# GO SPACE Y !

Michal Rolirad
12th February 2022

## Table of Content

## Executive Summary

- Summary of performance
  - Data Collection from SpaceX API
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Interactive Visual Analytic with Folium and Plotly Dash
  - Predictive Analysis using Classification Models
- Summary of results
  - Exploratory Data Analysis result
  - Interactive Analytic in screenshots
  - Predictive Analysis result

## Introduction

- Space X Falcon 9 vs. Others
  - Space X Falcon 9 – each rocket launch – $62 million dollars;
  - other providers – each rocket launch – $165 million dollars;
- Why?
  - Space X can recover and reuse Stage One – expensive component of the rocket, if it lands successfully.
- The problem is?
  - The successful landing is not guaranteed
- Question to find answer to:
  - What factors determine whether the rocket will land successfully

## Introduction cd.

- What to do?:
  - Train machine learning models to predict whether the landing is successful
- What for?:
  - With this knowledge Space Y will be able to copy the aspects of successful landings, save millions and conquer the market

## Methodology

- Data collection methodology:
  - Data was collected from Space X and Wikipedia using API and web scraping.
- Perform data wrangling
  - Columns were modified
  - One-hot encoding was applied to categorical features
- 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 was splited into train and test
  - Various ML classification models were tested using train data
  - Best model was tested using test data

## Data Collection

The data was collected using various methods:

- Data was collected using get request to the SpaceX API,
- Next, the response content was parsed into a pandas Data Frame,
- Then data was cleaned - checked for missing values which were converted where necessary into the mean of the column.
- In addition, web scraping from Wikipedia was performed for Falcon 9 launch records.
- The objective was to extract the launch records as HTML table, parse the table and convert it to a pandas Data Frame for future analysis.

## Data Wrangling

- So how to find out whether a landing was successful or not?
- According to our dataframe, there are 8 different outcomes, each of which can be considered a Success or Failure:
  - True ASDS → Successful landing to drone ship
  - True RTLS → Successful landing on a ground pad
  - True Ocean → Successful landing in ocean
  - None None → Failed to land
  - None ASDS → Failed to land
  - False ASDS → Failed to land on drone ship
  - False RTLS → Failed to land on ground pad
  - False Ocean → Failed to land in ocean

## Data Wrangling

- Therefore, a new column, ‘class’, was created to differentiate between successful and unsuccessful landings:
    - 1 → successful landing
    - 0 → unsuccessful landing
- From here, some EDA was performed to:
    - calculate the number of launches on each site,
    - calculate the number and occurrence of each orbit,
    - calculate the number and occurrence of mission outcome per orbit type.

## EDA with Data Visualization

- We explored the data by visualizing the relationship between:
  - flight number and payload,
  - flight number and launch Site,
  - payload and launch site,
  - success rate of each orbit type,
  - flight number and orbit type,
  - payload and orbit type,
  - success rate and yearly trend.

## EDA with SQL

- To get insight from the data, some queries were made (data was loaded into a PostgreSQL database) to find out for instance:
  - The names of unique launch sites in the space mission.
  - The total payload mass (kg) 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 with Folium

- Using the color-labeled marker clusters an interactive map was created, where one can see:
  - Where each launch site is located,
  - How many successful vs unsuccessful launches occurred at each location
  - The distance to nearest coastline, city, railway and highway

## Dashboard with Plotly Dash

- An interactive dashboard with Plotly dash was built, where one can see:
  - A pie chart showing the total launches by a certain site (which can be changed via a dropdown menu),
  - A scatter graph showing the relationship between Outcome and Payload Mass (Kg) for the different booster version
- The dashboard provides insight into the launch sites and payload masses relationships with the outcomes.

## Predictive Analysis (Classification)

- Data was loaded using numpy and pandas, standardized and split into train and test portions.
- Logistic Regression, Support Vector Machine, Decision Tree and K Nearest Neighbours models were built and tuned with different hyper-parameters  using GridSearchCV and training data
- The models were then tested with best parameters and testing data to compute accuracy score and plot confusion matrix.
- The best classification model was found.

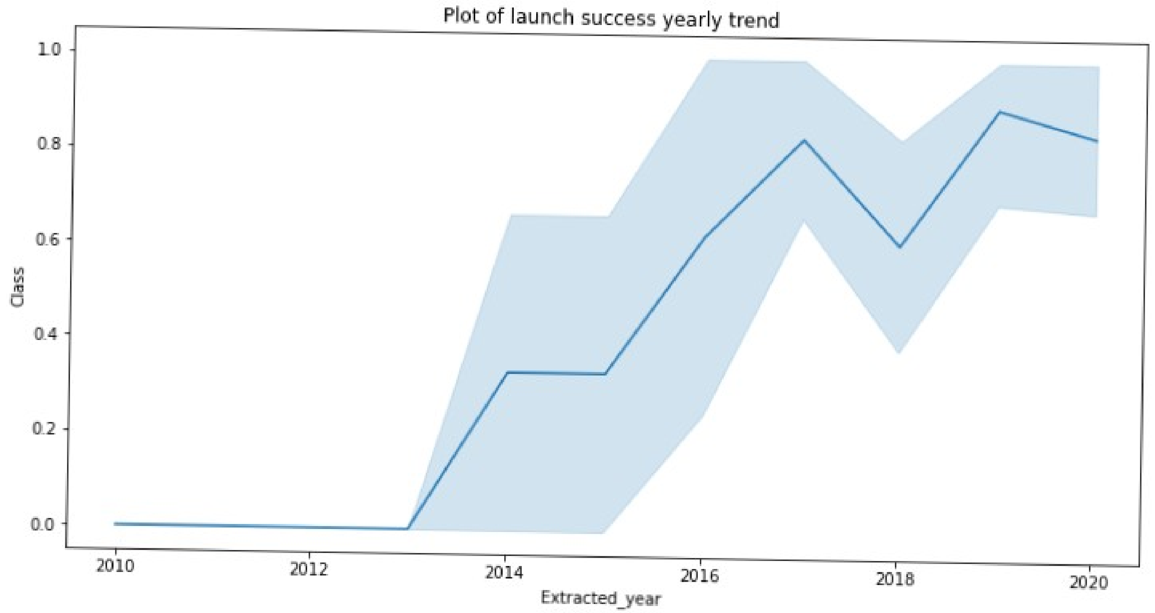
## Results

EDA Results:

- SpaceX has gotten better at launching rockets overtime. Launches are most successful when launched in 2017 or later
- Each launch site is somehow successful, however site KSC LC-39A appears to be ideal with a success rate of over 75%
- Each model performed about equally, correctly predicting a recovery outcome at a rate of 83.33%

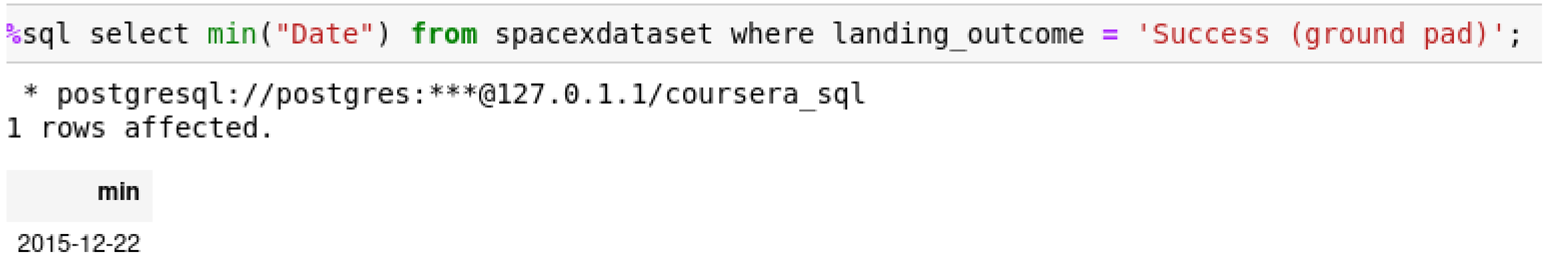
# Launch Success Yearly Trend

- we can observe that success rate kept on increasing since 2013 till 2020.



## First Successful Ground Landing Date

- We observed that the date of the first successful landing outcome on ground pad was 22nd December 2015

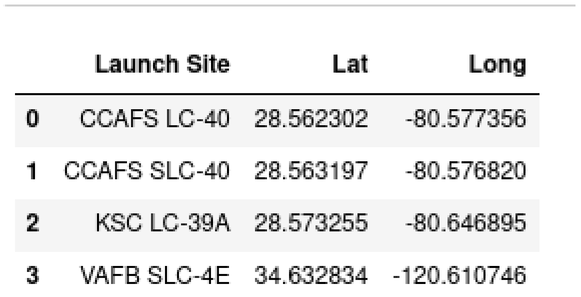


```
%sql select min("Date") from spacexdataset where landing_outcome = 'Success (ground pad)';

 * postgresql://postgres:***@127.0.1.1/coursera_sql
1 rows affected.
```

| min |
|---|
| 2015-12-22 |

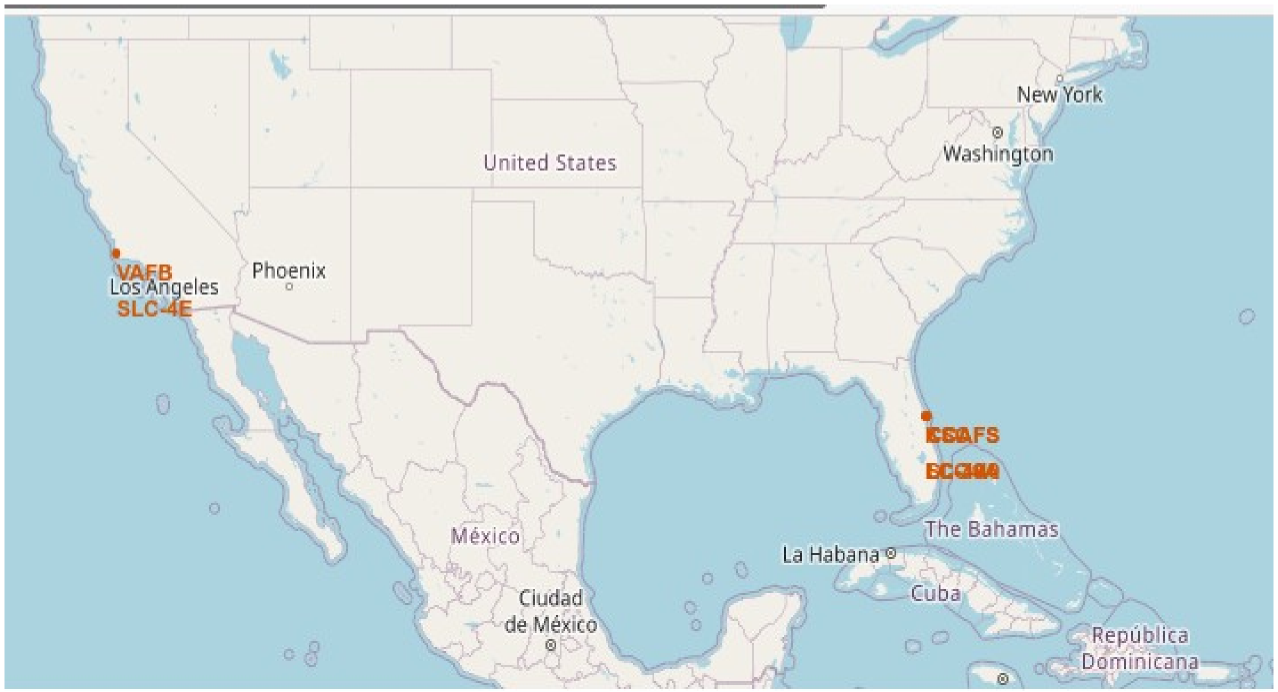
## All Launch Site Names

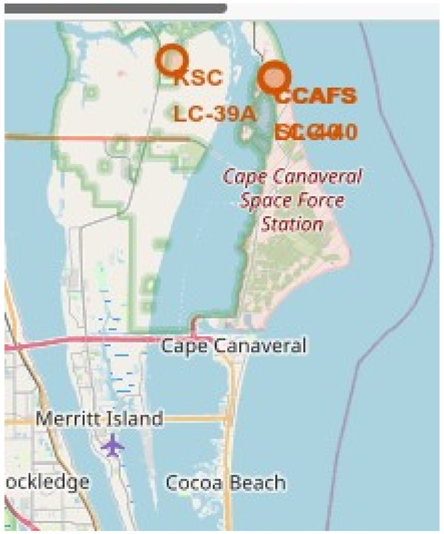
- There are four unique launch site locations from the Space X data.



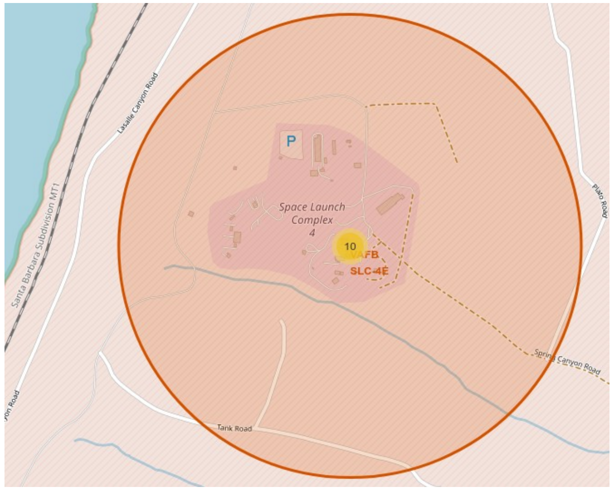
| | Launch Site | Lat | Long |
|---|---|---|---|
| **0** | CCAFS LC-40 | 28.562302 | -80.577356 |
| **1** | CCAFS SLC-40 | 28.563197 | -80.576820 |
| **2** | KSC LC-39A | 28.573255 | -80.646895 |
| **3** | VAFB SLC-4E | 34.632834 | -120.610746 |

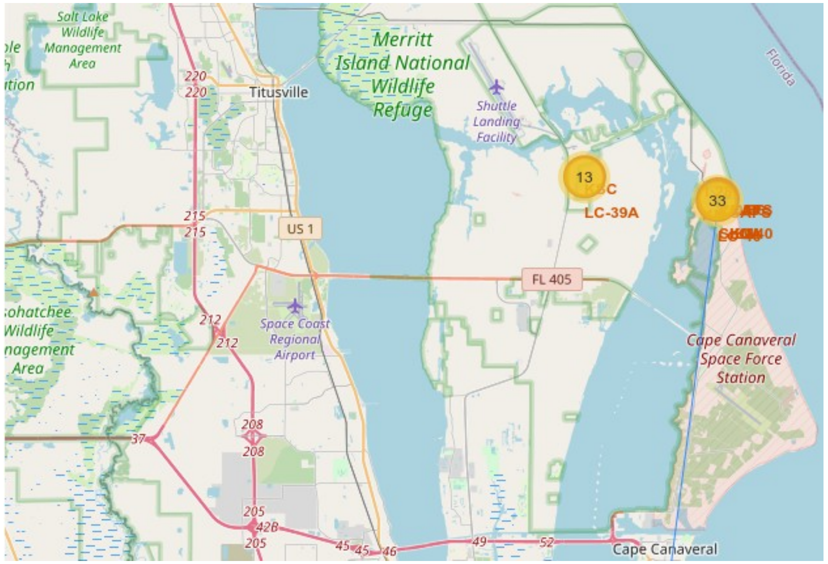
# All Launch Site Locations



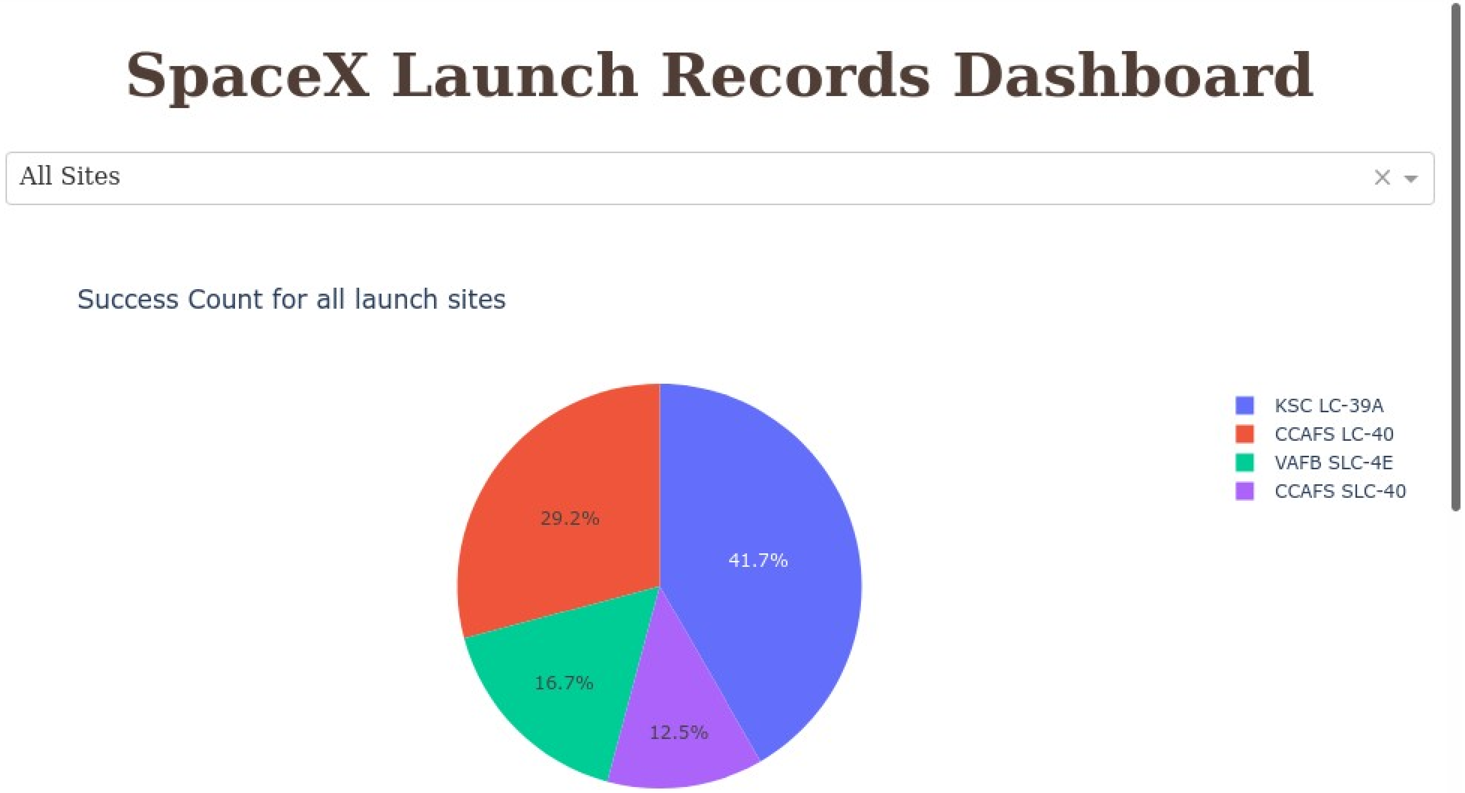


# Distances between a launch site to its proximities




Salt Lake Wildlife Management Area
Titusville
Merritt Island National Wildlife Refuge
Shuttle Landing Facility
13
LC-39A
33
US 1
FL 405
Space Coast Regional Airport
Cape Canaveral Space Force Station
Cape Canaveral
Florida

## Pie Chart of Successful Launches by Site

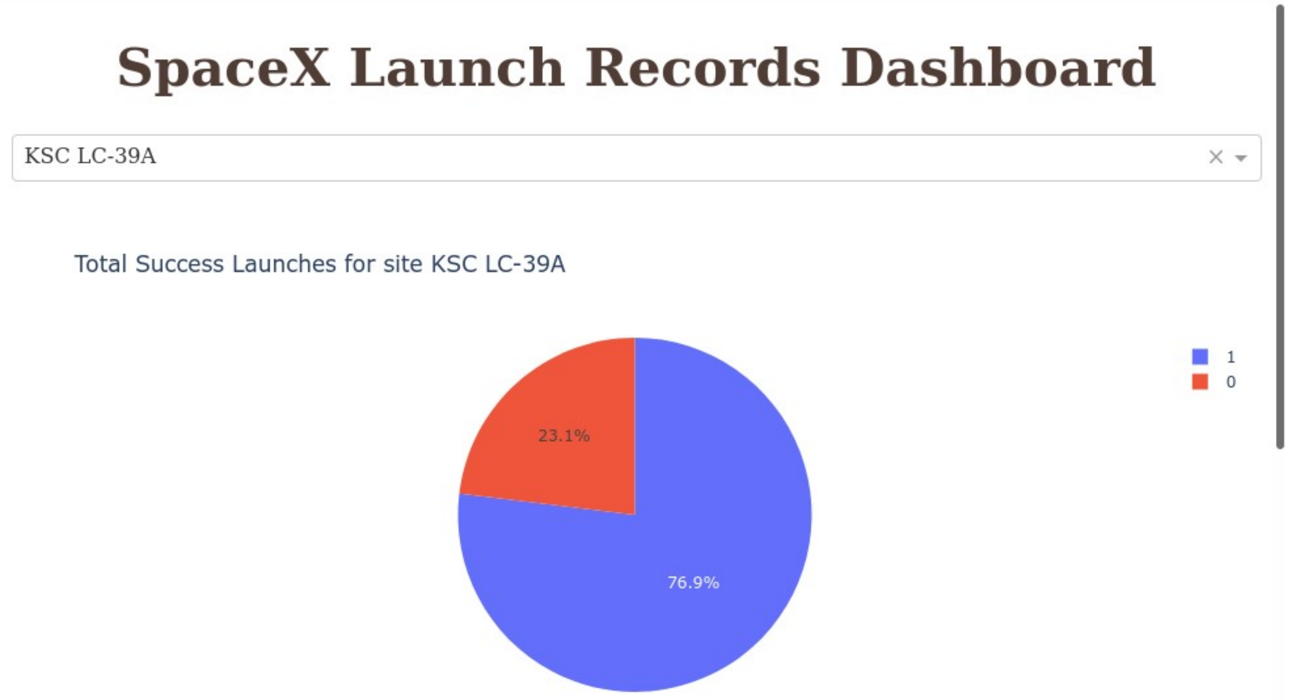


SpaceX Launch Records Dashboard

All Sites

Success Count for all launch sites

## Most Proportionally Successful Launch Site



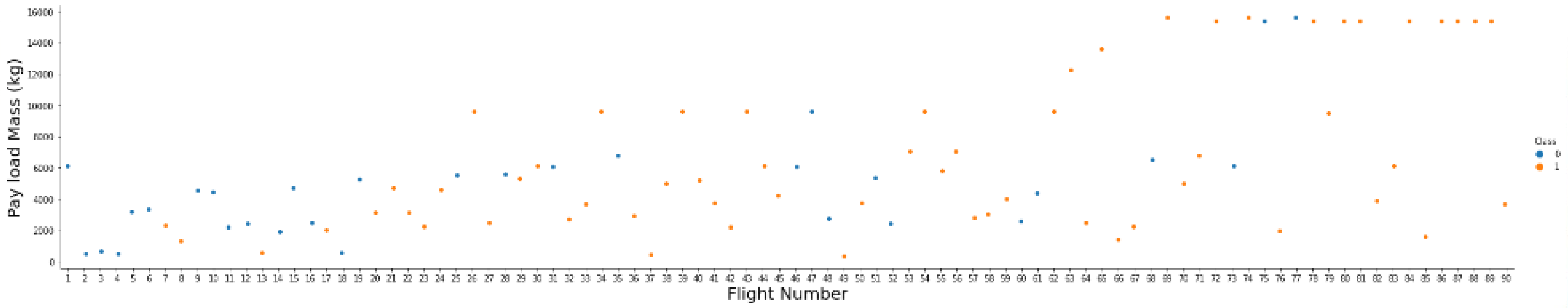
SpaceX Launch Records Dashboard

KSC LC-39A

Total Success Launches for site KSC LC-39A

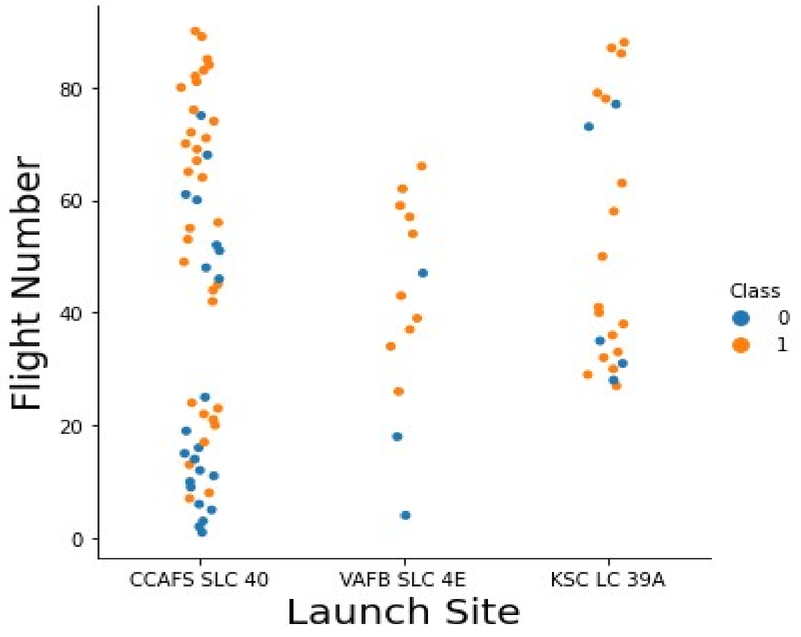
# Flight Number vs Payload Mass

- We see that as the flight number increases, the first stage is more likely to land successfully. The payload mass is also important; it seems the more massive the payload, the less likely the first stage will return.



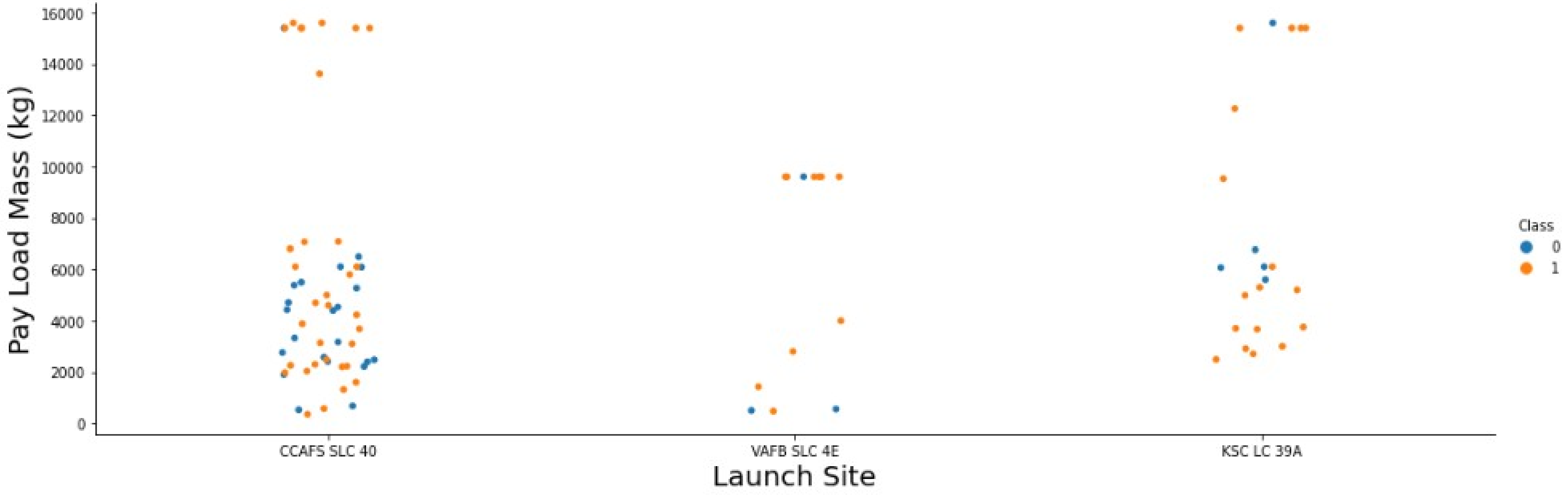
## Flight Number vs Launch Site

Rate of success has grown over time at each site, though KSC LC-39A seems to be the most consistent



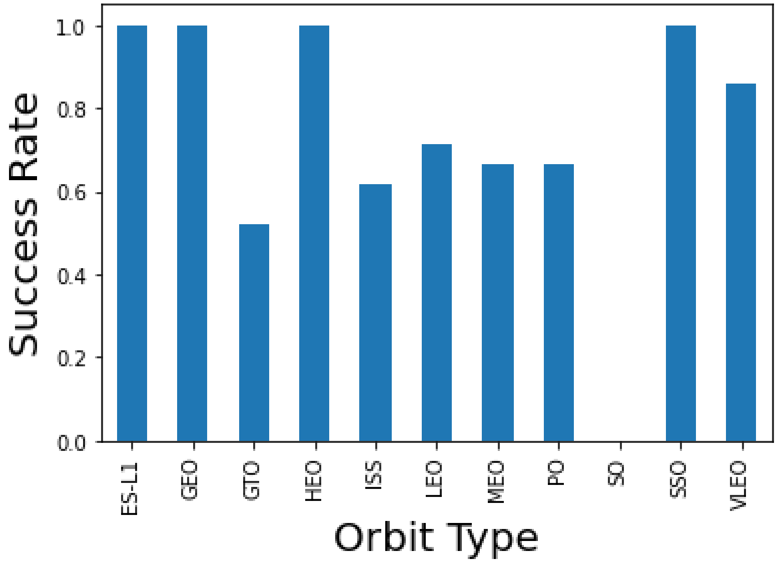
# Payload Vs. Launch Site

- We can observe that for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).



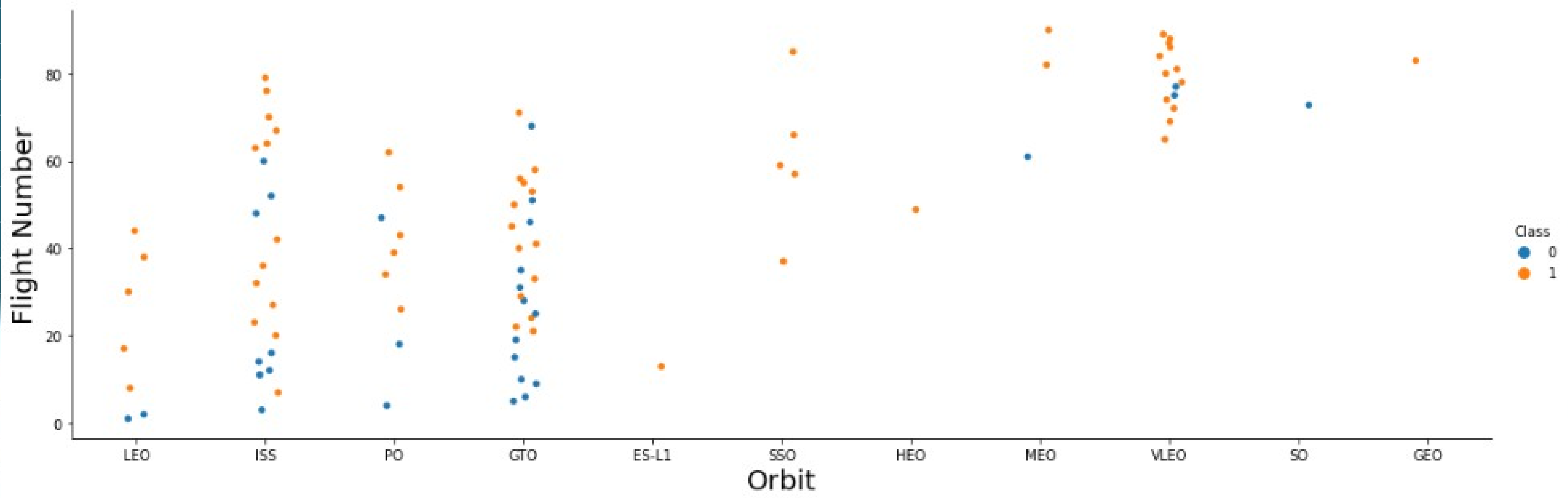
## Orbit Type vs. Success Rate

- We can see that there is a relationship between success rate and orbit type. ES-L1, SSO, HEO and GEO seems to be the most reliable.



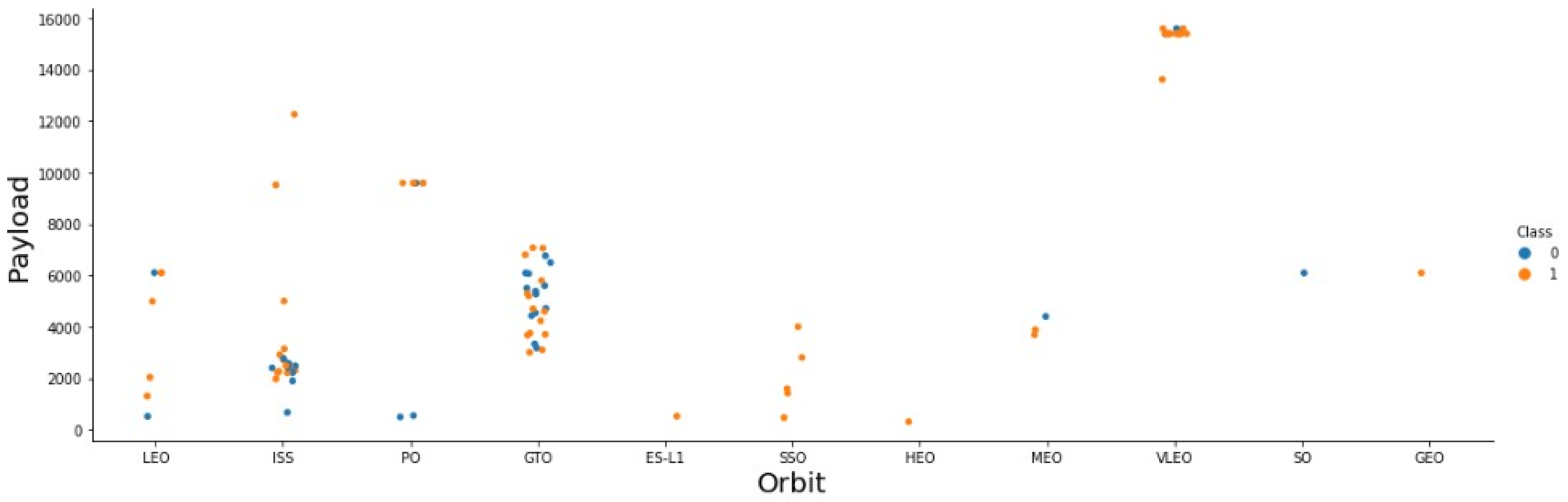
# Flight Number vs. Orbit Type

- However, ES-L1, HEO and GEO have just had only 1 launch each. VLEO seems to have both good success rate and high sample size.



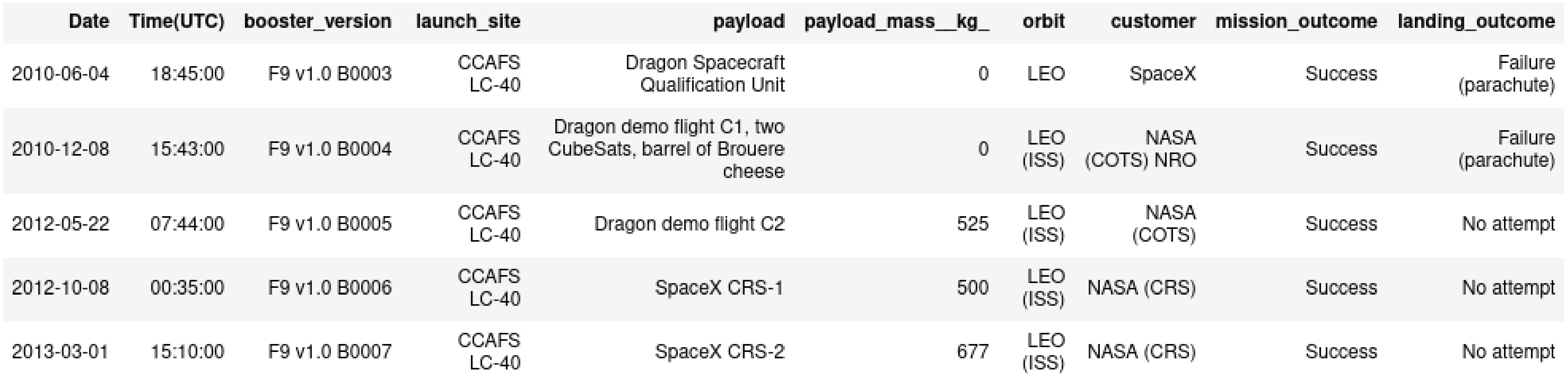
# Payload vs. Orbit Type

Vleo success may be due to heavy payloads. However LEO and SSO may be due to light payloads.



# SQL Queries

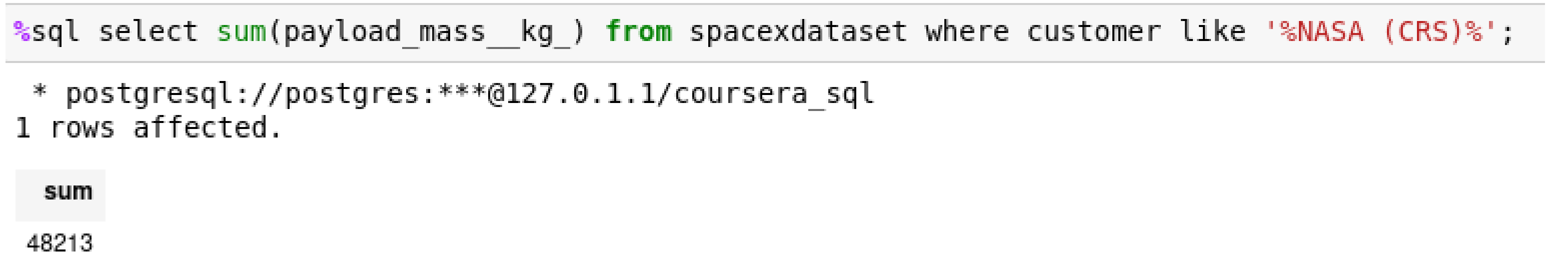
Here we can find first 5 records for launch site names that begin with 'CCA' :



| | 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 | 07:44:00 | F9 v1.0 B0005 | CCAFS LC-40 | Dragon demo flight C2 | 525 | LEO (ISS) | NASA (COTS) | Success | No attempt |
| | 2012-10-08 | 00:35:00 | 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 |

## Total Payload Mass

Here is the total payload mass (in kg) carried by boosters from NASA (CRS) :



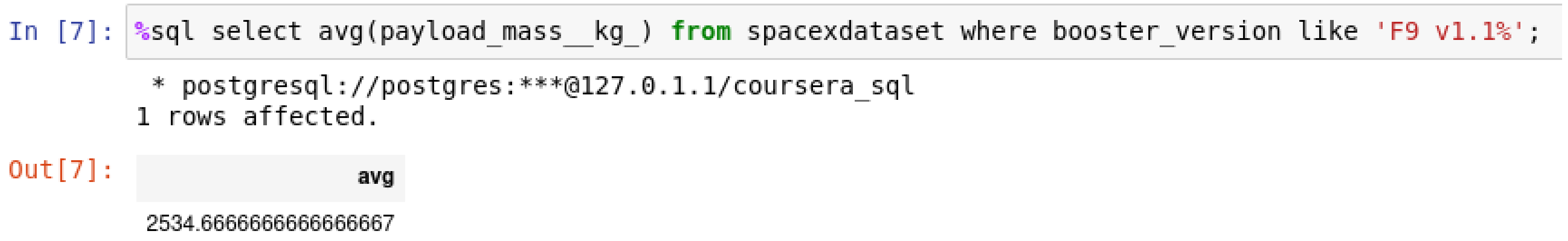
```
%sql select sum(payload_mass__kg_) from spacexdataset where customer like '%NASA (CRS)%';
```

```
 * postgresql://postgres:***@127.0.1.1/coursera_sql
1 rows affected.
```

| sum |
|---|
| 48213 |

# Average Payload Mass by F9 v.1.1

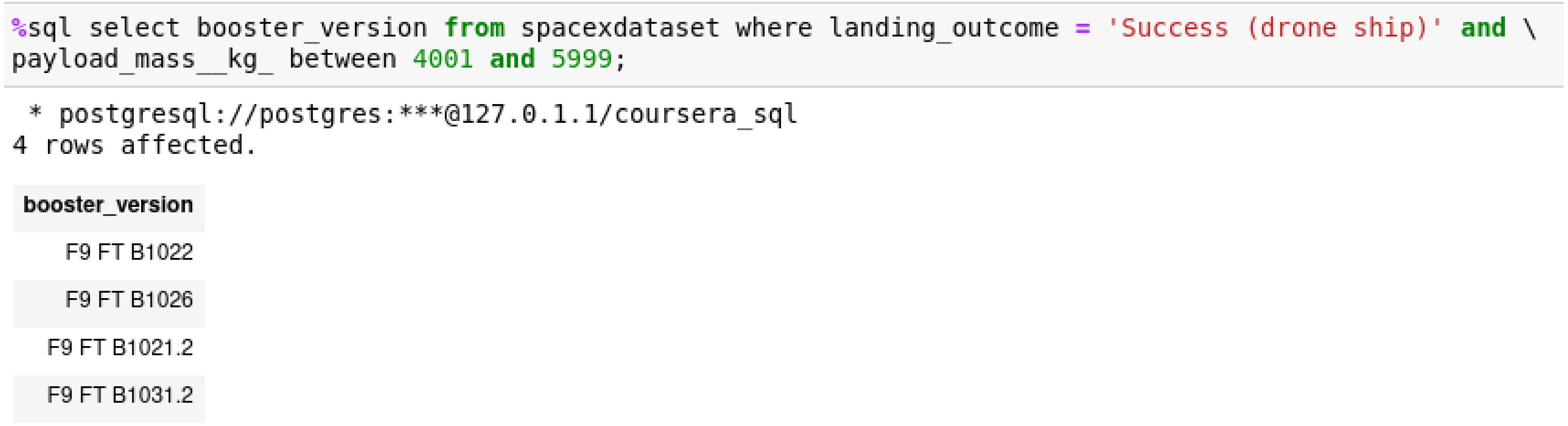
- On average, rockets of booster version F9 v1.1 carry a mass of:



```
In [7]: %sql select avg(payload_mass__kg_) from spacexdataset where booster_version like 'F9 v1.1%';

         * postgresql://postgres:***@127.0.1.1/coursera_sql
        1 rows affected.

Out[7]:                avg
        2534.666666666666667
```

# Successful Drone Ship Landing with Payload between 4000 and 6000

- This is a list of the names of boosters which have successfully landed on a drone ship and had payload mass greater than 4000kg but less than 6000kg:



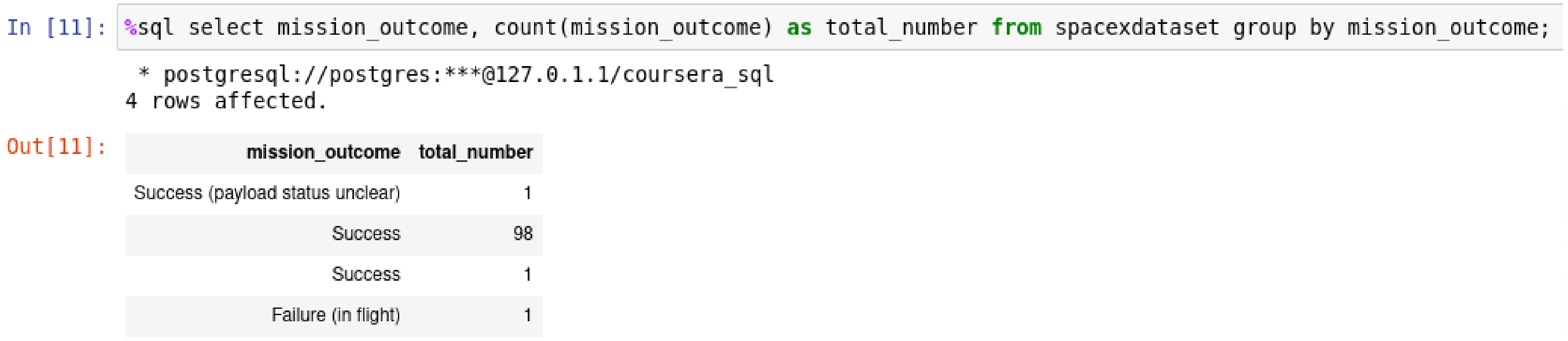
```
%sql select booster_version from spacexdataset where landing_outcome = 'Success (drone ship)' and \
payload_mass__kg_ between 4001 and 5999;
```

```
 * postgresql://postgres:***@127.0.1.1/coursera_sql
4 rows affected.
```

| booster_version |
|---|
| F9 FT B1022 |
| F9 FT B1026 |
| F9 FT B1021.2 |
| F9 FT B1031.2 |

# Total Number of Successful and Failure Mission Outcomes

- In total, there were 101 missions, out of which :



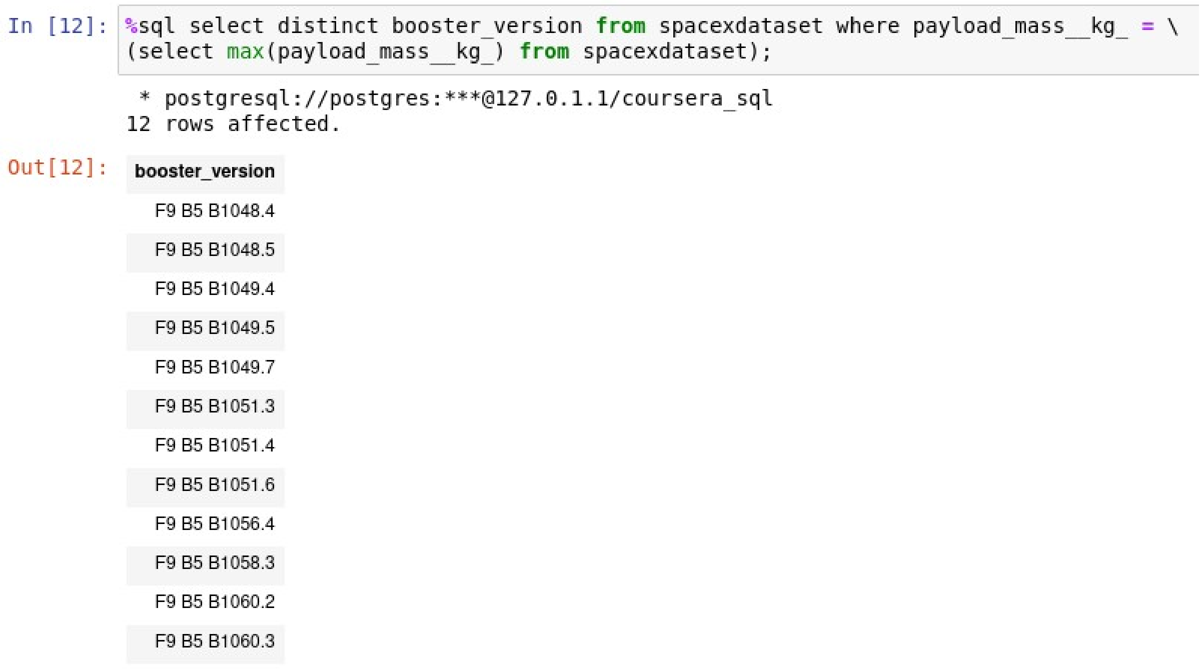
```
In [11]: %sql select mission_outcome, count(mission_outcome) as total_number from spacexdataset group by mission_outcome;

 * postgresql://postgres:***@127.0.1.1/coursera_sql
4 rows affected.
```

Out[11]:

| mission_outcome | total_number |
|---|---|
| Success (payload status unclear) | 1 |
| Success | 98 |
| Success | 1 |
| Failure (in flight) | 1 |

# Boosters Carried Maximum Payload

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

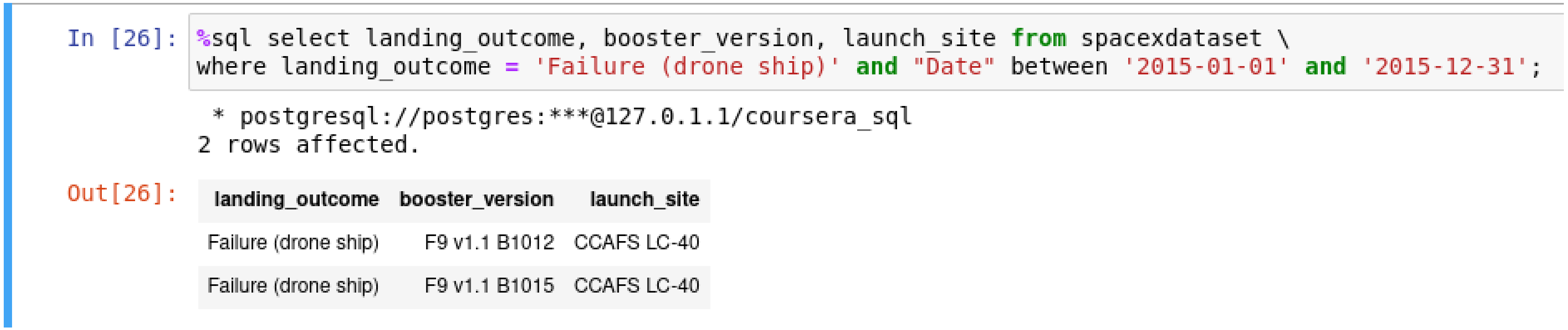


```
In [12]: %sql select distinct booster_version from spacexdataset where payload_mass__kg_ = \
         (select max(payload_mass__kg_) from spacexdataset);

          * postgresql://postgres:***@127.0.1.1/coursera_sql
         12 rows affected.
Out[12]:
```

| booster_version |
|---|
| F9 B5 B1048.4 |
| F9 B5 B1048.5 |
| F9 B5 B1049.4 |
| F9 B5 B1049.5 |
| F9 B5 B1049.7 |
| F9 B5 B1051.3 |
| F9 B5 B1051.4 |
| F9 B5 B1051.6 |
| F9 B5 B1056.4 |
| F9 B5 B1058.3 |
| F9 B5 B1060.2 |
| F9 B5 B1060.3 |

## 2015 Fail Records

- In 2015, there were two launches which resulted in a failed landing in drone ships :

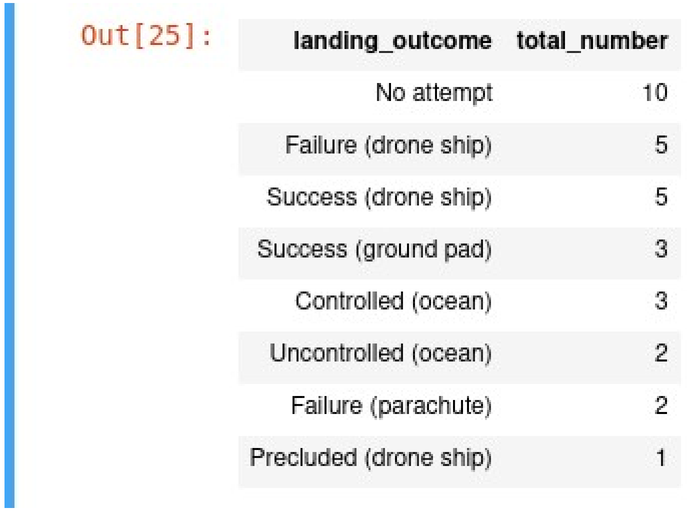


```
In [26]: %sql select landing_outcome, booster_version, launch_site from spacexdataset \
         where landing_outcome = 'Failure (drone ship)' and "Date" between '2015-01-01' and '2015-12-31';

          * postgresql://postgres:***@127.0.1.1/coursera_sql
         2 rows affected.

Out[26]:
```

| landing_outcome | booster_version | launch_site |
|---|---|---|
| Failure (drone ship) | F9 v1.1 B1012 | CCAFS LC-40 |
| Failure (drone ship) | F9 v1.1 B1015 | CCAFS LC-40 |

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

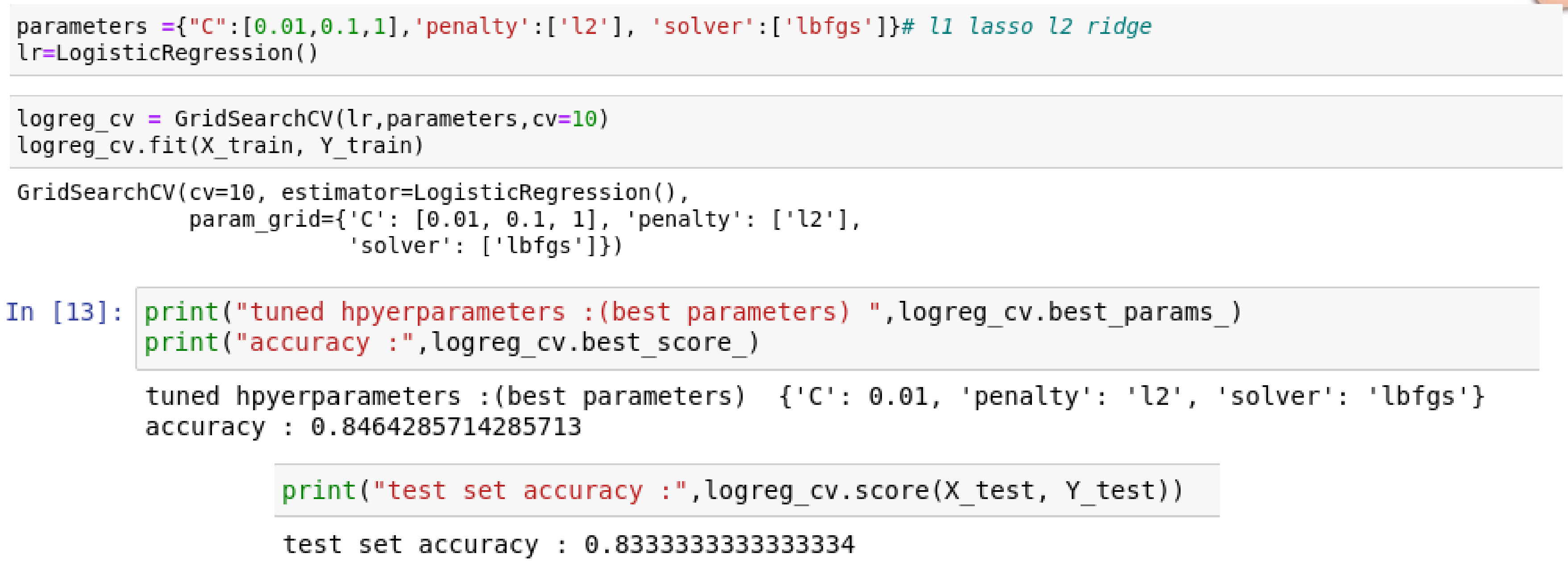
- This is a rank of the types and number of landing outcomes (in descending order) between dates 2010-06-04 and 2017-03-20:



Out[25]:

| landing_outcome | total_number |
|---|---|
| No attempt | 10 |
| Failure (drone ship) | 5 |
| Success (drone ship) | 5 |
| Success (ground pad) | 3 |
| Controlled (ocean) | 3 |
| Uncontrolled (ocean) | 2 |
| Failure (parachute) | 2 |
| Precluded (drone ship) | 1 |

## Classification Accuracy – Logistic Regression



```
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_train, Y_train)
```

```
GridSearchCV(cv=10, estimator=LogisticRegression(),
             param_grid={'C': [0.01, 0.1, 1], 'penalty': ['l2'],
                         'solver': ['lbfgs']})
```

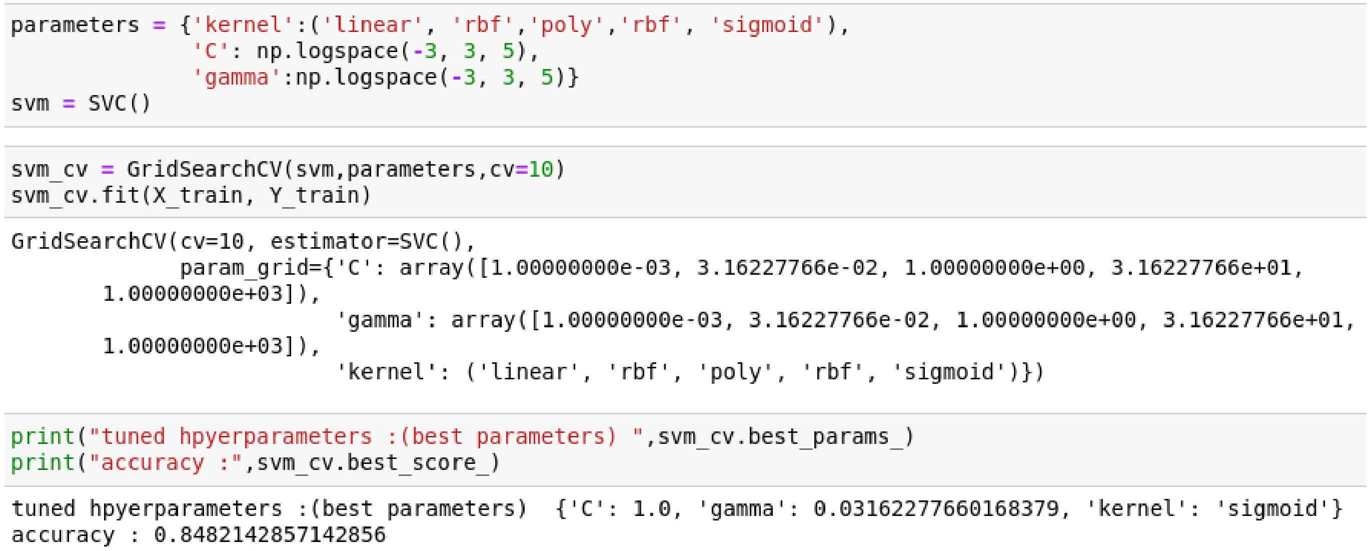
```
In [13]: print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
         print("accuracy :",logreg_cv.best_score_)
```

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

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

```
test set accuracy : 0.8333333333333334
```

# Classification Accuracy – SVM



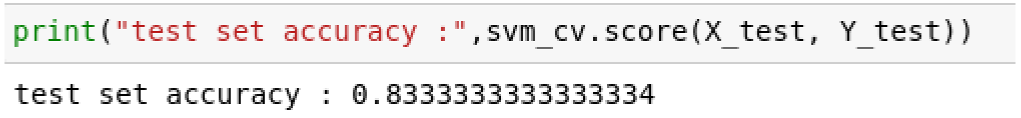
```
parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
              'C': np.logspace(-3, 3, 5),
              'gamma':np.logspace(-3, 3, 5)}
svm = SVC()
```

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

```
GridSearchCV(cv=10, 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_)
```

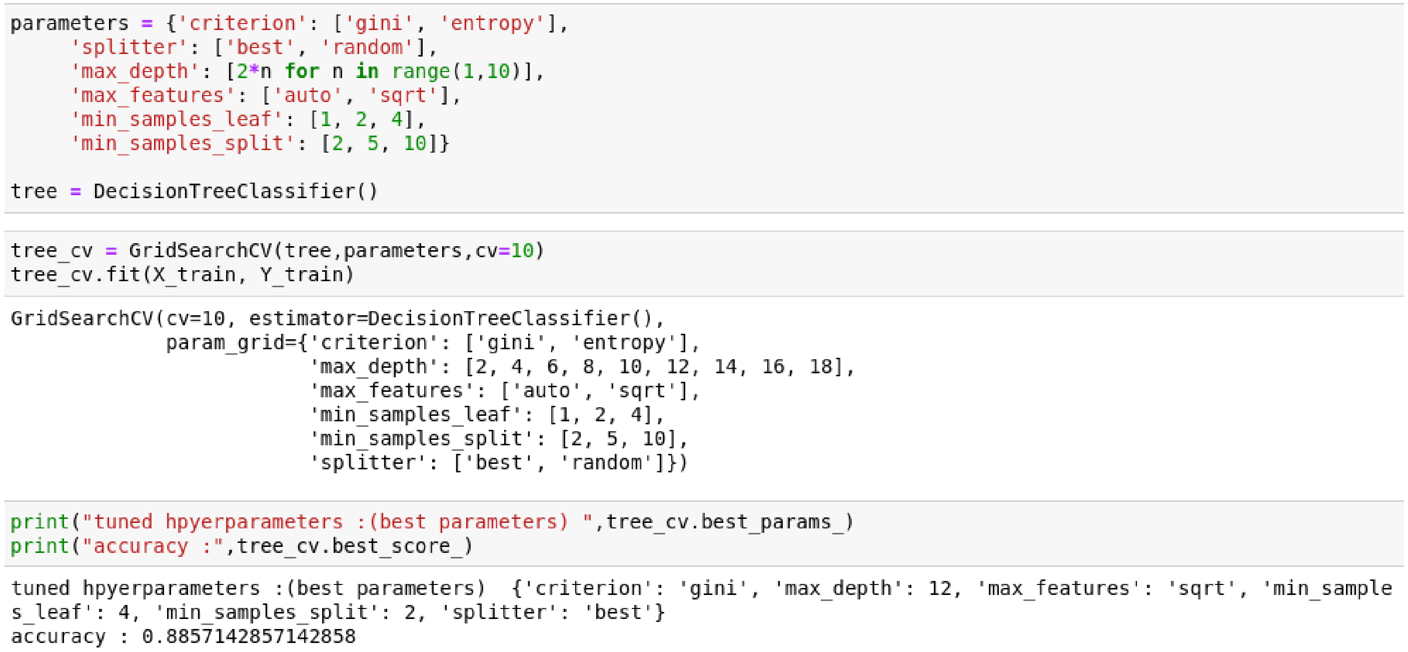
```
tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
accuracy : 0.8482142857142856
```



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

```
test set accuracy : 0.8333333333333334
```

# Classification Accuracy – Decision Tree



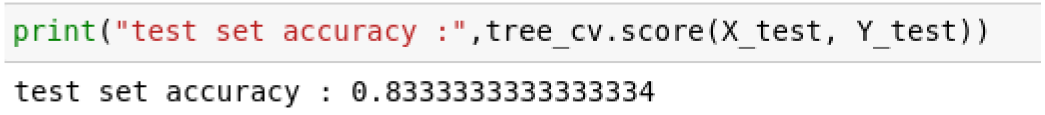
```
parameters = {'criterion': ['gini', 'entropy'],
     'splitter': ['best', 'random'],
     'max_depth': [2*n for n in range(1,10)],
     'max_features': ['auto', 'sqrt'],
     'min_samples_leaf': [1, 2, 4],
     'min_samples_split': [2, 5, 10]}

tree = DecisionTreeClassifier()
```

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

```
GridSearchCV(cv=10, 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_)
```

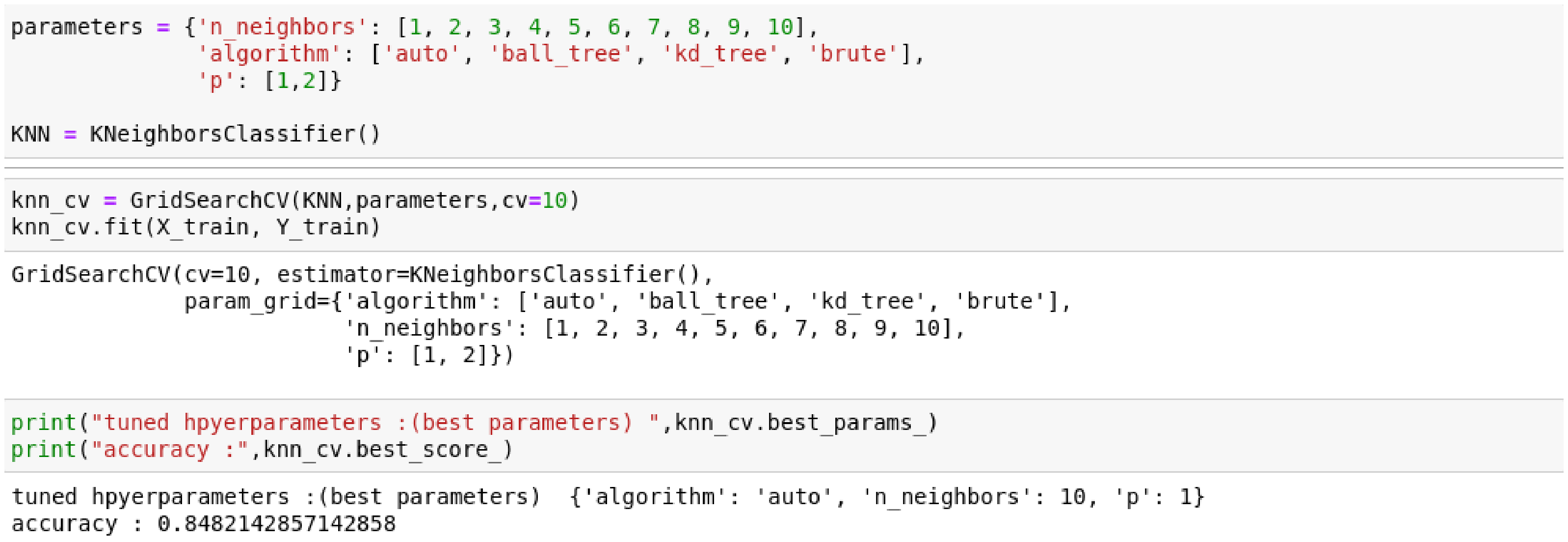
```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 12, 'max_features': 'sqrt', 'min_sample
s_leaf': 4, 'min_samples_split': 2, 'splitter': 'best'}
accuracy : 0.8857142857142858
```



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

```
test set accuracy : 0.8333333333333334
```

# Classification Accuracy – KNN



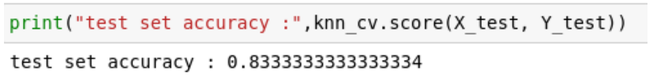
```
parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
              'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
              'p': [1,2]}

KNN = KNeighborsClassifier()
```

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

```
GridSearchCV(cv=10, 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': 10, 'p': 1}
accuracy : 0.8482142857142858
```

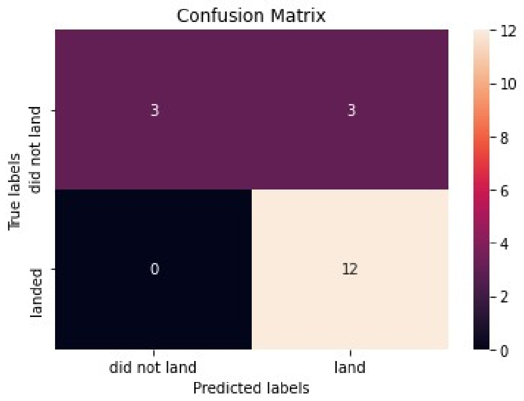


```
print("test set accuracy :",knn_cv.score(X_test, Y_test))
```

```
test set accuracy : 0.8333333333333334
```

# Accuracy and Confusion Matrix

- As we can see, all the presented models, tested on our test data, exhibit identical scores (83.33%)
- The confusion matrix shows that the major problem is the false positives .i.e., unsuccessful landing marked as successful landing by all the models.
- It is also possible that the tested portion was too small.



## Conclusions

- We can conclude that:
  - Rate of launch success has grown over time at each site, though KSC LC-39A seems to be the most consistent - 76,9 % success ratio,
  - Success rate kept on increasing between 2013 and 2020,
  - VLEO orbit seems to have both good success rate and high sample size,
  - In general the more massive the payload, the less likely the first stage will return. LEO and SSO successes may be due to light payloads. However, Vleo success may be due to heavy payloads.
- Our machine learning model can predict the outcome of a given landing with a 83.33% degree of accuracy.

# Appendix

## Thank you!