# Diabetes Prediction of Telephonic Health Survey using XGBoost


| | |
|---|---|
| Name: | **Adam Alvares** |
| Registration No./Roll No.: | 21008 |
| Institute/University Name: | IISER Bhopal |
| Program/Stream: | EECS |
| Problem Release date: | August 17, 2023 |
| Date of Submission: | November 19, 2023 |


## 1 Introduction

The dataset in this project is collected over a telephonic health survey collected by Centers of disease control, USA. The datasets consist of training data, training data targets and testing data. The training data consists of 228312 rows and 21 columns. The testing data consists of 25368 rows and 21 columns. This is a multiclass classification problem, where the number of classes are 3, where class 0 is diabetes, class 1 is prediabetes and class 2 is diabetes. This is an extremely imbalanced dataset where the number of instances in the classes 0,1,2 are 192333,4168,31811 respectively in the training target data as seen in Figure 1.

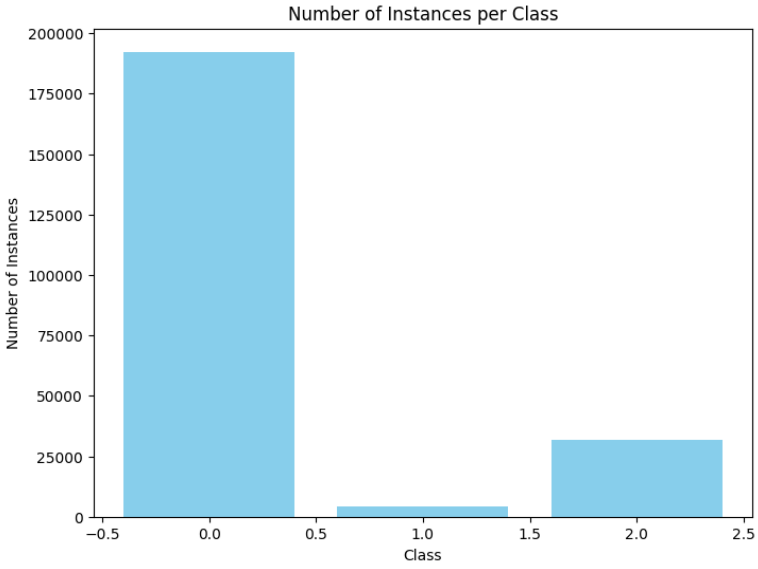


Figure 1: Overview of Data Set

## 2 Methods

The proposed method includes feature selection based on correlation analysis, chi squared test and feature importance. The code generates a heatmap to visualize the correlation matrix of the features amongst themselves and between each feature and the target variable. An Extra Trees Classifier is trained on the dataset to evaluate the importance of each feature. The top 10 features with the highest importances are plotted. With the help of these three methods, some features were dropped from the

dataset. The features dropped were: PhysHlth, PhysActivity, Fruits, AnyHealthcare, NoDocbcCost, Income, Sex, Smoker, HvyAlcoholConsump, MentHlth, DiffWalk. The dataset was then split into training and testing sets for model evaluation. 20 of the data will be used for testing, and the remaining 80 will be used for training. Stratified sampling was used since it helps maintain the same class distribution in both the training and testing sets. The code utilizes the RandomOverSampler from the imbalanced-learn library to address class imbalance in the training data (X train and y train). It also used undersampling and SMOTE oversampling [1] The sampling strategy dictionary specifies the desired number of samples for each class. The oversampling is applied, and the class distribution is printed using Counter. This helps ensure a more balanced representation of classes in the training set. Min Max Scaling was done on the training set, in order to prevent biasness for features with high values. The code uses the XGBoost classifier, a gradient boosting algorithm, for training the data. The model is then trained on the oversampled training data (X train and y train). During training, XGBoost builds a series of decision trees to make predictions, with each subsequent tree aiming to correct errors made by the previous ones. The trained model was evaluated on the test set using metrics such as confusion matrix, precision, recall, and F1 score. Macro-averaged precision and recall were computed, as well as precision, recall, and F1 score for each class. The code sets up hyperparameter grid parameters for tuning the XGBoost classifier. It then uses GridSearchCV to perform a cross-validated grid search with stratified k-fold (5 splits). The training of the grid search was done on the oversampled training data, while the cross validation was done on the unsampled, original test data : X test and y test using eval. The hyperparameters are optimized to maximize the macro-averaged F1 score. After the grid search, the best-performing estimator and its corresponding hyperparameters are obtained. The goal is to identify the hyperparameter values that result in the best model performance on the given test set.

Table 1: Performance Of Different Classifiers Using features using oversampling

| Classifier | Precision | Recall | F-measure |
|---|---|---|---|
| XG Boost | 0.452 | 0.5218 | 0.412 |
| Light GBM | 0.449 | 0.516 | 0.412 |
| Decision Tree | 0.402 | 0.434 | 0.388 |
| Bagging | 0.45 | 0.523 | 0.412 |
| Random Forest | 0.405 | 0.439 | 0.39 |
| Logistic Regression | 0.458 | 0.522 | 0.418 |

Table 2: Performance Of Different Classifiers Using features using undersampling

| Classifier | Precision | Recall | F-measure |
|---|---|---|---|
| XG Boost | 0.451 | 0.518 | 0.418 |
| Light GBM | 0.441 | 0.502 | 0.404 |
| Decision Tree | 0.397 | 0.433 | 0.348 |
| Bagging | 0.450 | 0.520 | 0.418 |
| Random Forest | 0.413 | 0.456 | 0.364 |
| Logistic Regression | 0.453 | 0.517 | 0.425 |

## 3 Experimental Setup and Results

The results for oversampling and undersampling for all the classifiers have been shown in Table 1 for RandomOverSampler and Table 2 for RandomUnderSampler and Table 3 for SMOTE Oversampling. The data being imbalanced, evaluation metrics like precision, recall, f1 score and confusion matrix have been collected for each classifier. Grid search was implemented only on the XGBoost Classifier. The grid search was trained on the oversampled training data, and the grid was cross validated

Table 3: Performance Of Different Classifiers Using features using SMOTE oversampling

| Classifier | Precision | Recall | F-measure |
|---|---|---|---|
| XG Boost | 0.440 | 0.506 | 0.406 |
| Light GBM | 0.439 | 0.510 | 0.409 |
| Decision Tree | 0.398 | 0.432 | 0.379 |
| Bagging | 0.439 | 0.507 | 0.406 |
| Random Forest | 0.402 | 0.436 | 0.384 |
| Logistic Regression | 0.439 | 0.511 | 0.416 |

Table 4: Confusion Matrices of Different Classifiers using RandomOverSampler

| Actual Class | Predicted Class | | |
|---|---|---|---|
| | 0 | 1 | 2 |
| 0 | 23357 | 9657 | 5453 |
| 1 | 190 | 355 | 289 |
| 2 | 874 | 2100 | 3388 |

Extreme Gradient Boost

| Actual Class | Predicted Class | | |
|---|---|---|---|
| | 0 | 1 | 2 |
| 0 | 23466 | 9269 | 5462 |
| 1 | 202 | 324 | 304 |
| 2 | 866 | 2008 | 3488 |

Light GBM

| Actual Class | Predicted Class | | |
|---|---|---|---|
| | 0 | 1 | 2 |
| 0 | 26516 | 5292 | 6659 |
| 1 | 382 | 150 | 302 |
| 2 | 2569 | 1031 | 2762 |

Decision Tree

| Actual Class | Predicted Class | | |
|---|---|---|---|
| | 0 | 1 | 2 |
| 0 | 23410 | 9639 | 5418 |
| 1 | 193 | 355 | 286 |
| 2 | 879 | 2116 | 3367 |

Bagging with XGB

| Actual Class | Predicted Class | | |
|---|---|---|---|
| | 0 | 1 | 2 |
| 0 | 26605 | 5079 | 6783 |
| 1 | 385 | 141 | 308 |
| 2 | 2471 | 972 | 2919 |

Random Forest

| Actual Class | Predicted Class | | |
|---|---|---|---|
| | 0 | 1 | 2 |
| 0 | 24487 | 9451 | 4529 |
| 1 | 215 | 372 | 247 |
| 2 | 995 | 2287 | 3080 |

Logistic Regression

on the original, unsampled cross validation data using the parameter eval_set. The XGBoost hyperparameters that were tuned were learning_rate, max_depth, min_child_weight, n_estimators, gamma. These parameters are essential in preventing overfitting and learning the minority classes. Through the tables for performance of each classifier, it is observed that Decision Tree and Random Forest Classification performed considerably worse than the other classification models. The reason could be mainly because the Decision Tree and default Random Forest were more prone to overfitting of the training data, owing to the creation of multiple copies of class 1 and class 2. It was also observed that SMOTE did not perform as well as RandomOverSampler, since the relationship between the features and targets were not captured well by SMOTE.

## 4 Results and Discussion

It is clear that the dataset is heavilty imbalanced. Random oversampling helps solve this issue by creating copies in the minority class, balancing the class distribution and preventing the model from being biased towards the majority class. XGBoost combines weak learners to produce a strong predictive model. XGBoost also provides hyperparameters for regularization, such as max_depth and gamma, allowing control over model complexity. This can be beneficial in preventing overfitting, especially

when dealing with oversampled data. Those are the clear merits, however, in the framework, the class 1 training data was oversampled from 3303 instances to 190000 instances, and the class 2 training data from 25506 instances to 170000 instances. This means that the goal to increase precision, recall and f1 score of the cross validation data is counterproductive since the training data is now overfitting to the model due to the enormous amount of class 1 and class 2 samples created . Table 4, shows the experimental results of all the classification models used in the form of a confusion matrix. As we can see, the XGBoost model is predicting class 0 well, however the model is falsely predicting class 1 alot. While the recall of class 1 is acceptable, its precision is low. Class 2 is being predicted decently.

## 5 Conclusion

Future research in this framework may focus on developing oversampling techniques that dynamically adapt to changes and features in the dataset, techniques that can automate and adjust sampling strategies and weights of classes based on the dataset characteristics could be further researched and explored. Also, new methods to create new synthetic data/feature vectors that is biologically suitable to match a particular class label can be explored, since SMOTE does not do an acceptable job in oversampling.

## References

[1] Francisco Mesquita, José Maurício, and Gonçalo Marques. Oversampling techniques for diabetes classification: a comparative study. In *2021 International Conference on e-Health and Bioengineering (EHB)*, pages 1–6, 2021.