Skills Network

(https://skills.network/?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=1
SkillsNetwork-Channel-SkillsNetworkCoursesIBMDS0321ENSkillsNetwork26802033-2022-01-01)

Space X Falcon 9 First Stage Landing Prediction

Assignment: Machine Learning Prediction

Estimated time needed: **60** minutes

Space X 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 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 space X for a rocket launch. In this lab, you will create a machine learning pipeline to predict if the first stage will land given the data from the preceding labs.

Several examples of an unsuccessful landing are shown here:

Most unsuccessful landings are planed. Space X; performs a controlled landing in the oceans.

Objectives

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

Import Libraries and Define Auxiliary Functions

In [ ]:
```
import piplite
await piplite.install(['numpy'])
await piplite.install(['pandas'])
await piplite.install(['seaborn'])
```

We will import the following libraries for the lab

In [1]:

```
# Pandas is a software library written for the Python programming langu
import pandas as pd
# NumPy is a library for the Python programming language, adding suppor
import numpy as np
# Matplotlib is a plotting library for python and pyplot gives us a Mat
import matplotlib.pyplot as plt
#Seaborn is a Python data visualization library based on matplotlib. It
import seaborn as sns
# Preprocessing allows us to standarsize our data
from sklearn import preprocessing
# Allows us to split our data into training and testing data
from sklearn.model_selection import train_test_split
# Allows us to test parameters of classification algorithms and find th
from sklearn.model_selection import GridSearchCV
# Logistic Regression classification algorithm
from sklearn.linear_model import LogisticRegression
# Support Vector Machine classification algorithm
from sklearn.svm import SVC
# Decision Tree classification algorithm
from sklearn.tree import DecisionTreeClassifier
# K Nearest Neighbors classification algorithm
from sklearn.neighbors import KNeighborsClassifier
```

This function is to plot the confusion matrix.

In [2]:

```
def plot_confusion_matrix(y,y_predict):
    "this function plots the confusion matrix"
    from sklearn.metrics import confusion_matrix

    cm = confusion_matrix(y, y_predict)
    ax= plt.subplot()
    sns.heatmap(cm, annot=True, ax = ax); #annot=True to annotate cells
    ax.set_xlabel('Predicted labels')
    ax.set_ylabel('True labels')
    ax.set_title('Confusion Matrix');
    ax.xaxis.set_ticklabels(['did not land', 'land']); ax.yaxis.set_tic
```

In [4]:

```
pip install js
```

```
Collecting jsNote: you may need to restart the kernel to use updated p
ackages.

  Downloading js-1.0.tar.gz (2.5 kB)
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Collecting fanstatic
  Downloading fanstatic-1.3-py3-none-any.whl (32 kB)
Requirement already satisfied: setuptools in c:\users\danie\anaconda3
\lib\site-packages (from js) (63.4.1)
Collecting WebOb>=1.2
  Downloading WebOb-1.8.7-py2.py3-none-any.whl (114 kB)
     -------------------------------------- 115.0/115.0 kB 1.7 MB/s et
a 0:00:00
Building wheels for collected packages: js
  Building wheel for js (setup.py): started
  Building wheel for js (setup.py): finished with status 'done'
  Created wheel for js: filename=js-1.0-py3-none-any.whl size=2884 sha
256=4fc02b494e5f08a0beea00143afcbbb86453ed82ccfa1a9917f1ada72920e75d
  Stored in directory: c:\users\danie\appdata\local\pip\cache\wheels\6
f\91\12\9fc79cc62b07127faf39b5f3afcc6606e659bb54743a00bebb
Successfully built js
Installing collected packages: WebOb, fanstatic, js
Successfully installed WebOb-1.8.7 fanstatic-1.3 js-1.0

WARNING: Retrying (Retry(total=4, connect=None, read=None, redirect=No
ne, status=None)) after connection broken by 'ProtocolError('Connectio
n aborted.', ConnectionResetError(10054, 'An existing connection was f
orcibly closed by the remote host', None, 10054, None))': /simple/js/
```

Load the dataframe

Load the data

In [6]:

```
# from js import fetch
# import io

URL1 = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cl
# resp1 = await fetch(URL1)
# text1 = io.BytesIO((await resp1.arrayBuffer()).to_py())
# data = pd.read_csv(text1)
data = pd.read_csv(URL1)
```

In [7]:

```
data.head()
```

Out[7]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flig
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	

In [8]:

```
URL2 = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cl
# resp2 = await fetch(URL2)
# text2 = io.BytesIO((await resp2.arrayBuffer()).to_py())
# X = pd.read_csv(text2)
X = pd.read_csv(URL2)
```

In [9]:
```
X.head(100)
```

Out[9]:

	FlightNumber	PayloadMass	Flights	Block	ReusedCount	Orbit_ES-L1	Orbit_GEO	Oı
0	1.0	6104.959412	1.0	1.0	0.0	0.0	0.0	
1	2.0	525.000000	1.0	1.0	0.0	0.0	0.0	
2	3.0	677.000000	1.0	1.0	0.0	0.0	0.0	
3	4.0	500.000000	1.0	1.0	0.0	0.0	0.0	
4	5.0	3170.000000	1.0	1.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	
85	86.0	15400.000000	2.0	5.0	2.0	0.0	0.0	
86	87.0	15400.000000	3.0	5.0	2.0	0.0	0.0	
87	88.0	15400.000000	6.0	5.0	5.0	0.0	0.0	
88	89.0	15400.000000	3.0	5.0	2.0	0.0	0.0	
89	90.0	3681.000000	1.0	5.0	0.0	0.0	0.0	

90 rows × 83 columns

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']).

In [10]:
```
Y = pd.Series(data['Class'].to_numpy())
```

In [12]:
```python
print(type(Y))
print(Y.head(100))
```

```
<class 'pandas.core.series.Series'>
0     0
1     0
2     0
3     0
4     0
     ..
85    1
86    1
87    1
88    1
89    1
Length: 90, dtype: int64
```

TASK 2

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

In [13]:
```python
# students get this
transform = preprocessing.StandardScaler()
X = transform.fit_transform(X)
```

In [15]:
```python
X
```

Out[15]:
```
array([[-1.71291154e+00, -1.94814463e-16, -6.53912840e-01, ...,
        -8.35531692e-01,  1.93309133e+00, -1.93309133e+00],
       [-1.67441914e+00, -1.19523159e+00, -6.53912840e-01, ...,
        -8.35531692e-01,  1.93309133e+00, -1.93309133e+00],
       [-1.63592675e+00, -1.16267307e+00, -6.53912840e-01, ...,
        -8.35531692e-01,  1.93309133e+00, -1.93309133e+00],
       ...,
       [ 1.63592675e+00,  1.99100483e+00,  3.49060516e+00, ...,
         1.19684269e+00, -5.17306132e-01,  5.17306132e-01],
       [ 1.67441914e+00,  1.99100483e+00,  1.00389436e+00, ...,
         1.19684269e+00, -5.17306132e-01,  5.17306132e-01],
       [ 1.71291154e+00, -5.19213966e-01, -6.53912840e-01, ...,
        -8.35531692e-01, -5.17306132e-01,  5.17306132e-01]])
```

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
```

In [16]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2
```

we can see we only have 18 test samples.

In [17]:
```
Y_test.shape
```

Out[17]:
```
(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` .

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

In [18]:
```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1
lr=LogisticRegression()
```

In [19]:
```
logreg_cv = GridSearchCV(lr, parameters, cv=10)
logreg_cv.fit(X_train, Y_train)
```

Out[19]:
```
GridSearchCV(cv=10, estimator=LogisticRegression(),
             param_grid={'C': [0.01, 0.1, 1], 'penalty': ['l2'],
                         'solver': ['lbfgs']})
```

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_` .

In [20]:
```
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

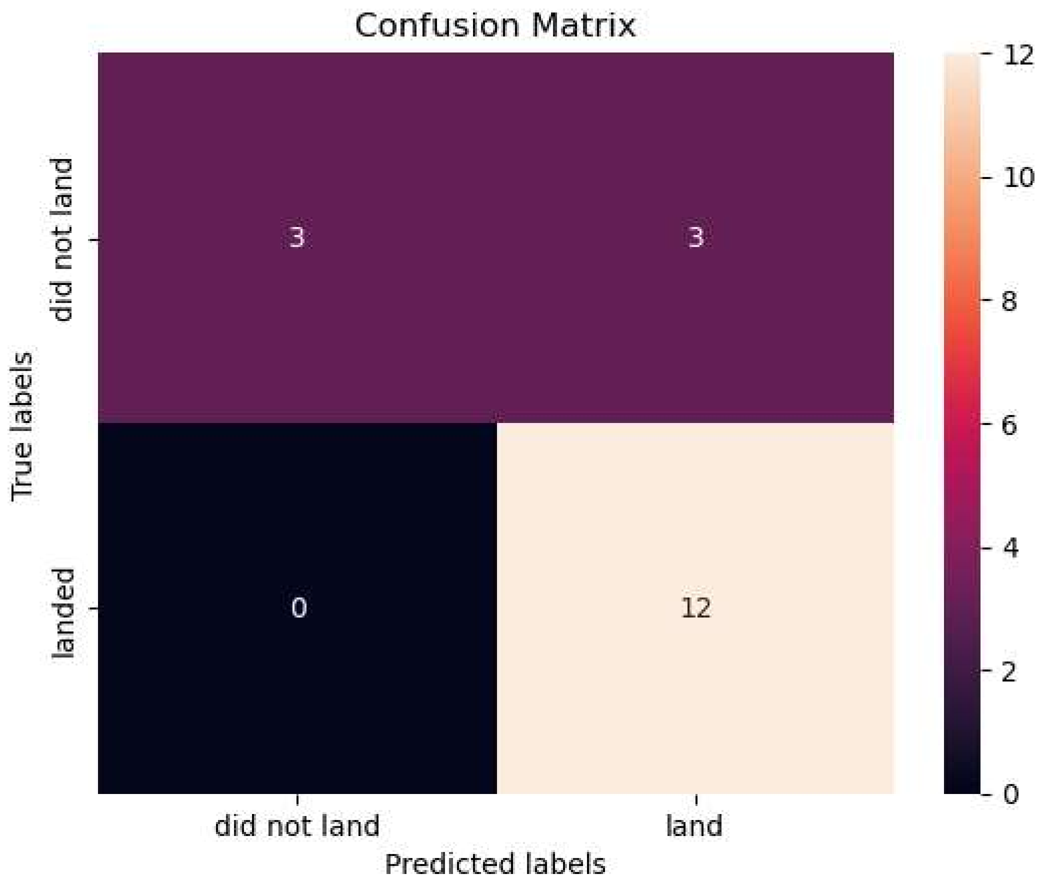
Calculate the accuracy on the test data using the method `score` :

In [21]:
```
test_accuracy = logreg_cv.score(X_test, Y_test)


print("Test Accuracy:", test_accuracy)
```

```
Test Accuracy: 0.8333333333333334
```

Lets look at the confusion matrix:

In [22]:
```
yhat=logreg_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Examining the confusion matrix, we see that logistic regression can distinguish between the different classes. We see that the major problem is false positives.

TASK 6

Create a support vector machine object then create a `GridSearchCV` object `svm_cv` with cv - 10. Fit the object to find the best parameters from the dictionary `parameters` .

In [24]:
```
parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
              'C': np.logspace(-3, 3, 5),
              'gamma':np.logspace(-3, 3, 5)}
svm = SVC()
```

In [25]:
```
# Create a GridSearchCV object with 10-fold cross-validation
svm_cv = GridSearchCV(svm, parameters, cv=10)

# Fit the GridSearchCV object to the training data
svm_cv.fit(X_train, Y_train)
```

Out[25]:
```
GridSearchCV(cv=10, estimator=SVC(),
             param_grid={'C': array([1.00000000e-03, 3.16227766e-02,
1.00000000e+00, 3.16227766e+01,
       1.00000000e+03]),
                         'gamma': array([1.00000000e-03, 3.16227766e-0
2, 1.00000000e+00, 3.16227766e+01,
       1.00000000e+03]),
                         'kernel': ('linear', 'rbf', 'poly', 'rbf', 's
igmoid')})
```

In [26]:
```
print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
print("accuracy :",svm_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.031622
77660168379, 'kernel': 'sigmoid'}
accuracy : 0.8482142857142856
```

TASK 7

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

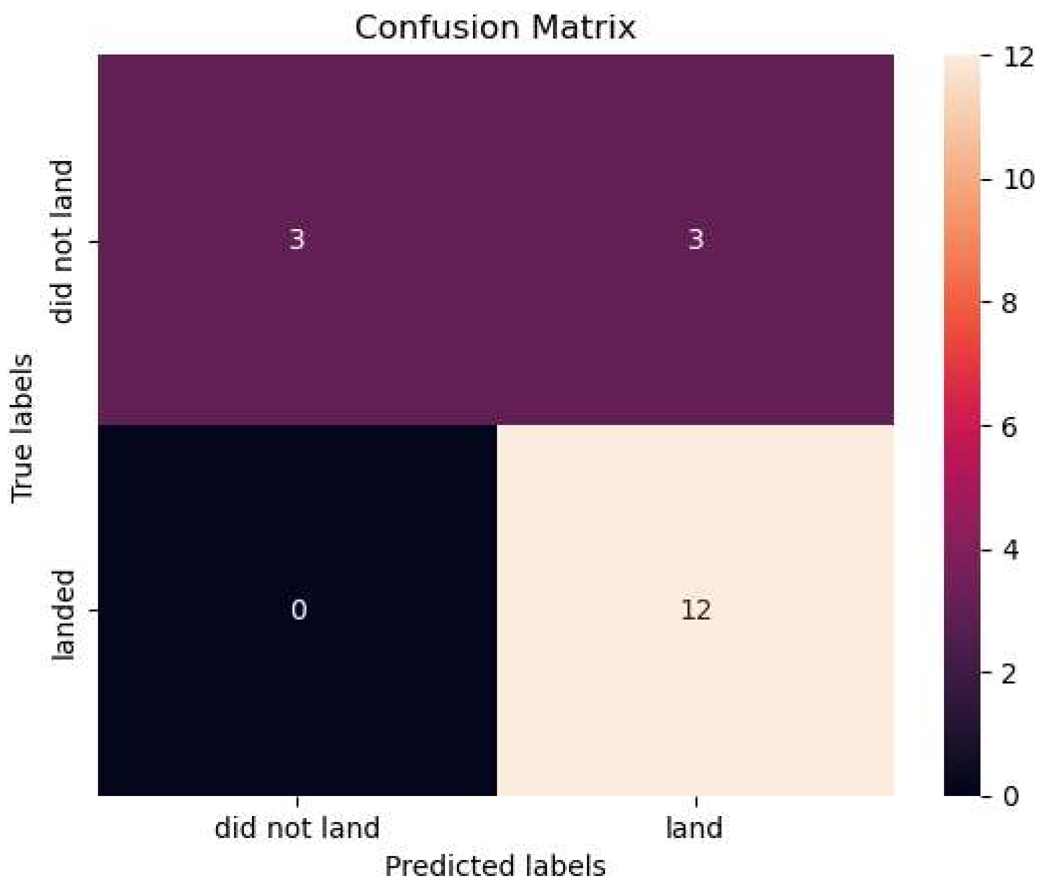
In [27]:
```
test_accuracy = svm_cv.score(X_test, Y_test)

print("Test Accuracy:", test_accuracy)
```

```
Test Accuracy: 0.8333333333333334
```

We can plot the confusion matrix

In [28]:

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



TASK 8

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` .

In [29]:

```
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()
```

In [30]:

```
# Create a GridSearchCV object with 10-fold cross-validation
tree_cv = GridSearchCV(tree, parameters, cv=10)

# Fit the GridSearchCV object to the training data
tree_cv.fit(X_train, Y_train)
```

Out[30]:

```
GridSearchCV(cv=10, estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 4, 6, 8, 10, 12, 14, 16, 1
8],
                         'max_features': ['auto', 'sqrt'],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [2, 5, 10],
                         'splitter': ['best', 'random']})
```

In [31]:

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

```
tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'ma
x_depth': 4, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_sampl
es_split': 10, 'splitter': 'random'}
accuracy : 0.8892857142857145
```

TASK 9

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

In [32]:

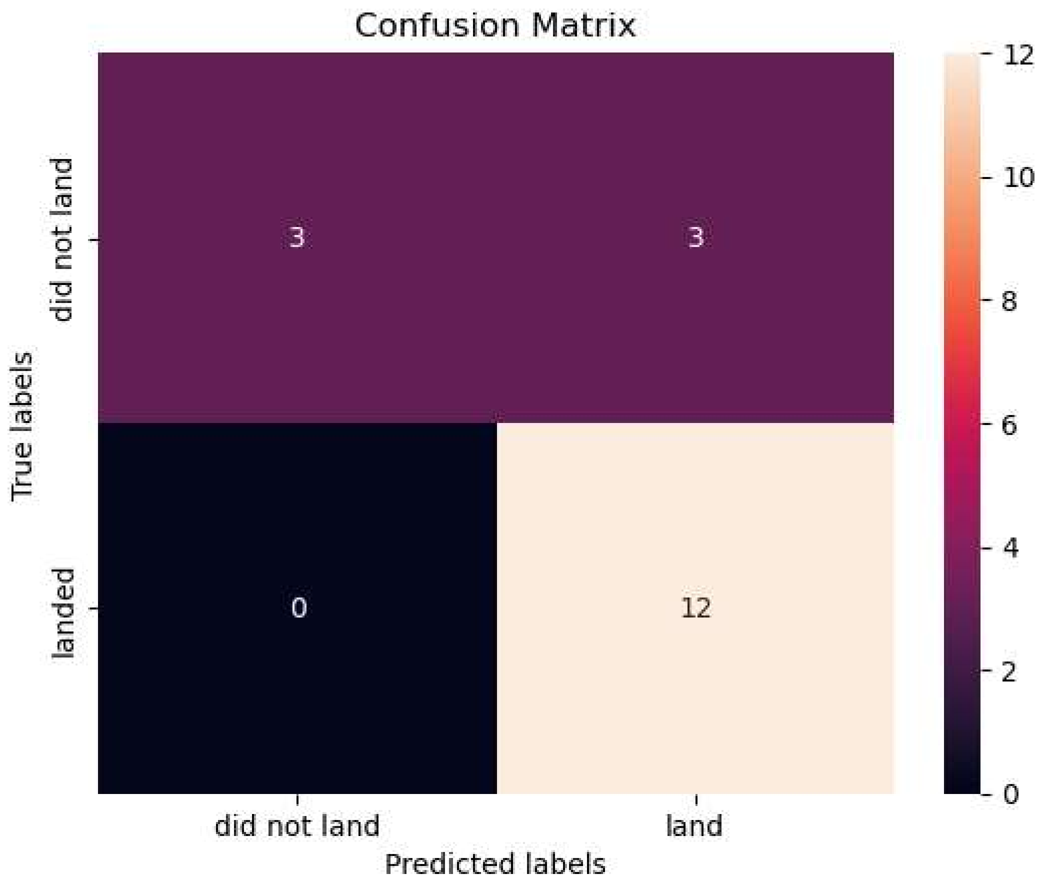
```
test_accuracy = tree_cv.score(X_test, Y_test)
print("Test Accuracy:", test_accuracy)
```

```
Test Accuracy: 0.8333333333333334
```

We can plot the confusion matrix

In [33]:

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



TASK 10

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` .

In [34]:

```
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()
```

In [35]:

```
# Create a GridSearchCV object with 10-fold cross-validation
knn_cv = GridSearchCV(KNN, parameters, cv=10)

# Fit the GridSearchCV object to the training data
knn_cv.fit(X_train, Y_train)
```

```
C:\Users\danie\anaconda3\lib\site-packages\sklearn\neighbors\_cla
ssification.py:228: FutureWarning: Unlike other reduction functio
ns (e.g. `skew`, `kurtosis`), the default behavior of `mode` typi
cally preserves the axis it acts along. In SciPy 1.11.0, this beh
avior will change: the default value of `keepdims` will become Fa
lse, the `axis` over which the statistic is taken will be elimina
ted, and the value None will no longer be accepted. Set `keepdims
` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
C:\Users\danie\anaconda3\lib\site-packages\sklearn\neighbors\_cla
ssification.py:228: FutureWarning: Unlike other reduction functio
ns (e.g. `skew`, `kurtosis`), the default behavior of `mode` typi
cally preserves the axis it acts along. In SciPy 1.11.0, this beh
avior will change: the default value of `keepdims` will become Fa
lse, the `axis` over which the statistic is taken will be elimina
ted, and the value None will no longer be accepted. Set `keepdims
` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
C:\Users\danie\anaconda3\lib\site-packages\sklearn\neighbors\_cla
```

In [36]:

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

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

TASK 11

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

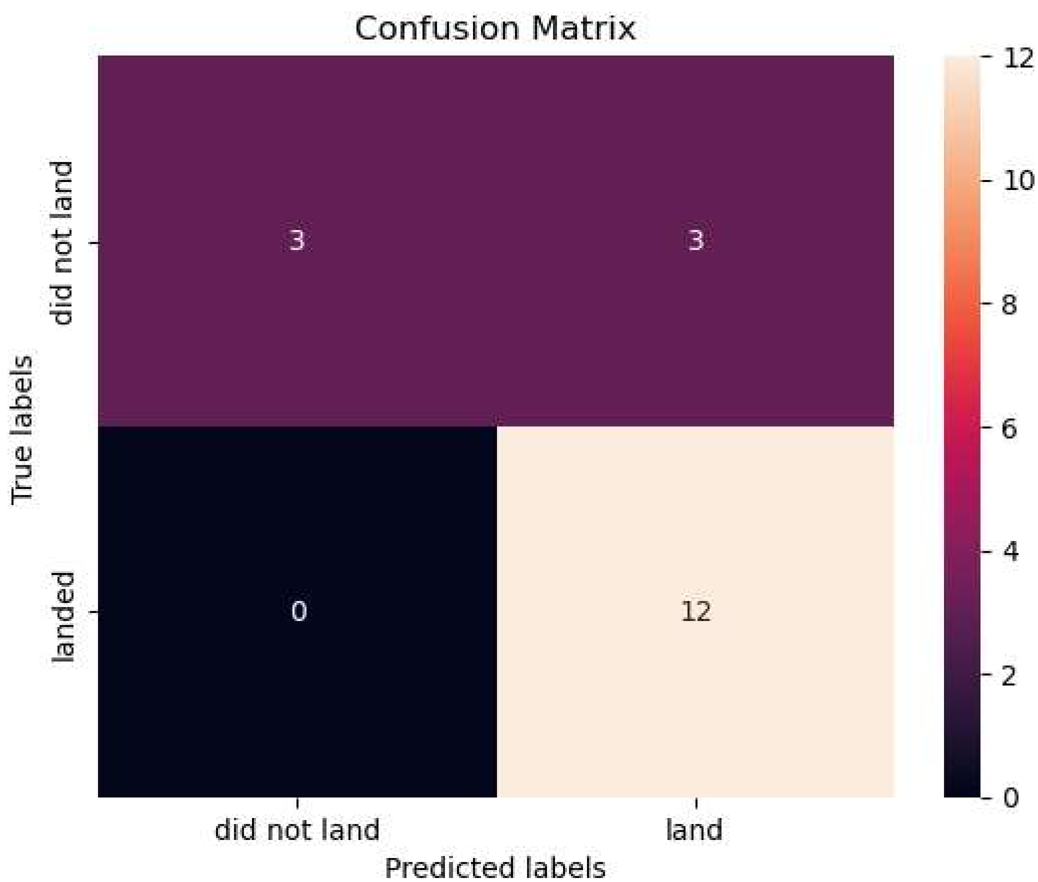
In [37]:
```
test_accuracy = knn_cv.score(X_test, Y_test)

print("Test Accuracy:", test_accuracy)
```

```
Test Accuracy: 0.8333333333333334

C:\Users\danie\anaconda3\lib\site-packages\sklearn\neighbors\_classifi
cation.py:228: FutureWarning: Unlike other reduction functions (e.g. `
skew`, `kurtosis`), the default behavior of `mode` typically preserves
the axis it acts along. In SciPy 1.11.0, this behavior will change: th
e default value of `keepdims` will become False, the `axis` over which
the statistic is taken will be eliminated, and the value None will no
longer be accepted. Set `keepdims` to True or False to avoid this warn
ing.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

We can plot the confusion matrix

In [38]:
```
yhat = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

```
C:\Users\danie\anaconda3\lib\site-packages\sklearn\neighbors\_classifi
cation.py:228: FutureWarning: Unlike other reduction functions (e.g. `
skew`, `kurtosis`), the default behavior of `mode` typically preserves
the axis it acts along. In SciPy 1.11.0, this behavior will change: th
e default value of `keepdims` will become False, the `axis` over which
the statistic is taken will be eliminated, and the value None will no
longer be accepted. Set `keepdims` to True or False to avoid this warn
ing.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```



TASK 12

Find the method performs best:

In [40]:

```
# Print the test accuracies
logreg_acc = logreg_cv.score(X_test, Y_test)
svm_acc = svm_cv.score(X_test, Y_test)
tree_acc = tree_cv.score(X_test, Y_test)
knn_acc = knn_cv.score(X_test, Y_test)
print("Logistic Regression Accuracy:", logreg_acc)
print("SVM Accuracy:", svm_acc)
print("Decision Tree Accuracy:", tree_acc)
print("K-NN Accuracy:", knn_acc)
```

```
Logistic Regression Accuracy: 0.8333333333333334
SVM Accuracy: 0.8333333333333334
Decision Tree Accuracy: 0.8333333333333334
K-NN Accuracy: 0.8333333333333334

C:\Users\danie\anaconda3\lib\site-packages\sklearn\neighbors\_classifi
cation.py:228: FutureWarning: Unlike other reduction functions (e.g. `
skew`, `kurtosis`), the default behavior of `mode` typically preserves
the axis it acts along. In SciPy 1.11.0, this behavior will change: th
e default value of `keepdims` will become False, the `axis` over which
the statistic is taken will be eliminated, and the value None will no
longer be accepted. Set `keepdims` to True or False to avoid this warn
ing.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

Authors

[Pratiksha Verma (https://www.linkedin.com/in/pratiksha-verma-6487561b1/?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=1(
SkillsNetwork-Channel-SkillsNetworkCoursesIBMDS0321ENSkillsNetwork865-2022-01-01)

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2022-11-09	1.0	Pratiksha Verma	Converted initial version to Jupyterlite