CHAPTER 5
IMPLEMENTATION AND RESULTS

5.1. Implementation

This chapter will explain the coding implementation for each algorithm, starting from XGBoost.

```
1. df=pd.read_csv(r'.../dataset_comparison - compared_data.csv')
2. df = df.loc[:, ~df.columns.str.contains('^Unnamed')]
3.
4. df.head()
```

The first is to read the dataset's CSV file, on line 1, pd.read_csv from pandas is used. Line 2, the code is used to remove 'Unnamed' rows as it tends to happen on CSVs. Line 4 is used to see the first few entries on the CSV, to check if the rows and their contents are correct.

```
5. X=df.iloc[:,:-3]
6. y=df.iloc[:,6]
```

Because there are multiple features, to get all of the necessary features, line 5 is used. X variable is used to contain features in the specific rows from the dataset using df.iloc which will contain all the rows starting from the first row until three rows before the last. on line 6, the variable y will be the labels, which will be taken from row 6.

```
7. le = LabelEncoder()
8. y = le.fit_transform(y)
```

To change the labels that are contained in variable y to an array of encoded values LabelEncoder() will be used, it will be contained to le as seen on line 7, and in line 8 the array of encoded labels will be contained back in y variable.

```
9. symptoms_list = []
10.
11. print('Gusi: masukan angka 0-5 terhadap keluhan: ')
12. gusi_list = ['tidak ada', 'sakit', 'bengkak', 'berdarah sulit sembuh',
    'abses/nanah', 'gingivitis']
```

```
13.
14. for number, symptom in enumerate(gusi_list):
15.     print(number, symptom)
16.
17. symptoms1 = int(input('Keluhan gigi: '))
18.
19. symptoms_list.append(symptoms1)
```

Line 9-19 is essentially just an example of how the symptoms will be shown and then inputted. Line 9 is the list that will contain all the symptoms inputted. Line 11-15 is to show the list of symptoms in that particular area of the mouth, line 14 uses enumerate to show the numbers for each symptom and also for the loop's range. Line 15 is the input in integer data type. Line 19 is to append the symptom inputted to symptoms_list.

```
20. X_train,X_test,y_train,y_test =
   train_test_split(X,y,test_size=0.2,random_state=0)
```

Line 20 is to split the features (X) and the labels (y) to training set and test set, it uses test_size = 0.2 which means that it is an 80% training set and 20% test set split. For the parameter random state, any number can be used, 0 is to simply ensure that the data split is the same every time.

```
21. if symptoms_list.count(0) >= 4:
22.     print('Not enough information')
23. else:
24.     inp_pred =  symptoms_list
25.     inp_pred = np.array(inp_pred).reshape((1,-1))
```

Line 21-25 is to ensure that there are more than four inputs (value 0<), otherwise the program would not be able to predict the symptoms very well. The most important part is on line 25, the inputs need to be reshaped as array (1, -1) to match the required input shape of the program.

```
26.     predict = model.predict(inp_pred)
27.     predict_proba = model.predict_proba(inp_pred)
```

Line 26 is the prediction function that only predicts the strongest probability of an illness, while line 27 will list the probability of illnesses as the output.

```
28.     print('Berikut adalah kemungkinan penyakit yang dimiliki:')
29.     pp_count = 0
30.     for pp_count in range(max_y_list+1):
31.         print(list_set_y_list_inverse[pp_count]+      ':      '      +
   str(predict_proba[0][pp_count]*100)[:5]+'%')
32.         pp_count += 1
```

Line 28-32 is to list out all probability of illness in percentage form. It uses inverse encoded labels in order to show what the illness actually is instead of its encoding as a number.

```
33. print(predict)
34. print(le.inverse_transform(predict)+        ':         '        +
   str(predict_proba[0][predict]*100)[:5]+'%')
```

Similar to line 28-32, line 33-34 essentially just shows the probability of illness in percentage form, but line 33-34 only shows the strongest probability of illness.

For random forest, there are a few key differences in preprocessing compared to XGBoost. The implementation of the random forest will be down below.

```
35. data=pd.DataFrame({
36.     'gusi':dentist.loc[:,'gusi'],
37.     'gigi':dentist.loc[:,'gigi'],
38.     'bibir':dentist.loc[:,'bibir'],
39.     'lidah':dentist.loc[:,'lidah'],
40.     'tenggorokan':dentist.loc[:,'tenggorokan'],
41.     'area_mulut':dentist.loc[:,'area_mulut'],
42.     'labels':dentist.loc[:,'labels']
43. })
```

The first explanation for the random forest is the feature locating process. Line 35-43 locates the feature rows on the dataset and names them the same way, and it will be contained in the data variable as a data frame.

```
44. X=data[['gusi', 'gigi', 'lidah', 'bibir', 'tenggorokan', 'area_mulut']]
```

```
45. y=data['labels']
46. X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size)
```

In line 44-45, the rows that are located and contained in the data variable will then be separated with two new variables X and y, for features and labels respectively. Afterward, on line 46, the X and y variable will be separated again into the training set and test set.

```
47. species_idx1 = rf.predict_proba([row])[0]
48. print(species_idx1)
49.
50. species_idx2 = rf.predict([row])[0]
51. print(species_idx2)
```

Line 47-48 is similar to 50-51 the only difference is line 47-48 shows all probability of illness, while line 50-51 will only show the strongest probability of illness.

For multilayer perceptrons, TensorFlow will be used and there are some similar processes with XGBoost's preprocessing.

```
52. df=pd.read_csv(r'.../dataset_comparison - compared_data.csv')
53. df = df.loc[:, ~df.columns.str.contains('^Unnamed')]
```

Similar to XGBoost, the first is to read the dataset's CSV file, on line 52, pd.read_csv from pandas is used. Line 53, the code is used to remove 'Unnamed' rows.

```
54. X, y = df.values[:, :-3], df.values[:, 6]
55.
56. le = LabelEncoder()
57.
58. X = X.astype('float32')
59. y = le.fit_transform(y)
60.
61. X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
62. print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
63.

64. n_features = X_train.shape[1]
```

Also similar to XGBoost’s preprocessing, X and y will be divided using pandas, because there are multiple rows of features, in order to get all of the necessary features, in line 54, X variable will contain the features while y variable will contain the labels. On line 56, label encoder can be called first and then contained in variable le. On line 58, because the multilayer perceptron requires the input to be float, the original features in integer data type have to be converted into float32 data type, and in line 59 the labels in y variable are encoded. For line 61 the X and y variable is split into the training set and test set, with the test size of 0.2 which means 80% of the data is for training, while 20% is for testing. Line 62 .shape is used so X and y will be in two-tuples form, so they will look like (125, 6) (14, 6) (125,) (14,), the first array of the tuple is the amount of data, while the second array is the amount of features.

```
65. model = Sequential()

66. model.add(Dense(10,                              activation='relu',
   kernel_initializer='he_normal', input_shape=(n_features,)))

67. model.add(Dense(8,                               activation='relu',
   kernel_initializer='he_normal'))

68. model.add(Dense(4, activation='softmax'))
```

In line 65, the sequential model is used for the MLP, and in line 66-67 dense layer is used with 10 dense layers as the first layer, and 8 dense layers as the second layers, using ReLU as activation function, and in line 68, the output layer is softmax with four possible outputs, as it is a multi-classification program.

```
69. row2 = []

70.

71. print('Gusi: masukan angka 0-5 terhadap keluhan: ')

72. gusi_list  =  ['tidak  ada',  'sakit',  'bengkak',  'berdarah  sulit
   sembuh', 'abses/nanah', 'gingivitis']

73.

74. for number, symptom in enumerate(gusi_list):

75.     print(number, symptom)

76.
```

```
77. symptoms1 = float(input('Keluhan gigi: '))
78.
79. row2.append(symptoms1)
```

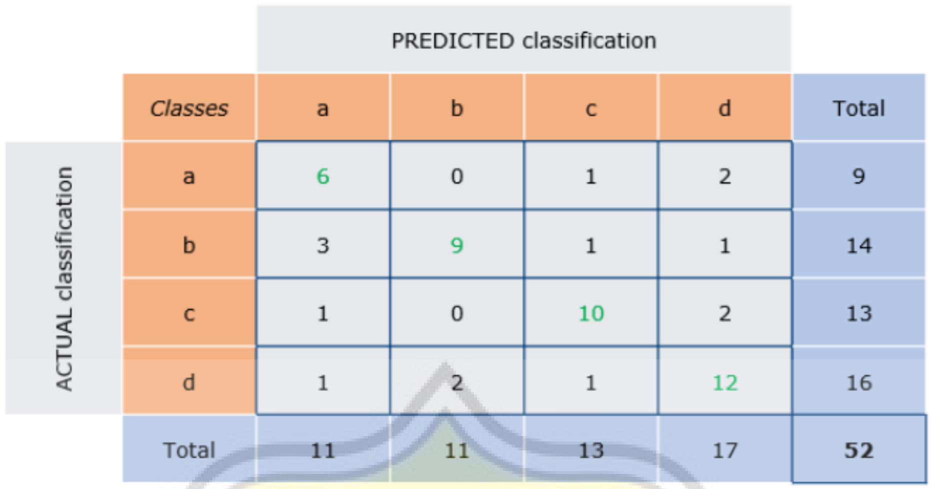
Similar to XGBoost, line 69-79 are an example of how the symptoms will be shown and then inputted. Line 69 is the list that will contain all the symptoms inputted. Line 71-75 is to show the list of symptoms in that particular area of the mouth. Line 77 is the input in the float data type as the MLP is trained with float. Line 79 is to append the symptom inputted to list symptoms1.

```
80. row = row2
81. yhat = model.predict([row])
82. for i in range(max(y+1)):
83.     print(str(le.inverse_transform([i])) +str(yhat[0][i]))
84.
85. print(str(le.inverse_transform([argmax(yhat)]))
   +str(yhat[0][argmax(yhat)]))
```

Line 80-81 is to predict from the list of appended input. Line 83 is to print the percentages of all the prediction confidence of each illness. Line 85 is to show the most probable illness.

5.2. Results

This chapter will discuss the results gathered from the algorithms when used with the dataset. Starting from the confusion matrix for the benchmark to calculate the accuracy, precision (Figure 5.2), recall (Figure 5.3), and F1-score (Figure 5.4) for the tests done. Starting with the explanation of the confusion matrix, with the numbers in the diagonal position meaning they are the True Positives, as seen in Figure 5.1 (True Positives in green-colored numbers).



		PREDICTED classification				
	Classes	a	b	c	d	Total
ACTUAL classification	a	6	0	1	2	9
	b	3	9	1	1	14
	c	1	0	10	2	13
	d	1	2	1	12	16
	Total	11	11	13	17	**52**

Figure 5.1 **Example of a multi-class confusion matrix**

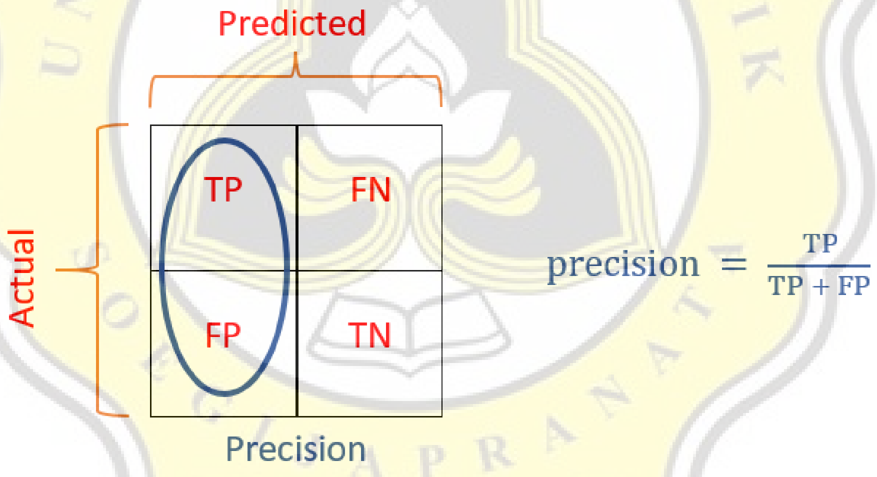


Figure 5.2 **Precision on confusion matrix**

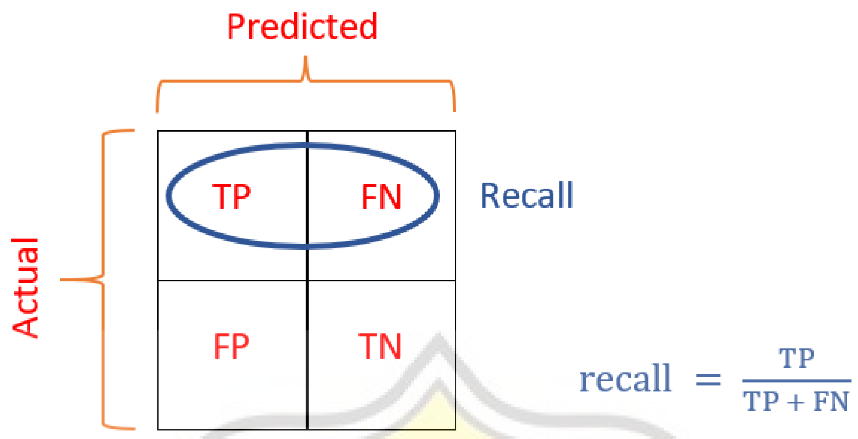


Figure 5.3 **Recall on confusion matrix**

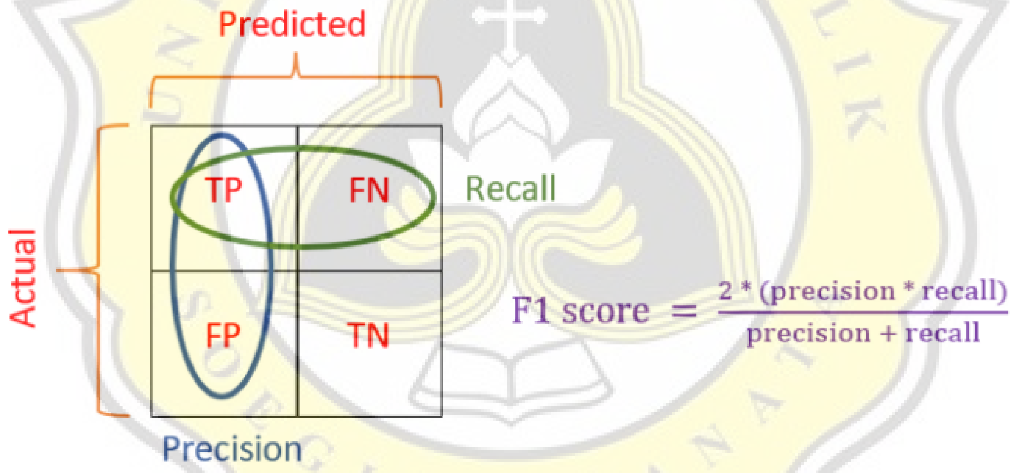


Figure 5.4 **F1-score on confusion matrix**

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 - Score = \frac{2 \times (precision \times recall)}{precision + recall}$$

Referring to Figure 5.1, and the accuracy will be

$$Accuracy = \frac{6 + 9 + 10 + 12}{52} = 0.52$$

To calculate the precision, recall, and F1-score. Referring back to Figure 5.1, we will take class "a" as an example to count the precision, take the top-left number (True Positive) which will be 6, and then divide it by the summation of numbers below it which will be (3 + 1 + 1 = 5), so the precision will be

$$Precision = \frac{6}{6 + 5} = 0.54$$

And for accuracy (referring to Figure 5.1), we will take a look at class "b" to calculate its precision, starting with the True Positive which will be 9, and then we'll take the number surrounding the True Positive and do summation on them, which will be (3 + 1 + 1 = 5), the calculation of recall (referring to Figure 5.1) will be

$$Recall = \frac{9}{9 + 5} = 0.64$$

And for the F1-score (referring to Figure 5.1) will be

$$F1 - score = \frac{2 \times (0.54 \times 0.64)}{(0.64 + 0.54)} = 0.58$$

After seeing the results of confusion matrix, and setting the benchmark for accuracy, precision, recall, and F1-score, results of multiple testings done on the dataset and the program will also be shown, such as seeing the result when actual symptoms are inputted, and also when fake symptoms are inputted and the observation. Test to see the result if the dataset is reduced with the test only done with entries that are more agreeable by annotators (agreed on four or more symptoms of an illness entry). Test with a 50:50, 70:30, 20:80 split of the training set and test set. And tests by trying different parameters in the algorithms. Finally, the conclusion for the overall analysis will be done, seeing the results of each algorithm and comparing them, interpreting why certain results are achieved.

The first is to see the confusion matrix results from all of the algorithms, all of the algorithms are using 80:20 split. Starting XGBoost's confusion matrix after being run ten different times, as seen in Figure 5.5-5.14

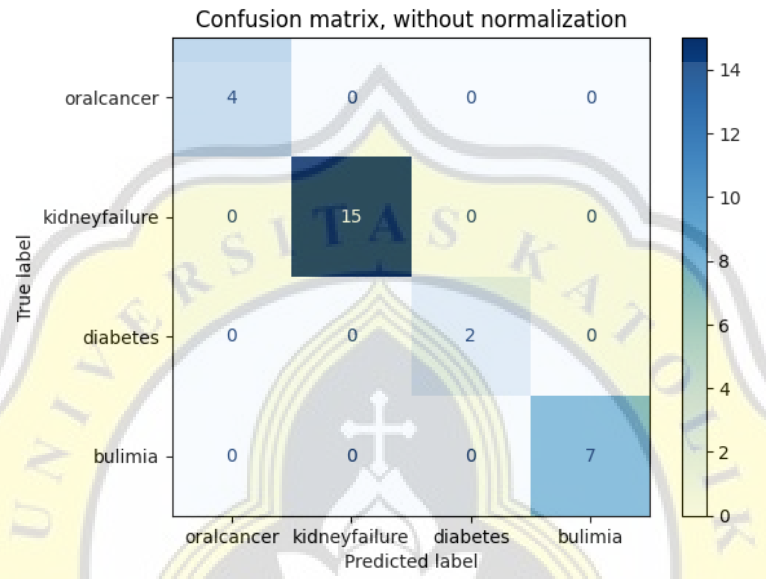


Figure 5.5 **Confusion matrix of XGBoost, run 1**

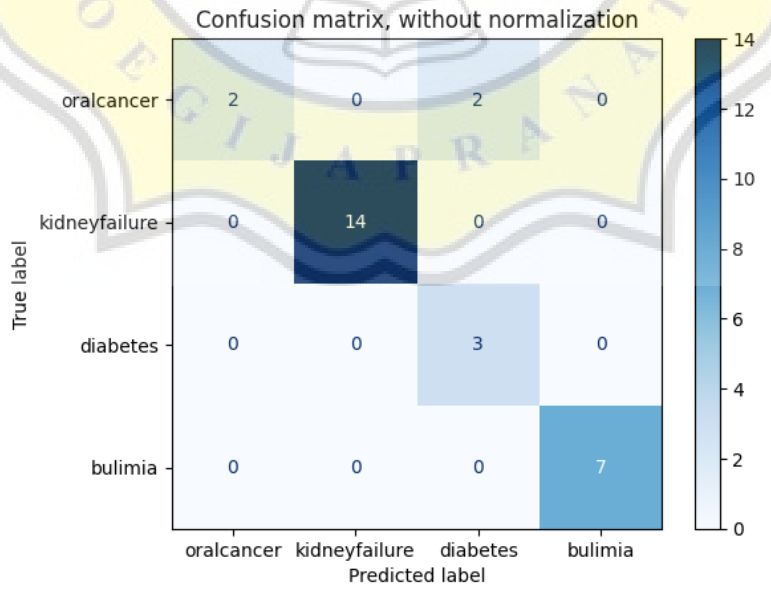
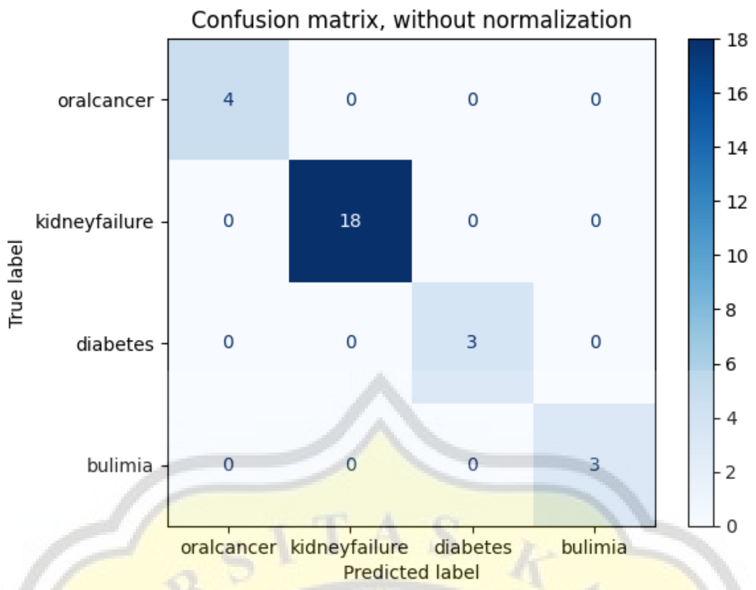
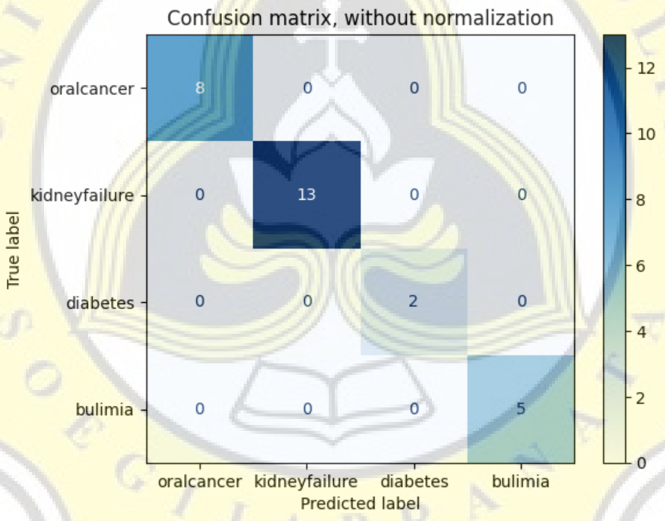


Figure 5.6 **Confusion matrix of XGBoost, run 2**



	oralcancer	kidneyfailure	diabetes	bulimia
oralcancer	4	0	0	0
kidneyfailure	0	18	0	0
diabetes	0	0	3	0
bulimia	0	0	0	3

Figure 5.7 **Confusion matrix of XGBoost, run 3**



	oralcancer	kidneyfailure	diabetes	bulimia
oralcancer	8	0	0	0
kidneyfailure	0	13	0	0
diabetes	0	0	2	0
bulimia	0	0	0	5

Figure 5.8 **Confusion matrix of XGBoost, run 4**

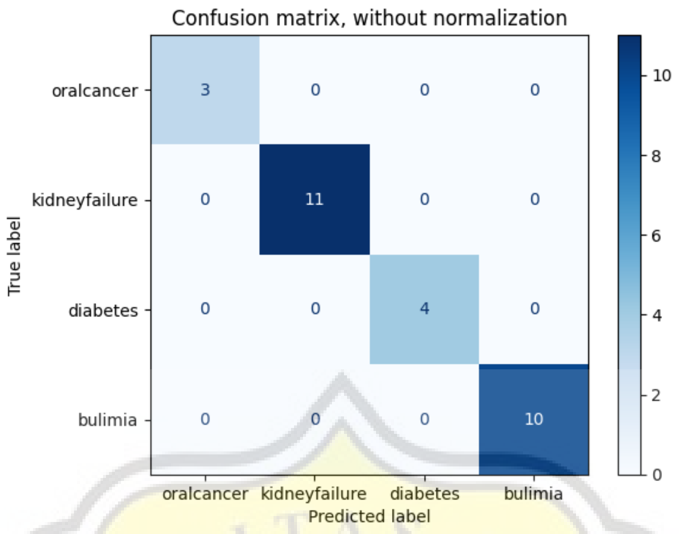


Figure 5.9 **Confusion matrix of XGBoost, run 5**

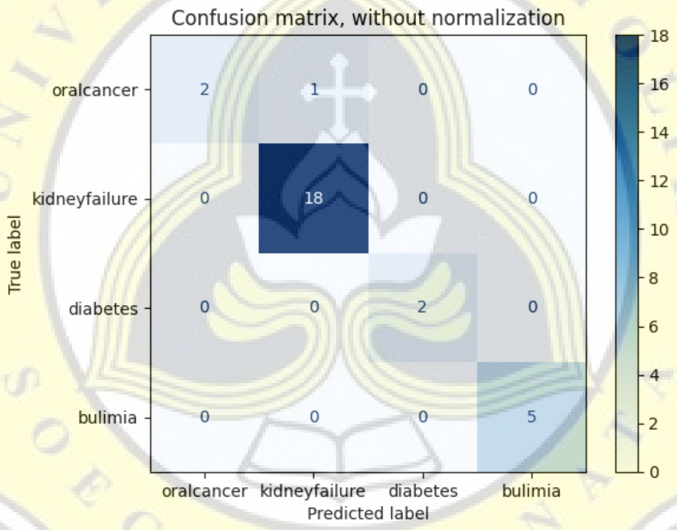


Figure 5.10 **Confusion matrix of XGBoost, run 6**

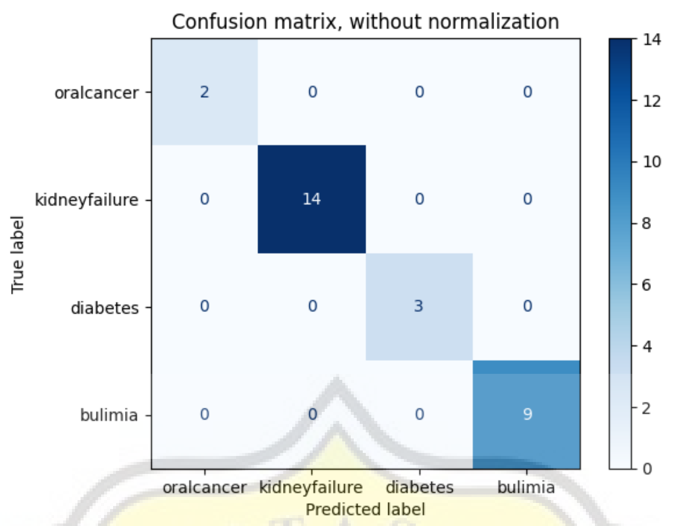


Figure 5.11 **Confusion matrix of XGBoost, run 7**

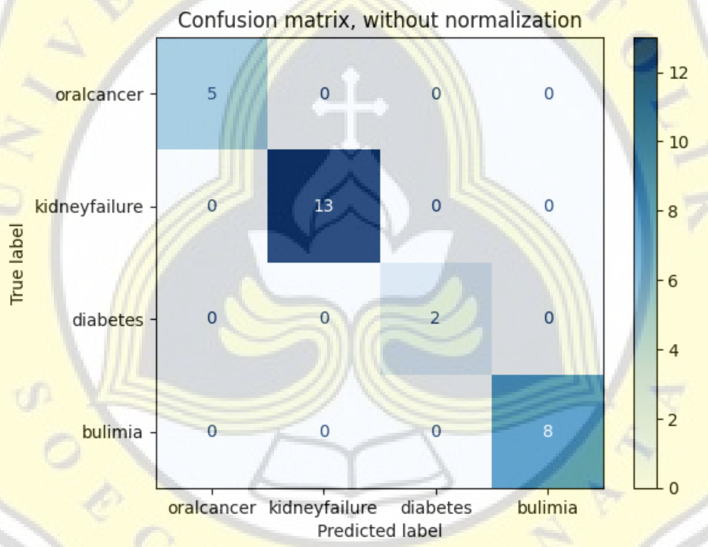


Figure 5.12 **Confusion matrix of XGBoost, run 8**

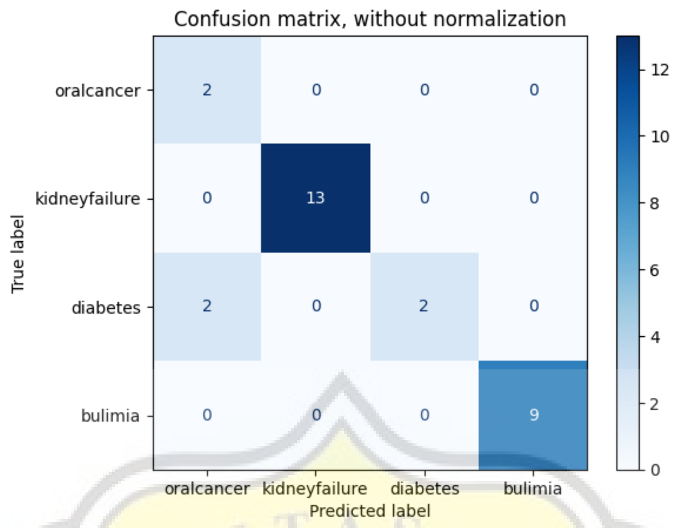


Figure 5.13 **Confusion matrix of XGBoost, run 9**

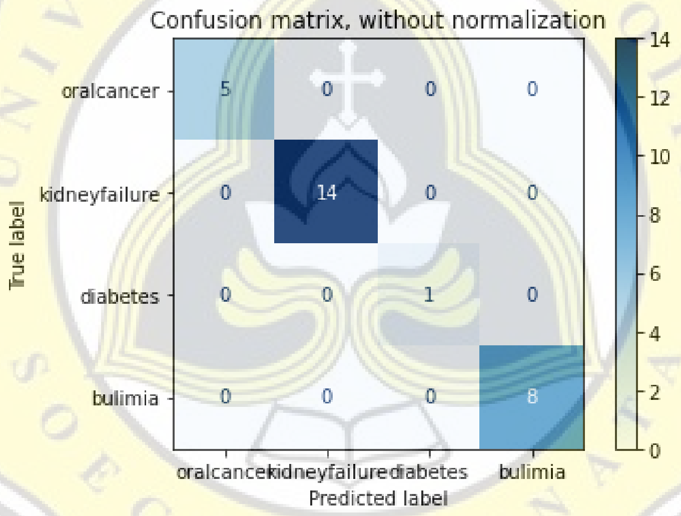


Figure 5.14 **Confusion matrix of XGBoost, run 10**

The XGBoost's confusion matrix shows somewhat pleasing results after ten runs as the matrices from the runs are mostly perfect diagonals, this means that there are mostly no false positives or false negatives in the confusion matrix results, and all the predicted values are predicted correctly, but this could also be an issue, that will be discussed later on. The label that gets misclassified the most is interestingly oral cancer with diabetes and vice versa, as it shares similar features, especially with diabetes. The average result of the accuracy, precision, recall, and F1-score of the XGBoost's confusion matrix from the runs are

$$Average\ Accuracy = \frac{(1 + 0.86 + 1 + 1 + 1 + 1 + 1 + 1 + 0.92 + 1)}{10} = 0.98$$

$$Average\ Precision\ (Oral\ Cancer) = \frac{(1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 0.5 + 1)}{10}$$
$$= 0.95$$

$$Average\ Precision\ (Kidney\ Failure) = \frac{(1 + 1 + 1 + 1 + 1 + 0.95 + 1 + 1 + 1 + 1)}{10}$$
$$= 0.995$$

$$Average\ Precision\ (Diabetes) = \frac{(1 + 0.4 + 1 + 1 + 1 + 1 + 1 + 1 + 0.5 + 1)}{10}$$
$$= 0.89$$

$$Average\ Precision\ (Bulimia) = \frac{(1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1)}{10} = 1$$

$$Average\ Recall\ (Oral\ Cancer) = \frac{(1 + 0.5 + 1 + 1 + 1 + 0.67 + 1 + 1 + 0.5 + 1)}{10}$$
$$= 0.867$$

$$Average\ Recall\ (Kidney\ Failure) = \frac{(1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1)}{10} = 1$$

$$Average\ Recall\ (Diabetes) = \frac{(1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 0.5 + 1)}{10} = 0.95$$

$$Average\ Recall\ (Bulimia) = \frac{(1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1)}{10} = 1$$

$$Average\ F1 - Score\ (Oral\ Cancer) = \frac{2 \times (0.95 \times 0.867)}{0.95 + 0.867} = 0.91$$

$$Average\ F1 - Score\ (Kidney\ Failure) = \frac{2 \times (0.995 \times 1)}{0.995 + 1} = 0.99$$

$$Average\ F1 - Score\ (Diabetes) = \frac{2 \times (0.89 \times 0.95)}{0.89 + 0.95} = 0.92$$

$$Average\ F1 - Score\ (Bulimia) = \frac{2 \times (1 \times 1)}{1 + 1} = 1$$

The random forest's confusion matrix is similar to XGBoost's confusion matrix, results after ten runs as the matrices from the runs are also mostly perfect diagonals, this once again means that in most runs, there are no false positives or false negatives, and all the predicted values are predicted correctly as seen in Figure 5.15-5.24.

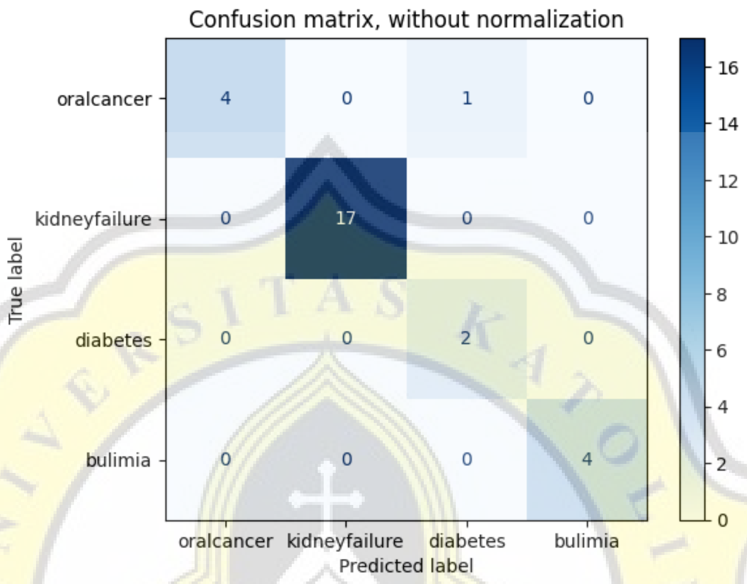


Figure 5.15 **Confusion matrix of random forest, run 1**

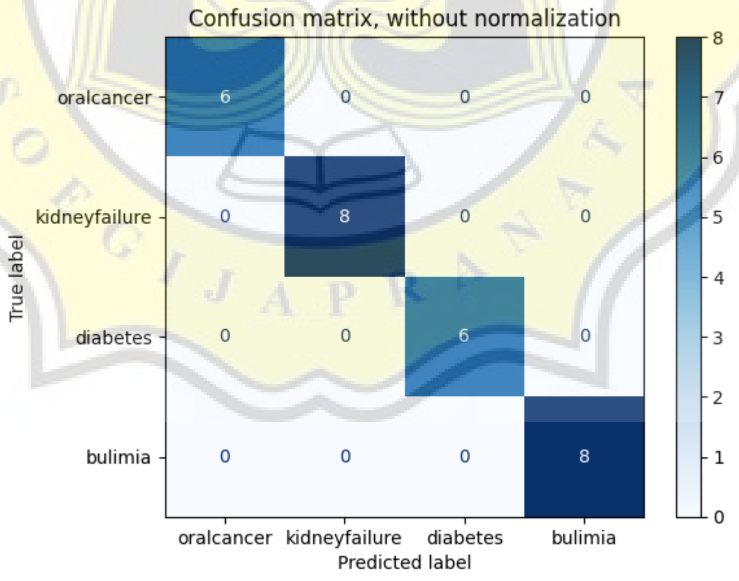


Figure 5.16 **Confusion matrix of random forest, run 2**

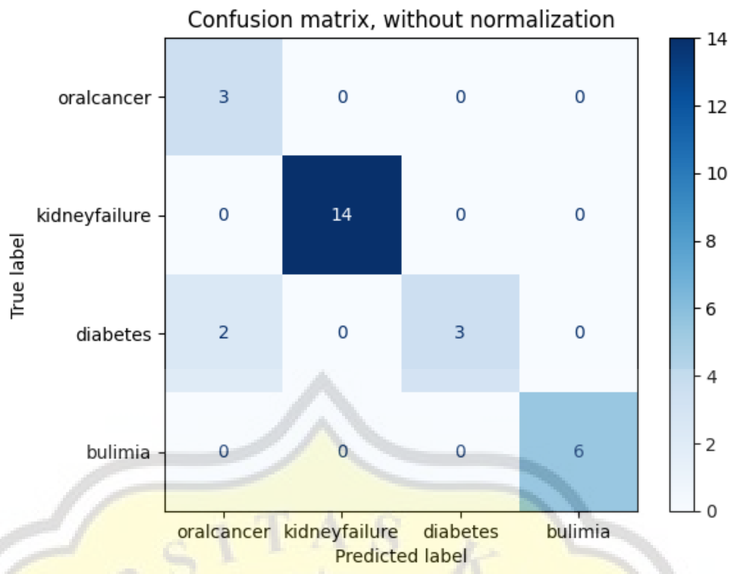


Figure 5.17 **Confusion matrix of random forest, run 3**

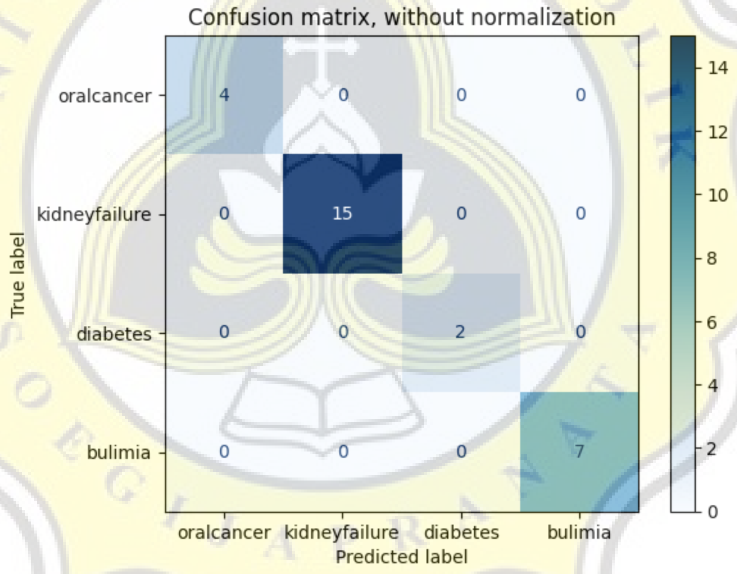


Figure 5.18 **Confusion matrix of random forest, run 4**

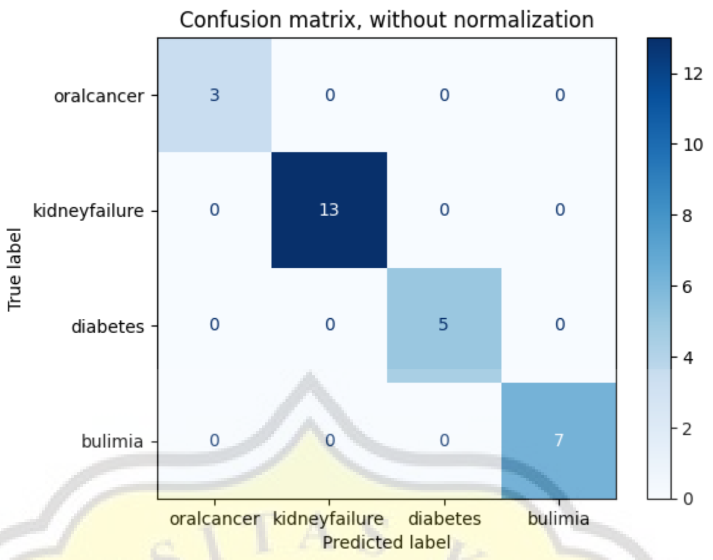


Figure 5.19 **Confusion matrix of random forest, run 5**

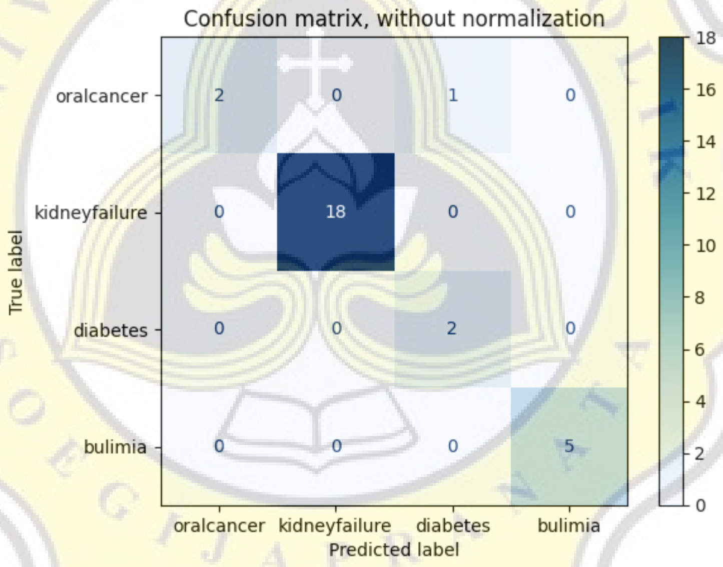


Figure 5.20 **Confusion matrix of random forest, run 6**

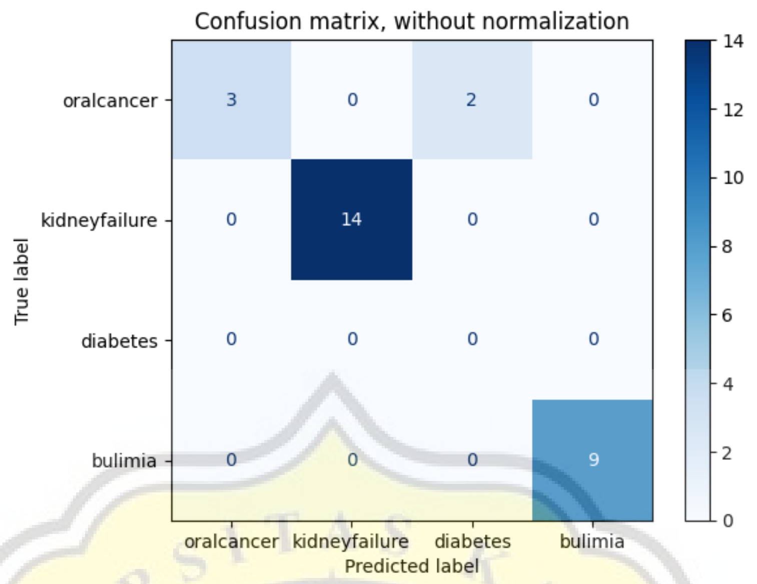


Figure 5.21 **Confusion matrix of random forest, run 7**

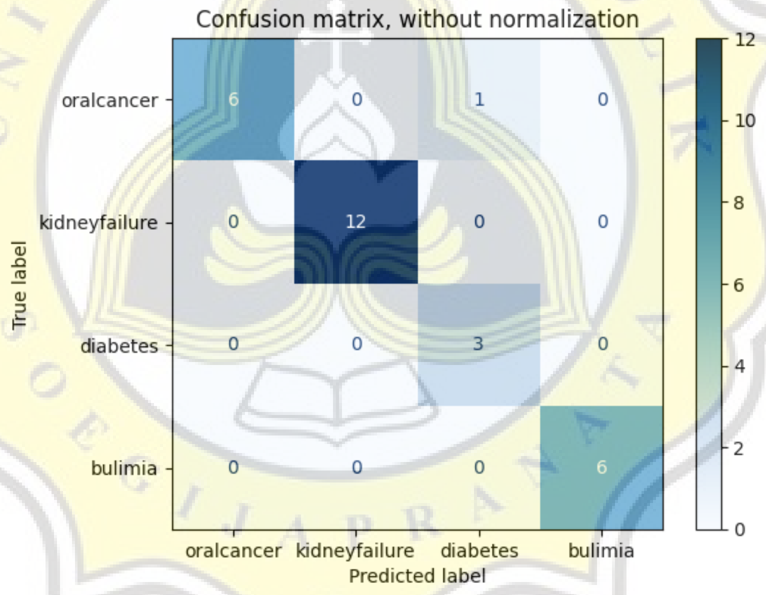


Figure 5.22 **Confusion matrix of random forest, run 8**

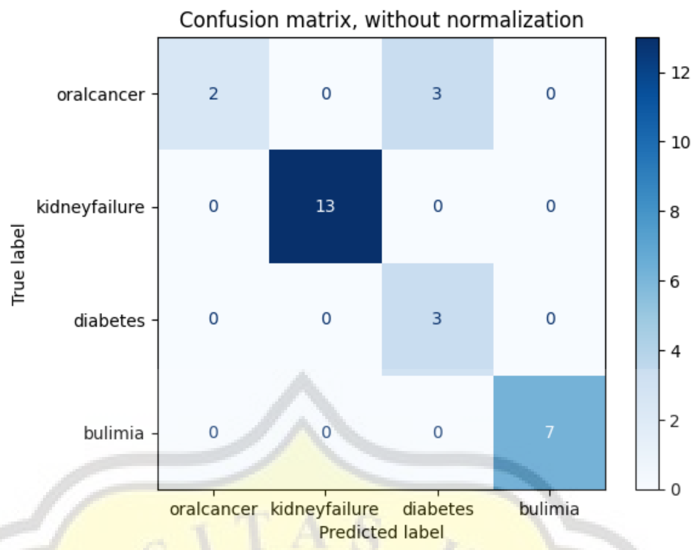


Figure 5.23 **Confusion matrix of random forest, run 9**

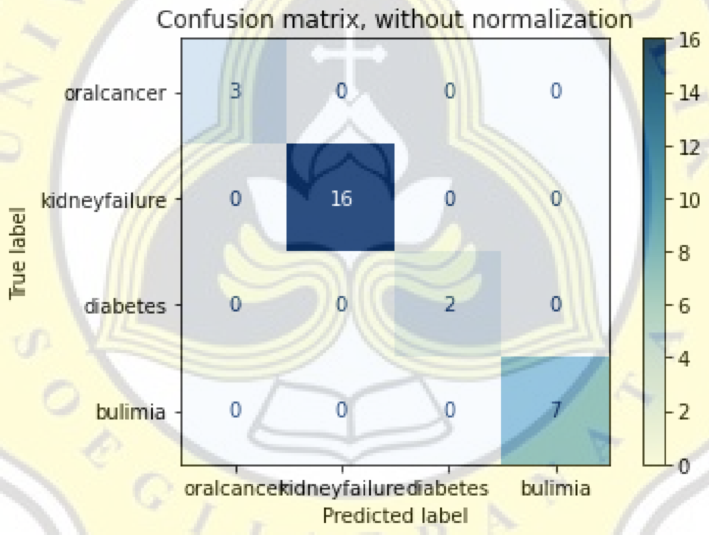


Figure 5.24 **Confusion matrix of random forest, run 10**

After running the program ten times, the result of the confusion matrix and its accuracy is almost always the same for each run, the accuracy being 1.0 or 100%. From the results shown, the random forest's calculation of the average accuracy, precision, recall, and F1-score will be

$$Average\ Accuracy = \frac{(0.96 + 0.93 + 1 + 1 + 1 + 1 + 0.93 + 0.96 + 0.90 + 1)}{10}$$

$$= 0.97$$

$$Average\ Precision\ (Oral\ Cancer) = \frac{(1+1+0.6+1+1+1+1+1+0.5+1)}{10}$$

$$= 0.96$$

$$Average\ Precision\ (Kidney\ Failure) = \frac{(1+1+1+1+1+1+1+1+1+1)}{10}$$

$$= 1$$

$$Average\ Precision\ (Diabetes) = \frac{(1+.1+1+1+1+1+0.67+0.75+0.5+1)}{10}$$

$$= 0.802$$

$$Average\ Precision\ (Bulimia) = \frac{(1+1+1+1+1+1+1+1+1+1)}{10} = 1$$

$$Average\ Recall\ (Oral\ Cancer) = \frac{(1+1+0.6+1+1+1+1+1+1+1)}{10} = 9.6$$

$$Average\ Recall\ (Kidney\ Failure) = \frac{(1+1+1+1+1+1+1+1+1+1)}{10} = 1$$

$$Average\ Recall\ (Diabetes) = \frac{(1+1+1+1+1+1+1+1+1+1)}{10} = 1$$

$$Average\ Recall\ (Bulimia) = \frac{(1+1+0.6+1+1+1+0+1+0.5+1)}{10} = 0.81$$

$$Average\ F1 - Score\ (Oral\ Cancer) = \frac{2 \times (0.96 \times 0.1)}{0.96+1} = 0.98$$

$$Average\ F1 - Score\ (Kidney\ Failure) = \frac{2 \times (1 \times 1)}{1+1} = 1$$

$$Average\ F1 - Score\ (Diabetes) = \frac{2 \times (0.802 \times 1)}{0.802+1} = 0.89$$

$$Average\ F1 - Score\ (Bulimia) = \frac{2 \times (1 \times 0.81)}{1+0.81} = 0.89$$

While similar results can be found for XGBoost and random forest, the same cannot be said with TensorFlow's multilayer perceptrons. First, it needs to be explained that the labels are in their encoded form, so label 0 will be oral cancer, 1 will be kidney failure, 2 will be diabetes, and 3 will be bulimia for XGBoost and random forest both keep the results perfectly diagonal consistently on each run of the program, but TensorFlow's multilayer perceptrons result does not have a consistent perfect diagonal as seen in Figure 5.25-5.34.

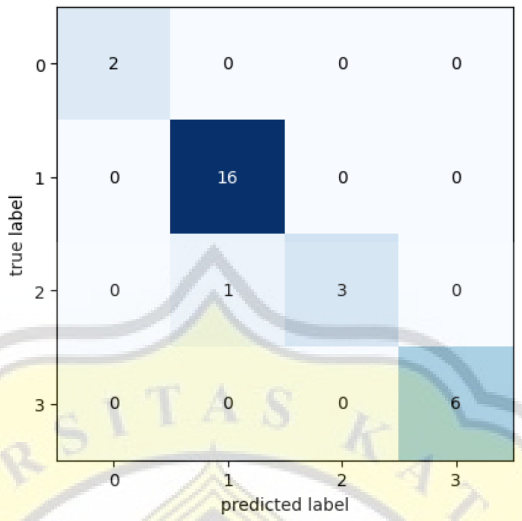


Figure 5.25 **Confusion matrix of TensorFlow's multilayer perceptrons, run 1**

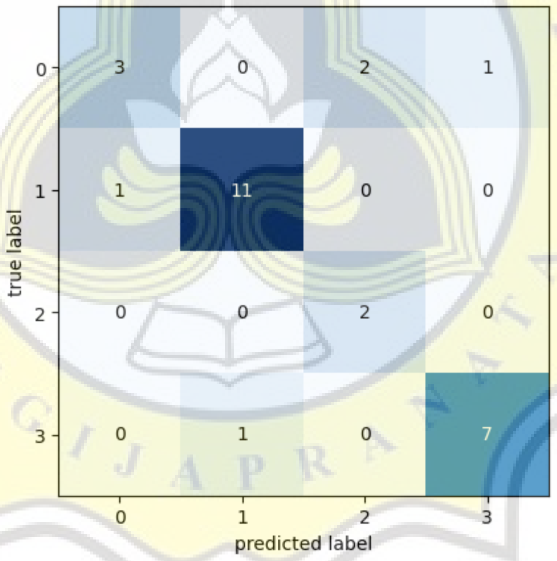


Figure 5.26 **Confusion matrix of TensorFlow's multilayer perceptrons, run 2**

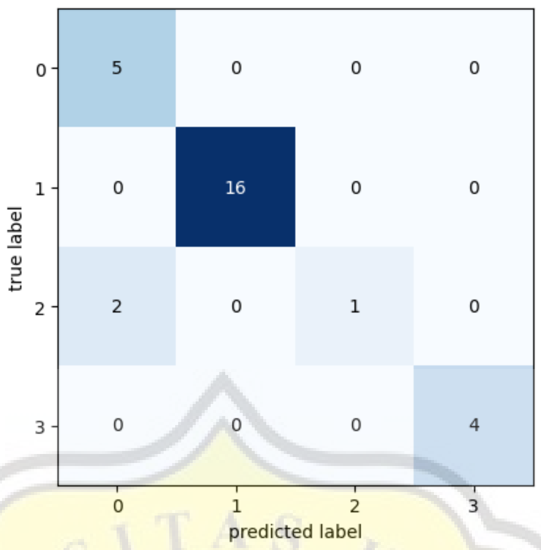


Figure 5.27 **Confusion matrix of TensorFlow's multilayer perceptrons, run 3**

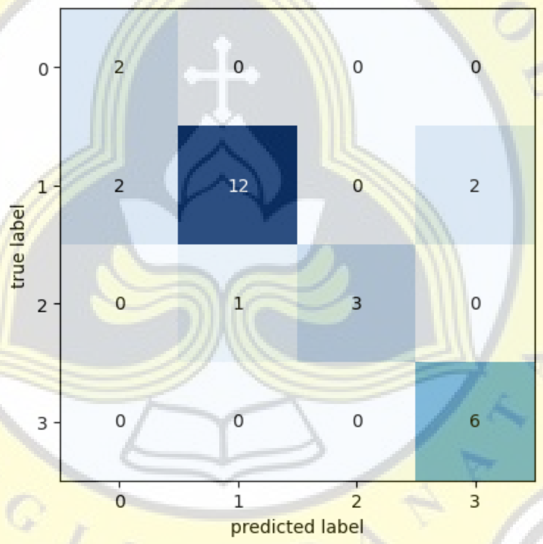


Figure 5.28 **Confusion matrix of TensorFlow's multilayer perceptrons, run 4**

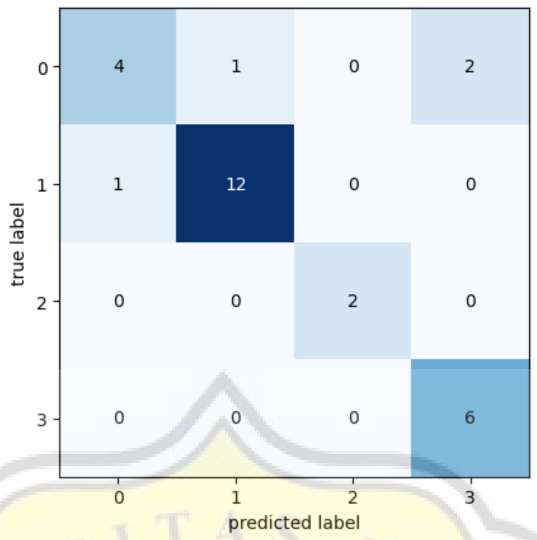


Figure 5.29 **Confusion matrix of TensorFlow's multilayer perceptrons, run 5**

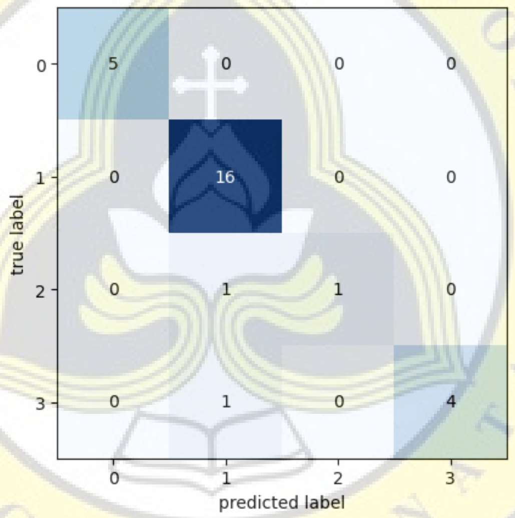


Figure 5.30 **Confusion matrix of TensorFlow's multilayer perceptrons, run 6**

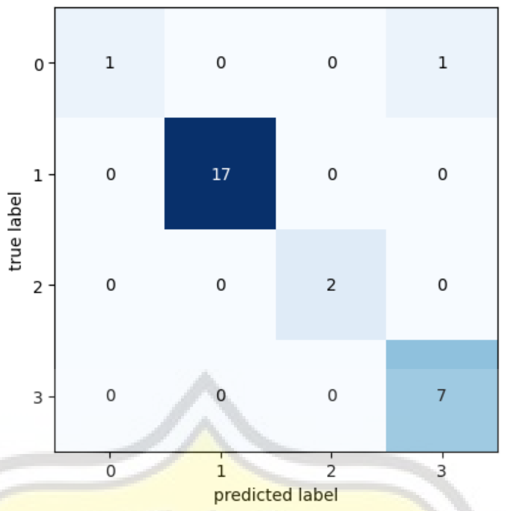


Figure 5.31 **Confusion matrix of TensorFlow's multilayer perceptrons, run 7**

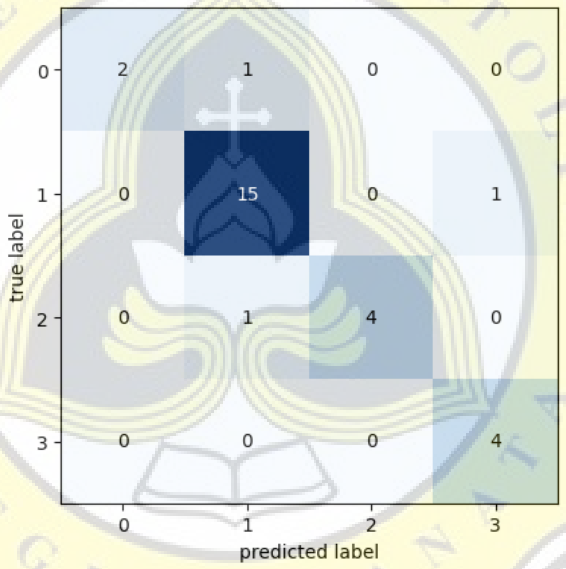


Figure 5.32 **Confusion matrix of TensorFlow's multilayer perceptrons, run 8**

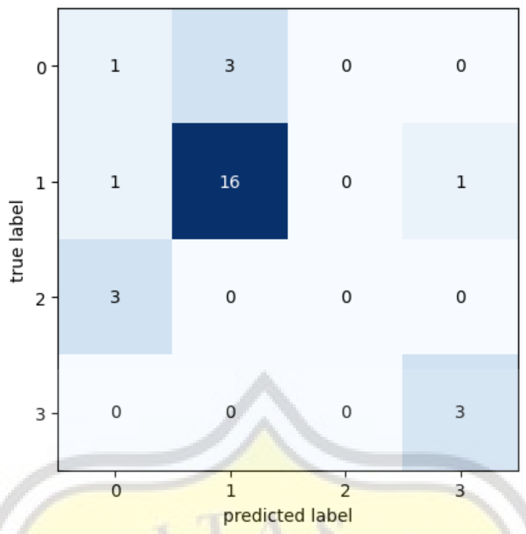


Figure 5.33 **Confusion matrix of TensorFlow's multilayer perceptrons, run 9**

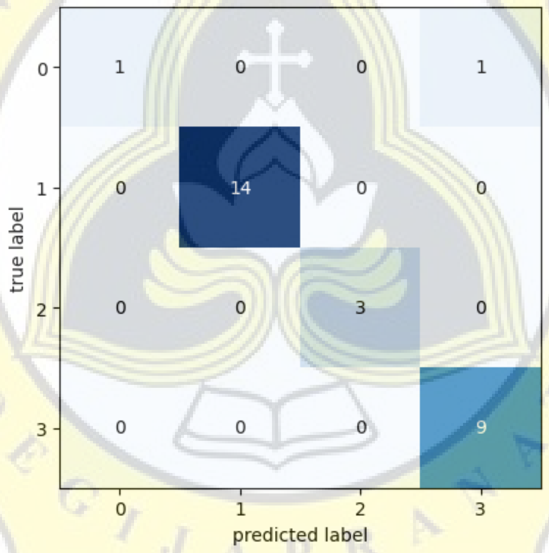


Figure 5.34 **Confusion matrix of TensorFlow's multilayer perceptrons, run 10**

The TensorFlow's multilayer perceptrons confusion matrix show favorable results, as after ten runs as the matrices from the there are many diagonal patterns even though not as much as XGBoost nor random forest. This means that there are more false positives and false negatives that are happening. The label that gets misclassified the most is also oral cancer with diabetes and vice versa, similar to the previous algorithms, as it shares similar features, especially with diabetes. The average result of the accuracy, precision, recall, and F1-score of the TensorFlow's multilayer perceptrons confusion matrix from the runs are

$$Average\ Accuracy = \frac{(0.96+0.86+0.93+0.82+0.86+0.93+0.96+0.89+0.71+0.96)}{10} = 0.89$$

$$Average\ Precision\ (Oral\ Cancer) = \frac{(0.75+1+0.71+0.5+0.8+1+1+1+0.2+1)}{10} = 0.796$$

$$Average\ Precision\ (Kidney\ Failure) = \frac{(0.94+0.92+1+0.92+0.89+1+0.89+0.88+0.84+1)}{10} = 0.93$$

$$Average\ Precision\ (Diabetes) = \frac{(1+0.5+1+1+1+1+1+1+0+1)}{10} = 0.85$$

$$Average\ Precision\ (Bulimia) = \frac{(1+0.88+1+0.75+0.75+1+0.88+0.8+0.75+0.9)}{10} = 0.87$$

$$Average\ Recall\ (Oral\ Cancer) = \frac{(0.5+1+1+1+0.57+0.67+0.5+0.67+0.2+0.5)}{10} = 0.66$$

$$Average\ Recall\ (Kidney\ Failure) = \frac{(0.94+0.92+1+0.92+0.92+0.89+1+0.88+0.89+1)}{10} = 0.94$$

$$Average\ Recall\ (Diabetes) = \frac{(0.74+1+0.33+0.75+1+0.5+1+0.8+0+1)}{10} = 0.71$$

$$Average\ Recall\ (Bulimia) = \frac{(1+0.88+1+1+1+0.8+1+1+1+1)}{10} = 0.97$$

$$Average\ F1-Score\ (Oral\ Cancer) = \frac{2\times(0.796\times0.66)}{0.796+0.66} = 0.72$$

$$Average\ F1 - Score\ (Kidney\ Failure) = \frac{2 \times (0.93 \times 0.94)}{0.93 + 0.94} = 0.93$$

$$Average\ F1 - Score\ (Diabetes) = \frac{2 \times (0.85 \times 0.71)}{0.85 + 0.71} = 0.77$$

$$Average\ F1 - Score\ (Bulimia) = \frac{2 \times (0.87 \times 0.97)}{0.87 + 0.97} = 0.92$$

After running all the programs each with a different algorithm, XGBoost and random forest triumph over TensorFlow's multilayer perceptrons with the average accuracy of 0.98, 0.97, and 0.89 respectively. The XGBoost and random forest results are more similar compared to TensorFlow's multilayer perceptron. The result of XGBoost and random forest are mostly perfect diagonals, with perfect scores, however as established when explaining the result of XGBoost, this could be an indication of issues and this does not always mean that the program will do well on a larger dataset, as after running the program ten times, the result of the confusion matrix and its accuracy is mostly the same for each run, the accuracy being 1.0 or 100%. This is a probable indication that overfitting occurs. Overfitting in this scenario most likely occurs because of the small amount of data. The reason why accuracy should not achieve 100% can be seen in Figure 5.35, to put it simply, if accuracy reaches 100% while still accurate when predicting, the result will be perceived as doubtful as it exceeds human capability, even the experts regarding the matter. Another issue found is data imbalance, as seen in Figure 5.36-5.38.

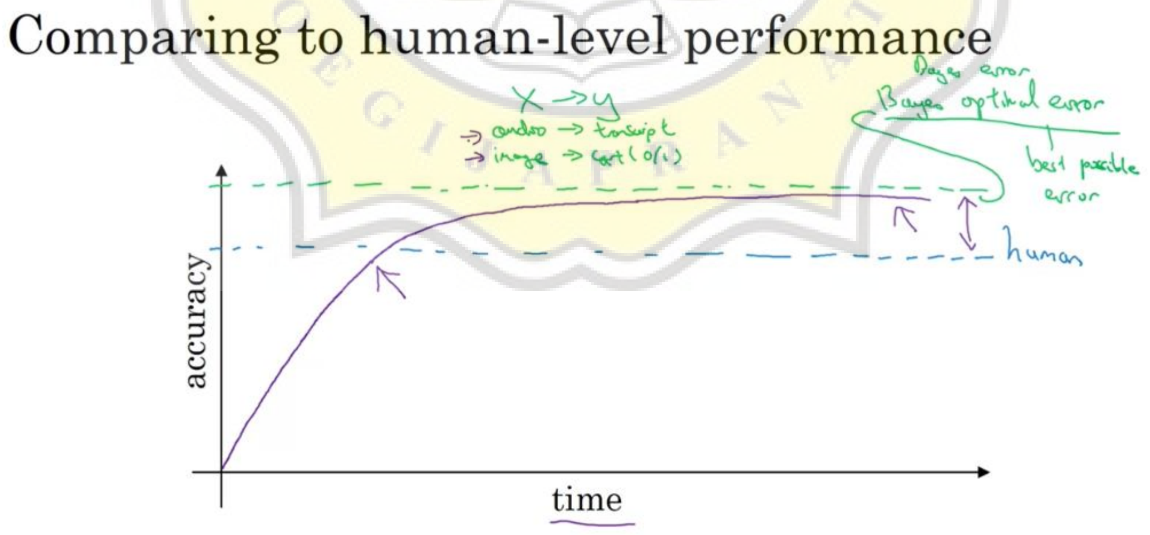
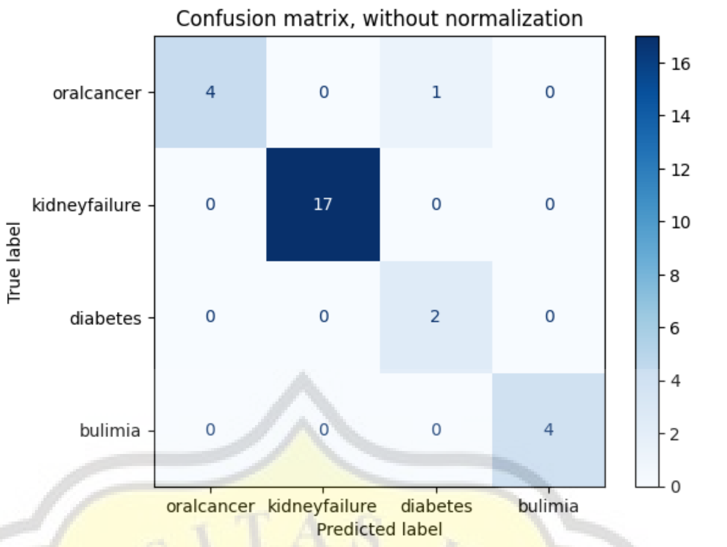
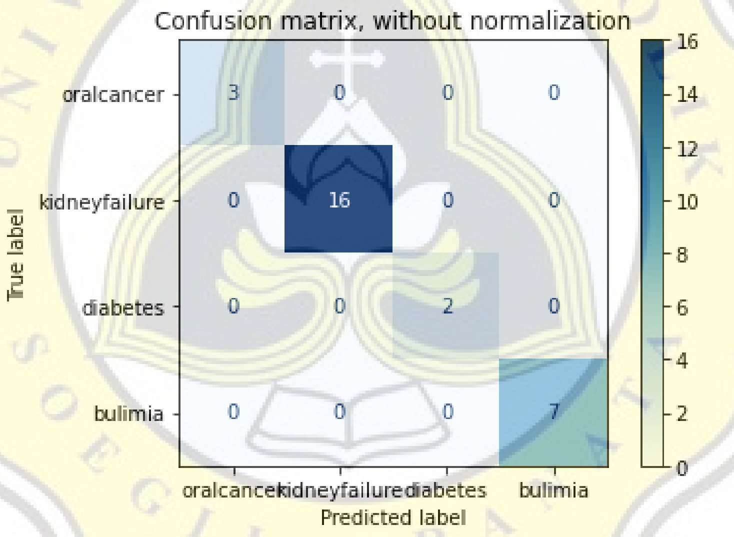


Figure 5.35 **Bayes optimal error, the limit of the highest acceptable machine's accuracy**



	oralcancer	kidneyfailure	diabetes	bulimia
oralcancer	4	0	1	0
kidneyfailure	0	17	0	0
diabetes	0	0	2	0
bulimia	0	0	0	4

Figure 5.36 **An example from the XGBoost's confusion matrix**



	oralcancer	kidneyfailure	diabetes	bulimia
oralcancer	3	0	0	0
kidneyfailure	0	16	0	0
diabetes	0	0	2	0
bulimia	0	0	0	7

Figure 5.37 **An example from the random forest's confusion matrix**

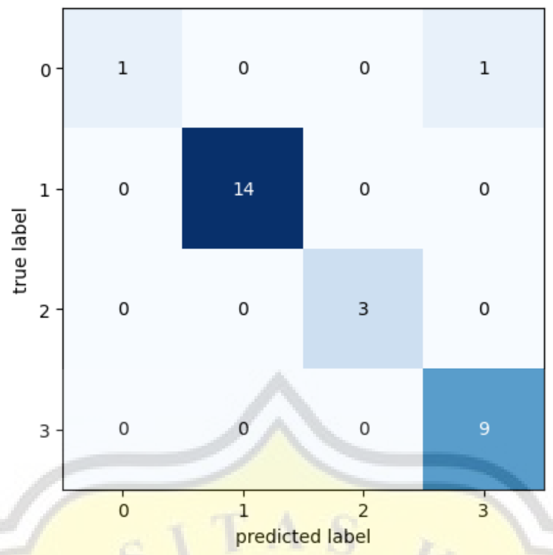


Figure 5.38 **An example from TensorFlow's multilayer perceptrons' confusion matrix**

Referring to Figure 5.36-5.38 it can be seen that there is another issue, which is data imbalance in the dataset, it is visible from the gradient differences in the matrix, as darker tone means more data is in the dataset, and in this case, kidney failure is seen prominently in darker tone in the gradient. One other prominent issue is that oral cancer and diabetes are often misclassified, and it occurs in all the algorithms as seen in Table 5.1, the diabetes and oral cancer symptoms are not really similar. This observation is interesting as the data in the dataset has diabetes entries higher than any other illness, hence more symptoms, and the other illnesses have similar numbers of data. The explanation that can be given is that some illnesses are closer in symptoms than other in terms of severity, hence why kidney failure entries as seen in Table 5.2, has more data as it is seen as more unique.

Table 5.1. Diabetes and oral cancer symtomps

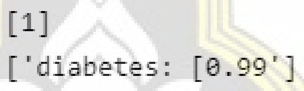
1	4	2	3	3	7	diabetes
1	1	2	3	3	7	diabetes
1	1	2	2	0	7	diabetes
3	1	2	2	3	7	diabetes
1	1	2	2	0	8	diabetes
1	5	8	4	1	6	oralcancer
1	5	8	4	2	6	oralcancer

0	5	8	4	1	9	oralcancer
0	5	8	4	2	9	oralcancer
1	5	8	4	1	8	oralcancer

Table 5.2. Kidney failure symptoms

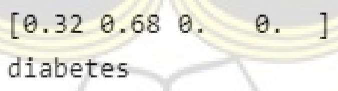
1	0	4	0	7	3	kidneyfailure
1	0	1	0	7	8	kidneyfailure
1	0	4	0	7	8	kidneyfailure
1	0	1	0	7	10	kidneyfailure
1	0	4	0	7	10	kidneyfailure

After seeing the result of the confusion matrix, we know the average accuracy, precision, recall, and F1-score, of each algorithm, because of this it is next the actual prediction of each illness will be observed. Starting from diabetes with the symptom of 3, 1, 6, 3, 0, 7, the result is seen in Figure 5.1-5.3.



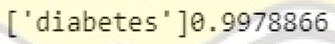
```
[1]
['diabetes: [0.99']
```

Figure 5.1 **Prediction of XGBoost for diabetes symptoms**



```
[0.32 0.68 0.   0.  ]
diabetes
```

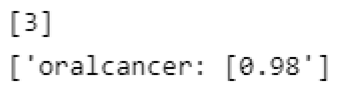
Figure 5.2 **Prediction of random forest for diabetes symptoms**



```
['diabetes']0.9978866
```

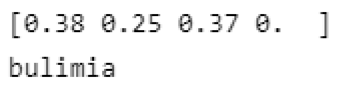
Figure 5.3 **Prediction of TensorFlow's multilayer perceptrons for diabetes symptoms**

Referring to Figure 5.1-5.3, it can be seen that the accuracy of the prediction confidence between XGBoost and TensorFlow's multilayer perceptrons are tied with 99% saying that the symptoms given are symptoms from diabetes, while the random forest algorithm prediction confidence is only at 68%. Afterward, the test is done with oral cancer symptoms 0, 5, 8, 4, 1, 9 the result is seen in Figure 5.4-5.6.



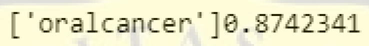
```
[3]
['oralcancer: [0.98']
```

Figure 5.4 **Prediction of XGBoost for oral cancer symptoms**



```
[0.38 0.25 0.37 0.  ]
bulimia
```

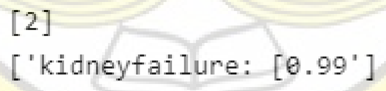
Figure 5.5 **Prediction of random forest for oral cancer symptoms**



```
['oralcancer']0.8742341
```

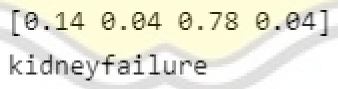
Figure 5.6 **Prediction of TensorFlow's multilayer perceptrons for oral cancer symptoms**

Referring to Figure 5.4-5.6, has similar result with the previous prediction, the accuracy of the prediction confidence between XGBoost is the highest with prediction confidence with 98%, TensorFlow's multilayer perceptrons confidence at about 87%, saying that the symptoms given are symptoms from oral cancer. In contrast, the random forest algorithm misclassified the symptoms and predicted that they came from bulimia. Afterward, the test is done with kidney failure symptoms 1, 0, 1, 0, 7, 3 seen in Figure 5.7-5.9.



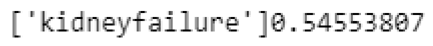
```
[2]
['kidneyfailure: [0.99']
```

Figure 5.7 **Prediction of XGBoost for kidney failure symptoms**



```
[0.14 0.04 0.78 0.04]
kidneyfailure
```

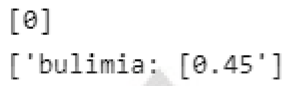
Figure 5.8 **Prediction of random forest for kidney failure symptoms**



```
['kidneyfailure']0.54553807
```

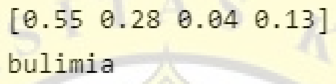
Figure 5.9 **Prediction of TensorFlow's multilayer perceptrons for kidney failure symptoms**

Referring to Figure 5.7-5.9, has similar result with the previous prediction, the accuracy of the prediction confidence between XGBoost is the highest with prediction confidence with 99%, and surprisingly, the random forest came with second highest prediction confidence with 78%, and TensorFlow's multilayer perceptrons prediction confidence is the lowest at 54%. Afterward, the test is done with bulimia symptoms 1, 0, 4, 0, 3, 1 seen in Figure 5.10-5.12.



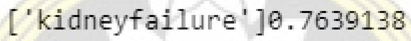
```
[0]
['bulimia: [0.45']
```

Figure 5.10 **Prediction of XGBoost for bulimia symptoms**



```
[0.55 0.28 0.04 0.13]
bulimia
```

Figure 5.11 **Prediction of random forest for bulimia symptoms**

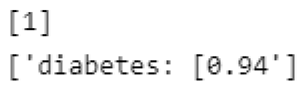


```
['kidneyfailure']0.7639138
```

Figure 5.12 **Prediction of random forest for kidney failure symptoms**

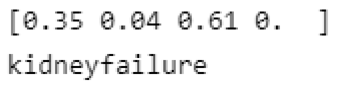
Referring to Figure 5.10-5.12, is also interesting, the accuracy of the prediction confidence XGBoost is only the second highest with prediction confidence with 45%, and surprisingly, the random forest came with the highest prediction confidence with 55%. Tensorflow's multilayer perceptrons prediction misclassified the illness as kidney failure with confidence of 76%. The conclusion that can be gathered is that XGBoost is consistently getting the result of the symptoms correctly, while most of the time having the highest confidence as well, TensorFlow's multilayer perceptrons came second as the prediction confidence at times is not as high as XGBoost, and misclassification happens towards certain illness' symptoms. And in the last position in terms of performance came random forest, with the lowest confidence overall, and getting certain illness' symptoms misclassified as well.

The next test is done with false inputs with two false symptoms inputted to see what the algorithms will predict compared to the full dataset. The test is using input 1, 1, 1, 1, 1, 1, then 10, 10, 10, 10, 10, 10, and the result is as seen in Figure 5.13-5.18.



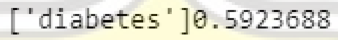
```
[1]
['diabetes: [0.94']
```

Figure 5.13 **Prediction of XGBoost with input 1, 1, 1, 1, 1, 1**



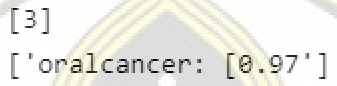
```
[0.35 0.04 0.61 0.  ]
kidneyfailure
```

Figure 5.14 **Prediction of random forest with input 1, 1, 1, 1, 1, 1**



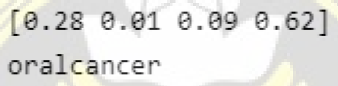
```
['diabetes']0.5923688
```

Figure 5.15 **Prediction of TensorFlow's multilayer perceptrons with input 1, 1, 1, 1, 1, 1**



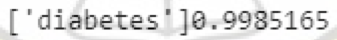
```
[3]
['oralcancer: [0.97']
```

Figure 5.16 **Prediction of XGBoost with input 10, 10, 10, 10, 10, 10**



```
[0.28 0.01 0.09 0.62]
oralcancer
```

Figure 5.17 **Prediction of random forest with input 10, 10, 10, 10, 10, 10**

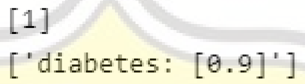


```
['diabetes']0.9985165
```

Figure 5.18 **Prediction of TensorFlow's multilayer perceptrons with input 10, 10, 10, 10, 10, 10**

Referring to Figure 5.13-5.18, the false inputs' result shows that XGBoost shows similarity to both of the algorithms in different input scenarios. XGBoost's prediction is the same when trying to classify input 1, 1, 1, 1, 1, 1 with the prediction confidence at 97% and TensorFlow multilayer perceptrons' prediction confidence at 59% with both classifying 1, 1, 1, 1, 1, 1 as diabetes. Random forest classified those symptoms as kidney failure with a prediction confidence of 61%. While with input 10, 10, 10, 10, 10, 10, XGBoost classification is similar to the random forest as both classified the symptoms as oral cancer, with XGBoost prediction confidence at 97% and random forest at 62%. TensorFlow's multilayer perceptrons, on the other hand, classified the symptoms as diabetes with a prediction confidence of 99%.

The next test is to test when only entries that are more agreeable by annotators (agreed on four or more symptoms of an illness entry) meaning there will be 77 entries, to see the confusion matrix result, and its accuracy. The test input will be from diabetes 3, 1, 6, 3, 0, 7. Figure 5.19, 5.21, and 5.23 shows the prediction confidence, Figure 5.20, 5.22, and 5.24 shows the confusion matrix.



```
[1]
['diabetes: [0.9]']
```

Figure 5.19 **Prediction confidence of XGBoost with data consisting of only more agreeable entries**

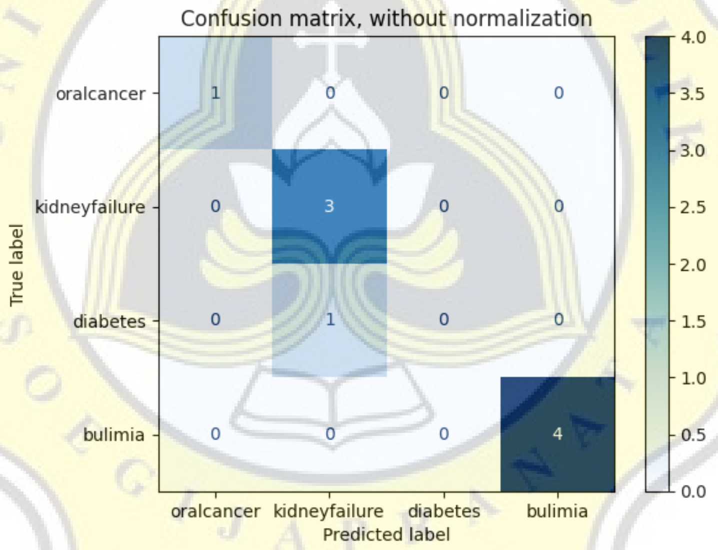


Figure 5.20 **Confusion matrix of XGBoost with data consisting of only more agreeable entries**

$$Accuracy = \frac{8}{9} = 0.89$$

$$Precision\ (Oral\ Cancer) = \frac{1}{1} = 1$$

$$Precision\ (Kidney\ Failure) = \frac{3}{4} = 0.75$$

$$Precision\ (Diabetes) = 0$$

$$Precision\ (Bulimia) = \frac{1}{1} = 1$$

$$Recall\ (Oral\ Cancer) = \frac{1}{1} = 1$$

$$Recall\ (Kidney\ Failure) = \frac{3}{3} = 1$$

$$Recall\ (Diabetes) = \frac{0}{1} = 0$$

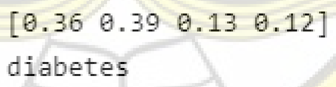
$$Recall\ (Bulimia) = \frac{4}{4} = 1$$

$$F1 - Score\ (Oral\ Cancer) = \frac{2 \times (1 \times 1)}{1 + 1} = 1$$

$$F1 - Score\ (Kidney\ Failure) = \frac{2 \times (0.75 \times 1)}{0.75 + 1} = 0.86$$

$$F1 - Score\ (Diabetes) = 0$$

$$F1 - Score\ (Bulimia) = \frac{2 \times (1 \times 1)}{1 + 1} = 1$$



```
[0.36 0.39 0.13 0.12]
diabetes
```

Figure 5.21 **Prediction confidence of random forest with data consisting of only more agreeable entries**

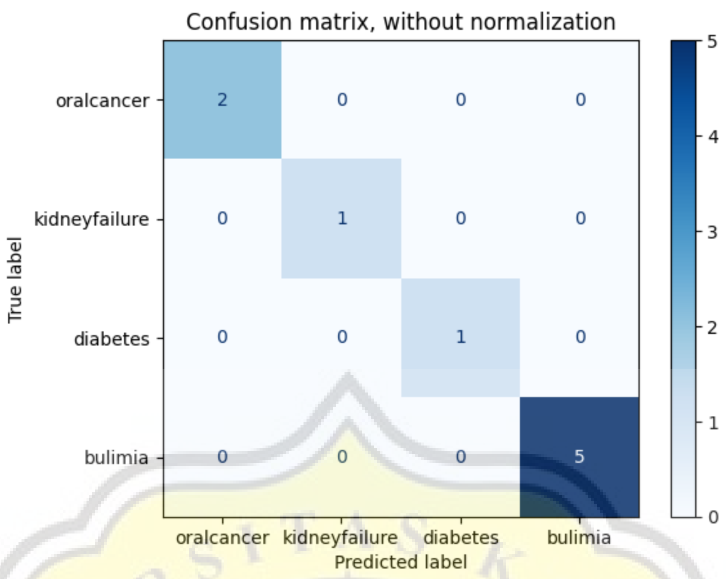


Figure 5.22 **Confusion matrix of random forest with data consisting of only more agreeable entries**

$$Accuracy = \frac{9}{9} = 1$$

$$Precision\ (Oral\ Cancer) = \frac{1}{1} = 1$$

$$Precision\ (Kidney\ Failure) = \frac{1}{1} = 1$$

$$Precision\ (Diabetes) = \frac{1}{1} = 1$$

$$Precision\ (Bulimia) = \frac{5}{5} = 1$$

$$Recall\ (Oral\ Cancer) = \frac{1}{1} = 1$$

$$Recall\ (Kidney\ Failure) = \frac{1}{1} = 1$$

$$Recall\ (Diabetes) = \frac{1}{1} = 1$$

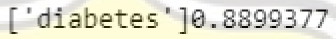
$$Recall\ (Bulimia) = \frac{5}{5} = 1$$

$$F1 - Score\ (Oral\ Cancer) = \frac{2 \times (1 \times 1)}{1 + 1} = 1$$

$$F1 - Score\ (Kidney\ Failure) = \frac{2 \times (1 \times 1)}{1 + 1} = 1$$

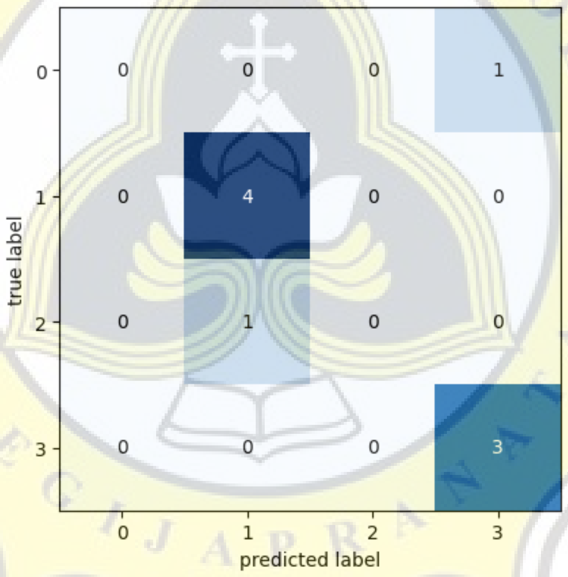
$$F1 - Score\ (Diabetes) = \frac{2 \times (1 \times 1)}{1 + 1} = 1$$

$$F1 - Score\ (Bulimia) = \frac{2 \times (1 \times 1)}{1 + 1} = 1$$



```
['diabetes']0.8899377
```

Figure 5.23 **Prediction confidence of TensorFlow's multilayer perceptrons with data consisting of only more agreeable entries**



	0	1	2	3
0	0	0	0	1
1	0	4	0	0
2	0	1	0	0
3	0	0	0	3

Figure 5.24 **Confusion matrix of TensorFlow's multilayer perceptrons with data consisting of only more agreeable entries**

$$Accuracy = \frac{7}{9} = 0.78$$

$$Precision\ (Oral\ Cancer) = 0$$

$$Precision\ (Kidney\ Failure) = \frac{4}{5} = 0.8$$

$$Precision\ (Diabetes) = 0$$

$$Precision\ (Bulimia) = \frac{3}{4} = 0.75$$

$$Recall\ (Oral\ Cancer) = 0$$

$$Recall\ (Kidney\ Failure) = \frac{4}{4} = 1$$

$$Recall\ (Diabetes) = 0$$

$$Recall\ (Bulimia) = \frac{3}{3} = 1$$

$$F1 - Score\ (Oral\ Cancer) = 0$$

$$F1 - Score\ (Kidney\ Failure) = \frac{2 \times (0.8 \times 1)}{0.8 + 1} = 0.89$$

$$F1 - Score\ (Diabetes) = 0$$

$$F1 - Score\ (Bulimia) = \frac{2 \times (0.83 \times 1)}{0.83 + 1} = 0.86$$

From the observation with data consisting of only more agreeable entries , it can be seen on Figure 5.19, 5.21, and 5.23 that all algorithms have their accuracy decreased with a smaller, but more accurate dataset with XGBoost having prediction confidence of 90%, random forest at 39%, and TensorFlow's multilayer perceptron at 89%. XGBoost in this scenario is second best after the random forest, with an accuracy of 0.89, compared to the random forest's 1. And the worst performing in this scenario is TensorFlow's multilayer perceptrons with an accuracy of 0.78. Interestingly, the dataset became more imbalanced towards bulimia. Referring to 5.20, 5.22, and 5.24, the gradient has a darker tone towards bulimia, and it can be seen that bulimia has more data as well in the confusion matrix.

The next test is with a 50:50 split of training and testing to also see what the algorithms will predict, the confusion matrix result, and its accuracy in Figure 5.25-5.30.

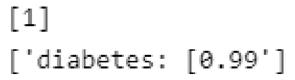

```
[1]
['diabetes: [0.99']
```

Figure 5.25 **Prediction of XGBoost for diabetes symptoms**

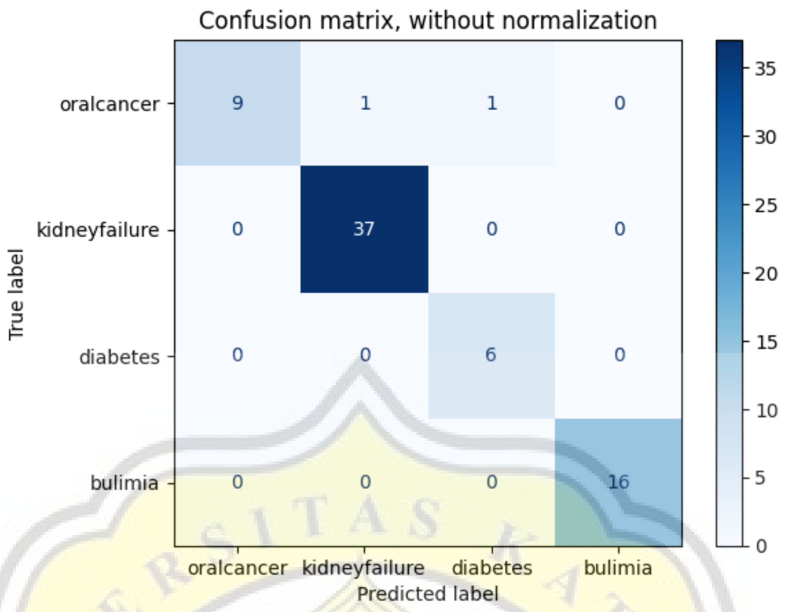


Figure 5.26 **Confusion matrix of XGBoost for diabetes symptoms with 50:50 training and test split**

$$Accuracy = \frac{68}{70} = 0.97$$

$$Precision\ (Oral\ Cancer) = \frac{9}{9} = 1$$

$$Precision\ (Kidney\ Failure) = \frac{37}{38} = 0.97$$

$$Precision\ (Diabetes) = \frac{6}{7} = 0.86$$

$$Precision\ (Bulimia) = \frac{16}{16} = 1$$

$$Recall\ (Oral\ Cancer) = \frac{9}{11} = 0.82$$

$$Recall\ (Kidney\ Failure) = \frac{37}{37} = 1$$

$$Recall\ (Diabetes) = \frac{6}{6} = 1$$

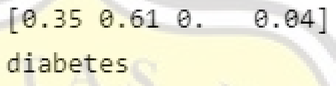
$$Recall\ (Bulimia) = \frac{16}{16} = 1$$

$$F1 - Score\ (Oral\ Cancer) = \frac{2 \times (1 \times 0.90)}{1 + 0.90} = 0.90$$

$$F1 - Score\ (Kidney\ Failure) = \frac{2 \times (0.97 \times 1)}{0.97 + 1} = 0.98$$

$$F1 - Score\ (Diabetes) = \frac{2 \times (1 \times 1)}{1 + 1} = 0.92$$

$$F1 - Score\ (Bulimia) = \frac{2 \times (1 \times 1)}{1 + 1} = 1$$



```
[0.35 0.61 0.   0.04]
diabetes
```

Figure 5.27 **Prediction of random forest for diabetes symptoms 50:50 training and test split**

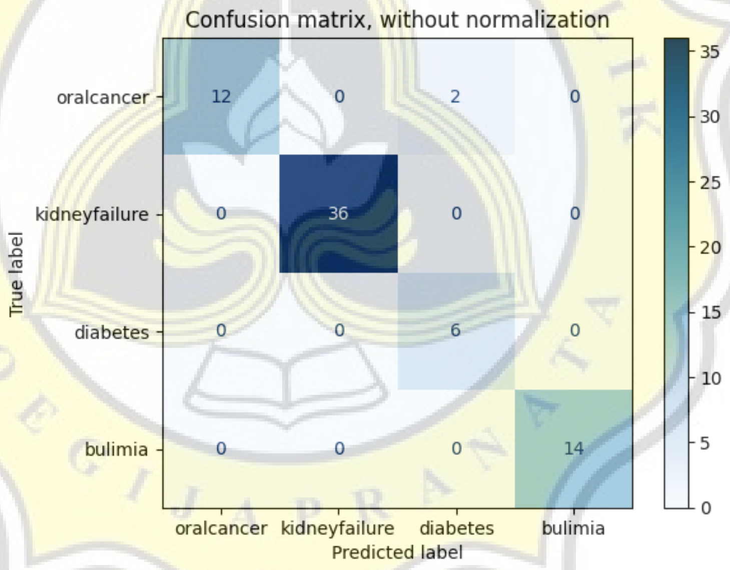


Figure 5.28 **Confusion matrix of random forest for diabetes symptoms 50:50 training and test split**

$$Accuracy = \frac{68}{70} = 0.97$$

$$Precision\ (Oral\ Cancer) = \frac{14}{14} = 1$$

$$Precision\ (Kidney\ Failure) = \frac{36}{36} = 1$$

$$Precision\ (Diabetes) = \frac{6}{8} = 0.75$$

$$Precision\ (Bulimia) = \frac{14}{14} = 1$$

$$Recall\ (Oral\ Cancer) = \frac{12}{14} = 0.86$$

$$Recall\ (Kidney\ Failure) = \frac{36}{36} = 1$$

$$Recall\ (Diabetes) = \frac{6}{6} = 1$$

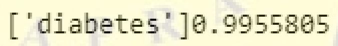
$$Recall\ (Bulimia) = \frac{14}{14} = 1$$

$$F1 - Score\ (Oral\ Cancer) = \frac{2 \times (1 \times 0.86)}{1 + 0.86} = 0.92$$

$$F1 - Score\ (Kidney\ Failure) = \frac{2 \times (1 \times 1)}{1 + 1} = 1$$

$$F1 - Score\ (Diabetes) = \frac{2 \times (1 \times 0.75)}{1 + 0.75} = 0.86$$

$$F1 - Score\ (Bulimia) = \frac{2 \times (1 \times 1)}{1 + 1} = 1$$



```
['diabetes']0.9955805
```

Figure 5.29 **Prediction of TensorFlow's multilayer perceptrons for diabetes symptoms 50:50 training and test split**

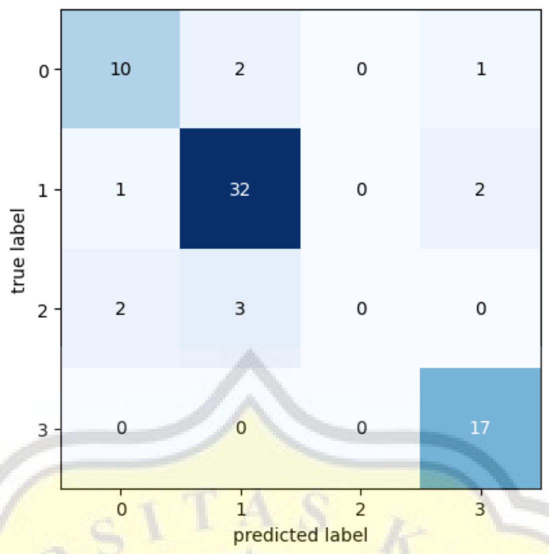


Figure 5.30 **Confusion matrix of TensorFlow's multilayer perceptrons for diabetes symptoms 50:50 training and test split**

$$Accuracy = \frac{59}{70} = 0.84$$

$$Precision\ (Oral\ Cancer) = \frac{10}{13} = 0.77$$

$$Precision\ (Kidney\ Failure) = \frac{32}{37} = 0.86$$

$$Precision\ (Diabetes) = 0$$

$$Precision\ (Bulimia) = \frac{17}{20} = 0.85$$

$$Recall\ (Oral\ Cancer) = \frac{10}{13} = 0.77$$

$$Recall\ (Kidney\ Failure) = \frac{32}{35} = 0.91$$

$$Recall\ (Diabetes) = 0$$

$$Recall\ (Bulimia) = \frac{17}{17} = 1$$

$$F1 - Score\ (Oral\ Cancer) = \frac{2 \times (0.77 \times 0.77)}{0.77 + 0.77} = 0.77$$

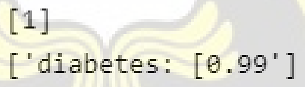
$$F1 - Score\ (Kidney\ Failure) = \frac{2 \times (0.86 \times 0.91)}{0.86 + 0.91} = 0.88$$

$$F1 - Score\ (Diabetes) = 0$$

$$F1 - Score\ (Bulimia) = \frac{2 \times (0.85 \times 1)}{0.85 + 1} = 0.92$$

From the observation for 50:50 training test split, it can be seen on Figure 5.25, 5.27, and 5.29 that all algorithms have similar result with 80:20 training test split, with XGBoost having prediction confidence of 99%, random forest at 61%, and TensorFlow's multilayer perceptron at 99%. XGBoost and random forest in this scenario have the best result, with an accuracy of 0.97. And the worst performing in this scenario is TensorFlow's multilayer perceptrons with an accuracy of 0.84. Interestingly, the dataset became more imbalanced towards bulimia. Referring to 5.26, 5.28, and 5.30, the result of the confusion matrix is also similar with 80:20 split with the gradient having a darker tone towards kidney failure, indicating data imbalance.

The next test is with a 70:30 split of training and testing to also see what the algorithms will predict, the confusion matrix result, and its accuracy in Figure 5.31-5.36.



```
[1]
['diabetes: [0.99']
```

Figure 5.31 **Prediction of XGBoost for diabetes symptoms 70:30 training and test split**

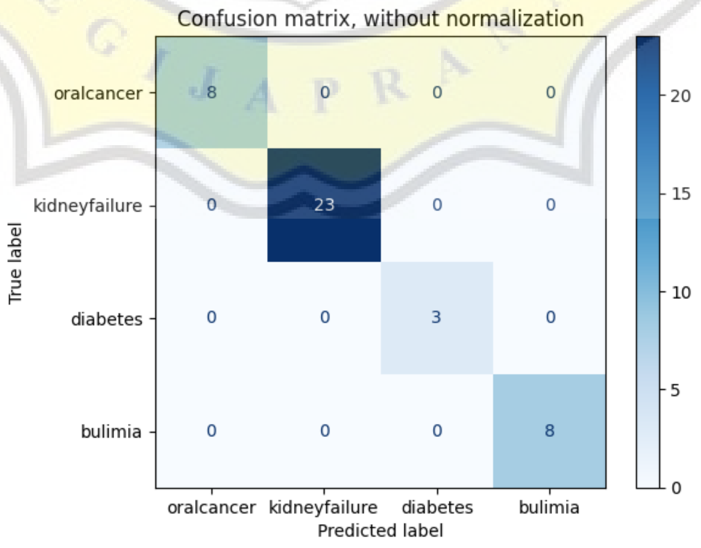


Figure 5.32 **Confusion matrix of XGBoost for diabetes symptoms 70:30 training and test split**

$$Accuracy = \frac{42}{42} = 1$$

$$Precision\ (Oral\ Cancer) = \frac{8}{8} = 1$$

$$Precision\ (Kidney\ Failure) = \frac{23}{23} = 1$$

$$Precision\ (Diabetes) = \frac{3}{3} = 1$$

$$Precision\ (Bulimia) = \frac{8}{8} = 1$$

$$Recall\ (Oral\ Cancer) = \frac{8}{8} = 1$$

$$Recall\ (Kidney\ Failure) = \frac{23}{23} = 1$$

$$Recall\ (Diabetes) = \frac{3}{3} = 1$$

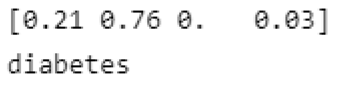
$$Recall\ (Bulimia) = \frac{8}{8} = 1$$

$$F1 - Score\ (Oral\ Cancer) = \frac{2 \times (1 \times 1)}{1 + 1} = 1$$

$$F1 - Score\ (Kidney\ Failure) = \frac{2 \times (1 \times 1)}{1 + 1} = 1$$

$$F1 - Score\ (Diabetes) = \frac{2 \times (1 \times 1)}{1 + 1} = 1$$

$$F1 - Score\ (Bulimia) = \frac{2 \times (1 \times 1)}{1 + 1} = 1$$



```
[0.21 0.76 0.   0.03]
diabetes
```

Figure 5.33 **Prediction of random forest for diabetes symptoms 70:30 training and test split**

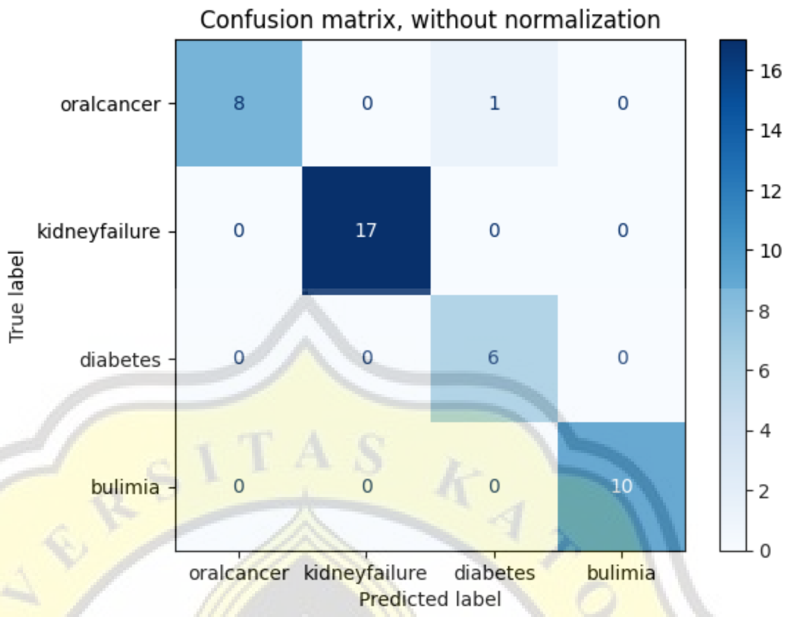


Figure 5.34 **Prediction of random forest for diabetes symptoms 70:30 training and test split**

$$Accuracy = \frac{41}{42} = 0.98$$

$$Precision\ (Oral\ Cancer) = \frac{8}{8} = 1$$

$$Precision\ (Kidney\ Failure) = \frac{17}{17} = 1$$

$$Precision\ (Diabetes) = \frac{6}{7} = 0.86$$

$$Precision\ (Bulimia) = \frac{10}{10} = 1$$

$$Recall\ (Oral\ Cancer) = \frac{8}{9} = 0.89$$

$$Recall\ (Kidney\ Failure) = \frac{17}{17} = 1$$

$$Recall\ (Diabetes) = \frac{6}{6} = 1$$

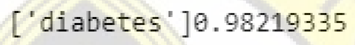
$$Recall\ (Bulimia) = \frac{10}{10} = 1$$

$$F1 - Score\ (Oral\ Cancer) = \frac{2 \times (1 \times 0.89)}{1 + 0.89} = 0.94$$

$$F1 - Score\ (Kidney\ Failure) = \frac{2 \times (1 \times 1)}{1 + 1} = 1$$

$$F1 - Score\ (Diabetes) = \frac{2 \times (0.86 \times 1)}{0.86 + 1} = 1$$

$$F1 - Score\ (Bulimia) = \frac{2 \times (1 \times 1)}{1 + 1} = 0.92$$



```
['diabetes']0.98219335
```

Figure 5.35 **Prediction of TensorFlow's multilayer perceptrons for diabetes symptoms 70:30 training and test split**

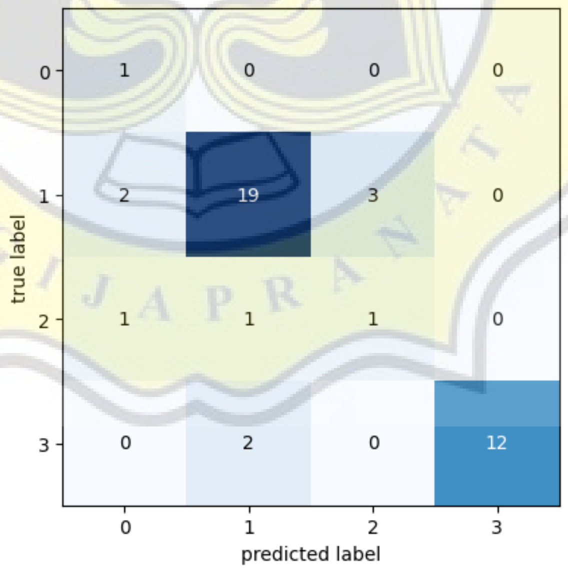


Figure 5.36 **Confusion matrix of TensorFlow's multilayer perceptrons symptoms 70:30 training and test split**

$$Accuracy = \frac{33}{42} = 0.79$$

$$Precision\ (Oral\ Cancer) = \frac{1}{4} = 0.25$$

$$Precision\ (Kidney\ Failure) = \frac{19}{22} = 0.86$$

$$Precision\ (Diabetes) = \frac{1}{4} = 0.25$$

$$Precision\ (Bulimia) = \frac{12}{12} = 1$$

$$Recall\ (Oral\ Cancer) = \frac{1}{1} = 1$$

$$Recall\ (Kidney\ Failure) = \frac{19}{24} = 0.79$$

$$Recall\ (Diabetes) = \frac{1}{3} = 0.33$$

$$Recall\ (Bulimia) = \frac{12}{14} = 0.86$$

$$F1 - Score\ (Oral\ Cancer) = \frac{2 \times (0.25 \times 1)}{0.25 + 1} = 0.40$$

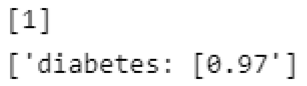
$$F1 - Score\ (Kidney\ Failure) = \frac{2 \times (0.86 \times 0.79)}{0.86 + 0.79} = 0.82$$

$$F1 - Score\ (Diabetes) = \frac{2 \times (0.25 \times 0.33)}{0.25 + 0.33} = 0.28$$

$$F1 - Score\ (Bulimia) = \frac{2 \times (1 \times 0.86)}{1 + 0.86} = 0.92$$

From the observation for 70:30 training test split, it can be seen on Figure 5.31, 5.33, and 5.35 that all algorithms have somewhat similar result with 80:20 and 50:50 training test split, main difference being random having way better accuracy, with XGBoost having prediction confidence of 99%, random forest at 76%, and TensorFlow’s multilayer perceptron at 98%. XGBoost have the best accuracy with 1, random forest in this scenario achieved the accuracy of 0.98. And the worst performing in this scenario is TensorFlow’s multilayer perceptrons with an accuracy of 0.76.

The next test is with a 20:80 split of training and testing to also see what the algorithms will predict, the confusion matrix result, and its accuracy in Figure 5.37-5.42.



```
[1]
['diabetes: [0.97']
```

Figure 5.37 **Prediction of XGBoost for diabetes symptoms 20:80 training and test split**

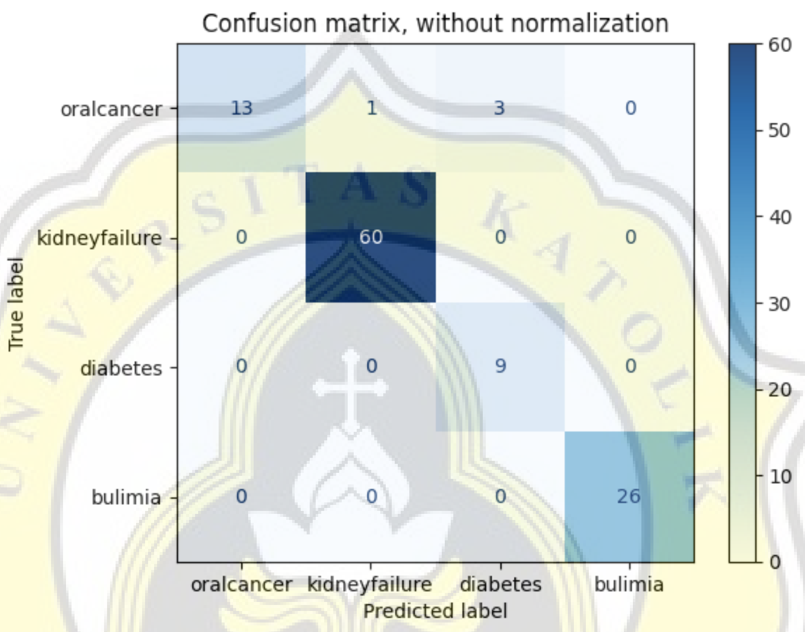


Figure 5.38 **Confusion matrix of XGBoost for diabetes symptoms 20:80 training and test split**

$$Accuracy = \frac{108}{112} = 0.96$$

$$Precision\ (Oral\ Cancer) = \frac{13}{13} = 1$$

$$Precision\ (Kidney\ Failure) = \frac{60}{61} = 0.98$$

$$Precision\ (Diabetes) = \frac{9}{12} = 0.75$$

$$Precision\ (Bulimia) = \frac{26}{26} = 1$$

$$Recall\ (Oral\ Cancer) = \frac{13}{17} = 0.76$$

$$Recall\ (Kidney\ Failure) = \frac{60}{60} = 1$$

$$Recall\ (Diabetes) = \frac{9}{9} = 1$$

$$Recall\ (Bulimia) = \frac{26}{26} = 1$$

$$F1 - Score\ (Oral\ Cancer) = \frac{2 \times (1 \times 0.76)}{1 + 0.76} = 0.86$$

$$F1 - Score\ (Kidney\ Failure) = \frac{2 \times (0.98 \times 1)}{0.98 + 1} = 0.99$$

$$F1 - Score\ (Diabetes) = \frac{2 \times (0.75 \times 1)}{0.75 + 1} = 0.86$$

$$F1 - Score\ (Bulimia) = \frac{2 \times (1 \times 1)}{1 + 1} = 1$$

```
[0.31 0.53 0.04 0.12]
diabetes
```

Figure 5.39 **Prediction of random forest for diabetes symptoms 20:80 training and test split**

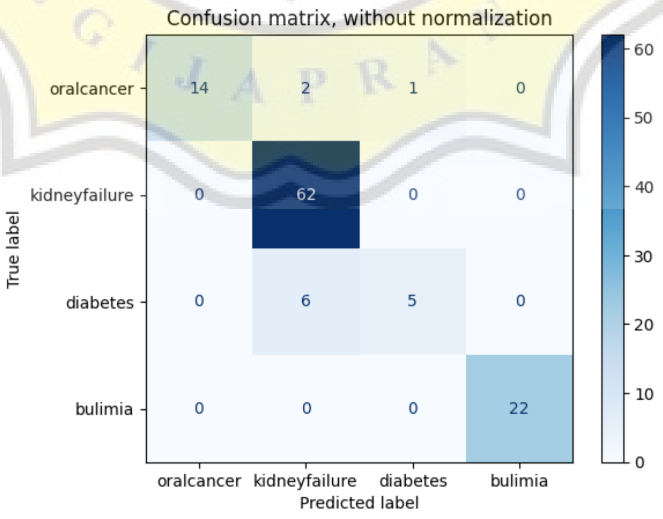


Figure 5.40 **Prediction of random forest for diabetes symptoms 20:80 training and test split**

$$Accuracy = \frac{103}{112} = 0.92$$

$$Precision\ (Oral\ Cancer) = \frac{14}{14} = 1$$

$$Precision\ (Kidney\ Failure) = \frac{62}{70} = 0.89$$

$$Precision\ (Diabetes) = \frac{5}{6} = 0.83$$

$$Precision\ (Bulimia) = \frac{22}{22} = 1$$

$$Recall\ (Oral\ Cancer) = \frac{14}{17} = 0.82$$

$$Recall\ (Kidney\ Failure) = \frac{62}{62} = 1$$

$$Recall\ (Diabetes) = \frac{5}{11} = 0.45$$

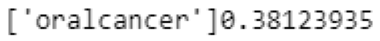
$$Recall\ (Bulimia) = \frac{22}{22} = 1$$

$$F1 - Score\ (Oral\ Cancer) = \frac{2 \times (1 \times 0.82)}{1 + 0.82} = 0.90$$

$$F1 - Score\ (Kidney\ Failure) = \frac{2 \times (0.89 \times 1)}{0.89 + 1} = 0.94$$

$$F1 - Score\ (Diabetes) = \frac{2 \times (0.83 \times 0.45)}{0.83 + 0.45} = 0.58$$

$$F1 - Score\ (Bulimia) = \frac{2 \times (1 \times 1)}{1 + 1} = 1$$



```
['oralcancer']0.38123935
```

Figure 5.41 **Prediction of TensorFlow multilayer perceptrons for diabetes symptoms 20:80 training and test split**

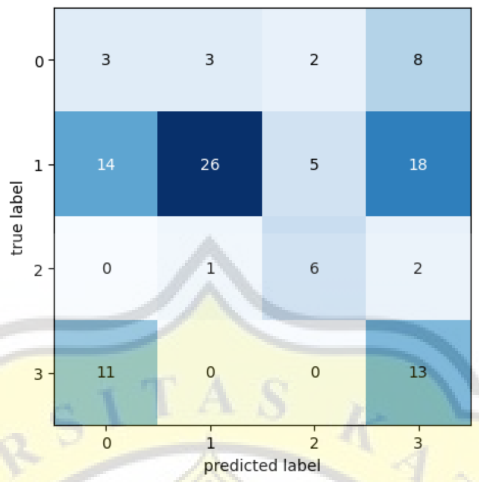


Figure 5.42 **Prediction of TensorFlow multilayer perceptrons for diabetes symptoms 20:80 training and test split**

$$Accuracy = \frac{38}{112} = 0.33$$

$$Precision\ (Oral\ Cancer) = \frac{3}{28} = 0.11$$

$$Precision\ (Kidney\ Failure) = \frac{26}{30} = 0.87$$

$$Precision\ (Diabetes) = \frac{6}{13} = 0.46$$

$$Precision\ (Bulimia) = \frac{13}{41} = 0.32$$

$$Recall\ (Oral\ Cancer) = \frac{3}{16} = 0.19$$

$$Recall\ (Kidney\ Failure) = \frac{26}{41} = 0.41$$

$$Recall\ (Diabetes) = \frac{6}{9} = 0.67$$

$$Recall\ (Bulimia) = \frac{13}{24} = 0.54$$

$$F1 - Score\ (Oral\ Cancer) = \frac{2 \times (0.11 \times 0.19)}{0.11 + 0.19} = 0.14$$

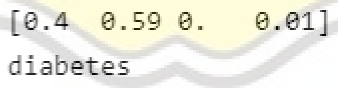
$$F1 - Score\ (Kidney\ Failure) = \frac{2 \times (0.87 \times 0.41)}{0.87 + 0.41} = 0.56$$

$$F1 - Score\ (Diabetes) = \frac{2 \times (0.46 \times 0.67)}{0.46 + 0.67} = 0.55$$

$$F1 - Score\ (Bulimia) = \frac{2 \times (0.32 \times 0.14)}{0.32 + 0.14} = 0.19$$

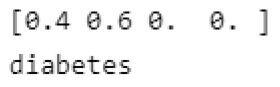
From the observation for 20:80 training test split, it is expected that the algorithms will perform less with less training data, it can be seen on Figure 5.37, 5.39, and 5.41. XGBoost is still having prediction confidence of 97%, random forest at 53%, and TensorFlow's multilayer perceptron predicted the wrong result with the result being oral cancer. XGBoost have the best accuracy with 96, random forest is the second best with the accuracy of 0.96. And the worst performing in this scenario is TensorFlow's multilayer perceptrons with an accuracy of 0.33, significantly worse than the other scenarios.

The final tests will be done for random forest and TensorFlow multilayer perceptrons with 80:20 split, but random forest having its n_estimator (number of trees) changed to several values, half of the benchmark (100 trees). And for TensorFlow multilayer perceptron will have several hidden layers added. Starting from random forest's test, the value of of how many trees for the random forest will first be changed 10, one-tenth of the original number of trees, then to 25, then to 50. Seen in Figure 5.43 is the benchmark with 100 trees. And Figure 5.44-5.46 will show the results when the number of trees is changed.



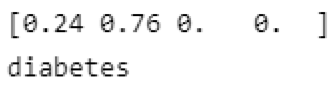
```
[0.4  0.59 0.   0.01]
diabetes
```

Figure 5.43 **Prediction of random forest for diabetes symptoms 20:80 training and test split, 100 trees**



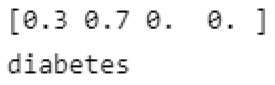
```
[0.4 0.6 0.  0. ]
diabetes
```

Figure 5.44 **Prediction of random forest for diabetes symptoms 20:80 training and test split, 10 trees**



```
[0.24 0.76 0.   0.  ]
diabetes
```

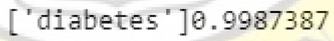
Figure 5.45 **Prediction of random forest for diabetes symptoms 20:80 training and test split, 25 trees**



```
[0.3 0.7 0.  0. ]
diabetes
```

Figure 5.46 **Prediction of random forest for diabetes symptoms 20:80 training and test split, 50 trees**

From the results of Figure 5.44-5.46 compared to 5.43, we can see that with trees numbers decreased on a couple of cases less value gives better results. Even with only 10 trees, the result is somewhat similar to 100 trees, and the best result in this scenario is found when only 25 trees are used. The explanation of this can be seen in the conclusion. The second test, which is for TensorFlow, the test will use three new hidden layers, starting from one and eventually rising to three, all three consisting of 16 nodes with ReLU activation function, dense layers. Originally the layers for TensorFlow multilayer perceptrons only consist of three layers, one input layer consisting of 10 nodes, one hidden dense layer with ReLU activation function consisting of eight nodes, and the final layer is the output layer, four nodes of softmax layer. The result of the benchmark layers is shown in Figure 5.47



```
['diabetes']0.9987387
```

Figure 5.47 **Prediction of TensorFlow multilayer perceptron with three layers**

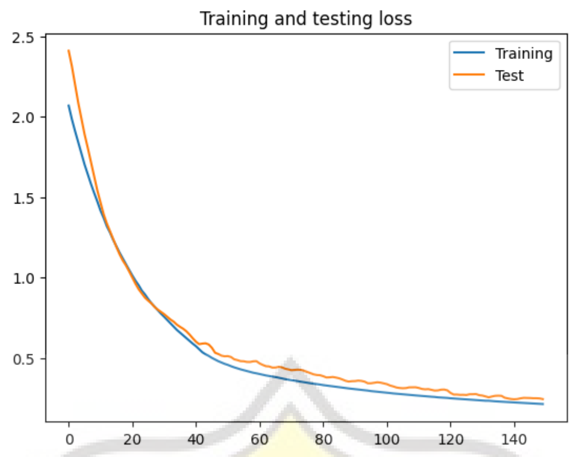


Figure 5.48 **Loss curve of TensorFlow multilayer perceptron with three layers**

```
['diabetes']0.99947137
```

Figure 5.49 **Prediction of TensorFlow multilayer perceptron with one additional dense layer consisting of 16 nodes with ReLU activation function**

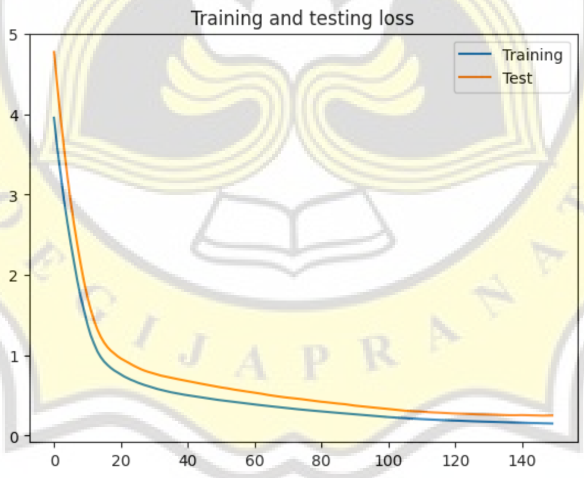


Figure 5.50 **Loss curve of TensorFlow multilayer perceptron with one additional dense layer consisting of 16 nodes with ReLU activation function**

```
['diabetes']0.9653162
```

Figure 5.51 **Prediction of TensorFlow multilayer perceptron with two additional dense layers consisting of 16 nodes with ReLU activation function**

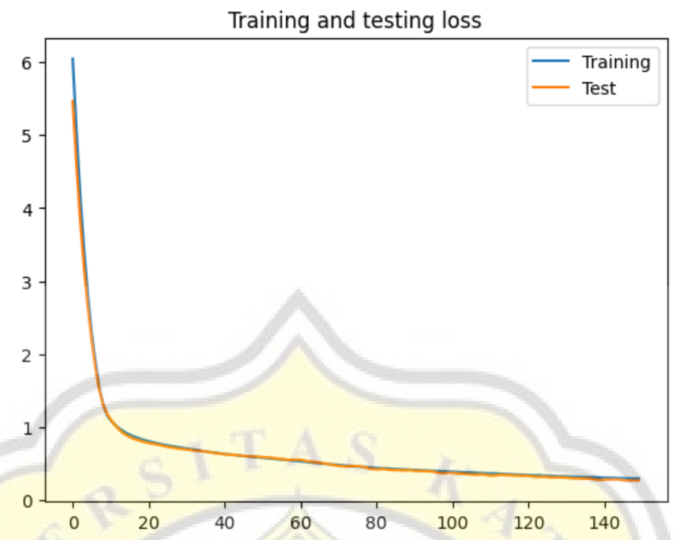


Figure 5.52 **Loss curve of TensorFlow multilayer perceptron with two additional dense layers consisting of 16 nodes with ReLU activation function**

```
['diabetes']1.0
```

Figure 5.53 **Prediction of TensorFlow multilayer perceptron with three additional dense layers consisting of 16 nodes with ReLU activation function**

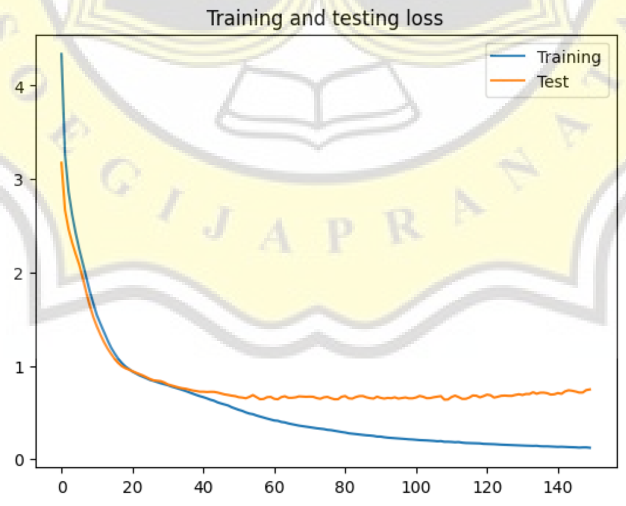
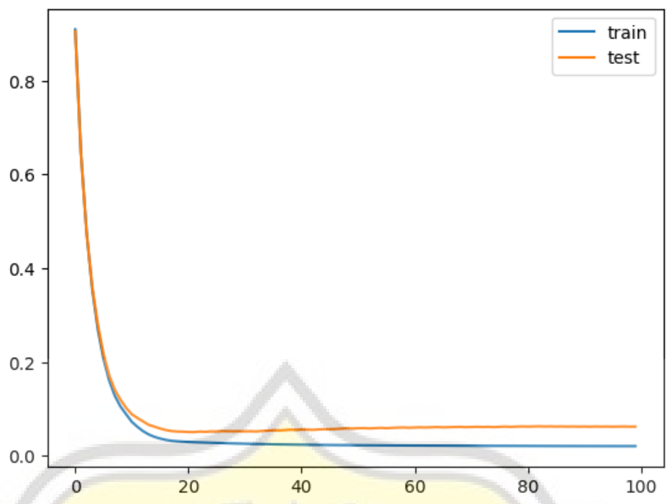


Figure 5.54 **Loss curve of TensorFlow multilayer perceptron with three additional dense layers consisting of 16 nodes with ReLU activation function**

From the hidden layer tests above, we can see that in most cases, in Figure 5.48, 5.50, and 5.52, the loss curve shows good result as training and test set final result are close to one another. Interestingly when four hidden layers are used in the scenario, the overfitting occurs, as the test loss curve deviate with quite a margin from the training set result, as seen in Figure 5.54. Even when overfitting occurs, the run manages to get similar results above 0.95 confidence score.

Finally, the overall conclusion of the observed results will be explained. The dataset that is used for this project's scenario is quite small, therefore it will be interesting to see which algorithm handles this sort of problem better. In this scenario, the best performing algorithm is the main algorithm of this project XGBoost consistently has the highest accuracy in its confusion matrix, and also consistent correct prediction with almost all of the scenarios given to it compared to the comparative algorithms. The second TensorFlow's multilayer perceptron, which can be surprising because the random forest's average accuracy gathered from the confusion matrix shows that random forest consistently has better accuracy than multilayer perceptron, but when using the algorithm for actual prediction, the random forest gets more results wrong than TensorFlow's multilayer perceptrons. Even when both XGBoost and TensorFlow multilayer perceptrons both got good results, as shown in Figure 5.55 and Figure 5.56. There is actually a reason why random forest perform worse than both of the algorithms and even a reason for why random forest at times shows better prediction when having less trees. It is because each tree is grown to the largest extent possible and there is no pruning. The generalization error variance is decreasing to zero in the Random Forest when more trees are added to the algorithm. However, the bias of the generalization does not change. This can happen for example if the dataset is relatively simple, and therefore the fully-grown trees perfectly learn its patterns and predict very similarly. And because of the dataset is small, and there are similar symptoms, this problem occurs for this project.



S

Figure 5.55 **Loss curve of XGBoost**

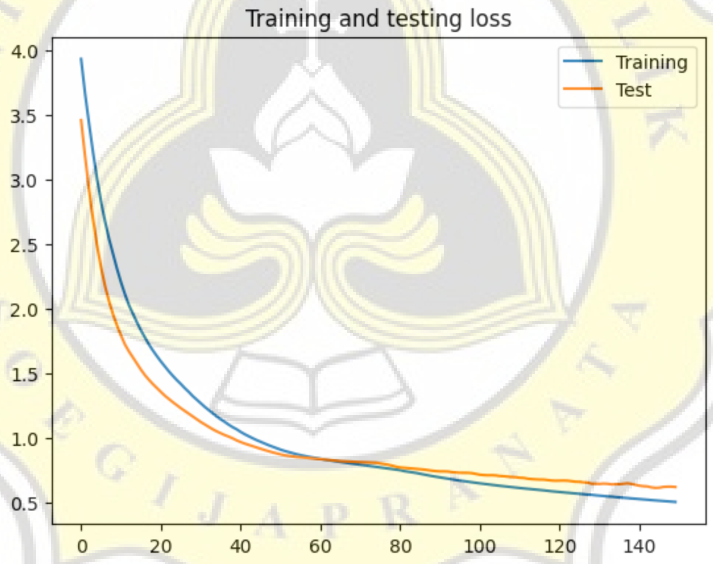


Figure 5.56 **Loss curve of TensorFlow multilayer perceptrons**