MVA Final Project

Javier Ferrando Monsonis
Marcel Porta Valles
Mehmet Fatih ??agil

February 20, 2018

```r
library(magrittr)
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(stringr)
library(ggplot2)

library(data.table)
```

```
##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```r
library(dplyr)
library(magrittr)
library(ggplot2)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(ggExtra)
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```r
library(factoextra)
```

```
## Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at https://goo.gl/13EFCZ
```

```r
library(stringr)
library(FactoMineR)
```

```r
#library(kableExtra)
library(knitr)
```

```r
training_set <- train[,-(1:4)]
col_order <- colnames(training_set)

test_set<- d_ss[,-1][,-(2:4)]
colnames(test_set)[1] <- "team1win"
test_set <- test_set[, col_order]
training_set$elo_diff <- NULL
test_set$elo_diff <- NULL
training_set$diff_rank <- NULL
test_set$diff_rank <- NULL
```

```r
kable(training_set[sample(nrow(train), 6), ][,1:10])
```

	team1win	t1_rank_n	t2_rank_n	t1_season_elo	t2_season_elo	elo_prob_1	t1_mpie	t2_mpie	t1
356	1	3	6	1856.137	1887.825	0.4545238	0.6262022	0.6137100	17.
685	0	6	3	1801.625	1926.582	0.3275457	0.5565142	0.5550160	12.
245	0	3	2	1959.413	1885.683	0.6045406	0.5616776	0.6209522	7.
419	1	2	7	2041.983	1914.979	0.6750457	0.6355244	0.5715324	25.
290	0	4	5	1879.756	1726.106	0.7077496	0.6142832	0.6469077	16.
454	1	2	15	1894.235	1575.011	0.8626647	0.6026319	0.5514985	18.

```r
#Train model with train data
#Add predictions to dss
#Merge d_ss with test_outcome_tournament (games that occurred) -> validation
#validation has target and Pred for every game that occurered 2014-2018
#Apply LogLoss to validation$Pred and validation$team1win


# #logistic regression model: differences
# model <- glm(team1win ~
#                 diff_rank +
#                 t1_rank_n +
#                 #t2_rank_n +
#                 t1_season_elo +
#                 t2_season_elo +
#                 #elo_prob_1 +
#                 t1_mpie +
#                 t2_mpie +
#                 t1_netrtg +
#                 t2_netrtg
#               ,
#
#               data = training_set, family = binomial)

model <- glm(team1win ~ .
             ,

             data = training_set, family = binomial)
```

```r
#Predict on every possible matchup
predict <- data.frame(Pred = predict(model, newdata = test_set, type = 'response'))
d_ss <- d_ss %>% mutate(Pred = predict$Pred)# %>% dplyr::select(ID, Pred) Change sample submission pred

d_ss_fin <- sample_submission %>% mutate(Pred = d_ss$Pred) #only matchup and prediction -> Results for
#write.csv(d_ss_fin, "submission_stage_2.csv", row.names = FALSE)

summary(model)
```

```
##
## Call:
## glm(formula = team1win ~ ., family = binomial, data = training_set)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3242  -0.8802   0.2700   0.8947   2.5987
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -7.206253   4.727048  -1.524  0.12739
## t1_rank_n     -0.089333   0.039574  -2.257  0.02398 *
## t2_rank_n      0.112791   0.041428   2.723  0.00648 **
## t1_season_elo  0.003624   0.003197   1.133  0.25708
## t2_season_elo -0.002286   0.003169  -0.721  0.47082
## elo_prob_1    -0.774010   2.608352  -0.297  0.76666
## t1_mpie        6.509649   4.661552   1.396  0.16258
## t2_mpie        2.520763   4.635233   0.544  0.58656
## t1_netrtg      0.001556   0.018493   0.084  0.93293
## t2_netrtg     -0.021832   0.018828  -1.160  0.24624
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 988.02  on 712  degrees of freedom
## Residual deviance: 764.18  on 703  degrees of freedom
## AIC: 784.18
##
## Number of Fisher Scoring iterations: 5
```

```r
kable(summary(model)$coefficients)
```

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-7.2062527	4.7270477	-1.5244722	0.1273908
t1_rank_n	-0.0893334	0.0395736	-2.2574005	0.0239831
t2_rank_n	0.1127913	0.0414276	2.7226099	0.0064768
t1_season_elo	0.0036236	0.0031973	1.1333294	0.2570760
t2_season_elo	-0.0022855	0.0031693	-0.7211487	0.4708180
elo_prob_1	-0.7740098	2.6083520	-0.2967428	0.7666628
t1_mpie	6.5096493	4.6615522	1.3964553	0.1625774
t2_mpie	2.5207630	4.6352331	0.5438266	0.5865608
t1_netrtg	0.0015563	0.0184935	0.0841542	0.9329338
t2_netrtg	-0.0218315	0.0188280	-1.1595228	0.2462432

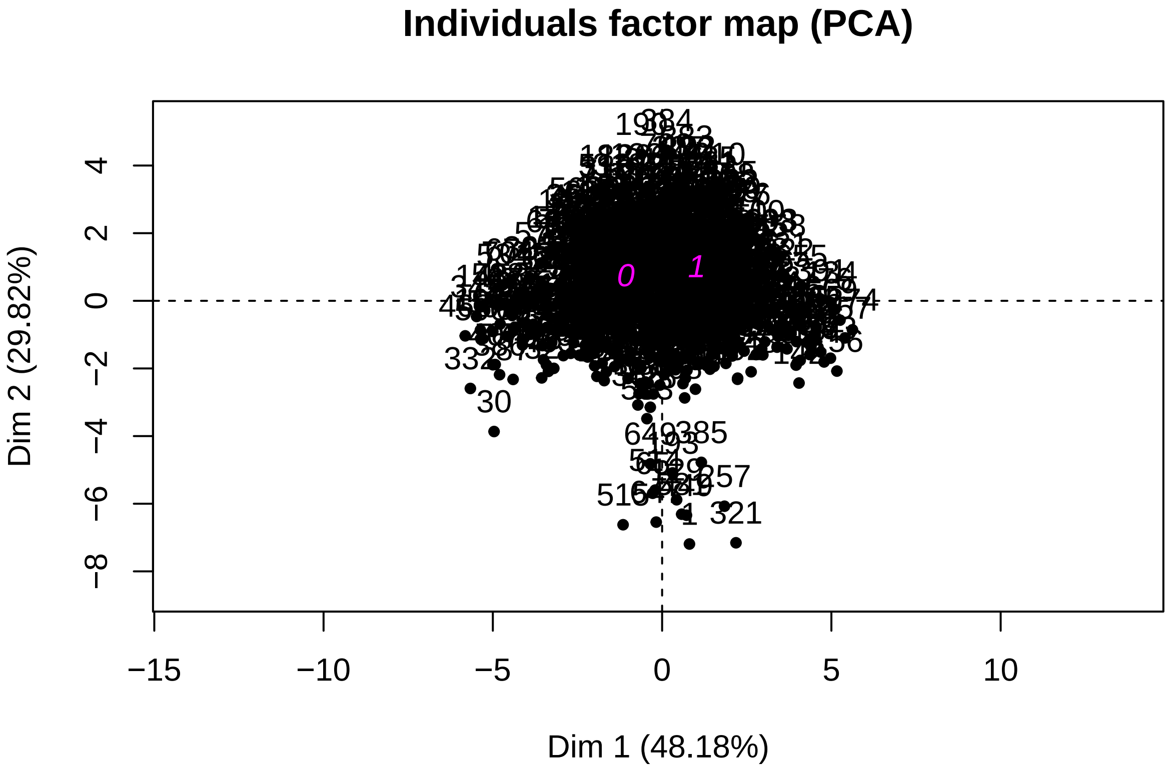
```r
#Merge every possible matchup result predictions with real games and check test error
test_result <- merge(x = test_outcome_tournament, y = d_ss[2:5], by=c("team1id","team2id","season"), al

library(MLmetrics)
```

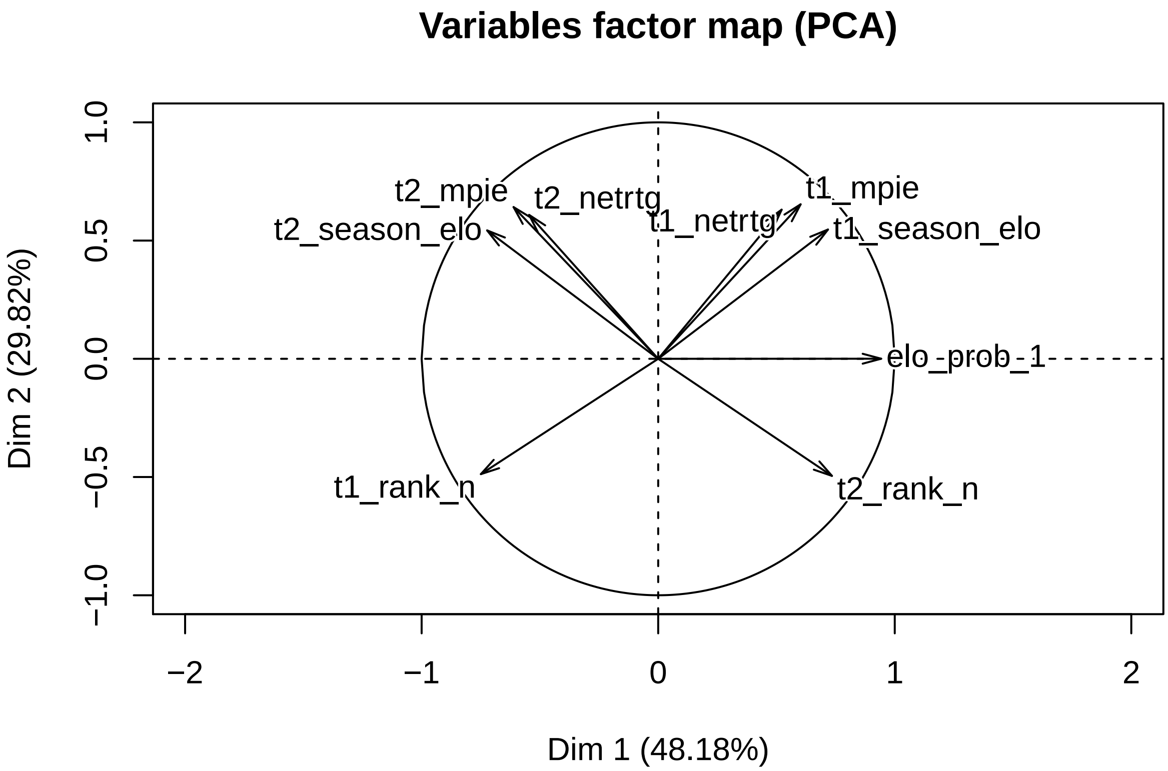
```
##
## Attaching package: 'MLmetrics'

## The following object is masked from 'package:base':
##
##     Recall
```

```r
#library(forecast)

LogLoss(y_pred = test_result$Pred, y_true = test_result$team1win)
```

```
## [1] 0.5589786
```

```r
pca_ncaa <- PCA(training_set, quali.sup = 1, scale.unit = TRUE,  graph = TRUE)
```






```
# #Regularized Logistic Regression
# #Total fail-> predictions wrong
#
# set.seed(123)
# library(glmnet)
# x <- model.matrix(team1win~., training_set)
# y <- training_set$team1win
# cv.lasso <- cv.glmnet(x, y , alpha = 1, family = "binomial")
# # Fit the final model on the training data
# model <- glmnet(x, y, alpha = 1, family = "binomial",
#                 lambda = cv.lasso$lambda.min)
# plot(cv.lasso)
# # Display regression coefficients
# coef(model)
#
# #### glmnet test
#
# x.test <- model.matrix(team1win~., test_set)
#
# probabilities <- model %>% predict(newx = x.test)
# ```
#
# ```{r}
# #Neural Network
# #Scale inputs
# library(nnet)
# library(caret)
# library(neuralnet)
#
```

```
#
# train_nnet_scaled <- as.data.frame(scale(training_set[-1]))
# train_nnet_scaled <- cbind(training_set$team1win,train_nnet_scaled)
# colnames(train_nnet_scaled)[1] <- "team1win"
# #train_nnet_scaled$team1win<- factor(train_nnet_scaled$team1win, labels=c(0,1))#not for neuralnet
#
# #nn1 <- nnet(team1win~t1_rank_n+t2_rank_n+elo_diff,data=train_nnet_scaled,entropy=T,size=100,decay=0,
# #predict <- data.frame(Pred = predict(nn1, newdata = d_ss_nnet_scaled))
#
# #10 fold cv trial
#
# ## We first split the available data into learning and test sets, selecting randomly 2/3 and 1/3 of t
# ## We do this for a honest estimation of prediction performance
#
# names <- colnames(train_nnet_scaled)[-1] #choose the names you want
# a <- as.formula(paste('team1win ~ ' ,paste(names,collapse='+')))
# a
# #neuralnet DOESN'T need factors as target
# nn <- neuralnet(a, data=train_nnet_scaled, hidden=c(1), linear.output=FALSE, threshold=0.01)
# nn$result.matrix
# plot(nn)
#
#
#
# #nnet by means of train function, needs factors as target
# # model <- train(team1win~., data=train_nnet_scaled, method='nnet', maxit = 300,
# #                  trControl=trainControl(method='cv'))
# test_set_scaled <- scale(test_set)
#
# #predict <- data.frame(Pred = predict (model, newdata=d_ss_nnet_scaled, type="prob"))
#
# test_set_scaled <- subset(test_set_scaled, select = colnames(test_set_scaled)[-1])
# nn.results <- compute(nn, test_set_scaled)
#
# #train in nnet prediction
# #d_ss <- d_ss %>% mutate(Pred = predict$Pred.1)# %>% dplyr::select(ID, Pred) Change sample submission
#
# #neuralnet prediction
# #d_ss$Pred <- NULL
# d_ss <- d_ss %>% mutate(Pred = as.numeric(nn.results$net.result))
#
#
#
#
#
# # d_ss_fin <- sample_submission %>% mutate(Pred = d_ss$Pred) #only matchup and prediction -> Results
# # #write.csv(d_ss_fin, "submission_stage_2.csv", row.names = FALSE)
# #
# #
# # #############
# # set.seed(43)
# # N <- nrow(train_nnet_scaled)
# # learn <- sample(1:N, round(2*N/3))
```

```
# # (sizes <- 2*seq(1,10,by=2)) #different sizes
# #
# # ## specify 10x10 CV
# # trc <- trainControl (method="repeatedcv", number=10, repeats=10)
# #
# # model.10x10CV <- train (team1win ~., data = train_nnet_scaled, subset=learn, method='nnet', maxit =
#
#
# ```
#
# ```{r}
# #We can only test with 2014-2018 data
# #Merge every possible matchup result predictions with real games and check test error
# test_result <- merge(x = test_outcome_tournament, y = d_ss[2:5], by=c("team1id","team2id","season"),
#
# library(MLmetrics)
# #library(forecast)
#
# LogLoss(y_pred = test_result$Pred, y_true = test_result$team1win)
#
# test_result$Pred<- factor(test_result$Pred, labels=c(0,1))#
# Accuracy(y_pred = test_result$Pred, y_true = test_result$team1win)
#
# ```
#
# ```{r,echo=FALSE}
# library(rpart)
# train_tree <- training_set
# train_tree$team1win<- factor(train_tree$team1win, labels=c(0,1))#not for neuralnet
#
# DecisionTree = rpart(team1win ~ ., data=train_tree,control=rpart.control(cp=0.001, xval=10),method='c
# printcp(DecisionTree)
#
# treeSize = DecisionTree$cptable[,2]+1 #nsplit
# treeImpurity = DecisionTree$cptable[,3] #rel error
# cvImpurity = DecisionTree$cptable[,4] #xerror
#
# plot(treeSize, treeImpurity, main="R(T)", xlab="size of the tree", ylab="Relativity Impurity", type="
# lines(treeSize, cvImpurity ,type="o", col='blue')
# legend("topright", c("All training data","CV training data"), col=c('red', 'blue'), lty=1)
# ```
#
# ```{r, echo=FALSE}
# DecisionTree$cptable = as.data.frame(DecisionTree $cptable)
# ind = which.min(DecisionTree$cptable$xerror)
# xerr <-DecisionTree$cptable$xerror[ind]
# xstd <-DecisionTree$cptable$xstd[ind]
#
# i = 1
# while (DecisionTree$cptable$xerror[i] > xerr+xstd){
#   i = i+1
# }
# #alfa = DecisionTree$cptable$CP[i]
```

```
# alfa = DecisionTree$cptable$CP[3]
#
# optimal <- prune(DecisionTree, cp=alfa)
# par(mfrow = c(1,1), xpd = NA)
# plot(optimal)
# text(optimal, use.n=T,cex=0.8,col="blue")
#
# #Tree prediction
# rpart_pred <- predict(DecisionTree,test_set,type='prob')[,1]
# rpart_pred_class <- predict(DecisionTree,test_set,type='class')
# d_ss <- d_ss %>% mutate(Pred = predict(DecisionTree,test_set,type='prob')[,1])
# #d_ss <- d_ss %>% mutate(Pred = predict(DecisionTree,test_set,type='class'))
# ```
#
# ```{r, echo=FALSE}
# library(randomForest)
# train_tree <- training_set
# train_tree$team1win<- factor(train_tree$team1win, labels=c(0,1))#not for neuralnet
#
# #Convert d_ss_tree$team1win to categorical values
# test_set_rf <- test_set
# test_set_rf$team1win <- NULL
# test_set_rf$team1win <- sample(c(0, 1), nrow(test_set_rf), replace=TRUE)
#
# test_set_rf <- test_set_rf[, col_order]
# test_set_rf$team1win<- factor(test_set_rf$team1win, labels=c(0,1))
#
# random_forest <- randomForest(formula = team1win ~.,
#                          data=train_tree,
#                          mtry=3,      # three predictor-vars selected randomly at each split
#                          xtest=test_set_rf[-1],
#                          ytest=test_set_rf$team1win,
#                          #ytest=as.factor(audit_imp$Adjusted[testRows]),
#                          importance=T,
#                          ntree=500,   # acceptably large value to ensure each sample row is predicted
#                                       # at least 2-digit nbr of times on average
#                          nodesize = 50,
#                          maxnodes = 40,
#                          norm.votes=T,
#                          keep.forest=TRUE)
#
# #rf_predictions_prob <- predict(random_forest, test_set_rf, type='prob')
# rf_predictions_class <- predict(random_forest, test_set_rf, type='class')
# #d_ss <- d_ss %>% mutate(Pred = rf_predictions_prob[,2])#For prob
# d_ss <- d_ss %>% mutate(Pred = rf_predictions_class)
# #
# # cf <- confusionMatrix(factor(df_rf_predictions), factor(rf_test$target), positive="1", dnn = c("Pre
# # draw_confusion_matrix(cf)
# ```
#
#
# ```{r, echo=FALSE}
# library(caret)
```

```
# train_tree <- train[,-(1:4)]
# #train_tree$team1win<- factor(train_tree$team1win, labels=c(0,1))
# train_tree$team1win<- factor(train_tree$team1win, labels=c("win","loss"))#not for neuralnet
#
# # Example of Bagging algorithms
# control <- trainControl(method="repeatedcv", number=10, repeats=3)
# seed <- 7
# metric <- "logLoss"
# # Bagged CART
# set.seed(seed)
# fit.treebag <- train(team1win~., data=train_tree, method="treebag", metric=metric, trControl=control)
# # Random Forest
# set.seed(seed)
# fit.rf <- train(team1win~., data=train_tree, method="rf", metric=metric, trControl=control)
# # summarize results
# bagging_results <- resamples(list(treebag=fit.treebag, rf=fit.rf))
# summary(bagging_results)
# dotplot(bagging_results)
# ```
#
# ```{r, echo=FALSE}
#
# # Example of Stacking algorithms
# # create submodels
# train_ensemble <- training_set
# train_ensemble$diff_rank <- NULL
# train_ensemble$elo_diff <- NULL
# train_ensemble$team1win<- factor(train_ensemble$team1win, labels=c("win","loss"))#not for neuralnet
#
# library(caretEnsemble)
# control <- trainControl(method="repeatedcv", number=10, repeats=10, savePredictions='all', classProbs=
# algorithmList <- c('lda', 'glm', 'svmRadial')#knn disaster
# #algorithmList <- c('rpart', 'glm','svmRadial')
# set.seed(7)
# metric <- "logLoss"
# models <- caretList(team1win~., data=train_ensemble, trControl=control, methodList=algorithmList, met
#
#
#
# greedy_ensemble <- caretEnsemble(
#   models,
#   metric="logLoss",
#   trControl=control)
# summary(greedy_ensemble)
#
# kable(modelCor(resamples(models)))
#
# summary(greedy_ensemble)
# results <- resamples(models)
# summary(results)
# dotplot(results)
# ensemble_pred <- predict(greedy_ensemble, newdata=test_set,type='prob')
# d_ss <- d_ss %>% mutate(Pred = ensemble_pred)#F
```