NOVA IMS Information Management School

MEGI

Mestrado em Estatística e Gestão de Informação

Master Program in Statistics and Information Management

A COMPARATIVE STUDY OF TREE-BASED MODELS FOR CHURN PREDICTION: A CASE STUDY IN THE TELECOMMUNICATION SECTOR

Ibrahem Hamdy Abdelhamid Kandel

Dissertation presented as partial requirement for obtaining the Master's degree in Statistics and Information Management

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa

NOVA IMS Information Management School

MGI

Mestrado em Gestão de Informação
Master Program in Information Management

A COMPARATIVE STUDY OF TREE-BASED MODELS FOR CHURN PREDICTION: A CASE STUDY IN THE TELECOMMUNICATION SECTOR

Ibrahem Hamdy Abdelhamid Kandel

Dissertation presented as partial requirement for obtaining the Master's degree in Statistics and Information Management

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

2018

Title: A COMPARATIVE STUDY OF TREE-BASED MODELS FOR CHURN PREDICTION: A CASE STUDY IN THE

Student
full name Ibrahem Kandel

MEGI

2018

Title: A COMPARATIVE STUDY OF TREE-BASED MODELS FOR CHURN PREDICTION: A CASE STUDY IN THE

Student
full name Ibrahem Kandel

MGI

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

A COMPARATIVE STUDY OF TREE-BASED MODELS FOR CHURN PREDICTION: A CASE STUDY IN THE TELECOMMUNICATION SECTOR

by

Ibrahem Hamdy Abdelhamid Kandel

Dissertation presented as partial requirement for obtaining the Master's degree in Information Management, with a specialization in Market Research and CRM

Advisor: Professor Roberto Henriques

November 2018

ACKNOWLEDGEMENTS

I would like to express gratitude towards my master's thesis advisor, Professor Roberto Henriques, for providing guidance on my thesis.

ABSTRACT

In the recent years the topic of customer churn gains an increasing importance, which is the phenomena of the customers abandoning the company to another in the future. Customer churn plays an important role especially in the more saturated industries like telecommunication industry. Since the existing customers are very valuable and the acquisition cost of new customers is very high nowadays. The companies want to know which of their customers and when are they going to churn to another provider, so that measures can be taken to retain the customers who are at risk of churning. Such measures could be in the form of incentives to the churners, but the downside is the wrong classification of a churners will cost the company a lot, especially when incentives are given to some non-churner customers. The common challenge to predict customer churn will be how to pre-process the data and which algorithm to choose, especially when the dataset is heterogeneous which is very common for telecommunication companies' datasets. The presented thesis aims at predicting customer churn for telecommunication sector using different decision tree algorithms and its ensemble models.

KEYWORDS

Churn prediction; Predictive modelling; Decision trees; Data mining; Customer churn; CRM; Random Forests; Decision Trees; Ensemble; Bagging

INDEX

LIST OF ABBREVIATIONS AND ACRONYMS

ANN Artificial Neural Networks
CCP Customer Churn Prediction
CRM Customer Relationship Management
DT Decision trees
RF Random Forests
GBM Gradient Boosting Machine
SVM Support vector machines
MNAR Missing not at random
MAR Missing at random
MCAR Missing completely at random
KNN K-nearest neighbor algorithm
SMOTE Synthetic Minority Oversampling Technique
SGBT Stochastic Gradient Boost Trees
AdaBoost Adaptive Boosting
PCA Principal component analysis
MI Mutual Information
LMT Logistic Model Trees
OOB Out of bag
AUC Area Under the ROC Curve
ROC Receiver operating characteristic
KDD Knowledge discovery in databases
CART Classification and Regression Trees

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

1. INTRODUCTION

In today's business, a lot of advertising and offers appears everywhere in television, internet and billboards which puts the customer in a lot of offers like free subscription or discounts, so many customers change their service providers to get the benefit of these offers. The term churn is now well known in today's businesses, customer churn is the scenario where the customer loses interest in continuing his business in the company and seeks another company to fulfill his needs, this can have a huge impact on the profits of the company (Buckinx & Van Den Poel, 2005). Right now, it's one of the most important topics in saturated business, especially in the rapidly growing telecommunication sector, knowing that the cost of acquiring new customers is as high as ten times to retain the existing customers (Lu & Ph, 2002), mainly due to fierce competition between telecommunication companies. It's very important to the companies to reduce the revenue loss due to the leaving customers, the reasons for churn differs from company to another and from customer to another. Mainly there are two types of churners: the involuntary churners "forced to leave" and the voluntary churners, the involuntary churners happens when the company decide to end the contract due to many reasons like the case when they don't satisfy their part of agreement between them and the company, while the voluntary churners is the type that most of the researches are done on, because these customers are mainly the most profitable customers and the company wants to retain them as their customers. The reasons for voluntary churns differs a lot from one customer to another, mainly in telecommunication it's due to higher cost, quality, lack of features, privacy issues and others (Sharma & Sachdeva, 2017).

The Customer churn prediction can be defined as assigning a probability to leave the company next to each customer in the company dataset, to make it viable to the firm to indicate which customer has the highest propensity to churn, then for the company to set a threshold to indicate which probability to churn they want to tackle, then the customers clusters whom crossed this threshold will be targeted by a specially designed marketing plan to increase their retention. Different approaches was introduced to increase customer retention (Fig.1), while targeting all of the churners clusters will be Monterey exhausting, that's why the companies must select wisely which of these customers to target, instead of targeting every one also known as uplift modeling (Coussement, Lessmann, & Verstraeten, 2017).

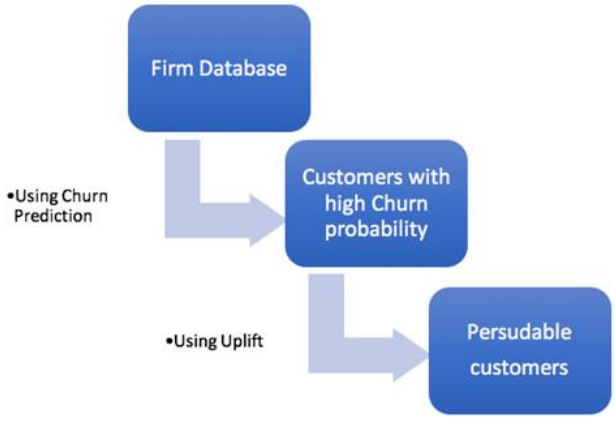

Fig.1.1 Customer Churn Model

The steps of the process will be: using predictive modeling to detect which customer has the highest probability to churn, then by using uplifting modeling to detect which customers can be targeted with the retention plan. The customer churn topic is discussed many times in telecommunication sector (Mahajan, Misra, & Mahajan, 2015), many kinds of research were done to know what are the optimization algorithms to use to expect when the customers are going to churn before they do, knowing that about 30 percent of the customers are going to churn per year (Lu & Ph, 2002). To acquire a new customer, the company is going to

spend ten times more than to retain the already existing customers, so by now to retain the existing customers is more profitable than acquiring new customers (Lu & Ph, 2002). By the help of the recent techniques in data mining, companies succeed in having insights about their customer behavior either by using the usual Sociodemographic variables or by using more advanced variables like their customer call center data (Hadden, Tiwari, Roy, & Ruta, 2006).

There are many machine learning techniques being adopted in Customer churn like Logistic regression, Decision trees, Artificial neural networks and Support vector machines (Hadden, Tiwari, Roy, & Ruta, 2007; Mahajan, Misra, & Mahajan, 2015). Now in the era of big data a lot of researchers try to draw the light on the importance of inclusion of the customer data from the social platforms to be included in the customer behavior analysis (Bi, Cai, Liu, & Li, 2016; Liu & Zhuang, 2015; Singh & Singh, 2017). Because datasets of the companies are usually containing redundant values, missing values among others which decreases any model predictive power, that's where the data preprocess is vital in the process of building any model. The challenges that must be treated before implementing the model is imbalance problem, missing values, curse of dimensionality and others The imbalance problem can be defined as having one class outnumbered the other class, in the case of customer churn its usually that the non-churner customers' numbers are much higher than the churner customer (Burez & Van den Poel, 2009), which will have a huge influence on most of machine learning algorithms. The challenge of having degenerative features which are features with zero variance, meaning that there is no change in the values of that features, which makes it useless in the prediction model also it will create a numerical difficulty.

The occurrences of missing data in any dataset brings a huge impact on the performance of any machine learning algorithm, many algorithms had been used to tackle the challenge of missing values (Noh, Kwak, & Han, 2004), but still the uncertainty of these values will result in low predictive power of the model because of the bias created by the new values. The curse of dimensionality can be treated by using a proper feature reduction techniques, which mainly falls into three categories: filter methods, wrapper methods and embedded methods (Raza & Qamar, 2017). Identify the true churner from the retained customer is crucial, because of the high cost of retaining a non-churner customer, which consider resource wasting so the issue of the model accuracy is very important (Khan, Jamwal, & Sepehri, 2010), to assess the quality of the model can be detected by using the AUC of the ROC curve (Vuk, 2006), which is commonly used in the classification problems.

The model proposed in this thesis is decision tree-based models, the importance of using decision tree models is that they are fast to compute and highly interpretable. Which have a great advantage for companies to not only estimate the propensity of each customer to churn, but also to understand which features are important to retain the customers in the future, also it has a great advantage over many machine learning algorithms, which the decision trees can deal with heterogeneous data, defined as data that contain numerical and categorical features. In this paper, the used dataset is the KDD Cup 2009 small dataset (Guyon, Lemaire, Vogel, et al., 2009), which was provided by the French telecommunication company Orange for predicting three binary classifications: churn, upselling and cross-selling. The main objective of this thesis is to evaluate and analyze of performance of tree-based models to predict the churn probability of customers for telecommunication sector, the proposed machine learning models are decision trees, Random forest and Gradient Boost Machine. The results will be assessed by using AUC of the ROC curve as proposed by the evaluating committee of the challenge.

The remainder of the paper is organized as follows: in section 2 discussion about the literature review made on papers presented using the KDD 2009 dataset. In section 3 the data pre-process step. In section 4 the models proposed in the study. In section 5 the results of each model. In section 6 Conclusions. In section 7 future studies.

2. LITERATURE REVIEW

The classification task of customer churn has been studied in different sectors where its applicable like: banking sector (Ali & Arıtürk, 2014; Mohanty & Naga Ratna Sree, 2018; Şen & Bayazıt, 2015), insurance companies (Morik & Köpcke, 2004; R. Zhang, Li, Tan, & Mo, 2017; W. Zhang, Mo, Li, Huang, & Tian, 2016) online social networks (Ngonmang, Viennet, & Tchuente, 2012; Oentaryo, Lim, Lo, Zhu, & Prasetyo, 2012; Óskarsdóttir et al., 2016), Churn in online gaming (Borbora & Srivastava, 2012; Castro & Tsuzuki, 2015; Liao, Chen, Liu, & Chiu, 2015) and telecommunication sector which is the task of this paper. The literature review section will be divided into two parts: The first part is a brief overview of previous studies about customer churn in telecommunication sector and the second part is a brief overview of previous studies done on KDD 2009 dataset.

2.1. CHURN PREDICTION

In the study done by Pendharkar (2009) the author proposed two genetic-algorithm based on neural network to predict customer churn for telecommunication sector, to their knowledge that was the first paper to implement genetic algorithm to predict churn for telecommunication sector. They used a real-world dataset with 195,956 customers and four variables then They split the dataset into 70% for training and 30% for testing. The first genetic algorithm was used for prediction and the second was used to increase the performance of the first one. They compared the performance of their algorithm to a statistical z-score model using many evaluating criterions including ROC curve. Their findings were that the medium sized neural networks with number of hidden layers equal to twice the number of variables performed better than the statistical z-score model, but it was more computationally expensive than the statistical model.

In the study done by Owczarczuk (2010) the author tested some supervised machine learning algorithm to predict churn for prepaid clients of a Polish telecommunication company. The real-world dataset they used had 1381 variables and 85,274 customers. The author compared interpretable models like logistic regression, Fisher linear discriminant analysis and decision Trees. The author selected 50 variables based on Student's t-test. The author main findings were that logistic regression outperformed other algorithms that the author used, and he stated that decision trees were very unstable for the dataset he used. Also, the author argues that the usage of ensemble models is improper to predict customer churn for telecommunication sector.

In the study done by Azeem, Usman, & Fong (2017) the authors used fuzzy classifiers algorithm to predict customer churn for telecommunication sector. They used a real-world dataset from South Asian telecommunication company that had 722 variables and 600,000 customers. To their knowledge that was the first paper to implement fuzzy classification algorithm to predict churn for telecommunication sector. Feature selection was based on domain knowledge, and the authors chose the most important 84 variables. they used 80% of the dataset for training and the reminder 20% to test the model. The dataset contained only numeric features, they used min-max algorithm to normalize all the dataset to be in the range of 0-1. The authors Used oversampling technique to balance the dataset since, it had only 9% churners and the percentage after the balancing was 60% non-churners and 40% churners. They used AUC of the ROC curved to evaluate the model. They compared the performance of some fuzzy classifier namely, FuzzyNN, VQNN, OWANN and FuzzyRoughNN with other non-fuzzy algorithms: neural network, logistic regression, decision tree C4.5, Support vector machine, Adaboost, Gradient boost machine and Random Forests. Their main findings were: that fuzzy classifiers are more accurate than other classifiers to predict customer churn. The worst performing classifier was the C4.5 decision tree, then the ensemble algorithms achieved an average AUC 0f 0.5725, the best classifier was OWANN (Fuzzy-rough ownership) with AUC of 0.68.

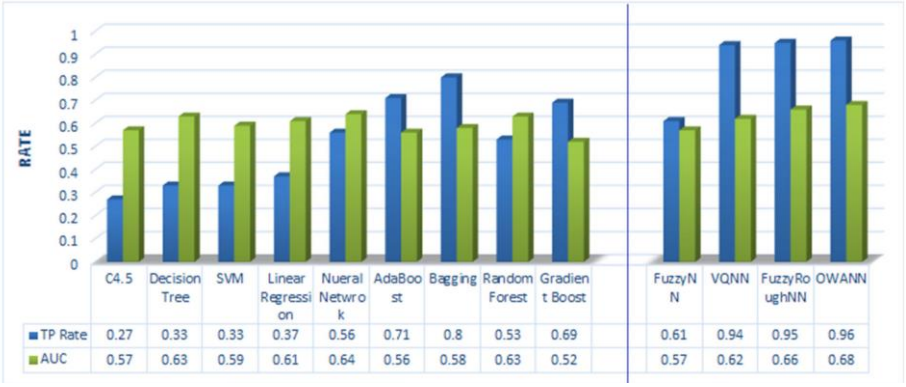

Fig.2.1. Results of paper (Azeem et al., 2017)

In paper by authors Effendy, Adiwijaya, & Baizal (2014), the authors worked on an Indonesian telecommunication dataset which has only 0.7% churners, their objective was to tackle the problem of imbalance in their dataset by combining both SMOTE (Chawla, Bowyer, Hall, & Kegelmeyer, 2002) to the minority class and undersampling algorithm to majority class. By using only the weighted random forest it didn't improve the accuracy of the model so they included the SMOTE algorithm for the minority class and simple under sampling for the majority class to balance the dataset. They removed any feature with more than 80% of missing value, also redundant features were remove as well. Also, they converted all the categorical features into numeric. To validate the model, they used K-Fold validation with 10 folds. They compared the performance of the model without treating the imbalance to the models with under sampling and over sampling. By combining over sampling with under sampling it increased the model performance significantly rather than using the weighted random forest without balancing the dataset. Weighted Random Forest was introduced because the random forest algorithm is biased towards the majority class (Chen, Liaw, & Breiman, 2004), a penalty can be added on misclassifying the minority class. By adding weights to each class, the majority class has less misclassification cost than the minority class. The weights adjustment is done in two phases: the induction phase and the terminal phase. Weighted majority vote is used to determine the predicted class of each data point. By aggregating the weighted vote of each tree to get the final predicted class. Out-of-bag can be used to tune the weights used.

In the study done by Hassouna, Tarhini, Elyas, & Abou Trab (2015) the authors made a comparison between logistic regression and decision trees using a real-world dataset provided by an English mobile operator. They used two balanced datasets (50% Churners): one for training the algorithm with 17 variables and 19,919 customers and another dataset for testing with 17 variables and 15,519 customers. CART, C5.0 and CHAID decision trees were used to be compared to logistic regression. They used AUC of the ROC curve, top-decile and overall accuracy to evaluate and compare between different models. Their main findings were that the C5.0 decision tree outperformed other algorithms including Logistic regression with AUC of 0.763. Their findings contradicted with the findings of the author (Owczarczuk, 2010) where he stated that logistic regression is more accurate to predict customer churn for telecommunication sector.

In the study done by Jayaswal, Prasad, & Agarwal (2016) the authors compared between C5.0 decision tree and two main decision trees ensembles: Random Forests and Gradient Boosting Machine to predict customer churn for telecommunication sector using a publicly available dataset. The dataset had 21 variables and 3333 customers with 15% churners, the authors followed the findings of (Burez & Van den Poel, 2009) which stated that sampling techniques will not affect the performance of the models used and so they didn't treated the imbalance in the dataset. They used 75% of the dataset for training and 25% for testing. Their main findings were that: The GBM outperformed the Random Forest and C5.0 for this dataset.

In paper by authors Amin et al. (2017) the authors explored rough set approach (Pawlak, 1982) to the churn problem, which outperformed other machine learning algorithms for their used dataset. They split the dataset into 70% for training and 30% for validation. They compared the rough set approach to the state of the art algorithms used in churn prediction like ANN, DT and SVM. They noted that for this dataset the rough set approach achieved better accuracy than all other compared algorithms.

In paper by authors Kamalraj & Malathi (2013), the authors studied the performance and accuracy of C4.5 (J Ross Quinlan, 1993) and C5.0 decision trees. The study was done on a public dataset with 3333 customer. C5.0 decision tree achieved higher accuracy with better performance in regards to computational power.

2.2. KDD CUP 2009

In paper by authors Lo et al. (2009), the authors combined the three classification task (churn, appetency and up sell) into one joint Multi-class classification task. Three ensemble techniques were used to classify this multi-class classification task: The first technique the authors used was L1-regularized maximum entropy because of its robustness against outliers but its main drawback is that it cannot deal with missing values because this model will assume no missing data, so the authors added a binary feature next to each feature with missing value to indicate that this feature had missing value, then all the categorical features were hot-encoded which resulted in 111,670 additional features to the dataset, and to scale the dataset the authors used Log and linear scaling. The second technique: a heterogeneous base learner was introduced to handle the numeric and categorical features which was a tree classifier "decision stump" then coupling the base learner with Adaboost algorithm to increase the predictive performance of the base learner. Finally, the third ensemble was a selective Naive Bayes classifier that automatically treat the categorical features with high cardinality by grouping them also, the NB classifier will discretize the numerical features. Then the authors applied a post processing algorithm "SVM" to adjust the predictive power across the three classification tasks, because for some data points the classifier will assign higher probabilities to more than one label. Overfitting was avoided using K-fold cross validation to better train the classifiers. The authors ranked the third place with AUC of 0.8461 for the slow track competition (15,000 features and 50,000 instances).

In paper by authors Xie & Coggeshall (2009), the authors used Stochastic gradient boosting tree as a classifier to predict the three class labels for the large and small datasets. The authors bagged 5 gradient boosted decision trees using different seeds and then took the average votes of these trees models to increase the model performance. The authors chosen the log-likelihood as a loss function for the stochastic gradient boosting tree. To preprocess the dataset, the authors binned high cardinality categorical features "greater than 1000 unique levels" into 100 new levels. Also, the authors discretized 10 numerical features that had small unique values. For imputing missing values, the authors imputed the categorical features with a risk factor and imputed numerical features with adding another feature that contain an identification to identify these features as having missing values. The authors removed redundant features with very low variance also removed quasi-constant features. Also, SGBT algorithm was used for feature selection as a wrapper method. To treat the imbalance in the dataset down sampling was used to down sampled the majority class label. The authors split the dataset into 75% for training and 25% for testing the algorithm. Then the authors used 5-fold validation for the training dataset. The authors tried logistic regression algorithm, but it scored lower than the SGBT. Also, SVM was tried that had an approximate score as SGBT but with more computational time. The authors ranked the second in the fast track with large dataset (AUC = 0.8448) and third place in the slow track for the large dataset (AUC = 0.8478) and scored an AUC of 0.8478 for the small dataset.

In paper by authors Kurucz et al. (2009), the authors worked on the large dataset which contain 15,000 features and 50,000 instances. The authors tried different feature reduction algorithms including: distance, information and dependence measures but these algorithms didn't perform well as they tended to over fit the data by selecting non-predictive features or selecting highly correlated features and so these measures can decrease the generalizability of the model. The feature selection algorithm the authors opted for was the LogitBoost due to its robustness against outliers. The final classifier the authors opted was LogitBoost combined with ADTrees by taking their average classification. The algorithms the authors opted scored AUC of 0.8457.

In paper by authors Niculescu-Mizil et al. (2009), the authors worked on the large dataset considered categorical missing values as a separate value, and imputed the numeric features using the mean of the feature. Also, an additional indicator feature was added to indicate that this feature had a missing value. The authors hot-encoded the top 10 common values of each categorical feature to make all the dataset to be numeric, afterwards the authors standardize the dataset, then all the redundant features were removed from the dataset. The authors used 10-fold validation technique to validate their results on the training dataset. The authors built a library of base classifiers: RF, Boosted trees, LR, SVM, DT, NB, KNN and Regularized Least Squares Regression. Some classifiers were trained on reduced datasets using PCA, Pearson correlation and MI. The authors noted that the best individual classifier on Churn label was Boosted trees followed by LR then RF. The highest AUC obtained by using only the base classifiers was 0.7354 for churn label. The authors used an ensemble selection algorithm with greedy forward stepwise selection to select an ensemble from the library of base classifiers which yielded an AUC of 0.7563 for churn label. The AUC of churn increased to 0.7629 after adding new features generated from tree induction and co-clustering. The final AUC of Churn was 0.7651 and the average AUC overall labels was 0.8519.

In paper by author Yabas (2013), the author used the small dataset which contains 230 features and 50,000 instances. For the preprocess step the authors removed all the redundant features and any feature with more than 99% missing values. The authors imputed the numerical missing values with the mean of the features and imputed the categorical features with by placing a value “Missing”. The authors normalized the numerical features to be in the range 0 and 1. To handle the high cardinality of the categorical features the authors kept only the top 10 most frequent levels and grouped the rest of levels into a level called “others”. The authors trained 250 models with 20 algorithms. The authors tried NN, AdaBoost and SVM but pointed out these models are very computationally expensive and reported that their results are very poor compared to other algorithms they used and so the authors eliminated these models from there analysis. The base model algorithms proposed by the authors included an ensemble of C4.5 decision trees, logistic regression, bagging and random forests. The authors pointed out that the RF was the best performing base classifier. The authors argued that each classifier they used needs a specific preprocessing step and so the authors built a specific preprocessing step for each classifier which will construct a meta-classifier and then all the meta-classifiers with AUC greater than or equal 0.69 will vote for every instance and the majority of these votes will predict the label of each test instance. The meta-classifiers that exceeded the threshold AUC were LR, DT with Bagging, RF. The final AUC achieved was 0.7230 across all the three labels (churn, appetency and up-selling).

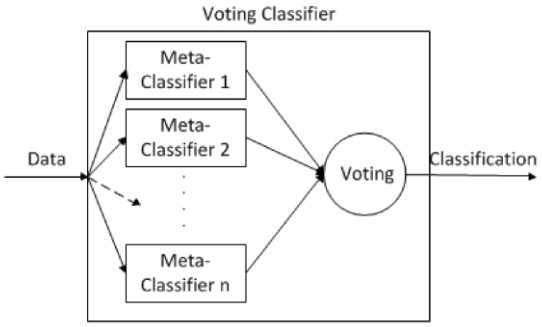

Fig.2.2. Voting classifier (Yabas, 2013)

In paper by authors Doetsch et al. (2009), the authors used small dataset, they reformulated the task as having multi-task approach where they trained a single classifier with three binary outputs instead of single classification task, which yielded an increase of 0.015 to the AUC. They used Information Gain ratio as a feature selection algorithm and likelihood-ratio test as a ranking algorithm, then they tried MLP, SVM, LMT and boosted decision stumps. The best-performing technique was by stacking the output of boosted decision stumps algorithm as a level 0 algorithm then using logistic model tree with AUC splitting criteria as level 1. The

algorithm used to implement the boosted decision stumps is the AdaBoost.MH algorithm and for LMT they implemented their own modification which was a C4.5 tree where each node estimates a LogitBoost model.

In paper by authors Miller et al. (2009), the authors used large dataset, the missing values for the numerical and categorical features were treated as a separate level, used decision trees and boosting, in their attempt to treat the high cardinality categorical features, they tried to aggregate levels with less than 1,000 observations into 20 levels based on their response, but that created the problem of overfitting, so to prevent this problem they tried to aggregate the features independent of their responses instead they aggregated based on their observations, by creating four groups being the first group includes the features that had 1000 observations, and the second group is the levels that has 500 to 999 observations, and the third group are levels with 250-499 observations, and the forth are levels that had 1-250 levels, which prevented the overfitting problem and increased the AUC by 0.0015 compared to the single-level aggregation.

2.3. MISSING VALUES LITERATURE REVIEW

One of the most challenging obstacles in machine learning is imputing missing values, where Imputation of missing values is the treatment to the problem of missing data by replacing it with values (Batista & Monard, 2003), to obtain a complete data set with no missing values. Firstly, the missing data mechanism should be known cause depending on the mechanism a proper method should be implied, where missing data mechanism describes the process that determines each data points likelihood of being missing, there are three main categories (Enders, 2010):

1- MCAR: Missing completely at random, can be thought like random missing values where the missing values is unrelated to the dataset, the probability of having missing values in any feature doesn't depend either on it or other variables in the dataset. Can be verified by using a complete feature in the dataset by comparing the mean of the observed data points by the mean of the missing data points and the two means shouldn't be different.
2- MAR: Missing at random, the probability of having missing values in any feature in the dataset doesn't depend on its value, but instead it depends on other features in the dataset. Cannot be verified unless the values of missing features are known.
3- MNAR: Missing not at random, the probability of having missing values in any feature in the data set depends on the values of the feature itself, cannot be verified unless the values of missing features are known.

Deletion methods:

- Listwise deletion: only the observed values are retained and the rest is removed from the analysis(Schafer, 1999), but its main disadvantage that it can result in biased results (Graham, 2009) also it will remove a lot of useful information from the dataset, only accepted for 5% missing values.
- Pairwise deletion: used when estimating covariance or correlation matrices, for each pair it will use the complete cases for both.

Single imputation methods: single value of each missing data point, its main disadvantage is that one single value cannot rightly predict the missing value and if there's not only one value is enough for this missing value (Rubin, 1988)

- Mean imputation: Impute the missing values with the mean of the available data points of the same feature, produces unbiased estimates of the variance.
- Regression imputation: Impute the missing values using regression to predict the missing values using other features as predictors.
- Stochastic regression imputation: Impute the missing values using regression but it will add a stochastic value to the predictions of the regression imputations, produces an unbiased estimate with MAR datasets.

- Hot deck imputation: the data are clustered into several cluster then the missing value will be imputed with the mean or the mode of the cluster it falls in, prediction models where the creation of a machine learning model to estimate the missing value using other features in the dataset.
- Baseline observation carried forward BOCF: Impute the missing values by taking the baseline observation as a replacement for the missing data.
- Last observation carried forward LOCF: Impute the missing values by taking the last observed value as a replacement for the missing data.
- Strawman imputation: The numerical missing values will be substituted by the median, while the categorical missing values will be substituted by the most frequent level, has a great advantage of being rapid, for NMAR missing values the strawman imputation technique can have higher accuracy for highly missing values more than 75% (Shah, Bartlett, Carpenter, Nicholas, & Hemingway, 2014).

Multiple imputation methods: By having multivariate complex dataset an adequate missing data method is needed, multiple imputation (Rubin, 1987, 1996) is commonly used to predict missing values in complex datasets where many features has missing values. Where it Creates several data sets with different values of each missing data point, it will take into account the uncertainty of each missing value meaning that not only one value can rightly predict the missing value (Azur, Stuart, Frangakis, & Leaf, 2011). Two major methods to impute missing values using multiple imputation are: joint modeling (JM) and the fully conditional specification (FCS) method, where the joint modeling method is based on normality and linearity of all features and the imputations will be generated by linear regression, and so the JM is only suited for continuous features. While Fully conditional specification does not assume normality or linearity of the features of the study and so it can be used either for continuous or categorical features.

Under assumption that the missing values in this dataset is MAR, multiple imputation technique can be used like: MICE which is a multiple imputation technique (van Buuren & Oudshoorn, 2011) based on the method of fully conditional specifications which has the basic assumption that the missing values are missing at random, meaning that their probability of being missing depends only on the observed data (Azur et al., 2011). The imputation method depends on the nature of variable distribution, whether its continuous or binary the used algorithm will be linear regression or logistic regression respectively (Azur et al., 2011). Fig.3.4. summarize different techniques to impute missing values.

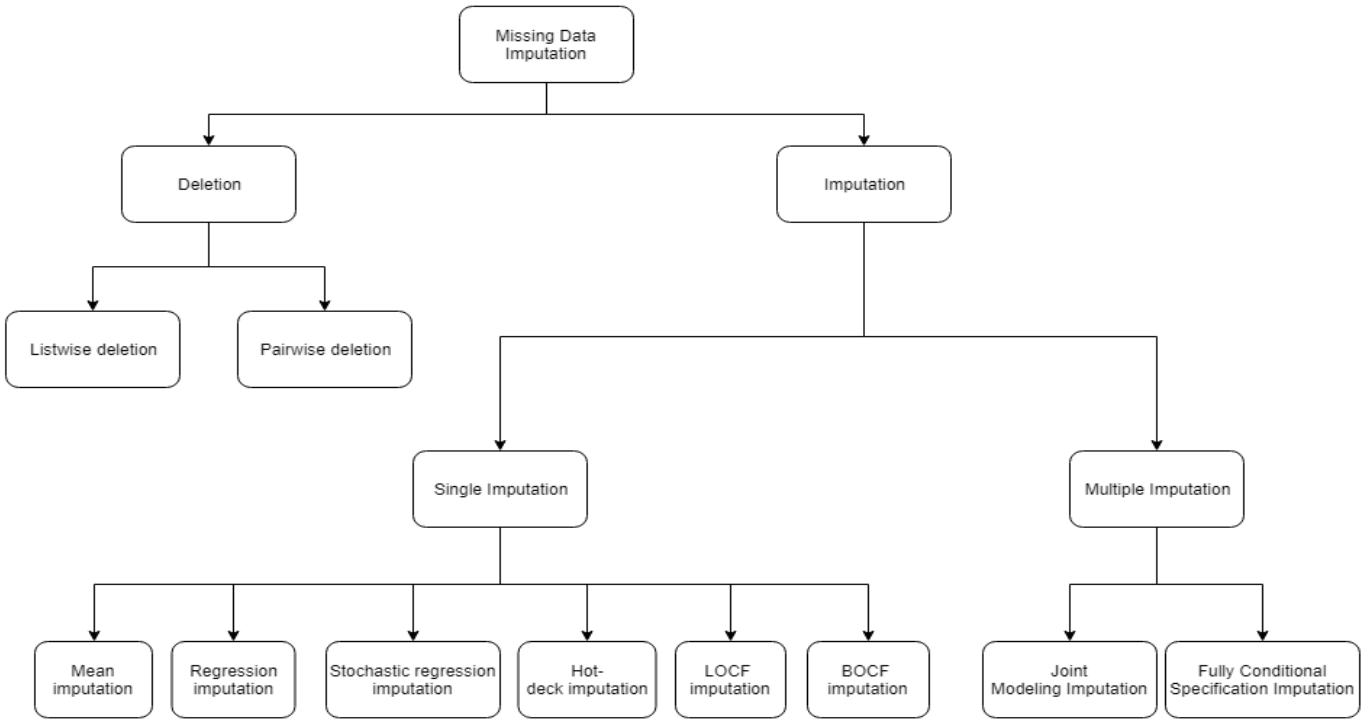

Fig.2.3. Missing Values Imputation Methods

2.4. DECISION TREE LITERATURE REVIEW

By looking at the next figure by the authors García, Nebot, & Vellido (2017) which states that DT is the most used model in the recent Churn prediction literature, due to its interpretability and ease of deployment, also because of its hierarchical shape which can present to decision makers as a set of rules. Decision trees recursively partition the data space and fit a simple prediction rule within each partition, starting at the root of the tree to classify a new data point then follows the branch indicated by the result of the decision rule, until a leaf node is reached, the class of the leaf node is the class of the new data point.

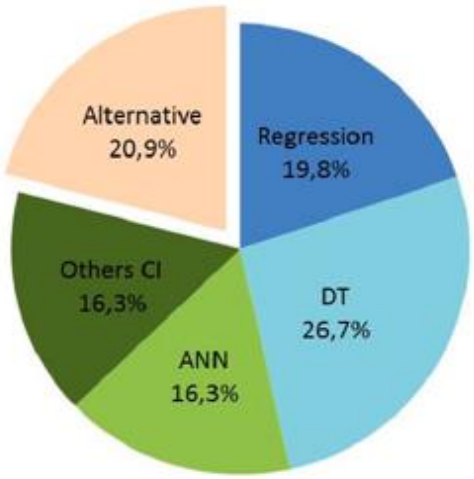

Fig.2.4. Models used in Churn Prediction (García et al., 2017)

Two main categories of decision trees can be found in literature: classification trees and regression trees (Rokach & Maimom, 2014), where the classification trees are the designed for categorical target feature and its prediction power can be measured using misclassification cost, the regression trees are designed for continuous target feature and its prediction power can be measured using square difference between the observed and predicted values. The used dataset has a binary output, so the used algorithm will be classification decision tree.

Classification decision trees produces rectangular disjoints sets of the dataset, by recursively partitioning the dataset one feature each time (Loh, 2011). A decision tree structure consists of either a leaf node or a decision node, where the leaf node contains the predicted class and the decision node contain a decision rule to further split the data (Miroslav Kubat, 2015). To split the data a splitting criterion is used, where a single or multiple feature can be used, the single splitting criterion named univariate split and the multiple features named multivariate split (C. E. Brodley & Utgoff, 1995), univariate decision trees are more common because of its simplicity and comprehensibility (Murthy, 1996) but its main drawback that it will only divide the space with a boundary that is orthogonal to the feature being used (C. Brodley & Utgoff, 1992).

There are many types of decision trees proposed in literature named THAID (Messenger & Mandell, 1972), ID3 (J R Quinlan, 1986), C4.5 (J Ross Quinlan, 1993), CART (Breiman, Friedman, Stone, & Olshen, 1984), CHAID (Kass, 1980), FACT (Loh & Vanichsetakul, 1988), CRUISE (H. Kim & Loh, 2001), QUEST (Loh & Shih, 1997) and CTREE (Hothorn, Hornik, & Zeileis, 2006). To better understand their use, commonly used decision tree algorithm will be explained in this section.

- If $P_1, \dots, P_K$ is a fraction of record belonging to the K different classes in a node N:
1-Then the Gini-index $G(N)$ of the node N is defined as: $G(N) = 1 - \sum_{i=1}^{K} P_i^2$,where the value of $G(N)$ lies between 0 and $(1 - \frac{1}{K})$, the smaller the value of $G(N)$ the greater the skew, If the classes are evenly balanced the value is $(1 - \frac{1}{K})$.

2- The entropy $E(N)$ of the node N is defined as: $E(N) = -\sum_{i=1}^{K} P_i . \log P_i$, where the value of the entropy lies between 0 and $\log K$, when the records are perfectly balanced among different classes the value of entropy is $\log K$ which is the maximum entropy while the smaller the entropy the greater the skew in the data.
The Gini-index and entropy provide an effective way to evaluate the quality of a node in terms of its level of discrimination between the different classes.
Given a N labeled training set $\{(x_i, y_i)\}$ for $i = 1,2, \dots N$, $x_i \in \mathbb{R}^n$ and the target variable "Churn or Not" $y_i \in \{0,1\}$

Entropy and Information gain

Entropy for two classes classification: by having S set of examples and $P_{\oplus}$ is the proportion in the class $\oplus$, and $P_{\ominus}$ is the proportion in the class $\ominus$, knowing that $P_{\oplus} = 1 - P_{\ominus}$

$$E(S) = -P_{\oplus} . \log_2 P_{\oplus} - P_{\ominus} . \log_2 P_{\ominus}$$

The generalization of Entropy for i target classes, so by having S set of instances and P_i is the fraction of S set that has output value of i

$$E(S) = -\sum_{i=1}^{n} P_i \log_2 P_i$$

the change in the Entropy can be measured by Information Gain (Hssina, Merbouha, Ezzikouri, & Erritali, 2014)

$$Gain(p, T) = E(S) - \sum_{J=1}^{n} \left(P_j * E(P_j) \right)$$

2.4.1. **THAID (Messenger & Mandell, 1972)**

The first published classification tree algorithm was Theta Automatic Interaction Detection THAID which splits a node by exhaustively searching overall X and S for the split that minimizes the total impurity of its two child nodes.

2.4.2. **ID3 (J R Quinlan, 1986):**

ID3 decision tree was designed for nominal features only, continuous data have to be converted into nominal features using binning or other embedding methods before it can be used. Uses information gain as splitting criterion, Usually doesn't guarantee an optimal solution, cause it will got stuck in local minima due to its greedy partition strategy. Doesn't apply any pruning strategy so it has high risk of overfitting the data.

2.4.3. **C4.5 (J Ross Quinlan, 1993):**

C4.5 decision tree was an evolution of **ID3** (J R Quinlan, 1986) uses entropy for its impurity function called gain ratio, multiway split yields a binary split if the selected variable is numerical; if it is categorical, the node is split into C subnodes, where C is the number of categorical values. Can handle numeric attributes by splitting the attribute's value range into two subsets. Specifically, it searches for the best threshold that maximize the gain ratio criterion. All values above the threshold constitute the first subset and all other values constitute the second subset. Uses a pruning procedure which removes branches that do not contribute to the accuracy and replace them with leaf nodes. Can split a node into more than two children nodes, their number depending on the characteristics of the X variable. Uses Pessimistic Error Pruning. Require that the target attribute will have only discrete values

C4.5 Decision Tree Algorithm:

Let X be the training set: grow (T)

1. Find the feature x that contribute the most information about the class labels.
2. Divide T into subsets of T_i, each must have different values at x
3. For each T_i:
   If all examples in T_i belongs to the same class, then create a leaf labeled with this class, otherwise, apply the same process recursively to each training subset: Grow (T_i)

Algo.1. C4.5 Decision Tree Algorithm

Decision Tree root choice from discrete features C4.5 Algorithm:

1. Calculate the entropy of training set T using the probabilities P_{pos} of the positive labeled class and P_{neg} of the negatively labeled class
$$H(T) = -P_{pos} \log_2 P_{pos} - P_{neg} \log_2 P_{neg}$$
2. For each attribute x that divides T into subsets T_i with related sizes P_i:
3. Calculate the entropy of each subset T_i
4. Calculate the average entropy: $H(T, x) = \sum_i P_i . H(T_i)$
5. Calculate the information gain: $I(T, x) = H(T) - H(T, x)$

Algo.2. C4.5 Root Choice for Discrete Features

Decision Tree root choice from numeric features C4.5 Algorithm:

7. For each numeric feature x_i:
8. Sort the training example by values of x_i
9. Find the candidate threshold θ_{ij} as those lying between data points of different labels
10. For each θ_{ij} determine the amount of information contributed by the Boolean feature thus created
11. Choose the pair $[x_i , \theta_{ij}]$ that has the highest information gain.

Algo.3. C4.5 Root Choice for Numeric Features

2.4.4. **CART (Breiman et al., 1984):**

CART decision tree algorithm uses Gini-index (Light & Margolin, 1971) as a splitting criterion. It first grows an overly large tree and then prune it to a smaller size using 10-fold cross-validation Cost Complexity Pruning to minimize an estimate of the misclassification error. Minimal Cost-complexity pruning: The basic idea of cost-complexity pruning is to calculate a cost function for each internal node meaning that all nodes that are neither leaves nor roots in the decision tree.

The cost function is given by:

$$R_\alpha(T) = R(T) + \alpha . |Leaves(T)|$$

Where

$$R(T) = \sum_{t \in Leaves(T)} r(t) . p(t) = \sum_{t \in Leaves(T)} R(T)$$

$R(T)$ the misclassification cost of the sub-tree obtained after pruning of a certain branch.

α the cost complexity parameter for that branch, for each value of α there is a unique smallest tree that minimizes the cost complexity measure $R_\alpha(T)$.
Gradually increasing the complexity parameter α starting from 0, will lead to a nested sequence of trees decreasing in size. The optimal tree size is determined by 10-fold cross validation, the data set is divided into $N = 10$ subsets, one of the subsets is used as independent test set while the other $N - 1$ subsets are used as training test, the tree growing and pruning procedure is repeated N times, each time with a different subset as test set. For each tree the misclassification error is calculated, the tree with the lowest misclassification error is the optimal tree. Breiman proposed the $1\ S.E.$ rule which will select the smallest tree whose cost complexity measure is within 1 standard error of the cost complexity for the tree with the minimum misclassification error. CART is implemented in the R system(Team, 2009) as RPART (Therneau & Atkinson, 1997).

2.4.5. **CHAID (Kass, 1980):**

CHAID decision tree uses P-value with a Bonferroni correction as a splitting criterion. Has an advantage of splitting the node into more than two children nodes. Usually doesn't prune the result tree. According to Wei-Yin (2014) the major drawback of using the greedy search approach in CART and C4.5 decision trees is its selection bias. Selection bias usually happens when the variables with high cardinality will have a high chance to be selected to split nodes more than the low cardinality variables. For the decision tree to be unbiased all the variables must have the same chance to be selected given that all the features are independent of the target feature.

Unbiased variable selection Decision Trees: As being proposed by **FACT** decision tree (Loh & Vanichsetakul, 1988) **CRUISE**, **GUIDE** and **QUEST** decision trees use splitting criteria based on two-steps. The first step, each feature is tested for association with the target feature and then the most significant feature is selected to split based on it. The second step, an exhaustive search over all the dataset is performed, and because every feature has an equal chance to be selected if and only if each feature is independent of the target feature. So that approach is unbiased.

2.4.6. **FACT (Loh & Vanichsetakul, 1988):**

In FACT decision tree the splits at each node will depend on the number of the classes in each variable. If any categorical features appeared in the dataset it will be transformed into dummy variables (H. Kim & Loh, 2001). For **FACT** to have univariate splits, the features will be ranked using F-test of ANOVA then linear discriminant analysis will be applied to the top features to split the nodes based on it. The tree size will be determined by ANOVA test (H. Kim & Loh, 2001).

2.4.7. **QUEST (Loh & Shih, 1997):**

In QUEST decision tree it will split the features into two categories: Ordered and Unordered features, then it will run ANOVA test on the ordered features and Chi-Square test on the unordered features and so the selection bias will be eliminated. **QUEST** Supports univariate and linear combination splits. As in **CART** (Breiman et al., 1984) it uses ten-fold cross-validation to prune the trees.

2.4.8. **CRUISE (H. Kim & Loh, 2001):**

CRUISE decision tree was an improvement of **QUEST** (Loh & Shih, 1997). Splits each node of the tree into multiple children nodes depending on the number of classes of the target variable. It avoids the selection bias by using two statistical tests: Chi-Square test and F-Test at each node to select which feature to split based on. Uses the same pruning method as **CART** (Breiman et al., 1984).

2.4.9. **CTREE (Hothorn, Hornik, & Zeileis, 2006):**

CTREE decision tree was introduced to avoid selection bias CTREE uses permutation tests where it uses p-values from permutation distributions of influence function-based statistics to select split variables. Does not use pruning; it uses stopping rules based on multiplicity adjusted p-values to determine tree size. The algorithm is implemented in the R package PARTY.

In the following table, different DT are summarized based on their target function, splitting criterion, pruning and splits techniques.

Algorithm	Target feature	Splitting Criterion	Pruning	Splits
ID3	Discrete	information gain	No	Multiway splits
C4.5	Discrete	gain ratio	Pre-pruning Pessimistic Error Pruning	Multiway splits
CART	Mixed	Twoing Criteria	Pre-pruning Cost Complexity Pruning	Binary splits
CHAID	Mixed	chi-square	NO	Multiway splits
FACT	Mixed	ANOVA	ANOVA	Multiway splits
CRUISE	Mixed	contingency table chi-square tests	Pre-pruning Cost Complexity Pruning	Multiway splits
QUEST	Mixed	chi-square ANOVA	Post-pruning Cost Complexity Pruning	Binary splits
CTREE	Mixed	p-values	Bonferroni-adjusted p-values	Multiway splits

Table 1 Summary of Decision trees algorithms

3. METHODOLOGY

To conduct a data mining project a systematic process should be followed. Mainly two process had introduced (Olson & Delen, 2008): The CRISP-DM (Wirth & Hipp, 2000) Cross Industry Standard Process for Data Mining process which is widely used by a lot of data mining practitioners (Olson & Delen, 2008) and the **S**EMMA process **S**ample, **E**xplore, **M**odify, **M**odel, **A**sses. As being pointed out by the authors of Azevedo & Filipe Santos (2008) CRISP-DM can be more comprehensive than the SEMMA model, and so it will be followed in this research.

The CRISP model consists of six stages Fig.3.1., the first step is the business understanding: which is about understanding the business objectives and needs, and to develop a project plan with time frame of when the whole data mining project will take, and a time frame for each stage. The Second stage is the data preparation step: Start with data collection, data visualization, data summary and to discover insights from the data, then to check the data quality. The third step is the Data preparation: Also known as data preprocessing phase, where the final dataset will be constructed from the initial dataset. The forth step is the modelling stage: various machine learning algorithms are applied, and their parameters are tuned to suit the dataset understudy. The fifth step is the Evaluation stage: the models obtained from the previous step are evaluated and their accuracy and performance are assessed, frequently will be reverted to previous step for better choosing different model. The sixth step is the deployment stage: the final stage where the model built will be deployed to be used by the user, usually in a form of an API.

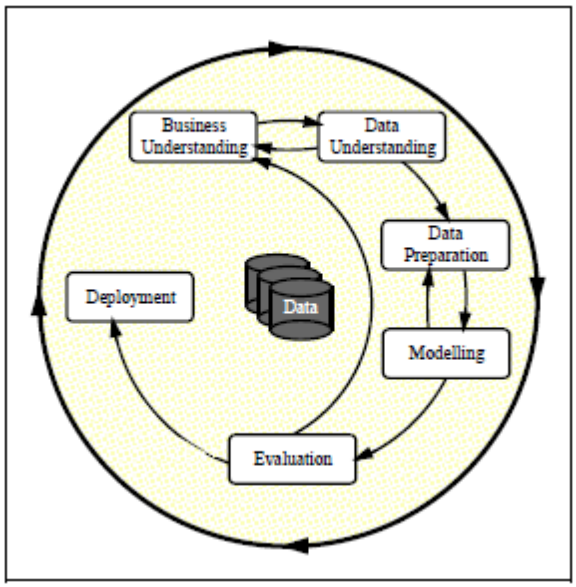

Fig.3.1. (Wirth & Hipp, 2000) CRISP-DM model

3.1. DATASET

The dataset being used in this paper is a real-world dataset provided by the French telecommunication company Orange to develop three Customer Relationship Management algorithms to improve the company marketing strategies (Guyon, Lemaire, Vogel, et al., 2009):

1- Customer Churn: A binary feature which can be defined as probability that the customers will leave the company voluntary to another for better offer or other reasons as stated in the introduction.
2- Customer appetency: A binary feature which can be defined as Probability that the customer will buy a new product from the company
3- Customer up-sell: A binary feature which can be defined as Probability that the customer will upgrade his original plan to another one.

The dataset was released as a part of the 15th edition of Knowledge Discovery and Data Mining (KDD) annual competition, which is hold annually to explore new algorithms and to enrich the scientific community. In 2009 the French telecommunication company Orange released a challenge to detect these labels. The challenge had two paths: The fast path and a slower one, for the fast path the contestants had to submit their result in five days and for the slow path the contestants had one month to submit their results. Two datasets were released: large dataset which contains 15,000 features (14,770 numerical features and 260 Categorical features) and 100,000 instances, 50,000 to train the dataset and 50,000 to test the algorithms and a small dataset which contains 230 features (190 numerical features and 40 Categorical features) and 50,000 instances to train the dataset and another 50,000 to test the algorithms. Submissions were evaluated using AUC of the ROC curve, with the average AUC of the three class labels were used to rank the contestants. All the features presented in the large and in the small datasets were anonymized for the customer privacy and no meta-information was provided to explain any of the features. Table 1 shows the winners of this competition for the large and the small datasets.

KDD 2009 dataset was chosen because of its availability and its suitability for data mining techniques because the dataset had many challenging tasks from being multi-class classification task also, the dataset contains missing values that needed to be imputed and so many imputation algorithms can be experimented using this dataset, the dataset is high dimensional and so many feature selection techniques can be studied and examined on this dataset. The dataset used in this paper is the small dataset for the sake of computational resources available. The dataset is heterogeneous meaning that it contains numerical and categorical features. Also, the dataset is high imbalanced meaning that the number of positive instances is much lesser than the negative instances, for churn the number of positive instances is only 3764 out of 50,000 and for appetency only 764 out of 50,000 and for up-sell only 1400 out of 50,000. All the results reported here are the AUC of the ROC curve of the small dataset of the KDD 2009 challenge.

3.2. DATA PRE-PROCESS

Dealing with high dimensionality dataset is very challenging for machine learning algorithms and before applying any algorithm a proper preprocess is required (Kumar & Sonajharia, 2014). According to the authors Han & Kamber (2000) The preprocessing step is vital for building the model, which can be used to remove noisy and redundant data to improve the model performance and accuracy.

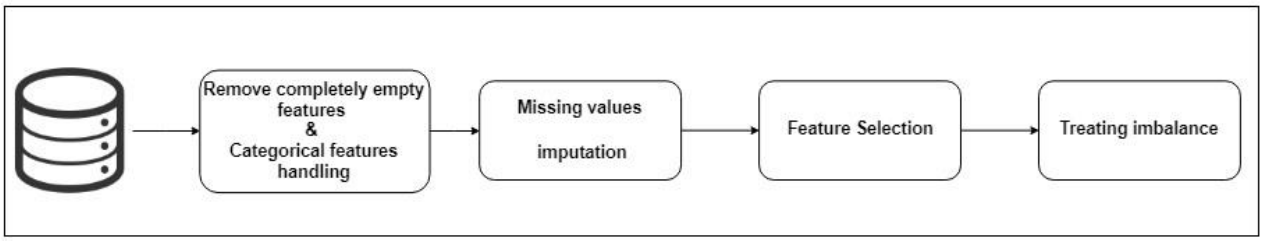

Fig.3.2. The Flowchart of the Preprocess Step

Prior to using the dataset in data mining models high cardinality categorical features should be handled, treat incomplete data, treat imbalance in the data if found and to remove empty features and impute missing values if found. Especially for classification tasks the number of features used must be minimum, if the dimensionality is large this will increase the computation cost also it will produce error and decrease the performance of the used algorithms (Ertel, Black, & Mast, 2017). By having one feature with above average values compared to the other features, it will overpower the rest of the features by doing any analysis, to overcome this issue the usage of normalization is preferable to maintain the balance in the data, even many software packages will do it by default, to treat this a linear scaling will be used meaning to scale the numerical features to range of $[0,1]$. Models that are built on smooth functions such as regression models, are affected greatly by different scales in that dataset (Casari & Zheng, 2018). While, tree-based models that are based on splitting technique are robust against different scales in that dataset (Casari & Zheng, 2018). The used models in this research are tree-based models, so no scaling will be performed.

3.2.1. **Categorical features handling**

By having a high cardinality categorical feature in the data set will not only decrease the accuracy of the model, it will also increase the computational time needed. High cardinality can be defined as non-ordered fields having large number of unique values can reach up to thousands or even millions (Micci-Barreca, 2001). Under the assumption that any feature with more than 500 unique level is only text with no predictive power, these features will be removed from the modeling process. Any levels with less than 5% prevalence will be combined into a new group called "Others". Var217 has 13990 levels, means that each level has less than 1% prevalence in the dataset, also Var214 which has 15415 levels and the highest level appeared 74 times in the dataset with prevalence of 1.5%. Also, Var192 and Var204 has many levels with less than 1% of each level so both will be removed from the analysis. As being proposed by (Miller et al., 2009) the attempt to resolve this issue will be by creating four groups of levels based on the number of observations, where levels with more than 1000 observations will be grouped together, levels with exposure between 500-999 will be grouped together, levels with exposure between 250-499 levels will be grouped together, levels with exposure between 1-250 will be grouped together.

3.2.2. **Imputing Missing Values**

By using R Library(healthcareai) we can see the count of missing values in each variable, the following table have the percentage of missing values with respect to each variable.

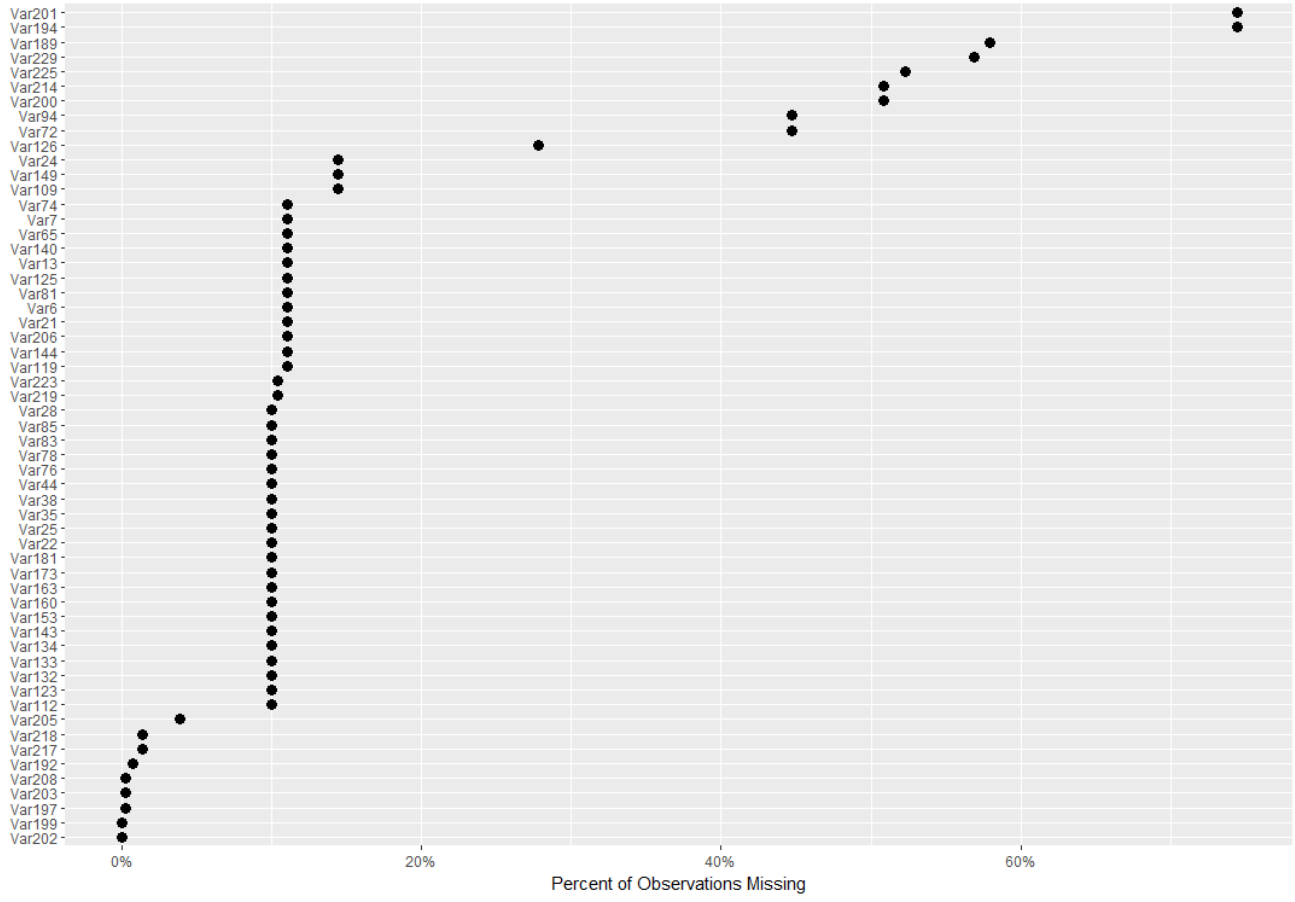

Fig.3.3. Missing values distribution

In practice one important parameter for MICE imputation method is the number of datasets to create to impute the missing value, the factors affecting this decision are: The size of the dataset, the amount of missing values and computational resources available, for the dataset in hand imputing 10 datasets with 5 imputations seemed feasible. Before running the MICE algorithm, all the empty and variables with more than 50% missing value were removed from the analysis under the assumption that these variables cannot be used. By removing all the totally empty features, then to remove any variable with more than 50% missing value, 77 variables will

be retained for building the model. For MICE the features with the least amount of missing values will be imputed first, then used in subsequent imputations, then the next feature with the second least amount of missing values will be imputed then used with the first in subsequent imputations. Missing values will be imputed using mice, hot-deck and mean then their accuracy will be assessed using AUC of the ROC to choose the best method to impute the missing values.

3.2.3. **Feature Selection**

As being introduced in Bellman (1961) the curse of dimensionality can be defined as having too many dimension in the dataset that will make the classifier takes a lot of time also it will decrease its predictive power. The cornerstone of data preparation for any type of algorithm is choosing which feature to use and to exclude any redundant features that could increase computational power and reduce the prediction power (Kalousis, Prados, & Hilario, 2007). Also, using feature selection reduce the overfitting of the model, and thus increasing the stability of the model (Kumar & Sonajharia, 2014). According to the authors Wu, Qi, & Wang (2009) feature selection has two main approaches which are filter methods and wrapper methods based on the relationship between the features and the algorithm in use.
Filter methods: no feedback from the used algorithm, evaluation is being done by using feature selection criterion like information gain, distance measure, dependency or consistency, but its main drawback that it requires an exhaustive search over all the dataset which in case of telecommunication can contains hundreds of features. While, wrapper methods use feedback from the used algorithm where the prediction power of the model is used as the main criteria of selecting which feature to be used. In this thesis the wrapper method using Random Forests (Breiman, 2001) will be chosen because the variable selection and supervised machine learning will be optimized together as a complete learning system (Svetnik, Liaw, & Tong, 2004). The main idea of the algorithm is that it is based on iteratively removing the lowest ranking features, then to assess the model AUC using K-Fold Cross Validation. As being suggested by Determan (2015) the same algorithm can be repeated using Gradient boosting machine (Friedman, 2001). The AUC result of the random forest algorithm and the Gradient Boosting machine are reported in the results section and the best performing algorithm will be used to select the most important features.

Feature Selection using Random Forest Algorithm (Svetnik, Liaw, & Tong, 2004):

1- Using K-Fold validation to split the dataset into training and validation sets.
2- Using all the features of the training dataset to train the random forest algorithm.
3- Record the rank of every feature according to their importance.
4- Half of the least important features will be removed.
5- Train another random forest on the rest of the features.
6- Record the rank of the rest of features according to their importance.
7- Half of the least important features will be removed.
8- Steps 2-7 will be repeated from 10 to 50 times.
9- Select the top ranked features.

Algo.4. Feature Selection using Random Forest Algorithm

3.2.4. **Treating imbalance**

The Imbalance problem is very common in the field of machine learning (YANG & WU, 2006), because usually the desired label is not well represented in the dataset. The imbalance in the dataset in some cases can be very high which can reach up to 10,000,000 times i.e. the majority class is higher than the minority class by 10,000,000 times (J. Wu, Brubaker, Mullin, & Rehg, 2008). The case in telecommunication sector is the churners

percentage is most likely to be the minority class. There are a lot of studies on methods and algorithms on how to treat dataset unbalance mainly the algorithms presented in the paper from Batista, Prati, & Monard (2004), the authors presented a comparative study of ten popular imbalance handling techniques which can be summarized into two main groups: random and non-random techniques, random techniques are non-heuristic techniques which include under-sampling and over-sampling techniques, where random under-sampling is to remove data points randomly from the majority class to balance the dataset. Random over-sampling is to replicate the minority data points randomly to balance the dataset but empirical studies showed that over-sampling tends to overfit the data (Prati, Batista, & Monard, 2008).

A lot of Non-random techniques was introduced mainly to put some heuristics in selecting which data point to remove instead of doing it randomly. A few examples of undersampling non-random techniques includes Tomek links (Tomek, 1976) which is an undersampling technique where data points belonging to the majority class are the data points to be removed from the model. Formally Tomek Link can be defined as: by having two classes in the dataset '+' and '-', assume two data points each belongs to different class: x_i, x_j and the distance between them is $d(x_i, x_j)$ then x_i and x_j are Tomek link if for any data point x_l, $d(x_i, x_j) < d(x_i, x_l)$ *or* $d(x_i, x_j) < d(x_j, x_l)$, in a summary for any two data points if they are Tomek Link then either one of them are noise or both of the data points located on the boundary between '+' and '-'.

Another non-random undersampling algorithm is One-sided selection (M Kubat & Matwin, 2000) which can be defined as: by having a subset S which is selected randomly from population T where each data point in T has an equal probability to be chosen. The OSS algorithm reduces the population space T by removing the majority class examples and so it will create a new population $O = T - C$ where C is the minority class data points. Then a subset A is selected from T where A will correctly classify T by using $KNN = 1$. Then noisy and border line data points will be removed from O. The OSS algorithm is showed in Algo.2.

One of the most widespread non-random oversampling technique was introduced by Chawla, Bowyer, Hall, & Kegelmeyer (2002) called SMOTE , which stands for Synthetic Minority Oversampling Technique. Instead of replicating the minority data points SMOTE produces new synthetic minority data points using the feature space, based on weighted average of the K-NN. SMOTE has three hyper parameters that need to be tuned in first: 1- The over sampling percentage needed to create new data points for the minority class. These new data points won't be replicate of the already exist data points but instead, it will be generated using KNN to reduce overfitting (Up & In, 2017). 2- The under-sampling percentage needed to delete some majority data points. 3- KNN used to impute the synthetic data points. (Chawla et al., 2002) used KNN=5 in their analysis. The AUC result of the OSS, Tomek Links and SMOTE are reported in the results section. Different approaches to treat Imbalance can be found in Fig.3.11.

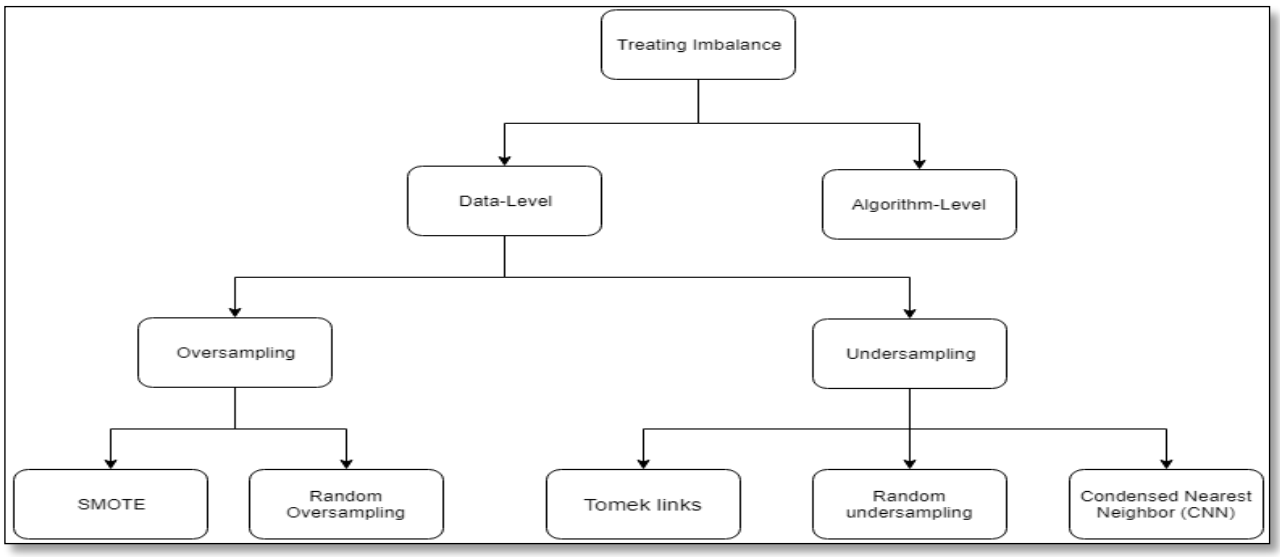

Fig.3.4. Different Approaches to treat Imbalance

3.3. DECISION TREE ENSEMBLES

Because the DT usually suffers from the Bias-Variance error which will reduce the model ability to predict an unseen data (Dietterich & Bae, 1995), the term Bias (Mitchell, 1980) means that how on average are the predicted values differ from the actual values, bias is mainly made from the assumptions made during training the model which will prevent the model from learning an important relationships between the variables and the labels, bias also known as underfitting. Variance means how different will the predictions of the model be at the same point if different samples are taken from the same population which originate from the small specific changes in the training set, where the model will learn the noise in the training dataset, instead of learning the correct pattern which can also be known as overfitting. The classification error can be decomposed to three non-negative values variance, bias and noise (Kohavi & Wolpert, 1996). According to Domingos (2000) who developed a unified theory for bias-variance decomposition to include both classification and regression: Given $y_i = n(f(x_i))$ where $f(x_i)$ is the true label of x_i and $n(.)$ is a noise that changed the true label of $f(x_i)$, by having a training set of $\{(x_1, y_1), \dots, (x_m, y_m)\}$ the learning algorithm will produce a classifier c, giving that $y^* = n(f(x^*))$ be the resulting label of an unseen instance x^*, and $c(x^*)$ is the resulting label, the error is defined as $L(c(x^*), y^*)$. The difference between the predicted and the true label of a data instance is known as loss function. In the classification context the loss function is 0 or 1, where $L(y', y)$ is 0 if $y' = y$ and 1 otherwise, Given that the training set S was taken from population $P(S)$ so the main prediction is $y_m(x^*) = argmin_{y'} E_P[L(y', c(x^*))]$, the main prediction is the most common vote of $c(x^*)$, where the bias is $B(x^*) = L(y^m, f(x^*))$, and Variance will be $V(x^*) = E[L(c(x^*), y^m)]$ lastly the noise is $N(x^*) = E[L(y^*, f(x^*))]$. Bauer & Kohavi (1999) showed that for Naïve Bayes classifiers, simpler models will tend to have small variance error in the cost of higher bias error. Fig.10 shows the relationship between model complexity and bias-variance errors as presented in (Hansen, 2000).

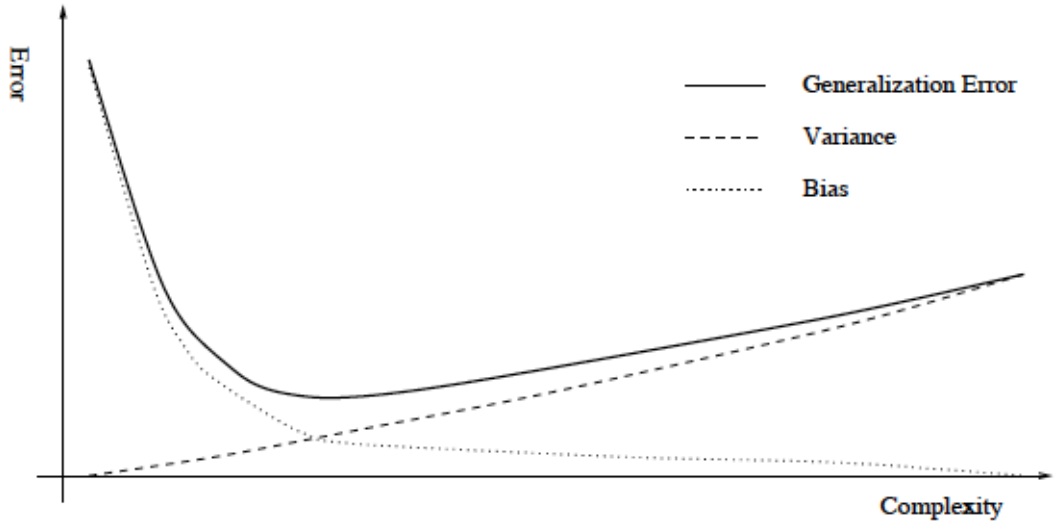

Fig.3.5. Bias-Variance VS. Model complexity (Hansen, 2000)

The bias-variance trade-off error applies to all forms of supervised machine learning so basically, what is needed is a trade-off between bias and variance errors, and that's where Ensemble models come to play.
Many algorithms introduced to minimize the effect of bias-variance errors, another challenge is the noise which is a kind of error that caused by observed values variance, like changes that are unpredictable, this type of error is irreducible. Instead of using one classifier to predict an instance many algorithms can be used, that's where ensemble learners come in hand, where an ensemble is a group of weak classifiers combined to create a stronger classifier, which will help in reducing bias and or variance of data in hand. A classifier h is a function that tries to match a variables vector x to classes in $C = \{C_1, C_2, \dots, C_n\}$. While an ensemble classifier will

consist of a set of classifiers $H = \{h_1, h, \dots, h_k\}$, where the output of the ensemble H depends on the output of each classifier.

3.3.1. **Bootstrap aggregation (Bagging) & Random Forests**

Because decision trees usually suffer from high variance (Breiman, 1996), bagging was introduced to reduce the high variance created by decision trees, cause by averaging set of observations it will reduces the variance. Bootstrap aggregation means: taking repeated samples with replacement from the dataset and averaging their predictions and that will decrease the variance caused by the decision trees (James, Witten, Hastie, & Tibshirani, 2014). It was first introduced by Breiman (1996), the main idea is to reduce the variance of the data in hand so helping reducing overfitting (Bauer & Kohavi, 1999), by creating subsamples of the training instances then to use these subsamples to train the model. Bagging Algorithm Flowchart is presented in Fig. 4.4.

The Bagging Algorithm (Breiman, 1996):
1- Generates N training datasets by random sampling with replacement
2- Setting N estimators: one of each training set, knowing that these estimators are fitted independently from each other, so the model sets in parallel
3- The Ensemble predictions is the average of the individual predictions made from N models created Bagging is more likely in reducing variance than in reducing bias.
*For error estimates while training the model:
*Out-of-bag is the data points that are not selected by bootstrap at each iteration 'On average each data point will be out-of-bag 36% of times'
4- In each iteration predict the out-of-bag data points using the model grown with the subsamples
5- Calculating OOB error by averaging the out-of-bag predictions

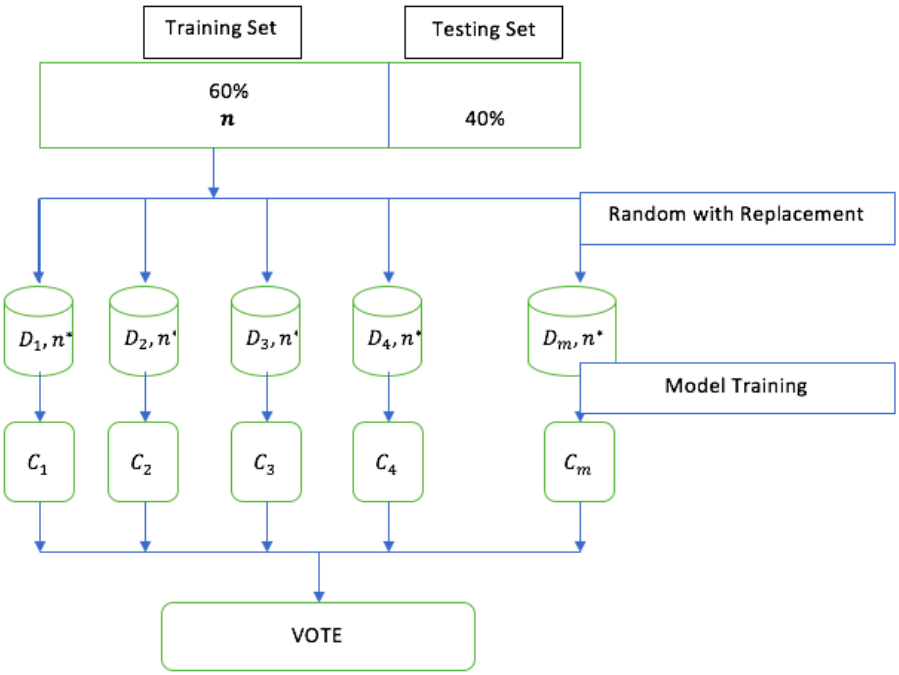

Fig.3.6. Bagging Algorithm Flowchart

One of the main disadvantages of bagging method is that it will produce a correlated bootstrap trees (James, Witten, Hastie, & Tibshirani, 2014). In bagging the bootstrap process is done only on the data points and not the features and so, this will lead to decreasing the variance of the decision trees but the bias term will not

decrease. Breiman (2001) introduced the random forests where the features will be randomly sampled with the data points and this will help in decorrelating the trees and so it will decrease the bias as well (Liaw & Wiener, 2002). Breiman (2001) stated that by having a large number of trees, the Random Forest won't overfit the training data. In summary the main difference between bagging and random forest is that RF has the second level of randomness which is when optimizing each node split, only a random subsample without replacement of the attributes will be evaluated with the purpose of further decorrelation the estimators. Also, because random forests will use bagging for the data points this will produce OOB, cause if n observations will be drawn randomly with replacement from the dataset, then 37% of the n observations will be left out and those observations can be used as a validation sample to test the random forest predictability. RF requires two hyperparameters adjustments: number of trees used and the max depth of each tree in the model. Hyperparameter optimization for the RF will be performed using H2O random search function.

The Random Forest Algorithm (Liaw & Wiener, 2002):
1- Selecting n_{tree} bootstrapped subsamples from the training dataset
*Given $m_{try} = P$;where P is the number of features of the dataset, m_{try} is the randomly sample from the features of the training dataset
2- Developing an unpruned classification tree for each of these subsamples, Choose the best split from m_{try}
3- Predict the unseen data by aggregation of the n_{tree} using majority vote
*For error estimates while training the model:
*Out-of-bag is the data points that are not selected by bootstrap at each iteration 'On average each data point will be out-of-bag 36% of times'
4- In each iteration predict the out-of-bag data points using the model grown with the subsamples
5- Calculating OOB error by averaging the out-of-bag predictions

Algo.5. Random Forest Algorithm

3.3.2. **Boosting**

First introduced by Freund & Schapire (1997). In boosting which means converts weak learners 'base learners' to strong learners, weak means that its slights better than random guess, strong means that it has a more generalizability, which's can be applied to new dataset and predict well, the trees are grown sequentially, means that each tree is grown using the information from previously grown trees, each tree is fit on a modified version of the original dataset, the tree is fit using the current residuals. Suppose that the training instances in space x are drawn $\mathrm{i.i.d.}$ from distribution D and the true function is f, suppose the space x is composed of three parts $\mathrm{x_1, x_2, x_3}$ and each takes 1/3 amount of the distribution and a weak learner has 50% classification error on this problem, the first weak learner is $\mathrm{h_1}$which correctly classify $\mathrm{x_1}$, $\mathrm{x_2}$but has a wrong classification on $\mathrm{x_3}$, the idea of boosting is to correct the mistakes done by $\mathrm{h_1}$so by deriving a new distribution $\mathrm{D_1}$from D which makes the mistakes of $\mathrm{h_1}$more clear meaning that it will concentrate more on $\mathrm{x_3}$, then a classifier $\mathrm{h_2}$will be trained on $\mathrm{D_1}$but the classifier $\mathrm{h_2}$ correctly classify $\mathrm{x_1}$, $\mathrm{x_3}$but has a wrong classification on $\mathrm{x_2}$, so by combining $\mathrm{h_1}$ and $\mathrm{h_2}$ will have $\mathrm{x_1}$ classified right and some errors on $\mathrm{x_2}$and $\mathrm{x_3}$, then will derive a new distribution $\mathrm{D_2}$to make the mistakes of the combined classifiers more clear and train a new classifier $\mathrm{h_3}$from the distribution $\mathrm{D_2}$so $\mathrm{h_3}$ has a correct classification on $\mathrm{x_2}$and $\mathrm{x_3}$ then by combining $\mathrm{h_1}$, $\mathrm{h_2}$and $\mathrm{h_3}$we will have a perfect classifier.

3.3.2.1. **AdaBoost**

Introduced by Freund & Schapire (1997) The residual learning is implemented through the concept of sample weights. At the iteration b the training samples have the weight $\mathrm{w_i^b}$ for $\mathrm{i = 1, \dots N}$ which is equal to the current error of $\mathrm{E}\left(\hat{\mathrm{f}}(\mathrm{x_i})\right)$ then the decision tree $\hat{\mathrm{f}}^{\mathrm{b}}(\mathrm{x})$ is fit to minimize the error $\mathrm{E_b} = \sum_{\mathrm{i=1}}^{\mathrm{N}} \mathrm{E}\left(\hat{\mathrm{F}}(\mathrm{x}) + \lambda_{\mathrm{b}}\hat{\mathrm{f}}^{\mathrm{b}}(\mathrm{x})\right)$, the

AdaBoost algorithm uses an exponential error/loss function $E\left(\hat{F}(x_i)\right) = e^{-y_i \hat{F}(x_i)}$, the choice of the error function $E(\,)$ also determines the way weights $w_i^b$and the weighting parameter λ_b is updated at different steps. The misclassified samples get a higher weight than the correctly classified samples, the weights attached to samples are used to inform the training of the weak learner in this case the DT to be grown in a way that favor splitting the sets of samples with high weights. Then the model is modified to include a learning rate γ as a regularization term that shrinks the contribution of all the models.
The model is updated as $\hat{F}(x) \leftarrow \hat{F}(x) + \gamma\lambda_b \hat{f}^b(x)$. The main tuning parameters of AdaBoost are the learning rate γ, the number of estimators used B and the decision tree related parameters like depth of the tree d and the number of samples per split. In summary by having a classifier $h_1(x)$ trained from the training dataset, then after weighing the misclassified samples we will have $h_2(x)$, then $h_3(x)$ until $h_M(x)$ then the final classifier will be combined majority weighted vote to produce the final classifier
$h(x) = \text{sign}[\sum_{m=1}^{M} \alpha_m h_m(x)]$.

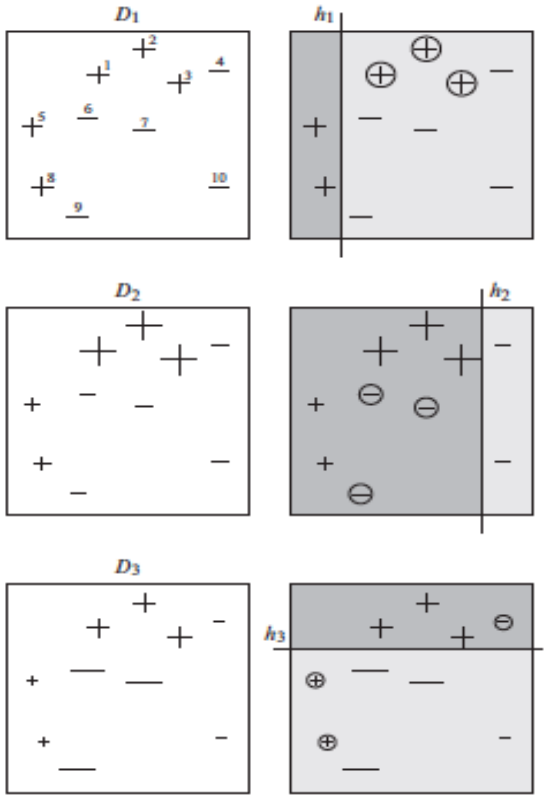

Fig.3.7. AdaBoost (Schapire, R. E. and Freund, 2012)

AdaBoost Algorithms has many extensions one of them is Adaboost.M1 (Alfaro, Gamez, & García, 2013). Adaboost.M1 can be defined as: by having a training set $T_n = \{(x_1, y_1), \ldots, (x_i, y_i), \ldots, (x_n, y_n)\}$, where y_i is the label which takes values of 1,2,…, k then the weight $w_b(i)$ is assigned to each observation x_i and is initially set to $1/n$ and the weight value will be updated after each iteration. A basic classifier $C_b(x_i)$ is built on the new training set $T_b$and is applied to every training step the classifier error is $e_b = \sum_{i=1}^{n} w_b(i)\, I(C_b(x_i) \neq y_i)$, $I(.)$is the indicator function which outputs 1 if the inner argument is true and 0 otherwise. From the error in the b-the iteration the constant $\alpha_b$is calculated and used for weight updating, according to Freund and Schapire $\alpha_b = \ln((1 - e_b)/e_b)$, then the new weight for the $(b+1)$-th iteration will be $w_{b+1}(i) = w_b(i)\exp\left(\alpha_b I(C_b(x_i) \neq y_i)\right)$, this process is repeated every step for $b = 1,2,\ldots,B$. Finally, the ensemble classifier calculates for each class the weighted sum of its votes, then the class with the majority votes is assigned $C_f(x_i) = \arg\max_{j \in Y} \sum_{b=1}^{B} \alpha_b I(C_b(x_i) = j)$

AdaBoost.M1 Algorithm:

1- Start with $w_b(i) = 1/n$,$i = 1,2, \dots, n$

2- Repeat for $b = 1,2, \dots, B$

a) Fit the classifier $C_b(x_i) = \{1,2, \dots, k\}$ using weights $w_b(i)$ on T_b

b) Compute $e_b = \sum_{i=1}^{n} w_b(i)\, \mathbf{I}(C_b(x_i) \neq y_i)$ and $\alpha_b = \ln((1 - e_b)/e_b)$

c) Update the weights $w_{b+1}(i) = w_b(i)\exp(\alpha_b \mathbf{I}(C_b(x_i) \neq y_i))$ and normalize them

3- Output the final classifier $C_f(x_i)$ =arg $\max_{j \in Y} \sum_{b=1}^{B} \alpha_b \mathbf{I}(C_b(x_i) = j)$

Algo.6. AdaBoost.M1 Algorithm

3.3.2.2. **Gradient Boosting Machine**

First introduced by Friedman (2001) which is an ensemble technique that improves the predictive power of the classifier by reducing the model bias through iteratively minimizing the error. It combines gradient-based optimization and boosting, where gradient-based optimization uses gradient computations to minimize a model's loss function with respect to the training dataset. Gradient boosting is a gradient descent technique, where the general idea of gradient descent is to tune the parameters in a sequential way either to maximize or minimize a loss function. Gradient descent can be applied to any differentiable function. By using the stochastic gradient descent one can overcome the problem of having many plateaus and off the local minima of the loss function (Mandt, D. Hoffman, & M. Blei, 2017). The initial model should be weak like a decision stump. Friedman (2001) recommends using 6-9 nodes trees as a base classifier. Stochastic gradient boosting was also introduced by Friedman (2002) to reduce the variance results from gradient boosting by adding randomness in the used algorithm. Meaning that to take subsamples of the training dataset for each iteration. It has many benefits: It will reduce the computational power needed, also it will reduce over-fitting and so it will help in identifying the number of trees needed (De'ath, 2007). As being pointed out by Lemmens & Croux (2006), GBM achieved better results in predicting churn, and so it will be used in our model. GBM requires three hyperparameters adjustments: number of trees used, max depth of each tree in the model and the learning rate. Hyperparameter optimization for the GBM will be performed using H2O random search function.

3.3.3. **Stacked Ensembles**

As mentioned above ensemble learning uses multiple models to improve the prediction power of the model, instead of using a single classifier or even ensemble to predict, we can combine multiple ensembles in one model by using Stacking method, which is a supervised method that finds the optimal combination of different classifiers. Stacking was first introduced by Wolpert (1992), then its theoretical guarantees were proved by (J, C, & E, 2007), its main idea is to use the classification of different classifiers instead of the original features, while the original target label remains the same. Fig 4.6 presents the stacking ensemble workflow.

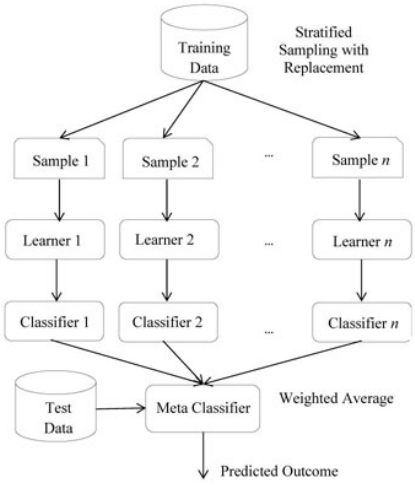

Fig.3.8. Stacking Ensemble (Sikora & Al-Laymoun, 2014)

Stacking mainly consists of two levels: The first level which is known as base learner (level-0), usually different classifiers are learned from the original features of the dataset, then the predictions of each of the classifiers are grouped to create a new dataset. In the new dataset, each data point is associated with the real label and the classifications from the previous step. The second level (level-1) is where a learning algorithm called meta-learner is used to predict the final output (Graczyk, Lasota, Trawiński, & Trawiński, 2010). To increase the performance of the stacking ensemble, class probabilities can be used instead of the predicted class from level-0 to create the new, to take into account the confidences of each of (level-0) classifiers (Menahem, Shabtai, Rokach, & Elovici, 2009; Ming Ting & Witten, 2002). After the hyperparameter optimization of RF and GBM using random search, they will be used to construct the stacking ensemble. The results of the stacking ensemble and its contribution to the model is presented in the result section.

3.4. RESAMPLING TECHNIQUES

The most common challenge in predictive modelling is that the model will learn the train set but will perform badly on the test set that's why the test set error and train set error is very crucial for the generalizability of the model, the common method in validation of the model is by using either bootstrap or using K-fold validation to validate the model (Steyerberg et al., 2001).

3.4.1. **Leave-One-Out- Cross-Validation**

Instead of creating two separated subsets for training and testing, a single data point is selected (x_1, y_1) for the validation of the algorithm, and the remaining data points $\{(x_2, y_2), \dots, (x_n, y_n)\}$ are used to train the model. Meaning that the model will be trained on the $n-1$ data points, and the prediction $\hat{y}_1$ is made to estimate the leave out data point. Because the test data point (x_1, y_1) was not used in training the model, the estimate error of that data point will be unbiased for the test error, however it will be a poor estimate of the error because it only depends on one data point (x_1, y_1). To overcome this issue, another data point will be selected (x_2, y_2) for the validation of the algorithm, and the model will be trained on the remaining $n-1$ data points then to compute the error based on that data point. This approach will be repeated n times and in each time a different data point will be used to validate the model, then all these errors will be averaged to compute the cross validation. The main drawback of this method that it is very time consuming especially with large number of observations this method can take hours. The next equation is the Error cross validation.

$$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i^{(-i)})$$

3.4.2. **K-Fold Cross-Validation**

Another resampling technique being used to assess the quality of the model, which involves splitting the dataset into K equal folds of equal sizes, where the first fold "also known as held out set" will be treated as validation set and the remaining $K-1$ folds will be the training set. This approach will be repeated K times, where in every time a different fold will be assigned. Usually the used K is 10 folds (Reitermanov, 2010). In this thesis 10-fold validation will be used to cross validate the training set performance.

The K-Fold Algorithm:

Input of the dataset T and number of folds k and performance function $error$ and for the computational models $L_1, L_2, \dots, L_m, m \geq 1$.

Divide the dataset T into k equal subsets: $T_1, \dots, T_k$.

$i = 1, \dots, k$:

$T_v \leftarrow T_i, T_{tr} \leftarrow \{T/T_i\}$, For $j = 1, \dots, m$: Train model L_j on T_{tr} and use T_v for assessing the performance where $E_v^j(i) = error\left(L_j(T_v)\right)$ when the $E_v^j(i)$ is satisfied then the stop-criterion is achieved and the training will stop.

For $j = 1, \dots, m$ the model performance will be evaluated by $E_v^j = \frac{1}{k} \cdot \sum_{i=1}^{k} E_v^j(i)$.

Algo.7. K-Fold Algorithm

3.5. Evaluation and Hyperparameter Optimization

3.5.1. **Model Evaluation using AUC of the ROC curve**

To assess the quality and accuracy of the trained classifier, many measures has developed to test the model performance to predict its performance on an unseen data.

For classification machine learning tasks, a confusion matrix is constructed to assess the model quality, it categorizes the model predictions according to whether they match the correct label or not. It has mainly four values:

TP: True positive (A positive example, classified as positive example)

TN: True negative (A negative example, classified as negative example)

FP: False positive (A negative example, but classified as positive example)

FN: False negative (A positive example, but classified as negative example)

	Predicted Class Positive	**Predicted Class Negative**
Actual Class Positive	True Positive TP	False Negative FN
Actual Class Negative	False Positive FP	True Negative TN

Table 2 Confusion matrix

Success rate was introduced to assess the model ability to predict true positives and true negatives "also known as accuracy"

$$Success\ rate = \frac{TP + TN}{TP + TN + FP + FN}$$

The error rate measure was introduced to assess the proportion of false positives and false negatives:

$$Error\ Rate = \frac{FP + FN}{TP + TN + FP + FN}$$

Sensitivity "also known as true positive rate" was introduced to assess the model ability to correctly classify the positive examples as positive:

$$Sensitivity = \frac{TP}{TP + FN}$$

While Specificity "also known as true negative rate" measures the model ability to correctly classify the negative examples as negative:

$$Specificty = \frac{TN}{TN + FP}$$

To visualize the model performance, ROC curve was introduced to visually examine the trade-off between sensitivity and specificity. Which become one of the most powerful used algorithms to measure the model quality (Fawcett, 2006). The main idea of the ROC curve is the plot of the specificity of the algorithms which is the percentage of the correctly classified negatives against the sensitivity which is the percentage of the correctly classified positives of the algorithm (Fawcett, 2003). The Roc curve has a diagonal line which represents a random guess model meaning that the model cannot differentiate between true positives and false positives, this diagonal line can be considered as the baseline where models can be judged. The best model has a curve that passes through the top left corner "100% Sensitivity" and has 0% false positive rate.

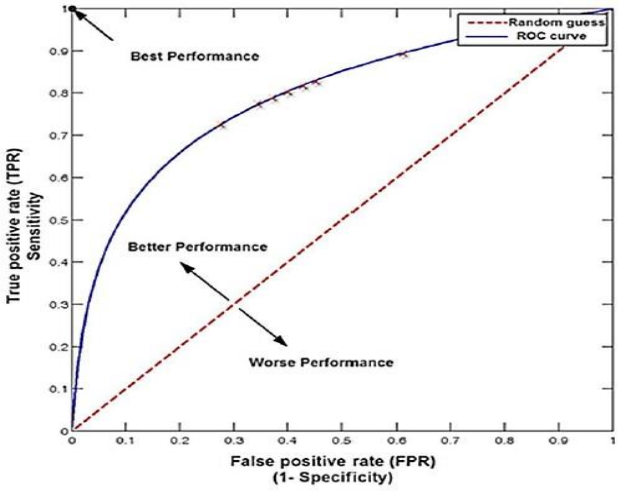

Fig.3.9. ROC curve (Hassouna et al., 2015)

To measure the quality of the model using ROC curve a statistic known as AUC "Area under the ROC curve" is used, which treats the ROC diagram as a two-dimension square and measures the area under the curve. AUC has a minimum value of 0.5 and the maximum value of 1, where 0.5 represent a model with no predictive power and 1 represents a model with 100% predictive power. The AUC is calculated by the following formula (Vuk, 2006)

$$\mathrm{AUC} = \int_0^1 \frac{\mathrm{TP}}{\mathrm{P}}\, \mathrm{d}\frac{\mathrm{FP}}{\mathrm{N}} = \frac{1}{\mathrm{PN}} \int_0^{\mathrm{N}} \mathrm{TP}\ \mathrm{dFP}$$

Where $TP + FN = P$ and $TN + FP = N$

3.5.2. **Hyperparameter Optimization**

Hyperparameter Optimization is used to adjust the model parameters but not its internal model parameters, which is what the model will learn from the dataset itself. Hyperparameter space is a set of all values of the model external parameters. The process of tuning the model parameters to give the best accuracy in the shortest time is called Hyperparameter Optimization. Two main hyperparameter optimization algorithms have been introduced by authors Bergstra, Bardenet, Bengio, & Kégl (2011): traditional grid search and random grid search. The traditional grid search searches all the possible grid for the algorithm, but its main drawback is that it will take a lot of time and computational resources especially for deep datasets. While the random grid search will take less time and so less computational power. As being noted by authors Bergstra & Bengio (2012) in their empirical study that random search performs better results than grid search for the same time frame. In this thesis random search will be applied to the RF and GBM using H2O random search function, where it will search for the number of trees and max depth needed for the RF to achieve better performance. In addition to the number of trees and max depth, this function will search for the best learning rate for the GBM to achieve the best results.

3.6. THE PROPOSED ENSEMBLE SYSTEM

Fig.6.3. presents the proposed workflow that will be followed. The initial step is data-preprocessing (Fig.3.2.) which will remove noise data points and redundant features from the dataset. Data pre-processing stage consists of removing completely empty features under assumption that these features cannot be imputed, then high cardinality categorical features will be treated either by removing the features with more than 500 unique level or by combine levels with less than 5% appearance in the dataset, then missing values will be imputed, then the most important features will be selected based on feature importance matrix of RF and GBM. The final step of the pre-processing stage is treating the imbalance in the dataset. The output of data pre-processing stage will be noise free data which will be used for further modelling stages. The next stage is sampling the dataset, the sampling is done to split the dataset into two disjoint subsets, one subset will be used to train the model and using K-fold validation to validate the result of the training set. The second subset will be used to test the model and predict its generalizability. Mainly the sampling stage is done to prevent the over fitting of the model and to increase its generalizability. The last stage will be training the dataset using Stacking over the algorithms using H2O platform (H2O.ai, 2014).

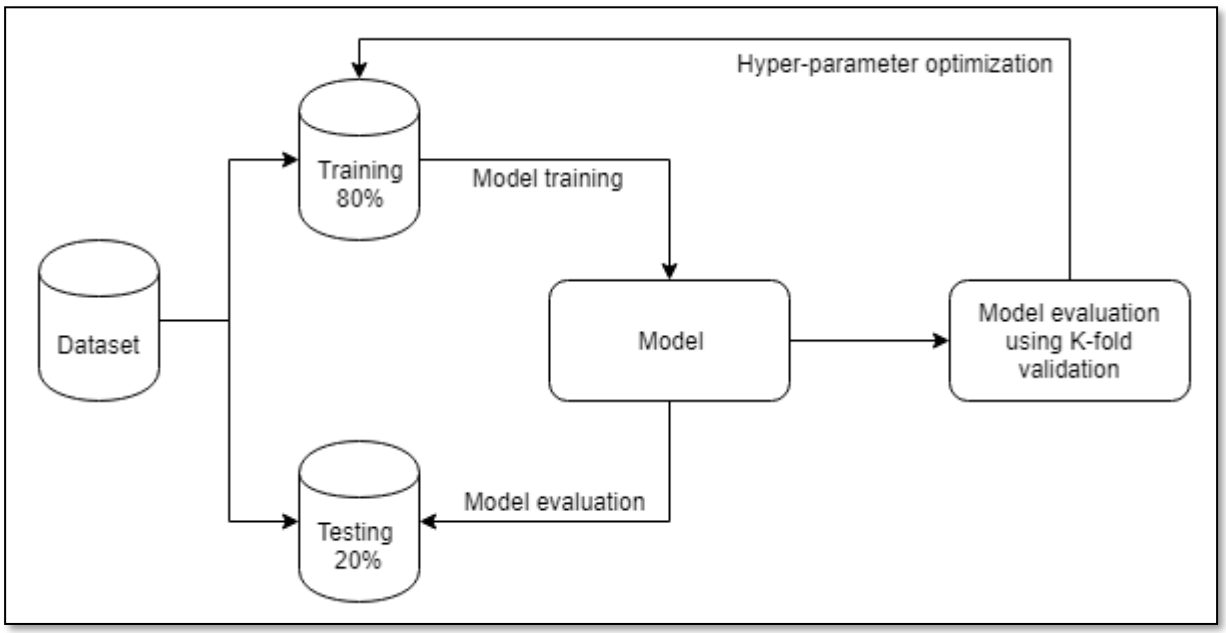

Fig.3.10. The Flowchart of the proposed model.

3.7. H2O PLATFORM

H2O (H2O.ai, 2014) is a machine learning platform implemented in R and Python, this platform uses pure Java language as its backend. H2O base algorithms are distributed across all the cores in a multi-node cluster (LeDell, 2015) which facilitate the large computational power needed for ensemble techniques like gradient boost machine and Random Forests. Due to its easily and rapid implementation, H2O platform is used as a main R Library to model this large and heterogeneous dataset.

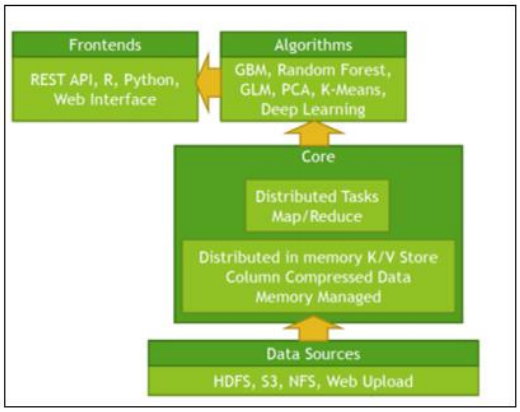

Fig.3.11. H2O Platform architecture (Ha, Nguyen Ha, & Nguyen Thi Bao, 2017)

4. RESULTS

This section discusses the results obtained from the presented methodology. (Section 7.1) Represents the results obtained from the preprocess step, (Section 7.2) Represents the final results obtained. All the results are assessed by AUC of the ROC curve as proposed by the challenge committee

4.1. PREPROCESS RESULTS

The results of the preprocess step are presented in this section. The dataset contains three different labels: Churn, Appetency and Up-selling. Each of these labels will be used to construct a separate dataset and every dataset will be preprocessed in different manner. For every step a 10-fold validation was used and 80% of the dataset was used to train the model and the rest 20% was used as a test set to assess the performance

4.1.1. **Imputing Missing Values for the three datasets**

We start preprocessing this dataset by comparing different techniques to impute missing values. Table 7.1 shows the performance of different missing values imputation techniques in combination with three base classifiers and their results were assessed by using AUC of the ROC curve after the imputation. The best performance in each column is in bold. The results show that different techniques yielded different results with different classifiers.

4.1.1.1. **Imputing missing values for Churn dataset**

Different imputing methods was performed on the churn dataset to impute the missing values, to assess the performance of each imputation technique the AUC of the ROC curve of each technique is measures against **C4.5 DT, RF and GBM** as shown in Table 3**.** The highest AUC was achieved by Hot-Deck imputation technique and so this technique will be used to impute the missing value in the churn dataset.

	Mean	Hot-Deck	Hot-Deck with Indicators	Mice
C4.5	0.5061	0.5116	**0.5231**	0.5169
RF	0.6407	0.6531	**0.6888**	0.6398
GBM	0.6683	0.6794	**0.7162**	0.6641

Table 3 The AUC results for Churn Dataset Missing Value Imputation

4.1.1.2. **Imputing missing values for Appetency dataset**

The same technique is used for Appetency dataset; the best performing technique was Hot-Deck imputation. So, this technique will be used to impute the missing value in the Appetency dataset. The results are shown in Table 4.

	Mean	Hot-Deck	Hot-Deck with Indicators	Mice
C4.5	0.5	0.5	0.5	0.5
RF	0.7464	**0.7686**	0.7445	0.7407
GBM	0.8027	**0.8107**	0.7975	0.8067

Table 4 The AUC results for Appetency Missing Value Imputation

4.1.1.3. **Imputing missing values for Up-Selling dataset**

The same technique is used for Up-Selling dataset, the best performing technique was MICE imputation. So, this technique will be used to impute the missing value in the Appetency dataset. The results are shown in Table 5.

	Mean	Hot-Deck	Hot-Deck with Indicators	Mice
C4.5	**0. 6969**	0. 6465	0. 6934	0. 6262
RF	0. 8115	0. 8334	0. 8229	**0. 8380**
GBM	0. 8394	0. 8544	0. 8445	**0. 8579**

Table 5 The AUC results for Up-Selling Missing Value Imputation

4.1.2. **Balancing the datasets**

This subsection discusses different algorithms to balance the datasets, three algorithms were used and their results were assessed by using AUC after the balancing. The algorithm with the highest AUC will be used to balance the dataset.

4.1.2.1. **Balancing the Churn Dataset**

Different balancing algorithms were used to balance the dataset. To assess the performance of each algorithm two classifiers (RF and GBM) were used and their AUC were measured before and after applying each algorithm. The initial AUC of the RF was 0.6796 and for GBM was 0.7032306. After applying SMOTE the AUC results increased by more than 10% which can indicate that SMOTE algorithm overfitted the dataset which is common for SMOTE algorithm and so the OSS was used instead for balancing the dataset. The results are shown in Table 6.

	SMOTE	TOMEK	OSS
RF	0.8565	0.6533	**0.**6592
GBM	0.8414	0.6865	**0.**6818

Table 6 The AUC results for Churn Dataset Balancing

4.1.2.2. **Balancing the Appetency Dataset**

The same technique is used for Appetency dataset. The initial AUC of the RF was 0.7760 and for GBM was 0.7814. SMOTE algorithm behaved the same as Churn Dataset and so the OSS was used to balance the Appetency dataset. Results are shown in Table 7.

	SMOTE	TOMEK	OSS
RF	0.8830	0.7522	**0.**7608
GBM	0.8774	0.7744	**0.**7834

Table 7 The AUC results for Appetency Dataset Balancing

4.1.2.3. **Balancing the Up-Selling dataset**

The same technique is used for Up-Selling dataset. The initial AUC of the RF was 0.8172 and for GBM was 0.8427. SMOTE algorithm behaved the same as Churn Dataset and so the OSS was used to balance the Appetency dataset. Results are shown in Table 8.

	SMOTE	TOMEK	OSS
RF	0. 8754	0. 8358	**0.** 8345
GBM	0. 8692	0. 8513	**0.** 8583

Table 8 The AUC results for Up-Selling Dataset Balancing

4.1.3. **Feature Selection**

This subsection discusses different algorithms to select the most important features in the dataset, the feature importance matrix of RF and GBM was used to decide which features are more important for the model to be selected. The features that will score the highest AUC will be used.

4.1.3.1. **Feature Selection of the Churn Dataset**

The initial AUC for RF was 0.6796 and for GBM was 0.7032. Using Random Forests to select the most important 25 features in the dataset, then to run RF on these important features and to assess the AUC. Also, the GBM is used to assess the predictive power of these features by running the algorithm on the original dataset and the reduced dataset. Feature selected using GBM achieved higher AUC than RF and so it will be selected for feature Selection. Results are shown in Table 9.

	Feature Selection using Random Forests	Feature Selection using Gradient Boosting Machine
RF	0. 6153	0. 6635
GBM	0. 6482	0. 6935

Table 9 The AUC results for Churn Dataset after Feature Selection

4.1.3.2. **Feature Selection of the Appetency Dataset**

The initial AUC for RF was 0.7760 and for GBM was 0.7814. Using Random Forests to select the most important 25 features in the dataset, then to run RF on these important features and to see the AUC. Also, the GBM is used to assess the predictive power of these features by running the algorithm on the original dataset and the reduced dataset. Feature selected using GBM achieved higher AUC than RF and so it will be selected for feature Selection. Results are shown in Table 10.

	Feature Selection using Random Forests	Feature Selection using Gradient Boosting Machine
RF	0. 6953	0. 7579
GBM	0. 7158	0. 7712

Table 10 The AUC results for Appetency Dataset after Feature Selection

4.1.3.3. **Feature Selection of the Up-Selling Dataset**

The initial AUC for RF was 0. 8172661 and for GBM was 0. 8427427. Using Random Forests to select the most important 25 features in the dataset, then to run RF on these important features and to see the AUC. Also, the GBM is used to assess the predictive power of these features by running the algorithm on the original dataset and the reduced dataset. Feature selected using Random Forests achieved higher AUC than GBM and so it will be selected for feature Selection. Results are shown in Table 11.

	Feature Selection using Random Forests	Feature Selection using Gradient Boosting Machine
RF	0. 8251	0. 8256
GBM	0. 8423	0. 8380

Table 11 The AUC results for Up-Selling Dataset after Feature Selection

4.2. FINAL MODEL AUC RESULTS

This section discusses the results obtained after applying Random Forests, GBM and Stacking them to the cleaned dataset obtained from the preprocess step. Every classifier is applied to the three datasets and the AUC of the ROC curve was measured. From Table 12 we can see that Stacking did increase the performance of the classifiers by an average of 2%.

	Churn	Appetency	Up-Selling	Average
Random Forest	0.6635	0.7579	0.8251	0.7488
GBM	0.6935	0.7712	0.8423	0.769
Stacking	0.7111	0.78	0.845	0.7787

Table 12 The AUC results of the Final model

4.3. ANALYSIS OF THE RESULTS

The machine used was Core i5 with 8GB RAM and all the experiments have been conducted using R statistical software. Training and testing ratio for each dataset was fixed at 80:20 ratios. Class imbalance was handled only for the training datasets. Testing datasets were kept separated all over the experiment and were used only in the end to assess the performance of the final classifiers. The present research work aims at identifying an appropriate workflow to deal with a challenging dataset that contains missing values, imbalance, high cardinality and of mixed nature. Also, this work aims at comparing the performance of different decision trees-based ensemble techniques to detect churn. After analyzing the results, it has shown that ensemble techniques improve the performance of the classifiers and the stacking of these ensembles further improved the results. Churn dataset was the hardest dataset whose AUC was the lowest compared to the other two datasets. Imputation techniques used increased the performance of the classifiers with different measures, the best performing technique was the hot-deck imputation technique which yielded good results for Churn and appetency datasets, while MICE gave the best performance for the Up-Sell dataset. The best performing balancing technique was the OSS which gave a descent increase in the AUC without overfitting the datasets. The feature importance of GBM yielded an outstanding result compared to the Random Forests for churn and appetency datasets while Random Forests was the same as GBM for the up-sell dataset. The Stacking the ensembles increased a lot the classifiers performance. The final average AUC result was 0.7787 which will

place us in the top 10% of this competition. The following table presents the top winner using the same dataset.

Prize	Team	Country	Fast track		Slow track	
			Rank	Score	Rank	Score
1	IBM Research	USA	1	0.8493	1	0.8521
2	ID Analytics	USA	2	0.8448	3	0.8479
3	David Slate & Peter Frey	USA	3	0.8443	8	0.8443
1	University of Melbourne	Australia	27	0.8250	2	0.8484
2	Financial Engineering Group, Inc.	Japan	4	0.8443	4	0.8477
3	National Taiwan University	Taiwan	20	0.8332	5	0.8461

Table 13 The KDD 2009 winners (Guyon, Lemaire, Boullé, Dror, & Vogel, 2009)

4.4. KEY FINDINGS

- Multiple methods for imputation produces better results than single imputation methods.
- SMOTE should be treated with careful cause in some cases it can cause overfitting.
- Wrapper methods for feature selection can take less computational effort and produce better results than filter methods.
- Tree-based ensembles produced better results than single trees through decreasing the variance in the dataset.
- Churn can be predicted by using tree-based ensembles with high accuracy.
- H2O R package is a powerful tool that's more than capable of handling large, deep and heterogeneous datasets.

5. CONCLUSION AND FUTURE WORK

For future work, it would be useful to explore the deep learning methods using neural networks as it also been reported to be effective in dealing with big data. The challenges of predicting customer churn in telecommunication sector are mainly the high imbalance in the dataset between the churners and the loyal customers, also the presence of high cardinality categorical features and the fact that the dataset will be deep and large to be tackled by the normal machine learning algorithms.
The possibility of customers to churn from the telecommunication companies has emerged as to be a real threat for telecommunication companies because of the lost revenue and the high cost of acquisitions of new customers. And so, the churn prediction right now is vital for telecommunication companies, even thought that churn is inevitable, it can be managed and predicted so a corrective measure can be taken to keep it at a safe level.

In this paper a comparison was made between different supervised machine learning algorithms to detect customer churn for telecommunication sector using the KDD 2009 Orange dataset. Gradient boosted trees significantly outperformed other decision trees ensembles like random forest and AdaBoost. Moreover, a preprocess technique was proposed to better deal with large dataset with a lot of missing values, high cardinality categorical features and high dimensionality. It was shown that using the proposed methodology the telecommunication companies can gain a higher accuracy for detecting the churners and so to be able to strengthen the ability of the company for customer retention. Additionally, a better dimensionality reduction technique was introduced to select the most important features from the dataset.

This paper has shown the performance of decision tree machine learning algorithms and the advantage of ensemble decision trees as well. Not only the methodology proposed in this paper can be used by telecommunication companies but also can be tested on any classification machine learning problems, like loan default or fraud detection. In future work beside exploring the deep learning methods, we plan to investigate additional preprocess steps to better improve the accuracy of the algorithms and to decrease the computational power needed and to explore the performance of other boosting algorithms beyond Gradient Boosting machines. Also, to use a larger dataset with meta information from the telecommunication industry to maximize the predictive power of our results.

6. BIBLIOGRAPHY

Alfaro, E., Gamez, M., & García, N. (2013). adabag: An R Package for Classification with Boosting and Bagging. *Journal of Statistical Software; Vol 1, Issue 2 (2013)* . https://doi.org/10.18637/jss.v054.i02

Ali, Ö., & Arıtürk, U. (2014). *Dynamic churn prediction framework with more effective use of rare event data: The case of private banking*. *Expert Systems with Applications* (Vol. 41). https://doi.org/10.1016/j.eswa.2014.06.018

Azeem, M., Usman, M., & Fong, A. C. M. (2017). A churn prediction model for prepaid customers in telecom using fuzzy classifiers. *Telecommunication Systems*, *66*(4), 603–614. https://doi.org/10.1007/s11235-017-0310-7

Azevedo, A., & Filipe Santos, M. (2008). *KDD, semma and CRISP-DM: A parallel overview*.

Azur, M., Stuart, E., Frangakis, C., & Leaf, P. (2011). Multiple imputation by chained equations: what is it and how does it work? *International Journal of Methods in Psychiatric Research*, *20*(1), 40–49. https://doi.org/doi: 10.1002/mpr.329

Batista, G., & Monard, M.-C. (2003). An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence*, *17*(5–6), 519–533. https://doi.org/10.1080/713827181

Batista, G., Prati, R., & Monard, M. (2004). A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data. *SIGKDD Explor. Newsl.*, *6*(1), 20–29. https://doi.org/doi: 10.1145/1007730.1007735

Bauer, E., & Kohavi, R. (1999). An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. *Machine Learning*, *36*(1), 105–139. https://doi.org/10.1023/A:1007515423169

Bellman, R. (1961). *Adaptive Control Processes*. Princeton University Press. Retrieved from citeulike-article-id:2662577

Bergstra, J. S., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for Hyper-Parameter Optimization. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 24* (pp. 2546–2554). Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimization.pdf

Bi, W., Cai, M., Liu, M., & Li, G. (2016). A Big Data Clustering Algorithm for Mitigating the Risk of Customer Churn. *IEEE Transactions on Industrial Informatics*, *12*(3), 1270–1281. https://doi.org/10.1109/TII.2016.2547584

Borbora, Z. H., & Srivastava, J. (2012). User Behavior Modelling Approach for Churn Prediction in Online Games. In *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing* (pp. 51–60). https://doi.org/10.1109/SocialCom-PASSAT.2012.84

Breiman, L. (1996). Bagging Predictors. *Machine Learning*, *24*(2), 123–140. https://doi.org/10.1023/A:1018054314350

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Breiman, L., Friedman, J., Stone, C., & Olshen, R. A. (1984). *Classification and Regression Trees (Wadsworth Statistics/Probability)*. Chapman & Hall/CRC. Retrieved from citeulike-article-id:801011

Brodley, C. E., & Utgoff, P. E. (1995). Multivariate Decision Trees. *Machine Learning*, *19*(1), 45–77. https://doi.org/10.1023/A:1022607123649

Brodley, C., & Utgoff, P. (1992). *Multivariate versus Univariate Decision Tree*. Amherst, Massachusetts:

Department of Computer Science, University of Massachusetts. Retrieved from citeulike-article-id:10466882

Burez, J., & Van den Poel, D. (2009). Handling class imbalance in customer churn prediction. *Expert Systems with Applications*, *36*(3, Part 1), 4626–4636. https://doi.org/https://doi.org/10.1016/j.eswa.2008.05.027

Casari, A., & Zheng, A. (2018). *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*. O'Reilly Media.

Castro, E. G., & Tsuzuki, M. S. G. (2015). Churn Prediction in Online Games Using Players' Login Records: A Frequency Analysis Approach. *IEEE Transactions on Computational Intelligence and AI in Games*, *7*(3), 255–265. https://doi.org/10.1109/TCIAIG.2015.2401979

Chawla, N. V, Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE : Synthetic Minority Over-sampling Technique, *16*, 321–357.

Chen, C., Liaw, A., & Breiman, L. (2004). Using Random Forest to Learn Imbalanced Data.

Coussement, K., Lessmann, S., & Verstraeten, G. (2017). A comparative analysis of data preparation algorithms for customer churn prediction : A case study in the telecommunication industry. *Decision Support Systems*, *95*, 27–36. https://doi.org/10.1016/j.dss.2016.11.007

De'ath, G. (2007). *Boosted Trees for Ecological Modeling and Prediction*. *Ecology* (Vol. 88). https://doi.org/10.1890/0012-9658(2007)88[243:BTFEMA]2.0.CO;2

Dietterich, T. G., & Bae, E. J. (1995). Machine Learning Bias, Statistical Bias, and Statistical Variance of Decision Tree Algorithms, *3*.

Domingos, P. (2000). A unified bias-variance decomposition and its applications. In *In Proc. 17th International Conf. on Machine Learning* (pp. 231–238). Retrieved from citeulike-article-id:4031882

Enders, C. K. (2010). *Applied missing data analysis*. New York: Guilford Press.

Ertel, W., Black, N., & Mast, F. (2017). *Introduction to artificial intelligence* (Second edition). Springer International Publishing.

Fawcett, T. (2003). ROC Graphs : Notes and Practical Considerations for Data Mining Researchers ROC Graphs : Notes and Practical Considerations for Data Mining Researchers.

Fawcett, T. (2006). An introduction to ROC analysis, *27*, 861–874. https://doi.org/10.1016/j.patrec.2005.10.010

Freund, Y., & Schapire, R. E. (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, *55*(1), 119–139. https://doi.org/https://doi.org/10.1006/jcss.1997.1504

Friedman, J. (2001). Greedy Function Approximation: A gradient boosting machine. *The Annals of Statistics*, *29*(5), 1189–1232. Retrieved from citeulike-article-id:525214

Friedman, J. (2002). Stochastic Gradient Boosting. *Comput. Stat. Data Anal.*, *38*(4), 367–378. https://doi.org/doi: 10.1016/s0167-9473(01)00065-2

García, D. L., Nebot, À., & Vellido, A. (2017). Intelligent data analysis approaches to churn as a business problem : a survey. *Knowledge and Information Systems*, *51*(3), 719–774. https://doi.org/10.1007/s10115-016-0995-z

Graczyk, M., Lasota, T., Trawiński, B., & Trawiński, K. (2010). Comparison of Bagging, Boosting and Stacking Ensembles Applied to Real Estate Appraisal BT - Intelligent Information and Database Systems. In N. T.

Nguyen, M. T. Le, & J. Świątek (Eds.) (pp. 340–350). Berlin, Heidelberg: Springer Berlin Heidelberg.

Graham, J. (2009). Missing Data Analysis: Making It Work in the Real World. *Annual Review of Psychology*, *60*(1), 549–576. https://doi.org/doi: 10.1146/annurev.psych.58.110405.085530

Guyon, I., Lemaire, V., Boullé, M., Dror, G., & Vogel, D. (2009). Analysis of the KDD Cup 2009: Fast Scoring on a Large Orange Customer Database. In G. Dror, M. Boullé, I. Guyon, V. Lemaire, & D. Vogel (Eds.), *Proceedings of KDD-Cup 2009 Competition* (Vol. 7, pp. 1–22). New York, New York, USA: PMLR. Retrieved from http://proceedings.mlr.press/v7/guyon09.html

Guyon, I., Lemaire, V., Vogel, D., Guyon, I., Lemaire, V., & Vogel, D. (2009). The 2009 Knowledge Discovery and Data Mining Competition (KDD Cup 2009) Analysis of the KDD Cup 2009 : Fast Scoring on a Large Orange Customer Database, (June 2014).

H2O.ai. (2014). H2O Machine Learning Platform.

Ha, S., Nguyen Ha, N., & Nguyen Thi Bao, H. (2017). *A hybrid feature selection method for credit scoring*. *EAI Endorsed Transactions on Context-aware Systems and Applications* (Vol. 4). https://doi.org/10.4108/eai.6-3-2017.152335

Hadden, J., Tiwari, A., Roy, R., & Ruta, D. (2007). Computer assisted customer churn management: State-of-the-art and future trends. *Computers and Operations Research*, *34*(10), 2902–2917. https://doi.org/10.1016/j.cor.2005.11.007

Han, J., & Kamber, M. (2000). *Data Mining: Concepts and Techniques*. *San Francisco: Morgan kaufmann;*

Hansen, J. V. (2000). *Combining Predictors: Meta Machine Learning Methods and Bias/Variance and Ambiguity Decompositions*.

Hassouna, M., Tarhini, A., Elyas, T., & Abou Trab, M. S. (2015). Customer Churn in Mobile Markets: A Comparison of Techniques. *International Business Research*, *8*(6), 224–237. https://doi.org/10.5539/ibr.v8n6p224

Hothorn, T., Hornik, K., & Zeileis, A. (2006). Unbiased Recursive Partitioning: A Conditional Inference Framework. *Journal of Computational and Graphical Statistics*, *15*(3), 651–674. https://doi.org/10.1198/106186006X133933

Hssina, B., Merbouha, A., Ezzikouri, H., & Erritali, M. (2014). A comparative study of decision tree ID3 and C4.5, (2), 13–19.

J, van der L. M., C, P. E., & E, H. A. (2007). Super Learner. *Statistical Applications in Genetics and Molecular Biology*. https://doi.org/10.2202/1544-6115.1309

Jayaswal, P., Prasad, B. R., & Agarwal, S. (2016). An Ensemble Approach for Efficient Churn Prediction in Telecom Industry. *International Journal of Database Theory and Application*, *9*(8), 211–232. https://doi.org/10.14257/ijdta.2016.9.8.21

Kalousis, A., Prados, J., & Hilario, M. (2007). Stability of feature selection algorithms: a study on high-dimensional spaces. *Knowledge and Information Systems*, *12*(1), 95–116. https://doi.org/10.1007/s10115-006-0040-8

Kamalraj, N., & Malathi, A. (2013). *A Survey on Churn Prediction Techniques in Communication Sector*. *International Journal of Computer Applications* (Vol. 64). https://doi.org/10.5120/10633-5373

Kass, G. V. (1980). An Exploratory Technique for Investigating Large Quantities of Categorical Data. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, *29*(2), 119–127. https://doi.org/10.2307/2986296

Khan, A. A., Jamwal, S., & Sepehri, M. M. (2010). Applying Data Mining to Customer Churn Prediction in an

Internet Service Provider. *International Journal of Computer Applications*, *9*(7), 8–14. https://doi.org/10.5120/1400-1889

Kim, H., & Loh, W.-Y. (2001). Classification Trees With Unbiased Multiway Splits. *Journal of the American Statistical Association*, *96*(454), 589–604. https://doi.org/10.1198/016214501753168271

Kim, J., Han, Y., & Lee, J. (2016). Data Imbalance Problem solving for SMOTE Based Oversampling : Study on Fault Detection Prediction Model in Semiconductor Manufacturing Process. *Advanced Science and Technology Letters*, (133), 79–84.

Kohavi, R., & Wolpert, D. (1996). Bias Plus Variance Decomposition for Zero-One Loss Functions. Retrieved from citeulike-article-id:4372274

Krawczyk, B., Torgo, L., Branco, P., & Moniz, N. (2017). *Influence of minority class instance types on SMOTE imbalanced data oversampling Przemyss law Skryjomski*. *Proceedings of Machine Learning Research* (Vol. 74). Retrieved from http://proceedings.mlr.press/v74/skryjomski17a/skryjomski17a.pdf

Kubat, M. (2015). An Introduction to Machine Learning. *Machine Learning Methods for Ecological Applications*, 1–237. https://doi.org/10.1007/978-3-319-20010-1

Kubat, M., & Matwin, S. (2000). *Addressing the Curse of Imbalanced Training Sets: One-Sided Selection*. *Fourteenth International Conference on Machine Learning*.

Kumar, V., & Sonajharia, M. (2014). *Feature Selection: A literature Review*. *The Smart Computing Review* (Vol. 4). https://doi.org/10.6029/smartcr.2014.03.007

Kurucz, M., Siklósi, D., Bíró, I., Csizsek, P., Fekete, Z., Iwatt, R., … Szabó, A. (2009). KDD Cup 2009 @ Budapest: Feature Partitioning and Boosting. In *Proceedings of the 2009 International Conference on KDD-Cup 2009 - Volume 7* (pp. 65–75). JMLR.org.

LeDell, E. (2015). Scalable Ensemble Learning and Computationally Efficient Variance Estimation. https://doi.org/10.1017/CBO9781107415324.004

Liao, H., Chen, K., Liu, D., & Chiu, Y. (2015). Customer Churn Prediction in Virtual Worlds. In *2015 IIAI 4th International Congress on Advanced Applied Informatics* (pp. 115–120). https://doi.org/10.1109/IIAI-AAI.2015.265

Liaw, A., & Wiener, M. (2002). Classification and Regression by randomForest. *R News*, *2*(3), 18–22. Retrieved from citeulike-article-id:1121494

Light, R. J., & Margolin, B. H. (1971). An Analysis of Variance for Categorical Data. *Journal of the American Statistical Association*, *66*(335), 534–544. https://doi.org/10.2307/2283520

Liu, Y., & Zhuang, Y. (2015). Research Model of Churn Prediction Based on Customer Segmentation and Misclassification Cost in the Context of Big Data, (June), 87–93. https://doi.org/10.4236/jcc.2015.36009

Lo, H.-Y., Chang, K.-W., Chen, S.-T., Chiang, T.-H., Ferng, C.-S., Hsieh, C.-J., … Vogel, D. (2009). *An ensemble of three classifiers for KDD cup 2009: Expanded linear model, heterogeneous boosting, and selective naıve Bayes* (Vol. 7).

Loh, W.-Y. (2011). Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *1*(1), 14–23. https://doi.org/10.1002/widm.8

Loh, W.-Y., & Shih, Y.-S. (1997). Split Selection Methods for Classification Trees. *Statistica Sinica*, *7*(4), 815–840. https://doi.org/10.2307/24306157

Loh, W.-Y., & Vanichsetakul, N. (1988). Tree-Structured Classification Via Generalized Discriminant Analysis. *Journal of the American Statistical Association*, *83*(403), 715–725. https://doi.org/10.2307/2289295

Mahajan, V., Misra, R., & Mahajan, R. (2015). Review of Data Mining Techniques for Churn Prediction in Telecom. *Journal of Information and Organizational Research*, *39*(2), 183–197.

Mandt, S., D. Hoffman, M., & M. Blei, D. (2017). *Stochastic Gradient Descent as Approximate Bayesian Inference*. *Journal of Machine Learning Research* (Vol. 18).

Menahem, E., Shabtai, A., Rokach, L., & Elovici, Y. (2009). Improving malware detection by applying multi-inducer ensemble. *Computational Statistics & Data Analysis*, *53*(4), 1483–1494. https://doi.org/https://doi.org/10.1016/j.csda.2008.10.015

Messenger, R., & Mandell, L. (1972). A Modal Search Technique for Predictibe Nominal Scale Multivariate Analys. *Journal of the American Statistical Association*, *67*(340), 768–772. https://doi.org/10.2307/2284634

Micci-Barreca, D. (2001). *A Preprocessing Scheme for High-Cardinality Categorical Attributes in Classification and Prediction Problems. SIGKDD Explorations* (Vol. 3). https://doi.org/10.1145/507533.507538

Miller, H., Clarke, S., Lane, S., Lonie, A., Lazaridis, D., Petrovski, S., & Jones, O. (2009). Predicting customer behaviour: The University of Melbourne's KDD Cup report. In G. Dror, M. Boullé, I. Guyon, V. Lemaire, & D. Vogel (Eds.), *Proceedings of KDD-Cup 2009 Competition* (Vol. 7, pp. 45–55). New York, New York, USA: PMLR. Retrieved from http://proceedings.mlr.press/v7/miller09.html

Ming Ting, K., & Witten, I. (2002). *Issues in Stacked Generalization*. https://doi.org/10.1613/jair.594

Mitchell, T. (1980). The Need for Biases in Learning Generalizations, 184–191. Retrieved from citeulike-article-id:4058496

Mohanty, R., & Naga Ratna Sree, C. (2018). Churn and Non-churn of Customers in Banking Sector Using Extreme Learning Machine BT - Proceedings of the Second International Conference on Computational Intelligence and Informatics. In V. Bhateja, J. M. R. S. Tavares, B. P. Rani, V. K. Prasad, & K. S. Raju (Eds.) (pp. 51–58). Singapore: Springer Singapore.

Morik, K., & Köpcke, H. (2004). Analysing Customer Churn in Insurance Data – A Case Study BT - Knowledge Discovery in Databases: PKDD 2004. In J.-F. Boulicaut, F. Esposito, F. Giannotti, & D. Pedreschi (Eds.) (pp. 325–336). Berlin, Heidelberg: Springer Berlin Heidelberg.

Murthy, S. (1996). *On Growing Better Decision Trees from Data*. Johns Hopkins University. Retrieved from citeulike-article-id:8776546

Napierala, K., & Stefanowski, J. (2016). Types of minority class examples and their influence on learning classifiers from imbalanced data. *Journal of Intelligent Information Systems*, *46*(3), 563–597. https://doi.org/10.1007/s10844-015-0368-1

Ngonmang, B., Viennet, E., & Tchuente, M. (2012). Churn Prediction in a Real Online Social Network Using Local CommunIty Analysis. In *2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining* (pp. 282–288). https://doi.org/10.1109/ASONAM.2012.55

Niculescu-Mizil, A., Perlich, C., Swirszcz, G., Sind-Hwani, V., Liu, Y., Melville, P., … Vogel, D. (2009). Winning the KDD Cup Orange Challenge with Ensemble Selection. *JMLR: Workshop and Conference Proceedings*, *7*, 23–34. Retrieved from http://proceedings.mlr.press/v7/niculescu09/niculescu09.pdf

Noh, H., Kwak, M., & Han, I. (2004). *Improving the prediction performance of customer behavior through multiple imputation*. *Intell. Data Anal.* (Vol. 8). https://doi.org/10.3233/IDA-2004-8604

Oentaryo, R. J., Lim, E.-P., Lo, D., Zhu, F., & Prasetyo, P. K. (2012). Collective churn prediction in social network. *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*, (August), 210–214.

Olson, D., & Delen, D. (2008). *Advanced Data Mining Techniques*. *Springer. USA*.

https://doi.org/10.1007/978-3-540-76917-0

Óskarsdóttir, M., Bravo, C., Verbeke, W., Sarraute, C., Baesens, B., & Vanthienen, J. (2016). A comparative study of social network classifiers for predicting churn in the telecommunication industry. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)* (pp. 1151–1158). https://doi.org/10.1109/ASONAM.2016.7752384

Owczarczuk, M. (2010). Churn models for prepaid customers in the cellular telecommunication industry using large data marts. *Expert Systems with Applications*, *37*(6), 4710–4712. https://doi.org/https://doi.org/10.1016/j.eswa.2009.11.083

Pawlak, Z. (1982). Rough sets. *International Journal of Computer & Information Sciences*, *11*(5), 341–356. https://doi.org/10.1007/BF01001956

Pendharkar, P. C. (2009). Genetic algorithm based neural network approaches for predicting churn in cellular wireless network services. *Expert Systems with Applications*, *36*(3, Part 2), 6714–6720. https://doi.org/https://doi.org/10.1016/j.eswa.2008.08.050

Prati, R. C., Batista, G. E. A. P. A., & Monard, M. C. (2008). A Study with Class Imbalance and Random Sampling for a Decision Tree Learning System. In M. Bramer (Ed.) (pp. 131–140). Boston, MA: Springer US.

Quinlan, J. R. (1986). Induction of Decision Trees. *Mach. Learn.*, *1*(1), 81–106. https://doi.org/10.1023/A:1022643204877

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Raza, S., & Qamar, U. (2017). *Understanding and Using Rough Set Based Feature Selection: Concepts, Techniques and Applications*. https://doi.org/10.1007/978-981-10-4965-1

Reitermanov, Z. (2010). Data Splitting, 31–36.

Rokach, L., & Maimom, O. (2014). *Data mining with decision trees: theory and applications*. *Data Mining and Knowledge Discovery*. https://doi.org/10.1007/978-0-387-09823-4

Rubin, D. B. (1987). *Multiple Imputation for Nonresponse in Surveys*. John Wiley & Sons, Inc., Hoboken, NJ, USA. https://doi.org/10.1002/9780470316696

Rubin, D. B. (1988). *AN OVERVIEW OF MULTIPLE IMPUTATION*.

Rubin, D. B. (1996). Multiple Imputation after 18+ Years. *Journal of the American Statistical Association*, *91*(434), 473–489. https://doi.org/10.1080/01621459.1996.10476908

Schafer, J. L. (1999). Multiple imputation: a primer. *Statistical Methods in Medical Research*, *8*(1), 3–15. https://doi.org/10.1177/096228029900800102

Schapire, R. E. and Freund, Y. (2012). *Boosting: Foundations and Algorithms.* MIT Press.

Şen, K., & Bayazıt, N. G. (2015). Customer churn modelling in banking. In *2015 23nd Signal Processing and Communications Applications Conference (SIU)* (pp. 2384–2387). https://doi.org/10.1109/SIU.2015.7130361

Shah, A. D., Bartlett, J. W., Carpenter, J., Nicholas, O., & Hemingway, H. (2014). Comparison of parametric and Random Forest MICE in imputation of missing data in survival analysis, 1–17.

Sikora, R., & Al-Laymoun, O. (2014). *A modified stacking ensemble machine learning algorithm using genetic algorithms*. https://doi.org/10.4018/978-1-4666-7272-7.ch004

Singh, I., & Singh, S. (2017). Framework for Targeting High Value Customers and Potential Churn Customers in Telecom using Big Data Analytics. *International Journal of Education and Management Engineering*, *7*(1), 36–45. https://doi.org/10.5815/ijeme.2017.01.04

Steyerberg, E. W., Harrell, F. E., Borsboom, G. J. J. M., Eijkemans, M. J. C. R., Vergouwe, Y., & Habbema, J. D. F. (2001). Internal validation of predictive models : Efficiency of some procedures for logistic regression analysis, *54*, 774–781.

Svetnik, V., Liaw, A., & Tong, C. (2004). *Variable Selection in Random Forest with Application to Quantitative Structure–Activity Relationship*. *Proceedings of the 7th Course on Ensemble Methods for Learning Machines*.

Team, R. D. C. (2009). *R: A Language and Environment for Statistical Computing*. Vienna, Austria OR - R Foundation for Statistical Computing. Retrieved from citeulike-article-id:4132730

Therneau, T. M., & Atkinson, E. J. (1997). *An introduction to recursive partitioning using the {RPART} routines*. Retrieved from citeulike-article-id:3462048

Tomek, I. (1976). Two Modifications of CNN. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-6*(11), 769–772. https://doi.org/10.1109/TSMC.1976.4309452

Up, S., & In, S. (2017). Handling imbalanced dataset in supervised learning using family of SMOTE algorithm . Over Sampling Algorithms based on SMOTE.

van Buuren, S., & Oudshoorn, K. (2011). mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, *45*(3), 1–67. Retrieved from citeulike-article-id:10495340

van der Maaten, L., & Hinton, G. (2008). Visualizing Data using t-SNE. *Journal of Machine Learning Research*, *9*, 2579–2605. Retrieved from citeulike-article-id:3749741

Vuk, M. (2006). ROC Curve , Lift Chart and Calibration Plot, *3*(1), 89–108.

Wei-Yin, L. (2014). Fifty Years of Classification and Regression Trees. *International Statistical Review*, *82*(3), 329–348. https://doi.org/10.1111/insr.12016

Wirth, R., & Hipp, J. (2000). *CRISP-DM: Towards a standard process model for data mining*. *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining*.

Wolpert, D. (1992). *Stacked Generalization*. *Neural Networks* (Vol. 5). https://doi.org/10.1016/S0893-6080(05)80023-1

Wu, J., Brubaker, S. C., Mullin, M. D., & Rehg, J. M. (2008). Fast Asymmetric Learning for Cascade Face Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *30*(3), 369–382. https://doi.org/10.1109/TPAMI.2007.1181

Wu, Y., Qi, J., & Wang, C. (2009). The study on feature selection in customer churn prediction modeling. In *2009 IEEE International Conference on Systems, Man and Cybernetics* (pp. 3205–3210). https://doi.org/10.1109/ICSMC.2009.5346171

Xie, J., & Coggeshall, S. (2009). A Combination of Boosting and Bagging for KDD Cup 2009 - Fast Scoring on a Large Database, 35–43.

Yabas, U. (2013). Churn Prediction in Subscriber Management for Mobile and Wireless Communications Services, 991–995.

YANG, Q., & WU, X. (2006). 10 CHALLENGING PROBLEMS IN DATA MINING RESEARCH. *International Journal of Information Technology & Decision Making*, *05*(04), 597–604. https://doi.org/10.1142/S0219622006002258

Zhang, R., Li, W., Tan, W., & Mo, T. (2017). Deep and Shallow Model for Insurance Churn Prediction Service. In *2017 IEEE International Conference on Services Computing (SCC)* (pp. 346–353). https://doi.org/10.1109/SCC.2017.51

Zhang, W., Mo, T., Li, W., Huang, H., & Tian, X. (2016). The Comparison of Decision Tree Based Insurance Churn Prediction between Spark ML and SPSS. In *2016 9th International Conference on Service Science (ICSS)* (pp. 134–139). https://doi.org/10.1109/ICSS.2016.25

7. APPENDIX

One-Sided Selection Algorithm:
1- Let S be the dataset, Assign all minority class to cluster C, and one randomly majority class data point.
2- Using $K-NN$ with $K=1$ to classify S using examples from cluster C, then evaluate the performance of $K-NN$ and move the misclassified data points to C.
3- Remove all Tomek Links majority data points from C, Then assign the rest data points to cluster T.

One-Sided Algorithm

K Nearest Neighbor Algorithm:
Let X the new data point to be classified:

1- From the training dataset identify the KNN of X
2- The Class that most frequently associated with X is denoted c_i
3- For two class domains, an odd K should be selected to prevent ties
4- Give label of X like c_i

Euclidean distance similarity measure:

$$d(x,y)=\sqrt{(x_i-y_i)}$$

With n continuous features:
$x=(x_1,x_2,\dots,x_n), y=(y_1,y_2,\dots.,y_n)$

$$d_E(x,y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)}$$

For mixed features:

$$d_M(x,y)=\sqrt{\sum_{i=1}^{n}d(x_i-y_i)}$$

For discrete:

$$d(x_i,y_i)=0\,,x_i\neq y_i$$
$$d(x_i,y_i)=1\,,x_i=y_i$$

KNN Algorithm

Line	Code
1	***Start***
2	**for** i <- 1 **to** 10(k-nearest neighbors for 10)
3	Compute k-nearest neighbors, and save the indices in the number of attribute.
4	**end for**
5	**while**
6	Choose a random number between 1 and k, call it nn. Choose one of the k-nearest neighbors of 10.
7	**for** j <- 1 **to** number of attribute.
8	dif = MinorityClassSample(attribute(nn)(i)) - MinorityClass Sample[i][j]
9	gap = rand() // between 0 and 1
10	NewClassSample[newindex][j] = MinorityClassSample[i][j] + gap *dif
11	**end for**
12	newindex ++
13	**end while**
14	**End**

SMOTE Algorithm (J. Kim, Han, & Lee, 2016)