

Practicum 3

Predicting NYC Green Taxi Tips with CRISP-DM & k-NN

Group 10

DA5020

Presented by: Hetvi Haresh Chudasama, Nandini Singh,
Qiuyu Liu, Raghavdeep Sharma, Trang Tu, Yinan Yan





BUSINESS UNDERSTANDING

The NYC Taxi and Limousine Commission (TLC) wants to understand and predict tipping behavior in green taxi rides.

Understanding what factors influence passengers to tip more can help:

- Improve service strategies,
- Optimize driver training,
- Boost driver motivation and earnings,
- Enhance overall passenger experience.

Predicting Tips impacts:

- Economy
- Performance Incentives
- Data- Driven Decisions



DATA OVERVIEW

★ DATASET SUMMARY

```
47 ````{r}
48 # Get a glimpse of the dataset structure
49 str(tripdata_df)
50 ````

'data.frame': 1048575 obs. of 19 variables:
 $ VendorID      : Factor w/ 2 levels "1","2": 2 2 2 2 2 2 2 2 2 ...
 $ lpep_pickup_datetime : POSIXct, format: "2018-01-01 00:18:00" "2018-01-01 00:30:00" "2018-01-01 00:07:00" "2018-01-01 00:32:00" ...
 $ lpep_dropoff_datetime: POSIXct, format: "2018-01-01 00:24:00" "2018-01-01 00:46:00" "2018-01-01 00:19:00" "2018-01-01 00:33:00" ...
 $ store_and_fwd_flag   : Factor w/ 2 levels "N","Y": 1 1 1 1 1 1 1 1 1 ...
 $ RatecodeID       : Factor w/ 7 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 ...
 $ PULocationID    : num  236 43 74 255 255 255 189 189 129 226 ...
 $ DOLocationID    : num  236 42 152 255 255 161 65 225 82 7 ...
 $ passenger_count  : num  5 5 1 1 1 1 5 5 1 1 ...
 $ trip_distance    : num  0.7 3.5 2.14 0.03 5.63 1.71 3.45 1.61 1.87 ...
 $ fare_amount      : num  6 14.5 10 -3 3 21 8.5 14.5 10 7.5 ...
 $ extra           : num  0.5 0.5 0.5 -0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
 $ mta_tax          : num  0.5 0.5 0.5 -0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
 $ tip_amount       : num  0 0 0 0 0 0 3.16 0 0 ...
 $ tolls_amount     : num  0 0 0 0 0 0 0 0 0 ...
 $ improvement_surcharge: num  0.3 0.3 0.3 -0.3 0.3 0.3 0.3 0.3 0.3 0.3 ...
 $ total_amount     : num  7.3 15.8 11.3 -4.3 4.3 ...
 $ payment_type     : Factor w/ 5 levels "1","2","3","4",..: 2 2 2 3 2 2 2 1 2 2 ...
 $ trip_type        : Factor w/ 2 levels "1","2": 1 1 1 1 1 1 1 1 1 ...
 $ trip_duration    : num  6 16 12 1 1 30 10 17 13 7 ...
```

DATA QUALITY & MISSING VALUES

- **ehail_fee**: 100% missing (dropped from analysis).
- **fare_amount** and **total_amount**: 5 missing values each.
- 10,135 records have pickup and drop-off times that are exactly the same (likely data errors).

```
93 ````{r}
94 # Check if the pickup time is later than or equal to the drop-off time
95 inconsistent_data <- tripdata_df %>%
96   filter(tripdata_df$lpep_pickup_datetime >= tripdata_df$lpep_dropoff_datetime)
97
98 nrow(inconsistent_data)
99
100 head(inconsistent_data, 5)
101 ````
```

DATA TYPE ISSUES & FIXES

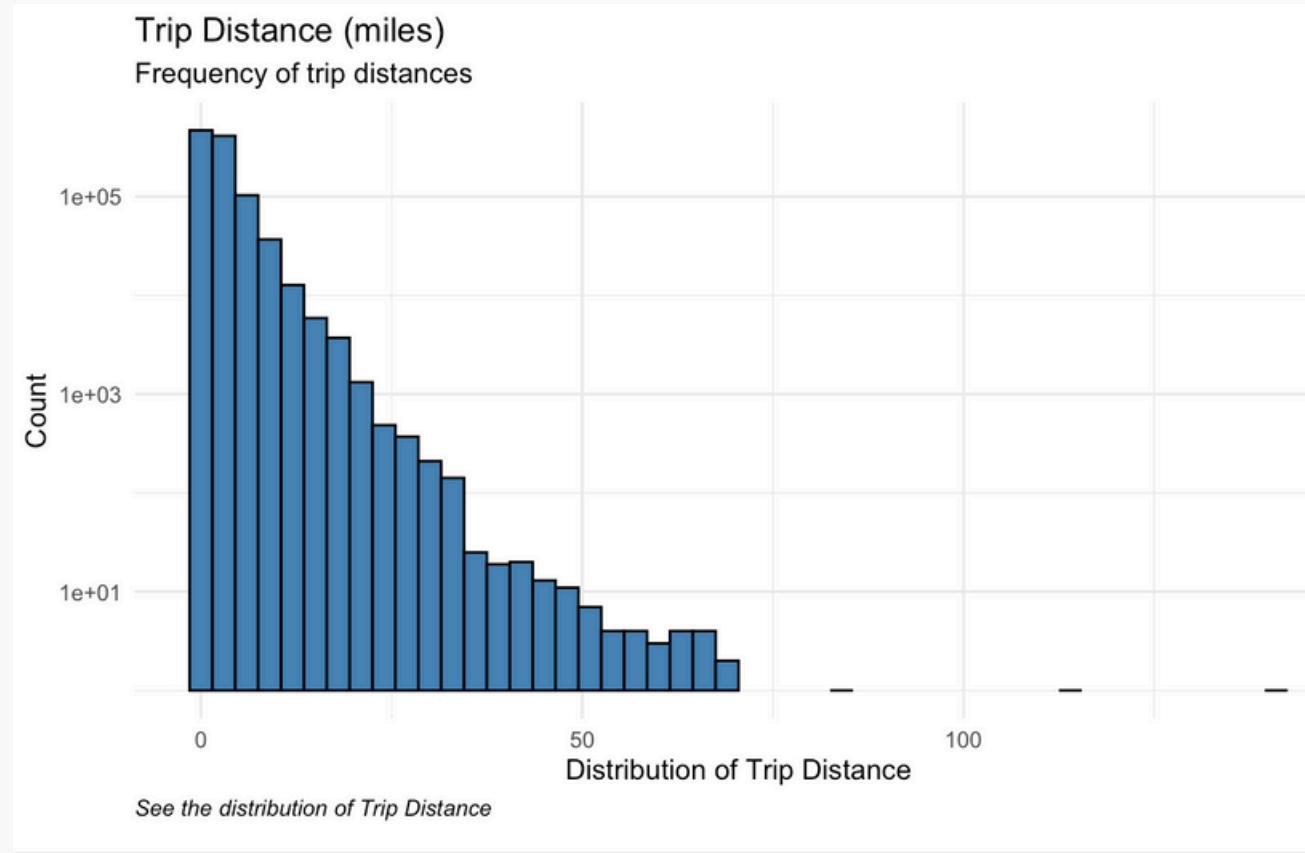
- **lpep_pickup_datetime**, **lpep_dropoff_datetime**: Converted to POSIXct.
- **fare_amount**, **total_amount**: Converted from character to numeric.
- **ehail_fee**: Detected as logical, but dropped due to NAs.

```
75 ````{r}
76 # Convert the pick up and drop off date and time to a Date object
77 tripdata_df$lpep_pickup_datetime <- as.POSIXct(tripdata_df$lpep_pickup_datetime, format="%m/%d/%Y %H:%M")
78 tripdata_df$lpep_dropoff_datetime <- as.POSIXct(tripdata_df$lpep_dropoff_datetime, format="%m/%d/%Y %H:%M")
79 ````
```

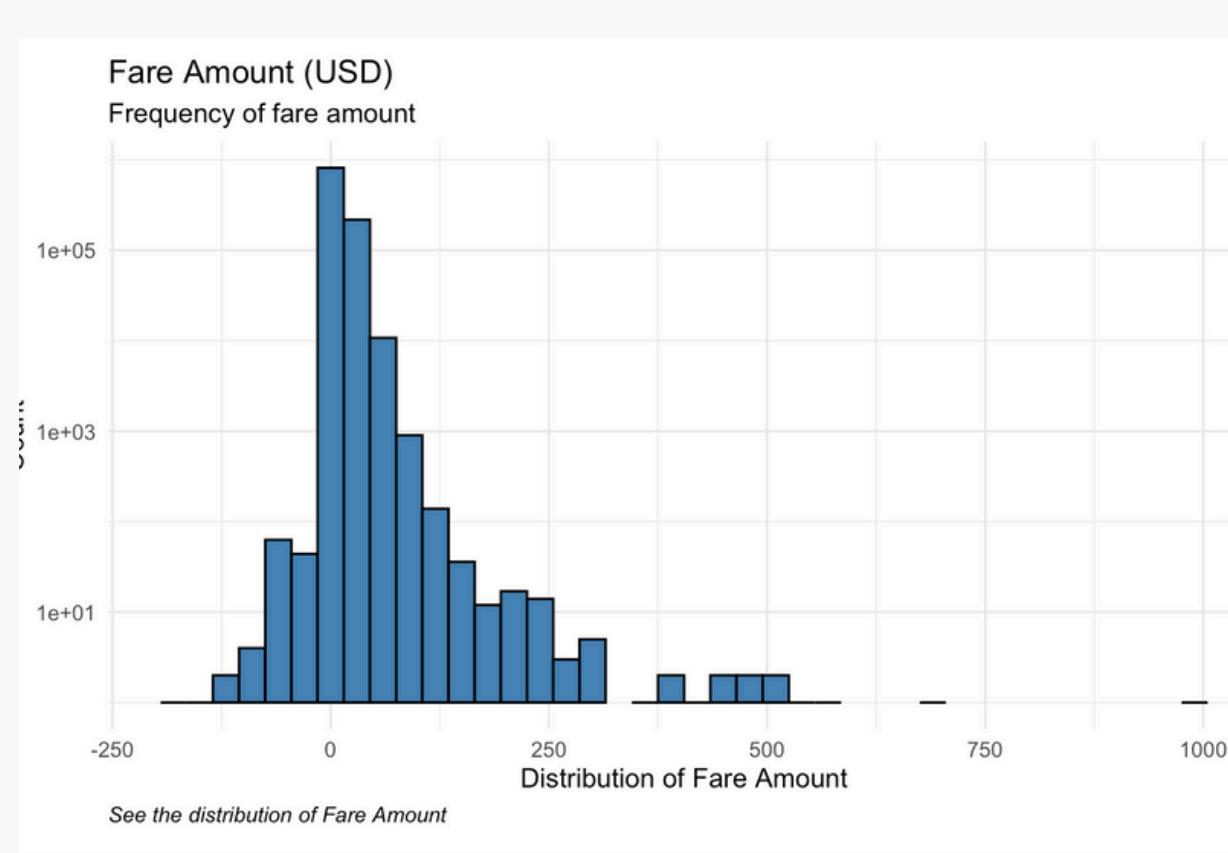
⚠ OUTLIERS

- **fare_amount**, **total_amount**: Negative values (e.g., -183) – unrealistic, flagged as outliers.
- **Extra**, **mta_tax**, **tip_amount**, **improvement_surcharge**: Contain negative values (e.g., -4.5, -2.72).
- **RateCodeID**: Should range from 1-6, but has 99 – a clear outlier.

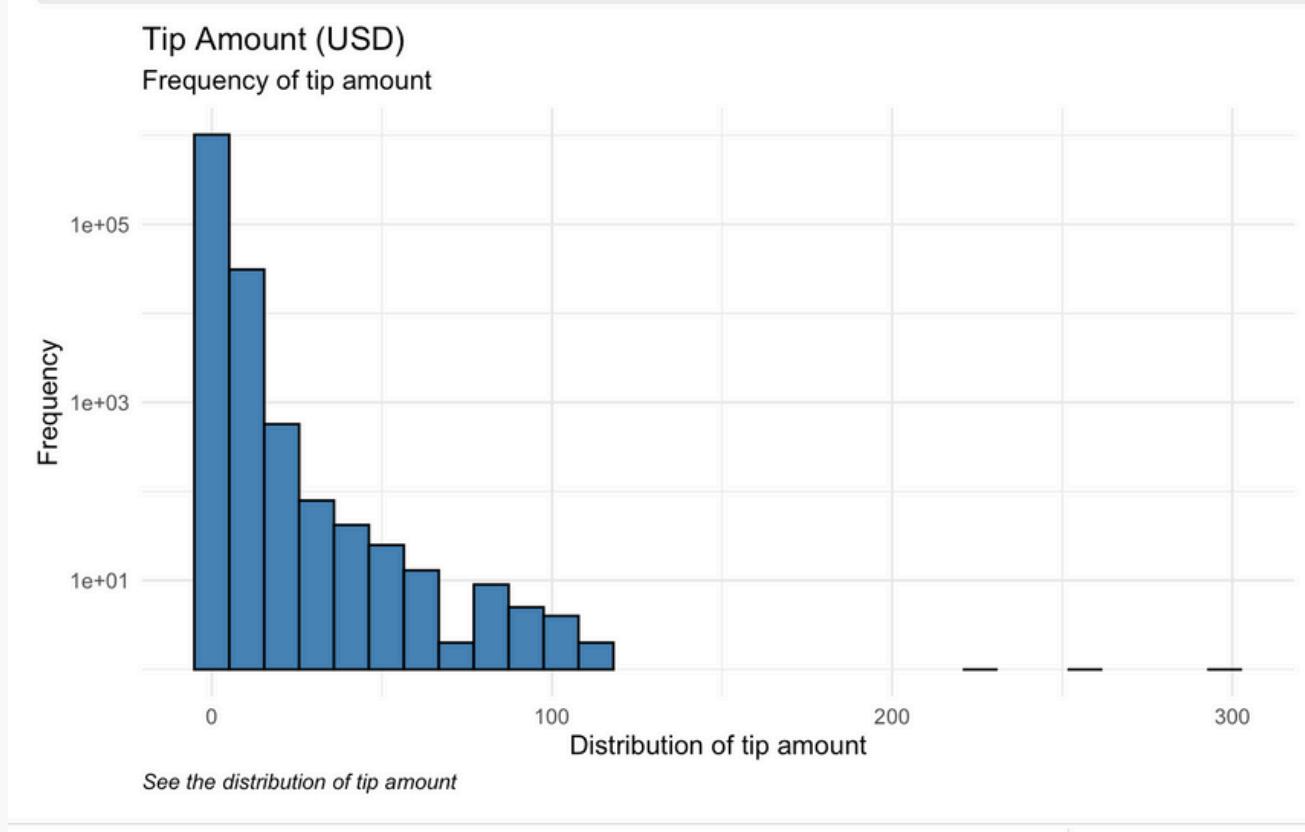
DATA UNDERSTANDING



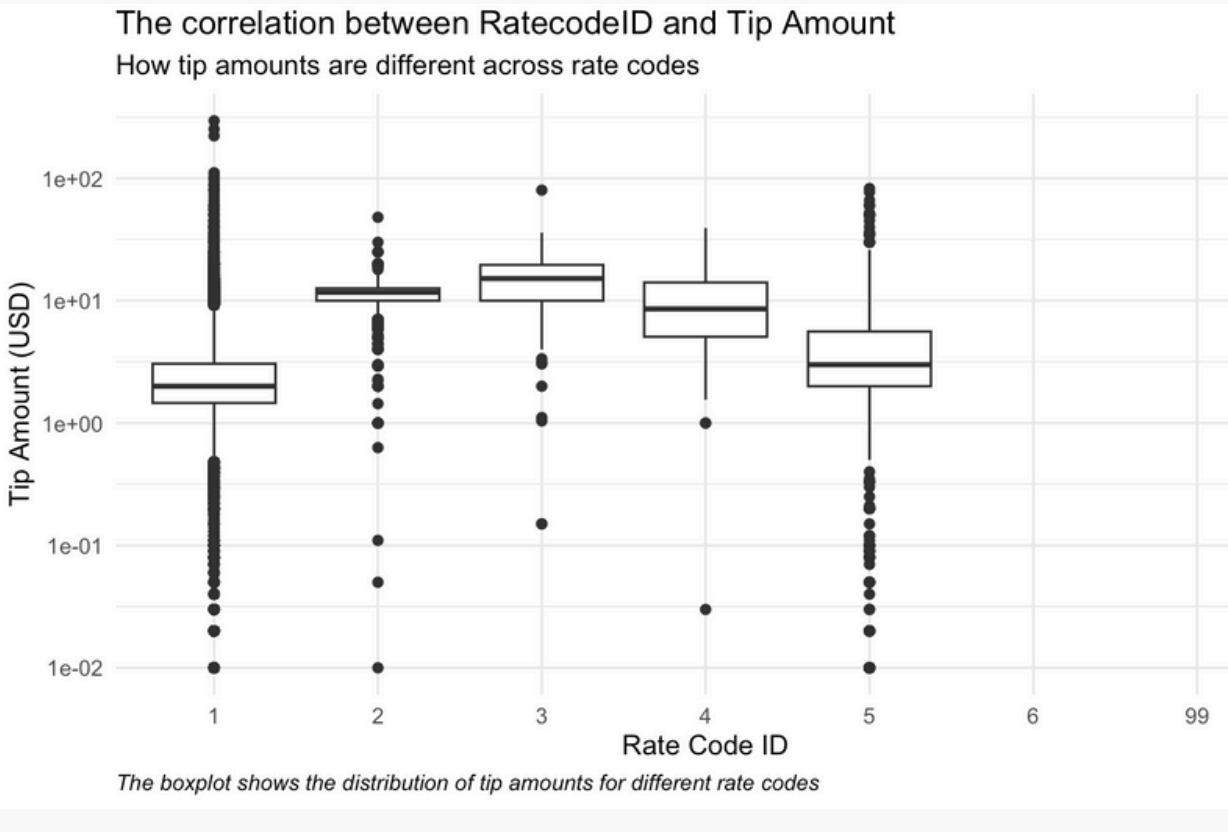
- Right-skewed, majority short trips
- Mean > Median, long-tail outliers
- Clean outliers on right



- Right-skewed, clustered near 0
- Negative values present → data error
- High fares need review

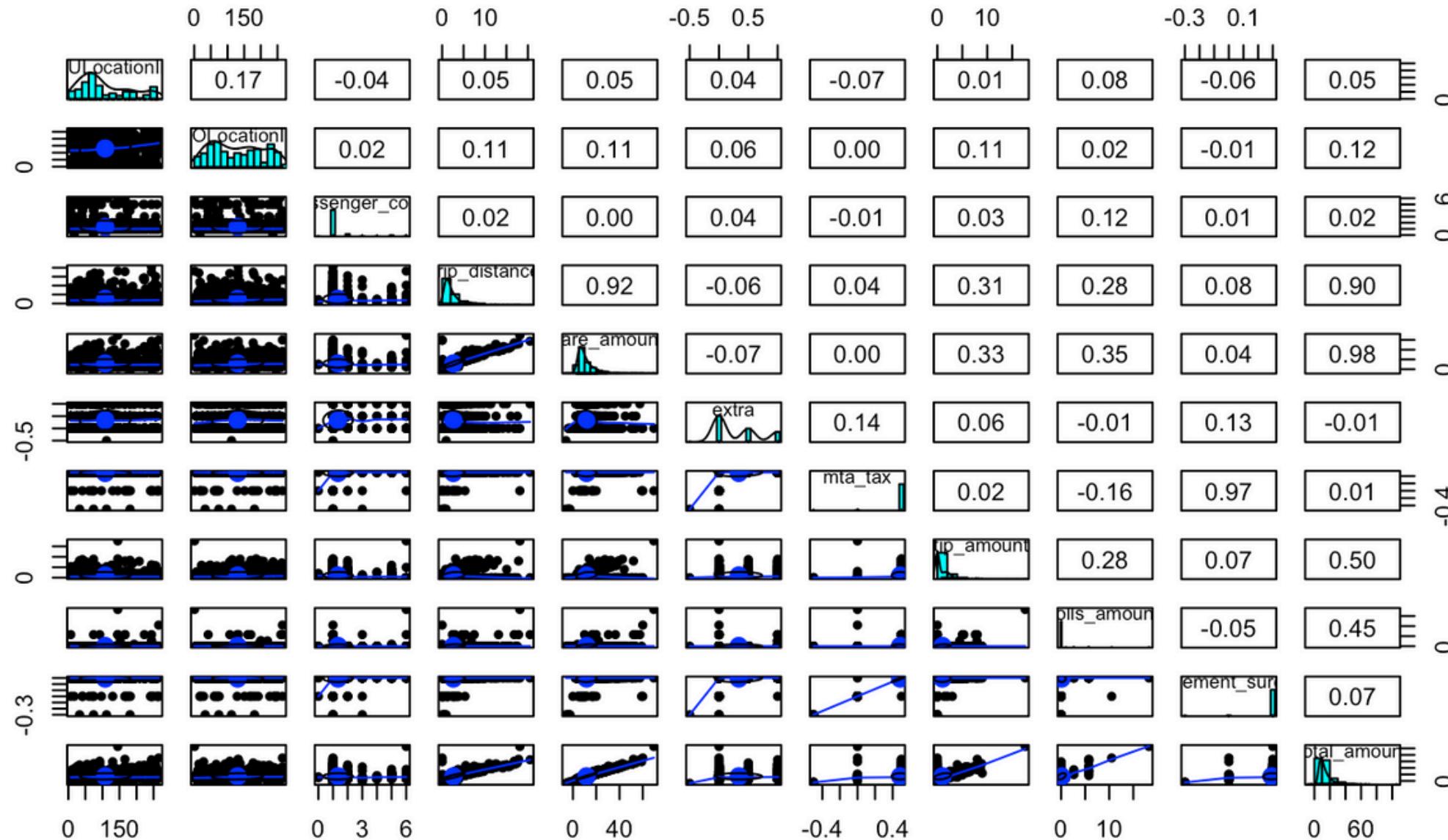


- Right-skewed, low tips are common
- Few large tips → potential outliers
- Shows tipping pattern



- RateCodes 2-4 have higher median tips
- Code 99 = outlier
- All codes have tip outliers
- Category 1 shows most extreme outliers

CORRELATION MATRIX & KEY RELATIONSHIPS



- Tip correlates with **Fare (0.35)** & **Distance (0.31)**

- Fare & Total Amount highly correlated (0.98)** → Remove to avoid data leakage

- Trip Distance ↔ Fare (0.90)** → Expected & valid relationship

- Multicollinearity risk:** Drop fixed charges (mta_tax, surcharge)

MISSING VALUES

```

430
431 ``{r}
432 # Check missing values
433 missing_values <- colSums(is.na(tripdata_df))
434
435 print(missing_values)
436 ```

```

VendorID	lpep_pickup_datetime	lpep_dropoff_datetime	store_and_fwd_flag	RatecodeID
0	0	0	0	0
PULocationID	DOLocationID	passenger_count	trip_distance	fare_amount
0	0	0	0	5
extra	mta_tax	tip_amount	tolls_amount	ehail_fee
0	0	0	0	1048575
improvement_surcharge	total_amount	payment_type	trip_type	
0	5	0	3	

OUTLIER ANALYSIS

```

462 ``{r}
463 # Calculate key descriptive statistics
464 mean_tip_amount <- mean(tripdata_df$tip_amount)
465 sd_tip_amount <- sd(tripdata_df$tip_amount)
466
467 # Calculate the z-score
468 tripdata_df$z3 <- abs((mean_tip_amount - tripdata_df$tip_amount) / sd_tip_amount)
469 tipAmount_outliers <- which(tripdata_df$z3 > 3)
470
471 dim(tripdata_df[tipAmount_outliers,])
472
473 head(tripdata_df[tipAmount_outliers,],5)
474 ```

```

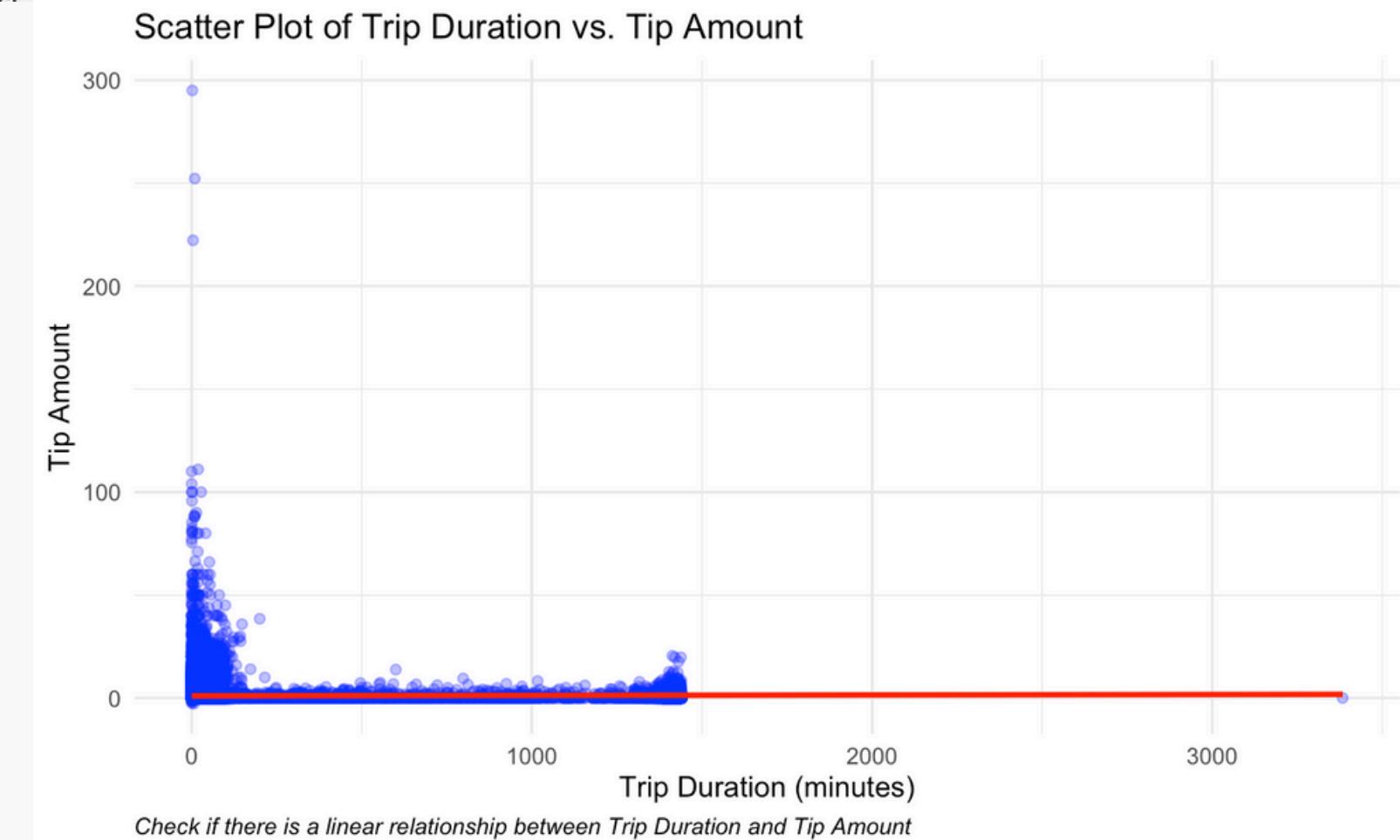
14444 outliers in the tip_amount variable

FEATURE SELECTION + ENGINEERING

- **✓ Useful predictors:** fare_amount, extra, tolls_amount, total_amount
- **🚫 Excluded variables:** mta_tax, improvement_surcharge, passenger_count, location IDs, trip_distance
- **📈 R² = 0.9803** → Strong model performance after exclusion
- **🔍 Low p-values** → All predictors statistically significant
- **⚠ Multicollinearity:** Dropped trip_distance, total_amount & fare_amount not used together

⌚ **Added trip_duration:** Median = 10 min, Max = 3383 min → right-skewed, outliers present

```
566 ````{r}  
567 # Create trip_duration column in minutes  
568 tripdata_df569 tripdata_df$lpep_pickup_datetime,  
570 units = "mins"))
```



📊 **trip_duration vs. tip_amount:** Pearson r ≈ 0.012 → very weak correlation

DATA PREPARATION

Missing Values & Outliers Removal

- Remove & Filter

- Remove NAs column email_fee

```
tripdata_df$ehail_fee <- NULL
```

- Remove all rows with missing values in columns, fare_amount, total_amount, and trip_type

```
tripdata_df_cleaned <- tripdata_df %>%  
  filter(!is.na(trip_type)) %>%
```

- Filter all positive values in those features

```
# Remove records where dropoff time is earlier than pickup time  
tripdata_df_cleaned <- tripdata_df %>%  
  filter(lpep_pickup_datetime < lpep_dropoff_datetime)  
  
# Keep only trips with valid duration and passenger count  
tripdata_df_cleaned <- tripdata_df %>%  
  filter(trip_duration >0) %>%
```

- Extracted the tip_amount as target variable

```
# Extract data that contains tip_amount data  
tripdata_tip_amount <- tripdata_df$tip_amount
```

- Applied Z-score method ($|z| > 3$) on all numeric features

- Perform z-score outlier detection on all numeric columns(value with a z-score greater than 3 is considered an outlier and removed).
- Outliers are values that lie more than 3 standard deviations away from the mean.

```
# Remove outliers using z-score ( $|z| > 3$ ) in all numeric columns  
remove_outliers <- function(data) {  
  numeric_columns <- names(data)[sapply(data, is.numeric)]  
  
  # Loop through each numeric column  
  for (col in numeric_columns) {  
    mean_val <- mean(data[[col]], na.rm = TRUE) # mean  
    sd_val <- sd(data[[col]], na.rm = TRUE) # standard deviation  
    z <- abs((data[[col]] - mean_val) / sd_val) # z-score for each value  
  
    # Removes extreme outliers beyond 3 standard deviations  
    data <- data[z <= 3 | is.na(z), ]  
  }  
  
  return(data)  
}  
  
# Apply outlier removal  
tripdata_df_cleaned <- remove_outliers(tripdata_df_cleaned)
```

DATA PREPARATION

- Normalization and the result (min-max)

- **Apply min-max scaling to normalize dataset**
 - **All numeric variables will be scaled to a range between 0 and 1.**

```
#Select numeric and remove target variable
tripdata_numeric_knn <- tripdata_df_cleaned %>% select_if(is.numeric)
%>% select(-tip_amount)

# Create a pre-processing model using min-max normalization
preproc_minmax <- preprocess(tripdata_numeric_knn, method=c("range"))

# Apply the normalization model to the numeric data
norm_minmax <- predict(preproc_minmax, tripdata_numeric_knn)

# Check the result
summary(norm_minmax)
```

Warning: No variation for for: mta_tax, improvement_surcharge

PULocationID	DOLocationID	passenger_count
Min. :0.0000	Min. :0.0000	Min. :0.0000
1st Qu.:0.1756	1st Qu.:0.2214	1st Qu.:0.2500
Median :0.3015	Median :0.4695	Median :0.2500
Mean :0.4019	Mean :0.4753	Mean :0.2835
3rd Qu.:0.6221	3rd Qu.:0.7099	3rd Qu.:0.2500
Max. :1.0000	Max. :1.0000	Max. :1.0000
trip_distance	fare_amount	extra
Min. :0.00000	Min. :0.0000	Min. :0.0000
1st Qu.:0.08807	1st Qu.:0.2000	1st Qu.:0.0000
Median :0.14679	Median :0.2833	Median :0.5000
Mean :0.19757	Mean :0.3361	Mean :0.3549
3rd Qu.:0.26055	3rd Qu.:0.4333	3rd Qu.:0.5000
Max. :1.00000	Max. :1.0000	Max. :1.0000
mta_tax	tolls_amount	improvement_surcharge
Min. :0.5	Min. :0.00e+00	Min. :0.3
1st Qu.:0.5	1st Qu.:0.00e+00	1st Qu.:0.3
Median :0.5	Median :0.00e+00	Median :0.3
Mean :0.5	Mean :2.21e-05	Mean :0.3
3rd Qu.:0.5	3rd Qu.:0.00e+00	3rd Qu.:0.3
Max. :0.5	Max. :1.00e+00	Max. :0.3
total_amount	trip_duration	
Min. :0.0000	Min. :0.00000	
1st Qu.:0.2500	1st Qu.:0.01942	
Median :0.3317	Median :0.02913	
Mean :0.3886	Mean :0.03659	
3rd Qu.:0.4904	3rd Qu.:0.04854	
Max. :1.0000	Max. :1.00000	

All numeric variables are now scaled to a range between 0 and 1.

Normalization & Data Encoding

- Data encode and the result

- Encode all categorical features to create dummy variables
 - Nominal variables - one-hot encoded using the `dummyVars()` function
 - Binary variable such as `'store_and_fwd_flag'`, `'VendorID'`, and `'trip_type'` - label-encoded
 - The resulting dataset contains only numeric features

```
# Create dummy variables
encode_variables <- function(df) {
  # One-hot encode categorical variables using dummyVars()
  vars_to_dummy <- c("payment_type", "RatecodeID")
  dummy_model <- dummyVars(~ ., data = df[, vars_to_dummy])
  dummy_df <- as.data.frame(predict(dummy_model, newdata = df))

  # Encode binary variable store_and_fwd_flag
  df$store_and_fwd_flag <- ifelse(df$store_and_fwd_flag == "Y", 1, 0)
  df$VendorID <- ifelse(df$VendorID == "1", 1, 0)
  df$trip_type <- ifelse(df$trip_type == "1", 1, 0)

  # Remove original nominal vars
  df_encoded <- df %>%
    select(-all_of(vars_to_dummy))

  # Combine cleaned df with dummy variables
  final_df <- bind_cols(df_encoded, dummy_df)
  return(final_df)
}

$ payment_type.1      : num  0 0 0 0 0 0 0 0 0 1 1 ...
$ payment_type.2      : num  1 1 1 1 1 1 1 1 0 0 ...
$ payment_type.3      : num  0 0 0 0 0 0 0 0 0 0 0 ...
$ payment_type.4      : num  0 0 0 0 0 0 0 0 0 0 0 ...
$ payment_type.5      : num  0 0 0 0 0 0 0 0 0 0 0 ...
$ RatecodeID.1         : num  1 1 1 1 1 1 1 1 1 1 1 ...
$ RatecodeID.2         : num  0 0 0 0 0 0 0 0 0 0 0 ...
$ RatecodeID.3         : num  0 0 0 0 0 0 0 0 0 0 0 ...
$ RatecodeID.4         : num  0 0 0 0 0 0 0 0 0 0 0 ...
$ RatecodeID.5         : num  0 0 0 0 0 0 0 0 0 0 0 ...
$ RatecodeID.6         : num  0 0 0 0 0 0 0 0 0 0 0 ...
$ RatecodeID.99        : num  0 0 0 0 0 0 0 0 0 0 0 ...
```

K-NN REGRESSION MODELING

Prepare Modeling

Splitted the dataset into training_(80%) and testing_(20%)_sets

```
# Tripdata_df_encoded will be the one to use for the splitting

set.seed(123) # for reproducibility
train_index <- createDataPartition(tripdata_df_encoded$tip_amount, p = 0.8,
list = FALSE)

# Set the data in training and testing sets
data_train <- tripdata_df_encoded[train_index, ]
data_test <- tripdata_df_encoded[-train_index, ]

# Check the dimensions of the data_train and data_test
dim(data_train)
dim(data_test)
```

Result

```
[1] 722910    29
[1] 180727    29
```

The `**data_train**` has 80% of the total data. It has 722910 rows and 29 columns, used to "teach" the KNN model—this is where it stores all the examples it will later use to find the k-nearest neighbors when making predictions.

The `**data_test**` has 20% of the total data. It has 180727 rows and 29 columns, used to evaluate the model's performance on unseen data, simulating how it would perform in a real-world setting.

K-NN REGRESSION MODELING

Develop the k-nn regression model.

Use a function: `knn_predict(data_train, data_test, k)`.

```
# Create function knn_predict
knn_predict <- function(data_train, data_test, k) {
  # Extract predictors (numeric columns except tip_amount)
  train_X <- data_train %>% select(where(is.numeric)) %>% select(-tip_amount)
  test_X <- data_test %>% select(where(is.numeric)) %>% select(-tip_amount)

  # Extract target variable
  train_y <- data_train$tip_amount
  test_y <- data_test$tip_amount

  # Use knn.reg to predict
  knn_output <- knn.reg(train = train_X, test = test_X, y = train_y, k = k)

  # Extract predictions
  predictions <- knn_output$pred

  # Calculate Mean Squared Error
  mse <- mean((predictions - test_y)^2, na.rm = TRUE)

  return(mse)
}
```

"k= 5 Mean Squared Error (MSE) = 0.115154130395568'

- This k-NN model uses numeric features to predict taxi tips.
- With k = 5, it achieves an MSE of 0.115, or about 11.5 cents off per prediction.
- Although tipping behavior is hard to predict, the model performs well because of the low MSE error.

MODEL EVALUATION

Defined a function to test multiple k values using KNN

```
# Define a range of k values
k_values <- c(1, 3, 5, 11, 15, 25, 31, 35, 50, 75, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000)
# random 20 numbers
```

MSE Results by k Value

k <dbl>	mse <dbl>	k <dbl>	mse <dbl>
1	0.1379918	100	0.2715905
3	0.1126836	200	0.3462940
5	0.1151541	300	0.3941127
11	0.1325874	400	0.4281083
15	0.1432449	500	0.4538292
25	0.1664780	600	0.4743941
31	0.1785540	700	0.4913409
35	0.1862819	800	0.5056349
50	0.2110994	900	0.5178411
75	0.2444370	1000	0.5287554

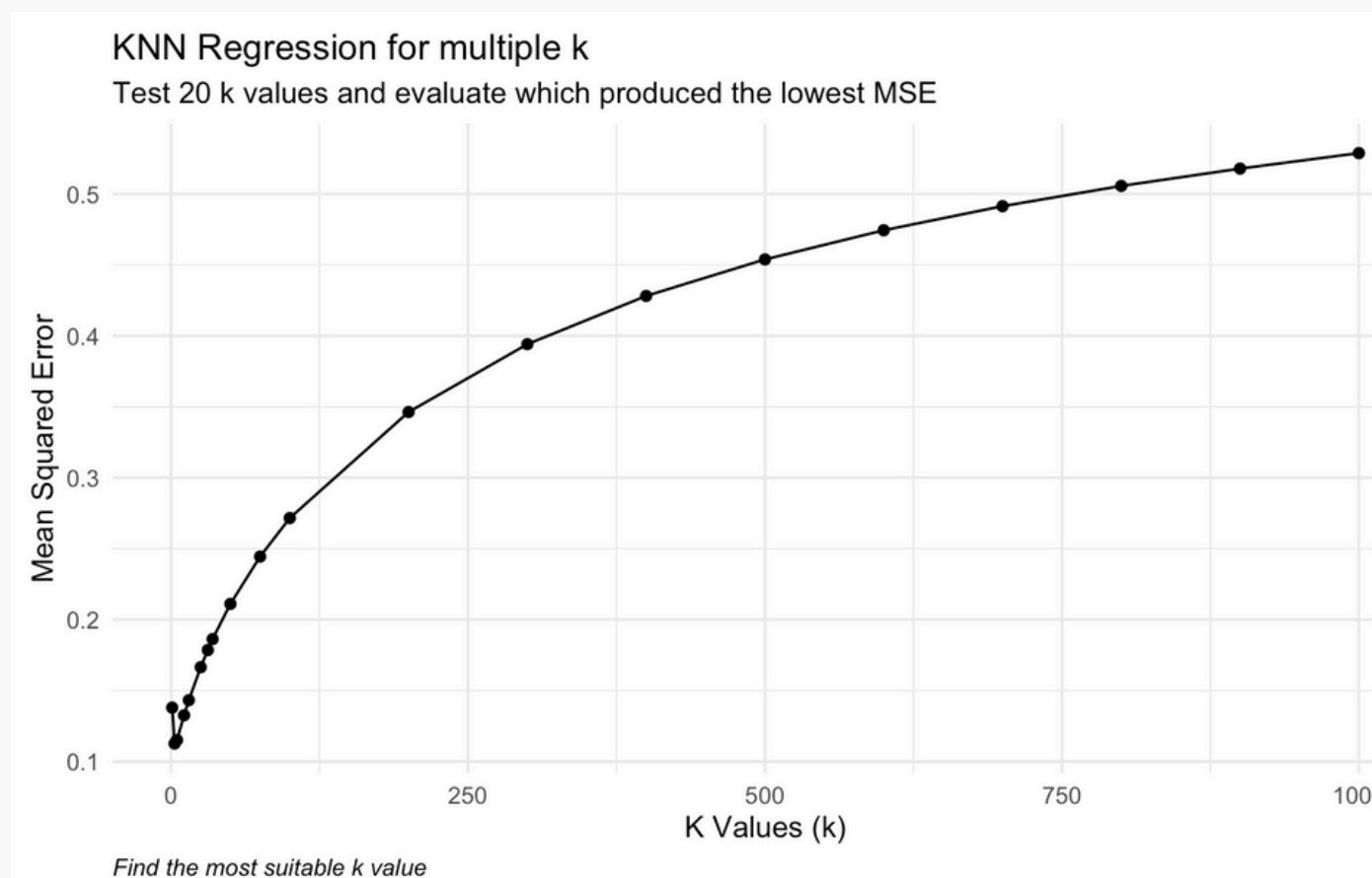
- MSE is lowest when k = 3 (0.1126836).
- The higher k is, the higher MSE becomes
- k = 3 offers a good balance between bias and variance and is less sensitive to noise.

Final Thoughts on the Model

Optimal k Value Identified

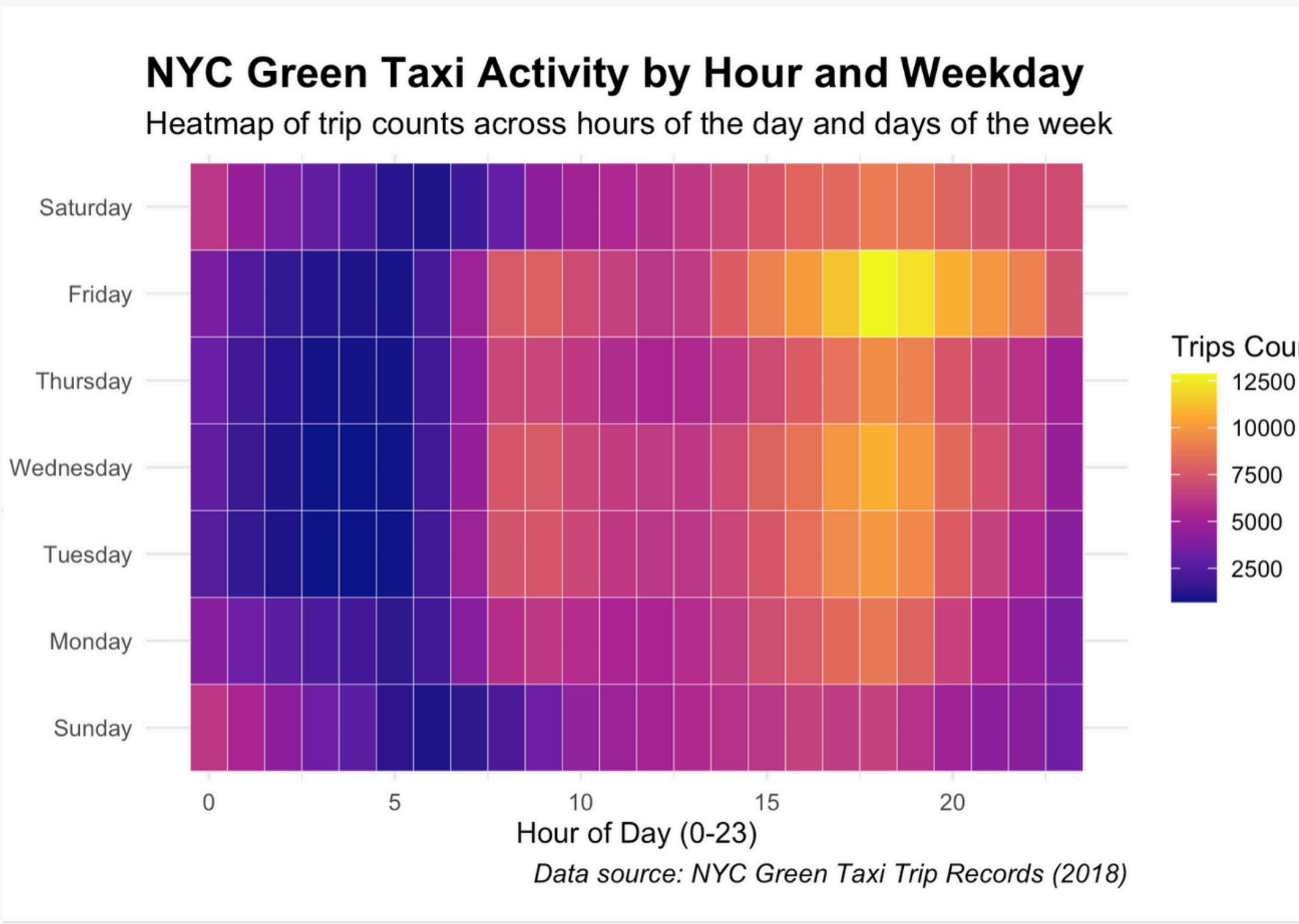
```
# Find the k value that gives the lowest MSE
best_k <- test_multiple_k_results$k[which.min(test_multiple_k_results$mse)]
print(best_k)
```
[1] 3
```

## Error Trend Across k Values



**From the line chart, we observe that the MSE keeps increasing as k increases, which suggests the model becomes too simple and starts underfitting. Larger k values don't perform well.**

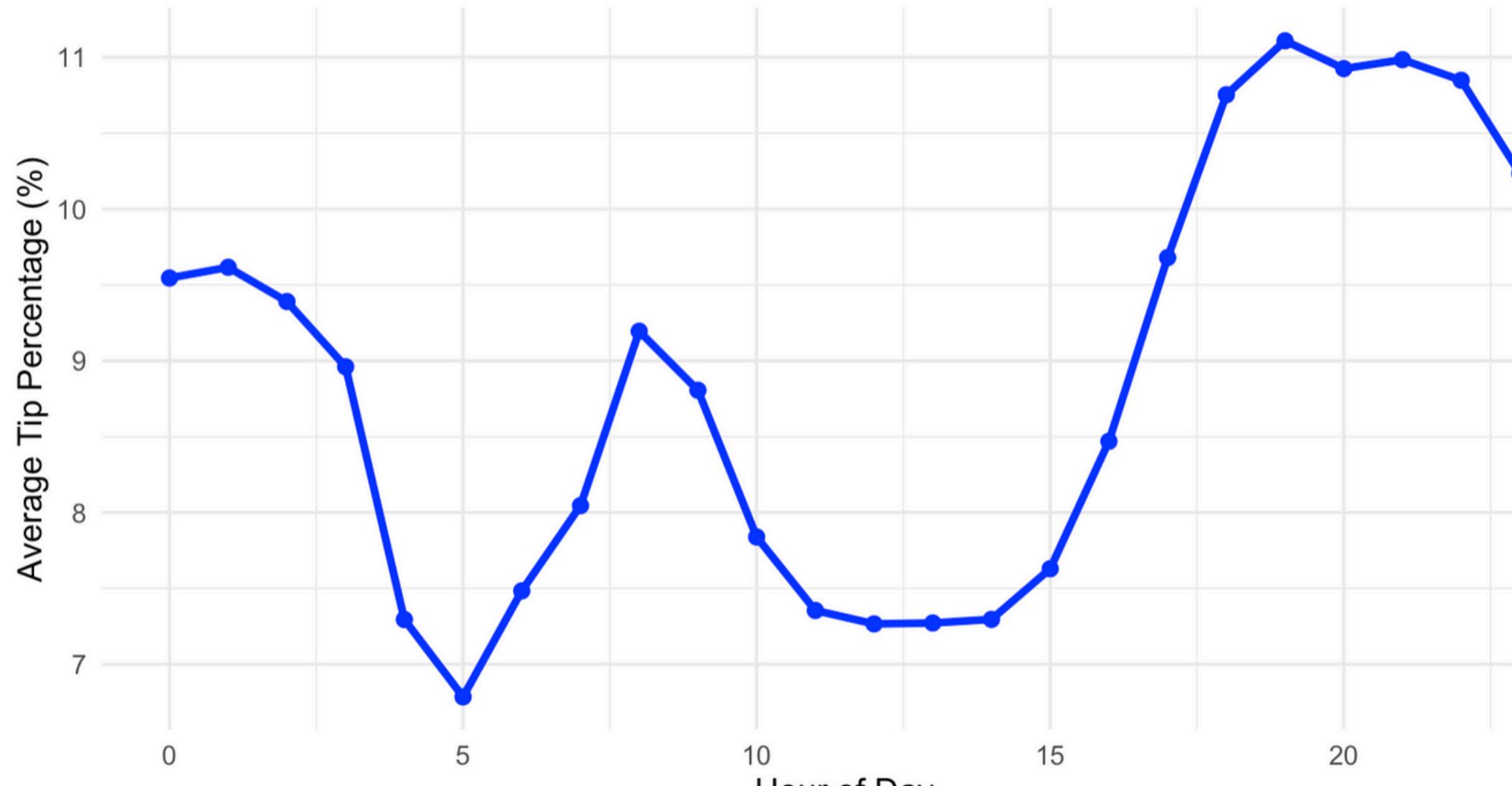
# Understanding NYC Taxi Trips



- Peak taxi activity occurs on Friday evenings (around 6–7 PM), indicating high demand for end-of-week travel and outings.
- Lowest activity is observed during early morning hours (12 AM – 6 AM) across all days, showing minimal taxi usage overnight.

## Average Tip Percentage by Hour of Day

How tipping behavior (as a percentage of fare) varies over the day



Data source: NYC Green Taxi Trip Records (2014)

- The best tipping behavior is seen in the evenings, especially from 5 PM to 9 PM, while the worst is early morning (around 4–6 AM) and midday.
- Strong upward trend beginning around 4 PM, peaking at over 11% by 7–8 PM.

# CONCLUSION

- This model is suitable for **short-term forecasting** as an initial analyst tool.
- Feature engineering: **more informative predictors** such as **trip duration, passenger's satisfaction scores** increase the model accuracy.
- **Advanced machine learning algorithms**, such as **decision trees, random forests, or gradient boosting**, may help model these complexities.

# THANK YOU

