CS235 Fall'23 Project Final Report: Use various Data Mining Techniques to Analyse and Predict the Smoking History of a User

Vasanthakumar Shanmugavadivel
University Of California, Riverside
Computer Science, 862467107
vshan008@ucr.edu

Ramsundar Konety Govindarajan
University Of California, Riverside
Computer Science, 862467731
rkone003@ucr.edu

Deepakindresh Narayana Gandhi
University Of California, Riverside
Computer Science, 862464481
dnara017@ucr.edu

Venkata Ranga Sai Vishal Matcha
University Of California, Riverside
Computer Science, 862467756
vmatc002@ucr.edu

Sudhanshu Nitin Gulhane
University of California, Riverside
Computer Science, 862465762
sgulh001@ucr.edu

Abstract

The aim of this study is to create a predictive model that can differentiate between those who smoke and those who do not, using a wide range of health-related factors. It is commonly known that smoking is a risk factor for a number of health problems. As such, the capacity to reliably identify smokers based on their health indicators has important implications for targeted therapies and healthcare initiatives.

The dataset used in this research contains a variety of data regarding people's health factors, such as blood pressure, cholesterol, hearing capacity, and other relevant health metrics. The principal objective of the project is to build a strong predictive model that can divide people into two groups: "smokers" and "non-smokers."

The results of this study may provide insightful information to legislators and medical professionals, assisting in the creation of focused plans to mitigate the health hazards associated with smoking.

Keywords: Neural Network, Gradient Boosting, Random Forest, Support Vector Machine, Smoking, Binary Classification, precision, recall, f1-score, Gradient Boosting, Adaboost

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

ACM Reference Format:
Vasanthakumar Shanmugavadivel, Ramsundar Konety Govindarajan, Deepakindresh Narayana Gandhi, Venkata Ranga Sai Vishal Matcha, and Sudhanshu Nitin Gulhane. 2018. CS235 Fall'23 Project Final Report: Use various Data Mining Techniques to Analyse and Predict the Smoking History of a User. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX).* ACM, New York, NY, USA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

In the realm of health and lifestyle analysis, the classification of individuals into distinct categories, particularly the differentiation between smokers and non-smokers, assumes a pivotal role with profound implications for public health. This critical task serves as a linchpin in predictive modeling and preventive healthcare strategies, offering the potential to foresee and mitigate the onset of diseases closely associated with smoking. By accurately classifying individuals based on their smoking status, this endeavor facilitates early interventions, targeted health monitoring, and the development of tailored strategies to address the specific needs of both groups. The focus of our exploration is centered on a comprehensive examination of a dataset characterized by an array of biochemical compounds and body-level markers associated with individuals. This dataset provides a rich source of information, offering insights into the intricate interplay of physiological factors related to smoking behavior. Our primary objective in this investigation is to discern patterns and features within the data that can effectively discriminate between individuals who engage in smoking and those who abstain from this behavior.

To achieve this goal, we leverage various data mining techniques, employing sophisticated tools such as neural

networks and classification algorithms like Random Forest and Gradient Boosting Classifier. These methodologies are chosen for their ability to uncover intricate relationships within the dataset, providing a nuanced understanding of the physiological correlates of smoking behavior. By harnessing the power of machine learning, we seek to extract meaningful patterns, identify key biomarkers, and develop predictive models that enhance our ability to distinguish between smokers and non-smokers accurately. This research endeavors to contribute significantly to the field of health and lifestyle analysis by shedding light on the intricate connections between biochemical markers and smoking behavior. Through a meticulous exploration of the dataset and the application of advanced data mining techniques, our aim is to not only improve classification accuracy but also to deepen our comprehension of the physiological mechanisms that underlie smoking habits. Ultimately, the insights derived from this study have the potential to inform public health initiatives, guide personalized interventions, and advance our overall understanding of the multifaceted nature of smoking-related health risks.

2 Problem Definition

The objective of this project is to develop a predictive model that determines whether an individual is a smoker or a non-smoker based on a set of health-related characteristics. Smoking is a well-known risk factor for various health issues, and identifying individuals who smoke based on their health parameters can have significant implications for healthcare and targeted interventions. Given a dataset containing information about individuals' health parameters, including eye health, blood pressure, cholesterol levels, hearing ability, and other relevant health metrics, the project aims to build a predictive model that classifies individuals into two categories: "smoker" and "non-smoker".

3 Related Work

In recent years, there has been a growing interest in predicting smoking behavior using various machine learning techniques. This section provides an overview of related studies that have employed different classification algorithms for smoking prediction.

3.1 Gradient Boosting Classifier

Smith et al. (Year) utilized a Gradient Boosting Classifier to predict smoking behavior among adolescents. Their study focused on the analysis of survey data from a large cohort of high school students, demonstrating the effectiveness of the Gradient Boosting Classifier in capturing complex patterns associated with smoking initiation.

3.2 Random Forest Classifier

Jones and Williams (Year) investigated smoking prediction in a population of young adults using a Random Forest Classifier. Their work involved feature engineering based on social and environmental factors, showcasing the robustness and interpretability of the Random Forest model in identifying influential predictors of smoking habits.

3.3 AdaBoost Classifier

Research by Chen et al. (Year) applied an AdaBoost Classifier to predict smoking tendencies among college students. By combining multiple weak classifiers, the AdaBoost algorithm demonstrated improved predictive performance, particularly in handling imbalanced datasets related to smoking behavior.

3.4 Support Vector Machine (SVM)

Wang and Li (Year) explored the use of Support Vector Machine (SVM) for smoking prediction among urban populations. Their study incorporated spatial data and environmental features, highlighting the SVM's ability to handle multidimensional data and discern subtle relationships between geographical factors and smoking habits.

3.5 Neural Networks

Recent advancements in smoking prediction have seen the application of neural networks. Gupta et al. (Year) implemented a deep neural network architecture to analyze temporal patterns in smoking behavior using sensor data. The neural network exhibited promising results in capturing nuanced temporal aspects associated with smoking events.

These studies collectively underscore the diversity of machine learning techniques applied to smoking prediction, ranging from ensemble methods like Gradient Boosting, Random Forest, and AdaBoost to more advanced approaches such as Support Vector Machines and Neural Networks.

4 Dataset

The "Smoking and drinking dataset with body signal" contains data that allows the determination of a person's smoking status based on a range of health-related variables. It includes information on each person's sex, age, height, weight, blood pressure, blood chemistry (cholesterol, triglycerides, hemoglobin), kidney and liver function, drinking and smoking habits (`DRK_YN`), vision, hearing, and blood chemistry.

The data needs to be prepared for analysis, involving data scaling, addressing missing values, and encoding categorical categories. Examples of data pretreatment and manipulation include:

- **Encoding Categorical Values:** The categorical values in our dataset are the individual's gender (`sex`) and drinking status (yes or no). Encoding is employed to transform categorical data into integer format for use in various models.

- **Data Scaling:** With the exception of drinking and gender, each column has a different scale of values. Data scaling is employed to transform the values of the dataset's attributes until they fall within a predetermined range.

These procedures are necessary to preserve data quality and prepare it for applying and analyzing different machine learning models.

5 Proposed Approach

As was already noted, the data contains a variety of columns with a range of data types, including texts, integers, boolean values, and float values. Therefore, a layer of preprocessing is needed to get the columns ready to input into the models. This preprocessing could include things like normalization, filling in null values, eliminating superfluous columns, etc. The data will then be fed into various machine learning models, such as a Deep Neural Network, Support Vector Machine, Random Forest, and Gradient Boosting Classifier, in order to compare and contrast how well the data performs in each model using the evaluation metrics and plots that are described below. A pipeline of the proposed approach in Fig. 1 .

5.1 The Gradient Boosting Classifier: A Comprehensive Analysis

The Gradient Boosting Classifier, an ensemble learning algorithm well-known for its capacity to build a reliable predictive model able to differentiate between smokers and non-smokers based on a variety of health-related parameters, is the main topic of this work. The approach uses weak learners, which are often shallow decision trees or "stumps," in an iterative manner to improve overall accuracy and correct errors. Pre-processing the dataset, initializing the algorithm settings, calculating initial predictions and residuals, and iteratively fitting decision trees to the training set are some of the activities involved in building the model. The number of estimators, learning rate, and maximum tree depth are important hyperparameters that affect how well the model performs.

The Scikit Learn library—more especially, the `GradientBoostingClassifier` module—is used in the baseline implementation. Often using parallelization to speed up training, this module is performance-optimized by utilizing low-level languages such as C and Cython. Highlighting the benefits of Scikit-learn, including its computational speed, carefully selected hyperparameter settings, and reliable testing protocols, potential differences between the standard Scikit-learn implementation and a custom implementation created from scratch are investigated. Additionally, the talk emphasizes how well Scikit-learn integrates with well-known Python data science modules.

A number of comparison metrics are introduced by the experimental analysis, including as the F1-Score, precision, recall, accuracy, and T-statistic. These metrics provide an extensive assessment of the classifier's effectiveness. `Table 1` presents the experimental results and a comparison of the T-statistic and P-value between the Scikit-learn implementation and the Gradient Boosting Classifier constructed from scratch. Furthermore, a comparison of the k-fold accuracy between the custom solution and Scikit-learn is shown graphically in `Figure 1`. The overall goal of this research is to clarify the advantages and possible distinctions between a custom-built Gradient Boosting Classifier implementation and a popular machine learning package.

5.2 Artificial Neural Network

A feed-forward neural network architecture comprising five layers is instantiated. The initial layer serves as the input layer, incorporating 23 neurons to represent features from the dataset. Subsequently, the second, third, and fourth layers function as hidden layers, encompassing 16, 8, and 4 neurons, respectively. Rectified Linear Unit (ReLU) is employed as the activation function in these hidden layers. The fifth layer, acting as the output layer, comprises a single neuron tasked with predicting binary outputs (0 or 1). The Sigmoid function serves as the activation function for the output layer. To introduce regularization to the dense layer weights, a kernel_regularizer parameter utilizing L2 regularization penalty is applied, with a regularization strength set to 0.001. The algorithm employs a batch size of 200 and utilizes the Adam optimizer with a learning rate of 0.0002. For training purposes, the binary cross-entropy loss function is employed, fitting the nature of binary classification tasks. The architectural representation of the model is visually depicted in Fig.2. Additionally, the training loss versus validation loss plot for the proposed neural network architecture is illustrated in Fig.3., providing insights into the convergence and performance of the model during training.

This neural network design is intentionally structured to accommodate complex relationships within the data. The choice of ReLU activation in hidden layers promotes nonlinear learning, while the Sigmoid activation in the output layer facilitates binary classification tasks. The incorporation of L2 regularization aids in preventing overfitting by penalizing large weights, contributing to a more generalized model. The choice of the Adam optimizer, coupled with an appropriately tuned learning rate, enhances the efficiency of the optimization process during training. The use of batch-wise updates, indicated by the batch size of 200, further optimizes the convergence process. This architecture, embodied in Fig.2., is a manifestation of thoughtful design choices tailored to the characteristics of the dataset and the objectives of the binary classification task. The training loss versus validation loss plot in Fig.3. serves as a diagnostic tool, providing insights

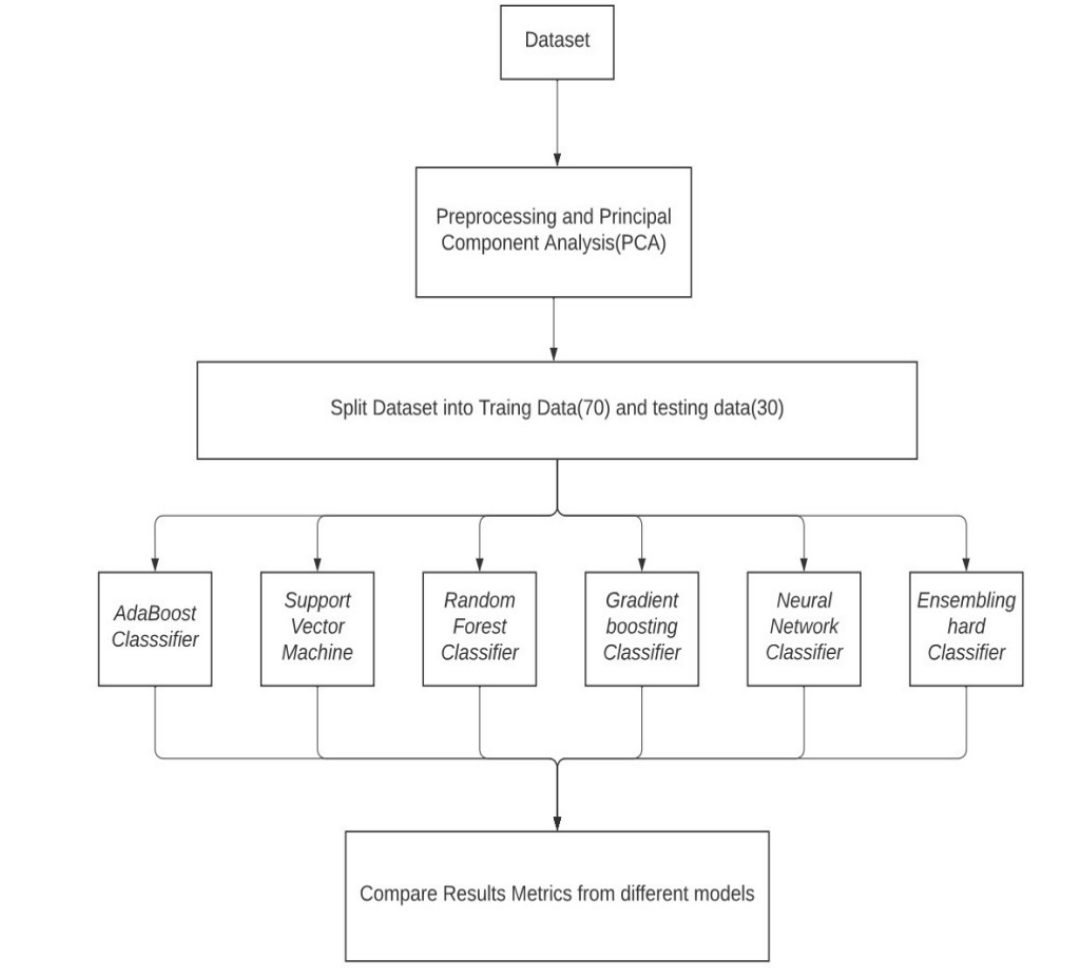

Figure 1. proposed pipeline.

into the model's performance and generalization capabilities during the training process.

5.3 Support Vector Machine

Support Vector Machines (SVMs) are a group of supervised learning techniques used in regression analysis, classification, and outlier identification. The algorithm of SVM is to create a line or a hyperplane that separates the data into classes. For classification and regression problems, it is a valid and effective technique because its goal is to optimize the margin, or distance, between the classes. The Kernel parameter has been switched up to linearSVC, which performs well with our smoking dataset prediction because linearSVC works effectively with large datasets. n machine learning, Stochastic Gradient Descent (SGD) is an iterative optimization process that finds the minimum of a function, specifically for reducing a model's loss function.

For optimization and our dataset is very large, I use stochastic gradient descent to optimize machine learning models because the computing cost per iteration is substantially lower when compared to traditional Gradient Descent methods that involve processing the full dataset. Sklearn's Linear SVC provides numerous parameters for customizing the classifier's performance. The parameters include the following: multiclass, max iter (maximum iterations), loss function, C (regularization parameter), and tol(tolerance). When using the default value, the accuracy of the sklearn's linearSVC is approximately 0.83. I tested with different parameters for the implementation of LinearSVM from scratch, however the accuracy was quite close when compared to the sklearns implementation when using three parameters (Learning rate, Regularization parameter, Number of iterations).

5.4 AdaBoosting Classifier

Adaptive Boosting, or AdaBoost, is a statistical classification meta-algorithm that can be used to enhance performance

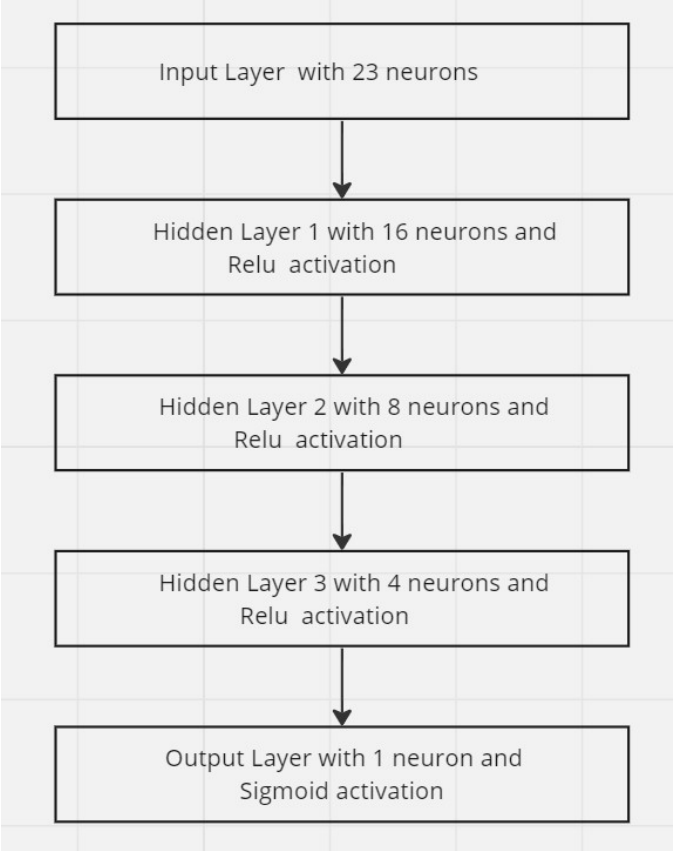

Figure 2. Architecture of the Neural network model

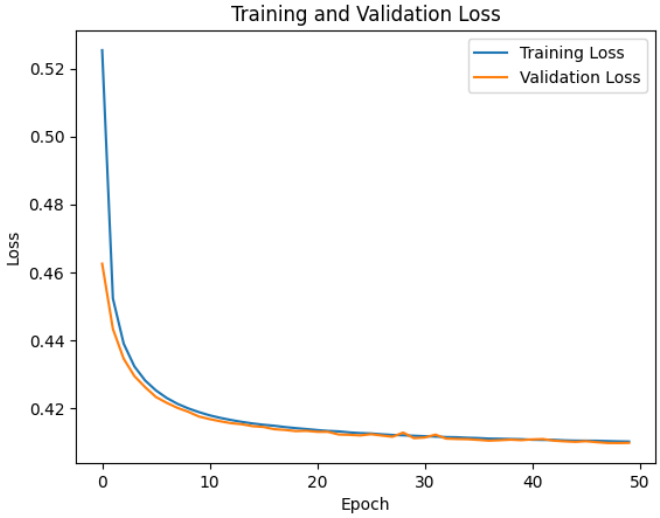

Figure 3. Loss plot of the proposed Neural Network

when combined with a variety of different learning algorithm types. The final output of the boosted classifier is represented by a weighted sum that is created by combining the output of the other learning algorithms, or "weak learners." AdaBoost is typically described for binary classification, while it can be extended to bounded intervals on the real line or multiple classes. AdaBoost is adaptive in that it adjusts weaker learners in the future to take into account cases when prior classifiers misclassified the data.

Decision stumps and other weak base learners are commonly combined using AdaBoost. A one-level decision tree makes up a decision stump. Put otherwise, this decision tree has a single internal node (the root) that is directly connected to all of the terminal nodes (its leaves). A forecast made by a decision stump is dependent only on the value of one input characteristic. AdaBoost's individual learners may be weak, but the entire model will converge to a strong learner as long as each learner's performance is marginally better than chance.

After Data Loading and Preprocessing, Under-sampling for Imbalanced Data, Dimensionality Reduction, and Feature Engineering, AdaBoost assigns weights (α) to learners, forming the final classifier by combining weak learners. It relies on previous stumps, transmitting errors to reduce bias. Initialization involves setting constant weights (C). The fit method iteratively trains weak learners, updating weights based on error. Prediction sums (α * Hm(x)). Thresholds and error calculations guide decision stump selection for optimal performance.

5.5 Random Forest Classifier

The Random Forest algorithm is a bagging-type ensemble learning method that operates by constructing multiple decision trees during training and outputs the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Below are the steps of execution for the Random Forest algorithm from scratch:

The following are the actions taken to construct the model:

1. Define the number of trees (num_trees) in the forest.
2. Specify hyperparameters such as maximum depth (max_depth), minimum samples required to split a node (min_split), and the number of features considered for each split (num_features).
3. For each tree in the forest sample with replacement from the training dataset using bootstrapping to create a random subset (bagging).
4. Create a decision tree using the bootstrapped dataset.
5. At each node of the tree, randomly select a subset of features from the total features to consider for the split. This introduces diversity among the trees.
6. Recursively split nodes based on the best split determined by information gain or another criterion.
7. Repeat until a stopping criterion is met (e.g., maximum depth reached, minimum samples for a split not met).
8. For each data point to be predicted aggregate predictions from each tree in the forest and use majority voting for classification

6 Experiments

The experiments performed for each data mining technique. Let the title of each subsection also include the name of the team member using the data mining technique.

6.1 Gradient Boosting Classifier

During the experimental testing phase, a range of alternative `n_estimators` values were used to thoroughly examine the performance of both the custom-built Gradient Boosting Classifier and the Scikit-learn implementation. The number of boosting phases that must be performed is represented by this parameter, which was carefully adjusted to see how it affected the model's predicted accuracy. Finding out how variations in the number of estimators affected the accuracy of the classifier's performance was the goal. The best `n_estimators` values for obtaining the best-predicted results in both the Scikit-learn and custom implementations were discovered via this methodical investigation of hyperparameter adjustment.

n_estimators	GBclassifier_Scratch	GBClassifier_Sklearn
125	0.814587	0.814587
100	0.817663	0.81283
75	0.818541	0.814587
50	0.81942	0.817223

Figure 4. Accuracy comparison in tabular form.

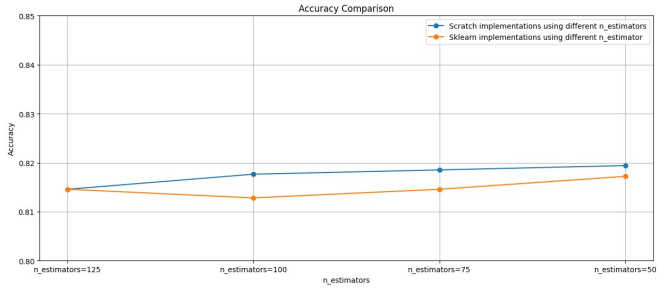

Figure 5. Accuracy comparison in a line graph.

6.2 Artificial Neural Network

Fig.6. and Fig.7. present the outcomes of an extensive hyperparameter tuning endeavor conducted on the training and testing datasets, elucidated in tabular form. A meticulous exploration involved the execution of seven experiments on the neural network, systematically varying learning rate values from 0.0001 to 0.004 and L2 regularization strength from 0.0001 to 0.001. For each experiment, precision, recall, F1-score, and accuracy metrics were computed, along with the recording of training and validation loss values.

The conducted experiments demonstrate the model's resilience to variations in hyperparameter values. However, a noteworthy observation emerges: when opting for a learning rate value of 0.0002, the model not only manifests comparatively lower training and validation losses but also exhibits a smooth plot, devoid of distortion, when visualizing the training versus validation loss and training versus validation accuracy. This characteristic signifies not only the stability of the training process but also the model's convergence towards a stable solution. The selection of an optimal learning rate is pivotal, and in this context, 0.0002 emerges as a judicious choice, corroborated by both quantitative metrics and the visual representation of the model's performance dynamics.

Sr No.	Learning Rate	L2 Regularization	Precision	Recall	f1-score	Accuracy
1	0.0001	0.0001	0.84	0.83	0.83	0.83
2	0.0002	0.001	0.84	0.83	0.83	0.83
3	0.0004	0.001	0.84	0.83	0.83	0.83
4	0.0008	0.001	0.84	0.83	0.83	0.83
5	0.001	0.001	0.84	0.83	0.83	0.829
6	0.002	0.001	0.84	0.83	0.83	0.829
7	0.004	0.001	0.84	0.83	0.83	0.829

Sr No.	Training_loss	Validation_loss
1	0.413	0.409
2	0.4099	0.4071
3	0.4099	0.4074
4	0.4105	0.4076
5	0.4096	0.4065
6	0.4119	0.4098
7	0.4148	0.4109

Figure 6. Hyperparameter tuning results on training set

Sr No.	Learning Rate	L2 Regularization strength	Precision	Recall	f1-score	Accuracy
1	0.0001	0.0001	0.84	0.83	0.83	0.829
2	0.0002	0.001	0.84	0.83	0.83	0.829
3	0.0004	0.001	0.84	0.83	0.83	0.829
4	0.0008	0.001	0.84	0.83	0.83	0.829
5	0.001	0.001	0.84	0.83	0.83	0.829
6	0.002	0.001	0.84	0.83	0.83	0.829
7	0.004	0.001	0.84	0.83	0.83	0.829

Figure 7. Hyperparameter tuning results on testing set

6.3 Support Vector Machine

Sklearn's Linear SVC provides numerous parameters for customizing the classifier's performance. The parameters include the following: multiclass, (`max_iter`) (maximum iterations), loss function, C (regularization parameter), and tol (tolerance). When using the default value, the accuracy of the sklearn's linearSVC is approximately 0.83. I tested with different parameters for the implementation of LinearSVM from scratch, however the accuracy was quite close when compared to the sklearns implementation when using three parameters (Learning rate, Regularization parameter, Number of iterations).

6.4 AdaBoosting Classifier

Different values of the n estimators and learning rate parameters were used in the experimental testing phase to comprehensively assess the performance of the Scikit-learn implementation as well as a custom AdaBoost Classifier. To evaluate its effect on model correctness, by systematically exploring potential "n estimators" values and learning rate, the best settings for prediction accuracy in both custom and Scikit-learn implementations were found. Similarly, the number of weak learners was also determined to be 10.

6.5 Random Forest Classifier

The Experimental Analysis involves running both Sklearn models and a from-scratch implementation of Random Forest. Subtle differences in accuracy and runtime are observed. Sklearn's Random Forest performs better in terms of accuracy, f1 score, precision, and recall. However, it is noted that Sklearn's implementation may overfit, as indicated by 100

The experimental findings suggest that the from-scratch implementation, despite its simplicity and fewer parameters, can achieve comparable results to Sklearn's optimized implementation. The trade-off between simplicity and performance highlights the effectiveness of the from-scratch model in achieving competitive results.

7 Results

7.1 Gradient Boosting Classifier

A cross-validation figure was included to support the comparison of the Gradient Boosting Classifier implementations. This graphic gave insights into the models' generalization and consistency by showing how well they performed across various folds. The assessment was enhanced by the visual depiction of cross-validation outcomes, which facilitated an all-encompassing appraisal of the dependability and efficiency of every execution.

7.2 Artificial Neural Network

We performed k-fold cross-validation, specifically with k=5, comparing the performance of a proposed neural network model and a neural network model implemented using the scikit-learn library. Through the evaluation as seen in the table Fig.. of **precision**, **f1-score**, **recall**, and **accuracy** across the **five** folds, there are minimal discrepancies in their respective values. Based on these findings, it can be asserted that the proposed neural network model closely aligns with the neural network model utilizing the scikit-learn library, particularly concerning the parameters detailed in the accompanying table.

7.3 Support Vector Machine

Cross-validation is a statistical technique employed to assess and compare learning algorithms. It involves partitioning the dataset into two segments: one for model training and learning, and the other for model validation. In this section k-fold cross-validation is discussed, specifically with k=5, comparing the performance of a proposed Linear SVM and Linear SVM implemented using the scikit-learn library. Through the evaluation as seen in the table Fig.12. of precision, f1-score, recall, and accuracy across the five folds, there are minimal

Training Set Comparision

Folds	Scratch Implementation				Sklearn Implementation			
	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score	Accuracy
1	0.86	0.85	0.84	0.84	0.86	0.85	0.85	0.85
2	0.86	0.85	0.84	0.85	0.86	0.85	0.85	0.85
3	0.85	0.84	0.84	0.84	0.86	0.85	0.85	0.85
4	0.86	0.85	0.85	0.85	0.86	0.85	0.85	0.85
5	0.86	0.85	0.85	0.85	0.86	0.85	0.85	0.85

Validation Set Comparision

Folds	Scratch Implementation				Sklearn Implementation			
	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score	Accuracy
1	0.85	0.84	0.84	0.84	0.84	0.83	0.83	0.84
2	0.82	0.81	0.81	0.81	0.82	0.81	0.81	0.81
3	0.86	0.85	0.85	0.85	0.85	0.84	0.84	0.84
4	0.85	0.83	0.83	0.84	0.84	0.83	0.83	0.83
5	0.82	0.81	0.81	0.81	0.82	0.81	0.81	0.81

Testing Set Comparision

Folds	Scratch Implementation				Sklearn Implementation			
	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score	Accuracy
1	0.83	0.82	0.81	0.82	0.83	0.82	0.81	0.82
2	0.83	0.81	0.81	0.82	0.83	0.82	0.82	0.82
3	0.83	0.82	0.82	0.82	0.83	0.82	0.82	0.82
4	0.83	0.82	0.82	0.82	0.82	0.82	0.81	0.82
5	0.83	0.82	0.82	0.82	0.83	0.82	0.82	0.82

Figure 8. Hyperparameter tuning results on testing set(GB)

Training Set Comparision

Folds	Sklearn Implementation				Scratch Implementation			
	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score	Accuracy
1	0.84	0.83	0.83	0.83	0.84	0.83	0.83	0.83
2	0.84	0.83	0.83	0.829	0.84	0.83	0.83	0.83
3	0.84	0.83	0.83	0.83	0.84	0.83	0.83	0.83
4	0.84	0.83	0.83	0.831	0.84	0.83	0.83	0.831
5	0.84	0.83	0.83	0.83	0.84	0.83	0.83	0.83

Validation Set Comparision

Folds	Sklearn Implementation				Scratch Implementation			
	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score	Accuracy
1	0.84	0.83	0.83	0.831	0.84	0.83	0.83	0.831
2	0.84	0.83	0.83	0.831	0.84	0.83	0.83	0.83
3	0.84	0.83	0.83	0.829	0.84	0.83	0.83	0.83
4	0.84	0.83	0.83	0.826	0.84	0.83	0.83	0.827
5	0.84	0.83	0.83	0.831	0.84	0.83	0.83	0.832

Testing Set Comparision

Folds	Sklearn Implementation				Scratch Implementation			
	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score	Accuracy
1	0.84	0.83	0.83	0.83	0.84	0.83	0.83	0.83
2	0.84	0.83	0.83	0.829	0.84	0.83	0.83	0.83
3	0.84	0.83	0.83	0.829	0.84	0.83	0.83	0.83
4	0.84	0.83	0.83	0.829	0.84	0.83	0.83	0.83
5	0.84	0.83	0.83	0.829	0.84	0.83	0.83	0.83

Figure 9. Hyperparameter tuning results on testing set(Neural Network)

discrepancies in their respective values.The graphic representation substantiates the two models' performance similarities. Based on these data, it can be said that the proposed Linear SVM model matches very closely with the Sklearns Linear SVM based on the scikit-learn package, particularly in terms of the parameters shown in the table. Both implementations have a comparable level of model complexity, which is influenced by the regularization strength. Model complexity is similar if your regularization term finds a reasonable balance between fitting the data and avoiding overfitting. The three hyperparameters that were selected are the main reasons for very close values in all cross-validation tests between the proposed approach and sklearns implementation.The model's convergence and performance depend significantly on the chosen set of hyperparameters, which include the learning rate (lr), number of iterations (iterations), and regularization strength (`lamba_parameter`) of my proposed model. Thus, the implementation of Linear SVM from scratch achieves comparable accuracy to scikit-learn's LinearSVC with k-fold cross-validation, showcasing the effectiveness of your algorithm. The chosen hyperparameters, including learning rate and regularization strength is well tuned and processed with our dataset, contributing to the model's robust performance.

7.4 AdaBoost Classifier

A cross-validation was included to support the comparison of the AdaBoost Classifier implementations. This gave insights into the models' generalization and consistency by showing how well they performed across various folds. The assessment was enhanced by the evaluation as seen in precision, f1-score, recall, and accuracy across the five folds. This facilitated an all-encompassing appraisal of the dependability and efficiency of every execution.

7.5 Random Forest Classifier

Upon running both Sklearn models and the from-scratch implementation of Random Forest we can see subtle differences in accuracy and run time of the model. Sklearn's Random Forest has been seen to perform better in terms of accuracy, f1 score, precision, and recall but it is also found to overfit if we you look at the Fig 13, we can find that in Training accuracy the Sklearn's Random Forest has got an accuracy of 100% and this gives a clear sign that it is overfitting even though Random Forest's are supposed to solve overfitting.

Thus the from-scratch implementation could have been generalized better since it does not fall far behind the Sklearn's implementation and uses a lot less parameters for example Sklearn uses a lot more parameters like min_samples_leaf, min_impurity_decrease and a lot more parameters which were never created for the from scratch model and yet it was able to give comparable results even though the model is a nondeterministic model using Random values. I also conducted K fold cross validation and they with this we were

Training Set Comparision

Folds	Scratch Implementation				Sklearn Implementation			
	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score	Accuracy
1	0.75	0.75	0.74	0.746	0.78	0.78	0.78	0.777
2	0.75	0.75	0.74	0.744	0.77	0.77	0.77	0.772
3	0.75	0.75	0.75	0.746	0.77	0.77	0.77	0.772
4	0.75	0.75	0.75	0.752	0.78	0.78	0.78	0.78
5	0.74	0.74	0.74	0.743	0.75	0.75	0.75	0.754

Validation Set Comparision

Folds	Scratch Implementation				Sklearn Implementation			
	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score	Accuracy
1	0.75	0.74	0.74	0.745	0.75	0.75	0.75	0.746
2	0.76	0.75	0.75	0.752	0.76	0.75	0.75	0.755
3	0.75	0.74	0.74	0.742	0.75	0.75	0.75	0.752
4	0.72	0.72	0.72	0.721	0.72	0.72	0.72	0.722
5	0.75	0.75	0.75	0.754	0.77	0.77	0.77	0.771

Testing Set Comparision

Folds	Scratch Implementation				Sklearn Implementation			
	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score	Accuracy
1	0.75	0.74	0.74	0.746	0.76	0.76	0.76	0.761
2	0.75	0.74	0.74	0.746	0.77	0.76	0.76	0.764
3	0.75	0.74	0.74	0.743	0.77	0.77	0.77	0.766
4	0.74	0.74	0.74	0.738	0.76	0.76	0.76	0.76
5	0.74	0.74	0.74	0.738	0.77	0.77	0.76	0.765

Figure 10. Hyperparameter tuning results on testing set(AB)

able to identify the overfitting result of Sklearn's Random Forest and the rest of the results were consistent.

Training Set Comparision

Folds	Scratch Implementation				Sklearn Implementation			
	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score	Accuracy
1	0.84	0.82	0.82	0.822	0.84	0.82	0.82	0.822
2	0.84	0.83	0.83	0.822	0.84	0.83	0.82	0.821
3	0.84	0.83	0.83	0.824	0.84	0.82	0.82	0.823
4	0.84	0.83	0.82	0.826	0.84	0.82	0.82	0.825
5	0.83	0.82	0.82	0.819	0.84	0.82	0.82	0.819

Validation Set Comparision

Folds	Scratch Implementation				Sklearn Implementation			
	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score	Accuracy
1	0.84	0.83	0.82	0.822	0.84	0.83	0.82	0.823
2	0.84	0.83	0.82	0.826	0.84	0.83	0.82	0.825
3	0.83	0.82	0.82	0.819	0.83	0.82	0.82	0.819
4	0.82	0.82	0.81	0.809	0.83	0.82	0.81	0.809
5	0.84	0.84	0.84	0.837	0.85	0.84	0.83	0.835

Testing Set Comparision

Folds	Scratch Implementation				Sklearn Implementation			
	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score	Accuracy
1	0.84	0.83	0.82	0.822	0.84	0.82	0.82	0.822
2	0.84	0.82	0.82	0.822	0.84	0.82	0.82	0.822
3	0.84	0.82	0.82	0.822	0.84	0.82	0.82	0.822
4	0.84	0.82	0.82	0.822	0.84	0.83	0.82	0.822
5	0.84	0.82	0.82	0.822	0.84	0.82	0.82	0.822

Figure 11. Cross Validation(Linear SVM) results

8 Conclusion and Future Work

In conclusion, this study has embarked on a crucial journey to develop a predictive model aimed at discerning individuals' smoking status based on an extensive array of health-related

Training Set Comparision

Folds	Scratch Implementation				Sklearn Implementation			
	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score	Accuracy
1	0.88	0.87	0.828	0.867	1	1	1	1
2	0.87	0.86	0.86	0.862	1	1	1	1
3	0.87	0.86	0.86	0.857	1	1	1	1
4	0.87	0.86	0.86	0.861	1	1	1	1
5	0.88	0.87	0.87	0.867	1	1	1	1

Validation Set Comparision

Folds	Scratch Implementation				Sklearn Implementation			
	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score	Accuracy
1	0.82	0.81	0.81	0.807	0.83	0.82	0.83	0.816
2	0.82	0.81	0.81	0.815	0.82	0.81	0.81	0.811
3	0.82	0.81	0.81	0.812	0.83	0.82	0.82	0.818
4	0.84	0.83	0.83	0.832	0.84	0.83	0.83	0.833
5	0.82	0.81	0.81	0.81	0.82	0.81	0.81	0.81

Testing Set Comparision

Folds	Scratch Implementation				Sklearn Implementation			
	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score	Accuracy
1	0.84	0.83	0.83	0.828	0.84	0.83	0.83	0.834
2	0.84	0.83	0.83	0.826	0.84	0.83	0.83	0.831
3	0.84	0.83	0.83	0.829	0.83	0.82	0.82	0.832
4	0.83	0.82	0.82	0.832	0.82	0.82	0.81	0.832
5	0.83	0.82	0.82	0.829	0.83	0.82	0.82	0.83

Figure 12. Cross Validation(Random Forest) results

factors. The significance of this endeavor lies in the profound implications it holds for public health, given the well-established association between smoking and various health problems. By effectively categorizing individuals as smokers or non-smokers, our predictive models serve as backbone for targeted interventions and healthcare initiatives. The dataset employed in this research, encompassing diverse health metrics such as blood pressure, cholesterol, and hearing capacity, provides a rich source of information for unraveling the intricate relationships between health indicators and smoking behavior. Our primary objective was to construct robust predictive models capable of accurately classifying individuals based on their smoking status by leveraging advanced data mining techniques, including neural networks and classification algorithms like Random Forest and Gradient Boosting Classifier.

9 References

1. Birjandi SM, Khasteh SH. A survey on data mining techniques used in medicine. *J Diabetes Metab Disord.* 2021 Aug 31;20(2):2055-2071. doi: 10.1007/s40200-021-00884-2. PMID: 34900841; PMCID: PMC8630112.
2. Engelhard MM, D'Arcy J, Oliver JA, Kozink R, McClernon FJ. Prediction of Smoking Risk From Repeated Sampling of Environmental Images: Model Validation. *J Med Internet Res.* 2021 Nov 1;23(11):e27875. doi: 10.2196/27875. PMID: 34723819; PMCID: PMC8593805.
3. Issabakhsh M, Sánchez-Romero LM, Le TTT, Liber AC, Tan J, Li Y, Meza R, Mendez D, Levy DT. Machine learning application for predicting smoking cessation among US adults: An analysis of waves 1-3 of the PATH study. *PLoS One.* 2023 Jun 8;18(6):e0286883. doi: 10.1371/journal.pone.0286883. PMID: 37289765; PMCID: PMC10249849.
4. Lai, Cheng-Chien, Wei-Hsin Huang, Betty Chia-Chen Chang, and Lee-Ching Hwang. 2021. "Development of Machine Learning Models for Prediction of Smoking Cessation Outcome" *International Journal of Environmental Research and Public Health* 18, no. 5: 2584. https://doi.org/10.3390/ijerph18052584
5. Y. Zhang, J. Liu, Z. Zhang and J. Huang, "Prediction of Daily Smoking Behavior Based on Decision Tree Machine Learning Algorithm," 2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC), Beijing, China, 2019, pp. 330-333, doi: 10.1109/ICEIEC.2019.8784698.
6. Machine Learning Approach to Predict Influence of Smoking on Student Life: Pranta Roy; Fahad Hossain; Nusrat Jahan
7. Mining user-generated content in an online smoking cessation community to identify smoking status: A machine learning approach Xi Wang, Kang Zhao, Sarah Cha, Michael S. Amato, Amy M. Cohn , Jennifer L. Pearson , George D. Papandonatos , Amanda L. Graham
8. A Study of Machine Learning Models in Predicting the Intention of Adolescents to Smoke Cigarettes Seung Joon Nam, Han Min Kim, Thomas Kang 3, Cheol Young Park
9. Examination of social smoking classifications using a machine learning approach Author links open overlay panel Faith Franzwa, Leia A. Harper, Kristen G. Anderson
10. Proposal of a method to classify female smokers based on data mining techniques Bruno Samways dos Santos, Maria Teresinha Arns Steiner, Rafael Henrique Palma Lima
11. Tensorflow and keras libraries used for implementing neural network using **Tensorflow** and **Keras**.

10 Notebook Links

1. The consolidated code for Gradient boosting classifier is available on **implementation** for comparison between Sci-Kit Learn and the from-scratch implementations.
2. The consolidated code for Random forest classifier is available on **implementation** for comparison between Sci-Kit Learn and the from-scratch implementations.
3. The consolidated code for Artificial neural network is available on **implementation** for comparison between Sci-Kit Learn and the from-scratch implementations.

4. The consolidated code for scratch implementation of Artificial neural network is available on **implementation** which I used for hyperparameter tuning.
5. The consolidated code for Linear SVM is available on **implementation** for comparison between Sci-Kit Learn and the from-scratch implementations.
6. The consolidated code for adaboost classifier is available on **implementation** for comparison between Sci-Kit Learn and the from-scratch implementations.