Assignment_3

October 9, 2024

1 IMPLEMENTATION OF K-NEAREST NEIGHBORS ALGORITHM.

1.1 Importing Libraries and Dataset

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
df = pd.read_csv('diabetes.csv')
```

[4]:
```
df
```

[4]:
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           ...      ...            ...            ...      ...  ...
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     Pedigree  Age  Outcome
0       0.627   50        1
1       0.351   31        0
2       0.672   32        1
3       0.167   21        0
4       2.288   33        1
..        ...  ...      ...
763     0.171   63        0
764     0.340   27        0
765     0.245   30        0
766     0.349   47        1
```

```
767     0.315   23        0

[768 rows x 9 columns]
```

[5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Pregnancies     768 non-null    int64
 1   Glucose         768 non-null    int64
 2   BloodPressure   768 non-null    int64
 3   SkinThickness   768 non-null    int64
 4   Insulin         768 non-null    int64
 5   BMI             768 non-null    float64
 6   Pedigree        768 non-null    float64
 7   Age             768 non-null    int64
 8   Outcome         768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

[6]:
```
df.isnull().sum()
```

[6]:
```
Pregnancies      0
Glucose          0
BloodPressure    0
SkinThickness    0
Insulin          0
BMI              0
Pedigree         0
Age              0
Outcome          0
dtype: int64
```

[8]:
```
X = df.drop('Outcome', axis=1)
y = df['Outcome']
```

[10]:
```
from sklearn.model_selection import train_test_split
```

[11]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

1.2 Scaling the values

[13]:
```
from sklearn.preprocessing import StandardScaler
```

[14]:
```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

1.3 Training the Model

[15]:
```
from sklearn.neighbors import KNeighborsClassifier
```

[16]:
```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
```

[16]:
```
KNeighborsClassifier()
```

[17]:
```
y_pred = knn.predict(X_test)
```

1.4 Evaluating Model

[26]:
```
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score,␣
↪recall_score, classification_report
```

[20]:
```
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)
```

```
Confusion Matrix:
 [[79 20]
 [27 28]]
```

[22]:
```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.6948051948051948
```

[23]:
```
error_rate = 1 - accuracy
print("Error Rate:", error_rate)
```

```
Error Rate: 0.30519480519480524
```

[24]:
```
precision = precision_score(y_test, y_pred)
print("Precision:", precision)
```

```
Precision: 0.5833333333333334
```

[25]:
```
recall = recall_score(y_test, y_pred)
print("Recall:", recall)
```

```
Recall: 0.509090909090909
```

1.5 Classification Report

[27]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.75      0.80      0.77        99
           1       0.58      0.51      0.54        55

    accuracy                           0.69       154
   macro avg       0.66      0.65      0.66       154
weighted avg       0.69      0.69      0.69       154
```

[ ]:
```

```