# Import libraries

In [149…

```
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")
```

# Data loading and preprocessing

In [122…

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

In [123…

```
df
```

Out[123…

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | Pedig |
|---|---|---|---|---|---|---|---|
| **0** | 6 | 148 | 72 | 35 | 0 | 33.6 | 0 |
| **1** | 1 | 85 | 66 | 29 | 0 | 26.6 | 0 |
| **2** | 8 | 183 | 64 | 0 | 0 | 23.3 | 0 |
| **3** | 1 | 89 | 66 | 23 | 94 | 28.1 | 0 |
| **4** | 0 | 137 | 40 | 35 | 168 | 43.1 | 2 |
| **...** | ... | ... | ... | ... | ... | ... | |
| **763** | 10 | 101 | 76 | 48 | 180 | 32.9 | 0 |
| **764** | 2 | 122 | 70 | 27 | 0 | 36.8 | 0 |
| **765** | 5 | 121 | 72 | 23 | 112 | 26.2 | 0 |
| **766** | 1 | 126 | 60 | 0 | 0 | 30.1 | 0 |
| **767** | 1 | 93 | 70 | 31 | 0 | 30.4 | 0 |

768 rows × 9 columns

In [124…

```
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
```

In [125…

```
df.describe().T
```

Out[125…

| | count | mean | std | min | 25% | 50% | |
|---|---|---|---|---|---|---|---|
| **Pregnancies** | 768.0 | 3.845052 | 3.369578 | 0.000 | 1.00000 | 3.0000 | 6 |
| **Glucose** | 768.0 | 120.894531 | 31.972618 | 0.000 | 99.00000 | 117.0000 | 140 |
| **BloodPressure** | 768.0 | 69.105469 | 19.355807 | 0.000 | 62.00000 | 72.0000 | 80 |
| **SkinThickness** | 768.0 | 20.536458 | 15.952218 | 0.000 | 0.00000 | 23.0000 | 32 |
| **Insulin** | 768.0 | 79.799479 | 115.244002 | 0.000 | 0.00000 | 30.5000 | 127 |
| **BMI** | 768.0 | 31.992578 | 7.884160 | 0.000 | 27.30000 | 32.0000 | 36 |
| **Pedigree** | 768.0 | 0.471876 | 0.331329 | 0.078 | 0.24375 | 0.3725 | 0 |
| **Age** | 768.0 | 33.240885 | 11.760232 | 21.000 | 24.00000 | 29.0000 | 41 |
| **Outcome** | 768.0 | 0.348958 | 0.476951 | 0.000 | 0.00000 | 0.0000 | 1 |

In [126…

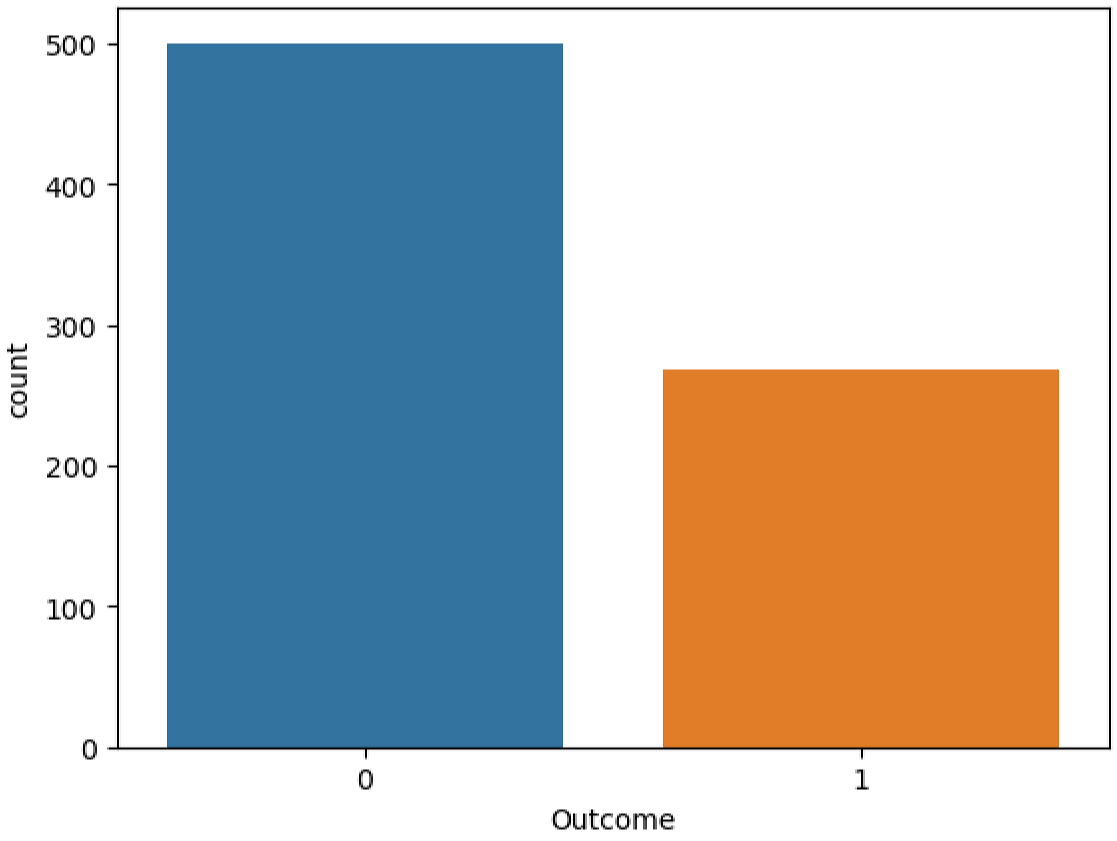
```
df["Outcome"].value_counts()
```

Out[126…

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

In [127…

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



# Upsampling

In [128…

```
negative_data = df[df["Outcome"] == 0]
positive_data = df[df["Outcome"] == 1]
```

In [129…

```
positive_upsample = resample(positive_data,
                             replace=True,
                             n_samples=int(0.9*len(negative_data)),
                             random_state=42)
```

In [130…

```
new_df = negative_data
new_df = new_df.append(positive_upsample)
```

In [131…

```
new_df.shape
```

Out[131…

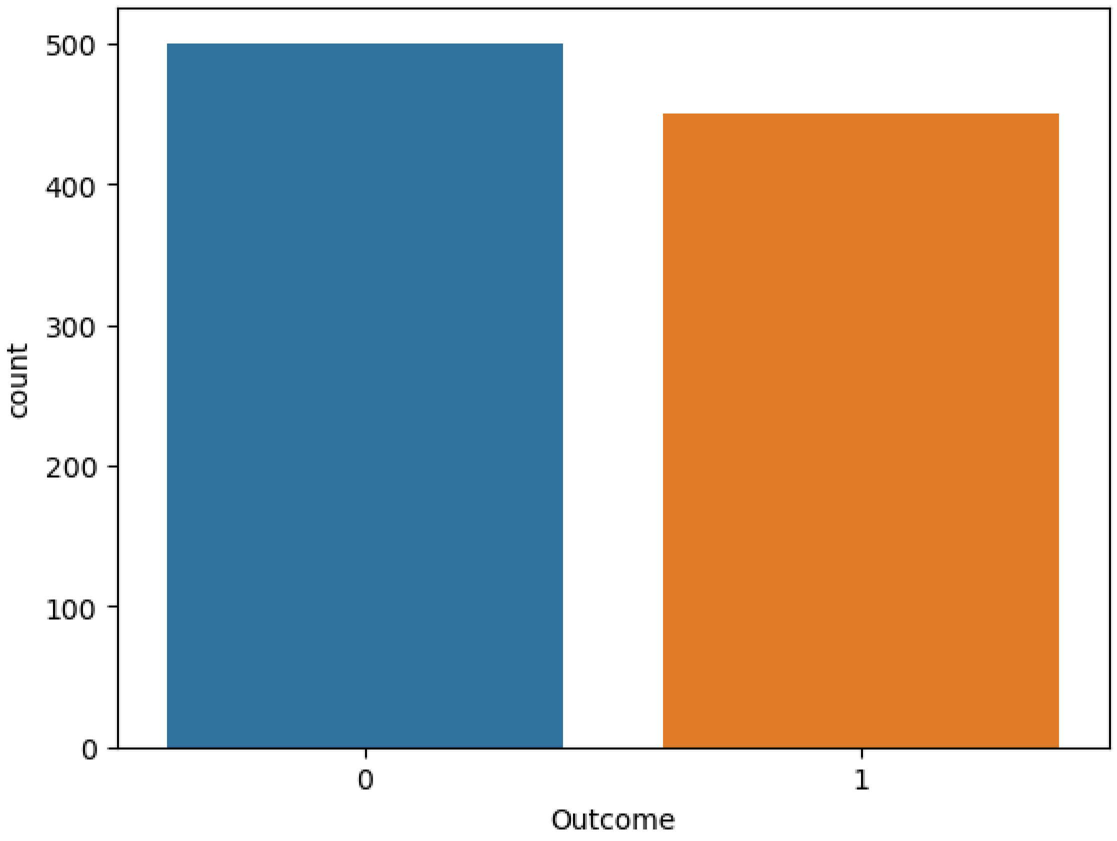
```
(950, 9)
```

In [132…

```
new_df = new_df.sample(frac=1)
```

In [133…

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



```
In [134…  x = new_df.drop("Outcome", axis=1)
          y = new_df[["Outcome"]]
```

```
In [135…  scaler = MinMaxScaler()
          scaled_values = scaler.fit_transform(x)
```

```
In [136…  x_train, x_test, y_train, y_test = train_test_split(scaled_values, y, test_s
```

# KNN with elbow plot

```
In [137…  k_values = [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 3
          accuracy_values = []
```

```
In [138…  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)
```

```
 0%|          | 0/25 [00:00<?, ?it/s]
```

```
In [143…  px.line(x=k_values, y=accuracy_values)
```

In [144…

```
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]
```

In [145…

```
knn_model = KNeighborsClassifier(n_neighbors=optimal_k)
```

In [146…

```
knn_model.fit(x_train, y_train)
```

Out[146…

```
▾       KNeighborsClassifier
KNeighborsClassifier(n_neighbors=1)
```

In [147…

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

In [148…

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

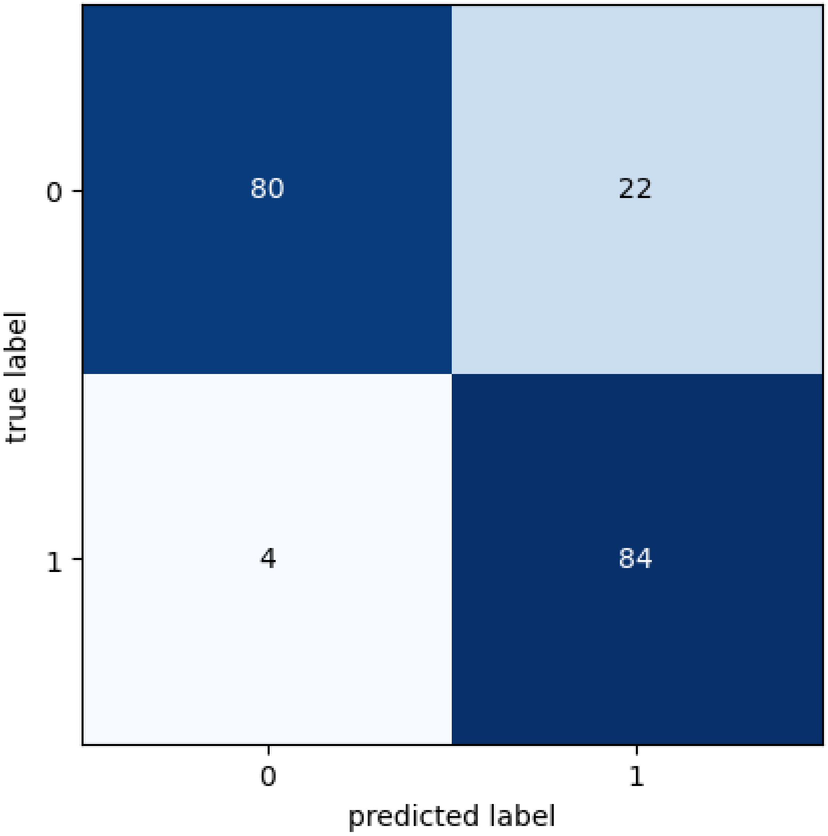
```
              precision    recall  f1-score   support

           0       0.95      0.78      0.86       102
           1       0.79      0.95      0.87        88

    accuracy                           0.86       190
   macro avg       0.87      0.87      0.86       190
weighted avg       0.88      0.86      0.86       190
```

In [152…

```python
cm = metrics.confusion_matrix(y_test, y_pred)
plot_confusion_matrix(cm)
plt.show()
```



In [161…

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

In [162…

```python
false_positive_rate, true_positive_rate, threshold = metrics.roc_curve(y_tes
```

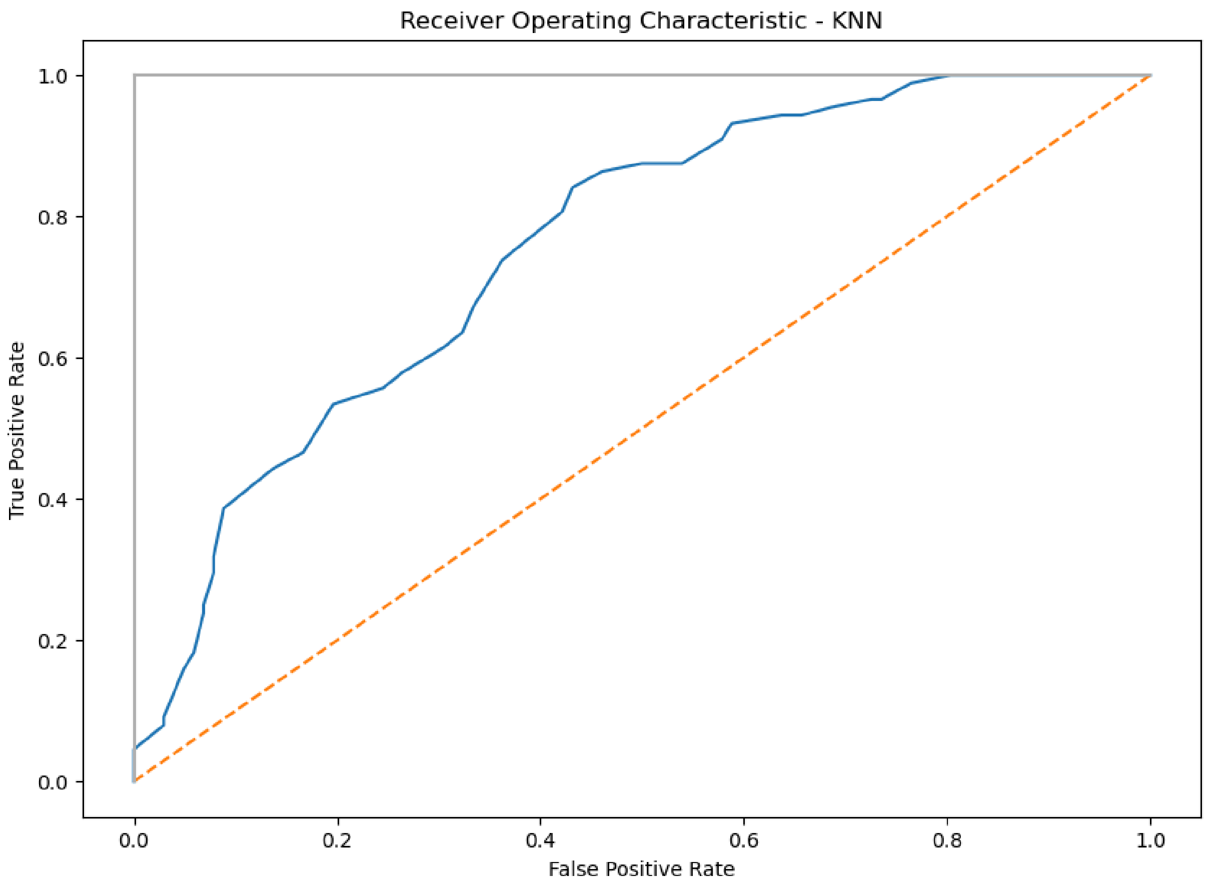
In [163…

```python
print('roc_auc_score for DecisionTree: ', metrics.roc_auc_score(y_test, y_sc
```

```
roc_auc_score for DecisionTree:  0.7575200534759358
```

In [165…

```python
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