Experiment - 3

1. **Explain the multi-collinearity and variance inflation factor in dimension reduction. Use VIF to reduce the number of variables which have correlation with other independent variables.**
2. **T-distributed stochastic neighbour embedding technique for Dimensionality Reduction**
3. **Neural auto encoder technique for Dimensionality reduction.**

Aim: To understand the concepts of the multi-collinearity and variance inflation factor,

T-distributed stochastic neighbor embedding technique and Neural auto encoder technique in dimension reduction.

Description:

- There are three diagnostics we can run using R to identify multi-collinearity:
  - Review the correlation matrix for predictor variables that correlate highly.
  - Compute the Variance Inflation Factor (henceforth VIF) and the tolerance statistic.
  - Compute Eigenvalues.
- For a given predictor (p), multicollinearity can assessed by computing a score called the **variance inflation factor (or VIF)**, which measures how much the variance of a regression coefficient is inflated due to multicollinearity in the model.
- Values of VIF that exceed 10 are often regarded as indicating multicollinearity, but in weaker models values above 2.5 may be a cause for concern.
- **t-SNE** is a nonlinear dimensionality reduction technique that is well suited for embedding high dimension data into lower dimensional data (2D or 3D) for data visualization.

Program:

```
#loading the MASS package
```

```
#boston data set

library(MASS)
data(package="MASS")
boston<-Boston
dim(boston)
names(boston)

#LOAD RANDOM FOREST PACKAGE

require(randomForest)

#SET SEED AND CREATE A SAMPLE TRAINING SET OF 300 OBSERVATIONS

set.seed(101)
train=sample(1:nrow(boston),300)

#LETS FIT THE RANDOM FOREST AND SEE HOW IT WORKS

rf.boston=randomForest(medv~.,data=boston,subset=train)
rf.boston

#LETS STORE THE MEAN SQUARE ERROR ON THE OBJECT (OUT OF BAG ERROR)
oob.err=double(13)

# TEST ERROR: MEAN SQUARE ERROR WHICH IS EQUALS TO MEAN((medv-pred)^2)

test.err=double(13)

for(mtry in 1:13)
```

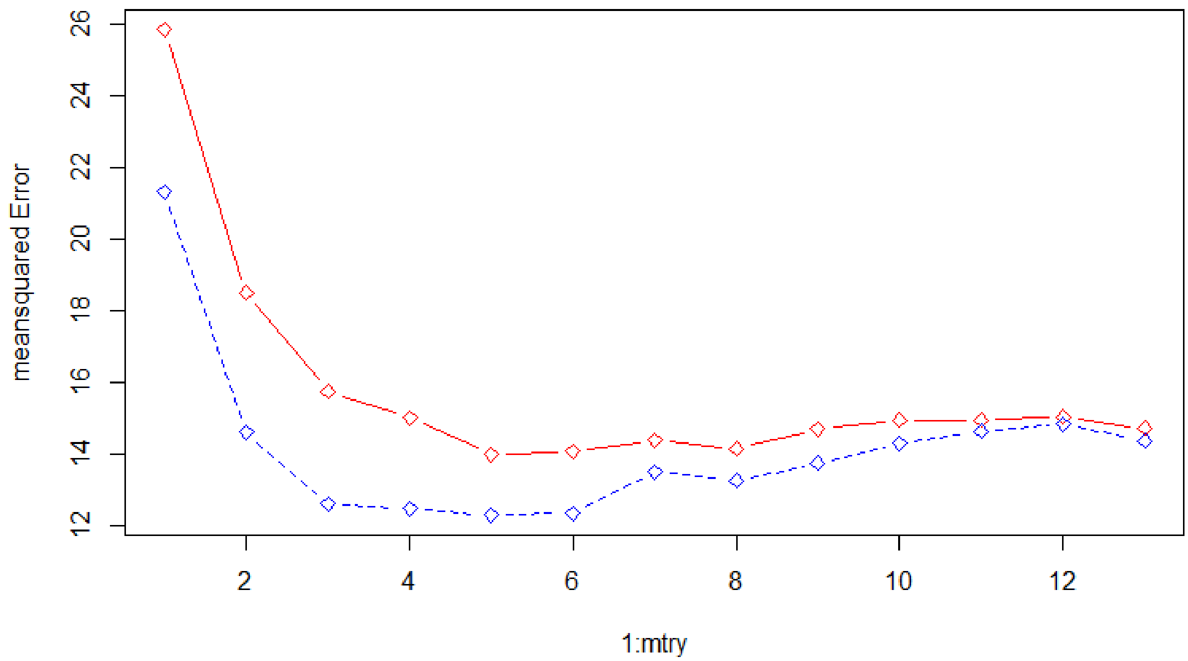
```
{
fit=randomForest(medv~.,data=boston,subset=train,mtry=mtry,ntree=350)
oob.err[mtry]=fit$mse[350]
pred=predict(fit,boston[-train,])
test.err[mtry]=with(boston[-train,], mean((medv-pred)^2))
}
#lets plot the
matplot(1:mtry, cbind(test.err,oob.err),pch=23,col=c("red","blue"),type="b",ylab="meansquared
Error")
legend("topright",legend=c("OOB","Test"),pch=23,col=c("red","blue"))
```

Output:

```
> rf.boston

Call:
 randomForest(formula = medv ~ ., data = boston, subset = train)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 4

          Mean of squared residuals: 12.30718
                    % Var explained: 85.13
> matplot(1:mtry, cbind(test.err,oob.err),pch=23,col=c("red","blue"),type="b",ylab="meansquared Error")
```



meansquared Error
12
14
16
18
20
22
24
26
2
4
6
8
10
12
1:mtry