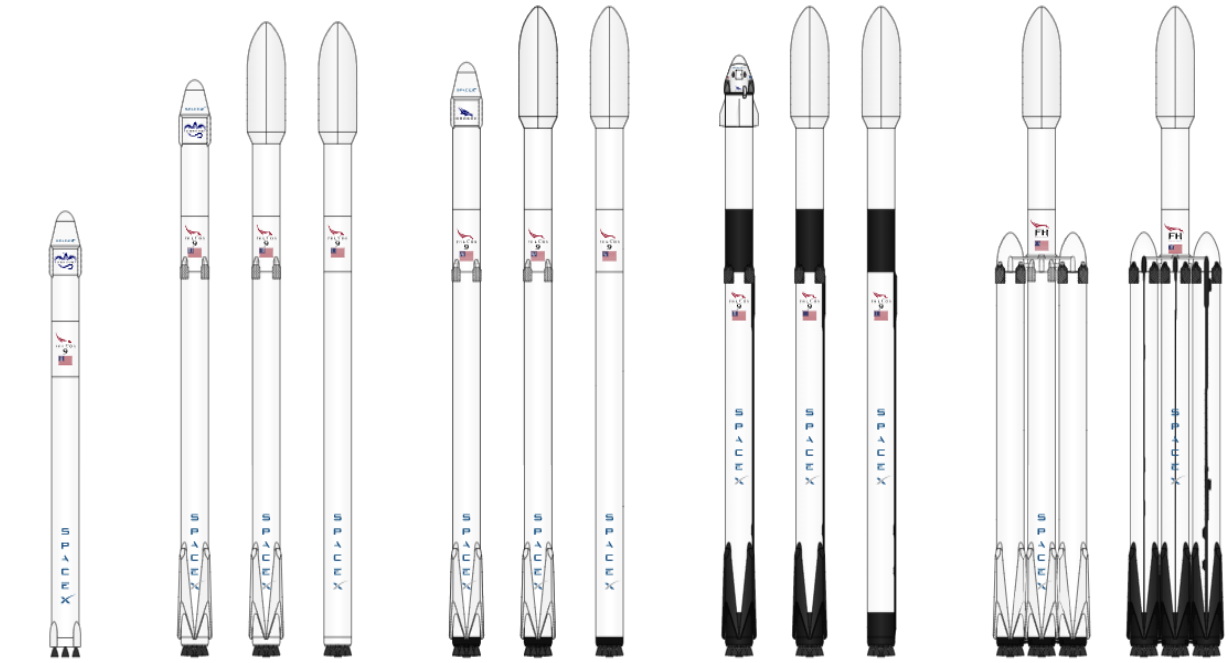
IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Ashweej Shenoy
Oct 03, 2022

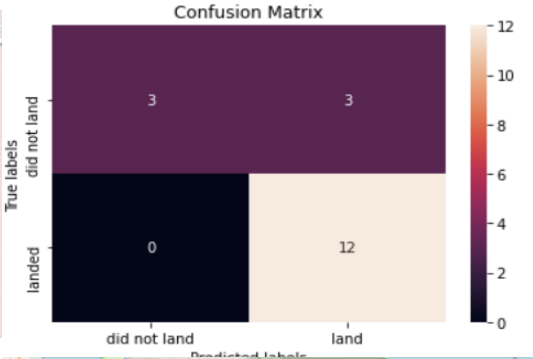
Outline

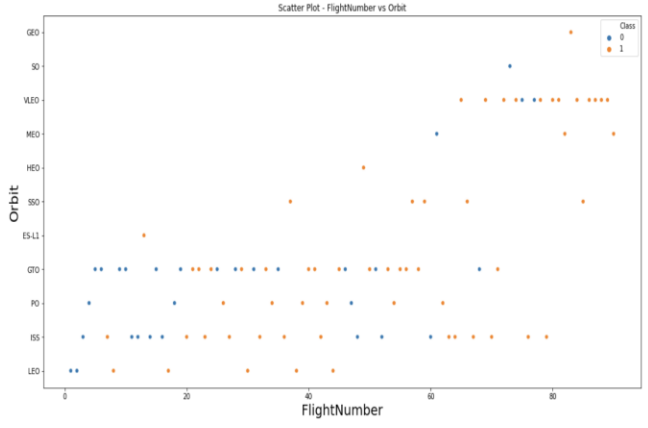
- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion

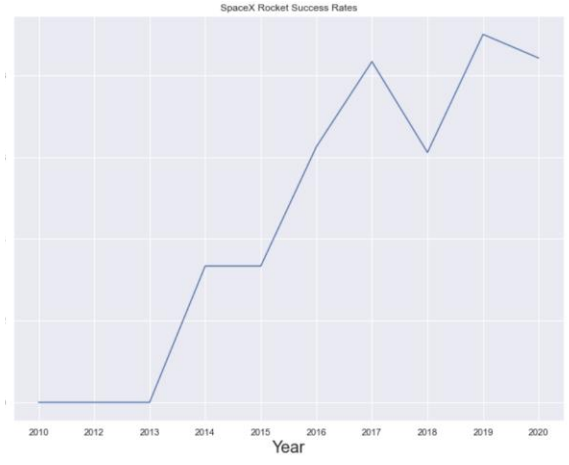


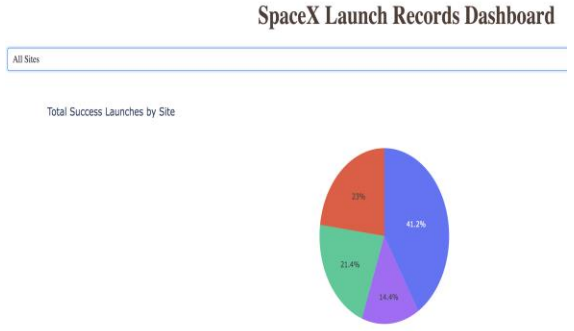
Executive Summary

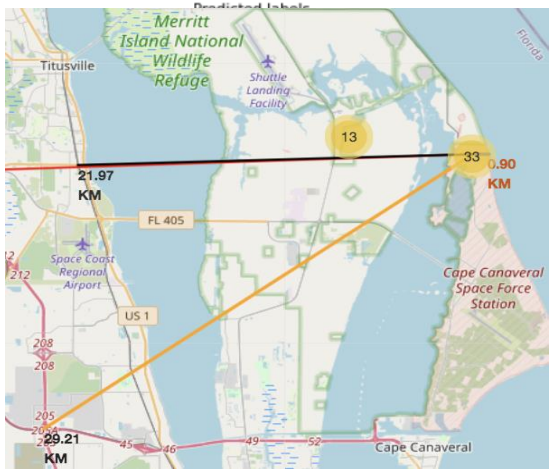
- Summary of Methodologies
  - Data Collection with API, Web-scraping & SQL
  - Data Wrangling & Analysis
  - Interactive Dashboard & Maps with Folium
  - Predictive Analysis for classification model – Logistic Regression, SVM, Decision Tree and KNN
- Summary of all results
  - Interactive Data Visualization
  - Best Model for Classification











Introduction

- **Project background and context**
  - SpaceX 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 SpaceX 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 SpaceX for a rocket launch. In this module, you will be provided with an overview of the problem and the tools you need to complete the course.
- **Project Objective**
  - Use publicly available SpaceX launch data to
    - To create a dashboard for the team
    - To determine whether SpaceX will reuse the first stage through a Machine learning model
- **Problems you want to find answers**
  - To find factors that Rocket will land successfully

Section 1

Methodology

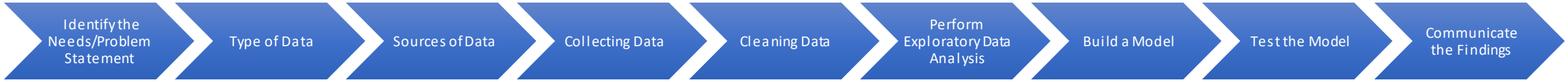
Methodology

Executive Summary

- Data collection methodology:
  - SpaceX REST API
  - Web scraping from Wikipedia
- Perform data wrangling
  - Clean the Data
  - One Hot Encoding
  - Drop data not necessary for Machine Learning Algorithm
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Scatter Plot & Bar Chart to understand data
  - SQL Analysis using SQLite
- Perform interactive visual analytics using Folium and Plotly Dash
  - Build a Plotly Dashboard
  - Folium Map Visualization
- Perform predictive analysis using classification models
  - Build and Evaluate Models – Logistic Regression, Decision Tree, SVM, KNN
  - Best parameters for each model – GridSearchCV

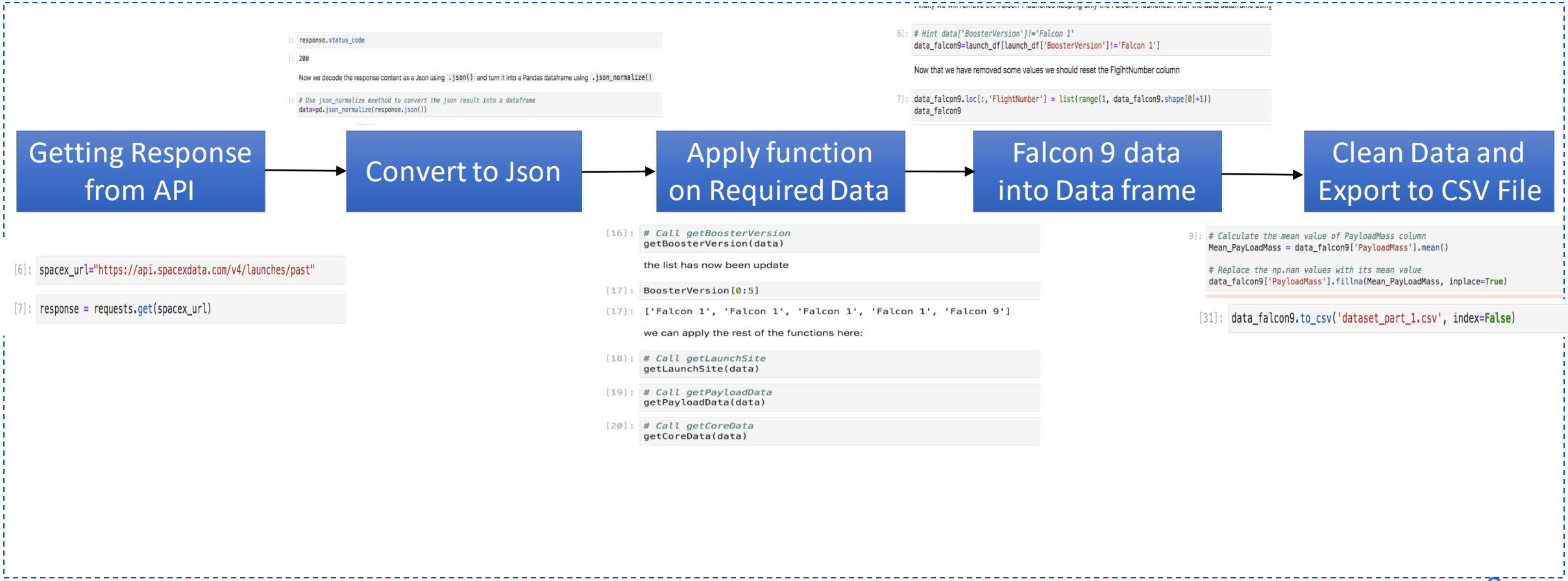
Data Collection

- Data Collection is a set of steps undertaken to collect all necessary data required to answer the question



Data Collection – SpaceX API

- Github URL to Data Collection SpaceX API



Data Collection - Scraping

GitHub URL for Web scraping notebook

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

```
[5]: # use requests.get() method with the provided static_url
     # assign the response to a object
     html_data=requests.get(static_url)
     html_data.status_code
```

```
[5]: 200
```

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

```
[6]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
     soup = BeautifulSoup(html_data.text, 'html.parser')
```

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

```
[7]: # Use soup.title attribute
     soup.title
```

```
[7]: <title>List of Falcon 9 and Falcon Heavy launches — Wikipedia</title>
```

```
[10]: column_names = []

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

Check the extracted column names

```
[11]: print(column_names)
```

```
['Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome']
```

```
[12]: 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']=[]
```

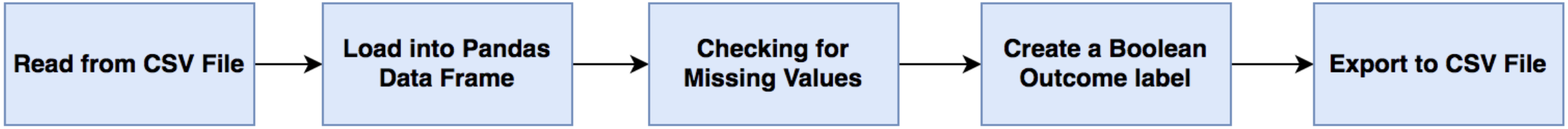
```
[14]: df=pd.DataFrame(launch_dict)
```

```
[16]: df.to_csv('spacex_web_scraped.csv', index=False)
```

- Getting Response from HTML
- Create Beautiful Soup Object
- Search for tables in Webpage
- Extract Columns from Table
- Create Dictionary and add data extracted from webpage
- Convert dictionary to a Pandas Dataframe
- Save the Webscraped data into CSV file

Data Wrangling

- In Data Wrangling, we clean the data sets, join multiple data sets so that it is easier to access and analyze
- GitHub URL - Data wrangling




```
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	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.561857
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561857
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577366	28.561857
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003	-120.610829	34.632093
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004	-80.577366	28.561857
5	6	2014-01-06	Falcon 9	3325.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1005	-80.577366	28.561857
6	7	2014-04-18	Falcon 9	2296.000000	ISS	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0	0	B1006	-80.577366	28.561857
7	8	2014-07-14	Falcon 9	1316.000000	LEO	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0	0	B1007	-80.577366	28.561857
8	9	2014-08-05	Falcon 9	4535.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1008	-80.577366	28.561857
9	10	2014-09-07	Falcon 9	4428.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1011	-80.577366	28.561857

```
df.isnull().sum()/df.count()*100
```

```
df.dtypes
```

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class =[]

for key, value in df['Outcome'].items():
    if value in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

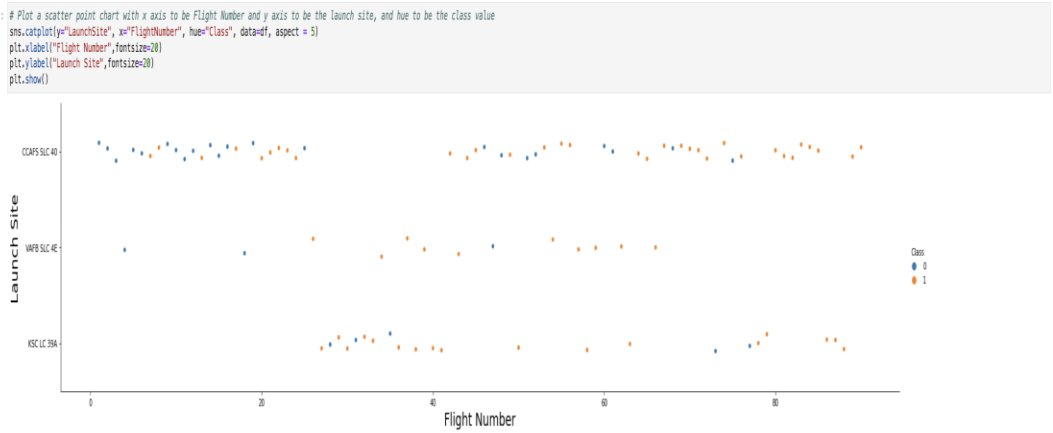
```
df.head(5)
```

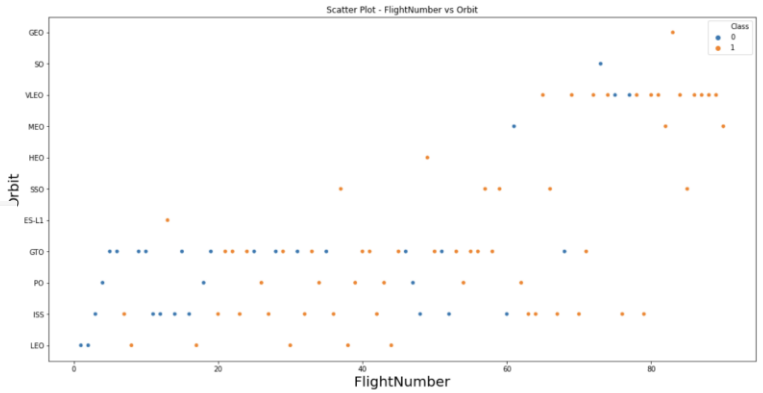
	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.561857	0
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561857	0
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577366	28.561857	0
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003	-120.610829	34.632093	0
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004	-80.577366	28.561857	0

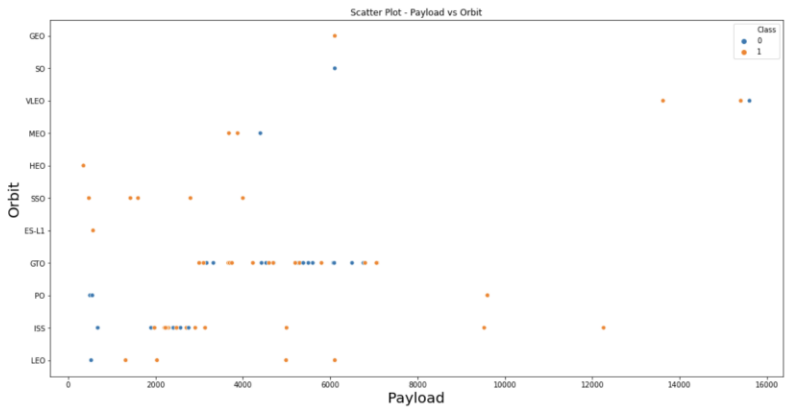
```
df.to_csv("dataset_part_2.csv", index=False)
```

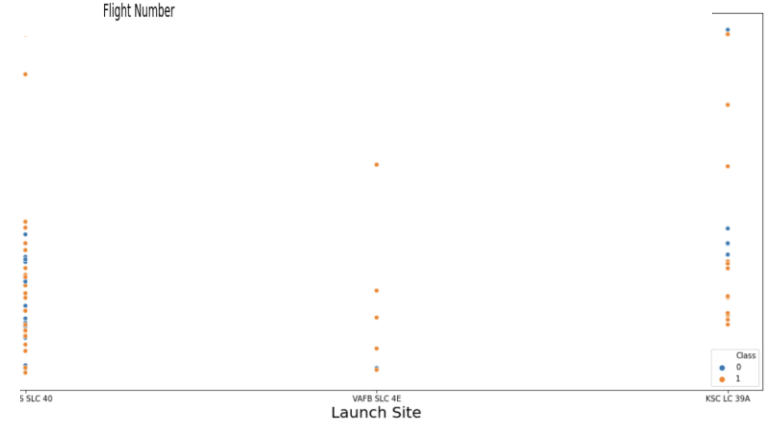
EDA with Data Visualization

- Here we summarize the data and visualize the data to understand the characteristics of the data.
- GitHub URL EDA with data visualization
- Scatter Plot
  - Launch Site vs Flight Number
  - Launch Site Vs Payload Mass
  - Flight Number vs Orbit
  - Payload vs Orbit







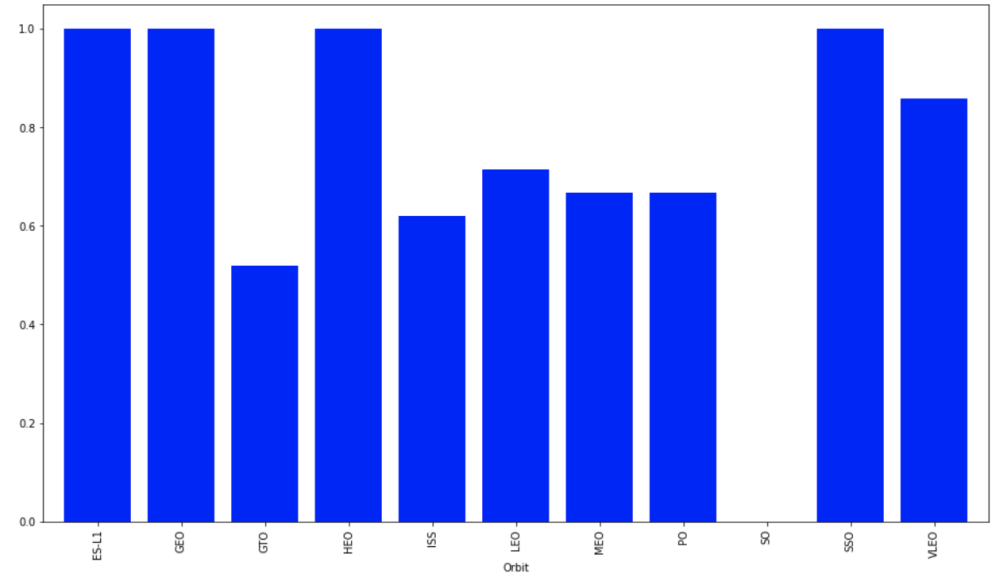


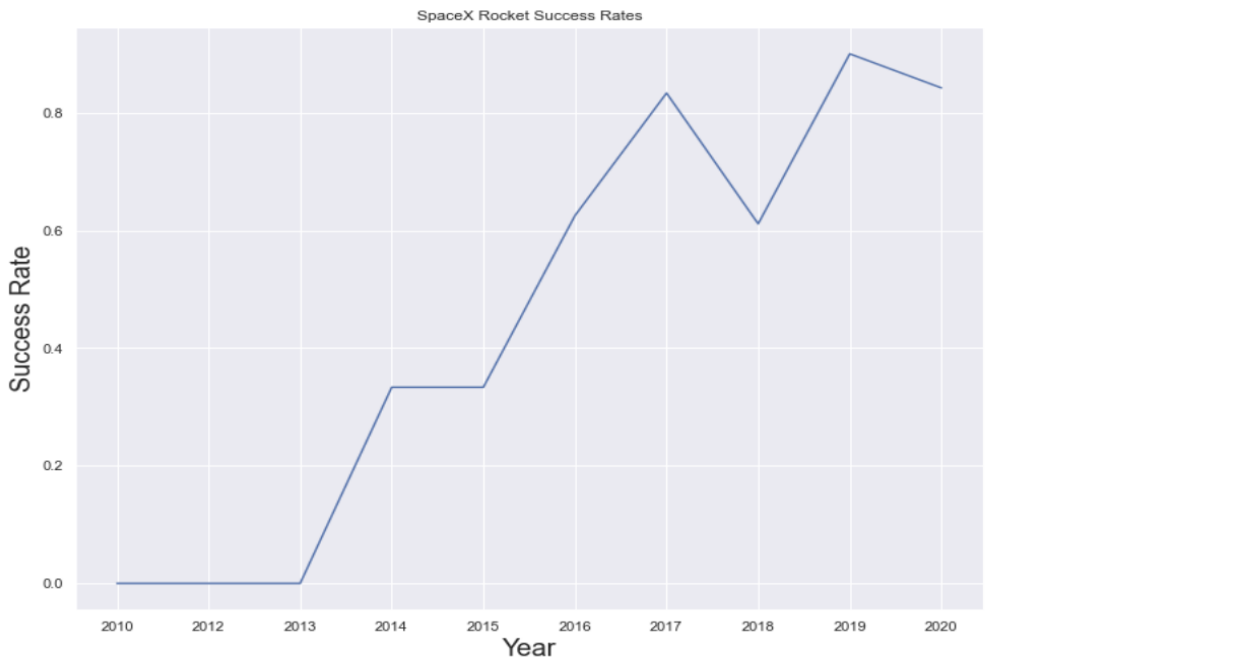
EDA with Data Visualization Contd..

- Bar Chart
- Line Chart of Success Rate vs Year

```
[6]: # HINT use groupby method on Orbit column and get the mean of Class column
df.groupby('Orbit')['Class'].mean().plot(kind='bar', figsize=(16, 9), color='blue', zorder=2, width=0.8)

[6]: <AxesSubplot:xlabel='Orbit'>
```





EDA with SQL

- SQL queries were written for SQLLite
- GitHub URL - EDA with SQL notebook
- SQL queries were written to gather data from given 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 succesful landing outcome in ground pad was acheived.
  - 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 using a subquery
  - List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.
  - Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order

Build an Interactive Map with Folium

- *Folium* builds on the data wrangling strengths of the Python ecosystem and the mapping strengths of the Leaflet.js library. Here we use latitude and longitudes of each launch site and add Circle markers for each site. Further, we add label to each marker and segregate the Success and Failure for each launch site.

Sl No	Map Object	Code	Result
1	Map Marker	folium.Marker (<<parmaters>>)	Map object to make mark on the map
2	Icon Marker	folium.Icon(<<paramaters>>)	To create an icon on map
3	Circle Marker	folium.Circle(<<parameters)	To create a Circle where Marker is placed
4.	PolyLine	folium.PolyLine(<<parameters>>)	To create a line between 2 points on map
5	Marker Cluster	MarkerCluster()	Used in case of map containing many markers

- GitHub URL Interactive map with Folium map

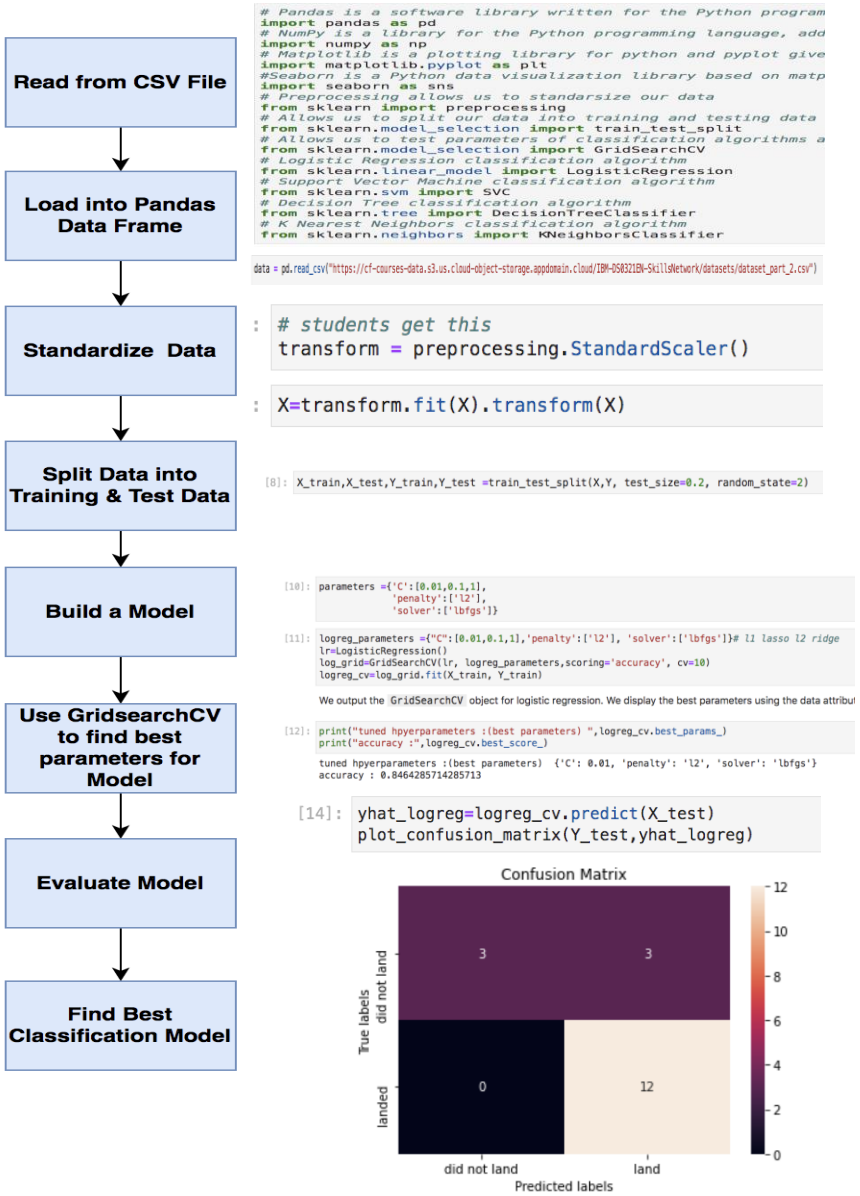
Build a Dashboard with Plotly Dash

- Summarize what plots/graphs and interactions you have added to a dashboard

Object	Code	Result
Dash	import dash import dash_html_components as html import dash_core_components as dcc from dash.dependencies import Input, Output	Dash open source allows for Dash apps to run through local server on your laptop. Dash core components library contains higher level components – sliders, drop down etc. Dash also provides html tags as user friendly python classes
Dropdown	dcc.Dropdown (<<parameters>>)	To create dropdown for launch site
Rangslider	dcc.RangeSlider(<<parameters)	To create RangeSlider for Payload Mass selection
Pie Chart	Plotly.Express.pie(<<parameters>>)	Pie chart to display Success % display
Scatter Plot	Plotly.Express.scatter(<<parameters>>)	Correlation between Payload & Success for all sites

- GitHub URL of Plotly Dash lab

Predictive Analysis (Classification)



- Load cleaned and feature engineered dataset into a dataframe
- Transform data into Numpy arrays
- Standardize & Transform Data
- Split the data into Training & Test data
- Fit our datasets into the GridsearchCV to train the model and find best parameters for the Classification models
- Check the accuracy of each classification model
- Get best hyper-parameters for each model
- Plot Confusion Matrix
- Model with best accuracy score, Jaccard Score and F1 Score is recognized as the best classification model

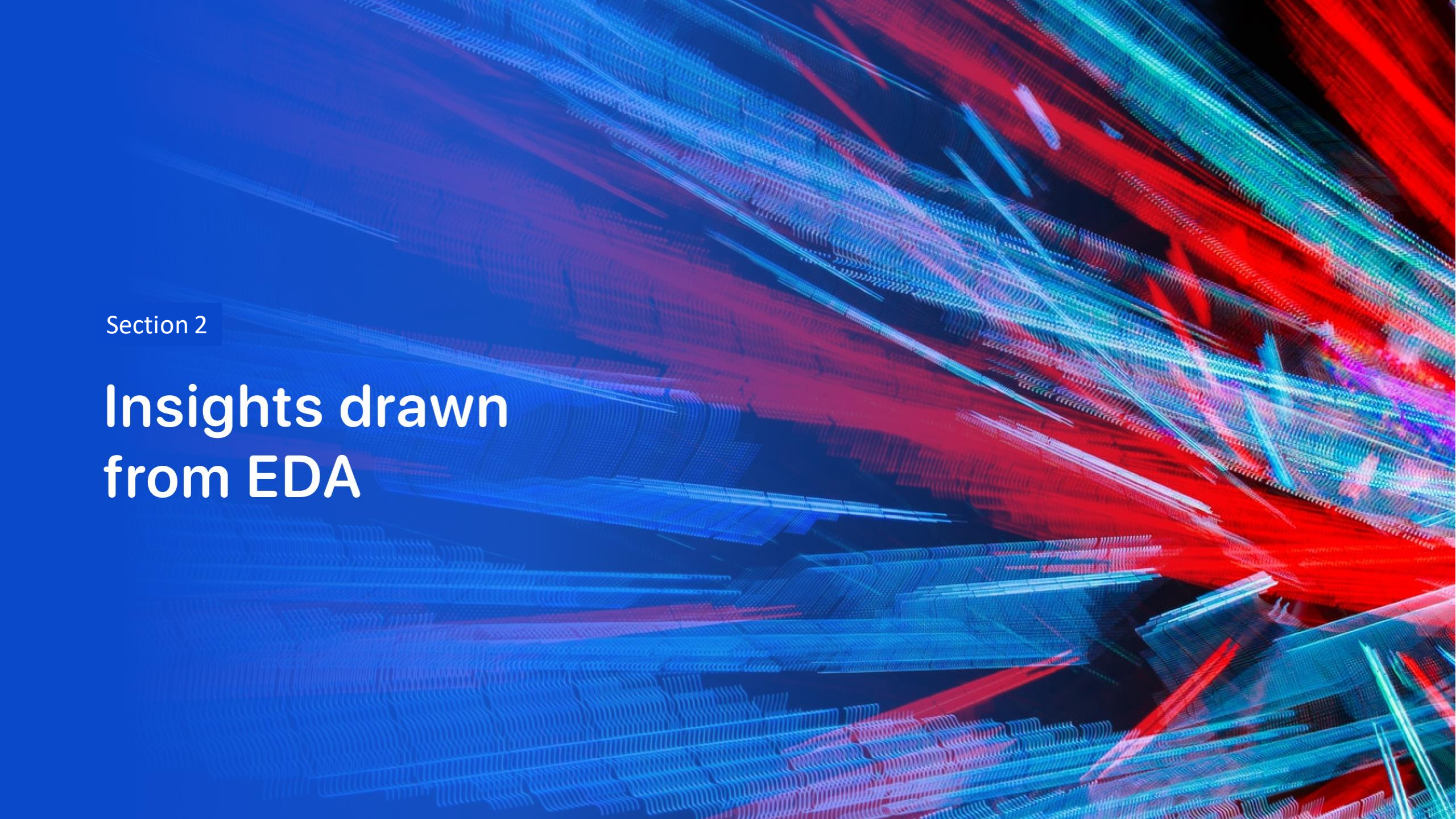
GitHub URL - Predictive analysis

Results

- Exploratory Data Analysis Results
- Interactive Analytics Results
- Predictive Analysis Results

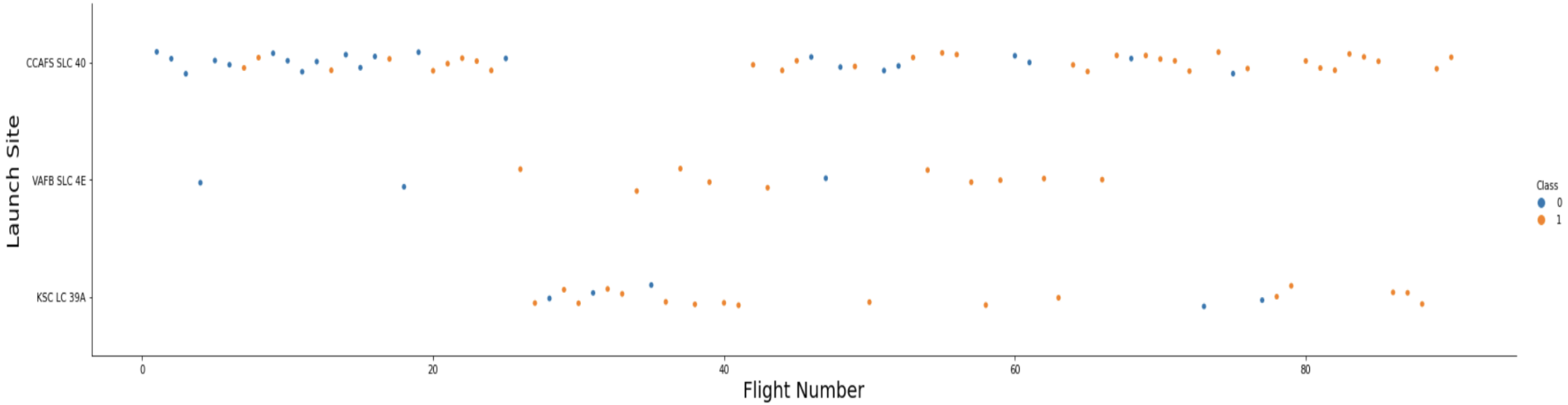
Section 2

Insights drawn from EDA



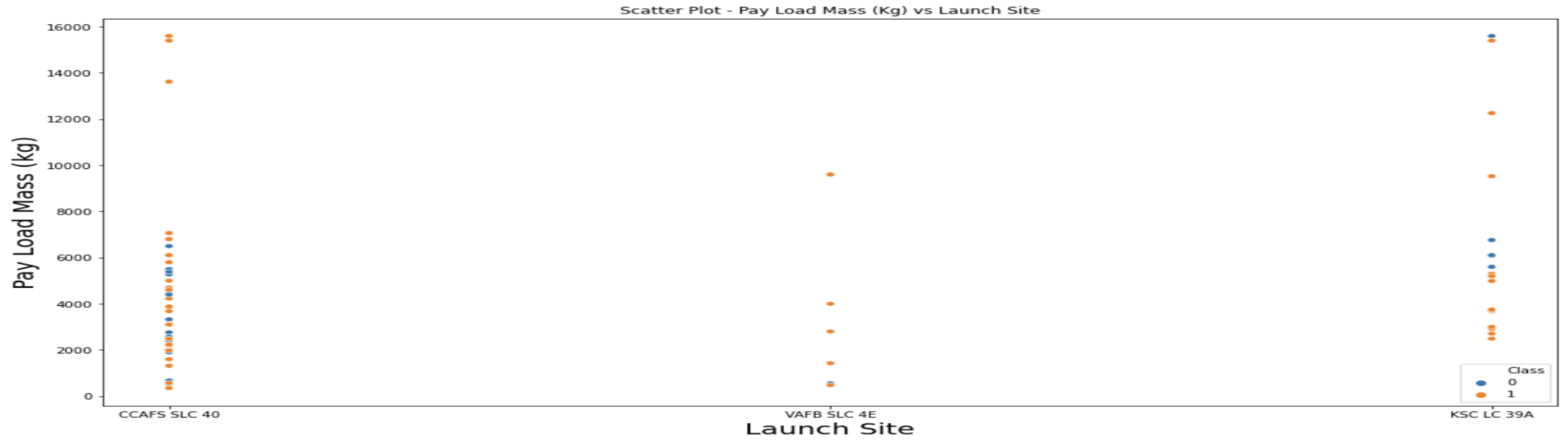
Flight Number vs. Launch Site

Higher Flight Numbers, irrespective of Launch site, the success rate is increasing



Payload vs. Launch Site

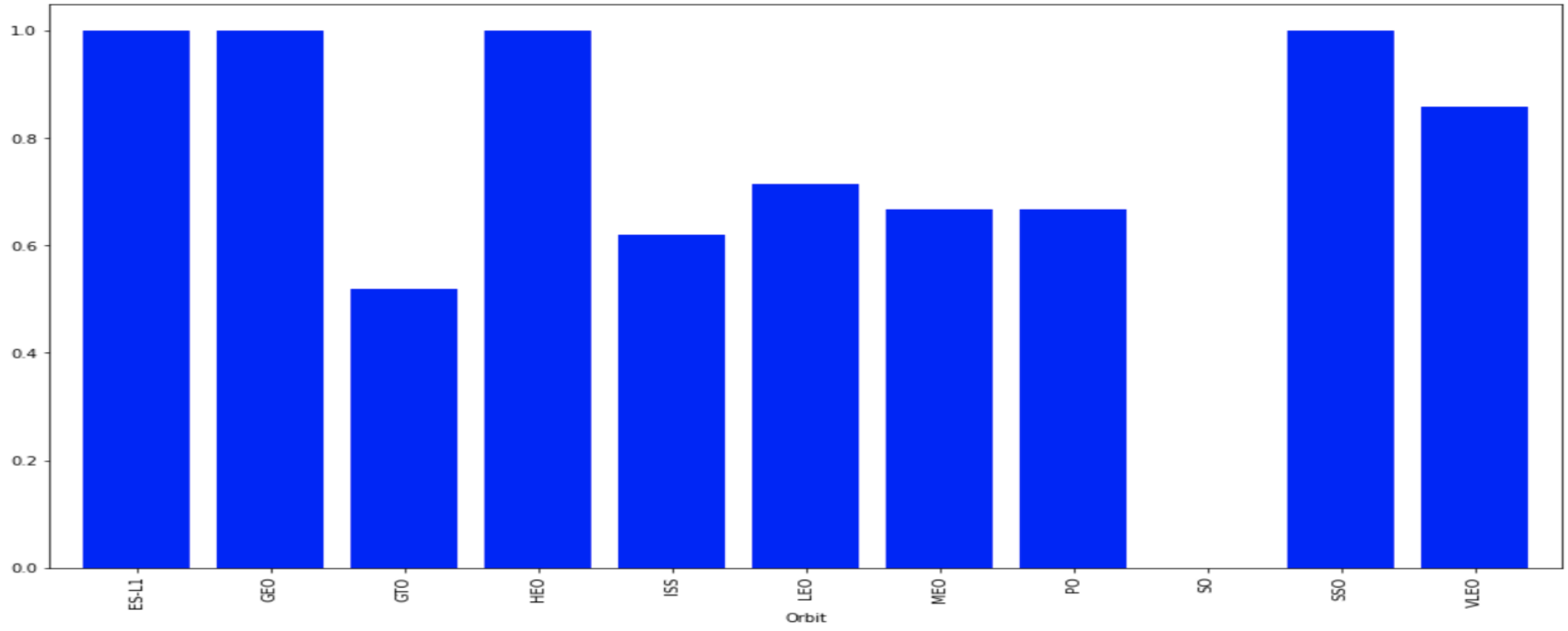
Greater the Pay Load Mass (>7000 Kgs) higher the Success Rate; but no clear pattern for pay load mass less than 7000 Kgs



Success Rate vs. Orbit Type

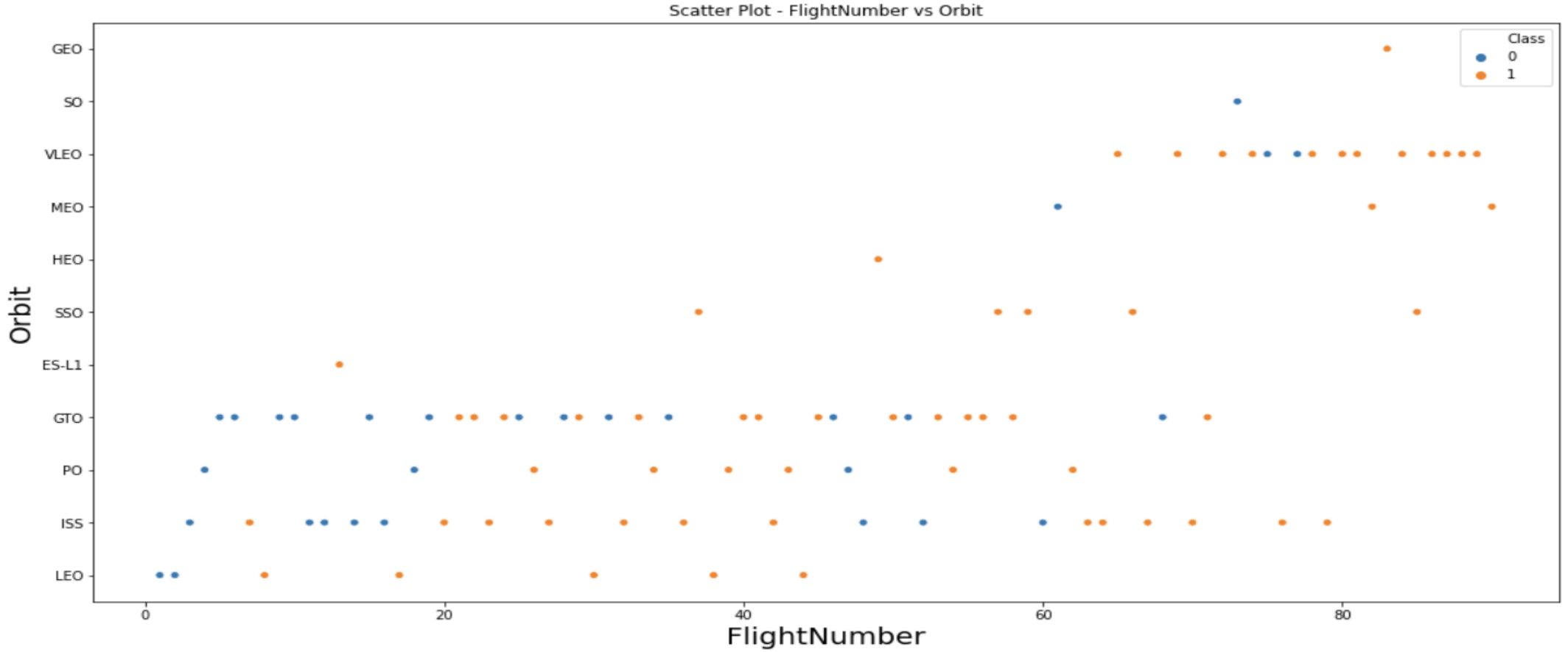
Highest Success Rate for Orbits –ES-L1, GEO, HEO,SSO

[6]: <AxesSubplot:xlabel='Orbit'>



Flight Number vs. Orbit Type

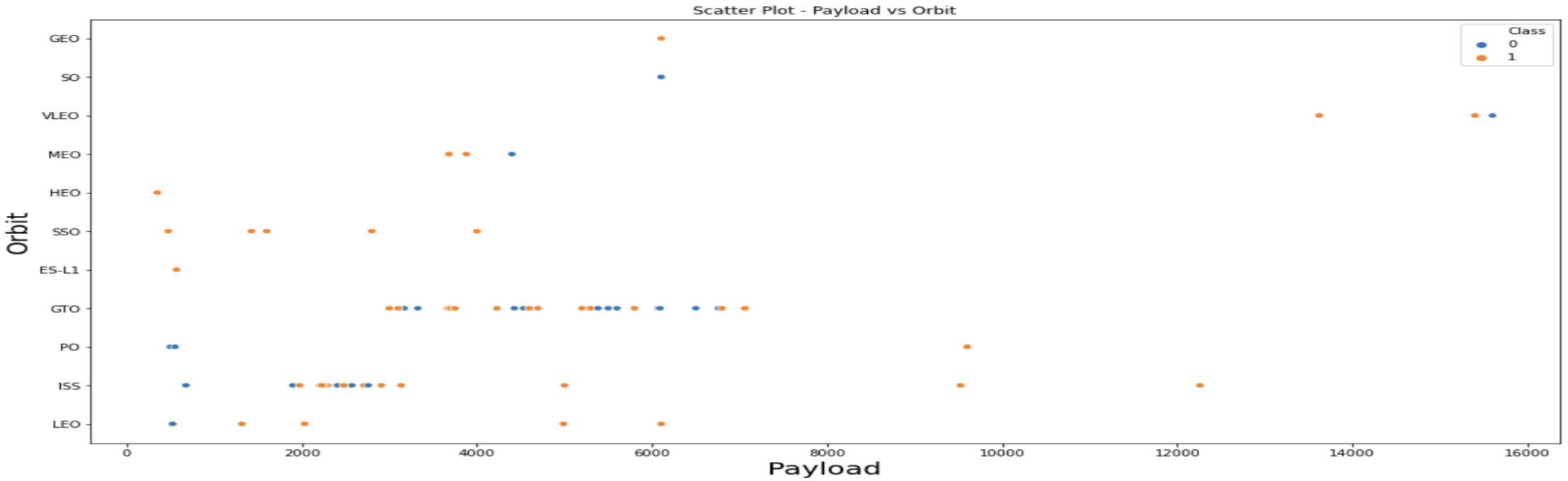
LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.



Payload vs. Orbit Type

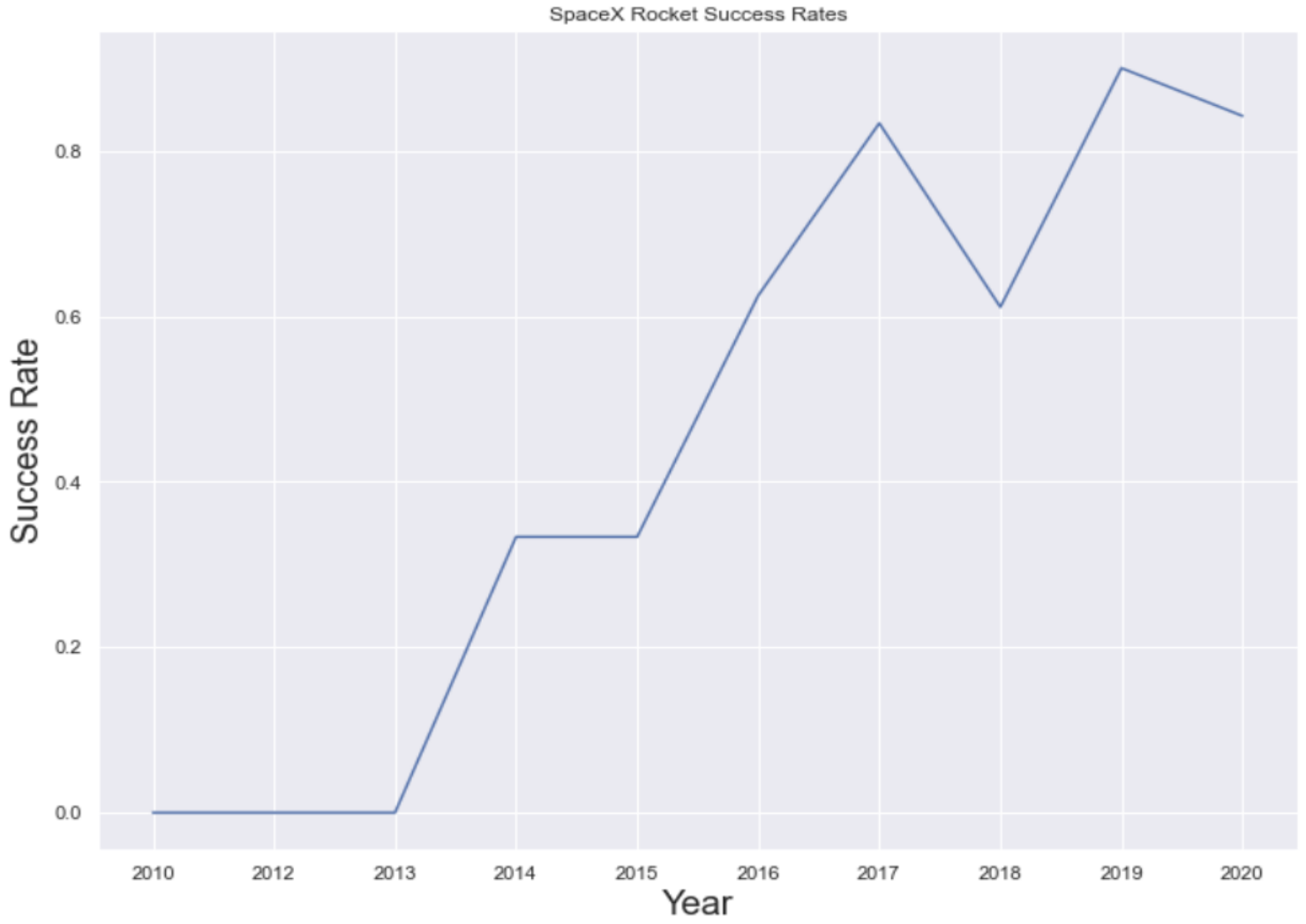
With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS.

For GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.



Launch Success Yearly Trend

Rate of Success drastically improve after 2013 with dip in 2018



All Launch Site Names

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

```
[7]: %sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL;
 * sqlite:///my_data1.db
Done.
```

[7]:

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

SQL Query with Distinct shows unique value

Query Results – Unique Launch Sites

Launch Site Names Begin with 'CCA'

```
8]: %sql SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
 * sqlite:///my_data1.db
Done.
```

8]:

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

SQL Query with Like keyword& Limit

Query Results

- Limit 5 – restricts query results to 5 rows
- Like keyword – allows for wild character search; 'CCA%' implies any launch site that begins with 'CCA'

Total Payload Mass

```
[9]: %sql SELECT SUM(PAYLOAD_MASS__KG_) AS 'TOTAL PAYLOAD MASS FOR NASA(CRS)' FROM SPACEXTBL WHERE CUSTOMER='NASA (CRS)';
 * sqlite:///my_data1.db
Done.
[9]: TOTAL PAYLOAD MASS FOR NASA(CRS)
                                45596
```

SQL Query with Sum and Where clause

Query Results for Total payload mass for NASA(CRS)

- Sum function calculates total for column on Payload_mass__kg_ on filtered data set
- where clause filters data for customer 'NASA (CRS)'

Average Payload Mass by F9 v1.1

```
[10]: %sql SELECT AVG(PAYLOAD_MASS__KG_) AS 'AVERAGE PAYLOAD MASS FOR BOOSTER F9 v1.1' FROM SPACEXTBL WHERE BOOSTER_VERSION ='F9 v1.1';
 * sqlite:///my_data1.db
Done.
[10]:
```

AVERAGE PAYLOAD MASS FOR BOOSTER F9 v1.1
2928.4

SQL Query with Avg and Where clause

Query Results for Average payload mass for Booster F9 v1.1

- Avg function calculates the average in the column Payload_mass__Kg_ on filtered data set
- where clause filter the data set for booster version F9 v1.1

First Successful Ground Landing Date

```
[11]: %sql SELECT MIN(SUBSTR("DATE",7,10)||"-"||TRIM(SUBSTR("DATE",3,4),"-")||"-"||SUBSTR("DATE",1,2)) AS "FIRST SUCCESSFUL LANDING"  FROM SPACEXTBL WHERE "Landing _Outcome" ='Success (ground pad)'
 * sqlite:///my_data1.db
Done.
[11]: FIRST SUCCESSFUL LANDING
      2015-12-22
```

SQL Query with Min and Where clause

Query Results for First Successful Ground landing

- Min function finds the minimum value in the column Date on filtered data set
- Where clause filters the table on column Landing_Outcome ='Success (ground pad)'

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql SELECT DISTINCT BOOSTER_VERSION AS 'BOOSTERS' FROM SPACEXTBL WHERE "Landing _Outcome" = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ <6000
```

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

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

SQL Query with Distinct and Where clause

Query Results for Successful Drone Ship Landing with payload mass between 4000 and 6000 Kgs

- Where clause filters the table for landing_outcome ='Success (drone ship) and payload _mass__kg_ greater than 4000 kgs but less than 6000 kgs

Total Number of Successful and Failure Mission Outcomes

```
%sql SELECT SUM(CASE WHEN MISSION_OUTCOME LIKE '%Success%' THEN 1 else 0 END) AS "SUCCESSFUL MISSION", \
    SUM(CASE WHEN MISSION_OUTCOME LIKE '%Failure%' then 1 else 0 END) AS "FAILURE MISSION" \
FROM SPACEXTBL;
```

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

SUCCESSFUL MISSION	FAILURE MISSION
100	1

SQL Query with Sum function

Query Results for total number of successful and failure missions

- Case clause as a subquery to get the count of successful and failure mission outcomes
- Successful and Failure outcomes are represented with boolean values

Boosters Carried Maximum Payload

```
%sql SELECT DISTINCT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ IN (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
 * sqlite:///my_data1.db
Done.
```

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

SQL Query with subquery

Query Results for distinct boosters that carried maximum payload

- Max function is used to find the maximum payload mass in the subquery used in where clause

2015 Launch Records

```
%sql SELECT substr(Date, 4, 2) AS MONTH_NUM, CASE substr(Date, 4, 2) \
WHEN "01" THEN "JAN" \
WHEN "02" THEN "FEB" \
WHEN "03" THEN "MAR" \
WHEN "04" THEN "APR" \
WHEN "05" THEN "MAY" \
WHEN "06" THEN "JUN" \
WHEN "07" THEN "JUL" \
WHEN "08" THEN "AUG" \
WHEN "09" THEN "SEP" \
WHEN "10" THEN "OCT" \
WHEN "11" THEN "NOV" \
ELSE  "DEC" END AS "MONTH NAME", BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE SUBSTR(DATE,7,10)='2015' AND "Landing _Outcome" = 'Failure (drone ship)';
```

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

MONTH_NUM	MONTH NAME	Booster_Version	Launch_Site
01	JAN	F9 v1.1 B1012	CCAFS LC-40
04	APR	F9 v1.1 B1015	CCAFS LC-40

SQL Query for the year 2015 and failed landing outcomes in drone ship

Query Results display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015

- Sqlite has limited date functions, hence using substr function to filter for 2015 and Case function to display the month name
- Where clause filters for landing outcome ="Failure (drone ship)"

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

```
%sql SELECT "Landing _Outcome",COUNT("Landing _Outcome") AS "Rank success count between 2010-06-04 and 2017-03-20" FROM SPACEXTBL \
WHERE "Landing _Outcome" LIKE '%Success%' AND \
(SUBSTR("DATE",7,10)||"-"||TRIM(SUBSTR("DATE",3,4),"-")||"-"||SUBSTR("DATE",1,2))>'2010-06-04' AND \
(SUBSTR("DATE",7,10)||"-"||TRIM(SUBSTR("DATE",3,4),"-")||"-"||SUBSTR("DATE",1,2))<'2017-03-20' \
GROUP BY "Landing _Outcome" ORDER BY COUNT("Landing _Outcome") DESC;
```

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

Landing _Outcome	Rank success count between 2010-06-04 and 2017-03-20
Success (drone ship)	5
Success (ground pad)	3

Query Results

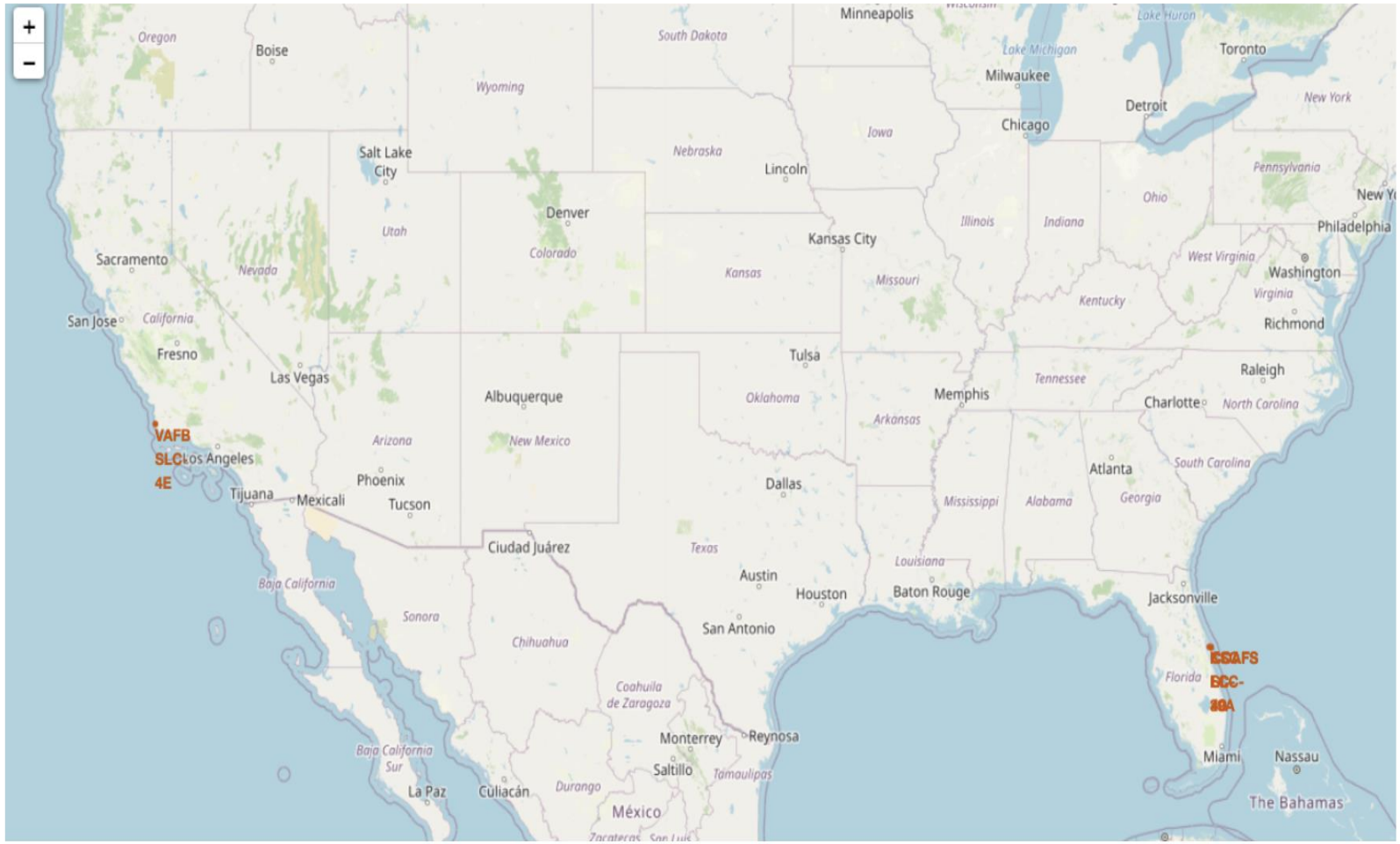
SQL Query for ranking successful landing outcome between 2010-06-04 and 2017-03-20

- Like Clause allows for wild character search for Success Outcome.
- Sqlite has limited function related date, hence substring (substr) function is used to limit date between 2010-06-04 and 2017-03-20
- Count function counts different landing outcomes based on group by clause
- Desc sorts the display results in descending order


Section 3
Launch Sites
Proximities Analysis

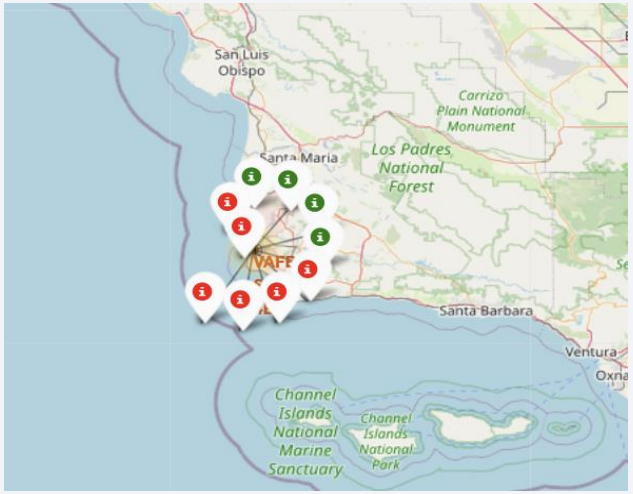
All Launch Sites

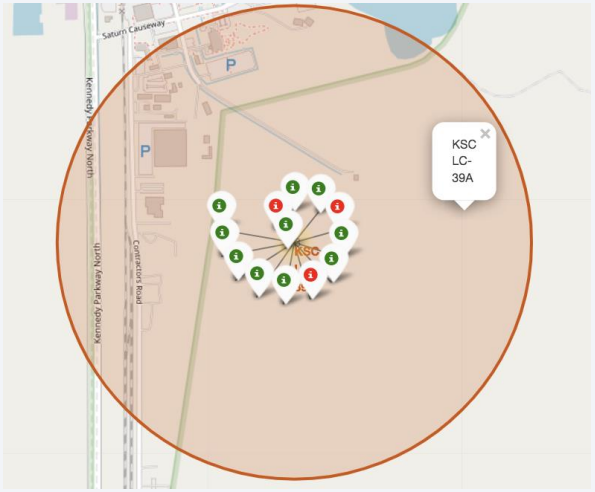
- All launch sites are marked on the map
- Launch sites are on east coast and west coast of USA

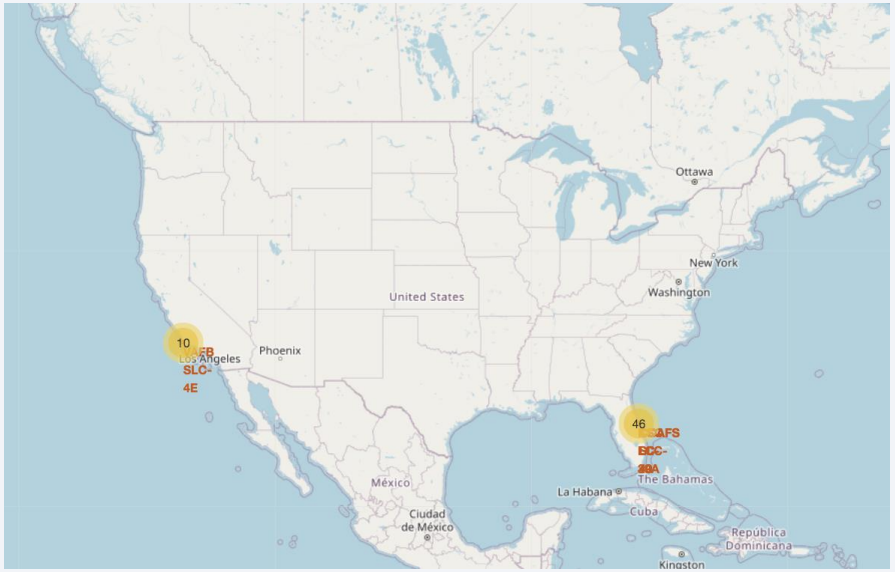


Success and Failure at Launch Site

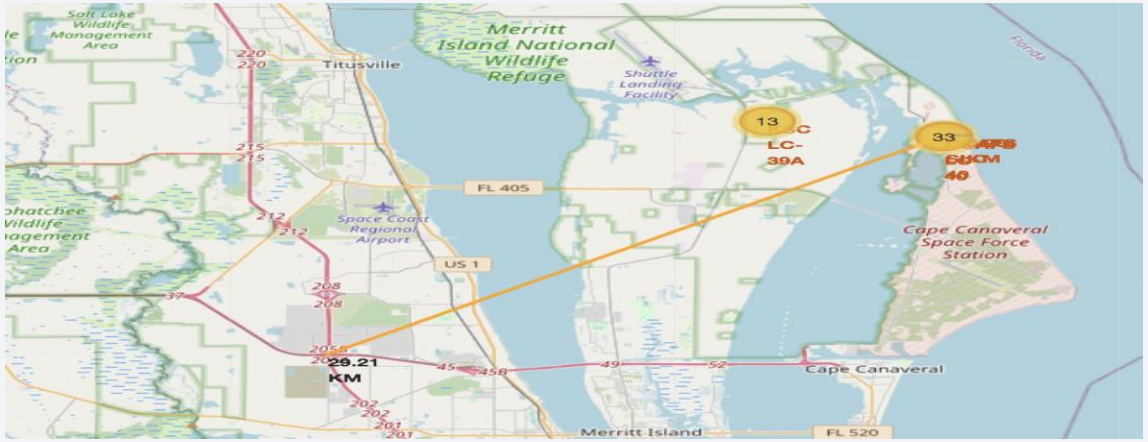
- Green – Successful Launches
- Red – Failure
- KSC LC –39A has more successful launches hence greater probability of success



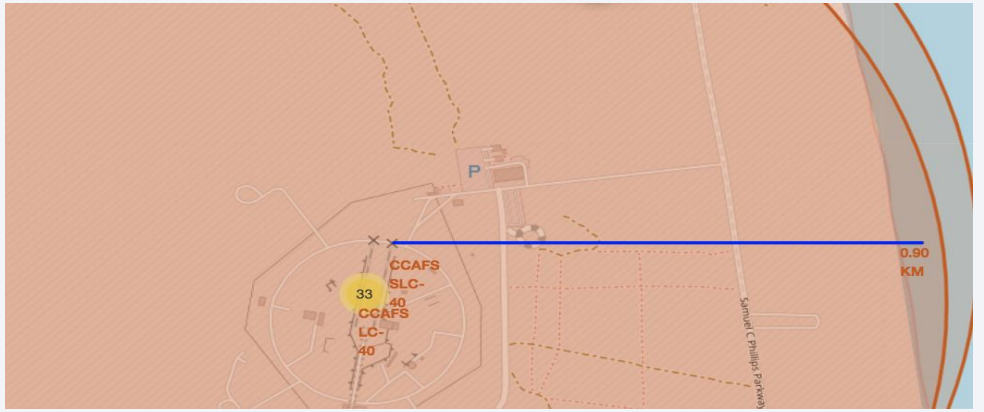




Launch Site Distances

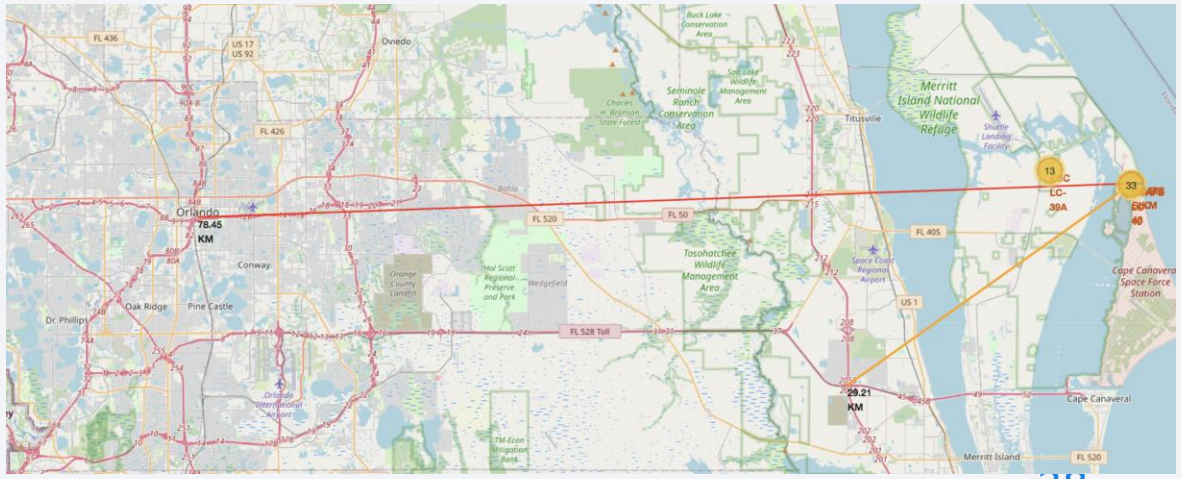


Distance to Highway (orange)

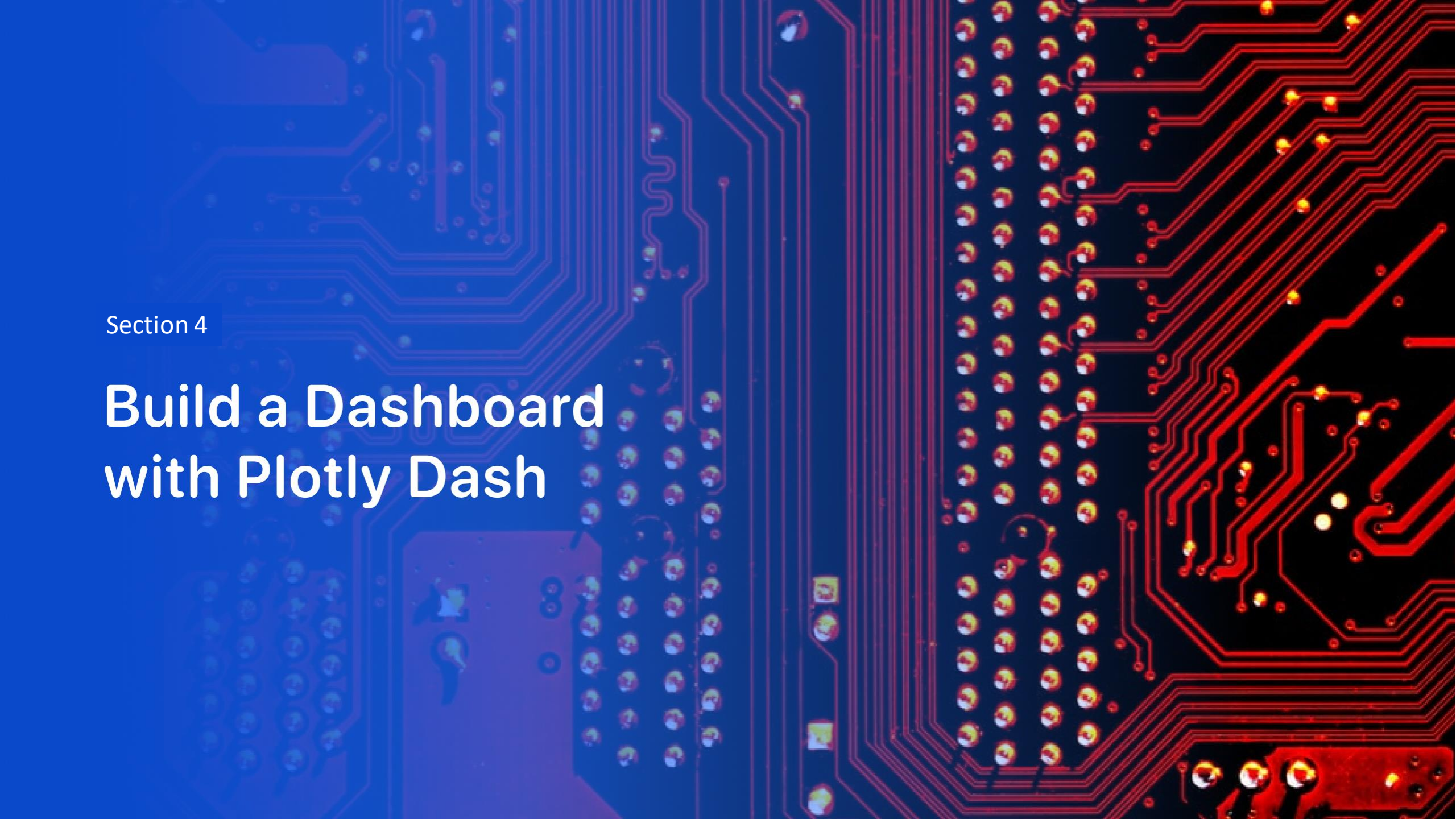


Distance to Coast (Blue)

Distance to Railway (Black)

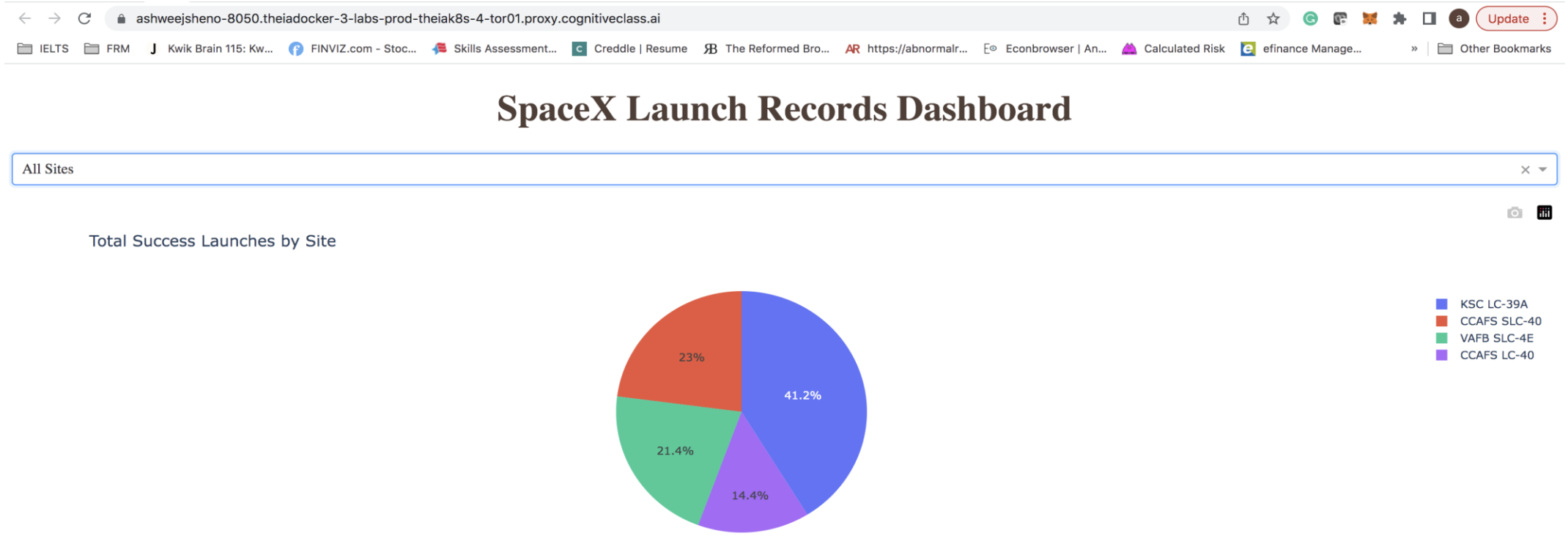


Distance to Orlando City (Red)


Section 4
Build a Dashboard with Plotly Dash

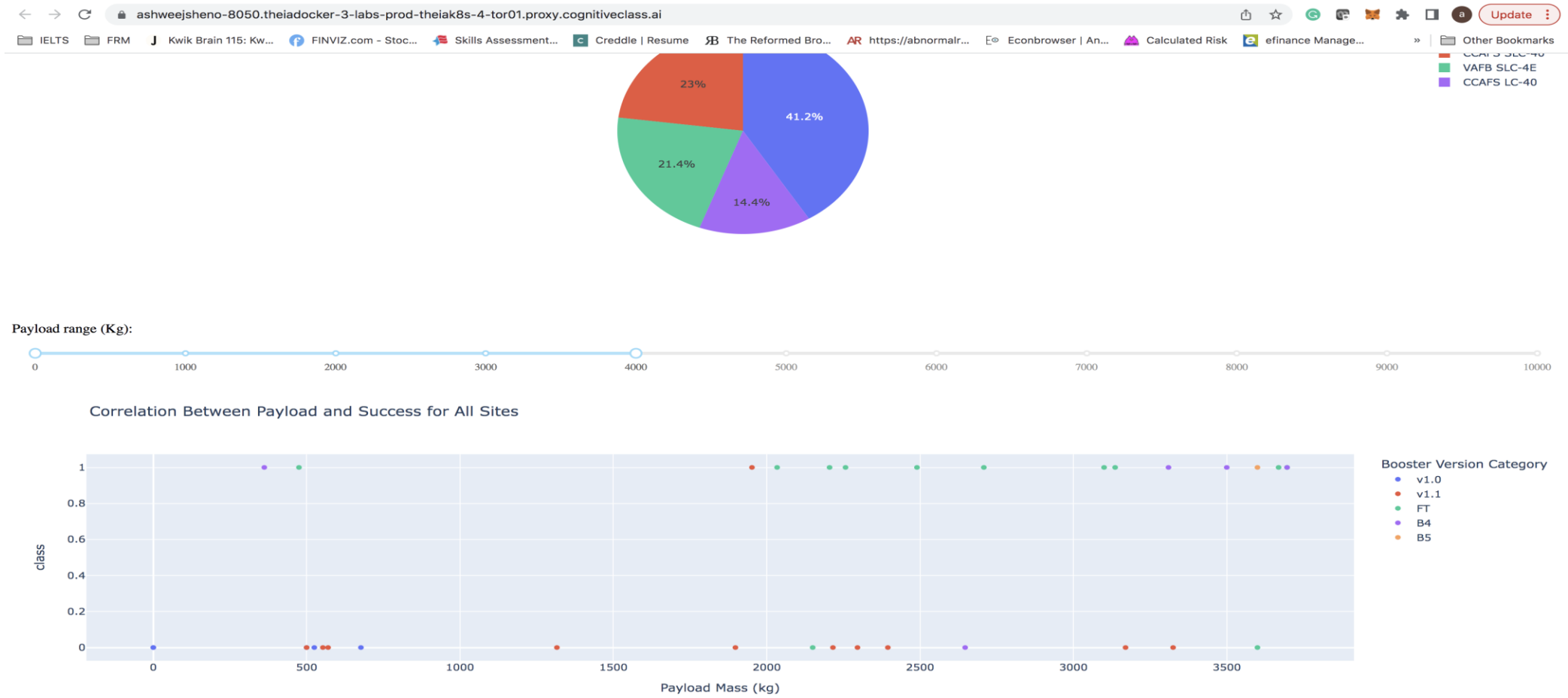
Launch Successes for All Launch Sites

KSC LC-39 A has most successful launches from all other sites



Payload vs. Launch Outcome scatter plot

Success for low payload range (<4000 kgs) is higher than high payload range



Observations from Dashboard

- Which site has the highest launch success rate?
  - **KSC LC-39 A**
- Which payload range(s) has the highest launch success rate?
  - **2000-10000 kg**
- Which payload range(s) has the lowest launch success rate?
  - **0- 1000kgs**
- Which F9 Booster version (v1.0, v1.1, FT, B4, B5, etc.) has the highest
  - **FT**

Section 5

Predictive Analysis (Classification)

Classification Accuracy

- All the models have same accuracy score on the test data set

	Accuracy Score	Jaccard Score	F1-Score
Logistic Regression	0.833333	0.8	0.888889
SVM	0.833333	0.8	0.888889
Decision Tree	0.833333	0.8	0.888889
KNN	0.833333	0.8	0.888889

- If the model is applied to entire data set, the accuracy score, Jaccard Score and F1 Score is as below

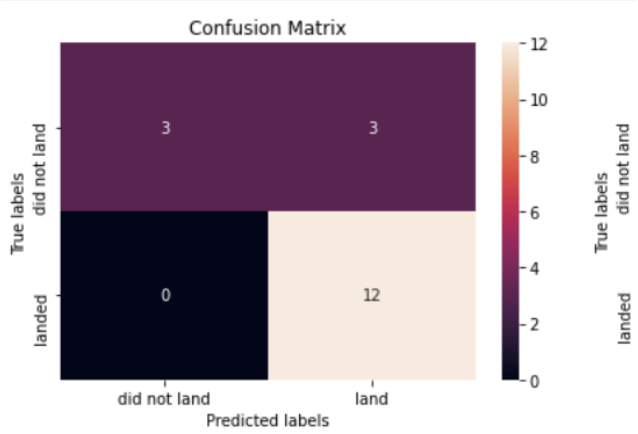
[36]:

	Accuracy Score	Jaccard Score	F1-Score
Logistic Regression	0.866667	0.833333	0.909091
SVM	0.877778	0.845070	0.916031
Decision Tree	0.844444	0.845070	0.916031
KNN	0.855556	0.819444	0.900763

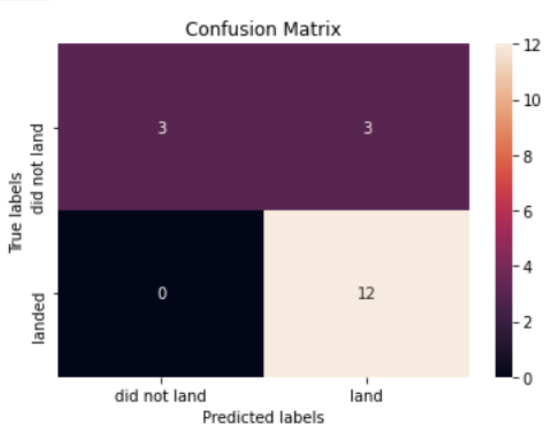
- SVM Model has slightly higher accuracy score, both SVM and Decision Tree have same F1 score

Confusion Matrix

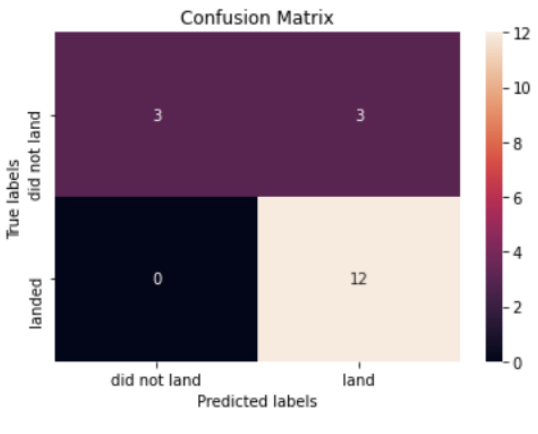
- All the models have the same confusion matrix



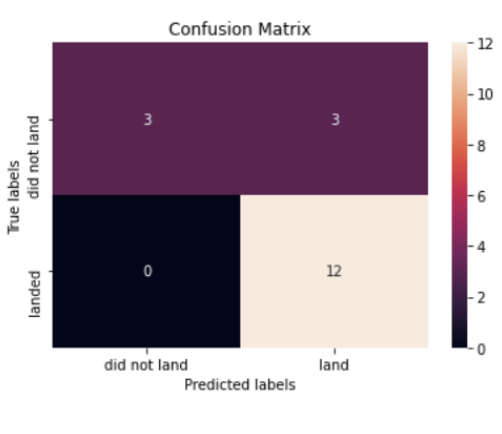
Logistic Regression



SVM



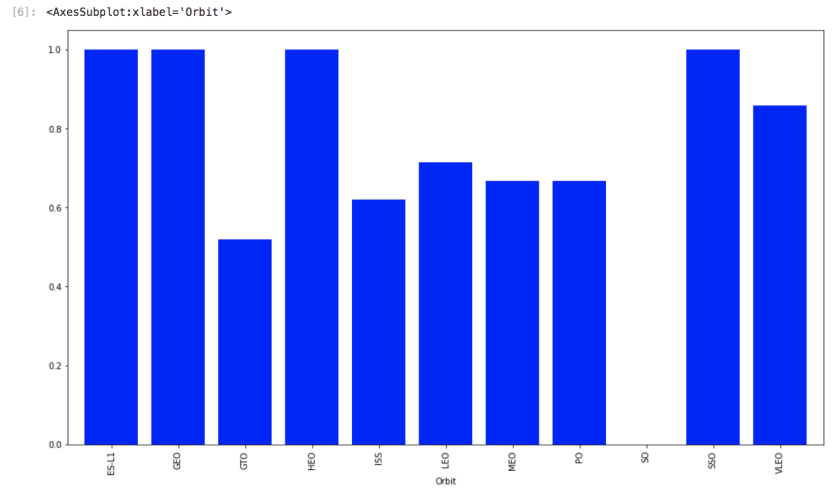
Decision Tree

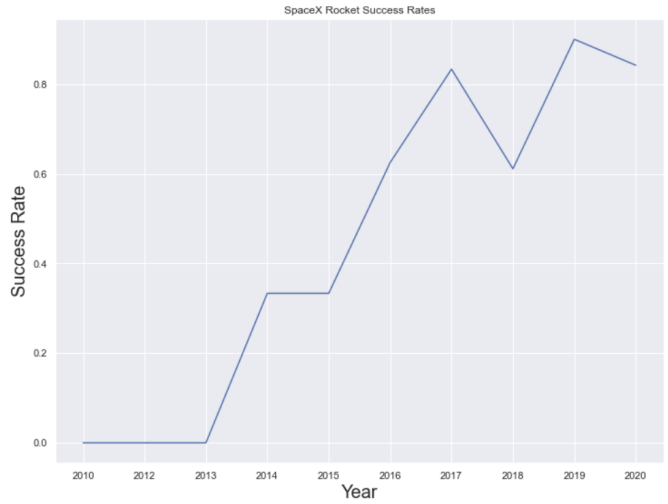


KNN

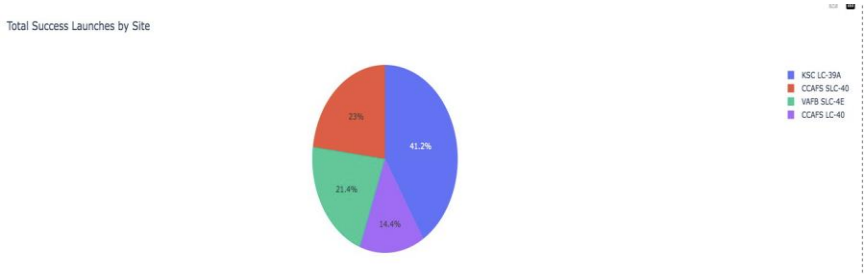
Conclusions

- Higher Success Rate for Orbits –ES-L1, GEO, HEO,SSO
- Success Rate of SpaceX launches have drastically improved after 2013
- KSC LC-39A has most successful launches,
- SVM Model is the best classifier for the given data set





	Accuracy Score	Jaccard Score	F1-Score
Logistic Regression	0.866667	0.833333	0.909091
SVM	0.877778	0.845070	0.916031
Decision Tree	0.844444	0.845070	0.916031
KNN	0.855556	0.819444	0.900763



Appendix

- Github URL : https://github.com/ashweej18/DataScience-Capstone-Project

Thank you!