BOĞAZİÇİ UNIVERSITY

IE 582 PROJECT

GROUP 11

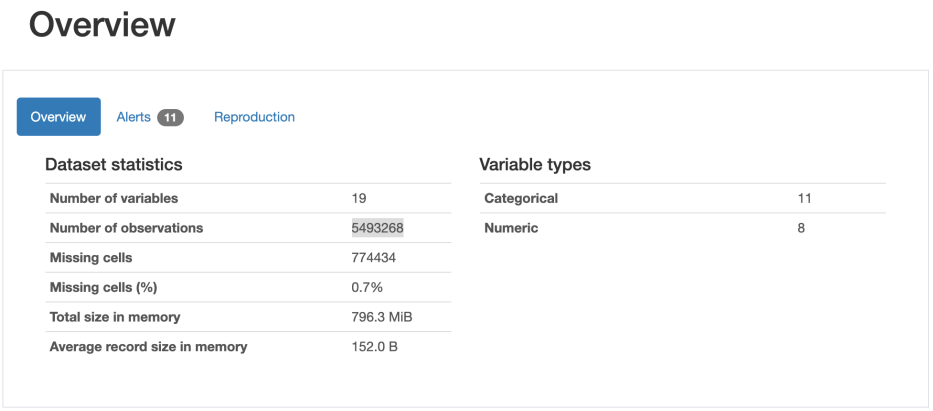
Enes Fevzeddin Kacar	2016402420
Goktug Ovren	2021705045

TABLE OF CONTENTS

1. INTRODUCTION

The technique of obtaining valuable information from a large volume of raw data is known as data mining. Over the last several decades, this notion has grown in popularity around the world and is now frequently applied in fields such as science and business. Behavioral analysis of data based on trends, prediction of outcomes, forecasting on future values, building of decision-based structures, analysis of massive data, and clustering are some of the primary uses of data mining.The goal of this project is to employ sophisticated data mining algorithms to successfully evaluate the supplied data and make predictions for solving a binary classification issue in several instances.

The train and test data are the two primary elements of the data supplied for this work. The train data comprises 18 features for occurrences, as well as binary target values for all of these instances, whereas the test data has 4.8 m instances for the same 18 features but without the target values. The basic goal is to create a framework that learns from the training data and uses it to predict the test data's target values. It's worth noting that there may be too many features compared to the amount of occurrences, resulting in models that "understand" the train data "too well." To put it another way, overfitted models should be avoided since they may perform exceptionally well on some datasets.



2. RELATED LITERATURE

2.1. Models/Algorithms

2.1.1. Decision Tree

A decision tree is a basic tree-like structure with nodes and branches. At each node, data is separated based on any of the input attributes, resulting in two or more output branches. The number of created branches grows as the iterative process progresses, and the original data gets partitioned. This process is repeated until a node is created in which all or almost all of the data belongs to the same class, at which point additional splits are no longer allowed. The CART acronym stands for Decision Tree Algorithms (Classification and Regression Trees).

2.1.2. Random Forest

Another strong and widely used supervised learning technique is random forest. It's an ensemble method that takes into account the findings of many categorization algorithms of the same or different types. It enables the rapid discovery of important data from large databases. The most significant benefit of random forest is that it depends on a collection of different decision trees to arrive at any answer.

2.1.3. Logistic Regression

Logistic regression is a classification algorithm used to assign observations to a discrete set of classes. Unlike linear regression which outputs continuous number values, logistic regression transforms its output using the logistic sigmoid function to return a probability value which can then be mapped to two or more discrete classes.

2.1.4. LightGBM

LightGBM is a gradient boosting framework that uses tree based learning algorithms. It is designed to be distributed and efficient with the following advantages:

- Faster training speed and higher efficiency.
- Lower memory usage.
- Better accuracy.
- Support of parallel, distributed, and GPU learning.
- Capable of handling large-scale data.

3. Approach

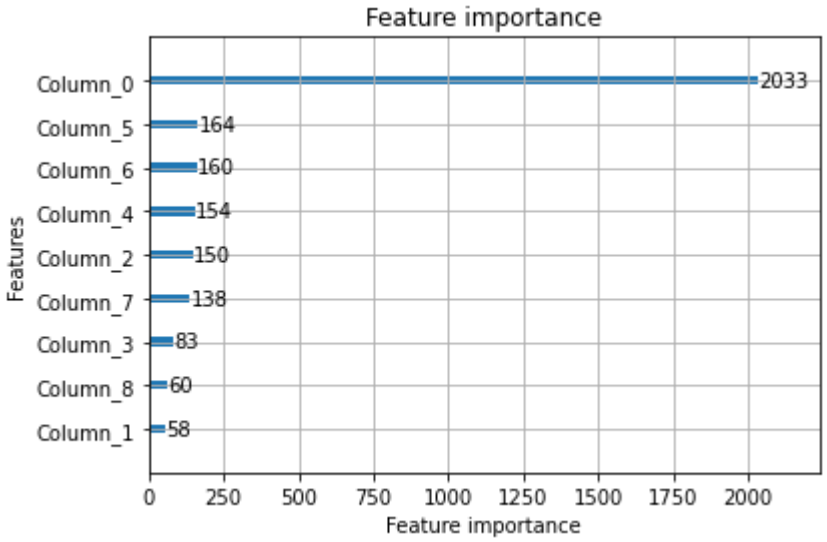
- Firstly, we looked for the correlations, but we couldn’t see any significant correlation between structured(numeric) columns and the target columns. Unfortunately, we weren't able to use unstructured(categorical) columns in the correlation matrix.
- We looked for anomalies for numeric columns but it looks like there are no misregistered values, so we didn’t make any changes in values.
- Our biggest issue was handling categorical variables, which form most of our data. Since these columns are not-ordinal data, we should use the OHE(one-hot-encoding) method to make them structured data. However, we couldn’t be able to use OHE even 1/50 of our data because of memory constraints. There should be about 400k

columns to handle all of the values. So, we decided to use the columns which are probably strongly correlated with our target columns, *product gender* and *user action* columns to implement OHE.

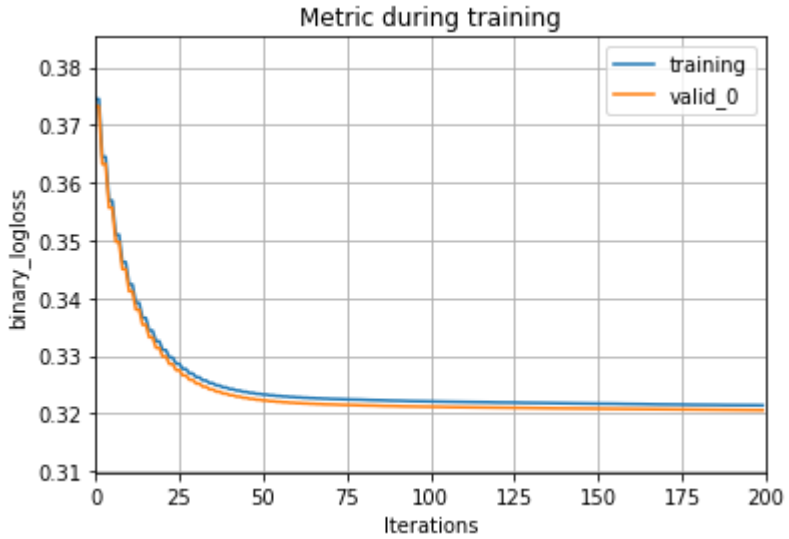
- When we started to create the model, we used the Logistic Regression and Random Forest model. We scaled price values between 0 and 1 for the logistic regression model. As we predicted from the results of reviewed articles, the random forest model performed significantly better.
- We implemented 10K cross validation to tune the hyperparameters of our logistic regression and random forest models.
- Since we have an unbalanced data set, we got significantly better results when we assigned the class_weight variable to ‘balanced’.
- Then we tried to implement, Light Gradient Boosting Model (LightGBM) and we got slightly better results with a boosting model.

4. Results

LightGBM First model: By Using the LightGBM model we created a feature importance graph. As it can be seen from the graph, the selling price column had a very high importance value, which shouldn’t be that high. We might make some mistakes while handling the data.



column names in order:
sellingprice,
user_action_basket,
user_action_favorite,
user_action_order,
user_action_search,
user_action_visit,
product_gender_Erkek,
product_gender_Kadın,
product_gender_Unisex



	precision	recall	f1-score	support
0	0.58	0.22	0.32	156420
1	0.89	0.98	0.93	1040987
accuracy			0.88	1197407
macro avg	0.74	0.60	0.63	1197407
weighted avg	0.85	0.88	0.85	1197407

LightGBM Second model:

```
Training until validation scores don't improve for 50 rounds.
[200]   train's binary_logloss: 0.561658        train's auc: 0.778667   valid's binary_logloss:
0.55463 valid's auc: 0.777429
[400]   train's binary_logloss: 0.56013 train's auc: 0.7802     valid's binary_logloss: 0.55323
valid's auc: 0.778801
[600]   train's binary_logloss: 0.559169        train's auc: 0.781184   valid's binary_logloss:
0.552421        valid's auc: 0.779646
[800]   train's binary_logloss: 0.558579        train's auc: 0.781759   valid's binary_logloss:
0.551937        valid's auc: 0.780099
[1000]  train's binary_logloss: 0.558069        train's auc: 0.78226    valid's binary_logloss:
0.551526        valid's auc: 0.7805
[1200]  train's binary_logloss: 0.557689        train's auc: 0.782616   valid's binary_logloss:
0.551231        valid's auc: 0.780774
[1400]  train's binary_logloss: 0.557418        train's auc: 0.782847   valid's binary_logloss:
0.551015        valid's auc: 0.780932
[1600]  train's binary_logloss: 0.557193        train's auc: 0.78305    valid's binary_logloss:
0.550835        valid's auc: 0.781058
[1800]  train's binary_logloss: 0.557009        train's auc: 0.783209   valid's binary_logloss:
0.550691        valid's auc: 0.781145
[2000]  train's binary_logloss: 0.556863        train's auc: 0.783346   valid's binary_logloss:
0.550588        valid's auc: 0.781193
[2200]  train's binary_logloss: 0.55675 train's auc: 0.78344    valid's binary_logloss: 0.55051
valid's auc: 0.78123
[2400]  train's binary_logloss: 0.556657        train's auc: 0.783518   valid's binary_logloss:
0.550445        valid's auc: 0.781263
Early stopping, best iteration is:
[2426]  train's binary_logloss: 0.556645        train's auc: 0.783531   valid's binary_logloss:
0.550437        valid's auc: 0.781269
LGBMClassifier(class_weight='balanced', learning_rate=0.05, n_estimators=10000,
               objective='binary', random_state=50, reg_alpha=0.1,
               reg_lambda=0.1, subsample=0.8)
```

Decision tree results

	fit_time	score_time	test_roc_auc	train_roc_auc	test_balanced_accuracy	train_balanced_accuracy	test ort	train ort
0	15.307790	1.231427	0.837325	0.855609	0.656330	0.664251	0.746828	0.759930
1	15.321142	1.135144	0.837071	0.855607	0.654468	0.663548	0.745770	0.759578
2	17.076938	1.188607	0.837191	0.855656	0.655525	0.663541	0.746358	0.759598
3	14.977336	1.407796	0.836078	0.855832	0.654525	0.663626	0.745301	0.759729
4	15.743449	1.167983	0.836009	0.855821	0.653988	0.664162	0.744999	0.759992
Mean	15.685331	1.226191	0.836735	0.855705	0.654967	0.663826	0.745851	0.759765
SD	0.737072	0.096019	0.000571	0.000101	0.000845	0.000314	0.000670	0.000169

Logistic regression results

	fit_time	score_time	test_roc_auc	train_roc_auc	test_balanced_accuracy	train_balanced_accuracy	test ort	train ort
0	37.398607	0.879138	0.759424	0.759951	0.512251	0.511784	0.635838	0.635867
1	33.553721	0.754013	0.767099	0.767217	0.518716	0.518590	0.642908	0.642903
2	35.488016	0.748465	0.765643	0.767185	0.573062	0.572369	0.669352	0.669777
3	40.514200	0.834343	0.768447	0.766858	0.511367	0.511416	0.639907	0.639137
4	40.765487	0.992832	0.765728	0.764949	0.571734	0.572509	0.668731	0.668729
Mean	37.544006	0.841758	0.765268	0.765232	0.537426	0.537334	0.651347	0.651283
SD	2.806086	0.090198	0.003097	0.002770	0.028670	0.028777	0.014622	0.014844

5. Conclusion and Future Work

- Dealing with categorical data was our biggest issue in this project. And, unfortunately we couldn’t use most of the categorical variables. There might be some models that can deal with categorical data that will handle this problem much better.
- Getting good performance with unbalanced data was a challenge. We tried class weight variables to deal with it but according to results we didn’t do a good job. We might put equal weighted classes into the models so they can learn rare seen classes better.
- In our model, we didn’t use the time column. There may be a correlation between time column and the target so it would be used a variable.
- To handle memory constraints, we might try different libraries which are created for handling very data so we can implement the OHE for categorical variables.

- After a successful implementation of the OHE, we should consider the PCA(Principal component analysis) because we will have so many dimensions after OHE.
- Our model resulted relatively good about ROC-AUC score, however the results are not good for Balanced Accuracy Score. Because our fit module was trying to increase AUC score but not BAS score. We would use a model that increases BAS score at the same time.

6. Reference

https://economictimes.indiatimes.com/definition/data-mining

http://theprofessionalspoint.blogspot.com/2019/02/difference-between-gbm-gradient.html

https://stackoverflow.com/questions/49984506/caretxgtree-there-were-missing-values-in-resampled-performance-measures

https://scikit-learn.org/stable/

https://www.kaggle.com/willkoehrsen/start-here-a-gentle-introduction

https://scikit-learn.org/stable/modules/tree.html

https://lightgbm.readthedocs.io/en/latest/

7. Code

Github folder link: https://github.com/BU-IE-582/fall21-ovren1/tree/main/Final

Notebook: ovren1/blob/main/Final/project_worksheet.ipynb

Report: https://bu-ie-582.github.io/fall21-ovren1/Final/project_worksheet.html