Lifestyle Prediction via Biomarkers

Shirazush Salekin Chowdhury
Bradley Department of Electrical and Computer Engineering
Virginia Tech
Blacksburg, VA
[redacted]@vt.edu

Xiaomeng Wang
Bradley Department of Electrical and Computer Engineering
Virginia Tech
Blacksburg, VA
[redacted]@vt.edu

Alberta Dadeboe
Bradley Department of Electrical and Computer Engineering
Virginia Tech
Blacksburg, VA
[redacted]@vt.edu

***Abstract*—Lifestyle influences not only everyone's quality of life but has a huge impact on the whole society. An unhealthy lifestyle such as habitual smoking and drinking would have a tremendous yet complex impact on one's biochemical system. This project leverages machine learning algorithms to predict lifestyle based on biological measurements and results, aiming to gain better insights into how the stimuli influence an individual's inner body situation, which could be used by health institutions for better in-time interventions and management. This project benchmarked nine machine learning algorithms, including three ensemble classifiers. The Majority Voting Ensemble classifier observed the best performance, which yielded an F1-score of 80.15% and an accuracy of 75.85%. Conversely, the Naïve Bayes and K-Nearest Neighbors algorithms demonstrated the least favorable performance, with an F1-scores of 73.27% and 74.25%, and accuracies of 71.77% and 70.15%**

Keywords—supervised learning, addiction, smoking, drinking, lifestyle, stress, Logistic Regression, Decision Tree, Naïve Bayes, SVM, Random Forest, MLP, Gradient Boosting, Ensemble Classifier

I. SPECIFIC AIMS

The specific aims for our lifestyle prediction project are as follows:

a) Benchmarking of different machine learning algorithms for lifestyle prediction.
   For measuring the performance of the algorithms, we will use the following metrics:
   a) Accuracy
   b) Precision
   c) Recall
   d) F1-score
   e) ROC Curve and AUC-ROC Value
   f) Mean Squared Error
   g) Confusion Matrix
b) Based on the best-performing individual classifiers, develop and evaluate ensemble classifiers.
c) Employing feature engineering techniques to create efficient classifiers.

This report aims to address the following inquiries:

1. **Objective (What):** The classification task involves distinguishing between healthy and unhealthy lifestyles based on biomarkers.
2. **Purpose (Why):** The goal is to develop a machine learning model that facilitates timely interventions and effective management for health institutions.
3. **Approach (How):** The methodology involves optimizing individual machine learning algorithms and subsequently leveraging them to construct an enhanced ensemble classifier.

II. BACKGROUND

Lifestyle has the capability to impact various aspects of a person's life. Numerous benefits can be associated with a lifestyle that is generally portrayed as healthy; improved mental health, reduced risk of mental disease, enhanced physical function and prolonged lifespan are a few of the many benefits. The primary lifestyle-related risk factors that can be easily modified are excessive alcohol consumption, smoking, excessive body weight, and insufficient physical activity [1]. An unhealthy lifestyle leads to delayed seeking for healthcare, resulting in critical health conditions and yet more healthcare expenditure

[2]. [3] shows that heavy smoking has the largest possibility of experiencing at least one of 6 common chronic diseases, including congestive heart failure, chronic obstructive respiratory disease, diabetes, lung cancer, myocardial infarction, and stroke. Monitoring these factors can prove relevant to understanding the extent to which an individual's lifestyle can be considered as healthy or not. The most prominent risk factors affecting a substantial amount of the world's population are smoking and drinking, which can easily be traced using biomarkers that are affected by the extent of indulgence in these activities. In this proposal, we aim to identify the best approach to classify lifestyles into healthy or unhealthy based on biomarkers with the use of machine learning algorithms. We plan to test a group of algorithms and determine the best-performing ones for this specific classification task, after which we will develop ensemble classifiers based on top-performing individual classifiers and apply feature engineering to increase the efficiency of the classifiers.

III. Research Design and Method

The problem at hand is to contribute an efficient and well-performing classification for lifestyle based on the biomarkers that can identify smokers and drinkers to facilitate early intervention. The challenges presented by this classification problem include finding an appropriate dataset with the right features that can produce unbiased results, identifying and optimizing relevant features, tuning hyperparameters of individual classifiers, selecting the right algorithms to benchmark, properly evaluating performance, and developing appropriate ensemble classifiers. As such, we are tasked with developing a method to address these challenges and satisfy our aims. These challenges in effect define our research design objectives. All of the files for this project including vector-based graphs, and Jupyter notebooks can be accessed from this site: http://blog.wangxm.com:8086/2023/12/aml23-final-project-lifestyle-prediction-via-biomarkers/.

A. Related Works

The modern society with an emphasis on efficiency and outcome could easily induce unhealthy lifestyles for people, especially college students. A study shows that many students, including graduate students in universities feel stressed [4]. In [5] the behavior of college students' concurrent consumption of cigarettes and alcohol is studied and shows that stress and social environment among others are the potential risk factors for inducing concurrent drinking and smoking behavior. [6] discusses common factors for the consumption of nicotine and alcohol and how those chemicals affect the brain's biochemical system interactively.

Determining which variables and biomarkers contribute the most to predicting smokers and drinkers can guide our dataset selection process. Authors in [7] identified that biomarkers including HDL-cholesterol, triglyceride, total cholesterol, and body mass index were among the top twenty factors that contributed the most to alcohol prediction. Serum cotinine, urine NNAL and 3HC, nail nicotine, and hair nicotine were identified in [8] as indicators of active smoking. Cholesterol ratios and glucose levels were also found to be credible biomarkers for determining smoking status [9]. Various predictors including medical lab results like fMRI as well as non-medical related data including demographics, and social network data, are gathered to classify urges or behaviors of additions [10]. The study in [9] on the other hand concludes that smoking behavior causes higher aging rates despite factors like cholesterol ratios and fasting glucose levels between smokers and nonsmokers.

Different procedures and approaches for identifying and optimizing relevant features have been proposed with some employing models that facilitate feature selection. [9] trains a set of supervised feed-forward deep neural networks with anonymized blood profile dataset to predict the smoking status and aging effects, 320 random forest models are used for feature selection with maximum available samples, those selected features are used for training and predicting the purpose of the deep neural networks. In [11], a sliding window mechanism is employed to process live raw data to extract discrete time and frequency domain features, with the models producing a high prediction accuracy for smoking activity.

It is also important to understand the effectiveness of different model implementations for this kind of classification to help narrow down our focus on identifying the most relevant ones. A systematic review [10] on the applications of addiction studies using machine learning summarizes 17 publications on addiction. For supervised learning, ensemble methods, regression, and classification are used, unsupervised learning and reinforcement learning are also included in the review. Decision trees, random forests, support vector machines, and neural network-based deep learning models were found to be widely employed for clinical observations because they are efficient at extracting patterns from patient data. [12] [13]. In [14], the K-Nearest Neighbor and logistic regression model performed similarly to decision trees when used for alcohol prediction.

B. Methods

To achieve our specific goals, we have carried out the following procedures:

The raw dataset was prepared through pre-processing, involving data cleansing and in-depth exploration using exploratory data analysis to gain a deeper understanding. Outliers were identified and removed from the dataset. Following the removal of outliers, we proceeded with feature engineering, a process encompassing the encoding of label data and the extraction of new features from the original dataset. The overall data preprocessing steps are depicted in Fig. 1.

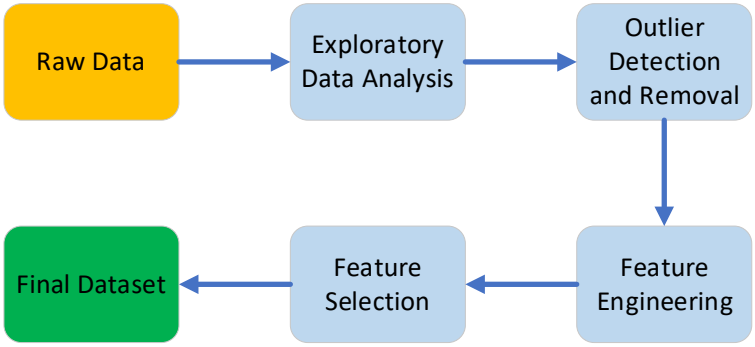


Fig. 1. Overall data preprocessing steps

After that, we applied feature selection techniques, such as Linear Discriminant Analysis (LDA), Random Forest, ANOVA, Kendall's Tau, and Mutual Information Gain to enhance model performance, decrease the possibility of overfitting and improve interpretability.

To fulfill our first aim, the dataset was split into two portions, training, and testing. We experimented with both 80/20 and 70/30 percent training and testing split. To validate our training dataset, we applied k-fold cross-validation for all the algorithms. After the first classification iteration, the performance of the model was checked based on the expected prediction. If the prediction is not accurate enough, hyperparameter tuning is done again until the output reaches expectations. Finally, the performances of the individual models are checked using the data reserved for testing. Because our dataset is quite large, containing approximately 1 million samples, we aimed to utilize as many samples as possible. Initially, we planned to experiment with all samples and 50% of the samples. However, due to constraints in computational resources and the need for hyperparameter tuning, we had to narrow down our sample size to a more manageable 20 thousand samples, randomly selected.

In opting for random sample selection, our goal was to maintain the robustness and representativeness of our training dataset. Additionally, we sought to uphold the statistical validity of our machine learning models, fostering improved generalization and predictive performance on unseen data. This approach

allowed us to strike a balance between the need for comprehensive exploration and the practical constraints imposed by our resources.

We have successfully attained our goal of achieving a minimum accuracy of 70 to 80% for all models through meticulous hyperparameter tuning. Notably, the Majority Voting Classifier, employing 'soft' voting, achieved an impressive F1-score of 80.15%, surpassing the performance of all individual classifiers. This accomplishment marks the success of our project.

For our second goal, we took the best-performing individual classifiers and teamed them up to create a powerful ensemble classifier. The idea behind this was to enhance the model's ability to make predictions and surpass what the individual classifiers could achieve on their own.

Following that, to achieve our third and final goal, we have applied feature engineering techniques to extract new features, for example, the Kidney Function Index was extracted from the serum creatinine, age, and sex features. It helped us to create an efficient classifier to better enhance the outcomes of the lifestyle predictions.

The flowchart in Fig. 2 illustrates our workflow.

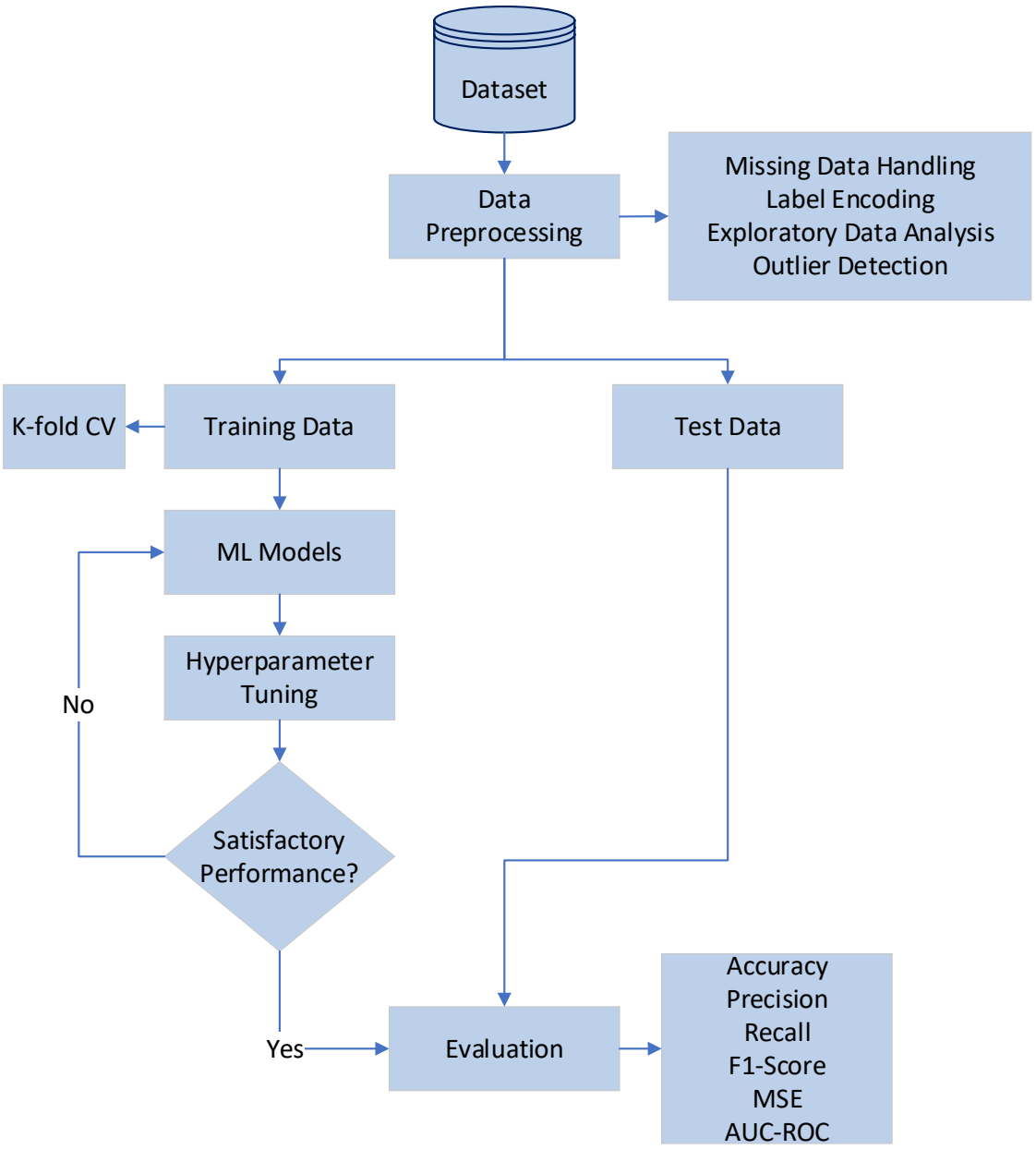
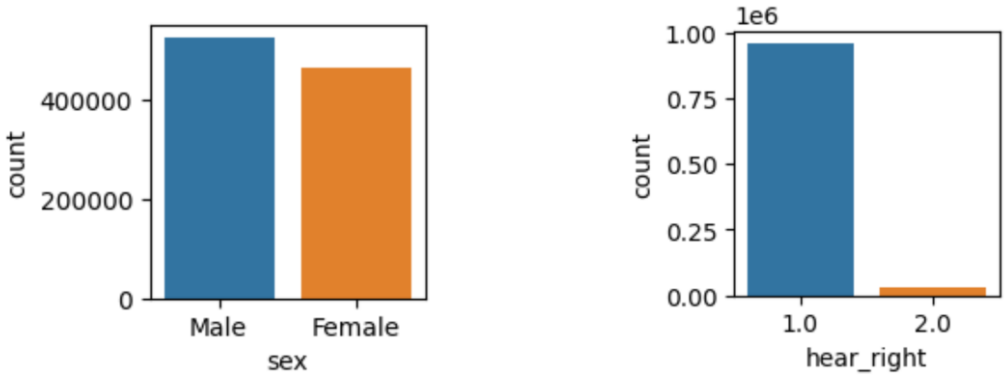


Fig. 2. Flowchart of our proposed workflow

C. Dataset

The dataset, “Smoking and Drinking Dataset with body signal” is taken from Kaggle which is originally collected from the National Health Insurance Service in Korea [30]. It has a total 24 features (sex, age, height, weight, waistline, sight_left, sight_right, hear_left, hear_right, systolic bood pressure (sbp), diastolic blood pressure (dbp), blds, total cholesterol, high-density lipoprotein cholesterol, low-density lipoprotein cholesterol, triglyceride levels, hemoglobin, urine protein, serum creatinine, aspartate aminotransferase, alanine transaminase, gamma-glutamyl transferase, smoking status, alcohol consumption indicator) and around 1 million samples. The dataset manifests inherent imbalances across its features, with discernible disparities among its constituent attributes. Particularly, the feature denoted as "sex" displays a comparatively equitable distribution, while the "hear_left" and “hear_right” features emerge as the most conspicuously imbalanced, as illustrated in Fig. 3(a) and Fig. 3(b) respectively. For our imbalanced dataset, the F1-score is an important performance metric in addition to the accuracy score.



(a) (b)

Fig. 3. (a) Number of male and female participants in the dataset and (b) number of participants having normal (1) and abnormal (2) hearing.

After classifying into the various states of drinking and smoking, we have combined the two features, smoking state, and drinking state, into healthy and unhealthy lifestyles as explained in Table I below:

TABLE I. LIFESTYLE PREDICTION

Smoking State	***Drinking State***	***Lifestyle***
Never	No	Healthy
Used to smoke but quit	No	Healthy
Still Smoke	No	Unhealthy
Never	Yes	Unhealthy
Used to smoke but quit	Yes	Unhealthy
Still Smoke	Yes	Unhealthy

D. Data Preprocessing

Data preprocessing is the foundation for evaluating machine learning algorithms, and properly handling the data before feeding into the models. This part focuses on how the dataset is preprocessed before being fed into the machine-learning models for prediction purposes. The following steps were carried out sequentially:

1. **Changing the dataset format from CSV to Parquet:**
   The original dataset was in CSV format with around a million samples and more than 100 MB file size. We have converted the CSV file to a parquet file. To enhance processing speed and achieve

efficient data compression while retaining the integrity of the original data, we opted to convert the CSV file to the Parquet file format. The advantages of employing the Parquet format are highlighted by a comparison between the CSV and Parquet file formats, detailed in Table II.

TABLE II. CSV VS PARQUET COMPARISON

File Type	*Reading Speed*	*File Size*
CSV	1.71 seconds	104.49 MB
Parquet	70.6 milliseconds	17.09 MB

From Table II, we can conclude that the parquet file format is 22.22 times faster and 6.11 times more compressed compared to the CSV file format while preserving the original data.

2. Outlier Detection and Removal

Outliers were identified in the dataset and addressed using two distinct methods:

a) Visual examination of data through box plots, utilizing the Interquartile Range (IQR)
b) Implementation of Z-Score

a) Visual examination of data through box plots, utilizing the Interquartile Range (IQR)

We have employed Tukey's Fence Method to identify potential outliers in the dataset based on the spread of the data. In Tukey's boxplot, outliers are identified based on the interquartile range (IQR), which is the range between the first quartile (Q1) and the third quartile (Q3) of the data. The "fences" in Tukey's method are used to determine the range within which data points are considered typical, and points beyond these fences are flagged as potential outliers. The upper and lower bound of the fence is defined as follows:

Upper Bound: Q3 + 1.5 x IQR
Lower Bound: Q1 – 1.5 x IQR

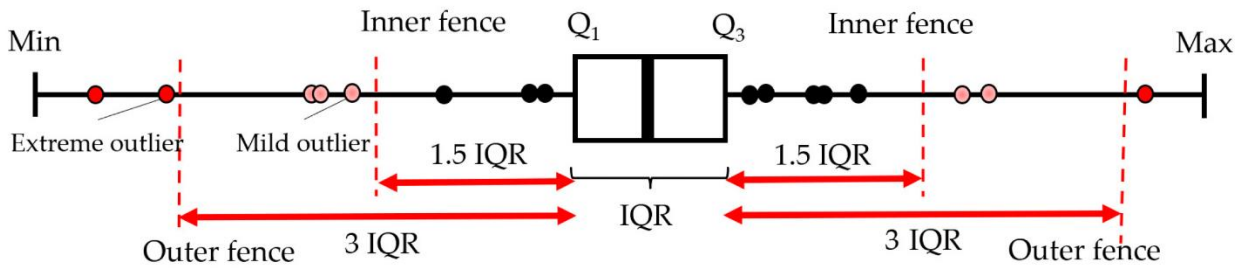


Fig. 4. Tukey's Fence Method for Outlier Detection [32]

We have found outliers in 15 out of 24 features. The features with outliers are: waistline, sight_left, sight_right, sbp, dbp, blds, tot_chole, hdl_chol, ldl_cho, triglyceride, urine_protein, serum_creatinine, sgot_ast, sgot_alt, gamma_gtp.

For plotting the boxplot with IQR we have tied considering the first quartile, Q1 as the 25 percentile, and the third quartile, Q3 as the 75 percentile. Fig. 5 shows the boxplot with Tukey's Fence Method for waistline (cm) with 25% to 75% as IQR.

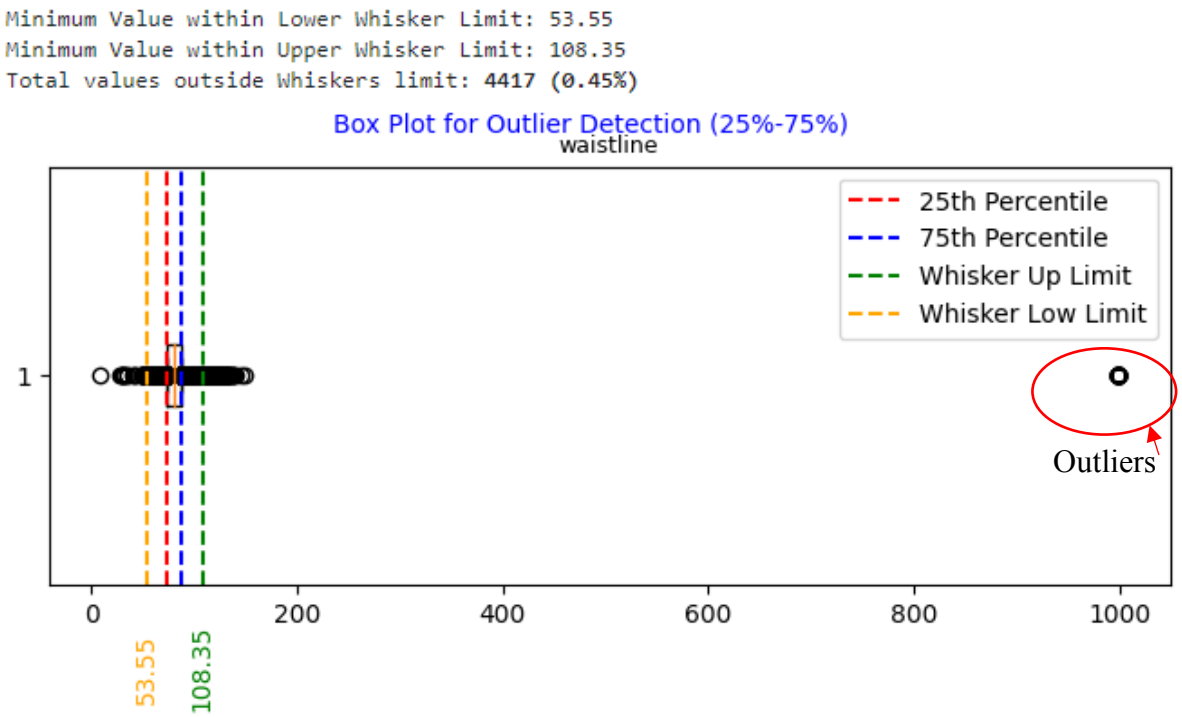


Fig. 5. Boxplot with Tukey's Fence Method for waistline with 25% to 75% as IQR

As seen from the data, there are some clear outliers. However, using 25% to 75% as IQR also considers some real values as outliers, for example, a waistline of more than 108.35 cm is considered to be an outlier. We improved it by considering the IQR range from 5% to 95% by which we can incorporate more than 99% of data from all features. As an example boxplot with Tukey's Fence Method for waistline with 5% to 95% as IQR is depicted in Fig. 6.

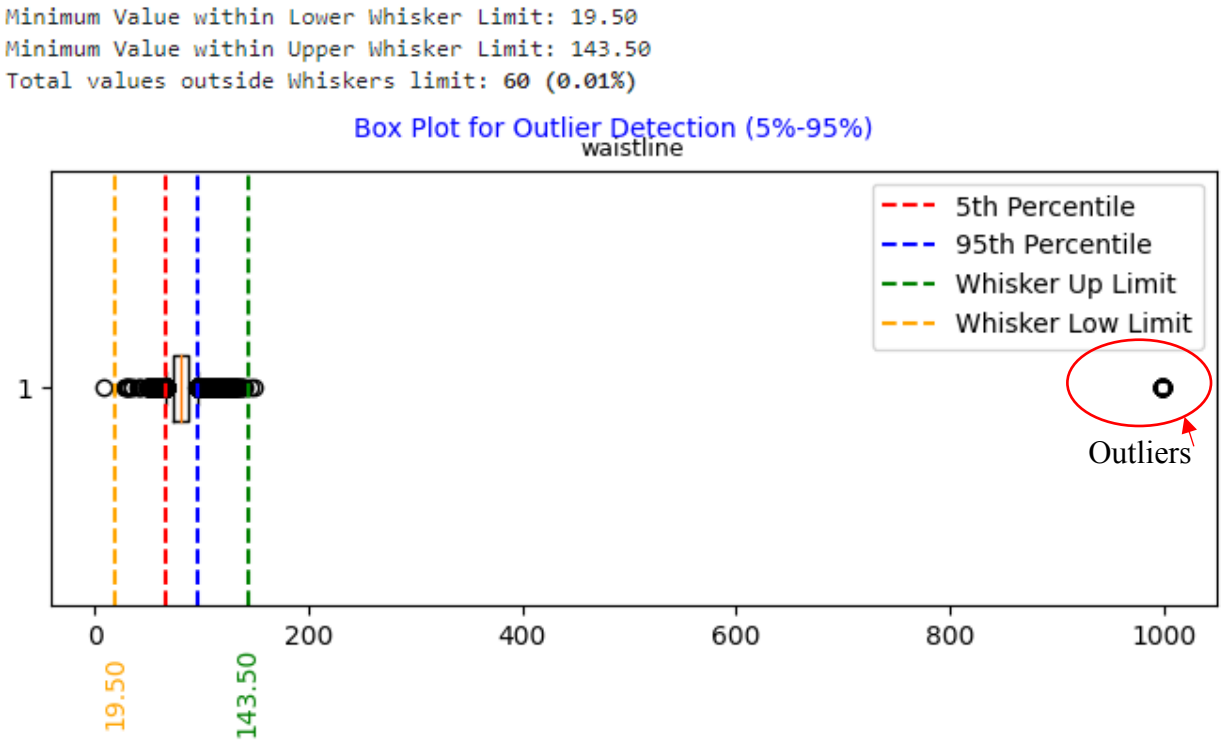


Fig. 6. Boxplot with Tukey's Fence Method for waistline with 5% to 95% as IQR

b) Z-Score

We have also used the Z-score in addition to the IQR method to find the potential outliers and data distribution in our dataset. It is a statistical method used to identify outliers in a dataset by measuring how far individual data points deviate from the mean in terms of standard deviations. Calculating Z-scores for each data point allows us to pinpoint outliers beyond a set threshold. Outliers represent data points significantly differing from the average, and the Z-score provides a quantifiable measure of this deviation, enabling nuanced assessments of data variability.

The Z-Score for a data point, X in a dataset with mean, μ and standard deviation, σ is calculated using the following formula:

$$Z = \frac{x-\mu}{\sigma}$$

We utilized Z-score outlier detection across all features, yielding results comparable to the IQR method. Through experimentation with deviations of 6σ, 8σ, and 10σ, we observed that a 10σ threshold enables the preservation of over 99% of data for each feature while effectively removing outliers. This stringent threshold ensures robust outlier identification, contributing to the refinement of our dataset. Fig. 7 depicts the Z-scores and normal distribution for the waistline measured in cm.

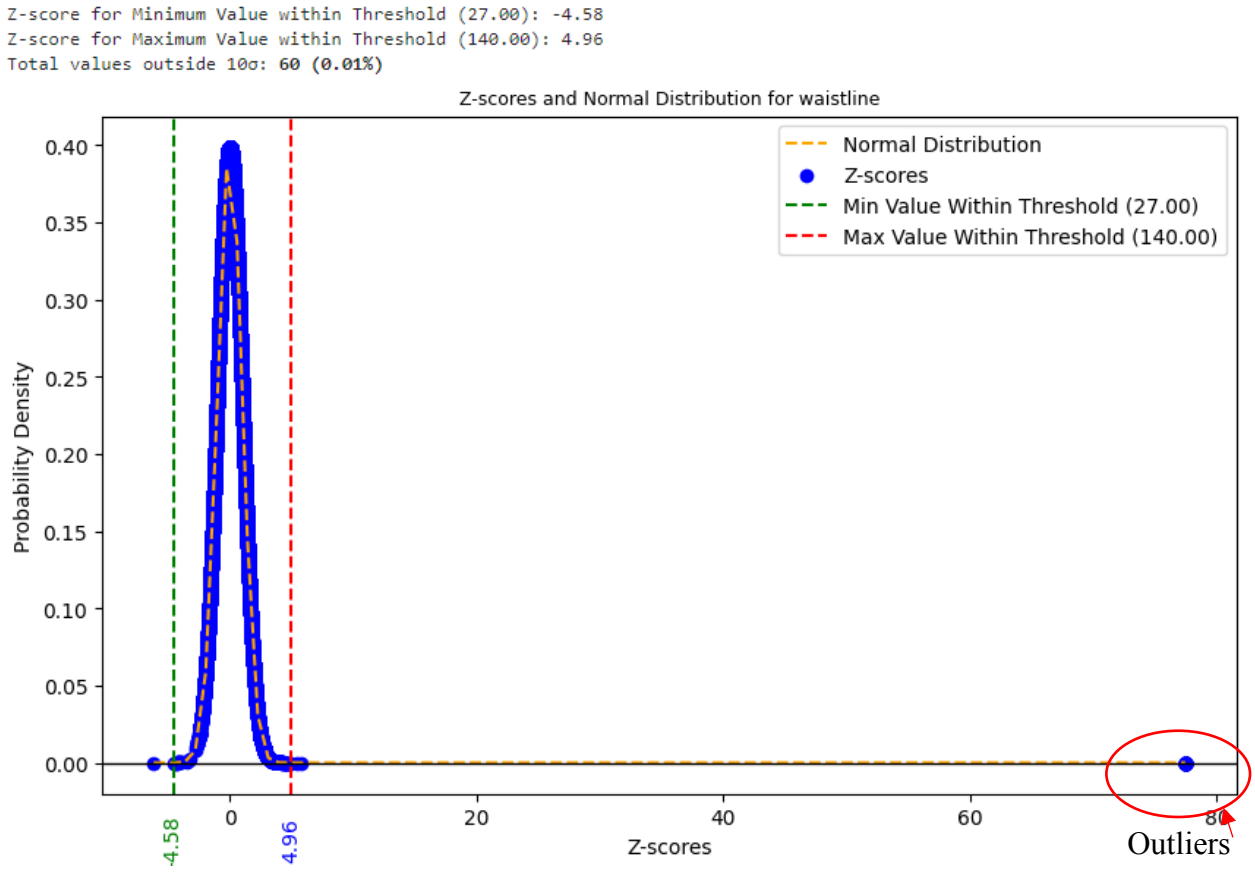


Fig. 7. Z-scores and Normal Distribution for waistline with 10σ Standard Deviation

The outliers are not concentrated on a particular portion of the dataset, rather it is spread over the whole dataset as shown in Fig. 8.

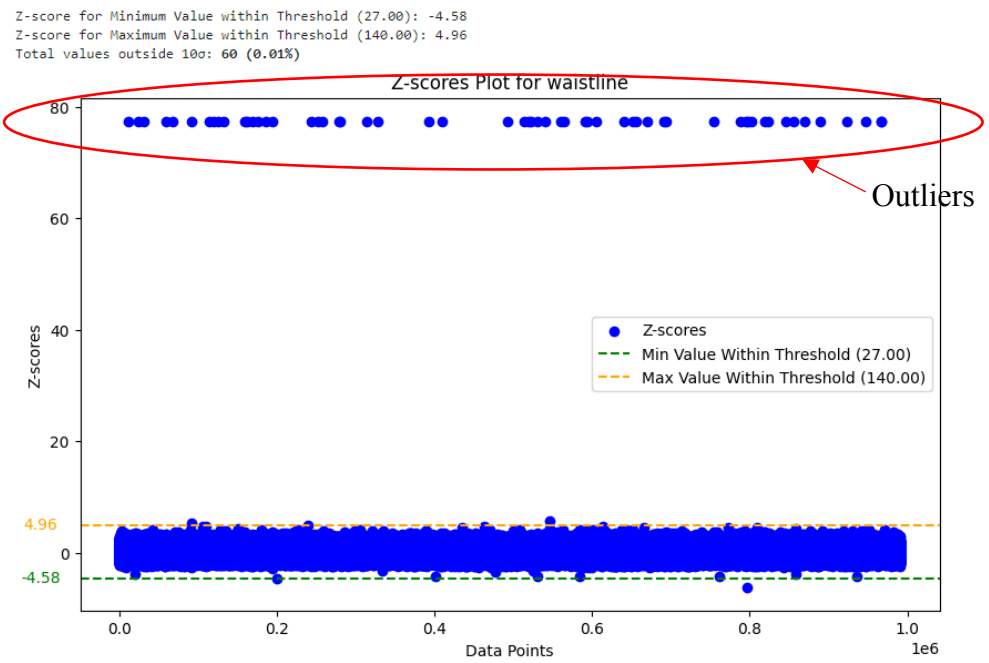


Fig. 8. Z-scores and Outlier Samples

3. Correlation and Nullity Check of the Dataset

We have also checked the correlation between the features to see how they are correlated to each other. Fig. 9 shows the correlation matrix of our dataset.

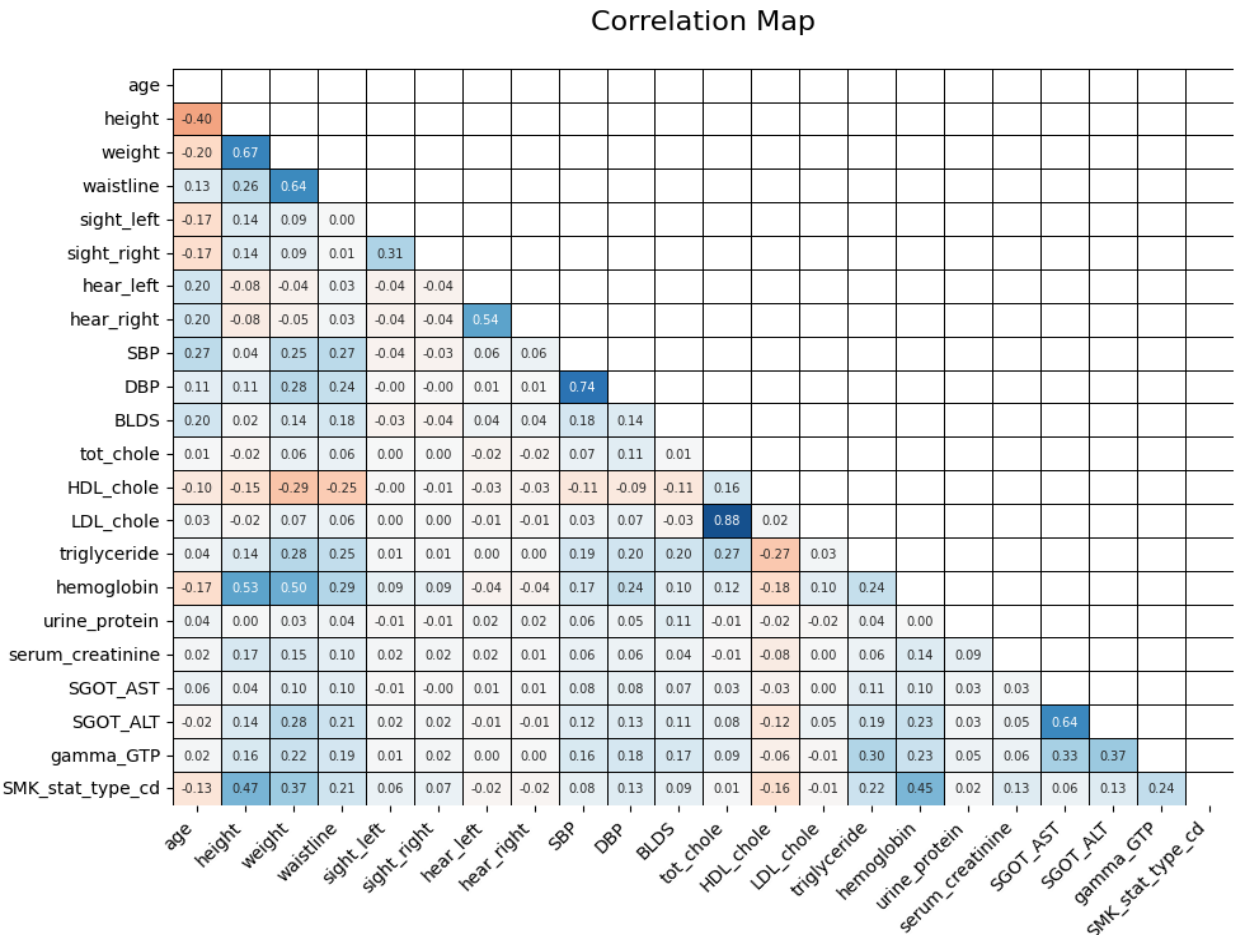


Fig. 9. Correlation Map for the Original Dataset

From the correlation map, we can see that there is not much correlation between the features. (Low-density lipoprotein Cholesterol (LDL_chole) and Total Cholesterol (tot_chole) show the maximum (0.88) amount of correlation as expected since total cholesterol is derived from LDL cholesterol. There is no correlation between the hearing condition and Gamma-Glutamyl Transferase (gamma_GTP).

Finally, we have checked the nullity bar diagram to check whether there is any missing value in our dataset. The nullity bar graph is depicted in Fig. 10.

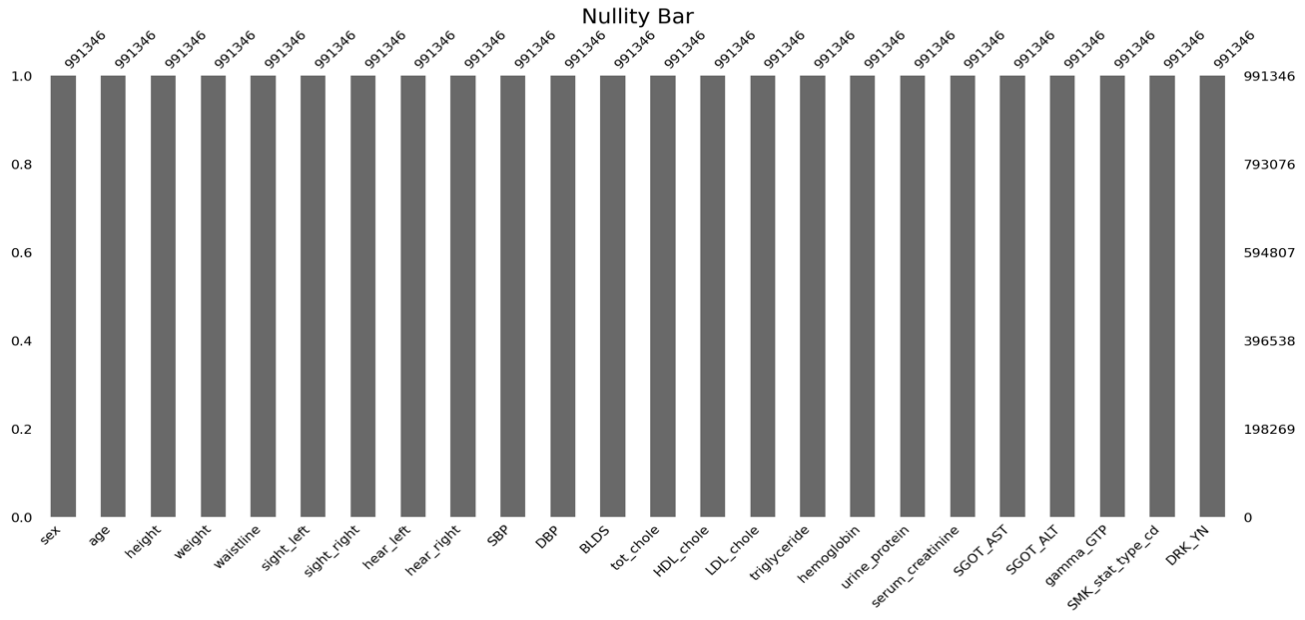


Fig. 10. Nullity Bar Diagram to Check for the Missing Data

From the graph, it can be seen that there is no missing data in our dataset. Therefore, we don't need to do any data imputation for the missing data.

E. Feature Engineering

We will employ feature engineering techniques to optimize features to improve the performance of our models.

Table III shows some features and the equations that we will be using in training our models.

TABLE III. EQUATIONS FOR FEATURE ENGINEERING

Feature	***Key Variables***	***Equation***	***Description***
Kidney Function Index (Glomerular Filtration Rate - eGFR)	Urine Protein, Serum Creatinine (Scr), age, and gender	142 * (min(standardized Scr/K, 1)$^{\alpha}$) * (max(standardized Scr/K, 1) - 1.200) * 0.9938^{Age} * 1.012 [if female] K = 0.7 (females) or 0.9 (males) α = -0.241 (females) or -0.302 (males) [27]	eGFR is an overall kidney function index that gives an assessment of kidney function which can help monitor conditions that can harm kidneys (including high alcohol intake) [28]

Total Cholesterol HDL Ratio	Low-density lipoprotein cholesterol (LDL), high-density lipoprotein cholesterol (HDL), triglycerides(Tri)	Total Cholesterol/HDL or $\frac{(HDL + LDL + 0.2 \cdot Tri)}{HDL}$	Total cholesterol HDL ratio gives in indication of heart health due to presence of good cholesterol levels (HDL) [29]. Cholesterol levels are influenced by lifestyle choices including alcohol consumption.
Cardiovascular Health Index	Triglyceride HDL Ratio, Total Cholesterol HDL Ratio, Blood Pressure Index(BPI)	$\left(\frac{\text{Tri}}{HDL} + \frac{(HDL + LDL + 0.2 \cdot Tri)}{HDL} + \frac{sbp}{dbp}\right)\Big/_3$	This is a metric that sums up the health of the cardiovascular system (CV), taking into account factors including food, physical activity, blood pressure, glucose, total cholesterol, smoking, and body mass index.
Liver Function Index	Enzyme levels: Aspartate Aminotransferase (AST), Alanine Transaminase (ALT), and Gamma-Glutamyl Transferase (GGT)	$\left(\frac{AST}{ALT} + \frac{GGT}{AST}\right)\Big/_2$	Liver function index monitor the enzyme levels produced by the liver to assess its overall health [29].

F. Feature Selection

After the essential data outliner removal and EDA process, feature engineering, the feature selection technique is employed as there are more than 50 features, including numerical features, binary categorical, and one hot categorical feature.

For the feature selection process, several methods including ANOVA, Mutual information gain, Kendall's Tau, Linear Discriminant Analysis (LDA), and Random Forest based feature weights are applied. Table IV shows the equations of the feature selection techniques used.

TABLE IV. FEATURE SELECTION TECHNIQUES

Feature Selection Technique	Equation	Key Variables
ANOVA	1. $F = \frac{MSB}{MSW}$	F = F-statistic MSB = Mean Square Between Groups MSW = Mean Square Within Groups
Mutual Information Gain	2. $MI(X,Y) = \sum_{x \in X} \sum_{y \in Y} P(x,y) \cdot \log\left(\frac{P(x,y)}{P(x) \cdot P(y)}\right)]$	$P(x, y))$ = Joint Probability Mass Function of X and Y $P(x)$ and $P(y)$ = Marginal Probability Mass Functions of X and Y Respectively
Kendall's Tau	3. $\tau = \frac{n_c - n_d}{n}$, where $n = n_c + n_d$	n_c = Number of Concordant Pairs n_d = Number of discordant Pairs
LDA	4. $w = {S_w}^{-1}(m_1 - m_0)$	S_w = Within-Class Scatter Matrix m_1 and m_0 = Means of The Feature Vectors for Class 1 and Class 0, Respectively
Random Forest	5. $Feature\ Importance = \frac{1}{N}\sum_{i=1}^{N} Importance\_i$	N = Number of Decision Trees In The Random Forest

Among all those methods, Linear Discriminant Analysis needs the input predictors to be standardized. The following graph shows the scaled importance values among all features ranked by "ANOVA" scores. It should also be noticed that the absolute values for two metrics which are marked with "ABS" in the legends are using absolute values for better illustration as well as ranking purposes later on. The Linear Discriminant Analysis legend is also annotated with "STD" to indicate the features columns are standardized before the Linear Discriminant Analysis. Since this project aims at binary classification, the parameter is easier to assign for LDA. Even though standardization is experimented on each of the metrics, only the LDA plot has the most significant change in terms of plot shape, without standardization, one feature will have an extremely large weight compared to all other remaining features. The standardization is mandatorily applied for some machine learning models' training and predicting tasks.

After serious consideration of the selected features, categorical features are removed for the following reasons: categorical features in selected features will also have their corresponding numerical features, for example: categorical features like "AGE_CAT" depend on the numerical "age" feature; in addition, such one-hot encoded columns would not provide enough information than numerical columns. As the categorical features generated by original numerical features are removed, 40 features remain. When considering the cumulative contribution of each feature based on all the feature importance measurements and by aiming to maintain at least 80% of all feature weight, 25 out of 40 features will be selected. After consulting with other group members on the feature selection process, three features with higher importance scores: "SIGHT", "AVG_SIGHT" and "gamma_gtp" are replaced by three other features with lower importance scores "CARDIOVASCULAR_HEALTH_INDEX", "WHT_R", "HEAR" with the concern that those replaced features would be represented by the some of the remaining 22 features, thus this is the only part of the process applied manually based on 5 of the feature importance measurements. Fig.11 shows the normalizd importance scores for feature selection.

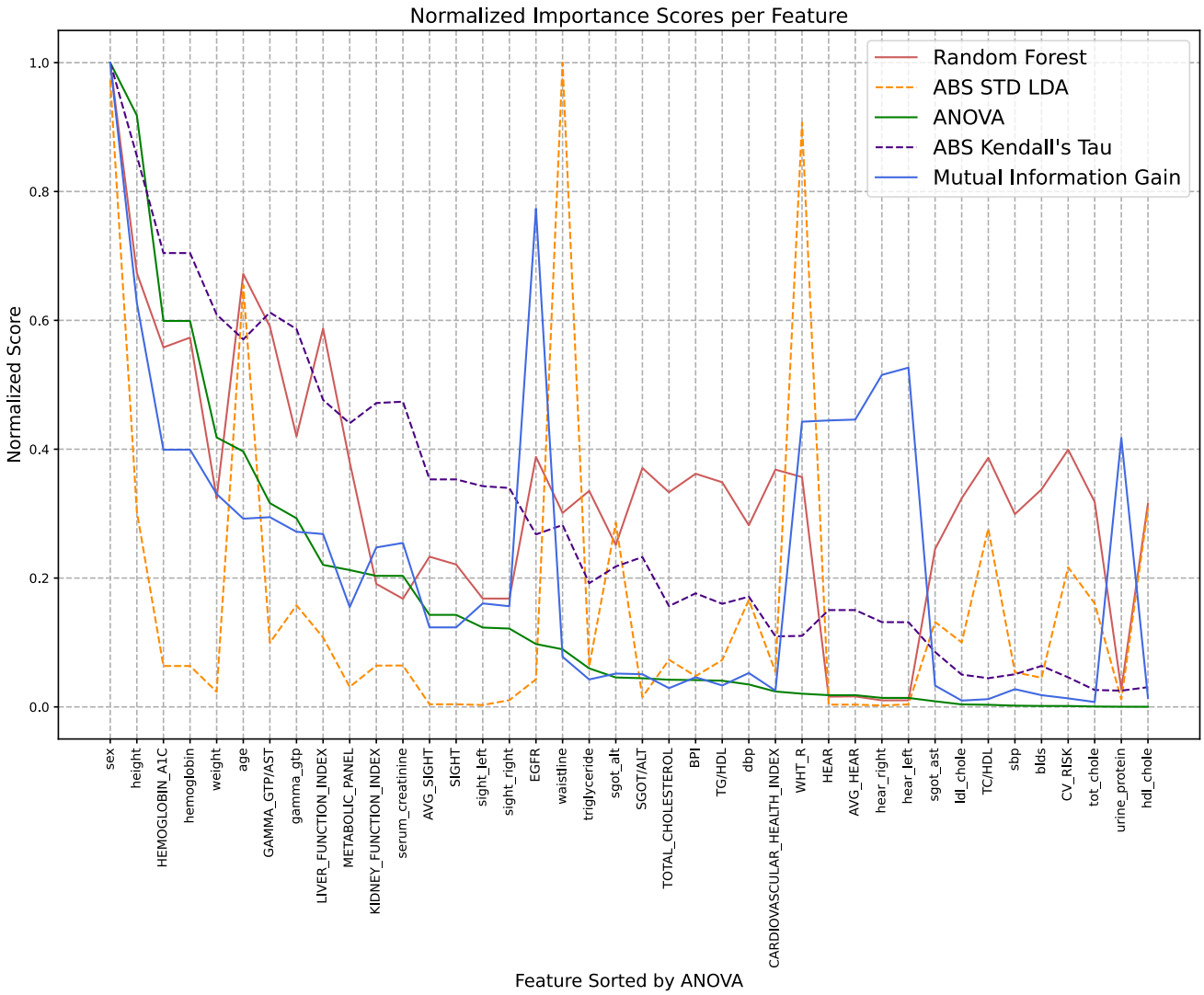


Fig. 11. Normalized Importance Scores for Feature Selection

The feature importance shown above shows that there are features where not all metrics have similar values. For example, one feature's importance is high for Random Forest based feature selection method, but other metrics are low. After serious consideration, this feature will be included and the metric from specific feature selection method will not be regarded as an outliner and discarded. With this idea into consideration, the feature selection based on the feature importance is done as follows:

- Store name of features in a list data structure.
- For each measurement metric, 5 highest features will be selected, append each of the feature in the list data structure.
- If one feature is already stored in the feature list data structure, the next lower ranked feature will be selected and compared with the feature list, until all 5 features are added for a measurement metric.
- As the feature list contains 25 items, sum value for each of the feature will be used to rank the 25 features in the list data structure, those ranked features will be used for testing machine learning models' performance.

G.Selection of models

Different machine learning models will be evaluated to determine the peak-performing ones, informing us of the best approach for healthy or unhealthy lifestyle prediction using the selected dataset. The models were selected based on their unique advantages for classification based on prior literature.

A brief explanation of all the models to be used is described below:

1. **Logistic Regression**
   Logistic regression is a supervised machine learning algorithm mainly used for classification problems. Logistic regression uses the logistic function, also known as the sigmoid function, to model the relationship between the input features and the probability of the binary outcome. The shape of the sigmoid function is S-shaped and is responsible for limiting the output of the cost function so that it can map its output between 0 and 1 [15]. Logistic regression is fundamentally a regression model that employs regression techniques to estimate the probability of a specific data point or entry belonging to a particular category [16]. The sigmoid function is given by [17]:

$$P(Y = 1 \mid X) = \frac{1}{1+e^{-(b_0+b_1X)}} \tag{1}$$

2. **Decision Tree**
   A decision tree is a machine learning algorithm where each node shows a feature (attribute), each link (branch) shows a decision (rule) and each leaf shows an outcome (categorical or continuous value). There are two nodes in a decision tree, namely, the Decision Node and the Leaf Node. Decision nodes have multiple branches and are used to make decisions. Leaf nodes on the other hand give the outcomes of those decisions and do not create any additional branches or decisions [18]. It has a tree-like structure that begins with a root node and proceeds to develop additional branches, forming a structure resembling a tree. Attribute selection measures (ASMs) are techniques employed in decision tree algorithms to identify the most suitable attributes or features for partitioning data at each internal node of the tree. These measures aim to determine the attribute that offers the highest amount of information or discriminative power for classifying or predicting the target variable. There are multiple ways by which ASM can be measured, for example, information gain, Gini index, etc.
   Information gain (IG) quantifies the alteration in entropy that occurs when a dataset is divided using a particular attribute [19]. It is given by:

$$IG(S,A) = E(S) - \sum_{\text{v} \in \text{Values }(A)} \frac{|S_v|}{|S|} E(S_v) \tag{2}$$

   Where E (S) is the current entropy of the subset S, before splitting. |S| is the number of instances in S and A is an attribute in S.
   The Gini index is a metric employed within the CART (Classification and Regression Tree) algorithm to evaluate the level of impurity or purity during the formation of a decision tree and is given by the following equation [20].

$$GINI\ Index\ (L) = 1 - \sum_{i=1}^{j} p_i^2 \tag{3}$$

   Where, p_i is the relative frequency of class i at the node L.

3. **Naïve Bayes**
   The Naïve Bayes classifiers [21] are a group of Bayes' theorem-based probabilistic classifiers. Naïve describes the notion that strong independence assumptions between features are applied in these types of classifiers. Many types of Naïve Bayes classifiers like Gaussian naïve Bayes, multinomial naïve Bayes as well as Bernoulli naïve Bayes classifiers are used depending on the type of data thanks to their simplicity, efficiency as well as scalability.
   Naïve Bayes is shown as follows with C_i for class i of I total classes and $\boldsymbol{x} = (x_1, x_2, \cdots x_N)$ which is the vector that contains N features:

$$p(C_i|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|C_i)p(C_i)}{p(\boldsymbol{x})} \tag{4}$$

Where, $p(C_i|\boldsymbol{x})$ is the posterior probability, $p(C_i)$ is the prior probability, $p(\boldsymbol{x})$ is the evidence and $p(\boldsymbol{x}|C_i)$ is named likelihood. When considering the nominator of the above equation which is related to C_i together with the naïve assumption, it could be written as follows:

$$p(\boldsymbol{x}|C_i)p(C_i) = \frac{1}{p(\boldsymbol{x})} p(C_i) \prod_{n=1}^{N} p(x_n) \tag{5}$$

Thus, the classifier would assign the class as its prediction based on all input features using the following rule to minimize misclassification probability:

$$\hat{y} = \underset{i \in \{1, \cdots, I\}}{\operatorname{argmax}} p(C_i) \prod_{n=1}^{N} p(x_n) \tag{6}$$

4. K-Nearest Neighbors (KNN)

KNN uses proximity to make predictions about a particular data point, making the assumption that similar points are found near each other. The technique finds 'K' training samples with similar attributes to test samples. The training samples are described as Nearest Neighbors. The smaller the value of 'K', the greater the likelihood of overfitting [16] [22].

5. Support Vector Machine (SVM)

The objective of support vector machines is to create a decision boundary, known as a hyperplane, between two classes such that the hyperplane is far enough from the closest data point called support vectors. Support Vector Machines allow predictions to be made from one or more feature vectors [23]. SVM uses linear models to implement non-linear class boundaries.
Given a dataset $(x_1, y_1), \cdots, (x_i, y_i), \cdots, (x_n, y_n)$, $x_i \in R_d$ and $y \in (-1, +1)$
where x_i = feature vector and y_i = class label

The hyperplane can be described as:

$$w \cdot x_i^T + b = 0 \tag{7}$$

where w = weight vector, x = input feature vector, and b = bias

$$\begin{cases} w \cdot x_i^T \geq +1 & if\ y = +1 \\ w \cdot x_i^T \leq -1 & if\ y = -1 \end{cases}$$

SVM finds 'w' and 'b' such that the hyperplane separates the data with a margin of $1/\|w\|^2$
The values of vector x_i for which $|y_i|(wx_i^T + b) = 1$ are termed as support vectors [23].

6. Multi-Layer Perceptron (MLP)

MLP is a neural network model with multiple inputs and output layers where stacked neurons receive inputs, assign weights, and combine the weights and inputs in a weighted sum with arbitrary activation functions to compute the output value [24].

The learning mechanisms used in MLP include propagating the weighted sum linear combination to the next layer up until the output through feed forwarding. The weights needed to minimize the cost function are adjusted through backpropagation. This is done in an iterative fashion until the weights are adjusted and until the cost function is minimized to achieve an optimal accuracy.

In feedforwarding, the Mean Squared Error will be calculated while in backpropagation, the gradient will be computed [24].

7. **Ensemble Learning**
In the realm of machine learning, ensemble methods combine various algorithms to achieve improved results and performance compared to individual algorithms used in isolation. While a statistical ensemble can be infinitely large, a machine learning ensemble consists of a finite set of diverse models, providing greater flexibility and versatility in leveraging these alternative approaches [25]. A Voting Classifier is a machine learning approach that combines multiple models into an ensemble to improve the accuracy of predictions. It consolidates the results, which are then fed as input into the Voting Classifier, aiding in making predictions by following a majority voting approach. This approach avoids the need for individual model creation and their accuracy calculations. It generates predictions by favoring the majority vote for each outcome [26].

IV. Evaluation Procedure and Performance Measures

Performance Metrics [31]

1. **Confusion Matrix:** The confusion matrix summarizes the classified instances in a table so the performance of the model can be easily visualized and assessed. It facilitates easy calculation of accuracy, precision, recall, and F1-score.
We will consider a "healthy" prediction to be positive and an "unhealthy" prediction to be negative. Table V shows the lifestyle prediction confusion matrix.

TABLE V. Lifestyle Prediction Confusion Matrix

		Actual State	
		Healthy	Unhealthy
Predicted Score	Healthy	True Positive (TP)	False Positive (FP)
	Unhealthy	False Negative (FN)	True Negative (TN)

TP = Number of true-positive predictions
FP = Number of false-positive predictions
TN = Number of true-negative predictions
FN = Number of false-negative predictions

2. **Accuracy:** This is a ratio of the correctly classified instances to the total number of instances in the dataset. We expect an accuracy of at least 70 – 80% for well-performing models.

$$Accuracy = \frac{\mathrm{TP+TN}}{\mathrm{TP+TN+FP+FN}} \tag{8}$$

3. **Recall:** This is a ratio of correctly predicted positive instances to the total classified instances.

$$Recall = \frac{TP}{TP+FN} \tag{9}$$

4. **Precision:** This is a ratio of correctly predicted positive instances to the total number of positive instances.

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP+FP}} \tag{10}$$

5. **F1 Score:** This is a harmonic means of recall and precision.

$$F\text{-}measure = 2 \times \frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} \quad (11)$$

6. **ROC Curve and AUC:** An ROC curve, short for the Receiver Operating Characteristic curve, is generated by graphing True Positives (TP) against False Positives (FP) across various threshold settings. The ROC curve is created by plotting the cumulative distribution function of True Positives on the y-axis against the cumulative distribution function of False Positives on the x-axis. The model's ability to distinguish between classes improves with higher Area Under the Curve (AUC) values.
   We will consider an AUC value of at least 0.85 to be appropriate since we are handling medical data.
7. **Mean Squared Error (MSE):** The Mean Squared Error evaluates how close a regression line is to a given set of data points. It calculates the mean of the squared discrepancies between the predicted and actual values. Smaller MSE values signify superior model performance as they reflect reduced prediction errors. MSE is given by the following equation:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \quad (12)$$

   Where, N is the number of samples, y_i is the actual (observed) value of the i^{th} data point and $\hat{y}_i$ is the predicted value of the i^{th} data point.

V. EXPERIMENTAL DESIGN

A. Experimental Setup

a) Quantitative Analysis:

i. ***Stochastic sampling using half of the data:*** Since our dataset contains around a million samples, to minimize the training time, we will stochastically sample 50% of our dataset.

ii. ***Taking the full dataset:*** We will retrain our model with the full dataset to evaluate whether taking the full dataset can improve the model's accuracy or not.

iii. ***Train-test split:*** We will split our dataset between train and split with two different proportions as follows:

   a. 80% training and 20% testing
   b. 70% training and 30% testing

We will try to evaluate which of the above-mentioned ratio work best for our models.

b) Cross-Validation*:*

Cross-validation is a machine-learning technique that splits the data into two different subsets, one is used for training and the other is used for validation. This process is repeated multiple times using different validation sets and averaging the evaluation metrics for improving the performance estimation. There are various types of cross-validation techniques, such as k-fold cross-validation, leave-one-out cross-validation (LOOCV), etc. In our experiment, we will be using the k-fold cross-validation technique. We will do the following k-fold cross-validation techniques in our experiments:

1. 5-fold cross-validation
2. 10-fold cross-validation

c) Server setup:

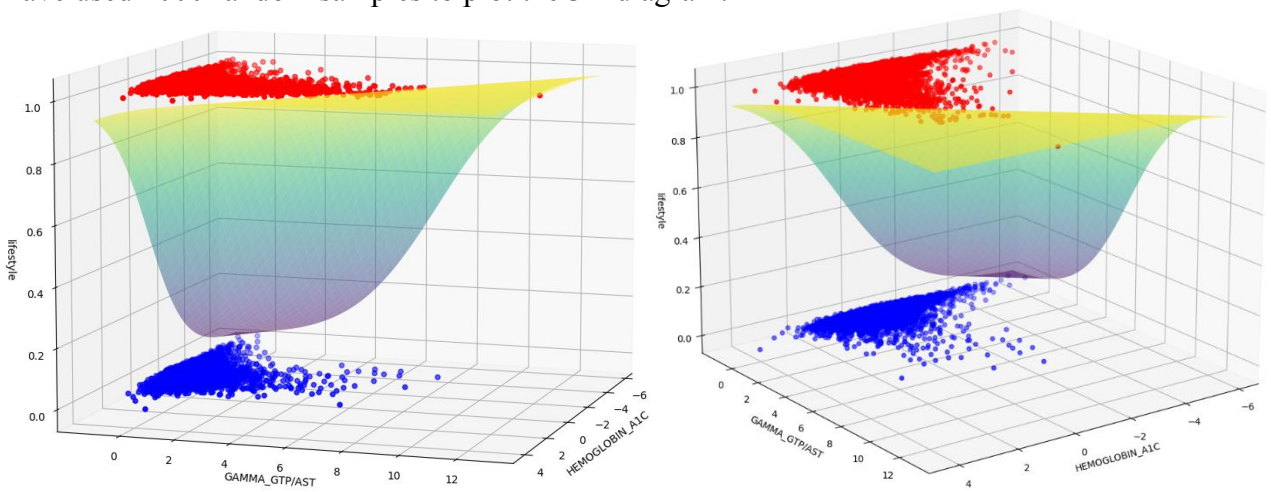
The software environment needs to be installed on remote servers for running machine learning models, especially for users who do not have administrator's privilege and the systemwide Python environment is outdated or the Anaconda suite is not installed. This guide

http://blog.wangxm.com:8086/2023/11/install-and-use-jupyter-notebook shows the steps for setting up the Anaconda machine learning environment for running Python-based Jupyter notebook on a remote server via a web service, the service could be transmitted through a secure shell connection to obtain better computing resources with the specific setup to navigate through the clusters http://blog.wangxm.com:8086/2023/11/tunneling-among-clusters. For this project, python-based machine learning packages like 'pandas', 'numpy', 'matplotlib', 'plotly' as well as the foundational library named sci-kit learn are used. This tutorial http://blog.wangxm.com:8086/2023/10/a-tutorial-on-scikit-learn/ gives step-by-step guidelines on how to use scikit-learn to run basic machine learning models like Naïve Bayes, Logistic Regression, Random Forest as well as Neural Network or Multilayer Perceptron on the Iris dataset for multi-class classification tasks.

B. Experiments on Machine Learning Models

1. Logistic Regression

The first machine learning model that we used was Logistic Regression for classifying our data between healthy and unhealthy lifestyles. Logistic regression uses the logistic function (also known as the sigmoid function) to model the probability of an instance belonging to the positive class. Fig. 12 illustrates the sigmoid decision boundary between Gamma_GTP/AST, Hemoglobin_A1C for two different angles. We have used 1000 random samples to plot the 3D diagram.



(a) (b)

Fig. 12. Features Separated by a Sigmoid Decision Boundary for two different viewing angles (a) Elev: 9°, Azim: 18° (b) Elev: 18°, Azim: 54°

For logistic regression algorithm we have executed several experiments to achieve good accuracy and F1-scores. The following experiments were carried out:

a) Conducting a quantitative analysis on both the entire dataset and a randomly sampled 50% subset.
b) Exploring variations with all 25 features, 13 features (50% subset), and 7 features (25% subset).
c) Utilizing different train vs. test splitting ratios, specifically 80/20 and 70/30.
d) Implementing 5-fold and 10-fold cross-validation techniques.
e) Performing hyperparameter tuning through Grid Search CV, Randomized Search CV, and Bayesian Search CV.
f) Employing L1 (Lasso) and L2 (Ridge) regularization techniques to mitigate overfitting in the model.

From our experiments we had found that the 80/20 train-test split ratio exhibited superior performance compared to the 70/30 split ratio. The application of L1 (Lasso) and L2 (Ridge) regularization, coupled

with 5-fold and 10-fold cross-validation, yielded nearly identical results. Moreover, the performance metrics remained consistent across hyperparameter tuning methods, including Grid Search CV, Randomized Search CV, and Bayesian Search CV. Consequently, we have reported the results in Table VI based on the 80/20 train-test split ratio for the 50% of our dataset, employing 5-fold L1 regularization and the Bayesian Search hyperparameter tuning method. We also showed the results for taking all 25 features, 13 features, and 7 features.

TABLE VI. PERFORMANCE METRICS FOR LOGISTIC REGRESSION ALGORITHM

	Bayes 25 Features	**Bayes 13 Features**	**Bayes 7 Features**
C	0.0264	0.0026	0.0125
Accuracy	74.96%	74.74%	74.51%
Precision	76.69%	76.53%	76.52%
Recall	78.04%	77.75%	77.15%
F1	**77.36%**	77.14%	76.84%
MSE	0.2504	0.2526	0.2549
AUC-ROC	0.8220	0.8187	0.8144
Confusion Matrix	[30772 12229] [11463 40663]	[30572 12429] [11598 40528]	[30661 12340] [11910 40216]

From the table above, we can conclude that the performance remains similar when we use at least 50% features from our dataset. After experimenting with different parameters as discussed before, we achieved 77.36% and 74.96% F1-score and accuracy respectively. The regularization parameter "C" was in range from 0.0026 to 0.0264 indicating a stronger regularization. In other words, the algorithm tried to simplify the model by penalizing large coefficients, which helped to prevent overfitting.

The Area under the Receiver Operating Characteristic ROC curve for Logistic Regression indicating its values for three different hyperparameter tuning methods is illustrated in Fig. 13. The highest AUC-ROC value, 0.820, was achieved when considering all 25 features indicating a "good" classification ability. AUC-ROC is a measure of how well a model distinguishes between classes, specifically the trade-off between recall or sensitivity (true positive rate) and fall-out (false positive rate).

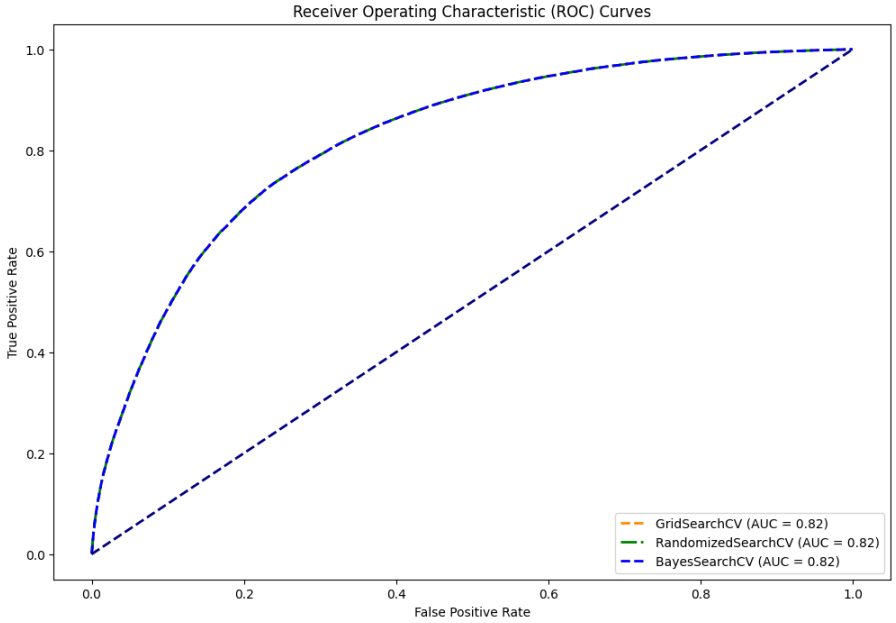


Fig. 13. Receiver Operating Characteristic (ROC) curve for Logistic Regression indicating its values for three different hyperparameter tuning methods

Overall, the performance metrics can be summarized as follows:

1. Accuracy: 74.96%
2. Precision: 76.69%
3. Recall: 78.04%
4. F1-Score: 77.36%
5. MSE: 0.2504
6. AUC-ROC: 0.8220
7. Confusion Matrix
   [30772 12229]
   [11463 40663]

2. Decision Tree

In the implementation of the decision tree algorithm, the following steps were taken:

- Preliminary Training and Performance Evaluation
- Manual Hyperparameter Tuning
- Performance Evaluation with Manually Tuned Hyperparameters
- Hyperparameter Tuning Using GridSearchCV, RandomizedSearchCV, and BayesianCV
- Performance Evaluation with GridSearch, RandomizedSearch, and Bayesian Tuned

a) Preliminary Training and Performance Evaluation

Using a sample size of 20000, a preliminary classification tree was built and trained with the performance from predictions using the testing dataset presented. Fig. 14 shows confusion matrix of preliminary Decision Tree classifier.

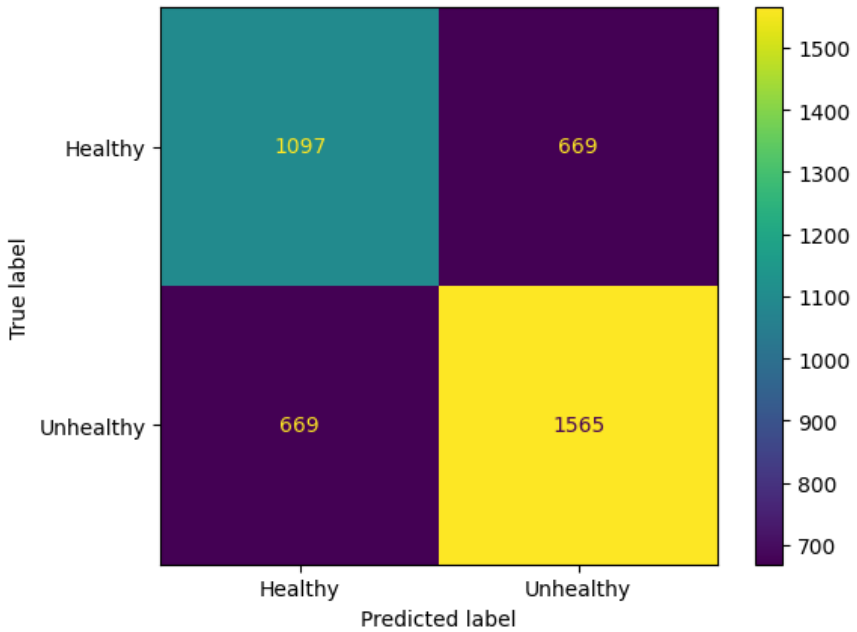


Fig. 14. Confusion Matrix of Preliminary Decision Tree Classifier

b) Manual Hyperparameter Tuning

For manual hyperparameter tuning, the alpha, maximum depth, and maximum number of leaf nodes were selected.

The ideal value for 5-fold (and 10-fold cross validation) for the alpha parameter was determined by extracting the various alpha values accessible for the decision tree (leaving out the maximum alpha value because it would prune all the leaves) and then plotting a graph of the various alpha values with their corresponding mean accuracies to select the alpha value with the highest mean accuracy (Fig. 15 and 16). The best alpha value used for model evaluation was the alpha value with the highest accuracy

score out of the ideal values from 5-fold and 10-fold cross validation. The various alpha values can be found in Table VII.

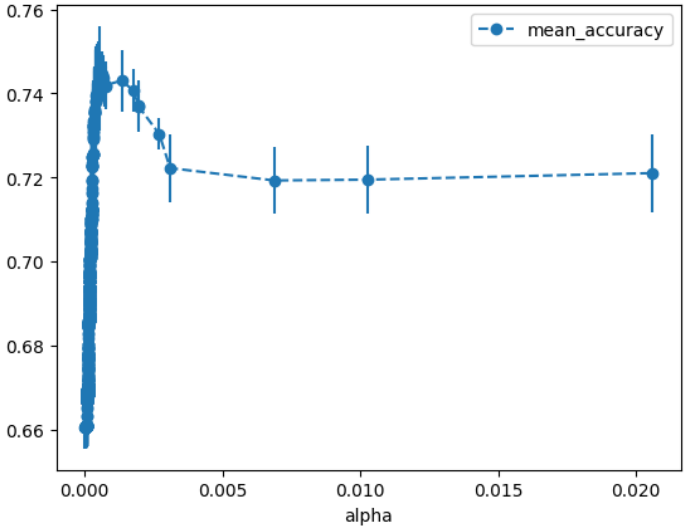


Fig. 15. Graph showing plot of alpha versus mean accuracy for 5-fold cross validation

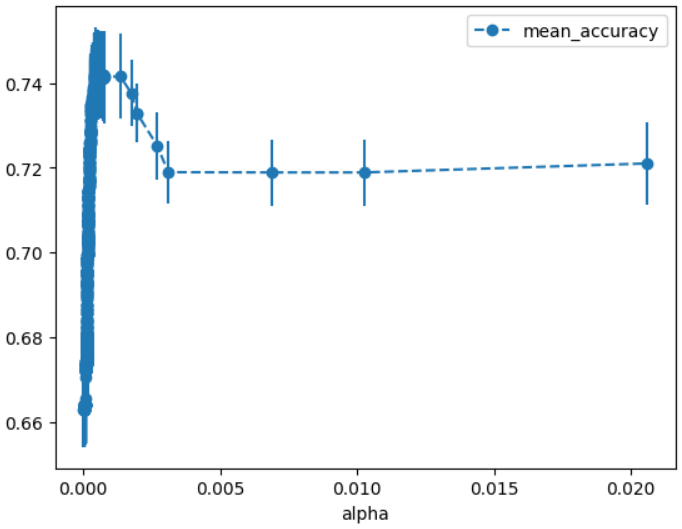


Fig. 16. Graph showing plot of alpha versus mean accuracy for 10-fold cross validation

Determining the ideal maximum depth for 5-fold cross validation (and 10-fold cross validation) involved plotting finding the range of maximum depths available for this dataset (a minimum of 1 to the determined highest maximum depth value possible) and plotting two graphs; a graph of the maximum depth values versus the corresponding accuracies for the training and training datasets and a graph of the maximum depth values versus the corresponding prediction errors for the training and training datasets (Fig. 17 and 18). The ideal value for both 5-fold and 10-fold cross validation was selected based on the maximum accuracy (or minimum error) for the testing data - this is also the point where both training and testing plots intersect. The selected value can be found in Table VII.

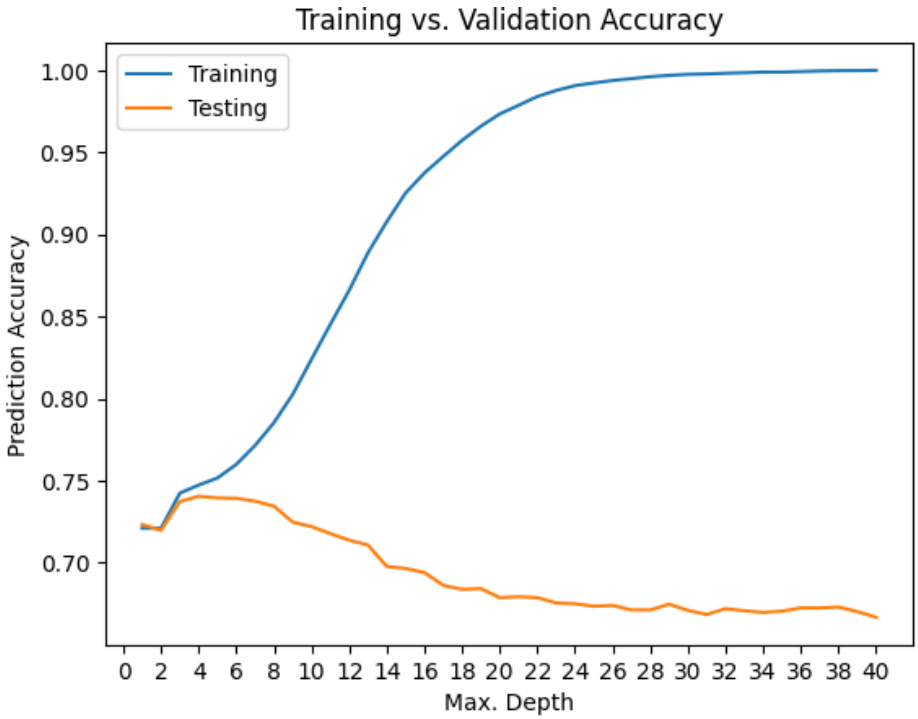


Fig. 17. Graph showing plot of max. depth versus prediction accuracy for training and testing dataset

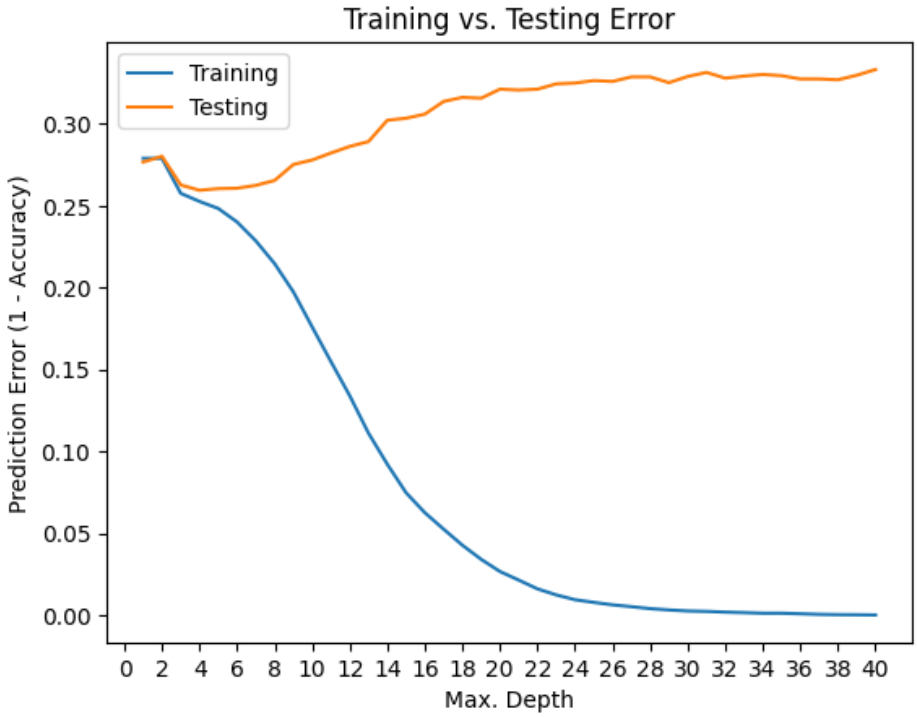


Fig. 18. Graph showing plot of max. depth versus prediction error for training and testing dataset

Similar to the alpha parameter, the ideal value for 5-fold (and 10-fold cross validation) for the maximum leaf nodes parameter was determined by finding the range of maximum leaf nodes possible for this dataset (a minimum of 2 to the determined highest maximum leaf nodes value possible) and then plotting a graph of the various maximum leaf nodes values with their corresponding mean accuracies to select the value with the highest mean accuracy (Fig. 19 and 20). The best maximum leaf nodes value used for model evaluation was the one with the highest accuracy score out of the ideal values from 5-fold and 10-fold cross validation. The various maximum leaf nodes values can be found in Table VII.

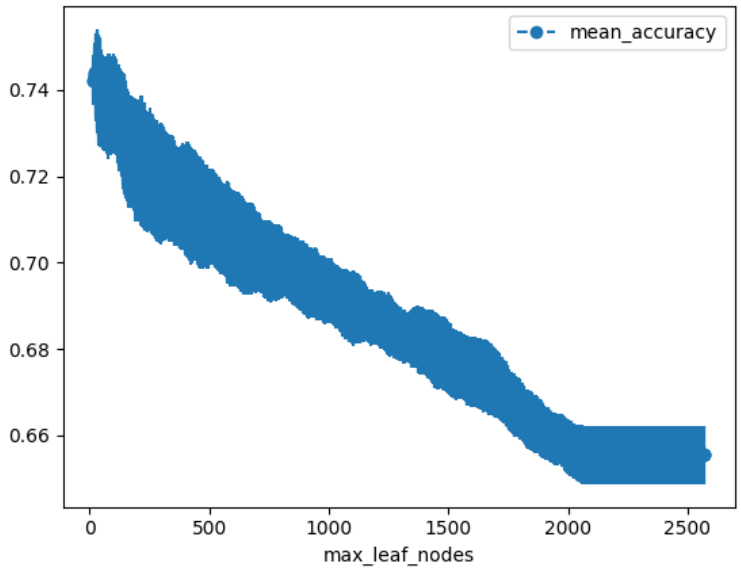


Fig. 19. Graph showing plot of max. leaf nodes versus mean accuracy for 5-fold cross validation

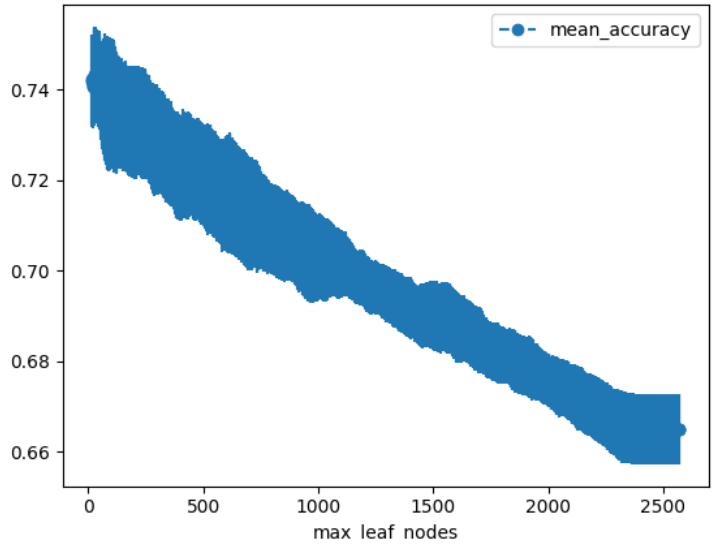


Fig. 20. Graph showing plot of max. leaf nodes versus mean accuracy for 10-fold cross validation

TABLE VII. IDEAL PARAMETERS FOR MANUAL HYPERPARAMETER TUNING

	5-fold cross validation	**Accuracy with 5-fold CV**	**10-fold cross validation**	**Accuracy with 10-fold CV**	**Selected Parameter Value**
Ideal Alpha	0.0007	0.7444	0.0005	0.7426	0.0007
Ideal Max Depth	-	-	-	-	4
Ideal Max. Leaf Nodes	23	0.7440	35	0.7438	23

c) Performance Evaluation with Manually Tuned Hyperparameters

Decision tree classifiers were built each with the considered parameters for hyperparameter tuning set to the selected ideal values. The results are summarized in Table VIII and the individual confusion matrix for the classifier trees built with the ideal alpha, max depth, and max leaf nodes values in Fig. 21, 22 and 23 respectively.

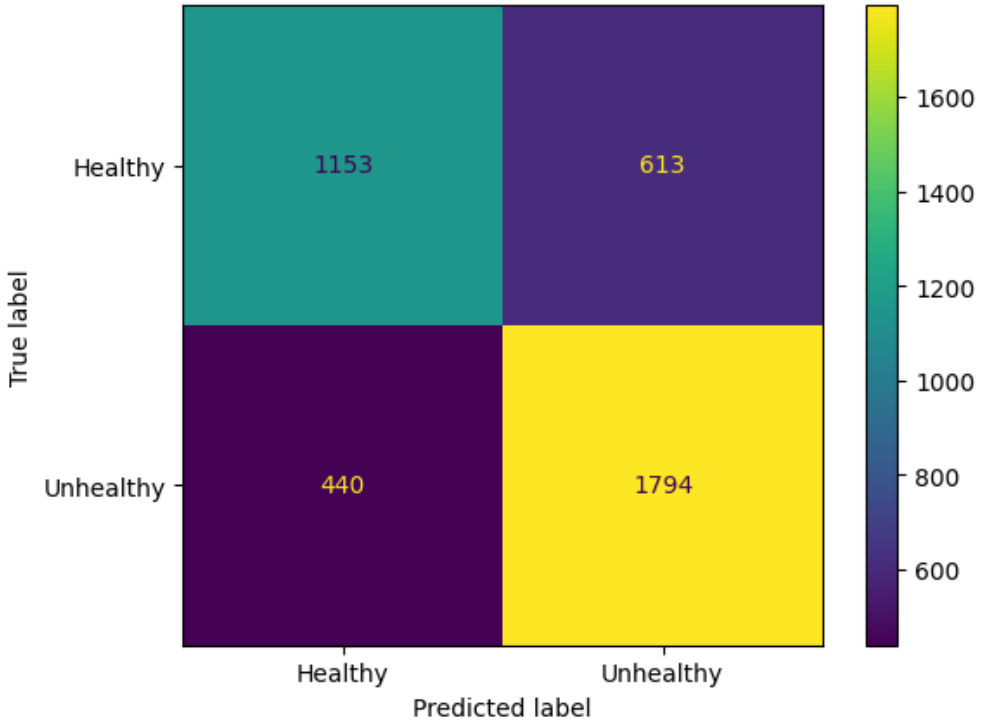


Fig. 21. Confusion Matrix of Decision Tree Classifier with Alpha Set To The Ideal

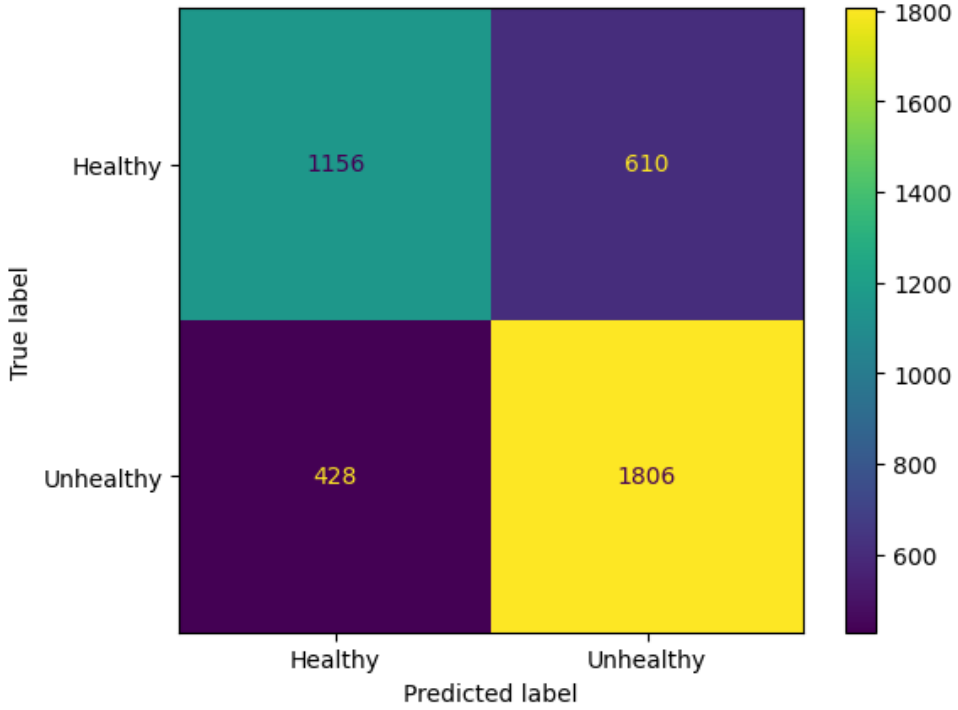


Fig. 22. Confusion Matrix of Decision Tree Classifier with Max. Depth Set To The Ideal

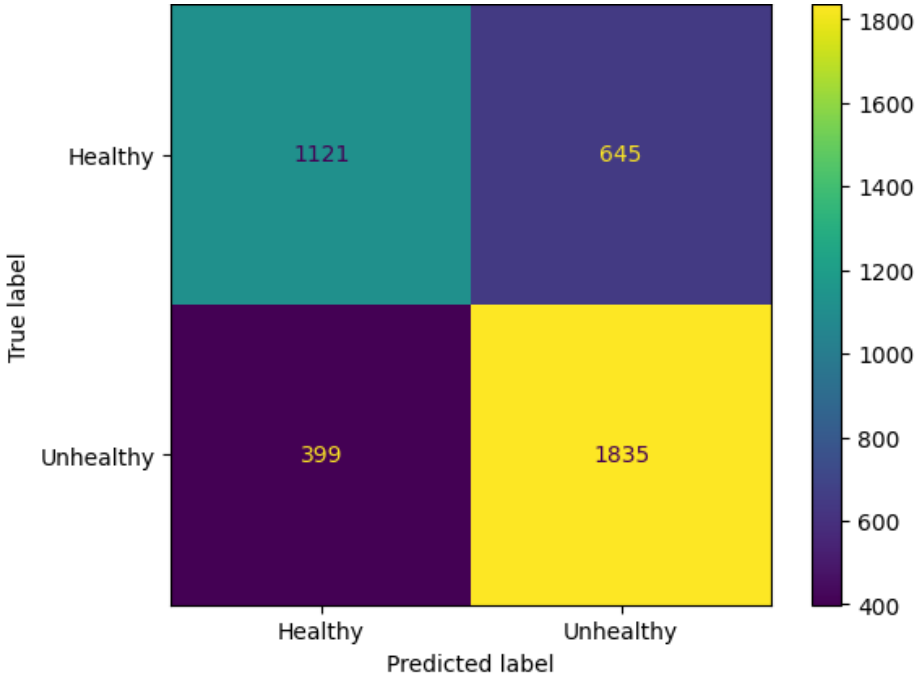


Fig. 23. Confusion Matrix of Decision Tree Classifier with Max. Leaf Nodes Set To The Ideal

TABLE VIII. PERFORMANCE EVALUATION OF DECISION TREE WITH IDEAL HYPERPARAMETER VALUES

	Accuracy	**Precision**	**Recall**	**F1 Score**	**Weighted F1 Score**	**Area Under ROC curve**	**Mean Squared Error**
Ideal Alpha	0.7368	0.7453	0.8030	0.7731	0.7349	0.8095	0.2632
Ideal Max Depth	0.7405	0.7475	0.8084	0.7768	0.7385	0.8108	0.2595
Ideal Max. Leaf Nodes	0.7390	0.7399	0.8214	0.7785	0.7360	0.8118	0.2610

d) Hyperparameter Tuning Using GridSearchCV, RandomizedSearchCV, and BayesianCV

The criterion for node level selection, maximum depth, minimum samples split, and maximum leaf nodes parameters were optimized using grid search by creating an instance of a search object (with a decision tree classifier and the hyperparameters) and conducting a search over the parameter space given for the individual hyperparameters.

The options given for the various hyperparameters are specified below:

Criteria: gini and entropy
Maximum Depth: 1 – 21
Minimum Samples Split: 2 – 11
Maximum Leaf Nodes: 3 - 36

After doing the search, the ideal parameters for the various search techniques are summarized in Table IX below:

TABLE IX. IDEAL HYPERPARAMETER VALUES FOR GRIDSEARCHCV, RANDOMIZEDSEARCHCV, AND BAYESSEARCHCV

	Criterion	Max. Depth	Max. Leaf Nodes	Min. Samples Split
GridSearchCV	gini	5	19	2
RandomizedSearchCV	entropy	8	19	3
BayesSearchCV	gini	5	18	2

e) Performance Evaluation with GridSearch, RandomizedSearch, and Bayesian Tuned Hyperparameters

Decision tree classifiers were built with the considered parameters for hyperparameter tuning set to the selected ideal values derived from each of the parameter space search techniques. The results are summarized in Table X and the individual confusion matrix for the classifier trees built with the ideal parameters from GridSearchCV, RandomizedSearchCV, and BayesSearchCV are shown in Fig. 24, 25 and 26 respectively.

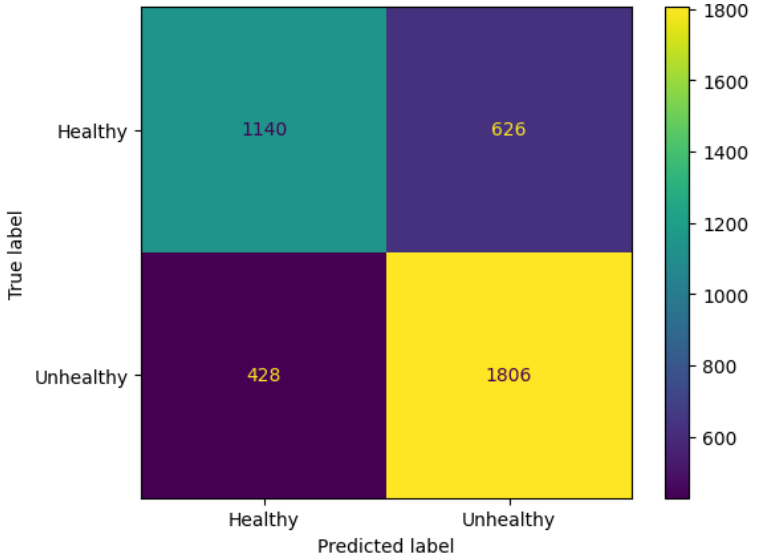


Fig. 24. Confusion Matrix of Decision Tree Classifier with hyperparameters from GridSearchCV

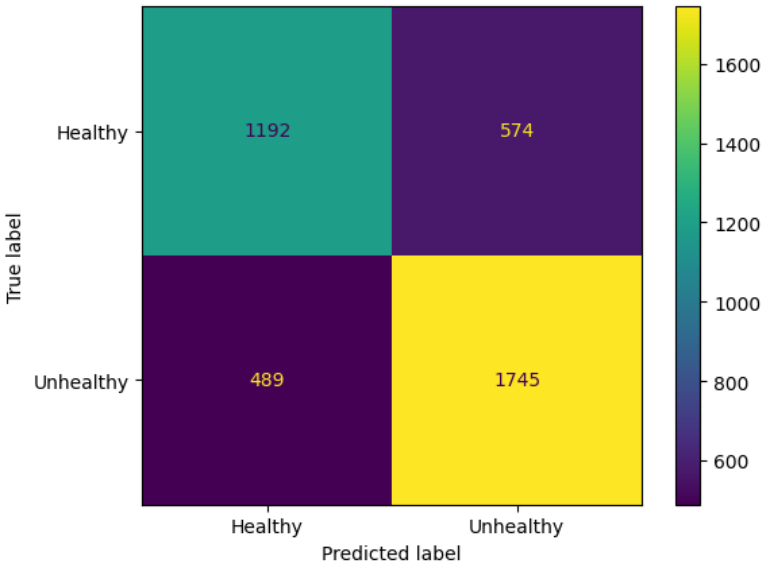


Fig. 25. Confusion Matrix of Decision Tree Classifier with hyperparameters from RandomizedSearchCV

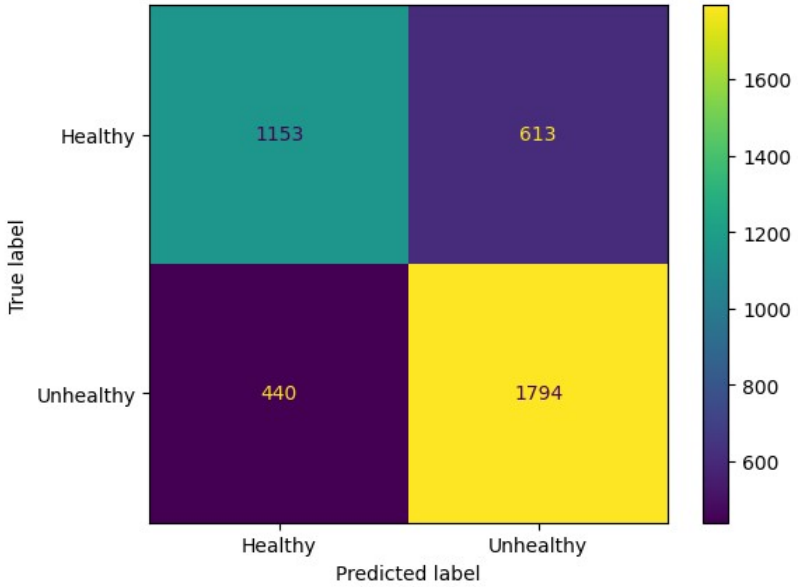


Fig. 26. Confusion Matrix of Decision Tree Classifier with hyperparameters from BayesSearchCV

TABLE X. PERFORMANCE EVALUATION OF DECISION TREE WITH FOR GRIDSEARCHCV, RANDOMIZEDSEARCHCV, AND BAYESSEARCHCV

	Accuracy	**Precision**	**Recall**	**F1 Score**	**Weighted F1 Score**	**Area Under ROC curve**	**Mean Squared Error**
GridSearchCV	0.7365	0.7426	0.8084	0.7741	0.7343	0.8090	0.2635
RandomizedSearchCV	0.7342	0.7525	0.7811	0.7665	0.7335	0.8072	0.2657
BayesSearchCV	0.7368	0.7453	0.8030	0.7731	0.7349	0.8091	0.2632

Overall, the performance metrics can be summarized as follows:

1. Accuracy: 73.68%
2. Precision: 74.53%
3. Recall: 80.30%
4. F1-Score: 77.31%
5. MSE: 0.2632
6. AUC-ROC: 0.8091
7. Confusion Matrix
   [1153 613]
   [440 1794]

3. Naïve Bayes

The Naïve bayes model simply considers that each feature is independent from others, thus the bayes equation could be simplified. Fortunately, the correlation matrix shows that the correlations between features are low and thus even such simple algorithm is expected to achieve reasonable prediction results. For this classification task, most numerical features show gaussian distribution in terms of likelihood, the Gaussian distribution based naïve bayes model is selected in the sci-kit learn library.

In order to have a better understanding on the relationships among features and classification labels is shown as follows in Fig. 27 which relies on Seaborn to generate this pair plot.

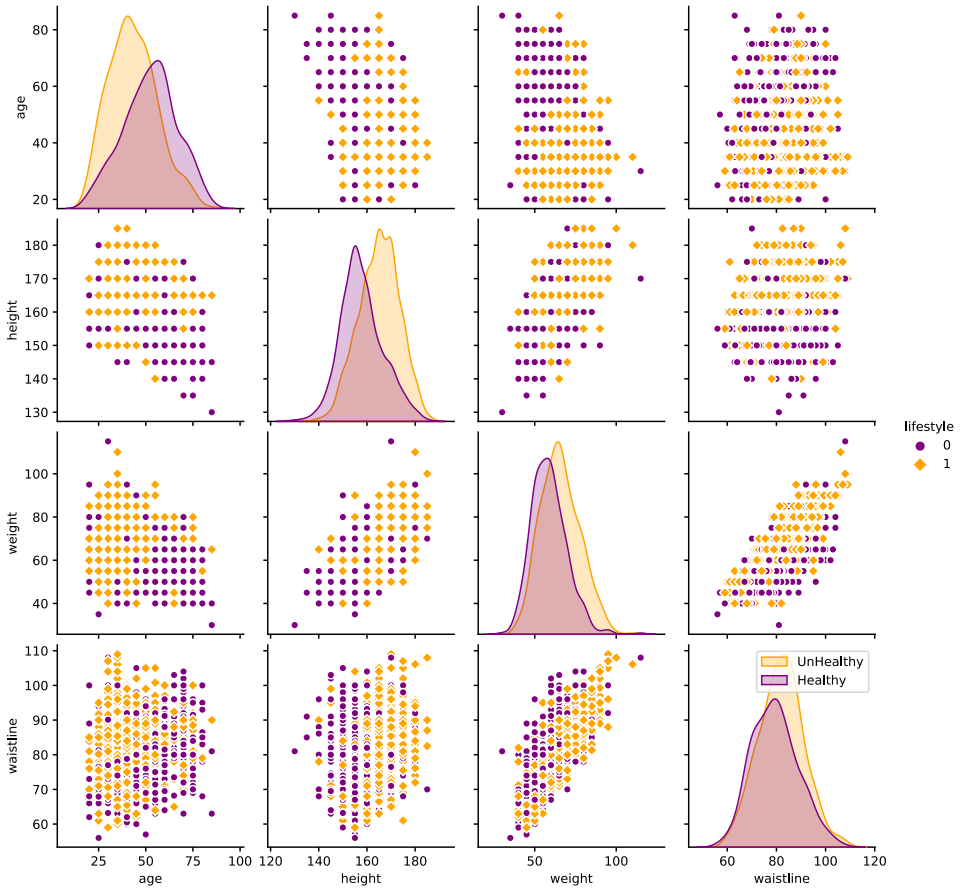


Fig. 27. Pair plot of top 4 featues in the dataset with classification labels

It is straightforward to see the probability distribution function for each class for features “age”, “height”, “weight” and “waistline”. All those show normal distribution for “Healthy” and “Unhealthy” labels. There are a few features manifest different distributions but most of the features have normal distribution probability. Thus for the model of Naïve Bayes, the Gaussian Distribution based Naïve Bayes model is select, in sci-learn it is called “NaiveNB()”. Gaussian distribution parameters, namely mean and deviation are generated from the training data.

Naïve Bayes algorithm has many advantages even through its accuracy is far from accurate: it is very fast to calculate and train the model, thus the tuning is really easy as there’s only one parameter to tune; visualization on Naïve Bayes model is also available and practitioners and readers could gain insights about the dataset.

For illustration’s purpose, two numerical features namely “Height” and “Weight” of all predictors are utilized for prediction based on Naïve Bayes model using Gaussian Distribution are used to predict the lifestyle.

Although Naïve Bayes algorithm is not accurate, we could have an intuitive understanding of the decision-making process by showing the decision boundary of the binary classification, here are some of the illustrations on the decision boundary based on various combinations of features that would help us to learn more about the dataset intuitively.

The following Fig. 28, 29, and 30 show the decision boundary based on 1000 randomly selected samples as using more data on the figure results in a larger file size and longer rendering time which could be around 200MB for the whole dataset yet there’s no significant plot differences in terms of the decision boundary figures.

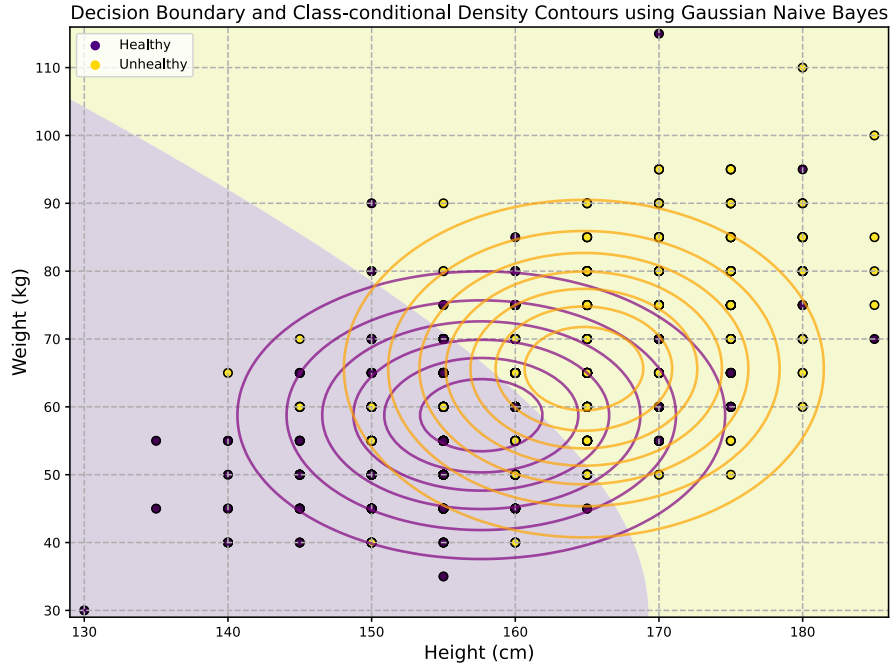


Fig. 28. Decision Boundary and Class-conditional Density Contours using Gaussian Naïve Bayes

Based on the figure with x axis as height in centimeter and weight in weight in kilogram unit, it is straight forward to infer that people who are lighter and shorter would maintain healthy lifestyle, on the contrary people who are heavier and higher struggle to enjoy a healthy lifestyle. It can also infer from the plot that there are fewer people who are lighter and higher or who are heavier and shorter, which makes sense since height and weight has a positive correlation. Although there's a correlation between "Weight" and "Height" as taller people generally weigh more than shorter people, Naïve Bayes algorithm does not consider those correlations. The following 2 figures shows the decision boundary between "Height" and "Age" and "Weight" and "Age".

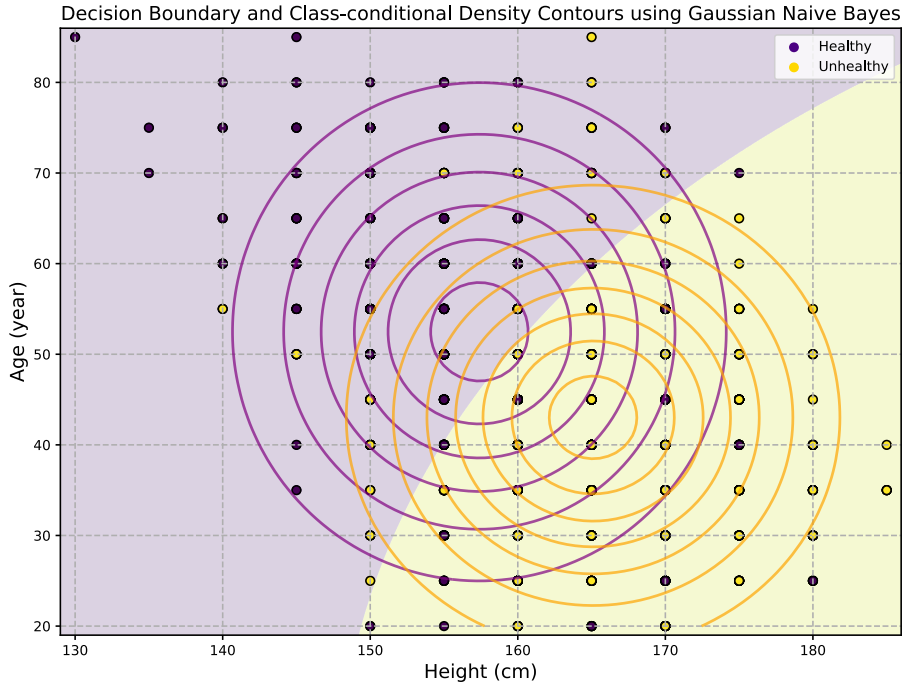


Fig. 29. Decision Boundary and Class-conditional Density Contours using Gaussian Naïve Bayes

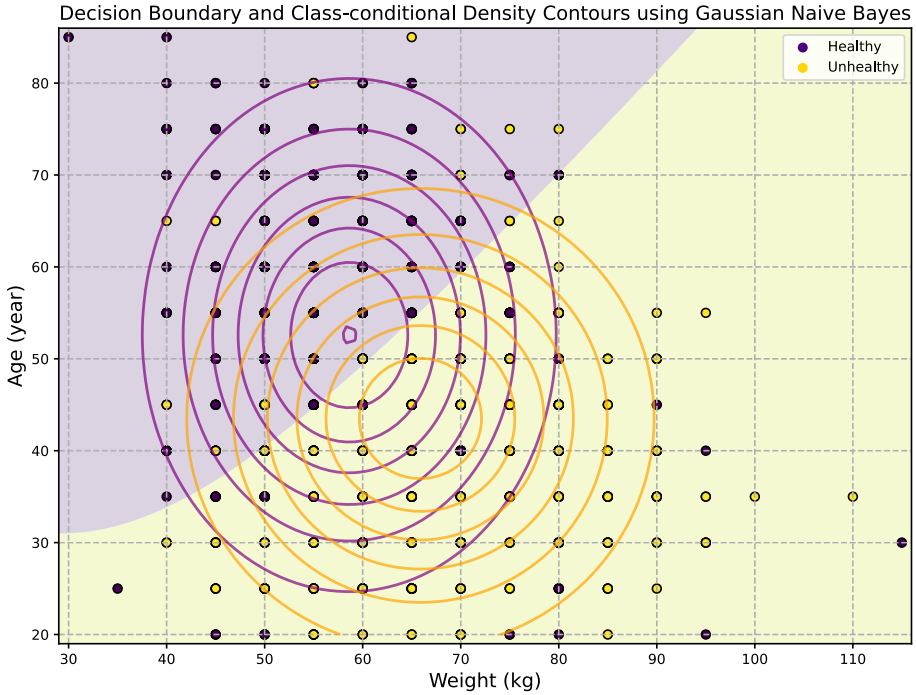


Fig. 30. Decision Boundary and Class-conditional Density Contours using Gaussian Naïve Bayes

For hyperparameter tuning, as we only considered Gaussian based Naïve Bayes, other models like randomized search or Bayes search would not be used here, there's only one parameter to be tuned: *var_smoothing*. It is used to deal with conditions caused by zero variance in the features. The search space for *var_smoothing* is from 10^{-11} to 1. In this hyperparameter tuning, only GridSearch is used thanks to the simplicity of Naïve Bayes algorithm. By varying the hyperparameters of cross-validation value, data split size of 0.7 or 0.8 as well as top features, it is surprising to find that best metrics happen with top 25% features. Table XI shows the performance metrics for the Naïve Bayes Algorithm.

TABLE XI. PERFORMANCE METRICS FOR NAÏVE BAYES ALGORITHM

Naïve Bayes Tuning Results			
	7 Features	13 Features	25 Features
Smoothing	1.26×10^{-5}	2.29×10^{-6}	2.90×10^{-11}
Accuracy	71.77%	71.45%	70.08%
Precision	76.32%	76.39%	77.11%
Recall	70.46%	69.47%	66.63%
F1	73.27%	72.77%	71.49%
MSE	0.2823	0.2855	0.2918
AUC-ROC	0.7191	0.7166	0.7127
Confusion Matrix	[62924 22842] [30867 73622]	[63341 22425] [31902 72587]	[65108 20658] [34863 69626]

There's no big difference for iterating through parameters like cross-validation values, or data splitting ratio, only amount of features taken into consideration would impact the performance as more features results in worse performance. It is possible that 2 or 3 features would have better results than 7 features, which indicates that decision boundary figures shown above could have the best performance and yet they are easy to visualize. Naïve Bayes model can be used as a baseline to judge a dataset and it can be really efficient to do classifications or predictions within a very small amount of time as everyday people apply

this algorithm to evaluate or predict tasks in a quantitative or qualitative manner based on previous events within a short amount of time frame.

Overall, the performance metrics can be summarized as follows:

1. Accuracy: 71.77%
2. Precision: 76.32%
3. Recall: 70.46%
4. F1-Score: 73.27%
5. MSE: 0.2823
6. AUC-ROC: 0.7191
7. Confusion Matrix
   [62924 22842]
   [30867 73622]

4. K-Nearest Neighbor (KNN)

In the implementation of the K-Nearest Neighbors algorithm, the following steps were taken:

- Visualization of the Dataset
- Manual Hyperparameter Optimization
- Performance Evaluation with Manually Tuned Hyperparameters
- Hyperparameter Tuning Using GridSearchCV, RandomizedSearchCV, and BayesianCV
- Performance Evaluation with GridSearch, RandomizedSearch, and Bayesian Tuned Parameters

a) Visualization of the Dataset

The total number of features, 25, were plotted using the seaborn data visualization library based on matplotlib. The graph showcases how the values of different features relate to or impact the values of the target variable. This shows how the values of different features relate to or impact the values of the target variable (dependent variable). This provides an understanding of the impact of the various features on the decision made on whether a person is healthy or not. This is shown in Fig. 31 below.

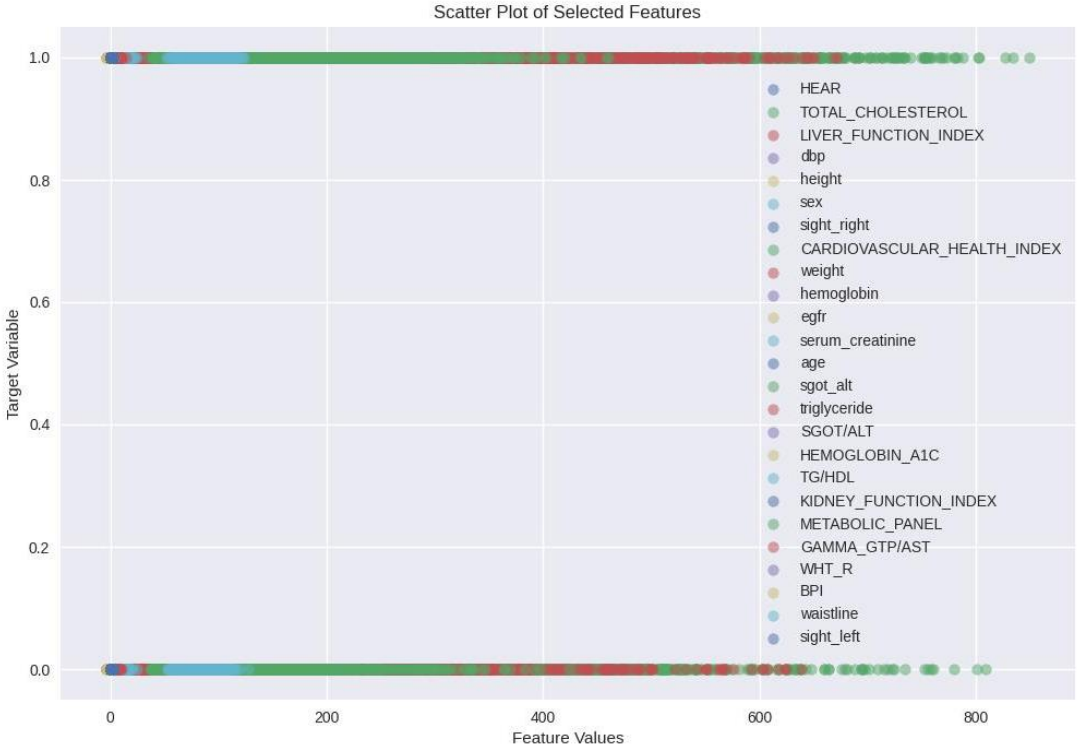


Fig. 31. Dataset Visualization for features used for training KNN model

b) Manual Hyperparameter Optimization

In order to find the ideal value for the number of nearest neighbors, K, the curve of validation error and K for a range of K values from 1 to 60 was plotted (Fig. 32) and the corresponding K value to the point in the curve at which the minimum error occurs (denoted in orange) is chosen as the ideal K value. The ideal K value for this dataset using manual hyperparameter optimization is **39**.

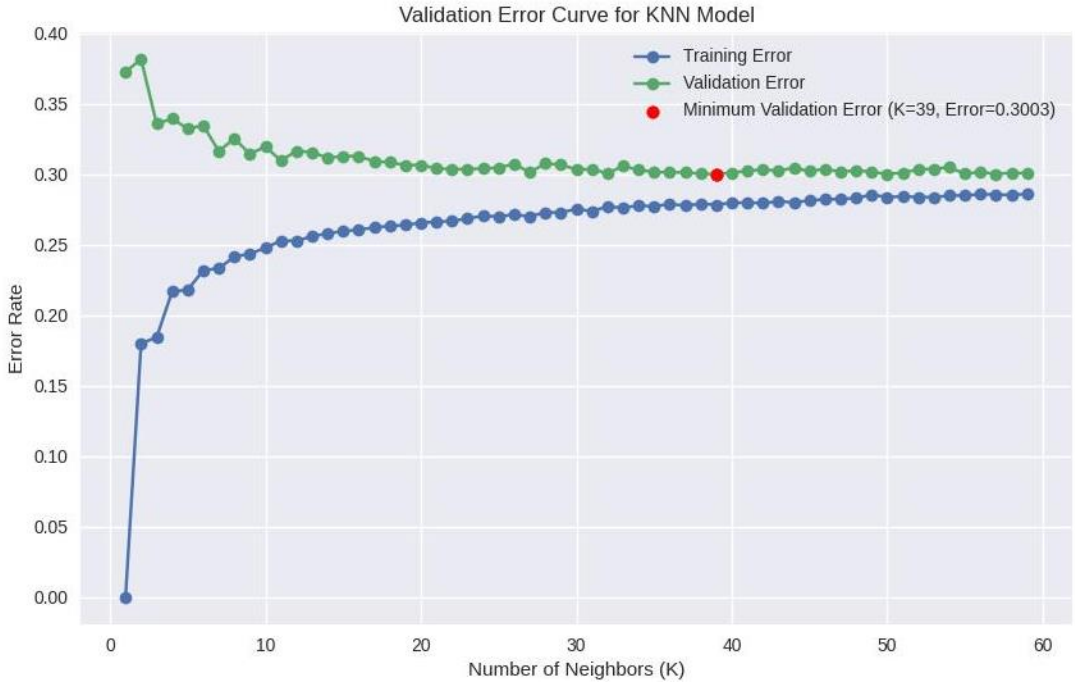


Fig. 32. Graph showing curve of validation error and K

c) Performance Evaluation Using Manually Tuned Hyperparameters

A K-Nearest Neighbors classifier was built with the value of the number of nearest neighbors set to the ideal K value. The result is summarized in Table XII and the confusion matrix for the classifier tree is shown in Fig. 33.

Fig. 33. Confusion Matrix for Manual Hyperparameter Optimization

TABLE XII. PERFORMANCE EVALUATION OF K-KNEAREST NEIGHBORS WITH IDEAL K VALUE

	Ideal K Value	Accuracy	Precision	Recall	F1 Score	Weighted F1 Score	Area Under ROC curve	Mean Squared Error
K	39	0.6997	0.7140	0.7713	0.7416	0.6975	0.7607	0.3003

d) Hyperparameter Tuning Using GridSearchCV, RandomizedSearchCV, and BayesianCV

The criterion for the selection of the K value was optimized using grid search by creating an instance of a search object (with a K-Nearest Neighbors classifier and the hyperparameter) and conducting a search over the parameter space given for the individual hyperparameters.

The range of K values was given from 1 to 60. After doing the search, the ideal parameters for the various search techniques using 5-fold cross validation are summarized in Table XIII and the ideal parameters for the various search techniques using 10-fold cross validation are summarized in Table XIV.

TABLE XIII. IDEAL HYPERPARAMETER VALUES FOR GRIDSEARCHCV, RANDOMIZEDSEARCHCV, AND BAYESSEARCHCV WITH 5-FOLD CROSS VALIDATION

	K value
GridSearchCV	36
RandomizedSearchCV	46
BayesSearchCV	38

TABLE XIV. IDEAL HYPERPARAMETER VALUES FOR GRIDSEARCHCV, RANDOMIZEDSEARCHCV, AND BAYESSEARCHCV WITH 10-FOLD CROSS VALIDATION

	K value
GridSearchCV	44
RandomizedSearchCV	46
BayesSearchCV	44

e) Performance Evaluation with GridSearch, RandomizedSearch, and Bayesian Tuned Parameters

K-Nearest Neighbors classifiers were built with the K value set to the selected ideal values derived from each of the parameter space search techniques. The results for 5-fold cross validation are summarized in Table XV and the individual confusion matrix for the classifier trees built with the ideal parameters from GridSearchCV, RandomizedSearchCV, and BayesSearchCV are shown in Fig. 34, 35, and 36 respectively. The results for 10-fold cross validation are summarized in Table XVI and the individual confusion matrix for the classifier trees are shown in Fig. 37, 38, and 39 respectively.

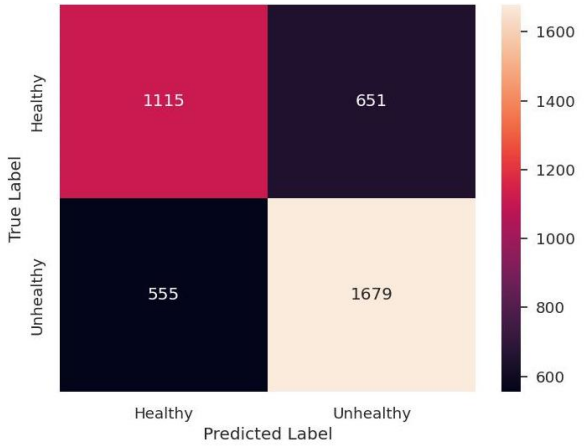


Fig. 34. Confusion Matrix of K-Nearest Neighbours Classifier with hyperparameters from GridSearchCV with 5-fold cross validation

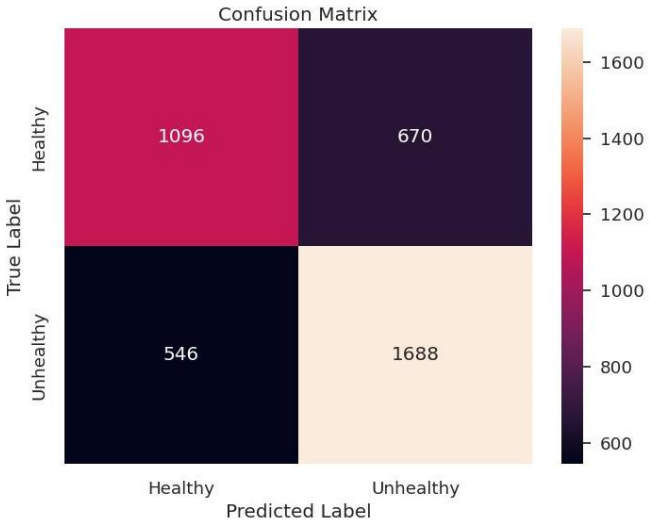


Fig. 35. Confusion Matrix of K-Nearest Neighbours Classifier with hyperparameters from RandomizedSearchCV with 5-fold cross validation

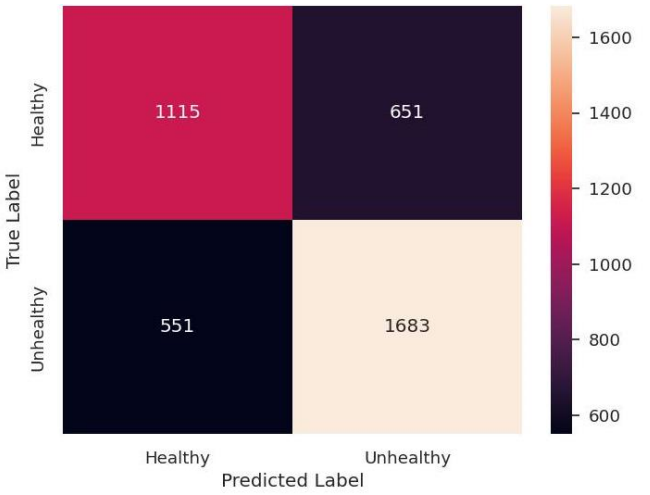


Fig. 36. Confusion Matrix of K-Nearest Neighbours Classifier with hyperparameters from BayesSearchCV with 5-fold cross validation

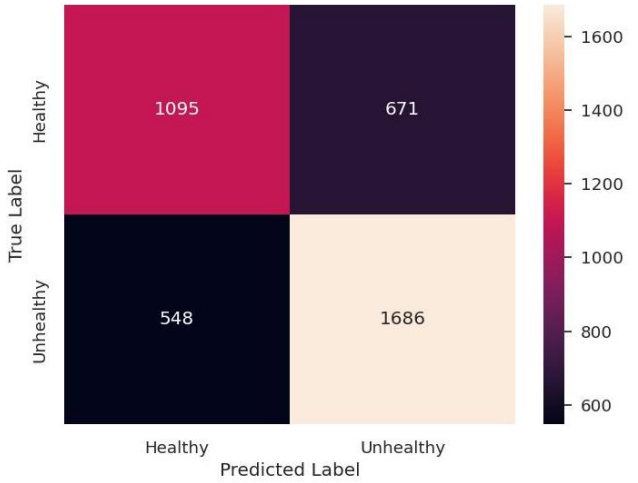


Fig. 37. Confusion Matrix of K-Nearest Neighbours Classifier with hyperparameters from GridSearchCV with 10-fold cross validation

Fig. 38. Confusion Matrix of K-Nearest Neighbours Classifier with hyperparameters from RandomizedSearchCV with 10-fold cross validation

Fig. 39. Confusion Matrix of K-Nearest Neighbours Classifier with hyperparameters from BayesSearchCV with 10-fold cross validation

TABLE XV. PERFORMANCE EVALUATON OF K-NEAREST NEIGHBORS FOR GRIDSEARCHCV, RANDOMIZEDSEARCHCV, AND BAYESSEARCHCV WITH 5-FOLD CROSS VALIDATION

	Accuracy	**Precision**	**Recall**	**F1 Score**	**Weighted F1 Score**	**Area Under ROC curve**	**Mean Squared Error**
GridSearchCV	0.7045	0.7212	0.7515	0.7412	0.7011	0.7605	0.3081
RandomizedSearchCV	0.7031	0.7224	0.7674	0.7419	0.6914	0.7615	0.3065
BayesSearchCV	0.7015	0.7241	0.7554	0.7425	0.7021	0.7684	0.3014

TABLE XVI. PERFORMANCE EVALUATON OF K-NEAREST NEIGHBORS FOR GRIDSEARCHCV, RANDOMIZEDSEARCHCV, AND BAYESSEARCHCV WITH 10-FOLD CROSS VALIDATION

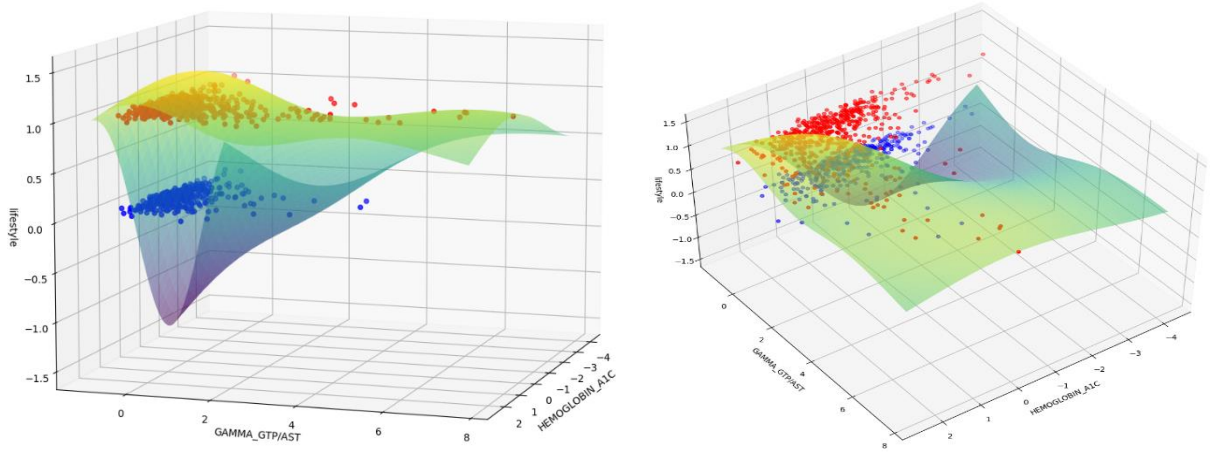
	Accuracy	**Precision**	**Recall**	**F1 Score**	**Weighted F1 Score**	**Area Under ROC curve**	**Mean Squared Error**
GridSearchCV	0.7051	0.7220	0.7518	0.7322	0.6944	0.7619	0.3092
RandomizedSearchCV	0.7047	0.7221	0.7679	0.7435	0.6949	0.7621	0.3041
BayesSearchCV	0.7029	0.7251	0.7567	0.7340	0.6981	0.7645	0.3033

Overall, the performance metrics can be summarized as follows:

1. Accuracy: 70.15%
2. Precision: 72.41%
3. Recall: 75.54%
4. F1: 74.25%
5. MSE: 0.3014
6. AUC-ROC: 0.7684
7. Confusion Matrix
   [1115 651]
   [551 1683]

5. Support Vector Machine (SVM)

In our lifestyle classification model, we identified the Support Vector Machine as one of the top-performing models, exhibiting overall good accuracy and F1-scores. A 3D diagram showing three features (Gamma_GTP/AST, Hemoglobin_A1C, Lifestyle) that are separated by a non-linear decision boundary surface using a Radial Basis Function (RBF) kernel as illustrated in Fig. 40 for two different viewing angles. We have used 1000 random samples to plot the 3D diagram.



(a) (b)

Fig. 40. Features Separated by Non-Linear Decision Boundary Surface using RBF Kernel for two different viewing angles (a) Elev: 9°, Azim: 18° (b) Elev: 45°, Azim: 54°

The most striking feature of SVM is its kernel trick where it implicitly maps input data into a higher-dimensional space without explicitly computing the transformation by using a kernel function, such as radial basis function (RBF) or polynomial kernel. In our project, we conducted multiple experiments aimed at enhancing our model's performance and achieving higher F1-scores. Due to limitations in computational resources and the complexity of hyperparameter tuning, we opted to train our model on a subset of the data, specifically 20,000 randomly selected samples. Throughout our experiments, we explored different feature subsets, including all 25 features, the top 13 features (50%), and the top 7 features (25%).

Through experimentation, we investigated different train-test split ratios, specifically 80/20 and 70/30, and observed that the 80/20 split consistently outperforms the 70/30 split across all metrics.

All of our experiments were run for both 5-fold cross validation and 10-fold cross validation. However, we have found that 5-fold cross validation performs almost similarly compared to 10-fold cross validation. Therefore, we have only reported the results for 5-fold cross validation.

The hyperparameter tuning process involved the following steps:

1. Utilizing Grid Search CV, Randomized Search CV, and Bayesian Search CV to systematically search for optimal hyperparameters.
2. Exploring various kernels, such as "linear," "RBF," and "poly," to assess their impact on model performance. We have used default degree (degree =3) for ploy kernel.

3. Tuning both "C" values and "Gamma" values to identify the most suitable parameters for our model to avoid both overfitting and underfitting.

Table XVII shows the performance metrics by tuning the hyperparameters taking all 25 features into account. We have found that RBF kernel best suits our dataset. “C” value is the regularization parameter that controls the bias-variance trade-off of our model. In our case with Grid Search, “C” value is 1 which indicates “moderate regularization” resulting in a balance between achieving a smooth decision boundary and minimizing training errors while trying to maintain a reasonable margin between classes. Gamma parameter determines the influence of a single training sample. For Grid Search, gamma value 0.1 indicates a relatively low influence of a single training sample. For the Random and Bayes Search CV, higher values of C parameter are balanced by lower values of gamma parameter. Since our dataset is moderately imbalanced, we have mainly focused on the F1-scores. The best performance was achieved using Bayesian Search CV with an RBF kernel, having "C" set to 215.44 and gamma to 0.001, resulting in 74.40% accuracy and a 77.59% F1-score.

TABLE XVII. HYPERPARAMETER VALUES FOR GRIDSEARCHCV, RANDOMIZEDSEARCHCV, AND BAYESSEACHCV FOR 25 (100%) FEATURES

Metric	Grid	Random	Bayes
25 Features			
C	1	215.44346	215.443469
Kernel	RBF	RBF	RBF
Gamma	0.1	0.001	0.001
Accuracy	74.30%	74.28%	74.40%
Precision	75.77%	75.78%	75.90%
Recall	79.36%	79.27%	79.36%
F1	77.53%	77.49%	**77.59**%
MSE	0.2570	0.2573	0.2560
AUC-ROC	0.8011	0.8185	0.8202
Confusion Matrix	[1199 567] [461 1773]	[1200 566] [463 1771]	[1203 563] [461 1773]

Table XVIII showcases performance metrics after hyperparameter tuning with the top 13 features (50%). We replicated the aforementioned steps, yielding comparable results in performance metrics.

TABLE XVIII. HYPERPARAMETER VALUES FOR GRIDSEARCHCV, RANDOMIZEDSEARCHCV, AND BAYESSEACHCV FOR 13 (50%) FEATURES

Metric	Grid	Random	Bayes
13 Features			
C	1	2.1544346	2.15443469
Kernel	RBF	RBF	RBF
Gamma	0.1	0.1	0.1
Accuracy	74.68%	74.55%	74.65%
Precision	76.84%	76.69%	77.18%
Recall	78.25%	78.20%	77.53%
F1	77.53%	77.44%	77.36%
MSE	0.2533	0.2545	0.2535
AUC-ROC	0.7989	0.7959	0.8041
Confusion Matrix	[1239 527] [486 1748]	[1235 531] [487 1747]	[1254 512] [502 1732]

Finally, In Table XIX, we present the performance metrics obtained by fine-tuning hyperparameters while considering only the top 7 features (25%). The results showcase similarities with good accuracy and F1-scores, indicating consistent and satisfactory performance.

TABLE XIX. HYPERPARAMETER VALUES FOR GRIDSEARCHCV, RANDOMIZEDSEARCHCV, AND BAYESSEACHCV FOR 7 (25%) FEATURES

Metric	Grid	Random	Bayes
7 Features			
C	10	0.4641588	0.464158883
Kernel	RBF	RBF	RBF
Gamma	0.1	0.46415888	0.464158883
Accuracy	74.38%	74.40%	74.40%
Precision	76.48%	76.28%	76.26%
Recall	78.16%	78.60%	78.65%
F1	77.31%	77.43%	77.43%
MSE	0.2563	0.2560	0.2560
AUC-ROC	0.7885	0.7837	0.7814
Confusion Matrix	[1229 537] [488 1746]	[1220 546] [478 1756]	[1219 547] [477 1757]

The Area under the Receiver Operating Characteristic ROC curve for SVM indicating its values for three different hyperparameter tuning methods is illustrated in Fig. 41. The highest AUC-ROC value, 0.8202, was achieved when considering all 25 features indicating a "good" classification ability. AUC-ROC is a measure of how well a model distinguishes between classes, specifically the trade-off between recall or sensitivity (true positive rate) and fall-out (false positive rate).

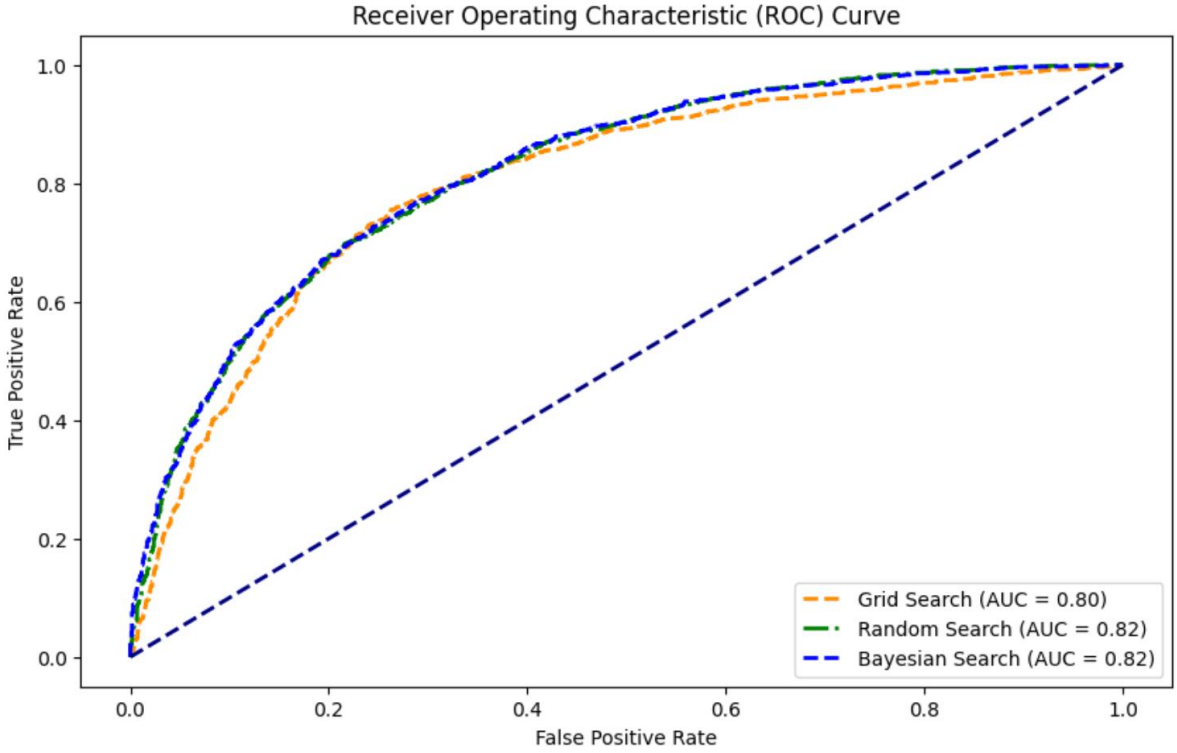


Fig. 41. Receiver Operating Characteristic (ROC) curve for SVM indicating its values for three different hyperparameter tuning methods

Overall, the performance metrics can be summarized as follows:

1. Accuracy: 74.40%
2. Precision: 75.90%
3. Recall: 79.36%
4. F1: 77.59%
5. MSE: 0.2560
6. AUC-ROC: 0.8202
7. Confusion Matrix
   [1203 563]
   [461 1773]

6. Multi-layer Perceptron (MLP)

As one of the most important machine learning models, MLP will be studied for this project, there are many parameters to tune and yet this training and tuning process is a time consuming. A Large amount of computing powers are needed the model update process, in order to work on the simulation and hyperparameter tuning process, some tasks for this more will sample a part of the whole dataset to test its performance. For the setup which will not be used for hyperparameter tuning, the default parameters are:

- one hidden layer with 10 neurons
- relu activation function
- adam solver, alpha equals 0.0001
- batch size is 64
- learning rate is constant

Proportion of dataset is first to be considered in order to have a better understanding on performance of the model. The following figure shows the performance of the model regarding the percentage of dataset being used. It can be shown that a larger dataset improves the performance, yet more time and computing resources are needed. Fig. 42 illustrates Fig. 42. 5-Fold CV Weighted F1 Score over Samples using MLP.

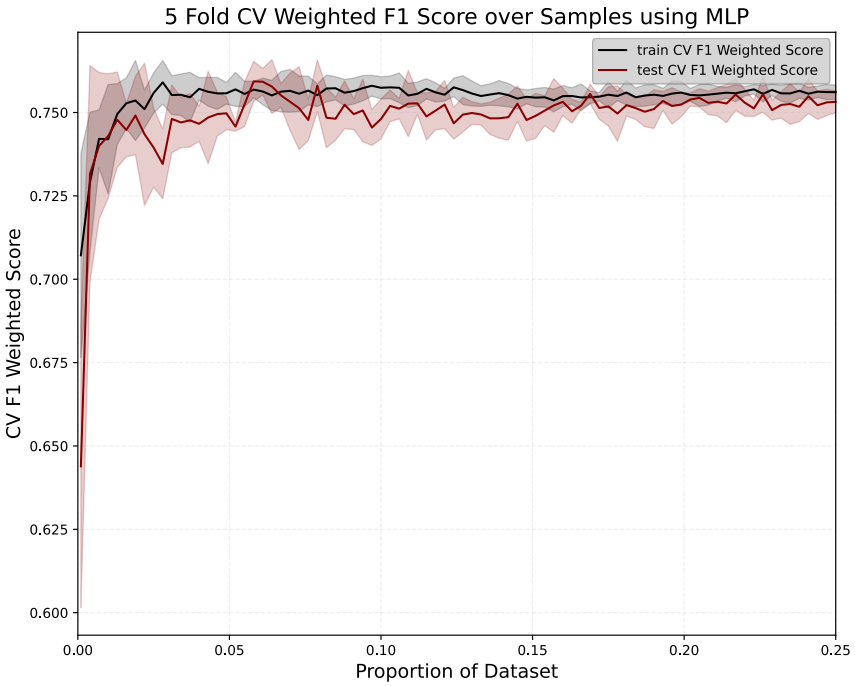


Fig. 42. 5-Fold CV Weighted F1-Score over Samples using MLP

In the beginning, small size of dataset would dramatically impact the model performance, later on more dataset elevates the F1 weighted score, both training and testing cross validation scores stabilizes while training score outperformances testing one with no indications of overfitting, although alpha is used as the

default value and it is not tuned to penalize the overfitting problem. This simulation stops at 25% percent as further simulation would spend a lot of time.
One thing we want to examine is how the parameter alpha in MLP is able to avoid the problem of overfitting. Larger value results in larger penalization to avoid overfitting problem. Fig. 43 shows the performance of the machine learning model with respect to variable alpha value.

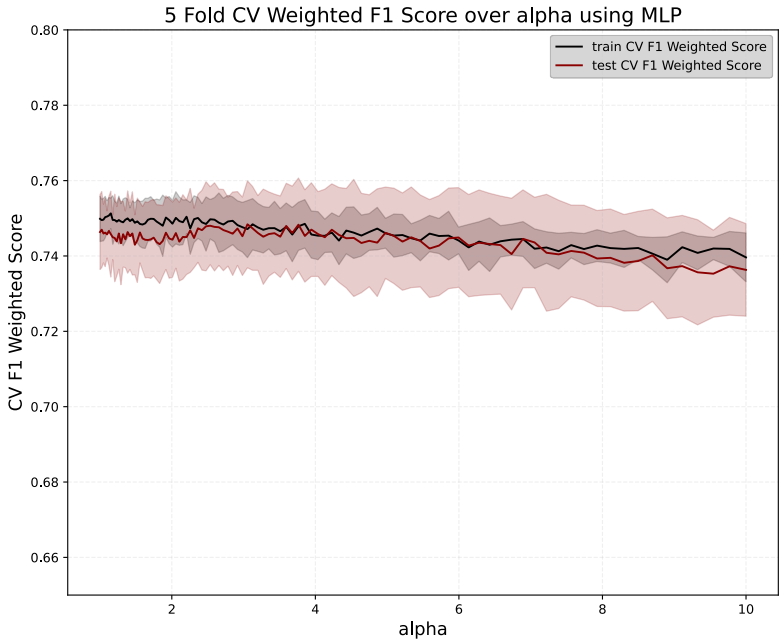


Fig. 43. 5-Fold CV Weighted F1-Score over alpha using MLP

As MLP is known for its network structure, this series of simulations are focusing on tuning the number of neurons, as we leant in class, MLP by definition has 1 hidden layer, thus 1 hidden layer is focused first and later another layer is variations are also being considered.
For the single hidden layer condition shown in Fig. 44, numbers of neurons from 1 to 100 are being considered; there's a difference between training and testing scores. Generally speaking, making the network more complex by adding more neurons in the hidden layer may not increase the performance.

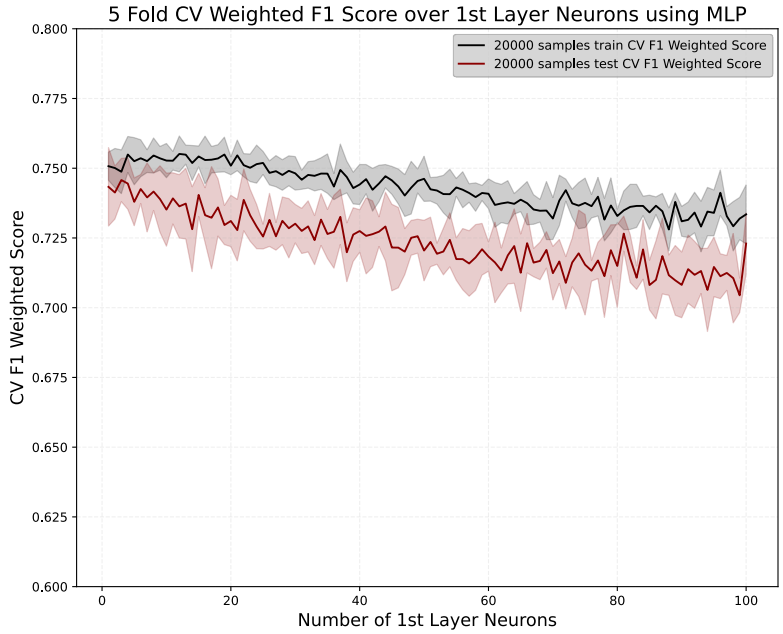


Fig. 44. 5-Fold CV Weighted F1-Score over 1st Layer Neurons using MLP

If a second layer with 11 neurons is added and changing the number of neurons in the first layer from 1 to 100 shows that there's performance degradation, Fig. 45 provides such setup.

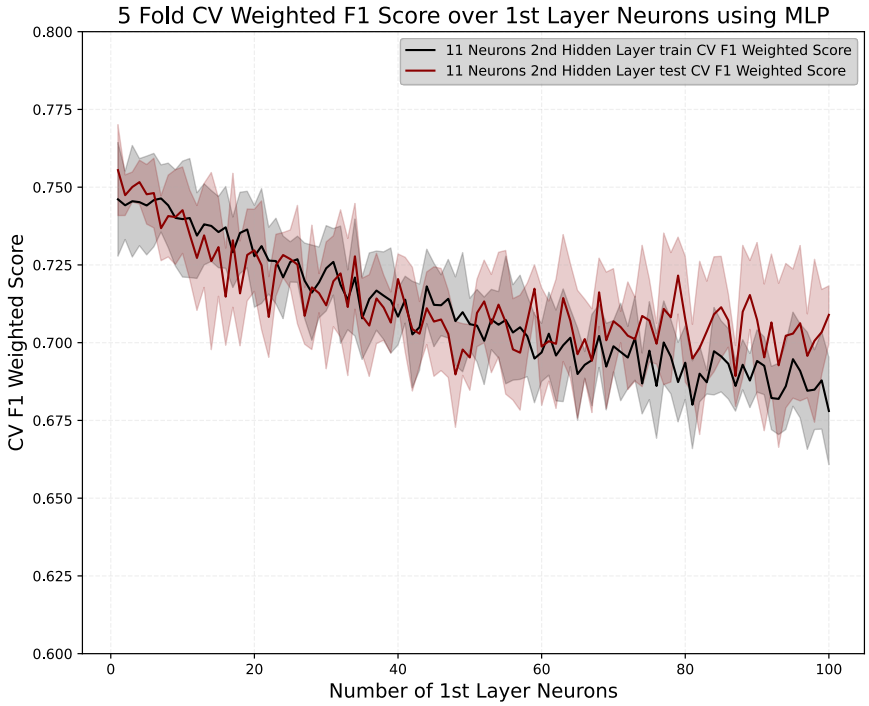


Fig. 45. 5-Fold CV Weighted F1 Score over 1st Layer Neurons using MLP with 11 2nd Layer Neurons

Trying to varying the neurons of the 2nd layer may not improve the perforamance as here's the comparison between 6 neurons 2nd hidden layer and 11 neurons 2nd hidden layer as shown in Fig. 46.

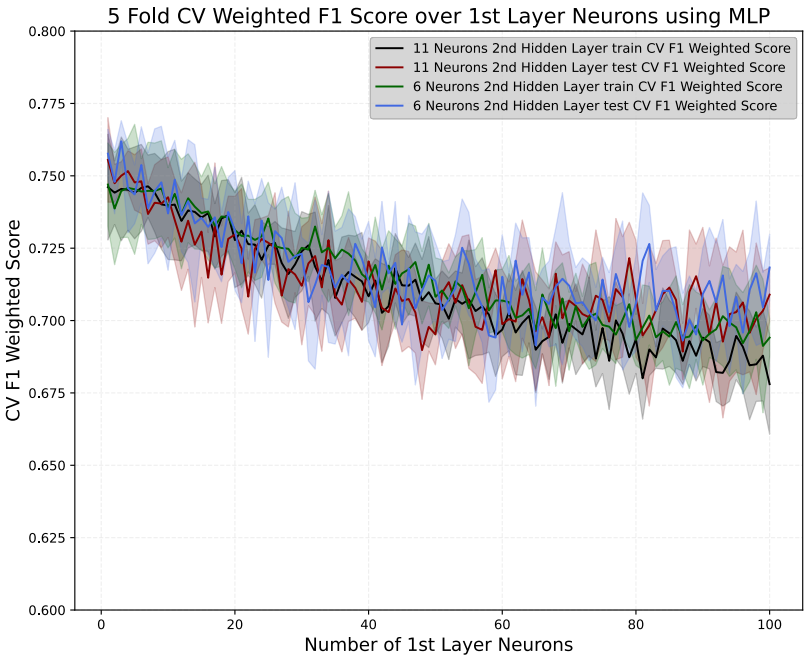


Fig. 46. 5-Fold CV Weighted F1 Score over 1st Layer Neurons using MLP with 6 and 11 2nd Layer Neurons

It could be judged that mlp based neural network is very popular in academic and industry, adding complexity to it may not be the only option to improve the performance, even worse, more complexity results in the downgraded performance as more computing resoures are used.

Although we anticipate MLP would take a lot of time for hyperparameter tuning, the simulation process takes more than is anticipated. Thus, a portion of dataset will be used for tuning the model, In this part, 10,000 values are sampled from the dataset. In addition, for hyperparameter tuning algorithm, Randomized Search and Bayes Search are used since Grid Search takes a long time to iterate through the whole lists of parameters, the following list contains the search space for the parameters.

- 'hidden_layer_sizes': [(1,),(2,),(5,),(10,),(15,),(20,),(30,)]
- 'activation': ['tanh', 'relu','logistic']
- 'solver': ['sgd', 'adam']
- 'alpha': [0.0001, 0.001, 0.01]
- 'batch_size': [32, 64, 128]

The most optimized parameters are shown as follows in Table XX:

TABLE XX. PERFORMANCE METRIC FOR MULTY-LAYER PERCEPTRON

CV	**CV=5**	**CV=5**	**CV=5**
Top Featues	**25 Features**	**13 Features**	**7 Features**
Algorithm	Random Search	Random Search	Bayes Search
Accuracy	76.75%	74.45%	74.4%
Precision	76.36%	74.06%	73.97%
Recall	76.18%	73.60%	73.67%
F1-Score	76.26%	73.76%	78.79%
MSE	0.2325	0.2555	0.256
AUC	0.8467	0.8216	0.813
Confusion Matrix	[624 244] [221 911]	[583 285] [226 906]	[592 276] [236 896]

7. Random Forest

In the implementation of the random forest algorithm, the following steps were taken:

- Preliminary Training and Performance Evaluation
- Hyperparameter Tuning Using GridSearchCV, RandomizedSearchCV, and BayesianCV
- Performance Evaluation with GridSearch, RandomizedSearch, and Bayesian Tuned

a) Preliminary Training and Performance Evaluation

A preliminary classification tree was built and trained with the performance evaluated to report on how the model performed before optimizing the hyperparameters.

The performance and confusion matrix are represented in Table XXI and Fig. 47 respectively.

TABLE XXI. PERFORMANCE EVALUATION OF PPRELIMINARY RANDOM FOREST MODEL

	Accuracy	**Precision**	**Recall**	**F1 Score**	**Weighted F1 Score**	**Area Under ROC curve**	**Mean Squared Error**
Preliminary RF model	0.7360	0.7577	0.7753	0.7664	0.7355	0.8163	0.32640

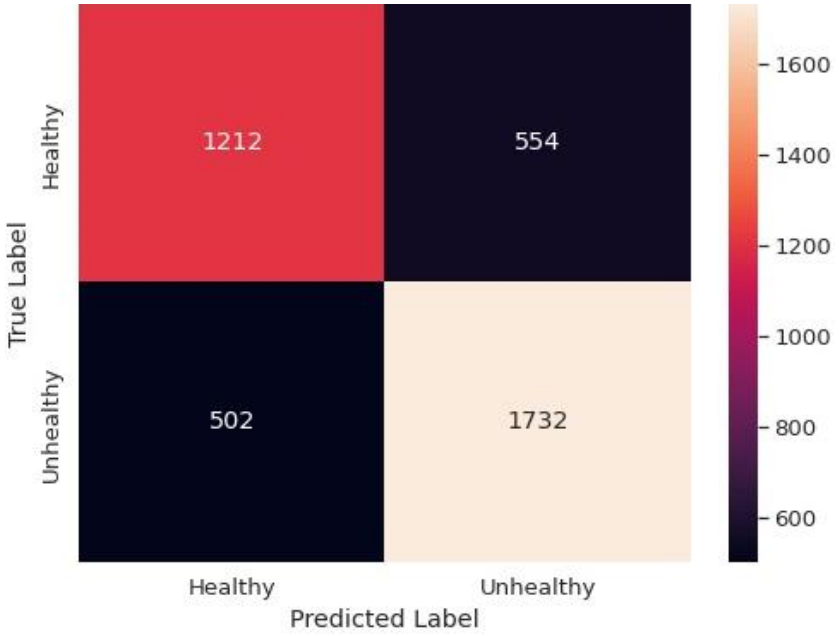


Fig. 47. Graph Showing Preliminary Performance of Random Forest Model

b) Hyperparameter Tuning Using GridSearchCV, RandomizedSearchCV, and BayesianCV

The criterion for maximum depth, number of estimators, and minimum samples per leaf were optimized using grid search, randomized search, and Bayesian search by creating an instance of a search object (with a decision tree classifier and the hyperparameters) and conducting a search over the parameter space given for the individual hyperparameters.

The options given for the various hyperparameters are specified below:

Maximum Depth: [2,3,5,10,20]

Minimum Samples Leaf: [5,10,20,50,100,200]

Number of estimators: [10,25,30,50,100,200]

After doing the search, the ideal parameters for the various search techniques using 5-fold cross validation are summarized in Table XXII and the ideal parameters for the various search techniques using 5-fold cross validation are summarized in Table XXIII.

TABLE XXII. IDEAL HYPERPARAMETER VALUES FOR GRIDSEARCHCV, RANDOMIZEDSEARCHCV, AND BAYESSEARCHCV WIT 5-FOLD CROSS VALIDATION

	Max. Depth	**Min. Samples Leaf**	**Number of estimators**
GridSearchCV	10	20	50
RandomizedSearchCV	20	10	200
BayesSearchCV	10	20	50

TABLE XXIII. IDEAL HYPERPARAMETER VALUES FOR GRIDSEARCHCV, RANDOMIZEDSEARCHCV, AND BAYESSEARCHCV WIT 5-FOLD CROSS VALIDATION

	Max. Depth	**Min. Samples Leaf**	**Number of estimators**
GridSearchCV	10	10	200
RandomizedSearchCV	10	20	100
BayesSearchCV	10	10	200

c) Performance Evaluation with GridSearchCV, RandomizedSearchCV, and BayesianCV Parameters

Random Forest classifiers were built with the considered parameters for hyperparameter tuning set to the selected ideal values derived from each of the parameter space search techniques. The results for 5-fold cross validation are summarized in Table XXIV and the individual confusion matrix for the classifier trees built with the ideal parameters from GridSearchCV, RandomizedSearchCV, and BayesSearchCV are shown in Fig. 48, 49, and 50 respectively.

The results for 10-fold cross validation are summarized in Table XXV and the individual confusion matrix for the classifiers are shown in Fig. 51, 52, and 53 respectively.

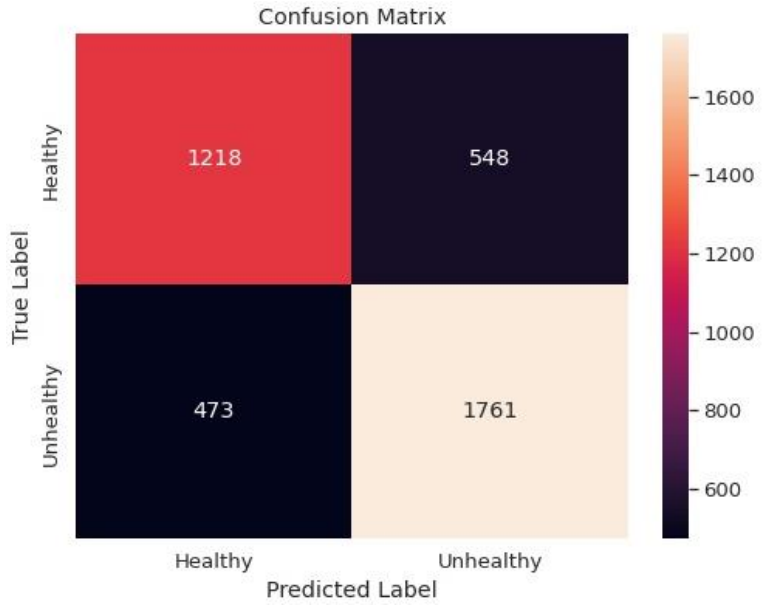


Fig. 48. Confusion matrix of Random Forest Classifier with hyperparameters from GridSearchCV with 5-fold cross validation

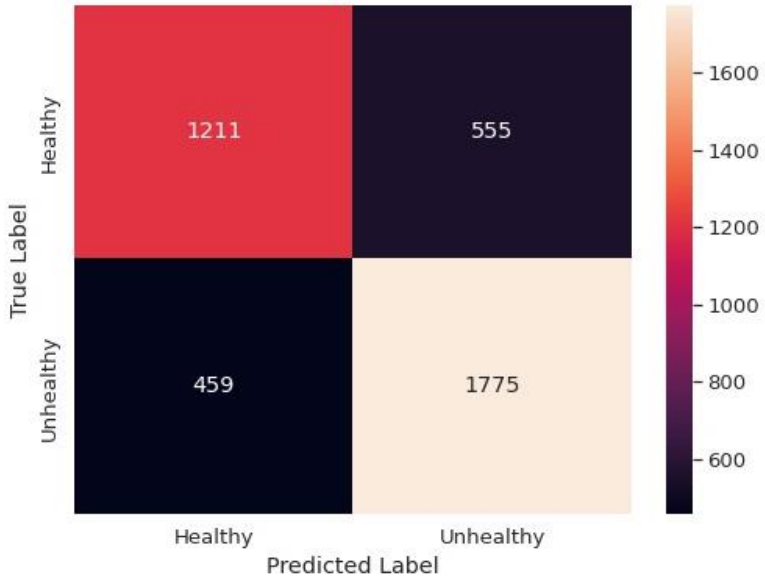


Fig. 49. Confusion matrix of Random Forest Classifier with hyperparameters from RandomizedSearchCV with 5-fold cross validation

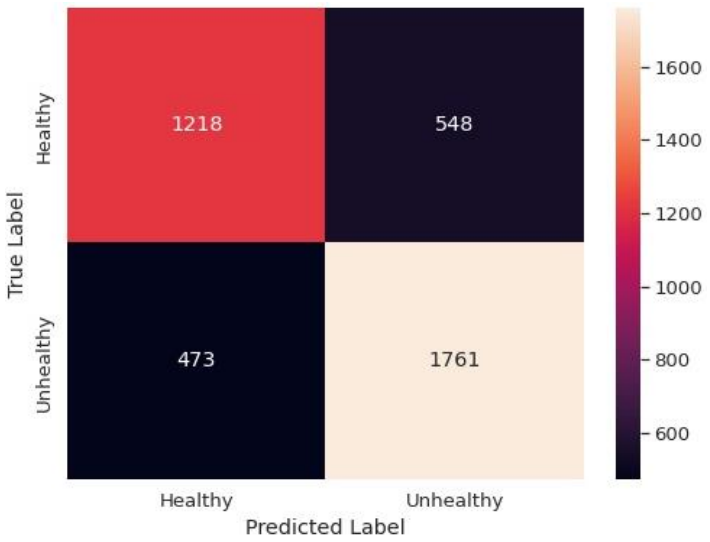


Fig. 50. Confusion matrix of Random Forest Classifier with hyperparameters from BayesSearchCV with 5-fold cross validation

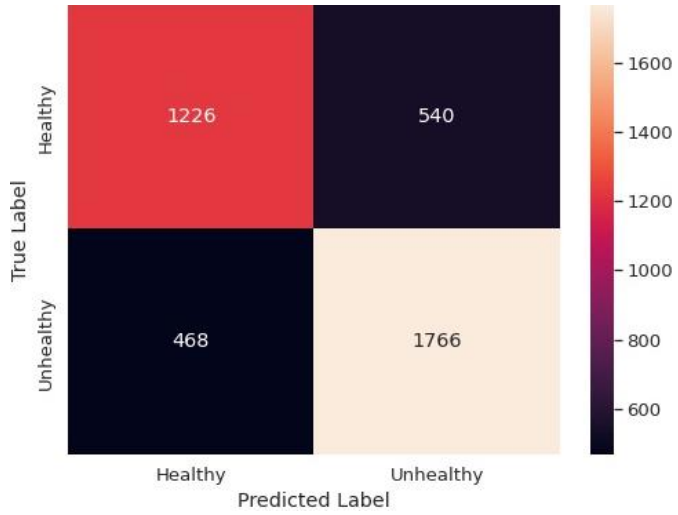


Fig. 51. Confusion matrix of Random Forest Classifier with hyperparameters from GridSearchCV with 10-fold cross validation

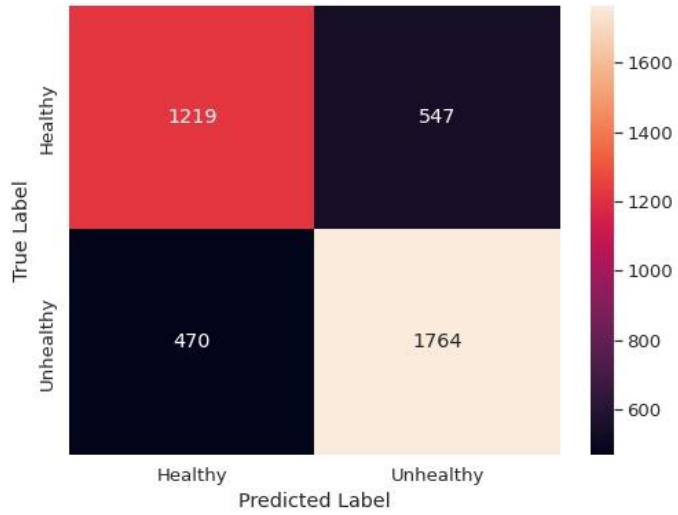


Fig. 52. Confusion matrix of Random Forest Classifier with hyperparameters from RandomizedSearchCV with 5-fold cross validation

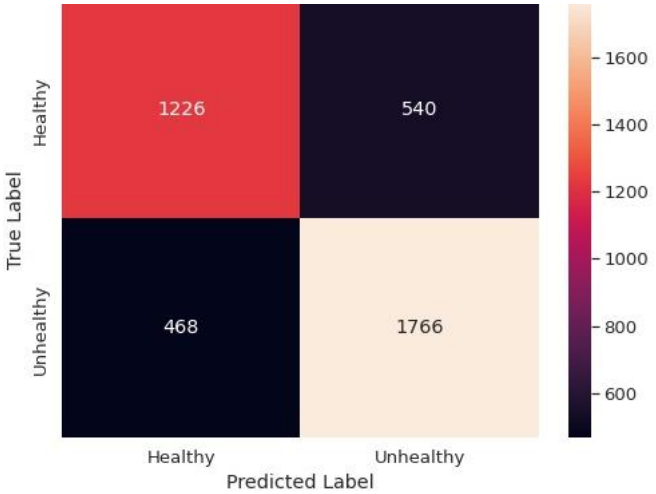


Fig. 53. Confusion matrix of Random Forest Classifier with hyperparameters from BayesSearchCV with 5-fold cross validation

TABLE XXIV. PERFORMANCE EVALUATION OF RANDOM FOREST CLASSIFIER WITH GRIDSEARCHCV, RANDOMIZEDSEARCHCV, AND BAYESSEARCHCV WIT 5-FOLD CROSS VALIDATION

	OOB	Accuracy	Precision	Recall	F1	Weighted F1	AUR	MSE
GridSearchCV	0.7499	0.74	0.76	0.79	0.78	0.74	0.82	0.26
RandomizedSearchCV	0.7516	0.75	0.76	0.79	0.78	0.75	0.82	0.25
BayesSearchCV	0.7499	0.74	0.76	0.79	0.78	0.74	0.82	0.26

TABLE XXV. PERFORMANCE EVALUATION OF RANDOM FOREST CLASSIFIER WITH GRIDSEARCHCV, RANDOMIZEDSEARCHCV, AND BAYESSEARCHCV WIT 10-FOLD CROSS VALIDATION

	OOB	Accuracy	Precision	Recall	F1	Weighted F1	AUR	MSE
GridSearchCV	0.7491	0.75	0.77	0.79	0.78	0.75	0.82	0.25
RandomizedSearchCV	0.7505	0.75	0.76	0.79	0.78	0.75	0.82	0.25
BayesSearchCV	0.7491	0.75	0.77	0.79	0.78	0.75	0.82	0.25

Overall, the performance metrics can be summarized as follows:

1. Accuracy: 75%
2. Precision: 77%
3. Recall: 79%
4. F1: 78%
5. MSE: 0.25
6. AUC-ROC: 0.82
7. Confusion Matrix
   [1226 540]
   [468 1766]

8. Gradient Boosting

In this part, both gradient boosting and XGBoost algorithms are taken into consideration as they do share similarities. Several parameters are considered and varied to measure the weighted F1 score, it is noted that when varying one parameter, other parameters are not the optimized values, thus F1 score may not be

really high, but those figures would show how F1 score changes with those variables. Nevertheless, the hyperparameter tuning results will be shown later.
When varying the number of estimators for both Gradient Boosting and XGBoost on the number of estimators as shown in Fig. 54, the performance improves by the number of estimators or trees. There's also no indications there would be a overfitting phenomenon occur. Training and testing F1 weighted scores for both of the algorithms overlap based on the figure.

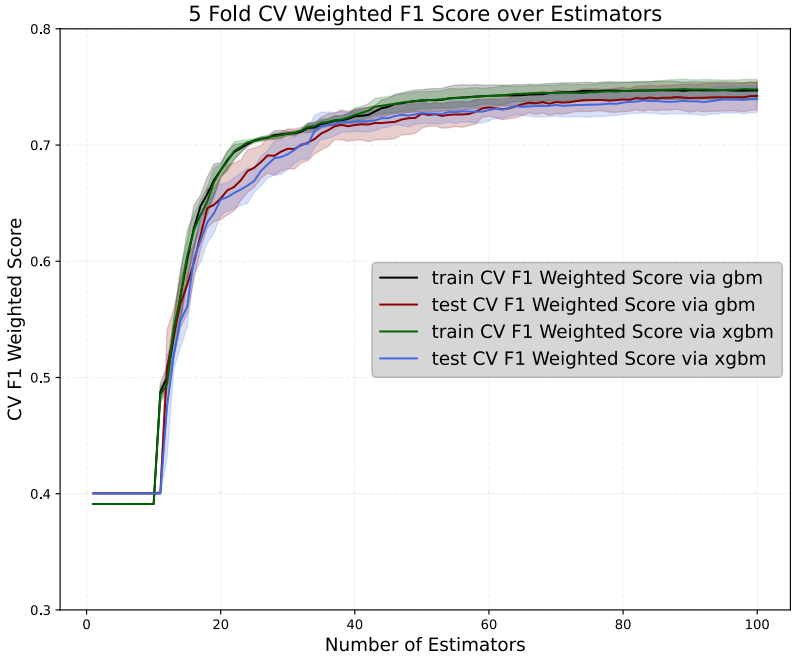


Fig. 54. 5-Fold Weighted F1 Score over Estimators

As this project has a large dataset, question on how well a model works with various amount of data, in the Fig. 55, maximum of 10% of data is used on Gradient Boosting and XGBoost, both have similar training and testing weighted f1 scores with similar variance from 5-fold cross validation. By increasing the sampling size, the performance can be improved but there's a threshold that increasing sampling dataset would not affect the results.

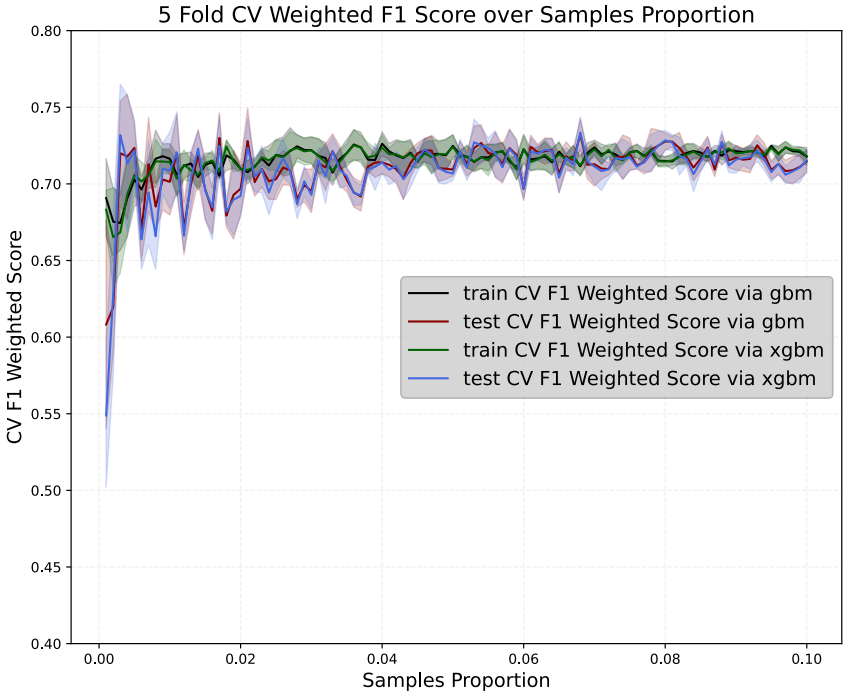


Fig. 55. 5-Fold CV Weighted F1 Score over Samples Proportion

The tree depth is one useful parameter in not only Gradient Boosting and XGBoost, but it can also be used in Random Forest which is also included in this study. Fig. 56 shows the 5-fold training and testing scores with different tree depth. From the figure, both Gradient Boosting and XGBoost have same optimized tree depth value, when such parameter goes beyond the optimized value, the scores for training and testing diminish.

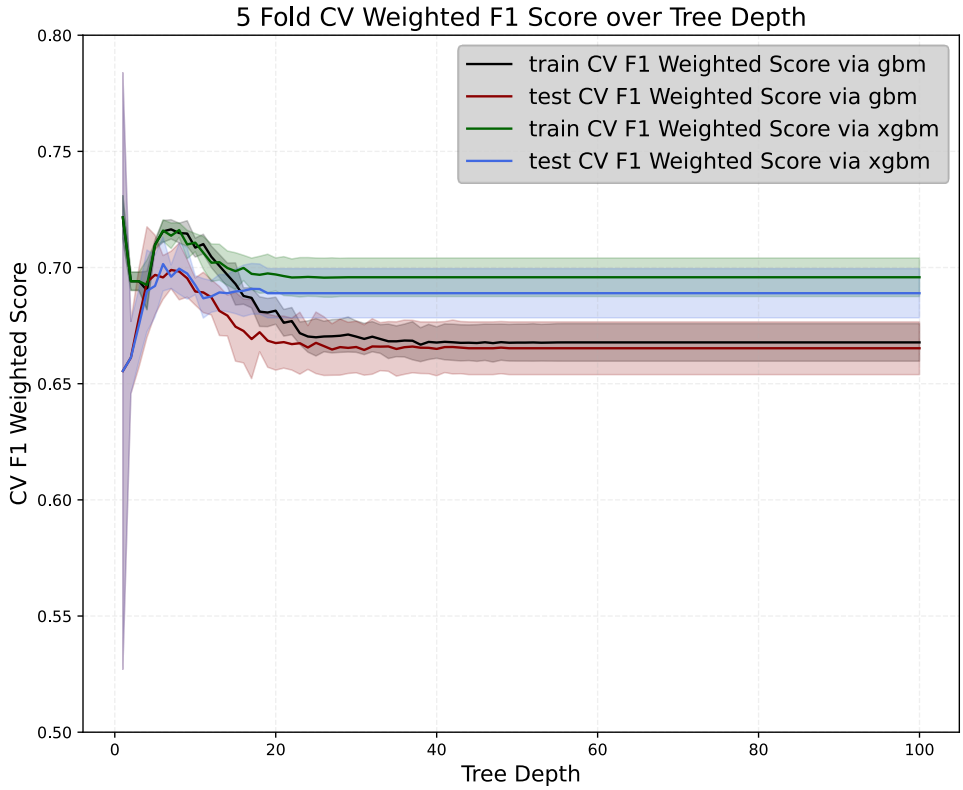


Fig. 56. 5-Fold CV Weighted F1 Score over Tree Depth

Similar to multi-layer perceptron, Gradient Boosting and XGBoost also require considerable amount of computing power in order to train the model, thus Randomized Search and Bayes Search are used because of the efficiency. Hyperparameter tuning for both Gradient Boosting and XGBoost is based on the following search space.

- 'n_estimators': [20, 50, 100, 200],
- 'learning_rate': [0.001, 0.01, 0.1, 0.2],
- 'max_depth': [3, 5, 7, 9, 15, 20, 30],
- 'min_samples_split': [2, 4, 6, 8, 10, 15, 20, 30],
- 'subsample': [0.5, 0.7, 1]

The following tables, namely Table XXVI and Table XXVII show the tuned parameters for Gradient Boosting and XGBoost, as with MLP and different from Naïve Bayes, both Gradient Boosting and XGBoost improve the performance by including more features in the training and testing process. XGBoost has better performance than Gradient Boosting when all 25 features are included.

TABLE XXVI. PERFORMANCE METRIC FOR GRADIENT BOOSTING

CV	**CV=5**	**CV=5**	**CV=5**
Top Features	**25 Features**	**13 Features**	**7 Features**
Algorithm	Bayes Search	Random Search	Bayes Search
Accuracy	75.58%	74.24%	73.93%

Precision	75.41%	74.04%	73.70%
Recall	75.10%	73.71%	73.45%
F1-Score	75.21%	73.82%	78.54%
MSE	0.2441	0.2576	0.2608
AUC	0.8310	0.8066	0.7998
Confusion Matrix	[6330 2684] [2199 8787]	[6168 2846] [2306 8680]	[6190 2824] [2391 8595]

TABLE XXVII. PERFORMANCE METRIC FOR XGBOOST

CV	CV=5	CV=5	CV=5
Top Features	25 Features	13 Features	7 Features
Algorithm	Random Search	Bayes Search	Random Search
Accuracy	76.85%	74.2%	73.95%
Precision	76.51%	73.76%	73.71%
Recall	76.15%	73.47%	72.80%
F1-Score	76.29%	73.59%	73.04%
MSE	0.2315	0.258	0.2605
AUC	0.8384	0.8148	0.8113
Confusion Matrix	[615 253] [210 922]	[590 278] [238 894]	[556 312] [209 923]

9. Majority Voting Classifier

For the Majority Voting Classifier our idea was to combine the predictions of the best individual classifiers to make a final prediction to improve overall predictive performance and robustness. We have used three individual classifiers, namely, Support Vector Machine, Multi-Layer Perceptron, Random Forest Classifier for creating Majority Voting Classifier. We've used 10000 randomly selected samples from our dataset and split it into 80/20 train-test ratio. We have also executed 5-fold and 10-fold cross validation to our model and found almost similar results. Two different voting types, namely, soft and hard, were examined in pursuit of achieving the best F1-score. In 'hard' voting type each classifier votes for a class, and the class with the most votes is chosen as the final prediction. On the other hand, for the 'soft' voting type instead of a binary vote, each classifier assigns a probability to each class. The final prediction is based on the class with the highest cumulative probability. Table XXVIII shows the performance metrics for the Majority Voting Classifier using 10-fold cross validation with "soft" voting.

TABLE XXVIII. PERFORMANCE METRIC FOR THE MAJORITY VOTING CLASSIFIER WITH "SOFT" VOTING

voting = 'soft'	CV=10	CV=10	CV=10
	25 Features	13 Features	7 Features
Mean CV Accuracy	74.57%	76.95%	75.95%
Accuracy	75.85%	76.95%	75.95%
Precision	74.94%	80.20%	78.58%
Recall	86.13%	78.71%	79.06%
F1-Score	**80.15%**	79.45%	78.82%
MSE	0.2415	0.2305	0.2405
AUC	0.8427	0.8343	0.8267
Confusion Matrix	[542 326] [157 975]	[648 220] [241 891]	[624 244] [237 895]

As from Table XXVIII, we can clearly see that the F1-score for 25 Features reached our expected 80% score. We have achieved this by carefully tuning each of our algorithm so that individually they can perform their level best, and them from the top-performing individual classifier, we have created our own ensemble classifier, called the Majority Voting Classifier to further extend their performance and finally achieved our goal of getting F1-score of 80% accuracy score.

In Table XXIX, we have also reported the results when we use "hard" as our voting classifier. The results showed slightly inferior performance as compared to the 'soft' voting type.

TABLE XXIX. PERFORMANCE METRIC FOR THE MAJORITY VOTING CLASSIFIER WITH "HARD" VOTING

voting = 'hard'	**CV=10**	**CV=10**	**CV=10**
	25 Features	**13 Features**	**7 Features**
Mean CV Accuracy	74.57%	76.95%	75.95%
Accuracy	74.90%	76.25%	75.50%
Precision	73.44%	78.10%	78.11%
Recall	87.19%	80.65%	78.80%
F1-Score	79.73%	79.36%	78.45%
MSE	0.2510	0.2375	0.2450
Confusion Matrix	[511 357] [145 987]	[612 256] [219 913]	[618 250] [240 892]

Overall, the performance metrics can be summarized as follows:

1. Accuracy: 75.85%
2. Precision: 74.94%
3. Recall: 86.13%
4. F1: 80.15%
5. MSE: 0.2415
6. AUC-ROC: 0.8427
7. Confusion Matrix:
   [542 326]
   [157 975]

VI. RESULTS AND DISCUSSION

A. Results

In this section we will show the results of our models and discuss their performance. Fig. 57 shows the accuracy plot for our classifiers. We have found the maximum accuracy with Extreme Gradient Boosting classifier which is around 76.85%. The closest to it is the Multi-layer Perceptron Classifier with an accuracy of 76.75%. The worst-performing classifier in this list is K-Nearest Neighbors with 70.15% accuracy. It is expected since our dataset is not balanced leading to biased prediction for KNN. Along with KNN, Naïve Bayes Classifier also underperforms compared to its other counterpart having an accuracy of 71.77%

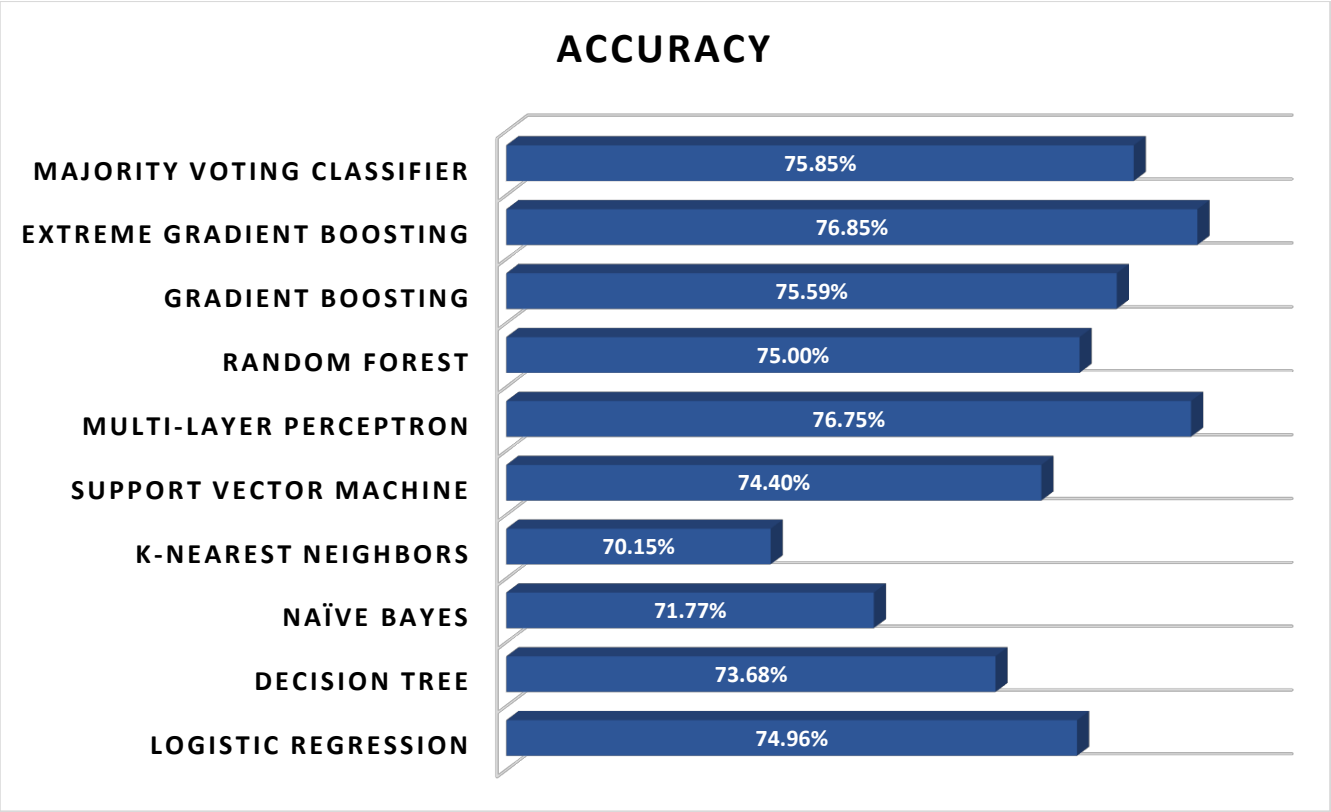


Fig. 57. Accuracy Scores for Each Individual Classifier

Fig. 58 illustrates the 'Precision' performance metric for each of the classifiers.

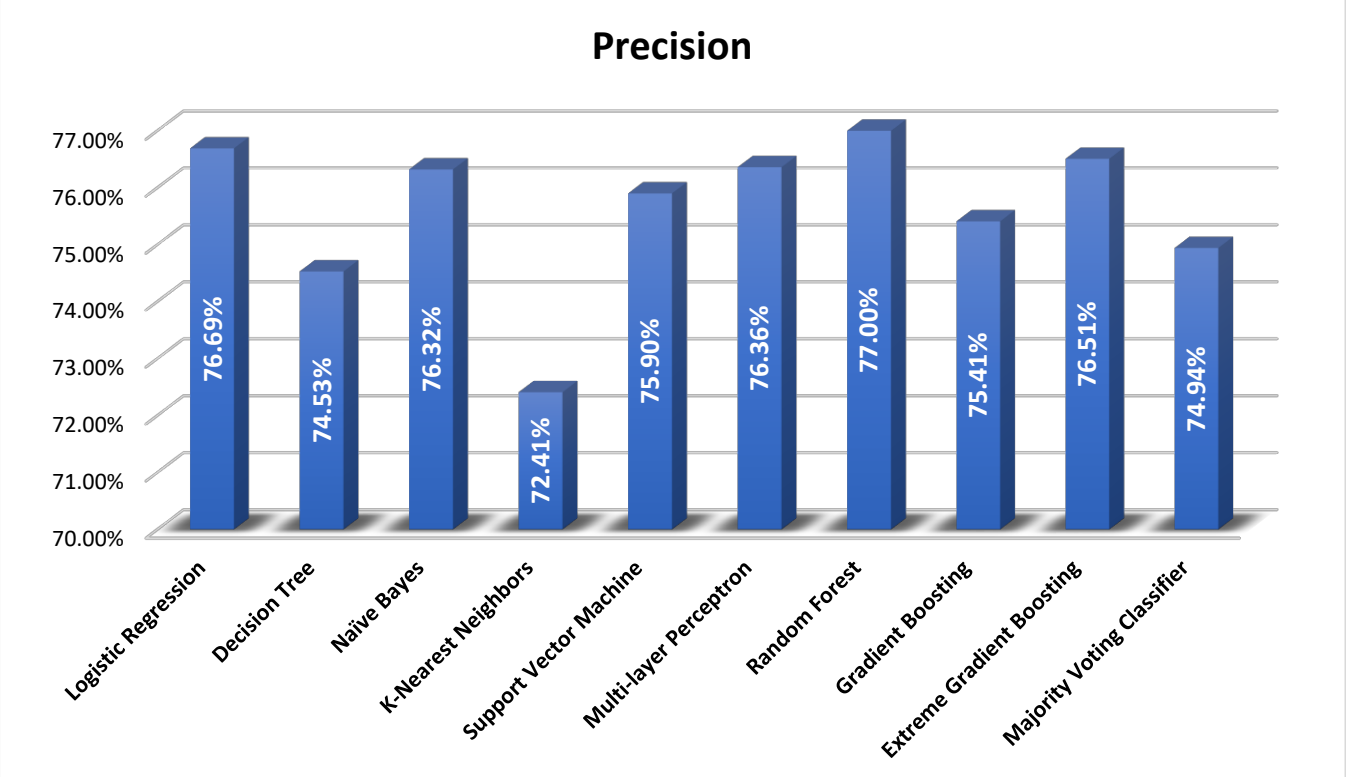


Fig. 58. Precision Scores for Each Individual Classifier

The top-performing model in terms of precision is the Random Forest Classifier having a precision score of 77%. Most of our algorithms have good precision scores while KNN underperforms in this metric with a score of 72.41%.

Fig. 59 depicts the "Recall" performance metric for our classifiers.

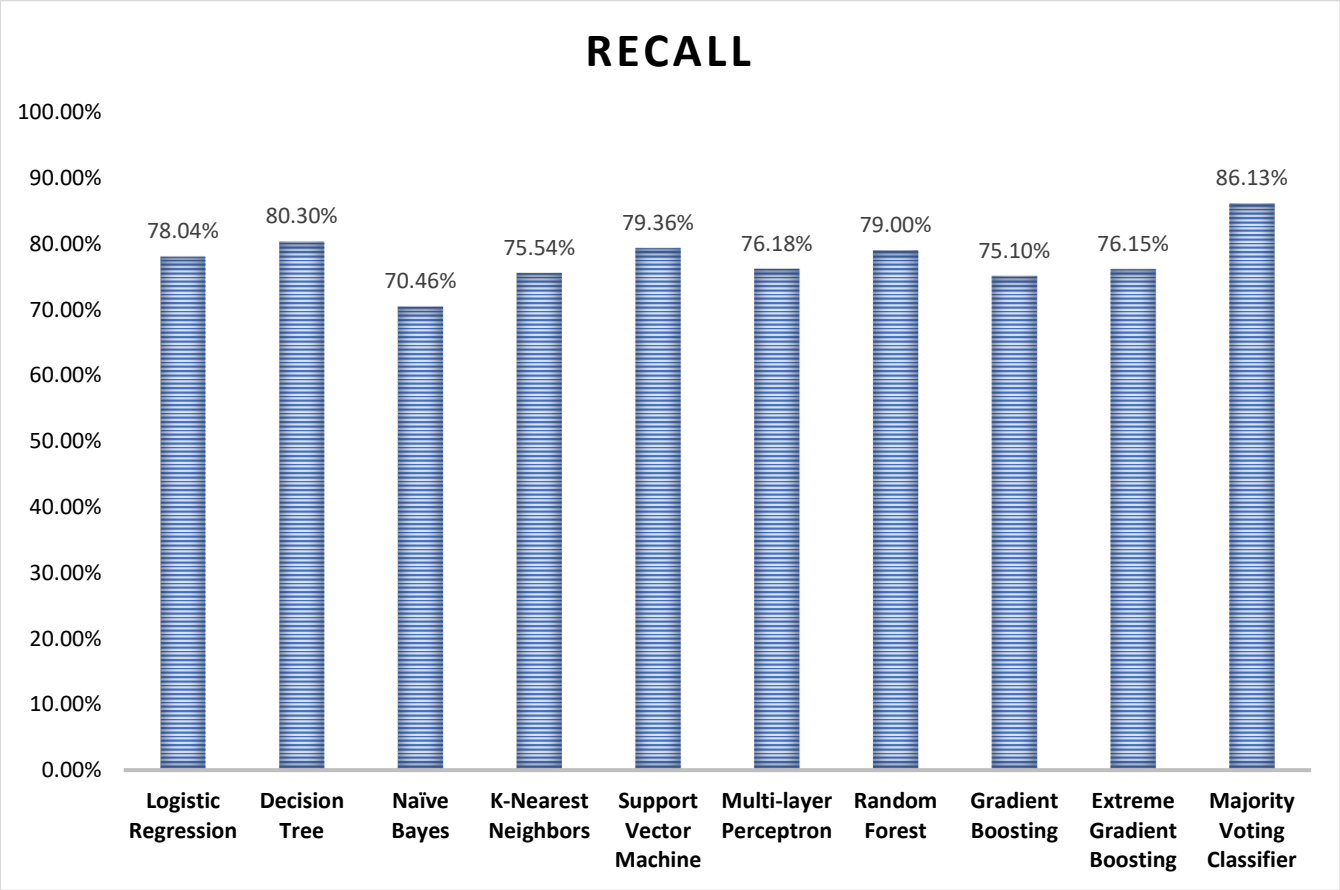


Fig. 59. Recall Scores for Each Individual Classifier

In terms of Recall score the Majority Voting Classifier (MVC) outperforms other classifiers by a large margin of around 6%. The Recall score for MVC is 86.13%, In contrast, the Naïve Bayes classifier falls short, demonstrating a less satisfactory performance with a Recall score just above 70%.

Since our dataset is imbalanced, F1-score becomes the most important performance metric for our dataset. Fig. 60 illustrates the F1-scores for each of our classifiers.

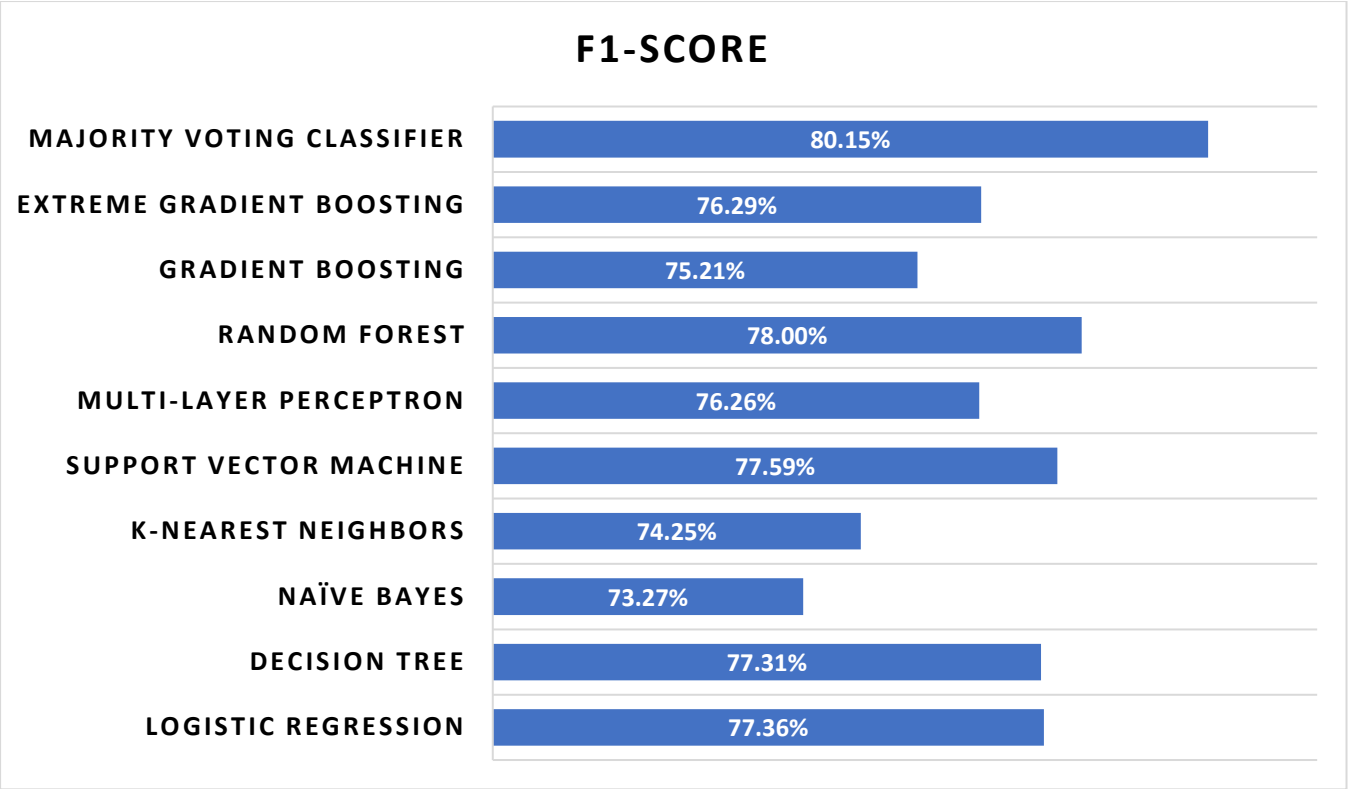


Fig. 60. F1-Scores for Each Individual Classifier

Since our primary aim was to achieve F1-score around 80%, we tried to tune each of the induvial classifier through hyperparameter tuning and used top-three performing individual classifiers to create an ensemble classifier, namely Majority Voting Classifier (MVC). From Fig. 60, we can see that the MVC outshines other classifiers with a pretty good F1-score of 80.15% that culminated in our successful achievement of the primary goal. However, as expected, Naïve Bayes classifier underperforms again with an F1-score of 73.27% only.

Moving forward, Fig. 61 demonstrates the Mean Square Error (MSE) performance metrics.

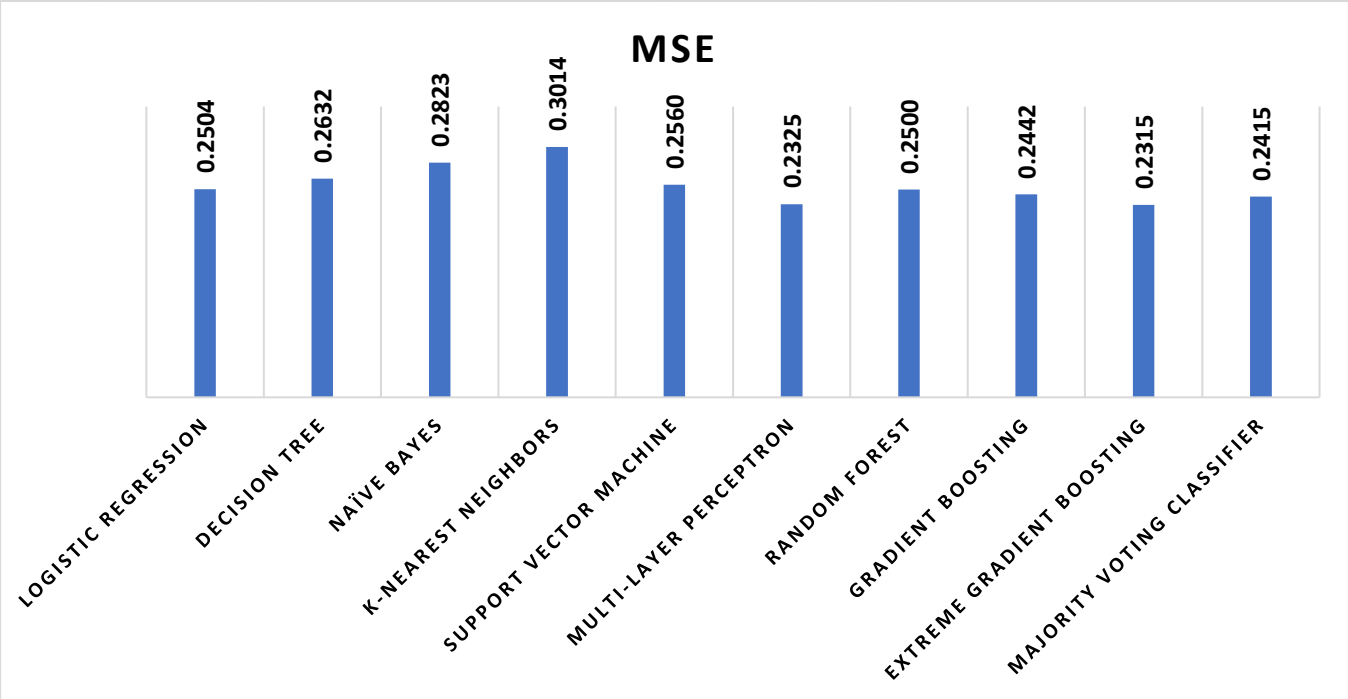


Fig. 61. MSE Scores for Each Individual Classifier

Given that a lower Mean Squared Error (MSE) is preferable, from Fig. 61 we can conclude that KNN exhibits the poorest performance with an MSE score of 0.3014 while Extreme Gradient Boosting performs the best with an MSE score of 0.2315.

Finally, Area under the Receiver Operating Characteristic Curve (AUC-ROC) performance metrics are shown in Fig. 62.

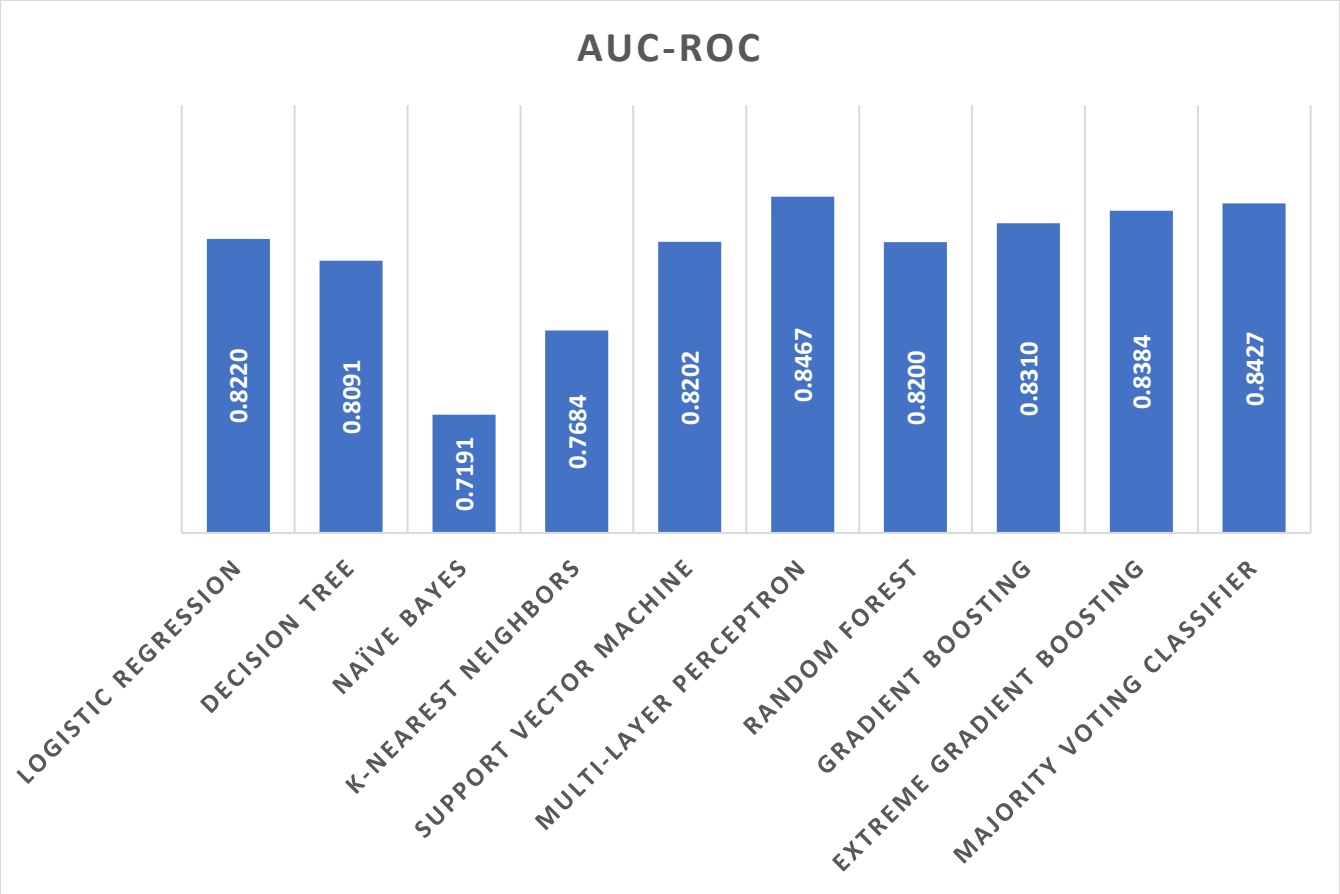


Fig. 62. AUC-ROC Scores for Each Individual Classifier

AUC-ROC scores for all classifiers are more than 0.8 except for KNN and Naïve Bayes classifiers with less than 0.77 AUC-ROC scores.

B. Discussions

In our proposal report, we expected to achieve at least 70% to 80% accuracy and F1-scores as well as 0.85 AUC-ROC score. After doing a significant number of experiments in each of the individual classifiers, we can say that we have successfully achieved these scores including all three of our specific goals.

1. We have successfully benchmarked nine individual classifiers for lifestyle prediction using seven performance metrics.
2. We have also achieved our major goal of 80% F1-score through the Majority Voting Ensemble Classifier using the top three best-performing individual classifiers.
3. By employing various feature engineering techniques, we've enhanced the efficiency and accuracy of our classifiers.

VII. TIMELINES

As shown in the proposal, many time consuming computationally intensive experiments need to be done in time, thus designing a schedule for fulfilling the aims is vital. This section describes the timeline and the tasks assigned for each member of the team for collaboration to fulfill the aims for the lifestyle classification. Table XXX shows the timeline and the name of the person who will be working on the specific tasks to complete the project.

TABLE XXX. TIMELINE FOR THE FINAL REPORT

Dataset Preprocessing		
Task List	**Name of the Person**	**Timeline**
Data Cleaning and Exploratory Data Analysis, and Outlier Detection	Shirazush Salekin Chowdhury	October 09 to October 13 (1 week)
Feature Engineering, and Feature Scaling	Alberta Dadeboe	October 16 to October 20 (1 week)
Feature Selection	Xiaomeng Wang	October 16 to October 20 (1 week)
Model Building, Validation and Performance Evaluation		
Logistic Regression	Shirazush Salekin Chowdhury	October 23 to November 03 (2 weeks)
Decision Tree	Alberta Dadeboe	October 23 to November 03 (2 weeks)
Naïve Bayes	Xiaomeng Wang	October 23 to November 03 (2 weeks)
KNN	Alberta Dadeboe	November 06 to November 10 (1 week)
Support Vector Machine	Shirazush Salekin Chowdhury	November 06 to November 10 (1 week)
Multi-Layer Perceptron	Xiaomeng Wang	November 06 to November 10 (1 week)
Ensemble Method		
Random Forest	Alberta Dadeboe	November 13 to November 17 (1 week)
Gradient Boosting	Xiaomeng Wang	November 13 to November 17 (1 week)
Majority Voting Classifier	Shirazush Salekin Chowdhury	November 13 to November 17 (1 week)
	Experiments	
Quantitative (Sampling with 20,000 data)	All members working on each respective algorithm	October 23 to November 17 (4 weeks)
Quantitative (Using all samples)		
Hyperparameter Tuning		
Report Writing for the Final		
Final Report	All members	November 20 to November 28 (1.5 weeks)
Final Report Formatting and Finalizing	All members	November 29 to December 06 (1 week)

VIII. Conclusion

In conclusion, Lifestyle profoundly impacts both individual well-being and societal dynamics. This project, utilizing machine learning algorithms, aims to predict lifestyle based on biological measurements, offering valuable insights into the intricate effects of stimuli on one's biochemical system. The assessment involved benchmarking nine machine learning algorithms, including three ensemble classifiers. Notably, the Majority Voting Ensemble classifier exhibited superior performance, achieving an F1-score of 80.15% and an accuracy of 75.85%. In contrast, the Naïve Bayes and K-Nearest Neighbors algorithms demonstrate comparatively modest outcomes, with F1-scores of 73.27% and 74.25%, and accuracies of 71.77% and 70.15%, respectively. Finally, we can say that we have successfully achieved what we have aimed for in our proposal report.

References

[1] Nivukoski, Ulla, et al. "Impacts of Unfavourable Lifestyle Factors on Biomarkers of Liver Function, Inflammation and Lipid Status." PLOS ONE, vol. 14, no. 6, June 2019, p. e0218463. PLoS Journals, https://doi.org/10.1371/journal.pone.0218463.

[2] Li, C., Sun, J. The impact of current smoking, regular drinking, and physical inactivity on health care-seeking behavior in China. BMC Health Serv Res 22, 52 (2022). https://doi.org/10.1186/s12913-022-07462-z.

[3] Ng, R., Sutradhar, R., Yao, Z., Wodchis, W. P., & Rosella, L. C. (2020). Smoking, drinking, diet and physical activity-modifiable lifestyle risk factors and their associations with age to first chronic disease. International journal of epidemiology, 49(1), 113–130. https://doi.org/10.1093/ije/dyz078.

[4] Oswalt, S. B., & Riddock, C. C. (2007). What to Do about Being Overwhelmed: Graduate Students, Stress and University Services. College Student Affairs Journal, 27(1), 24-44.

[5] Witkiewitz, K., Desai, S. A., Steckler, G., Jackson, K. M., Bowen, S., Leigh, B. C., & Larimer, M. E. (2012). Concurrent drinking and smoking among college students: An event-level analysis. Psychology of addictive behaviors : journal of the Society of Psychologists in Addictive Behaviors, 26(3), 649–654. https://doi.org/10.1037/a0025363.

[6] Little H. J. (2000). Behavioral mechanisms underlying the link between smoking and drinking. Alcohol research & health : the journal of the National Institute on Alcohol Abuse and Alcoholism, 24(4), 215–224.

[7] Kim S-Y, Park T, Kim K, Oh J, Park Y and Kim D-J (2021) A Deep Learning Algorithm to Predict Hazardous Drinkers and the Severity of Alcohol-Related Problems Using K-NHANES. Front. Psychiatry 12:684406. doi: 10.3389/fpsyt.2021.684406.

[8] Hsieh, S. Jean, et al. 'Biomarkers Increase Detection of Active Smoking and Secondhand Smoke Exposure in Critically Ill Patients'. Critical Care Medicine, vol. 39, no. 1, Jan. 2011, pp. 40–45. PubMed Central, https://doi.org/10.1097/CCM.0b013e3181fa4196.

[9] Mamoshina, P., Kochetov, K., Cortese, F. et al. Blood Biochemistry Analysis to Detect Smoking Status and Quantify Accelerated Aging in Smokers. Sci Rep 9, 142 (2019). https://doi.org/10.1038/s41598-018-35704-w.

[10] Mak, K. K., Lee, K., & Park, C. (2019). Applications of machine learning in addiction studies: A systematic review. Psychiatry research, 275, 53–60. https://doi.org/10.1016/j.psychres.2019.03.001.

[11] Thakur, S.S., Poddar, P. & Roy, R.B. Real-time prediction of smoking activity using machine learning based multi-class classification model. Multimed Tools Appl 81, 14529–14551 (2022). https://doi.org/10.1007/s11042-022-12349-6.

[12] Chhetri, Bijoy, et al. 'How Machine Learning Is Used to Study Addiction in Digital Healthcare: A Systematic Review'. International Journal of Information Management Data Insights, vol. 3, no. 2, Nov. 2023, p. 100175. ScienceDirect, https://doi.org/10.1016/j.jjimei.2023.100175.

[13] Yahyaoui, Amani, et al. 'A Decision Support System for Diabetes Prediction Using Machine Learning and Deep Learning Techniques'. 2019 1st International Informatics and Software Engineering Conference (UBMYK), 2019, pp. 1–4. IEEE Xplore, https://doi.org/10.1109/UBMYK48245.2019.8965556.

[14] Bonnell, L., Littenberg, B., Wshah, S., & Rose, G. L. (2020, May 1). A Machine Learning Approach to Identification of Unhealthy Drinking. Journal of the American Board of Family Medicine; American Board of Family Medicine. https://doi.org/10.3122/jabfm.2020.03.190421.

[15] Bisong, E. (2019). Logistic Regression. In: Building Machine Learning and Deep Learning Models on Google Cloud Platform. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-4470-8_20.

[16] H. Yusuff, N. Mohamad, U.K. Ngah, A.S.Yahaya, "Breast Cancer Analysis using Logistic Regression", International Journal of Recent Research and Applied Studies, Jan 2012, Volume 10, Issue 1, pp 14- 22.

[17] Conceptual Understanding of Logistic Regression for Data Science Beginners." Analytics Vidhya, https://www.analyticsvidhya.com/blog/2021/08/conceptual-understanding-of-logistic-regression-for-data-science-beginners/, Accessed: Oct. 6, 2023.

[18] Patel, Harsh H., and Purvi Prajapati. "Study and analysis of decision tree based classification algorithms." International Journal of Computer Sciences and Engineering 6.10 (2018): 74-78.

[19] JavaTpoint. "Machine Learning - Decision Tree Classification Algorithm." [Online]. Available: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm, Accessed: Oct. 6, 2023.

[20] Tangirala, Suryakanthi. "Evaluating the impact of GINI index and information gain on classification using decision tree classifier algorithm." International Journal of Advanced Computer Science and Applications 11.2 (2020): 612-619.

[21] I. Rish. IJCAI 2001 workshop on empirical methods in artificial intelligence , 3, page 41--46. IBM New York, (2001).

[22] R. MurtiRawat, S. Panchal, V. K. Singh and Y. Panchal, "Breast Cancer Detection Using K-Nearest Neighbors, Logistic Regression and Ensemble Learning," 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, 2020, pp. 534-540, doi: 10.1109/ICESC48915.2020.9155783.

[23] Huang, Shujun, et al. "Applications of Support Vector Machine (SVM) Learning in Cancer Genomics." Cancer Genomics & Proteomics, vol. 15, no. 1, Dec. 2017, pp. 41–51. PubMed Central, https://doi.org/10.21873/cgp.20063.

[24] Bento, Carolina. "Multilayer Perceptron Explained with a Real-Life Example and Python Code: Sentiment Analysis." Medium, 30 Sept. 2021, [Online]. Available:https://towardsdatascience.com/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentiment-analysis-cb408ee93141, Accessed: Oct. 6, 2023.

[25] Ms. Steffi Thomas, Mr. Atharva Joshi, Ms. Rutu Kalhapure, Mr. Deepraj Bhosale, Prof. Deepali Sonawane, 2020, Bionic ARM for Prosthetist, International Journal Of Engineering Research & Technology (IJERT) ICSITS – 2020 (Volume 8 – Issue 05).

[26] "ML | Voting Classifier using Sklearn." GeeksforGeeks, [Online], Available: https://www.geeksforgeeks.org/ml-voting-classifier-using-sklearn, Accessed: Oct. 6, 2023.

[27] "CKD-EPI Creatinine Equation (2021)." National Kidney Foundation, 1 Oct. 2021, [Online], Available: https://www.kidney.org/professionals/kdoqi/gfr_calculator/formula, Accessed: Oct 6, 2023.

[28] Levey AS, Stevens LA, Schmid CH, et al. A new equation to estimate glomerular filtration rate. Ann Intern Med. May 5 2009; 150(9): 604-612.

[29] How to Calculate Total Cholesterol from HDL and LDL. 30 Aug. 2022, https://www.medicalnewstoday.com/articles/how-to-calculate-total-cholesterol.

[30] S. Her, "Smoking & Drinking Dataset," Kaggle, [Online]. Available: https://www.kaggle.com/datasets/sooyoungher/smoking-drinking-dataset. Accessed: Oct. 6, 2023.

[31] Vidiyala, Ramya. "Performance Metrics for Classification Machine Learning Problems." Medium, 26 July 2020, [Online]. Available: https://towardsdatascience.com/performance-metrics-for-classification-machine-learning-problems-97e7e774a007, Accessed: Oct. 6, 2023.

[32] Dastjerdy, B.; Saeidi, A.; Heidarzadeh, S. Review of Applicable Outlier Detection Methods to Treat Geomechanical Data. *Geotechnics* 2023, *3*, 375-396. https://doi.org/10.3390/geotechnics3020022