IBM Developer
SKILLS NETWORK

A helping hand from Data Science for Spacex

Krishnan Veerendran

4th Feb 2022

Contents

Executive Summary

- Various tasks performed in gathering, exploring, analysing and concluding in this project are mentioned here
  - Summary of Methodologies
    - Data Collection
    - Data Wrangling
    - EDA – using SQL, Pandas and MatplotLib
    - EDA – Visualization with Folium and Dashboard with Plotly Dash
    - Predictive Analysis – Classification
    - Summary of Methodologies
  - Conclusions
    - Reports from Data Visualization
    - Results from Interactive Visualization
    - Results from Predictive Analysis

Background Introduction

- The purpose behind this exercise is to find out the successful landing of Stage 1 of Falcon 9 rockets from the various other independent parameters which would have influenced the success of the stage 1 landing. The learnings and predictions will help other rocket launchers to reduce their cost since the low cost of Spacex program is attributed to the successful landing of stage 1.
- And the questions to be answered are,
  - What are the parameters that influences the successful landing of stage 1
  - The relationship importance between the parameters and the success
  - What needs to be accomplished to ensure better success rate

Methodology – Data Collection API

- Data Collection
  - This was performed by reading the data from https://api.spacexdata.com/v4/launches/past and the read data is converted to a data frame from a JSON format
  - The variable names are changed to more meaningful ones
  - The Falcon 9 related information was filtered out since this is of our interest.
  - Now the filtered out data frame has 90 rows with 17 columns

Methodology – Data Collection WEB scrapping

- Data Collection
  - This was performed by web scrapping the data from "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
  - The HTML read data is converted to a data frame
  - And the data frame is transferred to a csv file
  - `df.to_csv('spacex_web_scraped.csv', index=False)`

Methodology Continuation

- Data Wrangling
  - **Data wrangling** is the process of cleaning, structuring and enriching raw data into a desired format for better decision making
    - The 'PayLoadMass' column had 5 null values which were replaced by the 'mean' of the column

```
In [69]: # Calculate the mean value of PayloadMass column

         #sub2['income'].fillna((sub2['income'].mean()), inplace=True)
         data_falcon9['PayloadMass'].fillna((data_falcon9['PayloadMass'].mean()), inplace=True)
         data_falcon9.isnull().sum()
```

Methodology Data Wrangling Continuation

The ‘outcome’ column of the data frame had string values which needs to be converted into binary values. The binary values are stored under new column called ‘Class’. This will help us predict the landing success in 1 or 0.

Using the `Outcome`, create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome`; otherwise, it's one. Then assign it to the variable `landing_class`:

In [37]:
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
i=-1
while i< 89:
    i=i+1
    if df.iloc[[i],[6]].values == 'True ASDS':
        #print('ok 1')
        landing_class.append(1)
    elif df.iloc[[i],[6]].values == 'True RTLS':
        #print('ok 2')
        landing_class.append(1)
    elif df.iloc[[i],[6]].values == 'True Ocean':
        #print('ok 3')
        landing_class.append(1)
    else :
        #print('not ok')
        landing_class.append(0)
print(len(landing_class))
```

```
90
```

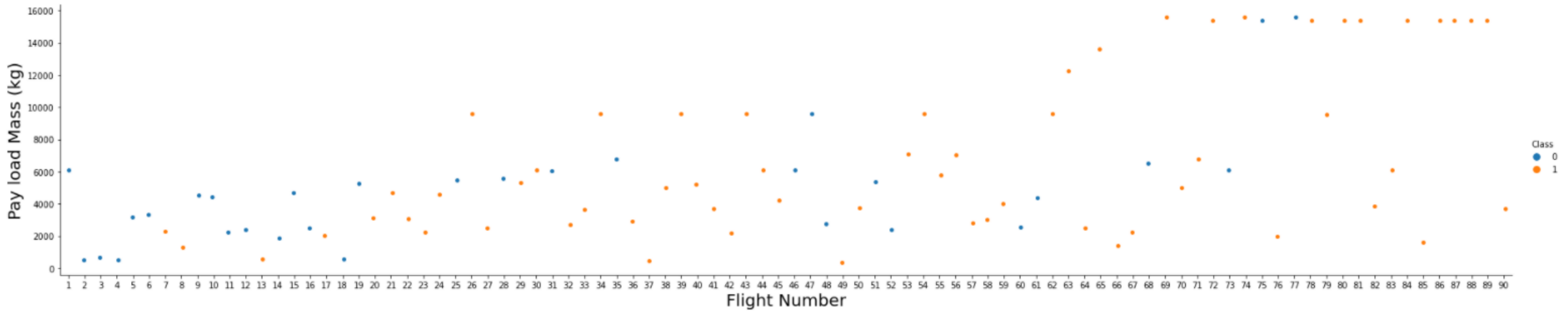
This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

In [39]:
```
df['Class']=landing_class
df[['Class']].head(25)
```

Exploratory data analysis (EDA) using visualization

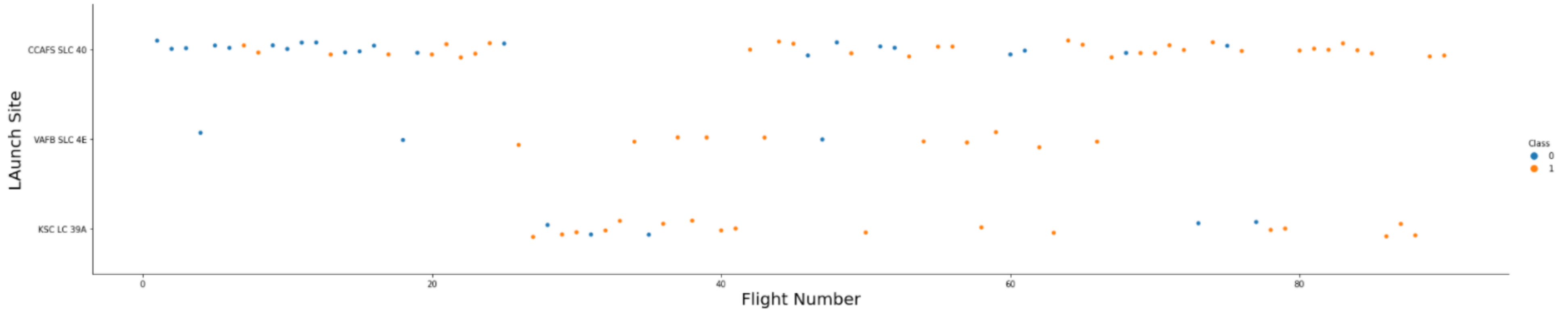
In [4]:
```
sns.catplot(y="PayloadMass", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```

It is observed that the success rate increases with flight number



In [5]:
```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("LAunch Site",fontsize=20)
plt.show()
```

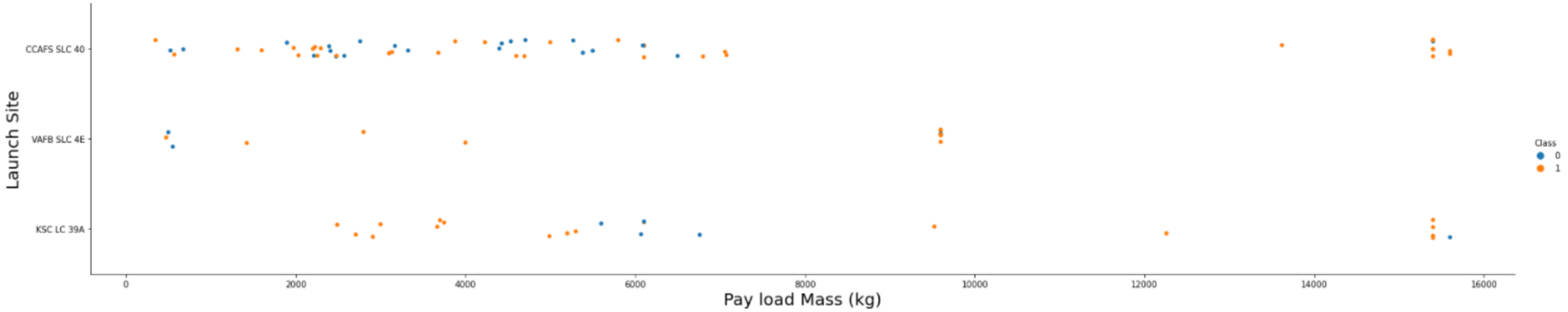
It is observed that the success rate increases with flight number for all Launching sites



Exploratory data analysis (EDA) using visualization

In [6]:
```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("Pay load Mass (kg)",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```

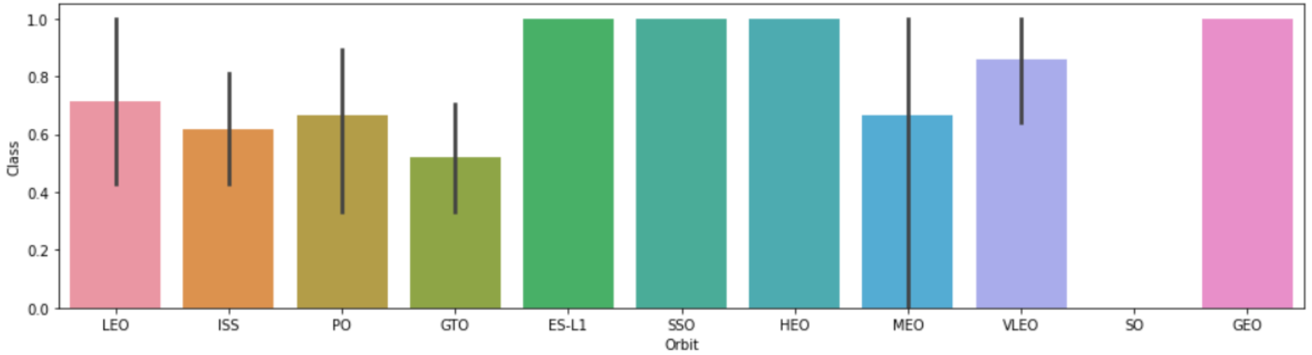
It is observed that the success rate increases with Pay load mass for all sites, more so for CCAFS SLC 40



In [69]:
```
# HINT use groupby method on Orbit column and get the mean of Class column
#df1 = df.groupby(['Orbit']).mean()
#df1 = df.reset_index()
plt.figure(figsize=(16, 4))
# plot barplot
sns.barplot(x=df1.Orbit, y=df1.Class)
#df1.head()
```

It is observed that the success rate is 100% for orbits ES-L1, SSO, HEO and GEO. 0% for SO

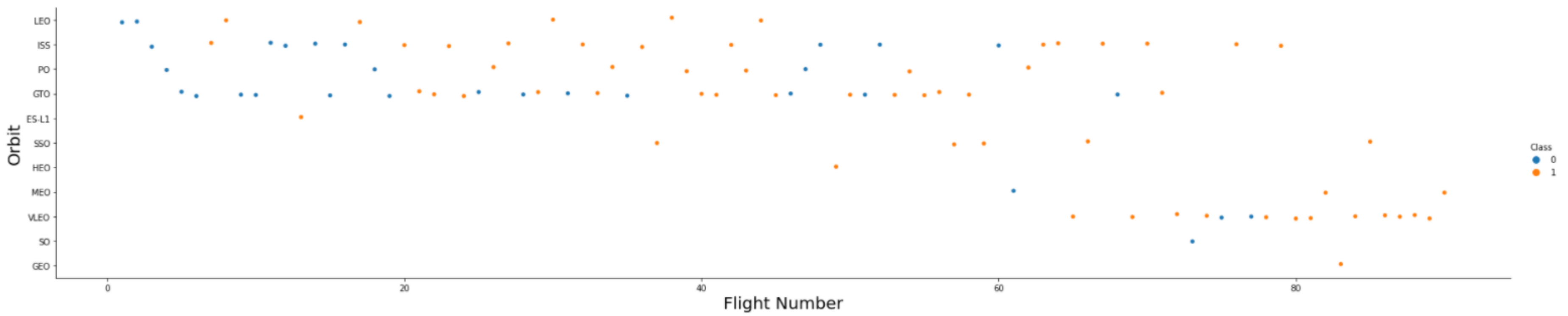
Out[69]: <AxesSubplot:xlabel='Orbit', ylabel='Class'>



Exploratory data analysis (EDA) using visualization

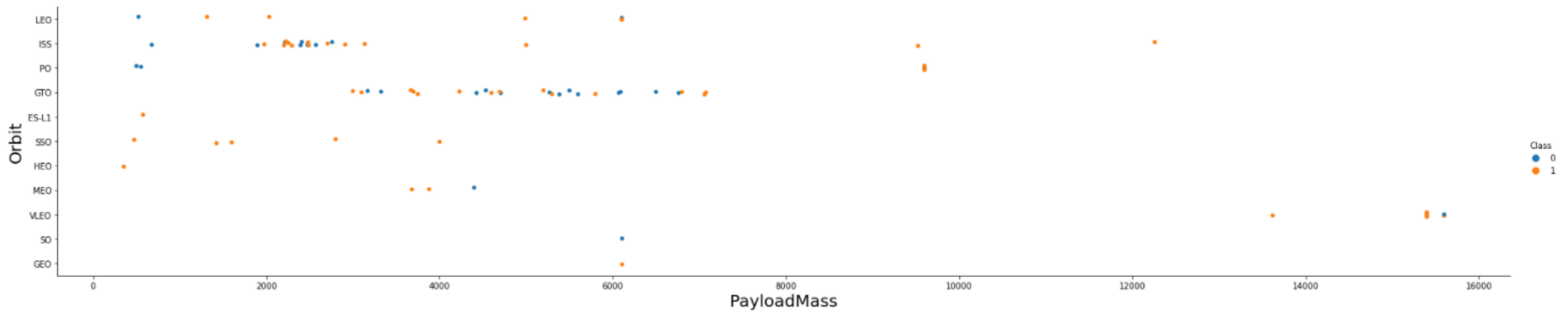
In [8]:
```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```

It is observed that the success rate increases with Flight number for LEO, ISS, PO, GTO, VLEO



In [9]:
```
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("PayloadMass",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```

It is observed that the Pay Load has no impact on success for GTO. But improved for ISS and PO

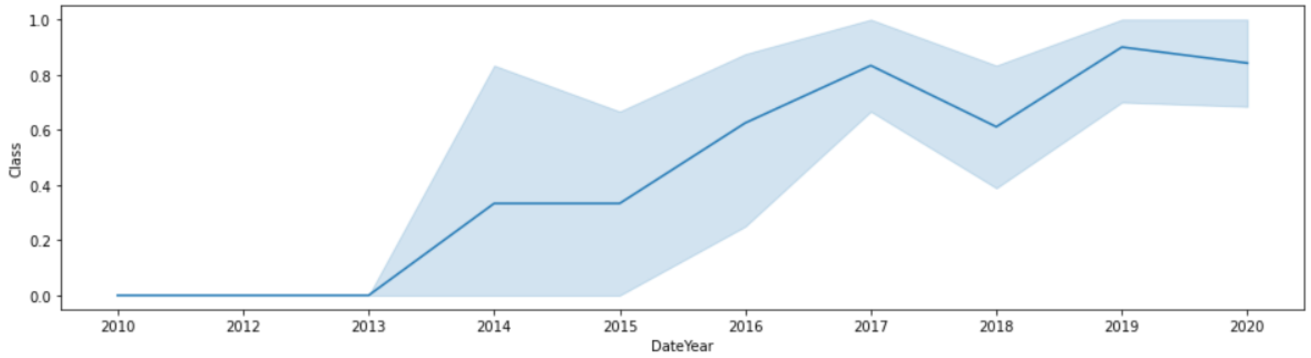


Exploratory data analysis (EDA) using visualization

In [73]:
```python
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
df3['DateYear'] = Extract_year(df['Date'])
#df3 = df.groupby(['Date']).mean()
#df3 = df.reset_index()
plt.figure(figsize=(16, 4))
# plot barplot
sns.lineplot(x=df3.DateYear, y=df3.Class)
```

It is observed that the success rate steadily increases from year 2013 onwards

Out[73]: <AxesSubplot:xlabel='DateYear', ylabel='Class'>



Exploratory data analysis (EDA) with SQL Results

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

In [5]:
```
%sql select DISTINCT(LAUNCH_SITE) from SPACEXTBL
```

```
 * ibm_db_sa://xjw20989:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

Launch Site names

Out[5]:

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

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

In [6]:
```
%sql select * from SPACEXTBL where LAUNCH_SITE LIKE 'CCA%' LIMIT 5
```

```
 * ibm_db_sa://xjw20989:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

Out[6]:

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

Exploratory data analysis (EDA) with SQL Results

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

```
In [9]: %sql select SUM(Payload_Mass__Kg_) from SPACEXTBL where customer LIKE 'NASA (CRS)'

 * ibm_db_sa://xjw20989:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

Out[9]:

1
45596

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

```
In [9]: %sql select AVG(Payload_Mass__Kg_) from SPACEXTBL where booster_version LIKE 'F9 v1.1'

 * ibm_db_sa://xjw20989:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

Out[9]:

1
2928

Exploratory data analysis (EDA) with SQL Results

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

Hint:Use min function

In [10]:
```
%sql select MIN(DATE) from SPACEXTBL where landing__outcome LIKE 'Success (ground pad)'
```

```
 * ibm_db_sa://xjw20989:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

Out[10]:

1
2015-12-22

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

In [16]:
```
%sql select booster_version from SPACEXTBL where ((payload_mass__kg_ > 4000) & (payload_mass__kg_ < 6000) & (landing__outcome LIKE 'Success (drone ship)'))
```

```
 * ibm_db_sa://xjw20989:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

Out[16]:

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

Exploratory data analysis (EDA) with SQL Results

List the total number of successful and failure mission outcomes

In [21]:

```
%sql select (select COUNT(mission_outcome) from SPACEXTBL where mission_outcome LIKE '%Success%') as Success, \
(select COUNT(mission_outcome) from SPACEXTBL where mission_outcome LIKE '%Fail%') as Failure from SPACEXTBL LIMIT 1;
```

```
 * ibm_db_sa://xjw20989:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

Out[21]:

success	failure
100	1

%sql select booster_version from SPACEXTBL where payload_mass__kg_ = (select MAX(payload_mass__kg_) from SPACEXTBL)

```
 * ibm_db_sa://xjw20989:***@ba99a9e6-d59e-4883-8fc0-
d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/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
```

Exploratory data analysis (EDA) with SQL Results

List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

In [49]:
```
%sql select landing__outcome, booster_version, launch_site from SPACEXTBL where ((DATE LIKE '2015%') & (landing__outcome LIKE 'Failure (drone ship)'))
```

```
 * ibm_db_sa://xjw20989:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

Out[49]:

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 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

In [22]:
```
%sql select landing__outcome, COUNT(*) as count_outcome from SPACEXTBL where DATE > '06-04-2010' AND \
DATE < '03-20-2017' GROUP BY landing__outcome ORDER BY count_outcome DESC
```

```
 * ibm_db_sa://xjw20989:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

Out[22]:

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

Exploratory data analysis (EDA) visualization with Folium

All 4 sites are marked using the co-ordinates read from the file (URL link). Three in Florida and one in California

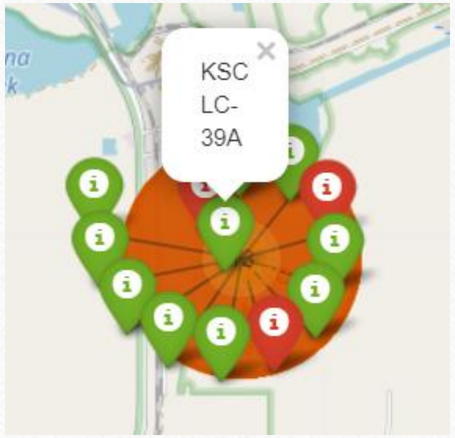
	Launch Site	Lat	Long
0	CCAFS LC-40	28.562302	-80.577356
1	CCAFS SLC-40	28.563197	-80.576820
2	KSC LC-39A	28.573255	-80.646895
3	VAFB SLC-4E	34.632834	-120.610746

Exploratory data analysis (EDA) visualization with Folium

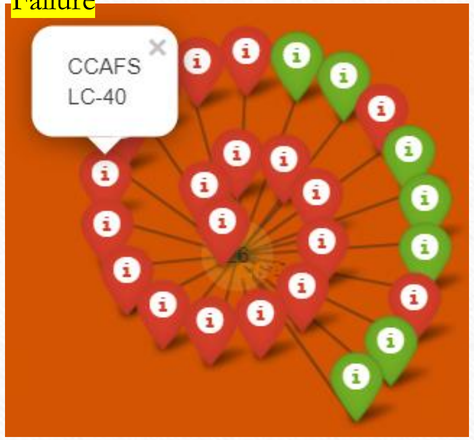
```
# Add marker_cluster to current site_map
from folium.plugins import MarkerCluster
nasa_coordinate = [29.559684888503615, -95.0830971930759]
site_map = folium.Map(location=nasa_coordinate, zoom_start=5)
marker_cluster = MarkerCluster().add_to(site_map)

# for each row in spacex_df data frame
# create a Marker object with its coordinate
# and customize the Marker's icon property to indicate if this launch was successed or failed,
# e.g., icon=folium.Icon(color='white', icon_color=row['marker_color']
#marker_cluster = MarkerCluster().add_to(site_map)
for i, row in spacex_df.iterrows():
    Lat = spacex_df.at[i,'Lat']
    Lng = spacex_df.at[i,'Long']
    m_color = spacex_df.at[i,'marker_color']
    pop = spacex_df.at[i,'Launch Site']

    circle4 = folium.Circle(location=[Lat, Lng], radius=1000, color='#d35400', fill=True).add_child(folium.Popup('VAFB SLC-4E'))
    marker4 = folium.map.Marker(location=[Lat, Lng], icon=DivIcon(icon_size=(20,20), icon_anchor=(0,0),html='<div style="font-size: 12; color:#d35400;"><b>%s</b>
    site_map.add_child(circle4)
    site_map.add_child(marker4)
    folium.Marker(location=[Lat, Lng], popup=pop,icon=folium.Icon(color=m_color)).add_to(marker_cluster)
    #folium.MarkerCluster(marker_cluster).add_to(site_map)


site_map
```

VAFB SLC 4E – 4 Success 6 Failure

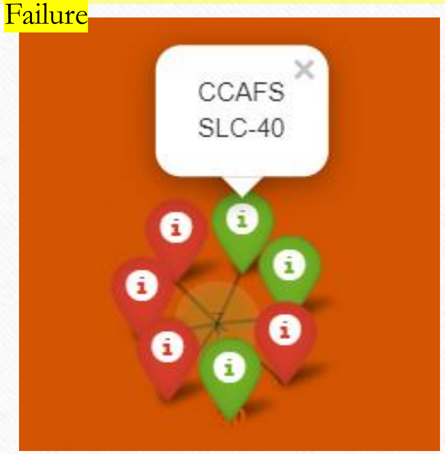
KSC LC 39A – 10 Success 3 Failure



CCAFS LC 40 – 7 Success 19 Failure



CCAFS SLC 40 – 3 Success 4 Failure

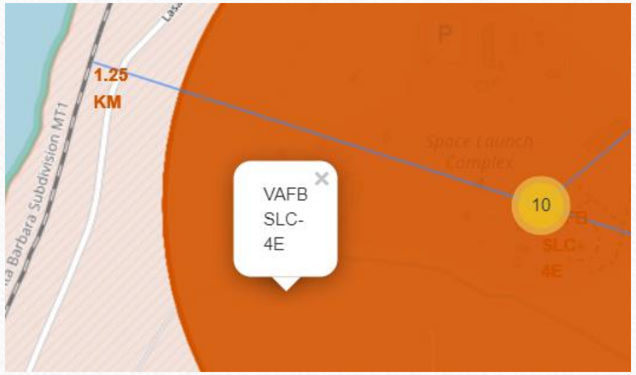


Exploratory data analysis (EDA) visualization with Folium

VAFB SLC 4E to nearest coastline distance 1.25 KM

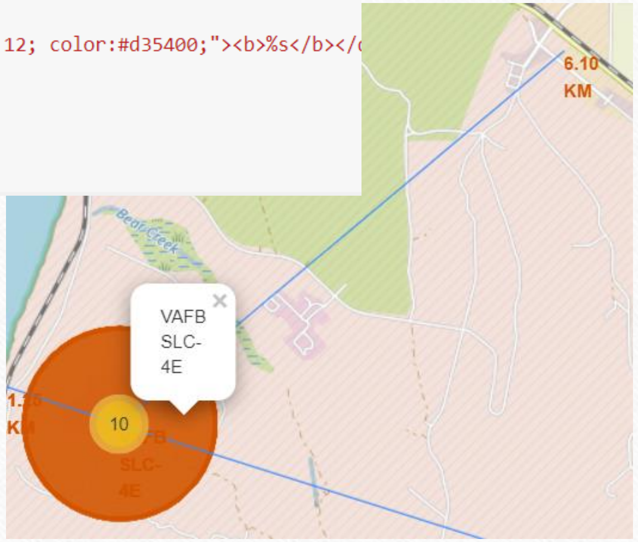
```
# find coordinate of the closet coastline
# e.g.,: Lat: 28.56367  Lon: -80.57163
Lat = spacex_df.at[28,'Lat']
Lng = spacex_df.at[28,'Long']
distance_coastline = calculate_distance(Lat, Lng, 34.63626, -120.62375)
print(distance_coastline)
```

```
1.24967252745819
```



VAFB SLC 4E to highway distance 6.10 KM

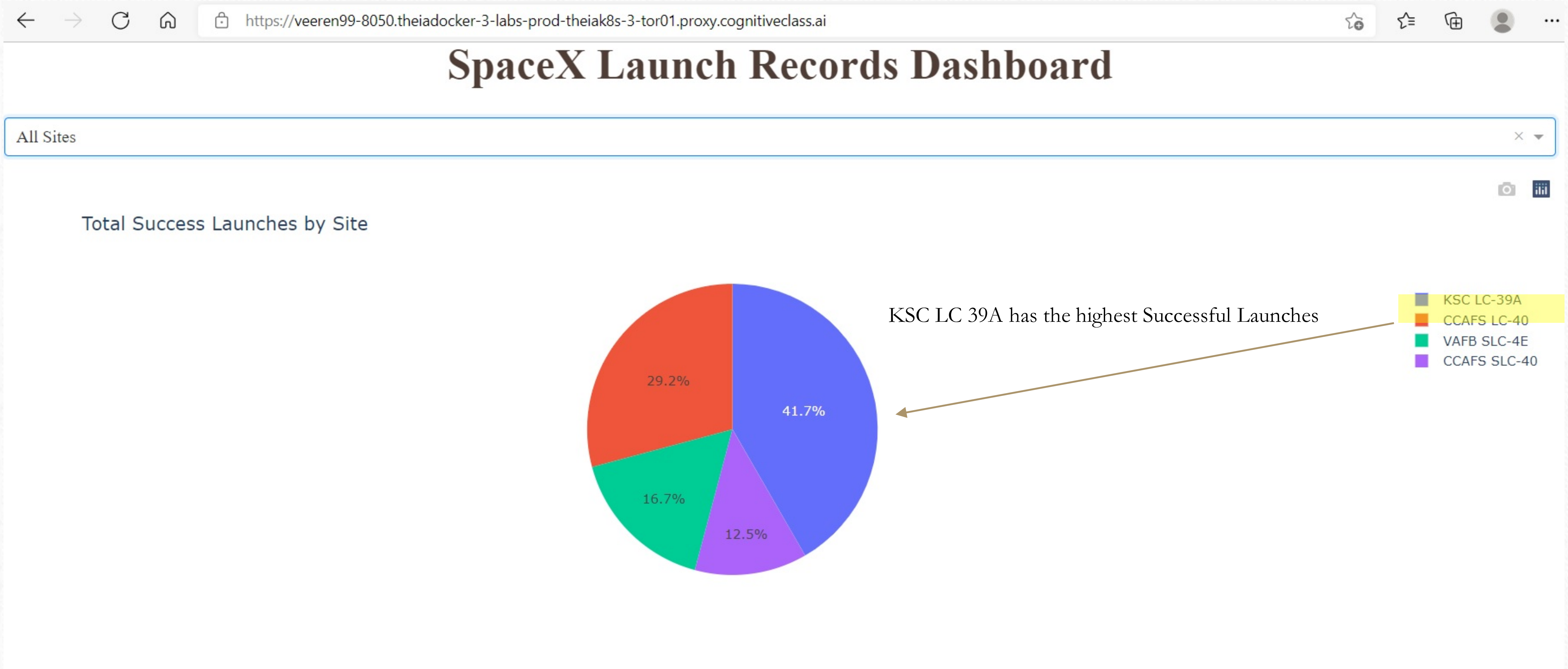
```
HW_Coordinate = [34.66811, -120.55965]
distance_coastline = calculate_distance(Lat, Lng, 34.66811, -120.55965)
HW_distance_marker = folium.Marker(HW_Coordinate, icon=DivIcon(icon_size=(20,20), icon_anchor=(0,0), html='<div style="font-size: 12; color:#d35400;"><b>%s</b></d
#site_map.add_child(circle5)
site_map.add_child(HW_distance_marker)
HW_coordinates = [[34.66811, -120.55965],[Lat, Lng]]
lines=folium.PolyLine(HW_coordinates, weight=1)
site_map.add_child(lines)
```



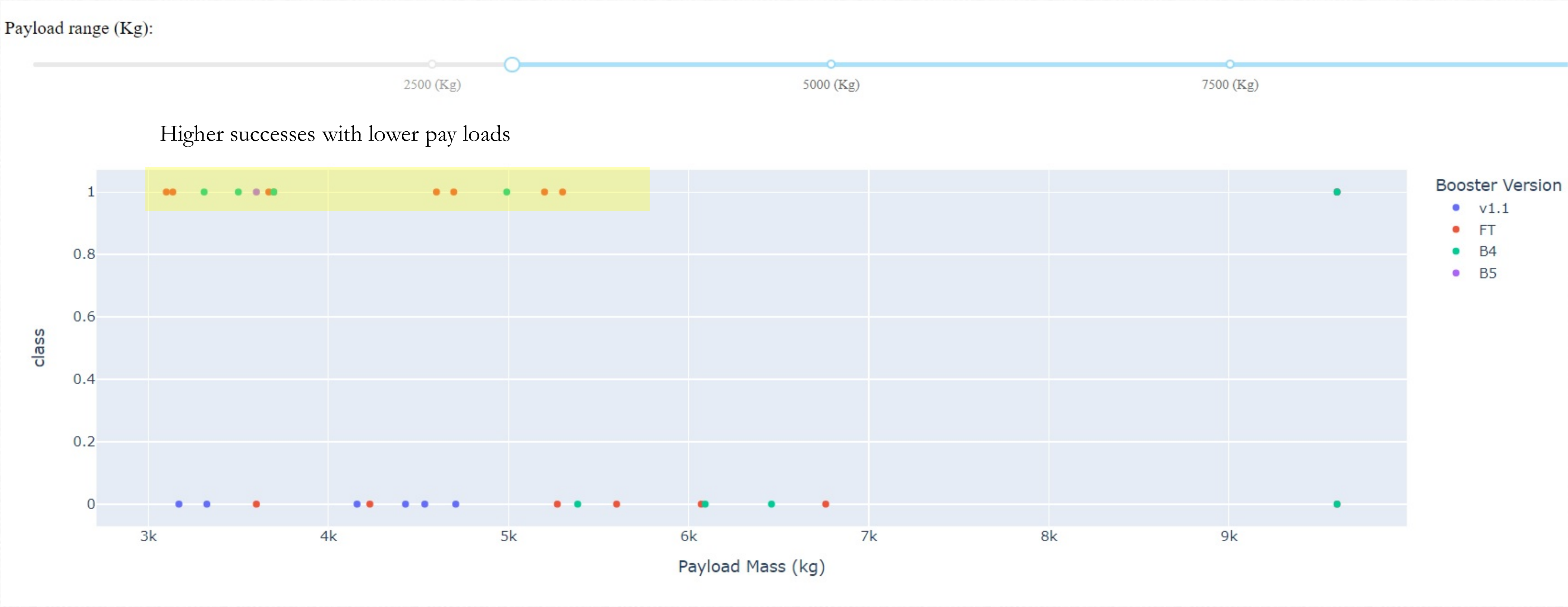
Exploratory data analysis (EDA) visualization with Folium

```
LA_Coordinate = [34.01114, -118.22569]
distance_coastline = calculate_distance(Lat, Lng, 34.01114, -118.22569)
LA_distance_marker = folium.Marker(LA_Coordinate, icon=DivIcon(icon_size=(20,20), icon_anchor=(0,0), html='<div style="font-size: 12; color:#d35400;"><b>%s</b></
#site_map.add_child(circle5)
site_map.add_child(LA_distance_marker)
LA_coordinates = [[34.01114, -118.22569],[Lat, Lng]]
lines=folium.PolyLine(LA_coordinates, weight=1)
site_map.add_child(lines)
```

Exploratory data analysis (EDA) visualization with Plotly



Exploratory data analysis (EDA) visualization with Plotly



Classification Predictive Analytics – Logistic Regression

```
In [10]: from sklearn.model_selection import GridSearchCV
```

```
In [11]: parameters ={'C':[0.01,0.1,1],
                      'penalty':['l2'],
                      'solver':['lbfgs']}
```

```
In [12]: from sklearn.linear_model import LogisticRegression

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

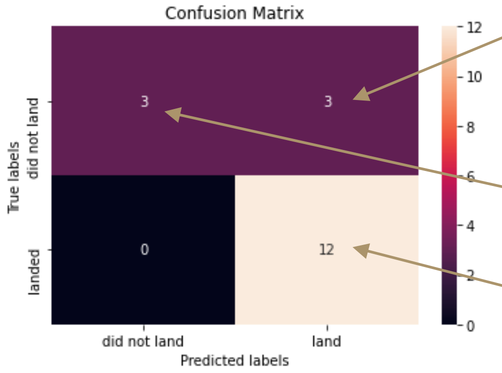
```
tuned hpyerparameters :(best parameters)  {'C': 0.1, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8196428571428571
```

```
In [13]: logreg_cv.score(X_test, Y_test)
         print("accuracy :",logreg_cv.best_score_)
```

```
accuracy : 0.8196428571428571
```

Lets look at the confusion matrix:

```
In [14]: yhat=logreg_cv.predict(X_test)
         plot_confusion_matrix(Y_test,yhat)
```



Identified 3 failures as Success - Incorrect

Identified 3 failures correctly

Identified all 12 Successes correctly

Classification Predictive Analytics – Decision Tree Classifier

In [15]:
```
from sklearn.tree import DecisionTreeClassifier, export_graphviz

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]}
```

In [16]:
```
tree_cv=GridSearchCV(DecisionTreeClassifier(), parameters, cv=10)
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': 18, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 2, 'splitter': 'best'}
```

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

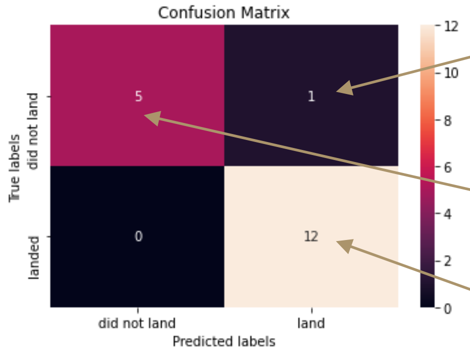
In [17]:
```
tree_cv.fit(X_test,Y_test)
#print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)
```

```
/opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages/sklearn/model_selec
  warnings.warn(("The least populated class in y has only %d"
```

```
accuracy : 0.95
```

We can plot the confusion matrix

In [18]:
```
yhat = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Identified 1 failure as Success - Incorrect

Identified 5 failures correctly

Identified all 12 Successes correctly

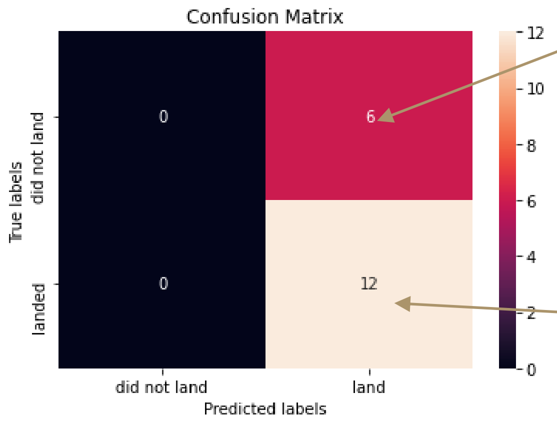
Classification Predictive Analytics – KNN Classifier

```
In [19]: parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                       'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
                       'p': [1,2]}
```

```
In [20]: knn_cv=GridSearchCV(KNeighborsClassifier(), parameters, cv=10)
         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': 3, 'p': 1}
```

```
In [21]: knn_cv.fit(X_test,Y_test)
         #print("tuned hpyerparameters :(best parameters) ",knn_cv.best_params_)
         print("accuracy :",knn_cv.best_score_)

         /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages/sklearn/model_selection/_split.py
           warnings.warn(("The least populated class in y has only %d"
         accuracy : 0.6
```

```
In [23]: yhat = knn_cv.predict(X_test)
         plot_confusion_matrix(Y_test,yhat)
```



Identified 6 failure as Success - Incorrect

Identified all 12 Successes correctly

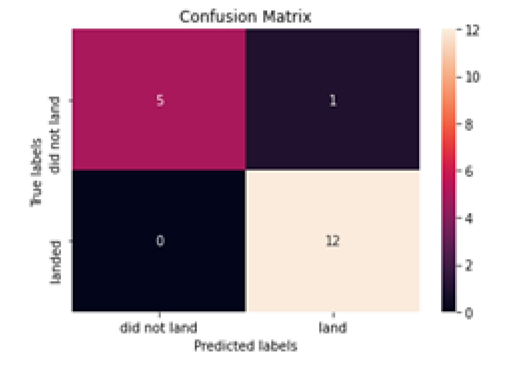
Classification Predictive Analytics – Best Model

```
In [10]: Y_test
Out[10]: array([1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1])
```

The 'Y_test' array has 18 results out of which 12 are successful and 6 are failure

The 'Decision Tree Classifier' with an accuracy score of 0.95 is the most accurate

It has identified 5 out of 6 failures correctly and 12 out of 12 successes correctly



Conclusions…..

- It is observed that the success rate has improved over the years, from 2013 onwards
- It is observed that the success rate is higher with lower payloads
- Orbits like ES-L1, SSO, HEO and GEO has 100% success rate
- Successes were achieved in Drone Ship(5), Controlled Ocean(3), Ground Pad(3) and Uncontrolled Ocean (2)
- KSC LC 39A has more success of the 4 sites, with 10 successes out of 13. This has contributed to 41.7% success of the overall success.

Conclusions…..

- Decision Tree Classifier based model has the best accuracy with score of 0.95
- Decision Tree Classifier has identified all successes(12) and 5 out of 6 failures correctly out of a sample of 18 records in the Y_test.

Launching from KSC LC 39A with lower payloads into one of the orbits (ES-L1, SSO, HEO and GEO) and landing it back to Drone ship has the highest possibility for success

And the way ahead with this…..

- The Predictive model's ability can be further enhanced with the inclusion of following,
  - By recording 'Weather data' during launch
  - By recording 'Weather data' during landing such as wind speed, moisture, air temperature etc
  - Analysis based on the time of the day (launch and landing)
  - By recording the position of the earth in its orbit and position of the moon in its orbit to see weather this information have any impact on the success.
  - By recording the 'Trajectory' information to see weather applying any corrective action during landing will lead to improvement in success rate

Innovative Insights

Rare Insight.....

- Between 00:00:00 to 06:00:00, there were 25 launches with all the mission outcomes as success and 18 landings are success.(72%)
- Between 06:00:00 to 12:00:00, there were 11 launches with all the mission outcomes as success and 6 landings are success.
- Between 12:00:00 to 18:00:00, there were 29 launches with all the mission outcomes as success except one and 17 landings are success.
- Between 18:00:00 to 00:00:00, there were 36 launches with all the mission outcomes as success and 20 landings are success.
- Launching between 00:00:00 to 06:00:00 has higher chances of successful landing