CP322 Final Project

Connor Kanalec - 180871330
Tharssan Srikanthan - 180398580

April 6th, 2023

Abstract

The purpose of the project was to use the methods described in the course to analyze data. For our project we decided to use a Thyroid disease dataset that had more than 300 cases and 29 variables. With this data we did data pre-processing, split the data, planned models and trained them. The models we chose were KNN and Decision Tree. After testing we got an accuracy score of 98.66% for the decision tree and 97.53% for the KNN model. To summarize our findings, we believe models like this can be used in the medical field to test patients.

1 Introduction to Project

Hyperthyroidism is a common disorder that affects millions of people worldwide. It has many symptoms that include weight loss, anxiety, and a rapid heartbeat. Early detection of the disease is crucial to prevent these symptoms and improve the patients health. Machine learning models can be a valuable tool in predicting cases by analyzing patient data, including thyroid hormone levels, age, gender, and other relevant medical history. In this project, we wanted to create a machine learning model that can predict cases based on the patient data we obtained.

2 Pre-Processing and Exploratory Data Analysis

Once you have your dataset, you need to preprocess it to make it suitable for machine learning. This includes cleaning the data, handling missing values, encoding categorical variables, and scaling numerical features. This step is crucial as the accuracy of your model will depend on the quality of your data.

2.1 Dataset Collection

We were able to obtain our dataset from the database from UCI (University of California Irvine. Our dataset was already split into training data and test data for us. The training data had 2800 instances while the test data had 972 instances. Both had 29 instances. The attributes included

information about the patient like age, and gender. Another medical yes or no question like if they were pregnant, or sick. The last part of the attributes were asking the patient if they had taken a certain test and the attribute beside that was the measurement of the test.

2.2 Data Pre-processing

The first step we took to process the data was to drop any columns that were not needed. This included TBG and TBG measured as this was a test that was not conducted for any patient. We also dropped the referral source which was where they got certain patient data from. We changed column values to their appropriate data types (Ex. Age set to Float). Next, we knew we had a good amount of missing data in our sets. These were represented by "?". We placed all the question marks with NaN. For the continuous attributes we replaced the missing values with their respective column averages. For our "Sex" column we decided to replace the missing values with a new class. We represented this class with "?". Next, we sliced our result column as each value had a unique number at the end. By slicing this column the values can now be grouped into the different classes. For outliers we thought there were some in the test measurements though after some research we concluded these measurements were possible. We had a single outlier in age which we just replaced with the average of the column. Because of this the missing values from the medical tests attributes are now filled. This makes the Yes or No columns asking if the patient took the test or not to be irrelevant. We dropped these columns for this reason. Lastly, we used get_dummies for encoding for the classification values.

3 Methodology

3.1 Introduction to Python for Machine Learning

To build our model we decided to use python.. Python is a great language to create machine learning models. It is easy to learn and use, flexible, has a large set of libraries and scalability to handle large data sets.

3.2 Platform and Machine Configurations Used

The platform we used to create the models was VSCODE.

3.3 Data Split

Data splitting is the process of dividing the data into training, validation, and test sets. This step can help you evaluate the performance of the model on new data.

As mentioned before our data was already split into a training and test set.

3.4 Model Planning

The next step is to select the appropriate machine learning or deep learning models (2 - 3) that can be used to solve the problem. This involves evaluating different models and selecting the one that is best suited for the problem. Based on the nature of our data, we had to choose between models that were capable of performing multi-class classification. Our data had a range of features that we could use to classify patients into one of five classes: negative, T3 toxic, hyperthyroid, goitre, or secondary toxic. That immediately eliminated all regression type models where data output was discrete, not continuous. Also not all classification models on scikit-learn support multi classification which eliminates more models from our pool. According to scikit-learn, that narrowed down our selection to roughly six different models that were classification and supported multi-class classification. From the six models we choose K-nearest Neighbours Classifier and Decision Tree Classifier as these models contrasted each other with

one being a tree based model and the other one was not. We have also had experience using these models before and were knowledgeable of how they worked and their key hyperparameters that could be tuned to get a better result.

3.5 Description of the Models

The first model we used was the Decision Tree Classifier Model. Decision Trees are a tree like model that use piecewise methodology to train itself to data. Decision Trees have a number of hyperparameters a user can tune to modify the model to better fit their data such as - max depth, min samples split, and min samples leaf (Towards Data Science, 2019). Some pros of the Decision Tree Classifier Model are they are easy to interpret, only require basic data pre-processing such as creation of dummy variables and fixing mixing data, and some cons are they do do not generalize as well, especially if hyperparameters are overtuned, leading to overfitting (Scikit Learn, 2023). Our second model called K-Nearest Neighbours classifies data on how close it is related to already classified points. The already classified points are from the train data. When test data needs to be classified, the k most similar data (k being odd) is used to classify the test data point based on which class is most common amongst the nearest neighbors. Some pros of the K-Nearest Neighbours Classifier is that there really isn't a training process, as mentioned before, the train data is used as nearest neighbors to classify new data. This saves computation time as no training is needed. Another pro of this model is it is easy to understand and interpret. Some cons of this model include that in very large datasets the computation cost could grow large, as distances must be calculated for every new data point. And a second con is data with lots of features makes the data less easy to interpret as distance in multidimensional data is harder to interpret (Medium, 2020).

3.6 Model Training and Optimization:

Our model training was relatively straightforward but did take some extra steps to optimize the model. For the Decision Tree Classifier model we choose to set a random state, so that when we trained our model and then tested the model, the model performance would be the same, this is called a pure function. Pure functions are helpful because when inputs are constant, outputs will also be constant, so you can accurately judge the performance of the model. Then we could compare with trial and error, tweaking the parameters of the model to see if we could get better performance on our test data. So we set our Decision Tree Classifier to a random state, so it was pure, and then we used hyperparameter tuning on the depth of the tree with cross fold validation of 10 to optimize the tree's depth. This hyperparam tuning proved to significantly increase the performance metrics of the model especially in the weighted average of the performance metrics across all class outputs. Our model training for K-Nearest Neighbours Classifier was similar, but we did not have to set a random state for it. We did use hyperparameter tuning for this model as well, and we chose to tune the k value, the number of nearest neighbors used in classifying a datapoint. We also used cross fold validation of size 10 in hyperparameter tuning to ensure that the model was testing on enough folds to find a close to optimal, if not optimal, value. Similarly to the Decision Tree Classifier, we trained the model multiple times and checked performance metrics, and continued to play around with the models parameters using trial and error to improve our performance.

3.6 Model Evaluation

Once the model is trained, the next step is to evaluate its performance on a validation dataset. We evaluated our two models on the same metrics: accuracy, precision, recall, and f1 score. Furthermore, we took those metrics for every class - negative, t3 toxic, goitre, hyperthyroid, and secondary toxic, and took a weighted average and a macro average of them. The weighted

average was weighted so that the negative which had 947/972 test cases had the highest weight as it appeared the most in data, which makes sense. The macro average however, evenly weighted each of the classes' performance giving hyperthyroid, for example, as equal weight to negative even though it only appeared 17/972 times in the test data. This was insightful though because the weighted average heavily relied on the success or unsuccessful of the prediction of negative cases which occurred by far the most, but the macro average could give us some insight on how the models predicted the remaining 4 classes which totalled out to the remaining 25 cases in the test data. This was also important because these were the cases when patients actually had a medical problem. Both models were similar in terms of overall accuracy with Decision Tree Classifier having a score of 98.7% and K-Nearest Neighbors having a score of 97.5%. So both models were extremely good in these scores. Where the Decision Tree Classifier performed much better was in identifying the classes that appeared less often in the data sets (every class other than negative). Because of this, Decision Tree Classifier was our best model and more details will be talked about in the results section of the report.

Results

4.4 Interpretation of the Results

Once the model is optimized, the final step is to fit the model and report final test error for each model Present the information with support Screen Shots/ Figures (Each Figure must be numbered and Description of Figure must be provided)

Since our data was multi-class classification we chose to use K-Nearest Neighbours and Decision Tree Models. Patients could be classified into 1 of 5 classes - 0: T3 Toxic, 1: Goitre, 2: Hyperthyroid, 3: Negative, 4: Secondary Toxic. Unfortunately, the train data had no cases of

Secondary Toxic and the test data only had 1 case. Due to this, our model was never trained on instances of secondary toxic cases and therefore, never would predict a secondary toxic case. This wasn't a major issue for our performance scores, as it was the least common classification out of all 5 classes, and only appeared once in train and test combined.

We trained both K-Nearest Neighbours and Decision Tree Models using Hyperparam tuning to find the ideal parameters for each model, in order to get the best performance metrics. We found that k=3 neighbors was optimal and decision tree depth = 4 was optimal.
In comparing the models, both models were similar in terms of overall accuracy, but Decision Tree was substantially superior in terms of metrics such as precision, recall, and f1 for individual class predictions.

Overall accuracy was heavily influenced by accuracy of negative predictions as negative cases were 947 out of 972 patients in y_test. Both models had extremely high accuracy due to their very good ability to predict negative cases (knn = 97.5 percent accuracy and decision tree = 98.6 percent accuracy) Where the Decision Tree Model was substantially better than KNN was for Goitre and Hyperthyroid class predictions. These 2 classes were the 2nd and 3rd most occuring class behind Negative at 17/972 instances in y_test and 5/972 instances in y_test (respectively). The Decision Tree had high scores for such low instances of these classes was impressive in our minds as it was trained on low instances. The Decision Tree has a precision score of 100 percent for the 5 cases of Goitre, meaning it had no false positives, and whenever it predicted positive, it was true! The Decision Tree has a precision score of 80% percent for the 17 cases of Hyperthyroid, which is very solid for a low amount of instances. Once again, overall, both models had extremely high accuracy scores of 97.5 percent the, meaning they overall were

both great and had a f1 score of 99 percent for Negative cases which means that patients are positively predicted negatively very accurately and the coverage of all negative cases is high.

The Decision Tree was overall better because when patients actually had a problem such as Goitre or Hyperthyroid, the Decision Tree model was better at predicting so. This is why we decided to show the macro avg, since Negative cases are heavily weighted in the Weighted Average since they are 947 out of 972 patients in y_test, we used Macro Avg to show how well the models predicted the other classes such as Goitre, Hyperthyroid, Secondary Toxic, and T3 Toxic.The Macro avg was better in Decision Tree since it was better at predicting Goitre and Hyperthyroid. But neither models could predict the 1 case of Secondary Toxic as it was never present in the train data. And neither model predicted the 2 cases of T3 Toxic.

4.2 Performance Metrics

We used model accuracy, f1 score, recall, and precision score to compare our models.

4.3 Results Table

See Appendix 1 for the results table for the models

4.5 Visualization

See appendix 2-5 for the visualization to help to the difference between the models.

5 Conclusion

In conclusion, the hyperthyroidism machine learning model we created was successful at finding patterns and predicting positive cases. These results prove that the use of machine learning in this field has the potential to aid health care workers and potentially better the lives for thousands who suffer from this disease. For future models, it would be important to test out different techniques to predict cases. Also, there may be other factors that can help predict positive cases. Research on this could lead to more complex though efficient models.

References

1.12. multiclass and multioutput algorithms. scikit. (n.d.). Retrieved April 6, 2023, from https://scikit-learn.org/stable/modules/multiclass.html

Mithrakumar, M. (2019, November 12). *How to tune a decision tree?* Medium. Retrieved April 6, 2023, from https://towardsdatascience.com/how-to-tune-a-decision-tree-f03721801680

1.10. decision trees. scikit. (n.d.). Retrieved April 6, 2023, from https://scikit-learn.org/stable/modules/tree.html

Soni, A. (2020, July 3). *Advantages and disadvantages of KNN*. Medium. Retrieved April 6, 2023, from https://medium.com/@anuuz.soni/advantages-and-disadvantages-of-knn-ee06599b9336#:~:text=Easy%20to%20implement%20as%20only,then%20performance%20is%20very%20low.

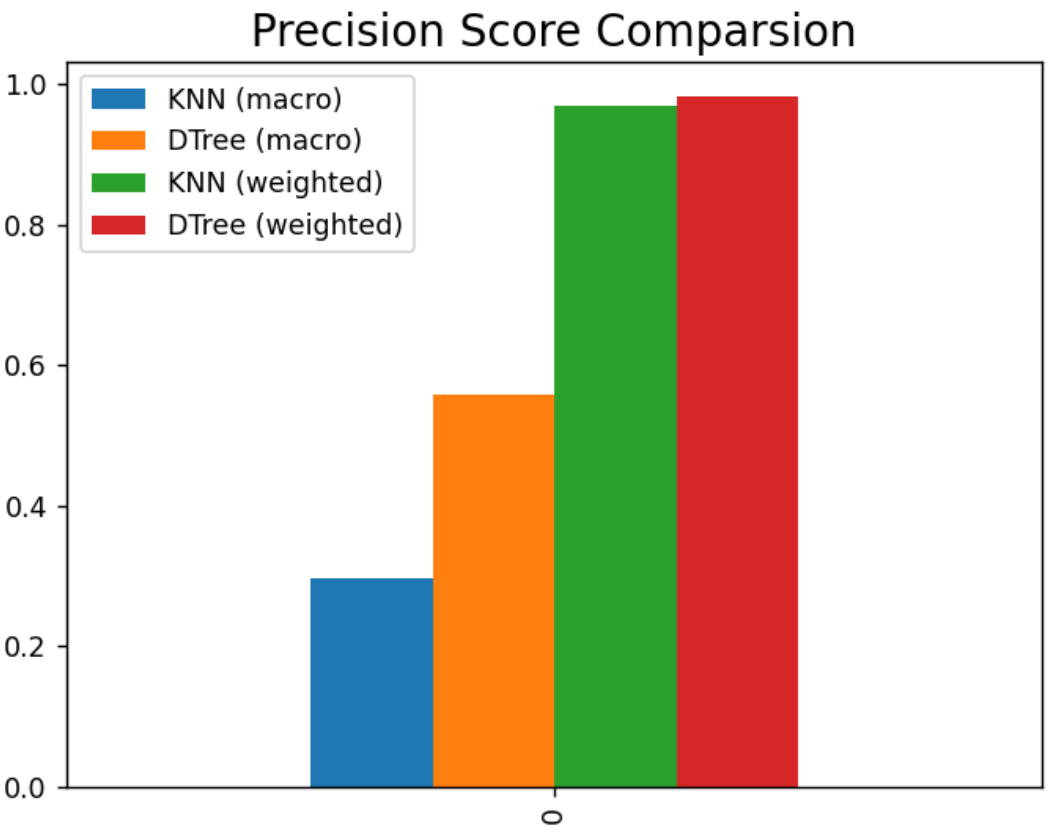
Appendices

Appendix 1

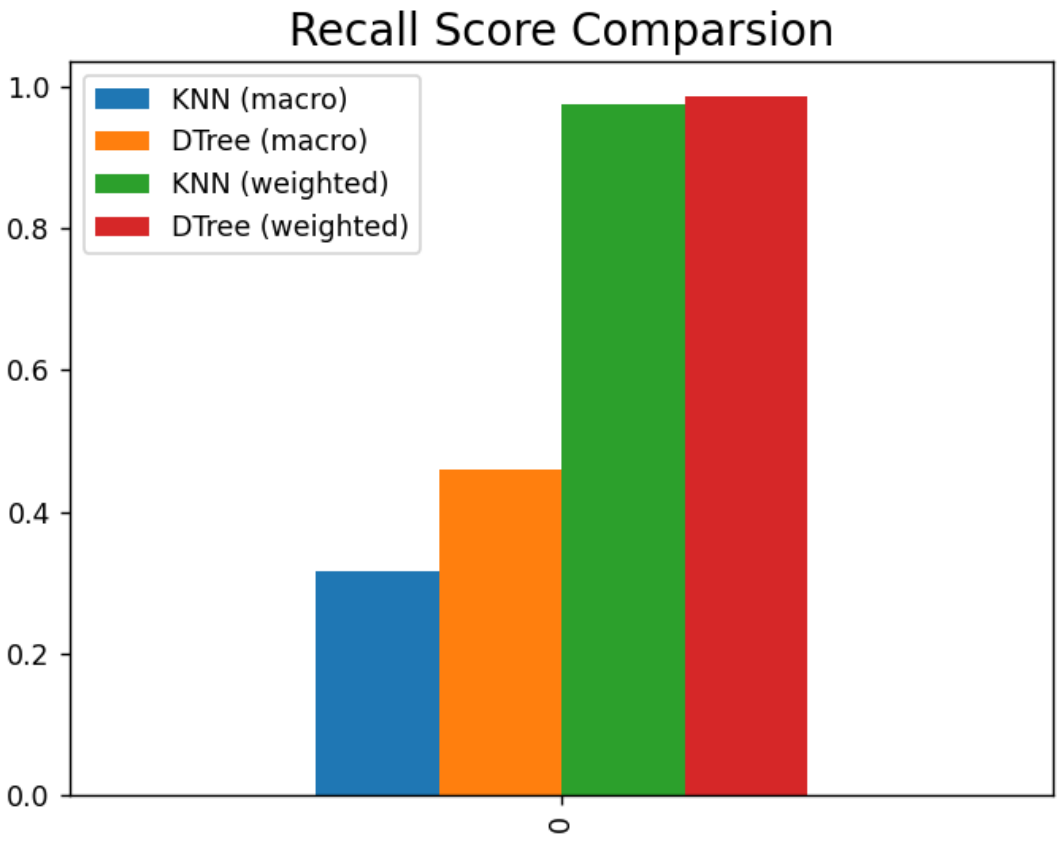
```
The accuracy achieved by decision tree model: 0.9866255144032922
################################################################
The classification report of decision tree model:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         2
           1       1.00      0.60      0.75         5
           2       0.80      0.71      0.75        17
           3       0.99      1.00      0.99       947
           4       0.00      0.00      0.00         1

   micro avg       0.99      0.99      0.99       972
   macro avg       0.56      0.46      0.50       972
weighted avg       0.98      0.99      0.98       972
 samples avg       0.99      0.99      0.99       972
```

```
The accuracy achieved by knn model: 0.9753086419753086
##################################################################
The classification report of knn model:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         2
           1       0.00      0.00      0.00         5
           2       0.50      0.59      0.54        17
           3       0.99      0.99      0.99       947
           4       0.00      0.00      0.00         1

   micro avg       0.98      0.98      0.98       972
   macro avg       0.30      0.32      0.31       972
weighted avg       0.97      0.98      0.97       972
 samples avg       0.98      0.98      0.98       972
```

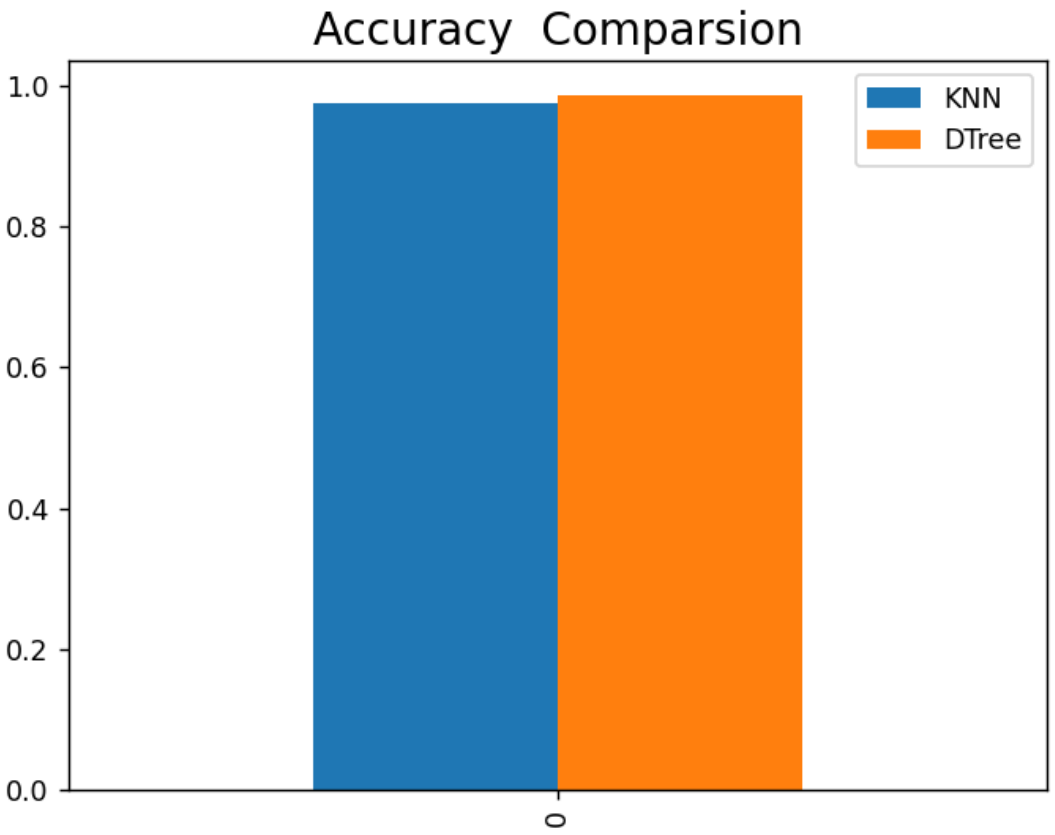
Appendix 2


Precision Score Comparsion
KNN (macro)
DTree (macro)
KNN (weighted)
DTree (weighted)
1.0
0.8
0.6
0.4
0.2
0.0
0

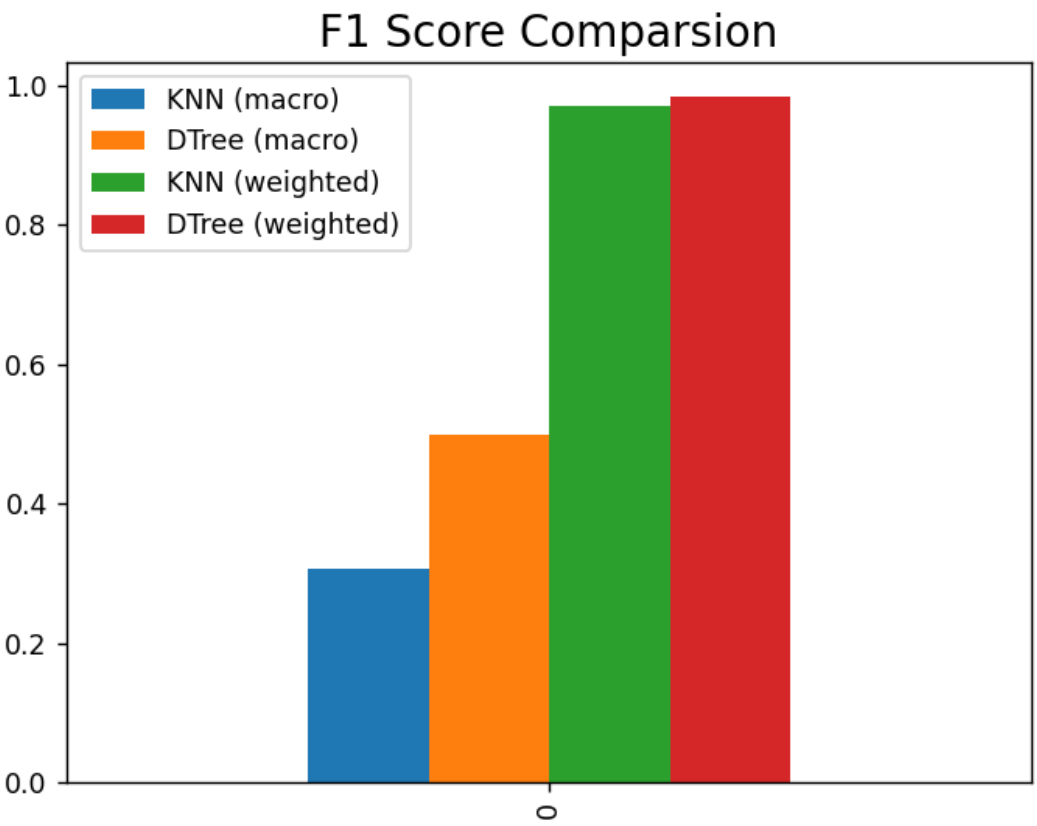
Appendix 3


Recall Score Comparsion
KNN (macro)
DTree (macro)
KNN (weighted)
DTree (weighted)
1.0
0.8
0.6
0.4
0.2
0.0
0

Appendix 4



Appendix 5


F1 Score Comparsion
KNN (macro)
DTree (macro)
KNN (weighted)
DTree (weighted)
1.0
0.8
0.6
0.4
0.2
0.0
0