**--Getting the coefficient correlation between the store\_nbr and the cluster they belong to:**

> stores <- read\_csv("class/hello/papers/Data analysis with R/stores.csv")

Rows: 54 Columns: 5

-- Column specification -----------------------------------------------------------------------

Delimiter: ","

chr (3): city, state, type

dbl (2): store\_nbr, cluster

i Use `spec()` to retrieve the full column specification for this data.

i Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

> View(stores)

> cor(stores$store\_nbr, stores$cluster)

[1] -0.05928399

**-------Plotting the correlation graph**

> library(ggplot2)

> ggplot(stores) + aes(x = store\_nbr, y = cluster) + geom\_point(colur = "#0c4c8a")+ theme\_minimal()

**---time series modelling**

#STEP 1

> View(stores\_clusters\_1)

> plot.ts(stores\_clusters\_1)

#STEP 2

# MODELING THE TIME SERIES OF VARIOUS CITY CLUSTER

> library(ggplot2) ggplot(stores, aes(city, cluter)) + geom\_line()

Error: unexpected symbol in "library(ggplot2) ggplot"

> library(ggplot2)

> ggplot(stores, aes(city, cluster)) + geom\_line()

#STEP 3

# PREDCICT THE SHOP CLUSTERS

plot(diff(log(stores$cluster)),type='l', main='log returns plot')

#STEP 4 STATIONARISE OUR TIME SERIES

>adf.test(diff(log(as.numeric(stores$cluster))), alternative="stationary", k=0)

#STEP 5

#DETERMINE OUR ACF AND PACF ON OUR DATASET

>acf(diff(log(stores$cluster)))

>pacf(diff(log(stores$cluster)))

#STEP 6

#CALL OUR FORECAST LIBRARY INSIDE ARIMA MODEL

> library(forecast)

> (fit <- arima(diff(log(stores$cluster)), c(3, 0, 1)))

#FITTING OUR ARIMA

> fitARIMA <- auto.arima(diff(log(stores$cluster)), trace=TRUE)

#CHECKNG TO SEE HOW OUR ARIMA MODEL FITTED WITH THE TRAINING DATASET

> plot(as.ts(diff(log(store$clUster))) )

> lines(fitted(fitARIMA), col="red")

#STEP 7

#MAKING A PREDICTION BASED ON OUR ARIMA MODEL

 >futurVal <- forecast(fitARIMA,h=5, level=c(99))

> plot(forecast(futurVal))

**##RNN MODEL**

##STEP 1

##SETTING OUR MODEL PARAMETERS

> View(stores)

> max\_len <- 6

> batch\_size <- 32

> total\_epochs <- 15

> set.seed(123)

> store\_type <- stores$type

## STEP 2

##HERE WE SELECT THE DATASET COLUMN THAT WE WANT TO FORECAST

> table(store\_type)

store\_type

 A  B  C  D  E

 9  8 15 18  4

## STEP 3

 ##WE USE MOVING BLOCK SUB SAMPLING TO ENABLE US CUT OUR VECTOR INTO SMALL BITS FOR EASIER SAMPLING

> start\_indexes <- seq(1, length(store\_type)- (max\_len + 1), by = 3)

> cluster\_matrix <- matrix(nrow = length(start\_indexes), ncol = max\_len + 1)

> for (i in 1:length(start\_indexes)){cluster\_matrix [i,] + max\_len)]}

> for (i in 1:length(start\_indexes))

{cluster\_matrix [i,] <-store\_type[start\_indexes[i]:(start\_indexes[i] + max\_len)]}

#REMOVE WARNING MESSAGES

> dev.off()

null device

## STEP 4

#WE REMOVE N/A values and converting our matrix to NUMERIC

> for (i in 1:length(start\_indexes))

{cluster\_matrix [i,] <-store\_type[start\_indexes[i]:(start\_indexes[i] + max\_len)]}

> cluster\_matrix <- cluster\_matriX \* 1

> cluster\_matrix <- suppressWarnings(as.numeric(cluster\_matrix) \* 1)

> if(anyNA(cluster\_matrix)){

+     cluster\_matrix <- na.omit(cluster\_matrix)

+ }

## STEP 5

#SEPARATE OUR DATA INTO PREVIOUS DAYS AND THEN DEFINE DAYS WE WANT TO PREDICT FOR THE STORE TYPE IN Y VARIABLE

X <- cluster\_matrix[,-ncol(cluster\_matrix)]

y <- cluster\_matrix[,ncol(cluster\_matrix)]

## STEP 6

# THIS INDEXING WILL SEPARATE OUR DATA INTO TRAINING AND TESTING UNITS

training\_index <- createDataPartition(y, p = .9,

                                  list = FALSE,

                                  times = 1)

## STEP 7

# THEN TRAIN THE DATA

X\_train <- array(X[training\_index,], dim = c(length(training\_index), max\_len, 1))

y\_train <- y[training\_index]

## STEP 7

# THEN TEST THE DATA

X\_test <- array(X[-training\_index,], dim = c(length(y) - length(training\_index), max\_len, 1))

y\_test <- y[-training\_index]

## STEP 8

# DEFINE A NEW MODEL FOR THE STORES DATASET

stores\_cluster\_model <- keras\_model\_sequential()

# DEFINE NEW DIMENSIONS FOR INPUT DATA

dim(X\_train)

## STEP 9

# THEN DEFINE INPUT LAYER OF THE MODEL

stores\_cluster\_model %>%

    layer\_dense(input\_shape = dim(X\_train)[2:3], units = max\_len)

stores\_cluster\_model %>%

    layer\_simple\_rnn(units = 6)

stores\_cluster\_model %>%

    layer\_dense(units = 1, activation = 'sigmoid')

# TO GET A SUMMARY OF THE MODEL STRUCTURE USE summary command

summary(stores\_cluster\_model)

stores\_cluster\_model %>% compile(loss = 'binary\_crossentropy',

                  optimizer = 'RMSprop',

                  metrics = c('accuracy'))

## STEP 10

# TRAINING THE MODEL

stores\_cluster\_training\_model <- stores\_cluster\_model %>% fit(

    x = X\_train,

    y = y\_train,

    batch\_size = batch\_size,

    epochs = total\_epochs,

    validation\_split = 0.1)

## STEP 11

# PREVIEW THE MODEL

stores\_cluster\_training\_model

# PLOT THE RESULTING MODEL AS TRAINED

plot(stores\_cluster\_training\_model)

## STEP 12

# CONDUCT AN RNN FORCASTING ON THE STORE TYPE

class\_prediction <- stores\_cluster\_model %>% predict\_class\_prediction(X\_test, batch\_size = batch\_size)