IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Khaled Al-Gumaei
02.01.2024

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion
- Appendix

Executive Summary

The project SpaceY aims to make the space travels affordable for everyone. To this end, we trained a machine learning model and use public information of SpaceX to predict to determine if the first stage will land successfully.

- Summary of methodologies
  - Data collection using API and scraping
  - Data Warngling
  - Exploratory Data Analysis (EDA)
  - Interactive visual analytics and dashboards
  - Predictive Analysis
- Summary
  - Exploratory data analysis results
  - Interactive Analytics results
  - Predictive analytics results

Introduction

Project

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 SpaceY for a rocket launch. we will analyze the Falcon 9 public data to predict if the first stage will land successfully.

Problem

- What factors determine if the rocket will land successfully?
- What the best machine learning model can be used for prediction?
- What operationg conditions need to be in place to ensure successful landing program?

Section 1

Methodology

Methodology

- Data collection methodology:

  Data has been collected using two approaches

  1. SpaceX REST API
  2. Web scraping

- Perform data wrangling

  After collecting the data we check the missing data ,and data types and do on of the following to clean the data :

  1. Replace the missing data with one-Using mean or so.
  2. Change data type of the data.
  3. Represent categorical data using integer or float dummy numbers -one hot encoding

Methodology

- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Data has been preprocssed, standardized, and split into train and test datasets.
  - Data has been analyzed using different machine learning models
  - The accuracy of the different models was evaluated

Data Collection – SpaceX API

In this data collection approach, we

- Requested and parsed the SpaceX launch data using the GET request
- Filtered the dataframe to remove the Falcon 1 launches keeping only the Falcon 9 launches

Now let's start requesting rocket launch data from SpaceX API with the following URL:

In [14]:
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

In [15]:
```
response = requests.get(spacex_url)
```

Check the content of the response

In [16]:
```
print(response.content)
```

```
b'[{"fairings":{"reused":false,"recovery_attempt":false,"recovered":false,"ships":
s2.imgbox.com/3c/0e/T8iJcSN3_o.png","large":"https://images2.imgbox.com/40/e3/GypS
ch":null,"media":null,"recovery":null},"flickr":{"small":[],"original":[]},"pressk
```

Data Collection - Scraping

In this data collection approach, we

- Requested the Falcon9 Launch Wiki page from its URL
- Extract all column/variable names from the HTML table header
- Create a data frame by parsing the launch HTML tables

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

[6]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url)
response
```

[6]:
```
<Response [200]>
```

Create a `BeautifulSoup` object from the HTML `response`

[7]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.text, 'html.parser')
```

Print the page title to verify if the `BeautifulSoup` object was created properly

[8]:
```
# Use soup.title attribute
soup.title
```

[8]:
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

Data Wrangling

In this data collection approach, we

- We dealt with the missing values in the column PayloadMass by calculating the .mean() and replace Null values with this mean using np.nan function
- Calculated the number of launches on each site
- Calculated the number and occurrence of each orbit
- Calculated the number and occurence of mission outcome of the orbits
- Created a landing outcome label from Outcome column

Task 3: Dealing with Missing Values ¶

Calculate below the mean for the `PayloadMass` using the `.mean()`. mean you calculated.

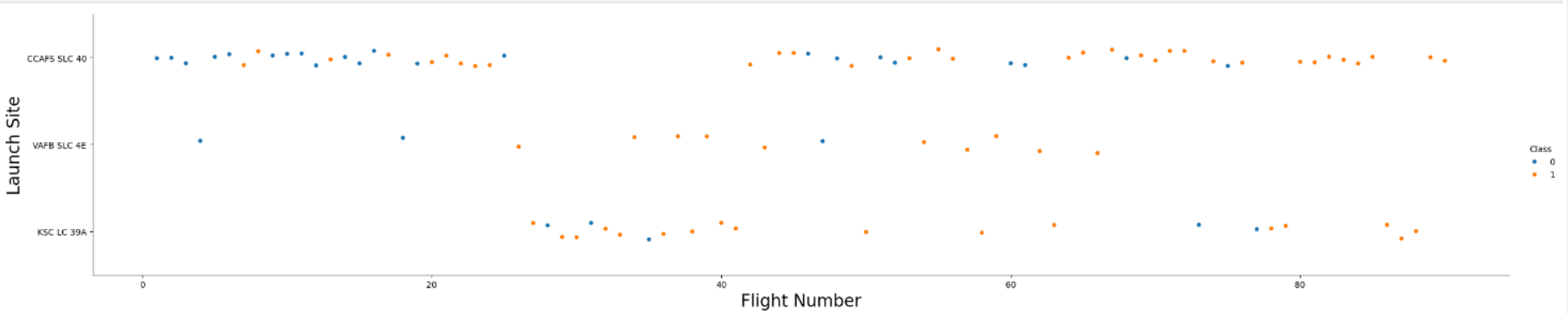
```
# Calculate the mean value of PayloadMass column
PayloadMass_mean = data_falcon9['PayloadMass'].mean()
PayloadMass_mean
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].fillna(value=PayloadMass_mean, inplace=Tru
```

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

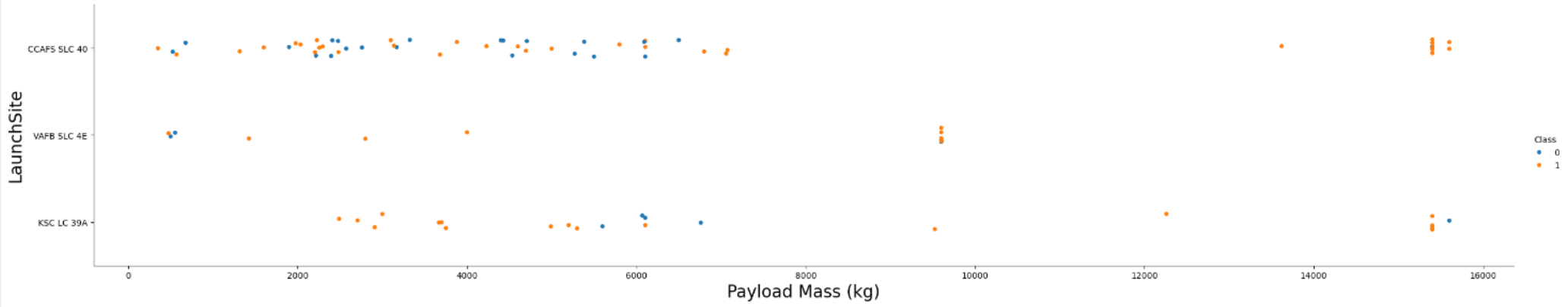
```
FlightNumber      0
Date              0
BoosterVersion    0
PayloadMass       0
Orbit             0
LaunchSite        0
Outcome           0
Flights           0
GridFins          0
Reused            0
Legs              0
LandingPad       26
Block             0
```

EDA with Data Visualization

- Visualize the relationship between Flight Number and Launch Site

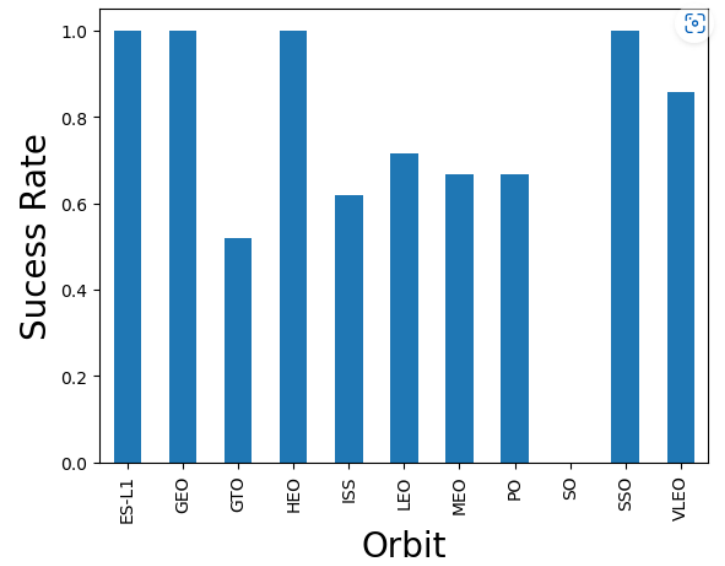


- Visualize the relationship between Payload and Launch Site

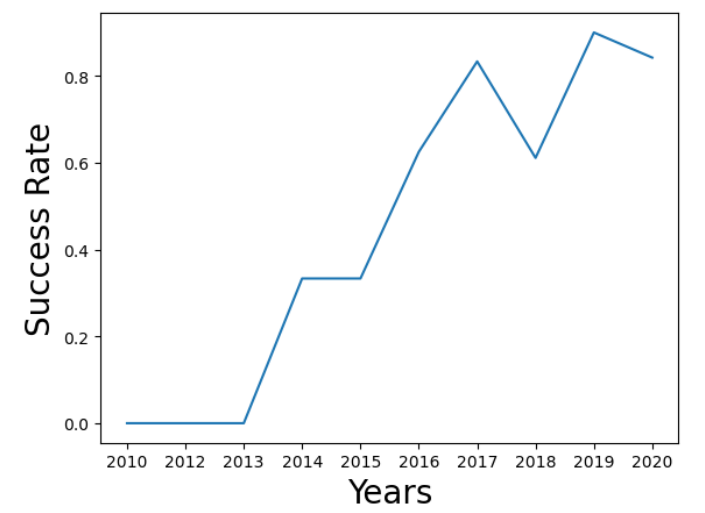


EDA with Data Visualization

- Visualize the relationship between success rate of each orbit type



- Visualize the launch success yearly trend



EDA with SQL

- Dataset includes a record for each payload carried during a SpaceX mission into outer space
- Dataset downloaded as csv file and uploaded to DB2 database

- **Task 1**: Display the names of the unique launch sites in the space mission
- **Task 2**: Display 5 records where launch sites begin with the string 'CCA'
- **Task 3**: Display the total payload mass carried by boosters launched by NASA (CRS)
- **Task 4**: Display average payload mass carried by booster version F9 v1.1
- **Task 5**: List the date when the first successful landing outcome in ground pad was achieved.

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

launch_site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

payloadmass
619967

payloadmass
6138

1
2010-06-04

EDA with SQL

- **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
- **Task 7**: List the total number of successful and failure mission outcomes
- **Task 8**: List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
- **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
- **Task 10**: Rank the count of successful landing_outcomes between the date 2010-06-04 and 2017-03-20 in descending order.

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

missionoutcomes
1
99
1

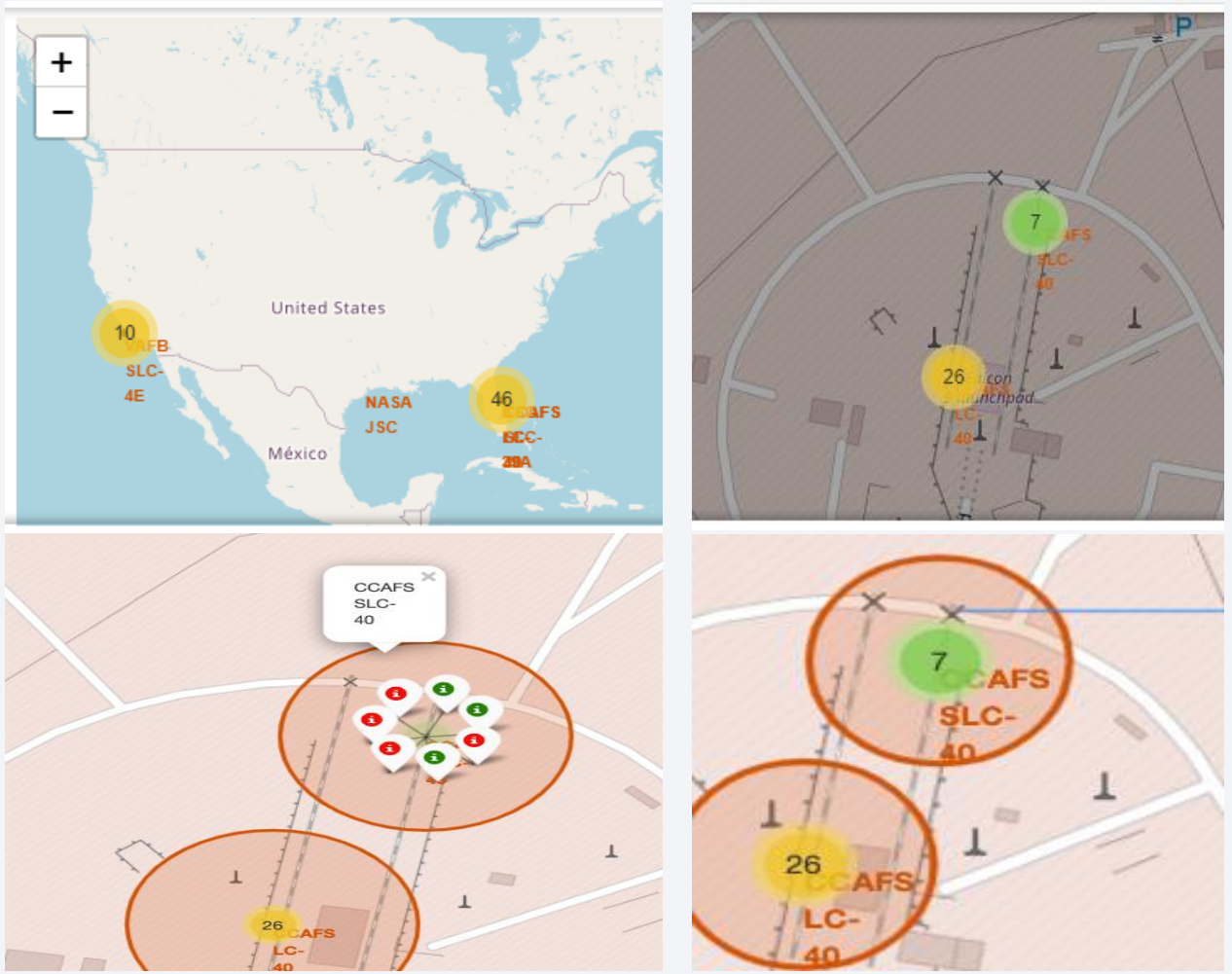
boosterversion
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

landing_outcome
No attempt
Success (ground pad)
Success (drone ship)
Success (drone ship)
Success (ground pad)
Failure (drone ship)
Success (drone ship)
Success (drone ship)
Success (drone ship)
Failure (drone ship)
Failure (drone ship)
Success (ground pad)
Precluded (drone ship)
No attempt
Failure (drone ship)
No attempt
Controlled (ocean)
Failure (drone ship)
Uncontrolled (ocean)
No attempt
No attempt
Controlled (ocean)
Controlled (ocean)
No attempt
No attempt
Uncontrolled (ocean)
No attempt
No attempt
No attempt

1	mission_outcome	booster_version	launch_site
1	Success	F9 v1.1 B1012	CCAFS LC-40
2	Success	F9 v1.1 B1013	CCAFS LC-40
3	Success	F9 v1.1 B1014	CCAFS LC-40
4	Success	F9 v1.1 B1015	CCAFS LC-40
4	Success	F9 v1.1 B1016	CCAFS LC-40
6	Failure (in flight)	F9 v1.1 B1018	CCAFS LC-40
12	Success	F9 FT B1019	CCAFS LC-40

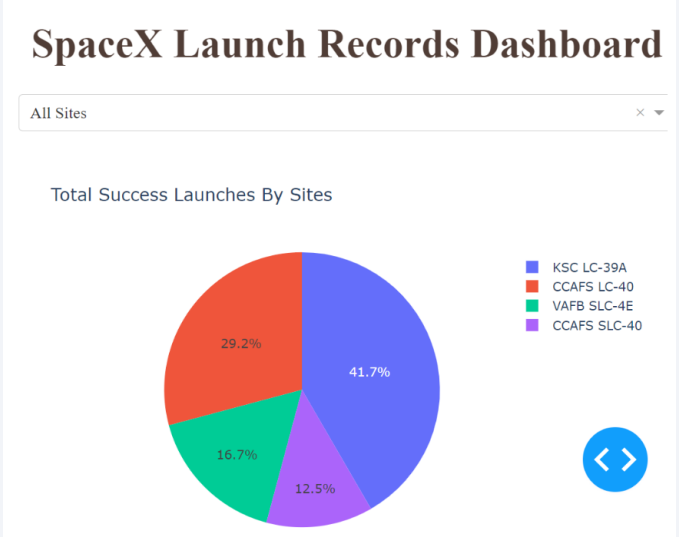
Build an Interactive Map with Folium

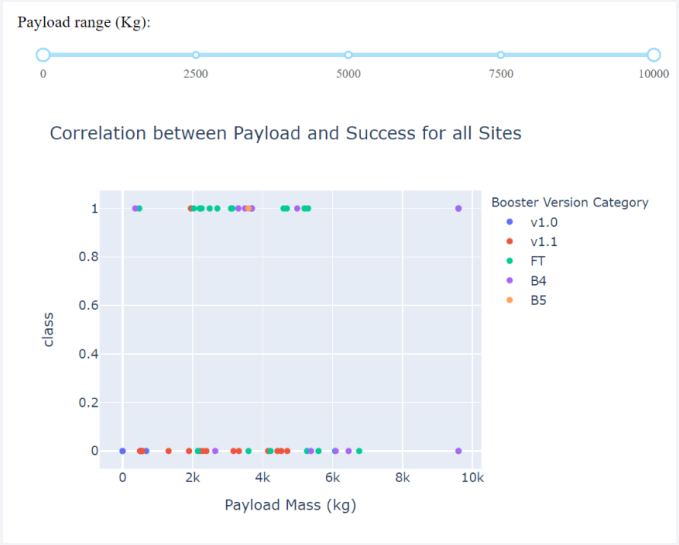
- Mark all launch sites on a map
- Mark the success/failed launches for each site on the map
- Calculate the distances between a launch site to its proximities



Build a Dashboard with Plotly Dash

- Download a skeleton dashboard application and dataset
- Add a Launch Site Drop-down Input Component
- Add a callback function to render success-pie-chart based on selected site dropdown
- Add a Range Slider to Select Payload
- Add a callback function to render the success-payload-scatter-chart scatter plot





Predictive Analysis (Classification)

Data Preperation

- Load the data into a dataframe
- Create a NumPy array from the column Class
- Standardize the data in X then reassign it to the variable X using the provided transformer.
- Use the function train_test_split to split the data X and Y into training and test data

```
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)
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,)
```

we can see we only have 18 test samples.

Predictive Analysis (Classification)

Model Training and Evaluation

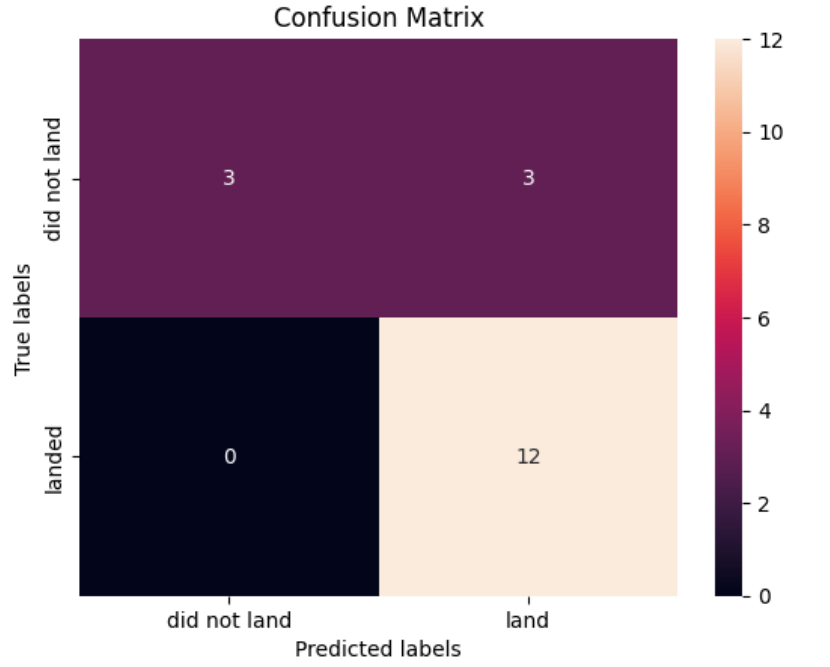
1. **logistic regression**

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

```
0.8333333333333334
```

Lets look at the confusion matrix:

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



Predictive Analysis (Classification)

Model Training and Evaluation

1. **Support vector machine (SVM)**

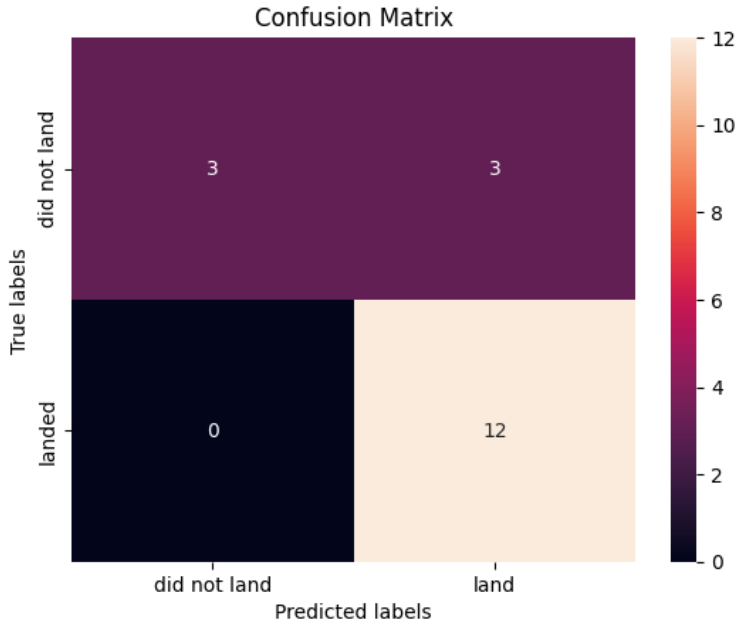
Calculate the accuracy on the test data using the method `score`:

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

```
0.8333333333333334
```

We can plot the confusion matrix

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



Predictive Analysis (Classification)

Model Training and Evaluation

1. Decision Tree Classifier

tuned hpyerparameters :(best parameters) {'criterion': 'entropy', 'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'best'}
accuracy : 0.875

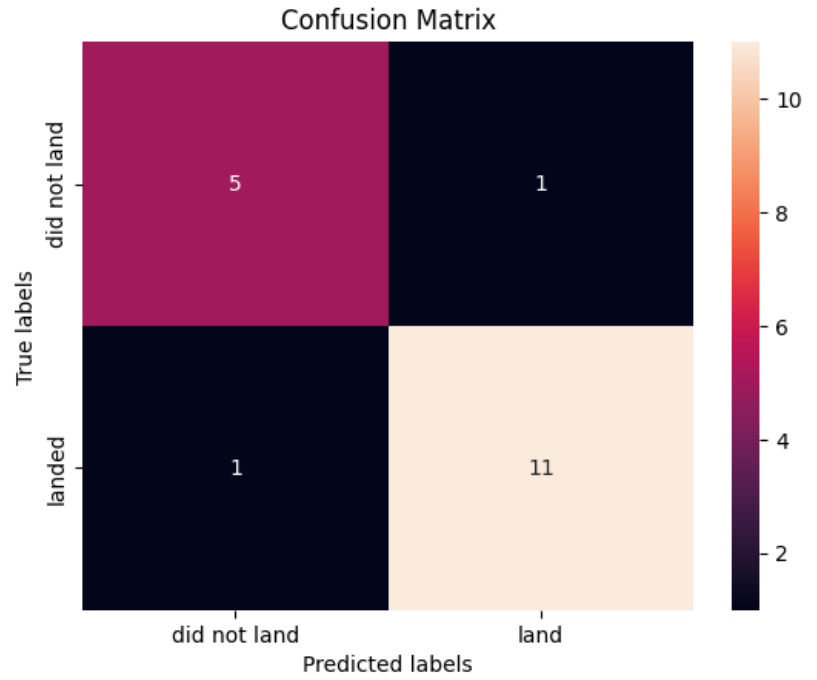
Calculate the accuracy of tree_cv on the test data using the method `score` :

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

```
0.8888888888888888
```

We can plot the confusion matrix

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



Predictive Analysis (Classification)

Model Training and Evaluation

1. **K nearest neighbors**

```
{'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
```

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

```
tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
accuracy : 0.8482142857142858
```

Results & Conclusions

Result

- My insights from the analysis is that most launches were from Kennedy Space center .The reason behind this includes: It’s near SpaceX production factory.
- Most launches were form KSC PAD 39A since most of them were to VLEO,GEO or ISS which makes it a good site to launch from .
- Falcon heavy launches mostly to full payload to maximize use of the falcon payload capacity .
- Probability of booster landing increases of time by use of the data collected from failing .
- SpaceX first successful booster Landing happened on 06/05/2016.

Conclusion

Using Existing Data and Analyzing the data ,SpaceX and other rocket companies can be able to see the best way to reduce the cost of launches, and evolve before there tradition costly launches lead to their absoluteness and losing their client .

Thank you!