IE582 - Fall 2021 Project

Group - 8

Çiğdem Renkli - 2020802021

Özlem Akekmekçi - 2020749000

1. Introduction

1.1. Problem Description

In this report, we will propose a classification model for a gender prediction of the website customers based on their e-commerce clickstream data. We will work on a dataset that consists of actions of the users on an e-commerce website. By considering it as a binary classification problem for simplicity and due to the structure of the training data, we applied various proper algorithms and evaluated them to improve the performance of the classification on the test data. It is an unstructured dataset; so, we need to extract features first. We try to extract the most distinguishing features and try to build a stable and well-performing model for gender prediction.

In this report, after introducing the problem and the summary of the approach and the dataset, we will discuss the related literature by focusing on the existing studies on gender prediction that consider learning methods that we have also evaluated during the submission period. In the approach part, we will briefly discuss all approaches we have used including preprocessing, feature extraction and transformation, validation, train data alternatives, parameter tuning, and model selection. We will also give information about the feature engineering alternatives. Then, by comparing the performances of different feature engineering alternatives and alternative approaches, we will propose the details of our final submission. In the conclusion part, we will summarize the findings and suggest some alternatives to having a better approach. Then, we will provide our codes and Github links for our codes.

1.2. Summary of the Proposed Approach

Our aim is to produce the best predictions for gender prediction in terms of performance and stability. For this purpose, we follow the steps below:

- Data Preprocessing: We apply some preprocessing steps to the data.
- Feature Engineering: We try 4 alternatives for feature engineering.
  Alternative 1: We extract features trying to use every information in the data. We code a total of 1539 features; their explanations are presented in the file named "Extracted Features and Explanations.csv" uploaded to Github. Also, we mention the details in the Approach part of this document
  Alternative 2: k-means two class-encoding studied in [1] because it is mentioned that BoW representations are successful and this is an extended version to them. We use k = 50.
  Alternative 3: Random tree (RT) encoding with terminal node representation is explained in [1] because it is mentioned as robust, fast, and producing competitive results.
  Alternative 4: We do not extract any features and try training the model on the raw data. Then we produce predictions on the raw test data and take the average over actions of a customer to calculate the probabilities.
- Train Data Selection: We notice that we have imbalanced data. In order to prevent the effects of imbalance, we try undersampling and oversampling. Also, we get results on training without any special sampling.

In addition to sampling based on target ratio, we try sampling data according to customers' frequency of actions. We split the data into two: customers having more than 120 (above the median, below the 3rd quartile) actions are called experienced customers and the others are called cold starters. We try to train models with just experienced samples and just cold-started samples.

- Predictive Modelling: We try several algorithms to train our models: Random Forests, Boosting Tree Algorithms (LightGBM and XGBoost), Logistic Regression, K Nearest Neighbors (KNN). We tune their parameters checking the performance metrics on separate validation data and the test data provided.
- Model selection: Based on the performance of the test data we decide on the best models and try their ensembled results. We cannot see the performance on the full test data but performance on a part of it is provided through our submissions.

1.3. Descriptive Analysis

- Train data has 5,493,268 rows and 19 columns. Test data has 2,324,814 rows and 19 columns. However, while working on the project we noticed that there are many rows having exactly the same values for all of the columns. We think that this may be caused by a data quality problem, so we drop the duplicate rows from both data sets. After dropping the duplicate rows, train data has 2,077,356 rows and test data has 877,989 rows.
- We have unique_id as the id of the users; gender as the class information and type as the type of data. Other columns except these three can be used for extracting information.
- We check distinct value counts of features and their null ratios in both train and test data; presented in the table below.

	Distinct value counts		**Null ratios %**	
Column Name	**train**	**test**	**train**	**test**
time_stamp	1,583,129	770,634	-	-
contentid	482,796	256,994	0.00	0.00
user_action	5	5	-	-
sellingprice	35,433	24,356	1.54	1.62
product_name	441,470	239,085	0.11	0.10
brand_id	33,328	23,941	0.11	0.10
brand_name	33,332	23,943	0.11	0.10
businessunit	83	83	0.11	0.10
product_gender	3	3	11.29	11.99
category_id	2,240	2,022	0.11	0.10
Level1_Category_Id	10	10	0.11	0.10
Level1_Category_Name	10	10	0.11	0.10
Level2_Category_Id	95	94	0.11	0.10
Level2_Category_Name	95	94	0.11	0.10
Level3_Category_Id	739	709	0.11	0.10
Level3_Category_Name	739	709	0.11	0.10
gender	2	-	-	100.00
unique_id	5,618	2,380	-	-
type	1	1	-	-

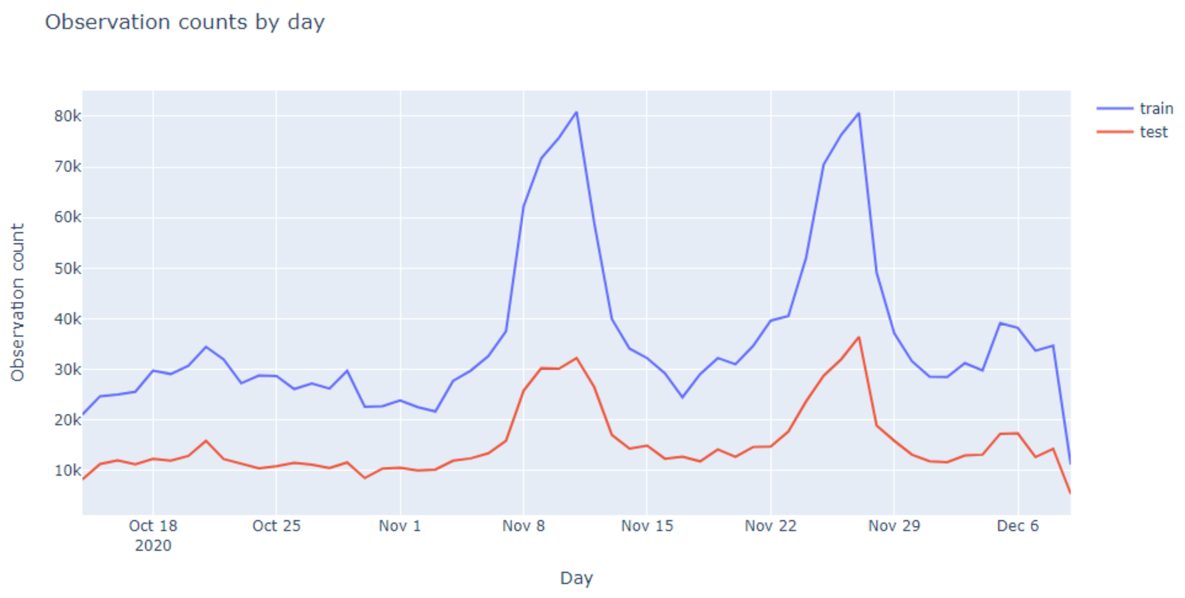
Columns named content_id, product_name, brand_id and brand_name have too many distinct values although they are not continuous variables. We should either remove these features or reduce the number of distinct values before use.
Except for the label column "gender", features have similar null ratios in train and test data, which is desirable.

- We check if there are any intersections of train and test data in terms of unique_id and see there are no common customers of two data sets.
- We check types of the columns, examine the content of each of them and produce the table below:

Column Name	**Data Type**	**Comments**
time_stamp	object	Timestamp
contentid	float64	Seems as float but it has no numerical meaning, it is categorical
user_action	object	Categorical
sellingprice	float64	Continuous
product_name	object	Categorical
brand_id	float64	Seems as float but it has no numerical meaning, it is categorical
brand_name	object	Categorical
businessunit	object	Categorical
product_gender	object	Categorical
category_id	float64	Seems as float but it has no numerical meaning, it is categorical
Level1_Category_Id	float64	Seems as float but it has no numerical meaning, it is categorical
Level1_Category_Name	object	Categorical
Level2_Category_Id	float64	Seems as float but it has no numerical meaning, it is categorical
Level2_Category_Name	object	Categorical
Level3_Category_Id	float64	Seems as float but it has no numerical meaning, it is categorical
Level3_Category_Name	object	Categorical
gender	object	Target
unique_id	int64	Id
type	object	Rejected, just shows the type of the data

- Then we check the gender ratio but first we create a new binary column: 1 if female, 0 if male, and name it as female_label. We will use it as target variable when training models. We have 5,618 customers in the train data and 3,679 of them are females. So, the female ratio is 65.5% indicating that this is an imbalanced data set. When we calculate the female ratio of raw action data, we see that most of the actions are taken by females, its ratio is 83.7%.

- We check the daily action counts and see the graph below:



Both train and test data include observations from October 14th to December 9th, 2020. The periods between November 7-13 and November 23-29 are probably campaign periods as we see peaks.

We calculate new columns based on timestamp:

- Weekend_flag: 1 if weekend; 0 otherwise
- Campaign_flag: 1 if campaign period; 0 otherwise
- Day_name: name of the day (Monday, Tuesday, …)
- Hour: the hour of the action
- 4_hour_interval: An ordinal column indicating that in which 4-hour interval of the day the action takes place
- 8_hour_interval: An ordinal column indicating that in which 8-hour interval of the day the action takes place

- Then, we examine columns one-by-one and take some actions on them
  - Contentid: It seems that every contentid has a distinct product_name. We check if this is true for all ids. We see that a contentid does not have more than 1 distinct product_names. But there are some zeros, which means product_name is null. There are 632 contentids having product_name null out of 482,796 distinct contentid. So, they are very few. Contentid can be eliminated.
  - User_action: We check the female ratio for values of this column.

user_action	gender	percentage
basket	F	82.05%
	M	17.95%
favorite	F	92.06%
	M	7.94%
order	F	78.65%
	M	21.35%
search	F	80.64%
	M	19.36%
visit	F	85.04%

	M	14.96%

Male ratios of basket, order and search are very close to each other. The male ratio of visit is slightly lower than them. The male ratio of favorite is very lower than the male ratio of other categories. So, having a feature indicating whether user_action is a favorite or not can be discriminative.

- Sellingprice: We check the distribution and see there are zeros in the column. There are no cases having the action is "order" and zero price. Yet, having zero price is not normal. There are 72,790 such records, which is approximately %3.5 of train data. Then, we check if the distribution of sellingprice changes by gender:

gender	count	mean	std	min	1%	10%	25%	50%	75%	90%	99%	max
F	1712300	209.97	928.22	0	0	24.9	50	87.49	170	323.5	2668	226320
M	333043	360.19	1684.16	0	0	24.9	59.6	120	249.9	500	5299	226320

Mean and median of sellingprice is higher when the customer is male. This feature can help to discriminate.

- product_name: It is hard to process this free-text information. We may use it by counting distinct values.
- brand_id and brand_name: These two features have very similar distinct value counts, most probably they have similar information. We check and see that other than 4 brand_ids, every brand_id has a distinct brand_name. Those 4 have 2 different brand_names but it seems that those names indicate the same brands. We can drop brand_id column and just use brand_name column. There are too many distinct values, we have to reduce them. We may use most-frequent values and convert the remaining ones to 'other' category. Values corresponding to %40 of the train data are taken as most-frequent ones.
- Businessunit: We may use most-frequent values and convert the remaining ones to 'other' category. Values corresponding to %75 of the train data are taken as most-frequent ones.
- product_gender: We check female ratio for values of this column.

product_gender	gender	Percentage
Erkek	F	47.0%
	M	53.0%
Kadın	F	94.5%
	M	5.5%
Unisex	F	81.8%
	M	18.2%

- category_id: We may use most-frequent values and convert the remaining ones to 'other' category. Values corresponding to %30 of the train data are taken as most-frequent ones.
- Level1_Category_Id and Level1_Category_Name: They have the same information we can drop Level1_Category_Id and just use Level1_Category_Name.
- Level2_Category_Id and Level2_Category_Name: They have the same information; we can drop Level2_Category_Id and just use

Level2_Category_Name. We may use most-frequent values and convert the remaining ones to 'other' category. Values corresponding to %75 of the train data are taken as most-frequent ones.

- Level3_Category_Id and Level3_Category_Name: They have the same information we can drop Level3_Category_Id and just use Level3_Category_Name. We may use most-frequent values and convert the remaining ones to 'other' category. Values corresponding to %75 of the train data are taken as most-frequent ones

2. Related Literature

Prediction of the website audience demographics and customer gender prediction based on e-commerce clickstream data is a commonly discussed problem in the literature.
Individual and behavioral targeting of customers and personalized advertisement is constructed based on the idea of increasing advertisement effectiveness and average click-through rates(CTR) by choosing the best-matching advertisement for each user or user group. [1] The selection of best-matching advertisements is done by applying data mining techniques to the collected user data. Also, since some of the customers are not likely to provide their personal information due to various reasons including privacy, it is important to predict such customers' characteristics based on their clickstream data. Although there is no mutually agreed one exact technique that works best for the demographic estimation of the customers and the performance of the techniques also varies according to each data structure, there are still groups of techniques and variables that are found as significant in most of the papers.

Random Forest, Support Vector Machine(SVM) and Bayesian Networks are popular learning methods in this area. [2] First two of them are the methods that we have discussed in the class. Random Forests are trying to predict a gender by using decision trees on different samples and choosing the class with the majority vote, SVM classifies gender by using the hyperplanes with the maximum margin in the vector space of the objects and works well in high dimensional spaces due to Kernel method. Also, Neural Networks and Gradient Boosting Decision Trees are also other classification algorithms used to predict the genders of online customers.[1][3][4] Classification performance of the Random Forest for both binary and multi-class demographic outcome classification problems is found to be much superior to the other proposed techniques and it is suggested that Random Forests are better in dealing with large feature sets since it uses classification trees as base classifiers and inherent random feature selection. [1][2] In [1] which finds the best performance with Random Forests, in Random Forest classification, they use bagging to replace standard decision trees with randomized Classification and Regression Decision Trees (CART) which builds decision trees based on Gini's impurity index for splitting and performs random feature selection at each tree node. As in the bagging strategy in [1], it is also found that to combine multiple algorithms as an ensemble learning methods such as GBM, XGBoost, and Random Forest are performing better in predicting gender than other algorithms in terms of accuracy. [5] Although ensemble methods are found superior in the existing limited studies, there are not many studies on the use of ensemble methods in gender prediction, especially for the GBM and XGBoost methods.

In this project, adding to various methods we learned in the class, we also performed lightGBM and XGBoost since it improved the performance in our study and proposed superiority of these methods suggested in the literature. LightGBM [6] is a boosted tree algorithm developed by Microsoft and it generally produces satisfying results for classification problems. While other methods develop trees horizontally and levels-by-level, LightGBM grows in a vertical and leaf-by-leaf way. It selects the leaf by measuring the delta loss and chooses the leaf with the highest loss.

The advantages of this method are to reduce loss more when compared to other tree-based learners, handle high amounts of data with the usage of less memory, places premium on precision, and enable GPU learning. [8] Since the method is not recommended in small datasets and performs better when there exists a large amount of data, it is reasonable to perform this classification method with our large dataset. Extreme Gradient Boosting (XGBoost) [7] is another boosting tree algorithm producing well-performing results and "designed for speed and performance". [8] It implements algorithms under the Gradient Boosting and provides more accurate classifications. The most important reason for the success of XGBoost is scalability. It works much faster than other algorithms and scales to billions of data in memory-limited settings. It uses a novel algorithm to handle sparse data and parallel tree boosting, so parallel and distributed computing make learning faster. [9]

Add to proposed classification algorithms, using supporting resampling techniques to resolve imbalance problems or generating new features to capture the user variation are also important parts of the preprocessing to improve the accuracy of the selected classifier. [1][2] As resampling techniques, oversampling and undersampling are two different alternatives that make the imbalanced date more balanced. In addition to data-level resampling methods, there exists another alternative algorithm-level cost-sensitive learning method that also considers the misclassification cost and reweights misclassifications of the instances. [2] However, since misclassification costs for all instances are the same in our proposed problem, we will only consider resampling methods and choose one of them after comparing their performances.

In the feature generation part, we consider the important variables in the existing literature. In the study [4] that makes gender prediction of online customers by using Artificial Neural Networks, the important variables are determined as "average time spent before making a categorized click, total time spent on the site, average time spent between clicks, clicked item, order of the clicked item, day, existence of a discounted item in the basket." Another study [2] on gender prediction uses features on the day, month, year, starting and ending hours, duration of visit, the number of products, and average time per product. Similarly, [1] also considers the time dimension that includes hours and days of the visiting pattern, frequency dimension, and intensity dimension that considers the time spent and the number of page requests within the website.

3. Approach

We perform the steps below and build different models using different variations of these steps. Our main aim is to find the best-performing predictions on the part of the test data provided to us through our submissions. In this part of the document, we list and explain all approaches that we use but first, we should give details of our feature engineering alternatives.

Alternative 1: We extract features trying to use every information in the data. We code a total of 1539 features including:

- We calculate statistics (count,sum,mean,median,minimum, maximum) of sellingprice for different values of categorical columns (user_action,brand_name,businessunit,product_gender,category_id,Level1_Category_Name, Level2_Category_Name,Level3_Category_Name,weekend_flag,campaign_flag,day_name,hour,4_hour_interval,8_hour_interval). For example, we calculate the sum of sellingprice with user_action is "order" for each of the unique_ids.
- We calculate seconds between consecutive actions (mean, median, maximum and minimum) and distinct values of categorical columns for different aggregations of time (overall, daily, weekly,hourly). For example, the average of the seconds between consecutive actions in a day and the number of different user_action values are calculated for each unique_id. Also, we

calculated these features for each type of user_action; for example, the average of the seconds between consecutive *orders* in a day is calculated.

- How many operations are there before ordering a product on average (in 1-hour, in 3 hours, in 6 hours, in 24 hours) are calculated.
- How many visits are there before ordering a product on average (in 1-hour, in 3 hours, in 6 hours, in 24 hours) are calculated.
- How many searches are there before ordering a product on average (in 1-hour, in 3 hours, in 6 hours, in 24 hours) are calculated.
- How many favorites are there before ordering a product on average (in 1-hour, in 3 hours, in 6 hours, in 24 hours) are calculated.
- Time between basket and order (in seconds) on average is calculated.
- Time between two consecutive orders (in seconds) on average is calculated.

<u>Alternative 2</u>: First we apply binary encoding for our categorical features. Then, we use imputation with mean values and use min-max scaling to scale our features. Finally, we apply k-means two class-encoding studied in [1] with k = 50.
<u>Alternative 3</u>: First we apply binary encoding for our categorical features and then Random tree (RT) encoding with terminal node representation explained in [10]. We use two alternatives here: 10 random trees with depth 20 and 20 random trees with depth 10.
<u>Alternative 4</u>: First we apply binary encoding for our categorical features and then we do not extract any features and try training the model on the raw data.

We have the following approaches:

- Preprocessing: We use simple imputation with mean values when imputation is needed. We use min-max scaling when it is needed; for example, before training logistic regression we use both imputation and scaling but we do not use these preprocessing steps before using the LightGBM algorithm because it can handle null values and does not require scaling.
- Feature extraction: We have 4 alternatives as mentioned.
- Train-validation split: We randomly split the train data into train and validation using stratified sampling. Target variable and experienced flag (1 if the customer has more than 120 actions; 0 otherwise) as strata variables. When tuning the hyperparameters of algorithms we check validation performance.
- Train data: We have 3 alternatives in terms of sampling on the target variable:
    - No sampling
    - Undersampling: Minority class is unchanged. A random sample of the majority class is selected so that both classes have the same number of rows.
    - Oversampling using SMOTE: Synthetic observations of minority class are produced so that it can have the same number of observations as the majority class.

  Also, we have 3 alternatives in terms of sampling on customers' action frequency:
    - No sampling
    - Sample of just experienced customers
    - Sample of cold starter customers
- Feature transformation: We have 3 alternatives:
    - No transformation
    - Coarse class transformation: This transformation builds a simple decision tree for each feature and represents a value as target ratio of the leaf that the value belongs to.
- Parameter tuning: For every algorithm, parameters are tuned using the Grid Search approach and checking the performance on validation data.

- Model selection: We use KNN, Logistic Regression, Random Forest, LightGBM and XGBoost algorithms and their ensembles. We covered KNN, Logistic Regression, and Random Forest in our lecture. For the other two algorithms, we have provided information in the literature review part of this report.

4. Results

First, we provide results of first 8 approaches:

		Performance on he Provided Part of the Test Data		
No	**Approach**	**Area Under ROC**	**Balanced Error Rate**	**Overall Performance**
1	Feature engineering alternative 1 LightGBM with tuned parameters	0.8500	0.6681	0.7591
2	Feature engineering alternative 1 Imputation with mean values Min-max scaling KNN with tuned parameters	0.5824	0.5824	0.5824
3	Feature engineering alternative 1 Undersampling LightGBM with tuned parameters	0.8471	0.7788	0.8129
4	Feature engineering alternative 4 LightGBM with tuned parameters Predictions are obtained on raw test data and their mean is calculated	0.8565	0.6543	0.7554
5	Feature engineering alternative 4 Undersampling LightGBM with tuned parameters Predictions are obtained on raw test data and their mean is calculated	0.8598	0.6716	0.7657
6	Feature engineering alternative 2 LightGBM with tuned parameters	0.8357	0.5691	0.7024
7	Feature engineering alternative 3 (10 Random Trees; Depth 20) LightGBM with tuned parameters	0.8393	0.6277	0.7335

8	Feature engineering alternative 3 (20 Random Trees; Depth 10) LightGBM with tuned parameters	0.8423	0.5828	0.7125

Above table shows the performances of our four different feature engineering alternatives with the same algorithm. Here, we see that alternative 1 provides the best performance; so, we continue with it. Also, as we see in Approach 3, KNN performs worse than LightGBM, so we do not use KNN for other trials.

The table below shows performances of other approaches:

		Performance on he Provided Part of the Test Data		
No	Approach	Area Under ROC	Balanced Error Rate	Overall Performa nce
9	Feature engineering alternative 1 Train sample: cold starters LightGBM with tuned parameters	0.8510	0.7771	0.8140
10	Feature engineering alternative 1 Train sample: cold starters XGB with tuned parameters	0.8554	0.7780	0.8167
11	Ensemble of Approach no 10 and 11 Average of predictions	0.8555	0.7742	0.8149
12	Feature engineering alternative 1 Imputation using mean values Train sample: experienced customers & undersampling XGB with tuned parameters	0.8759	0.8041	0.8400
13	Feature engineering alternative 1 Imputation using mean values Train sample: all customers & undersampling XGB with tuned parameters	0.8705	0.7895	0.8300
14	Ensemble of Approach no 12 and 13 Cold starters are predicted by approach 13, experienced are predicted by 12	0.8768	0.8040	0.8404
15	Feature engineering alternative 1 Train sample: all customers LightGBM with tuned parameters	0.8817	0.7738	0.8278

16	Feature engineering alternative 1 Train sample: all customers LightGBM with tuned parameters	0.8803	0.7681	0.8242
17	Feature engineering alternative 1 Train sample: experienced customers & undersampling Imputation using mean values XGB with tuned parameters	0.8797	0.8178	0.8488
18	Feature engineering alternative 1 Train sample: all customers Coarse class transformation Logistic regression with L1 regularization	0.8720	0.7570	0.8145
19	Feature engineering alternative 1 Train sample: all customers Coarse class transformation XGB with tuned parameters	0.8615	0.7580	0.8097
20	Feature engineering alternative 1 Train sample: all customers Coarse class transformation Feature selection: only features with importance >0 are taken. Importance is obtained from the model in Approach 19 Logistic regression with L1 regularization	0.8735	0.7496	0.8116
21	Feature engineering alternative 1 Train sample: all customers Coarse class transformation Feature selection: Spearman correlation between features are calculated. Also, univariate area under ROC values are calculated for each feature. Among features having correlation >= 0.9, the one with highest univariate AUC value is selected Logistic regression with L1 regularization	0.8752	0.7542	0.8147
22	Feature engineering alternative 1 Train sample: all customers Coarse class transformation Feature selection: Spearman correlation between features are calculated. Also, univariate area under ROC values are calculated for each feature. Among features having correlation >= 0.9, the one with highest univariate AUC value is selected XGB with tuned parameters	0.8644	0.7609	0.8126
23	Feature engineering alternative 1 Imputation using mean values Train sample: oversampling XGB with tuned parameters	0.8757	0.7728	0.8243

24	Feature engineering alternative 1 Imputation using mean values Train sample: oversampling & experienced customers XGB with tuned parameters	0.8256	0.6873	0.7564
25	Feature engineering alternative 1 Imputation using mean values Train sample: oversampling XGB with tuned parameters	0.8730	0.7880	0.8305
26	Feature engineering alternative 1 Train sample: all customers Imputation with mean values Feature selection: Spearman correlation between features are calculated. Also, univariate area under ROC values are calculated for each feature. Among features having correlation >= 0.9, the one with highest univariate AUC value is selected XGB with tuned parameters	0.8763	0.7842	0.8303

From the table above, we have some observations:

- Compared approaches 21 and 22, Logistic Regression and XGBoost perform similarly on coarse classed data but the coarse class transformation in general causes to perform worse than no transformation approaches.
- Undersampling generally works best.
- Models trained on experienced customers generally work better than models trained on cold starter customers and all customers.

Please note that the numbers given to approaches in this report and in our group submission sheet are not the same. In our group submission sheet, we have some better-performing submissions than the ones reported here but those submissions have some inconsistent elements so we choose not to report them here.

Our final submission follows the approach below:

- It is an ensemble approach. Average of the predictions obtained from the following two models are used.
- Model 1: Feature engineering alternative 1 is used. No sampling is applied; whole train data is used. LightGBM with tuned parameters (max_depth=2,learning_rate=0.1,n_estimators=200,min_child_samples=20,colsample_bytree=1) is trained.
- Model 2: Feature engineering alternative 1 is used. Imputation with mean values is applied. Undersampled train data is used. XGBoost with tuned parameters (max_depth=50',n_estimators=200,min_child_samples=20,colsample_bytree=0.5) is trained.

5. Conclusions and Future Work

In this report, we have proposed different approaches to the gender prediction problem by using the clickstream data of an e-commerce website. In the final submission, we selected a method by comparing the evaluated performances of different feature engineering alternatives, learning approaches, and resampling methods. We selected an ensemble approach on two different methods in the final submission. Among feature engineering alternatives, we used same best-performing alternative in both two models. It is the alternative that we extract features trying to use every information in the data and it includes 1539 features. Since it may cause an high dimensionality problem, in the future work, it may be reasonable to work with a learning method that works well in high-dimensional spaces in a binary classification setting such as SVM. However, since SVM requires so much time for large datasets, it may be required to work with different implementations of SVM that performs well in both large and high dimensional datasets. Also, using such large number of features may have caused multicollinearity and overfitting problems. So, using different methods for feature selection or different feature enginering alternatives may be reasonable to apply in the further studies. For the resampling method, undersampling performed better in most of the alternatives and we applied undersampling in model 2 and no sampling in model 1. Although we consider the misclassification costs for all instances same in our problem, it may be reasonable to check whether the misclassification cost should differ for different genders. For instance, if one class is more likely to spend more, it may be reasonable to give more weight to the instances of this class. Or similarly, if the personalized advertisements developed based on this classification are more effective in the shopping behavior of one of the classes, it may be required to increase the weight of the instances, so that the cost of misallocation of this more advertisement sensitive class. As learning approaches we used two popular algorithms based on Gradient Boosted Machines: LightGBM and XGBoost, both with tuned parameters. These algorithms are the best performing models among the applied approaches. XGBoost works well with sparse data and there exists some sort of a sparsity in the prices in our dataset. Also since tree based algorithms are likely to perform better in such binary prediction problems and LightGBM is an algorithm based on decision tree methods that performs better in large datasets compared to other tree-based algorithms, it is reasonable to have high performance with this model. It is also expected to have performance improvement when we use the ensemble methods as suggested in the literature.

6. Code

Working codes of final submission:

https://github.com/BU-IE-582/fall21-CigdemRenkli/blob/e8d0f49eebdcb2387917eef8cce80b3d51caaa5b/Project/Final%20Submission.ipynb

Link for all documents:
https://github.com/BU-IE-582/fall21-CigdemRenkli/tree/gh-pages/Project

Final Submission.ipynb: Includes working codes from the very beginning of data analysis to the end of producing final predictions (as requested in the project document)

Models V01.ipynb & Models V02.ipynb: Include models build when grid search is applied for parameter tuning of different algorithms

Submissions - V01.ipynb, Submissions - V02.ipynb & Submissions - V03.ipynb: Includes codes of all our submissions

Coarse Classing & Correlation Check.ipynb: Includes coarse class transformation codes and correlation check of features

7. References

[1] De Bock, Koen W., and Dirk Van den Poel, "Predicting website audience demographics for web advertising targeting using multi-website clickstream data" *in Fundamenta Informaticae,* 98.1, 2010, 49-70.

[2] Duong, Duc, Hanh Tan, and Son Pham, "Customer gender prediction based on E-commerce data" *in 2016 Eighth International Conference on Knowledge and Systems Engineering (KSE)*, IEEE, 2016.

[3] Lu, Siyu, et al., "Genderpredictor: a method to predict gender of customers from e-commerce website" in *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, Vol. 3, IEEE, 2015.

[4] Silahtaroglu, G., "Predicting gender of online customer using artificial neural networks" in *Proceedings of International Conference on Management and Information Technology*, New York, 28th August, 2015.

[5] Al-Zuabi, Ibrahim Mousa, Assef Jafar, and Kadan Aljoumaa, "Predicting customer's gender and age depending on mobile phone data" in *Journal of Big Data 6.1,* 2019, 1-16.

[6] Ke, G., Qi, M., Thomas, F., Taifeng, W., Wei, C., Weidong, M., Qiwei, Y., & Tie-Yan, L., "LightGBM: a highly efficient gradient boosting decision tree", in *Advances in Neural Information Processing Systems*, 2017.

[7] Chen, T., & Guestrin, C., "XGBoost: A Scalable Tree Boosting System" in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[8] Vel, Shanmuga "Gender Prediction And Performance Analysis of Voice Using Ensemble Alghoritms" in *International Research Journal of Modernization in Engineering Technology and Science*, Vol. 3, 2021.

[9] Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system" in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016.

[10] Küçükaşcı, Emel Şeyma,and Mustafa Gökçe Baydoğan, "Bag encoding strategies in multiple instance learning problems" *Information Sciences,* 467, 2018, 559-578.