2022

Using classification algorithms to determine risk of developing heart disease

FIN WATLING – CI512

BRIGHTON UNIVERSITY

Using classification algorithms to determine risk of developing heart disease

Functionality

For this project I chose to investigate how we could use classification algorithms to predict if a person was likely to develop heart disease based on physical information about themselves. I wanted to create a system not to be used as a diagnostic test but instead a tool that an individual or doctor could use to accurately measure a person's risk of developing heart disease in their current state. Hopefully, an at-risk user would see that they were likely to develop heart disease if they continue their habits and so it would inspire some change amongst these individuals.

As the prediction is not going to be used in a diagnostic setting, the minimum prediction accuracy that I am comfortable with is 85%. This number gives us a good estimation of risk, although, we must inform the user that whilst the risk of false positives is alarming, the more dangerous outcome is a false negative as the patient may think they aren't at risk when really, they are. This another reason why the tool should only be used in combination with real diagnostic tests to determine a clinically accurate result.

The inputs and outputs of the system should be simple, with the user inputting their BMI, sex, age, smoking status and dietary information and receiving a yes or no output as to whether the system predicts that their current lifestyle and habits will lead to heart disease.

Dataset Analysis

My dataset is a cleaned version of the 2015 BRFSS health survey. This survey is conducted by the Centers for Disease Control and Prevention in the United States. The data from the survey was gathered over the telephone using Random Digit Dialing (RDD) methods. This means that there should be no biases in our dataset as people were selected at random to answer the survey (CDC, 2018). The dataset was originally 253,681 entries long and included many unnecessary columns that would have not benefitted my results. For these reasons I only imported the first 80,000 rows and dropped the columns unrelated to my task.

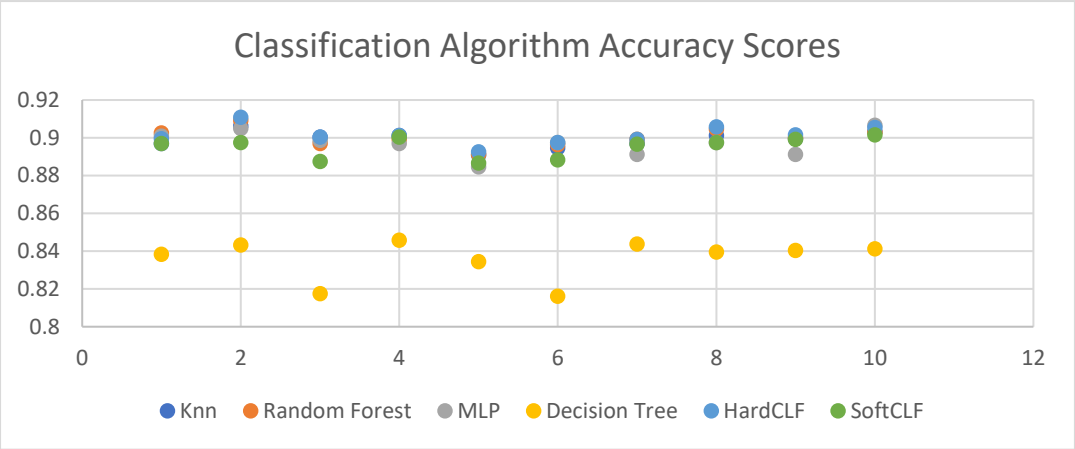
These included: mental health score, difficulty walking, education score, income, physical health score, healthcare status, healthcare price and general health score. Some of these columns such as general health score may sound like they would be useful to my task, however, as these values were either binary or scored out of 15, I found it hard to determine how these could be used to help my classification implementation and so they were dropped.

After dropping those columns, we are left with the following values: heart disease (binary), high blood pressure (binary), high cholesterol (binary), BMI, smoker (binary), previous stroke status (binary), diabetes status (binary), physically active (binary), regularly eating fruits and vegetables (both binary), heavy alcohol consumption (binary), sex (binary), and age. To me, all this information is crucial to my study, and I have used heart disease (binary) as the target attribute.

Algorithm selection

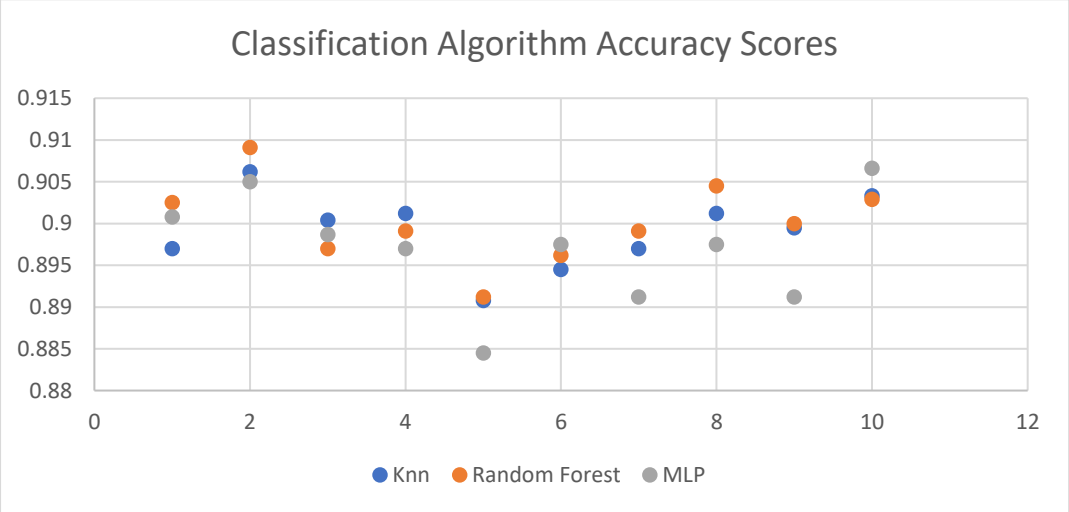
I used a train test split using 70% for training and 30% for testing for all the algorithms that I tested throughout this selection process. This was to get an idea of how the algorithms accuracy scores compared without using k fold validation.

As we were trying to predict a binary value (if the user was likely to develop heart disease or not) I had a good idea of the types of algorithms I wanted to use to get the best results. I tested k-nearest neighbor, random forest, multi-layer perceptron classifier (MLP). I used the accuracy score method on each result to get an accurate train/test accuracy score. I also used soft and hard ensemble voting classifiers to compare the results to and see which classifier had the most impact.



As you can see, there is one clear outlier. The decision tree classifier's accuracy score was consistently lower than all the other results. This could be due to how the decision tree algorithm was interpreting our data. As our data was real-world, it can be hard for the algorithm to interpret, and overfitting is a big issue with the decision tree algorithm, this is where the model applies greater meaning to less important data and can throw off the prediction (Seif, 2018).

For this reason, I removed decision tree from the graph so that we could see a clearer picture of how the other algorithms faired once compared with each other.



As you can see from the graph above, the grouping of accuracy scores from the KNN, random forest and MLP algorithms are very close. Although, on closer inspection we can see that the MLP drops below 89.5% on 30% of its scores. This score, although acceptable in our model, is not as good as KNN or random forest. For this reason, I decided to remove MLP from the shortlist.

Because the KNN and Random Forest algorithm's scores were so close, I decided to make a table to visualize the average accuracy scores of each algorithm over the 10 tests to better determine which I should use as my model.

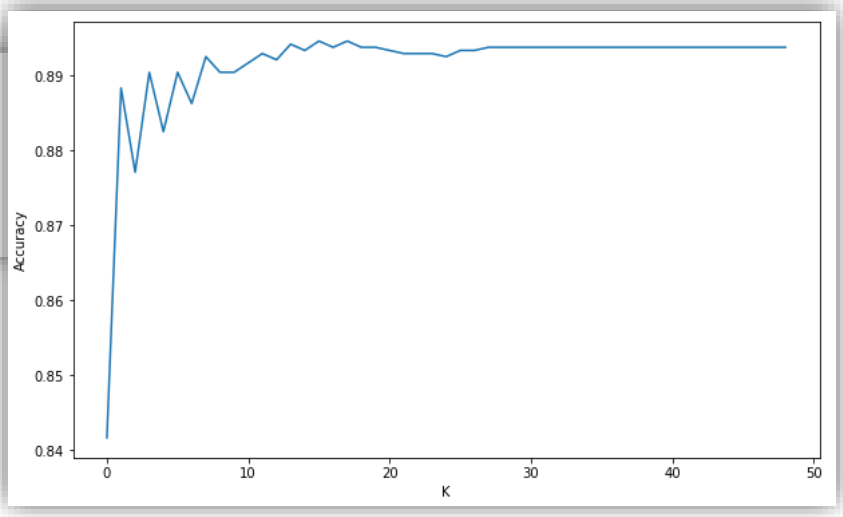
i	KNN	Random Forest	MLP	Decision Tree	HardCLF	SoftCLF
1	0.897	0.9025	0.9008	0.8383	0.8995	0.897
2	0.9062	0.9091	0.905	0.8433	0.9108	0.8975
3	0.9004	0.897	0.8987	0.8175	0.9004	0.8875
4	0.9012	0.8991	0.897	0.8458	0.9012	0.9004
5	0.8908	0.8912	0.8845	0.8345	0.8925	0.8866
6	0.8945	0.8962	0.8975	0.8162	0.8975	0.8883
7	0.897	0.8991	0.8912	0.8437	0.8991	0.8966
8	0.9012	0.9045	0.8975	0.8395	0.9058	0.8975
9	0.8995	0.9	0.8912	0.8404	0.9016	0.8991
10	0.9033	0.9029	0.9066	0.8412	0.9054	0.9016
Average	0.89911	0.90016	0.897	0.83604	0.90138	0.89521

The table above shows the scores over 10 iterations. We can see that between the KNN and random forest algorithms, the random forest had the best average accuracy with just over 90%, KNN coming next best with 89.9%. This is even better than I was hoping for as I initially planned to not accept accuracy scores less than 85%.

Because of this small difference, I wanted to see how much of an effect changing the "n_neighbours" variable had on the accuracy of the KNN algorithm's results. To do this I made a for loop in my program that appended the results of each iteration to an array. I then used matplotlib's pyplot to plot a graph to show how the accuracy changed with each iteration. This graph is pictured below.

```
#KNeighborsClassifier
knnscore = []
for i in range(1,50):

    clf = KNeighborsClassifier(n_neighbors=i)
    clf = clf.fit(X_train, Y_train)
    Ykneighbours_prediction = clf.predict(X_test)
    knnscore.append(accuracy_score(Y_test,Ykneighbours_prediction))
```



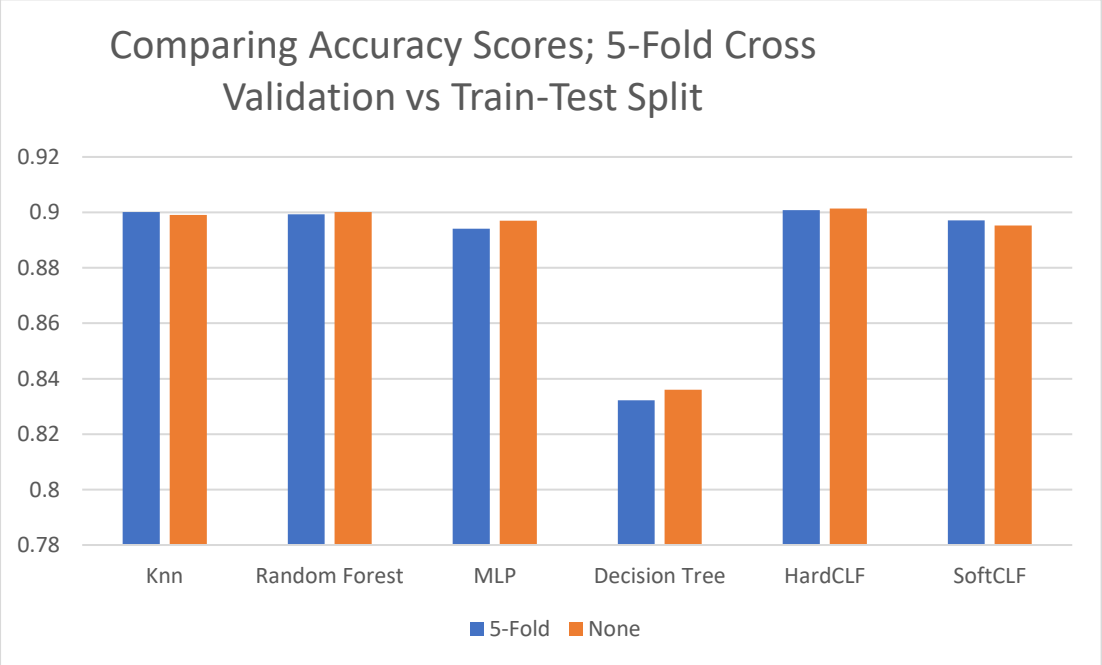
When analyzing the graph, there isn't much of an improvement in accuracy after around 10 (which is what n_neighbours was set to during the testing I have done so far). This tells me that even when adjusting this variable, the random forest algorithm is still the better choice, for this reason if I was to choose only one algorithm for this project, I would choose the random forest classifier.

However, as stated earlier, I have been using some ensemble voting classifiers to see if I could make the accuracy scores even better. And although soft voting seems to have a detrimental effect on the results, the hard voting improves them significantly. By using multiple algorithms paired with the hard voting classifier. We were able to bring the average accuracy score down to 90.1%.

K-Fold Cross Validation Results

Once I had established the best results without cross validation, I wanted to see how it would differ using cross validation. I used 5-Fold cross validation for each algorithm and compared its score to the best average score of the previous tests. I then plotted this bar chart from my data to see the differences (Brownlee, 2018).

	Knn	Random Forest	MLP	Decision Tree	HardCLF	SoftCLF
5-Fold	0.9001	0.8993	0.8941	0.8322	0.9008	0.8971
None	0.8991	0.9001	0.897	0.83604	0.90138	0.89521



As you can see, the differences between using k-fold validation and just using a regular train test split proved minimal, however it showed that the accuracy of the prediction did differ slightly. The HardCLF score was still the best and had the least difference in accuracy between the two tests, so that is the algorithm that I would use for this problem.

References

Brownlee, J., 2018. *A Gentle Introduction to k-fold Cross-Validation.* [Online]
Available at: https://machinelearningmastery.com/k-fold-cross-validation/

CDC, 2018. *BRFSS Frequently Asked Questions (FAQs).* [Online]
Available at: https://www.cdc.gov/brfss/about/brfss_faq.htm

Seif, G., 2018. *A Guide to Decision Trees for Machine Learning and Data Science.* [Online]
Available at: https://towardsdatascience.com/a-guide-to-decision-trees-for-machine-learning-and-data-science-fe2607241956