BLG 454E Learning From Data
Term Project Report

Furkan Artunç, Ibrahim Aliu

***Abstract*— Data filled with orders of a company is analyzed together with the information of each item either is returned or not with purpose of foreseeing returns of future sales. This process will reduce the number of returns without having an real effect on customer satisfaction.**

I. INTRODUCTION

In order to predict if a sale will be returned back or kept, we analyze annual data of a company with plenty of features and the status of returns for each sale. This information will be used to reduce return cost of the company by analyzing the data to predict returns of each future sale of the company. Given training data is not pure, we have to clean the data first, work some methods to train network using training data, and also use test data for prediction, export the results for each instance.

Kaggle names: Artunc, Ibrahim Aliu
Team name: 150130048_150130901
Score: 10679.00000
Rank: 16

II. DATA SET USED

Annual data (train data) [1] of the company is not directly ready to be used. Data lacks in some features of sales by being empty, also need to have some constraints in some features in order to be right. There are lowercase and uppercase differences in data too. Before applying any method, we need to clean the data first.

We cleaned the data by dropping instances that have empty feature values. We have tried to take mode for these values to have a bigger training data, but it apparently did not work well and created noise for our predictions. Also we converted all string values into lowercase too. In addition, in size feature, lots of the instances have 'unsized' value. We converted them into three most repeated values.

Test data have some dummy values too. In order to fix them, we used modes of the columns (most repeated values for each column) to replace these improper values. Some features of the data are not numeric but string and date values. After cleaning the data properly, string values of features must be encoded into numeric values. We used labelencoder from sklearn.preprocessing [5] to encode labels.

III. METHODS USED

We used ExtraTreesClassifier method from sklearn.ensemble [6] for feature selection model. But feature selection reduced the accuracy of the data, so we cancel using feature selection method.

Then, we trained our network by using XGBClassifier module from xgboost library [2]. In addition, we also use RandomForestClassifier, train network with random forest from sklearn[7], LogisticRegression, train network from sklearn.linear_model [8], and created prediction label for each method.

Finally, we took mode of both prediction labels for each instance in order not to depend only to single method, but choosing the predictions democratically between three methods. Here is detailed information about training methods that we used in our project.

XGBoost:

XGBoost is short for "Extreme Gradient Boosting", inspired by Friedman's paper for Gradient Boosting [2]. XGBoost is used for supervised learning problems, where we use the training data to predict a target variable. Lots of people in Kaggle uses XGBoost because of its performance.

The model that xgboost using is tree ensembles. The tree ensemble model is a set of classification and regression trees (CART). A CART is a bit different from decision trees. In CART, a real score is associated with each of the leaves, which gives us richer interpretations that go beyond classification. Usually, multiple trees are used in practice because single tree is not strong enough to predict. This model is tree ensemble model, which sums the prediction of multiple trees together.

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in F \tag{1}$$

where:
K: the number of trees
f: function in the functional space F
F: the set of all possible CARTs.

Objective function to optimize can be written as

$$obj(\theta) = \sum_{i}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{2}$$

This model uses additive training. For the parameters, what is needed to be learned are these functions f_i, with each containing the structure of the tree and the leaf scores. This is much harder than traditional optimization problem. It is not easy to train all the trees at once. Instead, this uses use an additive strategy: fix what we have learned, and add one new tree at a time.

$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i) \tag{3}$$

We can optimize every loss function, including logistic regression and weighted logistic regression, using exactly the same solver that takes g_i and hi as input. In XGBoost, we define the complexity as:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} \omega_j^2 \tag{4}$$

Here:
ω: the vector of scores on leaves
T: the number of leaves.

Logistic Regression:

Logistic regression is the regression analysis to conduct when the dependent variable is binary. This is also a predictive analysis. For the binary logistic regression, we have some assumptions:

The dependent variable is binary.
There should be no outliers.
There should be no high correlations.

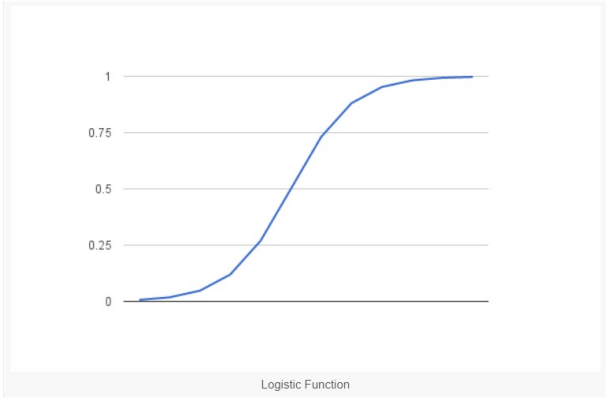

Figure 1 [3]

The logistic function (showed above in Figure 1), also called the sigmoid function was developed by statisticians to describe some properties in nature. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.

$$\frac{1}{1 + e^{-value}} \tag{5}$$

Input values are combined linearly using weights to predict an output value [3]. A key difference from linear regression is that the output value being modeled is a binary value either 0 or 1, rather than a numeric value. Logistic regression is a linear method, but the predictions are transformed using the logistic function.

$$y = \frac{e^{b_0 + b_1 x}}{1 + e^{b_0 + b_1 x}} \tag{6}$$

where:
b_0 is the bias
b_1 is the coefficient for the single input value

The coefficients of the logistic regression algorithm must be estimated from the training data. This is done using maximum-likelihood estimation.

Random Forest:

Random Forests is a method that grows many classification trees. Each tree gives a classification for the input, and similar to voting systems today, the output is selected which takes the most of the votes.

There are two main effect that effect accuracy of the forest. If the correlation between two trees in the forest increases, the error rate of the forest increases too. Also, a tree with low error rate is a strong classifier. Not increasing these individual tree strengths decreases the accuracy of the forest too.

When the training set for the current tree is drawn by sampling with replacement, about 33% of the cases are left out of the sample. This out-of-bag data is used to get a running unbiased estimate of the classification error as trees are added to the forest. [4]

After each tree is built, all of the data are run down the tree, and proximities are computed for each pair of cases. At the end, it has to be normalized by dividing by the number of trees. Proximities are used in replacing missing data, locating outliers, and producing illuminating low-dimensional views of the data [4].

In random forests, there is no need for cross-validation or a separate test set to get an unbiased estimate of the test set error. It is estimated internally during the run.

IV. RESULTS

At first, we only used XGBoost method to train our network. To improve accuracy, we also used Linear Regression and Random Forest methods. In order to raise the accuracy even more, we could have done some feature engineering to data. One of the solutions could be feature transformation. To remove skewness, we could have normalized our features.

Also, in size features, there are many units mixed inside. We could have convert all the units into a single size unit by changing other units to corresponding value in that unit.

V. CONCLUSIONS

To conclude, we used an annual data of a company as the training data to train a model for predicting the returns of future sales. For testing our model, we predicted test data orders and assigned return numbers 0 as will not be returned and 1 as will be returned. To improve accuracy of our results, we used three models together and assigned the most selected output for each instance.

REFERENCES

[1] İ. Bilgen, O. S. Saraç, "Prediction of return in online shopping," in *Signal Processing and Communications Applications Conference(SIU)* , 2015 23th, pp. 2577-2580, IEEE, 2015

[2] "Introduction to Boosted Trees." [Online]. Available: http://xgboost.readthedocs.io/en/latest/model.html. [Accessed: 25-May-2018].

[3] J. Brownlee, "Logistic Regression for Machine Learning," 2016. [Online]. Available: https://machinelearningmastery.com/logistic-regression-for-machine-learning/. [Accessed: 25-May-2018].

[4] N. Donges, "The Random Forest Algorithm," 22-Feb-2018. [Online]. Available: https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd. [Accessed: 25-May-2018].

[5] "sklearn.preprocessing.LabelEncoder." [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html. [Accessed: 25-May-2018].

[6] "sklearn.ensemble.ExtraTreesClassifier." [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html. [Accessed: 25-May-2018].

[7] "sklearn.ensemble.RandomForestClassifier." [Online]. Available: http://scikit-learn.org/stable/modules/generated/] sklearn.ensemble.RandomForestClassifier.html. [Accessed: 25-May-2018].

[8] "sklearn.linear_model.LogisticRegression." [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html. [Accessed: 25-May-2018].