Dataset Overview:

The sinking of the Titanic is one of the most infamous shipwrecks in history.
On April 15, 1912, during her maiden voyage, the widely considered “unsinkable” RMS Titanic sank after colliding with an iceberg. Unfortunately, there weren’t enough lifeboats for everyone onboard, resulting in the death of 1502 out of 2224 passengers and crew.
While there was some element of luck involved in surviving, it seems some groups of people were more likely to survive than others.
Train.csv will contain the details of a subset of the passengers on board (891 to be exact) and importantly, will reveal whether they survived or not, also known as the “ground truth”.
The test.csv dataset contains similar information but does not disclose the “ground truth” for each passenger.

Key Features:

pclass: A proxy for socio-economic status (SES)
1st = Upper
2nd = Middle
3rd = Lower

age: Age is fractional if less than 1. If the age is estimated, is it in the form of xx.5

sibsp: The dataset defines family relations in this way...
Sibling = brother, sister, stepbrother, stepsister
Spouse = husband, wife (mistresses and fiancés were ignored)

parch: The dataset defines family relations in this way...
Parent = mother, father
Child = daughter, son, stepdaughter, stepson
Some children travelled only with a nanny, therefore parch=0 for them.

Survived: Survived (contains your binary predictions: 1 for survived, 0 for deceased)

Input :
Load the data:

```
# Library

library(tidyverse)      # collection of best packages
library(caret)          # machine learning functions
library(MLmetrics)      # machine learning metrics
library(car)            # VIF calculation
library(class)          # k-NN


# Dataset

train <- read.csv('train.csv', na.strings=c('', 'NA'))
test <- read.csv('test.csv', na.strings=c('', 'NA'))
PassengerId <- test$PassengerId
glimpse(train)
dim(train)

# Data Cleaning
anyDuplicated(train)

#Checking null values

colSums(is.na(train))
colSums(is.na(test))

# Handling null values

#1. `Cabin`

unique(train$Cabin)

train$Cabin <- replace_na(train$Cabin, 'X0')
test$Cabin <- replace_na(test$Cabin, 'X0')


#2. `Age`

train$Surname <- sapply(str_split(train$Name, ','), `[`, 1) %>% str_trim()
temp <- sapply(str_split(train$Name, ','), `[`, 2)
```

```
train$Title <- sapply(str_split(temp, '\\.'), `[`, 1) %>% str_trim()
train <- train %>% select(-Name)

test$Surname <- sapply(str_split(test$Name, ','), `[`, 1) %>% str_trim()
temp <- sapply(str_split(test$Name, ','), `[`, 2)
test$Title <- sapply(str_split(temp, '\\.'), `[`, 1) %>% str_trim()
test <- test %>% select(-Name)

unique(train$Title)
unique(test$Title)


test[test$Title == 'Dona', 'Title'] = 'Mrs'


age_by_title <- train %>%
  group_by(Title) %>%
  summarise(median = median(Age, na.rm = TRUE))

train <- merge(train, age_by_title)
train[is.na(train$Age), 'Age'] <- train[is.na(train$Age), 'median']
train <- train %>% select(-median)

test <- merge(test, age_by_title)
test[is.na(test$Age), 'Age'] <- test[is.na(test$Age), 'median']
test <- test %>% select(-median)


#3. `Embarked`


table(train$Embarked)


train$Embarked <- replace_na(train$Embarked, 'S')
test$Embarked <- replace_na(test$Embarked, 'S')


#4. `Fare`

fare_by_pclass <- train %>%
  group_by(Pclass) %>%
  summarise(median = median(Fare, na.rm = TRUE))

train <- merge(train, fare_by_pclass)
train[is.na(train$Fare), 'Fare'] <- train[is.na(train$Fare), 'median']
train <- train %>% select(-median)
```

```
test <- merge(test, fare_by_pclass)
test[is.na(test$Fare), 'Fare'] <- test[is.na(test$Fare), 'median']
test <- test %>% select(-median)


colSums(is.na(train))
colSums(is.na(test))
```

After handling missing values there are no missing values

```
train <- train %>%
  mutate_at(vars(Pclass, Title, Survived, Sex, Cabin, Embarked), as.factor)

test <- test %>%
  mutate_at(vars(Pclass, Title, Survived, Sex, Cabin, Embarked), as.factor)

glimpse(train)
```

Metrics, Validation, and Class Imbalance

```
prop.table(table(train$Survived))
```

Modeling

```
train <- train %>% select(-c(Cabin))
test <- test %>% select(-c(Cabin))
head(train)
```

k-Nearest Neighbors

Data normalization

```
train_scaled <- scale(x = train %>% select(c('Age', 'SibSp', 'Parch', 'Fare')))
test_scaled <- scale(x = test %>% select(c('Age', 'SibSp', 'Parch', 'Fare')),
           center = attr(train_scaled, "scaled:center"),
           scale = attr(train_scaled, "scaled:scale"))
head(train_scaled)
pred_cols <- train_scaled[,c('Age', 'SibSp', 'Parch', 'Fare')]
head(pred_cols)
target_col <- train$Survived
head(target_col)


# Set the random seed for reproducibility
set.seed(42)
```

```
# Create the training and test indices
train_indices <- createDataPartition(target_col, p = 0.8, list = FALSE)


# Split the data into training and test sets

train_data <- pred_cols[train_indices, ]
train_labels <- target_col[train_indices]
test_data <- pred_cols[-train_indices, ]
test_labels <- target_col[-train_indices]

# View the head of the training data and target
head(train_data)
head(train_target)

# View the head of the test data and target
head(test_data)
head(test_target)


#KNN

knn_with_distance_measure<-
function(train_data,test_data,train_labels,k,distance_measure){
  predicted_labels<-
knn(train=train_data,test=test_data,cl=train_labels,k=k,prob=TRUE,use.all=TRUE)
  return(predicted_labels)
}

#Set the values of k

k_values<-c(3,5,7)

#  Vector to store accuracy
accuracies <- vector()

# Apply KNN
for (k in k_values){
  #Apply KNN with Euclidean distance
  euc_pred <- knn_with_distance_measure(train_data ,test_data ,train_labels,k,"euclidean")

  #Manhattan
  man_pred <- knn_with_distance_measure(train_data ,test_data
,train_labels,k,"manhattan")

  #Maximum distance
  max_pred <- knn_with_distance_measure(train_data ,test_data ,train_labels,k,"maximum")
```

```
  #Evaluate accuracy
  accuracy_euclidean<-sum(euc_pred==test_labels)/length(test_labels)
  accuracy_manhattan<-sum(man_pred==test_labels)/length(test_labels)
  accuracy_maximum<-sum(max_pred==test_labels)/length(test_labels)

  # Store accuracy
  accuracies <- c(accuracies,accuracy_euclidean,accuracy_manhattan,accuracy_maximum)

  #Print the accuracy for the current k value
  cat("Accuracy for k=",k,"\n")
  cat("Euclidean Distance:",accuracy_euclidean,"\n")
  cat("Manhattan Distance:",accuracy_manhattan,"\n")
  cat("Maximum Distance:",accuracy_maximum,"\n")
  cat("\n")

}

#Create a data frame for accuracies
accuracy_df<-data.frame(Distance=rep(c("Euclidean","Manhattan","Maximum"),
                        length(k_values)),K=rep(k_values,each=3),Accuracy=accuracies)


ggplot(accuracy_df,aes(x=K,y=Accuracy,color=Distance,group=Distance))+
  geom_line()+
  geom_point()+
  labs(title="Accuracy of k-NN with Different Distance Measures",
       x="k",
       y="Accuracy",
       color="Distance Measure")+
  theme_minimal()
```

Output:

1.

```
> glimpse(train)
Rows: 891
Columns: 12
$ PassengerId <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34…
$ Survived    <int> 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0,…
$ Pclass      <int> 3, 1, 3, 1, 3, 3, 1, 3, 3, 2, 3, 1, 3, 3, 3, 2, 3, 2, 3, 3, 2, 2, 3, 1, 3, 3, 3, 1, 3, 3, 1, 1, 3, 2, 1, 1, 3, 3, 3, 3, 3, 2,…
$ Name        <chr> "Braund, Mr. Owen Harris", "Cumings, Mrs. John Bradley (Florence Briggs Thayer)", "Heikkinen, Miss. Laina", "Futrelle, Mrs. J…
$ Sex         <chr> "male", "female", "female", "female", "male", "male", "male", "male", "female", "female", "female", "female", "male", "male",…
$ Age         <dbl> 22, 38, 26, 35, 35, NA, 54, 2, 27, 14, 4, 58, 20, 39, 14, 55, 2, NA, 31, NA, 35, 34, 15, 28, 8, 38, NA, 19, NA, NA, 40, NA, N…
$ SibSp       <int> 1, 1, 0, 1, 0, 0, 0, 3, 0, 1, 1, 0, 0, 1, 0, 0, 4, 0, 1, 0, 0, 0, 0, 0, 3, 1, 0, 3, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 2, 1, 1, 1,…
$ Parch       <int> 0, 0, 0, 0, 0, 0, 0, 1, 2, 0, 1, 0, 0, 5, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 5, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,…
$ Ticket      <chr> "A/5 21171", "PC 17599", "STON/O2. 3101282", "113803", "373450", "330877", "17463", "349909", "347742", "237736", "PP 9549", …
$ Fare        <dbl> 7.2500, 71.2833, 7.9250, 53.1000, 8.0500, 8.4583, 51.8625, 21.0750, 11.1333, 30.0708, 16.7000, 26.5500, 8.0500, 31.2750, 7.85…
$ Cabin       <chr> NA, "C85", NA, "C123", NA, NA, "E46", NA, NA, NA, "G6", "C103", NA, NA, NA, NA, NA, NA, NA, NA, NA, "D56", NA, "A6", NA, NA, …
$ Embarked    <chr> "S", "C", "S", "S", "S", "Q", "S", "S", "S", "C", "S", "S", "S", "S", "S", "S", "Q", "S", "S", "C", "S", "S", "Q", "S", "S", …
> dim(train)
[1] 891  12
>
```

2.

```
> #Checking null values
>
> colSums(is.na(train))
PassengerId    Survived      Pclass        Name         Sex         Age       SibSp       Parch      Ticket        Fare       Cabin    Embarked
          0           0           0           0           0         177           0           0           0           0         687           2
> colSums(is.na(test))
PassengerId      Pclass        Name         Sex         Age       SibSp       Parch      Ticket        Fare       Cabin    Embarked
          0           0           0           0          86           0           0           0           1         327           0
>
```

3.After removing null values

```
> colSums(is.na(train))
     Pclass       Title PassengerId    Survived         Sex         Age       SibSp       Parch      Ticket        Fare       Cabin    Embarked
          0           0           0           0           0           0           0           0           0           0           0           0
    Surname
          0
> colSums(is.na(test))
     Pclass       Title PassengerId         Sex         Age       SibSp       Parch      Ticket        Fare       Cabin    Embarked     Surname
          0           0           0           0           0           0           0           0           0           0           0           0
>
```

4.

```
> head(train)
  Pclass Title PassengerId Survived  Sex Age SibSp Parch     Ticket    Fare Cabin Embarked          Surname
1      1  Capt         746        0 male  70     1     1 WE/P 5735 71.0000   B22        S           Crosby
2      1   Col         695        0 male  60     0     0    113800 26.5500    X0        S             Weir
3      1   Col         648        1 male  56     0     0     13213 35.5000   A26        C  Simonius-Blumer
4      1   Don          31        0 male  40     0     0  PC 17601 27.7208    X0        C        Uruchurtu
5      1    Dr         633        1 male  32     0     0     13214 30.5000   B50        C Stahelin-Maeglin
6      1    Mr         371        1 male  25     1     0     11765 55.4417   E50        C           Harder
>
```

5.

```
>
> prop.table(table(train$Survived))

        0         1
0.6161616 0.3838384
>
```

6.

```
> head(train_scaled)
           Age     SibSp      Parch        Fare
[1,]  3.0613503  0.4325504  0.7671990  0.78070266
[2,]  2.3075051 -0.4742788 -0.4734077 -0.11378180
[3,]  2.0059670 -0.4742788 -0.4734077  0.06632249
[4,]  0.7998146 -0.4742788 -0.4734077 -0.09022135
[5,]  0.1967384 -0.4742788 -0.4734077 -0.03429443
[6,] -0.3309533  0.4325504 -0.4734077  0.46761700
> pred_cols <- train_scaled[,c('Age', 'SibSp', 'Parch', 'Fare')]
> head(pred_cols)
           Age     SibSp      Parch        Fare
[1,]  3.0613503  0.4325504  0.7671990  0.78070266
[2,]  2.3075051 -0.4742788 -0.4734077 -0.11378180
[3,]  2.0059670 -0.4742788 -0.4734077  0.06632249
[4,]  0.7998146 -0.4742788 -0.4734077 -0.09022135
[5,]  0.1967384 -0.4742788 -0.4734077 -0.03429443
[6,] -0.3309533  0.4325504 -0.4734077  0.46761700
> target_col <- train$Survived
```

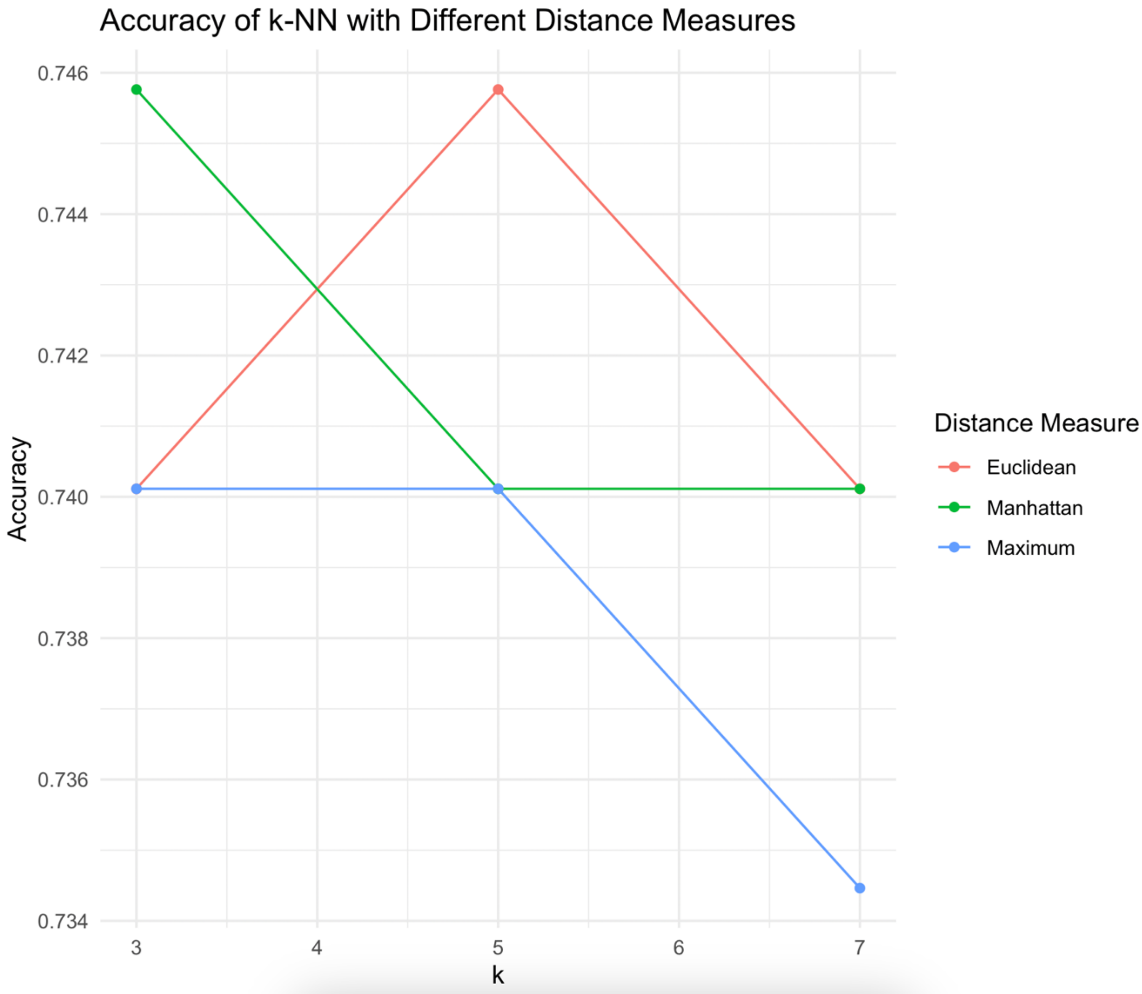
7.

```
> head(train_data)
           Age     SibSp      Parch        Fare
[1,]  3.0613503  0.4325504  0.7671990  0.78070266
[2,]  2.3075051 -0.4742788 -0.4734077 -0.11378180
[3,]  2.0059670 -0.4742788 -0.4734077  0.06632249
[4,]  0.7998146 -0.4742788 -0.4734077 -0.09022135
[5,] -0.3309533  0.4325504 -0.4734077  0.46761700
[6,]  1.4782753 -0.4742788 -0.4734077 -0.12627440
> head(train_labels)
[1] 0 0 1 0 1 1
Levels: 0 1
>
> # View the head of the test data and target
> head(test_data)
          Age      SibSp      Parch        Fare
[1,] 0.1967384 -0.4742788 -0.4734077 -0.03429443
[2,] 3.1367349 -0.4742788 -0.4734077  0.34813440
[3,] 0.6490455 -0.4742788 -0.4734077 -0.64805768
[4,] 1.4028908  0.4325504 -0.4734077  0.14882837
[5,] 0.7998146 -0.4742788 -0.4734077 -0.64805768
[6,] 0.5736610 -0.4742788  0.7671990 -0.05039314
> head(test_labels)
[1] 1 0 0 1 0 0
Levels: 0 1
>
```

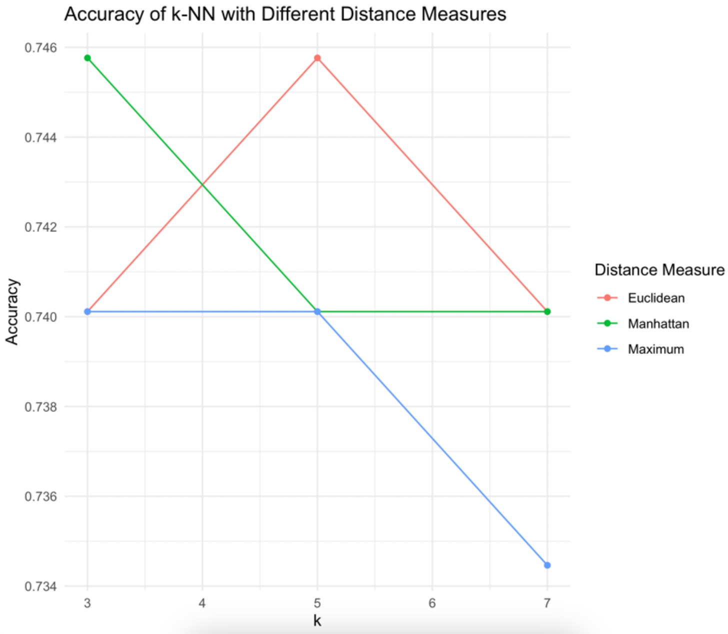
8.

```
Accuracy for k= 3
Euclidean Distance: 0.740113
Manhattan Distance: 0.7457627
Maximum Distance: 0.740113

Accuracy for k= 5
Euclidean Distance: 0.7457627
Manhattan Distance: 0.740113
Maximum Distance: 0.740113

Accuracy for k= 7
Euclidean Distance: 0.740113
Manhattan Distance: 0.740113
Maximum Distance: 0.7344633

>
```

9.


Accuracy of k-NN with Different Distance Measures
0.746
0.744
0.742
0.740
0.738
0.736
0.734
Accuracy
3
4
5
6
7
k
Distance Measure
Euclidean
Manhattan
Maximum

Discussion and Analysis:



The graph above shows how the accuracy of the knn algorithm with different values of k on using the 3 distance measuring methods (Euclidean, Manhattan, and Maximum Dimension).

It can be seen that the accuracy was the highest when the value of k was 3 in manhattan and maximum distance measure method. In this instance, Manhattan distance had the highest accuracy.

When the value of k was increased to 5, the accuracies of all the distance methods dropped, except euclidean but the Maximum dimension was the highest. For value k=5 the accuracy to Euclidean distance is high.
In the end, when the value of k was set to 7, the accuracy dropped again. However, this time, Manhattan and Eucledian had the same accuracies and maximum dimension had the lowest accuracy.

From this observation we have found that most accurate value for k is 3. Most accurate value for the method is Euclidean and Manhattan .
Based on this findings , it can be said that the highest accuracy for the value of k(3) which is Manhattan distance .