**REPORT CA1**

Sidhant Naveria, Abhineet Misra

Institute of Systems Science, National University of Singapore, Singapore 119615

**Abstract**

when ever a company launch new product, it becomes very difficult for users to predict the price of the product. In current scenario, we will be using shoes price dataset and we will make a prediction of the prices using 3 different models and 1 ensemble. We will compare the performance of these models and will try to increase the accuracy.

**Introduction**

Every time a company introduce new products, it becomes very difficult for consumers to analyse the price quoted by the company. We are trying to build a model which will help the consumer to predict the price of the product. This is a regression base problem and we will be using Linear regression, Decision tree, Neural network to tackle this problem. We will build 3 models using above algorithm to get the best result. We will be using RMSE, MSE to compare performance of each model. In the end, we will apply ensemble technique to check, if we can get better results by combining all the 3 models.

**Tools/Techniques**

**Data pre-processing**

First problem, we are facing is with the dataset. This dataset has 48 columns and 33800 rows. Most of the columns have null data and we will drop the columns if null data is more than 70 %. To pre-process null data columns, we will be using different techniques. We will check other columns if we could find related data, then we will replace that null value with that related data. For eg we have 2 columns 'manufacturer' and ‘brand’, we will fill the null value of ‘brand’ using 'manufacturer' column and in the end, we will drop the 'manufacturer' column as the values in both the columns are same. Other techniques, which we will deploy to handle the null values is by replacing the null values with the mode of the column (as the columns is categorical, we will use mode, in case of continuous columns we will use mean of the column).

We have columns 'prices.amountMax','prices.amountMin' and we will be combining these 2 columns by creating a new column and populating each row with the mean of these 2 columns.

For categorical columns we used binary encoder, which have more than 100 categories. One hot encoding was introducing more columns than binary encoder, to avoid curse of dimensionality we have used binary encoder. For some columns which have very less categories, we used category coding.

**Performance of the model**

**Linear regression model**

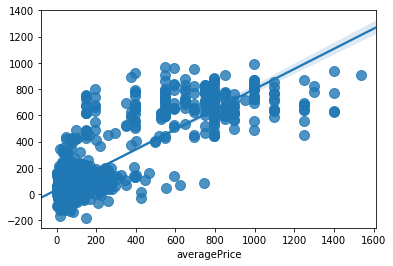
accuracy= 0.7244591319064707

RMSE for test data= 112.52572022109638

RMSE for training data= 128.73709949391062

MSE for test data= 69.3346709710667

MSE for training data= 72.47758021967199

****

MSE for test data and MSE for training data are very close to each other. Which shows there is no overfitting in the model

**Regression tree (Decision tree)**

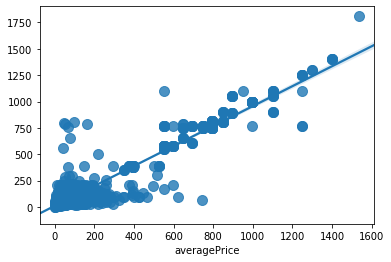
accuracy= 0.9026116174484783

RMSE for test data= 66.8978817751109

RMSE for training data= 71.29065590199201

MSE for test data= 34.41356846208595

MSE for training data= 31.74894158061407

****

Decision tree is giving better accuracy for the model and even MSE score has gone down. Max\_depth hyper parameter has helped to increase the accuracy and decrease the MSE. Till max depth = 9 accuracy increased, after that accuracy still increases, but model started to get overfitted.

**Neural Network**

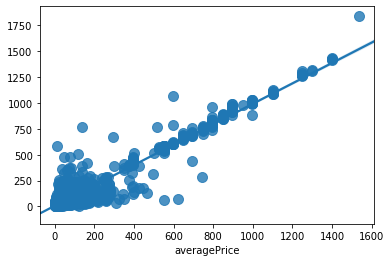
accuracy= 0.9258815008615487

RMSE for test data= 58.360915824423415

RMSE for training data= 48.5800571622207

MSE for test data= 33.46490335060334

MSE for training data= 22.35066047758133



Accuracy improved with MLPregressor. We have used learning rate as adaptive which keeps the learning rate constant to ‘learning\_rate\_init’ as long as training loss keeps decreasing. Each time two consecutive epochs fail to decrease training loss by at least tol, or fail to increase validation score by at least tol if ‘early\_stopping’ is on, the current learning rate is divided by 5. Using adaptive learning improved the accuracy and reduced the RMSE as well.

**Gradient Boosting**

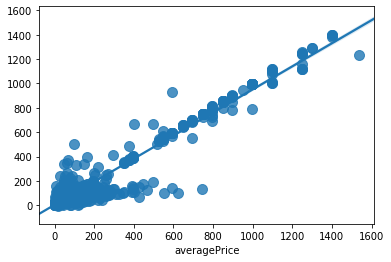
accuracy= 0.9457812209182959

RMSE for test data= 49.91529671200755

RMSE for training data= 62.99548901689683

MSE for test data= 26.068148680133707

MSE for training data= 17.306787136676473



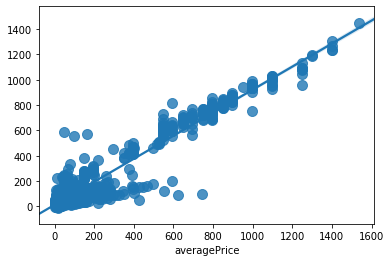
Accuracy has improved with gradient boosting. After max depth of 8 model starts to overfit

**Ensemble model**

accuracy= 0.9323939987791123

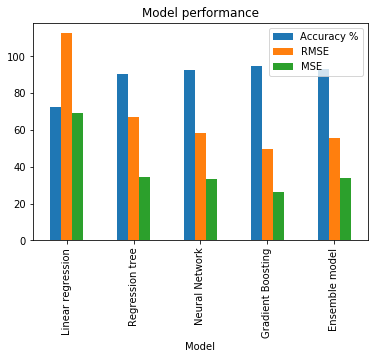
RMSE for test data= 55.73800428543187

MSE for test data= 34.052186402989825

****

**Model summary**

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Accuracy %** | **RMSE** | **MSE** |
| **Linear regression** | 72.445 | 112.52572022109638 | 69.3346709710667 |
| **Regression tree** | 90.261 | 66.8978817751109 | 34.41356846208595 |
| **Neural Network** | 92.588 | 58.360915824423415 | 33.46490335060334 |
| **Gradient Boosting** | 94.578 | 49.91529671200755 | 26.068148680133707 |
| **Ensemble model** | 93.239 | 55.73800428543187 | 34.052186402989825 |



**Conclusion:**

Gradient Boosting give best accuracy with lowest RMSE and MSE for the given dataset.