Title

Hassan Mohamed

OUTLINE

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

EXECUTIVE SUMMARY

- SpaceX cost benefit: $60 million vs. NASA
- The success of landing depends on payload and orbit
  - Sub Point 1
  - Sub Point 2
  - Sub Point 3
- Chances of mission success is 99%
- Point4
- Point5

INTRODUCTION

- SpaceX cost benefit: $60 million vs. $165 million for other competitors
- Savings due to recovery of stage 1, therefore reducing materials cost
- Cost of launch is dependent on our ability to determine location of stage 1
- The success of landing depends on payload and orbit
  - Sub Point 1
  - Sub Point 2
  - Sub Point 3
- Chances of mission success is ~46%

  - Sub Point1
  - Sub Point2

AIM

- Our aim is to: get insight into the data to determine which factors lead to the success of the launch

- What did you want to achieve with this project? Can you break the aim into sections?

METHODOLOGY

- Point1
- Point2
- Point3
- Point4
  - Sub Point1
  - Sub Point2

Data Collection

- Request to the SpaceX API
- Clean the requested data
- **Task 1: Request and parse the SpaceX launch data using the GET request**

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

- The API response was parsed as a JSON object using the .json() function and transformed into a pandas dataframe with .json_normalize().
- Data cleaning was performed, including checking for and handling missing values.
- Additionally, web scraping was conducted using BeautifulSoup to collect Falcon 9 launch records from Wikipedia.
- The goal was to extract launch records from an HTML table, parse the content, and convert it into a pandas dataframe for further analysis.

Data collection Notebook

- **SpaceX Falcon 9 first stage Landing Prediction**
- https://github.com/kobosh/ibmdatasciencecapstoneproject/blob/master/jupyter-labs-spacex-data-collection-api%20(1).ipynb

Data Wrangling

- SpaceX API
  - Website has all the data for all the rockets but we will choose SpaceX F9, this type of rocket and it will be single payload ...

```
# Create a data from launch_dict
print(launch_dict)
df=pd.DataFrame(launch_dict)
```

```
{'FlightNumber': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 3
```

Show the summary of the dataframe

```
# Show the head of the dataframe
df=pd.DataFrame(launch_dict)
```

Data Wrangling

- Task 2: Filter the dataframe to only include Falcon 9 launches

```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

```
data_falcon9.isnull().sum()
```

Data Wrangling

- Task 3: Dealing with Missing Values

```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data Wrangling Notebook

- https://github.com/kobosh/ibmdatasciencecapstoneproject/blob/master/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA Sqlite Notebook

- **Overview of the DataSet**
- SpaceX has gained worldwide attention for a series of historic milestones.
- It is the only private company ever to return a spacecraft from low-earth orbit, which it first accomplished in December 2010. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars wheras other providers cost upward of 165 million dollars each, much of the savings is because Space X 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.
- This dataset includes a record for each payload carried during a SpaceX mission into outer space.

- https://github.com/kobosh/ibmdatasciencecapstoneproject/blob/master/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Load the SQL extension and establish a connection with the database

```
[4]: import csv, sqlite3
     import prettytable
     prettytable.DEFAULT = 'DEFAULT'

     con = sqlite3.connect("my_data1.db")
     cur = con.cursor()
```

```
[5]: !pip install -q pandas
```

```
[6]: %sql sqlite:///my_data1.db
```

```
[7]: import pandas as pd
     df = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/labs/module_2/data/Spacex.csv")
     df.to_sql("SPACEXTBL", con, if_exists='replace', index=False,method="multi")
```

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

```
[11]: cur.execute("SELECT distinct Launch_Site FROM spacextable")
      launchsites=cur.fetchall();
      for site in launchsites:
          print(site[0],end=',')
```

```
CCAFS LC-40,VAFB SLC-4E,KSC LC-39A,CCAFS SLC-40,
```

```
[12]: cur.execute("SELECT Customer FROM spacextable")
      launchsites=cur.fetchall();
      for c in launchsites:
          print(c[0])
```

Display 5 records where launch sites begin with the string CCA

```
cur.execute("SELECT Launch_Site FROM spacextable where Launch_Site like 'CCA%' LIMIT 5")
launchsites=cur.fetchall();
```

```
for site in launchsites:
     print(site[0])
```

```
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
```

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

```
[15]: cur.execute("SELECT PAYLOAD_MASS__KG_ FROM spacextable where Customer='NASA (CRS)'
      launchsites=cur.fetchall();
```

```
[15]: for site in launchsites:
          print(site[0])
```

List the date when the first successful landing outcome in ground pad was achieved

[17]:

```
cur.execute("SELECT MIN(Date)  FROM spacextable where Landing_Outcome='Success'")
launchsites=cur.fetchall();
```

[18]:

```
for site in launchsites:
     print(site)
```

```
('2018-07-22',)
```

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

```
cur = con.cursor()
cur.execute("SELECT Booster_Version FROM spacextable WHERE Landing_Outcome = 'Success' and PAYLOAD_MASS__KG_>4000")
launchsites=cur.fetchall();
for p in launchsites:
    print(p[0])
```

```
F9 B5B1047.1 •••
```

```
F9 B5B1047.1
F9 B5B1048.1
F9 B5 B1046.2
F9 B5B1049.1
F9 B5 B1047.2
```

List the total number of successful and failure mission outcomes

```
cur.execute("SELECT COUNT(Mission_Outcome )FROM spacextable")
launchsites=cur.fetchall();
for p in launchsites:
    print(p)
```

```
(101,)
```

List the names of the booster versions which have carried the maximum payload mass

```
cur.execute("SELECT DISTINCT Booster_Version FROM spacextable WHERE  PAYLOAD_MASS__KG_  IN (select MAX(PAYLOAD_MASS__KG_) from spacextable )")
launchsites=cur.fetchall();
for p in launchsites:
    print(p)
```

```
('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 ',)
```

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.

```
launchsites=cur.execute("""
    SELECT strftime('%m', Date) AS month, Landing_Outcome, Booster_Version, Launch_Site
    FROM spacextable
    WHERE substr(Date, 1, 4) = '2015'
    AND Landing_Outcome = 'Failure (drone ship)';
""")
for p in launchsites:
    print(p)
```

```
('01', 'Failure (drone ship)', 'F9 v1.1 B1012', 'CCAFS LC-40')
('04', 'Failure (drone ship)', 'F9 v1.1 B1015', 'CCAFS LC-40')
```

Rank the count of landing outcomes

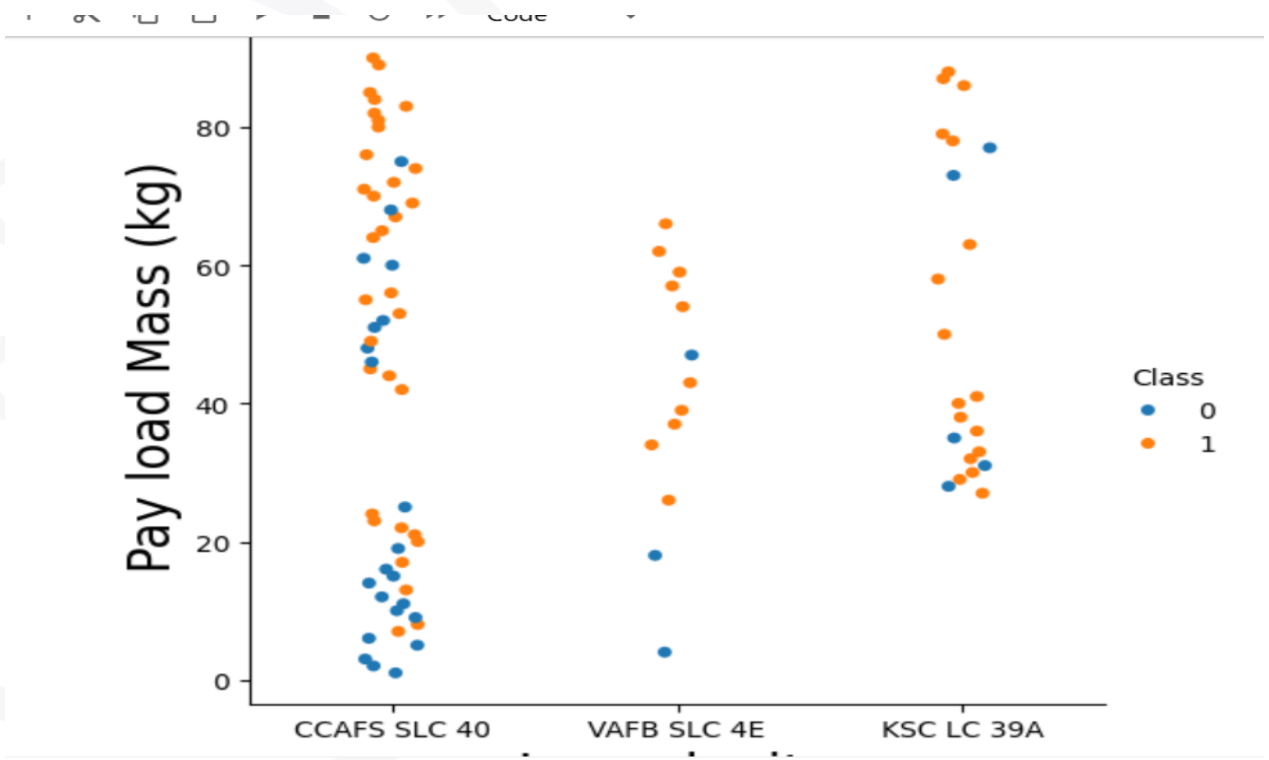
```
cur.execute(""" SELECT Landing_Outcome AS outcome, COUNT(*) AS outcome_count
FROM spacextable
WHERE Date BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY Landing_Outcome
ORDER BY outcome_count DESC;
      """)
launchsites=cur.fetchall();
for p in launchsites:
    print(p)
```

```
('No attempt', 10)
('Success (drone ship)', 5)
('Failure (drone ship)', 5)
('Success (ground pad)', 3)
('Controlled (ocean)', 3)
('Uncontrolled (ocean)', 2)
('Failure (parachute)', 2)
('Precluded (drone ship)', 1)
```

EDA Data Visualization

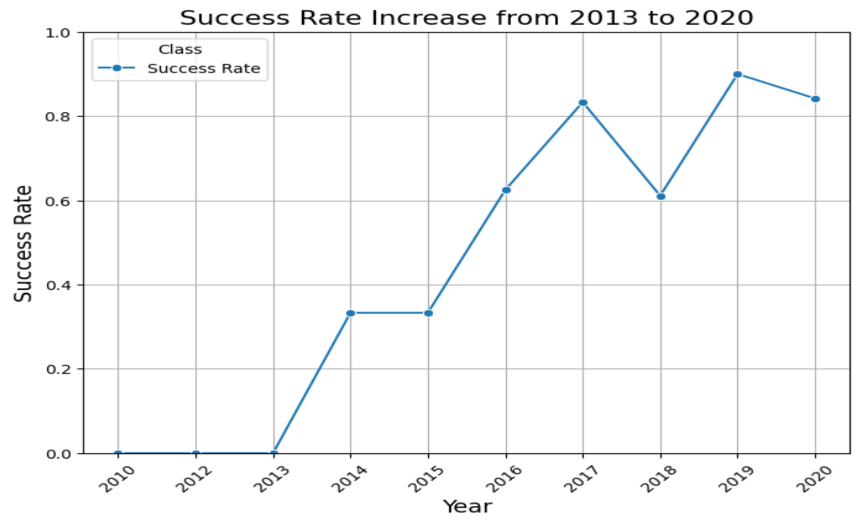
- **Exploring and Preparing Data**
- https://github.com/kobosh/ibmdatasciencecapstoneproject/blob/master/edadataviz%20(1).ipynb

Relationship Flight Number / Launch Site



Success Rate Increase 2013-2020

```
[75]: plt.figure(figsize=(8,6))
sns.lineplot(x='Date', y='Class', data=grpDate, marker='o', label='Success Rate')

plt.xlabel("Year", fontsize=14)
plt.ylabel("Success Rate", fontsize=14)
plt.title("Success Rate Increase from 2013 to 2020", fontsize=16)
plt.xticks(rotation=45)
plt.ylim(0, 1)  # Keeping the success rate within valid range
plt.legend(title="Class")
plt.grid(True)
```

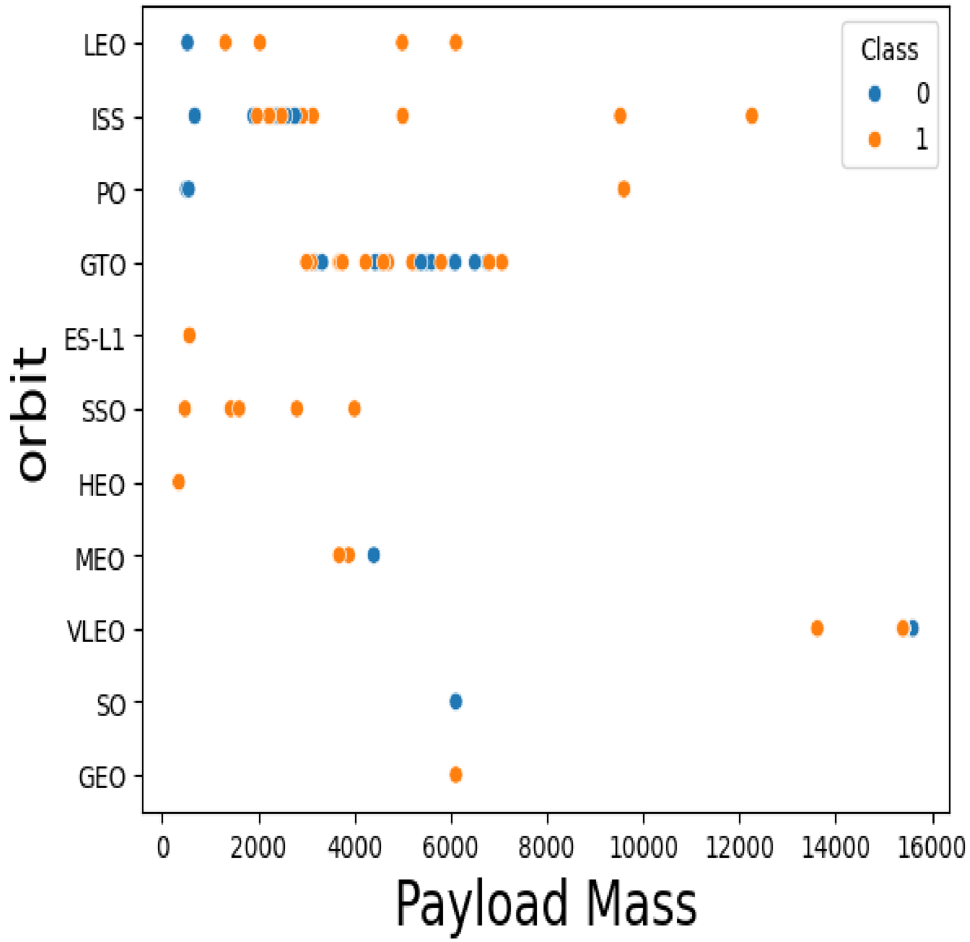


you can observe that the sucess rate since 2013 kept increasing till 2020

Features Engineering

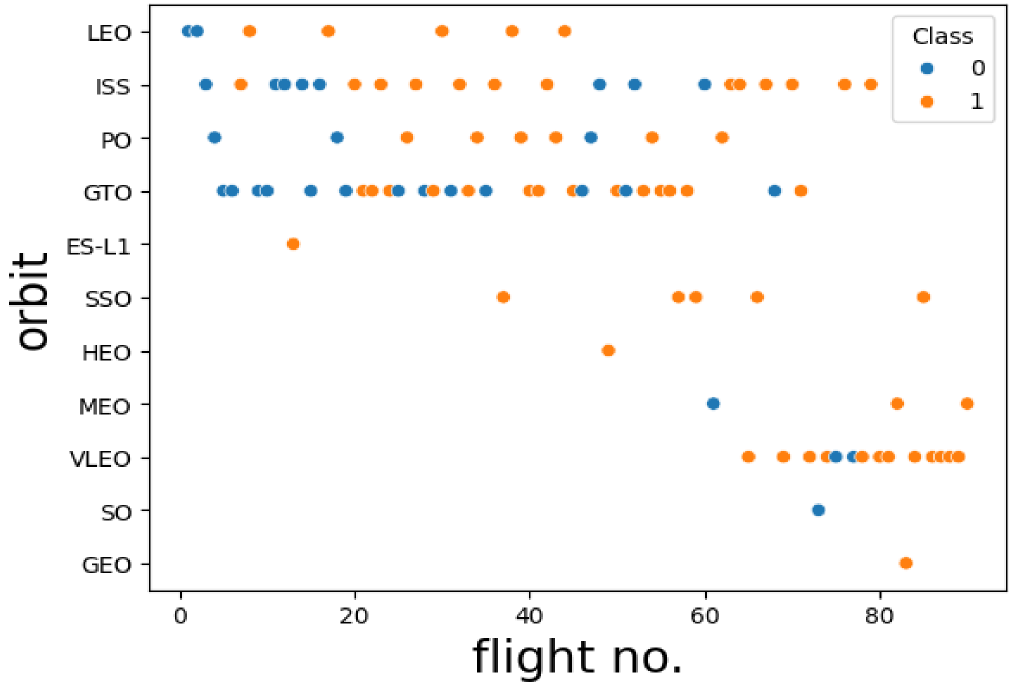
Relationship payload/orbit

```
sns.scatterplot(y="Orbit", x="PayloadMass", hue="Class", data=df)#
plt.xlabel("Payload Mass",fontsize=20)
plt.ylabel("orbit",fontsize=20)
plt.show()
```



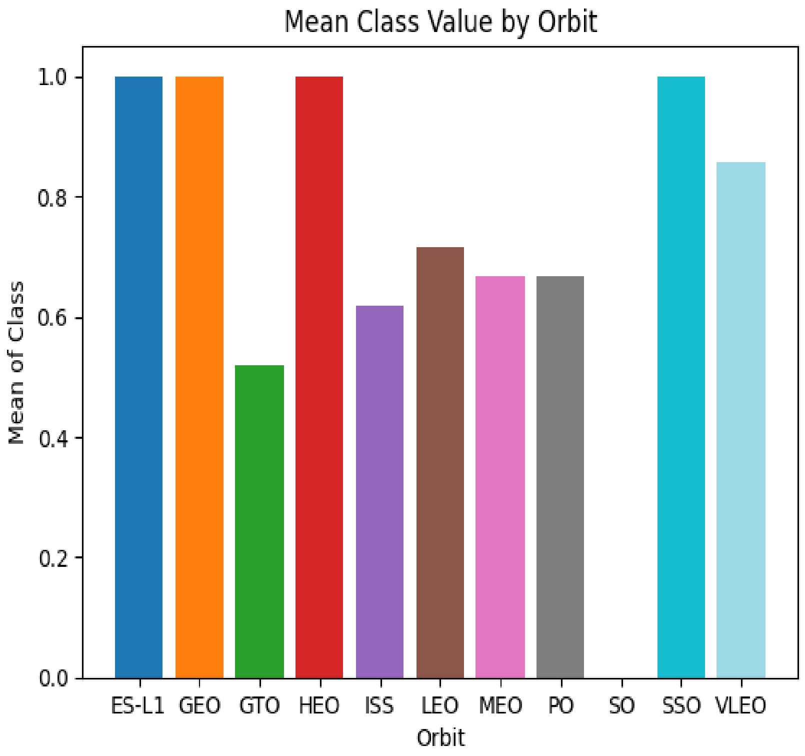
Relationship Flight Number / Orbit Type

```
sns.scatterplot(y="Orbit", x="FlightNumber", hue="Class", data=df)#
plt.xlabel("flight no.",fontsize=20)
plt.ylabel("orbit",fontsize=20)
plt.show()
```



Success Rate and Orbit Type

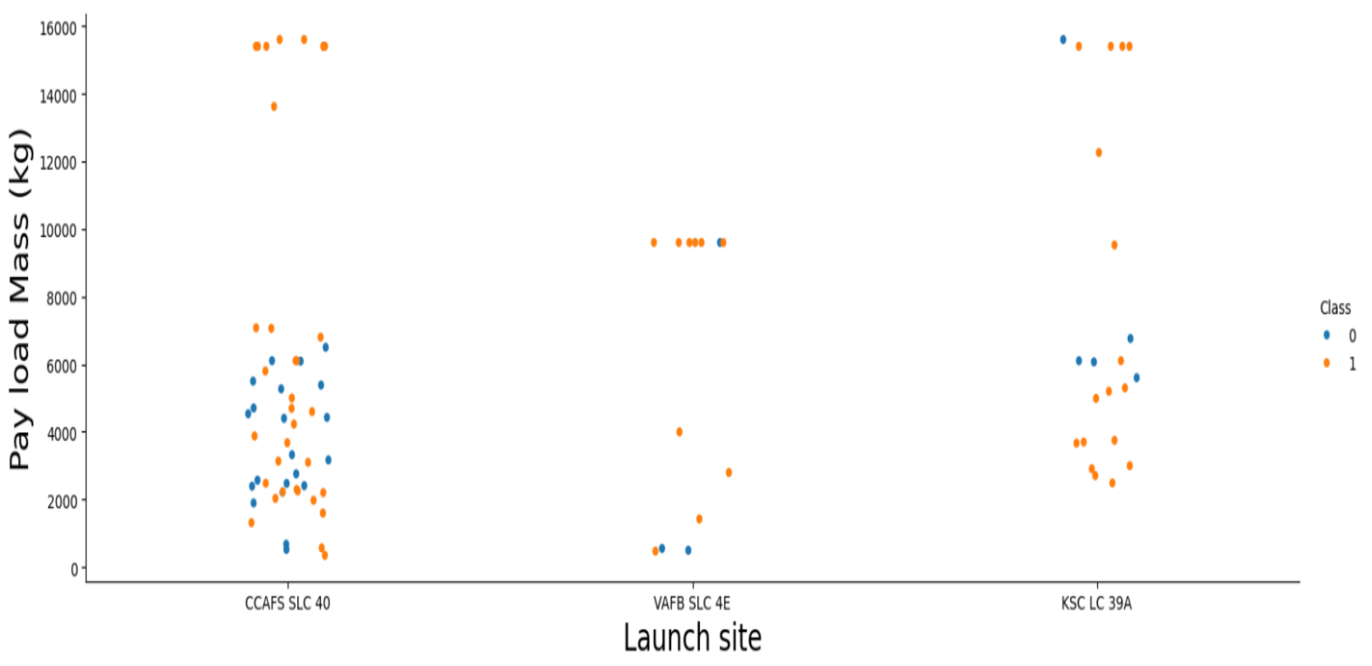
```
[79]: # HINT use groupby method on Orbit column and get the mean of Class column
ggrpOrbit=df[['Orbit' ,'Class']].groupby('Orbit',as_index=False)['Class'].mean()
num_orbits = grpOrbit['Orbit'].nunique()
cmap = plt.get_cmap('tab20', num_orbits)  # 'tab20' has a good range of distinct colors
colors = cmap(np.linspace(0, 1, num_orbits))
# Plotting
fig, ax = plt.subplots()
bars = ax.bar(grpOrbit['Orbit'], grpOrbit['Class'], color=colors)
plt.xlabel('Orbit')
plt.ylabel('Mean of Class')
plt.title('Mean Class Value by Orbit')
plt.show()
```



#

Relationship Payload Mass/Launch Site

```
#hue to be the class value
sns.catplot(y="PayloadMass", x="LaunchSite", hue="Class", data=df, aspect = 5)
plt.xlabel("Launch site",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```

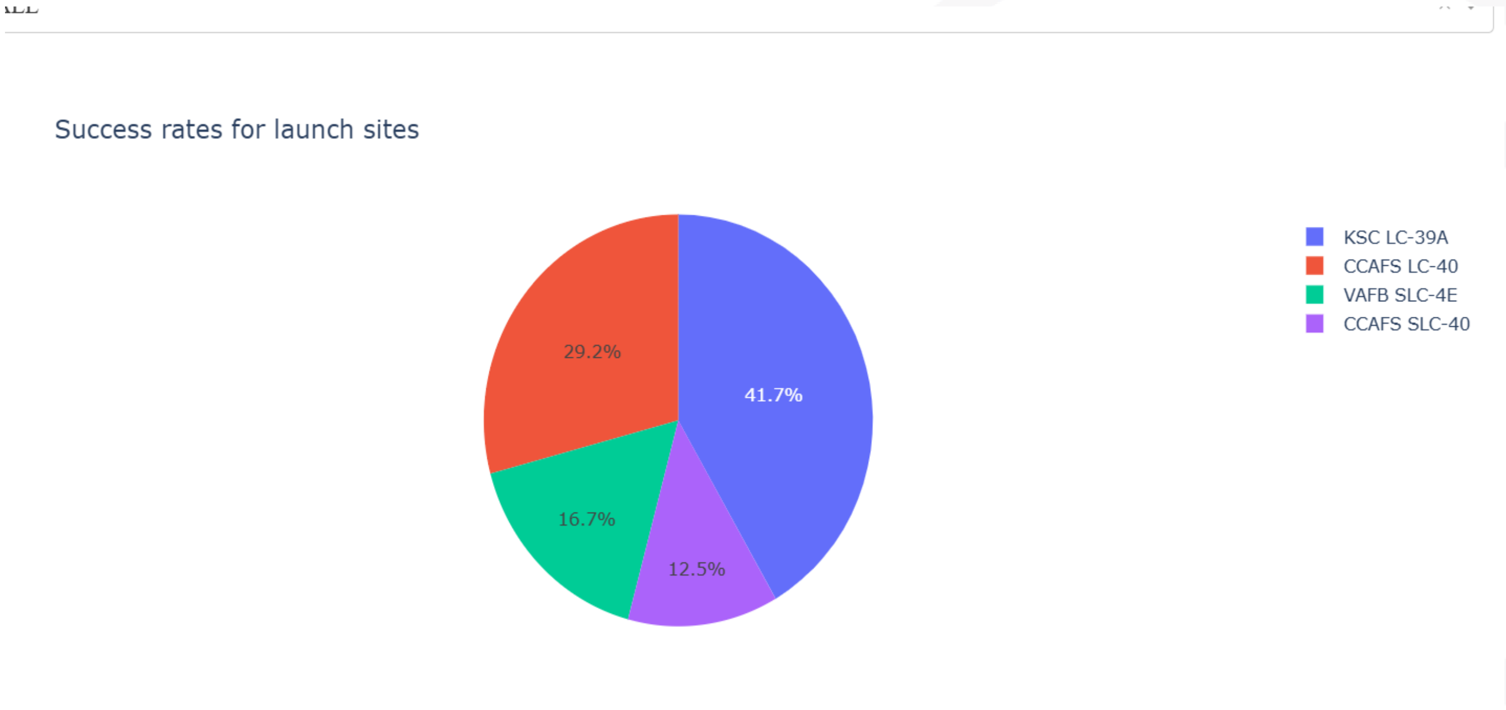


Plotly Dashboard

- Plotly Dash application for users to perform interactive visual analytics on SpaceX launch data in real-time.

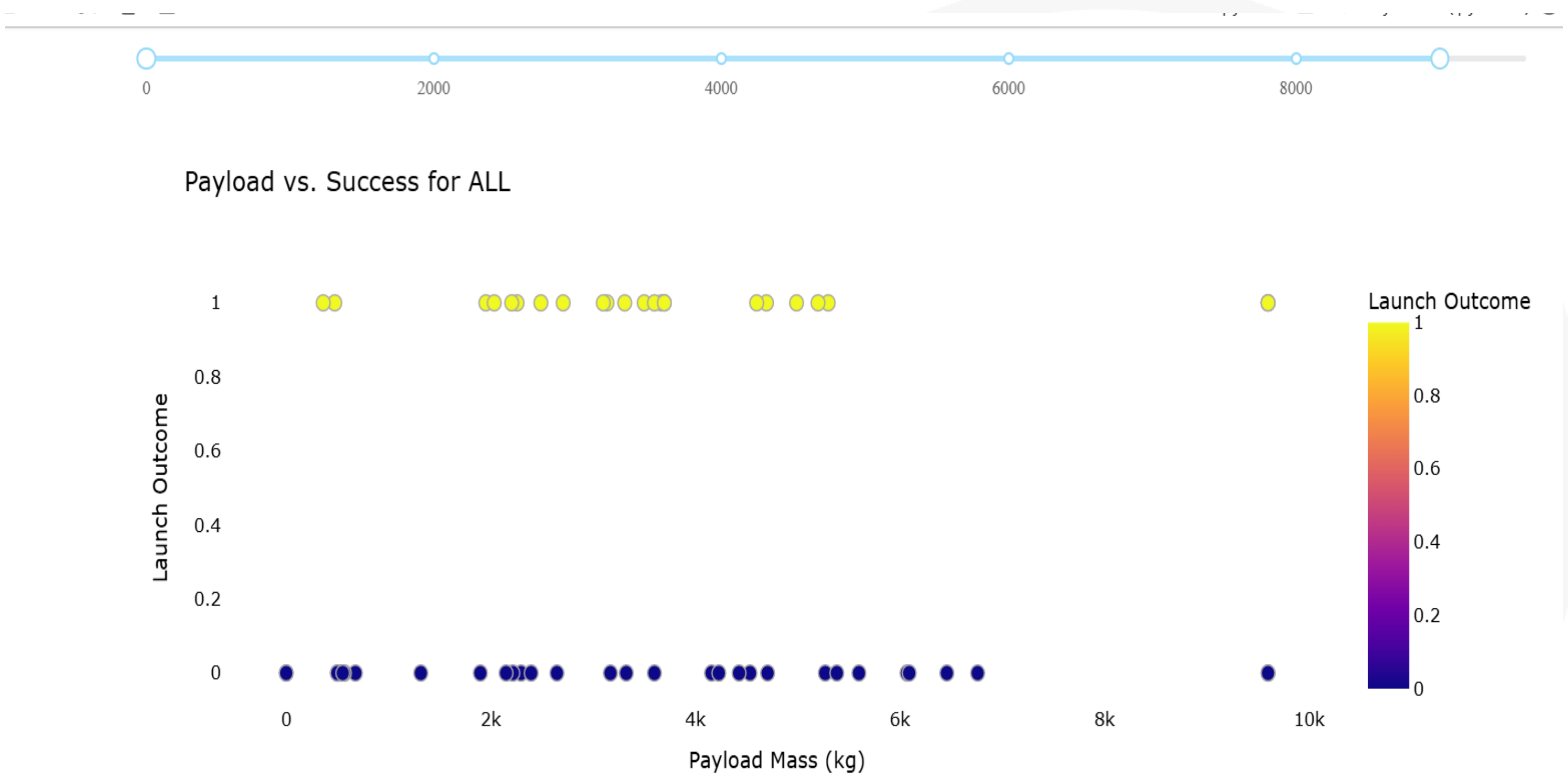
- https://github.com/kobosh/ibmdatasciencecapstoneproject/blob/master/spacex_dash_app.py

Success Rates for All Sites



Success Rate of CCAFSLC_40

Payload vs Success for All



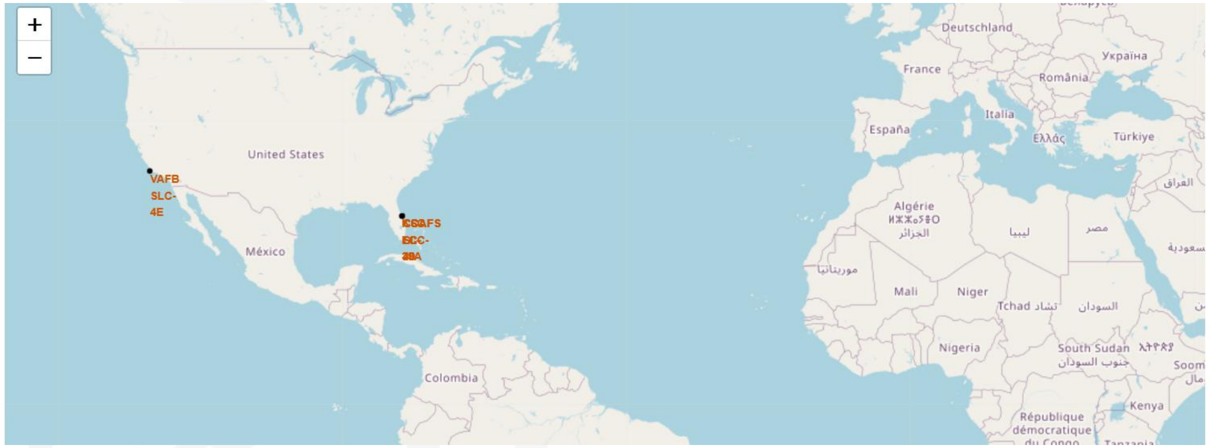

Payload range (Kg):
0
2000
4000
6000
8000
Payload vs. Success for CCAFS LC-40
Launch Outcome
1
0.8
0.6
0.4
0.2
0
2000
2500
3000
3500
4000
4500
5000
Launch Outcome
1
0.8
0.6
0.4
0.2
0

Interactive Visual Analytics Notebook

- **Interactive Visual Analytics with Folium**
- https://github.com/kobosh/ibmdatasciencecapstoneproject/blob/master/lab_jupyter_launch_site_location.ipynb

Mark all Sites

```python
site_map = folium.Map(location=nasa_coordinate, zoom_start=5)
site_map
for  site in launch_sites_df.iterrows():

    loc=[]
    loc.append( site[1]['Lat'])
    loc.append(site[1]['Long'])
    label=site[1][0]
    marker=folium.map.Marker(loc, icon=DivIcon(icon_size=(20,20),icon_anchor=(0,0),
                                   html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % label, ))
    circle=folium.Circle(loc, radius=1000, color='#000000', fill=True).add_child(folium.Popup(label))
    site_map.add_child(marker)
    site_map.add_child(circle)
site_map
```



success/failed launches for each site on the map

TODO: Create a new column in `launch_sites` dataframe called `marker_color` to store the marker colors based on t

```
# Function to assign color to launch outcome
def assign_marker_color(launch_outcome):
    if launch_outcome == 1:
        return 'green'
    else:
        return 'red'

spacex_df['marker_color'] = spacex_df['class'].apply(assign_marker_color)
spacex_df.tail(10)
```

<div> •••

```
for index, record in spacex_df.iterrows():
    print(record)
    break
```

```
Launch Site     CCAFS LC-40
Lat               28.562302
Long             -80.577356
class                     0
marker_color            red
Name: 0, dtype: object
```

TODO: For each launch result in `spacex_df` data frame, add a `folium.Marker` to `marker_cluster`

```
# Add marker_cluster to current site_map
```

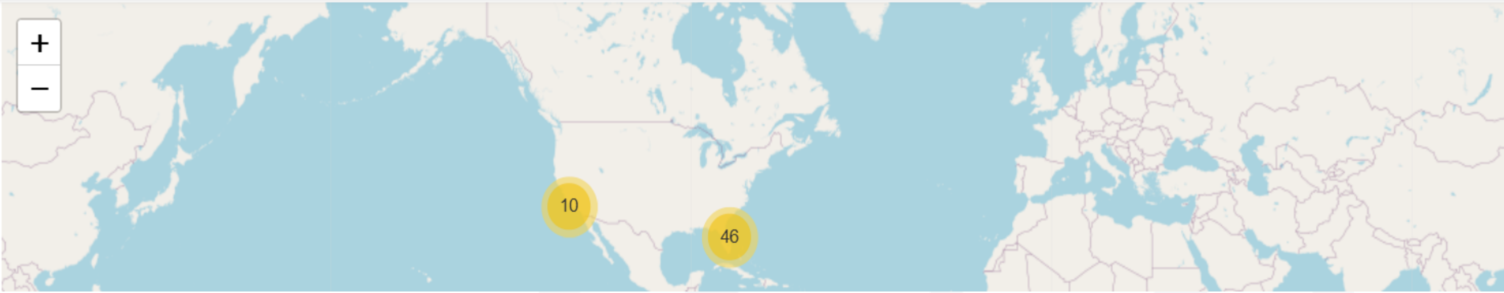
Clustered Launch Sites

TODO: For each launch result in `spacex_df` data frame, add a `folium.Marker` to `marker_cluster`

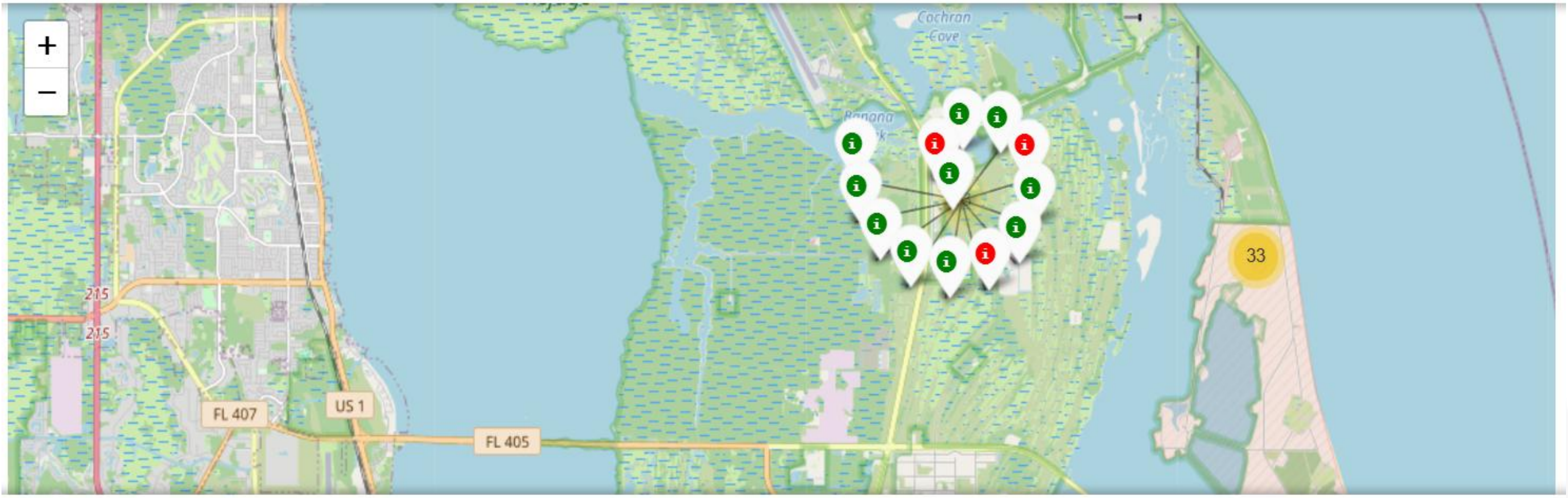
[170]:

```python
# Add marker_cluster to current site_map
marker_cluster = MarkerCluster()
map_center =[31.0, -99.0] # [spacex_df['Lat'].mean(), spacex_df['Long'].mean()]
site_map=folium.Map(map_center,zoom_start=5)
site_map.add_child(marker_cluster)
for site_lat, site_long, marker_color in zip(spacex_df['Lat'], spacex_df['Long'], spacex_df['marker_color']):
    site_coordinate = [site_lat, site_long]
    marker = folium.map.Marker(
        site_coordinate,
        # Create an icon as a text label
        icon=folium.Icon(color='white',
                         icon_color=marker_color)
    )
    marker.add_to(marker_cluster)
site_map
```

[170]:



Launch Success



Machine Learning Prediction Notebook

- Perform exploratory Data Analysis and determine Training Labels
- create a column for the class
- Standardize the data
- Split into training data and test data
- -Find best Hyperparameter for SVM, Classification Trees and Logistic Regression
- Find the method performs best using test data

- https://github.com/kobosh/ibmdatasciencecapstoneproject/blob/master/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Apply Logistic Regression

```
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, scoring='accuracy')
logreg_cv.fit(X_train,Y_train)
```

Logistic Reg Confusion Matrix

```
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 : 0.8464285714285715
```

TASK 5

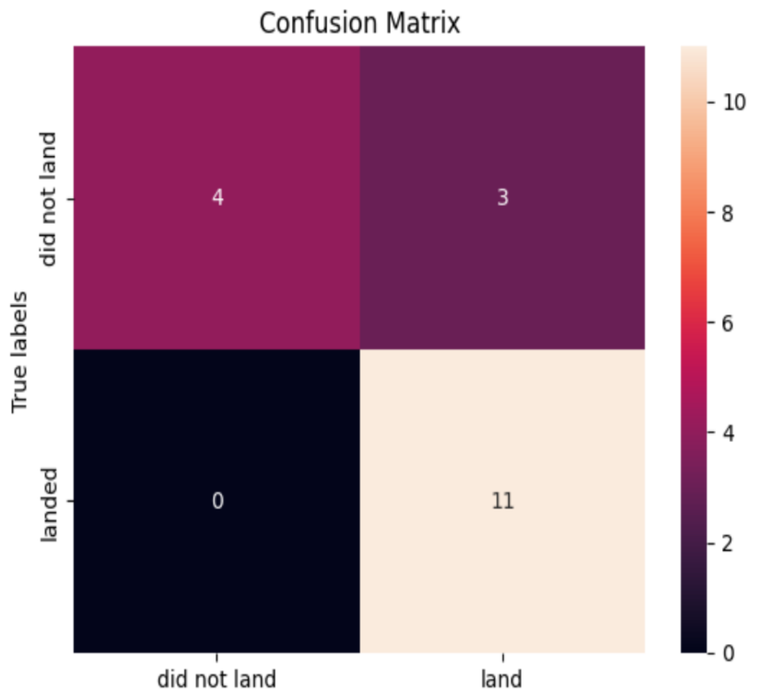
Calculate the accuracy on the test data using the method `score` :

```
accuracy = logreg_cv.score(X_test, Y_test)
print("Test accuracy:", accuracy)
```

```
Test accuracy: 0.8333333333333334
```

Lets look at the confusion matrix:

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



Vector Machine Prediction

[58]:
```
parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
              'C': np.logspace(-3, 3, 5),
              'gamma':np.logspace(-3, 3, 5)}
svm = SVC()
```

[62]:
```
svm_cv=GridSearchCV(svm,parameters, cv=10,scoring='accuracy')
svm_cv.fit(X_train,Y_train)
```

[62]:
```
GridSearchCV
best_estimator_: SVC
SVC
```

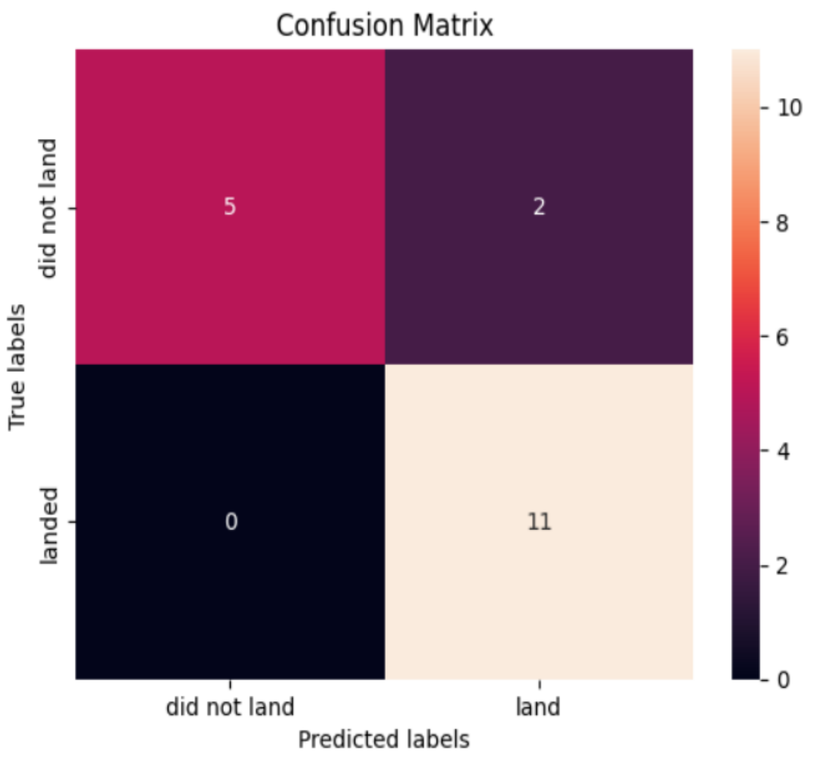
[63]:
```
print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
print("accuracy :",svm_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
accuracy : 0.8321428571428571
```

TASK 7

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

[66]:
```
accuracy=svm_cv.score(X_test,Y_test)
accuracy
```



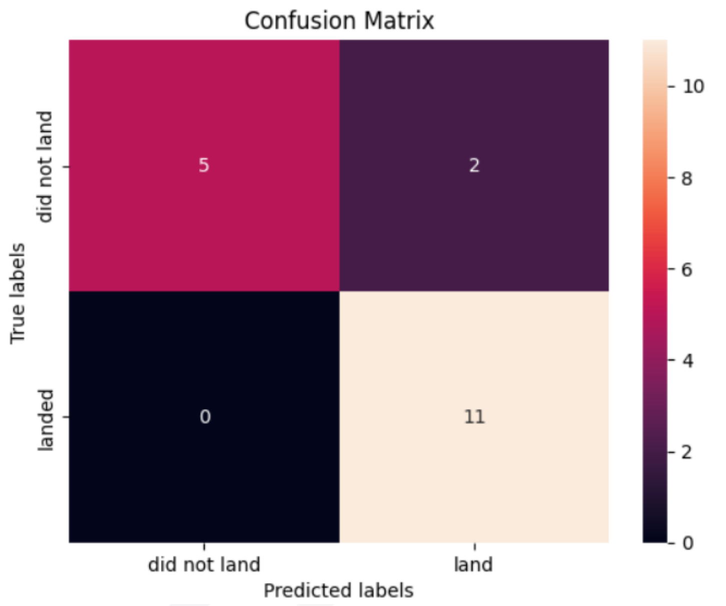
Decision Tree Classifier

Create a decision tree classifier object then create a `GridSearchCV` object `tree_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

```
[70]: 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()
```

```
[76]: tree_cv=GridSearchCV(tree,parameters,cv=10,scoring='accuracy')
tree_cv.fit(X_train,Y_train)
```

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



K Nearest Neighbors

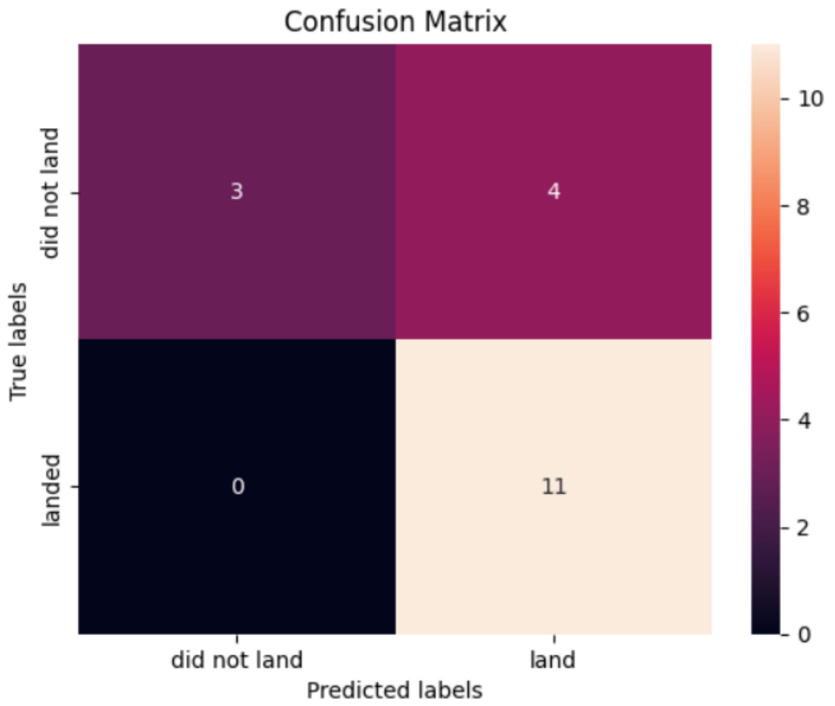
Create a k nearest neighbors object then create a `GridSearchCV` object `knn_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

```
[84]: 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()
```

```
[88]: knn_cv=GridSearchCV(KNN,parameters,cv=10,scoring='accuracy')
      knn_cv.fit(X_train,Y_train)
```

```
[88]: ▸ GridSearchCV
      ▸ best_estimator_: KNeighborsClassifier
          ▸ KNeighborsClassifier
```

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

      tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 5, 'p': 1}
      accuracy : 0.8589285714285715
```



RESULTS

- Source of Data:
- Point2
- Point3
- Point4
  - Sub Point1
  - Sub Point2

PROGRAMMING LANGUAGE TRENDS

Current Year

<Bar chart of top 10 programming languages for the current year goes here.>

Next Year

< Bar chart of top 10 programming languages for the next year goes here.>

PROGRAMMING LANGUAGE TRENDS - FINDINGS & IMPLICATIONS

Findings

- Finding 1
- Finding 2
- Finding 3

Implications

- Implication 1
- Implication 2
- Implication 3

DATABASE TRENDS

Current Year

< Bar chart of top 10 databases for the current year goes here >

Next Year

< Bar chart of top 10 databases for the next year goes here.>

DATABASE TRENDS - FINDINGS & IMPLICATIONS

Findings

- Finding 1
- Finding 2
- Finding 3

Implications

- Implication 1
- Implication 2
- Implication 3

DASHBOARD

<The GitHub link of the Cognos/Looker Studio dashboard goes here.>

DASHBOARD TAB 1

Screenshot of dashboard tab 1 goes here

DASHBOARD TAB 2

Screenshot of dashboard tab 2 goes here

DASHBOARD TAB 3

Screenshot of dashboard tab 3 goes here

DISCUSSION

OVERALL FINDINGS & IMPLICATIONS

Findings

- Finding 1
- Finding 2
- Finding 3

Implications

- Implication 1
- Implication 2
- Implication 3

CONCLUSION

- Point 1
- Point 2
- Point 3
- Point 4

APPENDIX

- Include any relevant additional charts, or tables that you may have created during the analysis phase.

JOB POSTINGS

In Module 1 you have collected the job posting data using Job API in a file named "job-postings.xlsx". Present that data using a bar chart here. Order the bar chart in the descending order of the number of job postings.

POPULAR LANGUAGES

In Module 1 you have collected the job postings data using web scraping in a file named “popular-languages.csv”. Present that data using a bar chart here. Order the bar chart in the descending order of salary.