SUPER MARKET SALES PREDICTION

GROUP 9 (Addanki)

Ravipati Swarna, 220101086

Kasani Keerthi Sarika, 220101049

Talari Syama Sameeksha ,220101097

Problem Statement:

In this project, we explore a dataset of 8,523 supermarket sales records and implemented machine learning models that **predict sales** based on various features related to products, customers, and store operations.

ML Models Used and the reason to choose them:

1. Linear regression - As a baseline model

2. Random Forest Regressor - For capturing non-linear relationships

3. Gradient Boosting - For incremental improvement of predictions

Metrics Used for evaluation:

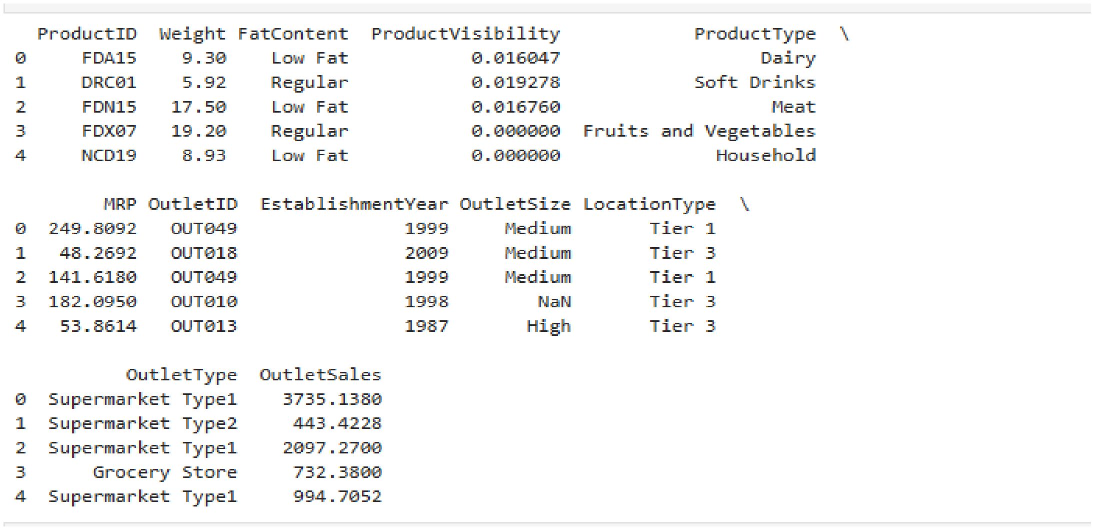
We used R2 and RMSE primarily to compare the ML Models

- **RMSE**: Measures the average magnitude of prediction errors (lower the better).
- **R^2: Indicates** the proportion of variance in the dependent variable (higher the better performance).

How we Proceeded further?

1. Took a dataset for "Big market" Sales from Kaggle - Dataset Link

DATASET FEATURES


```
  ProductID  Weight FatContent  ProductVisibility            ProductType  \
0     FDA15    9.30    Low Fat           0.016047                  Dairy
1     DRC01    5.92    Regular           0.019278            Soft Drinks
2     FDN15   17.50    Low Fat           0.016760                   Meat
3     FDX07   19.20    Regular           0.000000  Fruits and Vegetables
4     NCD19    8.93    Low Fat           0.000000              Household

        MRP OutletID  EstablishmentYear OutletSize LocationType  \
0  249.8092   OUT049               1999     Medium       Tier 1
1   48.2692   OUT018               2009     Medium       Tier 3
2  141.6180   OUT049               1999     Medium       Tier 1
3  182.0950   OUT010               1998        NaN       Tier 3
4   53.8614   OUT013               1987       High       Tier 3

         OutletType  OutletSales
0  Supermarket Type1    3735.1380
1  Supermarket Type2     443.4228
2  Supermarket Type1    2097.2700
3      Grocery Store     732.3800
4  Supermarket Type1     994.7052
```

2. Understanding the features in the dataset

3. Filled the **missing values** with the appropriate values

4.Feature engineering:

- Dropped the columns which were irrelevant to predict the Sales (ProductId, OutletID)
- Checked for Unique values in categorical data, and made changes in them accordingly
- Replaced Establishment with OutletAge (= CurrentYear – Establishment Year)
- Transformed the ProductVisibility column by applying a logarithmic function (nplog1p)to it.
- Added a MRP_weight feature , to consider the combined relation between them

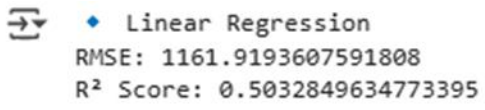
5.Split the data into train and test sets

6.Trained Models

7. Compared with LLM

Performance between models:

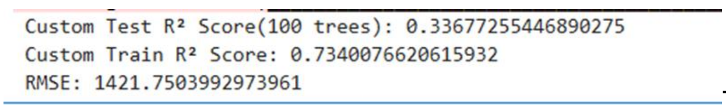
1. Linear Regression:



```
◆ Linear Regression
RMSE: 1161.9193607591808
R² Score: 0.5032849634773395
```

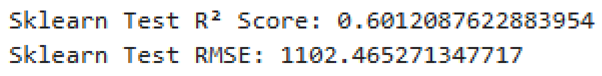
Input features were first converted to `float64` and normalized to ensure stable training. A bias term was manually added to the feature matrix. The model was trained using 1000 iterations of gradient descent with a learning rate of 0.001. The model was trained using **Batch Gradient Descent**, where the entire training dataset is used in each iteration to update the weights (theta).

2. Random forest Regressor:



```
Custom Test R² Score(100 trees): 0.33677255446890275
Custom Train R² Score: 0.7340076620615932
RMSE: 1421.7503992973961
```

The n_estimators in the random forest were set to 100, more than that only caused to even further degraded performance. It performed worse than the linear regression, which could be the cause that the model is trying to predict more complexity in the dataset.

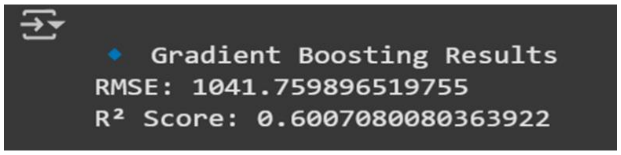


```
Sklearn Test R² Score: 0.6012087622883954
Sklearn Test RMSE: 1102.465271347717
```

These were the values even when the built-in libraries were used, so the problem might lie in the dataset. The difference between 0.33 and 0.60 is mostly because the trees in scikit are being split randomly at every state, while in the custom function (creates more diverse tree's), we implemented, the random selection only happened at the start of the tree.

Random forest might perform better if we had complex data, which could explain the underlying causes, which might have an impact on the Sales.

3. Gradient Boosting:



```
🔹 Gradient Boosting Results
RMSE: 1041.759896519755
R² Score: 0.6007080080363922
```

Gradient Boosting builds trees sequentially, each one correcting the previous errors. This allows it to **capture non-linear relationships** and **interactions between features** better than the other two models. Unlike Random Forests, which treats all samples equally, and pays special attention to outliers by training each new tree on the errors.

It took ~20mins to train with number of estimators (decision trees to be built) as 100 and learning rate as 0.1.

Comparison With LLM:

We compared our best-performing model, Gradient Boosting, with a Large Language Model (LLM) for predicting outlet sales using the same feature set. A representative subset of test data was used to generate structured prompts for the LLM. Both models produced sales predictions, allowing for a direct performance comparison. Gradient Boosting consistently delivered better accuracy and reliability on this structured dataset.

LLM

- **RMSE: 1079.47**
- **R^2 Score**: **0.571**

Gradient Boosting

- **RMSE: 1041.76**
- **R^2 Score**: **0.600**

How we worked as team?

We worked as a team, exploring multiple regression models including Linear Regression, Random Forest, and Gradient Boosting (tried other models too but couldn't finish the implementation of them). Over the span of few sessions (each ~2.5hrs), we learned the working principles of each model from scratch, implemented them, and discussed their performance on our dataset. Although each of us took the lead in understanding different models in depth, we worked together on all parts — from preprocessing and model tuning to evaluation and interpretation. This helped us to develop a good understanding of regression algorithms and improved our ability to apply them effectively to real-world data than doing it individually

Conclusion:

From our model's performance analysis, we conclude that, unlike Linear Regression, which underfits due to its simplicity, and Random Forest, which struggles with sharp variations, Gradient Boosting iteratively corrects its errors by focusing on difficult cases. With an R^2 of 0.60 and RMSE of 1041.7, Gradient Boosting proved better performance among the models.