IE 582

Statical Learning for Data Mining

Fall 2021

Final Report

by

Hatice Pınar YILDIRIM

İrem Gülçin ZIRHLIOĞLU

Course Instructor	: Mustafa Gökçe BAYDOĞAN
Date of Submission	: 24.01.2022
Group Number	: 7
Assistant	: İlayda ÇELENK

Department of Industrial Engineering

Boğaziçi University

Bebek, Istanbul

TABLE OF CONTENTS

1. INTRODUCTION

1.1. Problem Description

The purpose of the project is to build a model by using the described methods in the class to make classification to predict the gender in the provided dataset. In the provided real world dataset, some information about an online retailer is given. It contains some actions of the users on the website. The classification is done to make prediction about the gender.

The dataset is unstructured. The first problem of the project is to make feature extraction to make the regular data matrix before making the classification model. In the raw dataset, there are 18 feature columns. By using the described methods in the class, feature extraction is done.

The measurements of the performance of the classification model are balanced error rate and the area under the ROC curve.

1.2. Descriptive Analysis of the Given Data

As mentioned before, the dataset is a real-world dataset which is taken from the website and is occurred from the customers' actions. The raw dataset is unstructured. The dataset consists of 18 inputs and 1 output. The detailed information about features in the dataset is given in Table 1.

Table 1 Inputs and output and their descriptions

	FEATURE	DESCRIPTION
INPUT	time_stamp	: timestamp of the action
	contentid	: id of the product
	user_action_	: type of the action (i.e. visit, search, basket and etc.)
	sellingprice	: price of the item
	product_name	: name of the product
	brand_id	: brand id of the product
	brand_name	: brand of the product
	businessunit	: business unit information for the product
	product_gender	: gender of the product if defined
	category_id	: category id in which product is belonging to
	Level1_Category_Id	: category id in the first hierarchy
	Level1_Category_Name	: category name in the first hierarchy
	Level2_Category_Id	: category id in the second hierarchy
	Level2_Category_Name	: category name in the second hierarchy
	Level3_Category_Id	: category id in the third hierarchy
	Level3_Category_Name	: category name in the third hierarchy
	unique_id	: id of the user
	type	: type of the data (just for information purposes)
OUTPUT	gender	: class information

In the train set, the amount of train example is 5493268 while it is 2324814. In the dataset, 10 of the input is categorical while the rest is numerical as can be seen at Table 2.

Table 2 Type of the features

FEATURE NAME	TYPE
TIME_STAMP	Categorical
CONTENTID	Numerical
USER_ACTION_	Categorical
SELLINGPRICE	Numerical
PRODUCT_NAME	Categorical
BRAND_ID	Numerical
BRAND_NAME	Categorical
BUSINESSUNIT	Categorical
PRODUCT_GENDER	Categorical
CATEGORY_ID	Numerical
LEVEL1_CATEGORY_ID	Numerical
LEVEL1_CATEGORY_NAME	Categorical
LEVEL2_CATEGORY_ID	Numerical
LEVEL2_CATEGORY_NAME	Categorical
LEVEL3_CATEGORY_ID	Numerical
LEVEL3_CATEGORY_NAME	Categorical
UNIQUE_ID	Numerical
TYPE	Categorical

The output is also categorical which is represented as “F” and “M”. There are 3415912 duplicate data in the provided train set. These data are dropped. After, the unique values for each feature are examined. These can be seen at Table 3.

Table 3 Number of unique values

FEATURE NAME	# OF UNIQUE VALUES
TIME_STAMP	1583129
CONTENTID	482796
USER_ACTION_	5
SELLINGPRICE	35433
PRODUCT_NAME	441470
BRAND_ID	33328
BRAND_NAME	33332
BUSINESSUNIT	83
PRODUCT_GENDER	3
CATEGORY_ID	2240
LEVEL1_CATEGORY_ID	10
LEVEL1_CATEGORY_NAME	10
LEVEL2_CATEGORY_ID	95
LEVEL2_CATEGORY_NAME	95
LEVEL3_CATEGORY_ID	739
LEVEL3_CATEGORY_NAME	739
UNIQUE_ID	5618
TYPE	1
GENDER	2

Because of “type” has one unique value, it does not give information for training so it is dropped. After that number of null values are examined as can be seen at Table 4.

Table 4 Number of null values

FEATURE NAME	# OF NULL VALUES
TIME_STAMP	0
CONTENTID	2
USER_ACTION_	0
SELLINGPRICE	32013
PRODUCT_NAME	2184
BRAND_ID	2184
BRAND_NAME	2184
BUSINESSUNIT	2184
PRODUCT_GENDER	234595
CATEGORY_ID	2184
LEVEL1_CATEGORY_ID	2184
LEVEL1_CATEGORY_NAME	2184
LEVEL2_CATEGORY_ID	2184
LEVEL2_CATEGORY_NAME	2184
LEVEL3_CATEGORY_ID	2184
LEVEL3_CATEGORY_NAME	2184
UNIQUE_ID	0
TYPE	0
GENDER	0

The features are evaluated individually. For “time_stamp”, there is only 2020 as the year and December, November and October as the months. The gender distribution monthly is found as in the Table 5.

Table 5 Monthly gender distribution

GENDER	DECEMBER	NOVEMBER	OCTOBER
FEMALE	0.846	0.829	0.853
MALE	0.154	0.171	0.147

For “user_action”, there are 5 unique values as, visit, search, favorite, basket and order. For “product_gender”, there are 3 unique values and too much null values. These unique values are examined by gender as in Table 6.

Table 6 Gender distribution for product gender

GENDER	UNISEX	KADIN	ERKEK	NULL
FEMALE	0.818	0.945	0.530	0.806
MALE	0.182	0.055	0.470	0.194

2. APPROACH

2.1. Pre-Processing

Both train and test data are transformed into a format where each row represents a unique id. There are 90 columns including index, unique id and created features. Gender column is added to the train data. There are a total of 4 categorical features and 84 numerical features.

Numerical features have different scales. To obtain robust results they are scaled with min-max scaler. Min-max scaler scales each column to have a maximum value of 1 and minimum value of 0, preserving its distribution.

2.1.1. Numerical Features

Correlation within numerical features is seemed to be high. A heatmap of correlation between each feature pair is given in Figure 1. To reduce correlated variables PCA is applied to the scaled numeric features.

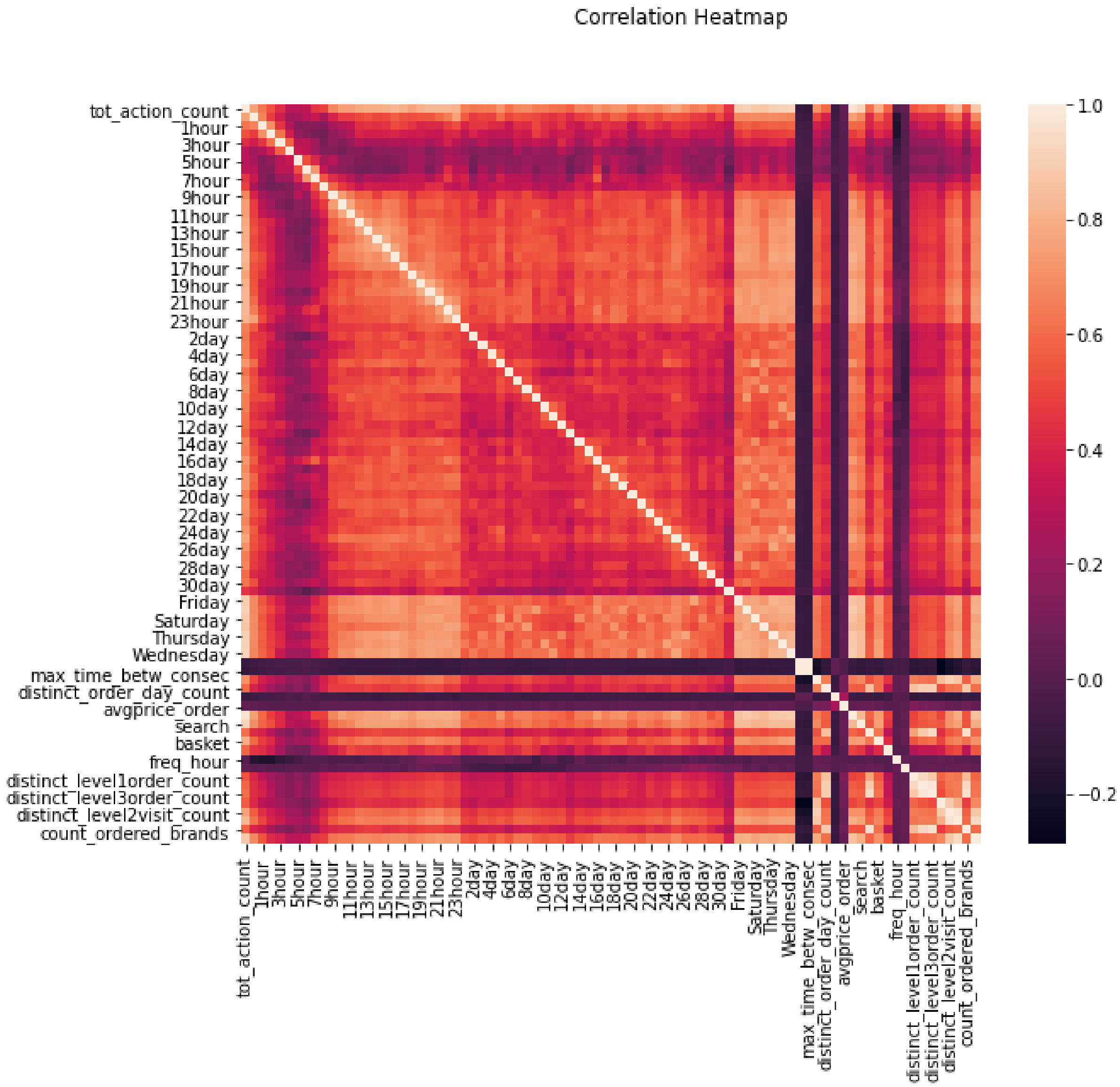

Figure 1 Correlation Heatmap for Train Data

The goal of PCA is to obtain uncorrelated variables that explains most of the variance between the original variables possibly in a space with reduced dimensions. At first PCA is applied without any dimensional constraints to have the maximum of the variance explained. Variance explained with each additional dimension and cumulative explained variance is given in Figure 1.

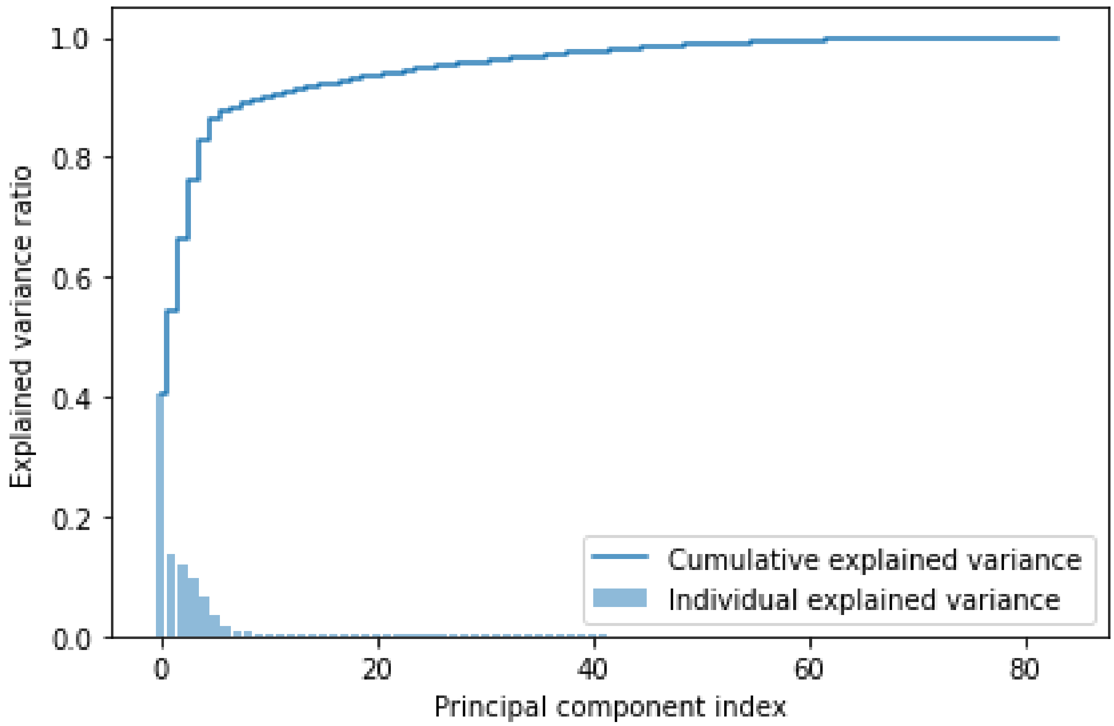

Figure 2 Explained Variance with PCA

Cumulative sum of eigenvalues which indicates the explained variance is given in Table 7. The table shows the number of eigenvalues (#) corresponding to the dimensions and cumulative variance explained (Exp.Var.) respectively.

The PCA has showed that with around 40 variables 0.9754 of the variance can be explained as it is seen in Table 7. 84 numerical features are reduced to 40 uncorrelated principal components by applying PCA. This approach is applied both to train and test data.

Table 7 Cumulative Explained Variance by Principal Components

#	EXP.VAR.	#	EXP.VAR.	#	EXP.VAR.	#	EXP.VAR.	#	EXP.VAR.
1	0.4070	**18**	0.9277	**35**	0.9675	**52**	0.9889	**69**	0.9984
2	0.5426	**19**	0.9309	**36**	0.9692	**53**	0.9897	**70**	0.9986
3	0.6629	**20**	0.9339	**37**	0.9708	**54**	0.9905	**71**	0.9988
4	0.7618	**21**	0.9368	**38**	0.9723	**55**	0.9913	**72**	0.9990
5	0.8277	**22**	0.9396	**39**	0.9738	**56**	0.9920	**72**	0.9992
6	0.8626	**23**	0.9422	**40**	0.9753	**57**	0.9927	**74**	0.9994
7	0.8779	**24**	0.9448	**41**	0.9767	**58**	0.9934	**75**	0.9995
8	0.8844	**25**	0.9472	**42**	0.9781	**59**	0.9940	**76**	0.9996
9	0.8905	**26**	0.9496	**43**	0.9794	**60**	0.9946	**77**	0.9997
10	0.8959	**27**	0.9519	**44**	0.9806	**61**	0.9951	**78**	0.9998
11	0.9011	**28**	0.9541	**45**	0.9818	**62**	0.9957	**79**	0.9999
12	0.9055	**29**	0.9562	**46**	0.9830	**63**	0.9962	**80**	1
13	0.9098	**30**	0.9582	**47**	0.9841	**64**	0.9966	**81**	1
14	0.9136	**31**	0.9602	**48**	0.9852	**65**	0.9971	**82**	1
15	0.9174	**32**	0.9621	**49**	0.9862	**66**	0.9975	**83**	1
16	0.9210	**33**	0.9640	**50**	0.9872	**67**	0.9978	**84**	1
17	0.9245	**34**	0.9658	**51**	0.9880	**68**	0.9982		

2.1.2. Categorical Features

There are 4 categorical features. The first one is “freq_level1_cat” which has 10 levels. Second and third are “freq_level2_cat” and “freq_level3_cat”, have 79 and 393 levels respectively. The last one is “freq_dow” and has 7 levels, one for each day of week (Monday, Tuesday, Wednesday, Thursday, Friday, Saturday and Sunday). Categorical variables are encoded using One-Hot Encoding.

Among the encoded columns only the ones that appear both in train and test data are kept. There is one column that appears only in train data which is “freq_level3_cat_Akıllı Bileklik”, the column is dropped from the features.

488 variables obtained with One-Hot Encoding. To reduce the number of features column sums are calculated. Columns that appear in less than 10 instances are removed. This operation reduced the number to 166 columns.

There are a lot of “freq_level2_cat” and “freq_level3_cat” levels that are encoded. "freq_level3_cat_Çorap","freq_level3_cat_Şampuan","freq_level3_cat_Çay","freq_level3_cat_Köpek Ürünleri", "freq_level3_cat_Kedi Ürünleri","freq_level2_cat_Pet Shop" are dropped since they thought to be insignificant to separate female and male customers. Encoded categorical variables reduced to 160 columns both in train and test data.

Combining numeric and categorical features final feature set is obtained. Finalized correlation heatmap is given in Figure 3. Features that has correlation coefficients more than 0.8 ('freq_level3_cat_Abiye & Mezuniyet Elbisesi', 'freq_level3_cat_Bebek Bakım', 'freq_level3_cat_Dijital Kart & Kupon', 'freq_level3_cat_Otomobil','freq_level3_cat_Parfüm', 'freq_level3_cat_Saat') are dropped.

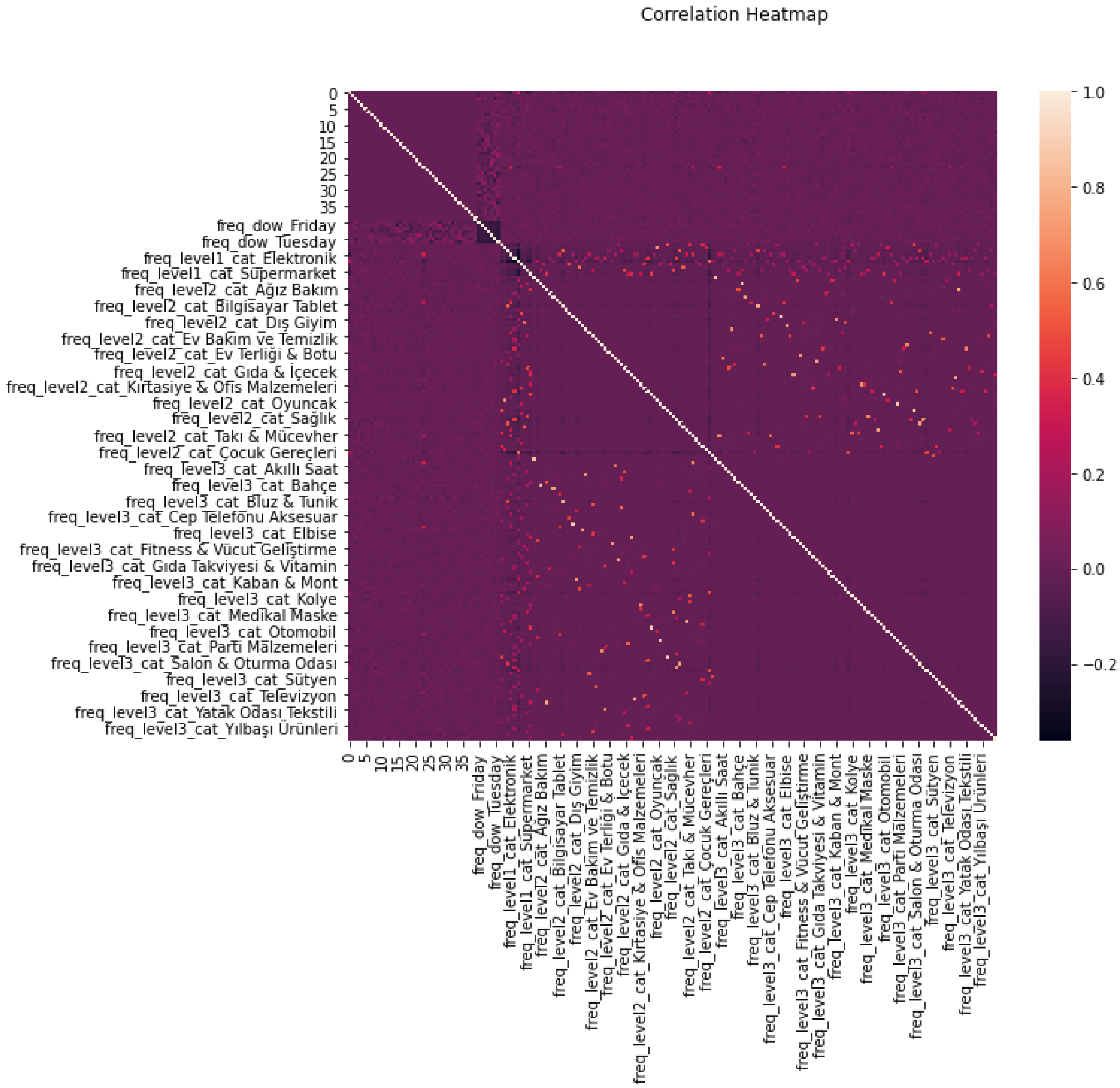

Figure 3 Finalized Features Correlation Heatmap

2.2. Model Training

All models are trained, and parameters are tuned using grid search with stratified 10-fold 5-times repeated cross-validation. Train data is split into two stratified sets, one for training and one for checking accuracy. All possible models are trained with regular data and oversampled data, accuracies checked on the test data that is not used in the model training.

During the cross validation two types of scoring which are ROC-AUC and Balanced Accuracy. Since cross validation scores obtained with balanced accuracy and ROC-AUC scorings should not be compared, it is best to compare models with balanced accuracy and models with ROC-AUC separately or all models can be compared based on test accuracies.

2.2.1. Random Forest

For the random forest classifier max_depth which determines the maximum depth of the trees and n_estimators which is the number of trees in the forest are tuned with the grid given in Table 8. ROC-AUC scoring and Balanced Accuracy scoring are used both with regular and oversampled data. Best models are given in Table 8.

Table 8 Grid for Random Forest Hyperparameter Tuning

PARAMETER	GRID
MAX_DEPTH	2,3,4
N_ESTIMATORS	200, 300, 400, 500, 800

2.2.1.1. Cross-Validation Scoring with ROC-AUC

The best random forest model is found as

RandomForestClassifier(max_depth=4, n_estimators=500, random_state=582).

The best score obtained during cross-validation is 0.7263.

Accuracy on test data is 0.6142.

In addition, using oversampled data model is trained again. The best model is found to be **RandomForestClassifier(max_depth=4, n_estimators=400, random_state=582).**

The best score obtained during cross-validation is 0.7255.

Accuracy on test data is 0.6124.

2.2.1.2. Cross-Validation Scoring with Balanced Accuracy

The best random forest model with regular data is found as,

RandomForestClassifier(max_depth=4, n_estimators=200, random_state=582).

The best score obtained during cross-validation is 0.5000.

Accuracy on test data is 0.6552.

Using oversampled data model is trained again. The best model is found to be **RandomForestClassifier(max_depth=4, n_estimators=800, random_state=582).**

The best score obtained during cross-validation is 0.6631.

Accuracy on test data is 0.6136.

Table 9 Random Forest Best Models

MODEL	OVERSAMPLING	SCORING	AVG CV SCORE	EVALUATION DATA ACCURACY
RANDOM FOREST	No	ROC_AUC	0.7263	0.6142
RANDOM FOREST	Yes	ROC_AUC	0.7255	0.6125
RANDOM FOREST	No	BALANCED_ACCURACY	0.5001	0.6552
RANDOM FOREST	Yes	BALANCED_ACCURACY	0.6631	0.61365

2.2.2. XGBoost for Classification

For the XGBoost classifier max_depth which determines the maximum depth of the trees and eta which is the learning rate are tuned with the grid given in Table 10. ROC-AUC scoring and Balanced Accuracy scoring are used both with regular and oversampled data. Best models are given in Table 10.

Table 10 Grid for XGBoost Hyperparameter Tuning

PARAMETER	GRID
MAX_DEPTH	4,5,6,7,10,12,14,16
ETA	0.1, 0.3, 0.5, 0.7

2.2.2.1. Cross-Validation Scoring with ROC-AUC

The best model is found as

**XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
eta=0.1, eval_metric='auc', gamma=0, gpu_id=-1,
importance_type=None, interaction_constraints='',
learning_rate=0.100000001, max_delta_step=0, max_depth=4,
min_child_weight=1, missing=nan, monotone_constraints='()',
n_estimators=100, n_jobs=8, num_parallel_tree=1, predictor='auto',
random_state=582, reg_alpha=0, reg_lambda=1, scale_pos_weight=1,
seed=582, subsample=1, tree_method='exact', validate_parameters=1, ...)**

The best score obtained during cross-validation is 0.7453.

Accuracy on test data is 0.6944.

With oversampled data model the best model is found to be

**XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
eta=0.1, eval_metric='auc', gamma=0, gpu_id=-1,
importance_type=None, interaction_constraints='',
learning_rate=0.100000001, max_delta_step=0, max_depth=16,
min_child_weight=1, missing=nan, monotone_constraints='()',
n_estimators=100, n_jobs=8, num_parallel_tree=1, predictor='auto',
random_state=582, reg_alpha=0, reg_lambda=1, scale_pos_weight=1,
seed=582, subsample=1, tree_method='exact', validate_parameters=1, ...)**

The best score obtained during cross-validation is 0.9217.

Accuracy on test data is 0.6961.

Since the best model obtained has max_depth parameter as 16 and eta as 0.1 which are the maximum and minimum values in the provided grid, another search is conducted with an extended grid that is given in Table 11.

Table 11 Extended Grid for XGBoost Hyperparameter Tuning

PARAMETER	GRID
MAX_DEPTH	16,18,20,22,24
ETA	0.05, 0.06, 0.08, 0.1, 0.2

The best model is found with regular data and extended grid as

XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
eta=0.1, eval_metric='auc', gamma=0, gpu_id=-1,
importance_type=None, interaction_constraints='',
learning_rate=0.100000001, max_delta_step=0, max_depth=16,
min_child_weight=1, missing=nan, monotone_constraints='()',
n_estimators=100, n_jobs=8, num_parallel_tree=1, predictor='auto',
random_state=582, reg_alpha=0, reg_lambda=1, scale_pos_weight=1,
seed=582, subsample=1, tree_method='exact', validate_parameters=1, ...)

The best score obtained during cross-validation is 0.7257.

Accuracy on test data is 0.6949.

With oversampled data and extended grid the best model is found to be

XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
eta=0.2, eval_metric='auc', gamma=0, gpu_id=-1,
importance_type=None, interaction_constraints='',
learning_rate=0.200000003, max_delta_step=0, max_depth=22,
min_child_weight=1, missing=nan, monotone_constraints='()',
n_estimators=100, n_jobs=8, num_parallel_tree=1, predictor='auto',

random_state=582, reg_alpha=0, reg_lambda=1, scale_pos_weight=1,

seed=582, subsample=1, tree_method='exact', validate_parameters=1, ...)

The best score obtained during cross-validation is 0.9221.

Accuracy on test data is 0.6896.

2.2.2.2. Cross-Validation Scoring with Balanced Accuracy

Best model with regular data is found as

XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,

colsample_bynode=1, colsample_bytree=1, enable_categorical=False,

eta=0.1, eval_metric='auc', gamma=0, gpu_id=-1,

importance_type=None, interaction_constraints='',

learning_rate=0.100000001, max_delta_step=0, max_depth=6,

min_child_weight=1, missing=nan, monotone_constraints='()',

n_estimators=100, n_jobs=8, num_parallel_tree=1, predictor='auto',

random_state=582, reg_alpha=0, reg_lambda=1, scale_pos_weight=1,

seed=582, subsample=1, tree_method='exact', validate_parameters=1, ...)

The best score obtained during cross-validation is 0.6319.

Accuracy on test data is 0.6961.

Using oversampled data model is trained again. Best model is found to be **XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,**

colsample_bynode=1, colsample_bytree=1, enable_categorical=False,

eta=0.1, eval_metric='auc', gamma=0, gpu_id=-1,

importance_type=None, interaction_constraints='',

learning_rate=0.100000001, max_delta_step=0, max_depth=16,

min_child_weight=1, missing=nan, monotone_constraints='()',

n_estimators=100, n_jobs=8, num_parallel_tree=1, predictor='auto',

random_state=582, reg_alpha=0, reg_lambda=1, scale_pos_weight=1,

seed=582, subsample=1, tree_method='exact', validate_parameters=1, ...)

The best score obtained during cross-validation is 0.8238.

Accuracy on test data is 0.6961.

With the oversampled data the model has parameter values 16 for max_depth and 0.1 for eta. Therefore, using the extended grid given in Table 11. XGBoost cross-validation is run again.

Using oversampled data and extended grid the best model is found to be

The best score obtained during cross-validation is

Accuracy on test data is

Table 12 XGBoost Best Models

MODEL	OVERSAMPLING	SCORING	AVG CV SCORE	EVALUATION DATA ACCURACY
XGBOOST	No	ROC_AUC	0.7453	0.6944
XGBOOST	Yes	ROC_AUC	0.9217	0.6961
XGBOOST (EXTENDED GRID)	No	ROC_AUC	0.7257	0.6949
XGBOOST (EXTENDED GRID)	Yes	ROC_AUC	0.9221	0.68961
XGBOOST	No	BALANCED_ACCURACY	0.6319	0.6961
XGBOOST	Yes	BALANCED_ACCURACY	0.8238	0.6961
XGBOOST (EXTENDED GRID)	Yes	BALANCED_ACCURACY	0.8255	0.6896

2.2.3. Logistic Regression

For logistic regression different penalty grids are used for different solvers. Solvers, penalty grid for each solver and best cross-validation scores are given in Table 13. Models trained with oversampled data outperformed others considering the average cross-validation AUC scores. Best scored models are the ones with L1.

Table 13 Logistic Regression Best Models

MODEL	OVERSAMPLING	SCORING	SOLVER	PENALTY GRID	AVG CV SCORE
LOGISTIC REGRESSION	No	ROC_AUC	Liblinear	L1, L2	0.7353
LOGISTIC REGRESSION	Yes	ROC_AUC	Liblinear	L1, L2	0.7574
LOGISTIC REGRESSION	No	ROC_AUC	Newton-CG	L2, None	0.7317
LOGISTIC REGRESSION	Yes	ROC_AUC	Newton-CG	L2, None	0.7547
LOGISTIC REGRESSION	No	ROC_AUC	LBFGS	L2, None	0.7317
LOGISTIC REGRESSION	Yes	ROC_AUC	LBFGS	L2, None	0.7548
LOGISTIC REGRESSION	No	ROC_AUC	Sag	L2, None	0.7317
LOGISTIC REGRESSION	Yes	ROC_AUC	Sag	L2, None	0.7552
LOGISTIC REGRESSION	No	ROC_AUC	Saga	L1, L2, Elasticnet, None	0.7356
LOGISTIC REGRESSION	Yes	ROC_AUC	Saga	L1, L2, Elasticnet, None	0.7576

3. RESULTS

Logistic regression models performed poorly when compared to random forest and Xgboost models therefore they are not considered in the submission phase. Overview of models chosen considering ROC-AUC and balanced accuracy measures are given in Table 14 and Table 15 separately.

Table 14 ROC_AUC Random Forest and XGBoost Model Comparison

MODEL	OVERSAMPLING	SCORING	AVG CV SCORE	EVALUATION DATA ACCURACY
RANDOM FOREST	No	ROC_AUC	0.7263	0.6142
RANDOM FOREST	Yes	ROC_AUC	0.7255	0.6125
XGBOOST	No	ROC_AUC	0.7453	0.6944
XGBOOST	Yes	ROC_AUC	0.9217	0.6961
XGBOOST (EXTENDED GRID)	No	ROC_AUC	0.7257	0.69495
XGBOOST (EXTENDED GRID)	Yes	ROC_AUC	0.9221	0.6896

Table 15 Balanced Accuracy Random Forest and XGBoost Model Comparison

MODEL	OVERSAMPLING	SCORING	AVG CV SCORE	EVALUATION DATA ACCURACY
RANDOM FOREST	No	BALANCED_ACCURACY	0.5001	0.6552
RANDOM FOREST	Yes	BALANCED_ACCURACY	0.6631	0.6136
XGBOOST	No	BALANCED_ACCURACY	0.6319	0.6961
XGBOOST	Yes	BALANCED_ACCURACY	0.8238	0.6961
XGBOOST (EXTENDED GRID)	Yes	BALANCED_ACCURACY	0.8255	0.6896

During the submission period selected models which are XGBoost models obtained with ROC-AUC scoring, regular grid and oversampled data, ROC-AUC scoring, extended grid and oversampled data, balanced accuracy scoring, regular grid and oversampled data are tried to see performance on the actual evaluation data. The best scores are obtained with the model which is obtained with ROC-AUC scoring, extended grid and oversampled data. The selected model for the final submission is

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
        colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
        eta=0.2, eval_metric='auc', gamma=0, gpu_id=-1,
        importance_type=None, interaction_constraints='',
        learning_rate=0.200000003, max_delta_step=0, max_depth=22,
        min_child_weight=1, missing=nan, monotone_constraints='()',
        n_estimators=100, n_jobs=8, num_parallel_tree=1, predictor='auto',
        random_state=582, reg_alpha=0, reg_lambda=1, scale_pos_weight=1,
        seed=582, subsample=1, tree_method='exact', validate_parameters=1, ...)
```

4. CONCLUSION AND FUTURE WORK

In this project, aim was to build a model for classification to predict the gender. Provided data was a real-world data and it was unstructured. By feature engineering, data was transformed to a structured data. There were both numerical and categorical data type. Besides, number of duplicated data and null variables were too much. With elimination and by using dummy variable method, 10 categorical and 8 numerical features were transformed to 4 categorical and 84 numerical features. Principal component analysis (PCA) was applied to the numerical features. After obtaining a proper dataset, types of models were applied. These were random forest, XGBoost, and Logistic Regression. The performance of the logistic regression was the poorest one. When the performance of the random forest and XGBoost were compared, XGBoost's performance was the best one. However, the overall performance was not good enough. To get better result, more feature engineering can be done. Besides, the amount of the training examples is enough to apply deep learning. It can perform better result.

5. CODE

Hatice Pınar YILDIRIM - https://bu-ie-582.github.io/fall21-hpinaryildirim/files/IE582-PROJECT.html

İrem Gülçin ZIRHLIOĞLU - https://bu-ie-582.github.io/fall21-iremgulcin/IE_Project/IE582-project-codes-final.html