CM7_updated

February 3, 2021

1 K Nearest Neighbors Algorithm

1.1 Required Libraries

[1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn import metrics
from sklearn.preprocessing import RobustScaler
sns.set()
from sklearn.model_selection import GridSearchCV
```

1.2 KNN on Heart Diseases Dataset

[2]:
```
df_heart= pd.read_csv("heart_disease_missing.csv")
df_heart=df_heart.interpolate(method ='linear', limit_direction ='forward')
```

Splitting the data in train validation and test sets

[3]:
```
X = df_heart.iloc[:, 0:13].values
y = df_heart.iloc[:,13].values
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.
 ↪4,random_state=275)
X_vali, X_test, y_vali, y_test =train_test_split(X_test, y_test, test_size=0.
 ↪5,random_state=275)
fe_sc=RobustScaler()
X_train=fe_sc.fit_transform(X_train)
X_vali=fe_sc.fit_transform(X_vali)
X_test=fe_sc.fit_transform(X_test)
```

Training using best value of k Accuracy,f1_score, and auc for best value of K are 0.76,0.79 and 0.78.

[4]:
```
classifier = KNeighborsClassifier()
classifier.fit(X_train,y_train)
```

[4]:
```
KNeighborsClassifier()
```

[5]:
```
y_output= classifier.predict(X_test)
print(accuracy_score(y_test,y_output))
print(f1_score(y_test,y_output))
fpr, tpr, thresholds = metrics.roc_curve(y_test, y_output, pos_label=None)
metrics.auc(fpr, tpr)
```

```
0.7674418604651163
0.7916666666666666
```

[5]:
```
0.7771493212669683
```

Improving the classifier Accuracy, f1_score, and auc for best value of K are 0.88,0.90 and 0.87.

[6]:
```
classifier = KNeighborsClassifier(35,p=1,metric='minkowski',weights='uniform')
classifier.fit(X_train,y_train)
y_output= classifier.predict(X_vali)
a=accuracy_score(y_vali,y_output)
```

[7]:
```
y_output= classifier.predict(X_test)
print(accuracy_score(y_test,y_output))
print(f1_score(y_test,y_output))
fpr, tpr, thresholds = metrics.roc_curve(y_test, y_output, pos_label=None)
metrics.auc(fpr, tpr)
```

```
0.8837209302325582
0.9056603773584906
```

[7]:
```
0.8733031674208145
```

1.3 KNN on Iris Dataset

[8]:
```
df_iris= pd.read_csv("iris_dataset_missing.csv")
df_iris=df_iris.interpolate(method ='linear', limit_direction ='forward')
```

Splitting the data in train validation and test sets

[9]:
```
X = df_iris.iloc[:, 0:4].values
y = df_iris.iloc[:,4].values
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.
 ↪4,random_state=275)
X_vali, X_test, y_vali, y_test =train_test_split(X_test, y_test, test_size=0.
 ↪5,random_state=275)
fe_sc=StandardScaler()
X_train=fe_sc.fit_transform(X_train)
```

```
X_vali=fe_sc.fit_transform(X_vali)
X_test=fe_sc.fit_transform(X_test)
```

Training using best value of k Accuracy and f1_score for best value of K are 0.95,0.95.

[10]:
```
classifier = KNeighborsClassifier(1)
classifier.fit(X_train,y_train)
```

[10]:
```
KNeighborsClassifier(n_neighbors=1)
```

[11]:
```
y_output= classifier.predict(X_test)
y_output
print(accuracy_score(y_test,y_output))
print(f1_score(y_test,y_output,average='macro'))
```

```
0.9523809523809523
0.952136752136752
```

Improving the classifier Accuracy and f1_score for best value of K are 0.95,0.95

[12]:
```
classifier = KNeighborsClassifier(1,p=2,metric='euclidean',weights='uniform')
classifier.fit(X_train,y_train)
```

[12]:
```
KNeighborsClassifier(metric='euclidean', n_neighbors=1)
```

[13]:
```
y_output= classifier.predict(X_test)
y_output
print(accuracy_score(y_test,y_output))
print(f1_score(y_test,y_output,average='macro'))
```

```
0.9523809523809523
0.952136752136752
```

1.4 References

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html
https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.interpolate.html
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html