Practical No 4

Title: : 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

```
[5]: pip install mlxtend

Collecting mlxtend
  Downloading mlxtend-0.23.4-py3-none-any.whl.metadata (7.3 kB)
Requirement already satisfied: scipy>=1.2.1 in c:\users\admin\anaconda3\lib\site-packages (from mlxtend) (1.13.1)
Requirement already satisfied: numpy>=1.16.2 in c:\users\admin\anaconda3\lib\site-packages (from mlxtend) (1.26.4)
Requirement already satisfied: pandas>=0.24.2 in c:\users\admin\anaconda3\lib\site-packages (from mlxtend) (2.2.2)
Requirement already satisfied: scikit-learn>=1.3.1 in c:\users\admin\anaconda3\lib\site-packages (from mlxtend) (1.5.1)
Requirement already satisfied: matplotlib>=3.0.0 in c:\users\admin\anaconda3\lib\site-packages (from mlxtend) (3.9.2)
Requirement already satisfied: joblib>=0.13.2 in c:\users\admin\anaconda3\lib\site-packages (from mlxtend) (1.4.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\admin\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\admin\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\admin\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\admin\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\admin\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (24.1)
Requirement already satisfied: pillow>=8 in c:\users\admin\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\admin\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\admin\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\admin\anaconda3\lib\site-packages (from pandas>=0.24.2->mlxtend) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\admin\anaconda3\lib\site-packages (from pandas>=0.24.2->mlxtend) (2023.3)
Requirement already satisfied: six>=1.5 in c:\users\admin\anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib>=3.0.0->mlxtend) (1.16.0)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\admin\anaconda3\lib\site-packages (from scikit-learn>=1.3.1->mlxtend) (3.5.0)
Downloading mlxtend-0.23.4-py3-none-any.whl (1.4 MB)
   ---------------------------------------- 0.0/1.4 MB ? eta -:--:--
   ---------- ----------------------------- 0.5/1.4 MB 4.2 MB/s eta 0:00:01
   ---------------------------------------- 1.4/1.4 MB 3.5 MB/s  0:00:00
```

[6]:
```
import pandas as pd
import numpy as np
import plotly.express as px
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.utils import resample
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from mlxtend.plotting import plot_confusion_matrix
from tqdm.notebook import tqdm
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

[7]:
```
df = pd.read_csv("F:/11 ANJALI VILAD COLLEGE/11 Prof Anjali Phaltane/MACHINE LEARNING/ML LAB LP-III/LP-III ML CODE/PRACTICAL  NO 5/diabetes.csv")
```

[8]:
```
df
```

[8]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Pedigree	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

[9]:
```
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
```

[10]:
```
df.describe().T
```

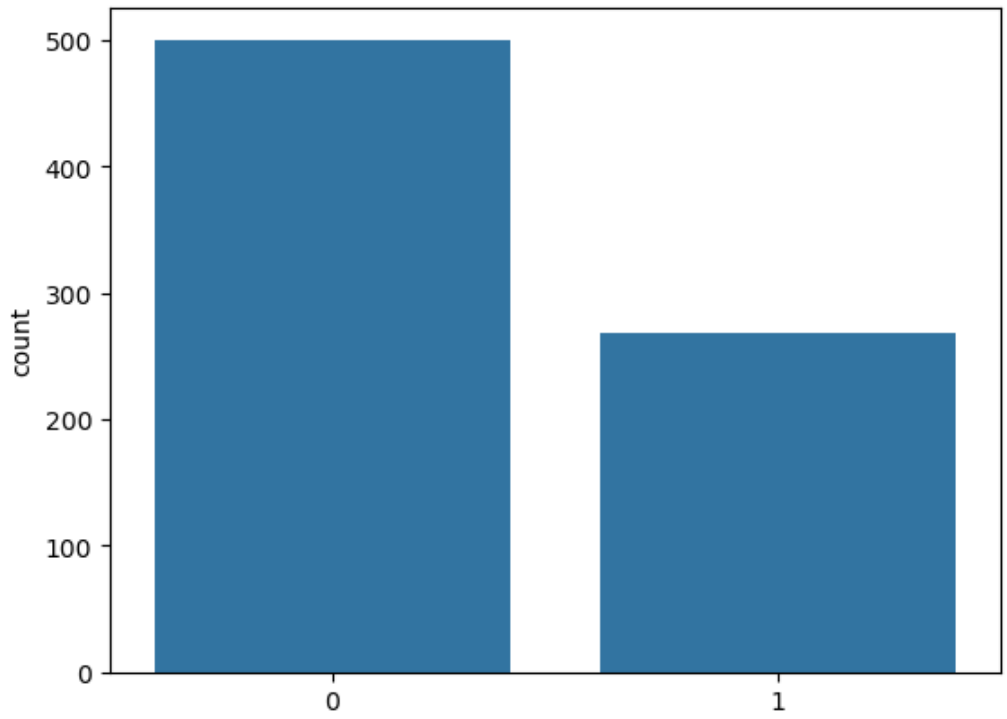
[10]:

	count	mean	std	min	25%	50%	75%	max
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.0000	6.00000	17.00
Glucose	768.0	120.894531	31.972618	0.000	99.00000	117.0000	140.25000	199.00
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000	72.0000	80.00000	122.00
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000	23.0000	32.00000	99.00
Insulin	768.0	79.799479	115.244002	0.000	0.00000	30.5000	127.25000	846.00
BMI	768.0	31.992578	7.884160	0.000	27.30000	32.0000	36.60000	67.10
Pedigree	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.62625	2.42
Age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.00000	81.00
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.0000	1.00000	1.00

[11]:
```
df["Outcome"].value_counts()
```

[11]:
```
Outcome
0    500
1    268
Name: count, dtype: int64
```

[12]:
```
sns.countplot(data=df, x=df["Outcome"])
plt.show()
```



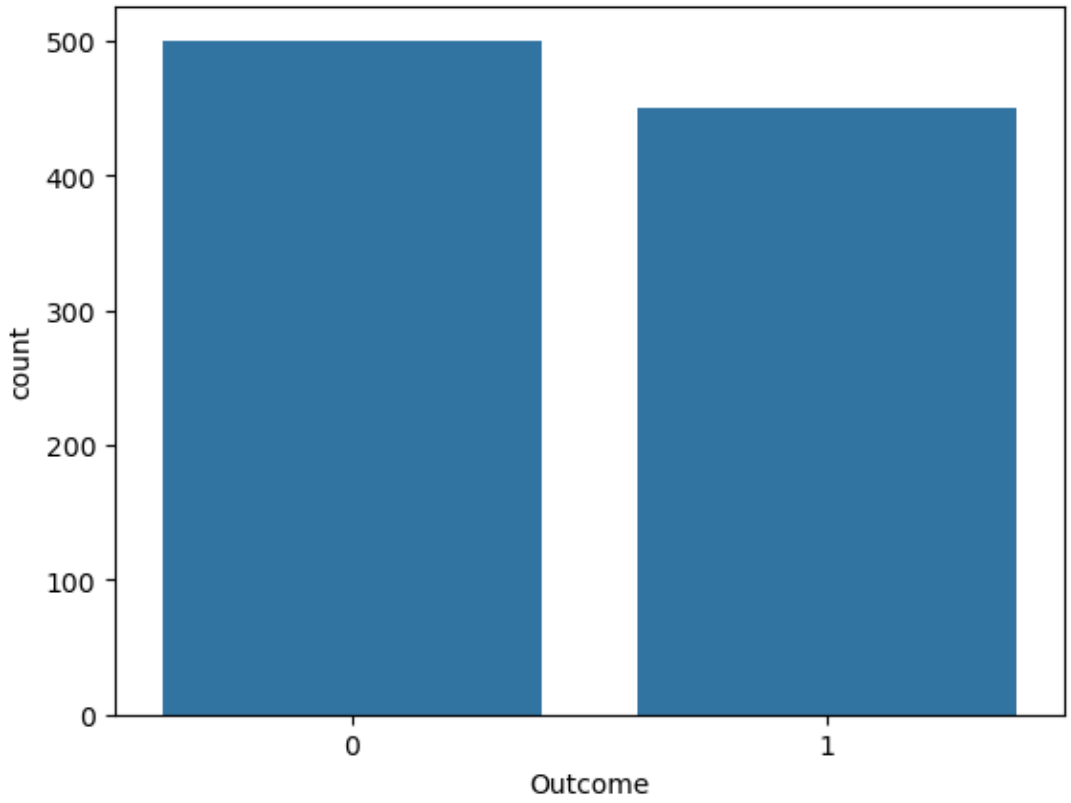
[17]:
```
import pandas as pd
from sklearn.utils import resample

# Separate majority and minority classes
negative_data = df[df["Outcome"] == 0]
positive_data = df[df["Outcome"] == 1]

# Upsample the positive (minority) class
positive_upsample = resample(positive_data,
                             replace=True,  # sample with replacement
                             n_samples=int(0.9 * len(negative_data)),  # target 90% of negatives
                             random_state=42)

# Combine both to form a new balanced dataset
new_df = pd.concat([negative_data, positive_upsample])

# Optional: Shuffle the combined data
new_df = new_df.sample(frac=1, random_state=42).reset_index(drop=True)
```

[18]:
```
new_df.shape
```

[18]:
```
(950, 9)
```

[19]:
```
new_df = new_df.sample(frac=1)
```

[20]:
```
sns.countplot(data=new_df, x=new_df["Outcome"])
plt.show()
```



[21]:
```
x = new_df.drop("Outcome", axis=1)
y = new_df[["Outcome"]]
```

[22]:
```
scaler = MinMaxScaler()
scaled_values = scaler.fit_transform(x)
```

[23]:
```
x_train, x_test, y_train, y_test = train_test_split(scaled_values, y, test_size=0.2)
```

[24]:
```
k_values = [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49]
accuracy_values = []
```

[26]:
```
for i in tqdm(range(len(k_values))):
    model = KNeighborsClassifier(n_neighbors=k_values[i])
    model.fit(x_train, y_train)
    y_pred = model.predict(x_test)
    accuracy = metrics.accuracy_score(y_test, y_pred)
    accuracy_values.append(accuracy)
```

Error displaying widget

Error displaying widget

[31]:
```
optimal_k = -1
optimal_accuracy = -1
for i in list(zip(k_values, accuracy_values)):
    if i[1] > optimal_accuracy:
        optimal_k = i[0]
        optimal_accuracy = i[1]
```

[32]:
```
knn_model = KNeighborsClassifier(n_neighbors=optimal_k)
```

[34]:
```
knn_model.fit(x_train, y_train)
```

[34]:
```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=1)
```

[35]:
```
y_pred = knn_model.predict(x_test)
```

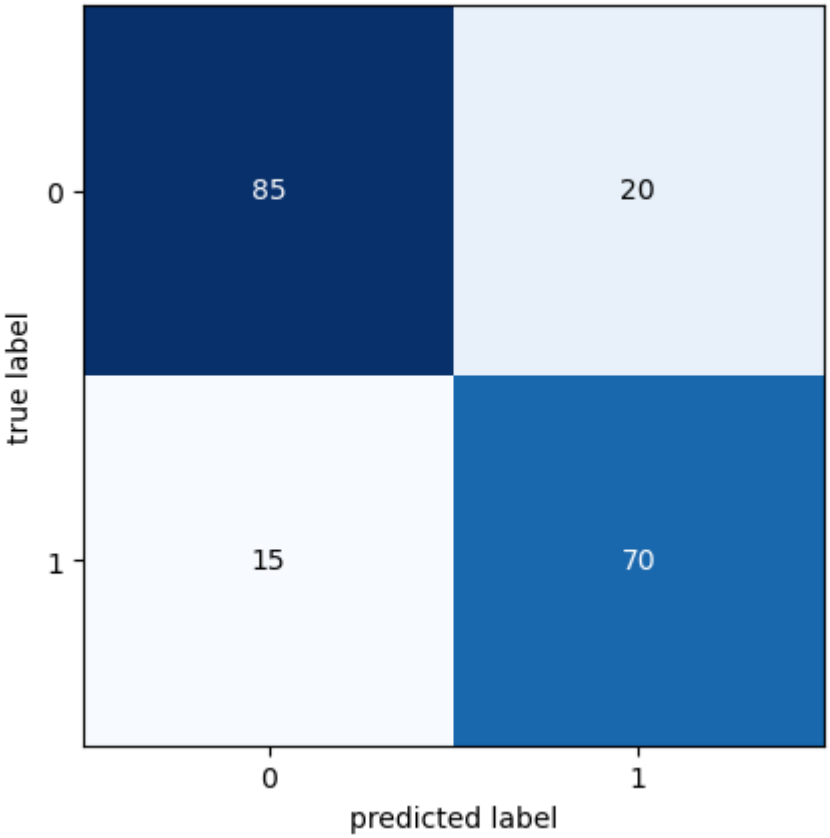
[36]:
```
print(metrics.classification_report(y_test, y_pred))
```

[35]:
```
y_pred = knn_model.predict(x_test)
```

[36]:
```
print(metrics.classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.85      0.81      0.83       105
           1       0.78      0.82      0.80        85

    accuracy                           0.82       190
   macro avg       0.81      0.82      0.81       190
weighted avg       0.82      0.82      0.82       190
```

[37]:
```
cm = metrics.confusion_matrix(y_test, y_pred)
plot_confusion_matrix(cm)
plt.show()
```



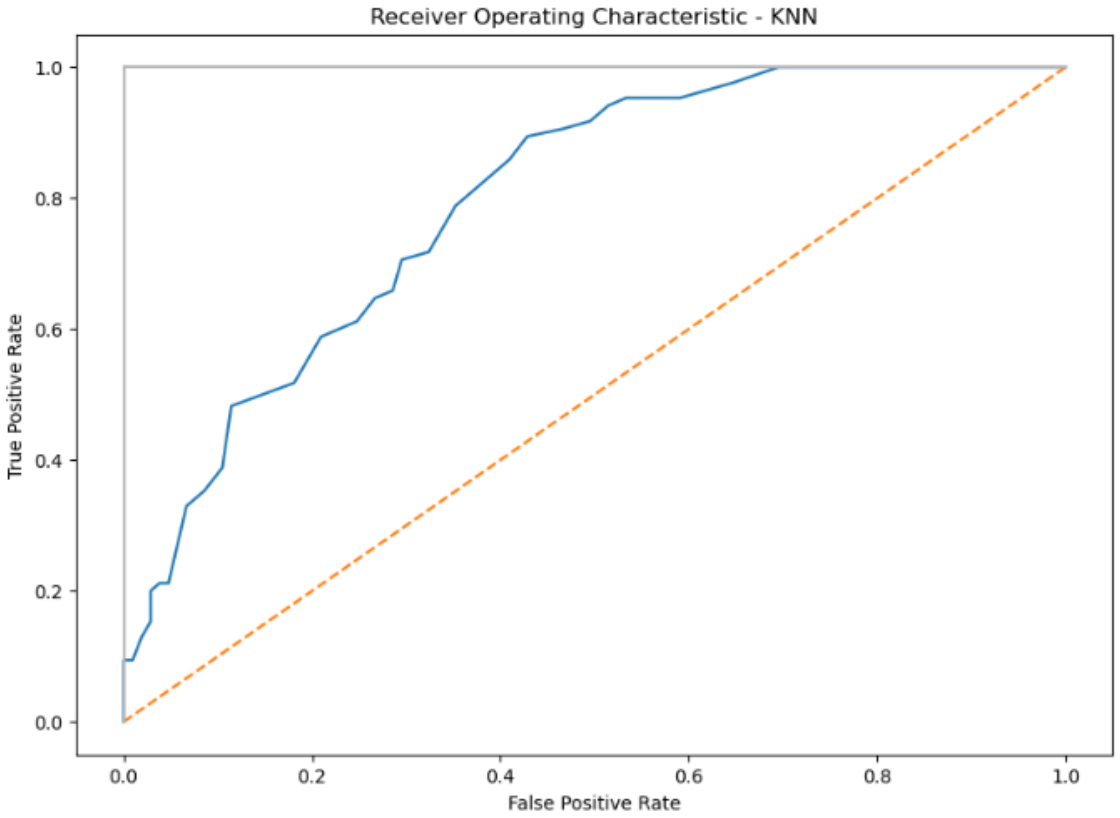
[38]:
```
y_score = model.predict_proba(x_test)[:,1]
```

[39]:
```
false_positive_rate, true_positive_rate, threshold = metrics.roc_curve(y_test, y_score)
```

[40]:
```
print('roc_auc_score for DecisionTree: ', metrics.roc_auc_score(y_test, y_score))
```

```
roc_auc_score for DecisionTree:  0.7961344537815126
```

[41]:
```
plt.subplots(1, figsize=(10,7))
plt.title('Receiver Operating Characteristic - KNN')
plt.plot(false_positive_rate, true_positive_rate)
plt.plot([0, 1], ls="--")
plt.plot([0, 0], [1, 0] , c=".7"), plt.plot([1, 1] , c=".7")
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```


Receiver Operating Characteristic - KNN
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0