atchircUtils.R

atchirc

Fri Apr 07 00:14:06 2017

```
#
#      USER UTLITIES PACKAGE
#
# Frequently performed activites are formulated into
#  functions so as to avoid code repetition
#
# Atchireddy Chavva(atchireddi@gmail.com)
#
# Some useful keyboard shortcuts for package authoring:
#
#   Build and Reload Package:  'Ctrl + Shift + B'
#   Check Package:             'Ctrl + Shift + E'
#   Test Package:              'Ctrl + Shift + T'
```

```
library(ggplot2)
library(cowplot)



# ************************************************************************
#                   EDA Utils ----
# ************************************************************************

# Generates univariate plots for Numerical variables
#       Compute Skewness & Kurtosis
#       Compute slope & Intercept for Q-Q plot
#       Generates Consolidated boxplots
#       Individual boxplots
#       Individual Histplots
#       Q-Q plots
# @param df DataFrame, All columns must be numerical
# @param prefix Name tag, generated plots would be prefixed
# @return Generates plots under ../output/ folder
# @example
#
plotNumericalUnivariate  <- function(df,prefix) {
  nrows       <- nrow(df)
  df_num      <- data.frame()
  df_slp_int  <- data.frame()
  for (i in colnames(df)) {

    #   1. . . . Collate Variables ---
    df_num <- rbind(data.frame(var=rep(i,nrows),
                               val=df[[i]]
    ),
    df_num)
```

```
  #  2. . . . Caliculate slope & intercept ---
  # Find the slope and intercept of the line that passes through the 1st and 3rd
  # quartile of the normal q-q plot

  y     <- quantile(df[[i]], c(0.25, 0.75)) # Find the 1st and 3rd quartiles
  x     <- qnorm( c(0.25, 0.75))            # Find the matching normal values on the x-axis
  slope <- diff(y) / diff(x)                # Compute the line slope
  intr   <- y[1] - slope * x[1]             # Compute the line intercept
  mn     <- mean(df[[i]])                   # source for annotation co-ordinates
  sd     <- sd(df[[i]])                     # source for annotation co-ordinates
  df_slp_int <- rbind(data.frame(var=i,slp=slope,intcpt=intr,                    # slope & Intercept f
                                 skw=sprintf("Sk=%.2f",skewness(df[[i]])),      # Skewness
                                 kurt=sprintf("Ku=%.2f",kurtosis(df[[i]])),     # Kurtosis
                                 sky=mn+sd,               # y-cord for skewness annotation
                                 kuy=mn-sd),              # y-cord for Kurtosis Annotation
                      df_slp_int)
}

df_num$var <- as.factor(df_num$var) # convert variables to factors

# 3. . . . Combined Boxplot ---
pcb <- ggplot(df_num,
              aes(var,val)
)
pcb <- pcb + geom_boxplot(colour="blue",
                          outlier.color = "red"
)
# pcb <- pcb + stat_summary(fun.data = give.n, geom = "text")
pcb <- pcb + ggtitle("BOXPLOTs (Numerical variables)")
pcb <- pcb + theme(text=element_text(size = 10),          # title, labels
                   axis.text.x=element_text(angle = 45,
                                            vjust = 0.5), # rotate & align x-axis labels
                   axis.text = element_text(size = 8)     # axis tick labels
)
pcb <- pcb + labs(x="",y="")


# 4. . . . Individual Boxplots ---
pib <- ggplot(df_num,
              aes(val,val)
)
pib <- pib + geom_boxplot(colour="blue",
                          outlier.colour = "red")
pib <- pib + facet_wrap(~var,
                        scales = "free")
pib <- pib + ggtitle(" BOXPLOTs (Numerical Variables) ")
pib <- pib + theme(axis.text.x=element_text(angle = 90), # x-axis labels rotated 90deg
                   axis.text=element_text(size = 8),     # axis tick labels
                   text=element_text(size = 10)          # title, labels
)
pib <- pib + labs(x="",y="")

# 5. . . . Individual Histplots ---
```

```
pih <- ggplot(df_num,
              aes(val)
)
pih <- pih + geom_histogram(fill="darkgreen",aes(y=..density..))
pih <- pih + geom_density(colour="darkred")
pih <- pih + facet_wrap(~var,
                        scales = "free")
pih <- pih + ggtitle("DISTRIBUTION (Numerical Variables) ")
pih <- pih + theme(axis.text.x=element_text(angle = 90), # x-axis labels rotated 90deg
                   axis.text=element_text(size = 8),     # axis tick labels
                   text=element_text(size = 10)          # title, labels
)
pih <- pih + geom_text(data = df_slp_int,
                       aes(label=skw),
                       x=Inf,
                       y=Inf,
                       hjust = 1.2,
                       vjust = 2.2,
                       parse = T,
                       size=3)                           # skew annotation
pih <- pih + geom_text(data = df_slp_int,
                       aes(label=kurt),
                       x=Inf,
                       y=Inf,
                       hjust = 1.2,
                       vjust = 1.2,
                       parse = T,
                       size=3)                           # Kurt annotation

pih <- pih + labs(x="",y="")


# 6. . . . Combined Q-Q plots ---
pqq <- ggplot(df_num) + stat_qq(aes(sample=val),
                                color="red")             # qqplot
pqq <- pqq + facet_wrap(~var,
                        scales = "free")                 # facet_wrap around Var
pqq <- pqq + geom_abline(data = df_slp_int,
                         aes(intercept=intcpt,slope=slp),
                         color="blue")                   # qqline
pqq <- pqq + geom_text(data = df_slp_int,
                       aes(label=skw),
                       x=-Inf,
                       y=Inf,
                       hjust = -0.2,
                       vjust = 2.2,
                       parse = T,
                       size=3)                           # skew annotation
pqq <- pqq + geom_text(data = df_slp_int,
                       aes(label=kurt),
                       x=-Inf,
                       y=Inf,
                       hjust = -0.2,
```

```
                        vjust = 1.2,
                        parse = T,
                        size=3)                          # Kurt annotation
  pqq <- pqq + theme(text=element_text(size=10),         # title, labels
                     axis.text=element_text(size = 8)    # axis tick labels
  )
  pqq <- pqq + ggtitle("Q-Q Plots (Numerical variables)")
  pqq <- pqq + labs(x="",y="")


  # 7. . . . Save Plots ---
  # pdf("../output/univariate_numerical.pdf",paper = 'USr',width = 11)
  png(filename = paste("../output/",prefix,"_univariate_numerical%02d.png",sep = ""),
      width = 11.69,
      height = 8.27,
      units = "in",
      res = 288)
  # par(mfrow=c(3,1))  # Not Working
  print(plot_grid(pcb,pib))
  print(plot_grid(pib,pih))
  print(pqq)
  dev.off()

  # can't find a way to impose boxplot on histogram
}



# Dimensional Variable Univariate(barlot) Analysis plots
plotDimensionUnivariate <- function(df,dim) {
  v <- df[[ dim ]]
  p <- qplot(x=reorder(v,v,function(x)-length(x))) +
    theme(axis.text.x=element_text(angle = 45,hjust = 1)) +
    labs(x="",title=dim)
  print(p)
}



plotsDimensionsVsResponse <- function(data, dim, resp) {
  # since reorder with aes_string not working, will create local df
  # and used hardcode colnames for plotting

  local_df <- data[c(dim,resp)]
  colnames(local_df) <- c("dim", "resp")

  # barplot
  b <- ggplot(local_df, aes(x=reorder(dim,dim,function(x)-length(x)))) +
    geom_bar(aes(y=(..count..))) +
    theme(axis.text.x=element_text(angle = 45, hjust = 1)) +
    geom_text(aes(y=(..count..),
                  label = scales::percent( (..count..)/sum(..count..))),
```

```
              stat = "count",
              vjust = -0.5
    ) +
    labs(x="", title=dim)

    # percent barplot
  pb <- ggplot(local_df, aes(x=reorder(dim,dim,function(x)-length(x)),fill=resp)) +
    geom_bar(position = "fill") +
    theme(axis.text.x=element_text(angle = 45, hjust = 1)) +
    labs(x="", title=dim)

  # gridplot
  plot_grid(b,pb)
}



# percent barplot of multi-variate categorical variables
plotDimensionVsResponse <- function(data,dim,response) {
  ggplot(data, aes_string(dim)) +
    geom_bar(aes_string(fill = response), position = "fill") +
    theme(axis.text.x=element_text(angle = 45,vjust = 0.5))
}



# ******************************************************************************
#                   FEATURE ENGINEERING Utils ----
# ******************************************************************************

# Binning Model Year, 3 bins
#  2011-2015, 2006-2010, <2006
binYear <- function(year) {
  year <- as.numeric(year)
  if(year>2010) {
    return("2011_2015")
  } else if (year>2005) {
    return("2006_2010")
  } else {
    return("<2006")
  }
}



# Convert 2-level Categorical Variable to numeric type
twolevelDimensionToNum <- function(df) {
  for (i in colnames(df)) {
    levels(df[[i]]) <- c(1,0)
    df[[i]] <- as.numeric(levels(df[[i]]))[df[[i]]]
  }
  return(df)
}
```

```r
# COnvert categorical dependant variable to numeric type
# "Yes" <- 1
# "No"  <- 0
YesNoTonum <- function(v) {
  v <- as.character(v)
  v[v=="Yes"] <- "1"
  v[v=="No"]  <- "0"
  v <- as.numeric(v)
  return(v)
}


# ********************************************************************************
#                    MISSING VALUES TREATMENT Utils ----
# ********************************************************************************
# Replace missing Categorical variable with Mode
replaceWithMode <- function(v) {
  mod <- names(sort(-table(v)))[1]
  v[is.na(v)] <- mod
  return(v)
}

# Replace missing values with Mean
replaceWithMean <- function(v) {
  v[is.na(v)] <- mean(v,na.rm = T)
  return(v)
}



# Replace missing values with Median
replaceWithMedian <- function(v) {
  v[is.na(v)] <- median(v,na.rm = T)
  return(v)
}



# Replace missing values with Zero
replaceWithZero <- function(v) {
  v[is.na(v)] <- 0
  return(v)
}



# Mark missing values with string
replaceWithString <- function(v,str) {
  v <- as.character(v)
  v[is.na(v)] <- str
  v <- as.factor(v)
```

```
  return(v)
}



# ****************************************************************************
#                    DATA TYPE CORRECTION Utils ----
# ****************************************************************************

# Replace value of  Factor variable
replaceDimensionFactor <- function(v, oldval, newval) {
  v <- as.character(v)
  v[v==oldval] <- newval
  v <- as.factor(v)
  return(v)
}



# ****************************************************************************
#                    OUTLIER TREATMENT Utils ----
# ****************************************************************************
# Cap/floor outliers @ 25th and 95th percentile
capOutliers <- function(v) {
  # make sure NO missing values
  quantiles <- quantile( v, c(.05, .95 ) )
  v[ v < quantiles[1] ] <- quantiles[1]
  v[ v > quantiles[2] ] <- quantiles[2]
  return(v)
}

# ****************************************************************************
#                    DATA UNDERSTANDING Utils ----
# ****************************************************************************

# Summarize NA's
naSummary <- function(df) {
  obs <- nrow(df)
  cs <- colSums(is.na(df))  # NA's per column
  cls <- sapply(df,class)
  df <- data.frame(Vars = names(cs), NAS = cs, class=cls, row.names = NULL)
  df <- df[order(df$NAS),]
  df$perNAS <- 100*(df$NAS/obs)
  df
}

# ****************************************************************************
#                    MODEL EVALUATION Utils ----
# ****************************************************************************

# Plots MOdel performance, provided model and ref.labels
#  mdl -->  Model
#  ref -->  reference labels
```

```
#  clk -->  Line Colour
#  ad  -->  Should performance curve to be added to existing Plot
plotModelPerformance <- function(model,ref_label,color,add) {
  pred_prob <- predict(model, type = "response")
  mdl_score <- prediction(pred_prob, ref_label)
  perf      <- performance(mdl_score, "tpr", "fpr")
  plot(perf,col=color,add=add)
}



# Greps Model R2 Values
getModelR2 <- function(mdl) {
  mdl_sumry <- capture.output(summary(mdl))
  print(mdl_sumry[grepl("R-squared",mdl_sumry)])
}



# Extracts Coefficients from Model Summary
covertModelSummaryToDF <- function(smry) {
  # extract Coefficints sections
  start_coeff <- grep("Intercept",smry)         # starting coeff
  end_coeff   <- grep("---",smry)               # end coeff
  coeff       <- smry[start_coeff:end_coeff-1]  # extract coeff
  coeff       <- gsub("< ","<",coeff[-1])       # clean-up
  # Create unqual list of list to DataFrame
  coeff      <- strsplit(coeff,"\\s+")
  df_coeff   <- as.data.frame(do.call(rbind,lapply(coeff,'length<-',6)))
  colnames(df_coeff) <- c("var","Estimate","Std.Error",
                          "t-value","Pr(>|t|)","Significance") # add proper colnames
  return(df_coeff)
}



# function to consolidate model summary and model vif,
# shows merged output as table
# Inputs   : linear model
# Outputs  :  model summary and vif returned as df
viewModelSummaryVIF <- function(mdl) {

  # Model VIF
  print("....Generating Model VIF ....")
  mdl_vif   <- vif(mdl)
  df_vif    <-  data.frame(var=names(mdl_vif),
                           vif=as.numeric(mdl_vif))

  # Model Summary
  print("....Generating Model Summary ....")
  mdl_sumry <- capture.output(summary(mdl))
  df_sumry  <- mdlsmry2df(mdl_sumry)
```

```
  # Merger Summary and VIF  by variable
  print(".... Merging Summary and VIF ....")
  df <- merge(df_sumry, df_vif, by="var")
  return(df)
}
```

‘‘‘