Report

BigMart Products Outlet Sales Analysis and Prediction

Data loading

In that part we load train and test data and we checked types of data and null values and some statistic information like mean, min and max ….

Exploratory Data Analysis

The most important in that section we visualize our data. Every feature we have in our data. In the main goal to check if we have something wrong like the same attribute but written with different forms as we discover with attribute **low fat** and **LF** in the **Fat content variable.**

Secondly, we checked the problem of balanced data and also, we check the distribution of our data, especially the variable that are of types float (non-categorical).

We detect outliers in target variable and also in other variables (type float), but in this model we didn't remove any outlier in target variable in contract we remove them in other variables (type float).

We replace all missing values in categorical variable we most often appear in categorical variable in the mean time we replace other nan variables with **median.**

Finally, we encode the categorical variable with **ONE HOST ENCODER.**

Modeling

We used a variance of algorithm for this regression problem such as **linear regression model, random forest regression and XGBoost ..**

For linear regression we can see the result as bellow

```
Training---------------
---------------Model Performance--------------
Score  0.5610041329122064
MAE  837.8939940757457
MSE 1274468.3107270973
RMSE  1128.9235185463617


Test---------------------
---------------Model Performance--------------
Score  0.5661816189918984
MAE  832.0984973112957
MSE 1271909.2061534196
RMSE  1127.789522097727
```

The result of train data and test data is stable as we can see from the R2 score

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y}_i)^2}$$

The best result was given by the Random Forest Regressor and XGBoot without any transformation on our data.

```
Training---------------
---------------Model Performance-------------
Score  0.9358749020972884
MAE  299.01155003352335
MSE 186164.40683468786
RMSE  431.4677355662737


Test---------------------
---------------Model Performance-------------
Score  0.5776958391902476
MAE  776.0783704208056
MSE 1238150.7410604325
RMSE  1112.7222209789973
```

Figure 1: result of Random forest

```
Training---------------
---------------Model Performance-------------
Score  0.6907656738600481
MAE  669.7572271439657
MSE 897751.8441548335
RMSE  947.4976750128907


Test---------------------
---------------Model Performance-------------
Score  0.5890547575313851
MAE  765.9433900541578
MSE 1204847.6044426048
RMSE  1097.6555035358792
```

Figure 2 : result of XGboost

The difference between to result that **the random forest** fit data well but in test part fails in the meantime the **xgboost** doesn't fit well but give best result in testing part which is logic because we don't have a huge difference between score of training and score of the test.

After that we apply power transform on target variable and other variable of type float.

$$y_i^{(\lambda)} = \begin{cases} ((y_i+1)^\lambda - 1)/\lambda & \text{if } \lambda \neq 0, y \geq 0 \\ \log(y_i+1) & \text{if } \lambda = 0, y \geq 0 \\ -((-y_i+1)^{(2-\lambda)} - 1)/(2-\lambda) & \text{if } \lambda \neq 2, y < 0 \\ -\log(-y_i+1) & \text{if } \lambda = 2, y < 0 \end{cases}$$

Figure 3: Yeo–Johnson transformation

You could see the graph of transformation in the jupyter notebook in the section of **transformation.**

The result with random forest with default parameter as bellow

```
Training---------------
---------------Model Performance--------------
Score  0.9522532108683132
MAE  0.1683669235676465
MSE 0.04774678913168681
RMSE  0.21851038678215462


Test--------------------
---------------Model Performance--------------
Score  0.674904137327655
MAE  0.4424988240852887
MSE 0.3250958626723451
RMSE  0.570171783476125
```

Figure 4:The result with random forest with default parameters

We fit also the XGBoost with the train data that we got from power transform, which gave us the result bellow.

```
Training---------------
---------------Model Performance--------------
Score  0.7301093784879331
MAE  0.40080336674624034
MSE 0.2698906215120669
RMSE  0.5195099821101293


Test--------------------
---------------Model Performance--------------
Score  0.6955005122119338
MAE  0.4297577935503689
MSE 0.3044994877880663
RMSE  0.5518147223371865
```

Figure 5: result of xgboost with power transform data

Feature engineering

As the final step we choose the XGBoost algorithm to continue with for feature engineering. For that we started with PCA for now and in the next version of my project I will add more transformation for feature engineering.

```
Training---------------
---------------Model Performance--------------
Score  0.7495994353126728
MAE  0.3859254771515629
MSE 0.2504005646873272
RMSE  0.5004004043636727


Test--------------------
---------------Model Performance--------------
Score  0.6730995475547408
MAE  0.4479646988035649
MSE 0.3269004524452593
RMSE  0.5717520900226419
```

Figure 6 : the result of xgboost after applying PCA

Conclusion

We don’t look for fine tuning the parameter right for the XGBoost, but we will do it for the next version because takes too much time.

For now we are going to test our unlabeled data with XGBoost model without PCA.