Big Mart Sales Prediction Project Documentation

Problem Statement

The objective of this project is to develop a machine learning model capable of accurately predicting the sales of products in a retail environment, specifically for Big Mart. By analyzing historical sales data, the model aims to forecast future sales, helping improve inventory management, targeted marketing, and resource allocation. The data provided by Big Mart spans across 1559 products from 10 stores in different cities, collected over the year 2013. The dataset includes various product and store attributes that influence sales performance.

Data Description

The dataset consists of several features, categorized as numerical and categorical, which describe the products and stores:

Numerical Features

1. **Item_Weight**: The weight of the product in kilograms.
2. **Item_Visibility**: The percentage of the total display area in a store allocated to the product.
3. **Item_MRP**: Maximum Retail Price (list price) of the product.
4. **Outlet_Establishment_Year**: The year in which the store was established.
5. **Item_Outlet_Sales**: Sales of the product in a particular store. This is the target variable we aim to predict.

Categorical Features

1. **Item_Identifier**: Unique product ID (to be dropped later).
2. **Item_Fat_Content**: The fat content of the product (e.g., low fat, regular).
3. **Item_Type**: The category to which the product belongs.
4. **Outlet_Identifier**: Unique store ID.
5. **Outlet_Size**: The size of the store in terms of ground area covered.
6. **Outlet_Location_Type**: The type of city in which the store is located.
7. **Outlet_Type**: Whether the outlet is a grocery store or a supermarket.

Missing Data Handling

The dataset contains missing values in some features:

- **Outlet_Size**: 2410 missing values. This categorical feature is imputed using the mean of the available data.

- **Item_Weight**: 1463 missing values. This numerical feature is imputed with the mode.

Exploratory Data Analysis (EDA)

Univariate Analysis

The following observations were made after plotting and analyzing the numerical features:

- **Item_Weight**: The weight of items ranges from 5 kg to 20 kg.
- **Item_Visibility**: This feature is right-skewed, indicating that most products have low visibility compared to a few highly visible ones.
- **Item_MRP**: There is a concentration of products with prices between 100 and 180 MRP.
- **Outlet_Establishment_Year**: Most stores were established in specific years such as 1985 and 1998, with no new stores between 1990 and 1995.
- **Item_Outlet_Sales**: The distribution of sales is right-skewed, suggesting a few products are highly successful while others perform poorly. A transformation may be applied to achieve a more Gaussian distribution.

A box plot was used to identify outliers in the dataset:

- **Item_Weight**, **Item_MRP**, and **Outlet_Establishment_Year** have no significant outliers.
- **Item_Visibility** and **Item_Outlet_Sales** show outliers, which may need to be removed.

Additionally, the distribution of data in the training and test sets is nearly identical.

Categorical Features Analysis

Bar charts were used to visualize the distribution of categorical features:

- **Item_Fat_Content**: This feature contains inconsistent entries ('Low Fat', 'low fat', 'LF'). These should be consolidated into a single category (e.g., 'LF').
- **Item_Type**: There are 16 categories, which may be too many. Feature engineering can group similar items together, such as combining 'Soft Drinks' and 'Hard Drinks' into a 'Drinks' category or 'Snack Foods', 'Frozen Foods', and 'Seafood' into a 'Foods' category.
- **Outlet_Location_Type**: This feature can be encoded numerically, with 'Tier 1' as 1, 'Tier 2' as 2, and 'Tier 3' as 3.

Bivariate Analysis

The correlation between numerical variables and the target variable, **Item_Outlet_Sales**, was examined. A heatmap was generated to visualize the relationships, revealing that **Item_Visibility** and **Outlet_Establishment_Year** are negatively correlated with sales.

Feature Engineering

To improve the model's performance, several steps were taken:

- **Outlier Removal**: Outliers in the **Item_Visibility** and **Item_Outlet_Sales** features were removed using the Interquartile Range (IQR) method.
- **Feature Cleaning**: Inconsistent values in the **Item_Fat_Content** feature were standardized (e.g., 'Low Fat', 'low fat', and 'LF' were consolidated into 'LF').
- **Feature Transformation**: The **Outlet_Size** feature was encoded numerically with 'Small' as 1, 'Medium' as 2, and 'Large' as 3.
- **Dropped Irrelevant Features**: Features like **Item_Identifier** and **Outlet_Identifier** were dropped, as they do not provide meaningful information for prediction.

Model Development and Evaluation

The model will be built using appropriate machine learning algorithms, such as linear regression or decision trees, depending on the data distribution and feature importance. The model will be evaluated based on metrics like Root Mean Squared Error (RMSE), and R-squared (R^2).

Linear Regression

We began by training a **Linear Regression** model to predict **Item_Outlet_Sales**. The following results were obtained:

- **Training Score**: 0.4871
- **Test Score**: 0.4825
- **RMSE (Root Mean Squared Error)**: 1092.02
- **R^2 Score**: 0.4825

The linear regression model provides a relatively low performance on both training and testing datasets, with an R^2 score of 0.4825. This indicates that the model can explain only about 48% of the variance in the sales data.

Random Forest

We then proceeded with a **Random Forest** model to capture non-linear relationships and interactions between features. The results were:

- **Training Score**: 0.9343
- **Test Score**: 0.5143
- **RMSE**: 1057.99
- **R^2 Score**: 0.5143

The random forest model performed significantly better on the training set, with a training score of 0.9343, but it still exhibited some overfitting as the test score was notably lower at 0.5143. The RMSE of 1057.99 reflects the error in predicting the sales.

Random Forest with Hyperparameters Tuning

Next, we tuned the hyperparameters of the Random Forest model to improve its performance. We used the following parameters:

- **n_estimators**: 200
- **max_depth**: 10
- **min_samples_split**: 10
- **min_samples_leaf**: 5
- **max_features**: 'sqrt'

The performance after tuning the model was as follows:

- **Training Score**: 0.6587
- **Test Score**: 0.5719
- **RMSE**: 993.20
- **R^2 Score**: 0.5719

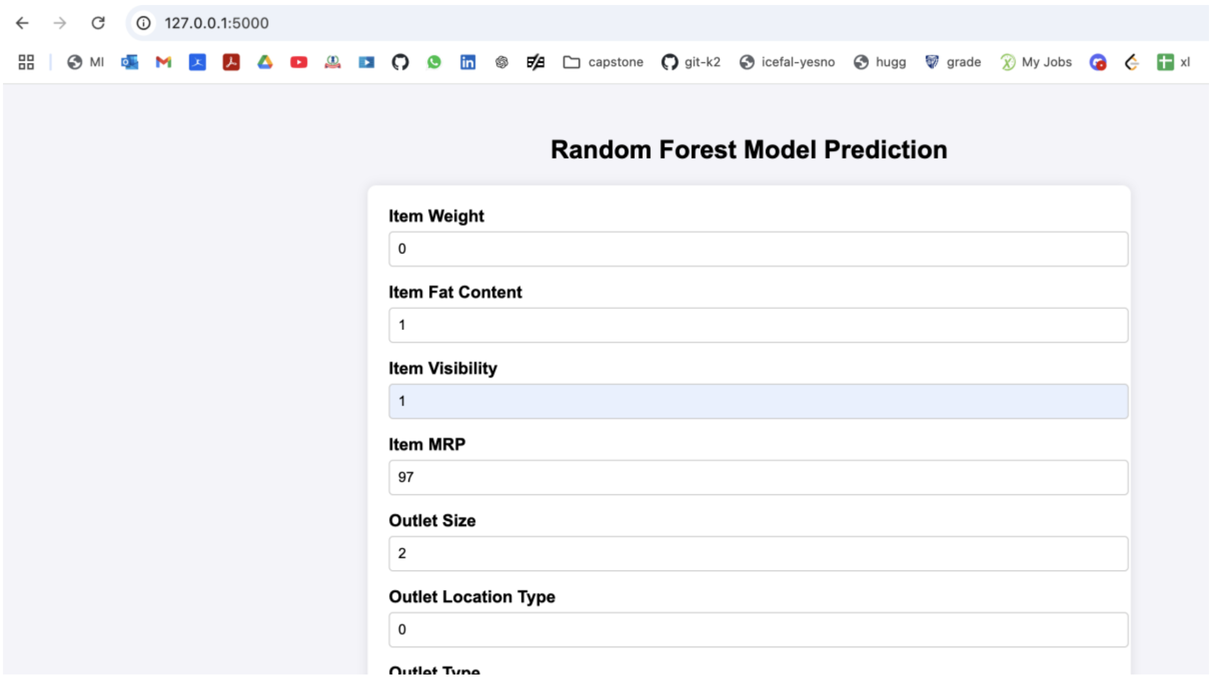
Additionally, we performed cross-validation, which resulted in the following scores:

- **Cross-validation Scores**: [0.5674, 0.5597, 0.5985, 0.5536, 0.5869]
- **Average Cross-validation Score**: 0.5732

The tuned Random Forest model showed a more balanced performance with a test score of 0.5719 and a relatively lower RMSE of 993.20. The average cross-validation score of 0.5732 indicates a more stable model performance across different subsets of the data.

Web application

I developed a **Web Application** to provide an interactive user interface for stakeholders to access the sales predictions and visualize the results. The web application was built using **Flask** (a Python web framework) and **HTML**, ensuring a simple yet effective platform for users to interact with the model.



Random Forest Model Prediction

Item Weight: 0

Item Fat Content: 1

Item Visibility: 1

Item MRP: 97

Outlet Size: 2

Outlet Location Type: 0

Outlet Type

Conclusion

In conclusion, the **Random Forest** model with tuned hyperparameters outperforms the **Linear Regression** model in terms of both training and test scores. It also demonstrates better generalization when evaluated with cross-validation. The final model with hyperparameters tuning offers a good trade-off between training performance and generalization to unseen data.