Open in Colab

Iris Flower Classification using KNN

Objective

- **The aim is to classify iris flowers among three species from measurements of sepals and petals' length and width.The central goal here is to design a model using KNN Classifier that makes useful classifications for new flowers or, in other words, one which exhibits good generalization.**
- **The iris data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant.**

Submission by:-

Name - Akarshit Srivastava

Section - B

Class Roll no - ***05***

University Roll no - ***181500056***

Importing essential Libraries

In [1]:

```
import numpy as np
import pandas as pd
from sklearn import datasets
import seaborn as sns
from math import sqrt
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score,recall_score,confusion_matrix,classification_r
eport,r2_score,mean_squared_error
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import StratifiedKFold
kFold = StratifiedKFold(n_splits=5)
```

Loading Dataset

In [2]:

```
iris = datasets.load_iris()

##Converting to pandas dataframe
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['species'] = pd.Series(iris.target)
```

In [3]:

```
# concise summary of a DataFrame

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   species            150 non-null    int64
dtypes: float64(4), int64(1)
memory usage: 6.0 KB
```

In [4]:

```
# statistical details

df.describe()
```

Out[4]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000
std	0.828066	0.435866	1.765298	0.762238	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000
75%	6.400000	3.300000	5.100000	1.800000	2.000000
max	7.900000	4.400000	6.900000	2.500000	2.000000

In [5]:

In [6]:

In [ ]:

Feature Scaling and Data Splitting

In [7]:

```
# removing target class from dataset

y=df['species']
X= df.drop('species',axis=1)
```

In [8]:

```
# Dataset splitting

X_train, X_test, y_train, y_test = train_test_split(X, y ,test_size=0.3,random_state=10)
```

In [9]:

```
# Using Standard scaler for feature scaling
```

```
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

KNN Classifier

We will be using KNN CLassifier for this classification problem and will be using Euclidean distance to select nearest Neighbors. Euclidean distance is given by:-

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Self-Define function

- **We will develope our own function to create a KNN CLassfier.**

USING Sk-Learn Library

- **We will use KNeighborsClassifier from scikit-learn and will use gridSearch cv to find the best value of k.**

In [12]:

```
knn_clf = KNeighborsClassifier()
param_grid = {'n_neighbors' : [1,2,3,4,5,7,8,9,10,11,12]}
grid_search = GridSearchCV (knn_clf, param_grid, cv=kFold,scoring = 'recall_weighted', r
eturn_train_score=True)
grid_search.fit(X_train, y_train)
```

Out[12]:

```
GridSearchCV(cv=StratifiedKFold(n_splits=5, random_state=None, shuffle=False),
             error_score=nan,
             estimator=KNeighborsClassifier(algorithm='auto', leaf_size=30,
                                            metric='minkowski',
                                            metric_params=None, n_jobs=None,
                                            n_neighbors=5, p=2,
                                            weights='uniform'),
             iid='deprecated', n_jobs=None,
             param_grid={'n_neighbors': [1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 12]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
             scoring='recall_weighted', verbose=0)
```

In [13]:

```
grid_search.best_params_
#grid_search.best_score_
```

Out[13]:

```
{'n_neighbors': 11}
```

In [14]:

```
knnclassifier = KNeighborsClassifier(n_neighbors=11)
knnclassifier.fit(X_train, y_train)
y_pred = knnclassifier.predict(X_test)
```

Evaluation

In [ ]:

In [16]:

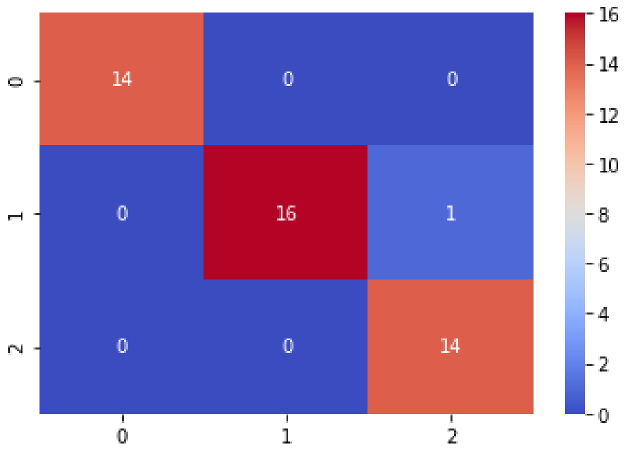
```
# Confusion Matrix

cm=confusion_matrix(y_test, y_pred)
print("Confusion matrix of classifier : \n",cm)
print("\n")
sns.heatmap(cm, annot=True,cmap ='coolwarm')
```

```
Confusion matrix of classifier :
 [[14  0  0]
 [ 0 16  1]
 [ 0  0 14]]
```

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3b69eac150>
```



In [17]:

```
# Classification report of our model.

t=["Iris-setosa","Iris-versicolor","Iris-virginica"]
print(classification_report(y_test, y_pred,target_names=t))
```

```
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        14
Iris-versicolor       1.00      0.94      0.97        17
 Iris-virginica       0.93      1.00      0.97        14

       accuracy                           0.98        45
      macro avg       0.98      0.98      0.98        45
   weighted avg       0.98      0.98      0.98        45
```

In [19]:

```
rmse = sqrt(mean_squared_error(y_test, y_pred))
print("RMSE value = %.2f"%rmse)
print("R2 Score= %.2f"%r2_score(y_test, y_pred))
#print('Train Accuracy score:',knn_train_accuracy)
knn_test_accuracy=accuracy_score(y_test, y_pred)
print('Test Accuracy score: ',knn_test_accuracy)
```

```
RMSE value = 0.15
R2 Score= 0.96
Test Accuracy score:  0.9777777777777777
```

We got test Accuracy of 97.78% on the iris Dataset using sklearn library using KNN Classifier with number of neighbors=11.

Thank you