UNIVERSITY OF LONDON

Machine Learning ST3189

Big Mart Sales Prediction in Machine Learning

Name: Cynthia Khong Qing

Student Number: 210407112

Table of Contents

Introduction

Retail is an industry that heavily leverages analytics to optimize its business processes. It is marked by diverse consumer behaviours, product preferences, and ever-changing market trends. Data science techniques play a crucial role in intelligently managing tasks such as inventory management, identifying target customer segments, and refining retail product offerings.

For this project, we have chosen to analyse the Big Mart Sales Dataset, comprising 8523 rows and 12 columns including various attributes such as product weight and outlet size. Our analysis is conducted using Python within a Jupyter Notebook environment.

Our primary objective of this project is to build robust predictive models to forecast "Item Outlet Sales" to automate the assessment of sales performance across different outlets and products. Our approach includes both unsupervised and supervised learning techniques. Unsupervised learning such as k-means clustering and principal component analysis (PCA) will help to uncover hidden patterns and relationships within the dataset. For supervised learning algorithms including regression and classification models are used to build predictive models based on identified patterns.

By doing so, we aim to gain valuable insights into the key factors that influence sales performance. These insights will enable retailers to make informed decisions and drive business growth effectively.

Data Set and Exploratory Data Analysis (EDA)

2.1 Dataset Description

A dataset sourced from a reliable source - Kaggle.com, titled "Big Mart Sales Dataset", has been selected for this project. This dataset includes various characteristics of items and outlets and consists of 8523 rows and 12 columns, with the target variable as "Item_Outlet_Sales". This dataset is considered a small-sized dataset and contains sufficient data for meaningful analysis.

The structure of the dataset is shown below:

Variable Name	Description	Sample Data
Item_Identifier	Unique product ID	FDA15; DRC01; …
Item_Weight	Weight of product	9.3; 5.92; …
Item_Fat_Content	Content of product (Low fat or Regular)	Low Fat; Regular; …
Item_Visibility	The percentage of total display area of all products in a store allocated to the product	0.016047301; 0.019278216; …
Item_Type	Category of the product	Dairy; Soft Drinks; …
Item_MRP	Maximum Retail Price of the product	249.8092; 48.2692; …
Outlet_Identifier	Unique store ID	OUT049; OUT018; …
Outlet_Establishment_Year	The establishment year of the store	1999; 2009; …
Outlet_Size	The size of the store	Medium; High; …
Outlet_Location_Type	The type of city in which the store is located	Tier 1; Tier 3; …
Outlet_Type	Type of the store	Supermarket Type 1; Grocery Store; …
Item_Outlet_Sales	Sales of the product	3735.138; 2097.27; …

2.2 Substantive Issue

Determine the most influential variables on item outlet sales for Big Mart. To address this, a few research questions are conducted below:

1. What are the primary factors influencing the Item Outlet Sales in the dataset?
2. How do product attributes, store characteristics, and other factors contribute to variations in sales performance?

3. Which predictive model demonstrates the highest performance in accurately forecasting item outlet sales?

2.3 Exploration Data Analysis (EDA)

All steps in this part will be conducted using Python code. Prior to the EDA, data cleaning and data pre-processing are performed to prepare the dataset for exploration.

A quick understanding of data is performed:

From Figure 2, we can observe that there are 1463 and 2410 missing values of Item Weight and Outlet Size respectively.

```
Data columns (total 12 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Item_Identifier            8523 non-null   object
 1   Item_Weight                7060 non-null   float64
 2   Item_Fat_Content           8523 non-null   object
 3   Item_Visibility            8523 non-null   float64
 4   Item_Type                  8523 non-null   object
 5   Item_MRP                   8523 non-null   float64
 6   Outlet_Identifier          8523 non-null   object
 7   Outlet_Establishment_Year  8523 non-null   int64
 8   Outlet_Size                6113 non-null   object
 9   Outlet_Location_Type       8523 non-null   object
 10  Outlet_Type                8523 non-null   object
 11  Item_Outlet_Sales          8523 non-null   float64
dtypes: float64(4), int64(1), object(7)
```

Figure 1: Basic Data Information

```
Item_Identifier               0
Item_Weight                1463
Item_Fat_Content              0
Item_Visibility               0
Item_Type                     0
Item_MRP                      0
Outlet_Identifier             0
Outlet_Establishment_Year     0
Outlet_Size                2410
Outlet_Location_Type          0
Outlet_Type                   0
Item_Outlet_Sales             0
dtype: int64
```

Figure 2: Sum of missing values of the data

Data Cleaning and Preprocessing

Referring to Figure 3, the distribution of item weight is within the range of 5.0 to 20.0. Therefore, we imputed the mean weigh value to address the missing values in Item_Weight.

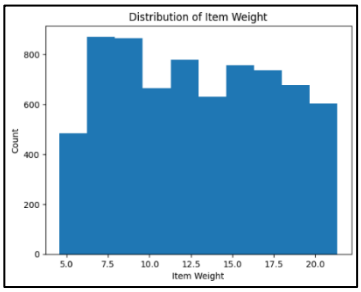

Figure 3: Distribution of Item Weight

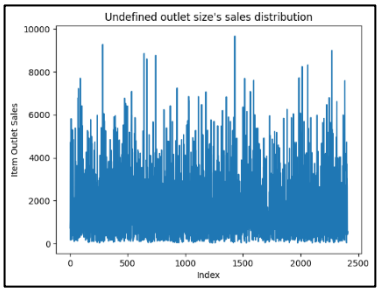

Figure 4: Sales Distribution of Undefined Outlet Size

In Figure 2 and 4, there were 2410 missing values of Outlet_Size, which accounts for approximately 25% of the dataset. Besides, the sales distribution of the missing values of was substantial. A new category which labels as 'Undefined' for Outlet_Size is introduced to address this issue and ensure the integrity of the dataset.

Feature Engineering

Feature engineering is often used to derive important features from existing features to improve the performance of machine learning models. In this case, the column 'Outlet_Establishment_Year' is replaced by 'Years_Since_Establishment' by calculating the difference between 2023 and the year of the establishment year of the store.

EDA

Since the dataset is cleaned and pre-processed, we are ready to conduct Exploratory Data Analysis (EDA). The primary objective of EDA is to gain valuable insights and a deeper understanding of dataset. The EDA is conducted using Python libraries including seaborn (imported as sns) and matplotlib.pyplot (imported as plt). These libraries are essential to create clear and informative visualizations.

In Figure 5, Supermarket Type 3 records the highest sales, whereas Grocery stores have the lowest sales. Moreover, the distribution of the fat content appears to be uniform across different types.

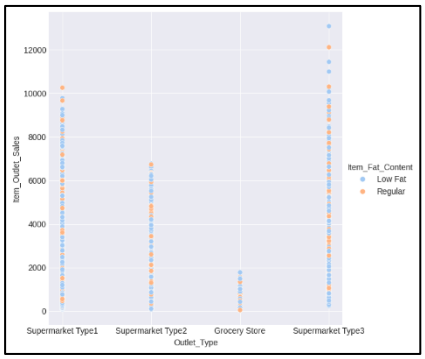

Figure 5: Relationship between Outlet Type and Sales

According to Figure 6, a correlation heat map is conducted to identify multicollinearity among variables which is important to build predictive models. The highest correlation observed is between 'Item_MRP' and 'Item_Outlet_Sales' with a coefficient of 0.57, which indicates a moderate positive relationship. Besides, in Figure 7, we find that sales do not appear to be significantly affected by the price of item. Fruit and vegetables having median price, has achieved the highest sales. These visualizations provide valuable insights into the dataset.

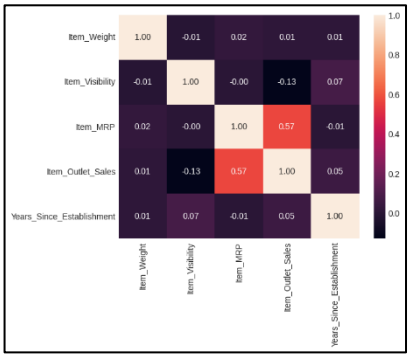

Figure 6: Correlation Heat Map

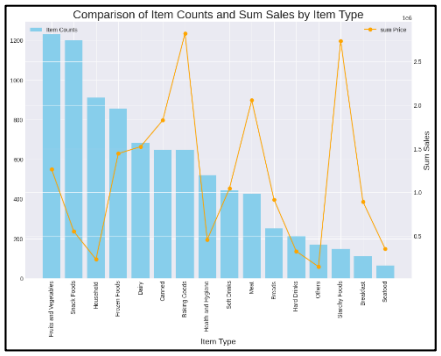

Figure 7: Comparison of Item Counts and Sum Sales by Item Type

Data Selection

The unique identifiers do not contribute to the predictive modelling process. Furthermore, since the information regarding the fat content of items is already represented in 'Item_Type' column. Therefore, the columns 'Item_Identifier', 'Outlet_Identifier' and 'Item_Fat_Content' have been dropped from the dataset.

Unsupervised Learning

Unsupervised learning is a type of machine learning that is absence of target variable and utilizes self-learning algorithms to analyse and cluster unlabelled data sets. (Google Cloud, n.d.) It is often used as part of exploratory data analysis to uncover underlying structures within the data. Unsupervised learning tasks include clustering similar data points, reducing the dimensionality of data, and detecting anomalies or outliers. It discovers hidden patterns and insights without the need for any guidance. However, assessing the results obtained from unsupervised learning can be more challenging compared to supervised learning.

3.1 Methodology

After data cleaning and preprocessing, the features were scaled using Standard Scaler to standardize them onto a unit scale (mean = 0, standard deviation = 1). This step ensures that each feature contributes equally to the analysis, which is important for unsupervised learning models that are sensitive to the variance of initial variables.

Subsequently, Principal Component Analysis (PCA) was conducted to reduce the dimensionality of the dataset while retaining its essential information. While our dataset did not have a large number of features and we were unsure about the presence of multicollinearity, we decided to use PCA as an exploratory data analysis technique to better understand the structure and variance within the data.

The reduced dimensionality data obtained from PCA was then utilized in K-means clustering. We determined the optimal number of clusters to be 2 using elbow method and silhouette score. K-means clustering effectively groups similar data points together, allowing us to discover patterns and structures within the dataset. This technique is suitable for our analysis as it enables exploration of item outlet sales by grouping similar outlets or products based on their characteristics.

3.2 Analysis

3.2.1 Principal Analysis Component (PCA)

PCA is a dimensionality reduction technique used to transform high-dimensional data into a lower-dimensional space while preserving crucial information. (GfG, 2023) It achieves this by generating principal components, which are normalized linear combinations of the original variables ordered by the amount of variance explained in the data.

Using PCA can help simplify datasets and capture the most important patterns and make subsequent analysis more efficient. Overall, it is a powerful statistical method useful for exploratory data analysis, data visualization, and machine learning tasks.

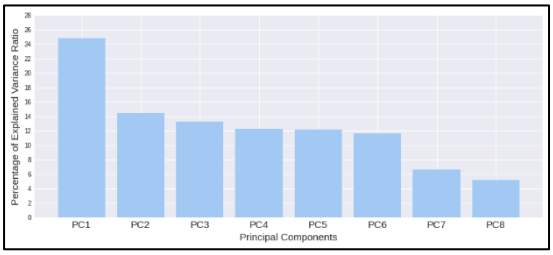

Figure 8: Explained Variance Ratio by Principal Components

	PC_1	PC_2	PC_3	PC_4	PC_5
Item_Weight	0.015401	-0.043411	-0.555585	-0.666481	-0.486329
Item_Visibility	-0.132905	0.607364	0.011779	0.155937	-0.149945
Item_Type	0.005987	-0.194069	-0.568627	0.000047	0.728686
Item_MRP	-0.000700	-0.066619	-0.556484	0.718323	-0.374684
Outlet_Size	-0.571038	-0.228361	0.074483	0.032456	-0.078752
Outlet_Location_Type	0.467266	-0.245977	0.128431	0.101461	-0.173466
Outlet_Type	0.545134	-0.302493	0.102219	0.039689	-0.078459
Years_Since_Establishment	0.374793	0.618844	-0.160263	-0.050748	0.164869

Table 1: Principal Component Loadings for Features

From Figure 8 and Figure 9, the optimal number of principal components is selected as 5 based on the 76.71% of explained variance ratio. PC1 emerged as the most significant component, explaining 24.74% of the variance. This suggests that PC1 captures a substantial amount of information in the dataset, potentially reflects geographical and market characteristics of the outlets due to the high importance of outlet type and location in this component.

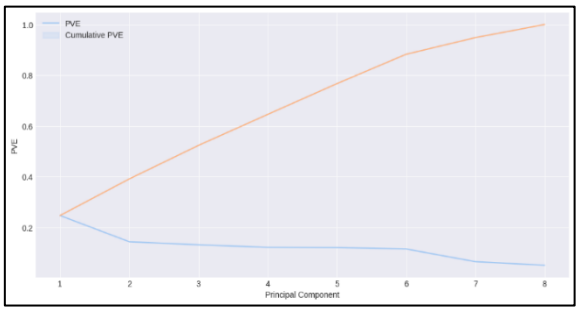

Figure 9: Cumulative Proportion of Variance Explained (PVE) by Principal Component

Table 1 shows the contribution of each variable to the principal components with outlet type and location exhibit relatively high importance in PC1, while years since establishment and visibility of item are relatively significant in PC2.

These 5 principal components, selected for their importance in capturing variance, will serve as input features for our clustering algorithms.

3.2.2 K-Means Clustering

K-means clustering is a popular unsupervised machine learning algorithm that aims to group similar data points together and discover hidden patterns within the data. The number of clusters (K) must be predetermined before running the algorithm which represents the number of centroids required in the dataset. Choosing an optimal K is crucial for effective clustering.

	K	WCSS	silhouette_score
0	2	37506.063324	0.295984
1	3	32733.790002	0.212317
2	4	28693.394993	0.229966
3	5	25847.635972	0.230642
4	6	23636.589820	0.219640
5	7	21880.966251	0.219068
6	8	20549.984559	0.202399
7	9	19376.085693	0.197584
8	10	18355.654359	0.197745
9	11	17470.103416	0.199591
10	12	16684.541175	0.206720
11	13	16072.822281	0.203485
12	14	15526.273582	0.200255
13	15	14974.620712	0.202592

Table 2: K-Means Clustering Performance Metrics

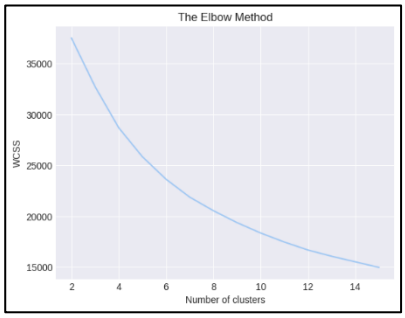

Figure 10: The Elbow Method

The K values was evaluated using the within-cluster sum of squares (WCSS) and silhouette score. WCSS measures the dispersion of data points around their assigned centroids. The silhouette score which ranges from -1 to 1, reflects how well data points are assigned to their clusters, with higher scores suggesting better separation between clusters.

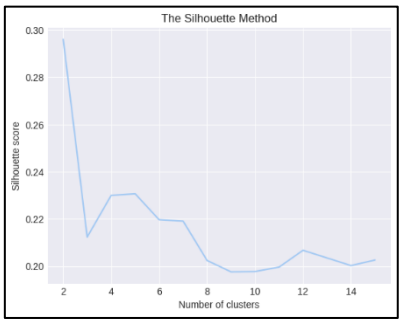

Figure 11: The Silhouette Method

Based on the analysis of Figure 10, Figure 11, and Table 2, we determined 2 as the optimal number of clusters, with a WCSS value of 37506.06332 and a silhouette score of 0.2960. While these metrics suggest some degree of clustering, there is room for improvement in the clustering performance. This could potentially be due to the inherent complexity of the data or limitations of K-means in capturing nuanced relationships between features.

```
Mean Sales by Cluster:
Cluster
0    1945.306032
1    2664.906068
Name: Item_Outlet_Sales, dtype: float64

Median Sales by Cluster:
Cluster
0    1579.6105
1    2251.0698
Name: Item_Outlet_Sales, dtype: float64
```

Figure 12: Mean Sales and Median Sales by Cluster

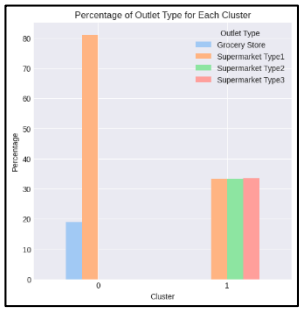

Figure 13: Percentage of Outlet Type for Each Cluster

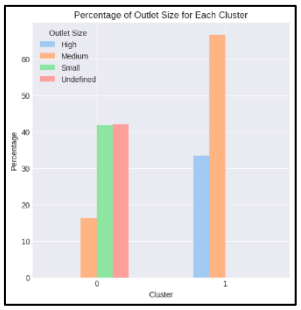

Figure 14: Percentage of Outlet Size for Each Cluster

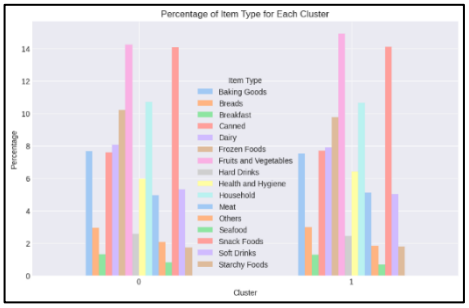

Figure 15: Percentage of Item Type for Each Cluster

Findings from Clustering Analysis:

According to Figure 12, Cluster 1 has a notably higher mean sales of 2664.9061 compared to Cluster 0 with a mean sales value of 1945.3060.

From Figure 13, most Grocery Store outlets are assigned in Cluster 0, while Supermarket Type 1 outlets are distributed across both Cluster 0 and Cluster1. Supermarket Type 2 and Supermarket Type 3 are both assigned to Cluster 1, indicating that these two types of outlets have higher mean sales.

Referring to Figure 14, outlet categorized as “High” size are assigned to Cluster 1, while outlet categorized as “Medium” size are distributed across Cluster 0 and Cluster 1. On the other hand, outlet categorized as “Small” and “Undefined” are clustered in Cluster 0 with lower mean sales.

From Figure 15, the distribution of item type is relatively balanced across both clusters.

3.3 Conclusion

Based on the findings from the PCA and K-Means Clustering analyses, the type of might be important factors influencing sales performance, whereas the impact of item type is comparatively lower. The lower impact of item type may be leveraged for targeted marketing strategies to enhance overall sales performance. These insights can inform strategic decision-making processes.

Supervised Learning

Supervised Learning is a type of machine learning that utilizes a labelled training data set to train algorithms to predict outcomes and recognize patterns. The training data set includes input and outputs for model to learn. There are two types of supervised learning – Regression and Classification, which will be further discuss later.

4.1 Regression

Regression is a supervised learning algorithm which is used to predict continuous target variable. It involves training a model on a dataset to reveal patterns and make accurate predictions for new data points. There are three common metrics to evaluate the regression model, including mean squared error (MSE), mean absolute error (MAE) and R-Squared (coefficient of determination).

4.1.2 Methodology

Three regression models will be built using Python libraries: Linear Regression, Gradient Boosting Regression and Random Forest Regression. These models were chosen for their effectiveness in regression tasks and widespread use in the machine learning community.

After the datasets have been cleaned and pre-processed, the input features (X) and target variable (Y) will be split into training set and testing using the ratio 70:30 with training size covers 70% and testing size covers 30% of the data. The ‘random_state’ parameter will be set to 42 to maintain consistency across model runs.

Each regression model will be trained using the training data. The same train-test split data will be applied to each model in this project to ensure consistent evaluation. The models will be imported using the 'scikit-learn' library, a popular machine learning library in Python. A simple hyperparameter tuning will be conducted to optimize model performance.

Finally, the performance of each model will be evaluated using the root mean squared error (RMSE) and the results will be compared and discussed.

4.1.3 Linear Regression

Linear regression is an algorithm that models the relationship between a dependent variable Y and one or more independent variables X. (Researcher et al., 2023) The basic form of the linear regression equation is:

$$Y \approx \beta_0 + \beta_1 X + \epsilon$$

In this equation, β_0 represents the intercept (or bias term), β_1 represents the slope of the model and ϵ is the error term, which captures the difference between the observed and predicted values of Y.

The goal of the model is to determine the line best fits the data by minimizing the sum of squared differences between the observed and predicted values of the target variable.

Linear regression is a relatively simple algorithm compared to other complex models such as gradient boosting. Therefore, default hyperparameters are used and the tuning is not performed in this section.

```
RMSE of train set in Linear Regression: 1220.7634
RMSE of test set in Linear Regression: 1175.1642
```

Figure 16: RMSE of Linear Regression

Figure 16 shows the root mean square error (RMSE) of linear regression. RMSE of train set is 1220.7634 and RMSE on test set is 1175.1642. This value represents the average difference between the actual and predicted item outlet sales value for the train and test datasets.

4.1.4 Gradient Boosting Regression (GBR)

Gradient boosting is a powerful boosting algorithm that used ensemble method to build predictive model by training weak learners which are decision trees in a sequential manner. (Shoichiro, 2021) The decision trees are tree structure, starting from a single root node and branching based on conditions and head towards the leaves, the goal leaf is the prediction result. GBR builds these trees one at a time, with each new tree focusing on correcting the mistakes of the previous ones. This iterative process improves the model's overall accuracy.

```
RMSE of train set in GBR: 1037.919
RMSE of test set in GBR: 1060.1905
```

Figure 17: RMSE of Gradient Boosting Regression

Gradient Boosting Regression (GBR) hyperparameters were also tuned to optimize the model. The learning rate was adjusted to balance the contribution of each tree to the ensemble with a lower rate which leads to finer adjustments during training. Max depth was set to limit the complexity of individual trees, preventing overfitting. Finally, the number of estimators was chosen to determine the total number of trees trained sequentially.

Referring to Figure 17, the result of root mean square error (RMSE) on train set and test set in GBR is 1037.919 and 1060.1905 respectively. This value represents the average difference between the actual and predicted item outlet sales value for the train and test datasets.

4.1.5 Random Forest Regression

Random forest regression is a bootstrapping algorithm that used ensemble learning method to build a large number of decision trees during training and outputs the mean prediction of these individual trees. Each tree is trained a random sample from the dataset when generating the splits, adding an element of randomness to prevent overfitting. (Whitfield, 2023)

A random forest regressor is first implemented with default parameter settings. After that, RandomizedSearchCV from Scikit-Learn is employed to perform a random search over the given parameters for the optimal hyperparameter combination. The cross-validation (cv) parameter is set to 5, indicating that the dataset is divided into 5 equal parts. The model is trained on 4 of these folds and validated on the remaining 1-fold. This process is repeated 5 times with each fold served as the validation set once. Moreover, the verbose level is set to 2 which means that the detailed output of the search progress will be printed for transparency and easier to debug.

```
RMSE of train set in Random Forest Regression: 430.5505
RMSE of test set in Random Forest Regression: 1116.2225
```

Figure 18: RMSE of Random Forest Regression before tuning hyperparameters

```
RMSE of train set in Random Forest Regression: 917.6079
RMSE of test set in Random Forest Regression: 1067.6192
```

Figure 19: RMSE of Random Forest Regression after tuning hyperparameters

Hyperparameter optimization significantly improved model performance. The initial low RMSE on the train set (430.5505) suggests potential overfitting. Overfitting often occurs when the model learns noise in the training data, resulting in an overly complex model. After tuning, the RMSE of the random forest regression model is 917.6079 on the train set and 1067.6192 on the test set, which indicates a more balance performance.

4.1.6 Evaluation and Discussion of the models' results

Choosing the right model and appropriate evaluation metrics is crucial for accurately assessing performance in predicting the continuous target variable, "Item_Outlet_Sales." Root mean squared error (RMSE) was chosen as the evaluation metric due to its effectiveness in measuring the average difference between actual and predicted values. It is useful when large errors are undesirable and its robustness to outliers because it squares the errors.

The performance metrics of each model are summarized as follows:

Evaluation Metrics	Linear Regression	Gradient Boosting Regression (GBR)	Random Forest Regression
RMSE on train set	1220.7634	1037.9190	917.6079
RMSE on test set	1175.1642	1060.1905	1067.6192

From Figure 20 shows a wide distribution of sales value. The standard deviation of 1706.50 suggests the sales are spread out from the mean which indicates inconsistency in sales patterns. Therefore, the RMSE around 1000 is acceptable.

```
count     8523.000000
mean      2181.288914
std       1706.499616
min         33.290000
25%        834.247400
50%       1794.331000
75%       3101.296400
max      13086.964800
```

Figure 20: Descriptive Statistics of 'Item Outlet Sales'

Regarding the model performance, Linear Regression exibits the weakest performance. GBR is the top performer on the test set while Random Forest Regression demonstrates the best performance on train set.

In conclusion, GBR is considered as the preferred model due to its balanced performance on both train and test sets.

4.2 Classification

Classification is a machine learning technique used to categorize data into different categories. The model is trained using the given dataset and then classifies new observation into the correct classes. Unlike regression which predicts continuous numeric values, classification predicts categories based on the features provided in the dataset.

4.2.1 Methodology

After data cleaning and preprocessing, a new column named 'Sales_Category' was created with four classess (Q1, Q2, Q3 and Q4) based on the quartiles of the 'Item_Outlet_Sales' features. Then, 'Item_Outlet_Sales' was replaced by 'Sales_Category' as the target classes for classification.

Four classification model were built using Python libraries: Logistic Regression, Support Vector Machine Classification (SVM), Random Forest Classification and XGBoost Classification. These models were chosen for their proven effectiveness in classification tasks. The scikit-learn library and xgboost library are imported to construct these models.

The input features (X) and target variable (Y) will be split into training and testing sets using the ratio 70:30 with training set covers 70% and testing set covers 30% of the data. The 'random_state' parameter will be set to 42 to maintain consistency across model runs. The same train and test sets will be utilized for all models to ensure consistency. A basic hyperparameter tuning process will be implemented to optimize model performance by adjusting the key hyperparameters for every model.

Finally, the performance of each model was evaluated using standard classification metrics: accuracy, percision, recall and F1-score. The results will be compared and discussed to determine the most suitable classification model.

4.2.2 Logistic Regression

Logistic regression is a statistical method used for binary classification that predicts the probability of a data point that belongs to one of two possible categories. The One-vs-Rest (OVR) strategy extends logistic regression to multiclass classification. In OvR, a separate binary classifier is trained for each class and treats its corresponding class as positive and all other classes as negative. Each classifier outputs a probability of the data point belonging to its positive class. Finally, the class with the highest predicted probability is assigned as the final prediction. (Pramoditha, 2023)

The hyperparameters of logistic regression were tuned to optimize the model. The regularization parameter (C) was adjusted to balance model complexity and overfitting, and the 'liblinear' solver was chosen for its efficiency with our dataset.

```
Accuracy  = 0.535
Precision = 0.5133
Recall    = 0.5342
f1_score  = 0.5089
```

Figure 21: Evaluation Metrics of Logistic Regression

The model correctly predicts the class for about 54% of the data points (accuracy). When it predicts a certain class, it is correct about 51% of the time (precision). It also correctly identifies about 53% of the data points that actually belong to that class (recall). The F1-score combines precision and recall,which is 51%.

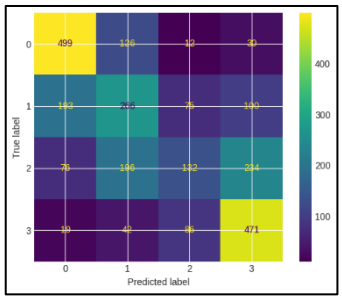

Figure 22: Confusion Matrix of Logistic Regression model

From Figure 22, the confusion matrix shows that the model performs best on Class 0 and Class 3, with the most correct predictions. It has more difficulty distinguishing between Class 1 and Class 2.

4.2.3 Support Vector Machine Classification (SVM)

SVM is a versatile type of algorithm used for both classification and regression task. In classification, SVM aims to find the optimal hyperplane (decision boundary) that separates data points belongs to different classes. SVM map data points to a higher dimensions to handle non-linear relationships between features and classes. The SVM algorithm searches for the hyperplane that maximizes the margin between the closest data points from different classes. These closest points are called support vectors and play a crucial role in defining the decision boundary. Finally, new data points are classified by determining which side of the hyperplane they fall on. (Navlani, 2019)

The feature is scaled before implementing this model as SVM model is sensitive to the scales of features. Hyperparameters of SVM were tuned to optimize the model. The hyperparameters of the SVM model were tuned to optimize its performance. The regularization parameter (C) was adjusted to balance model complexity and overfitting, while the 'rbf' kernel was chosen for its flexibility in capturing complex patterns in the data.

```
Accuracy  = 0.5858
Precision = 0.5845
Recall    = 0.5847
f1_score  = 0.5835
```

Figure 23: Evaluation Metrics of SVM

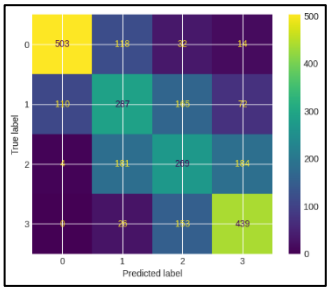

Figure 24: Confusion Matrix of SVM

Referring to Figure 23, The SVM model correctly predicts the class for about 58.58% of the data points (accuracy). When it predicts a certain class, it is correct about 58.45% of the time (precision). It also correctly identifies about 58.47% of the data points that actually belong to that class (recall). Overall, the model demonstrates balanced performance with an F1-score of 58.35%.

From Figure 24, the model performs best on Class 0 and Class 3, with the most correct predictions. It has more difficulty distinguishing between Class 1 and Class 2.

4.2.4 Random Forest Classification

Random Forest Classification is an ensemble learning method that constructs many decision trees during the training. Each decision tree in the forest is trained on a random subset of the training data. This randomness helps to reduce overfitting and improve robustness. Finally, the prediction is made by aggregating the predictions from all the trees.

```
Accuracy  = 0.5706
Precision = 0.5715
Recall    = 0.5695
f1_score  = 0.5692
```

Figure 25: Evaluation Metrics of Random Forest Classification

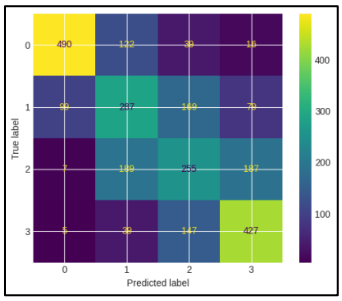

Figure 26: Confusion Matrix Random Forest Classification

The hyperparameters were tuned to optimize the model. The number of decision trees was tuned, which provides a good balance between accuracy and computational efficiency.

The Random Forest Classification model correctly predicts the class for about 57.06% of the data points (accuracy). It is correct about 57.15% when predicting a certain class (precision). Besides, it also correctly identifies about 56.95% of the data points that actually belong to the class (recall). The F1-score of 56.92% combines precision and recall.

From Figure 26, the model performs best on Class 0 and Class 3, with the most correct predictions. It has more difficulty distinguishing between Class 1 and Class 2.

4.2.5 XGBoost Classification

XGBoost which means eXtreme Gradient Boosting, is an advanced implementation of the gradient boosting algorithm. It is a type of ensemble learning algorithm that combine multiple weak learners, typically decision trees, to create a strong predictive model. Each tree is bult sequentially by correcting the errors made by the previous ones. Besides, XGBoost employs a technique called tree pruning to control the complexity of individual decision trees. Finally, XGBoost combines the predictions from all the trees. (NVIDIA, n.d.)

```
Accuracy  = 0.5933
Precision = 0.5919
Recall    = 0.5922
f1_score  = 0.5887
```

Figure 27: Evaluation Metrics of XGBoost Classification after tuning

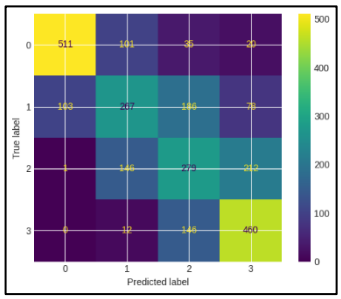

Figure 28: Confusion Matrix of XGBoost

The XGBoost classification model is applied. Its performance is evaluated before and after hyperparameter tuning using grid search. The initial model only achieved and accuracy of 55.49%, indicating some misclassification between certain classes. After tuning key hyperparameters, such as learning rate, tree depth and feature sampling, the model's performance improved significantly. The accuracy of the model after tuning is 59.33%. The precision score achieved 59.19%. The recall is 59.22% and f1-score which is the combination of precision and recall is 58.87%.

From Figure 28, the model performs best on Class 0 and Class 3, with the most correct predictions. It has more difficulty distinguishing between Class 1 and Class 2.

4.2.6 Evaluation and Discussion of the models' results

To evaluate the results, four important metrics were chosen: accuracy, precision, recall, and F1-score. Accuracy measures the overall percentage of correct predictions made by the model, but it can be misleading with imbalanced datasets. Precision measures the proportion of predicted positive cases that were actually positive, while recall measures the proportion of actual positive cases that were correctly predicted as positive. The F1-score combines precision and recall into a single score. It is often used to deal with imbalanced datasets. (Walker II, n.d.)

The performance of the classification models is summarised below:

Evaluation Metrics	Logistic Regression	Support Vector Machine (SVM)	Random Forest Classification	XGBoost Classification
Accuracy	0.535	0.5858	0.5706	0.5933
Precision	0.5123	0.5825	0.5715	0.5919
Recall	0.5342	0.5846	0.5695	0.5922
F1-score	0.5089	0.5822	0.5692	0.5887

Overall, all models achieved accuracy scores between 53.5% and 59.33%, indicating moderate performance. XGBoost emerged as the best performing model with the highest accuracy (59.33%), followed closely by SVM (58.58%), then Random Forest (57.06%), and lastly Logistic Regression (53.5%).

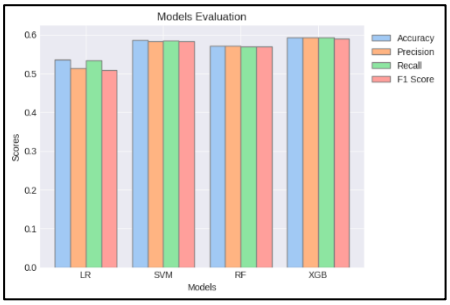

Figure 29: Classification Model Evaluation

Logistic Regression is a linear model which sensitive to outliers, has the lowest performance. This suggests that the relationship in the data might be non-linear which Logistic Regression might not be able to capture the. In contrast, SVM, Random Forest and XGBoost can handle non-linear relationships and have a better performance in this case. Besides, the ensemble nature of XGBoost and Random Forest allows them to handle outliers more effectively, leading to a better performance.

	Feature	Importance
6	Outlet_Type	0.645197
3	Item_MRP	0.117005
7	Years_Since_Establishment	0.041704
0	Item_Weight	0.040615
1	Item_Visibility	0.040491
4	Outlet_Size	0.038714
5	Outlet_Location_Type	0.038213
2	Item_Type	0.038061

Figure 30: Feature Importance of XGBoost Classification

By observing the confusion matrices of each model (Figure 22, 24, 26, 28), it was found that all models performed well on Class 0 and Class 3 but struggled to distinguish between Class 1 and Class 2. This suggests that these classes might share similar features or characteristics, making it difficult for the models to differentiate between them. This could be due to overlapping features, which can lower the overall accuracy.

From Figure 30, we can observe the high importance of 'Outlet_Type' in XGBoost Classification, indicating that this feature is important for predicting the target variable.

Overall, XGBoost is the preferred model for classification in this dataset. It achieved the highest accuracy, precision, recall and F1-score among the tested models. While other models like SVM and Random Forest also performed well.

Conclusion and recommendations

In conclusion, this analysis revealed that outlet type is the primary factor influencing sales while the type, weight and visibility of item is less important in this case. Among the tested models, Gradient Boosting Regressor (GBR) demonstrated the best performance among the regressors and XGBoost Classifier excelled among classifiers. This highlights that the boosting models perform well for this dataset. GBR can be used for predicting the Item Outlet sales, but the result may only be moderately accurate.

We recommend Big Mart to prioritize strategies related to outlet type, such as tailoring product assortment and promotions to specific store demographics. GBR and XGBoost can be used as forecasting tools and regularly updating them with new data can aid in inventory management and resource allocation.

This study was limited by the available data, which only focused on outlets and items. Further research could explore the impact of external factors such as the economic conditions and competitor activity on sales performance.

By implementing these recommendations and conducting further research, Big Mart can gain a deeper understanding of the factors driving sales and make data-driven decisions to optimize performance and profitability.

References

[1] Google Cloud, “What is unsupervised learning? | google cloud,” Google, https://cloud.google.com/discover/what-is-unsupervised-learning#:~:text=Unsupervised%20learning%20in%20artificial%20intelligence,any%20explicit%20guidance%20or%20instruction. (accessed Apr. 1, 2024).

[2] aishwarya. 27 GfG, “Principal component analysis(pca),” GeeksforGeeks, https://www.geeksforgeeks.org/principal-component-analysis-pca/ (accessed Apr. 1, 2024).

[3] V. K. A. Researcher et al., “What is linear regression?- Spiceworks,” Spiceworks, https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-linear-regression/#:~:text=Linear%20regression%20is%20an%20algorithm,machine%20learning%20for%20predictive%20analysis. (accessed Apr. 4, 2024).

[4] Y. Shoichiro, “Regression analysis using gradient boosting regression tree: Aurora Articles,” Regression analysis using gradient boosting regression tree, https://www.nec.com/en/global/solutions/hpc/articles/tech14.html#:~:text=Gradient%20boosting%20regression%20trees%20are,leaf%20is%20the%20prediction%20result. (accessed Apr. 1, 2024).

[5] B. Whitfield, “Random Forest regression in Python explained,” Built In, https://builtin.com/data-science/random-forest-python (accessed Apr. 1, 2024).

[6] R. Pramoditha, “Logistic regression for multiclass classification - 3 strategies you need to know,” Medium, https://rukshanpramoditha.medium.com/logistic-regression-for-multiclass-classification-3-strategies-you-need-to-know-0a3e74574b96#:~:text=One%2Dvs%2DRest%20(OvR)%20multiclass%20strategy,-When%20there%20are&text=This%20method%20creates%20one%20logistic,models%20in%20the%20following%20way. (accessed Apr. 1, 2024).

[7] A. Navlani, “Scikit-learn SVM tutorial with Python (Support Vector Machines),” DataCamp, https://www.datacamp.com/tutorial/svm-classification-scikit-learn-python (accessed Apr. 2, 2024).

[8] “What is XGBoost?,” NVIDIA Data Science Glossary, https://www.nvidia.com/en-us/glossary/xgboost/ (accessed Apr. 3, 2024).

[9] S. M. Walker II, “F-score: What are accuracy, precision, recall, and F1 score?,” Klu, https://klu.ai/glossary/accuracy-precision-recall-f1 (accessed Apr. 3, 2024).