**Technical Progress Report**

This is a technical progress report on our Diabetes Management project. In this phase, we tried to develop a classification model and considering feedback and suggestions, we tried to gather user data automatically through wearable device (Fitbit) instead of asking the users to enter information.

**Data**

We performed EDA on our diabetes\_012\_health\_indicators\_BRFSS2015 dataset and observed its quality. Since all values are numeric and there are no missing values, we did not have to perform any preprocessing.

**Classification**

The first step is to split the data into train, validation, and test set. For this, we used train\_test\_split module from sklearn library.

We split the dataset as 80% train, 10% validation and 10% test set.

A computer screen with text

Description automatically generated

Figure : Train test split

Above segment of code first separates the target variable (y) from the independent variables (X). The next two lines perform the split. First it splits into train set and a temp set in 80:20 ratio. The next split results in 10% validation set and 10% test set.

A black background with white text

Description automatically generated

Figure : Number of data points in each set

Figure 2 shows the shape of each set obtained after train validation and test split.

We used the stratify attribute, as seen in figure 1, to distribute the labels evenly in each set. Since we are using a part of data rather than the entire data for training, we wanted to make sure that the distribution of label is preserved in the training set. Our motive was to make the train set as much similar as possible to the original data.

A pie chart with numbers and a red triangle

Description automatically generatedA blue and red pie chart with numbers and a red triangle

Description automatically generatedA blue and red pie chart with text

Description automatically generated

Figure : Label distribution

The above pie charts are generated using plotly library; and help us make sure that the distribution of labels is preserved.

This train set was used for training our classifiers. Our dataset has a huge class imbalance issue and upon research, we found certain methods that are effective for such cases. We tried Random Forest, XGBoost and LightGBM. We also tried resampling techniques. We explain each method further in more details.

**1. Random Forest classifier**

We used grid search cross validation in 5 folds across a specified parameter grid to determine appropriate hyperparameters for the classifier. The scoring parameter used was weighted roc auc score. Our findings are as follows:

A screenshot of a computer code

Description automatically generated

Figure : Grid search for Random Forest classifier

We then fit training data with the model with best parameters. After training, we first check the important features using feature\_importances\_ method. Following pie chart gives an idea about important features according to this random forest classifier:

A colorful pie chart with text

Description automatically generated

Figure : Random Forest Feature importance

Now we check model performance on validation set.

A screenshot of a computer program

Description automatically generated

Figure : Confusion matrix validation set

A screenshot of a computer

Description automatically generated

Figure : Classification report validation set

The confusion matrix and classification report suggest that the model was only good in predicting non diabetic cases.

**2. XGBoost classifier**

After Random Forest, we tried XGBoost classifier. We similarly performed grid search look for appropriate parameters based on best weighted roc auc score. Once we obtain best parameters, we check feature importances and performance. Following are the results:

A screenshot of a computer program

Description automatically generated

Figure : Grid search on XGBclassifier

Best features selected from param grid are 'booster': 'gbtree', 'learning\_rate': 0.1, 'max\_depth': 2.

A colorful pie chart with numbers

Description automatically generated

Figure : Feature importances by XGB classifier

The feature importance obtained from XGBoost classifier makes more sense in the real world, compared to that of Random Forest classifier. This is just for analysis. We are planning to take help of shap values to determine important features in.

A screenshot of a computer program

Description automatically generated

Figure : Confusion matrix and classification report

Performance analysis shows that XGBoost is also not that good for our 3 class classification problem.

**3. LightGBM**

Since we are dealing with significant class imbalance, we relied on boosting method. As XGB model failed, we tried another booster model, LightGBM. We get the best model based on grid search for the best weighted roc auc score. We even tried adding custom weights to this model to see if it helps with class imbalance issue.

This model does not have feature importance parameter, so we directly move on to performance analysis.

A computer screen with colorful text

Description automatically generated

Figure : LightGBM training with custom weights

A screenshot of a computer program

Description automatically generated

Figure : LightGBM performance

The above figure shows that LightGBM is also not very effective in dealing with class imbalance.

**4. Resampling and Random Forest**

Looking at all the results we decided to try out the same methods again using resampling techniques. For this, the imbalanced\_learn module is used. We tried under sampling class 0 and over sampling class 1, and then checked performance of the best random forest model with best parameters that is now trained on resampled training dataset.

A screenshot of a computer program

Description automatically generated

Figure : Random Forest with resampled training data

As expected, resampling did help in achieving significant improvement with a weighted roc auc score of 0.89. The only concern here is the low recall score for class 0.

**5. Resampling and XGBoost**

We also checked how the resampled training set changes performance of XGBoost model.

A screenshot of a computer program

Description automatically generated

Figure : XGBoost performance on resampled training set

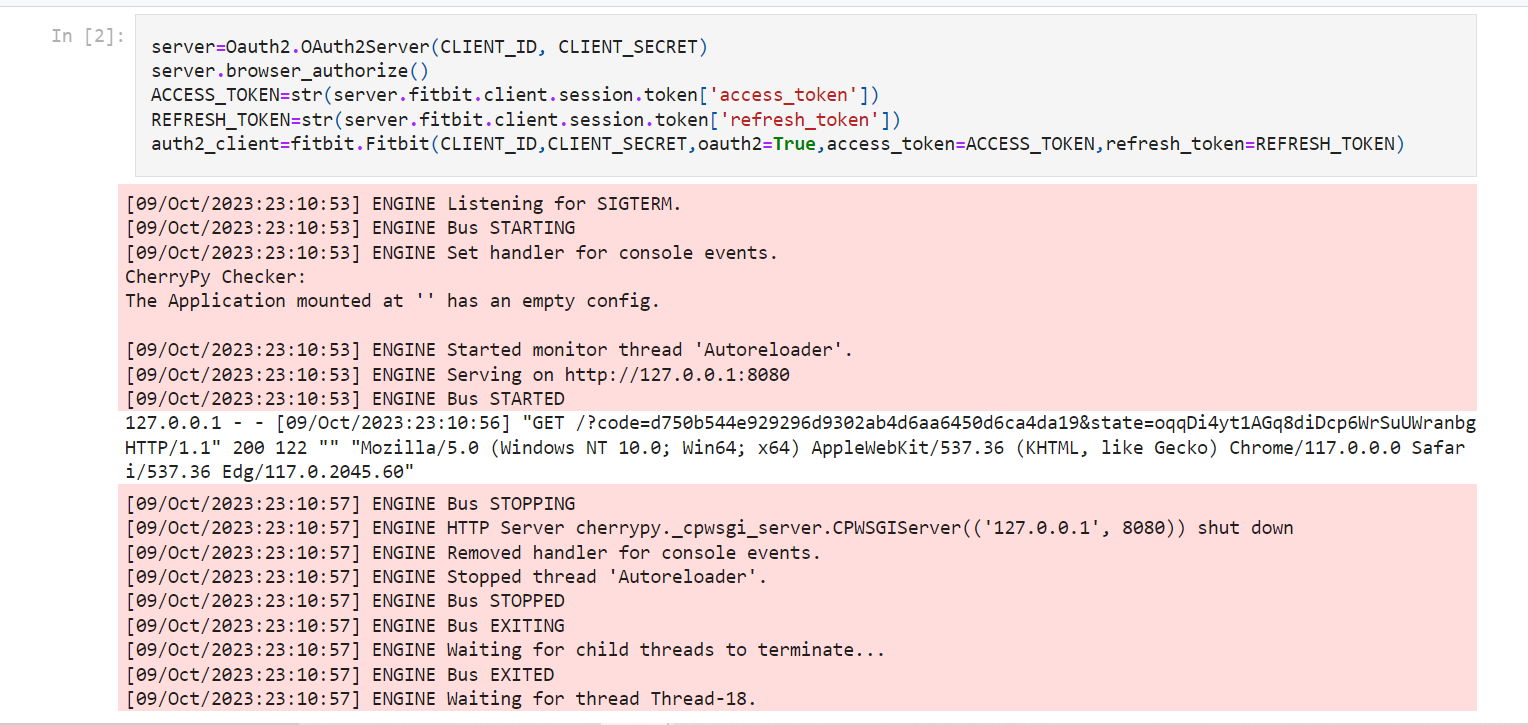
We observed that random forest performed much better.

We will further try hyperparameter tuning and some other methods like support vector machines or neural networks, in order to attain better performance for our classifier model.

FitBit:

The Fitbit API allows developers to access health and fitness data such as movement, sleep, heart rate from Fitbit devices. Access to this data is granted by creating a registered application through a Fitbit developer account, following OAuth 2.0 authorization. The API provides specific endpoints for different types of data, and the responses are in JSON format.

**Authorization to access the fitbit data:**



**Requesting the data from Fitbit:**



**Sample Data extracted from Fitbit:**

