IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Yuxue Zhou

September 11th, 2022

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection Using API, Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis (EDA) with SQL
  - EDA with Data Visualization
  - Interactive Visualization with Folium
  - Machine Learning Prediction
- Summary of all results
  - Exploratory Data Analysis Results
  - Interactive Maps and Dashboard
  - Predictive Results

Introduction

- Project background and context
  - This project is aimed to predict if the Falcon 9 first stage will land successful. By accomplish this, we need to collect data, wrangle the data, analyze data and create data visualization for easier observation, and use machine learning to predict the success rate of Falcon 9 first stage landing
  - 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 you want to find answers
  - What are some affecting factors to the success rate
  - What machine learning algorithm should we use to predict successful landing?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Describe how data was collected
- Perform data wrangling
  - Describe 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 sets were collected through two different ways, SpaceX API and web scrap Wikipedia page
  - SpaceX REST API:
    - We first make a SpaceX REST API call to request information using GET request
    - API returns a JSON file then normalize the JSON file and create Pandas data frame
    - Filter and clean data to only what we need
    - Replace missing payload mass data with calculated mean
  - Webscraping:
    - Request Falcon9 launch data from Wiki page using its URL by performing HTTP GET method as an HTTP response
    - Using BeautifulSoup from HTML response to extract data
    - Create a data frame by parsing the launch HTML table

Data Collection – SpaceX API

1. Request rocket launch data from SpaceX REST API

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

```
response = requests.get(spacex_url)
```

2. Normalize JSON to data frame

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

3. Get data

```
getBoosterVersion(data)
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
```

4. Combine into a dictionary

```
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}
```

5. Create Pandas data frame From the dictionary

```
dict_df = pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
```

Notebook

Data Collection - Webscraping

1. HTTP GET method to get HTTP response from Wiki

```
response = requests.get(static_url)
```

2. Create an object of BeautifulSoup

```
soup = BeautifulSoup(response.content, 'html.parser')
```

3. Find all tables and select the table needed

```
html_tables = soup.find_all('table')
first_launch_table = html_tables[2]
```

4. Extract Column Names

```
for row in first_launch_table.find_all('th'):
    name = extract_column_from_header(row)
    if(name != None and len(name) > 0):
        column_names.append(name)
```

5. Create Dictionary

```
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']=[]
```

6. Create data frame from dictionary

```
df = pd.DataFrame.from_dict(launch_dict, orient='index')
df = df.transpose()
```

Notebook

Data Wrangling

- Explore data to determine the label for training supervised models
  - Calculate number of launches on each site
  - Calculate number and occurrence of each orbit
  - Calculate number and occurrence of mission outcome per orbit type
- Create landing outcome label by convert string variables into categorical variables with 1 meaning land successfully 0 otherwise.
  - Class = 1; landed successfully
    - True Ocean - successfully landed to a specific region of the ocean
    - True RTLS - successfully landed to a ground pad
    - True ASDS - successfully landed to a drone ship
  - Class = 0; did not land successfully
    - None ASDS - represent a failure to land
    - None None - represent a failure to land
    - False Ocean - unsuccessfully landed to a specific region of the ocean
    - False RTLS - unsuccessfully landed to a ground pad
    - False ASDS - unsuccessfully landed to a drone ship
- Notebook

1. Calculate the number of launches on each site

```
df['LaunchSite'].value_counts()
```

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

2. Calculate number and occurrence of each orbit

```
df["Orbit"].value_counts()
```

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

3. Calculate number and occurrence of mission outcome per orbit type

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

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

4. Create a landing outcome label from Outcome column

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

```
0 True ASDS
1 None None
2 True RTLS
3 False ASDS
4 True Ocean
5 False Ocean
6 None ASDS
7 False RTLS
```

```
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 = []

for outcome in df['Outcome']:
    if outcome in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)

df['Class']=landing_class
```

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

lumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
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
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
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
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
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

EDA with Data Visualization

- **Scatter plots**: shows relationship between variables and non-linear patterns; easy to observe
  - Flight Number vs. Launch Site
  - Payload vs. Launch Site
  - FlightNumber vs. Orbit type
  - Payload vs. Orbit type

- **Bar chart**: shows relationships between numeric and categorical variables
  - Success Rate vs Orbit Type

- **Line graph**: shows changes and trends over time
  - Year vs. Class

- Notebook

EDA with SQL

- The following queries were performed
  - 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 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. 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

- Circle, marker, marker cluster with color coded which red means fail and green means success, poly lines were added to the site map
  - Highlighted circle area with text lable on each launch site on the site map
  - Mark the success/failed launches for each site on the map with green for successful landing and red for failed landing
  - Calculate the distances between a launch site to its proximities (city, railroad, highway, and coastline)
- By adding these objects to the map, we can easily observe important messages such as that launch sites are close to the coastline, railroad, and highways, number of successful and failed landing at each launch site.
- Notebook

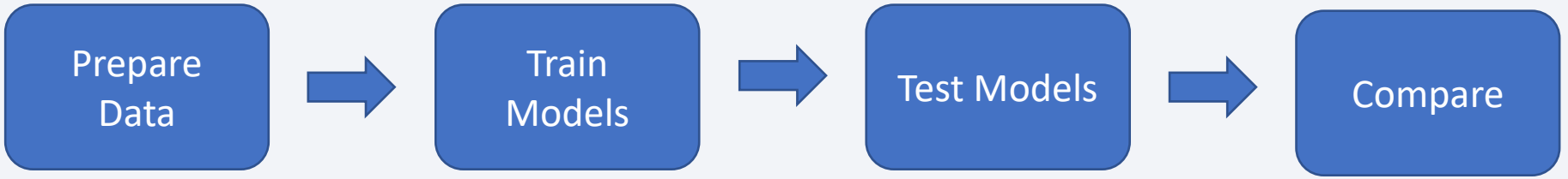
Build a Dashboard with Plotly Dash

- The dashboard has a drop-down list, a pie chart, a range slider, and a scatter plot
  - The drop-down list allows user to select certain or all launch site as needed
  - The pie chart shows successful count for each and all launch sites with percentage on the chart. If mouse is hover over, it shows launch site name and successful count. The legend is on the right to indicate the color and corresponding launch site
  - The range slider is there so that user can slide to select a range of payload mass. The scatter plot changes accordingly
  - The scatter plot shows the relationship between payload mass and successful count for each site. Each data point is color coded according to the booster version

- Notebook

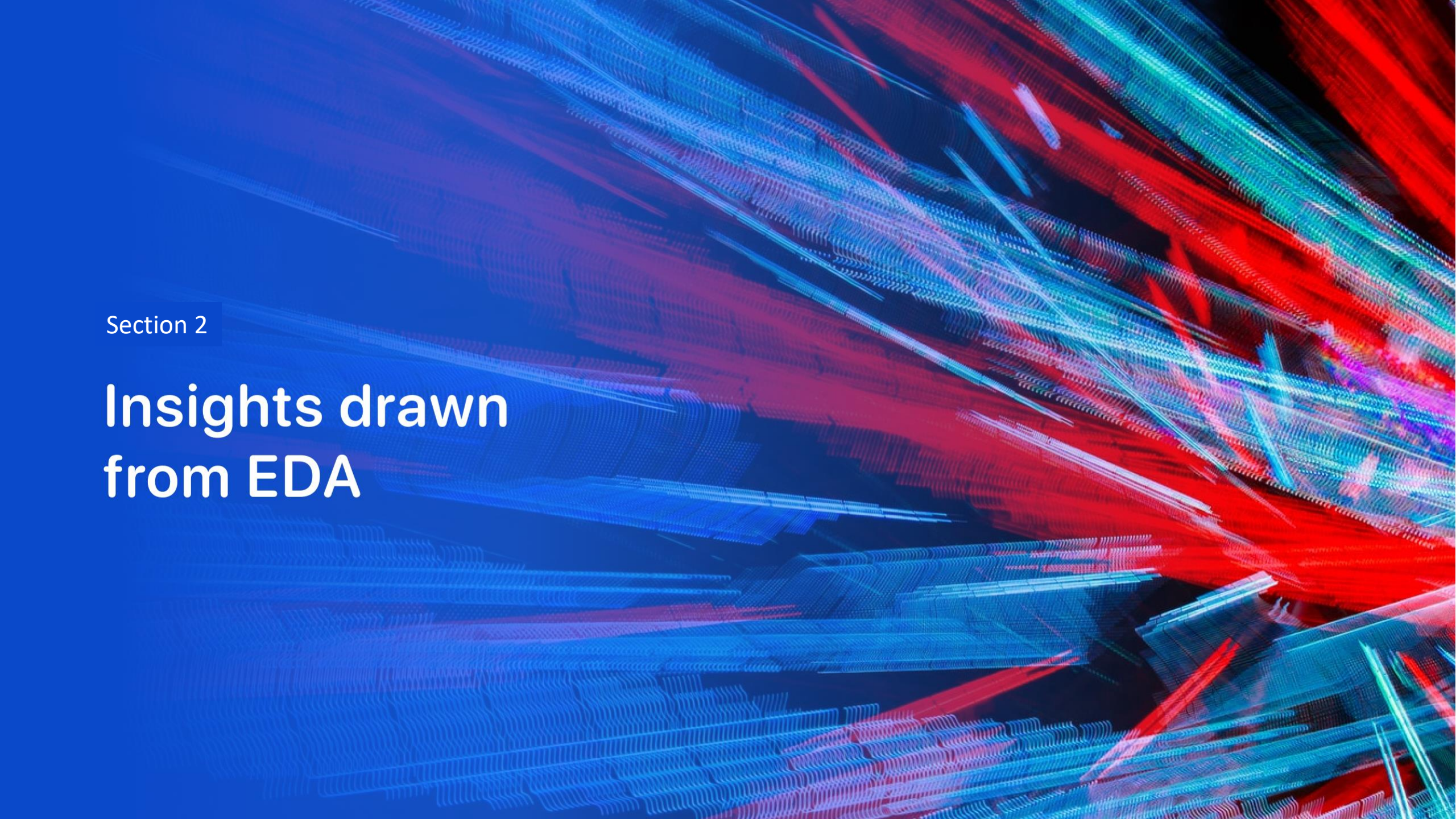
Predictive Analysis (Classification)

- For data preparation, we load data, standardize the data and split data into training and testing sets
- Select hyperparameters for each machine learning algorithm using the function GridSearchCV.
- Display the best parameters using the data attribute best_params_ and the accuracy on the validation data using the data attribute best_score_
- Calculate the accuracy on the test data using the method score
- Plot confusion matrix for each machine learning algorithm
- Compare all 4 models based on their accuracy
- Notebook

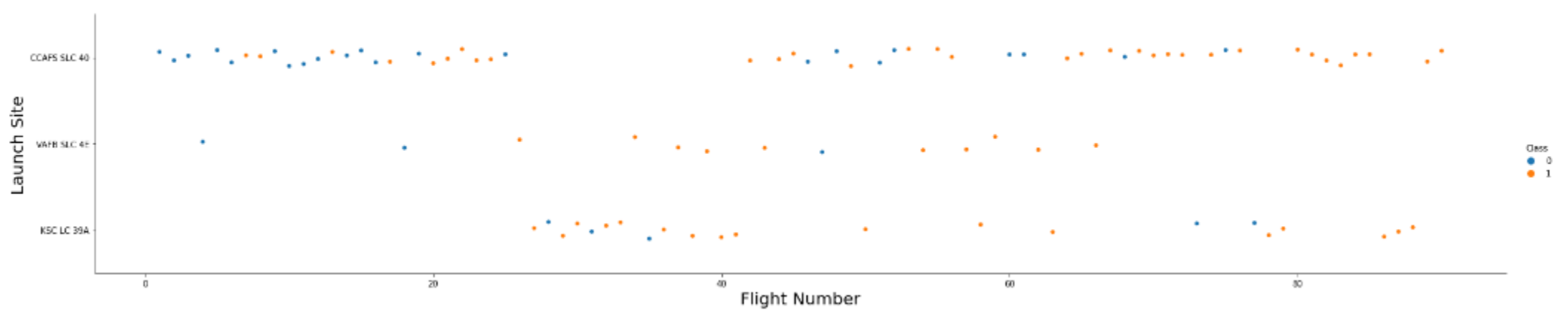


Results

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

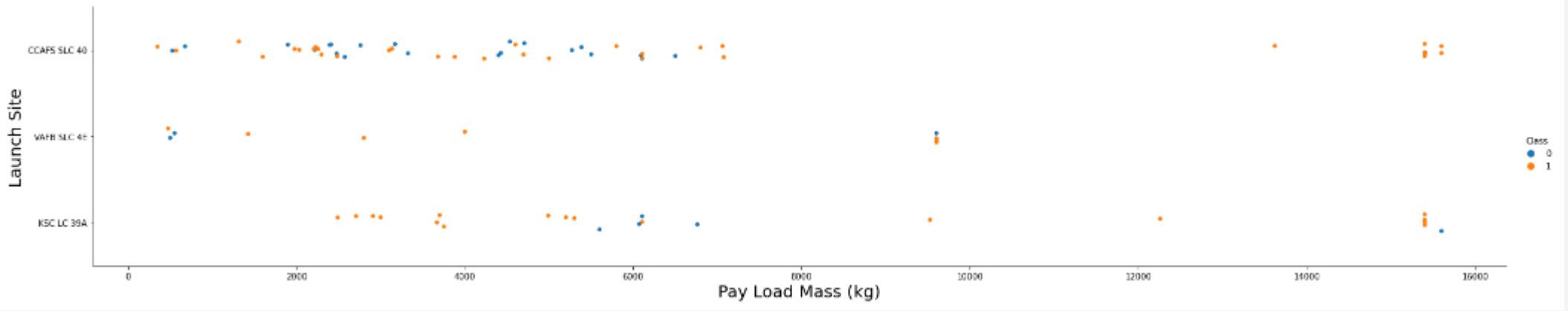

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



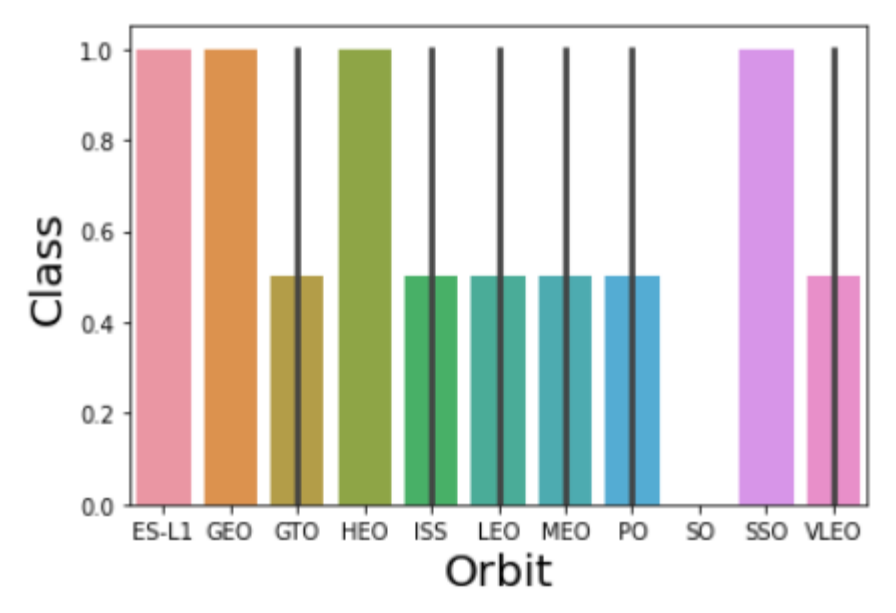
- The greater flight number, the higher success rate. Especially when the flight number is greater than 20

Payload vs. Launch Site



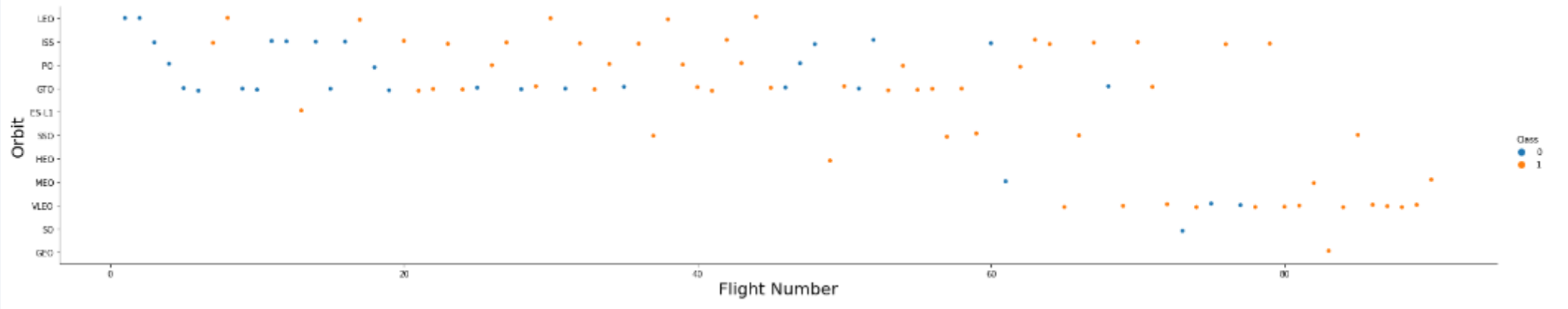
- Depends on the launch site, payload could a factor on the success rate.
- For example, KSC LC-39A tends to have higher success rate when payload mass less than 6000 kg or higher than 9000kg
- CCAFS SLC-40 favors heavier payload mass

Success Rate vs. Orbit Type



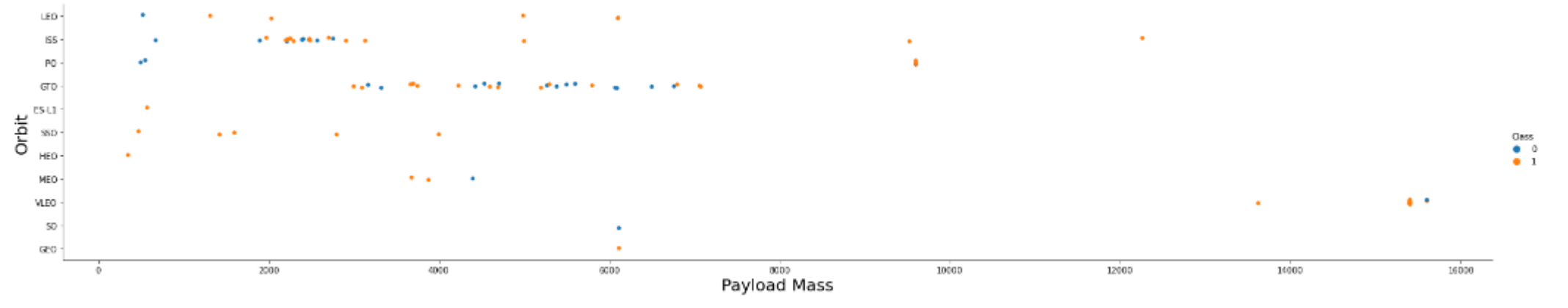
- According to the bar chart, we can see that orbit ES-L1, GEO, HEO, and SSO has the highest success rate

Flight Number vs. Orbit Type



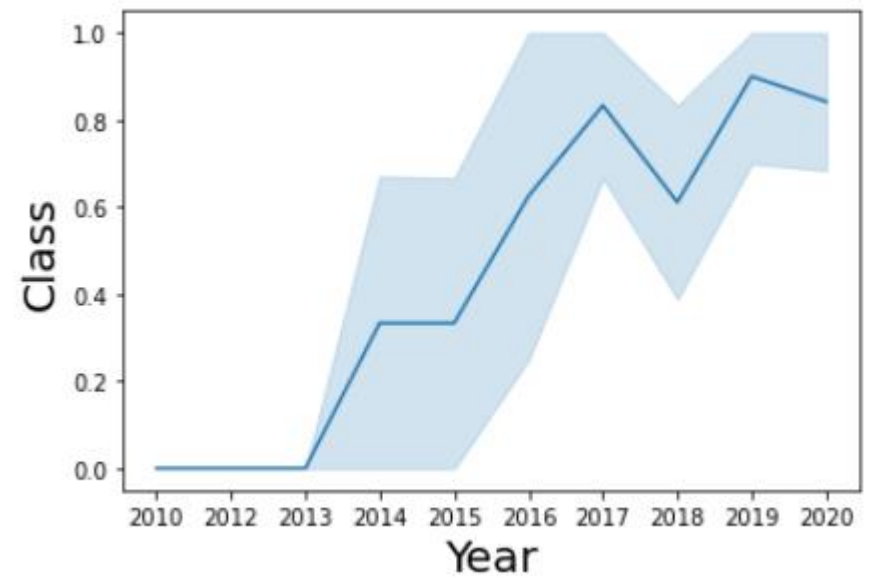
- Based on this scatter plot, we can see the success rate based on the flight number and orbit. Below are some findings:
  - Orbit VLEO has better success rate with greater flight number.
  - Orbit LEO has better success rate as the flight number increases
  - Obit GTO does not have a clear relation on success rate and flight number

Payload vs. Orbit Type



- The higher payload mass for orbit LEO and ISS, the better success rate;
- Orbit SSO has had successful launches when payload are round 4000 kg or less
- Orbit GTO doesn't have a clear relation, but we can still observe that the heavier payload mass the higher failure rate

Launch Success Yearly Trend



- The success rate has rapidly increased after year of 2013 with a small dip in 2018 but increased between 2018 and 2019

All Launch Site Names

- SQL Query:

```
%sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL;
```

- Result with all launch site names:

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

- The keyword **DISTINCT** was used to remove duplicates and only display the unique names

Launch Site Names Begin with 'CCA'

```
In [18]: %sql SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
 * ibm_db_sa://jtp80286:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

Out[18]:

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

- By using keyword LIKE and wildcard %, we can find launch site that starts with “CCA”
- Keyword LIMIT limits certain number of records to be returned

Total Payload Mass

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE CUSTOMER = 'NASA (CRS)';
```

```
 * ibm_db_sa://jtp80286:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

1
45596

- Function SUM was used to calculate the total payload mass and filtered the data to only when customer is NASA
- The result is 45596 kg

Average Payload Mass by F9 v1.1

- SQL Query:

```
In [20]: %sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE BOOSTER_VERSION = 'F9 v1.1';
```

- Result:
  - The average payload mass by F9 v1.1 IS 2928 KG

```
Out[20]:
   1
2928
```

- We use function AVG to calculate the average payload mass and filter the data only to F9 v1.1 in the WHERE clause

First Successful Ground Landing Date

- SQL Query:

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

- Result:

1
2015-12-22

- We use MIN to find the minimum date (first date) when ground pad landing was success ful

Successful Drone Ship Landing with Payload between 4000 and 6000

- SQL Query:

```
: %sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE LANDING__OUTCOME = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000;
```

- Result:

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

- We filter the data to successful drone ship and payload between 4000 and 6000 in the WHERE clause using AND with BETWEEN … AND…

Total Number of Successful and Failure Mission Outcomes

- SQL Query:

```
%sql SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TotalNumber FROM SPACEXTBL GROUP BY MISSION_OUTCOME;
```

- Result

mission_outcome	totalnumber
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

```
%sql SELECT COUNT(MISSION_OUTCOME) AS Success FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE 'Success%';
```

success
100

```
%sql SELECT COUNT(MISSION_OUTCOME) AS Failure FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE 'Failure%';
```

failure
1

- We use COUNT to count the total number of successful or failure outcomes

Boosters Carried Maximum Payload

- SQL Query:

```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

- Result:

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

- We use DISTINCT to find only the unique booster version
- MAX function is used to filter the data to only the maximum payload

2015 Launch Records

- SQL Query:

```
%sql SELECT LANDING__OUTCOME, BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE LANDING__OUTCOME = 'Failure (drone ship)' AND YEAR(DATE) = 2015;
```

- Result:

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

- Filter the data to year of 2015 and landing outcome is failure drone ship in the where clause

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

- SQL Query:

```
%sql SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) AS TotalNumber FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY LANDING__OUTCOME ORDER BY TotalNumber DESC;
```

- Result:

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

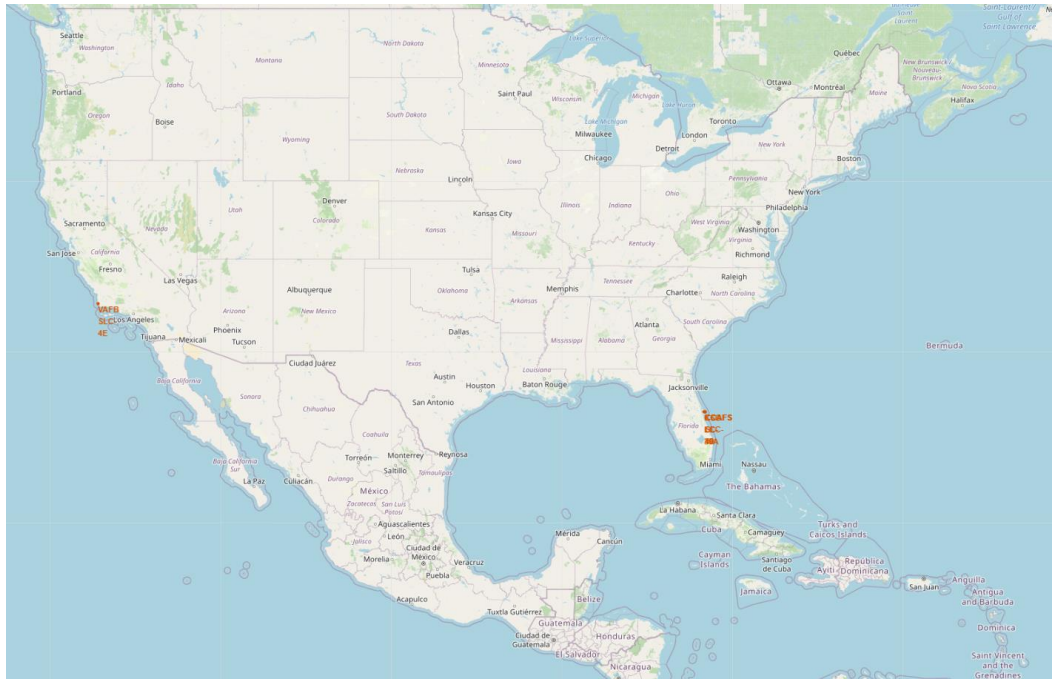
- The SQL query returns a table of landing outcomes and COUNT the total number of that landing outcome BETWEEN 2010-06-04 AND 2017-03-20 ORDER BY the total number in DESC descending order

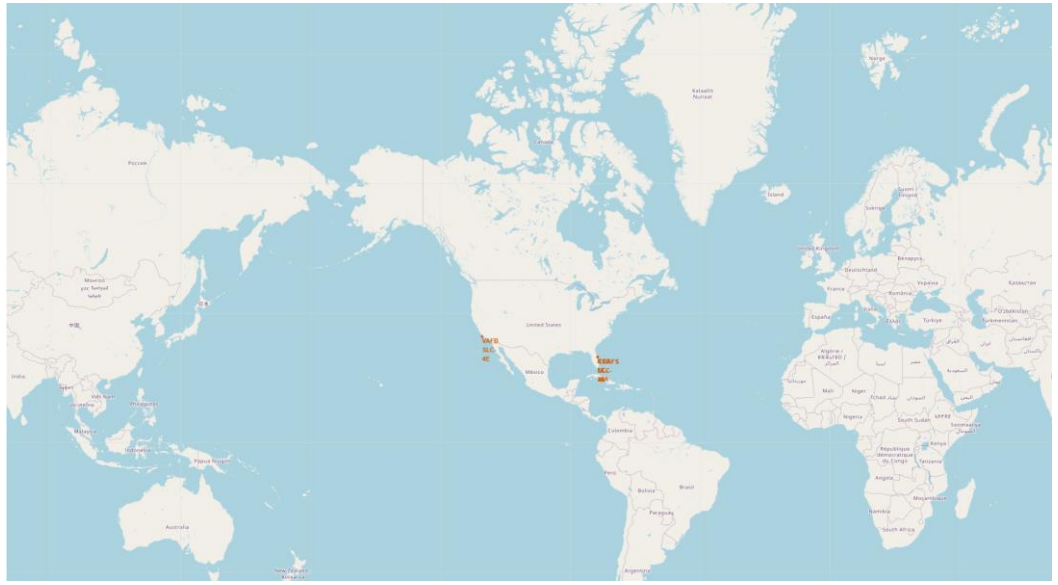
Section 3

Launch Sites Proximities Analysis

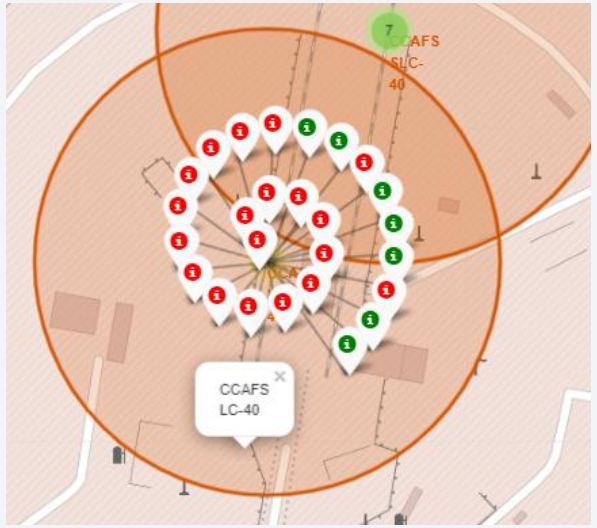
All Launch Sites on Global Map

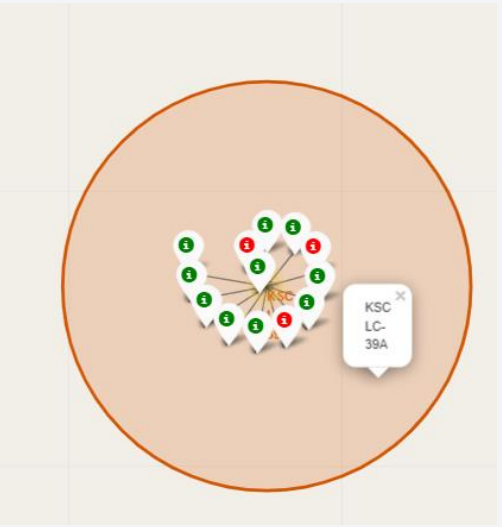
- All launch sites are located in the United State on East and West coast





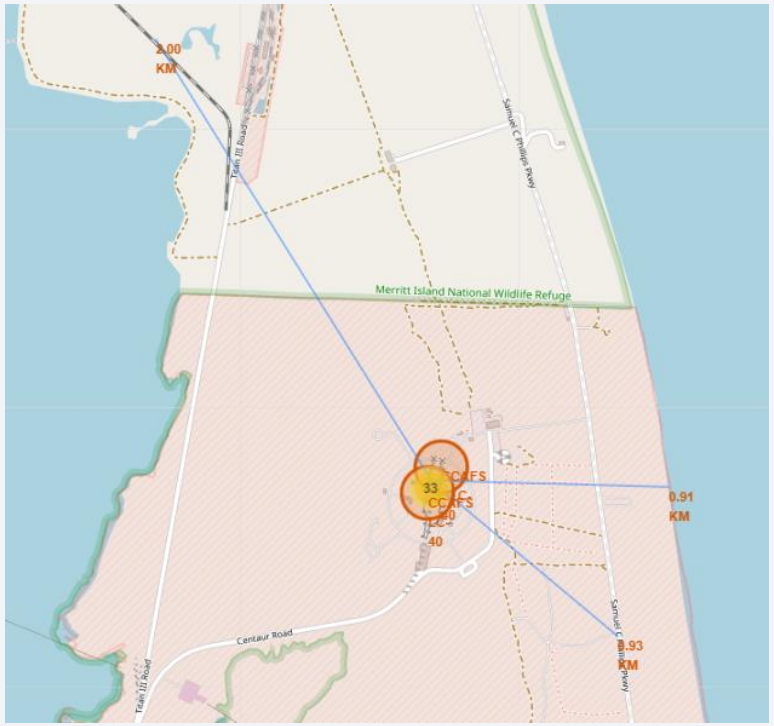
Launch Outcomes for Launch Site

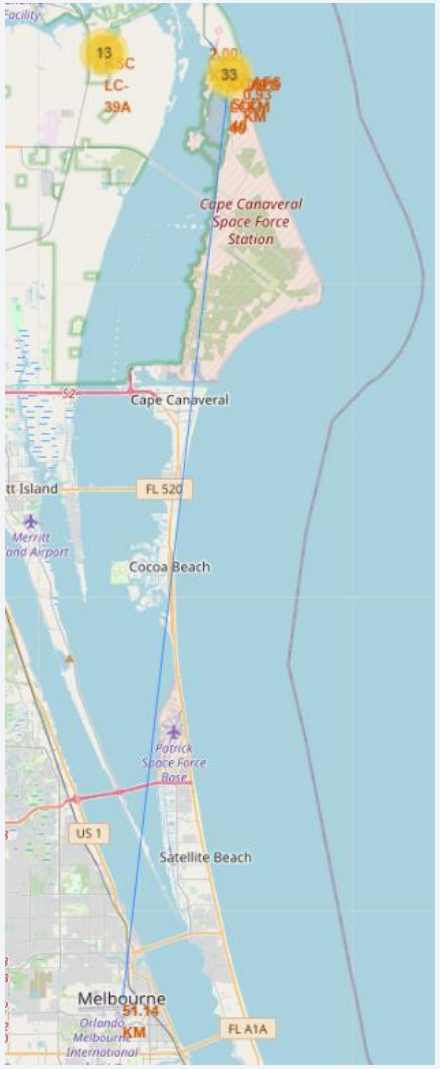




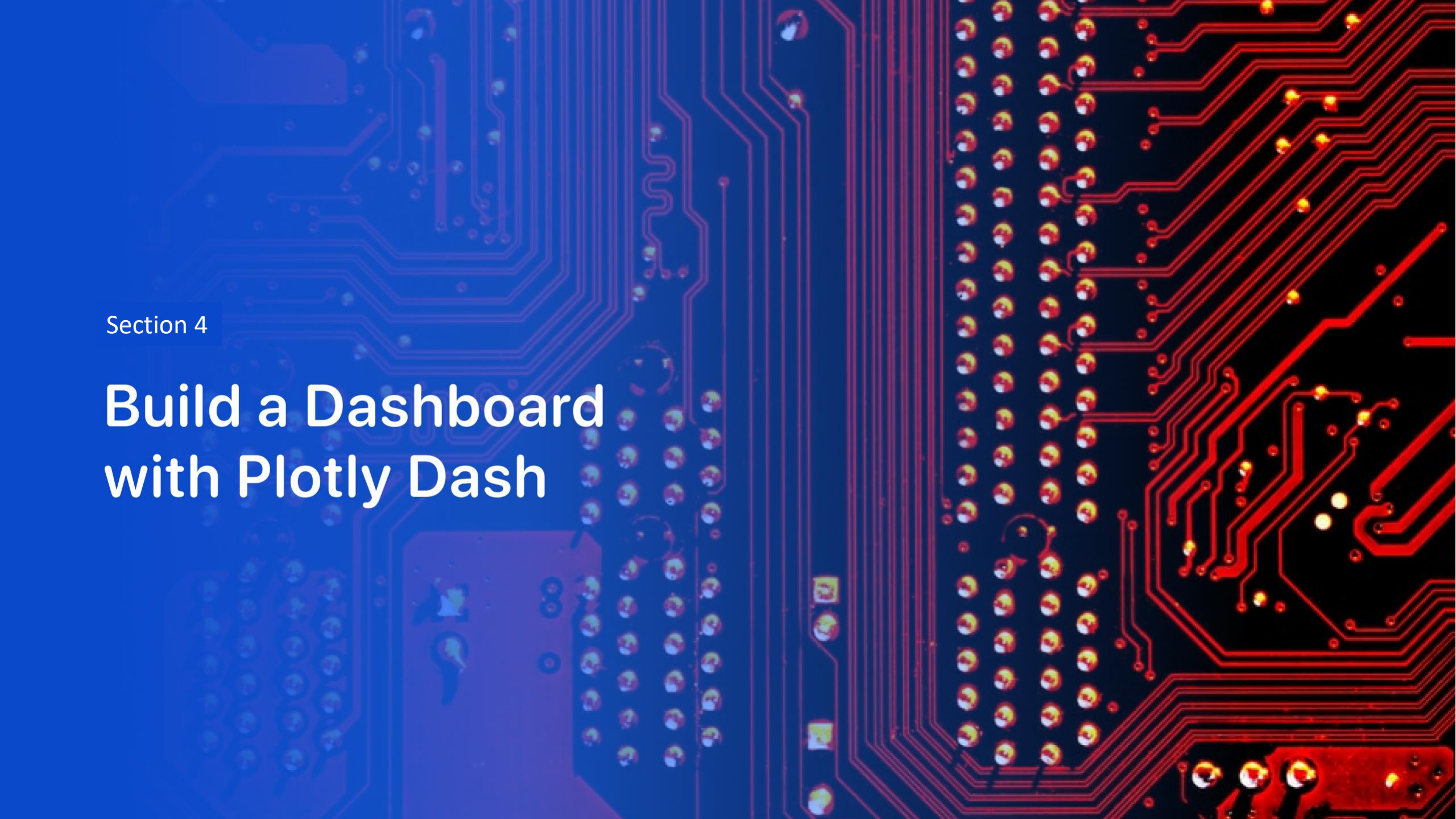
- Green represents successful launches
- Red represents unsuccessful launch
- KSC LC-39A has a more successful launches
- CCAFS LC-40 has more failures

Distance between CCAFS SCL-40 to its proximities

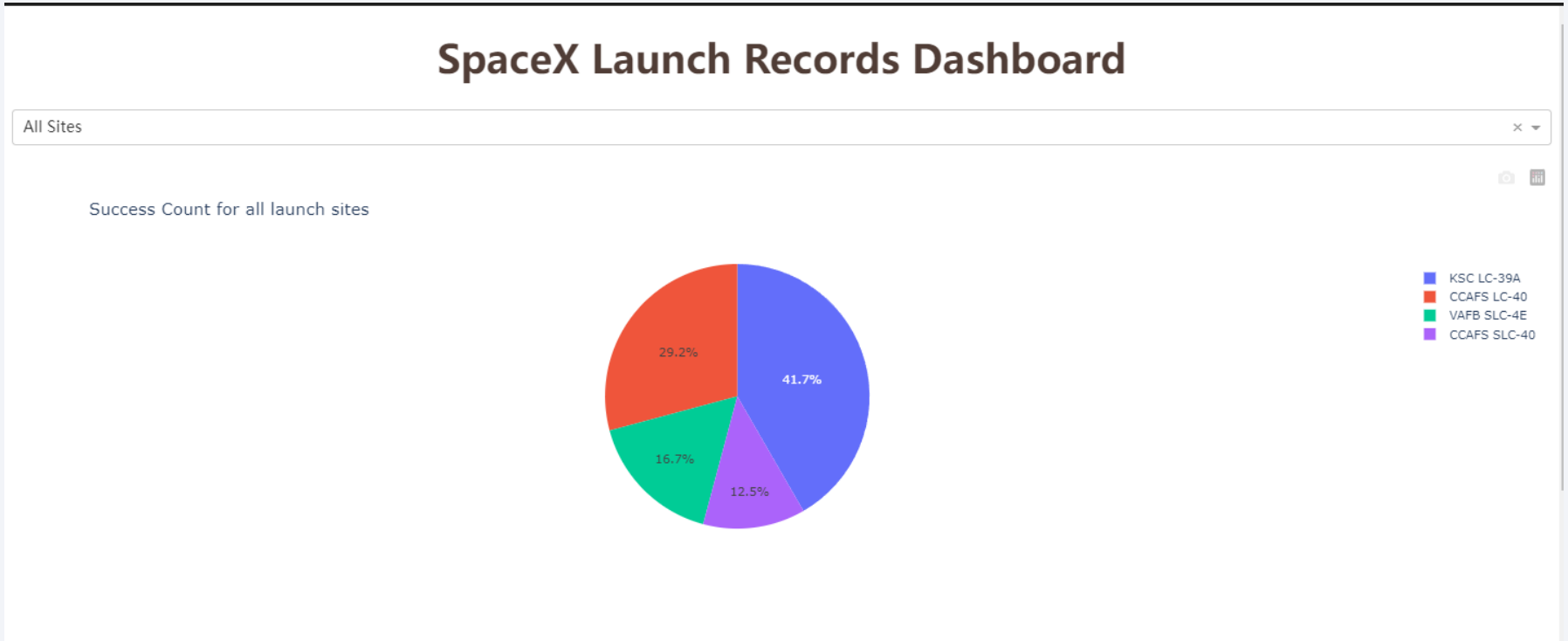




- Are launch sites in close proximity to railways?
  - Yes
- Are launch sites in close proximity to highways?
  - Yes
- Are launch sites in close proximity to coastline?
  - Yes
- Do launch sites keep certain distance away from cities?
  - No

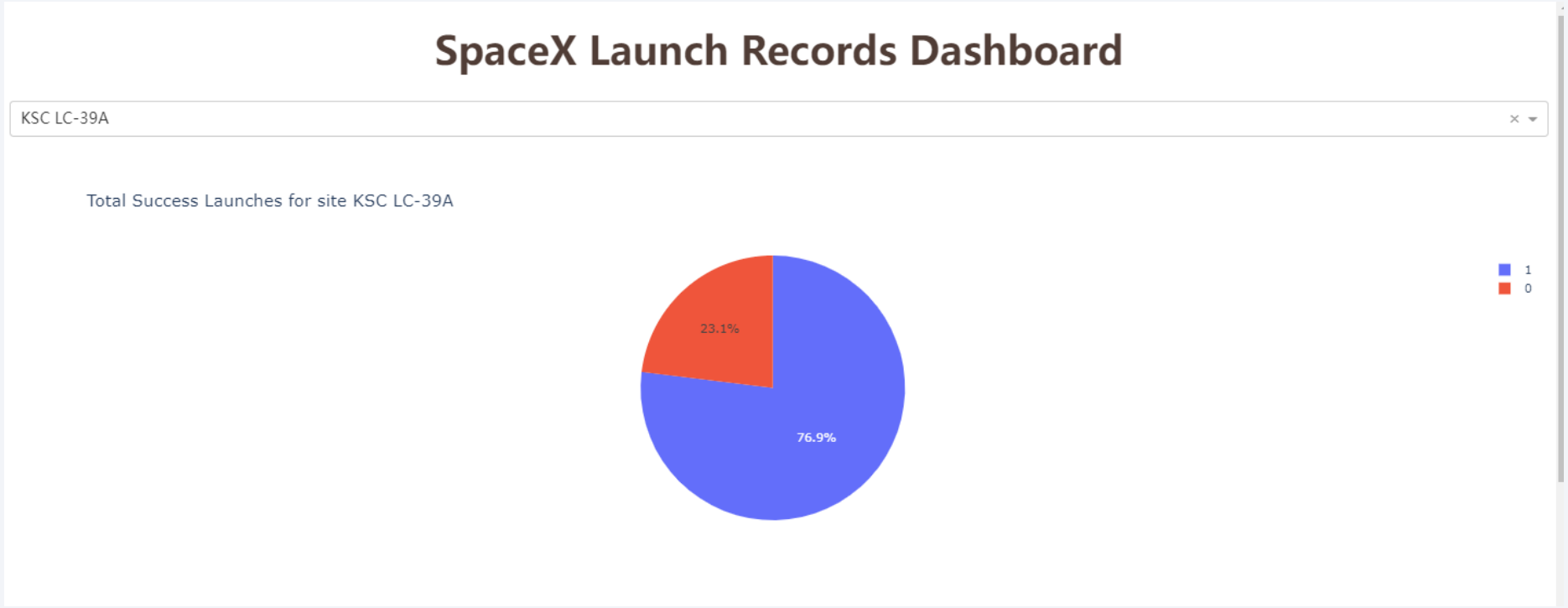

Section 4
Build a Dashboard with Plotly Dash

Success Count by Launch Sites



- Based on the pie chart, we can see that KSC LC-39A has the highest success rate while CCAFS SCL-40 has the lowest.

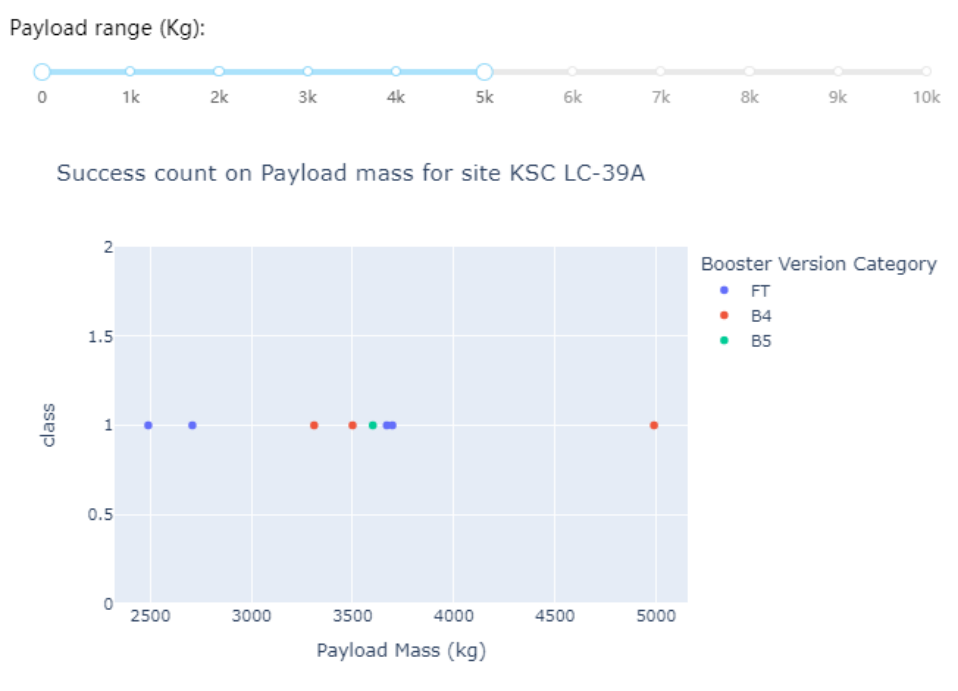
Total Success Launches for Site KSC LC-39A



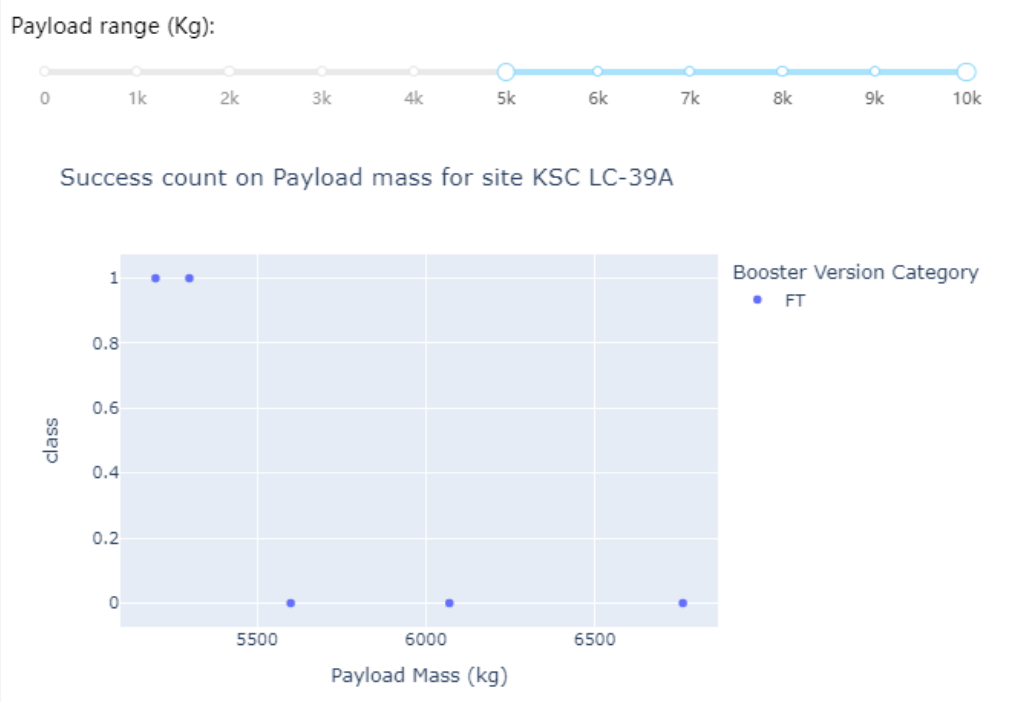
- According the pie chart, KSC LC-39A has rate of 76.9% of successful launches and 23% failure rate

Payload vs. Launch Outcome

- Payload range 0 – 5000 KG



- Payload range 5000 – 10000 KG



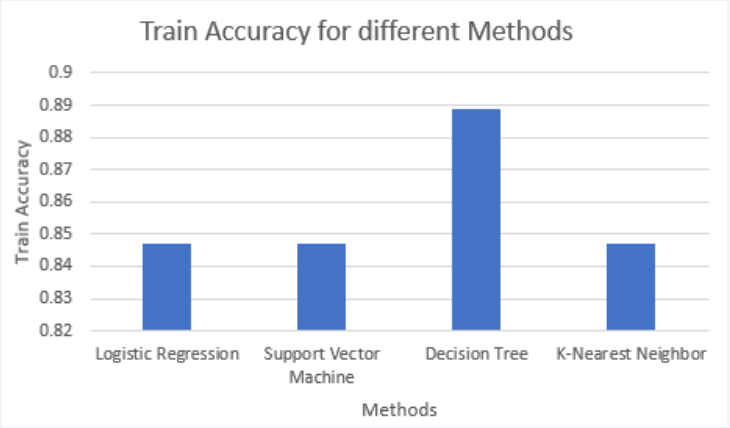
- Lower payload mass tends to have higher success rate

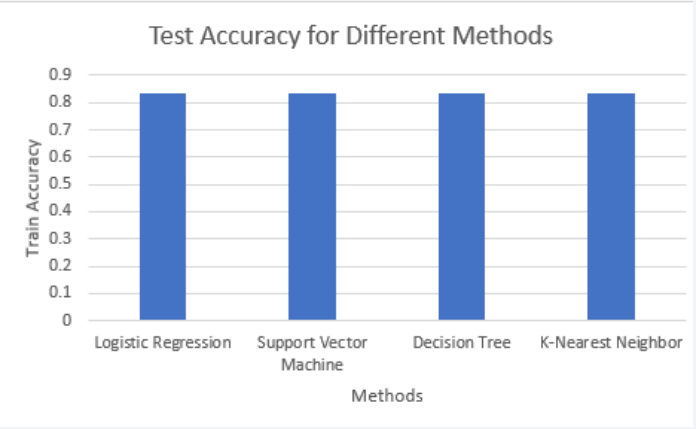
Section 5

Predictive Analysis (Classification)

Classification Accuracy

Models	Train	Test
Logistic Regression	0.8472222222222222	0.8333333333333334
Support Vector Machine	0.8472222222222222	0.8333333333333334
Decision Tree	0.8888888888888888	0.8333333333333334
K-Nearest Neighbor	0.8472222222222222	0.8333333333333334



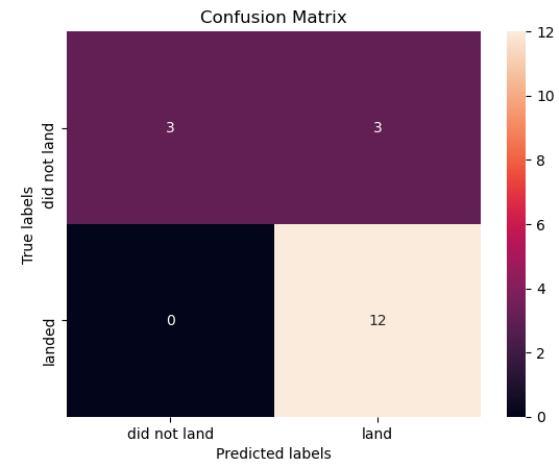


- All four methods have the same accuracy on test data, however, Decision Tree performed better on training data with the best parameters shown below

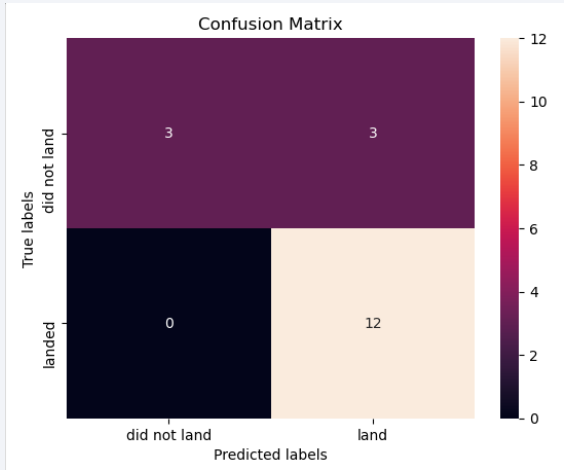
```
: print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
  print("accuracy :",tree_cv.best_score_)

    tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 8, 'max_features': 'sqr
    t', 'min_samples_leaf': 4, 'min_samples_split': 5, 'splitter': 'random'}
    accuracy : 0.8888888888888888
```

Confusion Matrix

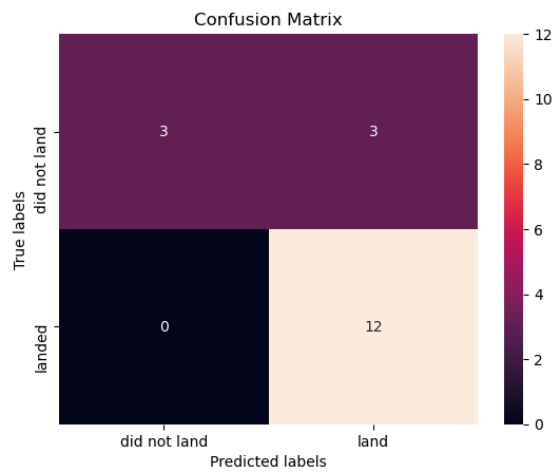
- Logistic Regression



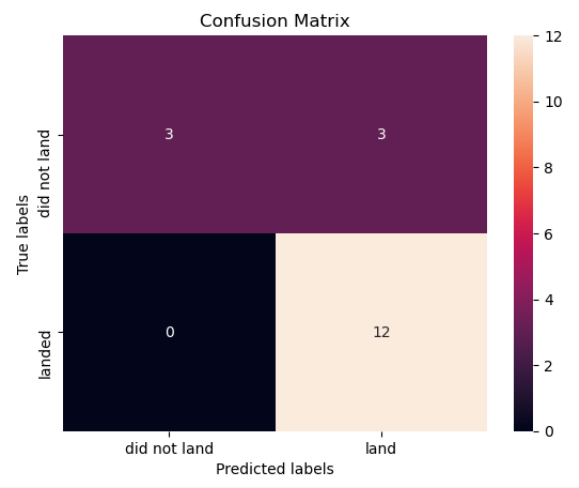
- Support Vector Machine



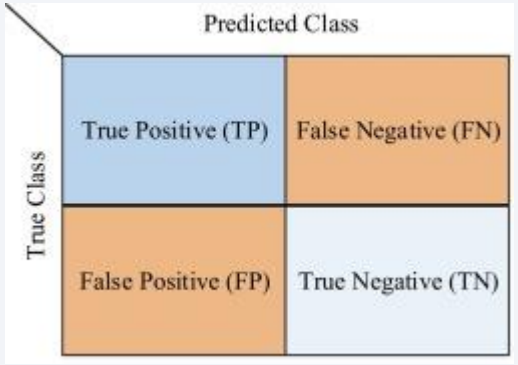
- Decision Tree



- K-Nearest Neighbor



- Since the test accuracy was the same for all four models, the confusion matrix are the same
- The main problem of these models is False Positive such that failure will be predicted as success



Conclusions

- Throughout the entire project, we can conclude that the success rate has close relations with many different affecting factors such as the payload mass, flight numbers, orbit, etc.
- The payload mass can be an affecting factor to the success rate but in different ways. For example, for orbits LEO and ISS, the higher payload mass, the better success rate. However, for orbit SSO, it tends to have successful launches when payload are round 4000 kg or less
- The greater flight number, the higher success rate. Especially when the flight number is greater than 20 with the fact that we take the knowledge of previous flight into account to improve.
- Orbits ES-L1, GEO, HEO, and SSO has the highest success rate
- For this particular project, Decision Tree performed better than the other three models even though the test accuracy were the same, Decision Tree scored higher training accuracy

Appendix

- https://api.spacexdata.com/v4/launches/past
- https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922
- https://github.com/yuxyuxrara/IBM_Data_Science/blob/main/Data%20Collection%20with%20Web%20Scraping%20lab.ipynb
- https://github.com/yuxyuxrara/IBM_Data_Science/blob/main/Data%20Wrangling.ipynb
- https://github.com/yuxyuxrara/IBM_Data_Science/blob/main/DataCollectionAPI.ipynb
- https://github.com/yuxyuxrara/IBM_Data_Science/blob/main/EDA%20with%20SQL.ipynb
- https://github.com/yuxyuxrara/IBM_Data_Science/blob/main/EDA%20with%20Visualization.ipynb
- https://github.com/yuxyuxrara/IBM_Data_Science/blob/main/Interactive%20Visual%20Analytics%20with%20Folium.ipynb
- https://github.com/yuxyuxrara/IBM_Data_Science/blob/main/Machine%20Learning%20Prediction.ipynb
- https://github.com/yuxyuxrara/IBM_Data_Science/blob/main/SpaceX%20Dashboard.py

Thank you!