REPORT DOCUMENTATION: PREDICTING DELIVERY EFFICIENCY USING HISTORICAL MARKET AND STORE DATA

BY: YASIN KHAN AND MAVIN RODRIGUES

INTRODUCTION

The goal of this project is to predict delivery times based on historical market and store data using machine learning techniques. The key objective is to improve the accuracy of delivery time predictions by leveraging important factors such as the number of items, distinct item count, labor availability, and market characteristics. To achieve this, we applied two models: ***Linear Regression and K-Nearest Neighbors (KNN)***. We evaluated their performance based on Mean Squared Error (MSE) and R-Squared (R^2) metrics. This project aims to optimize delivery operations and enhance customer satisfaction by improving delivery time predictions.

The dataset used includes historical delivery data with variables like *`total_items`, `distinct_items`, `onshift_dashers`, `market_id`, and `delivery_time`*. Proper preprocessing, model training, and evaluation were critical to understanding how well each model performs under the given conditions.

DATA PREPROCESSING

Before model training, preprocessing is essential to ensure data quality and proper formatting.

1. Handling Missing Data: Missing values were replaced with the median of the respective features to prevent bias from extreme values.

2. Categorical Conversion: Market IDs, a categorical variable, were converted into factors for compatibility with the models.

3. Data Splitting: The dataset was divided into a training set (70%) and a test set (30%) to evaluate model performance fairly.

Code for Preprocessing:

```
# Load required libraries

library(tidyverse)

library(caret)

# Load the dataset
```

```
data <- read.csv("historical_data.csv")

# Handle missing values

data <- data %>%

  mutate(across(everything(), ~ ifelse(is.na(.), median(., na.rm = TRUE), .)))

# Convert categorical variables to factors

data$market_id <- as.factor(data$market_id)

# Split into training and test sets

set.seed(123)

trainIndex <- createDataPartition(data$delivery_time, p = 0.7, list = FALSE)

train_data <- data[trainIndex, ]

test_data <- data[-trainIndex, ]
```

The preprocessing ensures that the models can handle both numeric and categorical variables, and the data split allows for performance evaluation on unseen data.

MODEL 1: LINEAR REGRESSION:

Why Linear Regression?

Linear regression is a simple and interpretable model ideal for identifying linear relationships between features like the number of items and labor availability and the target variable, `delivery_time`. It serves as a baseline model for comparison.

CODE FOR MODEL DEVELOPMENT:

```
# Fit a linear regression model

linear_model <- lm(delivery_time ~ total_items + distinct_items + onshift_dashers + market_id, data = train_data)

# Make predictions on the test data

test_data$predicted_lm <- predict(linear_model, test_data)
```

***CODE FOR MODEL EVALUATION*:**

```
# Calculate RMSE and R-squared

rmse_lm <- sqrt(mean((test_data$delivery_time - test_data$predicted_lm)^2))

r_squared_lm <- 1 - (sum((test_data$delivery_time - test_data$predicted_lm)^2) /
sum((test_data$delivery_time - mean(test_data$delivery_time))^2))

# Print performance metrics

cat("RMSE:", rmse_lm, "\nR-squared:", r_squared_lm)
```

RESULTS:

- MSE: 302.15
- R^2: 0.205

Linear regression provides a simple model for predicting delivery times, with moderate performance as indicated by the R^2 value, showing limited explanatory power.

MODEL 2: K-NEAREST NEIGHBORS (KNN)

Why KNN?

KNN predicts delivery time by averaging the delivery times of the k-nearest neighbors. It can capture complex, non-linear relationships between the input features and delivery times, making it a useful contrast to linear regression.

CODE FOR MODEL DEVELOPMENT:

```
library(class)

# Normalize the data

preProcValues <- preProcess(train_data[, c('total_items', 'distinct_items', 'onshift_dashers',
'market_id')], method = c("center", "scale"))

train_data_scaled <- predict(preProcValues, train_data)

test_data_scaled <- predict(preProcValues, test_data)
```

```
# Fit KNN model

k <- 5

knn_pred <- knn(train = train_data_scaled[, c('total_items', 'distinct_items', 'onshift_dashers', 'market_id')],

        test = test_data_scaled[, c('total_items', 'distinct_items', 'onshift_dashers', 'market_id')],

        cl = train_data$delivery_time, k = k)

test_data$predicted_knn <- as.numeric(knn_pred)
```

CODE FOR MODEL EVALUATION:

```
# Calculate RMSE and R-squared

rmse_knn <- sqrt(mean((test_data$delivery_time - test_data$predicted_knn)^2))

r_squared_knn <- 1 - (sum((test_data$delivery_time - test_data$predicted_knn)^2) / sum((test_data$delivery_time - mean(test_data$delivery_time))^2))

# Print performance metrics

cat("RMSE:", rmse_knn, "\nR-squared:", r_squared_knn)
```

RESULTS:

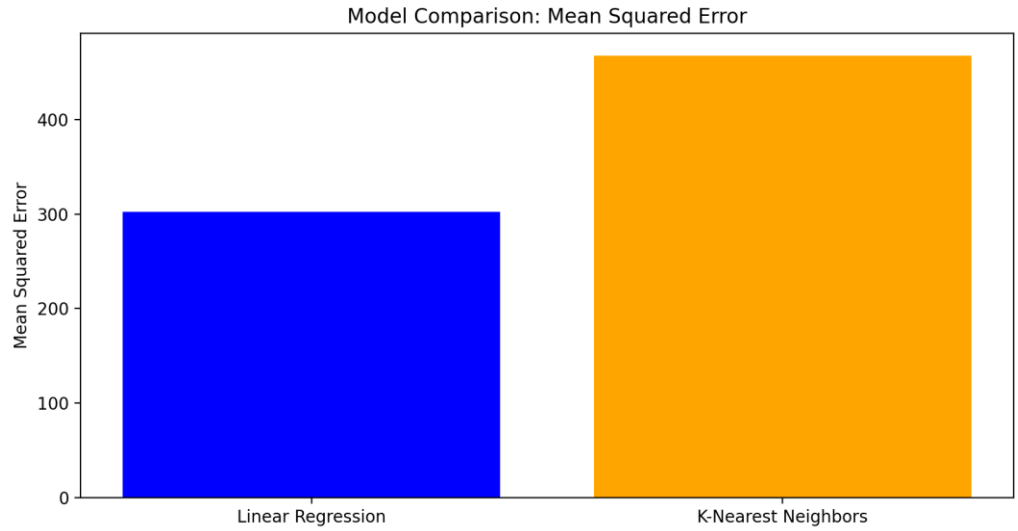
- MSE: 467.82
- R^2: -0.232

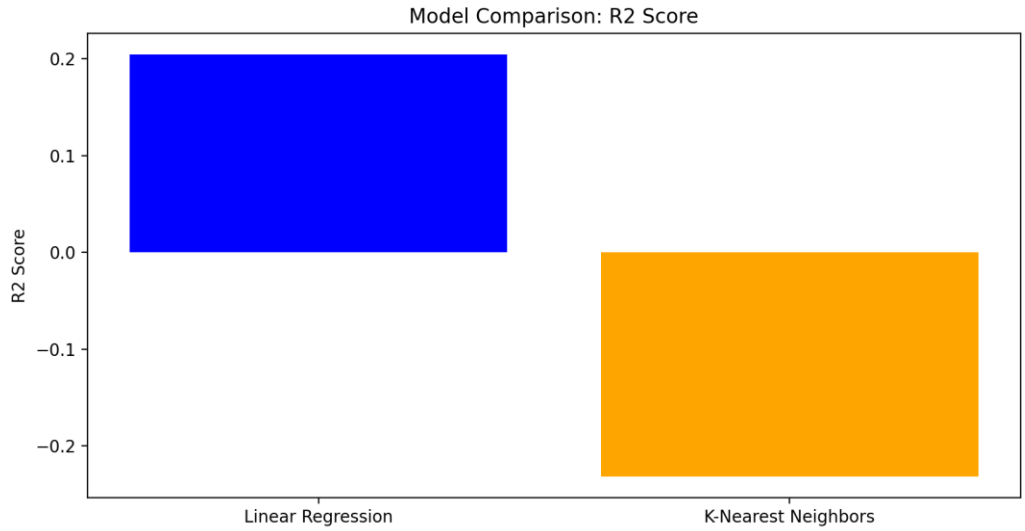
KNN showed weaker performance compared to linear regression, with a negative R^2 score indicating poor predictive power. This result is likely due to the model's sensitivity to data scaling and parameter tuning.

DISCUSSION

Both models were evaluated on the test set using *Mean Squared Error (MSE) and R-Squared (R^2)* scores. The linear regression model outperformed KNN in this case, as evidenced by a lower MSE and a positive R^2 score. However, while linear regression is easy to interpret, it assumes a linear relationship between the predictors and the target variable, which may not capture the complexities of real-world data. On the other hand, KNN could potentially perform better with more careful tuning of parameters and scaling of features.

VISUAL COMPARISON OF THE 2 MODELS:





For future work, hyperparameter tuning, particularly for the KNN model (adjusting the number of neighbors, k), and exploring other models such as ***Random Forest or XGBoost*** could improve the predictive performance.

CONCLUSION:

In this project, we applied two machine learning models***, Linear Regression and K-Nearest Neighbors (KNN)***, to predict delivery times using historical market and store data. While linear regression offered better accuracy in this specific context, the KNN model revealed its limitations when applied without optimal tuning. Future exploration into more sophisticated models and techniques can potentially lead to better prediction results, optimizing delivery operations and enhancing customer satisfaction.