Tytonidae Tympanometry

Applying Machine Learning to predict hearing loss using wideband tympanometry

Di Yao
22795234

Karan Rebello
22868277

Anitha Raghupathy
22773933

Cheng Nian
23053313

Aminul Islam
22884375

Daniel Chegwidden
21282744

October 2021

https://github.com/danielchegwidden/tytonidae-tympanometry

Contents

1 Tytonidae Tympanometry

The Tytonidae Tympaometry (TyTy) project was born out of a need for audiologists to better understand wideband tympanometry (WBT). WBT data is used to classify ears as either normal, or with conductive conditions such as Otitis Media. The subjects in this data are children, and not identifying conductive conditions has consequences throughout their life. The societal cost of these conditions is estimated at $20,000 across 10 years as children struggle to learn and engage at school, amongst other effects.

TyTy attempts to use WBT data to improve the classification accuracy of the current receiver operating characteristic (ROC) analysis that is performed, as well as provide audiologists with insights as to how to better analyse the large quantities of data that is being generated. If this is achieved, then clinicians are able to spend less time analysing data, and more time interacting with patients, resulting in more patients being seen and better care being delivered.

2 Models. Models. Models.

The Machine Learning approach that attempted to beat the ROC results involved Logistic Regression, Support Vector Machines, Decision Trees, Random Forests, and K-Nearest Neighbour as the modelling approaches taken. These were selected as having good classification applications, as well as a variety of complexity and interpretability implications.

The data was transformed through processing pipelines to ensure consistency and reproduceability. For each model, the appropriate pipeline could be used with different parameters to see how these transformations affected the final results. An example of this is pressure matching, where the full data was filtered down to a single observation per ear based on the pressure reading when the data was collected.

Logistic Regression was chosen as the base model and the starting point due to its simplicity and interpretability. This model creates a formula that outputs either a positive or negative class depending on the cutoff probability. For all of the different pipeline transformations, this model gave a baseline to beat using the same data. The ROC results that are currently being used may have slightly different transformations and it was important to be comparing like-for-like results. Tuning of the model hyperparameters resulted in 85% accuracy on the unseen test set, and this was what the more complex models attempt to beat.

One step up in complexity from Logistic Regression is Support Vector Machines (SVM). This model attempts to find a decision boundary, or hyperplane, that separates the data points into the positive and negative class. SVM works well with small and clean data which is what the WBT data is like, and can deal with the many frequencies present whilst maintaining its interpretability. The best results from the SVM model was 89% accuracy on the test set.

The next modelling approach explored was Decision Trees, which identify important features and the best split of the data at each of these features. This results in a binary decision at each node of the tree that splits into two leaf nodes where the process is repeated. The results of the training process show this in Figure 1 where the blue nodes how a positive class prediction, and the orange nodes show a negative one. The Decision Tree model is interpretable as the tree can be followed from the root node down to a specific leaf node to identify which features are contributing to the prediction. The colouring also indicates the confidence of the prediction with the darker colours indicating the degree of pureness in the node. The best results from the Decision Tree was 90% on the test set, which jumps up to 93% when using cross-validation.

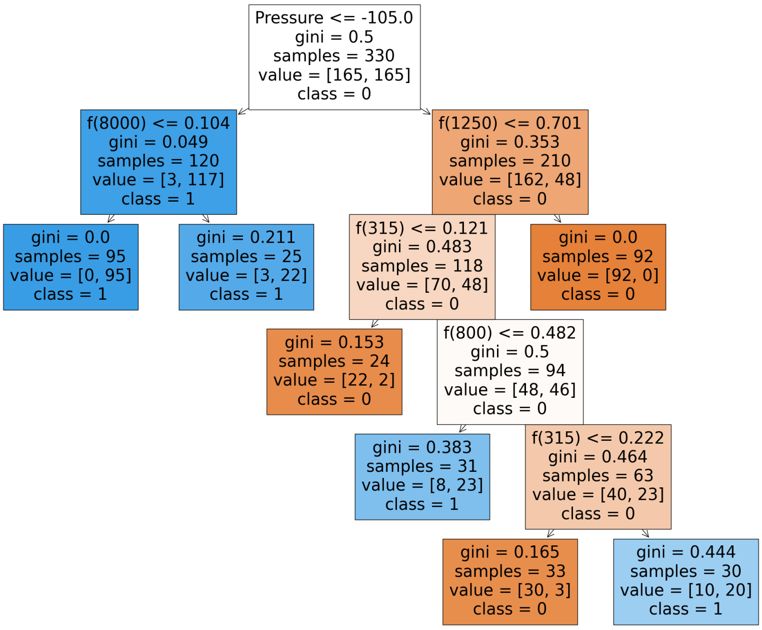

Figure 1: The best Decision Tree model after being trained

Building on from Decision Trees, Random Forests are ensemble of many of these trees grouped together and then averaging out their individual results. This takes longer to run than a single tree, but uses a wisdom-of-the-crowd approach to reduce the variance from the forest. The downside of this approach is that the interpretablity drops due to the averaging across many trees, which means that the results or a specific tree are both inaccessible and irrelevant to the performance of the overall model. The performance of the Random Forest on the test set was 89%.

The final model considered was K-Nearest Neighbours, which is an unsupervised technique that attempts to group similar data points together. The number of groups that this model uses is the hyperparameter k, which results showed to be 116 in the optimal model, leading to an accuracy of 89%. However this does not make much sense as ideally k would be 2, with one group for positive patients and another for negative.

3 Finding Number 1

For the models that were explored, the best results for each are displayed in Table 1. This shows that the more complex models performed better than the baseline set by Logistic Regression, as well as that breaking above the 90% accuracy was a difficult challenge that only the Decision Tree model achieved.

Model	Accuracy (%)
Logistic Regression	85
Support Vector Machine	89
Decision Tree	93
Random Forest	89
K-Nearest Neighbour	89

Table 1: Summary of the different model performance

The other metric of interest in addition to accuracy is sensitivity. This measures how many children with conductive conditions are correctly identified as part of the positive class. The higher the sensitivity, the less positive patients that slip through the gaps. For the Decision Tree with the best accuracy, its sensitivity is 87%, which means it only classifies 13% of positive patients as negatives, or false negatives. Both the accuracy and the sensitivity are important metrics to consider for the final model.

The accuracy results in Table 1 are not the only consideration for the best model, the interpretability is a major component as well. This means that the model not only needs to perform well, it must provide insights into why it makes a prediction, or what data is important to consider when making a prediction. Because of this requirement, both the Random Forest and K-Nearest Neighbour models were eliminated from contention. The both obscure why they are predicting a certain outcome which do not allow the audiology clinicians to improve their understanding of what data is important to be analysing. Logistic Regression was eliminated too due to being the lowest performer amongst the interpretable models.

TO ADD: Strengths/Limitations + Explainability

4 Looking Ahead

5 Contribution

To Add: Gantt Charts

Member	Number	Project Tasks	Skills
Di Yao	22795234	1. Research Wideband Absorbance Data and scope the project 2. Review code and approve changes using Pull Requests 3. Perform modelling using K-Nearest Neighbour and then Random Forest	Python Machine Learning Git and GitHub
Karan Rebello	22868277	1. Research Wideband Absorbance Data and scope the project 2. Review code and approve changes using Pull Requests 3. Perform modelling using Random Forest	Python Machine Learning
Anitha Raghupathy	22773933	1. Research Wideband Absorbance Data and scope the project 2. Review code and approve changes using Pull Requests 3. Perform modelling using Support Vector Machines	Python Machine Learning
Cheng Nian	23053313	1. Data Cleaning and Transformations including pressure-matching function 2. Review code and approve changes using Pull Requests 3. Perform modelling using Support Vector Machines	Python Machine Learning
Aminul Islam	22884375	1. Data Cleaning and Transformations 2. Perform modelling using Logistic Regression	Python Machine Learning

Daniel Chegwidden	21282744	1. Create GitHub Repository and set up applicable automation and structure around Commits and Pull Requests 2. Manage code integration to ensure a workable code base was maintained and applicable functions and structures were in place to support consistent and reproduceable analysis 3. Review code and approve changes using Pull Requests 4. Perform modelling using Decision Tree	Team Leadership Meeting Organisation Python Machine Learning Git and GitHub Software Engineering