Employee Absenteeism Project Report

Harshal Rajendra Kothawade

August 17, 2018

Contents

1 Introduction **2**
1.1 Project Description . . . 2
1.2 Problem statement . . . 2
1.3 Data . . . 2

2 Methodology **4**
2.1 Pre Processing . . . 4
2.1.1 Missing Value Analysis . . . 4
2.1.2 Outlier Analysis . . . 4
2.1.3 Feature Selection . . . 6
2.1.4 Feature Scaling . . . 9
2.2 Modeling . . . 10
2.2.1 Model Selection . . . 10
2.2.2 Multiple Linear Regression . . . 10
2.2.3 Decision Tree . . . 11
2.2.4 Random Forest . . . 12

3 Conclusion **13**
3.1 Model Evaluation . . . 13
3.1.1 Accuracy . . . 13
3.2 Model Selection . . . 13
3.3 Solution . . . 14
3.3.1 Problem 1 . . . 14
3.3.2 Problem 2 . . . 16

A R Code **17**
A.1 Exploratory Data Analysis . . . 17
A.2 Missing Value Analysis . . . 17
A.3 Outlier Analysis . . . 18
A.4 Feature Selection . . . 18
A.5 Model Selection . . . 19
A.6 Conclusion . . . 19
A.7 Complete Code . . . 20

B Python Code **25**
B.1 Exploratory Data Analysis . . . 25
B.2 Missing Value Analysis . . . 26
B.3 Outlier Analysis . . . 26
B.4 Feature Selection . . . 26
B.5 Model Selection . . . 27
B.6 Conclusion . . . 27
B.7 Complete Code . . . 28

Chapter 1

Introduction

1.1 Project Description

XYZ is a courier company. As we appreciate that human capital plays an important role in collection, transportation and delivery. The company is passing through genuine issue of Absenteeism.

1.2 Problem statement

The company has shared it dataset and requested to have an answer on the following areas:

1. What changes company should bring to reduce the number of absenteeism?
2. How much losses every month can we project in 2011 if same trend of absenteeism continues?

1.3 Data

Dataset Details:
Dataset Characteristics: Timeseries Multivariant
Number of Attributes: 21
Missing Values : Yes

Attribute Information:
1. Individual identification (ID)
2. Reason for absence (ICD) (stratified into 28 categories)
3. Month of absence
4. Day of the week (Monday (2), Tuesday (3), Wednesday (4), Thursday (5), Friday (6))
5. Seasons (summer (1), autumn (2), winter (3), spring (4))
6. Transportation expense
7. Distance from Residence to Work (kilometers)
8. Service time
9. Age
10. Work load Average/day
11. Hit target
12. Disciplinary failure (yes=1; no=0)
13. Education (high school (1), graduate (2), postgraduate (3), master and doctor (4))
14. Son (number of children)
15. Social drinker (yes=1; no=0)
16. Social smoker (yes=1; no=0)
17. Pet (number of pet)
18. Weight
19. Height
20. Body mass index
21. Absenteeism time in hours (target)

Following is the glimse of the actaual data:

Table 1.1: Employee Absenteeism sapmle Data (Columns 1-6)

ID	Reason for absence	Month of absence	Day of the week	Seasons	Transportation expense
11	26	7	3	1	
36	0	7	3	1	
3	23	7	4	1	
7	7	7	5	1	
11	23	7	5	1	
3	23	7	6	1	
10	22	7	6	1	

Table 1.2: Employee Absenteeism sapmle Data (Columns 7-11)

Distance from Residence to Work	Service time	Age	Work load Average/day	Hit target
36	13	33	2,39,554	97
13	18	50	2,39,554	97
51	18	38	2,39,554	97
5	14	39	2,39,554	97
36	13	33	2,39,554	97
51	18	38	2,39,554	97
52	3	28	2,39,554	97

Table 1.3: Employee Absenteeism sapmle Data (Columns 12-16)

Disciplinary failure	Education	Son	Social drinker	Social smoker
0	1	2	1	0
1	1	1	1	0
0	1	0	1	0
0	1	2	1	1
0	1	2	1	0
0	1	0	1	0
0	1	1	1	0

Table 1.4: Employee Absenteeism sapmle Data (Columns 17-21)

Pet	Weight	Height	Body mass index	Absenteeism time in hours
1	90	172	30	4
0	98	178	31	0
0	89	170	31	2
0	68	168	24	4
1	90	172	30	2
0	89	170	31	NA
4	80	172	27	8

Chapter 2

Methodology

2.1 Pre Processing

Any model requires that we look at the data before we start modeling. However, in data mining terms looking at data refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is often called as Exploratory Data Analysis.

We can start looking by checking data types of imported data and then analysing it to check whether the data given is as per standards mentioned in the problem statement. After checking this, we see that all the variable are of continuous or numeric type. And some of the values in dataset are not following data standards mentioned in problem statement. For example, variables reason for absence and month of absence contains value of 0 for some observations, this can't be the case as it is clearly mentioned that reasons for absence are categorized from 1 to 28 types and from data it can be observed that month has values from 1 to 12. So we must change this values to proper type (only exception is where the target variable Absenteeism time in hours is also 0). Moreover, all the variable are of numeric type but when we check the actual unique values and range of the variable, it can be seen that many can be converted to factors like Reason for absence, Day of the week, Son, Social drinker, Education, Disciplinary failure etc. Check the code A.1. After making this exploratory data analysis changes, we can move ahead with data preprocessing techniques.

2.1.1 Missing Value Analysis

Missing values in any variable can adversely affect the accuracy of model and hamper the prediction result. So treating missing values before model development is very important. As our data contains missing values, we must do missing value analysis for data. First, check percentage of missing values for each variable. If missing value percentage is greater than 30%, we have to drop that column from model development. By doing this, we may lose precious information but even after imputing missing values for this variable, it will be biased because we have imputed it manually. But it is not the case for our dataset, so we can impute the missing values either by statistical imputation (mean or mode) or by KNN imputation method. But by closer analysis, it is found that few variable like Son, Social drinker, Pet, Education etc. depends on the variable ID, so we can impute this variable values as according to other value of same ID observation. By doing this, we will preserve the integrity of the data corresponding to ID variable, and other variable i.e. not dependent on ID variables can be imputed by the KNN imputation as per the code A.2

2.1.2 Outlier Analysis

The outliers are the values of variables which fall beyond the normal range of the variable values and considered as exception. So it is better to remove them to make data normally distributed. But it is not the case always, sometimes outliers are telling something about the target variable. So we must check this before processing of the outliers. Now in our case, some of the variables are containing outliers. The figure 2.1 shows the boxplots of all the numeric variables and figure 2.2 shows the histogram of the numeric variable. But we will not process the outliers of variable depending on ID to preserve the data integrity. But others can be processed using boxplot method. Check the code A.3 for outlier analysis.

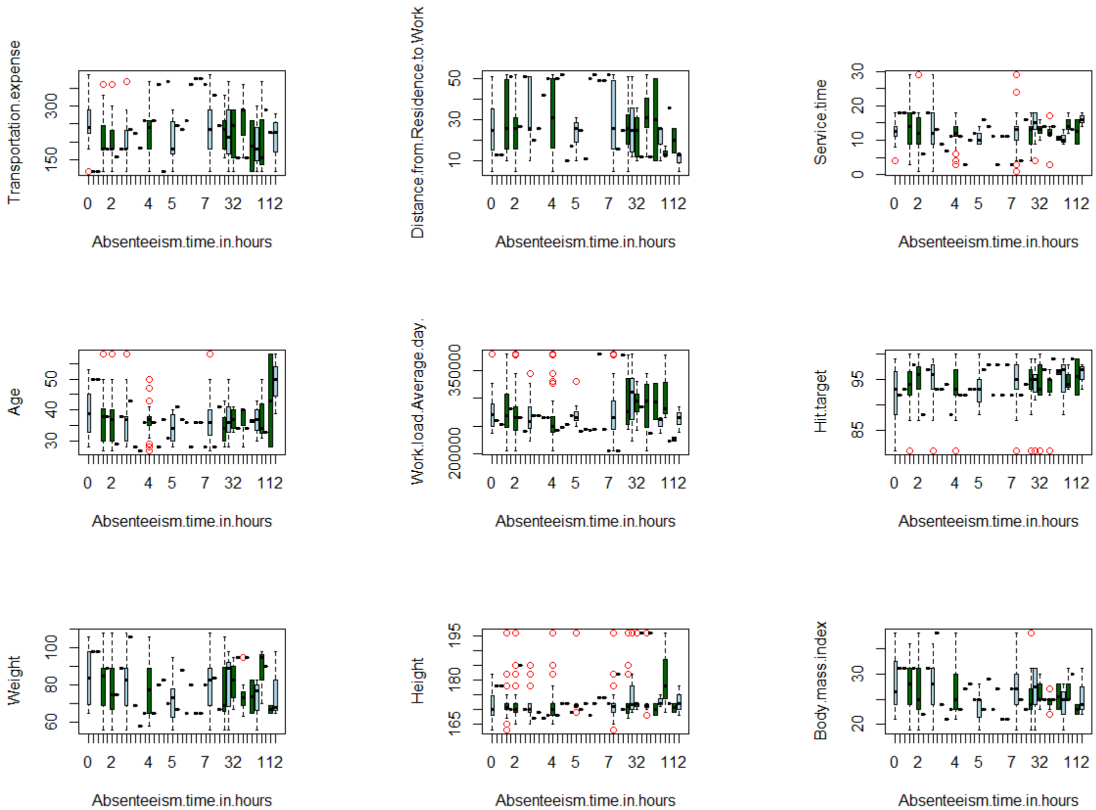

Figure 2.1: Boxplot of Variables with Outliers

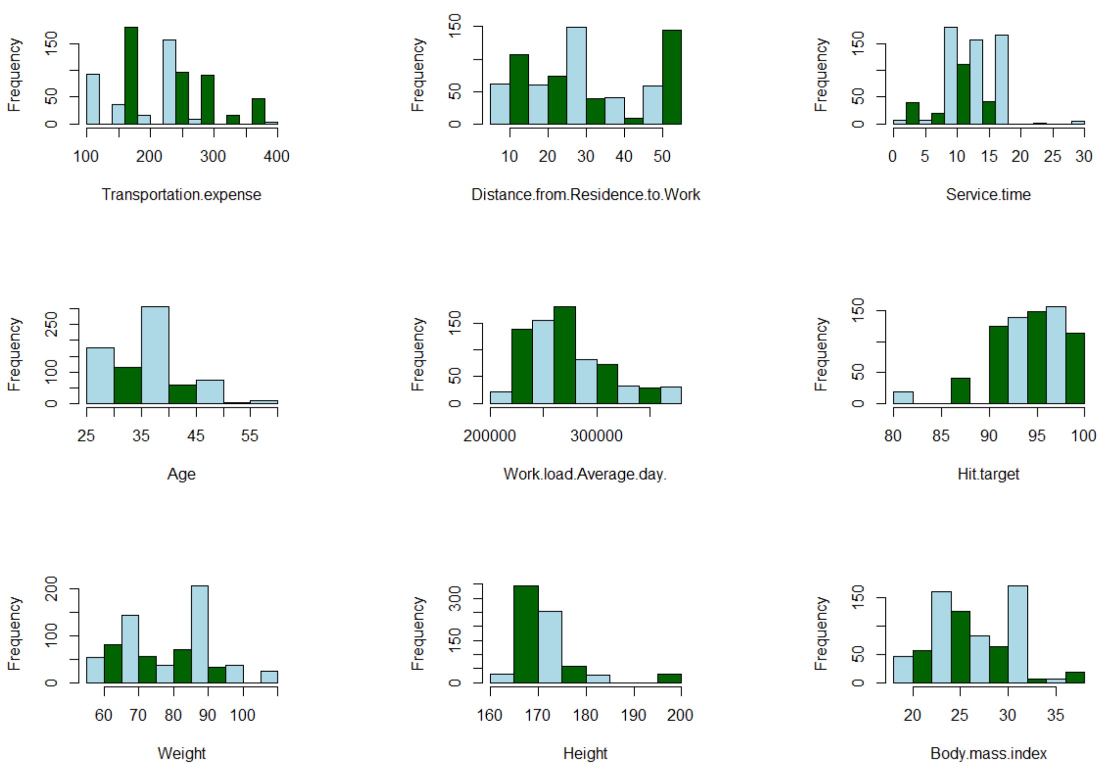

Figure 2.2: Histogram of Variables with Outliers

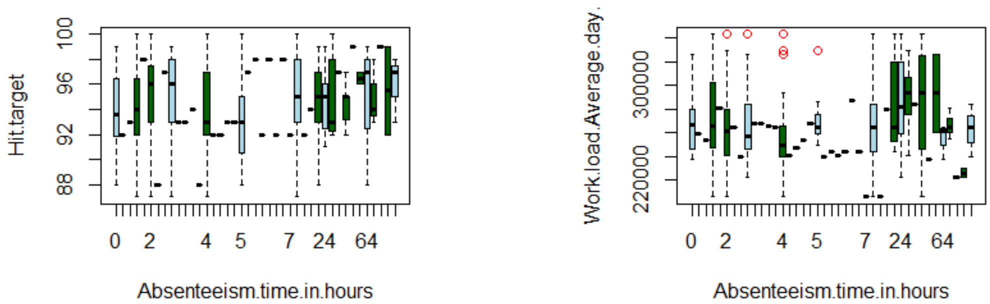

Figure 2.3: Boxplot of Variables without Outliers

2.1.3 Feature Selection

Before performing any type of modeling we need to assess the importance of each predictor variable in our analysis. There is a possibility that many variables in our analysis are not important at all to the problem of classification. There are several methods of doing that. We have used the correlation analysis to check collinearity between the variables and anova test to check dependence of target variable on the independent variables. The code A.4 shows the implementation of this methods.

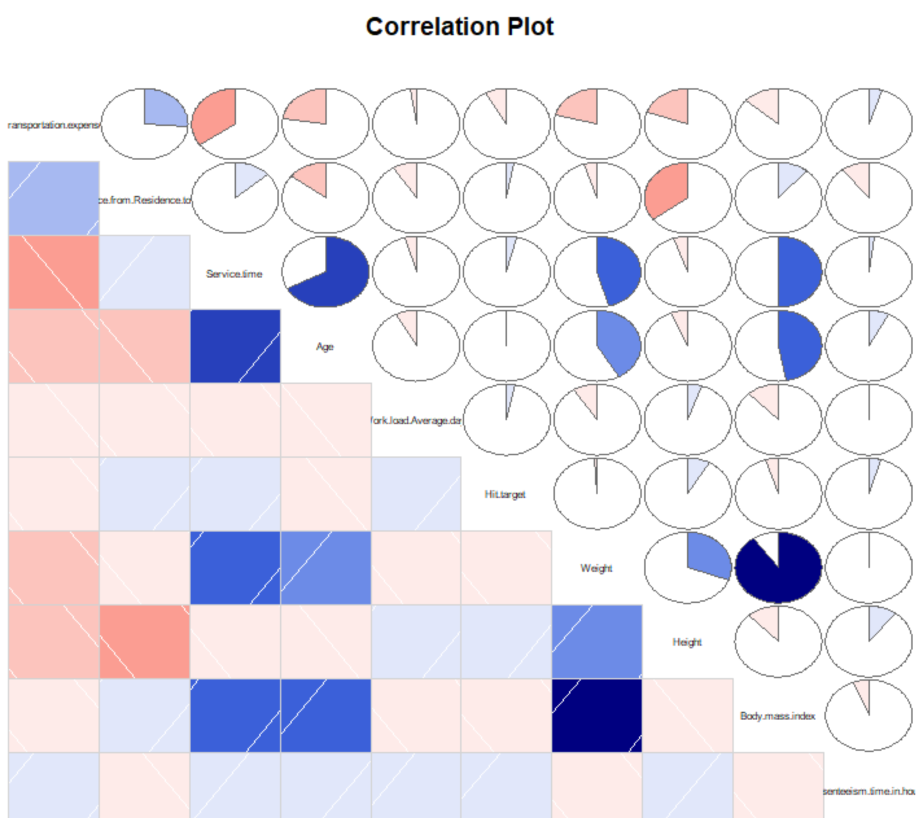

Figure 2.4: Correlation Plot for Employee Absenteeism Data

As it can be seen from the figure 2.4 that Weight and Body Mass Index are correlated with each other. So to tackle problem of multi collinearity, we can remove the variable Weight or Body Mass Index. Also with Anova technique we can find the actual variable dependence of each variable with target variable.

Following are the results of ANOVA method:

```
[1] "ID"
                    Df Sum Sq Mean Sq F value   Pr(>F)    
EmployeeData[, i]   35  13122   374.9    2.23 7.91e-05 ***
Residuals          704 118353   168.1                     
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
[1] "Reason.for.absence"
                    Df Sum Sq Mean Sq F value Pr(>F)    
EmployeeData[, i]   27  30219  1119.2    7.87 <2e-16 ***
Residuals          712 101256   142.2                   
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
[1] "Month.of.absence"
                    Df Sum Sq Mean Sq F value Pr(>F)  
EmployeeData[, i]   12   3370   280.9   1.594 0.0883 .
Residuals          727 128105   176.2                 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
[1] "Day.of.the.week"
                    Df Sum Sq Mean Sq F value Pr(>F)  
EmployeeData[, i]    4   2214   553.6   3.148  0.014 *
Residuals          735 129261   175.9                 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
[1] "Seasons"
                    Df Sum Sq Mean Sq F value Pr(>F)
EmployeeData[, i]    3    537   178.9   1.006   0.39
Residuals          736 130939   177.9               
[1] "Transportation.expense"
                    Df Sum Sq Mean Sq F value Pr(>F)
EmployeeData[, i]    1    300   300.1   1.688  0.194
Residuals          738 131175   177.7               
[1] "Distance.from.Residence.to.Work"
                    Df Sum Sq Mean Sq F value  Pr(>F)   
EmployeeData[, i]    1   1434  1434.1   8.138 0.00445 **
Residuals          738 130041   176.2                   
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
[1] "Service.time"
                    Df Sum Sq Mean Sq F value Pr(>F)
EmployeeData[, i]    1     42   42.04   0.236  0.627
Residuals          738 131433  178.09               
[1] "Age"
                    Df Sum Sq Mean Sq F value Pr(>F)  
EmployeeData[, i]    1    737   736.6   4.158 0.0418 *
Residuals          738 130739   177.2                 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
[1] "Work.load.Average.day."
                    Df Sum Sq Mean Sq F value Pr(>F)
EmployeeData[, i]    1      9    8.55   0.048  0.827
Residuals          738 131467  178.14               
[1] "Hit.target"
                    Df Sum Sq Mean Sq F value Pr(>F)
EmployeeData[, i]    1    248   248.4   1.397  0.238
Residuals          738 131227   177.8               
```

```
[1] "Disciplinary.failure"
                  Df Sum Sq Mean Sq F value Pr(>F)
EmployeeData[, i]   1    469   469.1   2.643  0.104
Residuals         738 131006   177.5
[1] "Education"
                  Df Sum Sq Mean Sq F value Pr(>F)
EmployeeData[, i]   3    273   91.06   0.511  0.675
Residuals         736 131202  178.26
[1] "Son"
                  Df Sum Sq Mean Sq F value   Pr(>F)
EmployeeData[, i]   4   3766   941.6   5.419 0.000265 ***
Residuals         735 127709   173.8
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
[1] "Social.drinker"
                  Df Sum Sq Mean Sq F value Pr(>F)
EmployeeData[, i]   1    551   551.1   3.106 0.0784 .
Residuals         738 130924   177.4
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
[1] "Social.smoker"
                  Df Sum Sq Mean Sq F value Pr(>F)
EmployeeData[, i]   1    214   213.9   1.202  0.273
Residuals         738 131262   177.9
[1] "Pet"
                  Df Sum Sq Mean Sq F value Pr(>F)
EmployeeData[, i]   5   1171   234.1   1.319  0.254
Residuals         734 130305   177.5
[1] "Weight"
                  Df Sum Sq Mean Sq F value Pr(>F)
EmployeeData[, i]   1      5    4.83   0.027  0.869
Residuals         738 131471  178.14
[1] "Height"
                  Df Sum Sq Mean Sq F value  Pr(>F)
EmployeeData[, i]   1   1460  1460.2   8.288 0.00411 **
Residuals         738 130015   176.2
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
[1] "Body.mass.index"
                  Df Sum Sq Mean Sq F value Pr(>F)
EmployeeData[, i]   1    420   419.5   2.363  0.125
Residuals         738 131056   177.6
[1] "Absenteeism.time.in.hours"
                  Df Sum Sq Mean Sq   F value Pr(>F)
EmployeeData[, i]   1 131475  131475 1.593e+33 <2e-16 ***
Residuals         738      0       0
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

We can analyse from the importance of variable from p-value result of ANOVA. Smaller the value of p higher the importance of variable. The threshold value of p is 0.05 i.e. if value of p is less than 0.05, we select the variable for model development. By this analysis, we can drop the variables Weight, Education, Service time, Work load Average day, Seasons, Social smoker, Pet, Hit target, Transportation expense, Body mass index, Disciplinary failure, Month of absence, Social drinker as all this have p-value greater than 0.05.

2.1.4 Feature Scaling

From figure 2.2, it is clear that the range of all the variables is not same. Some variables range of hundreds while other have range of thousands. We need to normalise this, so that model should not be more prone towards the high value variables. We can do this either by standardization or normalization. Standardization is more suited for the data which is normally distributed. As for Churn Reduction data is not normally distributed. So we can't use standardization technique, Normalization is more suitable for such data set. After normalization all numeric data values will be between 0 and 1. Fig 2.7 shows the variables after normalization which have range 0 to 1.

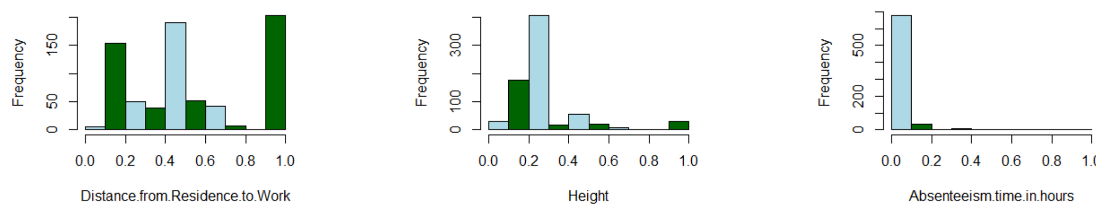

Figure 2.7: Variable Histogram After Normalization

2.2 Modeling

2.2.1 Model Selection

After pre-processing of data, we must proceed with model development. For Employee Absenteeism Project, we want to find what changes company should make to reduce the Absenteeism problem and also what are the expected loss in the year 2011 per month if same trend continues? So we need find the importance of each variable with respect to target variable to suggest the changes for company and predict the result of next year for same data to calculate the losses of company due to absenteeism. For doing this, we can use following regression models:

1. Multiple Linear Regression
2. Decision Tree Regression
3. Random Forest Regression
4. Support Vector Regressor

2.2.2 Multiple Linear Regression

```
Call:
lm(formula = Absenteeism.time.in.hours ~ ., data = train)

Residuals:
     Min       1Q   Median       3Q      Max
-0.43700 -0.02819 -0.00263  0.01563  0.83796

Coefficients: (5 not defined because of singularities)
                               Estimate Std. Error t value Pr(>|t|)
(Intercept)                   -0.005243   0.082679  -0.063 0.949461
ID2                            0.058272   0.216299   0.269 0.787724
ID3                            0.083731   0.461694   0.181 0.856158
ID5                            0.031010   0.107595   0.288 0.773301
ID6                            0.055149   0.212511   0.260 0.795341
ID7                            0.178208   0.086935   2.050 0.040872 *
ID8                            0.067156   0.282830   0.237 0.812405
ID9                            0.219089   0.057788   3.791 0.000167 ***
ID10                           0.095922   0.474607   0.202 0.839911
ID11                           0.054857   0.290085   0.189 0.850082
ID12                           0.005371   0.464204   0.012 0.990772
ID13                           0.086755   0.078173   1.110 0.267604
ID14                           0.075323   0.033453   2.252 0.024759 *
ID15                           0.077474   0.232861   0.333 0.739492
ID16                           0.070924   0.113336   0.626 0.531727
ID17                           0.056388   0.132103   0.427 0.669665
ID18                           0.048716   0.068101   0.715 0.474707
ID19                           0.079736   0.452472   0.176 0.860187
ID20                           0.104382   0.450317   0.232 0.816786
ID21                           0.028534   0.064064   0.445 0.656214
ID22                           0.039999   0.175068   0.228 0.819363
ID23                           0.080734   0.439610   0.184 0.854360
ID24                           0.053498   0.164535   0.325 0.745198
ID25                          -0.001171   0.072103  -0.016 0.987048
ID26                           0.126040   0.182588   0.690 0.490311
ID27                           0.043486   0.360624   0.121 0.904065
ID28                           0.034292   0.174652   0.196 0.844418
ID29                           0.044794   0.108498   0.413 0.679883
ID30                           0.031530   0.192475   0.164 0.869943
ID31                           0.014728   0.077627   0.190 0.849593
ID32                           0.078272   0.430413   0.182 0.855768
ID33                           0.025961   0.164619   0.158 0.874751
ID34                           0.011488   0.030759   0.373 0.708939
ID35                           0.094269   0.401725   0.235 0.814563
ID36                           0.058502   0.037658   1.554 0.120902
Reason.for.absence2            0.087492   0.037307   2.345 0.019390 *
Reason.for.absence3            0.037856   0.051445   0.736 0.462157
Reason.for.absence4            0.064514   0.077528   0.832 0.405706
Reason.for.absence5            0.058201   0.058362   0.997 0.319112
```

```
Reason.for.absence6             0.035389   0.076026   0.465 0.641781
Reason.for.absence7             0.281759   0.046785   6.022 3.23e-09 ***
Reason.for.absence8             0.045941   0.039530   1.162 0.245686
Reason.for.absence9             0.045814   0.053646   0.854 0.393495
Reason.for.absence10            0.507276   0.076120   6.664 6.74e-11 ***
Reason.for.absence11            0.093428   0.032596   2.866 0.004320 **
Reason.for.absence12            0.084711   0.032557   2.602 0.009531 **
Reason.for.absence13            0.211579   0.049462   4.278 2.25e-05 ***
Reason.for.absence14            0.134237   0.028603   4.693 3.44e-06 ***
Reason.for.absence15            0.071306   0.036019   1.980 0.048260 *
Reason.for.absence16            0.065971   0.104334   0.632 0.527464
Reason.for.absence17            0.017815   0.068226   0.261 0.794102
Reason.for.absence18            0.047978   0.108272   0.443 0.657857
Reason.for.absence19            0.071434   0.036946   1.933 0.053715 .
Reason.for.absence20            0.158203   0.030458   5.194 2.95e-07 ***
Reason.for.absence21            0.082518   0.058561   1.409 0.159398
Reason.for.absence22            0.067577   0.032717   2.065 0.039368 *
Reason.for.absence23            0.030204   0.026250   1.151 0.250397
Reason.for.absence24            0.079045   0.066561   1.188 0.235545
Reason.for.absence25            0.023807   0.033098   0.719 0.472285
Reason.for.absence26            0.067560   0.031072   2.174 0.030128 *
Reason.for.absence27            0.040778   0.029696   1.373 0.170282
Reason.for.absence28            0.030236   0.027239   1.110 0.267494
Day.of.the.week2               -0.000439   0.013957  -0.031 0.974921
Day.of.the.week3               -0.010386   0.013515  -0.768 0.442568
Day.of.the.week4               -0.033292   0.014280  -2.331 0.020115 *
Day.of.the.week5               -0.013692   0.014485  -0.945 0.344966
Distance.from.Residence.to.Work -0.088518  0.542206  -0.163 0.870380
Son2                                  NA         NA      NA       NA
Son3                                  NA         NA      NA       NA
Son4                                  NA         NA      NA       NA
Son5                                  NA         NA      NA       NA
Height                                NA         NA      NA       NA
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1002 on 525 degrees of freedom
Multiple R-squared:  0.345,	Adjusted R-squared:  0.2627
F-statistic:  4.19 on 66 and 525 DF,  p-value: < 2.2e-16
```

Figure 2.9: Multiple Linear Regression

As you can see the Adjusted R-squared value, we can explain only about 25% of the data using our multiple linear regression model. This is not very impressive.

2.2.3 Decision Tree

As with implementation of Decision Tree for regression, we get the importance for each variable for predicting target variable. It is clearly visible from figure 2.10

```
Variable importance
             Reason.for.absence                              ID
                             42                              30
                         Height Distance.from.Residence.to.Work
                              9                               8
                            Son                 Day.of.the.week
                              6                               4
```

Figure 2.10: Decision Trees Summary

Following is the glimpse of the Decision Tree:

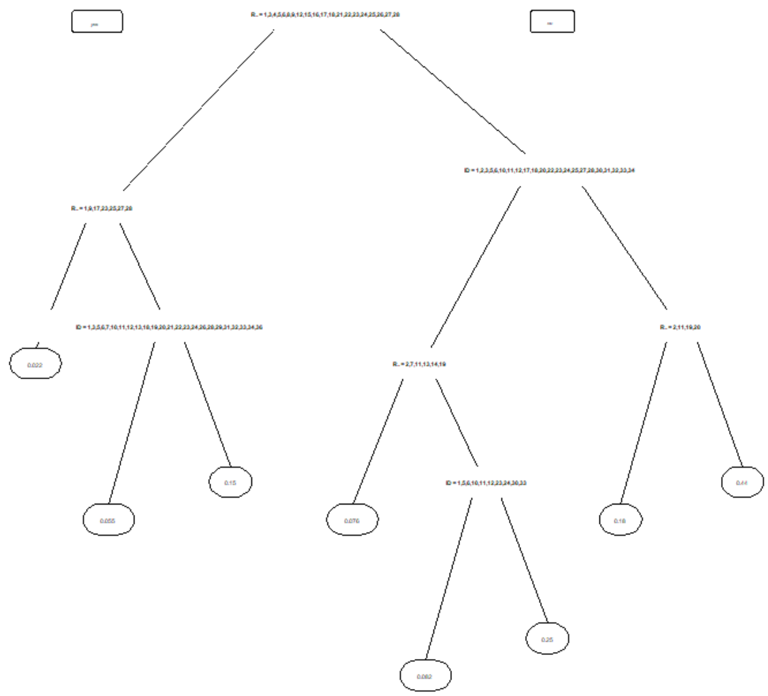

Figure 2.11: Decision Trees

2.2.4 Random Forest

Glimpse of first tree of Random Forest Tree:

```
> getTree(RF_regressor, 1, labelVar=TRUE)
   left daughter right daughter                 split var  split point status  prediction
1              2              3        Reason.for.absence 2.684349e+08     -3 0.059310308
2              4              5                        ID 2.576029e+10     -3 0.051502889
3              6              7                        ID 6.858526e+10     -3 0.471684008
4              8              9                        ID 2.441113e+10     -3 0.036073325
5             10             11                        ID 6.871947e+10     -3 0.094665327
6             12             13           Day.of.the.week 4.000000e+00     -3 0.053642903
7              0              0                      <NA> 0.000000e+00     -1 0.973333333
8             14             15                        ID 2.149188e+10     -3 0.026536496
9             16             17                        ID 1.342702e+09     -3 0.053390200
10            18             19                        ID 3.435967e+10     -3 0.084589938
11            20             21           Day.of.the.week 2.400000e+01     -3 0.161666667
12             0              0                      <NA> 0.000000e+00     -1 0.027595377
13             0              0                      <NA> 0.000000e+00     -1 0.066666667
14            22             23        Reason.for.absence 1.000000e+00     -3 0.015957447
15            24             25           Day.of.the.week 1.600000e+01     -3 0.028707741
16            26             27                    Height 5.151515e-01     -3 0.046311621
17            28             29        Reason.for.absence 2.684150e+08     -3 0.056757679
18            30             31                        ID 6.012115e+10     -3 0.077147444
19            32             33           Day.of.the.week 2.700000e+01     -3 0.096556301
20             0              0                      <NA> 0.000000e+00     -1 0.012500000
21             0              0                      <NA> 0.000000e+00     -1 0.178240741
22            34             35                    Height 1.969697e-01     -3 0.000000000
23            36             37                    Height 1.212121e-01     -3 0.019736842
24            38             39                    Height 2.575758e-01     -3 0.022114791
```

Figure 2.12: Random Forest Tree

Chapter 3

Conclusion

3.1 Model Evaluation

3.1.1 Accuracy

After model development, it is important to check its accuaracy. For time series data, the accuracy matrix used are MSE(Mean Square Error) or RMSE(Root Mean Square Error). Following are the results of all the implemented model with RMSE and MSE.

```
> RMSE(lm_predict,y_test)
[1] 0.1538157
> MSE(lm_predict,y_test)
[1] 0.02365928
```

(a) Multiple Linear Regression

```
> RMSE(DT_predict,y_test)
[1] 0.1578754
> MSE(DT_predict,y_test)
[1] 0.02492463
```

(b) Decision Tree

```
> RMSE(RF_predict,y_test)
[1] 0.1394911
> MSE(RF_predict,y_test)
[1] 0.01945776
```

(c) Random Forest

```
> RMSE(SVR_predict,y_test)
[1] 0.1467268
> MSE(SVR_predict,y_test)
[1] 0.02152876
```

(d) Support Vector Regressor

3.2 Model Selection

As it can be clearly seen from the MSE or RMSE result, the Random Forest Regression is performing best for this Employee Absenteeism Dataset. So we can freeze the Random Forest model for the predictions of this problem.

3.3 Solution

3.3.1 Problem 1

What changes company should bring to reduce the number of absenteeism?

1. Company should take some disciplinary action against employees with ID3, ID9 and ID11(As per Figure 3.2).

3	452.15946
11	450.00000
14	394.36434
28	348.29539
34	345.56542
36	312.92456

(a) Mean Absenteeism time

9	32.750000
7	25.000000
26	16.600000
14	13.598770
13	11.985075
11	11.250000

(b) Sum Absenteeism time

Figure 3.2: Absenteeism time in hours

2. Moreover, the absenteeism issue of the employees is mostly due to following health conditions (Check Figure 3.4):

   - Diseases of the nervous system
   - Diseases of the eye and adnexa
   - Diseases of the circulatory system
   - Diseases of the respiratory system
   - Diseases of the digestive system
   - Diseases of the skin and subcutaneous tissue
   - Diseases of the musculoskeletal system and connective tissue
   - Diseases of the genitourinary system
   - Injury, poisoning and certain other consequences of external causes

   Company should take some preventive measure to avoid employment with this disease conditions or check the validity of claim made by the employee to reduce the absenteeism issue.

```
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.0356621  0.0711724  -0.501 0.616488
ID2          -0.0186026  0.1867828  -0.100 0.920696
ID3          -0.0898859  0.4061844  -0.221 0.824931
ID4           0.0121530  0.1052505   0.115 0.908109
ID5          -0.0039544  0.0956226  -0.041 0.967026
ID6          -0.0181420  0.1869857  -0.097 0.922737
ID7           0.1793373  0.0766512   2.340 0.019593 *
ID8          -0.0344914  0.2514416  -0.137 0.890934
ID9           0.1691480  0.0520967   3.247 0.001225 **
ID10         -0.0861745  0.4170390  -0.207 0.836358
ID11         -0.0300305  0.2548513  -0.118 0.906233
ID12         -0.1571458  0.4083505  -0.385 0.700485
ID13          0.0459059  0.0687363   0.668 0.504456
ID14          0.0592150  0.0297209   1.992 0.046735 *
ID15         -0.0047132  0.2045883  -0.023 0.981627
ID16          0.0267053  0.0841813   0.317 0.751164
ID17         -0.0002437  0.1161563  -0.002 0.998327
ID18          0.0158129  0.0598147   0.264 0.791580
```

```
ID19                            -0.0885264  0.3991376  -0.222 0.824541
ID20                            -0.0665031  0.3964486  -0.168 0.866833
ID21                             0.0193335  0.0613691   0.315 0.752831
ID22                            -0.0315898  0.1541951  -0.205 0.837736
ID23                            -0.0802030  0.3870856  -0.207 0.835919
ID24                            -0.0300290  0.1448086  -0.207 0.835784
ID25                            -0.0180105  0.0634883  -0.284 0.776741
ID26                             0.0512237  0.1609089   0.318 0.750326
ID27                            -0.0858640  0.3173641  -0.271 0.786818
ID28                            -0.0302109  0.1539525  -0.196 0.844485
ID29                            -0.0157274  0.0795427  -0.198 0.843322
ID30                            -0.0581995  0.1681144  -0.346 0.729308
ID31                            -0.0010095  0.0727297  -0.014 0.988930
ID32                            -0.0862818  0.3787226  -0.228 0.819853
ID33                            -0.0348649  0.1448297  -0.241 0.809837
ID34                             0.0135753  0.0273468   0.496 0.619767
ID35                            -0.0507841  0.3565977  -0.142 0.886797
ID36                             0.0421093  0.0337494   1.248 0.212573
Reason.for.absence2              0.0920914  0.0313651   2.936 0.003437 **
Reason.for.absence3              0.0458497  0.0480707   0.954 0.340530
Reason.for.absence4              0.0604694  0.0619572   0.976 0.329422
Reason.for.absence5              0.0722792  0.0497026   1.454 0.146349
Reason.for.absence6              0.0576999  0.0599309   0.963 0.336007
Reason.for.absence7              0.2241533  0.0394153   5.687 1.93e-08 ***
Reason.for.absence8              0.0827095  0.0325277   2.543 0.011222 *
Reason.for.absence9              0.0561546  0.0458394   1.225 0.220993
Reason.for.absence10             0.3657169  0.0534103   6.847 1.70e-11 ***
Reason.for.absence11             0.1005720  0.0278913   3.606 0.000334 ***
Reason.for.absence12             0.0843633  0.0274660   3.072 0.002215 **
Reason.for.absence13             0.1914501  0.0404433   4.734 2.69e-06 ***
Reason.for.absence14             0.1335383  0.0232616   5.741 1.43e-08 ***
Reason.for.absence15             0.0692031  0.0301567   2.295 0.022053 *
Reason.for.absence16             0.0349329  0.0758245   0.461 0.645158
Reason.for.absence17             0.0246108  0.0634171   0.388 0.698081
Reason.for.absence18             0.0594864  0.1030202   0.577 0.563846
Reason.for.absence19             0.0762491  0.0292347   2.608 0.009305 **
Reason.for.absence20             0.1660448  0.0249614   6.652 5.99e-11 ***
Reason.for.absence21             0.0692014  0.0455732   1.518 0.129367
Reason.for.absence22             0.0734363  0.0261955   2.803 0.005203 **
Reason.for.absence23             0.0356055  0.0208215   1.710 0.087721 .
Reason.for.absence24             0.0586720  0.0615296   0.954 0.340651
Reason.for.absence25             0.0364916  0.0273031   1.337 0.181827
Reason.for.absence26             0.0643311  0.0254028   2.532 0.011554 *
Reason.for.absence27             0.0426055  0.0239525   1.779 0.075733 .
Reason.for.absence28             0.0336732  0.0216097   1.558 0.119646
Day.of.the.week2                 0.0014850  0.0116739   0.127 0.898815
Day.of.the.week3                -0.0080052  0.0115188  -0.695 0.487319
Day.of.the.week4                -0.0288019  0.0122677  -2.348 0.019174 *
Day.of.the.week5                -0.0114386  0.0120578  -0.949 0.343142
Distance.from.Residence.to.Work  0.1150146  0.4766667   0.241 0.809404
Age                                     NA         NA      NA       NA
Son2                                    NA         NA      NA       NA
Son3                                    NA         NA      NA       NA
Son4                                    NA         NA      NA       NA
Son5                                    NA         NA      NA       NA
Height                                  NA         NA      NA       NA
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.09756 on 672 degrees of freedom
Multiple R-squared:  0.2994,	Adjusted R-squared:  0.2296 
F-statistic: 4.286 on 67 and 672 DF,  p-value: < 2.2e-16
```

Figure 3.4: Multiple Linear Regression Summary

3.3.2 Problem 2

How much losses every month can we project in 2011 if same trend of absenteeism continues?

To calculate the loss incurred by company due to absenteeism. We need to consider few parameters as nothing is mentioned about loss in the problem statement. So, I have assumed following things for calculation of loss:

1. Company incurred the average loss of 1000 rupees per hour for employee absenteeism.

2. If the employee is completed the target, then no loss is incurred i.e. if Hit Target = 100, then loss = 0.

3. The loss depends on the Disciplinary failure by factor of 2000.

4. The loss also depends on the Age and Education of employees i.e. if employee with higher education and more Age (experience) will be more important to the company and will have more impact on company earning.

   The implementation A.6 of the loss calculating function.

Appendix A

R Code

A.1 Exploratory Data Analysis

```
for (i in (1:nrow(EmployeeData))){
  if (EmployeeData$Absenteeism.time.in.hours[i] != 0 ||
            is.na(EmployeeData$Absenteeism.time.in.hours[i])){
    if(EmployeeData$Reason.for.absence[i] == 0 ||
            is.na(EmployeeData$Reason.for.absence[i])){
      EmployeeData$Reason.for.absence[i] = NA
    }
    if(EmployeeData$Month.of.absence[i] == 0 ||
            is.na(EmployeeData$Month.of.absence[i])){
      EmployeeData$Month.of.absence[i] = NA
    }
  }
}

EmployeeData$ID <- as.factor(EmployeeData$ID)
EmployeeData$Reason.for.absence <- as.factor(EmployeeData$Reason.for.absence)
EmployeeData$Month.of.absence <- as.factor(EmployeeData$Month.of.absence)
EmployeeData$Day.of.the.week <- as.factor(EmployeeData$Day.of.the.week)
EmployeeData$Seasons <- as.factor(EmployeeData$Seasons)
EmployeeData$Disciplinary.failure <- as.factor(EmployeeData$Disciplinary.failure)
EmployeeData$Education <- as.factor(EmployeeData$Education)
EmployeeData$Son <- as.factor(EmployeeData$Son)
EmployeeData$Social.drinker <- as.factor(EmployeeData$Social.drinker)
EmployeeData$Social.smoker <- as.factor(EmployeeData$Social.smoker)
EmployeeData$Pet <- as.factor(EmployeeData$Pet)
```

A.2 Missing Value Analysis

```
# Get Missing Values for all Variables
missingValueCheck <- function(data){
  for (i in colnames(data)){
    print(i)
    print(sum(is.na(data[i])))
  }
  print("Total")
  print(sum(is.na(EmployeeData)))
}
missingValueCheck(EmployeeData)

# Impute values related to ID
Depenedent_ID <- c("ID","Transportation.expense","Service.time","Age","Height",
```

```
                         "Distance.from.Residence.to.Work","Education","Son","Weight",
                         "Social.smoker","Social.drinker","Pet","Body.mass.index")
Depenedent_ID_data <- EmployeeData[,Depenedent_ID]
Depenedent_ID_data <- aggregate(. ~ ID, data = Depenedent_ID_data,
                          FUN = function(e) c(x = mean(e)))
for (i in Depenedent_ID) {
  for (j in (1:nrow(EmployeeData))){
    ID = EmployeeData[j,"ID"]
    if(is.na(EmployeeData[j,i])){
      EmployeeData[j,i] = Depenedent_ID_data[ID,i]
    }
  }
}
# Impute values for other variables
EmployeeData = knnImputation(EmployeeData, k = 7)
missingValueCheck(EmployeeData)
```

A.3 Outlier Analysis

```
# Histogram
for(i in num_cnames){
  hist(EmployeeData[,i], xlab=i, main=" ", col=(c("lightblue","darkgreen")))
}

# BoxPlot
num_cnames <- num_cnames[ num_cnames != "Absenteeism.time.in.hours"]
for(i in num_cnames){
  boxplot(EmployeeData[,i]~EmployeeData$Absenteeism.time.in.hours,
          data=EmployeeData, main=" ", ylab=i, xlab="Absenteeism.time.in.hours",
          col=(c("lightblue","darkgreen")), outcol="red")
}

# Removing ID related varaibles from num_cnames
for (i in Depenedent_ID){
  num_cnames <- num_cnames[ num_cnames != i]
}
# Replace all outliers with NA and impute
for(i in num_cnames){
  val = EmployeeData[,i][EmployeeData[,i] \%in\% boxplot.stats(EmployeeData[,i])$out]
  EmployeeData[,i][EmployeeData[,i] \%in\% val] = NA
}

# Impute Missing Values
missingValueCheck(EmployeeData)
EmployeeData = knnImputation(EmployeeData, k = 7)
```

A.4 Feature Selection

```
# Correlation Plot
corrgram(EmployeeData, upper.panel=panel.pie, text.panel=panel.txt,
         main = "Correlation Plot")

# ANOVA
cnames <- colnames(EmployeeData)
for (i in cnames){
  print(i)
  print(summary(aov(EmployeeData$Absenteeism.time.in.hours~EmployeeData[,i],
                    EmployeeData)))
}
```

```r
# Dimensionality Reduction
EmployeeData <- subset(EmployeeData, select = -c(Weight,Education,Service.time,
                                                    Social.smoker,Body.mass.index,
                                                    Work.load.Average.day.,Seasons,
                                                    Transportation.expense,Pet,
                                                    Disciplinary.failure,Hit.target,
                                                    Month.of.absence,Social.drinker))
```

A.5 Model Selection

```r
###################### Multiple Linear Regression ########################
num_data <- train[sapply(train, is.numeric)]
 vifcor(num_data, th=0.9)

lm_regressor <- lm(Absenteeism.time.in.hours~.,data = train)
summary(lm_regressor)
#Predict for new test cases
for (i in cat_cnames){
  lm_regressor$xlevels[[i]] <- union(lm_regressor$xlevels[[i]],
                                     levels(X_test[[i]]))
}
lm_predict = predict(lm_regressor,newdata=X_test)
RMSE(lm_predict,y_test)
MSE(lm_predict,y_test)

#########################Decision Trees for regression ###################
DT_regressor = rpart(Absenteeism.time.in.hours~.,data = train, method="anova")
#Predict for new test cases
DT_predict = predict(DT_regressor, X_test)
RMSE(DT_predict,y_test)
MSE(DT_predict,y_test)

##################### Random Forest ###########################
RF_regressor = randomForest(x = X_train, y = y_train, ntree = 100)
#Predict for new test cases
RF_predict = predict(RF_regressor, X_test)
RMSE(RF_predict,y_test)
MSE(RF_predict,y_test)

############################ SVR ###########################
SVR_regressor = svm(formula = Absenteeism.time.in.hours ~ .,
                 data = train, type = 'eps-regression')
#Predict for new test cases
SVR_predict = predict(SVR_regressor, X_test)
RMSE(SVR_predict,y_test)
MSE(SVR_predict,y_test)
```

A.6 Conclusion

```r
# Calculating Losses
p2_data = EmployeeData[,-8]
#Predict for new test cases
p2_predict = predict(RF_regressor, p2_data)

# Convert predict values back to original scale
p2_predict <- (p2_predict * 120)

# Add predicted values to the DataSet
p2_dataSet <- merge(DataSet,p2_predict,by="row.names",all.x=TRUE)
```

```
# Calculate the total Loss
Loss = 0
Loss <- 0
for (i in 1:nrow(p2_dataSet)){
  if (p2_dataSet$Hit.target[i] != 100)
    if (p2_dataSet$Age[i] >= 25 &&  p2_dataSet$Age[i] <= 32){
      Loss = Loss + as.numeric(p2_dataSet$Disciplinary.failure[i]) * 2000 +
        (as.numeric(p2_dataSet$Education[i]) + 1) * 500 + p2_dataSet$y[i] * 1000
    }else if(p2_dataSet$Age[i] >= 33 &&  p2_dataSet$Age[i] <= 40){
      Loss = Loss + as.numeric(p2_dataSet$Disciplinary.failure[i]) * 2000 +
        (as.numeric(p2_dataSet$Education[i]) + 2) * 500 + p2_dataSet$y[i] * 1000
    }else if(p2_dataSet$Age[i] >= 41 &&  p2_dataSet$Age[i] <= 49){
      Loss = Loss + as.numeric(p2_dataSet$Disciplinary.failure[i]) * 2000 +
        (as.numeric(p2_dataSet$Education[i]) + 3) * 500 + p2_dataSet$y[i] * 1000
    }else if(p2_dataSet$Age[i] >= 50 &&  p2_dataSet$Age[i] <= 60){
      Loss = Loss + as.numeric(p2_dataSet$Disciplinary.failure[i]) * 2000 +
        (as.numeric(p2_dataSet$Education[i]) + 4) * 500 + p2_dataSet$y[i] * 1000
    }
}
# To calculate loss per month
Loss = Loss/12
```

A.7 Complete Code

```
# Clean current environment
rm(list=ls())

# Load require Packages
p <- c("xlsx","DMwR","corrgram","caret","usdm","rpart","DataCombine","randomForest",
       "e1071")
lapply(p, require, character.only=TRUE)
rm(p)

EmployeeData <- read.xlsx("Absenteeism_at_work_Project.xls", sheetIndex = 1)

###################### Exploratory Data Analysis ########################
# Check variable types
cnames <- colnames(EmployeeData)

# Check number of unique variables
for (i in cnames){
  print(i)
  print(aggregate(data.frame(count = EmployeeData[,i]),
                  list(value = EmployeeData[,i]), length))
}

# Data Preprocessing
preprocessing <- function(EmployeeData){
  EmployeeData$ID <- as.factor(EmployeeData$ID)
  for (i in (1:nrow(EmployeeData))){
    if (EmployeeData$Absenteeism.time.in.hours[i] != 0 ||
             is.na(EmployeeData$Absenteeism.time.in.hours[i])){
      if(EmployeeData$Reason.for.absence[i] == 0 ||
             is.na(EmployeeData$Reason.for.absence[i])){
        EmployeeData$Reason.for.absence[i] = NA
      }
      if(EmployeeData$Month.of.absence[i] == 0 ||
             is.na(EmployeeData$Month.of.absence[i])){
        EmployeeData$Month.of.absence[i] = NA
      }
```

```
    }
  }
  EmployeeData$Reason.for.absence <- as.factor(EmployeeData$Reason.for.absence)
  EmployeeData$Month.of.absence <- as.factor(EmployeeData$Month.of.absence)
  EmployeeData$Day.of.the.week <- as.factor(EmployeeData$Day.of.the.week)
  EmployeeData$Seasons <- as.factor(EmployeeData$Seasons)
  EmployeeData$Disciplinary.failure <- as.factor(EmployeeData$Disciplinary.failure)
  EmployeeData$Education <- as.factor(EmployeeData$Education)
  EmployeeData$Son <- as.factor(EmployeeData$Son)
  EmployeeData$Social.drinker <- as.factor(EmployeeData$Social.drinker)
  EmployeeData$Social.smoker <- as.factor(EmployeeData$Social.smoker)
  EmployeeData$Pet <- as.factor(EmployeeData$Pet)
  return(EmployeeData)
}
EmployeeData <- preprocessing(EmployeeData)

#selecting only factor
get_cat <- function(data) {
  return(colnames(data[,sapply(data, is.factor)]))
}
cat_cnames <- get_cat(EmployeeData)

#selecting only numeric
get_num <- function(data) {
  return(colnames(data[,sapply(data, is.numeric)]))
}
num_cnames <- get_num(EmployeeData)

# Covert factor varaible values to labels
for (i in cat_cnames){
  EmployeeData[,i] <- factor(EmployeeData[,i],
                             labels = (1:length(levels(factor(EmployeeData[,i])))))
}

################################# Data PreProcessing #########################
# Missing Value Analysis
# Get Missing Values for all Variables
missingValueCheck <- function(data){
  for (i in colnames(data)){
    print(i)
    print(sum(is.na(data[i])))
  }
  print("Total")
  print(sum(is.na(EmployeeData)))
}
missingValueCheck(EmployeeData)

# Impute values related to ID
Depenedent_ID <- c("ID","Transportation.expense","Service.time","Age","Height",
                   "Distance.from.Residence.to.Work","Education","Son","Weight",
                   "Social.smoker","Social.drinker","Pet","Body.mass.index")
Depenedent_ID_data <- EmployeeData[,Depenedent_ID]
Depenedent_ID_data <- aggregate(. ~ ID, data = Depenedent_ID_data,
                     FUN = function(e) c(x = mean(e)))
for (i in Depenedent_ID) {
  for (j in (1:nrow(EmployeeData))){
    ID = EmployeeData[j,"ID"]
    if(is.na(EmployeeData[j,i])){
      EmployeeData[j,i] = Depenedent_ID_data[ID,i]
    }
```

```
    }
}
# Impute values for other variables
EmployeeData = knnImputation(EmployeeData, k = 7)
missingValueCheck(EmployeeData)

# Outlier Analysis
# Histogram
for(i in num_cnames){
  hist(EmployeeData[,i], xlab=i, main=" ", col=(c("lightblue","darkgreen")))
}

# BoxPlot
num_cnames <- num_cnames[ num_cnames != "Absenteeism.time.in.hours"]
for(i in num_cnames){
  boxplot(EmployeeData[,i]~EmployeeData$Absenteeism.time.in.hours,
          data=EmployeeData, main=" ", ylab=i, xlab="Absenteeism.time.in.hours",
          col=(c("lightblue","darkgreen")), outcol="red")
}

# Removing ID related varaibles from num_cnames
for (i in Depenedent_ID){
  num_cnames <- num_cnames[ num_cnames != i]
}
# Replace all outliers with NA and impute
for(i in num_cnames){
  val = EmployeeData[,i][EmployeeData[,i] \%in\% boxplot.stats(EmployeeData[,i])$out]
  EmployeeData[,i][EmployeeData[,i] \%in\% val] = NA
}

# Impute Missing Values
missingValueCheck(EmployeeData)
EmployeeData = knnImputation(EmployeeData, k = 7)

# Copy Employee Data into DataSet for further analysis
DataSet <- EmployeeData

######################## Fature Selection ##########################
# Correlation Plot
corrgram(EmployeeData, upper.panel=panel.pie, text.panel=panel.txt,
         main = "Correlation Plot")

# ANOVA
cnames <- colnames(EmployeeData)
for (i in cnames){
  print(i)
  print(summary(aov(EmployeeData$Absenteeism.time.in.hours~EmployeeData[,i],
                    EmployeeData)))
}

# Dimensionality Reduction
EmployeeData <- subset(EmployeeData, select = -c(Weight,Education,Service.time,
                                                 Social.smoker,Body.mass.index,
                                                 Work.load.Average.day.,Seasons,
                                                 Transportation.expense,Pet,
                                                 Disciplinary.failure,Hit.target,
                                                 Month.of.absence,Social.drinker))

######################### Fature Normalization ########################
cat_cnames <- get_cat(EmployeeData)
```

```
num_cnames <- get_num(EmployeeData)

# Fature Scaling
for (i in num_cnames){
  EmployeeData[,i] <- (EmployeeData[,i] - min(EmployeeData[,i])) /
                      (max(EmployeeData[,i]) - min(EmployeeData[,i]))
}

############################## Model Development ##############################
# Data Divide
X_index = sample(1:nrow(EmployeeData), 0.8 * nrow(EmployeeData))
X_train = EmployeeData[X_index,-8]
X_test = EmployeeData[-X_index,-8]
y_train = EmployeeData[X_index,8]
y_test = EmployeeData[-X_index,8]

train = EmployeeData[X_index,]
test = EmployeeData[-X_index,]

#calculate RMSE
RMSE = function(y, yhat){
  sqrt(mean((y - yhat)^2))
}

#calculate MSE
MSE = function(y, yhat){
  (mean((y - yhat)^2))
}

############################## Multiple Linear Regression ##############################
num_data <- train[sapply(train, is.numeric)]
vifcor(num_data, th=0.9)

lm_regressor <- lm(Absenteeism.time.in.hours~.,data = train)
summary(lm_regressor)
#Predict for new test cases
for (i in cat_cnames){
  lm_regressor$xlevels[[i]] <- union(lm_regressor$xlevels[[i]],
                                     levels(X_test[[i]]))
}
lm_predict = predict(lm_regressor,newdata=X_test)
RMSE(lm_predict,y_test)
MSE(lm_predict,y_test)

#################################Decision Trees for regression ##################
DT_regressor = rpart(Absenteeism.time.in.hours~.,data = train, method="anova")
#Predict for new test cases
DT_predict = predict(DT_regressor, X_test)
RMSE(DT_predict,y_test)
MSE(DT_predict,y_test)

############################## Random Forest ########################
RF_regressor = randomForest(x = X_train, y = y_train, ntree = 100)
#Predict for new test cases
RF_predict = predict(RF_regressor, X_test)
RMSE(RF_predict,y_test)
MSE(RF_predict,y_test)

############################## SVR ##############################
SVR_regressor = svm(formula = Absenteeism.time.in.hours ~ .,
```

```
                data = train, type = 'eps-regression')
#Predict for new test cases
SVR_predict = predict(SVR_regressor, X_test)
RMSE(SVR_predict,y_test)
MSE(SVR_predict,y_test)

############################## Problems ##################################
# Suggesting the changes
lm_regressor_p1 <- lm(Absenteeism.time.in.hours~.,data = EmployeeData)
summary(lm_regressor_p1)

# Calculating Losses
p2_data = EmployeeData[,-8]
#Predict for new test cases
p2_predict = predict(RF_regressor, p2_data)

# Convert predict values back to original scale
p2_predict <- (p2_predict * 120)

# Add predicted values to the DataSet
p2_dataSet <- merge(DataSet,p2_predict,by="row.names",all.x=TRUE)

# Calculate the total Loss
Loss <- 0
Loss <- 0
for (i in 1:nrow(p2_dataSet)){
  if (p2_dataSet$Hit.target[i] != 100)
    if (p2_dataSet$Age[i] >= 25 &&  p2_dataSet$Age[i] <= 32){
      Loss = Loss + as.numeric(p2_dataSet$Disciplinary.failure[i]) * 2000 +
        (as.numeric(p2_dataSet$Education[i]) + 1) * 500 + p2_dataSet$y[i] * 1000
    }else if(p2_dataSet$Age[i] >= 33 &&  p2_dataSet$Age[i] <= 40){
      Loss = Loss + as.numeric(p2_dataSet$Disciplinary.failure[i]) * 2000 +
        (as.numeric(p2_dataSet$Education[i]) + 2) * 500 + p2_dataSet$y[i] * 1000
    }else if(p2_dataSet$Age[i] >= 41 &&  p2_dataSet$Age[i] <= 49){
      Loss = Loss + as.numeric(p2_dataSet$Disciplinary.failure[i]) * 2000 +
        (as.numeric(p2_dataSet$Education[i]) + 3) * 500 + p2_dataSet$y[i] * 1000
    }else if(p2_dataSet$Age[i] >= 50 &&  p2_dataSet$Age[i] <= 60){
      Loss = Loss + as.numeric(p2_dataSet$Disciplinary.failure[i]) * 2000 +
        (as.numeric(p2_dataSet$Education[i]) + 4) * 500 + p2_dataSet$y[i] * 1000
    }
}
# To calculate loss per month
Loss <- Loss/12
```

Appendix B

Python Code

B.1 Exploratory Data Analysis

```
EmployeeData["ID"] = EmployeeData["ID"].astype(str)
    EmployeeData["Reason for absence"] =
                        EmployeeData["Reason for absence"].astype(str)
    EmployeeData["Month of absence"] = EmployeeData["Month of absence"].astype(str)
    EmployeeData["Day of the week"] = EmployeeData["Day of the week"].astype(str)
    EmployeeData["Seasons"] = EmployeeData["Seasons"].astype(str)
    EmployeeData["Disciplinary failure"] =
                        EmployeeData["Disciplinary failure"].astype(str)
    EmployeeData["Education"] = EmployeeData["Education"].astype(str)
    EmployeeData["Son"] = EmployeeData["Son"].astype(str)
    EmployeeData["Social drinker"] = EmployeeData["Social drinker"].astype(str)
    EmployeeData["Social smoker"] = EmployeeData["Social smoker"].astype(str)
    EmployeeData["Pet"] = EmployeeData["Pet"].astype(str)

    # Change NaN string values back to NaN
    EmployeeData["ID"] = EmployeeData["ID"].replace("nan",np.nan)
    EmployeeData["Reason for absence"] =
                    EmployeeData["Reason for absence"].replace("nan",np.nan)
    EmployeeData["Month of absence"] =
                    EmployeeData["Month of absence"].replace("nan",np.nan)
    EmployeeData["Day of the week"] =
                    EmployeeData["Day of the week"].replace("nan",np.nan)
    EmployeeData["Seasons"] = EmployeeData["Seasons"].replace("nan",np.nan)
    EmployeeData["Disciplinary failure"] =
                    EmployeeData["Disciplinary failure"].replace("nan",np.nan)
    EmployeeData["Education"] = EmployeeData["Education"].replace("nan",np.nan)
    EmployeeData["Son"] = EmployeeData["Son"].replace("nan",np.nan)
    EmployeeData["Social drinker"] =
                    EmployeeData["Social drinker"].replace("nan",np.nan)
    EmployeeData["Social smoker"] =
                    EmployeeData["Social smoker"].replace("nan",np.nan)
    EmployeeData["Pet"] = EmployeeData["Pet"].replace("nan",np.nan)

    # Covert factor varaible values to labels
    for i in range(0, len(EmployeeData.columns)):
        if(EmployeeData.iloc[:,i].dtypes == 'object'):
            EmployeeData.iloc[:,i] = pd.Categorical(EmployeeData.iloc[:,i])
            EmployeeData.iloc[:,i] = EmployeeData.iloc[:,i].cat.codes
            EmployeeData.iloc[:,i] = EmployeeData.iloc[:,i].astype('object')

    # Convert -1 values back to NaN
    for i in cnames[0]:
```

```
        for j in range(0,rows):
            if EmployeeData.loc[j,i] == -1:
                EmployeeData.loc[j,i] = np.nan
```

B.2 Missing Value Analysis

```
def missingValueCheck(data):
    print(data.isna().sum())
missingValueCheck(EmployeeData)

# Impute values related to ID
Depenedent_ID =  ["ID","Transportation_expense","Service_time","Age","Height",
                  "Distance_from_Residence_to_Work","Education","Son","Weight",
                  "Social_smoker","Social_drinker","Pet","Body_mass_index"]
Depenedent_ID_data = EmployeeData[Depenedent_ID].copy()
Depenedent_ID_data = Depenedent_ID_data.groupby("ID").max()
Depenedent_ID.remove('ID')

for i in Depenedent_ID:
    for j in range(0,rows):
        RI = EmployeeData["ID"][j]
        if np.isnan(EmployeeData.loc[j,i]):
            EmployeeData[i][j] = Depenedent_ID_data.loc[RI,i]

# Impute values for other variables with KNN
EmployeeData = pd.DataFrame(KNN(k = 7).complete(EmployeeData),
                            columns = EmployeeData.columns)
EmployeeData = preprocessing(EmployeeData)
missingValueCheck(EmployeeData)
```

B.3 Outlier Analysis

```
# Get cnames of numeric varaibles not dependent on ID
numeric_cnames = ['Work_load_Average/day_','Hit_target']

# Impute Outliers with NA
for i in numeric_cnames:
    q75, q25 = np.nanpercentile(EmployeeData.loc[:,i],[75, 25])
    iqr = q75 - q25
    min = q25 - (iqr*1.5)
    max = q75 + (iqr*1.5)
    EmployeeData.loc[EmployeeData[i] < min, i] = np.nan
    EmployeeData.loc[EmployeeData[i] > max, i] = np.nan
#Impute with KNN
EmployeeData = pd.DataFrame(KNN(k = 7).complete(EmployeeData),
                            columns = EmployeeData.columns)
EmployeeData = preprocessing(EmployeeData)
missingValueCheck(EmployeeData)
```

B.4 Feature Selection

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_regression
X = EmployeeData.iloc[:,:-1].values
y = EmployeeData.iloc[:,20].values
# Create an SelectKBest object to select features with two best ANOVA F-Values
selector = SelectKBest(f_regression, k=7)
# Apply the SelectKBest object to the features and target
X = selector.fit_transform(X, y)
selector.scores_
```

B.5 Model Selection

```
# Multiple Linear Regression
from sklearn.linear_model import LinearRegression
lm_regressor = LinearRegression()
lm_regressor.fit(X_train, y_train)
lm_predict = lm_regressor.predict(X_test)
RMSE(y_test, lm_predict)
MSE(y_test, lm_predict)

# Decision Tree Regressor
from sklearn.tree import DecisionTreeRegressor
DT_regressor = DecisionTreeRegressor()
DT_regressor.fit(X_train, y_train)
DT_predict = DT_regressor.predict(X_test)
RMSE(y_test, DT_predict)
MSE(y_test, DT_predict)

# Random Forest Regressor
from sklearn.ensemble import RandomForestRegressor
RF_regressor = RandomForestRegressor()
RF_regressor.fit(X_train, y_train)
RF_predict = RF_regressor.predict(X_test)
RMSE(y_test, RF_predict)
MSE(y_test, RF_predict)

# Support Vector Regressor
from sklearn.svm import SVR
SVR_regressor = SVR(kernel='rbf')
SVR_regressor.fit(X_train, y_train)
SVR_predict = SVR_regressor.predict(X_test)
RMSE(y_test, SVR_predict)
MSE(y_test, SVR_predict)
```

B.6 Conclusion

```
# Suggesting the changes
RF_regressor_p1 = RandomForestRegressor().fit(X, y)
RF_regressor_p1.feature_importances_

# Calculating Losses
p2_data = X
#Predict for new test cases
p2_predict = RF_regressor.predict(p2_data)
p2_predict = pd.DataFrame(p2_predict)
p2_predict.columns = ['predictions']
# Add predicted values to the DataSet
p2_frames = [EmployeeData, p2_predict]
p2_dataSet = pd.concat(p2_frames, axis=1)

# Calculate the total Loss
Loss = 0
for i in range(0,p2_dataSet.shape[0]):
    if (p2_dataSet['Hit_target'][i] != 100):
        if (p2_dataSet['Age'][i] >= 25 and  p2_dataSet['Age'][i] <= 32):
            Loss += (p2_dataSet['Disciplinary_failure'][i] + 1) * 2000 +
                    (p2_dataSet['Education'][i] + 1 + 1) * 500 +
                    p2_dataSet['predictions'][i] * 1000
        elif (p2_dataSet['Age'][i] >= 33 and  p2_dataSet['Age'][i] <= 40):
            Loss += (p2_dataSet['Disciplinary_failure'][i] + 1) * 2000 +
```

```
                        (p2_dataSet['Education'][i] + 1 + 2) * 500 +
                        p2_dataSet['predictions'][i] * 1000
        elif (p2_dataSet['Age'][i] >= 41 and  p2_dataSet['Age'][i] <= 49):
            Loss += (p2_dataSet['Disciplinary failure'][i] + 1) * 2000 +
                        (p2_dataSet['Education'][i] + 1 + 3) * 500 +
                        p2_dataSet['predictions'][i] * 1000
        elif (p2_dataSet['Age'][i] >= 50 and  p2_dataSet['Age'][i] <= 60):
            Loss += (p2_dataSet['Disciplinary failure'][i] + 1) * 2000 +
                        (p2_dataSet['Education'][i] + 1 + 4) * 500 +
                        p2_dataSet['predictions'][i] * 1000

# To calculate loss per month
Loss = Loss/12
print(Loss)
```

B.7 Complete Code

```
# Import Libraries
import os
import numpy as np
import pandas as pd
from fancyimpute import KNN

# Load Data
xls = pd.ExcelFile("Absenteeism_at_work_Project.xls")
EmployeeData = xls.parse()

# Get Cnames
def get_cname(data):
    all_cnames = []
    num_cnames = []
    cat_cnames = []
    for i in data.columns:
        all_cnames.append(str(i))
        if(data[i].dtype == "object"):
            cat_cnames.append(str(i))
        else:
            num_cnames.append(str(i))
    cnames = [all_cnames, num_cnames, cat_cnames]
    return(cnames)

# Get cnames - cnames[0] - all cnames;
# cnames[1] - numeric cnames;
# cnames[2] - categorical cnames
cnames = get_cname(EmployeeData)

#rows, cols = EmployeeData.shape
rows = EmployeeData.shape[0] #gives number of row count
cols = EmployeeData.shape[1] #gives number of col count

# Exploratory Data Analysis
for i in cnames[0]:
    print(str(i) + "    " + str(type(EmployeeData[i][1])))

# Change Data as per problem requirement
    for i in range(0,rows):
        if EmployeeData["Absenteeism time in hours"][i] != 0:
            if EmployeeData["Reason for absence"][i] == 0:
                EmployeeData["Reason for absence"][i] = np.nan
            if EmployeeData["Month of absence"][i] == 0:
```

```
                    EmployeeData["Month_of_absence"][i] = np.nan

def preprocessing(EmployeeData):
    # Change into require Data Types
    EmployeeData["ID"] = EmployeeData["ID"].astype(str)
    EmployeeData["Reason_for_absence"] =
                                EmployeeData["Reason_for_absence"].astype(str)
    EmployeeData["Month_of_absence"] = EmployeeData["Month_of_absence"].astype(str)
    EmployeeData["Day_of_the_week"] = EmployeeData["Day_of_the_week"].astype(str)
    EmployeeData["Seasons"] = EmployeeData["Seasons"].astype(str)
    EmployeeData["Disciplinary_failure"] =
                                EmployeeData["Disciplinary_failure"].astype(str)
    EmployeeData["Education"] = EmployeeData["Education"].astype(str)
    EmployeeData["Son"] = EmployeeData["Son"].astype(str)
    EmployeeData["Social_drinker"] = EmployeeData["Social_drinker"].astype(str)
    EmployeeData["Social_smoker"] = EmployeeData["Social_smoker"].astype(str)
    EmployeeData["Pet"] = EmployeeData["Pet"].astype(str)

    # Change NaN string values back to NaN
    EmployeeData["ID"] = EmployeeData["ID"].replace("nan",np.nan)
    EmployeeData["Reason_for_absence"] =
                        EmployeeData["Reason_for_absence"].replace("nan",np.nan)
    EmployeeData["Month_of_absence"] =
                        EmployeeData["Month_of_absence"].replace("nan",np.nan)
    EmployeeData["Day_of_the_week"] =
                        EmployeeData["Day_of_the_week"].replace("nan",np.nan)
    EmployeeData["Seasons"] = EmployeeData["Seasons"].replace("nan",np.nan)
    EmployeeData["Disciplinary_failure"] =
                        EmployeeData["Disciplinary_failure"].replace("nan",np.nan)
    EmployeeData["Education"] = EmployeeData["Education"].replace("nan",np.nan)
    EmployeeData["Son"] = EmployeeData["Son"].replace("nan",np.nan)
    EmployeeData["Social_drinker"] =
                        EmployeeData["Social_drinker"].replace("nan",np.nan)
    EmployeeData["Social_smoker"] =
                        EmployeeData["Social_smoker"].replace("nan",np.nan)
    EmployeeData["Pet"] = EmployeeData["Pet"].replace("nan",np.nan)

    # Covert factor varaible values to labels
    for i in range(0, len(EmployeeData.columns)):
        if(EmployeeData.iloc[:,i].dtypes == 'object'):
            EmployeeData.iloc[:,i] = pd.Categorical(EmployeeData.iloc[:,i])
            EmployeeData.iloc[:,i] = EmployeeData.iloc[:,i].cat.codes
            EmployeeData.iloc[:,i] = EmployeeData.iloc[:,i].astype('object')

    # Convert -1 values back to NaN
    for i in cnames[0]:
        for j in range(0,rows):
            if EmployeeData.loc[j,i] == -1:
                EmployeeData.loc[j,i] = np.nan

    return EmployeeData

EmployeeData = preprocessing(EmployeeData)
cnames = get_cname(EmployeeData)

# Missing Value analysis
def missingValueCheck(data):
    print(data.isna().sum())
missingValueCheck(EmployeeData)
```

```python
# Impute values related to ID
Depenedent_ID =  ["ID","Transportation expense","Service time","Age","Height",
                  "Distance from Residence to Work","Education","Son","Weight",
                  "Social smoker","Social drinker","Pet","Body mass index"]
Depenedent_ID_data = EmployeeData[Depenedent_ID].copy()
Depenedent_ID_data = Depenedent_ID_data.groupby("ID").max()
Depenedent_ID.remove('ID')

for i in Depenedent_ID:
    for j in range(0,rows):
        RI = EmployeeData["ID"][j]
        if np.isnan(EmployeeData.loc[j,i]):
            EmployeeData[i][j] = Depenedent_ID_data.loc[RI,i]

# Impute values for other variables with KNN
EmployeeData = pd.DataFrame(KNN(k = 7).complete(EmployeeData),
                            columns = EmployeeData.columns)
EmployeeData = preprocessing(EmployeeData)
missingValueCheck(EmployeeData)

# Outlier Analysis
# Get cnames of numeric varaibles not dependent on ID
numeric_cnames = ['Work load Average/day ','Hit target']

# Impute Outliers with NA
for i in numeric_cnames:
    q75, q25 = np.nanpercentile(EmployeeData.loc[:,i],[75, 25])
    iqr = q75 - q25
    min = q25 - (iqr*1.5)
    max = q75 + (iqr*1.5)
    EmployeeData.loc[EmployeeData[i] < min, i] = np.nan
    EmployeeData.loc[EmployeeData[i] > max, i] = np.nan
#Impute with KNN
EmployeeData = pd.DataFrame(KNN(k = 7).complete(EmployeeData),
                            columns = EmployeeData.columns)
EmployeeData = preprocessing(EmployeeData)
missingValueCheck(EmployeeData)

# Feature Selection
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_regression
X = EmployeeData.iloc[:,:-1].values
y = EmployeeData.iloc[:,20].values
# Create an SelectKBest object to select features with two best ANOVA F-Values
selector = SelectKBest(f_regression, k=7)
# Apply the SelectKBest object to the features and target
X = selector.fit_transform(X, y)
selector.scores_

# Feature Scaling
#Nomalisation
from sklearn.preprocessing import normalize
X = normalize(X, norm='l2')
#for i in range(0,(X_kbest.shape[1]-1)):
#    X_kbest[i] = (X_kbest[i] - np.min(X_kbest[i])) /
                 (np.max(X_kbest[i]) - np.min(X_kbest[i]))

# Error Matrix
from sklearn.metrics import mean_squared_error
from math import sqrt
```

```python
def RMSE(y, pred):
    print(sqrt(mean_squared_error(y, pred)))

def MSE(y, pred):
    print(mean_squared_error(y, pred))

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Multiple Linear Regression
from sklearn.linear_model import LinearRegression
lm_regressor = LinearRegression()
lm_regressor.fit(X_train, y_train)
lm_predict = lm_regressor.predict(X_test)
RMSE(y_test, lm_predict)
MSE(y_test, lm_predict)

# Decision Tree Regressor
from sklearn.tree import DecisionTreeRegressor
DT_regressor = DecisionTreeRegressor()
DT_regressor.fit(X_train, y_train)
DT_predict = DT_regressor.predict(X_test)
RMSE(y_test, DT_predict)
MSE(y_test, DT_predict)

# Random Forest Regressor
from sklearn.ensemble import RandomForestRegressor
RF_regressor = RandomForestRegressor()
RF_regressor.fit(X_train, y_train)
RF_predict = RF_regressor.predict(X_test)
RMSE(y_test, RF_predict)
MSE(y_test, RF_predict)

# Support Vector Regressor
from sklearn.svm import SVR
SVR_regressor = SVR(kernel='rbf')
SVR_regressor.fit(X_train, y_train)
SVR_predict = SVR_regressor.predict(X_test)
RMSE(y_test, SVR_predict)
MSE(y_test, SVR_predict)

############################## Problems ##################################
# Suggesting the changes
RF_regressor_p1 = RandomForestRegressor().fit(X, y)
RF_regressor_p1.feature_importances_

# Calculating Losses
p2_data = X
#Predict for new test cases
p2_predict = RF_regressor.predict(p2_data)
p2_predict = pd.DataFrame(p2_predict)
p2_predict.columns = ['predictions']
# Add predicted values to the DataSet
p2_frames = [EmployeeData, p2_predict]
p2_dataSet = pd.concat(p2_frames, axis=1)

# Calculate the total Loss
Loss = 0
for i in range(0,p2_dataSet.shape[0]):
    if (p2_dataSet['Hit_target'][i] != 100):
```

```python
        if (p2_dataSet['Age'][i] >= 25 and  p2_dataSet['Age'][i] <= 32):
            Loss += (p2_dataSet['Disciplinary failure'][i] + 1) * 2000 +
                        (p2_dataSet['Education'][i] + 1 + 1) * 500 +
                        p2_dataSet['predictions'][i] * 1000
        elif (p2_dataSet['Age'][i] >= 33 and  p2_dataSet['Age'][i] <= 40):
            Loss += (p2_dataSet['Disciplinary failure'][i] + 1) * 2000 +
                        (p2_dataSet['Education'][i] + 1 + 2) * 500 +
                        p2_dataSet['predictions'][i] * 1000
        elif (p2_dataSet['Age'][i] >= 41 and  p2_dataSet['Age'][i] <= 49):
            Loss += (p2_dataSet['Disciplinary failure'][i] + 1) * 2000 +
                        (p2_dataSet['Education'][i] + 1 + 3) * 500 +
                        p2_dataSet['predictions'][i] * 1000
        elif (p2_dataSet['Age'][i] >= 50 and  p2_dataSet['Age'][i] <= 60):
            Loss += (p2_dataSet['Disciplinary failure'][i] + 1) * 2000 +
                        (p2_dataSet['Education'][i] + 1 + 4) * 500 +
                        p2_dataSet['predictions'][i] * 1000

# To calculate loss per month
Loss = Loss/12
print(Loss)
```