IBM DATA SCIENCE CAPSTONE PROJECT

Christopher Atkinson

06/02/2023

IBM Developer

SKILLS NETWORK

OUTLINE

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

EXECUTIVE SUMMARY

- Summary of methodologies
  - Data collection
  - Data wrangling
  - EDA with data visualization
  - EDA with SQL
  - Interactive mapping with folium
  - Creating dashboards with Plotly Dash
  - Predictive analysis with machine learning
- Summary of results
  - EDA results
  - Interactive maps and Dashboard
  - Machine learning results

INTRODUCTION

- Project background and context

  SpaceX is one of the most successful space exploration companies in the world, one of the reasons for that is that they advertise Falcon 9 rocket launches on their website with a cost of 62 million dollars; other providers cost upwards 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. Spaces X's Falcon 9 launch like regular rockets.

- Problems to be answered
  - How do things like payload mass, launch site, and orbits affect the success of a first stage landing?
  - Does the success rate change from year to year?
  - What is the best algorithm to be used for binary classification in our case?

METHODOLOGY

- Data collection methodology
  - The SpaceX rest API
  - Web scraping Wikipedia
- Data Wrangling methodology
  - Filtering data
  - Dropping or modifying missing values
  - Using one hot encoding for classification
- Perform EDA using visualization and SQL
- Perform interactive visual analytics using classification models

DATA COLLECTION

- Data was collected from the SpaceX rest API and from Wikipedia using data scraping
  - The columns from the Wikipedia web scraping are as follows:
    - Flight No., Launch Site, Payload, PayloadMass, Orbit, Customer, Launch, Outcome, Version Booster, Booster Landing, Date, Time.
  - The columns for the rest API are as follows:
    - FlightNumber, Date, BoosterVersion, PayloadMass, Orbit, LaunchSite, Outcome, Flights, GridFins, Reused, Legs, LandingPad, Block, ReusedCount, Serial, Longitude, Latitude.

DATA COLLECTION – SPACEX REST API

- The steps to collect data from SpaceX's free rest API are:
  - Request API and parse for launch data
  - Filter data down to just the falcon 9 launches
  - Replace missing values to fill out the dataset
  - Url for more detail on the data collection process https://github.com/chrisatkinson16/IBM-Applied-Data-Science-Capstone/blob/main/jupyter-labs-spacex-data-collection-api-final.ipynb

DATA COLLECTION – WIKIPEDIA

- Wikipedia also has some data on SpaceX launches that might be useful so to get that data in a format we need, we use web scraping
  - Request data from Wikipedia
  - Create a beautiful soup object out of the HTML response
  - Extract column names and parse the table to collect the data
  - Build a dictionary using the collected data and then creating a data frame from that
  - Export it to a .csv file
  - https://github.com/chrisatkinson16/IBM-Applied-Data-Science-Capstone/blob/main/jupyter-labs-webscraping-final.ipynb

DATA WRANGLING

- In the dataset several of the cases where the booster did not land successfully, this is where EDA comes in and we analyse the dataset to convert failures and successes into training labels 1 for success and 0 for failure
- https://github.com/chrisatkinson16/IBM-Applied-Data-Science-Capstone/blob/main/labs-jupyter-spacex-Data%20wrangling-final.ipynb

EDA WITH DATA VISUALIZATION

- Plotting charts for:
  - Flight number vs payload mass, flight number vs launch site, payload mass vs launch site, orbit vs success rate, flight number vs orbit, payload vs orbit, and the success rate yearly trend
- we get a picture of how these variables come into play and have an affect on the success or failure of a launch
- https://github.com/chrisatkinson16/IBM-Applied-Data-Science-Capstone/blob/main/jupyter-labs-eda-dataviz-final.ipynb

EDA WITH SQL

- Using SQL we can look at certain subsets of the data to see if we can spot any patterns in the data that might lead to success or failure. Queries such as:
  - Displaying the names of unique launch sites
  - Displaying the total payload mass carried by NASA (CRS) boosters
  - Listing the total number of successful and failed missions
  - Etc.
- the full list of commands used and their outputs can be seen here https://github.com/chrisatkinson16/IBM-Applied-Data-Science-Capstone/blob/main/jupyter-labs-eda-sql-coursera-final.ipynb

INTERACTIVE MAPS WITH FOLIUM

- Using folium it is possible to create interactive maps that show data such as:
  - Markers where every launch happened
  - Highlights to tell if the launch succeeded or not
  - Lines to indicate the distance between two coordinates
- https://github.com/chrisatkinson16/IBM-Applied-Data-Science-Capstone/blob/main/lab_jupyter_launch_site_location_final.ipynb

DASHBOARDS WITH PLOTLY DASH

- The dashboard has dropdowns, pie charts, range sliders, and scatter plots.
  - The dropdown allows the user to choose a launch site or all of them
  - The pie charts show the total success and failure rate for the launch site chosen
  - The range slider allows for a payload mass to be selected
  - The scatter plot shows the relationship between success and payload mass
- https://github.com/chrisatkinson16/IBM-Applied-Data-Science-Capstone/blob/main/dash_interactivity.py

PREDICTIVE ANALYSIS (ML)

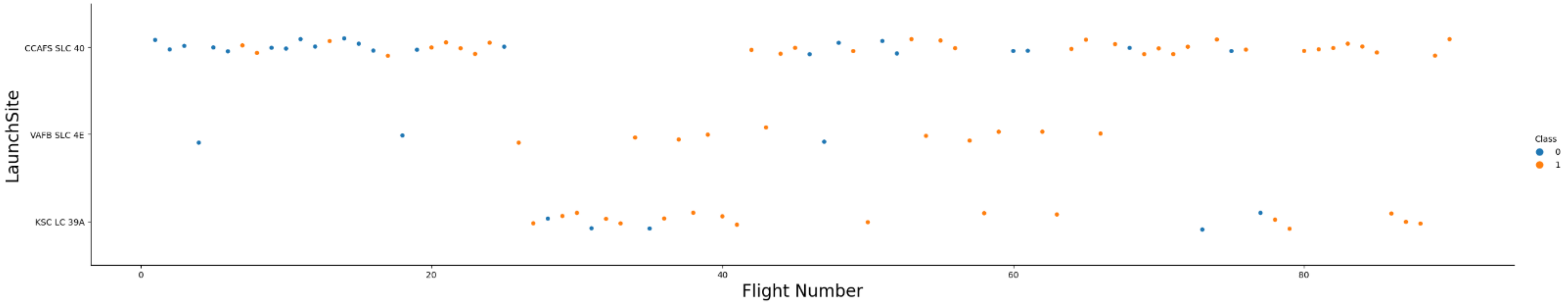
- Prepare the data (load, normalize, split into train and test sets)
- Prepare the model (select ML algorithm, set parameters, train models)
- Evaluate the model (get the best hyperparameters, compute accuracy, plot confusion matrix)
- Comparison (compare the accuracy of models, chose the one with the best accuracy)
- https://github.com/chrisatkinson16/IBM-Applied-Data-Science-Capstone/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5_final.ipynb

RESULTS

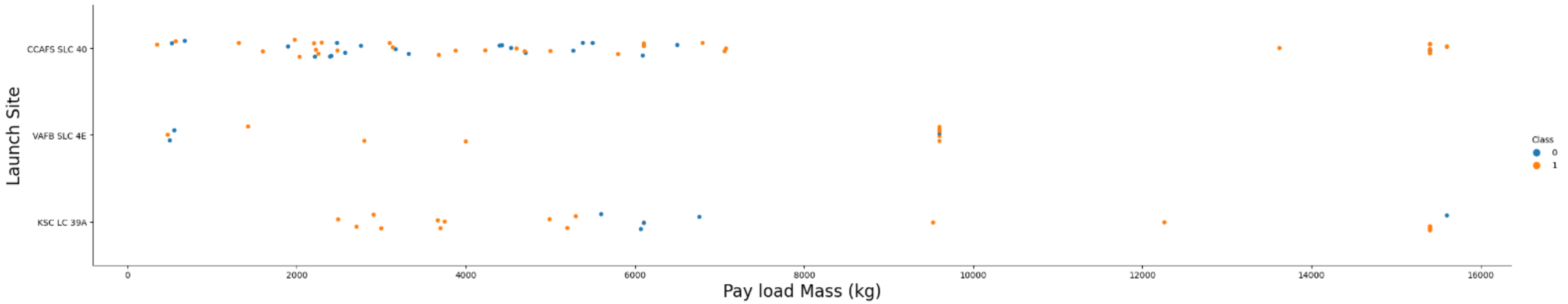
- EDA results
- Interactive analysis results
- Predictive analysis results

EDA WITH VISUALIZATION

FLIGHT NUMBER VS LAUNCH SITE



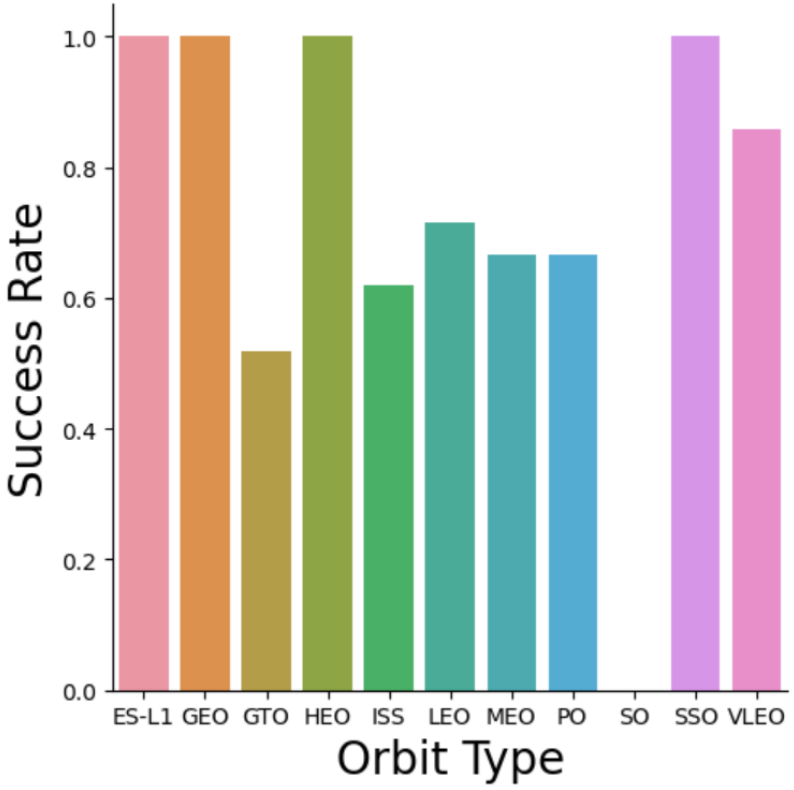
PAYLOAD MASS VS LAUNCH SITE



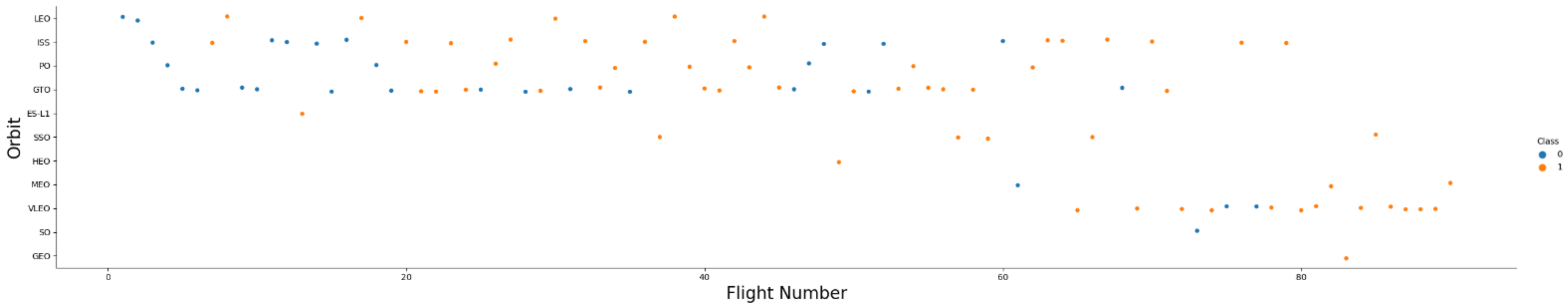
IBM Developer

SKILLS NETWORK

SUCCESS RATE VS ORBIT TYPE



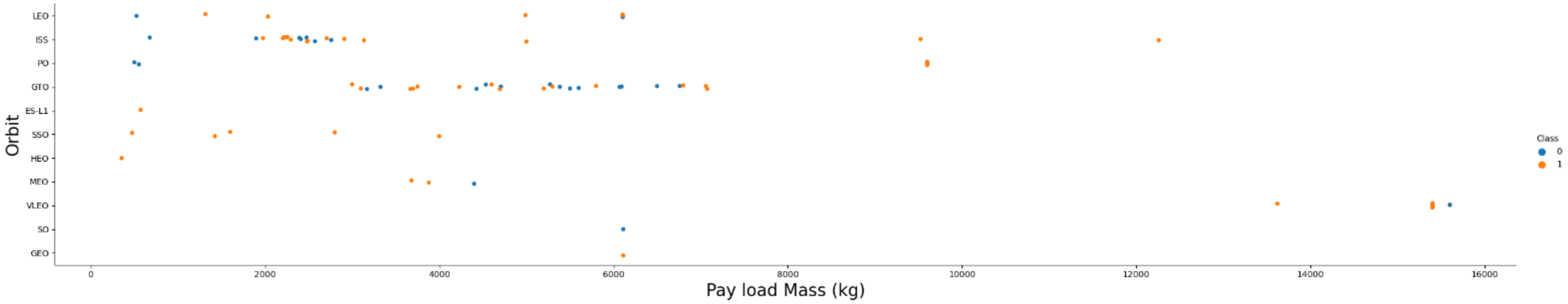
FLIGHT NUMBER VS ORBIT



IBM Developer

SKILLS NETWORK

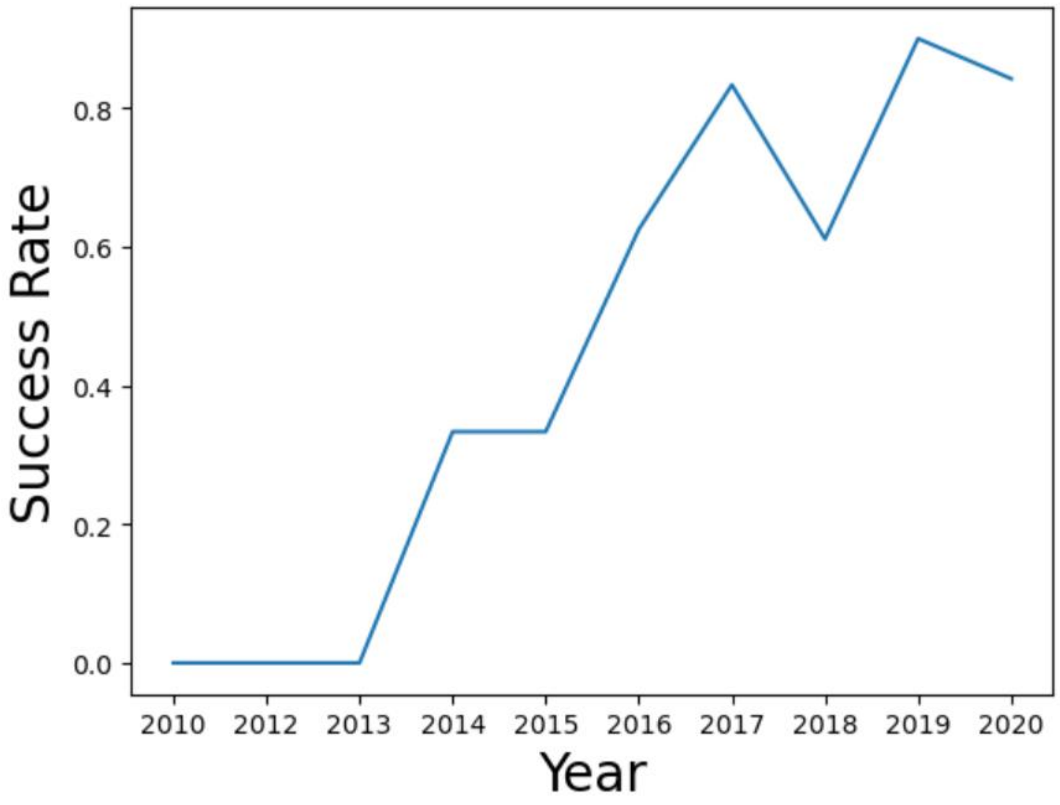
PAYLOAD MASS VS ORBIT



IBM Developer

SKILLS NETWORK

LAUNCH SUCCESS YEARLY TREND



EDA WITH SQL

LAUNCH SITE NAMES

```
%sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL ORDER BY 1;
```

```
 * db2://qsj97991:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30426/bludb
Done.
```

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

LAUNCH SITE NAMES THAT BEGIN WITH 'CCA'

```
sql SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```

```
 * db2://qsj97991:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30426/bludb
Done.
```

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

PAYLOAD MASS CARRIED BY NASA (CRS) BOOSTERS

```
sql SELECT SUM(PAYLOAD_MASS__KG_) AS TOTAL_PAYLOAD_MASS FROM SPACEXTBL WHERE CUSTOMER LIKE '%NASA (CRS)%';
```

```
 * db2://qsj97991:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30426/bludb
Done.
```

total_payload_mass
48213

AVERAGE PAYLOAD MASS CARRIED BY BOOSTER VERSION 'F9 V1.1'

```
sql SELECT AVG(PAYLOAD_MASS__KG_) AS AVERAGE_PAYLOAD_MASS FROM SPACEXTBL WHERE BOOSTER_VERSION LIKE '%F9 v1.1%';
```

```
 * db2://qsj97991:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30426/bludb
Done.
```

average_payload_mass
2534

DATE OF THE FIRST SUCCESSFUL LANDING USING A GROUND PAD

```
sql SELECT min(DATE) AS FIRST_SUCCESSFUL_LANDING FROM SPACEXTBL WHERE LANDING__OUTCOME = 'Success (ground pad)';
```

```
 * db2://qsj97991:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30426/bludb
Done.
```

first_successful_landing
2015-12-22

NAMES OF THE BOOSTERS WHO HAVE SUCCESS IN DRONE SHIP AND HAVE PAYLOAD MASS BETWEEN 4000 AND 6000 KG

```
sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000 AND LANDING__OUTCOME = 'Success (drone ship)'
```

```
 * db2://qsj97991:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30426/bludb
Done.
```

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

TOTAL NUMBER OF SUCCESSFUL AND FAILED MISSIONS

```
sql SELECT MISSION_OUTCOME, COUNT(*) AS COUNT FROM SPACEXTBL GROUP BY MISSION_OUTCOME;
```

```
 * db2://qsj97991:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30426/bludb
Done.
```

mission_outcome	COUNT
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

NAMES OF BOOSTERS WHICH HAVE CARRIED THE MAXIMUM PAYLOAD

```
sql SELECT BOOSTER_VERSION from SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT max(PAYLOAD_MASS__KG_) from SPACEXTBL);
```

```
 * db2://qsj97991:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30426/bludb
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

FAILED LANDINGS BETWEEN 2010-06-04 AND 2017-03-20, IN DESCENDING ORDER

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

```
sql SELECT LANDING__OUTCOME, COUNT(*) AS COUNT FROM SPACEXTBL WHERE
```

INTERACTIVE MAPS WITH FOLIUM

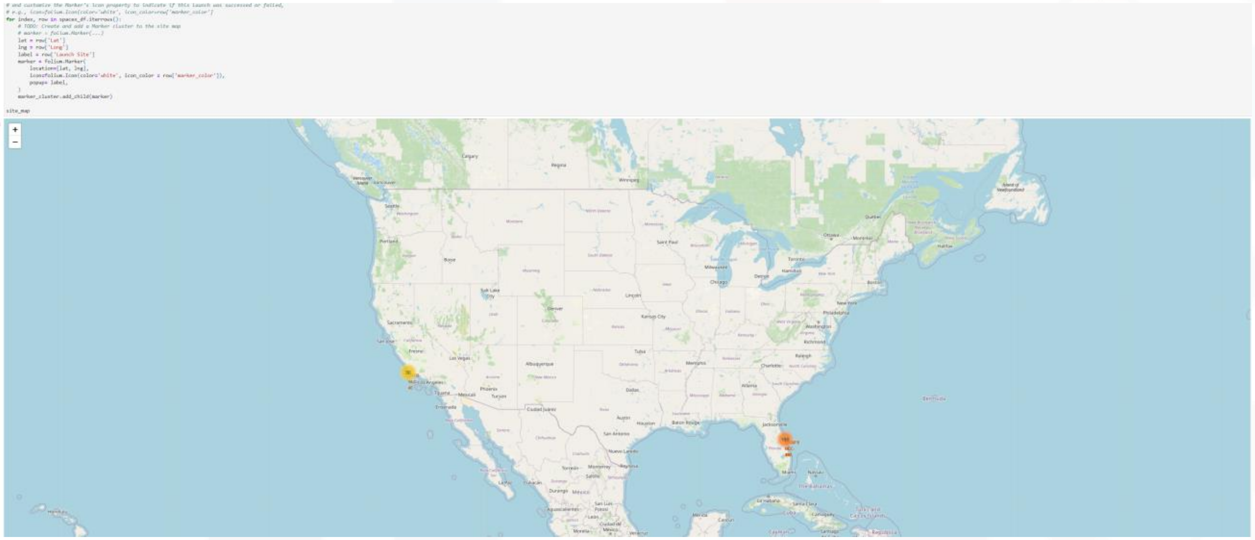
MAP OF ALL LAUNCH SITES

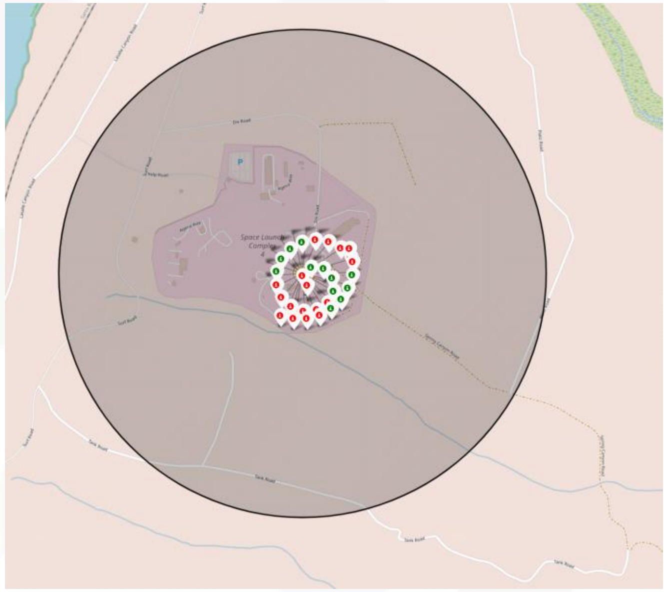
```python
# Initial the map
site_map = folium.Map(location=nasa_coordinate, zoom_start=5)
# For each launch site, add a Circle object based on its coordinate (Lat, Long) values. In addition, add Launch site name as a popup label
for ix, row in launch_sites_df.iterrows():
    ls_name = row['Launch Site']
    ls_lat  = row['Lat']
    ls_long = row['Long']

    #print('{}\t{}\t{}'.format(ls_name, ls_lat, ls_long))

    coordinate = [ls_lat, ls_long]
    circle = folium.Circle(coordinate, radius=1000, color='#000000', fill=True).add_child(folium.Popup(ls_name + ' region'))
    marker = folium.map.Marker(coordinate, icon=DivIcon(icon_size=(20,20),icon_anchor=(0,0), html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % ls_name, ))

    site_map.add_child(circle)
    site_map.add_child(marker)

site_map
```

IBM Developer

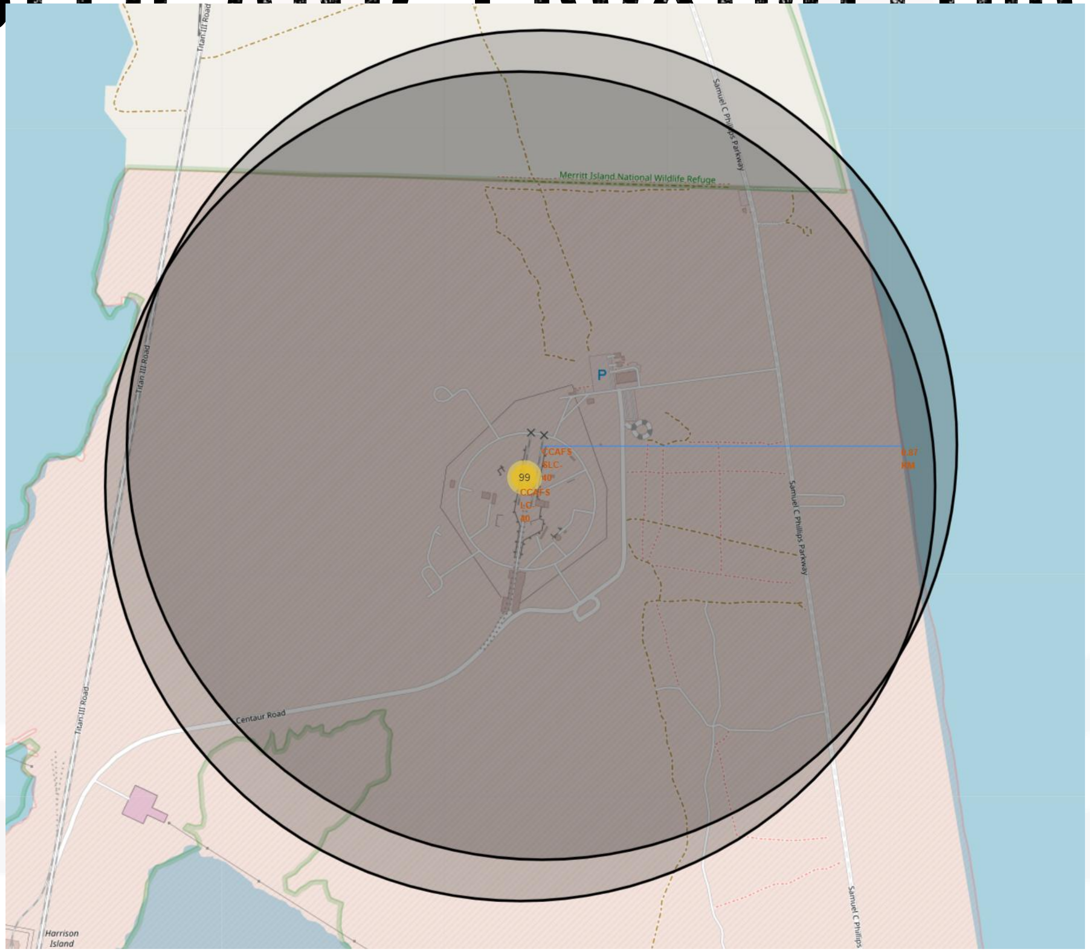
SKILLS NETWORK

MAP WITH ALL FAILED AND SUCCEEDED LAUNCHES MARKED



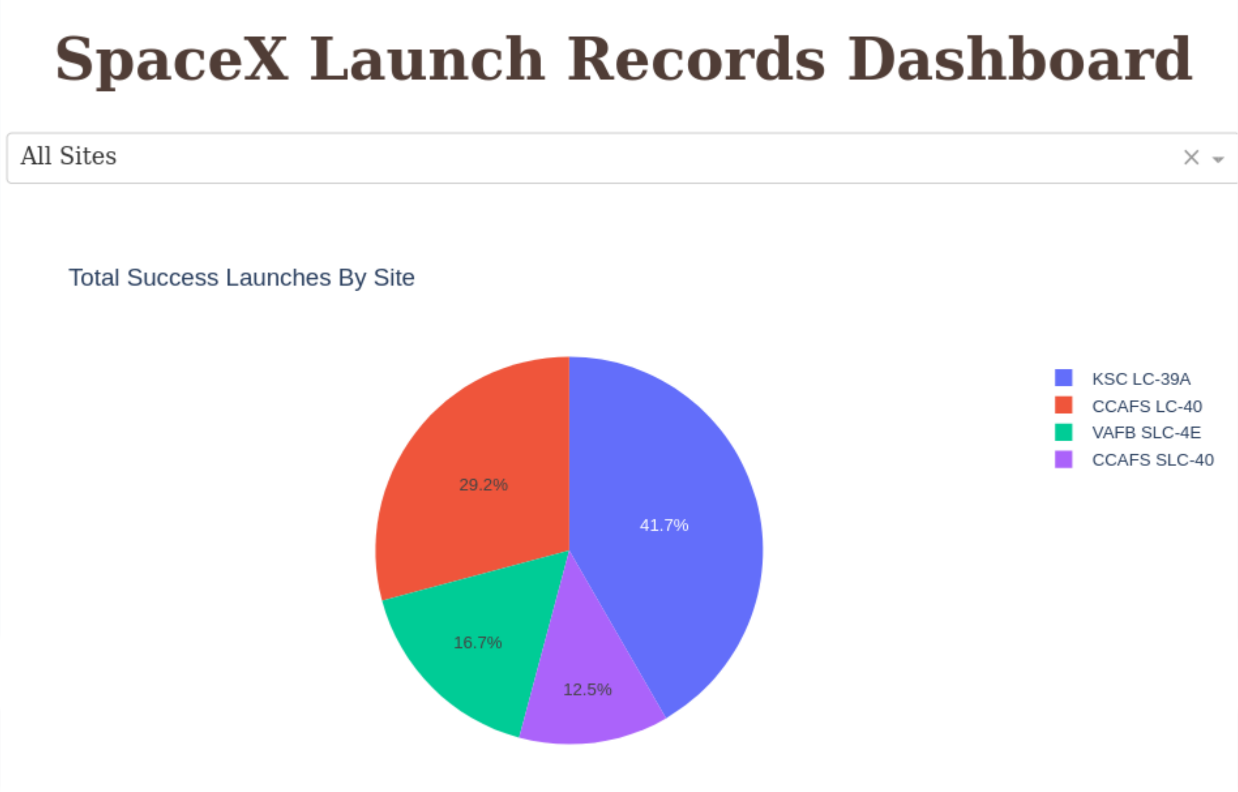


MAP WITH THE DISTANCE BETWEEN LAUNCH SITE AND PROXIMITIES



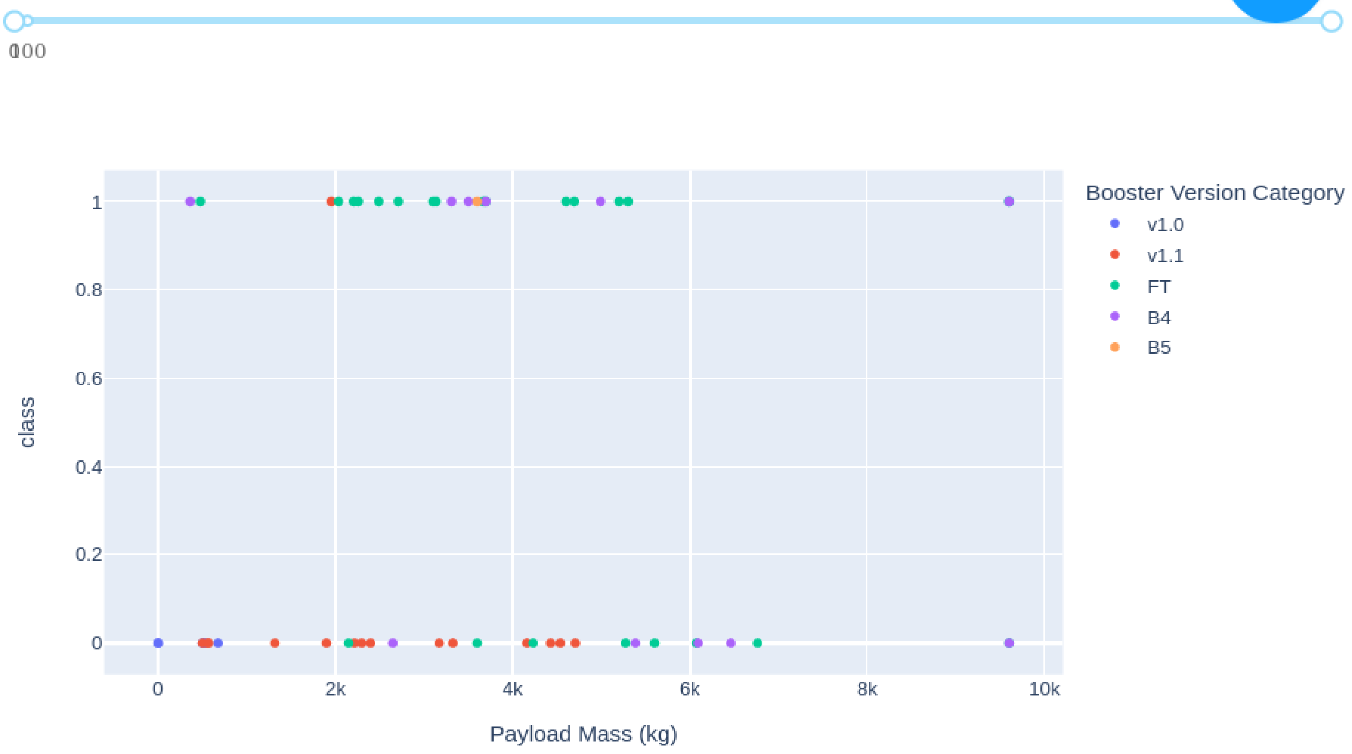
PLOTLY DASH APP

PIE CHART FOR ALL SITES

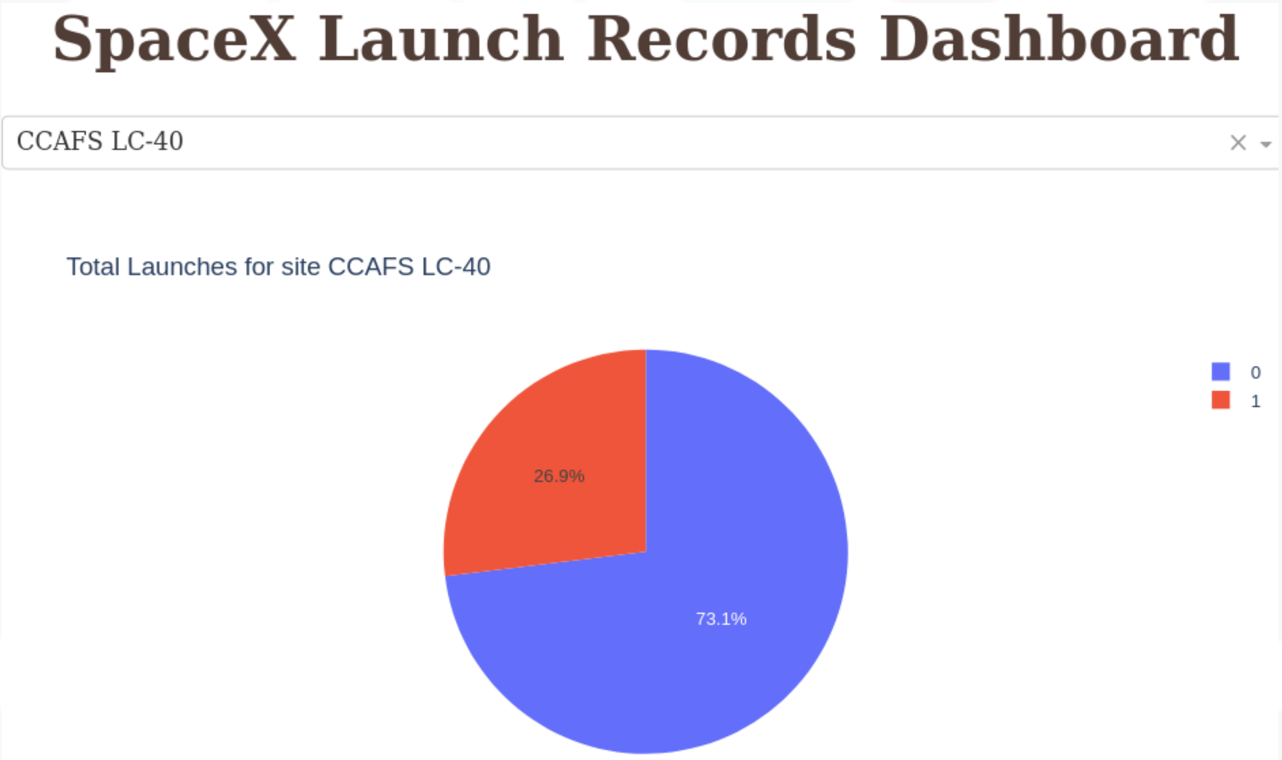


SCATTER PLOT OF THE PAYLOAD MASS WITH SLIDER

Payload range (Kg):



PIE CHART FOR CCAFS LC-40



PREDICTIVE ANALYSIS

TASK 1

Create a NumPy array from the column `Class` in `data`, by applying the method `to_numpy()` then assign it to the variable `Y` ,make sure the output is a Pandas series (only one bracket df['name of column']).

```
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])
```

TASK 2

Standardize the data in `X` then reassign it to the variable `X` using the transform provided below.

```
# students get this
transform = preprocessing.StandardScaler()
```

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

We split the data into training and testing data using the function `train_test_split`. The training data is divided into validation data, a second set used for training data; then the models are trained and hyperparameters are selected using the function `GridSearchCV`.

TASK 3

Use the function train_test_split to split the data X and Y into training and test data. Set the parameter test_size to 0.2 and random_state to 2. The training data and test data should be assigned to the following labels.

`X_train, X_test, Y_train, Y_test`

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
```

we can see we only have 18 test samples.

```
Y_test.shape
```

```
(18,)
```

TASK 4

Create a logistic regression object then create a GridSearchCV object `logreg_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

```
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, 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 a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

We output the `GridSearchCV` object for logistic regression. We display the best parameters using the data attribute `best_params_` and the accuracy on the validation data using the data attribute `best_score_`.

```
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
```

TASK 5

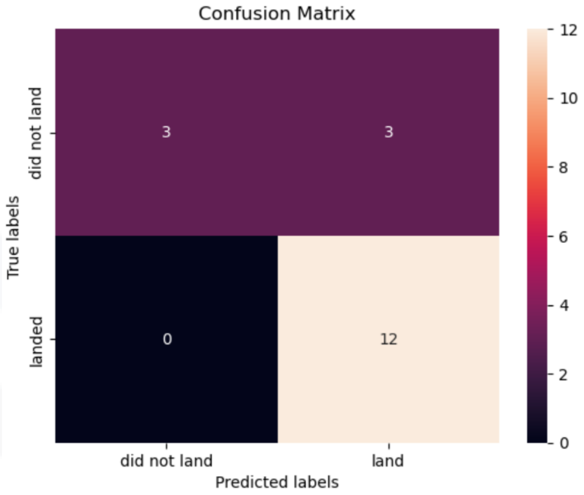
Calculate the accuracy on the test data using the method `score` :

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

```
0.8333333333333334
```

Lets look at the confusion matrix:

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



TASK 12

Find the method performs best:

```
from sklearn.metrics import jaccard_score, f1_score
jaccard_scores = [
                  jaccard_score(Y, logreg_cv.predict(X), average='binary'),
                  jaccard_score(Y, svm_cv.predict(X), average='binary'),
                  jaccard_score(Y, tree_cv.predict(X), average='binary'),
                  jaccard_score(Y, knn_cv.predict(X), average='binary'),
                 ]

f1_scores = [
             f1_score(Y, logreg_cv.predict(X), average='binary'),
             f1_score(Y, svm_cv.predict(X), average='binary'),
             f1_score(Y, tree_cv.predict(X), average='binary'),
             f1_score(Y, knn_cv.predict(X), average='binary'),
            ]

accuracy = [logreg_cv.score(X, Y), svm_cv.score(X, Y), tree_cv.score(X, Y), knn_cv.score(X, Y)]

scores = pd.DataFrame(np.array([jaccard_scores, f1_scores, accuracy]),
                      index=['Jaccard_Score', 'F1_Score', 'Accuracy'],
                      columns=['LogReg', 'SVM', 'Tree', 'KNN'])
scores
```

	LogReg	SVM	Tree	KNN
Jaccard_Score	0.833333	0.845070	0.835821	0.819444
F1_Score	0.909091	0.916031	0.910569	0.900763
Accuracy	0.866667	0.877778	0.877778	0.855556

CONCLUSION

- In conclusion, by analyzing this data and manipulating it with various methods we can see which parameters and other affects can lead to success and failure in a SpaceX launch