SIT720 Machine Learning

Task 11.1

Michael Rideout
Student Id: 225065259

Introduction

Reproducibility of scientific findings is a core foundation of the scientific method. In this report we will attempt to reproduce the finding from the academic paper “An efficient stacking-based ensemble technique for early heart attack prediction” (Bhagat et al. 2024) Through this we will investigate some pertinent issues surrounding the reproducibility of machine learning experiments in the field of computer science. This report is organised as follows:

Part 1 - Reproduction
In this section we will attempt to reproduce the findings of the academic paper in question. We will focus in particular on the areas of data preparation and model architecture. Missing information will be addressed and general recommendations made to improve the reproducibility of experiments.

Part 2 - Alternative Solution
In this section we will propose an alternative solution to the reference paper, again focusing on data preparation and model architecture. We will conform to providing all information necessary to reproduce the results.

Part 3 - Video Presentation
A link to the video presentation for this report.

0.1. Part 1 - Academic Paper Reproduction

In this section we attempt to reproduce the results for the academic paper “An efficient stacking-based ensemble technique for early heart attack prediction” (Bhagat et al. 2024). Where there is insufficient information to reproduce the implementation, assumptions will be made and stated. We divide this part into four section Data Preprocessing, Model Architecture, Comparative Results and Replication Findings

0.1.1. Data Preparation

The paper provides the URL for the dataset used in the study. It is:
https://www.kaggle.com/datasets/aasheesh200/framingham-heart-study-dataset
Manual modification of the file was done as the end of line character was only a carriage return, so this was translated into a new line character.

0.1.1.1. Missing Values

The paper states the following with regards to missing values:

“We also identify the missing values from the dataset and found missing data in the education, Risperdal, BPMeds, totChol, BMI, heart rate, and glucose columns. The glucose column had the most missing values, which was filled up using the data’s mode of 75.0. Other missing values from various columns are deleted since they did not significantly contribute to the dataset.**“** [1]

The paper lists the following columns as having missing values (education, Risperdal, BPMeds, totChol, BMI, heartRate and glucose). Risperdal doesnt exist in the dataset as a feature. The paper provides details on the method used to impute missing data for the glucose feature, but it doesnt state what methods were used in the other columns.

We have assumed the mean was used for all other columns. We imputed all missing values for all columns that had missing values and that were of a numeric type. Those columns were 'male', 'age', 'education', 'currentSmoker', 'cigsPerDay', 'BPMeds', 'prevalentStroke', 'prevalentHyp', 'diabetes', 'totChol', 'sysBP', 'diaBP', 'BMI', 'heartRate', 'glucose', 'TenYearCHD'. Mean was used as mode performed not as well as mean.

0.1.1.2. Outliers

The part of the paper relevant to outliers was

“Our dataset’s totChol and sysBP columns have Removable Outliers that may be removed. Identify outliers by checking the highest and minimum quartiles and removing the rows that include the outliers.” [1]

We implemented outlier removal using the standard IQR (Inter Qurartile Range). The authors did not state how many rows were removed by this process so it is hard to verify if the criteria was used.

0.1.1.3. Standardisation

The authors stated that a standard scaler was used. We have implemented the same.

0.1.1.4. Class imbalance

The authors described this approach to handling the class imbalance of the target:
“To address this issue, data balance was accomplished using the smote approach [33]. Oversampling and undersampling are two methods used in this procedure. It creates artificial samples for the marginalized minority. This strategy is much easier to overcome the overfitting problem of random oversampling. After completing the data balancing process, 3394 health and heart disease patients are in the database.” [1]

Based on this SMOTE was used to rebalance the target classes. The parameters used for this were not described. Neither was the number of instances in the dataset that we being rebalanced after preprocessing had been completed.

The authors did describe the number instances after SMOTE has been performed, 3394. Our SMOTE processing left us with 6594 records, therefore we performed resampling to align to 3394 instances (of each target class instance counts)

0.1.1.5. Correlation analysis

The authors addressed correlation amongst features:

“Pearson’s coefficient (PCC) eliminates unnecessary, duplicated, and redundant features from the data. Correlation coefficients range from −1 to 1, depending on the data. For values around −1, features are loosely connected and have little effect on model performance. For values near 1, features are firmly coupled and significantly impact model performance. When calculating the PCC, a 0.85% threshold is used. If the correlation value is higher than this threshold, the feature is discarded; if it is lower, it is maintained. After doing a feature co-relation analysis, all attributes are identical.” [1]

Our implementation produced the same results as the authors, there were no features that had a correlation greater than +/- 0.85 so no columns were removed due to this.

0.1.1.6. Feature Selection

The authors performed feature selection but they did not explicitly state what set of features they used when training the model. They stated:

“To classify data correctly, it is essential to identify the appropriate characteristics. According to Fig. 2, the RF feature selection method identifies the important 10 features. It aims to reduce dimensionality by selecting the essential feature that may improve the models’ performance. Even in machine learning approaches, this is a common feature selection strategy that has proven effective.” [1]

Figure 1 indicates that feature extraction was part of the overall architecture so it is highly probable that they did perform action

Figure 2 shows the feature importance ranked, and as the authors mentioned the 10 most important features, one can assume that the list of feature they used was "age", "sysBP", "BMI", "totChol", "diaBP", "glucose", "heartRate", "cigsPerDay", "education", "prevalentHyp" based on the 10 ten features in Figure 2

Our attempts to match this feature importance finding using Random Forests did not produce the same results. Our results disagreed in that the male feature was in our top 10 but prevalentHyp wasnt. We used the assumed top 10 list derived from Figure 2

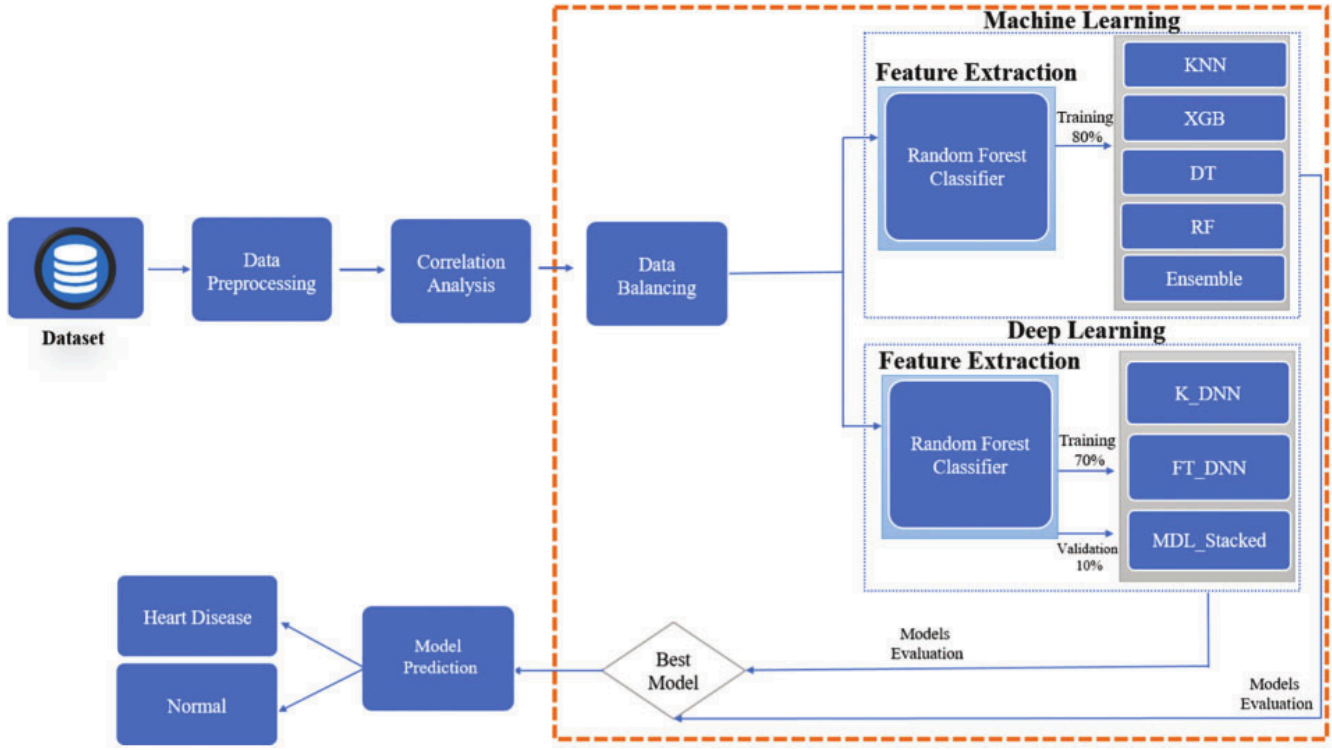

Figure 1: ML Architecture

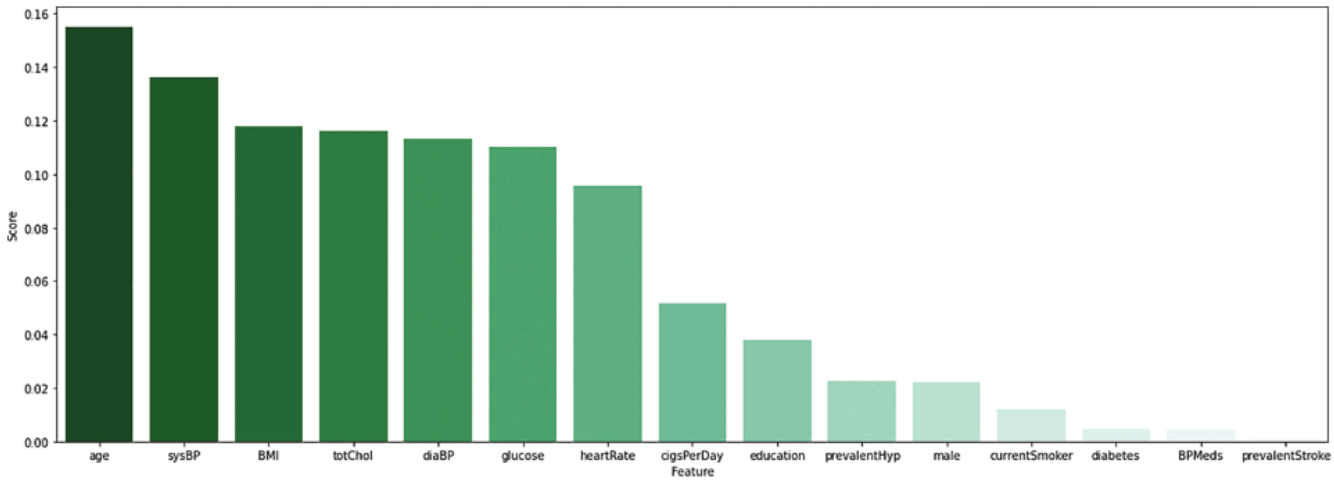

Figure 2: Feature Importance

0.1.1.7. Train / Test Splitting

The following was stated for the training / testing ratio split for deep learning models:
“70% of the data is utilized for training the ML and DL models, while the remaining 30% is used to assess the model’s accuracy” [1]

We therefore assumed that this was the split that was used for the other ML models. The information that is missing in this regard is the strategy of the split and if a random strategy had been used, what was the random seed for this.

0.1.2. Model Architecture

As shown in Figure 1, the paper’s model architecture has two groupings. Machine Learning classifiers and Deep Learning classifiers

0.1.2.1. Machine Learning Classifiers

The paper describes four machine learning classifiers RF, DT, KNN and XBG. However it does not specify any hyperparameters for any of these classifiers. Therefore we will assume default classifier hyperparameters as provided by scikit learn, the most popular python based ML library. We do set the random seed for any classifier that has this property for deterministic reruns of training.

0.1.2.2. Deep Learning Classifiers

FT-DNN Architecture
The authors specified the FT-DNN architecture as a 4-layer feedforward neural network with 16, 12, 8, and 4 neurons respectively. ReLU was used as the activation function, Adam as the optimizer, and binary crossentropy as the loss function, with a learning rate of 0.001.

We have replicated this configuration.

Training hyperparameters were not specified so common defaults used of epochs = 400, batch_size=32 validation_split = 0.1

DNN Architecture

The authors specified the DNN architecture as a 4-layer feedforward neural network with 12, 10, 8, and 6 neurons in the layers respectively, using ReLU activation, Adam optimizer, and binary crossentropy loss. They did not specify the evaluation metrics or the learning rate.
We were able to implement the same architecture.

Training hyperparameters were not specified so common defaults used of epochs = 400, batch_size=32 validation_split = 0.1

0.1.3. Comparative Results

Our attempt to replicate the paper's finding came across several discrepancies, resulting in an underperforming predictive model. The final model, the MDSLM model, had differences in the f1-score with our replication scoring 83.34, 11.4% lower than the 94.06 described in the reference paper.

Almost all of the models in the replication had a decline in performance in evaluation metrics compared to the reference paper’s. This underperformance is a result of the challenges in recreating the original experimental setup.

There were some interesting results for the Deep Learning models. We observed higher f1-scores for both models in our replication. This increase in performance of these two models was overwhelmed by the underperformance of the majority of the models.

Model	Metric	Replication Results	Paper Results	Difference (%)
XGB	Accuracy	89.17	94.02	-5.16
XGB	Precision	90.57	94.01	-3.66
XGB	Recall	86.84	94.01	-7.63
XGB	F1-score	88.67	94.01	-5.68
XGB	AUC-ROC	95.32	98	-2.73
RF	Accuracy	89.46	93.45	-4.27
RF	Precision	88.74	93.53	-5.12
RF	Recall	89.78	93.21	-3.68
RF	F1-score	89.26	93.25	-4.28
RF	AUC-ROC	96.2	93	3.44
DT	Accuracy	79.59	92.35	-13.82
DT	Precision	77.87	92.23	-15.57
DT	Recall	81.24	92.22	-11.91
DT	F1-score	79.52	91.22	-12.83

DT	AUC-ROC	79.63	91	-12.49
KNN	Accuracy	79.49	94.03	-15.46
KNN	Precision	71.66	94.03	-23.79
KNN	Recall	95.87	94.02	1.97
KNN	F1-score	82.02	94.02	-12.76
KNN	AUC-ROC	88.83	98	-9.36
FT-DNN	Accuracy	74.17	80.19	-7.51
FT-DNN	Precision	71.56	77.03	-7.1
FT-DNN	Recall	78.09	86.77	-10
FT-DNN	F1-score	74.68	69.43	7.56
FT-DNN	AUC-ROC	80.69	86.2	-6.39
DNN	Accuracy	73.36	76.73	-4.39
DNN	Precision	69.09	72.85	-5.16
DNN	Recall	82.12	86.19	-4.72
DNN	F1-score	75.04	67.32	11.47
DNN	AUC-ROC	78.17	83.1	-5.93
ML Ensemble	Accuracy	90.03	94.1	-4.33
ML Ensemble	Precision	91.33	94.04	-2.88
ML Ensemble	Recall	87.92	94.05	-6.52
ML Ensemble	F1-score	89.59	94.05	-4.74
ML Ensemble	AUC-ROC	96.21	99	-2.82
MDLSM	Accuracy	83.18	94.14	-11.64
MDLSM	Precision	80.62	94.25	-14.46
MDLSM	Recall	86.25	94.06	-8.3
MDLSM	F1-score	83.34	94.06	-11.4
MDLSM	AUC-ROC	91.52	99	-7.56

Table 1: Replication and Original Model Evaluation Metrics

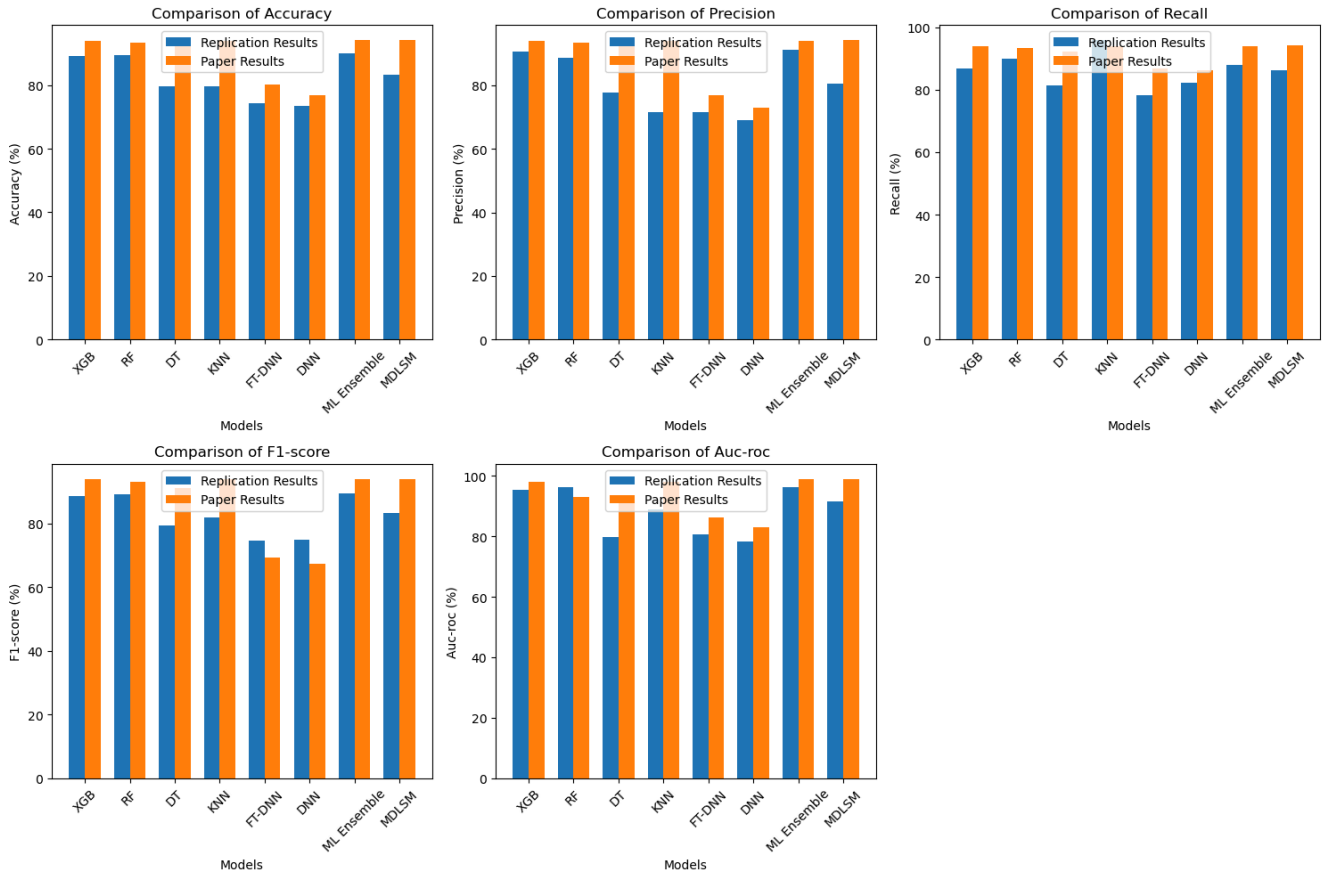

Figure 3: Replication Evaluations Results Comparisons

0.1.4. Replication Findings

The attempt to replicate the "An efficient stacking-based ensemble technique for early heart attack prediction" paper's results revealed notable performance differences, even though the overall architecture was replicated.

The missing information which hampered our ability to reproduce the authors findings accurately included:

- **Model Hyperparameters:** Specific hyperparameters used for each model (RF, DT, KNN, XGB).
- **Deep Learning Training Hyperparameters:** Training parameters for the deep learning models, such as epochs, batch size, and validation split details.
- **Train/Test Split:** Exact ratio used (e.g., 70/30) and the methodology for the split (e.g., random, stratified) including the random seed if applicable.

- **Data Cleaning Row Counts:** Precise counts of rows removed or modified at each step of the data cleaning process (missing values, outliers).
- **Library Versions:** Versions of libraries used for model implementation (e.g., scikit-learn, TensorFlow/Keras).
- **Random Seeds:** Random seeds used for model initialization and any randomized processes.
- **Data Preprocessing Details:** More detailed information about the exact steps taken during data preprocessing.

While the replication effort generally followed the paper's outline, the substantial differences in performance, particularly for the final ensemble model, highlight challenges in reproducing the original findings. This underscores the importance of providing comprehensive and explicit details in research papers to ensure reproducibility.

0.2. Part 2 - Alternative Solution

In this part, we describe our alternative implementation to the reference paper's for predicting heart disease.

0.2.1. Motivation

For our solution, we have chosen to use the AutoGluon [2], an automl framework for python. Our motivation to use this framework is because of the paper "A Comprehensive Benchmark of Machine and Deep Learning Across Diverse Tabular Datasets" [3] . This research provides a thorough comparison of machine learning and deep learning methods for a substantial set of tabular datasets. In Table 4 the results show that AutoGluon was the most performant framework for tabular datasets of all methods that were tested.

We also performed improved data preprocessing of the heart disease dataset, detailed below.

0.2.2. Experiment Setup

In this section we will provide all details for the setup and configuration used for our experiment

0.2.2.1. Environment

The pertinent library versions for this experiment in python (version 3.12) were:

Library	Version
pandas	2.2.3
scikit-learn	1.5.2
autogluon	1.3.1
imbalance	0.13.0

Table 2: Library Versions

0.2.2.2. Data Preparation

Here we describe all the data preparations steps undertaken before the model training

0.2.2.2.1. Missing Values Imputation

Before missing values were handled the following is the missing value count per feature:

Feature	Missing Value Count
age	0
education	105
currentSmoker	0
cigsPerDay	29
BPMeds	53
prevalentStroke	0
prevalentHyp	0
diabetes	0
totChol	50
sysBP	0
diaBP	0
BMI	19
heartRate	1
glucose	388
TenYearCHD	0

Table 3: Missing Values Count

The missing values were imputed using the median for the feature. After imputation, no features had missing values

0.2.2.2.2. Data Skewness

A histogram of all numeric features can be seen in Figure 4

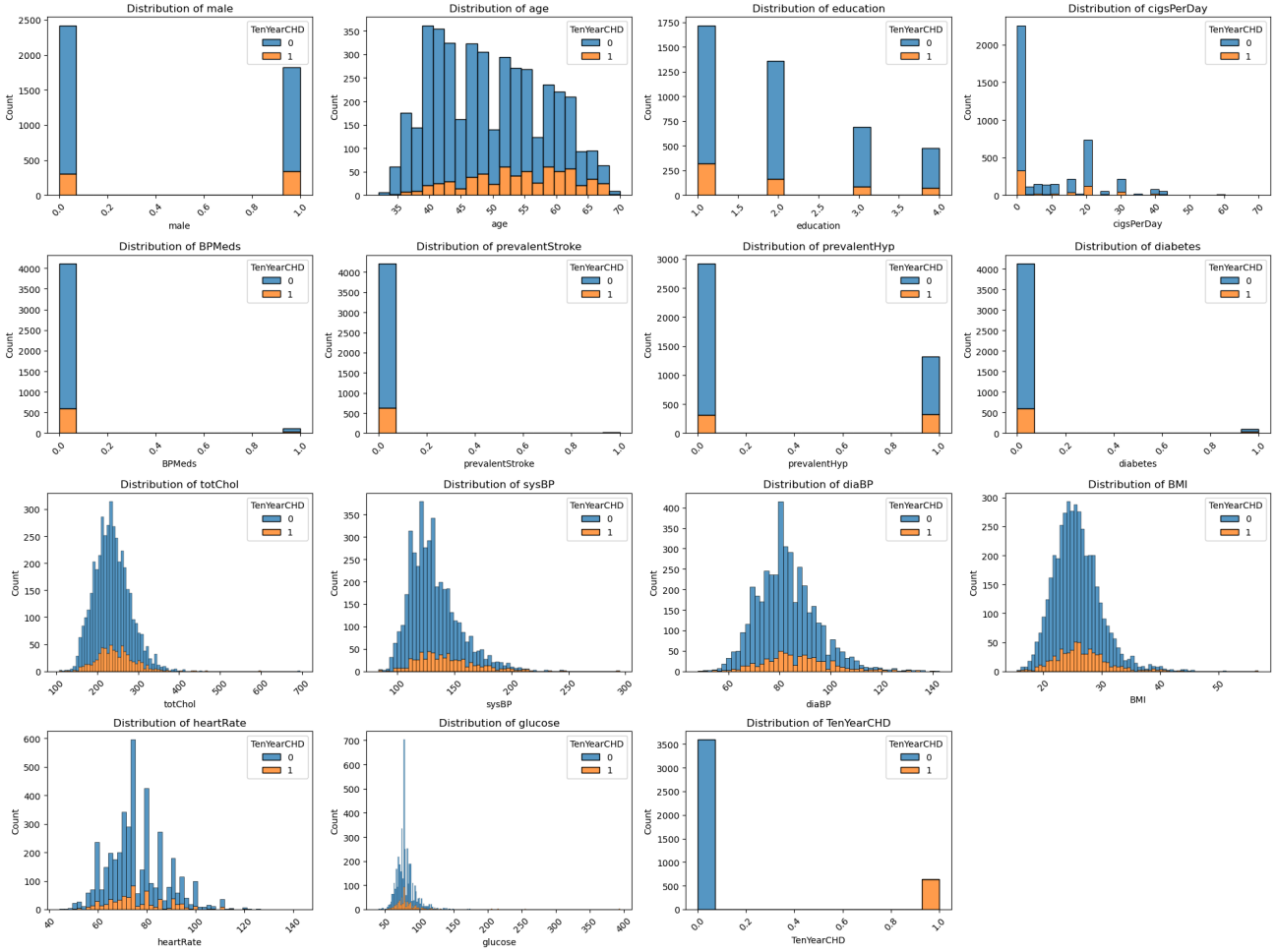

Figure 4: Feature Histograms

From this it can be seen that the following features are skewed and were therefore transformed in an attempt to normalise their distribution. The transformations were:

Feature	Transform
totChol	log
sysBP	log
diaBP	log
BMI	log

heartRate	log
glucose	log(log(log()))

Table 4: Skewed Features and their Transforms

cigsPerDay was skewed but it is a categorical feature represented numerically, so binning was performed to reduce the right tail of the cigarettes per day smoked. The bin boundaries were: -float('inf'), 0, 10, 20, 30, float('inf')

After these transforms were applied, the transformed features now have a more normal distribution as seen in Figure 5

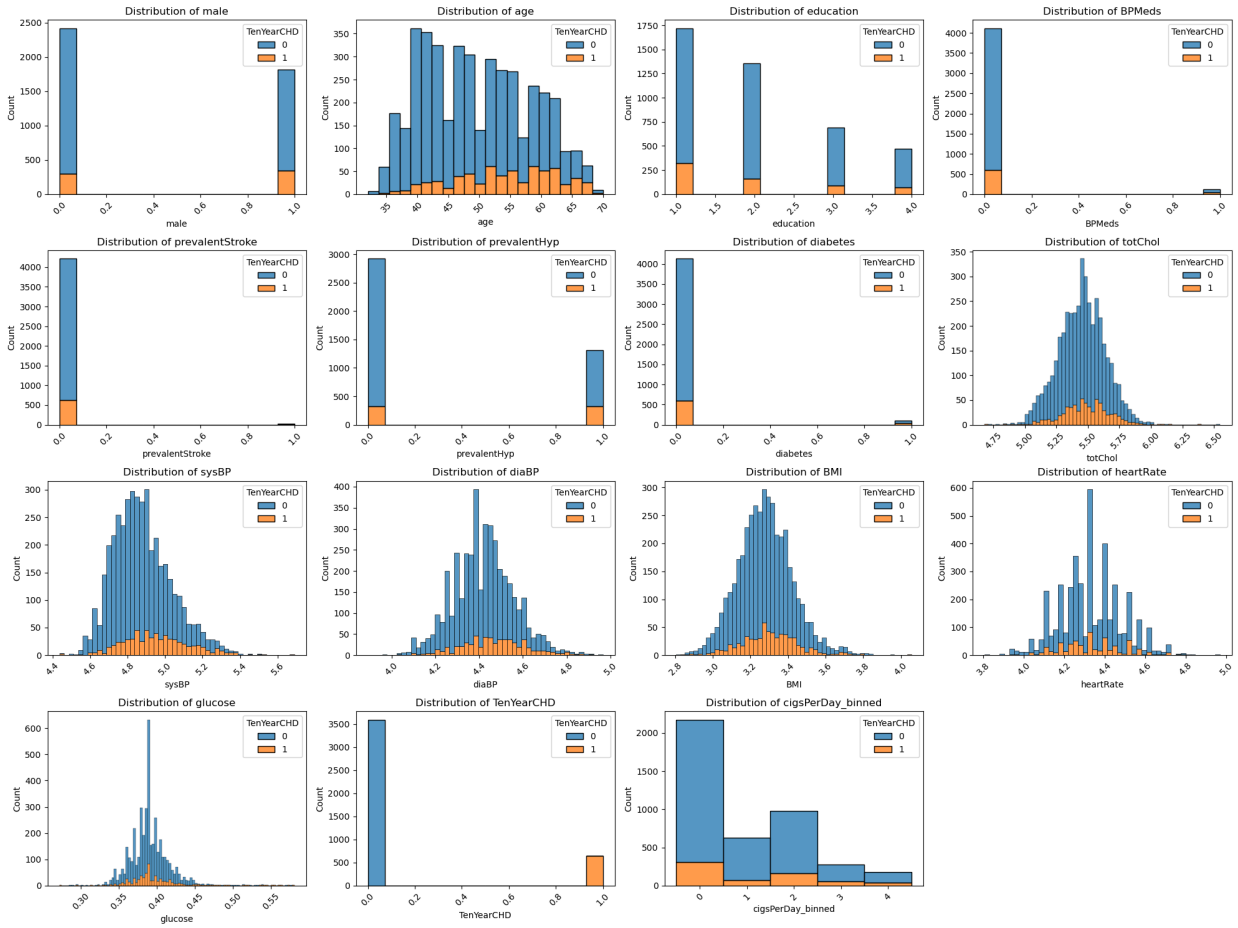

Figure 5: Feature Histograms Post Skew Handling

0.2.2.2.3. Data Standardisation

Data was standardised using the scikit learn's StandardScaler, all with default parameters

0.2.2.2.4. Class Imbalance

The target feature (TenYearCHD) had the following class count:

Class	Count
0	3596
1	644

Which indicates that the instances of heart disease far outweigh the number of instances of having heart disease. Due to this we perform SMOTE rebalancing with a random seed of 42. After rebalance the class count was the following:

Class	Count
0	3596
1	3596

0.2.2.2.5. Train / Test Split

Scikit learn train_test_split method was used to perform random splitting of the dataset into train / test sets with a test size of 0.2 and random seed of 42

Model Architecture

Autogluon was used to train the best fitting model using the following parameters

Tabulor predictor parameters were:

- Eval_metric = ‘f1’

Fit parameters were:

- Presents = ‘best_quality’
- time_limit=600
- num_bag_folds=5
- num_stack_levels=2
- random_seed=42

The time limit of 10 minutes was considered to be a reasonable amount of computing time. num_of_bag_folds is the number of folds used by bagging methods. 5 was considered sufficient and not too onerous computationally. Num_stack_levels is the number of stacking levels used in stacking ensembles. 2 was considered to be a good balance between complexity and computational requirements The resulting best model (which is the model used by AutoGluon to make the final predictions) was a stacked ensemble model consisting of LightGBM models.

The hyperparameters for the model can be found here
https://drive.google.com/file/d/1p4Nzn3yUDkkpldp23GnnCv6HaqCo3HBG/view?usp=drive_link
They are not included in this document as there are too many parameters to list.

The resulting LightGBM model has these features as being most important

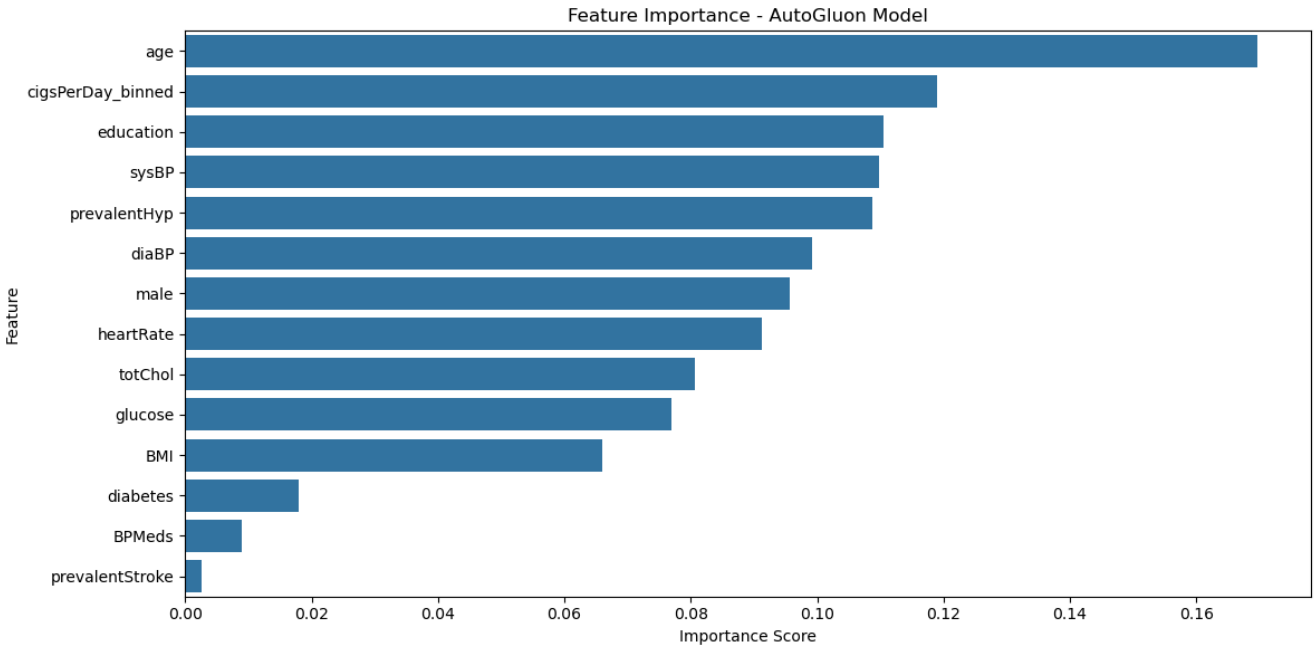

Figure 6: AutoGluon Feature Importance

0.2.3. Results

The comparative results between the LightGBM AutoGluon model, and the reference paper results are detailed in Table 5.

Metric	Alternate Solution	Paper Results	Difference (%)
Accuracy	97.43	94.14	3.49%
Precision	97.96	94.25	3.94%
Recall	96.69	94.06	2.80%
F1-score	97.32	94.06	3.47%
AUC-ROC	99.43	99	0.43%

Table 5: Alternative Solution vs Paper Solution Evaluation Results

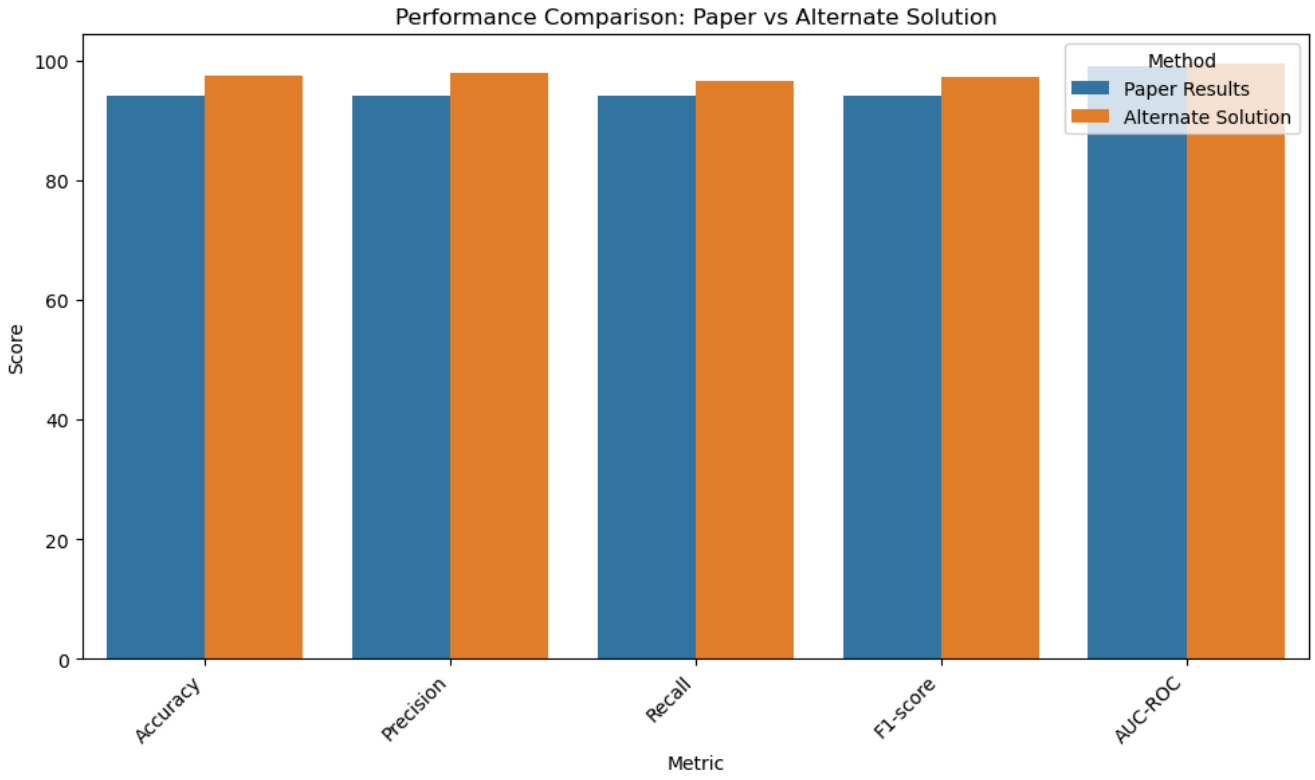

Figure 7: Paper vs Alternate Solution Performance

As can be seen in Table 5, across all measures(apart from AUC) there was a 3% improvement in our alternate solution when comparing the published results of the reference paper. We attribute this improvement to two factors, improved data preprocessing and the use of brute force model architecture and feature search (provided by AutoGluon)

In total AutoGluon generated 28 models that beat the reference paper's f1-score of 94.06. These results align with the findings of [3] that AutoGluon is one of the best performers on tabular data.

Discussion

In order to compare our alternate solution to other implementations, apart from the reference paper, we searched for papers who reported creating machine learning models in order to predict the ten year coronary heart disease of participants and who used the dataset, the Framingham Heart Study.

Paper Title	Best Reported F1-score for TenYearCHD (Model achieving it)
"Visualizing the Full Spectrum Optimization of K-Nearest Neighbors From Data Preprocessing to Hyperparameter Tuning and K-Fold Validation for Cardiovascular Disease Prediction" [4]	0.9608 (Optimized KNN on 8 PCA components)
"FGSCare: A Feature-driven Grid Search-based Machine Learning Framework for Coronary Heart Disease Prediction" [5]	0.457 (Gradient Boosting, on original 16 imputed features)
"Improving Coronary Heart Disease Prediction Using Random Forest with a Modified Minority Synthetic Oversampling Technique on an Imbalanced Dataset" [6]	0.9326 (Random Forest + MMSOT + Grid Search)
"Improved Heart Disease Prediction Using Particle Swarm Optimization Based Stacked Sparse Autoencoder" [7]	0.973 (SSAE + PSO)

Table 6: Comparable Studies on FHS Dataset

As can be seen from the Table 6, our alternate solution for this machine learning task has compared very favourably to other studies generating predictions on the same dataset. Our f1-score of 97.32 exceeded or matched all studies in this search.

Future work that could be undertaken would be to perform hyperparameter tuning on AutoGluon. An ensemble of AutoGluon and other automl frameworks might be one avenue of investigation. Further refinement of the data processing and transformation of the features may also be advisable.

Part 3 - Video Presentation

The video presentations for this report can be found here - https://youtu.be/67Bv8nP2duY

0.3. Reference

1. Bhagat, M., Sharma, A. & Agarwal, P. An efficient stacking-based ensemble technique for early heart attack prediction. *Multimed Tools Appl* (2024).
2. Erickson, N., Mueller, J., Shirkov, A., Zhang, H., Larroy, P., Li, M., & Smola, A. (2020). *AutoGluon-Tabular: Robust and accurate AutoML for structured data* (arXiv:2003.06505). arXiv. https://arxiv.org/abs/2003.06505
3. Shmuel, A., Glickman, O., & Lazebnik, T. (2024). *A comprehensive benchmark of machine and deep learning across diverse tabular datasets*. arXiv. https://arxiv.org/abs/2408.14817
4. Joseph, J., & Kartheeban, K. (2025). Visualizing the Full Spectrum Optimization of K-Nearest Neighbors From Data Preprocessing to Hyperparameter Tuning and K-Fold Validation for Cardiovascular Disease Prediction. Informatica, 49(2). https://doi.org/10.31449/inf.v49i2.7774
5. Zhenqi Li, Jiayi Zhang, Yuchen Wang, Jinjiang You, Junhao Zhong, & Soumyabrata Dev. (2025). FGSCare: A Feature-driven Grid Search-based Machine Learning Framework for Coronary Heart Disease Prediction. https://doi.org/10.13140/RG.2.2.19089.54884
6. M, J. R., K, N. R., & A, K. M. (2025). Improving Coronary Heart Disease Prediction Using Random Forest with a Modified Minority Synthetic Oversampling Technique on an Imbalanced Dataset. International Journal of Electrical and Electronics Engineering, 12(3), 100–113. https://doi.org/10.14445/23488379/IJEEE-V12I3P111
7. Mienye, I. D., & Sun, Y. (2021). Improved Heart Disease Prediction Using Particle Swarm Optimization Based Stacked Sparse Autoencoder. Electronics, 10(19), 2347. https://doi.org/10.3390/electronics10192347