Return Path - R Model

October 21, 2016

The R Model builds on the test and train files from the Python data pipeline. A Random Forest model is trained with 500 trees and a depth of 3.

Ideally, a cross valdiation should be done, however the initial model seems to work with good accuracy.

Following is the code.

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.2.5
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
## Build a Random Forrest Model
setwd("C:/Users/gnagaraj/Downloads")

#Prepare the Test and Train Datasets
train_df=read.csv("Train_split.csv",colClasses = c("Discretized_Read_Rate"="factor"))
test_df=read.csv("Test_split.csv",colClasses = c("Discretized_Read_Rate"="factor"))

#Drop unwanted columns
train_df <- subset(train_df,select = -c(X,id,read_rate))
test_df <- subset(test_df,select = -c(X,id,read_rate))

#Train a Random Forrest Model with 500 trees and depth of 3
rf <- randomForest(Discretized_Read_Rate ~ .,data = train_df)

#Predict the test data
predictions<-predict(rf,subset(test_df,select = -c(Discretized_Read_Rate)))

#Calculate the Confusion Matrix
cm<-table(as.numeric(unlist(test_df['Discretized_Read_Rate'])),as.numeric(predictions
))
print(cm)
```

```
##
##        1    2    4    5    6    7    8    9   10   11   12   13   14
##   1 2627    0    0    0    0   86   17   10    5    1    1    0    0
##   2    6    0    0    6    0    1    0    4    0    3    2    1    0
##   3    6    0    1    7    0    1    0    1    1    0    2    0    0
##   4    1    0    0   11    1    0    1    0    2    0    2    0    0
##   5    2    0    1  189    1    0    0    0    2    1    3    0    0
##   6    4    0    0   32   19    0    0    1    1    0    0    0    0
##   7  246    0    1    0    0  174   18   19    2    2    0    0    0
##   8  122    0    0    0    0   83   80   15    6    4    5    0    0
##   9   48    0    0    2    0   36   35   35    7    2    1    0    0
##   10  34    1    0    3    0   11   27   13   22    8    3    0    0
##   11  25    0    0    4    0    8    7   14    5   11    5    0    0
##   12  12    0    0    3    0    3    6    5   11   10   12    1    0
##   13  12    0    2    7    0    3    1    6    2    9    4    1    0
##   14   4    0    0    8    0    1    2    4    2    1    2    1    1
```

```
#Print the Accuracy
print(c("Accuracy",sum(diag(cm)/sum(cm))))
```

```
## [1] "Accuracy"          "0.620308542482155"
```