SpaceX Falcon-9 First Stage Landing Prediction

IBM'S Applied Data Science Capstone Project

Presented by

R.Yaswanth Reddy

Content

Executive Summary

Objective: Predict the success of Falcon 9 first-stage landings to estimate launch costs.

Key Insights:

- SpaceX's reusability reduces launch costs to $62M, while competitors charge $165M.
- Predicting landing outcomes provides a reliable method for cost estimation.

Approach:

- Collected and cleaned data from SpaceX API and external sources.
- Built machine learning models (Logistic Regression, Random Forest, XGBoost).
- Evaluated models based on accuracy, precision, and recall.

Outcome:

- Developed a cost estimation framework and an interactive dashboard for predictions.

Stages of the Capstone Project

Collection of the Data

Data Wrangling

Exploratory Data

EDA by Visualization

Predictive Analysis by Machine Learning

Introduction

- The commercial space industry is growing rapidly, with companies like SpaceX, Blue Origin, and Rocket Lab leading the charge.

- SpaceX has revolutionized launch economics through **reusability**, especially the **Falcon 9 first stage**, reducing costs dramatically.

- Predicting whether the first stage lands successfully helps estimate the **true cost of a launch**.

- In this project, we act as **fictional data scientists at “Space Y”**, building a machine learning model to **predict landing outcomes** based on publicly available data.

Falcon 9 Rocket – Key Facts

- **Developer:** SpaceX (founded by Elon Musk)
- **First Launch:** June 4, 2010
- **Type:** Two-stage orbital launch vehicle
- **Reusability:** First orbital-class rocket capable of reflight, significantly reducing launch costs
- **Payload:** Can carry up to 22,800 kg to low Earth orbit (LEO)
- **Uses:** Satellite deployment, ISS resupply, crewed missions (e.g., Crew Dragon)
- **Engines:** 9 Merlin engines on the first stage, 1 vacuum-optimized Merlin on the second
- **Notable Feature:** First stage booster often lands vertically after launch, on land or drone ships

Methodology

1. Data Collection methodology
    - Require the data from SpaceX API
    - Collect data from a Wikipedia page
2. Data Collection Methodology
    - Perform EDA to find some problems
    - Determine what would be the label for training Supervised Learning
3. Perform exploratory data analysis (EDA) using visualization and SQL
4. Perform interactive visual analytics using Folium and Plotly Dash
5. Perform predictive analysis using classification models

Data Collection

API

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

Web Page

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

Collecting the Data- SpaceX API

- Request and parse the SpaceX launch data using the GET request
- Filter the dataframe to only include Falcon 9 launches

Data Scraping - Wikipedia

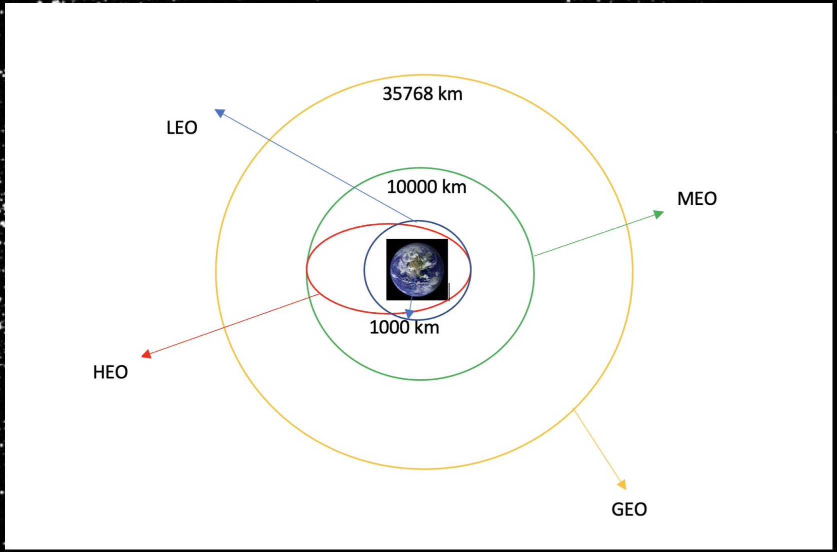
1. Web scraping Falcon 9 and Falcon Heavy Launches Records from Wikipedia
2. Request the Falcon 9 Launch Wiki page from its URL
3. Extract all column/variable names from the HTML table header
4. Create a data frame by parsing the launch HTML tables

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False
5	8	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False
6	10	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False
7	11	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False
8	12	2013-12-03	Falcon 9	3170.0	GTO	CCSFS SLC 40	None None	1	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...
89	102	2020-09-03	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	2	True	True	True
90	103	2020-10-06	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	3	True	True	True
91	104	2020-10-18	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	6	True	True	True
92	105	2020-10-24	Falcon 9	15600.0	VLEO	CCSFS SLC 40	True ASDS	3	True	True	True
93	106	2020-11-05	Falcon 9	3681.0	MEO	CCSFS SLC 40	True ASDS	1	True	False	True

90 rows × 17 columns

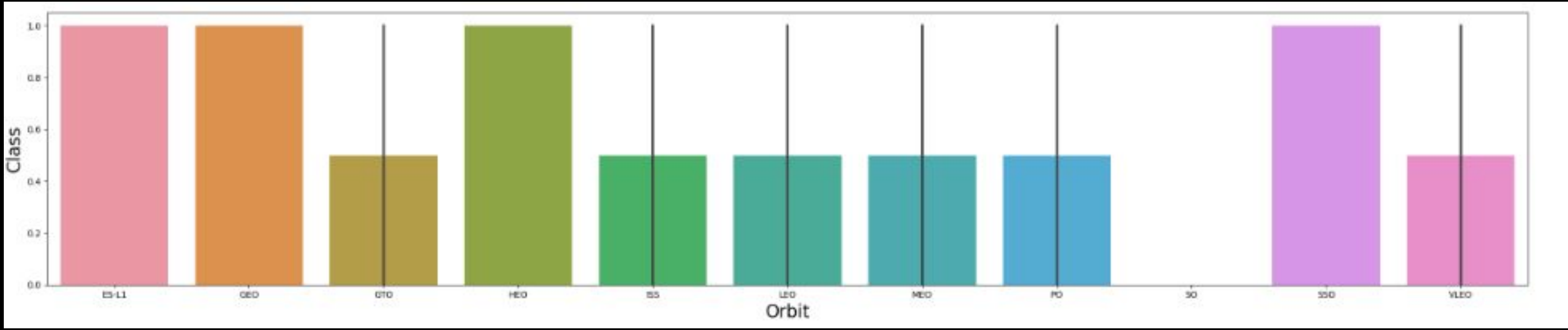
Data Wrangling

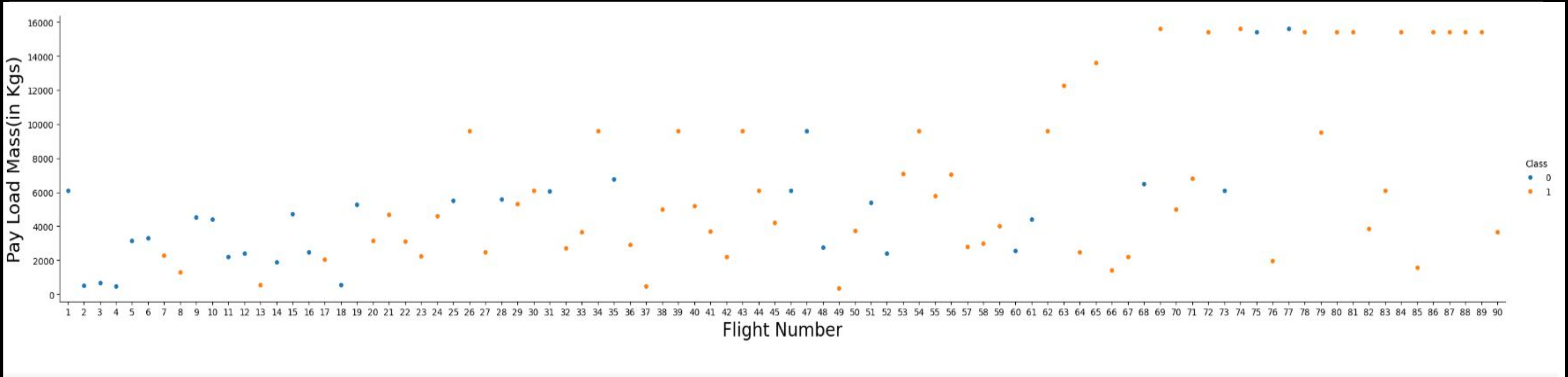
- Calculate the number of launches on each site
- Calculate the number and occurrence of each orbit
- Calculate the number and occurence of mission outcome per orbit type
- Create a landing outcome label from Outcome column



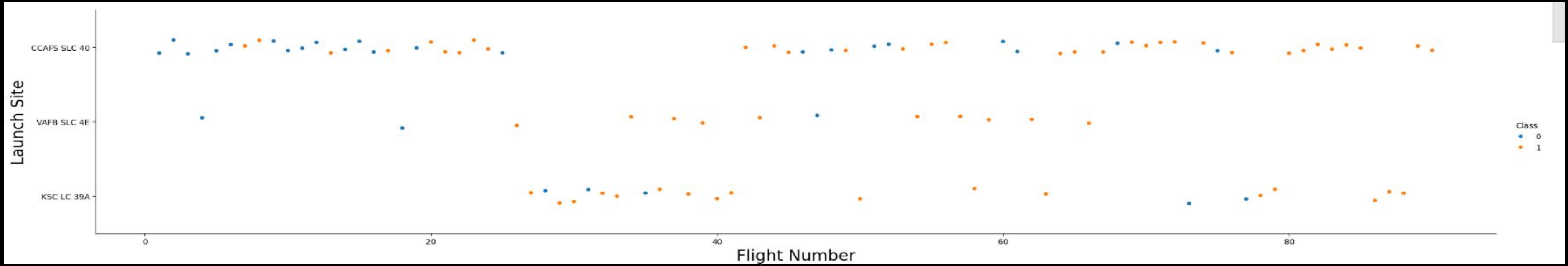
EDA with Visualization

- We can plot out the FlightNumber vs. PayloadMassand overlay theoutcome of the launch
- Visualize the relationship between Flight Number and Launch Site
- Visualize the relationship between Payload and Launch Site
- Visualize the relationship between success rate of each orbit type
- Visualize the relationship between FlightNumber and Orbit type
- Visualize the relationship between Payload and Orbit type
- Visualize the launch success yearly trend

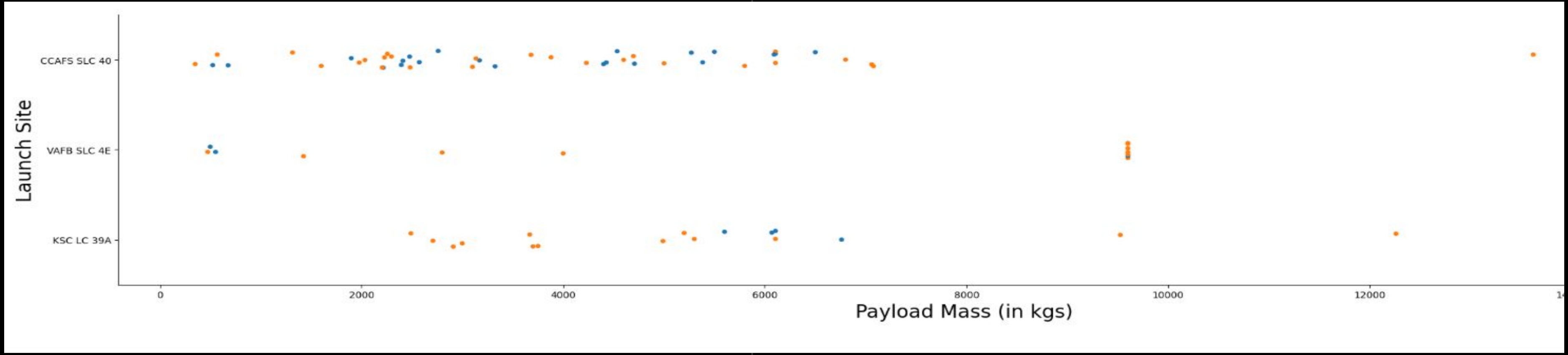




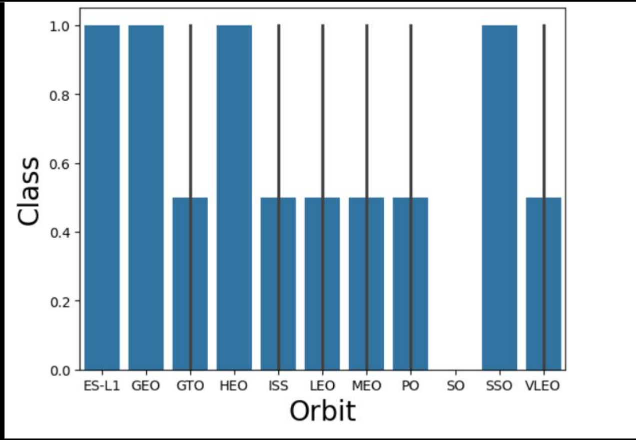
We can observe that the different launch sites have different success rates.
CCAFS LC-4Ø , has a success rate of 60.96, while KSC LC-39A and VAFB SLC 4E has a success rate of 77%.



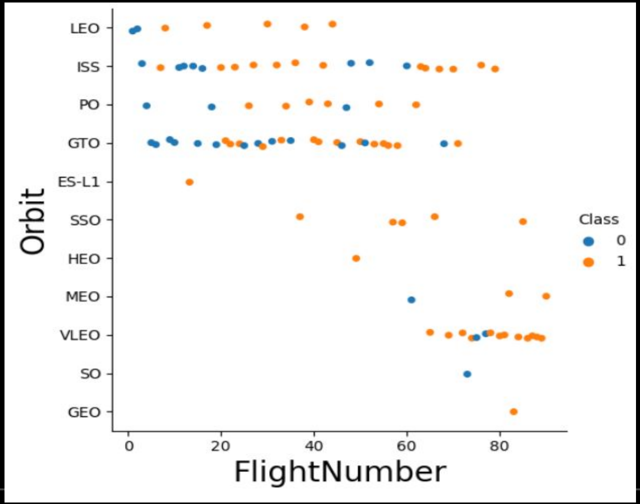
Using the function catplot , we plotted FlightNumber vs LaunchSite, set the parameter x parameter to FlightNumber, set the y to Launch Site and set the parameter hue to 'class'



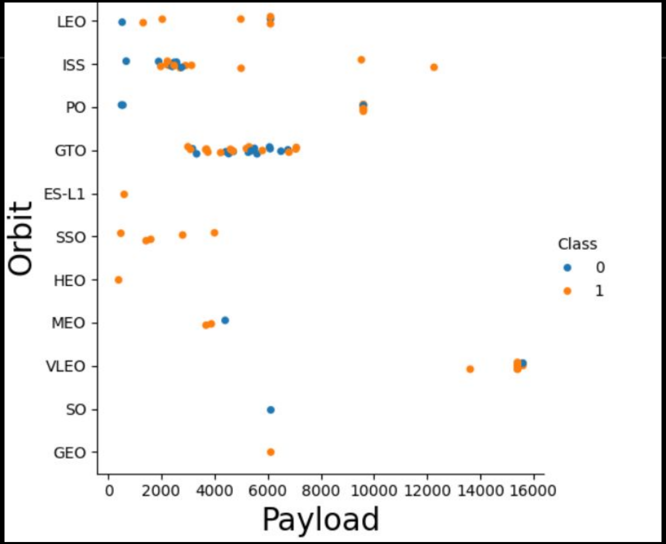
Relationship between launch sites and their payload mass



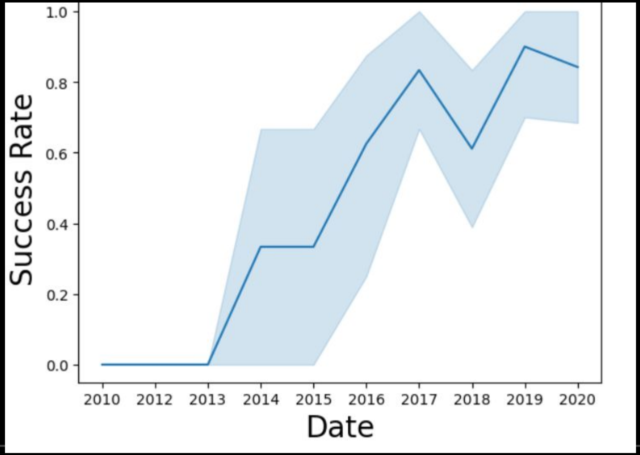
Relationship between success rate of each orbit type



Relationship between FlightNumber and Orbit type



Relationship between Payload and Orbit type



Launch success yearly trend

EDA With SQL

```
%sql select distinct(LAUNCH_SITE) from SPACEXTBL
```

* sqlite:///my_data1.db
Done.

[7]:

Launch_Site
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40
None

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

```
%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
06/04/2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0.0	LEO	SpaceX	Success	Failure (parachute)
12/08/2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0.0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
22/05/2012	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525.0	LEO (ISS)	NASA (COTS)	Success	No attempt
10/08/2012	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500.0	LEO (ISS)	NASA (CRS)	Success	No attempt
03/01/2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677.0	LEO (ISS)	NASA (CRS)	Success	No attempt

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

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

```
%sql select sum(PAYLOAD_MASS__KG_) from SPACEXTBL where CUSTOMER = 'NASA (CRS)'
```

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

sum(PAYLOAD_MASS__KG_)
45596.0

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

```
[10]:
%sql select avg(PAYLOAD_MASS__KG_) from SPACEXTBL where BOOSTER_VERSION = 'F9 v1.1'
```

```
 * sqlite:///my_data1.db
Done.
[10]:
```

avg(PAYLOAD_MASS__KG_)
2928.4

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)
01/08/2018

Task 6

List the 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 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

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

```
%sql select Booster_Version from SPACEXTBL where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTBL)
```

```
 * 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

Task 9

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.

```
%sql SELECT SUBSTR(Date,4,2) AS Month, Booster_Version, Launch_site FROM SPACEXTBL WHERE Landing_Outcome LIKE 'Failure%drone%' AND SUBSTR(Date,7,4) = '20
```

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

Month	Booster_Version	Launch_Site
10	F9 v1.1 B1012	CCAFS LC-40
04	F9 v1.1 B1015	CCAFS LC-40

Task 10

Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

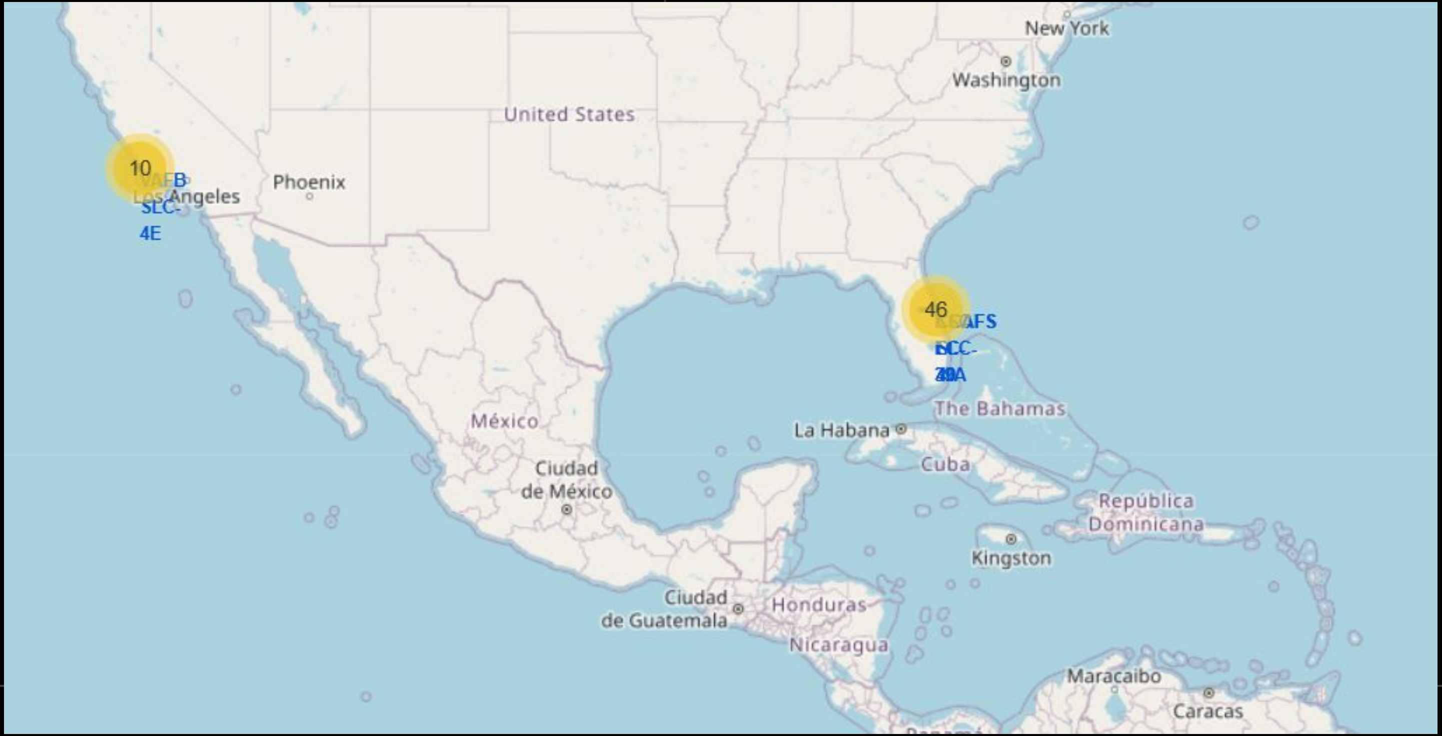
```
%sql SELECT Landing_Outcome, COUNT(*) AS Numbers FROM SPACEXTBL WHERE Landing_Outcome LIKE 'Success%' AND Date BETWEEN '04-06-2010' AND '20-03-2017' GROUI
```

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

Landing_Outcome	Numbers
Success	20

Interactive Visual Analytics with Folium

- Mark all launch sites on a map
- Mark the success/failed launches for each site on the map
- Calculate the distances between a launch site to its proximities



-

Machine Learning Prediction

```
Y = data["Class"].to_numpy()
Y
```

```
array([0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1,
       1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1])
```

```
transform=preprocessing.StandardScaler()
X=transform.fit_transform(X)
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
print ('Train set:', X_train.shape,  Y_train.shape)
print ('Test set:', X_test.shape,  Y_test.shape)
```

```
Train set: (72, 83) (72,)
Test set: (18, 83) (18,)
```

```
Y_test.shape
```

```
(18,)
```

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

```python
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
lr=LogisticRegression()
logreg_cv = GridSearchCV(lr, param_grid=parameters,scoring='accuracy', cv=10)
logreg_cv.fit(X_train, Y_train)
logreg_cv.best_params_
```

```
{'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
```

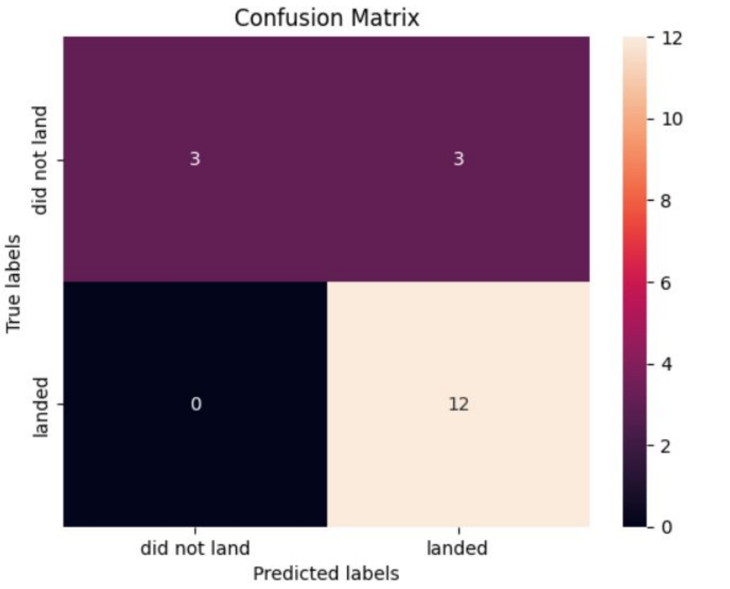
```python
print("tuned hyperparameters:(best parameters)", logreg_cv.best_params_)
print("accuracy:", logreg_cv.best_score_)
```

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

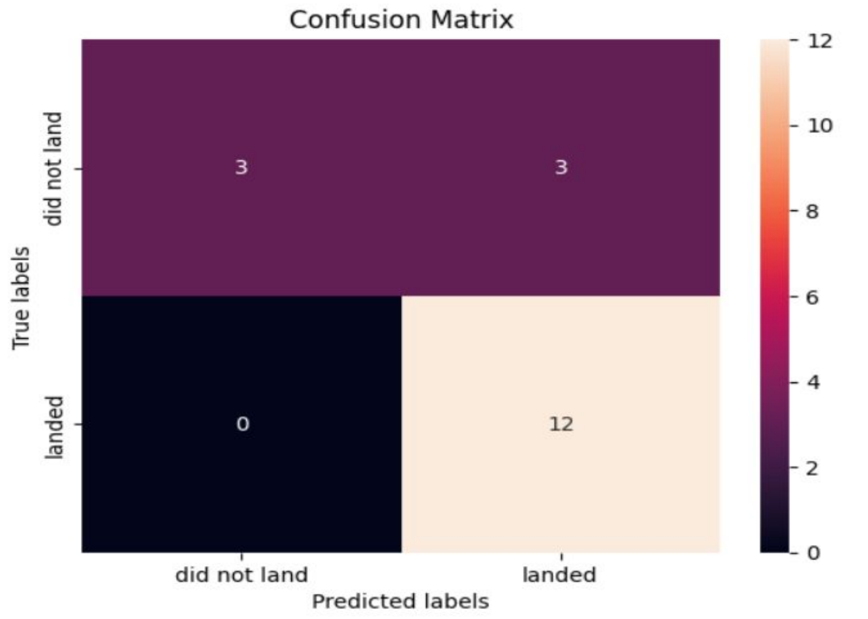
```python
logreg_cv.score(X_test,Y_test)
```

```
0.8333333333333334
```

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




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



```
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, param_grid=parameters,scoring='accuracy', cv=10)
svm_cv.fit(X_train, Y_train)
svm_cv.best_params_
```

```
{'C': np.float64(1.0),
 'gamma': np.float64(0.03162277660168379),
 'kernel': 'sigmoid'}
```

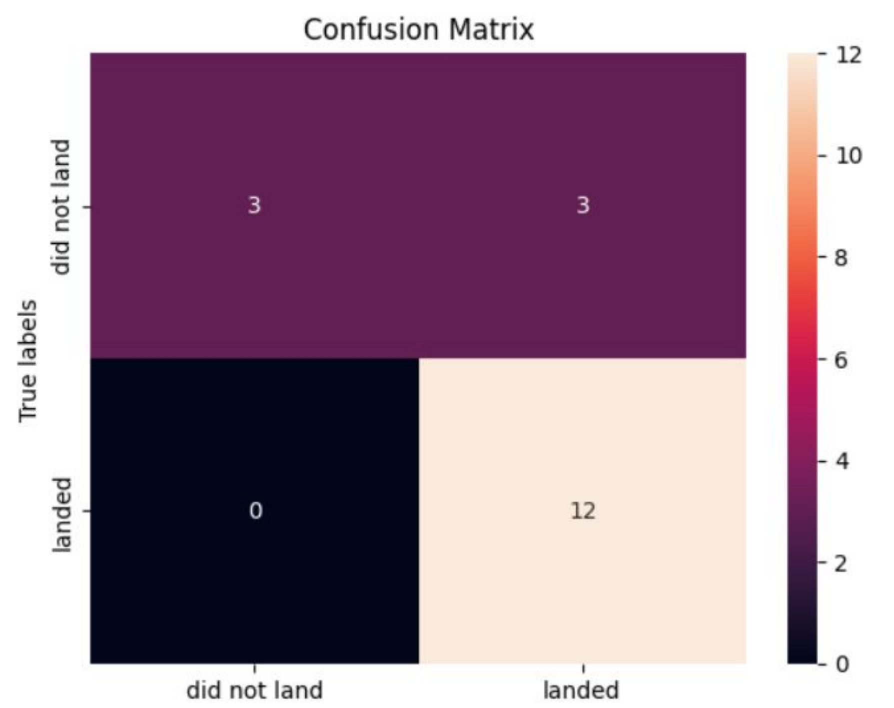
```
print("tuned hyperparameters:(best parameters)",svm_cv.best_params_)
print("accuracy:",svm_cv.best_score_)
```

```
tuned hyperparameters:(best parameters) {'C': np.float64(1.0), 'gamma': np.float64(0.03162277660168379), 'kernel': 'sigmoid'}
accuracy: 0.8482142857142856
```

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

```
0.8333333333333334
```

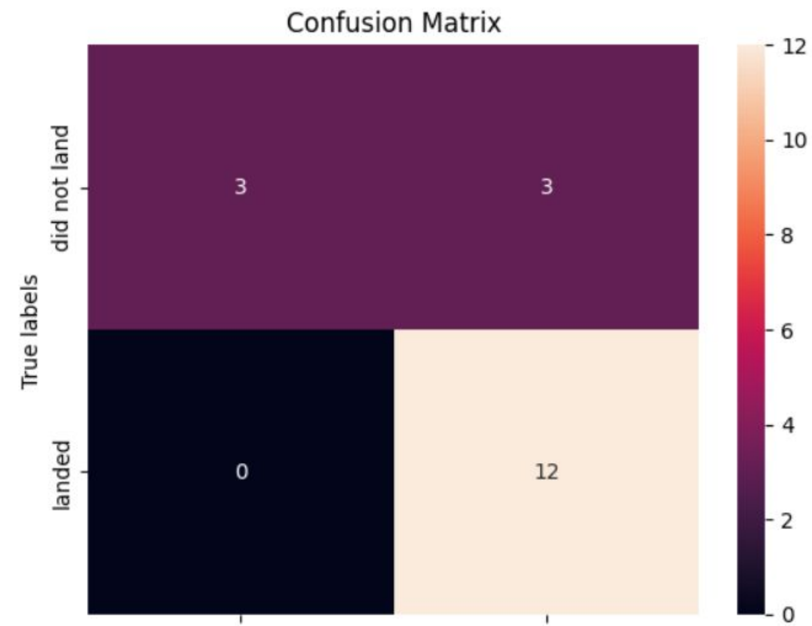
```
yhat=tree_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```




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

0.8333333333333334

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



Appendix Resources

https://github.com/Yaswanthramireddy18/falcon9-landing-prediction-ds

THANK YOU!