American International University

Data Warehousing & Data Mining (report)

Title: House rent, area Data Clustering using KNN

Name: Sharia Arfin Tanim

ID: 20-42096-1

SEC: A

**Submitted To**

Dr. Akinul Islam Jony

**Project Overview/Discussion:** In this project, basic data analysis and preprocessing was conducted on selected dataset and K nearest neighbors algorithm was applied on the dataset using R programming language.

**Dataset Overview:** The dataset consists of several housing properties, each represented by a row. The target variable is the price, and the other features are used as predictors to build a model that can predict the price of a property based on its characteristics.

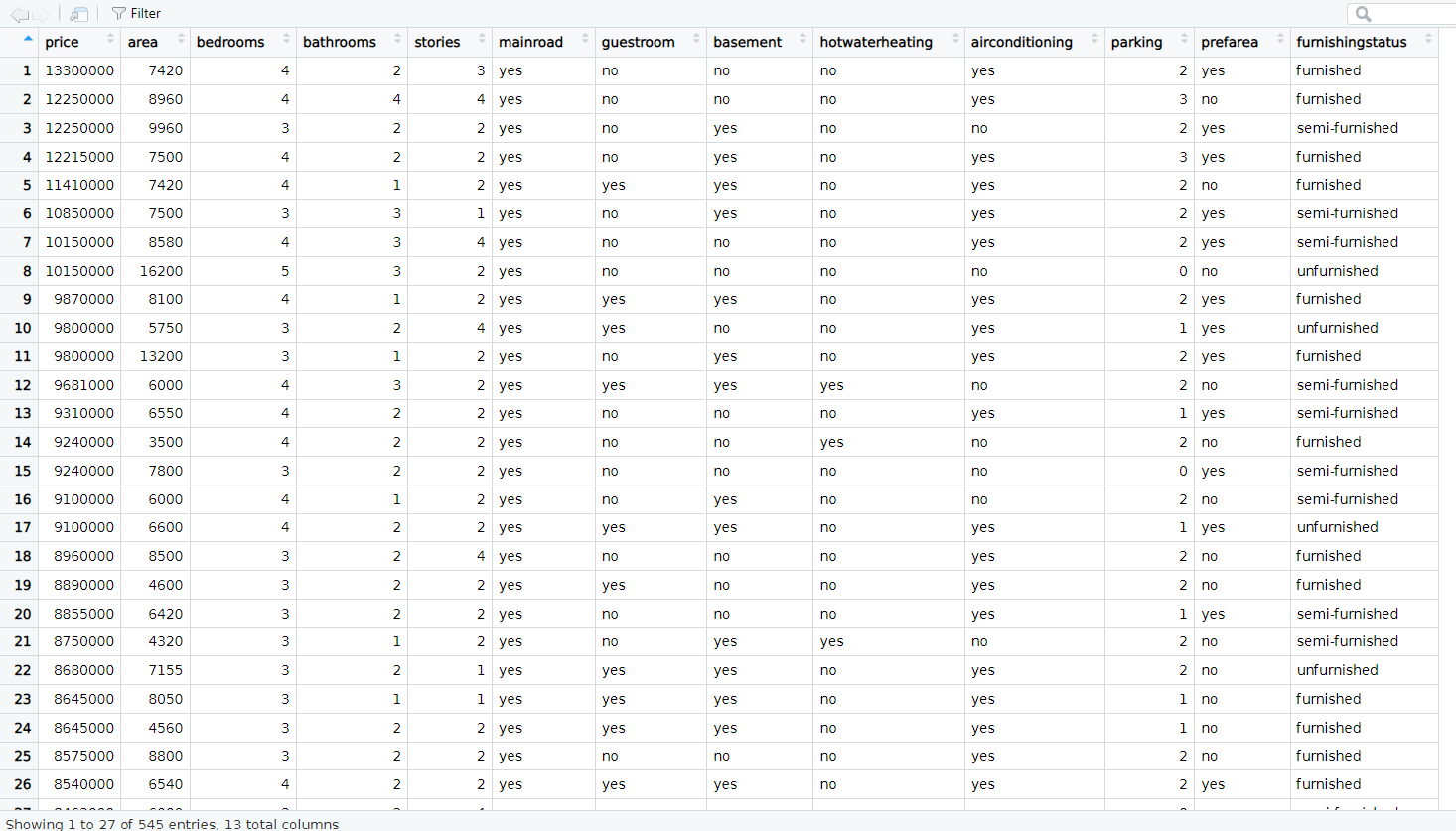
<https://www.kaggle.com/datasets/harishkumardatalab/housing-price-prediction?resource=download>

The dataset appears to be a mix of numerical and categorical features, and it is used for regression or predictive modeling tasks related to housing prices.

**ATTRIBUTE INFORMATION:** Input variables

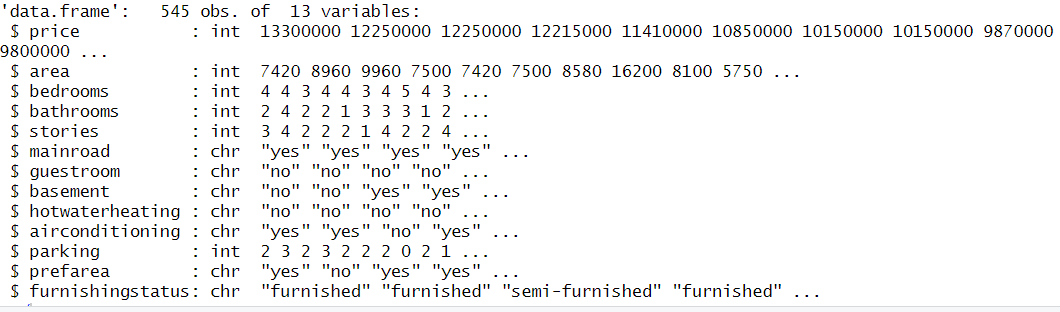
1. **Price:** The price of the house.
2. **Area:** The total area of the house in square feet.
3. **Bedrooms:** The number of bedrooms in the house.
4. **Bathrooms:** The number of bathrooms in the house.
5. **Stories:** The number of stories in the house.
6. **Mainroad:** Whether the house is connected to the main road (Yes/No).
7. **Guestroom:** Whether the house has a guest room (Yes/No).
8. **Basement:** Whether the house has a basement (Yes/No).
9. **Hot water heating:** Whether the house has a hot water heating system (Yes/No).
10. **Airconditioning:** Whether the house has an air conditioning system (Yes/No).
11. **Parking:** The number of parking spaces available within the house.
12. **Prefarea:** Whether the house is located in a preferred area (Yes/No).
13. **Furnishing status:** The furnishing status of the house (Fully Furnished, Semi-Furnished, Unfurnished).
14. **Importing or read the Dataset**

Code: dataset <- read.csv("C:/Users/Asus/Downloads/archive/Housing.csv", header = TRUE)



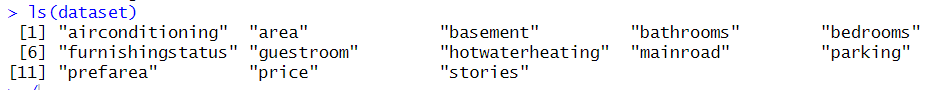
1. **Structure of the Dataset:**

Code: str(dataset)



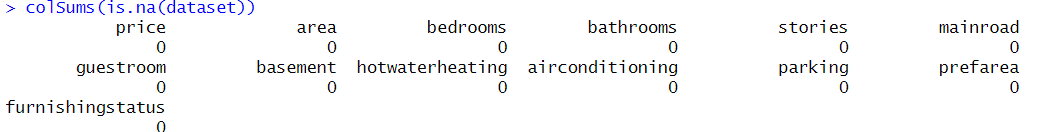
**3.Attributes**

Code:



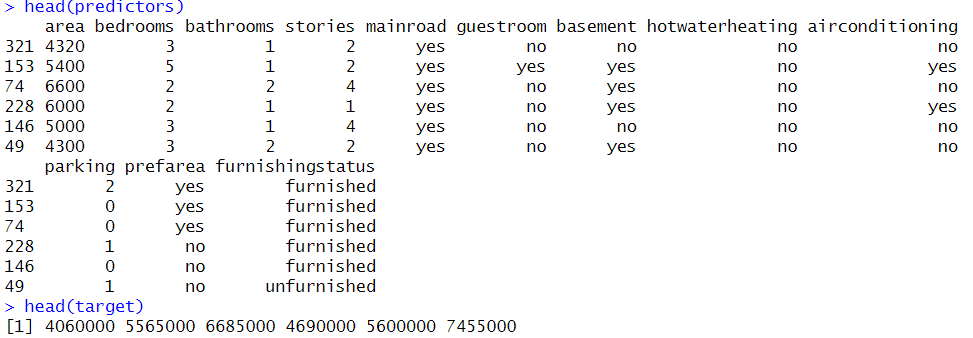
**4.Check for missing values:**

Code:



Set the predict variables and the target variable:

Code:



**6. Split the data into training and testing sets:**

split into training and testing sets using a 80:20 ratio

# Calculate the number of rows for training and testing

n\_total <- nrow(dataset)

n\_train <- round(0.8 \* n\_total)

n\_test <- n\_total - n\_train

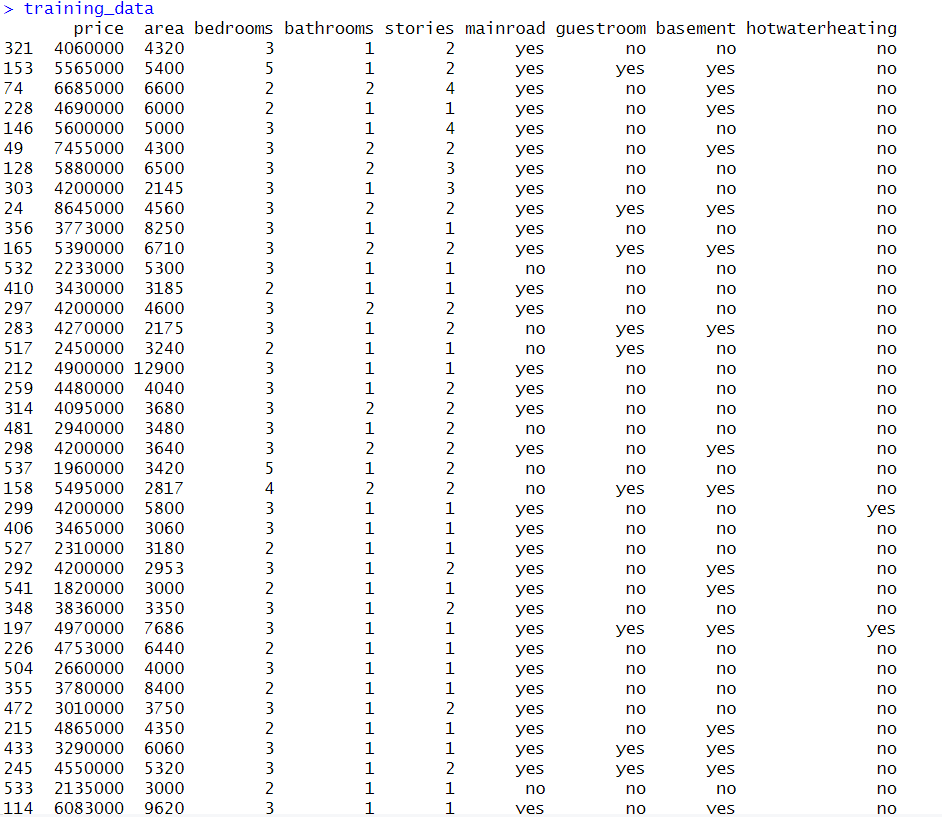
# Randomly shuffle the rows of the dataset

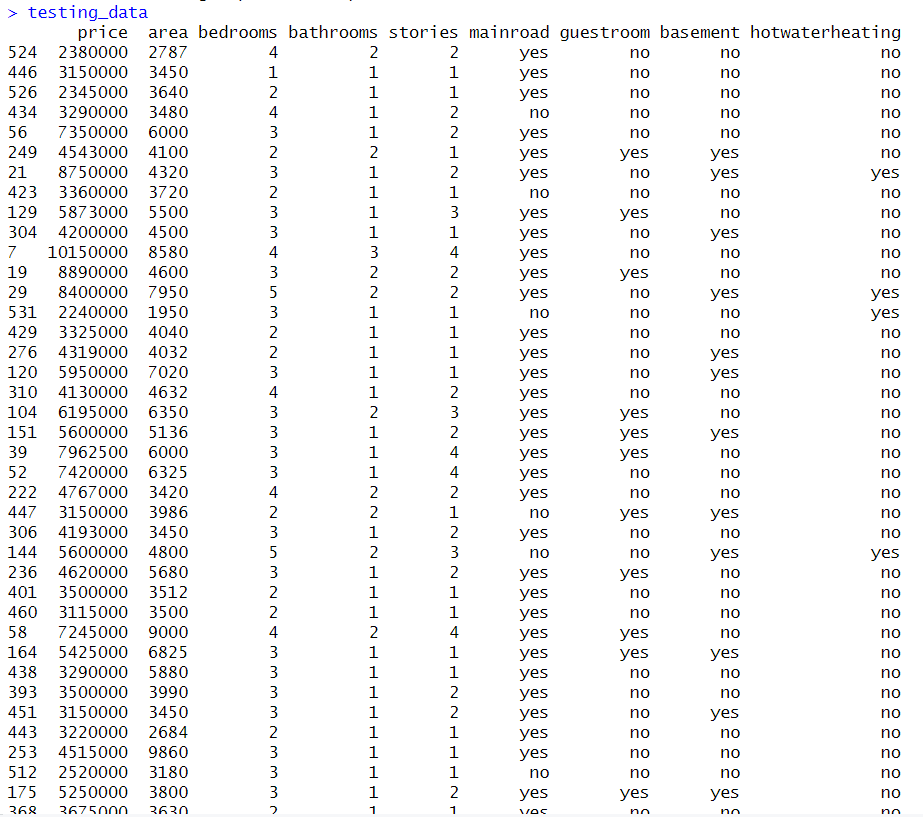
my\_dataset <- dataset[sample(n\_total), ]

# Split the dataset into training and testing sets

training\_data <- my\_dataset[1:n\_train, ]

testing\_data <- my\_dataset[(n\_train + 1):(n\_train + n\_test), ]





**7. implementing a k nearest neighbors and setting the values of k:**

knn\_with\_distance\_measure <- function(train\_data, test\_data, train\_labels, k, distance\_function) {

n\_train <- nrow(train\_data)

n\_test <- nrow(test\_data)

predicted\_labels <- character(n\_test)

for (i in 1:n\_test) {

distances <- numeric(n\_train)

for (j in 1:n\_train) {

distances[j] <- distance\_function(test\_data[i, ], train\_data[j, ])

}

sorted\_indices <- order(distances)

k\_indices <- sorted\_indices[1:k]

k\_nearest\_labels <- train\_labels[k\_indices]

predicted\_labels[i] <- names(sort(table(k\_nearest\_labels), decreasing = TRUE)[1])

}

return(predicted\_labels)

}

**8. Initialize vectors to store accuracies:**



**9.set the value of k and Apply k-NN for each k value and distance measure Apply k-NN with Euclidean distance, Manhattan distance & maximum distance:**

k\_values <- c(3, 5, 7)

for (k in k\_values) {

# Euclidean distance

euclidean\_predictions <- knn\_with\_distance\_measure(train\_data, test\_data, train\_labels, k, distance\_measure = "euclidean")

accuracy\_euclidean <- sum(euclidean\_predictions == test\_labels) / length(test\_labels)

cat("Accuracy for k =", k, "with Euclidean Distance:", accuracy\_euclidean, "\n")

# Manhattan distance

manhattan\_predictions <- knn\_with\_distance\_measure(train\_data, test\_data, train\_labels, k, distance\_measure = "manhattan")

accuracy\_manhattan <- sum(manhattan\_predictions == test\_labels) / length(test\_labels)

cat("Accuracy for k =", k, "with Manhattan Distance:", accuracy\_manhattan, "\n")

# Maximum distance

maximum\_predictions <- knn\_with\_distance\_measure(train\_data, test\_data, train\_labels, k, distance\_measure = "maximum")

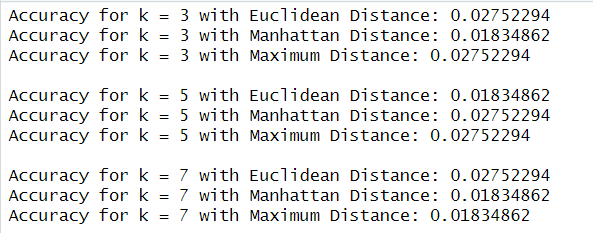
accuracy\_maximum <- sum(maximum\_predictions == test\_labels) / length(test\_labels)

cat("Accuracy for k =", k, "with Maximum Distance:", accuracy\_maximum, "\n")

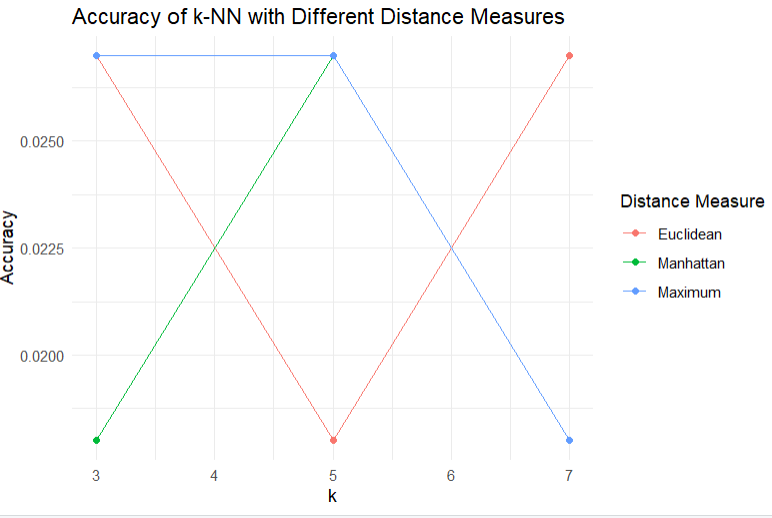
Print the accuracy for the current k value:

accuracies <- c(accuracies, accuracy\_euclidean, accuracy\_manhattan, accuracy\_maximum)

cat("\n")



**10. Create a data frame for accuracies and plotting**



**11.conclusion**: The graph above shows how the accuracy of the knn algorithm varies with different values of k on using the 3 distance measuring methods (Euclidean, Manhattan, and Maximum Dimension). For all k value = 3,5,7 Maximum distance had the highest accuracy.