



**Department of Computer Science
American International University-Bangladesh**

Course Name: DATA WAREHOUSING AND DATA MINING

“Project on Classification of Diamond Evaluation using K-Nearest Neighbor (k-NN)”

Section: C

Supervised By:

Dr. Akinul Islam Jony

Associate Professor & Head-In-Charge [Undergraduate Program],
Computer Science

Submitted By:

Maisha Shams (20-43359-1)

Sumaiya Malik (20-43688-2)

Soily Ghosh Sneha (20-43702-2)

Tazin Alam (20-43848-2)

Submission Date: July 4, 2023.

Project Title:

1. Introduction:

This project revolves around a comprehensive dataset comprising the prices and various attributes of a vast collection of round cut diamonds. With close to 6,000 diamond entries, this dataset provides valuable insights into the factors influencing diamond prices. The dataset includes information such as price, carat weight, cut quality, dimensions (length, width, depth), total depth percentage, and table width.

Understanding the nuances that determine diamond prices is of immense interest to diamond traders, jewelers, and consumers alike. By exploring this dataset, we can gain valuable knowledge about the relationships between the diamond's physical attributes and its market value. The dataset encompasses a wide range of diamond prices, spanning from \$326 to \$18,823. Each diamond's carat weight is provided, representing the diamond's size, and the cut quality is categorized into five levels: Fair, Good, Very Good, Premium, and Ideal. Additionally, the dataset includes the dimensions of the diamonds, with length (x), width (y), and depth (z) measured in millimeters. The depth percentage is derived from the depth and the average of the length and width, and the table width represents the width of the diamond's top relative to its widest point.

By exploring the relationships between these variables, we can uncover patterns and insights into how factors such as carat weight, cut quality, color grade, clarity grade, and dimensions impact diamond prices. This analysis can aid in making informed decisions regarding diamond purchases, pricing strategies, and market trends.

In this project, we will delve into the dataset, employing various statistical and visual analysis techniques to explore the relationships and patterns within the data. By doing so, we aim to gain a deeper understanding of the diamond market and its pricing dynamics.

There are 3 popular classification models in Data Mining- Naïve Bayes, K- Nearest Neighbor (k-NN) and Decision Tree. These are some popular classification algorithms that can be used for classifications but for our project we will be only using K-Nearest Neighbor (k-NN).

2. Project Overview:

2.1 Project Objective:

This project involves analyzing a dataset of over 6,000 round cut diamonds to understand the factors that influence their prices. The goal is to develop a predictive model using the k-Nearest Neighbors (k-NN) algorithm. By considering attributes such as carat weight, depth, table and dimensions, the model will estimate diamond cut qualities. The project outcome includes an accurate predictive model, insights into attribute importance, and improved decision-making for diamond traders and consumers in pricing, valuation, market trends and the quality of diamonds.

2.2 Description:

The dataset includes various attributes such as price, carat weight, cut quality, dimensions (length, width, depth), total depth percentage, and table width. The price of each diamond is provided in US dollars, ranging from \$326 to \$18,823. We will focus on predicting diamond cut quality based on these attributes using the k-NN algorithm.

Our approach involves several steps. First, we will preprocess the dataset by exploring and cleaning it, handling missing values, and transforming categorical variables into numerical representations suitable for the k-NN algorithm. Next, we will select the most relevant features that strongly influence diamond cut, considering correlations and conducting feature importance analysis. To evaluate the performance of our model accurately, we will split the dataset into training and testing sets. Additionally, we will apply scaling and normalization techniques to ensure that all attributes are on a similar scale, as the k-NN algorithm heavily relies on distance calculations. Implementing the k-NN algorithm, we will measure the similarity between diamonds using a distance metric such as Euclidean distance and Manhattan distance. By considering the attributes of the k nearest neighbors, we will predict the price of a given diamond. The model's performance will be evaluated through metrics such as accuracy, precision, recall, and comparing the predicted quality with the actual quality in the test set.

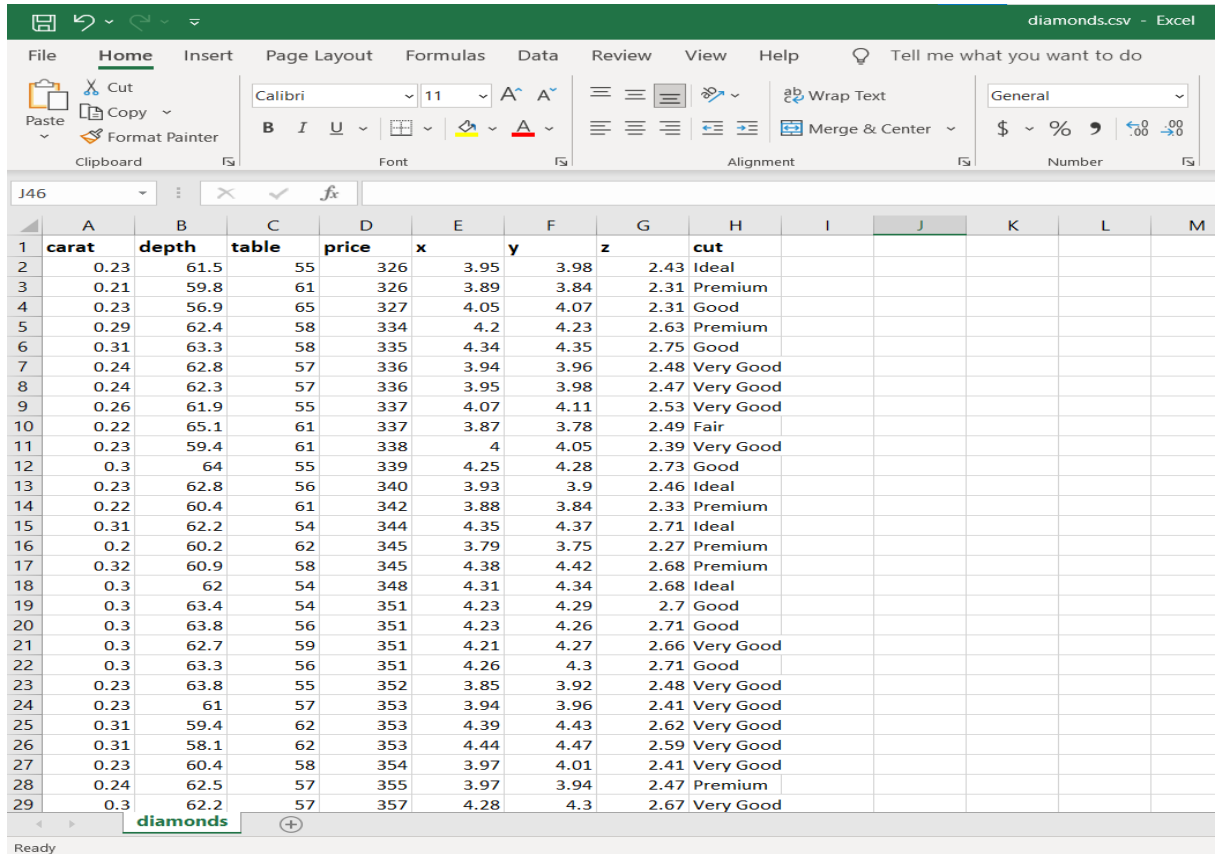
By analyzing the results and examining the attribute weights assigned by the k-NN model, we will gain insights into the relative importance of different diamond characteristics in determining their cut quality. This analysis will provide valuable information for diamond traders, jewelers, and consumers, aiding in decision-making related to pricing, valuation, and market trends.

2.3 Outcome of the Project:

The outcome of this project is a developed predictive model using the k-Nearest Neighbors (k-NN) algorithm that can accurately estimate the prices of round cut diamonds based on their attributes. This model provides a valuable tool for diamond traders, jewelers, and consumers to make informed decisions regarding diamond purchases and sales. By considering attributes such as carat weight, cut quality and dimensions, the model generates accurate cut predictions. Additionally, the project yields insights into the relative importance of different attributes in determining diamond cut, enabling a deeper understanding of the market dynamics. This information empowers diamond industry professionals to adjust pricing strategies, identify opportunities for value optimization, and stay competitive in a rapidly changing market. Ultimately, the project's outcome contributes to enhancing decision-making processes and improving market insights within the diamond industry.

3. Data Set:

3.1 The Real Data Set:



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	carat	depth	table	price	x	y	z	cut					
2	0.23	61.5	55	326	3.95	3.98	2.43	Ideal					
3	0.21	59.8	61	326	3.89	3.84	2.31	Premium					
4	0.23	56.9	65	327	4.05	4.07	2.31	Good					
5	0.29	62.4	58	334	4.2	4.23	2.63	Premium					
6	0.31	63.3	58	335	4.34	4.35	2.75	Good					
7	0.24	62.8	57	336	3.94	3.96	2.48	Very Good					
8	0.24	62.3	57	336	3.95	3.98	2.47	Very Good					
9	0.26	61.9	55	337	4.07	4.11	2.53	Very Good					
10	0.22	65.1	61	337	3.87	3.78	2.49	Fair					
11	0.23	59.4	61	338	4	4.05	2.39	Very Good					
12	0.3	64	55	339	4.25	4.28	2.73	Good					
13	0.23	62.8	56	340	3.93	3.9	2.46	Ideal					
14	0.22	60.4	61	342	3.88	3.84	2.33	Premium					
15	0.31	62.2	54	344	4.35	4.37	2.71	Ideal					
16	0.2	60.2	62	345	3.79	3.75	2.27	Premium					
17	0.32	60.9	58	345	4.38	4.42	2.68	Premium					
18	0.3	62	54	348	4.31	4.34	2.68	Ideal					
19	0.3	63.4	54	351	4.23	4.29	2.7	Good					
20	0.3	63.8	56	351	4.23	4.26	2.71	Good					
21	0.3	62.7	59	351	4.21	4.27	2.66	Very Good					
22	0.3	63.3	56	351	4.26	4.3	2.71	Good					
23	0.23	63.8	55	352	3.85	3.92	2.48	Very Good					
24	0.23	61	57	353	3.94	3.96	2.41	Very Good					
25	0.31	59.4	62	353	4.39	4.43	2.62	Very Good					
26	0.31	58.1	62	353	4.44	4.47	2.59	Very Good					
27	0.23	60.4	58	354	3.97	4.01	2.41	Very Good					
28	0.24	62.5	57	355	3.97	3.94	2.47	Premium					
29	0.3	62.2	57	357	4.28	4.3	2.67	Very Good					

Figure 1: The CSV file of the dataset of Diamond.

3.2 Source:

Site: https://rpubs.com/GinaMoreno/course1_4

3.3 Details:

1. Number of Instances: 6000

2. Number of Attributes: 8

3. Missing Attribute Values: None

4. Class Values: Ideal, Premium, Good, Very Good, Fair

5. Attributes: Carat, Color, Clarity, Depth, Table, Price, X, Y, Z and Cut.

K-Nearest Neighbor:

K-Nearest Neighbors (k-NN) is a simple yet effective machine learning algorithm used for classification and regression tasks. It operates on the principle of similarity, where the prediction for a new data point is based on the majority vote (for classification) or averaging (for regression) of its k nearest neighbors in the training dataset. The algorithm does not make any assumptions about the data distribution and can handle both numerical and categorical data. However, choosing the right value for k and distance metric is crucial for optimal performance. While k-NN is easy to implement, it may suffer from the curse of dimensionality and requires careful data preprocessing. Overall, k-NN is a versatile algorithm that can provide valuable insights and predictions in various applications.

ADVANTAGES:

- 1. Easy implementation:** K-NN is a straightforward technique that does not require model creation or training, making it simple to utilize.
- 2. Non-parametric:** Because K-NN is non-parametric, it does not presume a particular functional form for the distribution of the underlying data.
- 3. High accuracy:** K-NN can frequently offer high accuracy, particularly when the data set is substantial and varied.
- 4. Flexibility:** K-NN is adaptable to a variety of data formats, including numerical, categorical, or binary data, and may be utilized for both classification and regression problems.

DISADVANTAGES:

- 1. Requires a lot of computation:** K-NN is a lazy learning method that skips the model-building stage.
- 2. Noise-sensitive:** K-NN is noise-sensitive to outliers, noisy data, and irrelevant features. This may skew the distance measurement and produce subpar results.
- 3. Selecting the appropriate value for K:** K-NN algorithm performance depends on selecting the appropriate value for K, which denotes the number of nearest neighbors to take into account. A low K number can result in overfitting, whereas a high K value can result in underfitting.
- 4. Unbalanced data:** K-NN assumes that each feature is equally important, which may not hold with unbalanced datasets.

THE PSEUDOCODE FOR THIS ALGORITHM:

1. Insert the test and training data.

2. Choose K's value.

3. For every test data point:

-calculate the Euclidean distance between all of the training data points

- list-store and sort the Euclidean distances

- choose the top k points.

- depending on the majority of classes present in the selected points, assign a class to the test point.

4. End.

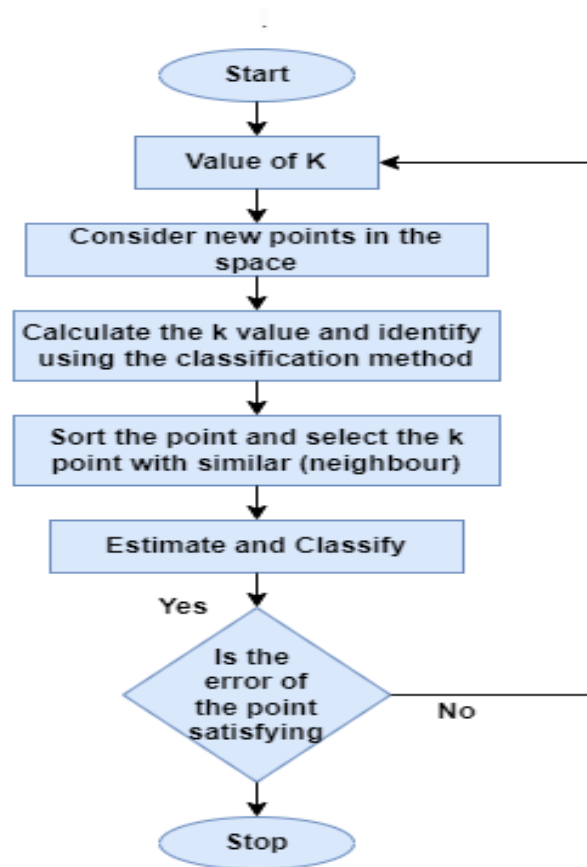


Figure 2: Flowchart of K-NN.

As mentioned in the above figure -1 the description of the classification method is given where the new point in the space is identified and is estimated along with a few similar points already available in the space.

	carat	depth	table	price	x	y	z	cut
1	0.23	61.5	55.0	326	3.95	3.98	2.43	Ideal
2	0.21	59.8	61.0	326	3.89	3.84	2.31	Premium
3	0.23	56.9	65.0	327	4.05	4.07	2.31	Good
4	0.29	62.4	58.0	334	4.20	4.23	2.63	Premium
5	0.31	63.3	58.0	335	4.34	4.35	2.75	Good
6	0.24	62.8	57.0	336	3.94	3.96	2.48	Very Good
7	0.24	62.3	57.0	336	3.95	3.98	2.47	Very Good
8	0.26	61.9	55.0	337	4.07	4.11	2.53	Very Good
9	0.22	65.1	61.0	337	3.87	3.78	2.49	Fair
10	0.23	59.4	61.0	338	4.00	4.05	2.39	Very Good
11	0.30	64.0	55.0	339	4.25	4.28	2.73	Good
12	0.23	62.8	56.0	340	3.93	3.90	2.46	Ideal
13	0.22	60.4	61.0	342	3.88	3.84	2.33	Premium
14	0.31	62.2	54.0	344	4.35	4.37	2.71	Ideal
15	0.20	60.2	62.0	345	3.79	3.75	2.27	Premium
16	0.32	60.9	58.0	345	4.38	4.42	2.68	Premium
17	0.30	62.0	54.0	348	4.31	4.34	2.68	Ideal
18	0.30	63.4	54.0	351	4.23	4.29	2.70	Good
19	0.30	63.8	56.0	351	4.23	4.26	2.71	Good
20	0.30	62.7	59.0	351	4.21	4.27	2.66	Very Good
21	0.30	63.3	56.0	351	4.26	4.30	2.71	Good

Showing 1 to 21 of 6,000 entries, 8 total columns

Figure 3: After inserting the dataset of diamond cut in R studio.

Using Different values of k for K-NN:

Code:

```
data = read.csv('D:/Sumaiya Malik/New Desktop/Data Mining/project/diamonds.csv')
```

```
ran <- sample(1:nrow(data),0.7 * nrow(data))
```

```
normalization <-function(x) { (x -min(x))/(max(x)-min(x)) }
```

```
data_normalization <- as.data.frame(lapply(data[,c(1,2,3,4,5,6,7)], normalization))
```

```
data_train <- data_normalization[ran,]
```

```
data_test <- data_normalization[-ran,]
```

```
data_target <- as.factor(data[ran,2])
```

```
test_target <- as.factor(data[-ran,2])

library(class)

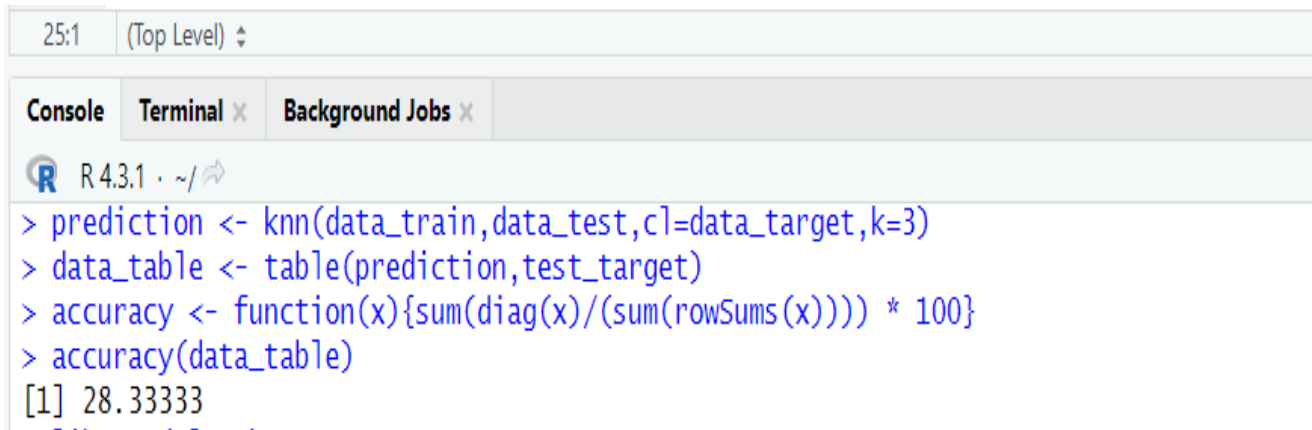
prediction <- knn(data_train,data_test,cl=data_target,k=3)

data_table <- table(prediction,test_target)

accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}

accuracy(data_table)
```

Using the value k=3:

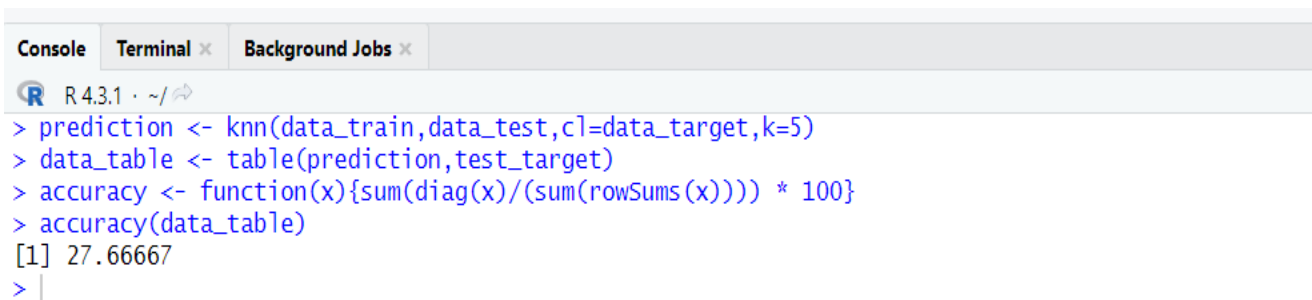


The screenshot shows an R console window with the following code and output:

```
> prediction <- knn(data_train,data_test,cl=data_target,k=3)
> data_table <- table(prediction,test_target)
> accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
> accuracy(data_table)
[1] 28.33333
```

Figure 4: The accuracy is 28.3333%, when k=3.

Using the value k=5:

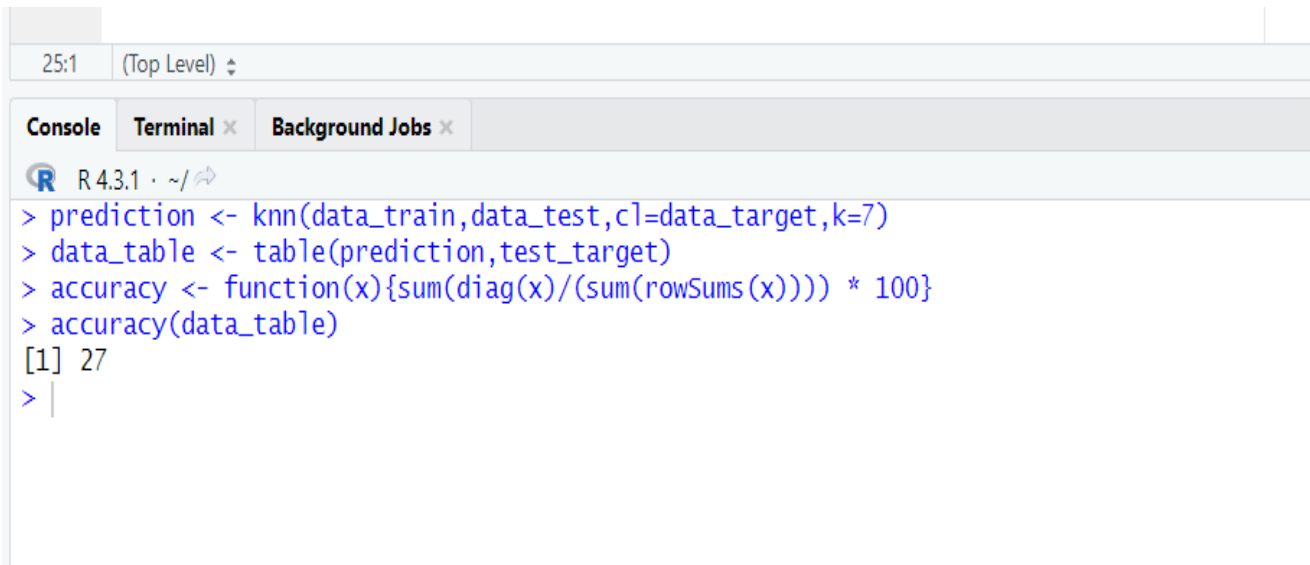


The screenshot shows an R console window with the following code and output:

```
> prediction <- knn(data_train,data_test,cl=data_target,k=5)
> data_table <- table(prediction,test_target)
> accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
> accuracy(data_table)
[1] 27.66667
> |
```

Figure 5: The accuracy is 27.6667%, when k=5.

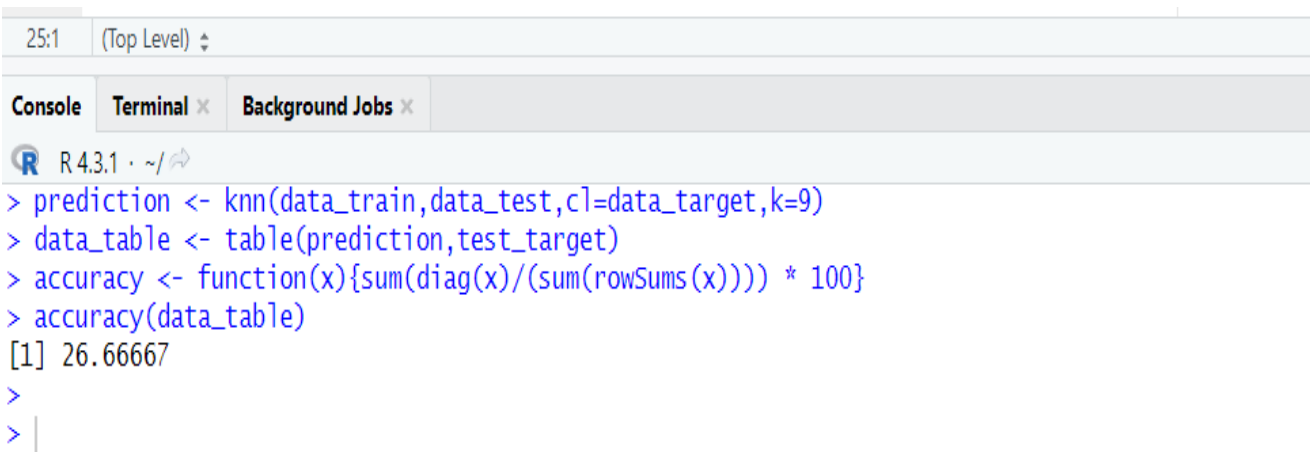
Using the value k=7:

A screenshot of an R console window. The title bar shows '25:1' and '(Top Level)'. Below the title bar are three tabs: 'Console', 'Terminal', and 'Background Jobs'. The 'Console' tab is active, showing the R prompt 'R 4.3.1 · ~/'. The code entered is: > prediction <- knn(data_train,data_test,cl=data_target,k=7), > data_table <- table(prediction,test_target), > accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}, > accuracy(data_table). The output is [1] 27.

```
25:1 (Top Level)
Console Terminal Background Jobs
R 4.3.1 · ~/
> prediction <- knn(data_train,data_test,cl=data_target,k=7)
> data_table <- table(prediction,test_target)
> accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
> accuracy(data_table)
[1] 27
> |
```

Figure 6: The accuracy is 27%, when k=7.

Using the value k=9:

A screenshot of an R console window. The title bar shows '25:1' and '(Top Level)'. Below the title bar are three tabs: 'Console', 'Terminal', and 'Background Jobs'. The 'Console' tab is active, showing the R prompt 'R 4.3.1 · ~/'. The code entered is: > prediction <- knn(data_train,data_test,cl=data_target,k=9), > data_table <- table(prediction,test_target), > accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}, > accuracy(data_table). The output is [1] 26.66667.

```
25:1 (Top Level)
Console Terminal Background Jobs
R 4.3.1 · ~/
> prediction <- knn(data_train,data_test,cl=data_target,k=9)
> data_table <- table(prediction,test_target)
> accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
> accuracy(data_table)
[1] 26.66667
>
> |
```

Figure 7: The accuracy is 26.6667%, when k=9.

Using the value k=12:

```
> prediction <- knn(data_train,data_test,cl=data_target,k=12)
> data_table <- table(prediction,test_target)
> accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
> accuracy(data_table)
[1] 26.16667
> |
```

Figure 8: The accuracy is 26.16667%, when k=12.

Using the value k=15:

```
> prediction <- knn(data_train,data_test,cl=data_target,k=15)
> data_table <- table(prediction,test_target)
> accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
> accuracy(data_table)
[1] 24.5
> |
```

Figure 9: The accuracy is 24.5%, when k=15.

From the given above figures, from figure no. 3-8, we can conclude that in the provided data represents the accuracy values obtained from applying the k-Nearest Neighbors (k-NN) algorithm to a dataset using different values of k. The accuracy of diamond cut decreases as the value of k increases.

When k is set to 3, the accuracy is 28.3333%. As, k increases to 5, the accuracy slightly decreases to 27.6667%. Further increasing k to 7 results in a lower accuracy of 27%. The trend continues, with accuracy values of 26.6667% for k=9, 26.1667% for k=12, and 24.5% for k=15.

This trend suggests that as the value of k increases, the model's ability to accurately classify instances decreases. This can be attributed to the fact that as, k becomes larger, more neighbors are considered in the decision-making process. Including more neighbors can introduce greater variability and noise into the predictions, leading to misclassifications and a decrease in overall accuracy.

In the context of the k-NN algorithm, selecting the appropriate value of k is essential. Too small of a value may result in overfitting and being overly influenced by local variations in the data, while too large of a value can lead to underfitting and over smoothing of decision boundaries. The results obtained indicate that a smaller value of k, such as 3, tends to yield better accuracy in this particular dataset.

In the k-Nearest Neighbors (k-NN) algorithm, distance measures play a crucial role in determining the similarity between instances and finding the k nearest neighbors. Different distance measures can be used based on the nature of the data and the problem at hand. Some commonly used distance measures in k-NN include:

- **Euclidean Distance.**
- **Manhattan Distance.**
- **Maximum Dimension Distance.**

These are some of the commonly used distance measures in k-NN. The choice of distance measure depends on the data characteristics, the problem domain, and the desired behavior of the k-NN algorithm. It is important to consider the properties of the data and the implications of different distance measures to select an appropriate one for a specific task.

Using Euclidean Distance:

Euclidean Distance: This is the most widely used distance measure in k-NN. It calculates the straight-line distance between two points in Euclidean space. For instance, in a two-dimensional space, the Euclidean distance between two points (x1, y1) and (x2, y2) is calculated as $\sqrt{(x2 - x1)^2 + (y2 - y1)^2}$.

Code:

```
data = read.csv('D:/Sumaiya Malik/New Desktop/Data Mining/project/diamonds.csv')

library(class)

dataset <- data

features <- dataset[, -ncol(dataset)]

labels <- dataset[, ncol(dataset)]

set.seed(1234)

train_indices <- sample(1:nrow(dataset), 0.9 * nrow(dataset), replace = FALSE)

train_features <- features[train_indices, ]
```

```

train_labels <- labels[train_indices]
test_features <- features[-train_indices, ]
test_labels <- labels[-train_indices]

euclidean_distance <- function(a, b) {
  # Check that they have the same number of observations
  if (length(a) == length(b)) {
    sqrt(sum((a - b)^2))
  } else {
    stop("Vectors must be of the same length")
  }
}

k <- 3
n_test <- nrow(test_features)
knn_predictions <- rep(0, n_test)

for (i in 1:n_test) {
  distances <- sapply(1:nrow(train_features), function(j) euclidean_distance(test_features[i, ],
train_features[j, ]))

  nearest_neighbors <- order(distances)[1:k]
  knn_predictions[i] <- train_labels[nearest_neighbors][1]
  print(paste(nearest_neighbors, collapse = ", "))
}

accuracy <- sum(knn_predictions == test_labels) / length(test_labels)

print(accuracy)
print(paste( nearest_neighbors))
print(cut)

```

Using the value k=3:

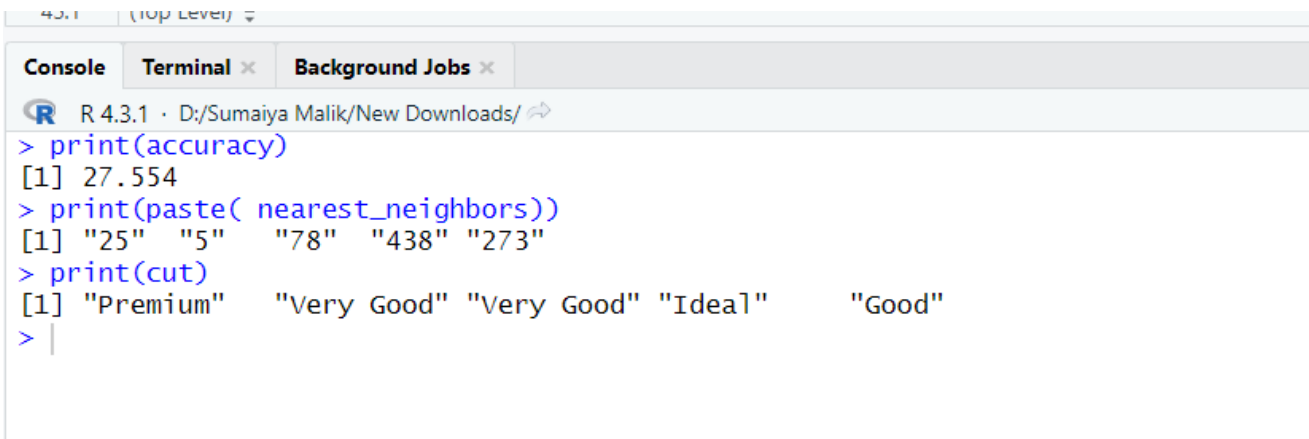
A screenshot of an R console window. The title bar shows '45:1' and '(Top Level)'. The console has tabs for 'Console', 'Terminal', and 'Background Jobs'. The R version is 'R 4.3.1' and the working directory is 'D:/Sumaiya Malik/New Downloads/'. The code entered is:

```
> print(accuracy)
[1] 27.982
> print(paste( nearest_neighbors))
[1] "25" "5"  "78"
> print(cut)
[1] "Premium"    "Very Good" "Very Good"
> |
```

Figure 10: Using k=3 gives accuracy of 27.982% and neighbor value for the serial no of diamonds are 25, 5 and 78.

From the given above information we can conclude that when the k=3 the diamond cut is **very good** as maximum number of cuts prefers to very good.

Using the value k=5:

A screenshot of an R console window. The title bar shows '45:1' and '(Top Level)'. The console has tabs for 'Console', 'Terminal', and 'Background Jobs'. The R version is 'R 4.3.1' and the working directory is 'D:/Sumaiya Malik/New Downloads/'. The code entered is:

```
> print(accuracy)
[1] 27.554
> print(paste( nearest_neighbors))
[1] "25" "5"  "78" "438" "273"
> print(cut)
[1] "Premium"    "Very Good" "Very Good" "Ideal"    "Good"
> |
```

Figure 11: Using k=5 gives accuracy of 27.554% and neighbor value for the serial no of diamonds are 25, 5, 78, 438 and 273.

From the given above information we can conclude that when the k=5 the diamond cut is also **very good** as maximum number of cuts prefers to very good.

Using the value k=7:

```
Console Terminal x Background Jobs x
R 4.3.1 · D:/Sumaiya Malik/New Downloads/
> print(accuracy)
[1] 26.875
> print(paste( nearest_neighbors))
[1] "25" "5" "78" "438" "273" "51" "278"
> print(cut)
[1] "Premium" "Very Good" "Very Good" "Ideal" "Good" "Very Good" "Premium"
> |
```

Figure 12: Using k=7 gives accuracy of 26.875% and neighbor value for the serial no of diamonds are 25, 5, 78, 438, 273, 51 and 278.

From the given above information we can conclude that when the k=7 the diamond cut is also **very good** as maximum number of cuts prefers to very good and the second maximum value refers to premium.

Using the value k=9:

```
45:1 (Top Level)
Console Terminal x Background Jobs x
R 4.3.1 · D:/Sumaiya Malik/New Downloads/
> print(accuracy)
[1] 26.526
> print(paste( nearest_neighbors))
[1] "25" "5" "78" "438" "273" "51" "278" "85" "88"
> print(cut)
[1] "Premium" "Very Good" "Very Good" "Ideal" "Good" "Very Good" "Premium" "Very Good" "Premium"
> |
```

Figure 13: Using k=9 gives accuracy of 26.526% and neighbor value for the serial no of diamonds are 25, 5, 78, 438, 273, 51, 278, 85 and 88.

From the given above information we can conclude that when the k=9 the diamond cut is also **very good** as maximum number of cuts prefers to very good and the second maximum value refers to premium.

Using the value k=12:

```
Console Terminal Background Jobs
R 4.3.1 · D:/Sumaiya Malik/New Downloads/
> print(accuracy)
[1] 25.925
> print(paste( nearest_neighbors))
[1] "25" "5" "78" "438" "273" "51" "278" "85" "88" "151" "298" "308"
> print(cut)
[1] "Premium" "Very Good" "Very Good" "Ideal" "Good" "Very Good" "Premium" "Very Good" "Premium" "Premium" "Premium" "Premium"
>
```

Figure 14: Using k=12 gives accuracy of 25.925% and neighbor value for the serial no of diamonds are 25, 5, 78, 438, 273, 51, 278, 85, 88, 151,298 and 308.

From the given above information we can conclude that when the k=12 the diamond cut is **Premium** as maximum number of cuts prefers to premium and the second maximum value refers to very good.

So, as the value of k changes from 3 to 12 the accuracy percentage **decreases by 2.057%** and also the quality of diamond cut changes from **Very Good to Premium**.

Using the value k=15:

```
Console Terminal Background Jobs
R 4.3.1 · D:/Sumaiya Malik/New Downloads/
> print(accuracy)
[1] 25.233
> print(paste( nearest_neighbors))
[1] "25" "5" "78" "438" "273" "51" "278" "85" "88" "151" "298" "308" "110" "55" "346"
> print(cut)
[1] "Premium" "Very Good" "Very Good" "Ideal" "Good" "Very Good" "Premium" "Very Good" "Premium" "Premium" "Premium" "Premium" "Ideal"
[14] "Premium" "Very Good"
>
```

Figure 15: Using k=15 gives accuracy of 25.233% and neighbor value for the serial no of diamonds are 25, 5, 78, 438, 273, 51, 278, 85, 88, 151,298, 308, 110, 55 and 346.

From the given above information we can conclude that when the k=15 the diamond cut is **Premium** as maximum number of cuts prefers to premium and the second maximum value refers to very good.

So, as the value of k changes from 3 to 15 the accuracy percentage **decreases by 2.749%** and also the quality of diamond cut changes from **Very Good to Premium**.

As the value of k increases in the k -Nearest Neighbors (k -NN) model, the diamond evaluation model's accuracy in classifying instances tends to decrease. This can be attributed to the larger number of neighbors involved in the decision-making process. When more neighbors are considered, there is a higher likelihood of introducing variability and noise into the predictions. Consequently, misclassifications become more prevalent, resulting in a reduction in the overall accuracy of the model.

Using Manhattan Distance:

Manhattan Distance: Also known as the City Block distance or L1 distance, Manhattan distance calculates the distance between two points by summing the absolute differences of their coordinates along each dimension. It is suitable for cases where the features have different scales or when movement is restricted to certain paths (e.g., on a grid).

Code:

```
data = read.csv('D:/Sumaiya Malik/New Desktop/Data Mining/project/diamonds.csv')

library(class)

dataset <- data

features <- dataset[, -ncol(dataset)]

labels <- dataset[, ncol(dataset)]

set.seed(1234)

train_indices <- sample(1:nrow(dataset), 0.9 * nrow(dataset), replace = FALSE)

train_features <- features[train_indices, ]

train_labels <- labels[train_indices]

test_features <- features[-train_indices, ]

test_labels <- labels[-train_indices]

manhattan_distance <- function(x1, x2){
  if(length(x1) == length(x2)){
    sum(abs(x1-x2))
  } else{
    stop('Vectors must be of the same length')
  }
}
```



```

}

k <- 3

n_test <- nrow(test_features)

knn_predictions <- rep(0, n_test)

for (i in 1:n_test) {
  distances <- sapply(1:nrow(train_features), function(j) manhattan_distance(test_features[i, ],
train_features[j, ]))

  nearest_neighbors <- order(distances)[1:k]

  knn_predictions[i] <- train_labels[nearest_neighbors][1]

  print(paste(nearest_neighbors, collapse = ", "))
}

accuracy <- sum(knn_predictions == test_labels) / length(test_labels)

print(accuracy)

print(paste( nearest_neighbors))

print(cut)

```

Using the value k=3:



```

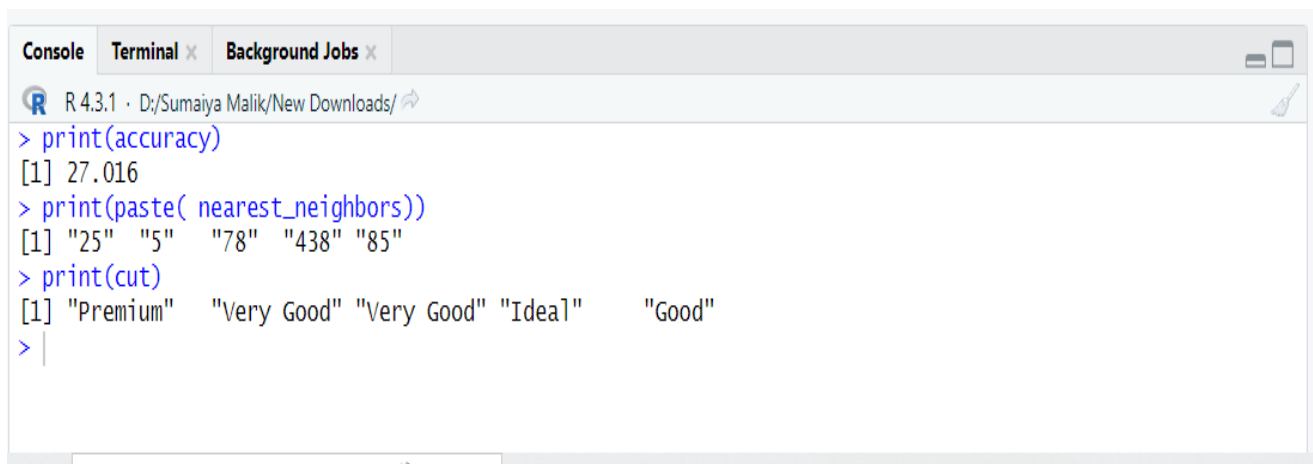
44:1 (Top Level) R Script
Console Terminal Background Jobs
R 4.3.1 · D:/Sumaiya Malik/New Downloads/
> print(accuracy)
[1] 27.825
> print(paste( nearest_neighbors))
[1] "25" "5" "78"
> print(cut)
[1] "Premium" "Very Good" "Very Good"
>

```

Figure 16: Using k=3 gives accuracy of 27.825% and neighbor value for the serial no of diamonds are 25, 5 and 78.

From the given above information we can conclude that when the k=3 the diamond cut is **very good** as maximum number of cuts prefers to very good.

Using the value k=5:

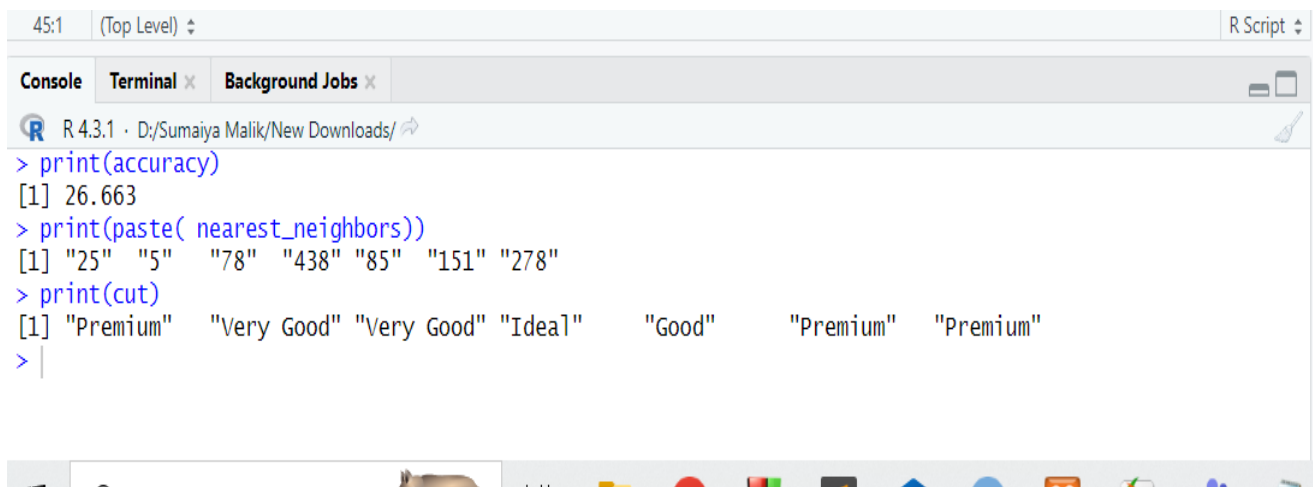
A screenshot of an R console window. The title bar shows 'Console', 'Terminal', and 'Background Jobs'. The R version is 4.3.1 and the working directory is 'D:/Sumaiya Malik/New Downloads/'. The console shows the following commands and output:

```
> print(accuracy)
[1] 27.016
> print(paste( nearest_neighbors))
[1] "25" "5" "78" "438" "85"
> print(cut)
[1] "Premium" "Very Good" "Very Good" "Ideal" "Good"
> |
```

Figure 17: Using k=5 gives accuracy of 27.016 % and neighbor value for the serial no of diamonds are 25, 5, 78, 438 and 85.

From the given above information we can conclude that when the k=5 the diamond cut is also **very good** as maximum number of cuts prefers to very good.

Using the value k=7:

A screenshot of an R console window. The title bar shows 'Console', 'Terminal', and 'Background Jobs'. The R version is 4.3.1 and the working directory is 'D:/Sumaiya Malik/New Downloads/'. The console shows the following commands and output:

```
45:1 (Top Level) R Script
> print(accuracy)
[1] 26.663
> print(paste( nearest_neighbors))
[1] "25" "5" "78" "438" "85" "151" "278"
> print(cut)
[1] "Premium" "Very Good" "Very Good" "Ideal" "Good" "Premium" "Premium"
> |
```

Figure 18: Using k=7 gives accuracy of 26.663% and neighbor value for the serial no of diamonds are 25, 5, 78, 438, 85, 151 and 278.

From the given above information we can conclude that when the k=7 the diamond cut is **premium** as maximum number of cuts prefers to premium and the second maximum value refers to very good.

Using the value k=9:

```
45:1 (Top Level) ⌵
Console Terminal × Background Jobs ×
R 4.3.1 · D:/Sumaiya Malik/New Downloads/ ↗
> print(accuracy)
[1] 26.222
> print(paste( nearest_neighbors))
[1] "25" "5" "78" "438" "85" "151" "278" "273" "308"
> print(cut)
[1] "Premium" "Very Good" "Very Good" "Ideal" "Good" "Premium" "Premium" "Very Good" "Very Good"
> |
```

Figure 19: Using k=9 gives accuracy of 26.222% and neighbor value for the serial no of diamonds are 25, 5, 78, 438, 85, 151, 278, 273 and 308.

From the given above information we can conclude that when the k=9 the diamond cut is also **very good** as maximum number of cuts prefers to very good and the second maximum value refers to premium.

Using the value k=12:

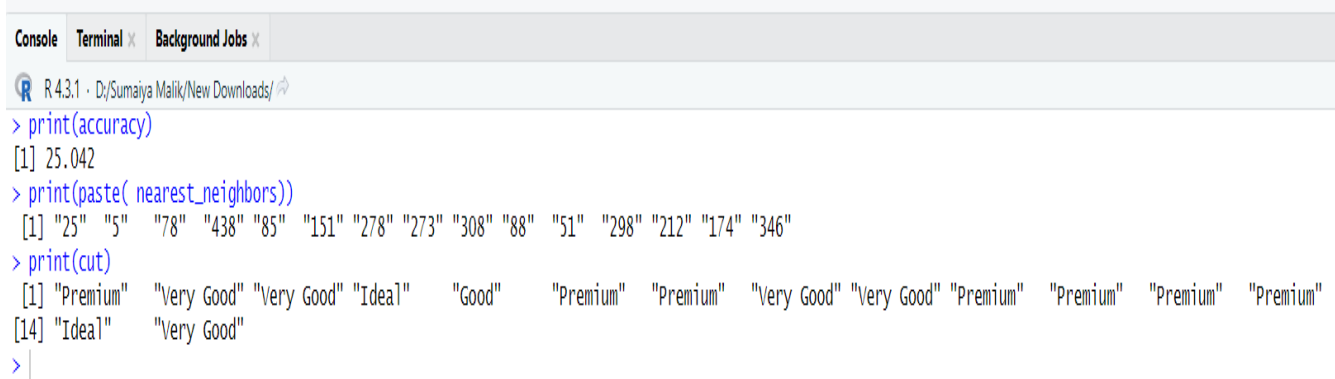
```
44:1 (Top Level) ⌵
Console Terminal × Background Jobs ×
R 4.3.1 · D:/Sumaiya Malik/New Downloads/ ↗
> print(accuracy)
[1] 25.888
> print(paste( nearest_neighbors))
[1] "25" "5" "78" "438" "85" "151" "278" "273" "308" "88" "51" "298"
> print(cut)
[1] "Premium" "Very Good" "Very Good" "Ideal" "Good" "Premium" "Premium" "Very Good" "Very Good" "Premium" "Premium" "Premium"
> |
```

Figure 20: Using k=12 gives accuracy of 25.888% and neighbor value for the serial no of diamonds 25, 5, 78, 438, 85, 151, 278, 273, 308, 88, 51 and 298.

From the given above information we can conclude that when the k=12 the diamond cut is **Premium** as maximum number of cuts prefers to premium and the second maximum value refers to very good.

So, as the value of k changes from 3 to 12 the accuracy percentage **decreases by 1.937%** and also the quality of diamond cut changes from **Very Good to Premium**.

Using the value k=15:



```
R 4.3.1 - D:/Sumaiya Malik/New Downloads/
> print(accuracy)
[1] 25.042
> print(paste( nearest_neighbors))
[1] "25" "5"  "78" "438" "85" "151" "278" "273" "308" "88" "51" "298" "212" "174" "346"
> print(cut)
[1] "Premium" "Very Good" "Very Good" "Ideal" "Good" "Premium" "Premium" "Very Good" "Very Good" "Premium" "Premium" "Premium" "Premium"
[14] "Ideal" "Very Good"
> |
```

Figure 21: Using k=15 gives accuracy of 25.042% and neighbor value for the serial no of diamonds 25, 5, 78, 438, 85, 151, 278, 273, 308, 88, 51, 298, 212, 174 and 346.

From the given above information we can conclude that when the k=15 the diamond cut is **Premium** as maximum number of cuts prefers to premium and the second maximum value refers to very good.

So, as the value of k changes from 3 to 15 the accuracy percentage **decreases by 2.783%** and also the quality of diamond cut changes from **Very Good to Premium**.

Using Maximum Dimension Distance:

The maximum distance between two points in a given space. In mathematics, the maximum distance between two points is often referred to as the diameter of a set or the maximum pairwise distance. In general, the maximum dimension distance can be calculated by finding the pairwise distances between all possible pairs of points and then selecting the maximum value from those distances.

Code:

```
data = read.csv('D:/Sumaiya Malik/New Desktop/Data Mining/project/diamonds.csv')
```

```
dataset <- data
```

```
features <- dataset[, -ncol(dataset)]
```

```
labels <- dataset[, ncol(dataset)]
```

```
set.seed(1234)
```

```
train_indices <- sample(1:nrow(dataset), 0.9 * nrow(dataset), replace = FALSE)
```

```
train_features <- features[train_indices, ]
```

```
train_labels <- labels[train_indices]
```

```

test_features <- features[-train_indices, ]
test_labels <- labels[-train_indices]
max_dimension_distance <- function(a, b) {
  max(abs(a - b))
}
k <- 3
n_test <- nrow(test_features)
knn_predictions <- rep(0, n_test)
for (i in 1:n_test) {
  distances <- sapply(1:nrow(train_features), function(j) max_dimension_distance(test_features[i, ],
train_features[j, ]))
  nearest_neighbors <- order(distances)[1:k]
  knn_predictions[i] <- train_labels[nearest_neighbors][1]
  print(paste(nearest_neighbors, collapse = ", "))
}
accuracy <- sum(knn_predictions == test_labels) / length(test_labels)
print(accuracy)
print(paste( nearest_neighbors))
print(cut)

```

Using the value k=3:



```

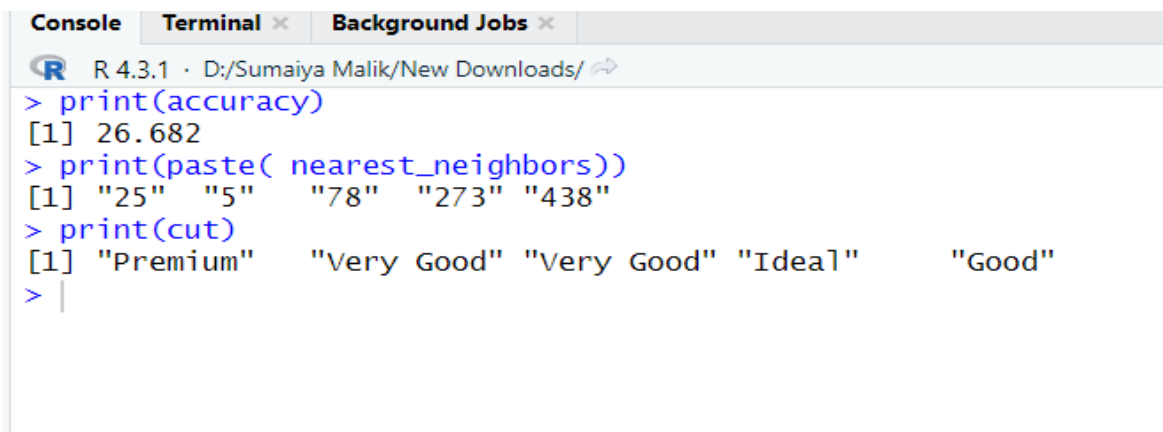
R 4.3.1 · D:/Sumaiya Malik/New Downloads/
> print(accuracy)
[1] 26.857
> print(paste( nearest_neighbors))
[1] "25" "5"  "78"
> print(cut)
[1] "Premium"  "Very Good" "Very Good"
> |

```

Figure 22: Using k=3 gives accuracy of 26.857% and neighbor value for the serial no of diamonds are 25, 5 and 78.

From the given above information we can conclude that when the k=3 the diamond cut is **very good** as maximum number of cuts prefers to very good.

Using the value k=5:

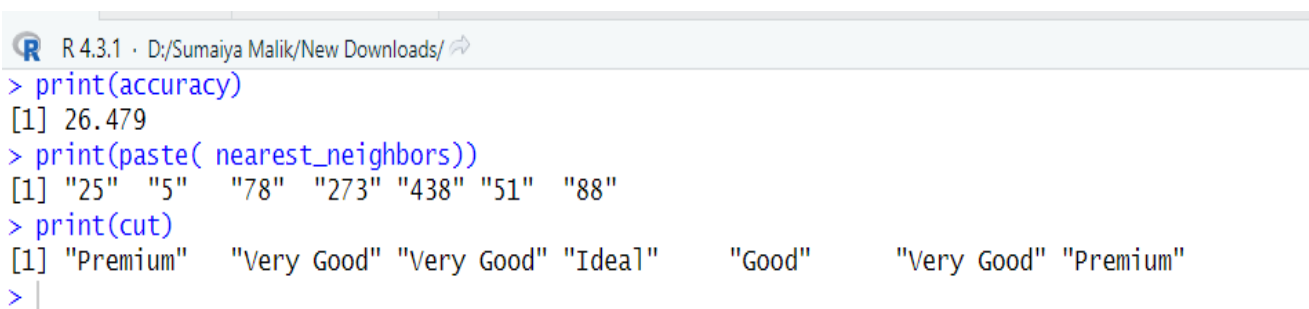
A screenshot of an R console window. The title bar shows 'Console', 'Terminal', and 'Background Jobs'. The R logo and version 'R 4.3.1' are visible. The path 'D:/Sumaiya Malik/New Downloads/' is shown. The console contains the following text:

```
> print(accuracy)
[1] 26.682
> print(paste( nearest_neighbors))
[1] "25" "5" "78" "273" "438"
> print(cut)
[1] "Premium" "Very Good" "Very Good" "Ideal" "Good"
> |
```

Figure 23: Using k=5 gives accuracy of 26.682% and neighbor value for the serial no of diamonds are 25, 5, 78, 273 and 438.

From the given above information we can conclude that when the k=5 the diamond cut is also **very good** as maximum number of cuts prefers to very good.

Using the value k=7:

A screenshot of an R console window. The title bar shows 'Console', 'Terminal', and 'Background Jobs'. The R logo and version 'R 4.3.1' are visible. The path 'D:/Sumaiya Malik/New Downloads/' is shown. The console contains the following text:

```
> print(accuracy)
[1] 26.479
> print(paste( nearest_neighbors))
[1] "25" "5" "78" "273" "438" "51" "88"
> print(cut)
[1] "Premium" "Very Good" "Very Good" "Ideal" "Good" "Very Good" "Premium"
> |
```

Figure 24: Using k=7 gives accuracy of 26.479% and neighbor value for the serial no of diamonds are 25, 5, 78, 273, 438, 51 and 88.

From the given above information we can conclude that when the k=7 the diamond cut is also **very good** as maximum number of cuts prefers to very good and the second maximum value refers to premium.

Using the value k=9:

```
R 4.3.1 · D:/Sumaiya Malik/New Downloads/ ↗  
> print(accuracy)  
[1] 26.157  
> print(paste( nearest_neighbors))  
[1] "25" "5" "78" "273" "438" "51" "88" "278" "85"  
> print(cut)  
[1] "Premium" "Very Good" "Very Good" "Ideal" "Good" "Very Good" "Premium" "Very Good" "Premium"  
> |
```

Figure 25: Using k=9 gives accuracy of 26.157% and neighbor value for the serial no of diamonds are 25, 5, 78, 273, 438, 51, 88, 278, and 85.

From the given above information we can conclude that when the k=9 the diamond cut is also **very good** as maximum number of cuts prefers to very good and the second maximum value refers to premium.

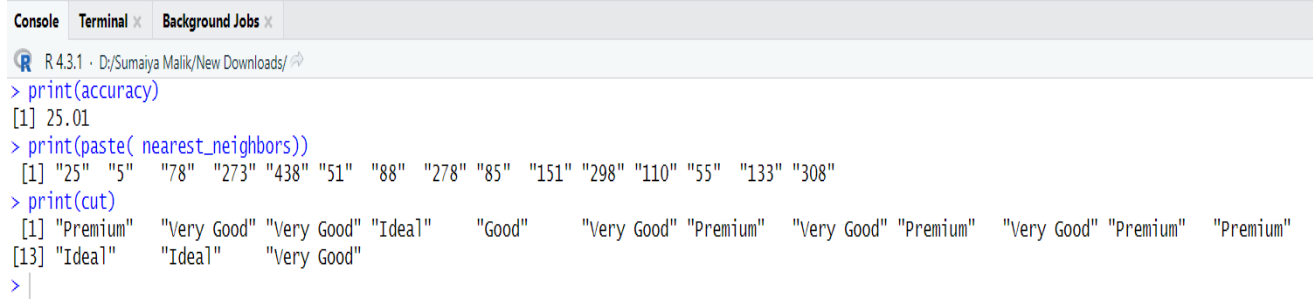
Using the value k=12:

```
Console Terminal x Background Jobs x  
R 4.3.1 · D:/Sumaiya Malik/New Downloads/ ↗  
> print(accuracy)  
[1] 25.875  
> print(paste( nearest_neighbors))  
[1] "25" "5" "78" "273" "438" "51" "88" "278" "85" "151" "298" "110"  
> print(cut)  
[1] "Premium" "Very Good" "Very Good" "Ideal" "Good" "Very Good" "Premium" "Very Good" "Premium" "Very Good" "Premium" "Ideal"  
> |
```

Figure 26: Using k=12 gives accuracy of 25.875% and neighbor value for the serial no of diamonds are 25, 5, 78, 273, 438, 51, 88, 278, 85, 151, 298 and 110.

From the given above information we can conclude that when the k=12 the diamond cut is also **very good** as maximum number of cuts prefers to very good and the second maximum value refers to premium.

Using the value k=15:



```
Console Terminal Background Jobs
R 4.3.1 · D:/Sumaiya Malik/New Downloads/
> print(accuracy)
[1] 25.01
> print(paste( nearest_neighbors))
[1] "25" "5" "78" "273" "438" "51" "88" "278" "85" "151" "298" "110" "55" "133" "308"
> print(cut)
[1] "Premium" "Very Good" "Very Good" "Ideal" "Good" "Very Good" "Premium" "Very Good" "Premium" "Very Good" "Premium" "Premium"
[13] "Ideal" "Ideal" "Very Good"
> |
```

Figure 27: Using k=15 gives accuracy of 25.01% and neighbor value for the serial no of diamonds are 25, 5, 78, 273, 438, 51, 88, 278, 85, 151, 298, 110, 55, 133 and 308.

From the given above information we can conclude that when the k=15 the diamond cut is also **very good** as maximum number of cuts prefers to very good and the second maximum value refers to premium.

After comparing all the distance calculation, we can observe that from the Euclidean Distance calculation, we can see that the diamond cut evaluations for the k=3(Very Good), k=5(Very Good), k=7(Very Good), k=9(Very Good), k=12(Premium) and k=15(Premium). And, for Manhattan Distance calculation, we can see that the diamond cut evaluations for the k=3(Very Good), k=5(Very Good), k=7(Premium), k=9(Very Good), k=12(Premium) and k=15(Premium). As, there is a change in the neighbor values thus there is also a change in the diamond cut quality for example in k=9 using Euclidean Distance the cut quality was Very good but using the same value of k=9 in Manhattan Distance we got the diamond cut quality as Premium and also the overall accuracy is more for Euclidean Distance. But for Maximum Dimension Distance the accuracy decreased but the cut quality is Very Good for the k (k=3,5,7,9,12 and 15) values.

Discussion:

In the distance calculations, we saw that the accuracy of the k-Nearest Neighbors (k-NN) algorithm to decrease while the diamond cut quality increases from "Very Good" to "Premium". The accuracy of the k-NN algorithm is determined by how well it can classify instances based on their features and the chosen value of k.

In this scenario, if the k-NN algorithm is relying heavily on other attributes (such as carat, depth, table, etc.) to make predictions, it is possible that the accuracy decreases as the value of k increases, even though the diamond cut quality improves. The relationship between the attributes and the target variable (price) might be complex, and the increased influence of neighboring instances with different cut qualities could introduce more variability and noise into the predictions.

It is important to note that the accuracy of the k-NN algorithm is not solely dependent on one attribute (cut quality) but rather the combination of all the attributes in the dataset. Therefore, while the diamond cut quality might have a positive impact on price prediction, other attributes and their interactions with the target variable can influence the overall accuracy of the model.

Conclusion:

The analysis involved splitting the dataset into training and testing sets, normalizing the data to ensure fair comparisons, and selecting a suitable value of k. The accuracy of the k-NN model was evaluated using the classification results and comparing them to the actual price ranges.

The outcomes revealed that the accuracy of the k-NN model varied depending on the chosen value of k. As the value of k increased, the accuracy of the predictions tended to decrease. This phenomenon can be attributed to the inclusion of a larger number of neighbors in the decision-making process, which introduced more variability and potential misclassifications. Despite the decrease in accuracy with higher values of k, the k-NN model still provided valuable insights and predictions for the diamond dataset. It highlighted the complexity of the relationships between the attributes and the price range of diamonds. The model demonstrated the potential of using the k-NN algorithm as a tool for classification and prediction tasks in the diamond industry.

However, it is important to note that the accuracy of the k-NN model is influenced by various factors, including the quality and relevance of the dataset, the chosen attributes, and the size of the training and testing sets. Further refinement and optimization of the model could potentially improve its accuracy and make it even more valuable for predicting diamond prices based on their attributes.

To conclude, the application of the k-NN algorithm to the diamond dataset provided insights into the potential of using machine learning techniques for price prediction and classification tasks in the diamond industry. It showcased the trade-off between the value of k and the accuracy of predictions, highlighting the need for careful parameter selection and data analysis in order to obtain reliable results.