**Data Preparation and Preprocessing**

**Exploratory Data Analysis**

1. **Reviewed the data**: The first few rows of the dataset were examined to understand the structure and content of the data.

2. **Checked the shape of the dataset**: The dataset contains 4000 rows and 8 columns.

3. **Identified the features**: The dataset includes features such as "Sold\_On", "Size", "Ingredients\_Cost", "Design\_Complexity", "Time\_Taken", "Price", "Amount", and "Gender",indenfiy what’s numerical and categorical columns

4. **Visualization on categorical columns:** to more understand data

**Data Cleaning & Transformation**

1. **Checked for missing values**: The dataset did not contain any missing values.

2. **Handled categorical features**: The categorical features "Sold\_On", "Size", "Design\_Complexity", and "Gender" were converted to numeric values using the LabelEncoder from scikit-learn on Size, Design Complexity and Gender and using get\_dummies on Sold on column to convert categorical data into binary matrix .

3. **Scaled numeric features**: The numeric features "Ingredients\_Cost", "Time\_Taken", "Price", and "Amount" were standardized using the StandardScaler from scikit-learn to ensure they are on a similar scale.

4. **Check Outlier and handle it:** check outlier using Interquartile range method and visualization use boxplot, remove any value more than upper bound or less than lower bound.

**Feature Selection**

**Correlation Analysis:** show correlation between columns by heatmap.

**Feature Importance:**  Use feature importance from tree-based models ( Random Forest) to rank features. We detect that Top feature based on importance:['Ingredients\_Cost', 'Design\_Complexity', 'Time\_Taken'],So we drop data and drop not necessary columns

X-> ['Size', 'Ingredients\_Cost', 'Design\_Complexity', 'Time\_Taken', 'Amount']

y->[‘Price’]

**Model Selection and Evaluation**

For regression tasks, algorithms such as Linear Regression, Decision Trees, Random Forest Regressor, Gradient Boosting Regressor, and Support Vector Regressor are considered.

Split the dataset into training and testing sets (e.g., 80% training and 20% testing) to evaluate the model’s performance on unseen data.

**To evaluate the performance of these models, I used the following metrics:**

-**Mean Absolute Error (MAE):** Measures the average absolute difference between the predicted and actual values.

**-Mean Squared Error (MSE):** Measures the average squared difference between the predicted and actual values.

**- R-squared (R²):** Measures the proportion of the variance in the target variable that is predictable from the independent variables.

**Results and Discussion**

The performance of the three models on the test set is as follows:

A black screen with numbers and letters

Description automatically generated

The Gradient Boosting Regressor achieved the best performance, with an MAE of 1.08, an MSE of 1.87, and an R² of 0.86. This suggests that the model can capture the non-linear relationships between the features and the target variable effectively.

Use Grid SearchCV to determine best hyperparameter and save best model to use it to predict on new data.

In conclusion, the Gradient Boosting Regressor was the best-performing model for predicting the "“Number”" of items sold based on the given features. The model was able to capture the non-linear relationships and feature interactions effectively, resulting in a high R-squared value and low error metrics.

**Challenges**

One of the challenges faced during the analysis was the presence of categorical features, which required encoding before they could be used in the models. The LabelEncoder was used to convert some categorical features to numeric values, which helped the models to understand and learn from these features.

Another challenge was the potential for overfitting, especially with the more complex models like Random Forest and Gradient Boosting. To address this, I performed a grid search to optimize the hyperparameters of these models, which helped to improve their generalization performance.