# APPENDIX

```
1. import math
2. import numpy as np
3. import pandas as pd
4. import seaborn as sns
5. import matplotlib.pyplot as plt
6.
7. #LIBRARY
8.
9. from sklearn.model_selection import train_test_split
10. from sklearn import preprocessing
11. from sklearn.metrics import accuracy_score
12. from sklearn.naive_bayes import GaussianNB
13. from numpy import mean
14. --------------------IMPORT & READ DATA--------------------
15. breast_cancer    =    pd.read_csv('C:\\Users\Hp\Downloads\jurnal    yg
   dipakai\BCData.csv', delimiter=';')
16. breast_cancer
17. breast_cancer.shape
18. breast_cancer['Class'].value_counts()
19. breast_cancer['Bare_Nuclei'].value_counts()
20. --------------------DATA CLEANING--------------------
21. data = breast_cancer.drop('Sample_code_number', axis=1)
22. --------------------Delete empty value '?'----------------------
23. data1 = data.loc[~data.eq('?').any(1)]
24. data1['Bare_Nuclei'].value_counts()
25. data1['Class'].value_counts()
26. import csv
27. --------------------SAVE THE CLEAN DATA TO data1.csv-------------------
   -
28. data1.to_csv(r'C:\\Users\Hp\Downloads\jurnal   yg   dipakai\data1.csv',
   index='false?')
29. data1.columns
30. data1=data1.astype({"Bare_Nuclei":int}, errors = 'raise')
31. data1.info()
32. data1.shape
33. --------------------VISUALIZE THE COUNT--------------------
34. sns.countplot(data1.Class,label="count")
35. plt.show()
36. data1.hist(figsize=(20,15));
37. --------------------CHECK IF THERE IS A NULL VALUE--------------------
38. data1.isnull().sum()
39. data1.isna().sum()
40. df1= data1.sample(n=683, replace=False)
41. --------------------SET VAR. INDEPENDEN--------------------
42. X = df1.drop('Class',axis=1)
43. --------------------SET VAR. DEPENDEN--------------------
44. y = df1.Class
45. ---------------SPLIT DATA INTO TRAINING AND TESTING---------------
46. # Splitting
47. X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,
   shuffle= True, random_state=42)
48. # Printing shapes of testing and training sets :
```

```
print("shape of original dataset :", df1.shape)
print("\nshape of input data training :", X_train.shape)
print("\nshape of input data testing :", X_test.shape)
# Sort data into classes
Xy2 = X_train[y == 2]
Xy4 = X_train[y == 4]
print(Xy2.shape, Xy4.shape)
--------------------CALCULATE PRIORS--------------------
priory2 = len(Xy2) / len(X_train)
priory4 = len(Xy4) / len(X_train)
print(priory2, priory4)
-----------------------NAÏVE BAYES CLASSIFIER---------------------
from sklearn.naive_bayes import GaussianNB

model = GaussianNB()
nbTrain= model.fit(X_train, y_train)
# Menghitung distribusi kelas pada data training
nbTrain.class_count_
# Predict the classes on the test data, and return the probabilities for each class
y_proba = nbTrain.predict_proba(X_test)
y_proba
# Menentukan hasil prediksi kelas dari x_test
y_prediction=model.predict(X_test)
y_prediction
arr = y_test.to_numpy()
--------------------CONFUSION MATRIX--------------------
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(arr, y_prediction)
print(cm)
--------------------VISUALIZE CONFUSION MATRIX--------------------
import seaborn as sns
group_names = ['True Neg','False Pos','False Neg','True Pos']
group_counts = ["{0:0.0f}".format(value) for value in
                cm.flatten()]
group_percentages    =    ["{0:.2%}".format(value)    for    value    in cm.flatten()/np.sum(cm)]
labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
ax = sns.heatmap(cm, annot=labels, fmt='', cmap='Blues')
ax.set_title('Seaborn Confusion Matrix with labels\n\n');
ax.set_xlabel('\nPredicted Label')
ax.set_ylabel('True Label ');
## Ticket labels - List must be in alphabetical order
ax.xaxis.set_ticklabels(['False','True'])
ax.yaxis.set_ticklabels(['False','True'])
## Display the visualization of the Confusion Matrix.
plt.show()
--------------------RESULT OF NAÏVE BAYES--------------------
from sklearn.metrics import classification_report
print(classification_report(arr, y_prediction))
print("Actual diagnosis is \n {} \nThe Prediction is \n {} \nWhere 2 is benign and 4 is malignant".format(arr, y_prediction))
--------------------ROC VISUALIZE--------------------
from sklearn.metrics import roc_auc_score
```

```
score = roc_auc_score(arr, y_proba[:, 1])
print(f"ROC AUC: {score:.4f}")
def calculate_tpr_fpr(arr, y_prediction):
    # Calculates the confusion matrix and recover each element
    #cm = confusion_matrix(arr, y_prediction)
    TN = cm[0, 0]
    FP = cm[0, 1]
    FN = cm[1, 0]
    TP = cm[1, 1]
    # Calculates tpr and fpr
    tpr =  TP/(TP + FN) # sensitivity - true positive rate
    fpr = 1 - TN/(TN+FP) # 1-specificity - false positive rate
    return tpr, fpr
def get_n_roc_coordinates(arr, y_proba, n = 50):
    tpr_list = [0]
    fpr_list = [0]
    for i in range(n):
        threshold = i/n
        y_prediction = y_proba[:, 1] > threshold
        tpr, fpr = calculate_tpr_fpr(arr, y_prediction)
        tpr_list.append(tpr)
        fpr_list.append(fpr)
    return tpr_list, fpr_list
def plot_roc_curve(tpr, fpr, scatter = True):
    plt.figure(figsize = (5, 5))
    if scatter:
        sns.scatterplot(x = fpr, y = tpr)
    sns.lineplot(x = fpr, y = tpr)
    sns.lineplot(x = [0, 1], y = [0, 1], color = 'green')
    plt.xlim(-0.05, 1.05)
    plt.ylim(-0.05, 1.05)
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate")
# Calculates 10 coordinates of the ROC Curve
tpr, fpr = get_n_roc_coordinates(arr, y_proba)
# Plots the ROC curve
plot_roc_curve(tpr, fpr)
```

PAPER NAME

17.K1.0018.docx

WORD COUNT

4509 Words

CHARACTER COUNT

22286 Characters

PAGE COUNT

23 Pages

FILE SIZE

460.3KB

SUBMISSION DATE

Jul 20, 2022 12:53 PM GMT+7

REPORT DATE

Jul 20, 2022 12:53 PM GMT+7

## 15% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 8% Internet database
- 8% Publications database
- Crossref database
- Crossref Posted Content database
- 12% Submitted Works database

## Excluded from Similarity Report

- Bibliographic material
- Quoted material
- Cited material
- Small Matches (Less then 10 words)