Customer Churn Prediction on the Cell2Cell Dataset by Deep Learning

Name: R. Abdulkadir
Student number: U521037 (2059286)

THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN DATA SCIENCE & SOCIETY
DEPARTMENT OF COGNITIVE SCIENCE & ARTIFICIAL INTELLIGENCE
SCHOOL OF HUMANITIES AND DIGITAL SCIENCES
TILBURG UNIVERSITY

Thesis committee:

Supervisor: Prof. Dr. E. Postma
Second reader: Dr. Ç. Güven

Tilburg University
School of Humanities & Digital Sciences
Department of Cognitive Science & Artificial Intelligence
Tilburg, The Netherlands
14-01-2022

Contents

Preface

This thesis completes my master program Data Science & Society: Business & Governance at Tilburg University.

I would like to thank my professors and assistant professors for teaching me the art of data science in the past year. I learned a lot of concepts and became a programmer only in the span of a year.

Special thanks to my supervisor Prof. Dr. E. Postma for coming with interesting points of views whenever I was stuck and to Dr. Ç. Güven for being my second reader and teaching me machine learning.

Lastly. I would like to thank my family and friends who supported me when needed during the track of the master and specifically during the writing of this thesis.

Rwand Abdulkadir
Tilburg, January 2022

Abstract

Customer churn prediction is the identification of customers that are possibly going to stop using a service. This service is often paid periodically. Because of this, it is important for companies as a lost customer equals to lost revenue. Customer churn has extensively been researched in the fields of machine learning and deep learning. A commonly used dataset is the Cell2Cell dataset. This dataset is unbalanced and contains numerical and categorical information about customers of a telecom company. Although machine learning has often been applied to this dataset with promising results, deep learning has not been applied often. There are two previous studies done on this dataset in regard to deep learning (Umayaparvathi & Iyakutti, 2017), (Ahmed, Khan, Khan, Basit, & Lee, 2019). The 2017 study has room for improvement in terms of performance. For this reason, this thesis is focused on improving performance on the Cell2Cell dataset by using deep learning techniques. Previous deep learning research did not apply resampling techniques to tackle the problem of imbalance. This study applied several resampling techniques to overcome the problem of imbalance. This is done by building two baselines. One for the dataset with all the features, and one for the dataset with a subset of features that are picked by the mutual information feature selection algorithm. Next, unsampled and resampled training sets are applied to the networks. The resampling techniques are random undersampling, random oversampling and SMOTE.

After building, fitting, and tuning the models, the models were evaluated. The evaluation metrics are accuracy and Area Under the Curve (AUC). With accuracy, the model can be compared to previous studies. The best performing model turned out to be the model without the resampling applied. This model performed with an accuracy of 70.45 and an AUC of 0.62. Because of this result, it can be concluded that resampling techniques did not improve performance.

.

Data Source/Code/Ethics statement

- "Work on this thesis did not involve collecting data from human participants or animals. Student research with human participants only requires evaluation by the REDC if the collected data will be used by supervisors or with a serious intent to publish the results in a journal that requires formal ethical clearance. Student research with human participants does not have to be evaluated by the REDC if the collected data will not be used by the supervisor and will not be used for publications in journals that require ethical clearance. In that case, self-evaluation with the checklist below is sufficient.
- The original owner of the data and code used in this thesis retains ownership of the data and code during and after the completion of this thesis. In cases the code or data is obtained from external sources, the author of this thesis acknowledges that they do not have any legal claim to this data or code. In case the data or code used in this thesis was obtained from an external source was added to, changed, or was the basis for the acquisition of new data or code, and if the additional data was collected from human participants or animals: The author of this thesis has evaluated his/her project according to the "Ethics checklist Student research with human participants".
- If the data will be used for research projects for TiU-based researchers, the supervisor of this research project obtained an evaluation of compliance from the TSHD Research Ethics and Data Committee.
- The code used in this thesis is publicly available [https://github.com/Rwand-code/THESIS]

1. Introduction

1.1 Context description

The telecommunication (telecom) market is saturating at an increasing growth (Capgemini, 2015). This saturation leads to telecom providers focusing on retaining customers instead of acquiring them. Customer churn prediction is identifying the customers that are possibly going to stop using a service.

Predicting customer churn is crucial for subscription-based services. This is because of the customer acquisition costs and the customer retention costs. Various studies have shown that retaining customers is more cost-efficient than acquiring new customers (Reichheld, 2001), (Dolatabadi & Keynia, 2017). Retaining customers is therefore one of the most important aspects of a service providing business. Customer churn has been extensively researched by using traditional machine learning and deep learning approaches. See section 2.1 for a literature overview about previous research on customer churn prediction.

1.2 Scientific Relevance

Customer churn is mostly researched with machine learning techniques. Deep learning has not been applied often. For the Cell2Cell dataset, two prior studies used deep learning techniques to predict customer churn. One of these applied transfer learning on a convolutional neural network (CNN) and showed promising results (Ahmed, Khan, Khan, Basit, & Lee, 2019). The other research focused on feedforward neural networks and convolutional neural networks (Umayaparvathi & Iyakutti, 2017). Using convolutional neural networks for the Cell2Cell dataset is illogical. These types of networks require neighborhood relationships between the instances. The features in this dataset have no neighborhood relationships to each other. Therefore, using a CNN is unreasoned. In addition, both of these researches have limitations in the application and have room for improvement in the performance. Application limitations are the lack of feature selection, sampling methods and hyperparameter optimization.

The objective of this thesis is to improve the performance of the current deep learning applications on the Cell2Cell dataset. These are the research of 2017 and of 2019 described above. The scientific relevance of the research reported in this thesis follows from three reasons: feature

selection, sampling methods, and hyperparameter optimization. Each of these reasons will be described below.

Feature selection

Feature selection is selecting the most relevant input features for the model. This is important because it reduces overfitting and training time, while simultaneously improving performance (Zebari, Abdulazeez, Zeebaree, & Saeed, 2020). The 2017 and the 2019 paper both did not apply feature selection methods because the goal of both the studies was to have the features automatically detected. The authors may have decided that because of automatic feature extraction, feature selection is not necessary. However, a neural network can still suffer from overfitting when there are too many features as too many features lead to an increase in model complexity. This increase in model complexity can lead to an overfitting model.

Sampling methods

The Cell2Cell dataset is an imbalanced dataset. This means that the target variable label distribution is uneven. The distribution ratio for this dataset is 71:29. 79 referring to non-churn as the majority class and 29 to churn as the minority class. Doing experiments with these distributions lead to overlooked minority classes. This means performing well on the majority class, but poorly on the minority class (Amin, et al., 2016). A method of addressing this problem is by resampling the data. The two types are undersampling and oversampling. Undersampling deletes instances from the majority class, oversampling resamples instances from the minority class. Other than random oversampling, there is a technique named Synthetic Minority Oversampling Technique (SMOTE). Instead of random resampling, SMOTE creates synthetic instances by using interpolation between samples of the minority class. All of these methods have their advantages and disadvantages, as can be found in section 3.1.4: Resampling methods.

The problem of imbalance contributes to the final limitation of previous research, namely using accuracy as the measuring method. When the data is imbalanced accuracy is no longer a proper method as it leads to erroneous conclusions. With the Cell2Cell dataset, the model can lead to an accuracy of 70% while not correctly predicting any minority class cases correctly. This seems like a satisfactory performance, but in reality, the metric does not represent performance. A proper

metric for an imbalanced dataset with more negative class samples should capture the false positive rate well. AUC is able to capture the false positives rates and true positive rates. Because of this, the metric takes the imbalanced data into account (Shuyu, 2020).

Hyperparameter optimization

Hyperparameters are configurations to the model that decide how the model behaves. These parameters cannot be learned by the model itself. It can only be learned by a validation set or manually tuned by the builder of the model. Hyperparameters have influence on training time, convergence, and performance. The previously mentioned papers did not apply hyperparameter optimization. This could have contributed to the performance. Optimizing these parameters can lead to a better performing model (Shankar, Zhang, Liu, Wu, & Chen, 2020).

1.3 Research questions

The following research question and its related sub-questions are used to further analyze this research:

The main research question is:

To what extent can deep learning improve customer churn prediction when using sampling methods?

This research question is focused on deep learning for churn because the currently applied deep learning conducted research on the Cell2Cell dataset has limitations. There is no feature selection method applied and no sampling techniques are used to tackle the problem of unbalanced classes while simultaneously, accuracy is used as a measuring method.

To answer the main question, 2 sub questions are answered:

1. *To what extent does feature selection improve performance in predicting customer churn in the Cell2Cell dataset?*

The raw dataset has 58 variables. After preprocessing, this number increases to 69 variables. Feeding all these variables in the neural network will likely make the network overfit. Whether this will be true is tested by conducting experiments with and without feature selection.

2. *Which sampling method provides the best performance?*

Class imbalance makes it harder to predict the minority class. A way to address this is by resampling the training data. The training data has been oversampled by using random undersampling, random oversampling and SMOTE. These resampled models will be compared to a baseline model without resampling.

1.4 Findings

Two baseline models are built. One for the dataset with all the features, and one for the dataset with a subset of features that are picked by the mutual information feature selection algorithm. Next, unsampled and resampled training sets are applied to the networks. The resampling techniques are random undersampling, random oversampling and SMOTE. This results in a total of 8 different training sets, including the unsampled models. After building, fitting, and tuning the models, the models were evaluated. The evaluation metrics are accuracy and Area Under the Curve (AUC). With accuracy, the model can be compared to previous studies.

It turned out that the best performing model is the unsampled dataset with an AUC of 0.62 and an accuracy of 70.45. These improvements however, did not improve previous studies. In fact, the accuracy of decreased. Although, this is not a sign of a worse performing model because accuracy does not caption imbalance, it does show that resampling did not improve performance significantly.

2. Related work

This chapter explores previous conducted research on customer churn. Previous research gives an understanding on which methods are effective and which have limitations. Furthermore, previous research can reveal how this study can contribute to the currently available literature.
The chapter starts with the explanation of customer churn and how machine learning and deep learning is used to predict churn for the Cell2Cell dataset (2.1). Next, the limitations of current research are discussed (2.2). Lastly, section 2.3 proposes the method that will be used in this research.

2.1 Customer churn

Predicting customer churn is extensively researched in many markets such as social media, banking, and subscription-based services such as the telecom industry. CCP is a classification problem. The aim of CCP is classifying the customers into two categories: churn and non-churn. To tackle this problem for telecom, traditional machine learning and deep learning techniques are used to classify the groups. Customer churn classification tasks often deal with imbalanced data. This means that the target classes are imbalanced. For churning, the majority class is non-churn and the minority class is churn.

Customer churn in the telecom industry

A study in 2021 predicted customer churn by comparing various machine learning algorithms. The contribution of this study was the proposal of mutual information feature selection in combination with ranker methods. The feature selection with ranker method outperformed the models without feature selection. The best performing model was the naïve bayes with an accuracy and F-measure of 86.86 and 82.30, respectively (Saheed & Hambali, 2021).
Churn prediction by a Convolutional Neural Network (CNN) has also been applied in the literature. This CNN scored an accuracy and F-score of 86.9 and 92, respectively (Mishra & Reddy, 2017). These are remarkable when looking at a snapshot of the dataset provided by the paper. This snapshot shows that the features in this dataset have no spatial relationships. The strength of a CNN is learning from these spatial relationships. When these are nonexistent, it is illogical to use CNN’s.

A 2018 study focused on churn with the emphasis on imbalanced data. This research applied SMOTE and random undersampling with bagging techniques. This is to not lose potential crucial information with the undersampling, but simultaneously not overfitting much with SMOTE. After resampling, the model was fitted by using the C4.5 decision tree algorithm. The results showed that the combination of random undersampling and SMOTE improved the F-score. The F-score reportedly increased with 56 percent (Hartati, Adiwijaya, & Bijaksana, 2018). This is a promising result for the problem of imbalanced data.

Customer churn on the Cell2Cell dataset

Various studies have been done on the Cell2Cell dataset to predict customer churn. A 2021 study analyzed three datasets in the telecom industry, including the Cell2Cell dataset. The authors created an integrated churn and segmentation framework by using machine learning techniques. The used datasets had imbalanced classes for the predicting variable. To address this problem, they applied a Synthetic Minority Oversampling Technique (SMOTE) to the training set and used Accuracy, F1-score, and AUC to measure the performance of the models. The best performing model for the Cell2Cell dataset was a Random Forest model with an accuracy of 63.09. The authors suggest using different under sampling and oversampling techniques in the future as the accuracy, F1 score, and AUC had room for improvement. Also, other feature extraction and clustering techniques can be used to further improve on this research (Wu, Yau, Ong, & Chong, 2021).

A study in 2017 had a deep learning approach to this dataset by comparing a small Feed Forward Network (FNN) with 3 layers, a large FNN with 4 layers and a CNN with 7 layers. The goal of this study was to use automatic feature selection by the deep learning model. Because of this, no feature selection methods are used. The small FNN performed the lowest with an accuracy of 64.8. The Large FNN and CNN performed the same with an accuracy of 71.67 (Umayaparvathi & Iyakutti, 2017). Compared to the baseline of 71.27, it is clear that the performance did not increase significantly.

Similar to the 2018 study, the features of the Cell2Cell dataset have no neighborhood relationships. This makes the usage of a CNN unjustified for the Cell2Cell dataset. Also, it is important to note that this study used accuracy as the performance measure, while not taking the imbalanced nature of the data into account. Accuracy is not a proper measurement when the target class of the data is

imbalanced. The Cell2Cell dataset has a 72:28 imbalance. This means that if the model predicted every sample as the majority class, the accuracy of the model will still be 72%. This is a misleading result and a common problem with classification tasks (Menardi & Torelli, 2014).
Other than using accuracy as a performance metric, the authors also did not take the imbalanced data into account and therefore did not resample the data. Resampling the data and using different performance measures can improve this research. Lastly, the paper did not apply hyperparameter optimization. This could have contributed to the performance. Optimizing these parameters can lead to a better performing model (Shankar, Zhang, Liu, Wu, & Chen, 2020).

A 2019 study tried to improve the aforementioned results by proposing transfer learning models for the CNN. The used models are AlexNet, Inception-ResNet-V2 and a custom CNN consisting of 6 layers. The result of this study showed an accuracy of 68.16 and an Area Under the Curve (AUC) of 0.74. Moreover, the study mentions that classification tasks as churning tend to classify all samples as non-churn, as non-churn is the majority class. This leads to high accuracy but low precision. To capture the true functionality of the model, the AUC is used as a measurement (Ahmed, Khan, Khan, Basit, & Lee, 2019). In addition, the hyperparameters are not tuned and, Similar to the 2018 study, the usage of a CNN is unjustified as there are no neighborhood relationships between the variables in the dataset.

2.2 Current research limitations

The challenging part in customer churn is in the imbalance of the target classes. The non-churning class is more prominent than the churning class. This makes sense from a business perspective; a company would not exist if there would be more churn than non-churn, but this makes it harder to predict churn. Class imbalance occurs when the class proportions are not similarly distributed. This imbalance might lead to models performing well on the majority group, but poorly on the minority group (Amin, et al., 2016).

The current deep learning methods have not applied feature selection, resampling and hyperparameter optimization to the Cell2Cell dataset. The 2017 paper however, did use stratified cross validation to avoid the problem of imbalanced data. Whether this stratified method worked is not clear, because there is no metric used that takes the imbalance into account. Moreover, the

FNN's in this have arbitrary chosen layers. 3 layers for a 'small' network and 4 layers for a 'large' network. The reasoning behind the chosen parameters is also not stated.

2.3 Proposed methodology

Feature selection will be applied to the model. Previous research did not apply this because the goal was to automatically select features. Whether this contributed to the poor results will be analyzed. This will be done by conducting models with feature selection and without feature selection. The chosen feature selection method is mutual information feature selection. This method showed improvements in performance in previous research (Saheed & Hambali, 2021). Moreover, current deep learning research on the Cell2Cell dataset has not applied sampling techniques to tackle the problem of imbalanced target classes. The methodology proposed in this research will use multiple sampling techniques. The chosen techniques are random under sampling, random oversampling and SMOTE. These methods are chosen because of promising results (Amin, et al., 2016). Next, the hyperparameters of the feedforward neural networks from the 2017 paper (Mishra & Reddy, 2017) can be improved to lead to more optimal results. Instead of building large and small neural networks with arbitrary chosen layers and neurons, the layers, neurons, and other hyperparameters can be chosen by hyperparameter tuning. Additionally, proper metrics will be used to measure the performance of the model. The dataset target ratio is 71:29. The majority group is the negative class of non-churn. A proper metric for an imbalanced dataset with more negative class samples should capture the false positive rate well. AUC is able to capture the false positives rates and true positive rates. Because of this, the metric takes the imbalanced data into account (Shuyu, 2020).

3. Method

Chapter 3 provides the approach to the model. 3.1 describes what the preprocessing steps are and why they have been taken. 3.2 Describes the mutual information feature selection method and why it has been chosen. Subsequently, 3.3 will discuss the sampling methods that will be applied. feedforward neural networks and the hyperparameter optimization. Next, 3.4 describes the feedforward neural networks and the hyperparameter optimization. Finally, the proposed method is outlined.

3.1 Data preprocessing

Data preprocessing is the process of taking raw data and applying different techniques to make the dataset suitable for building a model. The techniques used in this study are cleaning, encoding, splitting, normalization and applying different sampling techniques. The goal of data preprocessing is to process the data in a way that is optimal for the model that is going to be used. This makes the preprocessing dependent on the type of model that is used (Jangir, 2021). In this study FNN are used and therefore, the preprocessing will be done to optimally feed the feedforward neural network.

With preprocessing it is important to prevent the data from leaking. Data leakage happens when a model is built with data other than the training data (Gutierrez, 2014). The leakage does not happen on purpose, but rather accidently. An example of data leakage is when normalization happens before the split. In this case the testing data is also calculated and used for the distribution of the data. This means that the information is ‘leaked’ into the testing data. When this happens, the model is partially built on testing data. This will lead to high performances while the model did not perform well, it only remembered the data it was built on.

3.1.1 Cleaning

Data cleaning is the process of identifying, removing, and replacing data that is incorrect. Real world datasets often have irrelevant data, duplicates, missing data, and outliers. Leaving this type of data in the dataset can cause errors and introduce bias.

3.1.2 Dummy coding categorical data

Dummy coding is done to convert categorical data into numerical data. This is done by assigning a binary value of 1 or 0. 1 is assigned when the data sample belongs to the class and 0 is assigned to all other classes.

3.1.3 Splitting

Splitting a dataset makes it possible to evaluate the model on data that is not seen before by the model. In this research the dataset is split in three parts: training, validation, and test. The training set is used to fit the model. After fitting the model, the model's hyperparameters get tuned on the validation set. The final set is the test set. The test set gives an unbiased evaluation of the model after fitting and tuning the hyperparameters. The evaluation is unbiased as the model has not seen this test set before.

3.1.4 Normalization

After splitting the data, the data gets normalized on the training set. Normalization is scaling input variables between 0 and 1. Normalization makes fitting a model more efficient in time and computational power. There are multiple types of normalization that are applied to range data. The goal of normalization is scaling the input variables to minimize bias of one variable to another (Jayalakshmi & A, 2011). In this study, Min-Max normalization is applied to the data. This method preserves the relationships between the data and scales all variables within the same range. The disadvantage of Min-Max scaling is that it does not handle outliers well. This is not a limitation in this study however, as the outliers are handled in the cleaning part.

3.2 Feature selection

Two type of training sets will be fed to the network. One type is the training set that contains all the features after dummy coding. This is a total of 69 features. The other training set will be a subset of the features. This subset is chosen by mutual information feature selection. Mutual information feature selection is a feature selection method that will be further discussed in the section below.

Mutual Information Feature selection

The selected subset of the features is always a subset of the original dataset the process of selecting the best features for the model. This process is applied to remove features from the dataset that have no added value for building the model.

Mutual information measures the dependency between two random variables. This can be seen as the intersection between the two random variables. This is similar to Pearson's correlation coefficient but there is a difference. The difference is that mutual information is the calculated distance in probability whereas Pearson's correlation coefficient is not. Mutual information can never have a negative value. The higher the value, the higher the dependency is between the two random variables. The equation of mutual information is defined as follows:

$$I(X,Y) = \sum p(x,y)\ [ln\ p(x,y) - ln\ \ p(x)p(y)] \quad (1)$$

$p(x,y)$ is the weighted probability of *X* and *Y*. $p(x)$ *and* $p(y)$ are the marginal probabilities of *X* and Y.

3.3 Resampling methods

The training data will be resampled. As stated before, the dataset is imbalanced. There are two ways to resample the data to make it balanced: Under-sampling and oversampling.

Random undersampling

Undersampling is reducing the majority target group to a certain size. Often this is to match the same size as the minority target group. The advantage of this method is retaining the class distribution of the data. Undersampling also reduces computational time as the majority class is reduced. This will always lead to less computation time. The disadvantage of under sampling is losing potentially critical information to make correct predictions. Moreover, undersampling techniques often produce noisy data (Kim, Jo, & Shin, 2016). Undersampling is only used when the dataset is large enough. When the dataset is large enough, there will be enough data in the reduced sample that it represents real world data.

Random oversampling

Oversampling is oversampling the minority target group to a certain size. This method oversamples by copying a random sample of the minority class. Like undersampling, this is often to match the size of the other group, in this case the majority group. The disadvantages of oversampling are potential overfitting and increased computational power (Kim, Jo, & Shin, 2016).

SMOTE and its variations

SMOTE is an acronym for Synthetic Minority Oversampling Technique. A paper in 2002 came up with this approach as an alternative to oversampling. SMOTE was tested on diverse datasets and performed better than random oversampling and undersampling (Chawla, Bowyer, Hall, & Kegelmeyer, 2002).

3.4 Deep learning

Deep learning is a subfield of machine learning that uses neural networks with multiple hidden layers. A neural network is considered deep when it has multiple hidden layers between the input and output layer. The purpose of multiple hidden layers is to solve problems that are too complex for traditional machine learning algorithms. Because of the complexity and multiple hidden layers, deep learning models require lots of data to train (Hu, Lingyang, Jian, Weiqing, & Jiang, 2021).

3.4.1 Deep Feedforward Neural Networks

A feedforward neural network is a deep learning model that feeds data forward in only one direction. The goal of the network is to find a line that splits the data into two classes. The network consists of layered nodes and weights, The weights get multiplied by the input plus a bias term. This results in an output value. Bias is important since bias ensures that the decision boundary will not always go through the origin. This makes it possible to yield multiple kinds of decision boundaries. Without bias, the decision boundary will always go through the origin (Hockenmaier, 2013). Data in a FNN starts at the input nodes. It gets feedforward from the input nodes to the hidden nodes in the hidden layer. The final layer is the output node. The application of multiple layers is how a FNN distinguishes itself from a traditional machine learning model (Goodfellow, Bengio, & Courville, 2016).

3.4.2 Hyperparameter optimization

After building the model, the model is trained on the training set. After training, the model is tuned on the validation set. FNN's have multiple hyperparameters that can be tuned. Hyperparameters can relate to the structure of the network, but also the variables that determine how the network is trained (Diaz, Fokoue, Nannicini, & Samulowitz, 2017). The hyperparameters that will be tuned are the number of layers, number of nodes, activation function, drop rate, regularization rate, and the learning rate. See Section 4.5.3: Hyperparameter tuning for the detailed explanation of the hyperparameters.

3.5 Proposed method

The focus of this research is on feature selection, resampling methods, and hyperparameter optimization. Because of this, the following steps will be taken:

The first step is building two baseline feedforward neural network models. One baseline model is for the dataset without feature selection, the other baseline is for the dataset with mutual information feature selection applied. The second step is feeding both of the baseline models with unsampled and resampled data. These are unsampled, randomly undersampled, randomly oversampled and SMOTE. The last step is optimizing the hyperparameters of all the models by automatically tuning the hyperparameters with the Keras tuner (O'Malley, et al., 2019). This will lead to a total of 16 models that will be built.

4 Experimental Setup

4.1 Description Cell2Cell dataset

The dataset is a customer churn dataset retrieved from Kaggle (Kaggle, 2018). The name of this dataset is: telecom churn (cell2cell). It contains 71,047 instances, 58 features and 3491 missing values. The data is split into two csv. files. There are two limitations to this dataset. First, the dataset has been split into two csv. files. One of the files is for training and the other for holdout. The training set consists of 51,048 instances and the holdout of 19,999. The training set has labels, and the holdout set does not. This makes the holdout set unusable for supervised machine learning techniques. The second issue is the imbalance. The description of the dataset claims balanced target classes, but the target classes are distributed with 36,336 for the majority class of "non-churn" and 14,711 for the minority class of "churn", this means a distribution of 71% and 29%, respectively. Figure 4.1 shows the distribution in bars. 'Yes', refers to churning and 'No' refers to not churning. This plot visualizes the imbalance clearly.

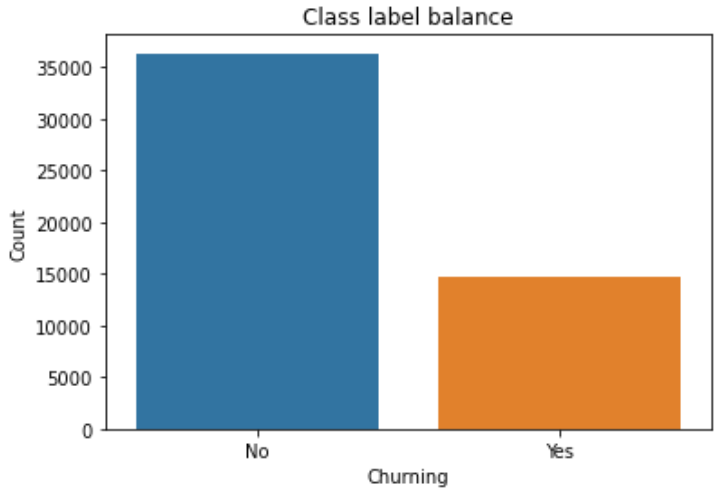

Figure 4.1: Class label distribution for the Cell2Cell dataset

The dataset has multiple data types and consists of continuous and categorical features. When looking at the quantiles of each feature, it is clear that the data is on different scales. See figure 4.2 below for the quantiles and statistics of the first 6 features.

	MonthlyRevenue	MonthlyMinutes	TotalRecurringCharge	DirectorAssistedCalls	OverageMinutes	RoamingCalls
count	50891.000000	50891.000000	50891.000000	50891.000000	50891.000000	50891.000000
mean	58.834492	525.653416	46.830088	0.895229	40.027785	1.236244
std	44.507336	529.871063	23.848871	2.228546	96.588076	9.818294
min	-6.170000	0.000000	-11.000000	0.000000	0.000000	0.000000
25%	33.610000	158.000000	30.000000	0.000000	0.000000	0.000000
50%	48.460000	366.000000	45.000000	0.250000	3.000000	0.000000
75%	71.065000	723.000000	60.000000	0.990000	41.000000	0.300000
max	1223.380000	7359.000000	400.000000	159.390000	4321.000000	1112.400000

Figure 4.1: Statistical descriptors for the first 6 features in the Cell2Cell dataset.

It is clear that there is a remarkable difference in scales looking at the features. A clear example is MonthlyRevenue and DirectorAssistedCalls. Every calculated statistic is on a significantly different scale.

Other than different scales, the statistics appear to show outliers in the data. To further investigate these, boxplots and density plots are plotted. See figure 4.3 and 4.4 for the distribution plot and boxplot for the MonthlyRevenue feature as an example:

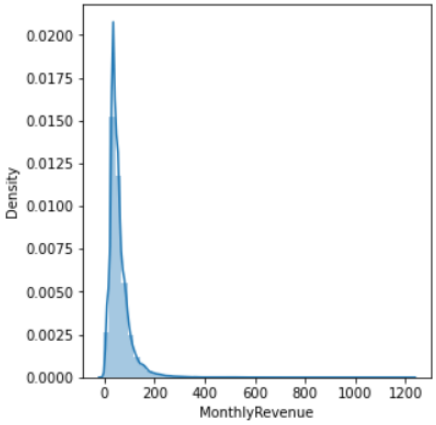

Figure 4.3 Density plot for the MonthlyRevenue feature

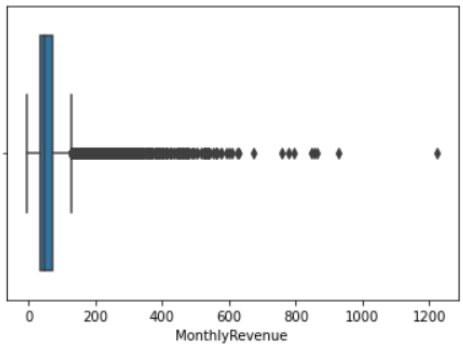

Figure 4.4: Boxplot for the MonthlyRevenue feature

Identifying the different scales and outliers is important to note for further modeling. Not treating the different scales and outliers can lead to bias, overfit and other limitations of the model.

4.2 Applying preprocessing

The data is imported by using Pandas' read_csv function (Pandas, 2021). First the irrelevant columns are dropped. These are CustomerID and ServiceArea. CustomerID is irrelevant to predicting as the only purpose is the reference to the customer. ServiceArea is not relevant because it is encrypted in a way that the data cannot be interpreted. Next step is detecting duplicated rows. When attempting to detect these, it appeared that there are no duplicated rows in this dataset. Moreover, the dataset has little missing values for the continuous data. Total rows of missing data counts for 0.025% of all the data. Because of this low amount, these rows are dropped from the dataset.

When looking at the categorical variables it appears that there are no missing values. This is misleading however when the unique values per feature are shown. This reveals that there are two categorical features with the value 'unknown'. These features are HandsetPrice and MaritalStatus. The 'unknown' value is significantly larger than the other values in these features. 57% for HandsetPrice and 39% for MaritalStatus. Using domain knowledge, these features do not appear to be relevant for predicting customer churn. The HandsetPrice is the additional costs a customer pays for the contract to obtain the phone. Since this is a onetime cost, it likely will not affect the choice of churning. MaritalStatus is the marital state of the customer. This also appears not to be relevant for customer churn. Because of the aforementioned reasons, the two features are dropped from the dataset.

Binary values in the dataset have strings as values. The values are 'Yes' and 'No'. These values have been changed to 0 and 1. 0 referring to 'No' and 1 to 'Yes'. This is needed because further processing of the data cannot be done with string values. The last step of the cleaning process is handling outliers. First the outliers are detected by plotting the distribution of the data by boxplots. Figure 4 shows that there are data points outside of the whiskers. These data points are considered outliers. Instead of removing outliers and losing valuable data, the outliers are handled by replacing them with median values. All values above the 99^{th} percentile is replaced with the median value. This technique is applied to continuous features only.

The last step before splitting the data is encoding. For encoding the data, the categorical features are dummy coded using the get_dummies function of Pandas (Pandas, 2021). This function gives dummy variables for CreditRating and Occupation. The remaining categorical features are binary.

Binary features do not need encoding. After encoding the data is split in training, validation, and test. This is done by using the train_test_split function of sklearn (Scikit Learn, 2021), To generate a validation set, the test set is split. The ratios are 80/10/10 for training, validation, and test, respectively.

Before feeding data to the network the data needs to be normalized. This reduces training time and overfitting. For normalization the MinMaxScaler function is used from sklearn (Scikit Learn, 2021). This function is applied to the training, validation and test set separately. Later in the application of different sampling methods to the neural network, the resampled training sets are normalized as well.

4.3 Mutual Information feature selection

Feature selection is done by using mutual information feature selection. This is done by using the mutual_info_classif and SelectKBest functions of sklearn (Scikit Learn, 2021), (Scikit Learn, 2021). The SelectKBest function takes the mutual_info_classif function as an argument and picks the K best performing features. This K argument can be set by the K parameter. The initial dataset had 69 features when dummy coded. To keep the model from overfitting, only 20 features are chosen to fit the model with. The functions were only applied to the training set to prevent data leakage. To have consistent features in all splits, the validation and test set are transformed. This gives all the sets the same features and the same dimensions.

4.4 Applying sampling methods

Similar to feature selection, the resampling of data is applied only to the training set to prevent data leakage. Three different types of resampling are used from the imblearn library in Python (Imb Learn, 2021). The functions are RandomUnderSampler, RandomOverSampler and SMOTE. The functions generated training samples that are further being used in the experimental process.

4.5 Experimental procedure

4.5.1 Building baseline models

The task is to predict customer churn by using feedforward neural networks. The first step is to import the libraries needed. These are Keras, TensorFlow and sklearn. After importing the libraries, the architecture of the network is built. The simple architectures are the baseline of the model. There are two baseline models. One is for the network without feature selection, the other is for the network with mutual information feature selection. Figure 4.5 is the architecture for the network without feature selection, 4.6 is the model with mutual information feature selection.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 64)                4480
_________________________________________________________________
dense_1 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 33
=================================================================
Total params: 6,593
Trainable params: 6,593
Non-trainable params: 0
```

Figure 4.5: baseline feedforward neural network without feature selection

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 32)                672
_________________________________________________________________
dense_4 (Dense)              (None, 16)                528
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 17
=================================================================
Total params: 1,217
Trainable params: 1,217
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.6: baseline feedforward neural network with mutual information feature selection

These models have 1 input layer, 1 hidden layer, and 1 output layer. The activation functions for the input layer and the hidden layer are ReLU. ReLU is chosen because of faster computation time and reducing the vanishing gradient problem. The output layer has sigmoid as an activation function. Sigmoid is chosen because it outputs a probability between 0 and 1 and with that, introduces nonlinearity into the model (Talathi & Vartak, 2016). This probability output helps with calculating various metrics for the evaluation of the model.

4.5.2 Fitting data

Next, the architectures are applied to the sampled and unsampled training sets. These are the non-sampled, randomly undersampled, randomly oversampled and SMOTE - dataset. All the models have the same parameters. The parameters are batch_size=32, epoch=50. Batch size 32 is chosen to be processed in parallel with the CPU. 32 is not computationally expensive and simultaneously effective (Bengio, 2012) Batch sizes that are large can lead to poor generalization. Epoch is set to 50. This is done to prevent the model from overfitting and make it computationally less expensive.

4.5.3 Tuning hyperparameters

The following step is tuning the hyperparameters. Tuning the hyperparameters is the stage that turns the baseline feedforward neural networks into deep feedforward neural networks. This is because of the added depth of the model by the tuning.

Tuning is done by using the Keras Tuner (O'Malley, et al., 2019). The Keras Tuner is a library to tune hyperparameters of Keras models. Before tuning can start, a model is needed to be built in which the desired hyperparameters are defined. In this research, the number of layers, number of nodes, activation function, drop rate, regularization rate, and the learning rate are optimized. The range of layers is set between 2 and 10. This is to reduce overfitting. When applying many layers, the model might overfit and the gradients might vanish. The number of neurons is set between 32 and 128 with a step size of 32. The chosen activation functions are ReLU, Tanh and LeakyReLU. LeakyReLU is similar to ReLU. The difference is that LeakyReLU does not assign negative values to 0 like ReLU does. Instead, LeakyReLU returns negative values close to 0. This prevents the neurons from dying. A neuron 'dies' when its output is always 0 (Zhang, Zou, & Shi, 2017). This is common with ReLU. This is why it is a good practice to include LeakyReLU when tuning the hyperparameters. The drop rate goes from 0.0 to 0.5 with increments of 0.1. The used regularization method is l2. The rates for this method are 0.01,0.001,0.1,0.005,0.05.

After the building of the tuner, the tuner gets compiled. During compiling the learning rate, optimizer, loss function and measuring metrics get chosen. the learning rate is set between 1e-2 and 1e-4. Learning rates are the rates at which the gradient is reached. The smaller the gradient, the slower gradient descent is. When the gradient is increased, the model has a possibility of overshooting the minimum. When the learning rate is too small, the gradient can get stuck in a local minimum. This is why it is important to experiment with different learning rates. The chosen optimizer is Adam because Adam gives fast convergence. The loss function is binary cross entropy as this research deals with a binary classification problem. Lastly, the measuring metrics are accuracy and AUC. Accuracy is chosen to make comparisons with previous studies that used accuracy as a metric. AUC is chosen as this metric takes the imbalanced nature of the target class into account.

When the model is built, the tuner can be initialized. The tuning options are Random Search, Bayesian Optimization and Hyperband. Hyperband. For this research Bayesian Optimization is chosen. Bayesian Optimization calculates the probability of the function for best hyperparameters beforehand and updates this estimate by using more samples (Wu, et al., 2019). The objective of this tuner is set to maximize the AUC and, finally, the max_trials parameter is set to 5. This parameter controls the number of trials with each search.

After building the tuner, the tuner is applied to all the training sets. The tuner then returns the optimal hyperparameters for each training set. The hyperparameters are subsequently used to fit new models. This resulted into eight new models with the non-sampled, random under sampled, random oversampled and SMOTE- data, for the dataset without feature selection and the dataset with mutual information feature selection.

4.6 Implementation

The model is implemented by using Python 3.8.5. The libraries Pandas, NumPy, Seaborn, Scikit-learn, Imblearn, Matplotlib, TensorFlow, Keras and the Keras Tuner.

4.7 Evaluation criteria

4.7.1 Confusion Matrix

A confusion matrix is a table used to interpret the performance of a classifier on the test set. The table provides the amount of correct and incorrect predictions made by the classification model. There are four possible combinations of values in a confusion matrix: True positives (TP), false positives (FP), false negatives (FN) and true negatives (TN) (Luque, Carrasco, Martin, & Heras, 2019). These combinations make it possible to derive various metrics to measure the model. The most intuitive metric is accuracy. This metric explains the correct predictions over the total predictions. The limitation of this metric is that it only correctly represents the model's prediction when the target label is balanced.

Previous deep learning studies on the Cell2Cell dataset only reported the accuracy of the model. To be able to compare results from this thesis to these previous studies, accuracy is reported.

4.7.2 Area Under the Receiving Operator Curve (AUROC)

The AUROC will be used to assess the classification performance of the models. This is a proper metric for this dataset because it takes the imbalanced nature of the data into account. The AUROC contains two measures. The ROC and the AUC. The ROC curve is a two dimensional curve and the AUC is a single number derived from the ROC. ROC is a curve for a classifier with a binary output. The ROC derives two types of metrics from the confusion matrix; True positive rate (TPR) and false positive rate (FPR). The ROC curve tries to maximize TPR and minimize FPR. ROC captures the distribution of the classes effectively without sacrificing TPR for FPR and vice versa.

5. Results

5.1 No feature selection

Performance of models before tuning hyperparameters:

Sampling method	Accuracy	AUC
No sampling	66.17	0.59
Random undersampling	55.89	0.56
Random oversampling	58.41	0.56
SMOTE	58.64	0.57

Performance of models after tuning hyperparameters

Sampling method	Accuracy	AUC
No sampling	69.85	0.61
Random undersampling	52.89	0.57
Random oversampling	61.37	0.56
SMOTE	60.33	0.57

5.1.1 Unsampled model

The first model is the unsampled model. No sampling techniques have been applied to this model. This means that the target classes are imbalanced. Before tuning, the model performed with an accuracy of 66.17 and an AUC of 0.59. Looking at the validation loss in figure 5.1, it appears that the model tends to overfit at around 5 epochs.

After tuning the performance of the model changed to an accuracy of 69.85 and an AUC of 0.61. The validation loss plot in figure 5.2 shows that the model started to overfit at around 10 epochs.

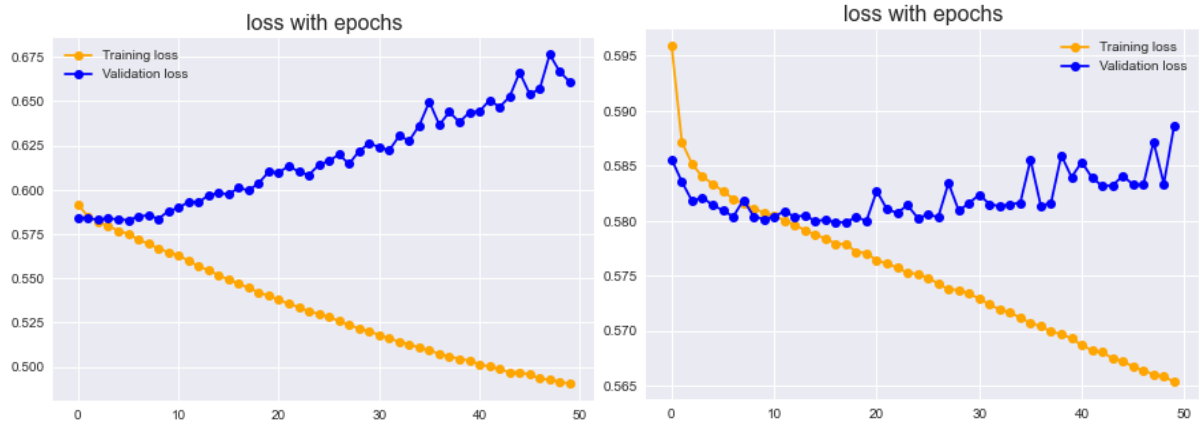

Figure 5.1: validation loss for unsampled model without feature selection before tuning (left) and after tuning (right).

5.1.2 Random under sampled model

This model used the random undersampling technique. Before tuning, the model performed with an accuracy of 55.89 and an AUC of 0.56. This method under sampled the majority class. This led to smaller data to train the model with. The result of not having sufficient training data is that the model cannot train well. This can be seen on the left plot of figure 5.2. Training loss represents how well training data is fitted the model whereas validation loss represents how well new data fits the model. There is a gap between the training loss and the validation loss. This gap shows that the model did not generalize well to new data and therefore underfitted. After tuning the gap between training and validation loss is closed. This is not a sign of better performance because the performance is similar to before tuning with an accuracy of 55.44 and AUC of 0.56.

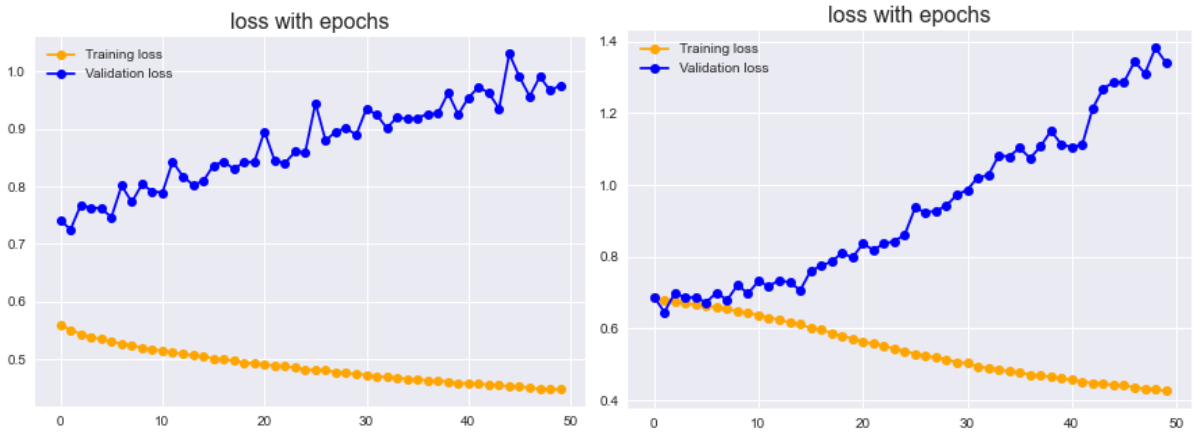

Figure 5.2: validation loss for randomly under sampled model without feature selection before tuning (left) and after tuning (right).

5.1.3 Random oversampled model

This model used the random oversampling technique. Before tuning, the model performed with an accuracy of 58.41 and an AUC of 0.56. This method oversampled the minority class. The validation loss for the randomly oversampled model in figure 5.3 shows a gap similar to the one in figure 5.2. This is unexpected as the model is provided with more data than in the randomly under sampled models. After tuning the behavior is again similar to the behavior in figure 5.2. The performance of the model after tuning turned to an accuracy of 61.37 and an AUC of 0.56. The AUC did not increase but the accuracy did. This is remarkable because the behavior in the plots appear to behave similar. To further assess this, the confusion matrices of the untuned and the

tuned models are looked into. Figure 5.4 shows that the untuned model has 687 true positives whereas the tuned model predicted 584 true positives. This shows that the accuracy is not a reliable measure when dealing with class imbalance.

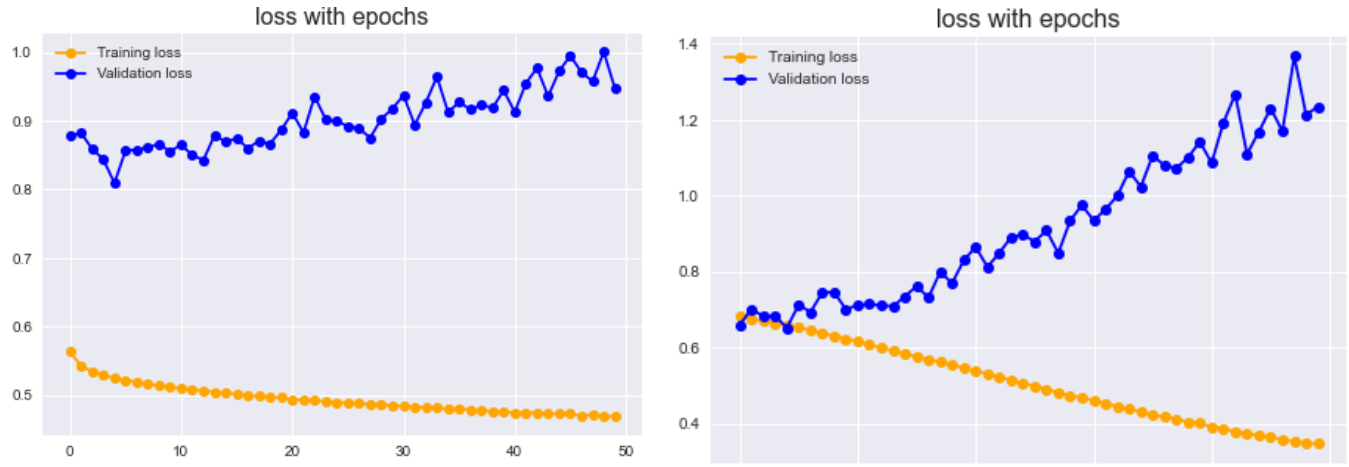

Figure 5.3: validation loss for randomly oversampled model without feature selection before tuning (left) and after tuning (right).

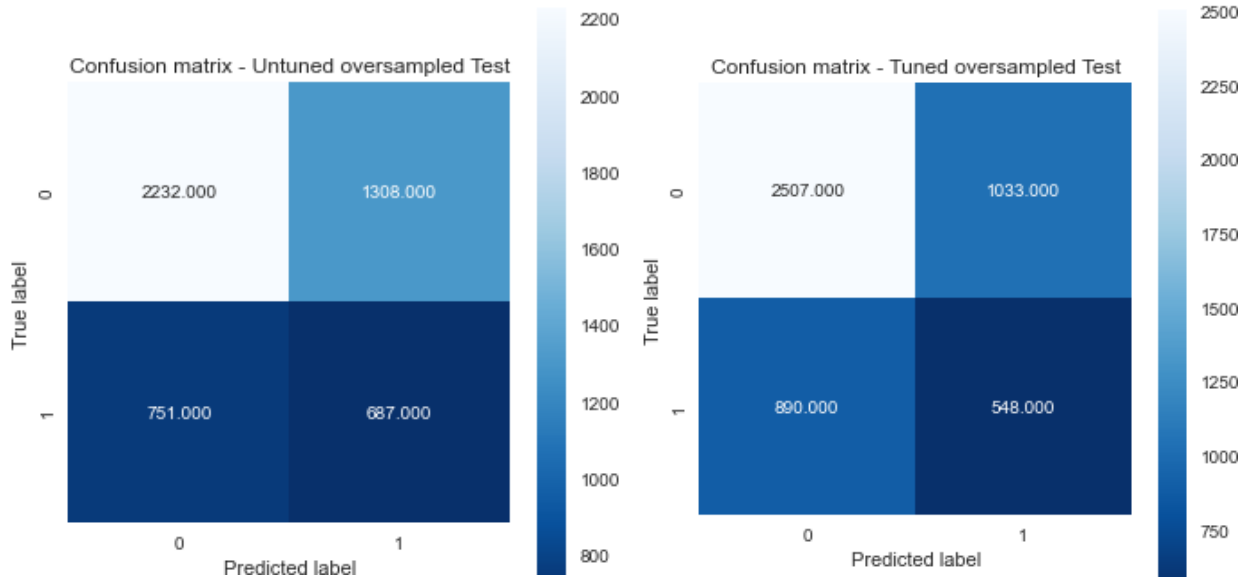

Figure 5.4: Confusion matrices for randomly oversampled model without feature selection before tuning (left) and after tuning (right).

5.1.4 SMOTE model

This model used SMOTE. Before tuning, the model performed with an accuracy of 58.64 and an AUC of 0.57. The left plot in figure 5.5 shows the validation loss of the untuned model. This is again, a similar situation to the under sampled and oversampled validation loss plots. Tuning of the model leads to an accuracy of 60.33 and an AUC of 0.57. This is the same situation as the randomly oversampled data. The accuracy appears to be higher, but in reality, the model performs worse on the minority class. The accuracy is higher because of a better performance on the majority class.

Figure 5.5: validation loss for SMOTE model without feature selection before tuning (left) and after tuning (right).

5.2 Mutual information feature selection

Performance of models before tuning hyperparameters.

Sampling method	**Accuracy**	**AUC**
No sampling	70.45	0.62
Random undersampling	58.90	0.62
Random oversampling	56.07	0.62
SMOTE	57.85	0.61

Performance of model after tuning hyperparameters:

Sampling method	**Accuracy**	**AUC**
No sampling	69.24	0.60

Random undersampling	69.49	0.62
Random oversampling	69.41	0.61
SMOTE	70.03	0.62

5.2.1 unsampled model

The first model is the unsampled model. Before tuning, the model performed with an accuracy of 70.45 and an AUC of 0.62. The validation loss plot in figure 5.6 shows that the model tends to overfit at around 10 epochs. After tuning, the performance decreased to an accuracy of 69.24 and an AUC of 0.60. This means that the default parameters of the baseline outperformed the parameters after tuning.

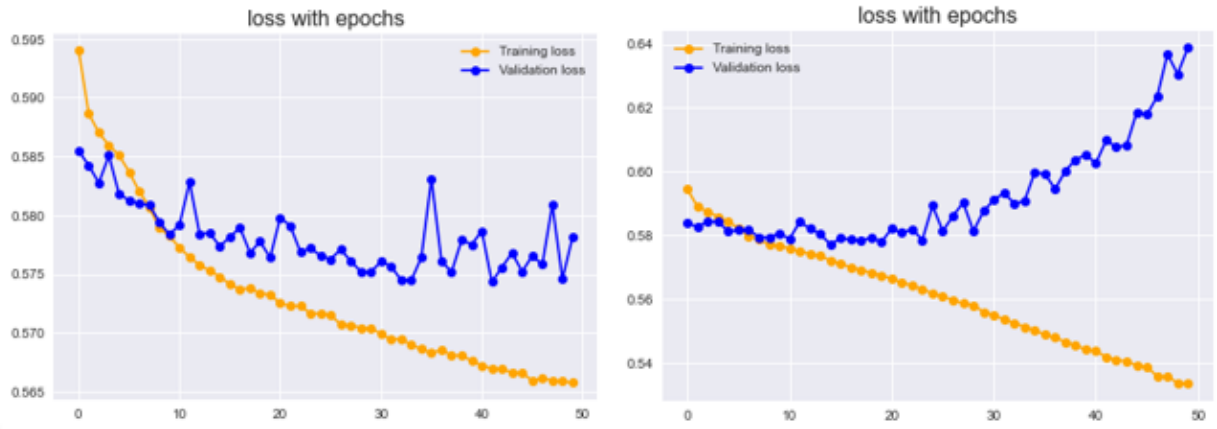

Figure 5.6: Validation loss for unsampled model without feature selection before tuning (left) and after tuning (right).

5.2.2 Random under sampled model

This model used random undersampling. Before tuning, the model performed with an accuracy of 58.90 and an AUC of 0.62. The validation loss in figure 5.7 seems to show lots of oscillation in the loss. This appears to be the case, but when looking at the y-axis the range of oscillation is between 0.64 and 0.7. After tuning the accuracy of the model increases, but the AUC stays the same. The validation plot after tuning shows a stable validation loss.

After tuning the model's accuracy is 69.49 and the AUC remains 0.62. Because the AUC is not changing, the change in accuracy is only for the negative class of non-churn.

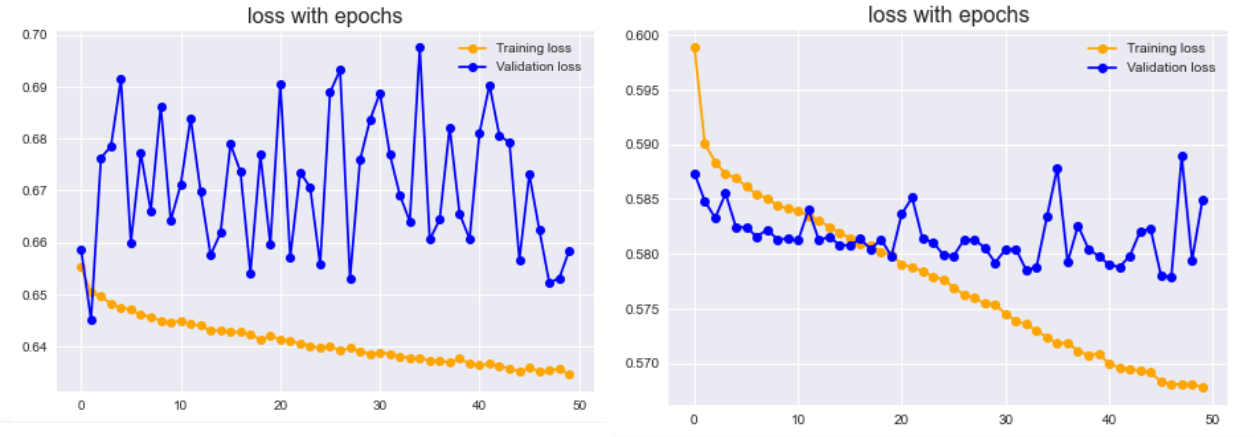

Figure 5.7: Validation loss for under sampled model without feature selection before tuning (left) and after tuning (right).

5.2.3 Random oversampled model

This model random oversampling. Before tuning, the model performed with an accuracy of 56.07 and an AUC of 0.62. The validation loss in figure 5.8 shows that there is lots of oscillation. This means that the model cannot generalize well with the given parameters. After tuning, the oscillation is decreased as can be seen on the plot on the right of figure 5.8. The tuner changed the accuracy of the model to 69.41 but decreased the AUC to 0.61. This means that the tuner made the model train better on the majority class, but not on the minority class.

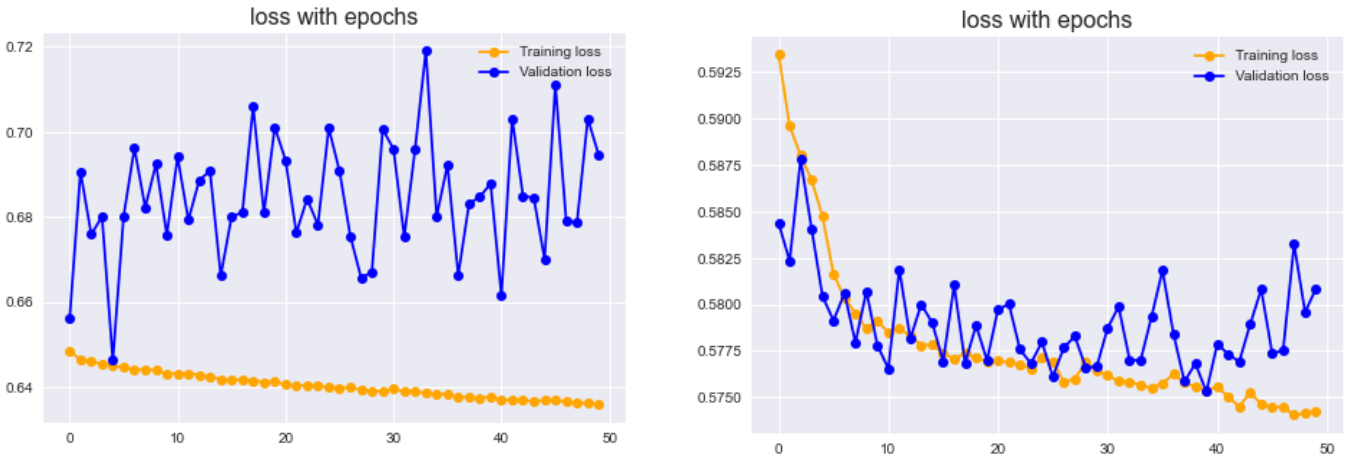

Figure 5.8: Validation loss for oversampled model without feature selection before tuning (left) and after tuning (right).

5.2.4 SMOTE model

This model used SMOTE. Before tuning, the model performed with an accuracy of 57.85 and an AUC of 0.61. The validation plot in figure 5.9 shows very similar behavior with the randomly oversampled model. This means that there are too many parameters to generalize well with SMOTE before tuning. After tuning, the oscillation decreases. The model's accuracy increased like in the randomly oversampled model. However, this time the AUC also increased. The model's performance changed to 70.03 and 0.62 for accuracy and AUC, respectively.

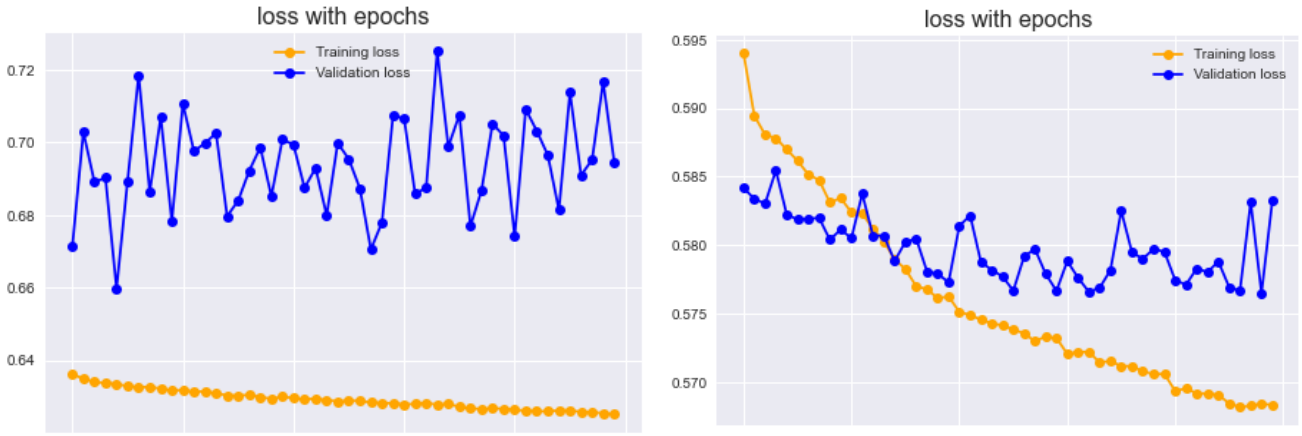

Figure 5.9: Validation loss for under sampled model without feature selection before tuning (left) and after tuning (right).

5.3 Summary of the results

Without feature selection

The best performing model without feature selection is the model without sampling methods after the tuning of the hyperparameters. This model performed with an accuracy of 69.85 and an AUC of 0.61. It is interesting to note that the models without sampling methods performed best both before and after the tuning of hyperparameters.

With mutual information feature selection

The best performing model with mutual information feature selection is the model with no sampling methods before tuning. The accuracy of this model is 70.45 and the AUC is 0.62. The models after tuning all performed within the range of 69.24 and 70.03 for accuracy and within the range of 0.60 and 0.62 for AUC. This means that they performed similar. The AUC for all the

models with feature selection, before and after tuning, are similar. The accuracies are not. The accuracy of the resampled models increased after tuning. This means that the tuner made the models perform better on the majority class, but not on the minority class.

6. Discussion and conclusion

The objective of this thesis was to improve the performance of deep learning techniques on the Cell2Cell dataset. The Cell2Cell dataset is often researched and has often led to performances that are not promising. This is likely due to the different scales, non-normally distributed data, missing values, and outliers. This chapter answers the sub-questions and the main question of this thesis. That were formulated in the introduction.

Sub questions

SQ1: To what extent does feature selection improve performance in predicting customer churn in the Cell2Cell dataset?

The best performing model without feature selection performs with an accuracy of 69.85 and an AUC of 0.61. The best performing model with mutual information feature selection leads to an accuracy of 70.45 and an AUC of 0.62. This indicates that the accuracy increased significantly while the AUC did not. When accuracy increases and AUC not, the improvement in performance is related to the majority class. In this case, there is no improvement for the minority class. This means that feature selection improved performance on predicting customers that will not churn but decreased in performance for customers that will churn.

SQ2: 2. *Which sampling method provides the best performance?*

The expectation of this thesis was that SMOTE would outperform all the other sampled and unsampled training sets. This did not happen because the best performing model is the unsampled model. Comparing this to previous literature, the performance slightly decreased (Umayaparvathi & Iyakutti, 2017). The baseline model is performing similar but less accurate. The previous study had an accuracy of 72.4 and the proposed baseline model scored a 70.5. The AUC of the proposed model is 0.62.

It is notable that the best performing model is the model without resampling methods applied. This was unexpected because the reason resampling is applied is to improve the performance. The expectation was that SMOTE would outperform the other methods as in the literature (Chawla, Bowyer, Hall, & Kegelmeyer, 2002) (Hartati, Adiwijaya, & Bijaksana, 2018), (Wu, Yau, Ong, &

Chong, 2021). SMOTE however, performed with an accuracy of 70.04 and an AUC of 0.62. This is similar to the best performing model of with an accuracy of 70.45 and an AUC of 0.62, but it shows that SMOTE did not improve the performance.

SMOTE has its limitations. It does not perform well on oversampling at the decision border. Moreover, SMOTE has the problem of overgeneralization for the minority class without taking the majority class into consideration (Maciejewski & Stefanowski, 2011). These limitations could have contributed to the poor performance on the SMOTE model. In future work, other variations of SMOTE can be used. An example is borderline-SMOTE. In borderline-SMOTE, the samples of the minority class on the decision border are oversampled (Han, Wang, & Mao, 2005).

Research question

RQ: *To what extent can deep learning improve customer churn prediction when using sampling methods?*

The results of this thesis show that applying sampling methods to improve performance, is not effective. The best performing resampled training set is SMOTE with an accuracy of 70.04 and an AUC of 0.62. This performance is not the overall best performance. The overall best performance was achieved by the model without resampling. The differences in performance are not significant and show that resampling does not improve the performance of the model. The AUC for the unsampled methods outperforms the resampled methods three out of four times.

A 2022 paper that was published after the proposal of this thesis conducted a similar experiment on the Cell2Cell dataset (Fujo, Subramanian, & Khder, 2022). This research used deep neural networks to predict customer churn and reached an accuracy and AUC of 79.38. These are significantly higher results than the results proposed in this paper and in the 2017 and 2019 papers. The difference between this thesis and the 2022 paper is in the preprocessing of the data, feature selection and validation. This thesis removed outliers, used dummy coding, and dropped all the rows with missing data. The 2022 paper did not remove outliers, used label encoding and imputed the missing data by the residual mean. Moreover, the feature selection used in this thesis is mutual information feature selection, whereas the 2022 paper used Lasso regression to select the features. The paper applied random oversampling to the Lastly, this thesis used hyperparameter tuning on

the validation set to tune the hyperparameters. The 2022 paper used 10-fold cross validation without tuning hyperparameters. A similarity between this thesis and the paper is the method of oversampling. Both this thesis and the paper use random oversampling to oversample the minority class. The problem with the application of random oversampling in the 2022 paper is that oversampling is done for the entire dataset. Not only on the training set. When the whole dataset is oversampled, data leakage happens. When data leakage happens, performance metrics appear to be higher. This is not correct however, because the model is partially built on testing data. This will lead to high performances while the model did not perform well, it only remembered the data it was built on.

Future research

Future research can focus on state-of-the-art class imbalance techniques. Generative Adversarial Networks (GAN) have been used to research the class imbalance problems in CCP (Li & Xie, 2020). A GAN learns through a competitive process of a Generator and a Discriminator. The Generator tries to fool the Discriminator. The Discriminator tries to not get fooled by distinguishing the generated data from the real data (Google Developers, 2021). This type of sampling data has not been used in the telecom industry yet, but it has been used in the banking industry before. A 2020 study proposed GAN as an alternative to random oversampling and SMOTE by comparing them. The conclusion was that GAN's outperform SMOTE and random oversampling with f-mean and accuracy values. GAN's showed an improvement to the synthetic sampling in SMOTE. The generated data by the GAN was more targeted. (Li & Xie, 2020)

References

Ahmed, U., Khan, A., Khan, S., Basit, A. H., & Lee, Y. (2019). *Transfer Learning and Meta Classification Based Deep Churn.* arXiv.

Amin, A., Anwar, S., N. M., Howard, N., Qadir, J. H., & Hussain, A. (2016). Comparing Oversampling Techniques to Handle the Class Imbalance Problem: A Customer Churn Prediction Case Study. *IEEE Access PP 99*, 7940-7957.

Bengio, Y. (2012). Practical Recommendations for Gradient-Based Training of Deep Architectures. *ArXiv* (pp. 1-33). Cornell University.

Capgemini. (2015, August 15). *Looking at Telecom industry trends by 2020 and beyond*. From Capgemini.com: https://www.capgemini.com/2015/08/looking-at-telecom-industry-trends-by-2020-and-beyond/

Chawla, N., Bowyer, K., Hall, L., & Kegelmeyer, W. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*.

Chen, B., Chen, H., & Li, M. (2021). Feature Selection Based on BP Neural Network and Adaptive Particle Swarm Algorithm. *Hindawi - Mobile Information Systems* , 1-11.

Diaz, G., Fokoue, A., Nannicini, G., & Samulowitz, H. (2017). An effective algorithm for hyperparameter optimization of neural networks. *IBM Journal of Research and Development*.

Dolatabadi, S., & Keynia, F. (2017). Designing of Customer and Employee Churn Prediction Model Based on Data Mining Method and Neural Predictor. *The 2nd International Conference on Computer and Communication Systems* (pp. 74-77). Beijing: IEEE.

Fujo, S., Subramanian, S., & Khder, M. (2022). Customer Churn Prediction in Telecommunication Industry Using Deep Learning. *Information Sciences Letters*.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Feedforward Networks.* Massachusetts: Massachusetts Institute of Technology: MIT.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning.* Cambridge: The MIT Press.

Google. (2021, May 24). *Introducing Convolutional Neural Networks*. From developers.google: https://developers.google.com/machine-learning/practica/image-classification/convolutional-neural-networks

Gutierrez, D. (2014, November 26). *Ask a Data Scientist: Data Leakage*. From inside BigData: https://insidebigdata.com/2014/11/26/ask-data-scientist-data-leakage/

Han, H., Wang, W., & Mao, B. (2005). Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. *International Conference on Intelligent Computing* (pp. 878-887). Hefei: ICIC.

Hartati, E., Adiwijaya, & Bijaksana, M. (2018). Handling imbalance data in churn prediction using combined SMOTE and RUS with bagging method. *Journal of Physics: Conference Series* (pp. 1-10). Vancouver: IOP Publishing .

Hockenmaier, J. (2013). *Introduction to Machine Learning.* From https://courses.engr.illinois.edu/: https://courses.engr.illinois.edu/cs446/sp2015/Slides/Lecture06.pdf

Hu, X., Lingyang, P., Jian, L., Weiqing, B., & Jiang. (2021). Model Complexity of Deep Learning: A Survey. *arXiv*.

Imb Learn. (2021). *Over-sampling*. From Imbalanced Learning: https://imbalanced-learn.org/stable/over_sampling.html

Jangir, H. (2021, January 12). *Important things to Keep in Mind during Data Preprocessing*. From Analytics Vidhya: https://www.analyticsvidhya.com/blog/2021/01/5-important-things-to-keep-in-mind-during-data-preprocessing-specific-to-predictive-models/

Jayalakshmi, T., & A, S. (2011). Statistical Normalization and Back Propagation for Classification. *International Journal of Computer Theory and Engineering Vol 3, No. 1*, 89-93.

Kaggle. (2018). *Telecom Churn (Cell2Cell)*. From Datasets: https://www.kaggle.com/jpacse/datasets-for-churn-telecom

Kim, H., Jo, N., & Shin, K. (2016). Optimization of cluster-based evolutionary undersampling for the artificial neural networks in corporate bankruptcy prediction. *Expert Systems With Applications 59*, 226-234.

Li, B., & Xie, J. (2020). Study on the Prediction of Imbalanced Bank Customer Churn Based on Generative Adversarial Network. *Journal of Physics: Conference Series*.

Luque, A., Carrasco, A., Martin, A., & Heras, A. (2019). The impact of class imbalance in classification performance metrics based on the binary confusion matrix. *Pattern Recognition*, 216-231.

Maciejewski, T., & Stefanowski, J. (2011). Local neighborhood extensions of SMOTE for mining imbalanced data. *IEEE Symposium on Computational Intelligence and Data Mining (CIDM).* Poznan: IEEE.

Menardi, G., & Torelli, N. (2014). Training and assessing classification rules with imbalanced data. *Data Mining and Knowledge Discovert 28*, 92-122.

Mishra, A., & Reddy, U. (2017). A Novel Approach for Churn Prediction Using Deep Learning on computational intelligence and computing research. *IEEE international conference on* (pp. 1-4). IEEE.

O'Malley, Tom, Bursztein, Elie, Long, & James. (2019). *KerasTuner*. From Keras: https://github.com/keras-team/keras-tuner

Pandas. (2021). *pandas.get_dummies*. From Pandas: https://pandas.pydata.org/docs/reference/api/pandas.get_dummies.html

Pandas. (2021). *pandas.read_csv*. From Pandas: https://pandas.pydata.org/docs/reference/api/pandas.read_csv.html

Reichheld, F. (2001). *Prescription for cutting costs.* Boston: Bain & Company.

Saheed, Y., & Hambali, M. (2021). Customer Churn Prediction in Telecom Sector with Machine Learning and Information Gain Filter Feature Selection Algorithms. *International Conference on Data Analytics for Business and Industry* (pp. 208 - 213). Sakheer: University of Bahrain.

Scikit Learn. (2021). *sklearn.feature_selection.mutual_info_classif*. From scikit learn: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_classif.html

Scikit Learn. (2021). *sklearn.feature_selection.SelectKBest*. From Scikit Learn: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html

Scikit Learn. (2021). *sklearn.model_selection.train_test_split*. From scikit learn: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

Scikit Learn. (2021). *sklearn.preprocessing.MinMaxScaler*. From Scikit Learn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

Shankar, K., Zhang, Y., Liu, Y., Wu, L., & Chen, C. (2020). Hyperparameter Tuning Deep Learning for Diabetic Retinopathy Fundus Image Classification. *IEEE Access*, 118164-118173.

Shuyu, Z. (. (2020). *AUC Maximization in Deep Neural Network Learning for Imbalanced Classificaiton Problems.*

Talathi, S., & Vartak, A. (2016). Imprving Performance of Recurrent Neural Network with ReLU Nonlinearity. *ICLR* (pp. 1-12). San Diego: Qualcomm Research.

TensorFlow. (2021). *Module: tf.keras.metrics*. From TensorFlow: https://www.tensorflow.org/api_docs/python/tf/keras/metrics

Umayaparvathi, V., & Iyakutti, K. (2017). Automated Feature Selection and Churn Prediction using Deep Learning Models. *International Research Journal of Engineering and Technology (IRJET)* , 1846-1854.

Vafeiadis, T., Diamantaras, K., Sarigiannidis, G., & Chatzisavvas, K. (2015). A comparison of machine learning techniques for customer churn prediction. *Simulation Modelling Practice and Theory 55*.

Wu, J., Chen, X., Zhang, H., Xiong, L., Lei, H., & Deng, S. (2019). Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization. *JOURNAL OF ELECTRONIC SCIENCE AND TECHNOLOGY, VOL. 17*, 26-39.

Wu, S., Yau, W.-C., Ong, T.-S., & Chong, S.-C. (2021). Integrated Churn Prediction and Customer Segmentation Framework for Telco Business. *IEEE Access*, 62120.

Zebari, R., Abdulazeez, A., Zeebaree, D., & Saeed, J. (2020). A Comprehensive Review of Dimensionality Reduction Techniques for Feature Selection and Feature Extraction. *Journal of Applied Science and Technology Trends*, 56-70.

Zhang, X., Zou, Y., & Shi, W. (2017). Dilated convolution neural network with LeakyReLU for environmental sound classification. *International Conference on Digital Signal Processing (DSP).* London: IEEE.