ML_LAB_KNN_Online_updated

September 29, 2022

0.1 29 Sep 2022

0.1.1 K NN

0.1.2 Dr Neeraj Gupta

```
[81]: import numpy as np
      import matplotlib.pyplot as plt
      import pandas as pd
```

```
[82]: df = pd.read_csv("Knn_data.csv")
```

```
[83]: df.head(100)
```

```
[83]:    Tid Refund  Marital Status   Taxable Income  Evade
      0    1      Yes          Single           125000     No
      1    2       No         Married           100000     No
      2    3       No          Single            70000     No
      3    4      Yes         Married           120000     No
      4    5       No        Divorced            95000    Yes
      5    6       No         Married            60000     No
      6    7      Yes        Divorced           220000     No
      7    8       No          Single            85000    Yes
      8    9       No         Married            75000     No
      9   10       No          Single            90000    Yes
```

```
[84]: df['Marital Status'].values
```

```
[84]: array(['Single', 'Married', 'Single', 'Married', 'Divorced', 'Married',
              'Divorced', 'Single', 'Married', 'Single'], dtype=object)
```

```
[85]: df['Refund ']
```

```
[85]: 0     Yes
      1      No
      2      No
      3     Yes
      4      No
      5      No
```

```
6    Yes
7     No
8     No
9     No
Name: Refund , dtype: object
```

[85]:

```

```

[86]:

```
from sklearn import preprocessing

# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

# Encode labels in column 'species'.
df['Marital Status']= label_encoder.fit_transform(df['Marital Status'])

df['Marital Status'].unique()
df.head()
```

[86]:

```
   Tid Refund   Marital Status   Taxable Income  Evade
0    1     Yes               2           125000     No
1    2      No               1           100000     No
2    3      No               2            70000     No
3    4     Yes               1           120000     No
4    5      No               0            95000    Yes
```

[87]:

```
from sklearn import preprocessing

# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

# Encode labels in column 'species'.
df['Refund ']= label_encoder.fit_transform(df['Refund '])

df['Refund '].unique()
df.head()
```

[87]:

```
   Tid  Refund   Marital Status   Taxable Income  Evade
0    1        1               2           125000     No
1    2        0               1           100000     No
2    3        0               2            70000     No
3    4        1               1           120000     No
4    5        0               0            95000    Yes
```

[87]:

```

```

[88]:

```
X = df.iloc[:, 1:-1].values
y = df.iloc[:, 4].values
print(X)
```

```
[[     1      2 125000]
 [     0      1 100000]
 [     0      2  70000]
 [     1      1 120000]
 [     0      0  95000]
 [     0      1  60000]
 [     1      0 220000]
 [     0      2  85000]
 [     0      1  75000]
 [     0      2  90000]]
```

[89]:

```
#Train & Test Split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1)
```

[90]:

```
print(X_train)
```

```
[[     1      1 120000]
 [     0      1  60000]
 [     0      2  90000]
 [     0      0  95000]
 [     0      1 100000]
 [     0      1  75000]
 [     1      0 220000]
 [     0      2  85000]
 [     1      2 125000]]
```

[91]:

```
#Feature Scaling
#from sklearn.preprocessing import StandardScaler
#scaler = StandardScaler()
#scaler.fit(X_train)
#X_train = scaler.transform(X_train)
#X_test = scaler.transform(X_test)
```

[92]:

```
#Training and Predictions
from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n_neighbors=3, metric='euclidean')␣
 ↪#metric='minkowski' #euclidean #minkowski

classifier.fit(X, y)
```

[92]:

```
KNeighborsClassifier(metric='euclidean', n_neighbors=3)
```

[93]:

```

```

[93]:

```
array(['No'], dtype=object)
```

[94]:

```
y_test
```

[94]:

```
array(['No'], dtype=object)
```

[95]:

```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[1]]
              precision    recall  f1-score   support

          No       1.00      1.00      1.00         1

    accuracy                           1.00         1
   macro avg       1.00      1.00      1.00         1
weighted avg       1.00      1.00      1.00         1
```

[101]:

```
#Given a new instance X=(Refund=No, Married, Income=120K), Predict whether the Evade is Yes or No.
test = [0 , 1 , 120000]
y_pred = classifier.predict([test])
y_pred
```

[101]:

```
array(['No'], dtype=object)
```