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

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

df.shape
```

```
(768, 9)
```

```
df
```

```
     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


     DiabetesPedigreeFunction  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]
```

```
df.head()
```

```
   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


   DiabetesPedigreeFunction  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
```

```
# input data
x = df.drop('Outcome', axis = 1)

# Output data
y = df['Outcome']

x.shape
```

```
(768, 8)
```

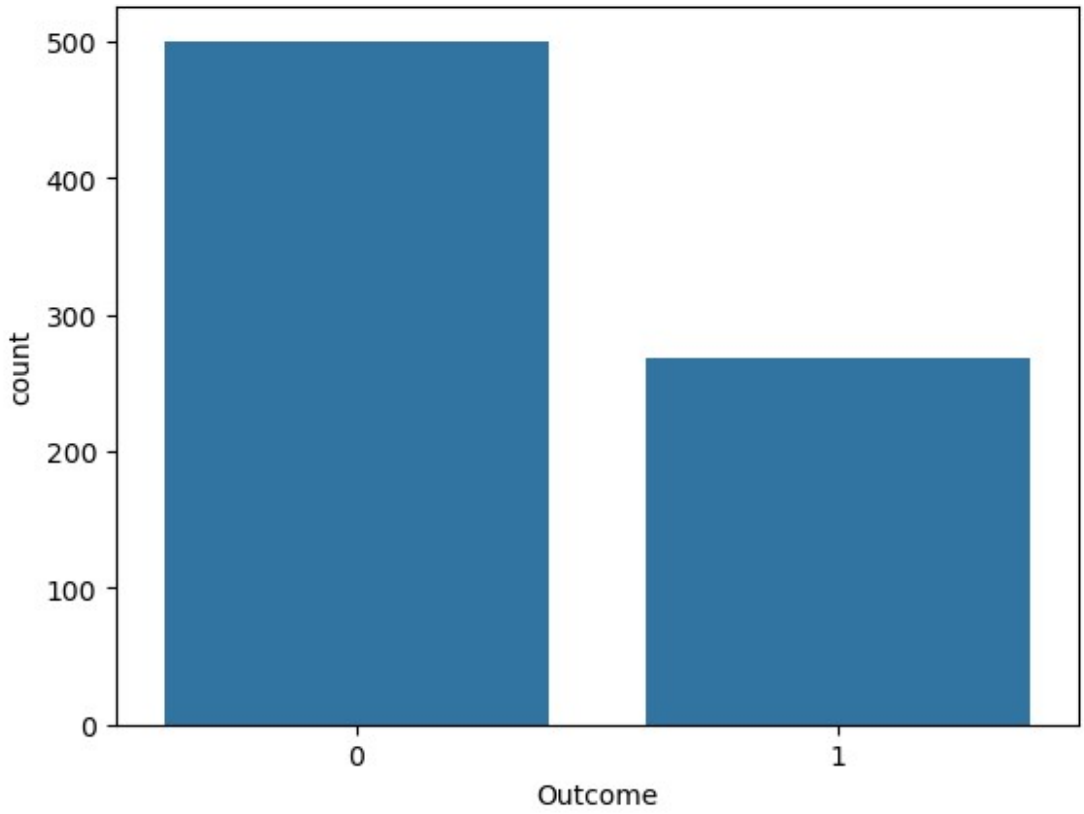
```
x.dtypes
```

```
Pregnancies                   int64
Glucose                       int64
BloodPressure                 int64
SkinThickness                 int64
Insulin                       int64
BMI                         float64
DiabetesPedigreeFunction    float64
Age                           int64
dtype: object
```

```
set(x.dtypes)
```

```
{dtype('int64'), dtype('float64')}
```

```
sns.countplot(x = y);
```




```
y.value_counts()
```

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

```
# Feature Scaling
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
x_scaled = scaler.fit_transform(x)

x_scaled
```

```
array([[0.35294118, 0.74371859, 0.59016393, ..., 0.50074516,
0.23441503,
        0.48333333],
       [0.05882353, 0.42713568, 0.54098361, ..., 0.39642325,
0.11656704,
        0.16666667],
       [0.47058824, 0.91959799, 0.52459016, ..., 0.34724292,
0.25362938,
        0.18333333],
       ...,
       [0.29411765, 0.6080402 , 0.59016393, ..., 0.390462  ,
```

```
0.07130658,
        0.15      ],
       [0.05882353, 0.63316583, 0.49180328, ..., 0.4485842 ,
0.11571307,
        0.43333333],
       [0.05882353, 0.46733668, 0.57377049, ..., 0.45305514,
0.10119556,
        0.03333333]])
```

```
# Cross Validation -75% training and 25% testing
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_scaled, y,
random_state = 0, test_size = 0.25)
```

```
x_scaled.shape
```

```
(768, 8)
```

```
x_train.shape
```

```
(576, 8)
```

```
x_test.shape
```

```
(192, 8)
```

KNN - K Nearest Neighbors

```
# import the class
from sklearn.neighbors import KNeighborsClassifier

# Create the object
knn = KNeighborsClassifier(n_neighbors = 5)

# Train the algorithm
knn.fit(x_train, y_train)
```

```
KNeighborsClassifier()
```

```
# Predict on test data
y_pred = knn.predict(x_test)
```

```
y_pred
```

```
array([1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0,
0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0,
1,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1,
1,
```

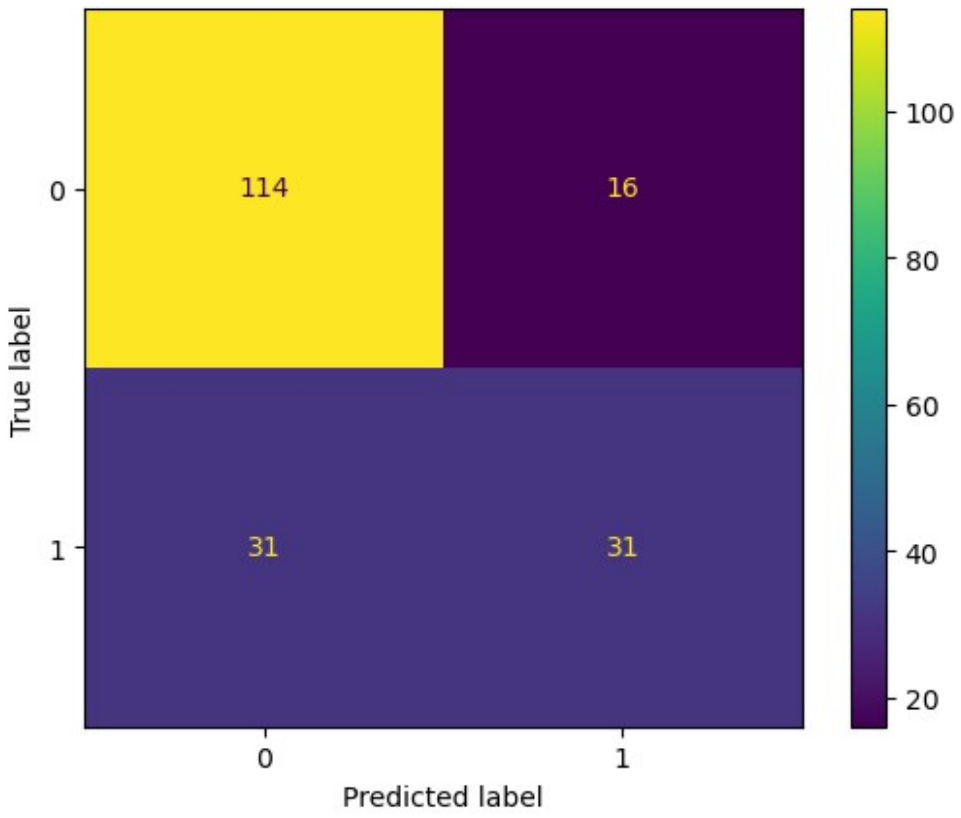
```
       1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1,
       0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0], dtype=int64)
```

```
# Import the evalution metrics
from sklearn.metrics import ConfusionMatrixDisplay, accuracy_score,
classification_report, precision_score, recall_score
```

Confusion Matrics, Accuracy, Error Rate, Precision, Recall

```
# Accuracy = (TP + TN)/ Total

# Error Rate = 1 - Accuracy
# Error Rate = (FP+FN)/Total

# Precision = TP / Predicted Yes

# Recall = TP / Actual Yes

# F1 Score = 2 * (Precision * Recall)/(Precision + recall)

ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x1c55cd15400>
```




```
y_test.value_counts()
```

```
Outcome
0    130
1     62
Name: count, dtype: int64
```

```
accuracy_score(y_test, y_pred)
```

```
0.7552083333333334
```

```
error_rate = 1 - accuracy_score(y_test, y_pred)
print(f"Error Rate: {error_rate}")
```

```
Error Rate: 0.24479166666666663
```

```
precision = precision_score(y_test, y_pred, average='binary')
print(f"Precision: {precision}")
```

```
Precision: 0.6595744680851063
```

```
recall = recall_score(y_test, y_pred, average='binary')
print(f"Recall: {recall}")
```

```
Recall: 0.5
```

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.79      0.88      0.83       130
           1       0.66      0.50      0.57        62

    accuracy                           0.76       192
   macro avg       0.72      0.69      0.70       192
weighted avg       0.75      0.76      0.75       192
```

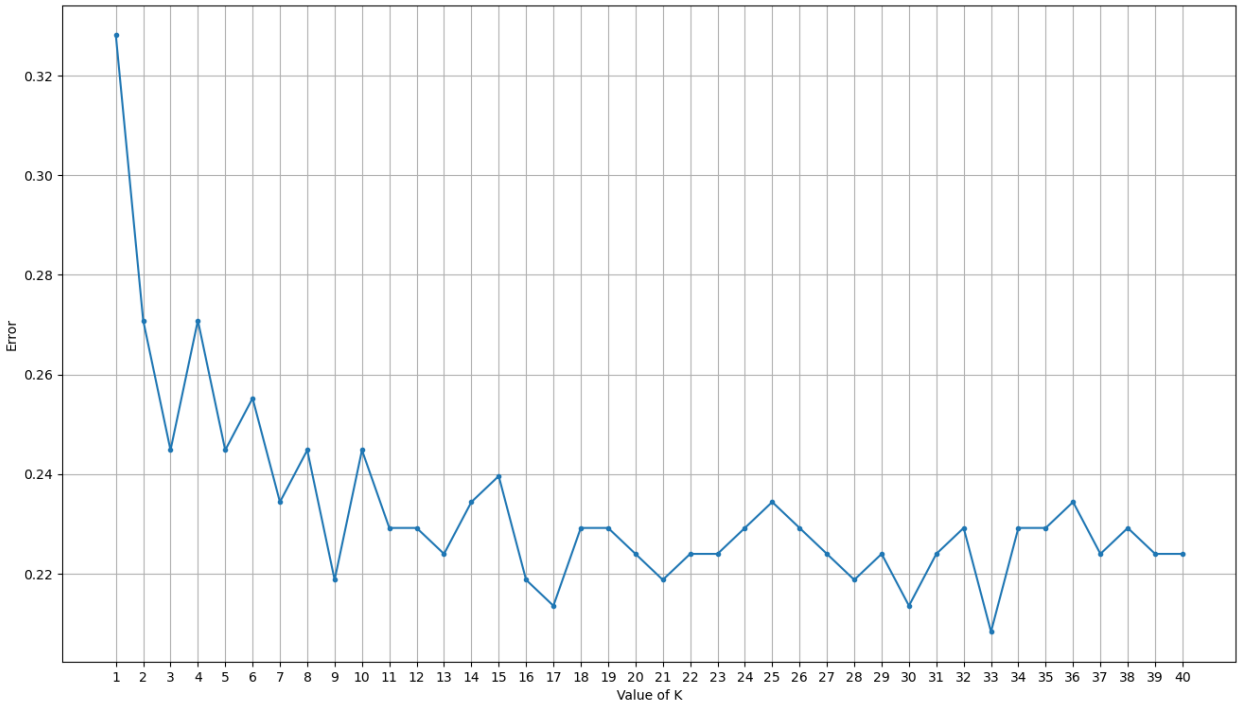
```
error = []
for k in range(1,41):
    knn = KNeighborsClassifier(n_neighbors = k)
    knn.fit(x_train, y_train)
    pred = knn.predict(x_test)
    error.append(np.mean(pred != y_test))

error
```

```
[0.328125,
 0.2708333333333333,
 0.24479166666666666,
 0.2708333333333333,
 0.24479166666666666,
 0.2552083333333333,
 0.234375,
 0.24479166666666666,
 0.21875,
 0.24479166666666666,
 0.22916666666666666,
 0.22916666666666666,
 0.22395833333333334,
 0.234375,
 0.23958333333333334,
 0.21875,
 0.21354166666666666,
 0.22916666666666666,
 0.22916666666666666,
 0.22395833333333334,
 0.21875,
 0.22395833333333334,
 0.22395833333333334,
 0.22916666666666666,
 0.234375,
 0.22916666666666666,
 0.22395833333333334,
 0.21875,
 0.22395833333333334,
```

```
 0.21354166666666666,
 0.22395833333333334,
 0.22916666666666666,
 0.20833333333333334,
 0.22916666666666666,
 0.22916666666666666,
 0.234375,
 0.22395833333333334,
 0.22916666666666666,
 0.22395833333333334,
 0.22395833333333334]
```

```
plt.figure(figsize = (16,9))
plt.xlabel('Value of K')
plt.ylabel('Error')
plt.grid()
plt.xticks(range(1,41))
plt.plot(range(1,41), error, marker = '.')
```

```
[<matplotlib.lines.Line2D at 0x1c5631fc800>]
```




```
knn = KNeighborsClassifier(n_neighbors = 33)

knn.fit(x_train, y_train)
```

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

```
y_pred = knn.predict(x_test)
```

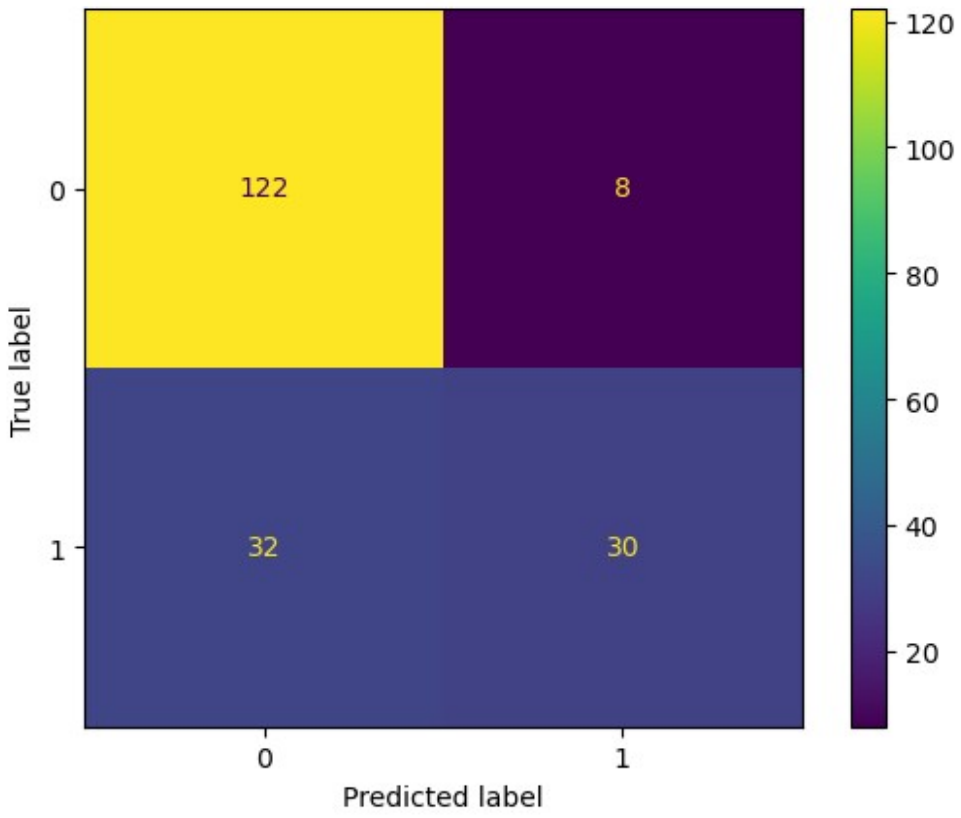
```
y_pred
```

```
array([1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0,
1,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
1,
       1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1,
       0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

Confusion Matrics, Accuracy, Error Rate, Precision, Recall

```
ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x1c5631ff800>
```




```
y_test.value_counts()
```

```
Outcome
0    130
1     62
Name: count, dtype: int64
```

```
accuracy_score(y_test, y_pred)
```

```
0.7916666666666666
```

```
error_rate = 1 - accuracy_score(y_test, y_pred)
print(f"Error Rate: {error_rate}")
```

```
Error Rate: 0.20833333333333337
```

```
precision = precision_score(y_test, y_pred, average='binary')
print(f"Precision: {precision}")
```

```
Precision: 0.7894736842105263
```

```
recall = recall_score(y_test, y_pred, average='binary')
print(f"Recall: {recall}")
```

```
Recall: 0.4838709677419355
```

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.79      0.94      0.86       130
           1       0.79      0.48      0.60        62

    accuracy                           0.79       192
   macro avg       0.79      0.71      0.73       192
weighted avg       0.79      0.79      0.78       192
```