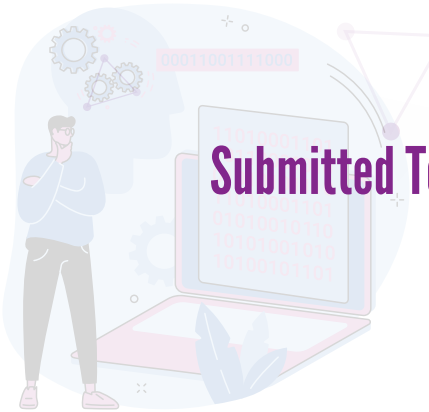
APPLICATION OF KNN TO CLASSIFY A CLIENT'S SUBSCRIPTION TO A LONG-TERM DEPOSIT AT A BANK



Submitted To: DR. AKINUL ISLAM JONY
ASSOCIATE PROFESSOR,DEPARTMENT HEAD
COMPUTER SCIENCE

Project by: SAJID IBNA MAHBUB (20-42109-1)
SAIMA SADIA RATRI (20-43793-2)
NAFIQUR RAHMAN ABIR (20-42165-1)
SULTANUL ARIFEEN HAMIM (20-42017-1)

Project Overview:

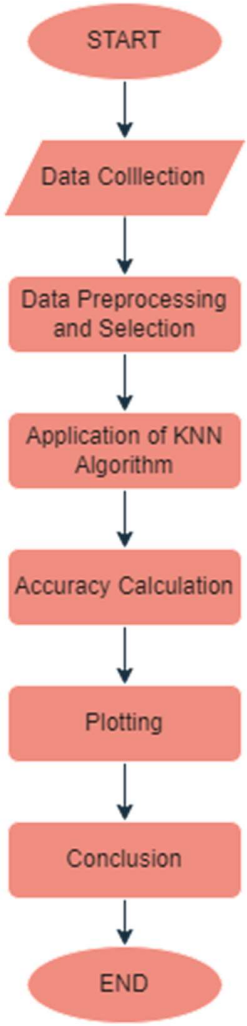
The "Banking Dataset Classification" project aims to predict whether a client will subscribe to a term deposit using the K-Nearest Neighbors (KNN) algorithm. The dataset used for this project consists of 5000 instances.
The dataset contains various features that provide valuable information about the clients and their interactions with the bank.
This project uses the programming language R for data analysis, preprocessing, and implementing the KNN algorithm. Here the target variable in this dataset is "Subscribed," indicating whether the client subscribed to a term deposit.

The KNN algorithm is a popular machine learning algorithm used for classification tasks. It works by classifying new instances based on the majority class of their nearest neighbours in the feature space. This project implements KNN to predict whether a client will subscribe to a term deposit based on the given features.

Before applying the KNN algorithm, The preprocess of the dataset has been completed. This preprocessing phase involves handling missing values, encoding categorical variables, and performing any necessary feature scaling or normalization. Additionally, the dataset is divided into training and testing sets to evaluate the trained model's performance accurately. Once the KNN model is trained on the training set, We evaluated it using the testing set to assess its predictive accuracy. Various evaluation metrics, such as accuracy and recall, measure the model's performance and determine its effectiveness in predicting whether a client will subscribe to a term deposit.

This project aims to build a predictive model to classify clients as potential subscribers or non-subscribers to a term deposit. This model can be helpful for banks in targeting their marketing campaigns effectively and improving their overall success rate.

Project Workflow:



Data Set:

This project explores a comprehensive dataset that forms the backbone of our analysis and research. In the "Banking Dataset Classification" project, the dataset serves as the foundation for understanding and analyzing various aspects of banking operations. It comprises a collection of structured information, meticulously gathered from diverse sources within the banking domain. The "Banking Dataset Classification" project aims to predict whether a client will subscribe to a term deposit. The dataset used for this project consists of 5000 instances.

The dataset available on Kaggle, which can be accessed through the provided link (https://www.kaggle.com/datasets/rashmiranu/banking-dataset-classification?select=new_train.csv), contains a comprehensive set of features that provide insights into the clients and their interactions with the bank.

Some of the critical attributes in the dataset include

1. **Age:** The client's age (numeric).
2. **Job:** The type of job the client is employed in (categorical).
3. **Marital Status:** The client's marital status (categorical).
4. **Education:** The education level of the client (categorical).

5. **Balance:** The current balance of the client's account (numeric).
6. **Housing:** Whether the client has a housing loan (categorical).
7. **Loan:** Whether the client has a personal loan (categorical).
8. **Contact:** The communication type used to contact the client (categorical).
9. **Campaign:** The number of contacts performed during the current campaign for the client (numeric).
10. **Poutcome:** The outcome of the previous marketing campaign (categorical).

Additionally, the dataset includes the target variable "Subscribed," which indicates whether the client subscribed to a term deposit (1 for "yes" and 0 for "no").By analyzing this dataset and applying machine learning algorithms, the goal is to build a predictive model that can accurately classify whether a client will subscribe to a term deposit based on the provided features. This model can then identify potential clients more likely to subscribe, enabling the bank to optimize its marketing strategies and improve its campaign success rate.

```
colnames(new_train)
colnames(new_test)
library(dplyr)
head(new_train)
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	pdays	previous	poutcome	y
1	49	1	1	1	1	2	2	0	9	3	227	4	999	0	3	no
2	37	2	1	2	2	2	2	1	9	3	202	2	999	1	2	no
3	78	3	1	3	2	2	2	0	5	1	1148	1	999	0	3	yes
4	36	4	1	2	2	1	2	1	3	1	120	2	999	0	3	no
5	59	3	2	2	2	2	2	0	4	2	368	2	999	0	3	no
6	29	4	3	2	2	2	2	0	6	3	256	2	999	0	3	no
7	26	5	3	1	2	2	2	1	6	3	449	1	999	0	3	yes
8	30	1	1	3	2	1	2	0	9	3	126	2	999	0	3	no
9	50	1	1	3	1	2	2	1	3	5	574	1	999	0	3	no
10	33	4	3	4	2	1	2	0	5	2	498	5	999	0	3	no
11	44	6	2	4	2	1	2	0	5	1	158	5	999	0	3	no
12	32	7	1	2	2	1	2	1	3	5	93	5	999	0	3	no
13	26	8	3	5	2	1	2	0	5	4	71	1	999	0	3	no
14	43	9	1	2	2	2	1	1	5	4	203	1	999	0	3	no
15	56	1	1	1	2	2	2	0	3	4	369	1	999	0	3	no
16	40	1	1	1	2	1	2	0	3	3	954	1	999	0	3	yes
17	32	4	2	2	2	1	2	0	6	2	105	1	999	0	3	no
18	47	7	3	5	1	2	2	1	3	2	148	5	999	0	3	no
19	50	6	3	3	2	1	2	1	3	2	98	9	999	0	3	no
20	34	4	3	2	2	2	1	0	1	2	288	2	3	1	1	yes
21	46	6	1	6	2	2	2	0	6	2	177	1	999	0	3	no
22	39	1	1	3	2	2	2	0	3	4	155	1	999	0	3	no
23	41	4	2	7	2	2	2	1	5	1	141	1	999	0	3	no
24	30	7	3	6	2	1	2	1	3	1	144	3	999	0	3	no
25	55	10	2	2	2	2	2	0	1	2	212	3	6	3	1	yes
26	33	1	3	3	2	1	2	0	8	5	146	1	999	1	2	no
27	46	6	1	4	2	2	2	0	2	3	325	2	999	0	3	no
28	38	1	2	4	2	1	1	0	9	3	291	1	999	0	3	no
29	36	4	2	2	1	1	1	0	5	3	103	1	999	0	3	no
30	30	7	3	5	2	1	2	0	6	4	121	1	999	0	3	no

Showing 1 to 30 of 32,950 entries, 16 total columns

Data Preprocessing:

Data pre-processing is a crucial step in preparing raw data for analysis or modeling. It involves cleaning, transforming, and enhancing the data to improve its quality and suitability for the task at hand. Common techniques include handling missing values, encoding categorical variables, scaling features, and removing noise, ensuring more accurate and meaningful results from data-driven processes. Here we used these pre-processing techniques on the data set prepare a complete dataset for analysis and visualization.

Data Transformation: Here data preprocessing on the new_train dataset is performed by converting categorical variables into factors with specific levels and assigning numeric labels to each level. This transformation allows for easier handling and analysis of the data, particularly for tasks such as modeling, statistical analysis, or visualization. It helps in representing categorical information numerically, making it suitable for various data analysis techniques.

```
unique(new_train$job)

unique(new_train$education)

unique(new_train$contact)

unique(new_train$poutcome)



new_train$job<-factor(new_train$job, levels = c("blue-
collar","entrepreneur","retired","admin.","student","services","technician","self-
employed","management","unemployed","unknown","housemaid"),

                      labels = c(1,2,3,4,5,6,7,8,9,10,11,12))



unique(new_train$marital)

new_train$marital <- factor(new_train$marital,

                            levels = c("married","divorced","single","unknown"),

                            labels = c(1,2,3,4))



new_train$education <- factor(new_train$education,

                              levels=
c("basic.9y","university.degree","basic.4y","high.school","professional.course","unknown","basic.6y","illiterate")
,
```

```
                        labels = c(1,2,3,4,5,6,7,8))
unique(new_train$default)


new_train$default <- factor(new_train$default,
                        levels= c("unknown","no","yes"),
                        labels = c(1,2,3))




unique(new_train$housing)


new_train$housing <- factor(new_train$housing,
                        levels = c("yes","no","unknown"),
                        labels = c(1,2,3))


unique(new_train$loan)


new_train$loan <- factor(new_train$loan,
                        levels = c("yes","no","unknown"),
                        labels = c(1,2,3))


unique(new_train$contact)


new_train$contact <- factor(new_train$contact,
                      levels = c("cellular","telephone"),
                      labels = c(0,1))
unique(new_train$month)
```

```
new_train$month <- factor(new_train$month,
                          levels = c("mar","apr","may","jun","jul","aug","sep","oct","nov","dec"),
                          labels = c(1,2,3,4,5,6,7,8,9,10))
length(unique(new_train$month))



unique(new_train$day_of_week)
length(unique(new_train$day_of_week))


new_train$day_of_week <- factor(new_train$day_of_week,
                          levels = c("mon","tue","wed","thu","fri"),
                          labels = c(1,2,3,4,5))



unique(new_train$poutcome)


new_train$poutcome <- factor(new_train$poutcome,
                             levels = c("success","failure","nonexistent"),
                             labels = c(1,2,3))
```

Output:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	pdays	previous	poutcome	y
1	49	1	1	1	1	2	2	1	9	3	227	4	999	0	3	no
2	37	2	1	2	2	2	2	2	9	3	202	2	999	1	2	no
3	78	3	1	3	2	2	2	1	5	1	1148	1	999	0	3	yes
4	36	4	1	2	2	1	2	2	3	1	120	2	999	0	3	no
5	59	3	2	2	2	2	2	1	4	2	368	2	999	0	3	no
6	29	4	3	2	2	2	2	1	6	3	256	2	999	0	3	no
7	26	5	3	1	2	2	2	2	6	3	449	1	999	0	3	yes
8	30	1	1	3	2	1	2	1	9	3	126	2	999	0	3	no
9	50	1	1	3	1	2	2	2	3	5	574	1	999	0	3	no
10	33	4	3	4	2	1	2	1	5	2	498	5	999	0	3	no
11	44	6	2	4	2	1	2	1	5	1	158	5	999	0	3	no
12	32	7	1	2	2	1	2	2	3	5	93	5	999	0	3	no
13	26	8	3	5	2	1	2	1	5	4	71	1	999	0	3	no
14	43	9	1	2	2	2	1	2	5	4	203	1	999	0	3	no
15	56	1	1	1	2	2	2	1	3	4	369	1	999	0	3	no
16	40	1	1	1	2	1	2	1	3	3	954	1	999	0	3	yes
17	32	4	2	2	2	1	2	1	6	2	105	1	999	0	3	no
18	47	7	3	5	1	2	2	2	3	2	148	5	999	0	3	no
19	50	6	3	3	2	1	2	2	3	2	98	9	999	0	3	no
20	46	6	1	6	2	2	2	1	6	2	177	1	999	0	3	no
21	39	1	1	3	2	2	2	1	3	4	155	1	999	0	3	no
22	41	4	2	7	2	2	2	2	5	1	141	1	999	0	3	no
23	30	7	3	6	2	1	2	2	3	1	144	3	999	0	3	no
24	33	1	3	3	2	1	2	1	8	5	146	1	999	1	2	no

Feature Reduction:

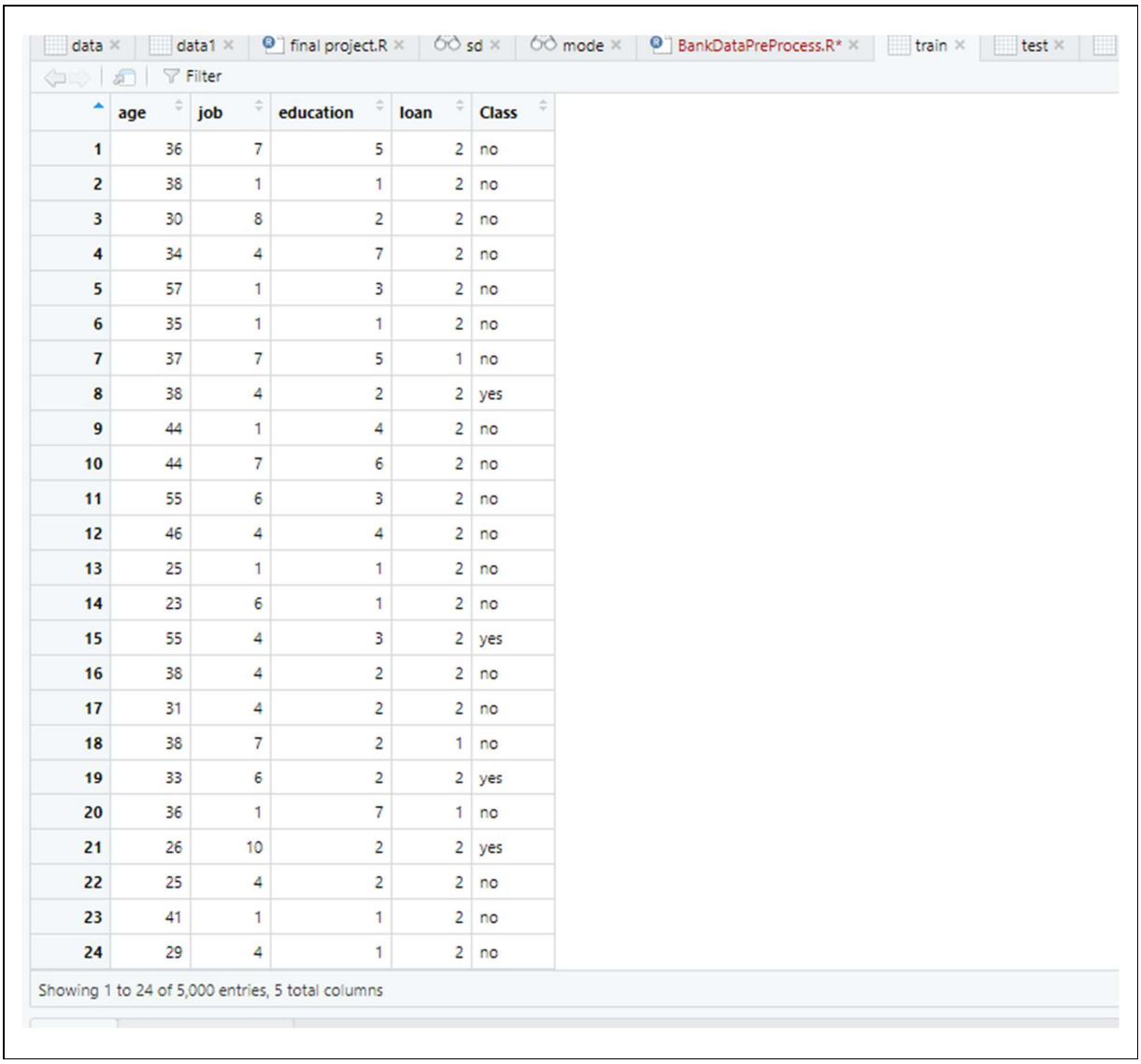
```
total_instances <- nrow(new_train)

train_indices <- sample(total_instances, 5000)

train_data <- new_train[train_indices, ]
```

Explanation: Here a subset of the new_train dataset for training purposes by randomly sampling 5000 instances. This is useful for creating a representative training set from a larger dataset, allowing for efficient training and evaluation of models.

Output:



data | data1 | final project.R | sd | mode | BankDataPreProcess.R* | train | test

Filter

	age	job	education	loan	Class
1	36	7	5	2	no
2	38	1	1	2	no
3	30	8	2	2	no
4	34	4	7	2	no
5	57	1	3	2	no
6	35	1	1	2	no
7	37	7	5	1	no
8	38	4	2	2	yes
9	44	1	4	2	no
10	44	7	6	2	no
11	55	6	3	2	no
12	46	4	4	2	no
13	25	1	1	2	no
14	23	6	1	2	no
15	55	4	3	2	yes
16	38	4	2	2	no
17	31	4	2	2	no
18	38	7	2	1	no
19	33	6	2	2	yes
20	36	1	7	1	no
21	26	10	2	2	yes
22	25	4	2	2	no
23	41	1	1	2	no
24	29	4	1	2	no

Showing 1 to 24 of 5,000 entries, 5 total columns

```
remaining_indices <- setdiff(1:total_instances, train_indices)

test_indices <- sample(remaining_indices, 2000)

test_data <- new_train[test_indices, ]

train_data <- subset(train_data, select = -c(pdays, previous))

test_data <- subset(test_data, select = -c(pdays, previous))
```

Explanation: Here the new_train dataset into two subsets: train_data with 5000 instances for training and test_data with 2000 instances for testing. It ensures that no instances are duplicated between the training and testing sets and removes specific columns (pdays and previous) from both sets before further analysis or modeling. This is a common approach to divide the data into separate training and testing sets for machine learning tasks to evaluate the model's performance on unseen data.

Output:

	age	job	education	loan	Class
1	41	6	4	2	no
2	29	4	2	1	no
3	47	8	1	2	yes
4	43	10	5	1	no
5	30	7	5	2	yes
6	40	6	4	2	no
7	55	9	2	2	no
8	48	1	1	2	yes
9	46	2	5	2	no
10	45	4	1	2	no
11	36	7	5	2	no
12	34	1	3	2	no
13	22	5	2	2	no
14	29	1	3	2	no
15	42	6	1	2	no
16	47	7	2	2	no
17	37	4	5	2	no
18	33	4	2	2	no
19	44	1	3	2	no
20	25	6	4	2	no
21	38	4	2	2	no
22	38	6	1	2	no
23	51	1	4	2	no
24	34	8	5	2	no

Normalization:

```
normalization=function(x)
{
  new_x=(x- min(x))/(max(x)-min(x))
  new_x=round(new_x, digits = 2)
  return(new_x)

}

train$age <- normalization( train$age)
train$job <- normalization(train$job)
train$education <- normalization(train$education)
```

Explanation: The normalization function is defined in the code, scaling a vector of values between 0 and 1, and rounding the outcome to two decimal places. The "age", "job", and "education" columns in the "train" dataset are then subjected to this normalization procedure.

Output:

age	job	education	loan	Class
0.25	0.55	0.57	2	no
0.28	0.00	0.00	2	no
0.17	0.64	0.14	2	no
0.23	0.27	0.86	2	no
0.55	0.00	0.29	2	no
0.24	0.00	0.00	2	no
0.27	0.55	0.57	1	no
0.28	0.27	0.14	2	yes
0.37	0.00	0.43	2	no
0.37	0.55	0.71	2	no
0.52	0.45	0.29	2	no
0.39	0.27	0.43	2	no
0.10	0.00	0.00	2	no
0.07	0.45	0.00	2	no
0.52	0.27	0.29	2	yes
0.28	0.27	0.14	2	no
0.18	0.27	0.14	2	no
0.28	0.55	0.14	1	no
0.21	0.45	0.14	2	yes
0.25	0.00	0.86	1	no
0.11	0.82	0.14	2	yes
0.10	0.27	0.14	2	no

KNN:

KNN, which stands for K-Nearest Neighbors, is a popular algorithm used in data mining and machine learning for classification and regression tasks. It is a non-parametric algorithm that can be used for both supervised and unsupervised learning.

```
euclidean_distance <- function(a1, a2,a3, b1, b2,b3) {
  sqrt((a1 - b1)^2 + (a2 - b2)^2 + (a3 - b3)^2)
}
```

```
manhattan_distance <- function(a1, a2,a3, b1, b2,b3) {
  abs(a1 - b1) + abs(a2 - b2) +  abs(a3 - b3)
}
```

```
max_dimensional_distance <- function(a1, a2,a3, b1, b2,b3) {
  max(abs(a1 - b1), abs(a2 - b2) ,abs(a3 - b3))
}
```

Explanation: Here in this code, we define the functions to calculate distances between two points. Euclidean distance calculates the straight-line distance, Manhattan distance calculates the sum of absolute differences along each dimension, and max-dimensional distance calculates the maximum absolute difference along any dimension.

```
# Function to perform k-NN classification
knn_classification <- function(test_data, train_data, k, distance_function) {
  predictions <- character(nrow(test_data))

  for (i in 1:nrow(test_data)) {
        distances <- numeric(nrow(train_data))

        for (j in 1:nrow(train_data)) {
        distances[j] <- distance_function(test_data$age[i], test_data$job[i], test_data$loan[i],
                              train_data$age[j], train_data$job[j], train_data$loan[i])
        }

        k_indices <- order(distances)[1:k]
        k_class <- train_data$Class[k_indices]
        predictions[i] <- names(which.max(table(k_class)))
  }

  return(predictions)
}
```

Explanation: Here a function called knn_classification that performs k-nearest neighbours (k-NN) classification. It takes test data, training data, a value of k, and a distance function as input. It iterates over each test data point and calculates the distances to all training data points using the provided distance function. It selects the k nearest neighbours based on the distances, determines the majority class among the neighbours, and assigns it as the predicted class for the test data point. Finally, it returns the predictions for all test data points.

```
k_values <- c(1,3,5,7,9,11,13,15,17,19,20)
```

Explanation: Various values for K are taken for calculation

Euclidean distance is one of the most widely used distance metrics. It calculates the straight-line distance between two points in Euclidean space.

```
#Euclidean
euAccuracy <- numeric(length(k_values))
m =1
for (k in k_values) {

 predictions <- knn_classification(test, train, k, euclidean_distance)
 accuracy <- sum(predictions == test$Class) / nrow(test)
 euAccuracy[m] <- accuracy
 recall <- sum(predictions == "yes" & test$Class == "yes") /
  sum(test$Class == "yes")

 cat("k:", k, "\n")
 cat("Accuracy:", accuracy * 100, "%\n")
 cat("Recall:", recall * 100, "%\n")
 cat("----------------------------\n")
 m=m+1 }
```

Explanation: Here, the accuracy and recall of the k-nearest neighbors (k-NN) classification algorithm using Euclidean distance has been done. It iterates over a range of k values, performs the k-NN classification on a test dataset using a training dataset, and computes the accuracy and recall for each k value. The results are printed for each iteration.

Manhattan Distance: Manhattan distance, also known as city block distance or L1 norm, calculates the sum of the absolute differences between corresponding feature values of two instances.

```
#Manhattan
train <- train[sample(nrow(train)), ]
test <- test[sample(nrow(test)), ]


manAccuracy <- numeric(length(k_values))
i =1
for (k in k_values) {


 predictions <- knn_classification(test, train, k, manhattan_distance)
 accuracy <- sum(predictions == test$Class) / nrow(test)
 manAccuracy[i] <- accuracy
 recall <- sum(predictions == "yes" & test$Class == "yes") /
   sum(test$Class == "yes")


 cat("k:", k, "\n")
 cat("Accuracy:", accuracy * 100, "%\n")
 cat("Recall:", recall * 100, "%\n")
 cat("----------------------------\n")
 i=i+1
```

```
}
```

Explanation: Here, k-nearest neighbors (k-NN) have performed classification using Manhattan distance. It randomly shuffles the training and test datasets. It iterates over a range of k values, performs k-NN classification on the test dataset using the training dataset and Manhattan distance, and calculates the accuracy and recall for each k value. The results are printed for each iteration.

```
#Max_Dimensional_Distance

p =1

train <- train[sample(nrow(train)), ]

test <- test[sample(nrow(test)), ]


maxAccuracy <- numeric(length(k_values))


for (k in k_values) {


 predictions <- knn_classification(test, train, k, max_dimensional_distance)

 accuracy <- sum(predictions == test$Class) / nrow(test)

 maxAccuracy[p] <- accuracy

 recall <- sum(predictions == "yes" & test$Class == "yes") /

   sum(test$Class == "yes")



 cat("k:", k, "\n")
```

```
 cat("Accuracy:", accuracy * 100, "%\n")
 cat("Recall:", recall * 100, "%\n")
 cat("----------------------------\n")
p=p+1
}
```

Explanation: This code performs k-nearest neighbours (k-NN) classification using the maximum dimensional distance metric. It shuffles the training and test datasets randomly. It iterates over a range of k values, performs k-NN classification on the test dataset using the training dataset and the maximum dimensional distance metric, and computes the accuracy and recalls for each k value. The results are printed for each iteration.

```
library(ggplot2)
library(dplyr)

library(hrbrthemes)

library(scales)

data <- data.frame(euAccuracy,manAccuracy,maxAccuracy, k_values)



 ggplot(data,aes(x=data$k_values, y=data$euAccuracy)) +
 geom_line( color="grey") +
 geom_point(shape=21, color="black", fill="#69b3a2", size=6) +
 scale_x_continuous(breaks = seq(0,20, 1)) +
```

```
  theme_ipsum() +
  labs(x = "K Values", y= "Accuracy")+
  ggtitle("KNN Using Euclidean Method")




  ggplot(data,aes(x=data$k_values, y=data$manAccuracy)) +
          geom_line( color="grey") +
    geom_point(shape=21, color="black", fill="#69b3a2", size=6) +
    scale_x_continuous(breaks = seq(0,20, 1)) +
    theme_ipsum() +
          labs(x = "K Values", y= "Accuracy")+
    ggtitle("KNN Using Manhatten Method")



  ggplot(data,aes(x=data$k_values, y=data$maxAccuracy)) +
          geom_line( color="grey") +
    geom_point(shape=21, color="black", fill="#69b3a2", size=6) +
    scale_x_continuous(breaks = seq(0,20, 1)) +
    theme_ipsum() +
          labs(x = "K Values", y= "Accuracy")+
          ggtitle("KNN Using Max Dimensional Distance")


eumean <- mean(data$euAccuracy)
```

```
manmean <- mean(data$manAccuracy)

maxmean <- mean(data$maxAccuracy)









# Create a data frame

df <- data.frame(Category = c("Euclidean", "Manhatten", "Max Dimensional"),

                                                    Value =
c(max(data$euAccuracy),max(data$manAccuracy),max(data$maxAccuracy)))




# Create a bar plot

ggplot(data = df, aes(x = Category, y = Value)) +

  geom_bar(stat = "identity", fill = "skyblue") +

  xlab("Method") +

  ylab("MAX Accuracy") +

  ggtitle("MAX Accuracy of Methods")
```

Explanation: This code utilizes the 'ggplot2' and 'dplyr' libraries to generate visualizations and perform data manipulations. It creates three line plots, each representing k-nearest neighbours (k-NN) classification accuracy using different distance metrics (Euclidean, Manhattan, and Max Dimensional). The plots display accuracy against various k values. A bar plot is also generated to compare the maximum accuracy achieved by each distance metric. The data used for plotting is stored in the 'data' data frame, and mean accuracy values are computed for each method and stored in 'eumean,' 'manmean', and 'maxmean' variabless.

Model Results and Comparisons:

Euclidean	Manhattan	Max Dimension Distance
k: 1 Accuracy: 79.95 % Recall: 18.90547 % ---------- k: 3 Accuracy: 86.9 % Recall: 9.950249 % ---------- k: 5 Accuracy: 87.3 % Recall: 7.960199 % ---------- k: 7 Accuracy: 88.15 % Recall: 2.487562 % ---------- k: 9 Accuracy: 88.75 % Recall: 0.9950249 % ---------- k: 11 Accuracy: 88.9 % Recall: 0.9950249 % ---------- k: 13 Accuracy: 89.95 % Recall: 0 % ---------- k: 15 Accuracy: 89.95 % Recall: 0 % ---------- k: 17 Accuracy: 89.95 % Recall: 0 % ---------- k: 19 Accuracy: 89.95 % Recall: 0 % ---------- k: 20 Accuracy: 89.95 % Recall: 0 % ----------	k: 1 Accuracy: 82.9 % Recall: 13.43284 % ---------- k: 3 Accuracy: 88.35 % Recall: 4.477612 % ---------- k: 5 Accuracy: 89.3 % Recall: 1.99005 % ---------- k: 7 Accuracy: 89.45 % Recall: 0.4975124 % ---------- k: 9 Accuracy: 89.25 % Recall: 1.99005 % ---------- k: 11 Accuracy: 89.9 % Recall: 0 % ---------- k: 13 Accuracy: 89.85 % Recall: 0 % ---------- k: 15 Accuracy: 89.95 % Recall: 0 % ---------- k: 17 Accuracy: 89.95 % Recall: 0 % ---------- k: 19 Accuracy: 89.95 % Recall: 0 % ---------- k: 20 Accuracy: 89.85 % Recall: 0 % ----------	k: 1 Accuracy: 82.5 % Recall: 12.43781 % ---------- k: 3 Accuracy: 87 % Recall: 7.462687 % ---------- k: 5 Accuracy: 87.75 % Recall: 4.477612 % ---------- k: 7 Accuracy: 89.55 % Recall: 1.99005 % ---------- k: 9 Accuracy: 89.85 % Recall: 0.9950249 % ---------- k: 11 Accuracy: 90 % Recall: 0.9950249 % ---------- k: 13 Accuracy: 89.95 % Recall: 0 % ---------- k: 15 Accuracy: 89.95 % Recall: 0 % ---------- k: 17 Accuracy: 89.95 % Recall: 0 % ---------- k: 19 Accuracy: 89.95 % Recall: 0 % ---------- k: 20 Accuracy: 89.95 % Recall: 0 % ----------

```
> max(data$euAccuracy)
[1] 0.8995
> max(data$manAccuracy)
[1] 0.8995
> max(data$maxAccuracy)
[1] 0.9
```

Explanation: This code calculates and returns the maximum accuracy achieved for each distance metric (Euclidean, Manhattan, and Max Dimensional) in the k-nearest neighbors (k-NN) classification. The maximum accuracy values are displayed for each distance metric separately. Here is the accuracy and recall values for different values of k (number of neighbors) in three different distance metrics: Euclidean, Manhattan, and Max Dimension Distance. The accuracy represents the percentage of correctly classified instances, while the recall indicates the percentage of true positive instances correctly identified.

For the Euclidean distance metric:

The highest accuracy achieved is 89.95%

The highest recall achieved is 12.43781% with k = 1.

For the Manhattan distance metric:

The highest accuracy achieved is 89.95% with k = 13, 15, 17, 19, and 20.

The highest recall achieved is 7.462687% with k = 3.

For the Max Dimension Distance metric:

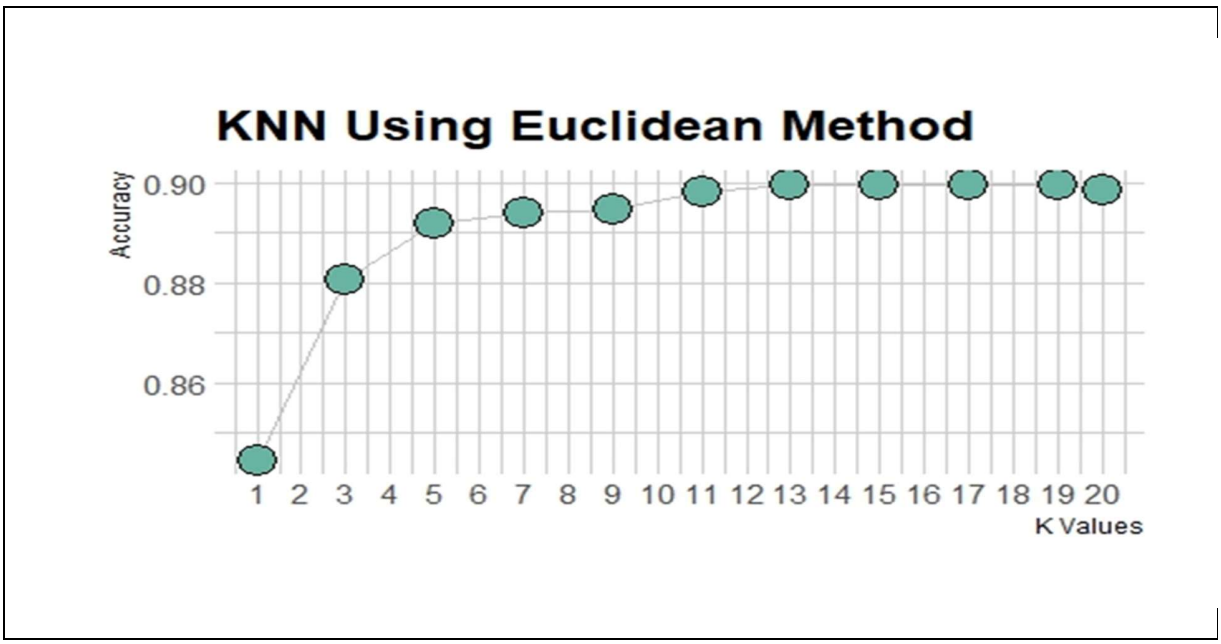
The highest accuracy achieved is 90% with k = 11.

The highest accuracy achieved is 89.95% with k = 13, 15, 17, 19, and 20.

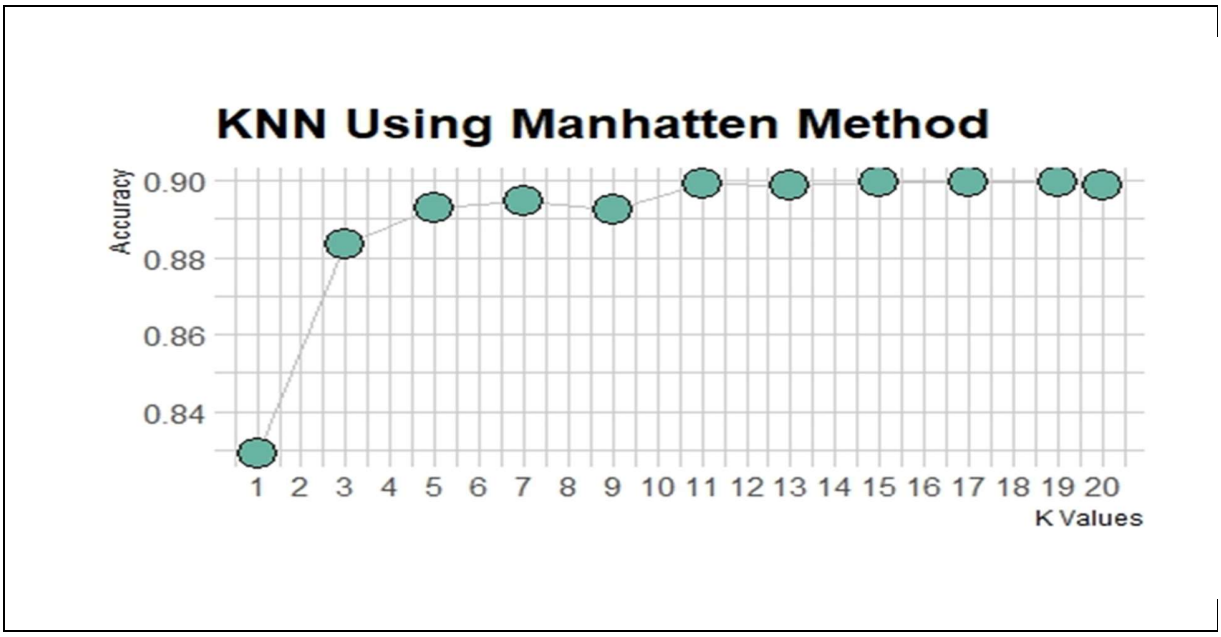
The highest recall achieved is 4.477612% with k = 5.

We can show that the maximum accuracy achieved for the Euclidean distance metric is 0.8995, while the maximum accuracy for both the Manhattan and Max Dimension Distance metrics is 0.8995 as well.

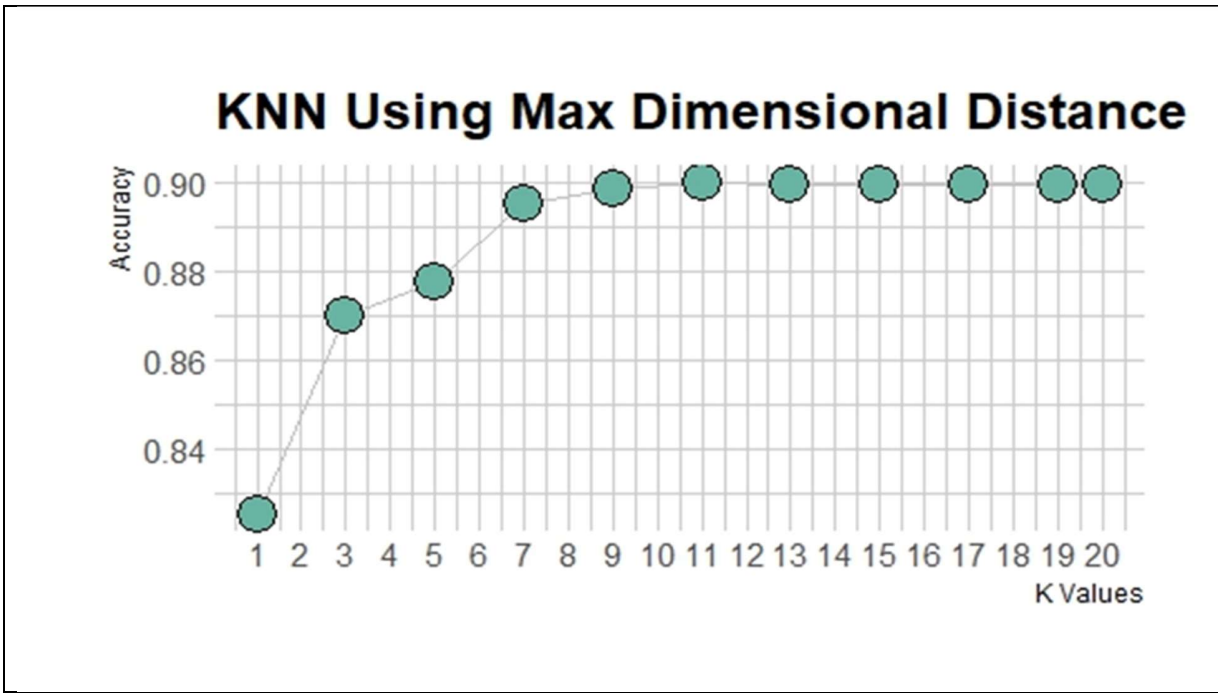
Plotting:



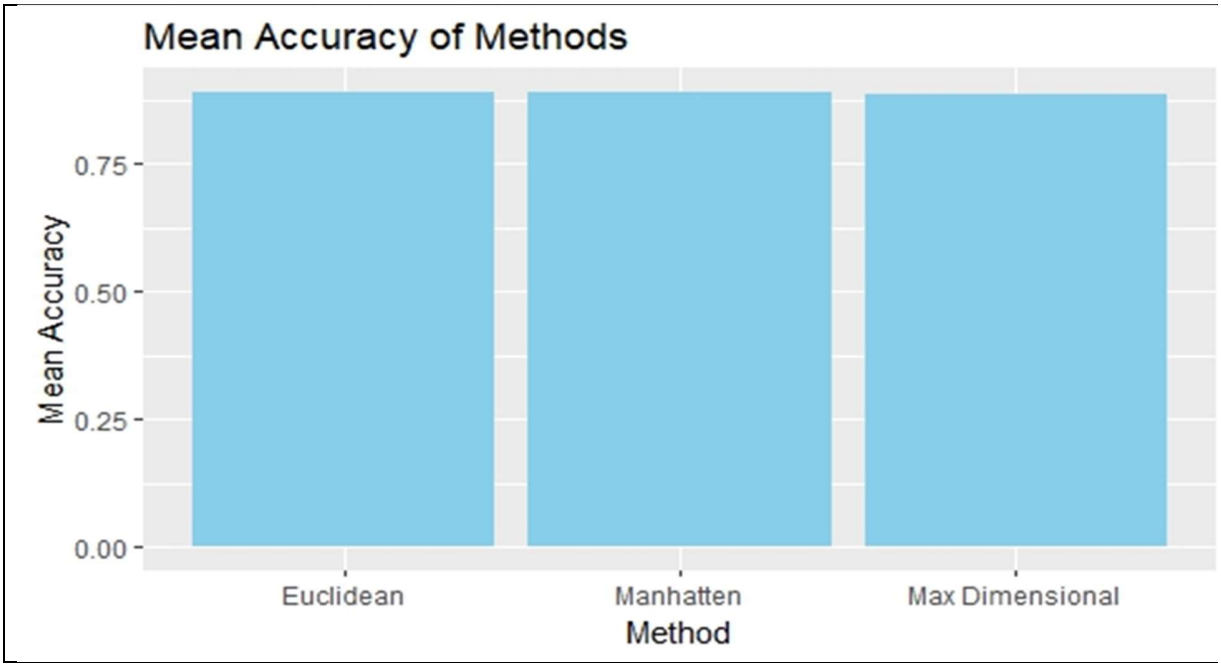
In the plot, the x-axis represents the different values of k, while the y-axis represents the corresponding accuracy values. The line plot shows how the accuracy changes as we vary the value of k. We can observe trends and patterns in the plot, such as increasing or decreasing accuracy with different values of k.



k_values represents the different values of k used in the KNN algorithm with the Manhattan distance metric, and accuracy represents the corresponding accuracy values achieved. You can replace these values with your own data.



In the plot, the x-axis represents the different values of k, while the y-axis represents the corresponding accuracy values. The line plot shows how the accuracy changes as the value of k varies. By analyzing the plot, you can observe trends and patterns in the accuracy with different values of k.



Limitation: One of the hardware limitations that arose when dealing with feature reduction techniques in data our projects is the computational complexity. Feature reduction involves analyzing and manipulating large amounts of data, which are computationally expensive and time-consuming.

Conclusion: In this "Banking Dataset Classification" project we implemented the K-Nearest Neighbors (KNN) algorithm to predict whether a client will subscribe to a term deposit. The KNN algorithm, a popular machine learning algorithm for classification tasks, was applied to classify clients as potential subscribers or non-subscribers to a term deposit. The dataset contained valuable client information, which was preprocessed and divided into training and testing sets. The KNN model accurately classified clients based on their nearest neighbors in the feature space. Evaluation metrics such as accuracy and recall measured the model's performance. The project's predictive model can assist banks in targeting marketing campaigns effectively and improving their success rate.