IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Aldo Gonzáles
04/02/2024

Outline

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

Executive Summary

Summary of methodologies:

For Data Collection were used:

- First source used for Data Collection was the Space X REST API that provide us with information about rocket used, payload delivered, launch specifications, landing specifications, and landing outcome.
- To obtain necessary data will use the requests library to obtain launch data of a response in JSON format with .json() method then will be converted into a Dataframe.
- Another source for Data Collection is Web scraping, a Python BeautifulSoup package will be used, which will extract HTML tables.

Executive Summary

Summary of methodologies:

For EDA with Data Visualization we used:

- Scatter plot, Bar chart and Line chart.

For EDA with SQL:

- Next commands were used: SELECT, FROM, WHERE, DISTINCT, AS, AND, BETWEEN, GROUP BY, ORDER BY, DESC, SUM(), AVG(), MIN(), COUNT(), MAX(), SUBSTR().

For Interactive Map with Folium following objects were used:

- Circle, Marker, Popup, MarkerCluster, Mouse Position, PolyLine

Executive Summary

Summary of methodologies:

To Build a Dashboard with Plotly Dash, next plots/graphs and interactions were used:

- Dropdown list, Pie chart, Slider, Scatter chart

For Predictive Analysis (Classification) were used the following models:

- Logistic regression, Support vector machine, Decision tree classifier, K nearest neighbors.
- For choose the best parameters will use the GridSearchCV() object and .score() object for test set sccuracy.

Executive Summary

Summary of all results:

The first results were obtained using the SQL extension:

- SpaceX has four main launch sites, located mainly in the state of Florida, the main Payload mass moving sites are KSC LC-39A and CCAFS SLC-40.
- The total Payload Mass for customer NASA CRS was 45,596 kg, which represents the third Customer in terms of Payload Mass. Including other areas (CCDev, CCP, CCD and CTS), NASA is SpaceX's main customer.
- The average Payload Mass carried by version F9 v1.1 is 2,534.66 kilograms. It is the second to be minor after version F9 v1.0.

Executive Summary

Summary of all results:

- The date of First successful landing in ground pad was 2015-12-22.
- The Booster versions with payload mass between 4,000 to 6,000 kilograms was listed.
- The Number of successful missions is 99 and one mission was failure in flight.
- Twelve Booster Version have the maximum payload mass of 15,600 kilograms.
- 30 landing outcomes was rank between 2010-06-04 and 2017-03-20.
- Interactive analytics were displayed, one for all sites and the other for CCAFS LC-40 site.

Executive Summary

Summary of all results:

The Confusion Matrix was elaborated for four models (Logistic regression, Support vector machine, Decision Tree classifier and K nearest neighbors):

- The best model was the Decision Tree classifier with only one false negative. The best parameters are:
    - Criterion: gini
    - Max depth: 16
    - Max features: log2
    - Min samples leaf: 1
    - Min sample split: 10
    - Splitter: best

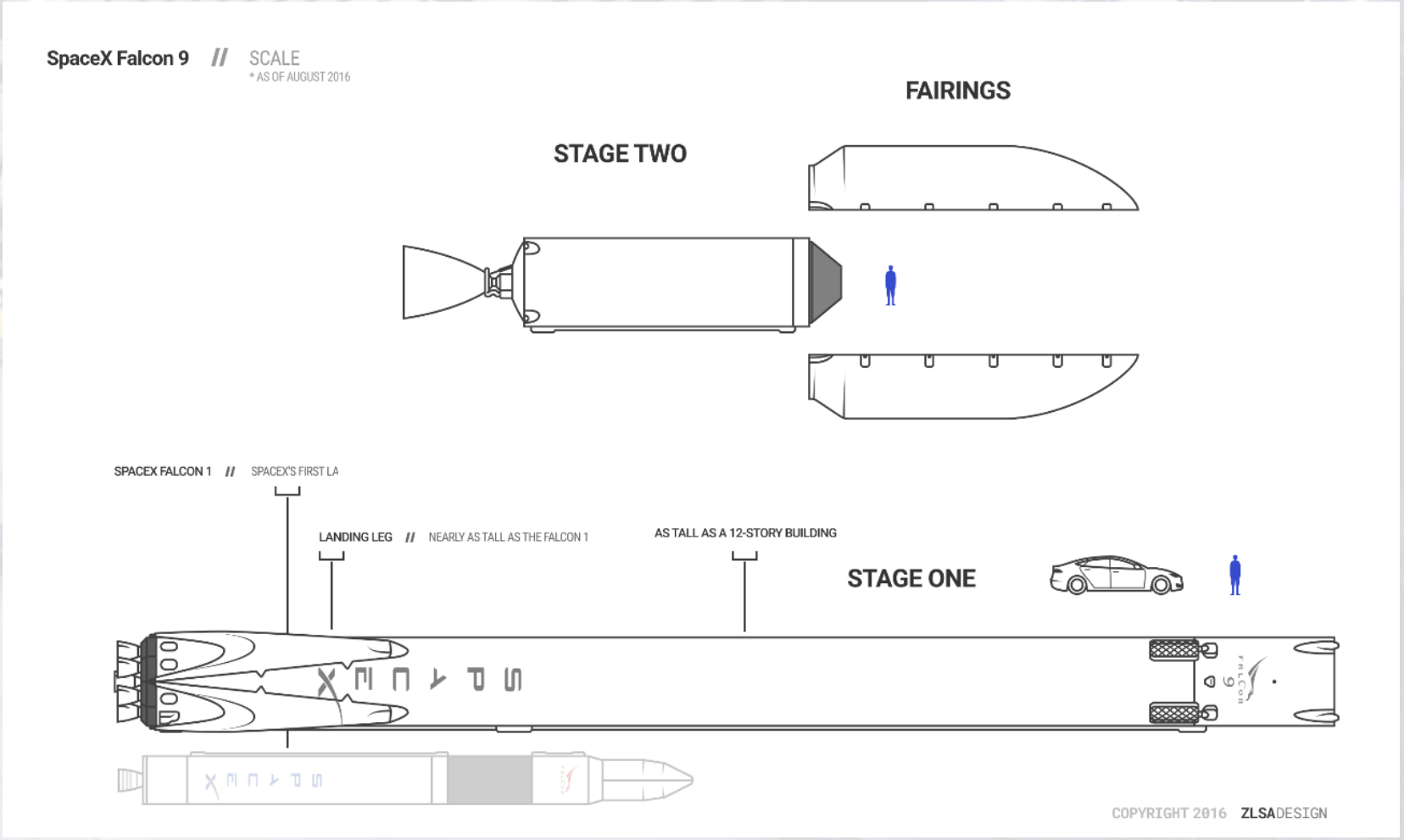
Introduction

Project background and context:

- There are already some companies that offer space travel, to the people and by providing satellites.
- One of the most successful is SpaceX, which announces launches of Falcon 9 rockets at cost of $62 million, while the competition does so at more than $165 million each.
- Part of the savings is because SpaceX is reusing the first stage with some success, lowering the cost of launches.

Introduction

SpaceX Falcon 9 scheme:



Introduction

Want to find answers for:

- The Project purpose is create models to predict success in the first stage landings.
- Knowing the success in landings, we will know the costs involved in missions.

Section 1

Methodology

Methodology

Executive Summary

- Data Collection
- Data Collection – SpaceX API
- Data Collection – Scraping
- Data Wrangling
- EDA with Data Visualization
- EDA with SQL

Methodology

Executive Summary

- Build an Interactive Map with Folium
- Build a Dashboard with Plotly Dash
- Predictive Analysis (Classification)
- Results

Data Collection

Was used three sources:

- Space X REST API that will provide rocket launch information
- The requests library to obtain the launch data of a response in JSON format.
- Web scraping with Python BeautifulSoup package

Data Collection – SpaceX API

For Data Collection were used some APIs like "API_call_spacex_api.json" . Also used some commands to create a Data frame with this data:

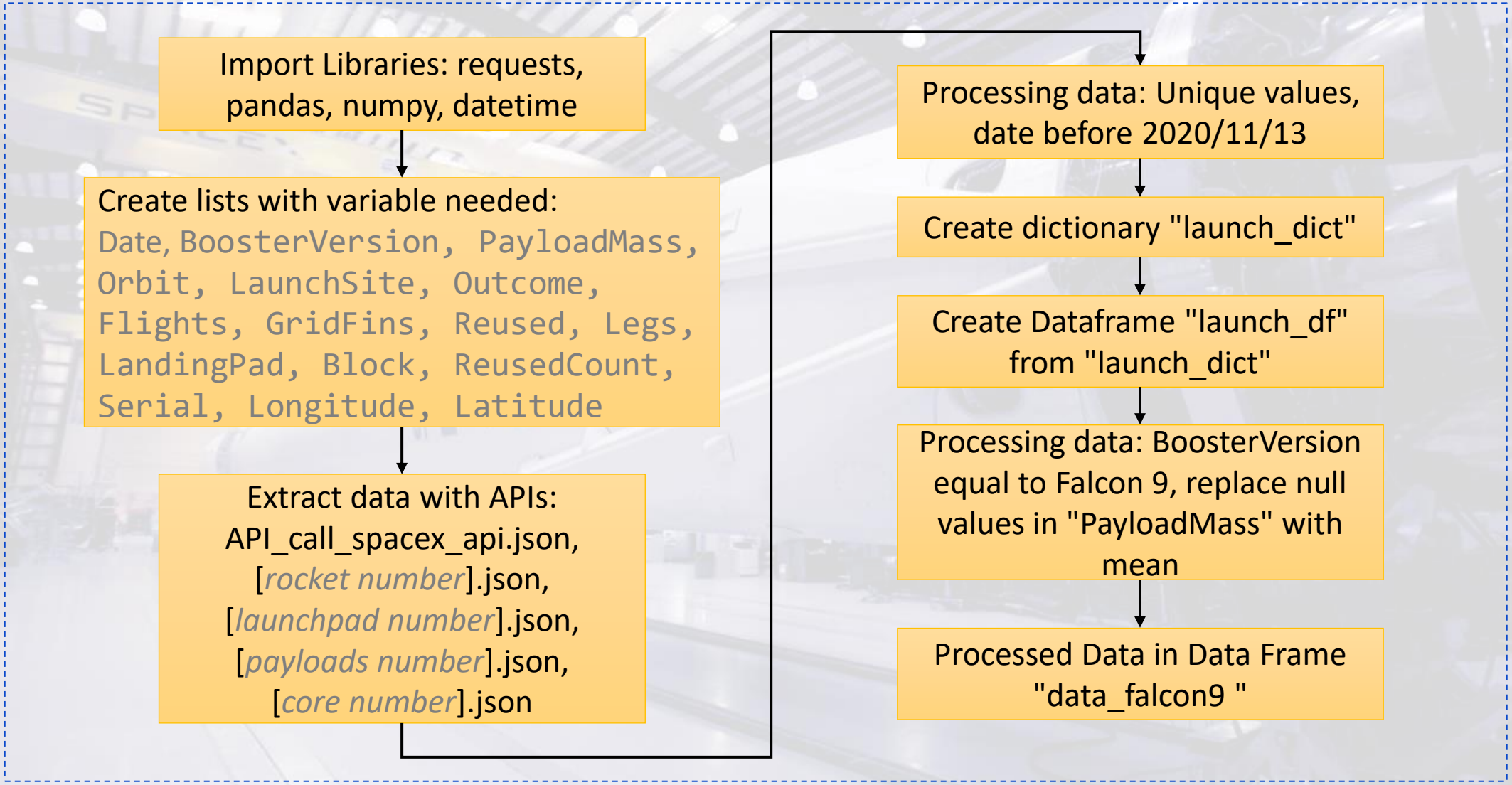
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

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

Also used some others APIs:

- https://api.spacexdata.com/v4/rockets/[*rocket number*].json
- https://api.spacexdata.com/v4/launchpads/[*launchpad number*].json
- https://api.spacexdata.com/v4/payloads/[*payloads number*].json
- https://api.spacexdata.com/v4/cores/[*core number*].json

Data Collection – SpaceX API



Data Collection – SpaceX API

The link to enter the Data Collection – SpaceX API on GitHub is:

https://github.com/AldoGonzales/ADS_Capstone/blob/0050a6d5f51ec3631279e8c88a4626604cfe2866/Week%201%20Lab%201%20Collecting%20the%20data%20.ipynb

Data Collection - Scraping

For Data Collection with Scraping were used GET method for extract data from Wiki page (List of Falcon 9 and Falcon Heavy launches) and assign to a Data frame. Also created a BeautifulSoup object for work with HTML data:

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

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

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

Entire contents of the BeautifulSoup object are assigned to the html_tables variable:

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

A dictionary will be created that will join all the tables that interest us from the Wikipage (launch_dict):

Data Collection - Scraping

```
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']=[]
```

A routine is created that uses the soup.find_all() function that searches for tables with the class "wikitable plainrowheaders collapsible" which are the tables that interest us:

Data Collection - Scraping

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        #verifique si el encabezado de la primera tabla es el número correspondiente al lanzamiento de un número
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
```

The following values are assigned to each row and position of the "launch_dict" dictionary: “Flight No.”, “Date”, “Time”, “Version Booster”, “Launch site”, “Payload”, “Payload mass”, “Orbit”, “Customer”, “Launch outcome” and “Booster landing”

Data Collection - Scraping

Finally we create the Dataframe “df1” with all data from dictionary “launch_dict”:

The link to enter Complete the Data Collection with Web Scraping on GitHub is:

https://github.com/AldoGonzales/ADS_Capstone/blob/main/Week%201%20Data%20Collection%20with%20Web%20Scraping.ipynb

Data Collection – Scraping

Import Packages: beautifulsoup4 and requests

Import Libraries: sys, requests, BeautifulSoup, re, Unicode data y pandas

Use GET method to extract data from Wiki page

A BeautifulSoup object is created

Obtain column titles with extract_column_from_header() function that extracts the information from the rows with label “th” and assigns it to the column_names list

Create Dictionary “launch_dict” that will join all tables that interest us from the Wiki page

A routine is created that uses the soup.find_all() function to extract data from tables with class "wikitable plainrowheaders collapsible" and assign to dictionary "launch_dict"

Finally we assign the content of the dictionary “launch_dict” to the Data frame “df1”

Data Wrangling

After import libraries pandas and numpy import CSV file and assign to Data frame "df":

```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_1.csv")
```

Below are some reports obtained from the Data frame:

- 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
```

Data Wrangling

- Number and occurrence of each orbit (more explanation at Appendix - Orbit Types)

```
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
```

- Number and occurrence of mission outcome of the orbits:

```
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
```

Data Wrangling

- 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)
```

```
df['Class']=landing_class
df[['Class']].head(10)
```

	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1
8	0
9	0

- Determine the success rate:

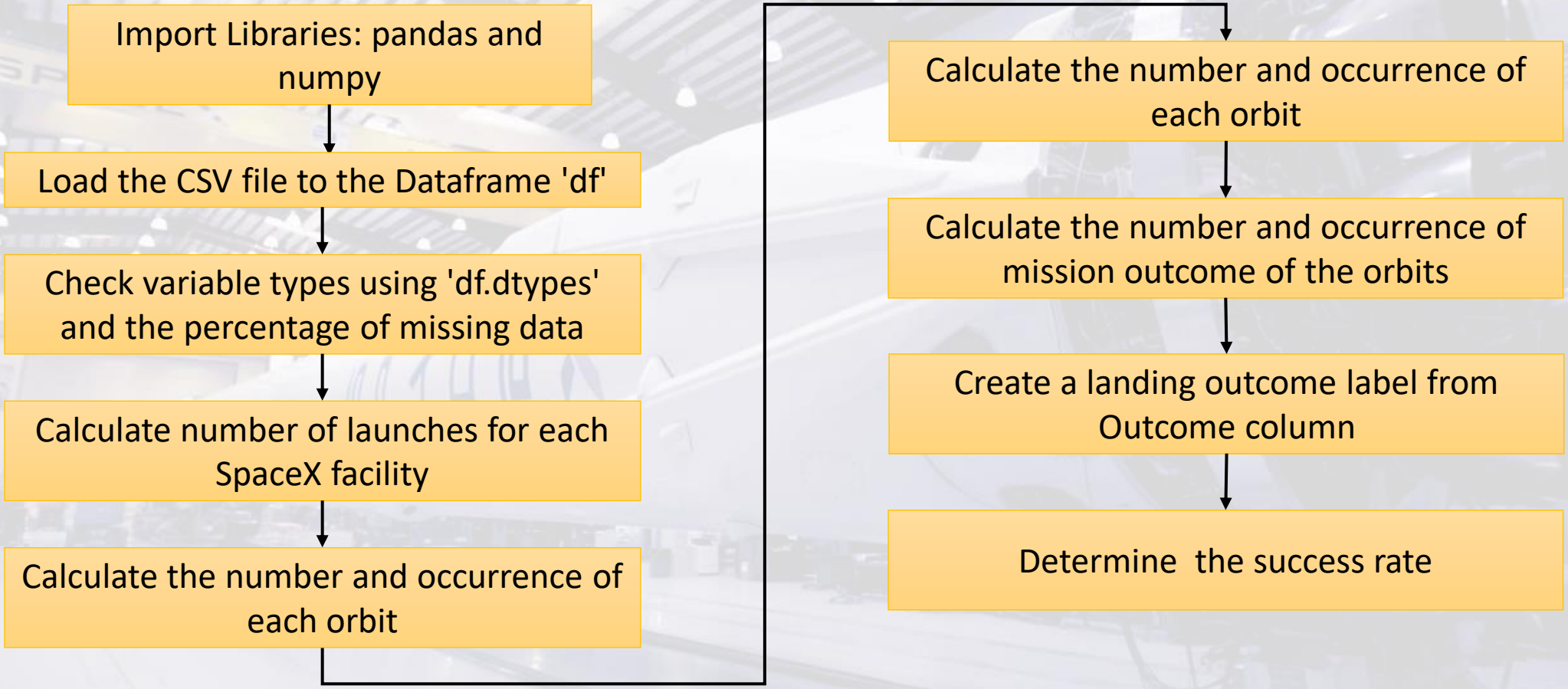
```
df["Class"].mean()
```

```
0.6666666666666666
```

The link to enter Complete the Data Collection with Web Scraping on GitHub is:

https://github.com/AldoGonzales/ADS_Capstone/blob/main/Week%201%20Lab%202%20Data%20wrangling.ipynb

Data Wrangling



EDA with Data Visualization

The plots used to visualize obtained data are the following:

- *Scatter plot - “FlightNumber” vs “Payload”*
- *Scatter plot - “FlightNumber” vs “Launch Site”*
- *Scatter plot - “Payload Mass (kg)” y “Launch Site”*
- *Bar chart - “Orbit type” with Success rate*
- *Scatter plot - “Flight Number” vs “Orbit Type”*
- *Scatter plot - “Payload Mass” vs “Orbit Type”*
- *Line chart – “Yearly Success rate”*

EDA with Data Visualization

The plots used to visualize obtained data are the following:

- Scatter plot was used because it's the best for showing the relationship between two variables in a cartesian plane.
- The bar chart is ideal for comparing categorical values like Orbit Type.
- The Line chart was used for showing time series such as the year in this case.

The link to enter completed EDA with data visualization notebook on GitHub is:

https://github.com/AldoGonzales/ADS_Capstone/blob/main/Week%202%20EDA%20with%20Visualization%20Lab.ipynb

EDA with SQL

Next the SQL performed queries using Database in CSV format:

- Display the names of launch sites:

```
%sql create table SPACEXTABLE as select * from SPACEXTBL where Date is not null
```

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

```
%sql SELECT * from SPACEXTABLE 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_) FROM SPACEXTABLE WHERE Customer = 'NASA (CRS)';
```

EDA with SQL

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

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Booster_Version LIKE 'F9 v1.1%';
```

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

```
%sql SELECT MIN(Date) FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (ground pad)';
```

- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000:

```
%sql SELECT Booster_Version FROM SPACEXTABLE WHERE LANDING_OUTCOME = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;
```

EDA with SQL

- List the total number of successful and failure mission outcomes:

```
%sql SELECT Mission_Outcome, COUNT(Mission_Outcome) AS TOTAL_NUMBER FROM SPACEXTABLE GROUP BY Mission_Outcome;
```

- List the names of the booster_versions which have carried the maximum payload mass:

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

- 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 SUBSTR(Date,6,2) AS Month, Landing_Outcome, Booster_Version, Launch_Site FROM SPACEXTABLE WHERE Landing_Outcome = 'Failure (drone ship)' AND SUBSTR(Date,1,4) = '2015';
```

EDA with SQL

- Rank the count of landing outcomes between the date 2010-06-04 and 2017-03-20, in descending order:

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

The link to enter SQL Notebook on GitHub is:

https://github.com/AldoGonzales/ADS_Capstone/blob/main/Week%202%20Assignment%20SQL%20Notebook%20for%20Peer%20Assignment.ipynb

Build an Interactive Map with Folium

The next table summarize the map objects used:

Circle	Marker	Popup	MarkerCluster	Mouse Position	PolyLine	Purpose of use
✓	✓	✓				NASA Johnson Space Center's coordinate
✓	✓	✓				Launch sites
	✓		✓			Launch's success
				✓		Coordinates in mouse position
	✓				✓	Distance coastline
	✓				✓	Closest highway
	✓				✓	Closest railroad
	✓				✓	Closest city

Build an Interactive Map with Folium

The following objects were used for:

- **Marker:** Is used for show important places on map.
- **Popup:** This object is used for unfold information when click.
- **Circle:** This object is used for show a circle around given coordinates.
- **MakerCluster:** Is used for clustering sites with numerical values showing the total sum when map is zoom out.
- **Mouse Position:** This object is used for showing latitude and longitude information on map.
- **PolyLine:** Is used for draw a line on map. In this case was used for show distance between two points on map.

Build an Interactive Map with Folium

The link to enter to Launch Sites Locations Analysis with Folium on GitHub is:

https://github.com/AldoGonzales/ADS_Capstone/blob/main/Week%203%20Launch%20Sites%20Locations%20Analysis%20with%20Folium.ipynb

Build a Dashboard with Plotly Dash

The plots/graphs and interactions used:

- Un Dropdown list for choosing one launch site or all sites from a list so can show a Pie chart graph with this selected site.
- A Pie chart showing percentage of successful launches on each site. Or the success rate in launching each site if "All Sites" is selected.
- A Slider to select a range for the Payload between zero and 10K kilograms. It can choose a minimum and a maximum value.
- A Scatter chart showing the relationship between payload mass and launch success. The chart also show the Booster version category using colors.

Build a Dashboard with Plotly Dash

The link to enter to Plotly Dash lab on GitHub is:

https://github.com/AldoGonzales/ADS_Capstone/blob/main/Week%203%20Build%20an%20Interactive%20Dashboard%20with%20Ploty%20Dash.py

Predictive Analysis (Classification)

- 'data' Dataframe was created from CSV file:

```
data = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_2.csv")
```

- ‘X' Dataframe was created from CSV file. This is the independent variables array that need to be transformed for standardize values:

```
X = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_3.csv')
```

```
transform = preprocessing.StandardScaler()

X = transform.fit_transform(X)
```

- 'data' Dataframe was created from CSV file:

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

Predictive Analysis (Classification)

We are going to test four classification models:

- Logistic regression
- Support vector machine
- Decision tree classifier
- K nearest neighbors

Each classification model will be tested with different parameters to choose the best parameters using the GridSearchCV() object.

Once the best parameters for each model have been chosen, they will be tested on the training data.

Predictive Analysis (Classification)

- Logistic regression:

```
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)
```

- Support vector machine:

```
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)
```

Predictive Analysis (Classification)

- Decision tree classifier:

```
parameters = {'criterion': ['gini', 'entropy'],
     'splitter': ['best', 'random'],
     'max_depth': [2*n for n in range(1,10)],
     'max_features': ['log2', 'sqrt'],
     'min_samples_leaf': [1, 2, 4],
     'min_samples_split': [2, 5, 10],
     'splitter': ['best', 'random']}

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

- K nearest neighbors:

```
parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
              'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
              'p': [1,2]}

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

Predictive Analysis (Classification)

Finally, the accuracy will be calculated with the Test data for each model:

- Logistic regression:

```
LR_accuracy = logreg_cv.score(X_test, Y_test)

print("test set accuracy :",LR_accuracy)
```

```
test set accuracy : 0.8333333333333334
```

- Support vector machine:

```
SVM_accuracy = svm_cv.score(X_test, Y_test)
print("test set accuracy :",SVM_accuracy)
```

```
test set accuracy : 0.8333333333333334
```

- Decision tree classifier:

```
DTree_accuracy = tree_cv.score(X_test, Y_test)
print("test set accuracy :",DTree_accuracy)
```

```
test set accuracy : 0.9444444444444444
```

Predictive Analysis (Classification)

Finally, the accuracy will be calculated with the Test data for each model:

- K nearest neighbors:

```
KNN_accuracy = knn_cv.score(X_test, Y_test)
print("test set accuracy :",KNN_accuracy)
```

```
test set accuracy : 0.8333333333333334
```

Results

- Exploratory data analysis results:

LAUNCH SITES:

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

TOTAL PAYLOAD MASS LAUNCHED BY NASA (CRS):

SUM(PAYLOAD_MASS__KG_)
45596

Results

AVERAGE PAYLOAD MASS CARRIED BY VERSION F9 v1.1:

AVG(PAYLOAD_MASS__KG_)
2534.6666666666665

FIRST SUCCESSFUL LANDING IN GROUND PAD DATE:

MIN(Date)
2015-12-22

BOOSTERS SUCCESS IN DRONE SHIP WITH PAYLOAD MASS GREATER THAN 4000 BUT LESS THAN 6000:

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

Results

NUMBER OF SUCCESSFUL AND FAILURE MISSION OUTCOMES:

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

BOOSTER VERSIONS WHICH HAVE CARRIED THE MAXIMUM PAYLOAD MASS:

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

Results

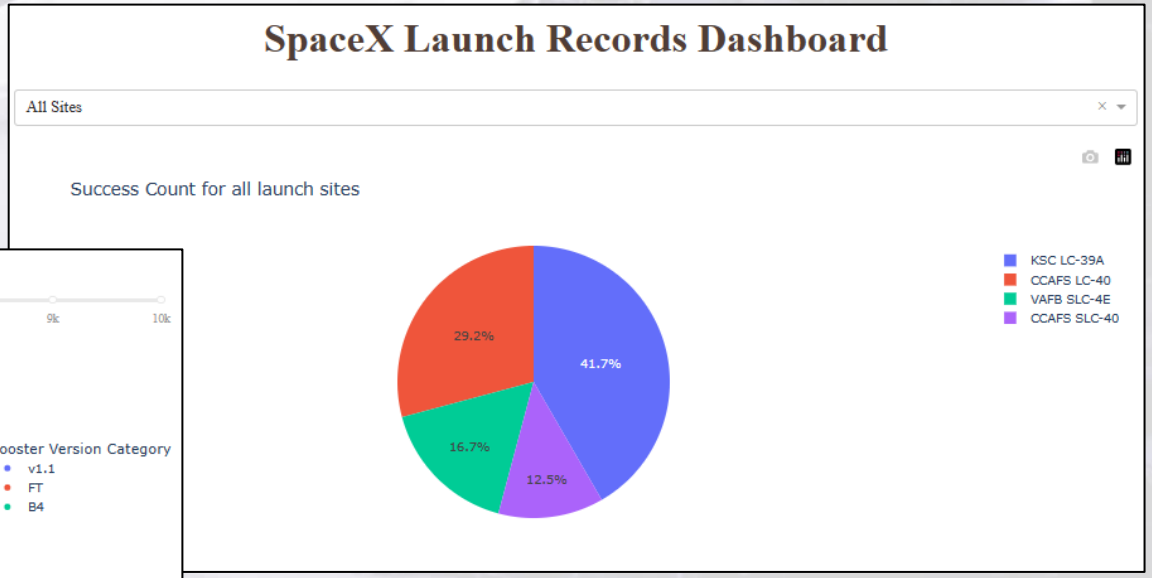
RANK THE COUNT OF LANDING OUTCOMES BETWEEN THE DATE 2010-06-04 AND 2017-03-20:

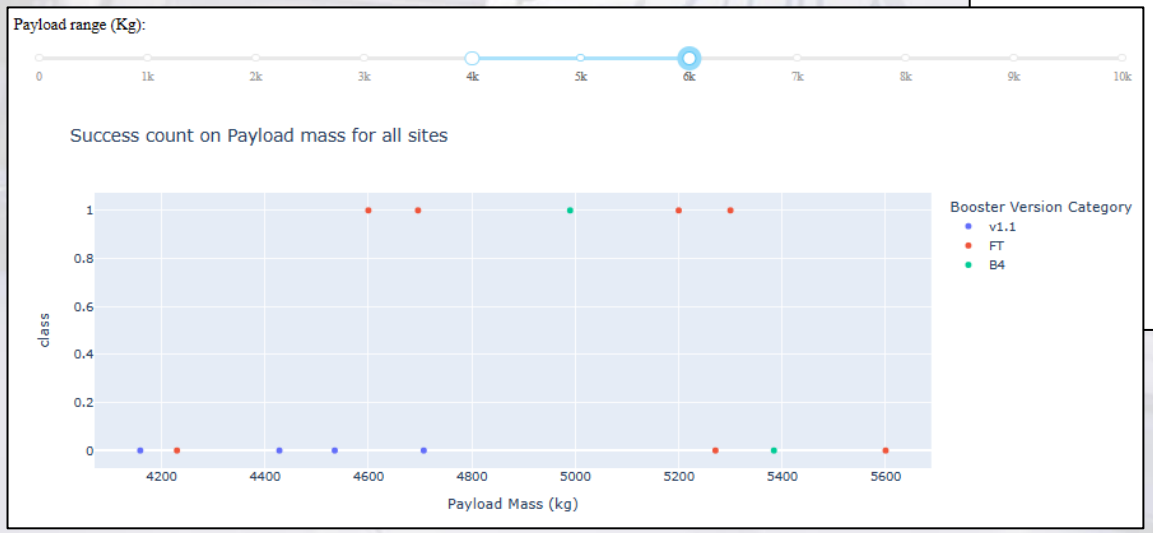
Landing_Outcome	TOTAL_LANDING
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Results

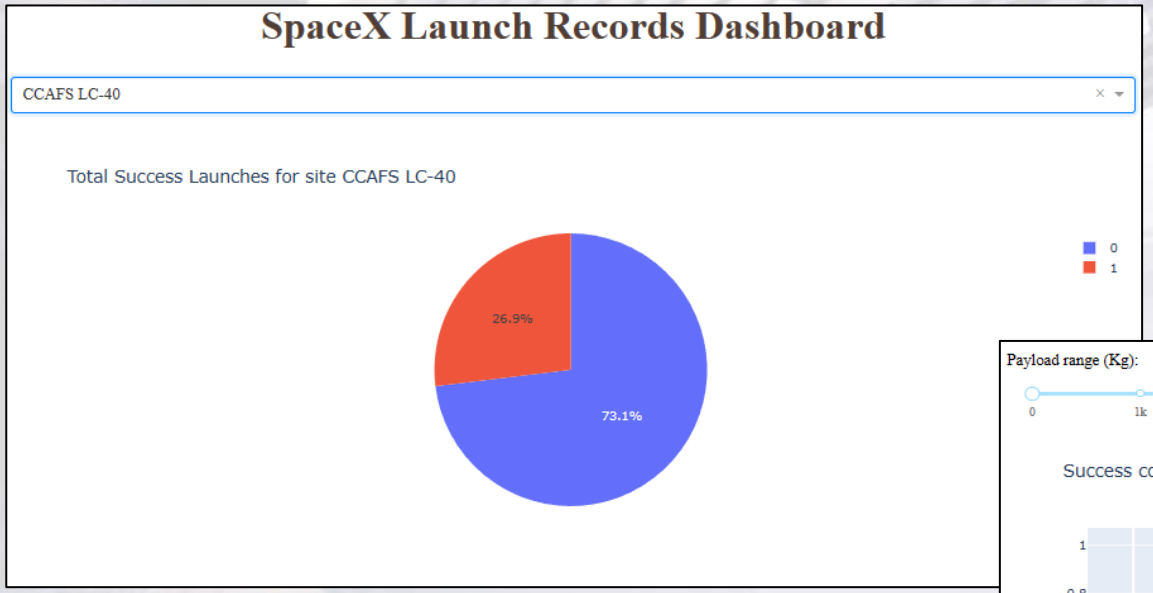
- Interactive analytics demo in screenshots

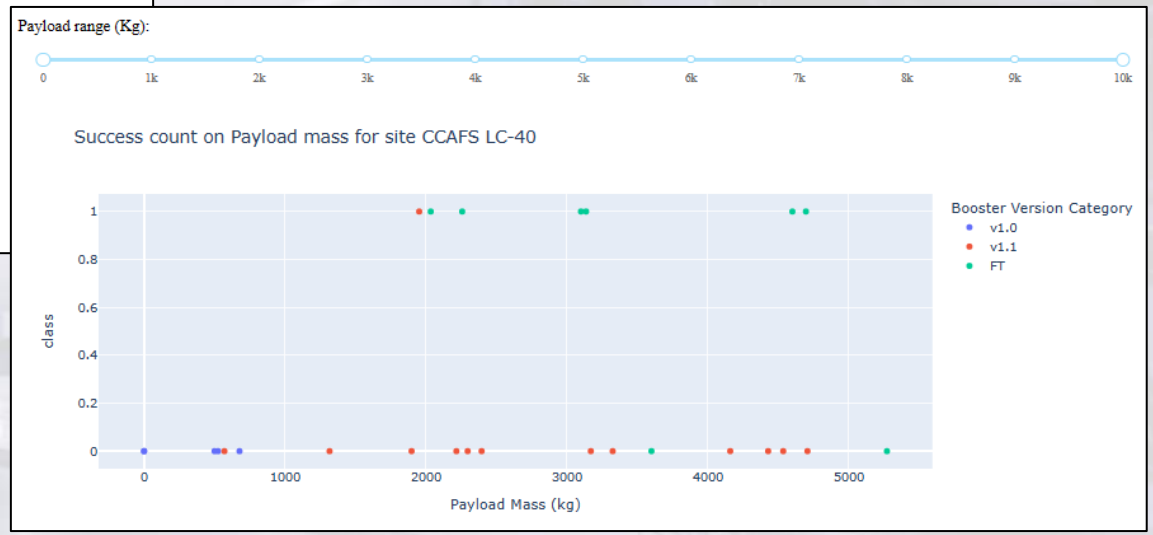
Next some screenshot examples of Interactive analytics:





Results


SpaceX Launch Records Dashboard
CCAFS LC-40
Total Success Launches for site CCAFS LC-40
26.9%
73.1%
0
1


Payload range (Kg):
0
1k
2k
3k
4k
5k
6k
7k
8k
9k
10k
Success count on Payload mass for site CCAFS LC-40
class
Payload Mass (kg)
Booster Version Category
v1.0
v1.1
FT

Results

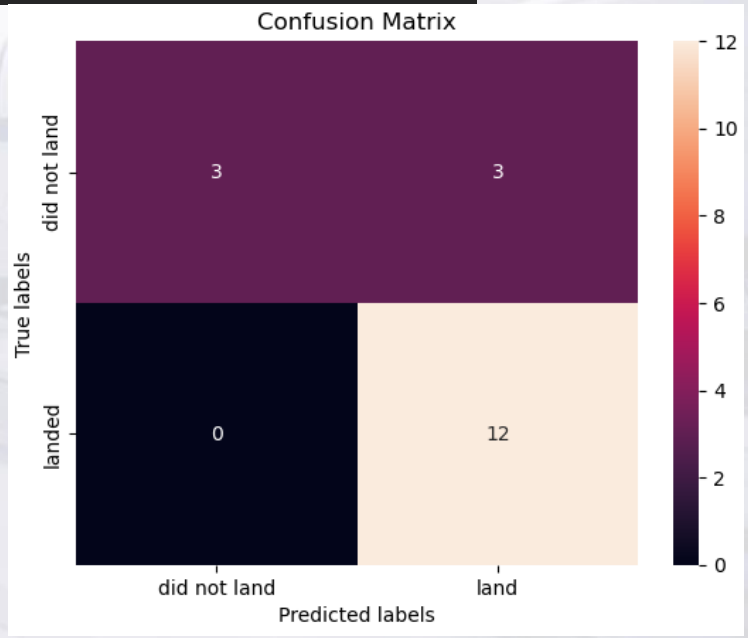
- Predictive analysis results

Four models were used and their best parameters were chosen:

a) Logistic regression:

```
tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
```

```
yhat=logreg_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

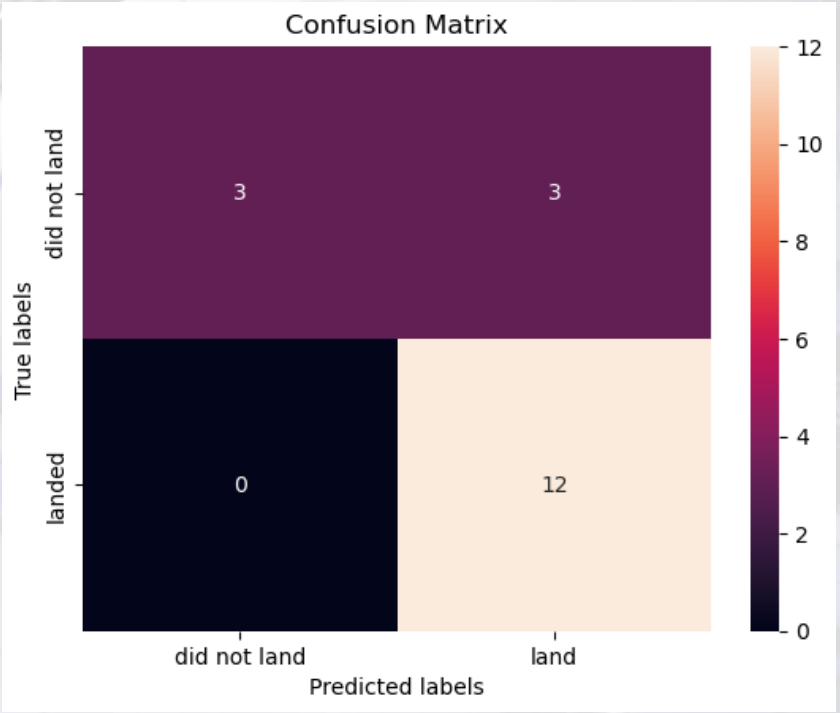


Results

b) Support vector machine:

```
tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
```

```
yhat=svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

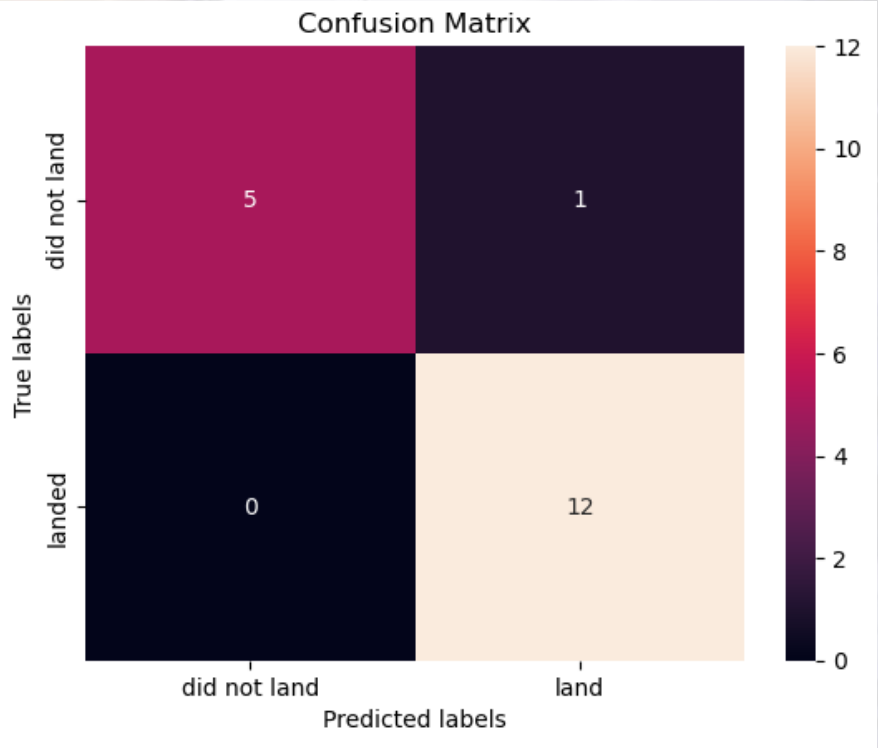


Results

c) Decision tree classifier:

```
tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 12, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 10, 'splitter': 'best'}
```

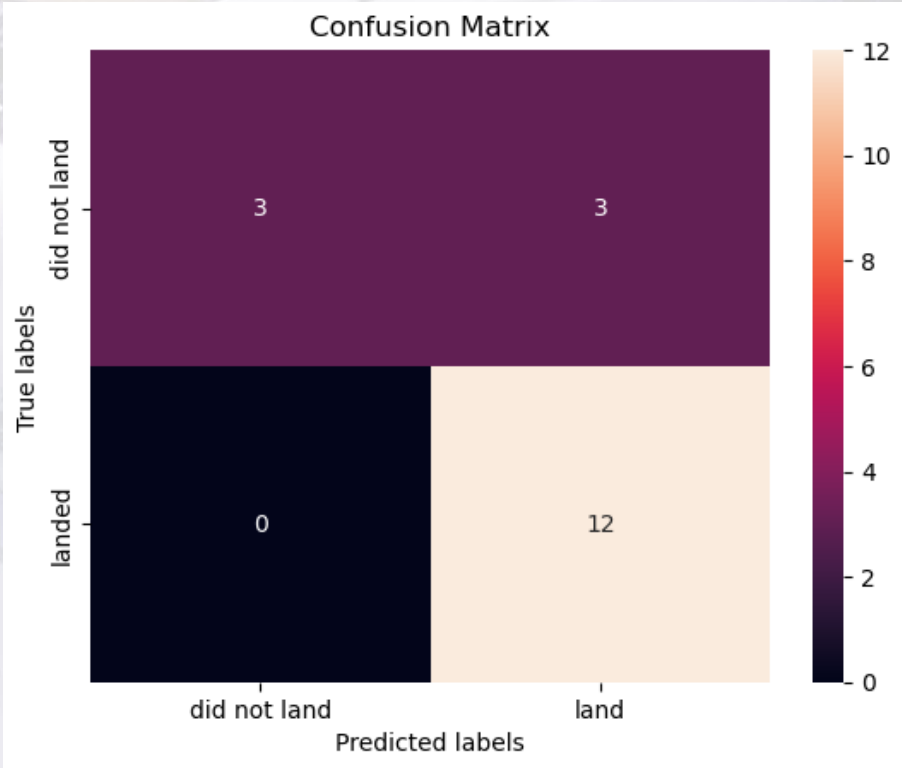
```
yhat = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

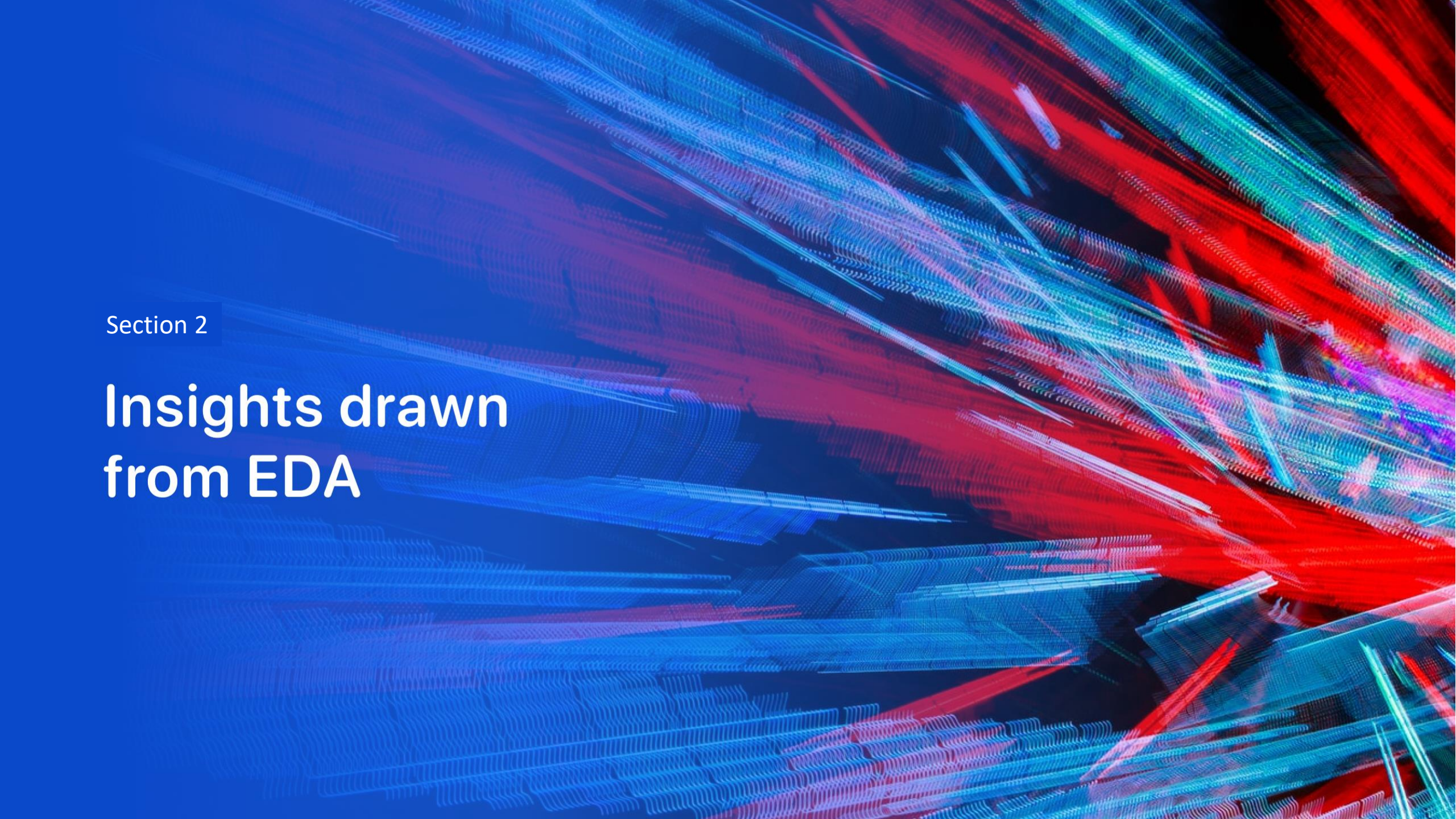


Results

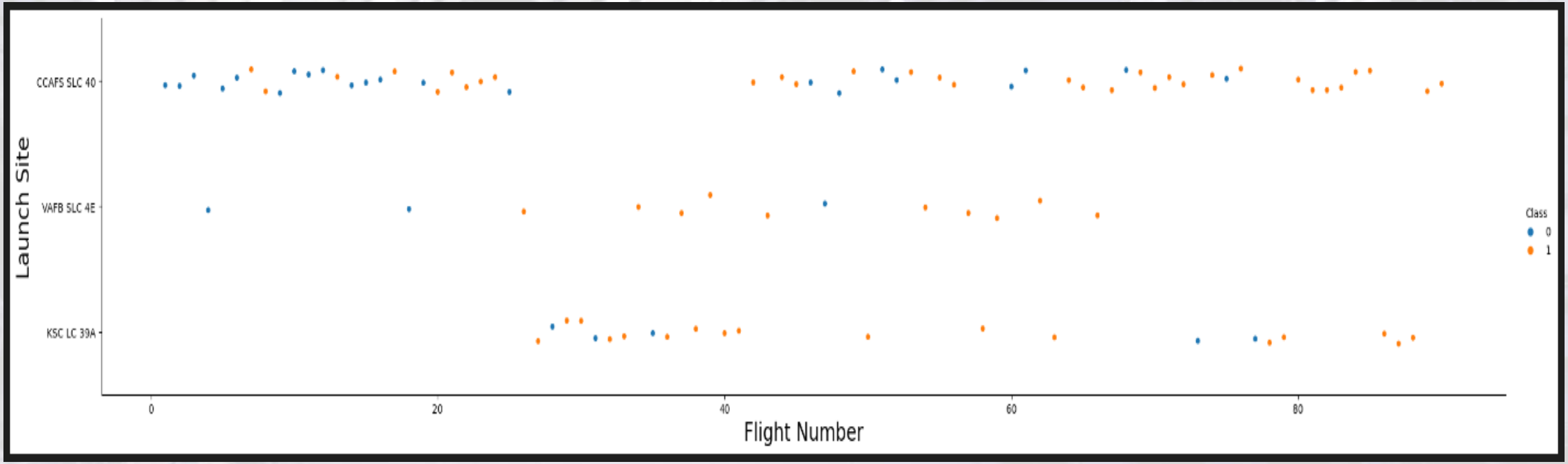
d) K nearest neighbors:

```
tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
yhat = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



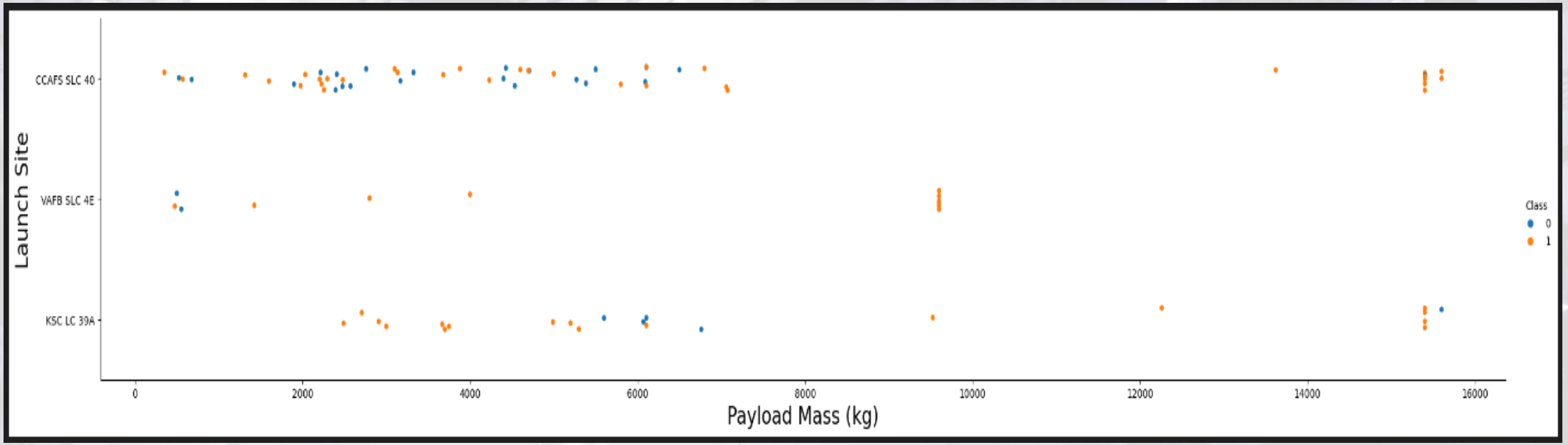

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



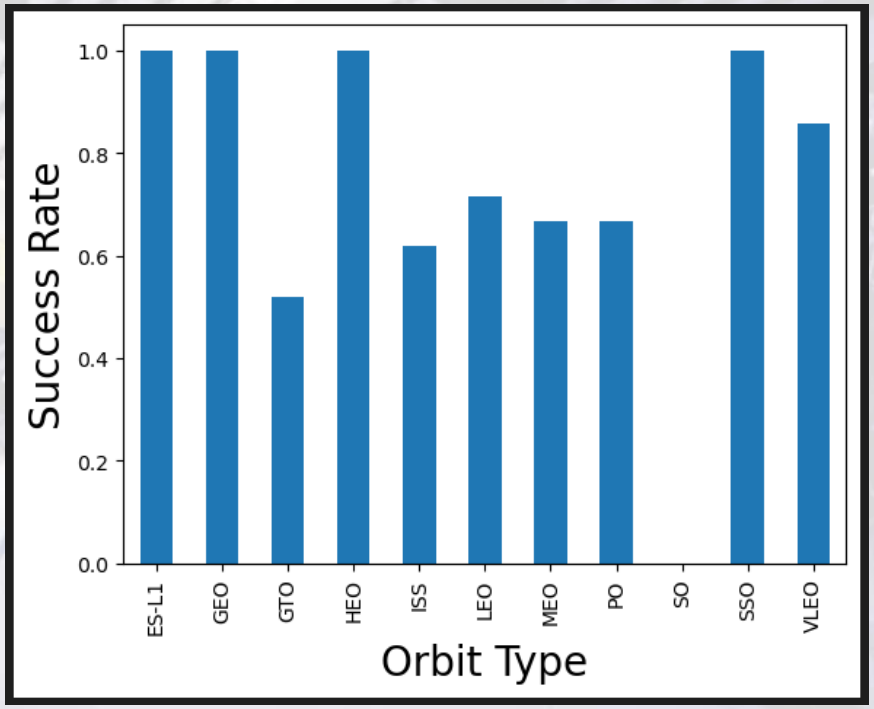
- The flight numbers of the different missions shows on X axis.
- The Launch Sites (CCAFS-SLC-40, WAFB-SLC-4E and KSC-LC-39A) are shown on Y axis.
- The color of each point on the graph shows in orange whether it succeeded or failed in blue.

Payload vs. Launch Site



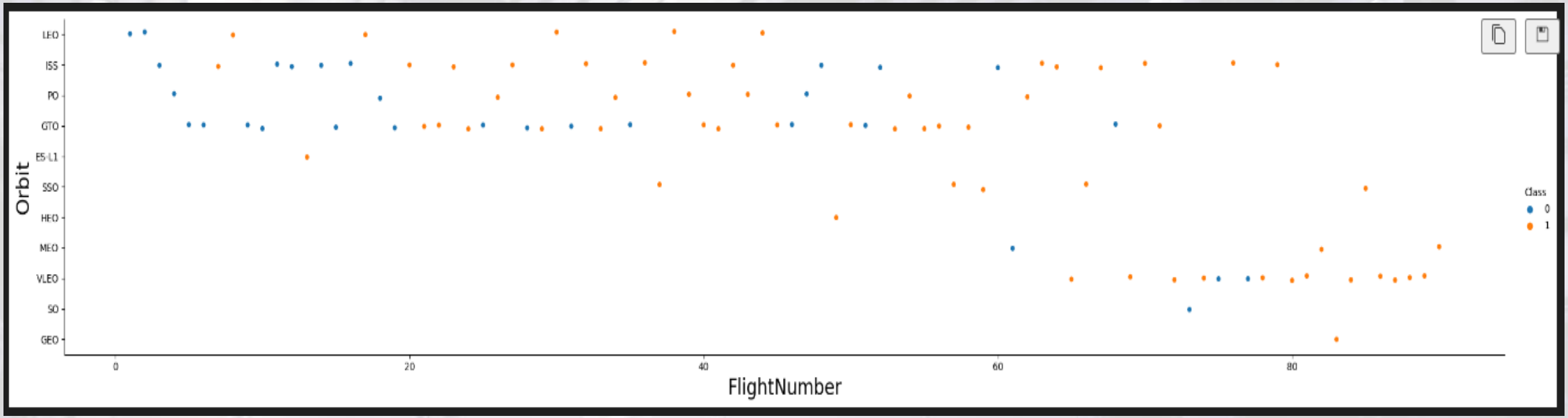
- The Payload Mass is shown on X axis with values between 0 and 16,000 kilograms.
- The Launch Sites (CCAFS-SLC-40, WAFB-SLC-4E and KSC-LC-39A) are shown on Y axis.
- The color of each point on the graph shows in orange whether it succeeded or failed in blue.

Success Rate vs. Orbit Type



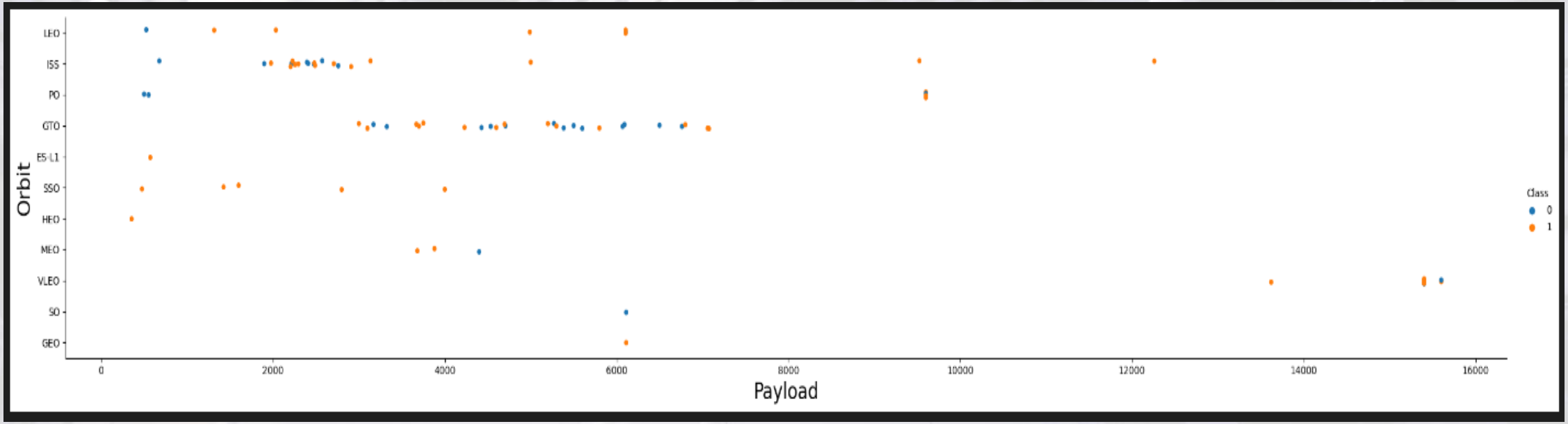
- Orbit Type is shown on X axis (ES-L1, GEO, GTO, HEO, ISS, LEO, MEO, PO, SO, SSO and VLEO).
- The success rate is shown on Y axis, 1 means 100% of success.

Flight Number vs. Orbit Type



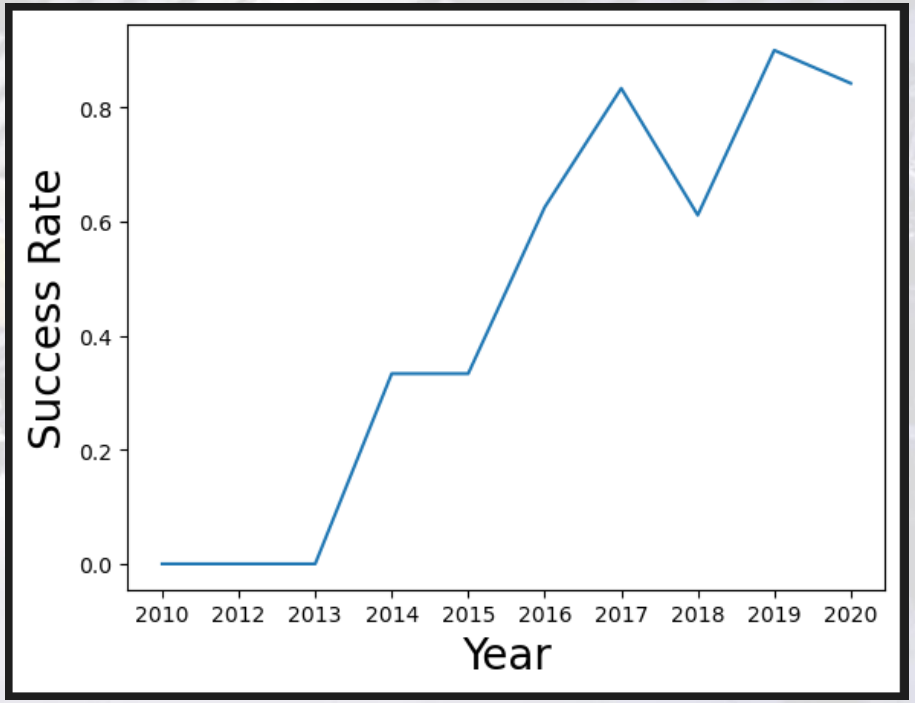
- Flight numbers of the different missions are shown on X axis.
- Orbit Type is shown on the Y axis (LEO, ISS, PO, GTO, ES-L1, SSO, HEO, MEO, VLEO, SO and GEO).
- The color of each point on the graph shows in orange whether it succeeded or failed in blue.

Payload vs. Orbit Type



- On X axis the Payload Mass is shown with values between 0 and 16,000 kilograms.
- Orbit Type is shown on the Y axis (LEO, ISS, PO, GTO, ES-L1, SSO, HEO, MEO, VLEO, SO and GEO).
- The color of each point on the graph shows in orange whether it succeeded or failed in blue.

Launch Success Yearly Trend



- The X axis shows the years of the missions from 2010 to 2020.
- The Y axis shows the average success rate for each year, with 1 being 100%.

All Launch Site Names

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

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

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

- The “DISTINCT” command was used to ensure that different values were selected from the LAUNCH_SITE field of the Database.

Launch Site Names Begin with 'CCA'

```
%sql SELECT * from SPACEXTABLE where (LAUNCH_SITE) LIKE 'CCA%' LIMIT 5;
```

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

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

- Used “LIKE” command to select sites starting with CCA% and limited to 5 records with “LIMIT 5” command.

Total Payload Mass

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Customer = 'NASA (CRS)';
```

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

SUM(PAYLOAD_MASS__KG_)
45596

- The “SUM()” command is used to calculate the total weight and “WHERE” command to indicate that we want the Customer to be NASA (CRS).

Average Payload Mass by F9 v1.1

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Booster_Version LIKE 'F9 v1.1%';
```

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

AVG(PAYLOAD_MASS__KG_)
2534.6666666666665

- The “AVG()” command was used to calculate the average payload mass carried and “LIKE” command to take the Booster_Version that starts with “F9 v1.1%”.

First Successful Ground Landing Date

```
%sql SELECT MIN(Date) FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (ground pad)';
```

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

MIN(Date)
2015-12-22

- The “MIN()” command was used to select minimum value of the date and “WHERE” command to indicate that “Landing_Outcome” should be equal to “Success (ground pad)”.

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql SELECT Booster_Version FROM SPACEXTABLE WHERE LANDING_OUTCOME = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;
```

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

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

- The “WHERE” command was used to select the boosters that have the “LANDING_OUTCOME” equal to “Success (drone ship)” and “PAYLOAD_MASS__KG_” between 4,000 and 6,000 kilograms.

Total Number of Successful and Failure Mission Outcomes

```
%sql SELECT Mission_Outcome, COUNT(Mission_Outcome) AS TOTAL_NUMBER FROM SPACEXTABLE GROUP BY Mission_Outcome;
```

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

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

- The “COUNT()” command was used to obtain the number of missions and “GROUP BY” command to have them grouped by Outcome type.

Boosters Carried Maximum Payload

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

```
 * 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

- The “DISTINCT” command was used to obtain the different Boosters without repeating and the “MAX()” command was used to obtain maximum value of Payload mass. It is important to mention that a subquery was used to be able to perform the query in a single line of instruction (nested queries).

2015 Launch Records

```
%sql SELECT SUBSTR(Date,6,2) AS Month, Landing_Outcome, Booster_Version, Launch_Site FROM SPACEXTABLE WHERE Landing_Outcome = 'Failure (drone ship)' AND SUBSTR(Date,1,4) = '2015';
```

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

Month	Landing_Outcome	Booster_Version	Launch_Site
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

- The “SUBSTR()” command was used to obtain the value of the month and year of the “Date” column, the “WHERE” command used to select the data “Failure (drone ship)” and the year “2015” .

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

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

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

Landing_Outcome	TOTAL_LANDING
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

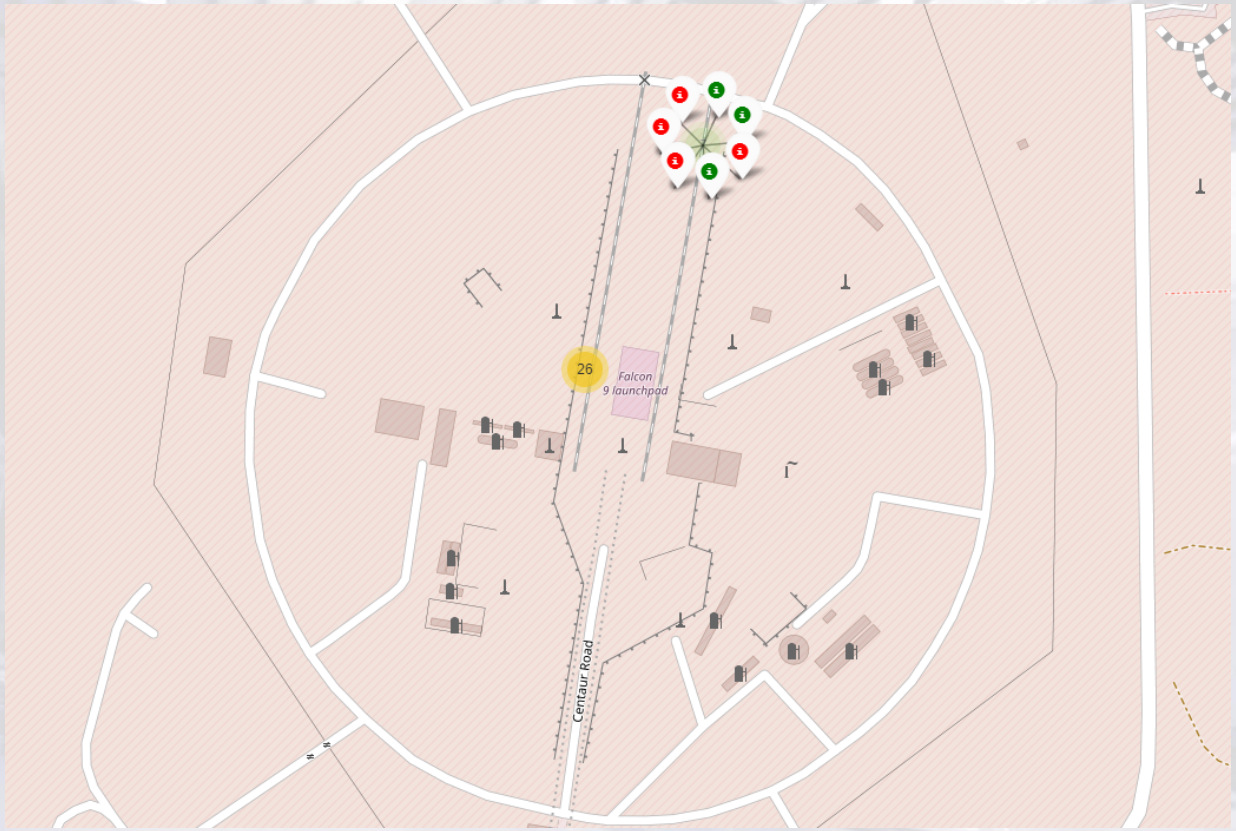
- The “COUNT()” command was used to count the Outcome amount. The “WHERE” command was used to select Date values with “BETWEEN” command for date range. The “GROUP BY” command was also used to group the Outcome into unique and ordered values using the “ORDER BY” and “DESC” commands.


Section 3
Launch Sites
Proximities Analysis

Space X Launch Sites:

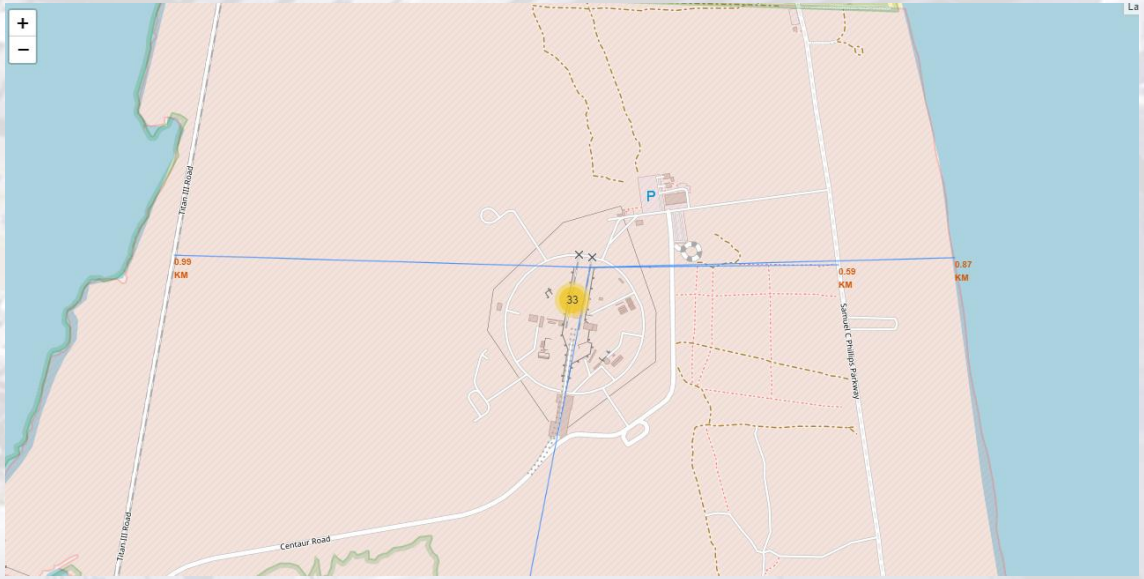
- For each launch site, a circle, a marker and a Popup were used to indicate the name of the launch site.

Map with launch results:

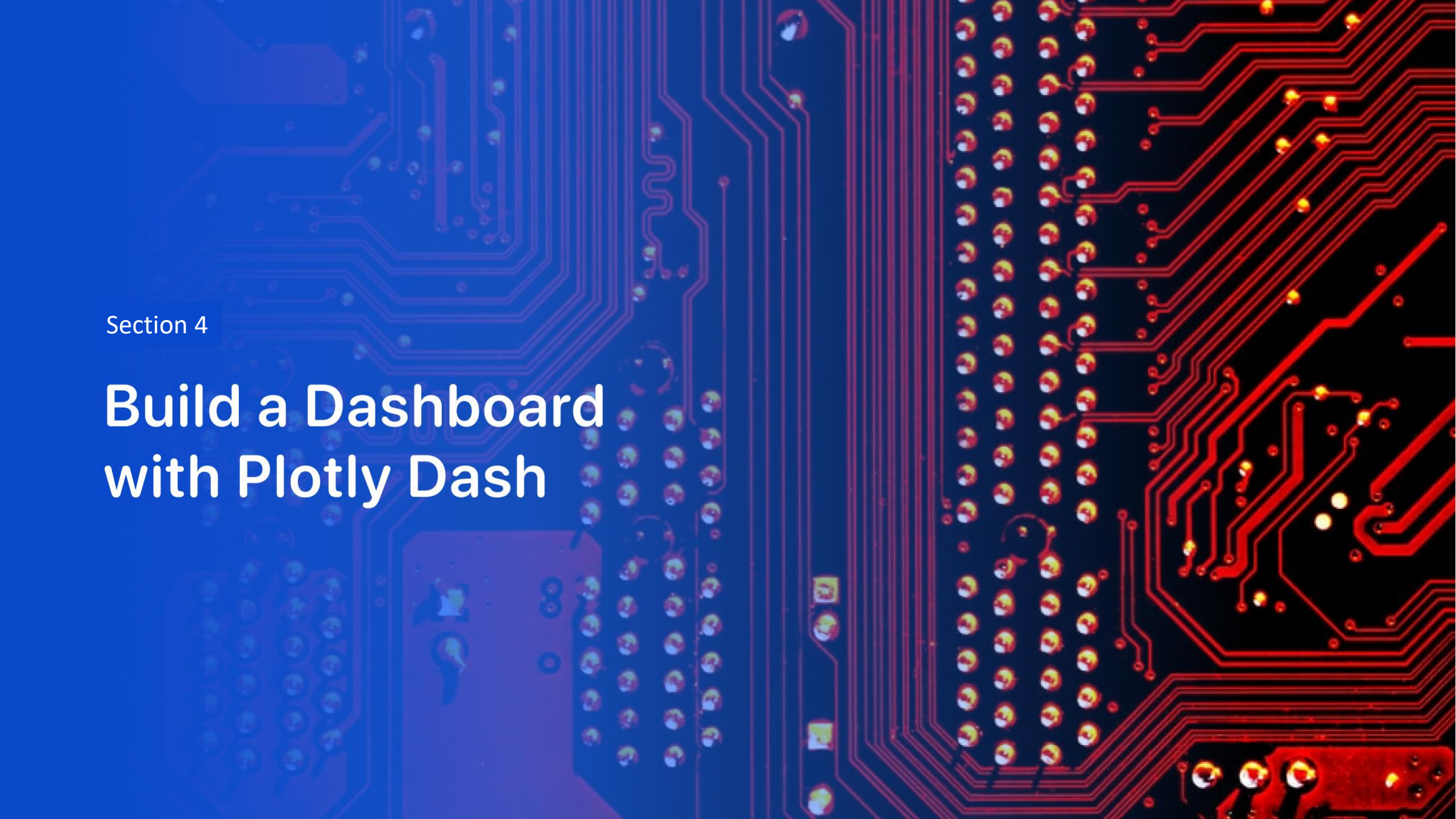


- For this map, a marker_cluster object was used to group the launches that are displayed when zoom in.
- A method was also used to give a color green if the launch was successful and red if this was failed.
- A marker object was also used to indicate the launch sites with their coordinates.

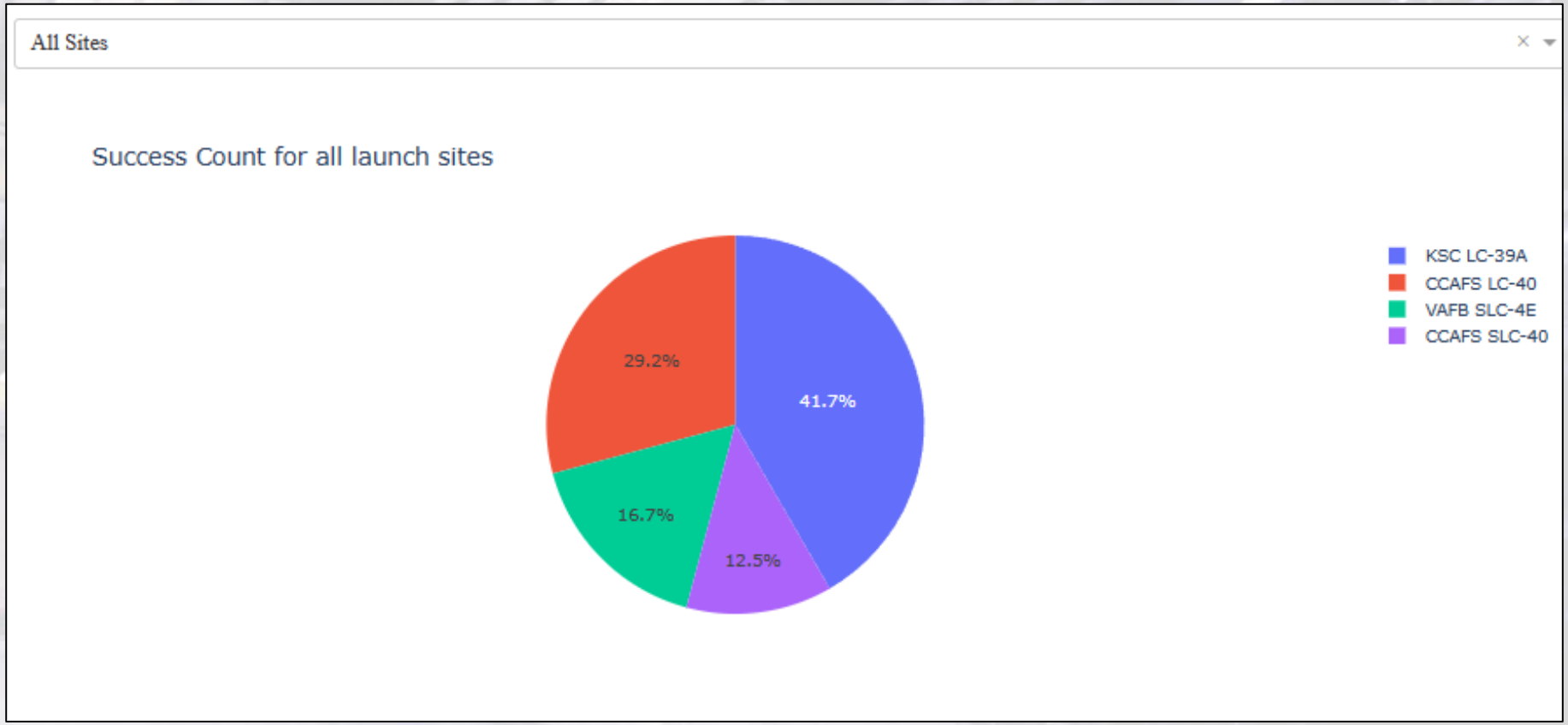
The closest railway, highway, coastline and city:



- The calculate_distance() function was defined that calculates the distance between two points given by coordinates (latitude and longitude).
- The coordinates of railway, highway, coastline and city closets were defined in the launch site.
- The marker object was used to indicate the place we want to measure with a distance label.
- The PolyLine object was also used to draw a line to the point we want to measure.

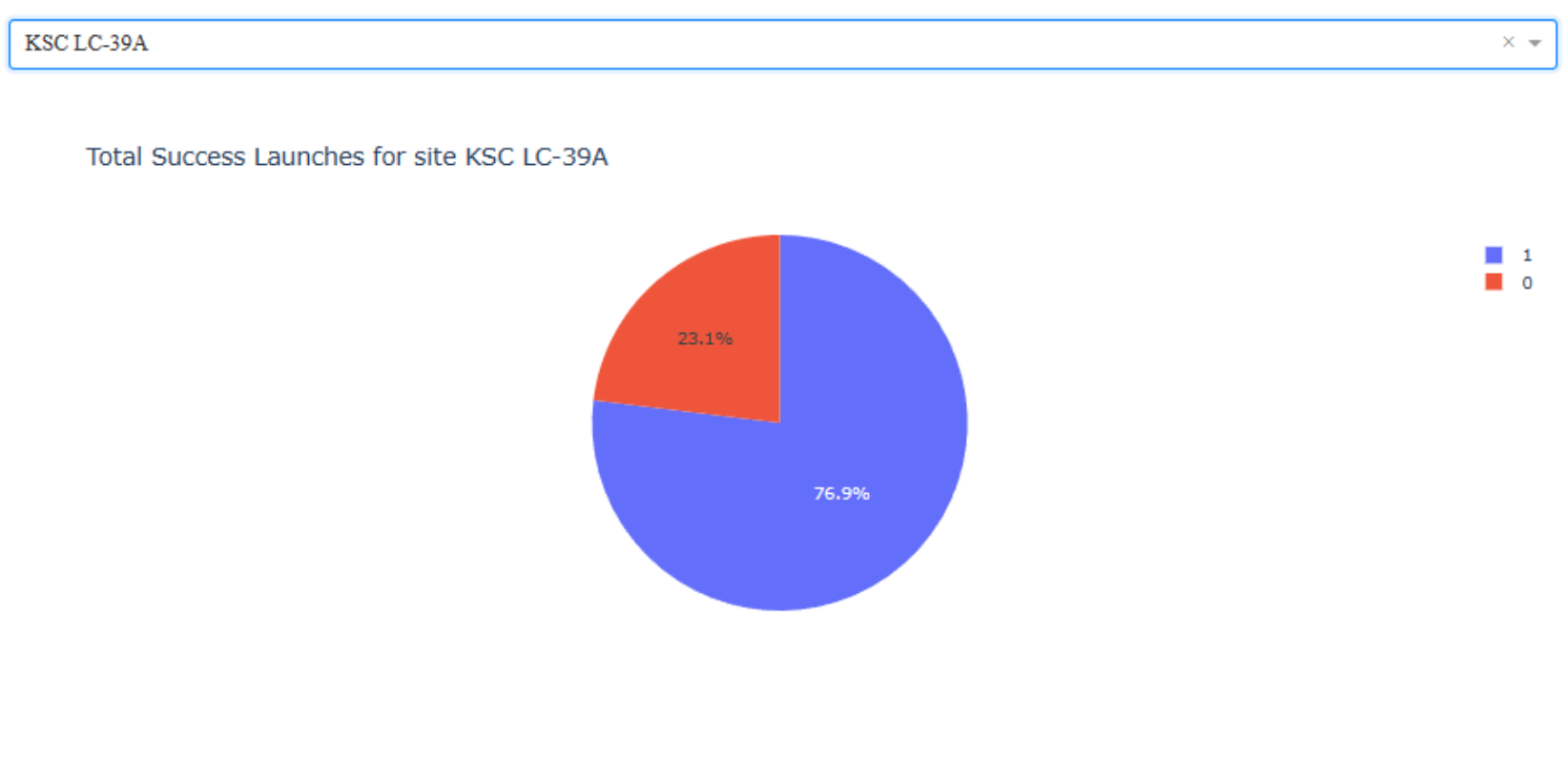

Section 4
Build a Dashboard with Plotly Dash

Distribution of Launch success quantity by site



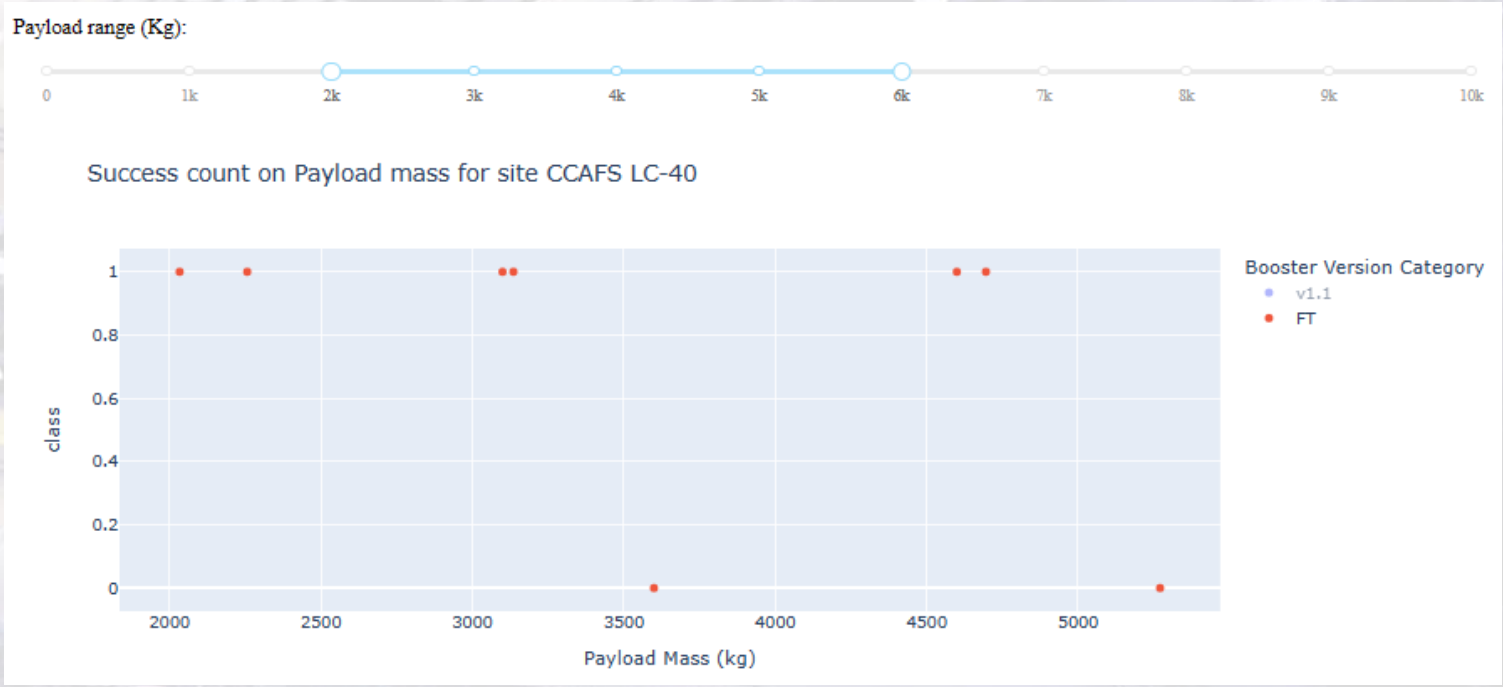
- In Pie chart can see KSC LC-39A site has the highest number of successful launches (41.7%).

KSC LC-39A site launch success ratio



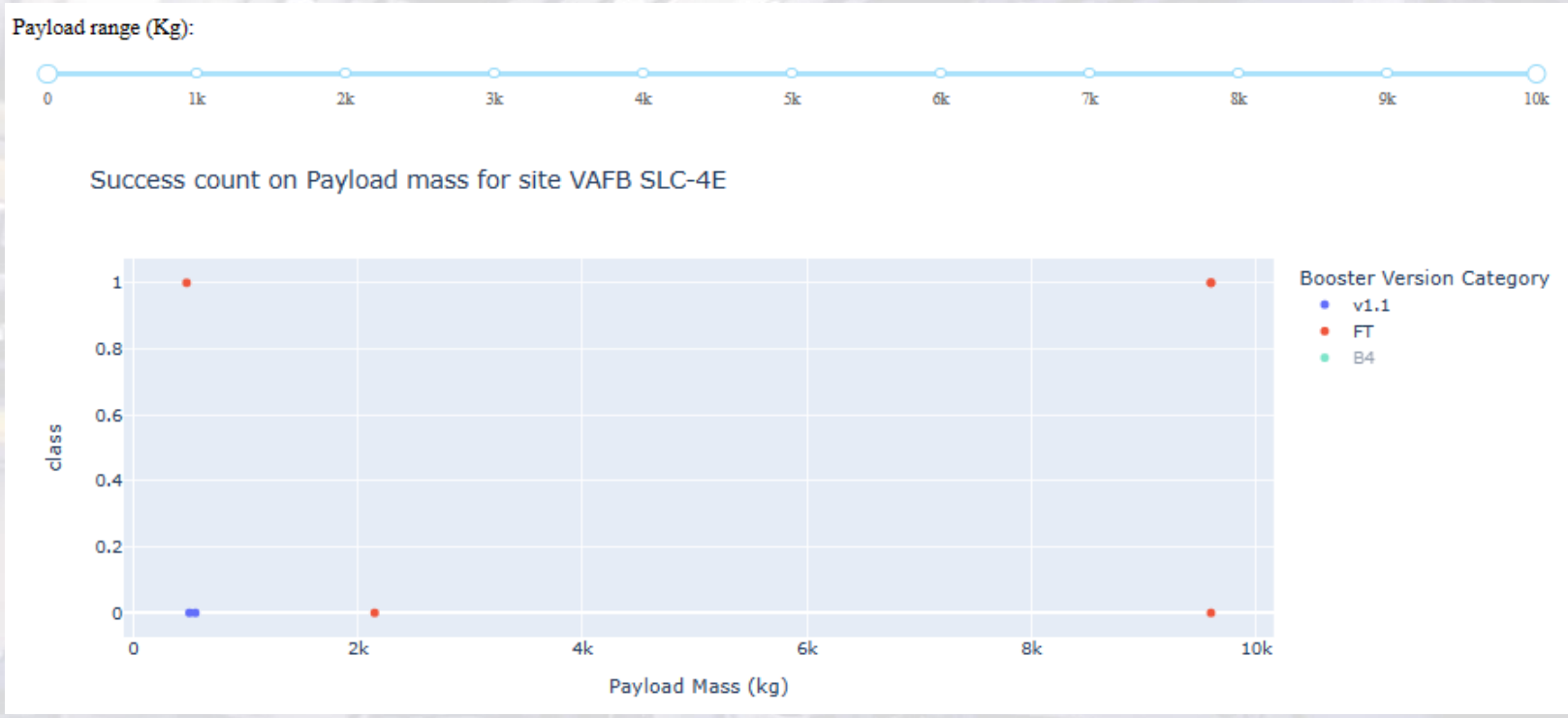
- The KSC LC-39A site has the highest success ratio with 76.9%.

Success ratio for CCAFS LC-40 with Payload between 2,000 and 6,000 kilograms



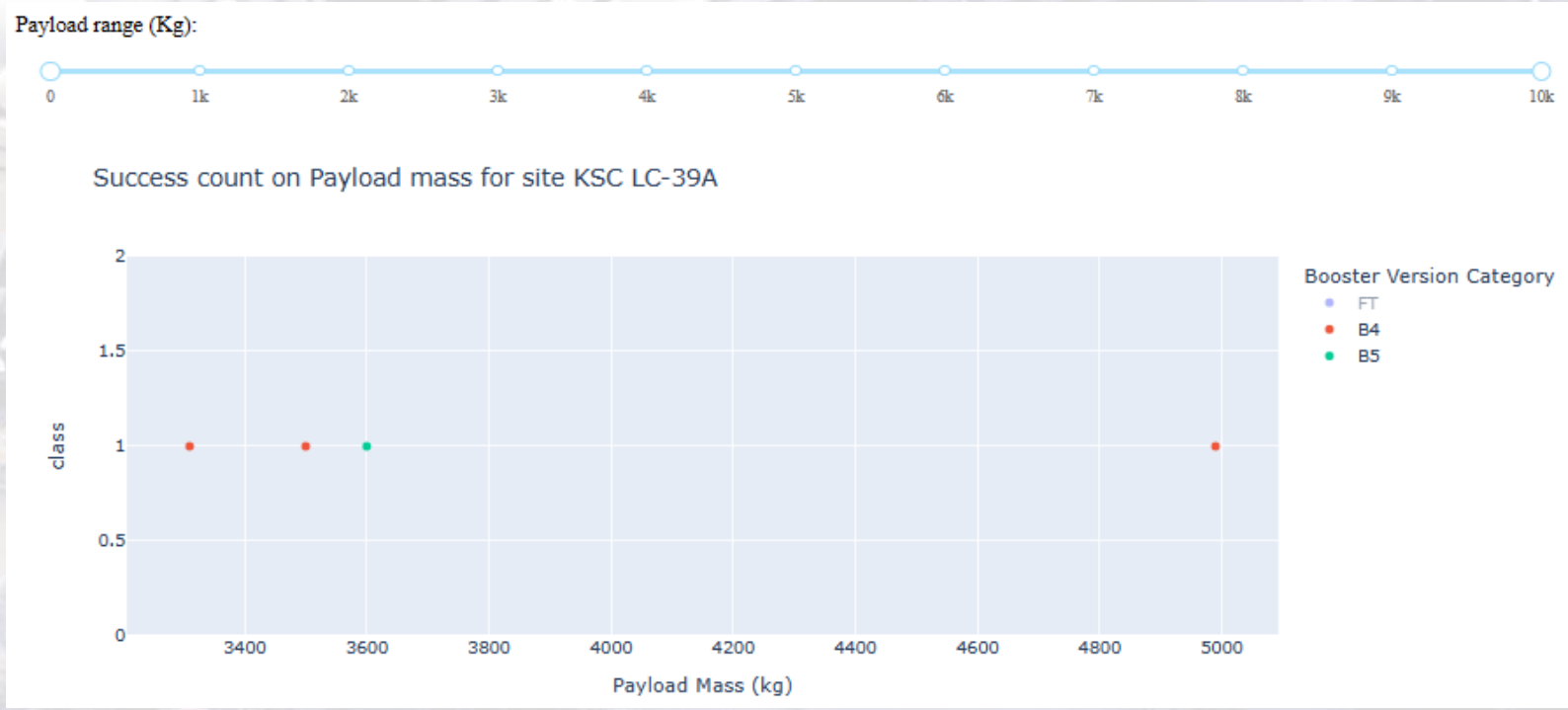
- The CCAFS LC-40 has success ratio of 26.9%. However in the Booster Version Category “FT”, with a payload between 2,000 and 6,000 kilograms, it has a success rate of 75%.

Success ratio for VAFB SLC-4E launch site



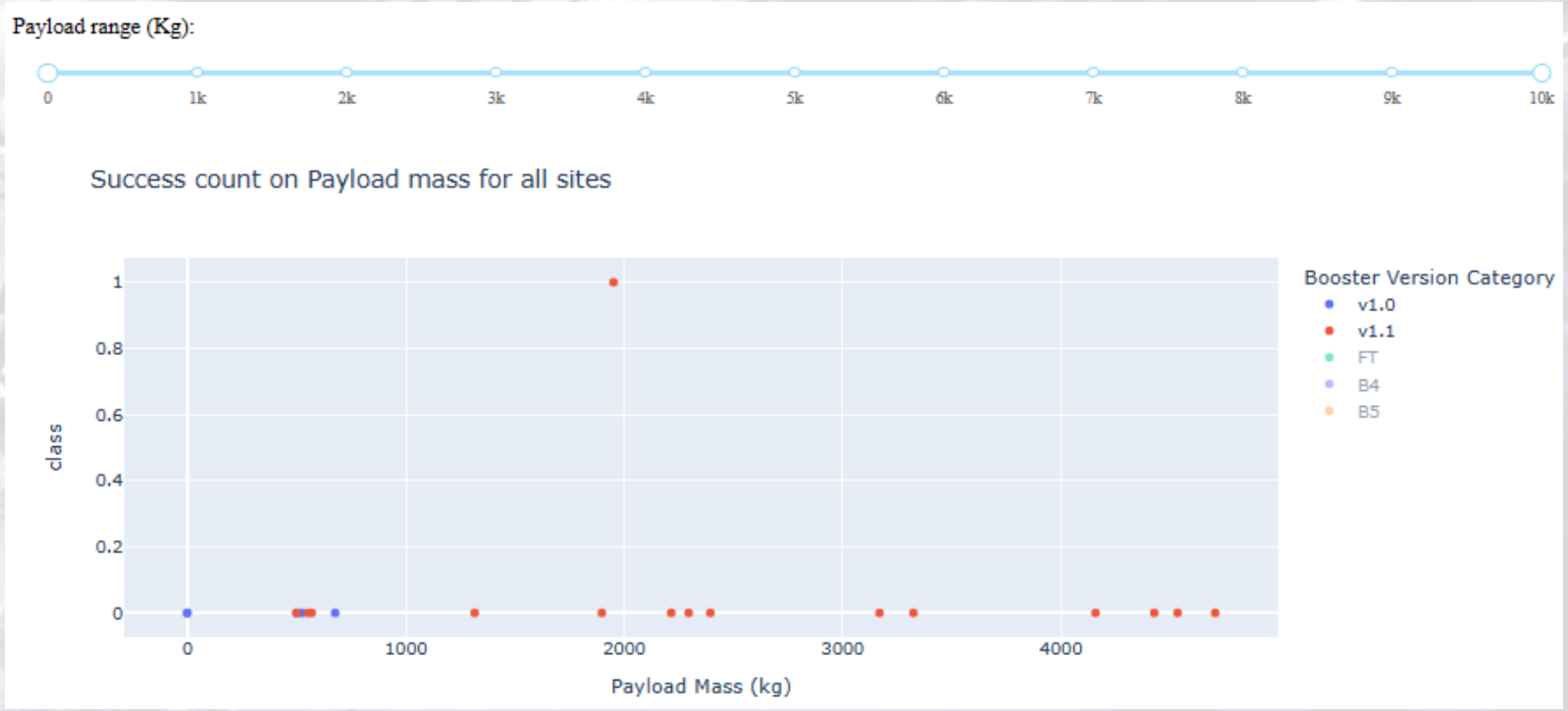
- For VAFB SLC-4E site, which has a 40% success ratio, the “FT” version Booster represents 75% of its successes.

Success ratio for versions B4 and B5 on KSC LC-39A site



- For KSC LC-39A site that has a success ratio of 76.9%, the Booster versions B4 and B5 have a success ratio of 100%.

Success ratio for Booster Version v1.0 and v1.1 in all sites



- Booster Version v1.0 and v1.1 have a very low success ratio, together it is 5% on all launch sites.

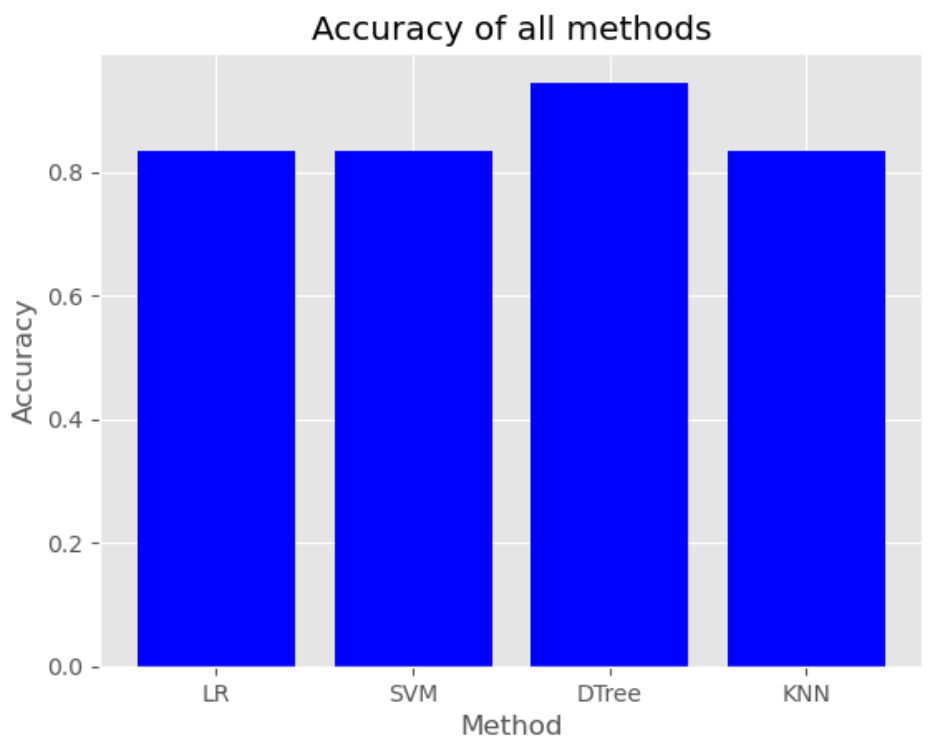
Success ratio with payload mass between 5,000 and 7,000 in all sites

- The success ratio with payload mass between 5,000 and 7,000 kilograms is low, close to 22% across all launch sites.

Section 5

Predictive Analysis (Classification)

Classification Accuracy



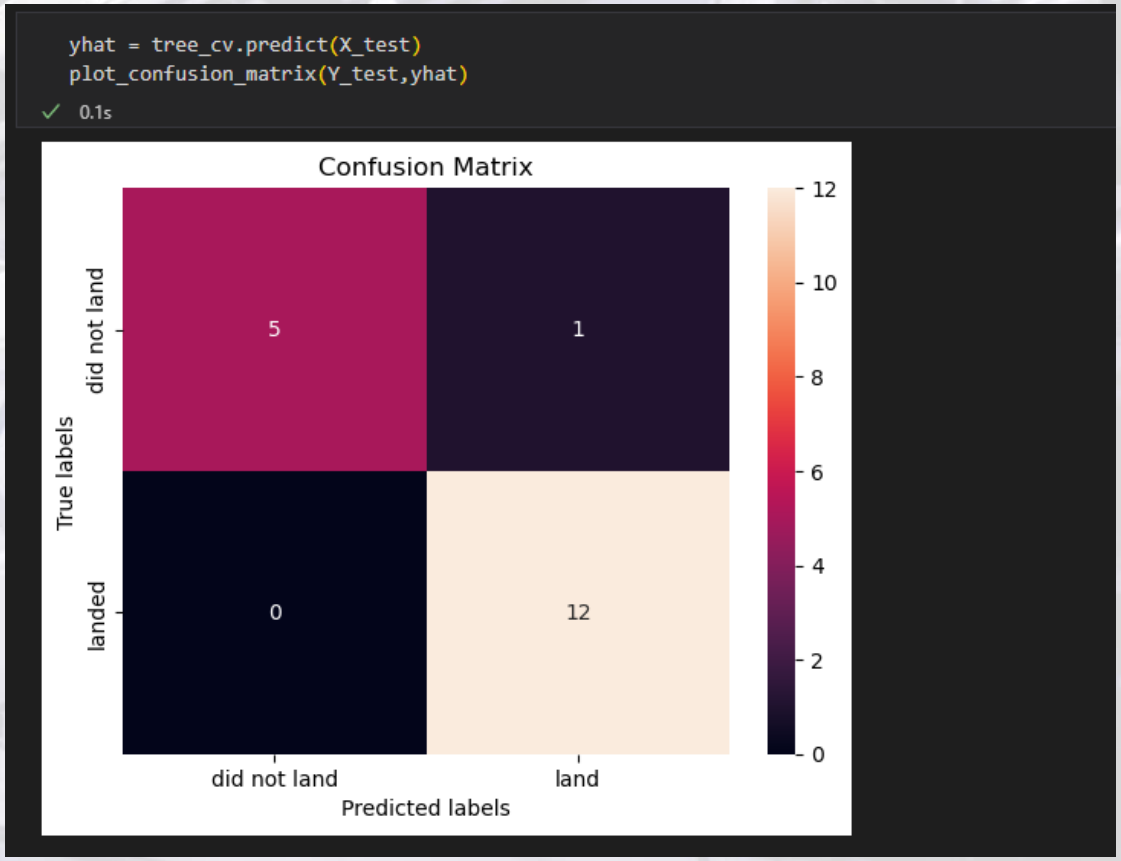

```
Accuracy for Logistics Regression method: 0.8333333333333334
Accuracy for Support Vector Machine method: 0.8333333333333334
Accuracy for Decision tree method: 0.9444444444444444
Accuracy for K nearsdt neighbors method: 0.8333333333333334
```

- The model that has the highest Accuracy is the Decision Tree method (0.9444).

Confusion Matrix

- Below is the Confusion Matrix for the best method, Decision Tree method that uses the following variables:

  - Criterion: gini
  - Max_depth: 12
  - Max_features: sqrt
  - Min_samples_leaf: 2
  - Min_samples_split: 5
  - Splitter: best



- A visualization of Tree is on Appendix - Decision Tree visualization

Conclusions

- Point 1

  Spacex has four launch sites, although CCAFS SLC-40 is the most used (33.7%), KSC LC-39A site is highest success with 77.3% success. (Appendix - Spacex.csv Table)

- Point 2

  According to database Spacex.csv, the CCAFS LC-40 launch site has been used until 2016.

- Point 3

  It took 5 years to have a successful launch (2015-12-22).

- Point 4

  Over the years the Payload Mass average increased to 11,577 kilograms by 2020

Conclusions

- Point 5

  The success of KSC LC-39A launch site is due to focused on the Booster Version Categories FT, B4 and B5 with an average success of 88.9%. (Appendix - spacex_launch_dash Table)

- Point 6

  The model selected to make predictions regarding the success of a mission will be the Decision Tree classifier with the best parameters found.

Appendix

Orbit Types:

- Low Earth orbit (LEO) is an Earth-centred orbit with an altitude of 2,000 km (1,200 mi) or less. Most of the manmade objects in outer space are in LEO.
- Very Low Earth Orbits (VLEO) can be defined as the orbits with a mean altitude below 450 km.
- A geosynchronous orbit (GTO) is a high Earth orbit that allows satellites to match Earth's rotation. Located at 22,236 miles (35,786 kilometers) above Earth's equator. At this hight satellite orbits at the same speed that the Earth.
- It is a Sun-synchronous orbit (SSO or SO) is a nearly polar orbit around a planet, in which the satellite passes over any given point of the planet's surface at the same local mean solar time.
- At the Lagrange points the gravitational forces of the two large bodies cancel out in such a way that a small object placed in orbit there is in equilibrium relative to the center of mass of the large bodies. ES-L1 is one such point between the sun and the earth.
- A highly elliptical orbit (HEO), is an elliptic orbit with high eccentricity.

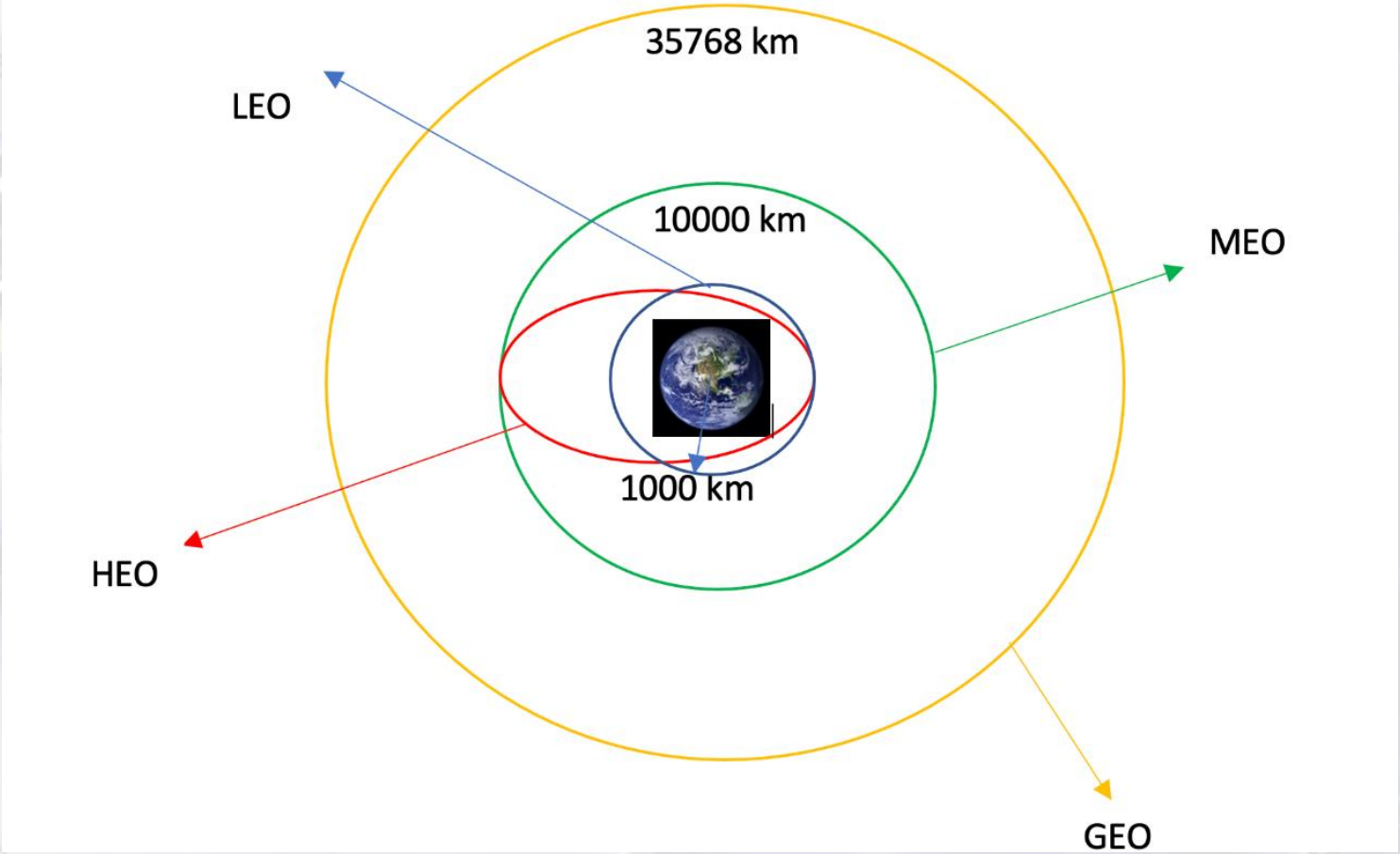
Appendix

Orbit Types:

- ISS is a modular space station (habitable artificial satellite) in low Earth orbit. It is a multinational collaborative project between five participating space agencies.
- MEO are geocentric orbits ranging in altitude from 2,000 km (1,200 mi) to just below geosynchronous orbit at 35,786 kilometers (22,236 mi). Also known as an intermediate circular orbit.
- HEO are geocentric orbits above the altitude of geosynchronous orbit (35,786 km or 22,236 mi).
- GEO is a circular geosynchronous orbit 35,786 kilometres (22,236 miles) above Earth's equator and following the direction of Earth's rotation.
- PO is one type of satellites in which a satellite passes above or nearly above both poles of the body being orbited.

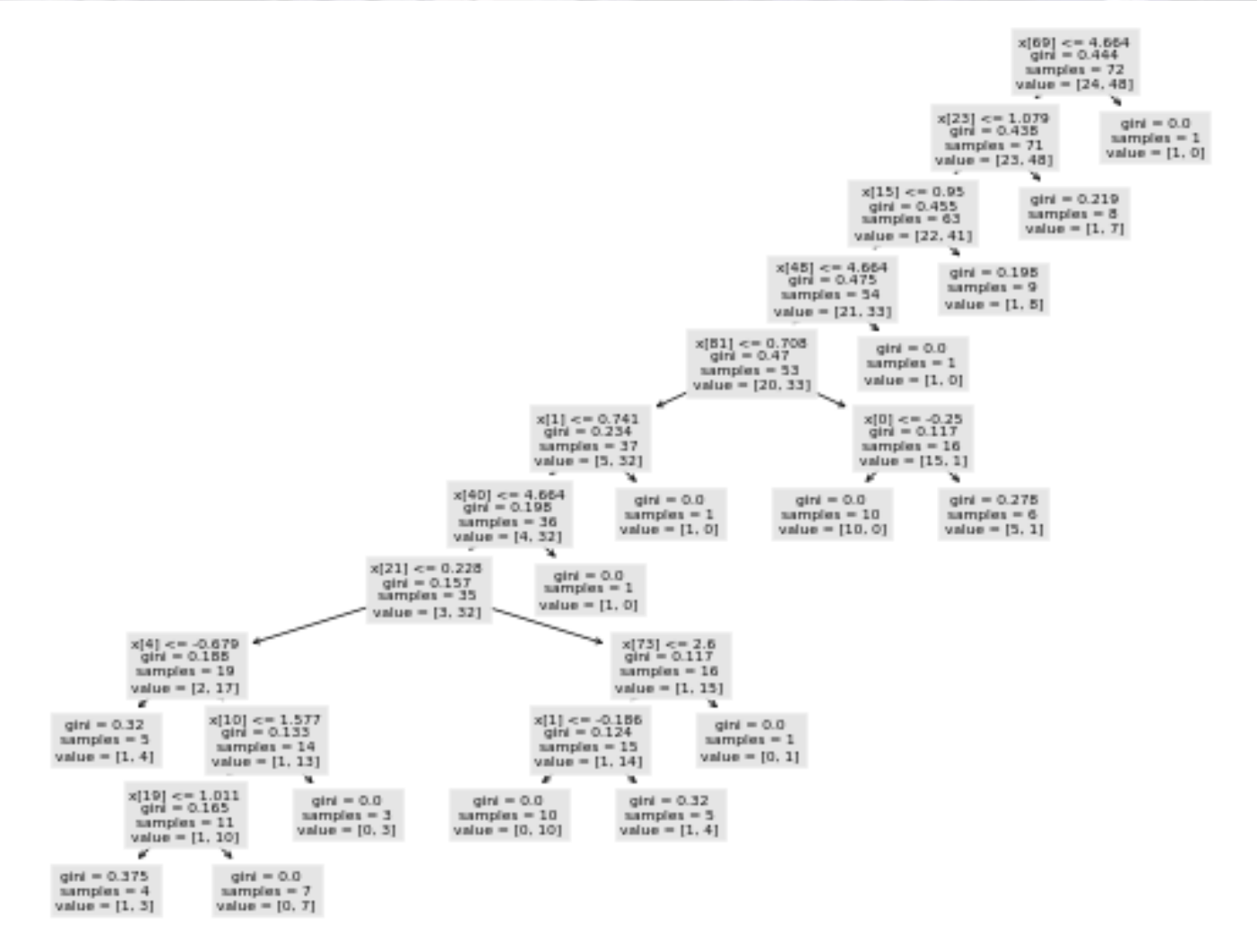
Appendix

Some Orbit types:



Appendix

Decision Tree visualization: Predictive best model:



Appendix

Tables with Spacex.cvs file:

Yearly total launches by Booster Version

Year	F9 B4	F9 B5	F9 FT	F9 v1.0	F9 v1.1	Total
2010				2		2
2012				2		2
2013				1	2	3
2014					6	6
2015			1		6	7
2016			7		1	8
2017	4		14			18
2018	8	10	2			20
2019		11				11
2020		24				24
Total	12	45	24	5	15	101

Appendix

Tables with Spacex.cvs file:

Yearly total launches by site

Year	CCAFS LC-40	CCAFS SLC-40	KSC LC-39A	VAFB SLC-4E	Total
2010	2				2
2012	2				2
2013	2			1	3
2014	6				6
2015	7				7
2016	7			1	8
2017		1	12	5	18
2018		12	2	6	20
2019		8	1	2	11
2020		13	10	1	24
Total	26	34	25	16	101
Percentage	25.7%	33.7%	24.8%	15.8%	

Appendix

Tables with Spacex.cvs file:

Yearly launches outcomes

Year	Failure	Success	Total
2010	1		1
2012	1		1
2013	3		3
2014	4	2	6
2015	4	2	6
2016	2	5	7
2017	3	15	18
2018	7	11	18
2019	1	9	10
2020	3	16	19
Total	29	60	89

Appendix

Tables with Spacex.cvs file:

Launches outcomes by Site

Launch site	Failure	Success	Total	Percentage
CCAFS LC-40	13	9	22	24.7%
CCAFS SLC-40	8	24	32	36.0%
KSC LC-39A	5	17	22	24.7%
VAFB SLC-4E	3	10	13	14.6%
Total	**29**	**60**	**89**	

Appendix

Tables with Spacex.cvs file:

Yearly payload mass average by Site

Year	F9 B4	F9 B5	F9 FT	F9 v1.0	F9 v1.1	Average
2010				0.0		0.0
2012				512.5		512.5
2013				677.0	1835.0	1449.0
2014					3019.3	3019.3
2015			2034.0		2613.5	2530.7
2016			3808.6		553.0	3401.6
2017	5350.0		5327.0			5332.1
2018	4780.3	5233.5	3190.0			4847.9
2019		7341.9				7341.9
2020		11577.3				11577.3
Average	4970.2	9132.2	4568.8	340.4	2534.7	6138.3

Appendix

Tables with Spacex.cvs file:

Yearly payload mass average by Site

Year	F9 B4	F9 B5	F9 FT	F9 v1.0	F9 v1.1	Average
2010				0.0		0.0
2012				512.5		512.5
2013				677.0	1835.0	1449.0
2014					3019.3	3019.3
2015			2034.0		2613.5	2530.7
2016			3808.6		553.0	3401.6
2017	5350.0		5327.0			5332.1
2018	4780.3	5233.5	3190.0			4847.9
2019		7341.9				7341.9
2020		11577.3				11577.3
Average	4970.2	9132.2	4568.8	340.4	2534.7	6138.3

Appendix

Table with spacex_launch_dash.cvs file:

Success launches from KSC LC-39A site

Launch Site	KSC LC-39A			
Booster version	**Failure**	**Success**	**Total**	**Percentage**
B4		3	3	100.0%
B5		1	1	100.0%
FT	3	6	9	66.7%
Total	**3**	**10**	**13**	**88.9%**

Thank you!