Data_Incubator_Challenge

Problem statement:

The problem statement which I chose to solve was to predict the price of NASDAQ Index. As we know, predicting the stock market prices is the major challenge for every investment banking company. Solving this problem will give us an insight about the influence of past prices to predict the future data. I have created a model by applying Artificial Neural Metworks (ANN) on the moving Average of the data to solve this problem

ANN

In machine learning, artificial neural networks (ANNs) are a family of statistical learning models inspired by biological neural networks (the central nervous systems of animals, in particular the brain) and are used to estimate or approximate functions that can depend on a large number of inputs and are generally unknown. The connections have numeric weights that can be tuned based on experience, making neural nets adaptive to inputs and capable of learning.

Use of artificial neural network

With five outputs corresponding to closing price of five consecutive days Inputs nodes to be varied for different trial No. of intermediate layers and no. of nodes within those layers can be decided based on experimentation(trial/error) We used 70% of available data for training and 30% for validation Training data and Validation data to be distributed across different time windows to reduce any temporal/seasonality bias

No other macro-economic indicators like Interest rate change, gold prices, fluctuations in commodity prices and currency rate exchange to be used as input

```
FALSE Loading required package: lubridate
```

```
FALSE
FALSE Attaching package: 'lubridate'
```

```
FALSE The following object is masked from 'package:base':
FALSE
FALSE     date
```

```
FALSE
FALSE Attaching package: 'plyr'
```

```
FALSE The following object is masked from 'package:lubridate':
FALSE
FALSE     here
```

Downloading and reading the data.

```
data <- read.csv("data_download.csv")
```

Data can also be downloaded from the web using now <- Sys.Date()

d=month(now)
e=day(now)
f=year(now)

p2 <- 1525113000 + as.integer(as.Date(paste(e,d,f,sep = “-”),“%d-%m-%Y”) - as.Date(“01-05-2018”,“%d-%m-%Y”))

p1 <- p2 - 365*24*60*60

url = paste(“https://finance.yahoo.com/quote/%5EIXIC/history?”,“period1=”, p1,“&period2=”,p2,“&interval=1d&filter=history&frequency=1d”)
print(paste(“Extracting data from the URL”,url))

down <- paste(“https://query1.finance.yahoo.com/v7/finance/download/%5EIXIC?”,“period1=”, p1,“&period2=”,p2,“&interval=1d&events=history&crumb=8FrhhbI2eD0”,sep=“”)
download.file(down, destfile = “./data_download.csv”)

Preprocessing

```
data$Date <- as.Date(data$Date,format = "%Y-%m-%d")

#For converting it into a timeseries data
ts <- ts(data)

#Flipping the data in ascending order
data1 = data[order(nrow(data):1),]
```

Adding columns which contains past data (1 to 5 days)

```
data1 <- data1[!is.na(data1$Close),]
length=nrow(data1)
data1$Price = c(data1$Close)
data1$Price_1 = c(0,head(data1$Close,-1))
data1$Price_2 = c(0,0,head(data1$Close,-2))
data1$Price_3 = c(0,0,0,head(data1$Close,-3))
data1$Price_4 = c(0,0,0,0,head(data1$Close,-4))
data1$Price_5 = c(0,0,0,0,0,head(data1$Close,-5))
```

Calculate simple moving average for 5, 10 and 50 days

```
data1$SMA_5=SMA(data1$Close,n=5)
data1$SMA_10=SMA(data1$Close,n=10)
data1$SMA_50=SMA(data1$Close,n=50)
```

Assignming 0 to NAs

```
data1$SMA_5[is.na(data1$SMA_5)]<-0
data1$SMA_10[is.na(data1$SMA_10)]<-0
data1$SMA_50[is.na(data1$SMA_50)]<-0

close=data.frame(data1$Close)
data1$tr=c(close[2:length,],0)
data1$tr2 = c(close[3:length,],0,0)
```

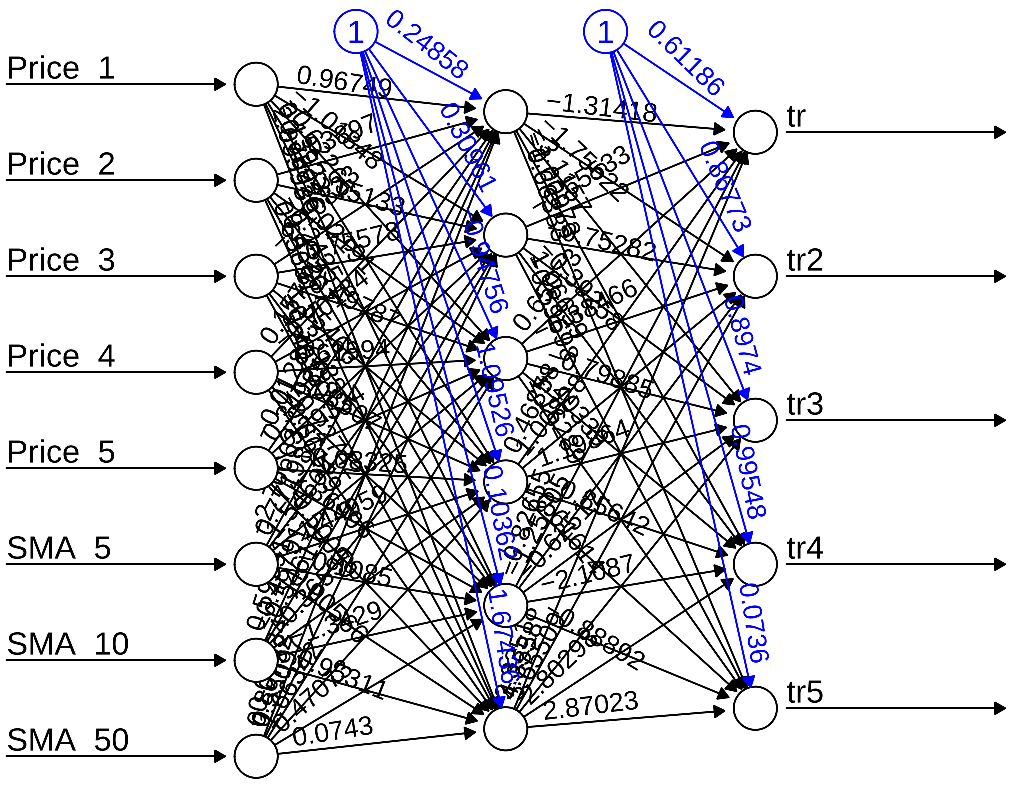
```r
data1$tr3 = c(close[4:length,],0,0,0)
data1$tr4 = c(close[5:length,],0,0,0,0)
data1$tr5 = c(close[6:length,],0,0,0,0,0)

write.csv(data1,"datafinal_model.csv")
datanew=read.table("datafinal_model.csv",header=TRUE,sep=",")
datanew=datanew[-c(1:50,length, length-1, length-2, length-3, length-4),];
data2=datanew[,9:22]
```

Normalising the data

```r
norm.fun = function(x){
    (x - min(x))/(max(x) - min(x))
}

#Applying norm function from coulmn 4 to coumn 21 in both train and validation data
data3=apply(data2[,(2:ncol(data2))],2,norm.fun)

#70% of observations is training data. Computing its length
train.length=round(length*0.70)

train.data=data3[1:train.length,] #Storing Traindata
val.data=data3[-c(1:train.length),-c(9,10,11,12,13)] #Remaining 30% is Validation data

#Variables used : Past 5 days prices
```

Modelling

```r
infy831<- neuralnet(tr + tr2 + tr3 + tr4 + tr5 ~ Price_1 + Price_2 + Price_3 +
                      Price_4 + Price_5 + SMA_5 + SMA_10 + SMA_50,
                    train.data,
                    hidden = c(6), threshold = 0.1,
                    stepmax = 10000, rep = 10,algorithm = "rprop+",
                    err.fct = "sse",
                    linear.output = TRUE, exclude = NULL,
                    constant.weights = NULL, likelihood = FALSE)
#Plot the model
plot(infy831, rep = "best");
```

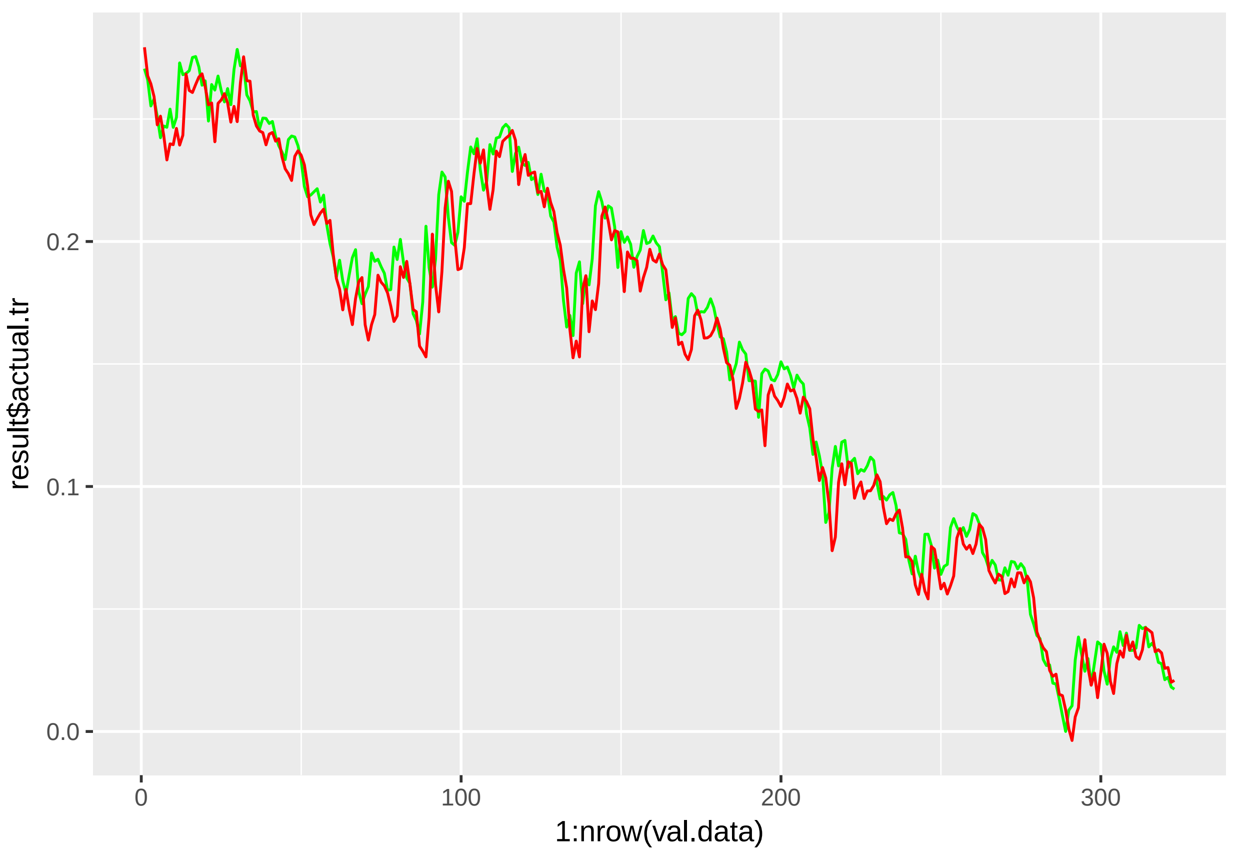


Error: 1.123403 Steps: 2015

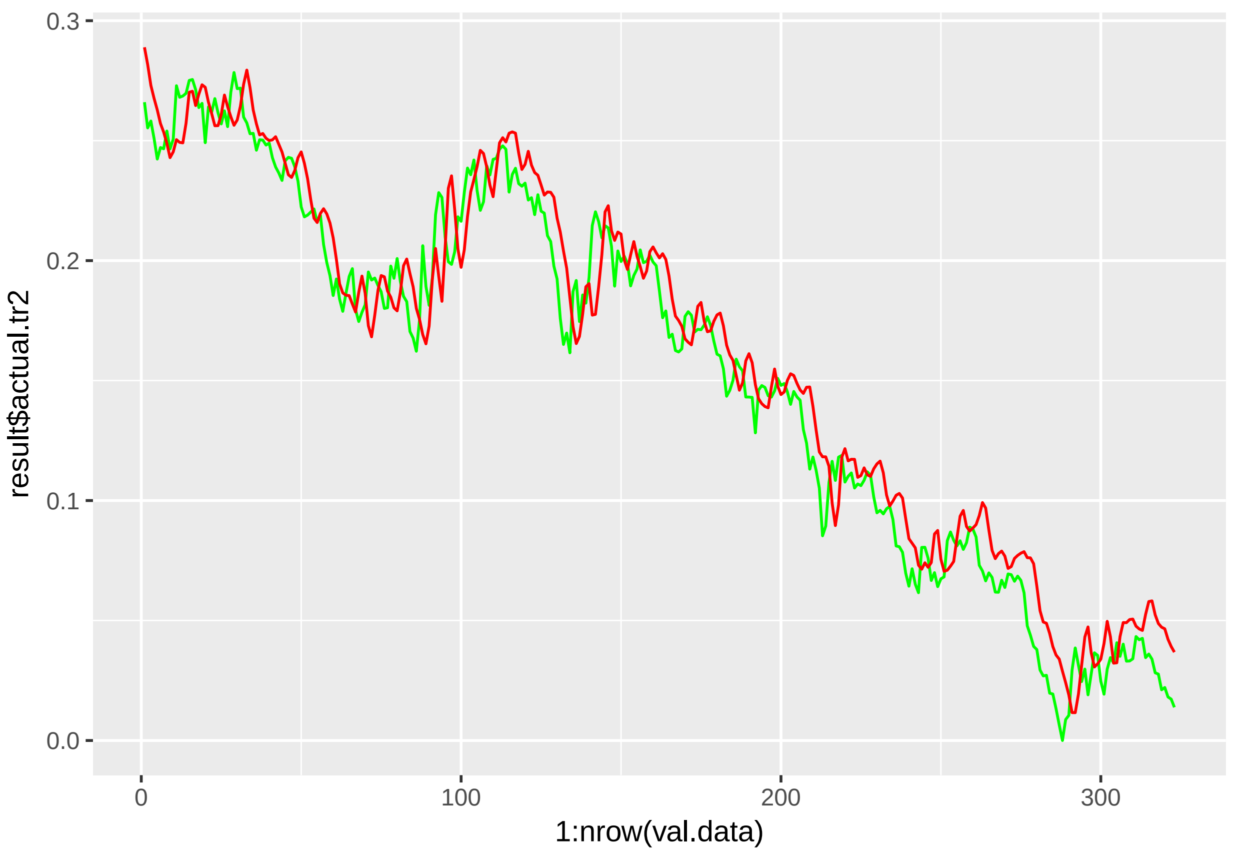
```
pred= compute(infy831, val.data,rep=1)
result = data.frame(actual = data3[-c(1:train.length),c(9,10,11,12,13)],
                    prediction = pred$net.result)
```

Plotting the actual and predicted values of the validation data

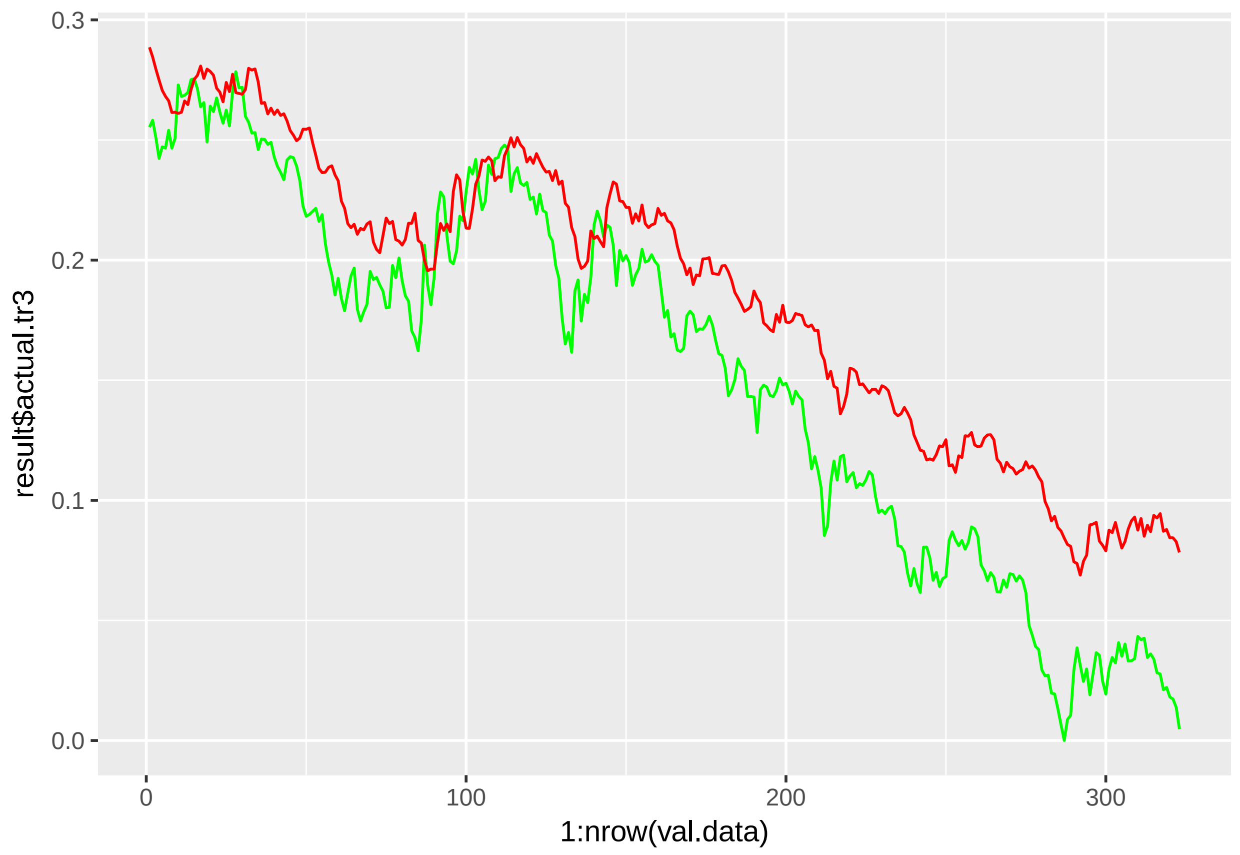
```
ggplot(result, aes(1:nrow(val.data))) +  # basic graphical object
    geom_line(aes(y=result$actual.tr), colour="green") +  # first layer
    geom_line(aes(y=result$prediction.1), colour="red")  # second layer
```



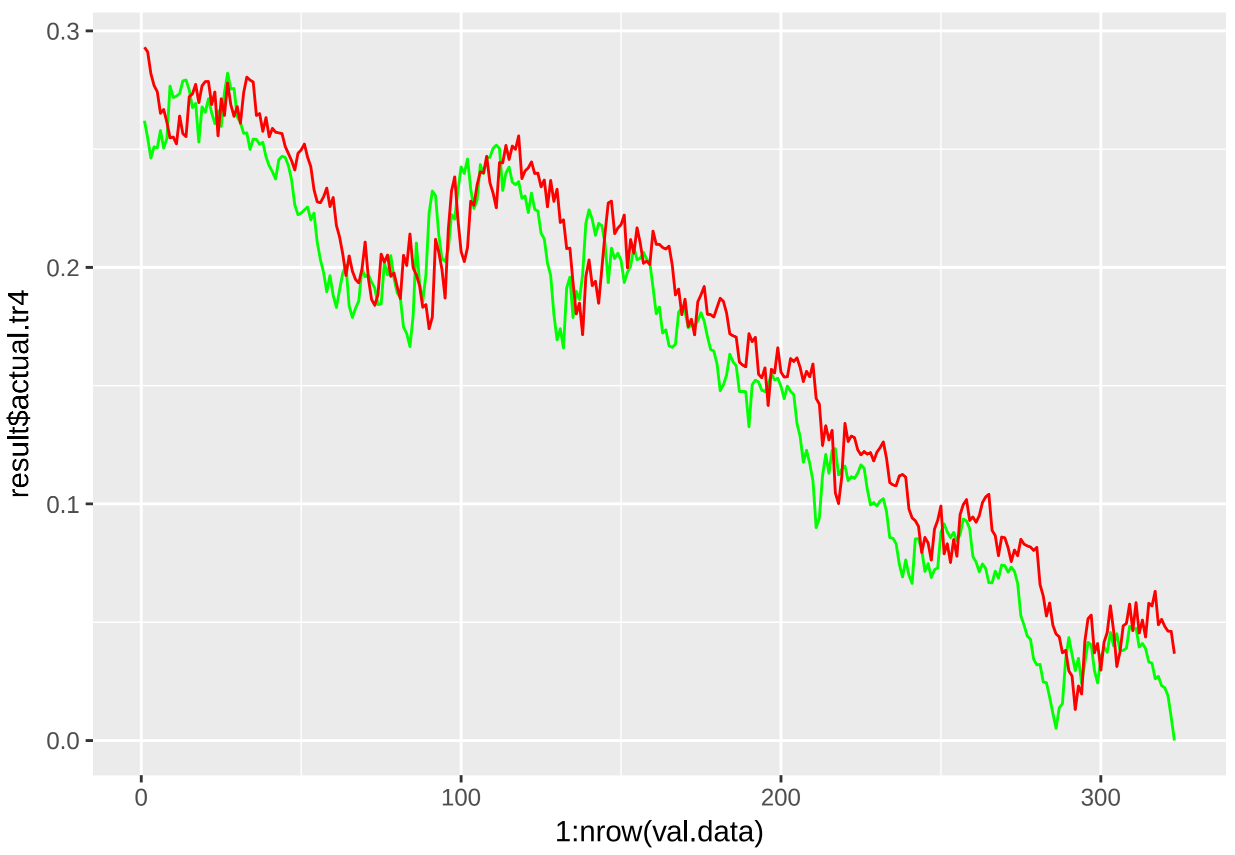
```
ggplot(result, aes(1:nrow(val.data))) +  # basic graphical object
    geom_line(aes(y=result$actual.tr2), colour="green") +  # first layer
    geom_line(aes(y=result$prediction.2), colour="red")  # second layer
```



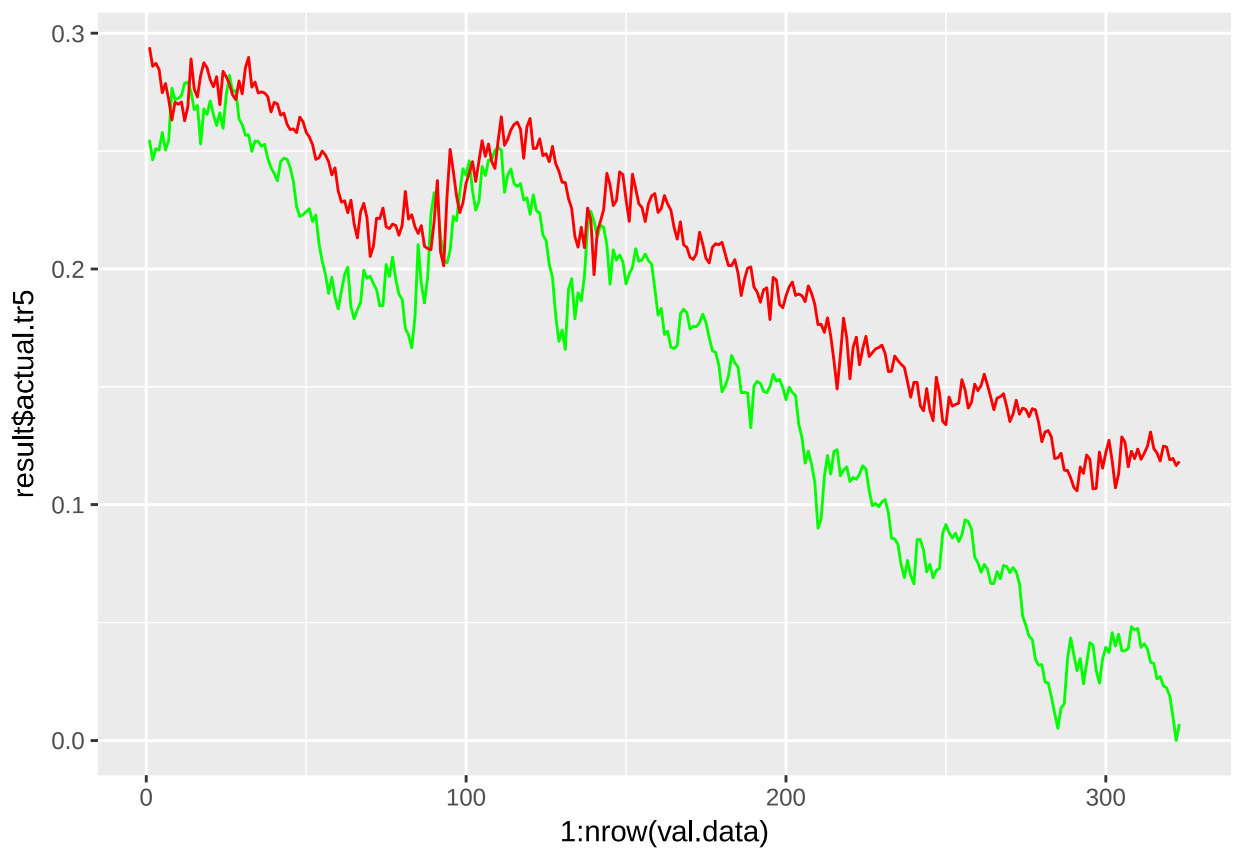
```
ggplot(result, aes(1:nrow(val.data))) +  # basic graphical object
    geom_line(aes(y=result$actual.tr3), colour="green") +  # first layer
    geom_line(aes(y=result$prediction.3), colour="red")  # second layer
```



```
ggplot(result, aes(1:nrow(val.data))) +  # basic graphical object
    geom_line(aes(y=result$actual.tr4), colour="green") +  # first layer
    geom_line(aes(y=result$prediction.4), colour="red")  # second layer
```



```
ggplot(result, aes(1:nrow(val.data))) +  # basic graphical object
    geom_line(aes(y=result$actual.tr5), colour="green") +  # first layer
    geom_line(aes(y=result$prediction.5), colour="red")  # second layer
```



Error

```
RMSE = c(((mean((result$actual.tr-result$prediction.1)^2))^0.5)
        ,((mean((result$actual.tr2-result$prediction.2)^2))^0.5)
        ,((mean((result$actual.tr3-result$prediction.3)^2))^0.5)
        ,((mean((result$actual.tr4-result$prediction.4)^2))^0.5)
        ,((mean((result$actual.tr5-result$prediction.5)^2))^0.5)
        )
RMSE
```

```
## [1] 0.01283041052 0.01611064279 0.03881724283 0.02140688906 0.05397484605
```

Observartion

Model was able to predict for around 200 days+ based on past three years of training with some error
But divergence increased for fifth year.

Scope for improvement:

1) Duration of training data could be modified to obtain maximum accuracy
2) Political, other economics factors and market sentiments could be included to improve the accuracy
3) Time series algorithms can be combined with ML algorithms to capture the seasonality and trend
4) Try to correlate prices of different markets