naive-bayes

May 27, 2023

Importing libraries

[36]:
```
import numpy as np
from pandas import read_csv
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
```

Dataset import and preprocessing

[60]:
```
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
```

Splitting dataset

[59]:
```
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,)
```

[58]:
```
print(X)
```

```
[[7.08079450e+00 2.04890455e+02 2.07913190e+04 … 1.03797831e+01
  8.69909705e+01 2.96313538e+00]
 [3.71608008e+00 1.29422921e+02 1.86300579e+04 … 1.51800131e+01
  5.63290763e+01 4.50065627e+00]
 [8.09912419e+00 2.24236259e+02 1.99095417e+04 … 1.68686369e+01
  6.64200925e+01 3.05593375e+00]
```

```
…
 [9.41951032e+00 1.75762646e+02 3.31555782e+04 … 1.10390697e+01
  6.98454003e+01 3.29887550e+00]
 [5.12676292e+00 2.30603758e+02 1.19838694e+04 … 1.11689462e+01
  7.74882131e+01 4.70865847e+00]
 [7.87467136e+00 1.95102299e+02 1.74041771e+04 … 1.61403676e+01
  7.86984463e+01 2.30914906e+00]]
```

Standardizing features and training

[39]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

[40]:
```
from sklearn.naive_bayes import GaussianNB
NBclassif = GaussianNB()
NBclassif.fit(X_train,y_train)
```

[40]:
```
GaussianNB()
```

[41]:
```
GaussianNB(priors = None, var_smoothing = 1e-09)
```

[41]:
```
GaussianNB()
```

Prediction and performance evaluation

[42]:
```
ypred = NBclassif.predict(X_test)
from sklearn.metrics import confusion_matrix,accuracy_score
cm = confusion_matrix(y_test,ypred)
acc = accuracy_score(y_test,ypred)
print(cm)
print("Accuracy: ",acc)
```

```
[[574  74]
 [335  99]]
Accuracy:  0.621996303142329
```

[43]:
```
from sklearn.metrics import classification_report
classy_rep = classification_report(y_test,ypred)
print(classy_rep)
```

```
              precision    recall  f1-score   support

         0.0       0.63      0.89      0.74       648
         1.0       0.57      0.23      0.33       434

    accuracy                           0.62      1082
   macro avg       0.60      0.56      0.53      1082
```

```
weighted avg       0.61      0.62      0.57      1082
```

Hyperparameter tuning

[44]:
```
np.logspace(0,-9,10)
```

[44]:
```
array([1.e+00, 1.e-01, 1.e-02, 1.e-03, 1.e-04, 1.e-05, 1.e-06, 1.e-07,
       1.e-08, 1.e-09])
```

[45]:
```
from sklearn.model_selection import RepeatedStratifiedKFold
cv = RepeatedStratifiedKFold(n_splits = 5, n_repeats = 3, random_state = 1)
```

[ ]:
```
from sklearn.preprocessing import PowerTransformer
from sklearn.model_selection import GridSearchCV
grid_param = {'var_smoothing':np.logspace(0,-9,100)}
grid_NB=GridSearchCV(estimator = NBclassif, param_grid = grid_param, cv = cv,
                     verbose = 1, scoring = 'accuracy')
data_trans=PowerTransformer().fit_transform(X_test)
grid_NB.fit(data_trans,y_test)
```

[46]:
```
grid_NB.best_score_

#Best score achieved using naive bayes is 61.7%. Using the best set of
#hyperparameters the following performance is achieved using grid search.
```

[46]:
```
0.6170620697502417
```

[47]:
```
grid_NB.best_params_
#Below value is the best of the hyperparameter
```

[47]:
```
{'var_smoothing': 0.3511191734215131}
```

[49]:
```
ypred = grid_NB.predict(X_test)
cm = confusion_matrix(y_test, ypred)
acc = accuracy_score(y_test, ypred)
print(cm)
print("Accuracy: ",acc)
```

```
[[596  52]
 [362  72]]
Accuracy:  0.6173752310536045
```

[50]:
```
cr = classification_report(y_test, ypred)
print(cr)
```

```
              precision    recall  f1-score   support

         0.0       0.62      0.92      0.74       648
```

1.0	0.58	0.17	0.26	434
accuracy			0.62	1082
macro avg	0.60	0.54	0.50	1082
weighted avg	0.61	0.62	0.55	1082