Linear Model, GAM, and Random Forests

In this project, I replicate one of the primary model specifications in Machain (2021).[1] I assess both the in-sample and out-of-sample predictive performance of this model specification using linear models, general additive models (GAMs), and random forests.

The original model specification, as detailed in the article, is estimated using a linear model. Machain's study focuses on the impact of training foreign military personnel in the United States on the relationship between the US and the host country. Specifically, it investigates how the number of foreign soldiers trained in the US influences the United Nations voting similarity between the US and the host country.

This project is structured as follows. First, I replicate the main model used in Machain (2021). Subsequently, I divide the data into training and testing sets, assessing the predictive capabilities of General Additive Models (GAMs) and Random Forests through cross-validation on the training data. Following this, I evaluate the out-of-sample predictive performance of linear models, GAMs, and Random Forests using the test data. Then, utilizing the entire dataset, I analyze the importance of features across the three models. Lastly, I delve into exploring nonlinear relationships using GAMs.

Table 1: Descriptive Statistics

Statistic	N	Mean	St. Dev.	Min	Max
Ideal Point Difference	2,598	−2.838	0.844	−4.618	−0.112
Military Trainee	2,598	1.113	2.693	0	29
Polity 2	2,598	3.786	6.264	−10	10
US Alliance	2,598	0.539	0.499	0	1
GDP Per Capita	2,598	12,424.220	18,471.680	219.187	111,968.300

Table 1 displays the descriptive statistics utilized in the estimations, with the Ideal Point Difference serving as the dependent variable in the analysis. This variable represents the absolute difference between the ideal point of the United States and that of a given country, calculated based on United Nations voting patterns. To facilitate interpretation, the author has multiplied this variable by -1. Positive coefficients indicate a positive relationship with the United States.

The Military Trainee variable represents the proportion of military personnel trained in the US relative

[1]Martinez Machain, Carla. "Exporting influence: US military training as soft power." *Journal of Conflict Resolution* 65.2-3 (2021): 313-341.

to the total military personnel in the host country, and it is treated as a continuous variable. Polity 2 is another continuous variable denoting regime type, while US alliance is a binary variable that is assigned a value of 1 for countries with an alliance with the US. Additionally, GDP per capita is a continuous variable representing the GDP per capita of the host state.

Table 2 presents the replication of Model 2.1 from Table 2 in Machain's study (2021). The estimates align precisely with those documented in the article. Notably, military trainees appear to exert a negative impact on the voting similarity between the US and host countries. Conversely, regime type, alliance with the US, and GDP per capita exhibit positive and statistically significant associations with a higher voting similarity with the US.

Table 2: Replication of Model 2.1 in Table 2 in Machain (2021)

	Dependent variable:
matraineesmilper	-0.014^{***}
	(0.004)
polity2	0.051^{***}
	(0.002)
usalliance	0.515^{***}
	(0.028)
gdppc	0.00001^{***}
	(0.00000)
Constant	-3.470^{***}
	(0.015)
Note:	*p<0.1; **p<0.05; ***p<0.01

1 In-Sample Predictive Performance of GAMs and Random Forests

Employing the identical independent variables as in Machain's study (2021), I employ a GAM and a Random Forest. To accomplish this, I implement cross-validation techniques on the training data. Table 3 displays the mean squared errors calculated using these two models on the training data, utilizing 5-fold cross-validation.

Table 3: Mean Squared Errors Computed Using Random Forest and GAM Models

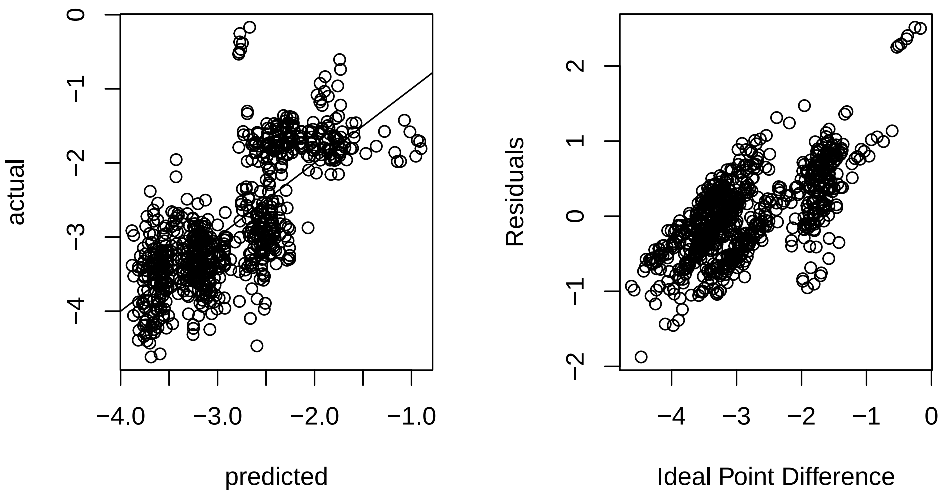
	Mean Squared Error
Random Forest	0.1846633
GAM	0.2566158

Notably, the Random Forest model yields a smaller mean squared error compared to the GAM, indicating that the Random Forest model offers a superior fit to the data.

2 Out-of-Sample Predictive Performance of the Linear Model, GAMs ,and Random Forests

Figures 1 through 3 depict plots designed to assess the out-of-sample predictive performance of the linear model, GAM, and random forest. In the plots on the left-hand side, predicted values are plotted on the x-axis against actual values on the y-axis. Meanwhile, in the plots on the right-hand side, the x-axis represents the ideal point difference variable, while the y-axis represents the residuals obtained by subtracting predictions from actual values in the test data.

Figure 1: Out-of-Sample Predictive Performance of the Linear Model



The plots representing actual and predicted values reveal that the random forest model exhibits the most accurate predictive performance out-of-sample, as it generates predictions closest to the actual values. In contrast, the GAM and linear model display inferior performance compared to the random forest model, evident from the widely scattered actual observations in the plots.

Figure 2: Out-of-Sample Predictive Performance of the GAM Model

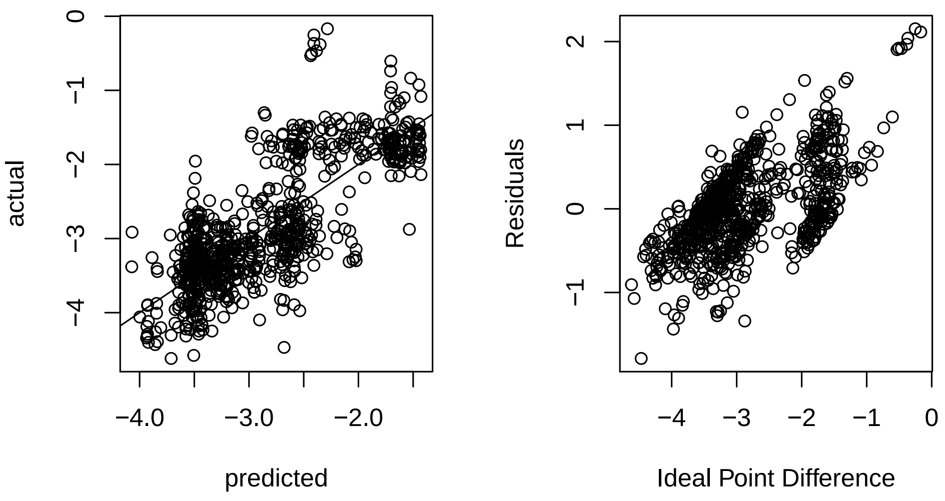
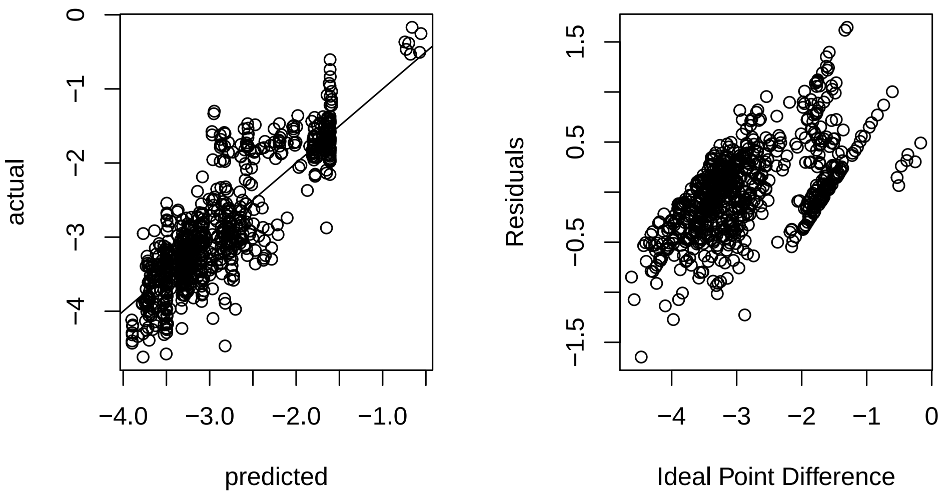


Figure 3: Out-of-Sample Predictive Performance of the Random Forest Model



Upon examining the residual plots, it becomes apparent that all models tend to overpredict high values and underpredict low values of the ideal point difference variable. Notably, predictions generated by all three models closely approximate the actual values for intermediate values of the ideal point distance variable.

In conclusion, the random forest model appears to demonstrate the most reliable predictive performance when applied to test data.

3 Evaluating Feature Importance

In this part of the project, I assess feature importance using the complete dataset across three models. I employ two strategies for this purpose. Initially, I introduce each covariate individually and compute the mean squared error. This approach reveals the impact of adding each variable on predictive performance. Subsequently, I systematically exclude one variable at a time to determine whether its omission influences predictive performance.

Table 4: Introducing Variables One by One

	MSE LM	MSE GAM	MSE RF
Military Trainee	0.71	0.69	0.63
Military Trainee + Polity 2	0.43 (38% ↓)	0.35 (49% ↓)	0.31 (51% ↓)
Military Trainee + Polity 2 + US Alliance	0.36 (16% ↓)	0.31 (12% ↓)	0.26 (15% ↓)
Military Trainee + Polity 2 + US Alliance + GDP pc	0.30 (16% ↓)	0.26 (14% ↓)	0.18 (29% ↓)

Table 4 displays the mean squared errors (MSEs) for three models when introducing variables individually. When considering only the military trainee variable, the random forest model demonstrates superior performance compared to the other models. Introducing the polity variable enhances the predictive performance of the random forest, GAM, and linear model by 51%, 49%, and 38%, respectively. Incorporating the US alliance variable notably improves the performance of the linear model more than the other models. Additionally, including the GDP per capita variable boosts the predictive performance of random forests by 29%, representing the highest increase among the variables considered.

In summary, the MSEs presented in Table 4 indicate that adding a variable tends to enhance the predictive performance of random forests more significantly than it does for other models. However, an exception to this trend is observed when incorporating the US alliance variable, where the linear model shows the most substantial improvement in predictive performance among the three models.

Table 5: Omitting One Variable at A Time

	% Change in MSE LM	% Change in MSE GAM	% Change in MSE RF
Military Trainee	0.4 % ↑	0.2 % ↑	10 % ↑
Polity 2	23 % ↑	25 % ↑	13 % ↑
US Alliance	15 % ↑	15 % ↑	15 % ↑
GDP pc	19 % ↑	16 % ↑	15 % ↑

In addition to this analysis, I also explore how excluding variables individually affects the predictive performance of the three models. Table 5 presents the percentage change in MSEs when omitting one variable at a time across the three models.

A notable observation is that excluding the military trainee variable does not significantly enhance the predictive performance of the linear model and GAM. However, it does improve the predictive performance of the random forest model by 10%.

Excluding the Polity 2 score results in the most substantial improvement in the predictive performance of GAM, with an increase of 25%.

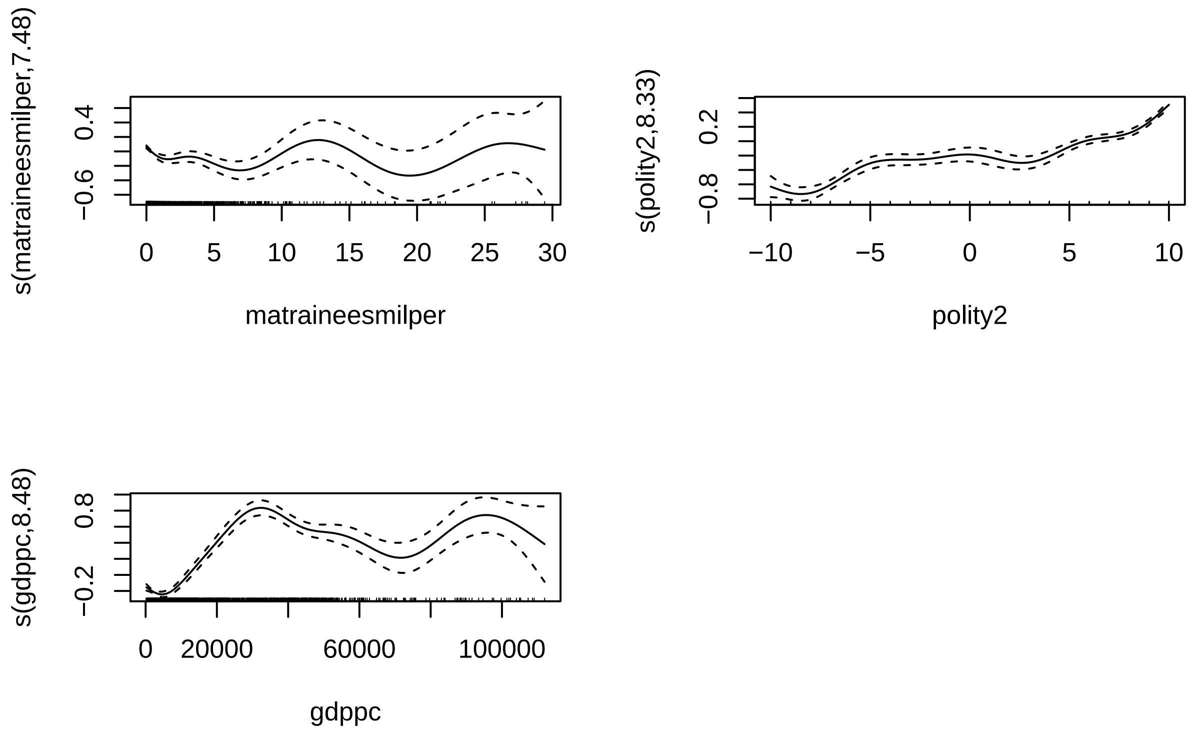
The exclusion of the US alliance variable affects the predictive performance of all three models similarly, resulting in a 15% improvement across the board.

Regarding the exclusion of the GDP per capita variable, it leads to the most significant improvement in the predictive performance of the linear model, although the differences across models are not particularly striking.

In general, the importance of features varies across these models. Variables that enhance the predictive performance of one model may diminish the predictive performance of another model.

4 Exploring Non-Linear Relationships Using GAM

Figure 4: Out-of-Sample Predictive Performance of the Random Forest Model



Using the continuous variables in the model, I establish non-linear relationships through a GAM model. Figure 4 illustrates the impact of each variable on the dependent variable.

We observe that the military training variable exhibits a somewhat non-linear relationship with the ideal point distance from the US. However, due to the wide confidence intervals, it is unclear whether this non-linear relationship is statistically significant.

The Polity 2 score appears to have a significant non-linear relationship with the ideal point distance from the US. Lower values of Polity 2 lead to a steady increase in the ideal point distance, while higher values result in a more pronounced increase.

Similarly, GDP per capita also demonstrates a non-linear correlation with the ideal point distance. Lower values of GDP per capita correspond to an increase in the ideal point distance, with moderate values showing minimal change. However, higher values of GDP per capita seem to decrease the ideal point distance from the US, suggesting an inverted U-shaped relationship between GDP per capita and the ideal point distance from the US.

R Code

```
#Sendinc Machine Learning Project


options(scipen = 10)

install.packages("mboost")

library(devtools); install_github("therneau/survival")

install.packages("mgcv")

install.packages("caret")

library(caret)

library(haven)

library(stargazer)

library(mlr)

library(mgcv)

#REPLICATION

rep_data <- read_dta("sj-dta-1-jcr-10.1177_0022002720957713.dta")


#main model Table 2 Model 1 in the original article

model1=lm(idealptdiff ~ matraineesmilper + polity2 + usalliance +gdppc, data=
    rep_data)

summary(model1)


coeftest(model1, vcov = vcovHC, cluster = ~ ccode)


#I am able to replicate the coefficients and standard errors as reported in the
    article

stargazer(coeftest(model1, vcov = vcovHC, cluster = ~ ccode))

#PREDICTIONS

#subsetting the data
```

```
data = subset(rep_data, select = c(idealptdiff,matraineesmilper,polity2,usalliance,
    gdppc))
data <- na.omit(data)
datad=as.data.frame(data)
stargazer(datad)

#creating train and test sets
set.seed(1234) #setting seed for reproducibility
dt = sort(sample(nrow(data), nrow(data)*.7))
train<-data[dt,]
test<-data[-dt,]
modelgam <- mgcv::gam(idealptdiff ~ matraineesmilper + polity2 + usalliance + gdppc
    ,data = data)

#GAM Model
#5 fold Cross Validation Using Training Data

set.seed(123)
train.control <- trainControl(method = "repeatedcv",
number = 5, repeats = 5)

modelgam <- train(idealptdiff ~., data = train, method = "gam",
trControl = train.control)
print(modelgam) #mse 0.2566158

#out of sample predictions for gam using test data
predictions <- modelgam %>% predict(test)
```

```
par(mfrow=c(1,2))

#plot of predicted and actual values

plot(predictions,test$idealptdiff,

xlab="predicted",ylab="actual")

abline(a=0,b=1)


#residual plot

resid.gam = test$idealptdiff - predictions

plot(test$idealptdiff, resid.gam,

xlab="Ideal Point Difference",

ylab="Residuals")


#Random Forest

task <- makeRegrTask(data=train, target = "idealptdiff")


#specifiying the random forest classifier

forest <- makeLearner("regr.randomForest")


forestParamSpace <- makeParamSet(

makeIntegerParam("ntree", lower = 300, upper = 300),

makeIntegerParam("mtry", lower = 1, upper = 4),

makeIntegerParam("nodesize", lower = 1, upper = 10),

makeIntegerParam("maxnodes", lower = 5, upper = 20))

set.seed(123) #setting seed for reproducibility


randSearch <- makeTuneControlRandom(maxit = 100)


cvForTuning <- makeResampleDesc("CV", iters = 5)
```

```
set.seed(123) #setting seed for reproducibility


parallelStartSocket(cpus = detectCores())


tunedForestPars <- tuneParams(forest, task = task,

resampling = cvForTuning,

par.set = forestParamSpace,

control = randSearch)


parallelStop()


tunedForestPars


#training the final model

tunedForest <- setHyperPars(forest, par.vals = tunedForestPars$x)

tunedForestModel <- mlr::train(tunedForest, task)


#Cross validating the model building


outer <- makeResampleDesc("CV", iters = 5)

forestWrapper <- makeTuneWrapper("regr.randomForest",

resampling = cvForTuning,

par.set = forestParamSpace,

control = randSearch)

set.seed(123) #setting seed for reproducibility

parallelStartSocket(cpus = detectCores())

cvWithTuningRandom <- resample(forestWrapper, task, resampling = outer)

parallelStop()

cvWithTuningRandom #0.1846633
```

```
#making predictions

predsRF <- data.frame(predict(tunedForestModel, newdata = test))


par(mfrow=c(1,2))


#plot of predicted and actual values

plot(predsRF$response,test$idealptdiff,

xlab="predicted",ylab="actual")

abline(a=0,b=1)


#residual plot

resid.rf = test$idealptdiff - predsRF$response

plot(test$idealptdiff, resid.rf,

xlab="Ideal Point Difference",

ylab="Residuals")


#Linear Model

set.seed(1234) #setting seed for reproducibility


modellm <- train(idealptdiff ~., data = train, method = "lm",

trControl = train.control)

print(modellm)


#out of sample predictions for gam using test data

predictionslm <- modellm %>% predict(test)


par(mfrow=c(1,2))

#plot of predicted and actual values
```

```
plot(predictionslm,test$idealptdiff,
xlab="predicted",ylab="actual")
abline(a=0,b=1)

#residual plot
resid.lm = test$idealptdiff - predictionslm
plot(test$idealptdiff, resid.lm,
xlab="Ideal Point Difference",
ylab="Residuals")

#Investigating Variable Importance. I use full data.

#FOR LINEAR MODEL using 5 fold cv with 5 iterations
#introducing variables one by one

set.seed(1234) #setting seed for reproducibility

modellm <- train(idealptdiff ~ matraineesmilper, data = data, method = "lm",
trControl = train.control)
print(modellm)
#mse
(0.8406471)^2 #0.7066875

set.seed(1234) #setting seed for reproducibility

modellm <- train(idealptdiff ~ matraineesmilper + polity2, data = data, method = "
   lm",
trControl = train.control)
print(modellm)
```

```
#mse

(0.6580383)^2 #0.4330144


set.seed(1234) #setting seed for reproducibility


modellm <- train(idealptdiff ~ matraineesmilper + polity2 + usalliance, data = data
    , method = "lm",
trControl = train.control)
print(modellm)


#mse
(0.6023114)^2 #0.362779


set.seed(1234) #setting seed for reproducibility


modellm <- train(idealptdiff ~ matraineesmilper + polity2 + usalliance +gdppc, data
     = data, method = "lm",
trControl = train.control)
print(modellm)


#mse
(0.549897 )^2 #0.3023867


#omitting one variable each time


#omitting gdppc
set.seed(1234) #setting seed for reproducibility


modellm <- train(idealptdiff ~ matraineesmilper + polity2 + usalliance , data =
   data, method = "lm",
trControl = train.control)
print(modellm)


#mse
(0.6023114)^2 #0.362779


#omitting us alliance


set.seed(1234) #setting seed for reproducibility


modellm <- train(idealptdiff ~ matraineesmilper + polity2 +gdppc, data = data,
   method = "lm",
trControl = train.control)
print(modellm)


#mse


(0.5900286)^2 #0.3481337


#omitting polity2
set.seed(1234) #setting seed for reproducibility


modellm <- train(idealptdiff ~ matraineesmilper + usalliance +gdppc, data = data,
   method = "lm",
trControl = train.control)
print(modellm)


#mse
```

```
(0.6105179 )^2 # 0.3727321


#omitting matraineesmilper

set.seed(1234) #setting seed for reproducibility


modellm <- train(idealptdiff ~ polity2 + usalliance +gdppc, data = data, method = "
    lm",

trControl = train.control)

print(modellm)


#mse


(0.550997)^2 # 0.3035977


#GAM MODEL using 5 fold cv with 5 iterations


#introducing variables one by one

set.seed(1234) #setting seed for reproducibility


modelgam <- train(idealptdiff ~., data = data, method = "gam",

trControl = train.control)

print(modelgam)


(0.5121207)^2 #0.2622676


set.seed(1234) #setting seed for reproducibility


modelgam <- train(idealptdiff ~matraineesmilper, data = data, method = "gam",

trControl = train.control)
```

```
print(modelgam)


(0.8323055)^2 #0.6927324


set.seed(1234) #setting seed for reproducibility


modelgam <- train(idealptdiff ~matraineesmilper + polity2, data = data, method = "
    gam",
trControl = train.control)
print(modelgam)


(0.5907091 )^2 #0.3489372


set.seed(1234) #setting seed for reproducibility


modelgam <- train(idealptdiff ~matraineesmilper + polity2 + usalliance, data = data
    , method = "gam",
trControl = train.control)
print(modelgam)


(0.5534938)^2 #0.3063554


set.seed(1234) #setting seed for reproducibility


modelgam <- train(idealptdiff ~matraineesmilper + polity2 + usalliance + gdppc,
    data = data, method = "gam",
trControl = train.control)
print(modelgam)
```

```
(0.5121207)^2 #0.2622676



#omitting one variable at a time


#omitting gdppc


set.seed(1234) #setting seed for reproducibility


modelgam <- train(idealptdiff ~matraineesmilper + polity2 + usalliance, data = data
    , method = "gam",
trControl = train.control)
print(modelgam)


(0.5534938)^2 #0.3063554


#omitting usalliance


set.seed(1234) #setting seed for reproducibility


modelgam <- train(idealptdiff ~matraineesmilper + polity2 + gdppc, data = data,
   method = "gam",
trControl = train.control)
print(modelgam)


(0.5510628)^2 #0.3036702


#omitting polity2
```

```
set.seed(1234) #setting seed for reproducibility


modelgam <- train(idealptdiff ~matraineesmilper + usalliance + gdppc, data = data,
   method = "gam",
trControl = train.control)
print(modelgam)


(0.5726987)^2 #0.3279838


#omitting matraineesmilper


set.seed(1234) #setting seed for reproducibility


modelgam <- train(idealptdiff ~ polity2 + usalliance + gdppc, data = data, method =
     "gam",
trControl = train.control)
print(modelgam)


(0.5127227)^2 # 0.2628846


#Random Forests


#introducing variables one by one


#for random forest
#adding variables one by one


taskfull <- makeRegrTask(data=data[1:2], target = "idealptdiff")
set.seed(123) #setting seed for reproducibility
```

```
parallelStartSocket(cpus = detectCores())

tunedForestPars <- tuneParams(forest, task = taskfull,

resampling = cvForTuning,

par.set = forestParamSpace,

control = randSearch)

parallelStop()

forestWrapper <- makeTuneWrapper("regr.randomForest",

resampling = cvForTuning,

par.set = forestParamSpace,

control = randSearch)

set.seed(123) #setting seed for reproducibility

parallelStartSocket(cpus = detectCores())

RFmsep <- resample(forestWrapper, taskfull, resampling = outer)

parallelStop()


RFmsep #0.6313493


#############

taskfull <- makeRegrTask(data=data[1:3], target = "idealptdiff")

set.seed(123) #setting seed for reproducibility

parallelStartSocket(cpus = detectCores())

tunedForestPars <- tuneParams(forest, task = taskfull,

resampling = cvForTuning,

par.set = forestParamSpace,

control = randSearch)

parallelStop()

forestWrapper <- makeTuneWrapper("regr.randomForest",

resampling = cvForTuning,

par.set = forestParamSpace,
control = randSearch)
set.seed(123) #setting seed for reproducibility
parallelStartSocket(cpus = detectCores())
RFmsep <- resample(forestWrapper, taskfull, resampling = outer)
parallelStop()

RFmsep #0.3055535

#############
taskfull <- makeRegrTask(data=data[1:4], target = "idealptdiff")
set.seed(123) #setting seed for reproducibility
parallelStartSocket(cpus = detectCores())
tunedForestPars <- tuneParams(forest, task = taskfull,
resampling = cvForTuning,
par.set = forestParamSpace,
control = randSearch)
parallelStop()
forestWrapper <- makeTuneWrapper("regr.randomForest",
resampling = cvForTuning,
par.set = forestParamSpace,
control = randSearch)
set.seed(123) #setting seed for reproducibility
parallelStartSocket(cpus = detectCores())
RFmsep <- resample(forestWrapper, taskfull, resampling = outer)
parallelStop()

RFmsep #0.2589227

#############
```

```
taskfull <- makeRegrTask(data=data[1:5], target = "idealptdiff")

set.seed(123) #setting seed for reproducibility

parallelStartSocket(cpus = detectCores())

tunedForestPars <- tuneParams(forest, task = taskfull,

resampling = cvForTuning,

par.set = forestParamSpace,

control = randSearch)

parallelStop()

forestWrapper <- makeTuneWrapper("regr.randomForest",

resampling = cvForTuning,

par.set = forestParamSpace,

control = randSearch)

set.seed(123) #setting seed for reproducibility

parallelStartSocket(cpus = detectCores())

RFmsep <- resample(forestWrapper, taskfull, resampling = outer)

parallelStop()


RFmsep #0.1835790


#omitting variables one by one


#omitting matraineesmilper

taskfull <- makeRegrTask(data=data[-2], target = "idealptdiff")

set.seed(123) #setting seed for reproducibility

parallelStartSocket(cpus = detectCores())

tunedForestPars <- tuneParams(forest, task = taskfull,

resampling = cvForTuning,

par.set = forestParamSpace,

control = randSearch)
```

```
parallelStop()
forestWrapper <- makeTuneWrapper("regr.randomForest",
resampling = cvForTuning,
par.set = forestParamSpace,
control = randSearch)
set.seed(123) #setting seed for reproducibility
parallelStartSocket(cpus = detectCores())
RFmsep <- resample(forestWrapper, taskfull, resampling = outer)
parallelStop()

RFmsep #0.2032006

#omitting polity2
taskfull <- makeRegrTask(data=data[-3], target = "idealptdiff")
set.seed(123) #setting seed for reproducibility
parallelStartSocket(cpus = detectCores())
tunedForestPars <- tuneParams(forest, task = taskfull,
resampling = cvForTuning,
par.set = forestParamSpace,
control = randSearch)
parallelStop()
forestWrapper <- makeTuneWrapper("regr.randomForest",
resampling = cvForTuning,
par.set = forestParamSpace,
control = randSearch)
set.seed(123) #setting seed for reproducibility
parallelStartSocket(cpus = detectCores())
RFmsep <- resample(forestWrapper, taskfull, resampling = outer)
parallelStop()
```

```
RFmsep #0.2092798


#omitting usalliance

taskfull <- makeRegrTask(data=data[-4], target = "idealptdiff")

set.seed(123) #setting seed for reproducibility

parallelStartSocket(cpus = detectCores())

tunedForestPars <- tuneParams(forest, task = taskfull,

resampling = cvForTuning,

par.set = forestParamSpace,

control = randSearch)

parallelStop()

forestWrapper <- makeTuneWrapper("regr.randomForest",

resampling = cvForTuning,

par.set = forestParamSpace,

control = randSearch)

set.seed(123) #setting seed for reproducibility

parallelStartSocket(cpus = detectCores())

RFmsep <- resample(forestWrapper, taskfull, resampling = outer)

parallelStop()


RFmsep #0.2126090


#omitting gdppc

taskfull <- makeRegrTask(data=data[-5], target = "idealptdiff")

set.seed(123) #setting seed for reproducibility

parallelStartSocket(cpus = detectCores())

tunedForestPars <- tuneParams(forest, task = taskfull,

resampling = cvForTuning,
par.set = forestParamSpace,

control = randSearch)

parallelStop()

forestWrapper <- makeTuneWrapper("regr.randomForest",

resampling = cvForTuning,

par.set = forestParamSpace,

control = randSearch)

set.seed(123) #setting seed for reproducibility

parallelStartSocket(cpus = detectCores())

RFmsep <- resample(forestWrapper, taskfull, resampling = outer)

parallelStop()


RFmsep #0.2126090


#specifying non-linear relationships with GAM


par(mfrow=c(2,2))


modelgam1 <- mgcv::gam(idealptdiff ~ s(matraineesmilper) +polity2 + usalliance+
   gdppc,data = data)
plot(modelgam1)


modelgam2 <- mgcv::gam(idealptdiff ~ matraineesmilper + s(polity2) + usalliance+
   gdppc,data = data)
plot(modelgam2)


modelgam3 <- mgcv::gam(idealptdiff ~ matraineesmilper + polity2 + usalliance + s(
   gdppc),data = data)
plot(modelgam3)
```

```
modelgam <- mgcv::gam(idealptdiff ~ s(matraineesmilper, bs='ps', sp=0.2) + s(
    polity2, bs='ps', sp=0.6) + s(usalliance, bs='ps', sp=0.6) + s(gdppc, bs='ps',
    sp=0.6),data = data)
plot(modelgam)
```