knn

May 18, 2023

[57]:
```
# Package and library import
import numpy as np
from pandas import read_csv
import matplotlib.pyplot as plt
from sklearn.impute import SimpleImputer
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split, GridSearchCV
```

[40]:
```
def load_dataset(fname):
  data = read_csv(fname,na_values = None)
  dataset = data.values
  X = dataset[:,:-1]
  imputer = SimpleImputer(strategy = 'mean')
  imputer.fit(X)
  X = imputer.transform(X)
  y = dataset[:,-1]
  y = imputer.fit_transform(np.array(y).reshape(-1, 1)).flatten()
  return X,y,dataset
```

[41]:
```
# Dataset importing and splitting
X,y,dataset = load_dataset('/content/drive/MyDrive/Data Analytics/water.csv')
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.
 ↪33,random_state = 1)
print('Train',X_train.shape,y_train.shape)
print('Test',X_test.shape,y_test.shape)
```

```
Train (2194, 9) (2194,)
Test (1082, 9) (1082,)
```

[42]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

[43]:
```
# Model training
from sklearn.neighbors import KNeighborsClassifier
```

```
knnclassifier = KNeighborsClassifier(n_neighbors = 5,metric = 'minkowski',␣
  ↪algorithm = 'auto')
knnclassifier.fit(X_train, y_train)
```

[43]:
```
KNeighborsClassifier()
```

[44]:
```
y_pred = knnclassifier.predict(X_test)
```

[45]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
cm = confusion_matrix(y_test,y_pred)
acc = accuracy_score(y_test,y_pred)
```

[46]:
```
print(cm)
print(acc)
```

```
[[499 149]
 [253 181]]
0.6284658040665434
```

[47]:
```
from sklearn.model_selection import GridSearchCV
# hyperparameter tuning
grid_par = {'n_neighbors':[5,6,7,8,9,10],'weights':['uniform','distance'],
          'metric':['euclidean','manhattan','minkowski']}
```

[48]:
```
gs = GridSearchCV(KNeighborsClassifier(),grid_par,verbose = 1,cv = 5,n_jobs =␣
  ↪-1)
```

[49]:
```
grid_res = gs.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 36 candidates, totalling 180 fits
```

[50]:
```
grid_res.best_score_
```

[50]:
```
0.6385548309254115
```

[51]:
```
grid_res.best_params_
```

[51]:
```
{'metric': 'euclidean', 'n_neighbors': 8, 'weights': 'distance'}
```

[52]:
```
#using the tuned parameters
knnclassifier = KNeighborsClassifier(n_neighbors=9,metric = 'euclidean',
                                     algorithm='auto')
knnclassifier.fit(X_train, y_train)
```

[52]:
```
KNeighborsClassifier(metric='euclidean', n_neighbors=9)
```

[53]:
```
y_pred = knnclassifier.predict(X_test)
```

[54]:
```
cm = confusion_matrix(y_test,y_pred)
acc = accuracy_score(y_test,y_pred)
```

[55]:
```
print(cm)
print(acc)
```

```
[[526 122]
 [274 160]]
0.634011090573013
```

[56]:
```
from sklearn.metrics import classification_report
classy_rep = classification_report(y_test,y_pred)
print(classy_rep)
```

```
              precision    recall  f1-score   support

         0.0       0.66      0.81      0.73       648
         1.0       0.57      0.37      0.45       434

    accuracy                           0.63      1082
   macro avg       0.61      0.59      0.59      1082
weighted avg       0.62      0.63      0.61      1082
```