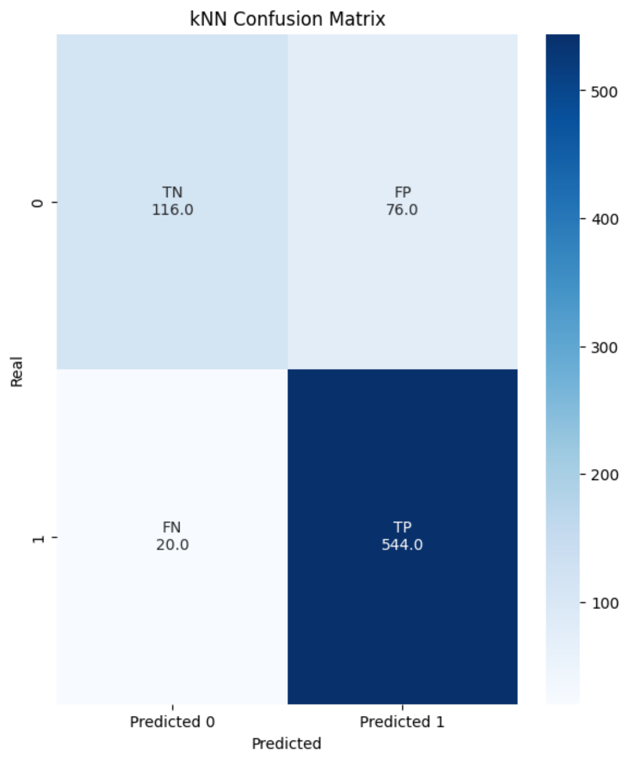
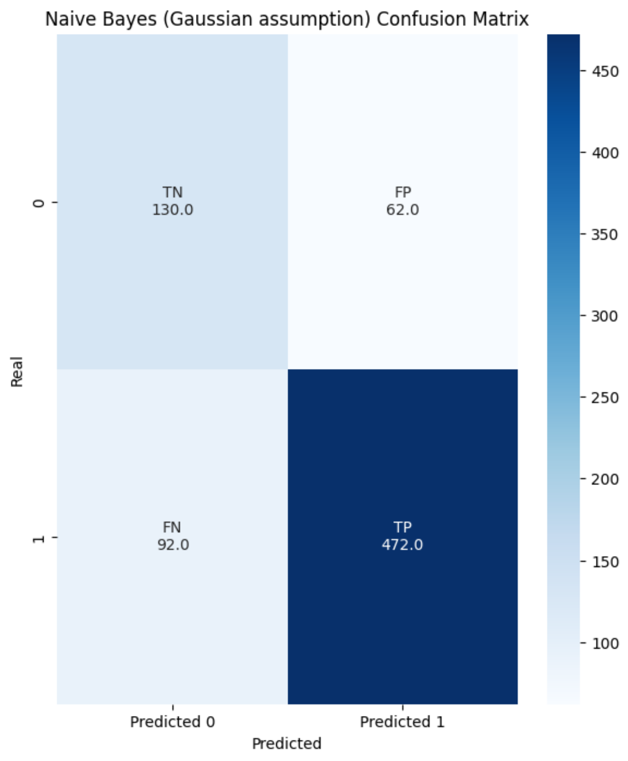
II. Programming and critical analysis

The code for the questions is in the Appendix of this document.

1)

Resulting Matrix:





2) **ttest_rel** function from **scipy.stats**, calculates the t-test on two related samples of scores. In this case, both classifiers' (kNN and Naive Bayes) scores are related since both are working in the same dataset. We used the parameter *alternative = 'greater'* once we're testing whether kNN's accuracy is statistically superior to Naive Bayes' accuracy. Therefore, we have the follow hypothesis:

- H0: kNN's accuracy is statistically equal to Naive Bayes' accuracy
- H1: kNN's accuracy is statistically superior to Naive Bayes' accuracy

After performing the test we obtained a *p-value* of 0.001368. This leads us to reject H0 at 1%, confirming that kNN's accuracy is statistically superior.

3) Note: The FAQ states that the answer could mention only two reasons.

In the previous question, we confirmed that kNN's accuracy is statistically superior to Naive Bayes' accuracy and so there are a couple of reasons to this observation:

- The Naive Bayes' model assumes that all dataset's features are mutually independent. However, in the dataset we're working with, the features are associated with each other which could lead to a loss of information and a smaller accuracy.

- The Naive Bayes' Gaussian assumption assumes the features follow a Normal distribution which can not be the case. Therefore, assuming this can underfit the model once we're not taking into account the features' distributions leading to a smaller accuracy.

III. APPENDIX

Code used in question 1) of **II. Programming and critical analysis**:

Loads the Parkinson Disease's Data:

```
import numpy as np
import pandas as pd
from scipy.io.arff import loadarff
import seaborn as sns
import matplotlib.pyplot as plt

data = loadarff('pd_speech.arff')
df = pd.DataFrame(data[0])
df['class'] = df['class'].str.decode('utf-8')

X = df.drop('class', axis=1)
Y = df['class']
```

Creates the 10-fold cross validator, the KNN and the Gaussian Naive Bayes. Calculates the confusion matrix of kNN and Naive Bayes:

```
from sklearn.model_selection import StratifiedKFold
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.preprocessing import StandardScaler

skf = StratifiedKFold(n_splits=10, shuffle=True, random_state=0)

knn = KNeighborsClassifier(n_neighbors=5, weights='uniform',
metric='euclidean')
nb = GaussianNB()

#function to calculate confusion matrix and accuracy so we dont repeat code
def calculate(x, X, Y):
   accuracy = []
   conf_mat = np.zeros((2, 2)) #inicializates the confusion matrix at 0's
   for train, test in skf.split(X, Y):
       X_train, X_test = X.iloc[train], X.iloc[test]
       Y_train, Y_test = Y.iloc[train], Y.iloc[test]
       #normalize data
       scaler = StandardScaler().fit(X_train)
       X_train = scaler.transform(X_train)
       X_test = scaler.transform(X_test)

       x.fit(X_train, Y_train)
       Y_pred = x.predict(X_test)

       accuracy.append(metrics.accuracy_score(Y_test, Y_pred))
       conf_mat += np.array(confusion_matrix(Y_test, Y_pred, labels=['0',
'1']))
   return conf_mat, accuracy

nb_confusion = pd.DataFrame(calculate(nb, X, Y)[0], index=['0', '1'],
columns=['Predicted 0', 'Predicted 1'])
knn_confusion = pd.DataFrame(calculate(knn, X, Y)[0], index=['0', '1'],
columns=['Predicted 0', 'Predicted 1'])
```

Plots the accuracies in a graph:

```
titles = ["kNN Confusion Matrix", "Naive Bayes (Gaussian assumption) Confusion
Matrix"]
matrices = [knn_confusion, nb_confusion]
plt.figure(figsize= (15, 8))
for i in range(len(matrices)):
   plt.subplot(1, 2, i+1)
   plt.title(titles[i])
   labels = np.array([["TN\n" + str(matrices[i].iat[0, 0]), "FP\n" +
str(matrices[i].iat[0, 1])], ["FN\n" + str(matrices[i].iat[1, 0]), "TP\n" +
str(matrices[i].iat[1, 1])]])
   sns.heatmap(matrices[i], annot=labels, fmt='', cmap="Blues")
   plt.xlabel("Predicted")
   plt.ylabel("Real")
```

Code used in question 2) of **II. Programming and critical analysis**:

```
#H0: kNN is statistically superior to Naive Bayes regarding accuracy
from scipy import stats
res = stats.ttest_rel(calculate(knn, X, Y)[1], calculate(nb, X, Y)[1],
alternative='greater')
print("kNN < nb? P-value =", res.pvalue)
```