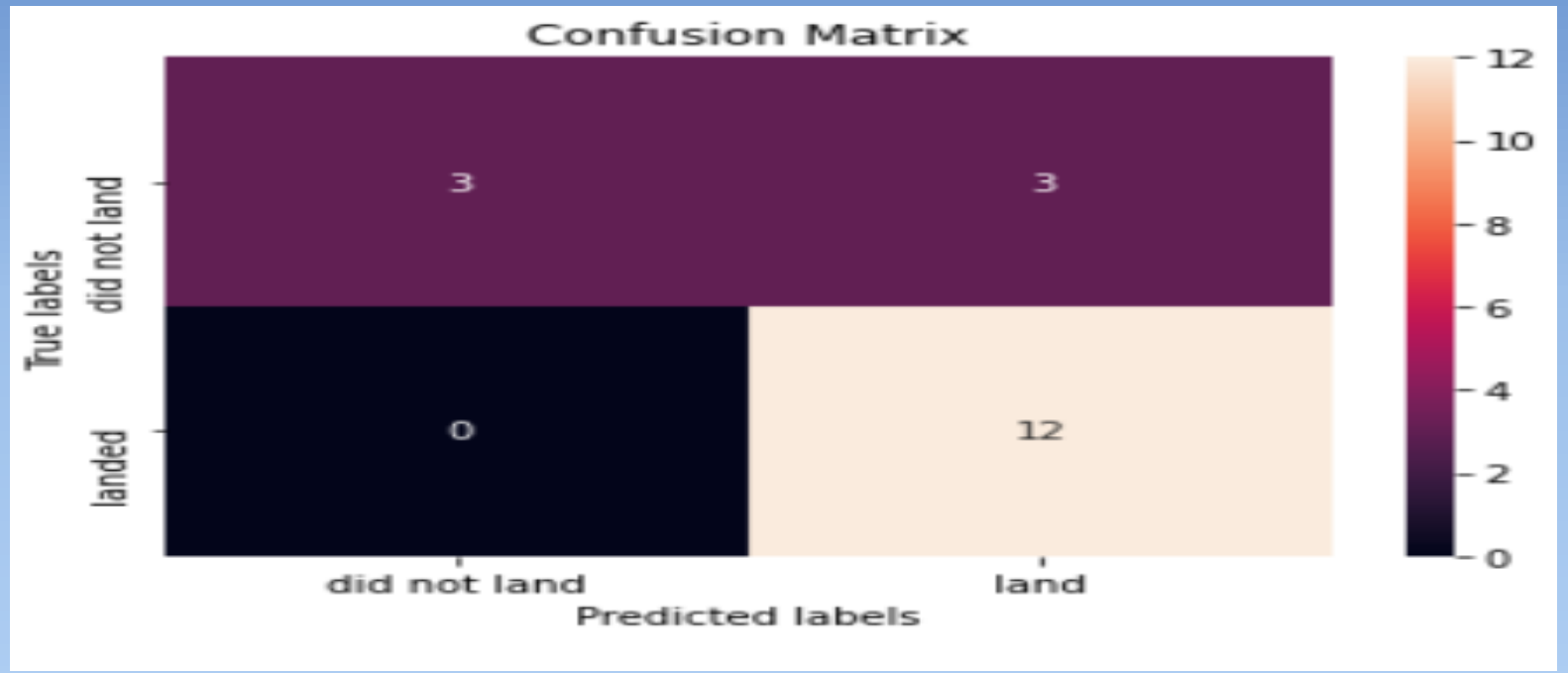
Classification Accuracy



- The above bar chart shows that all the models trained have accuracy of ~84%.

.

Confusion Matrix



- The confusion matrix of KNN has 3 FP (false positive) which means that they were classified as “land” even though “did not land” and 3 TN (true negative) which means that they were classified as “did not land” when they were “did not land”. 0 FN (false negative) which means all the values in “land” were classified correctly as “land”.

Conclusions

- We trained a KNN model with ~84%,this was found to be the performing model which was then used to predict the success rate of the launches.
- We observed that FT Booster version performs best in the range 0 to 10000 KG payload.
- We observed that KSC LC-39A and CCAFS LC-40 account for the most number of successful launches i.e they are the most suitable for launches.
- We observed that he success rate of ES-L1,GEO ,HEO ,SSO and VLEO appears to be significantly more than others indicating that they are more suitable for launches.

Appendix

- GitHub repo Link
- Template Design Credits

Thank you!