```
import pandas as pd
import numpy as np
from sklearn import metrics
```

```
df = pd.read_csv("diabetes.csv")
```

```
df
```

	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

```
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
```

```
df.shape
```

```
(768, 9)
```

```
df.head()
```

	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

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

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

```
df.columns
```

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

```
# Separate features (X) and the target variable (y)

X = df.drop(columns=['Outcome'])
Y = df["Outcome"]
```

```
X
```

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

768 rows × 8 columns

```
Y
```

```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

Data Scaling

```
# Standardize the features (scaling is important for KNN)

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaledX = scaler.fit_transform(X)
```

Training a Model

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(scaledX, Y, test_size=0.2, random_state=42)
```

Implement K-Nearest Neighbors algorithm

```
from sklearn.neighbors import KNeighborsClassifier
k = 3   #You can choose the number of neighbors
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(X_train,Y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=3)
```

```
#Make prediction
y_pred = knn.predict(X_test)
```

Confusion MAtric

```
conf_matrix = metrics.confusion_matrix(Y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)
```

```
Confusion Matrix:
[[80 19]
 [27 28]]
```

Accuracy

```
accuracy = metrics.accuracy_score(Y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.7012987012987013
```

Error Rate

```
error_rate = 1 - accuracy
print("Error Rate:", error_rate)
```

```
Error Rate: 0.2987012987012987
```

Precision Rate

```
precision = metrics.precision_score(Y_test, y_pred)
print("Precision:", precision)
```

```
Precision: 0.5957446808510638
```

Recall Rate

```
recall = metrics.recall_score(Y_test, y_pred)
print("Recall:", recall)
```

```
Recall: 0.509090909090909
```

```
f1 = metrics.f1_score(Y_test, y_pred)
print("F1 Score:", f1)
```

```
F1 Score: 0.5490196078431372
```

Classification Report

```
cr = metrics.classification_report(Y_test,y_pred)
print("Classification report: \n\n", cr)
```

```
Classification report:

                  precision    recall  f1-score   support

               0       0.75      0.81      0.78        99
               1       0.60      0.51      0.55        55

        accuracy                           0.70       154
       macro avg       0.67      0.66      0.66       154
    weighted avg       0.69      0.70      0.70       154
```