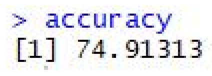
Decision tree using R

Program :

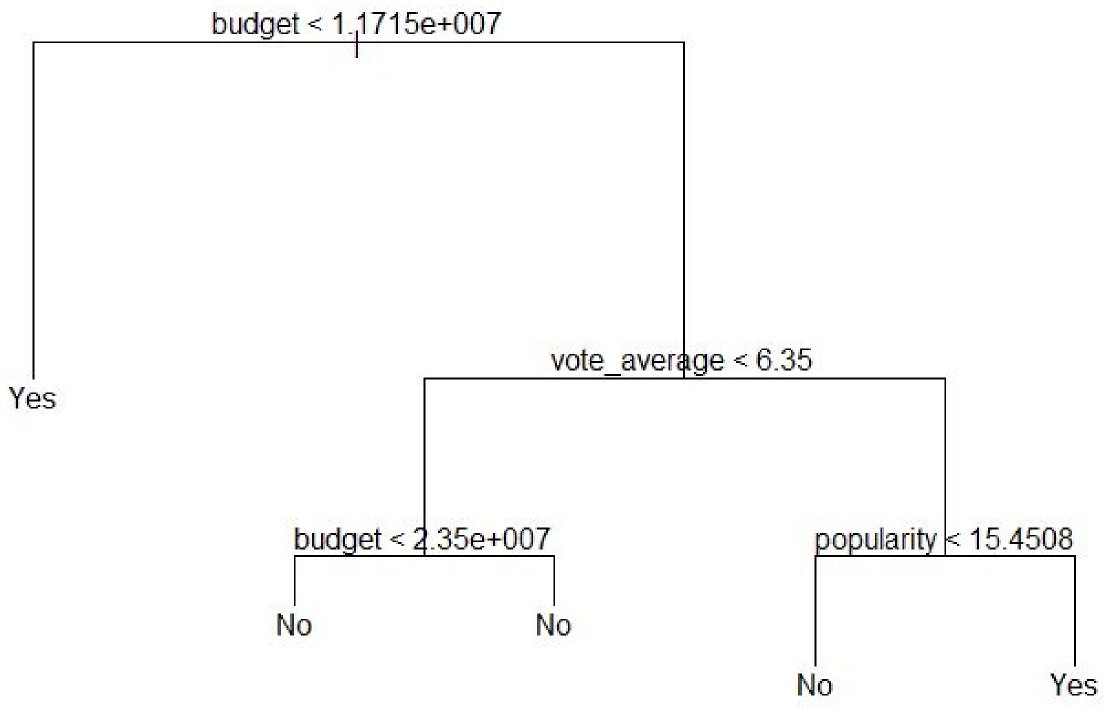
```
setwd("C:/RStudio/test")
data<-read.csv("movies_metadata.csv")
data<-na.omit(data)
library(dplyr)
data[, 3] <- as.numeric(as.character( data[, 3] ))
data[, 11] <- as.numeric(as.character( data[, 11] ))
data1 <- filter(data, budget > 26999, revenue > 30000, runtime > 0,
popularity > 0,vote_count>200)
data1<-select(data1, budget, runtime,popularity,vote_average,revenue)
library(tree)
set.seed(20)
data1$target= ifelse(data1$revenue >(4*data1$budget),"Yes","No" )
success=data1[data1$target=="Yes",]
failure=data1[data1$target=="No",]
success_trows <- sample(1:nrow(success), 0.8*nrow(success))
failure_trows <- sample(1:nrow(failure), 0.8*nrow(success))
tsuccess<-success[success_trows,]
tfailure<-failure[failure_trows,]
tdata<-rbind(tsuccess,tfailure)
test_success<-success[-success_trows,]
test_failure<-failure[-failure_trows,]
testData <- rbind(test_success, test_failure)
require(tree)
tree.data = tree(as.factor(target)~.-revenue,data=tdata)
tree.pred = predict(tree.data,testData,type="class")
match = (testData$target==tree.pred)
testData$pred = match
plot(tree.data)
```

```
text(tree.data,pretty = 0)
accuracy <- (nrow(testData[testData$pred=="TRUE",])/nrow(testData))*100
accuracy
```

Output :



```
> accuracy
[1] 74.91313
```



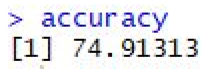
Random forest using R

Program :

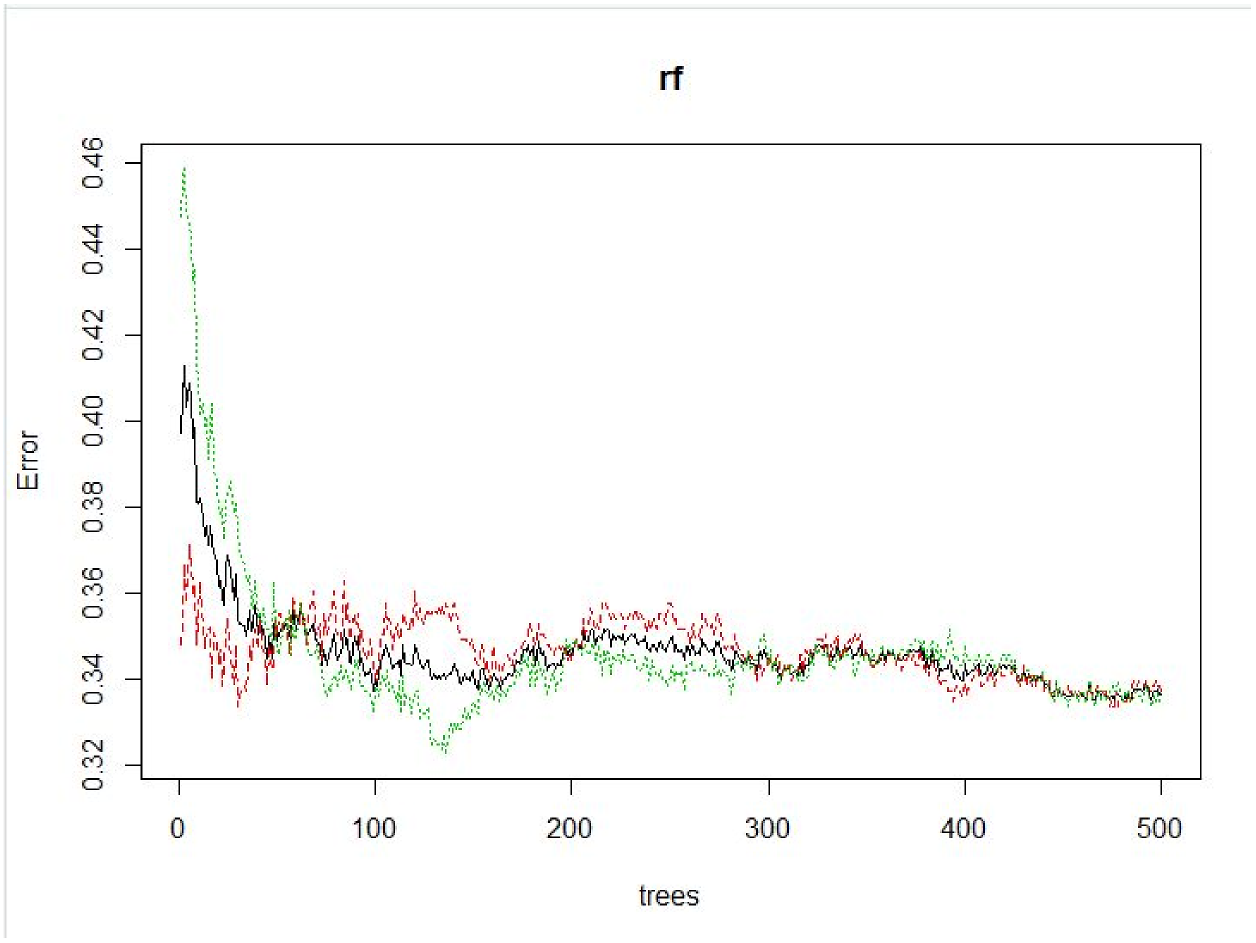
```
setwd("C:/RStudio/test")
data<-read.csv("movies_metadata.csv")
data<-na.omit(data)
library(dplyr)
data[, 3] <- as.numeric(as.character( data[, 3] ))
data[, 11] <- as.numeric(as.character( data[, 11] ))
data1 <- filter(data, budget > 26999, revenue > 30000, runtime > 0,
popularity > 0,vote_count>200)
data1<-select(data1, budget, runtime,popularity,vote_average,revenue)
library(tree)
set.seed(20)
data1$target= ifelse(data1$revenue >(4*data1$budget),"Yes","No" )
success=data1[data1$target=="Yes",]
failure=data1[data1$target=="No",]
success_trows <- sample(1:nrow(success), 0.8*nrow(success))
failure_trows <- sample(1:nrow(failure), 0.8*nrow(success))
tsuccess<-success[success_trows,]
tfailure<-failure[failure_trows,]
tdata<-rbind(tsuccess,tfailure)
test_success<-success[-success_trows,]
test_failure<-failure[-failure_trows,]
testData <- rbind(test_success, test_failure)
library(randomForest)
rf=randomForest(as.factor(target)~.-revenue, data = tdata,importance =T)
rf.pred <-predict(rf,testData)
match1 = (testData$target==rf.pred)
testData$rf.pred = match
plot(rf)
```

```
accuracy <-
(nrow(testData[testData$rf.pred=="TRUE",])/nrow(testData))*100
accuracy
```

Output :



```
> accuracy
[1] 74.91313
```

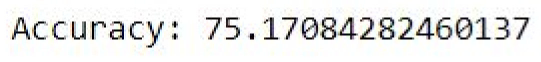


Naive Bayes using Python

Program :

```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('movies_metadata.csv')
df=df[["budget","genres","popularity","revenue","runtime","vote_average",
"vote_count"]]
df=df.dropna()
df['budget'] = df['budget'].astype(float)
df['target']=np.where(df['revenue']>4*df['budget'],'yes','no')
df=df[df.budget>29000]
df=df[df.vote_count>200]
df=df.drop(columns=["vote_count"])
df=df.drop(columns=["genres"])
trainRow = df.iloc[1:2000,:]
testRow = df.iloc[2000:,:]
features = trainRow[["budget","popularity","runtime","vote_average"]]
label = trainRow[["target"]]
testFeatures = testRow[["budget","popularity","runtime","vote_average"]]
testLabel = testRow[["target"]]
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(features,label)
y_pred = nb.predict(testFeatures)
from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(testLabel,y_pred)*100)
```

Output :



```
Accuracy: 75.17084282460137
```

KNN using Python

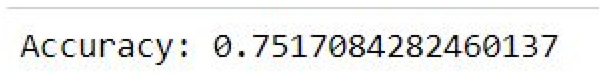
Program :

```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('movies_metadata.csv')
df=df[["budget","genres","popularity","revenue","runtime","vote_average",
"vote_count"]]
df=df.dropna()
df['budget'] = df['budget'].astype(float)
df['target']=np.where(df['revenue']>4*df['budget'],'yes','no')
df=df[df.budget>29000]
df=df[df.vote_count>200]
df=df.drop(columns=["vote_count"])
df=df.drop(columns=["genres"])
trainRow = df.iloc[1:2000,:]
testRow = df.iloc[2000:,:]
features = trainRow[["budget","popularity","runtime","vote_average"]]
label = trainRow[["target"]]
testFeatures = testRow[["budget","popularity","runtime","vote_average"]]
testLabel = testRow[["target"]]
```

```
from sklearn.neighbors import KNeighborsClassifier

model = KNeighborsClassifier(n_neighbors=1000)
model.fit(features,label)
predicted= model.predict(testFeatures)
from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(testLabel, predicted))
```

Output :



```
Accuracy: 0.7517084282460137
```