Potential Customer Detection for Upselling: A case study of Orange

Na Zhang

Introduction

Whether it is traditional sales or online sales on the Internet, using certain sales techniques can satisfy the needs of customers to the greatest extent and improve sales performance. In addition to acquiring more users, paying attention to the purchase behavior of individual customers can also improve sales performance. Upselling is one of the sales techniques that increase the average order value of individual customers. Amazon announced in 2006 that it achieved a 35% increase in revenue through upselling and cross-selling (Walker, 2017). Upselling refers to the sale of upgrades, add-ons, or other products or services enhanced the originals to a customer for a specific product or service (Upselling, 2018). The specific products or services must be extensible. The additional sales objects are related to the original products or services or even the same, and they have the effect of supplementing or strengthening or upgrading. However, not all customers will respond to the upselling offers. To identify potential customers and implement targeted marketing strategies will have a significant impact on the company's sales performance. Customer Relationship Management is an important enterprise strategy for companies to maximize the value of customers through information technology. It provides processes and information for companies to understand the needs of customers and enhance the relationship with customers (Kincaid, 2003). In CRM, Data mining is the technique that extract the hidden information, identify patterns and predict trends from customer data (alias Devi, 2015). Therefore, with the analysis and predicting capabilities of data mining techniques, companies have opportunities to target potential customers to deliver the upselling service.

This experiment is a case study of the telecommunication giant Orange for detecting responsive customers for upselling. Orange is the fourth largest telecom operator in the world. In the "turn the world into data" campaign, it is committed to data-driven business transformation and not only becomes a beneficiary in this process but also becomes a communicator (Orange, 2015). Orange built an internal platform to implement marketing application of data science. The platform automates the data mining process from data preparation to model deployment (Guyon, Lemaire, Boullé, Dror, & Vogel, 2009). Therefore, the customer data for this case study is prepared by the corresponding module. Detecting potential customers is a classification problem that classify the customers to be detected as potential customers and ordinary customers. The

report records the entire mining processes including understanding the data, handling the data with missing value, resampling minority data, selecting features, training models and evaluating trained models. Naïve bayes, decision tree and random forest are trained and evaluated for this task, and in this report, area under the ROC (receiver operating characteristic) Curve(AUC) performance as the measure is used to determine whether the mining results achieves the goal. The baseline for this task is that the AUC performance of evaluation of the trained model must be at least 0.84.

The rest of this report introduces the data mining techniques and reasons for the selected algorithms. Subsequent sections gradually describe the mining process. The datasets are introduced in the data understanding section. Then based on the knowledge of the datasets, the data will be preprocessed for better training results, and this is showed in the data preprocessing section. The modeling section details the training and parameter tuning processes of each model. The analysis and comparison of the results are discussed in the evaluation section. The judgment of this mining experiment is presented as the conclusion of this report.

Data mining techniques

Data mining refers to the process of excavating unknown and valuable information and knowledge from a large amount of data (Hand, 2007). It is a learning process, and this learning process can be divided into two categories: supervised learning and unsupervised learning. Supervised learning generates a model that maps the input to a suitable output by using the existing correspondence between input data and output data, typical supervised learning problem such as classification. It requires input data to be pre-labeled by the categories. Unsupervised learning models input data directly, such as clustering. Due to to the prevalence of data imbalance and noise problems in the classification task of customer behaviour prediction, some studies show that existing algorithms do not produce satisfactory results (Xie, Li, Ngai, & Ying, 2009). The running process of neural network algorithm as black box is not interpretative. Meanwhile, genetic algorithms have limitations of uncertainty of prediction possibilities, and other methods such as SVM and Bayesian multi-net cannot get favorable results (Lariviere & Van den Poel, 2004). Unlike neural networks, the advantages of decision tress are that it is easy to understand, the decision-making process is transparent, and the non-linear relationship between the features becomes more intuitive and predictable. However, its disadvantage is instability, which is greatly affected by data noise. Bagging technique can be applied to alleviate this problem for decision tree (Dietterich, 2000). Random forests as embedded learning algorithm is a combination of decision tree and bagging, and it has already applied in a wide range of applications (Verikas, Gelzinis, & Bacauskiene, 2011). In the application of customer churn prediction, several studies prove that random forests can get better performance in terms of

AUC than decision tree, support vector machine, multilayer perceptron and logistic regression (Xie, Li, Ngai, & Ying, 2009; Ying, Li, Xie, & Johnson, 2008; Coussement, & Van den Poel, 2008; Coussement, & Van den Poel, 2009). As another embedded learning, boosting is rarely used in this type of application. A study on KDD Cup 2009 orange competition uses boosting to improve performance of one of the base learners in the compound tri-classifier approach (Lo et al., 2009). For this task, selected classifiers based will be experimented, and they are boosting classifier embedded naïve bayes and decision tree respectively and random forests.

Naïve bayes classifier is a simple and effective common classification algorithm. As a probabilistic learner, it calculates the probability of occurrence of each category in class under the condition of the existence of the item to be classified, and this is based on the assumption of independence between independent variables (conditional feature independence) and the normality of continuous variables. The Naïve Bayesian model assumes that the attributes are mutually independent in the given output categories, and when the assumption is established, its learning speed is fast and the performance is good and this is why it is considered as a base learner for this task. However, this hypothesis is often not practical in real applications. When the number of features is large or the correlation between features is high, the effect of classification is not good (Dimitoglou, Adams, & Jim, 2012).

Another base learner for this task is J48 that is an implementation of C4.5 decision tree (Dimitoglou, Adams, & Jim, 2012). C4.5 is one of the most popular algorithms in the decision tree algorithm family, another one is its predecessor ID3. This tree-based classification algorithm uses a tree structure to establish a decision model based on the features of the data (Patil, & Sherekar, 2013). The key content of constructing a decision tree is to perform attribute selection metrics to select the best split attributes as decision nodes, and information gain, gain ratio and gini index are three commonly used attribute selection measures. ID3 algorithm uses information gain as the measure to select features. It takes the rate of decline of the information entropy as the criterion for selecting features, that is to select the feature with the highest information gain that has not been used yet. This algorithm adopts a top-down recursive method to compare feature values in the nodes of the decision tree and to branch downward from the node according to different values of the feature (Peddabachigari, Abraham, Grosan, & Thomas, 2007). The main drawback of ID3 is that the use of information gain as a criterion for selecting branch feature tends to favor features with a large number of values, and in some cases these features may not provide too much valuable information (Dimitoglou, Adams, & Jim, 2012). C4.5 is an improvement of the ID3 algorithm, and it uses the information gain ratio as a criterion for selecting branch features, which remedies the lack of ID3. C4.5 can handle discrete and continuous feature types

that discretize continuous features. After the decision tree is constructed, it performs pruning operations. However, the computational efficiency of C4.5 is low, especially for training samples with continuous feature values. Furthermore, it does not consider the correlation between the conditional features when selecting the branch feature, and only calculates the expected information between each conditional feature and the decision feature (class feature) in the dataset, which may affect the correctness of the feature selection.

Two embedded learning methods used in this task are boosting and bagging. They all combine existing classification or regression algorithms in a certain way to form a more powerful classifier. The main difference between the two is that the sampling method is different. Bagging adopts uniform sampling, and boosting is sampled according to the error rate, thus the classification accuracy of boosting is higher than that of bagging. The working mechanism of boosting is to repeatedly train weak classifiers on the training data that is composed of data that has been misclassified by weak classifiers previously trained and data that has not been used, and then combine the resulting weak classifiers into a composite classifier by voting mechanism (Freund, & Schapire, 1996). An improvement of the boosting algorithm named AdaBoost is used in this task, which is proposed by Freund and Schapire (1996). Adaboost is Adaptive Boosting, it can adaptively adjust the error rate of the weak learning algorithm. After several iterations, the error rate can achieve the desired effect. Adaboost does not need to accurately know the sample distribution. It adjusts the sample distribution after weak algorithm is trained and updates the weights of all training samples. The weights of the correctly classified samples in the sample space are reduced and the weights of samples that are misclassified will increase. The next time learning through the weak algorithms will pay more attention to these misclassified samples. Each trained weak classifier has a corresponding weight and has a greater weight for classifiers with a smaller classification error and each classifier can only be generated sequentially because the parameters of latter model need the result of the previous one. The version of AdaBoost.M1 is used for the experiment of this task, and the other version is AdaBoost.M2, which is different from AdaBoost.M1 that uses error rate as an error measure, it uses a more complex error measure, pseudo-loss (Freund, & Schapire, 1996). Bagging is an abbreviation for bootstrap aggregating (Breiman, 1996). It is a kind of sampling method with replacement, and it is an important statistical method in non-parametric statistics that estimates the statistical variance and then performs interval estimation. Bagging algorithm performs multiple trainings and obtains multiple models. The dataset for each training consists of specified number of training samples randomly drawn from the initial training dataset. Thus a certain training sample may exist multiple times in the dataset or does not appear. For the classification problem, multiple models classify the new sample, the one that gets the most is the final category of this sample. Because

the model in the bagging algorithm can be generated in parallel, its running time is shorter than the boosting algorithm.

Random forest is a combination of bagging method and decision tree CART (Breiman, 2017) algorithm (Breiman, 2001). It is characterized by using bagging for both samples and features, which means that randomly samples instances and features (Liaw, & Wiener, 2002). Random forest draws the bootstrap samples from the dataset, and this process is the same as the original bagging algorithm. Each node of each tree in the search for feature splitting does not find the feature that maximizes the measurement for all features. Instead, it randomly selects features in the feature set and find the optimal solution among the extracted features and applied the best feature to nodes to split. The experiment from Breiman(2001) shows the advantages of using random features, such as the better accuracy than Adaboost, more resistant to noise and faster running speed. The research of its application in real world also presents that it can handle high-dimensional dataset directly, it can exhibit the important features for classifying, it has an advantage over the training speed of large datasets (Rodriguez-Galiano et al., 2012).

Data Understanding

There are two datasets used for completing the classification task that are provided by the data preparation module of Orange customer analysis platform. The training dataset has 1667 instances and 230 features included class feature. There is no visible, identifiable customer information in the data. All the features are named after a combination of characters and numbers such as Var 230 that are not meaningful. The data are heterogeneous, 38 features excluding class feature are nominal that are category features, the rest are numeric. The labels for these category features are random combination of characters and numbers, so they do not provide any information for feature selection. Most features have missing values, and the proportion of missing values of 153 features is higher than 90% and 17 features have no value at all. Meanwhile, for some of the features with missing values less than 90% or without missing values, the distribution of values of these features is sparse. Visualization of the values of these features shows the presence of outliers. In this task, customers are divided into two categories, potential customers and general customers. Thus, for the class feature, there are two labels to represent each type of customer. The datasets have already been labeled for each instance. The potential customers are represented as 1(positive) while general customers are represented as -1(negative). The distribution of the class features shows that the number of the two categories is quite unbalanced. In the training dataset, the sample size of general customers is 1532, which accounts for 92% of the total sample size. For the testing dataset, it is composed of 3333 instances, and 271 of these represent potential customers, which accounts for only 8% of the total sample size. The samples of testing dataset are also pre-labeled. According to the understanding

of the dataset, there are many difficulties for the training dataset to overcome to improve the training effect before training the models. These difficulties can be summarized as high dimensionality, heterogeneous data, missing values, sparse values and imbalance in the number of instances of target classification and non-target classification.

Data Preprocessing

The data has been prepared in advance for training dataset and testing dataset. The training dataset is used to train the models, and the testing dataset is used to evaluate the trained models. Based on the previous understanding of the datasets, difficulties in the datasets require preprocessing to get better training results.

- Feature clean

The first difficulty that needs to be addressed is the processing of the features with high proportion of missing values to reduce the dimensionality of the dataset. In the training dataset, the proportion of missing values for 17 features is 100%. These features are removed directly from the training dataset and then other features with the percentage of missing values above 90% are dealt with. In order to determine whether these features are related to classification, the dataset containing these features and the dataset without these features are input respectively into the models to be trained. By comparing the accuracy and AUC performance of training results to guide the judgments about these features. To do this process, the dataset with full features is trained firstly with Naïve bayes and Decision tree classifiers respectively and evaluated using the training dataset itself. Then the same process is also implemented on the dataset without features that have no values. Similar proportions of missing values are clustered into different groups, so three groups of proportion are evaluated and they are 90+% missing values group, 70+% missing values group and 50+% missing values group. The evaluation results for datasets without different proportion of missing values are shown in Table 1. Based on the results, when the features with 50% missing values are removed, the accuracy and AUC score of each model both have a significant reduction, thus the features in this group are retained. Finally, the features with the proportion of missing values above 70% are eliminated from the dataset. After this process, there are 75 features left in the dataset.

	Naïve Bayes		Decision Tress	
	Accuracy	AUC	Accuracy	AUC
full features	93.4013 %	0.934	93.5213 %	0.801
without 100% missing value features	93.4013 %	0.934	93.5213 %	0.801
without 90+% missing value features	94.781 %	0.933	93.5213 %	0.801
without 70+% missing value features	94.841%	0.933	93.5213 %	0.801
without 50+% missing value features	93.7013%	0.917	91.9016 %	0.5

Table 1. Evaluation of the features without different percentage of missing values

- Feature selection

For performance considerations and to further reduce the dimensionality of the dataset, feature selection techniques are used to reduce more irrelevant and redundant features than manual selection. Although the number of features in the dataset has been reduced from 230 to 75 by manual selection, based on this number of features and the purpose of this process, subset evaluation algorithms are more suitable for feature selection than feature weighting evaluation algorithms, because the latter does not reduce the redundancy of features (Kira & Rendell, 1992). Correlation feature selection(CFS) is a filter-based method, with search strategy, it can select the subset with features uncorrelated to each other and highly correlated to the class (Hall, 2000). Wrapper method evaluate the features based on the specific classifier and the requirement of the measurement to get the most suitable feature subset for the best measurement performance, but compared to CFS, using this method will have a cost in computational efficiency (Langley, 1994). These two types of subset-based algorithms are all applied to the training dataset to get the best feature subset for this task. The search methods used with evaluators are Best First and Greedy Stepwise. Best First performs a greedy hill-climbing method with backtracking and it does not have restrictions on starting position and search direction (Pearl, 1984). Greedy Stepwise is a one-way search method and does not backtrack, but there is no limit to the search starting point and it stops the search when the change in the number of features of the subset results in a decrease in the evaluation (Russell, & Norvig, 2016). In this experiment, Weka as the tool is used to perform the feature selection process. And four combinations of evaluation algorithms (evaluator in Weka) and search algorithms are implemented for comparing which feature subset can provide optimal performance for mining. The features that are selected by each combination are shown in Table 2. The result of feature selection for each combination is saved as a separate dataset for subsequent data preprocessing and mining.

Evaluator	Search algorithm	Number of features	Names of features
Wrapper+Naïve Bayes EvaluationMeasure=AUC	Bestfirst	7	Var28(Num),Var126(Num),Var143(Num),Var173(Num),Var211(Nom),Var218(Nom), Var230(class)
Wrapper+Decision Tree EvaluationMeature=AUC	Bestfirst	8	Var13(Num),Var28(Num),Var85(Num),Var126(Num),Var144(Num),Var193(Nom),Var 218(Nom),Var230(class)
Wrapper+Decision Tree EvaluationMeature=accuracy	Bestfirst	5	Var7(Num),Var28(Num),Var126(Num),Var193(Nom),Var230(class)
CfsSubsetEval	Greedystep wise	3	Var126(Num),Var202(Nom),Var230(class)

Table 2. Feature Selection results

- Feature Completion

Although the dimensionality of the training data has been minimized by feature selection, the missing values of the selected features, the inhomogeneity of feature types and the imbalance of the number of samples of the positive class need further processing. Missing values can affect the training performance, and when the missing ratio is more than 15%, the impact is severe (Acuna, & Rodriguez, 2004). However, these effects may be positive, which means that these missing values are relevant classification indicators. In this case, missing values of such features will not be processed. There are three methods to treating missing data (Little, & Rubin, 2014). Case Deletion has been used to delete features with high levels of missing values. Imputation method such as the mean imputation will be used to treat the missing values of the rest features. For numeric features, the missing values are replaced with the means of the existing values of corresponding features. The modes will be used to replace the missing values of the nominal features. The last method is weighting estimation.

- Feature Consistency

The effect of different types of features on training performance needs to be judged by the results of training models. For category features, especially those with extensive categories, the impact of these features is determined by the results of training using the dataset without these features. Due to the presence of outliers in the values of the features and in order to avoid the neglect of the predictability of these outliers, numerical features are discretized only according to the needs of the trained model.

- Data Rebalancing

In the binary classification, most learning algorithms are designed on the basis of balanced datasets. When learning unbalanced datasets with high complexity, these algorithms cannot learn accurate predictability from the minority samples. Therefore, using actual data to evaluate these trained algorithm models will result in unsatisfactory accuracy (He, & Garcia, 2009). Using performance metric

other than accuracy such as AUC is a workaround, and another way to deal with this problem is to resample the samples of categories. He and Garcia (2009) claim that adjusting the number of two categories in the imbalanced dataset by resampling does improve the performance of the classifiers. There is study that have proved this statement (Estabrooks, & Japkowicz, 2004). However, there is also a study shows that the learning performance of the classifier on unbalanced datasets is commensurate to the learning performance on resampled dataset (Batista, Prati, & Monard, 2004). There are different sampling methods, such as random oversampling and under-sampling, informed under-sampling and synthetic sampling. In this task, random sampling and synthetic sampling methods are used to balance the data. Random sampling provides two mechanisms to alter the proportion of samples of two categories as needed. Oversampling adds a subset to the original dataset and the subset is made up of the copies of the randomly selected samples of minority class while under-sampling does the opposite. Under-sampling removes samples of majority class. In Weke, Resample preforms a random sampling, it's hybrid of oversampling and under-sampling method. There are some requirements to apply this method, such as the necessity of nominal features. The synthetic minority oversampling technique (SMOTE) is another resampling method that is also applied on the dataset in this task by Weka. It is an oversampling approach to add synthetic samples of minority class and their K nearest neighbors to the training dataset. The synthetic samples are obtained by merging the differences between two samples in the samples (Chawla, Bowyer, Hall, & Kegelmeyer, 2002)

Performance Measure

Confusion matrix is a specific matrix used to visualize the performance of the machine learning algorithms. Through the calculation of the elements in the confusion matrix, the predictive accuracy of the corresponding algorithm can be obtained and it is often used to measure the performance of algorithms mining with balanced dataset (Chawla, Bowyer, Hall, & Kegelmeyer, 2002). For unbalanced dataset, accuracy is not a fitting performance metric. Area Under the ROC Curve is an evaluation indicator that evaluates the performance of a binary model. ROC Curve refers to Receiver operating characteristic curve, its main analytical tool is a curve on a two-dimensional plane that is a unit square. AUC is the area under the ROC curve so its value is between 0 and 1.0. Different from the accuracy, AUC is not affected by the sample imbalance but represents the probability that the classifier tends to learn through positive samples. This important property can make it more objectively evaluate the performance of a classifier (Fawcett, 2006). Since in this task, the number of samples of the target class in the dataset only accounts for 8% of the total sample size. In order to more accurately evaluate the performance of classifiers, AUC is used as a performance evaluation criterion. The goal of the experiment is that the trained classifier can achieve predictive performance higher than 0.84 when use the testing dataset to evaluate.

Experiment and Modeling

This section describes the implementation process of the mining. The process includes not only algorithm selection and parameter tuning, but also the execution of methods used in the data preprocessing. The entire data mining process is run in the Weka environment. In this experiment, the test dataset is used as the evaluation dataset to evaluate the model while training the model.

Based on the results of feature selection, there are four trimmed datasets with different dimensions. Each dataset is input into the candidate learners. By comparing the learning results, an optimal dataset is selected as the final training dataset for subsequent processing and modeling. Table 3 shows the results of each learner trained on the datasets. A total of six classifiers are tested, including two standard learners and four meta-learners. The two standard learners are Naïve Bayes and J48 and the four meta-learners are AdaBoostM1 embedded with Naïve Bayes, AdaBoostM1 embedded with J48, Bagging embedded with Naïve Bayes and random forest. The results show that the training dataset, which is generated from wrapper evaluator based on Naïve bayes classifier with best-first search method, can provide the best learning effect for each learner compared to other three datasets. Therefore, this dataset is selected as the final training dataset. The next step for the experiment is to handle the missing values.

Feature subset	Naïve Bayes		Decision Tree C4.5(J48)		AdaBoostM1+Naïve Bayes		AdaBoostM1+J48		Bagging+Naïve Bayes		Random Forest	
	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy
Wrapper+Naïve Bayes EvaluationMeasure=AUC	0.815	91.8692%	0.793	93.9994%	0.79	91.8692%	0.84	93.9094%	0.809	91.7192 %	0.82	93.6694 %
Wrapper+Decision Tree EvaluationMeature=AUC	0.719	90.7591 %	0.793	93.9994 %	0.715	91.5692 %	0.752	92.0192 %	0.717	90.6091 %	0.779	92.0492 %
Wrapper+Decision Tree EvaluationMeature=accuracy	0.716	91.4191 %	0.797	94.3894 %	0.723	91.4191%	0.808	92.9493 %	0.718	91.2991 %	0.772	92.4692 %
CfsSubsetEval	0.685	91.8692%	0.5	91.8692 %	0.687	89.2889 %	0.72	89.3489 %	0.69	88.7789 %	0.532	89.829%

Table 3. Feature selection evaluation

In the experimental implementation phase, missing values in the final dataset are complemented by the ReplaceMissingValues filter of Weka. Table 4 shows the comparison of performance of six classifiers trained on the dataset with or without missing values. It can be known from experiments that the treatment of missing values has a significant positive effect on the AUC performance of bagging embedded with Naïve Bayes classifier. For the accuracy, all tree-based classifiers and bagging embedded with Naïve Bayes have improved accuracy. The next step assesses the impact of feature types in the final training dataset on training results.

Learners	With Missing Values		ReplaceMissingValues	
	AUC	Accuracy	AUC	Accuracy
Naïve Bayes	0.815	91.8692%	0.812	91.8692 %
Decision Tree C4.5(J48)	0.793	93.9994%	0.796	94.2694 %
AdaBoostM1+Naïve Bayes	0.79	91.8692%	0.784	91.8092 %
AdaBoostM1+J48	0.84	93.9094%	0.822	94.1494
Bagging+Naïve Bayes	0.809	91.7192 %	0.82	91.8692 %
Random Forest	0.82	93.6694 %	0.808	93.7294 %

Table 4. Comparison of results between datasets

The features in the final training dataset have two different types, numeric and nominal. The nominal features are removed from the dataset, and this dataset is input to the classifiers for evaluating the impact of these features. The dataset with or without nominal features is illustrated in the Table 5. Although these category features keep the accuracy of each learner flat or slightly increase, they have a negative effect on the AUC score of the learner except J48. Based on this results, the nominal features are retained in the dataset. Numerical features are not further processed. Rebalancing the samples distribution in the final training dataset is the next step for the experiment.

Learners	With nominal features		Without nominal feature	
	AUC	Accuracy	AUC	Accuracy
Naïve Bayes	0.812	91.8692 %	0.737	91.8692 %
Decision Tree C4.5(J48)	0.796	94.2694 %	0.796	94.3894 %
AdaBoostM1+Naïve Bayes	0.784	91.8092 %	0.716	91.8692 %
AdaBoostM1+J48	0.822	94.1494	0.812	94.1494 %
Bagging+Naïve Bayes	0.82	91.8692 %	0.741	91.8692 %
Random Forest	0.808	93.7294 %	0.798	93.9394 %

Table 5. Assessment for nominal features

Resample and SMOTE methods all have been practiced with various values of the specific parameter. All tested values of parameter for each method are listed in Table 6. The AUC score and accuracy of classifiers trained on each resampled dataset are also listed in Table 6. The best AUC of each classifier are highlighted. For classifiers other than AdaBoostM1 embedded with Naïve Bayes, the datasets generated by adjusting the sample distribution by random sampling obtain better training results. The optimal bias for the parameter of Resample method is 0.2 for J48 and Naïve Bayes bagging, and the dataset that is resampled with the parameter of 0.3 can train the best performance models for Naïve Bayes and J48 boosting classifiers. When the ratio of negative and positive samples in the

dataset is 5 to 8, random forests can train the highest AUC model on this dataset. For Naïve Bayes boosting classifier, the dataset increased the number of samples of minority class by 4 times is optimal. The best AUC performance is 0.851 from J48 classifier.

Resampleing techniques		Naïve Bayes		Decision Tree C4.5(J48)		AdaBoostM1+Naïve Bayes		AdaBoostM1+J48		Bagging+Naïve Bayes		Random Forest	
		AUC	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy
Without resampling		0.812	91.8692 %	0.796	94.2694 %	0.784	91.8092 %	0.822	94.1494	0.82	91.8692 %	0.808	93.7294 %
SMOTE Parameter=percentage	10%	0.811	91.8692 %	0.796	94.2694 %	0.794	91.8692 %	0.815	93.7894 %	0.817	91.8692 %	0.804	93.9394 %
	20%	0.812	91.8692 %	0.796	94.2694 %	0.783	91.6592 %	0.836	93.7894 %	0.817	91.8692 %	0.808	93.9694 %
	30%	0.81	91.8692 %	0.796	94.2394 %	0.774	85.6886 %	0.829	93.3393 %	0.816	91.8692 %	0.806	93.9094 %
	40%	0.813	93.0693 %	0.794	94.2694 %	0.805	93.0693 %	0.811	93.6094 %	0.815	91.8692 %	0.803	93.8194 %
	50%	0.813	93.2493 %	0.758	93.9094 %	0.805	93.2493 %	0.825	89.979 %	0.818	91.8692 %	0.801	93.4593 %
	60%	0.813	93.2193 %	0.793	94.2694 %	0.781	93.2193 %	0.817	92.9193 %	0.819	93.2193 %	0.801	93.8194 %
	70%	0.81	93.1593 %	0.782	93.9994 %	0.791	93.1593 %	0.821	93.2793 %	0.816	93.1893 %	0.797	93.7294 %
	80%	0.813	93.1593 %	0.795	94.2094 %	0.772	93.1593 %	0.816	92.5893 %	0.82	93.1893 %	0.804	93.6394 %
	90%	0.811	92.8893 %	0.764	93.8794 %	0.784	92.8893 %	0.816	92.9193 %	0.818	92.9793 %	0.802	93.7294 %
	100%	0.813	93.3093 %	0.791	93.9394 %	0.769	93.3093 %	0.83	93.2193 %	0.816	93.1593 %	0.808	93.7594 %
	200%	0.813	59.586 %	0.843	93.3693 %	0.758	68.4668 %	0.831	93.4293 %	0.811	63.0663 %	0.805	93.2193 %
	300%	0.81	45.0945 %	0.808	92.8593 %	0.68	45.0945 %	0.821	90.489 %	0.809	51.6952 %	0.811	92.4692 %
	400%	0.807	37.5638 %	0.838	91.5092 %	0.807	37.5638 %	0.813	88.8989 %	0.809	40.6241 %	0.803	91.1191 %
	500%	0.806	34.4734 %	0.834	88.5689 %	0.806	34.4734 %	0.809	89.0189 %	0.807	36.0636 %	0.796	89.2589 %
	600%	0.805	33.9934 %	0.841	87.8188 %	0.805	33.9934%	0.822	83.5584 %	0.803	36.0336 %	0.802	89.3789 %
	700%	0.804	32.7633 %	0.827	87.7588 %	0.624	32.7633%	0.822	82.0282 %	0.802	33.1233 %	0.797	86.7387 %
Resample(Weka) Parameter=biasToUniformClass	0.1	0.825	91.8692 %	0.757	94.2094 %	0.762	91.8692%	0.828	93.4293 %	0.825	91.8692 %	0.791	93.6694 %
	0.2	0.823	93.2193 %	0.851	93.9094%	0.8	93.1293 %	0.825	93.0393 %	0.826	92.9493 %	0.795	93.2793 %
	0.3	0.825	93.5494 %	0.833	93.5494 %	0.759	93.5494 %	0.85	93.4293 %	0.825	93.1593 %	0.8	93.0093 %
	0.4	0.823	92.7693 %	0.82	92.4992 %	0.79	92.7693 %	0.834	92.2592 %	0.825	92.7093 %	0.798	92.7393 %
	0.5	0.822	91.2691 %	0.818	91.9292 %	0.777	91.2691%	0.837	87.3387 %	0.822	91.3591 %	0.8	92.3192 %
	0.6	0.819	86.5587 %	0.822	91.3591 %	0.77	86.5587 %	0.833	89.529 %	0.821	86.3186 %	0.805	87.6988 %
	0.7	0.818	79.0879 %	0.814	90.099 %	0.783	79.0879 %	0.82	81.6982 %	0.819	78.4878 %	0.804	84.9985 %
	0.8	0.812	71.5572 %	0.797	85.2685 %	0.78	71.5572 %	0.829	83.6784 %	0.817	69.757%	0.807	84.3984 %
	0.9	0.824	68.2268 %	0.802	84.4884 %	0.789	67.9568 %	0.83	84.2784 %	0.82	66.2766 %	0.805	82.5983 %
	1.0	0.823	60.276 %	0.789	80.258 %	0.775	63.3663 %	0.826	84.4284 %	0.82	55.8656 %	0.809	80.8281%
	1.1	0.821	48.6949 %	0.806	77.9478 %	0.76	46.8947 %	0.839	74.4974 %	0.819	46.6847 %	0.808	76.6577 %
	1.2	0.821	43.5044 %	0.804	70.027 %	0.775	42.7543 %	0.832	72.9073 %	0.819	42.1842%	0.808	75.5776 %
	1.3	0.822	40.354 %	0.804	69.3969 %	0.788	40.144 %	0.819	76.1476 %	0.816	39.544%	0.813	75.1575 %
	1.4	0.822	38.0738 %	0.796	70.087 %	0.774	37.4137 %	0.833	73.8074 %	0.817	37.5338 %	0.81	74.2874 %
	1.5	0.815	36.2436 %	0.79	69.637 %	0.742	33.7534 %	0.823	72.7273 %	0.816	35.9736 %	0.81	73.1173 %

Table 6. Performance of rebalanced datasets

At this stage of the experiment, only two resampled datasets can be trained to meet the minimum requirements. In order to be able to train a better model, the parameters of each learner is optimized. Since feature type in the final training dataset is still heterogeneous, numeric features are converted to nominal features through supervised discretization when training Naïve bayes based classifiers. The minimum number of samples per leaf is tuned for the decision tree that is constructed by J48 classifier and the confidence factor also is configured. When using meta classifiers, the number of iterations has an effect on the enhancement of the underlying classifier, thus this parameter is tuned for both boosting and bagging. For bagging classifier, it is also important whether weights are used to represent the copies of samples. For random forest, in addition to the number of iterations, the depth of each tree is also an important factor affecting the performance of trained model. A pilot test for each classifier is conducted to reduce the number of candidate classifier for final modelling. The resampled datasets that can train the best model for each classifier are used for the pilot test. Tested parameter configuration and training results are listed in Table 7. Based on the results of pilot test for each classifier, two standard classifiers and boosting techniques will not be used for comprehensive modeling experiments. Compared to random forest, Naïve Bayes bagging classifier performs best performance in the pilot test. Next, these two classifiers will be

used for comprehensive modeling experiments. Each resampled dataset is input into the classifiers with various parameter configurations to train models.

Learners	Dataset	Parameters	Values	Performance	
				AUC	Accuracy
Naïve Bayes	Resample biasToUniformClass=0.3	useSupervisedDiscretization	FALSE	0.825	93.5494 %
			TRUE	0.866	93.4593 %
Decision Tree C4.5(J48)	Resample biasToUniformClass=0.2	E1-confidenceFactor,E2-minNumObj	E1=0.25 E2=2	0.851	93.9094%
			E1=1 E2=7	0.843	93.6694 %
AdaBoostM1+Naïve Bayes	SMOTE percentage=400%	A1-numIteration(AdaBoostM1), A2-useSupervisedDiscretization(Naïve Bayes)	A1=10 A2=False	0.807	37.5638 %
			A1=10 A2=True	0.829	85.8086 %
			A1=100 A2=True	0.823	84.7885
AdaBoostM1+J48	Resample biasToUniformClass=0.3	B1-numIteration(AdaBoostM1), B2-confidenceFactor(J48),B3-minNumObj(J48)	B1=10 B2=0.25 B2=2	0.85	93.4293 %
			B1=10 B2=1 B3=7	0.837	93.2493%
			B1=100 B2=1 B3=7	0.836	93.2193 %
Bagging+Naïve Bayes	Resample biasToUniformClass=0.2	C1-numIterations(Bagging),C2-representCopiesUsingWeights(Bagging), C3-useSupervisedDiscretization(Naïve Byes)	C1=10 C2=False C3=False	0.826	92.9493 %
			C1=10 C2=False C3=True	0.868	93.6994 %
			C1=100 C2=False C3=True	0.867	93.4293 %
Random Forest	Resample biasToUniformClass=1.3	D1-maxDepth,D2-numIterations	D1=0 D2=100	0.813	75.1575 %
			D1=2 D2=100	0.861	38.7639 %

Table 7. Results of pilot test

Evaluation

Table 8 lists the results obtained by training bagging classifier with Naïve bayes as base learner on datasets with various sample distributions. From the results it can be seen that almost all datasets can train models with performance over baseline by tuning parameters. When the base learner Naïve Bayes converts the numeric features to nominal features using supervised discretization, the AUC performance for each dataset has significant increase. When the number of iterations is increased to 300 and the copies of instances are represented by weights, bagging with Naïve bayes can train the best predictive model on the non-resampled dataset. The AUC of this model is 0.873. This result exceeds the benchmark to some extend.

Table 9 lists the results obtained by training random forest. When there is no limit to the depth of trees in random forest, each dataset cannot be trained to meet the baseline. However, the AUC can reach 0.87 with the depth of tree set to 2, and this is optimal AUC of Random forest.

Compared the results of the two classifiers, the model, which is trained from bagging with Naïve bayes on non-resampled dataset with the specific parameter configuration, is the final result of this experiment. The parameter configuration is that numIterations of bagging classifier is 300, represenCopiesUsingWeights of bagging is True and useSupervisedDiscretization of Naïve Bayes is True. The results also clearly show that AUC is not sensitive to imbalanced sample distribution of dataset, and this is very different from accuracy.

Bagging+Naïve Bayes		C1-numIterations(Bagging),C2-representCopiesUsingWeights(Bagging),C3-useSupervisedDiscretization(Naïve Byes)													
		C1=10 C2=False C3=False		C1=10 C2=False C3=True			C1=100 C2=True C3=True			C1=300 C2=True C3=True			C1=500 C2=True C3=True		
		AUC	Accuracy	AUC	Accuracy	Time	AUC	Accuracy	Time	AUC	Accuracy	Time	AUC	Accuracy	Time (s)
Without resampling		0.82	91.8692 %	0.869	94.3594 %	6.85	0.871	94.3594 %	11.88	0.873	94.3594 %	11.02	0.872	94.3594 %	11.33
SMOTE Parameter= percentage	10%	0.817	91.8692 %	0.869	94.3594 %	10.73	0.871	94.3594 %	8.95	0.872	94.3594 %	9.77	0.872	94.3594%	9.04
	20%	0.817	91.8692 %	0.871	94.1194 %	9.2	0.87	94.3594 %	9.1	0.869	94.3594 %	10.24	0.869	94.3594 %	9.58
	30%	0.816	91.8692 %	0.865	94.1494 %	6.18	0.87	94.1194 %	13.49	0.871	94.2994 %	7.43	0.873	94.2994 %	11.14
	40%	0.815	91.8692 %	0.867	94.1194 %	6.5	0.87	94.1194 %	6.37	0.87	94.1194 %	9.03	0.87	93.9094 %	13.36
	50%	0.818	91.8692 %	0.868	93.8794 %	7.94	0.87	94.0894 %	7.41	0.869	94.0894 %	9.02	0.869	94.0894 %	13.92
	60%	0.819	93.2193 %	0.868	93.8794 %	8.4	0.869	93.8494 %	8.79	0.869	93.9994 %	9.74	0.869	93.9994 %	9.17
	70%	0.816	93.1893 %	0.869	93.6394 %	9.76	0.867	93.8494 %	9.85	0.868	93.8794 %	9.89	0.868	93.8794 %	10.95
	80%	0.82	93.1893 %	0.868	93.8794 %	10.78	0.865	93.8794 %	10.8	0.867	93.8794 %	7.6	0.868	93.8794 %	12.44
	90%	0.818	92.9793 %	0.867	93.8794 %	5.81	0.867	93.8794%	11.89	0.867	93.8794 %	8.53	0.867	93.8794 %	13.32
	100%	0.816	93.1593 %	0.866	93.8494 %	6.88	0.868	94.0294 %	6.46	0.868	94.0294 %	9.01	0.868	94.0294 %	9.93
	200%	0.811	63.0663 %	0.864	93.9694 %	6.98	0.865	93.8794%	7.32	0.865	93.8794%	10.57	0.865	93.8794 %	11.26
	300%	0.809	51.6952 %	0.855	92.8593 %	8.75	0.857	93.7594 %	8.3	0.856	93.7894 %	12.38	0.857	93.7894 %	12.23
	400%	0.809	40.6241 %	0.854	84.3384 %	9.49	0.852	86.0186 %	9.91	0.852	86.3486 %	9.94	0.852	86.0786%	12.48
	500%	0.807	36.0636 %	0.853	82.8383 %	10.35	0.853	83.4083 %	11.82	0.853	83.1383 %	11.96	0.853	83.2283 %	10.93
	600%	0.803	36.0336 %	0.851	80.348 %	11.58	0.851	82.0282 %	9.3	0.852	82.1482%	11.34	0.852	82.3882 %	0.81
	700%	0.802	33.1233 %	0.846	80.408%	5.34	0.848	81.3681 %	6.68	0.849	80.378 %	12.21	0.849	80.498 %	11.7
Resample(Weka) Parameter= biasToUniformClass	0.1	0.825	91.8692 %	0.868	94.1494%	8.73	0.868	94.3294 %	11.8	0.868	94.3294 %	11.3	0.867	94.3294 %	10.36
	0.2	0.826	92.9493 %	0.868	93.6994 %	8.87	0.867	93.4293 %	6.95	0.865	93.4293 %	13.85	0.865	93.6994 %	11.76
	0.3	0.825	93.1593 %	0.867	93.4293 %	9.27	0.866	93.4593 %	7.46	0.864	93.6394 %	8.75	0.864	93.6394 %	11.64
	0.4	0.825	92.7093 %	0.865	93.4893 %	6.54	0.865	93.4593 %	8.34	0.864	93.4593 %	10.18	0.864	93.4593 %	10.55
	0.5	0.822	91.3591 %	0.855	92.7093 %	6.59	0.857	92.8293 %	9.36	0.857	92.5893 %	10.01	0.856	92.5593 %	11.37
	0.6	0.821	86.3186 %	0.854	84.7285 %	6.81	0.858	87.2187%	9.63	0.857	87.3387 %	10.9	0.857	87.3387 %	12.66
	0.7	0.819	78.4878 %	0.853	78.0978 %	7.04	0.857	79.718 %	12.16	0.856	79.658 %	12.57	0.856	79.928 %	12.49
	0.8	0.817	69.757%	0.851	72.3372 %	7.55	0.854	72.7873 %	6.53	0.853	72.7273 %	8.63	0.854	72.7873 %	9.81
	0.9	0.82	66.2766 %	0.849	70.7171 %	9.03	0.852	69.517 %	6.49	0.852	70.477 %	9.39	0.853	69.637 %	10.42
	1.0	0.82	55.8656 %	0.841	69.847 %	9.13	0.85	61.2661 %	7.95	0.849	61.2061 %	10.67	0.849	61.5062 %	11.13
	1.1	0.819	46.6847 %	0.842	60.8161 %	5.29	0.845	56.3156 %	8.17	0.846	56.4356 %	11.51	0.845	56.4356 %	12
	1.2	0.819	42.1842%	0.843	55.6556%	6.8	0.847	54.4554 %	8.34	0.845	54.3354 %	12.38	0.845	54.6055 %	10
	1.3	0.816	39.544%	0.84	54.4254 %	7.01	0.841	51.8452 %	9.53	0.841	51.9652 %	8.38	0.841	51.9652 %	10.7
	1.4	0.817	37.5338 %	0.835	51.5752 %	8.5	0.841	51.2151 %	6.42	0.842	50.9151 %	8.64	0.841	51.0051 %	12.14
	1.5	0.816	35.9736 %	0.832	51.1551 %	9.41	0.839	51.3351 %	6.59	0.841	51.2151 %	9.3	0.84	51.2751 %	12.08

Table 8. Performance of Bagging embedded with Naïve Bayes

Random Forest		D1-maxDepth,D2-numIterations										
		D1=100 D2=0		D1=100 D2=2			D1=100 D2=4			D1=200 D2=2		
		AUC	Accuracy	AUC	Accuracy	Time	AUC	Accuracy	Time	AUC	Accuracy	Time
Without resampling		0.808	93.7294 %	0.867	94.5095 %	5.08	0.863	94.4794 %	11.5	0.865	94.5395 %	8.58
SMOTE Parameter= percentage	10%	0.804	93.9394 %	0.869	94.5095 %	9	0.864	94.4194 %	8.48	0.869	94.5095 %	6.66
	20%	0.808	93.9694 %	0.868	94.5095 %	8.61	0.866	94.4194 %	8.94	0.867	94.5095 %	7.54
	30%	0.806	93.9094 %	0.865	94.1194 %	10.26	0.864	94.0894 %	10.69	0.865	94.1194 %	8.86
	40%	0.803	93.8194 %	0.864	94.3594 %	8.48	0.862	94.0894 %	6.62	0.867	94.4794 %	9.68
	50%	0.801	93.4593 %	0.865	94.0594 %	6.54	0.865	94.0894 %	7.21	0.868	94.3294 %	9.87
	60%	0.801	93.8194 %	0.866	94.0594 %	7.44	0.867	94.0294 %	6.92	0.867	94.0594 %	8
	70%	0.797	93.7294 %	0.865	94.0594%	11.53	0.863	94.0894 %	8.26	0.865	94.0594 %	7.58
	80%	0.804	93.6394 %	0.858	94.0294 %	8.85	0.866	94.0894 %	8.66	0.865	94.0594 %	8.1
	90%	0.802	93.7294 %	0.865	94.0594 %	9.39	0.868	94.0894 %	9.35	0.865	94.0594 %	8.58
	100%	0.808	93.7594 %	0.868	94.3594%	8.24	0.869	94.0894 %	6.83	0.866	94.0594 %	10.26
	200%	0.805	93.2193 %	0.866	94.0294%	6.2	0.867	93.9994 %	6.18	0.865	94.0294 %	11.24
	300%	0.811	92.4692 %	0.864	93.5494 %	6.79	0.858	93.8194 %	7.41	0.867	93.8794 %	11.04
	400%	0.803	91.1191 %	0.863	93.4593 %	7.09	0.863	93.8794 %	8.45	0.866	93.4293 %	6.76
	500%	0.796	89.2589 %	0.87	93.9994 %	7.89	0.865	93.9694 %	8.61	0.866	93.9994 %	7.43
	600%	0.802	89.3789 %	0.864	93.4293 %	8.96	0.863	93.8794 %	10.22	0.865	93.4293 %	9.92
	700%	0.797	86.7387 %	0.863	93.4593%	8.09	0.858	87.0087 %	7.18	0.865	93.3993 %	9.97
Resample(Weka) Parameter= biasToUniformClass	0.1	0.791	93.6694 %	0.867	94.4794 %	6.09	0.865	94.4494 %	11.22	0.868	94.4794 %	10.58
	0.2	0.795	93.2793 %	0.863	94.2994 %	8.4	0.863	94.3294 %	6.67	0.864	94.3294 %	9.11
	0.3	0.8	93.0093 %	0.868	94.0294 %	8.51	0.86	94.3294 %	7.33	0.868	94.0294 %	6.62
	0.4	0.798	92.7393 %	0.865	94.0294 %	8.65	0.857	94.1494 %	7.68	0.865	93.8194 %	6.88
	0.5	0.8	92.3192 %	0.866	93.4293 %	9.05	0.856	94.0294 %	8.64	0.866	93.4293 %	8.14
	0.6	0.805	87.6988 %	0.865	93.4293 %	10.06	0.863	93.9994 %	8.56	0.865	93.4293 %	8.86
	0.7	0.804	84.9985 %	0.866	93.4293 %	6.09	0.86	93.9994 %	7.34	0.866	93.4293 %	9.89
	0.8	0.807	84.3984 %	0.866	93.4293 %	7.2	0.862	93.9994 %	6.83	0.866	93.4293 %	10.42
	0.9	0.805	82.5983 %	0.867	79.628%	8.66	0.857	78.9379 %	7.56	0.866	79.2079 %	6.29
	1.0	0.809	80.8281%	0.861	74.9175 %	9.66	0.85	77.9778 %	9.31	0.866	74.9775 %	7.51
	1.1	0.808	76.6577 %	0.862	74.5575 %	10.2	0.858	76.3876%	12.16	0.866	74.5575 %	8.08
	1.2	0.808	75.5776 %	0.866	48.3948 %	10.87	0.859	54.0354 %	11.52	0.866	48.2148 %	8.52
	1.3	0.813	75.1575 %	0.861	38.7639 %	7.39	0.856	51.4851 %	9.88	0.861	38.7939 %	10.29
	1.4	0.81	74.2874 %	0.86	38.7639 %	6.09	0.854	47.8848 %	6.6	0.86	36.2436 %	11.19
	1.5	0.81	73.1173 %	0.857	31.1731 %	6.79	0.852	44.3144 %	8.28	0.858	29.3729 %	10.4

Table 9. Performance of Random forest

Conclusion

This work provided a solution for predicting potential responsive customers for conducting up-selling through data mining technology. The prediction of user behavior can help companies make business decisions. By targeting customers more accurately, companies can develop more targeted marketing strategies and minimize costs. This study takes French telecommunications company Orange as case study and build a predictive model of customer data from this company. The model building strategy includes data understanding, data preprocessing and modeling. The problems hidden in the data such as missing values and distribution imbalance are discovered through the understanding of the data. Through data preprocessing such as data cleaning, feature selection and resampling, these issues are handled one by one. Tree-based learners, probabilistic learners and even embedded learners are incorporated into the experiment to seek out an optimal model provider. Through experimental screening, Naïve bayes reinforced by bagging can train the best AUC performance model that is 0.873. Although Random forest performs well in many practices that solve similar research problems, it does not achieve the best result in this experiment. This experiment obtains relatively satisfactory experimental result and achieves the purpose of predicting customer behavior. For the future work, the processing of heterogeneous data needs to be improved, and experiments should be performed on the more parameter configurations of classifiers.

Reference

Acuna, E., & Rodriguez, C. (2004). The treatment of missing values and its effect on classifier accuracy. In Classification, clustering, and data mining applications (pp. 639-647). Springer, Berlin, Heidelberg.

alias Devi, P. I. (2015). Purpose of Data Mining for Analyzing Customer Data. International Journal, 3(4).

Batista, G. E., Prati, R. C., & Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD explorations newsletter*, *6*(1), 20-29.

Bhattacharyya, S., Jha, S., Tharakunnel, K., & Westland, J. C. (2011). Data mining for credit card fraud: A comparative study. *Decision Support Systems*, *50*(3), 602-613.

Breiman, L. (1996). Bagging predictors. Machine learning, 24(2), 123-140.

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5-32.

Breiman, L. (2017). *Classification and regression trees*. Routledge.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, *16*, 321-357.

Coussement, K., & Van den Poel, D. (2008). Churn prediction in subscription services: An application of support vector machines while comparing two parameter-selection techniques. Expert systems with applications, 34(1), 313-327.
4
Coussement, K., & Van den Poel, D. (2009). Improving customer attrition prediction by integrating emotions from client/company interaction emails and evaluating multiple classifiers. Expert Systems with Applications, 36(3), 6127-6134.

Dal Pozzolo, A., Caelen, O., Le Borgne, Y. A., Waterschoot, S., & Bontempi, G. (2014). Learned lessons in credit card fraud detection from a practitioner perspective. *Expert systems with applications*, *41*(10), 4915-4928.

Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine learning*, *40*(2), 139-157.

Dimitoglou, G., Adams, J. A., & Jim, C. M. (2012). Comparison of the C4. 5 and a Naïve Bayes classifier for the prediction of lung cancer survivability. *arXiv preprint arXiv:1206.1121*.

Estabrooks, A., Jo, T., & Japkowicz, N. (2004). A multiple resampling method for learning from imbalanced data sets. *Computational intelligence*, *20*(1), 18-36.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern recognition letters*, *27*(8), 861-874.

Freund, Y., & Schapire, R. E. (1996, July). Experiments with a new boosting algorithm. In *Icml* (Vol. 96, pp. 148-156).

Guyon, I., Lemaire, V., Boullé, M., Dror, G., & Vogel, D. (2009, June). Analysis of the kdd cup 2009: Fast scoring on a large orange customer database. In *Proceedings of the 2009 International Conference on KDD-Cup 2009-Volume 7* (pp. 1-22). JMLR. org.

Hall, M. (2000). Correlation-based feature selection for discrete and numeric class machine learning. Proceedings of the Seventeenth International Conference on Machine Learning (pp. 359–366).

Hand, D. J. (2007). Principles of data mining. *Drug safety*, *30*(7), 621-622.

He, H., & Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, *21*(9), 1263-1284.

Kincaid, J. W. (2003). Customer relationship management: getting it right!. Prentice Hall Professional.

Kira, K., & Rendell, L. (1992). The feature selection problem: Traditional methods and a new algorithm. Proceedings of the Tenth National Conference on Artificial Intelligence (pp. 129–134). Menlo Park: AAAI Press/The MIT Press.

Langley, P. (1994). Selection of relevant features in machine learning. Proceedings of the AAAI Fall Symposium on Relevance. AAAI Press.

Larivière, B., & Van den Poel, D. (2004). Investigating the role of product features in preventing customer churn, by using survival analysis and choice modeling: The case of financial services. *Expert Systems with Applications*, *27*(2), 277-285.

Liaw, A., & Wiener, M. (2002). Classification and regression by randomForest. *R news*, *2*(3), 18-22.

Little, R. J., & Rubin, D. B. (2014). *Statistical analysis with missing data* (Vol. 333). John Wiley & Sons.

Lo, H. Y., Chang, K. W., Chen, S. T., Chiang, T. H., Ferng, C. S., Hsieh, C. J., ... & Wang, C. H. (2009, June). An ensemble of three classifiers for kdd cup 2009: Expanded linear model, heterogeneous boosting, and selective naive bayes. In *Proceedings of the 2009 International Conference on KDD-Cup 2009-Volume 7* (pp. 57-64). JMLR. org.

Patil, T. R., & Sherekar, S. S. (2013). Performance analysis of Naive Bayes and J48 classification algorithm for data classification. *International Journal of Computer Science and Applications*, *6*(2), 256-261.

Pearl, J. (1984). Heuristics: intelligent search strategies for computer problem solving.

Peddabachigari, S., Abraham, A., Grosan, C., & Thomas, J. (2007). Modeling intrusion detection system using hybrid intelligent systems. *Journal of network and computer applications*, *30*(1), 114-132.

Rodriguez-Galiano, V. F., Ghimire, B., Rogan, J., Chica-Olmo, M., & Rigol-Sanchez, J. P. (2012). An assessment of the effectiveness of a random forest classifier for land-cover classification. *ISPRS Journal of Photogrammetry and Remote Sensing*, *67*, 93-104.

Russell, S. J., & Norvig, P. (2016). *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited.

Upselling. (2018). Retrieved from https://en.wikipedia.org/wiki/Upselling

Verikas, A., Gelzinis, A., & Bacauskiene, M. (2011). Mining data with random forests: A survey and results of new tests. *Pattern recognition*, *44*(2), 330-349.

Walker, T. (2017). How To Boost eCommerce Sales With Upselling. Retrieved from https://conversionxl.com/blog/upselling-techniques/

Xie, Y., Li, X., Ngai, E. W. T., & Ying, W. (2009). Customer churn prediction using improved balanced random forests. *Expert Systems with Applications*, *36*(3), 5445-5449.

Yan, L., Wolniewicz, R. H., & Dodier, R. (2004). Predicting customer behavior in telecommunications. *IEEE Intelligent Systems*, *19*(2), 50-58.

Ying, W., Li, X., Xie, Y., & Johnson, E. (2008, July). Preventing customer churn by using random forests modeling. In *Information Reuse and Integration, 2008. IRI 2008. IEEE International Conference on* (pp. 429-434). IEEE.

Yu, L., & Liu, H. (2003). Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the 20th international conference on machine learning (ICML-03)* (pp. 856-863).