Information Technology

Project Report - CNIT623-MLT Fall 2022

Impact of preprocessing on imbalanced datasets

K. Meher Hasanth,
Sahiti Kasim,
Isham Mahajan

supervised by
Professor Baijian Yang

Abstract

Preprocessing in imbalanced datasets is a challenging topic to solve. Imbalanced datasets often contain a disproportionate number of examples belonging to one class compared to the other classes. This may lead to the model not predicting the minority class with accuracy, which is a major problem in problems such as default prediction, in which case false positives (predicting that the customer will default, but they do not) are acceptable, but false negatives (predicting that the customer will not default, but they do) are not. Imbalanced datasets can also make it difficult to assess the performance of models trained, since simply predicting the majority class most of the time will result in an imbalanced high accuracy. Certain machine learning models such as KNNs can also perform disproportionately worse on imbalanced datasets. We aim to clarify the impact of certain preprocessing techniques on the training time, and performance metrics for a traditional non-parallel model, XGBoost, and a neural network with a 512x6 neuron architecture. More specifically, with XGBoost, we aim first to test the impact of filling null values with the column mean, median, and mode. This is made possible by the fact that XGBoost can handle null values, making it possible to get a control group. We then test the impact of categorical to numeric encoding techniques, and dimensionality reduction techniques. We then use random oversampling and undersampling and record their impact on training time and accuracy of the model. With neural networks, more usable data, and undersampling, we were able to achieve an accuracy of 89.02%.

Contents

1 Introduction

Default prediction is a problem of two-class classification. This issue is unbalanced by nature because more credit applications are rejected than approved. The underlying workings of the credit assignment, however, mean that the classification costs are different for the two groups. For instance, a financial institution can lose a potential customer if a good application is rejected. On the other hand, if a bad applicant is given credit, the financial institution suffers financial losses as well as potential costs related to any legal measures it may need to take to reclaim its investment.

Machine learning is now used for a variety of purposes beyond automation, including future prediction, customer loyalty analysis, standard recovery rate information, learning optimal coverage rates, investor sentiment modeling, fraud detection, stock price detection, client maintenance, programmed credit endorsement, extortion detection, executive showcasing, and executive risk management. Banks store a lot of past data on their customers. This data can be used to establish and maintain a clear relationship and linkage with customers in order to target them specifically for financial or specialized product offers.

2 Problem Statement

What is the impact of various preprocessing techniques on imbalanced datasets?

3 Related Work

This section focuses on the works that are most relevant to learning classifiers or preprocessing techniques with regard to the characteristics of unbalanced data.

A thorough analysis was conducted by Van Hulse et al. using 35 real-world datasets, 11 classifiers, and 7 preprocessing techniques in [1]. According to the imbalance ratio, they divided their datasets into 4 groups, and within each group, they compared the learning algorithms. For data with the most extreme imbalance ratio (5%), random undersampling performed better than other methods, according to the authors. In contrast to earlier research, they asserted that less complex random re-sampling techniques frequently outperformed more complex informed re-sampling techniques. They also observed that algorithms react differently to various preprocessing techniques and that it depends on the assessment measures after doing several trial setups (e.g., G-mean or F-measure showed higher improvements than AUC).

AUC was used to assess 7 learning algorithms across 20 real-world datasets in a research by Batista et al. [2] on the influence of the imbalance ratio. The scientists found that when the minority class constituted 10% or fewer of the data in all datasets, a substantial performance loss began to occur. Compared to other classifiers, SVM was less impacted by varying the class imbalance ratio for all except the most unbalanced distributions. Then, they examined the effectiveness of SMOTE and random oversampling as two preprocessing techniques, concluding that these techniques typically could not boost performance by more than 30%.

Another investigation into the influence of the imbalance ratio was conducted by Batista et al. [2], who used the AUC metric to assess 7 learning algorithms across 20 real-world datasets. The scientists found that when the minority class constituted 10% of the data or fewer, the performance loss started to become noticeable after averaging the findings across all datasets. With the exception of the most unbalanced distributions, SVM was less impacted by adjusting the class imbalance ratio than other classifiers. They then evaluated the effectiveness of two preprocessing techniques, random oversampling and SMOTE, and came to the conclusion that these techniques often could not enhance performance by more than 30%.

4 Exploratory Data Analysis

Exploratory data analysis, or EDA, is a strategy for analyzing datasets to highlight their key characteristics, frequently using pragmatic techniques. EDA is used to gather condensed insights about data to give us in-depth knowledge about it, which we can employ in the work of modeling in the future. Both graphical and non-graphical insights are possible. But graphical representations of data are valued more since they provide us with more details than non-graphical techniques. To determine the essential details of the data from a spreadsheet's worth of data or an attribute value takes time and effort. Seeing unsightly figures and trying to draw conclusions from them may also be frustrating and overwhelming as shown in figure 2. EDA methods have been created as a help in this circumstance.

During this stage, we first checked the dataset's various categorical variables. Each customer's aggregated profile features at each statement date are contained in the dataset. Features fall into the following broad groups after being anonymised and normalized as shown in figure 1:

- D_* = Delinquency variables (bad or criminal behavior, especially among young people)

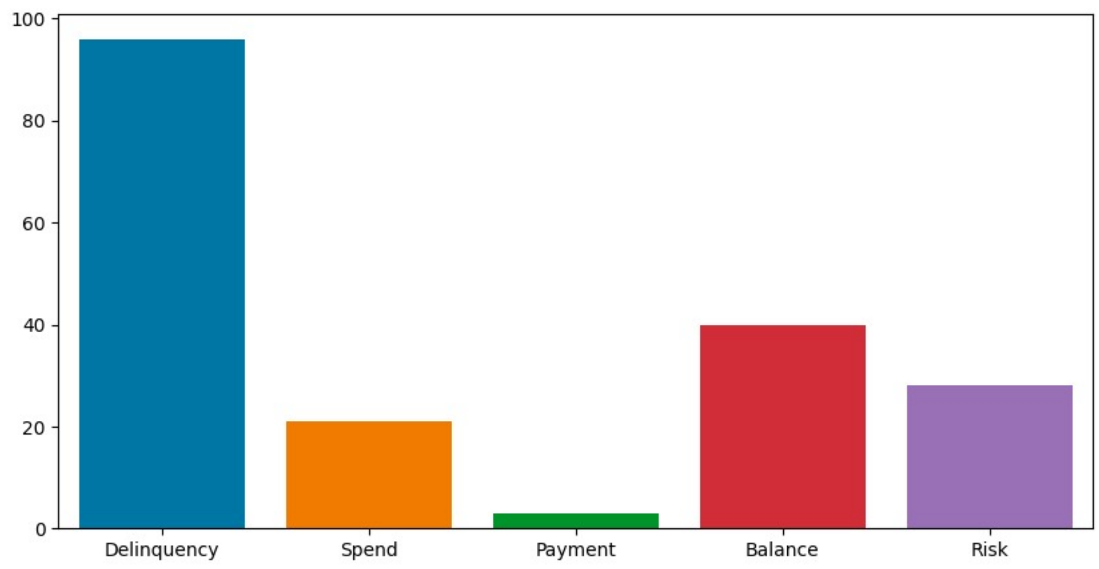


Figure 1: Aggregated customer profile feature categories

- S_* = Spend variables
- P_* = Payment variables
- B_* = Balance variables
- R_* = Risk variables

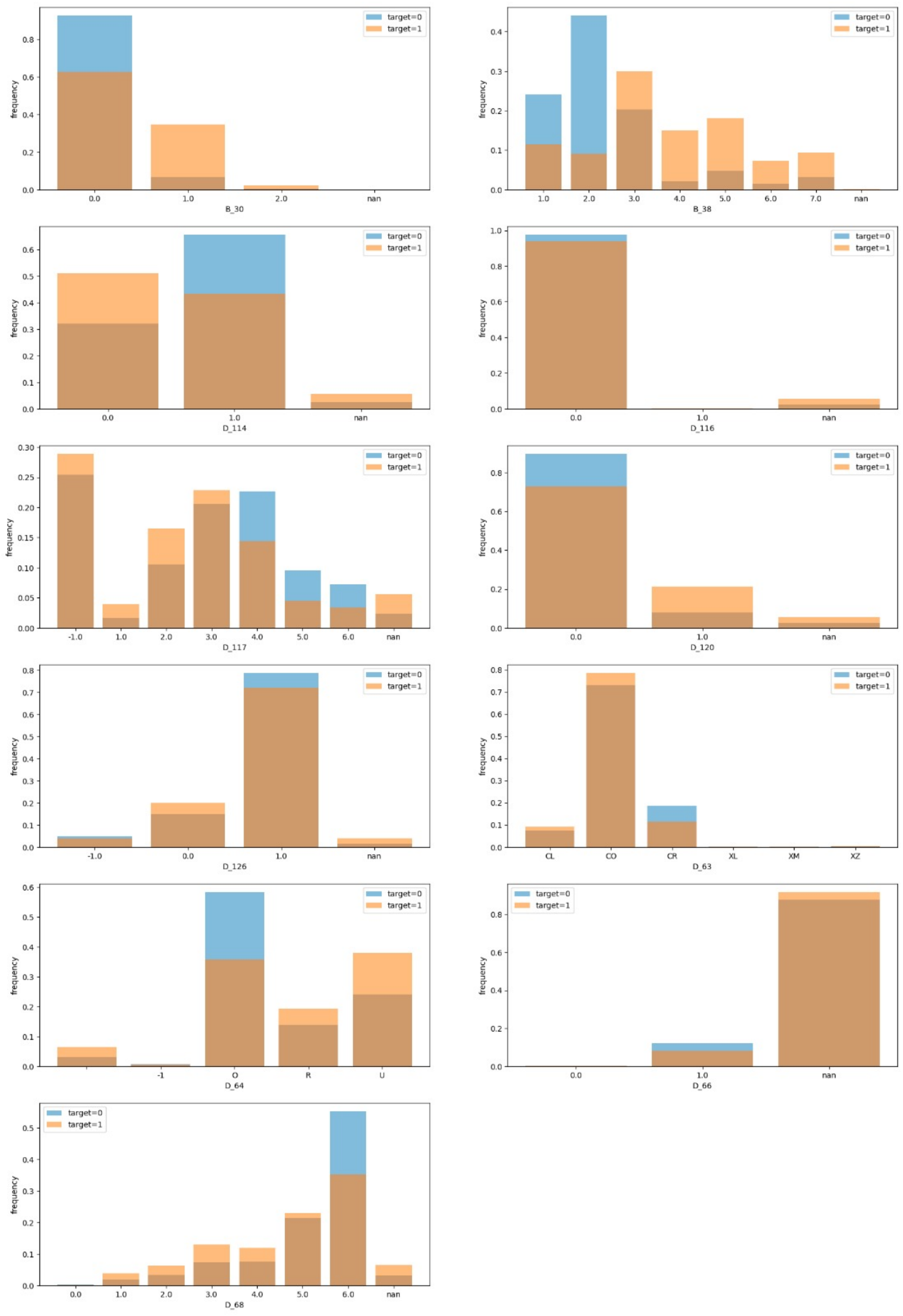
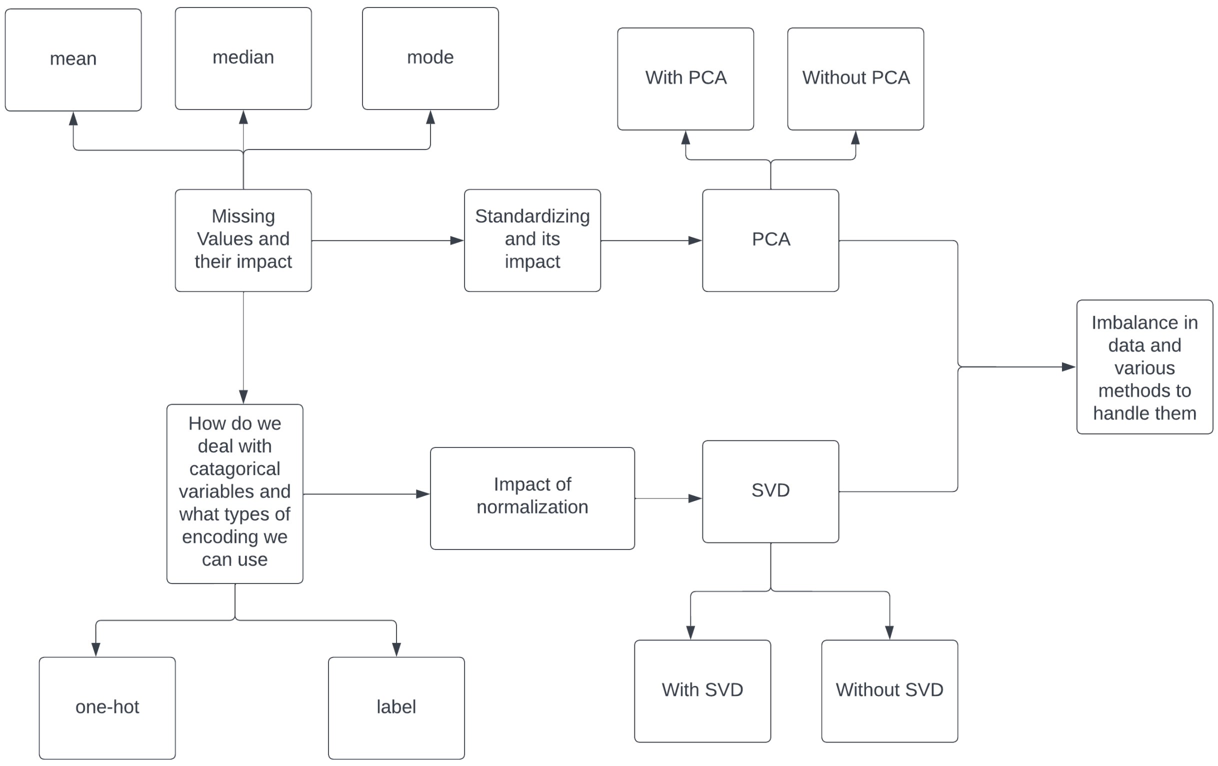


Figure 2: Proportions of features against the target values

5 Methodology


mean
median
mode
With PCA
Without PCA
Missing Values and their impact
Standardizing and its impact
PCA
Imbalance in data and various methods to handle them
How do we deal with catagorical variables and what types of encoding we can use
Impact of normalization
SVD
With SVD
Without SVD
one-hot
label

5.1 Data Preprocessing

The transformations we apply to our data before feeding it to the algorithm are referred to as preprocessing. Data preprocessing is a method for converting unclean data into a clean data set, i.e anytime data is received from various sources, it is collected in raw format, which makes analysis is impossible. It involves handling incompleteness, inconstancy, and errors in data.

Need of Data Preprocessing:

- **Drop Duplicate Rows**: We observed there were few duplicate data points in data and we removed them.
- **Drop Null Valued Rows**: We observed there were few null valued rows and we removed them.
- **Drop Redundant Columns**: We observed there were a few columns which had more than 80
- **Replacing Null or None Values**: We observed there were a lot of columns with NaN

values so we used different methods to replace them. We used mean, median and mode to replace the null values and carried forward with the highest accuracy achieved from them.

- **Normalization**: Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1.

5.2 Label Encoding

Label encoding is the process of making labels machine-readable by transforming them into a numeric form. Then, machine learning algorithms can decide more effectively how those labels should be used. It is a significant supervised learning preprocessing step for the structured dataset.

Since in label encoding we are assigning a numerical value to these labels for example Male and Female mapped to 0 and 1. But this can add bias in our model as it will start giving higher preference to the Female parameter as 1>0 and ideally both labels are equally important in the dataset. To deal with this issue we will use One Hot Encoding technique.

5.3 One Hot Encoding

One hot encoding creates new (binary) columns, indicating the presence of each possible value from the original data. This creates a binary column for each category and returns a sparse matrix or dense array. In this way, it gives equal importance to all parameters which is not the case in label encoding.

5.4 Class Imbalance

It was found while analyzing data that this dataset has only 2 classes 0 & 1, making it as a binary problem the dataset is imbalanced. This results in models to have poor predictive performance, specifically for the minority class. This is a problem because typically, the minority class is more important and therefore the problem is more sensitive to classification errors for the minority class than the majority class. In our dataset, we have 1660993 values of ‘0’ and 551587 values of ‘1’. which makes the rows with ‘0’ as majority class (75.07%) and rows with ‘1’ as minority class (24.93%). The figure 3 shows the distribution of the training dataset.

5.4.1 Oversampling

In oversampling, it randomly selects a few samples from the minority class and duplicates them in the training dataset prior to fitting a model. Although these samples don’t add any new

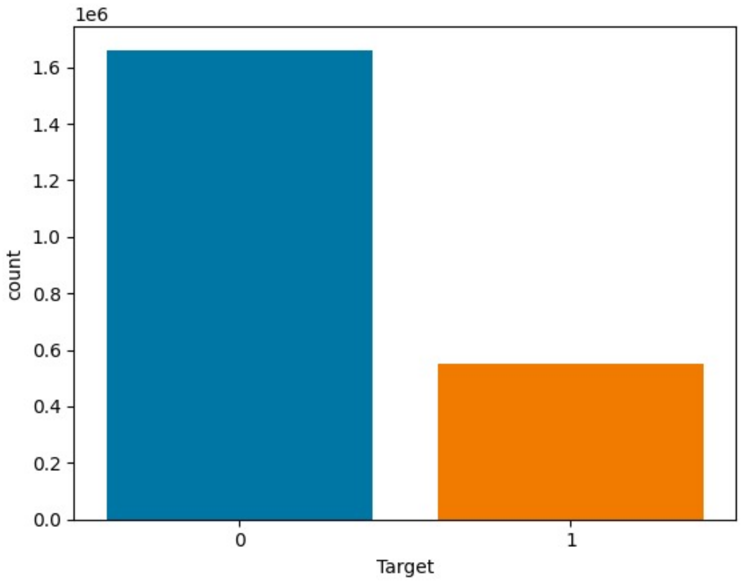


Figure 3: Distribution of targets against customer count

information to the model it helps us to increase the size of the minority data. Instead, we can synthesize new examples from the existing examples.

5.4.2 Undersampling

In undersampling, it randomly selects a few samples from the majority class and removes them in the training dataset prior to fitting a model. We may lose a lot of information with this step. Instead, we can selectively remove samples which do not show much effect on the dataset.

5.4.3 Synthetic Minority Oversampling Technique (SMOTE)

SMOTE is one of the most popular oversampling techniques that is developed by Chawla et al. [3]. Unlike random oversampling that only duplicates some random examples from the minority class, SMOTE generates examples based on the distance of each data (usually using Euclidean distance) and the minority class nearest neighbors, so the generated examples are different from the original minority class. In short, the process to generate the synthetic samples are as follows:

- Choose random data from the minority class.
- Calculate the Euclidean distance between the random data and its k nearest neighbors.
- Multiply the difference with a random number between 0 and 1, then add the result to the minority class as a synthetic sample.

- Repeat the procedure until the desired proportion of minority class is met.

This method is effective because the synthetic data that are generated are relatively close to the feature space on the minority class, thus adding new "information" on the data, unlike the original oversampling method.

5.5 XGBoost

Extreme Gradient Boosting (XGBoost) is a distributed, scalable gradient-boosted decision tree (GBDT) machine learning framework. It offers parallel tree boosting and is the top machine learning package for regression, classification, and ranking issues. This technique generates decision trees in a sequential manner. In XGBoost, weights are significant. All independent variables are given weights, which are subsequently used to feed information into the decision tree that forecasts outcomes. The second decision tree is then fed these variables after increasing the weight of variables that the tree incorrectly predicted. To create a robust and accurate model, these independent classifiers and predictors are then combined.

5.6 Dimensionality Reduction

5.6.1 Principle Component Analysis

Principal Component Analysis is a well-known unsupervised learning method for reducing data dimensionality. While minimizing information loss, it simultaneously improves interpretability. It makes data easier to plot in 2D and 3D and assists in identifying the dataset's most important features. Finding a series of linear combinations of variables is made easier by PCA.

5.6.2 Single Valued Decomposition

The SVD method essentially divides any matrix into three generic and familiar matrices through the use of the matrix factorization technique. Datasets with a lot of values are condensed using the singular value decomposition. For a more concise representation of these correlations, a Singular Value Decomposition study is supportive and produces findings. You can gain knowledge about temporal and spatial variations by using multivariate datasets. These variations exhibit data after the analysis.

6 Results

When initially implemented to handle null values, we can see from Table 1 that replacing the null values with mode produces the best results when trained with XGBoost, and the model with median takes the shortest amount of time to train. We can observe from Table 2 that by using PCA, the training time is decreased to 266 ms, however the model is likewise not ideal. After implementing SVD, the training time is reduced to 130 ms, but the model's accuracy is also decreased, as shown by Table 3.

Model	Precision	Recall	F1-score	Time
XGBoost	0.762720	0.760015	0.761365	513.335
After Using Mean	0.762866	0.757917	0.760384	441.554
After Using Mode	0.763554	0.761305	0.762428	436.924
After Using Median	0.763173	0.759168	0.761165	434.740

Table 1: Impact of null replacement with column metrics

XGBoost	Before PCA	After PCA
Time	736.860	266.520
Precision	0.76123	0.745803
Recall	0.75937	0.729554
Fl - Score	0.760305	0.737589

Table 2: Impact of applying PCA

XGBoost	Before SVD	After SVD(30)
Precision	0.76123	0.737751
Recall	0.75937	0.725895
F1-score	0.760305	0.731775
Time	736.860	130.087

Table 3: Impact of applying SVD(30)

If we have a closer look at the problem statement for the dataset, we can observe that we are supposed to use the dataset to predict if a customer will default in the future. The dataset is structured in the form of profile values for each customer over a period of time. We can see that they are uniformly distributed with about the same amount of time-series data for each customer. In order to train a model on this data, we can either:

- Take the average of all the columns of the data for each customer and train the model on the average.
- Process it like we would any other time-series data.

Taking the average would only work if all the categorical values for each customer are the same, since if there are more than one categorical values for each customer, we cannot average the values with any meaningful statistic.

If we want to process it like time-series data, we want to make sure that the target value that we have is the same for each row of the customer data, which we can then take as the y_train data. Fortunately, it turns out that that is the case!

With random undersampling, the accuracy increases by 1.5% and training time decreases three fold. With random oversampling, the accuracy increases similarly, but training time remains unaffected, as shown in Table 4.

Model Used	Training Time	Accuracy
Batching during training (tensorflow)	1hr+	77.125%
Batching then Training – regression (JAX)	1m 23s	77.125%
Batching then Training – classification (JAX)	1m 23s	87.68%
Batching then Training – classification + undersampling (JAX)	33.7s	89.00%
Batching then Training – classification + oversampling (JAX)	1m 32s	89.02%

Table 4: Impact of oversampling/undersampling

7 Conclusion

With this study, we were able to choose a few preprocessing techniques which had a significant impact. One Hot encoding was suitable for this dataset, as it does not have an ordinal relationship with categorical data, unlike Label encoding. We observe that filling null values with the mean of columns gives us a higher F1 score than median or mode. After performing SVD, we observed a decrease in training time when compared to PCA. From the exploratory data analysis, we think it is because of the sparse data present in our chosen dataset, since SVD tends to perform better with sparse data. Batching data during training allowed us to use single-node infrastructure. We used random undersampling and oversampling and found that undersampling reduces training time by up to three fold. However, both methods result in similar performance metrics.

8 Future Work

We would like to explore more machine learning preprocessing methods, such as standardization and multi-collinearity, to further improve the outcomes. Additionally, even though SMOTE provides balanced data, we would lose a number of data points if we improved the model. So, we want to look into various cutting-edge approaches such as Tomek and SMOTEEN.

References

[1] J. Van Hulse, T. M. Khoshgoftaar, and A. Napolitano, “Experimental perspectives on learning from imbalanced data,” in *Proceedings of the 24th International Conference on Machine Learning*, ser. ICML ’07. New York, NY, USA: Association for Computing Machinery, 2007, p. 935–942. [Online]. Available: https://doi.org/10.1145/1273496.1273614

[2] G. E. D. A. P. A. Batista, D. F. Silva, and R. C. Prati, “An experimental design to evaluate class imbalance treatment methods,” in *2012 11th International Conference on Machine Learning and Applications*, vol. 2, 2012, pp. 95–101.

[3] K. W. Bowyer, N. V. Chawla, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: synthetic minority over-sampling technique,” *CoRR*, vol. abs/1106.1813, 2011. [Online]. Available: http://arxiv.org/abs/1106.1813