**NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY**

(AN AUTONOMOUS INSTITUTION, AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM, APPROVED BY AICTE & GOVT.OF KARNATAKA

****

**COURSE-PROJECT REPORT**

**on**

**“HUMAN BEHAVIOUR ANALYSIS”**

*Submitted in partial fulfilment of the requirement for the award of Degree of*

*Bachelor of Engineering in*

*Computer Science and Engineering*

Submitted By

**Anuja Anu 1NT15CS030**

**Anurag Bhattacharjee 1NT15CS031**

**Dipsy Deep 1NT15CS055 Junaid M P 1NT15CS068**

Under the Guidance of

**Dr. Saroja Devi H**

Assoc Professor,

Department of Computer Science and Engineering

**(Accredited by NBA Tier-1)**

2017-18

**NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY**

(AN AUTONOMOUS INSTITUTION)

(AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM, APPROVED BY AICTE & GOVT.OF KARNATAKA)



**CERTIFICATE**

**This is to certify that the Mini Project Report entitled**

**“HUMAN BEHAVIOUR ANALYSIS”**

Is an authentic work carried out by

**Anuja Anu 1NT15CS030**

**Anurag Bhattacharjee 1NT15CS031**

**Dipsy Deep 1NT15CS055 Junaid M P 1NT15CS068**

In partial fulfilment of the requirements for the completion of 6th semester mini project work during the academic year 2017-2018.

1. Name& Signature of HOD 2.Name & Signature of the Guide

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dr. M. N. Thippeswamy Dr.Saroja Devi H

**ACKNOWLEDGEMENT**

The Satisfaction that accompanies the successful completion of any task would be Incomplete without the mention of the people who made it possible, whose Constant guidance and encouragement crowned out efforts with success.

We sincerely express our gratitude to our principal, Dr. **H C Nagaraj** for Constantly backing us during the course of our project.

We are very grateful to Dr. **Thippeswamy M N.** Head of the department Computer Science and engineering, for his strong support, without which the Project would not have been feasible.

We would like to express our heartfelt thanks to our guide **Dr.SarojaDevi H** Professor, Department of computer science and engineering for her valuable adviceAnd constant support throughout the entire course project. It would have beenImpossible for us to proceed further at every stage in the project, without her ableGuidance and motivation.

**ABSTRACT**

Understanding and predicting human behaviour has been of particular interest to researchers for many years. Moreover, the assumption that knowledge of attitudes will help in the task of predicting human behaviour has formed the basis for much consumer and social research. Attitudes are assumed to play an important role in human behaviour theory as the crucial link between what people think and what they do. But a new MIT study suggests an algorithm can predict someone’s behaviour faster and more reliably than humans can. It’s fairly common for machines to analyze data, but humans are typically required to choose which data points are relevant for analysis. In three competitions with human teams, a machine made more accurate predictions than 615 of 906 human teams. And while humans worked on their predictive algorithms for months, the machine took two to 12 hours to produce each of its competition entries. Mobile data is an extraordinary useful source of information about human mobility. Nowadays, no other source of data can provide such level of capillarity and reach within human population. That is the reason why it is the preferred data for analysing any human behaviour that includes human mobility.

**INTRODUCTION**

Machine learning is a branch in computer science that studies the design of algorithms that can learn. Typical machine learning tasks are concept learning, function learning or “predictive modelling”, clustering and finding predictive patterns. These tasks are learned through available data that were observed through experiences or instructions, for example. Machine learning hopes that including the experience into its tasks will eventually improve the learning. The ultimate goal is to improve the learning in such a way that it becomes automatic, so that humans like ourselves don’t need to interfere any more.

The goal of our project is to predict the manner in which they did the exercise. There is “classe” variable in the training set. You may use any of the other variables to predict with. Human personality recognition is becoming most important in the modern world. It helps human simplify their jobs and solve more complex tasks.

The below are the classes which are being used in the project in order to analyze the human behaviour:

|  |  |
| --- | --- |
| CLASS | HUMAN BEHAVIOUR |
| A | WALKING |
| B | SLEEPING |
| C | RUNNING |
| D | WALKING UPSTAIRS |
| E | WALKING DOWNSTAIRS |

**ALGORITHMS USED**

**1. RANDOM FOREST:**

**Improved Bagging algorithm**

**Random forests** or **random decision forests** are an [ensemble learning](https://en.wikipedia.org/wiki/Ensemble_learning) method for [classification](https://en.wikipedia.org/wiki/Statistical_classification), [regression](https://en.wikipedia.org/wiki/Regression_analysis) and other tasks, that operate by constructing a multitude of [decision trees](https://en.wikipedia.org/wiki/Decision_tree_learning) at training time and outputting the class that is the [mode](https://en.wikipedia.org/wiki/Mode_(statistics)) of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of [over fitting](https://en.wikipedia.org/wiki/Overfitting) to their [training set](https://en.wikipedia.org/wiki/Test_set)

**Pros:**

* Accuracy

**Cons:**

* Slow speed
* Interpretability
* Over fitting -> important to do cross-validation.

**2. BAGGING ALGORITHM**

**Bootstrap aggregating**, also called **bagging**, is a [machine learning ensemble](https://en.wikipedia.org/wiki/Ensemble_learning) [meta-algorithm](https://en.wikipedia.org/wiki/Meta-algorithm) designed to improve the stability and accuracy of [machine learning](https://en.wikipedia.org/wiki/Machine_learning) algorithms used in [statistical classification](https://en.wikipedia.org/wiki/Statistical_classification) and [regression](https://en.wikipedia.org/wiki/Regression_analysis). It also reduces [variance](https://en.wikipedia.org/wiki/Variance) and helps to avoid [overfitting](https://en.wikipedia.org/wiki/Overfitting). Although it is usually applied to [decision tree](https://en.wikipedia.org/wiki/Decision_tree_learning) methods, it can be used with any type of method. Bagging is a special case of the [model averaging](https://en.wikipedia.org/wiki/Ensemble_learning) approach. The bootstrap is a powerful statistical method for estimating a quantity from a data sample. This is easiest to understand if the quantity is a descriptive statistic such as a mean or a standard deviation.

**3. DECISION TREE ALGORITHM**

A tree has many analogies in real life, and turns out that it has influenced a wide area of **machine learning**, covering both **classification and regression**. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. A decision tree is drawn upside down with its root at the top.

It is one of the predictive modelling approaches used in [statistics](https://en.wikipedia.org/wiki/Statistics), [data mining](https://en.wikipedia.org/wiki/Data_mining) and [machine learning](https://en.wikipedia.org/wiki/Machine_learning). Tree models where the target variable can take a discrete set of values are called **classification trees**; in these tree structures, [leaves](https://en.wikipedia.org/wiki/Leaf_node) represent class labels and branches represent [conjunctions](https://en.wikipedia.org/wiki/Logical_conjunction) of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically [real numbers](https://en.wikipedia.org/wiki/Real_numbers)) are called **regression trees**.

**4. CONFUSION MATRIX**

A confusion matrix is a technique for summarizing the performance of a classification algorithm. Classification accuracy alone can be misleading if you have an unequal number of observations in each class or if you have more than two classes in your dataset. Calculating a confusion matrix can give you a better idea of what your classification model is getting right and what types of errors it is making.

**How to Calculate a Confusion Matrix**

**Below is the process for calculating a confusion Matrix.**

1. You need a test dataset or a validation dataset with expected outcome values.
2. Make a prediction for each row in your test dataset.
3. From the expected outcomes and predictions count:
   1. The number of correct predictions for each class.
   2. The number of incorrect predictions for each class, organized by the class that was predicted.

These numbers are then organized into a table or a matrix as follows:

* **Expected down the side**: Each row of the matrix corresponds to a predicted class.
* **Predicted across the top**: Each column of the matrix corresponds to an actual class.

The counts of correct and incorrect classification are then filled into the table.

The total number of correct predictions for a class go into the expected row for that class value and the predicted column for that class val

**SOURCE CODE**

**library**("caret")

set.seed(12345) *#set seed in random number generator for the sake of reproducibility.*

*# Download the training and testing data if have not done so.*

*#download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv", destfil="./pml-training.csv",)*

*#download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv", destfil="./pml-testing.csv",)*

*# Load training and testing data*

*# Replace invalid strings as "NA"*

trainingdata <- read.csv("./pml-training.csv",na.strings=c("NA","#DIV/0!",""))

testingdata <- read.csv("./pml-testing.csv",na.strings=c("NA","#DIV/0!",""))

dim(trainingdata)

## [1] 19622 160

dim(testingdata)

## [1] 20 160

*# Delete any columns containg NAs in testingdata*

training <- trainingdata[,colSums(is.na(testingdata))==0]

testing <- testingdata[,colSums(is.na(testingdata))==0]

*# Delete irrelevent columns [X, user\_name, raw\_timestamp\_part\_1, raw\_timestamp\_part\_2, cvtd\_timestamp, new\_window, num\_window]*

training <-training[,-c(1:7)]

testing <-testing[,-c(1:7)]

*# Take a look at the data after clearning*

dim(training)

## [1] 19622 53

dim(testing)

## [1] 20 53

#### Cross Validation

Split the original training data to subTraining(75%) and subTesting(25%) for cross validation. Fit the model using subTraining data and then predit using subTesting data. The accuracy of prediction should reflect the accuracy of the model.

*# Divide training data to subtraining and subtesting (75% subtraining, 25% subtesting)*

inTrain <- createDataPartition(y=training$classe, p=0.75, list=F,)

subTraining <- training[inTrain,]

subTesting <- training[-inTrain,]

#### Try various classification algorithm for comparison.

#### Decision Tree algorithm

**library**("rpart")

model\_dt <- rpart(classe ~., data=subTraining, method="class")

pred\_dt <- predict(model\_dt, subTesting, type="class")

res\_dt <- confusionMatrix(pred\_dt,subTesting$classe)

res\_dt

## Confusion Matrix and Statistics

## Reference

## Prediction A B C D E

## A 1260 156 33 40 23

## B 52 555 73 52 52

## C 24 136 575 83 95

## D 40 33 150 513 89

## E 19 69 24 116 642

## Overall Statistics

## Accuracy : 0.723

## 95% CI : (0.71, 0.735)

## No Information Rate : 0.284

## P-Value [Acc > NIR] : <2e-16

## Kappa : 0.649

## Mcnemar's Test P-Value : <2e-16

## Statistics by Class:

## Class: A Class: B Class: C Class: D Class: E

## Sensitivity 0.903 0.585 0.673 0.638 0.713

## Specificity 0.928 0.942 0.917 0.924 0.943

## Pos Pred Value 0.833 0.708 0.630 0.622 0.738

## Neg Pred Value 0.960 0.904 0.930 0.929 0.936

## Prevalence 0.284 0.194 0.174 0.164 0.184

## Detection Rate 0.257 0.113 0.117 0.105 0.131

## Detection Prevalence 0.308 0.160 0.186 0.168 0.177

## Balanced Accuracy 0.916 0.763 0.795 0.781 0.828

#### Bagging algorithm

**library**("ipred")

model\_bagging <- bagging(classe ~., data=subTraining)

pred\_bagging <- predict(model\_bagging, subTesting)

res\_bagging <- confusionMatrix(pred\_bagging, subTesting$classe)

res\_bagging

## Confusion Matrix and Statistics

## Reference

## Prediction A B C D E

## A 1391 6 0 0 0

## B 3 936 3 1 1

## C 1 7 847 10 4

## D 0 0 5 793 6

## E 0 0 0 0 890

## Overall Statistics

## Accuracy : 0.99

## 95% CI : (0.987, 0.993)

## No Information Rate : 0.284

## P-Value [Acc > NIR] : <2e-16

## Kappa : 0.988

## Mcnemar's Test P-Value : NA

## Statistics by Class:

## Class: A Class: B Class: C Class: D Class: E

## Sensitivity 0.997 0.986 0.991 0.986 0.988

## Specificity 0.998 0.998 0.995 0.997 1.000

## Pos Pred Value 0.996 0.992 0.975 0.986 1.000

## Neg Pred Value 0.999 0.997 0.998 0.997 0.997

## Prevalence 0.284 0.194 0.174 0.164 0.184

## Detection Rate 0.284 0.191 0.173 0.162 0.181

## Detection Prevalence 0.285 0.192 0.177 0.164 0.181

## Balanced Accuracy 0.998 0.992 0.993 0.992 0.994

#### Random Forest (improved bagging) algorithm

*# Use randomForest model to train and predict*

*#install.packages("randomForest")*

**library**("randomForest") *#Random forest for classification and regression*

## randomForest 4.6-10

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

model\_rf <- randomForest(classe ~., data=subTraining, na.action=na.omit)

pred\_rf <- predict(model\_rf, subTesting, type="class")

*# Summarize randomForest results.*

res\_rf <- confusionMatrix(pred\_rf,subTesting$classe)

res\_rf

## Confusion Matrix and Statistics

## Reference

## Prediction A B C D E

## A 1395 7 0 0 0

## B 0 938 3 0 0

## C 0 4 850 7 1

## D 0 0 2 797 4

## E 0 0 0 0 896

## Overall Statistics

## Accuracy : 0.994

## 95% CI : (0.992, 0.996)

## No Information Rate : 0.284

## P-Value [Acc > NIR] : <2e-16

## Kappa : 0.993

## Mcnemar's Test P-Value : NA

## Statistics by Class:

## Class: A Class: B Class: C Class: D Class: E

## Sensitivity 1.000 0.988 0.994 0.991 0.994

## Specificity 0.998 0.999 0.997 0.999 1.000

## Pos Pred Value 0.995 0.997 0.986 0.993 1.000

## Neg Pred Value 1.000 0.997 0.999 0.998 0.999

## Prevalence 0.284 0.194 0.174 0.164 0.184

## Detection Rate 0.284 0.191 0.173 0.163 0.183

## Detection Prevalence 0.286 0.192 0.176 0.164 0.183

## Balanced Accuracy 0.999 0.994 0.996 0.995 0.997

#### Select the model

Compare the accuracy of trained model on subTesting data, and choose highest accuracy model => randomForest. We observe slightly accuracy improvement for RandomForest compared with genetic Bagging algorithm.

df\_res <- data.frame(res\_dt$overall, res\_bagging$overall, res\_rf$overall)

df\_res

## res\_dt.overall res\_bagging.overall res\_rf.overall

## Accuracy 7.229e-01 0.9904 0.9943

## Kappa 6.486e-01 0.9879 0.9928

## AccuracyLower 7.101e-01 0.9873 0.9918

## AccuracyUpper 7.354e-01 0.9929 0.9962

## AccuracyNull 2.845e-01 0.2845 0.2845

## AccuracyPValue 0.000e+00 0.0000 0.0000

## McnemarPValue 4.774e-26 NaN NaN

#### Final Prediction Results

Apply the randomForest trained model on testing data, and get its testing results. RandomForest has 99.4% accuracy 95% CI : (0.992, 0.996), so I expect that less than 1 prediction can be wrong in 20 predictions.

*# Predict testing results using trained model*

res <- predict(model\_rf, testing, type="class")

res

## 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

## B A B A A E D B A A B C B A E E A B B B

## Levels: A B C D E

In the same way, the total number of incorrect predictions for a class go into the expected row for that class value and the predicted column for that class value.

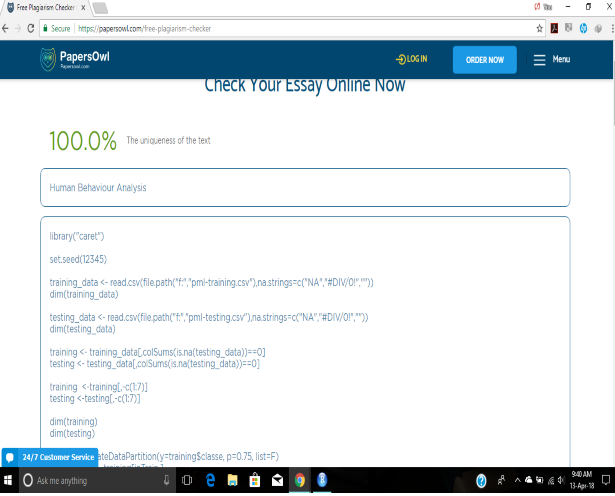
**CONCLUSION**

It was a wonderful experience for us while working on this project. This project took us through the various phases of software development and gave me real insight into the world of software development. We enjoyed each and every bit of work we had put into this project. This project is further extendable.

**REFERENCES**

* <https://www.rstudio.com/online-learning/>
* <https://imotions.com/blog/human-behavior/>

**PLAGIARISM CHECK**

****