Heart Disease Detection

Oğuzhan Eroğlu [1] **Harun Alperen Toktaş** [1]

Abstract

In this paper we introduce methods for heart disease detection and data sets analysis, We experiment with three different data sets. The first data set includes 303 records and 13 features + binary target. The second data set includes over 4000 records and 14 features + binary target. The third data set includes 70.000 records of patients data, 11 features + binary target. We aim to compare different machine learning algorithms on this data sets and their results. We are going to determine which algorithm is more suitable for the data sets we use for heart disease detection problem. We are going to also compare the results we have found for all algorithms with previous studies.

1. Keywords

Heart Disease Detection, Early Diagnosis, Prediction, Classification, AI For Good.

2. Introduction

We will introduce you to the project of the machine learning course that we are taking this semester. The theme of the projects of this semester is machine learning for good. So we decided to do Heart Disease Detection. The motivation behind choosing this project is to show people that machine learning techniques can be life-saving.

We miss a lot of things while dealing with the daily hustle and bustle. The most important of these is our health. Health is better than wealth. Millions of people die every year because of sudden heart diseases. These sudden cardiac disorders may result in sudden deaths, as well as life-long disturbances. Can we take precautions without letting such disturbances reduce our quality of life?

Early diagnosis is very important in the prevention of heart diseases. But most people are unaware that they have any problems, so they don't realize that they need to see a doctor for early diagnosis. By integrating machine learning techniques into real life, we can warn people for early detection and force them to take action. Thanks to our algorithm analysis for the Heart Disease Detection problem, people can receive early warning of heart disease using their smart watches, using their personal medical data in real time. At the same time, with a mobile application, people can get information about their health condition by entering their own medical data.

In this study, by using three different data sets, we tried to determine the risk of heart disease through the factors that cause. In line with our analyzes, we tried to find the algorithm that would give the most accurate results for these data sets and to determine what kind of changes the algorithms could show on different data sets. In addition, through these analyzes, we tried to make it easier for people to learn the risk of heart disease by entering their personal health information into the correct algorithms.At the same time, it may be possible for doctors to use their resources better.So we can help more people with the health budgets we have.

3. Related Work

As we said before, we have analyzed three different data sets. We have found a study [1] using the Machine Learning concept for the first data set only. The most important thing that distinguishes this study from others is its feature selection. The feature selection process and clear the noisy data process was done with a software tool called rapid miner. In this way, data preprocessing was also more successful. In addition, in another study [2] using the same software tool, they also used the cross-validation technique. What distinguishes this work from theirs is that they use 10-fold instead of 5-fold. Thus, the diversity required to make more generalization is provided. At the same time, they tripled the data set by creating a random sample of the using maximum and minimum values for each feature. In this way, they have increased the number of samples that are less to generalize in order to give better results.

4. The Approach

We have tried approaches that are known to work well on these data sets or similar data sets and have been proven in

[*]Equal contribution [1]Hacettepe University, Department of Computer Engineering. Correspondence to: <>.

BBM406 Fundamentals of Machine Learning, Fall 2019. Copyright 2019 by the author(s).

related works. We conducted experiments and analyzes on Knn, Naive Bayes, Logistic Regression, Decision Tree and Random Forest algorithms. In this section we are going to introduce these algorithms and discuss why they might be suitable for our problem and data sets.

Knn is a good baseline. So we are going to discuss about Knn first. The Knn algorithm can be used for classification and regression, but its use for classification is more common. Although Knn is quite simple, it can be functional. When we evaluate our training data as points on the plane in Knn algorithm, when we test a new sample, we see which class is the most common from the nearest K points. Since the Knn algorithm is not a parametric algorithm, it does not make any assumptions about the distribution of the data. This is precisely why it is an algorithm worth trying for all data sets. It is also possible to extend the Knn algorithm to weighted Knn and get better results.

We are going to talk about Naive Bayes after the Knn algorithm. The Naive Bayes algorithm is based on the assumption that the features in the data set are conditionally independent of each other. If the data sets we use contain features close to this assumption, we expect the Naive Bayes algorithm to work well. Therefore, we should first examine the relationship between the features of our data sets.

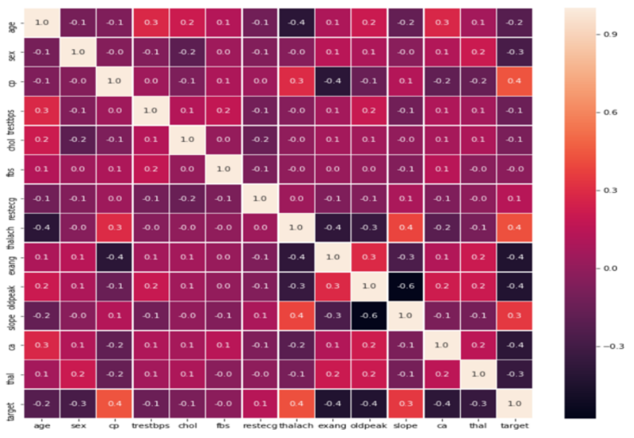


Figure 1. Correlation map for dataset 1.

It is a correlation map for UCI dataset (data set 1) we used. Correlations of features may be negative or positive. However, as shown in the correlation map, the correlations of the features with each other are quite small in absolute value. This shows us that the dependencies of the features are low and that they are very close to the assumption in the Naive Bayes algorithm. So we expect the Naive Bayes algorithm to work well.

...

...

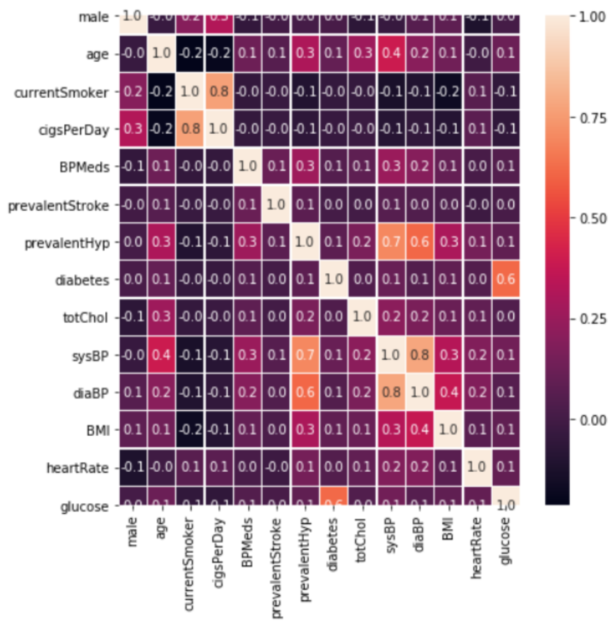


Figure 2. Correlation map for dataset 2.

It is a correlation map for data set 2 we used. Here, the correlation of some features with each other seems to be quite small in absolute value and the correlations of some features with each other are not so small in absolute value. Therefore, we can expect the Naive Bayes algorithm for data set 2 to work worse than data set 1.

The Naive Bayes algorithm can be implemented in different ways (Gaussian and Bernoulli) depending on whether the features in the data set used are mostly binary or continuous. We are going to decide which Naive Bayes algorithm is more suitable for each of our datasets and implement it accordingly.

After the Naive Bayes algorithm, we want to discuss about linear separability and therefore logistic regression algorithm since our data sets contain binary target values. The Logistic Regression algorithm classifies linearly separable data sets by drawing a linear boundary. If the data can be divided by a linear boundary, we expect the Logistic Regression algorithm to work with a high accuracy. Therefore, when we run the Logistic Regression algorithm, if we get a good result, we are going to realize that our data sets can be separated linearly. Moreover, since our target value is binary, it is possible to classify by drawing a single decision boundary. If our target value was not binary, we would have to draw more than one decision boundary with one vs all method.

Since the data sets we used consist of continuous and discrete features, we thought that the decision tree algorithm might be suitable for our problem. Decision trees are easier to interpret and understand, as it is not black box algorithms, like some other common algorithms(such as svm or neural networks). It is possible to understand the logic behind it. We ask to the features yes (true) or no (false) questions, and we try to reach the leaves and classify them by branching. The root feature of these tree structures is the best distinguishing feature for classification. The decision tree algorithm is a powerful, easy-to-train algorithm that does not require preliminary work on data. Another advantage of the decision tree is that it is robust to outliers and irrelevant inputs. The disadvantage of the decision tree is that it causes overfitting. The decision tree algorithm causes branching until the entropy of the leaf nodes are zero. That's what caused the overfitting.

In order to solve the overfitting problem caused by the decision tree algorithm and create a more general model, we are going to talk about random forest algorithm. The random forest algorithm is a more flexible version of the decision tree algorithm. It provides a better generalization and more accurate classification of new samples. Therefore, it provides an improvement in the accuracy rate. When constructing random forest algorithm, we select random samples from the original dataset and create a new dataset and run the decision tree algorithm on this new dataset. We repeat this decision tree process many times. The important thing to note here is that the data set we created contains the same number of samples as the original data set, and we can select the same sample more than once (Completely random). When creating decision trees, we must use a randomly selected subset of features at each step. Forming decision trees randomly in this way results in a wide variety of trees. Variety is what makes the random forest algorithm more efficient than the single decision tree algorithm. When testing a new instance, we test it on all random trees that we create and vote for the majority.

The last algorithm we are going to try is the SVM algorithm. Logistic regression algorithm will give good results if data can be separated linearly. However, if the data is not linearly separable, kernel tricks can make the data linearly separable. That's why we find the SVM algorithm worth trying.

The support vector machine algorithm detects the closest points on the data set from different classes and tries to divide the points between them in a linear manner. When doing this, it tries to divide the nearest points at the maximum distance (margin). Various kernel functions are available to divide multi-dimensional non-linearly divisible data sets in a linear manner. For example, 'linear', 'rgb', 'sigmoid', 'polynomial'. This is one of the strengths of Support Vector Machine Algorithm.

5. Experimental Results

We are going to start with the analysis of our base algorithm Knn. We used Sklearn train test split function. We splitted data 75 percent as train and 25 percent as test. And we used random state parameter as 4 from Sklearn.

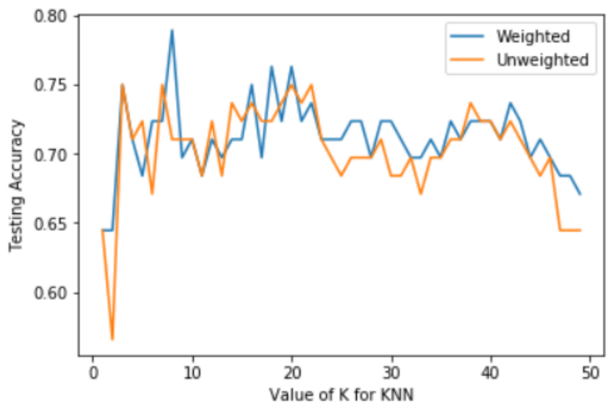


Figure 3. Accuracies for different K values for data set(1) with 303 records.

As can be seen from this graph, the best accuracy for normal KNN are obtained at 3, 7, 20 and 22 k values. We choose 3 for k value because less operation is possible with small k value. The maximum accuracy is 75 percent for normal KNN. Of course it is possible to increase this accuracy further. We tried Weighted KNN algorithm to do this. The best accuracy is the maximum value for k=8 for weighted KNN. The maximum accuracy is increased to 78.9 percent . There is an increase of 4 percent.

We are going to look at the results of our second data set.

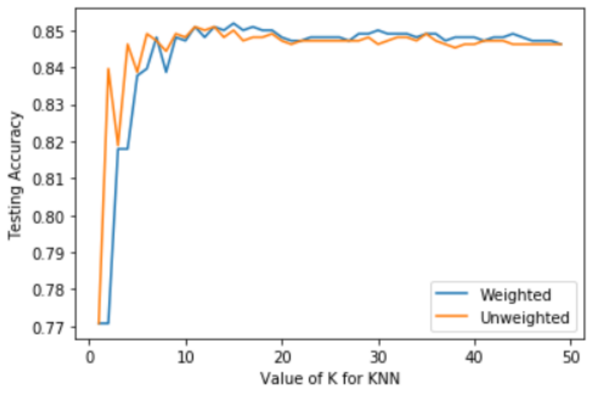


Figure 4. Accuracies for different K values for data set(2) with 4238 records.

As you can see from the graph, the accuracy has the maximum value for k=13, the maximum accuracy is 85 percent for normal KNN. The accuracy has the maximum value for k=15, the maximum accuracy is 85.1 percent for weighted KNN.

Data set 2 gave better results than data set 1 because it contained more samples. Because we need more data to make more accurate generalizations. The first thing that comes to mind here is that the number and characteristics of the features are different, but there are no major changes in size of feature dimension between the two datasets. This showed us that if the number of dimensions does not change much, increasing the number of samples can give better results.

While the weighted Knn algorithm gave better results on the first data set, it did not give better results on the second data set. We think the reason for this is class imbalance problem. Class imbalance is based on the fact that the group with a certain value in a class is more in number. In this data set, there are 3594 '0' labels and 644 '1' labels. There is an imbalanced class distribution since the distribution is not 50–50. The class with more data is called 'major class' and the less with 'minor class'. When dealing with imbalanced data, minority examples are given the same weight as majority examples in the existing KNN algorithm. Therefore, the weighted KNN algorithm does not give better results than the normal KNN algorithm.

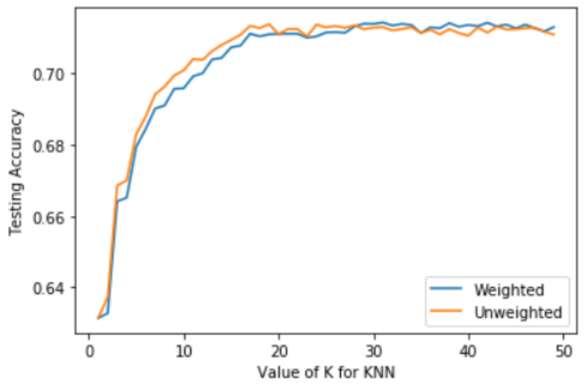
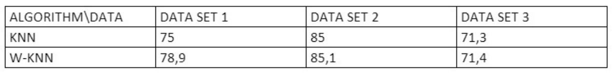


Figure 5. Accuracies for different K values for data set(3) with 70000 records.

As you can see from the graph in figure 5, the accuracy has the maximum value for k=20, the maximum accuracy is 71.3 percent for normal KNN. The accuracy has the maximum value for k=31, the maximum accuracy is 71,4 percent for weighted KNN.

When we examine the third data set for Knn algorithm, there is too much noise in the data. To give an example there are many people shorter than the shortest people in the world. It also has adult people who are 2.5 meter tall or in the range of 10-20 kilograms. This creates a question mark in mind and undermines the reliability of the data set. Data could not be properly distributed because the data set contained a large number of noisy and possibly fictitious data. So when we tried Naive Bayes, Logistic Regression and even Decision Tree algorithm on the 3rd data set, it caused us to get low accuracy. However, Knn algorithm gave the best results compared to others.

We think that the Knn algorithm gives better results on this data set than the other algorithms that we have used because of the majority application we have made in Knn algorithm. While majority voting held,the noisy data can remain in minority part. For example, if we select 12 as the k value, it makes a more accurate result if the number of noisy data is less than 6.



ALGORITHM\DATA	DATA SET 1	DATA SET 2	DATA SET 3
KNN	75	85	71,3
W-KNN	78,9	85,1	71,4

Figure 6. These are the accuracies for Knn algorithms.

We said that the Naive Bayes algorithm provides good results where the assumption of conditional independence is valid. When we looked at the correlations for the first and second data sets, we realized that this assumption was largely valid. And we said that, The Naive Bayes algorithm can be implemented in different ways depending on whether the features in the data set are continuous or binary.

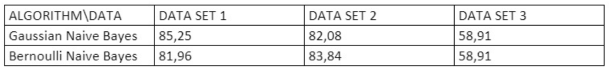
We used Sklearn train test split function. We splitted data 80 percent as train and 20 percent as test. And we used random state parameter as 0 from Sklearn.

When we used the Gaussian Naive Bayes algorithm in both data sets, we achieved 3 percent more accuracy on the first data set compared to the second data set with Gaussian Naive Bayes algorithm due to the fact that the continuous features were higher proportion in our first data set.

In the second data set, we thought that the Bernoulli Naive Bayes algorithm would give better results because the number of our binary features was higher than the first data set. When we performed the implementation, we showed that the second data set gave a better result with a accuracy difference of 1-2 percent.

Gaussian Naive Bayes algorithm won the first data set and Bernoulli algorithm won the second data set.

We have already said that the 3rd data set contains a large amount of noisy data When we implement the Naive Bayes Algorithm on this data set, we find the result of Gaussian Naive Bayes algorithm is 58.91 percent and the result of Bernoulli Naive Bayes algorithm is 58.91 percent. We did not expect both algorithms to produce the same result. We expected the Gaussian Naive Bayes algorithm to yields better result because the number of binary features was less. But it did not. Since the source of the dataset is unreliable, it is difficult to say certain information. However, with each analysis we have found that this data set is unreliable and contains a large amount of noisy data.



ALGORITHM\DATA	DATA SET 1	DATA SET 2	DATA SET 3
Gaussian Naive Bayes	85,25	82,08	58,91
Bernoulli Naive Bayes	81,96	83,84	58,91

Figure 7. These are the accuracies for Naive Bayes algorithms.

After the Naive Bayes algorithm, we are going to discuss the results of Logistic Regression algorithm.

When we applied logistic regression algorithm on the first data set, we found the accuracy rate as 90.16 percent. With this very high accuracy rate, we can observe that the data can be divided linearly along a boundary.

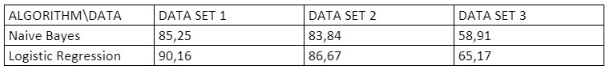
We used Sklearn train test split function. For data 1 : test size = 0.20, random state = 4, for data 2 : test size = 0.2, random state = 1, for data 3 : test size = 0.20, random state = 2

When we applied logistic regression algorithm on the second data set, we found the accuracy value to be 86.67 percent. Again, thanks to this very good accuracy rate, we can say that our data set can be divided along a linear boundary.

The reason that the accuracy value in the first data set is higher than the second data set may be that the first data set contains fewer samples. Because it is more difficult to generalize data sets with fewer samples.

When we obtained the results with the logistic regression algorithm on the third data set, the accuracy rate was 65.17 percent. Since our data set contains noisy data, it is not possible to say that the logistic regression algorithm has a strong accuracy on this data set.

When we try both Logistic Regression and Naive Bayes algorithm, it is possible to say that Logistic Regression algorithm makes more accurate decisions. Rather than features independent of each other for the classification process, we see that the linearity of the data set has more effect on these two data sets.



ALGORITHM\DATA	DATA SET 1	DATA SET 2	DATA SET 3
Naive Bayes	85,25	83,84	58,91
Logistic Regression	90,16	86,67	65,17

Figure 8. Comparative results of Naive Bayes and Logistic Regression algorithms.

We decided to use a decision tree algorithm because our data sets contain discrete and continuous features. We are going to now discuss the results.

We used sklearn train test split function. For data 1 : test size = 0.30, random state = 2, for data 2 : test size = 0.30, random state = 1, test size = 0.20, random state = 2

When we applied the decision tree algorithm on our first data set, we found the accuracy rate as 86.5. We found this accuracy to be reasonable and realistic because our data set contains both discrete and continuous data.

When we applied the decision tree algorithm on our second data set, we found the accuracy rate as 78.14. We have mentioned that the decision tree algorithm causes branching until the entropy of the leaf nodes makes zero and causes overfitting. Since our second data set is larger than the first data set, more complex tree structure has been formed. This reduces the possibility of generalization of our model. This is due to a lower accuracy rate when we test our model on test data. When we test our model on training data without separating it as training and test data, we get 100 percent accuracy. This is an indication that it is overfitting.

When we applied the decision tree algorithm in the third data set, we found the accuracy rate to be 63.5. The decision tree algorithm selects a distinctive feature at each step. This algorithm selects a new splitter and continues to create branching until the leaf nodes are completely pure. It was not easy to obtain purity due to the excessive noise in this dataset, and the relative accuracy on the test data was lower due to excessive branching.

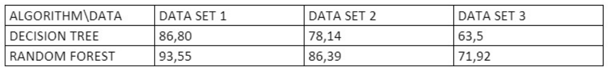
We have mentioned that the decision tree algorithm causes overfitting. We tried the random forest algorithm to prevent this overfitting and to build a more general model. We are going to discuss the consequences.

We used sklearn train test split function. For data 1 : test size = 0.10, random state = 2, for data 2 : test size = 0.30, random state = 1, test size = 0.20, random state = 2

For each of the three datasets, we generated one hundred random decision trees.

When we tried random forest algorithm in our first data set, we obtained the accuracy value as 93.55. In our second data set, the random forest algorithm accuracy was 86.39. In our third data set, random forest algorithm yielded 71.92 accuracy.

For all three data sets, the random forest algorithm gave higher accuracy than the decision tree algorithm. This is due to the fact that the decision tree algorithm has low generalization capacity and causes overfitting when it encounters new examples other than training data. The random forest algorithm solves this generalization problem by making the majority vote among the various decision trees.



ALGORITHM\DATA	DATA SET 1	DATA SET 2	DATA SET 3
DECISION TREE	86,80	78,14	63,5
RANDOM FOREST	93,55	86,39	71,92

Figure 9. Comparative results of Decision Tree and Random Forest algorithms.

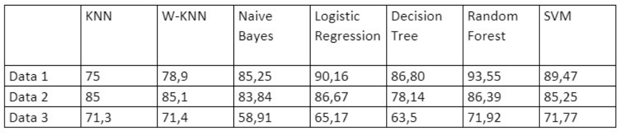
After the Logistic Regression algorithm, we tried the SVM algorithm. We will discuss the consequences.

We used Sklearn train test split function. For data 1 : test size = 0.25, random state = 2, for data 2 : test size = 0.20, random state = 3, test size = 0.20, random state = 0

When we ran the Support Vector Machine Algorithm in the first data set, we found that accuracy rate is 89.47 percent . In Logistic Regression algorithm, we found a value close to this accuracy value. In this way, we understood that the data could be separated linearly. Here, when we prefer linear kernel as the kernel function, the fact that the accuracy is close shows us how powerful the algorithms work on linear divisible data sets.

When we run the SVM algorithm in the second data set, we found the accuracy value to be 85.25 percent. With this very good accuracy value, we have once again realized that the second data set can also be divided linearly.

When we run the SVM algorithm in the third data set, we find the accuracy 71.77 percent. The SVM algorithm improves the probability of linearly dividing the data in this noisy data.



	KNN	W-KNN	Naive Bayes	Logistic Regression	Decision Tree	Random Forest	SVM
Data 1	75	78,9	85,25	90,16	86,80	93,55	89,47
Data 2	85	85,1	83,84	86,67	78,14	86,39	85,25
Data 3	71,3	71,4	58,91	65,17	63,5	71,92	71,77

Figure 10. These are the final results of all algorithms we tried.

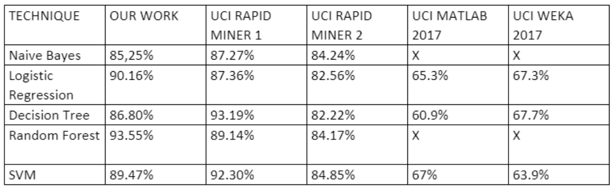
6. Conclusions

In the light of these analyzes, instead of solving a problem with single data set, we have benefited from all kinds of different data sets in solving this problem by using data sets pointing to the same problem. We obtained our first data set from a very reliable source. There were a few missing values in first data set. However, the number of samples was small. Thanks to this data set, we had an idea about the data sets that contain a small number of data. For example, the possibility of creating a bias has increased due to the small sample size. We understood that we needed more examples to make better generalizations. At the same time, thanks to previous studies on this data set, we realized that the feature selection method can be used to increase the accuracy.

If we talk about the second data set, the source of this data set was also very reliable. Since the number of samples in this data set is much higher than the previous data set, it is possible to make more accurate generalizations. Although the increase in the number of samples causes us to get less accuracy in some algorithms, we believe that the results are more reliable because it makes better generalization and does not memorize. We have also found that both Logistic Regression Algorithm and Support Vector Machine (linear kernel) Algorithm yield good results, so that these two sets of data can be divided along a linear line. We also discovered that the Naive Bayes algorithm works well thanks to the independent features in both data sets.

To speak for the third data set, we received this data set from Kaggle. However, the person who shared it did not share the source of the data. However, as we said before, the data contains a lot of noise. The question that comes to mind is why we use a corrupted data set. We made this data set to understand what the algorithms would yield for a corrupted data set and to see how the results could be obtained from the analysis.

As a result of our algorithm analysis, we realized that the best algorithm for this problem is random forest algorithm.



TECHNIQUE	OUR WORK	UCI RAPID MINER 1	UCI RAPID MINER 2	UCI MATLAB 2017	UCI WEKA 2017
Naive Bayes	85,25%	87.27%	84.24%	X	X
Logistic Regression	90.16%	87.36%	82.56%	65.3%	67.3%
Decision Tree	86.80%	93.19%	82.22%	60.9%	67.7%
Random Forest	93.55%	89.14%	84.17%	X	X
SVM	89.47%	92.30%	84.85%	67%	63.9%

Figure 11. Comparative results of our works and related works. UCI RAPID MINER 1[1], UCI RAPID MINER2[2], UCI MATLAB/WEKA[3]

7. Future Direction

As we mentioned before, As the study in the first dataset has achieved better results than we did, data preprocessing can be improved by using Rapid miner or another tool. At the same time, the capacity to generalize can be increased by using cross validation. We said there was a lot of noisy data on the third data set. We made this determination by examining the samples based on age, height and weight. Based on the results we obtained with the algorithm analysis, we concluded that the data is not very reliable, contains noise and may be artificially created. In the following steps, it may be possible to recover the data set from noisy data and obtain better results by performing a detailed data preprocessing for this data set. Of course, our ultimate goal was not to increase the accuracy of a data set more, but rather to see how the algorithms work on different data types according to the operating principles. We believe that our expectations are met.

References

[1] F. S. Alotaibi, "Implementation of machine learning model to predict heart failure disease," 2019.

[2] F. H. K. A. A. K. B. Saba Bashir, Zain Sikander Khan, "Improving heart disease prediction using feature selection approaches," 2019.

[3] S. Ekiz and P. Erdogmus, "Comparative study of heart disease classification," 2017.