Implement K-Nearest Neighbors algorithm on diabetes.csv dataset.

Compute confusion matrix, accuracy, error rate, precision and recall on the given dataset.

In [2]:

```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, ConfusionMatrixDisplay, classi
import matplotlib.pyplot as plt
```

Dataset Description:

We will try to build a machine learning model to accurately predict whether or not the patients in the dataset have diabetes or not? The datasets consists of several medical predictor variables and one target variable, Outcome. Predictor variables includes the number of pregnancies the patient has had, their BMI, insulin level, age, and so on.

In [3]:

```
df = pd.read_csv('diabetes.csv')
df.head()
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Pedigree	Ag
0	6	148	72	35	0	33.6	0.627	5
1	1	85	66	29	0	26.6	0.351	3
2	8	183	64	0	0	23.3	0.672	3
3	1	89	66	23	94	28.1	0.167	2
4	0	137	40	35	168	43.1	2.288	3

In [4]:

```
df = pd.read_csv('diabetes.csv')
df.head()
```

Out[4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Pedigree	Ag
0	6	148	72	35	0	33.6	0.627	5
1	1	85	66	29	0	26.6	0.351	3
2	8	183	64	0	0	23.3	0.672	3
3	1	89	66	23	94	28.1	0.167	2

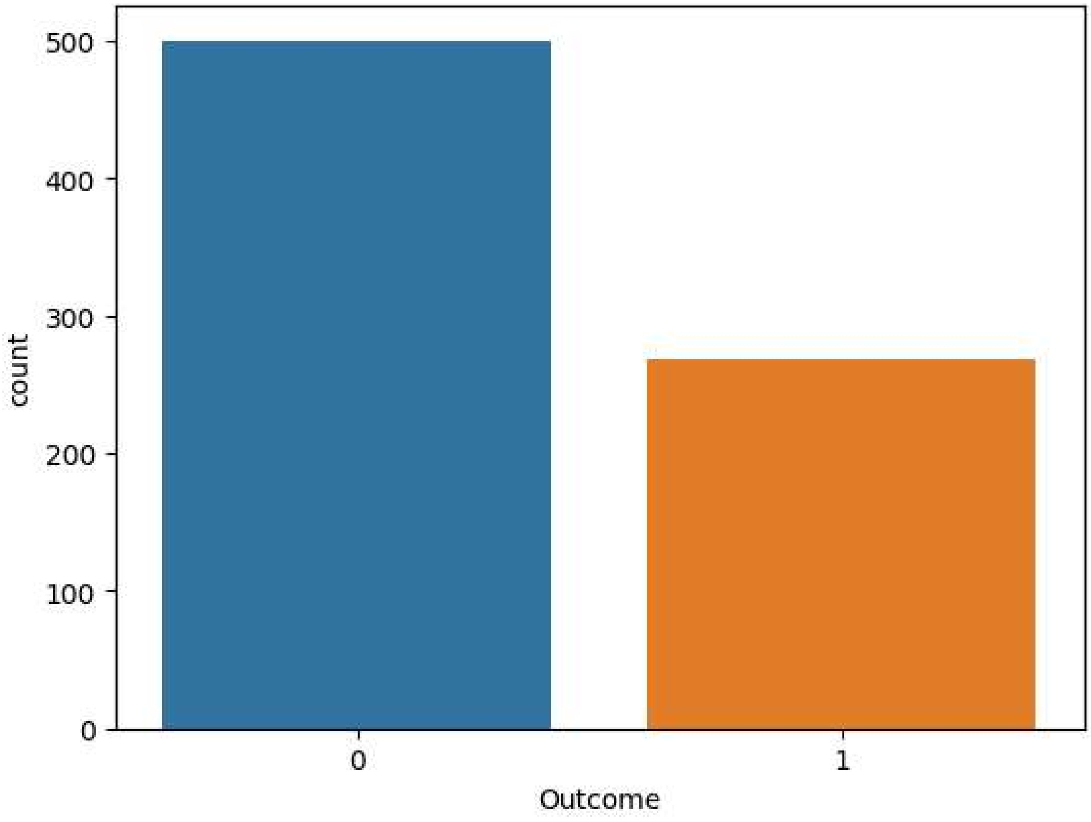
4	0	137	40	35	168	43.1	2.288

In [5]:

```
x = df.drop('Outcome', axis=1)
y = df['Outcome']
sns.countplot(x=y)
```

Out[5]:

```
<Axes: xlabel='Outcome', ylabel='count'>
```



In [6]:

```
y.value_counts()
```

Out[6]:

```
0    500
1    268
Name: Outcome, dtype: int64
```

In [7]:

```
scaler = MinMaxScaler()
x_scaled = scaler.fit_transform(x)
x_train, x_test, y_train, y_test = train_test_split(x_scaled, y, test_size=
x.shape
```

Out[7]:

```
(768, 8)
```

In [8]:

```
print("x_train.shape : ", x_train.shape, "\nx_test.shape : ", x_test.shape)
```

```
x_train.shape :  (537, 8)
x_test.shape :  (231, 8)
```

In [9]:

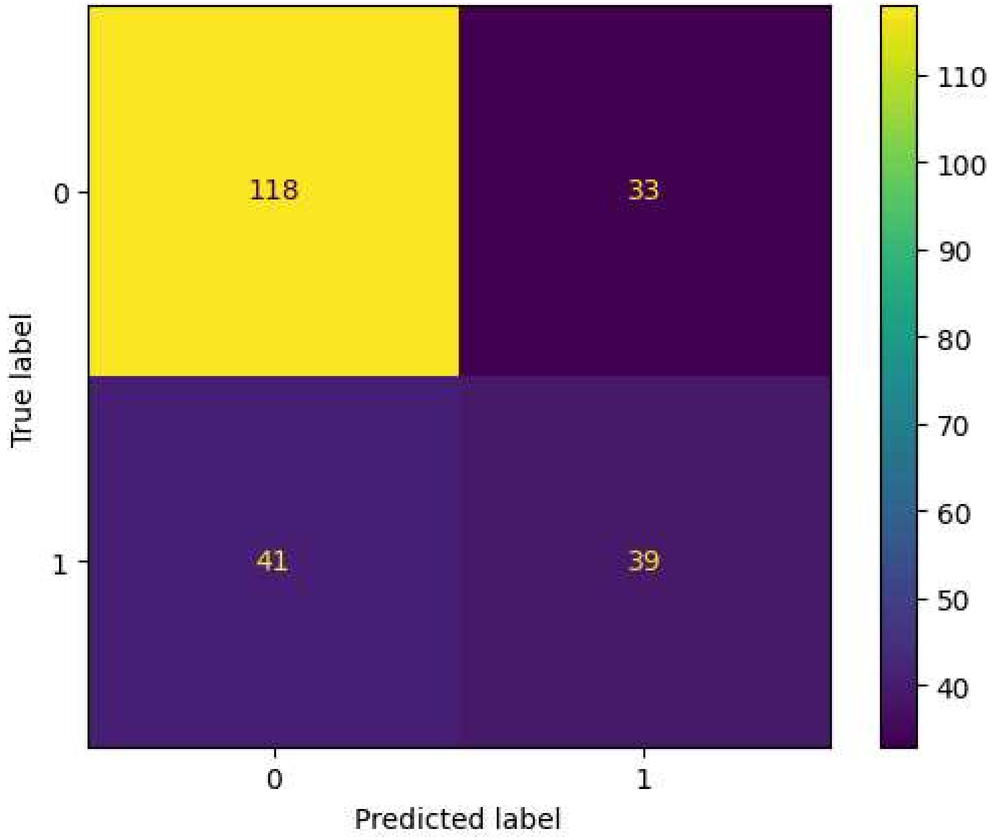
```
knn = KNeighborsClassifier(n_neighbors = 5)
knn.fit(x_train, y_train)
```

Out[9]:
```
KNeighborsClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [10]:
```
y_pred = knn.predict(x_test)
ConfusionMatrixDisplay.from_predictions(y_test,y_pred)
```

Out[10]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7d33140
1bdf0>
```



In [11]:
```
print(classification_report(y_test,y_pred))
```

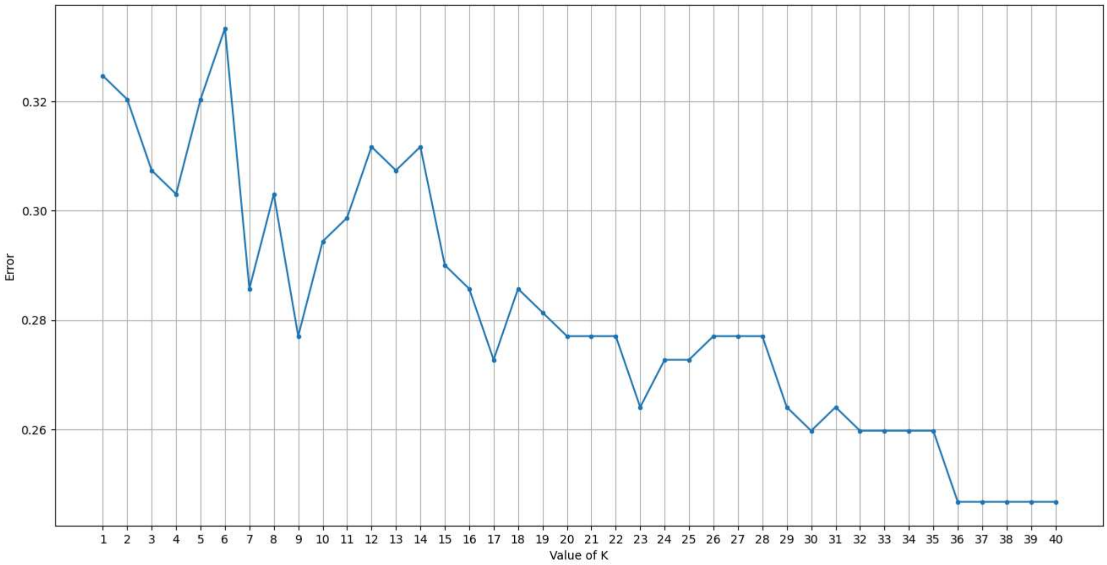
```
              precision    recall  f1-score   support

           0       0.74      0.78      0.76       151
           1       0.54      0.49      0.51        80

    accuracy                           0.68       231
   macro avg       0.64      0.63      0.64       231
weighted avg       0.67      0.68      0.68       231
```

In [12]:
```
error = []
for k in range(1,41):
    knn = KNeighborsClassifier(n_neighbors = k)
    knn.fit(x_train,y_train)
    pred = knn.predict(x_test)
    error.append(np.mean(pred != y_test))
```

In [13]:
```
plt.figure(figsize = (16,8))
plt.xlabel('Value of K')
plt.ylabel('Error')
plt.grid()
plt.xticks(range(1,41))
plt.plot(range(1,41),error,marker = '.')
```

Out[13]:

```
[<matplotlib.lines.Line2D at 0x7d3311523af0>]
```



In [14]:

```
knn=KNeighborsClassifier(n_neighbors = 33)
knn.fit(x_train,y_train)
```

Out[14]:

```
KNeighborsClassifier(n_neighbors=33)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [15]:

```
y_pred = knn.predict(x_test)
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.77      0.87      0.81       151
           1       0.67      0.50      0.57        80

    accuracy                           0.74       231
   macro avg       0.72      0.68      0.69       231
weighted avg       0.73      0.74      0.73       231
```