|  |
| --- |
|  |
|  | **Employee Absenteeism R Coding :-**  # clearing  rm(list=ls(all=T)) |
|  | # Setting working directory |
|  | setwd("C:/Users/Ujjwal/Desktop/Project 1") |
|  |  |
|  | # Loading libraries |
|  | library(ggplot2) |
|  | library(corrgram) |
|  | library(DMwR) |
|  | library(caret) |
|  | library(randomForest) |
|  | library(unbalanced) |
|  | library(dummies) |
|  | library(e1071) |
|  | library(Information) |
|  | library(MASS) |
|  | library(rpart) |
|  | library(gbm) |
|  | library(ROSE) |
|  | library(xlsx) |
|  | library(DataCombine) |
|  | library(rpart) |
|  |  |
|  | ## Reading the data |
|  | df = read.xlsx('Absenteeism\_at\_work\_Project.xls', sheetIndex = 1) |
|  |  |
|  |  |
|  | # \_\_\_\_\_\_\_\_\_\_\_\_\_Exploratory Data Analysis\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | # Shape of the data |
|  | dim(df) |
|  | # Viewing data |
|  | view(df) |
|  | # Structure of the data |
|  | str(df) |
|  | # Variable names of the data |
|  | colnames(df) |
|  | # From the above EDA and problem statement categorizing data in 2 category "continuous" and "categorical" |
|  | continuous\_vars = c('Distance.from.Residence.to.Work', 'Service.time', 'Age', |
|  | 'Work.load.Average.day.', 'Transportation.expense', |
|  | 'Hit.target', 'Weight', 'Height', |
|  | 'Body.mass.index', 'Absenteeism.time.in.hours') |
|  |  |
|  | catagorical\_vars = c('ID','Reason.for.absence','Month.of.absence','Day.of.the.week', |
|  | 'Seasons','Disciplinary.failure', 'Education', 'Social.drinker', |
|  | 'Social.smoker', 'Son', 'Pet') |
|  |  |
|  |  |
|  |  |
|  | # \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Missing Values Analysis\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | #Creating dataframe with missing values present in each variable |
|  | missing\_val = data.frame(apply(df,2,function(x){sum(is.na(x))})) |
|  | missing\_val$Columns = row.names(missing\_val) |
|  | names(missing\_val)[1] = "Missing\_percentage" |
|  |  |
|  | #Calculating percentage missing value |
|  | missing\_val$Missing\_percentage = (missing\_val$Missing\_percentage/nrow(df)) \* 100 |
|  |  |
|  | # Sorting missing\_val in Descending order |
|  | missing\_val = missing\_val[order(-missing\_val$Missing\_percentage),] |
|  | row.names(missing\_val) = NULL |
|  |  |
|  | # Reordering columns |
|  | missing\_val = missing\_val[,c(2,1)] |
|  |  |
|  | # Saving output result into csv file |
|  | write.csv(missing\_val, "Missing\_perc\_R.csv", row.names = F) |
|  |  |
|  | # # Plot |
|  | ggplot(data = missing\_val[1:18,], aes(x=reorder(Columns, -Missing\_percentage),y = Missing\_percentage))+ geom\_bar(stat = "identity",fill = "grey")+xlab("Variables")+ |
|  |  |
|  | ggtitle("Missing data percentage") + theme\_bw() |
|  |  |
|  | # Actual Value = 23 |
|  | # Mean = 26.68 |
|  | # Median = 25 |
|  | # KNN = 23 |
|  |  |
|  |  |
|  | # Mean Method |
|  | df$Body.mass.index[is.na(df$Body.mass.index)] = mean(df$Body.mass.index, na.rm = T) |
|  |  |
|  | # Median Method |
|  | d$Body.mass.index[is.na(df$Body.mass.index)] = median(df$Body.mass.index, na.rm = T) |
|  |  |
|  | # kNN Imputation |
|  | df = knnImputation(df, k = 3) |
|  |  |
|  | # Checking for missing value |
|  | sum(is.na(df)) |
|  |  |
|  |  |
|  | #\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Outlier Analysis\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | # BoxPlots - Distribution and Outlier Check |
|  |  |
|  | # Boxplot for continuous variables |
|  | for (i in 1:length(continuous\_vars)) |
|  | { |
|  | assign(paste0("gn",i), ggplot(aes\_string(y = (continuous\_vars[i]), x = "Absenteeism.time.in.hours"), data = subset(df))+ |
|  | stat\_boxplot(geom = "errorbar", width = 0.5) + |
|  | geom\_boxplot(outlier.colour="red", fill = "grey" ,outlier.shape=18, |
|  | outlier.size=1, notch=FALSE) + |
|  | theme(legend.position="bottom")+ |
|  | labs(y=continuous\_vars[i],x="Absenteeism.time.in.hours")+ |
|  | ggtitle(paste("Box plot of absenteeism for",continuous\_vars[i]))) |
|  | } |
|  |  |
|  | # Plotting plots together |
|  | gridExtra::grid.arrange(gn1,gn2,ncol=2) |
|  | gridExtra::grid.arrange(gn3,gn4,ncol=2) |
|  | gridExtra::grid.arrange(gn5,gn6,ncol=2) |
|  | gridExtra::grid.arrange(gn7,gn8,ncol=2) |
|  | gridExtra::grid.arrange(gn9,gn10,ncol=2) |
|  |  |
|  |  |
|  | # #Remove outliers using boxplot method |
|  |  |
|  | # #loop to remove from all variables |
|  | for(i in continuous\_vars) |
|  | { |
|  | print(i) |
|  | val = df[,i][df[,i] %in% boxplot.stats(df[,i])$out] |
|  | #print(length(val)) |
|  | df = df[which(!df[,i] %in% val),] |
|  | } |
|  |  |
|  | #Replace all outliers with NA and impute |
|  | for(i in continuous\_vars) |
|  | { |
|  | val = df[,i][df[,i] %in% boxplot.stats(df[,i])$out] |
|  | #print(length(val)) |
|  | df[,i][df[,i] %in% val] = NA |
|  | } |
|  |  |
|  | # Imputing missing values |
|  | df = knnImputation(df,k=3) |
|  |  |
|  |  |
|  | #\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Feature Selection\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |
|  | # Correlation Plot |
|  | corrgram(df[,continuous\_vars], order = F, |
|  | upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot") |
|  |  |
|  | # ANOVA test for Categorical variable |
|  | summary(aov(formula = Absenteeism.time.in.hours~ID,data = df)) |
|  | summary(aov(formula = Absenteeism.time.in.hours~Reason.for.absence,data = df)) |
|  | summary(aov(formula = Absenteeism.time.in.hours~Month.of.absence,data = df)) |
|  | summary(aov(formula = Absenteeism.time.in.hours~Day.of.the.week,data = df)) |
|  | summary(aov(formula = Absenteeism.time.in.hours~Seasons,data = df)) |
|  | summary(aov(formula = Absenteeism.time.in.hours~Disciplinary.failure,data = df)) |
|  | summary(aov(formula = Absenteeism.time.in.hours~Education,data = df)) |
|  | summary(aov(formula = Absenteeism.time.in.hours~Social.drinker,data = df)) |
|  | summary(aov(formula = Absenteeism.time.in.hours~Social.smoker,data = df)) |
|  | summary(aov(formula = Absenteeism.time.in.hours~Son,data = df)) |
|  | summary(aov(formula = Absenteeism.time.in.hours~Pet,data = df)) |
|  |  |
|  |  |
|  | # Dimension Reduction |
|  | df = subset(df, select = -c(Weight)) |
|  |  |
|  |  |
|  | #\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Feature Scaling\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | #Normality check |
|  | hist(df$Absenteeism.time.in.hours) |
|  |  |
|  | # Updating the continuous and catagorical variable |
|  | continuous\_vars = c('Distance.from.Residence.to.Work', 'Service.time', 'Age', |
|  | 'Work.load.Average.day.', 'Transportation.expense', |
|  | 'Hit.target', 'Height', |
|  | 'Body.mass.index') |
|  |  |
|  | catagorical\_vars = c('ID','Reason.for.absence','Disciplinary.failure', |
|  | 'Social.drinker', 'Son', 'Pet', 'Month.of.absence', 'Day.of.the.week', |
|  | 'Seasons', 'Education', 'Social.smoker') |
|  |  |
|  |  |
|  | # Normalization |
|  | for(i in continuous\_vars) |
|  | { |
|  | print(i) |
|  | df[,i] = (df[,i] - min(df[,i]))/(max(df[,i])-min(df[,i])) |
|  | } |
|  |  |
|  | # Creating dummy variables for categorical variables |
|  | library(mlr) |
|  | df = dummy.data.frame(df, catagorical\_vars) |
|  |  |
|  |  |
|  | #\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Model Development\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | #Cleaning the environment |
|  | rmExcept("df") |
|  |  |
|  | #Divide data into train and test using stratified sampling method |
|  | set.seed(123) |
|  | train.index = sample(1:nrow(df), 0.8 \* nrow(df)) |
|  | train = df[ train.index,] |
|  | test = df[-train.index,] |
|  |  |
|  | #Decision tree for classification |
|  | #Develop Model on training data |
|  | fit\_DT = rpart(Absenteeism.time.in.hours ~., data = train, method = "anova") |
|  |  |
|  | #Summary of DT model |
|  | summary(fit\_DT) |
|  |  |
|  | #write rules into disk |
|  | write(capture.output(summary(fit\_DT)), "Rules.txt") |
|  |  |
|  | #Lets predict for training data |
|  | pred\_DT\_train = predict(fit\_DT, train[,names(test) != "Absenteeism.time.in.hours"]) |
|  |  |
|  | #Lets predict for training data |
|  | pred\_DT\_test = predict(fit\_DT,test[,names(test) != "Absenteeism.time.in.hours"]) |
|  |  |
|  |  |
|  | # For training data |
|  | print(postResample(pred = pred\_DT\_train, obs = train[,107])) |
|  |  |
|  | # For testing data |
|  | print(postResample(pred = pred\_DT\_test, obs = test[,107])) |
|  |  |
|  |  |
|  | #\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Linear Regression\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | set.seed(123) |
|  |  |
|  | #Develop Model on training data |
|  | fit\_LR = lm(Absenteeism.time.in.hours ~ ., data = train) |
|  |  |
|  | #Lets predict for training data |
|  | pred\_LR\_train = predict(fit\_LR, train[,names(test) != "Absenteeism.time.in.hours"]) |
|  |  |
|  |  |
|  | #Lets predict for testing data |
|  | pred\_LR\_test = predict(fit\_LR,test[,names(test) != "Absenteeism.time.in.hours"]) |
|  |  |
|  | # For training data |
|  | print(postResample(pred = pred\_LR\_train, obs = train[,107])) |
|  |  |
|  | # For testing data |
|  | print(postResample(pred = pred\_LR\_test, obs = test[,107])) |
|  |  |
|  |  |
|  | #\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Random Forest\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |
|  | set.seed(123) |
|  |  |
|  | #Develop Model on training data |
|  | fit\_RF = randomForest(Absenteeism.time.in.hours~., data = train) |
|  |  |
|  | #Lets predict for training data |
|  | pred\_RF\_train = predict(fit\_RF, train[,names(test) != "Absenteeism.time.in.hours"]) |
|  |  |
|  | #Lets predict for testing data |
|  | pred\_RF\_test = predict(fit\_RF,test[,names(test) != "Absenteeism.time.in.hours"]) |
|  |  |
|  | # For training data |
|  | print(postResample(pred = pred\_RF\_train, obs = train[,107])) |
|  |  |
|  | # For testing data |
|  | print(postResample(pred = pred\_RF\_test, obs = test[,107])) |
|  |  |
|  |  |
|  | #\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_XGBoost\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |
|  | set.seed(123) |
|  |  |
|  | #Develop Model on training data |
|  | fit\_XGB = gbm(Absenteeism.time.in.hours~., data = train, n.trees = 500, interaction.depth = 2) |
|  |  |
|  | #Lets predict for training data |
|  | pred\_XGB\_train = predict(fit\_XGB, train[,names(test) != "Absenteeism.time.in.hours"], n.trees = 500) |
|  |  |
|  | #Lets predict for testing data |
|  | pred\_XGB\_test = predict(fit\_XGB,test[,names(test) != "Absenteeism.time.in.hours"], n.trees = 500) |
|  |  |
|  | # For training data |
|  | print(postResample(pred = pred\_XGB\_train, obs = train[,107])) |
|  |  |
|  | # For testing data |
|  | print(postResample(pred = pred\_XGB\_test, obs = test[,107])) |
|  |  |
|  |  |
|  |  |
|  | #\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Dimensionality Reduction using PCA\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |
|  |  |
|  | #principal component analysis |
|  | prin\_comp = prcomp(train) |
|  |  |
|  | #compute standard deviation of each principal component |
|  | std\_dev = prin\_comp$sdev |
|  |  |
|  | #compute variance |
|  | pr\_var = std\_dev^2 |
|  |  |
|  | #proportion of variance explained |
|  | prop\_varex = pr\_var/sum(pr\_var) |
|  |  |
|  | #cumulative scree plot |
|  | plot(cumsum(prop\_varex), xlab = "Principal Component", |
|  | ylab = "Cumulative Proportion of Variance Explained", |
|  | type = "b") |
|  |  |
|  | #add a training set with principal components |
|  | train.data = data.frame(Absenteeism.time.in.hours = train$Absenteeism.time.in.hours, prin\_comp$x) |
|  |  |
|  | # From the above plot selecting 45 components since it explains almost 95+ % data variance |
|  | train.data =train.data[,1:45] |
|  |  |
|  | #transform test into PCA |
|  | test.data = predict(prin\_comp, newdata = test) |
|  | test.data = as.data.frame(test.data) |
|  |  |
|  | #select the first 45 components |
|  | test.data=test.data[,1:45] |
|  |  |
|  |  |
|  | #\_\_\_\_\_\_\_\_\_\_\_\_Model Development after Dimensionality Reduction\_\_\_\_\_\_\_\_ |
|  |  |
|  | #\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Decision tree for classification\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |
|  | #Develop Model on training data |
|  | fit\_DT = rpart(Absenteeism.time.in.hours ~., data = train.data, method = "anova") |
|  |  |
|  |  |
|  | #Lets predict for training data |
|  | pred\_DT\_train = predict(fit\_DT, train.data) |
|  |  |
|  | #Lets predict for training data |
|  | pred\_DT\_test = predict(fit\_DT,test.data) |
|  |  |
|  |  |
|  | # For training data |
|  | print(postResample(pred = pred\_DT\_train, obs = train$Absenteeism.time.in.hours)) |
|  |  |
|  | # For testing data |
|  | print(postResample(pred = pred\_DT\_test, obs = test$Absenteeism.time.in.hours)) |
|  |  |
|  |  |
|  |  |
|  | #\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Linear Regression\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |
|  | #Develop Model on training data |
|  | fit\_LR = lm(Absenteeism.time.in.hours ~ ., data = train.data) |
|  |  |
|  | #Lets predict for training data |
|  | pred\_LR\_train = predict(fit\_LR, train.data) |
|  |  |
|  | #Lets predict for testing data |
|  | pred\_LR\_test = predict(fit\_LR,test.data) |
|  |  |
|  | # For training data |
|  | print(postResample(pred = pred\_LR\_train, obs = train$Absenteeism.time.in.hours)) |
|  |  |
|  | # For testing data |
|  | print(postResample(pred = pred\_LR\_test, obs =test$Absenteeism.time.in.hours)) |
|  |  |
|  |  |
|  | #\_\_\_\_\_\_\_\_\_\_\_\_\_\_Random Forest\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |
|  | #Develop Model on training data |
|  | fit\_RF = randomForest(Absenteeism.time.in.hours~., data = train.data) |
|  |  |
|  | #Lets predict for training data |
|  | pred\_RF\_train = predict(fit\_RF, train.data) |
|  |  |
|  | #Lets predict for testing data |
|  | pred\_RF\_test = predict(fit\_RF,test.data) |
|  |  |
|  | # For training data |
|  | print(postResample(pred = pred\_RF\_train, obs = train$Absenteeism.time.in.hours)) |
|  |  |
|  | # For testing data |
|  | print(postResample(pred = pred\_RF\_test, obs = test$Absenteeism.time.in.hours)) |
|  |  |
|  |  |
|  | #\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_XGBoost\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |
|  | #Develop Model on training data |
|  | fit\_XGB = gbm(Absenteeism.time.in.hours~., data = train.data, n.trees = 500, interaction.depth = 2) |
|  |  |
|  | #Lets predict for training data |
|  | pred\_XGB\_train = predict(fit\_XGB, train.data, n.trees = 500) |
|  |  |
|  | #Lets predict for testing data |
|  | pred\_XGB\_test = predict(fit\_XGB,test.data, n.trees = 500) |
|  |  |
|  | # For training data |
|  | print(postResample(pred = pred\_XGB\_train, obs = train$Absenteeism.time.in.hours)) |
|  |  |
|  | # For testing data |
|  | print(postResample(pred = pred\_XGB\_test, obs = test$Absenteeism.time.in.hours)) |