
SPACE X FALCON 9
Analysis
Nasmane Abdelhadi
23th August 2023
SPACEX

Table Of Content

Executive Summary

- Methodologies
  - Data Collection
  - Data Wrangling
  - EDA with SQL & Data Visualization
  - Interactive Visual Analytics with Folium and Plotly/Dash
  - Machine Learning Prediction
- Results
  - EDA results
  - Interactive Analytics results
  - Predictive Analytics results

Introduction

- 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.
- Main Question
  - Can We determine whether a rocket will land successfully or not?

Methodology

- Data collection methodology (Through API and Web Scraping)
- Data wrangling (One Hot Encoding)
- EDA using visualization and SQL
- Interactive visual analytics using Folium and Plotly Dash
- Predictive analysis using classification models

Data Collection

- The data was collected using various methods
  - Data collection was done using get-request to the SpaceX API.
  - Cleaning the data, checking for missing values and filling in missing values where necessary.
  - Web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.

SpaceX API

✓ Get-request to the SpaceX API to collect data

✓ Cleaning the requested data and doing some basic data wrangling and formatting.

Web Scraping

✓ Web scraping to Wikipedia Falcon 9 launch records with BeautifulSoup

✓ Parsing the table and converting it into a pandas DataFrame.

Data Wrangling

```
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()
```

[7]

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

```
# Apply value_counts on Orbit column
df['Orbit'].value_counts()
```

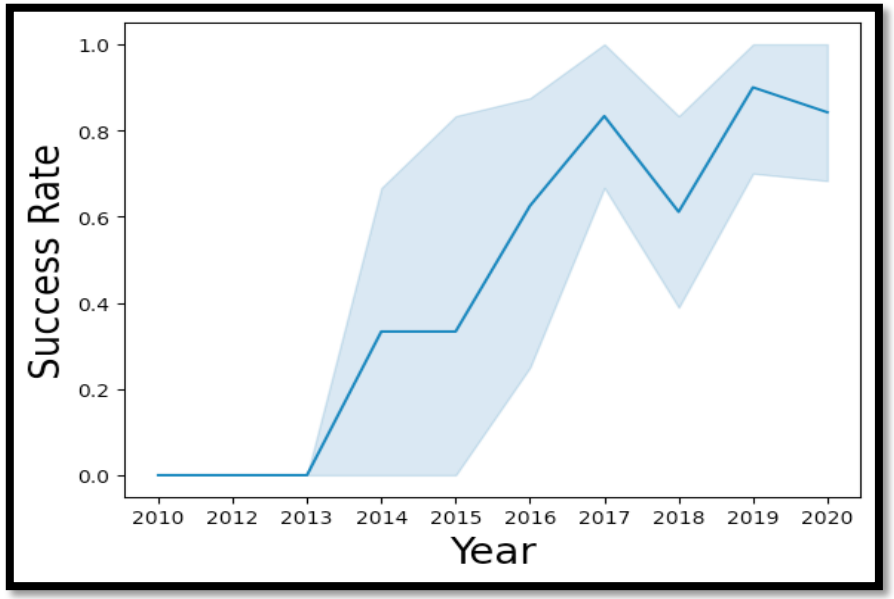
[8]

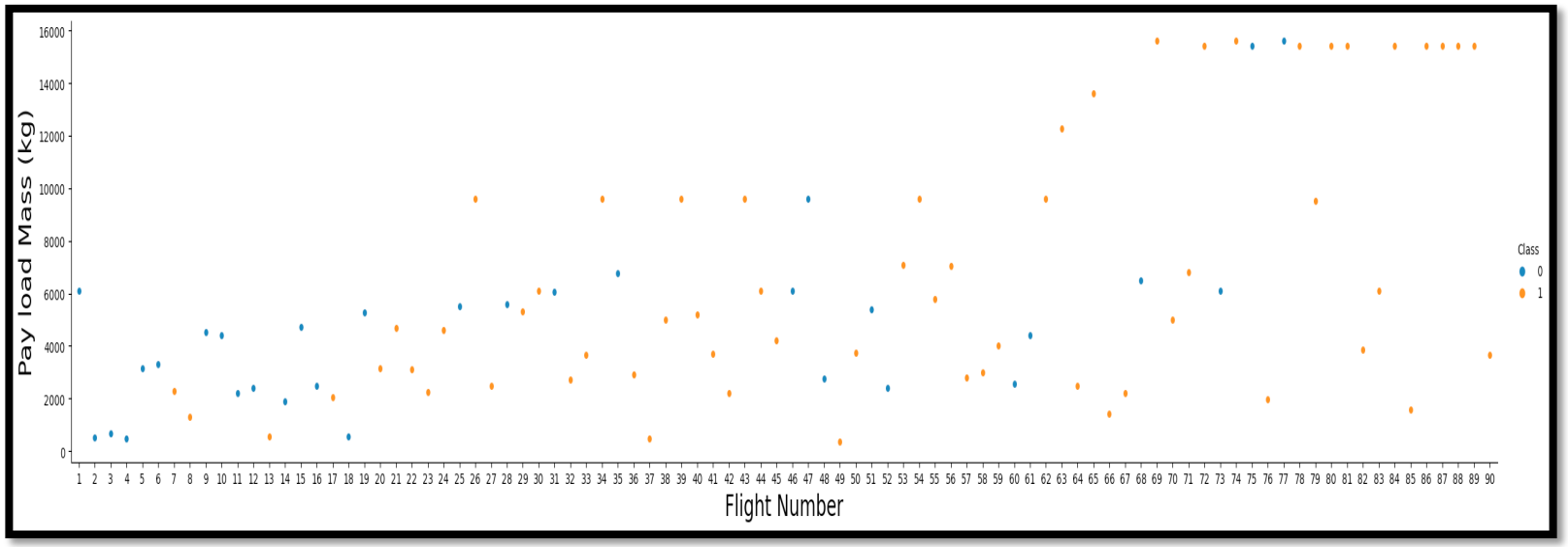
```
GTO     27
ISS     21
VLEO    14
PO       9
LEO      7
SSO      5
MEO      3
ES-L1    1
HEO      1
SO       1
GEO      1
Name: Orbit, dtype: int64
```

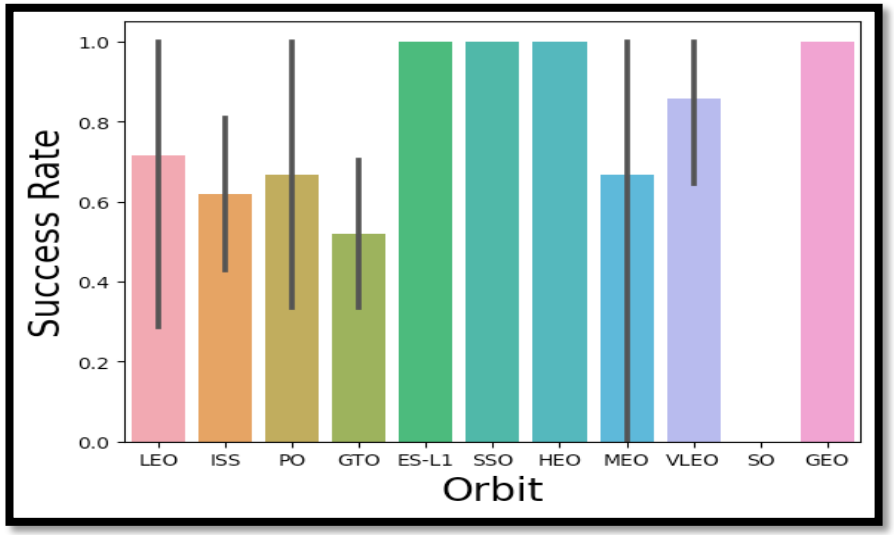
- ✓Performing EDA and determined the training labels.
- ✓Calculating number of launches at each site, and the number and occurrence of each orbits

EDA with Data Vizualization

- ✓Highlighting the relationship between flight number and launch Site, payload and launch site, success rate of each orbit type, flight number and orbit type, the launch success yearly trend.







EDA with SQL

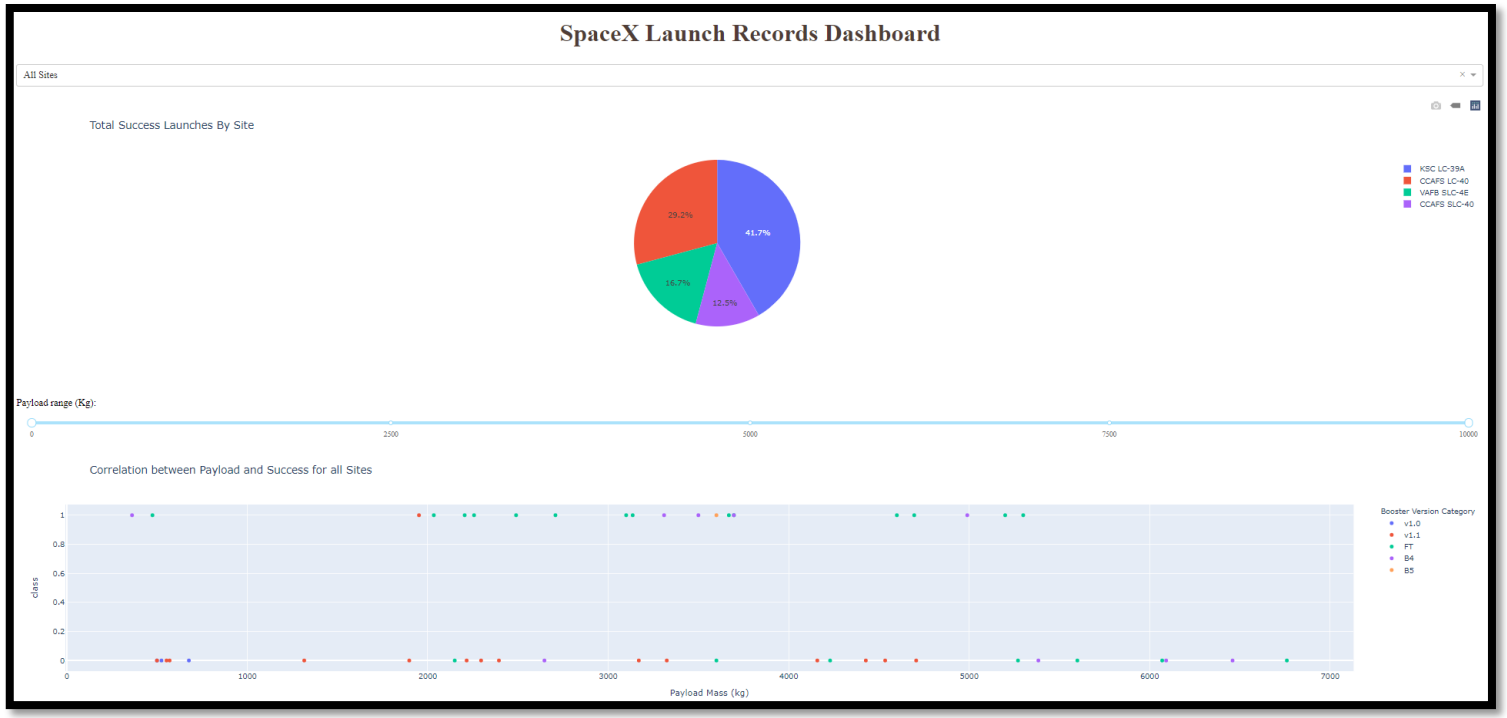
- ✓ Some Queries
  - ✓ The names of unique launch sites in the space mission.
  - ✓ The total payload mass carried by boosters launched by NASA (CRS)
  - ✓ The average payload mass carried by booster version F9 v1.1
  - ✓ The total number of successful and failure mission outcomes
  - ✓ The failed landing outcomes in drone ship, their booster version and launch site names.

Interactive Map Using Folium

- ✓Marking all launch sites, and adding map objects to highlight the success or failure of launches for each site on the folium map.
- ✓Assigning the feature launch outcomes to class 0 and 1.
- ✓Using the color-labeled marker clusters, and identifying which launch sites have relatively high success rate.

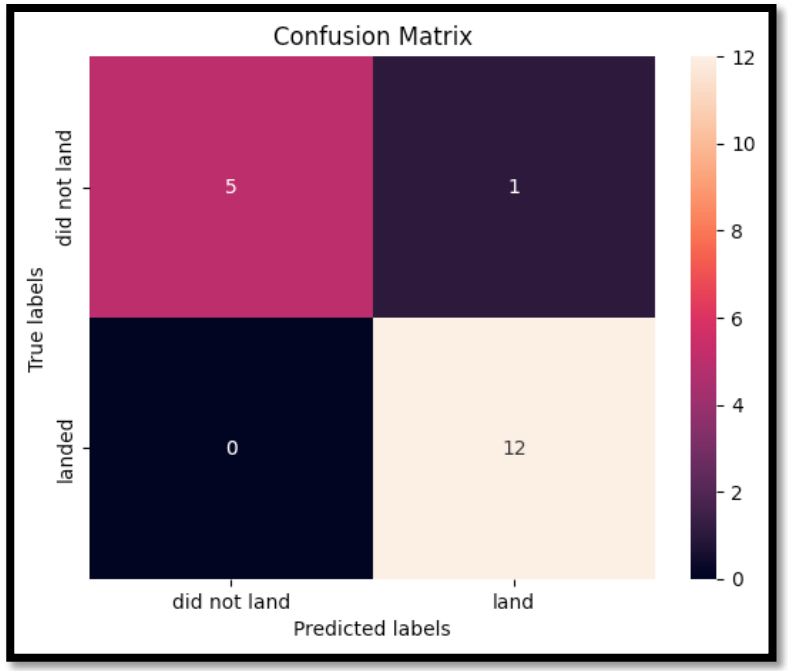
Building a Dashboard

- ✓Building an interactive dashboard with Plotly dash
- ✓Plotting pie charts showing the total launches by a certain sites
- ✓Plotting scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.



Predictive Analysis

- Splitting Data into Training/Testing parts
- Building different machine learning models and tune different hyperparameters using GridSearchCV.
- Using R2 score to select the best model.




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

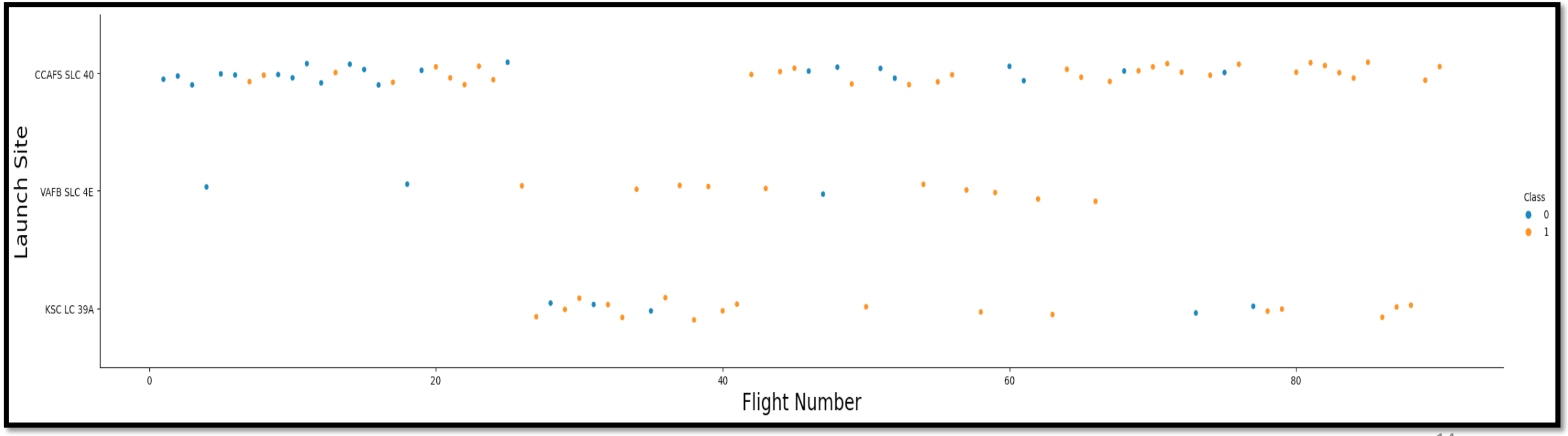
```
[13]
...  tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
     accuracy : 0.8222222222222222
```

Results

- EDA results
- Interactive Analytics results
- Predictive Analytics results

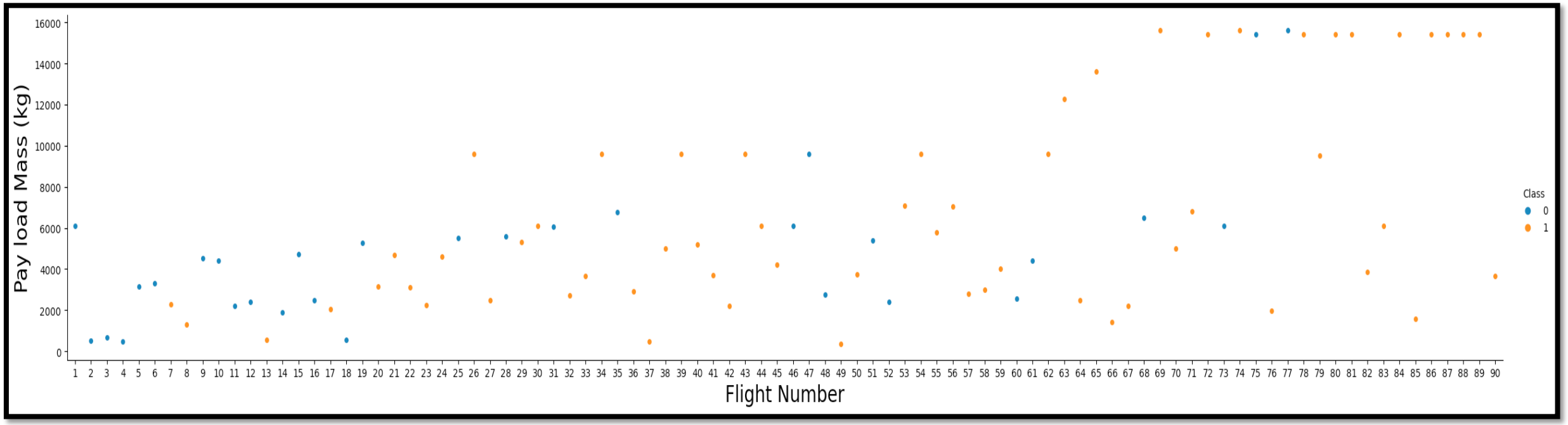
Flight Number vs Launch Site

- ✓We can see how each launch Site has done well recently because of the experience gained from flights



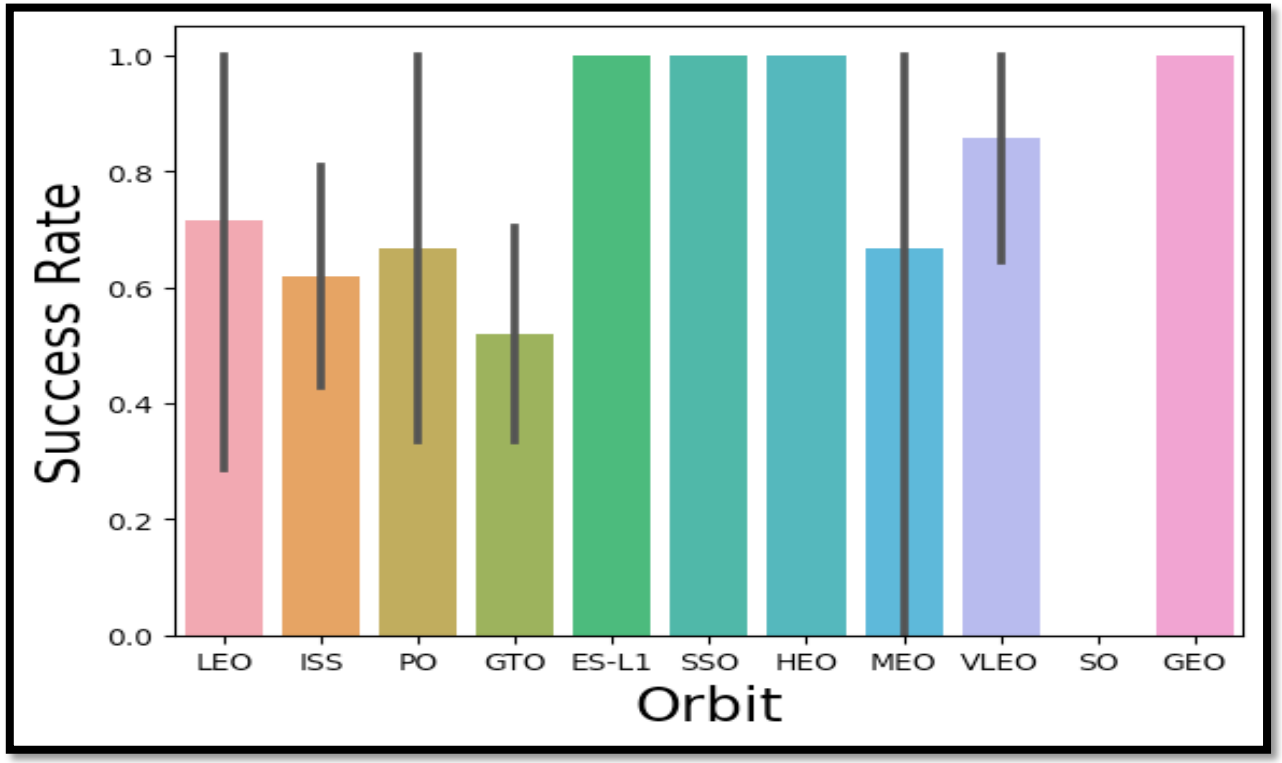
Flight Number vs Payload Mass

✓The higher the Payload Mass, the higher the success rate.



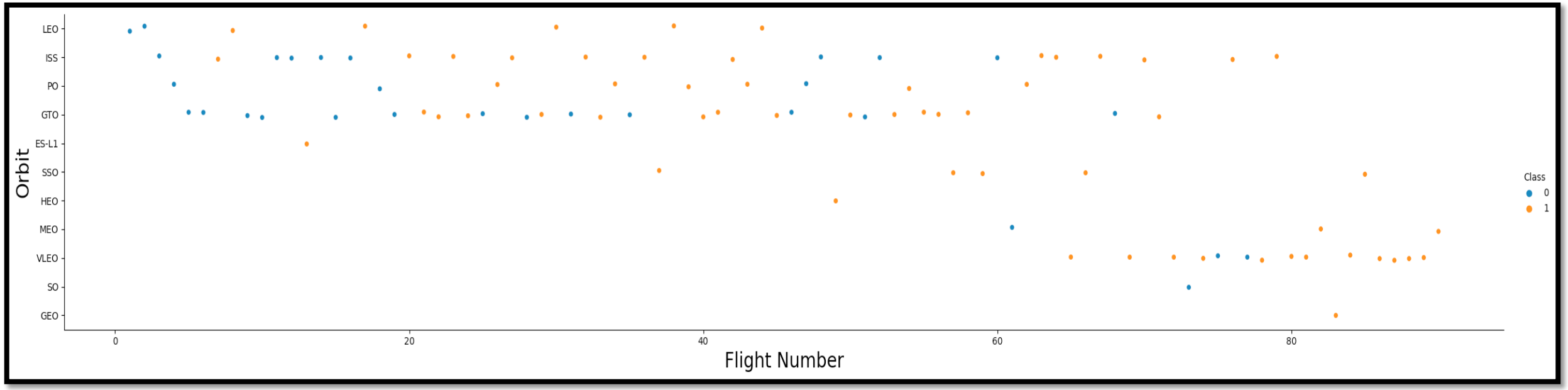
Success Rate per Orbit

- ✓SSO, HEO, ES-L1, GEO has the highest Success Rate



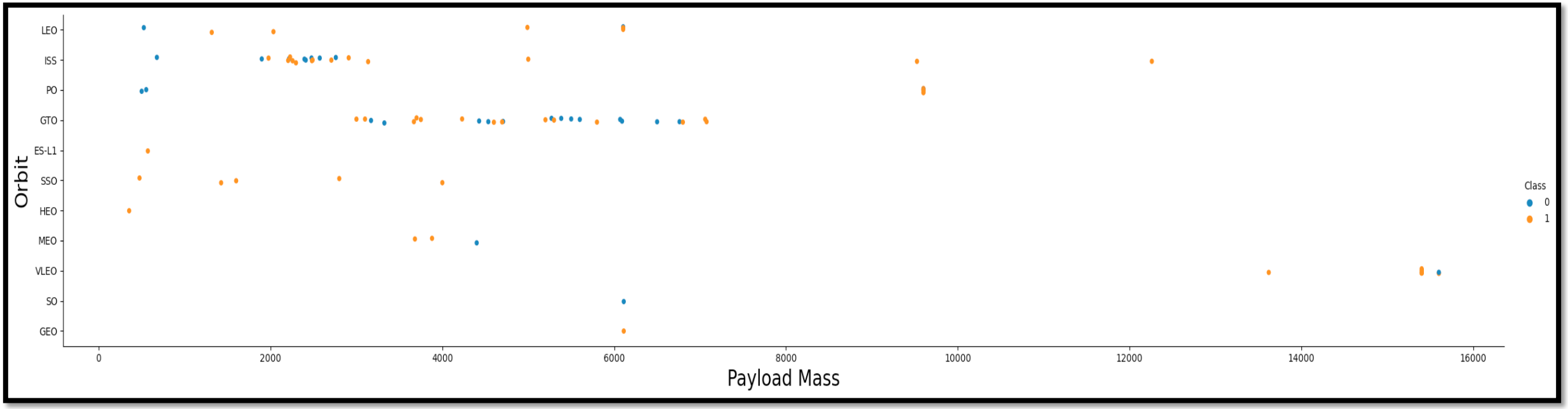
Flight Number for each Orbit

- ✓All flights of SSO, HEO, ES-L1, GEO were Successful



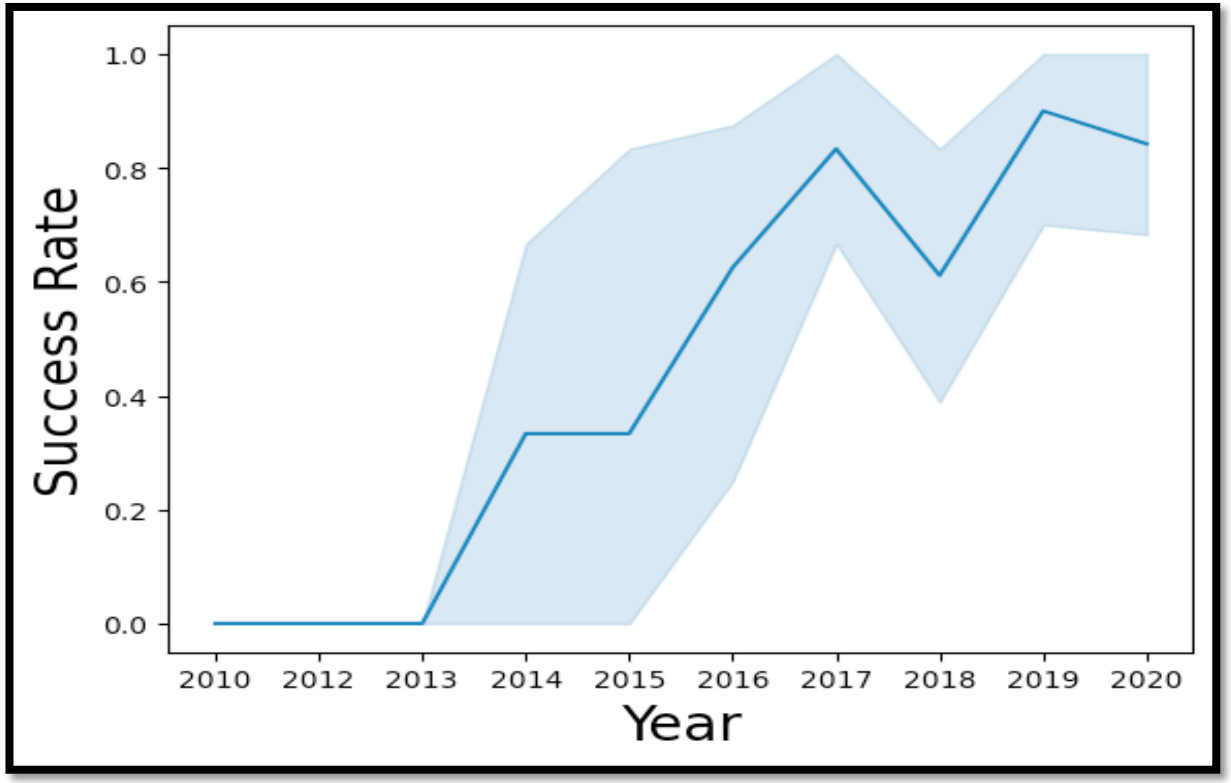
Payload Mass used for each Orbit

- ✓Even if the payload mass for SSO orbit is low, the flights were successful, also the higher the payload mass the higher the success rate



Launch Success Rate during the Years

- ✓The Success Rate is increasing over the years due to the experience gained.



Extracting Launch Sites Name's

```
%%sql
--The first query will serve to determine the name of columns
--SELECT * FROM SPACEXTABLE LIMIT 1;

SELECT DISTINCT("Launch_Site") FROM SPACEXTABLE;
```

[8]

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

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

Launch Sites Names beginning with CAA

```
%%sql

SELECT * FROM SPACEXTABLE
WHERE "Launch_Site" LIKE "CCA%"
LIMIT 5;
```

[9] Python

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass carried NASA's boosters

```
%%sql

SELECT SUM("PAYLOAD_MASS__KG_") FROM SPACEXTABLE
WHERE "Customer" LIKE "%NASA%";
```

[10]

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

SUM("PAYLOAD_MASS__KG_")
107010

Average Payload Mass carried by booster F9 V1.1

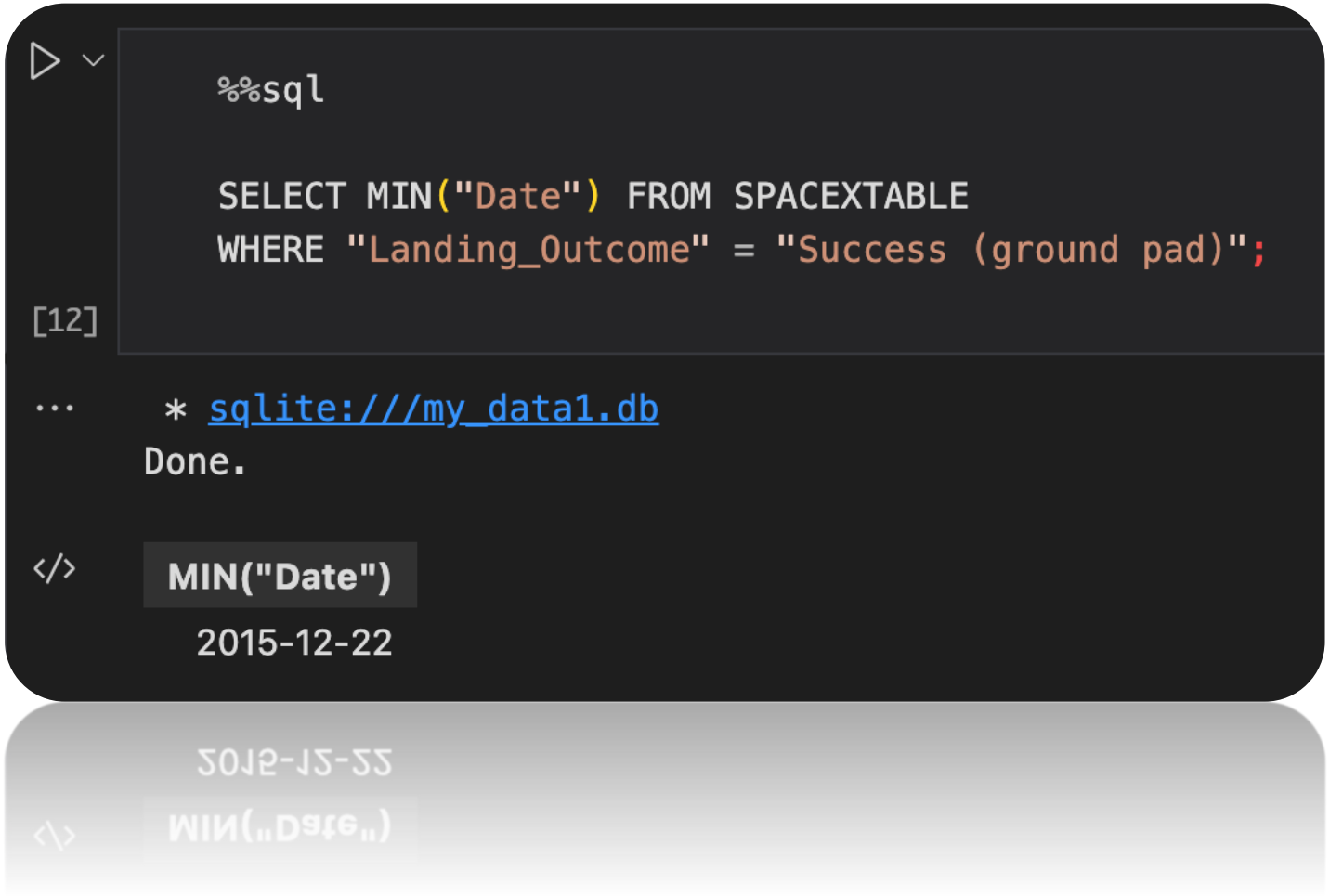
```
%%sql

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

[11]

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

AVG("PAYLOAD_MASS__KG_")
2534.6666666666665

Date of the first successful landing in ground pad



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_" BETWEEN 4000 AND 6000);
```

[13]

```
 * 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

```
[14]
    success = cur.execute('SELECT COUNT(*) FROM SPACEXTABLE WHERE "Mission_Outcome" LIKE "%Success%";')
    print(f"Number of successful mission outcomes: {success.fetchone()[0]}")


    failure = cur.execute('SELECT COUNT(*) FROM SPACEXTABLE WHERE "Mission_Outcome" LIKE "%Failure%";')
    print(f"Number of failure mission outcomes :{failure.fetchone()[0]}")
```

```
Number of successful mission outcomes: 100
Number of failure mission outcomes :1
```

Names of the booster_versions which have carried the maximum Payload Mass

```
%%sql

SELECT "Booster_Version" FROM SPACEXTABLE
WHERE "PAYLOAD_MASS__KG_" = (
                            SELECT MAX("PAYLOAD_MASS__KG_") FROM SPACEXTABLE
                            );
```

[15]

```
 * 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 records displaying 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');
```

[16]

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

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

Ranking the count of landing outcomes between the date 2010-06-04 and 2017-03-20

```
%%sql

SELECT "Landing_Outcome", COUNT("Landing_Outcome") AS "Total Landing_Outcome" FROM SPACEXTABLE
WHERE "Date" BETWEEN "2010-06-04" and "2017-03-20"
GROUP BY "Landing_Outcome"
ORDER BY "Total Landing_Outcome" DESC;
```

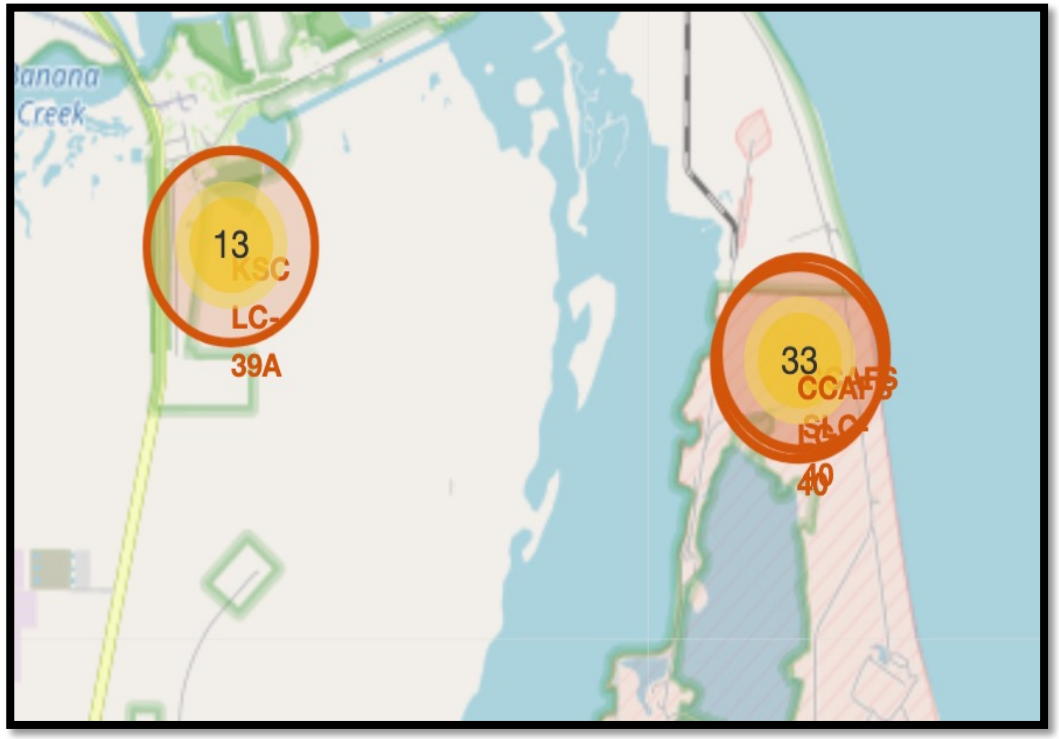
[18]

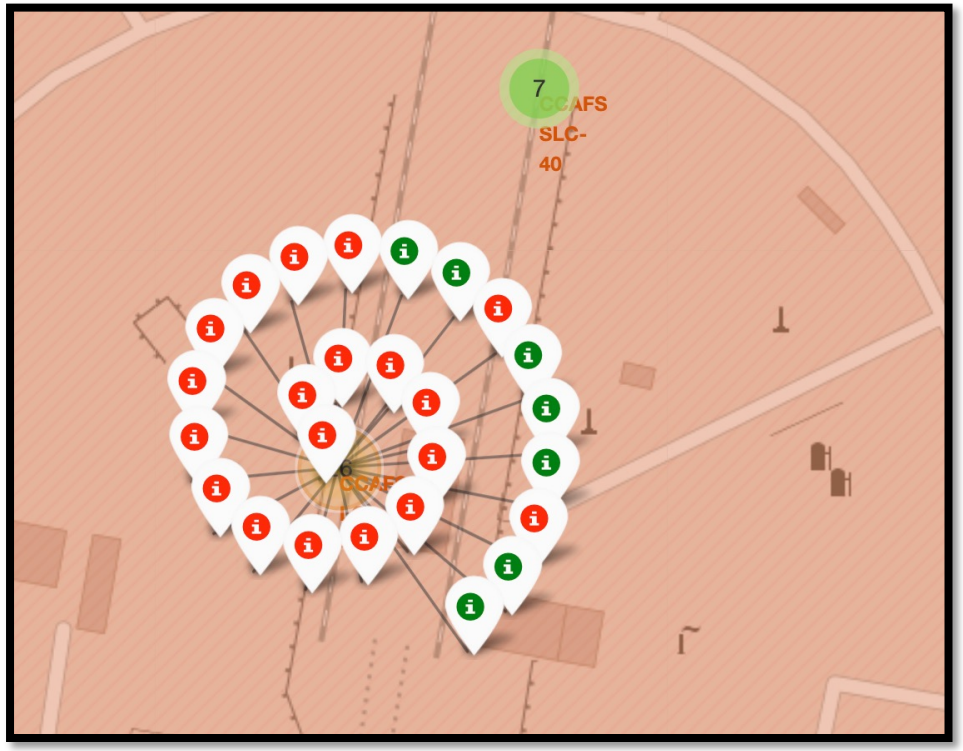
```
 * sqlite:///my_data1.db
Done.
```

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

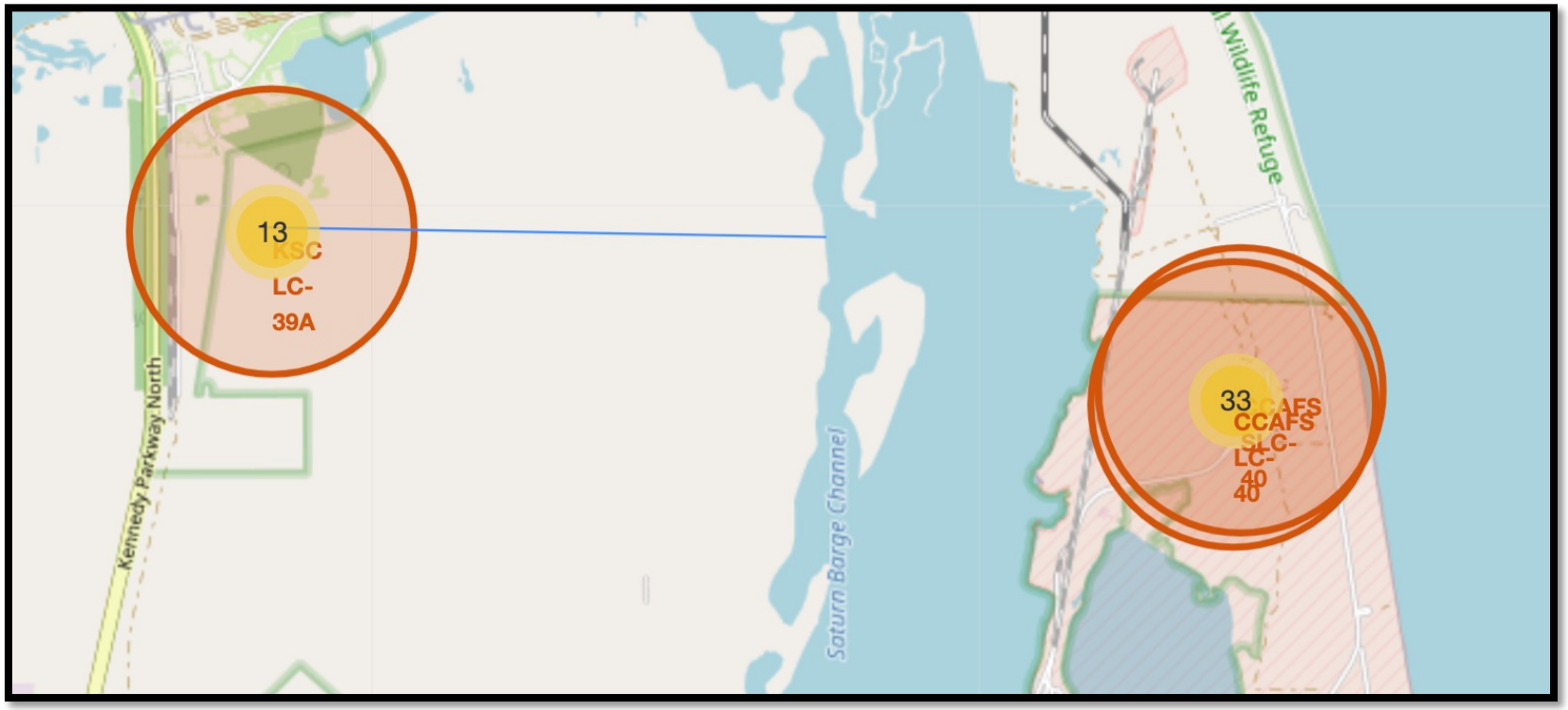
Marking the 3 Launch Sites on the Map

Marking All Flights using MarkerCluster



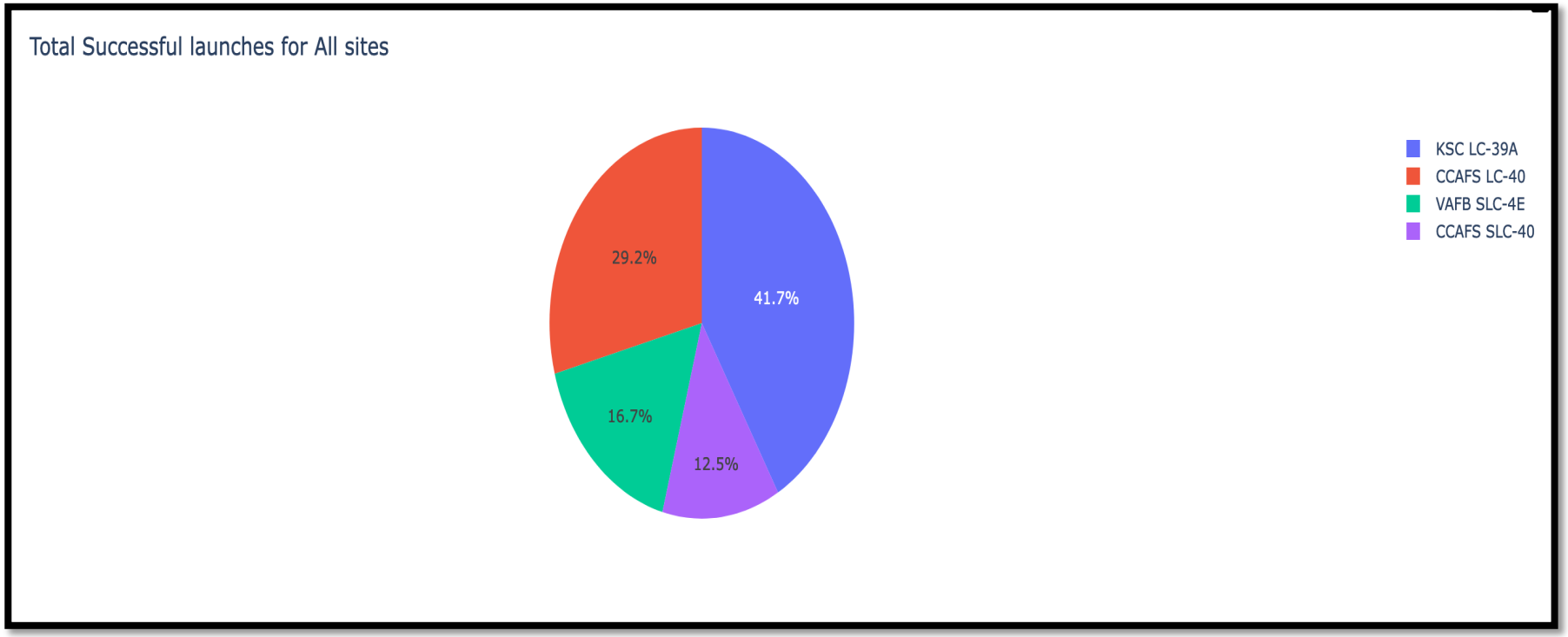


Finding the closest coastline to KSC station



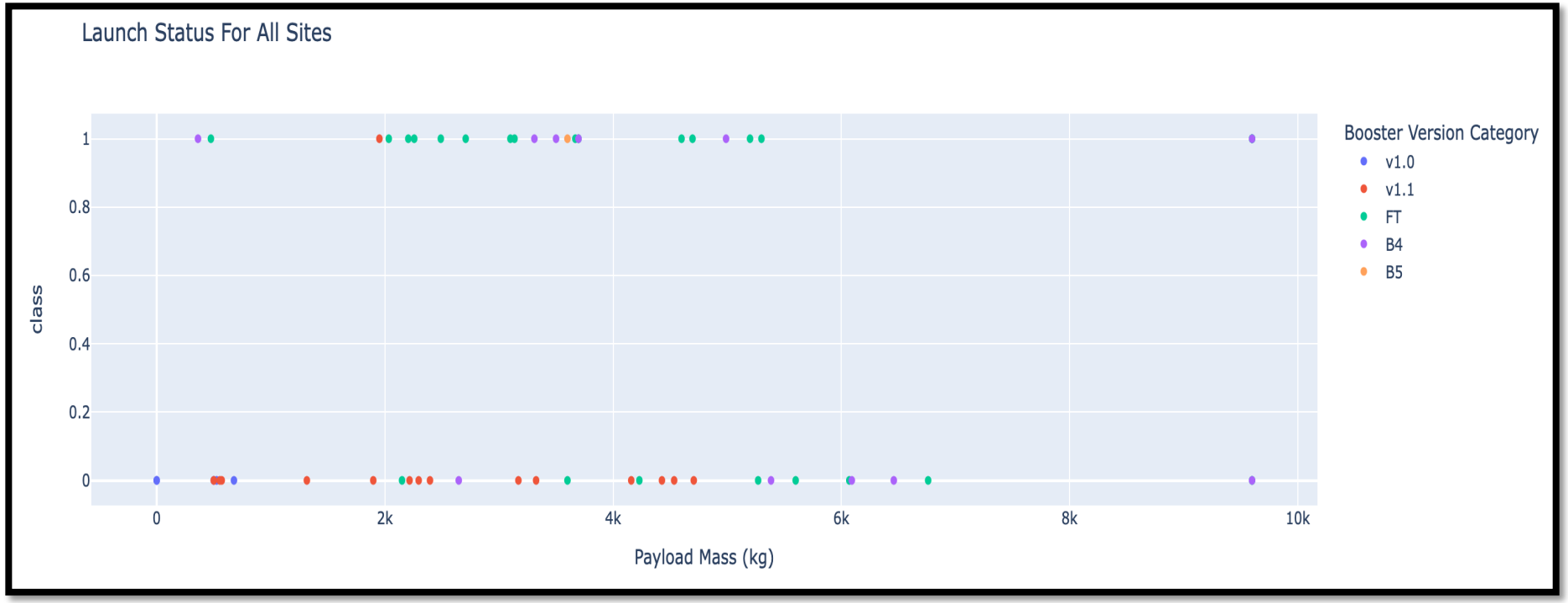
Total Successful launches for All sites

- ✓ KSC LC-39A has the heighest number of successful flights



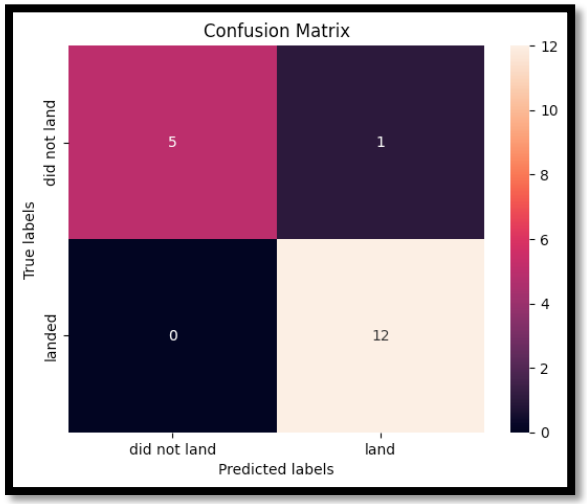
Total Successful launches for All sites

- ✓ It seems that there is no correlation between the payload mass and the status of the flight



Creating a logistic Regression Model

- ✓ It turns out that this model is the best model, since it has the highest R2 score on the test Data




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

lr=LogisticRegression()

logreg_cv = GridSearchCV(lr, parameters, cv=10)
logreg_cv.fit(X, Y)
```

GridSearchCV
estimator: LogisticRegression
LogisticRegression

We output the GridSearchCV object for logistic regression. We display the best parameters using the data att

```
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 : 0.8222222222222222
```

TASK 5

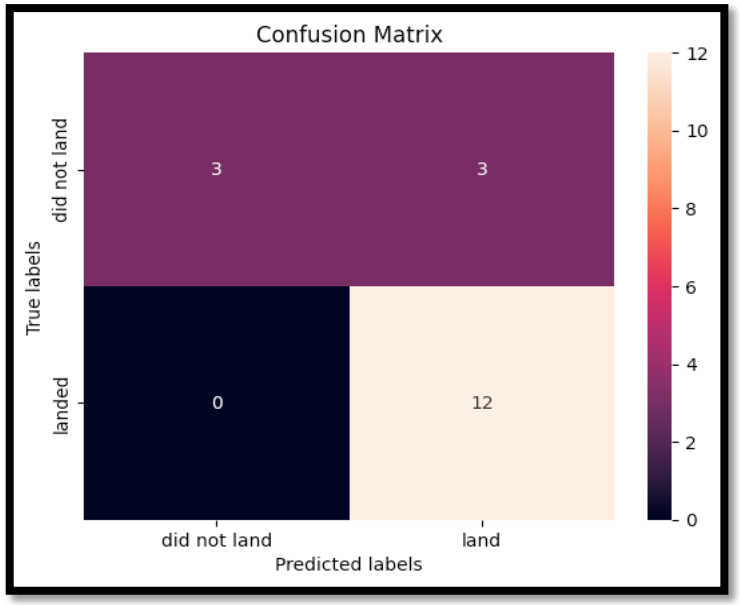
Calculate the accuracy on the test data using the method score:

```
r2 = logreg_cv.score(X_test, Y_test)
r2
```

```
0.9444444444444444
```

Creating a SVM Model

✓ This model is suitable, it has a good R2 score.




```
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)
```

GridSearchCV
estimator: SVC
SVC

```
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
```

TASK 7

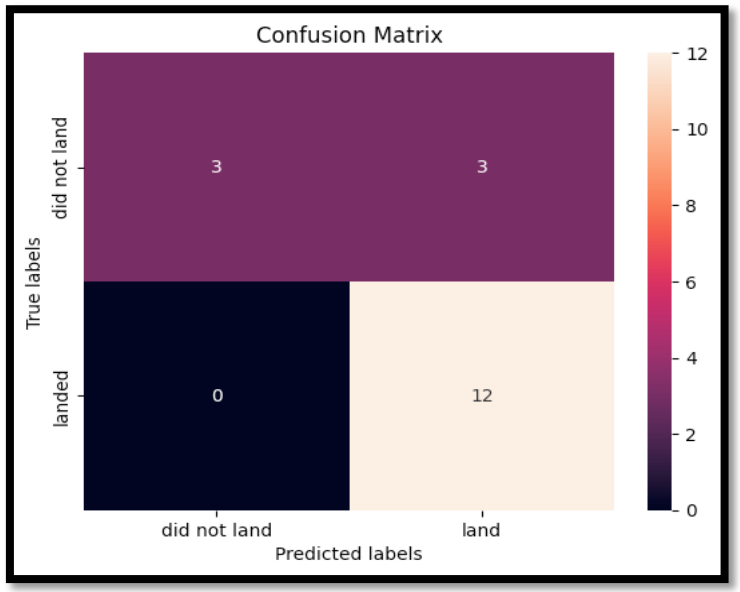
Calculate the accuracy on the test data using the method `score`:

```
r2 = svm_cv.score(X_test, Y_test)
r2
```

```
0.8333333333333334
```

Creating a Decision Tree Model

✓ This model is suitable, it has a good R2 score.




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

tree = DecisionTreeClassifier()
```

[21]

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

[22]

```
haaaa
```

GridSearchCV
estimator: DecisionTreeClassifier
DecisionTreeClassifier

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

[23]

```
tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 14, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_sample
accuracy : 0.8875
```

TASK 9

+ Code + Markdown

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

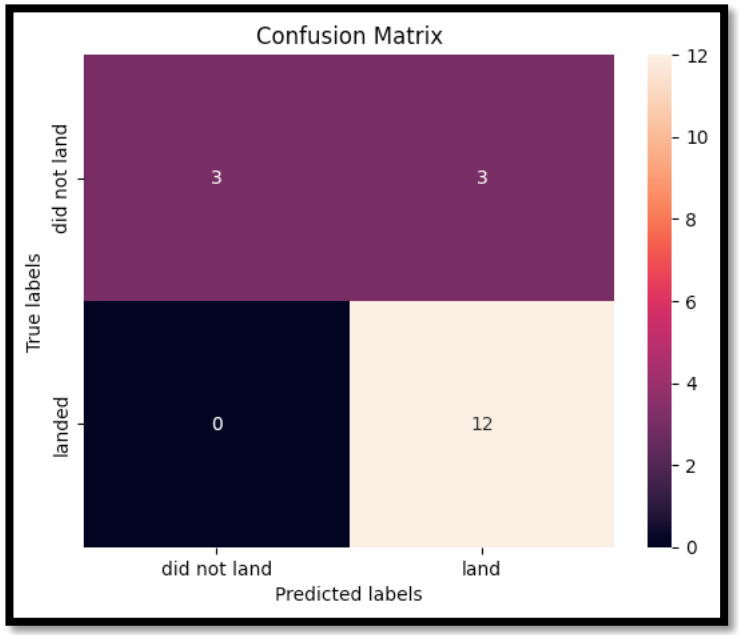
```
r2 = tree_cv.score(X_test, Y_test)
r2
```

[24]

```
0.8333333333333334
```

Creating a KNN Model

✓ This model is suitable, it has a good R2 score.




```
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()
```

26]

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

27]

GridSearchCV
estimator: KNeighborsClassifier
KNeighborsClassifier

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

28]

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

TASK 11

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

```
r2 = knn_cv.score(X_test, Y_test)
r2
```

29]

```
0.8333333333333334
```

Conclusions

We can conclude that:

- ✓The larger the flight amount at a launch site, the greater the success rate at a launch site because of the gain of experience
- ✓Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- ✓KSC LC-39A had the most successful launches of any sites.
- ✓The logistic regression model classifier is the best machine learning algorithm for this task.

Thanks

Github Link :
https://github.com/AbdoNiceman/IBM_certificate/tree/95d09593cf69e74dddef61cad857824091bb83c5/Applied%20Data%20Science%20Capstone