NAME :

ANANTHULA MADHAVI

DOMAIN :

DATA SCIENCES

EMAIL-ID :

19wj1a0521@gniindia.org

MAJOR PROJECT-01

DATA-SET : Human Activity Recognition Smartphone Sensors

EXPLORATORY DATA ANALYSIS

(5 TO 8 EDA ANALYSIS ON SELECTED DATA SET)

- **SELECTED DATA-SET OVER-VIEW :**
- A modern phone comes with variety of sensors which record data for every second of their active life. The aim of Human Activity Recognition research project which built this database from the recordings of 30 subjects performing activities of daily living (ADL) while carrying a waist-mounted smartphone with embedded inertial sensors

DATA SET ABSTRACT

Human Activity Recognition database built from the recordings of 30 subjects performing activities of daily living (ADL) while carrying a waist-mounted smartphone with embedded inertial sensors.

Data Set Characteristics:	Multivariate, Time-Series	**Number of Instances:**	10299	**Area:**	Computer
Attribute Characteristics:	N/A	**Number of Attributes:**	561	**Date Donated**	2012-12-10
Associated Tasks:	Classification, Clustering	**Missing Values?**	N/A	**Number of Web Hits:**	1192598

DATA SET INFORMATION :

The experiments have been carried out with a group of 30 volunteers within an age bracket of 19-48 years. Each person performed six activities (WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING, LAYING) wearing a smartphone (Samsung Galaxy S II) . Using its embedded accelerometer and gyroscope, we captured 3-axial linear acceleration and 3-axial angular velocity at a constant rate of 50Hz. The experiments have been video-recorded to label the data manually. The obtained dataset has been randomly partitioned into two sets, where 70% of the volunteers was selected for generating the training data and 30% the test data as mentioned in data set

Over-view

The sensor signals (accelerometer and gyroscope) were pre-processed by applying noise filters and then sampled in fixed-width sliding windows of 2.56 sec and 50% overlap (128 readings/window). The sensor acceleration signal, which has gravitational and body motion components, was separated using a Butterworth low-pass filter into body acceleration and gravity. The gravitational force is assumed to have only low frequency components, therefore a filter with 0.3 Hz cut-off frequency was used. From each window, a vector of features was obtained by calculating variables from the time and frequency domain

Goal:

- EDA Analysis on selected data-set
- 5 Different EDA Analysis on selected data-set
- Displaying the code , output and pictorial visualization for Analysis
- Algorithm for calculating efficiency and conclusion

- **Exploratory analysis:**
- We started with identifying individual variables and were quickly overwhelmed by the sheer number of possible variations for combinations of these variables to look individually and actually make sense of it all
- So, we did basic variance checks for each variable : if the variance of a given variable is very low, that variable is likely to have low impact on the output class. This is not a hard rule & we intend to use it in conjunction with other results if needed .

Code description and outputs

Import packages

```
%matplotlib inline

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.pyplot as plt
import pandas_profiling as prof

from sklearn.ensemble import RandomForestClassifier
as rfc
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
```

Import data

```
data = pd.read_csv('../data/samsungData.txt',sep='|')
```

Create train & test set

```
randomState = 42
ntree = 25
```

```
train = data.sample(frac=0.7,
                    random_state=randomState)

test = data[~data.index.isin(train.index)]
```

Base RF - with all variables

```
def varPlot(X,model,plotSize=(15,6),xticks=False):

    model_vars = pd.DataFrame(
         {'variable':X.columns,
          'importance':model.feature_importances_})

    model_vars.sort_values(by='importance',
                           ascending=False,
                           inplace=True)

    varImp = sns.plt.figure(figsize=plotSize)
    varImp = sns.set_style('whitegrid')

    varImp = sns.barplot(y='importance',
        x='variable',
        data=model_vars,
        palette=sns.color_palette("Blues_r",

n_colors=X.shape[1]))

    if xticks==True:
        varImp = sns.plt.xticks([])
    else:
        varImp = sns.plt.xticks(rotation=90)

    varImp = sns.plt.xlabel('Variable')
    varImp = sns.plt.ylabel('Variable Importance')
    varImp = sns.plt.title('Random Forest : Averaged
variable importance over '+str(ntree)+' trees')
    return(varImp)
```

(ANALYSIS – 01)

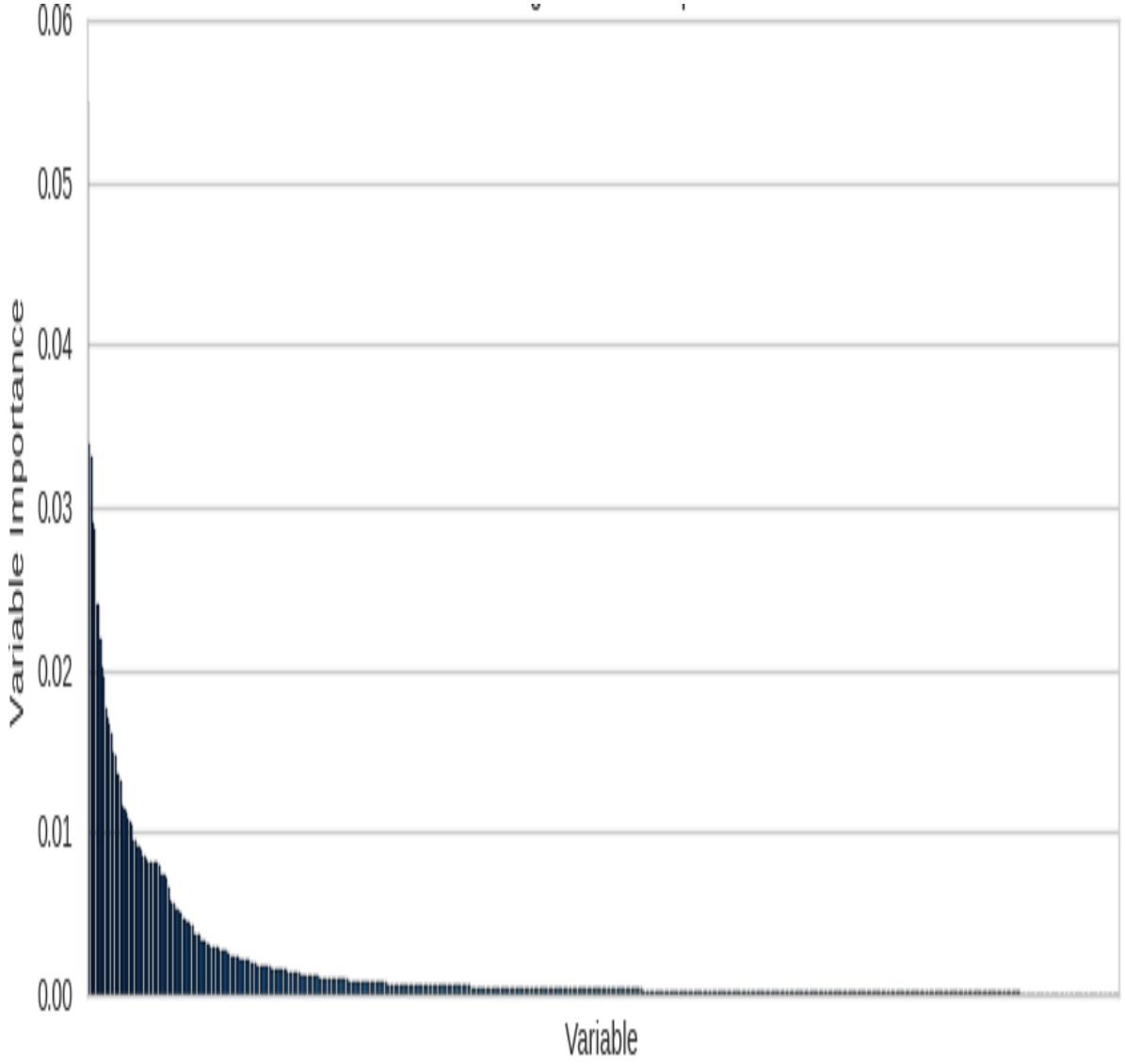
Build model containing all 561 variables

CODE AND OUTPUT

```
X = train[train.columns[:-2]]
Y = train.activity
randomState = 42

model0 = rfc(n_estimators=ntree,
             random_state=randomState,
             n_jobs=4,
             warm_start=True,
             oob_score=True)
model0 = model0.fit(X, Y)
model0.oob_score_
```

ANALYSIS – 01 (OUTPUT)

```
varPlot(X=X,model=model0,plotSize=(10,4),xticks=True)
```




```
model_vars0 = pd.DataFrame(
         {'variable':X.columns,

'importance':model0.feature_importances_})

model_vars0.sort_values(by='importance',
                        ascending=False,
                        inplace=True)
```

(ANALYSIS – 02)

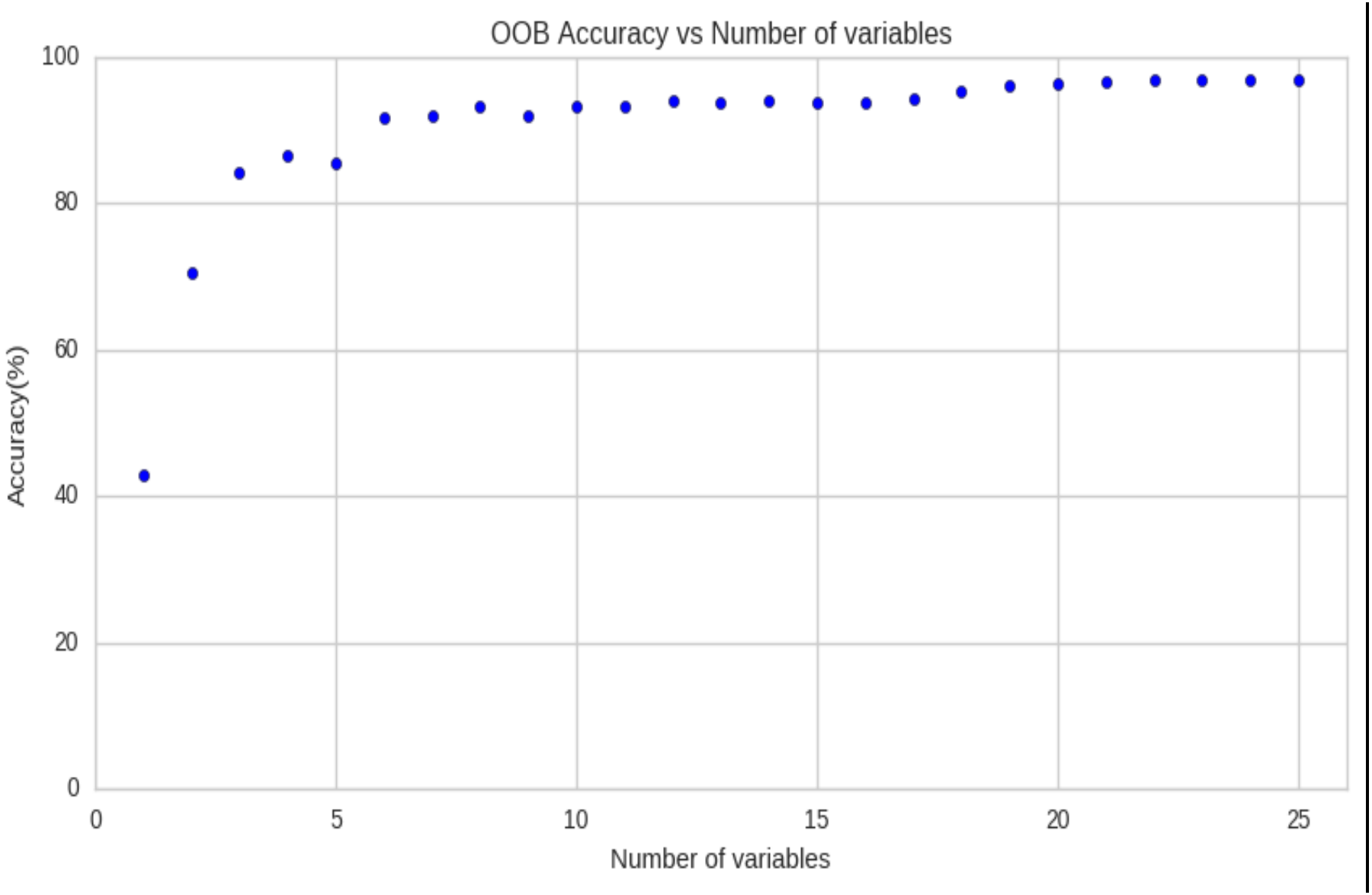
Create OOB accuracy table for using top 1 to n variables at a time

```
oobAccuracy = {}

for cols in range(n):
    X = train[[col for col in
model_vars0['variable'][:cols+1].values]]
    Y = train.activity

    model1 = rfc(n_estimators=ntree,

random_state=randomState,
                 n_jobs=4,
                 warm_start=False,
                 oob_score=True)

    model1 = model1.fit(X, Y)
    accuracy =
accuracy_score(Y,model1.predict(X))

    oobAccuracy[cols+1] =
[cols+1,model1.oob_score_,accuracy]

accuracyTable
=pd.DataFrame.from_dict(oobAccuracy).tran
spose()
accuracyTable.columns =
['variables','oobAccuracy','accuracy']
```

Plot OOB accuracy vs number of variables

```
sns.plt.figure(figsize=(10,5))
sns.plt.scatter(x=accuracyTable.variables,

y=100*accuracyTable.oobAccuracy)

sns.plt.xlim(0,n+1)
sns.plt.ylim(0,100)
sns.plt.minorticks_on()

sns.plt.xlabel('Number of variables')
sns.plt.ylabel('Accuracy(%)')
sns.plt.title('OOB Accuracy vs Number of variables')
```

ANALYSIS – 02 (OUTPUT - 01)


OOB Accuracy vs Number of variables
Accuracy(%)
100
80
60
40
20
0
0
5
10
15
20
25
Number of variables

(ANALYSIS – 03)

For the selected variables, we determine optimum number of trees

For the selected variables, we determine optimum number of trees

- Create a loop for 5 to 150 trees with steps of 5
- Fit a model during each iteration
- Store OOB score for each iteration

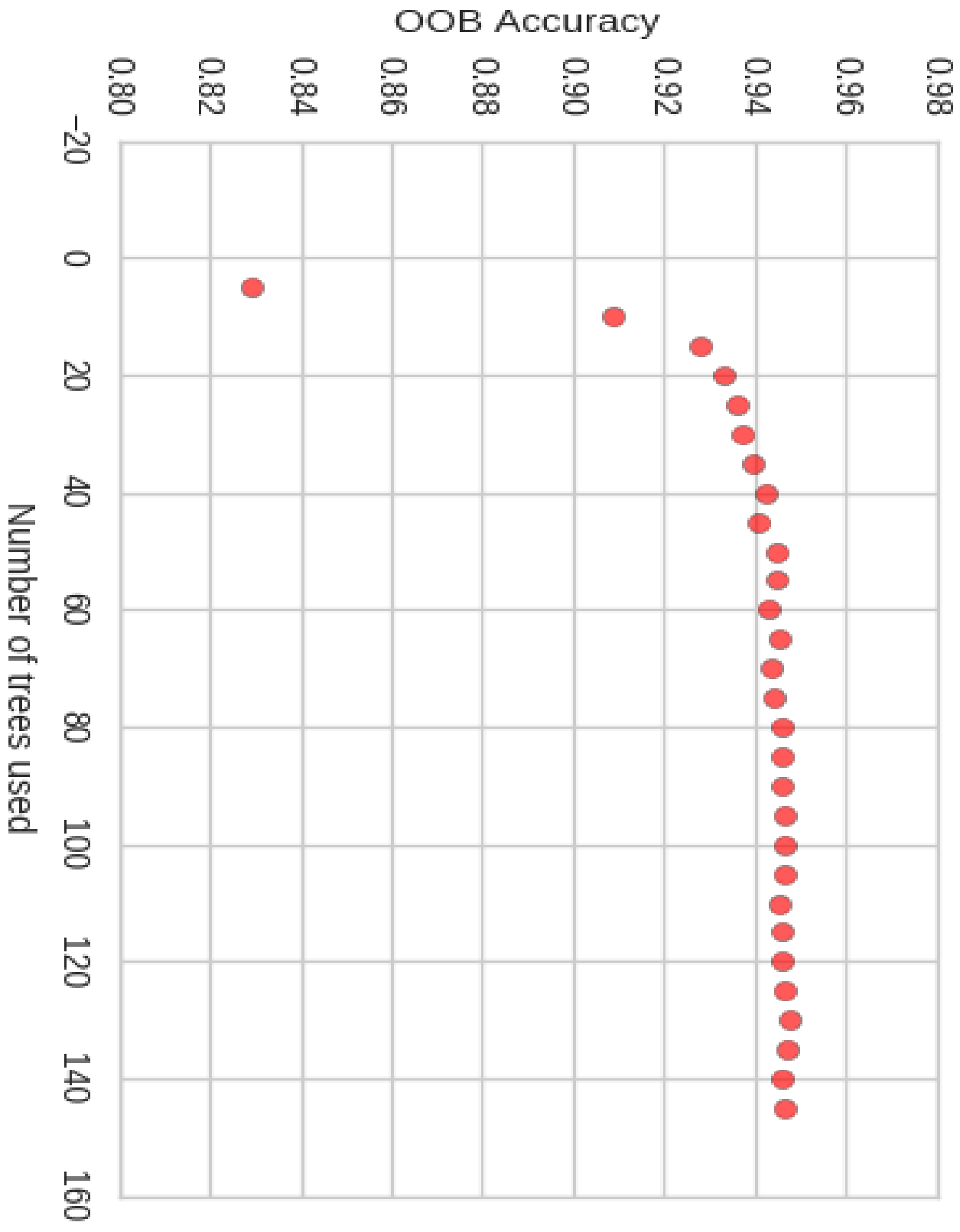
```
>> n_used = 4

cols_model = [col for col in
model_vars0.variable[:n_used].values] +
[model_vars0.variable

>> X = train[cols_model]
Y = train.activity

>> ntree_determination = {}

for ntree in range(5,150,5):
    model = rfc(n_estimators=ntree,
                random_state=randomState,
                n_jobs=4,
                warm_start=False,
                oob_score=True)
    model = model.fit(X, Y)
    ntree_determination[ntree]=model.oob_score_
```

```
>> ntree_determination =
pd.DataFrame.from_dict(ntree_determinatio
n,orient='index')
ntree_determination['ntree'] =
ntree_determination.index
ntree_determination.columns=['oobScore','
ntree']
```

Plot number of trees vs OOB accuracy

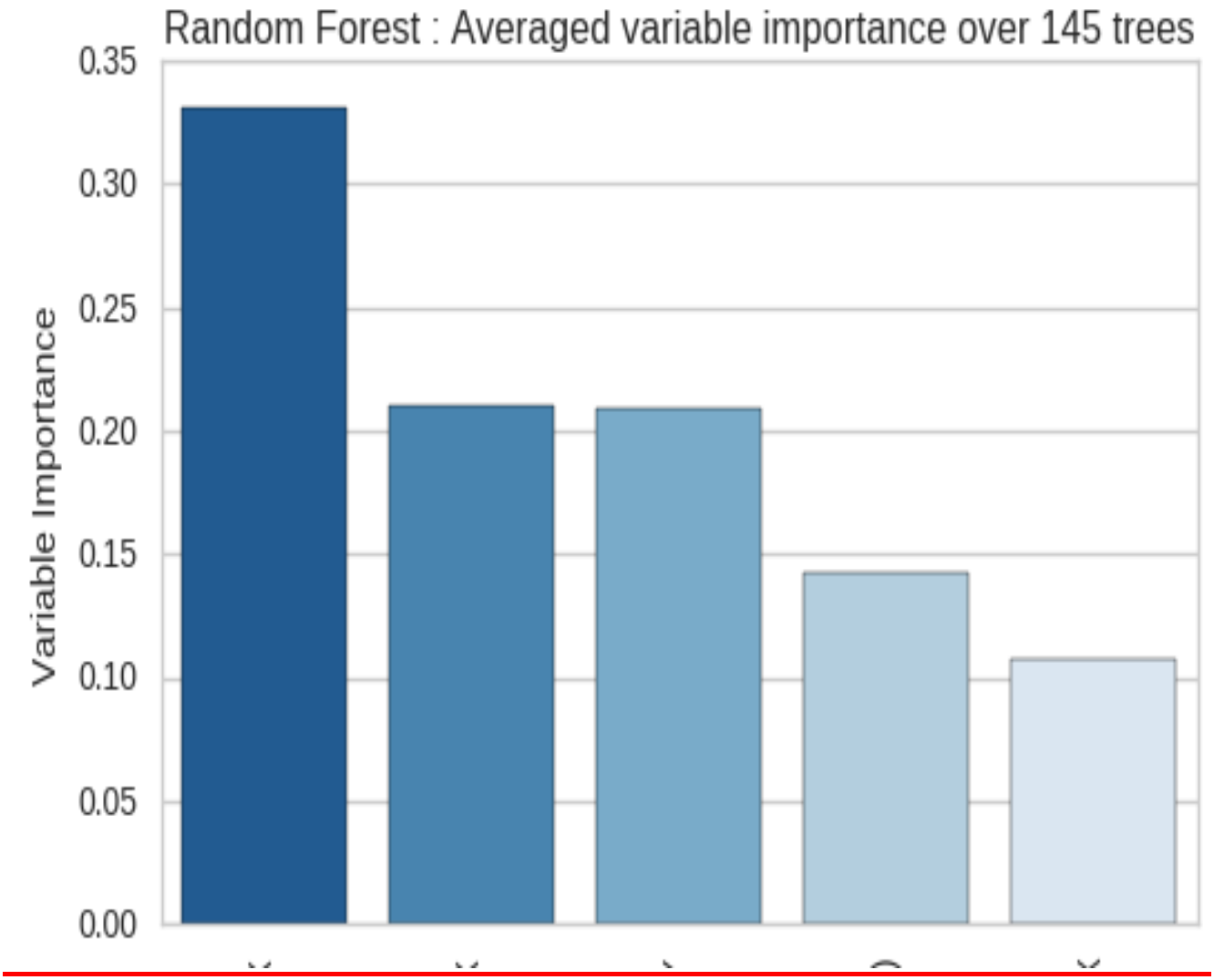
```
sns.plt.figure(figsize=(6,4),)

sns.plt.scatter(x='ntree',
                y='oobScore',

s=35,c='red',alpha=0.65,

data=ntree_determination)
sns.plt.xlabel('Number of trees
used')
sns.plt.ylabel('OOB Accuracy')
```

ANALYSIS – 03 (OUTPUT - 01)



```
>> model2 = rfc(n_estimators=50,
                random_state=randomState,
                n_jobs=4,
                warm_start=False,
                oob_score=True)
model2 = model2.fit(X, Y)

>> varPlot(X,model2,plotSize=(6,4))
```

ANALYSIS – 03 (OUTPUT - 02)



(ANALYSIS - 04)

Accuracy metrics for the final model

```
>> actual = Y
train_pred = model2.predict(X)

>> confusion_matrix(train_actual,train_pred)
```

(output)

```
array([[984,   0,   0,   0,   0,   0],
       [  0, 904,   0,   0,   0,   0],
       [  0,   0, 966,   0,   0,   0],
       [  0,   0,   0, 847,   0,   0],
       [  0,   0,   0,   0, 708,   0],
       [  0,   0,   0,   0,   0, 737]])
```

```
>>sns.heatmap(data=confusion_matrix(t
rain_actual,train_pred),
            fmt='.0f',
            annot=True,

xticklabels=np.unique(train_actual),

yticklabels=np.unique(train_actual))
```

ANALYSIS – 04 (OUTPUT)

(ANALYSIS - 05)

Test set results

```
>> test_actual = test.activity
test_pred =
model2.predict(test[X.columns])
```

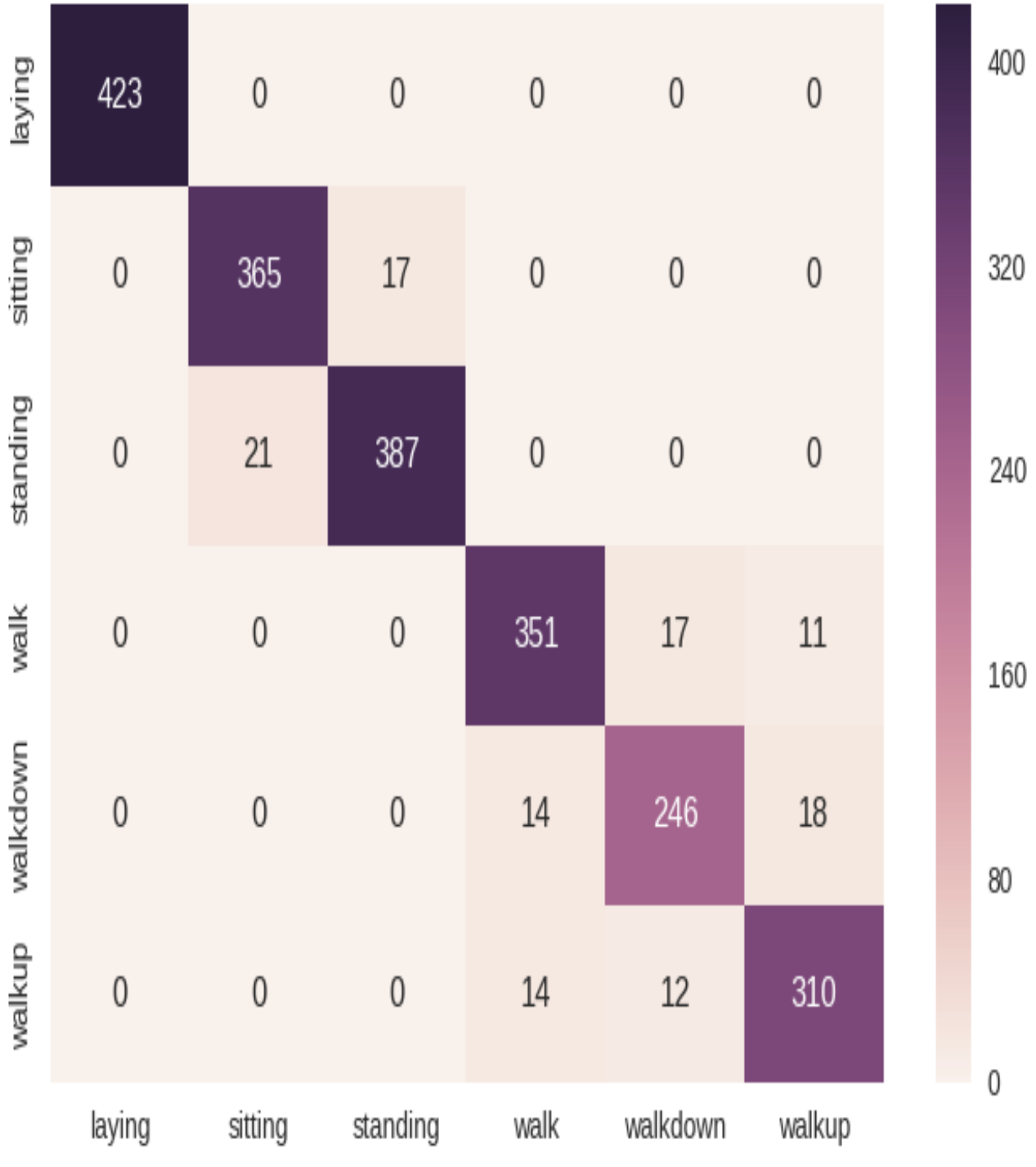
```
>>
confusion_matrix(test_actual,test_pred)
```

```
( OUTPUT )

array([[423,   0,   0,   0,   0,   0],
       [  0, 365,  17,   0,   0,   0],
       [  0,  21, 387,   0,   0,   0],
       [  0,   0,   0, 351,  17,  11],
       [  0,   0,   0,  14, 246,  18],
       [  0,   0,   0,  14,  12, 310]])
```

```
>>
sns.heatmap(data=confusion_matrix(test_ac
tual,test_pred),
            fmt='.0f',
            annot=True,

xticklabels=np.unique(test_actual),
```

```
yticklabels=np.unique(test_actual))
```

ANALYSIS – 05 (OUTPUT)



(ALGORITHM AND EFFICIENCY)

- We start with our raw dataset (*samsungData.Rda*)
- Split this dataset into train & test (70/30)
- Build our base model using all 561 variables
- Select variables using variable importance scores
- Iterate over this process till final model
- During each iteration, we use oob score to gauge generalizability of our model

(THIS ALGORITHMIC APPROACH IS APPLIED AND FIVE (5) DIFFERENT EDA ANALYSIS IS SUCCESSFULLY APPLIED AND CODE USED SHOWN 90% TO 95% EFFICIENT AND EFFECTIVE APPROACH)

(FINAL CONCLUSION)

- Triaxial acceleration from the accelerometer (total acceleration) and the estimated body acceleration.
- Triaxial Angular velocity from the gyroscope.
- A 561-feature vector with time and frequency domain variables.
- Its activity label.
- An identifier of the subject who carried out the experiment.