All the code in order to reproduce all the tables and figures and all the procedure:

```
library(lmtest)
library(randtests)
library(car)
if (!require("devtools"))
  install.packages("devtools")
devtools::install_github("debinqiu/snpar")
library(snpar)

assumptionsTests = function(model){
  par(mfrow=c(1,2))

  plot(model, which = 2, main="QQ Plot")
  plot(model, which = 3)
  print(shapiro.test(rstandard(model)))
  # Constant variance

  print("NCV Test")
  require(lmtest)
  print(ncvTest(model))

  print("Stud Residuals")
  StudResiduals = rstudent(model)
  yhat = fitted(model)
  par(mfrow=c(1,2))
  plot(yhat, StudResiduals, main="Studentized Residuals in predicted Values", xlab =
"Predicted Values")
  abline(h=c(-2,2), col=2, lty=2)
  plot(yhat, StudResiduals^2, main="Sudentized Residuals^2 in predicted Values", xlab =
"Predicted Values")
  abline(h=4, col=2, lty=2)
```

```
  plot(model, which=3, main="Linearity")

  yhatQuantiles=cut(yhat, breaks=quantile(yhat, probs=seq(0,1,0.25)), dig.lab=6)
  print("Yhat Quantiles Table")
  print(table(yhatQuantiles))

  print("Levene test of Homogenity")
  print(leveneTest(rstudent(model)~yhatQuantiles))
  boxplot(rstudent(model)~yhatQuantiles, main="Box plot of Yhat/Residuals differences")

  # Non linearity
  print("Non linearity")
  residualPlot(model, type='rstudent', main="Linearity")
  # residualPlots(model, plot=F, type = "rstudent", main="Linearity 2")

  # Independence
  plot(rstudent(model), type='l',main="Error plot (Residuals)")
  require(snpar)
  print(runs.test(model$res)) # p > 0.5 we go for non Randomness
  print("DW test")
  print(dwtest(model)) # # p > 0.5 True auto Correlation > 0
}


# Import the dataset in R and removing the first 3 columns which we don't need and one more (is_weekend)
setwd('C:\\Users\\User\\Desktop\\Μεταπτυχιακό\\1) Statistics for Business Analytics I\\R Labs\\Graded Assignments\\Main Assignment 2022-23')
training_dataset <- read.csv(file = "alldata_onlinenews_45.csv",header = TRUE, sep = ";", dec = ",")
training_dataset <- training_dataset[,!names(training_dataset) %in% c("id", "url", "timedelta", "is_weekend")]
```

```
training_dataset$data_channel_is_lifestyle <-
as.factor(training_dataset$data_channel_is_lifestyle)

levels(training_dataset$data_channel_is_lifestyle) <- list("No" = "0", "Yes" = "1")

training_dataset$data_channel_is_entertainment <-
as.factor(training_dataset$data_channel_is_entertainment)

levels(training_dataset$data_channel_is_entertainment) <- list("No" = "0", "Yes" = "1")

training_dataset$data_channel_is_bus <- as.factor(training_dataset$data_channel_is_bus)

levels(training_dataset$data_channel_is_bus) <- list("No" = "0", "Yes" = "1")

training_dataset$data_channel_is_socmed <-
as.factor(training_dataset$data_channel_is_socmed)

levels(training_dataset$data_channel_is_socmed) <- list("No" = "0", "Yes" = "1")

training_dataset$data_channel_is_tech <- as.factor(training_dataset$data_channel_is_tech)

levels(training_dataset$data_channel_is_tech) <- list("No" = "0", "Yes" = "1")

training_dataset$data_channel_is_world <-
as.factor(training_dataset$data_channel_is_world)

levels(training_dataset$data_channel_is_world) <- list("No" = "0", "Yes" = "1")

training_dataset$weekday_is_monday <- as.factor(training_dataset$weekday_is_monday)

levels(training_dataset$weekday_is_monday) <- list("No" = "0", "Yes" = "1")

training_dataset$weekday_is_tuesday <- as.factor(training_dataset$weekday_is_tuesday)

levels(training_dataset$weekday_is_tuesday) <- list("No" = "0", "Yes" = "1")

training_dataset$weekday_is_wednesday <-
as.factor(training_dataset$weekday_is_wednesday)

levels(training_dataset$weekday_is_wednesday) <- list("No" = "0", "Yes" = "1")

training_dataset$weekday_is_thursday <- as.factor(training_dataset$weekday_is_thursday)

levels(training_dataset$weekday_is_thursday) <- list("No" = "0", "Yes" = "1")

training_dataset$weekday_is_friday <- as.factor(training_dataset$weekday_is_friday)

levels(training_dataset$weekday_is_friday) <- list("No" = "0", "Yes" = "1")

training_dataset$weekday_is_saturday <- as.factor(training_dataset$weekday_is_saturday)

levels(training_dataset$weekday_is_saturday) <- list("No" = "0", "Yes" = "1")

training_dataset$weekday_is_sunday <- as.factor(training_dataset$weekday_is_sunday)

levels(training_dataset$weekday_is_sunday) <- list("No" = "0", "Yes" = "1")

# Exploratory data Analysis
```

```
str(training_dataset)
summary(training_dataset)

library(psych)
index <- sapply(training_dataset, class) == "numeric" | sapply(training_dataset, class) == "integer"
training_dataset_num <- training_dataset[,index]
training_dataset_num[] <- sapply(training_dataset_num, as.numeric)
str(training_dataset_num)
training_dataset_fac <- training_dataset[,!index]
str(training_dataset_fac)


# Visual Analysis for numerical variables
training_dataset_num_without_share <- training_dataset_num[,-45]
library(Hmisc)
hist.data.frame(training_dataset_num_without_share)

# Visual Analysis for factor variables
par(mfrow=c(1,1)); n <- nrow(training_dataset_fac)
barplot(sapply(training_dataset_fac,table)/n, horiz=T, las=1, col=2:3, ylim=c(0,16), cex.names=0.4, mgp = c(3, 0, 0), xlabel=F)
legend('top', fil=2:3, legend=c('No','Yes'), ncol=2, bty='n',cex=0.5)

# Correlation Matrix
par(mfrow=c(1,1))
library(corrplot)
corrplot(round(cor(training_dataset_num), 2), tl.cex=0.5)
# We don't see any correlation between shares and any other attribute

# Shares (our response) on factor variables
```

```
par(mfrow=c(3,5))
for(j in 1:13){
  boxplot(training_dataset_num[,45]~training_dataset_fac[,j],
xlab=names(training_dataset_fac)[j], ylab='Share',cex.lab=2.0)
}

##Anova for Shares among factor categories
#anova1<-aov(training_dataset$shares~ .,data=training_dataset_fac)
#summary(anova1)

#library(nortest)
#lillie.test(anova1$res);shapiro.test(anova1$res)
#leveneTest(training_dataset$shares~ .,data=training_dataset_fac) ## all are rejected sale is skewwed so we will proceed with median
#library(dplyr)
#kruskal.test(training_dataset$shares~ .,data=training_dataset_fac)## it is rejected as we expected
#TukeyHSD(anova1,conf.level=0.95) # Median of pairs are equal or not factorial stuff

# Start with Models
# Plan is to first do a LASSO to get a subset of Variables that matter and then go for step-wise processes
# Start with a model that includes everything

model = lm(shares~ .,data=training_dataset)
summary(model)
# As we can see, the Adjusted R^2 is 0.007903 which is very low and that means that our model
# isn't going to fit at all on our data

# Prepare the Lasso
```

```
X = model.matrix(model)[,-1]
library(glmnet)
lasso = cv.glmnet(X, training_dataset$shares,alpha = 1)
par(mfrow=c(1,1))
plot(lasso)

min = coef(lasso, s = "lambda.min") # How do I interpret this?
lse = coef(lasso, s = "lambda.1se")
plot(lasso$glmnet.fit, xvar = "lambda")
abline(v=log(c(lasso$lambda.min, lasso$lambda.1se)), lty =2)

selected = min[min[,1]!=0,]
# Do we keep all these?

selectedNames = c(names(selected)[-1],"shares")

collapsedNames = paste(selectedNames, collapse= " ")
existsIn = function (item, array, arrayAsString){
  if(item %in% array | grepl(item, arrayAsString)){
    return(item)
  }
}
newSelectedNames = sapply(colnames(training_dataset), existsIn, selectedNames,
collapsedNames)
newSelectedNames = newSelectedNames[!sapply(newSelectedNames, is.null)]

library(dplyr)
lassoModel = lm(shares ~ ., data = select(training_dataset,names(newSelectedNames)))
summary(lassoModel)
# As we can see, the Adjusted R^2 is 0.01282 which is still very low and that means that our
model
```

```
# isn't going to fit at all on our data but at least it is better than the previous one


# Prepare for backwards AIC

selectedLasso = c(names(lassoModel$coefficients), "shares")

collapsedLassoNames = paste(selectedLasso, collapse= " ")

newLassoSelectedNames = sapply(colnames(training_dataset), existsIn, selectedLasso, collapsedLassoNames)

newLassoSelectedNames = newLassoSelectedNames[!sapply(newLassoSelectedNames, is.null)]


#aicBaseModel = lm(shares ~ ., data = select(training_dataset,names(newLassoSelectedNames)))

#aicModel = step(aicBaseModel, direction='both')

#summary(aicModel)


aicModel = step(model, direction='both')

summary(aicModel)


plot(aicModel, which = 2)

shapiro.test(rstandard(aicModel))

# p value very small -> Linearity rejected


vif(aicModel)

round(vif(aicModel),2)


model5 <- lm(shares~n_unique_tokens+n_non_stop_unique_tokens+num_hrefs+data_channel_is_entertainment+data_channel_is_bus+data_channel_is_tech+kw_max_max+kw_avg_max+kw_avg_avg+self_reference_avg_sharess+LDA_02+global_rate_positive_words+title_sentiment_polarity+abs_title_sentiment_polarity-1,data=training_dataset)

summary(model5)

1-sum(model5$res^2)/((n-1)*var(training_dataset$shares))

round(vif(model5),2)
```

```
# Attempt non linear transformations

# Exponential model

selectedNames = c(names(aicModel$coefficients),"shares")


getCleanModelColumns = function(model, testData){


  selectedLasso = c(names(model$coefficients), "shares")

  collapsedLassoNames = paste(selectedLasso, collapse= " ")

  newLassoSelectedNames = sapply(colnames(testData), existsIn, selectedLasso,
collapsedLassoNames)

  newLassoSelectedNames = newLassoSelectedNames[!sapply(newLassoSelectedNames,
is.null)]


  return(select(testData,names(newLassoSelectedNames)))


}


baseDataExpo = getCleanModelColumns(aicModel, training_dataset)

str(baseDataExpo)

baseDataExpo$shares = log(training_dataset$shares + 1 )



expoModel = lm(log(shares)~., data = baseDataExpo)

summary(expoModel)

round(vif(expoModel),2)


aicModel2 = step(expoModel, direction='both')

summary(aicModel2)

1-sum(aicModel2$res^2)/((n-1)*var(log(baseDataExpo$shares)))

round(vif(aicModel2),2)
```

```
expoModel2 <-
lm(log(shares)~n_unique_tokens+n_non_stop_unique_tokens+num_hrefs+data_channel_is_l
ifestyle+data_channel_is_entertainment+data_channel_is_bus+kw_max_max+kw_avg_avg+
self_reference_min_shares+self_reference_avg_sharess+weekday_is_wednesday+LDA_02+
global_rate_positive_words+abs_title_sentiment_polarity,data=baseDataExpo)

summary(expoModel2)

1-sum(expoModel2$res^2)/((n-1)*var(log(baseDataExpo$shares)))

round(vif(expoModel2),2)

residualPlots(expoModel2,plot=F)

assumptionsTests(expoModel2)



quadModel <-
lm(log(shares)~n_unique_tokens+n_non_stop_unique_tokens+num_hrefs+data_channel_is_l
ifestyle

+data_channel_is_entertainment+data_channel_is_bus+kw_max_max+kw_avg_avg

+self_reference_avg_sharess+weekday_is_wednesday+LDA_02+global_rate_positive_words
        +abs_title_sentiment_polarity+I(kw_avg_avg^2)+I(self_reference_avg_sharess^2)
        ,data=baseDataExpo)

summary(quadModel)

round(quadModel$coefficients, 3)

# As we can see, the Adjusted R^2 is 0.1433 which is the best one that we could find so we
keep it for our predictions

1-sum(quadModel$res^2)/((n-1)*var(log(baseDataExpo$shares)))

round(vif(quadModel),2)

# I guess we have kind of normality

residualPlots(quadModel,plot=F)

assumptionsTests(quadModel)



library(caret)
```

```
ctrl <- trainControl(method = "cv", number = 10)


# 10 fold validation in train sample
model <-
train(log(shares)~n_unique_tokens+n_non_stop_unique_tokens+num_hrefs+data_channel_is_lifestyle
        +data_channel_is_entertainment+data_channel_is_bus+kw_max_max+kw_avg_avg

+self_reference_avg_sharess+weekday_is_wednesday+LDA_02+global_rate_positive_words
        +abs_title_sentiment_polarity+I(kw_avg_avg^2)+I(self_reference_avg_sharess^2)
        ,data=baseDataExpo, method = "lm", trControl = ctrl)
print(model)
model$finalModel
model$resample


test_dataset <- read.csv(file = "OnlineNewsPopularity_test.csv",header = TRUE, sep = ";", dec = ",")
test_dataset <- test_dataset[,!names(test_dataset) %in% c("id", "url", "timedelta", "is_weekend")]
test_dataset$data_channel_is_lifestyle <- as.factor(test_dataset$data_channel_is_lifestyle)
levels(test_dataset$data_channel_is_lifestyle) <- list("No" = "0", "Yes" = "1")
test_dataset$data_channel_is_entertainment <-
as.factor(test_dataset$data_channel_is_entertainment)
levels(test_dataset$data_channel_is_entertainment) <- list("No" = "0", "Yes" = "1")
test_dataset$data_channel_is_bus <- as.factor(test_dataset$data_channel_is_bus)
levels(test_dataset$data_channel_is_bus) <- list("No" = "0", "Yes" = "1")
test_dataset$data_channel_is_socmed <- as.factor(test_dataset$data_channel_is_socmed)
levels(test_dataset$data_channel_is_socmed) <- list("No" = "0", "Yes" = "1")
test_dataset$data_channel_is_tech <- as.factor(test_dataset$data_channel_is_tech)
levels(test_dataset$data_channel_is_tech) <- list("No" = "0", "Yes" = "1")
test_dataset$data_channel_is_world <- as.factor(test_dataset$data_channel_is_world)
```

```
levels(test_dataset$data_channel_is_world) <- list("No" = "0", "Yes" = "1")
test_dataset$weekday_is_monday <- as.factor(test_dataset$weekday_is_monday)
levels(test_dataset$weekday_is_monday) <- list("No" = "0", "Yes" = "1")
test_dataset$weekday_is_tuesday <- as.factor(test_dataset$weekday_is_tuesday)
levels(test_dataset$weekday_is_tuesday) <- list("No" = "0", "Yes" = "1")
test_dataset$weekday_is_wednesday <- as.factor(test_dataset$weekday_is_wednesday)
levels(test_dataset$weekday_is_wednesday) <- list("No" = "0", "Yes" = "1")
test_dataset$weekday_is_thursday <- as.factor(test_dataset$weekday_is_thursday)
levels(test_dataset$weekday_is_thursday) <- list("No" = "0", "Yes" = "1")
test_dataset$weekday_is_friday <- as.factor(test_dataset$weekday_is_friday)
levels(test_dataset$weekday_is_friday) <- list("No" = "0", "Yes" = "1")
test_dataset$weekday_is_saturday <- as.factor(test_dataset$weekday_is_saturday)
levels(test_dataset$weekday_is_saturday) <- list("No" = "0", "Yes" = "1")
test_dataset$weekday_is_sunday <- as.factor(test_dataset$weekday_is_sunday)
levels(test_dataset$weekday_is_sunday) <- list("No" = "0", "Yes" = "1")

baseDataExpotest = getCleanModelColumns(quadModel, test_dataset)
str(baseDataExpotest)
baseDataExpotest$shares = log(test_dataset$shares + 1 )

# 10 fold validation in test sample
modelvalidation <-
train(log(shares)~n_unique_tokens+n_non_stop_unique_tokens+num_hrefs+data_channel_is
_lifestyle
        +data_channel_is_entertainment+data_channel_is_bus+kw_max_max+kw_avg_avg

+self_reference_avg_sharess+weekday_is_wednesday+LDA_02+global_rate_positive_words

+abs_title_sentiment_polarity+I(kw_avg_avg^2)+I(self_reference_avg_sharess^2),data=base
DataExpotest, method = "lm", trControl = ctrl)
print(modelvalidation)
modelvalidation$finalModel
```

```
modelvalidation$resample


# Predictive Power
predict(quadModel,newdata=baseDataExpotest)
predict(quadModel,newdata=baseDataExpotest, interval = 'confidence')
```