SpaceX Final Assignment

Bayu Fathurrahman

Executive Summary

- Methodologies
  - Data Collection with API
  - Data Wrangling
  - EDA with SQL
  - EDA and Data Viz with pandas
  - Mapping with Folium
  - Machine Learning Prediction
- Results
  - EDA results
  - Classification modelling results

Table of Contents

- Introduction
- Methodology
- Results
- Discussion
- Conclusion
- Appendix

Introduction

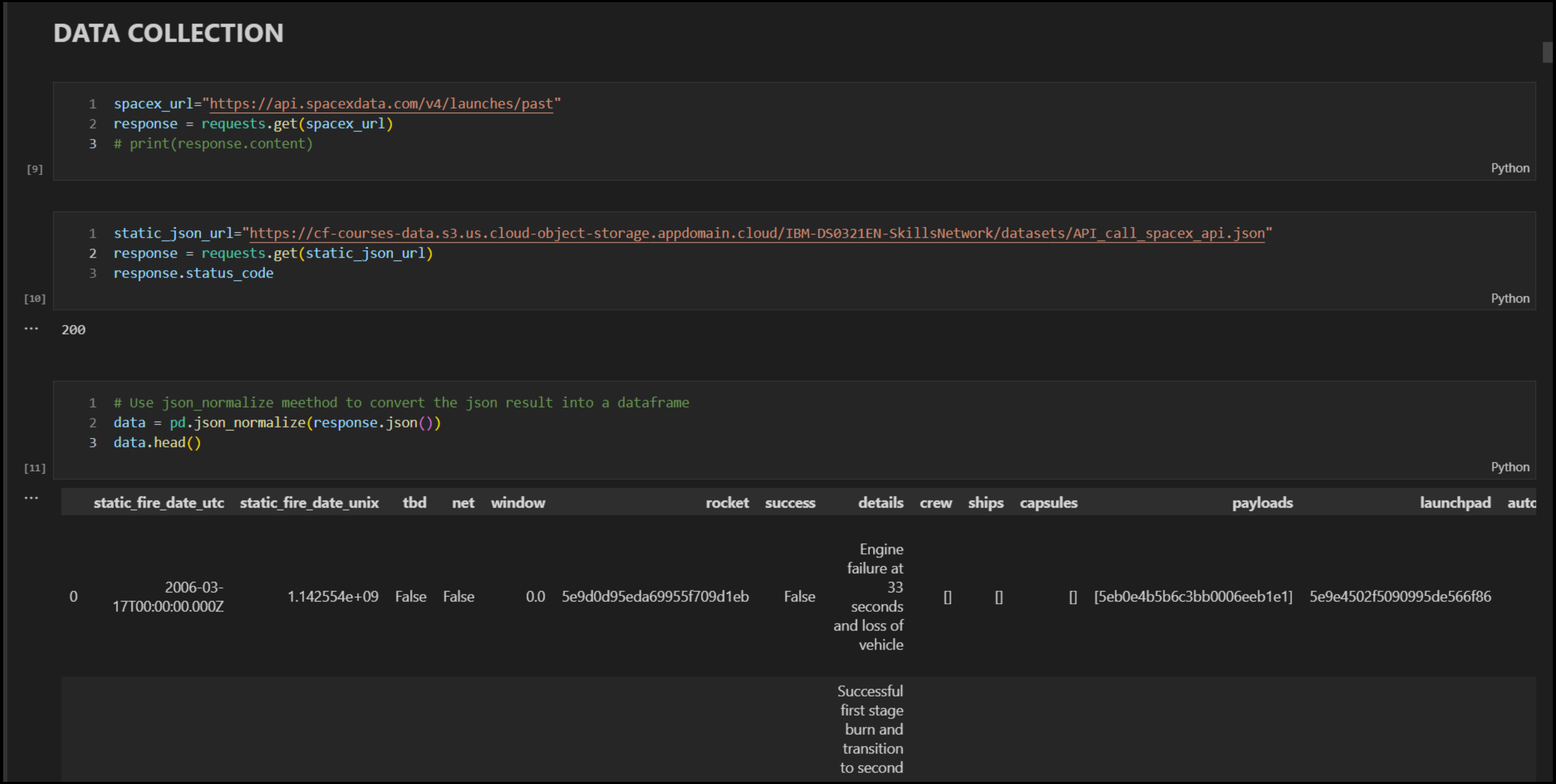
In this capstone, we will predict if the Falcon 9 first stage will land successfully. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. 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:

- Do we have an increasing rate of success over time?
- How does payload affects the success rate of the landing?
- Which features contribute to the success rate of the landing?

Methodology

Data collection with API



DATA COLLECTION

[9]

```python
spacex_url="https://api.spacexdata.com/v4/launches/past"
response = requests.get(spacex_url)
# print(response.content)
```

[10]

```python
static_json_url="https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json"
response = requests.get(static_json_url)
response.status_code
```

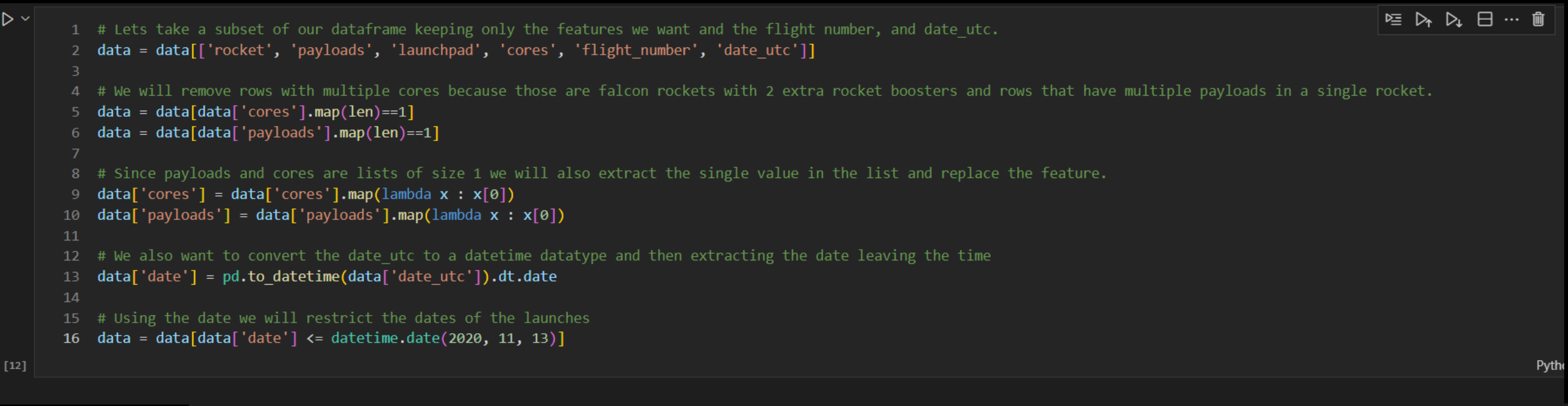
```
200
```

[11]

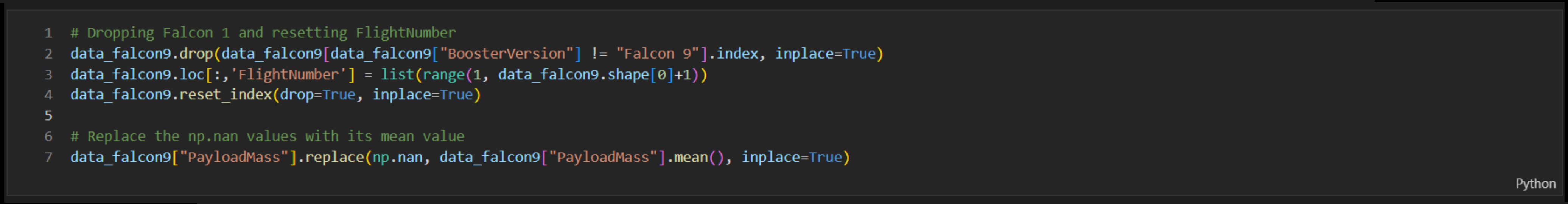
```python
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
data.head()
```

	static_fire_date_utc	static_fire_date_unix	tbd	net	window	rocket	success	details	crew	ships	capsules	payloads	launchpad	autc
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	False	0.0	5e9d0d95eda69955f709d1eb	False	Engine failure at 33 seconds and loss of vehicle	[]	[]	[]	[5eb0e4b5b6c3bb0006eeb1e1]	5e9e4502f5090995de566f86	
								Successful first stage burn and transition to second						

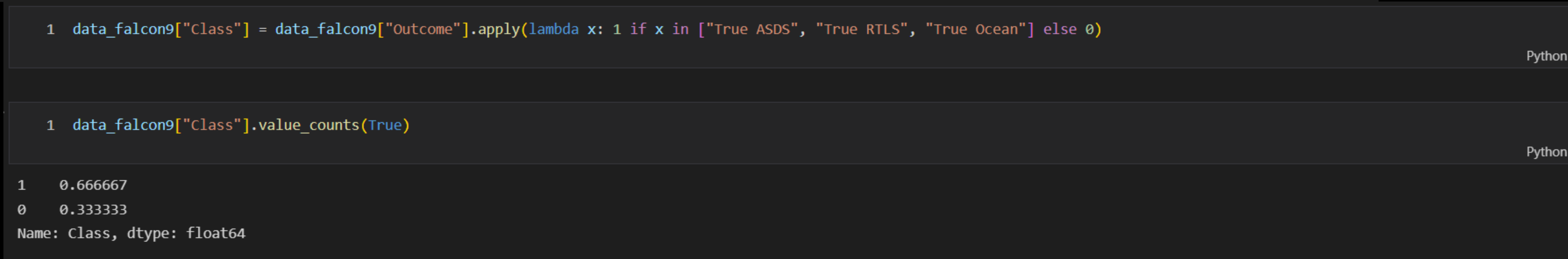
Data Wrangling



```python
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows that have multiple payloads in a single rocket.
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in the list and replace the feature.
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the time
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```



```python
# Dropping Falcon 1 and resetting FlightNumber
data_falcon9.drop(data_falcon9[data_falcon9["BoosterVersion"] != "Falcon 9"].index, inplace=True)
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9.reset_index(drop=True, inplace=True)

# Replace the np.nan values with its mean value
data_falcon9["PayloadMass"].replace(np.nan, data_falcon9["PayloadMass"].mean(), inplace=True)
```

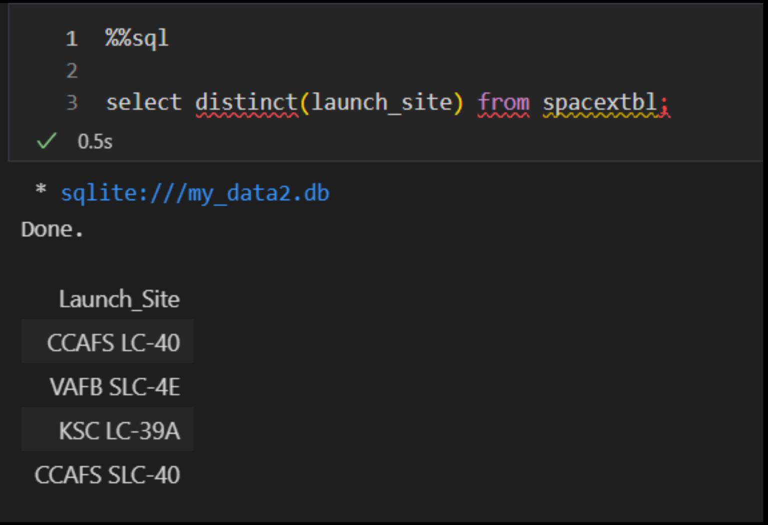


```python
data_falcon9["Class"] = data_falcon9["Outcome"].apply(lambda x: 1 if x in ["True ASDS", "True RTLS", "True Ocean"] else 0)
```

```python
data_falcon9["Class"].value_counts(True)
```

```
1    0.666667
0    0.333333
Name: Class, dtype: float64
```

EDA with SQL

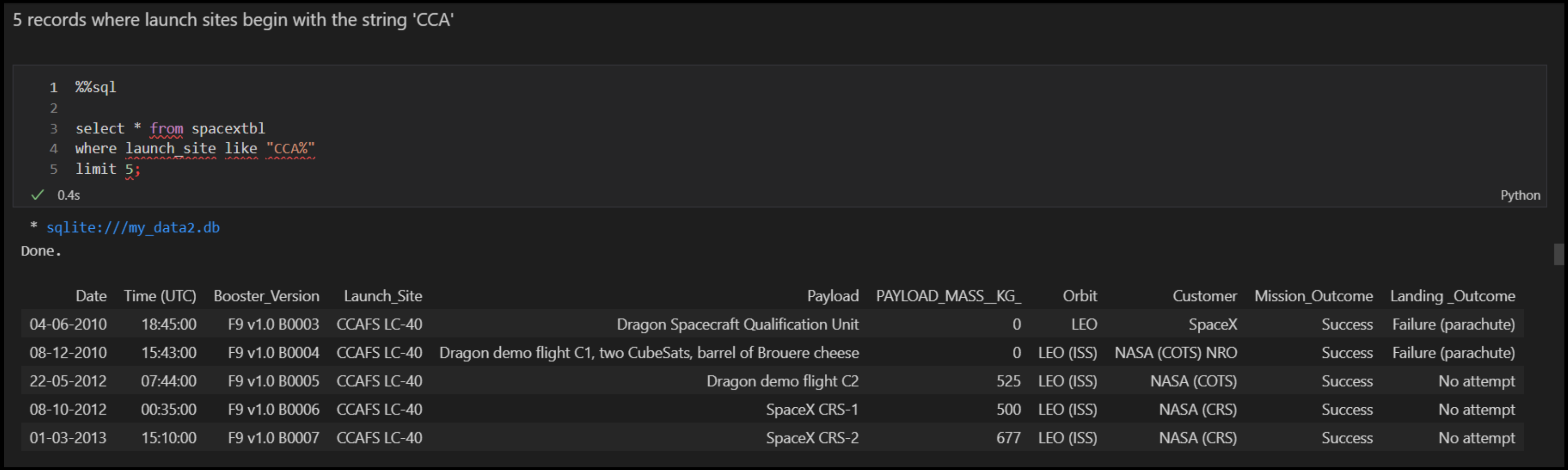


```
%%sql

select distinct(launch_site) from spacextbl;
```

✓ 0.5s

```
 * sqlite:///my_data2.db
Done.
```

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



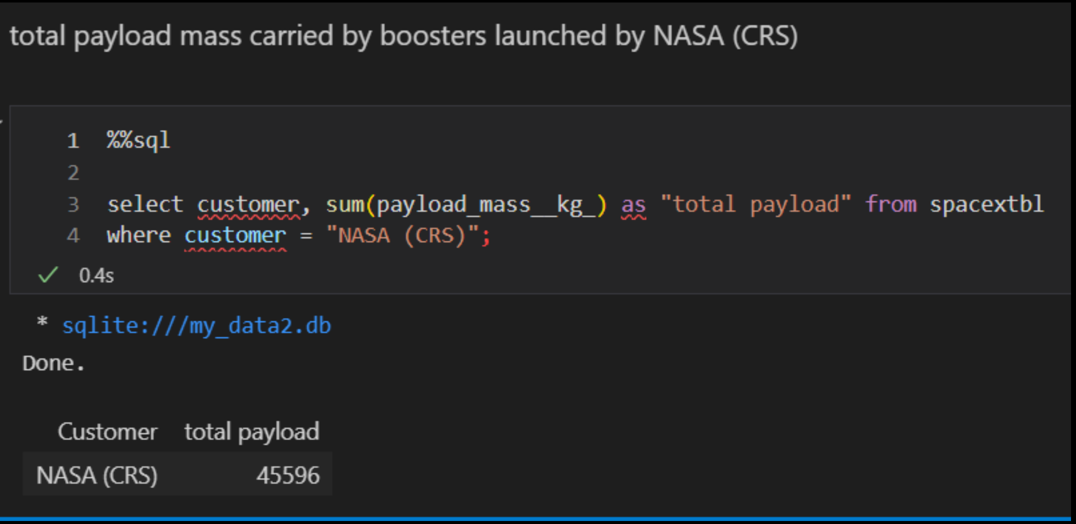
5 records where launch sites begin with the string 'CCA'

```
%%sql

select * from spacextbl
where launch_site like "CCA%"
limit 5;
```

✓ 0.4s Python

```
 * sqlite:///my_data2.db
Done.
```

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



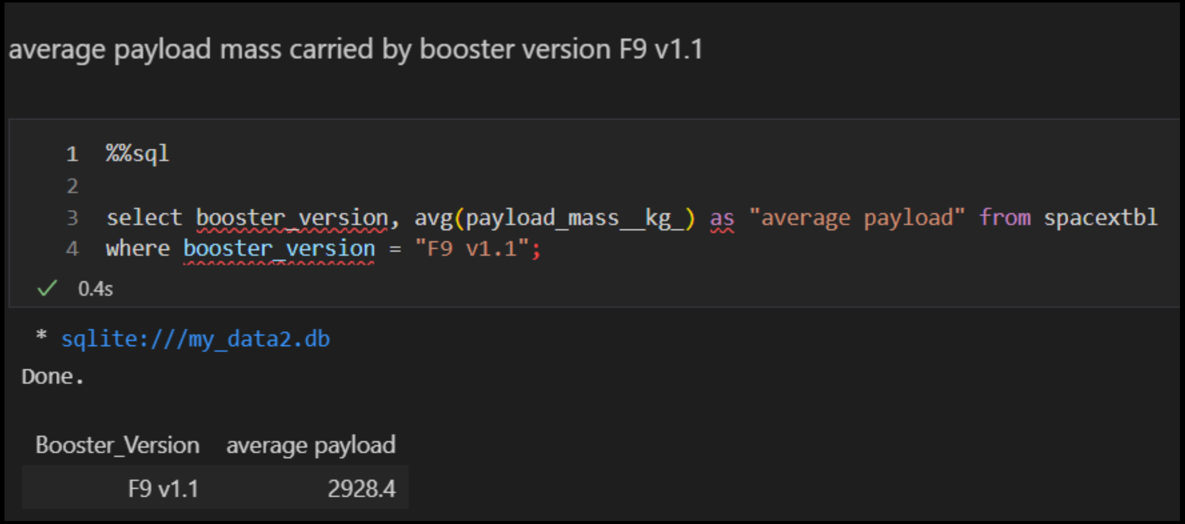
total payload mass carried by boosters launched by NASA (CRS)

```
%%sql

select customer, sum(payload_mass__kg_) as "total payload" from spacextbl
where customer = "NASA (CRS)";
```

✓ 0.4s

```
 * sqlite:///my_data2.db
Done.
```

Customer	total payload
NASA (CRS)	45596



average payload mass carried by booster version F9 v1.1

```
%%sql

select booster_version, avg(payload_mass__kg_) as "average payload" from spacextbl
where booster_version = "F9 v1.1";
```

✓ 0.4s

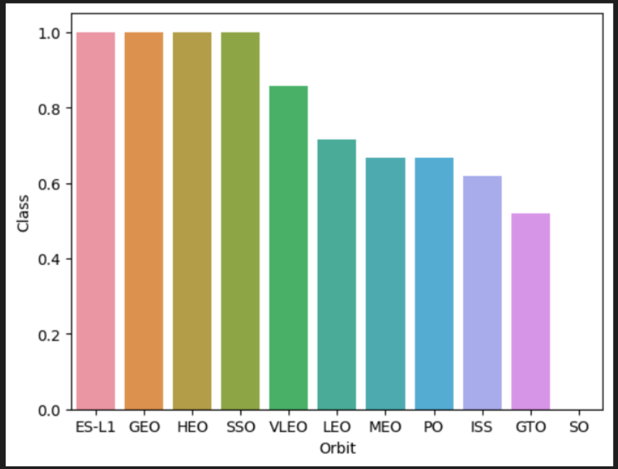
```
 * sqlite:///my_data2.db
Done.
```

Booster_Version	average payload
F9 v1.1	2928.4

EDA with pandas

```python
df_orbit = df.groupby("Orbit").mean()[["Class"]].sort_values("Class", ascending=False)
sns.barplot(x=df_orbit.index, y=df_orbit["Class"]);
```

Python



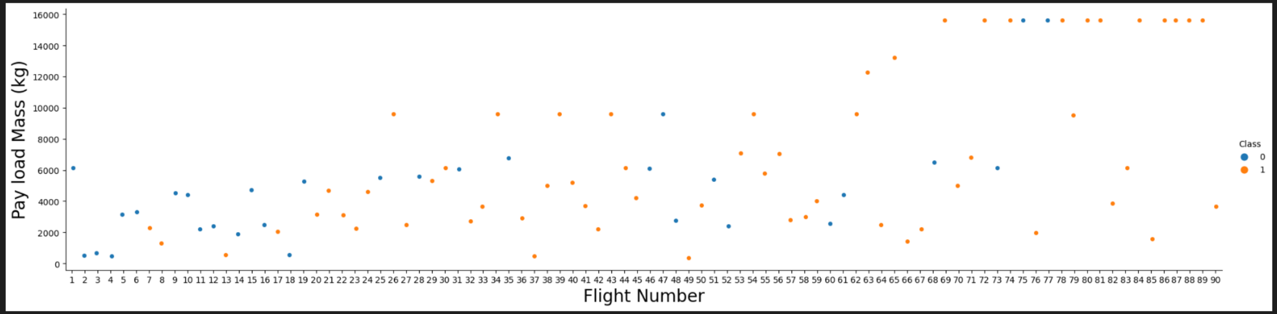
```python
df[df["Orbit"] == "SO"]
```

Python

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
72	73	2020-01-19	Falcon 9	6123.547647	SO	KSC LC 39A	None None	4	False	True	False	NaN	5	3	B1046	-80.603956	28.608058	0

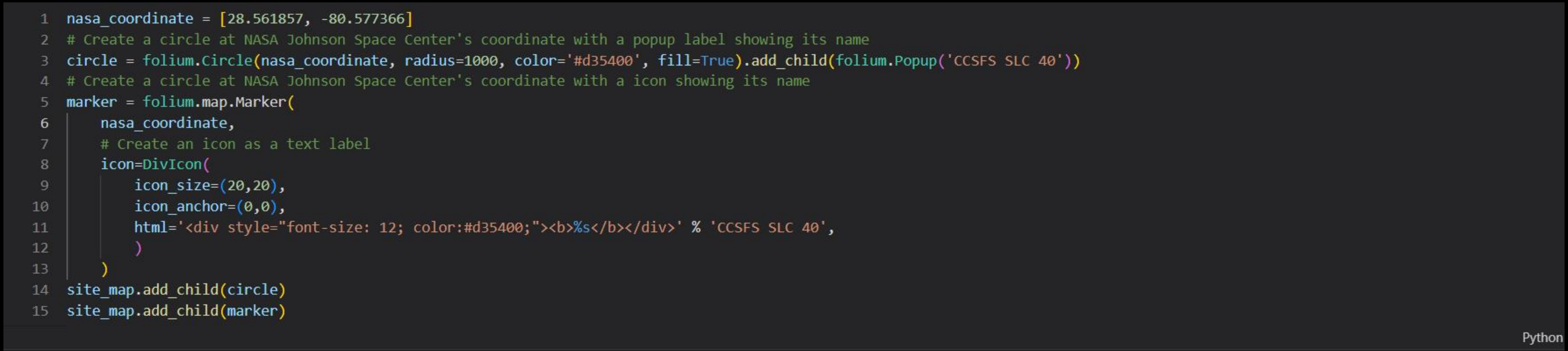
SO orbit has a success rate of 0% because there's only 1 launch attempt and it hasn't succeed. Meanwhile, Launch on orbit ES-L1, GEO, HEO, and SSO have 100% success rate

```python
style.use("default")
sns.catplot(y="PayloadMass", x="FlightNumber", hue="Class", data=df, aspect=4)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```

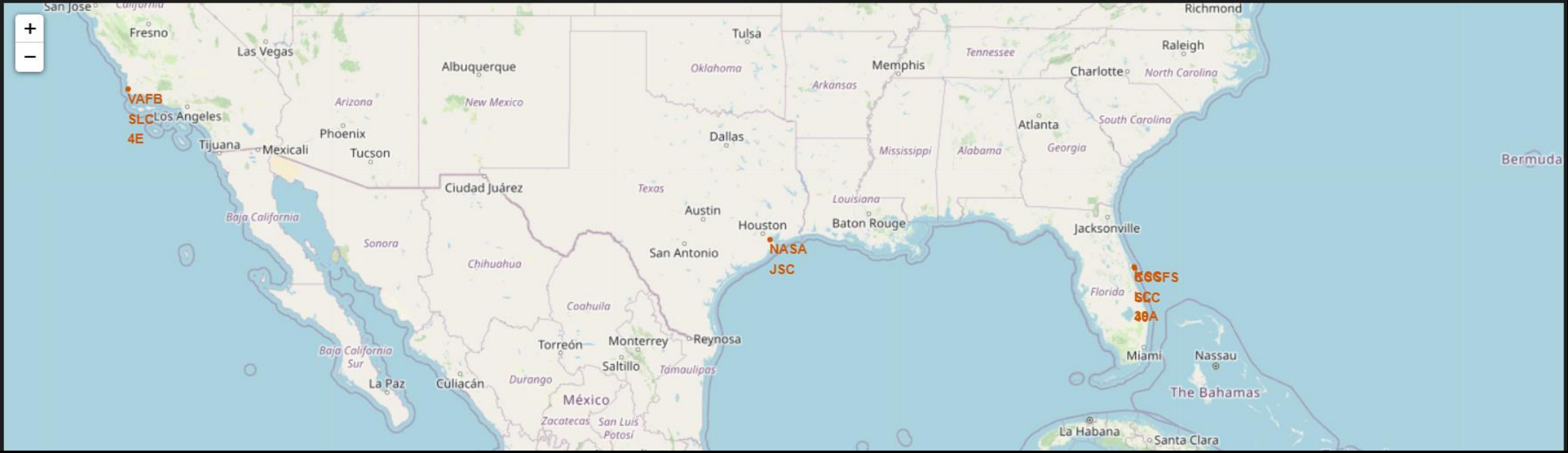


We see that as the flight number increases, the first stage is more likely to land successfully. The payload mass is also important; it seems the more massive the payload, the less likely the first stage will return.

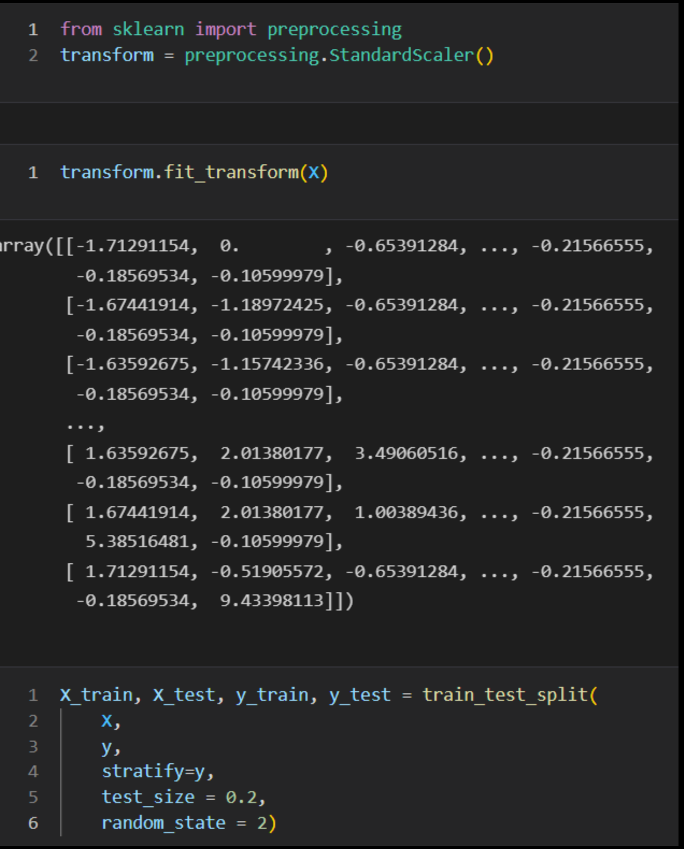
Mapping with Folium



```
nasa_coordinate = [28.561857, -80.577366]
# Create a circle at NASA Johnson Space Center's coordinate with a popup label showing its name
circle = folium.Circle(nasa_coordinate, radius=1000, color='#d35400', fill=True).add_child(folium.Popup('CCSFS SLC 40'))
# Create a circle at NASA Johnson Space Center's coordinate with a icon showing its name
marker = folium.map.Marker(
    nasa_coordinate,
    # Create an icon as a text label
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % 'CCSFS SLC 40',
        )
    )
site_map.add_child(circle)
site_map.add_child(marker)
```



Modelling

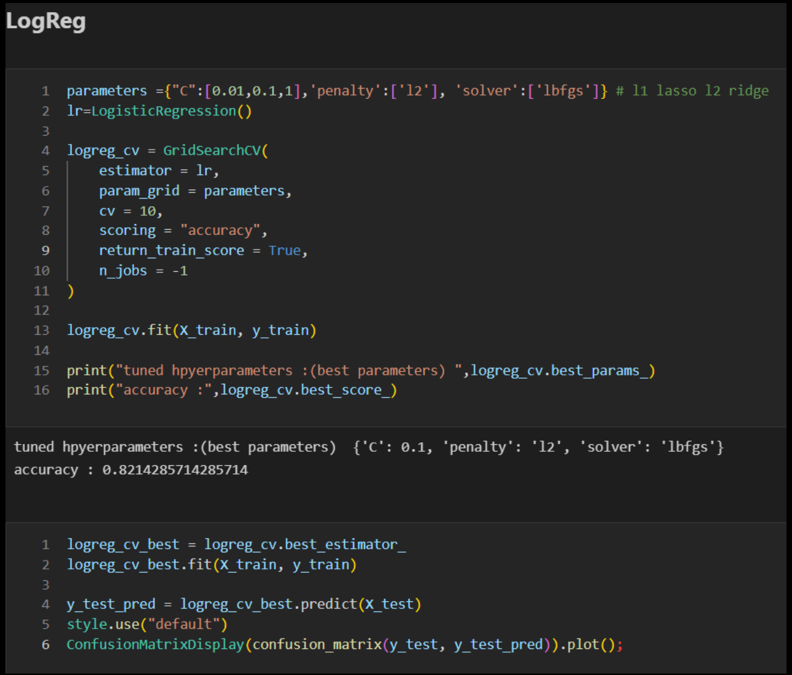


```
from sklearn import preprocessing
transform = preprocessing.StandardScaler()
```

```
transform.fit_transform(X)
```

```
rray([[-1.71291154,  0.        , -0.65391284, ..., -0.21566555,
        -0.18569534, -0.10599979],
       [-1.67441914, -1.18972425, -0.65391284, ..., -0.21566555,
        -0.18569534, -0.10599979],
       [-1.63592675, -1.15742336, -0.65391284, ..., -0.21566555,
        -0.18569534, -0.10599979],
       ...,
       [ 1.63592675,  2.01380177,  3.49060516, ..., -0.21566555,
        -0.18569534, -0.10599979],
       [ 1.67441914,  2.01380177,  1.00389436, ..., -0.21566555,
         5.38516481, -0.10599979],
       [ 1.71291154, -0.51905572, -0.65391284, ..., -0.21566555,
        -0.18569534,  9.43398113]])
```

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

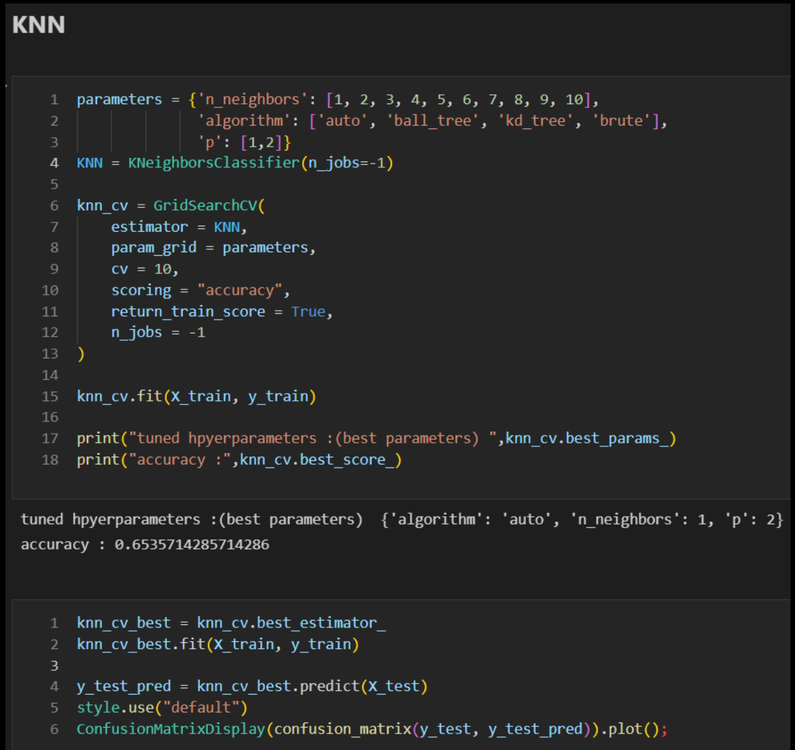


LogReg

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

logreg_cv = GridSearchCV(
    estimator = lr,
    param_grid = parameters,
    cv = 10,
    scoring = "accuracy",
    return_train_score = True,
    n_jobs = -1
)

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': 0.1, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8214285714285714
```

```
logreg_cv_best = logreg_cv.best_estimator_
logreg_cv_best.fit(X_train, y_train)

y_test_pred = logreg_cv_best.predict(X_test)
style.use("default")
ConfusionMatrixDisplay(confusion_matrix(y_test, y_test_pred)).plot();
```

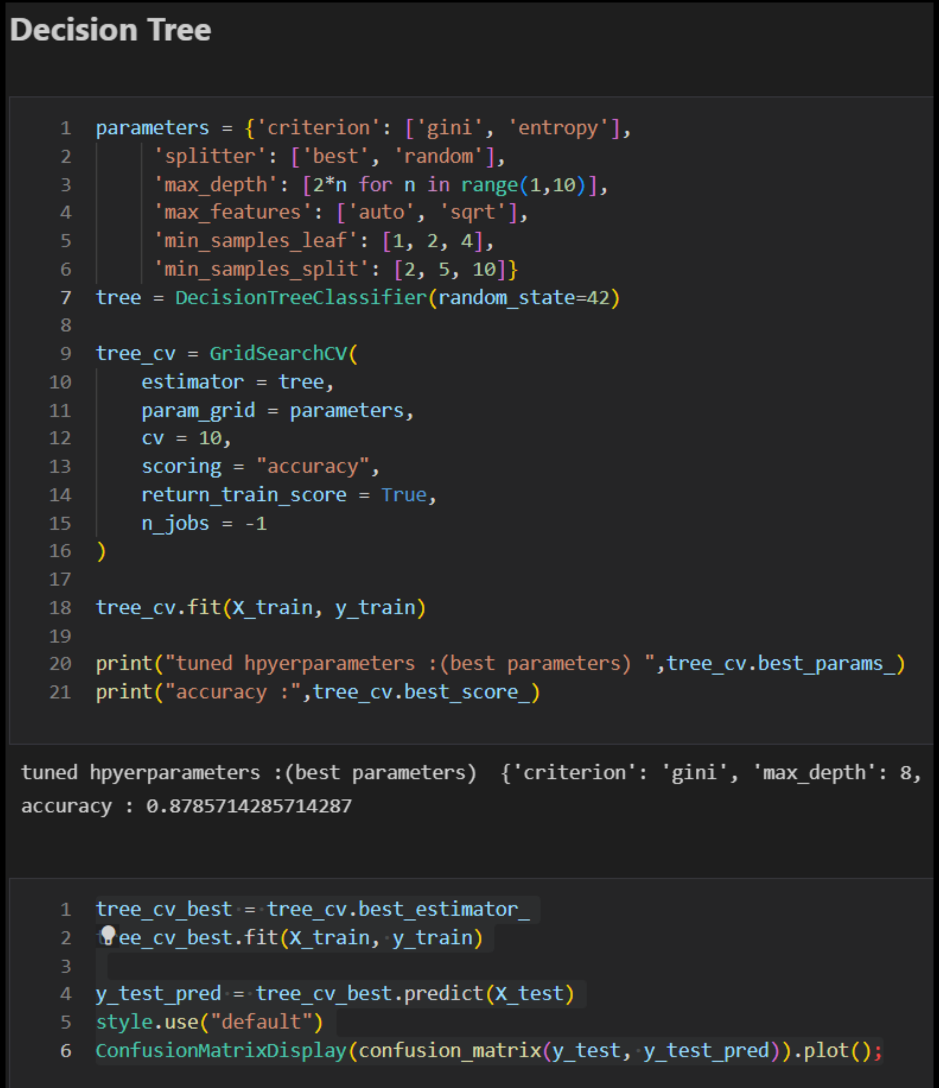


KNN

```
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(n_jobs=-1)

knn_cv = GridSearchCV(
    estimator = KNN,
    param_grid = parameters,
    cv = 10,
    scoring = "accuracy",
    return_train_score = True,
    n_jobs = -1
)

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': 1, 'p': 2}
accuracy : 0.6535714285714286
```

```
knn_cv_best = knn_cv.best_estimator_
knn_cv_best.fit(X_train, y_train)

y_test_pred = knn_cv_best.predict(X_test)
style.use("default")
ConfusionMatrixDisplay(confusion_matrix(y_test, y_test_pred)).plot();
```



Decision Tree

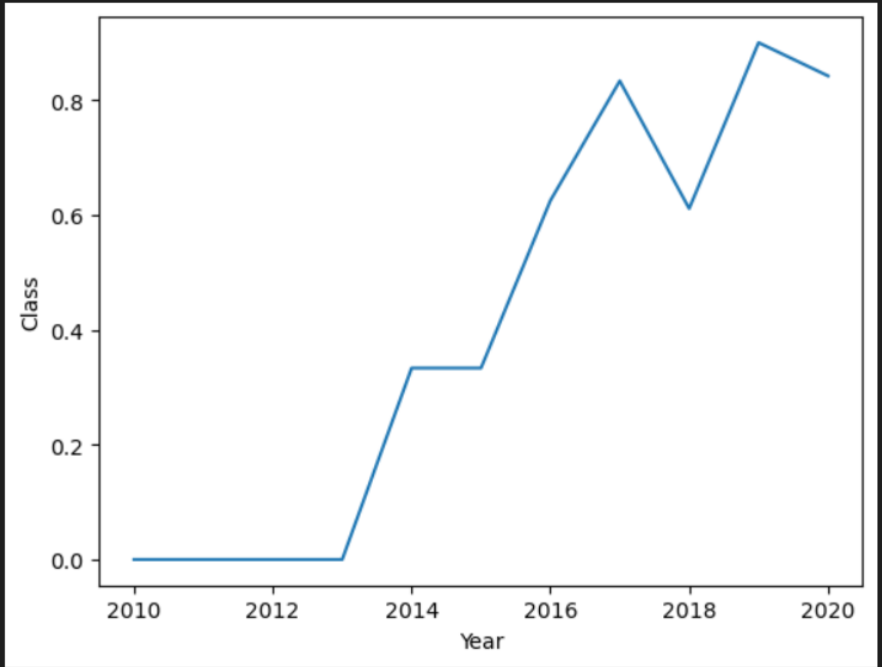
```
parameters = {'criterion': ['gini', 'entropy'],
     'splitter': ['best', 'random'],
     'max_depth': [2*n for n in range(1,10)],
     'max_features': ['auto', 'sqrt'],
     'min_samples_leaf': [1, 2, 4],
     'min_samples_split': [2, 5, 10]}
tree = DecisionTreeClassifier(random_state=42)

tree_cv = GridSearchCV(
    estimator = tree,
    param_grid = parameters,
    cv = 10,
    scoring = "accuracy",
    return_train_score = True,
    n_jobs = -1
)

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': 8,
accuracy : 0.8785714285714287
```

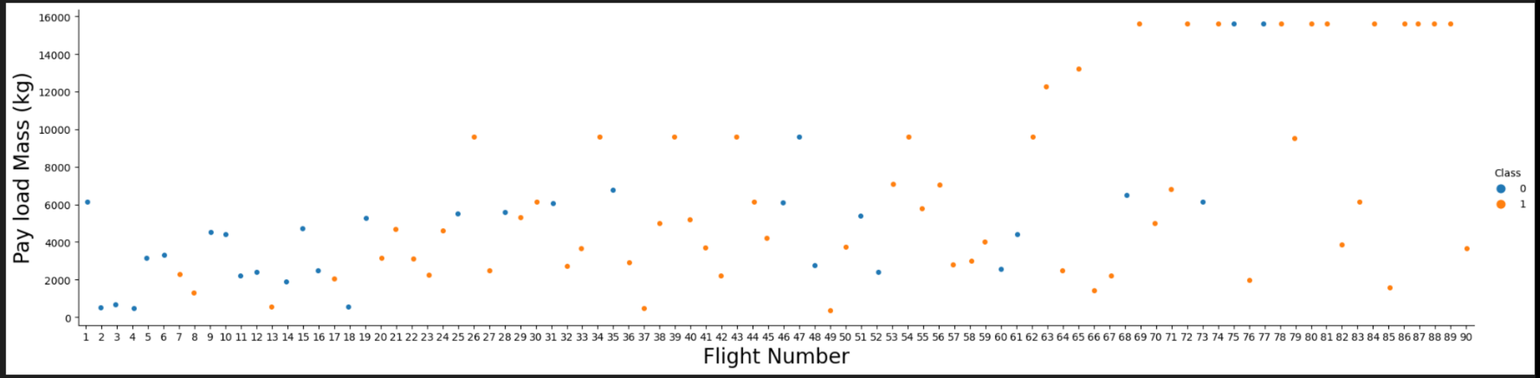
```
tree_cv_best = tree_cv.best_estimator_
 ee_cv_best.fit(X_train, y_train)

y_test_pred = tree_cv_best.predict(X_test)
style.use("default")
ConfusionMatrixDisplay(confusion_matrix(y_test, y_test_pred)).plot();
```

Results

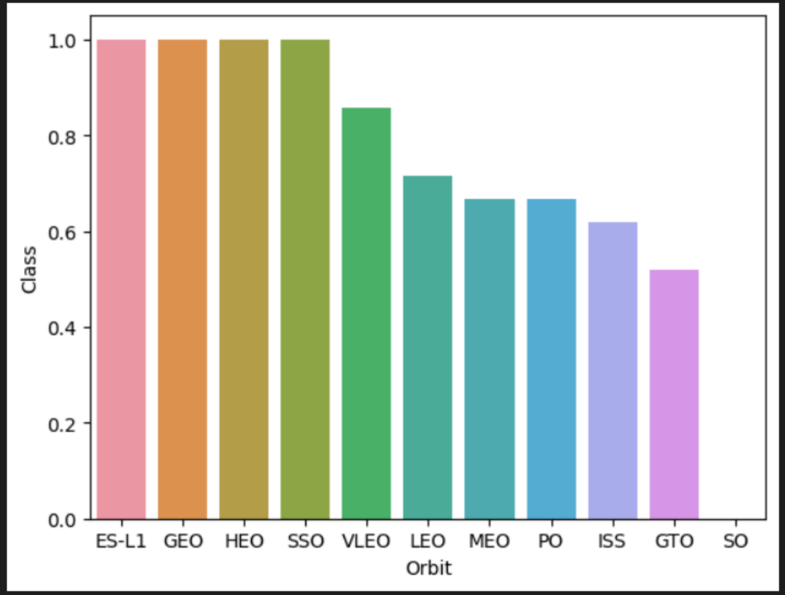
EDA Results



sucess rate since 2013 kept increasing until 2020



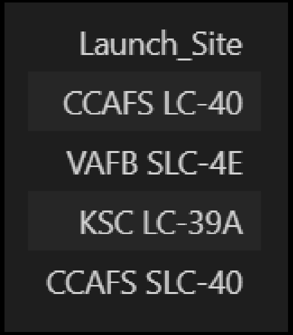
We see that as the flight number increases, the first stage is more likely to land successfully. The payload mass is also important; it seems the more massive the payload, the less likely the first stage will return.



SO orbit has a success rate of 0% because there's only 1 launch attempt and it hasn't succeed. Meanwhile, Launch on orbit ES-L1, GEO, HEO, and SSO have 100% success rate

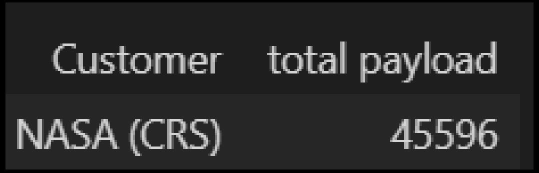
EDA Results

Name of the launch sites



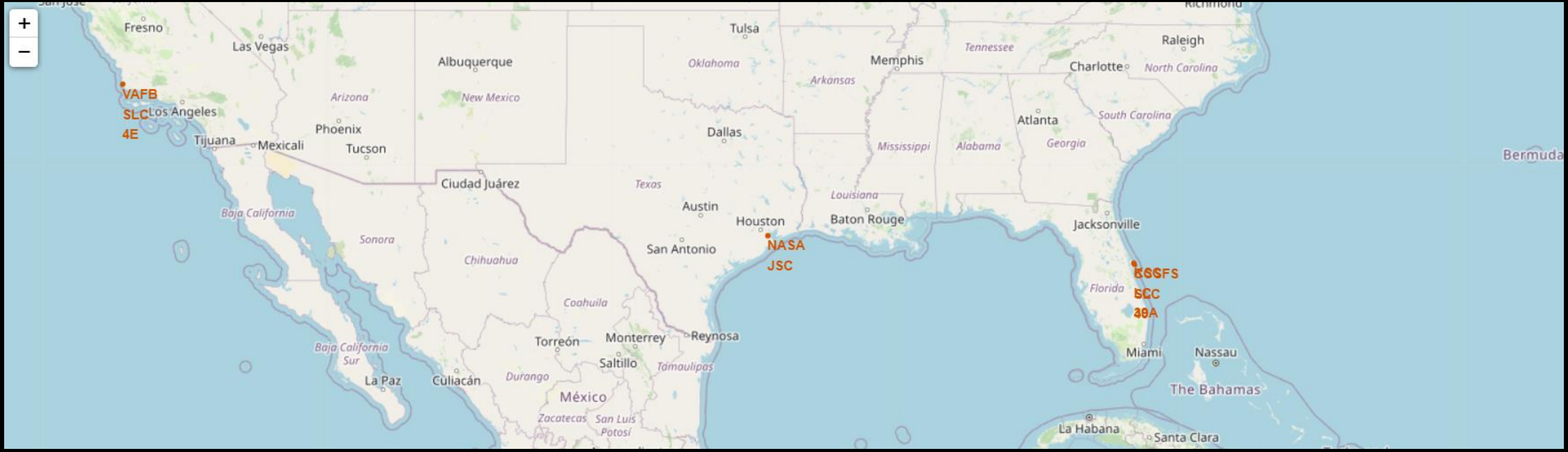
Launch_Site
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40

Total Payload Carried by NASA



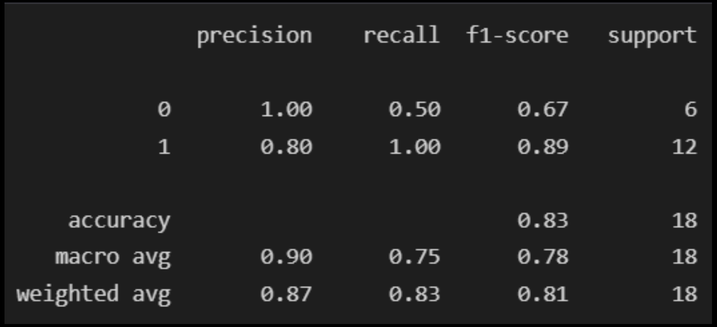
Customer	total payload
NASA (CRS)	45596

Launch site locations



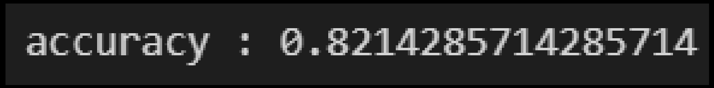
Modelling Results

Logreg Test Accuracy (Best Test Result)



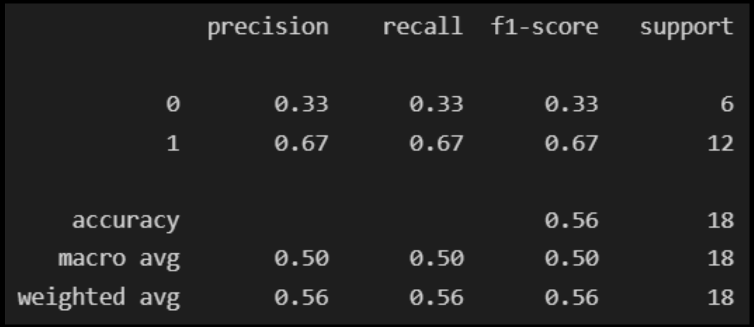
```
              precision    recall  f1-score   support

           0       1.00      0.50      0.67         6
           1       0.80      1.00      0.89        12

    accuracy                           0.83        18
   macro avg       0.90      0.75      0.78        18
weighted avg       0.87      0.83      0.81        18
```

Logreg Train Accuracy (Best fit)



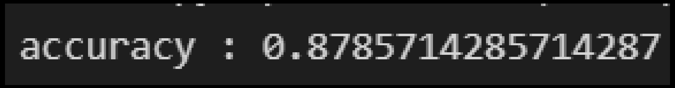
```
accuracy : 0.8214285714285714
```

Decision Tree Test Accuracy (Overfitting)



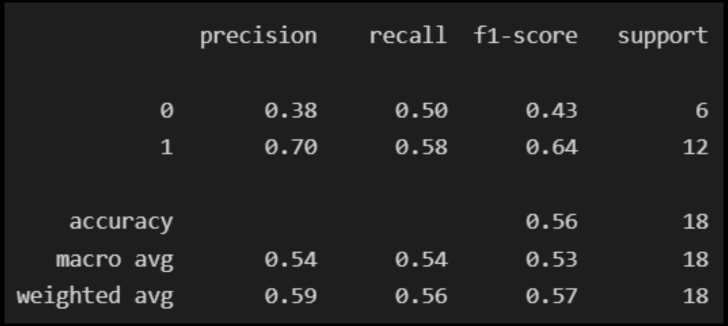
```
              precision    recall  f1-score   support

           0       0.33      0.33      0.33         6
           1       0.67      0.67      0.67        12

    accuracy                           0.56        18
   macro avg       0.50      0.50      0.50        18
weighted avg       0.56      0.56      0.56        18
```

Decision Tree Train Accuracy
(best TRAIN result)



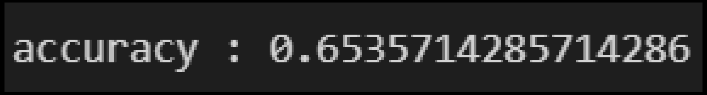
```
accuracy : 0.8785714285714287
```

KNN Test Accuracy (Underfitting)

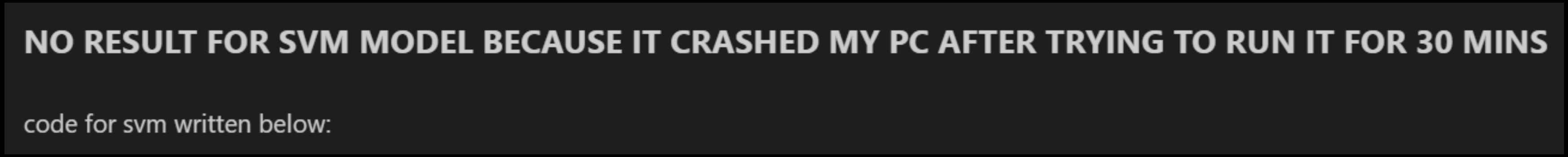


```
              precision    recall  f1-score   support

           0       0.38      0.50      0.43         6
           1       0.70      0.58      0.64        12

    accuracy                           0.56        18
   macro avg       0.54      0.54      0.53        18
weighted avg       0.59      0.56      0.57        18
```

KNN Train Accuracy
(n_neighbors=1 from gridsearch)



```
accuracy : 0.6535714285714286
```



NO RESULT FOR SVM MODEL BECAUSE IT CRASHED MY PC AFTER TRYING TO RUN IT FOR 30 MINS

code for svm written below:

Conclusion

- Success rate keeps increasing over time, as more and more launch is being done.
- Heavier payloads seems to have a better success on landing, however this might happen because of the fact that most of the heavy payloads are launched only after flight number is relatively high (correlates with our 1st point in this conclusion).
- Launch on orbit ES-L1, GEO, HEO, and SSO have 100% success rate.
- We find that Logistic Regression model is best fit for this dataset. (83% accuracy)

Thank you