Winning Space Race with Data Science

Lazar Danilovic
12/21/2021

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection
  - Data wrangling
  - EDA with data visualization
  - EDA with SQL
  - Building a Dashboard with Plotly Dash
  - Predictive analysis / Classification

- Summary of all results
  - Exploratory data analysis results
  - Ineractive analytics demo in screenshots
  - Predictive analysis results

Introduction

- Project background and context

  •The context and final goal of this project is the prediction of the Falcon 9(first stage) landing outcome, which can be correlated to the cost of a launch. The advertised cost of 'Falcon 9' services is 62 million dollars, almost half the price of competitors. The biggest advantage and 'low' price of SpaceX - Falcon 9 program coming from the reusability of first stage rockets.

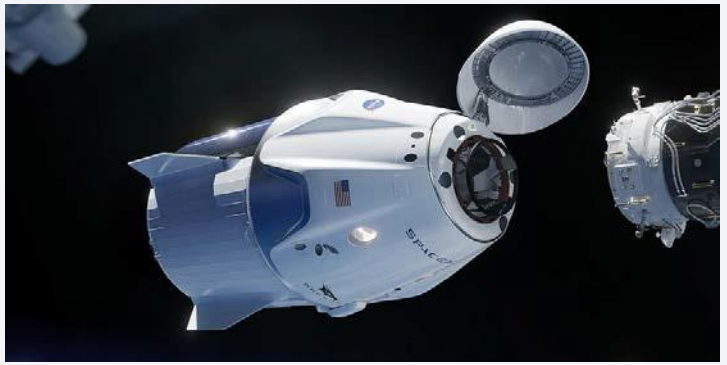
- Problems you want to find answers
  - What variables can affect landing outcome (successful or not)
  - Impact and effect of each variable on successful landing
  - Determination of best possible conditions (variables wise) for successful landing outcome

Section 1

Methodology

Methodology

Executive Summary



- Data collection methodology:
  - SpaceX Rest API
  - Web scrapping (www.Wikipedia.com)
- Perform data wrangling
  - One Hot Encoding, data preprocessing and conditioning for ML process
- 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 – SpaceX API

SpacX REST API

API returns
SpaceX data in .JSON format

Normalize data into flat
data file such (.csv)

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

response = requests.get(spacex_url)

data = pd.json_normalize(response.json())
```

```
#Global variables
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

```
df = df[df['BoosterVersion']!='Falcon 1']

df['PayloadMass'].replace(np.nan, df['PayloadMass'].mean(), inplace=True)

df.to_csv('dataset_part_1.csv', index=False)
```

GitHub Link: Data_Collection_API

Data Collection – WEB Scraping

Get HTML response
from ‘https://wikipedia’

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

response = requests.get(static_url)
```

Extract data using
‘BeautifulSoup’ module

```
soup = BeautifulSoup(response.text)

html_tables = soup.find_all('table')
```

Normalize data into flat
data file such (.csv)

```
launch_dict['Flight No.'] = []
launch_dict['Launch si
launch_dict['Payload']
launch_dict['Payload m
launch_dict['Orbit'] =
launch_dict['Customer'
launch_dict['Launch ou
# Added some new colum
launch_dict['Version B
launch_dict['Booster l
launch_dict['Date']=[]
launch_dict['Time']=[]
```

```
    flag=False
#get table element
row=rows.find_all('td')
#if it is number save cells in a dictonary
if flag:
    extracted_row += 1
    # Flight Number value
    # TODO: Append the flight_number into launch_dict with key
    #print(flight_number)
    launch_dict['Flight No.'].append(flight_number)
    datatimelist=date_time(row[0])

    # Date value
    # TODO: Append the date into launch_dict with key `Date`
    date = datatimelist[0].strip(',')
    launch_dict['Date'].append(date)
    #print(date)
```

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

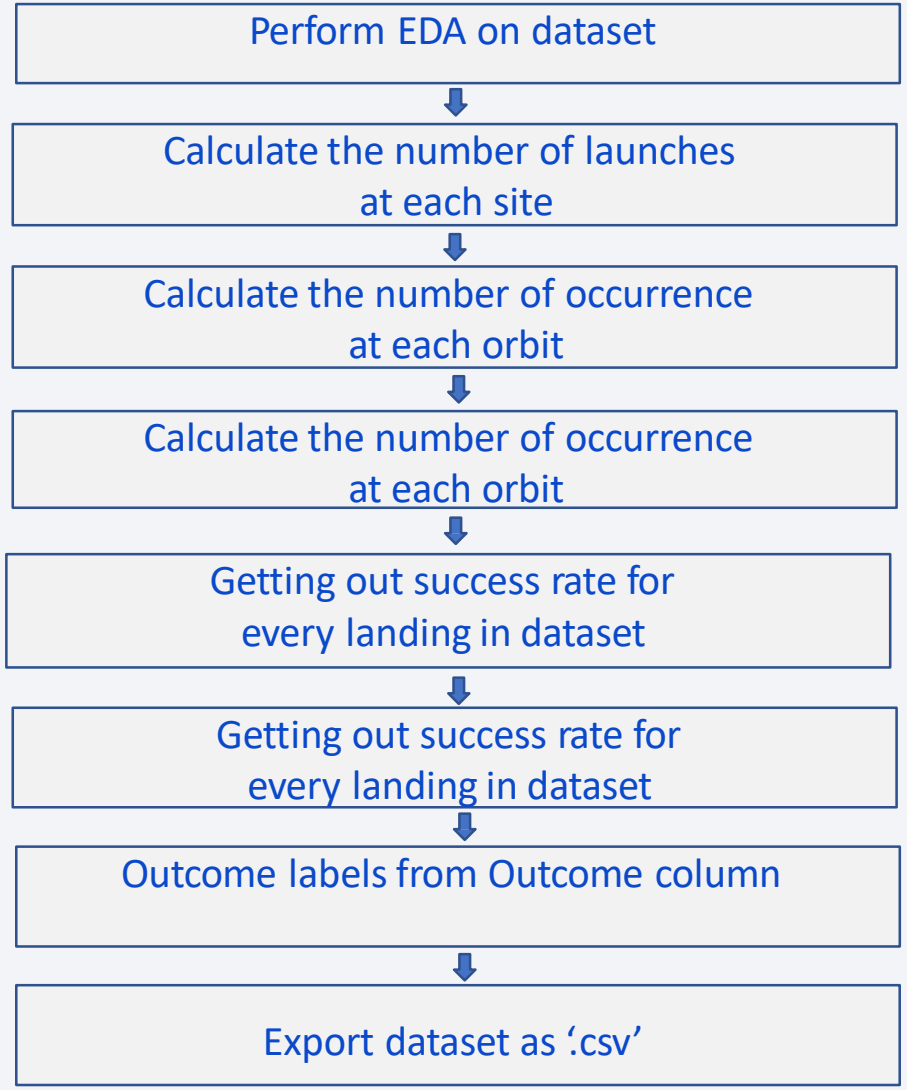
GitHub Link: Data Collection WEB Scraping

Data Wrangling

- Describe how data were processed

In the data set, there are several different cases where the booster did not land successfully. Sometimes a landing was attempted but failed due to an accident; for example, True Ocean means the mission outcome was successfully landed to a specific region of the ocean while False Ocean means the mission outcome was unsuccessfully landed to a specific region of the ocean. True RTLS means the mission outcome was successfully landed to a ground pad False RTLS means the mission outcome was unsuccessfully landed to a ground pad. True ASDS means the mission outcome was successfully landed on a drone ship False ASDS means the mission outcome was unsuccessfully landed on a drone ship. We mainly convert those outcomes into Training Labels with 1 means the booster successfully landed 0 means it was unsuccessful.

GitHub Link: Data Collection_Data Wrangling



EDA with Data Visualization

- Scatter Graphs being drawn
  - Flight number vs. Payload Mass
  - Flight number vs. Launch Site
  - Payload Mass vs. Launch Site
  - Orbit vs. Flight number
  - Payload Mass vs. Orbit Type
  - Orbit vs. Payload Mass
- Bar Graph Graphs being drawn
  - Success (mean) vs. Orbit
- Line Graph Graphs being drawn
  - Success rate per Year

GitHub Link: EDA with Data Wrangling

EDA with SQL

- SQL queries to gather information about the dataset.

  - Displaying the names of the unique '***launch sites***' in the space mission
  - Displaying 5 records where '***launch sites***' begin with the string 'KSC'
  - Displaying the total '***payload mass***' carried by boosters launched by NASA (CRS)
  - Displaying average '***payload mass***' carried by booster version F9 v1.1
  - Listing the date where the successful '***landing outcome***' in drone ship was achieved.
  - Listing the names of the boosters which have success in ground pad and have '***payload mass***' greater than 4000 but less than 6000
  - Listing the total number of successful and failure mission outcomes
  - Listing the names of the '***booster versions***' which have carried the maximum '***payload mass'*** versions, '***launch site***' for the months in year 2017
  - Ranking the count of successful '***landing outcomes***' between the date 2010-06-04 and 2017-03-20 in descending order.

GitHub Link: EDA with SQL

Build an Interactive Map with Folium

To visualize the Launch Data into an interactive map. We took the Latitude and Longitude Coordinates at each launch site and added a Circle Marker around each launch site with a label of the name of the launch site.

We assigned the dataframe launch_outcomes(failures, successes) to classes 0 and 1 with **Green** and **Red** markers on the map in a MarkerCluster()

Using Haversine's formula we calculated the distance from the Launch Site to various landmarks to find various trends about what is around the Launch Site to measure patterns. Lines are drawn on the map to measure distance to landmarks

GitHub Link: Launch_Site_FOLIUM

Build a Dashboard with Plotly Dash

Graphs- Pie Chart showing the total launches by a certain site/all:

- display relative proportions of multiple classes of data.
- size of the circle can be made proportional to the total quantity it represents.

Scatter Graph showing the relationship with Outcome and Payload Mass (Kg) for the different Booster Versions:

- It shows the relationship between two variables.
- It is the best method to show you a non-linear pattern.
- The range of data flow, i.e. maximum and minimum value, can be determined.
- Observation and reading are straightforward.

GitHub Link: Data with Plotly Dash

Predictive Analysis (Classification)

- Summarize how you built, evaluated, improved, and found the best performing classification model
- You need present your model development process using key phrases and flowchart
- Add the GitHub URL of your completed predictive analysis lab, as an external reference and peer-review purpose

GitHub Link: Classification_ML

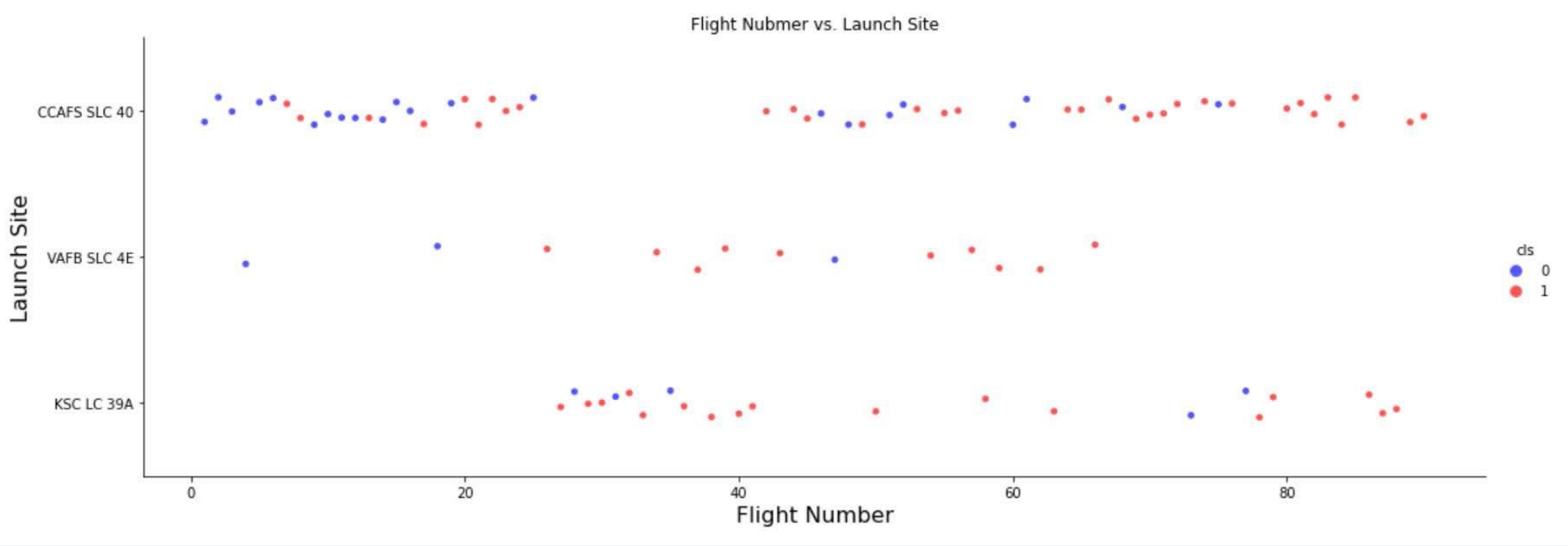
Results

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

Section 2

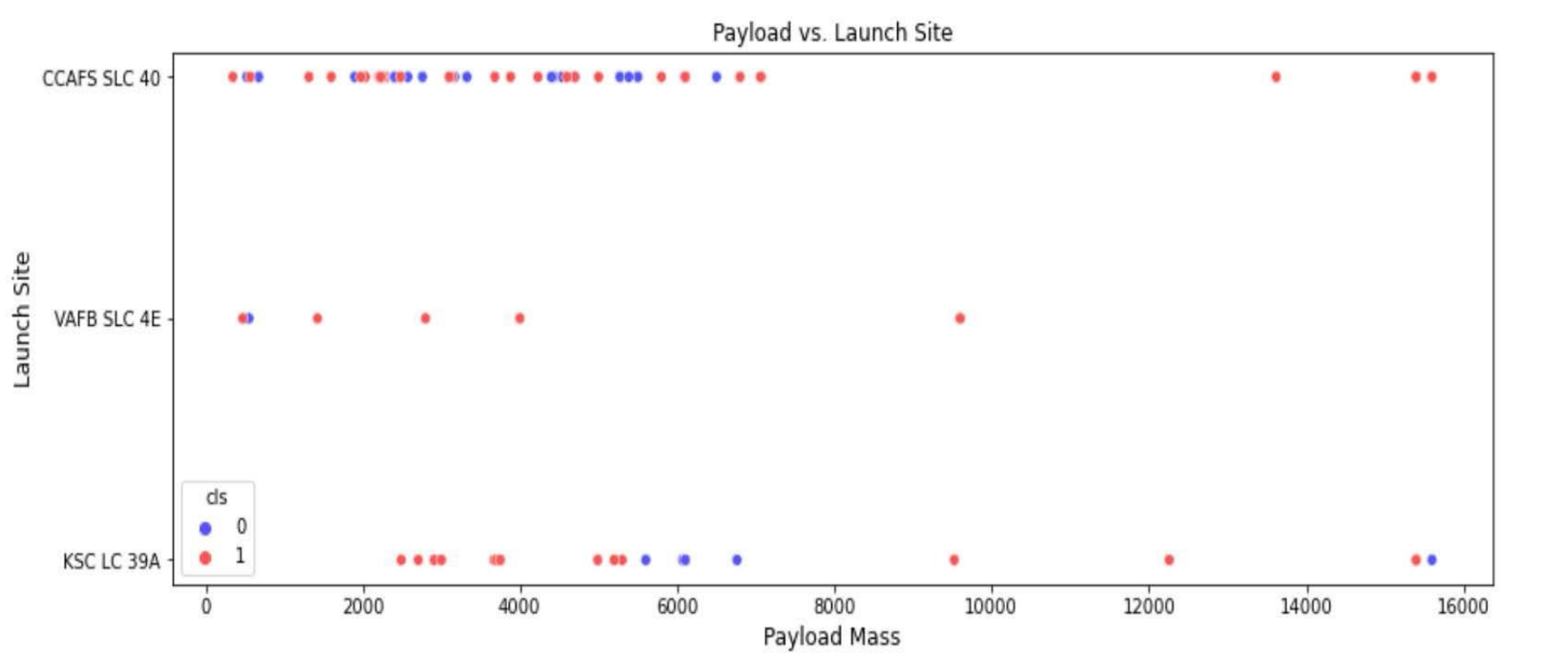
Insights drawn from EDA

Flight Number vs. Launch Site



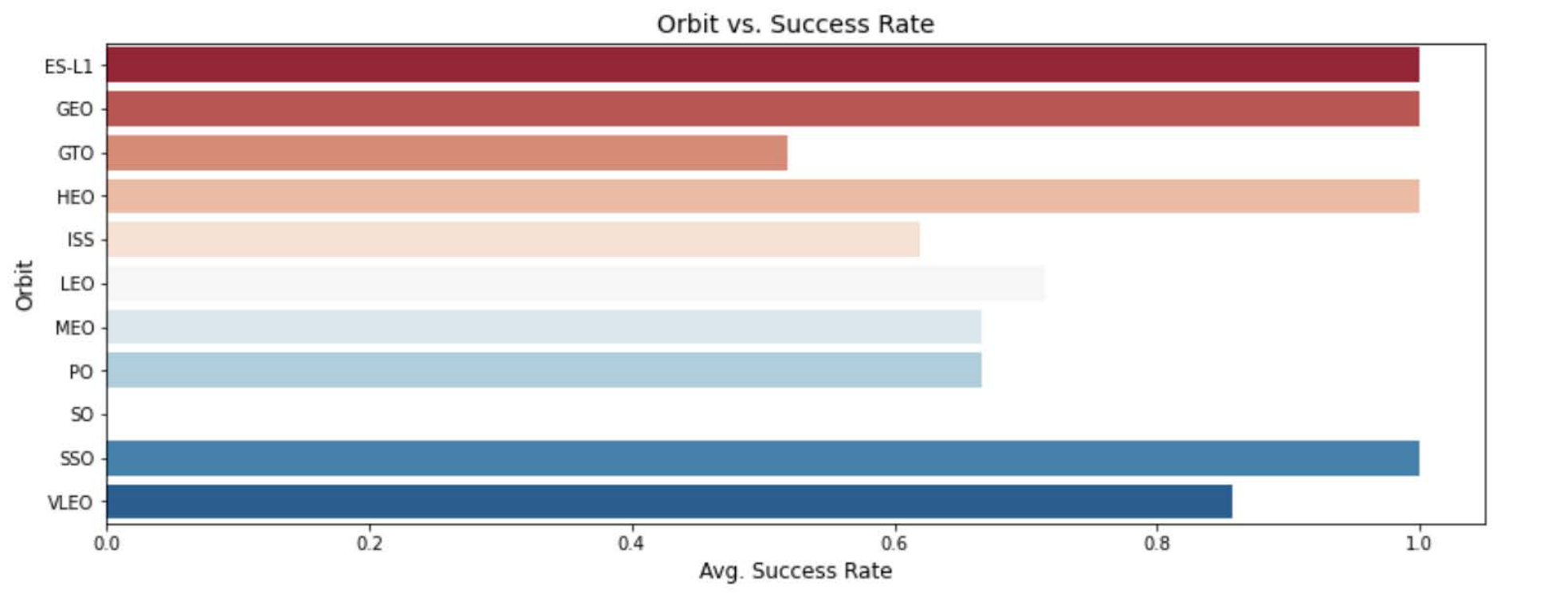
The more amount of flights at a launch site the greater the success rate at a launch site.

Payload vs. Launch Site



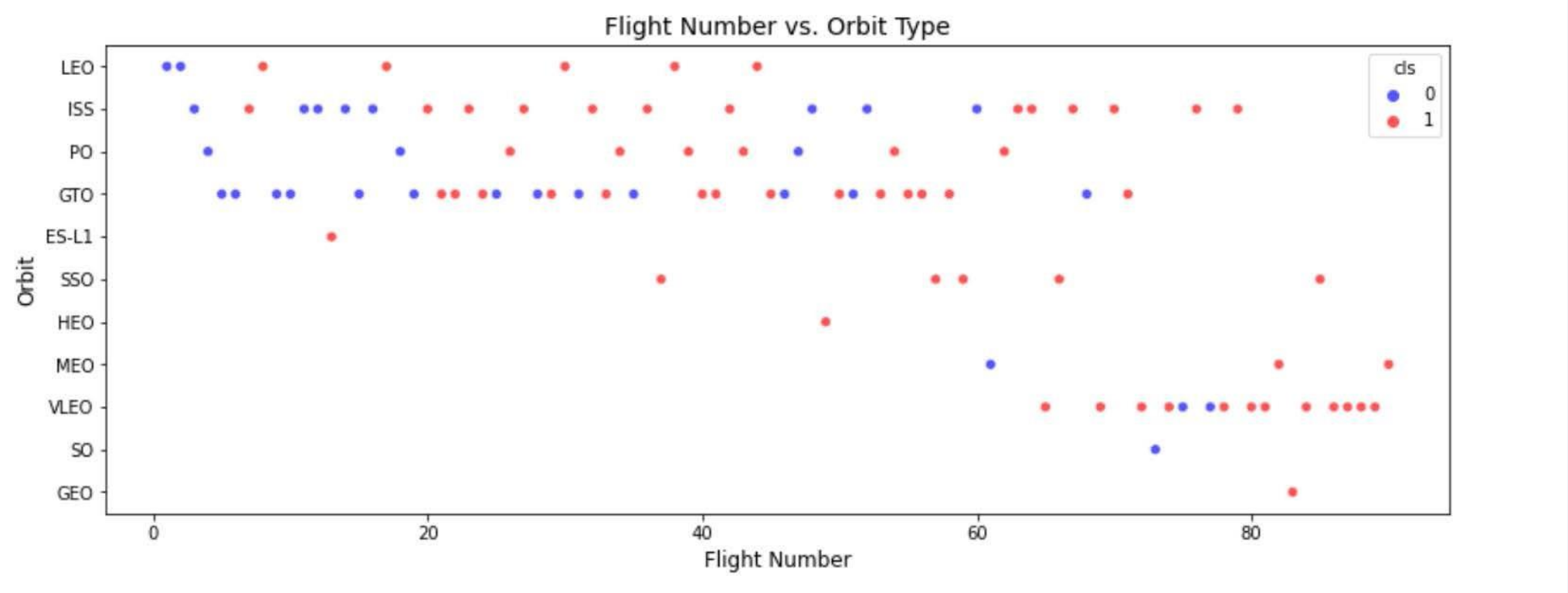
load mass for Launch Site CCAFS SLC 40 the higher the success rate for the Rocket. There is not quite a clear pattern to be found using this visualization to make a decision if the Launch Site is dependant on Pay Load Mass for a success launch.

Success Rate vs. Orbit Type



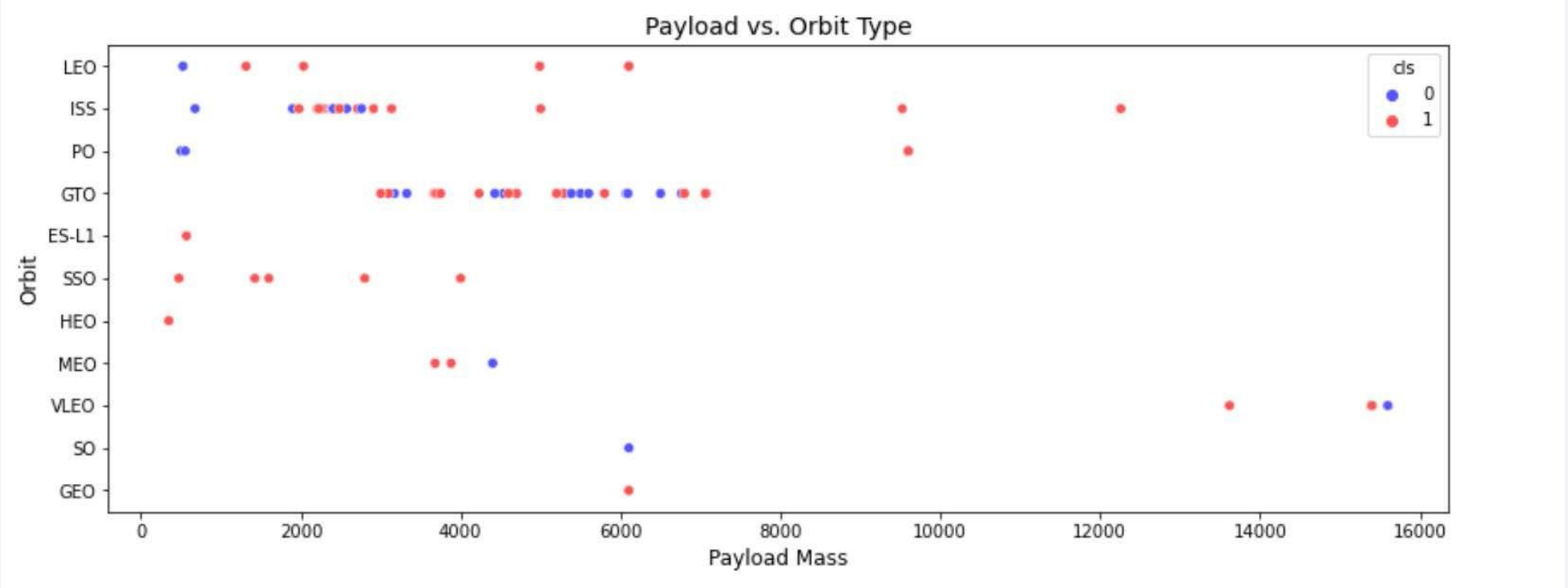
Orbit GEO,HEO,SSO,ES-L1 has the best Success Rate

Flight Number vs. Orbit Type



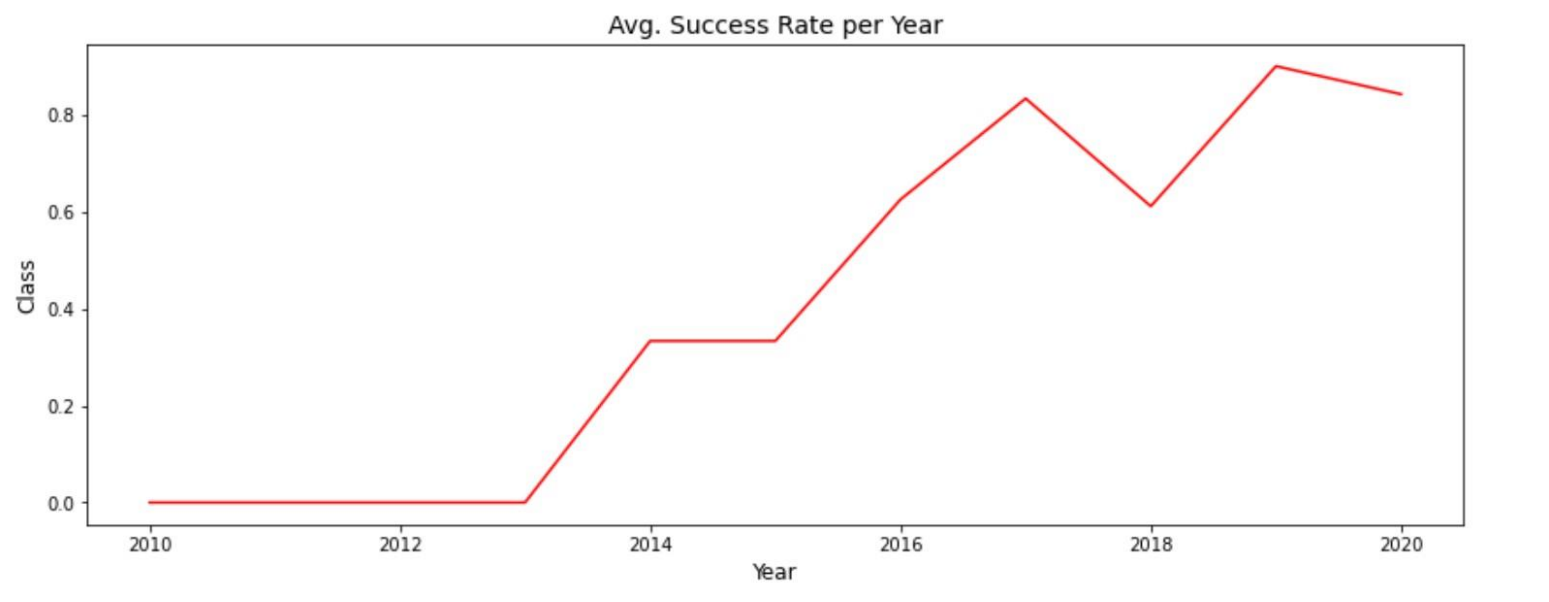
You should see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

Payload vs. Orbit Type

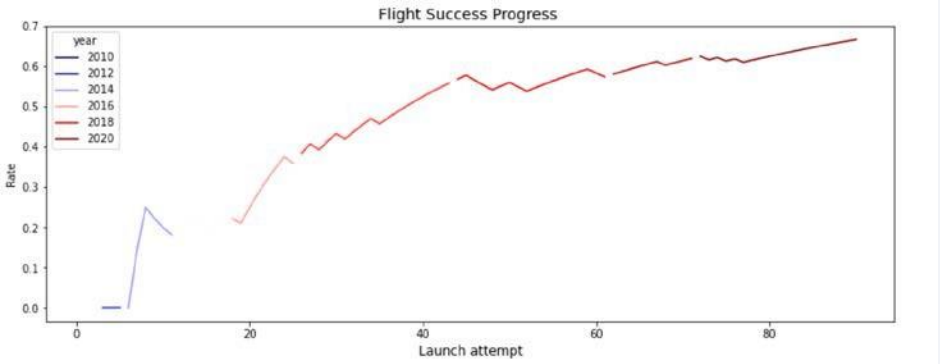


Heavy payloads have a negative influence on GTO orbits and positive on GTO and Polar LEO (ISS) orbits.

Launch Success Yearly Trend



We can observe that the success rate (since 2013) kept increasing in general till 2020. There was period (2018) with couple failures. On second (smaller) graph we presented success rate with each attempt (flight number), and show positive trend (getting higher) with each flight attempt.



All Launch Site Names

```
%sql select distinct launch_site from spcx
```

```
 * ibm_db_sa://dpx42411:***@dashdb-txn-sbox-yp-dal09-11.services.dal.bluemix.net:50000/BLUDB
Done.
```

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

QUERY EXPLAINATION: Using the word DISTINCT in the query means that it will only show Unique values in the 'launch_site' column from spcx

Launch Site Names Begin with 'CCA'

```
%sql select * from spcx where launch_site like 'CCA%' limit 5
```

```
 * ibm_db_sa://dpx42411:***@dashdb-txn-sbox-yp-dal09-11.services.dal.bluemix.net:50000/BLUDB
Done.
```

DATE	TIME	booster_version	launch_site	payload	payload_mass_kg	orbit	customer	mission_outcome	landing_outcome
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-08-12	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
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

QUERY EXPLAINATION: Using the 'limit' to show 5 records from 'spcx' and LIKE keyword has a wild card with the words 'CCA%' the percentage in the end suggests that the Launch_Site name must start with 'CCA'.

Total Payload Mass

```
%sql select sum(payload_mass_kg) as total from spcx where customer = 'NASA (CRS)'
```

```
 * ibm_db_sa://dpx42411:***@dashdb-txn-sbox-yp-dal09-11.services.dal.bluemix.net:50000/BLUDB
Done.
```

total
45596

QUERY EXPLAINATION: Using the function 'SUM' summates the total in the column 'payload_mass_kg'. The 'WHERE' clause filters the dataset to only perform calculations on Customer NASA (CRS)

Average Payload Mass by F9 v1.1

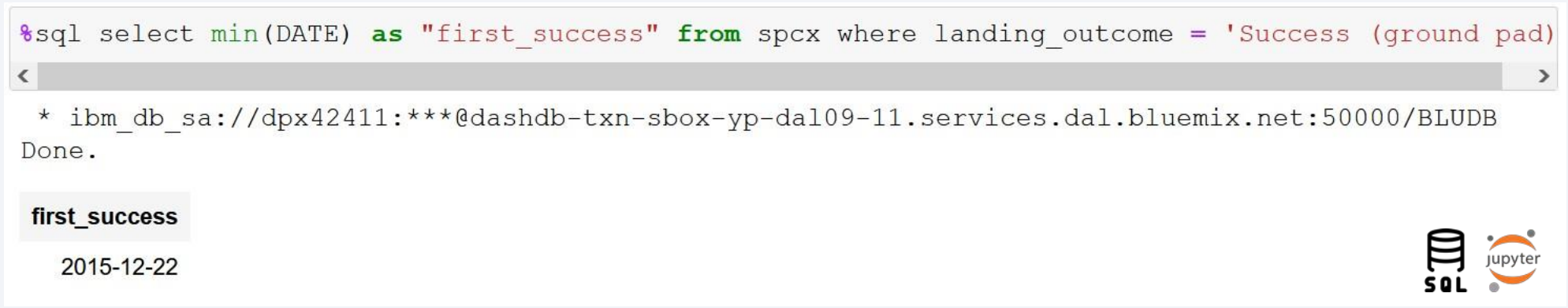
```
%sql select avg(payload_mass_kg) as average from spcx where booster_version ='F9 v1.1'
```

```
 * ibm_db_sa://dpx42411:***@dashdb-txn-sbox-yp-dal09-11.services.dal.bluemix.net:50000/BLUDB
Done.
```

average
2928.400000

QUERY EXPLAINATION: Using the function 'AVG' works out the average in the column 'payload_mass_kg'. The 'WHERE' clause filters the dataset to only perform calculations on 'booster_version' F9 v1.1

First Successful Ground Landing Date



QUERY EXPLAINATION: Using the function ‘MIN’ works out the minimum date in the column ‘DATE’. The ‘WHERE’ clause filters the dataset to only perform calculations on ‘landing_outcome’ - Success (drone ship)

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%%sql select booster_version as succsesful_drone_ship_boosters from spcx
where landing_outcome = 'Success (drone ship)' and  payload_mass_kg between 4000 and 6000
```

```
 * ibm_db_sa://dpx42411:***@dashdb-txn-sbox-yp-dal09-11.services.dal.bluemix.net:50000/BLUDB
Done.
```

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

QUERY EXPLAINATION: Selecting only 'booster_version', the 'WHERE' clause filters the dataset to 'landing_outcome' - Success (drone ship). We use 'between' instead (> …<), give us more elegant line and readable.

Total Number of Successful and Failure Mission Outcomes

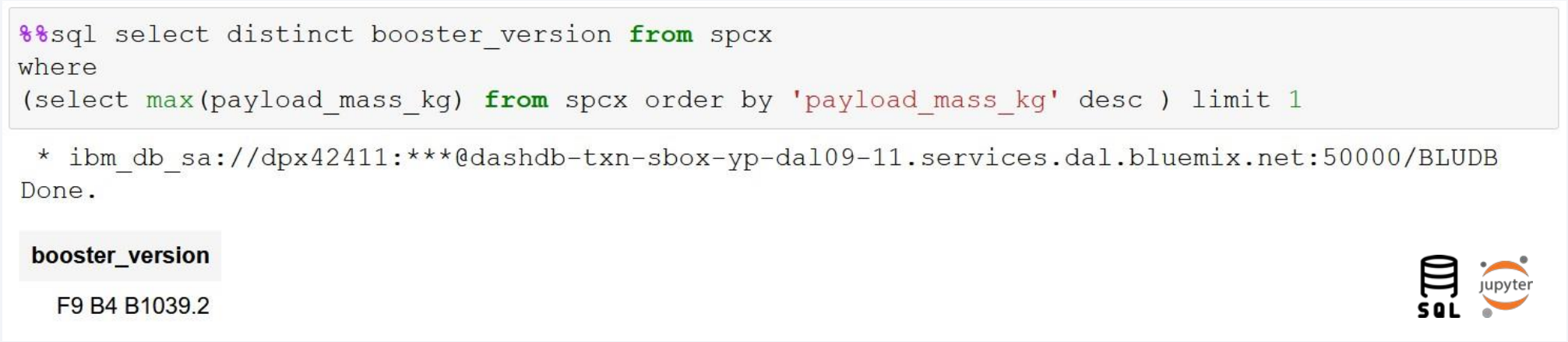
```
%%sql select count(*) as outcomes from spcx where mission_outcome like '%Success%'
union
select count(*) from spcx where mission_outcome like '%Failure%'
```

```
 * ibm_db_sa://dpx42411:***@dashdb-txn-sbox-yp-dal09-11.services.dal.bluemix.net:50000/BLUDB
Done.
```

outcomes
1
100

QUERY EXPLAINATION: instead using ‘subquery’ it’s easier to us ‘UNION’ with two line of selections.

Boosters Carried Maximum Payload



```
%%sql select distinct booster_version from spcx
where
(select max(payload_mass_kg) from spcx order by 'payload_mass_kg' desc ) limit 1
```

```
 * ibm_db_sa://dpx42411:***@dashdb-txn-sbox-yp-dal09-11.services.dal.bluemix.net:50000/BLUDB
Done.
```

booster_version
F9 B4 B1039.2

QUERY EXPLAINATION: Selecting unique values for 'booster_version'. The 'WHERE' is defined with 'subquery' > maximum of payload, descending order, selecting only 1 (first highest)

2015 Launch Records

```
%%sql select landing_outcome, booster_version, launch_site from spcx where landing_outcome like '%dron%'
and date between '2015-01-01' and '2015-12-31'
```

```
 * ibm_db_sa://dpx42411:***@dashdb-txn-sbox-yp-dal09-11.services.dal.bluemix.net:50000/BLUDB
Done.
```

landing_outcome	booster_version	launch_site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40
Precluded (drone ship)	F9 v1.1 B1018	CCAFS LC-40

QUERY EXPLAINATION: Selecting multiple columns > - Taking wild card for ‘landing_outcome’, ‘between’ was used to specify particular full year ‘2015’

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

```
%%sql select count(*) as outcome from spcx where landing_outcome = 'Failure (drone ship)'
union
select count(*) from spcx where landing_outcome = 'Success (ground pad)' and Date > '2010-06-04' and Date < '2017-03-20'
```

```
 * ibm_db_sa://dpx42411:***@dashdb-txn-sbox-yp-dal09-11.services.dal.bluemix.net:50000/BLUDB
Done.
```

outcome
5

```
t(*) as success from spcx where landing_outcome = 'Success (ground pad)' and Date > '2010-06-04' and Date < '2017-03-20'
```

```
 * ibm_db_sa://dpx42411:***@dashdb-txn-sbox-yp-dal09-11.services.dal.bluemix.net:50000/BLUDB
Done.
```

success
5

```
select count(*) as failur from spcx where landing_outcome = 'Failure (drone ship)' and Date > '2010-06-04' and Date < '2
```

```
 * ibm_db_sa://dpx42411:***@dashdb-txn-sbox-yp-dal09-11.services.dal.bluemix.net:50000/BLUDB
Done.
```

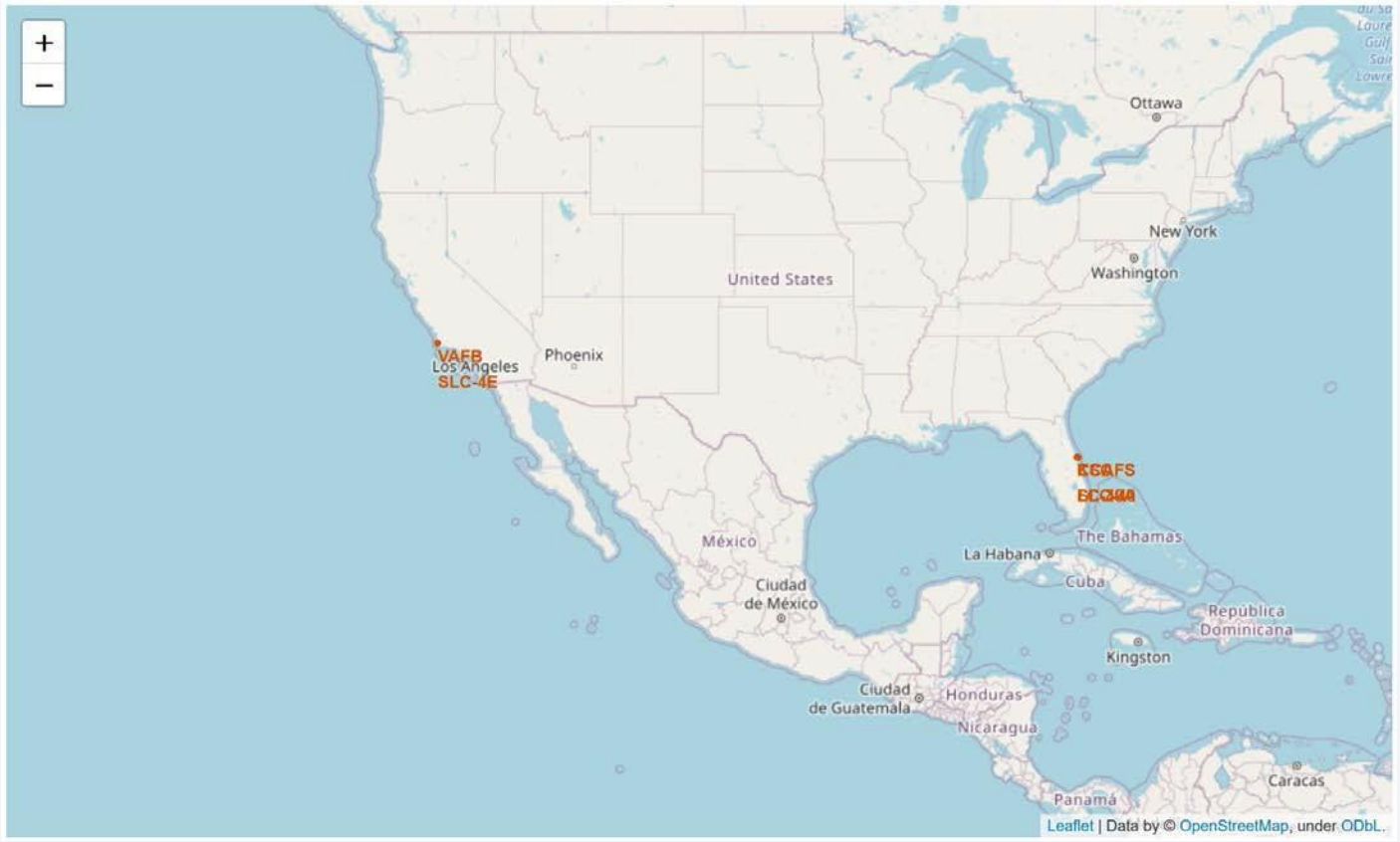
failur
5

QUERY EXPLAINATION: Function ‘COUNT’ counts records in column ‘WHERE’ filters data, the ‘WHERE’ is defined with: LIKE(wildcard) AND (conditions) AND (conditions)

Section 4

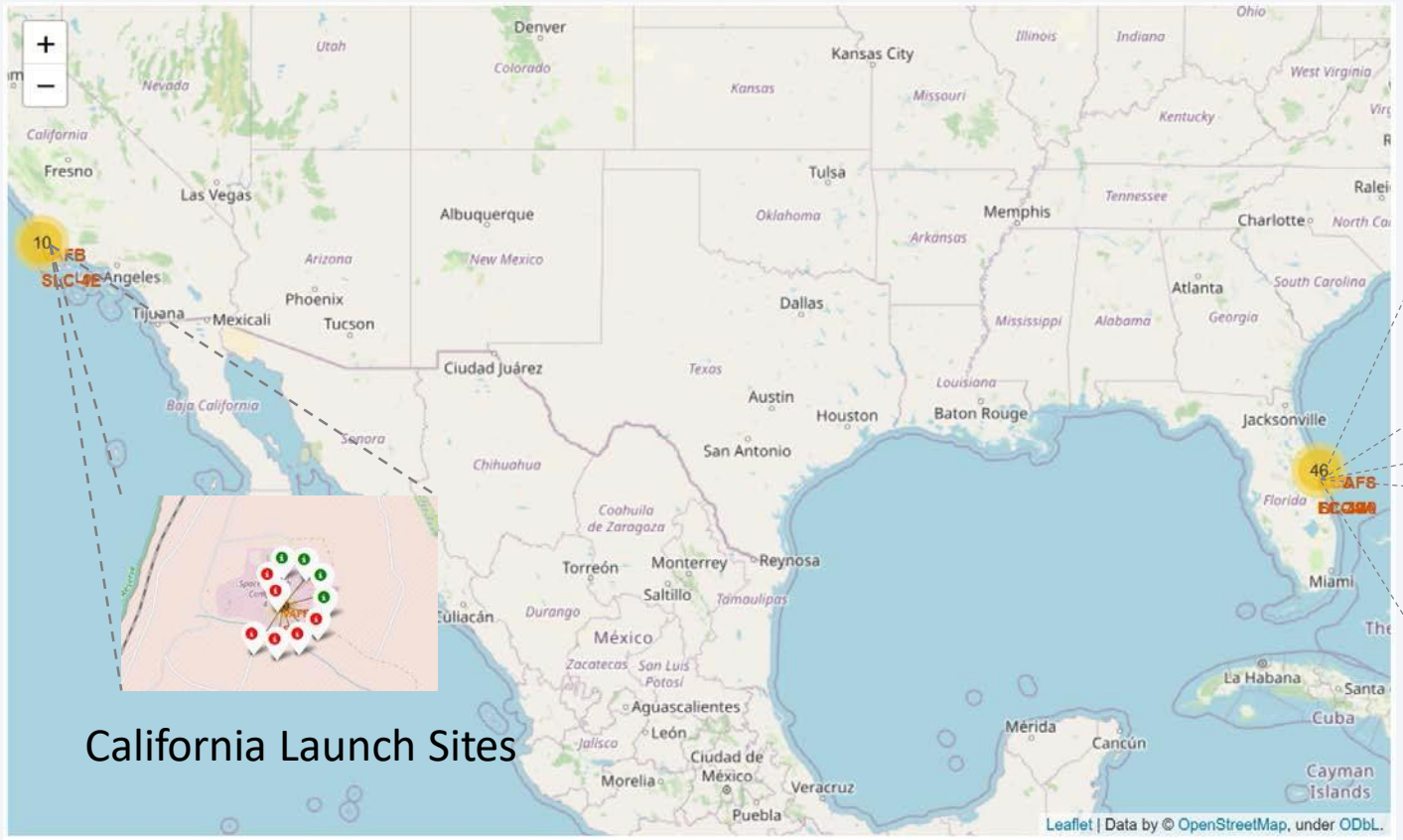
Launch Sites Proximities Analysis

Launch sites (USA) map markers

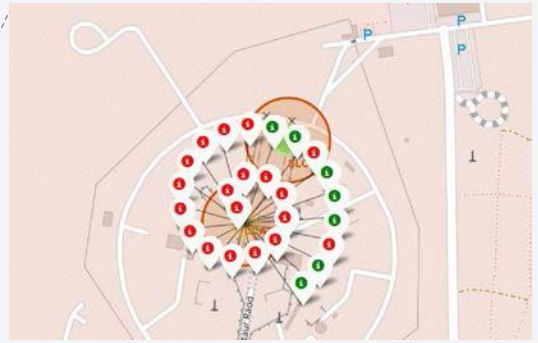


We can see that the SpaceX launch sites are in the United States of America coasts. Florida and California

Colour Labelled Markers



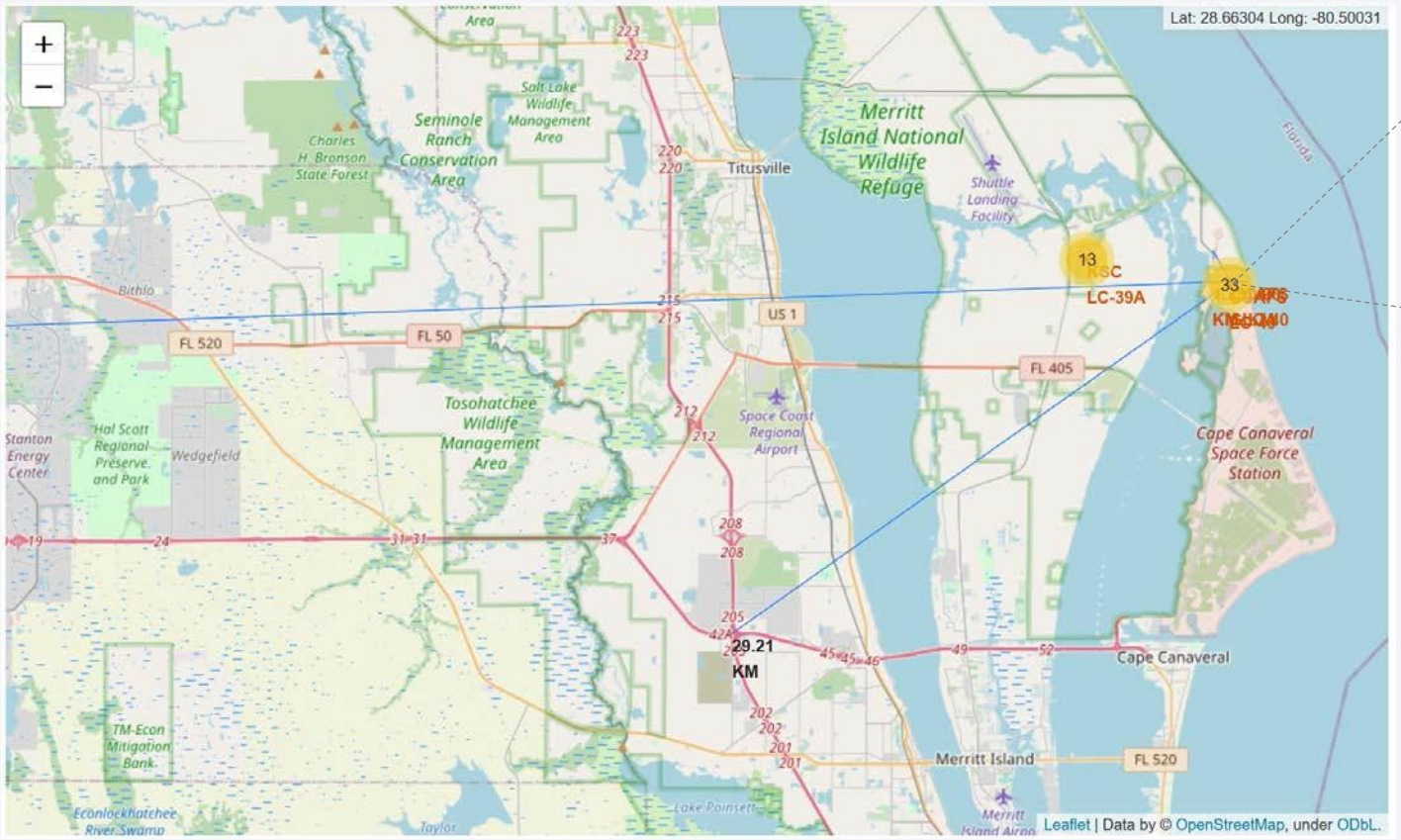
California Launch Sites



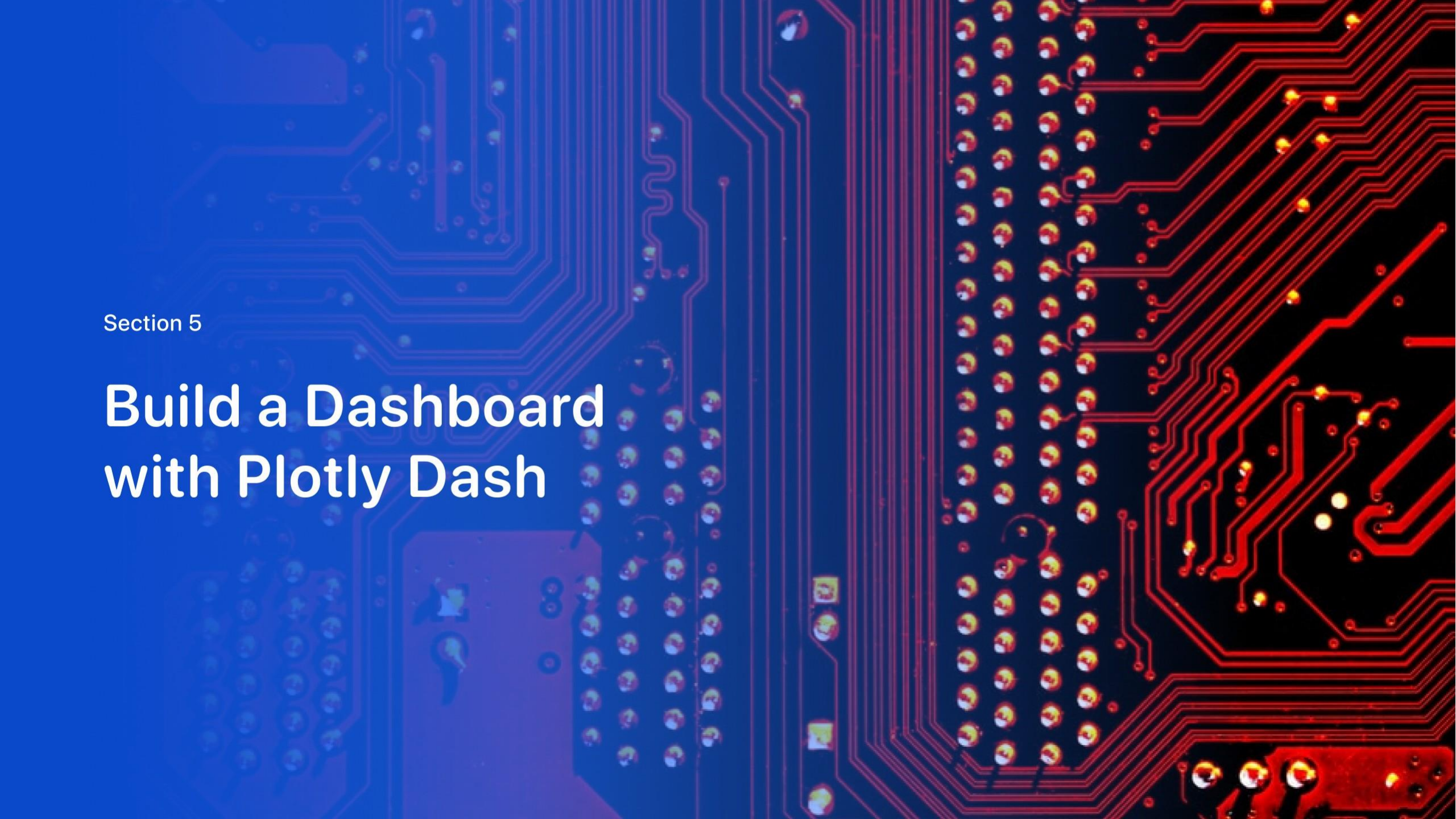
Florida Launch Sites

Green Marker shows successful Launches and **Red** Marker shows Failures

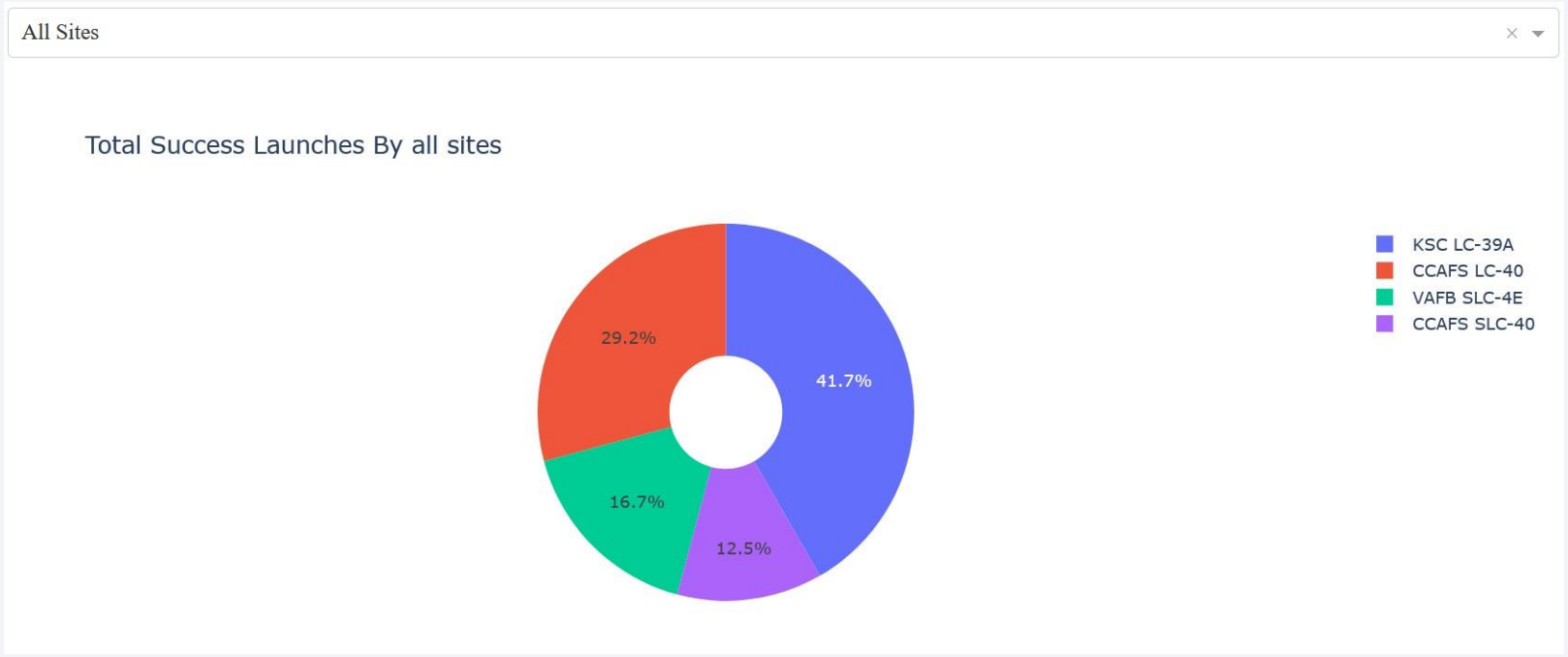
Launch Site distance to landmarks



Distance was presented with ‘polylines’ (blue) on map. We assign longitude and latitude coordinates to landmarks, by using ‘Haversine’ formula distances are calculated.

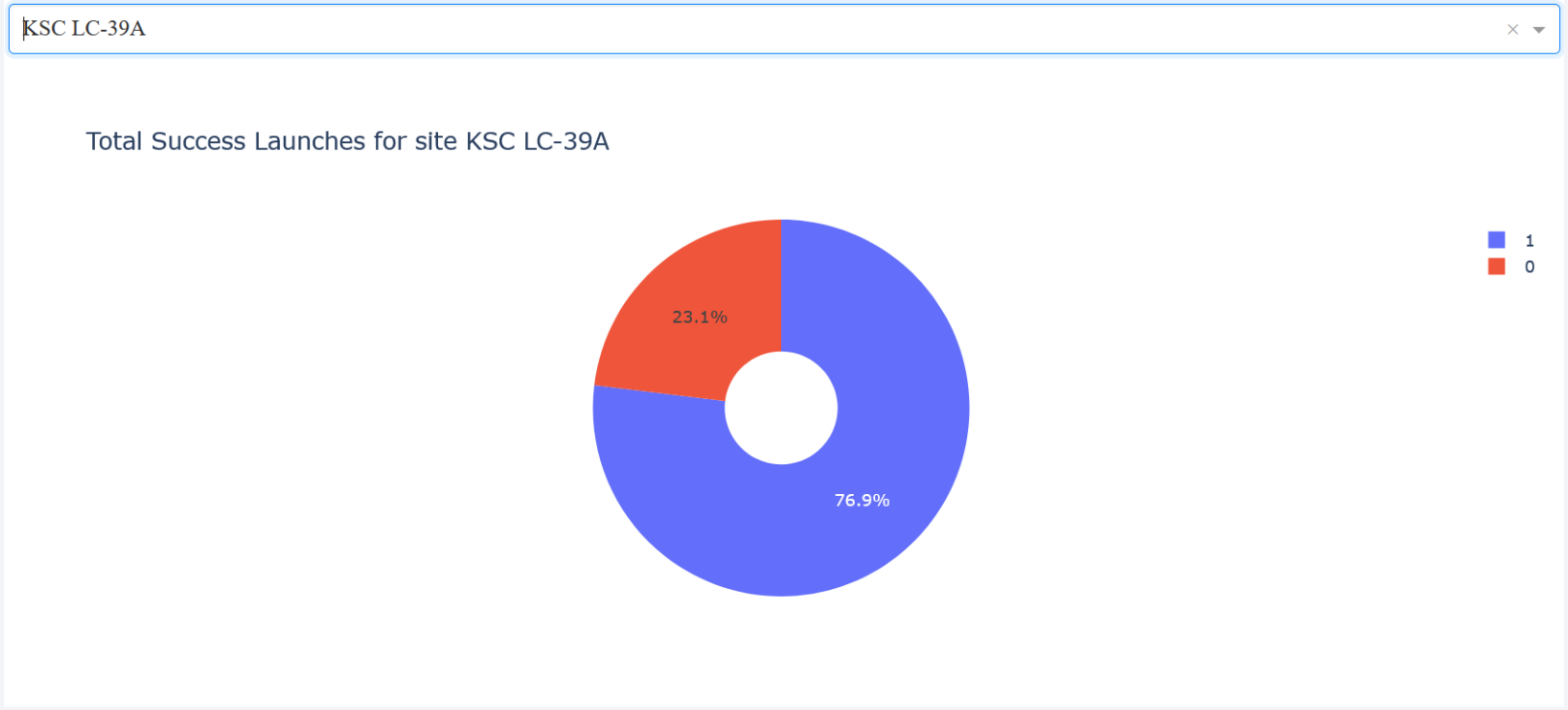

Section 5
Build a Dashboard with Plotly Dash

Dash – Pie chart of success percentage



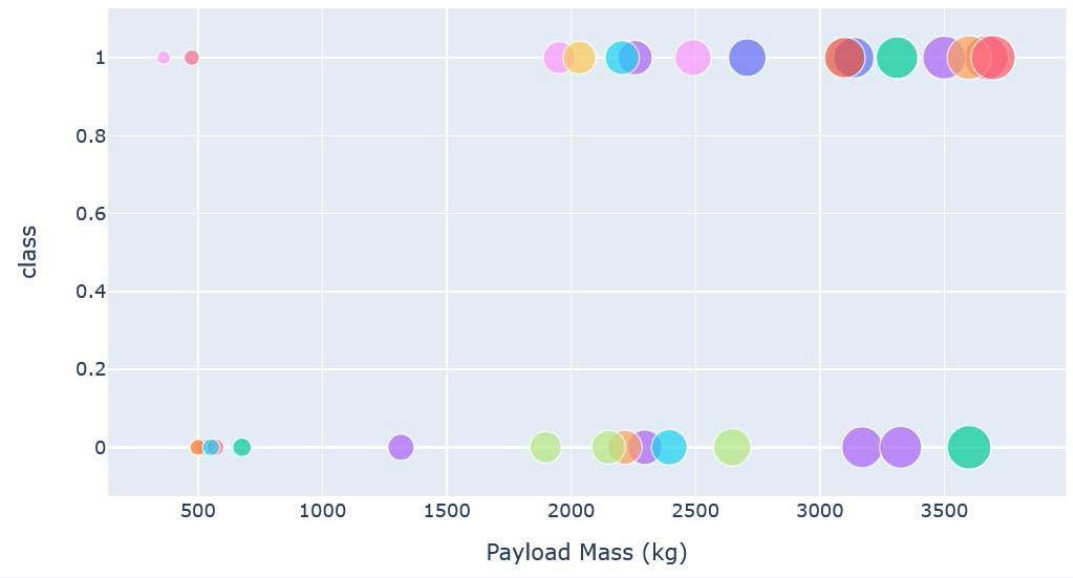
KSC LC-39A has highest successful rate comparing other launch sites

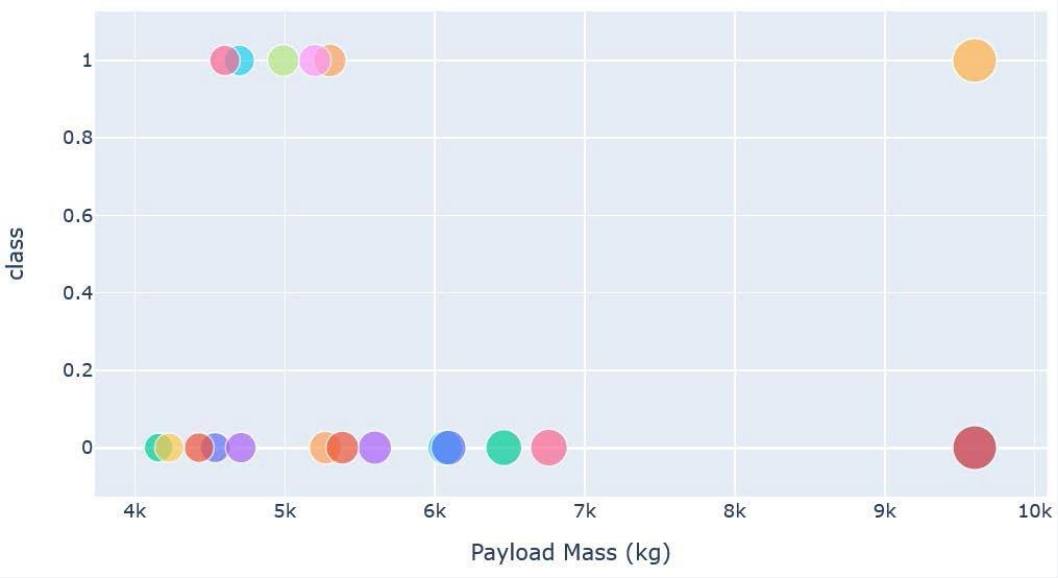
Dash – Highest launch success ration



KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

Dash – Payload vs. Launch Outcome





We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

Section 6

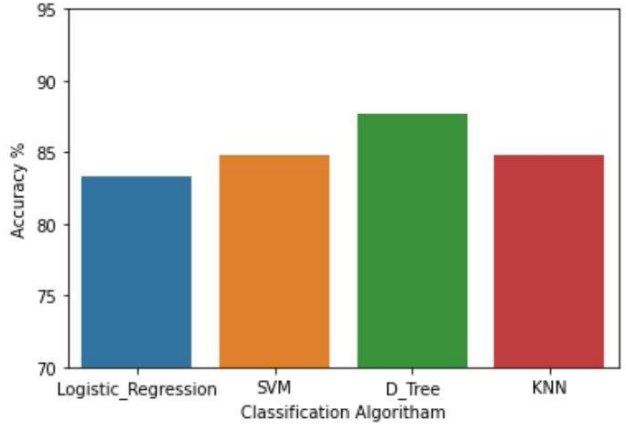
Predictive Analysis (Classification)

Classification Accuracy

```
algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Tree':
    print('Best Params is :',tree_cv.best_params_)
if bestalgorithm == 'KNN':
    print('Best Params is :',knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best Params is :',logreg_cv.best_params_)
```

```
Best Algorithm is Tree with a score of 0.8875000000000002
Best Params is : {'criterion': 'gini', 'max_depth': 16, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_s
plit': 5, 'splitter': 'best'}
```

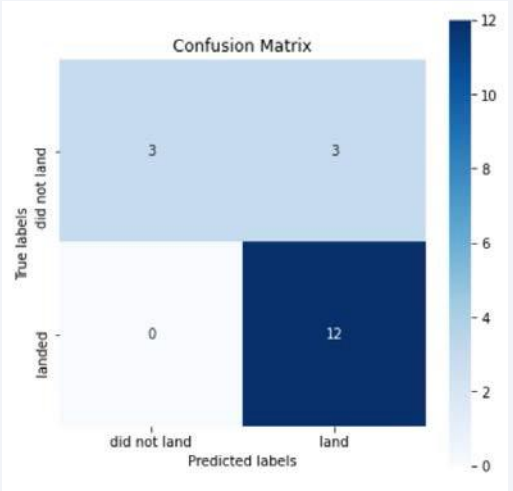
As you can see our accuracy is extremely close but we do have a winner its down to +/- couple percent's, bar plot show 'Decision Tree' as algorithm with best accuracy for this data set.



Confusion Matrix

```
algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Tree':
    print('Best Params is :',tree_cv.best_params_)
if bestalgorithm == 'KNN':
    print('Best Params is :',knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best Params is :',logreg_cv.best_params_)
```

```
Best Algorithm is Tree with a score of 0.8767857142857143
Best Params is : {'criterion': 'gini', 'max_depth': 10, 'max_features': 'auto', 'min_samples_
leaf': 1, 'min_samples_split': 10, 'splitter': 'random'}
```



Examining the confusion matrix, we see that Tree can distinguish between the different classes. We see that the problematic is false positives.

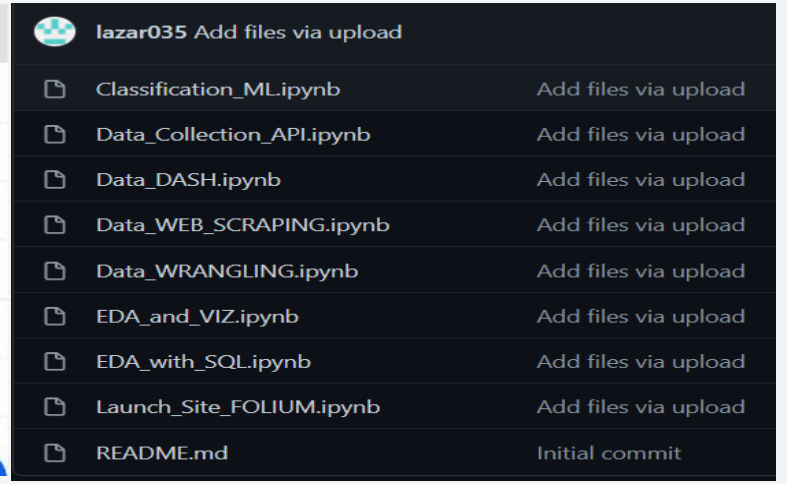
	Predicted 0	Predicted 1
Actual 0	TN	FP
Actual 1	FN	TP

Conclusions

- The Tree Classifier Algorithm is the best for Machine Learning for this dataset
- Low weighted payloads perform better than the heavier payloadsPoint 3
- The success rates for SpaceX launches is directly proportional time in years they will eventually perfect the launches
- We can see that KSC LC-39A had the most successful launches from all the sites
- Orbit GEO,HEO,SSO,ES-L1 has the best Success Rate

Appendix

Name	Shared	Scheduled	Status	Language	Last editor	Last modified
Data_Collection_API				Python 3.8	IBMid-674000I1P8	Sep 07, 2021
Data_WEB_SCRAPING				Python 3.8	IBMid-674000I1P8	Sep 07, 2021
EDA_and_VIZ					IBMid-674000I1P8	Sep 07, 2021
EDA_with_SQL				Python 3.8	IBMid-674000I1P8	Sep 07, 2021
Launch_Site_FOLIUM				Python 3.8	IBMid-674000I1P8	Sep 07, 2021
Data_DASH					IBMid-674000I1P8	Sep 07, 2021
Classification_ML					IBMid-674000I1P8	Sep 07, 2021



- All notebooks are exist on IBM Cloud ‘Watson Studio’ and they have been transferred to GitHub repository
- All work was done on Anacoda3 ‘JupyterLab’ and ‘Jupyter Notebook’

Thank you!