CS353 ML Lab MidSem

Name: K V Sumanth Reddy

Roll No: 181CO225

Batch: Section-2

Date: 09/03/2021

Q: Build a K-nearest neighbor algorithm to predict whether a patient is having cancer (Malignant tumor) or not (Benign tumor). Use Kaggle dataset from UCI Machine Learning Repository.

Dataset Used: Breast Cancer Wisconsin (Diagnostic) Data Set (https://www.kaggle.com/uciml/breast-cancer-wisconsin-data)

Importing Libraries and Dataset

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import confusion_matrix,explained_variance_sco
from sklearn.metrics import mean_squared_error,r2_score
from sklearn.metrics import accuracy_score, classification_report
```

```
dataset = pd.read_csv('data.csv')

#printing 5 sample tuples
dataset.sample(5)
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean
312	89382602	B	12.76	13.37	82.29	504.1
200	877501	B	12.23	19.56	78.54	461.0

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   id                       569 non-null    int64  
 1   diagnosis                569 non-null    object 
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
 32  Unnamed: 32              0 non-null      float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```

Data Preprocessing

```
# Drop unused columns and encoding M to 0 and B to 1
columns = ['Unnamed: 32', 'id', 'diagnosis']
data = {'M': 0, 'B': 1}

y = dataset['diagnosis'].map(data)
```

```
y = dataset['diagnosis'].map(data)
x = dataset.drop(columns, axis=1)
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size

print('Training dataset size:\nx_train -', len(x_train), '\ny_train
print('Testing dataset size:\nx_test -', len(x_test), '\ny_test -',
```

```
Training dataset size:
x_train - 455
y_train - 455

Testing dataset size:
x_test - 114
y_test - 114
```

```
KNN = []
for i in range(1, 8):
  KNNModel = KNeighborsClassifier(n_neighbors = i, metric = 'euclide
  KNN.append(KNNModel)
```

```
for i in range(7):
  KNN[i].fit(x_train, y_train)
```

Finding Accuracies for all models using the test dataset

```
train_accuracyKNN = []    #store training accuracies
test_accuracyKNN = []     #store testing accuracies

for i in range(7):
  print("---------------------------------")
  print('K = {}'.format(i + 1))
  train_accuracyKNN.append(KNN[i].score(x_train, y_train))
  test_accuracyKNN.append(KNN[i].score(x_test, y_test))
  y_pred = KNN[i].predict(x_test)

  print("Confusion Matrix:\n",confusion_matrix(y_test, y_pred))
  print(classification_report(y_test,y_pred))
  print("\nAccuracy: %.2f" %(accuracy_score(y_test, y_pred)*100))
  print("Mean Squared Error: %.2f" %(mean_squared_error(y_test, y_pr
  print("Explained Variance: %.2f" %(explained_variance_score(y_test
  print("R2 Score: %.2f" %(r2_score(y_test, y_pred)*100))
  print("\n\n")
  print("---------------------------------")
```

```
    accuracy                           0.94       114
   macro avg       0.93      0.93      0.93       114
```

```
   macro avg       0.95      0.95      0.95       114
weighted avg       0.94      0.94      0.94       114


Accuracy: 93.86
Mean Squared Error: 6.14
Explained Variance: 72.40
R2 Score: 72.37



--------------------------------
--------------------------------
K = 6
Confusion Matrix:
 [[35  3]
 [ 4 72]]
              precision    recall  f1-score   support

           0       0.90      0.92      0.91        38
           1       0.96      0.95      0.95        76

    accuracy                           0.94       114
   macro avg       0.93      0.93      0.93       114
weighted avg       0.94      0.94      0.94       114


Accuracy: 93.86
Mean Squared Error: 6.14
Explained Variance: 72.40
R2 Score: 72.37



--------------------------------
--------------------------------
K = 7
Confusion Matrix:
 [[35  3]
 [ 2 74]]
              precision    recall  f1-score   support

           0       0.95      0.92      0.93        38
           1       0.96      0.97      0.97        76

    accuracy                           0.96       114
   macro avg       0.95      0.95      0.95       114
weighted avg       0.96      0.96      0.96       114


Accuracy: 95.61
Mean Squared Error: 4.39
Explained Variance: 80.30
R2 Score: 80.26



--------------------------------
```

Best accuracy is at neighbours = 7 and the best mean difference between train and test accuracies is at neighbours = 3,5,6

```
fig = plt.figure()
ax = plt.axes()
plt.plot([1,2,3,4,5,6,7], train_accuracyKNN, label = 'Training Accu
plt.plot([1,2,3,4,5,6,7], test_accuracyKNN, label = 'Testing Accura
plt.xlabel('K (number of neighbours)')
plt.ylabel('Mean accuracy')
plt.title('Accuracy vs K')
plt.legend()
plt.savefig('Graph.png')
plt.show()
```