In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
```

In [3]:
```
df = pd.read_csv("datasets/diabetes.csv")
df.head()
```

Out[3]:

	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

In [4]:
```
df.shape
```

Out[4]:
```
(768, 9)
```

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

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

In [6]:
```
df.isna().sum()
```

Out[6]:
```
Pregnancies      0
Glucose          0
BloodPressure    0
SkinThickness    0
Insulin          0
BMI              0
Pedigree         0
Age              0
Outcome          0
dtype: int64
```

In [7]:
```
X = df.drop(["Outcome"], axis=1)
y = df["Outcome"]
```

In [8]:
```
X.shape
```

Out[8]:
```
(768, 8)
```

In [9]:
```
y.shape
```

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

In [10]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_stat
```

In [31]:
```
knn = KNeighborsClassifier(n_neighbors=3)
```

In [32]:
```
knn.fit(X_train, y_train)
```

Out[32]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform')
```

In [33]:
```
y_pred = knn.predict(X_test)
```

In [34]:
```
# accuracy score
metrics.accuracy_score(y_test, y_pred)
```

Out[34]:
```
0.7012987012987013
```

Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

In [35]:
```
from sklearn.metrics import confusion_matrix

#extracting true_positives, false_positives, true_negatives, false_negatives
print(confusion_matrix(y_test, y_pred))
tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
print("True Negatives: ",tn)
print("False Positives: ",fp)
print("False Negatives: ",fn)
print("True Positives: ",tp)
```

```
[[83 21]
 [25 25]]
True Negatives:  83
False Positives:  21
False Negatives:  25
True Positives:  25
```

In [36]:
```
#Accuracy
```

```
Accuracy = (tn+tp)*100/(tp+tn+fp+fn)
print("Accuracy {:0.2f}%:".format(Accuracy))
```

```
Accuracy 70.13%:
```

In [37]:

```
#Precision
Precision = tp/(tp+fp)
print("Precision {:0.2f}".format(Precision))
```

```
Precision 0.54
```

In [38]:

```
#Recall
Recall = tp/(tp+fn)
print("Recall {:0.2f}".format(Recall))
```

```
Recall 0.50
```

In [39]:

```
#Error rate
err = (fp + fn)/(tp + tn + fn + fp)
print("Error rate {:0.2f}".format(err))
```

```
Error rate 0.30
```