**R-Code:**

|  |
| --- |
|  |
|  | #Remove all objects stored  rm(list = ls()) |
|  |  |
|  | #Loading Libraries |
|  | libraries = c("plyr","dplyr", "ggplot2","rpart","dplyr","DMwR","randomForest", |
|  | "usdm","corrgram","DataCombine") |
|  | lapply(X = libraries, require, character.only = TRUE) |
|  | rm(libraries) |
|  |  |
|  | library(dummies) |
|  | library(caret) |
|  | library(rpart.plot) |
|  |  |
|  | #Loading the data |
|  | df = read.csv("day.csv",header=T) |
|  |  |
|  | #first few rows of data |
|  | head(df) |
|  |  |
|  | #str names of data |
|  | str(df) |
|  | ################# Exploratory Data Analysis ############## |
|  | df$dteday = as.Date(df$dteday) |
|  | df$season = as.factor(df$season) |
|  | df$yr = as.factor(df$yr) |
|  | df$mnth = as.factor(df$mnth) |
|  | df$holiday = as.factor(df$holiday) |
|  | df$weekday = as.factor(df$weekday) |
|  | df$workingday = as.factor(df$workingday) |
|  | df$weathersit = as.factor(df$weathersit) |
|  |  |
|  |  |
|  | ################## Feature Engineering ##################### |
|  |  |
|  | #Creating new variables |
|  | df$actual\_temp = df$temp \* 39 |
|  | df$actual\_atemp = df$atemp \* 50 |
|  | df$actual\_windspeed = df$windspeed \* 67 |
|  | df$actual\_hum = df$hum \* 100 |
|  |  |
|  | df$actual\_season = factor(x = df$season, levels = c(1,2,3,4), labels = c("Spring","Summer","Fall","Winter")) |
|  | df$actual\_year = factor(x = df$yr, levels = c(0,1), labels = c("2011","2012")) |
|  | df$actual\_workingday = factor(x = df$workingday, levels = c(0,1), labels = c("Holiday", "Working day")) |
|  | df$actual\_weathersit = factor(x = df$weathersit, levels = c(1,2,3,4), |
|  | labels = c("Clear","Cloudy/Mist","Rain/Snow/Fog","Heavy Rain/Snow/Fog")) |
|  |  |
|  | ################# Missing value analysis ############## |
|  |  |
|  | missing\_val = sapply(df,function(x){sum(is.null(df))}) |
|  | missing\_val |
|  |  |
|  | #There is no missing values so will move forward to data distribution |
|  |  |
|  | ########## Exploring data distribution by graphs ############### |
|  |  |
|  | #checking data distribution of categorical variables using bar graphs |
|  | bar\_season = ggplot(data = df,aes(x = actual\_season)) + |
|  | geom\_bar(fill = 'blue')+ggtitle("Count of Season") |
|  |  |
|  | bar\_season |
|  |  |
|  | bar\_weather = ggplot(data = df,aes(x = actual\_weathersit)) + |
|  | geom\_bar(fill = 'blue')+ggtitle("Count of Weatherlist") |
|  |  |
|  | bar\_weather |
|  |  |
|  | bar\_workingday = ggplot(data = df,aes(x = actual\_workingday)) + |
|  | geom\_bar(fill = 'blue')+ggtitle("Count of Workingday") |
|  |  |
|  | bar\_workingday |
|  |  |
|  | gridExtra::grid.arrange(bar\_season,bar\_weather,bar\_workingday,ncol=2) |
|  |  |
|  | #Checking data distribution numerical variables using histograms |
|  |  |
|  | hist\_temp = ggplot(data = df, aes(x =actual\_temp)) + ggtitle("Distribution of Temperature") + geom\_histogram(bins = 25,fill='blue') |
|  | hist\_atemp = ggplot(data = df, aes(x =actual\_atemp)) + ggtitle("Distribution of feeled Temprature") + geom\_histogram(bins = 25,fill='blue') |
|  | hist\_windspeed = ggplot(data = df, aes(x =actual\_windspeed)) + ggtitle("Distribution of Windspeed") + geom\_histogram(bins = 25,fill='blue') |
|  | hist\_hum = ggplot(data = df, aes(x =actual\_hum)) + ggtitle("Distribution of Humidity") + geom\_histogram(bins = 25,fill='blue') |
|  |  |
|  | gridExtra::grid.arrange(hist\_temp,hist\_atemp,hist\_windspeed,hist\_hum,ncol=2) |
|  |  |
|  | ########### Outlier Analysis ################ |
|  |  |
|  | continous\_var = c("actual\_temp","actual\_atemp","actual\_windspeed","actual\_hum") |
|  |  |
|  | for (i in 1:length(continous\_var)) |
|  | { |
|  | assign(paste0("gn",i), ggplot(aes\_string(y = continous\_var[i]), data = 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=continous\_var[i])+ |
|  | ggtitle(paste("Box plot for", continous\_var[i]))) |
|  | } |
|  | gridExtra::grid.arrange(gn1,gn3,gn2,gn4,ncol=2) |
|  |  |
|  | #Found that outliers in actual\_windspeed and actual\_hum |
|  | #Removing outliers in Windspeed |
|  |  |
|  | val = df[,19][df[,19] %in% boxplot.stats(df[,19])$out] |
|  | df = df[which(!df[,19] %in% val),] |
|  |  |
|  | #Removing outliers in humidity |
|  | val = df[,20][df[,20] %in% boxplot.stats(df[,20])$out] |
|  | df = df[which(!df[,20] %in% val),] |
|  |  |
|  | colnames(df) |
|  | ######################## Feature Selection ######################## |
|  |  |
|  | #Checking multicollinearity using VIF |
|  |  |
|  | df\_vif = df[,c('temp','atemp','windspeed','hum')] |
|  |  |
|  | vifcor(df\_vif1) |
|  |  |
|  |  |
|  | #Checking Collinearity by using Correlation graph |
|  | corrgram(df, order = FALSE,lower.panel = panel.shade,upper.panel = panel.pie, |
|  | text.panel = panel.txt, main="Correlation Graph") |
|  |  |
|  | #From above 2 Correlation analysis observed 'atemp' variable has multicollinearity problem |
|  | #Removing unwanted variables |
|  | df = subset(df,select = -c(holiday,instant,dteday,atemp,casual,registered,actual\_temp,actual\_atemp,actual\_windspeed, |
|  | actual\_hum,actual\_season,actual\_year,actual\_workingday,actual\_weathersit)) |
|  |  |
|  | #Taking copy of data |
|  | df2 = df |
|  | #df = df2 |
|  | colnames(df) |
|  |  |
|  | #Creating dummy variables for categorical variables to trick the Regression models |
|  |  |
|  | catagorical\_var = c('season','yr','mnth','weekday','workingday','weathersit') |
|  | df = dummy.data.frame(df, catagorical\_var) |
|  | colnames(df) |
|  |  |
|  | rmExcept(keepers = 'df') |
|  |  |
|  | ####################### Model Development ########################## |
|  |  |
|  | #Splitting data into train and test data |
|  | train\_index = sample(1:nrow(df), 0.8\*nrow(df)) |
|  | train = df[train\_index,] |
|  | test = df[-train\_index,] |
|  |  |
|  | #-------------------Decision Tree----------------------# |
|  |  |
|  | #training the data with rpart |
|  | dt\_model = rpart(cnt ~ ., data = train,method = "anova") |
|  | rpart.plot(dt\_model) |
|  |  |
|  | #Predicting test data |
|  | dt\_predictions = predict(dt\_model,test[,-34]) |
|  |  |
|  | #Create dataframe for actual and predicted values |
|  | df\_pred = data.frame("actual"=test[,34], "pred"=dt\_predictions) |
|  | head(df\_pred) |
|  | summary(dt\_model) |
|  |  |
|  | #Calculate RMSE and other error metrics |
|  | regr.eval(trues = test[,34], preds = dt\_predictions, stats = c("mae","mse","rmse","mape")) |
|  |  |
|  | #Calculate R-Squared |
|  | print(postResample(pred = dt\_predictions, obs = test[,34])) |
|  |  |
|  | # RMSE Rsquared MAE |
|  | #961.0064188 0.7676469 736.2084616 |
|  |  |
|  | #--------------------Linear Regression---------------------# |
|  |  |
|  | #Train the data using linear regression |
|  | lr\_model = lm(formula = cnt~., data = train) |
|  |  |
|  | #Check the summary of the model |
|  | summary(lr\_model) |
|  |  |
|  |  |
|  | #Predict the test cases |
|  | lr\_predictions = predict(lr\_model, test[,-34]) |
|  |  |
|  | #Create dataframe for actual and predicted values |
|  | df\_lin = cbind(df\_pred,lr\_predictions) |
|  | head(df\_lin) |
|  |  |
|  | #Calculate RMSE and other error metrics |
|  | regr.eval(trues = test[,34], preds = lr\_predictions, stats = c("mae","mse","rmse","mape")) |
|  |  |
|  | #Calculate R-Squared |
|  | print(postResample(pred = lr\_predictions, obs = test[,34])) |
|  |  |
|  | # RMSE Rsquared MAE |
|  | #847.8232089 0.8209717 629.3466018 |
|  |  |
|  |  |
|  | #--------------------------Random Forest-------------------# |
|  |  |
|  | #Train the data using random forest |
|  | rf\_model = randomForest(cnt~., data = train, ntree = 500) |
|  |  |
|  | #Predict the test cases |
|  | rf\_predictions = predict(rf\_model, test[,-34]) |
|  |  |
|  | #Create dataframe for actual and predicted values |
|  | df\_lin = cbind(df\_lin,rf\_predictions) |
|  | head(df\_lin) |
|  |  |
|  | #Calculate RMSE and other error metrics |
|  | regr.eval(trues = test[,34], preds = rf\_predictions, stats = c("mae","mse","rmse","mape")) |
|  | MAPE(test[,34], rf\_predictions) |
|  |  |
|  | #Calculate R-Squared |
|  | print(postResample(pred = rf\_predictions, obs = test[,34])) |
|  |  |
|  | # RMSE Rsquared MAE |
|  | #693.8210337 0.8804323 512.0623566 |