Diabetes Prediction: Leveraging Machine Learning for Early Detection

Prateek Mishra, Soham Purushan, Amy Zhou, Callum Guan

Abstract

Diabetes is one of the most common diseases in the world and is especially a major problem in the United States. This disease affects the body's ability to break down sugars, glucose in particular, in the blood. Glucose is an important source of energy for the cells that make up the muscles and tissues. It's also the brain's main source of fuel. Whether genetically developed or not, the main cause of diabetes varies by type. But no matter the type of diabetes, when the body loses the ability to break down blood sugar, large amounts build-up and remain in the bloodstream. Excessive amounts can lead to serious health issues such as heart disease and kidney failure.

This project evaluates various classical Machine Learning methods as well as a Deep Learning model in their ability to classify patients as non-diabetic, pre-diabetic, or diabetic. This is based on survey responses asking participants for measures of blood pressure, cholesterol, general health, and other factors which are used as predictors in building a classification model. We first use unsupervised learning methods to explore patterns in our data, with the main goal of reducing dimensionality through PCA and applying clustering algorithms to see if the diabetes target variable classes are well separated by the clusters. We also study multicollinearity and regularization for feature importance and selection. Then, we proceed with predictive modeling and find that tree-based methods like XGBoost as well as the supervised learning method Support Vector Machine outperform many other models. Our primary means of model evaluation is not only accuracy, but more so false negative rate. Since our data has severe class imbalances, as the vast majority of the individuals are non-diabetic Class 0, the model will mostly predict 0 and therefore have a high accuracy since most test observations will also be Class 0. Further examination shows us that the performance on predicting the minority class, diabetic individuals, is poor as we find high false negative rates. This means that patients will be told they do not have when they actually do, which is particularly concerning. As a result, we place a heavy focus on decreasing this false negative rate.

1 Introduction

Motivation and Goals.

The prevalence of diabetes is a growing global health concern, and accurate early detection plays a crucial role in effective management and prevention. This project aims to develop Machine Learning (ML) models to classify individuals as diabetic, pre-diabetic, or non-diabetic using health indicators such as BMI, physical activity, smoking status, etc. We leveraged data from 2015 Behavioral Risk Factor Surveillance System (BRFSS) surveys, and we wanted to explore different ML techniques to enhance classification performance. The Behavioral Risk Factor Surveillance System (BRFSS) is a survey that is

collected annually by the CDC. Each year, the survey collects responses from over 400,000 Americans on health-related risk behaviors, chronic health conditions, and the use of preventative services. For this project we used a cleaned version of the original dataset that is available on Kaggle from 2015. This original dataset[1] contains responses from 441,455 individuals and has 330 features.

Data.

As mentioned earlier the data comes from the Behavioral Risk Factor Surveillance System (BRFSS) conducted each year by the CDC. From this original raw data the kaggle dataset included three preprocessed and cleaned dataset transformed from this data. You can view the process of cleaning the original data by the provider of this dataset here[2]. We were given three datasets from kaggle that we will work with and whose class distributions for the response variable are shown in Figure 1. First there is diabetes_multiclass in which individuals are in one of three classes: 0 for no diabetes, 1 for pre-diabetes, and 2 for has diabetes. The classes are unbalanced as most observations are of Class 0. Next, we have diabetes_binary_unbalanced in which individuals are either Class 0 for no diabetes or Class 1 for has diabetes. The classes are unbalanced as most are of Class 0. Finally, we have diabetes_binary_balanced in which individuals are again either Class 0 or 1, but this time, there are an equal number of observations in Class 0 as there are in Class 1. It is important to note that both diabetes_multiclass and diabetes_binary_unbalanced have 253,680 observations and 21 features, while diabetes_binary_balanced has 70,692 observations and 21 features.

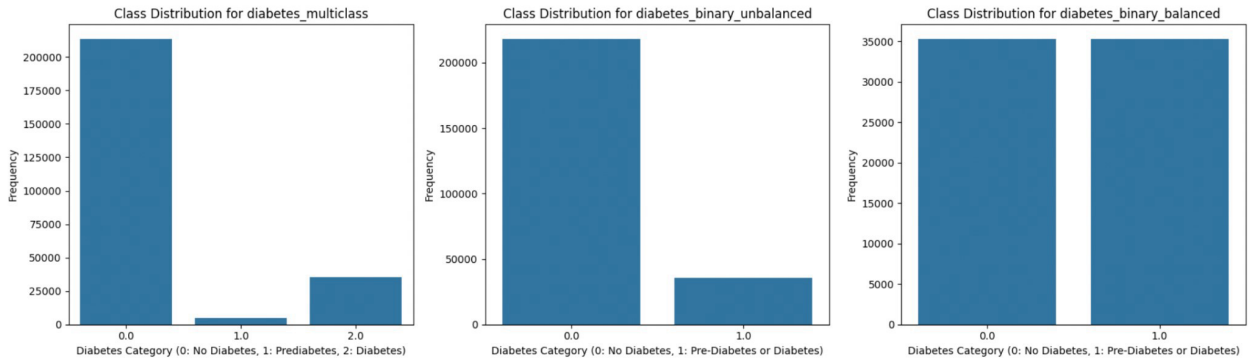


Figure 1

2 Exploratory Data Analysis (EDA)

We approach EDA in three main ways — use unsupervised learning techniques such as Principal Component Analysis and Clustering to discover underlying patterns in the data; assess multicollinearity through Variance Inflation Factor (VIF); use regularization techniques with logistic regression to see how to handle multicollinearity as well as feature selection.

Unsupervised Methods for Dimensionality Reduction and Pattern Recognition.

We want to see if using clustering algorithms will create clusters such that the diabetes and non-diabetes classes in the multiclass and binary settings are well-separated — meaning that each of the 3 multiclass clusters should have samples primarily one of Class 0, Class 1, or Class 2, while each of the 2 binary clusters should have samples primarily from Class 0 or Class 1. We applied PCA to the datasets to reduce the dimensionality to 2 principal components, allowing us to visualize our clusters on a cartesian plane.

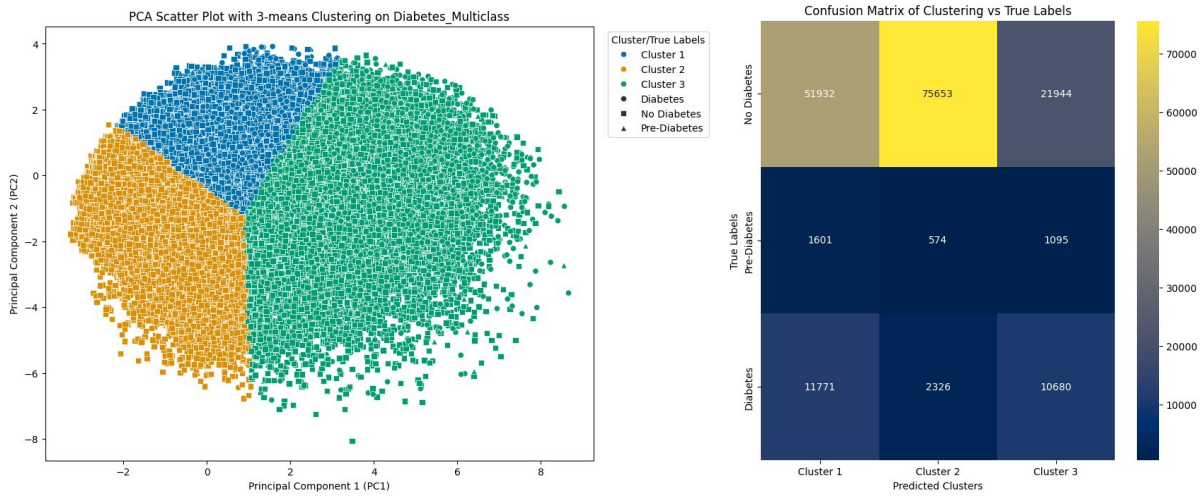


Figure 2

In the multiclass data (Figure 2), the 3 classes were not clearly well separated. In the confusion matrix, we are given the clusters and how many observations of each class they contained. Each class was generally spread across the 3 clusters, and no cluster had mainly one class. We try a similar implementation on the binary dataset (Figure 3). The diabetic individuals were almost split 50/50 between the two clusters, while the non-diabetic individuals were mostly in cluster 1, but a considerable amount was in cluster 2. Once again, the clusters did not separate the classes well. Since we have the true labels of individuals, their diabetes class, we will continue our analysis by studying supervised methods like decision trees and neural networks to take advantage of what is given in the data. We expect that supervised learning will be effective in our classification task.

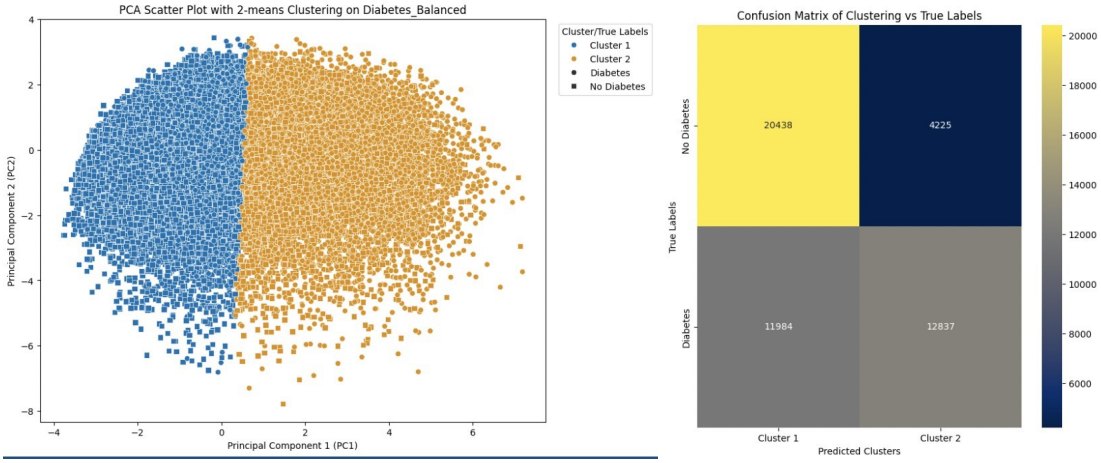


Figure 3

Assessing Multicollinearity.

Since we have 21 predictors, 20 of which are categorical, our data might be subject to linear dependence and thus may suffer from multicollinearity. We explore this through VIF values and a correlation matrix of

the features. The left side of Figure 4 displays the calculated VIF values and the right displays the correlation matrix. Looking at either the bottom triangular or the upper triangular of the correlation matrix will suffice since it is symmetric.

	Feature	VIF
0	HighBP	2.299761
1	HighChol	2.029660
2	CholCheck	23.187436
3	BMI	18.149913
4	Smoker	1.933042
5	Stroke	1.126777
6	HeartDiseaseorAttack	1.289820
7	PhysActivity	4.645314
8	Fruits	3.032775
9	Veggies	5.826886
10	HvyAlcoholConsump	1.083523
11	AnyHealthcare	20.839710
12	NoDocbcCost	1.215834
13	GenHlth	10.740162
14	MentHlth	1.463103
15	PhysHlth	1.999556
16	DiffWalk	1.838551
17	Sex	1.910708
18	Age	9.886830
19	Education	29.507416
20	Income	14.156118

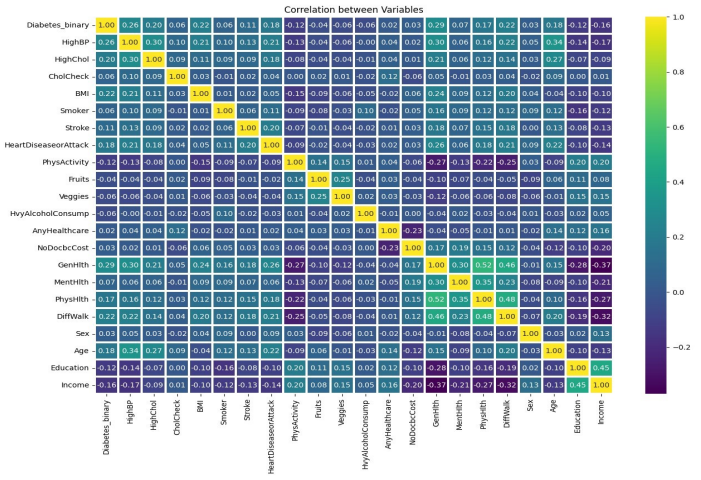


Figure 4

VIF values of 1 indicate no multicollinearity, values between 1-5 indicate a mild degree of multicollinearity, values greater than 5 indicate a high degree of multicollinearity, and values greater than 10 indicate severe multicollinearity. For example, the VIF for CholCheck is extremely concerning at 23.187, so we can look at the correlation matrix to see if it is highly correlated with many features. We find this in either the fourth row or fourth column, but we see that no correlation exceeds 0.1. Additionally, our highest correlation in general is .57 which is moderate. Although we have high VIF values for features like CholCheck, the pairwise correlations that we find from the correlation matrix are not very strong. This suggests that the high VIF may be a result of correlation of many combined features. So, dropping features like CholCheck is not warranted. We will continue our analysis by seeing if regularization techniques handle this multicollinearity well or nullify irrelevant predictors, producing model sparsity.

Feature Selection.

We use logistic regression alongside regularization techniques and p-values to explore feature associations, how good of a predictor is a particular feature, and to perform feature selection. We fit a baseline logistic regression model on the unbalanced binary dataset. By doing so, we want to achieve two things: evaluate its performance and explore what nuances and aspects of the data modeling process will need attention throughout our study and to also have a baseline performance to which we can compare the various ML models we use throughout our analysis.

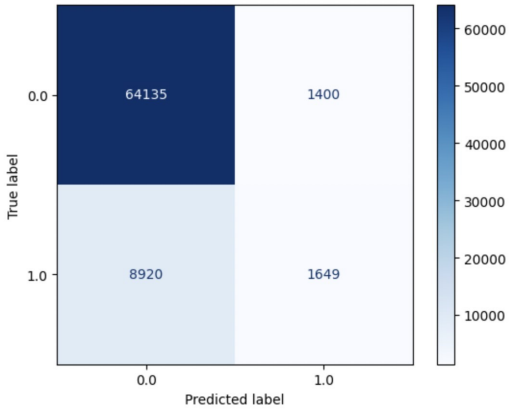


Figure 5

The performance of the logistic regression model is given by the confusion matrix in Figure 5. At first glance, we find that we have a considerably good accuracy of 86.44%, but when we take a deeper look, we find that this is because of the class imbalance. With only about 14% of our data in class 1, the model becomes biased and tends to mostly predict Class 0, and, as a result, it is accurate ~86% of the time since that is how many observations are of Class 0. This brings us to a major theme of our study and what we will primarily concern ourselves throughout the rest of this analysis. Because of the class imbalance, we achieve high accuracy, but when looking at how the model performs on each class, we find that we have a false negative rate (FNR) of 84.39% (8920 individuals classified as 0 when they are truly 1). This extreme value is highly concerning as, in the context of our data, we are predicting patients to be Class 0 when they are actually Class 1 very often. In other words, we would be telling a patient that they don't have diabetes when they actually do, and this misdiagnosis comes with dire consequences. Thus, the bulk of our study will be trying to decrease this FNR through class balancing techniques.

We now address feature selection. We use regularization techniques like ridge and lasso which apply L2 and L1 penalties, respectively, to penalize large feature coefficients. Ridge logistic regression decreases coefficients towards zero proportionally to magnitude, so larger coefficients tend to face larger penalties, which is beneficial in handling multicollinearity. Lasso can set certain coefficients which are irrelevant and are mostly noise variables exactly to 0, effectively performing feature selection. Ridge resulted in similar accuracy and FNR which might suggest multicollinearity will not be an issue going forward. Lasso set only 1 variable to 0, Smoker, which makes sense since Smoker had the highest p-value at 0.973, but ultimately it did not produce much model sparsity, which could be because the original creators of the data filtered the 300+ measured variables to the 22 that are presented to us.

3 Machine Learning Methods

Logistic Regression.

We continue our study of logistic regression in more of a modeling context to see how accurately we can classify patients' diabetes diagnosis. We first fit a model on the multiclass data whose results are shown on the left side of Figure 6. Once again, we achieve a high accuracy of 84.8%, but the performance on the

minority classes 1 and 2, displayed by the diagonal elements, is poor. 0 observations were predicted accurately for Class 1 and only 1884 observations were predicted correctly for Class 2.

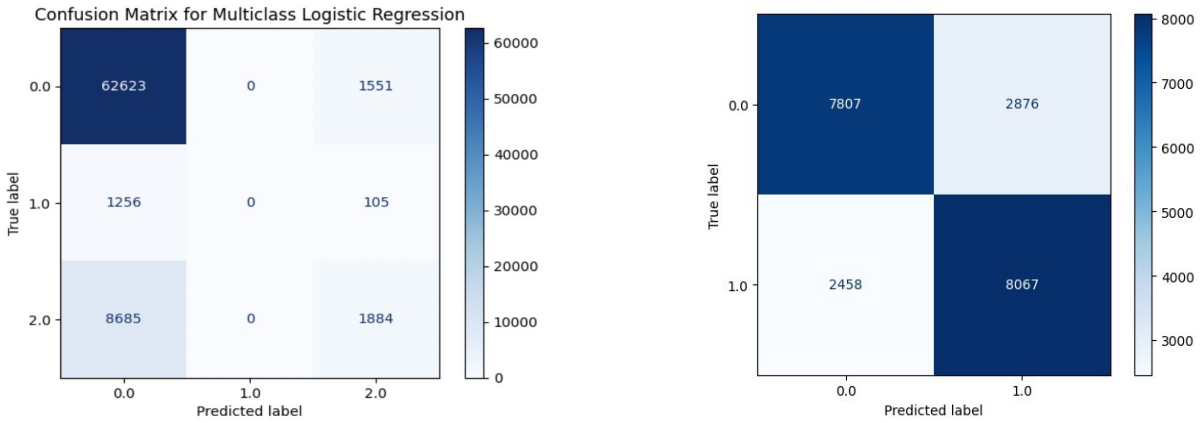


Figure 6

We then refer to the balanced binary setting and fit a logistic regression model to see if the high FNR persists. The results are shown on the right side of Figure 6. Balancing the dataset resulted in a much better FNR, 0.23, as opposed to 0.84 which we got from the binary unbalanced dataset in our EDA section. Even though this accuracy for binary balanced of 75% is lower than the accuracy for binary unbalanced of 86% seen previously, this model is preferable since we have a much lower FNR.

Random Forest.

We selected the tree-based Random Forest model as our second classification approach. Utilizing the Scikitlearn Python library, we implemented the Random Forest Classifier.

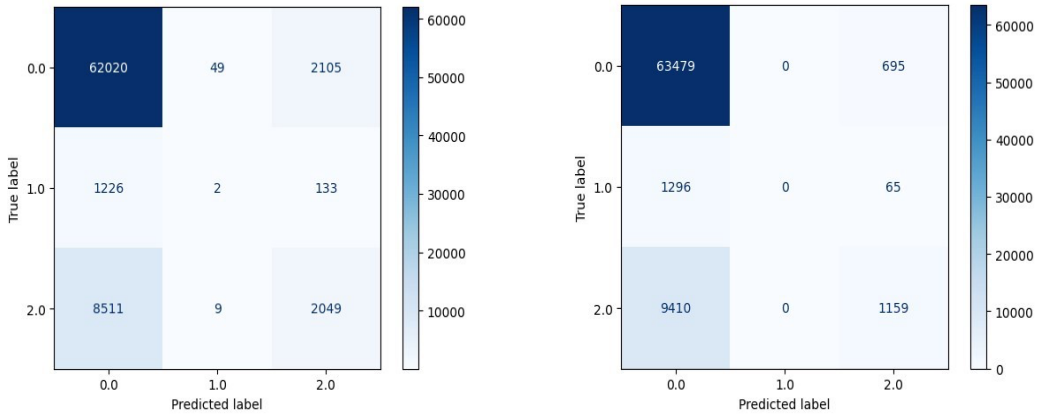


Figure 7

Initially, we established a baseline Random Forest model on the multiclass dataset, setting the number of trees to n_estimators = 100. The baseline model achieved a test accuracy score of 0.842. As observed from the confusion matrix (Figure 7), the model demonstrated high accuracy for class 0 (non-diabetes class) but exhibited a high false positive rate for the diabetes and pre-diabetes classes. After performing 3-fold cross

validation for hyperparameter tuning, the model attained a slightly higher test accuracy of 0.8493. However, the baseline model with default parameters still outperformed this tuned model, as it correctly classified more observations in the minority classes.

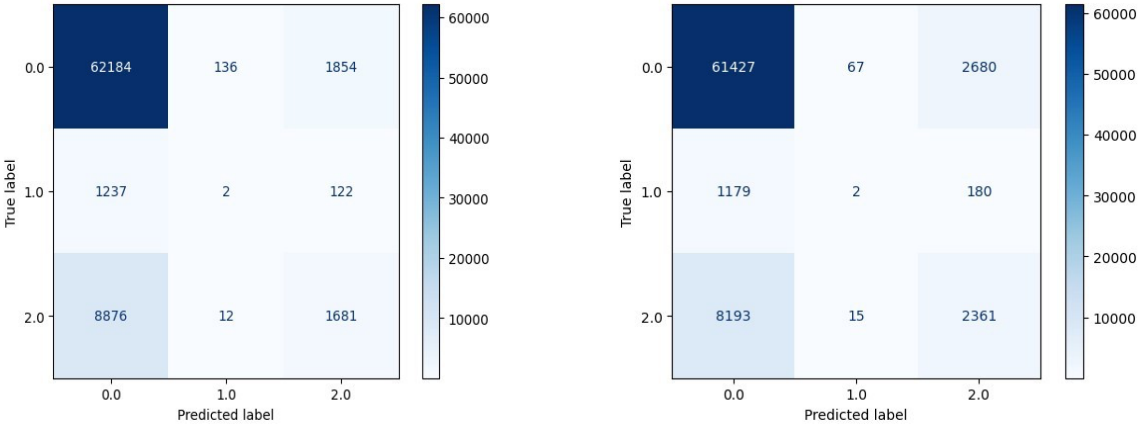


Figure 8

To address the misclassification of minority classes caused by data imbalance, we applied two techniques: class weighting and SMOTE. Class weighting assigns higher penalties to the misclassification of minority classes, while SMOTE (Synthetic Minority Oversampling Technique) generates synthetic samples for minority classes by interpolating the nearest neighbors of each sample. As illustrated by the confusion matrix (Figure 8), the model continued to face challenges in predicting the minority classes. Due to the significant severity of the class imbalance, class weighting alone was insufficient for our dataset. We decided that using SMOTE as we go forward would help address the imbalance severity better. SMOTE, maintaining a high test accuracy of 0.8381, showed slight improvements in predicting class 2; however, the accuracy for class 1 remained unchanged.

Subsequently, we trained another Random Forest model on the balanced binary dataset with default settings, which achieved a test accuracy of 0.741 (Figure 9). Although this accuracy is marginally lower, the model's performance is more consistent across the two classes, significantly reducing the false negative rate for the diabetes class.

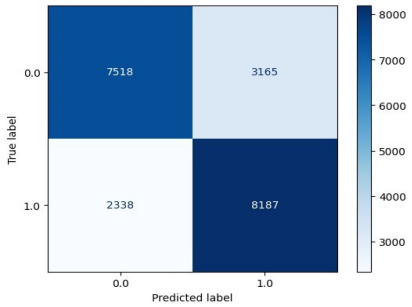


Figure 9

Boosting.

Boosting algorithms build decision trees sequentially where one iteration tries to correct the errors of the previous iteration. This combination of many weak learners provides a strong learner that is robust and highly effective. We choose to explore two boosting techniques: eXtreme Gradient Boosting (XGBoost) and Categorical Boosting (CatBoost). XGBoost uses the gradient of the loss function to fit new trees and corrects previous errors while deploying L1 and L2 penalties to counter overfitting, similar to the aforementioned regularization techniques. The extreme comes from its highly optimized and scalable implementation through parallel and distributed computing. CatBoost is designed to handle categorical data, so we choose to use this to see if it works well with our 20 of 21 categorical features.

XGBoost's baseline performance once again suffered high false negative rates due to class imbalance, so we used the SMOTE resampled training data and gathered the results shown in the left side of Figure 10. This gave us the best performance on Class 2 with about 48% of observations in the class being correctly classified. On the right hand side, we see the results from the binary setting where XGBoost achieved an improved FNR of 20.5%.

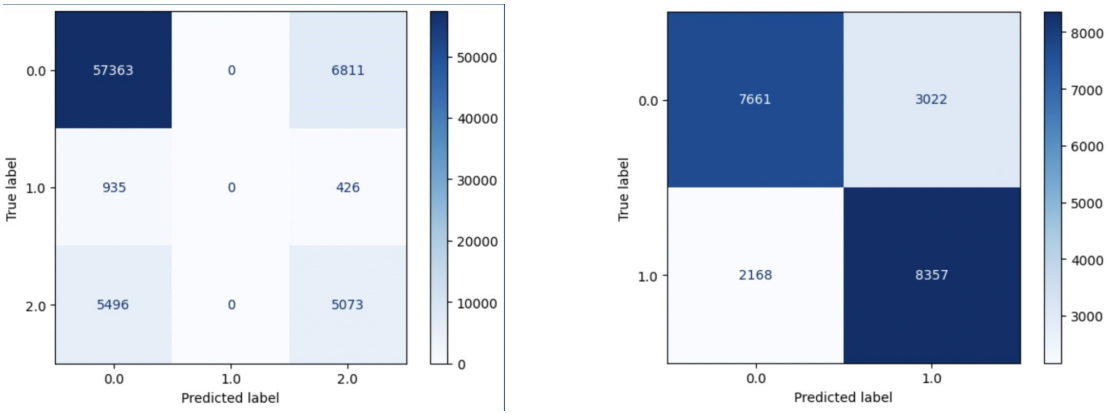


Figure 10

CatBoost experienced similar baseline performance as usual in this study, so we jumped to the balanced multiclass solution with SMOTE and found that CatBoost performed better than Random Forest, but not better than XGBoost.

K-Nearest Neighbors (KNN).

Our fourth approach was the KNN model. We employed the KNeighborsClassifier from the Scikit-learn library. Since SMOTE demonstrated effectiveness in improving class balance, we applied KNN to datasets resampled using SMOTE.

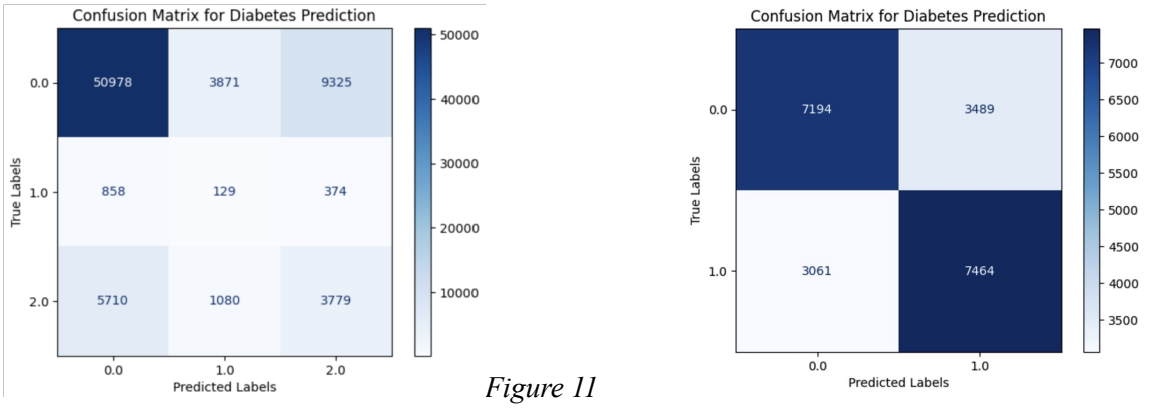


Figure 11

By testing different values of K, the number of nearest neighbors for each data point, we found that K = 2 yielded the highest accuracy score of 0.7212 for the multiclass dataset. Similar to the Random Forest model, class imbalances continued to impact the model's accuracy for the minority classes (classes 1 and 2). For the balanced binary dataset, we selected K = 3, which resulted in an optimal test accuracy score of 0.6938. Although this approach achieved a lower test accuracy overall, it exhibited a reduced FNR.

Support Vector Machine (SVM).

We tested various SVM models to evaluate their effectiveness on the multiclass dataset. These models included the linear SVM, linear SVM with Principal Component Analysis (PCA), linear SVM with PCA and SMOTE (Synthetic Minority Over-sampling Technique), and the SVM with a Radial Basis Function (RBF) kernel. Each of these models was designed to address different aspects of the data, such as class imbalance and feature dimensionality, in hopes of improving classification performance.

We first tried linear SVM. It aims to find the optimal hyperplane with a linear decision boundary to separate the classes. While this model achieved a test accuracy of 84.35%, it classified most instances as no diabetes (class 0). Clearly, the model struggled to differentiate between the other two classes (prediabetes and diabetes). Next, we tried PCA for dimensionality reduction to 2 dimensions before training the linear SVM. The test accuracy was 84.70%. The model still fared poorly in classifying the prediabetes class correctly.

We then used SMOTE alongside PCA, and it became better at classifying the majority of classes 0 and 2 in the right classes, but still did not have much improvement for class 1 (prediabetes).

Then, we used the Radial Basis Function (RBF) kernel, which is non-linear and calculates the similarity between data points based on their distance, with more flexible decision boundaries compared to linear SVMs. With a test accuracy of 83.60%, it still faced the same issue with prediabetes classification.

Finally, we used model selection through cross-validation to fine-tune the hyperparameters and determine the best-performing SVM model. This approach led to a model that used the RBF kernel, with low regularization and the gamma set to auto. The cross-validation accuracy reached 88.92%, while the test accuracy was 83.62%. We can see from its confusion matrix in (left matrix) that it has a lower FNR.

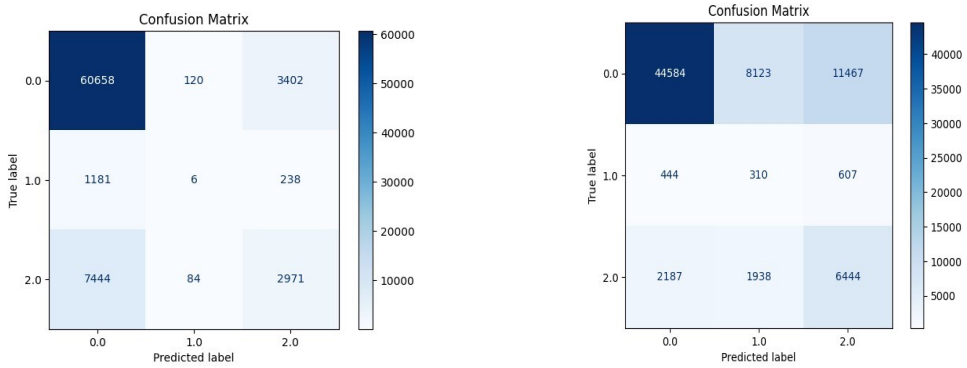
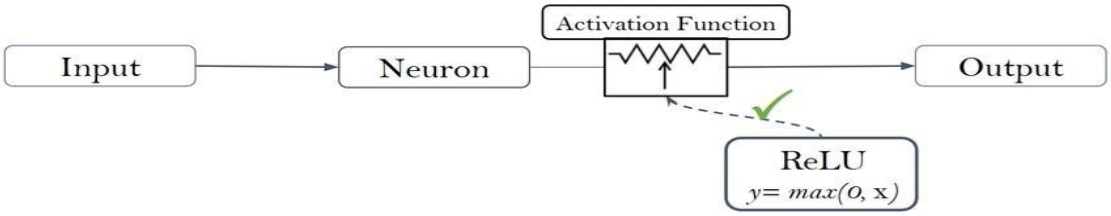


Figure 12

Deep Neural Network (DNN).

We used a DNN model along with SMOTE to address class imbalances. The DNN consists of interconnected layers (input, hidden, and output). Neurons in the network multiply inputs by weights, add a bias, and apply a ReLU (Rectified Linear Unit) activation function to learn complex non-linear patterns. Regularization techniques were used—dropout prevents overfitting by deactivating random neurons, while batch normalization normalizes the output of a layer by adjusting and scaling it, which helps speed up and stabilize training.



During training, the model predicts, compares its predictions to the labels, and adjusts the weights to minimize error over multiple epochs (iterations over the entire training set).

After training the model for 30 epochs with a batch size of 32, we evaluated its performance on the test set. The results showed that the DNN model achieved an accuracy of 0.67 on the multiclass dataset. However, like other models, the accuracy for the minority classes was lower, highlighting the challenge of dealing with imbalanced classes even after applying SMOTE. The confusion matrix and classification report revealed many misclassifications for class 1 and some for class 2.

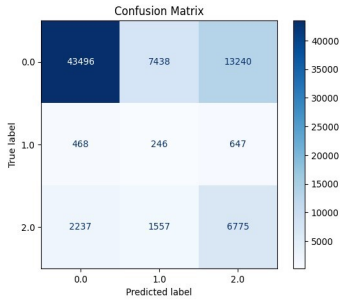


Figure 13

4 Conclusion

Results.

Most models achieved high accuracy, but only a few demonstrated good performance on the minority classes. Class imbalance posed a significant challenge, and only certain models effectively addressed this issue in the multiclass setting. The Linear SVM with PCA and SMOTE was particularly successful, achieving 61% accuracy for Class 2 and 23% for Class 1, while other models often recorded near 0% accuracy for these classes. Similarly, XGBoost showed better performance with 48% correct predictions for Class 2, outperforming many of the other models in this regard.

In contrast, we observed much better performance in the binary setting, with substantially smaller false negative rates (FNR) across the models. Logistic Regression had an FNR of 23%, Random Forest achieved 22%, XGBoost had 20.5%, and CatBoost recorded the lowest at 20.3%. These results highlight the models' improved ability to correctly classify the minority class in a binary classification task.

For future research, it may be beneficial to collect more data from patients with pre-diabetes and diabetes to address the class imbalance and improve model performance across all classes.

References

[1] Centers for Disease Control and Prevention (CDC). 2015. Behavioral Risk Factor Surveillance System. https://www.kaggle.com/datasets/cdc/behavioral-risk-factor-surveillance-system.

[2] Teboul, A. 2021. Diabetes Health Indicators Dataset Notebook. https://www.kaggle.com/code/alexteboul/diabetes-health-indicators-dataset-notebook

[3] Analytics Vidhya. 2021, 5 Techniques to Handle Imbalance Data for a Classification Problem. https://www.analyticsvidhya.com/blog/2021/06/5-techniques-to-handle-imbalanced-data-for-aclassificati on-problem/

[4] GeeksforGeeks. GradientBoosting vs AdaBoost vs XGBoost vs CatBoost vs LightGBM. https://www.geeksforgeeks.org/gradientboosting-vsadaboost-vs-xgboost-vs-catboost-vs-lightgbm/.

[5] Wikipedia. Feedforward Neural Network. https://en.wikipedia.org/wiki/Feedforward_neural_network.

[6] Berejnoi, A. How to Build a Feedforward Neural Network in Python. Medium. https://medium.com/@andresberejnoi/how-to-build-afeedforward-neural-network-in-python-andres-berej noi-a96668c025e5.

[7] PyImageSearch. Implementing Feedforward Neural Networks with Keras and TensorFlow. https://pyimagesearch.com/2021/05/06/implementing-feedforwardneural-networks-with-keras-and-tensor flow/.

[8] Eskandar, S. 2023. Introduction to RBF SVM: A Powerful Machine Learning Algorithm for NonLinear Data. Medium. https://medium.com/@eskandar.sahel/introduction-to-rbf-svm-a-powerfulmachine-learning-algorithm-for -non-linear-data-1d1cfb55a1a

[9] Data Headhunters Academy. Kernel Methods in SVM: Understanding the Mathematical Foundations. https://dataheadhunters.com/academy/kernel-methods-in-svm-understanding-themathematical-foundatio ns/