PROJECT TITLE

K-Nearest Neighbor (K-NN) Classifier implementation with three different distance measurement techniques and comparative discussion of k-values with the help of the R programming language.

PROJECT OVERVIEW

The main goal of the project is to establish a K-NN classifier for a randomly selected dataset with more than 5000 instances. With three alternative distance measuring techniques—Euclidean Distance, Manhattan Distance, and Maximum Dimension Distance—this study seeks to effectively implement the K-NN classifier model on the selected dataset. These three distinct distance metrics, each with a different k-value (1 to 20), demonstrate the model's accuracy in class prediction. Thus, test data and training data will be separated from the selected dataset. The attributes that the model will use to classify the class attribute will then be specified. Based on the expected outcomes, the test data will be used to estimate the model's accuracy and recall rate. Following that, the model will store the accuracy and recall data into a data frame with different k-values. Based on this finding, the project will attempt to demonstrate the accuracy and recall rate against various K-values and identify the best effective distance measure for each K-value.

PROJECT SOLUTION

The project solution will be based on the following steps –

1. **Dataset selection-** A dataset with several attributes and a class attribute has to be chosen before the model building. As a requirement, the dataset should contain at least 5000 instances, 2 attributes and 1 class attribute.

2. **Pre-processing -** Before the implementation begins, the dataset needs to be pre-processed and cleaned. For this, data cleaning, data integration, data transformation, data reduction and other techniques should be used.
3. **Training and Testing data-** After the pre-processing, the dataset should be separated into two parts – training data and testing data. The proportion of the separation is usually 70:30 (train: test) based on the type of dataset.

4. **Model Building-** Building the K-NN model is the next step toward the project.
    i. **Distance Measurement:** The model should include three different distance measuring techniques with three different equations.

$$\textit{Euclidean: } \boldsymbol{DE(a,b)} = \sqrt{\sum_{i=1}^{n} (\boldsymbol{b_i} - \boldsymbol{a_i})^2}$$

$$\textit{Manhattan: } \boldsymbol{DM(a,b)} = \sqrt{\sum_{i=1}^{n} |\boldsymbol{b_i} - \boldsymbol{a_i}|}$$

$$\textit{Max Dimension: } \boldsymbol{DMD(a,b)} = \max_{i=1}(|\boldsymbol{b_i} - \boldsymbol{a_i}|)$$

ii. **Choosing K-value:** After calculating the distance the next step is to choose the k-value. Depending on the k-value will determine the model's performance and efficiency.

iii. **Predicting Class:** Upon the results of the model, and the chosen values of K, it will predict the most occurred class attribute which is the nearest of the testing point.

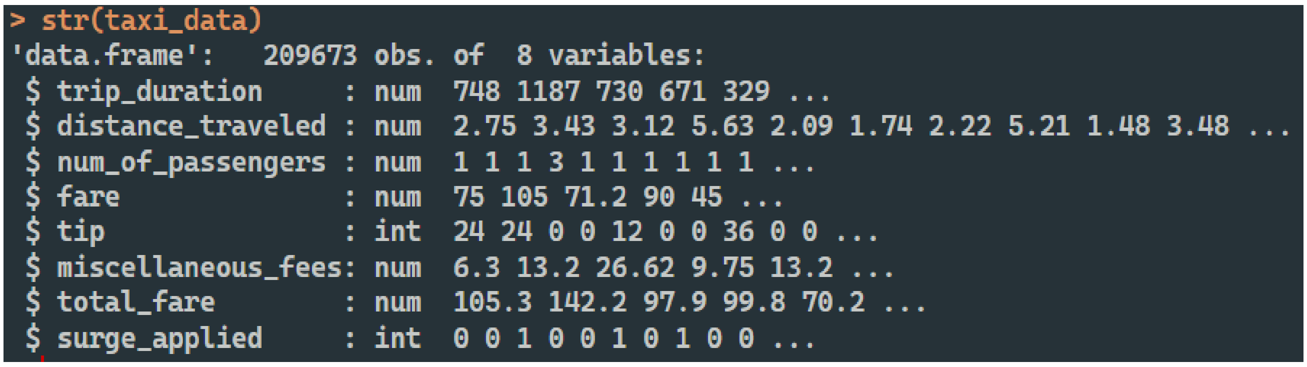
iv. **Accuracy:** After the prediction of the class, it will check whether the classified class is right or wrong. Considering all the prediction values with actual values it will calculate the accuracy of the K-NN model for corresponding distance metric and most effective k-value for the dataset.

5. **Result Analysis-** After acquiring all the results for each distance metric of K-NN model, it will analyze the results by plotting, making comparison table or, discussion.

6. **Further Improvement and Discussion-** The last step of the project is to verify the results, finding limitations and discussing the advantages and disadvantages of using K-NN for the chosen dataset.

PROJECT DATASET

The dataset that has been chosen for this project is "Taxi Fare" dataset. It was collected from *Kaggle* under the name of "taxi_fare.csv". This dataset contains total 8 variables (columns) and 209673 observations (rows). Below are the dataset details—



```
> str(taxi_data)
'data.frame':   209673 obs. of  8 variables:
 $ trip_duration     : num  748 1187 730 671 329 ...
 $ distance_traveled : num  2.75 3.43 3.12 5.63 2.09 1.74 2.22 5.21 1.48 3.48 ...
 $ num_of_passengers : num  1 1 1 3 1 1 1 1 1 1 ...
 $ fare              : num  75 105 71.2 90 45 ...
 $ tip               : int  24 24 0 0 12 0 0 36 0 0 ...
 $ miscellaneous_fees: num  6.3 13.2 26.62 9.75 13.2 ...
 $ total_fare        : num  105.3 142.2 97.9 99.8 70.2 ...
 $ surge_applied     : int  0 0 1 0 0 1 0 1 0 0 ...
```

This project will only focus on classifying the ‘surge_applied’ variable. This class attribute contains the categorical value as “0” and “1” which indicates if the taxi data includes any surge charge or not. Generally, a serge charge is considered or, calculated by the total duration of a taxi rent and the total distance it covered during the rent. Therefore, the project will be focusing only on the first two columns of the dataset which are “trip_duration” and “distance_traveled”. These two attributes will be chosen as the variable point for the K-NN model to calculate distance. Other attributes are negligible because these attributes do not affect the surge charge directly.

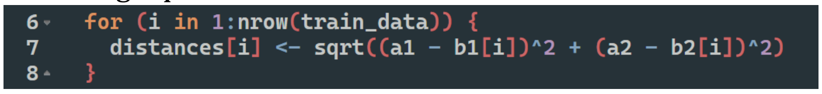
DATA PREPROCESSING

Before building the model, data pre-processing on the selected dataset is necessary. The pre-processing steps that have been used for this dataset is discussed below –

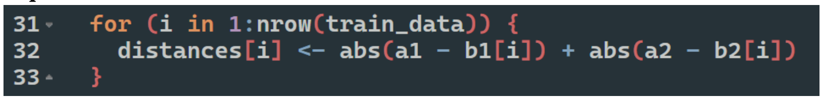
a. **Data Cleaning:** The whole dataset has been checked for any missing data and found no such data. Then it has been checked for missing data or, zero values in the data points and found 5 data points with that. The project removed those data and further checked for mismatched variables and no such data found. There were noisy data found and no major irrelevancy discovered.
b. **Data Reduction:** The first two data point attributes “trip_duration”, “distance_traveled” and the last class attribute “surge_applied” has been choosed for final dataset and rest of the columns were reduced from the main dataset.
c. **Data Transformation:** The class attribute was transformed from binary variable to “Yes” and “No” and integrated with the latest dataset. This new column was named “Class” and this column is being considered as the classification column for the project
d. **Feature Engineering:** The new ‘Class’ column was added to implement the classification model around it. The results of the model are directly involved and the feature is considered as the prediction column for the model.

MODEL IMPLEMENTATION

Distance Measurement Equation: The code implements three different distance measurement equations to compute the distances between data points: Euclidean distance, Manhattan distance, and Maximum Dimension distance.

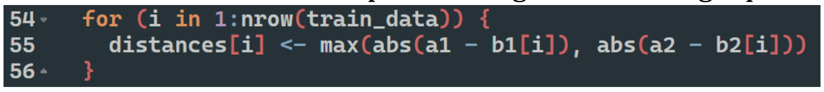
1. Euclidean Distance - The Euclidean distance is a commonly used distance metric in machine learning. It calculates the straight-line distance between two points in a multidimensional space. In the code, the Euclidean distance is computed using the following equation:


```
for (i in 1:nrow(train_data)) {
  distances[i] <- sqrt((a1 - b1[i])^2 + (a2 - b2[i])^2)
}
```

2. Manhattan Distance: The Manhattan distance, also known as the city block distance or L1 norm, calculates the distance between two points by summing the absolute differences of their coordinates. In the code, the Manhattan distance is computed using the following equation:


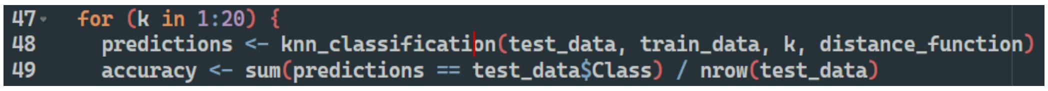
```
for (i in 1:nrow(train_data)) {
  distances[i] <- abs(a1 - b1[i]) + abs(a2 - b2[i])
}
```

3. Maximum Dimension Distance: The Maximum Dimension distance measures the maximum absolute difference between any pair of coordinates. In the code, the Maximum Dimension distance is computed using the following equation:



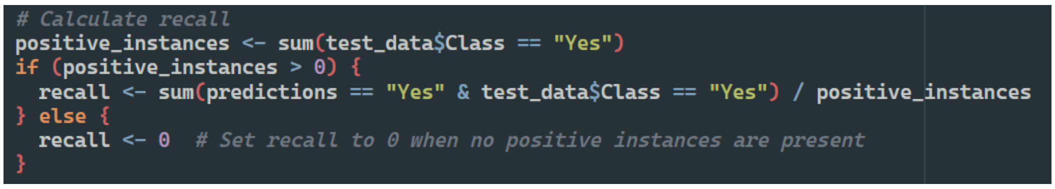
```r
  for (i in 1:nrow(train_data)) {
    distances[i] <- max(abs(a1 - b1[i]), abs(a2 - b2[i]))
  }
```

KNN Algorithm Implementation: The KNN algorithm is implemented in the knn_classification() function. It takes the training data, test data, training labels, the value of k, and the chosen distance metric as input in the parameters. The function uses the distance metric functions to calculate distance between each test data point and all training data. It identifies the k-nearest neighbors based on the smallest distances. Then, it predicts the class label for each test data point by selecting the most frequent label among the nearest neighbors. Finally, the predicted labels are returned.



```r
for (k in 1:20) {
  predictions <- knn_classification(test_data, train_data, k, distance_function)
  accuracy <- sum(predictions == test_data$Class) / nrow(test_data)
```

Evaluation with Different K-Values: In order to determine the optimal number of neighbors (k), a set of k-values (1 to 20) is defined. The code iterates over each distance metric and each k-value to evaluate the performance of the KNN model. For each combination, the knn_classification() function is called to obtain the predicted labels for the test data. The accuracy is calculated by comparing the predicted labels with the actual labels of the test data. Additionally, the true positive (TP) and false negative (FN) values are computed to calculate the recall rate of the results.



```r
# Calculate recall
positive_instances <- sum(test_data$Class == "Yes")
if (positive_instances > 0) {
  recall <- sum(predictions == "Yes" & test_data$Class == "Yes") / positive_instances
} else {
  recall <- 0  # Set recall to 0 when no positive instances are present
}
```

Subset Selection for Testing: To make the evaluation process more efficient, a subset of the test data is randomly selected. By reducing the size of the test data, the computation time is reduced while still providing a representative sample for evaluation. The subset is used to calculate accuracy and recall values for different k-values and distance metrics.

Results Compilation: The results of the evaluation are stored in the 'knn_results' data frame. It includes the distance metric, k-value, accuracy, and recall values for each combination. This data frame is then saved as a CSV file for further analysis and reporting.

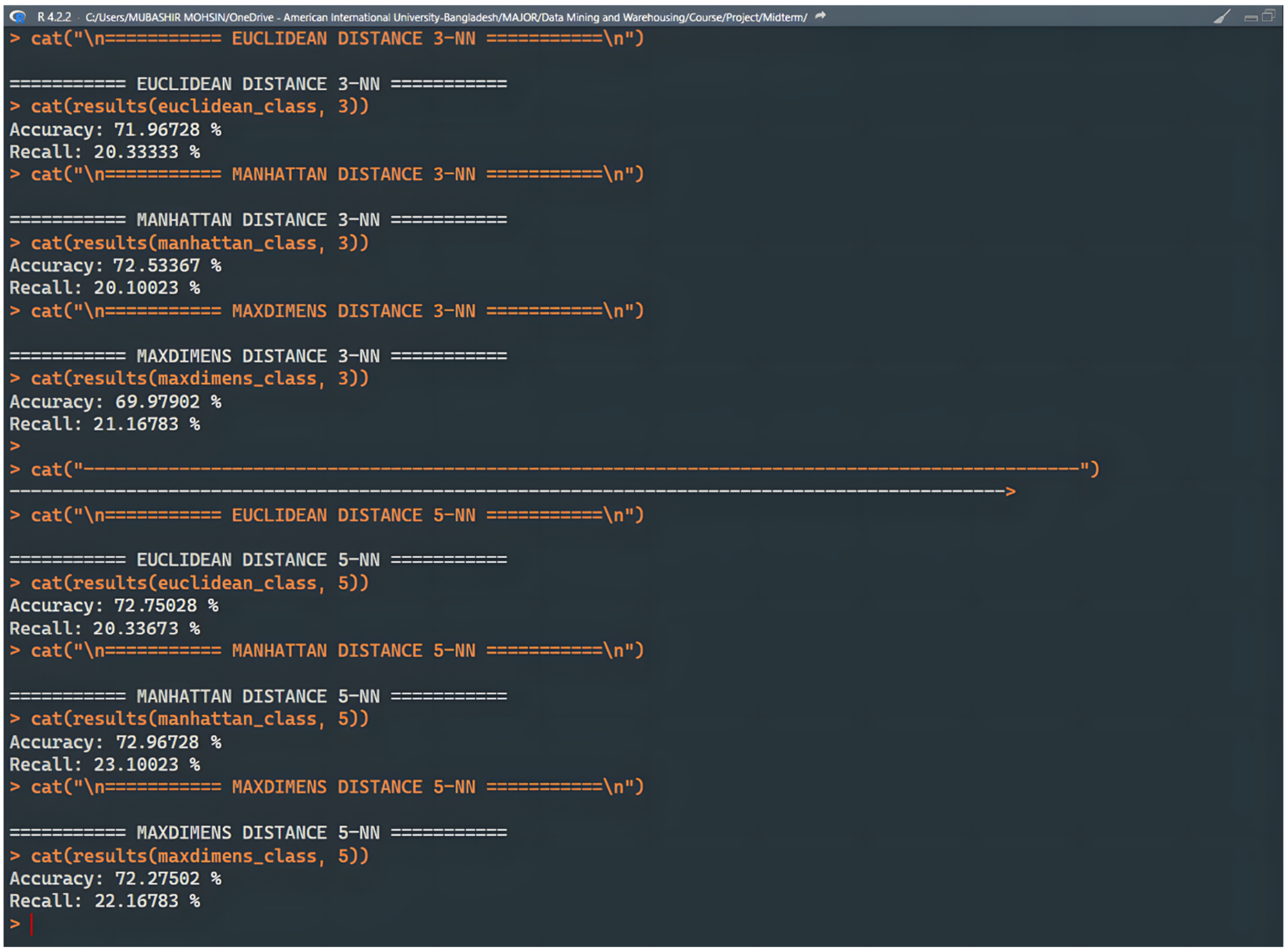
Choosing the Best Model: By examining the accuracy and recall values in the 'knn_results' data frame, it is possible to determine the best-performing model for the given dataset. The model with the highest accuracy or recall, depending on the evaluation criteria, can be selected as the optimal choice. The chosen model can be reported and further utilized for making predictions on new, unseen data.

RESULTS

After running the models for several times with random number of testing sets, the results show a decent accuracy and recall rate. The average accuracy is around 71%, while most of the time the accuracy for different distance metric and different k-values are within 68-72% of the total dataset. This is a decent accuracy percentage for a K-NN model and a pretty good one for the type of dataset that has been used. As the dataset attributes are irregular in manner and totally depends on the time-series data points, it is a good number of accuracies achieved from a K-NN model. On the other hand, for the irregularities among the data points, the recall remains a bit constant for every distance metric and k-values. The average recall rate is 24% for the all the distance metrics. The accuracy increases when the k-value increases too. Meanwhile, the recall rate increases when the k-value increases. The maximum accuracy is seen when the K-value is around 16 to 19 for all three distance metrices. The minimum recall rate is seen when the K-value is around 3 to 5. The maximum accuracy and recall rate for each distance metrices with the best k-value is given below:

Distance Metrices	K-Values	Accuracy (in %)	Recall (in %)
Euclidean Distance	19	72.087	20.188
Manhattan Distance	16	72.703	20.741
Maximum Dimension Distance	17	72.489	21.445

The below output shows the Accuracy and Recall for the k-value of 3 and 5 –


R 4.2.2 · C:/Users/MUBASHIR MOHSIN/OneDrive - American International University-Bangladesh/MAJOR/Data Mining and Warehousing/Course/Project/Midterm/

```
> cat("\n=========== EUCLIDEAN DISTANCE 3-NN ===========\n")

=========== EUCLIDEAN DISTANCE 3-NN ===========
> cat(results(euclidean_class, 3))
Accuracy: 71.96728 %
Recall: 20.33333 %
> cat("\n=========== MANHATTAN DISTANCE 3-NN ===========\n")

=========== MANHATTAN DISTANCE 3-NN ===========
> cat(results(manhattan_class, 3))
Accuracy: 72.53367 %
Recall: 20.10023 %
> cat("\n=========== MAXDIMENS DISTANCE 3-NN ===========\n")

=========== MAXDIMENS DISTANCE 3-NN ===========
> cat(results(maxdimens_class, 3))
Accuracy: 69.97902 %
Recall: 21.16783 %
>
> cat("-----------------------------------------------------------------------------------------")
---------------------------------------------------------------------------------------->
> cat("\n=========== EUCLIDEAN DISTANCE 5-NN ===========\n")

=========== EUCLIDEAN DISTANCE 5-NN ===========
> cat(results(euclidean_class, 5))
Accuracy: 72.75028 %
Recall: 20.33673 %
> cat("\n=========== MANHATTAN DISTANCE 5-NN ===========\n")

=========== MANHATTAN DISTANCE 5-NN ===========
> cat(results(manhattan_class, 5))
Accuracy: 72.96728 %
Recall: 23.10023 %
> cat("\n=========== MAXDIMENS DISTANCE 5-NN ===========\n")

=========== MAXDIMENS DISTANCE 5-NN ===========
> cat(results(maxdimens_class, 5))
Accuracy: 72.27502 %
Recall: 22.16783 %
>
```

DISCUSSION

The K-NN model exhibited promising performance on our dataset, achieving an average accuracy of around 71% and a consistent recall rate of 24%. The accuracy improved with higher K-values, reaching a maximum between 16 and 19, while the minimum recall rate was observed at K-values between 3 and 5.

Advantages:

- The K-NN model provided decent accuracy and recall rates, demonstrating its suitability for the dataset.

- The model's performance was consistent across different distance metrics, indicating robustness.

- The K-NN algorithm is relatively simple and easy to implement.

Disadvantages:

- The model's accuracy and recall rates may not be sufficient for certain high-stakes applications or datasets with strict performance requirements.

- The irregularities and time-series nature of the dataset attributes may pose challenges for the K-NN algorithm, potentially limiting its performance.

Despite these, the K-NN model offers a viable solution for the given dataset, with its simplicity, interpretability, and satisfactory performance. Further research could explore alternative algorithms or ensemble techniques to improve accuracy and address the limitations observed.

LIMITATIONS

The K-NN algorithm, despite its satisfactory performance on our dataset, has certain limitations that need to be considered.

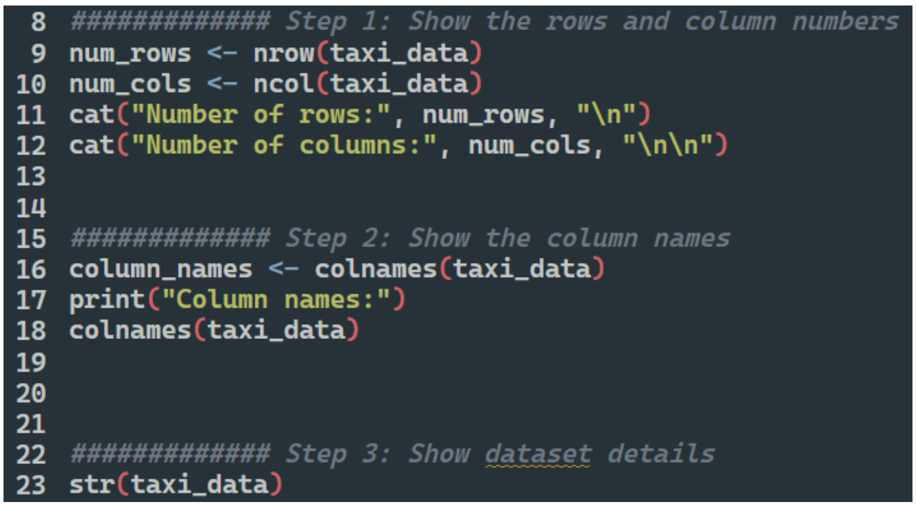
First, the choice of the optimal K-value is crucial in K-NN, as it determines the number of nearest neighbors considered for classification. The optimal K-value may vary depending on the dataset and its specific characteristics. It is essential to experiment with different K-values to find the most suitable one for a particular dataset.

Second, the K-NN algorithm assumes that similar instances have similar class labels. However, in complex datasets, this assumption may not always hold true. There can be instances that are similar in terms of their attributes but have different class labels. This can lead to misclassifications and reduced accuracy in such cases. It is important to assess the inherent structure and patterns in the data to determine whether the K-NN algorithm is appropriate.

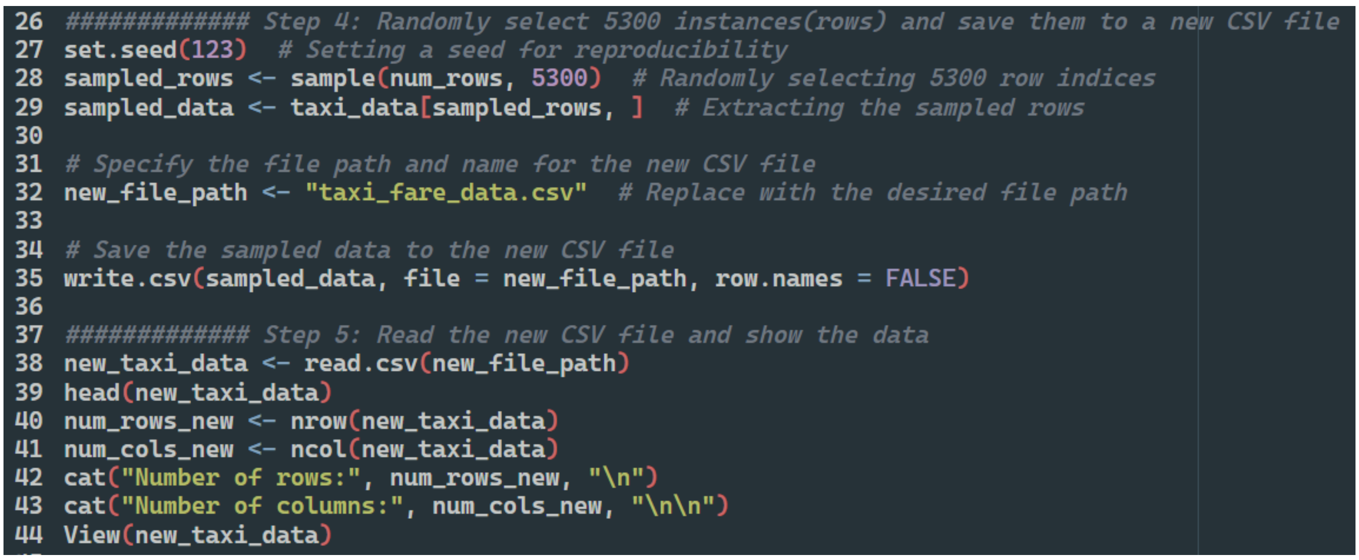
Lastly, the computational complexity of K-NN increases with the number of instances in the dataset. As the number of instances grows, the algorithm needs to calculate distances and compare each instance to all others, resulting in increased computation time. This makes K-NN less efficient for large datasets with a substantial number of instances. Consideration should be given to the computational resources available and the scalability of the algorithm when applying K-NN to large datasets.

CODE SNIPPETS

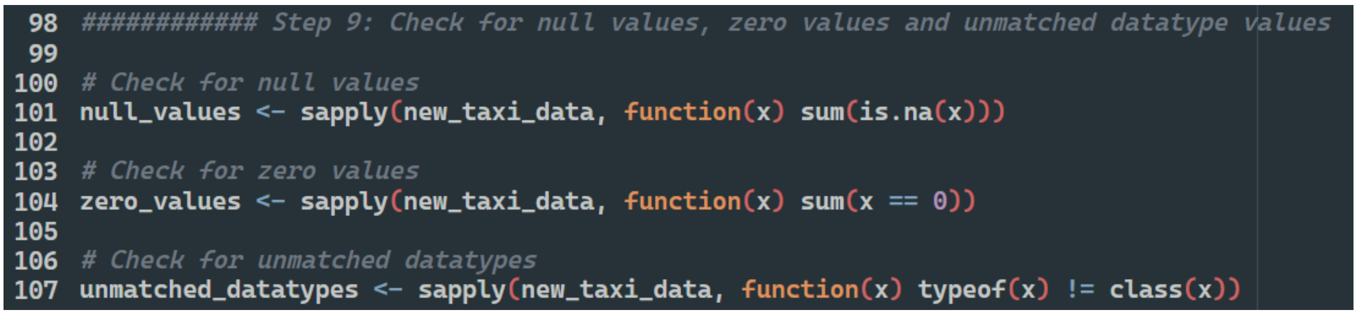
Pre-processing steps-



```r
############# Step 1: Show the rows and column numbers
num_rows <- nrow(taxi_data)
num_cols <- ncol(taxi_data)
cat("Number of rows:", num_rows, "\n")
cat("Number of columns:", num_cols, "\n\n")


############# Step 2: Show the column names
column_names <- colnames(taxi_data)
print("Column names:")
colnames(taxi_data)



############# Step 3: Show dataset details
str(taxi_data)
```

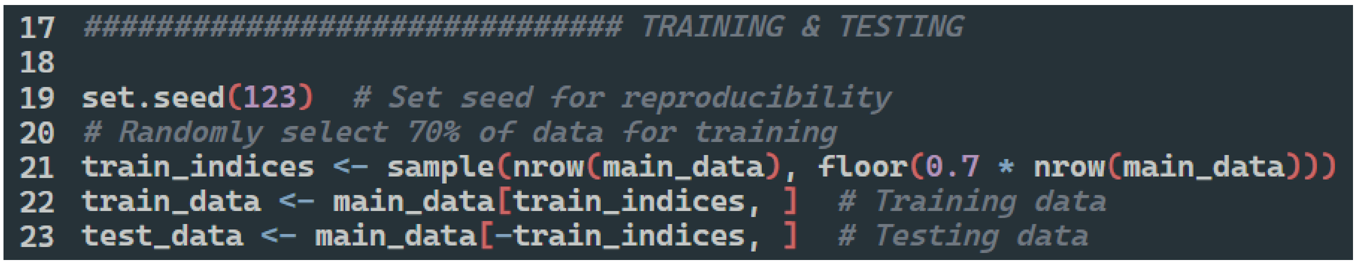


```r
############# Step 4: Randomly select 5300 instances(rows) and save them to a new CSV file
set.seed(123)  # Setting a seed for reproducibility
sampled_rows <- sample(num_rows, 5300)  # Randomly selecting 5300 row indices
sampled_data <- taxi_data[sampled_rows, ]  # Extracting the sampled rows

# Specify the file path and name for the new CSV file
new_file_path <- "taxi_fare_data.csv"  # Replace with the desired file path

# Save the sampled data to the new CSV file
write.csv(sampled_data, file = new_file_path, row.names = FALSE)

############# Step 5: Read the new CSV file and show the data
new_taxi_data <- read.csv(new_file_path)
head(new_taxi_data)
num_rows_new <- nrow(new_taxi_data)
num_cols_new <- ncol(new_taxi_data)
cat("Number of rows:", num_rows_new, "\n")
cat("Number of columns:", num_cols_new, "\n\n")
View(new_taxi_data)
```

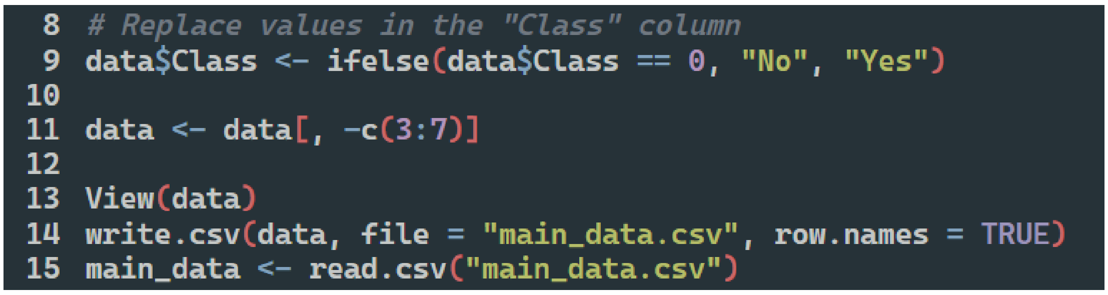


```r
############ Step 9: Check for null values, zero values and unmatched datatype values

# Check for null values
null_values <- sapply(new_taxi_data, function(x) sum(is.na(x)))

# Check for zero values
zero_values <- sapply(new_taxi_data, function(x) sum(x == 0))

# Check for unmatched datatypes
unmatched_datatypes <- sapply(new_taxi_data, function(x) typeof(x) != class(x))
```

Data segmentation steps-

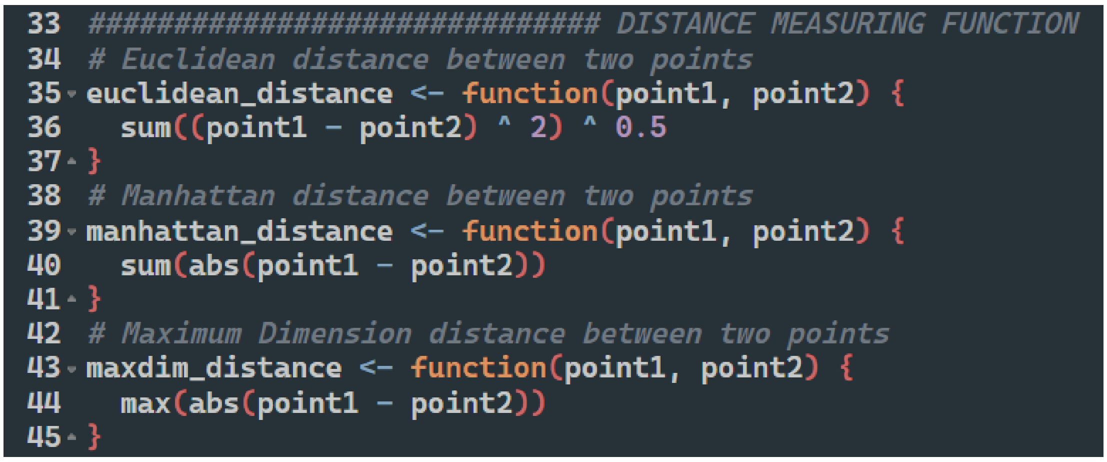


```r
################################## TRAINING & TESTING

set.seed(123)  # Set seed for reproducibility
# Randomly select 70% of data for training
train_indices <- sample(nrow(main_data), floor(0.7 * nrow(main_data)))
train_data <- main_data[train_indices, ]  # Training data
test_data <- main_data[-train_indices, ]  # Testing data
```



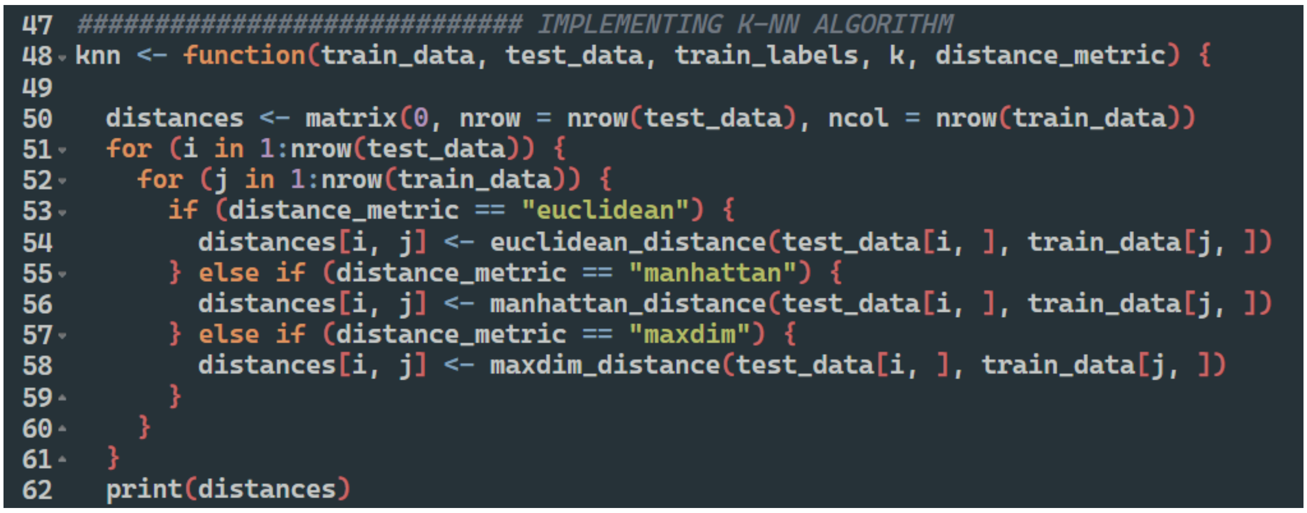
```r
# Replace values in the "Class" column
data$Class <- ifelse(data$Class == 0, "No", "Yes")

data <- data[, -c(3:7)]

View(data)
write.csv(data, file = "main_data.csv", row.names = TRUE)
main_data <- read.csv("main_data.csv")
```

Distance Measuring Functions-



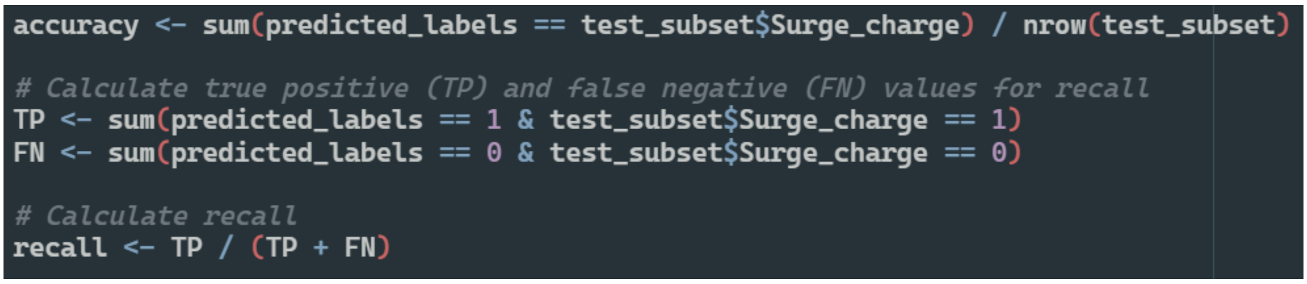
```r
################################ DISTANCE MEASURING FUNCTION
# Euclidean distance between two points
euclidean_distance <- function(point1, point2) {
  sum((point1 - point2) ^ 2) ^ 0.5
}
# Manhattan distance between two points
manhattan_distance <- function(point1, point2) {
  sum(abs(point1 - point2))
}
# Maximum Dimension distance between two points
maxdim_distance <- function(point1, point2) {
  max(abs(point1 - point2))
}
```

K-NN Implementation-



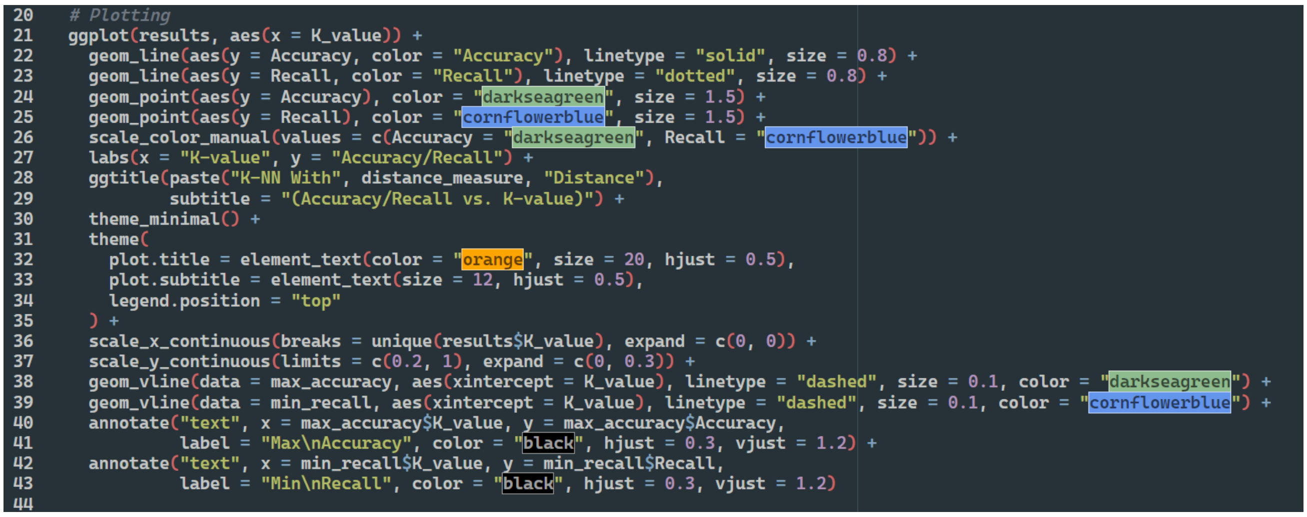
```r
############################## IMPLEMENTING K-NN ALGORITHM
knn <- function(train_data, test_data, train_labels, k, distance_metric) {

  distances <- matrix(0, nrow = nrow(test_data), ncol = nrow(train_data))
  for (i in 1:nrow(test_data)) {
    for (j in 1:nrow(train_data)) {
      if (distance_metric == "euclidean") {
        distances[i, j] <- euclidean_distance(test_data[i, ], train_data[j, ])
      } else if (distance_metric == "manhattan") {
        distances[i, j] <- manhattan_distance(test_data[i, ], train_data[j, ])
      } else if (distance_metric == "maxdim") {
        distances[i, j] <- maxdim_distance(test_data[i, ], train_data[j, ])
      }
    }
  }
  print(distances)
```

Accuracy and Recall Calculation-



```r
accuracy <- sum(predicted_labels == test_subset$Surge_charge) / nrow(test_subset)

# Calculate true positive (TP) and false negative (FN) values for recall
TP <- sum(predicted_labels == 1 & test_subset$Surge_charge == 1)
FN <- sum(predicted_labels == 0 & test_subset$Surge_charge == 0)

# Calculate recall
recall <- TP / (TP + FN)
```

Plotting Results-



```
# Plotting
ggplot(results, aes(x = K_value)) +
  geom_line(aes(y = Accuracy, color = "Accuracy"), linetype = "solid", size = 0.8) +
  geom_line(aes(y = Recall, color = "Recall"), linetype = "dotted", size = 0.8) +
  geom_point(aes(y = Accuracy), color = "darkseagreen", size = 1.5) +
  geom_point(aes(y = Recall), color = "cornflowerblue", size = 1.5) +
  scale_color_manual(values = c(Accuracy = "darkseagreen", Recall = "cornflowerblue")) +
  labs(x = "K-value", y = "Accuracy/Recall") +
  ggtitle(paste("K-NN With", distance_measure, "Distance"),
          subtitle = "(Accuracy/Recall vs. K-value)") +
  theme_minimal() +
  theme(
    plot.title = element_text(color = "orange", size = 20, hjust = 0.5),
    plot.subtitle = element_text(size = 12, hjust = 0.5),
    legend.position = "top"
  ) +
  scale_x_continuous(breaks = unique(results$K_value), expand = c(0, 0)) +
  scale_y_continuous(limits = c(0.2, 1), expand = c(0, 0.3)) +
  geom_vline(data = max_accuracy, aes(xintercept = K_value), linetype = "dashed", size = 0.1, color = "darkseagreen") +
  geom_vline(data = min_recall, aes(xintercept = K_value), linetype = "dashed", size = 0.1, color = "cornflowerblue") +
  annotate("text", x = max_accuracy$K_value, y = max_accuracy$Accuracy,
           label = "Max\nAccuracy", color = "black", hjust = 0.3, vjust = 1.2) +
  annotate("text", x = min_recall$K_value, y = min_recall$Recall,
           label = "Min\nRecall", color = "black", hjust = 0.3, vjust = 1.2)

```

VISUALIZATION

Performance of K-NN with different distance metrices, Accuracy/Recall vs. K-Value graph-

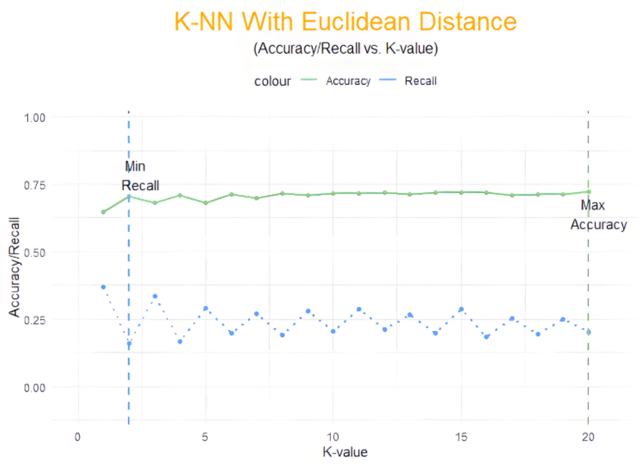


Fig: K-NN with Euclidean Distance

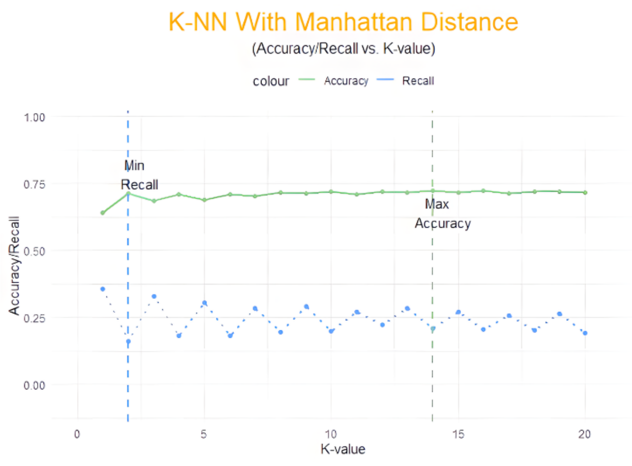


Fig: K-NN with Manhattan Distance

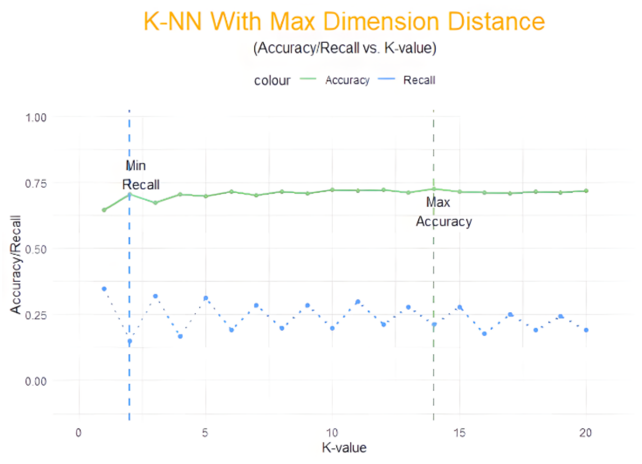


Fig: K-NN with Maximum Dimension Distance