# IE4211 Modelling & Analytics
# Academic Year 2023/2024 Semester 1
# Project Report
# Group 9

| Name | Matriculation Number |
|---|---|
| Chan Keng Jit | A0217465B |
| Toh Yi Zhi | A0222554L |
| Vikas Harlani | A0214360U |
| Yap Zi Chen | A0244939N |
| Zhang Xiao | A0239603E |

# Content Page

# 1. Introduction

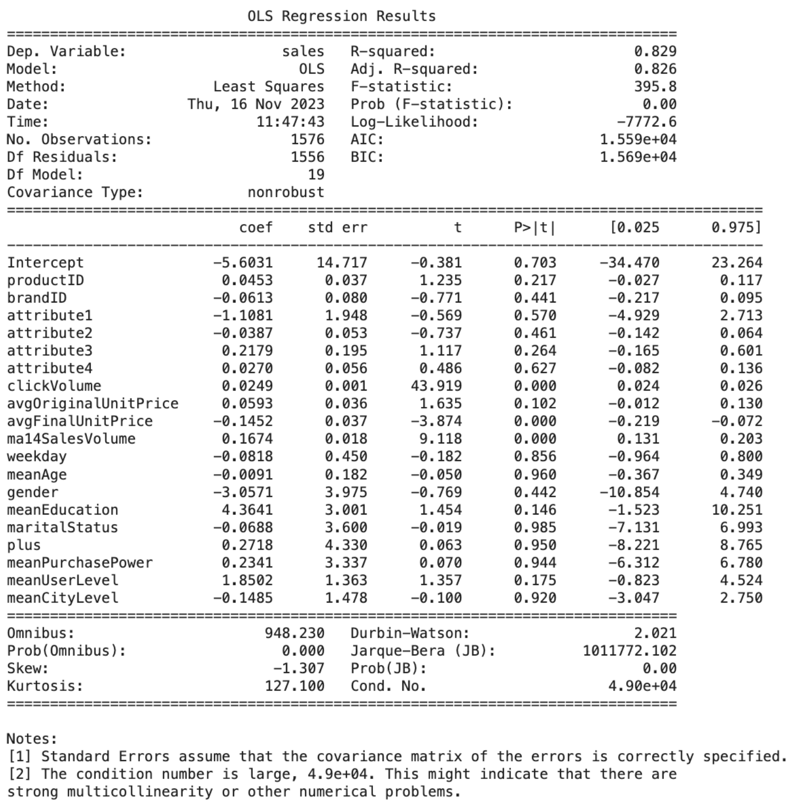
This project involves utilising sales data from an e-commerce company, with an objective of obtaining an accurate prediction using the training data and the test data available. The data available includes various attributes related to the products, the customers as well as the days. However, the sales column has been hidden in the test data set for prediction purposes. Following this, an inventory decision would have to be made and the objective is to maximise the profits induced from our inventory decision model.
The report would first delve into the data exploration and pre-processing of the provided datasets, as well as using various regression models and methodologies to derive the best prediction model, through the tuning of the various hyperparameters as well as the evaluation of the different regression models utilised. The derivation of the inventory decision model would also be explained in the report.

# 2. Data Exploration

This section aims to discuss the various findings through the process of data exploration of the training datasets.

## 2.1 Ordinary Least Squares (OLS) Regression

We did an ordinary least squares regression model initially on the raw train dataset in order to explore the dataset, by using statsmodel.formula.api.ols. Performing linear regression on the training dataset using all the predictors yields a model summarised as shown in Figure 1.
The R^2 and adjusted R^2 value is relatively high at 0.829 and 0.826 respectively. However, it could be due to the fact that the derivations for R^2 and adjusted R^2 are inherently biased estimators, WIth a highly complex dataset with a multitude of predictor variables, the statistical model would begin to describe the random errors in the datasets instead of the assessing the relationships between the variables, which would naturally achieve higher R-squared values. Furthermore, due to the datasets not being scaled or processed yet, the R^2 and adjusted R^2 values could be affected by a confluence of factors like multicollinearity amongst the explanatory variables as well as spurious regressions, which resulted in artificially high values.

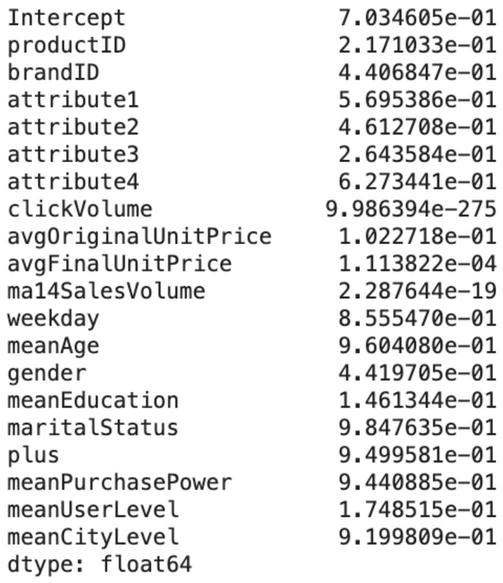


```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  sales   R-squared:                       0.829
Model:                            OLS   Adj. R-squared:                  0.826
Method:                 Least Squares   F-statistic:                     395.8
Date:                Thu, 16 Nov 2023   Prob (F-statistic):               0.00
Time:                        11:47:43   Log-Likelihood:                -7772.6
No. Observations:                1576   AIC:                         1.559e+04
Df Residuals:                    1556   BIC:                         1.569e+04
Df Model:                          19
Covariance Type:            nonrobust
=========================================================================================
                            coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------
Intercept                -5.6031     14.717     -0.381      0.703     -34.470      23.264
productID                 0.0453      0.037      1.235      0.217      -0.027       0.117
brandID                  -0.0613      0.080     -0.771      0.441      -0.217       0.095
attribute1               -1.1081      1.948     -0.569      0.570      -4.929       2.713
attribute2               -0.0387      0.053     -0.737      0.461      -0.142       0.064
attribute3                0.2179      0.195      1.117      0.264      -0.165       0.601
attribute4                0.0270      0.056      0.486      0.627      -0.082       0.136
clickVolume               0.0249      0.001     43.919      0.000       0.024       0.026
avgOriginalUnitPrice      0.0593      0.036      1.635      0.102      -0.012       0.130
avgFinalUnitPrice        -0.1452      0.037     -3.874      0.000      -0.219      -0.072
ma14SalesVolume           0.1674      0.018      9.118      0.000       0.131       0.203
weekday                  -0.0818      0.450     -0.182      0.856      -0.964       0.800
meanAge                  -0.0091      0.182     -0.050      0.960      -0.367       0.349
gender                   -3.0571      3.975     -0.769      0.442     -10.854       4.740
meanEducation             4.3641      3.001      1.454      0.146      -1.523      10.251
maritalStatus            -0.0688      3.600     -0.019      0.985      -7.131       6.993
plus                      0.2718      4.330      0.063      0.950      -8.221       8.765
meanPurchasePower         0.2341      3.337      0.070      0.944      -6.312       6.780
meanUserLevel             1.8502      1.363      1.357      0.175      -0.823       4.524
meanCityLevel            -0.1485      1.478     -0.100      0.920      -3.047       2.750
==============================================================================
Omnibus:                      948.230   Durbin-Watson:                   2.021
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          1011772.102
Skew:                          -1.307   Prob(JB):                         0.00
Kurtosis:                     127.100   Cond. No.                     4.90e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.9e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

*Fig 1. Simple Linear Regression with all the predictors*

The P-values of all the predictors are obtained as follows, with clickVolume observed to have an extremely low P-value as compared to the other predictor variables.

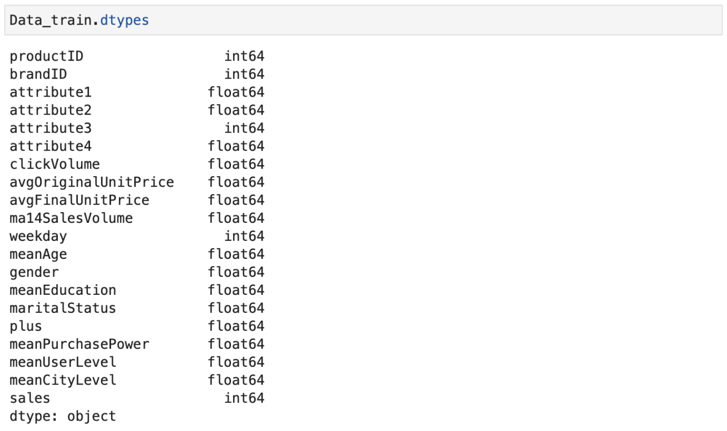


```
Intercept                7.034605e-01
productID                2.171033e-01
brandID                  4.406847e-01
attribute1               5.695386e-01
attribute2               4.612708e-01
attribute3               2.643584e-01
attribute4               6.273441e-01
clickVolume             9.986394e-275
avgOriginalUnitPrice     1.022718e-01
avgFinalUnitPrice        1.113822e-04
ma14SalesVolume          2.287644e-19
weekday                  8.555470e-01
meanAge                  9.604080e-01
gender                   4.419705e-01
meanEducation            1.461344e-01
maritalStatus            9.847635e-01
plus                     9.499581e-01
meanPurchasePower        9.440885e-01
meanUserLevel            1.748515e-01
meanCityLevel            9.199809e-01
dtype: float64
```

*Fig 2. P-values of all the predictors*

## 2.2 Data Types

The attributes observed are either of integer types or float types when we checked on the attributes available in the train dataset. Furthermore, we realised that ‘productID’, ‘brandID’, as well as ‘weekday’ are considered as numerical data as shown, which required us to convert them to categorical data types for proper processing.

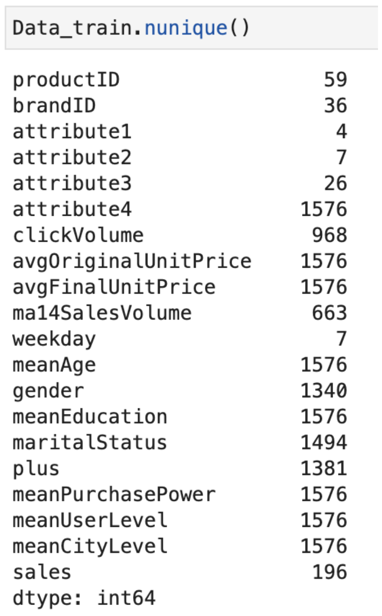


```
Data_train.dtypes
```

```
productID                int64
brandID                  int64
attribute1             float64
attribute2             float64
attribute3               int64
attribute4             float64
clickVolume            float64
avgOriginalUnitPrice   float64
avgFinalUnitPrice      float64
ma14SalesVolume        float64
weekday                  int64
meanAge                float64
gender                 float64
meanEducation          float64
maritalStatus          float64
plus                   float64
meanPurchasePower      float64
meanUserLevel          float64
meanCityLevel          float64
sales                    int64
dtype: object
```

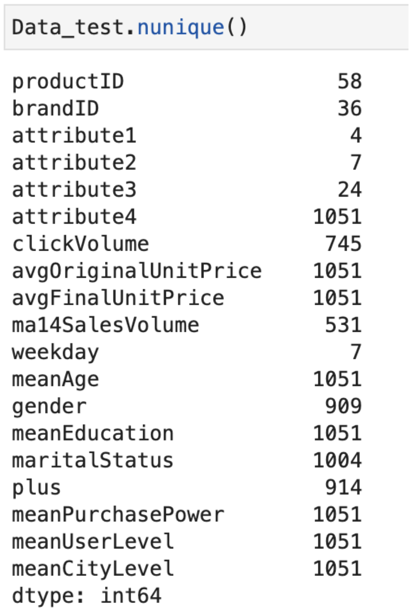
*Fig 3. Data Types in the train dataset*

## 2.3 Unique Values



```
Data_train.nunique()
```

```
productID                 59
brandID                   36
attribute1                 4
attribute2                 7
attribute3                26
attribute4              1576
clickVolume              968
avgOriginalUnitPrice    1576
avgFinalUnitPrice       1576
ma14SalesVolume          663
weekday                    7
meanAge                 1576
gender                  1340
meanEducation           1576
maritalStatus           1494
plus                    1381
meanPurchasePower       1576
meanUserLevel           1576
meanCityLevel           1576
sales                    196
dtype: int64
```



```
Data_test.nunique()
```

```
productID                 58
brandID                   36
attribute1                 4
attribute2                 7
attribute3                24
attribute4              1051
clickVolume              745
avgOriginalUnitPrice    1051
avgFinalUnitPrice       1051
ma14SalesVolume          531
weekday                    7
meanAge                 1051
gender                   909
meanEducation           1051
maritalStatus           1004
plus                     914
meanPurchasePower       1051
meanUserLevel           1051
meanCityLevel           1051
dtype: int64
```

*Fig 4. Unique values in the train dataset (left) and test dataset (right)*

We also observed that there were 2 extra unique values in ‘attribute3’ for the train data which had 26 unique values, as compared to the test data which had 24 unique values. Similarly, ‘productID’ had 59 unique values in the train data, while it had 58 unique values in the test data. Therefore, in order to utilize one-hot encoding, we had to append and combine both the train and test data, convert the necessary attributes such as ‘weekday’, ‘productID’, ‘brandID’, ‘attribute1’, ‘attribute2’, and ‘attribute3’ to categorical data, before splitting them up for further analysis.

## 2.4 Correlation Matrix

The correlation heatmap matrix shows that ‘avgFinalUnitPrice’ has a strong correlation with ‘avgOriginalUnitPrice’ at 0.93. Furthermore, there is a high correlation between ‘clickVolume’ and ‘sales’ at 0.90, as well as ‘attribute1’ and ‘attribute2’, at 0.55.

For such attributes, they would be retained in order for us to get a general comprehensive overview of the model performance with all the features included.

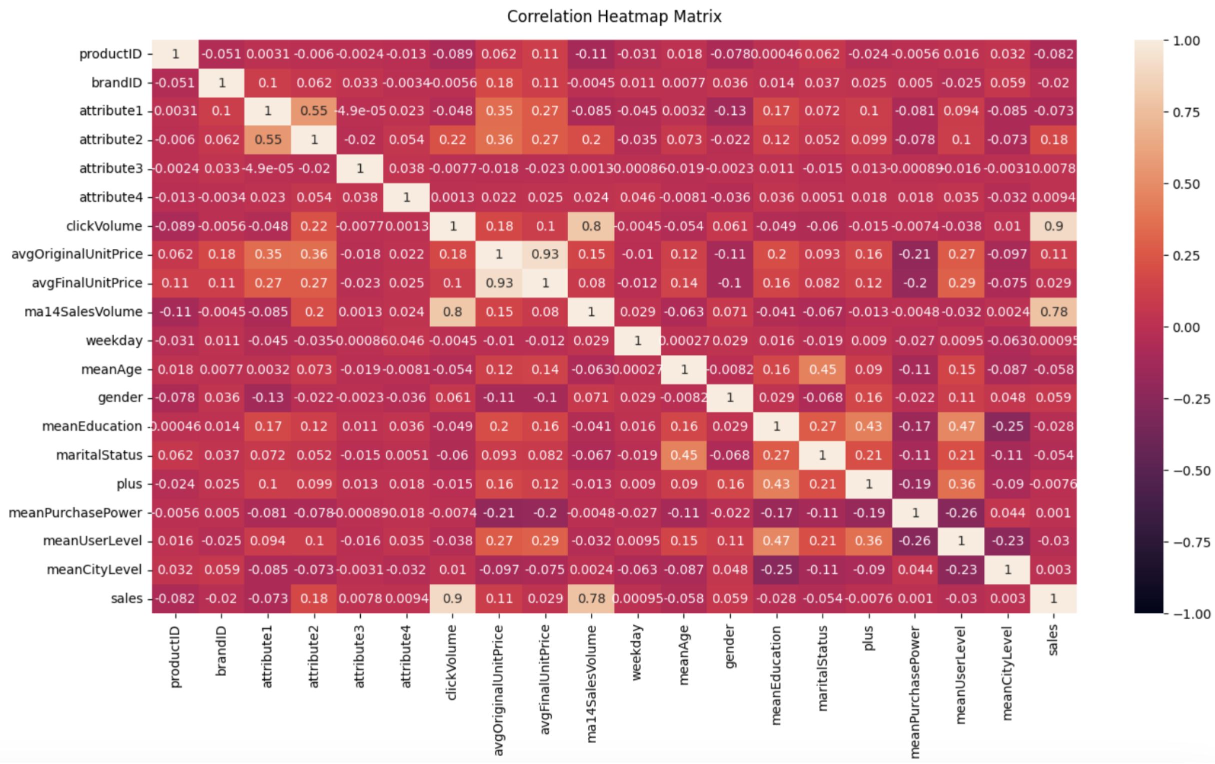


*Fig 5. Correlation Heatmap Matrix*

## 2.5 Scatter Plot

A scatter plot was generated to display the relationship between any two variables by plotting each observation as a point on a graph with one variable on the x-axis and the other on the y-axis as shown in Figure 4. In general, it helped to identify the non-linear patterns, trends and outliers in the train dataset. A more detailed view could be observed in the jupyter notebook file.

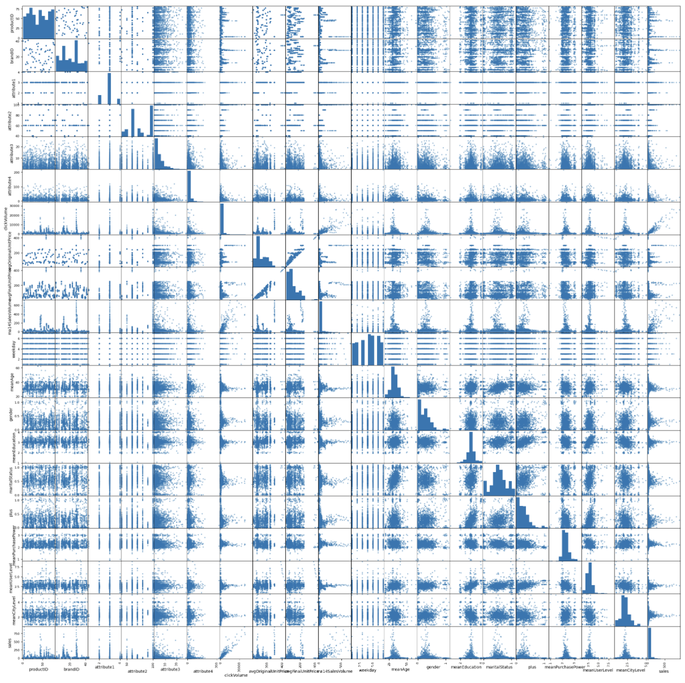


*Fig 6. Scatterplot Matrix*

# 3. Data Pre-Processing

In this section, the data pre-processing steps that were taken would be explained, according to different data methodology and regression models that would be running them.
The original datasets provided were ‘Data-train.csv’ as well as ‘Data-test.csv’.

## 3.1 Standardised Dataset

In the Stan processed dataset, we standardise the features so that the standard deviation of each feature is now one. However, we do not perform encoding for the categorical variables, so as to not overcomplicate the dataset at this point. The

StandardScaler library was used and was fitted onto only X_train, but used to transform both X_train and X_test, which would be explained afterwards.

## 3.2 One-Hot Encoding Dataset

For the data pre-processing, in the One-Hot Encoding Dataset, we performed One-Hot Encoding for the categorical features and then performed standardisation.

The (assumed) categorical predictors that we performed One-Hot Encoding are as follows:

1. weekday
2. product_ID
3. brand_ID
4. attribute1
5. attribute2
6. attribute3

Weekday had only 7 unique values and specified the days of the week, hence we concluded that it was categorical. Likewise, product_ID and brand_ID were categorical features as well. As for the attributes, attribute1 had only 4 unique values in both the train and test dataset, attribute2 had only 7 unique values in the train and test dataset, while attribute3 had 26 unique values in the train dataset and 24 unique values in the test dataset. Hence, they were likely to all be categorical.

The train dataset and test dataset was first combined, as the One-Hot Encoding could be performed together. Then, the categorical predictors were converted to String to be utilised as One-Hot Encoding (OHE). Subsequently, they were split back into train and test datasets.

Next, we performed standardisation on the train and test datasets that were split back as aforementioned. With StandardScaler(), we applied fit_transform() on the train dataset so that we were able to learn the scaling parameters on the train dataset and at the same time, we scaled the train data accordingly. On the other hand, we only used transform() on the test dataset because we used the scaling parameters learned on the train dataset to scale the test dataset.

In real world scenarios, only the train datasets are usually available, and a model is developed according to that to predict unseen instances of similar datasets. If the entire dataset was transformed with fit_transform(), the transformation would be done according to the unseen instances of the datasets, which results in an over-optimistic model which would be prepared by the metrics of the unseen samples as well and is thus not desirable as there is a greater inherent bias.

Lastly, the comma-separated values (CSV) files of the transformed datasets which were cleaned were saved, which could be used for running through the various regression models.

## 3.3 Log Sales

An alternative approach to using Sales as the response variable was explored by using log(Sales).
For the sales column, it was extracted from the original dataset and transformed into a log(sales) column. Since the sales data was rather skewed with an extreme range of sales values, through log transformation, it allowed the sample points to be closer together to approximately conform to near normality.

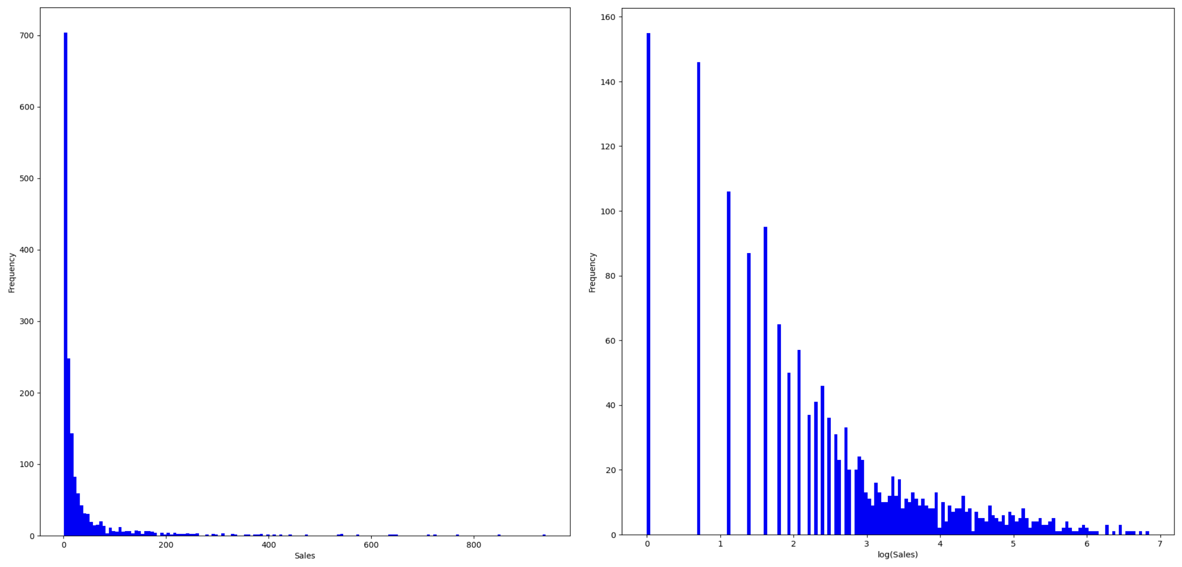


*Fig 7. Histogram of sales (left) and Histogram of log(sales) (right)*

As observed, the histogram of the original sales data is heavily right skewed, with a huge range of values encompassed, and thus may cause some models to underperform. Therefore, all the models would be compared to test the most efficient regression model.

# 4. Data Methodology and Regression Models

This section aims to delve into the various data methodologies and regression models that were utilised and tested for us to arrive at the most optimal Mean Squared Error (MSE). Due to the high overhead time costs required for running GridSearchCV to find the most optimal hyperparameters, all the codes involving the processes of GridSearchCV had been commented out, but were fully utilised while running the respective regression models.

## 4.1 Linear Regression

LASSO (Least Absolute Shrinkage and Selection Operator), and Ridge Regression were considered. Lasso regression is better when dealing with datasets containing numerous predictors. Ridge regression mitigates multicollinearity issues and stabilises coefficient estimates when predictors are highly correlated. Both LASSO and Ridge Regression will be considered below, by finding the best alpha which resulted in the smallest Mean Squared Error (MSE) using 10-Fold Cross Validation.

### 4.1.1 Least Absolute Shrinkage and Selection Operator (LASSO) Regression

| | |
|---|---|
| **Parameter Values** | Alpha = 5 |
| **Corresponding MSE** | 1412.90 |

*Table 1. Best parameters and corresponding MSE*

### 4.1.2 Ridge Regression

| | |
|---|---|
| **Parameter Values** | Alpha = 100 |
| **Corresponding MSE** | 1366.20 |

*Table 2. Best parameters and corresponding MSE*

## 4.2 Random Forest

Random Forest is a general approach that typically works well in both classification and regression tasks. It provides an improvement over decision trees and bagging by creating many trees, but also limiting the number of features available at each split. For our project, we explored using Random Forest to predict the sales. Specifically we used SKLearn's RandomForestRegressor library. Since the performance of Random Forest is highly dependent on the hyperparameters selected, a grid search of these parameters (with 10-fold cross validation) was employed to determine the best results. We were able to achieve a mean squared error of 978 with our best hyperparameters. The table below shows a summary of the results.

| | |
|---|---|
| **Parameter Values** | *n_estimators=1500,<br>ccp_alpha=0.03,<br>max_depth=8,<br>max_features=51 (total number of features divided by 3),<br>min_samples_leaf=1,<br>min_samples_split=2* |
| **Corresponding MSE** | 978.68 |

*Table 3. Best parameters and corresponding MSE for Random Forest*

## 4.3 Gradient Boosting

Gradient Boosting Regression is a robust and versatile machine learning technique known for excelling in capturing complex, non-linear relationships within data. Its ensemble learning approach combines weak learners, typically decision trees, to create a strong predictive model, making it less sensitive to outliers and providing high predictive performances. In the following sections, three different types of Gradient Boosting Regressors will be explored. Grid search with a 10-Fold Cross Validation was used across all three models, and the best results are shown below.

### 4.3.1 SKLearn Gradient Boosting

| | |
|---|---|
| **Parameter Values** | learning_rate=0.2,<br>n_estimators=100,<br>max_depth=3,<br>min_samples_leaf=2,<br>min_samples_split=10 |

| | |
|---|---|
| **Corresponding MSE** | 862.54 |

Table 4. Best parameters and corresponding MSE

### 4.3.2 CatBoost

CatBoost is a variation of the traditional gradient boosting which offers several advantages over the traditional gradient boosting methods. One notable benefit is its built-in support for handling categorical features, eliminating the need for manual preprocessing such as one-hot encoding. This feature simplifies the model development process and often leads to better performance, especially in scenarios where categorical variables play a significant role. CatBoost also uses an efficient implementation of ordered boosting, which improves training speed and convergence.

| | |
|---|---|
| **Parameter Values** | *min_data_in_leaf = 2, depth = 3, l2_leaf_reg = 3, iterations = 200, learning_rate = 0.2* |
| **Corresponding MSE** | 798.05 |

Table 5. Best parameters and corresponding MSE

### 4.3.3 eXtreme Gradient Boosting (XGBoost)

One key strength of XGBoost over traditional gradient boosting is its regularisation techniques, such as L1 and L2 regularisation, which help prevent overfitting and improve model generalisation. XGBoost also employs a parallelized implementation that enhances training speed, making it suitable for large datasets. It is also equipped with a built-in capability to handle missing data, reducing the need for extensive data preprocessing.

| | |
|---|---|
| **Parameter Values** | *colsample_bytree=0.4, gamma=0.2, learning_rate=0.15, max_depth=4, min_child_weight=1* |
| **Corresponding MSE** | 904.23 |

Table 6. Best parameters and corresponding MSE

## 4.4 Support Vector Machine Regression (SVM)

### 4.4.1 SVM - Parameter Grid Setup

As seen in the Data Exploration section (Section 2), the dataset is in a high-dimensional space. Hence, we decided to use the Support Vector Machine (SVM) algorithm which is effective with a large number of features (dimensions). We considered particular 4 parameters: kernel type (kernel), kernel coefficient (gamma; ignored by linear kernel), regularization parameter C, as well as degree of polynomial kernel function (irrelevant to other kernel types).

| **Parameters** | **Values of the Parameters** |
|---|---|
| SVM__Kernel | *‘poly’, ‘linear’, ‘rbf’* |
| SVM__Gamma | *‘scale’, ‘auto’* |
| SVM__C | *0.1, 1, 10, 100, 1000* |
| SVM__degree | *2, 3, 4* |

*Table 7. SVR Parameters Grid*

We chose the most common parameter values that appeared to be reasonable to the dataset. For example, assuming the best kernel is ‘poly’, we only choose SVM_degree = 2, 3, 4 because in real-life datasets, it's often unlikely to observe relationships that require a large degree polynomial to accurately represent the underlying patterns.

In our initial SVM model fitting attempts, we found that making the data processing more complicated doesn't always lead to better Mean Squared Error (MSE) scores in SVM Gridsearch, and sometimes a simpler dataset performs better than a dataset that was more processed. To explore this further, we decided to compare two datasets: one with less processing called "Stan_X_train_df" and another which was processed with more complexity, referred to as "OHE_X_train" in the Data Pre-Processing section.

| Dataset | OHE_X_train | *Stan_X_train_df* |
|---|---|---|
| Standardscaler | √ | √ |
| One-Hot Encoding | √ | |

*Table 8. SVM datasets*

### 4.4.2 SVM - Best parameters and corresponding MSE

We then perform GridSearchCV to find the best parameters for a SVR model. Here, we do not perform the PCA, i.e. the ‘param_grid’ used will omit ‘pca_n_components’. The results are as follows:

| | |
|---|---|
| **Dataset** | *Stan_X_train_df* |
| **Parameter Values** | *{C=1000, gamma='auto', kernel='rbf'}* |
| **Corresponding MSE** | 764.43 |

*Table 9. Best parameters and corresponding MSE on Stan_X_train_df*

| | |
|---|---|
| **Dataset** | OHE_X_train |
| **Parameter Values** | *{C=1000, gamma='auto', kernel='rbf'}* |
| **Corresponding MSE** | 1336.04 |

*Table 10. Best parameters and corresponding MSE on OHE_X_train*

We observed the lower Mean Squared Error (MSE) when fitting the SVM model on the less processed dataset, Stan_X_train_df, compared to OHE_X_train that we experimented with. This unique performance trend appears to be specific to SVM. In section 7.4 (Findings - SVM), we will delve into potential explanations for this notable observation.

We attempted to reduce the MSE further by only using a subset of the predictors, where we utilised the best subset selection method. After testing several combinations, we came across a subset that provided quite a substantial improvement in MSE. Table 11 below shows the results.

| | |
|---|---|
| **Dataset** | *Stan_X_train_df* |
| **Parameter Values** | *{C=1000, gamma='auto', kernel='rbf'}* |
| **Subset of predictors** | *['clickVolume', 'avgFinalUnitPrice', 'ma14SalesVolume', 'attribute1', 'attribute2', 'meanUserLevel']* |
| **Corresponding MSE** | 579.70 |

*Table 11. Best parameters and corresponding MSE on Stan_X_train_df while using subset of predictors*

This subset of predictors provide the information one would expect is most closely related to sales, such as click volume, price and previous sales volume. By focusing on the best subset selection, we are able to remove irrelevant features that did not contribute in the reduction of the MSE, which makes the model easier to interpret and less prone to overfitting, hence making it more generalisable.

# 5. Test Prediction

| Model | Best Result |
|---|---|
| Linear Regression (LASSO) | 1412.90 |
| Linear Regression (Ridge) | 1366.20 |
| Random Forest | 978.68 |
| SKLearn Gradient Boosting | 862.54 |
| CatBoost | 797.05 |
| XGBoost | 904.23 |
| SVM | 579.70 |

*Table 12: Best results for each model with its corresponding MSE*

As seen from the table above, SVM yielded the best MSE amongst all the other models. Thus, for our sales prediction on the test dataset, we decided to use the SVM model.

# 6. Inventory Decision

## 6.1 Empirical Distribution

To understand what type of distribution to use for our inventory model decision, we first decided to find out how the residual error between our prediction and the actual sales for the training set was distributed. To do this, we plotted the cumulative distribution frequency (CDF) graph of our error, as well as for the best fitting Normal, Exponential and Uniform Distribution curves. Figure 8 below shows the results of the plot.

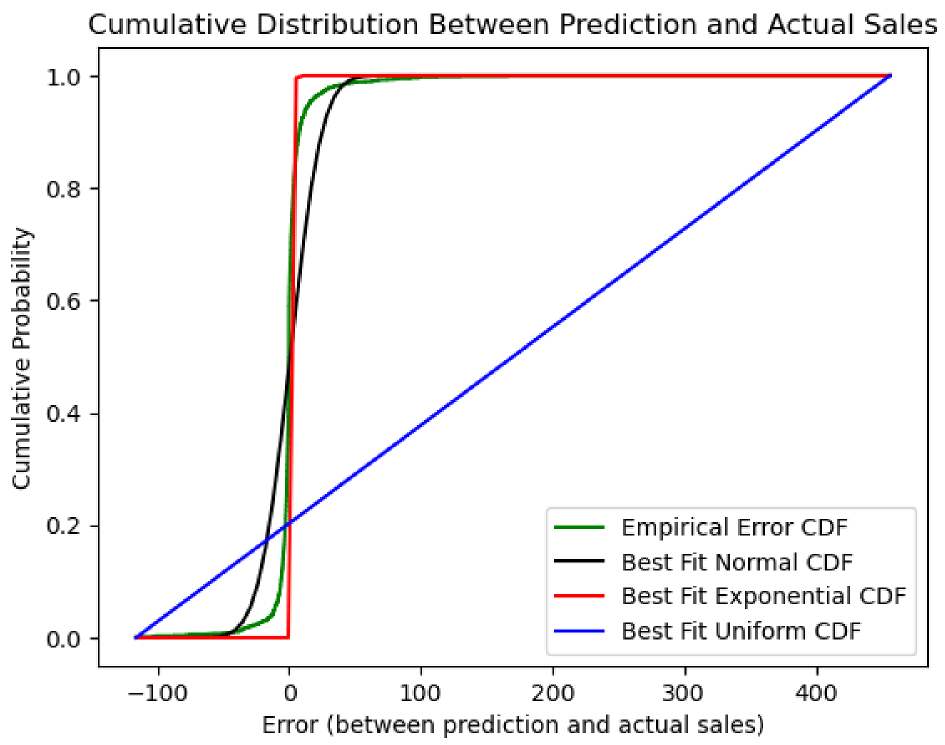


*Fig 8. A cumulative distribution frequency plot of the difference between prediction and actual results on train data.*

From the plot, it was clear that either a Normal Distribution or Exponential Distribution best fit our error distribution. We decided to explore both approaches further.

## 6.2 Normal Distribution Approach

In order to calculate our inventory decision using the normal distribution approach, the following formula was used.

$$\text{Optimal decision: } \mu + \Phi^{-1}(C_u/(C_u + C_o)) * \sigma$$

From the project description we could ascertain the overage cost as 4 and the underage cost as 8.

**μ:** Our sales prediction from the SVM model is taken to be the mu.
**σ:** An estimate of the standard deviation is calculated based on the training dataset, using the following formula:

$$\text{std\_est} = \sqrt{\frac{\sum_{i=1}^{n}(\text{Predicted}_i - \text{Actual}_i)^2}{(n-p-1)}}$$

*Fig 9. Formula used for estimating the standard deviation based on training dataset*

n specifies the number of observations, while p specifies the number of features in our dataset.

$\Phi^{-1}$: Inverse of the CDF of the standard normal distribution.

## 6.3 Exponential Distribution Approach

The second approach we tested is to assume the sales are exponentially distributed and estimate the lambda based on the samples. Our estimate of the lambda is 1/mean(residual_error) from the training dataset. Using this we calculate the y* and add it to our sales prediction. We also subtract the mean of the residual error from this amount, giving our inventory prediction.

Figure 10 below shows our formula for calculating y*, and Figure 11 shows our formula for calculating the optimal inventory decision.

$$y^* = -\frac{1}{\lambda}\log(1-t)$$

*Fig 10. Formula used for calculating y**

**t** = (Underage cost / (Underage cost + overage cost)) = ⅔
***λ*** = 1/mean(residual error between actual sales and prediction)

$$\text{Optimal Decision} = \mu - \text{mean}(\text{residual error}) + y^*$$

*Fig 11. Formula used for calculating optimal decision*

**μ:** Our sales prediction from the SVM model is taken to be the mu.

## 6.4 Comparing both approaches on training dataset

To compare the performance of both approaches, we calculated the profit each approach generates on the training dataset. We calculated the inventory decision by both approaches, and used the actual sales value as the demand. The following formula was used for profit calculation:

$$\text{Profit} = (p - s)\min(D,y) - (c-s)y$$

| Variable | Meaning | Value in our case |
|---|---|---|
| p | Sales price of item | 20 |
| c | Cost price of item | 12 |
| s | Salvage value of item | 8 |
| D | Demand of item | Actual sales amount |
| y | Inventory decision | Inventory decision using either approach |

*Table 13. Explanation of profit formula*

The results of the profit earned from both approaches is outlined in the table below. As the exponential approach earned a higher profit, we decided to use it for our testing dataset inventory decision.

| Approach | Profit Earned (from training dataset) |
|---|---|
| Normal Distribution | 351840.0 |
| Exponential Distribution | 375204.0 |

*Table 14. Profit results from both approaches*

# 7. Findings

## 7.1 Linear Regression

Due to the complexity of the problem, it is of no surprise that linear regression was the model that performed the worst out of all the models as it is highly susceptible to any slight outliers in the dataset, with erroneous predictions on the unseen dataset. Also, using linear regression assumes that there is an underlying linear relationship between the features and the result.

## 7.2 Gradient Boosting

While exploring Gradient Boosting, we have experimented on 3 different variations of Gradient Boosting Regressors which includes SKLearn Gradient Boosting, CatBoost, and XGBoost. Out of the three, CatBoost outperformed the other 2.

## 7.3 Support Vector Machine

The SVM regressor generally yields a lower MSE score compared to other models. Comparing the ‘linear’ kernel with ‘poly’ and ‘rbf’ kernel, we see a lower MSE in the non-linear kernel, which indicates that the underlying relationship between sales and the features is likely non-linear. This suggests that the sales data may have intricate patterns or dependencies that a linear model cannot capture effectively. The flexibility of non-linear kernels, such as 'poly' and 'rbf,' enables the SVM regressor to better adapt to these complex relationships, resulting in a more accurate prediction of sales. Moreover, The 'rbf' kernel in SVM tends to perform better because it can understand and represent more complex relationships between features and the target.

We also noticed that some of the sales SVM predicted were negative values, which should not be possible. Hence, we added a ReLU function on all predictions.

As mentioned before, we trained our model using two datasets, OHE_X_train and Stan_X_train_df, each with varying degrees of pre-processing. Notably, our findings indicate that SVM models perform better **without One-Hot Encoding**.

| **Dataset** | OHE_X_train | *Stan_X_train_df* |
|---|---|---|
| **MSE** | 1336.04 | 764.43 |

*Table 15. Comparison of SVM results on different datasets*

Upon employing One-Hot Encoding, we observed a significant rise in dimensionality, expanding from the original dataset's dimensions of (1576, 19) to (1576, 154). This notable increase in dimensionality introduces the potential challenge of the Curse of Dimensionality, a sensitivity in SVMs that could lead to overfitting and an increase in MSE during 10-Fold Cross Validation (Bengio, 2005). In local kernel algorithms like SVM, the risk of overfitting rises with the number of features, as the model may capture patterns specific to the training data but fails to generalise well to test data (Bengio, 2005). Consequently, we opted to fit the SVM with less processed data *Stan_X_train_df* without One-Hot Encoding.

# 8. Conclusion

After performing data analysis and trying out different machine learning models, we have concluded that SVR gives the best results for predicting the model. The SVR model is being used to estimate the sales for the test dataset. Also with the formula described in Chapter 6.3, we calculate the optimal inventory decision for the test dataset.

# 9. References

Yoshua Bengio, Olivier Delalleau, Nicolas LeRoux, (2005), *The Curse of Dimensionality for Local Kernel Machines,* P.O. Box6128, DowntownBranch, Montreal, H3C3J7, Qc, Canada