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 diabetes 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('DatosSinOutlier.RData')

set.seed(100)
clr=as.data.frame(t(DatosFinal$WT2D$S1_NORMAL$S1_CLR_WT2D))
a=Datos_sample$WT2D$disease
clr$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$WT2D$S1_NORMAL$S1_TSS_WT2D)) * 10e6
 + 1))
tss$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, class
Probs = TRUE,summaryFunction = twoClassSummary)
best_model = train(disease~., data=tssTrain, trControl = tr_fit,method = 'ranger',met
ric = 'ROC')
best_model
```

```
## Random Forest
##
##  77 samples
## 365 predictors
##   2 classes: 'n', 't2d'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 70, 68, 69, 70, 70, 69, ...
## Resampling results across tuning parameters:
##
##   mtry  splitrule   ROC        Sens       Spec
##     2   gini        0.7726667  0.5633333  0.798
##     2   extratrees  0.7279167  0.4516667  0.795
##   183   gini        0.7279167  0.5766667  0.701
##   183   extratrees  0.7315000  0.6316667  0.718
##   365   gini        0.7068333  0.5816667  0.694
##   365   extratrees  0.7178333  0.6433333  0.723
##
## 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 gave us the best AUC (0.7726) when cross-validated, with the entire training set, and predict the test set.

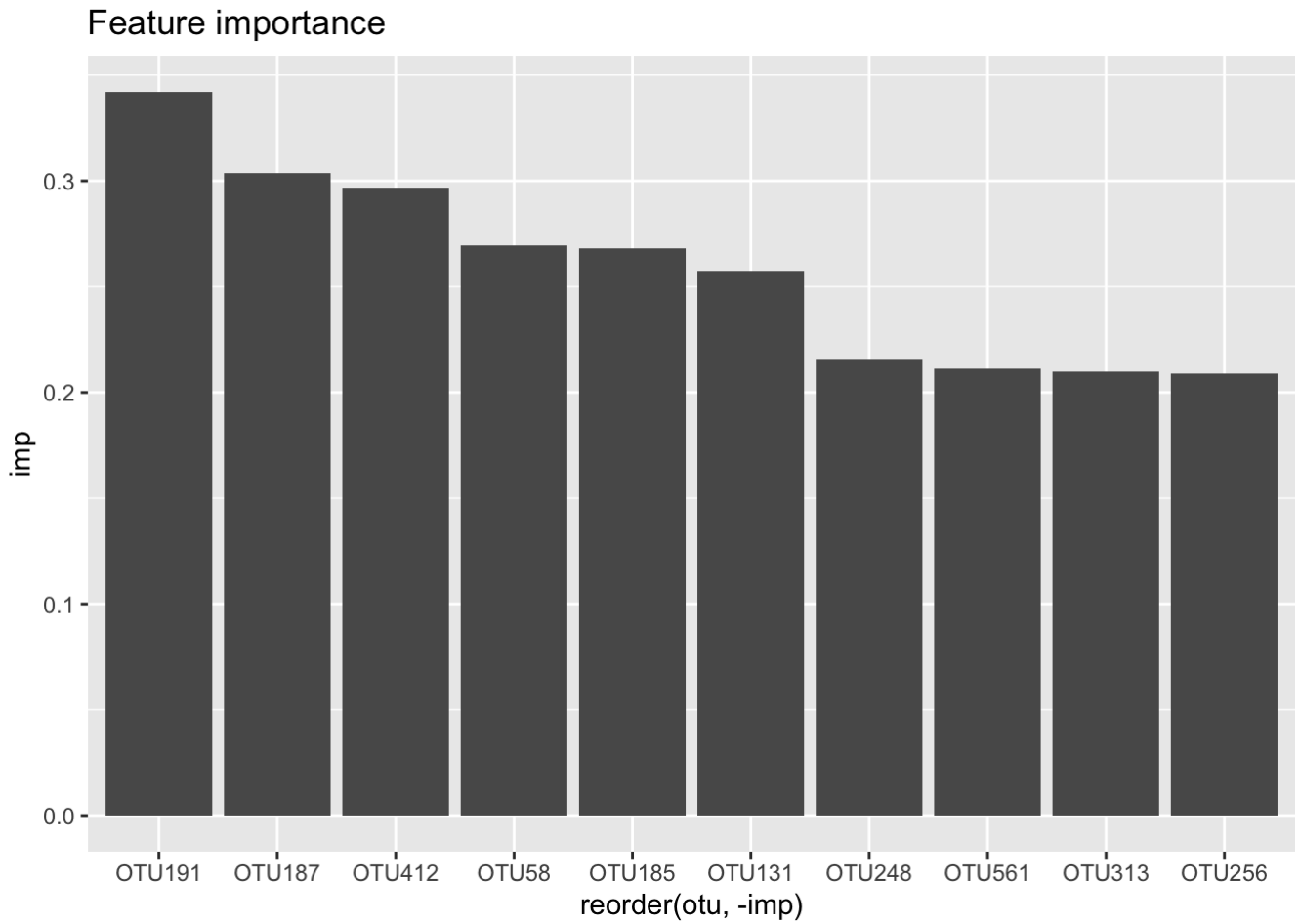
```
modelo_final = ranger(disease~.,data = tssTrain,mtry = 2,splitrule = 'gini',min.node.
size = 1,importance = 'impurity',probability=TRUE,keep.inbag = TRUE)

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: 0.6375
```

With the test set we managed to achieve an AUC of 0.6625, 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 i
mportance')
p
```




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

```
## [1] 354
```

There are 355 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 355. 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, class
Probs = TRUE,summaryFunction = twoClassSummary)
for (i in seq(10,355,15)){
  data= tssTrain[,c(rownames(d)[1:i],'disease')]
  best_model = train(disease~., data=data, trControl = tr_fit,method = 'ranger',metri
c = 'ROC')
  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=  0.823166666666667"
```

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

```
## [1] "NVars=  85"
```

```
params
```

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

The best result has been achieved using the 85 most important variables in the previous model and with the parameters:

- mtry = 43
- Splitrule = extratrees
- Min node size = 1

With an AUC score of 0.852 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)[1:85],'disease')]
modelo_final = ranger(disease~.,data = data_filt,mtry = 43,splitrule = 'extratrees',m
in.node.size = 1,importance = 'impurity',probability=TRUE,keep.inbag = TRUE)

pred = predict(modelo_final, data=tssTest[,c(rownames(d)[1:85])], 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))
```

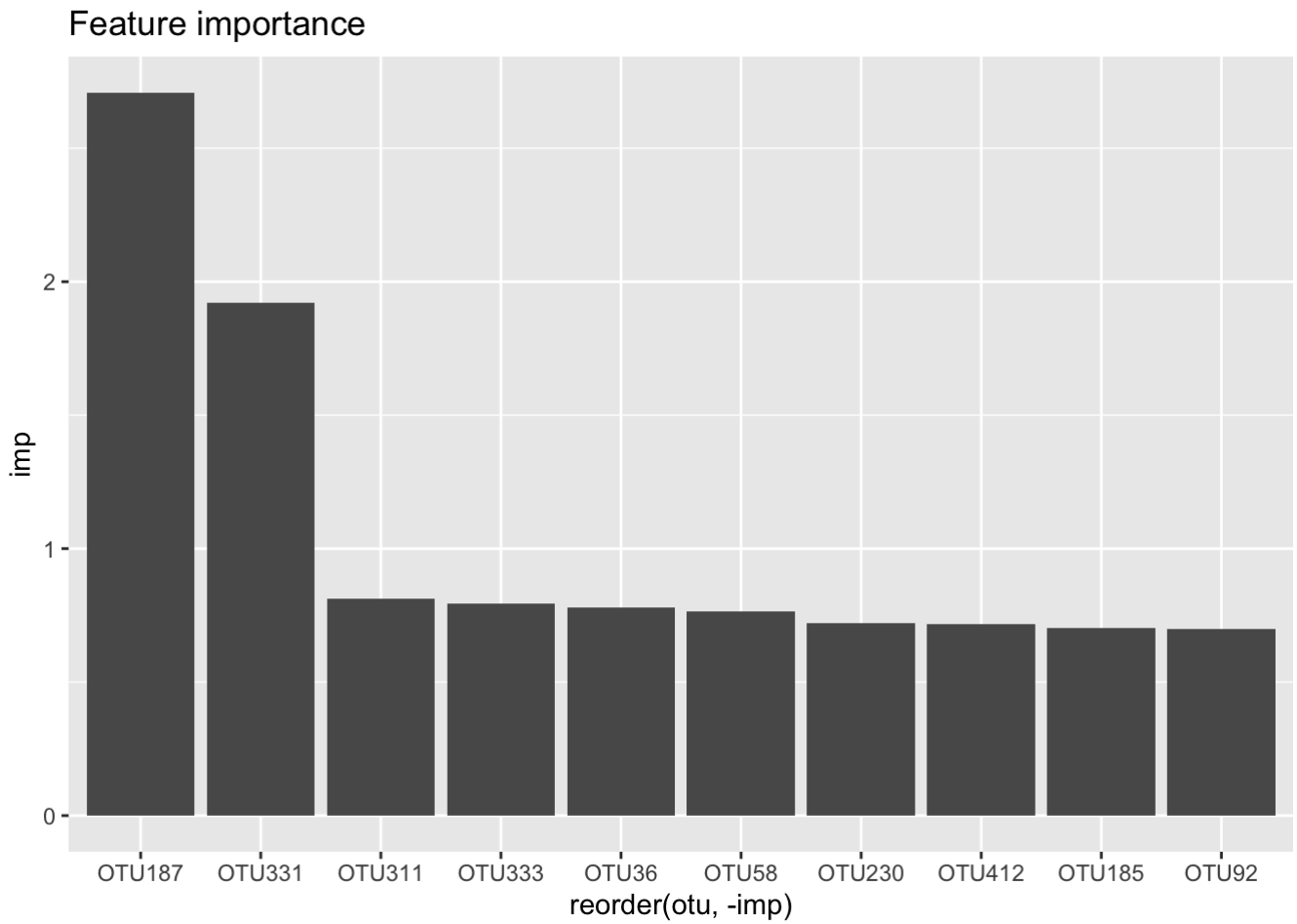
```
## [1] "AUC:  0.6"
```

```
print(paste('F1 Score: ',F1_Score(tssTest$disease,factor(ifelse(pred[,1] < 0.5, 't2d'
, 'n')),positive = 't2d')))
```

```
## [1] "F1 Score:  0.526315789473684"
```

Finally, training the model with the entire training set and making predictions on the test set, we obtain an area under the curve of 0.6937 and an F1 Score of 0.8181, values considerably higher than those obtained with the model without filtering variables, and much closer to 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 i
mportance')
p
```



As we can see in the graph, there are 2 variables that have a much higher importance than the rest, when integrating the name of the OTUS with the data frame 'Data_taxa', we find that these microorganisms are Alistipes_putredinis and Butyrivibrio_unclassified, both microorganisms are bacteria, but they have no taxonomic classification beyond the kingdom (Bacteria). Therefore, it does not seem that there is any family or genus of bacteria that has a great influence on the appearance of diabetes in humans, although more complete analyses would be necessary and with people specialised in this field to reach more robust conclusions, in this study we only limit ourselves to observing the results provided by the data.

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',met
ric = 'ROC')
best_model
```

```
## Random Forest
##
##  77 samples
## 365 predictors
##   2 classes: 'n', 't2d'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 69, 70, 68, 70, 70, 70, ...
## Resampling results across tuning parameters:
##
##   mtry  splitrule   ROC        Sens       Spec
##     2   gini        0.7806667  0.5016667  0.807
##     2   extratrees  0.7389167  0.4866667  0.804
##   183   gini        0.7215000  0.6000000  0.742
##   183   extratrees  0.7317500  0.6216667  0.727
##   365   gini        0.6970833  0.5883333  0.751
##   365   extratrees  0.7315000  0.6266667  0.745
##
## 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.
```

The best hyperparameter fit reports an AUC of 0.7806 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:

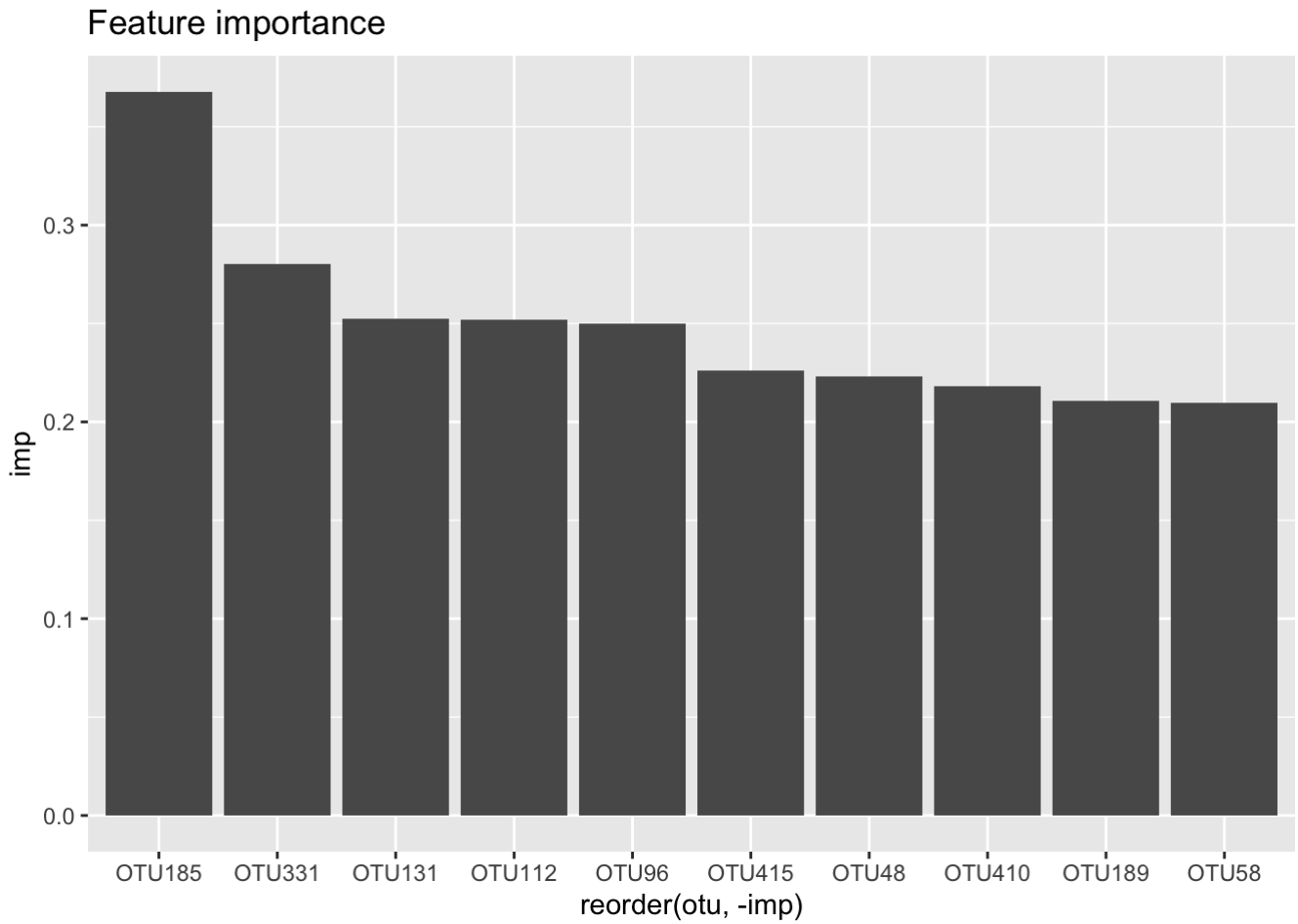
```
modelo_final = ranger(disease~.,data = clrTrain,mtry = 2,splitrule = 'gini',min.node.
size = 1,importance = 'impurity',probability=TRUE,keep.inbag = TRUE)

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: 0.675
```

Again, the AUC obtained is much lower than that obtained by Pasolli and that obtained with the cross-validation, so 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 i
mportance')
p
```




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

```
## [1] 355
```

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, class
Probs = TRUE,summaryFunction = twoClassSummary)
for (i in seq(10,355,15)){
  data= clrTrain[,c(rownames(d)[1:i],'disease')]
  best_model = train(disease~., data=data, trControl = tr_fit,method = 'ranger',metri
c = 'ROC')
  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=  0.845083333333333"
```

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

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

```
params
```

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

In this case, we obtain a 20% better AUC than with the previous preprocessing and using half the variables (40). Moreover, the hyperparameters are completely different:

- 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)[1:40],'disease')]
modelo_final = ranger(disease~.,data = data_filt,mtry = 2,splitrule = 'gini',min.nod
e.size = 1,importance = 'impurity',probability=TRUE,keep.inbag = TRUE)

pred = predict(modelo_final, data=clrTest[,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))
```

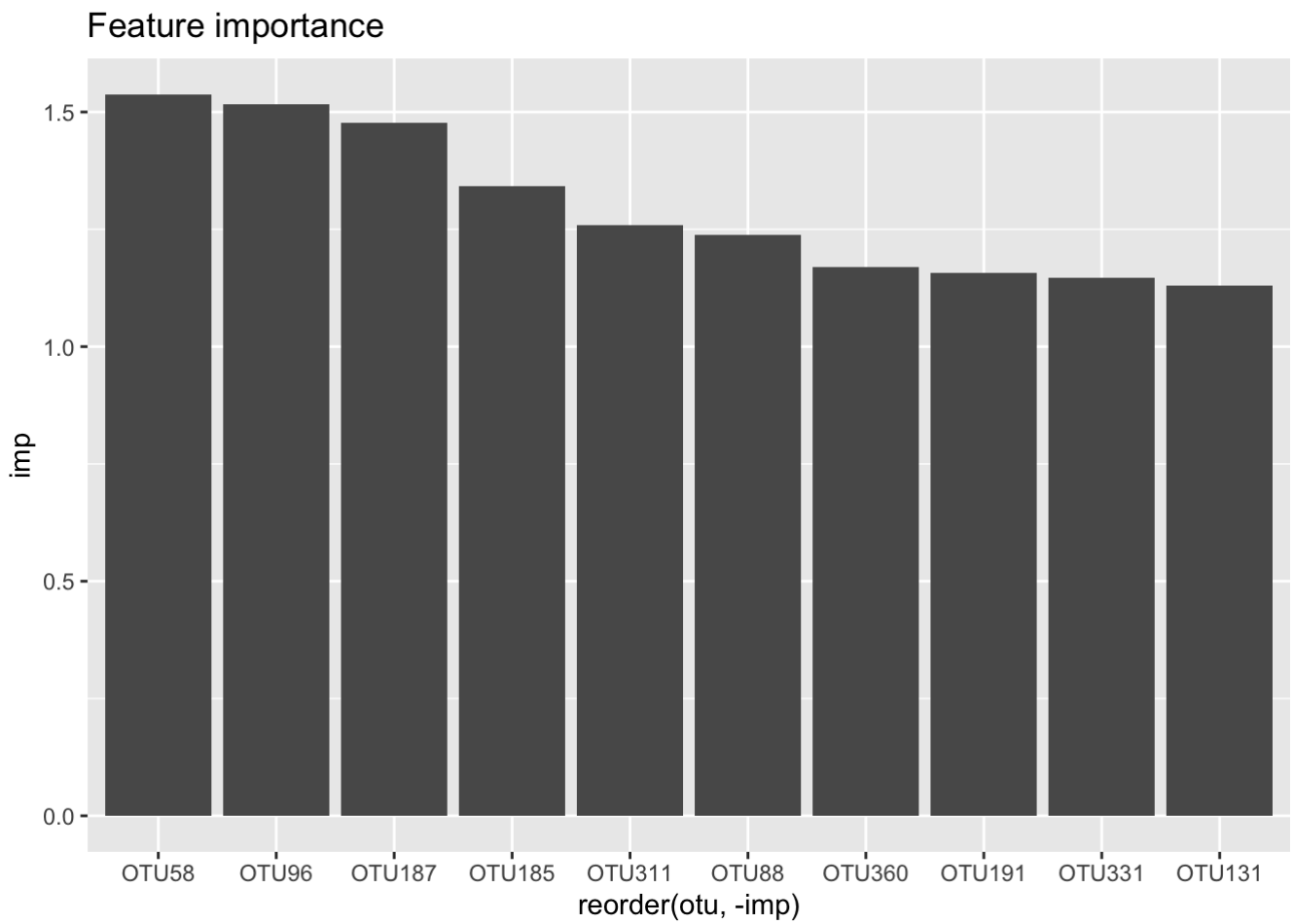
```
## [1] "AUC:  0.7"
```

```
print(paste('F1 Score: ',F1_Score(tssTest$disease,factor(ifelse(pred[,1] < 0.5, 't2d'
, 'n')),positive = 't2d')))
```

```
## [1] "F1 Score:  0.818181818181818"
```

With this preprocessing, we obtain an AUC of 0.712 and an F1 Score of 0.869, a slightly higher result than those obtained with the TSS preprocessing, which may indicate that with this transformation we can better separate the healthy individuals from the sick ones. 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 i
mportance')
p
```




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

```
## [1] 40
```

With this preprocessing we find that the first 10 most important variables have a much more similar value between them than in the TSS preprocessing, even so, although the order differs, we find quite a few coincidences in the top 10 most important variables in both models, such as the species 187, 311 or 331. This may lead us to think that the differences between healthy and diseased individuals are found in these species, but 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(100)
clr=as.data.frame(t(DatosFinal$WT2D$S2_NORMAL$S2_CLR_WT2D))
a=Datos_sample$WT2D$disease
clr$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$WT2D$S2_NORMAL$S2_TSS_WT2D)) * 10e6
 + 1))
tss$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, class
Probs = TRUE,summaryFunction = twoClassSummary)
best_model = train(disease~., data=tssTrain, trControl = tr_fit,method = 'ranger',met
ric = 'ROC')
best_model
```

```
## Random Forest
##
##  77 samples
## 162 predictors
##   2 classes: 'n', 't2d'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 70, 68, 69, 70, 70, 69, ...
## Resampling results across tuning parameters:
##
##   mtry  splitrule   ROC        Sens       Spec
##     2   gini        0.7680833  0.5783333  0.774
##     2   extratrees  0.7578333  0.5300000  0.767
##    82   gini        0.7400833  0.6016667  0.726
##    82   extratrees  0.7409583  0.6350000  0.718
##   162   gini        0.7320833  0.5983333  0.703
##   162   extratrees  0.7465000  0.6316667  0.718
##
## 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.
```

By adjusting the hyperparameters we obtain an AUC of 0.768 with the cross-validation with repetition, now we train this model with the entire training set and validate with the test set.

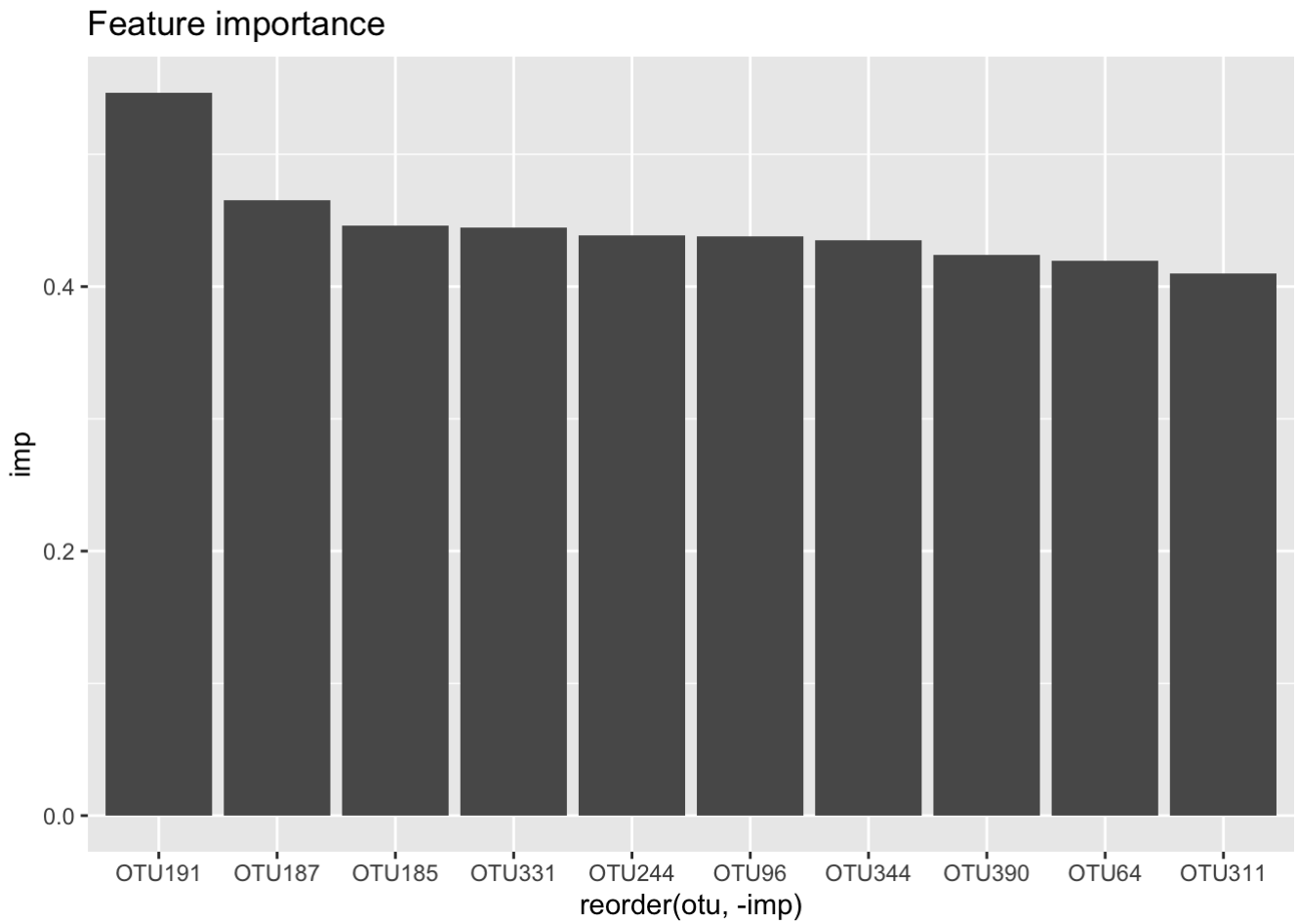
```
modelo_final = ranger(disease~.,data = tssTrain,mtry = 2,splitrule = 'gini',min.node.
size = 1,importance = 'impurity',probability=TRUE,keep.inbag = TRUE)

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: 0.675
```

As has happened in the previous cases, the AUC obtained with this model is less than 0.7, so we try to improve it by training new models selecting the most important variables. In this we only have 162 variables with 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 i
mportance')
p
```




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

```
## [1] 162
```

After training different models, we have obtained the best results by selecting the 40 most important features:

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

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

```
## [1] "AUC=  0.841125"
```

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

```
## [1] "NVars=  25"
```

```
params
```

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

The hyperparameters chosen with these variables are the following:

- mtry = 43
- Splitrule = extratrees
- Min node size = 1

With an achieved AUC of 0.858, the results seem to be better than without 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.nod
e.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))
```

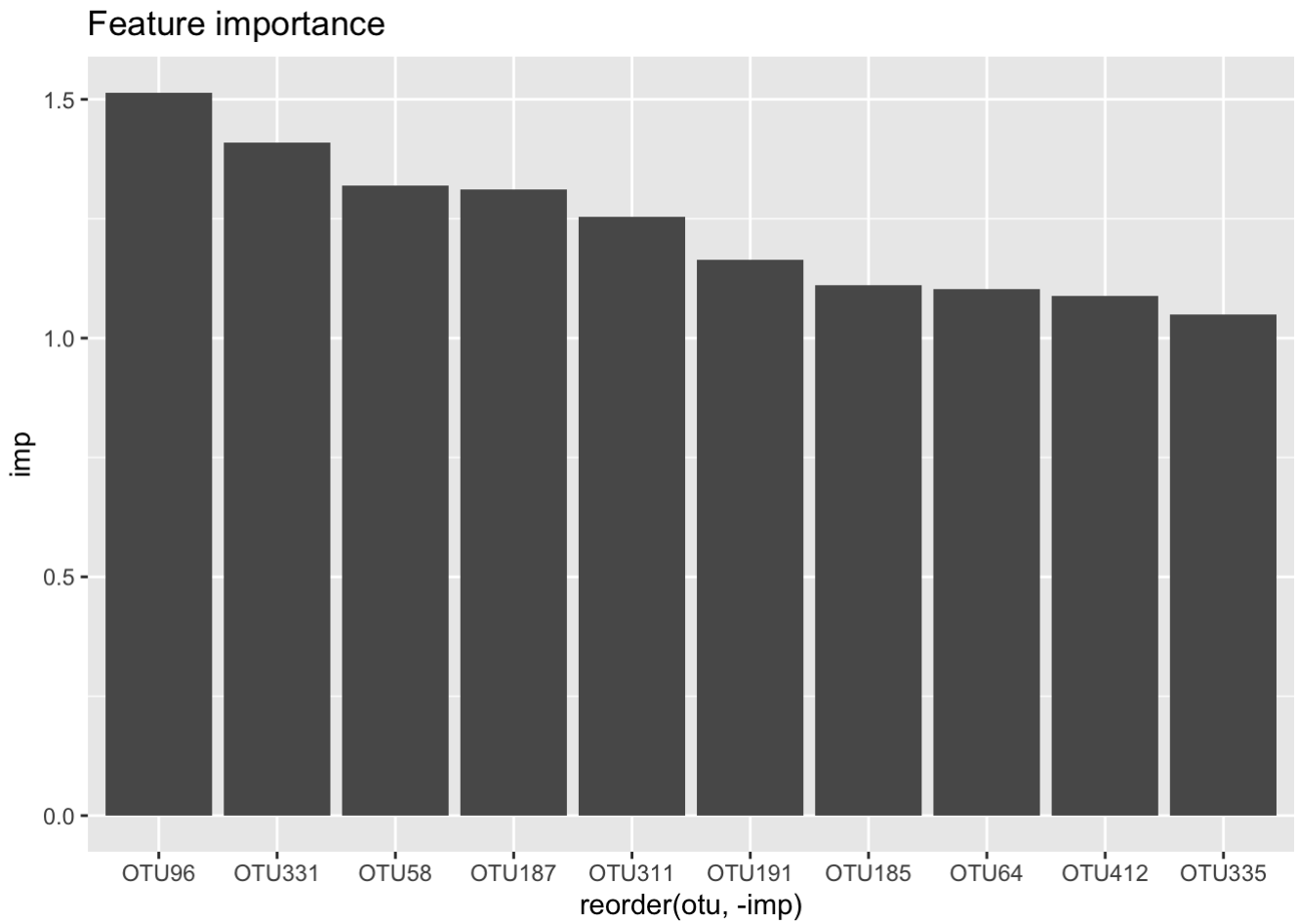
```
## [1] "AUC:  0.675"
```

```
print(paste('F1 Score: ',F1_Score(tssTest$disease,factor(ifelse(pred[,1] < 0.5, 't2d'
, 'n')),positive = 't2d')))
```

```
## [1] "F1 Score:  0.8"
```

When validating the model on the test set, we obtain an AUC of 0.6 and an F1 Score of 0.78, 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 i
mportance')
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',met
ric = 'ROC')
best_model
```

```
## Random Forest
##
##  77 samples
## 162 predictors
##   2 classes: 'n', 't2d'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 69, 70, 68, 70, 70, 70, ...
## Resampling results across tuning parameters:
##
##   mtry  splitrule   ROC        Sens       Spec
##     2   gini        0.7783333  0.6000000  0.782
##     2   extratrees  0.7294167  0.4633333  0.781
##    82   gini        0.7145417  0.5900000  0.725
##    82   extratrees  0.7349583  0.5916667  0.772
##   162   gini        0.7135833  0.5866667  0.753
##   162   extratrees  0.7411667  0.6183333  0.754
##
## 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 (0.7783), with the entire training set, and predict the test set.

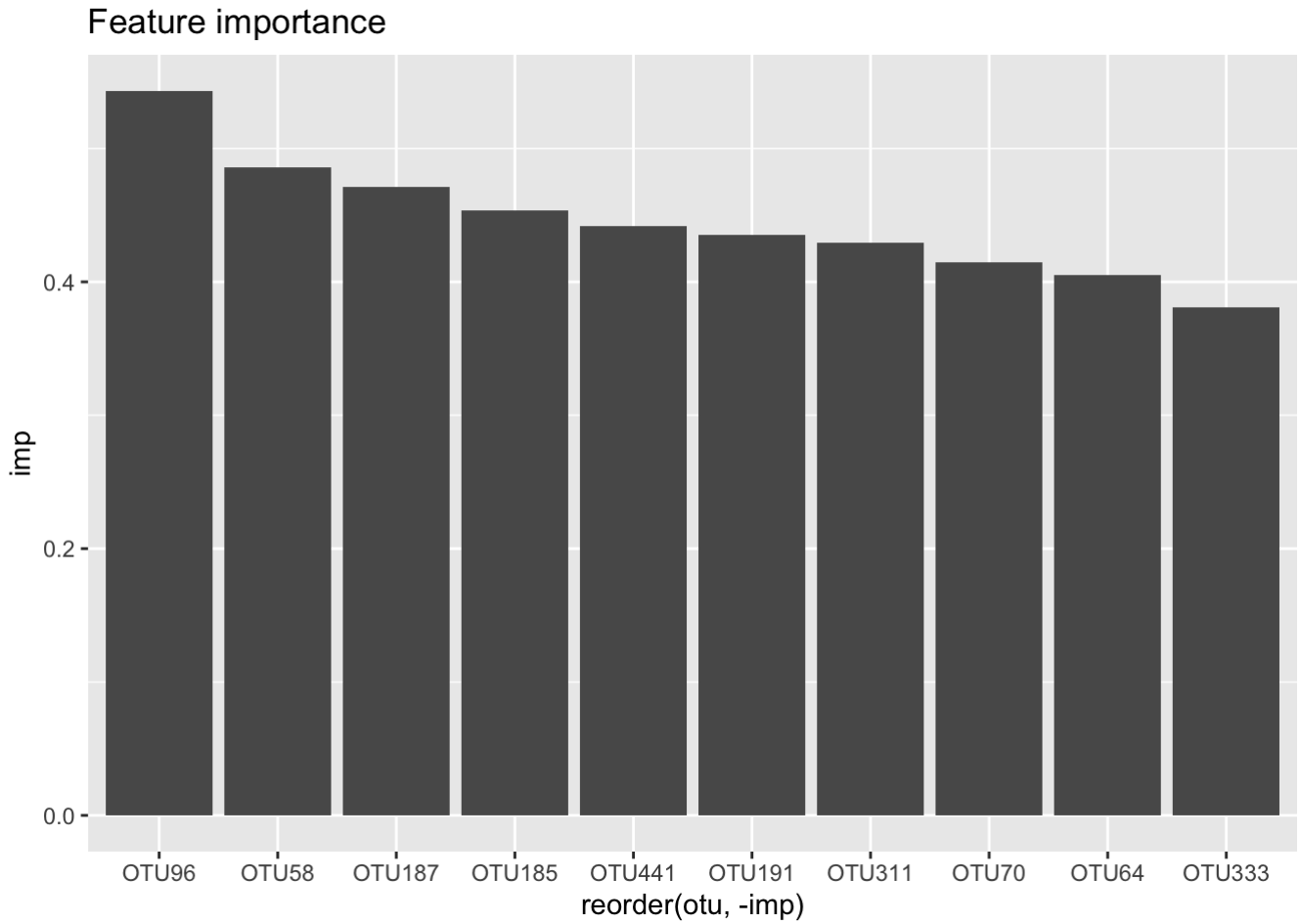
```
modelo_final = ranger(disease~.,data = clrTrain,mtry = 2,splitrule = 'gini',min.node.
size = 1,importance = 'impurity',probability=TRUE,keep.inbag = TRUE)

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: 0.6875
```

The AUC achieved with the test set is similar to the previous cases, less than 0.7. 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 i
mportance')
p
```




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

```
## [1] 162
```

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, class
Probs = TRUE,summaryFunction = twoClassSummary)
for (i in seq(10,nrow(as.data.frame(d[which(d$imp>0),])),15)){
  data= clrTrain[,c(rownames(d)[1:i],'disease')]
  best_model = train(disease~., data=data, trControl = tr_fit,method = 'ranger',metri
c = 'ROC')
  if (max(best_model$results$ROC)>max_auc){

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

```
## [1] "AUC=  0.854833333333333"
```

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

```
## [1] "NVars=  25"
```

```
params
```

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

In this case, we obtain an AUC of 85% better than with the previous preprocessing and using half of the variables (40). Also, the hyperparameters are completely different:

- 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)[1:25],'disease')]
modelo_final = ranger(disease~.,data = data_filt,mtry = 2,splitrule = 'extratrees',mi
n.node.size = 1,importance = 'impurity',probability=TRUE,keep.inbag = TRUE)

pred = predict(modelo_final, data=clrTest[,c(rownames(d)[1:25])], 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))
```

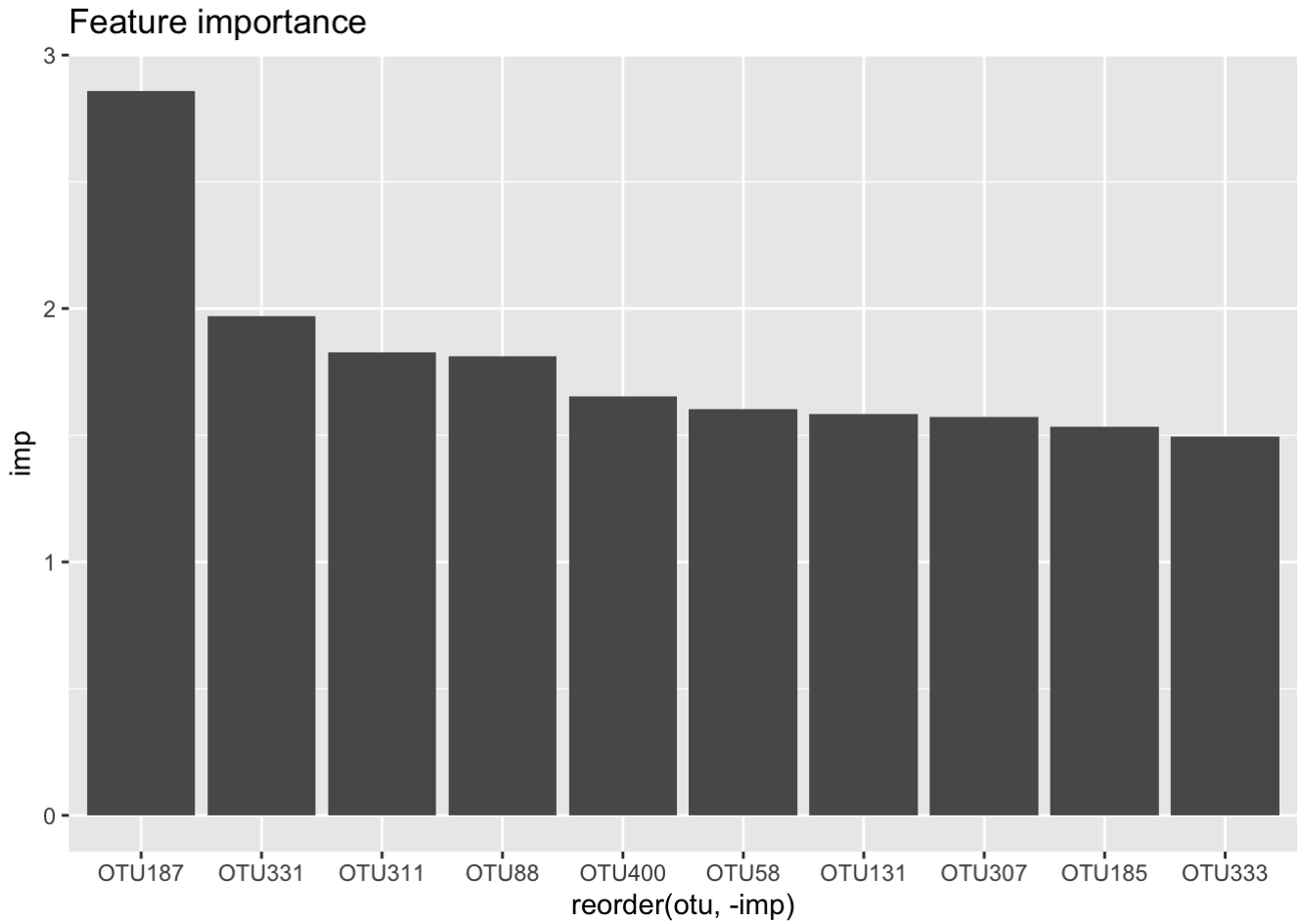
```
## [1] "AUC:  0.55"
```

```
print(paste('F1 Score: ',F1_Score(tssTest$disease,factor(ifelse(pred[,1] < 0.5, 't2d'
, 'n')),positive = 't2d')))
```

```
## [1] "F1 Score:  0.526315789473684"
```

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 i
mportance')
p
```




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

```
## [1] 25
```

With this graph we confirm our assumptions, since we return to see species 187, 331, 311, and 131 among others, so once again we think that filtering the data does not affect the results, since the variables that help to separate the classes are present in both filters of the data. Therefore, the differences in the success of the models of different filters (not preprocessed), may be due to the randomization of the partitions or the randomforest itself, since the variables that have the most weight are the same.

Conclusión

In conclusion, we can say that the differences between filtering are almost non-existent, while it seems that better results are obtained with the CLR processing than with the TSS, based on the AUC and the F1 obtained in the different models. In this way, with the CLR and filtering by the most important variables, we get very similar results to the best results obtained by Pasolli.

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.