codsoft-ml1-1

August 5, 2024

TASK-1 CREDIT CARD DETECTION

[1]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

[2]:
```
# loading the dataset to a Pandas DataFrame
credit_card_data = pd.read_csv('/content/creditcard (1) (1).csv')
```

[3]:
```
credit_card_data.head(5)
```

[3]:
```
   Time        V1        V2        V3        V4        V5        V6        V7  \
0     0 -1.359807 -0.072781  2.536347  1.378155 -0.338321  0.462388  0.239599
1     0  1.191857  0.266151  0.166480  0.448154  0.060018 -0.082361 -0.078803
2     1 -1.358354 -1.340163  1.773209  0.379780 -0.503198  1.800499  0.791461
3     1 -0.966272 -0.185226  1.792993 -0.863291 -0.010309  1.247203  0.237609
4     2 -1.158233  0.877737  1.548718  0.403034 -0.407193  0.095921  0.592941

         V8        V9  …       V21       V22       V23       V24       V25  \
0  0.098698  0.363787  … -0.018307  0.277838 -0.110474  0.066928  0.128539
1  0.085102 -0.255425  … -0.225775 -0.638672  0.101288 -0.339846  0.167170
2  0.247676 -1.514654  …  0.247998  0.771679  0.909412 -0.689281 -0.327642
3  0.377436 -1.387024  … -0.108300  0.005274 -0.190321 -1.175575  0.647376
4 -0.270533  0.817739  … -0.009431  0.798278 -0.137458  0.141267 -0.206010

        V26       V27       V28  Amount  Class
0 -0.189115  0.133558 -0.021053  149.62    0.0
1  0.125895 -0.008983  0.014724    2.69    0.0
2 -0.139097 -0.055353 -0.059752  378.66    0.0
3 -0.221929  0.062723  0.061458  123.50    0.0
4  0.502292  0.219422  0.215153   69.99    0.0

[5 rows x 31 columns]
```

[4]:
```
credit_card_data.shape
```

[4]: (5848, 31)

[5]:
```
features = credit_card_data.iloc[:,0:-1]
labels = credit_card_data.iloc[:,-1]
print(features.shape)
print(labels.shape)
```

```
(5848, 30)
(5848,)
```

[6]:
```
# Handle missing values in the target variable (labels) before splitting
import pandas as pd
from sklearn.impute import SimpleImputer

# Impute missing values in the target variable using the most frequent value
imputer = SimpleImputer(strategy='most_frequent')
labels = imputer.fit_transform(labels.values.reshape(-1, 1))  # Reshape for␣
 ↪single feature imputation
labels = pd.Series(labels.flatten())  # Convert back to Series

# Now split the data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(features, labels,␣
 ↪test_size=0.2, random_state=0)
```

[7]:
```
# Splitting the data into training and testing sets
x_train,x_test,y_train,y_test = train_test_split(features,labels,test_size=0.
 ↪2,random_state=0)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(4678, 30)
(1170, 30)
(4678,)
(1170,)
```

[8]:
```
#Building model
from sklearn import tree
decision_tree_model = tree.DecisionTreeClassifier()
decision_tree_model.fit(x_train,y_train)
```

[8]: DecisionTreeClassifier()

[9]:
```
# Handle missing values before model prediction
import pandas as pd
from sklearn.impute import SimpleImputer
```

```python

# Impute missing values using the mean strategy
imputer = SimpleImputer(strategy='mean')
x_train = imputer.fit_transform(x_train)
x_test = imputer.transform(x_test)

# Now predict using the imputed data
y_pred = decision_tree_model.predict(x_test)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:465: UserWarning: X does
not have valid feature names, but DecisionTreeClassifier was fitted with feature
names
  warnings.warn(
```

[10]:

```python
#Accuracy
from sklearn.metrics import accuracy_score
round(accuracy_score(y_test,y_pred)*100,2)
```

[10]:

```
99.83
```

[11]:

```python
#Classification report
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      1169
         1.0       0.00      0.00      0.00         1

    accuracy                           1.00      1170
   macro avg       0.50      0.50      0.50      1170
weighted avg       1.00      1.00      1.00      1170
```

[12]:

```python
#Confusion matrix
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test,y_pred))
```

```
[[1168    1]
 [   1    0]]
```