Online Appendix: How Cross-Validation Can Go Wrong and What to Do about it.

Contents

A Cross-Validating Cross-Validation in Political Science

The following table summarizes the results of our survey of the literature.

Author(s)	Journal
Cross-Validation for Validating New Measures/Instruments	
Weber (2011)	American Journal of Political Science
Ghitza and Gelman (2013)	American Journal of Political Science
Cantú (2014)	American Journal of Political Science
Besley and Reynal-Querol (2014)	American Political Science Review
König et al. (2013)	Political Analysis
Atkeson et al. (2014)	Political Analysis
Guess (2015)	Political Analysis
Greene and Cross (2017)	Political Analysis
Cross-Validation as a Robustness Check	
Engstrom (2012)	American Journal of Political Science
Bisbee and Larson (2017)	American Political Science Review
Cross-Validation to Estimate True Error	
Ahlquist and Wibbels (2012)	American Journal of Political Science
Grimmer et al. (2012)	American Political Science Review
Hill and Jones (2014)	American Political Science Review
Fariss (2014)	American Political Science Review
Montgomery et al. (2012)	Political Analysis
Grimmer and Stewart (2013)	Political Analysis
Montgomery et al. (2015)	Political Analysis
Caughey and Warshaw (2015)	Political Analysis
Cranmer and Desmarais (2017)	Political Analysis

Peterson and Spirling (2018)	Political Analysis
Cross-Validation for Model Tuning	
Hopkins and King (2010)	American Journal of Political Science
Boas and Hidalgo (2011)	American Journal of Political Science
Desmarais et al. (2015)	American Journal of Political Science
Abadie et al. (2015)	American Journal of Political Science
Wilkerson et al. (2015)	American Journal of Political Science
Lauderdale and Clark (2012)	American Political Science Review
Keele (2010)	Political Analysis
Carter and Signorino (2010)	Political Analysis
Imai and Strauss (2011)	Political Analysis
Glynn and Quinn (2011)	Political Analysis
Caughey and Sekhon (2011)	Political Analysis
Esarey and Pierce (2012)	Political Analysis
Keele and Minozzi (2013)	Political Analysis
Hainmueller and Hazlett (2014)	Political Analysis
Keele and Titiunik (2015)	Political Analysis
Montgomery et al. (2015)	Political Analysis
Xu (2016)	Political Analysis
Cranmer and Desmarais (2017)	Political Analysis
Other or no use of Cross-Validation	
Greenhill et al. (2011)	American Journal of Political Science
Rainey (2014)	American Journal of Political Science
Jessee (2016)	American Journal of Political Science
Beauchamp (2017)	American Journal of Political Science
Desmarais and Harden (2012)	Political Analysis
Iacus et al. (2012)	Political Analysis
Bowers et al. (2013)	Political Analysis

B Experimental Exploration of Problematic Cross-Validation

B.1 Three common performance measures

Generally, the results of a binary classifier (and any other classifier) can be summarized by a confusion matrix. In the case of binary classification this is a 2×2 table of the four possible classification outcomes of a model. The three performance measures – F_1 score, ROC-AUC and PR-AUC – can all be explained with the help of confusion matrices. To get class predictions from predicted probabilities of belonging to the positive class one has to set a threshold for positive prediction. Usually, the default value of this threshold for positive prediction is 0.5. However, any other value between 0 and 1 could be a sensible threshold for positive prediction.

Confusion Matrix

		Observed	
		Positive	*Negative*
Predicted	*Positive*	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

- F_1 **score**: The F_1 score is the harmonic mean of precision and recall at some threshold of positive prediction. In our experiment and application we set this threshold to 0.5. In terms of the confusion matrix, the F_1 score can be expressed as: $F_1 = \frac{2 \cdot TP}{2 \cdot TP + FN + FP}$

- **ROC-AUC score**: ROC-AUC stands for the *area under the receiver operating characteristic curve*. The receiver operating characteristic (ROC) curve is a plot of the true positive rate (TPR, a.k.a Recall or Sensitivity) against the false positve rate (FPR, a.k.a probability of false alarm) for confusion matrices of various thresholds. Where $TPR = \frac{TP}{TP+FN}$ (true positives divided by the sum of all observed positives) and $FPR = \frac{FP}{FP+TN}$ (false positives divided by the sum of observed negatives).

Plotting the two for various thresholds against each other yields the ROC curve. The area under the curve can than be calculated by taking the integral: ROC-AUC = $\int_{\infty}^{-\infty} TPR(T)FPR'(T)dT$ where T is the threshold parameter. A perfect classifier has a $ROC - AUC$ value of 1 and a random uninformative classifier has a $ROC - AUC$ value of 0.5. We calculate the $ROC - AUC$ values using the `roc` function from the R package `pROC` (Robin et al., 2011).

- **PR-AUC score**: PR-AUC stands for the *area under the precision recall curve*. The precision recall (PR) curve is a plot of the precision against the recall (a.k.a. TPR see above) for confusion matrices of various threshold. Where Precision = $\frac{TP}{TP+FP}$ (the ratio of correctly classified positives and all predicted positives) and Recall = $TPR = \frac{TP}{TP+FN}$. Plotting the two for various thresholds against each other yields the PR curve. Again taking the integral PR-AUC = $\int_{\infty}^{-\infty} \text{Precision}(T)\text{Recall}'(T)dT$. A perfect classifier will have a PR-AUC value of 1, the PR-AUC value of an uninformative classifier will depend on the class balance (ratio of observed Positives and observed Negatives in the data). In our experiment the true PR-AUC is 0.05. Note that since PR-AUC does not consider true negatives it is not affected by class imbalances (see Cranmer and Desmarais, 2017). We calculate the $PR - AUC$ values using the `aucPRp` function from the Replication Code provided by Cranmer and Desmarais (2016).

B.2 Code for the Experiment

```
################################################################
# Install and load Packages needed to run
################################################################

p_needed <- c("randomForest", "caret","ROCR", "pROC",
              "stepPlr", "doMC", "separationplot", "logistf")

packages <- rownames(installed.packages())
p_to_install <- p_needed[!(p_needed %in% packages)]
if (length(p_to_install) > 0) {
  install.packages(p_to_install)
}
print(sapply(p_needed, require, character.only = TRUE))

registerDoMC(cores = detectCores()-1)

source("AUC_PR.R")

confMat <- function(threshold = 0.5, pred, obs){
  if(class(obs)=="factor") obs <- as.numeric(obs)-1

  uMat <- matrix(threshold,length(obs),length(threshold),byrow=T)
  preds <- c(pred)-uMat > 0
  TP <- t(obs)%*%preds
  FP <- t(1-obs)%*%preds
  TN <- t(1-obs)%*%(1-preds)
  FN <- t(obs)%*%(1-preds)


  cmMat <- cbind(c(TP),c(FP),c(TN),c(FN))

  colnames(cmMat) <- c("tp","fp","tn","fn")

  cmMat <- data.frame(cmMat)
  #
  return(cmMat)
}
################################################################
# Experiments
################################################################

set.seed(20180503)


# prob1 indicates the distribution of 1s in the dependent variable

prob1 <- 0.05


# Generate uncorrelated experiment data

y <- rbinom(2000, 1, prob1)

x <- list(NULL)
for(i in 1:90){
  x[[i]] <- rnorm(2000)
}
```

```
X <- do.call(cbind, x)

df <- data.frame(y, X)

df$y <- as.factor(y)
levels(df$y) <- c("Negative", "Positive")


# Split into training and test data

sel <- sample(nrow(df), floor(0.25* nrow(df)))

test_data <- df[sel,]

training_data <- df[-sel,]


# Create folds for 10-fold cross-validation

nrFolds <- 10

# We use stratified cross-validation (to have similar distributions of 1s and 0s)
folds <- rep(NA, nrow(training_data))

folds[training_data[,1]=="Negative"] <-
  rep_len(1:nrFolds,
          sum(training_data[,1]=="Negative"))[sample(sum(training_data[,1]=="Negative"),
                                                     sum(training_data[,1]=="Negative"))]

folds[training_data[,1]=="Positive"] <-
  rep_len(1:nrFolds,
          sum(training_data[,1]=="Positive"))[sample(sum(training_data[,1]=="Positive"),
                                                     sum(training_data[,1]=="Positive"))]

####################
# Section 3 Experiments
####################

# Experiment 1

res <- NULL

F1 <- NULL
ROCAUC <- NULL
PRAUC <-NULL

rf1 <- randomForest(y~., data = training_data, ntree = 1000)

predictions <- predict(rf1, newdata = test_data, type = "prob")

confusion <- confMat(pred = predictions[,2], obs = test_data$y)

tmp_F1 <- 2*confusion$tp / (2*confusion$tp + confusion$fn + confusion$fp)

F1 <- c(F1, tmp_F1)

tmp_ROCAUC <- as.numeric(roc(test_data$y, predictions[,2])$auc)
```

```
ROCAUC <- c(ROCAUC, tmp_ROCAUC)

tmp_PRAUC <- aucPRp(predictions[,2], (as.numeric(test_data[,1])-1))

PRAUC <- c(PRAUC, tmp_PRAUC)

res <- rbind(res, data.frame(Model = "Procedure 1",
                             F1 = mean(F1, na.rm = T),
                             ROC_AUC = mean(ROCAUC, na.rm = T),
                             PR_AUC = mean(PRAUC)))



# Experiment 2

F1 <- NULL
ROCAUC <- NULL
PRAUC <-NULL
# actual cross validation
for(k in 1:nrFolds) {
  # actual split of the data
  fold <- which(folds == k)
  cv_training <- training_data[-fold,]
  cv_test <- training_data[fold,]

  rf_cv <- randomForest(y~., data = cv_training, ntree = 1000)
  predictions <- predict(rf_cv, newdata = cv_test ,type = "prob")

  confusion <- confMat(pred = predictions[,2], obs = cv_test$y)

  tmp_F1 <- 2*confusion$tp / (2*confusion$tp + confusion$fn + confusion$fp)

  F1 <- c(F1, tmp_F1)

  tmp_ROCAUC <- as.numeric(roc(cv_test$y, predictions[,2])$auc)

  ROCAUC <- c(ROCAUC, tmp_ROCAUC)

  tmp_PRAUC <- aucPRp(predictions[,2], (as.numeric(cv_test[,1])-1))

  PRAUC <- c(PRAUC, tmp_PRAUC)
}


res <- rbind(res, data.frame(Model = "Procedure 2",
                             F1 = mean(F1, na.rm = T),
                             ROC_AUC = mean(ROCAUC, na.rm = T),
                             PR_AUC = mean(PRAUC)))


# Experiment 3 - CV and down-sampling correctly done

F1 <- NULL
ROCAUC <- NULL
PRAUC <-NULL
# actual cross validation
for(k in 1:nrFolds) {
  # actual split of the data
  fold <- which(folds == k)
```

```
  cv_training <- training_data[-fold,]
  cv_test <- training_data[fold,]

  rf_cv <- randomForest(y~., data = cv_training, ntree = 1000,
                        sampsize = c(sum(cv_training$y=="Positive"),
                                     sum(cv_training$y=="Positive")))
  predictions <- predict(rf_cv, newdata = cv_test ,type = "prob")

  confusion <- confMat(pred = predictions[,2], obs = cv_test$y)

  tmp_F1 <- 2*confusion$tp / (2*confusion$tp + confusion$fn + confusion$fp)

  F1 <- c(F1, tmp_F1)

  tmp_ROCAUC <- as.numeric(roc(cv_test$y, predictions[,2])$auc)

  ROCAUC <- c(ROCAUC, tmp_ROCAUC)

  tmp_PRAUC <- aucPRp(predictions[,2], (as.numeric(cv_test[,1])-1))

  PRAUC <- c(PRAUC, tmp_PRAUC)
}


res <- rbind(res, data.frame(Model = "Procedure 3",
                             F1 = mean(F1, na.rm = T),
                             ROC_AUC = mean(ROCAUC, na.rm = T),
                             PR_AUC = mean(PRAUC)))

# Experiment 4 - CV and down-sampling wrongly done

# Downsample data prior to cross-validation
sel1 <- sample(nrow(training_data[training_data$y=="Positive",]),
               sum(training_data$y=="Positive"))
sel0 <- sample(nrow(training_data[training_data$y=="Negative",]),
               sum(training_data$y=="Positive"))

ds_training_data <- rbind(training_data[training_data$y=="Positive",][sel1,],
                           training_data[training_data$y=="Negative",][sel0,])

# Stratified cross-validation folds for down-sampled data

ds_folds <- rep(NA, nrow(ds_training_data))

ds_folds[ds_training_data[,1]=="Negative"] <- rep_len(1:nrFolds,
                                                       sum(ds_training_data[,1]=="Negative"))[sample(sum(ds_training_data[,1]=="Negative"),
                                                                                                              sum(ds_training_data[,1]=="Negative"))]

ds_folds[ds_training_data[,1]=="Positive"] <-  rep_len(1:nrFolds,
                                                        sum(ds_training_data[,1]=="Positive"))[sample(sum(ds_training_data[,1]=="Positive"),
                                                                                                               sum(ds_training_data[,1]=="Positive"))]



F1 <- NULL
ROCAUC <- NULL
PRAUC <-NULL
# actual cross validation

for(k in 1:nrFolds) {
```

```
  # actual split of the data
  fold <- which(ds_folds == k)
  cv_training <- ds_training_data[-fold,]
  cv_test <- ds_training_data[fold,]

  rf_cv <- randomForest(y~., data = cv_training, ntree = 1000,
                        sampsize = c(sum(cv_training$y=="Positive"),
                                     sum(cv_training$y=="Positive")))
  predictions <- predict(rf_cv, newdata = cv_test ,type = "prob")

  confusion <- confMat(pred = predictions[,2], obs = cv_test$y)

  tmp_F1 <- 2*confusion$tp / (2*confusion$tp + confusion$fn + confusion$fp)

  F1 <- c(F1, tmp_F1)

  tmp_ROCAUC <- as.numeric(roc(cv_test$y, predictions[,2])$auc)

  ROCAUC <- c(ROCAUC, tmp_ROCAUC)

  tmp_PRAUC <- aucPRp(predictions[,2], (as.numeric(cv_test[,1])-1))

  PRAUC <- c(PRAUC, tmp_PRAUC)
}


res <- rbind(res, data.frame(Model = "Procedure 4",
                             F1 = mean(F1, na.rm = T),
                             ROC_AUC = mean(ROCAUC, na.rm = T),
                             PR_AUC = mean(PRAUC)))


# Procedure 5 weird MSHK apparent error

F1 <- NULL
ROCAUC <- NULL
PRAUC <-NULL

tc <- trainControl(method="cv",
                 number=10,#creates CV folds - 10 for this data
                 summaryFunction=twoClassSummary, # provides ROC summary stats in call to model
                 classProb=T)

caret_rf <- train(y~.,
                metric = "ROC", method = "rf",
                ntree = 1000,
                sampsize=c(sum(cv_training$y=="Positive"),
                           sum(cv_training$y=="Positive")),
                trControl = tc, data = training_data)

predictions <- predict(caret_rf, type = "prob")

confusion <- confMat(pred = predictions[,2], obs = training_data$y)

tmp_F1 <- 2*confusion$tp / (2*confusion$tp + confusion$fn + confusion$fp)

F1 <- c(F1, tmp_F1)

tmp_ROCAUC <- as.numeric(roc(training_data$y, predictions[,2])$auc)
```

```
ROCAUC <- c(ROCAUC, tmp_ROCAUC)

tmp_PRAUC <- aucPRp(predictions[,2], (as.numeric(training_data[,1])-1))

PRAUC <- c(PRAUC, tmp_PRAUC)

res <- rbind(res, data.frame(Model = "Procedure 5",
                             F1 = mean(F1, na.rm = T),
                             ROC_AUC = mean(ROCAUC, na.rm = T),
                             PR_AUC = mean(PRAUC)))


# Procedure 6 weird MSHK oos error

F1 <- NULL
ROCAUC <- NULL
PRAUC <-NULL


predictions <- predict(caret_rf, test_data, type = "prob")

confusion <- confMat(pred = predictions[,2], obs = test_data$y)

tmp_F1 <- 2*confusion$tp / (2*confusion$tp + confusion$fn + confusion$fp)

F1 <- c(F1, tmp_F1)

tmp_ROCAUC <- as.numeric(roc(test_data$y, predictions[,2])$auc)

ROCAUC <- c(ROCAUC, tmp_ROCAUC)

tmp_PRAUC <- aucPRp(predictions[,2], (as.numeric(test_data[,1])-1))

PRAUC <- c(PRAUC, tmp_PRAUC)

res <- rbind(res, data.frame(Model = "Procedure 6",
                             F1 = mean(F1, na.rm = T),
                             ROC_AUC = mean(ROCAUC, na.rm = T),
                             PR_AUC = mean(PRAUC)))

######
# Section 4 Re-analysis of MSHK
######

set.seed(20180503)

setwd("~/Dropbox/4_PhD/4_PhD_Papers/pa-letter-civil-war/Revision 1/Misc/code/mn/experiment")

data <- read.csv(file = "SambnisImp.csv")

df <- data[,c("warstds", "ager", "agexp", "anoc", "army85", "autch98", "auto4",
              "autonomy", "avgnabo", "centpol3", "coldwar", "decade1", "decade2",
              "decade3", "decade4", "dem", "dem4", "demch98", "dlang", "drel",
              "durable", "ef", "ef2", "ehet", "elfo", "elfo2", "etdo4590",
              "expgdp", "exrec", "fedpol3", "fuelexp", "gdpgrowth", "geo1", "geo2",
              "geo34", "geo57", "geo69", "geo8", "illiteracy", "incumb", "infant",
              "inst", "inst3", "life", "lmtnest", "ln_gdpen", "lpopns", "major",
              "manuexp", "milper", "mirps0", "mirps1", "mirps2", "mirps3", "nat_war", "ncontig",
```

```
             "nmgdp", "nmdp4_alt", "numlang", "nwstate", "oil", "p4mchg",
             "parcomp", "parreg", "part", "partfree", "plural", "plurrel",
             "pol4", "pol4m", "pol4sq", "polch98", "polcomp", "popdense",
             "presi", "pri", "proxregc", "ptime", "reg", "regd4_alt", "relfrac",
             "seceduc","second", "semipol3", "sip2", "sxpnew", "sxpsq",
             "tnatwar", "trade", "warhist", "xconst")]

# Converting DV into Factor with names for Caret Library

df$warstds<-factor(
  df$warstds,
  levels=c(0,1),
  labels=c("peace", "war"))


training_df <- df[data$year < 1990,]
test_df <- df[data$year >= 1990,]

# Create folds for 10-fold cross-validation

nrFolds <- 10

# We use stratified cross-validation (to have similar distributions of 1s and 0s)
folds <- rep(NA, nrow(training_df))

folds[training_df[,1]=="peace"] <-
  rep_len(1:nrFolds,
          sum(training_df[,1]=="peace"))[sample(sum(training_df[,1]=="peace"),
                                              sum(training_df[,1]=="peace"))]

folds[training_df[,1]=="war"] <-
  rep_len(1:nrFolds,
          sum(training_df[,1]=="war"))[sample(sum(training_df[,1]=="war"),
                                              sum(training_df[,1]=="war"))]

# Procedure 1

res_re <- NULL

F1 <- NULL
ROCAUC <- NULL
PRAUC <-NULL

rf1 <- randomForest(warstds~., data = training_df, ntree = 1000)

rf1

predictions <- predict(rf1, newdata = test_df, type = "prob")

confusion <- confMat(pred = predictions[,2], obs = test_df$warstds)

tmp_F1 <- 2*confusion$tp / (2*confusion$tp + confusion$fn + confusion$fp)

F1 <- c(F1, tmp_F1)

tmp_ROCAUC <- as.numeric(roc(test_df$warstds, predictions[,2])$auc)

ROCAUC <- c(ROCAUC, tmp_ROCAUC)
```

```
tmp_PRAUC <- aucPRp(predictions[,2], (as.numeric(test_df[,1])-1))

PRAUC <- c(PRAUC, tmp_PRAUC)

res_re <- rbind(res_re, data.frame(Model = "Procedure 1",
                             F1 = mean(F1, na.rm = T),
                             ROC_AUC = mean(ROCAUC, na.rm = T),
                             PR_AUC = mean(PRAUC)))



# Procedure 2

F1 <- NULL
ROCAUC <- NULL
PRAUC <-NULL
# actual cross validation
for(k in 1:nrFolds) {
  # actual split of the data
  fold <- which(folds == k)
  cv_training <- training_df[-fold,]
  cv_test <- training_df[fold,]

  rf_cv <- randomForest(warstds~., data = cv_training, ntree = 1000)
  predictions <- predict(rf_cv, newdata = cv_test ,type = "prob")

  confusion <- confMat(pred = predictions[,2], obs = cv_test$warstds)

  tmp_F1 <- 2*confusion$tp / (2*confusion$tp + confusion$fn + confusion$fp)

  F1 <- c(F1, tmp_F1)

  tmp_ROCAUC <- as.numeric(roc(cv_test$warstds, predictions[,2])$auc)

  ROCAUC <- c(ROCAUC, tmp_ROCAUC)

  tmp_PRAUC <- aucPRp(predictions[,2], (as.numeric(cv_test[,1])-1))

  PRAUC <- c(PRAUC, tmp_PRAUC)
}


res_re <- rbind(res_re, data.frame(Model = "Procedure 2",
                             F1 = mean(F1, na.rm = T),
                             ROC_AUC = mean(ROCAUC, na.rm = T),
                             PR_AUC = mean(PRAUC)))


# Experiment 3 - CV and down-sampling correctly done

F1 <- NULL
ROCAUC <- NULL
PRAUC <-NULL
# actual cross validation
for(k in 1:nrFolds) {
  # actual split of the data
  fold <- which(folds == k)
  cv_training <- training_df[-fold,]
  cv_test <- training_df[fold,]
```

```
  rf_cv <- randomForest(warstds~., data = cv_training, ntree = 1000,
                        sampsize = c(sum(cv_training[,1]=="war"),
                                     sum(cv_training[,1]=="war")))
  predictions <- predict(rf_cv, newdata = cv_test, type = "prob")

  confusion <- confMat(pred = predictions[,2], obs = cv_test[,1])

  tmp_F1 <- 2*confusion$tp / (2*confusion$tp + confusion$fn + confusion$fp)

  F1 <- c(F1, tmp_F1)

  tmp_ROCAUC <- as.numeric(roc(cv_test[,1], predictions[,2])$auc)

  ROCAUC <- c(ROCAUC, tmp_ROCAUC)

  tmp_PRAUC <- aucPRp(predictions[,2], (as.numeric(cv_test[,1])-1))

  PRAUC <- c(PRAUC, tmp_PRAUC)
}


res_re <- rbind(res_re, data.frame(Model = "Procedure 3",
                             F1 = mean(F1, na.rm = T),
                             ROC_AUC = mean(ROCAUC, na.rm = T),
                             PR_AUC = mean(PRAUC)))

# Experiment 4 - CV and down-sampling wrongly done

# Downsample data prior to cross-validation
sel1 <- sample(nrow(training_df[training_df[,1]=="war",]),
               sum(training_df[,1]=="war"))
sel0 <- sample(nrow(training_df[training_df[,1]=="peace",]),
               sum(training_df[,1]=="war"))

ds_training_data <- rbind(training_df[training_df[,1]=="war",][sel1,],
                          training_df[training_df[,1]=="peace",][sel0,])

# Stratified cross-validation folds for down-sampled data

ds_folds <- rep(NA, nrow(ds_training_data))

ds_folds[ds_training_data[,1]=="peace"] <- rep_len(1:nrFolds,
                                                      sum(ds_training_data[,1]=="peace"))[sample(sum(ds_training_data[,1]=="peace"),
                                                                                                  sum(ds_training_data[,1]=="peace"))]

ds_folds[ds_training_data[,1]=="war"] <-  rep_len(1:nrFolds,
                                                       sum(ds_training_data[,1]=="war"))[sample(sum(ds_training_data[,1]=="war"),
                                                                                                 sum(ds_training_data[,1]=="war"))]



F1 <- NULL
ROCAUC <- NULL
PRAUC <-NULL
# actual cross validation

for(k in 1:nrFolds) {
  # actual split of the data
  fold <- which(ds_folds == k)
```

```
  cv_training <- ds_training_data[-fold,]
  cv_test <- ds_training_data[fold,]

  rf_cv <- randomForest(warstds~., data = cv_training, ntree = 1000,
                        sampsize = c(sum(cv_training[,1]=="war"),
                                     sum(cv_training[,1]=="war")))
  predictions <- predict(rf_cv, newdata = cv_test ,type = "prob")

  confusion <- confMat(pred = predictions[,2], obs = cv_test[,1])

  tmp_F1 <- 2*confusion$tp / (2*confusion$tp + confusion$fn + confusion$fp)

  F1 <- c(F1, tmp_F1)

  tmp_ROCAUC <- as.numeric(roc(cv_test[,1], predictions[,2])$auc)

  ROCAUC <- c(ROCAUC, tmp_ROCAUC)

  tmp_PRAUC <- aucPRp(predictions[,2], (as.numeric(cv_test[,1])-1))

  PRAUC <- c(PRAUC, tmp_PRAUC)
}


res_re <- rbind(res_re, data.frame(Model = "Procedure 4",
                             F1 = mean(F1, na.rm = T),
                             ROC_AUC = mean(ROCAUC, na.rm = T),
                             PR_AUC = mean(PRAUC)))


# Procedure 5 weird MSHK apparent error

F1 <- NULL
ROCAUC <- NULL
PRAUC <-NULL

# Original Code by MSHK just changing data.full to training_df

tc <- trainControl(method="cv",
                   number=10,#creates CV folds - 10 for this data
                   summaryFunction=twoClassSummary, # provides ROC summary stats in call to model
                   classProb=T)

model.rf<-train(as.factor(warstds)~.,
                metric="ROC", method="rf",
                sampsize=c(floor(0.0043*sum(training_df[,1]=="peace")),
                           floor(0.78*sum(training_df[,1]=="war"))), #Downsampling the class-imbalanced DV
                importance=T, # Variable importance measures retained
                proximity=F, ntree=1000, # number of trees grown
                trControl=tc, data=training_df)


predictions <- predict(model.rf, type = "prob")

confusion <- confMat(pred = predictions[,2], obs = training_df[,1])

tmp_F1 <- 2*confusion$tp / (2*confusion$tp + confusion$fn + confusion$fp)

F1 <- c(F1, tmp_F1)
```

```
tmp_ROCAUC <- as.numeric(roc(training_df[,1], predictions[,2])$auc)

ROCAUC <- c(ROCAUC, tmp_ROCAUC)

tmp_PRAUC <- aucPRp(predictions[,2], (as.numeric(training_df[,1])-1))

PRAUC <- c(PRAUC, tmp_PRAUC)

res_re <- rbind(res_re, data.frame(Model = "Procedure 5",
                             F1 = mean(F1, na.rm = T),
                             ROC_AUC = mean(ROCAUC, na.rm = T),
                             PR_AUC = mean(PRAUC)))

# Procedure 6 weird MSHK oos error

F1 <- NULL
ROCAUC <- NULL
PRAUC <-NULL



predictions <- predict(model.rf, test_df, type = "prob")

confusion <- confMat(pred = predictions[,2], obs = test_df[,1])

tmp_F1 <- 2*confusion$tp / (2*confusion$tp + confusion$fn + confusion$fp)

F1 <- c(F1, tmp_F1)

tmp_ROCAUC <- as.numeric(roc(test_df[,1], predictions[,2])$auc)

ROCAUC <- c(ROCAUC, tmp_ROCAUC)

tmp_PRAUC <- aucPRp(predictions[,2], (as.numeric(test_df[,1])-1))

PRAUC <- c(PRAUC, tmp_PRAUC)

res_re <- rbind(res_re, data.frame(Model = "Procedure 6",
                             F1 = mean(F1, na.rm = T),
                             ROC_AUC = mean(ROCAUC, na.rm = T),
                             PR_AUC = mean(PRAUC)))
```

C Problematic Cross-Validation in Muchlinski et al. (2016)

C.1 Description of data set used in Muchlinski et al. (2016)

Muchlinski et al. (2016) use a data set that is based on the data set provided by Hegre and Sambanis (2006). The dependent variable civil war onset is a binary measure of whether a civil war onset occurred for a given country in a given year[1]. The data set includes $7,140$ observations and a rich set of variables (while the original data set contains more variables, Muchlinski et al. (2016) use 90 of them).

C.2 Re-analysis of Muchlinski et al. (2016)

For our re-analysis, we split the data set into two parts. The training set contains all observations from 1945 to 1989, and the test data set all observations from 1990 to 2000. This is a natural split into training and test set for time series data. The training set contains $5,299$ observations, with 88 civil war onsets (0.017% civil war onsets). The test data set contains 1841 observations, with 28 civil war onsets (0.015% civil war onsets).

We mirror the experimental strategy used in section three of the paper, using the same six procedures as before. For each of the procedures, we calculate the same performance measures as before (F_1 score, ROC-AUC, PR-AUC).

- Procedure 1: For Procedure 1, we train a default random forest (with m_{try} set to 9 and the number of trees to $1,000$) without parameter tuning on the training set and report the performance on the test set.

[1]In the replication dataset (Muchlinski, 2015) provided by Muchlinski et al. (2016) the ratio of 1s (civil war onsets) and 0s (peace) is about 1:100. This class imbalance complicates the prediction.

- Procedure 2: For Procedure 2, we train a default random forest model without parameter tuning on the training set, but this time using 10-fold cross-validation. We then use the average across all 10 test folds to obtain the true error.

- Procedure 3: for Procedure 3, we combine 10-fold cross-validation and down-sampling correctly. This means we first split the entire data set into 10 folds, and then only down-sample the folds used for training while not touching the test folds.

- Procedure 4: for Procedure 4, we combine 10-fold cross-validation and down-sampling wrongly. This means we first down-sample the data set prior to the cross-validation, resulting in balanced training and test folds

- Procedure 5: for Procedure 5, we follow the exact modeling procedure of Muchlinski et al. (2016). This is, we combine down-sampling and parameter tuning in a single cross-validation and report the apparent error scores of the best model. The best model means the model with the m_{try} value with the smallest cross-validation error (in the case of Muchlinski et al. (2016), the model with $m_{try} = 2$ gives the minimal cross-validation error). This model was then used to predict the outcomes of the training data, which results in reporting the apparent error.

- Procedure 6: for Procedure 6, we use the model from procedure 5 and use it for an out-of-sample prediction on the test set. Because out-of-sample prediction gives a close approximation of the true error, we observe that the model performance in terms of PR-AUC drops from 0.43 in procedure 5 to only 0.07 in Procedure 6.

In Muchlinski et al. (2016), the authors report a ROC-AUC of 0.91 (page 96 in the original paper) obtained by the exact same procedure as outlined in Procedure 5 above. This means they train a random forest model with 10-fold cross-validation on the complete data set, where they use cross-validation to tune the value of m_{try}. The model with the best tuning parameter (i.e. the value of m_{try} that gives the minimum error across all folds of the cross-validation procedure) is then used to predict the same data set that was used in the training process.

```
library(caret)
set.seed(666) #the most metal seed for CV

#This method of data slicing - or CV - will be used for all logit models - uncorrected and corrected
tc<-trainControl(method="cv",
                 number=10,#creates CV folds - 10 for this data
                 summaryFunction=twoClassSummary, # provides ROC summary stats in call to model
                 classProb=T)

###Implementing RF (with CV) on entirety of data###

model.rf<-train(as.factor(warstds)~.,
                  metric="ROC", method="rf",
                  sampsize=c(30,90), #Downsampling the class-imbalanced DV
                    importance=T, # Variable importance measures retained
                   proximity=F, ntree=1000, # number of trees grown
                   trControl=tc, data=data.full)
model.rf$finalModel

library(ROCR)
attach(data.full) #have to attach the data to get probs for some reason

RF.1.pred<-predict(model.rf, data.full$warstds, type="prob")

pred.RF.1<-prediction(RF.1.pred$war, data$warstds)
perf.RF.1<-performance(pred.RF.1, "tpr", "fpr")

plot(perf.RF.1, add=T, lty=4)
```

They train the random forest model[2] with the R package `caret` (from Jed Wing et al., 2017). In the `trainControl` function they specify that 10-fold cross-validation should be applied. By default `caret` uses cross-validation in combination with random forests to tune the m_{try} parameter. In the call of the `train` function they furthermore down-sample their data. To get their ROC-AUC value they then call `predict` on the caret object `model.rf`. They do not specify a new data set. `caret` by default predicts (i.e. when no new data is specified) predicted probabilities for the training data[3], thus this can only be used to get the apparent error of a model.

Muchlinski et al. (2016), thus, report the apparent error of their model as expressed by

[2] Note that this is a different random forest model than the one they use for "out-of-sample prediction" below.

[3] This behavior of `predict.caret` is very different of the default behavior `predict.randomForest`, see below.

their ROC-AUC score, although they claim to report "cross-validated" scores (see footnote 7 in the original paper).

In short, Muchlinski et al. (2016) first train and optimize their model on the CWD, and then use the optimized model to predict the same data. This results in a serious misreport of model performance, because the random forest model actually just memorizes the pattern in the training data (a.k.a. overfitting).

C.3 Reporting random probabilities as out-of-sample predictions

Muchlinski et al. (2016) report that they trained their random forest model on a data set with observations from 1945—2000, and then updated this data set for all countries in Africa and the Middle East for the years 2001-2014, giving them 737 out-of-sample observations. Muchlinski et al. (2016) conclude that their random forest model correctly predicts nine out of twenty civil war onsets in their new out-of-sample data. This conclusion is prominently cited (see Cederman and Weidmann, 2017; Cranmer and Desmarais, 2017; Colaresi and Mahmood, 2017). They further report a table (Table 1 page 98 in the original article) with the predicted probabilities of civil war onsets in the out-of-sample data set.

However, we are unable to replicate these results. First, the data set used for the out-of-sample predictions (i.e. 2001-2014) contains fewer variables than the training data. With this data set it is, thus, not possible to obtain out-of-sample predictions. Second, our analysis of the replication code shows that Muchlinski et al. (2016) randomly draw 737 probabilities from the in-sample predictions and merge them to out-of-sample observations of civil war onset. The authors then compare those random probabilities with the true values of the out-of-sample-data. The corresponding author was not able to provide additional data or code to clear this up. In the following, we elaborate on these two points, and show the original replication code provided by Muchlinski (2015).

C.3.1 Incomplete out-of-sample data

We only could identify the main dependent variable (`warstds`), one ID variable (`year`, but no country ID variable), and eight predictors. These predictors do not have names of variables which were used in the training. Strict out-of-sample prediction is thus not possible with the provided data set.

C.3.2 Reporting random probabilities

The following code is the original code of the replication materials in the data verse. All comments in the code are the original comments as in Muchlinski (2015).

Here, Muchlinski et al. (2016) train the random forest model on the whole CWD (`data.full`) from 1945-2000. They use down-sampling (`sampsize =c(30, 90)`, the default `mtry` parameter $\sqrt{90} = 9$ and run 1000 trees. The object `RF.out` contains the trained model.

```
###Running Random Forest without CV to get OOB Error Rate -
results not shown in paper###

RF.out<-randomForest(as.factor(warstds)~., sampsize=c(30, 90),
importance=T, proximity=F, ntree=1000, confusion=T, err.rate=T, data=data.full)
```

Then, they use this model (object `RF.out`) to get out-of-bag predictions on the training data, that is, use the random forest to predict the probability of civil war onset for all observations in the training data.

```
yhat.rf<-predict(RF.out, type="prob") #taken from RF on whole data
###We used original CW data for training data here for all models/algorithms###
Yhat.rf<-as.data.frame(yhat.rf[,2])
```

By default of `predict.randomForest`, these are out-of-bag predictions on the training data, since in the `predict` command, no new data is specified. Therefore, this returns $7,140$ (the number of observations in the training data) predictions. The object `Yhat.rf` contains the predicted probabilities of a civil war onset (the second column of the object `yhat.rf`) for the training data.

Muchlinski et al. (2016) now take a random sample of 737 values (the number of observations in the out-of-sample data) from the predicted probabilities of the in-sample data (from the `Yhat.rf` object). The corresponding comment in the code does not help to clarify why this is done: `Selecting random samples to make pred and actual lengths equal`

```
###Selecting random samples to make pred and actual lengths equal###
set.seed(100)
predictors.rf<-Yhat.rf[sample(nrow(Yhat.rf), 737),]
```

In the next step, Muchlinski et al. (2016) then compare those random predicted probabilities with the true values of the out-of-sample-data, and obtain a confusion matrix providing the primary evidence for their conclusion. This confusion matrix compares true outcomes from out-of-sample observations with predicted probabilities for random in-sample observations.

```
library(SDMTools)
confusion.matrix(data3$warstds, predictors.rf, threshold=.5)
```

To sum up, in our re-analysis we find no evidence for the impressive predictive performance of random forest as reported in Muchlinski et al. (2016). Given their misunderstanding of cross-validation and based on a wrong out-of-sample prediction it is neither correct to conclude that "Random Forests correctly predicts nine of twenty civil war onsets in this out-of-sample data" (Muchlinski et al., 2016, 96) nor that "Random Forests offers superior predictive power compared to several forms of logistic regression in an important applied domain – the quantitative analysis of civil war" (Muchlinski et al., 2016, 101).

References

Abadie, A., A. Diamond, and J. Hainmueller (2015). Comparative Politics and the Synthetic Control Method. *American Journal of Political Science 59*(2), 495–510.

Ahlquist, J. S. and E. Wibbels (2012). Riding the Wave: World Trade and Factor-Based Models of Democratization. *American Journal of Political Science 56*(2), 447–464.

Atkeson, L. R., A. N. Adams, and R. M. Alvarez (2014). Nonresponse and Mode Effects in Self- and Interviewer-Administered Surveys. *Political Analysis 22*(3), 304–320.

Beauchamp, N. (2017). Predicting and Interpolating State-Level Polls Using Twitter Textual Data. *American Journal of Political Science 61*(2), 490–503.

Besley, T. and M. Reynal-Querol (2014). The Legacy of Historical Conflict: Evidence from Africa. *American Political Science Review 108*(2), 319–336.

Bisbee, J. and J. M. Larson (2017). Testing social science network theories with online network data: An evaluation of external validity. *American Political Science Review 111*(3), 502–521.

Boas, T. C. and F. D. Hidalgo (2011). Controlling the Airwaves: Incumbency Advantage and Community Radio in Brazil. *American Journal of Political Science 55*(4), 869–885.

Bowers, J., M. M. Fredrickson, and C. Panagopoulos (2013). Reasoning about Interference Between Units: A General Framework. *Political Analysis 21*(1), 97–124.

Cantú, F. (2014). Identifying Irregularities in Mexican Local Elections. *American Journal of Political Science 58*(4), 936–951.

Carter, D. B. and C. S. Signorino (2010). Back to the Future: Modeling Time Dependence in Binary Data. *Political Analysis 18*(3), 271–292.

Caughey, D. and J. S. Sekhon (2011). Elections and the Regression Discontinuity Design: Lessons from Close U.S. House Races, 1942-2008. *Political Analysis 19*(4), 385–408.

Caughey, D. and C. Warshaw (2015). Dynamic estimation of latent opinion using a hierarchical group-level IRT model. *Political Analysis 23*(2), 197–211.

Cederman, L.-E. and N. B. Weidmann (2017). Predicting armed conflict: Time to adjust our expectations? *Science* (355), 474–476.

Colaresi, M. and Z. Mahmood (2017). Do the robot: Lessons from machine learning to improve conflict forecasting. *Journal of Peace Research 54*(2), 193–214.

Cranmer, S. and B. Desmarais (2016). Replication data for: What can we learn from predictive modeling?

Cranmer, S. J. and B. A. Desmarais (2017). What can we Learn from Predictive Modeling? *Political Analysis 25*(2), 145–166.

Desmarais, B. A. and J. J. Harden (2012). Comparing Partial Likelihood and Robust Estimation Methods for the Cox Regression Model. *Political Analysis 20*(1), 113–135.

Desmarais, B. A., R. J. La Raja, and M. S. Kowal (2015). The Fates of Challengers in U.S. House Elections: The Role of Extended Party Networks in Supporting Candidates and Shaping Electoral Outcomes. *American Journal of Political Science 59*(1), 194–211.

Engstrom, E. J. (2012). The Rise and Decline of Turnout in Congressional Elections: Electoral Institutions, Competition, and Strategic Mobilization. *American Journal of Political Science 56*(2), 373–386.

Esarey, J. and A. Pierce (2012). Assessing Fit Quality and Testing for Misspecification in Binary-Dependent Variable Models. *Political Analysis 20*(4), 480–500.

Fariss, C. J. (2014). Respect for Human Rights has Improved Over Time: Modeling the Changing Standard of Accountability. *American Political Science Review 108*(2), 297–318.

from Jed Wing, M. K. C., S. Weston, A. Williams, C. Keefer, A. Engelhardt, T. Cooper, Z. Mayer, B. Kenkel, the R Core Team, M. Benesty, R. Lescarbeau, A. Ziem, L. Scrucca, Y. Tang, C. Candan, and T. Hunt. (2017). *caret: Classification and Regression Training*. R package version 6.0-78.

Ghitza, Y. and A. Gelman (2013). Deep Interactions with MRP: Election Turnout and Voting Patterns Among Small Electoral Subgroups. *American Journal of Political Science 57*(3), 762–776.

Glynn, A. N. and K. M. Quinn (2011). Why Process Matters for Causal Inference. *Political Analysis 19*(3), 273–286.

Greene, D. and J. P. Cross (2017). Exploring the Political Agenda of the European Parliament Using a Dynamic Topic Modeling Approach. *Political Analysis 25*(1), 77–94.

Greenhill, B., M. D. Ward, and A. Sacks (2011). The Separation Plot: A New Visual Method for Evaluating the Fit of Binary Models. *American Journal of Political Science 55*(4), 991–1003.

Grimmer, J., S. Messing, and S. J. Westwood (2012). How Words and Money Cultivate a Personal Vote: The Effect of Legislator Credit Claiming on Constituent Credit Allocation. *American Political Science Review 106*(4), 703–719.

Grimmer, J. and B. M. Stewart (2013). Text as Data: The Promise and Pitfalls of Automatic Content Analysis Methods for Political Texts. *Political Analysis 21*(3), 267–297.

Guess, A. M. (2015). Measure for measure: An experimental test of online political media exposure. *Political Analysis 23*(1), 59–75.

Hainmueller, J. and C. Hazlett (2014). Kernel Regularized Least Squares: Reducing Misspecification Bias with a Flexible and Interpretable Machine Learning Approach. *Political Analysis 22*(2), 143–168.

Hegre, H. and N. Sambanis (2006). Sensitivity Analysis of Empirical Results on Civil War Onset. *Journal of Conflict Resolution 50*(4), 508–535.

Hill, D. W. and Z. M. Jones (2014). An Empirical Evaluation of Explanations for State Repression. *American Political Science Review 3*(108), 661–687.

Hopkins, D. J. and G. King (2010). A Method of Automated Nonparametric Content Analysis for Social Science. *American Journal of Political Science 54*(1), 229–247.

Iacus, S. M., G. King, and G. Porro (2012). Causal Inference without Balance Checking: Coarsened Exact Matching. *Political Analysis 20*(1), 1–24.

Imai, K. and A. Strauss (2011). Estimation of Heterogeneous Treatment Effects from Randomized Experiments, with Application to the Optimal Planning of the Get-Out-the-Vote Campaign. *Political Analysis 19*(1), 1–19.

Jessee, S. (2016). (How) Can We Estimate the Ideology of Citizens and Political Elites on the Same Scale? *American Journal of Political Science 60*(4), 1108–1124.

Keele, L. (2010). Proportionally Difficult: Testing for Nonproportional Hazards in Cox Models. *Political Analysis 18*(2), 189–205.

Keele, L. and W. Minozzi (2013). How Much Is Minnesota Like Wisconsin? Assumptions and Counterfactuals in Causal Inference with Observational Data. *Political Analysis 21*(2), 193–216.

Keele, L. J. and R. Titiunik (2015). Geographic boundaries as regression discontinuities. *Political Analysis 23*(1), 127–155.

König, T., M. Marbach, and M. Osnabrügge (2013). Estimating Party Positions across Countries and Time – A Dynamic Latent Variable Model for Manifesto Data. *Political Analysis 21*(4), 468–491.

Lauderdale, B. E. and T. S. Clark (2012). The Supreme Court's Many Median Justices. *American Political Science Review 106*(4), 847–866.

Montgomery, J. M., F. M. Hollenbach, and M. D. Ward (2012). Improving Predictions Using Ensemble Bayesian Model Averaging. *Political Analysis 20*(3), 271–291.

Montgomery, J. M., S. Olivella, J. D. Potter, and B. F. Crisp (2015). An informed forensics approach to detecting vote irregularities. *Political Analysis 23*(4), 488–505.

Muchlinski, D. (2015). Replication Data for: Comparing Random Forests with Logistic Regression for Predicting Class-Imbalanced Civil War Onset Data.

Muchlinski, D., D. Siroky, J. He, and M. Kocher (2016). Comparing Random Forest with Logistic Regression for Predicting Class-Imbalanced Civil War Onset Data. *Political Analysis 24*(1), 87–103.

Peterson, A. and A. Spirling (2018). Classification Accuracy as a Substantive Quantity of Interest: Measuring Polarization in Westminster Systems. *Political Analysis 26*(1), 120–128.

Rainey, C. (2014). Arguing for a Negligible Effect. *American Journal of Political Science 58*(4), 1083–1091.

Robin, X., N. Turck, A. Hainard, N. Tiberti, F. Lisacek, J.-C. Sanchez, and M. Müller (2011). pROC: an open-source package for R and S+ to analyze and compare ROC curves. *BMC Bioinformatics 12*, 77.

Weber, T. (2011). Exit, Voice, and Cyclicality: A Micrologic of Midterm Effects in European Parliament Elections. *American Journal of Political Science 55*(4), 907–922.

Wilkerson, J., D. Smith, and N. Stramp (2015). Tracing the Flow of Policy Ideas in Legislatures: A Text Reuse Approach. *American Journal of Political Science 59*(4), 943–956.

Xu, Y. (2016). Generalized Synthetic Control Method : Causal Inference with Interactive Fixed Effects Models. *Political Analysis 25*(1), 57–76.