# RF_Cirrosis

Sergio Viana Llovell

2/5/2022

# Random forest model - Diabetes Dataset

In this appendix we will analyse the results of running the Randomforest technique on the 4 different preprocessing on the data we have on the cirrhosis disease. The procedure followed on the 4 preprocessed samples is the same as the one explained in the main report, so we will not repeat the explanation.

In all the pre-processing of the data we have, we are going to divide into train set and test set to perform the validation of the model, in our case, the test partition corresponds to 20% of the total observations we have.

# S1 Filter Process

First, we will fit the model on the data filtered with the S1 technique.

```
library(dbplyr)
library(knitr)
library(dplyr)
library(caret)
library(clValid)
library(stats)
library(vip)
library(randomForest)
library(ROCR)
library(ropls)
library(pROC)
library(MLmetrics)

load("C:/Users/sergi/OneDrive/Escritorio/UPV/3_curso/PROY3/DatosSinOutlier.RData")

set.seed(110)
clr=as.data.frame(t(DatosFinal$Cirrosis$S1_NORMAL$S1_CLR_CIRR))
a=Datos_sample$Cirrosis$cirrhotic
clr$disease=factor(a)
colnames(clr)=sub(' ','',colnames(clr))
trainFilas = createDataPartition(clr$disease, p=0.8, list=FALSE)
clrTrain = clr[trainFilas,]
clrTest = clr[-trainFilas,]


tss=as.data.frame(log(as.data.frame(t(DatosFinal$Cirrosis$S1_NORMAL$S1_TSS_CIRR)) * 10e6 + 1
))
tss$disease=factor(a)
colnames(tss)=sub(' ','',colnames(tss))
tssTrain =tss[trainFilas,]
tssTest = tss[-trainFilas,]
```

## TSS

First, we will focus on the TSS preprocessing. After performing the appropriate numerical transformation for this preprocessing, we proceed to training by repeated cross-validation to adjust the hyperparameters of the model. The results can be seen below:

```
library(e1071)
library(ranger)
set.seed(200)

tr_fit = trainControl(method= 'repeatedcv',repeats=5,search='grid',number = 10, classProbs = TRUE,summaryFunction = twoClassSummary)
best_model = train(disease~., data=tssTrain, trControl = tr_fit,method = 'ranger',metric = 'ROC')
best_model
```

```
## Random Forest
##
## 187 samples
## 530 predictors
##   2 classes: 'n', 'y'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 169, 168, 167, 169, 169, 168, ...
## Resampling results across tuning parameters:
##
##   mtry  splitrule   ROC        Sens       Spec
##     2   gini        0.9431531  0.9591111  0.7555556
##     2   extratrees  0.9245086  0.9740000  0.6988889
##    32   gini        0.9438321  0.9168889  0.8255556
##    32   extratrees  0.9356111  0.9411111  0.7780000
##   530   gini        0.9338741  0.8975556  0.8193333
##   530   extratrees  0.9363309  0.9260000  0.7964444
##
## Tuning parameter 'min.node.size' was held constant at a value of 1
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were mtry = 32, splitrule = gini
##  and min.node.size = 1.
```

Next, we train the model that gave us the best AUC (0.9438) when cross-validated, with the entire training set, and predict the test set.

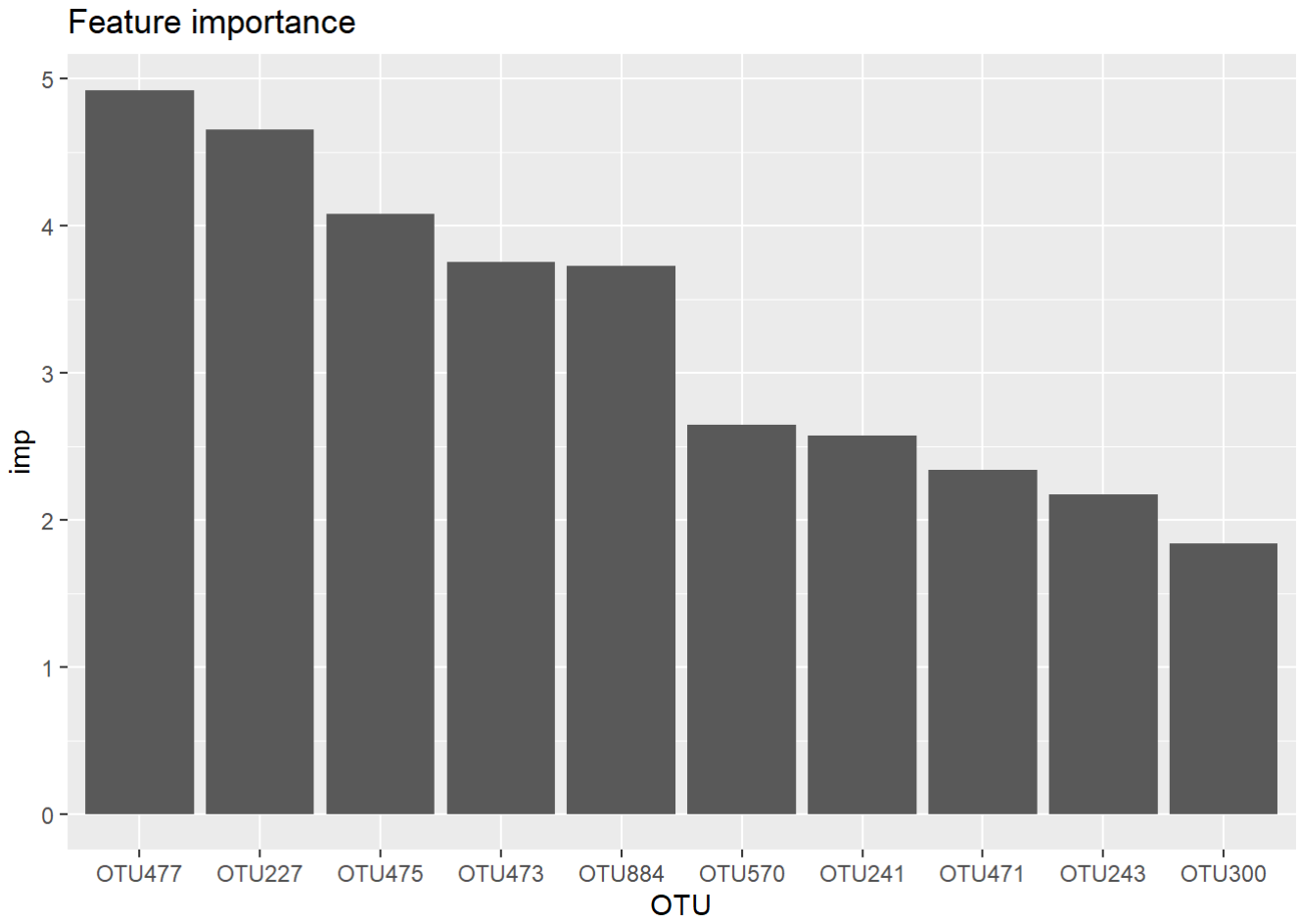
```
modelo_final = ranger(disease~.,data = tssTrain, mtry = 32,splitrule = 'gini',min.node.size = 1,importance = 'impurity',probability=TRUE,keep.inbag = TRUE)

pred = predict(modelo_final, data=tssTest, type="response")
pred = pred$predictions
ROC1_rf_mej <- roc(tssTest$disease,pred[,2])
a_rf_mej=auc(ROC1_rf_mej)
a_rf_mej
```

```
## Area under the curve: 0.9427
```

With the test set we managed to achieve an AUC of 0.9447, but we can try to improve it by training different random forest models, using only the variables that are most important in this model that we have just trained. Next, we calculate how many variables have an importance greater than 0, and we plot the 10 most important ones in a graph:

```
library(ggplot2)
imp=sort(importance(modelo_final),decreasing=TRUE)
d=as.data.frame(imp)
d$otu = rownames(d)
p = ggplot(d[1:10,],aes(y=imp,x=reorder(otu,-imp))) + geom_col() + ggtitle('Feature importanc
e') + xlab("OTU")
p
```




```
nrow(as.data.frame(d[which(d$imp>0),]))
```

```
## [1] 378
```

There are 371 variables with a Gini index (variable importance) greater than 0, so we now train different random forest models, selecting from the 10 most important variables, up to 371. All this training is still done by repeated cross-validation, as we have done with the previous model. The best model resulting from all those we have trained by filtering variables can be seen below:

```
max_auc = 0
variables=0
params=0
tr_fit = trainControl(method= 'repeatedcv',repeats=5,search='grid',number = 10, classProbs =
TRUE,summaryFunction = twoClassSummary)
for (i in seq(10,371,15)){
  data= tssTrain[,c(rownames(d)[1:i],'disease')]
  best_model = train(disease~., data=data, trControl = tr_fit,method = 'ranger',metric = 'RO
C')
  if (max(best_model$results$ROC)>max_auc){

    max_auc=max(best_model$results$ROC)
    variables=i
    params= best_model$bestTune
  }
}
print(paste('AUC= ',max_auc))
```

```
## [1] "AUC=  0.957172839506173"
```

```
print(paste('NVars= ',variables))
```

```
## [1] "NVars=  40"
```

```
params
```

```
##   mtry splitrule min.node.size
## 1    2      gini             1
```

The best result has been achieved using the 70 most important variables in the previous model and with the parameters:

- mtry = 2
- Splitrule = gini
- Min node size = 1

With an AUC score of 0.957 in the cross-validation, a value about 7% better than the one obtained using all variables. Next let's see what AUC score we get with the test set:

```
set.seed(30)
data_filt = tssTrain[,c(rownames(d)[1:70],'disease')]
modelo_final = ranger(disease~.,data = data_filt,mtry = 2,splitrule = 'gini',min.node.size =
1,importance = 'impurity',probability=TRUE,keep.inbag = TRUE)

pred = predict(modelo_final, data=tssTest[,c(rownames(d)[1:70])], type="response")
pred = pred$predictions
ROC1_rf_mej <- roc(tssTest$disease,pred[,2])
a_rf_mej=auc(ROC1_rf_mej)
print(paste('AUC: ',a_rf_mej))
```

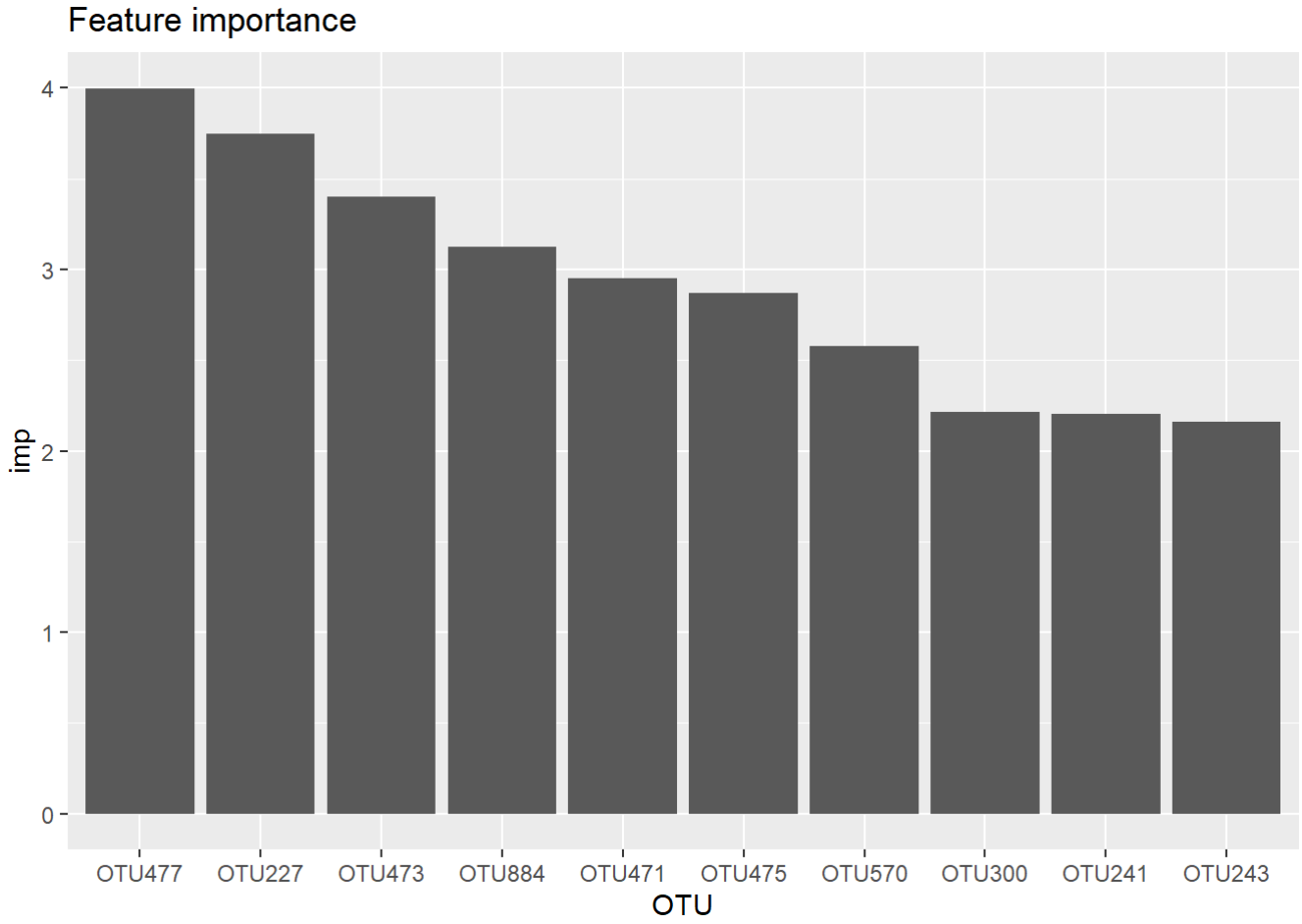
```
## [1] "AUC:  0.944664031620553"
```

```
print(paste('F1 Score: ',F1_Score(tssTest$disease,factor(ifelse(pred[,1] < 0.5, 'y', 'n'), levels = c('n','y')))))
```

```
## [1] "F1 Score:  0.904761904761905"
```

Finally, training the model with the entire training set and making predictions on the test set, we obtain an area under the curve of 0.954 and an F1 Score of 0.904, values slightly higher than those obtained with the model without filtering variables, and higher than the results obtained by Pasolli in his study. To conclude this preprocessing, we will now visualise and briefly explain the 10 most important species of microorganisms in our model:

```
imp=sort(ranger::importance(modelo_final),decreasing=TRUE)
d=as.data.frame(imp)
d$otu = rownames(d)
p = ggplot(d[1:10,],aes(y=imp,x=reorder(otu,-imp))) + geom_col() + ggtitle('Feature importance') + xlab('OTU')
p
```




As we can see in the graph, the 2 first variables have a similar importance, it being a little bit higher than the rest. When integrating the name of the OTUS with the data frame ‘Data_taxa’, we find that the most influential microorganisms are Veillonella; an anaerobic, Gram-negative diplococci found in the human oral, gastrointestinal, and vaginal microbiota. It has been correlated with disease severity and hepatic encephalopathy in liver diseases such as autoimmune hepatitis and cirrhosis. Their abundance has also been recently observed to be increased in alcoholic hepatitis, where postinflammatory infections are known to impact mortality.

# CLR

Secondly, we are going to perform the same process that we have just done with the CLR preprocessing. First, we perform the repeated cross-validation with the training data using all available variables:

```
set.seed(100)
best_model = train(disease~., data=clrTrain, trControl = tr_fit,method = 'ranger',metric = 'R
OC')
best_model
```

```
## Random Forest
##
## 187 samples
## 530 predictors
##   2 classes: 'n', 'y'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 169, 168, 168, 169, 169, 169, ...
## Resampling results across tuning parameters:
##
##   mtry  splitrule   ROC        Sens       Spec
##     2   gini        0.9408025  0.9582222  0.7644444
##     2   extratrees  0.9230741  0.9800000  0.6911111
##    32   gini        0.9450531  0.9097778  0.8228889
##    32   extratrees  0.9338407  0.9382222  0.7833333
##   530   gini        0.9221877  0.8722222  0.8233333
##   530   extratrees  0.9324185  0.9097778  0.7935556
##
## Tuning parameter 'min.node.size' was held constant at a value of 1
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were mtry = 32, splitrule = gini
##  and min.node.size = 1.
```

The best hyperparameter fit reports an AUC of 0.945 on cross-validation, the next step is to fit this model with the entire training set and study the results by validating the test set:

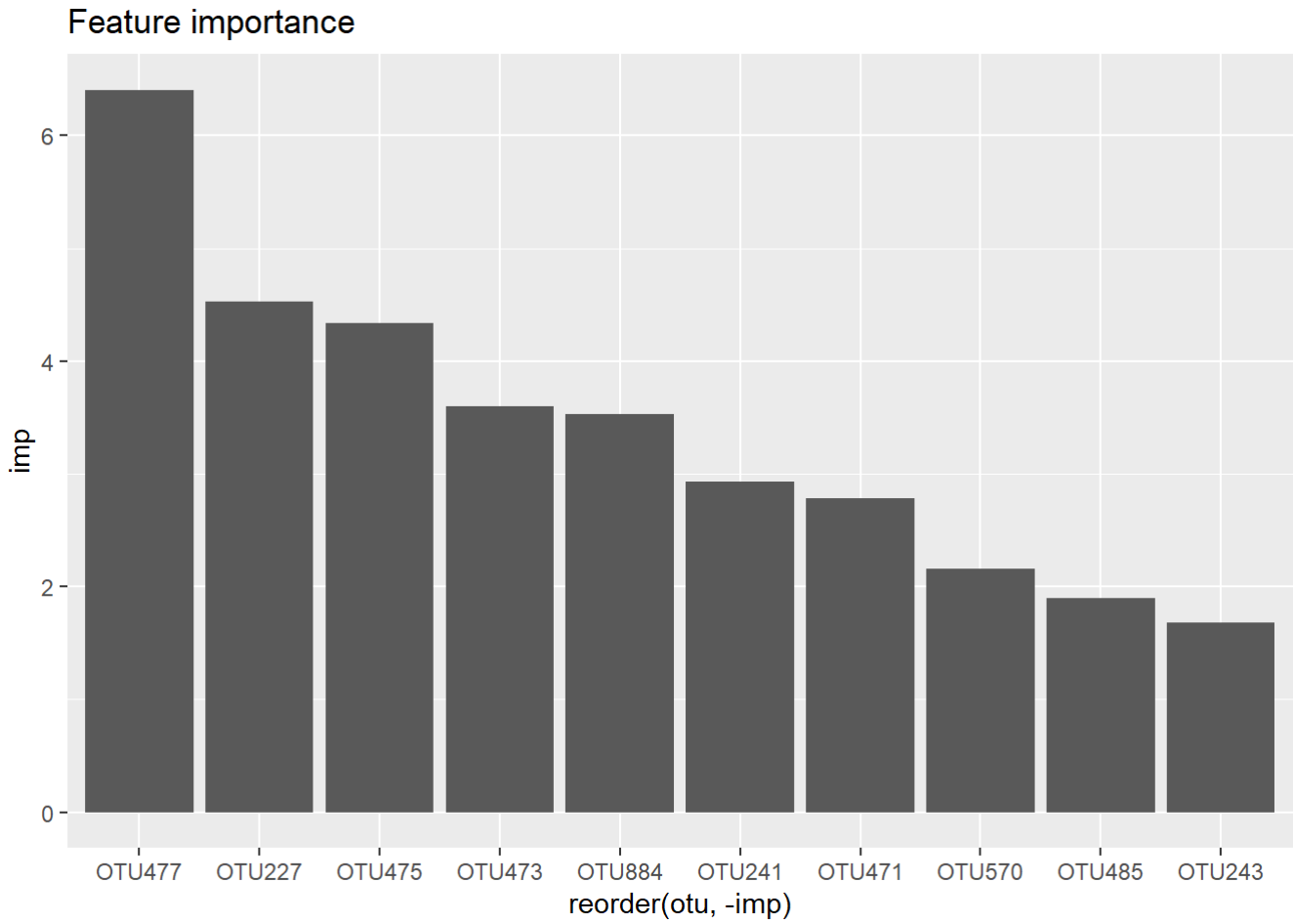
```
modelo_final = ranger(disease~.,data = clrTrain,mtry = 32,splitrule = 'gini',min.node.size =
1,importance = 'impurity',probability=TRUE,keep.inbag = TRUE)

pred = predict(modelo_final, data=clrTest, type="response")
pred = pred$predictions
ROC1_rf_mej <- roc(clrTest$disease,pred[,2])
a_rf_mej=auc(ROC1_rf_mej)
a_rf_mej
```

```
## Area under the curve: 0.9466
```

The AUC obtained is almost identical to the one obtained by Pasolli, as this result was obtained with cross-validation, we go back to the same process of filtering variables to improve the validation obtained. In this case, our most important variables are:

```
library(ggplot2)
imp=sort(importance(modelo_final),decreasing=TRUE)
d=as.data.frame(imp)
d$otu = rownames(d)
p = ggplot(d[1:10,],aes(y=imp,x=reorder(otu,-imp))) + geom_col() + ggtitle('Feature importanc
e')
p
```




```
nrow(as.data.frame(d[which(d$imp>0),]))
```

```
## [1] 385
```

We found the best model by filtering variables and performing repeated cross-validation:

```
set.seed(100)
max_auc = 0
variables=0
params=0
tr_fit = trainControl(method= 'repeatedcv',repeats=5,search='grid',number = 10, classProbs =
TRUE,summaryFunction = twoClassSummary)
for (i in seq(10,385,15)){
  data= clrTrain[,c(rownames(d)[1:i],'disease')]
  best_model = train(disease~., data=data, trControl = tr_fit,method = 'ranger',metric = 'RO
C')
  if (max(best_model$results$ROC)>max_auc){

    max_auc=max(best_model$results$ROC)
    variables=i
    params= best_model$bestTune
  }
}
print(paste('AUC= ',max_auc))
```

```
## [1] "AUC=  0.954572839506173"
```

```
print(paste('NVars= ',variables))
```

```
## [1] "NVars=  55"
```

```
params
```

```
##   mtry splitrule min.node.size
## 1    2      gini             1
```

In this case, we obtain the same AUC than before, but this time using less variables (55). Moreover, the hyperparameters are also the same:

- Mtry = 2
- Splitrule = gini
- Min node size = 1

Finally, we train this model with the whole training set and predict the test set:

```
set.seed(30)
data_filt = clrTrain[,c(rownames(d)[1:55],'disease')]
modelo_final = ranger(disease~.,data = data_filt,mtry = 2,splitrule = 'gini',min.node.size =
1,importance = 'impurity',probability=TRUE,keep.inbag = TRUE)

pred = predict(modelo_final, data=clrTest[,c(rownames(d)[1:55])], type="response")
pred = pred$predictions
ROC1_rf_mej <- roc(clrTest$disease,pred[,2])
a_rf_mej=auc(ROC1_rf_mej)

print(paste('AUC: ',a_rf_mej))
```

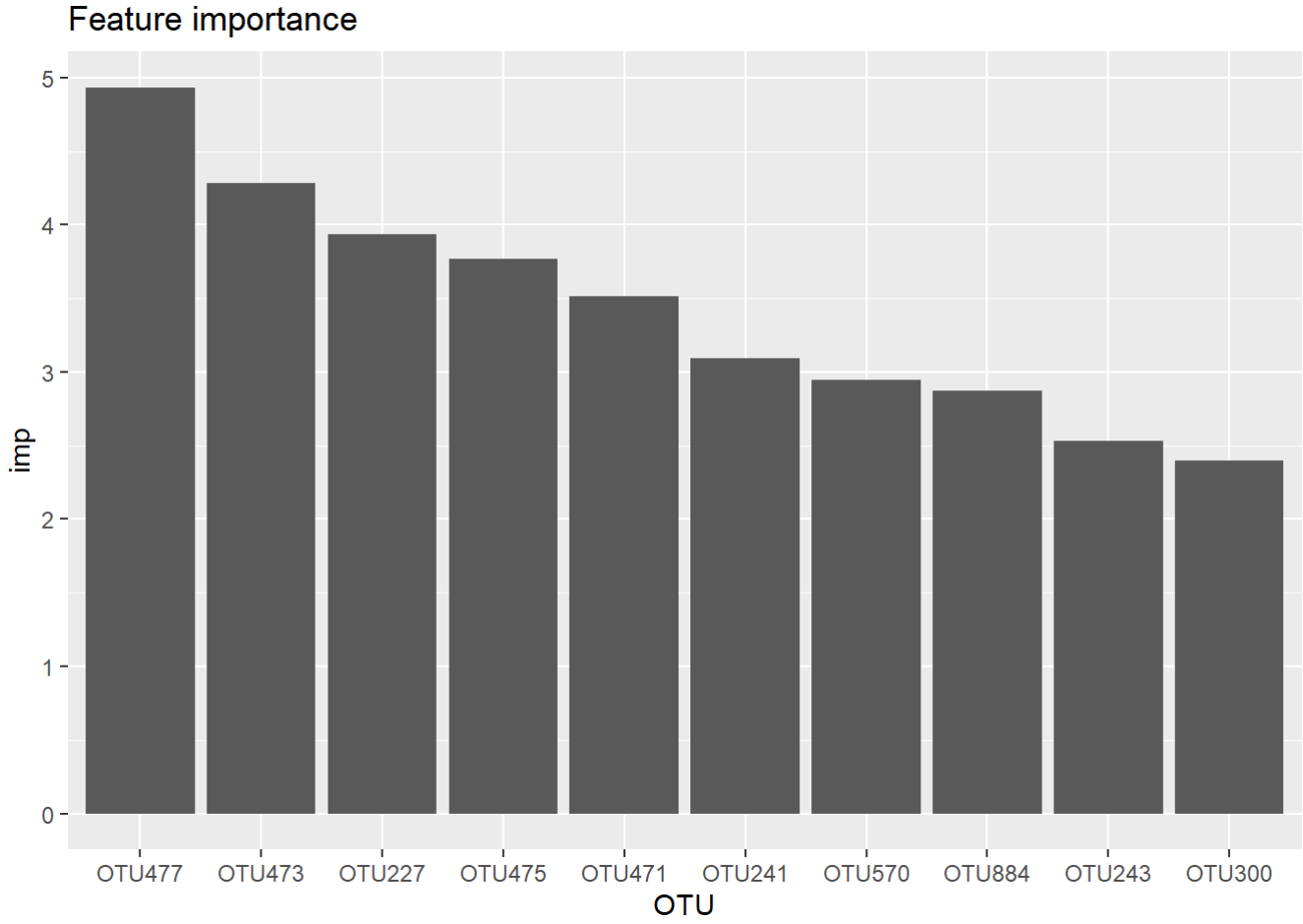
```
## [1] "AUC:  0.944664031620553"
```

```
print(paste('F1 Score: ',F1_Score(clrTest$disease,factor(ifelse(pred[,1] < 0.5, 'y', 'n'), le
vels = c('n','y')))))
```

```
## [1] "F1 Score:  0.904761904761905"
```

With this preprocessing, we obtain an AUC of 0.944 and an F1 Score of 0.904, a slightly lower result of the AUC than the obtained with the TSS preprocessing. We can now look at the most important variables in this model and see if there are similarities in both models:

```
library(ggplot2)
imp=sort(importance(modelo_final),decreasing=TRUE)
d=as.data.frame(imp)
d$otu = rownames(d)
p = ggplot(d[1:10,],aes(y=imp,x=reorder(otu,-imp))) + geom_col() + ggtitle('Feature importanc
e') + xlab('OTU')
p
```




```
nrow(as.data.frame(d[which(d$imp>0),]))
```

```
## [1] 55
```

With this preprocessing we find that the importance of the 10 most important OTUs decreases uniformly from the OTU 477 all the way to OTU 300. The first OTU being 477 remains in the first place; the other ones have been mixed up, but still remaining in similar positions, for example OTU 473, 475 and 277. It seems that depending on the numerical transformation that we use we can achieve a slightly different result when validating the models.

# S2 Preprocessed

Second, we are going to repeat the process with the other data filter we have, the S2 Normal filter. In the same way that we have done with the S1 filtering, we first divide the data into 80% to train the models and 20% to validate them. We will start by adjusting the RandomForest technique to the TSS preprocessing.

```
set.seed(110)
clr=as.data.frame(t(DatosFinal$Cirrosis$S2_NORMAL$S2_CLR_CIRR))
a=Datos_sample$Cirrosis$cirrhotic
clr$disease=factor(a)
colnames(clr)=sub(' ','',colnames(clr))
trainFilas = createDataPartition(clr$disease, p=0.8, list=FALSE)
clrTrain = clr[trainFilas,]
clrTest = clr[-trainFilas,]


tss=as.data.frame(log(as.data.frame(t(DatosFinal$Cirrosis$S2_NORMAL$S2_TSS_CIRR)) * 10e6 + 1
))
tss$disease=factor(a)
colnames(tss)=sub(' ','',colnames(tss))
tssTrain =tss[trainFilas,]
tssTest = tss[-trainFilas,]
```

## TSS

We proceed to training by repeated cross-validation to adjust the hyperparameters of the model as we have done before. The results can be seen below:

```
library(e1071)
library(ranger)
set.seed(200)

tr_fit = trainControl(method= 'repeatedcv',repeats=5,search='grid',number = 10, classProbs =
TRUE,summaryFunction = twoClassSummary)
best_model = train(disease~., data=tssTrain, trControl = tr_fit,method = 'ranger',metric = 'R
OC')
best_model
```

```
## Random Forest
##
## 187 samples
## 190 predictors
##   2 classes: 'n', 'y'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 169, 168, 167, 169, 169, 168, ...
## Resampling results across tuning parameters:
##
##   mtry  splitrule   ROC        Sens       Spec
##     2   gini        0.9381778  0.9477778  0.8024444
##     2   extratrees  0.9302444  0.9568889  0.7284444
##    96   gini        0.9329951  0.9044444  0.8177778
##    96   extratrees  0.9383506  0.9080000  0.8111111
##   190   gini        0.9333148  0.8977778  0.8240000
##   190   extratrees  0.9368988  0.9171111  0.8048889
##
## Tuning parameter 'min.node.size' was held constant at a value of 1
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were mtry = 96, splitrule = extratrees
##  and min.node.size = 1.
```

By adjusting the hyperparameters we obtain an AUC of 0.9383 with the cross-validation with repetition, now we train this model with the entire training set and validate with the test set.

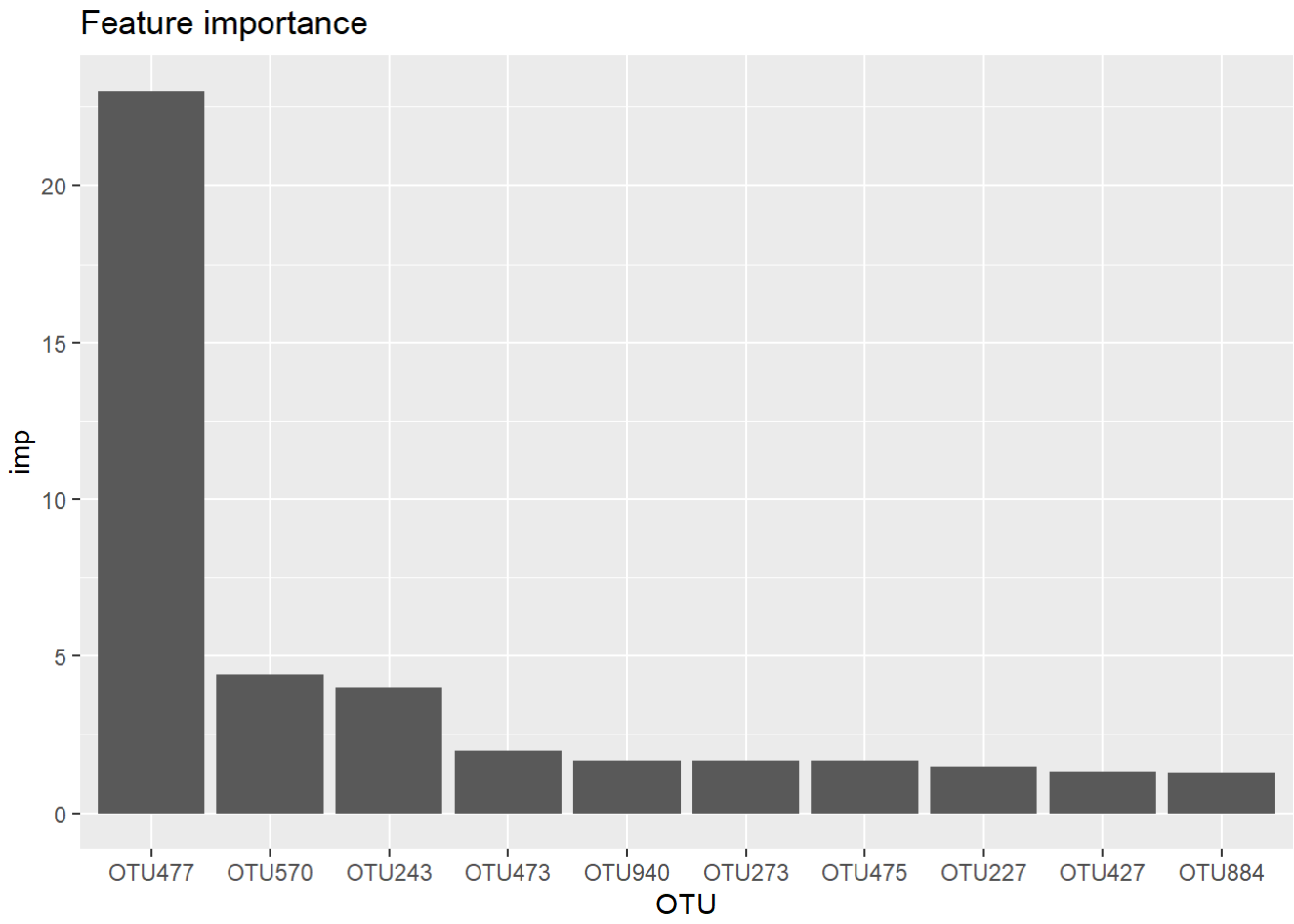
```
modelo_final = ranger(disease~.,data = tssTrain,mtry = 96,splitrule = 'extratrees',min.node.s
ize = 1,importance = 'impurity',probability=TRUE,keep.inbag = TRUE)

pred = predict(modelo_final, data=tssTest, type="response")
pred = pred$predictions
ROC1_rf_mej <- roc(tssTest$disease,pred[,2])
a_rf_mej=auc(ROC1_rf_mej)
a_rf_mej
```

```
## Area under the curve: 0.9427
```

As has happened in the previous cases, the AUC obtained with this model is 0.942, so we try to improve it by training new models selecting the most important variables. In this all of the variables have an importance greater than 0. Below we can see the most important:

```
library(ggplot2)
imp=sort(importance(modelo_final),decreasing=TRUE)
d=as.data.frame(imp)
d$otu = rownames(d)
p = ggplot(d[1:10,],aes(y=imp,x=reorder(otu,-imp))) + geom_col() + ggtitle('Feature importanc
e') + xlab('OTU')
p
```




```
nrow(as.data.frame(d[which(d$imp>0),]))
```

```
## [1] 190
```

Let's see if we can improve the results:

```
max_auc = 0
variables=0
params=0
tr_fit = trainControl(method= 'repeatedcv',repeats=5,search='grid',number = 10, classProbs =
TRUE,summaryFunction = twoClassSummary)
for (i in seq(10,nrow(as.data.frame(d[which(d$imp>0),])),15)){
  data= tssTrain[,c(rownames(d)[1:i],'disease')]
  best_model = train(disease~., data=data, trControl = tr_fit,method = 'ranger',metric = 'RO
C')
  if (max(best_model$results$ROC)>max_auc){

    max_auc=max(best_model$results$ROC)
    variables=i
    params= best_model$bestTune
  }
}
print(paste('AUC= ',max_auc))
```

```
## [1] "AUC=  0.958466666666667"
```

```
print(paste('NVars= ',variables))
```

```
## [1] "NVars=  40"
```

```
params
```

```
##   mtry splitrule min.node.size
## 1    2      gini             1
```

The hyperparameters chosen with these variables are the following:

- mtry = 2
- Splitrule = gini
- Min node size = 1

With an achieved AUC of 0.958 using 40, the results seem to be better than filtering the variables. Next we will see what AUC score we get with the test set:

```
set.seed(30)
data_filt = tssTrain[,c(rownames(d)[1:40],'disease')]
modelo_final = ranger(disease~.,data = data_filt,mtry = 2,splitrule = 'gini',min.node.size =
1,importance = 'impurity',probability=TRUE,keep.inbag = TRUE)

pred = predict(modelo_final, data=tssTest[,c(rownames(d)[1:40])], type="response")
pred = pred$predictions
ROC1_rf_mej <- roc(tssTest$disease,pred[,2])
a_rf_mej=auc(ROC1_rf_mej)
print(paste('AUC: ',a_rf_mej))
```

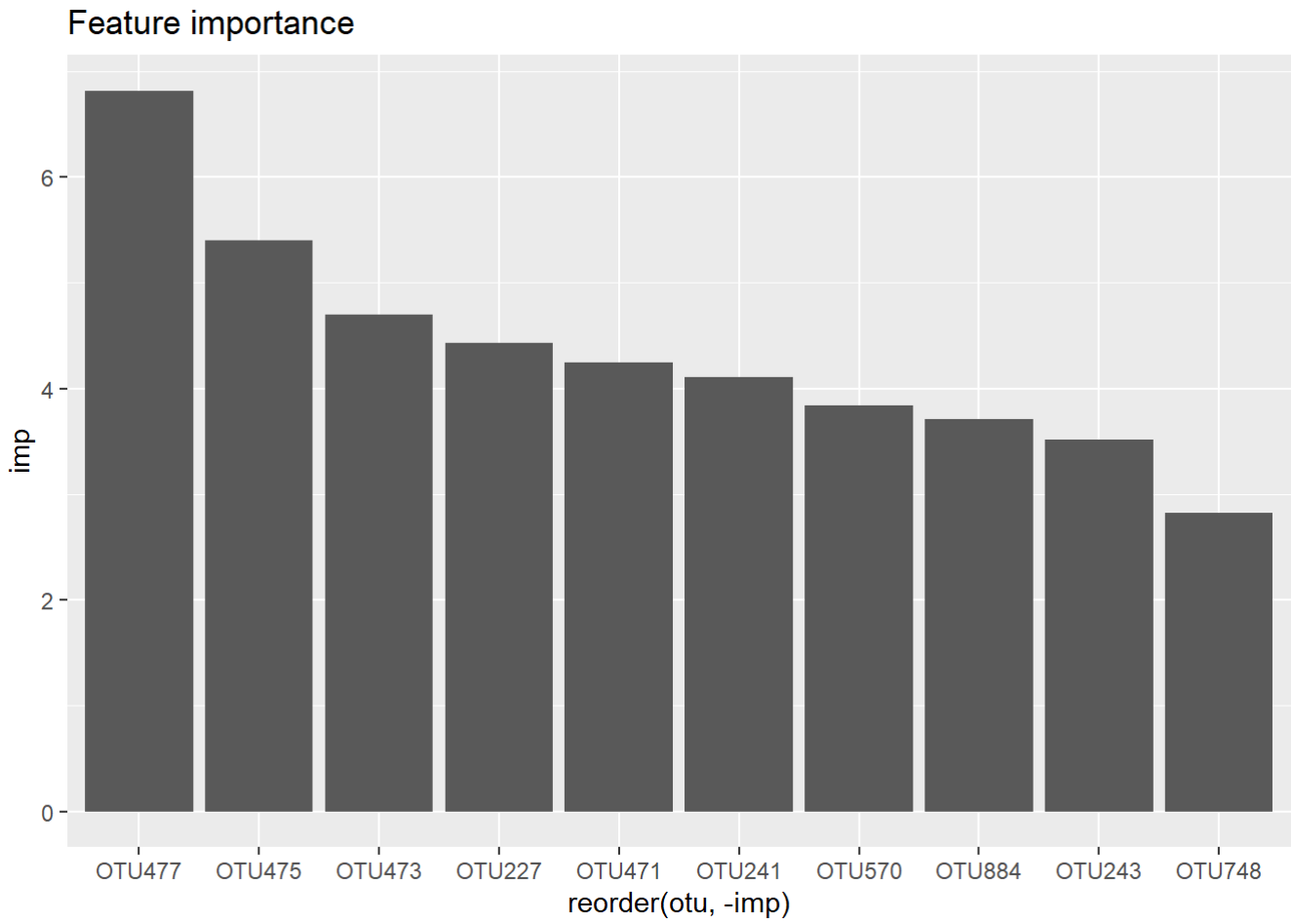
```
## [1] "AUC:  0.946640316205534"
```

```
print(paste('F1 Score: ',F1_Score(clrTest$disease,factor(ifelse(pred[,1] < 0.5, 'y', 'n'), le
vels = c('n','y')))))
```

```
## [1] "F1 Score:  0.909090909090909"
```

When validating the model on the test set, we obtain an AUC of 0.946 and an F1 Score of 0.909, values quite similar to those obtained with the same preprocessing but with S1 filtering. This may be due to the fact that if the variables in the S1 TSS model continue to be in the dataframe of the S2 filter, the results should be almost identical, since the most important variables should be the same and therefore the classification of the individuals should be the same. To test this hypothesis, we show the most important variables in this model:

```
imp=sort(importance(modelo_final),decreasing=TRUE)
d=as.data.frame(imp)
d$otu = rownames(d)
p = ggplot(d[1:10,],aes(y=imp,x=reorder(otu,-imp))) + geom_col() + ggtitle('Feature importanc
e')
p
```




As expected, practically the same variables appear in the graph as in the same graph of the model corresponding to the preprocessed S1-TSS, so the differences in the statistics may be due to randomForest randomization, since the data are the same in both models, since the variables that help to differentiate between individuals are present in both data sets.

# CLR

Finally, we are going to adjust the random forest on the last preprocessing that we have left, the CLR with the S2 filtering. From the results observed throughout the report, the results of this model should be quite similar to those obtained in the model with the preprocessed S1 CLR, since the variables that help to separate the classes well are the same in both preprocesses. . We will see if at the end of this analysis we can corroborate this hypothesis.

To do this, we first adjust the random forest using all the variables through cross-validation with repetition, as we have been doing in the 3 previous cases

```
set.seed(100)
best_model = train(disease~., data=clrTrain, trControl = tr_fit,method = 'ranger',metric = 'R
OC')
best_model
```

```
## Random Forest
##
## 187 samples
## 190 predictors
##   2 classes: 'n', 'y'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 169, 168, 168, 169, 169, 169, ...
## Resampling results across tuning parameters:
##
##   mtry  splitrule   ROC        Sens       Spec
##     2   gini        0.9419309  0.9451111  0.7975556
##     2   extratrees  0.9258469  0.9468889  0.7166667
##    96   gini        0.9291704  0.8808889  0.8104444
##    96   extratrees  0.9348531  0.9013333  0.8084444
##   190   gini        0.9229580  0.8677778  0.8253333
##   190   extratrees  0.9340765  0.8900000  0.8062222
##
## Tuning parameter 'min.node.size' was held constant at a value of 1
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were mtry = 2, splitrule = gini
##  and min.node.size = 1.
```

Next, we train the model that has given us the best AUC (0.941), with the entire training set, and predict the test set.

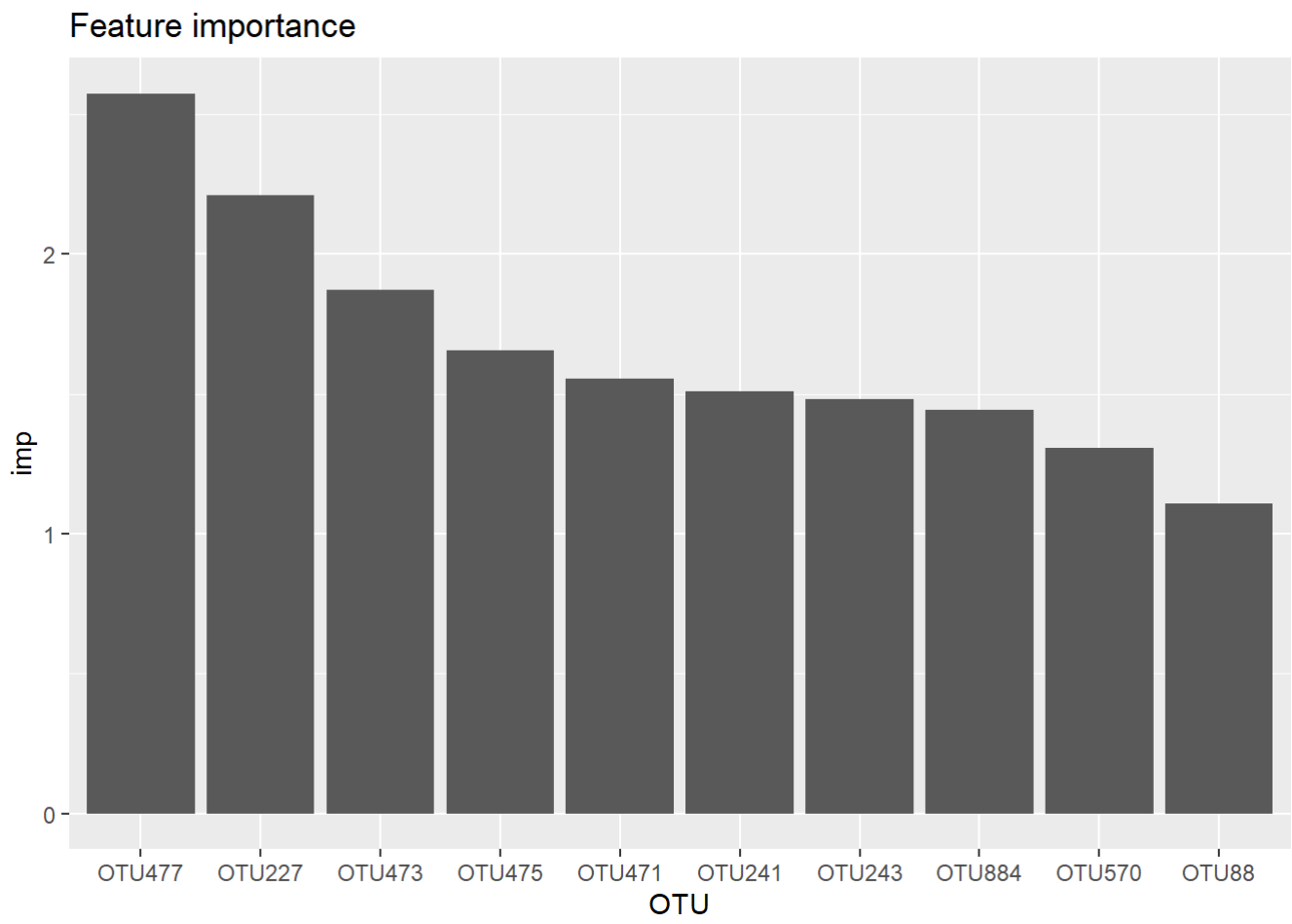
```
modelo_final = ranger(disease~.,data = clrTrain,mtry = 2,splitrule = 'gini',min.node.size = 1
,importance = 'impurity',probability=TRUE,keep.inbag = TRUE)

pred = predict(modelo_final, data=clrTest, type="response")
pred = pred$predictions
ROC1_rf_mej <- roc(clrTest$disease,pred[,2])
a_rf_mej=auc(ROC1_rf_mej)
a_rf_mej
```

```
## Area under the curve: 0.9466
```

The AUC achieved with the test set is similar to the previous cases: 0.9466. To improve it, we train models by selecting a different number of variables, in this case we have 162 variables with importance greater than 0. The most important ones can be seen below:

```
library(ggplot2)
imp=sort(importance(modelo_final),decreasing=TRUE)
d=as.data.frame(imp)
d$otu = rownames(d)
p = ggplot(d[1:10,],aes(y=imp,x=reorder(otu,-imp))) + geom_col() + ggtitle('Feature importanc
e') + xlab('OTU')
p
```




```
nrow(as.data.frame(d[which(d$imp>0),]))
```

```
## [1] 190
```

After training the different models by filtering variables, the best model obtained is the following:

```
set.seed(100)
max_auc = 0
variables=0
params=0
tr_fit = trainControl(method= 'repeatedcv',repeats=5,search='grid',number = 10, classProbs =
TRUE,summaryFunction = twoClassSummary)
for (i in seq(10,nrow(as.data.frame(d[which(d$imp>0),])),15)){
  data= clrTrain[,c(rownames(d)[1:i],'disease')]
  best_model = train(disease~., data=data, trControl = tr_fit,method = 'ranger',metric = 'RO
C')
  if (max(best_model$results$ROC)>max_auc){

    max_auc=max(best_model$results$ROC)
    variables=i
    params= best_model$bestTune
  }
}
print(paste('AUC= ',max_auc))
```

```
## [1] "AUC=  0.952076543209877"
```

```
print(paste('NVars= ',variables))
```

```
## [1] "NVars=  55"
```

```
params
```

```
##   mtry splitrule min.node.size
## 1    2      gini             1
```

In this case, we obtain an AUC of 0.952, better than with the previous preprocessing using 55 variables. Also, the hyperparameters are the same again:

- Mtry = 2
- Splitrule = gini
- Min node size = 1

Finally, we train this model with the entire training set and predict the test set:

```
set.seed(30)
data_filt = clrTrain[,c(rownames(d)[1:55],'disease')]
modelo_final = ranger(disease~.,data = data_filt,mtry = 2,splitrule = 'extratrees',min.node.s
ize = 1,importance = 'impurity',probability=TRUE,keep.inbag = TRUE)

pred = predict(modelo_final, data=clrTest[,c(rownames(d)[1:55])], type="response")
pred = pred$predictions
ROC1_rf_mej <- roc(tssTest$disease,pred[,2])
a_rf_mej=auc(ROC1_rf_mej)
print(paste('AUC: ',a_rf_mej))
```

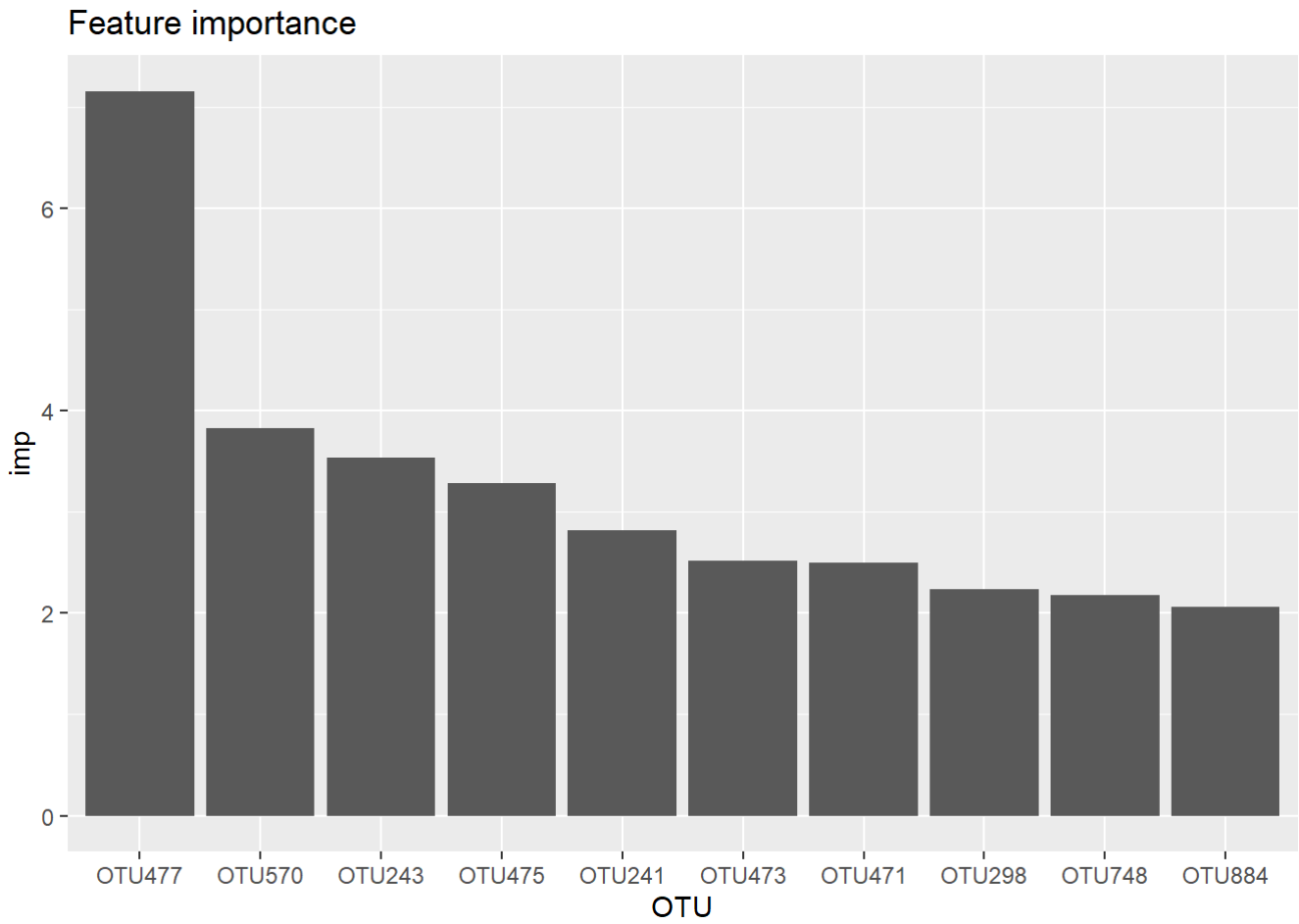
```
## [1] "AUC:  0.936758893280632"
```

```
print(paste('F1 Score: ',F1_Score(clrTest$disease,factor(ifelse(pred[,1] < 0.5, 'y', 'n'), le
vels = c('n','y')))))
```

```
## [1] "F1 Score:  0.909090909090909"
```

As expected, the results are again very similar to those obtained with the CLR preprocessing in the S1 filter. To finish confirming our hypothesis, we are going to see if the most important variables of this model also coincide with those of the previous 3:

```
library(ggplot2)
imp=sort(importance(modelo_final),decreasing=TRUE)
d=as.data.frame(imp)
d$otu = rownames(d)
p = ggplot(d[1:10,],aes(y=imp,x=reorder(otu,-imp))) + geom_col() + ggtitle('Feature importanc
e') + xlab('OTU')
p
```




```
nrow(as.data.frame(d[which(d$imp>0),]))
```

```
## [1] 55
```

With this graph we confirm our assumptions, since we return to see species 477 in first place and 475, 473 or 243 in the top 10 once again. With this particular disease we observe that all combinations of filtering and preprocessing end up in similar results. As expected, the variables that are the most influential remain the same regardless of the treatment of data.

# Conclusión

In conclusion, we can say that the differences between filtering and preprocessing are almost non-existent using random forest.

In addition, in the 4 trained models, the variables that are most important in the models are practically the same, and when relating them to the taxonomy of the species, we have not found a great degree of relationship between them, so it does not seem that said species have a lot of relationship with each other.