INTRODUCTION TO DATA SCIENCE

FINALTERM PROJECT REPORT

SUBMITTED BY

SADIR AHMED ZIDAN

ID:20-43263-1

SECTION:C

SUBMITTED TO

DR. ABDUS SALAM

Assistant Professor, CS

Sleep Health Lifestyle Dataset: The 400 rows and 13 columns of the Sleep Health and Lifestyle Dataset include a wide range of factors relating to sleep and daily activities. It covers details such as gender, age, occupation, amount and quality of sleep, physical activity level, stress level, BMI category, blood pressure, heart rate, number of steps taken each day, and whether sleep disorders exist or not. Important features of the dataset include-

1.Explore sleep duration, quality, and the variables affecting sleep patterns with comprehensive sleep metrics.
2.Analysis of physical activity levels, stress levels, and BMI categories as lifestyle factors.
3.Examine heart rate and blood pressure readings to determine cardiovascular health.
4.Analysis of Sleep Disorders: Determine the prevalence of sleep disorders such insomnia and sleep apnea.

<u>The dataset's columns include:</u>
Person ID: An individual ID to identify each person specifically.
Gender: The male or female gender of the subject.
Age: The person's age expressed in years.
Occupation: Profession of the individual.
Sleep Duration: The number of hours a person sleeps each day is measured.
Quality of Sleep: A rating of the sleeper's quality on a scale from 1 to 10.
Physical Activity Level: The amount of time spent moving each day, expressed in minutes.
Stress Level: A subjective assessment of the level of stress on a scale from 1 to 10.
BMI Category: The individual's BMI category, such as underweight, normal weight, or overweight.
Blood pressure (systolic/diastolic): The measurement of a person's blood pressure.
Heart Rate (bpm): The individual's average resting heart rate in beats per minute.
Daily Steps: The total number of steps a person take each day.
Sleep Disorder: Indicates whether a sleep condition is present or absent (None, Insomnia, Sleep Apnea).

DATA PREPROCESSING:

- ❑ **Reading the dataset file in csv format and printing the attributes type to check the names and datatypes of the attributes and viewing the dataset to check how it actually looks like.**

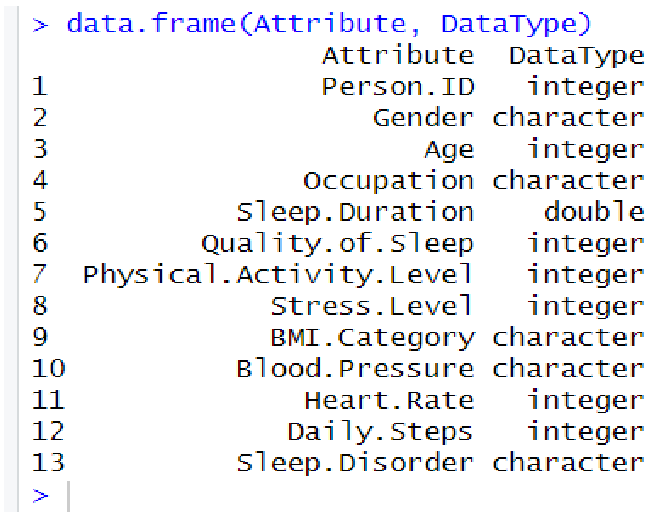
<u>CODE</u>:

```
SHL_Dataset <- read.csv("F:/Data Science/sleep health lifestyle
dataset/Sleep_health_and_lifestyle_dataset.csv",header = TRUE,sep = ",")
names(SHL_Dataset)
Attribute <- names(SHL_Dataset)
DataType <- c(typeof(SHL_Dataset$Person.ID), typeof(SHL_Dataset$Gender),
typeof(SHL_Dataset$Age),typeof(SHL_Dataset$Occupation),typeof(SHL_Dataset$Sleep.Dura
tion), typeof(SHL_Dataset$Quality.of.Sleep), typeof(SHL_Dataset$Physical.Activity.Level),
typeof(SHL_Dataset$Stress.Level),typeof(SHL_Dataset$BMI.Category),typeof(SHL_Dataset$
```

Blood.Pressure), typeof(SHL_Dataset$Heart.Rate), typeof(SHL_Dataset$Daily.Steps), typeof(SHL_Dataset$Sleep.Disorder))
data.frame(Attribute, DataType)
View(SHL_Dataset)

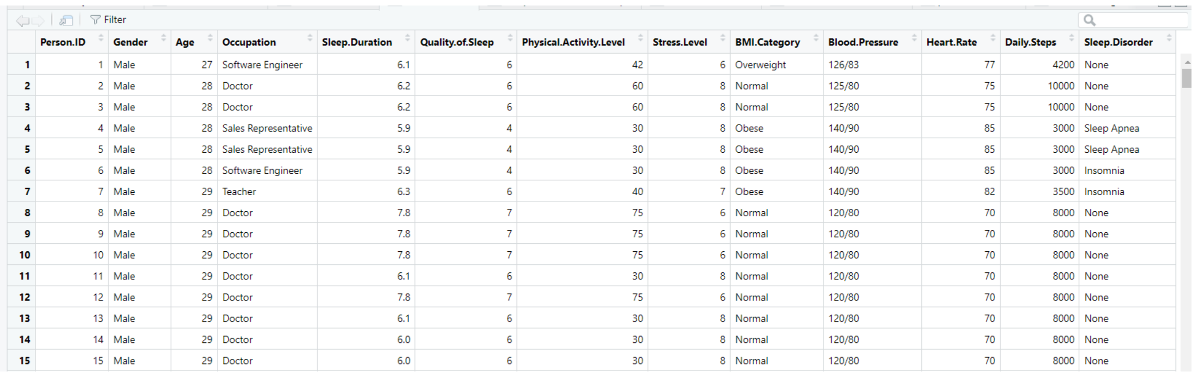
<u>OUTPUT</u>:

Attributes names and Datatypes



```
> data.frame(Attribute, DataType)
                Attribute  DataType
1               Person.ID   integer
2                  Gender character
3                     Age   integer
4              Occupation character
5          Sleep.Duration    double
6        Quality.of.Sleep   integer
7 Physical.Activity.Level   integer
8            Stress.Level   integer
9            BMI.Category character
10         Blood.Pressure character
11             Heart.Rate   integer
12            Daily.Steps   integer
13         Sleep.Disorder character
> 
```

Dataset View



Filter

	Person.ID	Gender	Age	Occupation	Sleep.Duration	Quality.of.Sleep	Physical.Activity.Level	Stress.Level	BMI.Category	Blood.Pressure	Heart.Rate	Daily.Steps	Sleep.Disorder
1	1	Male	27	Software Engineer	6.1	6	42	6	Overweight	126/83	77	4200	None
2	2	Male	28	Doctor	6.2	6	60	8	Normal	125/80	75	10000	None
3	3	Male	28	Doctor	6.2	6	60	8	Normal	125/80	75	10000	None
4	4	Male	28	Sales Representative	5.9	4	30	8	Obese	140/90	85	3000	Sleep Apnea
5	5	Male	28	Sales Representative	5.9	4	30	8	Obese	140/90	85	3000	Sleep Apnea
6	6	Male	28	Software Engineer	5.9	4	30	8	Obese	140/90	85	3000	Insomnia
7	7	Male	29	Teacher	6.3	6	40	7	Obese	140/90	82	3500	Insomnia
8	8	Male	29	Doctor	7.8	7	75	6	Normal	120/80	70	8000	None
9	9	Male	29	Doctor	7.8	7	75	6	Normal	120/80	70	8000	None
10	10	Male	29	Doctor	7.8	7	75	6	Normal	120/80	70	8000	None
11	11	Male	29	Doctor	6.1	6	30	8	Normal	120/80	70	8000	None
12	12	Male	29	Doctor	7.8	7	75	6	Normal	120/80	70	8000	None
13	13	Male	29	Doctor	6.1	6	30	8	Normal	120/80	70	8000	None
14	14	Male	29	Doctor	6.0	6	30	8	Normal	120/80	70	8000	None
15	15	Male	29	Doctor	6.0	6	30	8	Normal	120/80	70	8000	None

- ❑ **As I'll be applying kNN classification algorithm to this dataset, I need to convert all the categorical values to numeric values. In the above section, I found that there are 5 attributes with categorical values. Here is the code and output of converting their values into numerical values.**

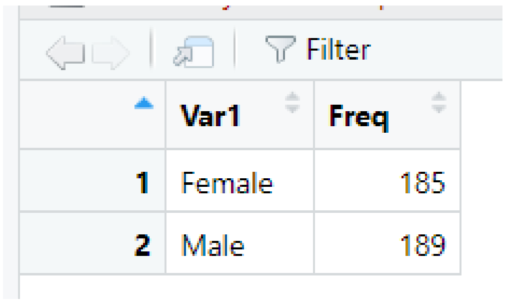
❖ **For Gender**

<u>CODE</u>:

Checking Actual frequency of Gender

Gender_Actual_Value_freq <- table(SHL_Dataset$Gender)
View(Gender_Actual_Value_freq)

OUTPUT:
Gender Actual Frequency



	Var1	Freq
1	Female	185
2	Male	189

- **Here, Gender has only 2 types of values as Male and Female. So, I'll convert Male as 1 and Female as 0.**

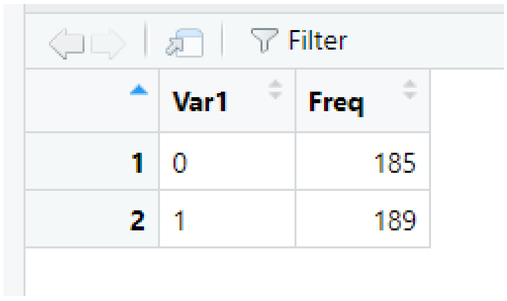
CODE:
Changing Male and Female to 1 and 0

```
SHL_Dataset_mod$Gender <- ifelse(SHL_Dataset_mod$Gender == "Male", 1, 0)
```

Checking changed Frequency

```
Gender_Changed_Value_freq <- table(SHL_Dataset_mod$Gender)
View(Gender_Changed_Value_freq)
```

OUTPUT:



	Var1	Freq
1	0	185
2	1	189

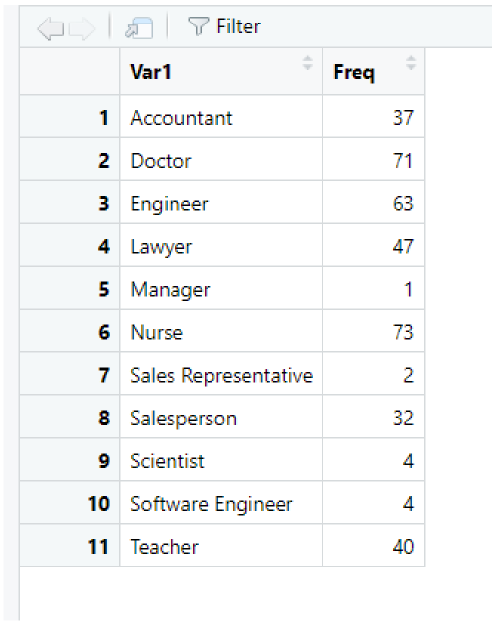
- **Here, Gender variable value is changed to 0 and 1.**

❖ **For Occupation**

CODE:
Checking Actual Frequency of Occupation

```
Occupation_actual_freq <- table(SHL_Dataset_mod$Occupation)
View(Occupation_actual_freq)
```

OUTPUT:



Filter

	Var1	Freq
1	Accountant	37
2	Doctor	71
3	Engineer	63
4	Lawyer	47
5	Manager	1
6	Nurse	73
7	Sales Representative	2
8	Salesperson	32
9	Scientist	4
10	Software Engineer	4
11	Teacher	40

- **Here, Occupation variable has 11 difference values. So, I'll be converting them into numeric values.**

CODE:

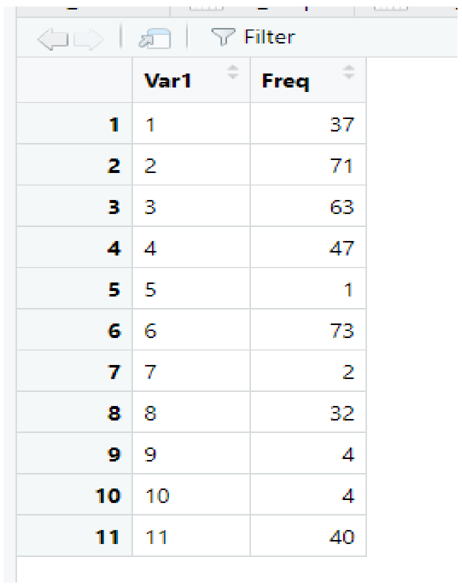
Changing into numeric values

```
SHL_Dataset_mod <- SHL_Dataset_mod %>%
  mutate_at(vars(Occupation), as.factor) %>%
  mutate(across(Occupation, as.integer))
```

Checking changed frequency of Occupation

```
Occupation_Changed_Value_freq <- table(SHL_Dataset_mod$Occupation)
View(Occupation_Changed_Value_freq)
```

OUTPUT:



Filter

	Var1	Freq
1	1	37
2	2	71
3	3	63
4	4	47
5	5	1
6	6	73
7	7	2
8	8	32
9	9	4
10	10	4
11	11	40

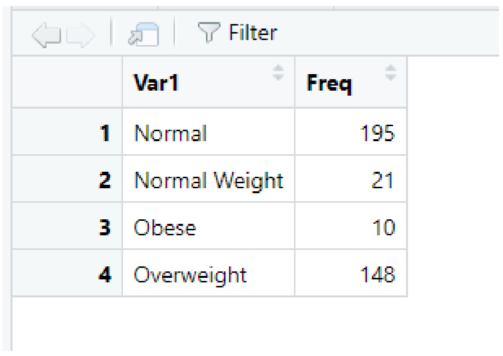
❖ **For BMI Category**

CODE:

Checking Actual Frequency of BMI Category

```
BMI_actual_freq <- table(SHL_Dataset_mod$BMI.Category)
View(BMI_actual_freq)
```

OUTPUT:



	Var1	Freq
1	Normal	195
2	Normal Weight	21
3	Obese	10
4	Overweight	148

- **Here, BMI Category variable has 4 difference values. So, I'll be converting them into numeric values.**

CODE:

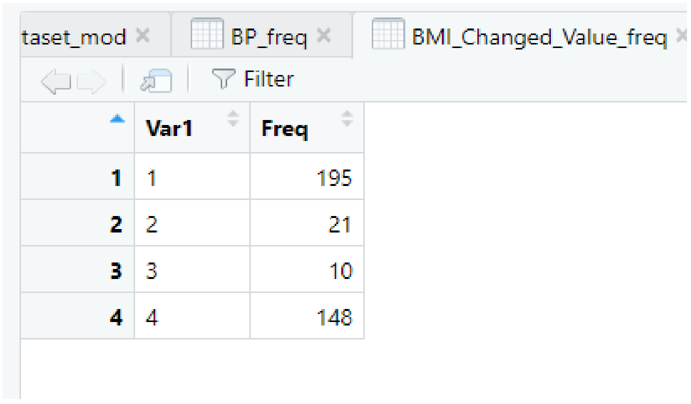
Changing into numeric values

```
SHL_Dataset_mod <- SHL_Dataset_mod %>%
  mutate_at(vars(BMI.Category), as.factor) %>%
  mutate(across(BMI.Category, as.integer))
```

Checking changed frequency of BMI Category

```
BMI_Changed_Value_freq <- table(SHL_Dataset_mod$BMI.Category)
View(BMI_Changed_Value_freq)
```

OUTPUT:



	Var1	Freq
1	1	195
2	2	21
3	3	10
4	4	148

- **Here, BMI Category variable has changed into numeric values.**

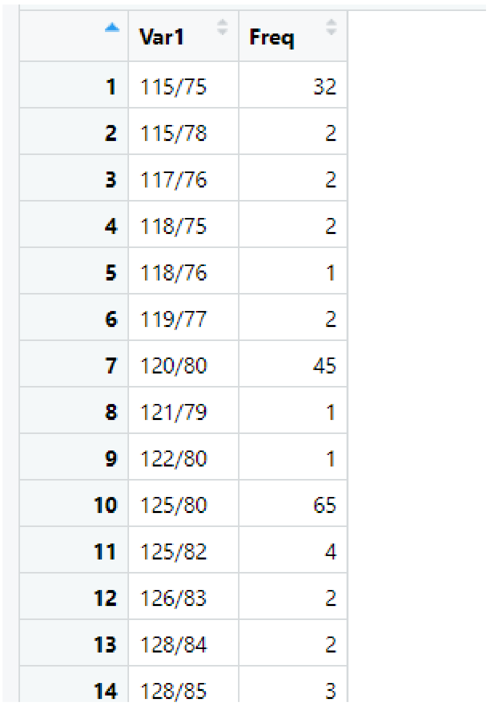
❖ **For Blood Pressure: As blood pressure is given as systolic and diastolic rate, I have extracted systolic and diastolic components first and then converted them into blood pressure types. Then converted to numeric values as follows.**

<u>**CODE:**</u>

Checking Actual Frequency of Blood Pressure

```
Blood_Pressure_freq <- table(SHL_Dataset_mod$Blood.Pressure)
View(Blood_Pressure_freq)
```

<u>**OUTPUT:**</u>



	Var1	Freq
1	115/75	32
2	115/78	2
3	117/76	2
4	118/75	2
5	118/76	1
6	119/77	2
7	120/80	45
8	121/79	1
9	122/80	1
10	125/80	65
11	125/82	4
12	126/83	2
13	128/84	2
14	128/85	3

- **Here, Blood Pressure variable has 14 difference values. So, I'll be converting them into numeric values. As all the values are between normal and Stage 2 hypertension, I have converted them into 4 numeric values as 120/80>Normal, 140/90<=Stage 1 Hypertension, 160/100<=Stage 2 Hypertension and the rest of the rates are Pre-Hypertension. Where the first one is systolic and second one is diastolic. So, Firstly I have categorized these values into 4 stages.**

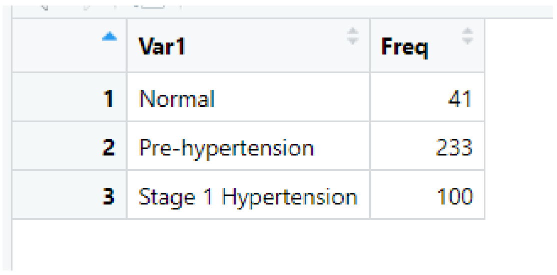
<u>**CODE:**</u>

Extracting Systolic and diastolic components and Categorizing Blood Pressure values

```
blood_pressure_component <- strsplit(SHL_Dataset_mod$Blood.Pressure, "/")
systolic <- as.numeric(sapply(blood_pressure_component, "[[", 1))
diastolic <- as.numeric(sapply(blood_pressure_component, "[[", 2))
```

SHL_Dataset_mod$Blood.Pressure <- ifelse(systolic < 120 & diastolic < 80, "Normal", ifelse(systolic >= 160 | diastolic >= 100, "Stage 2 Hypertension", ifelse(systolic >= 140 | diastolic >= 90, "Stage 1 Hypertension", "Pre-hypertension")))

OUTPUT:



	Var1	Freq
1	Normal	41
2	Pre-hypertension	233
3	Stage 1 Hypertension	100

- **Here, Blood Pressure categorized into stages as explained before**

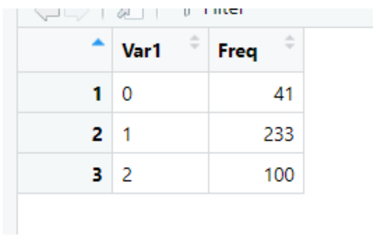
CODE:

Converting categorical Blood Pressure values to numeric

SHL_Dataset_mod$Blood.Pressure <- factor(SHL_Dataset_mod$Blood.Pressure, levels = c ("Normal", "Pre-hypertension", "Stage 1 Hypertension", "Stage 2 Hypertension"), ordered = TRUE)

SHL_Dataset_mod$Blood.Pressure <- as.numeric(SHL_Dataset_mod$Blood.Pressure) - 1

OUTPUT:



	Var1	Freq
1	0	41
2	1	233
3	2	100

❖ **For Sleep Disorder: Here, Sleep Disorder is the class variable, which determines whether a person has sleep disorder or not. Changing its value to numeric.**

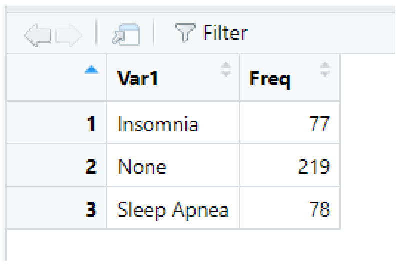
CODE:

Checking Actual Frequency of Sleep Disorder

SD_act_freq <- table(SHL_Dataset$Sleep.Disorder)

View(SD_act_freq)

Output:



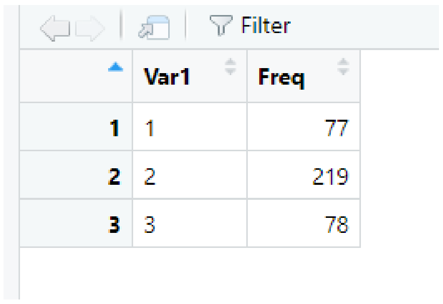
	Var1	Freq
1	Insomnia	77
2	None	219
3	Sleep Apnea	78

Code:

Changing into numeric

```
SHL_Dataset_mod <- SHL_Dataset_mod %>%
  mutate_at(vars(Sleep.Disorder), as.factor) %>%
  mutate(across(Sleep.Disorder, as.integer))
```

Output:



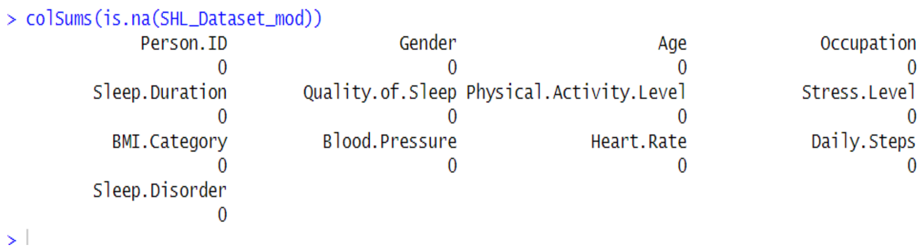
	Var1	Freq
1	1	77
2	2	219
3	3	78

- **Checking if there are any missing values exists in any of the attributes or not.**

CODE:

```
colSums(is.na(SHL_Dataset_mod))
```

OUTPUT:



```
> colSums(is.na(SHL_Dataset_mod))
              Person.ID                  Gender                     Age              Occupation 
                      0                       0                       0                       0 
         Sleep.Duration        Quality.of.Sleep Physical.Activity.Level            Stress.Level 
                      0                       0                       0                       0 
           BMI.Category          Blood.Pressure              Heart.Rate             Daily.Steps 
                      0                       0                       0                       0 
         Sleep.Disorder 
                      0 
> 
```

- No missing values detected.

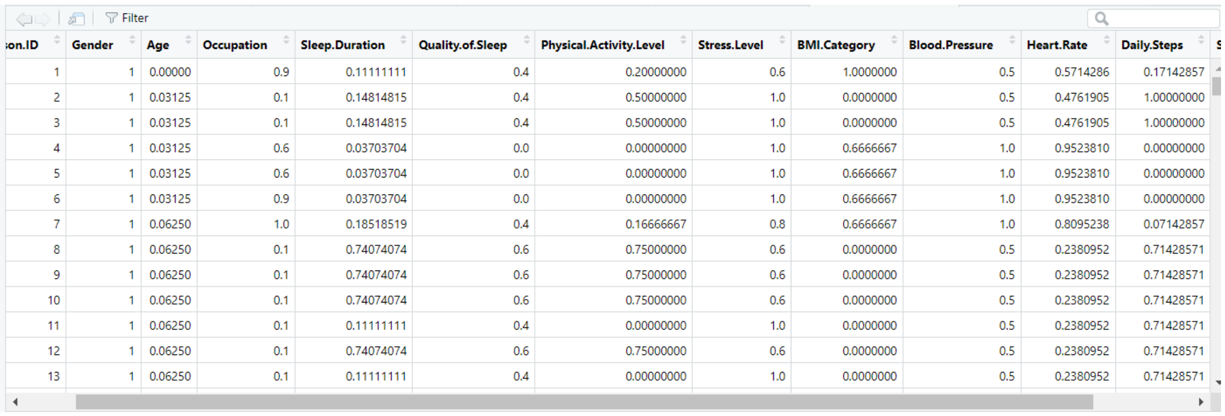
- **Normalizing all the numeric attributes to make all the variable values in a known range. Whereas normalization makes values in between 0 and 1. Min max normalization is used here. Gender variable is skipped because the values are already in 0,1 format. Also, class variable "Sleep Disorder" remains same as it determines class labels.**

CODE:

```
normalize <- function(x) {return((x - min(x)) / (max(x) - min(x)))}
num_of_column<-
c("Age","Occupation","Sleep.Duration","Quality.of.Sleep","Physical.Activity.Level",
          "Stress.Level","BMI.Category","Blood.Pressure","Heart.Rate","Daily.Steps")
```

```
SHL_Dataset_mod[num_of_column] <- lapply(SHL_Dataset_mod[num_of_column],
normalize)
```

OUTPUT:



son.ID	Gender	Age	Occupation	Sleep.Duration	Quality.of.Sleep	Physical.Activity.Level	Stress.Level	BMI.Category	Blood.Pressure	Heart.Rate	Daily.Steps	S
1	1	0.00000	0.9	0.11111111	0.4	0.20000000	0.6	1.0000000	0.5	0.5714286	0.17142857	
2	1	0.03125	0.1	0.14814815	0.4	0.50000000	1.0	0.0000000	0.5	0.4761905	1.00000000	
3	1	0.03125	0.1	0.14814815	0.4	0.50000000	1.0	0.0000000	0.5	0.4761905	1.00000000	
4	1	0.03125	0.6	0.03703704	0.0	0.00000000	1.0	0.6666667	1.0	0.9523810	0.00000000	
5	1	0.03125	0.6	0.03703704	0.0	0.00000000	1.0	0.6666667	1.0	0.9523810	0.00000000	
6	1	0.03125	0.9	0.03703704	0.0	0.00000000	1.0	0.6666667	1.0	0.9523810	0.00000000	
7	1	0.06250	1.0	0.18518519	0.4	0.16666667	0.8	0.6666667	1.0	0.8095238	0.07142857	
8	1	0.06250	0.1	0.74074074	0.6	0.75000000	0.6	0.0000000	0.5	0.2380952	0.71428571	
9	1	0.06250	0.1	0.74074074	0.6	0.75000000	0.6	0.0000000	0.5	0.2380952	0.71428571	
10	1	0.06250	0.1	0.74074074	0.6	0.75000000	0.6	0.0000000	0.5	0.2380952	0.71428571	
11	1	0.06250	0.1	0.11111111	0.4	0.00000000	1.0	0.0000000	0.5	0.2380952	0.71428571	
12	1	0.06250	0.1	0.74074074	0.6	0.75000000	0.6	0.0000000	0.5	0.2380952	0.71428571	
13	1	0.06250	0.1	0.11111111	0.4	0.00000000	1.0	0.0000000	0.5	0.2380952	0.71428571	

CORRELATION:

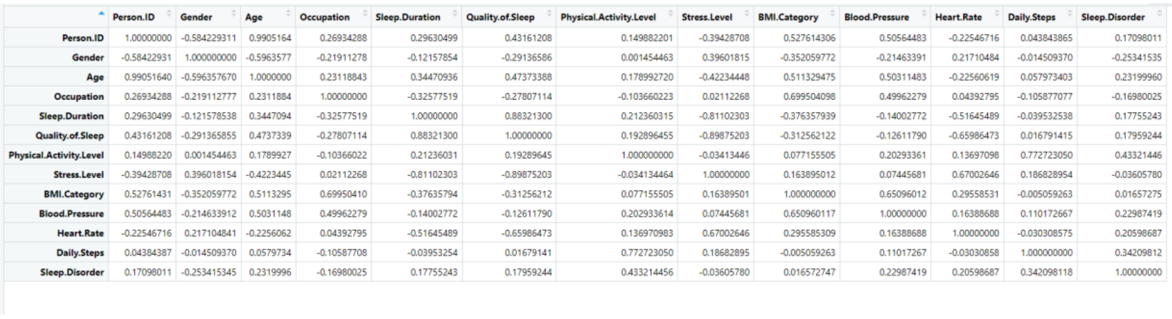
- ❑ **Correlation Matrix is found for each variable in terms of getting the relational value of each variable with another variable and also for finding relation with class variable.**

CODE:

```
correlation_matrix_main <-
cor(SHL_Dataset_mod[c("Person.ID","Gender","Age","Occupation","Sleep.Duration","Quality.of.Sleep","Physical.Activity.Level","Stress.Level","BMI.Category","Blood.Pressure","Heart.Rate","Daily.Steps")], SHL_Dataset_mod$Sleep.Disorder)
View(correlation_matrix_main)
print(correlation_matrix_main)
```

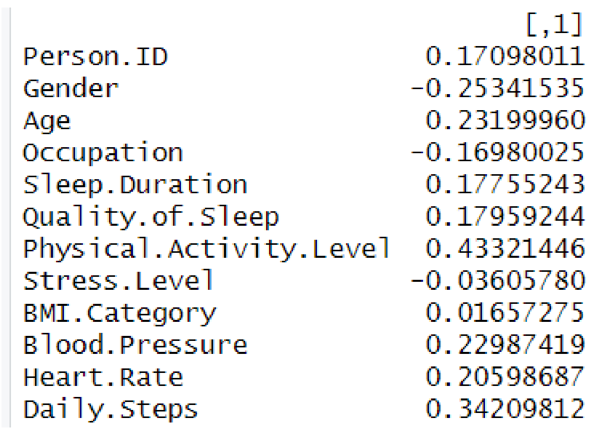
OUTPUT:

Correlation Matrix Table, Shows all the relations of all variables.



	Person.ID	Gender	Age	Occupation	Sleep.Duration	Quality.of.Sleep	Physical.Activity.Level	Stress.Level	BMI.Category	Blood.Pressure	Heart.Rate	Daily.Steps	Sleep.Disorder
Person.ID	1.00000000	-0.584229311	0.9905164	0.26934288	0.29630499	0.43161208	0.149882201	-0.39428708	0.527614306	0.50564483	-0.22546716	0.043843865	0.17098011
Gender	-0.58422931	1.000000000	-0.5963577	-0.21911278	-0.12157854	-0.29136586	0.001454463	0.39601815	-0.352059772	-0.21463391	0.21710484	-0.014509370	-0.25341535
Age	0.99051640	-0.596357670	1.0000000	0.23118843	0.34470936	0.47373388	0.178992720	-0.42234448	0.511329475	0.50311483	-0.22560619	0.057973403	0.23199960
Occupation	0.26934288	-0.219112777	0.2311884	1.00000000	-0.32577519	-0.27807114	-0.103660223	0.02112268	0.699504098	0.49962279	0.04392795	-0.105877077	-0.16980025
Sleep.Duration	0.29630499	-0.121578538	0.3447094	-0.32577519	1.00000000	0.88321300	0.212360315	-0.81102303	-0.376357939	-0.14002772	-0.51645489	-0.039532538	0.17755243
Quality.of.Sleep	0.43161208	-0.291365855	0.4737339	-0.27807114	0.88321300	1.00000000	0.192896455	-0.89875203	-0.312562122	-0.12611790	-0.65986473	0.016791415	0.17959244
Physical.Activity.Level	0.14988220	0.001454463	0.1789927	-0.10366022	0.21236031	0.19289645	1.000000000	-0.03413446	0.077155505	0.20293361	0.13697098	0.772723050	0.43321446
Stress.Level	-0.39428708	0.396018154	-0.4223445	0.02112268	-0.81102303	-0.89875203	-0.034134464	1.00000000	0.163895012	0.07445681	0.67002646	0.186828954	-0.03605780
BMI.Category	0.52761431	-0.352059772	0.5113295	0.69950410	-0.37635794	-0.31256212	0.077155505	0.16389501	1.000000000	0.65096012	0.29558531	-0.005059263	0.01657275
Blood.Pressure	0.50564483	-0.214633912	0.5031148	0.49962279	-0.14002772	-0.12611790	0.202933614	0.07445681	0.650960117	1.00000000	0.16388688	0.110172667	0.22987419
Heart.Rate	-0.22546716	0.217104841	-0.2256062	0.04392795	-0.51645489	-0.65986473	0.136970983	0.67002646	0.295585309	0.16388688	1.00000000	-0.030308575	0.20598687
Daily.Steps	0.04384387	-0.014509370	0.0579734	-0.10587708	-0.03953254	0.01679141	0.772723050	0.18682895	-0.005059263	0.11017267	-0.03030858	1.000000000	0.34209812
Sleep.Disorder	0.17098011	-0.253415345	0.2319996	-0.16980025	0.17755243	0.17959244	0.433214456	-0.03605780	0.016572747	0.22987419	0.20598687	0.342098118	1.00000000

<u>OUTPUT</u>: Correlation of all variables with class variable “Sleep Disorder”.



```
                               [,1]
Person.ID                0.17098011
Gender                  -0.25341535
Age                      0.23199960
Occupation              -0.16980025
Sleep.Duration           0.17755243
Quality.of.Sleep         0.17959244
Physical.Activity.Level  0.43321446
Stress.Level            -0.03605780
BMI.Category             0.01657275
Blood.Pressure           0.22987419
Heart.Rate               0.20598687
Daily.Steps              0.34209812
```

Correlation Matrix Plot, visualizing all variables and relations with values and based on strength using corplot library.

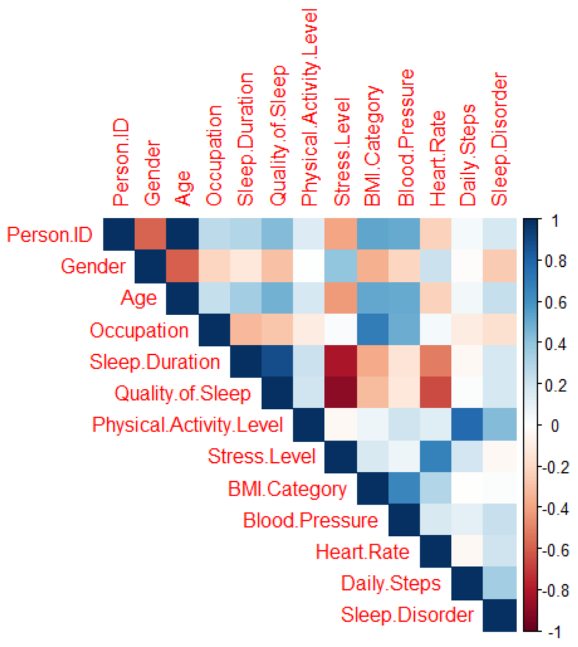
<u>CODE</u>:

install.packages("corrplot")

library(corrplot)

corrplot(correlation_matrix_main, method = "color", type = "upper")

<u>OUTPUT:</u>



- **As we can see, there is no variable which has “no correlation=0” with class variable. So, no attributes have been deleted.**

APPLYING kNN:

- ❑ **As all the variables are now numeric, so building kNN model to apply in the dataset. Class package is used and for kNN, k value is fixed to 5 for taking 5 nearest neighbours to detect accuracy.**

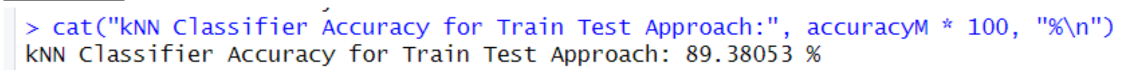
    - **First, finding the accuracy of kNN by dividing the dataset into train-test approach. And taking 70% for train and 30% for test. After that accuracy is measured.**

Train Test Approach

<u>CODE:</u>

```
install.packages("class")
library(class)

set.seed(7)
sample_indicesM <- sample(nrow(SHL_Dataset_mod), nrow(prac_datatset_8) * 0.7)
train_dataM <- SHL_Dataset_mod[sample_indicesM, ]
test_dataM <- SHL_Dataset_mod[-sample_indicesM, ]

featuresFM <- c("Gender", "Age", "Occupation", "Sleep.Duration", "Quality.of.Sleep",
"Physical.Activity.Level","Stress.Level","BMI.Category", "Blood.Pressure", "Heart.Rate",
"Daily.Steps")

train_featuresM <- train_dataM[featuresFM]
train_labelsM <- train_dataM$Sleep.Disorder
test_featuresM <- test_dataM[featuresFM]
test_labelsM <- test_dataM$Sleep.Disorder

k <- 5
predicted_labelsM <- knn(train_featuresM, test_featuresM, train_labelsM, k)

accuracyM <- sum(predicted_labelsM == test_labelsM) / length(test_labelsM)
cat("kNN Classifier Accuracy for Train Test Approach:", accuracyM * 100, "%\n")
print(accuracyM)
```

<u>OUTPUT:</u>



```
> cat("kNN Classifier Accuracy for Train Test Approach:", accuracyM * 100, "%\n")
kNN Classifier Accuracy for Train Test Approach: 89.38053 %
```

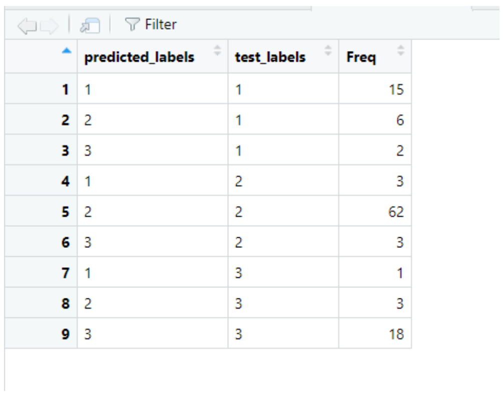
- **Here, the accuracy found for kNN train-test approach is 89.38%.**
- **Now, the confusion matrix is measured. Recall and precision value is also measured. Where confusion matrix shows the predicted and actual labels within a table. By which it can be determined that how many instances were predicted right or how**

the model performed through this approach. And Recall, Precision values will determine the True Positive rate and Positive Predicted value.

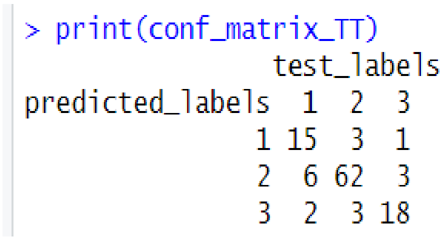
Code:

```
conf_matrix_TT <- table(predicted_labels, test_labels)
print(conf_matrix_TT)
View(conf_matrix_TT)
```

OUTPUT:



	predicted_labels	test_labels	Freq
1	1	1	15
2	2	1	6
3	3	1	2
4	1	2	3
5	2	2	62
6	3	2	3
7	1	3	1
8	2	3	3
9	3	3	18



```
> print(conf_matrix_TT)
                test_labels
predicted_labels  1  2  3
               1 15  3  1
               2  6 62  3
               3  2  3 18
```

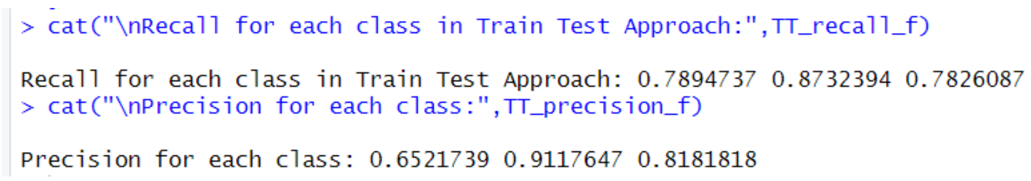
Confusion Matrix and tabular view with frequency values.

- **So, it determines that the model correctly predicted class 1,2,3 for 15, 62 and 18 times.**

CODE:

```
TT_recall_f <- numeric(nrow(conf_matrix_TT))
TT_precision_f <- numeric(nrow(conf_matrix_TT))

for (i in 1:nrow(conf_matrix_TT)) {
  TT_recall_f[i] <- conf_matrix_TT[i, i] / sum(conf_matrix_TT[i, ])
  TT_precision_f[i] <- conf_matrix_TT[i, i] / sum(conf_matrix_TT[, i])
}

cat("\nRecall for each class in Train Test Approach:",TT_recall_f)
cat("\nPrecision for each class:",TT_precision_f)
```

OUTPUT:



```
> cat("\nRecall for each class in Train Test Approach:",TT_recall_f)

Recall for each class in Train Test Approach: 0.7894737 0.8732394 0.7826087
> cat("\nPrecision for each class:",TT_precision_f)

Precision for each class: 0.6521739 0.9117647 0.8181818
```

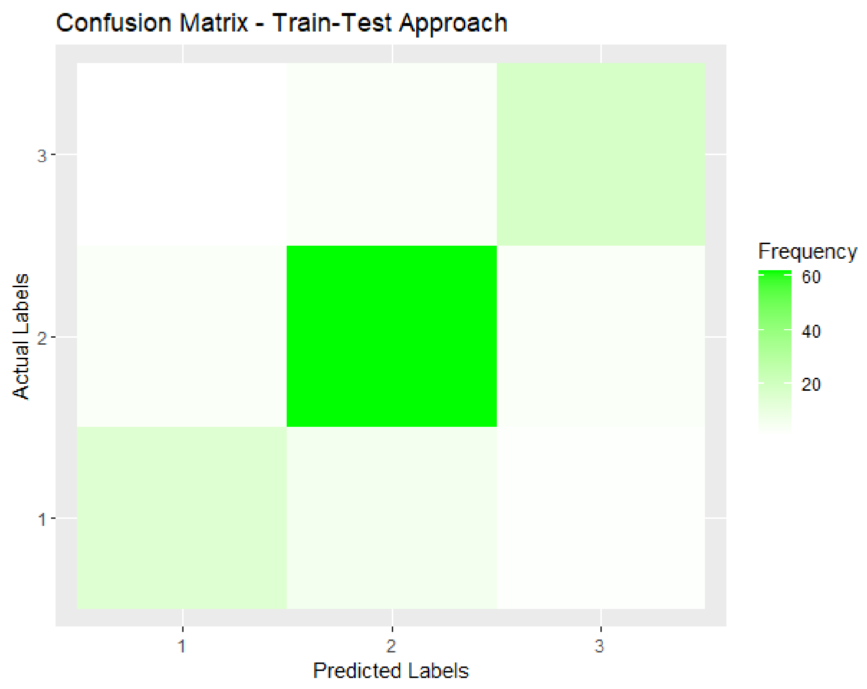
- **So, the recall value which shows the true positive rate of the model for each class is 0.79,0.87,0.78. And the precision value which shows the predicted positive rate of the model for each class is 0.65,0.91,0.82.**

- **After finding the confusion matrix, recall and precision values, they have been visualized through ggplot and tidyr library. Where confusion matrix shows the most correctly predicted values more highlighted. And recall precision values are shown comparing in bar plots.**

CODE:
For confusion Matrix

```
library(ggplot2)
conf_matrixM <- as.data.frame(as.table(conf_matrix_TT))
names(conf_matrixM) <- c("Predicted_Labels", "Actual_Labels", "Frequency")

ggplot(conf_matrixM, aes(x = Predicted_Labels, y = Actual_Labels, fill = Frequency)) +
  geom_tile() + labs(title = "Confusion Matrix - Train-Test Approach",   x = "Predicted Labels",y
= "Actual Labels") +  scale_fill_gradient(low = "white", high = "green")
```

OUTPUT:

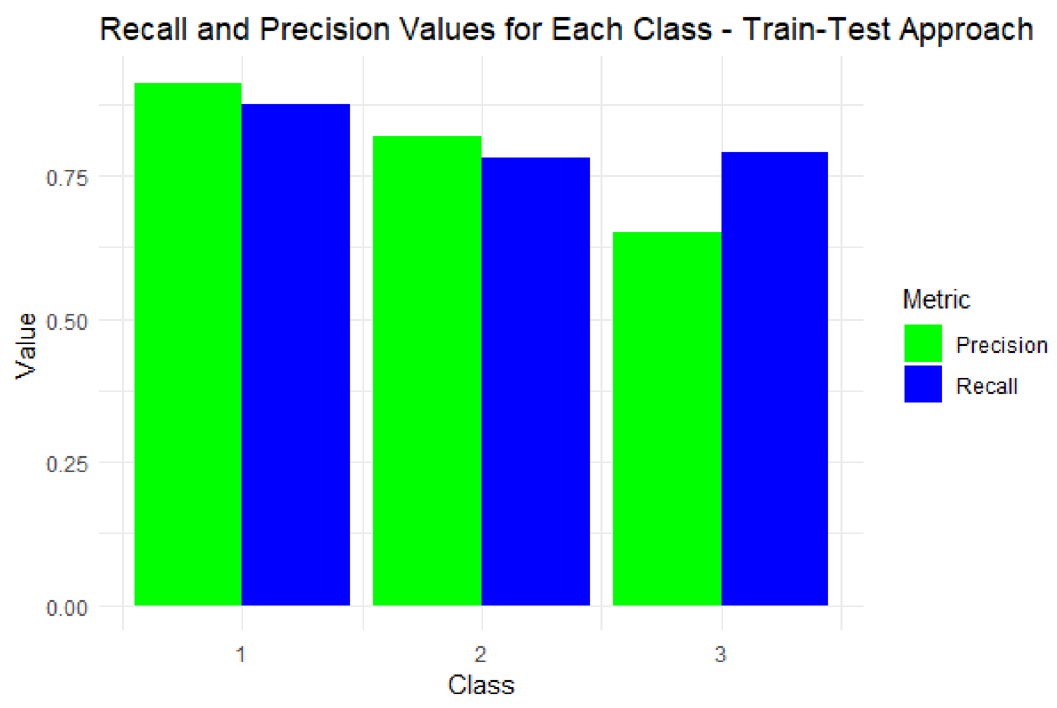


CODE:
For Recall and Precision

```
library(tidyr)
class_metrics_df <- data.frame(Class = unique(test_labelsM),Recall = TT_recall_f,Precision =
TT_precision_f)
class_metrics_df_long <- tidyr::gather(class_metrics_df, Metric, Value, Recall:Precision)
```

```
ggplot(class_metrics_df_long, aes(x = Class, y = Value, fill = Metric)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Recall and Precision Values for Each Class - Train-Test Approach",
       x = "Class", y = "Value") +
  scale_fill_manual(values = c("Recall" = "blue", "Precision" = "green")) +
  theme_minimal()
```

<u>OUTPUT:</u>



- **Secondly, finding the accuracy of kNN by 10-Fold Cross Validation approach. And taking number of folds=10 and k=5 for nearest neighbours. After that the model is built by folding the dataset into 10 fold where 1 fold will be determined as test and others will be determined as train fold.**

10-Fold Cross Validation Approach

<u>CODE:</u>

```
num_foldsff <- 10

accuracy_valuesff <- numeric(num_foldsff)
recall_valuesffzz <- matrix(0, nrow = num_foldsff, ncol = 3)  # 3 classes
precision_valuesffzz <- matrix(0, nrow = num_foldsff, ncol = 3)  # 3 classes

fold_featuresff <- c("Gender", "Age", "Occupation", "Sleep.Duration", "Quality.of.Sleep",
 "Physical.Activity.Level","Stress.Level","BMI.Category","Blood.Pressure","Heart.Rate",
"Daily.Steps")
```

```
for (fold in 1:num_foldsff) {
  set.seed(2)
  fold_indicesff <- sample(nrow(SHL_Dataset_mod), nrow(SHL_Dataset_mod) * 0.1)
  test_foldff <- SHL_Dataset_mod[fold_indicesff, ]
  train_foldff <- SHL_Dataset_mod[-fold_indicesff, ]

  train_fold_featuresff <- train_foldff[fold_featuresff]
  train_fold_labelsff <- train_foldff$Sleep.Disorder
  test_fold_featuresff <- test_foldff[fold_featuresff]
  test_fold_labelsff <- test_foldff$Sleep.Disorder

  k <- 5
 fold_predicted_labelsff<-knn(train_fold_featuresff,test_fold_featuresff,
train_fold_labelsff, k)

  accuracy_valuesff[fold] <- sum(fold_predicted_labelsff == test_fold_labelsff) /
length(test_fold_labelsff)
  print(accuracy_valuesff)
```

CODE FOR CONFUSION MATRIX- the confusion matrix is measured. Recall and precision value is also measured. Where confusion matrix shows the predicted and actual labels within a table. By which it can be determined that how many instances were predicted right in each fold and the mean value of them will be shown or how the model performed through the 10-fold approach, can be determined. And Recall, Precision values will determine the True Positive rate and Positive Predicted value of each class for the folds.

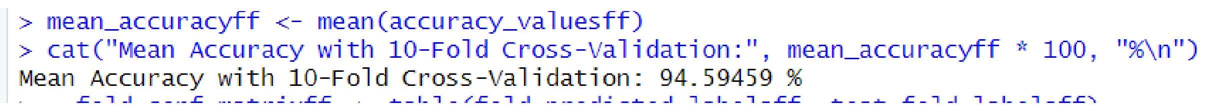
```
  fold_conf_matrixff <- table(fold_predicted_labelsff, test_fold_labelsff)
  cat("Confusion Matrix -for 10 Fold :\n")
  print(fold_conf_matrixff)
  View(fold_conf_matrixff)

  ffrecall <- numeric(nrow(fold_conf_matrixff))
  ffprecision <- numeric(nrow(fold_conf_matrixff))

  for (i in 1:nrow(fold_conf_matrixff)) {
    ffrecall[i] <- fold_conf_matrixff[i, i] / sum(fold_conf_matrixff[i, ])
    ffprecision[i] <- fold_conf_matrixff[i, i] / sum(fold_conf_matrixff[, i])
  }

  recall_valuesffzz[fold, ] <- ffrecall
  precision_valuesffzz[fold, ] <- ffprecision

}
mean_accuracyff <- mean(accuracy_valuesff)
```

mean_recall_valuesffzz <- colMeans(recall_valuesffzz)
mean_precision_valuesffzz <- colMeans(precision_valuesffzz)

cat("Mean Accuracy with 10-Fold Cross-Validation:", mean_accuracyff * 100, "%\n")
print(mean_precision_valuesffzz * 10)
print(mean_recall_valuesffzz * 10)

OUTPUT:

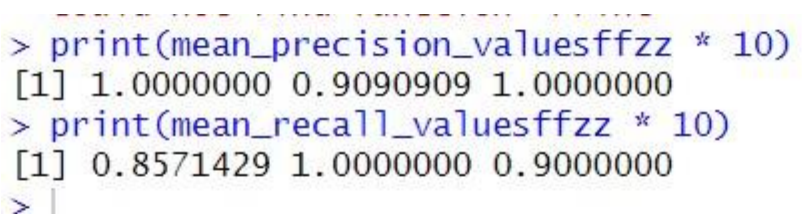
Accuracy for 10-fold cross validation.



```
> mean_accuracyff <- mean(accuracy_valuesff)
> cat("Mean Accuracy with 10-Fold Cross-Validation:", mean_accuracyff * 100, "%\n")
Mean Accuracy with 10-Fold Cross-Validation: 94.59459 %
```

- **So, the accuracy improved from train test approach, and it is now 94.5%**

Recall and Precision value for 10-fold cross validation.



```
> print(mean_precision_valuesffzz * 10)
[1] 1.0000000 0.9090909 1.0000000
> print(mean_recall_valuesffzz * 10)
[1] 0.8571429 1.0000000 0.9000000
> 
```

- **So, this models performance says, recall or true positive rate for each class is 1,0.91,1 and precision or predicted positive rate is 0.86,1,0.90.**

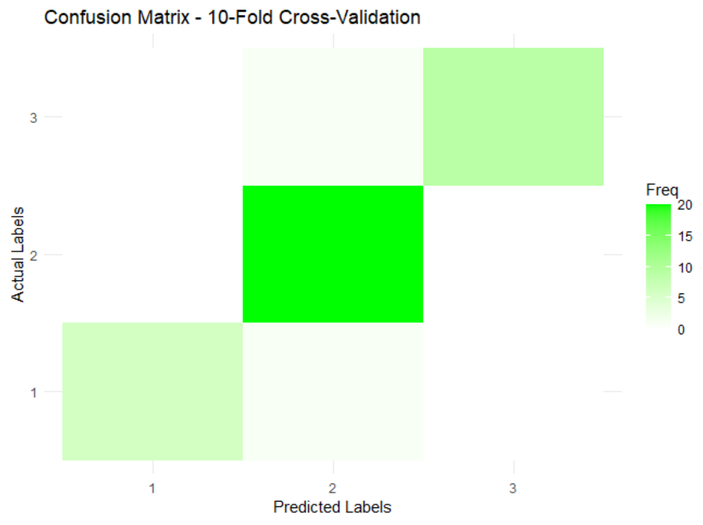
- **After finding the confusion matrix, recall and precision values, they have been visualized through ggplot and tidyr library. Where confusion matrix shows the most correctly predicted values more highlighted. And recall precision values are shown comparing in bar plots.**

CODE:

Confusion matrix

ggplot(as.data.frame(as.table(fold_conf_matrixff)),
aes(x = test_fold_labelsff, y = fold_predicted_labelsff, fill = Freq)) + geom_tile() +
labs(title = "Confusion Matrix - 10-Fold Cross-Validation",
x = "Predicted Labels", y = "Actual Labels") +scale_fill_gradient(low = "white", high = "green")
+theme_minimal()

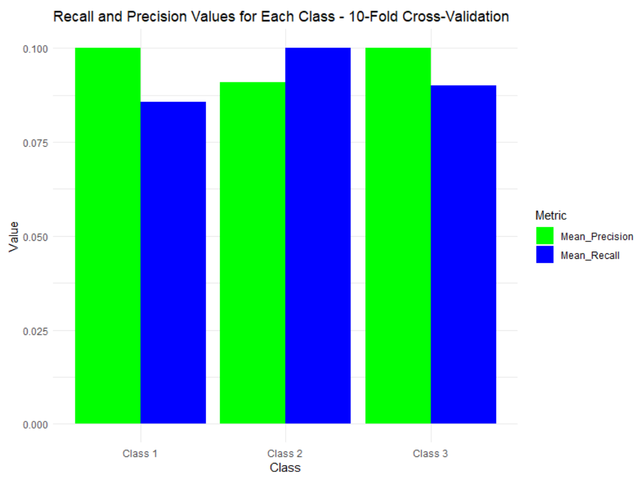
OUTPUT:



Recall and Precision for 10-fold.

```
class_metrics_dfzz <- data.frame(Class = c("Class 1", "Class 2", "Class 3"),
                    Mean_Recall = mean_recall_valuesffzz*10,
                    Mean_Precision = mean_precision_valuesffzz*10)

class_metrics_dfzz_long <- tidyr::gather(class_metrics_dfzz, Metric, Value, Mean_Recall:Mean_Precision)

ggplot(class_metrics_dfzz_long, aes(x = Class, y = Value, fill = Metric)) +
 geom_bar(stat = "identity", position = "dodge") +
 labs(title = "Recall and Precision Values for Each Class - 10-Fold Cross-Validation",
    x = "Class", y = "Value") + scale_fill_manual(values = c("Mean_Recall" = "blue", "Mean_Precision" = "green")) + theme_minimal()
```

OUTPUT:


Recall and Precision Values for Each Class - 10-Fold Cross-Validation
Value
Class 1
Class 2
Class 3
Class
Metric
Mean_Precision
Mean_Recall