|  |  |
| --- | --- |
|  |  |

Page 1

* Average Rating Column: Cleaning the average rating column by omitting the substring “out of 5 stars” to keep only the rating value, then converting the values to a numeric type and dropping the null rows in it as their number is relatively small.
* Manufacturer Column: Using LabelEncoder class to transform this text feature to numerical values.
* Seller Column: Using the built-in package Jason to work with this column data in order to calculate number of sellers for each product and its average price as the one product may be sold by one or more seller each with a different price. So, number of sellers and average price are two additional useful features that can be extracted from this column and can be added to the dataset to experiment with after filling the null values in number of sellers column and average price column each with its mean value.
* Number available in stock Column: Extracting the value of this column into 2 features; the first feature is the number available in stock and the second feature is the product status (new, collectible, used).

Then, filling the null values in the number available in stock column after converting it to numeric type with its mean value while filling the product status null rows by its mode value then transform its textual values into numerical ones using LabelEncoder.

Page 2

* Price column: Cleaning the price column value by omitting the ‘,’ and if it was written as a range like “12-20” we will take the average of them as its new value. As well as filling the null values by the corresponding value from the average price column that have been extracted from the seller column.
* Number of answered questions column: Converting the values to a numeric type and filling the null values with its median value.
* Number of reviews column: Converting the values to a numeric type and dropping the null rows in it as their number is relatively small.
* Product information column:

Extracting two useful information from this column; the first one is the best seller rank feature and the second one is the category rank (the rank of this product among its category). Then filling the null values in each column by its mean value.

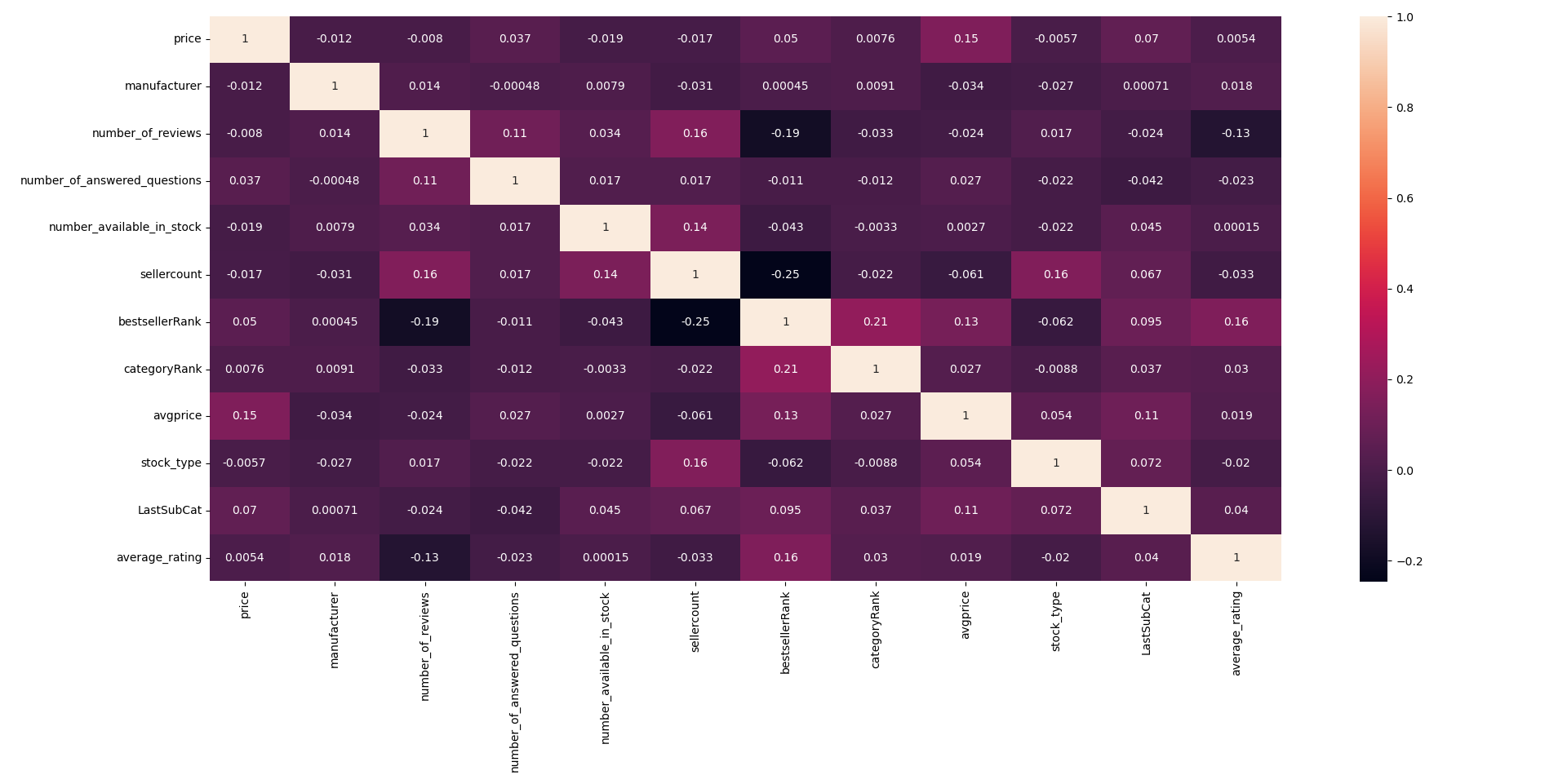
* Amazon category and sub category: Extracting the last sub category from this column and taking it in consideration to experiment with then transform its value into numerical type using LabelEncoder.

Page 3

* Unique id and product name columns are dropped (discarded).
* Using Standardization to re-scale the selected features value so that it has distribution with 0 mean value and variance equals to 1.
* Using correlation coefficient scores as an example of Filter methods. For feature selection, we are often interested in a positive score; with the larger the positive value, the larger the relationship, and, more likely, the feature should be selected for modeling.

Page 4

* After calculating correlation and plotting the heatmap to see the most correlated features, we decided to take the Top 1% Correlation training features with the average rating as the highest correlation value with the average rating column is 0.16 (Best seller rank column) as it is shown in the screenshot below:

****

* According to the heatmap and the threshold value (0.01) for selecting variables we can see that the variables which have higher correlation coefficient values with the average rating variable are best seller rank, Last sub category, category rank, average price, manufacturer. So, we are going to use these 4 variables for our prediction model.

Sizes of training, testing sets:

Page 5

Training set=70% of the data.

Testing set=30% of the data.

* Multiple Linear regression:
* Training Mean Square Error: 0.13263238656329054
* Testing Mean Square Error: 0.13609375041968863

In case we used cross validation with k=5 (5-fold cross validation):

* cross validation score: 0.1329798806560429
* Value of Intercept with Y-axis: 4.7105576493835395
* Coefficient of X:

manufacturer (X1): 0.00761654

avgprice (X2): -0.00087887

bestsellerRank(X3): 0.05922792

categoryRank(X4): -0.00168379

LastSubCat(X5): 0.01001413

* Polynomial regression of degree 3:
* Training Mean Square Error: 0.12981585144784982
* Testing Mean Square Error 0.1359314744429799
* Value of Intercept with Y-axis: 4.733497511752612
* Coefficient of X: Too many Coefficients (56) so they are already printed in the code.

Page 6

Why did we choose degree 3 in polynomial regression?

Training error will probably be decreased with the increase of model complexity until overfitting where the error reaches almost zero while testing error decreases until degree 3 and as we see when order complexity gets too much the testing error may come up again so we choose the polynomial of degree 3 because it has the minimum testing error among the other degrees as in degree 4 the testing error has increased while the training error had decreased which means that overfitting has happened (High Variance).

As we can see the testing mean square error in the polynomial model is less than the multiple linear models so it’s better to fit the data and to predict the average rating feature.

Because we can’t try all fitting of all order polynomials, regularization is used to penalize the system from under /overfitting and to control model complexity as well as solving the bias-variance tradeoff. In the two models we have used ridge regression with built-in cross-validation of the alpha parameter.

Multiple linear model:

Rigde Regularization test Mean Square Error: 0.13609374936682447

Polynomial linear model:

Rigde Regularization test Mean Square Error: 0.13588034295891707