BOĞAZİÇİ UNIVERSITY

DEPARTMENT OF MANAGEMENT INFORMATION SYSTEMS

PRICE RECOMMENDATION SYSTEM FOR ONLINE RETAIL SELLERS

Ceren Öyke
Görkem Çoklar
Hakan Utku Özdemir
Zeynep Günce Gündüz

July, 2021

PRICE RECOMMENDATION SYSTEM FOR ONLINE RETAIL SELLERS

by

Ceren Öyke
Görkem Çoklar
Hakan Utku Özdemir
Zeynep Günce Gündüz

Submitted to Department of Management Information Systems
as a requirement for the degree of
Bachelor of Science
in
Management Information Systems

Boğaziçi University
July, 2021

PRICE RECOMMENDATION SYSTEM FOR ONLINE RETAIL SELLERS

APPROVED BY:

Name of Advisor	Prof. Dr. Sona Mardikyan
Name of Jury Member1	Prof. Dr. Aslıhan Nasır
Name of Jury Member2	Prof. Dr. Meltem Özturan

DATE OF APPROVAL : .....

TABLE OF CONTENTS

ACKNOWLEDGEMENT

To begin with, we would like to thank our advisor, Mrs. Sona Mardikyan for her guidance, support, and help throughout the project.

ABSTRACT

Machine learning algorithms help to analyze complex data and find meaning in it. With the regression algorithms under this subject, predictions can be made based on historical data. In today's world data is vital for e-commerce and data science helps businesses to obtain a richer understanding of the customers. In the light of this information, our study aims to solve a real-world data science problem by collaborating with an e-commerce company in Turkey. The models created will be helpful to meet the need for a price recommendation system by eliminating human interference, speeding up the efficiency of the shopping experience. In this study, different regression models will be designed and the success of the models will be compared.

1. INTRODUCTION

Second-hand fashion, which has been rising in the e-commerce sector in the world recently, is becoming increasingly common in every category such as fashion, home&life and electronics. We have worked with one of the leading companies of Turkey in this sector and this company aims to give a personalized shopping experience further with advanced technology and to establish second-hand shopping habits in Turkey. It is simply a platform on which sellers can upload their used/refurbished products to sell.

In the company that we collaborated with, sellers seek the maximum price at which they can sell their products, while buyers try to find a product with maximum quality at a low price, therefore they would like to give pricing suggestions to its sellers. In their current system, to determine the price of the product, the seller should do product research, check its competitors' prices, condition of the product, various other factors, and then come up with the correct price for the product. After the seller uploads the product, buyers can bid on the product price and bargain for it. After the agreement between the buyer and the seller, the purchased products are shipped to the address of the buyer with the contracted cargo company. In the present price selection system, the risk is high, because if the price list is too high or too low compared with the market price, both sellers and buyers can be in a loss state. As a result, there is a need for price recommendations to lead the seller, and it should be done according to the uploaded product information from sellers. Because there is a wide range of products on the platform with various features, the system implementation is complex.

Machine learning techniques can be very useful to diminish complexity. To solve prediction problems with machine learning, there is a need for regression analysis which helps to find the correlation between variables and allows us to predict the output variable based on predictor variables. In this study, creating a price suggestion model by using the right machine learning methods to provide the customers a smooth shopping experience is aimed.

2. REVIEW OF THE LITERATURE, BACKGROUND OF THE PROBLEM

In this section, the literature about price suggestion systems are examined. Three examples discussed below are mainly considered while developing the model in this project.

One of the studies about price estimation is for predicting house pricing with the Random Forest model. Random Forests (RF) is a machine learning ensemble approach

based on weak learner decision trees that are frequently utilized in prediction. This study used the single-family house assessment price of 2015 of Arlington, Virginia USA and they had 27.649 rows of data. (Wang & Wu, 2018) They used the information of house price, size of the house, the year the house has been built, zip code of the house, and the location of the house to build the model. Finally, they found that Random Forests can capture the nonlinear hidden link between property price and location, as well as provide a superior overall estimation than standard linear regression.

Another price estimation study is about second-hand C2C platform data, which is close to our subject in terms of the objectives. Because secondhand platforms do not have limitations about how users set prices of their products, they preferred to proceed with vision-based price suggestions. (Han et al., 2020). The suggested vision-based pricing recommendation system is designed to provide effective price suggestions for online listed second-hand products based on visual elements derived from submitted photos as well as certain statistical data. The uploaded image, as well as other historical statistical characteristics received from the platform, are collected by the mobile app when a seller takes a photograph for a second-hand item and uploads it to the mobile application for an item listing. The image will then have its visual characteristics retrieved. The study built a binary classification model to assess whether an item picture is qualified for price prediction, and a regression model to forecast prices for products with qualified photos.

Another study of price suggestions, which we were inspired by, is based on the Mercari competition on the Kaggle platform (Mercari Price Suggestion Challenge, 2018) Mercari, which is Japan's largest community-powered buying platform, is one such online second-hand marketplace. Mercari aimed to offer price recommendations to its merchants. (Dave, 2020) This is challenging since merchants on the Mercari marketplace can sell practically anything or any combination of products. The data include the name of the listings, condition of the item, category and brand name, price, shipping term, and item description and because the prices are continuous variables, the machine learning problem is the regression. To solve the problem different regression models were built such as Lasso, Ridge, SGD, LGMB, FM FTRL, etc. and the Ridge gave the best result.

3. METHODOLOGY

This chapter covers predictive models briefly. These models can be trained over time to respond to new data and they largely overlap with the field of machine learning. We mainly focused on regression models that estimate relationships among variables, and how they relate to each other.

3.1. OLS (LINEAR) REGRESSION

Simple Linear Regression is a statistical model that is frequently used in machine learning (ML) regression problems. It is based on the connection between two variables can be described by the following formula:

$$y_i = \alpha + \beta x_i + \varepsilon_i$$

Figure 1.1

When the goal of Simple Linear Regression is to identify the parameters and for which the error term is the smallest the process is called by OLS (Ordinary Least Squared). Indeed, the model will minimize squared errors.

$$\hat{\alpha} = \min_{\alpha} \sum_{i=1}^{n} (y_i - \alpha - \beta x_i)^2 = \min_{\alpha} \sum_{i=1}^{n} \varepsilon_i^2$$

$$\hat{\beta} = \min_{\beta} \sum_{i=1}^{n} (y_i - \alpha - \beta x_i)^2 = \min_{\beta} \sum_{i=1}^{n} \varepsilon_i^2$$

Figure 1.2

The least squares method is a statistical technique that is used to find the smallest OLS regression using the least squares approach. In the least squares approach, a linear trend formed by a scatterplot of data points is considered. An OLS linear regression technique generates a line of best fit, which is the most accurate approach to describe the distribution of data points with a single line. The least squares approach says that the line fit using the OLS technique will have the least sum of squared deviations from the line.

The OLS regression technique of analysis involves fitting a regression plane onto a "cloud" of data with a linear trend (Fox, 2015). The regression plane does not touch every point in the data cloud, but it does model the partial connections between each slope and the outcome variable while maintaining the effects of the other variables constant (Fox, 2015). Therefore, estimating regressions coefficients are made by minimizing the sum of squares of the difference between values fitted into the model and the values in the data.

3.2. RIDGE AND LASSO REGRESSIONS

The term multicollinearity refers to the correlation between independent variables in a regression model. Since independent variables should stay independent, high correlation between these variables is not ideal. In the case of multicollinearity, OLS estimates can still be obtained, but there will be room for improvement. It's because small variations like adding or removing a few observations will lead to significant changes in the coefficient estimates. Melkumova & Shatskikh (2017) mentioned that Ridge and LASSO regression analysis methods perform regularization of the estimated coefficients to overcome the pitfalls of the OLS estimator.

Most of the regression models contain a high number of predictors while many of them do not really affect the outcome. Excluding these variables from the model provides easier interpretation. Therefore, the Ridge and LASSO methods provide an alternative to "subset selection" techniques reducing the number of predictors in the regression model where the coefficient estimates are close to or equal to zero. (Melkumova & Shatskikh, 2017)

As mentioned in the previous topic (OLS), the least squares fitting model estimates regression coefficients (β0, β1, . . . , βp) using the values that minimize the sum of squared error (Residual Sum of Squares).

$$\mathrm{RSS} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 .$$

Figure 2.1

Even though ridge regression is similar to least squares regression, the coefficients are determined by minimizing a different quantity. The ridge regression coefficient estimates, in particular, are the values that minimize the equation below, where the tuning parameter λ (λ ≥ 0) must be calculated independently. (James et al., 2013)

$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 = \mathrm{RSS} + \lambda \sum_{j=1}^{p} \beta_j^2$$

Figure 2.2

Ridge regression, like least squares, aims to find coefficient estimates that are well-fitting to the data while keeping the residual sum of squares to at the minimum. However, the second term in the equation (next to RSS) is called a shrinkage penalty. When coefficient estimates are near to zero, this penalty is minimal, and it has the effect of reducing coefficient estimates towards zero.The tuning parameter λ is utilized to control the interaction between these two factors and the regression coefficient estimations. When λ = 0, the penalty component has no impact, and the least squares estimates are produced using ridge regression. On the other hand, as λ→∞, the shrinkage penalty becomes more significant, the ridge regression coefficient estimates

will approach zero. Unlike least squares regression, which yields a single set of coefficient estimates for each value, ridge regression yields a unique set of coefficient estimates for each value. (James et al., 2013)

There is one clear drawback to ridge regression. Unlike best subset, forward stepwise, and backward stepwise selection, which pick models that contain just a subset of the variables, ridge regression selects models that incorporate all predictors. The shrinkage penalty will reduce all of the coefficients to zero, but none of them will be set to zero exactly unless $\lambda = \infty$. This may not be an issue in terms of prediction accuracy, but it can make model interpretation difficult in situations where the number of variables p is high. (James et al., 2013)

The lasso is an alternative to ridge regression that overcomes this issue.

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j| \;= \mathrm{RSS} + \lambda\sum_{j=1}^{p}|\beta_j|$$

Figure 2.3

The lasso reduces the coefficient estimates towards zero in the same way that ridge regression does. When the tuning parameter is big enough, however, lasso's penalty factor has the effect of causing some of the coefficient estimations to be precisely equal to zero. As a result, lasso models are typically easier to interpret than ridge regression models.

3.3. EVALUATION METRICS

Evaluation metrics are critical to explaining how accurate the regression models are. In the following paragraphs, details of the R^2, Adjusted R^2, MSE, RMSE, and RMSLE will be explained.

R^2 cl and adjusted R^2 are statistical measures that show how close the data are to the fitted regression line. They simply represent the proportion of the variance for a dependent variable that's explained by the predictor variables in the sample (R^2) and an estimate in the population (adjusted R^2). (Miles, 2014)

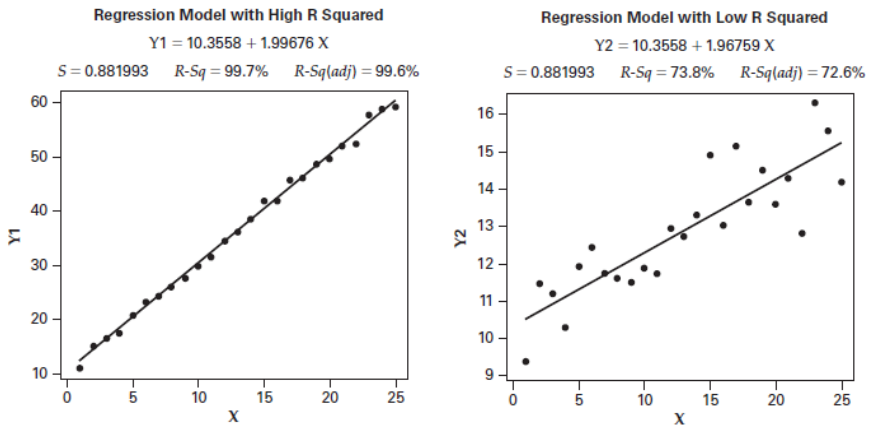

Figure 3.1 *Comparison of R² for Two Regression Models (Newbold et al, 2012)*

Their formulas are as follows:

$$\frac{\text{MSE(model)}}{\text{MSE(baseline)}} \qquad \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(\overline{y}_i - \hat{y}_i)^2}$$

Figure 3.2 *R-Squared Formula*

$$\bar{R}^2 = 1-\left(1-R^2\right)\left[\frac{n-1}{n-\left(k+1\right)}\right]$$

Figure 3.3 *Adjusted R-Squared Formula*

The best R² value is 1. R² values for endogenous latent variables can be described as substantial =0.26, moderate =0.13 and weak=0.02. (Cohen, 1988) When a new variable is added, R² always increases; and if one has enough variables, they will find that R² is equivalent to 1.00, but it does not explain the population (Miles, 2014). This indicates that R² has a limitation in the long run. This limitation can cause overfitting and return an unwarranted high R² value. To overcome this limitation, adjusted R² can be used because it penalizes you for adding variables that do not improve your model. In other words, in multivariate regression, we look at the adjusted R² value instead of R², because R² is often misleading. In the mathematical formula of adjusted R², n shows the sample size, and k represents the number of predictor variables (features) in the analysis. Adjusted R² is centered on 0 and can be negative, but R² is a proportion of variance, and it should not be negative. (Miles, 2014). When the sum-of-squares from the model is larger than the sum-of-squares from the horizontal line, a negative R² is observed, and it indicates the model does not fit the data well. Adjusted R² is always lower than the R².

Mean Square Error calculates the average squared difference between the estimated values and the actual value. The value of MSE is always greater than zero. The closer the predicted values are to the actual values, we obtain a smaller MSE; the further away from the actual values, we obtain a larger MSE value. A value close to zero represents a better regression model. The limitation of MSE is because of taking the square of a single bad prediction, the error can look worse than it is.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

Figure 3.4 MSE Formula

Root Mean Square Error (RMSE) is a measure of accuracy and the standard deviation of the residuals. It is the square root of MSE. Residuals show how far from the regression line data points are, and RMSE calculates how spread out these residuals are and tells you how concentrated the data is around the best fit line. RMSE can range from 0 to ∞ and lower values specify better accuracy. Because the errors are squared before they are averaged, RMSE is highly affected by the outlier values.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(Predicted_i - Actual_i)^2}{N}}$$

Figure 3.5 RMSE Formula

RMSLE is the RMSE of the log-transformed predicted and log-transformed actual values. It is accurate to use RMSLE when there is a wide range in the target variables and, we don't want to penalize these big differences when both the predicted and the actual are big numbers.

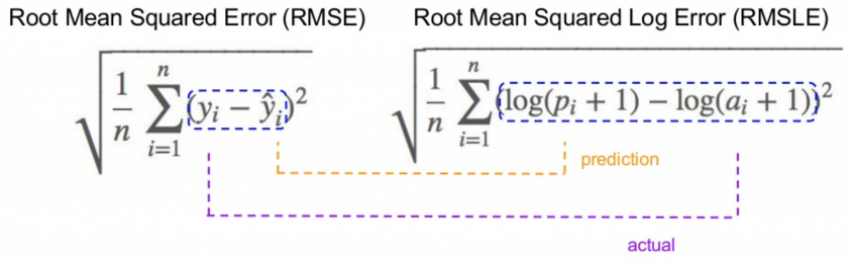

Figure 3.6 RMSE and RMSLE Formula Comparison

There is no universally good RMSE or RMSLE value. The goodness of RMSE and RMSLE are highly dependent on the range of values in the dataset that is used in the model. Comparing the RMSE and RMSLE values for different models and choosing the model with the lowest error is the key to reach the best model.

4. STATEMENT OF THE PROBLEM

In this chapter, the aim of study, dataset, data preprocessing and predicting accuracy evaluation are introduced. The study is carried out using Python software language and its Pandas, Numpy, Seaborn, Sklearn, Matplotlib, Statsmodels, Scipy and Wordcloud libraries for various data science tasks on the Jupyter Notebook application which is an open source development environment.

4.1. Problem Definition

Considering how many items are sold online, product pricing becomes much more difficult. Some products' costs follow significant seasonal patterns and are largely affected by brand names, whereas some products' costs fluctuate based on product specifications. Most of the online marketplaces allow their sellers to price their products without any restriction. On the other hand, some marketplaces have control mechanisms to prevent overpricing of certain products. However, this is a more serious problem for second-hand shopping platforms. Since second-hand platforms have a C2C business model, their sellers may list almost any product. Moreover, most of the products are unique, or somewhat different in terms of condition, quality etc. Therefore, controls on pricing in second-hand platforms are more challenging. To understand how these platforms cope with this problem, an interview was conducted with one of the Data Scientists at Turkey's one of the largest community-powered second hand shopping platforms.

According to this interview, their platform suggests a price for their sellers in the product submission page. This price recommendation is made using a markup percentage based on the original price of the product. However, this markup is predetermined so, it neither fully reflects the platform's selling behavior nor is the optimum selling price for the platform's health.

This study aims to make better price recommendations to sellers to improve the platform's health by giving sellers more accurate suggestions based on sold products using regression models so that both buyers and sellers are satisfied with the product pricing.

Projects steps include;

- Understanding established projects for price recommendation
- Conducting Literature Review
- Collecting data
- Preprocessing data
- Applying regression models
- Selecting the optimum model for suggestion by using evaluation metrics

4.2. Data Description

Data of this project is collected from a second-hand e-commerce platform in Turkey with the help of their Data Analytics team. The data contains information about sold products between 16th of February and 16th of June of 2021. There are a total of 2.556.419 rows meaning unique sold products and 18 columns. 5 columns are excluded from this project since four of them were ID's of sellers, products, categories and brands and the other one was product status which is 'SOLD' in all the rows in the dataset.

The table containing description about each column and corresponding data type is shown below;

Table 1. Data Explanation

Field Name	Type	Description
Product ID	Integer	Unique identifier of a product
Seller ID	Integer	Unique identifier of a seller
Brand ID	Integer	Unique identifier of a brand
Brand Type	String	The type of the brand ('POPULAR', 'LUX', 'ECONOMICAL', 'ULTRA_LUX')
Brand Title	String	The brand of a products among 6689 brands
Category ID	Integer	Unique identifier of a category
Condition	String	Condition of a product ('NEW_WITH_TAGS', 'LIKE_NEW', 'GENTLY_WORN')
Shipment Term	String	Shipment/cargo is whether paid by seller ('SELLER_PAYS') or paid by buyer ('BUYER_PAYS')
Product Status	String	Status of a product (The dataset only contains products with 'SOLD' status)
Star	Float	Average feedback score of sellers between 1 and 5, null if no feedback is given
Description	String	Description of a product entered by sellers
Zero Level Title	String	5 main categories: Women, Men, Home & Life, Kids & Baby, Electronics
First Level Title	String	37 first level sub-categories such as: Clothing, Bags, Cosmetics etc.
Second Level Title	String	315 second level sub-categories such as: Bottom

		Clothing, Outerwear, Underwear etc.
Third Level Title	String	532 third level sub-categories such as: Coats, Jackets, Shorts, Trousers etc. (if third level category not available, second level category is given at this field)
Original Price	Float	Original (store bought) price of a product, null if the original price is not available
Price	Float	Listing price of a product
Sold Price	Float	Selling price of a product, any discount is reduced

4.3. Exploratory Data Analysis — EDA

Before applying predictive models, it is significant to perform initial investigations on data. In the Exploratory Data Analysis chapter, firstly, the distribution of the sold price is analyzed. As can be seen below, the histogram of the sold price is right-skewed which indicates that the mean is greater than the median. Most of the products' sold prices are in the range of 0-250TL.

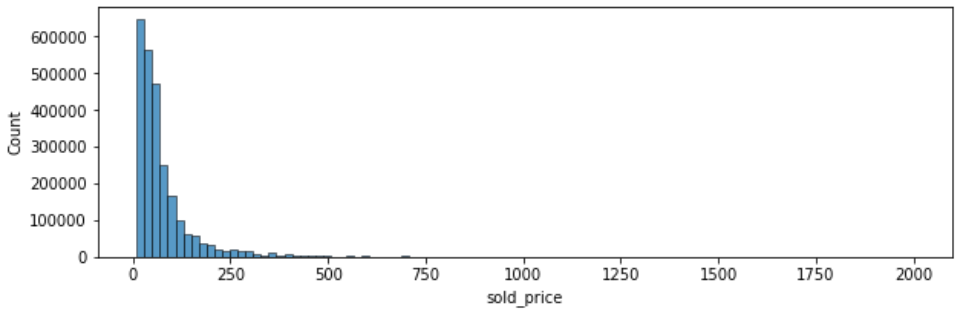

Figure 4.1 Distribution of Sold Price

To remove the skewness of the data, log transformation is applied to sold_price.

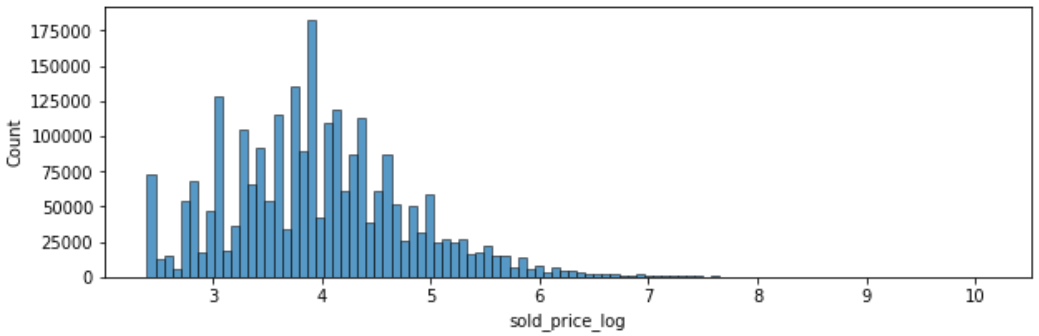

Figure 4.2 Distribution of Log of Sold Price

How the sold price changed when the seller or the buyer paid for the shipment is examined. As expected, product prices were higher when the seller paid for the shipment.

Figure 4.3 Shipment Term - Sold Price Comparison

The data has 6689 unique brands and the top selling brands are reviewed. Below chart shows that Zara is the first in this ranking and has approximately 175.000 sales for four months.

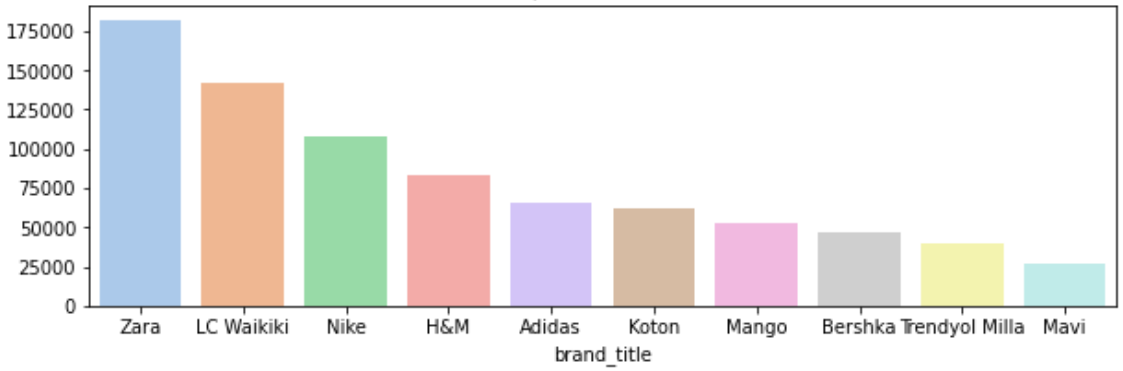

Figure 4.4 Top Seller Brands

The average sold price of each brand is calculated and the top 10 expensive brands are identified. While doing this study, only the brands that have at least 10 products are considered.

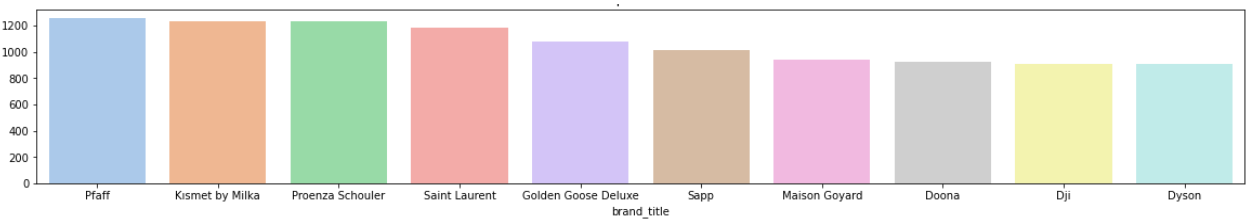

Figure 4.5 Most Expensive Brands

Word Clouds are a collection of words shown as a cloud. If a word becomes larger in the text, the more frequently it is used in the text. It can be determined by the large terms and hence the top subjects just by glancing at the cloud. Jupyter has a simple method to run command line commands from within the notebook, namely "wordcloud".

Matplotlib, pandas and wordcloud modules are used in this study to create a word cloud. Before highlighting significant textual data points in the item description column by using the word cloud function, the data needs to be cleaned (text preprocess). Some functions that check and delete the punctuations and insignificant words in the data are created.

Stop words in the data are not desired since only meaningful data should be included in the word cloud. Stopwords in Turkish have been eliminated by dividing each text in the item description section into words and then removing the word if it exists in the list of stop words. In addition, it is important to delete stopwords that are not useful for the regression models to improve process time. The following codes display the logic behind how the stopwords, punctuations and spaces are deleted and how the word cloud is built:

```
def text_preprocess(value, stopwords):
value = value.replace('\\r', ' ')
value = value.replace('\\"', ' ')
value = value.replace('\\n', ' ')
value = ' '.join(e for e in value.split() if e not in stopwords)
return value.lower().strip()

def generate_word_cloud(series):
df['description'] = df['description'].astype(str)
with open('stopwords.txt', 'r') as f:
stopwords = [word.strip() for word in f.readlines()]
wordcloud = WordCloud(collocations=False).generate(' '.join(text_preprocess(value,
stopwords) for value in series))
plt.figure(figsize = (12,6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.title(f'WordCloud for {series.name}')
plt.show()
generate_word_cloud(df['description'])
```

In the below image, it can be seen that most of the sellers are mentioning the brand names to attract customers. Additionally, most of the sellers are writing the words that describe the condition of the product.

Figure 4.6 Wordcloud for description

4.4. Data Preprocess

Data preparation is the process of converting raw data into a usable format. Because real-world data is used , it is possible to include numerous mistakes. However, the data that is used for this project was mostly clear, understandable, and does not include missing, noisy or inconsistent data. So, elimination of outliers was enough.

4.4.1. DATA CLEANING

In data science, insights and results are only as great as the data. Therefore, before starting to create our model we expect to have complete, correct, accurate, and relevant data. Four types of data can lead to result in erroneous predictions:

1. **Missing Data:** when no data value is stored for the variable.
2. **Noisy Data:** meaningless, error data.
3. **Inconsistent Data:** mismatch in duplicate records. The same data exists in different tables and their formats don't match.
4. **Intentional Problems:** Intentionally giving a value such as if the birthday is unknown, 1st of January can be assigned as value.

Our data was extracted from the database of the collaborated company, and while extracting, most of the missing values were already eliminated by building right queries in the Google Cloud Platform, Big Query. For instance, while taking the data, missing values of the third_level_title are replaced with the value of the second_level_title. Later, in Python, 83.092 duplicate rows are deleted in product_id. The noisy data, inconsistent data and intentional problems are not encountered in this study. Since it is important that the third_level_title is unique in the model that will be applied, the

"other" category is excluded. By doing this, in the modelling phase, a rise of r squared is observed.

4.4.2. OUTLIERS

Outlier is the data point in a dataset that's considerably distinct from the rest. Deciding what constitutes an outlier may be somewhat subjective, depending on the research and the amount of data collected even though the definition of outlier may appear basic. Visualization and mathematical functions are some of the ways to detect outliers.

Box plot and scatter plot are the most common visualization methods to detect outliers. The box plot is a graphical representation of numerical data groupings by their quartiles. Points that do not belong to the box can be plotted to represent outliers. The scatter plot is a form of plot or mathematical diagram that displays values for generally two variables for a collection of data using Cartesian coordinates. The data is represented as a series of dots, with the value of one variable determining the horizontal axis position and the value of the other variable determining the vertical axis position. The dots which are located apart from the population may be labelled as outlier.

IQR scores is one of the ways of using mathematical functions to eliminate outliers. The IQR approach is used by Box Plot to display data and outliers (data shape), but in order to receive a list of detected outliers, we must utilize a mathematical calculation and extract the outlier data. The IQR is the first quartile minus the third quartile; these quartiles can be seen on a box plot. It shows how the data is spread about the median.

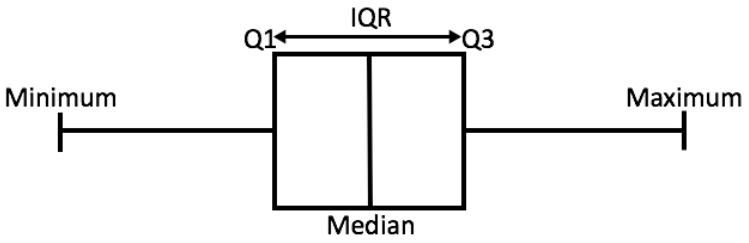

Figure 4.7 Box Plot

Our data had outliers too and after detecting these outliers with some of the methodologies that are mentioned above, the outliers should be transformed or removed from the data. First of all, the following function is created to see the outliers on the boxplot in Python.

```
def show_outliers(series):
sns.set(rc={'figure.figsize':(9,6)})
sns.boxplot(series, kde_kws = {"color" : "red"})
plt.title('Outliers')
plt.show()
```

By running this function the box plot which is used for graphically depicting groups of data through their quartiles is created.

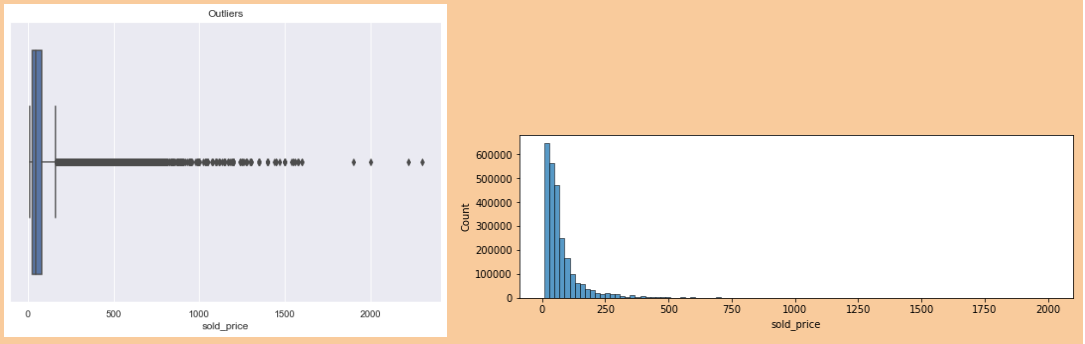

Figure 4.8 The box plot and histogram of sold price

As can be understood, the range of the sold price in our data is very wide (10TL to 25.000TL with the mean of 82TL) and the frequency in the high prices is low. Only the data that lies within lower and upper limits are statistically considered normal and to find the exact upper and lower limits, an IQR approach is used. The function below is created for detecting outliers by considering the interquartile range formula.

```
def is_outlier(s):
quartile_1 = np.round(s.quantile(0.25), 2)
quartile_3 = np.round(s.quantile(0.75), 2)
iqr = np.round(quartile_3 - quartile_1, 2)
lower_th = quartile_1 - (iqr * 1.5)
higher_th = quartile_3 + (iqr * 1.5)
return ~s.between(lower_th, higher_th)
```

Since we will do a study by grouping based on the third-level category, the outlier elimination was done in these small data groups instead of doing it in the whole data. In this way, we aimed to make estimates for this group by calculating a price average at the lowest category level, that is, at the third level, which is not affected by outliers. In the following code, the function that extreme values are named as abnormal and discarded from the entire series can be seen.

```
def remove_outliers(df, by = 'sold_price', filtered_column = 'thirdlevel_title_'):
copy_df = df.copy()
return copy_df[~copy_df.groupby(filtered_column)[by].apply(is_outlier)]
```

At the same time, grouping was also done according to brand title and outliers in these groups were eliminated. The studies carried out according to both the third level title and the brand title were evaluated in the next stages by examining the error metrics of the regression models.

4.4.3. CORRELATION

Correlation is a statistical metric that determines how closely two or more variables fluctuate. In basic words, it shows us how much one variable varies when another variable is changed slightly (Wijaya, 2021). Depending on the direction of the shift, it might take positive, negative, or zero values. When a dependent variable and an independent variable have a high correlation value, it means that the independent variable is highly significant in affecting the output. To create a more feasible model with greater accuracy in a multiple regression setting with numerous components, it is critical to identify the correlation between the dependent and all of the independent variables. It's important to note that having more features doesn't automatically mean greater accuracy. If they contain any irrelevant features that create unneeded noise in the model, adding more features may result in a decrease in accuracy (Wijaya, 2021).

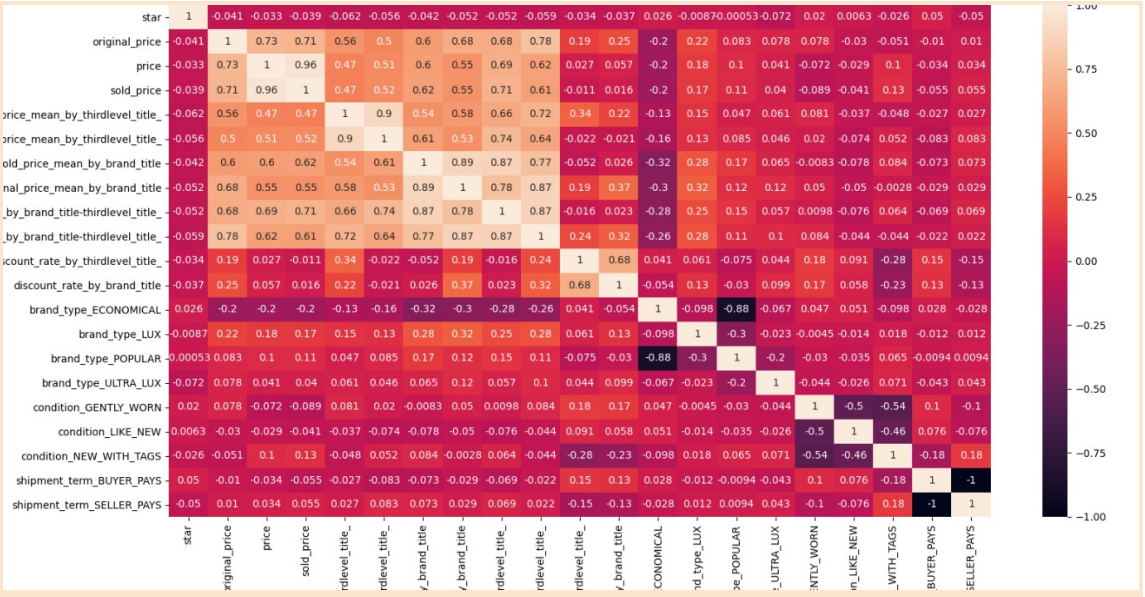

Figure 4.9

As it can be seen in the correlation matrix, there is weak negative correlation between star and sold_price (- 0.039). Thus, the variable of star is not included in the model. The other continuous variables, price and original_price have quite a stronger correlation with sold price. Since the price variable corresponds to the listing price of the product, it is not included in the model. Accordingly, sold and original price means by brand and third level titles has strong positive correlation between the sold price variable, so they are included in the model. Also, discount_rate_by_brand_title has a positive correlation between the sold price. The other variables are dummy variables, so they are not considered.

4.4.4. NORMALIZATION

The assumption of the normal distribution is essential for the applicability of parametric tests in data science. Normalization adjusts the values to a common scale without changing the range. In case a variable does not show a normal distribution (Gaussian),

there are various transformation techniques to apply. Some of the transformation techniques are as follows.

In square root transformation, the square root of each variable should be taken. If there are negative numbers, a constant should be added to each variable to make them all positive. The main advantage of the square root transformation is that it can be applied to zero values.

In log transformation, each variable of X is replaced by log(x) with base 10, base 2, or natural log. The logarithm of a number less than 0 is undefined, and like square root transformations, variables that contain values between 0 and 1 are treated differently than those above 1. (Sakia, 1992)

The Box-Cox transform is a statistical technique that has a highly corrective effect on skewed data. The function determines the conversion type of non-normal distribution to normal distribution according to the lambda parameter which is between 5 and -5 and the lambda shows the power to which all values should be raised. (Buthmann, 2018) The optimal value for lambda is the one that gives the best approximation for the normal distribution. When the optimal value for lambda equals 1, then the data is already Gaussian, and the Box-Cox transformation is not necessary. (Plummer, 2020) The Box-Cox Power transformation is suitable for the positive data, therefore if the data has negative values a constraint should be added. (Buthmann, 2018)

$$y(\lambda) = \begin{cases} \frac{y^{\lambda}-1}{\lambda}, & \text{if } \lambda \neq 0; \\ \log y, & \text{if } \lambda = 0. \end{cases}$$

The Box-Cox formula for the positive Y values

In this study three continuous variables (original price, sold price and item description word count) exist. Other continuous variables are also derived using these variables. The discount rate is calculated using the formula (Original Price - Sold Price) / Original Price. Sold price and original price averages were calculated separately in the brand title and third-level title groups. As a result Log and Box Cox transformations are applied to “sold_price_mean_by_thirdlevel_title_”, “original_price”, “original_price_mean_by_thirdlevel_title_”, “original_price_mean_by_brand_title”, “sold_price_mean_by_brand_title” columns. The normalization effect on evaluation metrics is examined in the modelling part.

4.4.5 CATEGORICAL VARIABLE ENCODING

One of the important parts in feature engineering is turning categorical variables into numerical variables. Because many machine learning algorithms cannot operate directly on categorical data, they require all input variables and output variables to be numeric. This means that categorical data must be converted to a numerical form. So, there are several methods for this conversion such as one hot encoding, label encoding, dummy

encoding etc. (Potdar et al., 2017). This project is focused on two methods: one hot encoding and dummy encoding.

One hot encoding is one of the popular methods for treating categorical variables. It is the process of creating dummy variables that are variables containing values such as 1 or 0 representing the presence or absence of the categorical value (Seger, 2018). Simply, it creates additional features based on the number of unique values in the categorical feature. Every unique value in the category will be added as a feature.

By including dummy variables in a regression model however, one should be careful of the Dummy Variable Trap. The Dummy Variable trap is a scenario in which the independent variables are multicollinear - a scenario in which two or more variables are highly correlated; in simple terms one variable can be predicted from the others (Cope, 2018).

Take the situation of gender as an illustration of the Dummy Variable Trap. Although it is unnecessary to include a dummy variable for each (if male is 0, female is 1, and vice versa), doing so will result in the following linear model:

```
y ~ b + {0|1} male + {0|1} female
```

The sum of all category dummy variables in the given model equals the row's intercept value - in other words, there is perfect multicollinearity (one value can be predicted from the other values). Intuitively, there is a duplication category: if the male category is removed, the female category is automatically defined (zero female value indicates male, and vice-versa).

The dummy variable trap can be avoided by dropping one of the categorical variables. If there are x categories, use x-1 in the model, the value left out can be thought of as the reference value, and the fit values of the remaining categories represent the change from this reference. This technique is dummy variable coding, often known as dummy encoding (Mahto, 2019). Even though dummy variable coding comes from the statistics field and one-hot encoding comes from the computer science field, they are exactly the same thing. The dummy variable trap can arise no matter what you name the technique, but it's less of a problem in machine learning since multicollinearity is only a problem in linear regression and can be reduced by regularization as mentioned in the Ridge and Lasso Regressions part. On the other hand, in statistics, the goal is to find the coefficients but in machine learning, the goal is to find the predictions, thus multicollinearity is a secondary concern.

In this study, dummy variables are created from the 3 categorical variables: brand type, condition and shipment term. Moreover, dummy variables created from third level category titles. However, there were over 500 dummy variables, so it was eventually abandoned for the model performance and interpretability.

5. RESULTS

In this study, different regression models, normalization techniques, outlier deletion applications and feature engineering techniques are applied to create a model to accurately suggest a product's selling price. The data is divided into two that are called test and train. A common split percentage of 20% for test data and 80% for train data is used for all the regression models developed. In this way, the performance of the model can be evaluated.

There were different possible combinations to create well performing models. Therefore, some functions are created using Python to run models easily and to be more flexible while generating these models. This way, normalizing and cleaning data, generating variables, creating models and calculating the evaluation metrics were possible just by changing and running a few lines of code.

The below table represents 18 of the best performing models. Every number on the table corresponds to a different model. Details of these models can be found in Appendix, Table 2.2, 2.3 and 2.4.

#	model	r2	adjusted r2	mse	rmse	rmsle
1	ridge	0,648	0,648	1.508,693	38,842	0,495
2	lasso	0,634	0,634	1.485,364	38,540	0,491
3	linear	0,632	0,632	1.503,755	38,778	0,497
4	linear	0,618	0,618	703,949	26,532	0,458
5	ridge	0,618	0,618	691,463	26,296	0,458
6	lasso	0,612	0,612	706,801	26,586	0,459
7	linear	0,650	0,650	645,209	25,401	0,440
8	ridge	0,647	0,647	646,375	25,424	0,441
9	lasso	0,644	0,644	659,705	25,685	0,439
10	linear	0,512	0,512	890,608	29,843	0,583
11	ridge	0,505	0,505	1.049,059	32,389	0,522
12	ridge	0,638	0,638	1.372,390	37,458	0,495
13	linear	0,534	0,534	864,681	29,405	0,588
14	linear	0,649	0,649	644,486	25,387	0,441
15	lasso	0,650	0,650	643,603	25,369	0,437
16	**ridge**	**0,657**	**0,657**	**633,367**	**25,167**	**0,439**
17	ridge(λ = 0)	0,652	0,652	642,774	25,353	0,440
18	ridge(λ = 0.5)	0,650	0,650	654,865	25,590	0,439

Table 2.1

Creation of the models above can be summarized as follows;

First of all, outliers are eliminated from the data. As mentioned earlier, outliers are eliminated according to groups. Two continuous variables (sold and original price) are grouped according to two categorical variables (third level title and brand) either simultaneously or seperate. After that, using IQR, the outliers are detected and eliminated. Between 150,000 and 450,000 rows of data excluded from the model depending on the combination used. Since there were almost 2 millions of rows remaining, the best performing model included all the 4 combinations. (Model 16 is highlighted on the above table)

For normalization, two techniques are used: log and box cox transformation. Log transformation performed better than box cox transformation with higher R^2 and lower RMSE (Models 10 - 13). However, in the best performing model, normalization is not applied.

As mentioned earlier, dummy variables are created from 3 categorical variables using one hot encoding. Moreover, new continuous variables are created from the existing variables; discount rates means, original price means and sold price means according to brand and third level title. All or some combination of these variables are included in the models with the already existing variables. Best performing model included all these variables.

Models are created based on 3 regression models: Linear (OLS), Ridge and Lasso. Ridge and Lasso models performed regularization and an alternative to subset selection. From the theoretical knowledge, it was expected Ridge regressions to have higher prediction accuracy then the Lasso regressions.

As shown in the above table, the best performing model (Model 16) is a Ridge regression and it performed the best R^2 and lowest RMSE value. After obtaining the best score in the ridge model, even better results were sought by changing the alpha value, but it was observed that the best value was the default value of 1. In this Ridge model, original and sold price outliers in both third level and brand title groups were determined and eliminated according to the IQR approach. Moreover, 8 continuous variables and 9 categorical variables were inserted into the model. While using the mean of the original price and sold price in different groups, discount rate, etc. as continuous variables; the product's status, shipment term, brand type in dummy format are used as categorical variables. As mentioned earlier, normalization (log and box cox transformations) was not performed in this model.

RMSE score of 25 for the Model 16 means that the model is predicting the sold price with +/- 25TL error rate. On the other hand, R^2 of 66% means the model explains 66% of the data.

To demonstrate a possible application of the model, a user interface is created. Users can simply enter the category, brand, brand type, shipment term, condition and original price of the product they want to sell and they can click the enter button. A recommended price will be generated by putting these inputs into the selected model.

This study will help sellers set reasonable and effective prices for their second-hand products without doing any market research. Alam (2015) states that the customer perceives pricing to be the most important element in deciding whether or not to purchase used goods and second-hand product buying intentions are heavily influenced by price. Therefore, sales rates and profit will increase on the e-commerce platform thanks to the right price suggestion. With the implementation of our model into a real platform, the platform's health is expected to be improved due to more accurate price recommendations to sellers based on sold items, ensuring that both consumers and sellers are pleased with product pricing.

6. CONCLUSION & FUTURE WORK

The project of suggesting prices to the sellers for a second-hand platform was a regression problem which required handling a large amount of data, approximately 2.5 million rows, and 13 columns. There were both continuous and categorical variables within the data. Because our data was mostly clean and understandable, the data preprocessing of the project was effortless. Even though the data was clean, understanding and analyzing the data was quite challenging. However, the most challenging part of the project was handling original prices and the categorical data. Because users of the platform are able to enter any price they want as the original price, it required handling outliers. Also, including categorical variables into a regression problem required a lot of research since there were 6000 brands and 500 product categories. Therefore, building a model to predict prices for the specific product details in all categories was quite challenging. After creating new variables, eliminating outliers and performing normalization on the data, a regression model with good performances was built.

The project has encountered some other difficulties as well. Since we needed to create a project as a group without meeting during the pandemic period, we tried to use cloud-based web tools, but we experienced both performance and speed problems due to the size of the data we have. There were also CPU issues since each team member had different kinds of hardware. In addition, while coding in Python, many functions had to be created by team members because the Python library did not meet the needs or trying different types of models with different combinations of techniques were time consuming.

According to information that is gathered from the literature, adding an image processing to predict prices for different categories may also be beneficial. Analyzing the quality of the image and the product, extracting the brand and the condition of the

product may increase the model accuracy because there is no need for the user input in that model except the image of the product.

In further studies, gathering more data can train the model developed in this study better, thus can lead to more accurate predictions. Moreover, there may be additional variables that can be used as predictors depending on the platform using the model.

On the other hand, Elastic Net regression which includes a combination of regularization techniques of Ridge and Lasso can be used to improve accuracy. Also, other machine learning applications could be further examined to understand possible improvements in the prediction such as Neural Network. These other techniques could make significant changes on the overall result of the project.

To sum up, we expect the model of this study to be used in a second-hand e-commerce platform to suggest prices for sellers to list their products. Sellers who price their products based on the model's recommendation, can increase their sales rates. Since the recommended price is an optimum price within +/- 25 TL margin, buyers are expected to be satisfied with the price and the platform can benefit from this in terms of revenue.

7. REFERENCES

1. Acharya, T. (2020, March 31). Understanding Feature extraction using Correlation Matrix and Scatter Plots. Medium. https://towardsdatascience.com/understanding-feature-extraction-using-correlation-matrix-and-scatter-plots-6c19e968a60c
2. Alam, M. D. (2015). Factors that Influence the decision when buying second-hand products.
3. Buthmann, A. (2018, February 17). Making Data Normal Using Box-Cox Power Transformation. ISixSigma. https://www.isixsigma.com/tools-templates/normality/making-data-normal-using-box-cox-power-transformation/
4. Cohen, J. 1988. Statistical Power Analysis for the Behavioral Sciences, 2nd Edition. Routledge
5. Cope, G. (2018). Dummy Variable Trap in Regression Models. Algosome Software Design Copyright 2009 Greg Cope All Rights Reserved. https://www.algosome.com/articles/dummy-variable-trap-regression.html
6. Dave, C. (2021, June 19). Mercari Price Suggestion Challenge — An End-to-End Machine Learning Case Study. Medium. https://medium.com/swlh/mercari-price-suggestion-challenge-an-end-to-end-machine-learning-case-study-4a6d833fa1c7
7. Han, L., Yin, Z., Xia, Z., Guo, L., Tang, M., & Jin, R. (2019). Vision-based Price Suggestion for Online Second-hand Items. Session 4C: Social Computing & Image Processing.

8. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning: with Applications in R (Springer Texts in Statistics) (1st ed. 2013, Corr. 7th printing 2017 ed.). Springer.
9. Kleiber, C. (2001). Finite sample efficiency of OLS in linear regression models with long-memory disturbances. Economics Letters, 72(2), 131-136. doi: 10.1016/s0165-1765(01)00423-2
10. Mahto, K. K. (2019, July 20). One-Hot-Encoding, Multicollinearity and the Dummy Variable Trap. Medium. https://towardsdatascience.com/one-hot-encoding-multicollinearity-and-the-dummy-variable-trap-b5840be3c41a
11. Malik, F. (2019, June 11). Understanding Value Of Correlations In Data Science Projects. Medium. https://medium.com/fintechexplained/did-you-know-the-importance-of-finding-correlations-in-data-science-1fa3943debc2
12. Melkumova, L., & Shatskikh, S. (2017). Comparing Ridge and LASSO estimators for data analysis. Procedia Engineering, 201, 746–755. https://doi.org/10.1016/j.proeng.2017.09.615
13. Mercari Price Suggestion Challenge. (2018). Kaggle. https://www.kaggle.com/c/mercari-price-suggestion-challenge
14. Miles, J. (2014). R squared, adjusted R squared. Wiley StatsRef: Statistics Reference Online.
15. Newbold, P., Carlson, W. L., & Thorne, B. (2013). Statistics for business and economics. Boston, MA: Pearson.
16. Ordinary Least Squares regression (OLS). (2021). Retrieved 1 July 2021, from https://www.xlstat.com/en/solutions/features/ordinary-least-squares-regression-ols
17. Plummer, A. (2020, September 7). Box-Cox Transformation: Explained - Towards Data Science. Medium. https://towardsdatascience.com/box-cox-transformation-explained-51d745e34203
18. Potdar, K., S., T., & D., C. (2017). A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers. International Journal of Computer Applications, 175(4), 7–9. https://doi.org/10.5120/ijca2017915495
19. Sakia, R. M. (1992). The Box-Cox Transformation Technique: A Review. The Statistician, 41(2), 169. https://doi.org/10.2307/2348250
20. Seger, C. S. (2018). An investigation of categorical variable encoding techniques in machine learning: Binary versus one-hot and feature hashing. DEGREE PROJECT IN TECHNOLOGY. Published.
21. Simple Linear Regression with Python. (2021). Retrieved 1 July 2021, from https://medium.datadriveninvestor.com/simple-linear-regression-with-python-1b028386e5cd
22. Wang, C., & Wu, H. (2018). A new machine learning approach to house price estimation. New Trends In Mathematical Science, 4(6), 165-171. doi: 10.20852/ntmsci.2018.327
23. Wijaya, C. Y. (2021, February 14). Correlation for data science | Towards Data

Science. Medium.
https://towardsdatascience.com/what-it-takes-to-be-correlated-ce41ad0d8d7f

APPENDIX A: Interview With A Data Scientist Of A Secondhand Ecommerce Platform

Görkem Çoklar: Hi, thank you for taking the time for us. To start with, Can you talk about your role at your current company?

TC: Yes, of course. I am currently working as a data scientist at one of Turkey's biggest e-commerce companies. My team is responsible for its second-hand shopping platform. I am currently working on improving the algorithm of the product listing.

Görkem Çoklar: That is great. So, the reason we have contacted you was to learn about your product price recommendation system. As a user of your platform, I have seen you give price suggestions at the product submission page. Can you give us some detail about how this system works and what it is based on?

TC: Our current product pricing system is based on the original price that the user enters. There is a predefined markup percentage and we suggest a product listing price based on discounting this percentage on the original price of the product. It is currently based on a very simple model.

Görkem Çoklar: Well, we were actually expecting a more complex system. Do you have any feedback from the users regarding the system's suggestions?

TC: Yes, we know that our current suggestions are not very useful since a significant amount of them do not price their products according to our suggestions.

Görkem Çoklar: Do you plan to improve your recommendation system?

TC: Yes, we are planning to create a more complex system based on product's category and the category's variance. However, we are waiting for the developments from our tech team.

Görkem Çoklar: That is great. Thank you for your time.

TC: You are welcome.

APPENDIX B: Python Code

1. Data Cleaning

```
import
logging
            from typing import List, Callable


            import numpy as np
            import pandas as pd



            logger = logging.getLogger(__name__)



            def interquantile_outlier(s: pd.Series):
               quartile_1 = np.round(s.quantile(0.25), 2)
               quartile_3 = np.round(s.quantile(0.75), 2)
               iqr = np.round(quartile_3 - quartile_1, 2)
               lower_th = quartile_1 - (iqr * 1.5)
               higher_th = quartile_3 + (iqr * 1.5)


               return ~s.between(lower_th, higher_th)



            def remove_duplicates(df: pd.DataFrame, columns: List[str]):
               removed_df = df.copy().drop_duplicates(columns)
```

```
    logger.info(f'Removed {df.shape[0] - removed_df.shape[0]}
duplicate rows')
    return removed_df




def remove_others(df: pd.DataFrame):
    new_df = df.copy()[df['thirdlevel_title_'] != 'Diğer']


    logger.info(f'Removed {df.shape[0] - new_df.shape[0]} other
rows')
    return new_df




def remove_missing(df: pd.DataFrame, columns: List[str]):
    removed_df = df.copy().dropna(axis=0, subset=columns)


    logger.info(f'Removed {df.shape[0] - removed_df.shape[0]}
missing rows')
    return removed_df




def remove_outliers_with_groups(
    df: pd.DataFrame, column: str, by: str, outlier_fn:
Callable[[pd.Series], pd.Series]
):
    copy_df = df.copy()


    removed_df =
copy_df[~copy_df.groupby(by)[column].apply(outlier_fn)]


    logger.info(f'Removed {df.shape[0] - removed_df.shape[0]}
outlier rows')
```

```
        return removed_df
```

2. Feature Creation

```
from typing
    import List


                        import numpy as np
                        import pandas as pd




                        def group_mean(df: pd.DataFrame, column: str,
                            by: List[str], suffix: str = 'mean'):
                            new_col_name =
                            f'{column}_{suffix}_by_{"-".join(by)}'


                            copy_df = df.copy()
                            copy_df[new_col_name] =
                            df.groupby(by)[column].transform(np.mean)


                            return copy_df




                        def discount_rate(df: pd.DataFrame, by: str):
                            copy_df = df.copy()


                            new_col_name = f'discount_rate_by_{by}'


                            copy_df[new_col_name] =
                            (df[f'original_price_mean_by_{by}'] -
                            df[f'sold_price_mean_by_{by}']) / df[
                                f'original_price_mean_by_{by}'
                            ]
                            return copy_df
```

```
def word_count(df: pd.DataFrame, column: str,
    suffix: str = 'wc'):
    copy_df = df.copy()

    copy_df[column] = copy_df[column].astype(str)

    new_col_name = f'{column}_{suffix}'

    copy_df[new_col_name] =
    copy_df[column].apply(lambda val:
    len(val.split()))

    return copy_df
```

3. Main

```
import
   argparse
                    import logging
                    import pprint
                    import sys
                    from typing import List


                    import pandas as pd
                    import seaborn as sns
                    from matplotlib import pyplot as plt
                    from sklearn.linear_model import LinearRegression, Ridge,
                       Lasso, ElasticNet


                    from metrics import ModelSummary
                    from data_cleaning import (
                       remove_missing,
                       remove_duplicates,
                       remove_outliers_with_groups,
                       interquantile_outlier,
```

```
    remove_others,
)
from transformation import boxcox_transform,
    one_hot_encoding, log_transform
from feature_creation import group_mean, discount_rate,
    word_count
from model_builder import build_model
from utils import compose_functions, timer


logging.basicConfig(stream=sys.stderr, format='%(asctime)s -
    %(name)s - %(levelname)s - %(message)s',
                    level=logging.DEBUG)
logger = logging.getLogger(__name__)




parser = argparse.ArgumentParser()
parser.add_argument('--name', '-n', type=str, help='Model
    name', default='Test')
parser.add_argument('--rows', '-r', type=int, help='Number of
    rows to use from dataset')
parser.add_argument('--file', '-f', type=open, help='Data
    source file name', default='data/data.csv')
parser.add_argument('--save', '-s', action='store_true',
    help='Save the generated model', default=True)




def create_processor(processor, params):
    pr_logger = logging.getLogger('processors')


    def f(df):
        with timer(processor.__name__, pr_logger, params):
            return processor(df, **params)


    return f
```

```
def run_flow(
    name, df, *args, X: List[str], y: str, model_factory,
    model_kwargs=None, exclude_X=False
):
    functions = []

    for function, params in args:
        functions.append(create_processor(function, params))

    process_data = compose_functions(*reversed(functions))

    processed_data = process_data(df)

    if model_kwargs is None:
        model_kwargs = {}

    model, X_test, y_test = build_model(
        processed_data,
        model_factory,
        X,
        y,
        model_kwargs=model_kwargs,
        exclude_X=exclude_X,
    )

    model_summary = ModelSummary(name, model, X_test, y_test)
    return model_summary


def show_correlation_heatmap(df):
```

```
    corr_matrix = df.corr()


    fig, ax = plt.subplots(figsize=(18, 9))
    ax.set_title('Correlation Matrix', fontsize=18)
    sns.heatmap(corr_matrix, annot=True, ax=ax)
    plt.show()



if __name__ == '__main__':
    args = parser.parse_args()


    logger.info(f'Reading {args.file.name}...')
    df = pd.read_csv(args.file, nrows=args.rows)
    logger.info(f'Found {df.shape[0]} rows')


    model_summary = run_flow(
        args.name,
        df,
        (remove_duplicates, {'columns': ['product_id']}),
        (remove_missing, {'columns': ['thirdlevel_title_',
    'original_price']}),
        (remove_others, {}),
        (
            remove_outliers_with_groups,
            {
                'column': 'sold_price',
                'by': 'thirdlevel_title_',
                'outlier_fn': interquantile_outlier,
            },
        ),
        (
            remove_outliers_with_groups,
            {
```

```
            'column': 'original_price',
            'by': 'thirdlevel_title_',
            'outlier_fn': interquantile_outlier,
        },
    ),
    (
        remove_outliers_with_groups,
        {
            'column': 'sold_price',
            'by': 'brand_title',
            'outlier_fn': interquantile_outlier,
        },
    ),
    (
        remove_outliers_with_groups,
        {
            'column': 'original_price',
            'by': 'brand_title',
            'outlier_fn': interquantile_outlier,
        },
    ),
    (group_mean, {'column': 'original_price', 'by':
['thirdlevel_title_']}),
    (group_mean, {'column': 'sold_price', 'by':
['thirdlevel_title_']}),
    (group_mean, {'column': 'sold_price', 'by':
['brand_title']}),
    (group_mean, {'column': 'original_price', 'by':
['brand_title']}),
    (group_mean, {'column': 'sold_price', 'by':
['brand_title', 'thirdlevel_title_']}),
    (group_mean, {'column': 'original_price', 'by':
['brand_title', 'thirdlevel_title_']}),
    (discount_rate, {'by': 'thirdlevel_title_'}),
    (discount_rate, {'by': 'brand_title'}),
    # (log_transform, {'column':
'sold_price_mean_by_thirdlevel_title_'}),
    # (log_transform, {'column': 'original_price'}),
```

```
    # (log_transform, {'column':
'original_price_mean_by_thirdlevel_title_'}),
    # (log_transform, {'column':
'original_price_mean_by_brand_title'}),
    # (log_transform, {'column':
'sold_price_mean_by_brand_title'}),
    # (log_transform, {'column':
'sold_price_mean_by_brand_title-thirdlevel_title_'}),
    # (log_transform, {'column':
'original_price_mean_by_brand_title-thirdlevel_title_'}),
    # (one_hot_encoding, {'column': 'thirdlevel_title_',
'prefix': 'category'}),
    (one_hot_encoding, {'column': 'brand_type', 'prefix':
'brand_type'}),
    (one_hot_encoding, {'column': 'condition', 'prefix':
'condition'}),
    (one_hot_encoding, {'column': 'shipment_term',
'prefix': 'shipment_term'}),
    (word_count, {'column': 'description'}),
    # (save_data, {}),
    X=[
        'product_id',
        'seller_id',
        'brand_id',
        'brand_title',
        'category_id',
        'product_status',
        'star',
        'description',
        'zerolevel_title',
        'firstlevel_title',
        'secondlevel_title',
        'price',
        'sold_price',
        'thirdlevel_title_',
        'brand_type',
        'condition',
        'shipment_term',
    ],
```

```
    y='sold_price',
    exclude_X=True,
    model_factory=Ridge
)

logger.info(pprint.pformat(model_summary.summary()))

if args.save:
    model_summary.save_model()
```

4. Metrics

```
import
datetime
  import math
  import pickle

  import numpy as np
  from sklearn.metrics import mean_squared_error, r2_score


  def root_mean_squared_error(y, y_pred):
     return math.sqrt(mean_squared_error(y, y_pred))


  class ModelSummary:
     def __init__(self, name, model, X_test, y_test):
         self.name = name
         self.model = model
         self.X_test = X_test
         self.y_test = y_test

     def _predict(self):
```

```
        return self.model.predict(self.X_test)


    def calc_metric(self, metric_func):
        y_pred = self._predict()
        return metric_func(self.y_test, y_pred)


    def rmsle(self):
        def _rmsle(target, prediction):
            return np.sqrt(np.square(np.log(prediction +
1) - np.log(target + 1)).mean())
        return self.calc_metric(_rmsle)


    def rmse(self):
        return self.calc_metric(root_mean_squared_error)


    def mse(self):
        return self.calc_metric(mean_squared_error)


    def r_squared(self):
        return self.calc_metric(r2_score)


    def adjusted_r_squared(self):
        dataset_len = self.y_test.shape[0]
        col_count = self.X_test.shape[1]
        r2 = self.r_squared()
        return 1 - (1 - r2) * (dataset_len - 1) /
(dataset_len - col_count - 1)


    def summary(self):
        return {
            'model': self.model,
            'mse': self.mse(),
```

```
                    'rmse': self.rmse(),
                    'rmsle': self.rmsle(),
                    'r-squared': self.r_squared(),
                    'adjusted-r-squared':
            self.adjusted_r_squared(),
                   }


       def save_model(self):
           with
        open(f'saved_models/{self.name}_{datetime.datetime.now()
        }.pickle', 'wb') as f:
                    pickle.dump(self.model, f)
```

5. Model Builder

```
import
logging
        from typing import List


        import pandas as pd
        from sklearn.model_selection import train_test_split


        from utils import timer




        logger = logging.getLogger(__name__)




        def create_features(
           df: pd.DataFrame,
           X: List[str],
           y: str,
           test_size: float,
           exclude_X: bool,
        ):
```

```
    df_train, df_test = train_test_split(df,
test_size=test_size)


    # train_categories =
df_train['thirdlevel_title_'].unique()
    # df_test =
df_test[df_test['thirdlevel_title_'].isin(train_categories)]


    if exclude_X:
        X_train = df_train.drop(X, axis=1)
        X_test = df_test.drop(X, axis=1)
    else:
        X_train = df_train[X]
        X_test = df_test[X]


    y_train = df_train[y]
    y_test = df_test[y]


    return X_train, y_train, X_test, y_test




def build_model(
    df: pd.DataFrame,
    model_factory,
    X: List[str],
    y: str,
    test_size: float = 0.2,
    model_kwargs: dict = None,
    exclude_X=False,
):
    with timer('Feature creation', logger):
        X_train, y_train, X_test, y_test = create_features(
            df, X, y, test_size, exclude_X
```

```
        )


    if model_kwargs is None:
        model_kwargs = {}


    model = model_factory(**model_kwargs)


    with timer('Model fitting', logger):
        model.fit(X_train, y_train)


    return model, X_test, y_test
```

6. Server

```
import csv


from flask import Flask, render_template, request


app = Flask(__name__)




@app.route('/', methods=['GET', 'POST'])
def index():
   categories = get_categories()
   brands = get_brands()


   if request.method == 'POST':
       form_data = request.form.to_dict()


       brand_type = [brand['brand_type'] for brand in brands if
brand['brand_title'] == form_data['brand']][0]
```

```
        ['original_price', 'original_price_mean', 'sold_price_mean',
'discount_rate', 'brand_type_ECONOMICAL',
         'brand_type_LUX', 'brand_type_POPULAR',
'brand_type_ULTRA_LUX', 'condition_GENTLY_WORN',
'condition_LIKE_NEW',
         'condition_NEW_WITH_TAGS', 'shipment_term_BUYER_PAYS',
'shipment_term_SELLER_PAYS']


        row = {
            'original_price': [form_data['original_price']],
            'original_price_mean': [form_data['original_price']]
        }




    return render_template('index.html', categories=categories,
brands=brands)




def get_categories():
    with open('data/categories.csv', encoding='utf-8') as f:
        reader = csv.DictReader(f)
        return list(reader)




def get_brands():
    with open('data/brands.csv', encoding='utf-8') as f:
        reader = csv.DictReader(f)
        return list(reader)




if __name__ == '__main__':
```

```
app.run(debug=True)
```

APPENDIX C: Model Results

#	model	r2	adjusted r2	mse	rmse	rmsle
1	ridge	0,648	0,648	1.508,693	38,842	0,495
2	lasso	0,634	0,634	1.485,364	38,540	0,491
3	linear	0,632	0,632	1.503,755	38,778	0,497
4	linear	0,618	0,618	703,949	26,532	0,458
5	ridge	0,618	0,618	691,463	26,296	0,458
6	lasso	0,612	0,612	706,801	26,586	0,459
7	linear	0,650	0,650	645,209	25,401	0,440
8	ridge	0,647	0,647	646,375	25,424	0,441
9	lasso	0,644	0,644	659,705	25,685	0,439
10	linear	0,512	0,512	890,608	29,843	0,583
11	ridge	0,505	0,505	1.049,059	32,389	0,522
12	ridge	0,638	0,638	1.372,390	37,458	0,495
13	linear	0,534	0,534	864,681	29,405	0,588
14	linear	0,649	0,649	644,486	25,387	0,441
15	lasso	0,650	0,650	643,603	25,369	0,437
16	**ridge**	**0,657**	**0,657**	**633,367**	**25,167**	**0,439**
17	ridge(λ = 0)	0,652	0,652	642,774	25,353	0,440
18	ridge(λ = 0.5)	0,650	0,650	654,865	25,590	0,439

Table 2.1

#	continuous variables	categorical variables	outliers deleted considering IQR	normalized colums	model	r2	adjusted r2	mse	rmse	rmsle
1	original_price, original_price_mean_by_thirdlevel_title_,sold_price_mean_by_thirdlevel_title_,original_price_mean_by_brand_title, discount_rate	brand_type_ECONOMICAL', 'brand_type_LUX', brand_type_POPULAR', 'brand_type_ULTRA_LUX', 'condition_GENTLY_WORN', condition_LIKE_NEW', 'condition_NEW_WITH_TAGS', shipment_term_BUYER_PAYS', 'shipment_term_SELLER_PAYS'	outliers of sold price and original price in each third level group	-	ridge	0,648	0,648	1.508,693	38,842	0,495
2	original_price, original_price_mean_by_thirdlevel_title_,sold_price_mean_by_thirdlevel_title_,original_price_mean_by_brand_title, discount_rate	brand_type_ECONOMICAL', 'brand_type_LUX', brand_type_POPULAR', 'brand_type_ULTRA_LUX', 'condition_GENTLY_WORN', condition_LIKE_NEW', 'condition_NEW_WITH_TAGS', shipment_term_BUYER_PAYS', 'shipment_term_SELLER_PAYS'	outliers of sold price and original price in each third level group	-	lasso	0,634	0,634	1.485,364	38,540	0,491
3	original_price, original_price_mean_by_thirdlevel_title_,sold_price_mean_by_thirdlevel_title_,original_price_mean_by_brand_title, discount_rate	brand_type_ECONOMICAL', 'brand_type_LUX', brand_type_POPULAR', 'brand_type_ULTRA_LUX', 'condition_GENTLY_WORN', condition_LIKE_NEW', 'condition_NEW_WITH_TAGS', shipment_term_BUYER_PAYS', 'shipment_term_SELLER_PAYS'	outliers of sold price and original price in each third level group	-	linear	0,632	0,632	1.503,755	38,778	0,497
4	original_price, original_price_mean_by_thirdlevel_title_,sold_price_mean_by_thirdlevel_title_,original_price_mean_by_brand_title, discount_rate	brand_type_ECONOMICAL', 'brand_type_LUX', brand_type_POPULAR', 'brand_type_ULTRA_LUX', 'condition_GENTLY_WORN', condition_LIKE_NEW', 'condition_NEW_WITH_TAGS', shipment_term_BUYER_PAYS', 'shipment_term_SELLER_PAYS'	outliers of sold price and original price in each third level and brand title group	-	linear	0,618	0,618	703,949	26,532	0,458
5	original_price, original_price_mean_by_thirdlevel_title_,sold_price_mean_by_thirdlevel_title_,original_price_mean_by_brand_title, discount_rate	brand_type_ECONOMICAL', 'brand_type_LUX', brand_type_POPULAR', 'brand_type_ULTRA_LUX', 'condition_GENTLY_WORN', condition_LIKE_NEW', 'condition_NEW_WITH_TAGS', shipment_term_BUYER_PAYS', 'shipment_term_SELLER_PAYS'	outliers of sold price and original price in each third level and brand title group	-	ridge	0,618	0,618	691,463	26,296	0,458
6	original_price, original_price_mean_by_thirdlevel_title_,sold_price_mean_by_thirdlevel_title_,original_price_mean_by_brand_title, discount_rate	brand_type_ECONOMICAL', 'brand_type_LUX', brand_type_POPULAR', 'brand_type_ULTRA_LUX', 'condition_GENTLY_WORN', condition_LIKE_NEW', 'condition_NEW_WITH_TAGS', shipment_term_BUYER_PAYS', 'shipment_term_SELLER_PAYS'	outliers of sold price and original price in each third level and brand title group	-	lasso	0,612	0,612	706,801	26,586	0,459
7	original_price, original_price_mean_by_thirdlevel_title_,sold_price_mean_by_thirdlevel_title_,original_price_mean_by_brand_title' , sold_price_mean_by_brand_thirdlevel_title_,original_price_mean_by_brand_thirdlevel_title	brand_type_ECONOMICAL', 'brand_type_LUX', brand_type_POPULAR', 'brand_type_ULTRA_LUX', 'condition_GENTLY_WORN', condition_LIKE_NEW', 'condition_NEW_WITH_TAGS', shipment_term_BUYER_PAYS', 'shipment_term_SELLER_PAYS'	outliers of sold price and original price in each third level and brand title group	-	linear	0,650	0,650	645,209	25,401	0,440

Table 2.2

#	continuous variables	categorical variables	outliers deleted considering IQR	normalized colums	model	r2	adjusted r2	mse	rmse	rmsle
8	original_price, original_price_mean_by_thirdlevel_title_,sold_price_mean_by_thirdlevel_title_,original_price_mean_by_brand_title, discount_rate	brand_type_ECONOMICAL', 'brand_type_LUX', brand_type_POPULAR', 'brand_type_ULTRA_LUX', 'condition_GENTLY_WORN', condition_LIKE_NEW', 'condition_NEW_WITH_TAGS', shipment_term_BUYER_PAYS', 'shipment_term_SELLER_PAYS'	outliers of sold price and original price in each third level and brand title group	-	ridge	0,647	0,647	646,375	25,424	0,441
9	original_price, original_price_mean_by_thirdlevel_title_,sold_price_mean_by_thirdlevel_title_,original_price_mean_by_brand_title, discount_rate	brand_type_ECONOMICAL', 'brand_type_LUX', brand_type_POPULAR', 'brand_type_ULTRA_LUX', 'condition_GENTLY_WORN', condition_LIKE_NEW', 'condition_NEW_WITH_TAGS', shipment_term_BUYER_PAYS', 'shipment_term_SELLER_PAYS'	outliers of sold price and original price in each third level and brand title group	-	lasso	0,644	0,644	659,705	25,685	0,439
10	original_price, original_price_mean_by_thirdlevel_title_,sold_price_mean_by_thirdlevel_title_,original_price_mean_by_brand_title, discount_rate	brand_type_ECONOMICAL', 'brand_type_LUX', brand_type_POPULAR', 'brand_type_ULTRA_LUX', 'condition_GENTLY_WORN', condition_LIKE_NEW', 'condition_NEW_WITH_TAGS', shipment_term_BUYER_PAYS', 'shipment_term_SELLER_PAYS'	outliers of sold price and original price in each third level and brand title group	box cox transformation on sold_price_mean_by_thirdlevel_title_, original_price, original_price_mean_by_thirdlevel_title_, original_price_mean_by_brand_title, sold_price_mean_by_brand_title	linear	0,512	0,512	890,608	29,843	0,583
11	original_price, original_price_mean_by_thirdlevel_title_,sold_price_mean_by_thirdlevel_title_,original_price_mean_by_brand_title, discount_rate_by_thirdlevel_title_,discount_rate_brand_title	brand_type_ECONOMICAL', 'brand_type_LUX', brand_type_POPULAR', 'brand_type_ULTRA_LUX', 'condition_GENTLY_WORN', condition_LIKE_NEW', 'condition_NEW_WITH_TAGS', shipment_term_BUYER_PAYS', 'shipment_term_SELLER_PAYS'	outliers of sold price in each third level and brand title group	-	ridge	0,505	0,505	1.049,059	32,389	0,522
12	original_price, original_price_mean_by_thirdlevel_title_,sold_price_mean_by_thirdlevel_title_,original_price_mean_by_brand_title, discount_rate_by_thirdlevel_title_,discount_rate_brand_title	brand_type_ECONOMICAL', 'brand_type_LUX', brand_type_POPULAR', 'brand_type_ULTRA_LUX', 'condition_GENTLY_WORN', condition_LIKE_NEW', 'condition_NEW_WITH_TAGS', shipment_term_BUYER_PAYS', 'shipment_term_SELLER_PAYS'	outliers of original price in each third level and brand title group	-	ridge	0,638	0,638	1.372,390	37,458	0,495
13	original_price, original_price_mean_by_thirdlevel_title_,sold_price_mean_by_thirdlevel_title_,original_price_mean_by_brand_title, discount_rate	brand_type_ECONOMICAL', 'brand_type_LUX', brand_type_POPULAR', 'brand_type_ULTRA_LUX', 'condition_GENTLY_WORN', condition_LIKE_NEW', 'condition_NEW_WITH_TAGS', shipment_term_BUYER_PAYS', 'shipment_term_SELLER_PAYS'	outliers of sold price and original price in each third level and brand title group	log transformation on sold_price_mean_by_thirdlevel_title_, original_price, original_price_mean_by_thirdlevel_title_, original_price_mean_by_brand_title, sold_price_mean_by_brand_title	linear	0,534	0,534	864,681	29,405	0,588

Table 2.3

#	continuous variables	categorical variables	outliers deleted considering IQR	normalized colums	model	r2	adjusted r2	mse	rmse	rmsle
14	original_price, original_price_mean_by_thirdlevel_title_,sold_price_mean_by_thirdlevel_title_,original_price_mean_by_brand_title' , sold_price_mean_by_brand_thirdlevel_title_,original_price_mean_by_brand_thirdlevel_title,discount_rate_by_thirdlevel_title_,discount_rate_brand_title	brand_type_ECONOMICAL', 'brand_type_LUX', brand_type_POPULAR', 'brand_type_ULTRA_LUX', 'condition_GENTLY_WORN', condition_LIKE_NEW', 'condition_NEW_WITH_TAGS', shipment_term_BUYER_PAYS', 'shipment_term_SELLER_PAYS'	outliers of sold price and original price in each third level and brand title group	-	linear	0,649	0,649	644,486	25,387	0,441
15	original_price, original_price_mean_by_thirdlevel_title_,sold_price_mean_by_thirdlevel_title_,original_price_mean_by_brand_title' , sold_price_mean_by_brand_thirdlevel_title_,original_price_mean_by_brand_thirdlevel_title,discount_rate_by_thirdlevel_title_,discount_rate_brand_title	brand_type_ECONOMICAL', 'brand_type_LUX', brand_type_POPULAR', 'brand_type_ULTRA_LUX', 'condition_GENTLY_WORN', condition_LIKE_NEW', 'condition_NEW_WITH_TAGS', shipment_term_BUYER_PAYS', 'shipment_term_SELLER_PAYS'	outliers of sold price and original price in each third level and brand title group	-	lasso	0,650	0,650	643,603	25,369	0,437
16	**original_price, original_price_mean_by_thirdlevel_title_,sold_price_mean_by_thirdlevel_title_,original_price_mean_by_brand_title' , sold_price_mean_by_brand_thirdlevel_title_,original_price_mean_by_brand_thirdlevel_title, discount_rate_by_thirdlevel_title_,discount_rate_brand_title**	**brand_type_ECONOMICAL', 'brand_type_LUX', brand_type_POPULAR', 'brand_type_ULTRA_LUX', 'condition_GENTLY_WORN', condition_LIKE_NEW', 'condition_NEW_WITH_TAGS', shipment_term_BUYER_PAYS', 'shipment_term_SELLER_PAYS'**	**outliers of sold price and original price in each third level and brand title group**	**-**	**ridge**	**0,657**	**0,657**	**633,367**	**25,167**	**0,439**
17	original_price, original_price_mean_by_thirdlevel_title_,sold_price_mean_by_thirdlevel_title_,original_price_mean_by_brand_title' , sold_price_mean_by_brand_thirdlevel_title_,original_price_mean_by_brand_thirdlevel_title,discount_rate_by_thirdlevel_title_,discount_rate_brand_title	brand_type_ECONOMICAL', 'brand_type_LUX', brand_type_POPULAR', 'brand_type_ULTRA_LUX', 'condition_GENTLY_WORN', condition_LIKE_NEW', 'condition_NEW_WITH_TAGS', shipment_term_BUYER_PAYS', 'shipment_term_SELLER_PAYS'	outliers of sold price and original price in each third level and brand title group	-	ridge($\lambda = 0$)	0,652	0,652	642,774	25,353	0,440
18	original_price, original_price_mean_by_thirdlevel_title_,sold_price_mean_by_thirdlevel_title_,original_price_mean_by_brand_title' , sold_price_mean_by_brand_thirdlevel_title_,original_price_mean_by_brand_thirdlevel_title,discount_rate_by_thirdlevel_title_,discount_rate_brand_title	brand_type_ECONOMICAL', 'brand_type_LUX', brand_type_POPULAR', 'brand_type_ULTRA_LUX', 'condition_GENTLY_WORN', condition_LIKE_NEW', 'condition_NEW_WITH_TAGS', shipment_term_BUYER_PAYS', 'shipment_term_SELLER_PAYS'	outliers of sold price and original price in each third level and brand title group	-	ridge($\lambda = 0.5$)	0,650	0,650	654,865	25,590	0,439

Table 2.4