DEPARTMENT OF INDUSTRIAL ENGINEERING

BOGAZICI UNIVERSITY

IE 582 STATISTICAL LEARNING FOR DATA MINING

Project:

Gender Prediction for an Imbalance Binary Classification Case

Prepared by Group 10

Burak Can Helvacı - 2020702081

Cumhur Kılıç - 2021700090

Pınar Özdemir - 2020808012

Submitted to Assoc. Prof. Mustafa Gökçe Baydoğan

ISTANBUL - 24.01.2022

TABLE OF CONTENT

1. INTRODUCTION

In this project, unstructured data of the online retailer was provided to build a classification model based on the methods described in the IE 582 course. After feature engineering steps, the aim of this project is to build a model which can work for the imbalanced binary classification problem of gender prediction and to predict genders of the users of an online retailer with high accuracy. We determined the problem as an imbalanced binary classification problem because our target column, “gender”, consists of two classes, “Kadın” and “Erkek”. Besides, there was a significant class imbalance among classes of “Kadın” (66%) and “Erkek” (34%) as in Table 1.1 and Figure 1.1 after feature engineering process.

Table 1.1. Distribution of Classes

Erkek	Kadın	Total
1939	3679	5618

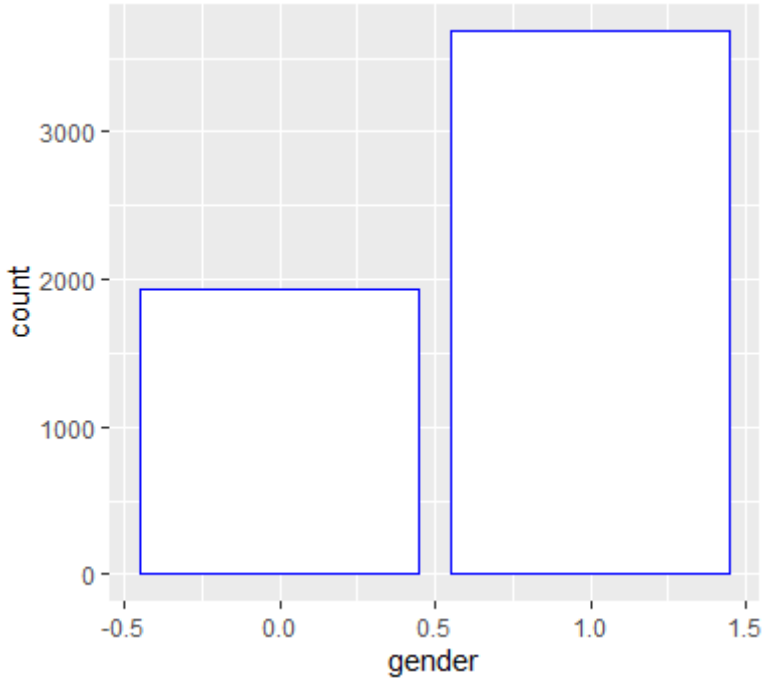

Figure 1.1. Distribution of gender on train data set after feature engineering (0=Erkek, 1=Kadın).

We tackled the problem with two coding languages R and Python.

Feature Engineering for R Part

In R language part, we investigated and analyzed the characteristics of the features with the help of "glimpse", "skim" and "dim" functions. Then, we applied functions of group by, count and sort several times to unstructured data for feature extraction purposes. Our resulting data table for train data and test data are in Table 1.2 and Table 1.3.

As mentioned earlier, the provided data is unstructured. For one customer, there might be thousands of records in this unstructured data. Therefore, feature extraction and preprocessing of the data are vital parts of this project. Before starting to model building process, regular data matrix should be obtained and every customer in the unstructured data should be expressed as one row of the regular data matrix.

Unstructured train data consists of 5,493,268 rows and 19 columns. Records with same unique_id and time_stamp were accepted as duplicate entries and only one of these records were kept. After removal of duplicate entries, data scales downed to 2,043,321 rows. Some features were removed from the unstructured data. Those are businessunit, product_name, brand_name, Level1_Category_Name, Level2_Category_Name and Level3_Category_Name. Businessunit and product_name was removed because they carry similar information as features named Level1_Category, Level2_Category and Level3_Category. Level1_Category_Name, Level2_Category_Name and Level3_Category_Name was removed because there are also Level1_Category_Id, Level2_Category_Id and Level3_Category_Id in the data and they carry same information. To understand data better, all blank entries were converted to NA, and total NA counts for all features were calculated. NA entries of selling price were converted to 0 for ease of calculation. Features that were created for features engineering purposes will be explained below.

Total number of action (log in), visit, search, favorite, basket and order were calculated for every user. It was realized that train dataset includes information of 5618 unique customers. Respective features are num_of_login, num_of_visit, num_of_search, num_of_fav, num_of_basket and num_of_ord.

Total number of log ins by user in work hours (8am-18pm) and total number of log ins by user at weekdays were calculated because, we thought these two features might be important to make distinction between male and female customers. In our opinion, customers who are more active

in work hours and weekdays are probably female customers due to employment rate of the woman in Turkey. Women workers approximately less than half of the male workers. Two related features were also created. Those two features were named is_weekday_log_more and is_workhour_log_more. First feature takes value of 1 if customer is more active in weekdays than weekends, it takes value of 0 elsewhere. Second feature takes value of 1 if customer is more active in work hours, it takes value of 0 elsewhere.

Sell_price_total, sell_price_ave, is_more_than_avg_selling features were created to able to make distinction about the gender of the customers. First one is total price of the products that customer has reached. Second one is average price of the products that customer has reached. Last one is binary feature which takes value of 1 if average price of the products a customer reached is greater than the average price of all products in the whole data.

Num_of_gender_woman, num_of_gender_man, and num_of_gender_unisex features are the number of times that a customer reached products with product gender of woman, number of times that a customer reached products with product gender of man and number of times that a customer reached products with product gender of unisex. Likewise, most_visited_gender_woman, most_visited_gender_man and most_visited_gender_unisex features were created to monitor the gender of the most reached product of the customers. Those are binary variables. Most_visited_gender_woman feature takes value of 1 if the product gender of the products that a customer reached are mostly woman and takes value of 0 elsewhere. Remaining two features were created with the same logic.

Lastly, visited_unique_brand_count, visited_unique_cat1_count, visited_unique_cat2_count and visited_unique_cat3_count features were created. Those are number of distinct brands a customer reached, number of distinct category 1 id's a customer reached, number of distinct category 2 id's a customer reached and number of distinct category 3 id's a customer reached respectively.

After these steps, regular data matrix was created. There are 5618 rows and and 26 columns in total. 26 columns include 25 extracted features and a response. All of the final features are numerical.

After feature engineering, imbalance problem on the data set was fixed by some resampling techniques like over (or up), under (or down), both (over + under) sampling, synthetic minority sampling technique (SMOTE), and randomly over sampling examples (ROSE). Furthermore, data was also weighted by 50/50 as following:

$$Weight_i = \frac{1}{Total\ number\ of\ instance\ that\ have\ some\ class\ with\ instance\ i} \times (0.5)$$

The best performance of SGB method was obtained by over resampling method because our data set, both train and test, have minority instances for “Erkek” class. Over sampling method works with minority class. It replicates the observations from minority class to balance the data. The performance obtained by under sampling method supplied also well results after over sampling method but removing observations may cause the training data to lose important information pertaining to majority class. Hence, over sampling technique was a suitable manipulation for our imbalance data set without loss information.

Table 1.2. Head of final train data set after sort-count step for model part

num_of_login	num_of_basket	num_of_fav	num_of_ord	num_of_search	num_of_visit
1616	108	92	1	671	744
2616	26	892	1	916	781
388	32	0	5	216	135
5441	300	130	13	2125	2873
4039	2616	1	5	588	3260
4667	144	329	9	831	3354
num_of_gender_woman	**num_of_gender_man**	**num_of_gender_unisex**	**num_of_login_workhour**	**num_of_login_otherhour**	**num_of_login_weekday**
316	25	961	445	1171	1616
1455	20	873	2311	305	2616
362	0	14	183	205	388
3993	159	876	2907	2534	5441
2885	17	1031	1559	2480	4039
1102	1125	1621	837	3830	4667
num_of_login_weekend	**is_weekday_log_more**	**is_workhour_log_more**	**sell_price_total**	**sell_price_ave**	**is_more_than_avg_selling**
0	1	0	264638.88	163.7616832	0
0	1	1	400910.08	153.2530887	0
0	1	0	106420.85	274.2805412	0
0	1	1	1563801.83	287.4107388	0
0	1	0	551369.11	136.5112924	0
0	1	0	888279.96	190.3321106	0
visited_unique_brand_count	**most_visited_gender_woman**	**most_visited_gender_man**	**most_visited_gender_unisex**	**visited_unique_cat1_count**	**visited_unique_cat2_count**
255	0	0	1	10	41
441	1	0	0	11	58
45	1	0	0	8	18
682	1	0	0	11	60
241	1	0	0	9	46
645	0	0	1	11	59

visited_unique_cat3_count	gender
87	1
183	1
32	1
179	1
122	1
201	0

Table 1.3. Head of final test data set after sort-count step for model part

unique_id	num_of_login	num_of_basket	num_of_fav	num_of_ord	num_of_search
9	1608	47	46	7	826
18	9882	328	428	65	2867
21	2139	18	272	2	682
25	1073	13	79	4	339
31	8046	268	227	3	2276
32	1342	30	125	6	352

num_of_visit	num_of_gender_woman	num_of_gender_man	num_of_gender_unisex	num_of_login_workhour	num_of_login_otherhour
682	1322	90	119	117	1491
6194	7215	699	1315	5254	4628
1165	1459	191	390	682	1457
638	679	47	232	604	469
5272	3720	112	2727	1501	6545
829	1027	2	278	544	798

num_of_login_weekday	num_of_login_weekend	is_weekday_log_more	is_workhour_log_more	sell_price_total	sell_price_ave
1250	358	1	0	359026.16	223.274975
7268	2614	1	1	1078626.88	109.150666
1477	662	1	0	620537.06	290.106152
441	632	0	1	244261.13	227.643178
5491	2555	1	0	1808405.29	224.758301
954	388	1	0	133580.86	99.5386438

is_more_than_avg_selling	visited_unique_brand_count	most_visited_gender_woman	most_visited_gender_man	most_visited_gender_unisex	visited_unique_cat1_count
1	140	1	0	0	11
0	698	1	0	0	11
1	244	1	0	0	10
1	185	1	0	0	10
1	1011	1	0	0	11
0	182	1	0	0	11

visited_unique_cat2_count	visited_unique_cat3_count
43	86
71	230
36	105
39	73
58	213
38	101

While train data set has the feature columns and the target column, test data set does not have the target column (gender). Thus, we built binary classification models using train set and calculated predictions on test set. All in all, our pathway consists of two branches. The first is to overcome the imbalance problem and the second is to predict gender for test data set by means of a suitable classifier training on train data. We studied with classifiers such as Decision Trees, Random Forest, XGBoost, GBM and Stochastic Gradient Boosting (SGB) with cross-validations for R part and evaluated the performance of our models by receiver operating characteristic (ROC), specificity (SPEC) and sensitivity (SENS) in training part. In addition to this, balanced error rate (BER), area under the receiver operating characteristic curve (AUC) and total score is calculated in test part.

Feature Engineering for Python Part

First, we removed the "time_stamp" from the given data and added the hour and day information from it as a new feature. we have removed features such as “content_id” that will not help the development of the model. Afterwards, we determined the features that represented both id feature and name feature like “Level1_Category_Id” and “Level1_Category_Name” and removed these features that contains name information from data. We converted all categorical features to numeric values using target encoding. Instead of using target encoding for product name and brand name, we implemented a different method. We created a vector from all the words that exist in the content of product name. we parsed the 15 most used words and reduced size of vector. Then, we applied same procedure for brand name feature. we added 30 new features to the train set. The new features we created from the product name are named sequentially from 0-14. The features that we created from the brand name were named between b0-b14. After the feature engineer, we started to train the model with 42 features.

In our second model, after converting the data with the same procedure, we considered the problem as a multiple instance problem. Because there was a lot of data with the same "unique_id" feature of the model, and the given target data with the same "unique_id" feature had to be the same. Therefore, we grouped the data with equal "unique_id" according to this feature and created new data according to the mean and quantiles of other features. In the same way, we have calculated the required feature values for each user by transforming the test data.

2. RELATED LITERATURE

Imbalance binary classification

Imbalanced classification is a supervised learning problem. In this situation, one class outnumbers other class by a large proportion. This problem is encountered more frequently in binary classification problems rather than multi-level classification problems. In binary classification case, the class with higher number of observations is called “majority class” and the other is called “minority class” (Brownlee, 2019). For our cases, the majority class is “Kadın” and the minority class is “Erkek”.

Imbalance distribution of the target variable in the data set is a problem for predictive performance, so there are various sampling techniques to deal with class imbalance problem. Used ones in this project are class weight, over (or up) sampling, under (or down) sampling, both (over + under) sampling, synthetic minority sampling technique (SMOTE), and randomly over sampling examples (ROSE). Each of these sampling techniques apply a different pathway to solve the imbalance among classes (Barandela et al., 2004; Martin, 2016; Branco et al., 2016; Menardi and Torelli, 2014).

Class weights

Coefficients of cost functions are basically adjusted by class weights, so increasing weightage of minority class results in higher penalty for error of minority class (“Erkek”). Hence, class weights approach helps to decrease this kind of errors (“Analytics Vidhya-2”, 2020).

Over (or up) sampling

Over sampling method works with minority class. It replicates the observations from minority class to balance the data. It is also known as up sampling. Similar to under sampling, this method is made up of two types: Random Oversampling and Informative Oversampling. Random oversampling balances the data by randomly oversampling the minority class. Informative oversampling uses a pre-specified criterion and synthetically generates minority class observations. One great benefit of over sampling is that there is no information loss owing to data removed. However, the overfitting is cons of this method, because oversampling simply adds replicated observations in the original data set, it ends up adding multiple observations of several types (“Analytics Vidhya-1”, 2016).

Under (or down) sampling

Under sampling method works with majority class. It decreases the number of observations from majority class to balance the data set. If the data set is huge, under sampling method is best to use. Moreover, reduction on the number of training samples improves run time and storage troubles. There are two types of it. These are Random and Informative. Random under sampling method randomly chooses observations from majority class which are eliminated until the data set gets balanced. Informative under sampling follows a pre-specified selection criterion to remove the observations from majority class. ("Analytics Vidhya-1", 2016; Kuhn et al., 2018).

SMOTE sampling

SMOTE is kind of mixture method of over and under sampling. It adopts an approach that optimizes the overfitting and loss information by over sampling the minority class and under sampling the majority class (Torgo, 2010; Chawla et al., 2002). In this case, the minority class is oversampled with replacement and majority class is under sampled without replacement ("Analytics Vidhya-1", 2016).

ROSE sampling

The data created from over sampling have the expected number of repeated observations. Data created from under sampling is deprived of important information from the original data. This leads to inaccuracies in the resulting performance. To not face with these issues, ROSE helps to generate data synthetically as well. The data created using ROSE is considered to provide a better estimate of original data. So, ROSE synthetizes samples by expending both attributes of minority and majority classes. This technique combined with a more robust algorithm (random forest, boosting) can lead to exceptionally high accuracy ("Analytics Vidhya-1", 2016; Lunardon et al., 2014; Menardi and Torelli, 2014).

3. APPROACH

Approach for R Part

In line with our project goal, we tried to find a binary classification model which can give predictions with highest possible accuracy for the test data. As stated previously, five different classification models (Decision Trees, Random Forest, Stochastic Gradient Boosting (SGB), XGBoost and GBM) were applied with cross-validation for R part ("caret" and "rpart" packages) of this project.

Decision Tree

Decision tree method with CART algorithm is one of the classification methods used for this project. If we briefly look at the decision tree and its algorithm, this approach depends on a tree-based model. In this method, the model tries to divide the feature space into smaller parts with similar responses. The predictions are produced by different models for each part or namely region. In addition to the algorithm of the decision tree method, it is not preferred for more complex cases, even if it requires low computation power. (Breiman et al., 1984).

As for tuning parameters for the decision tree method, complexity parameter (cp) and minbucket number (min.buc) were evaluated for improvement of the decision tree results. If we take a look at the task of these parameters in the decision tree model, while the complexity parameter controls the size of the decision tree, the number of minbucket examines the smallest number of observations let to use in a node.

In this project, different complexity parameters within 0.0005 and 0.007 values (cp = 0.0005, 0.01, 0.05, 0.005, 0.001, 0.007); and different minbucket values (min.buc = 6, 8, 10, 12) were benefitted for tuning the decision tree model.

Random Forest

As a mixture of classification and regression methods, random forest is made up of various uncorrelated decision trees. During the learning issue, all decision trees grow under randomization. In this forest, each tree can make a decision for a classification and then the class having the most votes decide the final classification.

The tuning parameter of random forest is "mtry". "mtyr" examines the number of variables suitable for splitting at each tree node. Default value of mtyr for classification is calculated by means of root mean square of the number of features (Breiman, 2001). In random forest

approach part of this project, different mtyr values (3, 5, 7, 9) were used for tuning the random forest model.

Stochastic Gradient Boosting (SGB)

Firstly, in literature, the random forest algorithm was combined with gradient boosting by Friedman and he created a new approach called as stochastic gradient boosting (Friedman, 2002). One of the outstanding methods for both classification and regression models is stochastic gradient boosting approach because it makes iteratively a group of trees. In other words, even if shallow trees are weak models, boosting of these weak models with suitable tuning parameters may give the best model in between the other methods.

As for the tuning parameters of gradient boosting method, number of trees, learning rate (or namely "shrinkage"), tree depth (Int.depth), and minimum number of observations in terminal nodes (min.obs.node) are the parameters of it. For this project, different number of trees values (400, 600, 700, 800, 900), different shrinkage values (0.05, and 0.01), different minimum number of observations values (5, 7, 9, 11, 13, 15, 17, 19), different tree depth values (5, 7, 9, 11, 13, 15, 17, 19) were used for tuning the model with stochastic gradient boosting model.

XGBoost

To tune the related parameters and evaluate the performance of the model caret package was used. 10 fold cross validation with 5 repetitions were used in the model building and parameter tuning process.

Nrounds, max_depth, eta, gamma, colsample_bytree, subsample and min_child_weight parameters were tuned for xgboost classifier of xgboost package. Those parameters are:

Nrounds ➔ Max number of boosting iterations.

Max_deth ➔ maximum depth of a tree

Eta ➔ Controls the learning rate. Used to prevent overfitting by making the boosting process more conservative. Lower value for eta implies larger value for nrounds: low eta value means model more robust to overfitting but slower to compute.

Gamma ➔ Minimum loss reduction required to make a further partition on a leaf node.

Colsample_bytree ➔ Subsample ratio of columns when constructing each tree.

Subsample ➔ Subsample ratio of the training instance. Setting it to 0.5 means that xgboost randomly collected half of the data instances to grow trees and this will prevent overfitting.

Min_child_weight ➔ minimum sum of instance weight (hessian) needed in a child.

For nrounds parameter 400; for max_depth parameter 4, 6 and 8; for eta parameter 0.05 and 0.1; for gamma parameter 0.01; for colsample_bytree and subsample parameters 0.5, 0.7 and 0.9; for min_child_weight parameter 1 were tried as possible parameters. So, in total, number of parameter combinations tried is 54 and with repeated cross validation 2700 models were built. As a performance evaluation metric, ROC was used. Best parameter combination found as nrounds = 400, max_depth = 4, eta = 0.05, gamma = 0.01, colsample_bytree = 0.9, min_child_weight = 1, subsample = 0.9. Value of the ROC is 0.858.

GBM

To tune the related parameters and evaluate the performance of the model caret package was used. 10 fold cross validation with 5 repetitions were used in the model building and parameter tuning process.

Interaction.depth, n.trees, shrinkage and n.minobsinnode were tuned for gbm classifier of gbm package. Those parameters are:

Interaction.depth ➔ Integer specifying the maximum depth of each tree.

N.trees ➔ Integer specifying the total number of trees to fit.

Shrinkage ➔ Shrinkage parameter applied to each tree in the expansion. Also known as the learning rate.

N.minobsinnode ➔ Integer specifying the minimum number of observations in the terminal nodes of the trees.

For interaction.depth parameter 3, 4, 6 and 8; for n.trees parameter 500; for shrinkage 0.02, 0.04, 0.06, 0.08, 0.1; for n.minobsinnode parameter 1 were tried as possible parameters. So, in total, number of parameter combinations tried is 4 and with repeated cross validation 200 models were built. As a performance evaluation metric, ROC was used. Best parameter combination found as n.trees = 500, interaction.depth = 8, shrinkage = 0.02 and n.minobsinnode = 1. Value of the ROC is 0.859.

Approach for Python Part

We have developed 2 different models (LightGBM) in python for the binary classification mentioned above. In these models, the best parameters were found by parameter searching with Optuna. The effects of these parameters on all data were observed with the cross-validation technique.

Optuna

Optuna is an automatic hyperparameter optimization software framework, particularly designed for machine learning. In the two models we used in the project; the parameter search part was made with Optuna. Since Optuna was successful in finding local values, fine tune parameter search was applied after finding the parameters. For example, an attempt was made to develop a model by using Optuna again to determine the number of leaves after the determined maximum depth of trees.

SHAP

SHAP (SHapley Additive exPlanations) is a game theoretic approach to explain the output of any machine learning model. It connects optimal credit allocation with local explanations using the classic Shapley values from game theory and their related extensions. We used the SHAP library to find and visualize feature importance values.

LightGBM

LightGBM is a gradient boosting framework that uses tree-based learning algorithm. LightGBM grows trees vertically, while other algorithms grow trees horizontally; this grows towards the tree leaf of the LightGBM while the other algorithm grows in level. It will select the leaf with the maximum delta loss for its growth. A leaf-wise algorithm can reduce loss more than a level-based algorithm while growing the same leaf (Ke et al., 2017). One of the advantages of using LightGBM is that it can handle the imbalance data with “is_unbalance” parameter and “auc” metric used for maximize auc score of models. "num_iterations", "max_depth", "num_leaves", "subsample", "learning_rate" parameters were used for tuning the model.

Multiple Instance Learning

Wikipedia definition: In machine learning, multiple-instance learning (MIL) is a variation on supervised learning. Instead of receiving a set of instances which are individually labeled, the learner receives a set of labeled bags, each containing many instances. In the simple case of multiple instance binary classification, a bag may be labeled negative if all the instances in it are negative. On the other hand, a bag is labeled positive if there is at least one instance in it which is positive. From a collection of labeled bags, the learner tries to either (i) induce a concept that will label individual instances correctly or (ii) learn how to label bags without inducing the concept (Baydoğan, 2021, Homework 4).

Overall Idea Behind of Approach for Python

We used optuna for parameter search. Although optuna cannot find global maximum or global minimum values, it is a fast algorithm for finding local values. We used cross validation during parameter search because we wanted the model to benefit from every data since the number of data was small. We used many parameters in parameter search except boost type("gbdt"), evaluation metric("auc") and unbalance ("is_unbalance") data type. We chose the parameters that give the average best result out of 10 k folds. As a result of 25 different parameter combinations, we created our model with the parameters that gave the best results.

After finding the best parameters, we divided the data into 2 as 85% train and 15% validation data. We trained the model with train data and selected parameters then checked the score with validation data. Finally, we predicted test data with the model we created. After estimating the test data in the final stage, we performed different operations for the two models. Since the first model made a prediction for all the test data, the user's prediction was created by averaging the values of the users with the same id in the test data. However, in the second model that Multiple Instance Learning, since the values of the test data were reduced to a single line as described in the data processing stage, the value resulting from the estimation was used directly.

Performance Evaluation

For R part

Balanced error rate (BER) and area under the receiver operating characteristic curve (AUC) are tried to be maximized as a performance evaluation. Whereas BER shows the average of the error calculated on each class, AUC is a calculation of two-dimensional area under the receiver

operating characteristic curve, that represents true positive rates to false positive rates at different classification thresholds.

4. RESULTS

Results for R Part

The selected models stated previously in Approach section of this project were used on training data and their results were used for making gender prediction on test data. Several tuning parameters were utilized for each approach such as tuning grids stating in "trainControl" function.

At the beginning of the project, large range of tuning parameters were employed especially for random forest. Parameter scale has shrunk as needed for the rest trials to keep up efficiency and computational performance (for both time and also rate). ROC results calculated by the training set were taken into consideration as the best tuning parameters when we were evaluating the model and parameter in terms of possible highest gender prediction. Moreover, sensitivity and specificity measures were also paid regard to make proper assumptions. The serial of the trial models in R part, parameters, results on the training set and gender prediction scores of each tried model for test data were given below in tables together with the reasons.

Table 4.1. Trial parameters and results for DT, RF and SGB methods with different sampling

#	Model	Sampling method	cp	min.buc	mtry	Tree number	Int. Depth	Shrinkage	Min. obs. node	ROC	SENS	Spec	Performans
1	DT	weighted	0.0005	10	-	-	-	-	-	0.80832	0.42527	0.88328	0.70805982
2	RF	weighted	-	-	7	-	-	-	-	0.84975	0.68603	0.8293	0.82437482
3	RF	up	-	-	7	-	-	-	-	0.84975	0.68603	0.8293	0.82437482
4	RF	smote	-	-	7	-	-	-	-	0.85062	0.68787	0.82903	0.82033959
5	RF	down	-	-	7	-	-	-	-	0.84882	0.68261	0.8287	0.81940715
6	RF	rose	-	-	3	-	-	-	-	0.76318	0.96246	0.3446	0.72194515
7	SGB	weighted	-	-	-	800	7	0.01	5	0.8558	0.46126	0.91492	0.7759875
8	SGB	up	-	-	-	800	7	0.01	7	0.85702	0.83434	0.73552	0.7355241
9	SGB	up	-	-	-	800	9	0.01	9	0.85721	0.82919	0.74156	0.84037191
10	SGB	up	-	-	-	800	9	0.01	11	0.85752	0.82815	0.74292	0.84090828
11	SGB	up	-	-	-	800	9	0.01	13	0.85712	0.83125	0.74145	0.83652849
12	SGB	up	-	-	-	800	13	0.01	15	0.85744	0.82001	0.75026	0.83917306
13	SGB	up	-	-	-	800	13	0.01	17	0.8573	0.82073	0.75096	0.83888534
14	SGB	down	-	-	-	400	13	0.01	13	0.85592	0.84486	0.72362	0.8374183
15	SGB	down	-	-	-	600	9	0.01	9	0.8558	0.84352	0.72813	0.83832232
16	SGB	down	-	-	-	600	9	0.01	7	0.8555	0.84208	0.72471	0.83921746
17	SGB	smote	-	-	-	800	9	0.01	9	0.85673	0.75739	0.80038	0.83417874
18	SGB	smote	-	-	-	800	9	0.01	11	0.85699	0.75532	0.80326	0.83280939
19	SGB	smote	-	-	-	800	11	0.01	13	0.85703	0.75316	0.80489	0.83387504
20	SGB	out-rose	-	-	-	800	9	0.01	9	0.99064	0.95785	0.95792	0.73684641
21	SGB	out-rose	-	-	-	800	9	0.01	11	0.99072	0.95814	0.95799	0.7350206
22	SGB	out-over	-	-	-	900	11	0.01	9	0.88811	0.87937	0.752	0.84273231
23	SGB	out-under	-	-	-	700	11	0.01	9	0.86458	0.84838	0.75163	0.83775931
24	SGB	out-both	-	-	-	900	11	0.01	9	0.90974	0.88581	0.78839	0.83305449
25	SGB	out-over	-	-	-	900	11	0.05	9	0.91992	0.90965	0.79402	0.82869956

We determined to calculate class weights and apply some resampling techniques as given in the Approach section, due to the fact that this is an imbalance binary classification problem. These resampling techniques are respectively up, down, smote and rose resampling methods. Furthermore, out-resampling techniques were also done on the data set for SGB method. These out-resampling methods were applied just for SGB method because, the best results with just resampling techniques were obtained by SGB models.

It can be seen on the Table 4.1, the up and down sampling techniques gave a noticeable enhancement in gender prediction scores. In addition to this, in line with that of specificity values, significant increment in sensitivity scores of the model using train data was also obtained. In terms of performance, down sampling method can be seen as a good candidate, but it resulted in the decreasing in specificity values. This can also be seen on the performance scores. Since project data has a huge class imbalance situation, down sampling reduces the sample size severely which could be the reason for these drops.

If we briefly look at random forest results, it can be seen that up sampling is also better choice for random forest in terms of prediction performance. After up sampling, smote, down and rose sampling gave good results for random forest approach. However, the scores of random forest method did not supply a significant improvement as well as SGB method.

In order to go beyond the sampling techniques, we adopted out-resampling techniques on SGB method as aforementioned. The logic here is to create synthetic instances to get rid of the class imbalance situation. Hence, the training process is then run on the new generated synthetic data sets. Even if the logic is same in both resampling techniques, the impact on prediction scores can be change.

It can be seen from Table 4.1, out-over (or up) resampling technique gave the best results in terms of prediction performance for SGB method. Respectively, out-over sampling was followed by out-under (or down), out-both (over+under), and out-rose resampling techniques. It proved also to us that over sampling method work well with minority class problem as in our project.

Apart from the performance, relative influence of each feature for SGM method with out-over sampling technique (int.depth=11, min.obs.=9, shrinkage=0.01, tree num.=900) is as in Table 4.2. and Figure 4.1.

Table 4.2. Relative influence values of each feature for SGM method with out-over sampling

Variable	Relative Influence
num_of_gender_woman	24.615134425
most_visited_gender_woman	21.504394046
num_of_gender_man	13.474813312
sell_price_ave	6.995689061
sell_price_total	5.638708612
num_of_login_workhour	3.175769425
num_of_fav	3.133533412
num_of_visit	2.695812777
num_of_search	2.271353861
num_of_gender_unisex	2.232479573
num_of_login_otherhour	2.085753284
most_visited_gender_man	1.932633119
num_of_login	1.821453618
visited_unique_brand_count	1.536151336
num_of_ord	1.508430620
visited_unique_cat1_count	1.297872404
visited_unique_cat2_count	1.293681240
visited_unique_cat3_count	0.997207604
num_of_basket	0.970084863
most_visited_gender_unisex	0.631740334
is_workhour_log_more	0.182559540
is_more_than_avg_selling	0.004743532
num_of_login_weekday	0.000000000
num_of_login_weekend	0.000000000
is_weekday_log_more	0.000000000

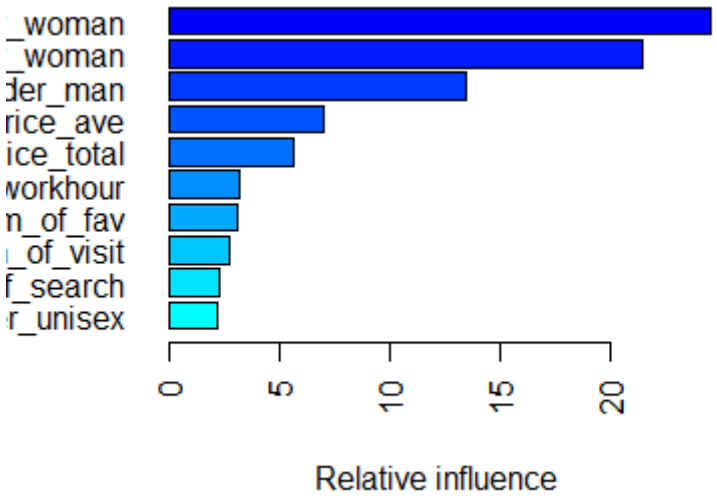

Figure 4.1. Relative influences of first ten features for SGM method with out-over sampling

Results for Python Part

The model was trained to give the best AUC score of the validation data and estimates of the test data were generated.

Here are the results we created with the first model:

- AUC Score: 0.8855605286
- BER Score: 0.8035485934
- Performance: 0.844554561

Here are the results we created with the second model:

- AUC Score: 0. 879092071611253
- BER Score: 0. 807289002557545
- Performance: 0.843190537084399

Feature importance was obtained using Shap as in Figure 4.2.

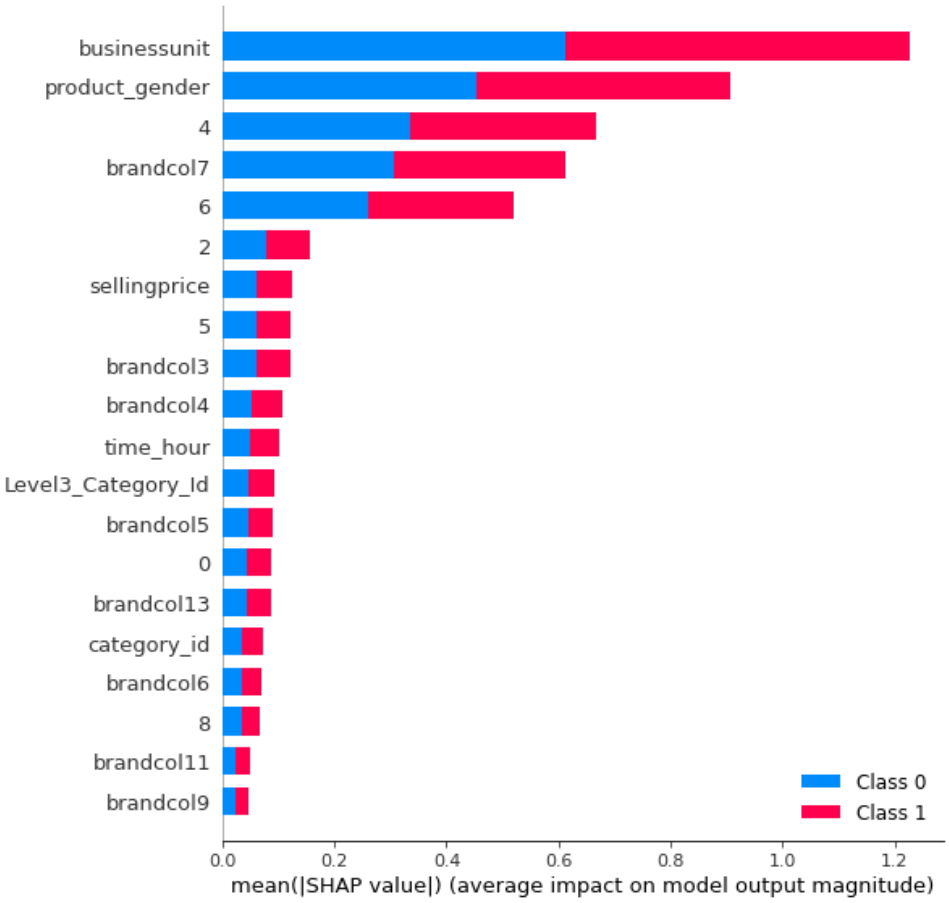

Figure 4.2. Feature importance for the second model

Finally, we created a new result set by averaging the results produced from these two models, and we got the highest score with this result.

Here are the results combination of models:

- AUC Score: 0.882541204887752
- BER Score: 0.817114236999148
- Performance: 0.84982772094345

According to our previous experience and the tests we have done on the results, the results created with combinations of different method give better results.

5. CONCLUSION AND FUTURE WORK

Conclusion and Future Work for R Part

We built a model for class imbalance problem of the project to get the best possible gender prediction result. Prediction results was based on balanced error rate and area under the ROC curve. We applied a lot of sort-count step on train and test data and created new data tables for both. We had two classes: “Kadın” and “Erkek”. There was a huge imbalance within these two classes even after feature engineering steps.

Besides classification models such as decision tree, random forest and stochastic gradient boosting methods, we adopted several resampling techniques to overcome class imbalance problem. These split into two as resampling and out-resampling techniques (up, down, smote and rose for resampling; out-over, out-under, out-both and out-rose for out-resampling). Better prediction scores were obtained by SGB method with out-over resampling. This showed that over sampling method work well with minority class problem as in our project. Moreover, it can be seen from the results, different weighted or resampled data set is the significant manner to solve the class imbalance problem. In literature, rose and smote sampling have a lot of superiority according to over (or up) and under (or down) sampling methods, but we thought that our data set needed different pre-processing step before them, so rose and smote sampling could not give better gender prediction score than over and under sampling techniques.

As for the parameter tuning step for models, various parameter and large grids were used for this project (especially for SGB method). We realized when we select the number of trees under (or upper) 400 (or 900), the gender prediction performance is getting worse, thus we limited the number of trees value within 400 to 900 range. For learning rate (or shrinkage), we used generally 0.01 and 0.05 values, but 0.01 was more effective choice for the models in terms of gender prediction performance score. For the complexity of the tree (interaction depth) and the minimum number of training set samples in a node to commence splitting (min.obs.node), 99 and 11 values provided better scores for gender prediction. Apart from the best prediction parameters, it was observed that another best prediction can be obtained by a new model having parameters totally different than that of the best model. Hence, a variety of parameters should be applied to a new approach without depending on the tuning parameters of the best result.

Conclusion and Future Work for Python Part

We created a modeling method to produce the best auc score in our classification estimates. Our evaluation metrics are given as "auc" within the two models.

Here is the example of the confusion matrix built on validation data:

array ([[430, 122], [53, 238]], dtype=int64)

Despite the high error rate in the results, we conclude that we achieved a good result both in terms of “auc” score and other metrics, considering the small number of data given and the difficulty of gender estimation.

Since we create the results from the results of two different models, there are future works on the python side where we can change both the models and the results. First of all, a model could be created with different gradient boosting algorithms and its results could be combined with the results we obtained. Additionally, we could create a custom evaluation metric ourselves. This metric could help us increase the average performance by learning from the "auc" and "bar" values, which are the evaluation method. Finally, we could improve the performance of the model by creating a model that predicts the empty values instead of giving them 0. Filling the NaN values first, and then training the model with the resulting data could increase the performance of model because model would learn better with these artificial values.

6. CODE

You can access the codes with these links:

For R Part:

1)https://github.com/BU-IE-582/fall21-ozdemirpinar/blob/gh-pages/Project-Group10/ProjectCodes_Group10_PinarOzdemir.R

2) https://github.com/BU-IE-582/fall21-helvaciburakcan/tree/gh-pages/Project

For Python Part:

1) https://github.com/BU-IE-582/fall21-cumhurkilic/tree/gh-pages/Project_code_files

7. REFERENCES

"Analytics Vidhya-1", (2016), Practical Guide to deal with Imbalanced Classification Problems in R. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2016/03/practical-guide-deal-imbalanced-classification-problems/ (Accessed: 15 January 2022)

"Analytics Vidhya-2", (2020), How to Improve Class Imbalance using Class Weights in Machine Learning, Analytics Vidhya,

https://www.analyticsvidhya.com/blog/2020/10/improve-class-imbalance-class-weights/ (Accessed: 17 January 2022).

Barandela, R., Valdovinos, R. M., Sánchez, J. S., and et al., (2004). The Imbalanced Training Sample Problem: Under or over Sampling? Structural, Syntactic, and Statistical Pattern Recognition, 806–814.

Baydoğan, M. G. (2021). IE 582 Statistical Learning for Data Mining, Homework 4.

Branco, P., Torgo, L., and Ribeiro, R. P., (2016), A survey of predictive modeling on imbalanced domains, ACM Computing Surveys, 49(2), 1-50.

Breiman, L., Friedman, J. H., Olshen, R., and et al., (1984), Classification and regression decision trees, Monterey, CA: Wadsworth & Brooks.

Breiman, L., (2001), Random Forests, Machine Learning, 45 (1), Springer: 5–32.

Brownlee, J., (2019), A Gentle Introduction to Imbalanced Classification, Machine Learning Mastery. https://machinelearningmastery.com/what-is-imbalanced-classification/ (Accessed: 20 January 2022)

Chawla, N. V., Bowyer, K. W., Hall, L. O., and et al., W. P., (2002), Smote: Synthetic minority over-sampling technique, Journal of Artificial Intelligence Research, 16, 321-357.

Friedman, J. H., (2001), Greedy Function Approximation: A Gradient Boosting Machine, Annals of Statistics, JSTOR, 1189–1232.

Ke, Guolin; Meng, Qi; Finley, Thomas; Wang, Taifeng; Chen, Wei; Ma, Weidong; Ye, Qiwei; Liu; Ti-Yan. (2017). LightGBM: A highly efficient gradient boosting decision tree. *Conference on Neural Information Processing System*s, NIPS. CA: USA.

Kuhn, M., Wing, J., Weston, S., and et al., (2018), Classification and Regression Training, R package version 6, 0–81.

Lunardon, N., Menardi, G., and Torelli, N., (2014), ROSE: a Package for Binary Imbalanced Learning, R Jorunal, 6, 82-92.

Martin, D. P., (2016), Handling Class Imbalance with R and Caret - An Introduction: Wicked Good Data. https://dpmartin42.github.io/posts/r/imbalanced-classes-part-1 (Accessed: 23 January 2022).

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

Torgo, L., (2010), Data Mining using R: learning with case studies, CRC Press, ISBN: 9781439810187.