IE4211 Modelling and Analytics
AY 2021/2022 Semester 2

Project Report
Group 4

Lim Main Ray	A0216824E
Cheng Zheng Ting	A0221611Y
Gina Leow Hui Jie	A0189185M
Linn Htet Aung	A0220705U
Linda Francis	A0188686A

1. Introduction

This project aims to predict the sales of the test data, with an objective of obtaining an accurate prediction. Following this, an inventory decision is to be made and the objective is to maximize the profit induced by our inventory decision. This report will first discuss the data exploration and will be followed by the data pre-processing performed on the provided datasets. Through hyperparameter tuning and evaluation of the various models we trained, the best prediction model is obtained. This report will also elaborate on how we made our inventory decision.

2. Data Exploration

This section aims to discuss the findings from our data exploration process.

2.1 Data Types

It was observed that the attributes are either of integer type or float type. Additionally, we also noticed that ‘productID’, ‘brandID’ and ‘weekday’ are considered as numerical data. Hence, before we train our model, these attributes are being converted to categorical data types.

```
data_train.dtypes
```

```
productID                int64
brandID                  int64
attribute1               int64
attribute2               int64
attribute3               int64
attribute4             float64
clickVolume              int64
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 1. Data Types in dataset

2.2 Unique values

There are 76 distinct productID, 30 distinct brandID, 4 distinct attribute1, 7 distinct attribute2, and 33 distinct attribute3 values in the dataset. The 7 distinct values in weekday represents the seven days of the week.

```
data.nunique()
```

```
productID                76
brandID                  30
attribute1                4
attribute2                7
attribute3               33
attribute4             2314
clickVolume            1244
avgOriginalUnitPrice    386
avgFinalUnitPrice      1938
ma14SalesVolume         769
weekday                   7
meanAge                 726
gender                  371
meanEducation           408
maritalStatus           353
plus                    381
meanPurchasePower       363
meanUserLevel           675
meanCityLevel           492
sales                   159
dtype: int64
```

Fig 2. # of distinct values in each attribute

Additionally, we also noticed that there were 5 extra distinct values in attribute3 for the train data, when compared to the test data. Hence, to allow for the one-hot-encoding to work, we appended the train and test dataset together, convert the necessary attributes to categorical data, before splitting them up again.

2.3 Correlation Matrix

The correlation matrix below shows that avgFinalUnitPrice has a strong correlation with avgOriginalUnitPrice at 0.89. There is also a high correlation between clickVolume and sales, at 0.65, and attribute1 and attribute2, at 0.58.

While we acknowledge the correlation between certain features, these features will not be removed before the model selection so as to get an overview of the model performance with all the features in-place.

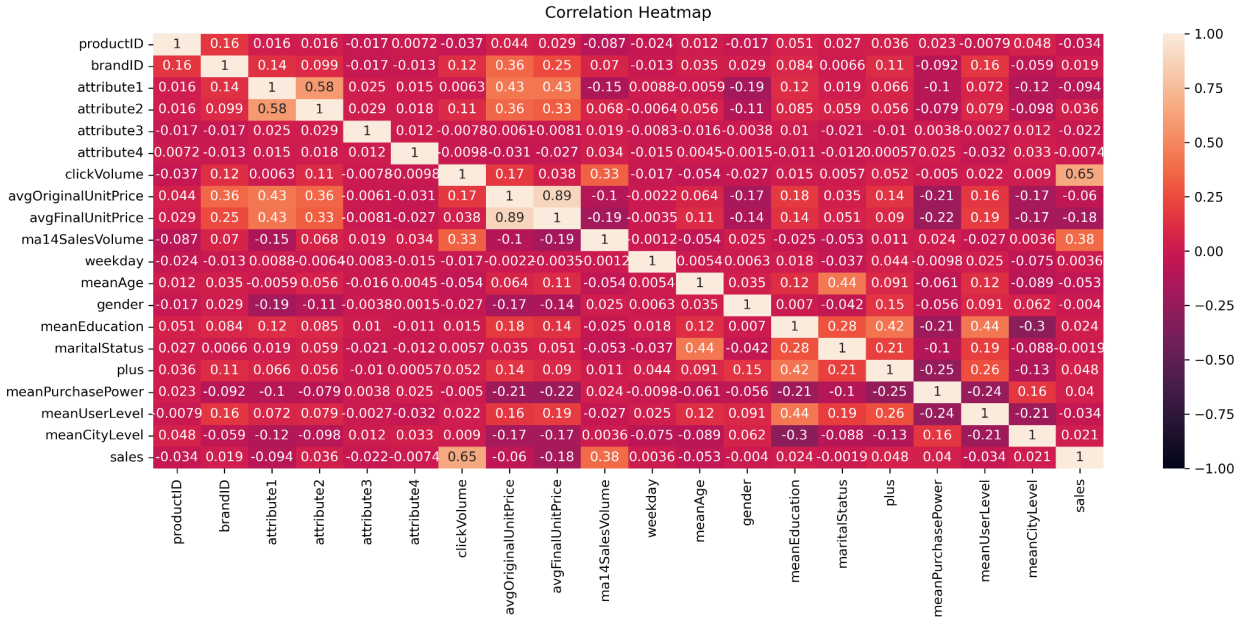


Fig 3. Correlation Matrix

3. Data Pre-Processing

In this section, we will explain the data pre-processing steps that we have taken, to transform the data according to the requirements of the different models.

3.1 Original dataset

Original dataset (given):

- "Data-train.csv"
- "Data-test.csv"
- "A_sales_train.csv"

3.2 Original dataset with One Hot Encoding

Original dataset with categorical values One-Hot-Encoded:

- "B_encoded_test_data.csv"
- "B_encoded_train_data.csv"

The following are the steps that we took to pre-process the data for this section:

1. Combine train and test data (to account for different number of attribute3 categorical values)
2. One-Hot-Encoding
3. Split into train and test data again

3.3 Normalized dataset with One Hot Encoding

Normalized original dataset with One-Hot-Encoding, intended for Principal Component Analysis (PCA)

- "A_Normalised_test_data.csv"
- "A_Normalised_train_data.csv"

The following are the steps that we took to pre-process the data for this section:

1. Combination of train and test data (to account for different number of attribute3 categorical values)
2. One-Hot-Encoding
3. Normalization
4. Split back into train and test data

3.4 Log Sales

The sales column is extracted from the original dataset and undergoes a natural log.

- "A_Log_sales_train.csv"

The following are the steps that we took to pre-process the data for this section:

1. Extract the sales column
2. log(sales)

By undergoing this log transformation, it will bring the sample points closer together as shown in the figure below:

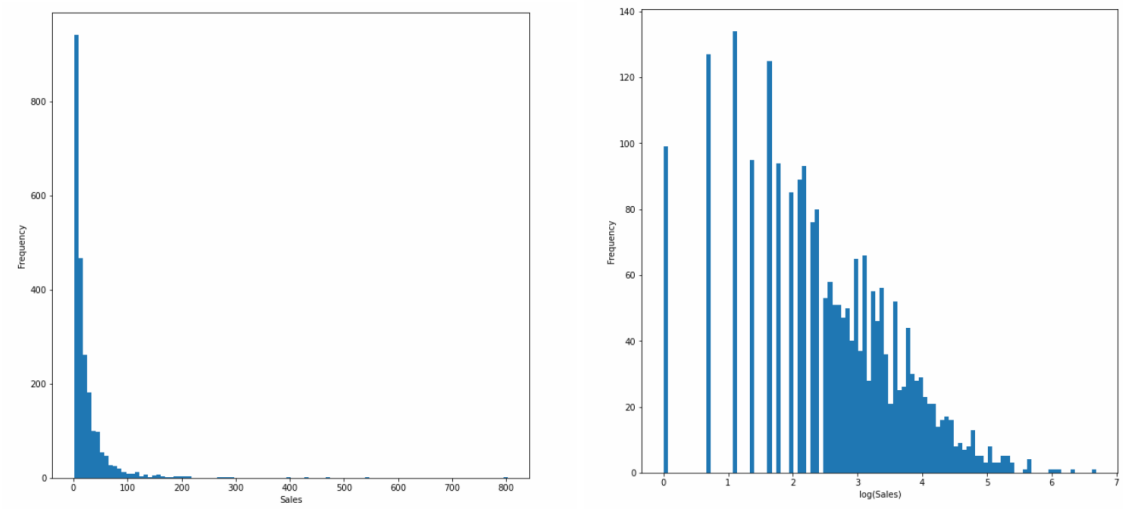


Fig 4. Histogram of sales (left), Histogram of log(sales) (right)

As seen above, the sales data has a large range of values and this may cause some models to underperform. We will compare the performance for all the models to test our hypothesis.

4. Principal Component Analysis (PCA)

For a selected set of models we have 2 different versions of the model, one with PCA and another without PCA to explore the effects and appropriate use case of PCA for this project.

Models that use PCA will use the following datasets to train.

- "A_Normalised_train_data.csv": as the predictors
- "A_sales_train.csv": as the response
- "A_Log_sales_train.csv": as the response

Both log(sales) and sales are used in separate models to compare the performance.
If PCA is not used, it will be explicitly stated in the name of the file, otherwise PCA can be assumed to be used in transforming the data. Refer to Figure 1 in the Appendix for a summary of the dataset used.

5. Inventory Decision

This section elaborates more on how the inventory decision is made.

5.1 Obtaining Empirical Distribution

Firstly, the residual is determined. For every prediction, the residual distribution is used to account for noise at each sample point. Using the Empirical Cumulative Distribution Function (CDF), the distribution of the residual is found, without assuming that the distribution of the residuals should follow any distribution.

Procedures to obtain Empirical Distribution:

1. Split the data into train (80%) and test datasets (20%).
2. Train a model using the train dataset
3. Residual = model.predict(test_predictors) - test_response
4. Use the list of Residuals from step 3 to generate an Empirical Cumulative Distribution Function (CDF).

```
size = residual.shape[0]
step=1/size
jumps = [step*i for i in range(1,size+1)]
Sample = sorted(residual)
plt.plot(Sample,jumps)
```

```
[<matplotlib.lines.Line2D at 0x21c667fca00>]
```

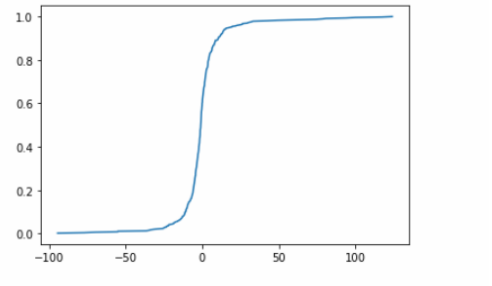


Fig 5. Empirical CDF

Empirical distribution allows us to approximate the distribution without prior assumptions of the distribution. Each residual is treated like a sample point. Thus, an Empirical cumulative distribution function graph could be constructed to approximate the underlying distribution.

Example:

Suppose 6 numbers are generated from a random number generator and the distribution of the random number is unknown to us. However, given the following outcome we can approximate the unknown distribution

Number (X)	1	2	3	4
Frequency	1	2	2	1
Pr(X = x)	1/6	2/6	2/6	1/6
Pr(X <= x)	1/6	3/6	5/6	1

In the same manner, we generate the empirical CDF.

Next, we can find the inverse of the CDF using the function below:

```
def inv(sample,jump,area):
    size = len(sample)
    i=0
    temp=jump[i]
    while temp<area:
        i+=1
        temp=jump[i]

    return sample[i]
```

Fig 6. Inverse Function

Iterate through the jump sequence until the area (temp) is more than or equal to the given area.

In hindsight, we could have used Binary Search instead of linear search to improve the time complexity.

5.2 Inventory Decision model

From the given information, price, cost, and salvage value are $20, $12, and $8 respectively. Given these, the values below were obtained.

Overage cost co = c – s = $12 - $8 = $4
Underage cost cu = p – c = $20 - $12 = $8
Critical Fractile $\alpha = \frac{c_u}{c_o + c_u} = \frac{p-c}{p-s} = \frac{20-12}{20-8} = 2/3 = 0.667$ (3 d.p.)

$$Offset = Inverse\ of\ Empirical\ CDF(Critical\ Fractile)$$

$$Inventory = [PredictedSales + Offset]$$

The inverse of the empirical CDF, evaluated at the critical fractile α (2/3), is used to calculate the offset. The inventory decision for each row is determined by adding the offset to the predicted values from the trained model.

$$Profit = (Price - Salvage) * \text{Min}(Demand, Inventory) - (Cost - Salvage) * Inventory$$

The profit is then determined from the demand (actual sales) and the inventory decision using the above formula.

However, it is difficult to compare profit when using 10-fold cross-validation since each sample may have different profit potential depending on the sales. We came up with a metric to resolve this issue.

$$MaxProfit = Demand * (Price - Cost)$$

Maximum profit is the profit when sales/demand are predicted perfectly and the exact inventory is allocated.

$$Efficiency = RealisedProfit/MaxProfit$$

We then define efficiency in the above formula. The higher the efficiency the better the model performance since more of the potential profits are being earned. This allows us to use 10-fold cross-validation for efficiency and compare across different models and distributions.

6. Models

6.1 HyperParameter Tuning

In order to find the optimal hyperparameter for each model, our group employed *GridSearch* & *Pipeline* to optimize the process of finding the most optimal hyperparameters and n_components for principal component analysis (PCA) for each model by reducing mean squared error. GridSearch is a function in sklearn where users are able to input possible values for the hyperparameters for the model and GridSearch will run through each possible permutation and output the mean squared error for each permutation. Pipeline is also another function in sklearn. Since we will first transform the data using PCA then train each model, we will need pipeline to connect output of PCA to the model. Thereafter, we use gridsearch to find the most optimal number of components for principal component analysis and hyperparameters for each model.

6.2 Model Evaluation Metric

We will look at several evaluation metrics to evaluate each model: *Maximum Attainable Profit*, *Fraction of Maximum Attainable Profit* and *Mean Squared Error (MSE)* for predicting sales. Firstly, we will look to minimize MSE that was found using GridSearch Cross-Validation. Next, maximum attainable profit refers to the maximum profit that could be generated if all predictions are correct. The model's performance would be evaluated by the percentage of max attainable profit it is able to attain.

6.3 Summary of Key Findings

The models are trained with the datasets as shown in Figure 1 in the Appendix. This section aims to summarize and compare the results of each model by comparing cross validation MSE.

	Cross Validation MSE
4.4B_Bagging(No_PCA)	478.68
5.1_Quantile_regression_forest(No_PCA)	484.88
4.2B_Random_forest(No_PCA)	489.66
4.3B_Gradient_boosting(No_PCA)	519.09
3.1B_KNN(No_PCA)	759.19
4.2B_Decision_tree(No_PCA)	1007.49
4.3A_Gradient_boosting	1680.36
4.2A_Random_forest	1714.13
4.1A_Decision_tree	1839.94
2.6_Poly_reg_lasso	1958.82
3.1A_KNN	1959.39
2.4_Lin_reg_lasso	3978.06
2.5_Poly_reg_ridge	4632.62
2.3_Lin_reg_ridge	4632.62
2.1A_Lin_reg	9838.15
2.2A_Poly_reg	11294.97
2.2B_Poly_reg(No PCA)	63463562929760522 24.00
2.1B_Lin_reg(No_PCA)	63463562929760522 24.00

Fig 7. 10-Cross Validation MSE

After training the model, it can be observed that the *Tree-model with Bagging* (using the original dataset with No PCA adjustments) is the best prediction model as it has the lowest cross-validation MSE of 478.68.

An out-of-sample analysis will be carried out to identify the best performing models.

	Normal	Exponential	Empirical
4.4B_Bagging(No_PCA)	0.66	0.74	0.74
5.1_Quantile_regression_forest(No_PCA)	0.65	0.74	0.74
4.2B_Random_forest(No_PCA)	0.65	0.74	0.74
4.3B_Gradient_boosting(No_PCA)	0.65	0.71	0.72
4.2B_Decision_tree(No_PCA)	0.55	0.66	0.66
3.1B_KNN(No_PCA)	0.59	0.66	0.65
4.2A_Random_forest	0.27	0.32	0.33
4.3A_Gradient_boosting	0.23	0.32	0.33
2.6_Poly_reg_lasso	0.27	0.28	0.32
3.1A_KNN	0.26	0.25	0.30
4.1A_Decision_tree	0.24	0.26	0.29
2.4_Lin_reg_lasso	0.12	0.20	0.24
2.5_Poly_reg_ridge	0.09	0.19	0.23
2.3_Lin_reg_ridge	0.09	0.19	0.23
2.1A_Lin_reg	-0.17	0.09	0.14
2.2A_Poly_reg	-0.19	0.08	0.14
2.1B_Lin_reg(No_PCA)	-7926499.81	-1151773.28	-1048389.16
2.2B_Poly_reg(No PCA)	-7926499.81	-1151773.28	-1048389.16

Fig 8. out-of-sample efficiency of predicting the profit

From the table above, the top 5 best performing models will be shortlisted for further analysis. The five models are:

1. Bagging
2. Quantile Regression Forest
3. Random Forest
4. Gradient Boosting
5. Decision Tree

It was noted that the better performing models are tree-based models trained on the original dataset (with no PCA adjustments).

	In-Sample-MSE
DecisionTree	54.353767
GradientBoosting	54.550886
Bagging	66.878244
RandomForest	70.676721
RandomForestQuantileRegressor	78.050718

Fig 9. In-Sample MSE

	In-Sample-Efficiency
Bagging	0.902692
RandomForestQuantileRegressor	0.896589
RandomForest	0.884954
GradientBoosting	0.859148
DecisionTree	0.839818

Fig 10. In-Sample Efficiency

From the further in-sample analysis, it was found that Decision tree, Gradient Boosting and Bagging have the better performing in-sample MSE and Bagging, Random Forest Quantile Regression and Random Forest have the better in-sample Efficiency. Further analysis will be carried out on the out-of-sample dataset.

	Out-Sample-MSE
RandomForestQuantileRegressor	296.629900
Bagging	314.264443
RandomForest	317.856150
GradientBoosting	489.094265
DecisionTree	1135.283027

Fig 11. Out of Sample MSE

From the out-of-sample analysis, Random Forest Quantile Regression was found to be the best performing model (MSE: 296.63) and is followed by Bagging (MSE: 314.26).

Since *Bagging* performs consistently *in both in-sample and out-of-sample datasets* (refer to Fig 8,9,10,11), it will be the chosen model to predict sales in order to make the inventory decision.

7. Interesting findings

1. For linear models, models trained using PCA-adjusted datasets perform significantly better than those trained using the unmodified but one-hot-encoded dataset.

This is because PCA will reduce the issue of having correlated features that decrease the performance of linear models. This improvement has been observed in the summary table above.

2. For tree-based models, the original dataset will give a better result than a dataset adjusted with PCA.

This is because the application of PCA will require categorical data to be transformed with One Hot Encoding (OHE). This poses a limitation for the tree-based models since the introduction of new feature columns will be trained as individual features by the model. The tree models will be restricted to a subset of the features, with a significantly greater number of features that are merely dummy variables taking values of 1 and 0, tree models cannot effectively split the data. This inturn will affect the model accuracy and therefore the original dataset will be used during tree-based model training.

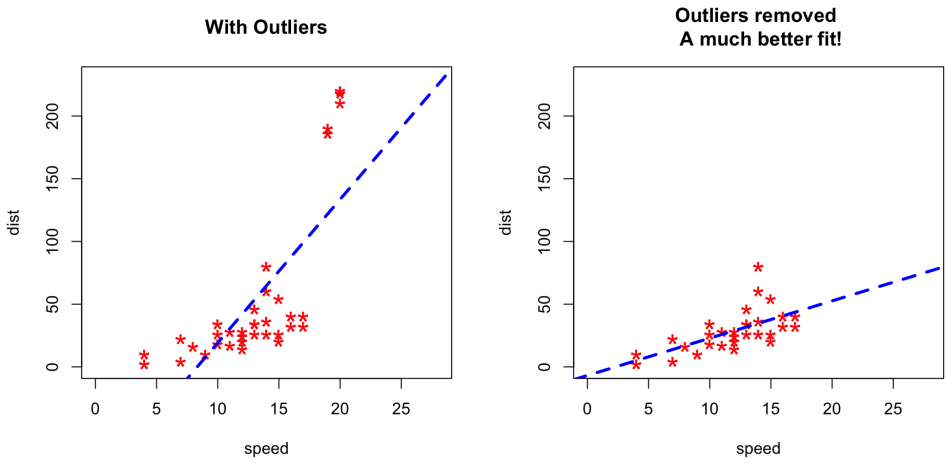
3. Tree models are robust to outliers, correlated features and do not require one-hot-encoding of categorical variables.

Tree models are robust to outliers because outliers do not play a significant role in determining the location of the split. Tree models are robust to correlated features because if two features are highly

correlated, only one of them will be chosen for the split. Tree models do not require one-hot-encoding because even splitting at a non-integer point is analogous to cleanly splitting different categories.

4. Linear models tend to perform better on log(sales) as response than on sales. However, there is no clear trend for tree models. Refer to Figure 3 in the Appendix.

Log(sales) has the effect of bringing the data points closer. The reason linear models perform better may be due to the reduced distance of the points. For linear models, points that are far away have a disproportionate influence on the model coefficient. The existence of outliers results in poor fitting of the model as demonstrated in the diagram below.



5. We hypothesize that a good inventory model compensates for a bad prediction model. For example, when the model was trained using Gradient Boosting (No PCA), the cross-validation MSE was 519.09, and the in-sample efficiency was 0.74. Meanwhile, when the model was trained using Decision Tree (No PCA), the cross-validation MSE was 1007.49, but the in-sample efficiency was still 0.74. Even though the Decision Tree (No PCA) performed worse, as seen from the higher MSE, the in-sample efficiency was still the same as the model with lower MSE. One possible explanation is that when a prediction model is bad, the residual will be larger, resulting in a larger offset, and this slightly compensates for the wrong prediction. The models tend to underestimate the true sales, thus residual is skewed towards the right. Moreover, the critical fractile is more than 0.5. Thus the offset tends to be positive and correct the underestimation.

8. Conclusion

In conclusion, our group found that *bagging* will be the best model to predict sales due to its consistency in model performance for both in-sample and out-of-sample datasets. Further studies are required to explore a greater range of hyperparameters and also to investigate the claims in our interesting findings.

Note for grader:

Notebooks that start with “6.X” are where the final top 5 models are trained on the whole train data and used to generate the sales prediction and inventory decisions for the test data. The notebook “Group4.ipynb” is used to select one of the results generated from the earlier step.

9. References

Quantile Regression Forests - Scikit-garden. (2022). Github.io. https://scikit-garden.github.io/examples/QuantileRegressionForests/

Outlier. (2022). Github.io. https://lumesserschmidt.github.io/stats_CoronaNet/other_outlier.html

10. Appendix

Figure 1: Models trained and datasets used for each model

	Table of Models Trained	Predictors	Predictors	Response	Response	Predictors	Predictors	Predictors	Predictors + Response
	Model	A_Normalised_train_data	A_Normalised_test_data	A_sales_train	A_Log_sales_train	B_encoded_train_data	B_encoded_test_data	Data-test	Data-train
Model Selection	2.1A Linear Regression	☑	☐	☑	☑	☐	☐	☐	☐
	2.1B Linear Regression (No PCA)	☐	☐	☑	☑	☑	☐	☐	☐
	2.2A Polynomial Regression	☑	☐	☑	☑	☐	☐	☐	☐
	2.2B Polynomial Regression (No PCA)	☐	☐	☑	☑	☑	☐	☐	☐
	2.3 Linear Regression (Ridge)	☑	☐	☑	☑	☐	☐	☐	☐
	2.4 Linear Regression (Lasso)	☑	☐	☑	☑	☐	☐	☐	☐
	2.5 Polynomial Regression (Ridge)	☑	☐	☑	☑	☐	☐	☐	☐
	2.6 Polynomial Regression (Lasso)	☑	☐	☑	☑	☐	☐	☐	☐
	3.1A K-Nearest Neighbour	☑	☐	☑	☑	☐	☐	☐	☐
	3.1B K-Nearest Neighbour (No PCA)	☐	☐	☑	☑	☑	☐	☐	☐
	4.1A Decision Tree	☑	☐	☑	☑	☐	☐	☐	☐
	4.1B Decision Tree (No PCA)	☐	☐	☐	☑	☐	☐	☐	☑
	4.2A Random Forest	☑	☐	☑	☑	☐	☐	☐	☐
	4.2B Random Forest (No PCA)	☐	☐	☐	☑	☐	☐	☐	☑
	4.3A Gradient Boosting	☑	☐	☑	☑	☐	☐	☐	☐
	4.3B Gradient Boosting (No PCA)	☐	☐	☐	☑	☐	☐	☐	☑
	4.4B Bagging (No PCA)	☐	☐	☐	☑	☐	☐	☐	☑
	5.1 Quantile Regression Forest (No PCA)	☐	☐	☐	☑	☐	☐	☐	☑
Prediction	6.1 Decision tree Prediction	☐	☐	☐	☐	☐	☐	☑	☑
	6.2 Random Forest Prediction	☐	☐	☐	☐	☐	☐	☑	☑
	6.3 Gradient Boosting Prediction	☐	☐	☐	☐	☐	☐	☑	☑
	6.4 Bagging Prediction	☐	☐	☐	☐	☐	☐	☑	☑
	6.5 Random Forest Quantile Regressor Prediction	☐	☐	☐	☐	☐	☐	☑	☑

Figure 2: Table of optimal Hyperparameters found using GridSearch

Table of Hyperparameters

Model	n_components (PCA)	degree (Polynomial Feature)	SLR_alpha	n_neighbors	ccp_alpha	n_estimators	learning_rate	min_samples_split	bootstrap	bootstrap_features
2.1A Linear Regression	100									
2.1B Linear Regression (No PCA)										
2.2A Polynomial Regression	100	1								
2.2B Polynomial Regression (No PCA)		1								
2.3 Linear Regression (Ridge)	170		1000							
2.4 Linear Regression (Lasso)	100		0.05							
2.5 Polynomial Regression (Ridge)	150	1	1000							
2.6 Polynomial Regression (Lasso)	170	2	0.1							
3.1A K-Nearest Neighbour	10			6						
3.1B K-Nearest Neighbour (No PCA)				10						
4.1A Decision Tree	150				0.005					
4.1B Decision Tree (No PCA)					0.5					
4.2A Random Forest	170				0.005	1000				
4.2B Random Forest (No PCA)					0.05	1000				
4.3A Gradient Boosting	170				0.005	100	0.1	0.02		
4.3B Gradient Boosting (No PCA)					0.1	100	0.5	0.02		
4.4B Bagging (No PCA)						1000			TRUE	FALSE
5.1 Quantile Regression Forest (No PCA)										
6.1 Decision tree Prediction					0.5					
6.2 Random Forest Prediction					0.05	1000				
6.3 Gradient Boosting Prediction					0.1	1000	0.5	0.02		
6.4 Bagging Prediction						1000			TRUE	FALSE
6.5 Random Forest Quantile Regressor Prediction						100				

Figure 3: Comparison between using Sales or Log(Sales) as response for models trained (Out-of-sample MSE and in-sample MSE)

	Out-sample MSE		In-sample MSE		
Model	Sales	log(sales)	Sales	log(sales)	10-fold Efficiency (Empirical)
2.1A Linear Regression	1879.52	1258.17	-	-	0.14
2.1B Linear Regression (No PCA)	487.95	523.72	-	-	-1048389.16
2.2A Polynomial Regression	1908.12	1271.10	-	-	0.14
2.2B Polynomial Regression (No PCA)	487.95	523.72	-	-	-1048389.16
2.3 Linear Regression (Ridge)	1566.16	1105.79	-	-	0.23
2.4 Linear Regression (Lasso)	1903.82	1086.55	-	-	0.24
2.5 Polynomial Regression (Ridge)	1534.36	1084.09	-	-	0.23
2.6 Polynomial Regression (Lasso)	1710.43	1394.15	-	-	0.32
3.1A K-Nearest Neighbour	1092.33	1097.07	-	-	0.30
3.1B K-Nearest Neighbour (No PCA)	476.04	451.12	-	-	0.65
4.1A Decision Tree	2987.02	1224.77	-	-	0.29
4.1B Decision Tree (No PCA)	427.40	699.43	-	-	0.66
4.2A Random Forest	2880.85	934.24	-	-	0.33
4.2B Random Forest (No PCA)	311.91	377.49	-	-	0.74
4.3A Gradient Boosting	2793.47	929.33	-	-	0.33
4.3B Gradient Boosting (No PCA)	519.09	646.44	-	-	0.72
4.4B Bagging (No PCA)	316.80	288.8	-	-	0.74
5.1 Quantile Regression Forest (No PCA)	311.98	-	-	-	0.74
6.1 Decision tree Prediction	1135.28	-	54.35	-	-
6.2 Random Forest Prediction	317.86	-	70.68	-	-
6.3 Gradient Boosting Prediction	489.09	-	54.55	-	-
6.4 Bagging Prediction	314.26	-	66.88	-	-
6.5 Random Forest Quantile Regressor Prediction	296.63	-	78.05	-	-