Regression and ANN/DL

Adesh Gadge

29 April 2019

# Section 1: Data preparation

## Dependencies

#### loading required libraries

library(caret)  
library(rpart)  
library(keras)  
library(MASS)  
library(recipes)  
library(MLmetrics)

## Data

#### Having look at the data

df <- read.csv("data.csv")  
#head(df)  
sum(!complete.cases(df))

## [1] 0

str(df)

## 'data.frame': 480 obs. of 17 variables:  
## $ gender : Factor w/ 2 levels "F","M": 2 2 2 2 2 1 2 2 1 1 ...  
## $ NationalITy : Factor w/ 14 levels "Egypt","Iran",..: 5 5 5 5 5 5 5 5 5 5 ...  
## $ PlaceofBirth : Factor w/ 14 levels "Egypt","Iran",..: 5 5 5 5 5 5 5 5 5 5 ...  
## $ StageID : Factor w/ 3 levels "HighSchool","lowerlevel",..: 2 2 2 2 2 2 3 3 3 3 ...  
## $ GradeID : Factor w/ 10 levels "G-02","G-04",..: 2 2 2 2 2 2 5 5 5 5 ...  
## $ SectionID : Factor w/ 3 levels "A","B","C": 1 1 1 1 1 1 1 1 1 2 ...  
## $ Topic : Factor w/ 12 levels "Arabic","Biology",..: 8 8 8 8 8 8 9 9 9 8 ...  
## $ Semester : Factor w/ 2 levels "F","S": 1 1 1 1 1 1 1 1 1 1 ...  
## $ Relation : Factor w/ 2 levels "Father","Mum": 1 1 1 1 1 1 1 1 1 1 ...  
## $ raisedhands : int 15 20 10 30 40 42 35 50 12 70 ...  
## $ VisITedResources : int 16 20 7 25 50 30 12 10 21 80 ...  
## $ AnnouncementsView : int 2 3 0 5 12 13 0 15 16 25 ...  
## $ Discussion : int 20 25 30 35 50 70 17 22 50 70 ...  
## $ ParentAnsweringSurvey : Factor w/ 2 levels "No","Yes": 2 2 1 1 1 2 1 2 2 2 ...  
## $ ParentschoolSatisfaction: Factor w/ 2 levels "Bad","Good": 2 2 1 1 1 1 1 2 2 2 ...  
## $ StudentAbsenceDays : Factor w/ 2 levels "Above-7","Under-7": 2 2 1 1 1 1 1 2 2 2 ...  
## $ Class : Factor w/ 3 levels "H","L","M": 3 3 2 2 3 3 2 3 3 3 ...

# Section 2: Oridinal Logistic Regression

## Data Preprocessing

##### Here we set the order of the class variable and also scale numeric independent variable and split data into training and testing dataframes.

df$Class <- factor(df$Class, levels=c("L","M","H"), ordered=TRUE)  
df$raisedhands <- scale(df$raisedhands, center = TRUE, scale = TRUE)  
df$VisITedResources <- scale(df$VisITedResources , center = TRUE, scale = TRUE)  
df$AnnouncementsView <- scale(df$AnnouncementsView , center = TRUE, scale = TRUE)  
df$Discussion <- scale(df$Discussion , center = TRUE, scale = TRUE)  
trainingRows <- sample(1:nrow(df), 0.7 \* nrow(df))  
training\_df <- df[trainingRows, ]  
testing\_df<-df[-trainingRows,]  
test\_df <- df[-trainingRows,1:length(df)-1 ]

## Model Building

##### Now, building the ordinal logistic regression and observing the summary of the model.

start\_OLR <-Sys.time()  
options(contrasts = c("contr.treatment", "contr.poly"))  
polrMod <- polr(Class ~ ., data=training\_df,Hess= TRUE)  
print(Sys.time() - start\_OLR)

## Time difference of 0.1585619 secs

summary(polrMod)

## Call:  
## polr(formula = Class ~ ., data = training\_df, Hess = TRUE)  
##   
## Coefficients:  
## Value Std. Error t value  
## genderM -1.00129 3.681e-01 -2.720e+00  
## NationalITyIran 2.05631 1.601e+00 1.285e+00  
## NationalITyIraq 2.45516 1.404e+00 1.749e+00  
## NationalITyJordan -0.60389 1.966e+00 -3.072e-01  
## NationalITyKW -4.29457 1.981e+00 -2.168e+00  
## NationalITylebanon -0.33742 2.862e+00 -1.179e-01  
## NationalITyLybia -31.85379 2.193e-13 -1.453e+14  
## NationalITyMorocco 0.04120 1.851e+00 2.225e-02  
## NationalITyPalestine -1.74544 2.113e+00 -8.261e-01  
## NationalITySaudiArabia -0.16435 2.232e+00 -7.363e-02  
## NationalITySyria -34.39582 1.072e+00 -3.208e+01  
## NationalITyTunis -4.40792 3.264e+00 -1.350e+00  
## NationalITyUSA 15.28753 1.373e-04 1.113e+05  
## NationalITyvenzuela 13.53840 8.030e-05 1.686e+05  
## PlaceofBirthJordan 2.14777 1.916e+00 1.121e+00  
## PlaceofBirthKuwaIT 5.90939 2.100e+00 2.815e+00  
## PlaceofBirthlebanon 1.62784 2.746e+00 5.929e-01  
## PlaceofBirthPalestine -0.35191 2.152e+00 -1.635e-01  
## PlaceofBirthSaudiArabia 3.85023 2.041e+00 1.887e+00  
## PlaceofBirthSyria 38.19873 1.072e+00 3.563e+01  
## PlaceofBirthTunis 6.27363 3.406e+00 1.842e+00  
## PlaceofBirthUSA 0.50057 2.082e+00 2.404e-01  
## StageIDlowerlevel 2.02669 2.154e+00 9.409e-01  
## StageIDMiddleSchool 21.93125 1.448e+00 1.515e+01  
## GradeIDG-04 1.80092 8.941e-01 2.014e+00  
## GradeIDG-05 -21.99777 1.808e-08 -1.217e+09  
## GradeIDG-06 -19.85424 7.470e-01 -2.658e+01  
## GradeIDG-07 -20.21037 7.101e-01 -2.846e+01  
## GradeIDG-08 -19.86496 8.824e-01 -2.251e+01  
## GradeIDG-09 0.48040 2.539e+00 1.892e-01  
## GradeIDG-10 0.40315 3.636e+00 1.109e-01  
## GradeIDG-11 4.34183 2.387e+00 1.819e+00  
## SectionIDB 0.19898 4.314e-01 4.613e-01  
## SectionIDC 0.54683 8.585e-01 6.370e-01  
## TopicBiology 0.29294 1.133e+00 2.586e-01  
## TopicChemistry -0.96224 1.116e+00 -8.621e-01  
## TopicEnglish 1.89798 1.248e+00 1.521e+00  
## TopicFrench 0.04861 7.329e-01 6.633e-02  
## TopicGeology -1.26909 1.002e+00 -1.266e+00  
## TopicHistory -1.52121 1.029e+00 -1.479e+00  
## TopicIT 0.12267 8.245e-01 1.488e-01  
## TopicMath 1.54539 1.060e+00 1.458e+00  
## TopicQuran 0.33512 1.083e+00 3.096e-01  
## TopicScience -1.17371 9.423e-01 -1.246e+00  
## TopicSpanish -0.47174 9.663e-01 -4.882e-01  
## SemesterS 0.02745 3.767e-01 7.287e-02  
## RelationMum 1.43011 4.069e-01 3.515e+00  
## raisedhands 0.65229 2.539e-01 2.569e+00  
## VisITedResources 0.98603 2.794e-01 3.529e+00  
## AnnouncementsView 0.43365 2.364e-01 1.834e+00  
## Discussion 0.34227 1.882e-01 1.819e+00  
## ParentAnsweringSurveyYes 1.29098 4.203e-01 3.072e+00  
## ParentschoolSatisfactionGood 0.29603 4.312e-01 6.865e-01  
## StudentAbsenceDaysUnder-7 3.24695 4.533e-01 7.163e+00  
##   
## Intercepts:  
## Value Std. Error t value   
## L|M 3.603800e+00 2.493100e+00 1.445500e+00  
## M|H 9.156300e+00 2.566900e+00 3.567000e+00  
##   
## Residual Deviance: 309.6198   
## AIC: 421.6198

## Model Interpretations

#### The categorical variables like TopicEnglish can be interpreted as: a student with topic Eng, as opposed to a base Topic student, is associated with a higher likelihood of having a higher performance. The t-value is greater than 2 and therefore is statistically significant at the 5% level

#### The continuous variables like raised hands can be interpreted as : with one unit increase in raisedhands the log of odds of having a higher student performance increases by 0.73924

#### Intercepts:

#### L|M: Log of odds of having student performance ‘Low’ versus having student performance ‘Medium’ or ‘High’ = 2.5881

#### M|H: Log of odds of having student performance ‘Medium’ versus having student performance ‘High’ = 9.1711

# Section 3: Artificial Neural Network

## Data Preprocessing

##### 

rec\_obj <- recipe(Class ~ ., data =df) %>%   
 # step\_discretize(tenure, options = list(cuts = 6)) %>%   
 #step\_log(TotalCharges) %>%   
 step\_dummy(all\_nominal(), -all\_outcomes()) %>%   
 step\_center(all\_predictors(), -all\_outcomes()) %>%   
 step\_scale(all\_predictors(), -all\_outcomes()) %>%   
 prep(data = df)

##### 

x\_train\_tbl <- bake(rec\_obj, newdata = training\_df)  
x\_test\_tbl <- bake(rec\_obj, newdata = testing\_df)   
y\_train <- as.numeric(unlist(x\_train\_tbl[,5])) -1  
y\_test <- as.numeric(unlist(x\_test\_tbl[,5])) -1  
x\_train\_tbl <-x\_train\_tbl[,-5]  
x\_test\_tbl <-x\_test\_tbl[,-5]

##### Encoding the target variable for Neural Network

library(keras)  
# One hot encode train target values  
trainLabels <- to\_categorical(y\_train)

## Warning in normalizePath(path.expand(path), winslash, mustWork):  
## path[1]="C:\Users\Adesh\ANACON~1\envs\tensorflow-gpu/python.exe": The  
## system cannot find the file specified

## Warning in normalizePath(path.expand(path), winslash, mustWork):  
## path[1]="C:\Users\Adesh\ANACON~1\envs\tensorflow-gpu=3.5.4/python.exe": The  
## system cannot find the file specified

# One hot encode test target values  
testLabels <- to\_categorical(y\_test)

## Model Building

#library(keras)  
model\_keras <- keras\_model\_sequential()  
model\_keras %>%   
 layer\_dense(units = 16,  
 kernel\_initializer = "uniform",  
 activation = "relu",  
 input\_shape = ncol(x\_train\_tbl)) %>%   
 layer\_dropout(rate = 0.3) %>%   
 layer\_dense(units = 8,  
 kernel\_initializer = "uniform",  
 activation = "relu") %>%   
 layer\_dropout(rate = 0.3) %>%   
 layer\_dense(units = 3,  
 kernel\_initializer = "uniform",  
 activation = "softmax") %>%   
 compile(optimizer = "adam",  
 loss = "categorical\_crossentropy",  
 metrics = c("accuracy")  
 )

##### Converting the input data into matrix

x\_train\_tbl <-as.matrix(x\_train\_tbl)

##### Fitting the data

start\_ANN1 <-Sys.time()  
history<-model\_keras %>% fit(  
 x\_train\_tbl,   
 trainLabels,   
 epochs = 150,   
 batch\_size = 5,   
 validation\_split = 0.2  
 )  
ANN1\_time<-Sys.time() - start\_ANN1

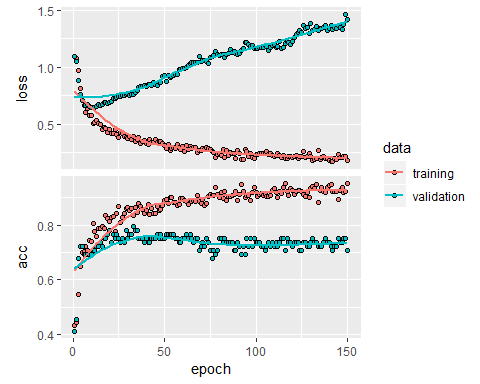
##### Time take by this model

print(ANN1\_time)

## Time difference of 22.48052 secs

##### Accuracy vs Epochs

plot(history)



##### model seems to overfit because the validation loss is low. Let’s try with L1 and L2 regularizations.

#library(keras)  
model\_keras2 <- keras\_model\_sequential()  
model\_keras2 %>%   
 layer\_dense(units = 16,  
 kernel\_initializer = "uniform",  
 activation = "relu",  
 input\_shape = ncol(x\_train\_tbl),  
 kernel\_regularizer=regularizer\_l2(0.01)) %>%   
 layer\_dropout(rate = 0.3) %>%   
 layer\_dense(units = 8,  
 kernel\_initializer = "uniform",  
 activation = "relu",  
 kernel\_regularizer=regularizer\_l2(0.01))%>%   
 layer\_dropout(rate = 0.3) %>%   
 layer\_dense(units = 3,  
 kernel\_initializer = "uniform",  
 activation = "softmax",  
 kernel\_regularizer=regularizer\_l2(0.01)) %>%   
 compile(optimizer = "adam",  
 loss = "categorical\_crossentropy",  
 metrics = c("accuracy")  
 )

##### Fitting the data

start\_ANN2 <-Sys.time()  
history2<-model\_keras2 %>% fit(  
 x\_train\_tbl,   
 trainLabels,   
 epochs = 150,   
 batch\_size = 5,   
 validation\_split = 0.2  
 )  
ANN2\_time<-Sys.time() - start\_ANN2

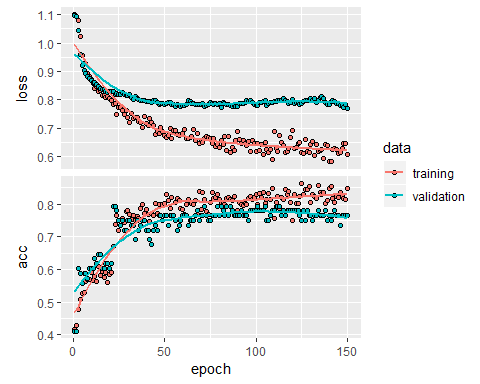
##### Time take by this model

print(ANN2\_time)

## Time difference of 22.51359 secs

##### Accuracy vs Epochs

plot(history2)



##### Still looks like overfitting, now increasing the regularization parameter

#library(keras)  
model\_keras3 <- keras\_model\_sequential()  
model\_keras3 %>%   
 layer\_dense(units = 16,  
 kernel\_initializer = "uniform",  
 activation = "relu",  
 input\_shape = ncol(x\_train\_tbl),  
 kernel\_regularizer=regularizer\_l2(0.015)) %>%   
 layer\_dropout(rate = 0.3) %>%   
 layer\_dense(units = 8,  
 kernel\_initializer = "uniform",  
 activation = "relu",  
 kernel\_regularizer=regularizer\_l2(0.015))%>%   
 layer\_dropout(rate = 0.3) %>%   
 layer\_dense(units = 3,  
 kernel\_initializer = "uniform",  
 activation = "softmax",  
 kernel\_regularizer=regularizer\_l2(0.015)) %>%   
 compile(optimizer = "adam",  
 loss = "categorical\_crossentropy",  
 metrics = c("accuracy")  
 )

##### Fitting the data

start\_ANN3 <-Sys.time()  
history3<-model\_keras3 %>% fit(  
 x\_train\_tbl,   
 trainLabels,   
 epochs = 150 ,   
 batch\_size = 5,   
 validation\_split = 0.2  
 )  
ANN3\_time<-Sys.time() - start\_ANN3

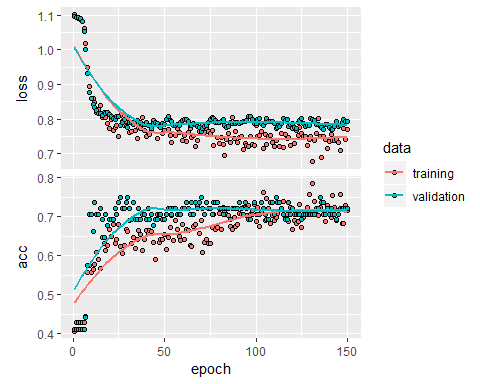
##### Time take by this model

print(ANN3\_time)

## Time difference of 22.65951 secs

##### Accuracy vs Epochs

plot(history3)

 ##### With our best model with validation, confusion matrix for test data and accuracy

classes <- model\_keras3 %>% predict\_classes(as.matrix(x\_test\_tbl))  
  
# Confusion matrix  
table(y\_test, classes)

## classes  
## y\_test 0 1 2  
## 0 33 4 0  
## 1 7 45 16  
## 2 0 12 27

#accuracy  
print('ACCURACY: ')

## [1] "ACCURACY: "

print(Accuracy(y\_test, classes))

## [1] 0.7291667

# Section 4: Random Forest

##### The models made above could not learn from the data that well, I think because neural network requires large amount of data to learn from. Random Forest on other hand are bagging of trees which can learn from lesser amount of data.

library(caret)  
library(rpart)  
start2 <-Sys.time()  
tunegrid <- expand.grid(.mtry=c(1:5))  
model\_rf <- train(Class ~ ., data =training\_df, method = "rf",  
 metric = "Accuracy",  
 tuneGrid=tunegrid  
 )  
Sys.time() - start2

## Time difference of 46.24644 secs

predict\_rf <- predict(model\_rf, newdata = test\_df)  
confusionMatrix(predict\_rf, testing\_df$Class)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction L M H  
## L 33 6 0  
## M 4 53 9  
## H 0 9 30  
##   
## Overall Statistics  
##   
## Accuracy : 0.8056   
## 95% CI : (0.7314, 0.8667)  
## No Information Rate : 0.4722   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.6965   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: L Class: M Class: H  
## Sensitivity 0.8919 0.7794 0.7692  
## Specificity 0.9439 0.8289 0.9143  
## Pos Pred Value 0.8462 0.8030 0.7692  
## Neg Pred Value 0.9619 0.8077 0.9143  
## Prevalence 0.2569 0.4722 0.2708  
## Detection Rate 0.2292 0.3681 0.2083  
## Detection Prevalence 0.2708 0.4583 0.2708  
## Balanced Accuracy 0.9179 0.8042 0.8418

##### Variable Importance is as followed.

varImp(model\_rf)

## rf variable importance  
##   
## only 20 most important variables shown (out of 60)  
##   
## Overall  
## VisITedResources 100.000  
## raisedhands 85.055  
## AnnouncementsView 68.054  
## StudentAbsenceDaysUnder-7 66.868  
## Discussion 53.637  
## RelationMum 38.128  
## ParentAnsweringSurveyYes 26.526  
## ParentschoolSatisfactionGood 16.346  
## genderM 15.307  
## SectionIDB 9.812  
## NationalITyKW 9.773  
## PlaceofBirthJordan 8.767  
## StageIDlowerlevel 8.102  
## PlaceofBirthKuwaIT 7.932  
## StageIDMiddleSchool 7.585  
## NationalITyJordan 7.579  
## SemesterS 7.484  
## TopicIT 7.154  
## TopicFrench 6.041  
## GradeIDG-08 5.945

# Section 5: Model Comparisons

### Highest Accuracy: random forest could achieve the highest accuracy, of around 76%(test accuracy),compared to the other two models, neural networks did come close to 75% but neural networks require lots of data to become very good.

### Faster Running speeds: The logsitic regression is the fastes to run because it doesn’t have many parameters to learn, As compared to neural network which takes the longest to train and random forest is in-between.