IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Sravya Kakarlapudi
4th April 2025

Outline

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

Executive Summary

Summary of methodologies

- SpaceX Data Collection using SpaceX API
- SpaceX Data Collection with Web Scraping
- SpaceX Data Wrangling
- SpaceX Exploratory Data Analysis using SQL
- SpaceX EDA Data Visualization using Python Pandas and Matplotlib
- SpaceX Launch Sites Analysis with Folium-Interactive Visual Analytics and Plotly Dash
- Space X Machine Learning Landing Prediction

Summary of all results

- EDA Results
- Interactive Visual Analytics and Dashboards
- Predictive Analysis (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.
- Problems to find answers for
  - In this capstone, we will predict if the Falcon 9 first stage will land successfully using data from Falcon 9 rocket launches advertised on its website.

Section 1

Methodology

Methodology

- Data collection methodology:
  - Describes how data sets were collected
- Perform data wrangling
  - Describes how data was processed
- 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

Data was first collected by making a get request to the SpaceX API. Then, defined a series of helper functions that helped to use the API to extract information using identification numbers in the launch data. After that, requested rocket launch data from the SpaceX API url.

To make the requested JSON results more consistent, created a static response object and then decoded the response content as a Json and turned it into a Pandas DataFrame.

Also performed Web Scraping using BeautifulSoup to collect Falcon 9 historical launch records from a Wikipedia page titled List of Falcon 9 and Falcon Heavy launches. The records were stored in a HTML table which were then parsed to convert it into a Pandas data frame.

Data Collection – SpaceX API

- Data was collected using SpaceX API (a RESTful API) by making a Get Request to the SpaceX API. Requested and parsed the SpaceX launch data using the GET request and decoded the response content as a Json result which was then converted into a Pandas dataframe.
- Here is the GitHub URL of the completed SpaceX API calls notebook. Data Collection

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

To make the requested JSON results more consistent, we will use the following static response object for this project:

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

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

```
[10]: response=requests.get(static_json_url)
```

```
[11]: response.status_code
```

```
[11]: 200
```

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

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

Using the dataframe `data` print the first 5 rows

```
[13]:  nnb# Get the head of the dataframe
data.head()
```

Data Collection – Web Scraping

- Performed web scraping to collect Falcon 9 historical launch records from a Wikipedia page using BeautifulSoup. Extracted the Falcon 9 launch records from HTML table of the Wikipedia page which was parsed to convert it to a dataframe.
- Here is the GitHub URL of the completed web scraping notebook. [Web Scraping for data collection]

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

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

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

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

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

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

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

Data Wrangling

- After obtaining and creating a pandas dataframe from the collected data, data was filtered using the ***BoosterVersion*** column to only keep the Falcon 9 launches, then dealt with the missing data values in the ***LandingPad*** and ***PayloadMass*** columns. For the ***PayloadMass*** column, missing data was replaced with the mean value of the column.
- Also perfomed some Exploratory Data Analysis (EDA) to find some patterns in the data and determine what would be the label for training supervised models.
- Here is the GitHub URL for the completed data wrangling related notebooks. [Data Wrangling]

TASK 4: Create a landing outcome label from Outcome column

Using the `Outcome`, create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome`; otherwise, it's one. Then assign it to the variable `landing_class`:

In [13]:
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
df['Class'] = df['Outcome'].apply(lambda x: 0 if x in bad_outcomes else 1)
df['Class'].value_counts()
```

Out[13]:
```
Class
1    60
0    30
Name: count, dtype: int64
```

This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

In [15]:
```
landing_class = df['Class']
df[['Class']].head(8)
```

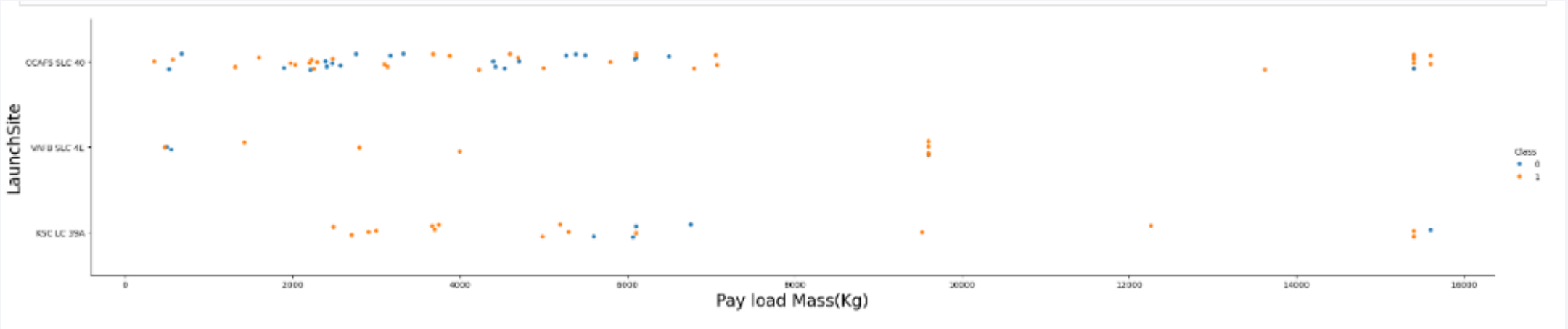
Out[15]:

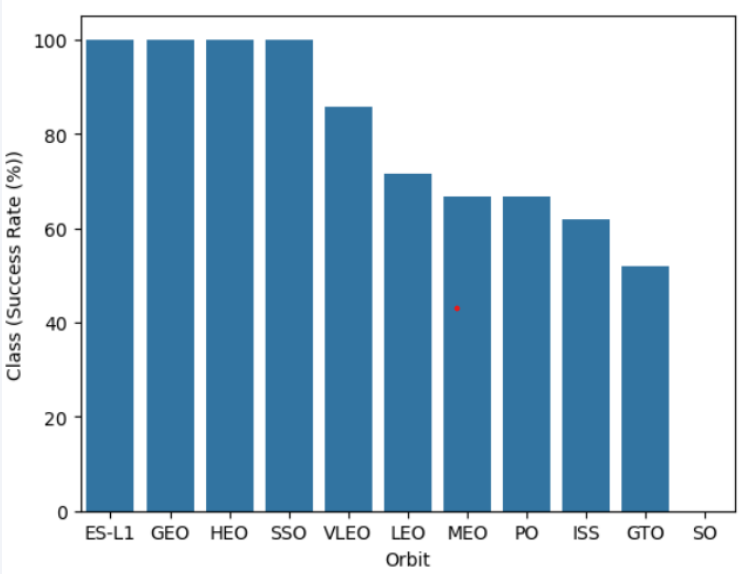
	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

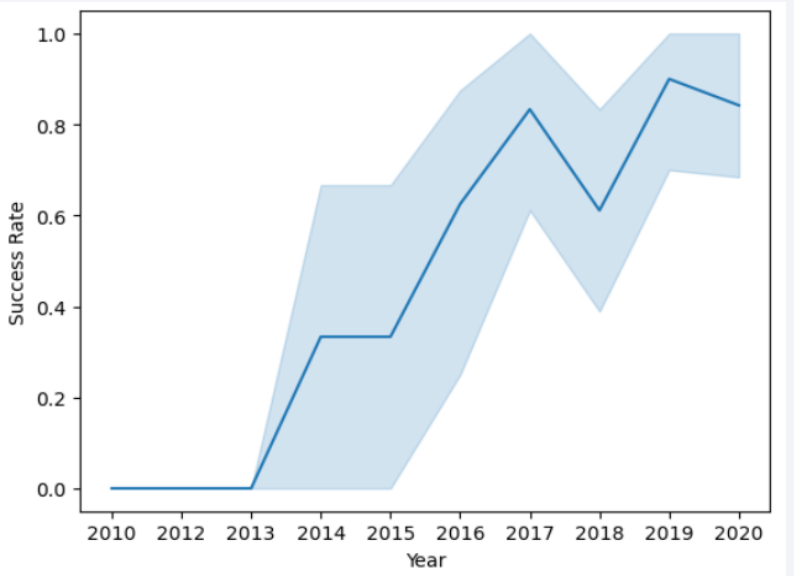
EDA with Data Visualization

- Performed data analysis and feature engineering using Pandas and Matplotlib.
- Used scatter plots to visualize the relationships between various features such as Orbit type, Payload Mass, Flight Number etc.
- Used bar chart to visualize the relationship between success rate of each orbit type.
- Used line plot to visualize the launch success yearly trend.
- Here is the GitHub URL of the completed EDA with data visualization notebook. Data Visualization

EDA with Data Visualization contd.. (Plots)







EDA with SQL

- The following SQL queries were performed for EDA:
- Display the names of unique launch sites in the space mission:

```
%sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;
```

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

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

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

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) as "Total Payload Mass(Kgs)", Customer FROM 'SPACEXTBL' WHERE Customer = 'NASA (CRS)';
```

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

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) as "Payload Mass Kgs", Customer, Booster_Version FROM 'SPACEXTBL' WHERE Booster_Version LIKE 'F9 v1.1%';
```


- Add the GitHub URL of your completed EDA with SQL notebook, as an

EDA with SQL contd..

- List the date when the first succesful landing outcome in ground pad was achieved:

```
%sql SELECT MIN(DATE) FROM 'SPACEXTBL' WHERE "Landing _Outcome" = "Success (ground pad)";
```

- List the total number of successful and failure mission outcomes

```
%sql SELECT "Mission_Outcome", COUNT("Mission_Outcome") as Total FROM SPACEXTBL GROUP BY "Mission_Outcome";
```

- 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

```
%%sql SELECT "Booster_Version", "Launch_Site" FROM SPACEXTABLE

WHERE "Landing_Outcome" = 'Failure (drone ship)' AND substr(Date,1,4) = '2015';
```

- Here is the GitHub URL of the completed EDA with SQL notebook. EDA with SQL

Build an Interactive Map with Folium

- Created Folium map to mark all the launch sites, and created map objects such as markers, circles, lines to mark the success or failure of launches for each launch site.
- Created a set of launch outcomes (failure=0 or success=1).
- Here is the GitHub URL of the completed interactive map with Folium map.

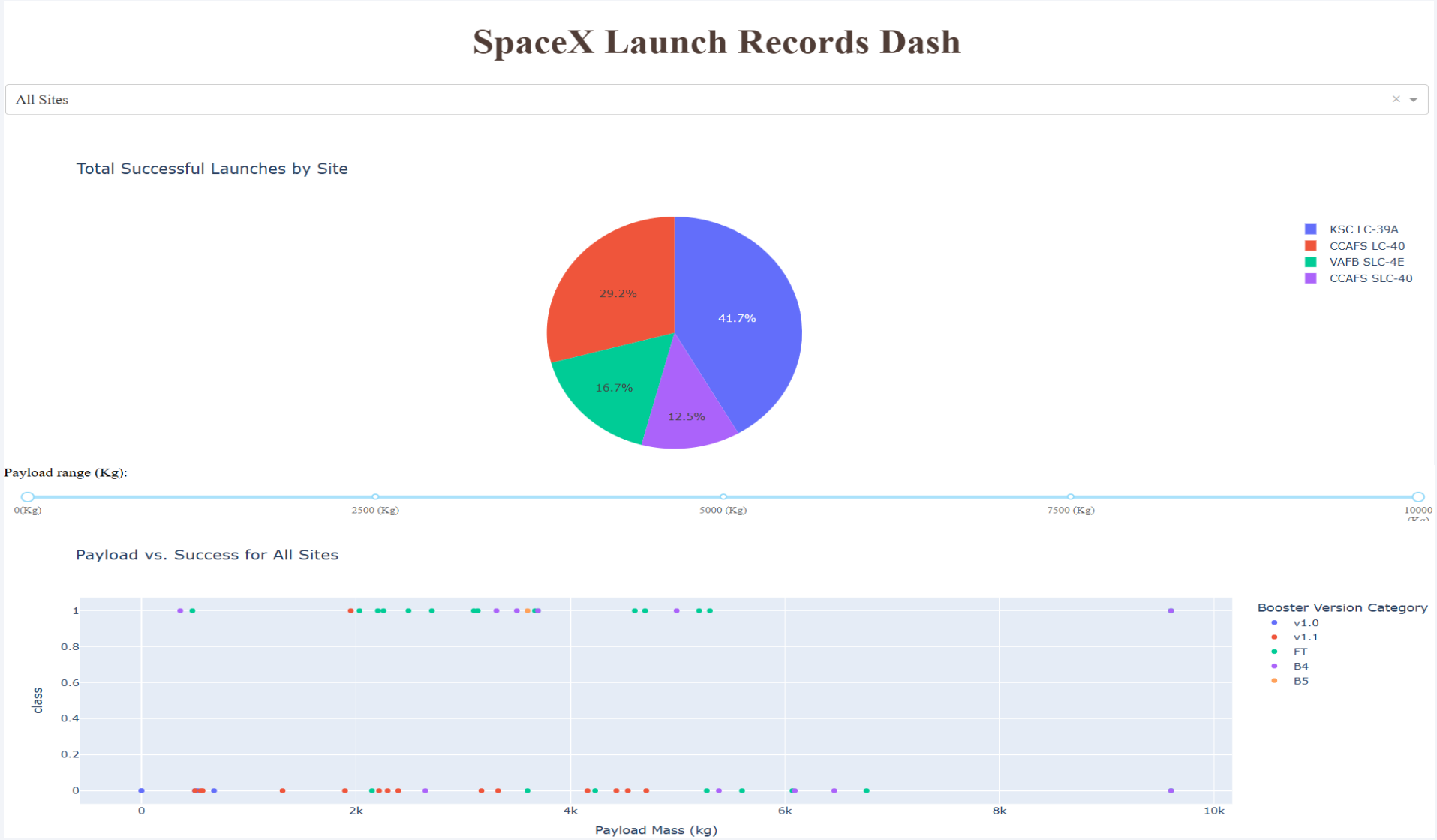
Interactive Map with Folium

Build a Dashboard with Plotly Dash

- Built an interactive dashboard application with Plotly Dash by:
  - Adding a Launch Site Drop-down input component.
  - Adding a callback function to render success-pie-chart based on selected site dropdown.
  - Adding a Range Slider to select payload.
  - Adding a callback function to render the success-payload-scatter-chart scatter plot.
- Here is the GitHub URL of the completed Plotly Dash lab.

  Dashboard with Plotly Dash

SpaceX Dash App



Predictive Analysis (Classification)

- After loading the data as a Pandas Dataframe, I set out to perform exploratory Data Analysis and determine training labels by;
  - Creating a Numpy array from the column 'Class' in data, by applying the method to_numpy(), then assigned it to the variable Y as the outcome variable.
  - Then standardized the feature dataset (X) by transforming it using preprocessing.StandardScaler() function from Sklearn.
  - After which the data was split into training and testing sets using the function 'train_test_split' from sklearn.model_selection with the test size parameter set to 0.2 i.e., 20% of the data would be test data.

Predictive Analysis (Classification) contd...

In order to find the best ML model that performs best with the test data i.e., between the models SVM, Decision Trees, K Nearest Neighbors and Logistic Regression;

- For each algorithm, I first created an object for that algorithm followed by creating a GridSearchCV object and then assigned them a set of parameters.
- For each model under evaluation, the GridSearchCV object was created with cv=10, then fit the training data to the GridSearchCV object to find the best Hyperparameters.
- I then displayed the best parameters by using the data attribute best_params_ and accuracy on the validation data using the data attribute best_score_.
- Finally, used the method 'score' to calculate the accuracy on the test data for each model and plotted a confusion matrix using the test and predicted outcomes of the respective model.

Predictive Analysis (Classification) contd...

- The table below shows the test data accuracy score for each of the models. This allows us to compare and decide which model performed better.

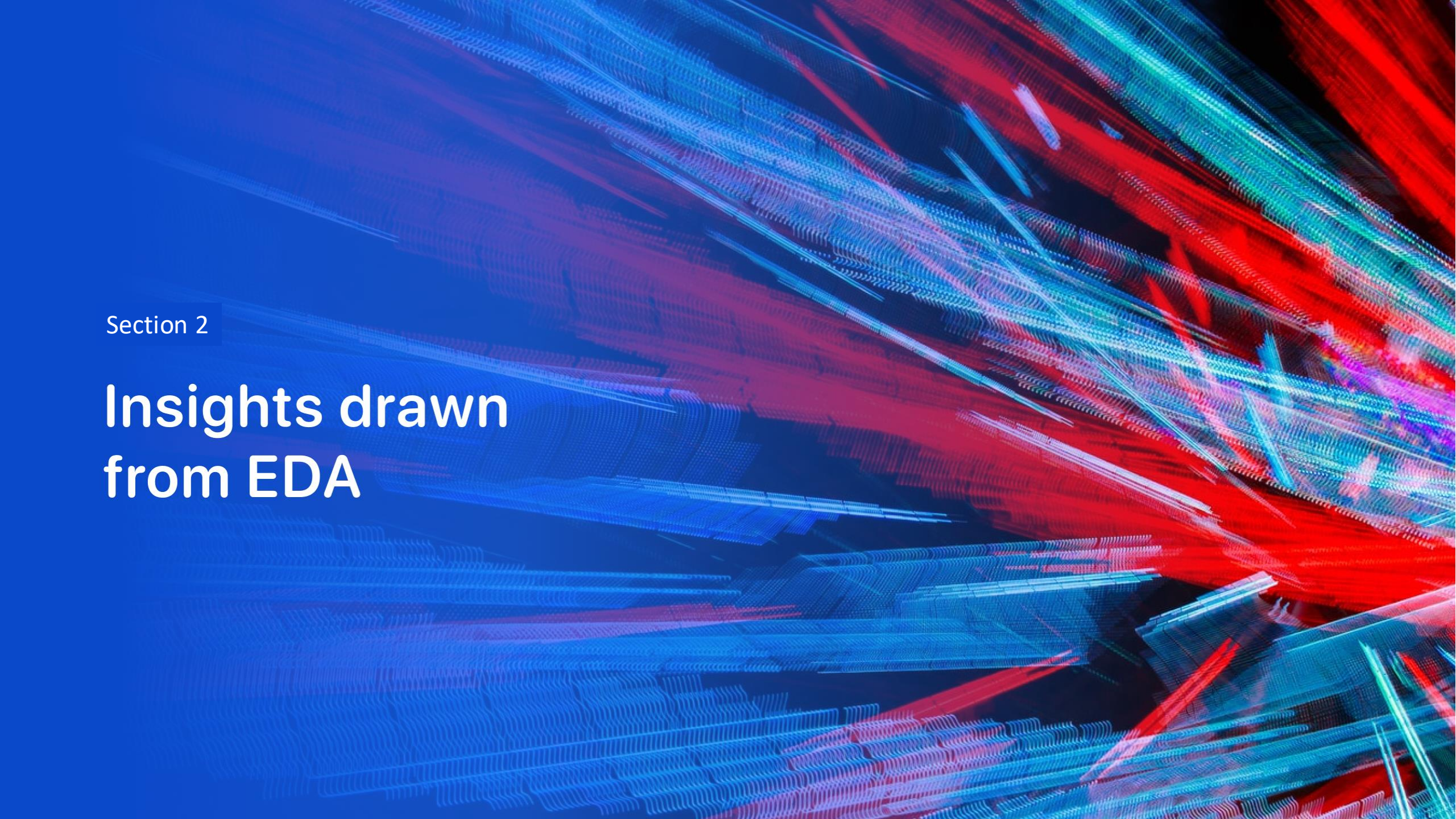
	Method	Test Data Accuracy
0	Logistic_Reg	0.833333
1	SVM	0.833333
2	Decision Tree	0.833333
3	KNN	0.833333

- Here is the Github Url of the completed predictive analysis notebook.

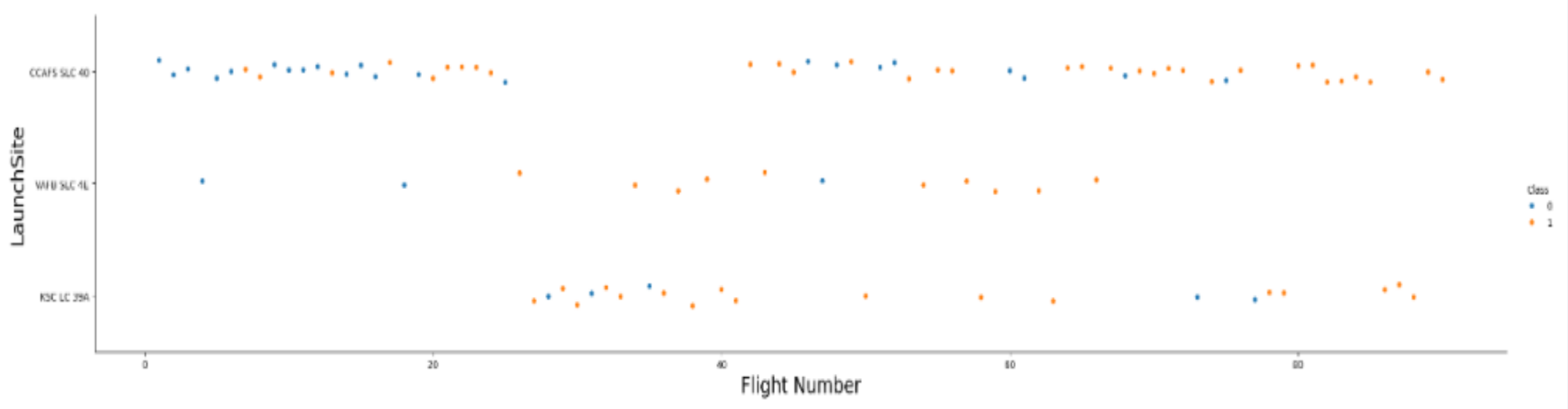
Predictive Analysis

Results

- Exploratory data analysis results
  - There is a correlation between launch site and success rate. Payload mass is also associated with the success rate. The more the payload mass, the less likely the first stage will return.
  - Majority of the mission outcomes have been successful.
- Interactive analytics demo in screenshots
- Predictive analysis results:
  - All the models performed with similar test data accuracy of 83.33%. Hence, we would need additional data for further research and analysis to improve the efficiency of each of these models and to determine which one outperforms the other.

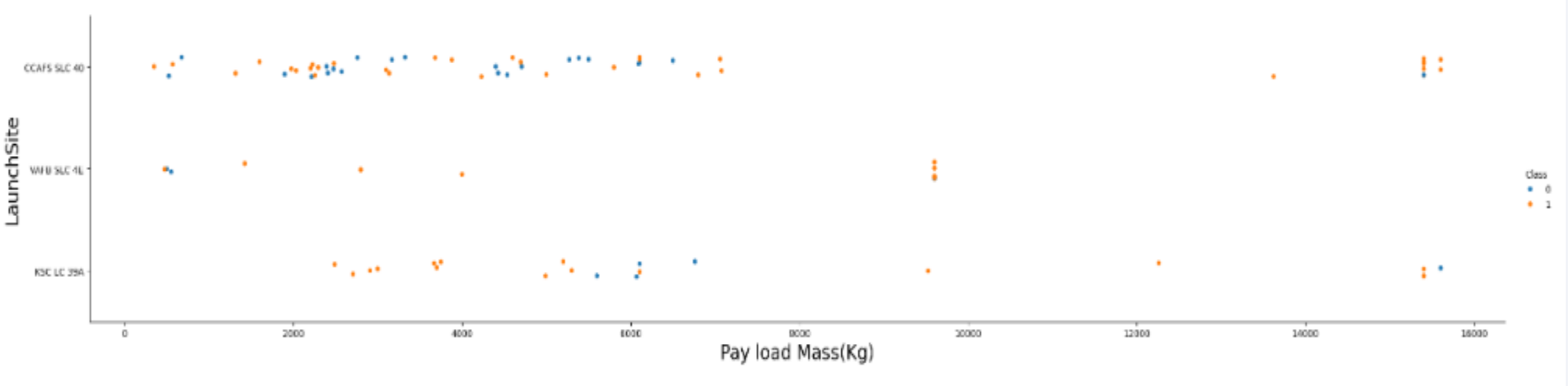

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



- We can deduce that as the flight number increases in each of the 3 launch sites, so does the success rate. For instance, the success rate for the VAFB SLC 4E launch site is 100% after the Flight number 50. Both KSC LC 39A and CCAFS SLC 40 have a 100% success rates after the 80th flight.

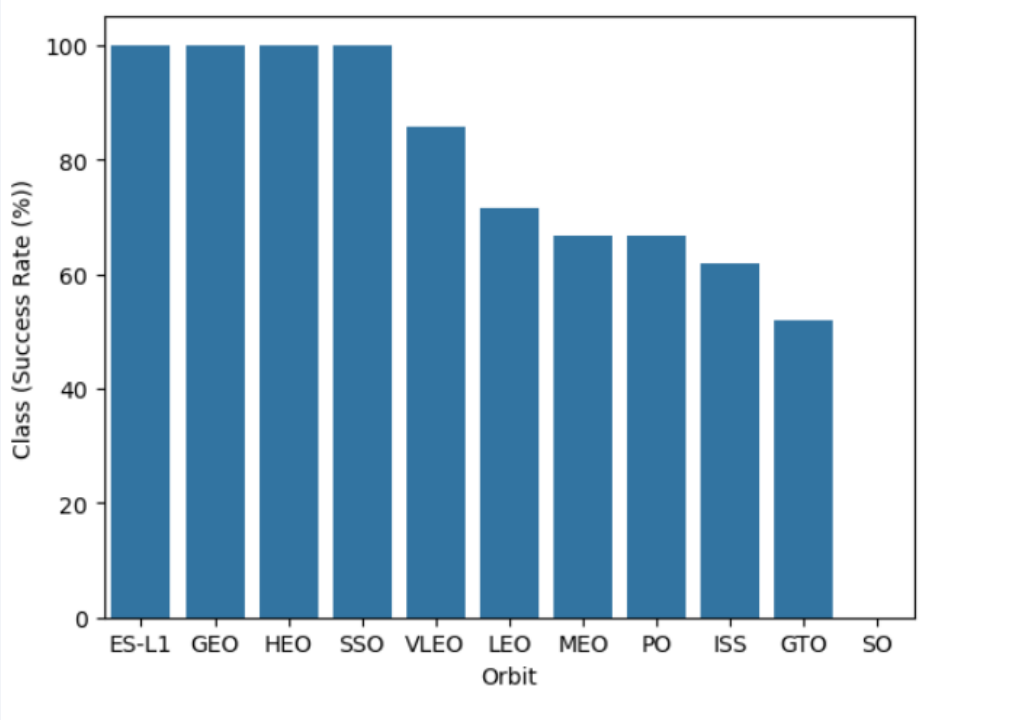
Payload vs. Launch Site



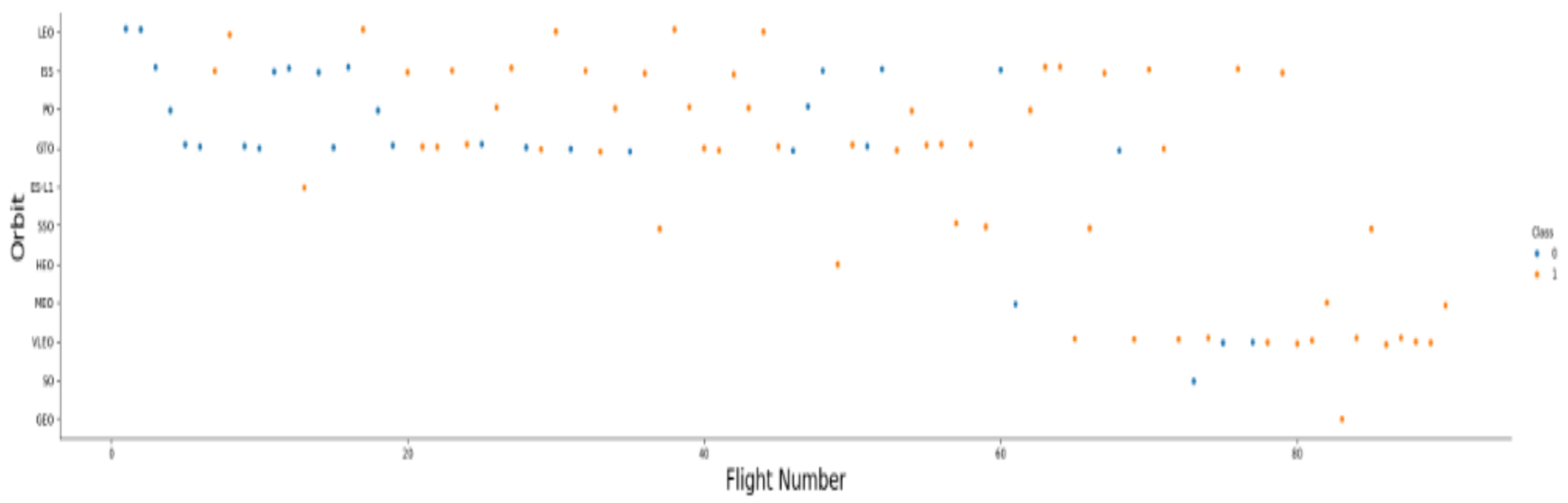
We can see that for the VAFB-SLC launch site there are no rockets launched for heavy payload mass(greater than 10000).

Success Rate vs. Orbit Type

- The orbit types that see the highest success rate are ES-L1, GEO, HEO and SSO. They have a 100% success rate. SO has the lowest success rate at 0%.

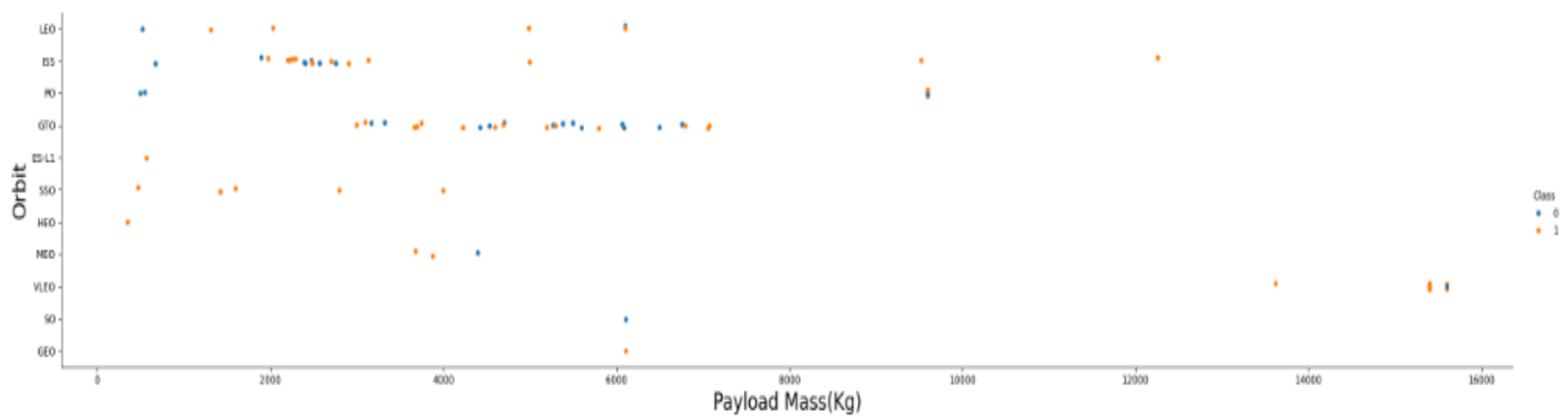


Flight Number vs. Orbit Type



- We can observe that in the LEO orbit, success seems to be related to the number of flights. Conversely, in the GTO orbit, there appears to be no relationship between flight number and success.

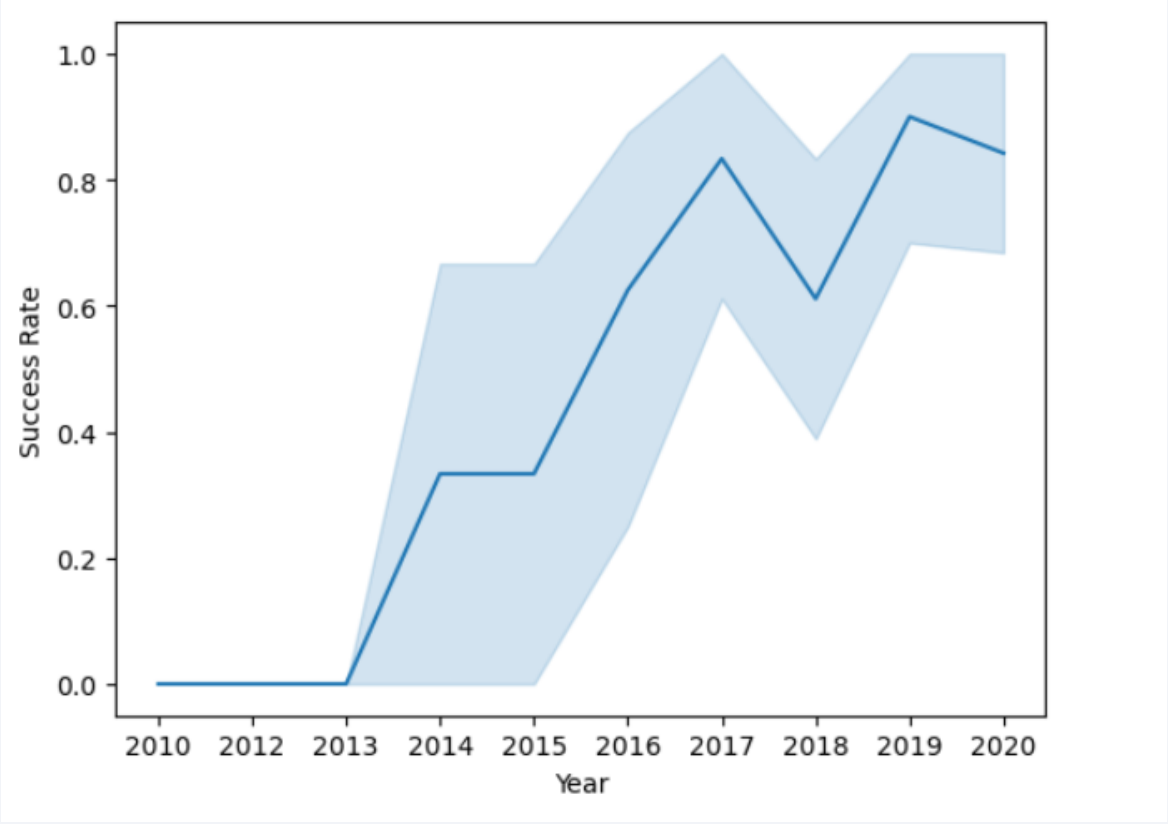
Payload vs. Orbit Type



- With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS.
- However, for GTO, it's difficult to distinguish between successful and unsuccessful landings as both outcomes are present.

Launch Success Yearly Trend

- We can observe that the success rate since 2013 kept increasing till 2020 although there was a small dip in 2018.



All Launch Site Names

- Find the names of the unique launch sites
- Used 'SELECT DISTINCT' statement to return only the unique launch sites from the 'LAUNCH_SITE' column of the SPACEXTBL table.

```
In [10]: %sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;
 * sqlite:///my_data1.db
Done.
```

Out[10]:

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

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`
- Used 'LIKE' command with '%' wildcard in 'WHERE' clause to select and display a table of 5 records using 'LIMIT', where launch sites begin with the string 'CCA'.

In [11]:
```
%sql SELECT * FROM 'SPACEXTBL' WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```

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

Out[11]:

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	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	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

Total Payload Mass

- Calculate the total payload carried by boosters from NASA
- Used the 'SUM' function to return and display the total sum of 'PAYLOAD_MASS_KG' column for customer 'NASA(CRS)'.

```
In [12]: %sql SELECT SUM(PAYLOAD_MASS__KG_) as "Total Payload Mass(Kgs)", Customer FROM 'SPACEXTBL' WHERE Customer = 'NASA (CRS)';
 * sqlite:///my_data1.db
Done.
```

Out[12]:

Total Payload Mass(Kgs)	Customer
45596	NASA (CRS)

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1
- Used the 'AVG' function to 'PAYLOAD_MASS__KG_' column to calculate the average, followed by a 'WHERE' clause to specify the booster version using 'LIKE' operator along with the '%' wildcard.

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) as "Payload Mass Kgs", Customer, Booster_Version FROM 'SPACEXTBL' WHERE Booster_Version LIKE 'F9 v1.1%';
```

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

Payload Mass Kgs	Customer	Booster_Version
2534.6666666666665	MDA	F9 v1.1 B1003

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad
- Used the 'MIN' function to the 'DATE' column from the SPACEXTBL table with a 'WHERE' clause of 'Landing_Outcome' = 'Success (ground pad)' to filter the results.

```
In [14]: %sql SELECT MIN(DATE) FROM 'SPACEXTBL' WHERE "Landing_Outcome" = "Success (ground pad)";
 * sqlite:///my_data1.db
Done.
Out[14]: MIN(DATE)
         2015-12-22
```

Successful Drone Ship Landing with Payload between 4000 and 6000

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000
- Used 'SELECT DISTINCT' on Booster Version and Payload columns from 'SPACXTBL' table with a 'WHERE' clause specifying 2 conditions. One for the Landing Outcome to be 'Success (drone ship)' and one for the 'Payload_Mass__kg_' to be greater than 4000 and less than 6000.

```
[15]: %sql SELECT DISTINCT Booster_Version, Payload FROM 'SPACEXTBL' WHERE Landing_Outcome = "Success (drone ship)"
          AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;
 * sqlite:///my_data1.db
Done.
```

[15]:

Booster_Version	Payload
F9 FT B1022	JCSAT-14
F9 FT B1026	JCSAT-16
F9 FT B1021.2	SES-10
F9 FT B1031.2	SES-11 / EchoStar 105

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes
- used the 'COUNT' function for the 'Mission_Outcome' column together with the 'GROUP BY' statement to get the total number of mission outcomes for each outcome.

```
[16]: %sql SELECT "Mission_Outcome", COUNT("Mission_Outcome") as Total FROM SPACEXTBL GROUP BY "Mission_Outcome";
 * sqlite:///my_data1.db
Done.
```

ut[16]:

Mission_Outcome	Total
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass
- Used a subquery to pass the 'MAX' function to the payload mass column and used it to list all the boosters that have carried the Max payload of 15600 kgs.

```
[17]: %sql SELECT "Booster_Version", Payload, "Payload_Mass__Kg_" FROM SPACEXTBL WHERE "Payload_Mass__Kg_"=
          (SELECT MAX("Payload_Mass__Kg_") FROM SPACEXTBL);
 * sqlite:///my_data1.db
Done.
```

[17]:

Booster_Version	Payload	PAYLOAD_MASS__KG_
F9 B5 B1048.4	Starlink 1 v1.0, SpaceX CRS-19	15600
F9 B5 B1049.4	Starlink 2 v1.0, Crew Dragon in-flight abort test	15600
F9 B5 B1051.3	Starlink 3 v1.0, Starlink 4 v1.0	15600
F9 B5 B1056.4	Starlink 4 v1.0, SpaceX CRS-20	15600
F9 B5 B1048.5	Starlink 5 v1.0, Starlink 6 v1.0	15600
F9 B5 B1051.4	Starlink 6 v1.0, Crew Dragon Demo-2	15600
F9 B5 B1049.5	Starlink 7 v1.0, Starlink 8 v1.0	15600
F9 B5 B1060.2	Starlink 11 v1.0, Starlink 12 v1.0	15600
F9 B5 B1058.3	Starlink 12 v1.0, Starlink 13 v1.0	15600
F9 B5 B1051.6	Starlink 13 v1.0, Starlink 14 v1.0	15600
F9 B5 B1060.3	Starlink 14 v1.0, GPS III-04	15600
F9 B5 B1049.7	Starlink 15 v1.0, SpaceX CRS-21	15600

2015 Launch Records

- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
- Used the 'substr' to get the year '2015' records along with the 'Landing_Outcome' set to 'Failure (drone ship)' in the 'WHERE' clause to get the failure landing outcomes in the year 2015.

In [42]:
```
%%sql SELECT "Booster_Version", "Launch_Site" FROM SPACEXTABLE

WHERE "Landing_Outcome" = 'Failure (drone ship)' AND substr(Date,1,4) = '2015';
```

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

Out[42]:

Booster_Version	Launch_Site
F9 v1.1 B1012	CCAFS LC-40
F9 v1.1 B1015	CCAFS LC-40

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

- 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
- Used 'Count' function on the landing outcomes along with a 'WHERE' clause to specify the dates using 'Between' and 'And' operators. Also gave 'order by' clause to list the outcomes in descending order.

In [44]:
```
%%sql SELECT "LANDING_OUTCOME", COUNT(*) as 'COUNT' FROM SPACEXTBL
WHERE substr(Date,1,4) || substr(Date,6,2) || substr(Date,9,2)
between '20100604' and '20170320' GROUP BY "Landing_Outcome" ORDER BY "COUNT" DESC;
```

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

Out[44]:

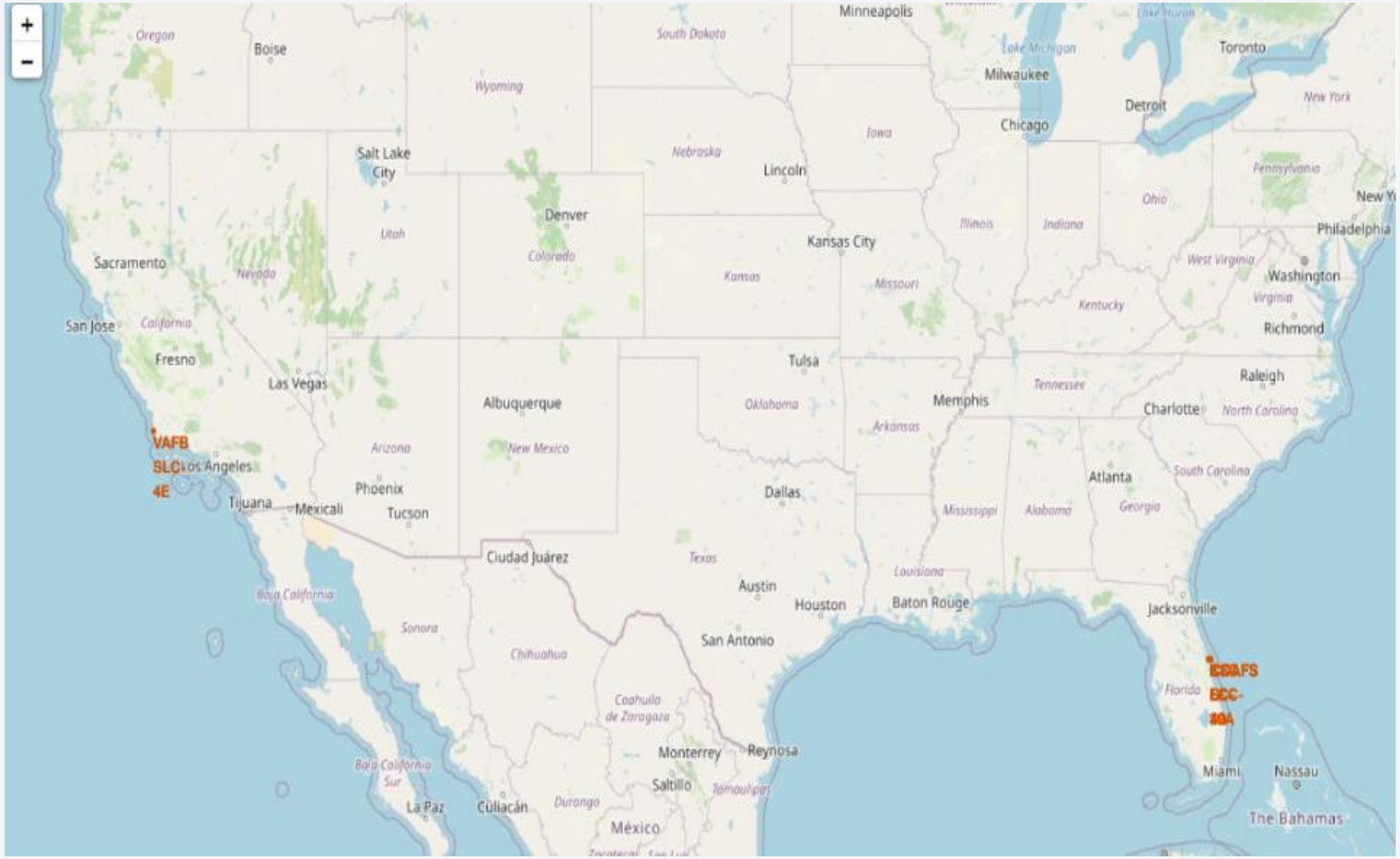
Landing_Outcome	COUNT
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

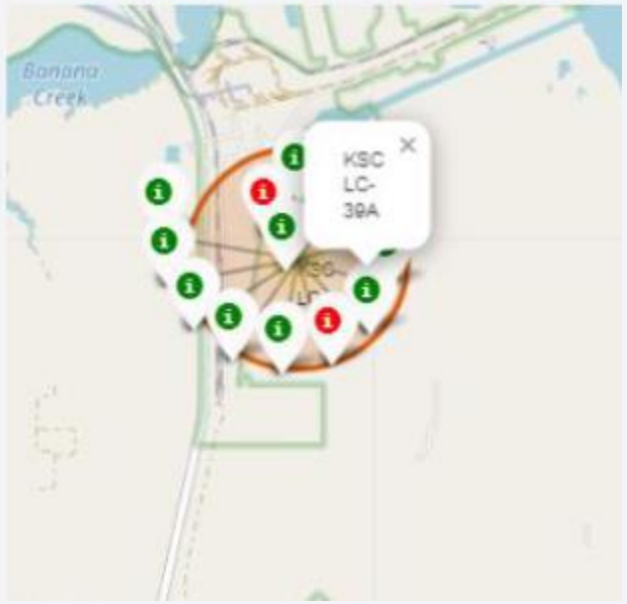
Launch Sites Proximities Analysis

Markers of All Launch Sites on Global Map

- All launch sites are in proximity to the equator line (loacted southwards of the US map).
- All launch sites are also in very close proximity to the coast.



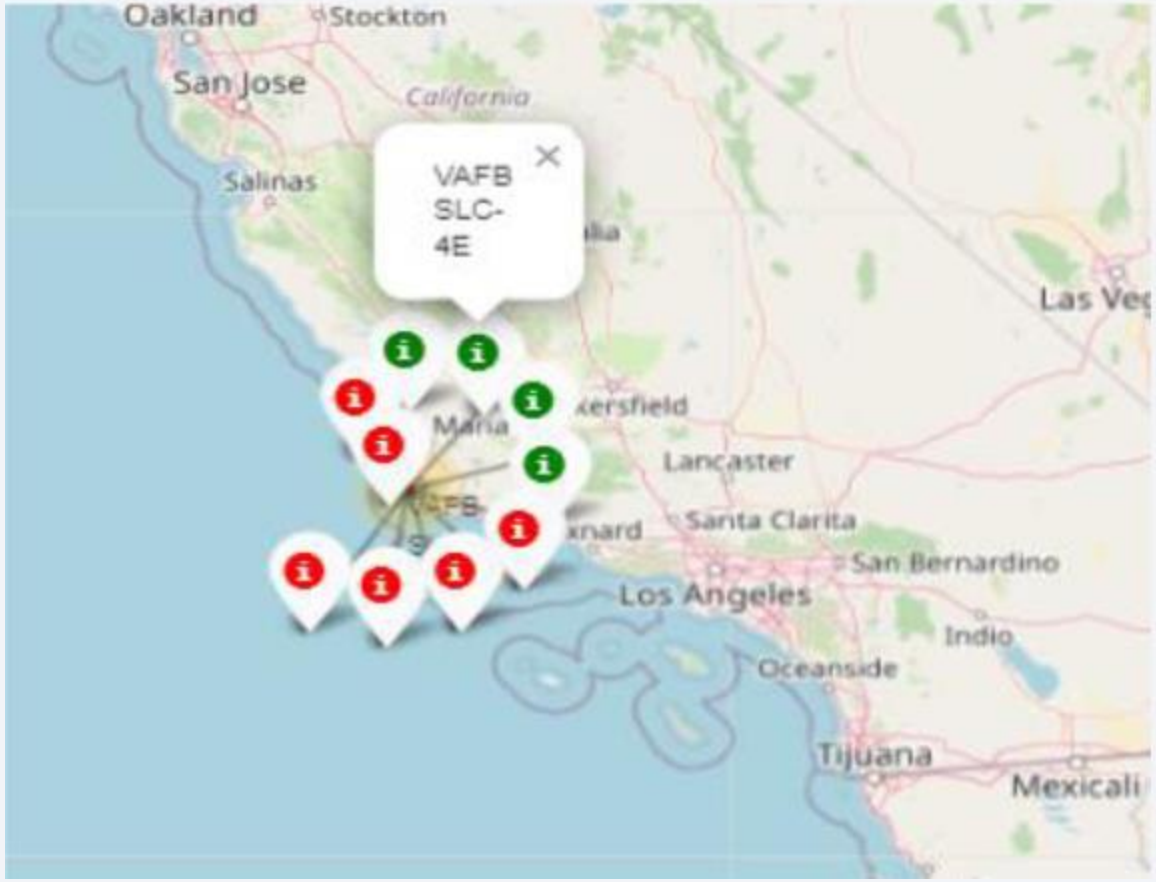
Launch Outcomes for each site on the Map



- In the Eastern coast (Florida), launch site KSC LCt-39A has relatively high success rates compared to CCAFS SCL-40 & CCAFS LC-40.

Launch Outcomes for each site on the Map

- In the west coast (California), launch site VAFB SLC-4E has relatively lower success rates than the KSC LC-39A launch site in the eastern coast (Florida).

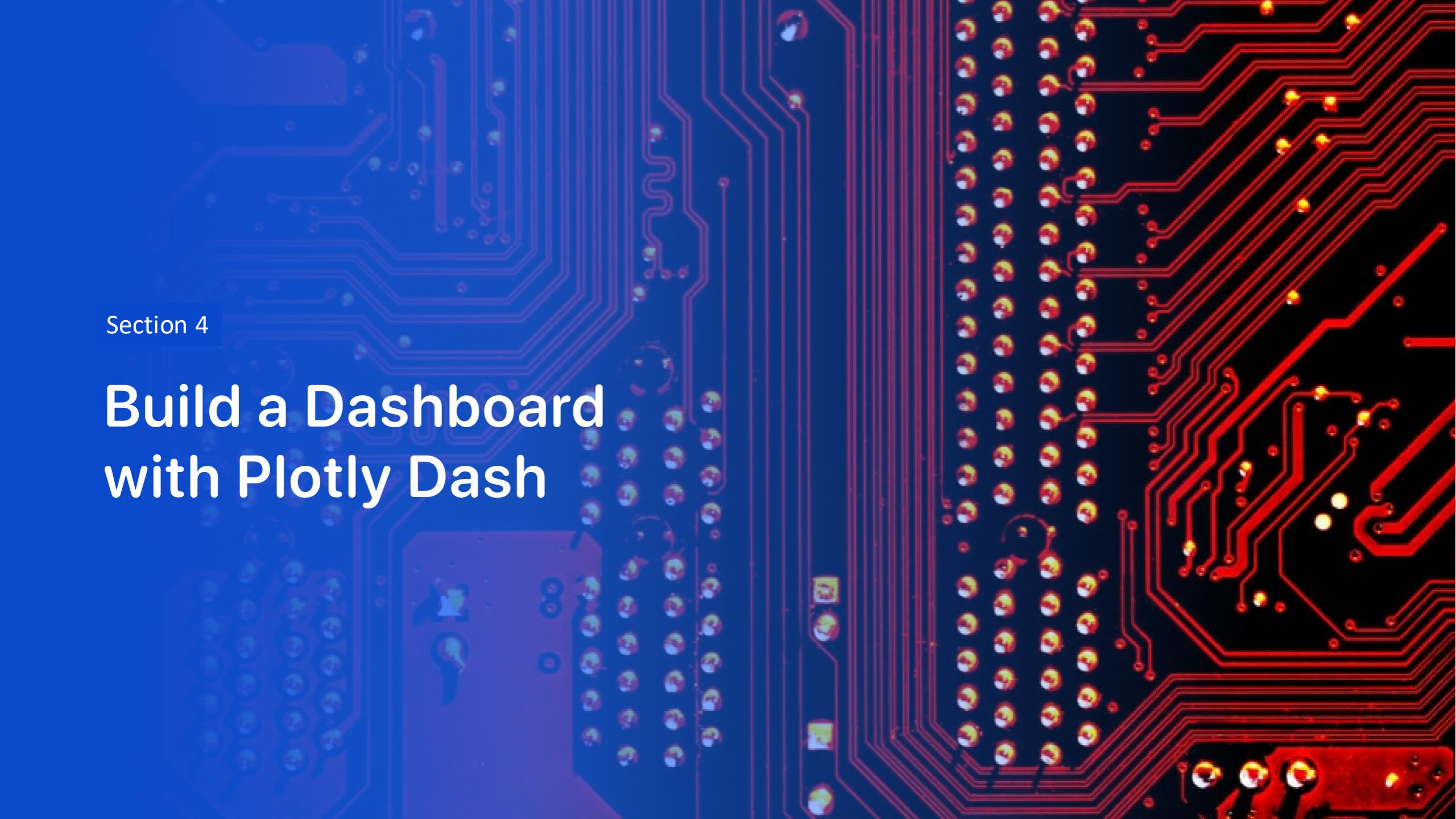


Distance between a launch site and the coastline.

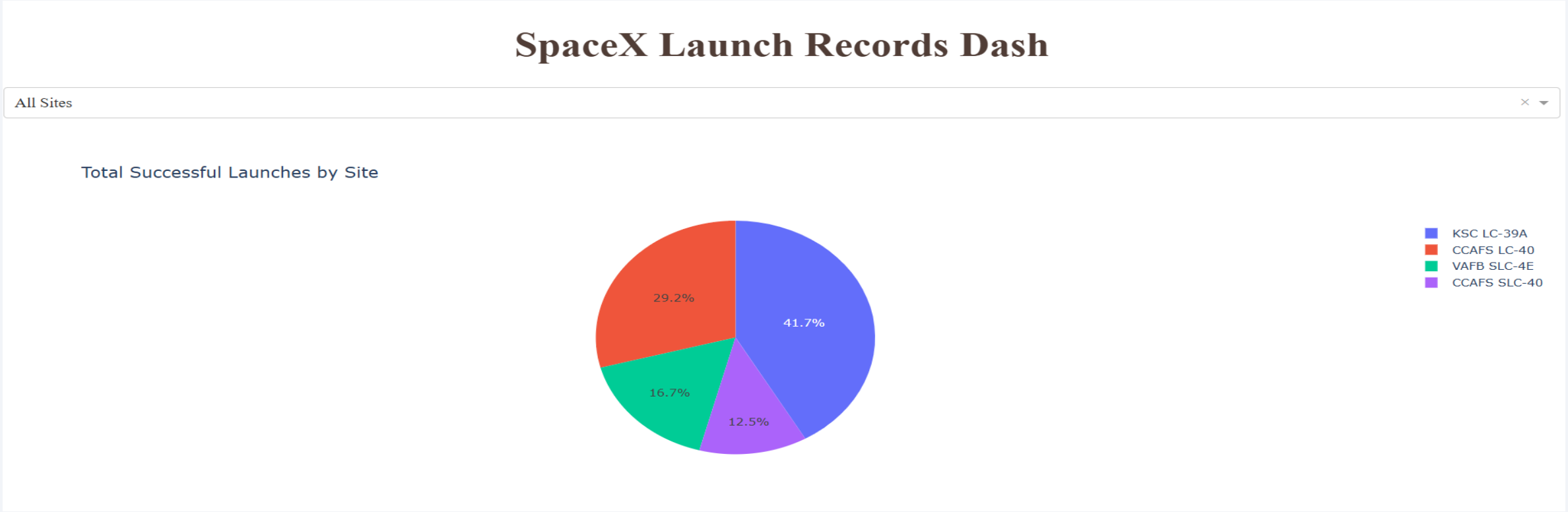
- Launch site CCAFS SLC-40 proximity to coastline is 0.86km.

Distance between a launch site and its proximities

- Launch site CCAFS SLC-40 closest to highway (Washington Avenue) is 23.19km.

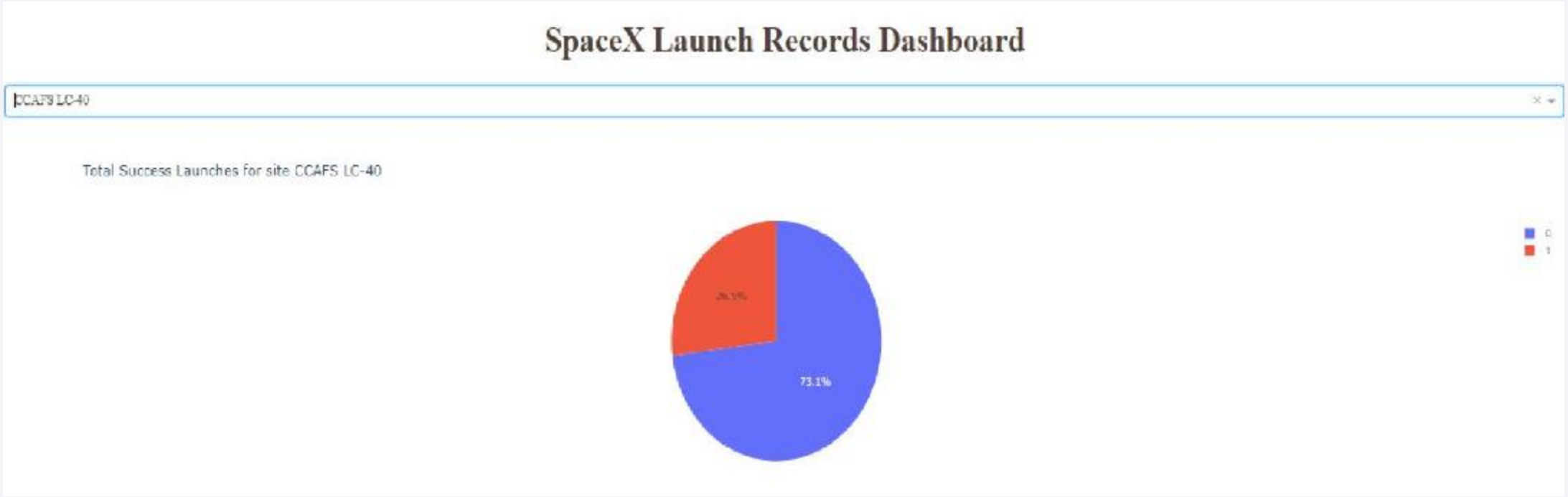

Section 4
Build a Dashboard with Plotly Dash

Pie-Chart for launch success count for all sites



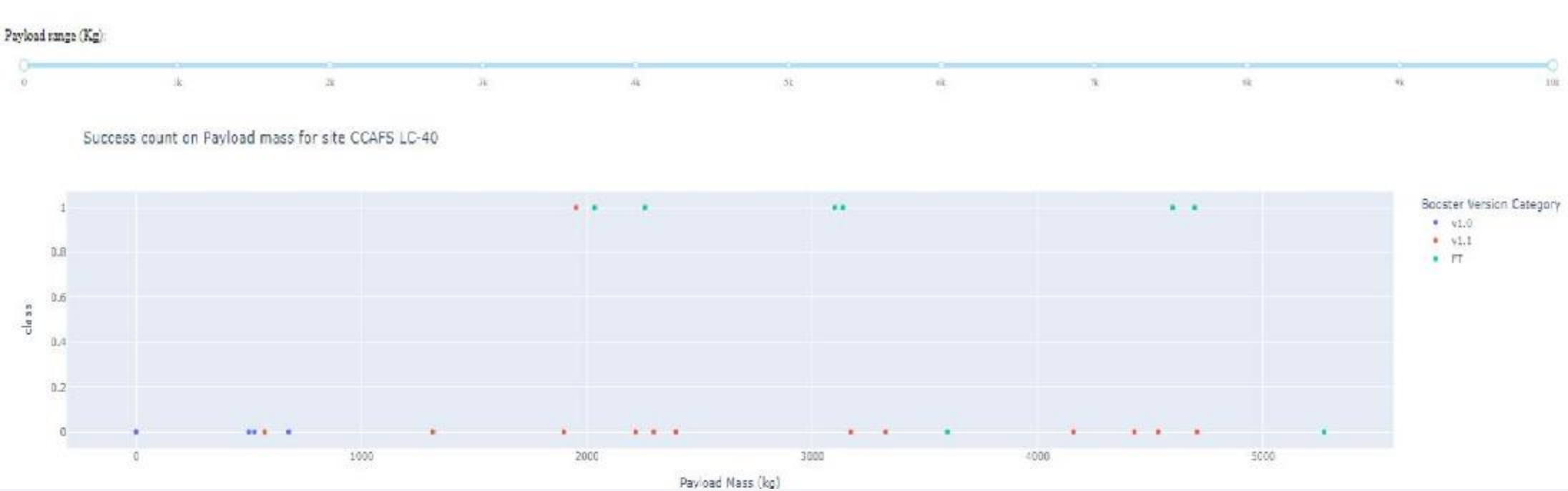
- Launch site KSC LC-39A has the highest launch success rate at 42% followed by CCAFS LC-40 at 29%, VAFB SLC-4E at 17% and lastly CCAFS SLC-40 with a success rate of 13%.

Pie Chart for the launch site with 2nd highest launch success ratio



- Launch site CCAFS LC-40 had the 2nd highest success ratio of 73% against 27% of failed launches.

Payload vs Launch Outcome scatter plot for all sites



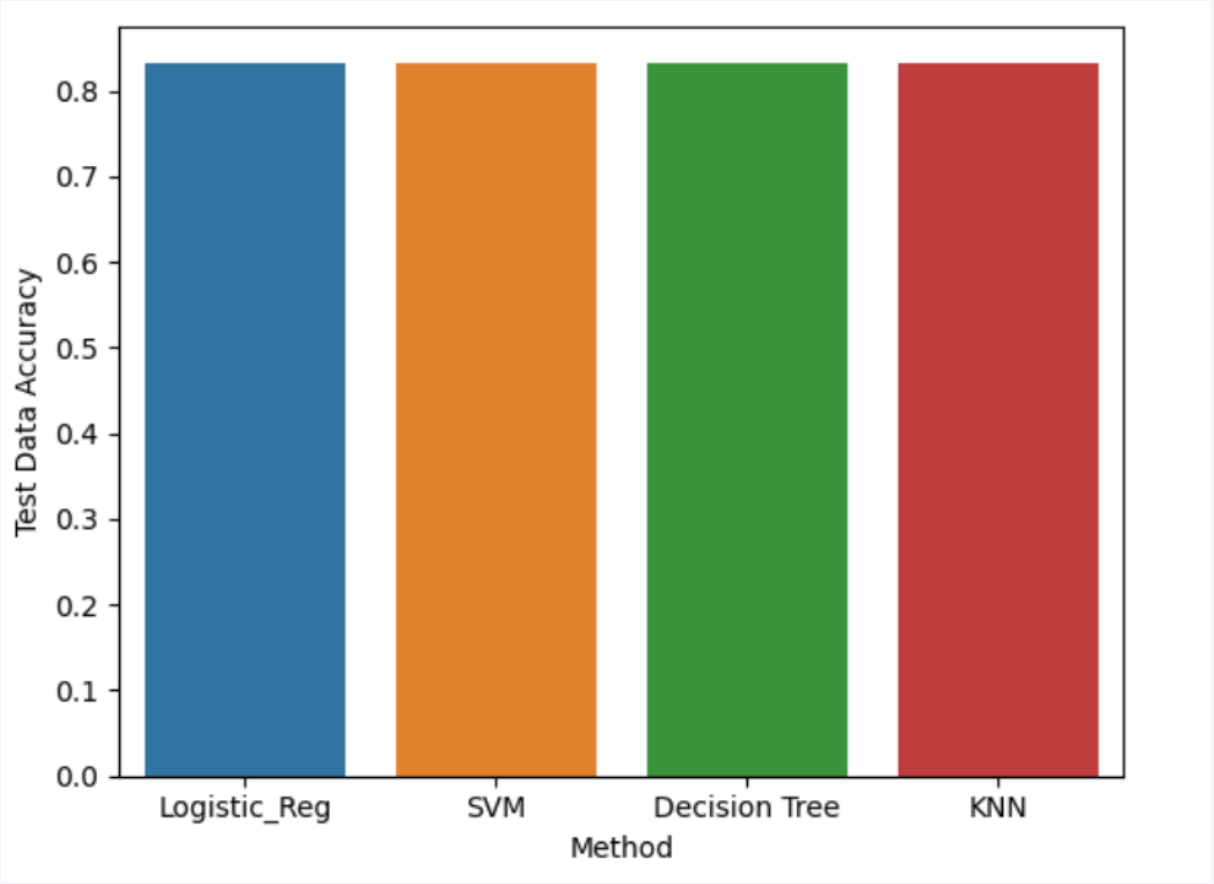
- For launch site CCAFS LC-40 the booster version FT has the largest success rate from a payload mass of >2000kg.

Section 5

Predictive Analysis (Classification)

Classification Models Accuracy

- All models perform equally on the test data. They all have the same accuracy of 83.33%.

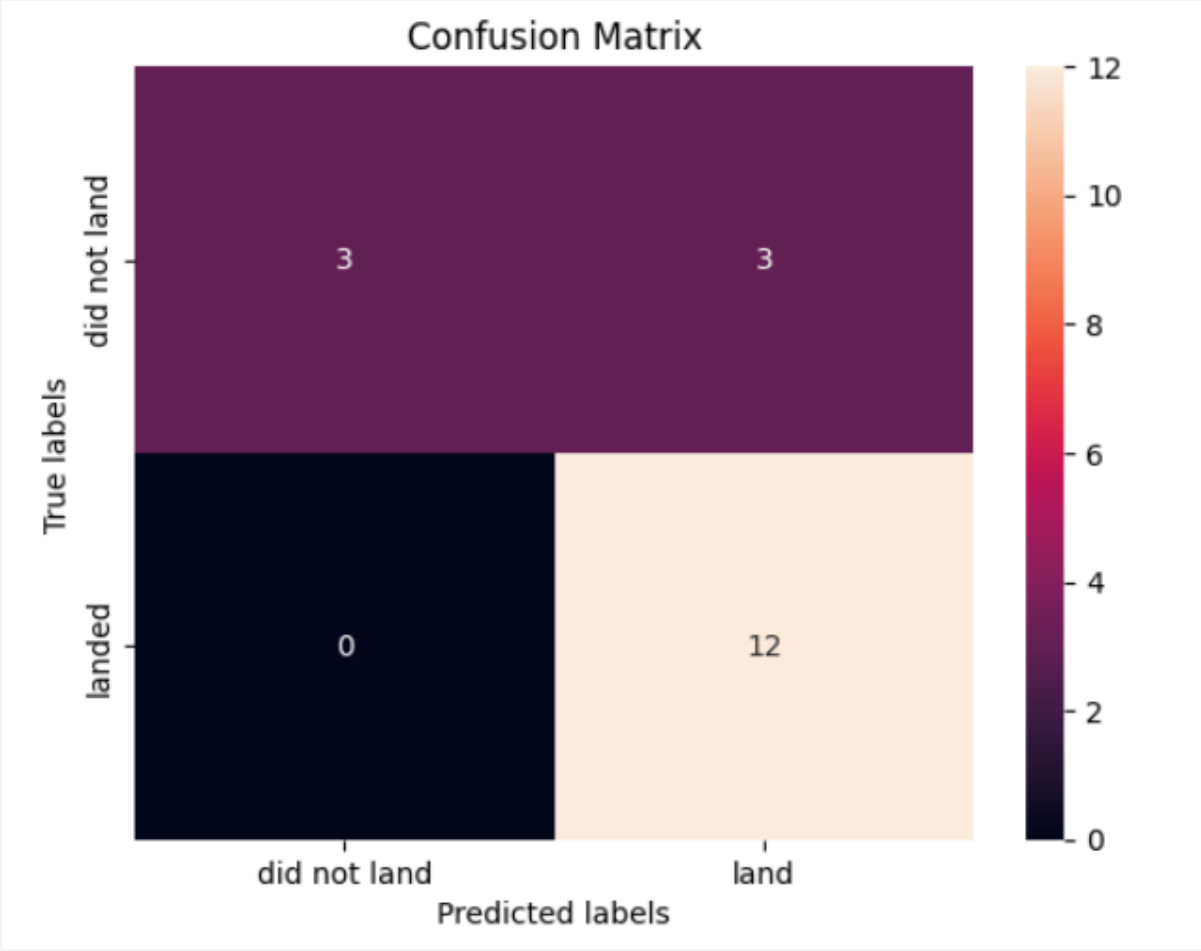


Confusion Matrix

- All the 4 classification models had the same confusion matrixes and were equally able to distinguish between different classes.

- The major problem is the False Positives for all the models.

- False Positive - 3 (True label is not landed, Predicted label is landed)



Conclusions

- Different launch sites have different success rates. CCAFS LC-40 has highest success rate of 60% while KSC LC-39A and VAFB SLC 4E have a success rate of 77%.
- We can deduce that, as the flight number increases in each of the 3 launch sites, so does the success rate.
- If we observe payload vs launch site scatter point chart, we find that for the VAFB SLC launch site there are no rockets launched for heavy payload mass (greater than 10000kg).
- Orbits ES L1, GEO, HEO & SSO have the highest success rates at 100%, with SO orbit having the lowest success rate at 0%.
- For the LEO orbit, success appears to be related to the number of flights; on the other hand, there seems to be no relationship between flight number and launch success when in GTO orbit.

Conclusions contd..

- With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS. However for GTO we cannot distinguish this well because both positive landing rate and negative landing rate (unsuccessful mission) are existing.
- And finally the success rate since 2013 kept increasing till 2020.

Appendix

- Github Url for the Notebooks - https://github.com/sravyaksn/IBM-Capstone-Project---SpaceX

Thank you!