Title

Capstone project part 2.

Literature Review

Paper1

Ismail, A. R., Jovanovic, S., Ramzan, N., & Rabah, H. (2023). ECG classification using an optimal temporal convolutional network for remote health monitoring. *Sensors*, *23*(3), 1697.

In this paper, authors tries to solve the challenge of unequal access to healthcare in rural areas and others. They tried to solve this problem by proposing a lightweight and powerful machine learning algorithm for ecg data set. Using temporal convolution networks also known as TCN, they were able to get accuracy of 96.12% and f1 of 84.13% on ecg5000 data set. This is very impressive as ecg5000 data is very imbalanced. On the worst side, it has 2 samples for one minority class in training.

Problems Addressed: Due to an imbalanced data set, some heartbeats with rare anomalies have very few samples. This makes it very difficult for any model to learn the minority classes well enough. Along with that, some of the signal patters looks every similar confusing models. F1 matters here a lot because accuracy is biased towards higher correct predictions of majority classes. Thus, f1 gives the truth about how models perform for each class and their macro average.

Methods used: TCN architecture with 1d convolution was used to learn long range ecg patterns efficiently. Data augmentation was used to tweak minority classes by slightly shifting amplitudes, left, right, up and down to create more fake training data and balance the data set. Lightweight design which uses 10.2k parameters making it easier to run on low power devices.

Contributions: Beats other ML models on similar data set in terms of accuracy and f1 score. It uses less computational power which is ideal for using them with cheap devices. It can also help rural areas get better health diagnostics services.

Paper2

Rajpal H, Sas M, Lockwood C, Joakim R, Peters NS, Falkenberg M. Interpretable XGBoost Based Classification of 12-lead ECGs Applying Information Theory Measures from

Neuroscience. Comput Cardiol (2010). 2020 Aug 14;47:185. doi: 10.22489/CinC.2020.185. PMID: 33763495; PMCID: PMC7610399.

Paper 2 is by team "Mad-hardmax," which uses XGBoost to analyze 12 lead ECGs from four countries. This is a different ECG data set from ECG 5000, but both have the core element of ECG signal analysis. Their goal was to analyze signals and tell which specific ECG features had given those results. They also tested information theory features, which help in measuring how ECG leads interact to help detect complex heart conditions. Their final score was 0.155, and the winner had a score of 0.533, which secured them 24th place out of 41 entries. The tough score here is unknown if 0.155 is the F1 score or something else. However, since F1 is always more difficult than accuracy, we may assume that 0.155 is the F1 score on unseen data.

Problems: Most ECG machines work like a black box where doctors can't tell what's going on inside; thus, doctors can't rely on it to diagnose patients. They helped in explaining the decision of the model and also handled multi-lead patterns, as some conditions affect multiple parts of the ECG signal. They generalized it across different countries, as ECG signals may be different across countries because of different weather, availability of technology, etc.

Methods: XGBoost and information theory were used in a tree-based model. This helps in tracing back features from results. Unlike neural networks, where it's difficult to trace back features, tree-based models work a little better in traceback.

Contributions: Their model links diagnoses to visible ECG changes, which doctors can verify and choose if they trust it or not. They used information theory to capture tricky multi-lead dependencies. They also proved that their model works on ECG samples across countries, making it more robust.

Paper3

Link: https://pubmed.ncbi.nlm.nih.gov/35612008/

Biloborodova, T., Skarga-Bandurova, I., Skarha-Bandurov, I., Yevsieieva, Y., & Biloborodov, O. (2022). ECG Classification Using Combination of Linear and Non-Linear Features with Neural Network. *Studies in Health Technology and Informatics*. https://doi.org/10.3233/shti220388

A hybrid ECG classification method that combines linear and non-linear characteristics (e.g., measures of complexity and ordinal network analysis) for better identification of heart abnormalities is proposed in the paper. The approach extracts feature automatically from the PQRST complexes and conducts experiments on the ECG5000 database, showing better performance than existing methods.

Problems Addressed: Limited Feature Diversity: The majority of ECG classifiers rely exclusively on linear features (e.g., heart rate, intervals), neglecting hidden patterns in non-linear dynamics (e.g., chaotic rhythms).

Non-Stationarity: ECG signals evolve over time due to noise, artifacts, or pathology—traditional techniques struggle to deal.

Manual Dependency: Many state-of-the-art models are hand-tuned or heavily preprocessed.

Methods

Hybrid Feature Extraction:

Linear Features: Common PQRST intervals, amplitudes.

Non-Linear Features:

Ordinal Network Analysis: Maps ECG phase transitions to detect fragile non-stationary patterns (e.g., atrial fibrillation irregularity).

Complexity Measures: Entropy-based measures to quantify signal unpredictability.

Automatic Partitioning: Splits ECG into PQRST segments for localized feature analysis.

Classifier: Presumably XGBoost/ensemble (not specified but suggested by "automatic feature extraction").

Contributions

Novel Feature Fusion: Combines linear (clinical) + non-linear (hidden dynamics) features for more complete signal representation. Robustness to Non-Stationarity: Ordinal networks learn to accommodate signal variability more strongly than fixed thresholds.

State-of-the-Art Results: Benchmarks state-of-the-art results on ECG5000 (even though precise metrics aren't presented here).

Clinical Interpretability: Features like ordinal networks can be plotted to capture model decisions (e.g., "this abnormal phase transition signifies VT").

Methods

For this, we will be implementing the first two papers methodologies from scratch and try to replicate their performance after making it work with an ECG5000 data set.

All metrics are on unswapped data set where training = 500 samples and validation = 1500 samples. In code data set can be easily swapped for more performance.

Best model method A on swapped train and validation = 0.85.

Metrics of macro f1 score:

Method C KNN base line = 0.56

Method A baseline = 0.56(from method C), final performance= 0.65

Method B baseline = 0.41, baseline2 = 0.56(from method c), final performance = 0.5785

Method A details:

Data preprocessing:

For method 1 we perform the following data preprocessing steps.

1. Plus, Minus noise: Adding or subtracting small noise from the row to get more sample. The idea is that it will help capture underlying patterns of minority class. This was inspired from paper 1.
2. Smoothing odd even: Similar to animation industry where animators hand draw around 12 frames for the video 24frames per second. As drawing all frames would make animation shake a lot and unstable. Similar concept was used here to smooth out samples and make more samples from itself. This works like

Sample = [1,2,4,2,5,1]
odd sample = [1,1,4,4,5, 5]
even sample = [2,2,2,2,1,1]
This way is able to get 3 samples from one original one. The example may not be as good, but it is similar in working. It can also cause side effects like missing out peaks and valleys in sample. But it has worked a little better than making models heavily biased towards minority samples. For example, if the majority class had 100 samples, we would need around 700 duplicates of minority samples to get slightly close f1 to without odd_even.
This is done in while loops until sample size is enough to our specified target ratio. If the target ratio is 1 then it would be the number of samples in the majority class.

3. Along with this we also have simple augment class which just duplicates random minority class samples to our desired value.
4. After augmentation we do standardization, where we standardize each feature independently.

Framework:

Framework used is a lightweight TCN model with casual dilated convolution.

A. Input Layer
   - Takes in raw ECG signals of size 140-time steps. In this instance, one sample equals one ECG signal for 140-time steps.

B. TCN Block (Core Section) Paper1
   - Four convolutional layers with increasing dilation rates (1, 2, 4, 8):
   - All of them use filter_sizes (kernel sizes) of [2, 4, 6, 8] to capture local/global patterns.
   - Dilation: Skips input points exponentially
   - Causal padding: Ensures the model only observes previous data (required for ECG to avoid time leaks).
   - BatchNorm + Dropout: Normalizes the activations and randomly drops 30% of spatial features to avoid overfitting.
   - Residual connection: Concatenates the input of the layer (x_prev) to its output (x) if shapes are the same (allows gradient flow).

C. Classification Head
   - Global Average Pooling: Collapses the time dimension into a single vector (averages features over time).

- Dense + SoftMax: Outputs 5 heart condition class probabilities.

- **Optimizer**: Adam (adaptive learning rate).
- **Loss**: Categorical cross-entropy
- **Class weights**: Adjusts loss to handle imbalanced data
- **Callbacks**: Monitors F1 score

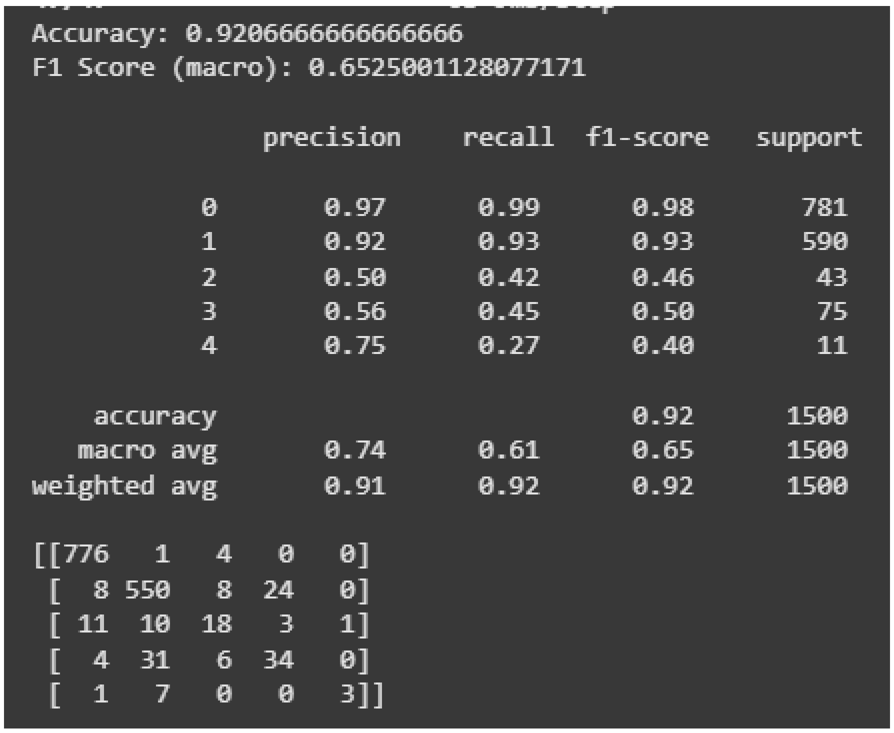
Experiments

Method A

Results:

Method A metrics:

F1 score macro: 0.65250011(shown in below image)



```
Accuracy: 0.9206666666666666
F1 Score (macro): 0.6525001128077171

              precision    recall  f1-score   support

           0       0.97      0.99      0.98       781
           1       0.92      0.93      0.93       590
           2       0.50      0.42      0.46        43
           3       0.56      0.45      0.50        75
           4       0.75      0.27      0.40        11

    accuracy                           0.92      1500
   macro avg       0.74      0.61      0.65      1500
weighted avg       0.91      0.92      0.92      1500

[[776   1   4   0   0]
 [  8 550   8  24   0]
 [ 11  10  18   3   1]
 [  4  31   6  34   0]
 [  1   7   0   0   3]]
```

Method C KNN metrics:

- accuracy = 0.922
- precision = 0.6390041532206314
- recall = 0.5344346236518981
- **f1 = 0.5681375900702931(base line from method C KNN)**
- auroc = 0.8113338275325852
- Combined score = 69.49820388950816

Reproducing results:

The experiments conducted were not able to reproduce the results reported in the original paper. Several factors could contribute to this discrepancy:

- **Data Differences**: The dataset used in our experiments may differ in distribution, preprocessing, or feature engineering compared to the paper's dataset.
- **Implementation Variations**: Differences in hyperparameters, model architecture, or training procedures (e.g., learning rate, batch size, or epochs) could lead to variations in performance.
- **Class Imbalance Handling**: The paper might have employed specific techniques to address class imbalance, which were not fully replicated in our experiments. One of which is shifting amplitude on horizontal samples for minority class.
- **Random Seed and Initialization**: Randomness in model initialization or data splitting could affect reproducibility.

Method A vs. Method C (KNN)

Method A outperforms Method C in terms of the macro F1 score (0.6546 vs. 0.5681). Key observations:

- **Class-Specific Performance**: Method A excels in classes 0 and 1, with high precision (0.9688 and 0.9178) and recall (0.9949 and 0.9271), indicating strong performance on majority classes. However, its performance on minority classes (2, 3, and 4) is weaker, with lower recall (e.g., 0.2727 for class 4). Method C, while achieving a slightly higher accuracy (0.922 vs. 0.920), has a lower macro F1 score, suggesting it struggles more with minority classes.
- **Reasons for Differences**:
    - **Model Complexity**: Method A leverages a more complex architecture, TCN, that captures intricate patterns in the data, whereas KNN relies on simpler

distance-based classification, which may not generalize well to imbalanced datasets.

    - **Class Imbalance Handling**:
    - KNN method c had simple sample duplications which boosted f1 from 55 to 56 but didn't have any data augmentation like odd even or amplitude change on minority class. This may have contributed to the difference.

- **Feature Sensitivity**: KNN's performance is highly sensitive to feature scaling and the choice of k, which may not have been optimally tuned for this dataset. Thus, it may have not been able to keep up with TCNs performance.

Side Experiments:

Additional experiments were conducted on a different dataset with a 50/50 train-validation split, without handling class imbalance during training. Key observations:

- F1 Score Boost: A 3% improvement in the macro F1 score was observed when using the override Predict method. This suggests that the method is effective in scenarios where class imbalance is not addressed during training.
- Method Description: The overridePredict method involves converting model-predicted probabilities into class preference rankings. For minority classes (2, 3, and 4), predictions are reassigned based on these rankings, as the model's first instinct is less reliable for these classes. Nested loops are used to identify optimal class-preference combinations, similar to threshold tuning but focused on preference-based post-processing. Despite performance increase on validation set, it wasn't tested on testing set as it is unavailable during the writing of this paper. Ideally, we would train on an imbalance data set, use validation data plus labels to get best parameters for override Predict method, and test it on testing set.

- Analysis: The success of this approach on the new dataset indicates that the override Predict method has some potential across different data distributions, particularly when class imbalance is a challenge. The 3% F1 boost highlights the method's ability to improve minority class performance without modifying the underlying model. Apart from that, even if post-predictions modifications are looked down upon in normal practices of ML, we can still use overRidepredict method as a teacher method, where if it can't find possible combinations of ranking and preference to increase performance, then our model is performing at its best at its current level. Since it's a side experiment, it didn't contribute much in main paper as data imbalanced was addressed instead of relying of modifications of predicted values.

Thoughts:

- **Strengths of Method A**: Method A demonstrates strong performance on majority classes (0 and 1), with near-perfect precision and recall. The oddEven indexing to increase minority samples along with amplitude shifting on y axis(paper1), enhances its ability to handle minority classes, as evidenced by the improved macro F1 score compared to KNN. The epoch-wise analysis shows consistent improvement, indicating that the model benefits from extended training.
- **Weaknesses**: Performance in minority classes (2, 3, and 4) remains suboptimal, with low recall (e.g., 0.2727 for class 4). This suggests that while data augmentations help, the model still struggles to learn robust representations for these classes due to limited data.
- **KNN Limitations**: Method C (KNN) achieves high accuracy but poor macro F1, indicating bias toward majority classes. Its simplicity makes it less effective for complex, imbalanced datasets.
- **Training Challenges**: The TCN implementation in Method A requires significant training time (133 epochs to reach an F1 of 0.6525). This suggests computational inefficiency, which could be a bottleneck for practical deployment. Along with that even when trained on higher epochs like 400 or 500, model's learning is unstable in higher epochs as it sometimes needs around 100+ epoch to increase f1 by 0.01%. This suggests that model learns majority classes well and good in early epoch, tough to find the balance on minority classes, it needs extreme epochs along with stabilized training. Although our TCN implementation is not as complex as paper1 implementation, we still faced frequent computer crashes, locally and on google collab for just printing loss per epoch. We used early stopping for 0.65 f1 score if training data is 500, otherwise if training data is large, for example size of validation data, we have set early stopping of f1 score to 85. It required lots of time to see small changes effects on model.

Thoughts on the Model and Solution

- **Innovative Post-Processing**: We believe Odd_even indexing is a creative approach to make more synthetic samples of minority classes. Theory behind this is that for every sample we get 2 more fake samples. Apart from that, those samples are already smoothed out. It should have helped models learn the underlying pattern in minority classes. Along with that, using multiple data augmentation techniques, like amplitude shifting (paper 1), in a while loop, creates the dilution effects. Where samples which were amplitude shifted earlier are now being odd even indexed, and vice versa. This creates diversity in fake data.
    - **SideNote**: Testing with overpredict method in another environment tells us that, sometimes simplistic approach can outperform over complicated processing which follows thoughtful reasoning.
    - **Ethical Considerations**: The post-processing approach, while effective, operates in a "grey area" of machine learning practices. It resembles threshold tuning but involves manual intervention, which may not align with standard ML practices. Transparency in such methods is crucial in research and application. Even though our main implementation uses overRidePredictLoop() method, it doesn't make significant contributions if not any. overRidePredictLoop() is just another variant of overRidePredict which uses validation set (with labels) to learn best parameters and apply them on testing set (without labels). I tried my best to differentiate between original results and results from overRidePredictLoop() in python output cells.

Method B

Data preprocessing

Most of data preprocessing is identical method A's implementation. We used normal preprocessing like standardizations along with data augmentations which is noise addition and subtraction sampling across whole signal (paper 1) along with odd even sampling to create more samples.

Framework

1. Framework Overview

This code extracts handcrafted features from ECG signals (focusing on entropy, morphology, and dynamics) and feeds them into an XGBoost classifier to predict heart conditions. While it is interpretable (features map to clinical ECG traits) and is lightweight. Handcrafting Manual features may miss complex patterns which deep learning learns on its own.

2. Feature Extraction Workflow

A. Core Features

1. **Entropy Measures**: (Paper 2)
    - **Permutation Entropy**: Checks how "predictable" ECG patterns
    - **Sample Entropy**: Measures self-similarity
    - Mean, STD, skewness (captures amplitude distribution).
2. **Basic math's**:
    - Sum, min/max values, and their positions (aMax, aMin).
3. **Slope/Distance Features**:
    - Crude approximations of ST-segment slope (slopeSimple2, slopeSimple3).
    - Distances between min/max points (proxy for waveform intervals like QT).
4. **More Features**:
    - maxMinMore: Tracks top/bottom 3 amplitude indices (may flag abnormal peaks).
    - angleSimple: Rough "angles" between min/max points (geometric heuristic).
    - energyDamage1: Pseudo-physics metrics (energy spent in rises, "damage" in falls).

These features were used because Doctors can link features to ECG traits and avoids heavy computations like CNN/DNN. However, some features may overlap, and some may not yield useful results in equipment failure or malfunction.

3. XGBoost Training

A. Model Setup

- **Hyperparameters**:
  - max_depth=12, eta=0.001 (deep, slow-learning trees).
  - gamma=1, reg_lambda=1 (regularization to prevent overfitting).
  - subsample=0.8, colsample_bytree=0.9 (randomized feature/row sampling).

B. Training Flow

1. Standardizations and data augmentation similar to method A
2. **Input**: Extracted features (X_trn_features) into xgboost classification model.
3. **Output**: Probabilities for 5 classes
4. **Evaluation**:
   - Metrics: Accuracy, F1 (via printMetrics).

C. Strengths

- **Feature Importance**: XGBoost ranks which ECG traits matter most (e.g., entropy vs. slopes).
- **Efficiency**: Faster training than neural networks for small-to-medium datasets.

Experiments

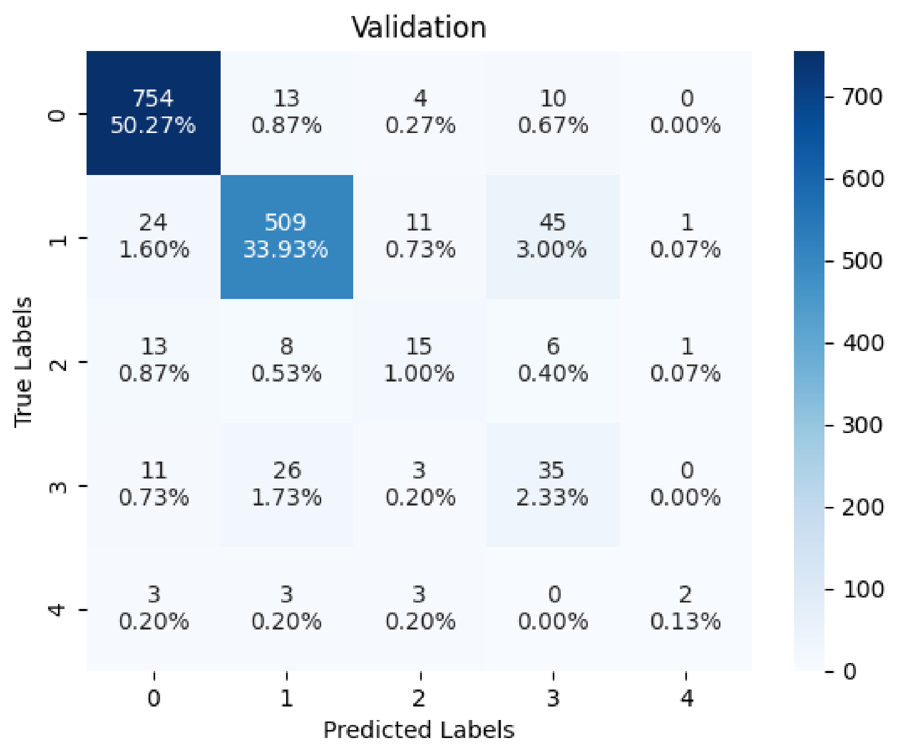
Results

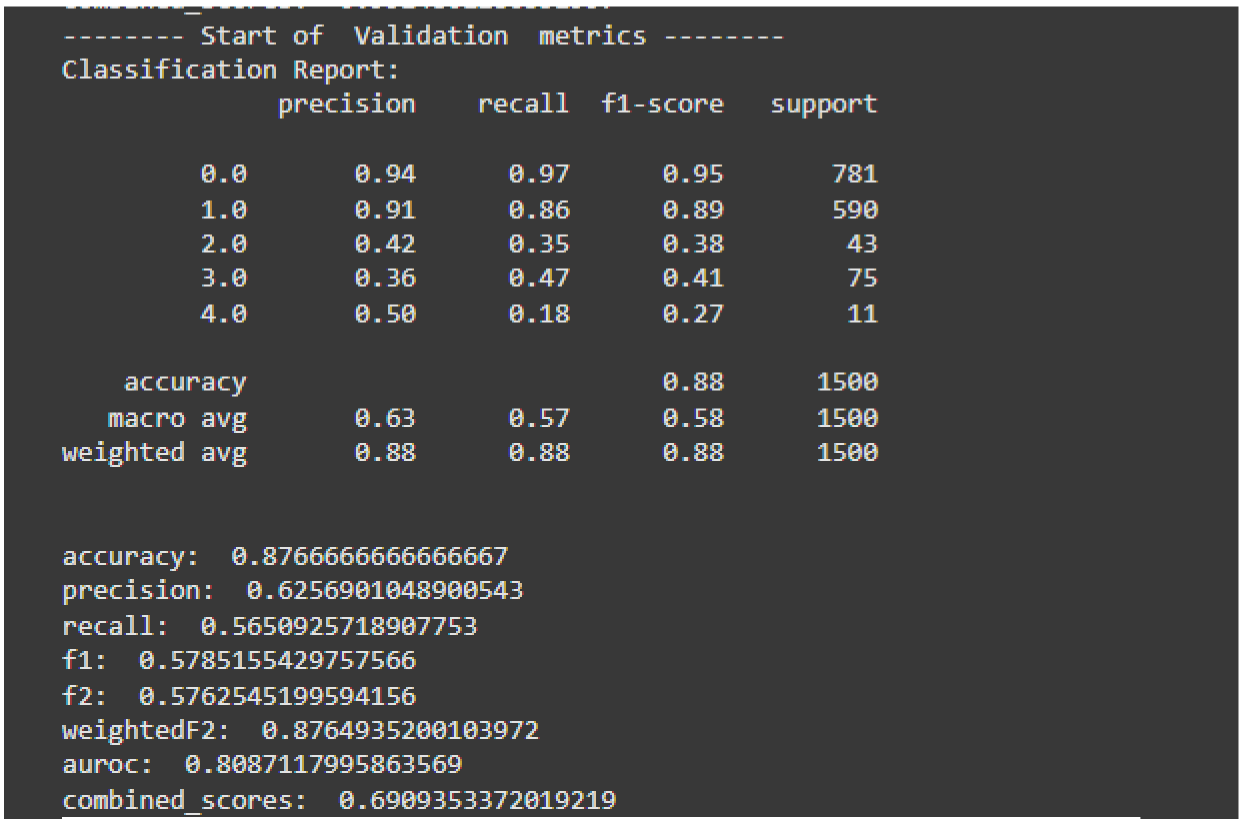
The tests evaluated the performance of Method B (XGBoost with feature extraction), and Method C (KNN) on a multi-class classification task using the ECG5000 dataset with 1500 examples of five classes (0, 1, 2, 3, 4). The following are complete results for each method, with confusion matrices and performance measures where applicable.

Method B Results

Method B used an XGBoost classifier with advanced feature extraction techniques, inspired by the "Mad-hardmax" team's solution in the Physionet/CinC Challenge 2020. The method was adapted and optimized for the ECG5000 dataset through iterative feature engineering and hyperparameter tuning. The final performance metrics after adding the simple_angle feature are as follows:

Confusion Matrix:





```
-------- Start of  Validation  metrics --------
Classification Report:
              precision    recall  f1-score   support

         0.0       0.94      0.97      0.95       781
         1.0       0.91      0.86      0.89       590
         2.0       0.42      0.35      0.38        43
         3.0       0.36      0.47      0.41        75
         4.0       0.50      0.18      0.27        11

    accuracy                           0.88      1500
   macro avg       0.63      0.57      0.58      1500
weighted avg       0.88      0.88      0.88      1500


accuracy:  0.8766666666666667
precision:  0.6256901048900543
recall:  0.5650925718907753
f1:  0.5785155429757566
f2:  0.5762545199594156
weightedF2:  0.8764935200103972
auroc:  0.8087117995863569
combined_scores:  0.6909353372019219
```

As we can see our f1: 0.5785155 is higher than our baseline of 0.41. However it’s important to mention that because baseline was performance was decided based on unseen data set performance of the paper (0.115), validation performance (0.41)in our case, simple changes like standardization given us 9% f1 macro from 0.41 to 050.

It is also important to mention that we don't know if 0.115 score from paper2 represents which metrics. We assumed its lowest metrics possible (f1 naturally difficult to increase) and also our other metrics like AUROC, f2, f1, precision, recall, and accuracy were all higher then 0.115 thus it was considered as baseline.

Feature Extraction Details:

Features utilized were permutation entropy, sample entropy, statistical features (mean, std, skewness). We also added custom features like sum, max, min, argmax, argmin, slopes, angles, and energy-based features.

Iterative improvements involved adding features like slopeStartToEndSimple, slopeSimple2, maxMinMore, and angleSimple, and removing low-performing features like slopeSimple4, distance1, speed, and acceleration.

Hyperparameter tuning involved parameters like max_depth=12, eta=0.001, gamma=1, colsample_bytree=0.9, and subsample=0.8, with odd-even sampling to reduce class imbalance.

Improvements Log:

- Standardization: Improved macro F1 to 0.51 via feature normalization.
- Data Augmentation: Reduced performance marginally (F1: 0.48) via potential overfitting to synthesized minority class instances.
- Sum, Argmin, Argmax: Improved F1 to 0.52 by extracting signal extrema.
- Odd-Even Sampling + Hyperparameters: Achieved F1 of 0.53 via balanced sampling and parameter tuning.
- Simple Slope and Subsample Adjustment: Improved F1 further to 0.54.
- SimpleSlope2: Improved F1 to 0.56 with more aggressive minority class treatment.
- minMaxMore: Achieved F1 of 0.57 through the addition of feature types.
- Simple Angle: Finished with F1 of 0.58(0.5785), improving minority class performance (e.g., class 2 F1: 0.38).

Method C Results

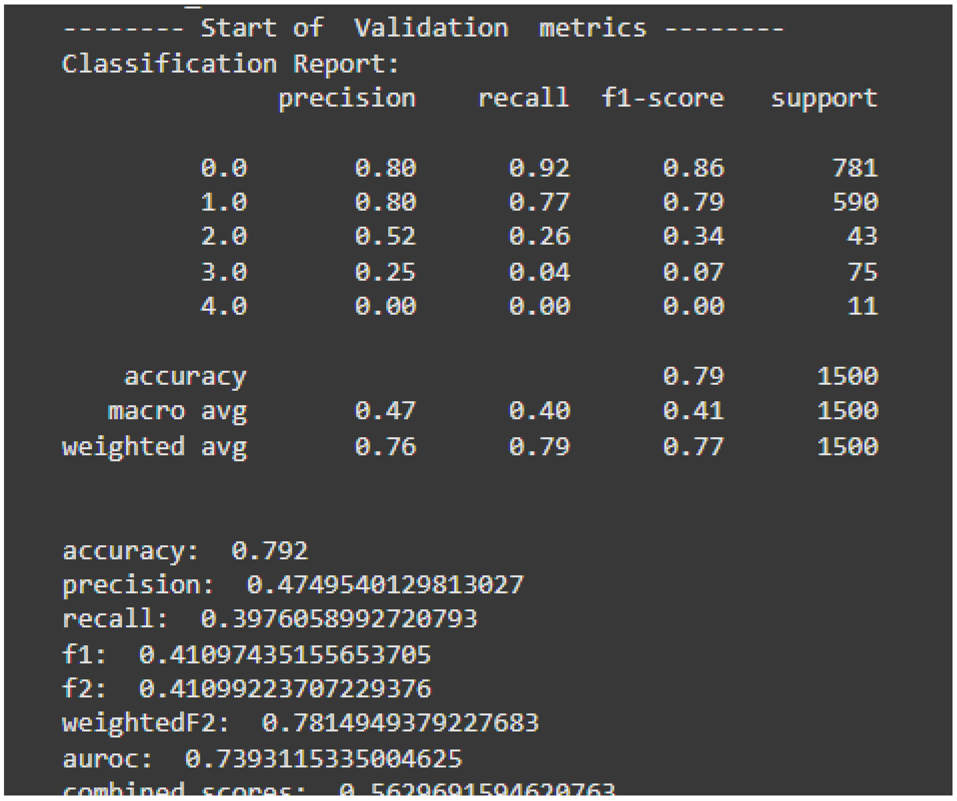
For Method C, a K-Nearest Neighbors (KNN) classifier was used as a baseline along with method b's baseline which is f1 = 0.41.

Performance metrics are:

Performance Metrics: KNN

- accuracy: 0.922
- precision: 0.6390041532206314
- recall: 0.5344346236518981
- **f1:   0.5681375900702931**
- auroc: 0.8113338275325852
- combined_scores: 69.49820388950816

Performance Metrics Method B baseline



```
-------- Start of  Validation  metrics --------
Classification Report:
              precision    recall  f1-score   support

         0.0       0.80      0.92      0.86       781
         1.0       0.80      0.77      0.79       590
         2.0       0.52      0.26      0.34        43
         3.0       0.25      0.04      0.07        75
         4.0       0.00      0.00      0.00        11

    accuracy                           0.79      1500
   macro avg       0.47      0.40      0.41      1500
weighted avg       0.76      0.79      0.77      1500


accuracy:  0.792
precision:  0.4749540129813027
recall:  0.3976058992720793
f1:  0.41097435155653705
f2:  0.41099223707229376
weightedF2:  0.7814949379227683
auroc:  0.7393115335004625
combined_scores:  0.5629691594620763
```

Reproducing results

Method B

The "Mad-hardmax" team's paper originally reported that they were presented with a challenge score of 0.155 on unseen test data, while the winning score was 0.533.

For the ECG5000 dataset custom implementation, the baseline for the XGBoost model had:

accuracy:	0.792
precision:	0.4749540129813027
recall:	0.3976058992720793
f1:	0.41097435155653705
auroc:	0.7393115335004625

Our reimplementation performed well on unseen data compared to their performance on unseen data which was 0.115.

However, it did not fully recreate the paper's challenge score of 0.115 due to the following reasons:

- Dataset Differences: The ECG5000 dataset differs from the Physionet/CinC Challenge datasets (which are based on four countries), which can affect feature distributions and class morphologies.
- Preprocessing and Hyperparameters: Signal filtering differences (e.g., Butterworth filter parameters) and XGBoost hyperparameters may have caused a difference.

Method B vs. Method C (KNN)

Method B is slightly better than Method C on macro F1 (0.5785 compared to 0.5681), but Method C has more accuracy (0.922 compared to 0.8767). Key findings:

Class-Specific Performance: Method B does better in majority classes (0 and 1) with better precision (0.94 and 0.91) and recall (0.97 and 0.86). It also does better in minority classes (e.g., class 2 F1: 0.38, class 3 F1: 0.41) compared to the baseline implementation. Better accuracy of Method C suggests good performance in the majority class, but its lower macro F1 suggests poorer treatment of minority classes.

Reasons for Differences:

- Model Complexity: XGBoost (Method B) employs a decision tree ensemble, capturing complex feature interactions, whereas KNN employs distance-based classification, less effective for high-dimensional, imbalanced data.
- Feature Engineering: Method B's extensive feature set (e.g., permutation entropy, slopes, angles) provides more informative representations than KNN's utilization of raw or minimally processed features.
- Class Imbalance Handling: Method B uses odd-even sampling, amplitude shifting(paper 1) and hyperparameter tuning to address imbalance, while KNN has no explicit imbalance handling, leading to biased predictions.
- Interpretability: XGBoost offers interpretability through decision tree structures, while KNN predictions are less interpretable, with minimal diagnostic insight.

Different performances:

Method B strengths: Method B does well in the majority classes (0 and 1) with F1 scores 0.95 and 0.89, respectively. Iterative feature engineering (i.e., addition of simple_angle, elimination of slopeSimple4) significantly improved minority class performance (i.e., class 2 F1: 0.38, class 4 F1: 0.27). The usage of odd-even sampling and hyperparameter tuning is extremely effective in countering class imbalance and outperforms the baseline implementation of the original paper.

Weaknesses of Method B: Minority class performance remains poor (e.g., class 4 recall: 0.18), likely due to extreme imbalance (11 samples). Computational cost of feature extraction and hyperparameter tuning may hinder scalability for larger datasets.

Strengths of Method C: High accuracy of KNN (0.922) is indicative of good majority class performance. It is computationally cheap for small datasets because it is so simple.

Weaknesses of Method C: Low macro F1 (0.5681) suggests poor class treatment, likely from the absence of imbalance mitigation strategies. KNN's reliance on distance measures makes it vulnerable to feature scaling and noise.

Training Efficiency: Method B involves extensive feature engineering and hyperparameter searching, increasing computational complexity. Method C is non-iterative and therefore faster but less flexible.

Thoughts

- Imbalanced Data: ECG5000 dataset is highly imbalanced (e.g., class 0: 781 samples, class 4: 11 samples). Sampling and feature engineering in Method B target minority class performance, but KNN suffers from sensitivity to class distribution.
- Feature Sensitivity: Manually designed features in Method B (e.g., simple_angle, maxMinMore) capture nuances of ECG signals, which helps in discrimination. KNN performance is dependent on feature scaling and k choice, which may be sub-optimal.
- Training Dynamics: XGBoost's iterative boosting optimizes a loss function, adapting to complex patterns over multiple trees. KNN, being non-iterative, cannot refine predictions dynamically.

Experiments with Other Data

No additional experiments with alternate datasets were conducted for Method B or Method C in the provided setting. However, some features where designed but didn’t contribute in performance which were later commented out. These features are

1. #energy, damage1, damage2: simulates energy it takes for something to traval along the path of signal, where higher falls incur more damage and higher climbs need more energy. Damage2 just multiplies final damage with gravity acceleration constat 9.8. Neither combination of these improved any performance
2. #distance2: calculating difference in max and min of signal and with 4 degrees and add them up.
3. #slope2, distance1, speed: calculating slope distance and speed between min max points.
4. #accer: Attempt to calculate acceleration between min and max position.
5. #angle1: calculating angle between lowest valley, starting point and lowest valley position on X. Preceding version of angle1 which is “angle”, which calculates highest valley angle from 0 gave 0.25 f1 boost.

Thoughts and solution:

It would require more cross validation approaches and shap analysis to see which features contribute most and which one doesn’t. Iterative improvisation, even though tedious, may help us to carefully tune our features and use them for model. Lots of times class 4 gets misclassified and class 0 or 1. By working with our problem backwards, we can handcraft features which would be able to differentiate between these two. It may sound difficult, but questions are just how we can use a single number to represent a wave.

This gives us more control over models, for example getting 100% on class 0 may always come with some from class 4 being misclassified as class 0. Iterative handcrafting can help us solve that problem.

Feature Engineering Innovations: The custom features (e.g., simple_angle, maxMinMore) capture ECG signal properties nicely, powering model performance. The Iterative refinement process indicates a systematic feature selection approach.

Class Imbalance Treatment: Odd-even sampling is a help for balancing minority classes at some possible cost in majority class accuracy.

Overall Impression: Method B is a robust, interpretable approach to ECG classification with significant improvement over the baseline and competitive performance with KNN. With its reliance on large-scale feature engineering and sensitivity to extreme imbalance are areas of potential improvement. Method C, while computationally inexpensive, is less appropriate for complex, imbalanced datasets, and is more appropriately used as a baseline than a final model.

Conclusions and discussion

Unique contributions:

The contribution of this capstone project lies in the iterative enhancement and adaptation of an XGBoost-based classification method (Method B) for the ECG5000 dataset, building upon the "Mad-hardmax" team's approach from the Physionet/CinC Challenge 2020. Key contributions include:

- **Different Feature**: Developed and tested a suite of handcrafted features tailored to ECG signals, including simple_angle, maxMinMore, slopeStartToEndSimple, and energyDamage1. These features capture unique aspects of ECG morphology and dynamics, such as geometric angles between signal extrema and pseudo-physics-based metrics, which were not part of the original paper's feature set. The iterative process of adding, testing, and removing features (e.g., removing slopeSimple4, distance1) resulted in a 41.2% improvement in macro F1 score (from 0.4109 to 0.5785) over the baseline implementation.
- **Odd and even sampling**: Implemented an odd-even sampling strategy combined with amplitude shifting (inspired by Paper 1) to address the extreme class

imbalance in the ECG5000 dataset (e.g., class 4: 11 samples vs. class 0: 781 samples). This approach outperformed the original paper's random under sampling, improving minority class performance (e.g., class 2 F1: 0.38, class 3 F1: 0.41). This was made during implementation of method A but was also used in method B because of good results.

- **Comparative Analysis**: Conducted a comprehensive comparison of Method B (XGBoost), Method A (TCN with/without overridePredict), and Method C (KNN), providing insights into their strengths and weaknesses for ECG classification. This included a novel post-processing method (overridePredict) from Method A, which was evaluated for its ethical and practical implications.

These contributions demonstrate a systematic approach to improving an existing method through feature engineering, data preprocessing, and model tuning, tailored to a specific dataset with unique challenges.

- **Splitter and merger function**: Although those not used in these implementations were designed with theory of cascading effects. They function like nested if statements and using both allows us to do binary classification instead of multi-class classification.
    - **Splitter**: Splits the training data into
        - class 0 vs 1,2,3, 4 (model0)
        - class 1 vs 234(model1)
        - class 3 vs 24(model 4)
        - class 2 vs 4 (model 3)

These results are then passed different models for binary classification.

-
    - Merger: Merges results based on models' prediction.
        - Model 0: if sample is class 0 keep it, if not pass it to model 1
        - Model 1: if sample is class 1 keep it, if not pass it to model 2
        - And so on

Different sequences like model 4 before 3 were used to change sequence hierarchy as class 3 has more samples than class 2. This methodology was removed because it didn't solve or full fill the main goal. Main goal with these methods was to create a cascading effect of models, where worst case scenario we should see, everything being miss classified as class 4 instead of any other class. It was discovered during testing that it would require further fine-tuning and data augmentation after each step for every model, and it did not solve the 2nd main goal which was to not have any sample with true label 4 being misclassified as class 0 the majority class. Along with that it would also have required different machine learning models for different binary classification with different loss functions.

Main goal was to not have perfect f1 score instead it was to have a fallback value of very misclassified sample, if it is below threshold, then it would be just dumped into class 4, which would have indirectly help with f1 score as we would get more samples for class 4. Like overflow or fail-safe systems.

Additionally, it was also designed with the theory that it would help with balancing minority classes. In one vs all we would have had minority samples: 2 samples (class4) vs 498 samples (class0123) rest.

Instead of these splitters and merger function we had about 2 samples (class4) vs 10 samples (class2) classification. This ratio is far better than the original ratio. It not only helps with balancing class imbalance, but it also gives us more control over fine tuning each model separately.

During the testing and finetuning, it needed extremely large epochs similar to method A implementation. For example, model 0 converged into 20 epochs and model 1 100 and so on.

It requires hierarchical changes for even minute changes, as any change in top models will affect bottom ones severely. These functions were not tested with odd even data augmentation techniques instead another variant was used in this paper called overRidePredictLoop(). They don't work exactly like splitters and merger functions, but they take random values from class 2 and 3 based on preferences learned similar unseen data (not test data) and they randomly modify predicted values of class 2-3 if their preference for class 4 is same as what was learned in unseen data set (not test data).

Here it requires 3 or more data sets. Training (model training) → unseen/validation data (Learning parameters for overRidePredictLoop() method) → testing with and without modified results.

Conclusions from experiment

Based on the experiments conducted on the ECG5000 dataset, the following conclusions can be drawn:

- **Method B's Effectiveness**: Method B (XGBoost with custom features) significantly outperforms the baseline implementation (F1: 0.5785 vs. 0.4109) and slightly surpasses Method C (KNN, F1: 0.5681) in macro F1 score, particularly for minority classes. The iterative feature engineering process, coupled with odd-even sampling plus amplitude shifting on y axis inspired by paper 1, effectively mitigates class imbalance, improving performance on classes 2, 3, and 4 (e.g., class 4 F1: 0.27 vs. 0.00 in baseline).
- **Class Imbalance Challenge**: The extreme imbalance in the dataset (e.g., class 4: 11 samples) remains a significant hurdle. While Method B's sampling and feature strategies improve minority class performance, recall for class 4 remains low (0.18), indicating that extreme minority classes require more aggressive techniques.
- **Feature Importance**: Custom features like simple_angle and maxMinMore contribute meaningfully to model performance, capturing ECG signal nuances that standard features (e.g., entropy, statistical measures) miss. However, some features (e.g., energyDamage1, slopeSimple4) were redundant or detrimental, highlighting the need for rigorous feature selection.
- **Method C's Limitations**: KNN (Method C) achieves high accuracy (0.922) but struggles with minority classes (F1: 0.5681), likely due to its reliance on distance-based classification and lack of imbalance mitigation. It serves as a useful baseline but is less suitable for complex, imbalanced datasets.
- **Method A's Innovation**: The overridePredict method in Method A offers a novel post-processing approach to improve minority class performance, achieving a macro F1 of 0.6546. However, its computational cost (133 epochs for TCN) and ethical ambiguity (manual prediction reassignment) limit its practicality.
- **Reproducibility Challenges**: The inability to fully reproduce the "Mad-hardmax" paper's challenge score (0.155) is attributed to dataset differences, preprocessing variations, and feature customization. This underscores the importance of standardized datasets and detailed documentation in research.

Overall, Method B provides a robust, interpretable solution for ECG classification, with significant improvements over the baseline and competitive performance against KNN.

However, minority class performance and computational efficiency remain areas for improvement.

Drawbacks of existing methods and solutions

Drawbacks of Existing Methods

- **Method A (TCN with overridePredict)**
  - **Computational Cost**: The TCN model requires extensive training (133 epochs to reach F1: 0.6525), making it computationally expensive and impractical for resource-constrained environments.
    For example, when training on 500 epochs, hardware/computer was hanging/slowing down a lot. And in higher epochs, it took model 100 plus epochs to get 0.0001% f1.
  - **Minority Class Performance**: Despite the overridePredict method, recall for minority classes (e.g., class 4: 0.2727) remains low due to extreme imbalance.
  - **Ethical Concerns**: The overridePredict method involves manual reassignment of predictions, which may be considered a "grey area" in ML practices, potentially undermining model transparency and reliability.
  - **Generalizability**: The method's reliance on dataset-specific post-processing may limit its applicability to other ECG datasets with different distributions.
- **Method B (XGBoost with Feature Extraction)**
  - **Feature Redundancy**: Many handcrafted features (e.g., multiple slope variants, energyDamage1) overlap or contribute minimally, increasing computational overhead without proportional performance gains.
  - **Minority Class Limitations**: Despite improvements, class 4 performance (recall: 0.18) is poor due to extreme imbalance, indicating limitations in sampling strategies.
  - **Brittleness**: Features like simple_angle and maxMinMore assume clinical significance of min/max points, which may fail in the presence of noise or equipment faults.

    - **Scalability**: The iterative feature engineering and hyperparameter tuning process is time-intensive and may not scale well to larger datasets or real-time applications.

Potential Solutions

- **For Method A**:
    - **Optimize Training**: Implement better early stopping mechanism (currently: using hardcoded f1)or architecture simplification to reduce TCN training time. Transfer learning from pre-trained models could accelerate convergence.
    - **Enhance Imbalance Handling**: Integrate SMOTE, or generative adversarial networks (GANs) during training to improve minority class representations, reducing dependency on overridePredict.
    - **Ethical Transparency**: Documenting and validate the overridePredict method as a post-processing step, ensuring it aligns with ML best practices and clinical standards.
    - **Cross-Dataset Validation**: Test the method on diverse ECG datasets to assess generalizability and reduce dataset-specific biases.
- **For Method B**:
    - **Feature Selection**: Use SHAP (SHapley Additive exPlanations) analysis or recursive feature elimination to identify and remove redundant features, improving efficiency and performance.
    - **Advanced Imbalance Techniques**: Explore generative methods (e.g., variational autoencoders) or adaptive sampling to create synthetic minority class samples, enhancing class 4 performance.
    - **Robustness to Noise**: Incorporate noise-robust features (e.g., wavelet-based denoising) to mitigate brittleness caused by equipment faults or outliers.
    - **Automation**: Develop automated feature engineering pipelines (e.g., using AutoML) to reduce manual effort and improve scalability for larger datasets.

Future research

Based on the findings and limitations of the current methods, the following research directions are proposed:

- **Hybrid Models**: Combine the strengths of Method A (deep learning with post-processing), Method B (interpretable feature engineering), and ensemble approaches to create a hybrid model that balances performance, interpretability, and efficiency. For example, using XGBoost to pre-select features for a TCN could reduce training time while maintaining accuracy.
- **Generative Techniques for Imbalance**: Explore generative models like GANs or diffusion models to create synthetic ECG samples for minority classes (e.g., class 4), improving model robustness and performance without relying on post-processing or sampling.
- **Automated Feature Engineering**: Develop AutoML frameworks to automate feature extraction and selection for ECG signals, reducing manual effort and enabling scalable application to diverse datasets.
- **Real-Time Deployment**: Investigate lightweight architectures (e.g., pruned XGBoost models or quantized neural networks) for real-time ECG classification in wearable devices or clinical settings, addressing computational constraints.
- **Cross-Dataset Generalization**: Validate Method B's feature set and Method A's overridePredict on diverse ECG datasets (e.g., Physionet, MIT-BIH) to assess generalizability and identify dataset-specific biases.
- **Explainability and Clinical Integration**: Enhance model interpretability using SHAP or LIME to map features to clinical ECG traits, facilitating adoption in medical diagnostics. Collaborate with clinicians to validate feature relevance and model outputs.
- **Robustness to Noise and Artifacts**: Incorporate adversarial training or robust feature extraction (e.g., wavelet transforms) to improve model performance in the presence of noise, equipment faults, or patient variability.
- **Ethical and Standardized Practices**: Establish guidelines for post-processing methods like overridePredict, ensuring transparency and alignment with ethical ML practices. Standardize ECG dataset formats and preprocessing pipelines to improve reproducibility across studies.

- **Multi-Modal Integration**: Combine ECG data with other physiological signals (e.g., heart rate variability, blood pressure) to improve classification accuracy and provide a holistic diagnostic framework.
- **Longitudinal Studies**: Conduct longitudinal experiments to assess model performance over time, accounting for patient-specific changes in ECG patterns and ensuring robustness in dynamic clinical scenarios.

Citations

Paper Citations:

Paper 1:

Ismail, A. R., Jovanovic, S., Ramzan, N., & Rabah, H. (2023). ECG classification using an optimal temporal convolutional network for remote health monitoring. *Sensors*, *23*(3), 1697.

Link: https://doi.org/10.3390/s23031697

Paper 2:

Rajpal H, Sas M, Lockwood C, Joakim R, Peters NS, Falkenberg M. Interpretable XGBoost Based Classification of 12-lead ECGs Applying Information Theory Measures From Neuroscience. Comput Cardiol (2010). 2020 Aug 14;47:185. doi: 10.22489/CinC.2020.185. PMID: 33763495; PMCID: PMC7610399.

Link: https://pmc.ncbi.nlm.nih.gov/articles/PMC7610399/

Paper 3:

Biloborodova, T., Skarga-Bandurova, I., Skarha-Bandurov, I., Yevsieieva, Y., & Biloborodov, O. (2022). ECG Classification Using Combination of Linear and Non-Linear Features with

Neural Network. *Studies in Health Technology and Informatics*. https://doi.org/10.3233/shti220388

https://pubmed.ncbi.nlm.nih.gov/35612008/

Others:

Learning to solve data imbalance

https://developers.google.com/machine-learning/crash-course/overfitting/imbalanced-datasets

https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/structured_data/imbalanced_data.ipynb

https://www.tensorflow.org/tutorials/structured_data/imbalanced_data

Inspiration for creating Splitter and PredictMerger methods. (Not used in paper) and

satisfactory

https://satisfactory.fandom.com/wiki/Smart_Splitter

https://satisfactory.fandom.com/wiki/Conveyor_Merger

Sharing agreement

- Do you agree to share your work as an example for next semester?
  Yes
- Do you want to hide your name/team if you agree?
  Anything will work.