IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Diego Asturiano Calva

16-05-2025

Outline

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

Executive Summary

- Summary of methodologies
  - Collection API
  - Web scraping
  - Data wrangling
  - EDA Using SQL
  - EDA Using Pandas and Matplotlib
  - Interactive Visual Analytics with Folium lab
- Summary of all results
  - Interactive Dashboard with Ploty Dash
  - Machine Learning Prediction

Introduction

- Project background and context

SpaceX advertises Falcon 9 rocket launches on its website, with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.

- Problems you want to find answers

We will predict if the Falcon 9 first stage will land successfully. SpaceX advertises Falcon 9 rocket launches on its website

Section 1

Methodology

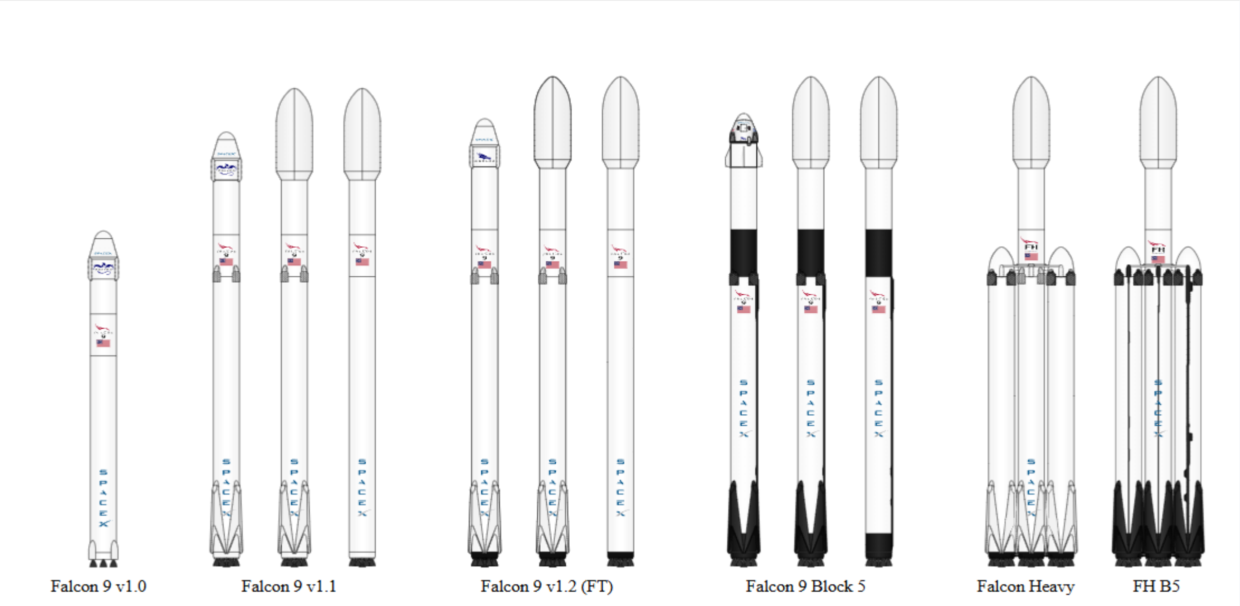
Methodology

Executive Summary

- Data collection methodology:
  - Data Collection API
  - Data Collection with Web Scraping
- Perform data wrangling
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models

Data Collection

- Falcon 9 space X data was collected by different methods and from different sources.
- Get request to the SpaceX API. Request and parse the SpaceX launch data using the GET request. JSON.
- Web scraping to compile historical Falcon 9 launch records from a Wikipedia page titled **List of Falcon 9 and Falcon Heavy launches**. Extract all column/variable names from the HTML table header. BeautifulSoup.



Data Collection – SpaceX API

- Let's start requesting rocket launch data from SpaceX API with the following URL:

  https://api.spacexdata.com/v4/launches/past

- To make the requested JSON results more consistent, I useed the following static response object for this project.

- GitHub SpaceX API calls notebook:https:

https://github.com/Diego-Asturiano/Data_Science_Capstone_Falcon9_SpaceX_Launch/blob/97debc8c469fb2301999bc7af02e6168c285020a/1.1%20Collection%20API.ipynb

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

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

Check the content of the response

```
print(response.content)
```

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

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

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

```
response.status_code
```

```
200
```

Data Collection - Scraping

- Request the Falcon9 Launch Wiki page from its URL:

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

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

- Extract all column names from the HTML table header. Then, create a data frame by parsing the launch HTML tables

- GitHub URL of the completed web scraping notebook:

https://github.com/Diego-Asturiano/Data_Science_Capstone_Falcon9_SpaceX_Launch/blob/97debc8c469fb2301999bc7af02e6168c285020a/1.2%20Webscraping.ipynb

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

```
# use requests.get() method with the provided static_url
# assign the response to a object
req = requests.get(static_url)
```

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

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

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

```
# Use soup.title attribute
soup.title
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

Starting from the third table is our target table contains the actual launch records.

```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

Data Wrangling

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

FlightNumber       0.000000
Date               0.000000
BoosterVersion     0.000000
PayloadMass        0.000000
Orbit              0.000000
LaunchSite         0.000000
Outcome            0.000000
Flights            0.000000
GridFins           0.000000
Reused             0.000000
Legs               0.000000
LandingPad        28.888889
Block              0.000000
ReusedCount        0.000000
Serial             0.000000
Longitude          0.000000
Latitude           0.000000
dtype: float64
```

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

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

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

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

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

Outcome
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: count, dtype: int64
```

```
Class
1    60
0    30
```

- The Exploratory Data Analysis (EDA) was performed to find some patterns in the data and determine what would be the label for supervised model training.
- The Space X data set from the previous section is loaded. The percentage of missing values for each attribute was identified and calculated.
- The following calculations were made:
  - Calculate the number of launches on each site
  - Calculate the number and occurrence of each orbit
  - Calculate the number and occurrence of mission outcome of the orbits
- Finally, a landing result tag was created from the Result column.
- GitHub URL of data wrangling related notebooks:

  https://github.com/Diego-Asturiano/Data_Science_Capstone_Falcon9_SpaceX_Launch/blob/97debc8c469fb2301999bc7af02e6168c285020a/01.3%20Data%20wrangling.ipynb

EDA with Data Visualization

- In this section we created scatter and bar charts with Python to analyze data in a Pandas data frame. We performed exploratory data analysis by manipulating data in a Pandas data frame and executed SQL queries to select and sort data.
    - Visualize the relationship between Flight Number and Launch Site. Scatterplot
    - Visualize the relationship between Payload Mass and Launch Site - Scatterplot
    - Visualize the relationship between success rate of each orbit type - bar plot
    - Visualize the relationship between Flight Number and Orbit type – Scatterplot
    - Visualize the relationship between Payload Mass and Orbit type - Scatterplot
    - Visualize the launch success yearly trend – Line Plot
- GitHub URL of EDA with data visualization notebook:

https://github.com/Diego-Asturiano/Data_Science_Capstone_Falcon9_SpaceX_Launch/blob/666c475415295fdc01b0778dfb7fff901c8e4c0f/2.2%20EDA%20Using%20Pandas%20and%20Matplotlib.ipynb

EDA with SQL

- Load the dataset into the corresponding table in a Db2 database.

```
%sql sqlite:///my_data1.db
```

```
import pandas as pd
df = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/labs/module_2/data/Spacex.csv")
df.to_sql("SPACEXTBL", con, if_exists='replace', index=False,method="multi")
```

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

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)

2. Display 5 records where launch sites begin with the string 'CCA'

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

EDA with SQL

3. Display the total payload mass carried by boosters launched by NASA (CRS)
4. Display average payload mass carried by booster version F9 v1.1
5. List the date when the first successful landing outcome in ground pad was achieved.
6. List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
7. List the total number of successful and failure mission outcomes.
8. List all the booster versions that have carried the maximum payload mass. Use a subquery.
9. 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.

3.

SUM(PAYLOAD_MASS__KG_)
45596

4.

AVG(PAYLOAD_MASS__KG_)
2534.666666666665

5.

MIN(Date)
2015-12-22

6.

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

7.

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

8.

Booster_Version	Payload	PAYLOAD_MASS__KG_
F9 B5 B1048.4	Starlink 1 v1.0, SpaceX CRS-19	15600
F9 B5 B1049.4	Starlink 2 v1.0, Crew Dragon in-flight abort test	15600
F9 B5 B1051.3	Starlink 3 v1.0, Starlink 4 v1.0	15600
F9 B5 B1056.4	Starlink 4 v1.0, SpaceX CRS-20	15600

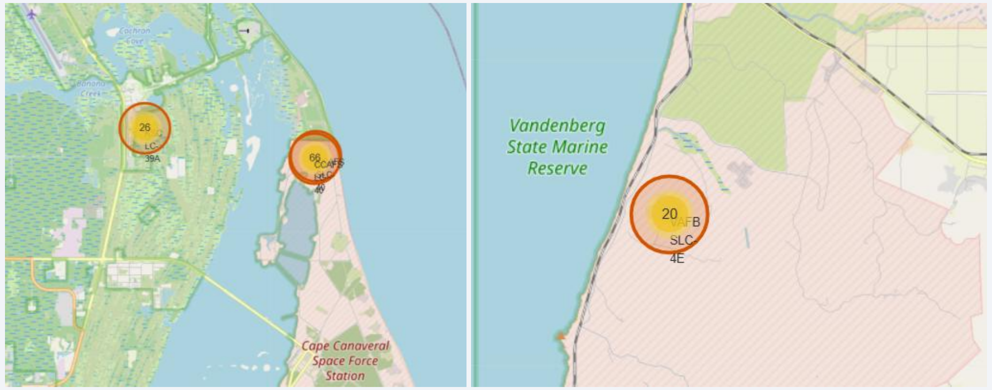
9.

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2016-06-15	14:29:00	F9 FT B1024	CCAFS LC-40	ABS-2A Eutelsat 117 West B	3600	GTO	ABS Eutelsat	Success	Failure (drone ship)
2016-03-04	23:35:00	F9 FT B1020	CCAFS LC-40	SES-9	5271	GTO	SES	Success	Failure (drone ship)
2016-01-17	18:42:00	F9 v1.1 B1017	VAFB SLC-4E	Jason-3	553	LEO	NASA (LSP) NOAA CNES	Success	Failure (drone ship)
2015-04-14	20:10:00	F9 v1.1 B1015	CCAFS LC-40	SpaceX CRS-6	1898	LEO (ISS)	NASA (CRS)	Success	Failure (drone ship)
2015-01-10	9:47:00	F9 v1.1 B1012	CCAFS LC-40	SpaceX CRS-5	2395	LEO (ISS)	NASA (CRS)	Success	Failure (drone ship)

- GitHub URL of EDA with SQL notebook: URL

Build an Interactive Map with Folium

- Finding an optimal location for building a launch site certainly involves many factors, we could discover some of the factors by analyzing the existing launch site locations.
- In this section the following folium maps were created:
  - Mark all launch sites on a map.
  - Mark the success/failed launches for each site on the map.



- GitHub URL of interactive map with Folium map:

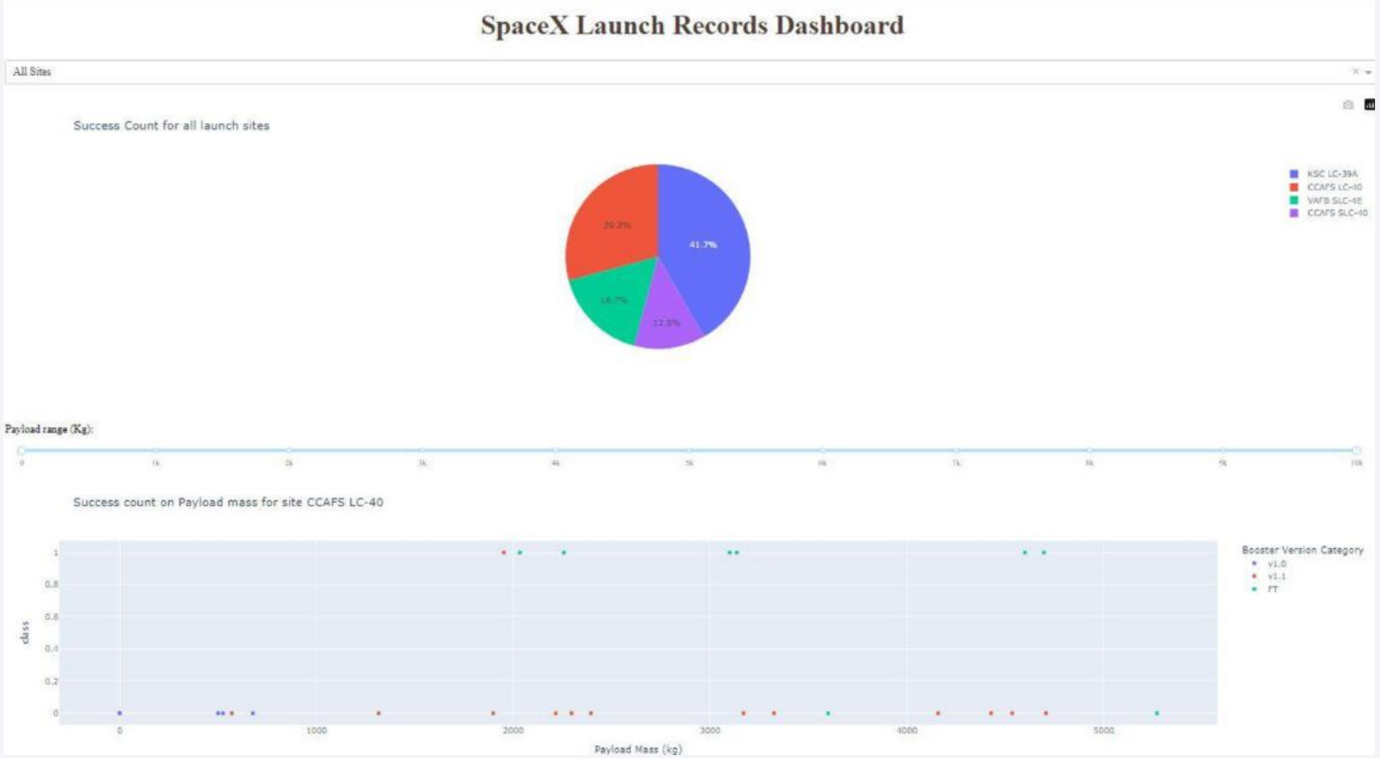
https://github.com/Diego-Asturiano/Data_Science_Capstone_Falcon9_SpaceX_Launch/blob/666c475415295fdc01b0778dfb7fff901c8e4c0f/3.1%20Interactive%20Visual%20Analytics%20with%20Folium%20lab.ipynb

Build a Dashboard with Plotly Dash

- An interactive dashboard containing pie charts and scatter plots was built to analyze launch records interactively with Plotly Dash.
- GitHub URL of Plotly Dash lab:

https://github.com/Diego-Asturiano/Data_Science_Capstone_Falcon9_SpaceX_Launch/blob/666c475415295fdc01b0778dfb7fff901c8e4c0f/3.2%20Build%20an%20Interactive%20Dashboard%20with%20Ploty%20Dash.py

- Ploty Dash App



Predictive Analysis (Classification)

- A predictive analysis of the data was performed, and training labels were determined.
  - Created a column for the class.
  - Standardize the data.
  - Split the data into training data and test data.
  - Find the best hyperparameter for SVM, classification trees and logistic regression.

1. Create a NumPy array from the column Class in data, by applying the method to_numpy().

```
y = data['Class'].to_numpy()
y.dtype
```

```
dtype('int64')
```

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

```
# students get this
transform = preprocessing.StandardScaler()
X = transform.fit_transform(X)
X
```

```
array([[-1.71291154e+00, -1.94814463e-16, -6.53912840e-01, ...,
        -8.35531692e-01,  1.93309133e+00, -1.93309133e+00],
       [-1.67441914e+00, -1.19523159e+00, -6.53912840e-01, ...,
        -8.35531692e-01,  1.93309133e+00, -1.93309133e+00],
       [-1.63592675e+00, -1.16267307e+00, -6.53912840e-01, ...,
        -8.35531692e-01,  1.93309133e+00, -1.93309133e+00],
       ...,
       [ 1.63592675e+00,  1.99100483e+00,  3.49060516e+00, ...,
         1.19684269e+00, -5.17306132e-01,  5.17306132e-01],
       [ 1.67441914e+00,  1.99100483e+00,  1.00389436e+00, ...,
         1.19684269e+00, -5.17306132e-01,  5.17306132e-01],
       [ 1.71291154e+00, -5.19213966e-01, -6.53912840e-01, ...,
        -8.35531692e-01, -5.17306132e-01,  5.17306132e-01]])
```

Predictive Analysis (Classification)

3. Use the function train_test_split to split the data X and Y into training and test data. Set the parameter test_size to 0.2 and random state to 2.

```
X_train, X_test, Y_train, Y_test
```

```
X_train, X_test, y_train, y_test = train_test_split(X , y, test_size=0.2,random_state=2)
```

we can see we only have 18 test samples.

```
y_test.shape
```

```
(18,)
```

4. Create a logistic regression object then create a GridSearchCV object logreg_cv with cv = 10.

```
parameters ={'C':[0.01,0.1,1],
             'penalty':['l2'],
             'solver':['lbfgs']}
```

```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
lr=LogisticRegression()

# Create a GridSearchCV object logreg_cv
logreg_cv = GridSearchCV(lr, parameters, cv=10)

#Fit the training data into the GridSearch object
logreg_cv.fit(X_train, y_train)
```

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

```
tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8464285714285713
```

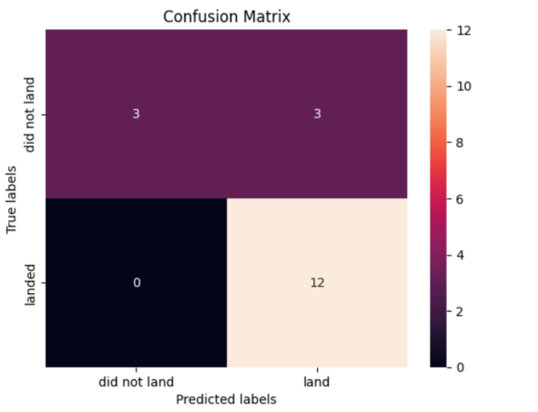
5. Calculate the accuracy on the test data using the method score.

```
print("Logistic Regression test data accuracy :",logreg_cv.score(X_test, y_test))
```

```
Logistic Regression test data accuracy : 0.8333333333333334
```

Lets look at the confusion matrix:

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



6. Create a support vector machine object then create a GridSearchCV object svm_cv with cv = 10.

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

```
tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
accuracy : 0.8482142857142856
```

Create a decision tree classifier object then create a GridSearchCV object tree_cv with cv = 10.

Predictive Analysis (Classification)

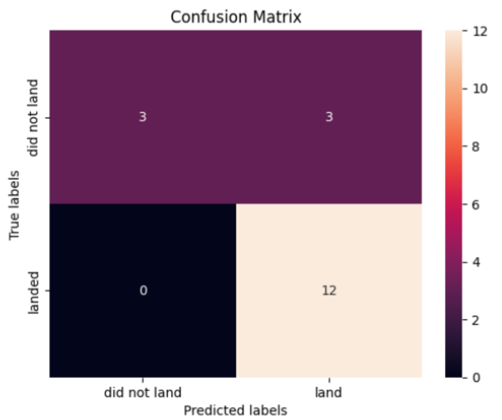
7. Calculate the accuracy on the test data using the method score:

```
print("SVM test data accuracy :",svm_cv.score(X_test, y_test))
```

```
SVM test data accuracy : 0.8333333333333334
```

We can plot the confusion matrix

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



8. Create a logistic regression object then create a GridSearchCV object logreg_cv with cv = 10.

```
tree_cv = GridSearchCV(tree, parameters, cv= 10)

#Fit the training data into the GridSearch object
tree_cv.fit(X_train, y_train)
```

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

```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 10, 'max_features': 'sqrt'
'splitter': 'random'}
accuracy : 0.8875
```

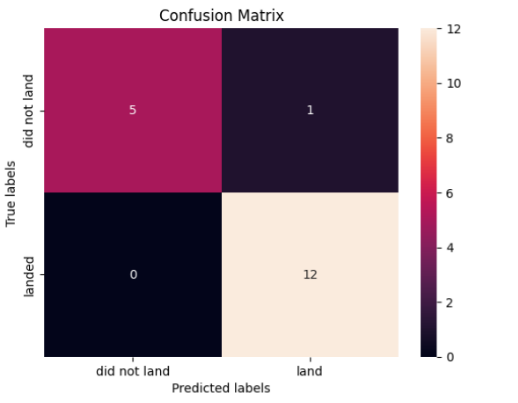
9. Calculate the accuracy of tree_cv on the test data using the method score:

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

```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 10,
0, 'splitter': 'random'}
accuracy : 0.8875
```

We can plot the confusion matrix

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



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

```
knn_cv = GridSearchCV(KNN, parameters, cv= 10)

#Fit the training data into the GridSearch object
knn_cv.fit(X_train, y_train)
```

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

```
tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
accuracy : 0.8482142857142858
```

Create a decision tree classifier object then create a GridSearchCV object tree_cv with cv = 10.

Predictive Analysis (Classification)

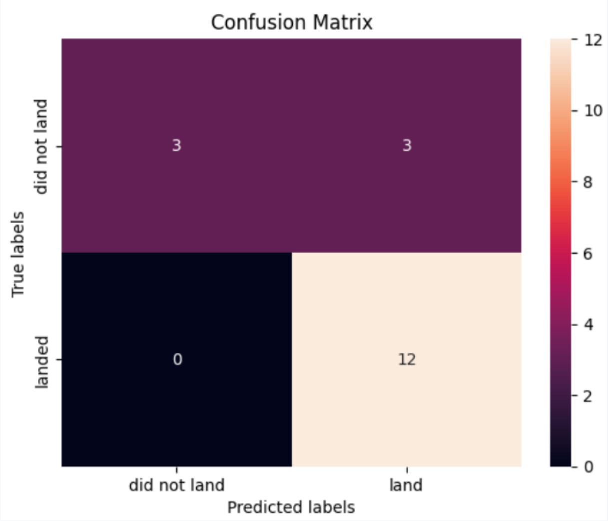
11. Calculate the accuracy of knn_cv on the test data using the method score:

```
knn_cv.score(X_test, y_test)
```

```
0.8333333333333334
```

We can plot the confusion matrix

```
yhat = knn_cv.predict(X_test)
plot_confusion_matrix(y_test,yhat)
```



12. Find the method performs best:

```
Report = pd.DataFrame({'Method' : ['Test Data Accuracy']})

knn_accuracy=knn_cv.score(X_test, y_test)
Decision_tree_accuracy=tree_cv.score(X_test, y_test)
SVM_accuracy=svm_cv.score(X_test, y_test)
Logistic_Regression=logreg_cv.score(X_test, y_test)

Report['Logistic_Reg'] = [Logistic_Regression]
Report['SVM'] = [SVM_accuracy]
Report['Decision Tree'] = [Decision_tree_accuracy]
Report['KNN'] = [knn_accuracy]

Report.transpose()
```

GitHub URL predictive análisis:
https://github.com/Diego-Asturiano/Data_Science_Capstone_Falcon9_SpaceX_Launch/blob/666c475415295fdc01b0778dfb7fff901c8e4c0f/4.%20Machine%20Learning%20Prediction.ipynb

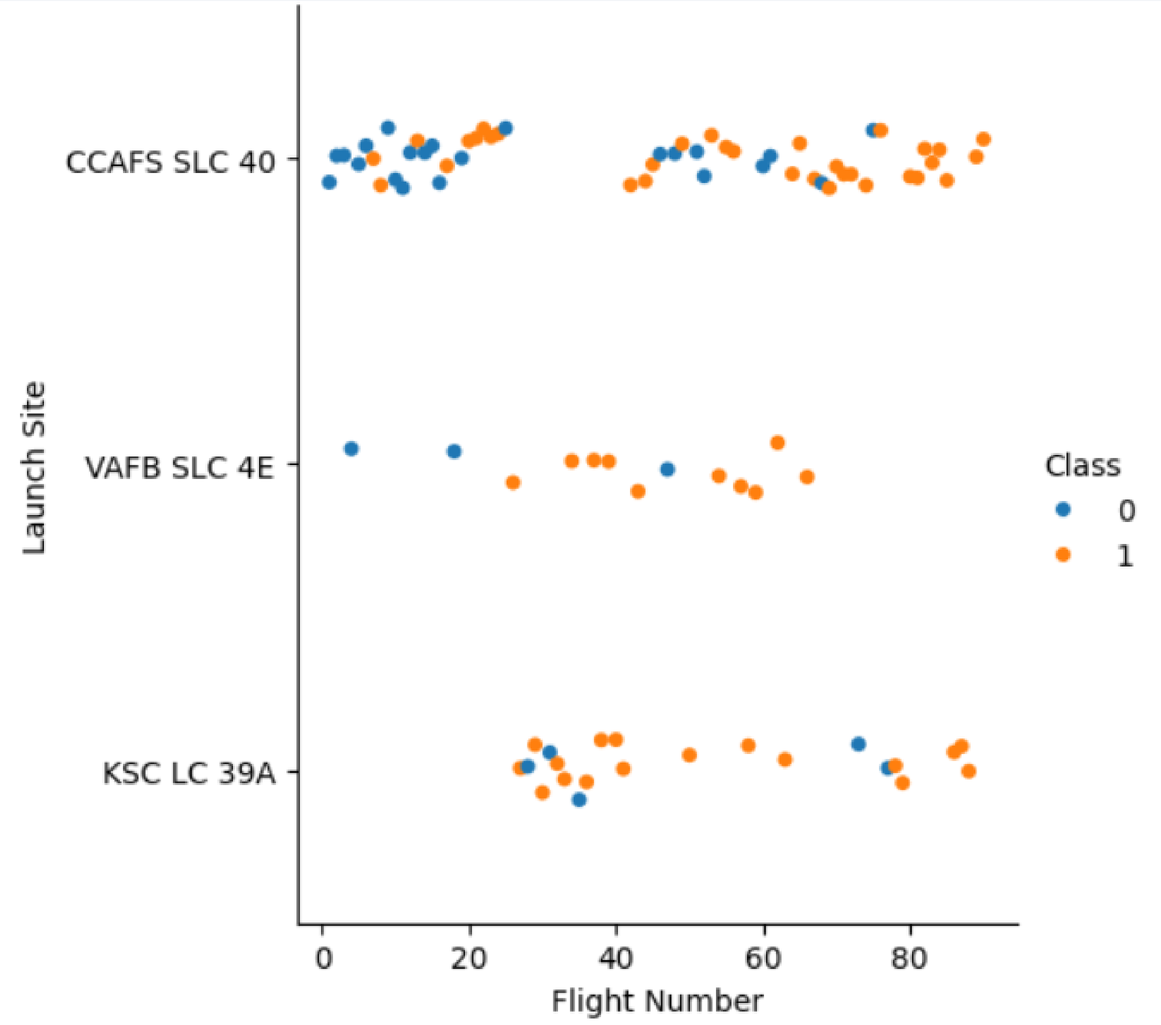
Results

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

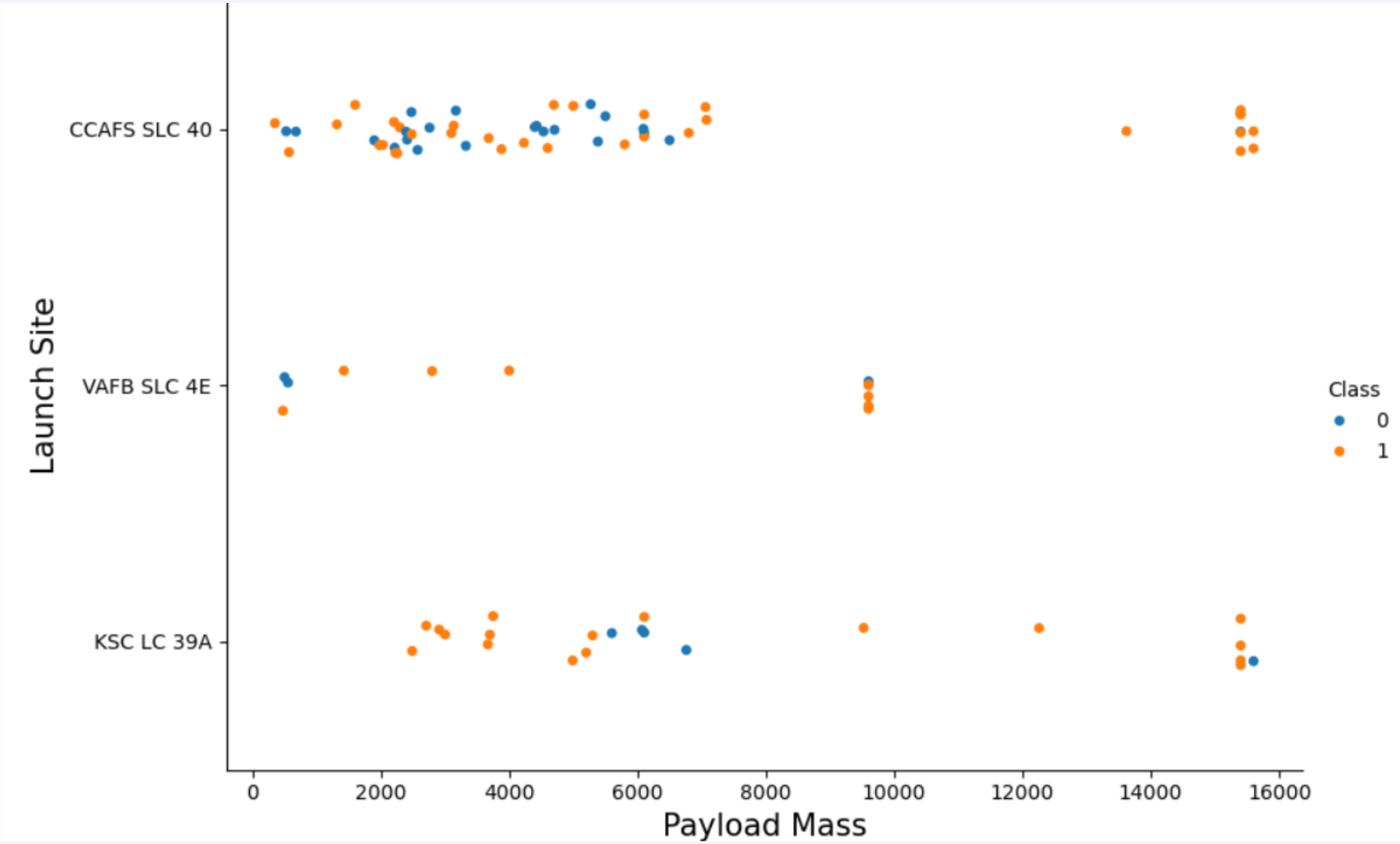
Section 2

Insights drawn from EDA

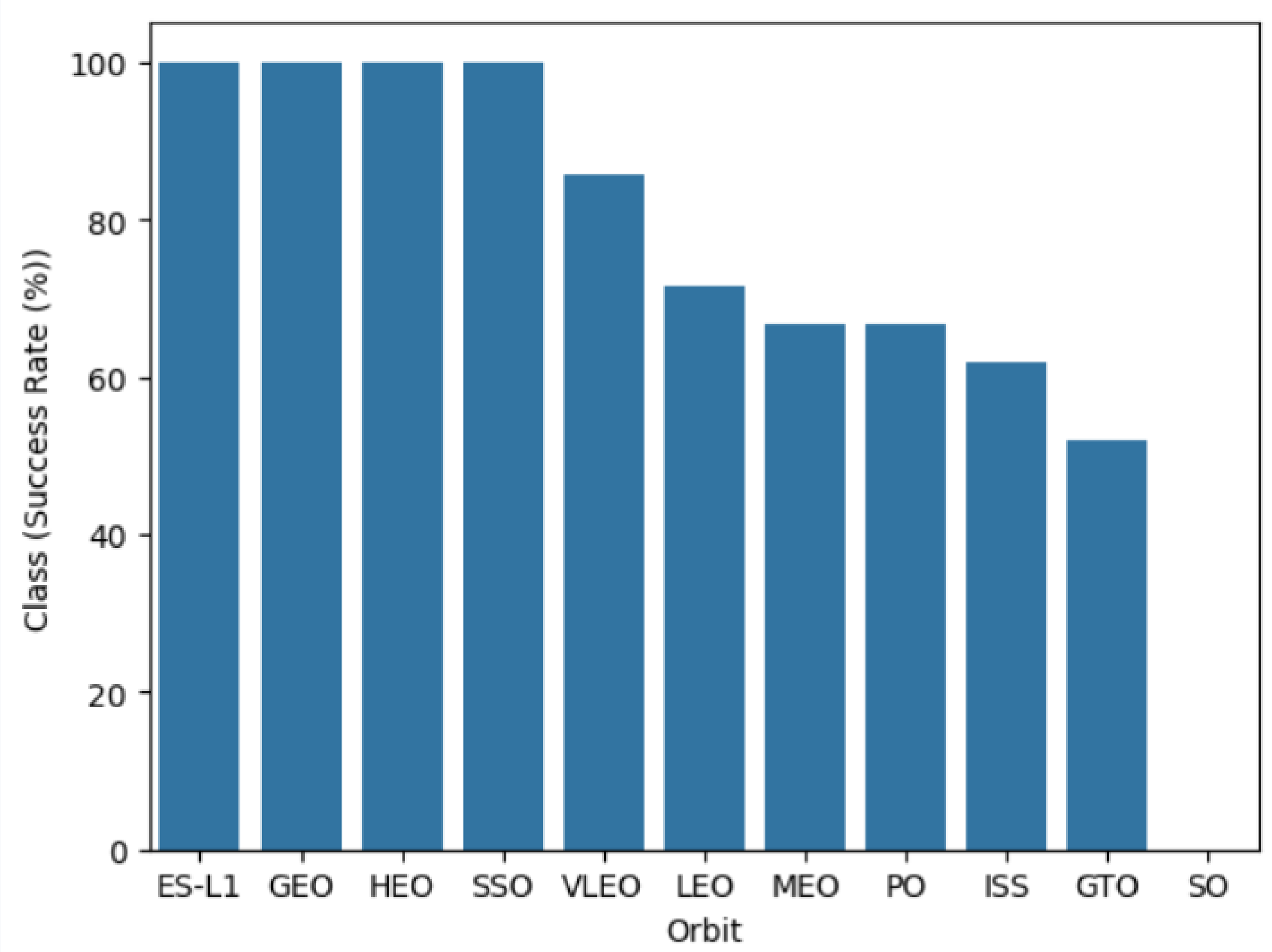
Flight Number vs. Launch Site



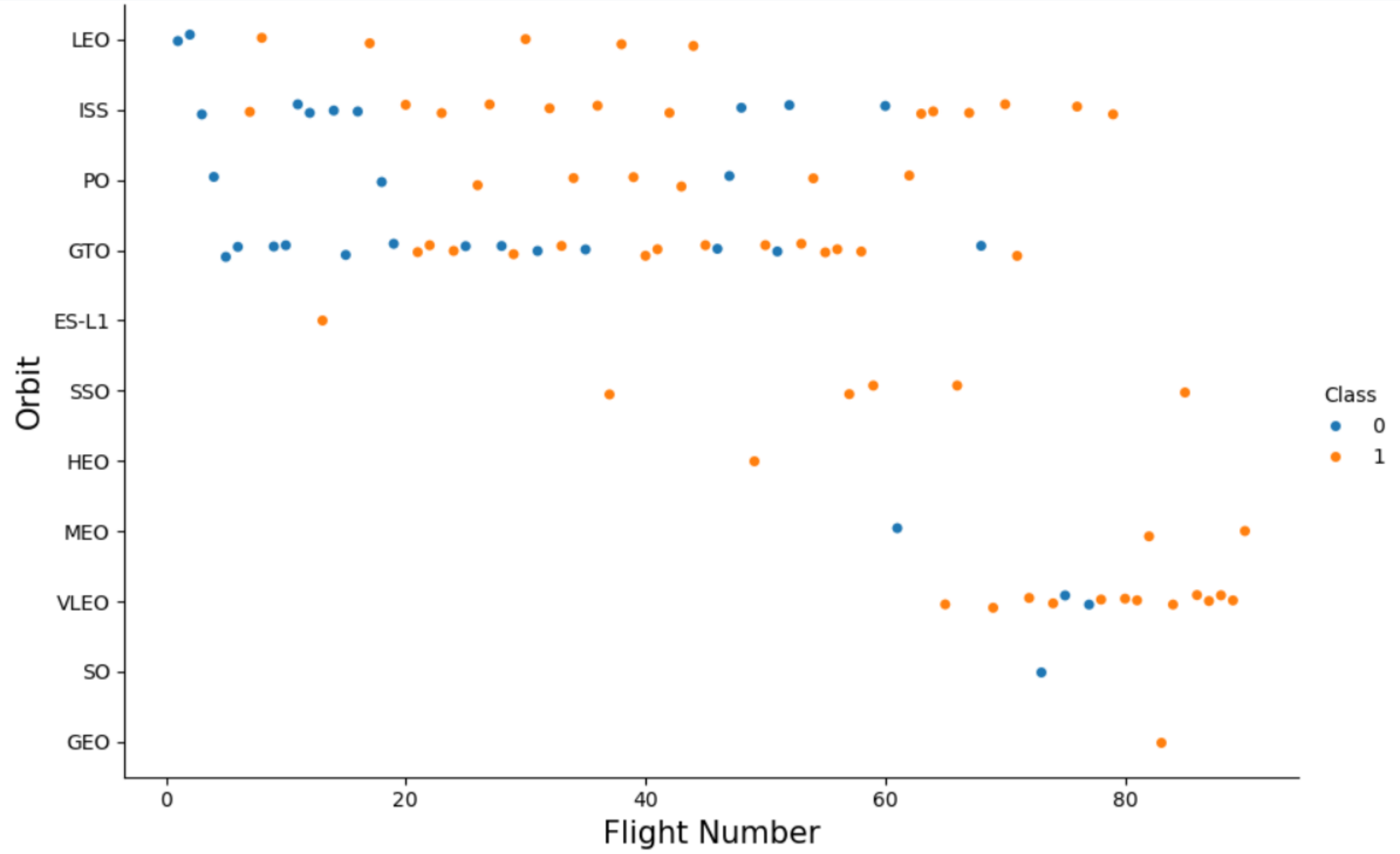
Payload vs. Launch Site



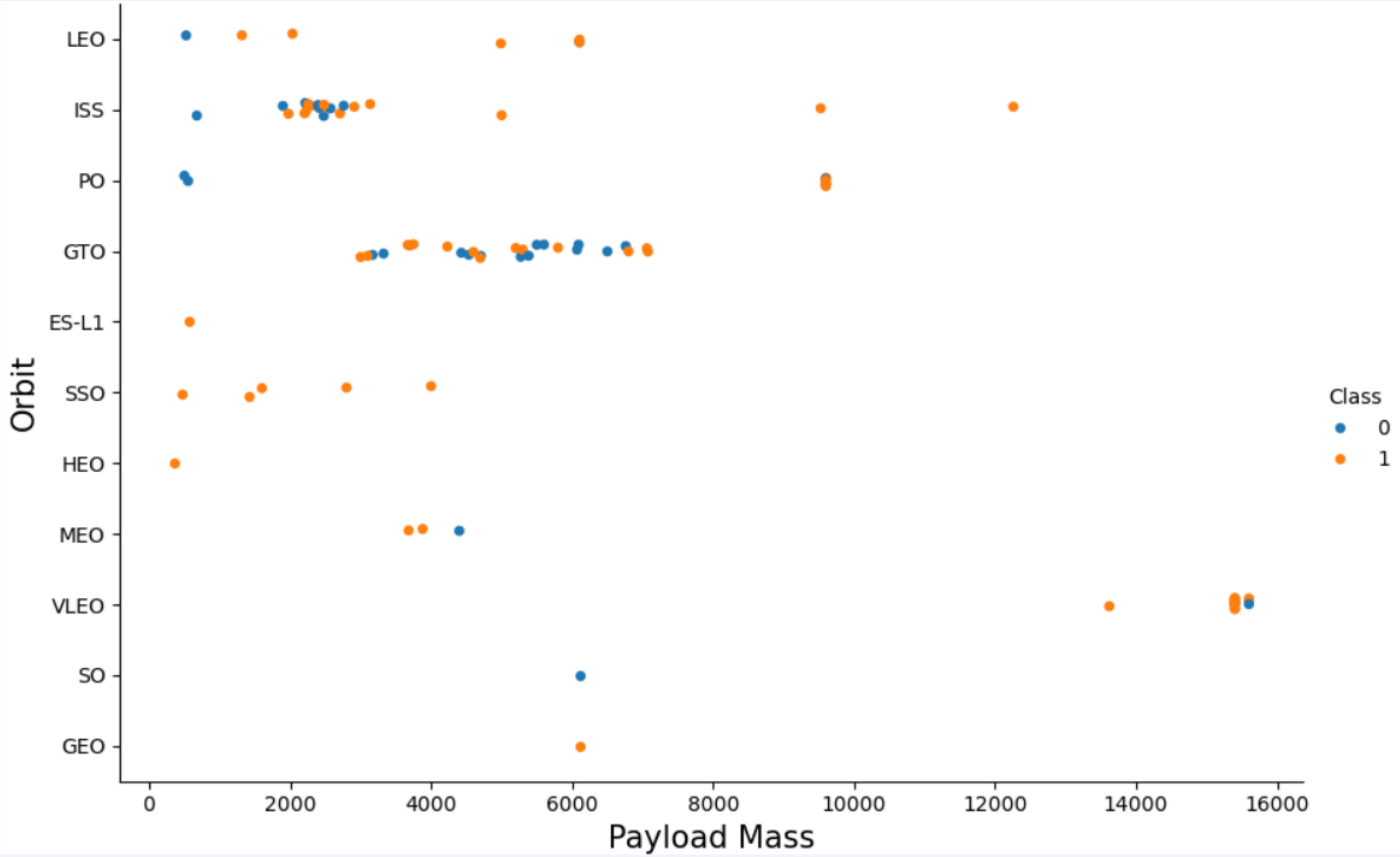
Success Rate vs. Orbit Type



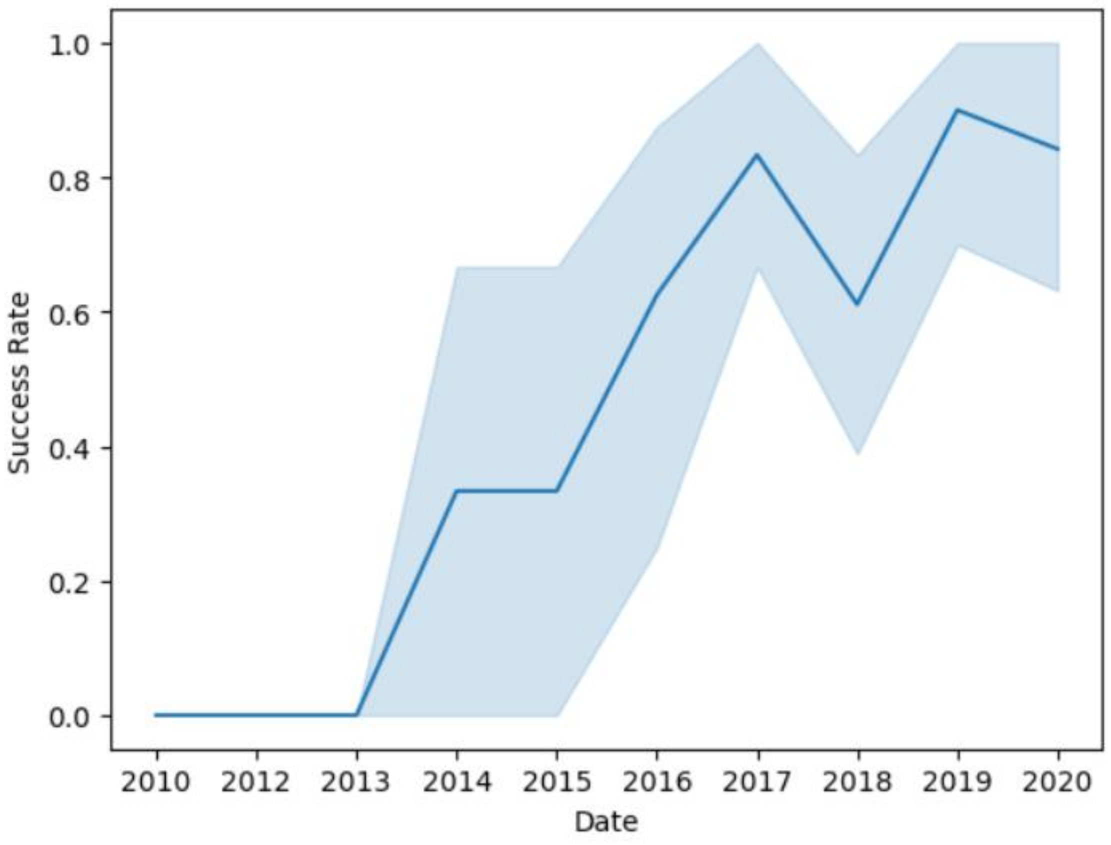
Flight Number vs. Orbit Type



Payload vs. Orbit Type



Launch Success Yearly Trend



you can observe that the sucess rate since 2013 kept increasing till 2020

All Launch Site Names

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

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

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

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

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`

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

```
%sql SELECT * FROM SPACEXTBL 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

Total Payload Mass

- Calculate the total payload carried by boosters from NASA

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

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

SUM(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

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

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

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

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

AVG(PAYLOAD_MASS__KG_)
2534.6666666666665

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad

List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

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

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

MIN(Date)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

```
%sql SELECT Booster_Version FROM SPACEXTBL 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

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes

```
%sql SELECT Mission_Outcome, COUNT(Mission_Outcome) FROM SPACEXTBL GROUP BY Mission_Outcome
```

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

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

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass

```
%sql SELECT Booster_Version, Payload, PAYLOAD_MASS__KG_ FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)
```

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

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

2015 Launch Records

- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

```
%sql SELECT substr(Date,7,4), substr(Date, 4, 2),"Booster_Version", "Launch_Site", Payload, "PAYLOAD_MASS__KG_", "Mission_Outcome", "Landing _Outcome"
```

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

substr(Date,7,4)	substr(Date, 4, 2)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Mission_Outcome	Landing _Outcome
2015	01	F9 v1.1 B1012	CCAFS LC-40	SpaceX CRS-5	2395	Success	Failure (drone ship)
2015	04	F9 v1.1 B1015	CCAFS LC-40	SpaceX CRS-6	1898	Success	Failure (drone ship)

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

- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

```
%sql SELECT * FROM SPACEXTBL WHERE Landing_Outcome LIKE 'Failure (drone ship)%' AND (Date BETWEEN '2010-06-04 ' AND '2017-03-20') ORDER BY Date DESC;
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2016-06-15	14:29:00	F9 FT B1024	CCAFS LC-40	ABS-2A Eutelsat 117 West B	3600	GTO	ABS Eutelsat	Success	Failure (drone ship)
2016-03-04	23:35:00	F9 FT B1020	CCAFS LC-40	SES-9	5271	GTO	SES	Success	Failure (drone ship)
2016-01-17	18:42:00	F9 v1.1 B1017	VAFB SLC-4E	Jason-3	553	LEO	NASA (LSP) NOAA CNES	Success	Failure (drone ship)
2015-04-14	20:10:00	F9 v1.1 B1015	CCAFS LC-40	SpaceX CRS-6	1898	LEO (ISS)	NASA (CRS)	Success	Failure (drone ship)
2015-01-10	9:47:00	F9 v1.1 B1012	CCAFS LC-40	SpaceX CRS-5	2395	LEO (ISS)	NASA (CRS)	Success	Failure (drone ship)

Section 3

Launch Sites Proximities Analysis

Launch sites on a map

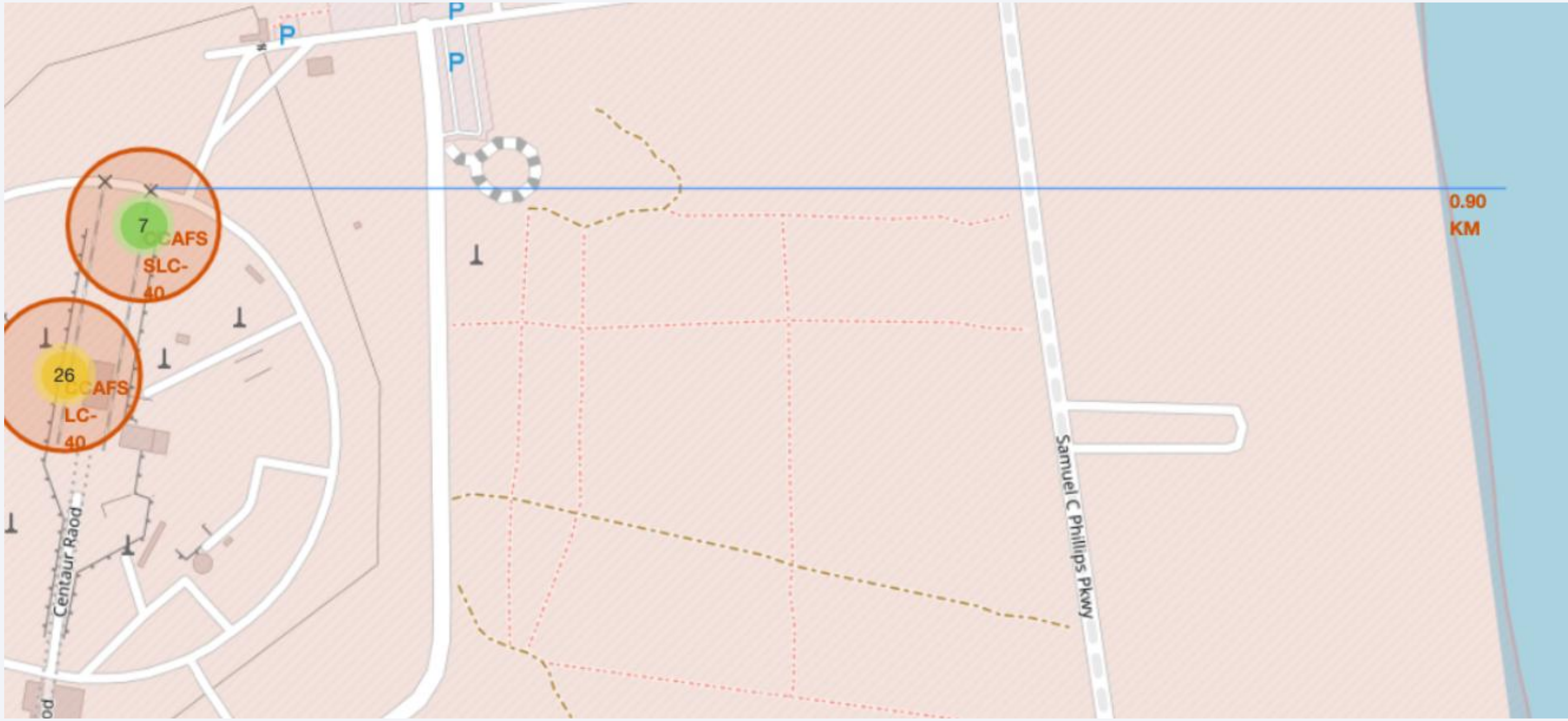
- We can see all the location markers of the launch points on a global map, we can see all the launch points very close to the coast.

Success/Failed launches for each site on the map

Show the color-labeled launch outcomes on the map.

Calculate the distances between a launch site to its proximities

Selected launch site to its proximities such as railway, highway, coastline, with distance calculated and displayed

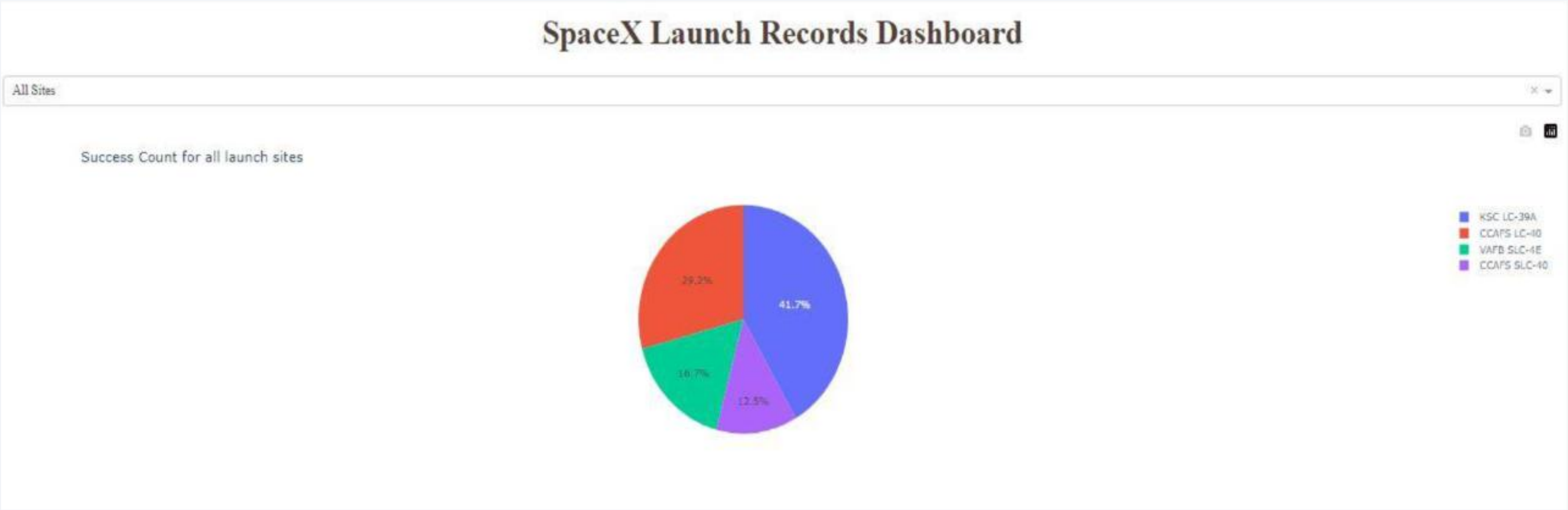


Section 4

Build a Dashboard with Plotly Dash

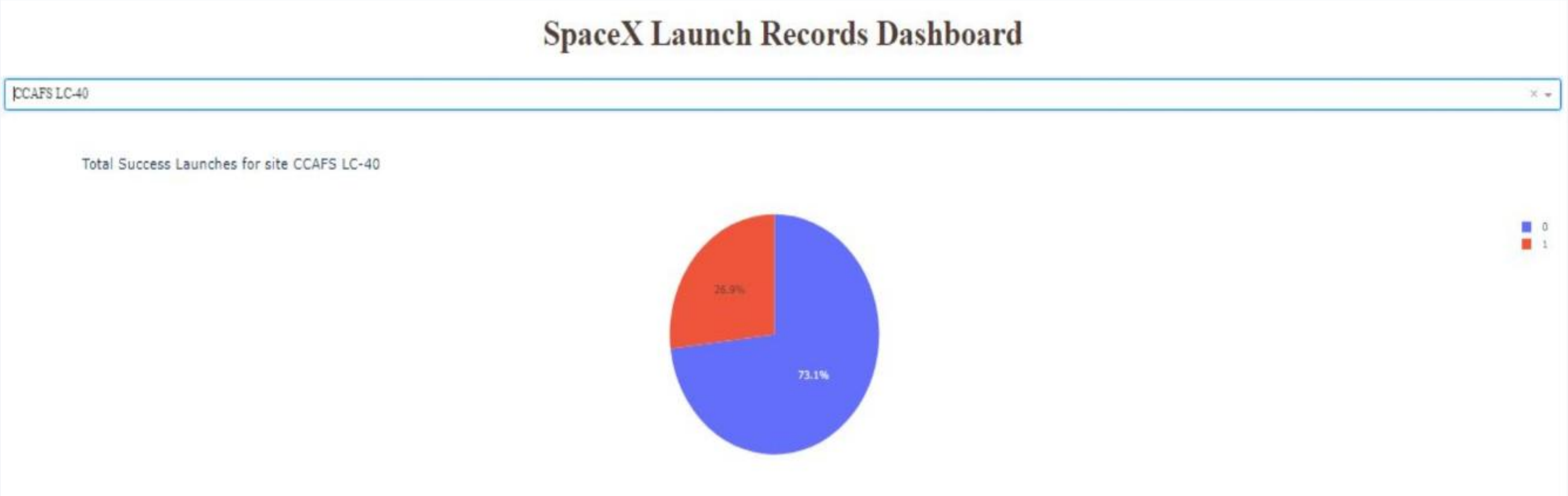
Launch success count for all launches sites

- Show the screenshot of launch success count for all sites, in a piechart.



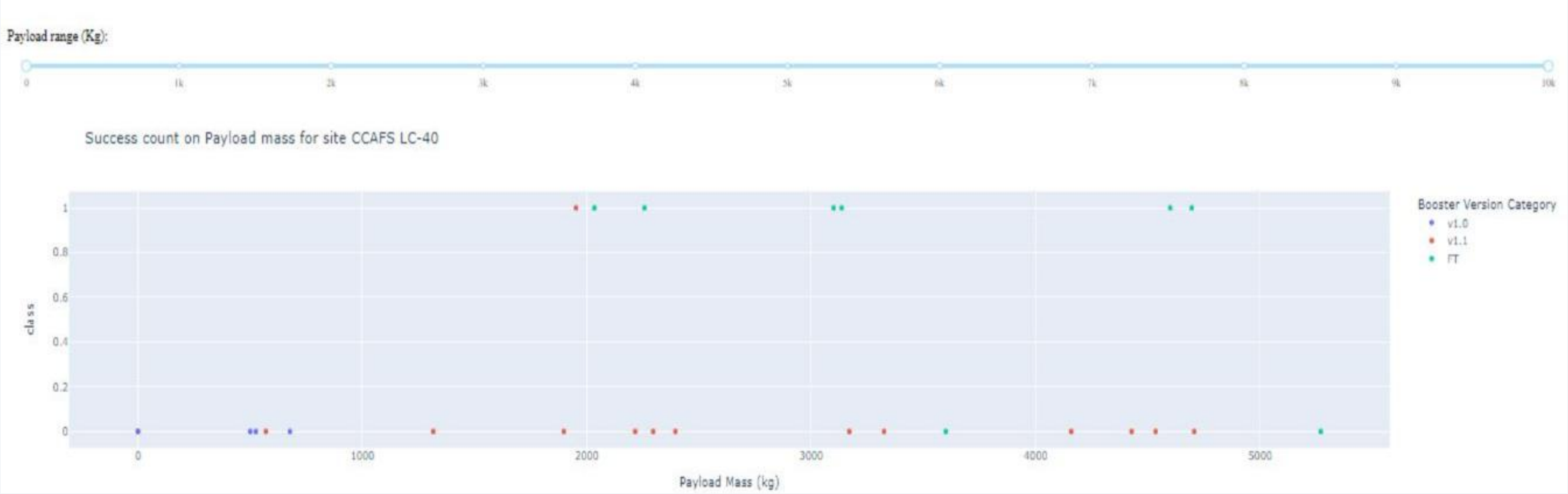
Launch site with highest launch success ratio

- Show the screenshot of the piechart for the launch site with highest launch success ratio.



Payload vs. Launch Outcome scatter plot for all sites

- Show screenshots of Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider.



Section 5

Predictive Analysis (Classification)

Classification Accuracy

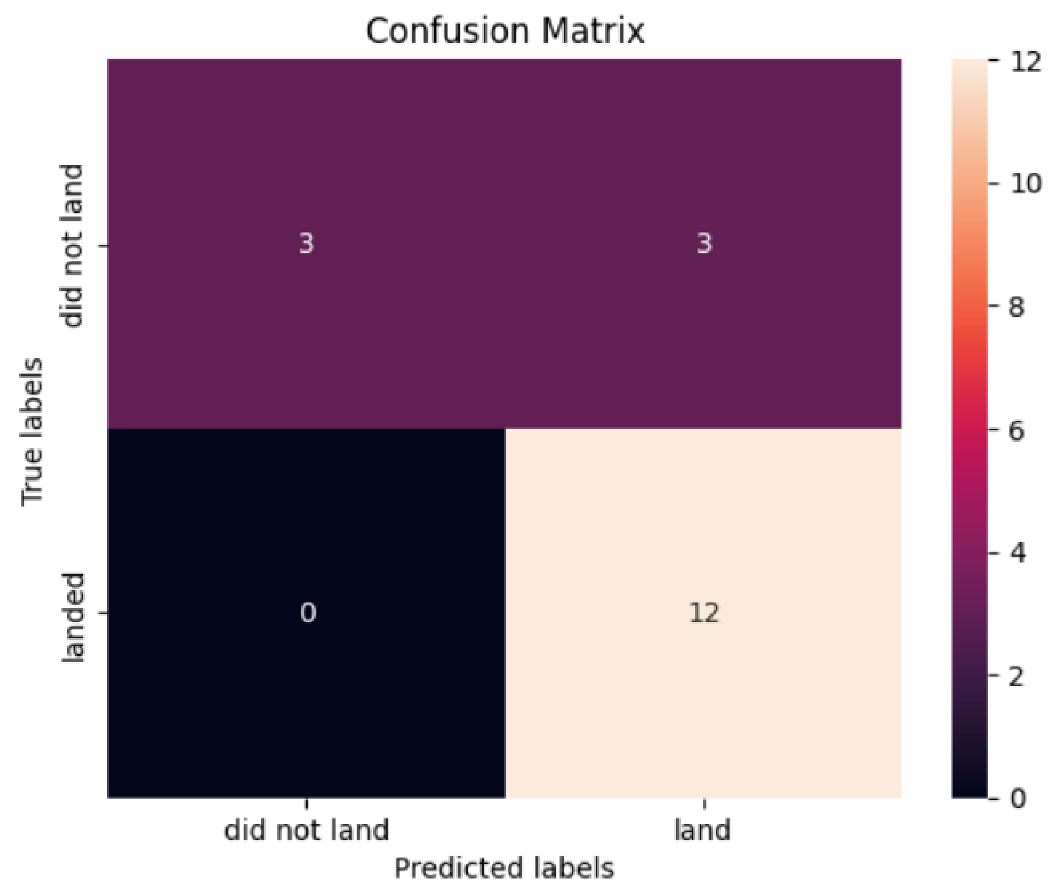
- Model accuracy for all built classification models, in a bar chart

Method	Test Data Accuracy
Logistic_Reg	0.833333
SVM	0.833333
Decision Tree	0.944444
KNN	0.833333

- The model who has the highest classification accuracy is Decision Tree

Confusion Matrix

- Confusion matrix of the best performing model. Accuracy of knn_cv on the test data using the method score.



Conclusions

- Machine learning techniques were applied to predict the landing success of the first leg of the rocket, using models such as logistic regression, decision tree and SVM.
- Different models were compared using metrics such as precision and accuracy, identifying which ones offer better predictions based on variables such as launch site, orbit type and payload.
- The project helps to make informed data science decisions.
- This experience simulates a real use case in the aerospace industry, showing how data science can support critical decisions in space missions.

Thank you!