# ML_Prac5

October 18, 2023

### 0.0.1 Aim: Implement K-Nearest Neighbors algorithm on diabetes.csv dataset. Compute confusion matrix, accuracy, error rate, precision and recall on the given dataset.

**Name: Chinmay Gokhale**

**Div: BE-A**

**Roll No. B211047**

[1]:
```
import pandas as pd
import numpy as np
from sklearn import metrics
```

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

[2]:
```
     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]
```

[3]:
```
df.shape
```

[3]:
```
(768, 9)
```

[4]:
```
# checking for null values
df.isnull().any().value_counts()
```

[4]:
```
False    9
Name: count, dtype: int64
```

[5]:
```
df.columns
```

[5]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'Pedigree', 'Age', 'Outcome'],
      dtype='object')
```

[6]:
```
df_x = df.drop(columns='Outcome', axis=1)
df_y = df['Outcome']
```

[8]:
```
print(df.isnull().sum())
```

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

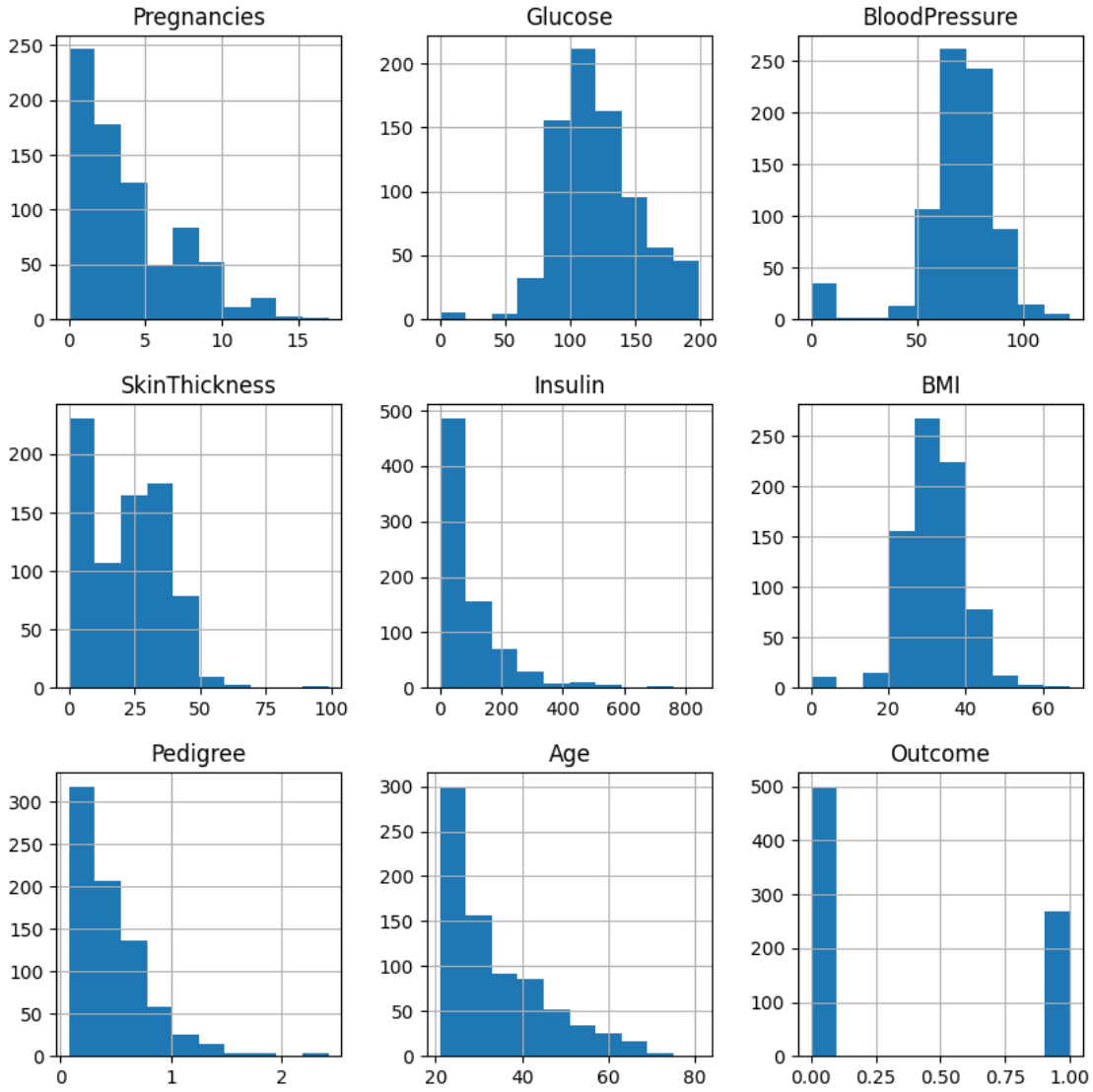
[9]:
```
from sklearn.preprocessing import StandardScaler
scale = StandardScaler()
scaledX = scale.fit_transform(df_x)
```

[10]:
```
# split into train and test
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(scaledX, df_y, test_size=0.
 ↪47, random_state=47)
```

[11]:
```
# KNN
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=7)

knn.fit(x_train, y_train)
y_pred = knn.predict(x_test)
```

[12]:
```
p = df.hist(figsize = (10,10))
```




[13]:
```
# Confusion matrix
cs = metrics.confusion_matrix(y_test,y_pred)
print("Confusion matrix: \n",cs)
```

```
Confusion matrix:
 [[195  40]
 [ 72  54]]
```

[14]:
```
# Accuracy score
ac = metrics.accuracy_score(y_test, y_pred)
print("Accuracy score: ",ac)
```

```
Accuracy score:  0.6897506925207756
```

[15]:
```
# Error rate (error_rate = 1- accuracy)
er = 1-ac
print("Error rate: ",er)
```

```
Error rate:  0.3102493074792244
```

[16]:
```
# Precision
p = metrics.precision_score(y_test,y_pred)
print("Precision: ", p)
```

```
Precision:  0.574468085106383
```

[17]:
```
# Recall
r = metrics.recall_score(y_test,y_pred)
print("Recall: ", r)
```

```
Recall:  0.42857142857142855
```

[18]:
```
# Classification report
cr = metrics.classification_report(y_test,y_pred)
print("Classification report: \n\n", cr)
```

```
Classification report:

              precision    recall  f1-score   support

           0       0.73      0.83      0.78       235
           1       0.57      0.43      0.49       126

    accuracy                           0.69       361
   macro avg       0.65      0.63      0.63       361
weighted avg       0.68      0.69      0.68       361
```

[ ]:
```

```