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

Dataset link : https://www.kaggle.com/datasets/abdallamahgoub/diabetes

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split

from sklearn import metrics
```

```
df=pd.read_csv('/content/drive/MyDrive/ML/diabetes (1).csv')
```

```
df.columns
```

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

Check for null values. If present remove null values from the dataset

```
df.isnull().sum()
```

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

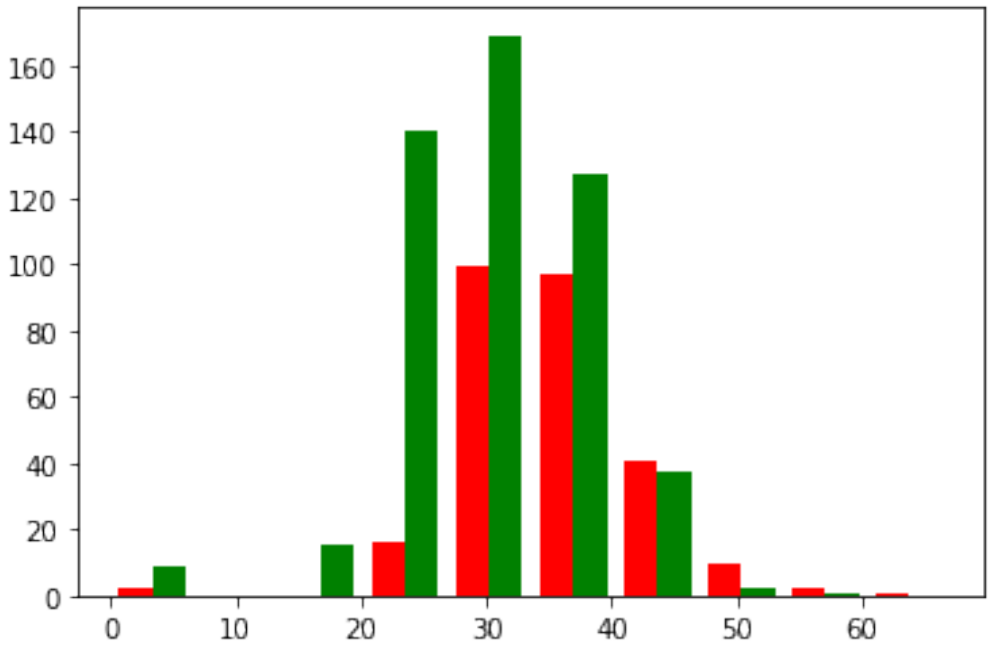
Outcome is the label/target, other columns are features

```
X = df.drop('Outcome',axis = 1)
y1 = df['Outcome']
```

```
x = df[df['Outcome']==1]['BMI']
y= df[df['Outcome']==0]['BMI']
```

```
plt.hist([x, y], color=['red', 'green'], label = ['exit', 'not_exit'])
```

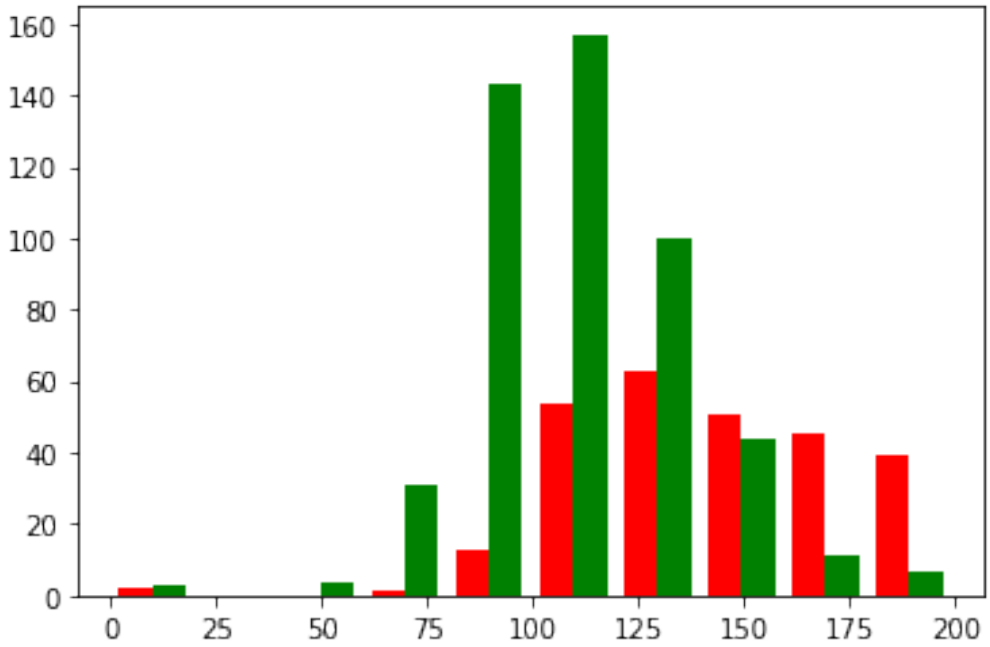
```
(array([[  2.,   0.,   0.,  16.,  99.,  97.,  41.,  10.,   2.,   1.],
        [  9.,   0.,  15., 140., 169., 127.,  37.,   2.,   1.,
0.]]),
 array([ 0.  ,  6.71, 13.42, 20.13, 26.84, 33.55, 40.26, 46.97, 53.68,
        60.39, 67.1 ]),
 <a list of 2 Lists of Patches objects>)
```




```
x = df[df['Outcome']==1]['Glucose']
y= df[df['Outcome']==0]['Glucose']

plt.hist([x, y], color=['red', 'green'], label = ['exit', 'not_exit'])
```

```
(array([[  2.,   0.,   0.,   1.,  13.,  54.,  63.,  51.,  45.,  39.],
        [  3.,   0.,   4.,  31., 143., 157., 100.,  44.,  11.,
7.]]),
 array([  0. ,  19.9,  39.8,  59.7,  79.6,  99.5, 119.4, 139.3, 159.2,
        179.1, 199. ]),
 <a list of 2 Lists of Patches objects>)
```




```
from sklearn.preprocessing import scale
X = scale(X)

# split into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y1, test_size =
0.3, random_state = 42)

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=7)

knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

print("Confusion matrix: ")
cs = metrics.confusion_matrix(y_test,y_pred)
print(cs)

Confusion matrix:
[[123  28]
 [ 37  43]]

print("Acccuracy ",metrics.accuracy_score(y_test,y_pred))

Acccuracy  0.7186147186147186
```

Classification error rate: proportion of instances misclassified over the whole set of instances. Error rate is calculated as the total number of two incorrect predictions (FN + FP) divided by the total number of a dataset (examples in the dataset.

Also error_rate = 1- accuracy

```
total_misclassified = cs[0,1] + cs[1,0]
print(total_misclassified)
total_examples = cs[0,0]+cs[0,1]+cs[1,0]+cs[1,1]
print(total_examples)
print("Error rate",total_misclassified/total_examples)
print("Error rate ",1-metrics.accuracy_score(y_test,y_pred))
```

```
65
231
Error rate 0.2813852813852814
Error rate  0.2813852813852814
```

```
print("Precision score",metrics.precision_score(y_test,y_pred))
```

```
Precision score 0.6056338028169014
```

```
print("Recall score ",metrics.recall_score(y_test,y_pred))
```

```
Recall score  0.5375
```

```
print("Classification report
",metrics.classification_report(y_test,y_pred))
```

```
Classification report                precision    recall  f1-score
support

           0       0.77      0.81      0.79       151
           1       0.61      0.54      0.57        80

    accuracy                           0.72       231
   macro avg       0.69      0.68      0.68       231
weighted avg       0.71      0.72      0.71       231
```