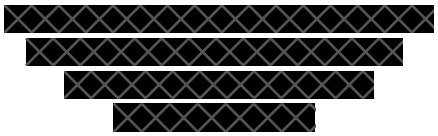
Predicting Customer Relationship with Boosting Ensemble Algorithms

Leon Schulz



***Abstract*—Customer Relationship Management (CRM) improves customer loyalty and profitability. Prediction of user behaviour based on historical user is a central method in CRM. This study examines which classification algorithms are suitable to analyze three key CRM metrics on a large marketing data set.**

***Index Terms*—customer management, classification, imbalanced data, missing values, boosting**

I. Introduction

Customer relationship management (CRM) is key for companies to understand their customer groups [1]. By sensing customer needs and remembering preferences, businesses learn from the past to enhance future customer relationships [2]. Thereby individual customers are retained and one-to-one relationships are further developed. This in turn improves customer loyalty and overall profitability [1].

CRM originated from advancements in information technology [1]. Today, data mining techniques are applied for the collection of vast amounts of customer data and their utilisation in strategic decision making [2]. Supervised machine learning algorithms enable the connection of input data to a specific target. In CRM these algorithms are applied to create scores which report the outcome in a single, interpretable number. A frequent application of such scoring methods is prediction. Here classification or estimation methods learn from past data to make predictions on current inputs.

The ACM Special Interstest Group on Knowledge Discovery and Data Mining (KDD) acknowledged the relevance of this topic in their 2009 KDD Cup [3]. Participants were provided with a large marketing database [4] and challenged to predict three target values related to customer retention and relationship development. The selected metrics were defined as the propensity of customers to stop doing business with the entity (churn), purchase a product or service (appetency), or buy an upgrade or add-on (up-selling). Given the binary nature of the target values, the problem setting suggests the use of classification algorithms.

As for the prevailing relevance of machine learning applications in CRM, we address the question:
"How can classification algorithms be applied to predict churn, appetency, and up-selling?"

II. Methods

Following the specifications of the KDD Cup 2009 [5], Area Under the Curve (AUC) was used as a performance measure. AUC is based on the principles of true and false positives and calculated by measuring the area under the receiving operator characteristic (ROC) curve. AUC is "equivalent to the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance" [6]. The entire study was realised in Python [7] and the AUC score implemented in the scikit-learn package [8] was used. All measures described below were taken to maximise this score. In the following sections the procedure is presented in a near chronological order.

A. Data

For the 2009 KDD Cup the French Telecom company Orange provided a marketing dataset of 100000 entries [4]. The data was equally split in train and test data. The full dataset contains 14740 variables. However, also a smaller dataset of 230 pre-selected variables was provided. Herein the first 190 variables are numerical and the last 40 are categorical. This dataset was selected for this study. Since Orange provided real customer data, all entries were fully encrypted to ensure anonymity. Hence, a qualitative interpretation of the variables is impossible. Additionally, target labels for the target variables were only made available for the train set. Therefore the performance of the model within this paper was measured against a validation set consisting of 10000 entries, or twenty percent of the train set.

Prior to the challenge the following characteristics of the dataset were disclosed [4], [9]: (1) large amount of missing values, (2) categorical variables with large number of features and (3) unbalanced class distribution. Through exploration of the data we were able to confirm these characteristics and comprehend their magnitude.

In the full train set 70% of entries are missing. However, there are significant differences in numerical and categorical variables. While 79% of numerical values are missing, only 22% are missing in categorical variables. Figure 1 visualizes this relationship and further indicates that in the full set 68% of variables (n=156) are missing more than 70% of their entries. Nineteen columns are completely empty. In contrast, not a single row was missing all values.

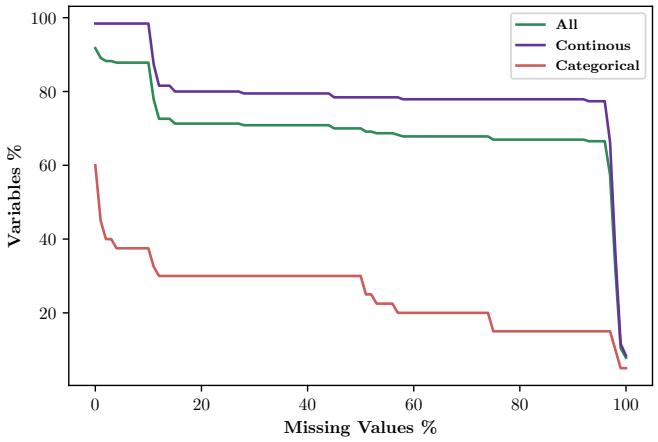


Fig. 1. Missing Values in Training Set

Considering the number of features of the categorical variables, we found multiple variables with over 1000 categories. On top of that strong imbalance in the distribution of the classes was observed. As portrayed in Figure 2, there are multiple variables in which individual categories take up almost 90% of all values.

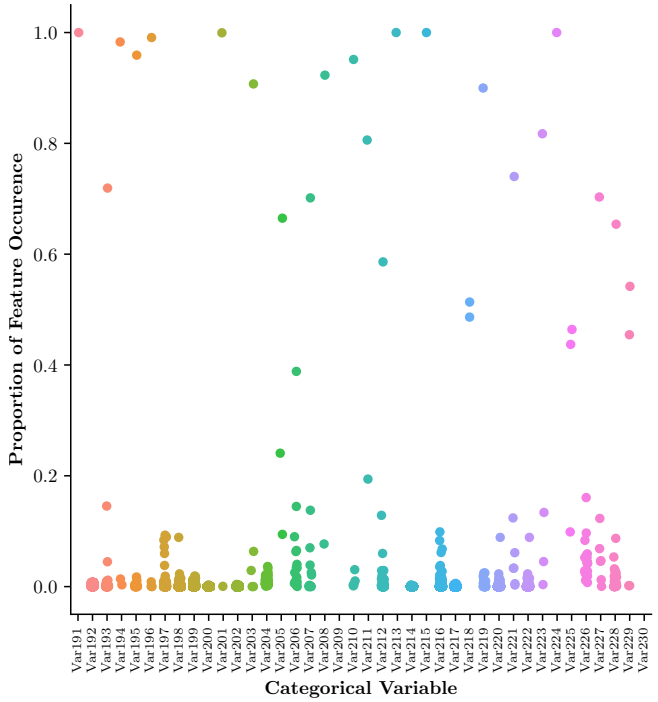


Fig. 2. Unique Feature Count Proportions in Categorical Variables

Similar imbalance is also present in the target labels. In all three outcome variables, true labels are much rarer than negative ones. In both churn and up-selling around 7.3% entries of the labels are true. Appetency only has 1.78 % (n=890) true labels.

Additionally, strong outliers and non-normal distributions in several continuous variables were norticed.

B. Preprocessing

Missing data not only reduces statistical power, but can also produce biased estimates which lead to invalid conclusions Kang (2013). Commonly, when treating missing data it is first classified in the three categories defined by Ruben (1976). However, since all variables are encrypted, we can not exclude that the variables are missing not at random (MNAR). Hence it can not be ruled out that selectively deleting entries in a variable leads to bias. For this reason variables are either deleted entirely or all missing values were imputed. The 19 variables which contained no entries were dropped. Additionally, deleting those variables which are missing 90% or more of their entries improved performance. On the remaining 76 variables, different imputation approaches were tested following [12]. For the numerical variables an imputation with the mean performed well. Missing values in categorical variables were replaced with a separate category.

In addition, categorical variables were treated in respect of their large number of attributes with few entries. Such rare labels can not only cause overfitting [13], but in this case also led to the validation set containing attributes that were not present in the training set. Classification algorithms can not interpret these characteristics. To solve this problem rare label encoding was applied. All attributes that make up less than 0.05% of the entries of a variable were combined into a new attribute. This method outperformed frequency encoding, where attributes are assigned labels according to the frequency of their occurrence.

Furthermore, the categorical variables had to be formatted for use in the classification algorithms. First, string attributes were converted into numbers using the Label Encoder method of scikit-learn [8]. Then all categorical variables were one hot encoded using the Pandas library [14]. This means that a separate binary variable was created for each attribute, resulting in over 3400 variables.

Classifiers performed badly on this train set. Imbalance of the target labels was perceived as the reason. Following Brownlee (2015) a variety of methods were applied. A combination of two resampling techniques proved successful. First, for each target variable, a stratified split of the sample into train and validation set was applied using scikit-learn [8]. Thereby, in each of the three pairs of train and validation set, the proportions of true and false target labels in the full sample is preserved for both train and validation set. Due to this separation, all previous steps had to be performed separately for all three target variables. Following this logic, oversampling of the minority class of true target labels was applied to all train sets. Herefore, scikit-learn [8] was employed once more.

C. Classification

Classification is a fundamental machine learning technique [16]. In this study three separate binary classification problems area analysed. In each case all variables are measured to

estimate a probability for each individual to belong to one of two target labels [17].

The first classification algorithm to be implemented was the Naive Bayes [8]. The algorithm is frequently used as a baseline model [18]. This is due to its simple assumption of conditional independence [16]. However, the algorithm was quickly discarded due to its poor performance. On a not yet resampled data set it barely exceeded an AUC of 0.5.

Second, logistic regression was applied [8]. While known as an efficient and powerful method in classification, several basic assumptions must be met [19]. These include independence of errors, linearity in the logit for continuous variables, absence of multicollinearity and lack of strongly influential outliers. Since the data heavily comprises multiple of these assumptions, the train set was strongly adapted to allow valid interpretation of the results. On an initial non-resampled data set the algorithm only narrowly outperformed Naive Bayes. Also a cost-sensitive implementation, which is suited for imbalanced data, did not significantly improve performance. For this reason and due to the described violation of the assumptions logistic regression was not considered for further course of action.

Reconsidering the basic characteristics of the data set, decision trees algorithms came into focus. Depending on their implementation these algorithms are not sensitive to missing data [20]. Although not entirely robust, they are also known to perform well on imbalanced data sets [21]. In particular, this property was tested with respect to ensembles of decision trees [22]. Ensemble methods are meta-algorithms which combine multiple learners to increase the predictive performance [23]. These methods include random forests [24] and boosting [25]. The fundamental difference between these methods lies in the construction of the trees and the way the results are combined. In random forests each tree is built independently and results are combined at the end. Boosting algorithms follow an additive model and combine results continously.

The performance of the following decision tree classifiers was compared: (1) Decision Tree Classifier (DTF), (2) Random Forest Classifier (RFT), (3) AdaBoost Classifier (AB) [26], (4) Histrogram-based Gradient Boosting Classification Tree (HGBM) [27] and (5) XGBoost (XGB) [28] . The ensemble methods (2-5) were selected based on their current popularity [29]. Algorithms 1-4 were implemented with scikit-learn [8]. XGBoost was implemented using the XGBoost library [30]. Table I depicts the performance of the algorithms on the resampled training set, without hyperparameter tuning.

TABLE I
AUC FOR DECISION TREE CLASSIFIERS - INTERIM RESULTS

	DTF	RFT	AB	HGBM	XGB
Churn	0.553	0.689	0.72	0.74	0.7
Appetency	0.568	0.768	0.791	0.81	0.78
Up-Selling	0.707	0.85	0.867	0.865	0.855
Mean	0.609	0.769	0.792	0.805	0.778

With a mean AUC of 0.805 HGBM performed best. HGBM is based on LightGBM, which is a gradient boosting decision tree that was specifically developed to speed up the training process [27]. This is achieved by combing Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Building (EFB). GOSS excludes data instances with small gradients, while EFB bundles mutually exclusive features. Based on its performance HGBM was selected for further improvement using hyperparameter tuning with scikit-learn's GridSearchCV [8]. Various combinations of relevant parameters were compared to reach optimal performance.

D. Results

The final score was calculated by repeating the presented procedure on multiple train and validation splits. These had the same 80/20 split but a different allocation of data points. HGBM produced a mean AUC score of 0.8147 on these repititions. Churn (AUC = 0.7363) was the most difficult problem. Appetency (AUC = 0.8382) was the second most difficult, making up-selling (AUC = 0.8697) the easiest. AUC scores are calculated from the ROC curves. Figure 3 presents the ROC curves of a selected iteration. Additionally, confusion matrices of this iteration on a probability threshold of 0.5 are attached in Appendix A. Table II compares the mean result with the official reference scores of the KDD Cup 2009 [31].

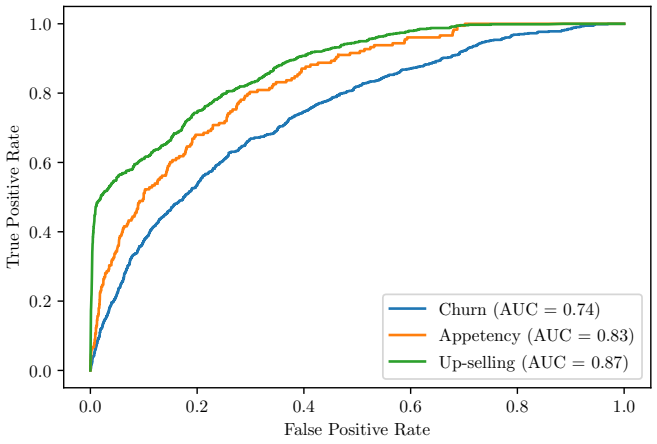


Fig. 3. ROC Curves

TABLE II
AUC - FINAL RESULTS

	Baseline	Orange	HGBM
Churn	0.6468	0.7435	0.7363
Appetency	0.6453	0.8522	0.8382
Up-Selling	0.7211	0.8975	0.8697
Mean	0.6711	0.8311	0.8147

This shows that the Naive Bayes Classifier (AUC = 0.6711), which serves as the baseline result, was outperformed with regard to all target labels. The presented results would therefore have qualified for the challenge. However, the in-house

classifier from Orange (AUC = 0.8311) was not surpassed. Hence, the produced scores would not have been sufficient to beat the challenge.

III. Discussion

In this study, a model was sought that best classifies three CRM target labels on a specific data set. Hence, results should not be understood as a generalisation for the prediction of churn, appetency or up-selling. Nevertheless, findings hint at interesting insights regarding the usefulness of certain methods for data sets with many missing values, a large number of characteristics in categorical variables, and a highly unbalanced class distribution.

Results show that boosting algorithms perform very well under these circumstances. Specifically the HGBM algorithm proved to be well suited. This result is in line with Rodriguez (2011) who acknowledged that such algorithms are known to perform well on imbalanced data sets. In the context of combating imbalanced classes resampling techniques and especially up-sampling has proven to be useful. This supports the findings of [32] who reached similar findings on large data sets in multi-classification problems. Finally, rare label encoding has proven to be a suitable method to deal with the many classes of categorical variables. Particularly a low threshold has proven to be efficient, indicating that performance increases when keeping features with multiple occurrences.

Considering the final AUC Score, it has to be acknowledged that the Orange in-house classifier was not beaten. However, if one compares the results, a different picture emerges. Only 4% (21/453) of the teams with valid submissions beat the in-house classifier. The achieved AUC of 0.8147 would have finished in the top 7% (32/453). To understand which alternative approaches could have improved on the model the insights of the official proceedings [33] and the results of the winners [34] were consulted.

First, it was noticed that users of decision trees did not simply replace missing entries [33]. Instead they used 'surrogate variables' at each division of the dichotomous trees. Second, discretization of the continuous variables was used [33]. The authors stated that this method was especially useful due to the existence of outliers and non-normality of distribution in continous variables. In addition, participants normalized and standardized continuous variables. [33]. With regard to model selection, boosting ensemble algorithms were successfully applied in the challenge [33]. However, other participants were very successful with bagging [24] and forward model selection [35]. This method should be highlighted as it was used by the winners of the challenge [34]. Also it appears noteworthy that the winners used a cross-validation approach, although due to time constraints this was limited to only two iterations.

IV. Limitations

As with all research our results should be interpreted with caution. First of all, it is important to mention that we calculate our AUC score on a classification of the validation set. The reason for this is the lack of access to the target labels of the test set. This is especially delicate since it has been found that boosting methods on this data set tend to overfit when applied on the training data [33].

Second, due to restrictions in computational power, the analysis was based on the small data set. However, results published after the challenge were all based on the large data set. The analysis of the large data set led to better results for all participants [33]. Hence, an application of the presented methods to the larger data set has the potential to improve performance.

Third, the study was limited in both time and in use of its programming language. While previous participants of the study did not replace missing values when using decision trees, the scikit-learn library [8] does not support missing values. Since the time frame did not allow a corresponding re-implementation of the models, simple imputations were applied. Especially the use of the mean imputation of the continuous variables is inconvenient, since it can lead to biased estimates of variances and covariances [36].

V. Conclusion

Churn, appetency and up-selling are key metrics in customer relationship management. Boosting ensemble algorithms emerged as an efficient method for their classification on the given data set. Especially the histogram-based gradient boosting classification tree proved to be a high-performance solution. From these results, no generalisations can be derived regarding the analysis of CRM metrics. However, the results do give an idea of which methods can be useful for other classification problems on data sets with similar characteristics. Imbalance was handled well by a combination of resampling and the use of ensembles of decision tree classifiers. In case of many missing values, simple imputation methods such as substitution by the mean and creation of separate categories were successful. The problem of categorical variables with many attributes was solved efficiently with rare label coding.

References

[1] Chen I.J. and Popovich K.. (2003). Understanding customer relationship management (CRM): People, process and technology. Business Process Management Journal, Vol. 9 No. 5, pp. 672-688.

[2] Berry M.J.A. and Linoff G.S. (2004). Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management. John Wiley & Sons.

[3] Assocation for Computing Machinery. (2009). KDD Cup 2009: Customer relationship prediction, viewed 16 September 2020, https://www.kdd.org/kdd-cup/view/kdd-cup-2009

[4] Assocation for Computing Machinery. (2009). KDD Cup 2009: Customer relationship prediction, viewed 16 September 2020, https://www.kdd.org/kdd-cup/view/kdd-cup-2009/Data

[5] Assocation for Computing Machinery. (2009). KDD Cup 2009: Customer relationship prediction, viewed 16 September 2020, https://www.kdd.org/kdd-cup/view/kdd-cup-2009/Tasks

[6] Fawcett T. (2006). An introduction to ROC analysis. Pattern Recognition Letters, Vol. 27, pp. 861-874.

[7] Van Rossum, G. & Drake, F.L., 2009. Python 3 Reference Manual, Scotts Valley, CA: CreateSpace.

[8] Pedregosa et al. (2011). Scikit-learn: Machine Learning in Python. JMLR 12, pp. 2825-2830.

[9] Lemaire V. (2009). KDD Challenge 2009. Orange Labs R&D., viewed 16 September 2020, http://www.vincentlemaire-labs.fr/kddcup2009/ChallengePresentation_03192009.pdf
[10] Kang H. (2013). The prevention and handling of the missing data. Korean journal of anesthesiology, 64(5), 402–406.
[11] Ruben D.B. (1976). Inference and Missing Data. Biometrika Vol. 63 No. 3, pp. 581–90.
[12] Gelman A. and Hill J. (2006). Missing-data imputation. In Data Analysis Using Regression and Multilevel/Hierarchical Models (Analytical Methods for Social Research, pp. 529-544). Cambridge: Cambridge University Press.
[13] Charfaoui Y. (2014). Hands-on with Feature Engineering Techniques: Common Issues in Datasets , viewed 16 September 2020, https://heartbeat.fritz.ai/hands-on-with-feature-engineering-techniques-common-issues-in-datasets-a0c2cf97b1a5
[14] McKinney W. (2010). Data Structures for Statistical Computing in Python. Proceedings of the 9th Python in Science Conference pp. 51–56.
[15] Brownlee, J. (2015). 8 Tactics to Combat Imbalanced Classes in Your Machine Learning Dataset, viewed 16 September 2020, https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/
[16] Zhang H. (2004). The Optimality of Naive Bayes. University of New Brunswick, Faculty of Computer Science.
[17] Parmigiani G. (2001). International Encyclopedia of the Social & Behvioral Sciences, pp. 3327-3334.
[18] University of Cambridge (2016). Further notes on Naive Bayes. Machine Learning and Real-world Data Handbook.
[19] Stolzfus J.C. (2011). Logistic Regression: A Brief Primer. Academic Emergency Medicine. Vol. 18, pp. 1099-1104.
[20] Pedregosa et al. (2011). Scikit-learn: Machine Learning in Python , viewed 16 September 2020, https://scikit-learn.org/stable/modules/tree.html
[21] Boyle T. (2019). Dealing with Imbalanced Data. towards data science, viewed 16 September 2020, https://towardsdatascience.com/methods-for-dealing-with-imbalanced-data-5b761be45a18::text=Decision%5C%20trees%5C%20frequently%5C%20perform%5C%20well,as%5C%20compared%5C%20to%5C%20logistic%5C%20regression.
[22] Rodriguez J.J., Diez-Pastor J.F., Garcia-Osorio C. (2011). Ensembles of Decision Trees for Imbalanced Data. In: Sansone C., Kittler J., Roli F. (eds) Multiple Classifier Systems. MCS 2011. Lecture Notes in Computer Science, Vol. 6713. Springer, Berlin, Heidelberg.
[23] Zhou S.K. (2016). Introduction to Medical Image Recognition, Segementation and Parsing. In: Medical Image Recognition Segmentation and Parsing, pp. 1-21.
[24] Breiman L. (2001). Random Forests. Machine Learning Vol. 45, pp. 5–32.
[25] Friedman J.H. (2000). Additive Logistic Regression: A Statistical View of Boosting. The Annals of Statistics. Vol. 28 No. 2, pp. 337-407.
[26] Freund Y. & Schapire R. (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting.
[27] Ke G. et al. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.
[28] Chen T. & Guestrin C. (2016). XGBoost: A Scalable Tree Boosting System. KDD '16, August 13-17, 2016, San Francisco, CA, USA.
[29] Slatery K. (2020). Decision Trees: Understanding the Basis of Ensemble Methods. towards data science, , viewed 16 September 2020, https://towardsdatascience.com/decision-trees-understanding-the-basis-of-ensemble-methods-e075d5bfa704
[30] XGBoost (2020). XGBoost for Python, viewed 16 September 2020, https://xgboost.readthedocs.io/en/latest/index.html
[31] Assocation for Computing Machinery (2009). KDD Cup 2009: Customer relationship prediction, viewed 16 September 2020, https://www.kdd.org/kdd-cup/view/kdd-cup-2009/Results
[32] Tanha J., Abdi Y., Samadi N. et al. (2020). Boosting methods for multi-class imbalanced data classification: an experimental review. J Big Data Vol. 7, N. 70
[33] Guyon I. (2009). Analysis of the KDD Cup 2009: Fast Scoring on a Large Orange Customer Database. Journal of Machine Learning Research. Workshop and Conference Proceedings Vol. 7, pp. 1-22.
[34] Niculescu-Mizil A. (2009). Winning the KDD Cup Orange Challenge with Ensemble Selection. Journal of Machine Learning Research. Workshop and Conference Proceedings Vol. 7, pp. 23-34.
[35] Caruana R. & Niculescu-Mizil A. (2006). An Empirical Comparison of Supervised Learning Algorithms. Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA.
[36] Soley-Bori M. (2013). Dealing with missing data: Key assumptions and methods for applied analysis.

APPENDIX

A. Confusion Matrices

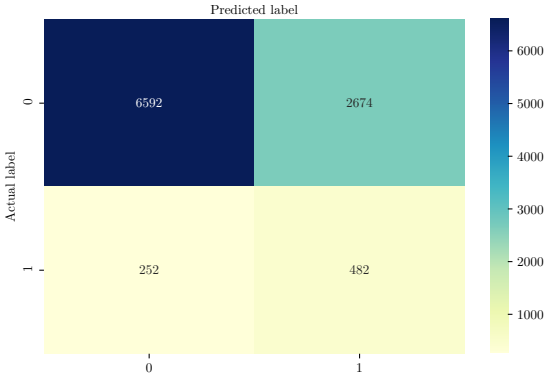


Fig. 4. Confusion Matrix - Churn (Threshold: 0.5)

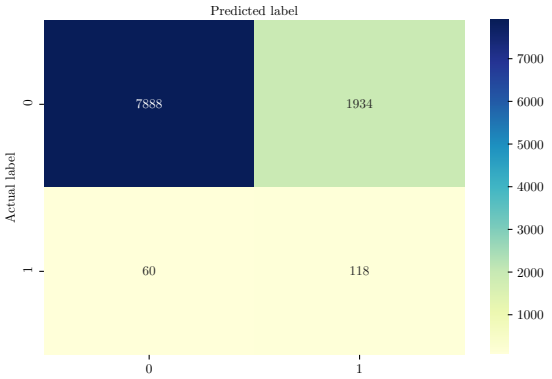


Fig. 5. Confusion Matrix - Appetency (Threshold: 0.5)

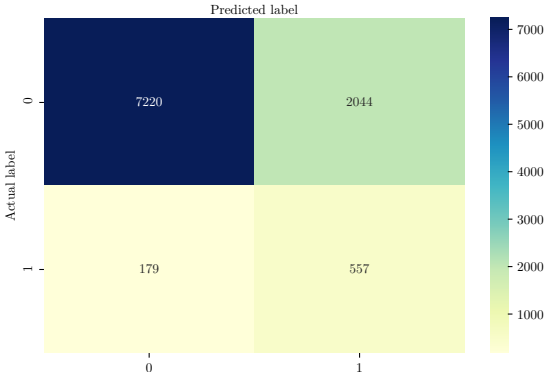


Fig. 6. Confusion Matrix - Up-selling (Threshold: 0.5)