Diabetes classifier

Gilberto Juárez Rangel
(Dated: May 4, 2024)

In this report we utilize the ‘Diabetes’ dataset to classify weather someone has or not diabetes by using Machine Learning classification models such as, k-nearest neighbors (KNN), support vector machines (SVM), decision trees and logistic regression. Before applying any model the data is first cleaned, transformed and balanced (downsampling and upsampling) to ensure a better performance. Finally we compare the different methods and their results to choose the best model.

I. INTRODUCTION

Diabetes is a chronic medical condition characterized by elevated levels of blood glucose (sugar) resulting from defects in insulin production, insulin action, or both. Insulin, a hormone produced by the pancreas, helps regulate blood sugar levels by facilitating the uptake of glucose from the bloodstream into cells to be used for energy. When the body either doesn’t produce enough insulin or becomes resistant to its effects, blood sugar levels can rise to unhealthy levels, leading to various complications.

Identifying diabetes early is crucial, otherwise it can lead to serious complications affecting various organs and systems in the body. Also an early diagnosis and appropriate management of diabetes can help individuals maintain better control over their blood sugar levels and reduce the risk of complications. In this project we intend to help identify diabetes faster by using some data that can be accessed easily with the right home equipment.

Since we are treating with patients we care more about recall than precision. For our case recall means how many of the diabetic patients were predicted diabetic and precision would be how many of the predicted diabetic patients were diabetic. Of course we prefer both recall and precision to be high as possible, otherwise we could just predict every patient to be diabetic which leads to an inefficient model.

For this we will be using the ‘Diabetes’ dataset which includes 8 features (6 numerical and 2 categorical) and 1 target. The target feature is binary, indicating weather the patient has diabetes (1) or not (0). The names of the features and target are pretty much straight forward so there is no need of explanation for each, except for ‘HbA1c_level’ which is the average blood glucose (sugar) levels for the last two to three months. Next you can find the feature names and their type of data.

- gender (Categorical)
- age (Numerical)
- hypertension (Binary)
- heart_disease (Binary)
- smoking_history (Categorical)
- bmi (Numerical)
- HbA1c_level (Numerical)
- blood_glucose_level (Numerical)

II. DATA EXPLORATION, CLEANING, FEATURING AND PREPROCESSING

A. Exploration

To understand the data we plotted the histograms of the numerical and binary data (including target) as shown in figure (). As we can see, there is a huge unbalance in the target, having few information of the patients with diabetes (This will be solved later using downsample and upsample techinques).

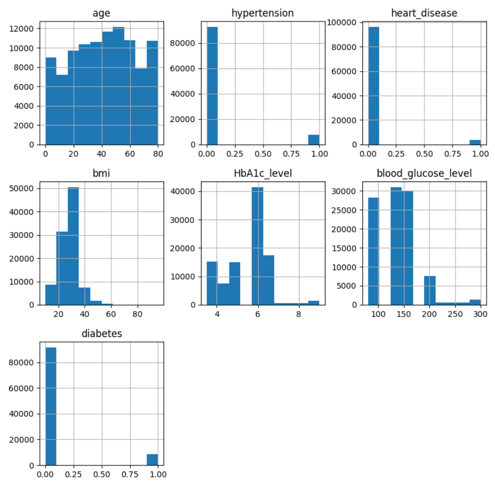


FIG. 1: Histogram of numerical and binary features and target of the ‘Diabetes’ dataset.

Since we have categorical data, we used the `.unique()` function of pandas and got the following categories for the ‘gender’ and ‘smoking_history’ features:

- gender: [’Female’, ’Male’, ’Other’]

- smoking_history: ['never', 'No Info', 'current', 'former', 'ever', 'not current']

Getting the proportions of the gender categories, which had 58.552% female, 41.43% male and 0.018% other. Since we have almost no data for the other category, there was no point on keeping that data and we proceed to remove it in the cleaning steps. For the proportions of the smoking_history categories the fewest was the 'ever' category with 4% which are 4 thousand rows of data that can be still enough for some models.

B. Cleaning

In order to clean the data we first looked for any missing values (i.e. null or 'nan' data). For this dataset we found none but, there was a category in the 'smoking_history' feature that was 'No Info' which for us is equivalent as a nan or null value and is equivalent to a 36% of the original dataset. Since there is a huge unbalance between the diabetic and non-diabetic patients, we must check that eliminating this information won't remove an important amount of the diabetic patients. After looking at it, we found out that only a 17% of the diabetic patients where part of the 'No Info' category. After analyzing this information, the 'No Info' category was removed from the dataset.

After that, we proceeded to search for nonsense data (wrong measurements). While searching information about the features of the dataset, we found out that a bmi lower than 11 can be fatal for female and lower than 13 can be fatal for male. So for the fatal values of bmi, we removed them since they could have been wrong measurements.

Finally, we checked for outliers and how much impact they could have on the dataset. Since we have unbalanced targets, we got the amount of diabetic patients that would be removed by the outliers. Even for 3 standard deviations of threshold, an important amount of diabetic patients would be removed living us with a bare minimum of relevant data.

C. Feature engineering

As feature engineering we only transformed the categorical variables. For the gender variables we turned it a binary variable since we removed the 'other' category, having 0 as female and 1 as male. For the smoking_history feature we applied the one hot encoding technique resulting in a total of 12 features.

D. Preprocessing

Since we have an unbalanced data we used both upsampling and downsampling techniques and also keep the unbalanced dataset in order to compare results. Since we want to optimize recall over precision (because we are treating with medical issues), we could expect better results by downsampling the dataset rather than upsampling.

Some ML models are sensitive to feature scaling. For simplicity we will use the `StandardScaler()` function of sklearn which transforms the data such that they have mean of 0 and standard deviation of 1. Since binary features are categories, we only applied this into the numerical data.

III. CLASSIFICATION MODELS RESULTS

In this section we will show the results of KNN, SVM, Decision Trees and Logistic Regression for each dataset, unbalanced, upsampled, and downsampled. For the unbalanced case we used a stratified splitting to avoid discrepancies in the training and test set. For the upsampled case, we applied the upsampling after splitting the training and test sets to avoid data leakage. This wasn't necessary for the downsample case since in downsampling we only cutoff data randomly of the majority class. For all cases we used cross-validation to calculate precision and recall without any bias of the train-test splitting.

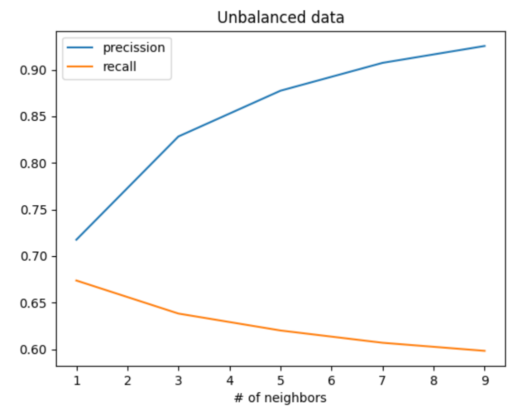
A. KNN

For the KNN model, precision and recall were obtained for a different number of neighbors. The results can be seen in figure 2 for each case

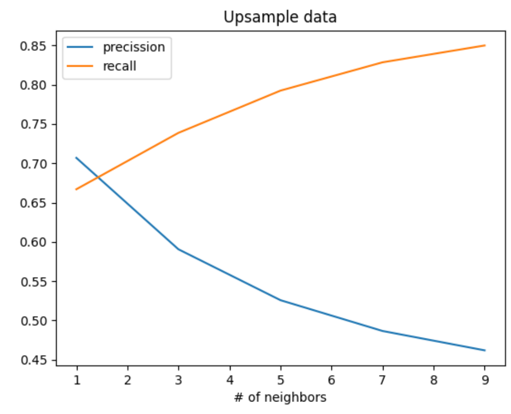
For the unbalanced class there is a decreasing recall and an increasing precision for increasing neighbors. Since we have more than 10 times non-diabetic patients than diabetic patients is natural to see this behavior when increasing the number of neighbors.

In the upsampled class we see an opposite behavior than in the unbalanced class, this can be a consequence of the upsampling which increases the number of diabetic patients based on the available information. This usually removes the small regions where non-diabetic patients are inside a bigger region of diabetic patients, this is illustrated by an example in figure 3.

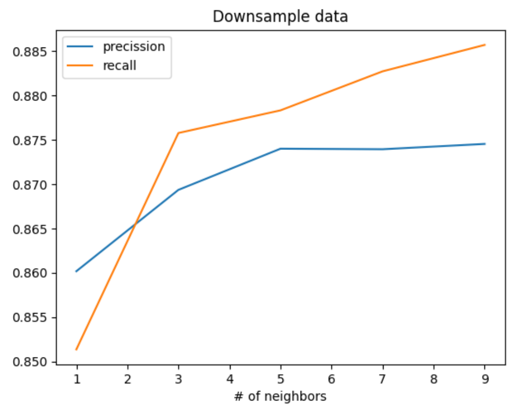
On the other side, for downsampling there is an increase in both recall and precision as the number of neighbors increase. In this case downsampling prevents wrong classification of non-diabetic patients caused just by a big discrepancy between the targets. Also downsampling is



(a)



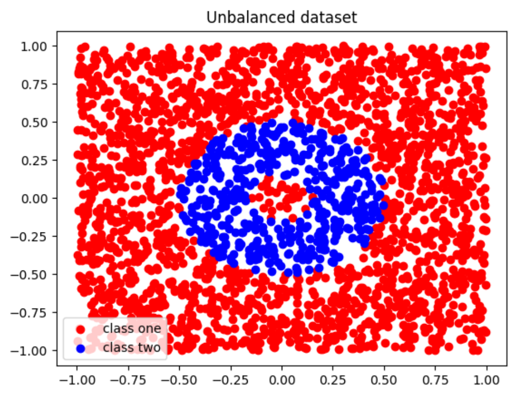
(b)



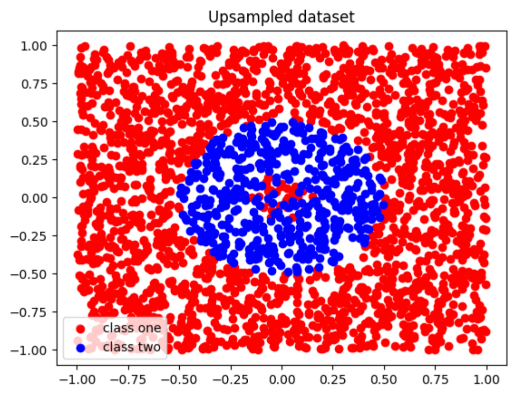
(c)

FIG. 2: Precision and recall for different number of neighbors in the KNN model.

not biased by synthetic data created by ourselves giving a better result.



(a)



(b)

FIG. 3: Demonstration of how upsampling can reduce precision.

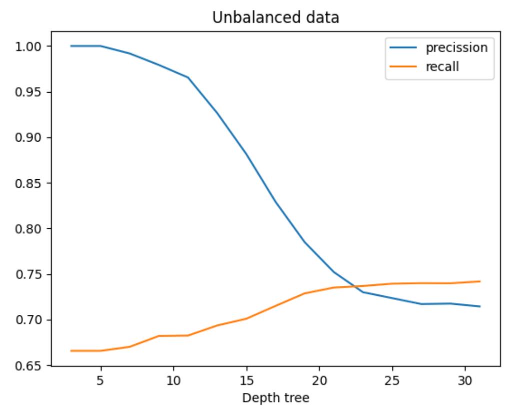
B. SVM

For the SVM model we used a linear support vector classifier with maximum iterations of 10,000. The results can be seen in table I.

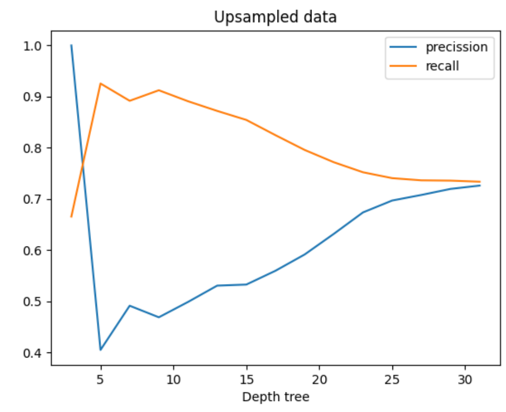
	Precision	Recall
Unbalanced	0.888	0.618
Upsampled	0.889	0.618
Downsampled	0.882	0.871

TABLE I: Precision and recall for each case with a linear support vector classifier.

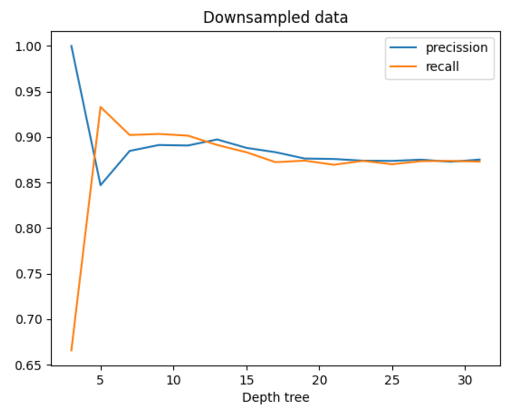
Applying upsampling shows almost no improvement as the unbalanced model, WHEN using SVM, creating synthetic data can just add samples on the same side divided by the hyperplane giving no advantage. On the otherside, downsampling balances the distribution of the majority class which allows a better splitting with a hyperplane increasing the recall.



(a)



(b)



(c)

FIG. 4: Precision and recall for different max depths in the Decision Tree model.

C. Decision Tree

For the decision tree model, precision and recall were obtained for a different max depth. The results can be seen in figure 4 for each case.

For the unbalanced class we see how precision decreases while regression increases when we increase the depth of the tree. Around a depth of 27 the system starts to stabilize and almost no change is observed in precision nor recall.

In the case of upsampled data regression increases and precision decreases rapidly in the first depths. Then both recall and precision go the other way around starting to stabilize at a depth of 37 having both a recall and precision around 0.75.

Finally in the downsampled dataset we get the opposite behavior than in the upsampled data. Nevertheless, this model has a similar precision and recall after a depth of 7, converging at a depth of 17 with a precision and recall around 0.87.

D. Logistic regression model

For the Logistic regression model we used a simple Logistic regression classifier. The results can be seen in table II.

	Precision	Recall
Unabalanced	0.860	0.639
Upsampled	0.880	0.868
Downsampled	0.878	0.867

TABLE II: Precision and recall for each case with a linear support vector classifier.

For both upsampled and downsampled models we get almost the same results, a small increase in the precision and more notable increase in the recall than in the unbalanced model.

IV. FINDING AND INSIGHTS

From the KNN models we have better recall and precision in the downsampled dataset. For this case, we can pick 5 nearest neighbors, this allows to have less distance calculations for further cases and has a good recall of 0.878 and precision of 0.873.

In the decision tree model there is now doubt that again the downsampled dataset has better results, from which we would take a maximum depth of 9. This decision is taken since we get a recall of 0.9 and a precision of 0.88. Also is important to take into account that this model has a better interpretation than anyone, with only a depth of 9.

In the SVM case without doubt we can take the downsampled case which has a better recall than the other ones. For logistic regression we could take the upsampled case which has a better precision and recall, nevertheless, we can trust more in the downsampled case that has almost the same recall and precision.

By grouping all the models in table () it is undoubtedly that the best model to choose is the decision tree. This model not only has the best precision and recall values but it can also be easily interpretable for the users and it might be the faster from all since we only take binary decisions given the data.

V. NEXT STEPS

In this project the decision tree model with a downsampled dataset achieved robust performance with high precision and recall without overfitting. This might be improved by identifying which features are not so active or useful and remove them from the dataset. Also creating new features with the known ones could improve the model, in my case I decided not to do this since the results obtained where satisfactory. Finally more data of diabetic patients could be very useful since we hade a relaiton of almost 11:1 between non-diabetic and diabetic patients.