1. Reinforcement Learning – Maze Problem

1.1 Problem Definition

The implemented code solves a maze navigation problem using Q-learning, a reinforcement learning algorithm that enables an agent to learn an optimal navigation policy. The maze is represented as a grid, with 0 denoting obstacles (non-walkable) and 1 representing walkable paths. The agent starts at a predefined position and aims to reach the goal position while maximizing rewards. This report outlines the implementation, training process, visualizations, and advantages of Q-learning for solving the maze navigation problem.

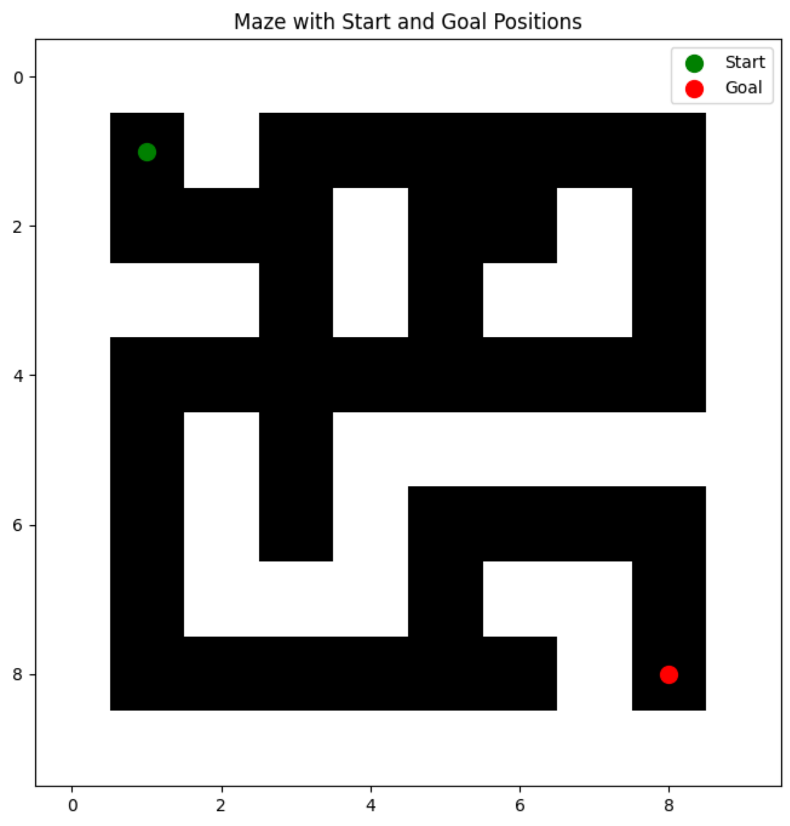
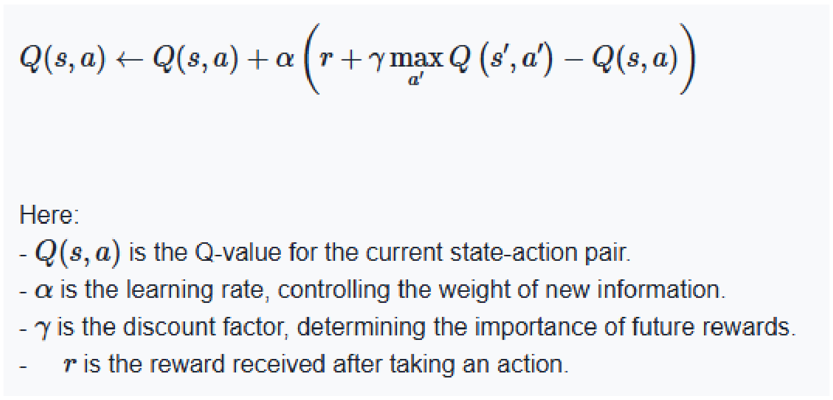


Figure 0.1 Provided Maze

The solution is to traverse through the maze from position (1,1) to position (8,8). For this a Q-learning model has been incorporated.

1.2 Q – Learning Overview

Q-learning is a model-free reinforcement learning algorithm that trains the agent through interactions with the environment. It utilizes a Q-table to store the expected utility of taking each possible action in every state. For this problem, the Q-table is a three-dimensional matrix where the rows and columns correspond to the maze's dimensions, and the third dimension represents the four possible actions: up, down, left, and right.



$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q\left(s', a'\right) - Q(s,a) \right)$$

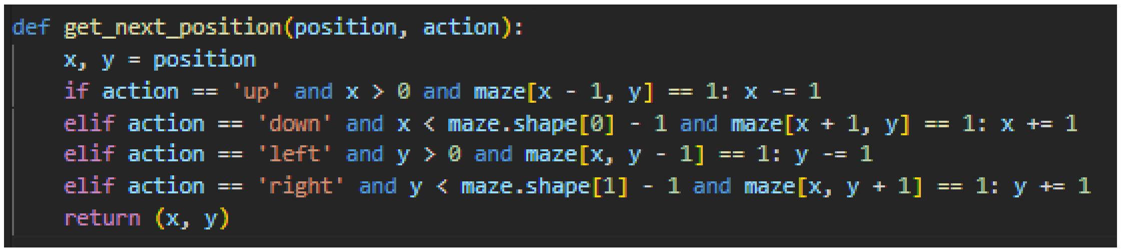
Here:
- $Q(s,a)$ is the Q-value for the current state-action pair.
- α is the learning rate, controlling the weight of new information.
- γ is the discount factor, determining the importance of future rewards.
- r is the reward received after taking an action.

Figure 0.2 Q-Learning equation

1.3 Components

1.3.1 State Transitions

The get_next_position function ensures valid state transitions by checking the maze boundaries and verifying if the target cell is walkable. Invalid actions, such as moving into walls or outside the maze grid, are prevented. This mechanism guarantees that the agent's movements remain feasible and adhere to the constraints of the environment.



```
def get_next_position(position, action):
    x, y = position
    if action == 'up' and x > 0 and maze[x - 1, y] == 1: x -= 1
    elif action == 'down' and x < maze.shape[0] - 1 and maze[x + 1, y] == 1: x += 1
    elif action == 'left' and y > 0 and maze[x, y - 1] == 1: y -= 1
    elif action == 'right' and y < maze.shape[1] - 1 and maze[x, y + 1] == 1: y += 1
    return (x, y)
```

Figure 0.3 Next position traversal constraints and rewarding

1.3.2 Exploration vs Exploitation and Q-Value update

To balance exploration and exploitation, the ε-greedy strategy is employed. With a probability of ε (set to 0.6 in this implementation), the agent explores by randomly selecting an action. Otherwise, it exploits its current knowledge by choosing the action with the highest Q-value for the current state. This balance is essential for effective learning, as excessive exploration leads to inefficiency, while over-reliance on exploitation can result in suboptimal solutions.

In this problem, the agent receives a reward of 10 for reaching the goal and -1 for all other steps. This incentivizes shorter paths and discourages unnecessary moves.

1.4 Training Process

The agent undergoes multiple training epochs to refine its policy. At the start of each epoch, the agent begins at the designated starting position and attempts to navigate to the goal. Each step involves:

1. Selecting an action based on the ε-greedy strategy.
2. Moving to the next valid state using the get_next_position function.
3. Receiving a reward based on the outcome of the action.
4. Updating the Q-value for the current state-action pair using the Q-learning equation.

Metrics such as cumulative rewards, the average Q-value, and the frequency of state visits are tracked during training. At regular intervals (e.g., every 100 epochs), the agent's progress is visualized through heatmaps and policy visualizations.

1.5 Observations on Policy Visualization and State Visit Heatmap

Policy Visualization and State Visit Heatmap at 100th Epoch

At the 100th epoch, the Q-values and policy visualization show the early stages of learning. The values in the Q-table are relatively uniform and less optimal in regions further from the goal, indicating that the agent is still exploring the maze and learning the best paths. For example:

- **Far from the Goal (Top-Left Region)**: The Q-values are generally negative, with values around -7.83 to -8.20, showing that the agent perceives these states as less valuable due to their distance from the goal.

- **Near the Goal (Bottom-Right Region)**: The Q-values gradually improve as the agent moves closer to the goal. For instance, cells near the goal have values like 10.60 for the optimal path, reflecting the reward structure where reaching the goal provides a significant positive reward.

The arrows representing the policy actions are not consistently aligned toward the goal, suggesting that the agent is still experimenting with exploratory actions. This is evident in areas where the policy does not yet converge to an optimal solution.

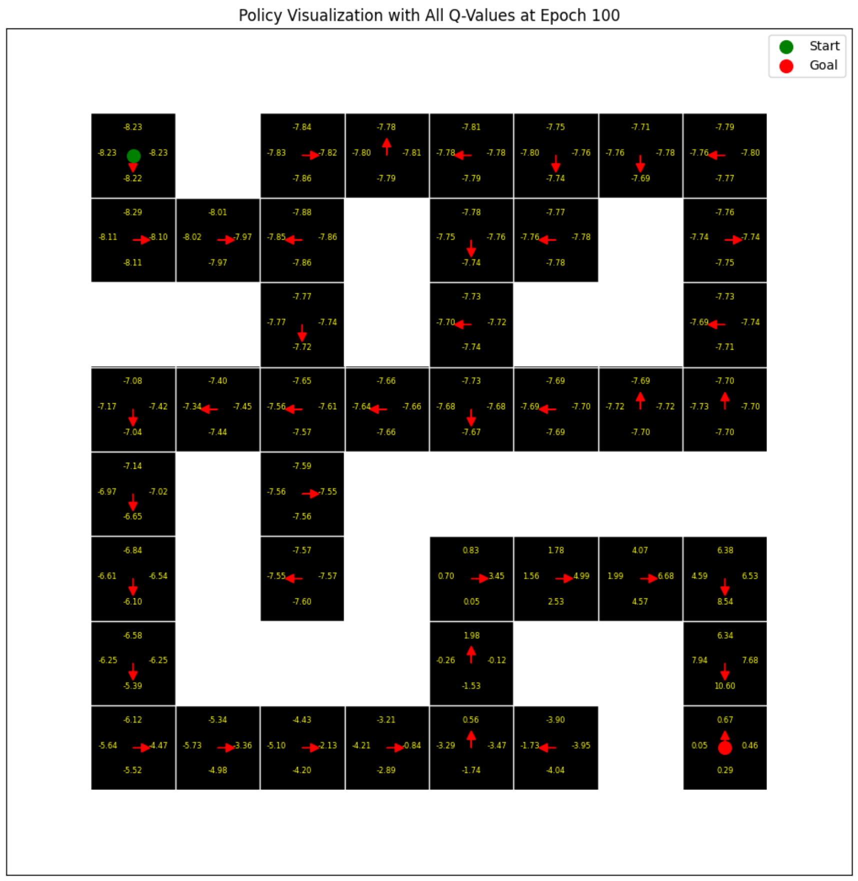


Figure 1.0.4 Policy visualization at 100th epoch

The state visit heatmap at the 100th epoch shows heavy exploration near the starting position (top-left corner). The number of visits diminishes as the agent moves further into the maze, reflecting a higher density of exploration in states close to the starting point. The heatmap also highlights:

- **Sparse Visits Near the Goal**: The goal area (bottom-right) shows fewer visits, indicating that the agent is still in the process of finding efficient paths to reach the goal.

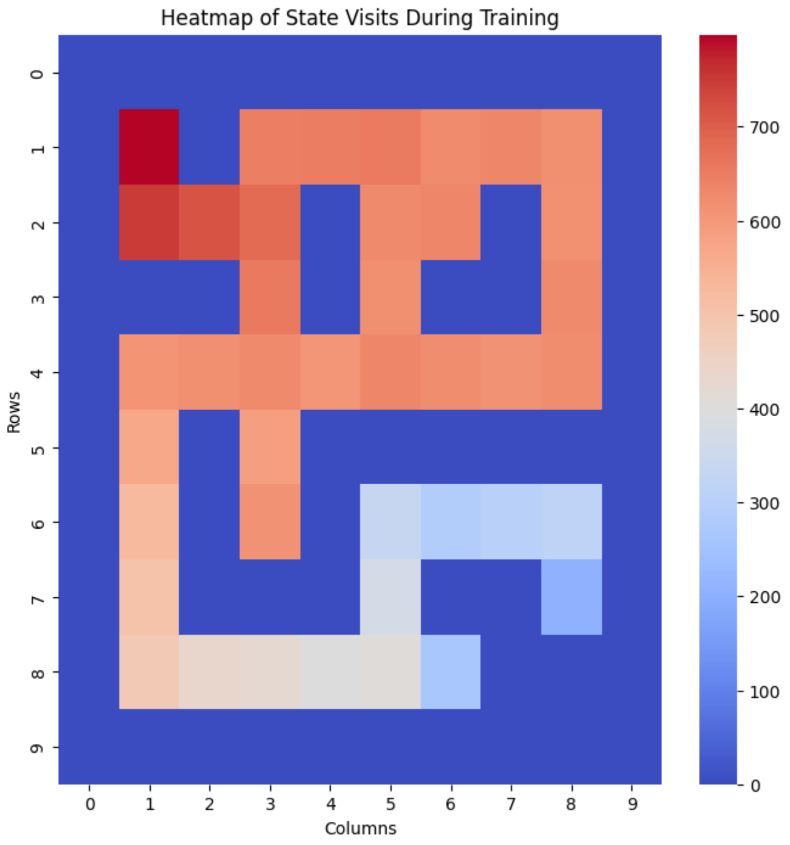


Figure 1.0.5 State Visit Heatmap at 100th epoch

Policy Visualization and State Visit Heatmap at 500th epoch

By the 500th epoch, the Q-values and policy visualization have significantly improved. The agent now demonstrates a more refined and convergent policy:

- **Optimal Path to the Goal**: The Q-values along the optimal path are now positive and significantly higher, reflecting the agent's understanding of the most rewarding actions. For example:
    - Cells leading directly to the goal show consistently high Q-values like 6.69, 8.34, and 10.60.
    - States outside the optimal path maintain low or negative Q-values, such as -7.52 or -7.97, indicating that these actions are suboptimal and less likely to be chosen.
- **Directional Consistency**: The arrows representing the policy actions now point directly toward the goal in most cases, particularly along the shortest path. This shows the agent has learned to avoid unnecessary detours and follow an efficient route.

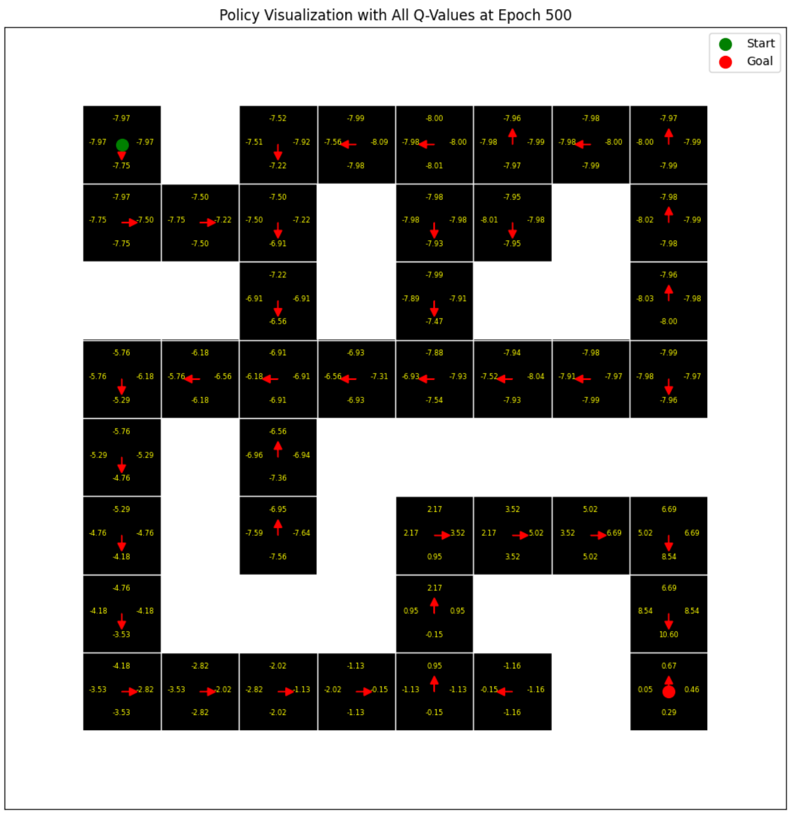


Figure 1.0.6 Policy Visualization at 500th Epoch

The state visit heatmap at the 500th epoch reflects a shift in the agent's behavior:

- **Frequent Visits Along the Optimal Path**: The agent now focuses its visits along the shortest path to the goal, as evidenced by higher state visit counts in these regions. The intensity near the start position decreases compared to earlier epochs, indicating reduced random exploration.

- **Exploration in Non-Optimal Areas**: While exploration continues in non-optimal regions, the frequency is significantly lower, demonstrating the agent's increasing reliance on exploitation as it learns the optimal policy.

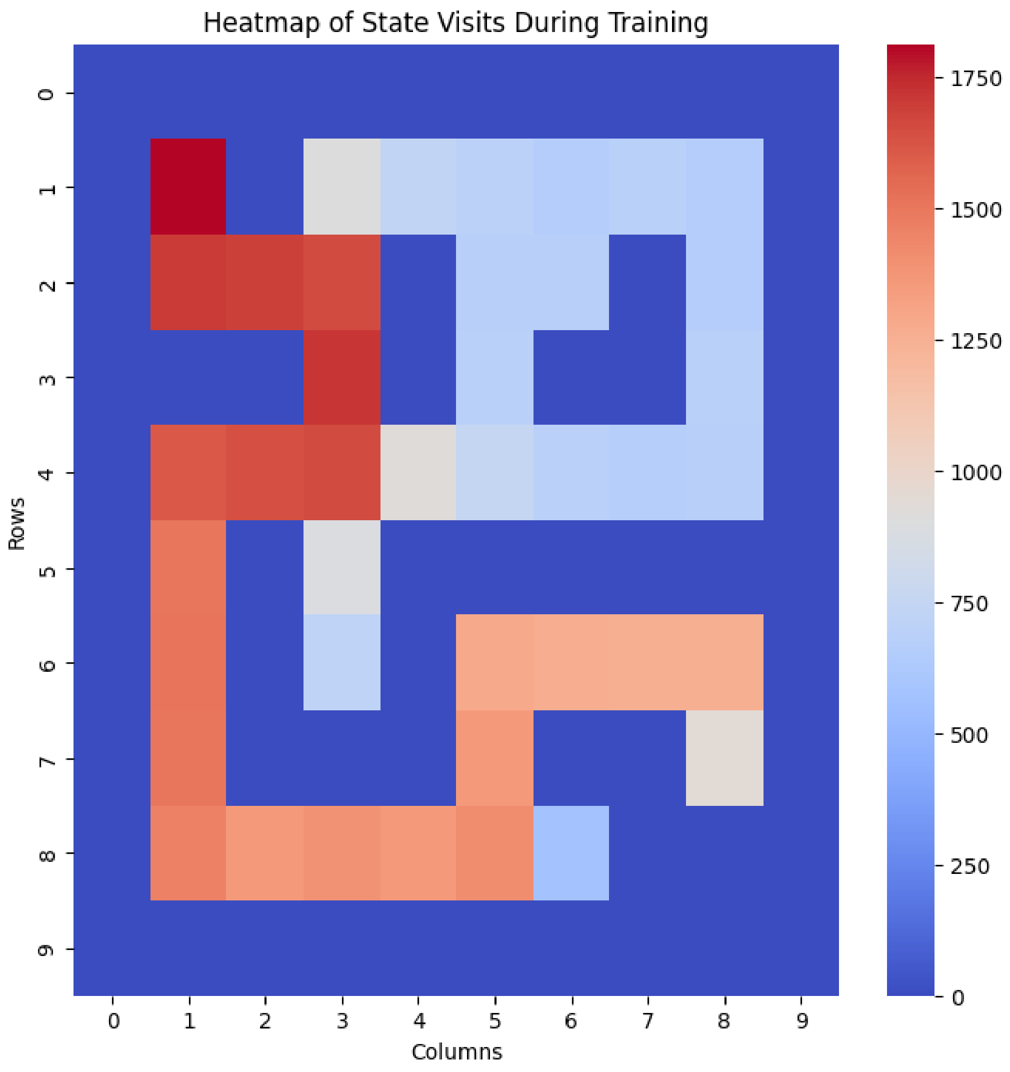


Figure 1.0.7 State Visit Heatmap at 500th Epoch

Key Observations on Policy Changes

1. **Convergence to an Optimal Policy**: Over time, the Q-values for states along the shortest path to the goal have increased, while those for suboptimal states have decreased, reflecting a well-learned policy.
2. **Reduced Exploration**: The agent's focus shifts from wide exploration (at epoch 100) to targeted exploitation (at epoch 500), as evident in the heatmaps.
3. **Improved Efficiency**: The average Q-values and rewards show steady improvement, indicating the agent's increasing efficiency in reaching the goal. For instance:
    - The Q-values near the goal improved from around 10.60 at epoch 100 to stable high values at epoch 500.
    - The directional policy actions become more consistent, reducing unnecessary deviations.
4. **Optimal Paths Solidification**: The repeated state visits along the optimal path highlight the agent’s confidence in following the learned policy.

2. Supervised Learning Model – Sales Forecast

2.1 Overview

Effective sales forecasting is a critical component of strategic decision-making in retail operations. By leveraging historical transaction data, businesses can predict future sales trends, optimize inventory levels, plan promotional activities, and improve overall operational efficiency. This project focuses on forecasting sales using a comprehensive dataset comprising multiple interconnected sources:

1. **Items Dataset**: Contains essential details about the products sold, including their unique codes, types, brands, and sizes. This dataset forms the foundation for understanding the characteristics of each item and their potential influence on sales patterns.
2. **Sales Dataset**: Provides two years of transaction-level data, including sales amounts, units sold, transaction timestamps, and supermarket-specific details. This dataset captures the temporal dynamics of sales and serves as the primary source for generating forecasts.
3. **Promotions Dataset**: Describes promotional activities, such as features and displays, linked to specific items in various supermarkets. This dataset allows us to analyze the impact of promotional campaigns on sales performance.
4. **Supermarkets Dataset**: Contains geographical and logistical details about supermarket locations, enabling spatial analysis of sales trends across different regions.

2.2 Objective

The primary goal of this project is to develop a supervised learning model capable of forecasting sales with high accuracy. By incorporating features derived from these datasets—such as historical sales trends, promotional activities, and product characteristics, the model aims to provide actionable insights to enhance business decision-making.

This holistic approach ensures that all aspects of the retail sales ecosystem, from product attributes to promotional impacts and regional variations, are accounted for, making the forecasts both robust and practical.

2.3 Methodology

2.3.1 Pre-processing of Data

Removing missing values, duplicates and irrelevant features

In the initial stages of data preprocessing, the **Size** column from the Items dataset was excluded due to its inconsistent nature. The column contained a mixture of string values with measurement metrics and ambiguous entries that did not clearly indicate the size of the unit, rendering it unreliable for analysis. Following this, a comprehensive summary of the data types for all columns in each dataset was generated to better understand their structure and ensure compatibility for further processing.

To ensure data integrity and accuracy, duplicates and missing values were systematically removed from all four datasets (Items, Sales, Promotions, and Supermarkets). This step eliminated redundant entries and incomplete records, reducing noise and improving the overall quality of the datasets.

Upon deeper analysis, it became apparent that the Promotions dataset had no meaningful overlap with the sales data. While the Sales dataset recorded transactions spanning weeks 1 to 28, the promotional campaigns detailed in the Promotions dataset occurred during weeks 44 to 98. This temporal disconnect indicated that promotional activities had no direct influence on the current sales records. Consequently, the Promotions dataset was deemed irrelevant for the forecasting task and removed from further analysis.

Similarly, the Supermarkets dataset was excluded to avoid redundancy and potential collinearity issues. The Sales dataset already included supermarket-specific details such as location, rendering the Supermarkets dataset superfluous. Additionally, attributes like postal codes in the Supermarkets dataset overlapped with Supermarket No., introducing the risk of multicollinearity.

By excluding the Promotions and Supermarkets datasets, the analysis was streamlined to focus solely on the relevant datasets—Items and Sales. This refinement ensured that the data used for forecasting was both relevant and free from unnecessary complexity, enhancing the accuracy and reliability of the predictive model.

In addition, under the domain of finance, negative sales often relate to sales returns and to remove a marked sale. In such situations, it’s better to consider that as a product return. It does not support the sales forecast. Therefore, negative values should be omitted.

Removing Outliers With relevance to Amount and Units (Before Aggregation)

The boxplots for both the **Amount** and **Units** variables reveal significant skewness and the presence of extreme outliers, which could distort the predictive capabilities of any forecasting model. These outliers are particularly problematic because they lie far outside the expected range of values, as illustrated by the extreme spikes in the data.

In the case of the **Amount** variable, it can be observed that values marginally close to zero and a sudden spike at around 6,000. Such extreme deviations suggest errors or rare events that could disproportionately influence the model's results. Similarly, for the **Units** variable, while the normal range lies around 1, a significant number of outliers are observed with values extending beyond 100. These extreme values could lead to a cumulative error, causing the model to misrepresent the underlying sales trends.

Choosing IQR instead of Z-Score:

Given the absence of a normal distribution for these variables, the Z-score method for outlier detection is unsuitable. Instead, opted for the Interquartile Range (IQR) method, which is more robust for handling non-normally distributed data. By using IQR, it is possible to systematically identify and remove outliers that fall outside the acceptable range, ensuring that the data better represents typical sales behavior.

This approach not only helps to remove anomalies but also mitigates the risk of skewed predictions caused by extreme values. By focusing on the core range of values depicted in the boxplots, we ensure the model is trained on clean and reliable data, enhancing the accuracy and reliability of the forecasting process.

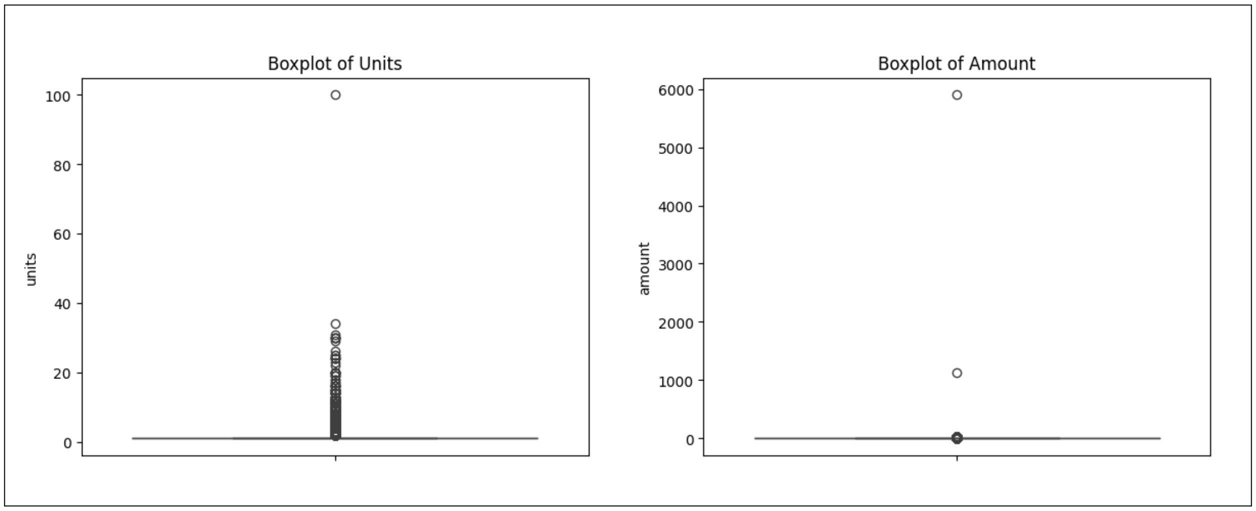


Figure 2.0.1 Boxplots of Units and Amount

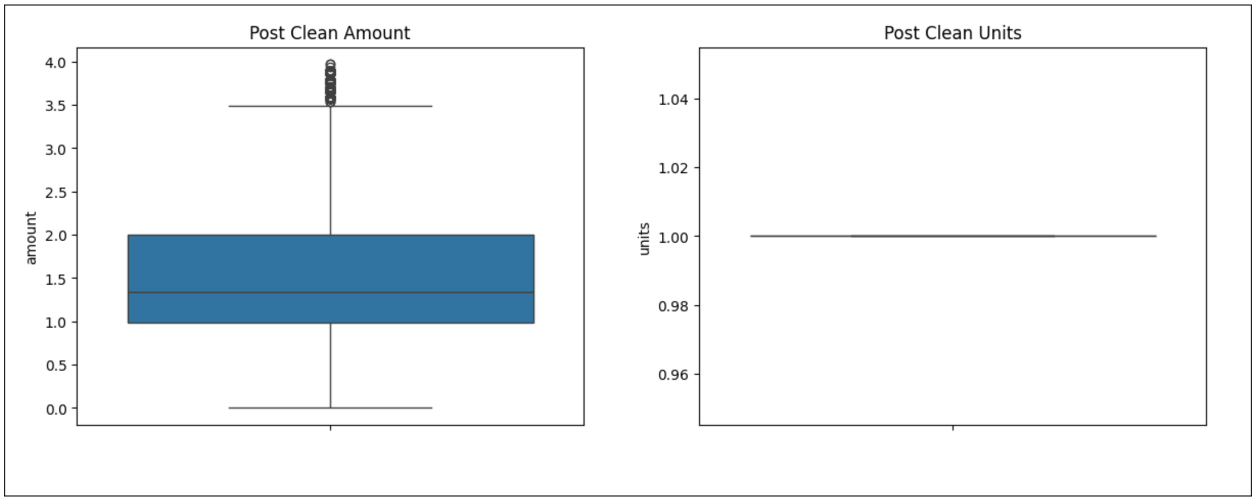


Figure 0.2 Post Removal boxplot of Units and Amount

Feature Engineering

After analyzing both the Sales and Items datasets, a comprehensive understanding of the nature of items being sold was established. To enable deeper insights, the datasets were merged based on the Code feature, creating a unified dataset that integrated product details with corresponding sales transactions. Following the merging, extensive feature engineering was undertaken to enhance the dataset's analytical depth.

Initially, the dataset was structured such that each record represented a single unit of transaction. To better capture patterns and trends, the sales data were aggregated at the item level across different days. This allowed for more meaningful temporal analysis by adding new features to provide additional context about the sales data.

Several engineered features were introduced to improve the dataset's granularity and predictive power:

1. **Day of Week**: Indicates the specific day of the week for each transaction, helping to identify patterns such as peak sales days.
2. **Is Weekend**: A binary feature that marks whether a transaction occurred on a weekend, enabling the analysis of sales variations between weekdays and weekends.
3. **Lag Sales**: This feature captures the sales figures from previous days (e.g., the immediate previous day or seven days prior). Lagged features provide historical context for each transaction, revealing trends and patterns in sales over time.
4. **Rolling Changes**: Rolling averages over defined time windows (e.g., 3-day or 7-day periods) were computed to smooth out fluctuations and observe gradual changes in sales behavior.

These features capture the cumulative effect of prior sales activity and are useful for forecasting.

5. **Total Amount**: Represents the aggregate sales amount for each grouping, providing an overall measure of revenue.
6. **Total Units**: Summarizes the total quantity of units sold within each grouping, indicating item-level demand.
7. **Basket Size Average**: Reflects the average basket size, giving insights into customer purchasing behavior.

The aggregation of the dataset was performed by grouping records based on key attributes, including Code, Type, Brand, Province, Week, Day, and Supermarket. This grouping ensured that all relevant contextual information was captured for each item and location. The engineered features, coupled with this aggregation, significantly enhanced the dataset's utility, laying the groundwork for accurate and insightful sales forecasting.

Once the aggregation was completed, another round IQR was run to remove the outliers. Before the outliers were removed below shows the boxplot of Total Amount before removal and after.

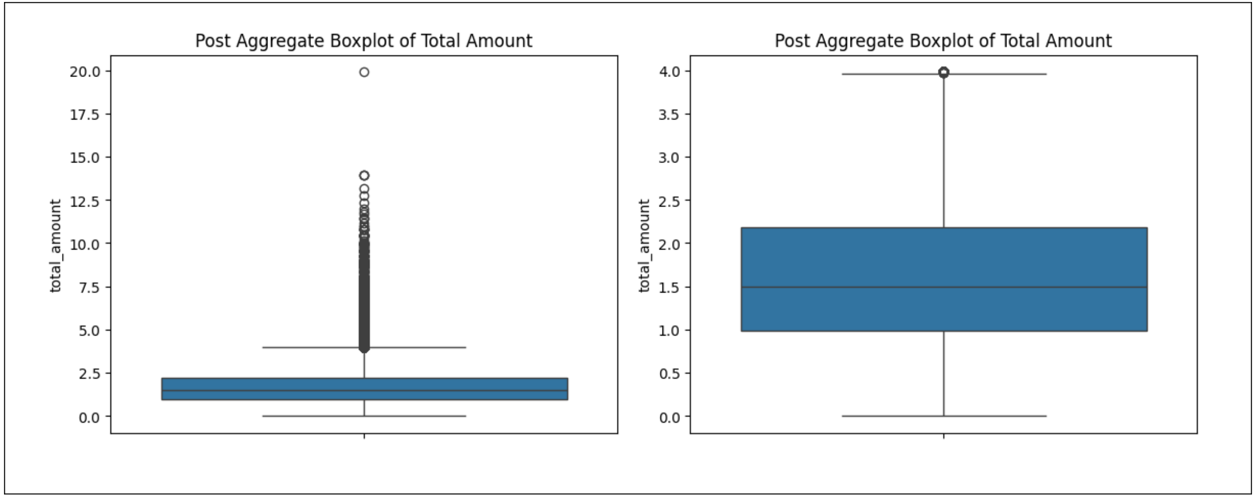


Figure 2.0.3 Pre-Post Boxplots of Total Amount

The first boxplot highlights the presence of numerous outliers in the Total Amount feature. These outliers are represented as individual points that lie significantly above the whisker, indicating values much higher than the interquartile range (IQR). The extreme values likely stem from rare or erroneous transactions, such as unusually high sales amounts that deviate from typical sales behavior. These outliers can skew statistical analyses and negatively impact the performance of predictive models, especially when regression techniques are involved.

The second boxplot illustrates the **Total Amount** distribution after applying outlier removal technique which is IQR. In this plot, the range of values is significantly reduced, and the extreme points observed in the first boxplot are no longer present. The data now conforms to a more compact and representative distribution, making it more suitable for predictive modeling. The maximum value lies closer to the upper whisker, while the central box represents the interquartile range more effectively.

Removing outliers reduces the risk of distortion in the analysis and enhances the predictive model's reliability. By focusing on typical sales patterns, the cleaned data provides a stronger foundation for accurate forecasting.

Correlation Matrix and ANOVA Scores

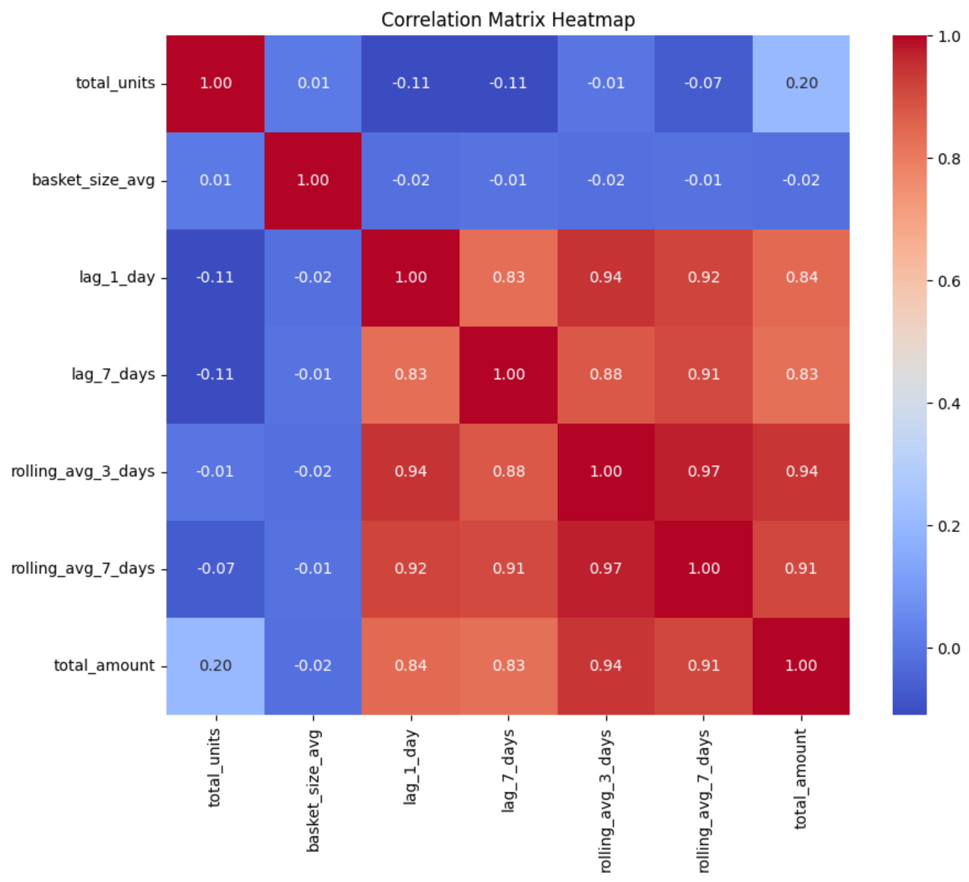


Figure 2.0.4 Correlation Matrix for Numerical Features

The correlation matrix highlights the relationships between numerical features and the target variable, Total Amount:

- **Rolling Averages (3-day and 7-day)**: These features exhibit the highest positive correlation with Total Amount (0.94 and 0.91, respectively). This indicates that recent trends in sales play a crucial role in determining future sales amounts.
- **Lag Sales (1-day and 7-day)**: Both lagged features also show strong positive correlations (0.84 and 0.83, respectively), emphasizing the importance of historical sales data.
- **Total Units**: While its correlation is weaker (0.20), it still reflects some connection with Total Amount.
- **Basket Size Average**: This feature has a near-zero negative correlation (-0.017), suggesting limited relevance to Total Amount.

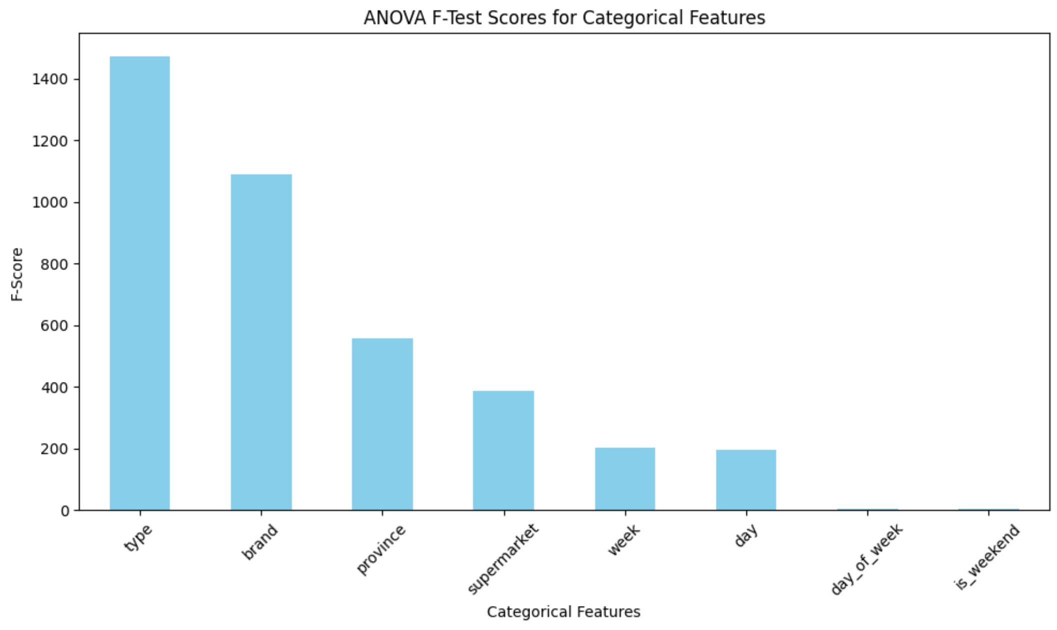


Figure 0.5 ANOVA Scores for categorical data

The ANOVA F-test evaluates the significance of categorical features in relation to the target variable:

- **Type and Brand**: These features have the highest F-scores (1,474.53 and 1,089.38), indicating that the type of item and its brand are critical determinants of sales.
- **Province and Supermarket**: These geographical attributes are moderately significant, with F-scores of 559.20 and 386.97, respectively.

- **Time-Based Features (Week, Day, Day of Week, Is Weekend)**: While these features show relatively low F-scores (201.41, 197.37, 4.92, and 3.20), their temporal nature makes them valuable for capturing sales trends and seasonality.

Based on the analysis:

1. **Numerical Features**:
    - **Rolling Averages** and **Lag Sales** are selected due to their strong correlations with Total Amount. These features effectively capture historical and trend-based patterns, making them essential for forecasting.
2. **Categorical Features**:
    - Despite their lower F-scores, temporal features like Week, Day, and Is Weekend are included because of their time relevance. These features contribute to understanding seasonal and weekly sales patterns.
    - To avoid multicollinearity, only Week is selected as a representative temporal feature, as it overlaps with other time-based engineered features.

The combination of high-correlation numerical features (Rolling Averages, Lag Sales) and time-relevant categorical features ensures a balanced feature set for robust sales forecasting. This selection captures both historical trends and temporal sales behavior, improving the predictive power of the model.

Dataset Split and Feature Selection (Post Aggregation)

The dataset was divided into training and testing sets, with 80% of the data allocated for training and the remaining 20% for testing. The selected features for the model included both categorical and numerical variables. The categorical features consisted of Type, Brand, Province, Week, Supermarket, Day of Week, and Is Weekend, while the numerical features included Lag 1 Day, Lag 7 Days, Rolling Avg 3 Days, and Rolling Avg 7 Days. The target variable was identified as Total Amount.

To prepare the data for modeling, feature scaling and encoding were applied to ensure compatibility and optimize performance. High-cardinality categorical features, such as Brand (95 unique values) and Supermarket (376 unique values), were encoded using **target encoding**. This method replaces each category with the mean of the target variable (Total Amount) for that category. Target encoding is particularly beneficial in this case, as it reduces the dimensionality of the dataset, avoiding the excessive expansion of features that would occur with one-hot encoding. It also preserves the relationship between the categorical values and the target variable, making it an efficient choice for high-cardinality features.

For other categorical features with fewer unique values, such as Week, Type, and Province, **one-hot encoding** was used. This method creates binary columns for each category, effectively transforming the categorical data into a format suitable for machine learning models. One-hot encoding is advantageous for features with a manageable number of categories, as it avoids introducing unnecessary complexity or redundancy.

Numerical features, including Lag 1 Day, Lag 7 Days, Rolling Avg 3 Days, and Rolling Avg 7 Days, were scaled using the **StandardScaler** from scikit-learn. This technique standardizes the numerical values by removing the mean and scaling them to unit variance. For instance, a feature like Lag 1 Day, with values ranging between 10 and 100, would be transformed to have a mean of 0 and a standard deviation of 1. Standardization ensures that all numerical features have consistent scales, which is crucial for models that are sensitive to feature magnitudes, such as linear regression and distance-based algorithms.

Once the preprocessing was complete, the data were split into feature matrices (X_train and X_test) and target variables (y_train and y_test). These processed datasets were then saved to ensure a consistent pipeline for training and evaluation, enabling seamless integration with subsequent modeling steps.

2.4 Evaluation

Two Random Forest models were created and evaluated using K-Fold cross-validation to assess their performance and consistency. The first model was initialized with **100 estimators** and evaluated using **5 splits** in the K-Fold cross-validation process. The second model was configured with **500 estimators** and evaluated with **10 splits**.

For the first model, the cross-validation process was carried out by dividing the training data into 5 folds, where 4 folds were used for training and the remaining fold for validation in each iteration. The negative mean squared error (MSE) was computed during each fold, which was then converted to its root mean squared error (RMSE) counterpart. The resulting cross-validation RMSE scores were [0.2435, 0.2380, 0.2423, 0.2435, 0.2454], with a mean RMSE of **0.2425** and a standard deviation of **0.0025**, indicating consistent performance across folds. Once cross-validation was complete, the model was fit on the entire training dataset to prepare for subsequent testing.

For the second model, the same process was repeated with 500 estimators and 10 folds. The increased number of estimators and splits was expected to enhance the model's predictive performance and stability. However, the resulting RMSE scores were nearly identical to those of the first model, with a mean RMSE of **0.2425** and a standard deviation of **0.0025**, showing no significant improvement despite the increased computational complexity.

This lack of performance gain suggests that increasing the number of estimators and K-Fold splits beyond a certain threshold yield diminishing returns, particularly when the model is already optimized, and the dataset is well-preprocessed. The consistent performance across both models demonstrates the robustness of the Random Forest algorithm and the stability of the dataset after preprocessing. While the second model used a higher number of estimators and more splits, the added computational cost did not translate into meaningful improvements, making the simpler first model a viable choice for this task. Therefore, for further analysis with the test dataset, the first model will be incorporated.

Key Metrics and Implications

Mean Absolute Percentage Error

6.23% reflects that, on average, the model's predictions deviate by 6.23% from the actual values. A MAPE value this low demonstrates that the model performs well in capturing the underlying trends in the data. However, some deviations for higher sales values, as suggested by the residuals, might slightly inflate this metric.

Regression Accuracy

This indicates that 85.36% of the predictions fall within a 10% range of the actual values. While this is a strong result, it also highlights that a small portion of the predictions, especially those for larger sales amounts, may deviate more significantly.

Root Mean Squared Error

This metric quantifies the average magnitude of the prediction errors, placing greater emphasis on larger errors due to its squared nature. It confirms that most errors are small, though a few outliers are likely to contribute to its magnitude. 0.2416 was achieved as Root Mean Squared Error.

Mean Absolute Error

The MAE value of 0.0858 represents the average absolute difference between the predicted and actual values. Unlike RMSE, it treats all errors equally and underscores the model's strong ability to predict with minimal average deviation.

R2 Score

With 91% of the variance in the Total Amount explained by the model, this metric affirms that the Random Forest model captures most of the variability in the target variable. However, the remaining 9% could be attributed to either unaccounted features or random noise in the dataset**.**

Actual vs Rounded Predicted Values

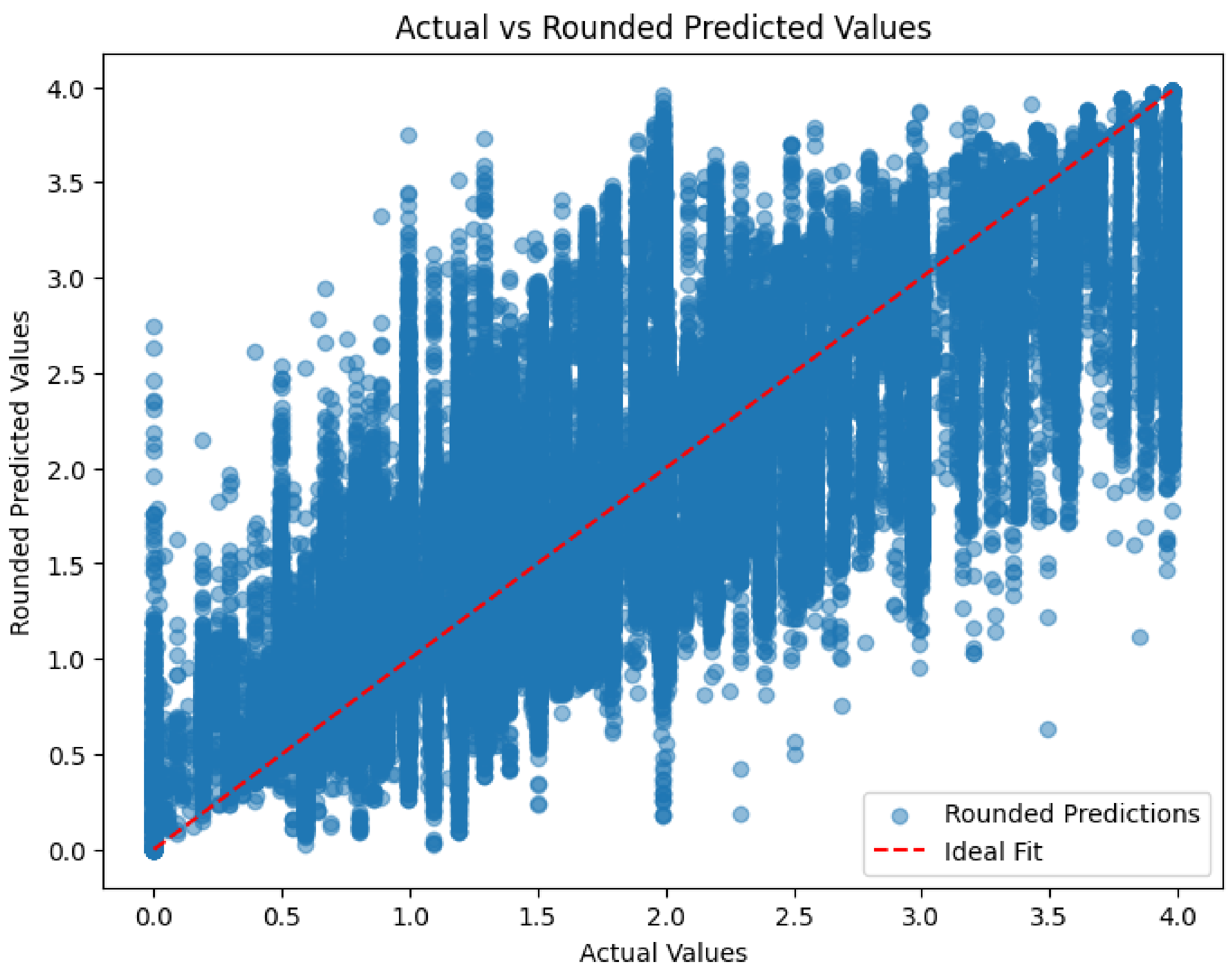


Figure 2.0.6 Actual vs Rounded Predicted Values

The scatter plot demonstrates a strong alignment between actual and predicted values, closely following the ideal fit line. However, a slight spread away from the line for higher values indicates that the model slightly underestimates larger sales amounts. This trend suggests a potential limitation in the model's ability to generalize for extreme values.

Residual Plot

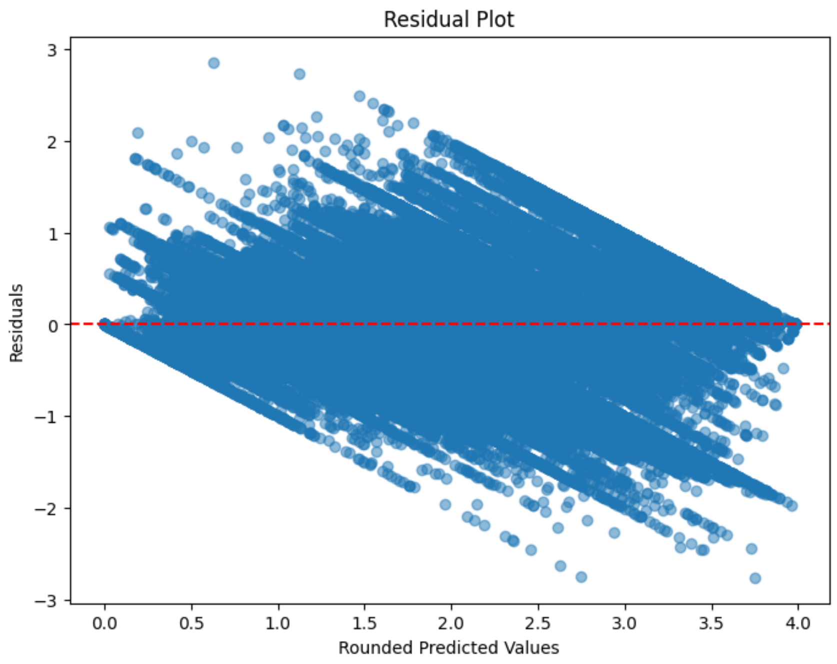


Figure 0.7 Residual Plot

The residual plot reveals a funnel-shaped pattern, indicative of heteroscedasticity. Residuals are tightly clustered around zero for smaller predictions but become increasingly spread for higher predicted values. This suggests that the model is less consistent when predicting larger sales amounts, leading to larger and more variable errors.

Histogram of Residuals

The histogram of residuals shows a sharp peak around zero, confirming that the majority of errors are small and centered near the actual values. The symmetrical shape further supports the model's overall reliability. However, the slightly heavier tails indicate occasional larger errors, corresponding to the observations in the residual plot.

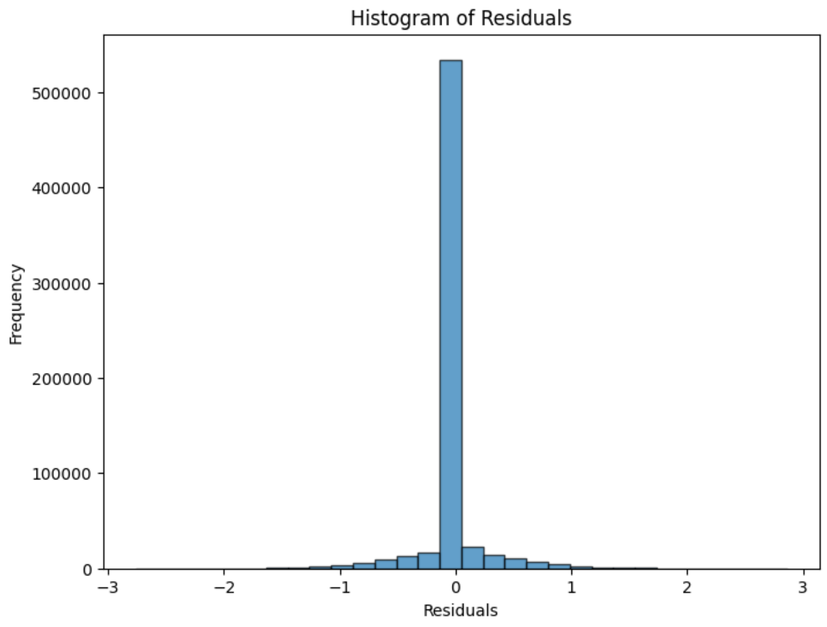


Figure 0.8 Histogram of Residuals

While the model demonstrates strong overall performance, evidenced by low error metrics and high explanatory power, it does show areas for potential improvement:

1. **Handling Heteroscedasticity**:
   - The funnel shape in the residual plot suggests that the model underestimates higher values. Transforming the target variable (e.g., applying a logarithmic or square root transformation) may stabilize the variance and improve predictions for larger values.
2. **Model Tuning**:
   - Adjusting hyperparameters, such as maximum depth or the minimum samples required at leaf nodes, might enhance the model's performance, especially for outliers.

The first Random Forest model provides a robust baseline for sales forecasting, with strong performance metrics and reliable predictions across most of the dataset. However, addressing the observed pattern of increasing residual variance for higher values could further enhance its accuracy and reliability, particularly for extreme sales scenarios.