

## CHAPTER 5

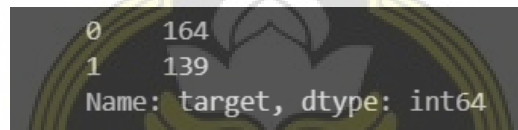
### IMPLEMENTATION AND RESULTS

#### 5.1. Implementation

The implementations of this research will be explained in this sub-chapter. The original dataset has a target value of 0, 1, 2, 3, and 4. The value with 1, 2, 3, and 4 that represent as the presence of heart disease will be replaced with 1. So, the target value will only have 0 as False and 1 as True. The process of replacing the target value can be seen below

```
1. df['target'] = df['target'].replace({
2.     2 : 1,
3.     3 : 1,
4.     4 : 1,
5. })
```

In line 1 code above, the data column “target” call replace function to replace the chosen value with the target value. The value 2-4 will be replaced with value 1 in the line 2-4 code above. The result of the dataset “target” column using this code can be seen in Figure 5.1 below.



```
0    164
1    139
Name: target, dtype: int64
```

**Figure 5.1** Heart Disease target column

Next, each of the handling missing values will be analyzed to see which handling missing values have better gained performance on the model. The first one is dropping missing data rows can be done using the code below.

```
1. check = df.isin(['?'])
2. selectedRow = df[check.any(axis=1)]
3. df_dropping = df.drop(selectedRow.index)
```

In line 1, the code will check if there is “?” data in the dataset rows. In line 2, getting the selected row that has “?” in the dataset and the selected row can be seen Figure 5.2 in below. After that, the code in line 3 will drop 6 of the selected row using the index and leave the dataset only to 297 rows.

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
87	53.0	0.0	3.0	128.0	216.0	0.0	2.0	115.0	0.0	0.0	1.0	0.0	?	0
166	52.0	1.0	3.0	138.0	223.0	0.0	0.0	169.0	0.0	0.0	1.0	?	3.0	0
192	43.0	1.0	4.0	132.0	247.0	1.0	2.0	143.0	1.0	0.1	2.0	?	7.0	1
266	52.0	1.0	4.0	128.0	204.0	1.0	0.0	156.0	1.0	1.0	2.0	0.0	?	1
287	58.0	1.0	2.0	125.0	220.0	0.0	0.0	144.0	0.0	0.4	2.0	?	7.0	0
302	38.0	1.0	3.0	138.0	175.0	0.0	0.0	173.0	0.0	0.0	1.0	?	3.0	0

Figure 5.2 Missing data value rows

From the previous checking, we know that only the “ca” and “thal” columns have the missing value with “?” data. So, replacing the median and the mode will only replace the “ca” and “thal” column values. The replacing with median can be seen in the code below.

```

1. df_median = df.replace({
2.     '?' : np.nan
3. })
4.
5. df_median['ca'] = df_median['ca'].fillna(df_median['ca'].median())
6. df_median['thal'] = df_median['thal'].fillna(df_median['thal'].median())

```

In line 1-3 code above, replace the “?” data with NaN (Not A Number) value and put it into the df\_median variable. Line 5 and 6 code above will replace the NaN value with the median of each column “ca” and “thal” value using the fillna function. The replacing with mode can be seen in the code below.

```

1. df_mode = df.replace({
2.     '?' : np.nan
3. })
4.
5. df_mode['ca'] = df_mode['ca'].fillna(df_mode['ca'].mode()[0])
6. df_mode['thal'] = df_mode['thal'].fillna(df_mode['thal'].mode()[0])

```

Code line 1-3 will replace the “?” data value with the NaN value and put it into the df\_mode variable. In line 5 and 6 will replace the NaN value with the mode of each column “ca” and “thal” value using the fillna function. Making the boxplot can be seen in the code below.

```

1. def make_boxplot(data, text):
2.     plt.boxplot(data,
3.         vert=True,
4.         labels=[text])
5.
6.     plt.title(text)
7.     plt.ylabel("Value")
8.     plt.grid(axis = 'y')
9.     plt.show()

```

Line 1 of the code defined make\_boxplot function with data and text as parameter above. Line 2-4 make boxplot visualisation using the data as parameter, vertical position for the boxplot, and labels using text value. Line 6 set the title of the boxplot. Line 7 set the y axis label of the boxplot with “Value”. Line 8 set y axis grid of the boxplot and line 9 show the boxplot. The function of removing outliers can be seen in code below.

```

1. def remove_outliers(feature):
2.     Q1 = np.percentile(X_outliers[feature], 25,
3.         interpolation = 'midpoint')
4.
5.     Q3 = np.percentile(X_outliers[feature], 75,
6.         interpolation = 'midpoint')
7.     IQR = Q3 - Q1
8.
9.     upper_value = (Q3+1.5*IQR)
10.    lower_value = (Q1-1.5*IQR)
11.    upper = np.where(X_outliers[feature] >= upper_value)
12.    lower = np.where(X_outliers[feature] <= lower_value)
13.
14.    X_outliers.drop(upper[0], inplace=True)
15.    X_outliers.drop(lower[0], inplace=True)
16.    X_outliers.reset_index(drop=True, inplace=True)
17.
18.    Y_outliers.drop(upper[0], inplace=True)
19.    Y_outliers.drop(lower[0], inplace=True)
20.    Y_outliers.reset_index(drop=True, inplace=True)
21.
22.    print("Upper : ", upper_value)
23.    print("Lower : ", lower_value)

```

Line 1 of the code above defined remove\_outliers function with feature as parameter. Line 2-3 declare Q1 variable using np.percentile function with data as parameter, 25 as the percentile value, and midpoint interpolation to handle if there is 2 same value at the 25 percentile value. Line 5-6 is same as line 2-3 but only it declaring Q3 using 75 as the percentile. Line 9-10 declare boundaries value of the data (upper and lower) using the percentile from Q1 and Q3 variable. Line 11-12 get the index of data where the value has less than lower value or has more than upper value. Line 14-20 drop the outliers data and reset the data index with drop

true which means deleting the current index and replace it with numeric index. Feature selection using chi square can be seen in code below.

```
1. selection = SelectKBest(score_func=chi2, k=i)
```

Line 1 code above declared selection variable and assign SelectKBest() function to get the best feature using chi2 as the score\_func and i indicate how many selected features (1-13). Feature selection using mutual information can be seen in code below.

```
1. selection = SelectKBest(score_func=mutual_info_classif, k=i)
```

Line 1 code above declared selection variable and assign SelectKBest() function to get the best feature using mutual\_info\_classif as the score\_func and i indicate how many selected features (1-13). Feature selection using anova can be seen in code below.

```
1. selection = SelectKBest(score_func=f_classif, k=i)
```

Line 1 code above declared selection variable and assign SelectKBest() function to get the best feature using f\_classif as the score\_func and i indicate how many selected features (1-13). Feature selection using forward feature selection using code below.

```
1. cv = KFold(n_splits=10, shuffle=True, random_state=seed)
2. selection = SequentialFeatureSelector(model, n_features_to_select=i,
    cv=cv)
```

Line 1 calls the 10-fold cross validation function and put it into the cv variable. Line 2 code above declared selection variable and assign SequentialFeatureSelection() function to get the best feature using XGBoost model as parameter, n\_features\_to\_select indicate how many selected features (1-12), and 10-fold cross validation. Feature selection using backward feature selection using code below.

```
1. cv = KFold(n_splits=10, shuffle=True, random_state=seed)
2. selection = SequentialFeatureSelector(model, n_features_to_select=i,
    direction="backward", cv=cv)
```

Line 1 calls the 10-fold cross validation function and put it into the cv variable. Line 2 code above declared selection variable and assign SequentialFeatureSelection() function to get the best feature using XGBoost model as parameter, n\_features\_to\_select indicate how many selected features (1-12), direction backward, and 10-fold cross validation . Feature selection using recursive feature elimination using code below.

```
1. selection = RFE(model, n_features_to_select=i)
```

Line 1 code above declared selection variable and assign RFE() function to get the best feature using XGBoost model as parameter, n\_features\_to\_select indicate how many selected features (1-13). The code to get the feature importance can be seen in code below.

```
1. thresholds = np.sort(model.feature_importances_)
```

Line 1 code above declared thresholds variable and assign the feature importance from the trained model and will be sorted lowest to highest. The thresholds variable will be used for the feature selection in the code below.

```
1. for thresh in thresholds:
2.     selection = SelectFromModel(model, threshold=thresh, prefit=True)
3.     feature_idx = selection.get_support()
4.     select_X = selection.transform(X.values)
```

In line 1 code above call the for looping function and declare thresh variable as the looping of each thresholds variable. Line 2 declared the selection variable and assign the SelectFromModel function to select the selected feature based on the thresh variable. Line 3 code above get the selected feature name. Line 4 transforms the X variable into only feature selected data. After preprocessing the data, the XGBoost will be tuned using the code below.

```
1. def xgbc(learning_rate, n_estimators, max_depth, min_child_weight,
2. gamma, subsample, colsample_bytree):
3.     model = XGBClassifier(
4.         learning_rate = learning_rate,
5.         n_estimators = int(n_estimators),
6.         max_depth = int(max_depth),
7.         min_child_weight = min_child_weight,
8.         gamma = gamma,
9.         subsample = subsample,
10.        colsample_bytree = colsample_bytree,
11.        seed = seed,
12.    )
13.
14.    cv = KFold(n_splits=10, shuffle=True, random_state=seed)
15.    accuracy = cross_val_score(model, X, Y, scoring='accuracy', cv=cv)
16.
17.    return np.mean(accuracy)
```

Line 1 of the code defined xgbc function with several parameters as shown in the code above. Line 3-12 calls the XGBClassifier function with parameters that takes input from the xgbc parameter and put it into the model variable. Line 14 calls the 10-fold cross validation function and put it into the cv variable. In line 15 will calculate the accuracy of the model with 10-fold cross validation as the data portioning. Line 17 will return the mean of the accuracy

variable as we are using 10-fold which means there will be 10 accuracy from 10 data portioning. The hyperparameter tuning using Bayesian Optimization will be done in the code below.

```

1. hyperparameter = {
2.     'learning_rate': (0.1, 1), # default 0.1
3.     'n_estimators' : (100, 250), # default 100
4.     'max_depth': (1, 15), # default 3
5.     'min_child_weight' : (0, 1), # default 1
6.     'gamma' : (0, 1), # default 0
7.     'subsample' : (0.4, 1), # default 1
8.     'colsample_bytree' : (0.4, 1), # default 1
9. }
10.
11.     xgbcBO = BayesianOptimization(
12.         f = xgbc,
13.         pbounds = hyperparameter,
14.         random_state = seed
15.     )
16.
17.     xgbcBO.maximize()

```

Line 1-9 code above declared hyperparameter variable and assign several parameters that will be tuned with the search space. Line 11-15 declares the xgbcBO variable and calls the BayesianOptimization function with the xgbc function that was explained before into the parameter with the hyperparameter variable as the parameter boundaries. Line 17 will run the xgbcBO to tune the hyperparameter.

## 5.2. Results

The result of this research will be shown in this sub-chapter. The result of handling missing values analysis can be seen in Table 5.1 below.

**Table 5.1.** Handling Missing Values Evaluation

Handling Missing Values	Evaluation				
	<i>Precision</i>	<i>Recall</i>	<i>F1-score</i>	<i>ROC Area</i>	<i>Accuracy</i>
Dropping missing value	83.36	76.32	78.82	89.37	81.55
Replacing with median	80.89	76.38	78.40	88.14	80.51
Replacing with mode	80.89	76.38	78.40	88.14	80.51

The result of dropping missing value gained 83.36% on precision, 76.32% on recall, 78.82% on recall, 89.37% on roc area, and 81.55% on accuracy. While replacing with median and replacing with mode gained the same 80.89% on precision, 76.38% on recall, 78.40% on f1-

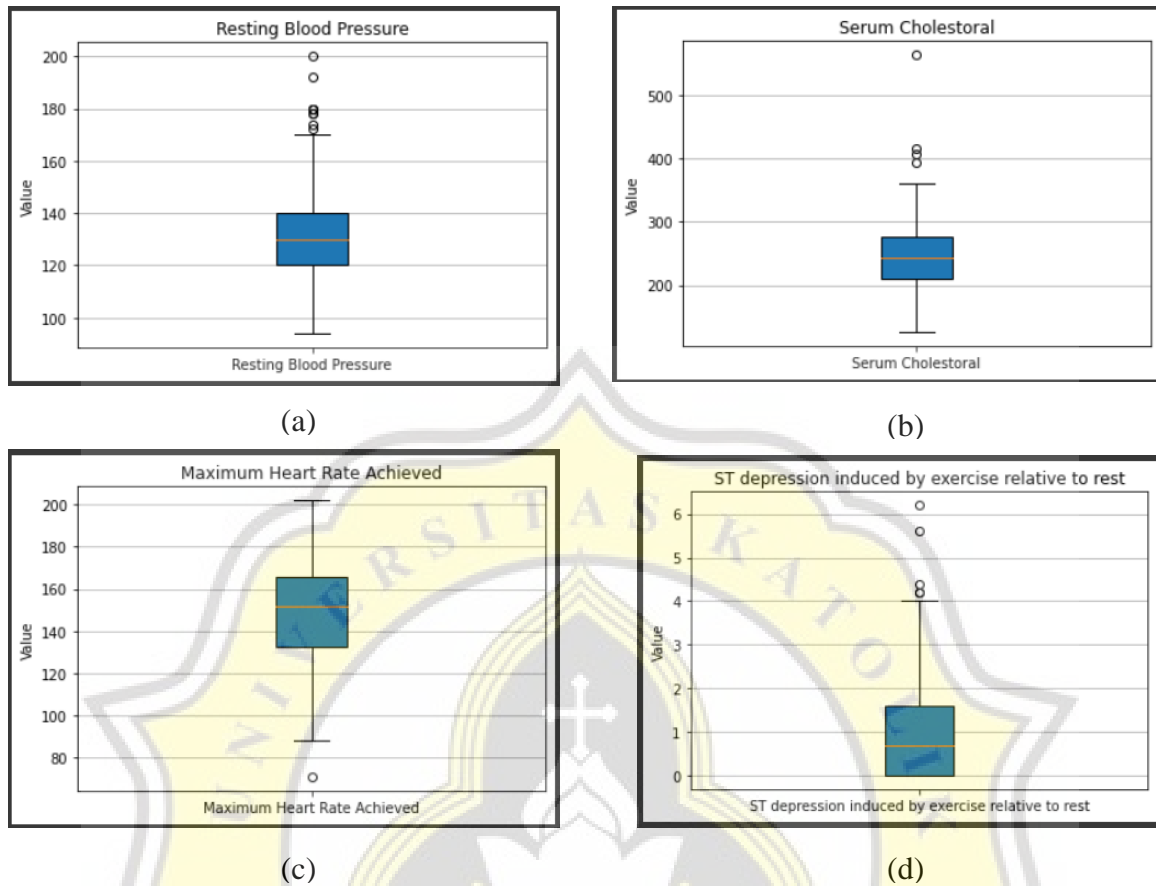


score, 88.14% on roc area, and 80.51% on accuracy. This happens because the median and the mode of the “ca” and “thal” columns have the same value as shown in **Error! Reference source not found.** below.

**Table 5.2.** Median and Mode of “ca” and “thal”

<b>Data Column</b>	<b>Math</b>	
	<i>Median</i>	<i>Mode</i>
ca	0	0
thal	3	3

Seeing the result from Table 5.1, dropping missing value have better overall performance than replacing with median and replacing with mode. Replacing with median and replacing with mode have lower precision which indicates that the model has a larger false positive value than the dropping missing value. But, replacing with median and replacing with mode has a slightly lower false negative value that makes it has better recall result. Dropping missing value have a better F1-score than both replacing with median and replacing with mode since the precision has higher results than the other two. Dropping missing value also has better roc area and accuracy that indicates this handling missing value is better than replacing with median and replacing with mode. So, the dropping missing value will be used in the preprocessing in training the model.



**Figure 5.3** (a) “trestbps” outliers, (b) “chol” outliers, (c) “thalach” outliers, and (d) “oldpeak” outliers

The outliers of data “trestbps”, “chol”, “thalach”, and “oldpeak” can be seen in Figure 5.3 above. The “trestbps” have upper bounds at 170 and lower bounds at 90. “chol” have upper bounds at 376 and lower bounds at 112. “thalach” have upper bounds at 216.25 and lower bounds at 82.25. And “oldpeak” have upper bounds at 4. Data range inside the upper and lower bounds of this numerical feature of the heart disease dataset are considered not outliers.

**Table 5.3.** Outliers Evaluation

XGBoost Model	Evaluation				
	<i>Precision</i>	<i>Recall</i>	<i>F1-score</i>	<i>ROC Area</i>	<i>Accuracy</i>
With outliers	83.36	76.32	78.82	89.37	81.55
Without outliers	79.47	73.55	75.16	87.05	79.94



Using the dropping missing value on the preprocessing, the outliers evaluation can be seen in Table 5.3 above. The result shows that removing outliers has lower performance than data with outliers. But, with the outliers has been removed, the data distribution for “trestbps”, “chol”, “thalach”, and “oldpeak” is normal and better for the model.

**Table 5.4.** Feature Selection Rankings with XGBoost

Rank	Feature Selection Techniques						
	<i>Chi Square</i>	<i>Mutual Information</i>	<i>ANOVA</i>	<i>Forward Feature Selection</i>	<i>Backward Feature Selection</i>	<i>Recursive Feature Elimination</i>	<i>Feature Importance</i>
1	thalach	thal	thal	thal	ca	thal	thal
2	ca	cp	ca	ca	oldpeak	ca	ca
3	thal	ca	oldpeak	cp	cp	cp	exang
4	oldpeak	slope	thalach	sex	fbs	oldpeak	cp
5	exang	oldpeak	cp	slope	exang	exang	sex
6	chol	exang	exang	fbs	age	sex	oldpeak
7	age	thalach	sex	exang	chol	age	thalach
8	cp	sex	slope	age	trestbps	thalach	age
9	sex	chol	age	chol	restecg	chol	chol
10	restecg	restecg	restecg	fbs	sex	fbs	restecg
11	slope	fbs	trestbps	thalach	thal	trestbps	fbs
12	trestbps	trestbps	chol	oldpeak	slope	restecg	trestbps
13	fbs	age	fbs	trestbps	thalach	slope	slope

Each feature rankings from each feature selection can be seen in Table 5.4 above. “thal”, “ca”, and “cp” has the most appearance on top 3 rankings from feature selection that are used in this research. While the “slope”, “fbs”, and “treshbps” has the most appearance on bottom 3 rankings from feature selection that are used in this research. This shows that “thal”, “ca”, and “cp” is most significant feature for XGBoost to predict target feature. While the “slope”, “fbs”, and “treshbps” is less significant feature for XGBoost to predict target feature.

**Table 5.5.** Chi Square Feature Selection with XGBoost

Feature Selection	Feature Selected	Evaluation				
		<i>Precision</i>	<i>Recall</i>	<i>F1-score</i>	<i>ROC Area</i>	<i>Accuracy</i>
Chi Square	13	82.86	78.65	79.1	90.25	83.32
	12	81.10	78.51	78.70	89.53	82.58
	11	85.65	80.22	81.94	90.52	85.63
	10	82.41	78.13	78.90	90.01	82.96
	9	80.57	76.13	76.93	91.07	81.48
	8	78.75	75.49	75.99	89.28	80.70
	7	80.78	74.08	75.58	88.27	80.33
	6	75.36	72.93	73.12	85.05	77.69
	5	75.21	72.45	71.70	85.43	76.48
	4	73.13	72.22	70.99	86.15	75.73
	3	75.16	75.74	74.22	84.93	78.42
	2	64.34	62.46	61.95	76.25	68.60
	1	60.71	51.91	54.61	64.56	64

The Chi Square feature selection result can be seen in Table 5.5 above. Top 11 feature selected from Chi Square feature selection has the best accuracy result. The top 11 “thalach”, “ca”, “thal”, “oldpeak”, “exang”, “chol”, “age”, “cp”, “sex”, “restecg”, and “slope” that are selected from Chi Square feature selection gained 83.65% on precision, 76.32% on recall, 78.82% on f1-score, 89.37% on roc area, and 81.55% on accuracy.

**Table 5.6.** Mutual Information Feature Selection with XGBoost

Feature Selection	Feature Selected	Evaluation				
		<i>Precision</i>	<i>Recall</i>	<i>F1-score</i>	<i>ROC Area</i>	<i>Accuracy</i>
Mutual Information	13	82.86	78.65	79.10	90.25	83.32
	12	79.96	79.18	78.45	87.74	82.22
	11	78.43	78.42	77.18	86.81	81.08
	10	80.70	76.37	77.34	88.98	81.44
	9	82.30	78.17	79.28	87.98	83.36
	8	78.30	77.42	77.06	88.36	81.11
	7	80.12	75.36	76.39	87.06	80.71
	6	78.76	76.55	76.58	86.44	80.68
	5	82.80	80.51	80.67	88.73	84.50

	4	81.96	76.78	78.46	87.96	82.99
	3	83.08	80.40	81.36	88.10	84.86
	2	74.51	65.59	67.81	82.43	74.19
	1	74.69	73.37	72.20	76.33	77.18

The Mutual Information feature selection result can be seen in Table 5.6 above. Top 3 feature selected from Mutual Information feature selection has the best accuracy result. The top 3 “thal”, “cp”, and “ca” that are selected from Mutual Information feature selection gained 83.08% on precision, 80.40% on recall, 81.36% on f1-score, 88.10% on roc area, and 84.86% on accuracy.

**Table 5.7.** ANOVA Feature Selection with XGBoost

Feature Selection	Feature Selected	Evaluation				
		<i>Precision</i>	<i>Recall</i>	<i>F1-score</i>	<i>ROC Area</i>	<i>Accuracy</i>
ANOVA	13	82.86	78.65	79.10	90.25	83.32
	12	81.10	78.51	78.70	89.53	82.58
	11	81.93	76.31	77.60	90.49	81.21
	10	80.09	75.15	76.14	90.66	80.67
	9	81.48	75.69	76.82	89.81	81.84
	8	79.82	75.75	76.70	88.28	81.07
	7	80.34	75.51	76.54	88.07	81.47
	6	78.76	76.55	76.58	86.44	80.68
	5	78.16	76.80	76.59	86.30	80.31
	4	73.13	72.22	70.99	86.15	75.73
	3	75.57	74.41	73.11	87.14	77.64
	2	75.04	81.52	76.60	86.54	79.91
	1	74.69	73.37	72.20	76.33	77.18

The ANOVA feature selection result can be seen in Table 5.7 above. Top 13 feature selected from ANOVA feature selection has the best accuracy result. This means that using all of the feature is still better than using the selected feature that are gained from ANOVA feature selection with 82.86% on precision, 78.65% on recall, 79.10% on f1-score, 90.25% on roc area, and 83.32% on accuracy.

**Table 5.8.** Forward Feature Selection with XGBoost

Feature Selection	Feature Selected	Evaluation				
		<i>Precision</i>	<i>Recall</i>	<i>F1-score</i>	<i>ROC Area</i>	<i>Accuracy</i>
Forward	13	82.86	78.65	79.10	90.25	83.32
	12	82.53	77.80	78.92	90.58	82.95
	11	83.27	78.84	79.87	90.74	83.73
	10	81.24	81.84	80.40	91.75	84.09
	9	81.90	80.69	80.35	91.93	84.49
	8	81.34	82.45	80.69	92.39	84.47
	7	82.71	81.80	80.99	89.80	84.49
	6	81.22	81.11	80.48	89.58	84.13
	5	80.79	82.07	80.54	90.13	84.13
	4	82.23	80.11	80.81	89.35	84.49
	3	83.08	80.40	81.36	88.10	84.86
	2	75.04	81.52	76.60	86.54	79.91
	1	74.69	73.37	72.20	76.33	77.18

The Forward feature selection result can be seen in Table 5.8 above. Top 3 feature selected from Forward feature selection has the best overall performance result. The top 3 “thal”, “ca”, and “cp” that are selected from Forward feature selection gained 83.08% on precision, 80.40% on recall, 81.36% on f1-score, 88.10% on roc area, and 84.86% on accuracy.

**Table 5.9.** Backward Feature Selection with XGBoost

Feature Selection	Feature Selected	Evaluation				
		<i>Precision</i>	<i>Recall</i>	<i>F1-score</i>	<i>ROC Area</i>	<i>Accuracy</i>
Backward	13	82.86	78.65	79.10	90.25	83.32
	12	81.38	81.65	80.34	91.32	84.09
	11	83.66	78.65	80.24	90.82	84.12
	10	82.45	78.42	79.48	87.99	83.36
	9	80.27	73.91	76.24	87.81	80.75
	8	77.87	74.40	75.80	86.67	80.78
	7	79.01	70.11	73.54	86.78	80.36
	6	75.73	74.20	74.40	86.11	79.59
	5	75.05	72.40	73.26	84.32	78.46
	4	77.95	73.20	75.06	84.41	80

	3	77.05	72.44	74.09	84.86	79.26
	2	70.42	68.39	68.77	79.35	74.77
	1	71.46	67.82	68.61	74.99	74.30

The Backward feature selection result can be seen in Table 5.9 above. Top 11 feature selected from Backward feature selection has the best accuracy result. The top 11 “ca”, “oldpeak”, “cp”, “fbs”, “exang”, “age”, “chol”, “trestbps”, “restecg”, “sex”, and “thal” that are selected from Backward feature selection gained 83.66% on precision, 78.65% on recall, 80.24% on f1-score, 90.82% on roc area, and 84.12% on accuracy.

**Table 5.10.** Recursive Feature Elimination with XGBoost

Feature Selection	Feature Selected	Evaluation				
		<i>Precision</i>	<i>Recall</i>	<i>F1-score</i>	<i>ROC Area</i>	<i>Accuracy</i>
Recursive Feature Elimination	13	82.86	78.65	79.10	90.25	83.32
	12	81.28	78.13	78.38	89.08	82.19
	11	79.97	74.55	75.79	88.88	80.73
	10	80.55	78.55	78.09	91.07	82.21
	9	80.57	76.13	76.93	91.07	81.48
	8	78.36	74.58	74.84	89.12	79.93
	7	80.69	73.96	75.54	81.24	81.08
	6	79.78	75.11	76.61	89.12	81.50
	5	79.06	75.06	76.18	88.23	81.11
	4	81.96	76.78	78.46	87.96	82.99
	3	83.08	80.40	81.36	88.10	84.86
	2	75.04	81.52	76.60	86.54	79.91
	1	74.69	73.37	72.20	76.33	77.18

The Recursive Feature Elimination result can be seen in Table 5.10 above. Top 3 feature selected from Recursive Feature Elimination has the best accuracy result. The top 3 “thal”, “ca”, and “cp” that are selected from Recursive Feature Elimination gained 83.08% on precision, 80.40% on recall, 81.36% on f1-score, 88.10% on roc area, and 84.86% on accuracy.

**Table 5.11.** Feature Importance Feature Selection with XGBoost

Feature Selection	Feature Selected	Evaluation				
		<i>Precision</i>	<i>Recall</i>	<i>F1-score</i>	<i>ROC Area</i>	<i>Accuracy</i>
Feature Importance	13	82.86	78.65	79.10	90.25	83.32
	12	81.28	78.13	78.38	89.08	82.19
	11	82	77.84	78.71	89.65	82.98
	10	82.41	78.13	89.90	90.01	82.96
	9	80.57	76.13	76.93	91.07	81.48
	8	78.36	74.58	74.84	89.12	79.93
	7	80.34	75.51	76.54	88.07	81.47
	6	79.78	75.11	76.61	89.12	81.50
	5	81.11	79.45	79.66	89.64	83.70
	4	81.86	81.51	81.02	88.77	84.47
	3	78.03	72.36	74.22	88.23	79.54
	2	75.04	81.52	76.60	86.54	79.91
	1	74.69	73.37	72.20	76.33	77.18

The Feature Importance feature selection result can be seen in Table 5.11 above. Top 4 feature selected from Feature Importance feature selection has the best accuracy result. The top 4 “thal”, “ca”, “exang”, and “cp” that are selected from Feature Importance feature selection gained 81.86% on precision, 81.51% on recall, 81.02% on f1-score, 88.77% on roc area, and 84.47% on accuracy.



**Table 5.12.** Feature Selection Evaluation Comparison with XGBoost

Feature Selection	Best Feature Selected	Evaluation				
		<i>Precision</i>	<i>Recall</i>	<i>F1-score</i>	<i>ROC Area</i>	<i>Accuracy</i>
Chi Square	11	85.65	80.22	81.94	90.52	85.63
Mutual Information	3	83.08	80.40	81.36	88.10	84.86
ANOVA	13	82.86	78.65	79.10	90.25	83.32
Forward Feature Selection	3	83.08	80.40	81.36	88.10	84.86
Backward Feature Selection	11	83.66	78.65	80.24	90.82	84.12
Recursive Feature Elimination	3	83.08	80.40	81.36	88.10	84.86
Feature Importance	4	81.86	81.51	81.02	88.77	84.47

The comparison of each feature selection can be seen in Table 5.12 above. Chi Square Feature Selection with 11 selected feature achieve the highest accuracy among the other feature selection. With “thalach”, “ca”, “thal”, “oldpeak”, “exang”, “chol”, “age”, “cp”, “sex”, “restecg”, and “slope” feature, the XGBoost model with Chi Square feature selection able to achieve 85.65% on precision, 80.22% on recall, 81.94% on f1-score, 90.52% on roc area, and 85.63% on accuracy.

**Table 5.13.** Feature Selection XGBoost Comparison

XGBoost Model	Evaluation				
	<i>Precision</i>	<i>Recall</i>	<i>F1-score</i>	<i>ROC Area</i>	<i>Accuracy</i>
With feature selection	85.65	80.22	81.94	90.52	85.63
Without feature selection	82.86	78.65	79.10	90.25	83.32

The Chi Square Feature Selection is used in this comparison as it is the best result than the other feature selection. XGBoost Algorithm with and without feature selection using dropping missing value technique performance can be seen in Table 5.13 above. The XGBoost with chi square feature selection gained 85.65% on precision, 80.22% on recall, 81.94% on f1-score, 90.52% on roc area, and 85.63% on accuracy. While the XGBoost without feature selection gained 82.86% on precision, 78.65% on recall, 79.10% on f1-score, 90.25% on roc

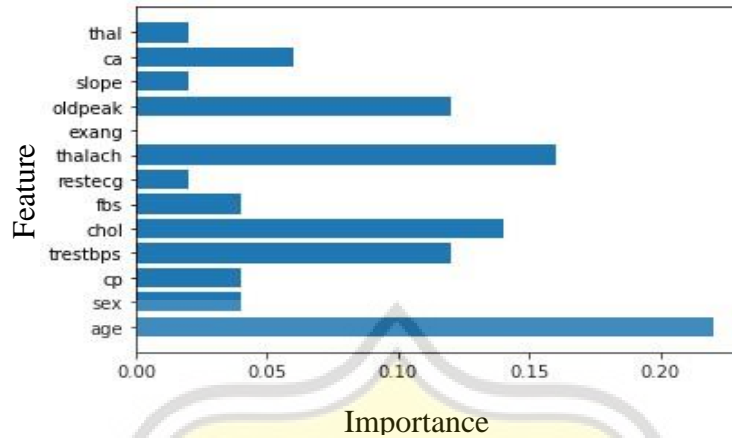
area, and 83.32% on accuracy. The result shows that XGBoost with feature selection has better performance than XGBoost without feature selection.

**Table 5.14.** Feature Selection Rankings with AdaBoost

Rank	Feature Selection Techniques					
	<i>Chi Square</i>	<i>Mutual Information</i>	<i>ANOVA</i>	<i>Forward Feature Selection</i>	<i>Backward Feature Selection</i>	<i>Recursive Feature Elimination</i>
1	thalach	thal	thal	thal	ca	thalach
2	ca	cp	ca	ca	slope	age
3	thal	ca	oldpeak	cp	cp	oldpeak
4	oldpeak	oldpeak	thalach	slope	sex	chol
5	exang	exang	cp	sex	chol	trestbps
6	chol	thalach	exang	exang	thalach	cp
7	age	slope	sex	restecg	chol	sex
8	cp	sex	slope	fbs	oldpeak	ca
9	sex	chol	age	trestbps	age	thal
10	restecg	restecg	restecg	oldpeak	restecg	fbs
11	slope	age	trestbps	age	exang	slope
12	trestbps	fbs	chol	chol	fbs	restecg
13	fbs	trestbps	fbs	thalach	thal	exang

**Table 5.15.** Feature Importance Rankings with AdaBoost

Rank	Feature Importance
1	age
2	thalach
3	chol
4	trestbps, oldpeak
5	ca
6	Sex, cp, fbs
7	Restecg, slope, thal
8	exang



**Figure 5.4** AdaBoost Feature Importance

Each feature rankings from each feature selection can be seen in Table 5.14 above. “thal”, “ca”, “cp”, and “thalach” has the most appearance on top 3 rankings from feature selection that are used in this research. While the “slope”, “fbs”, and “treshbps” has the most appearance on bottom 3 rankings from feature selection that are used in this research. This shows that “thal”, “ca”, “cp”, and “thalach” is most significant feature for AdaBoost to predict target feature. While the “slope”, “fbs”, and “treshbps” is less significant feature for AdaBoost to predict target feature. The feature importance for AdaBoost that can be seen in Table 5.15 above, have several feature that have the same importance. As shown in Figure 5.4 above, “thal”, “slope”, and “restecg” have the same importance, “fbs”, “cp”, and “sex” have the same importance, “oldpeak” and “trestbps” have the same importance.

**Table 5.16.** Chi Square Feature Selection with AdaBoost

Feature Selection	Feature Selected	Evaluation				
		<i>Precision</i>	<i>Recall</i>	<i>F1-score</i>	<i>ROC Area</i>	<i>Accuracy</i>
Chi Square	13	77.74	76.22	76.22	86.58	79.59
	12	75.93	74.94	74.25	82.80	78.48
	11	75.05	74.22	73.39	83.64	78.02
	10	76.31	73.60	73.54	83.74	78.39
	9	74.45	71.26	71.69	84.41	78.06
	8	75.64	74.13	73.88	80.79	78.80

	7	71.34	71.84	70.03	79.96	74.60
	6	75.05	76.28	74.88	80.71	78.75
	5	71.72	77.89	73.39	83.61	76.87
	4	71.59	77.27	73.04	84.29	76.48
	3	71.76	76.75	72.80	82.66	76.48
	2	64.02	64.70	62.47	77.25	68.16
	1	66.49	56.76	60.47	67.07	70.07

The Chi Square feature selection result can be seen in Table 5.16 above. Top 13 feature selected from Chi Square feature selection has the best accuracy result. This means that using all of the feature is still better than using the selected feature that are gained from Chi Square feature selection with 77.74% on precision, 76.22% on recall, 76.22% on f1-score, 86.58% on roc area, and 79.59% on accuracy.

**Table 5.17.** Mutual Information Feature Selection with AdaBoost

Feature Selection	Feature Selected	Evaluation				
		<i>Precision</i>	<i>Recall</i>	<i>F1-score</i>	<i>ROC Area</i>	<i>Accuracy</i>
Mutual Information	13	77.74	76.22	76.22	86.58	79.59
	12	76.61	77.32	76.23	85.07	79.22
	11	75.05	74.22	73.39	83.64	78.02
	10	74.82	78.32	76.07	85.41	79.22
	9	76.32	74.84	74.91	83.04	79.19
	8	78.12	76.89	76.50	86.46	79.94
	7	76.73	77.55	76.09	85.32	79.56
	6	79.22	77.07	77.43	86.42	81.50
	5	76.89	78.75	76.72	84.32	80.28
	4	75.66	75.13	74.50	82.90	78.42
	3	80.20	81.02	80.08	87.23	83.73
	2	74.95	67.41	69.05	82.92	75.31
	1	74.69	73.37	72.20	76.33	77.18

The Mutual Information feature selection result can be seen in Table 5.17 above. Top 3 feature selected from Mutual Information feature selection has the best accuracy result. The top 3 “thal”, “cp”, and “ca” that are selected from Mutual Information feature selection gained 80.20%

on precision, 81.02% on recall, 80.08% on f1-score, 87.23% on roc area, and 83.73% on accuracy.

**Table 5.18.** ANOVA Feature Selection with AdaBoost

Feature Selection	Feature Selected	Evaluation				
		<i>Precision</i>	<i>Recall</i>	<i>F1-score</i>	<i>ROC Area</i>	<i>Accuracy</i>
ANOVA	13	77.74	76.22	76.22	86.58	79.59
	12	75.93	74.94	74.25	82.80	78.48
	11	76.81	75.17	75.14	85.24	78.82
	10	74.44	75.98	73.89	84.63	77.66
	9	74.19	72.84	72.13	84.53	76.94
	8	78.12	76.89	76.50	86.46	79.94
	7	76.66	75.17	75.12	86.24	78.82
	6	74.22	76.60	74.45	84.75	78.45
	5	74.28	76.60	74.59	75.15	78.45
	4	71.59	77.27	73.04	84.29	76.48
	3	74.14	75.17	73.31	84.07	77.24
	2	71.85	80.09	74.38	85.29	77.61
	1	74.69	73.37	72.20	76.33	77.18

The ANOVA feature selection result can be seen in Table 5.18 above. Top 8 feature selected from ANOVA feature selection has the best accuracy result. The top 8 “thal”, “ca”, “oldpeak”, “thalach”, “cp”, “exang”, “sex”, and “slope” that are selected from ANOVA feature selection gained 78.12% on precision, 76.89% on recall, 76.50% on f1-score, 86.46% on roc area, and 79.94% on accuracy.

**Table 5.19.** Forward Feature Selection with AdaBoost

Feature Selection	Feature Selected	Evaluation				
		<i>Precision</i>	<i>Recall</i>	<i>F1-score</i>	<i>ROC Area</i>	<i>Accuracy</i>
Forward	13	77.74	76.22	76.22	86.58	79.59
	12	80.29	74.26	76.22	85.88	80.73
	11	80.65	77.17	77.73	84.62	81.07
	10	77.85	81.36	78.88	85.76	81.88
	9	81.13	81.71	80.76	87.15	82.64

	8	79.61	82.42	80.24	89.23	83.35
	7	82.66	82.42	81.77	89.93	84.87
	6	80.25	83.61	81.23	89.58	84.53
	5	79.88	83.49	81.01	89.18	84.13
	4	80.84	82.78	81.22	88.76	84.49
	3	80.20	81.02	80.08	87.23	83.73
	2	71.85	80.09	74.38	85.29	77.61
	1	74.69	73.37	72.20	76.33	77.18

The Forward feature selection result can be seen in Table 5.19 above. Top 7 feature selected from Forward feature selection has the best overall performance result. The top 7 “thal”, “ca”, “cp”, “slope”, “sex”, “exang”, and “restecg” that are selected from Forward feature selection gained 82.66% on precision, 82.42% on recall, 81.77% on f1-score, 89.93% on roc area, and 84.87% on accuracy.

**Table 5.20.** Backward Feature Selection with AdaBoost

Feature Selection	Feature Selected	Evaluation				
		<i>Precision</i>	<i>Recall</i>	<i>F1-score</i>	<i>ROC Area</i>	<i>Accuracy</i>
Backward	13	77.74	76.22	76.22	86.58	79.59
	12	77.43	75.78	76.30	85.39	81.14
	11	77.50	75.82	76.32	84.33	81.11
	10	75.82	76.30	75.67	83.28	80.38
	9	75.30	74.20	74.31	84.34	79.67
	8	75.22	75.53	74.81	83.23	78.87
	7	75.30	78.02	75.96	84.05	79.22
	6	78.91	73.65	74.97	82.71	78.85
	5	79.67	71.87	74.46	84.03	78.87
	4	86.14	73.92	79.07	87.38	82.25
	3	78.37	79.93	78.54	84.86	80.71
	2	79.68	57.87	66.63	79.34	75.83
	1	71.46	67.82	68.61	74.99	74.30

The Backward feature selection result can be seen in Table 5.20 above. Top 4 feature selected from Backward feature selection has the best accuracy result. The top 4 “ca”, “slope”,



“cp”, and “sex” that are selected from Backward feature selection gained 86.14% on precision, 73.92% on recall, 79.07% on f1-score, 87.38% on roc area, and 82.25% on accuracy.

**Table 5.21.** Recursive Feature Elimination with AdaBoost

Feature Selection	Feature Selected	Evaluation				
		<i>Precision</i>	<i>Recall</i>	<i>F1-score</i>	<i>ROC Area</i>	<i>Accuracy</i>
Recursive Feature Elimination	13	77.74	76.22	76.22	86.58	79.59
	12	78.80	77.49	77.58	86.02	80.71
	11	76.04	71.98	73.11	82.65	78.09
	10	79.37	71.20	74.40	83.22	79.57
	9	75.83	72.51	73.24	80.37	77.69
	8	71.49	70.44	70.41	82.74	76.98
	7	71.79	73.70	71.82	81.87	76.60
	6	61.50	61.50	60.69	73.26	67.45
	5	61.43	55.24	57.04	67.40	66.25
	4	63.80	58.04	58.98	70.22	67.42
	3	62.22	54.20	56.36	73.38	65.56
	2	61.03	56.61	57.38	70.36	64.80
	1	66.49	56.76	60.47	67.07	70.07

The Recursive Feature Elimination result can be seen in Table 5.21 above. Top 12 feature selected from Recursive Feature Elimination has the best accuracy result. The top 12 “thalach”, “age”, “oldpeak”, “chol”, “trestbps”, “cp”, “sex”, “ca”, “thal”, “fbs”, “slope”, and “restecg” that are selected from Recursive Feature Elimination gained 78.80% on precision, 77.49% on recall, 77.58% on f1-score, 86.02% on roc area, and 80.71% on accuracy.

**Table 5.22.** Feature Importance Feature Selection with AdaBoost

Feature Selection	Feature Selected	Evaluation				
		<i>Precision</i>	<i>Recall</i>	<i>F1-score</i>	<i>ROC Area</i>	<i>Accuracy</i>
Feature Importance	13	77.74	76.22	76.22	86.58	79.59
	12	78.80	77.49	77.58	86.02	80.71
	9	72.97	69.11	70.62	82.23	77.36
	6	72.13	67.53	68.56	76.27	74.70
	5	61.43	55.24	57.04	67.40	66.25

	3	62.76	55.06	56.63	67.61	64.76
	2	61.03	56.61	57.38	70.36	64.81
	1	52.65	50.72	49.18	59.23	56.05

The Feature Importance feature selection result can be seen in Table 5.22 above. There are only 1, 2, 3, 5, 6, 9, 12, and 13 feature selected done in feature importance as there are several same importance in the feature shown in Figure 5.4 above. Top 12 feature selected from Feature Importance feature selection has the best accuracy result. The top 12 “age”, “thalach”, “chol”, “trestbps”, “oldpeak”, “ca”, “sex”, “cp”, “fbs”, “restecg”, and “slope” that are selected from Feature Importance feature selection gained 78.80% on precision, 77.49% on recall, 77.58% on f1-score, 86.02% on roc area, and 80.71% on accuracy.

**Table 5.23.** Feature Selection Evaluation Comparison with AdaBoost

Feature Selection	Best Feature Selected	Evaluation				
		Precision	Recall	F1-score	ROC Area	Accuracy
Chi Square	13	77.74	76.22	76.22	86.58	79.59
Mutual Information	3	80.20	81.02	80.08	87.23	83.73
ANOVA	8	78.12	76.89	76.50	86.46	79.94
Forward Feature Selection	7	82.66	82.42	81.77	89.93	84.87
Backward Feature Selection	4	86.14	73.92	79.07	87.38	82.25
Recursive Feature Elimination	13	77.74	76.22	76.22	86.58	79.59
Feature Importance	12	78.80	77.49	77.58	86.02	80.71

The comparison of each feature selection can be seen in Table 5.23 above. Forward Feature Selection with 7 selected feature achieve the highest accuracy among the other feature selection. With 7 “thal”, “ca”, “cp”, “slope”, “sex”, “exang”, and “restecg” feature, the AdaBoost model with Forward feature selection able to achieve 82.66% on precision, 82.42% on recall, 81.77% on f1-score, 89.93% on roc area, and 84.87% on accuracy.

**Table 5.24. Model with Feature Selection Comparison**

Model	Best Feature Selected	Evaluation				
		<i>Precision</i>	<i>Recall</i>	<i>F1-score</i>	<i>ROC Area</i>	<i>Accuracy</i>
XGBoost with Chi Square feature selection	11	85.65	80.22	81.94	90.52	85.63
AdaBoost with Forward Feature Selection	7	82.66	82.42	81.77	89.93	84.87

The comparison of XGBoost and AdaBoost using the best feature selection for each model can be seen in Table 5.24 above. The result shows that XGBoost with Chi Square feature selection have better performance than AdaBoost with Forward Feature Selection. XGBoost with Chi Square Feature Selection with 11 selected feature (“thalach”, “ca”, “thal”, “oldpeak”, “exang”, “chol”, “age”, “cp”, “sex”, “restecg”, and “slope”) gained 85.65% on precision, 80.22% on recall, 81.94% on f1-score, 90.52% on roc area, and 85.63% on accuracy. While AdaBoost with Forward Feature Selection with 7 selected feature (“thal”, “ca”, “cp”, “slope”, “sex”, “exang”, and “restecg”) gained 82.66% on precision, 82.42% on recall, 81.77% on f1-score, 89.93% on roc area, and 84.87% on accuracy.

**Table 5.25. Model without Feature Selection Comparison**

Model	Evaluation				
	<i>Precision</i>	<i>Recall</i>	<i>F1-score</i>	<i>ROC Area</i>	<i>Accuracy</i>
XGBoost without feature selection	82.86	78.65	79.10	90.25	83.32
AdaBoost without feature selection	77.74	76.22	76.22	86.58	79.59

The comparison XGBoost without feature selection and AdaBoost without feature selection is shown in Table 5.25 above. The result shows that XGBoost without feature selection has better performance than AdaBoost without feature selection. XGBoost gained 82.86% on precision, 78.65% on recall, 79.10% on f1-score, 90.25% on roc area, and 83.32% accuracy. AdaBoost gained 77.74% on precision, 76.22% on recall, 76.22% on f1-score, 86.58% on roc area, and 79.59% on accuracy.