Predicting Absenteeism using Machine Learning Techniques

Laxman Panthi & Zubin Shah

Introduction, Problem Statement & Motivation:

Employees tend to remain absent for variety of reasons; may be because of some health-related issues, personal reasons and there are ample of factors dependent behind it. Employers expect the tasks to be completed on time so none of the project timelines are deviated and they are within the scope & budget of the project.

Specially talking about the courier companies, on time delivery is the priority for customer satisfaction and if the company fails on this, there is a potential loss of business, trust and brand image. Absenteeism had been becoming an issue for this industry and companies are still facing issues to find the root causes.

In this project we will consider this major issue of absenteeism and determine strategies to dealing with such issues. Firstly, we will study the database that has the information of collected records of absenteeism from work during the period of July/07 to July/2010. Later, based on the analysis strategies would be determined to find out solutions.

Data:

The given dataset is the information collected records of absenteeism from work during the period of July/07 to July/2010 in a Courier company. Absences certified with the International Classification of Diseases were stratified into 21 categories:

I Certain infectious and parasitic diseases

II Neoplasms

III Diseases of the blood and blood-forming organs and certain disorders involving the immune mechanism

IV Endocrine, nutritional and metabolic diseases

V Mental and behavioural disorders

VI Diseases of the nervous system

VII Diseases of the eye and adnexa

VIII Diseases of the ear and mastoid process

IX Diseases of the circulatory system

X Diseases of the respiratory system

XI Diseases of the digestive system

XII Diseases of the skin and subcutaneous tissue

XIII Diseases of the musculoskeletal system and connective tissue

XIV Diseases of the genitourinary system

XV Pregnancy, childbirth and the puerperium

XVI Certain conditions originating in the perinatal period

XVII Congenital malformations, deformations and chromosomal abnormalities

XVIII Symptoms, signs and abnormal clinical and laboratory findings, not elsewhere classified

XIX Injury, poisoning and certain other consequences of external causes

XX External causes of morbidity and mortality

XXI Factors influencing health status and contact with health services.

Apart from these categories, the database also has other attributes like:

-Individual identification (ID)

- Month of absence

-Day of the week (Monday (2), Tuesday (3), Wednesday (4), Thursday (5), Friday (6))

-Seasons (summer (1), autumn (2), winter (3), spring (4))

-Transportation expense

-Distance from Residence to Work (kilometers)

-Service time

-Age

-Work load Average/day

-Hit target

- Disciplinary failure (yes=1; no=0)

- Education (high school (1), graduate (2), postgraduate (3), master and doctor (4))

- Son (number of children)

-Social drinker (yes=1; no=0)

- Social smoker (yes=1; no=0)

- Pet (number of pet)

- Weight

- Height

- Body mass index

- Absenteeism time in hours (target)

Exploring the Data:

EDA techniques would be used for exploring the data to gain more insights. Before getting started we will first need to pre-process the data. Initially the data was imported and evaluated for any missing values. As indicated in the code below, several categorical variables were converted to factors and the target variable (absenteeism time in hours) was grouped and converted to categorical variable (which was further converted to factor). Data structure was then evaluated to validate proper conversion.

Groups for Hours of absenteeism were converted as follows:

Group 0: Number of hours=0

Group 1: Number of hours=1

Group 2: Number of hours=2

Group 3: Number of hours=3

Group 4: 4<=Number of hours<=7

Group 5: Number of hours=8

Group 6: Number of hours=>9

Data Pre-Processing & Exploratory Data Ananlysis

```
#setting the working directory
setwd("C:\\Users\\zusha01\\Desktop\\HU Analytics\\Sem 3\\ANLY 530 Machine Lea
rning 1\\Final Project")

# Importing the dataset
dataset <- read.csv('Absenteeism_at_work.csv')

#Exploring the dataset
head(dataset)
```

```
##   ID Reason.for.absence Month.of.absence Day.of.the.week Seasons
## 1 11                 26                7               3       1
## 2 36                  0                7               3       1
## 3  3                 23                7               4       1
## 4  7                  7                7               5       1
## 5 11                 23                7               5       1
## 6  3                 23                7               6       1
##   Transportation.expense Distance.from.Residence.to.Work Service.time Age
## 1                    289                              36           13  33
## 2                    118                              13           18  50
## 3                    179                              51           18  38
## 4                    279                               5           14  39
## 5                    289                              36           13  33
## 6                    179                              51           18  38
##   Work.load.Average.day Hit.target Disciplinary.failure Education Son
## 1               239.554         97                    0         1   2
## 2               239.554         97                    1         1   1
## 3               239.554         97                    0         1   0
## 4               239.554         97                    0         1   2
## 5               239.554         97                    0         1   2
## 6               239.554         97                    0         1   0
##   Social.drinker Social.smoker Pet Weight Height Body.mass.index
## 1              1             0   1     90    172              30
## 2              1             0   0     98    178              31
## 3              1             0   0     89    170              31
## 4              1             1   0     68    168              24
## 5              1             0   1     90    172              30
## 6              1             0   0     89    170              31
##   Absenteeism.time.in.hours
## 1                         4
## 2                         0
## 3                         2
## 4                         4
## 5                         2
## 6                         2
```

```r
#checking for NA
sum(is.na(dataset))
```

```
## [1] 0
```

```r
# Converting Categorical variables to factors
dataset$ID <- as.factor(dataset$ID)
dataset$Month.of.absence <- as.factor(dataset$Month.of.absence)
dataset$Reason.for.absence <- as.factor(dataset$Reason.for.absence)
dataset$Day.of.the.week <- as.factor(dataset$Day.of.the.week)
dataset$Seasons <- as.factor(dataset$Seasons)
dataset$Disciplinary.failure <- as.factor(dataset$Disciplinary.failure)
dataset$Education <- as.factor(dataset$Education)
dataset$Son <- as.factor(dataset$Son)
```

```r
dataset$Social.drinker <- as.factor(dataset$Social.drinker)
dataset$Social.smoker <- as.factor(dataset$Social.smoker)
dataset$Pet <- as.factor(dataset$Pet)

# Encoding Absentism time in hours into categorical data
dataset$Absenteeism <- ifelse(dataset$Absenteeism >= 4 & dataset$Absenteeism
<=7, 4, dataset$Absenteeism)
dataset$Absenteeism <- ifelse(dataset$Absenteeism == 8 , 5, dataset$Absenteei
sm)
dataset$Absenteeism <- ifelse(dataset$Absenteeism >= 9 , 6, dataset$Absenteei
sm)
dataset$Absenteeism <- as.factor(dataset$Absenteeism)

#removing old absenteeism column
dataset <- dataset[,-21]

#evaluating the structure of the dataset
str(dataset)
```

```
## 'data.frame':    740 obs. of  21 variables:
##  $ ID                             : Factor w/ 36 levels "1","2","3","4",..
: 11 36 3 7 11 3 10 20 14 1 ...
##  $ Reason.for.absence             : Factor w/ 28 levels "0","1","2","3",..
: 26 1 23 8 23 23 22 23 20 22 ...
##  $ Month.of.absence               : Factor w/ 13 levels "0","1","2","3",..
: 8 8 8 8 8 8 8 8 8 8 ...
##  $ Day.of.the.week                : Factor w/ 5 levels "2","3","4","5",..:
2 2 3 4 4 5 5 5 1 1 ...
##  $ Seasons                        : Factor w/ 4 levels "1","2","3","4": 1
1 1 1 1 1 1 1 1 1 ...
##  $ Transportation.expense         : int  289 118 179 279 289 179 361 260 1
55 235 ...
##  $ Distance.from.Residence.to.Work: int  36 13 51 5 36 51 52 50 12 11 ...
##  $ Service.time                   : int  13 18 18 14 13 18 3 11 14 14 ...
##  $ Age                            : int  33 50 38 39 33 38 28 36 34 37 ...
##  $ Work.load.Average.day          : num  240 240 240 240 240 ...
##  $ Hit.target                     : int  97 97 97 97 97 97 97 97 97 97 ...
##  $ Disciplinary.failure           : Factor w/ 2 levels "0","1": 1 2 1 1 1
1 1 1 1 1 ...
##  $ Education                      : Factor w/ 4 levels "1","2","3","4": 1
1 1 1 1 1 1 1 1 3 ...
##  $ Son                            : Factor w/ 5 levels "0","1","2","3",..:
3 2 1 3 3 1 2 5 3 2 ...
##  $ Social.drinker                 : Factor w/ 2 levels "0","1": 2 2 2 2 2
2 2 2 2 1 ...
##  $ Social.smoker                  : Factor w/ 2 levels "0","1": 1 1 1 2 1
1 1 1 1 1 ...
##  $ Pet                            : Factor w/ 6 levels "0","1","2","4",..:
2 1 1 1 2 1 4 1 1 2 ...
##  $ Weight                         : int  90 98 89 68 90 89 80 65 95 88 ...
```

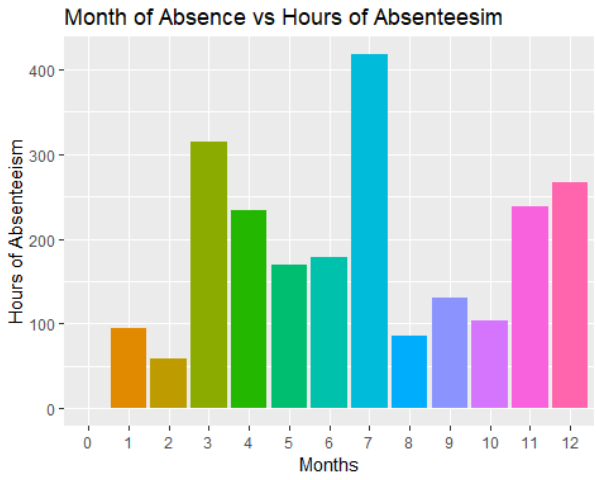
```
##  $ Height                          : int  172 178 170 168 172 170 172 168 1
96 172 ...
##  $ Body.mass.index                 : int  30 31 31 24 30 31 27 23 25 29 ...
##  $ Absenteeism                     : Factor w/ 7 levels "0","1","2","3",..:
5 1 3 5 3 3 6 5 7 6 ...
```

Exploratory Data Analysis:

The following questions could easily be addressed using EDA techniques based on the database:

1. **In which month are the hours of absenteeism the highest & lowest?**

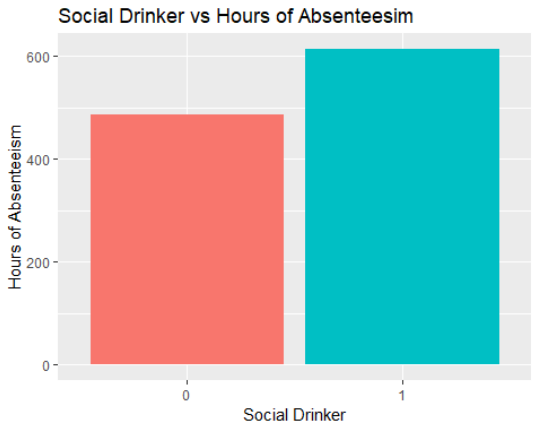
```
library(ggplot2)

## Warning: package 'ggplot2' was built under R version 3.5.2

#month of absence vs absenteeism
ggplot(data=dataset, aes(x=dataset$Month.of.absence, y= dataset$Absenteeism.t
ime.in.hours, fill= dataset$Month.of.absence)) +
       geom_bar(stat="sum") +
  ggtitle("Month of Absence vs Hours of Absenteesim")+xlab("Months")+ylab("Ho
urs of Absenteeism")+
  theme(legend.position="none")
```

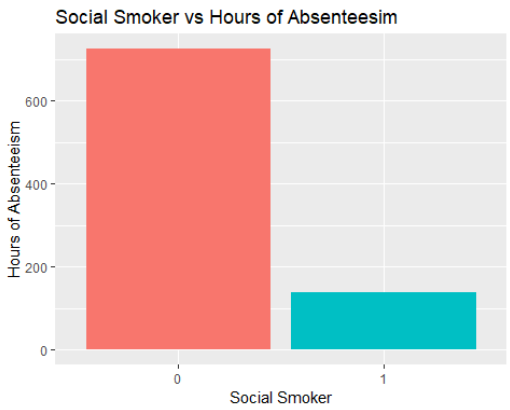


Based on the above, we can conclude the highest hours of absenteeism occur in the month of July and lowest in February.

2. **Do social drinkers and social smokers impact hours of absenteeism?**

```
#Social drinker vs Absenteeism
ggplot(data=dataset, aes(x=dataset$Social.drinker, y= dataset$Absenteeism.tim
e.in.hours, fill= dataset$Social.drinker)) +
       geom_bar(stat="sum") +
  ggtitle("Social Drinker vs Hours of Absenteesim")+xlab("Social Drinker")+yl
ab("Hours of Absenteeism")+
  theme(legend.position="none")


#Social smoker vs Absenteeism
ggplot(data=dataset, aes(x=dataset$Social.smoker, y= dataset$Absenteeism.time
.in.hours, fill= dataset$Social.smoker)) +
       geom_bar(stat="sum") +
  ggtitle("Social Smoker vs Hours of Absenteesim")+xlab("Social Smoker")+ylab
("Hours of Absenteeism")+
  theme(legend.position="none")
```



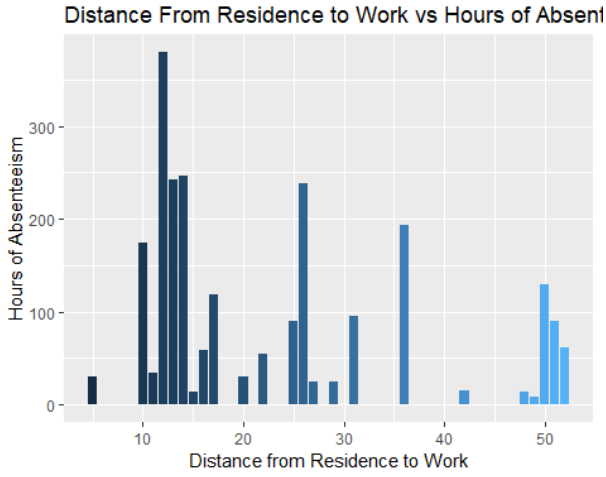


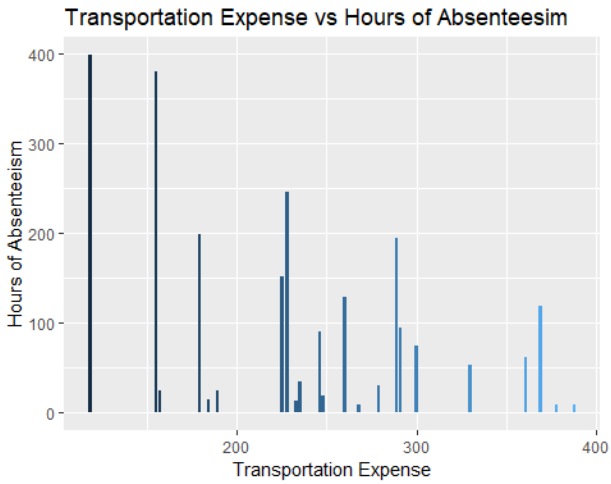
From above, we can conclude that social drinker and non-social smoker is absent more often compared to a non-social drinker and a social smoker.

3. **Does distance from residence to work and travel expense have any pattern to hours of absenteeism?**

```
#Distance from Residence vs Absenteeism
ggplot(data=dataset, aes(x=dataset$Distance.from.Residence.to.Work, y= datase
t$Absenteeism.time.in.hours, fill= dataset$Distance.from.Residence.to.Work))
+
       geom_bar(stat="sum") +
  ggtitle("Distance From Residence to Work vs Hours of Absenteesim")+xlab("Di
```

```
stance from Residence to Work")+ylab("Hours of Absenteeism")+
  theme(legend.position="none")

#Transportation expense vs Absenteeism
ggplot(data=dataset, aes(x=dataset$Transportation.expense, y= dataset$Absente
eism.time.in.hours, fill= dataset$Transportation.expense)) +
       geom_bar(stat="sum") +
  ggtitle("Transportation Expense vs Hours of Absenteesim")+xlab("Transportat
ion Expense")+ylab("Hours of Absenteeism")+
  theme(legend.position="none")
```



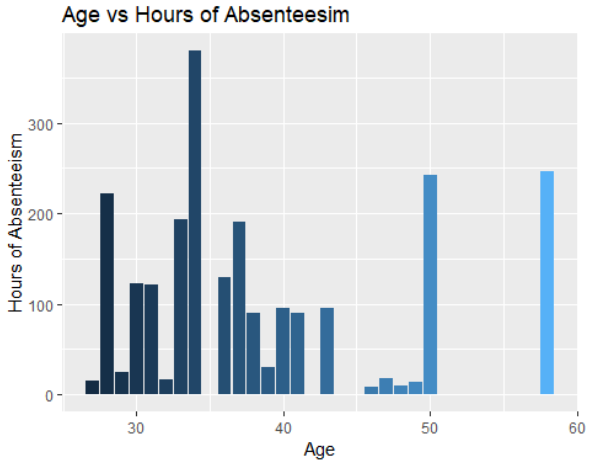


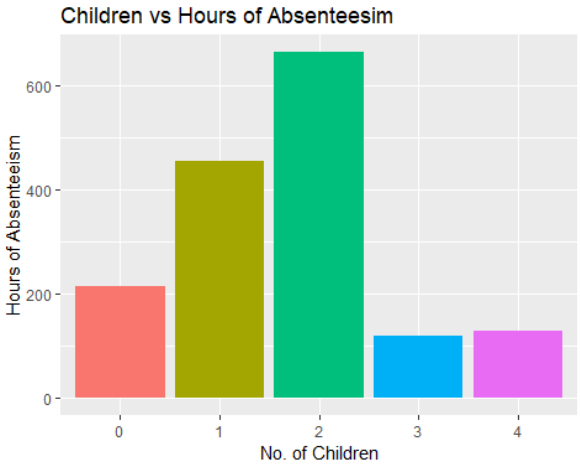
It looks like employees staying close-by are absent more often compared to those living far.

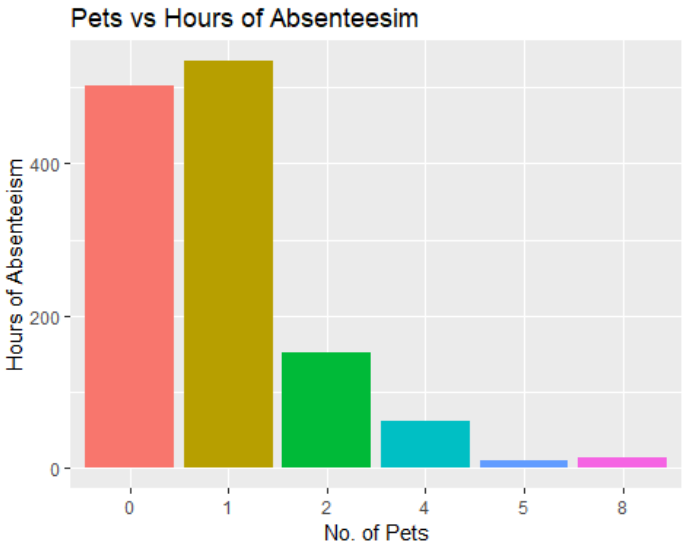
4. Does age, number of children and pets impact the hours of absenteeism?

```
#Age vs Absenteesim
ggplot(data=dataset, aes(x=dataset$Age, y= dataset$Absenteeism.time.in.hours,
fill= dataset$Age)) +
       geom_bar(stat="sum") +
  ggtitle("Age vs Hours of Absenteesim")+xlab("Age")+ylab("Hours of Absenteei
sm")+
  theme(legend.position="none")


#Children vs Absenteesim
ggplot(data=dataset, aes(x=dataset$Son, y= dataset$Absenteeism.time.in.hours,
fill= dataset$Son)) +
       geom_bar(stat="sum") +
```

```
  ggtitle("Children vs Hours of Absenteesim")+xlab("No. of Children")+ylab("H
ours of Absenteeism")+
  theme(legend.position="none")


#Pets vs Absenteesim
ggplot(data=dataset, aes(x=dataset$Pet, y= dataset$Absenteeism.time.in.hours,
fill= dataset$Pet)) +
       geom_bar(stat="sum") +
  ggtitle("Pets vs Hours of Absenteesim")+xlab("No. of Pets")+ylab("Hours of
Absenteeism")+
  theme(legend.position="none")
```







Employees with lower age, 1-2 children and 0-1 pets are absent more often compared to others.

These insights might be useful while hiring a new employee when their personal information is shared. But what happens if you already have an employee and you don't want to lose him?

Predictive Analytics:

Based on the above mention scenario, you would need to predict the number of hours of absenteeism when all the other information about the employee is provided. If we can predict this, then in most of the scenarios we could save our business being hampered by having some temp workers to cover it up.

Since the hours of absenteeism has been grouped to various categories, we would use classification techniques of machine learning to address this issue.

Different classification machine learning techniques have been tried like: K-Nearest Neighbor (KNN), Support Vector Machines (SVM -Linear & Kernel), Naïve Bayes, Decision Trees and Random Forest and further confusion matrix was created for each model and accuracy of the model was determined.

Before implementing any model, the data was divided in training set (80%) and test set (20%). Feature scaling technique was used to normalize the data. Initially the models used all the features to predict. Later, Gini Index was used for feature selection and model improvement was performed.

Machine Learning - Classification Techniques

```
# Splitting the dataset into the Training set and Test set
# install.packages('caTools')
library(caTools)
set.seed(12345)
split = sample.split(dataset$Absenteeism, SplitRatio = 0.8)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)


# Feature Scaling
training_set[,c(6:11, 18:20)] = scale(training_set[,c(6:11, 18:20)])
test_set[,c(6:11, 18:20)] = scale(test_set[,c(6:11, 18:20)])
```

KNN

```
#Finding the optimum number of k
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 3.5.2
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.5.3
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.5.2
```

```
set.seed(12345)
ctrl <- trainControl(method="repeatedcv",repeats = 5)
knnFit <- train(Absenteeism ~ ., data = training_set, method = "knn", trContr
ol = ctrl, preProcess = "center",tuneLength = 20)
k <- knnFit$bestTune[,1]

# Fitting K-NN to the Training set and Predicting the Test set results
library(class)
```

```
## Warning: package 'class' was built under R version 3.5.3
```

```
y_pred = knn(train = training_set[, -21],
             test = test_set[, -21],
             cl = training_set[, 21],
             k = k,
             prob = TRUE)

# Making the Confusion Matrix
cm = table(test_set[, 21], y_pred)
cm
```

```
##    y_pred
##      0  1  2  3  4  5  6
##   0  8  0  0  0  0  1  0
##   1  0  5  6  2  2  3  0
##   2  0  7 16  5  0  3  0
##   3  0  2  4  7  2  7  0
##   4  0  0  7  1  3  3  0
##   5  3  1  6  7  2 22  1
##   6  0  0  3  2  1  5  2
```

```
#Model Accuracy
knn_Accuracy <- sum(diag(cm))*100/sum(cm)
knn_Accuracy
```

```
## [1] 42.28188
```

SVM: Linear

```
# Fitting SVM to the Training set
# install.packages('e1071')
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.5.3
```

```
classifier = svm(formula = Absenteeism ~ .,
                 data = training_set,
                 type = 'C-classification',
                 kernel = 'linear')

# Predicting the Test set results
y_pred = predict(classifier, newdata = test_set[-21])

# Making the Confusion Matrix
cm = table(test_set[, 21], y_pred)
cm
```

```
##    y_pred
##      0  1  2  3  4  5  6
##   0  9  0  0  0  0  0  0
##   1  0  5  7  2  1  3  0
##   2  0  2 18  6  3  2  0
##   3  0  0  5  6  3  7  1
##   4  0  1  4  0  4  4  1
##   5  0  0  0  3  2 30  7
##   6  0  3  0  1  1  5  3
```

```
#Model Accuracy
SVM_Linear_Accuracy <- sum(diag(cm))*100/sum(cm)
SVM_Linear_Accuracy
```

```
## [1] 50.33557
```

SVM: Kernel

```
# Fitting Kernel SVM to the Training set
# install.packages('e1071')
library(e1071)
classifier = svm(formula = Absenteeism ~ .,
                 data = training_set,
                 type = 'C-classification',
                 kernel = 'radial')
# Predicting the Test set results
y_pred = predict(classifier, newdata = test_set[-21])

# Making the Confusion Matrix
cm = table(test_set[, 21], y_pred)
cm
```

```
##    y_pred
##      0  1  2  3  4  5  6
##   0  0  0  0  0  0  9  0
##   1  0  0  8  4  0  6  0
##   2  0  0 18  5  0  8  0
##   3  0  0 14  4  0  4  0
##   4  0  0  5  2  0  7  0
##   5  0  0  7  3  0 32  0
##   6  0  0  3  0  0 10  0
```

```
#Model Accuracy
SVM_Kernel_Accuracy <- sum(diag(cm))*100/sum(cm)
SVM_Kernel_Accuracy
```

```
## [1] 36.24161
```

Naive Bayes

```
# Fitting SVM to the Training set
# install.packages('e1071')
library(e1071)
classifier = naiveBayes(x = training_set[-21],
                        y = training_set$Absenteeism)

# Predicting the Test set results
y_pred = predict(classifier, newdata = test_set[-21])

# Making the Confusion Matrix
cm = table(test_set[, 21], y_pred)
cm
```

```
##    y_pred
##      0  1  2  3  4  5  6
##   0  9  0  0  0  0  0  0
##   1  0  5  4  4  4  0  1
##   2  0  5 12  8  3  2  1
##   3  0  2  8  5  2  3  2
##   4  0  1  2  3  3  4  1
##   5  0  2  5  3  3 21  8
##   6  0  2  3  0  2  2  4
```

```
#Model Accuracy
Naive_Bayes_Accuracy <- sum(diag(cm))*100/sum(cm)
Naive_Bayes_Accuracy
```

```
## [1] 39.59732
```

Random Forest

```
# Fitting Random Forest Classification to the Training set
# install.packages('randomForest')
library(randomForest)

## Warning: package 'randomForest' was built under R version 3.5.3

## randomForest 4.6-14

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

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

set.seed(12345)
classifier = randomForest(x = training_set[-21],
                          y = training_set$Absenteeism,
                          ntree = 500)

# Predicting the Test set results
y_pred = predict(classifier, newdata = test_set[-21])

# Making the Confusion Matrix
cm = table(test_set[, 21], y_pred)
cm

##    y_pred
##      0  1  2  3  4  5  6
##   0  9  0  0  0  0  0  0
##   1  0  6  6  1  2  2  1
##   2  0  4 15  7  1  4  0
##   3  0  0  7  3  4  8  0
##   4  0  1  3  1  3  5  1
##   5  0  0  2  2  0 33  5
##   6  0  0  2  2  0  8  1

#Model Accuracy
Random_Forest_Accuracy <- sum(diag(cm))*100/sum(cm)
Random_Forest_Accuracy

## [1] 46.97987
```

Model Improvement

There are different approaches that can be taken to optimize an already existing model. The optimization of model does not necessarily need to increase the accuracy of the model but also should simplify the model and help in its usability.

Methods of Model Optimization

- Feature Selection

  All the variables in the dataset might not always be useful to the model and they could act as suppressor in case they are highly correlated with the other variables. We implement a method called GINI Index method in our models which is a method of Feature Ranking. By ranking the features by importance, we identify and remove the features that have very high gini index (taking 50 as a cut-off)

- Hybrid Models / Dimensionality Reduction

  An unsupervised algorithm could be used to reduce the dimension of the dataset by labelling each observation with the cluster it would belong to. And, unsupervised methods like PCA could be used to reduce the dimension of model by creating PCA's.

  Can we implement Unsupervised Algorithm to the problem?

  Short answer is No. Since the data is labelled, the unsupervised learning becomes obsolete here as the outcome of the unsupervised learning is not rigid. However, the unsupervised model could be used to help us develop more features in the model.

- Tuning Hyper Parameters

  This method was tried out in the base decision tree model, where the model had a hard time in converging due to the limited number of default iterations it goes through while training the model. Also based on the type of the problem statement (classification or regression), the type of activation function needs to be changed in supervised learning methods.

FEATURE SELECTION

USING GINI INDEX

```
library(randomForest)
fit_rf=randomForest(Absenteeism~., data=dataset)
# Create an importance based on mean decreasing gini
importance(fit_rf)
```

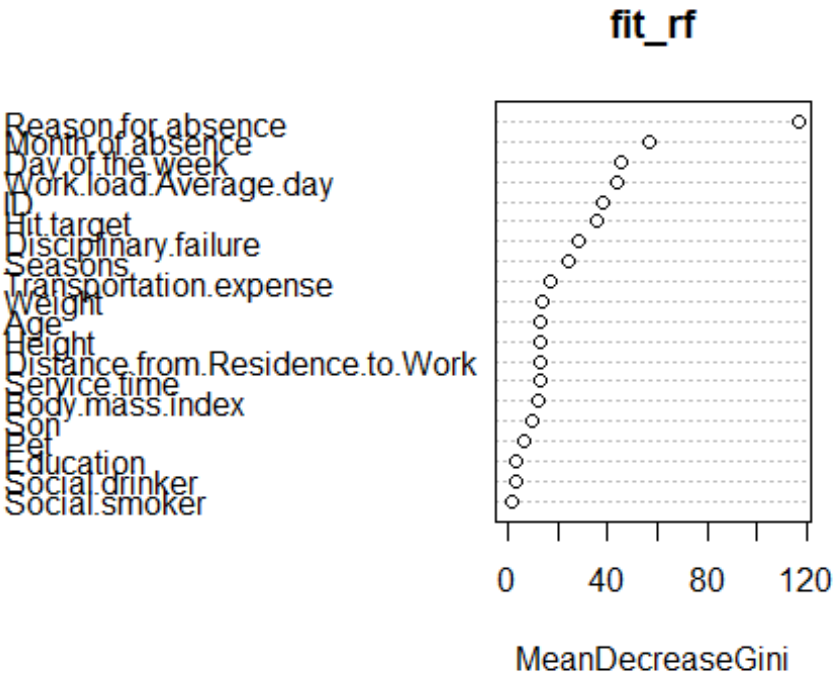
```
##                                 MeanDecreaseGini
## ID                                     38.217417
```

```
## Reason.for.absence                      117.293441
## Month.of.absence                         56.449868
## Day.of.the.week                          45.785465
## Seasons                                  24.020285
## Transportation.expense                   16.965968
## Distance.from.Residence.to.Work          12.651311
## Service.time                             12.463135
## Age                                      13.207693
## Work.load.Average.day                    43.615698
## Hit.target                               35.364419
## Disciplinary.failure                     28.471000
## Education                                 3.463373
## Son                                       9.353355
## Social.drinker                            3.317196
## Social.smoker                             1.267381
## Pet                                       6.500636
## Weight                                   13.820125
## Height                                   12.769449
## Body.mass.index                          11.917768
```

```
# compare the feature importance with varImp() function
varImp(fit_rf)
```

```
##                                   Overall
## ID                               38.217417
## Reason.for.absence              117.293441
## Month.of.absence                 56.449868
## Day.of.the.week                  45.785465
## Seasons                          24.020285
## Transportation.expense           16.965968
## Distance.from.Residence.to.Work  12.651311
## Service.time                     12.463135
## Age                              13.207693
## Work.load.Average.day            43.615698
## Hit.target                       35.364419
## Disciplinary.failure             28.471000
## Education                         3.463373
## Son                               9.353355
## Social.drinker                    3.317196
## Social.smoker                     1.267381
## Pet                               6.500636
## Weight                           13.820125
## Height                           12.769449
## Body.mass.index                  11.917768
```

```
# Create a plot of importance scores by random forest
varImpPlot(fit_rf)
```




```
#Idenfitying feature with index >50
which(fit_rf$importance[,1]>50)

## Reason.for.absence   Month.of.absence
##                  2                  3

training_set1 <- training_set[,c(2,3,21)]
test_set1 <- test_set[,c(2,3,21)]
```

IMPLEMETING FEATURE SELECTION:

There are two features that are deemed not useful from our finding above. Now, we will run our models again with the new train & test dataets.

SVM

```
# Fitting SVM to the Training set
# install.packages('e1071')
library(e1071)
classifier = svm(formula = Absenteeism ~ .,
                 data = training_set1,
                 type = 'C-classification',
                 kernel = 'linear')

# Predicting the Test set results
```

```r
y_pred = predict(classifier, newdata = test_set1[-3])

# Making the Confusion Matrix
cm = table(test_set1[, 3], y_pred)
cm

##    y_pred
##      0  1  2  3  4  5  6
##   0  9  0  0  0  0  0  0
##   1  0  4  6  0  4  4  0
##   2  0  2 23  0  3  3  0
##   3  0  2 10  0  2  8  0
##   4  0  2  4  0  3  5  0
##   5  0  1  1  0  0 39  1
##   6  0  0  0  1  0  9  3

#Model Accuracy
SVM_Improved <- sum(diag(cm))*100/sum(cm)
SVM_Improved

## [1] 54.36242

##Random Forest: NOT IMPROVED

 #Fitting Random Forest Classification to the Training set
# install.packages('randomForest')
library(randomForest)
set.seed(12345)
classifier = randomForest(x = training_set1[-3],
                          y = training_set1$Absenteeism,
                          ntree = 500)

# Predicting the Test set results
y_pred = predict(classifier, newdata = test_set1[-3])

# Making the Confusion Matrix
cm = table(test_set1[, 3], y_pred)
cm

##    y_pred
##      0  1  2  3  4  5  6
##   0  9  0  0  0  0  0  0
##   1  0  4  5  4  1  3  1
##   2  0  2 17  6  2  4  0
##   3  0  2  8  4  1  7  0
##   4  0  3  4  0  2  5  0
##   5  0  2  2  3  1 28  6
##   6  0  0  2  1  0  8  2
```

```r
#Model Accuracy
Random_Improved <- sum(diag(cm))*100/sum(cm)
Random_Improved
```

```
## [1] 44.2953
```

Naive Bayes

```r
# Fitting SVM to the Training set
# install.packages('e1071')
library(e1071)
classifier = naiveBayes(x = training_set1[-3],
                        y = training_set1$Absenteeism)

# Predicting the Test set results
y_pred = predict(classifier, newdata = test_set1[-3])

# Making the Confusion Matrix
cm = table(test_set1[, 3], y_pred)
cm
```

```
##    y_pred
##      0  1  2  3  4  5  6
##   0  9  0  0  0  0  0  0
##   1  0  6  4  0  4  4  0
##   2  0  5 20  2  1  2  1
##   3  0  3  9  0  2  8  0
##   4  0  1  3  0  5  5  0
##   5  0  4  2  1  0 27  8
##   6  0  0  0  3  0  7  3
```

```r
#Model Accuracy
Naive_Improved <- sum(diag(cm))*100/sum(cm)
Naive_Improved
```

```
## [1] 46.97987
```

KNN

```r
#Finding the optimum number of k
library(ISLR)
library(caret)
set.seed(12345)
ctrl <- trainControl(method="repeatedcv",repeats = 5)
knnFit <- train(Absenteeism ~ ., data = training_set1, method = "knn", trCont
rol = ctrl, preProcess = "center",tuneLength = 20)
k <- knnFit$bestTune[,1]
```

```r
# Fitting K-NN to the Training set and Predicting the Test set results
library(class)
y_pred = knn(train = training_set1[, -3],
             test = test_set1[, -3],
             cl = training_set1[, 3],
             k = k,
             prob = TRUE)

# Making the Confusion Matrix
cm = table(test_set1[, 3], y_pred)
cm
```

```
##    y_pred
##      0  1  2  3  4  5  6
##   0  9  0  0  0  0  0  0
##   1  0  4  5  0  3  5  1
##   2  0  3 21  0  1  6  0
##   3  0  1 12  1  0  8  0
##   4  0  0  7  0  3  4  0
##   5  4  2  7  1  0 27  1
##   6  0  0  0  1  0 10  2
```

```r
#Model Accuracy
knn_Improved <- sum(diag(cm))*100/sum(cm)
knn_Improved
```

```
## [1] 44.96644
```

Decision Tree

```r
# Fitting Decision Tree Classification to the Training set
# install.packages('rpart')
library(rpart)
classifier = rpart(formula = Absenteeism ~ ., data = training_set1, method =
"class")

# Predicting the Test set results
y_pred = predict(classifier, newdata = test_set1[-3], type = 'class')

# Making the Confusion Matrix
cm = table(test_set1[, 3], y_pred)
cm
```

```
##    y_pred
##      0  1  2  3  4  5  6
##   0  9  0  0  0  0  0  0
##   1  0  6  6  0  2  4  0
##   2  0  4 22  0  2  3  0
```

```
##   3  0  5 10  0  0  7  0
##   4  0  2  4  0  3  5  0
##   5  0  4  3  0  1 34  0
##   6  0  0  2  0  0 11  0

#Model Accuracy
Decision_Improved <- sum(diag(cm))*100/sum(cm)
Decision_Improved

## [1] 49.66443
```

The optimized model gave us some interesting results. Firstly, we were able to run the decision tree as we took out two variables with large number of levels. Other than that, our Linear SVM model improved significantly in the accuracy.

Results & Discussion

We implemented 5 machine learning techniques to the dataset about the absenteeism in the courier company. The results are tabulated as follows:

Base Model

Model	KNN	SVM	Naïve Bayes	Random Forest
Accuracy	42.3%	Linear – 50.3% Kernel – 36.2%	39.6%	47%

Optimized Model

Model	KNN	SVM	Naïve Bayes	Random Forest	Decision Tree
Accuracy	44.96%	Linear – 54.36%	46.97%	44.29%	49.66%