Business Case Study: Food delivery time estimator using Neural Networks

Problem Description

A food delivery time estimator using neural networks aims to predict the time it will take for food to be delivered from a restaurant to a customer's location. This prediction can help improve customer satisfaction and optimize delivery logistics. The goal of this case study is to estimate delivery times for food orders based on various factors such as the restaurant, order details, and the availability of delivery partners. The dataset provided contains information on multiple deliveries, with each row representing a unique delivery. Key features include market ID, order time, actual delivery time, restaurant category, order details, and delivery partner availability. Each column corresponds to a feature as explained below.

1. market_id: integer id for the market where the restaurant lies
2. created_at: the timestamp at which the order was placed
3. actual_delivery_time: the timestamp when the order was delivered
4. store_primary_category: category for the restaurant
5. order_protocol: integer code value for order protocol (how the order was placed ie: through porter, call to the restaurant, pre-booked, third party, etc)
6. total_items subtotal: the final price of the order
7. num_distinct_items: the number of distinct items in the order
8. min_item_price: the price of the cheapest item in the order
9. max_item_price: price of the costliest item in order
10. total_onshift_partners: number of delivery partners on duty at the time the order was placed
11. total_busy_partners: number of delivery partners attending to other tasks
12. total_outstanding_orders: total number of orders to be fulfilled at the moment

Methodology

1. **Exploratory Data Analysis**

   Import the data and analyze its structure to gain a basic understanding. This includes examining the variables, data types, and the general layout of the dataset. Once the structure is understood, proceed with typical exploratory data analysis steps. These involve checking the characteristics of the dataset, such as distribution, frequency, and any patterns present. Additionally, handle any null values in the data by either removing them or imputing values, ensuring the dataset is clean and ready for further analysis.

2. **Data preprocessing**

The data should be cleaned to ensure it's ready for analysis. This involves handling missing or incorrect values and preparing the data for feature engineering. A crucial step in this process is creating a target column that calculates the delivery time by subtracting the order timestamp (created_at) from the actual delivery time (actual_delivery_time). Additionally, extract relevant time features such as the hour of the day from the order timestamp and the day of the week. Understanding how pandas' datetime data type works and using its built-in functions is essential for these operations. The delivery time should then be calculated in minutes for consistency.

Afterward, encode any categorical columns to prepare the data for machine learning models. At this point, visualize the data to get a clearer understanding of various features through count and scatter plots. Check for outliers, which could skew the analysis, and remove them using appropriate methods. Once the data has been cleaned and outliers removed, visualize it again to see if the overall quality and insights have improved. This will help ensure the data is accurate and ready for modeling.

3. **Model building**

To start model building, the dataset should be split into training and testing sets to evaluate model performance effectively. After splitting, the data needs to be scaled, particularly if a neural network model is used, to ensure faster convergence and better accuracy. Once the data is scaled, a simple neural network can be created. This step involves experimenting with various configurations, including different numbers of layers, neurons, activation functions, and optimizers, to find the best-performing model.

Understanding and tuning these hyperparameters, such as activation functions and optimizers, is crucial to improving model performance. After finalizing the architecture, train the neural network for the required number of epochs to allow the model to learn and improve its predictions over time. This process will involve evaluating the model periodically to check for overfitting and make adjustments as necessary.

4. **Performance Evaluation**

   For performance evaluation, start by analyzing the model's behavior during training by plotting the loss values over time to check for convergence or potential overfitting. In addition to visualizing the loss, monitor the accuracy of the model on both the training and testing datasets to ensure it generalizes well to unseen data.

   After training, evaluate the model using various performance metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE). These metrics provide insight into the model's predictive accuracy and how well it performs in terms of error measurement, helping to identify areas for potential improvement.

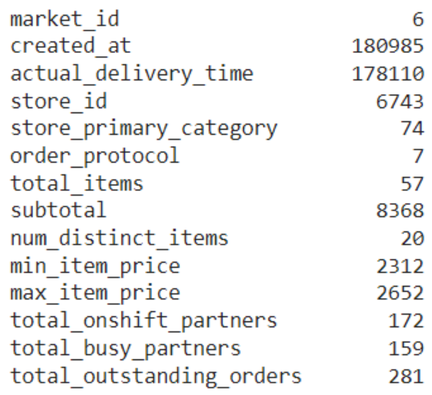
Business Questions to be answered from Analysis

1. Defining the problem statements and where can this and modifications of this be used?
2. List 3 functions the pandas datetime provides with one line explanation.
3. Short note on datetime, time delta, time span (period)
4. Why do we need to check for outliers in our data?
5. Name 3 outlier removal methods.
6. What classical machine learning methods can we use for this problem?
7. Why is scaling required for neural networks?
8. Briefly explain your choice of optimizer.
9. Which activation function did you use and why?
10. Why does a neural network perform well on a large dataset?

Analysis

1. **Exploratory Data Analysis**

- The dataset is imported and analysed to understand the structure. The data set consists of 197428 rows and 14 columns as listed in the problem description. The unique number of values in each column is summarized in Figure 4.1.



```
market_id                      6
created_at                180985
actual_delivery_time      178110
store_id                    6743
store_primary_category        74
order_protocol                 7
total_items                   57
subtotal                    8368
num_distinct_items            20
min_item_price              2312
max_item_price              2652
total_onshift_partners       172
total_busy_partners          159
total_outstanding_orders     281
```

Figure 4. 1: Unique number of values in each column of data set

- The six unique values of market_Id are 1, 2, 3, 4, 5, and 6. The 72% of orders are from market_id 1, 2, and 4.
- Out of 74 store primary categories, 39% of orders are of store primary categories of American, pizza, Mexican, burger, and sandwich.
- There are seven order protocols namely 1, 2, 3, 4, 5, 6, and 7 and 76% of orders are from order protocols 1, 3, and 5. Maximum orders were booked through the app (1.0) or re-booked (3.0).
- 81.11 % of orders are of 1,2,3 or 4 items order and 57 unique and up to 66 items were included in a single order.
- 88.14 % order of 1,2,3 or 4 distinct items order out of 20 distinct items.
- 76.51 % orders of 1 -3 items order.
- The rows with null values are removed from the data set.

2. Data preprocessing

- **Cleaning of data:** Each column checked for zero count value and removed from the dataset.
    - Subtotal is the total price of the order. If it is 0 or less than 0 can be interpreted as some error in the data. So, 16 rows with subtotal = 0 have been dropped.
    - There are 2190 rows with minimum item prices less than or equal to zero. So, dropping those rows.

    - There are 3519 rows with total on-shift partners value less than or equal to zero. So, removing those rows.
    - There are 576 rows with value total busy partners less than or equal to zero. So, removing those rows.
    - There are 260 rows with value total outstanding orders less than or equal to zero. So, removing those rows.
    - There are 686 rows with the minimum price of the order being greater than the maximum item price which is logically incorrect. So, dropping those rows.
- **Feature engineering:** The target column time taken ("delivery time in minutes") in each delivery from the order timestamp (created_at) and delivery timestamp (actual_delivery_time) was created by subtracting the order timestamp from the delivery timestamp and converted to datetime format for easily working with them.
- Two new columns "hour of the day", and "day of the week" were also added from the order time column "created_at". Two columns "created_at"', and "actual_delivery_time" are removed.
- **Encoding categorical columns:** Two non-numerical columns "'store_primary_category", and "store_id" are encoded using label encoding for further analysis.
- **Data visualization and cleaning:** Visualize various columns for a better understanding of Count and scatter plots.
    - The count plot of delivery hours is displayed in Figure 4.2. The greatest number of orders is during the overnight of which the peak is at 2 am (18.72%) followed by 1 am (14.63%) and 3 am (13.46%) and the least number of orders are during the afternoon time (~0%).

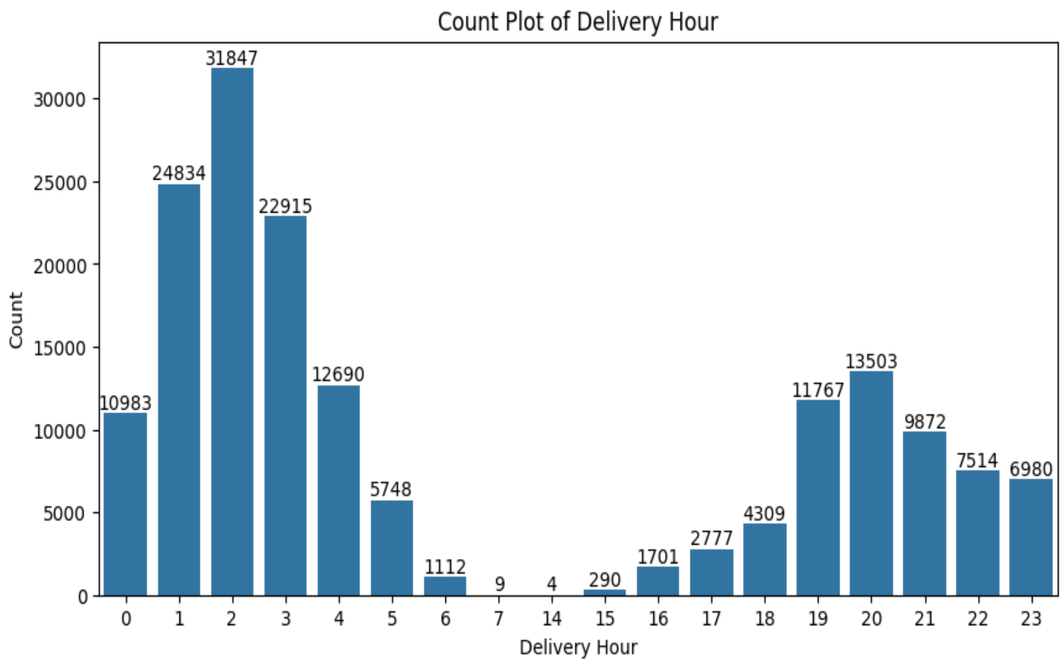


Figure 4. 1: Count plot of delivery hours

- ○ The count plot of the delivery week is displayed in Figure 4.3. About 49% of orders are on weekends (Friday, Saturday, and Sunday).
- ○ The count of market_id is shown in Figure 4.4. The 72% of orders are from market_id 1, 2, and 4.

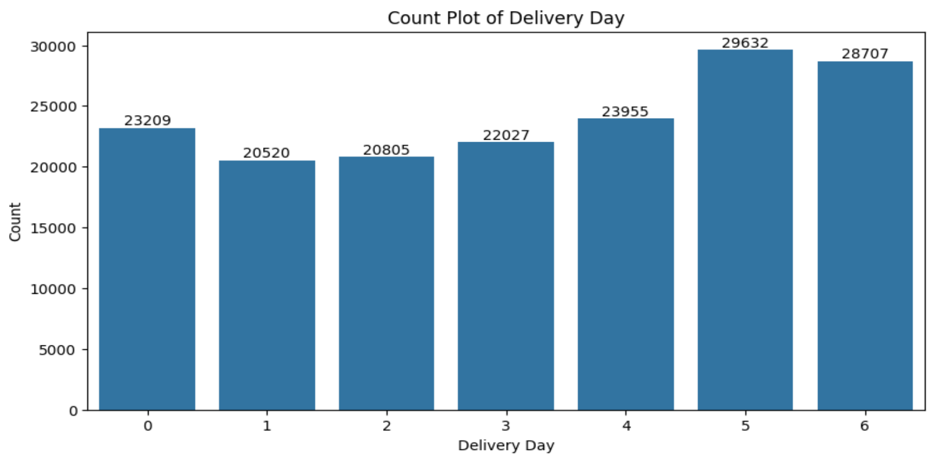


Figure 4. 2: Count Plot of Delivery Day

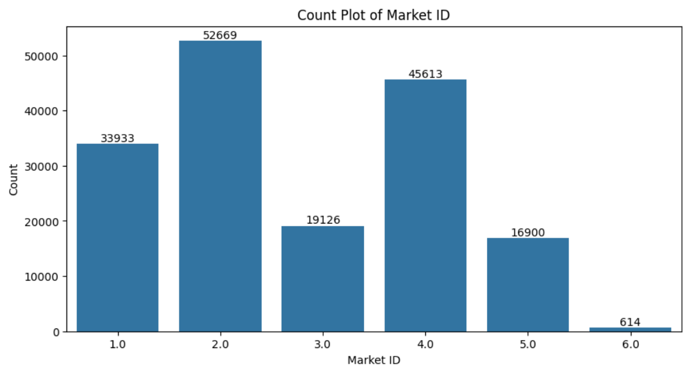


Figure 4. 3: Count plot of market_id

- ○ The count of order protocol is shown in Figure 4.5. 76% of orders are from order protocols 1, 3, and 5. Maximum orders were booked through the app (1.0) or re-booked (3.0).

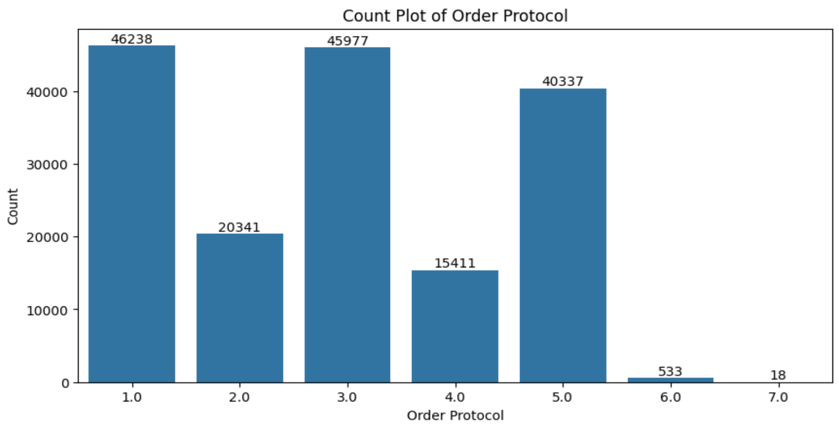


Figure 4. 4: Count plot of order protocol

- The heatmap showing the correlation between different features in the dataset is illustrated in Figure 4.6. Some key observations from the heatmap or correlation matrix are:
  - ○ 'time_taken' has a moderate positive correlation with 'total_outstanding_orders'. This suggests that as the number of outstanding orders increases, the delivery time also increases.
  - ○ 'time_taken' has a moderate negative correlation with 'total_onshift_partners'. This suggests that the delivery time tends to decrease as the number of delivery partners on shift increases.
  - ○ 'total_busy_partners' and 'total_outstanding_orders' have a strong positive correlation. This is expected, as more outstanding orders would lead to more busy partners.

    - 'total_items' and 'subtotal' have a strong positive correlation, which is expected as more items would lead to a higher total price.
    - 'num_distinct_items' and 'total_items' have a moderate positive correlation. This suggests that as the number of distinct items increases, the total number of items also tends to increase.
    - 'max_item_price' and 'min_item_price' have a moderate positive correlation. This suggests that as the price of the costliest item increases, the price of the cheapest item also tends to increase.
- **Check if the data contains outliers:** The boxplots (illustrated in Figure 4.7) of the dataset were plotted to identify the outliers.
    - The outliers were removed from numerical columns in the data frame by calculating the interquartile range (IQR) for each column and filtering out values that fall below the lower bound or above the upper bound, defined as 1.5 times the IQR from the first and third quartiles.
    - The Boxplots of the dataset after removing the outliers are shown in Figure 4.8.

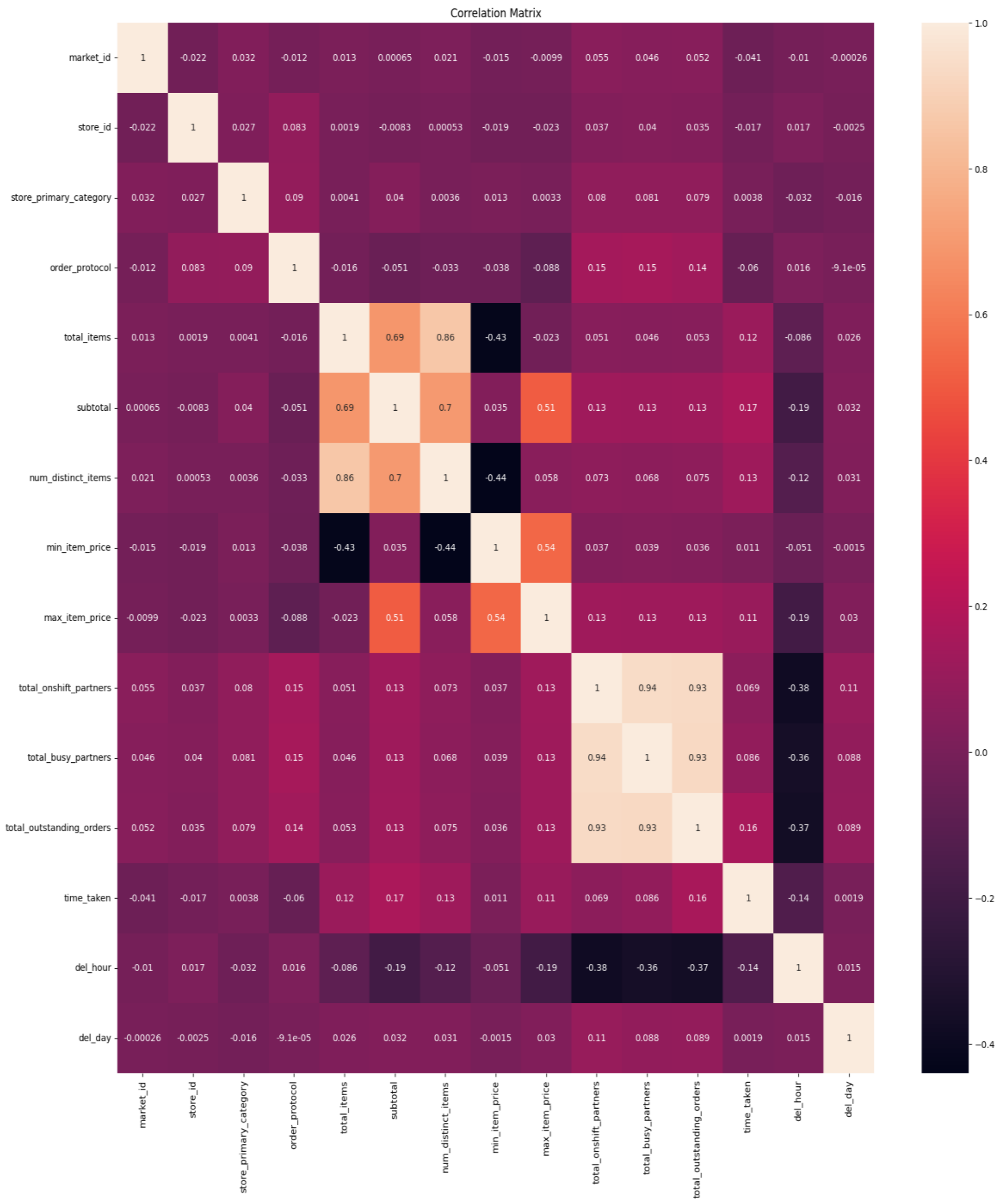


Figure 4. 5: Correlation matrix of the dataset.

3. Model building

- **Split the data in train and test:** The data set was split into a training set (80%) and a testing set (20%).
- The feature importance of each feature was calculated using random forest and displayed in Figure 4.9. The MSE, RMSE, MAE, and R2 score of the random forest model is given below
    - MSE: 151.43209362604435

- RMSE: 12.305774808034005
- MAE: 9.767052793324499
- R2_score: 0.22999711958573066

- **Scaling the data for neural networks:** MinMaxScaler from sklearn was used to scale the features in the dataset between 0 and 1. It then splits the scaled data into training and testing sets using an 80/20 split and a boxplot of the training set to check the distribution of each feature visualized in Figure 4.10.

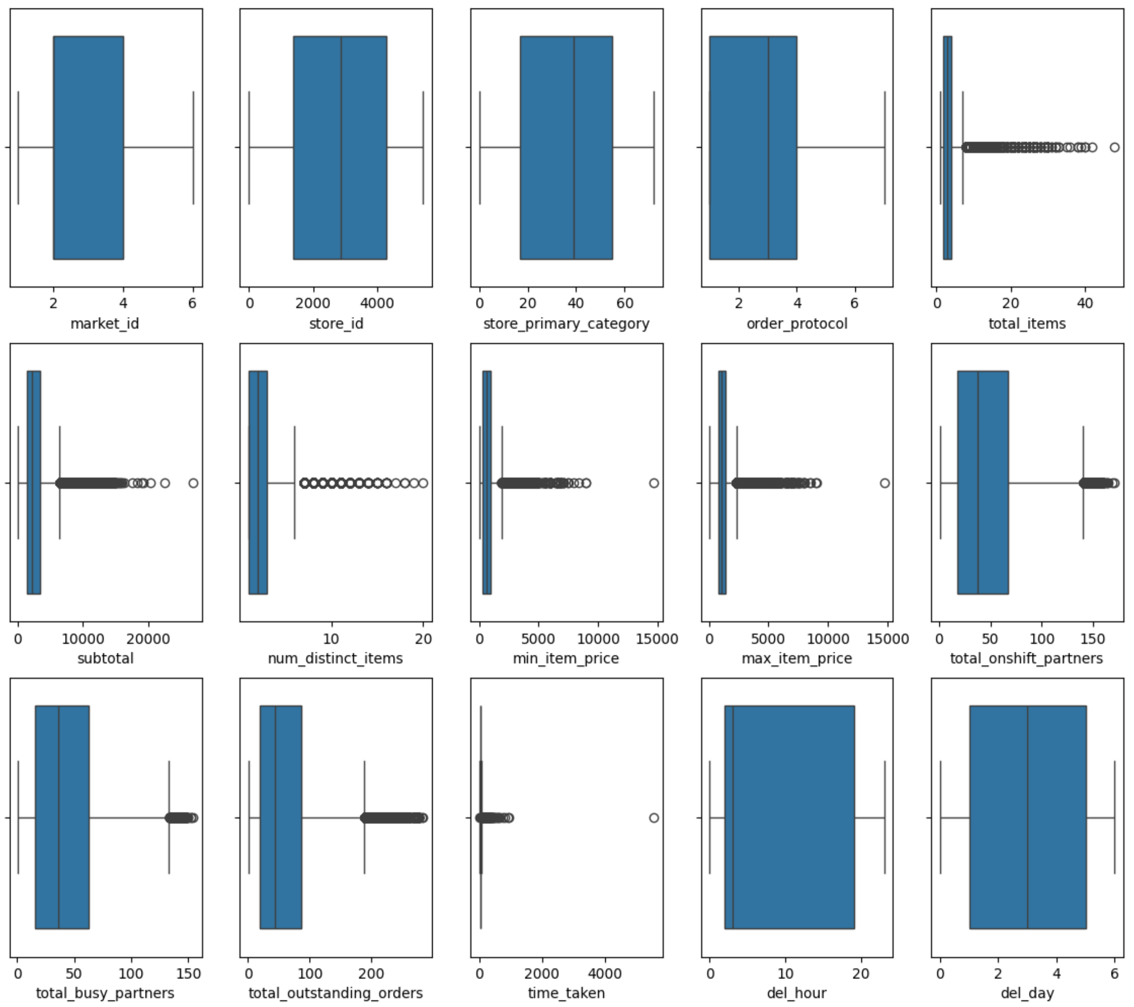


Figure 4. 6: Boxplot of the dataset before removing the outliers

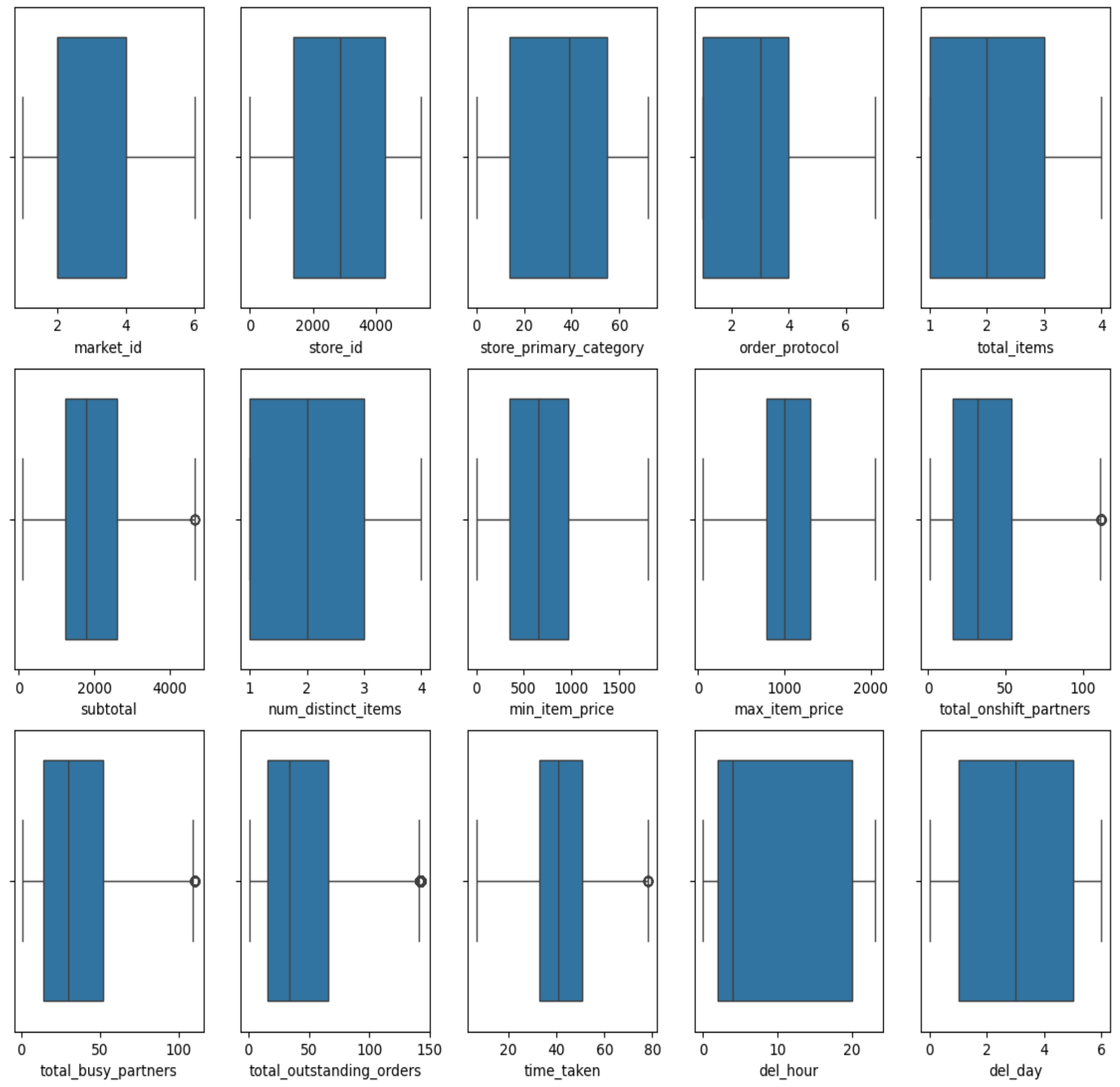


Figure 4. 7: Boxplot of the dataset after removing the outliers

- **3Creating a simple neural network:** A simple neural network model using TensorFlow's Keras API consists of four fully connected layers (Dense), with 64, 516, 1024, and 256 neurons respectively, using the ReLU activation function for non-linearity. The output layer has one neuron with a linear activation function, suitable for a regression task where the goal is to predict a continuous value.
- **Training the neural network for the required amount of epochs:** The neural network model was compiled using the mean squared error ('mse') as the loss function, the Adam optimizer with a learning rate of 0.01, and tracks the metrics 'mse', 'mae' (mean absolute error), and 'accuracy'. The model is then trained for 30 epochs with a batch size of 512, using 80% of the training data and 20% as

validation data, providing updates on the training process at each epoch (verbose=1).

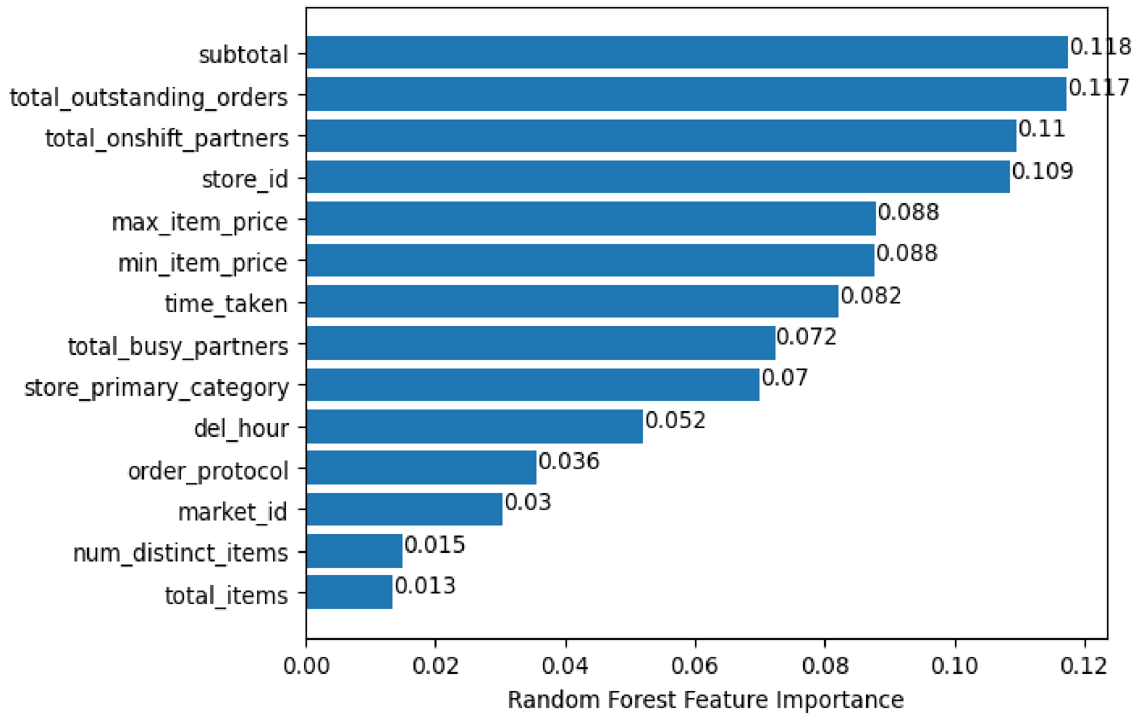


Figure 4. 8: Random Forest Feature Importance of the Dataset

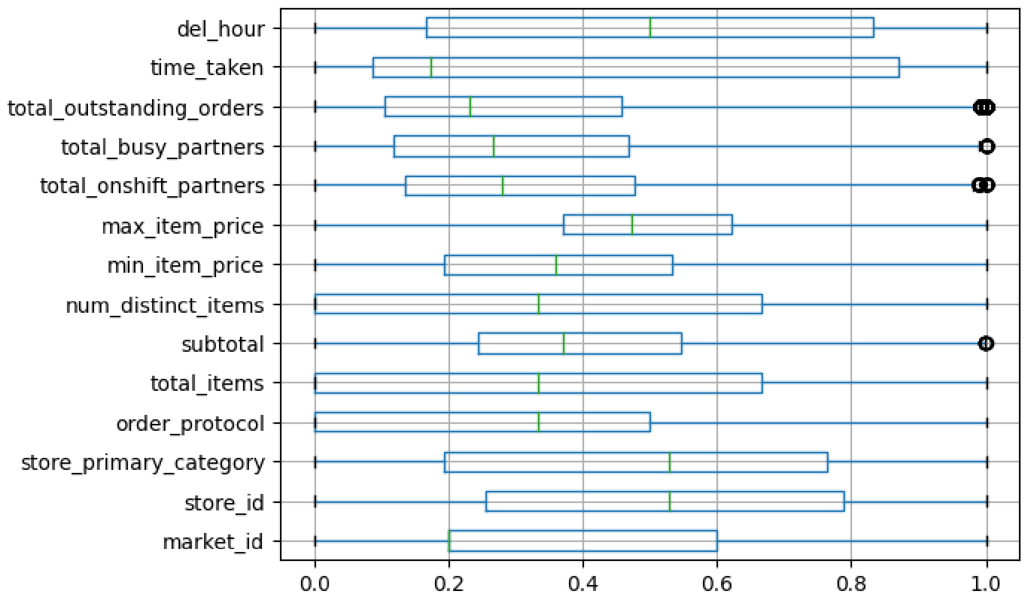


Figure 4. 9: Boxplot of the training set to check the distribution of each feature

4. Performance Evaluation

- **Plotting the losses and checking the accuracy of the model:** The training and validation loss over the epochs are plotted in Figure 4.11 and training and validation accuracy over the epochs are plotted in Figure 4.12.

- **Checking its various metrics like MSE, RMSE, MAE:** The MSE, RMSE, MAE, and MAPE of the given neural network model is given below
    - MSE : 140.94032127317803
    - RMSE : 11.871828893358344
    - MAE : 9.456927251564325
    - MAPE : 0.24514081968403026
  - Model was trained and tested using different configurations such as l1/l2 regularization, Dropout, and Batch Normalization.

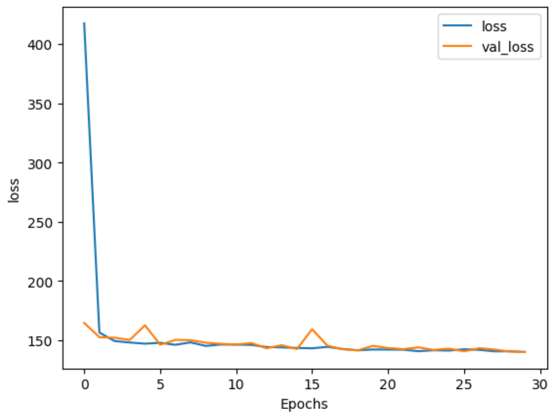


Figure 4. 10: Training and validation loss over the epochs

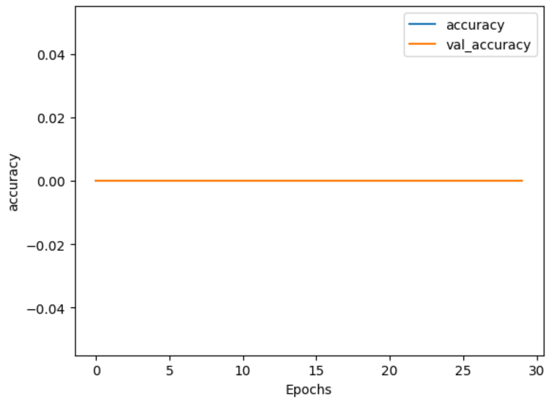


Figure 4. 11: Training and validation accuracy over the epochs

Insights and Recommendations

Insights

- The six unique values of market_Id are 1, 2, 3, 4, 5, and 6. The 72% of orders are from market_id 1, 2, and 4.
- Out of 74 store primary categories, 39% of orders are of store primary categories of American, pizza, Mexican, burger, and sandwich.
- There are seven order protocols namely 1, 2, 3, 4, 5, 6, and 7 and 76% of orders are from order protocols 1, 3, and 5. Maximum orders were booked through the app (1.0) or re-booked (3.0).
- 81.11 % of orders are of 1,2,3 or 4 items order and 57 unique and up to 66 items were included in a single order.
- 88.14 % order of 1,2,3 or 4 distinct items order out of 20 distinct items
- 76.51 % order of 1 -3 items order
- Dropped null or missing values.
- Subtotal is the total price of the order. If it is 0 or less than 0 then it means there is some error in the data. So, 16 rows with subtotal == 0 have been dropped.
- There are 2190 rows with min_item_price <=0. So, dropping those rows.
- There are 3519 rows with value <=0 for df["total_onshift_partners"]. So, removing those rows.
- There are 576 rows with value <=0 for df["total_busy_partners"]. So, removing those rows.
- There are 260 rows with value <=0 for df["total_outstanding_orders"]. So, removing those rows
- There are 686 rows with df.min_item_price>df.max_item_price. So, dropping those rows
- Created the target column “time_taken” (time taken in minutes for delivery) from order timestamp and delivery timestamp and deleted the columns df['actual_delivery_time'] and df['created_at'].
- Two new columns labeled “del_hour” and “del_day” were created from df['created_at'] columns which represent the delivery order created hour and day.
- Encoded 'store_primary_category' and ‘store_id’ using label encoding technique
- The greatest number of orders is during the overnight of which the peak is at 2 am (18.72%) followed by 1 am (14.63%) and 3 am (13.46%) and the least number of orders are during the afternoon time (~0%)
- About 49% of orders are on weekends (Friday, Saturday, and Sundays)

- 'time_taken' has a moderate positive correlation with 'total_outstanding_orders'. This suggests that as the number of outstanding orders increases, the delivery time also increases.
- 'time_taken' has a moderate negative correlation with 'total_onshift_partners'. This suggests that the delivery time tends to decrease as the number of delivery partners on shift increases.
- 'total_busy_partners' and 'total_outstanding_orders' have a strong positive correlation. This is expected, as more outstanding orders would lead to more busy partners.
- 'total_items' and 'subtotal' have a strong positive correlation, which is expected as more items would lead to a higher total price.
- 'num_distinct_items' and 'total_items' have a moderate positive correlation. This suggests that as the number of distinct items increases, the total number of items also tends to increase.
- 'max_item_price' and 'min_item_price' have a moderate positive correlation. This suggests that as the price of the costliest item increases, the price of the cheapest item also tends to increase.

Recommendations

- Further investigate the relationships between features and delivery time.
- Explore other machine learning algorithms (e.g., Gradient Boosting, Support Vector Regression).
- Fine-tune hyperparameters using techniques like grid search or Bayesian optimization.
- Explore more advanced deep learning architectures (e.g., Recurrent Neural Networks, Convolutional Neural Networks).
- Investigate the impact of different scaling methods on model performance.
- Consider using cross-validation to obtain a more robust model evaluation.
- Deploy the model to a production environment for real-time predictions.
- Monitor the model's performance over time and retrain it as needed.

Business Questions to be answered from Analysis

1. Defining the problem statements and where can this and modifications of this be used?

The goal is to estimate delivery times for food orders based on various factors such as the restaurant, order details, and the availability of delivery partners. The dataset provided contains information on multiple deliveries, with each row representing a unique delivery. Key features include market ID, order time, actual delivery time, restaurant category, order details, and delivery partner availability.

2. List 3 functions the pandas datetime provides with one line explanation.

- **to_datetime:**
  - Converts the input to datetime.
  - Example: pd.to_datetime(1490195805, unit='s') : Timestamp('2017-03-22 15:16:45')
- **dt (DatetimeAccessor):**
  - Accessor object for datetime like properties of the Series values.
  - Example: df['timestamp_column'].dt.day
- **date_range:**
  - Returns a fixed-frequency datetime index, with day (calendar) as the default frequency.
  - Example: pd.date_range(start='2022-01-01', end='2022-01-10', freq='D')

3. Short note on datetime, timedelta, time span (period)

- **Datetime:** In Python, the datetime module provides classes for working with dates and times. The datetime class is a part of this module and represents a point in time with both date and time components. Example:
  ```
  from datetime import datetime
   current_time = datetime.now()
   print(current_time) : prints the current time in format yyyy-mm-dd hh:mm:ss
  ```
- **Timedelta:** The timedelta class, also part of the datetime module, represents the duration between two dates or times. It allows you to perform arithmetic with datetime objects. Example:
  ```
  from datetime import datetime, timedelta
  yesterday = datetime.now() - timedelta(days=1)
  print(yesterday) prints the current time of day before in format yyyy-mm-dd hh:mm:ss
  ```

- Time Span (Period): In pandas, a time span or period is represented by the Period class. It's a fixed-frequency interval of time, such as a day, a month, or a year. It's useful for time-based indexing and other time-related operations. Example:

  ```
  import pandas as pd
  period = pd.Period('2022-03', freq='M')
  print(period)
  ```

4. Why do we need to check for outliers in our data?

   Checking for outliers in data is crucial for several reasons in data analysis and modeling:

   a. **Impact on Descriptive Statistics:** Outliers can significantly affect basic descriptive statistics, such as the mean and standard deviation. The mean is sensitive to extreme values, and the presence of outliers can distort its value, leading to a misrepresentation of the central tendency.

   b. **Model Performance:** Outliers can adversely impact the performance of statistical and machine learning models. Algorithms like linear regression are particularly sensitive to outliers as they may influence the slope and intercept of the regression line, leading to inaccurate predictions.

   c. **Assumption Violations:** Many statistical techniques assume that the data is normally distributed or follows a certain pattern. Outliers may violate these assumptions and lead to biased parameter estimates, affecting the validity of statistical inferences.

   d. **Data Quality and Integrity:** Outliers might indicate errors in data collection, data entry, or measurement. Identifying and addressing outliers is essential for ensuring the quality and integrity of the dataset, preventing misleading or incorrect conclusions.

   e. **Robustness of Models:** Some machine learning algorithms are less robust in the presence of outliers. For example, clustering algorithms or distance-based methods may assign undue importance to outliers, affecting the formation of clusters or the calculation of distances.

   f. **Data Exploration and Understanding:** Outliers can provide insights into interesting phenomena or anomalies in the data. Investigating outliers can lead to a better understanding of the underlying processes or uncover important information that may require special attention or further investigation.

g. **Visualizations and Interpretability:** Outliers can distort visualizations, making it challenging to interpret and draw meaningful insights from graphs or charts. Addressing outliers improves the clarity and interpretability of visual representations of the data.

h. **Model Generalization:** When building predictive models, the goal is often to create a model that generalizes well to new, unseen data. Outliers in the training data may not be representative of the broader population, potentially leading to poor generalization.

5. Name 3 outlier removal methods.
   - Z-Score Method
   - Interquartile Range (IQR) Method
   - Isolation Forest

6. What classical machine learning methods can we use for this problem?
   - Random Forest Regression
   - Gradient Boosting Regression (e.g., XGBoost)
   - Linear Regression

7. Why is scaling required for neural networks?
   - Faster and More Stable Convergence
   - Improved Performance

8. Briefly explain your choice of optimizer.
   Adam, which stands for Adaptive Moment Estimation, is a popular optimizer used in neural network training to adjust weights and minimize loss. It combines the strengths of two other optimizers, AdaGrad and RMSprop, addressing their limitations and offering several advantages:
   - Key Features of Adam:
     1. Adaptive Learning Rates
     2. Handles Sparse Gradients
     3. Fast Convergence
     4. Less Sensitive to Learning Rate Choice

5. Bias Correction

9. Which activation function did you use and why?
   - **Sigmoid Function:** This S-shaped function maps input values between 0 and 1. Pros: Often the first activation function encountered due to its simplicity. Outputs are easily interpretable as probabilities (between 0 and 1). Cons: Suffers from vanishing gradients for large negative inputs. Not zero-centered, which can cause issues in deeper networks.
   - **ReLU (Rectified Linear Unit):** The most widely used activation function in recent times. Outputs the input directly if it's positive, otherwise outputs zero. Pros: Computationally efficient, avoids vanishing gradients (gradient is simply 1 for positive inputs). Cons: Can suffer from the "dying ReLU" problem where neurons get stuck at zero if they receive negative inputs during training.
   - **Softmax Function:**Typically used in the output layer of a network for multi-class classification problems. Outputs a vector of probabilities where the elements sum to 1. Each element represents the probability of the input belonging to a specific class.

10. Why does a neural network perform well on a large dataset?
    - With a large dataset, the network has more information (data points) to learn from, allowing it to capture the intricate relationships and patterns within the data. This leads to a more accurate representation of the underlying function or relationship between features and target variables.
    - Reduced Variance and Improved Generalization
    - Learning Complex Relationships
    - A larger dataset provides more diverse gradient updates, guiding the optimization process towards a better minimum loss value. This can lead to faster convergence and potentially better performance.

Chapter 4: Business Case Study 4

Optimizers for Neural Networks Regression

Problem Description

The goal is to estimate delivery times for food orders based on various factors such as the restaurant, order details, and the availability of delivery partners. The dataset provided contains information on multiple deliveries, with each row representing a unique delivery. Key features include market ID, order time, actual delivery time, restaurant category, order details, and delivery partner availability. Each column corresponds to a feature as explained below.

13. market_id: integer id for the market where the restaurant lies
14. created_at: the timestamp at which the order was placed
15. actual_delivery_time: the timestamp when the order was delivered
16. store_primary_category: category for the restaurant
17. order_protocol: integer code value for order protocol (how the order was placed ie: through porter, call to the restaurant, pre-booked, third party, etc)
18. total_items subtotal: the final price of the order
19. num_distinct_items: the number of distinct items in the order
20. min_item_price: the price of the cheapest item in the order
21. max_item_price: price of the costliest item in order
22. total_onshift_partners: number of delivery partners on duty at the time the order was placed
23. total_busy_partners: number of delivery partners attending to other tasks
24. total_outstanding_orders: total number of orders to be fulfilled at the moment

Business Questions to be answered from Analysis

11. Defining the problem statements and where can this and modifications of this be used?
12. List 3 functions the pandas datetime provides with one line explanation.
13. Short note on datetime, time delta, time span (period)
14. Why do we need to check for outliers in our data?
15. Name 3 outlier removal methods.
16. What classical machine learning methods can we use for this problem?
17. Why is scaling required for neural networks?
18. Briefly explain your choice of optimizer.
19. Which activation function did you use and why?
20. Why does a neural network perform well on a large dataset?

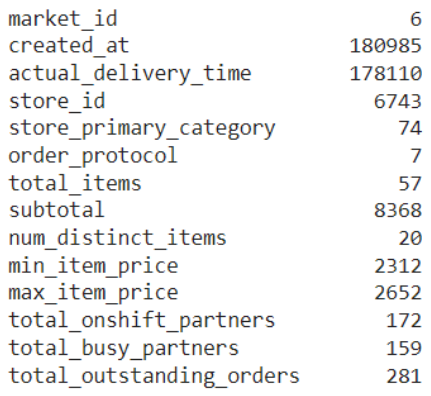
Methodology

5. Import the data and understand the structure of the data:
    - Usual exploratory analysis steps like checking the structure & characteristics of the dataset
    - Handling null values
6. Data preprocessing
    - Cleaning of data
    - Feature engineering: Creating the target column time taken in each delivery from order timestamp (created_at) and delivery timestamp (actual_delivery_time)
    - Getting the hour of the day from the order time and also the day of the week
    - Understanding pandas datetime data type and what function it provides by default
    - Get delivery time in minutes
7. Encoding categorical columns
8. Data visualization and cleaning
    - Visualize various columns for a better understanding of Count, and scatter plots
9. Check if the data contains outliers
    - Removing outliers by any method
    - Plotting the data again to see if anything has improved
10. Split the data in train and test
11. Scaling the data for neural networks.
12. Creating a simple neural network
    - Trying different configurations
    - Understanding different activation functions, optimizers and other hyperparameters.
13. Training the neural network for required amount of epochs
14. Plotting the losses and checking the accuracy of the model
15. Checking its various metrics like MSE, RMSE, MAE

Analysis

5. **Import the data and understand the structure of the data:**

- The data set consists of 197428 rows and 14 columns as listed in the problem description. The unique number of values in each column is summarized in Figure 4.1.



```
market_id                      6
created_at                180985
actual_delivery_time      178110
store_id                    6743
store_primary_category        74
order_protocol                 7
total_items                   57
subtotal                    8368
num_distinct_items            20
min_item_price              2312
max_item_price              2652
total_onshift_partners       172
total_busy_partners          159
total_outstanding_orders     281
```

Figure 4. 12: Unique number of values in each column of data set

- The six unique values of market_Id are 1, 2, 3, 4, 5, and 6. The 72% of orders are from market_id 1, 2, and 4.
- Out of 74 store primary categories, 39% of orders are of store primary categories of American, pizza, Mexican, burger, and sandwich.
- There are seven order protocols namely 1, 2, 3, 4, 5, 6, and 7 and 76% of orders are from order protocols 1, 3, and 5. Maximum orders were booked through the app (1.0) or re-booked (3.0).
- 81.11 % of orders are of 1,2,3 or 4 items order and 57 unique and up to 66 items were included in a single order.
- 88.14 % order of 1,2,3 or 4 distinct items order out of 20 distinct items
- 76.51 % order of 1 -3 items order

6. **Handling null values**
   - The rows with null values are removed from the data set.

7. **Data preprocessing**
   - **Cleaning of data:** Each column checked for zero count value and removed from the dataset
     - Subtotal is the total price of the order. If it is 0 or less than 0 can be interpreted as some error in the data. So, 16 rows with subtotal == 0 have been dropped.

- There are 2190 rows with min_item_price <=0. So, dropping those rows.
- There are 3519 rows with value <=0 for df["total_onshift_partners"]. So, removing those rows.
- There are 576 rows with value <=0 for df["total_busy_partners"]. So, removing those rows.
- There are 260 rows with value <=0 for df["total_outstanding_orders"]. So, removing those rows
- There are 686 rows with df.min_item_price>df.max_item_price. So, dropping those rows

- **Feature engineering:** The target column time taken ("delivery time in minutes") in each delivery from the order timestamp (created_at) and delivery timestamp (actual_delivery_time) was created by subtracting the order timestamp from the delivery timestamp and converted to datetime format for easily working with them.
- . Two new columns "hour of the day", and "day of the week" were also added from the order time column "created_at". Two columns "created_at"', and "actual_delivery_time" are removed.

8. Encoding categorical columns

- Two non-numerical columns "'store_primary_category", and "store_id" are encoded using label encoding for further analysis.

9. Data visualization and cleaning

- Visualize various columns for a better understanding of Count and scatter plots
- The count plot of delivery hours is displayed in Figure 4.2. The greatest number of orders is during the overnight of which the peak is at 2 am (18.72%) followed by 1 am (14.63%) and 3 am (13.46%) and the least number of orders are during the afternoon time (~0%).
- The count plot of the delivery week is displayed in Figure 4.3. About 49% of orders are on weekends (Friday, Saturday, and Sundays).
- The count of market_id is shown in Figure 4.4. The 72% of orders are from market_id 1, 2, and 4.

- The count of order protocol is shown in Figure 4.5. 76% of orders are from order protocols 1, 3, and 5. Maximum orders were booked through the app (1.0) or re-booked (3.0).

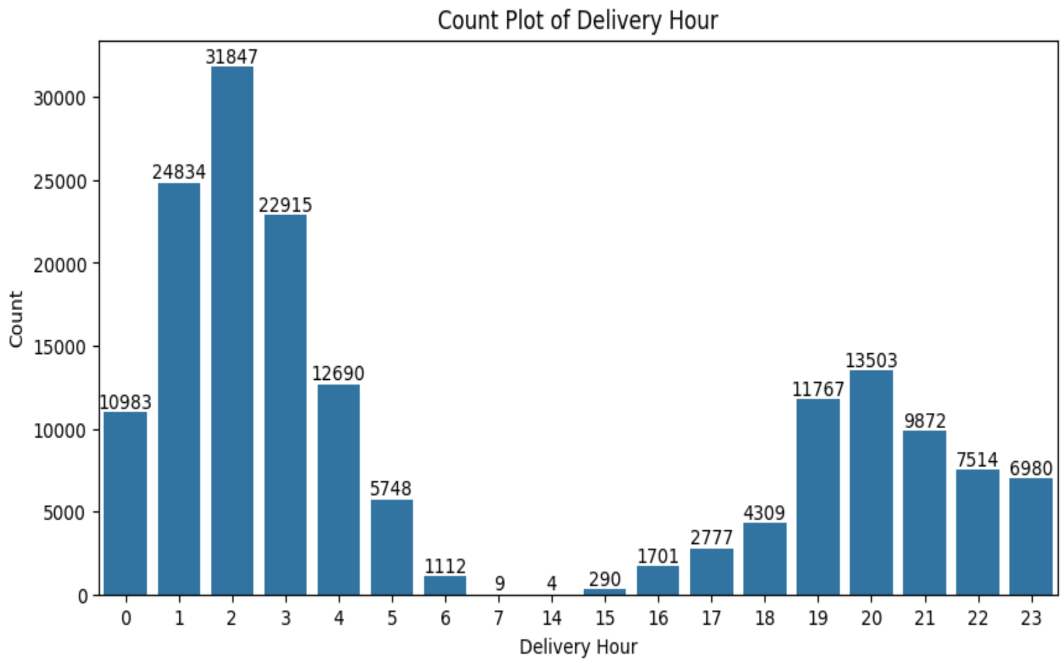


Figure 4. 13: Count plot of delivery hours

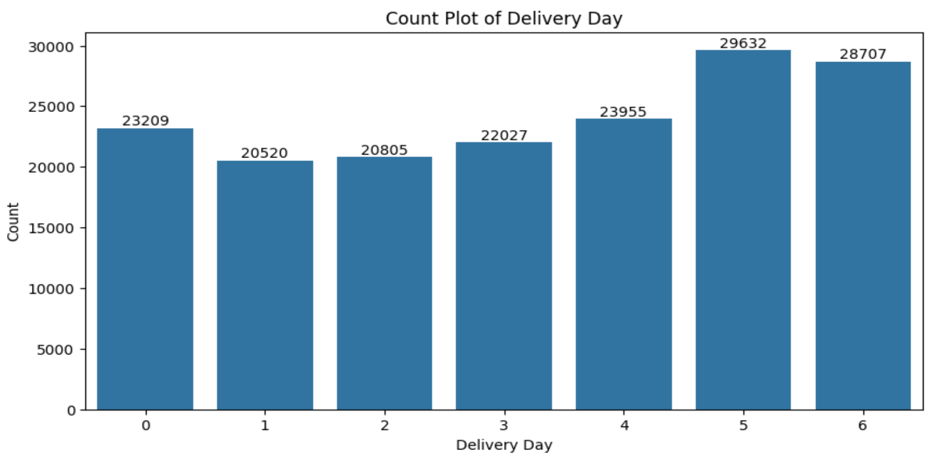


Figure 4. 14: Count Plot of Delivery Day

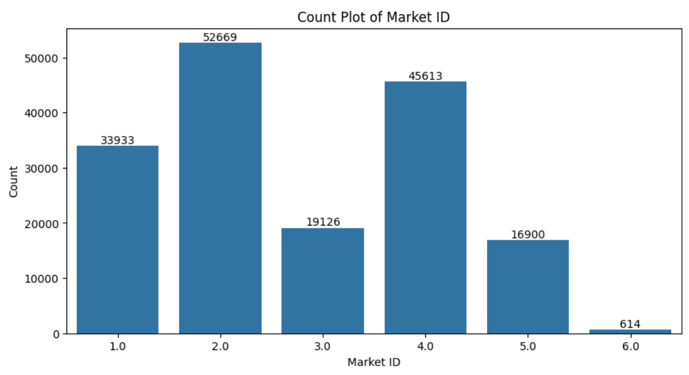


Figure 4. 15: Count plot of market_id

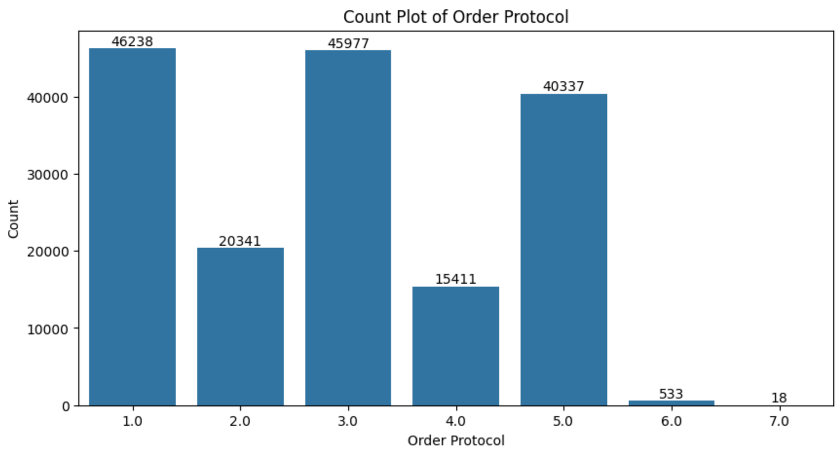


Figure 4. 16: Count plot of order protocol

- The heatmap shows the correlation between different features in the dataset is illustrated in Figure 4.6.
- Some key observations from the heatmap or correlation matrix are:
    - 'time_taken' has a moderate positive correlation with 'total_outstanding_orders'. This suggests that as the number of outstanding orders increases, the delivery time also increases.
    - 'time_taken' has a moderate negative correlation with 'total_onshift_partners'. This suggests that the delivery time tends to decrease as the number of delivery partners on shift increases.
    - 'total_busy_partners' and 'total_outstanding_orders' have a strong positive correlation. This is expected, as more outstanding orders would lead to more busy partners.
    - 'total_items' and 'subtotal' have a strong positive correlation, which is expected as more items would lead to a higher total price.

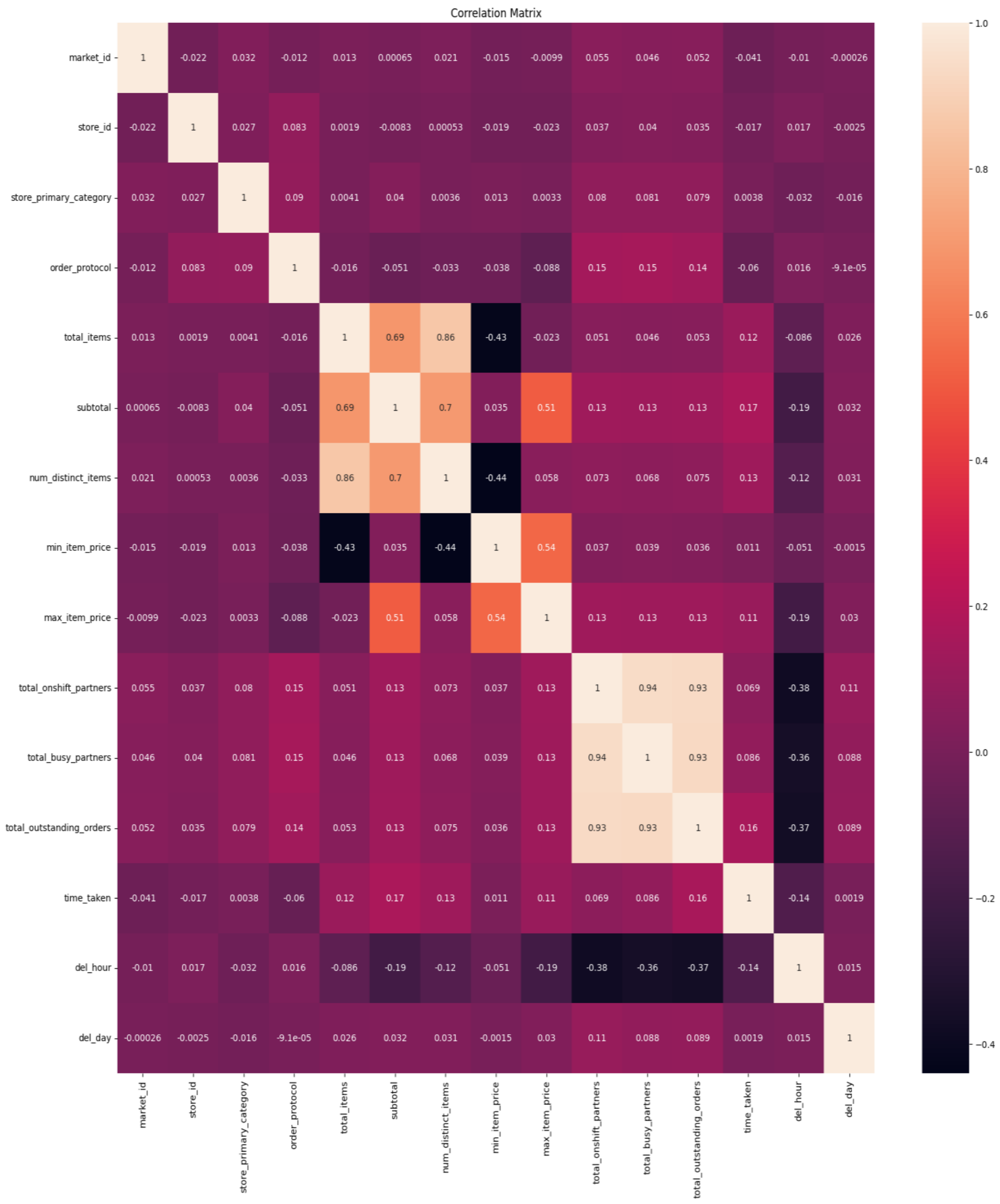
    - 'num_distinct_items' and 'total_items' have a moderate positive correlation. This suggests that as the number of distinct items increases, the total number of items also tends to increase.
    - 'max_item_price' and 'min_item_price' have a moderate positive correlation. This suggests that as the price of the costliest item increases, the price of the cheapest item also tends to increase.

10. Check if the data contains outliers

- The boxplots (illustrated in Figure 4.7) of the dataset were plotted to identify the outliers.
- The outliers were removed from numerical columns in the data frame by calculating the interquartile range (IQR) for each column and filtering out values that fall below the lower bound or above the upper bound, defined as 1.5 times the IQR from the first and third quartiles.
- The Boxplots of the dataset after removing the outliers are shown in Figure 4.8.

11. Split the data in train and test

- The data set was split into a training set (80%) and a testing set (20%).



	market_id	store_id	store_primary_category	order_protocol	total_items	subtotal	num_distinct_items	min_item_price	max_item_price	total_onshift_partners	total_busy_partners	total_outstanding_orders	time_taken	del_hour	del_day
market_id	1	-0.022	0.032	-0.012	0.013	0.00065	0.021	-0.015	-0.0099	0.055	0.046	0.052	-0.041	-0.01	-0.00026
store_id	-0.022	1	0.027	0.083	0.0019	-0.0083	0.00053	-0.019	-0.023	0.037	0.04	0.035	-0.017	0.017	-0.0025
store_primary_category	0.032	0.027	1	0.09	0.0041	0.04	0.0036	0.013	0.0033	0.08	0.081	0.079	0.0038	-0.032	-0.016
order_protocol	-0.012	0.083	0.09	1	-0.016	-0.051	-0.033	-0.038	-0.088	0.15	0.15	0.14	-0.06	0.016	-9.1e-05
total_items	0.013	0.0019	0.0041	-0.016	1	0.69	0.86	-0.43	-0.023	0.051	0.046	0.053	0.12	-0.086	0.026
subtotal	0.00065	-0.0083	0.04	-0.051	0.69	1	0.7	0.035	0.51	0.13	0.13	0.13	0.17	-0.19	0.032
num_distinct_items	0.021	0.00053	0.0036	-0.033	0.86	0.7	1	-0.44	0.058	0.073	0.068	0.075	0.13	-0.12	0.031
min_item_price	-0.015	-0.019	0.013	-0.038	-0.43	0.035	-0.44	1	0.54	0.037	0.039	0.036	0.011	-0.051	-0.0015
max_item_price	-0.0099	-0.023	0.0033	-0.088	-0.023	0.51	0.058	0.54	1	0.13	0.13	0.13	0.11	-0.19	0.03
total_onshift_partners	0.055	0.037	0.08	0.15	0.051	0.13	0.073	0.037	0.13	1	0.94	0.93	0.069	-0.38	0.11
total_busy_partners	0.046	0.04	0.081	0.15	0.046	0.13	0.068	0.039	0.13	0.94	1	0.93	0.086	-0.36	0.088
total_outstanding_orders	0.052	0.035	0.079	0.14	0.053	0.13	0.075	0.036	0.13	0.93	0.93	1	0.16	-0.37	0.089
time_taken	-0.041	-0.017	0.0038	-0.06	0.12	0.17	0.13	0.011	0.11	0.069	0.086	0.16	1	-0.14	0.0019
del_hour	-0.01	0.017	-0.032	0.016	-0.086	-0.19	-0.12	-0.051	-0.19	-0.38	-0.36	-0.37	-0.14	1	0.015
del_day	-0.00026	-0.0025	-0.016	-9.1e-05	0.026	0.032	0.031	-0.0015	0.03	0.11	0.088	0.089	0.0019	0.015	1

Figure 4. 17: Correlation matrix of the dataset.

- The feature importance of each feature was calculated using random forest and displayed in Figure 4.9. The MSE, RMSE, MAE, and R2 score of the random forest model is given below
  - MSE: 151.43209362604435
  - RMSE: 12.305774808034005
  - MAE: 9.767052793324499
  - R2_score: 0.22999711958573066

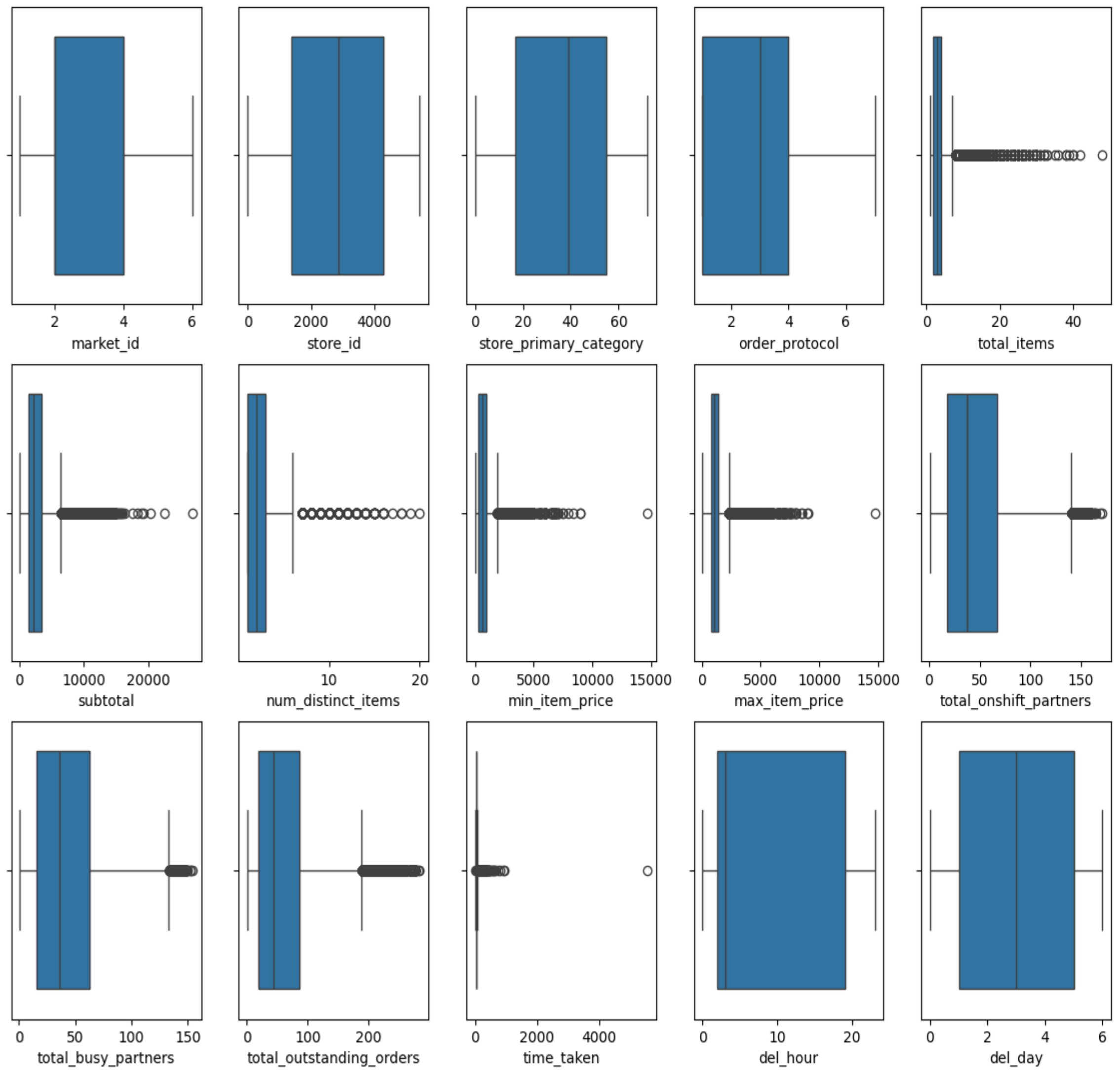


Figure 4. 18: Boxplot of the dataset before removing the outliers

12. Scaling the data for neural networks.

- MinMaxScaler from sklearn was used to scale the features in the dataset between 0 and 1. It then splits the scaled data into training and testing sets using an 80/20 split and a boxplot of the training set to check the distribution of each feature visualized in Figure 4.10.

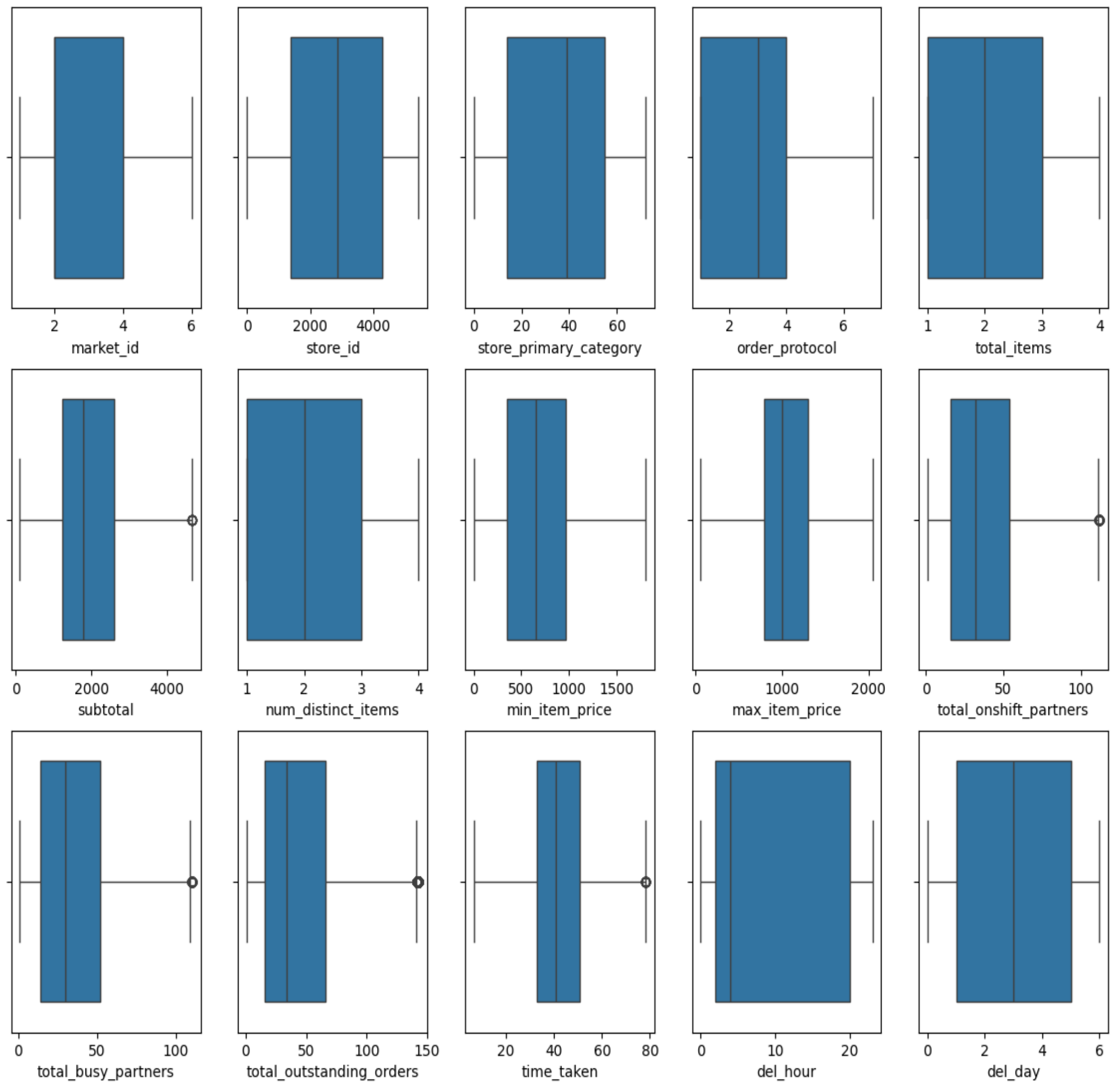


Figure 4. 19: Boxplot of the dataset after removing the outliers

13. Creating a simple neural network

- A simple neural network model using TensorFlow's Keras API consists of four fully connected layers (Dense), with 64, 516, 1024, and 256 neurons respectively, using the ReLU activation function for non-linearity. The output layer has one neuron with a linear activation function, suitable for a regression task where the goal is to predict a continuous value.

14. Training the neural network for the required amount of epochs

- The neural network model was compiled using the mean squared error ('mse') as the loss function, the Adam optimizer with a learning rate of 0.01, and tracks the metrics 'mse', 'mae' (mean absolute error), and 'accuracy'. The model is then trained for 30 epochs with a batch size of 512, using 80% of the training data and 20% as validation data, providing updates on the training process at each epoch (verbose=1).

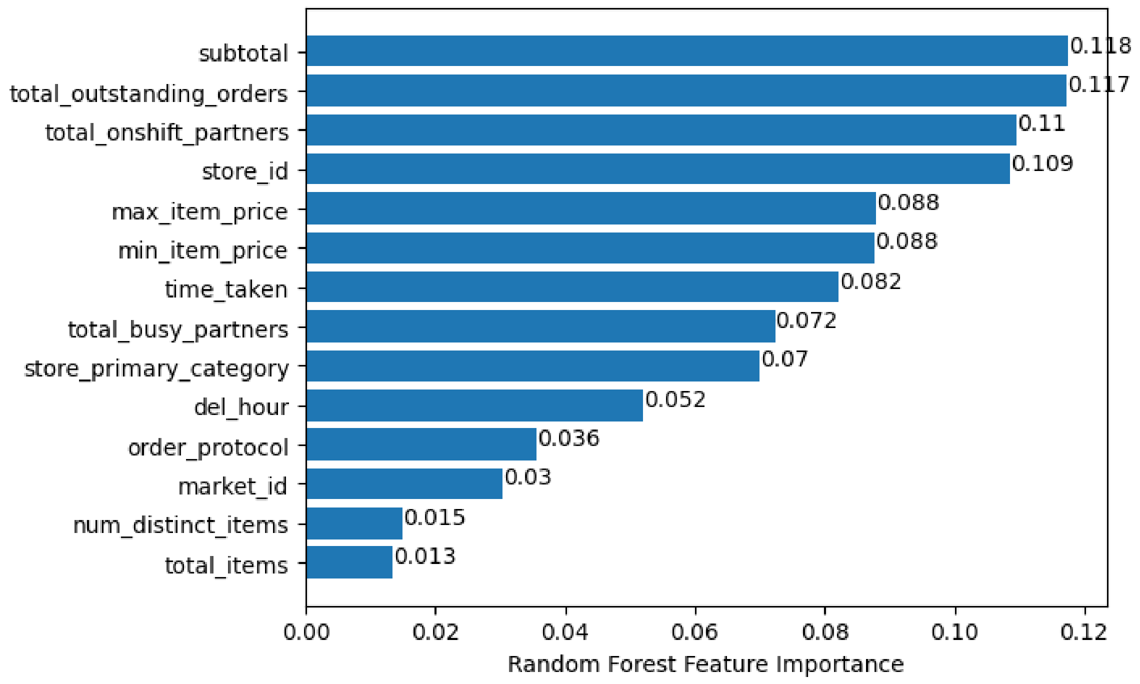


Figure 4. 20: Random Forest Feature Importance of the Dataset

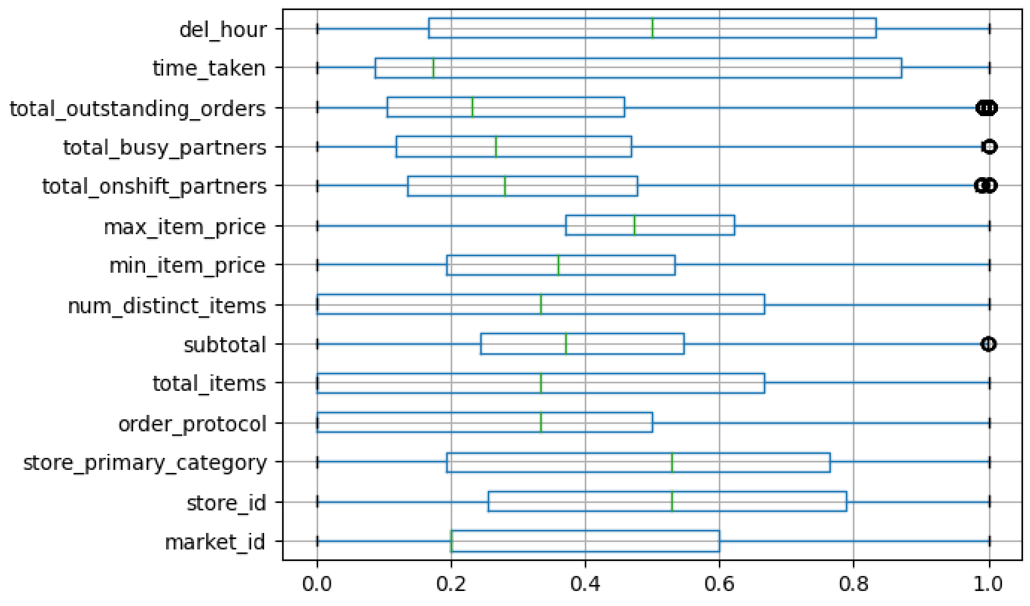


Figure 4. 21: Boxplot of the training set to check the distribution of each feature

15. Plotting the losses and checking the accuracy of the model

- The training and validation loss over the epochs are plotted in Figure 4.11 and training and validation accuracy over the epochs are plotted in Figure 4.12.

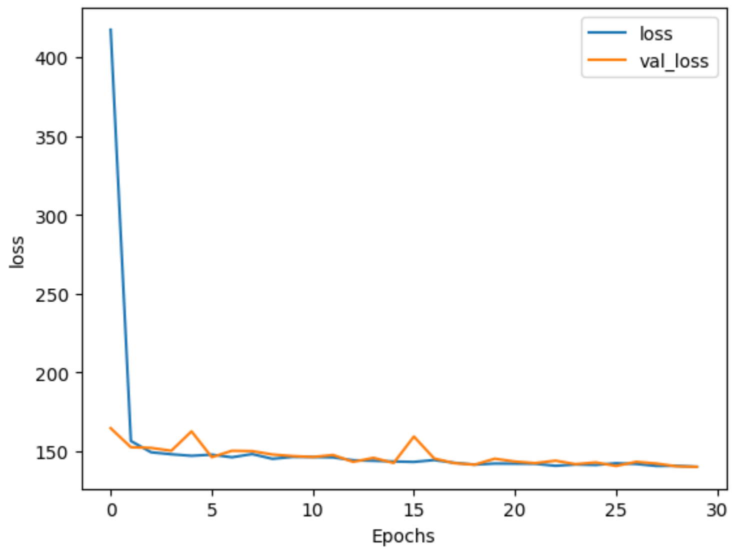


Figure 4. 22: Training and validation loss over the epochs

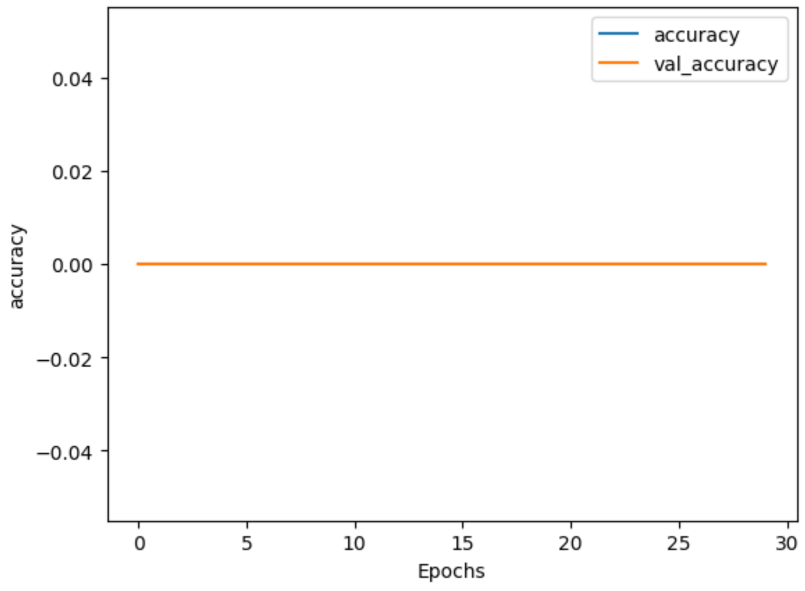


Figure 4. 23: Training and validation accuracy over the epochs

16. Checking its various metrics like MSE, RMSE, MAE

- The MSE, RMSE, MAE, and MAPE of the given neural network model is given below
  - MSE : 140.94032127317803
  - RMSE : 11.871828893358344
  - MAE : 9.456927251564325
  - MAPE : 0.24514081968403026
- Model was trained and tested using different configurations such as l1/l2 regularization, Dropout, and Batch Normalization.

Business Questions to be answered from Analysis

2. Defining the problem statements and where can this and modifications of this be used?

   - The goal is to estimate delivery times for food orders based on various factors such as the restaurant, order details, and the availability of delivery partners. The dataset provided contains information on multiple deliveries, with each row representing a unique delivery. Key features include market ID, order time, actual delivery time, restaurant category, order details, and delivery partner availability.

8. List 3 functions the pandas datetime provides with one line explanation.

   - **to_datetime:**
     - Converts the input to datetime.
     - Example: pd.to_datetime(1490195805, unit='s') : Timestamp('2017-03-22 15:16:45')
   - **dt (DatetimeAccessor):**
     - Accessor object for datetime like properties of the Series values.
     - Example: df['timestamp_column'].dt.day
   - **date_range:**
     - Returns a fixed-frequency datetime index, with day (calendar) as the default frequency.
     - Example: pd.date_range(start='2022-01-01', end='2022-01-10', freq='D')

9. Short note on datetime, timedelta, time span (period)

   - **Datetime:** In Python, the datetime module provides classes for working with dates and times. The datetime class is a part of this module and represents a point in time with both date and time components. Example:
     from datetime import datetime

current_time = datetime.now()

print(current_time) : prints the current time in format yyyy-mm-dd hh:mm:ss

- **Timedelta:** The timedelta class, also part of the datetime module, represents the duration between two dates or times. It allows you to perform arithmetic with datetime objects. Example:
  from datetime import datetime, timedelta
  yesterday = datetime.now() - timedelta(days=1)
  print(yesterday) prints the current time of day before in format yyyy-mm-dd hh:mm:ss
- Time Span (Period): In pandas, a time span or period is represented by the Period class. It's a fixed-frequency interval of time, such as a day, a month, or a year. It's useful for time-based indexing and other time-related operations. Example:

import pandas as pd

period = pd.Period('2022-03', freq='M')

print(period)

10. Why do we need to check for outliers in our data?

- Checking for outliers in data is crucial for several reasons in data analysis and modeling:
    a. **Impact on Descriptive Statistics:** Outliers can significantly affect basic descriptive statistics, such as the mean and standard deviation. The mean is sensitive to extreme values, and the presence of outliers can distort its value, leading to a misrepresentation of the central tendency.
    b. **Model Performance:** Outliers can adversely impact the performance of statistical and machine learning models. Algorithms like linear regression are particularly sensitive to outliers as they may influence the slope and intercept of the regression line, leading to inaccurate predictions.
    c. **Assumption Violations:** Many statistical techniques assume that the data is normally distributed or follows a certain pattern. Outliers may

violate these assumptions and lead to biased parameter estimates, affecting the validity of statistical inferences.

d. **Data Quality and Integrity:** Outliers might indicate errors in data collection, data entry, or measurement. Identifying and addressing outliers is essential for ensuring the quality and integrity of the dataset, preventing misleading or incorrect conclusions.

e. **Robustness of Models:** Some machine learning algorithms are less robust in the presence of outliers. For example, clustering algorithms or distance-based methods may assign undue importance to outliers, affecting the formation of clusters or the calculation of distances.

f. **Data Exploration and Understanding:** Outliers can provide insights into interesting phenomena or anomalies in the data. Investigating outliers can lead to a better understanding of the underlying processes or uncover important information that may require special attention or further investigation.

g. **Visualizations and Interpretability:** Outliers can distort visualizations, making it challenging to interpret and draw meaningful insights from graphs or charts. Addressing outliers improves the clarity and interpretability of visual representations of the data.

h. **Model Generalization:** When building predictive models, the goal is often to create a model that generalizes well to new, unseen data. Outliers in the training data may not be representative of the broader population, potentially leading to poor generalization.

11. Name 3 outlier removal methods?

- Z-Score Method
- Interquartile Range (IQR) Method
- Isolation Forest

12. What classical machine learning methods can we use for this problem?

- Random Forest Regression

- Gradient Boosting Regression (e.g., XGBoost)
- Linear Regression

13. Why is scaling required for neural networks?

- Faster and More Stable Convergence
- Improved Performance

9. Briefly explain your choice of optimizer.

- Adam, which stands for Adaptive Moment Estimation, is a popular optimizer used in neural network training to adjust weights and minimize loss. It combines the strengths of two other optimizers, AdaGrad and RMSprop, addressing their limitations and offering several advantages:
    - Key Features of Adam:
        1. Adaptive Learning Rates
        2. Handles Sparse Gradients
        3. Fast Convergence
        4. Less Sensitive to Learning Rate Choice
        5. Bias Correction

11. Which activation function did you use and why?

- **Sigmoid Function:** This S-shaped function maps input values between 0 and 1. Pros: Often the first activation function encountered due to its simplicity. Outputs are easily interpretable as probabilities (between 0 and 1). Cons: Suffers from vanishing gradients for large negative inputs. Not zero-centered, which can cause issues in deeper networks.
- **ReLU (Rectified Linear Unit):** The most widely used activation function in recent times. Outputs the input directly if it's positive, otherwise outputs zero. Pros: Computationally efficient, avoids vanishing gradients (gradient is simply 1 for positive inputs). Cons: Can suffer from the "dying ReLU" problem where neurons get stuck at zero if they receive negative inputs during training.

- **Softmax Function:**Typically used in the output layer of a network for multi-class classification problems. Outputs a vector of probabilities where the elements sum to 1. Each element represents the probability of the input belonging to a specific class.

12. Why does a neural network perform well on a large dataset?
    - With a large dataset, the network has more information (data points) to learn from, allowing it to capture the intricate relationships and patterns within the data. This leads to a more accurate representation of the underlying function or relationship between features and target variables.
    - Reduced Variance and Improved Generalization
    - Learning Complex Relationships
    - A larger dataset provides more diverse gradient updates, guiding the optimization process towards a better minimum loss value. This can lead to faster convergence and potentially better performance.

Actionable Insights

- The six unique values of market_Id are 1, 2, 3, 4, 5, and 6. The 72% of orders are from market_id 1, 2, and 4.
- Out of 74 store primary categories, 39% of orders are of store primary categories of American, pizza, Mexican, burger, and sandwich.
- There are seven order protocols namely 1, 2, 3, 4, 5, 6, and 7 and 76% of orders are from order protocols 1, 3, and 5. Maximum orders were booked through the app (1.0) or re-booked (3.0).
- 81.11 % of orders are of 1,2,3 or 4 items order and 57 unique and up to 66 items were included in a single order.
- 88.14 % order of 1,2,3 or 4 distinct items order out of 20 distinct items
- 76.51 % order of 1 -3 items order
- Dropped null or missing values.
- Subtotal is the total price of the order. If it is 0 or less than 0 then it means there is some error in the data. So, 16 rows with subtotal == 0 have been dropped.
- There are 2190 rows with min_item_price <=0. So, dropping those rows.

- There are 3519 rows with value <=0 for df["total_onshift_partners"]. So, removing those rows.
- There are 576 rows with value <=0 for df["total_busy_partners"]. So, removing those rows.
- There are 260 rows with value <=0 for df["total_outstanding_orders"]. So, removing those rows
- There are 686 rows with df.min_item_price>df.max_item_price. So, dropping those rows
- Created the target column “time_taken” (time taken in minutes for delivery) from order timestamp and delivery timestamp and deleted the columns df['actual_delivery_time'] and df['created_at'].
- Two new columns labeled “del_hour” and “del_day” were created from df['created_at'] columns which represent the delivery order created hour and day.
- Encoded 'store_primary_category' and ‘store_id’ using label encoding technique
- The greatest number of orders is during the overnight of which the peak is at 2 am (18.72%) followed by 1 am (14.63%) and 3 am (13.46%) and the least number of orders are during the afternoon time (~0%)
- About 49% of orders are on weekends (Friday, Saturday, and Sundays)
- 'time_taken' has a moderate positive correlation with 'total_outstanding_orders'. This suggests that as the number of outstanding orders increases, the delivery time also increases.
- 'time_taken' has a moderate negative correlation with 'total_onshift_partners'. This suggests that the delivery time tends to decrease as the number of delivery partners on shift increases.
- 'total_busy_partners' and 'total_outstanding_orders' have a strong positive correlation. This is expected, as more outstanding orders would lead to more busy partners.
- 'total_items' and 'subtotal' have a strong positive correlation, which is expected as more items would lead to a higher total price.
- 'num_distinct_items' and 'total_items' have a moderate positive correlation. This suggests that as the number of distinct items increases, the total number of items also tends to increase.

- 'max_item_price' and 'min_item_price' have a moderate positive correlation. This suggests that as the price of the costliest item increases, the price of the cheapest item also tends to increase.

Recommendations

- Further investigate the relationships between features and delivery time.
- Explore other machine learning algorithms (e.g., Gradient Boosting, Support Vector Regression).
- Fine-tune hyperparameters using techniques like grid search or Bayesian optimization.
- Explore more advanced deep learning architectures (e.g., Recurrent Neural Networks, Convolutional Neural Networks).
- Investigate the impact of different scaling methods on model performance.
- Consider using cross-validation to obtain a more robust model evaluation.
- Deploy the model to a production environment for real-time predictions.
- Monitor the model's performance over time and retrain it as needed.