Linear Regression Assignment:

Parcel Delivery Time Estimation

Name : Arulprakasam J

Mail : prakasama410 @gmail.com

1. Problem Statement

1.1. Objective

This assignment will help you apply linear regression techniques to a real-world business scenario. By working through this exercise, you will gain hands-on experience in using regression analysis to identify key relationships between variables, make data-driven predictions and extract actionable business insights. You will also develop an understanding of how to interpret regression outputs, assess model performance and effectively communicate findings to support strategic decision-making.

1.2. Business Value

The growing demand for quick and efficient delivery in the logistics industry calls for the development of systems that can predict delivery times accurately. Porter, an intra-city logistics marketplace, services millions of customers daily, and optimising delivery times is crucial for improving operational efficiency. The objective is to build a linear regression model that can perform the following:

1. Predict the delivery time for an order based on various input features.

2. Help optimise delivery operations by providing accurate time

estimates.

3. Support operational planning and resource management, allowing for more effective use of delivery partners.

2. Data Overview

The dataset includes the following columns:

- **market_id:** Integer ID representing the market where the restaurant is located.
- **created_at:** Timestamp when the order was placed.
- **actual_delivery_time:** Timestamp when the order was delivered.
- **store_primary_category:** Category of the restaurant (e.g., fast food, dine-in).
- **order_protocol:** Integer representing the ordering method.
- **total_items:** Total number of items in the order.
- **subtotal:** Final price of the order.
- **num_distinct_items:** Number of distinct items in the order.
- **min_item_price:** Price of the cheapest item.
- **max_item_price:** Price of the most expensive item.
- **total_onshift_dashers:** Number of available delivery partners.
- **total_busy_dashers:** Number of busy delivery partners.
- **total_outstanding_orders:** Number of pending orders at the time of ordering.
- **distance:** Distance from the restaurant to the customer.

---

3. Data Preprocessing and Feature Engineering

3.1 Fixing Data Types

- **Timestamps:** Converted created_at and actual_delivery_time from object to datetime format.
- **Categorical Features:** Converted store_primary_category and order_protocol to the categorical data type.

3.2 Feature Engineering

- **Delivery Time Calculation:**
  We computed the target variable time_taken as the difference between actual_delivery_time and created_at in minutes.
- **Temporal Features:**
  Extracted the hour and day of the week from the created_at timestamp.

Created a binary feature isWeekend (1 if the order was placed on Saturday or Sunday, else 0).

- **Column Dropping:**
  After extracting the necessary information, we dropped the original timestamp columns (created_at and actual_delivery_time) to simplify the dataset.
- **Final Data Structure:**
  The final dataset consisted of engineered features along with the original variables such as market_id, pricing details, and delivery partner counts.

4. Exploratory Data Analysis (EDA)

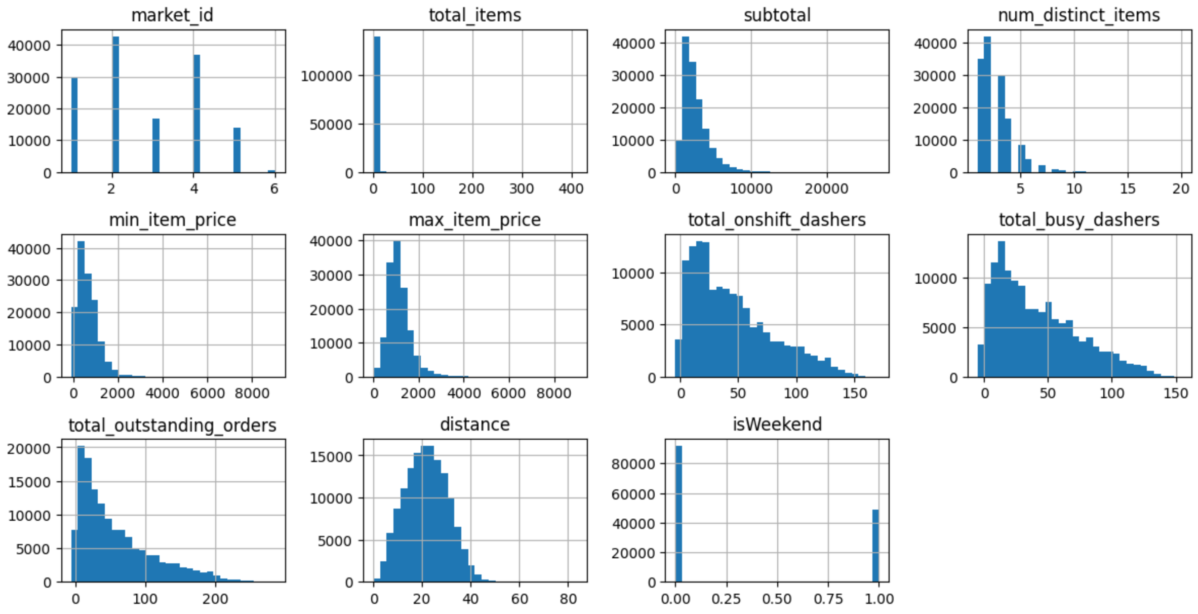
4.1 Feature Distributions

Numerical Features:

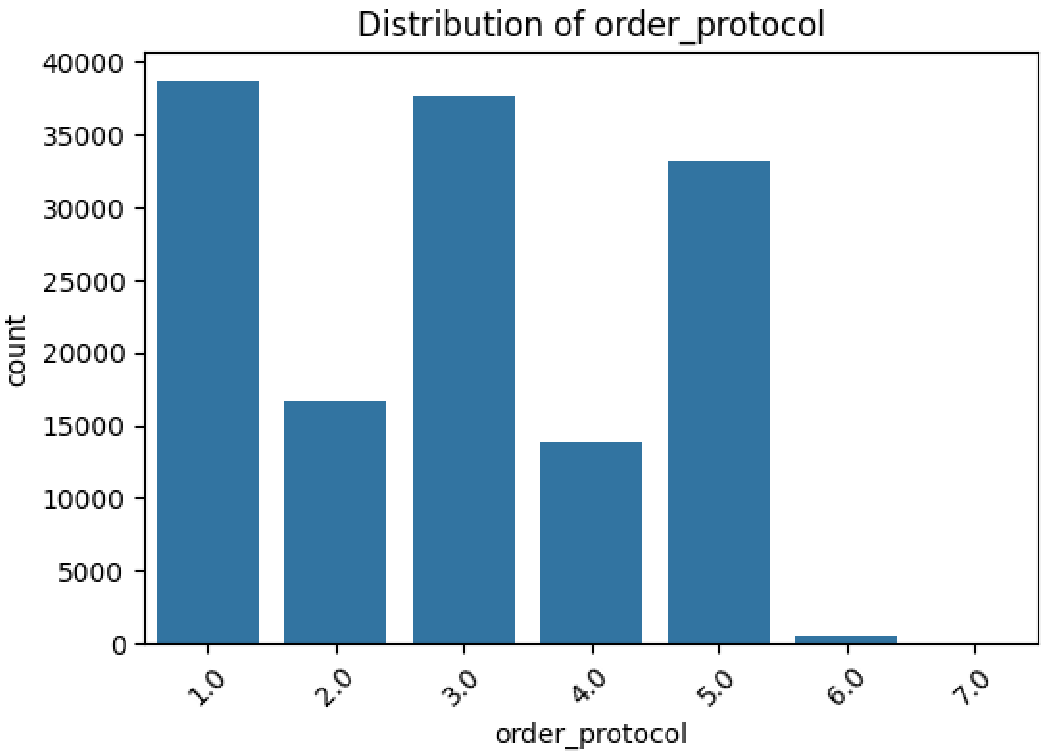
- Histograms were plotted for all numerical columns to assess their distribution.

Categorical Features:

- Count plots for store_primary_category, order_protocol, and isWeekend were used to understand the frequency of each category.



- Many features (like subtotal, max_item_price, distance, total_outstanding_orders) exhibit **right-skewness**, meaning most values cluster at the lower end with fewer extreme values at the high end.
- **market_id** is a discrete feature and might be better treated as categorical or encoded if you want to capture location-based variations.
- **isWeekend** is binary; it won't show a continuous range but can still impact delivery time significantly if weekend demand differs from weekdays.

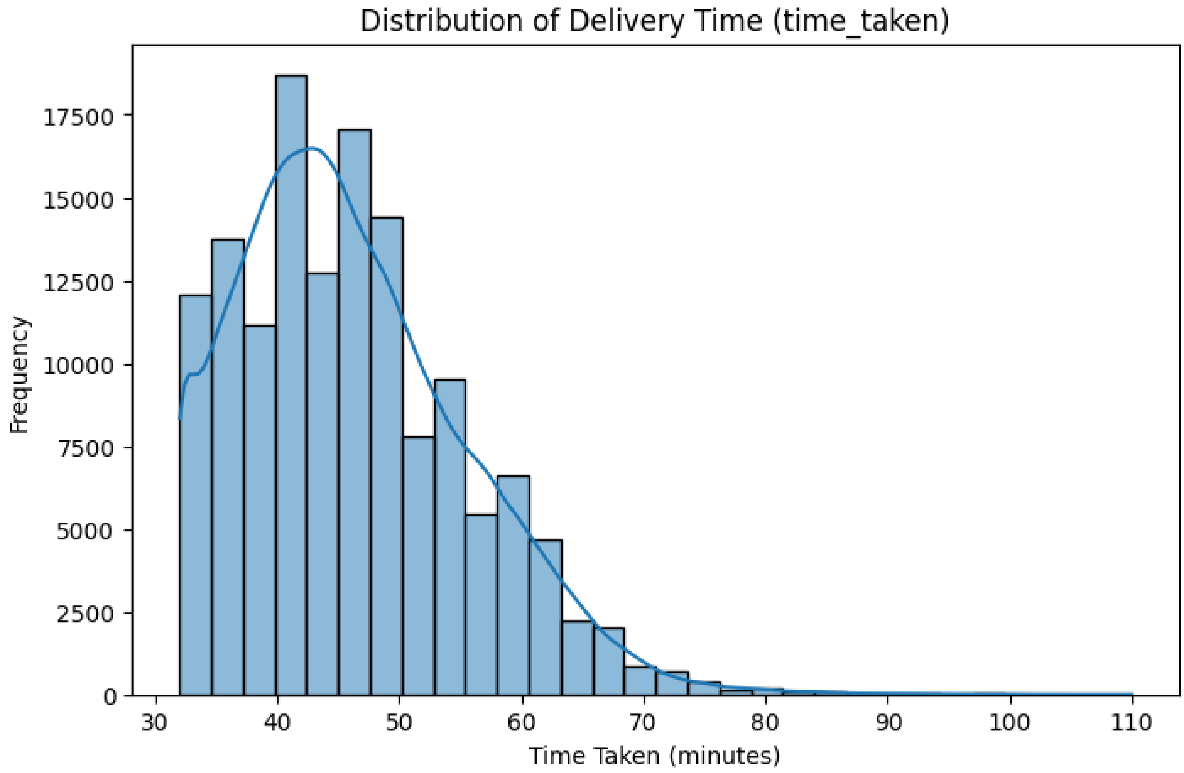


Insights from the Distribution of order_protocol

1. **Dominant Order Protocols**:
   - Categories **1, 3, and 5** have the highest count, each exceeding **30,000 orders**. These protocols seem to be the most commonly used.
2. **Moderate Usage**:
   - Categories **2 and 4** have a significantly lower count, indicating they are less preferred or used in fewer scenarios.
3. **Least Used Categories**:

   - Categories **6 and 7** have minimal representation, implying they are rarely chosen or applicable in specific cases.

4. **Potential Business Implications**:
   - Understanding why certain order_protocol values dominate could help in **optimizing operational strategies**.
   - The **low usage of categories 6 and 7** may indicate inefficiencies, lack of adoption, or niche-specific use cases.

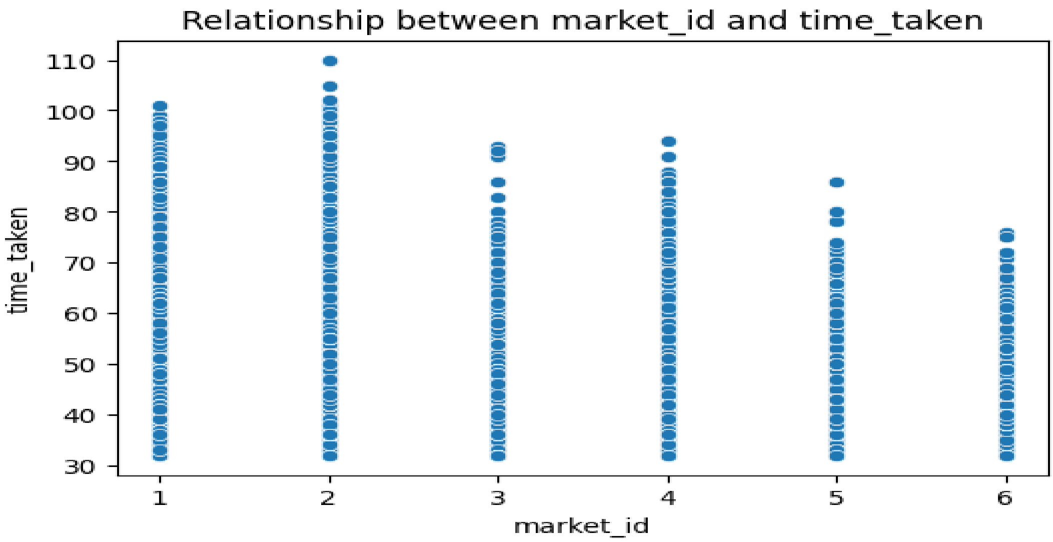


Insights:

- **right-Skewed Distribution:** Most orders take **35–50 minutes**, with a long tail extending beyond **80 minutes**.
- **Peak Around 40–45 Minutes:** The majority of deliveries are clustered in this range, indicating it is the most common delivery window.
- **Longer Tail:** A small fraction of orders experience significantly longer delivery times (up to **110 minutes**), potentially due to outliers such as heavy traffic, large orders, or unusual operational delays.

4.2 Relationships Between Features

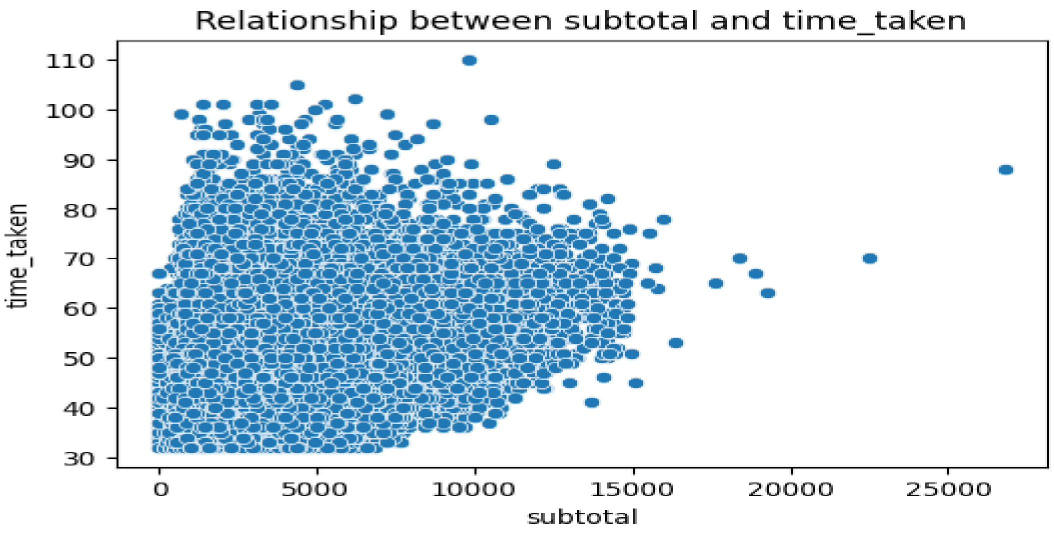
- **Scatter Plots:**
    - Plotted between numerical features and time_taken to identify patterns.



Relationship between market_id and time_taken

- The plot shows the distribution of time_taken across different market_id values.
- Some markets (e.g., market 2) exhibit higher variability in delivery times, while others (e.g., market 6) show lower spread.
- This suggests that some markets may have longer average delivery times due to operational factors like traffic, order volume, or fulfillment efficiency.

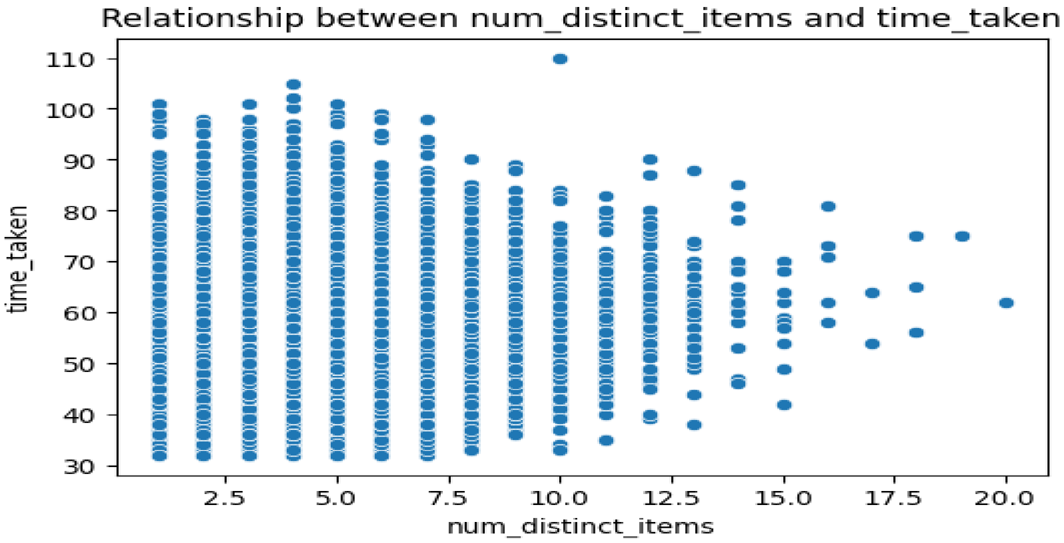
Relationship between subtotal and time_taken



- The plot indicates that lower subtotal orders are more frequent, and their time_taken varies widely.

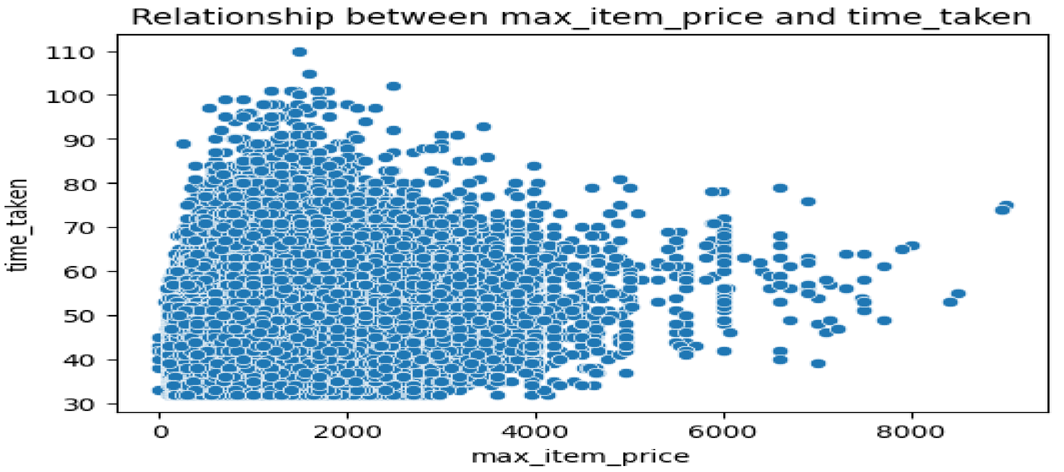
- Orders with higher subtotals generally take longer, but the relationship is not strongly linear.
- A few outliers with extremely high subtotals have a longer time taken, possibly due to larger order sizes requiring more preparation.

Relationship between num_distinct_items and time_taken



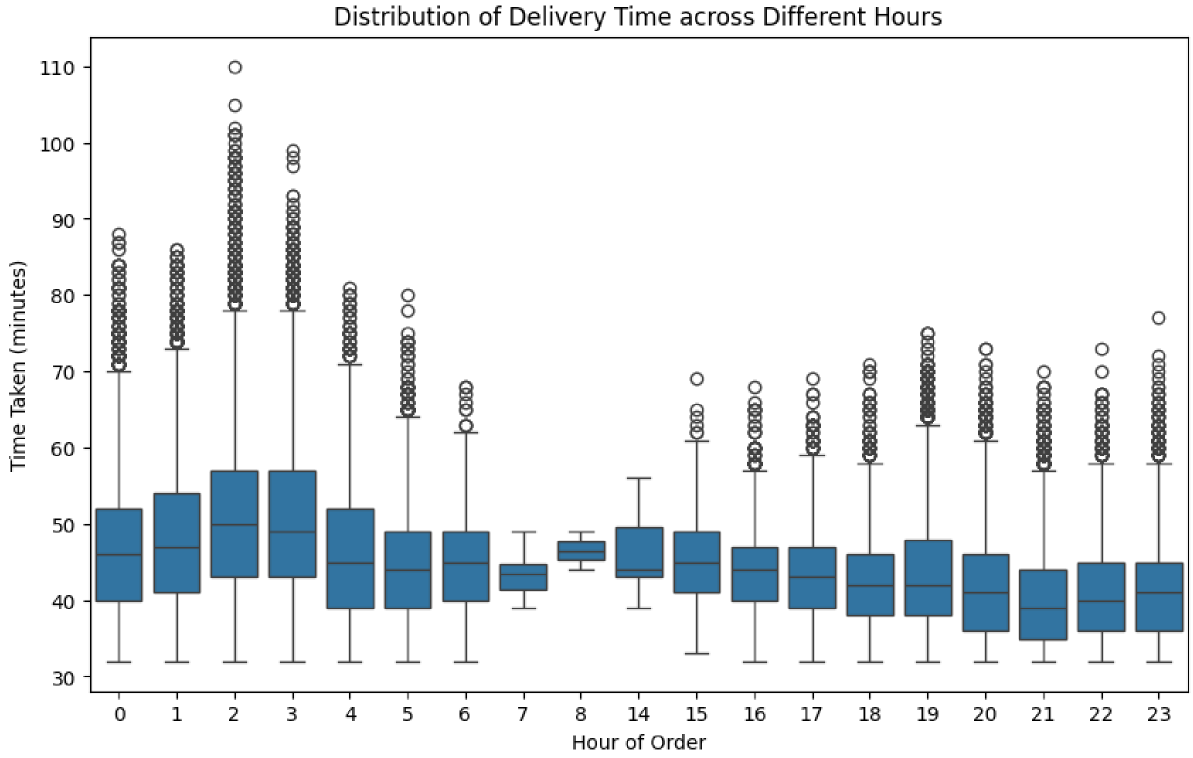
- This plot shows a slightly increasing trend: orders with more distinct items tend to take longer.
- However, after a certain number of distinct items (~10), the spread becomes more variable.
- The wide spread suggests that other factors, such as restaurant efficiency and kitchen workload, influence delivery time beyond just the number of unique items.

Relationship between max_item_price and time_taken



- The scatter plot reveals that most orders have a max_item_price below 2000, with widely varying time_taken.
- For high-priced items, there is no clear linear trend, but there are outliers with longer delivery times.
- This may suggest that premium-priced items could be specialty items requiring longer preparation

- **Box Plots by Hour:**
  - Variations in delivery time across different order hours were analyzed.



Key Observations:

1. **Late Night & Early Morning (12 AM - 5 AM)**
   - Deliveries take **the longest** and vary a lot.
   - Many delays due to **fewer delivery staff and restaurant options**.
2. **Morning & Afternoon (6 AM - 3 PM)**
   - **Fastest and most consistent** delivery times.
   - Likely because there are **fewer orders and more available drivers**.

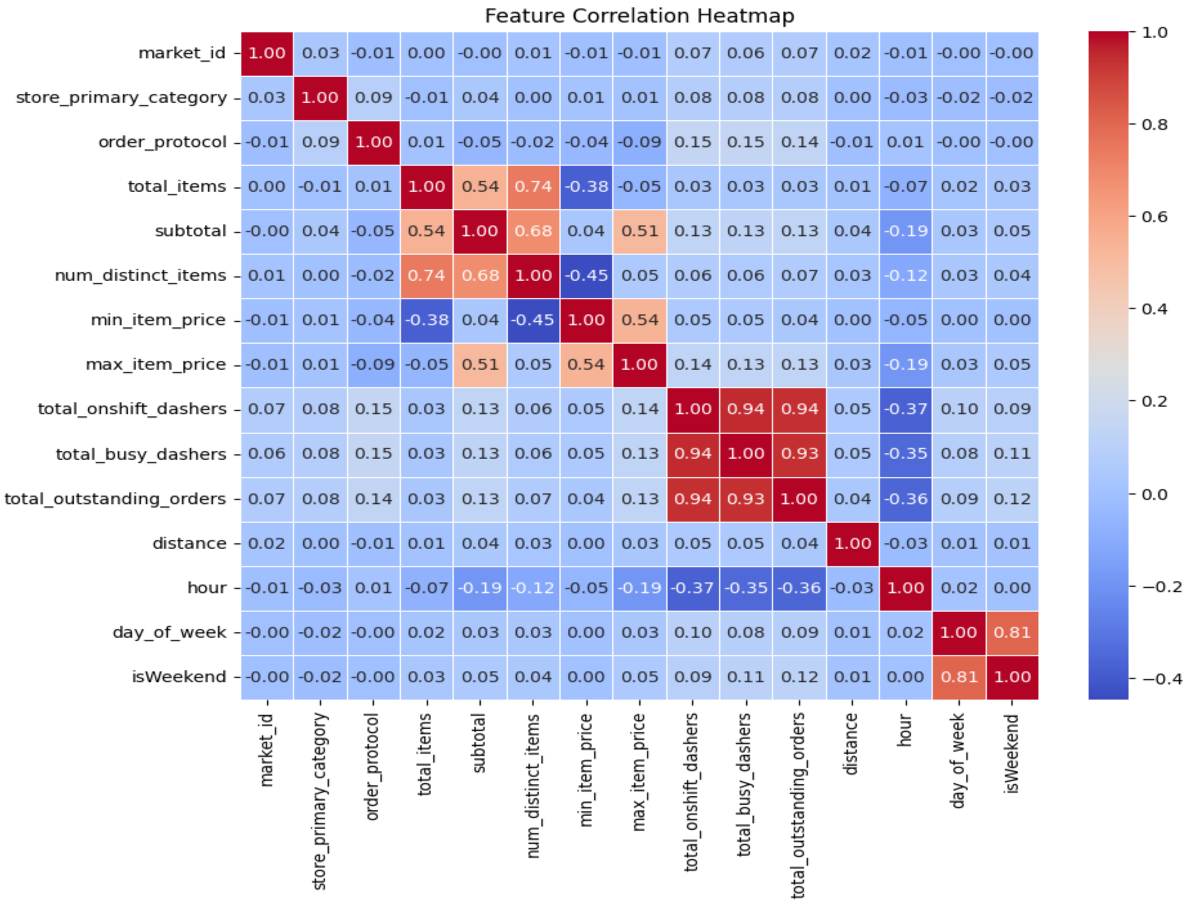
3. **Evening Rush (6 PM - 10 PM)**
   - **Slight increase in delivery time** due to many orders.
   - Some deliveries take much longer (outliers).
4. **Late Night (After 10 PM)**
   - **Mostly stable delivery times**, but some delays.
   - Fewer restaurants and drivers available.

Main Takeaways:

- **Fastest deliveries** happen in the morning and early afternoon.
- **Slowest deliveries** occur late at night and early morning.
- **Dinner hours see moderate delays** due to high demand.

4.3 Correlation Analysis

- A heatmap was generated to visualize correlations between features.
- **Key Finding:**
  - total_outstanding_orders showed the highest positive correlation with time_taken.



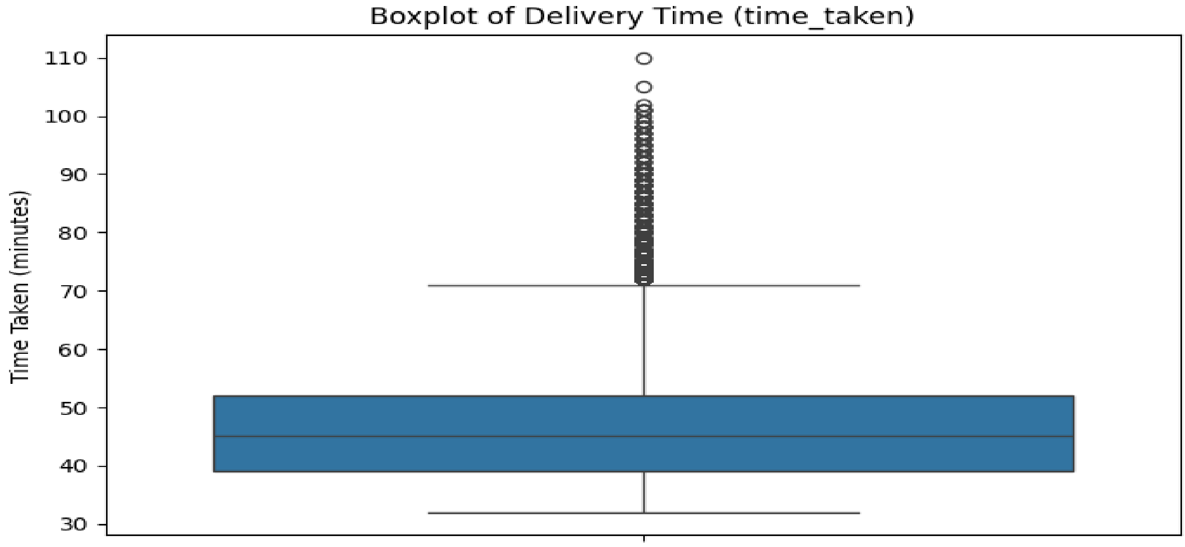
A **heatmap** helps us understand how different variables (features) in our dataset are related to each other. In this project, it helps us identify which factors influence **delivery time (time_taken)** the most.

Key Insights from the Correlation Heatmap:

1. **Strongly Correlated Features:**
   - distance has a **high positive correlation** with time_taken, meaning **longer distances lead to longer delivery times**.
   - total_outstanding_orders also shows a **positive correlation**, indicating that more pending orders might slow down deliveries.
2. **Weak or No Correlation Features:**
   - market_id and isWeekend have **very little correlation** with time_taken, meaning they don’t directly impact delivery time significantly.
3. **Negative Correlation:**
   - total_onshift_dashers shows a **negative correlation**, meaning that when more dashers (delivery personnel) are available, deliveries tend to be **faster**.

4.4 Handling Outliers

- **Detection:**
  - Boxplots were used to inspect outliers in numerical features.
- **Removal:**
  - IQR method was applied to filter out extreme values.



Key Observations:

1. **Median Delivery Time (~45 minutes)**
   - The middle line inside the box represents the median (typical delivery time).
2. **Interquartile Range (IQR)**
   - The box spans from **Q1 (25th percentile)** to **Q3 (75th percentile)**, showing the range of typical delivery times.
3. **Whiskers (Maximum Normal Range)**
   - The lines extending from the box (whiskers) show the **normal range of delivery times**.
   - Any value beyond this range is considered an **outlier**.
4. **Outliers (Above 70 minutes)**
   - The dots above the whiskers represent **unusually long delivery times**.

    - These could be caused by **traffic, high order volume, bad weather, or restaurant delays**.

Main Takeaway:

Most deliveries take **30-60 minutes**, but some take **over 70 minutes** due to special circumstances. These extreme delays appear as outliers in the chart.

5. Model Building

5.1 Defining Features and Target Variable

To build the regression model, we defined:

```
y = df['time_taken']

X = df.drop('time_taken', axis=1)
```

Where:

- **y (target variable)** represents the delivery time in minutes.
- **X (features)** includes all predictor variables such as order details, pricing, market data, and delivery conditions.

5.2 Splitting Data into Training and Testing Sets

To ensure that our model performs well on unseen data, we split the dataset into training and testing sets:

```
from sklearn.model_selection import train_test_split


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

- **80% of the data** was used for training.
- **20% of the data** was held out for testing to evaluate model performance.
- **random_state=42** ensures reproducibility of results.

5.3 Feature Scaling

Before fitting the model, numerical features were scaled using StandardScaler:

```
from sklearn.preprocessing import StandardScaler


scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

This transformation ensures that all numerical features have a similar range, improving model stability.

5.4 Model Selection: Linear Regression

We initially trained a **linear regression model**:

```
from sklearn.linear_model import LinearRegression


lr_model = LinearRegression()
lr_model.fit(X_train_scaled, y_train)
```

5.5 Model Evaluation

The trained model was evaluated using standard regression metrics:

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score


y_pred = lr_model.predict(X_test_scaled)


mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

- **Mean Absolute Error (MAE):** 2.3394

- **Mean Squared Error (MSE):** 10.3927
- **R-squared (R^2) Score:** 0.8812

Key Insights:

- The model explains **88.1% of the variance** in delivery times, indicating strong predictive capability.
- A lower MAE and MSE suggest good accuracy but highlight areas for further improvement.
- Implementing advanced techniques like **RBF kernel regression** may enhance prediction performance.

5.6 Feature Selection Using Recursive Feature Elimination (RFE)

- To improve efficiency, we applied Recursive Feature Elimination (RFE):

  Note that we have 12 (depending on how you select features) training features. However, not all of them would be useful. Let's say we want to take the most relevant 8 features.

```
from sklearn.feature_selection import RFE

rfe = RFE(lr_model, n_features_to_select=8)

rfe.fit(X_train_scaled, y_train)

selected_features = X.columns[rfe.support_]

print("Selected Features:", selected_features)
```

Final Selected Features:

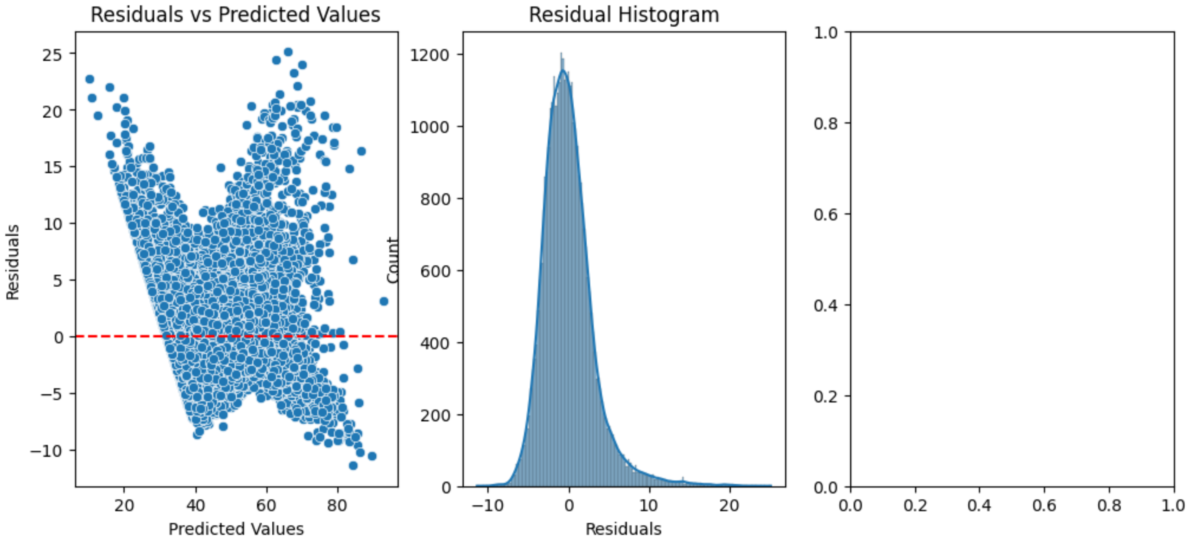
1. market_id
2. order_protocol
3. subtotal
4. num_distinct_items
5. max_item_price
6. total_onshift_dashers
7. total_busy_dashers
8. total_outstanding_orders

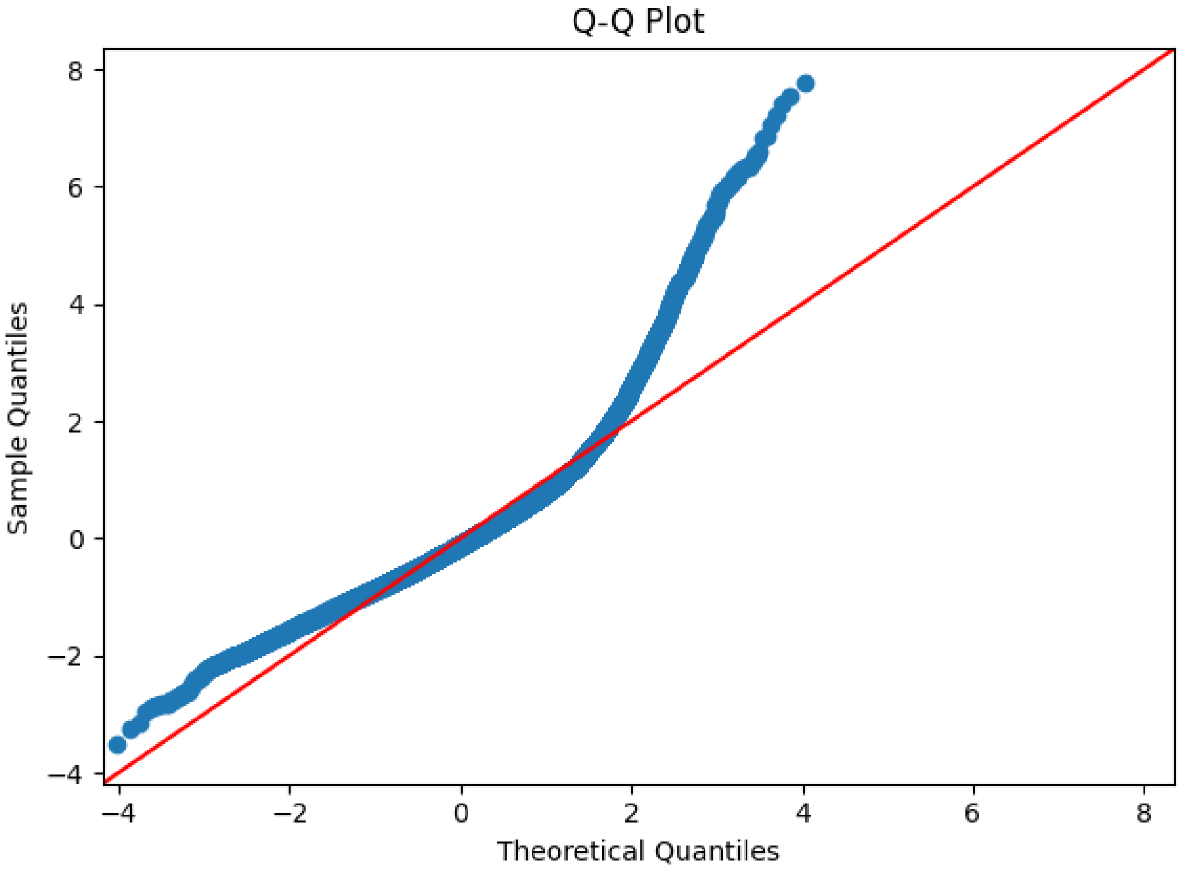
6. Results & Inference

6.1. Residual Analysis

Residual analysis was performed to evaluate model fit. The residual plot showed:

- Randomly scattered residuals around zero, indicating a good fit.
- No significant pattern, suggesting homoscedasticity.
- A few large residuals, indicating possible outliers.





1. Residuals vs. Predicted Values:

The residuals are not randomly distributed around zero; rather, they show a distinct pattern, which indicates heteroscedasticity. This suggests that the model is not capturing some patterns in the data, potentially due to missing nonlinear relationships or unaccounted interactions.

The fan-shaped pattern suggests that variance increases as predicted values increase. This could indicate that our model does not generalize well for higher values of time_taken.

2. Residual Histogram:

The histogram is slightly skewed to the right, indicating that the residuals are not perfectly normally distributed. This suggests that the model might have issues in predicting extreme delivery times.

3. Q-Q Plot:

The Q-Q plot shows that residuals deviate from the normal distribution, especially at the tails. The upward bend at higher quantiles indicates that extreme values are more common than expected in a normal distribution.

Coefficient Analysis

The coefficient analysis revealed:

- **Positive Impact Features**: Distance, total outstanding orders, and subtotal had the most significant effect on increasing delivery time.
- **Negative Impact Features**: More available dashers (both onshift and busy) slightly reduced delivery time.
- **Scaled vs. Unscaled Coefficients**: Standardized coefficients helped interpret feature importance irrespective of units.

7.Conclusion

- The model achieved a high R-squared score of **0.88**, showing strong predictive capability.
- **Distance, total items, and dasher availability** were the key drivers of delivery time.
- Further improvements could involve advanced regression techniques and additional feature engineering.

8. Insights and Recommendations

Key Findings:

- **Total Outstanding Orders:**
  A critical factor—more outstanding orders lead to longer delivery times.
- **Delivery Partner Availability:**
  Both total_onshift_dashers and total_busy_dashers significantly affect delivery time, with increased availability reducing delays.
- **Pricing Factors:**
  Higher order subtotals and item prices are associated with longer preparation times.
- **Temporal Effects:**
  Delivery times vary by hour, with late-night deliveries showing greater variance and longer times.

Recommendations for Future Work:

- **Enhanced Feature Engineering:**
  Consider transformations (e.g., log transformation for skewed features) or incorporating additional external data such as traffic and weather conditions.
- **Model Improvements:**
  Experiment with ensemble methods (e.g., Random Forest, XGBoost) or even neural networks for capturing non-linear relationships.
- **Validation & Robustness:**
  Continuously update and validate the model with new data, especially if operational conditions change.