da-lab-11

May 29, 2023

[ ]:
```
#importing packages
from pandas import read_csv
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import pandas as pd
```

[10]:
```
#Loading dataset from drive
data = pd.read_csv("/content/drive/MyDrive/Project/water_potability.csv")
data.head(6)
```

[10]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	\
0	NaN	204.890455	20791.318981	7.300212	368.516441	564.308654	
1	3.716080	129.422921	18630.057858	6.635246	NaN	592.885359	
2	8.099124	224.236259	19909.541732	9.275884	NaN	418.606213	
3	8.316766	214.373394	22018.417441	8.059332	356.886136	363.266516	
4	9.092223	181.101509	17978.986339	6.546600	310.135738	398.410813	
5	5.584087	188.313324	28748.687739	7.544869	326.678363	280.467916	

	Organic_carbon	Trihalomethanes	Turbidity	Potability
0	10.379783	86.990970	2.963135	0
1	15.180013	56.329076	4.500656	0
2	16.868637	66.420093	3.055934	0
3	18.436524	100.341674	4.628771	0
4	11.558279	31.997993	4.075075	0
5	8.399735	54.917862	2.559708	0

[11]:
```
## In place of missing values and NaN, mean value of column is imputed
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy = 'mean')
df_new = pd.DataFrame(imputer.fit_transform(data), columns=data.columns)
df_new.shape
```

[11]:
```
(3276, 10)
```

[12]:
```
df_new.head()
```

[12]:
```
          ph    Hardness        Solids  Chloramines     Sulfate  Conductivity  \
0  7.080795  204.890455  20791.318981     7.300212  368.516441    564.308654
1  3.716080  129.422921  18630.057858     6.635246  333.775777    592.885359
2  8.099124  224.236259  19909.541732     9.275884  333.775777    418.606213
3  8.316766  214.373394  22018.417441     8.059332  356.886136    363.266516
4  9.092223  181.101509  17978.986339     6.546600  310.135738    398.410813

   Organic_carbon  Trihalomethanes  Turbidity  Potability
0       10.379783        86.990970   2.963135         0.0
1       15.180013        56.329076   4.500656         0.0
2       16.868637        66.420093   3.055934         0.0
3       18.436524       100.341674   4.628771         0.0
4       11.558279        31.997993   4.075075         0.0
```

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[17]:
```
#Splitting dataset in X and Y
X = df_new.drop(columns=["Potability"])
y = df_new["Potability"]
```

[18]:
```
#Splittiing X and Y into Train and Test data
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,)
```

[19]:
```
#pre-processing
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.transform(X_test)
```

[20]:
```
#model building
from sklearn.naive_bayes import GaussianNB
NBclassif=GaussianNB()
NBclassif.fit(X_train,y_train)
```

[20]:
```
GaussianNB()
```

[21]:
```
#prediction
ypred=NBclassif.predict(X_test)
```

[22]:
```
#performance evaluation
from sklearn.metrics import confusion_matrix,accuracy_score
cm=confusion_matrix(y_test,ypred)
acc=accuracy_score(y_test,ypred)
```

[23]:
```
print(cm,acc)
```

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

accuracy of model for predictiong the wine quality is 62.19%.

[24]:
```
# Classification report to verify performance metrics
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
```

[25]:
```
#hyperparameter tuning
import numpy as np
```

[26]:
```
np.logspace(0,-9,10)
```

[26]:
```
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])
```

[27]:
```
#Importing library for hypertuning
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
```

[28]:
```
grid_param={'var_smoothing':np.logspace(0,-9,100)}
grid_NB=GridSearchCV(estimator=NBclassif, param_grid=grid_param,cv=cv,␣
 ↪verbose=1, scoring='accuracy')
```

[29]:
```
data_trans=PowerTransformer().fit_transform(X_test)
grid_NB.fit(data_trans,y_test)
```

```
Fitting 15 folds for each of 100 candidates, totalling 1500 fits
```

[29]:
```
GridSearchCV(cv=RepeatedStratifiedKFold(n_repeats=3, n_splits=5,
random_state=1),
             estimator=GaussianNB(),
             param_grid={'var_smoothing': array([1.00000000e+00, 8.11130831e-01,
6.57933225e-01, 5.33669923e-01,
       4.32876128e-01, 3.51119173e-01, 2.84803587e-01, 2.31012970e-01,
       1.87381742e-01, 1.51991108e-01, 1.23284674e-01, 1.00000000e-01,
       8.11130831e-02, 6.57933225e-02, 5.3…
       1.23284674e-07, 1.00000000e-07, 8.11130831e-08, 6.57933225e-08,
       5.33669923e-08, 4.32876128e-08, 3.51119173e-08, 2.84803587e-08,
       2.31012970e-08, 1.87381742e-08, 1.51991108e-08, 1.23284674e-08,
       1.00000000e-08, 8.11130831e-09, 6.57933225e-09, 5.33669923e-09,
       4.32876128e-09, 3.51119173e-09, 2.84803587e-09, 2.31012970e-09,
       1.87381742e-09, 1.51991108e-09, 1.23284674e-09, 1.00000000e-09])},
             scoring='accuracy', verbose=1)
```

[30]:
```
grid_NB.best_score_
```

[30]:
```
0.6170620697502417
```

[31]:
```
grid_NB.best_params_
```

[31]:
```
{'var_smoothing': 0.3511191734215131}
```

[32]:
```
ypred=grid_NB.predict(X_test)
```

[33]:
```
cm=confusion_matrix(y_test,ypred)
acc=accuracy_score(y_test,ypred)
print(cm)
print(acc)
```

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

[34]:
```
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
```

the accuracy of predicting is 0.62 with precision this could pretain to overfitting of data