ONLINE SUPPLEMENTARY DATA

Supplementary Data 1. Chart review guidelines for annotating and classifying axial spondyloarthritis.

The guidelines were used by rheumatologists who reviewed the medical records of sampled patients and classified patients as having axial spondyloarthritis (axSpA) or not having axSpA. With each patient, the chart review process required reviewers to consider relevant axSpA features and concepts. The guidelines also assisted with data interpretation to maximize classification consistency.

Data Review:

1: Step 1: Use Workspace view in eHOST* to review the following documents for evidence of axSpA:

a. Laboratory (includes HLA-B27, RF, CCP, CRP, ESR)
b. Radiology (including all imaging modalities for imaging inclusive of any joint)
    i. All pelvic radiology notes
    ii. Most recent spine radiology notes for all spine areas (cervical, thoracic, lumbar, etc.)
    iii. Most recent radiology notes for all peripheral joint areas
c. Clinician notes
    i. Most recent rheumatology note and initial rheumatology note or rheumatology consult
    ii. Most recent gastroenterology, ophthalmology, dermatology, orthopedics, pain clinic, physical medicine & rehabilitation, and primary care notes
d. Problem list with diagnoses
e. Medication list (includes biologic and non-biologic DMARDs)
f. Review additional documents, not listed above, to clarify or better describe concepts or patients, as appropriate per your clinical judgement

2: Step 2: Use Navigator view[¥] to identify SpA concepts automatically highlighted [€] by the eHOST program that were not previously captured with Step 1 annotation.

a. Use navigator links to take you to sections of notes containing the highlighted SpA terms and annotate the relevant text.

Annotation:

1. Annotate all data that contributes to your understanding of the diagnosis and phenotype
2. Highlight the most informative text for each concept (Do not necessarily need to highlight every mention of a concept in every note, but ok to duplicate if subsequently encountered text is more informative than previously annotated text).
3. Only annotate concepts related to SpA. For example, do not annotate “corneal erosions”
4. Annotate all DMARDs used for inflammatory arthritis, regardless of whether they are FDA approved for SpA
5. Do not need to annotate pending or inconclusive test results. For example, “Will order CRP” does not need to be annotated.
6. For discrepant data, annotate all relevant information and make comments about your conclusions

Classification:

1. Classify according to your clinical judgement. (Your clinical judgment trumps all other classification recommendations.)
2. If there is an ICD-9 code for an axSpA diagnosis (listed in Problem List or elsewhere) and you believe the patient does not have axSpA, highlight data that negates an axSpA diagnosis and make a comment to explain why you are overriding the ICD-9 code
3. Documentation of an axSpA diagnosis (via ICD9 code or clinical note) is NOT required to classify patients with axSpA, if sufficient data are available to convince you that the patient has axSpA.

<u>If you are uncertain of the axSpA diagnosis</u>, consider the following:

1. If a diagnosis of axSpA is listed, but there is NO supporting evidence[£] of axSpA, classify as uncertain.
2. If axSpA is listed as a diagnosis and there is conflicting evidence affirming and negating the axSpA diagnosis, classify as uncertain.
3. If axSpA is listed as a diagnosis, and there is supporting evidence of axSpA without conflicting negating evidence, classify as yes axSpA, even if the supporting evidence is less than you would require to confirm a diagnosis in your clinic (i.e. Assume that the information available to the examining provider was better than the information available via chart review.)

*Workspace view refers to an interface in the eHOST chart review software that enables chart reviewers to view multiple types of data relevant to axSpA on a single screen for each patient.

[¥]The Navigator view provides links to notes containing SpA terms that eHOST was programmed to pre-annotate (highlight). The Navigator view also provides links to all data previously annotated by the chart reviewers, organized according to specific axSpA concepts, to help reviewers identify concepts that have not previously been annotated.

[€]SpA concepts include terms such as spondyloarthritis, spondylitis, sacroiliitis, syndesmophyte, erosion, back pain, spine pain, neck pain, buttock pain, dactylitis, sausage digit, enthesitis, Achilles, plantar fascia, psoriatic, psoriasis, HLA-B27, uveitis, iritis, inflammatory bowel disease, Crohn's, ulcerative colitis, joint swelling, synovitis, inflammatory back pain, nail pitting, etc.

[£]Supporting evidence of axSpA may include the presence of features such as sacroiliitis, syndesmophytes, joint erosion, HLA-B27 positivity, chronic back pain without a more likely alternative cause, peripheral joint swelling without a more likely alternative cause, psoriasis, inflammatory bowel disease, uveitis, biologic or non-biologic DMARDs, family history of SpA, enthesitis, etc.

Supplementary Table 1. Importance ranking for the 49 variables in the Full Algorithm by Random Forest Gini Scores

Rank	Variable	ICD-9, when applicable	Mean Decrease Gini
1	# of AS ICD9 codes	720.0	68.93
2	Spond NLP		26.90
3	# of Rheumatology Visits		12.83
4	Sacroiliitis NLP		4.23
5	# Visits with providers (any type of provider)		4.22
6	Rheumatic Disease Comorbidity Index (RDCI)		3.89
7	Age at index date		3.56
8	Exposure to ≥ 1 biologic DMARD (yes/no)		3.45
9	# C-Reactive Protein tests		3.34
10	# of Unspecified inflammatory spondyloarthropathy	720.9x	3.13
11	# of Osteoarthrosis and allied disorders ICD9 codes	715.x	2.94
12	Region (Southeast/ North Atlantic/ Midwest/ Continental/ Pacific)		2.77
13	# Erythrocyte Sedimentation Rate tests		2.69
14	Time in VA system since index date		2.54
15	# of Psoriasis ICD9 code	696.1x	2.50
16	# of Arthropathy associated with Reiter's & nonspecific urethritis ICD9 codes	711.1x, 99.3	2.18
17	# Sciatica ICD9 codes	724.3	2.14
18	# of Low back pain ICD9 codes	724.2x	1.91
19	Rheumatoid Factor (positive/ negative/ pending/ uncertain)		1.62
20	# Backache, unspecified ICD9 codes	724.5x	1.49
21	# of Osteoarthrosis of the spine (includes DISH) ICD9 codes	721.x	1.48
22	HLA-B27 (positive/ negative/ pending/ uncertain)		1.45
23	# of Intervertebral disk disorders ICD9 codes	722.x	1.35
24	# of Rheumatoid Arthritis ICD9 codes	714.x	1.33
25	# of Sacroiliitis, not elsewhere classified ICD9 codes	720.2x	1.31
26	HLA-B27 (positive/ negative/ pending or uncertain)		1.24
27	# of Cervicalgia ICD9 codes	723.1x	1.13
28	Exposure to ≥ 1 non-biologic DMARD		1.07
29	# of Other specified inflammatory spondyloarthropathies ICD9 codes	720.8x	1.04
30	Race (White/ Black/ Other Race/ Unknown)		1.02
31	# of Ulcerative Colitis ICD 9 codes	556.x	0.98
32	# of Spinal stenosis lumbar thoracic cervical ICD9 codes	723.0, 724.0x	0.93
33	# of Neuritis or radiculitis ICD9 codes	723.4, 724.4	0.82
34	Cyclic Citrullinated Peptide antibody (pos/ neg/ pending/ uncertain)		0.80
35	# of Acute anterior uveitis, (non-infectious) ICD9 codes	364.00x, 364.01x, 364.02x, 364.04x, 364.05x	0.74
36	# Connective Tissue Disease ICD9 codes	710.x	0.67
37	Ethnicity (Non-Hispanic/ Hispanic/ Unknown)		0.54

38	Gender (male/ female)		0.53
39	# of Psoriatic arthritis ICD9 codes	696.0x	0.46
40	# of Gout ICD9 codes	274.x	0.40
41	# of Thoracic pain ICD9 codes	724.1x	0.35
42	# of Crohn’s disease ICD9 codes	555.x	0.32
43	# of Arthropathy associated with Crohn’s or ulcerative colitis ICD9 codes	713.1 AND either 555.x or 556.x	0.28
44	# of Polymyalgia Rheumatica ICD9 codes	725	0.14
45	Exposure to allopurinol or colchicine (yes/no)		0.11
46	# of Dorsalgias, unspecified ICD9 codes	724.9x, 723.9x	0.07
47	# of Paget’s disease ICD9 codes	731.0	0.01
48	# of Sarcoidosis ICD9 codes	135	0.01
49	# of Vasculitis ICD9 codes	273.2, 446.0, 446.4, 446.5, 446.7	0.01

*The variable importance rankings differed for the Full Algorithm and the High Feasibility Algorithm, since different sets of variables were assessed (i.e. the Full Algorithm included coded and NLP variables, while the High Feasibility Algorithm included coded variables only).

Supplementary Data 2. Method summary and R code for identifying patients with axial spondyloarthritis

We randomly divided the 600 patients with known axSpA status (yes vs. no) into training and testing subsets (75% vs. 25%).

For the Full Algorithm, we used 5-fold cross validation to find the best parameter `mtry` (number of variables randomly sampled as candidates at each split). Next, we built the algorithm with Random Forest in the training dataset and validated the algorithm in the testing dataset.

For the High Feasibility Algorithm, we selected the 16 variables with the highest importance scores (Mean Decrease Gini). We used error rates to guide the cutoff for the number of variables included in the algorithm. Subsequently, we selected the best `mtry`, built the algorithm with Random Forest in the training dataset and validated the algorithm in the testing dataset.

The Spond* NLP Algorithm included only the Spond NLP variable developed within a different dataset on the snippet level [Walsh et al. Arthritis Care Res (Hoboken) 2017]. With this project, we validated the Spond NLP Algorithm on the patient level.

We used R (version 3.5.1) with Random Forest (version 4.6-14) and Caret (version 6.0-80) packages.

R code

```
rm(list = ls())

## 00_1 loading packages

library(RODBC)

library(ggplot2)

library(caret)

library(lattice)

library(ROCR)

library(randomForest)

library(boot)

library(dplyr)

library(gsubfn)

library(beanplot)

## 00_2 function for confusion matrix calculation

confusion_cal = function(confusion){

  sen <- Hmisc::binconf(x=confusion[2,2], n=confusion[2,2]+confusion[2,1])

  spe <- Hmisc::binconf(x=confusion[1,1], n=confusion[1,1]+confusion[1,2])
  ppv <- Hmisc::binconf(x=confusion[2,2], n=confusion[1,2] + confusion[2,2])

  npv <- Hmisc::binconf(x=confusion[1,1], n=confusion[1,1] + confusion[2,1])

  acc <- Hmisc::binconf(x=confusion[1,1]+confusion[2,2],
n=confusion[1,1]+confusion[2,2]+confusion[1,2]+confusion[2,1])

  fp <- Hmisc::binconf(x=confusion[1,2],
n=confusion[1,1]+confusion[2,2]+confusion[1,2]+confusion[2,1])

  fn <- Hmisc::binconf(x=confusion[2,1],
n=confusion[1,1]+confusion[2,2]+confusion[1,2]+confusion[2,1])

  df<-data.frame(c(sen[1,1], sen[1,2], sen[1,3]),c(spe[1,1], spe[1,2],
spe[1,3]),c(ppv[1,1],ppv[1,2],ppv[1,3]),c(npv[1,1],
npv[1,2],npv[1,3]),c(acc[1,1],acc[1,2],acc[1,3]),c(fp[1,1],fp[1,2],fp[1,3]),c(fn[1,1],fn[1,2],fn[1,3]))

  names(df)<-c("sensitivity","specificity","ppv","npv","accuracy","false_positive","false_negative")

  row.names(df)<-c("value","CI_lo","CI_hi")

  return(df)

}

## 00_3 function for bootstrapping in algorithm performance estimation (on training and testing
subsets)

bootstrap_validation <- function (training_pf, testing_pf,loop_number,
training_data,testing_data,algorithm,mtry_set){

  names(training_pf) <-
c("sensitivity","specificity","ppv","npv","accuracy","false_positive","false_negative")

  names(testing_pf) <-
c("sensitivity","specificity","ppv","npv","accuracy","false_positive","false_negative")

    if (algorithm == "spond") {

    for (i in (1:loop_number)) { #update loop number

      training_sample <- sample_n(training_data, nrow(training_data), replace=TRUE) #update

      testing_sample <- sample_n(testing_data, nrow(testing_data), replace = TRUE) #update

      #building trees

      seed <- 9

      set.seed(seed)

      rf_random <- train(AxSpA ~ ., data = training_sample, method = "rf", metric = "Accuracy")

      #training performance on training group

      AxSpA.pred <- predict(rf_random, newdata = training_sample)
      AxSpA.confusion_training_cal <- confusion_cal(table(observed = training_sample[,1],
predict=AxSpA.pred))

      # AxSpA.confusion_training_cal <- confusion_cal(with(training_sample, table(AxSpA,
spond)))

      training_pf <- rbind(training_pf, AxSpA.confusion_training_cal[1,])

      ##testing performance on testing group

      AxSpA.pred <- predict(rf_random, newdata = testing_sample)

      AxSpA.confusion_testing_cal <- confusion_cal(table(observed =
testing_sample[,1],predict=AxSpA.pred))

      # AxSpA.confusion_testing_cal <- confusion_cal(with(testing_sample, table(AxSpA,
spond)))

      testing_pf <- rbind(testing_pf, AxSpA.confusion_testing_cal[1,])

      if (i %% 5 == 0) {print (i)}

    }

  } else {

    for (i in (1:loop_number)) {

      training_sample <- sample_n(training_data, nrow(training_data), replace=TRUE)

      testing_sample <- sample_n(testing_data, nrow(testing_data), replace = TRUE)

      training_x <- training_sample[,-dim(training)[2]]

      training_y <- training_sample[,dim(training)[2]]

      testing_x <- testing_sample[,-dim(training)[2]]

      testing_y <- testing_sample[,dim(training)[2]]

      ##building trees

      seed <- 9

      set.seed(seed)

      AxSpA.rf <- randomForest(x = training_x, y = training_y, xtest=testing_x, ytest = testing_y,
ntree = 1500, mtry = mtry_set, keep.forest = TRUE)

      # AxSpA.rf

      ##training performance on training group

      AxSpA.confusion_training <- AxSpA.rf$confusion

      AxSpA.confusion_training_cal <- confusion_cal(AxSpA.rf$confusion)

      training_pf <- rbind(training_pf, AxSpA.confusion_training_cal[1,])
      ##testing performance on testing group
      AxSpA.confusion_testing <- AxSpA.rf$test$confusion
      AxSpA.confusion_testing_cal <- confusion_cal(AxSpA.rf$test$confusion)
      testing_pf <- rbind(testing_pf, AxSpA.confusion_testing_cal[1,])
      if (i %% 5 == 0) {print (i)}
    }
  }
  return(list(training_pf, testing_pf))
}
## 00_4 data prepare
datat <- readRDS(file ="00_Data/axSpAanalytic2.rds")
## 00_5 prepare training and testing subset
set.seed(572)
intrain_1 <- caret::createDataPartition(y=datat$AxSpA, p = 0.75, list=FALSE)
intrain <- intrain_1
training_ori <- datat[intrain,]
testing_ori <- datat[-intrain,]
## 1 model with full variables
## 1_0 variables
slim_var<-c(
  "synthetic_DMARD", "biologic_DMARD","crystal_arthritis_med"##3
  ,"lastHLAB27","lastRF","lastCCP","cnt_CRP","cnt_ESR"##5
  ,"Sacroiliitis, not elsewhere classified"
  ,"Other specified inflammatory spondyloarthropathies"
  ,"Unspecified inflammatory spondyloarthopathy","Psoriatic arthritis"
  ,"Athropathy associated with Crohns or UC (AND)","Arthropathy associated with Reiters and
nonspecific urethritis"
  ,"Chrons","Ulcerative Colitis","Acute anterior uveitis, (non-infectious)"
  ,"Psoriasis","Low back pain","Thoracic pain","Dorsalgias, unspecified","Cervicalgia","Backache,
unspecified"##15
  ,"Osteoarthrosis and allied disorders","Osteoarthrosis of the spine (includes DISH)"
  ,"Intervertebral disk disorders","Spinal stenosis lumbar thoracic cervical"
  ,"Sciatica","Neuritis or radiculitis","Gout","RA"##8
  ,"AS_","CTD_","Vasculitis","PMR","Pagets","Sarcoidosis"##6
  ,"b27_PatLev_new","spond_patLev_new","sacroiliitis_patLev_new" ##3
  ,"ageAtIndexDate","ethnicityClass_PatLev","raceClass_PatLev_m","Gender"##4
  ,"RheumVisitCount","visit_cnt"##2
  ,"rheumatic_disease_comorbidity_score"##1
  ,"districts" ##1
  ,"duration" ##1
  ,"AxSpA", "groupName" ##2
)
## 1_1 dataset
training_slim <- training_ori[, which(names(training_ori) %in% slim_var)]
training_slim$AxSpA <- as.factor(training_slim$AxSpA)
training <- training_slim[,-1]
testing_slim <- testing_ori[,which(names(testing_ori) %in% slim_var)]
testing_slim$AxSpA <- as.factor(testing_slim$AxSpA)
testing <- testing_slim[,-1]
cat("the number of variables: ", (ncol(training) -1), "\n")
cat("the size of training dataset: ", nrow(training), "; testing dataset: ", nrow(testing), "\n")
cat("the size of positive and negative in training dataset: ", nrow(training[which(training$AxSpA
== 1),]), " ", nrow(training[which(training$AxSpA == 0),]), "\n")
cat("the size of positive and negative in testing dataset: ", nrow(testing[which(testing$AxSpA ==
1),]), " ", nrow(testing[which(testing$AxSpA == 0),]), "\n")
## 1_2 ##find the best mtry by caret::train
seed <- 7
metric <- "Accuracy"
trCtrl <- trainControl(method = "repeatedcv",number = 5, repeats = 3, search =
"random")##3repeats
set.seed(seed)
tunegridt <- expand.grid(.mtry = (10:25))
```

```
rf_random<-train(AxSpA ~ ., data = training, method = "rf", metric = metric
             , tuneGrid = tunegridt, trControl = trCtrl, preProcess = c("center","scale"), ntree =
1500)
rfPredict_2<-predict(rf_random, newdata = testing)
rf_random.confusionMatrix <- confusionMatrix(rfPredict_2, testing$AxSpA)
# ##find the best mtry by caret::train end --##according result above, select mtry = 21
## 1_3 training and testing
training_x <- training[,-dim(training)[2]]
training_y <- training[,dim(training)[2]]
testing_x <- testing[,-dim(training)[2]]
testing_y <- testing[,dim(training)[2]]
##building trees
seed <- 9
set.seed(seed)
AxSpA.rf <- randomForest(x = training_x, y = training_y, xtest=testing_x, ytest = testing_y, ntree
= 1500, mtry = 21, keep.forest = TRUE)
AxSpA.rf
imp_slim_full<-importance(AxSpA.rf, type=2)
##training performance on training group
AxSpA.confusion_slim_training <- AxSpA.rf$confusion
AxSpA.confusion_slim_training_cal <- confusion_cal(AxSpA.rf$confusion)
##testing performance on testing group
AxSpA.confusion_slim <- AxSpA.rf$test$confusion
AxSpA.confusion_slim_cal <- confusion_cal(AxSpA.rf$test$confusion)
##C-statistic on testing
AxSpAPredict<-predict(AxSpA.rf,testing,type="prob")
AxSpA.pr<-AxSpAPredict[,2]
AxSpA.pred <- prediction(AxSpA.pr, testing$AxSpA)
AxSpA.perf <- performance(AxSpA.pred,"tpr","fpr")
AxSpA.perf_AUC <- performance(AxSpA.pred,"auc") #calculate the AUC value
AxSpA.AUC <- AxSpA.perf_AUC@y.values[[1]]
AxSpA.perf_slim <- AxSpA.perf

AxSpA.AUC_slim <- AxSpA.AUC

##bootstrapping for training and testing performance

list[full_train,full_test] <-
bootstrap_validation(AxSpA.confusion_slim_training_cal[1,],AxSpA.confusion_slim_cal[1,],500,
training,testing,"full",21)

## 2 model with high feasibility variables

## 2_0 select variables

training_x <- training[,-c(50,40,39,38)]#remove NLP related variables

training_y <- training[,50]

testing_x <- testing[,-c(50,40,39,38)]#remove NLP related variables

testing_y <- testing[,50]

RF=randomForest(x = training_x, y = training_y, xtest=testing_x, ytest = testing_y, ntree = 1500,
keep.forest = TRUE);

RF

imp_slim<-importance(RF, type=2)

## pick the right number of variables

set.seed(600);

Cum=data.frame()

for (i in 1:10)

{res=rfcv(trainx=training_x, trainy=training_y, cv.fold=5, scale = "log", step=0.7, ntree=1000);

out=data.frame(Nvar=res$n.var,Err.cv=res$error.cv)[order(res$error.cv),];out

Cum=rbind(Cum, out)};

dim(Cum)

avg=tapply(Cum$Err.cv, Cum$Nvar, mean)

Newcum=data.frame(N=as.numeric(names(avg)), avg);

with(Newcum[c(4:11),], plot(N, avg,  type="o", lwd=1, xlab="Number of predictors", ylab="CV
error rate"))

##according to the results above, select important variables listed as below:

supper_feasible <- c("AS_", "RheumVisitCount",
"cnt_CRP","biologic_DMARD","visit_cnt","ageAtIndexDate"

                    ,"cnt_ESR","lastHLAB27","rheumatic_disease_comorbidity_score"
```

```
                  ,"Osteoarthrosis and allied disorders","Low back pain","districts","duration"
                  ,"Unspecified inflammatory spondyloarthopathy","Osteoarthrosis of the spine
(includes DISH)"
                  ,"Cervicalgia","AxSpA","groupName"
)
## 2_1 dataset
training_sf <- training_ori[, which(names(training_ori) %in% supper_feasible)]
training_sf$AxSpA <- as.factor(training_sf$AxSpA)
training <- training_sf[,-1]
testing_sf <- testing_ori[,which(names(testing_ori) %in% supper_feasible)]
testing_sf$AxSpA <- as.factor(testing_sf$AxSpA)
testing <- testing_sf[,-1]
cat("the number of variables: ", (ncol(testing)-1), "\n")
cat("the size of training dataset: ", nrow(training), "; testing dataset: ", nrow(testing), "\n")
cat("the size of positive and negative in training dataset: ", nrow(training[which(training$AxSpA
== 1),]), " ", nrow(training[which(training$AxSpA == 0),]), "\n")
cat("the size of positive and negative in testing dataset: ", nrow(testing[which(testing$AxSpA ==
1),]), " ", nrow(testing[which(testing$AxSpA == 0),]), "\n")
## 2_2 ##find the best mtry by caret::train
seed <- 7
metric <- "Accuracy"
trCtrl <- trainControl(method = "repeatedcv",number = 5, repeats = 3, search =
"random")##3repeats
set.seed(seed)
tunegridt <- expand.grid(.mtry = (2:11))
rf_random<-train(AxSpA ~ ., data = training, method = "rf", metric = metric
               , tuneGrid = tunegridt, trControl = trCtrl, preProcess = c("center","scale"), ntree =
1500)
rfPredict_2<-predict(rf_random, newdata = testing)
rf_random.confusionMatrix <- confusionMatrix(rfPredict_2, testing$AxSpA)
# ##find the best mtry by caret::train end ##according result above, select mtry = 8
## 2_3 training and testing
```

```
training_x <- training[,-dim(training)[2]]

training_y <- training[,dim(training)[2]]

testing_x <- testing[,-dim(training)[2]]

testing_y <- testing[,dim(training)[2]]

##building trees

seed <- 9

set.seed(seed)

AxSpA.rf <- randomForest(x = training_x, y = training_y, xtest=testing_x, ytest = testing_y, ntree
= 1500, mtry = 8, keep.forest = TRUE)

AxSpA.rf

##training performance on training group

AxSpA.confusion_superFeasible_training <- AxSpA.rf$confusion

AxSpA.confusion_superFeasible_training_cal <- confusion_cal(AxSpA.rf$confusion)

##testing performance on testing group

AxSpA.confusion_superFeasible <- AxSpA.rf$test$confusion

AxSpA.confusion_superFeasible_cal <- confusion_cal(AxSpA.rf$test$confusion)

##C-statistic on testing data set

AxSpAPredict<-predict(AxSpA.rf,testing,type="prob")

AxSpA.pr<-AxSpAPredict[,2]

AxSpA.pred <- prediction(AxSpA.pr, testing$AxSpA)

AxSpA.perf <- performance(AxSpA.pred,"tpr","fpr")

AxSpA.perf_AUC <- performance(AxSpA.pred,"auc") #calculate the AUC value

AxSpA.AUC <- AxSpA.perf_AUC@y.values[[1]]

AxSpA.perf_superFeasible <- AxSpA.perf

AxSpA.AUC_superFeasible <- AxSpA.AUC

##bootstrapping for training and testing performance

list[feasible_train,feasible_test] <-
bootstrap_validation(AxSpA.confusion_superFeasible_training_cal[1,],AxSpA.confusion_superF
easible_cal[1,],500, training,testing,"feasible",8)

## 3 model with Spond* NLP variables

## 3_0 select variables
```

```
patLev_nlp<-c("spond_patLev_new","AxSpA","groupName")

## 3_1 dataset

datat_all<-datat

datat_all$AxSpA <- as.factor(datat_all$AxSpA)

datat_all$spond [datat_all$spond_patLev_new == 1] <- 1

datat_all$spond [datat_all$spond_patLev_new == 3 | datat_all$spond_patLev_new == 2 |
datat_all$spond_patLev_new == 0] <- 0

##prepare the subset

training_spond <- training_ori[, which(names(training_ori) %in% patLev_nlp)]

training_spond$AxSpA <- as.factor(training_spond$AxSpA)

training_spond$spond [training_spond$spond_patLev_new == 1] <- 1

training_spond$spond [training_spond$spond_patLev_new == 3 |
training_spond$spond_patLev_new == 2 | training_spond$spond_patLev_new == 0] <- 0

training <- training_spond[,-c(1,2)]

testing_spond <- testing_ori[, which(names(testing_ori) %in% patLev_nlp)]

testing_spond$AxSpA <- as.factor(testing_spond$AxSpA)

testing_spond$spond [testing_spond$spond_patLev_new == 1] <- 1

testing_spond$spond [testing_spond$spond_patLev_new == 3
|testing_spond$spond_patLev_new == 2 | testing_spond$spond_patLev_new == 0] <- 0

testing <- testing_spond[,-c(1,2)]

cat("the number of varibles: ", (ncol(training)-1), "\n")

cat("the size of training dataset: ", nrow(training), "; testing dataset: ", nrow(testing), "\n")

cat("the size of positive and negative in training dataset: ", nrow(training[which(training$AxSpA
== 1),]), " ", nrow(training[which(training$AxSpA == 0),]), "\n")

cat("the size of positive and negative in testing dataset: ", nrow(testing[which(testing$AxSpA ==
1),]), " ", nrow(testing[which(testing$AxSpA == 0),]), "\n")

## 3_2 training and testing

##build random forest

rf_random <- train(AxSpA ~ ., data = training, method = "rf", metric = "Accuracy")

##training performance on training group

AxSpA.pred <- predict(rf_random, newdata = training)

AxSpA.confusion_spond_training <- table(observed = training[,1], predict = AxSpA.pred)
AxSpA.confusion_spond_training_cal <- confusion_cal(table(observed = training[,1], predict = AxSpA.pred))

##testing performance on testing group

AxSpA.pred <- predict(rf_random, newdata = testing)

AxSpA.confusion_spond <- table(observed = testing[,1], predict = AxSpA.pred)

AxSpA.confusion_spond_cal <- confusion_cal(table(observed = testing[,1], predict = AxSpA.pred))

##C-statistic

AxSpAPredict<-predict(rf_random, newdata = testing, type="prob")

AxSpA.pr<-AxSpAPredict[,2]

AxSpA.pred <- prediction(AxSpA.pr, testing$AxSpA)

AxSpA.perf <- performance(AxSpA.pred,"tpr","fpr")

AxSpA.perf_AUC <- performance(AxSpA.pred,"auc") #calculate the AUC value

AxSpA.AUC <- AxSpA.perf_AUC@y.values[[1]]

AxSpA.perf_spond <- AxSpA.perf

AxSpA.AUC_spond <- AxSpA.AUC

## bootstrapping for training and testing performance

list[spond_train,spond_test] <- bootstrap_validation(AxSpA.confusion_spond_training_cal[1,],AxSpA.confusion_spond_cal[1,],500, training,testing,"spond",21)

## 4 graph and table

## 4_1 The Full algorithm performance

## There are 49 variables from Medications, labs, ICDs, NLPs, demographics, visiting and comorbidity

##performance on training group

AxSpA.confusion_slim_training

AxSpA.confusion_slim_training_cal

##performance on testing group

AxSpA.confusion_slim

AxSpA.confusion_slim_cal

## 4_2 The Hihg feasibility algorithm

#The 16 super Feasible variables are
```

```
#"AS_", "RheumVisitCount", "cnt_CRP","biologic_DMARD","visit_cnt","ageAtIndexDate"

#,"cnt_ESR","lastHLAB27","rheumatic_disease_comorbidity_score"

#,"Osteoarthrosis and allied disorders","Low back pain","districts","duration"

#,"Unspecified inflammatory spondyloarthopathy","Osteoarthrosis of the spine (includes DISH)"

#,"Cervicalgia"

##performance on training group

AxSpA.confusion_superFeasible_training

AxSpA.confusion_superFeasible_training_cal

##performance on testing group

AxSpA.confusion_superFeasible

AxSpA.confusion_superFeasible_cal

## 4_3 The spond NLP performance

##performance on training group

AxSpA.confusion_spond_training

AxSpA.confusion_spond_training_cal

##performance on testing group

AxSpA.confusion_spond

AxSpA.confusion_spond_cal

## 4_4 C-statistic analysis on testing dataset

# get the range for the x and y axis

xrange <- range(AxSpA.perf_slim@x.values[[1]])

yrange <- range(AxSpA.perf_slim@y.values[[1]])

# set up the plot

plot(xrange,yrange, type="n",xlab="False Positive Rate", ylab="Ture Positive Rate",
mgp=c(2,1,0))

lines(AxSpA.perf_spond@x.values[[1]],AxSpA.perf_spond@y.values[[1]], lty=5,lwd=2)

lines(AxSpA.perf_superFeasible@x.values[[1]],AxSpA.perf_superFeasible@y.values[[1]],lty=3,l
wd = 2)

lines(AxSpA.perf_slim@x.values[[1]],AxSpA.perf_slim@y.values[[1]],lty=1,lwd = 2)

legend("bottomright",legend=c("Spond NLP Algorithm","High Feasibility Algorithm", "Full
Algorithm"),lty=c(5,3,1), lwd=c(2,2),col=c(1,1,1))
```

```
abline(a=0, b=1,lwd=2,lty=2,col="gray")

title("ROC curve", "AUC: Spond NLP Algorithm=0.86; High Feasibility Algorithm=0.94; Full
Algorithm=0.96")

# AxSpA.AUC_spond ##0.8649

# AxSpA.AUC_slim ##0.9580

# AxSpA.AUC_superFeasible ##0.9368

## 4_5 algorithms performance in training and test by beanplot

names(full_test) <- c("Sensitivity","Specificity","PPV","NPV","Concordance","False
Positive","False Negative")

names(feasible_test) <- c("Sensitivity","Specificity","PPV","NPV","Concordance","False
Positive","False Negative")

names(spond_test) <- c("Sensitivity","Specificity","PPV","NPV","Concordance","False
Positive","False Negative")

full_test$alg <- "Full Algorithm"

feasible_test$alg <- "High Feasibility Algorithm"

spond_test$alg <- "Spond NLP Algorithm"

full_test <- full_test[2:501,] %>% tidyr::gather("type","performance",1:7)

feasible_test <- feasible_test[2:501,] %>% tidyr::gather("type","performance",1:7)

spond_test <- spond_test[2:501,] %>% tidyr::gather("type","performance",1:7)

test <- rbind(rbind(full_test, feasible_test), spond_test)

test$performance <- 100*test$performance

test$alg <- factor(test$alg,levels = c("Full Algorithm","High Feasibility Algorithm","Spond NLP
Algorithm"))

test$type <- factor(test$type,levels =
c("Sensitivity","Specificity","PPV","NPV","Concordance","False Positive","False Negative"))

test_perf <- test %>% filter(type %in% c("Sensitivity","Specificity","PPV","NPV"))

test_concor <- test %>% filter(type %in% c("Concordance","False Positive","False Negative"))

names(full_train) <- c("Sensitivity","Specificity","PPV","NPV","Concordance","False
Positive","False Negative")

names(feasible_train) <- c("Sensitivity","Specificity","PPV","NPV","Concordance","False
Positive","False Negative")

names(spond_train) <- c("Sensitivity","Specificity","PPV","NPV","Concordance","False
Positive","False Negative")
```

```
full_train$alg <- "Full Algorithm"

feasible_train$alg <- "High Feasibility Algorithm"

spond_train$alg <- "Spond NLP Algorithm"

full_train <- full_train[2:501,] %>% tidyr::gather("type","performance",1:7)

feasible_train <- feasible_train[2:501,] %>% tidyr::gather("type","performance",1:7)

spond_train <- spond_train[2:501,] %>% tidyr::gather("type","performance",1:7)

train <- rbind(rbind(full_train, feasible_train), spond_train)

train$performance <- 100*train$performance

train$alg <- factor(test$alg,levels = c("Full Algorithm","High Feasibility Algorithm","Spond NLP
Algorithm"))

train$type <- factor(test$type,levels =
c("Sensitivity","Specificity","PPV","NPV","Concordance","False Positive","False Negative"))

train_perf <- train %>% filter(type %in% c("Sensitivity","Specificity","PPV","NPV"))

train_concor <- train %>% filter(type %in% c("Concordance","False Positive","False Negative"))

train_mean <-  train %>% group_by(alg,type) %>% summarise(mean_ = mean(performance),
delta = 1.96*sd(performance))

test_mean <-  test %>% group_by(alg,type) %>% summarise(mean_ = mean(performance),
delta = 1.96*sd(performance))

#beanplot for algorithm performance in training and testing subset.

ggplot(train_perf,aes(stringr::str_wrap(alg,5), performance)) + geom_violin() + facet_grid(~type)
+ scale_y_continuous(limits = c(0,100), breaks = seq(0,100, by=20)) + theme(axis.text.x =
element_text(size = 8, color = "black"), axis.text.y = element_text(size = 15, color = "black"), text
= element_text(size = 15, color = "black") ) + xlab(NULL) + ylab(NULL)

ggplot(test_perf,aes(stringr::str_wrap(alg,5), performance)) + geom_violin() + facet_grid(~type)
+ scale_y_continuous(limits = c(0,100), breaks = seq(0,100, by=20))+ theme(axis.text.x =
element_text(size = 8, color = "black"), axis.text.y = element_text(size = 15, color = "black"), text
= element_text(size = 15, color = "black") ) + xlab(NULL) + ylab(NULL)

#beanplot for algorithms' concordance in training and testing subset

ggplot(train_concor,aes(stringr::str_wrap(type,5), performance)) + geom_violin() +
facet_grid(~alg) + scale_y_continuous(limits = c(0,100), breaks = seq(0,100, by=20)) +
theme(axis.text.x = element_text(size = 10, color = "black"), axis.text.y = element_text(size = 15,
color = "black"), text = element_text(size = 15, color = "black") ) + xlab(NULL) + ylab(NULL)

ggplot(test_concor,aes(stringr::str_wrap(type,5), performance)) + geom_violin() +
facet_grid(~alg) + scale_y_continuous(limits = c(0,100), breaks = seq(0,100, by=20))+
theme(axis.text.x = element_text(size = 10, color = "black"), axis.text.y = element_text(size = 15,
color = "black"), text = element_text(size = 15, color = "black") ) + xlab(NULL) + ylab(NULL)
```