IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Data Science and Machine Learning Capstone Project

Maria Maura S. Tinao
June 1 , 2022

Outline

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

Executive Summary

Summary of methodologies

The following methodologies were used to to collect data from various sources.:

1. Data Collection through API
2. Data Collectiom with Web Scraping
3. Data Wrangling
4. Exploratory Data Analysis (EDA) with SQL
5. Exploratory Data Analysis (EDA) with Data Visualization
6. Interactive Visual Analytics with Folium
7. Machine Learning Prediction

Summary of all results

1. Exploratory Data Analysis (EDA) results
2. Interactive analytics demo in screenshots
3. Predictive analysis.

Introduction

Project background

Space tourism or space travel is gaining grounds through the participation of private companies such as SpaceX, Blue Origin and Virgin Galactic. Though the rates are afforded only by the world's affluent individuals, SpaceX is offering cheaper space travel with its Falcon 9 rocket launches project.

Space X advertises Falcon 9 rocket launches on its website, that it will land successfully with a cost of only 62 million dollars as compared to the other providers with a cost of 165 million dollars each. This is because SpaceX can reuse the first stage. To do this, it can be determined if the first stage will land, and the cost of launch can also be determined. This information can be used if an alternative company wants to bid against SpaceX for a rocket launch.

Problem

The main problem is to predict if the Falcon 9 first stage will land successfully. In details, this can be done by determining the factors for best successful landing rate, its relationships between rocket variables and its landing rate and the conditions for successful landings.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:

  From SpaceX API, Web Scraping and Falcon 9 and Falcon heavy launches records from Wikipedia

Perform data wrangling

  Determine the training labels of successful/unsuccessful landings

- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - create a column for the class
  - standardize the data
  - split into training data and test data
  - Find the best Hyperparameter for SVM, Classification Trees and Logistic Regression
  - Find the method performs best using test data

Data Collection – SpaceX API

Request to the SpaceX API

Clean the requested data

1. Request rocket launch data from SpaceX API.

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

response = requests.get(spacex_url)
```

2. Decode the response content as a Json using .json()

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

3. Data is stored to create a new dataframe:

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

Data Collection – SpaceX API

4. Filtered and cleaned the dataframe to include only Falcon 9 launches

```
data_falcon9 = launch_df[launch_df['BoosterVersion'] == 'Falcon 9']
```

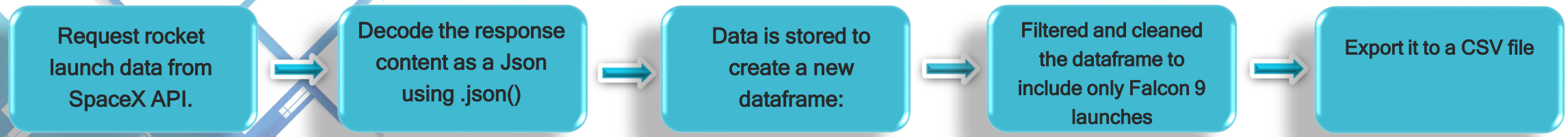
Reset the fine Flightnumber column

```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

5. Export it to a CSV file

```
data_falcon9.to_csv('dataset_part\_1.csv', index=False)
```

GITHUB URL: Data Collection API



Data Collection – Web Scraping

1. Perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

```
html_data = requests.get(static_url).text
```

2. Create a BeautifulSoup object from the HTML response

```
soup = BeautifulSoup(html_data, 'html5lib')
```

3. Find all tables on the wiki page

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

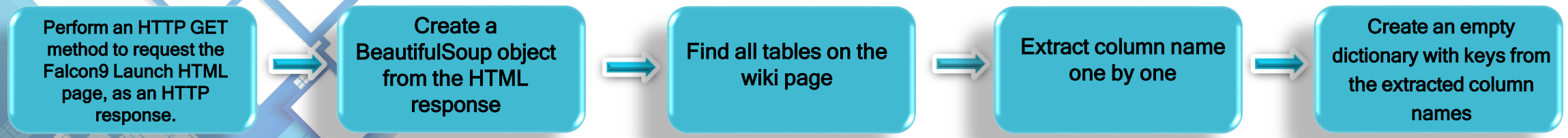
4. Extract column name one by one

```
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 an empty dictionary with keys from the extracted column names

```
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']=[]
```

Github URL: Data Collection with Webscraping



Data Collection – Data Wrangling

Performed Exploratory Data Analysis (EDA) to find some patterns in the data and determine what would be the label for training supervised models.

There are several different cases where the booster did not land successfully. Sometimes a landing was attempted but failed due to an accident. Examples:

- True Ocean means the mission outcome was successfully landed to a specific region of the ocean.
- False Ocean means the mission outcome was unsuccessfully landed to a specific region of the ocean.
- True RTLS means the mission outcome was successfully landed to a ground pad.
- False RTLS means the mission outcome was unsuccessfully landed to a ground pad.
- True ASDS means the mission outcome was successfully landed on a drone ship False ASDS means the mission outcome was unsuccessfully landed on a drone ship.

The outcomes are converted into Training Labels with 1 means the booster successfully landed 0 means it was unsuccessful.

Github URL: EDA with Data Wrangling

Data Collection – Data Wrangling

1. Determine the number of launches on each site.

```
df.Orbit.value_counts()
```

2. Determine the number and occurrence of each orbit

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

3. Determine the number and occurence of mission outcome per orbit type.

```
landing_outcomes = df.Outcome.value_counts()
landing_outcomes
```

4. Create a landing outcome label from Outcome column.

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

5. Determine the success rate.

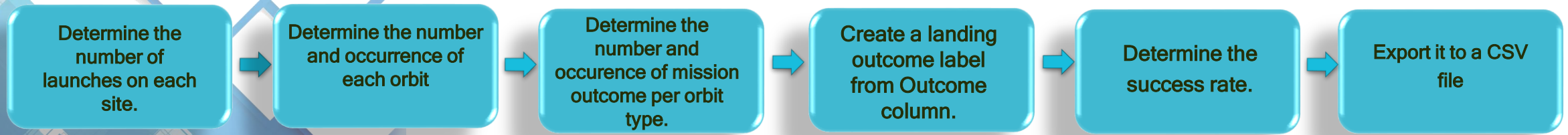
```
df['Class']=landing_class
df[['Class']].head(8)
```

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

6. Export it to a CSV file

```
df.to_csv("dataset_part\_2.csv", index=False)
```

Github URL: EDA Data Wrangling



EDA with Data Visualization

Scatter Point Charts – relationship between:

Flight Number and Launch Site

Payload and Launch Site

Flight number and Orbit Type

Payload and Orbit Type

Bar charts - relationship between:

Success rate of each orbit type

Line Chart

Launch success yearly trends

Github URL: EDA with Data Visualization

EDA with SQL

Load the SpaceX dataset in a DB2 database. The following SQL queries are performed:

- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'KSC‘
- 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 where the first succesful landing outcome in drone ship was acheived.
- List the names of the boosters which have success in ground pad 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 records which will display the month names, succesful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017
- Rank the count of successful landing_outcomes between the date 2010-06-04 and 2017-03-20 in descending order.

Github URL jupyter-labs-eda-sql-edx

Build an Interactive Map with Folium

The launch success rate may depend on many factors such as payload mass, orbit type, and so on. It may also depend on the location and proximities of a launch site, i.e., the initial position of rocket trajectories. Finding an optimal location for building a launch site certainly involves many factors and hopefully we could discover some of the factors by analyzing the existing launch site locations.

Objects created and added to a folium map:

- Mark all launch sites on a map
- Mark the success/failed launches for each site on the map
- Calculate the distances between a launch site to its proximities

The objects above are added to answer the following questions:

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

GitHub Interactive Visual Analytics with Folium.ipynb

Build a Dashboard with Plotly Dash

Pie chart was used to show the total successful launches count for all sites and to show the success vs. failed counts for the site

Scatter Chart was used to show the correlation between payload and launch success

Github URL: Interactive Plotly Dash

Predictive Analysis (Classification)

1. Create a NumPy array from the column Class in data, by applying the method to_numpy() then assign it to the variable Y

```
Y = data['Class'].to_numpy()
```

2. Standardize the data in X then reassign it to the variable X

```
transform = preprocessing.StandardScaler()

X = transform.fit_transform(X)
```

3. Use the function train_test_split to split the data X and Y into training and test data.

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

4. Create a logistic regression object then create a GridSearchCV object

```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}
lr=LogisticRegression()
logreg_cv = GridSearchCV(lr, parameters, cv = 10)
logreg_cv.fit(X_train, Y_train)
```

5. Calculate the accuracy on the test data using the method score.

```
methods = []
accuracy = []

methods.append('Logistic regression')
accuracy.append(logreg_cv.score(X_test, Y_test))
```

```
print("test set accuracy :",logreg_cv.score(X_test, Y_test))
```

Predictive Analysis (Classification)

6. Create a support object then create a GridSearchCV object svm_cv.

```
parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
              'C': np.logspace(-3, 3, 5),
              'gamma':np.logspace(-3, 3, 5)}
svm = SVC()
```

```
svm_cv = GridSearchCV(svm, parameters, cv = 10)
svm_cv.fit(X_train, Y_train)
```

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

7. Calculate the accuracy on the test data using the method score

```
methods.append('Decision tree classifier')
accuracy.append(tree_cv.score(X_test, Y_test))
print("test set accuracy :",tree_cv.score(X_test, Y_test))
```

8. Create a decision tree classifier object then create a GridSearchCV object tree_cv

```
methods.append('Support vector machine')
accuracy.append(svm_cv.score(X_test, Y_test))
print("test set accuracy :",svm_cv.score(X_test, Y_test))
```

9. Calculate the accuracy of tree_cv on the test data using the method score

```
methods.append('Decision tree classifier')
accuracy.append(tree_cv.score(X_test, Y_test))
print("test set accuracy :",tree_cv.score(X_test, Y_test))
```

```
tree_cv = GridSearchCV(tree,parameters,cv=10)
tree_cv.fit(X_train, Y_train)
```

Predictive Analysis (Classification)

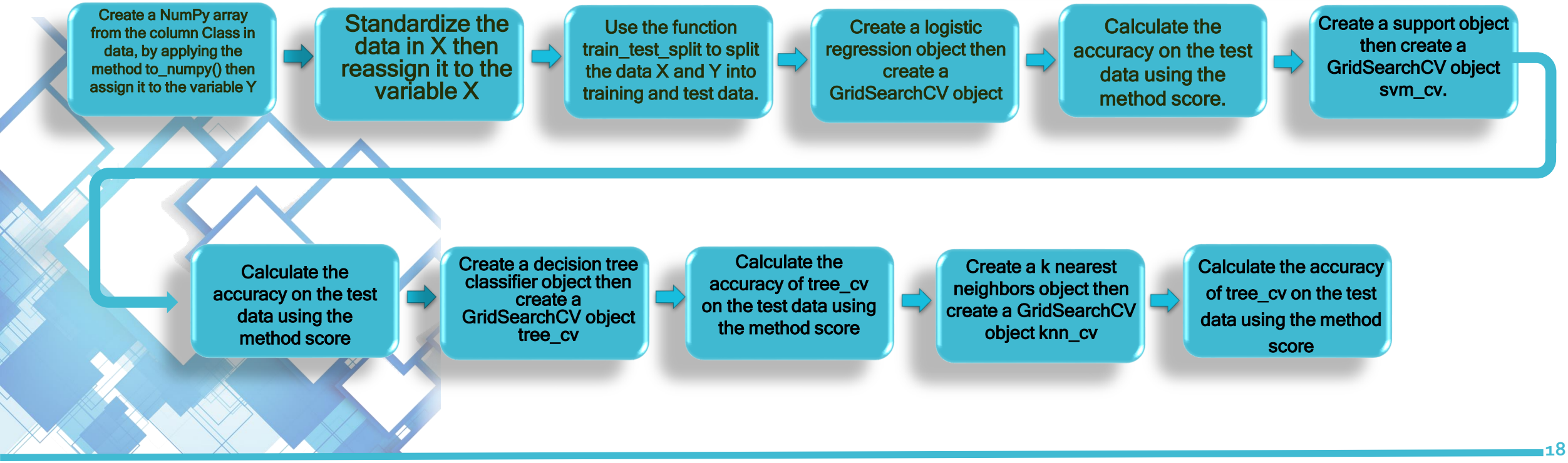
10. Create a k nearest neighbors object then create a GridSearchCV object knn_cv

```
knn_cv = GridSearchCV(KNN,parameters,cv=10)
knn_cv.fit(X_train, Y_train)
```

11. Calculate the accuracy of tree_cv on the test data using the method score

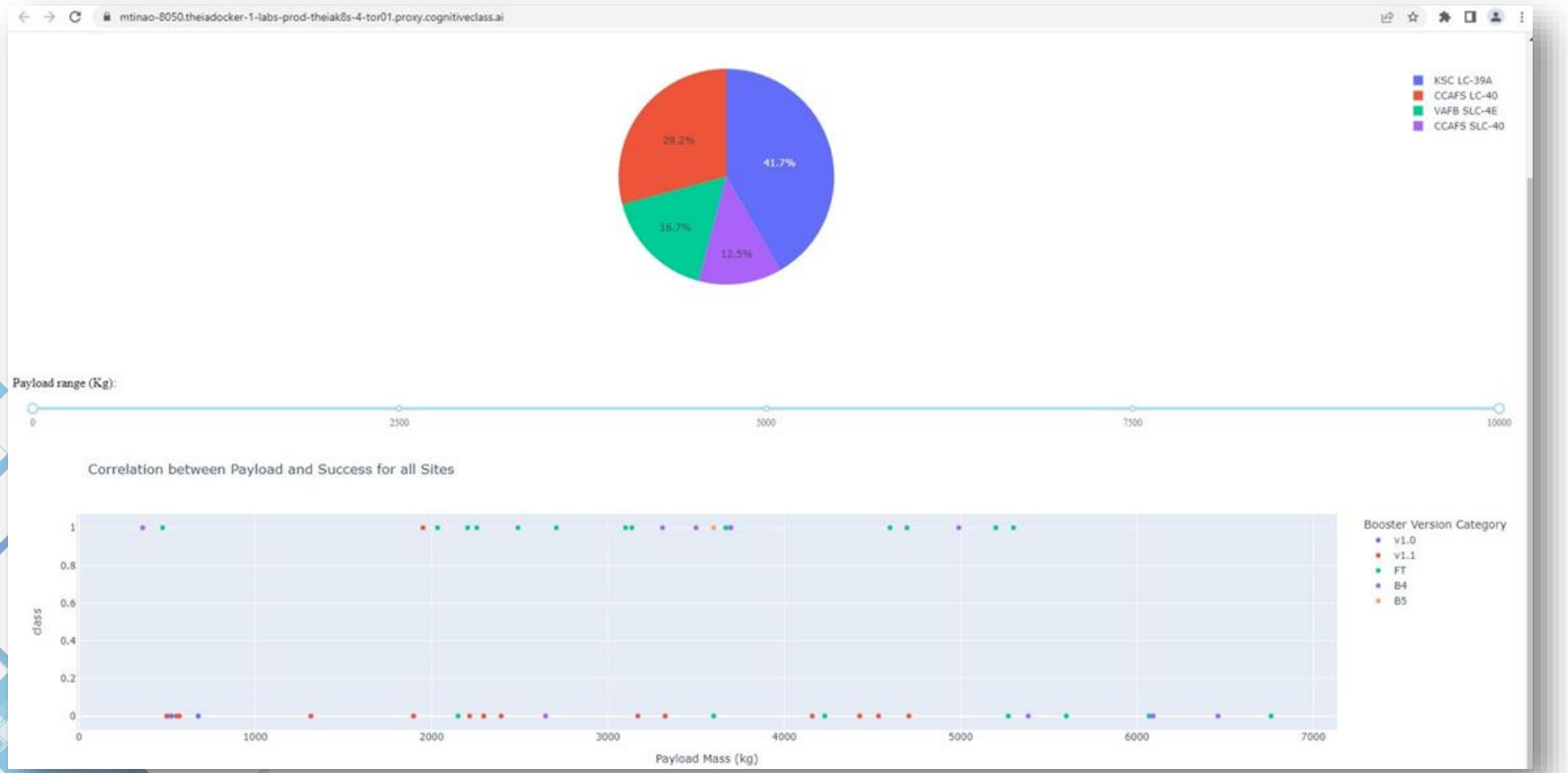
```
methods.append('K nearest neighbors')
accuracy.append(knn_cv.score(X_test, Y_test))
print("test set accuracy :",knn_cv.score(X_test, Y_test))
```

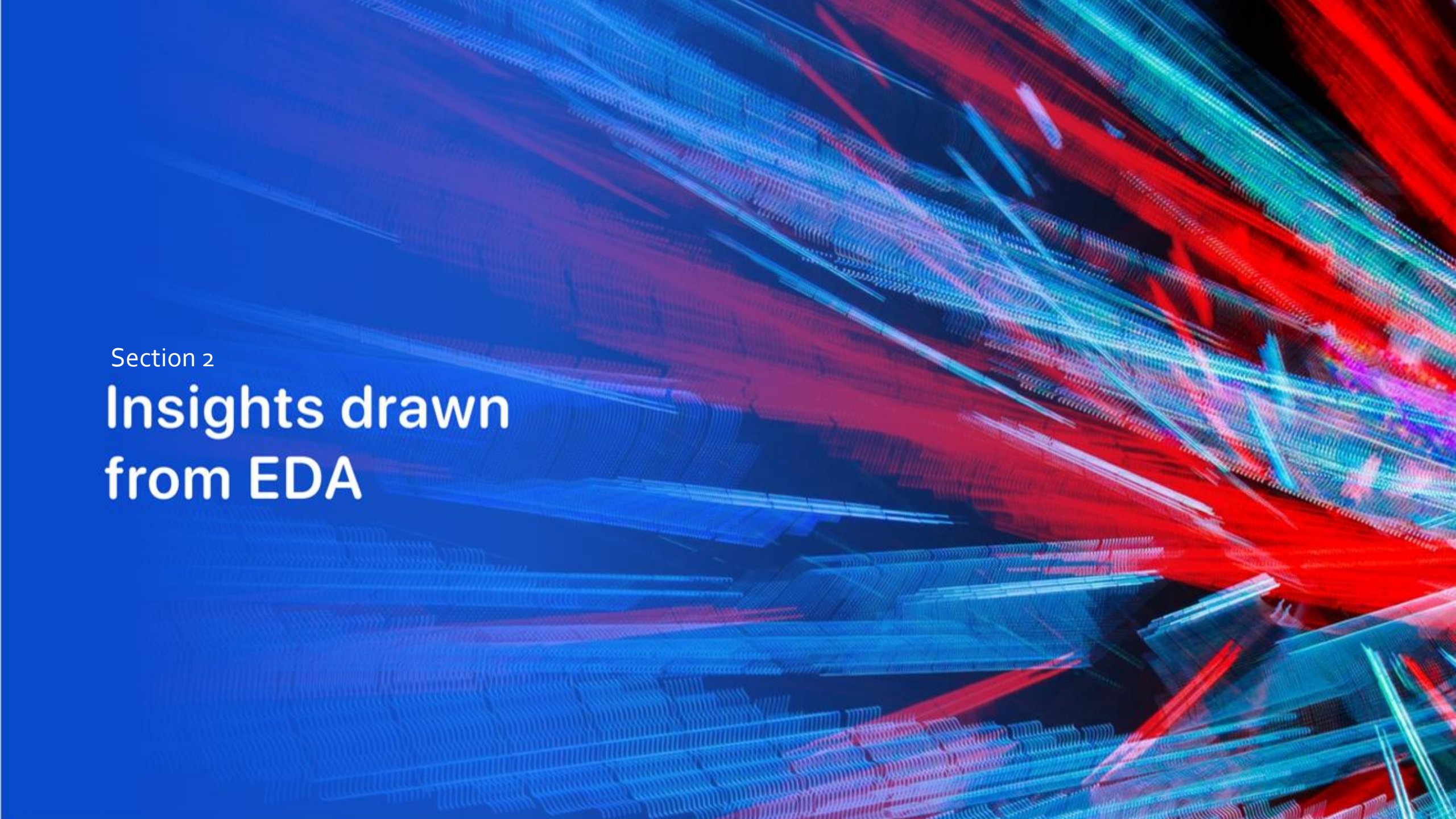
Github URL: [Machine Learning Prediction]



Results

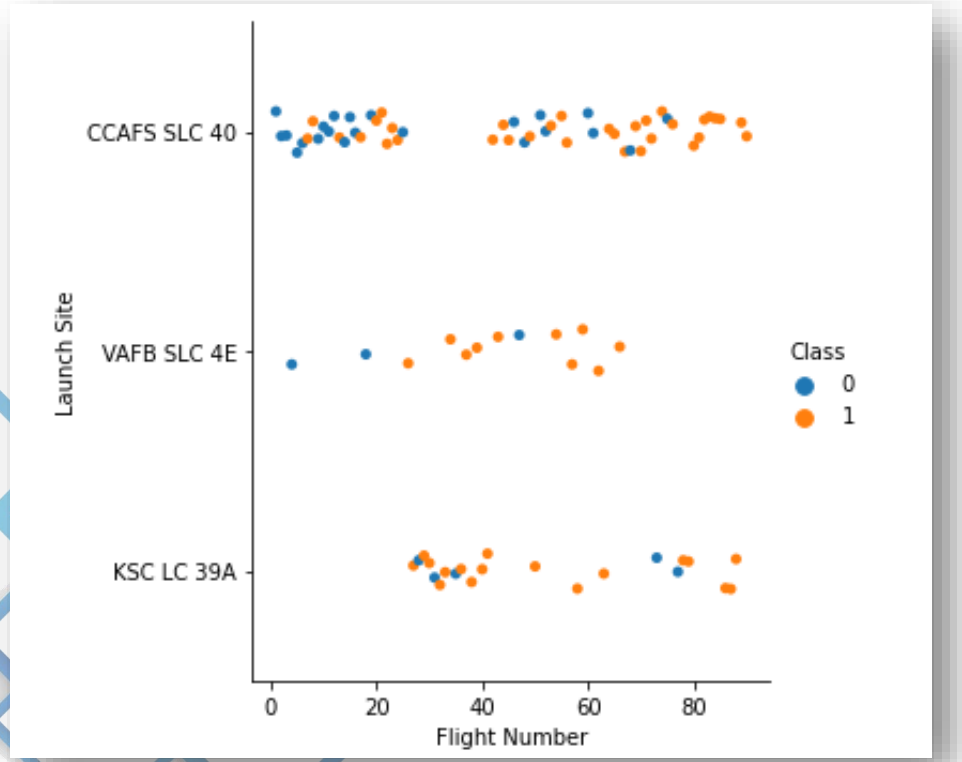
Dashboard with Plotly Dash




Section 2
Insights drawn
from EDA

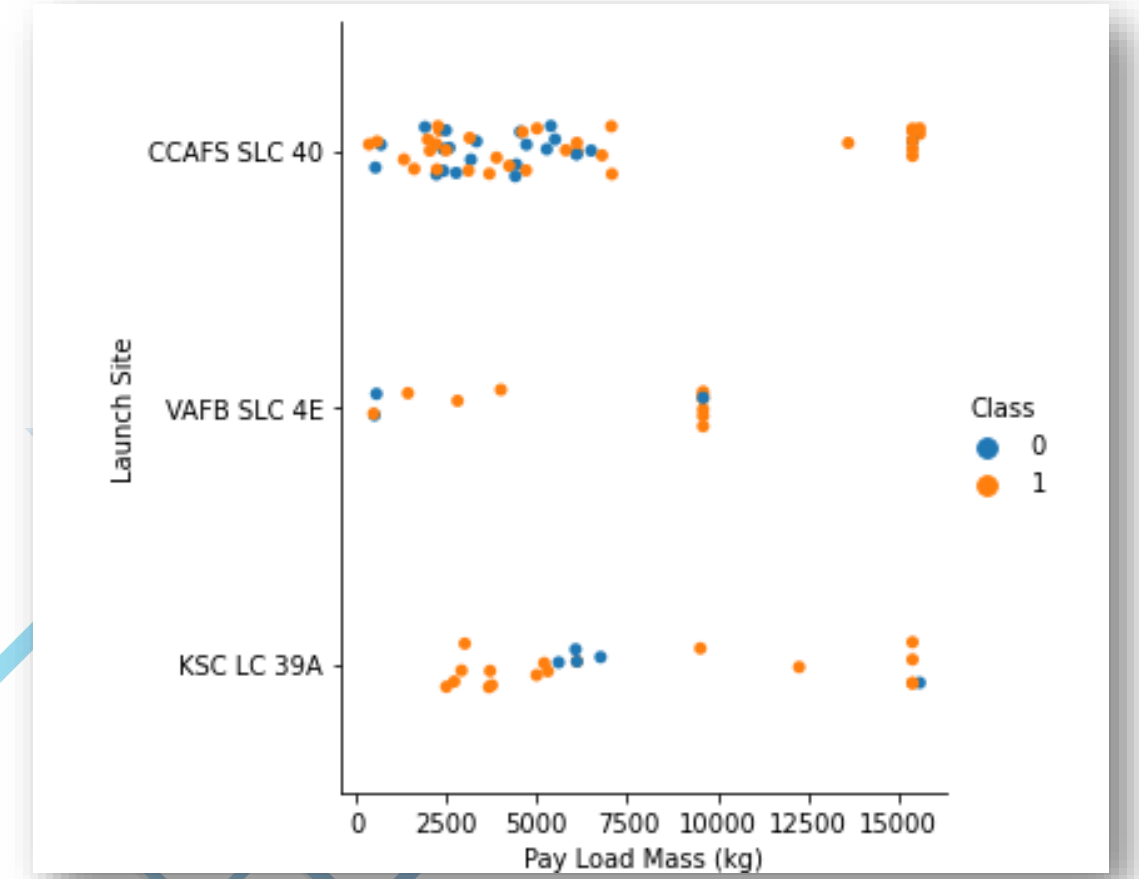
Flight Number vs. Launch Site

Scatter plot of Flight Number vs. Launch Site



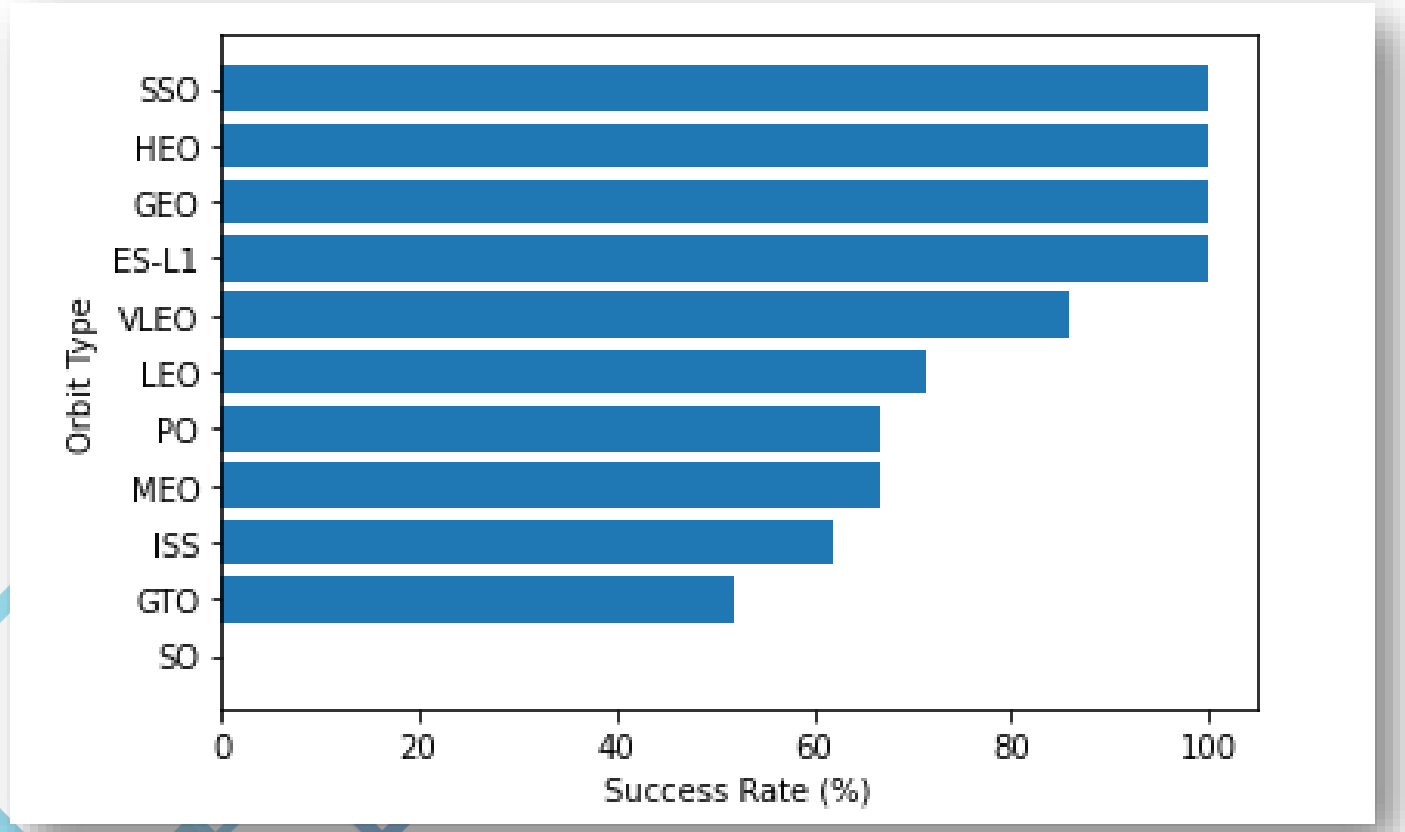
- Class 1 (orange) dominates the figure compared to Class 0 (blue). There are more successful launches (orange) than unsuccessful launches (blue).
- The number of dots represents the number of flights therefore, the more flights launch, the higher the success rate at a launch site as represented by the orange dots.

Payload vs. Launch Site



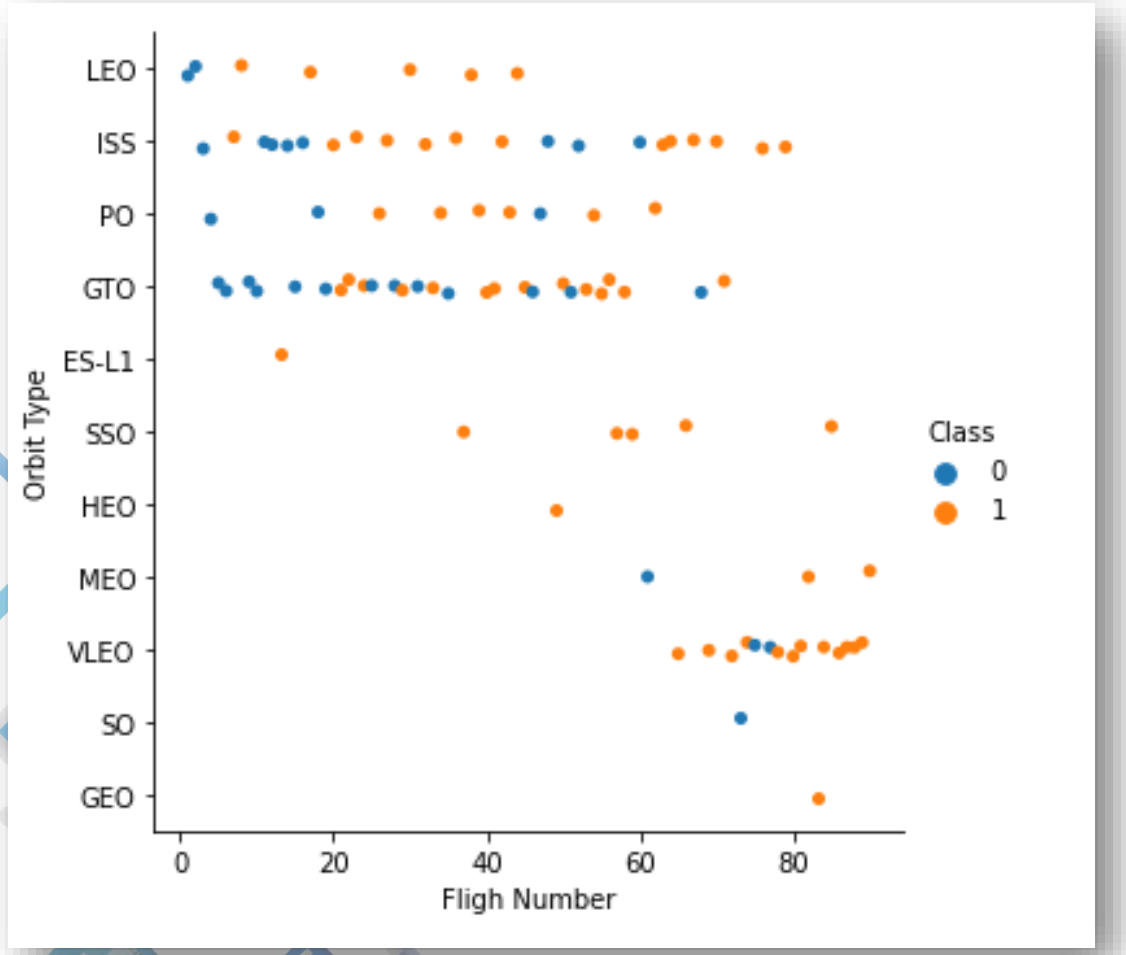
- For CCAFS SLC40 launchsite, there are more rockets launched for payload mass for less than 1000 kg payload mass.
- For VAFB-SLC launchsite there are no rockets launched for heavypayload mass (greater than 10000).
- For KSC LC 39A launchsite, there are less rockets launched up to for heavyload mass
- Therefore, greater success for rockets launched with less payload mass.

Success Rate vs. Orbit Type



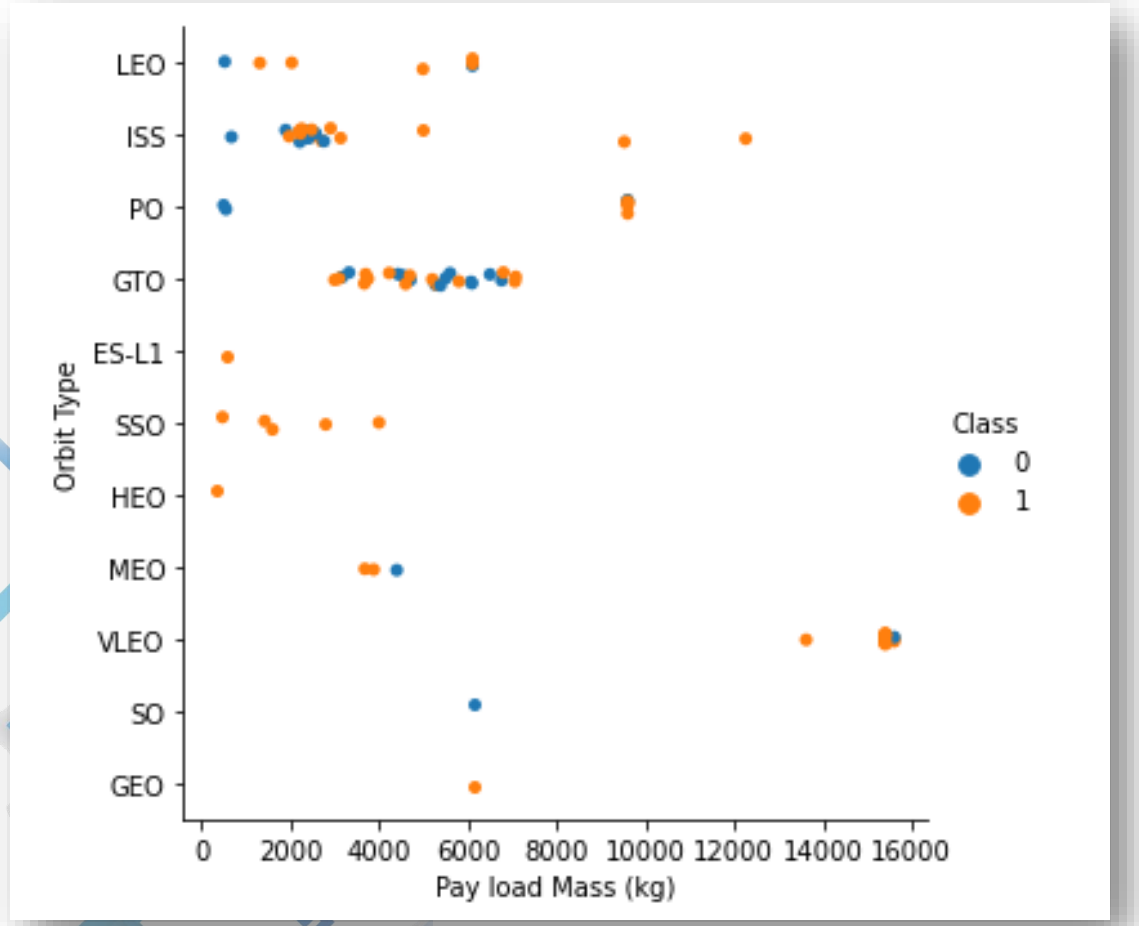
- SSO, HEO, GEO and ESL-1 orbit types have the highest success rates.
- GTO has only 50% of the success rate while SO failed.

Flight Number vs. Orbit Type



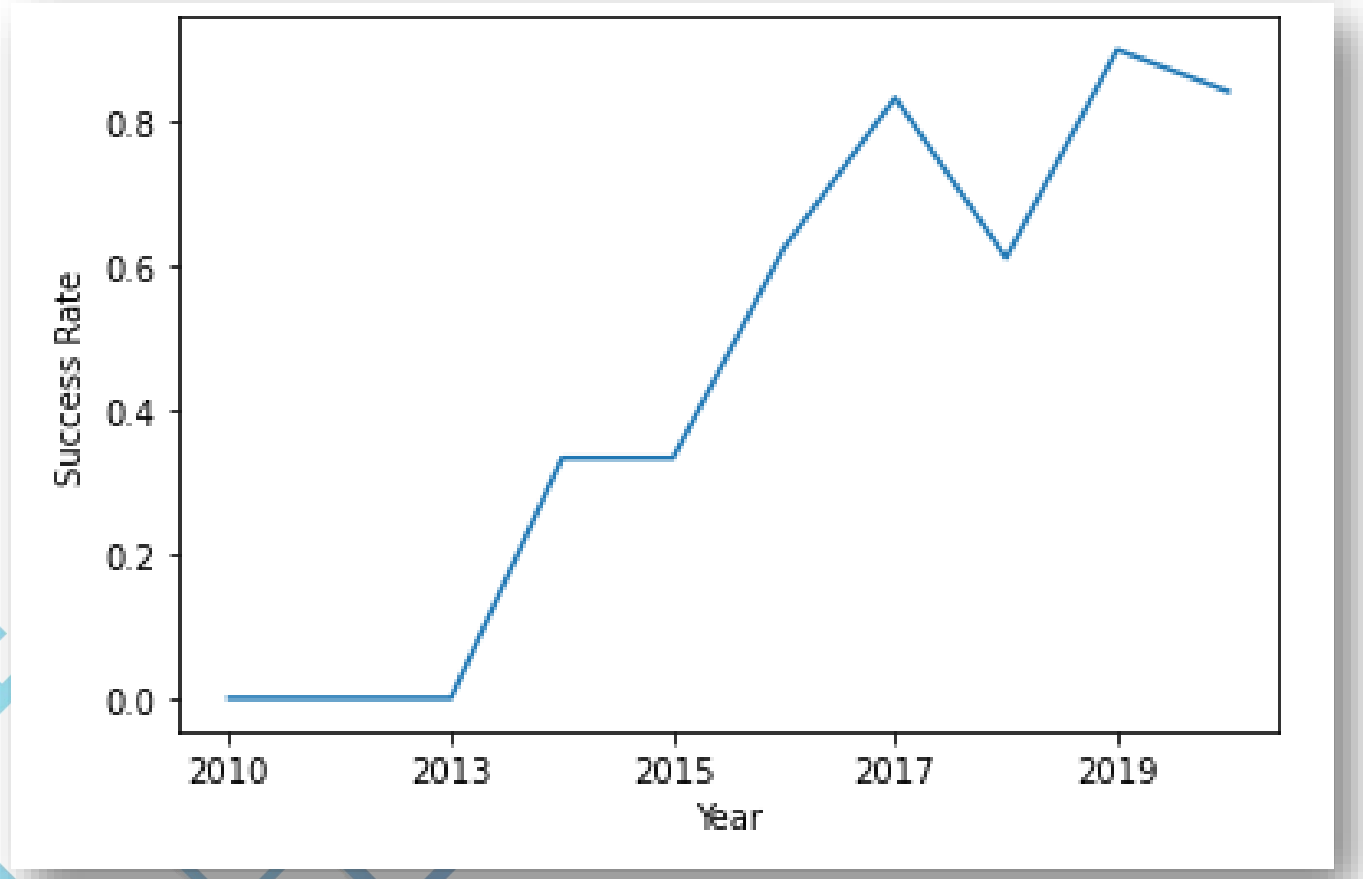
In the LEO orbit, success appears to be related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

Payload vs. Orbit Type



- There are more positive landing rates for PO, LEO and ISS.
- For GTO , one cannot distinguish this well as both positive and negative landing rate are very close to each other.

Launch Success Yearly Trend



Success rate since 2013 kept increasing till 2020 except in 2018.

Section 3

EDA with SQL

All Launch Site Names

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

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

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

The SELECT DISTINCT displays only distinct values in the Launch_Site column from the SpaceX table.

- There are four unique launch sites: CCAFS LC-40, CCAFS SLC-40, KSC LC-39A, VAFB SLC-4E.

Launch Site Names Begin with 'KSC'

Display 5 records where launch site begin with the string ‘KSC’

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

DATE	time_utc_	booster_version	launch_site	payload	payload_mass_kg_	orbit	customer	mission_outcome	landing_outcome
2017-02-19	14:39:00	F9 FT B1031.1	KSC LC-39A	None	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
2017-03-16	06:00:00	F9 FT B1030	KSC LC-39A	None	5600	GTO	EchoStar	Success	No attempt
2017-03-30	22:27:00	F9 FT B1021.2	KSC LC-39A	None	5300	GTO	SES	Success	Success (drone ship)
2017-05-15	23:21:00	F9 FT B1034	KSC LC-39A	None	6070	GTO	Inmarsat	Success	No attempt
2017-06-23	19:10:00	F9 FT B1029.2	KSC LC-39A	None	3669	GTO	Bulsatcom	Success	Success (drone ship)

- Only 5 records were displayed where launch sites begin with the string `KSC`.
- The LIKE operator is used in a WHERE clause to search for launch sites with the string beginning with ‘KSC’.
- LIMIT specifies a limited number of rows. In this case, it is 5 rows.

Total Payload Mass

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

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

total_payload_mass_kg
23589

- The SUM() function is an aggregate function that calculates the sum of column PAYLOAD_MASS__KG_ which is 23589.
- WHERE clause filters only the dataset specifically for customer ‘NASA (CRS)’

Average Payload Mass by F9 v1.1

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

```
%%sql
SELECT AVG(PAYLOAD_MASS__KG_) AS avg_payload_mass_kg
FROM SPACEXTBL
WHERE BOOSTER_VERSION = 'F9 v1.1'
```

avg_payload_mass_kg
2928

- The AVG() function returns the average value of column PAYLOAD_MASS__KG_.
- WHERE clause filters only If booster_version is F9 v1.1
- The average payload is 2928 kg.

First Successful Ground Landing Date

List the date where the first successful landing outcome in drone ship was acheived

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

first_successful_landing_date
2015-12-22

The MIN() function returns the earliest date in the column DATE. The first successful landing date is 2015-12-22.

- WHERE clause filters only if Landing__outcome is Success (ground pad)

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.

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

booster_version
F9 FT B1026
F9 FT B1021.2

The WHERE clause, filters only if Landing__outcome is ‘Success (drone ship)’.

The AND operator displays booster versions if additional condition PAYLOAD_MASS__KG_ is between 4000 and 6000.

For the booster_version, there are two: F9 FT B1026 and F9 FT1021.2

Total Number of Successful and Failure Mission Outcomes

List the total number of successful and failure mission outcomes

```
%%sql
SELECT MISSION_OUTCOME, COUNT(*) AS total_number
FROM SPACEXTBL
GROUP BY MISSION_OUTCOME
```

mission_outcome	total_number
Failure (in flight)	1
Success	55

- The COUNT() function returns the total number of columns.
- The GROUP BY statement, groups rows that have the same values into summary rows, to find the total number in each Mission_outcome.
- Of the 56 mission outcomes, 55 outcomes were a success and only 1 failed.

Boosters Carried Maximum Payload

List the names of the booster which have carried the maximum payload mass. Use a subquery.

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

booster_version	payload_mass__kg_
F9 B5 B1048.5	15600
F9 B5 B1049.7	15600
F9 B5 B1051.3	15600
F9 B5 B1051.4	15600
F9 B5 B1051.6	15600
F9 B5 B1056.4	15600
F9 B5 B1060.3	15600

- The MAX() function returns the largest value of the selected column.
- Then, the maximum value of the payload - MASS__KG_ is displayed using the MAX function.
- The booster version F9 B5 B1048.3 to 7 carry the maximum payload.

2015 Launch Records

List the records which will display the month names, succesful landing_outcomes in ground pad, booster versions, launch_site for the months in year 2017

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

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

- The WHERE clause, filters the dataset if Landing__outcome is Failure (drone ship).
- The AND operator displays a record if all the conditions separated by AND and is 2015.
- Only one landing failure in 2015.

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

Rank the count of successful landing_outcomes between the date 2010-06-04 and 2017-03-20 in descending order.

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

landing__outcome	total_number
Failure (drone ship)	3
No attempt	3
Success (drone ship)	3
Success (ground pad)	3
Controlled (ocean)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

The WHERE clause perform a search if the date is between 2010-06-04 and 2017-03-20.

The ORDER BY keyword sorts the records in ascending order.

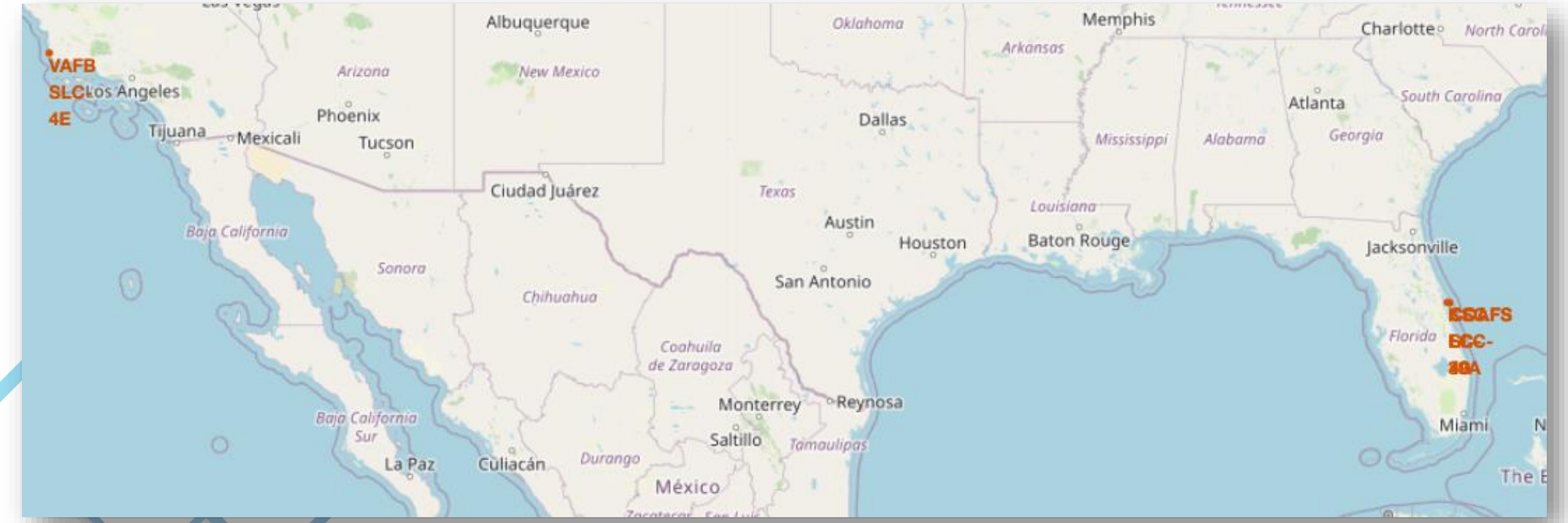
The count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20 was ranked with four landing outcomes having the same total number, two the same total number and lowest is 1.

Section 4

Launch Sites Proximities Analysis

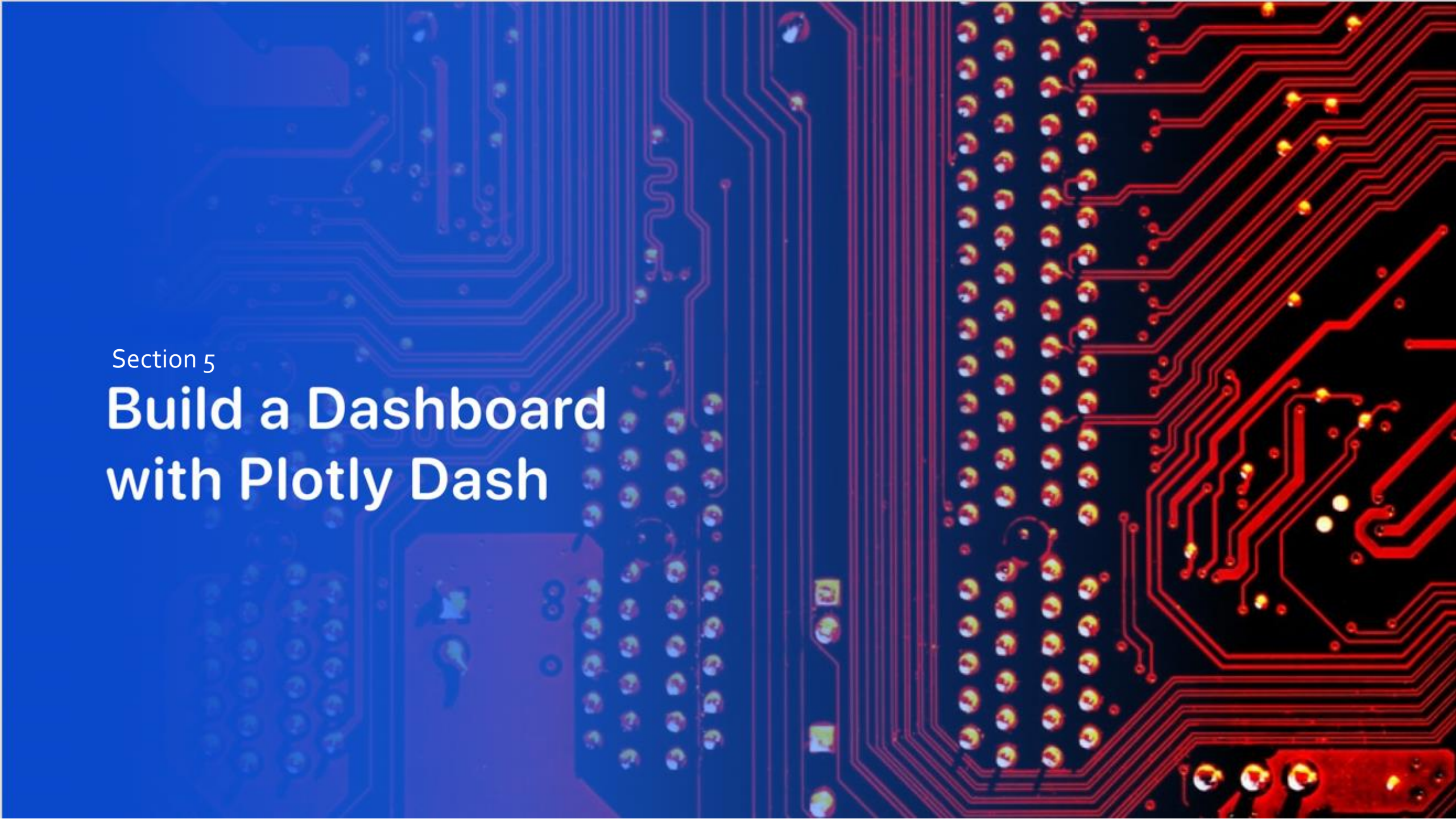
SpaceX Launch Sites

All launch sites are along the coastline.



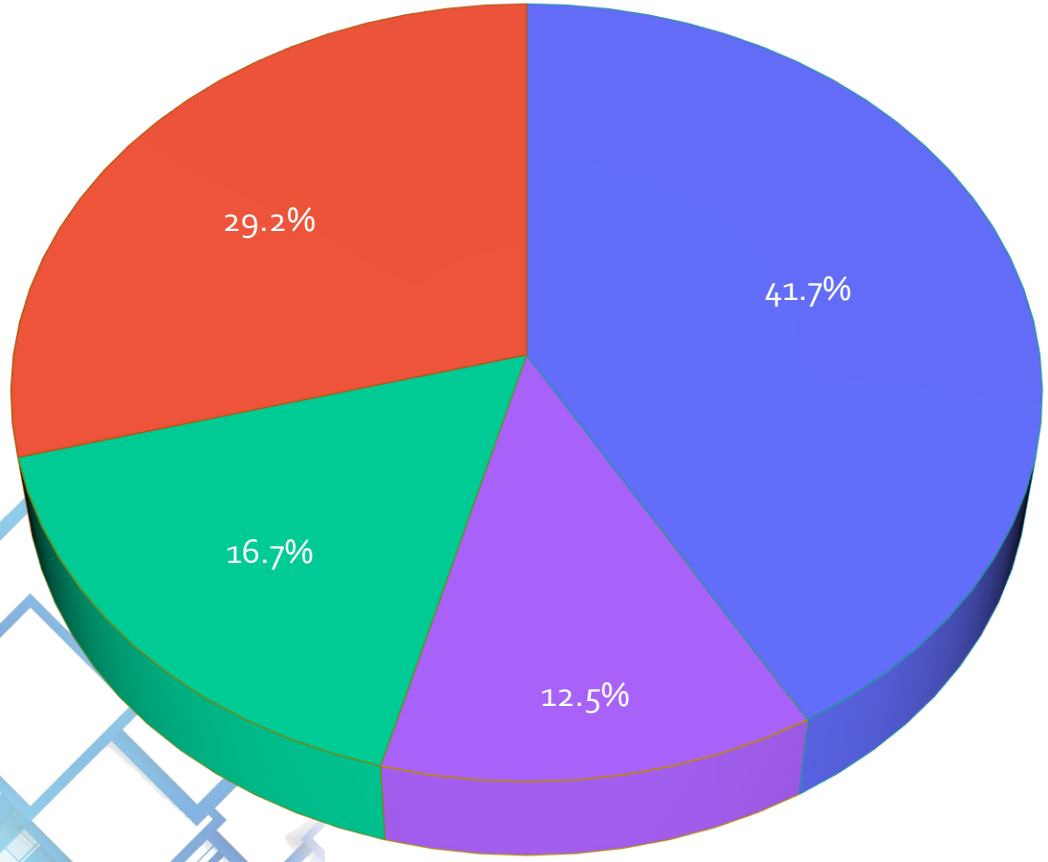
Launch Outcomes

One of the launch sites in Florida.


Section 5
Build a Dashboard with Plotly Dash

SpaceX Launch Records Dashboard

Total Success Launches By Site



KSC LC-39A

CCAFS LC-40

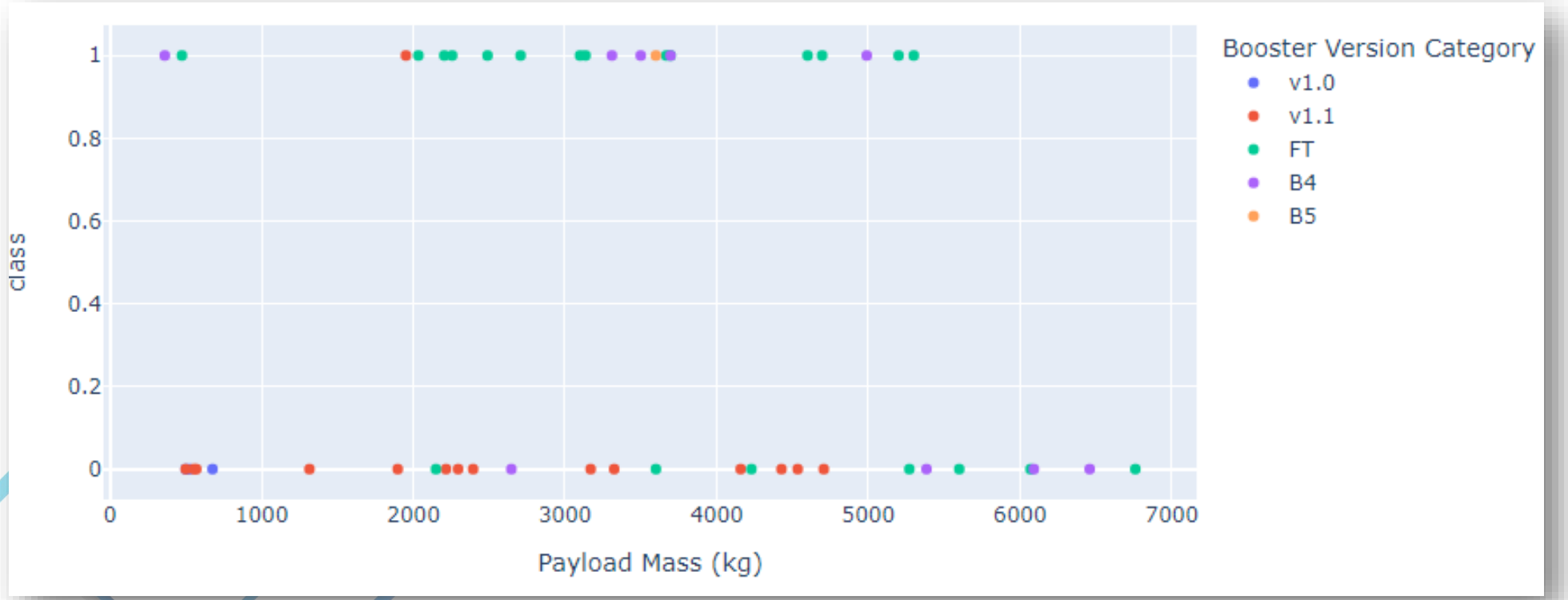
VAFB SLC-4E

CCAFS SLC-40

KSLC-39A has the most successful launches while CCAFS SLC-40 has the least.

Payload vs Launch Outcome Scatterplot for all sites

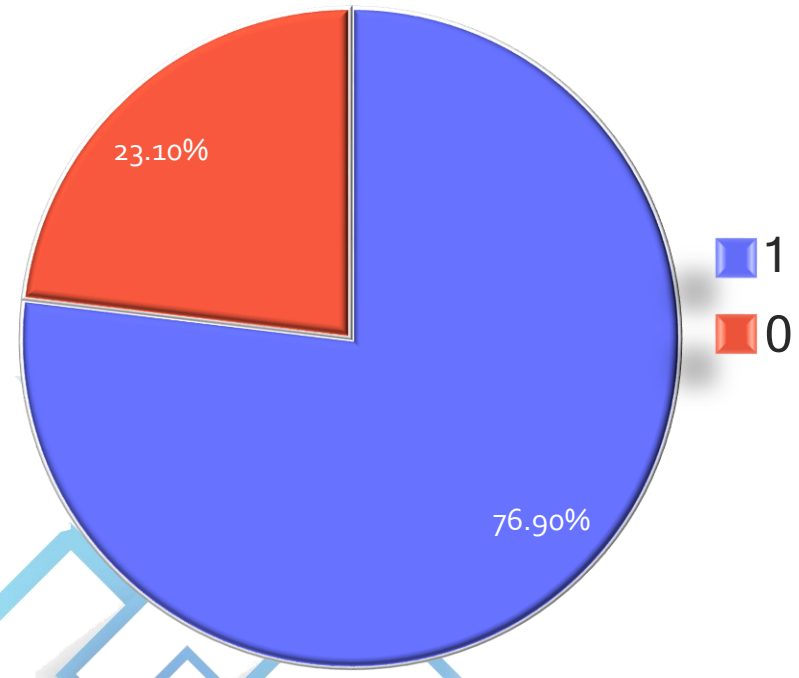
Correlation between Payload and Success for all Site

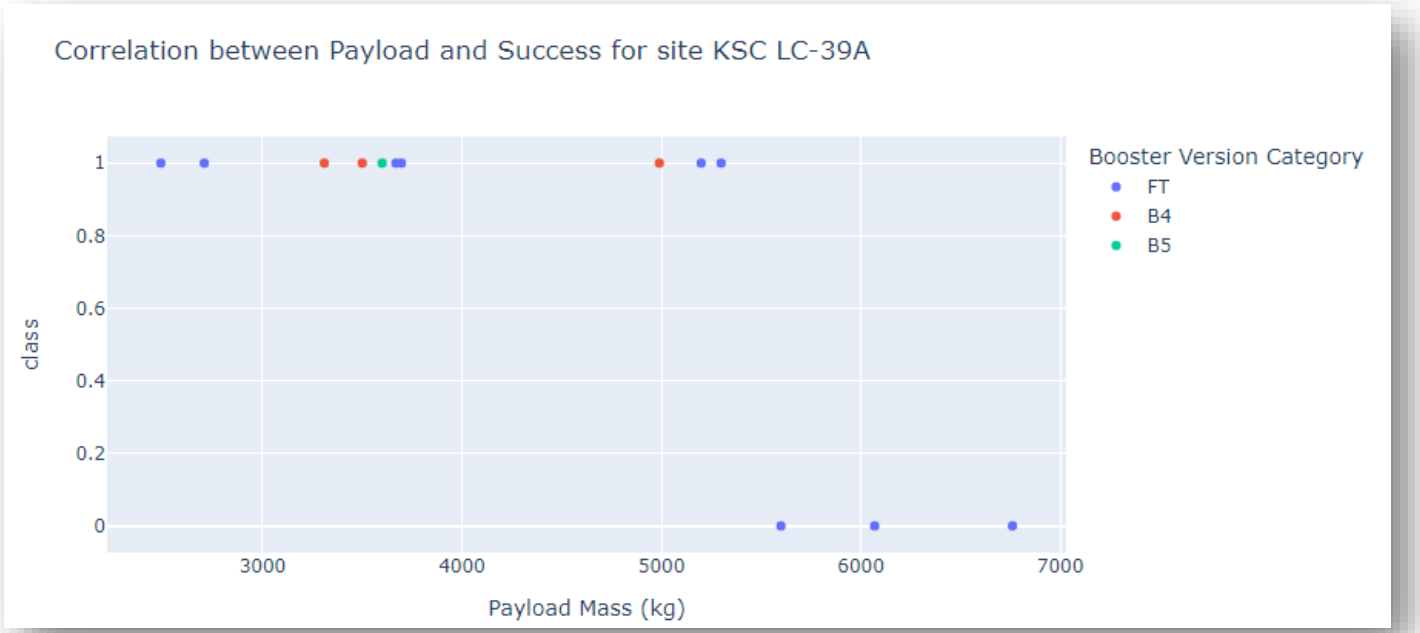


Scatterplot shows that the launch success rate of low weighted payloads(0-5000 kg) is higher than the heavy weighted payloads(5000-10000 kg)

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

Total Success Launched site KSC LC-39A





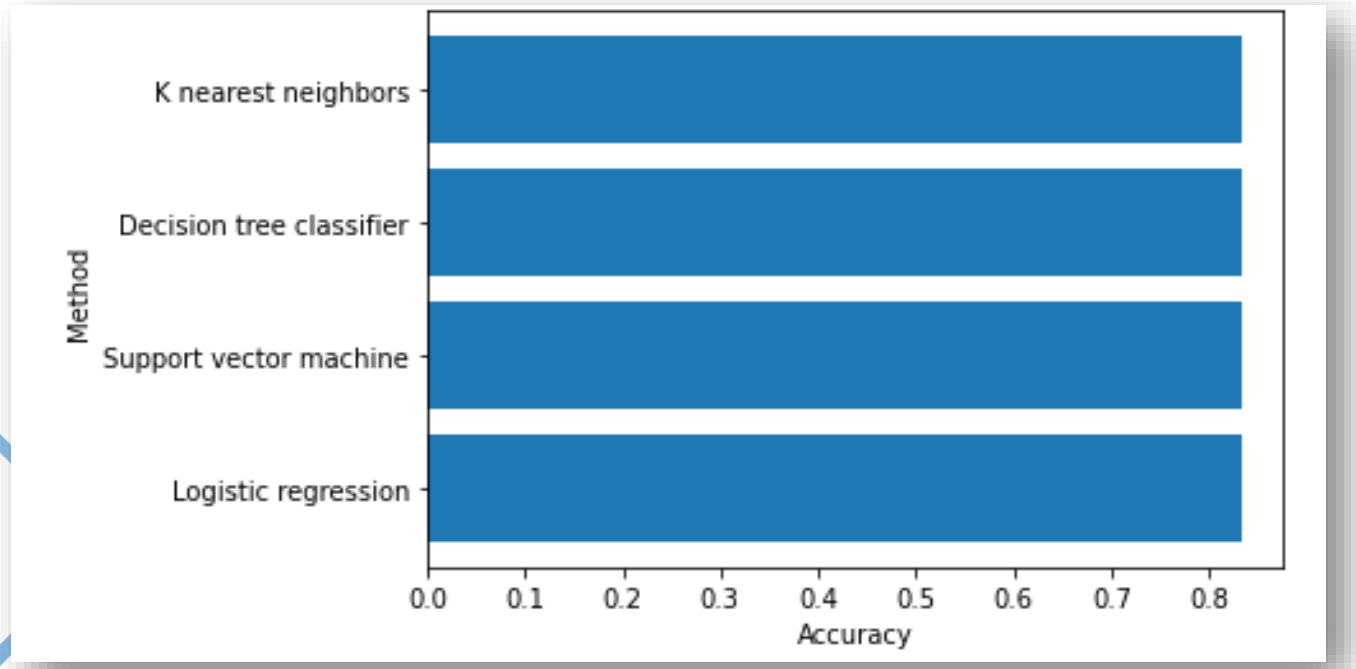
KSLC-39A has 76.9% or 10 landing successes (76.9%) and 23.1% or 3 landing failures. FT has the largest success rate for the booster version category.

Section 5

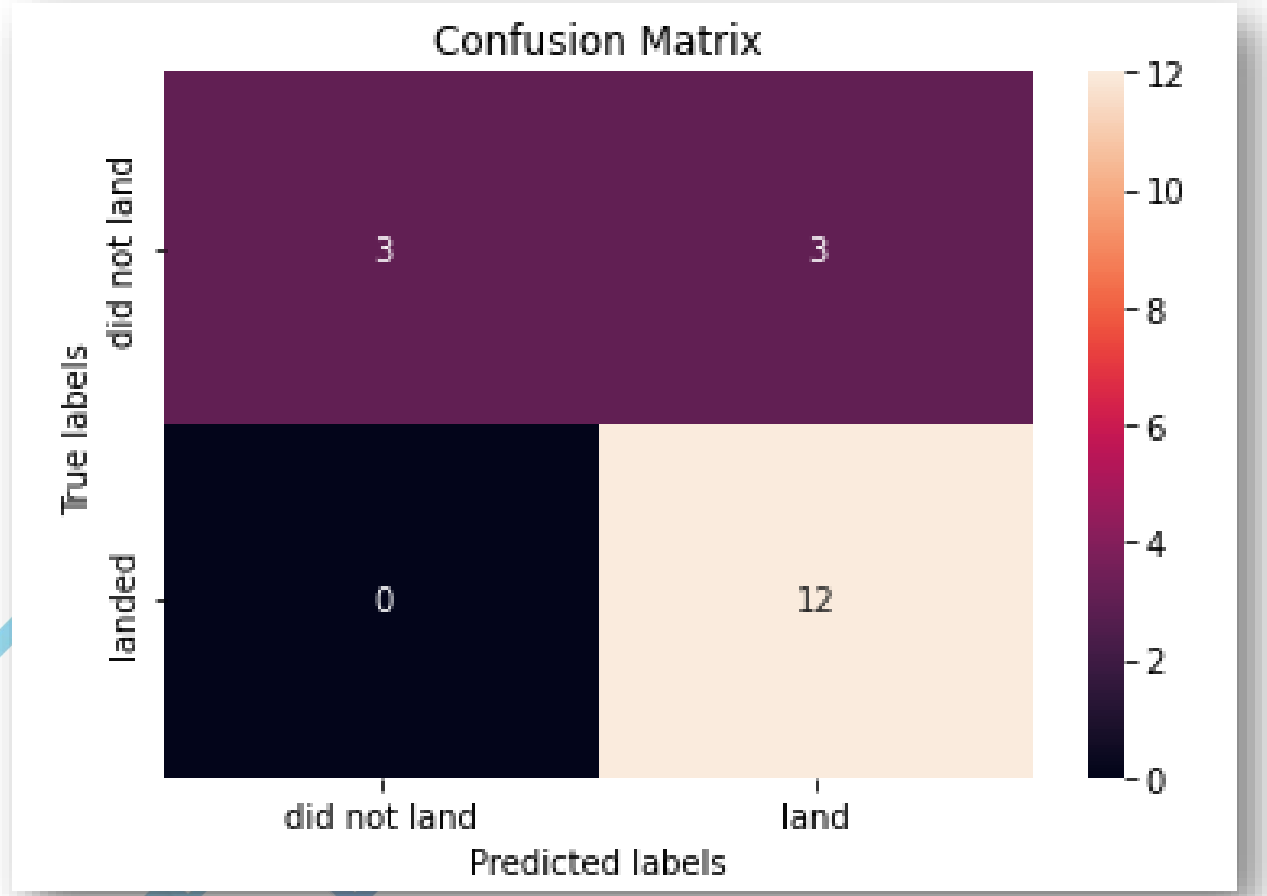
Predictive Analysis (Classification)

Classification Accuracy

- The accuracy rate of all models are the same with test set accuracy of 83.3%.



Confusion Matrix



- All models have the same confusion matrix.
- There are 12 successful landings as shown in the true positive of true labels and predicted labels. There are 3 failed landings under the true negative and 3 successful landings under the false positive of the true labels and predicted labels.
- Succesful landings prevailed for all models.

Conclusions

- The more flights launch, the higher the success rate at a launch site.
- SSO, HEO, GEO and ESL-1 orbit types have the highest success rate (100%).
- Success rate kept increasing since 2013 until 2020 except in 2018.
- All lunch sites are along the coastline.
- KSLC-39A has the most successful launches while CCAFS SLC-40 has the least.
- Launch success rate of low weighted payloads is higher than the heavy weighted payloads.
- All models have the same accuracy rate of 83.33%.
- Successful landings prevailed for all predictive models.

Appendix

Github URL

- edx.org/edX Data Science and Machine Learning Capstone Project

Thank you!