Final Project Report

Abstract

The heart, responsible for the pumping of blood and nutrients throughout the cardiovascular system, relies on electrical signals which can be measured and quantified. Many cardiovascular diseases result from, or are caused by, issues and irregularities in how the heart pumps blood. These irregularities can often be detected in the electrical signals captured by the electrocardiogram (ECG). We sought to construct machine learning models, based on ECG data and metadata, to help predict when a patient had a certain disease. The metadata and ECG data was provided by Physionet[1], to promote this type of investigation. The data was explored for duplicates, outliers, and missing values. Descriptive statistics were then generated. Feature engineering was then performed to calculate various relevant components of the QRST complex, P wave, and T wave. After splitting the data into a training and test set, various models were built from the training data set. These models were then evaluated against the test data set and model accuracy, precision, recall and f1 measures were reviewed. After reviewing the results of various models, it was decided to focus on creating a model to predict whether a patient had a disease or not (NORM vs not-NORM). It was determined that the various models performed similarly to each other, ranging from 0.63-0.68 for accuracy, 0.65-0.71 for precision, 0.09-0.48 for recall, and 0.16-0.57 for f1_measure. Overall, using logistic regression with clinical predictors, sex and age, with a subset of feature engineering predictors, BPM, BIF, and BIF2 provides optimal performance. This finding suggests that while it is possible to build a predictive model from the relevant data, more work is required to achieve higher levels of accuracy, precision, recall and f1measure.

Data and Methods

All data was provided by Physionet for the purpose of developing machine learning models for the interpretation of ECG readings. In addition to the ECG data, metadata was also provided. This data was composed of patient demographics, likelihoods for disease states, annotations, and information about the tools and personnel performing the ECG. In total, there were 34 separate features in the metadata. In terms of medical data, the dataset contains 21837 ECG measurements from 18885 patients. Each ECG comprised 12 clinical leads and was 10 seconds in length, resulting in 1000 data points per lead when

sampled at 100Hz, and 5000 data points when sampled at 500Hz. The analysis explored in this paper uses the 100HZ dataset. The 12 clinical leads are required for capturing different graphical views of the heart's electrical signals. Based on its positioning relative to the actual flow of electricity through the heart, Lead II is most commonly used for monitoring of the ECG. It is also the lead where the P wave, T wave, and QRS complex are most visible. Since our analysis is exploratory in nature, we opted for investigating lead 12 initially, due to the cleanliness of the data, and the lack of noise compared to lead II. After extensive testing and training on feature engineering, we discovered that some ecg_id's have missing ecg traces on the last two leads. Since we do not have labels on the leads, we were unable to train our machine learning program to select the best lead for feature extraction. Eventually, we used the ECG traces from lead II for our feature engineering since in principle it provides the most physiologically relevant data.

Summary Statistics

The dataset was first loaded onto a parallel computing network, where the Pyspark language could be used to perform analysis. After loading, a function (provided by Physionet) was altered and then used to assign probabilities of disease state for each ECG instance. This problem statement is multilabel in nature since at the time of an ECG, a patient could be at risk for various cardiovascular diseases at the same time. If the medical staff considered the patient to have a 50% probability of disease state or greater, the patient was assigned a label for that disease state. A patient could be labelled for up to 5 disease states; "NORM" for normal, "MI" for myocardial infarction, "STTC", for ST/T change, "CD" for conduction disturbance, and "HYP" for hypertrophy. The counts in Table A match the count in Physionet.

NORM	9528
MI	5486
STTC	5250
CD	4907
HYP	2655

Table A: counts by disease label

Clinically relevant features were also engineered from the ECG data. After feature engineering was performed (see below), correlations were calculated to see if any of the engineered features or metadata features were correlated with one another. It was discovered that only three cases had a correlation above 0.5 (height and weight, QR interval and PR interval, and the PRratio and TRratio). Though these features were somewhat correlated, each of them was considered important due their potential in identifying a disease state.

Data Cleaning

Data cleaning was then performed on both the metadata as well as the clinical ECG data. No duplicates were found across the 21837 observations. Each feature (eg: weight, height, age, sex, etc) was then checked for missing observations. Certain metadata features were found to have fairly large (greater than 50%) of their data missing. In these cases, the features are not included in the modeling.

After feature engineering was performed, the numeric data (from the metadata and engineered features) was explored for outliers. There were outliers discovered across all of these features. However, the decision was made not to exclude any outliers from the model building process, as a strong outlier could actually denote a disease state, and capturing that state would be important for the model. As will be noted in the feature engineering section, there was no curse of dimensionality present, due to the number of variables used, so it was not necessary to perform PCA.

Visualization

Given the unstructured nature of the ECG data, visualization was a necessary task to understand what an ECG looked like for a given disease state, as well as how each lead related to one another. Using the WFDB package, the waveform chart of each of the 12 leads produced by the ECG test on each patient were created and compiled to a PDF (Appendix B). The visualization of the ECG allowed for troubleshooting errors in engineering the features.

One such error was the instance of an inverted T wave. When trying to find the distance between the S wave and the T wave, there were samples where the T wave would not be registered. Visualizations of the waveform indicated that an inverted T wave was the issue. This was an important discovery to make, as an inverted T wave is indicative of heart problems and therefore may be useful as a feature.

Appendix B illustrates the variety of ECGs found in this dataset. B.1 indicates a ‘normal’ ECG, while B.7 is a detailed look at what the aforementioned inverted T wave may look like. An example of a ‘noisy’ ECG is also provided, as well as an ECG with a high heart rate.

Feature Engineering

The ECG data that was provided by Physionet must be converted from its raw form into relevant features for the model, meaning that features were extracted from the unstructured data. Before the features can be

extracted however, the data must be filtered for any noise present. Additionally, due to the various disease states, it is vital to ensure proper calculation of the R peak locations, as other features can be calculated based on that location. For example, heart rate can be calculated as the distance between the R peaks. Various methods were explored to calculate the R peak location. We started off with the find_peaks function in scipy.signal library. With a combination of defining the prominence and locating the best lead from one ecg_id, we were able to get reasonable readings of bpm (beats-per-minute) values. However, a closer inspection always revealed inconsistent readout in comparison to our common sense. One difficulty was to select the best lead for heart beat identification, and the lack of heartbeat labels obviously made training of such a function unattainable. We had to rely on our untrained eyes to decide whether such a function performed adequately, and whether the final bpm values were within acceptable range for normal human beings. For example, we deemed the number of heart beats within 10 second less than 5, or more than 35 unacceptable and they had to be trained again with an ecg trace from a different lead. Although our bpm values got more accurate with our ever-more-elaborate training functions, we still struggled to locate the most accurate bpm values. Part of the problem was that we need to employ different leads to locate correct heartbeats for different ecg_id's.

Eventually, the biosppy package was selected due to its powerful ability to both filter the noise from the data and accurately calculate the location of the R peaks. The package was also tested to determine if it could properly handle edge cases, such as noisy data, ECG's with relatively low amplitude R peaks, abnormal ECG's (for example, a peak being inverted), irregular amplitude and duration of R peaks, and high heart rate. In most cases, the package performed appropriately on the most physiologically relevant ecg trace from lead II, and examples can be found in Appendix C. A word of caution here: we still found incorrectly identified R peaks with this function, however the rate of such wrong identification was much less frequent in comparison to our earlier attempts. The identification of all R peaks immediately paved the way for us to extract two features of beat irregularity factors: bif and bif2, for the range, and the standard deviation of heartbeat intervals, respectively.

To identify and differentiate various disease states, identifying PQRST complexes is important. We spent a considerable amount of effort to extract these features. The final extraction of 5 features were

accomplished with the combination of a baseline correction routine that was accomplished with an algorithm of asymmetric least squares smoothing, a numpy split function with known R-peak indices, and aforementioned find_peaks functions with a various combination of prominence and width. We call these five features TR_interval, TR_ratio, PR_interval, PR_ratio, and QR_interval, as displayed in our powerpoint presentation. In our attempt to mimic physiological features of a PQRST complex, we envision our interval values are analogous to pathologically important values of QT interval, PR interval, and QRS complex. Although our function was able to generate five features for all ecg_id's, we have to recognize that our function has its shortcomings. First of all, for ecg traces with poor signal-to-noise ratios, wrong P or T peaks will be identified. More importantly, one important measure of the disease state STTC is a depressed ST peak, however our function was not designed to identify negative T peaks. These obvious drawbacks inevitably affect the performance of our machine learning models, especially in predicting disease states.

Cross Validation

The benchmark model was performed on the whole dataset in classifying NORM or not-NORM produced an accuracy of 56.3% on the training data, this number is basically the percentage of NORM cases among all patients. We moved on to train a number of models with train/test splitting, and we achieved around 70% for both precision and accuracy on test data.

In order to validate our machine learning models, to train hyperparameters of various models, and to prevent over-training, we opted to perform cross validation. For this particular dataset, the host website, physionet, advised users to keep stratfold 9 and 10 for validation and testing since these two datasets were inspected by at least two physicians and were deemed more accurate in terms of labels. We debated briefly about the merits of keeping these more accurate data for testing, whereas employing less accurate data for training. One principle of machine learning is that the model accuracy cannot be better than the accuracy of labels. Although with this principle in mind, we obliged with the website's suggestion, and only trained our model with stratfolds 1-8, and reserved data in stratfolds 9-10 for testing.

The CrossValidator function in the pyspark ml package was employed in our cross validation. In order to consistently tune our models, we built a pipeline of three stages consisting of VectorAssembler,

StandardScaler, and a desired model. CrossValidator was fed by this pipeline along with a sequence of comparable functions and parameters. For all cross validation, we opted for a 10-fold, and employed the same BinaryClassificationEvaluator. Different parameter grids consisting of two hyperparameters were employed for different machine learning models. Since some models took longer time to train, we decreased the parameter steps to speed up our model training. It is important to point out that BinaryClassificationEvaluator employs AUROC values to determine the best model, and this metric may not always be consistent with the best accuracy values.

Results

Initial model exploration

As previously noted, a patient could be classified with one or more disease states: NORM, MI, STTC, CD, and HYP. Various models were trained and tested to classify each of these disease states. The models created were logistic regression, random forest, gradient boosted trees, and linear support vector machines. Predictors both included in the dataset and derived using feature engineering were retained for model training. The predictors included from the dataset are sex and age. The predictors generated from feature engineering and implemented into the model training were beats per minutes, bif, bif2, TRinterval, TRratio,PRinterval, PRratio, and QRinterval,. All the features were scaled using each predictor's standard deviation. Based on the dataset's documentation we reserved stratfolds 9 and 10 for testing due to physicians' recommendations. All generated results and referenced tables can be found in Appendix A.

Various models were generated using a binary outcome of normal versus disease state, MI versus not MI, STTC versus not STTC, CD versus not CD, and HYP versus not HYP. The response variable was transformed for each of these models to reflect the binary outcome. For the logistic regression model, the maximum iteration taken for the solvers to converge was 10. The model with the highest accuracy classified whether a patient was diagnosed with class "HYP" or not and had an accuracy of 87.8%. The other models had an accuracy between 59% to 77%. A summary table between the logistic regression models and various parameters is summarized in Table 1. For some values, the precision could not be calculated based on the lack of true positive and false positive values.

A decision tree model was also trained using a maximum depth of 3 and tested on the dataset. Overall, the model with the highest accuracy classified whether a patient was diagnosed with HYP or not and had an accuracy of about 87.8%. This model accuracy was comparable to the previous best performing model. Overall, the lower range of accuracy for the remaining models was higher than the logistic regression model and ranged between 66.7% to 77.3%. We also note an improvement for the recall and f-measure of the disease state NORM from the logistic regression model. A summary table describing the performance metrics is summarized in Table 2.

A random forest model was implemented to explore any potential improvements when classifying NORM and STTC disease states. Overall, a maximum depth of 10 and 10 trees were used to train and test the model. Based on accuracy, the random forest model was better suited to classify the STTC disease state. However, when comparing all performance metrics (accuracy, precision, recall, and f-measure), this model was more capable of classifying the normal state. A summary table describing the performance metrics is summarized in Table 3.

A gradient boosted tree model was implemented to explore any potential improvements when classifying NORM disease states. A maximum depth of 10 and 10 maximum iterations was used to create the model. Overall, we found the performance metrics were comparable to the previous random forest model. A summary table describing the performance metrics is summarized in Table 4.

Finally, a linear SVM model was implemented to explore any potential improvements when classifying NORM disease states. A maximum iteration of 20 and regParam of 0.001 was used to implement the model. These model performance metrics were also comparable to the previous gradient boosted tree model and random forest model. This model proved to have no significant improvements to other models. In fact, the recall and f-measure for this model decreased in comparison the gradient-boosted and random forest model. Table 5 describes the performance metrics is summarized in Table 5.

Classification of normal versus diseased states

Based on the poor performance metrics, we decided to generate models with only the outcomes NORM versus not. For the following models, we performed cross-validation to select the hyperparameters with optimal performance. For logistic regression, we tested the performance with varying elasticNet and

regParam values. Overall, we find the optimal parameters for the logistic regression is a regParam value of 0.01 and elasticNetParam of 0.4. Overall, the model performance metrics are comparable to previous models created. However, in comparison to the logistic regression, the model's accuracy decreased and the precision increased. The ROC was found to be 0.759. A similar model was trained for the outcome MI and not. For this outcome, the optimal hyperparameter elasticNet of 0.01 and elasticNetParam of 0.6. The model contained a comparable accuracy as the previous exploratory logistic model. However, in the previous model we were not able to calculate precision. In this case, we find the precision was 0.694. The ROC for the model was found to be 0.759. Performance metrics for both logistic regression models can be found in Table 6.

The random forest, gradient boost tree, SVM models were implemented for the NORM versus not disease state. Overall, the model with the highest model performance according to the ROC value was the logistic regression. The model with the highest accuracy was random forest. A summary table describing the performance metrics is summarized in Tables 7-9.

Classification of normal versus diseased states - subsection of variables

We wanted to test whether including a smaller selection of variables would increase performance. Despite the noise in the data and in conjunction with the BioSPPY package, we were able to calculate R-peaks fairly accurately. Therefore, we decided to include clinical factors that had a low percentage of missing observations (age and sex) and features derived from R-peaks (bpm, bif, and bif2). Feature vectors were scaled prior to model training.

Logistic regression, random forest models, gradient-boosted trees, and SVM models were implemented using the subset of variables. The models' hyperparameters were tuned using cross-validation. Overall, all the models contained an ROC around 0.759 and other comparable accuracy, precision, and f-measures. The model with the worst f-measure was the SVM model. We do note that the exclusion of additional derived features such as the TR-interval and PR-interval increased the performance of the model across some of the statistical measures. A summary table describing the performance metrics is summarized in Table 10. Based on the hyperparameter tuning, the optimal model using a subset of variables is logistic regression. Overall, this model contains the highest accuracy, recall, and f-measure in comparison to all

other models. Table 11 compares the metrics between the two top performing models for the two separate sets of features used.

Model	Features	Optimal Hyper-parameters	Accuracy	Precision	Recall	F-Measure	ROC
Logistic Regression elasticNetParam = [0, 0.2, 0.4, 0.6, 0.8, 1.0] regParam = [0, 0.01, 0.05, 1, 2, 5]	Sex, age, bpm, bif, bif2	regParam = 0.01 elasticNetParam =0.6	0.681	0.696	0.484	0.571	0.7591
Random Forest maxDepth = [4, 7, 10, 12] numTrees = [10,100,200]	Sex, age, all derived features	maxDepth = 7 numTrees = 200	0.687	0.711	0.477	0.571	0.7581

Table 11: Summary Table between best models from each set of features

Conclusions

When using all the clinical predictors, sex and age, and all the feature engineered predictors, the optimal model is random forest. Logistic regression contains the highest recall and ROC, whereas random forest contains the highest accuracy and recall. When comparing performance metrics between random forest and logistic regression, random forest contains the higher precision.

Based on the hyperparameter tuning, the optimal model using a subset of variables is logistic regression. Overall, this model contains the highest accuracy, recall, and f-measure in comparison to all other models. The precision and ROC is also comparable to other models. It is worthy to note that the differences between the other performance metrics and the optimal model are negligible.

Overall, when comparing both of the models with subset features and all features, logistic regression with the subset variable performs best. The recall, f-measure, and ROC are higher for this model in comparison to the random forest model with all feature engineered variables.

It should be noted that the lack of model performance could be due to inaccurate feature engineering. Because the features were created from the raw data, but were not tested against any properly labelled versions of the features, there was no way to guarantee accurate feature generation. Lastly, greater care was taken with the labelling of the testing data set by physicians. This means that the models were trained on less reliably labelled data, which could further contribute to the lack of performance. The authors recommend future areas of study using neural networks below.

Future Work

Deep learning is becoming a more popular way to extract features from ECG data in order to produce accurate predictions at a higher rate. Ribeiro et al. suggest that end-to-end learning is a viable alternative method to predicting and classifying ECG raw data compared to the traditional method of using signal processing techniques to extract features from the data.[2]

Unscaled and scaled features, produced from the process described in the Data and Methods section, were passed through a simple neural network to try and see if there would be any improvement in accuracy over the other methods used in this assignment. However, the test accuracy of each neural network sat between 50 and 60 percent, while the training accuracy rose to as high as 90 percent.

Following the training of the model on the engineered features, a model was run on the raw ECG data with the target variable being whether an ECG was normal or not. The training accuracy rose quickly in comparison to the previous neural network, reaching 99 percent accuracy after 16 epochs.

The test accuracy was not substantially better than that of the previous neural networks with the raw ECG data passed through the neural network in this assignment. Ribeiro et al. note that given the complexity of classification and the limited number of classes in their experiment compared to the total number of disease states, "even if a DNN is able to recognize typical ECG abnormalities, further analysis by an experienced specialist will continue to be necessary to these complex exams."

[1] Wagner, P., Strodthoff, N., Bousseljot, R.-D., Kreiseler, D., Lunze, F. I., Samek, W., & Schaeffter, T. (2020). PTB-XL, a large publicly available electrocardiography dataset. *Scientific Data*, *7*(1), 154.

[2] Ribeiro, A.H., Ribeiro, M.H., Paixão, G.M.M. et al. Automatic diagnosis of the 12-lead ECG using a deep neural network. Nat Commun 11, 1760 (2020). https://doi.org/10.1038/s41467-020-15432-4.