<SpaceX Falcon9 First Stage Landing Prediction>

<Aly Mohamed Aly Nasr>

<10 Oct 2022>

IBM Developer

SKILLS NETWORK

OUTLINE

- Executive Summary
- Introduction
- Methodology
- Results
  - Visualization – Charts
  - Dashboard
- Discussion
  - Findings & Implications
- Conclusion
- Appendix

EXECUTIVE SUMMARY

- Data Collection with API & Web Scraping
- Data Wrangling
- EDA
  - EDA with SQL
  - EDA with Python/Data Visualization
  - Interactive Visual Analytics with Folium & Dash
- ML Predictive analysis
- EDA Results
- ML Results

INTRODUCTION

- Project Background & context
  - Space X 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 Space X can reuse the first stage.
  - The goal of this project is to predict if the first stage will land successfully so that we can determine the cost of a launch.
- Problems to solve
  - What are the factors that contribute to a successful landing?
  - What is the success rate of said landing?

METHODOLOGY

- Data Collection with API
  - Data collection was done using SpaceX Rest API
  - The Request was converted into a JSON file using the .json() method
  - The JSON file was then converted into a pandas Dataframe
  - The Data was then cleaned and missing values were dealt with.

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

METHODOLOGY

- Data Collection with Web Scraping
  - The Data was collected from the Falcon 9 Wikipedia page
  - The Request was parsed and the HTML tables were extracted
  - The Date was then converted into a pandas Dataframe

METHODOLOGY

- Data Wrangling
  - In this section, the outcome of the launch was categorized and encoded into a one-hot vector

METHODOLOGY

- EDA
  - In this section, the shape and properties of the Data was explored using SQL
  - Then, using Python libraries such as Matplotlib and Seaborn we observed the dependencies and correlation of the various features

METHODOLOGY

- Interactive Visual Analytics
  - Building an Interactive Map with Folium
  - All launch sites were marked on the map
  - Individual launches were classified according to outcome and color coded then added to the map

  - Building a Dashboard with Plotly Dash
  - The Dashboard include pie charts indicating the total number of launches by site and scatter graphs showing the relationship between the launch outcome and the payload mass (KG) for different booster versions

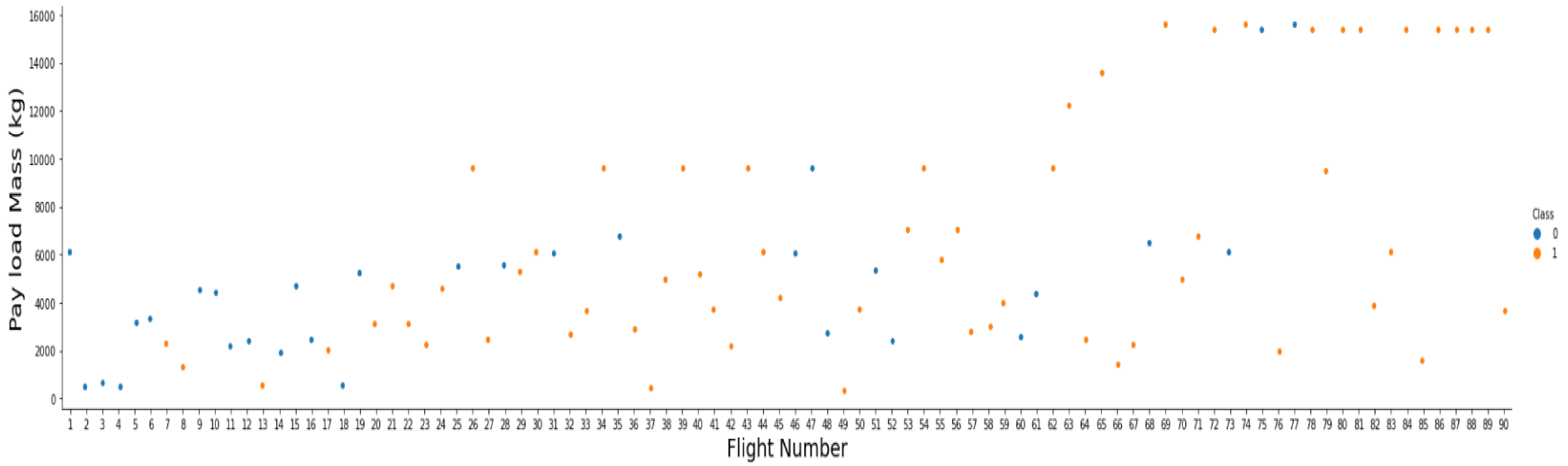
METHODOLOGY

- Predictive Analysis
  - The Data was loaded using numpy and pandas
  - The Data was then transformed, normalized and split into training and test sets
  - Different Algorithms were used and optimized by GridSearchSV
  - The Best Model was found successfully

RESULTS

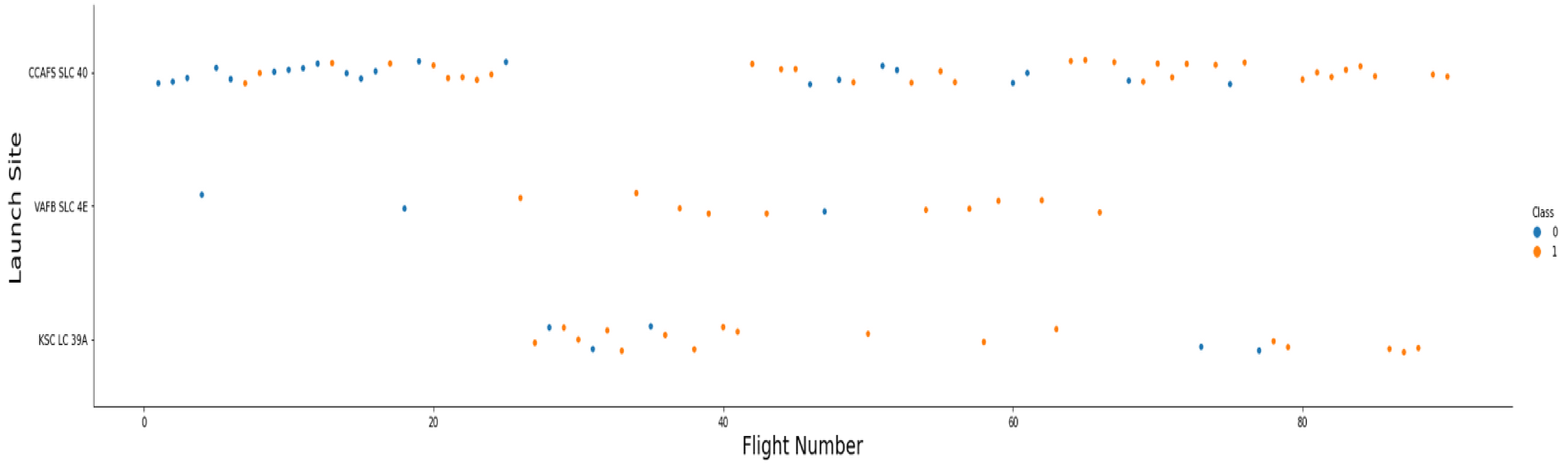
- EDA Results
  - Data Visualization
  - Using SQL
- Interactive Analytics
  - Folium Maps
  - Dashboard
- Predictive Analysis Results

EDA Using Data Visualization



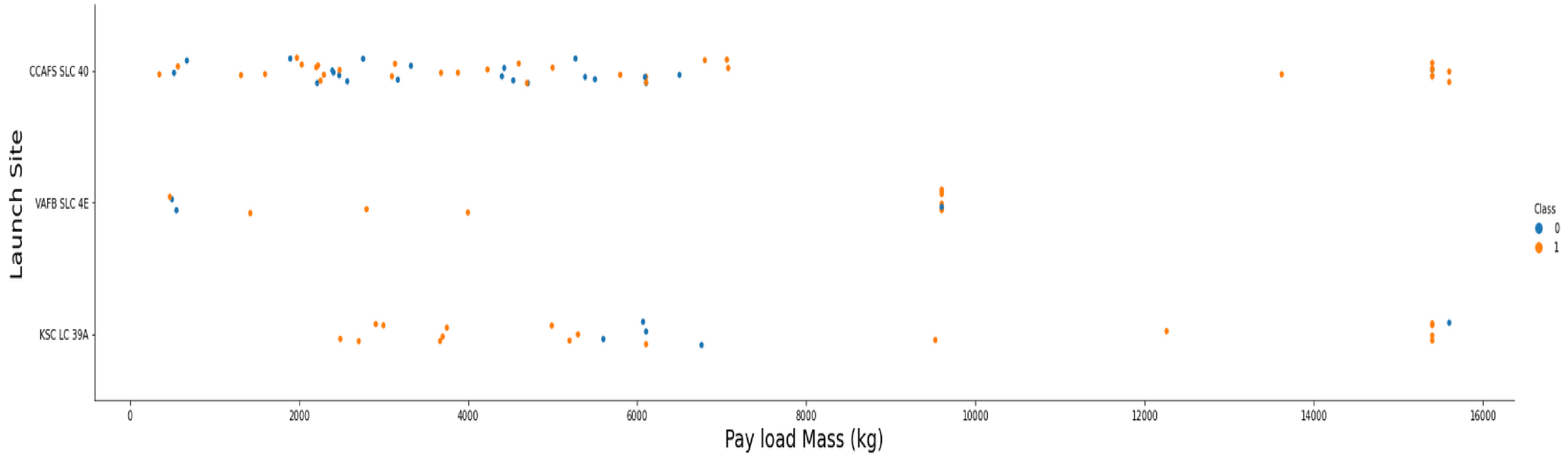
- From the Plot We see that the success rate of the landing increases in later launches.
- From this we can understand that SpaceX upgraded and optimized the landing methods and learned how to better recover the first stage after many failures
- We can see that the first 6 launches were all failures and the success rate only started increasing significantly after the 20th launch, while the Payload mass is observed to increase the success rate

EDA Using Data Visualization



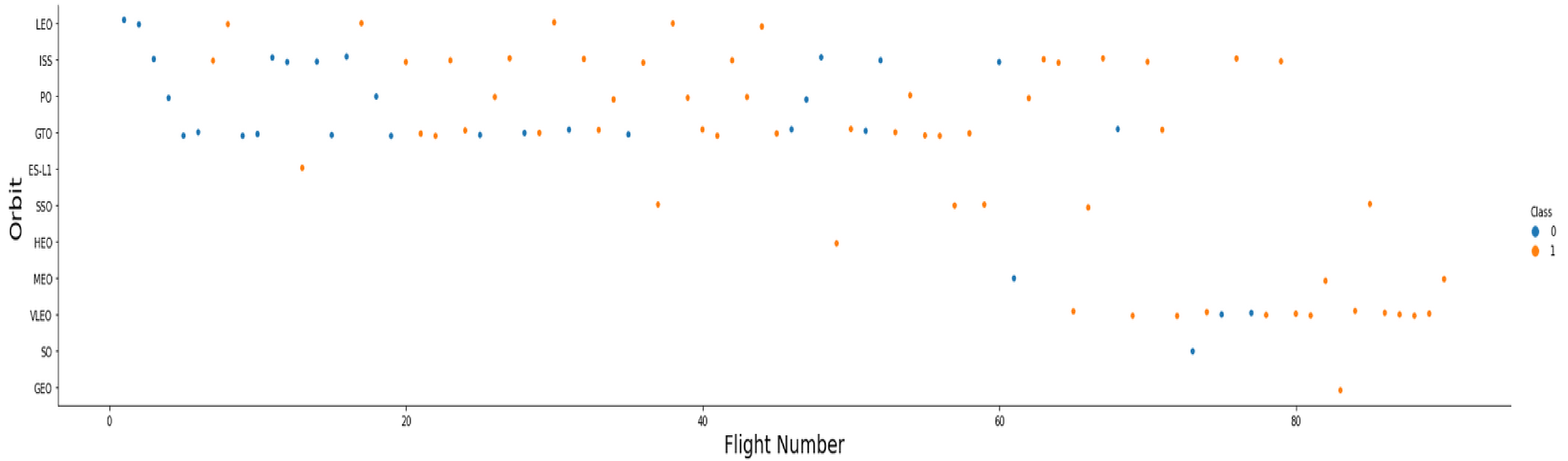
- From the Plot We see that the First and last launch sites are used more than the middle one

EDA Using Data Visualization



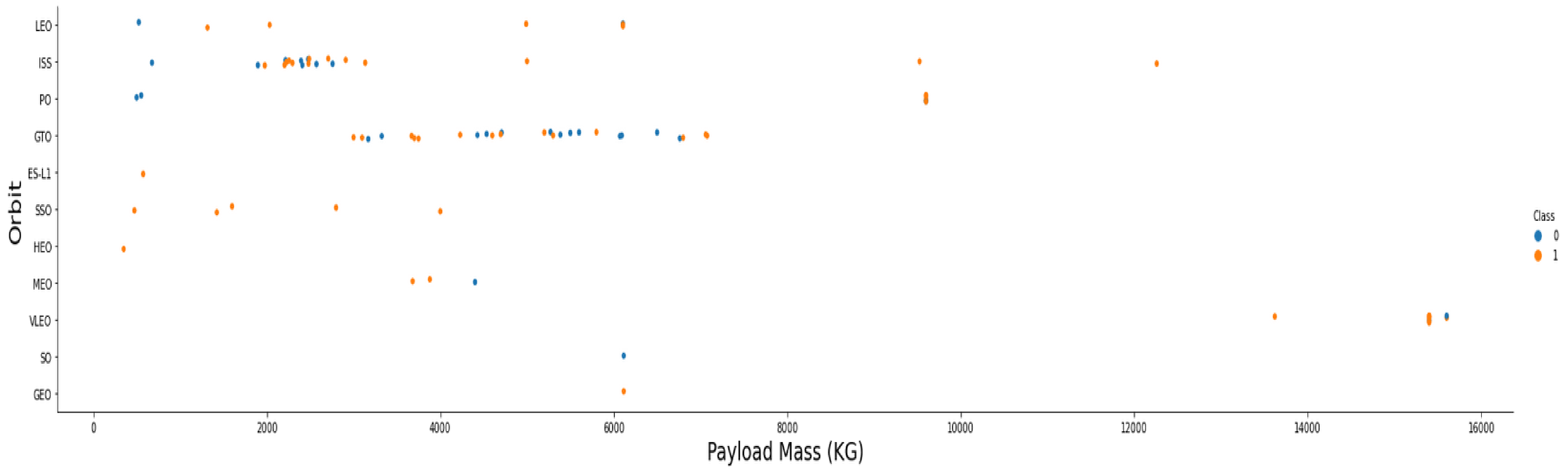
- From the Plot We see that the VAFB SLC 4E launch site is only used withing a specific payload mass range and all launched do not reach the 10,000 KG mark

EDA Using Data Visualization



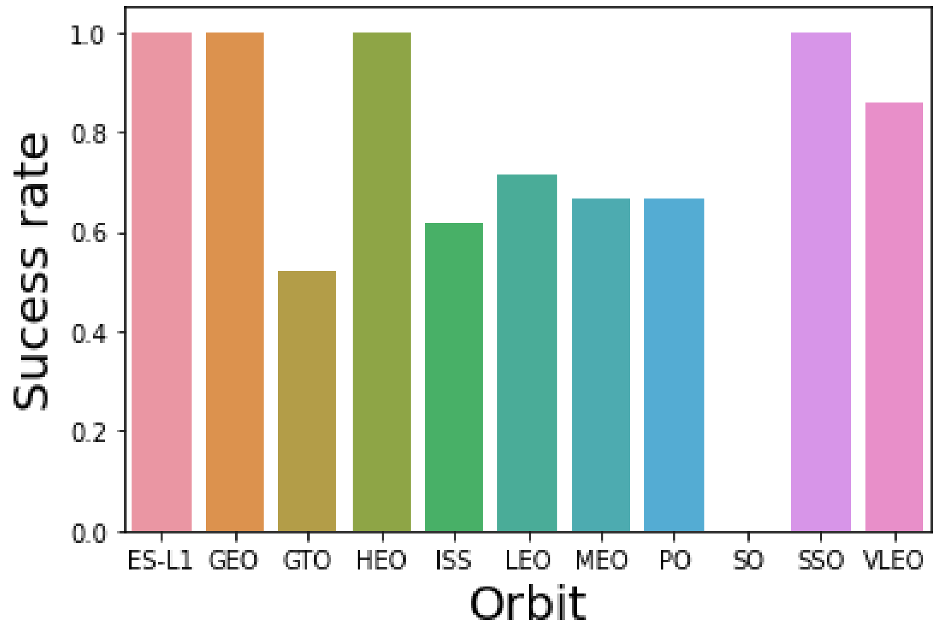
- From the Plot We see that some orbits such as the LEO orbits appears correlated to the flight number while no such relation can be observed from GTO orbit
- Some orbits have a very small number of example such as GEO, SO and ES-L1 and no insight can be observed from them.

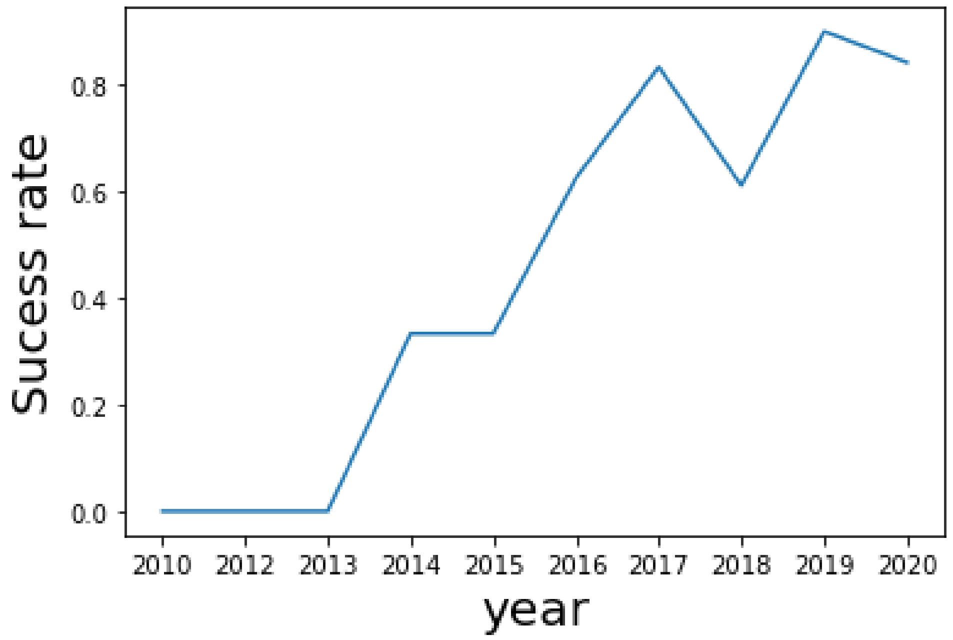
EDA Using Data Visualization



- From this Plot it can be observed that some orbits such as PO, LEO and ISS are more successful the heavier the payload is while no such relation can be observed from GTO orbit

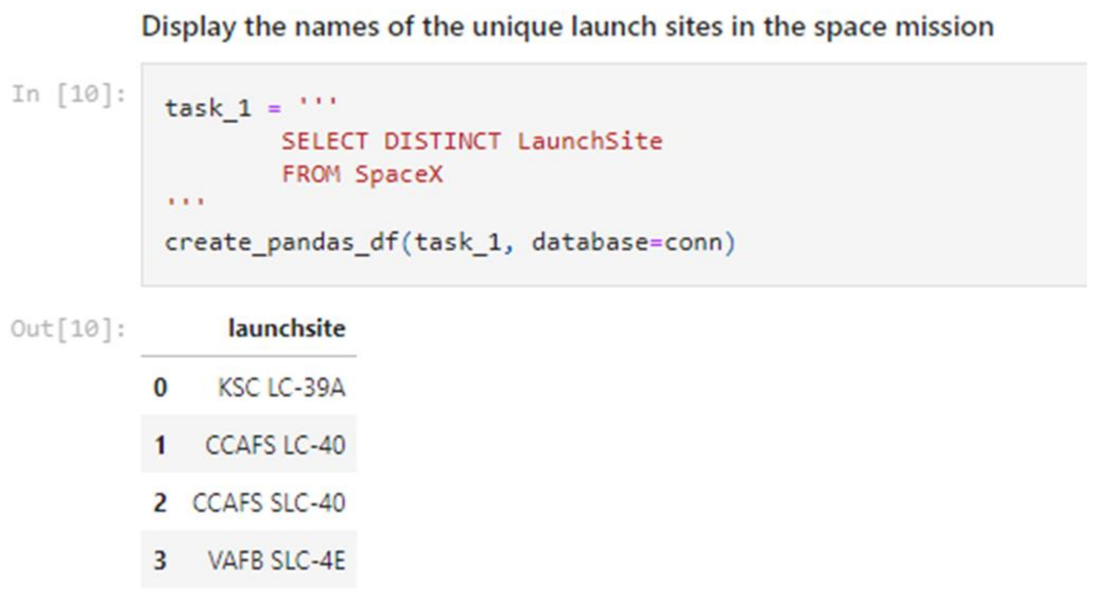
EDA Using Data Visualization





EDA Using SQL

The unique launch sites from the SpaceX data



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

```
In [10]:
task_1 = '''
        SELECT DISTINCT LaunchSite
        FROM SpaceX
'''
create_pandas_df(task_1, database=conn)
```

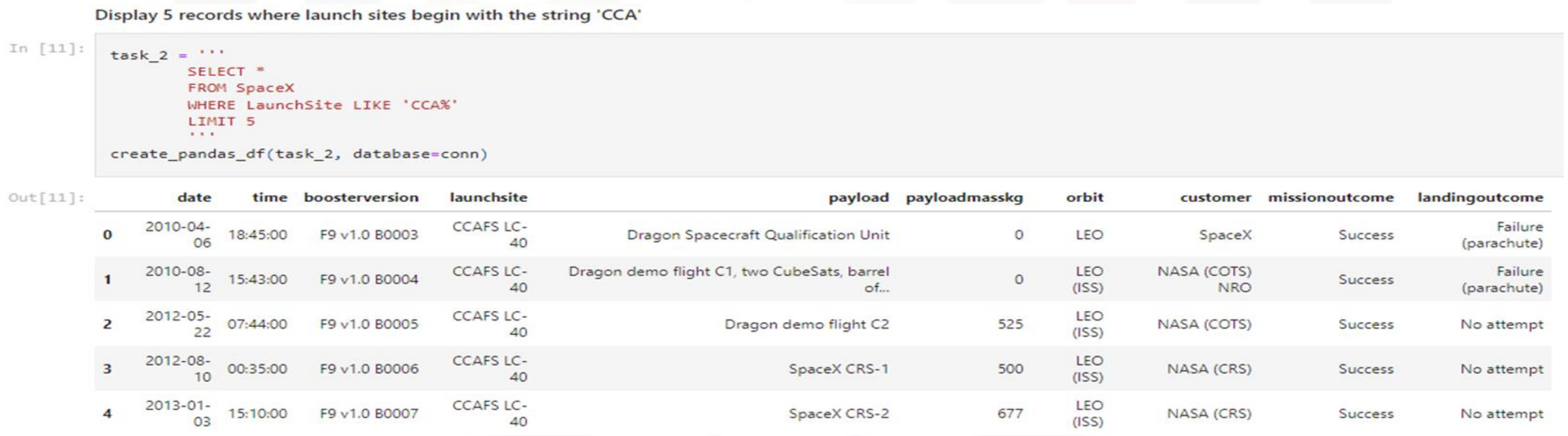
```
Out[10]:
     launchsite
0    KSC LC-39A
1    CCAFS LC-40
2    CCAFS SLC-40
3    VAFB SLC-4E
```

IBM Developer

SKILLS NETWORK

EDA Using SQL

Display 5 records where launch sites begin with `CCA`



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

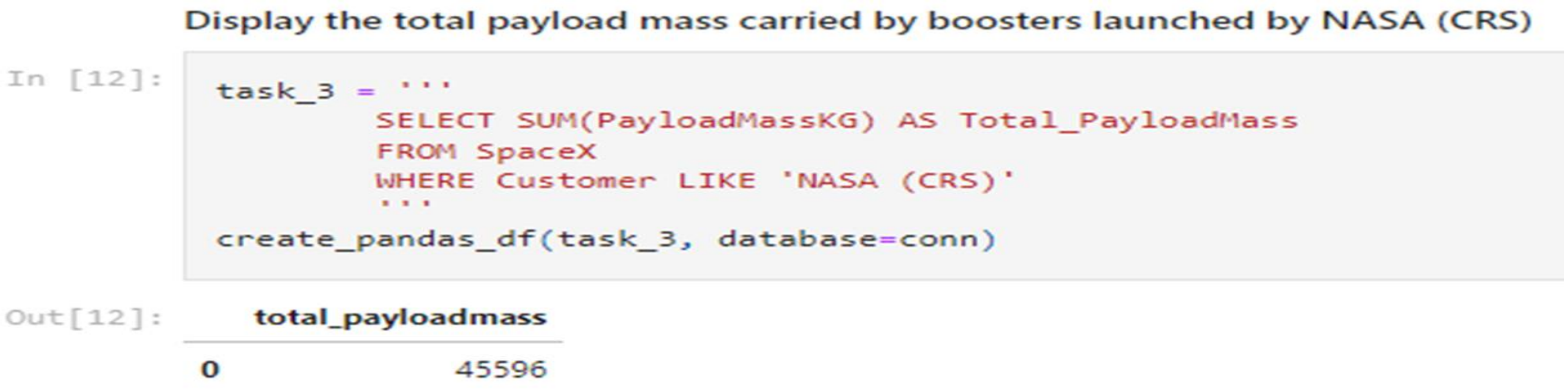
```
In [11]:
task_2 = '''
        SELECT *
        FROM SpaceX
        WHERE LaunchSite LIKE 'CCA%'
        LIMIT 5
        '''
create_pandas_df(task_2, database=conn)
```

Out[11]:

	date	time	boosterversion	launchsite	payload	payloadmasskg	orbit	customer	missionoutcome	landingoutcome
0	2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	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
3	2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

EDA Using SQL

The total payload mass carried by boosters from NASA (KG)

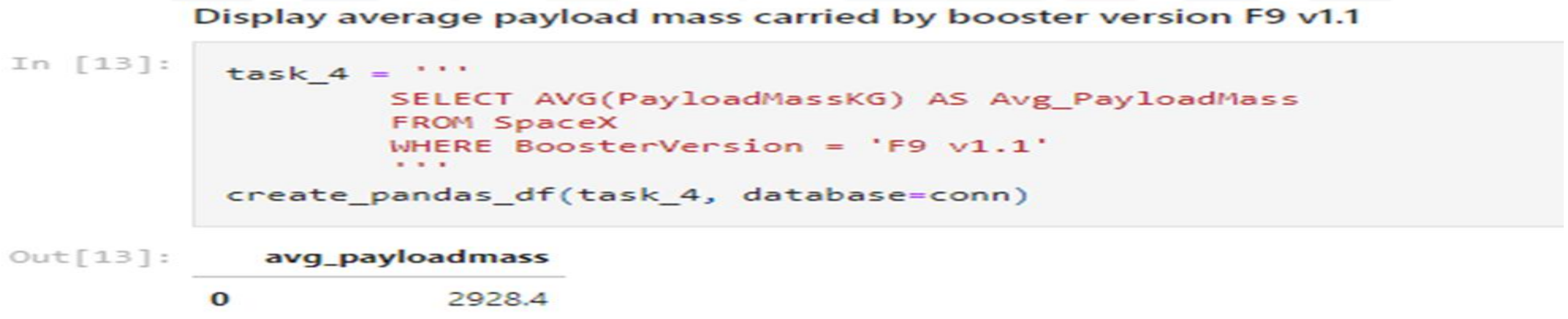


Display the total payload mass carried by boosters launched by NASA (CRS)

```
In [12]:  task_3 = '''
                  SELECT SUM(PayloadMassKG) AS Total_PayloadMass
                  FROM SpaceX
                  WHERE Customer LIKE 'NASA (CRS)'
                  '''
          create_pandas_df(task_3, database=conn)

Out[12]:     total_payloadmass
          0              45596
```

EDA Using SQL

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



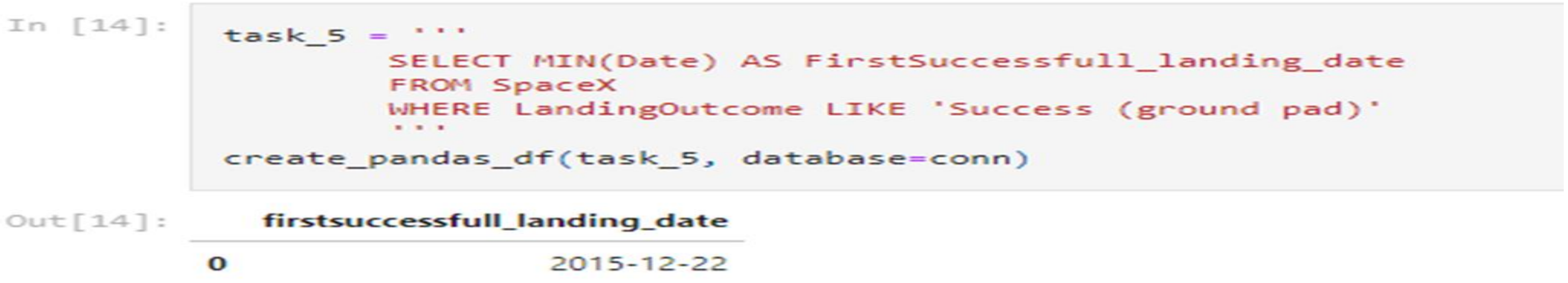
Display average payload mass carried by booster version F9 v1.1

```
In [13]:  task_4 = '''
                  SELECT AVG(PayloadMassKG) AS Avg_PayloadMass
                  FROM SpaceX
                  WHERE BoosterVersion = 'F9 v1.1'
                  '''
          create_pandas_df(task_4, database=conn)
```

```
Out[13]:      avg_payloadmass
          0            2928.4
```

EDA Using SQL

The date of the first successful landing outcome on ground pad



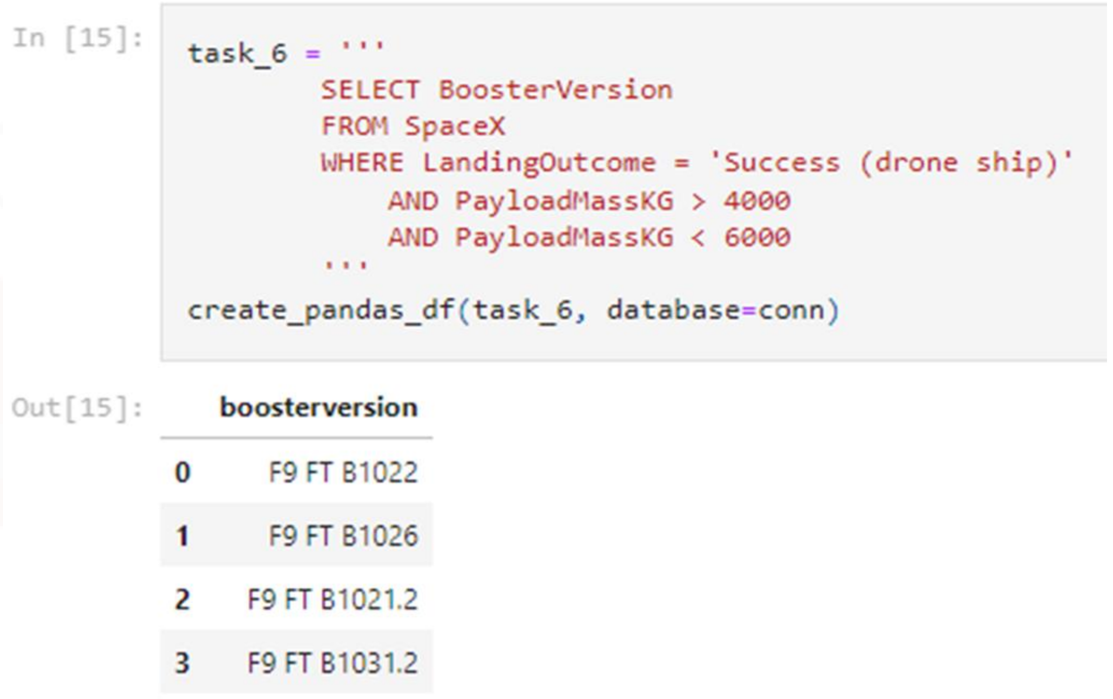
```
In [14]:  task_5 = '''
                  SELECT MIN(Date) AS FirstSuccessfull_landing_date
                  FROM SpaceX
                  WHERE LandingOutcome LIKE 'Success (ground pad)'
                  '''
          create_pandas_df(task_5, database=conn)

Out[14]:     firstsuccessfull_landing_date
          0                     2015-12-22
```

IBM Developer

SKILLS NETWORK

EDA Using SQL

Successful landing with payload mass greater than 4000 but less than 6000 based on booster version



```
In [15]:  task_6 = '''
                  SELECT BoosterVersion
                  FROM SpaceX
                  WHERE LandingOutcome = 'Success (drone ship)'
                      AND PayloadMassKG > 4000
                      AND PayloadMassKG < 6000
                  '''
          create_pandas_df(task_6, database=conn)
```

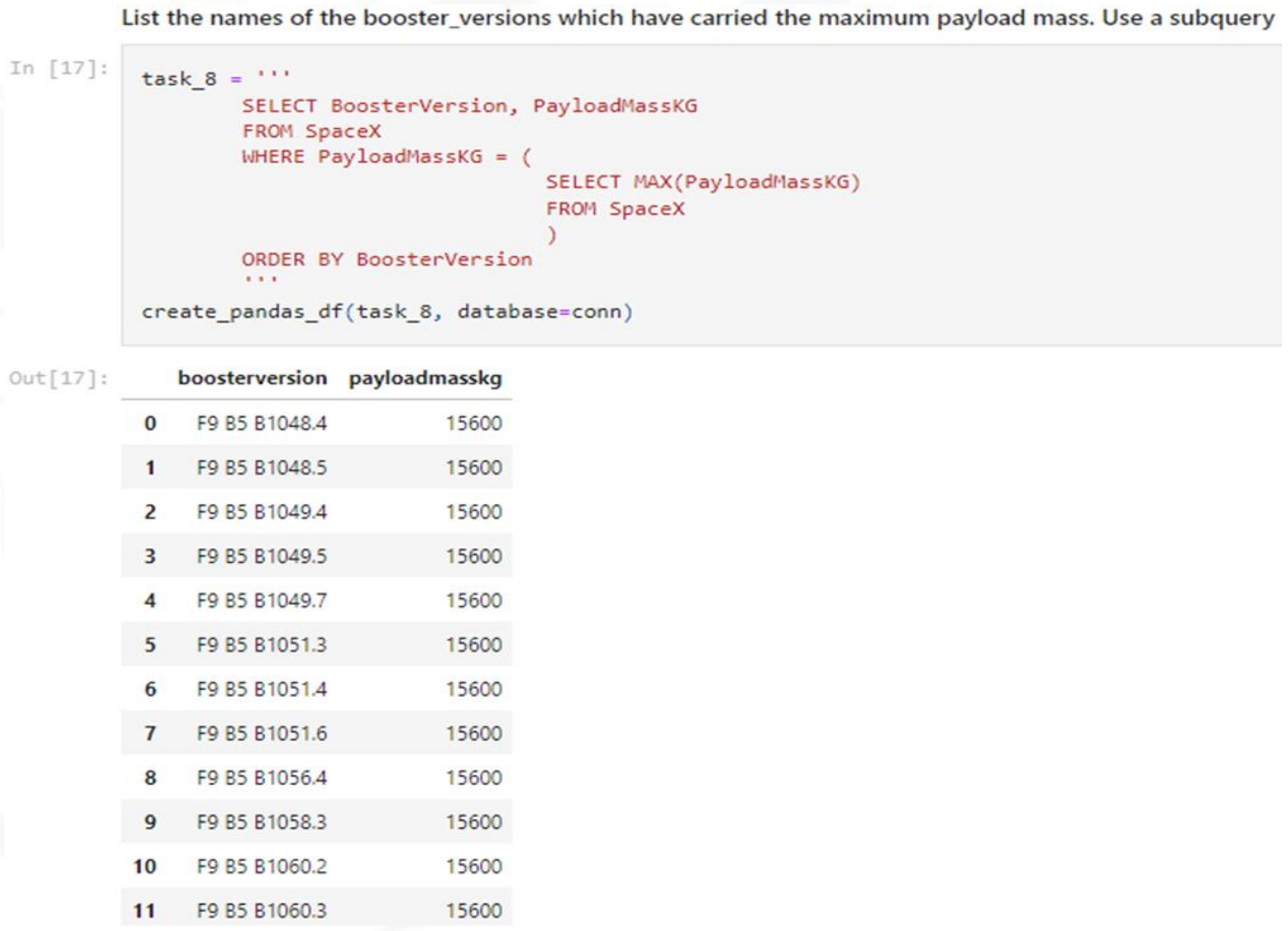
Out[15]:

	boosterversion
0	F9 FT B1022
1	F9 FT B1026
2	F9 FT B1021.2
3	F9 FT B1031.2

IBM Developer SKILLS NETWORK

EDA Using SQL

The boosters that have carried the maximum payload mass



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

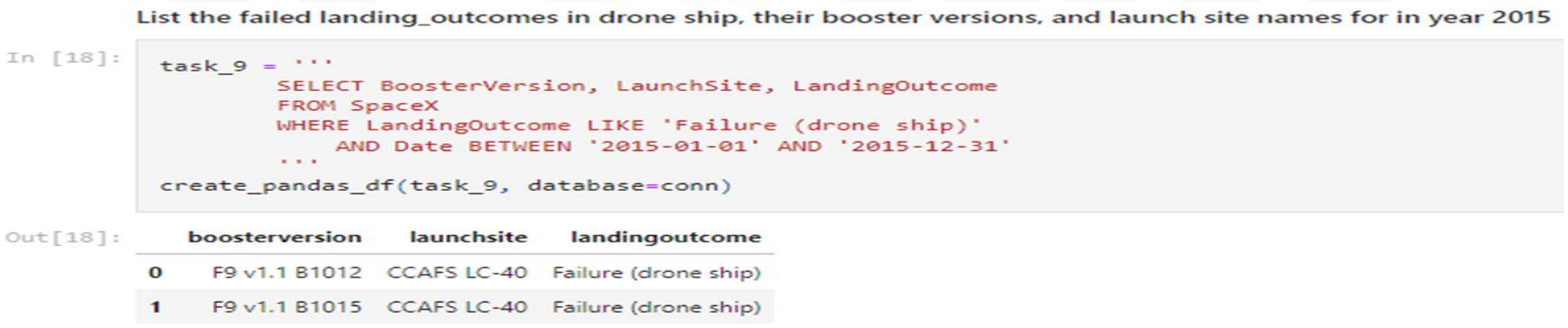
In [17]:
```
task_8 = '''
        SELECT BoosterVersion, PayloadMassKG
        FROM SpaceX
        WHERE PayloadMassKG = (
                                SELECT MAX(PayloadMassKG)
                                FROM SpaceX
                                )
        ORDER BY BoosterVersion
        '''
create_pandas_df(task_8, database=conn)
```

Out[17]:

	boosterversion	payloadmasskg
0	F9 B5 B1048.4	15600
1	F9 B5 B1048.5	15600
2	F9 B5 B1049.4	15600
3	F9 B5 B1049.5	15600
4	F9 B5 B1049.7	15600
5	F9 B5 B1051.3	15600
6	F9 B5 B1051.4	15600
7	F9 B5 B1051.6	15600
8	F9 B5 B1056.4	15600
9	F9 B5 B1058.3	15600
10	F9 B5 B1060.2	15600
11	F9 B5 B1060.3	15600

EDA Using SQL

Failed landing outcomes in drone ship, their booster version and launch site names for year 2015



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

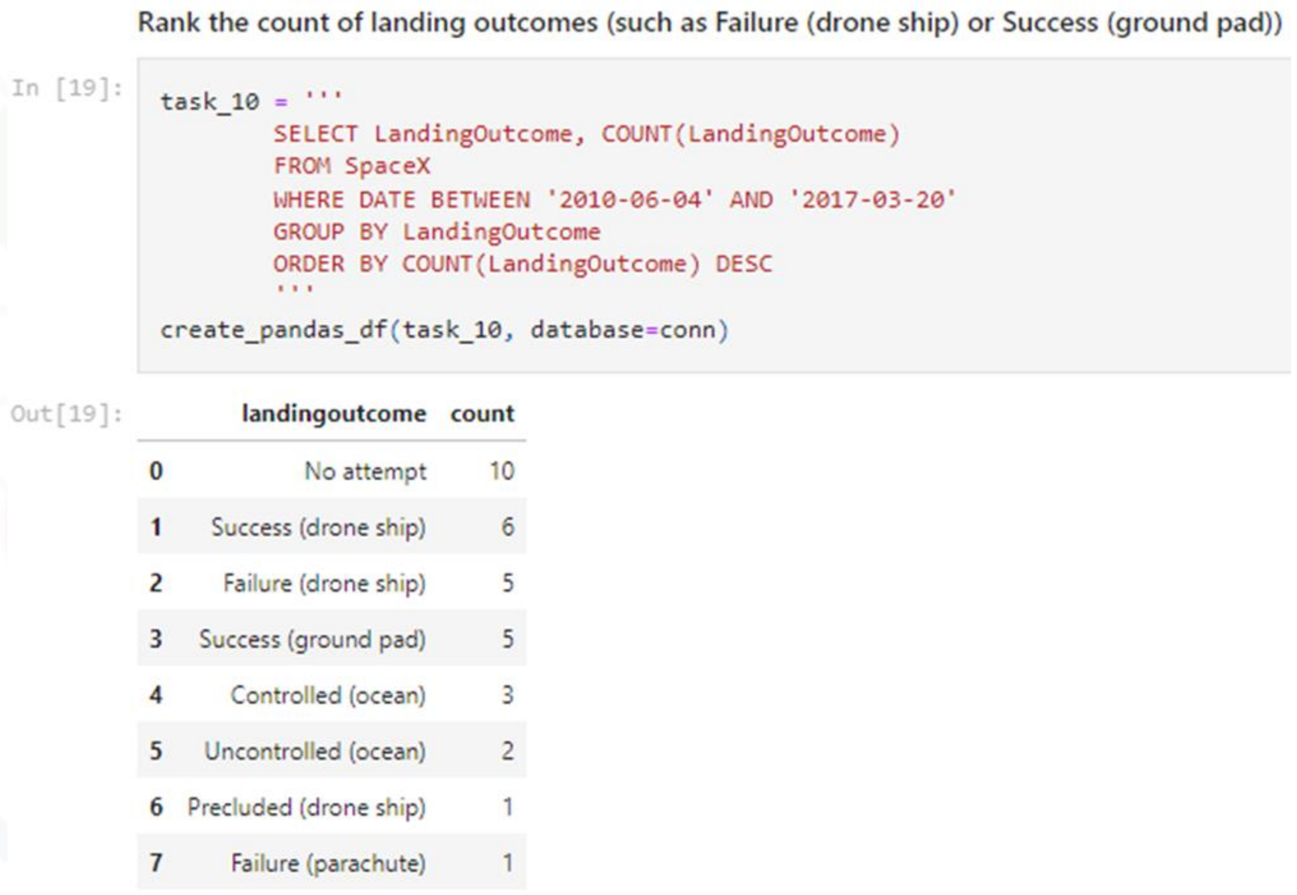
```
In [18]:  task_9 = '''
                  SELECT BoosterVersion, LaunchSite, LandingOutcome
                  FROM SpaceX
                  WHERE LandingOutcome LIKE 'Failure (drone ship)'
                      AND Date BETWEEN '2015-01-01' AND '2015-12-31'
                  '''
          create_pandas_df(task_9, database=conn)
```

Out[18]:

	boosterversion	launchsite	landingoutcome
0	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
1	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

EDA Using SQL

The landing outcomes count between 2010-06-04 and 2010-03-20

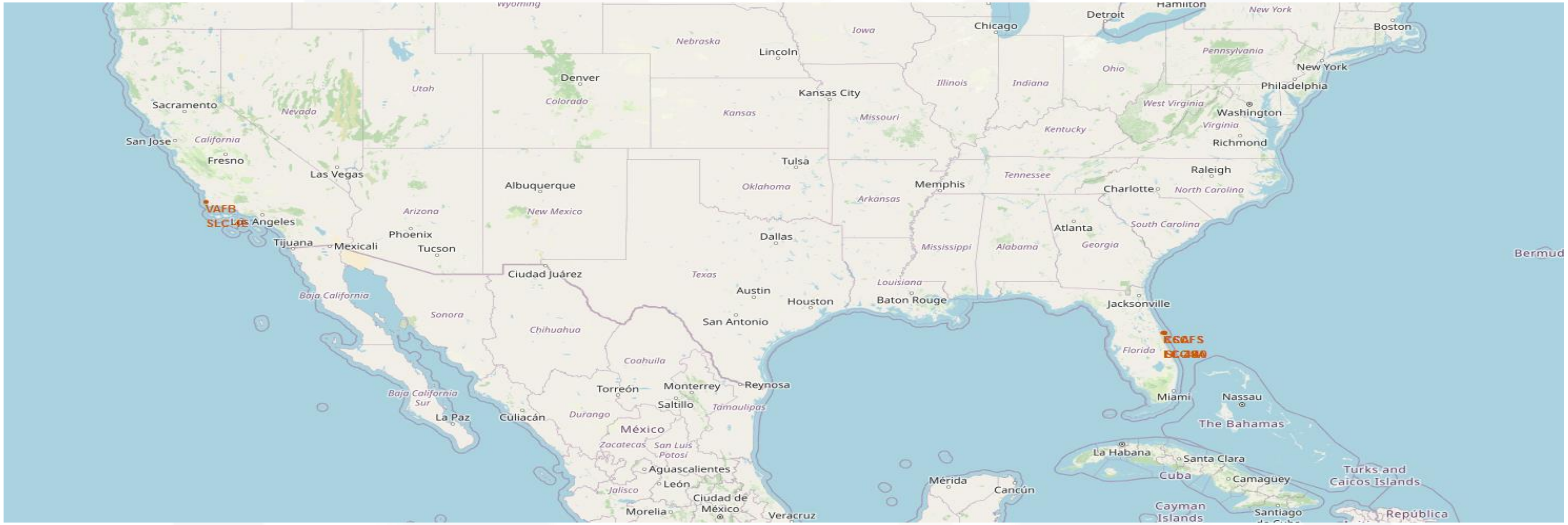


Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad))

In [19]:
```
task_10 = '''
        SELECT LandingOutcome, COUNT(LandingOutcome)
        FROM SpaceX
        WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
        GROUP BY LandingOutcome
        ORDER BY COUNT(LandingOutcome) DESC
        '''
create_pandas_df(task_10, database=conn)
```

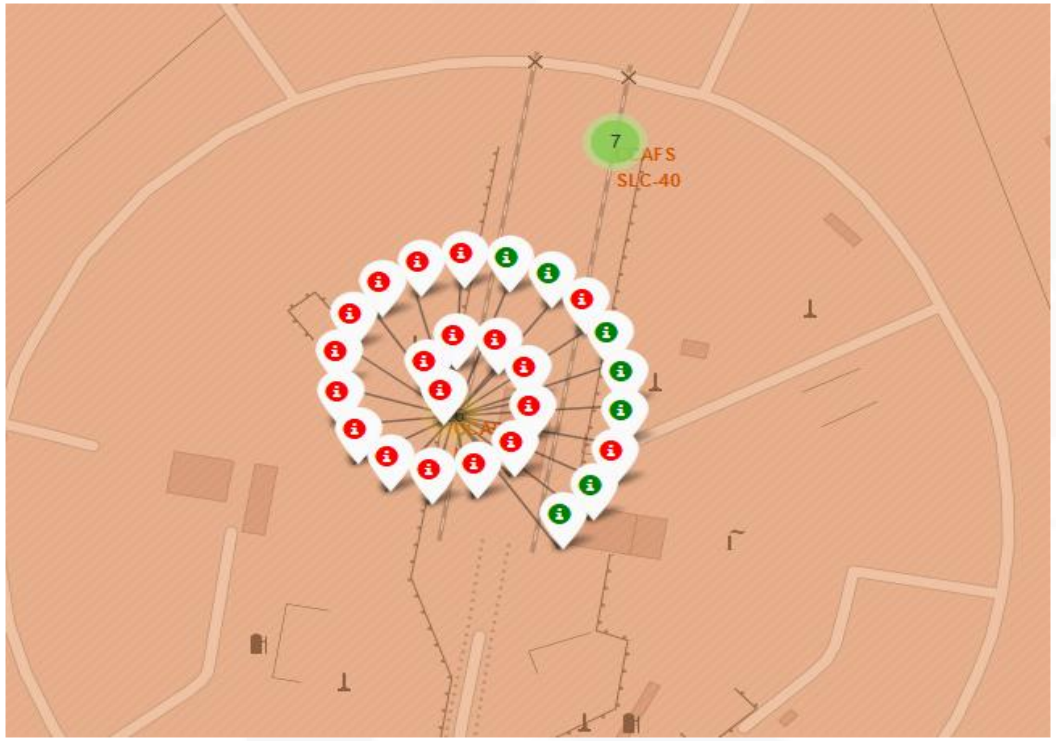
Out[19]:
```
         landingoutcome  count
0            No attempt     10
1  Success (drone ship)      6
2  Failure (drone ship)      5
3  Success (ground pad)      5
4     Controlled (ocean)      3
5   Uncontrolled (ocean)      2
6  Precluded (drone ship)     1
7     Failure (parachute)     1
```

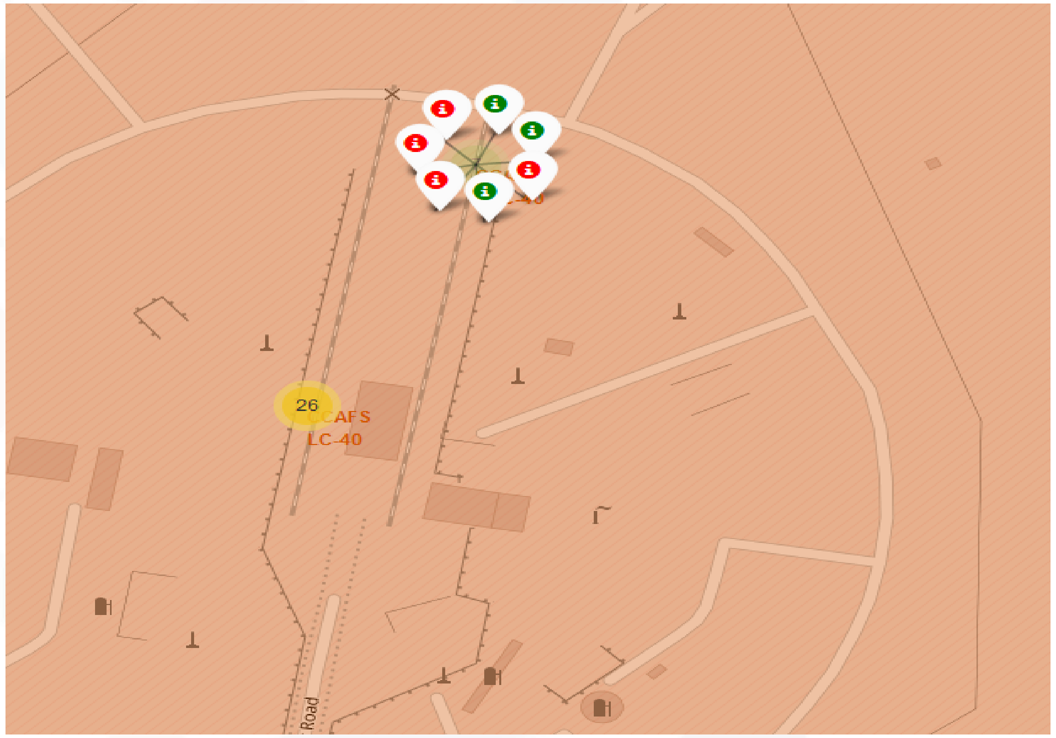
Interactive Analytics Using Folium Maps



The SpaceX launch sites along the coasts of the USA in Florida and California

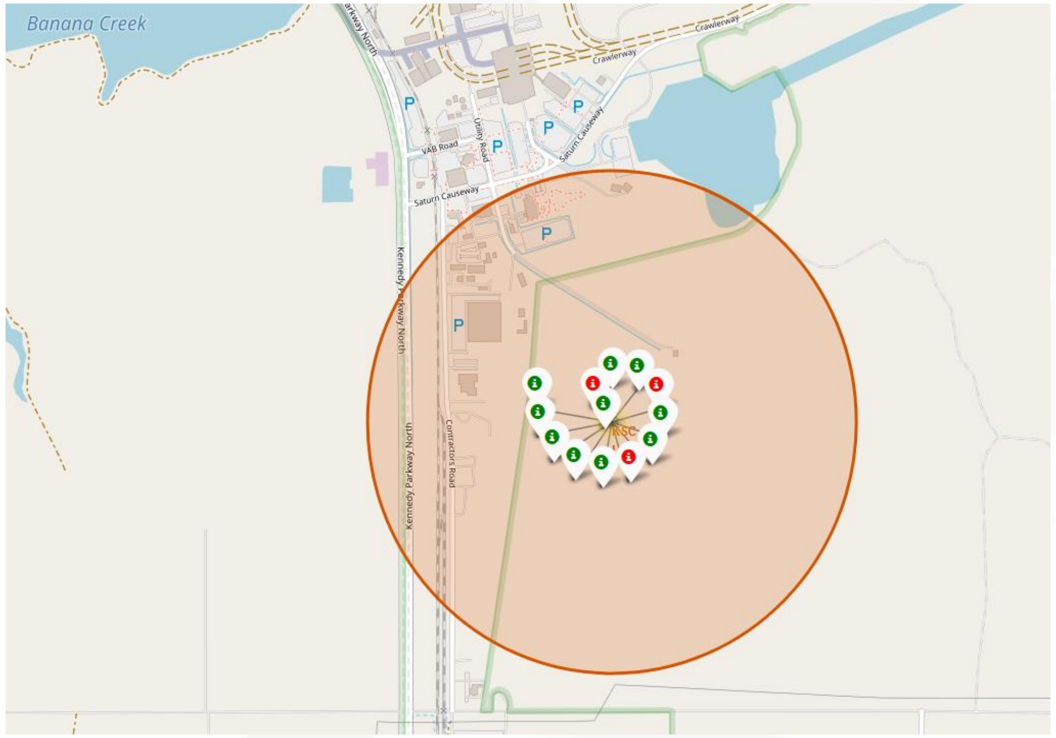
Interactive Analytics Using Folium Maps

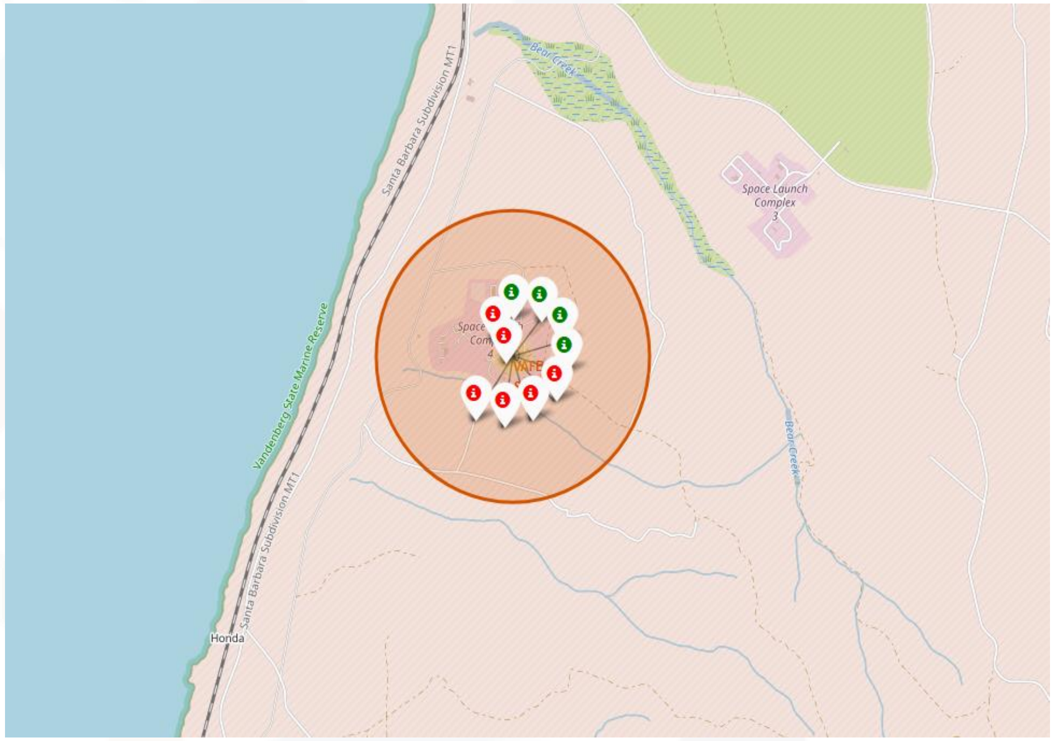




- The SpaceX launch sites CCAFS LC-40 and CCAFS SLC-40 on map
- The green markers represent successful launches

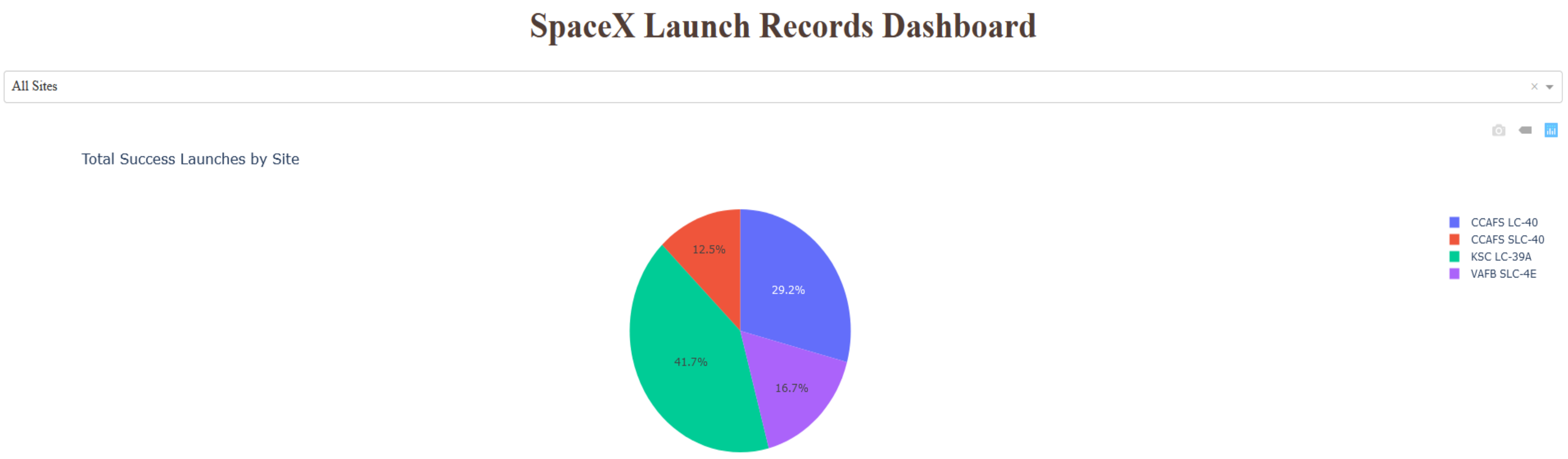
Interactive Analytics Using Folium Maps




Bear Creek
Santa Barbara Subdivision MT1
Space Launch Complex 3
Vandenberg State Marine Reserve
Honda
VAFB

- The SpaceX launch sites KSC LC-39A and VAFB SLC-4E on map
- The green markers represent successful launches

Interactive Analytics Using Dashboard

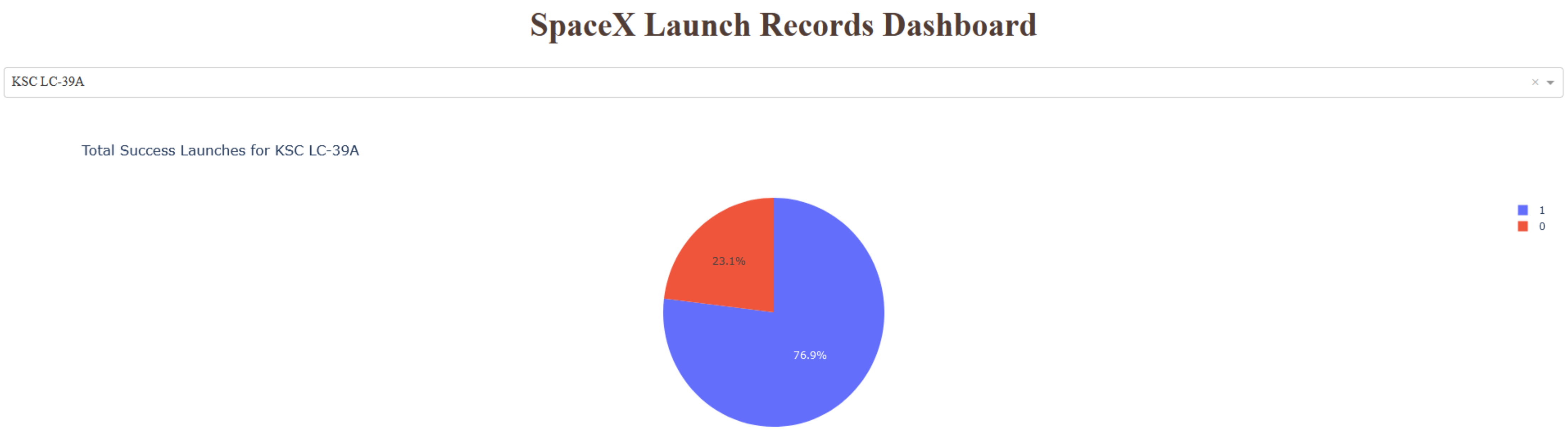

SpaceX Launch Records Dashboard

All Sites

Total Success Launches by Site

- This pie chart indicates the success rate of launches by launch site
- It can be seen that the KSC LC-39A has the highest success rate

Interactive Analytics Using Dashboard

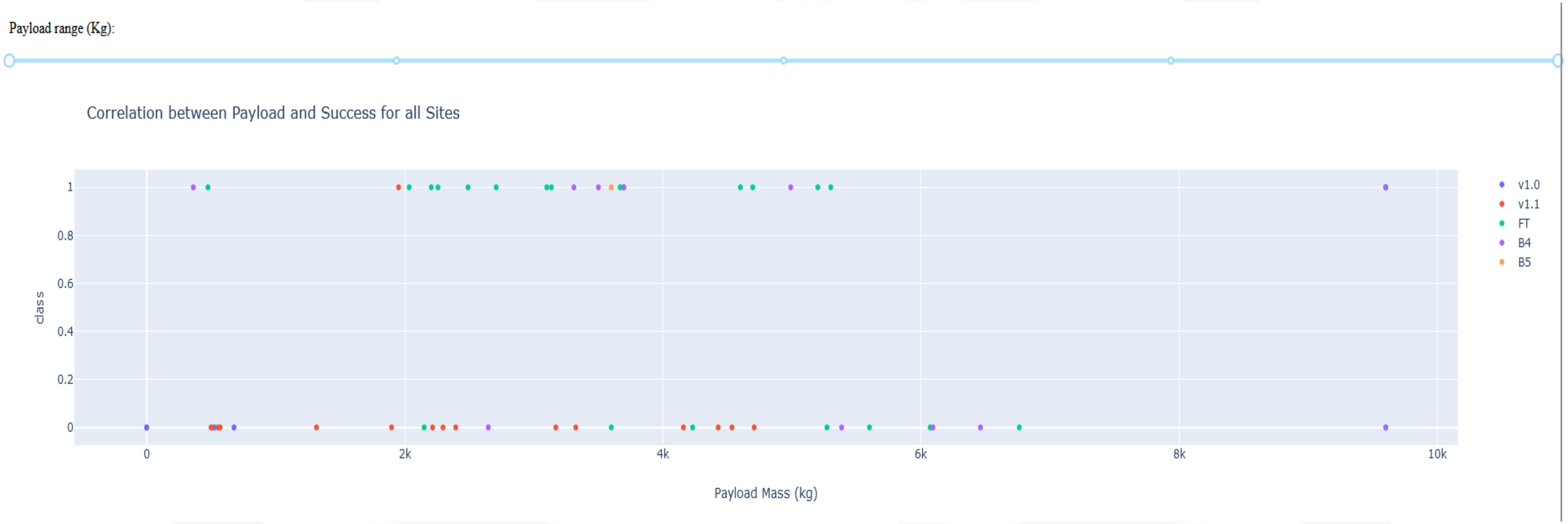

SpaceX Launch Records Dashboard

KSC LC-39A

Total Success Launches for KSC LC-39A

- This pie chart indicates the success rate of launches in KSC LC-39A

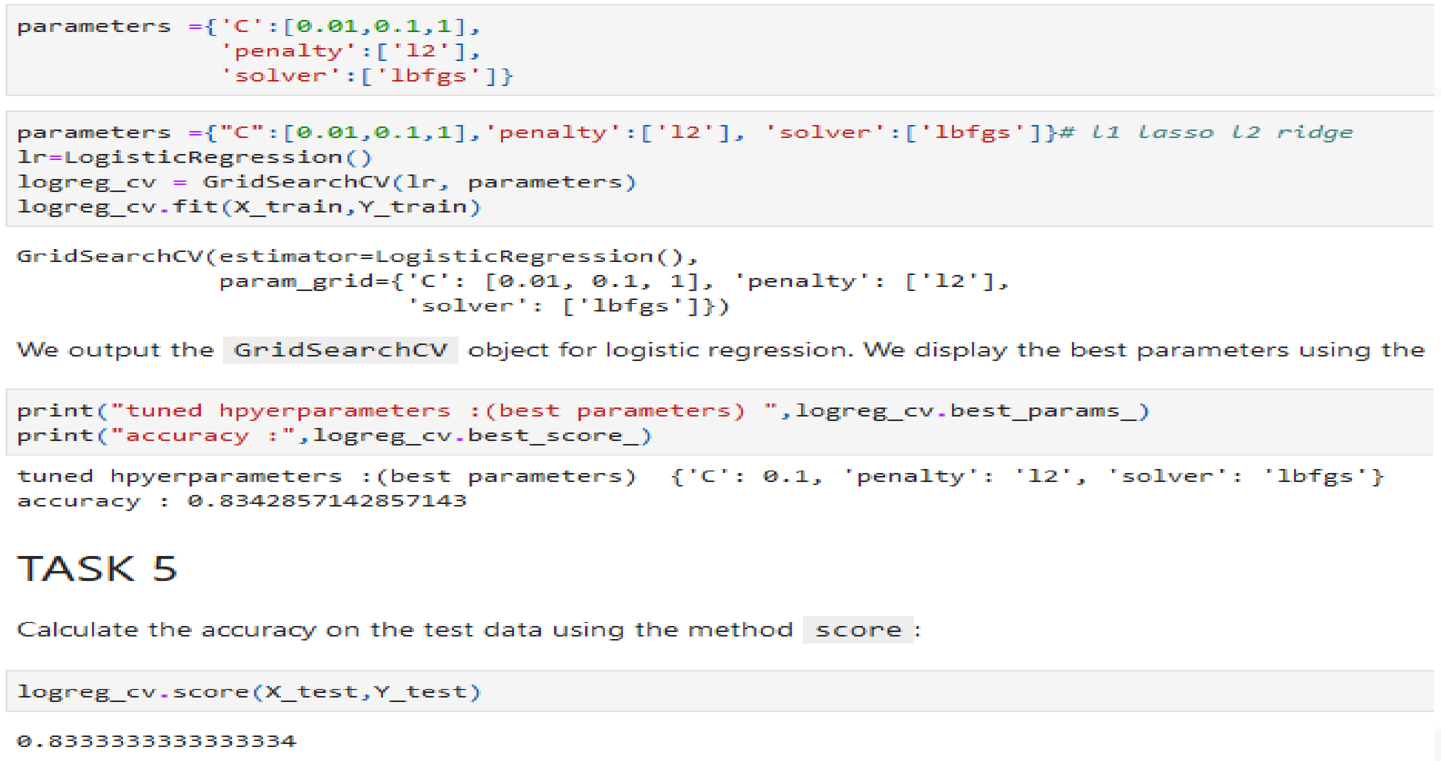
Interactive Analytics Using Dashboard



- Scatter plot of Payload vs Launch Outcome for all sites
- It can be observed that higher payload mass has lower success rate

Predictive Analysis Results

- Logistic regression Results



```
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()
logreg_cv = GridSearchCV(lr, parameters)
logreg_cv.fit(X_train,Y_train)
```

```
GridSearchCV(estimator=LogisticRegression(),
             param_grid={'C': [0.01, 0.1, 1], 'penalty': ['l2'],
                         'solver': ['lbfgs']})
```

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

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

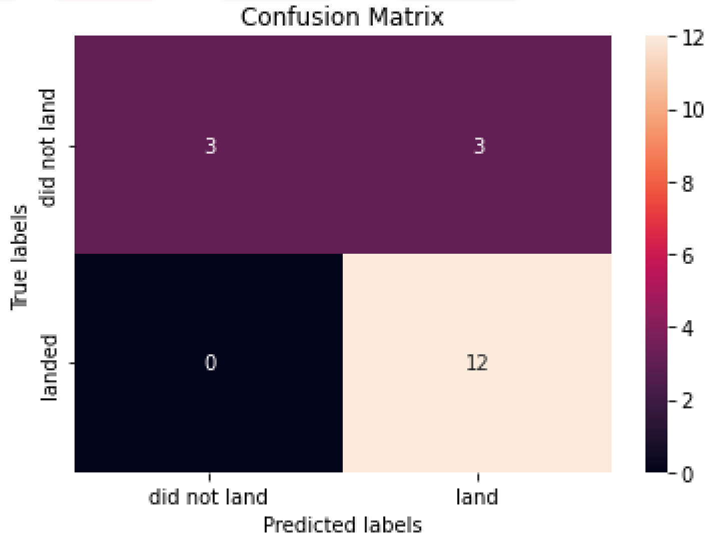
```
tuned hpyerparameters :(best parameters)  {'C': 0.1, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8342857142857143
```

TASK 5

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

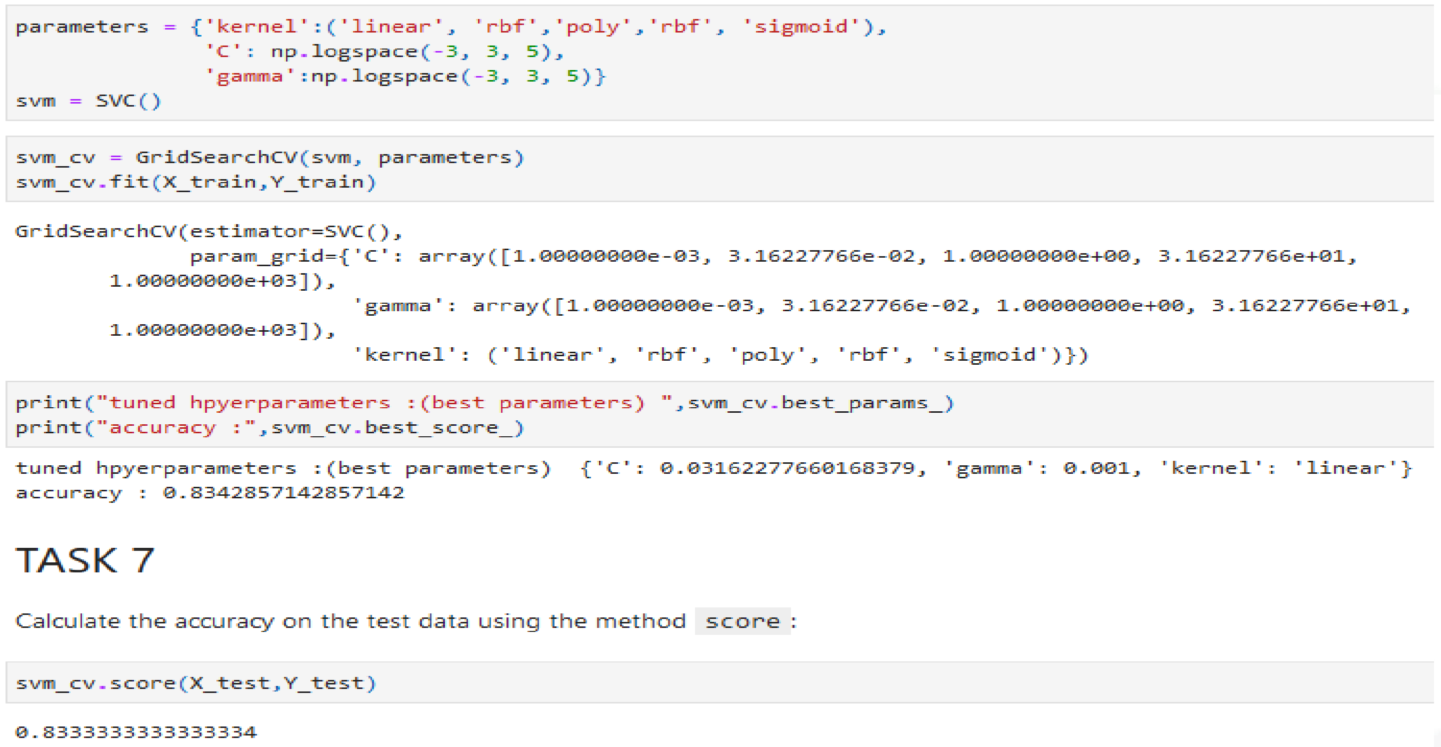
```
logreg_cv.score(X_test,Y_test)
```

```
0.8333333333333334
```



Predictive Analysis Results

- SVM Results



```
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)
svm_cv.fit(X_train,Y_train)
```

```
GridSearchCV(estimator=SVC(),
             param_grid={'C': array([1.00000000e-03, 3.16227766e-02, 1.00000000e+00, 3.16227766e+01,
       1.00000000e+03]),
                         'gamma': array([1.00000000e-03, 3.16227766e-02, 1.00000000e+00, 3.16227766e+01,
       1.00000000e+03]),
                         'kernel': ('linear', 'rbf', 'poly', 'rbf', 'sigmoid')})
```

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

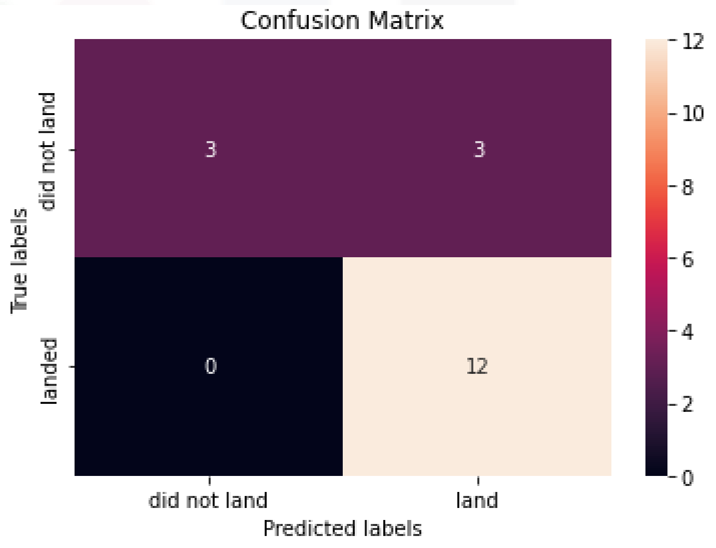
```
tuned hpyerparameters :(best parameters)  {'C': 0.03162277660168379, 'gamma': 0.001, 'kernel': 'linear'}
accuracy : 0.8342857142857142
```

TASK 7

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

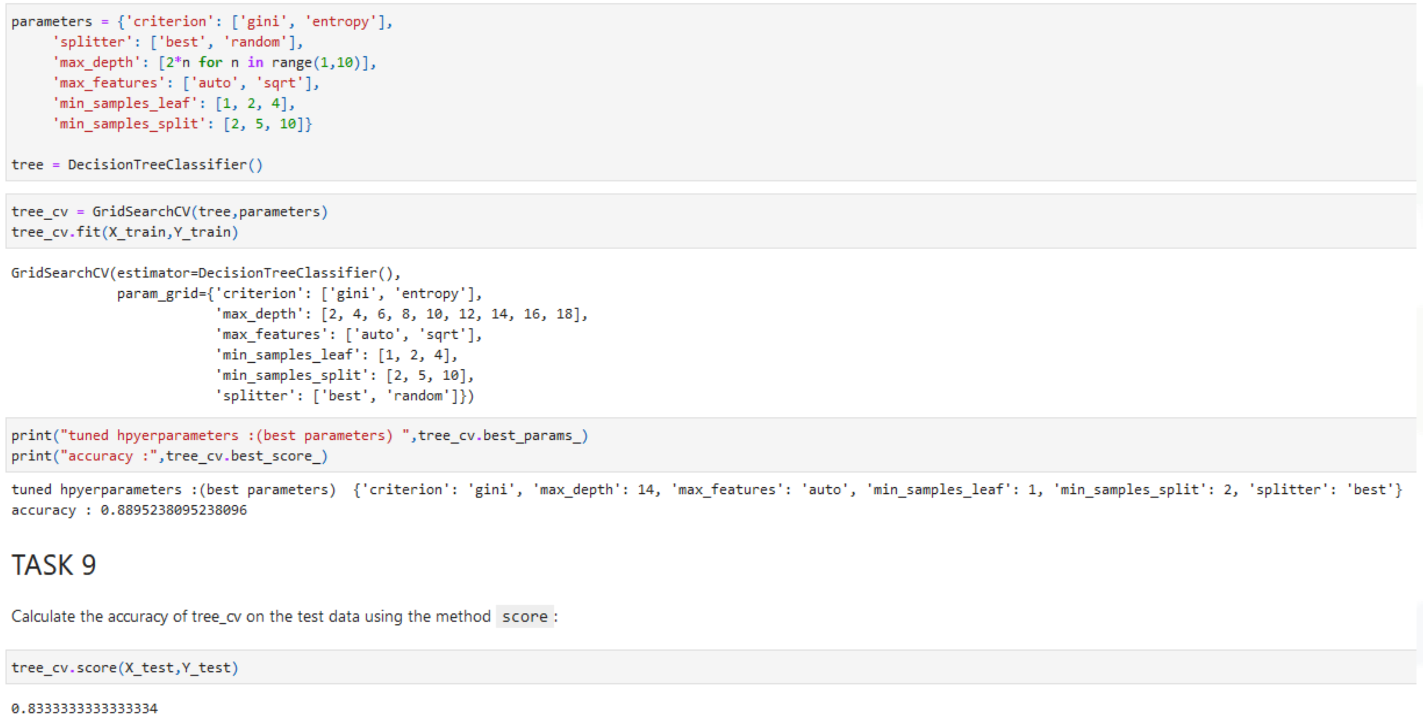
```
svm_cv.score(X_test,Y_test)
```

```
0.8333333333333334
```



Predictive Analysis Results

- Decision Tree Results



```
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()
```

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

```
GridSearchCV(estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 4, 6, 8, 10, 12, 14, 16, 18],
                         'max_features': ['auto', 'sqrt'],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [2, 5, 10],
                         'splitter': ['best', 'random']})
```

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

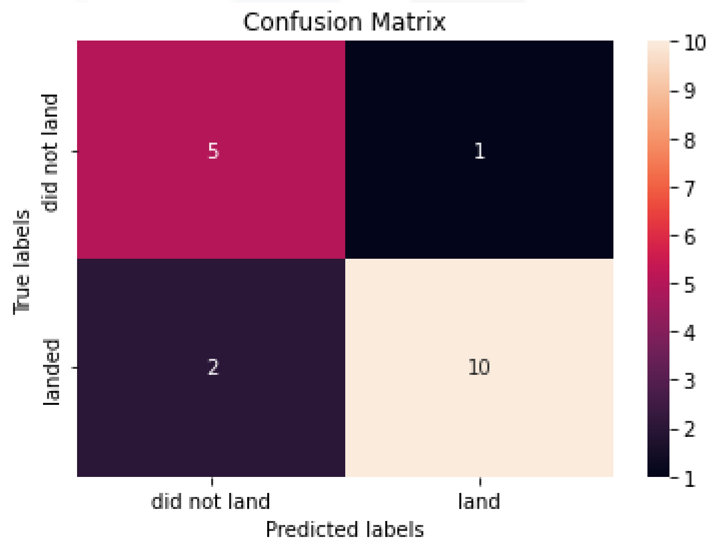
```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 14, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 2, 'splitter': 'best'}
accuracy : 0.8895238095238096
```

TASK 9

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

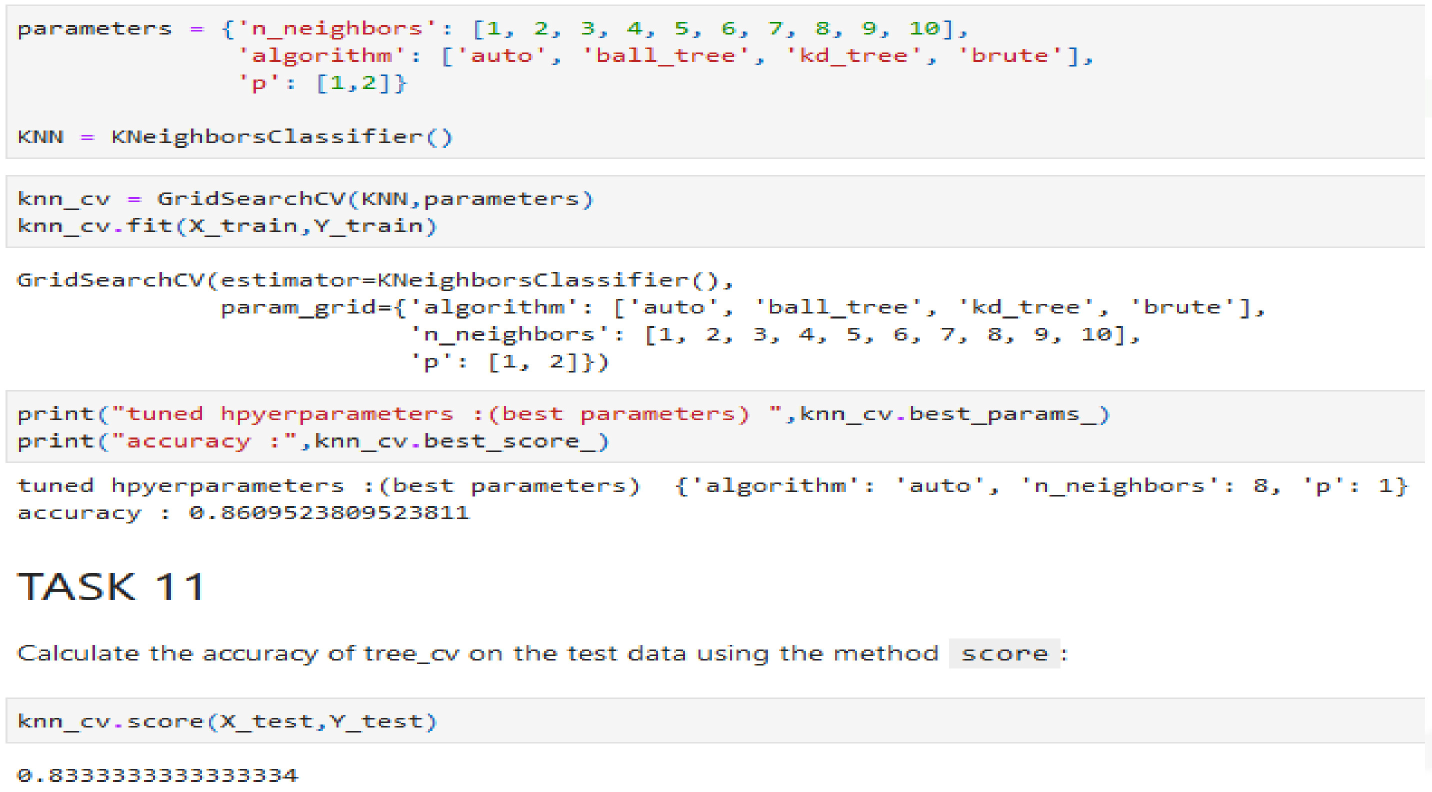
```
tree_cv.score(X_test,Y_test)
```

```
0.8333333333333334
```



Predictive Analysis Results

- KNN Results



```
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)
knn_cv.fit(X_train,Y_train)
```

```
GridSearchCV(estimator=KNeighborsClassifier(),
             param_grid={'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
                         'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                         'p': [1, 2]})
```

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

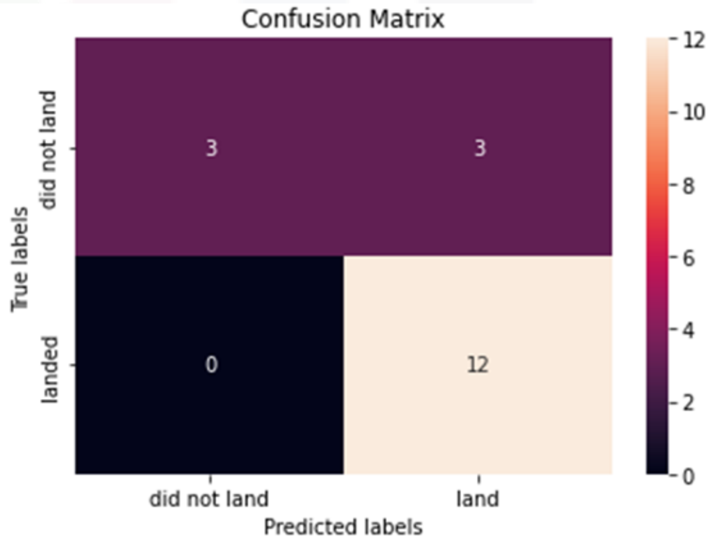
```
tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 8, 'p': 1}
accuracy : 0.8609523809523811
```

TASK 11

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

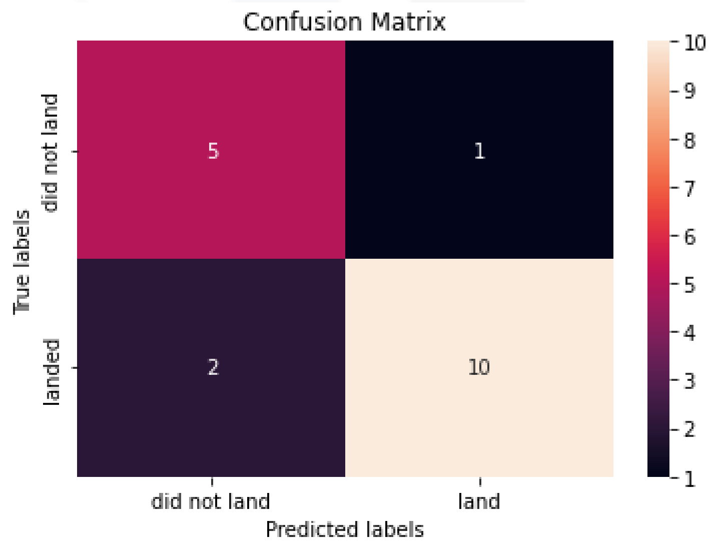
```
knn_cv.score(X_test,Y_test)
```

```
0.8333333333333334
```



Predictive Analysis Results

- The model with the best performance so far is the Decision Tree Model
- It has the score of 0.889 which is the highest so far in the training set
- The test set score is not significantly different from other models at 0.8333
- It can be observed from the confusion matrix that this model is more balanced while other models have a problem with False-positive
- The optimal parameters are



```
{'criterion': 'gini', 'max_depth': 14, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 2, 'splitter': 'best'}
```

<Thank You>

<Aly Mohamed Aly Nasr>

<10 Oct 2022>