Applying Random Forest Methods on Predictors for Diabetes

Danielle Novinski Victor Richerson Dinh Do

2023-11-19

Introduction

This is an introduction to random forests, a machine learning method widely used in a variety of applications such as ecology, medicine, astronomy, agriculture, traffic, and bioinformatics (Fawagreh, Gaber, and Elyan 2014). Random forest is an ensemble learning method composed of multiple decision trees. Breiman (Breiman 2001) proposed random forests to improve on decision trees for regression and classification tasks. Since this initial paper, many studies have been proposed to improve and extend the random forest method as well as use random forest for classification and regression techniques in a variety of applications. Our paper will provide an overview of the existing research and methods used.

Literature Review

Background

Decision trees are used for prediction and classification in a variety of artificial intelligence and machine learning tasks. Decision trees divide data by fields, creating subsets called nodes. Cutoffs are applied based on statistics (de Ville 2013). Researchers have sought to make improvements to decision tree classifiers, such as reducing bias and variance. Resampling training data and using multiple trees reduces bias and variance (de Ville 2013). Bootstrap aggregating (bagging) and AdaBoost were improvements over the standard decision tree. In addition, other multi-tree solutions came before the random forest method, including voting over multiple trees and using split methods of alternative splits and oblique splits (Breiman 1996). The Random Forest method is an ensemble method. Ensemble methods use several methods to solve a problem. Diversity among methods is indicated by methods making different errors. This leads to improvements in classifications (Fawagreh, Gaber, and Elyan 2014).

Dietterich (Dietterich 2000) compared a method of randomizing the split of C4.5 decision trees with AdaBoost and Bagging methods on 33 datasets. This study discussed the effect

of randomization leading to a diverse set of decision trees. The performance with "noise" or changed class labels for various percentages of the data was also tested since it was noted that AdaBoost did not do well with such noise, due to amplifying the effect of the noise. Breiman's paper, "Random Forests", (Breiman 2001) is attributed to be the first mention and description of the random forest technique. Random Forest classifiers are based on decision trees which have been generated from random subsets of the variables. The final classification is determined by majority vote of the decision trees.

Random Forests can be used for classification (predicting a class label) or regression (predicting a continuous value). An advantage of random forest is its performance with noisy data, since it does not weight the data and hence potentially amplify the weight of noise. Random forests do not overfit data due to the randomness of the selected variables used for the prediction. The "out of bag" estimate for a class label is the estimate using trees that do not contain the class label, and it is a good estimate of overall error rate. Random forests can have similar error rates as another machine learning algorithm, AdaBoost. Rodriguez, et. al. (Rodriguez, Kuncheva, and Alonso 2006) discuss bagging, boosting, and random forest methods and mention that AdaBoost is "equivalent to fitting an additive logistic regression model by a stage-wise estimation procedure," and also note that some of the math behind bounds and generalization error have been proven. It has also been noted that AdaBoost was successful because of imparting diversity to the ensemble, but there was a tradeoff in accuracy (Rodriguez, Kuncheva, and Alonso 2006).

Improvements and Expansions to Random Forest

Random forest extensions include non-parametric tests of significance, weighted voting, dynamic integration, weighted random sampling, online random forest (streamed / partial data), and genetic algorithms (Fawagreh, Gaber, and Elyan 2014). Verikas et. al (Verikas, Gelzinis, and Bacauskiene 2011) found that the random forest method is sensitive to tuning parameters such as number of variables selected and number of trees. Boulesteix et. al (Boulesteix et al. 2012) suggested modifications to the random forest approach regarding how individual trees are built, the construction of datasets, and how individual predictions are coalesced into one prediction.

For parameter tuning, if the number of trees increases with the number of predictors, the randomization process will lead to predictors having a good chance to be selected. Predictor selection is important. Some predictors are not informative. The number of predictors is important in order to moderate the effects of strong and weak predictors. The size of trees is determined by the splitting criteria. The size of the leaves is another parameter wherein good predictors could be overlooked if the leaf size threshold is too small. The authors suggest sampling without replacement to avoid bias, and sampling a certain number of observations from multiple classes (Boulesteix et al. 2012).

A study by Gardener et. al (Gardner and Chia-Tien Lo 2021) combined Random Forest methodology with PCA to improve precision. PCA is used to create new combination vectors for dependent features in this case. It is ideal to create a new feature when a dependency is found, but this methodology would take the guesswork out of it and automate the creation of the new features. The random forest technique would then incorporate the new features. This allows the single split of a new variable that combines multiple features. Another improvement is the fact that each tree has unique features due to the PCA algorithm being applied on each tree, so the trees are less correlated, which has been shown to increase accuracy.

Rodriguez et. al (Rodriguez, Kuncheva, and Alonso 2006) also attempted to improve on random forest methods by using PCA to create new features in a "rotation" method to classify the rotated data. The authors mention that rotation may not be optimal for feature extraction and there are other linear transformations/projections that could be used instead of PCA. This study used disjoint sets of data to create different classifiers but noted that it could be done use intersecting sets. PCA yields some zero eigenvectors and therefore the number of components varies. This study kept all the components rather than discard low ranking ones. The authors found that the rotation forest method has more accurate but less diverse (but still reasonably diverse) individual classifiers and that contrasts with random forests having more diverse but less accurate classifiers, concluding that the rotation method tested provided superior results.

Breiman in 2004 (Breiman 2004) attempted to create some simple random forest models where mathematical consistency could be proven, referring to a paper by Yi Lin & Yongho Jeon in 2002 (published 2006) (Y. Lin and Jeon 2006), that showed that the random forest method was a form of adaptive nearest neighbors, as the tree or split is a form of distance measure and therefore it is a nearest neighbor approach. The concept of strong and weak variables was introduced, as well as attempts to create equations to bound variance and bias for this simple model to prove consistency. This introduces the question: problems themselves aren't always consistent so can the solutions be consistent?

Applications

As the size of datasets increases (defined by the number of variables exceeding the number of observations), the performance of statistical methods decline and machine learning methods are preferred (Verikas, Gelzinis, and Bacauskiene 2011). Random forests have been used in several such studies related to biology and medicine. In applications such as these, there is usually a relationship between response and predictor variables and sometimes there are strong correlations between predictors. Random forests have helped with prediction, ranking of influential variables, and identifying variables with interactions. Other studies have found good performance in areas such as predicting customer churn, fraud detection, identifying bacteria and fish, and identifying eye disease (Verikas, Gelzinis, and Bacauskiene 2011).

Lin et al (S. Lin, Ji, and Pei 2019) analyzed a very large dataset in order to determine risk factors for diabetes, a disease affecting 425 million adults where chronic high blood sugar has effects such as damaging blood vessels, organs, nerves, and other complications. A study by Beaulac and Rosenthal (Beaulac and Rosenthal 2019) focused on prediction of students' major and completion status, finding they were able to predict program completion with 78.84% accuracy and predict choice of major with 47.41% accuracy. Moore, et. al. used the random forest method to predict conversion from MCI (mild cognitive impairment) to AD (Alzheimer's disease) through pairwise comparison of time series data (Moore, Lyons, and Gallacher 2019). The authors used a random forest on time series data with four data points and 60 trees, yielding 90% accuracy for the classification of neuroimage data of AD vs HC (healthy controls).

Methods

The general methodology of the Random Forest model is to create a number of decision trees that use a random subset of features for the prediction. For any given row, the data values are run through all of the decision trees that were created. The value for the class label is predicted based on the prediction by the majority of the trees (Breiman 2001). The reason for the randomness of the features is that it allows some decision tree models to be created with weaker predictive features, which increases model variability. Model variability is a key component for greater predictive accuracy, allowing weaker variables to still have some effect (Dietterich 2000). The criteria for splitting is generally to minimize Gini Impurity which is a measure of the homogeneity in the nodes. Low Gini Impurity implies that the items are more homogeneous (of the same class label).

Gini Impurity of a single node

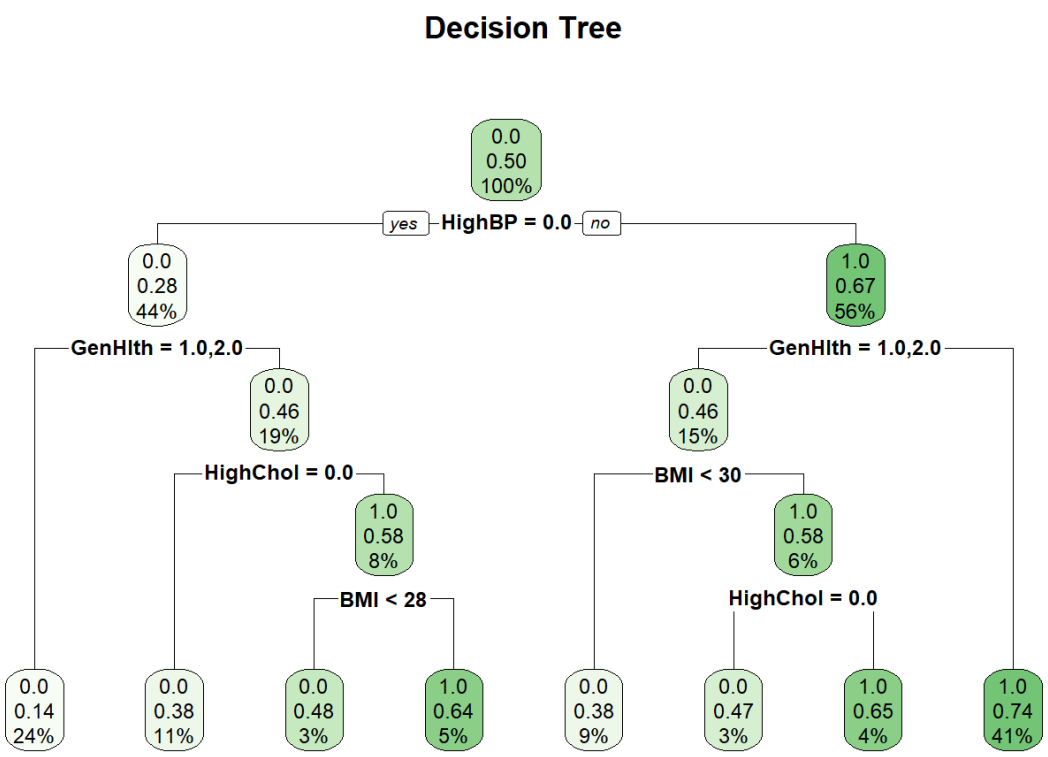
$$Gini(D) = 1 - \sum_{i=1}^{c} p(y_i)^2$$

where $p(y_i)$ is the probability that a sample belongs to class i, and c is the number of classes. As the sum of the probability squared approaches 1, Gini Impurity approaches 0. Impurity is 0 when all of the items are of the same class.

Gini Impurity of a node with child nodes (Robnik-Šikonja 2004)

$$Gini(A_i) = -\sum_{i=1}^{c} p(y_i)^2 + \sum_{j=1}^{m_i} p(v_{i,j}) \sum_{i=1}^{c} p(y_i|v_{i,j})^2$$

Below is a sample of a single decision tree.



The algorithm for random Random Forest is as follows (Dimitriadis and Liparas 2018):

(quoted directly from source)

The error rate for a random forest model can be calculated using a method called the "out of bag" estimate. This is calculated by the using prediction error rate for decision tree models in the random forest that do not contain the class label (Breiman 2001). Because the error rate can be approximated in this way, the data does not necessarily have to be split into test and train datasets as in other machine learning models (Breiman 2001). Regardless of that, splitting into test and train datasets continues to be a best practice, especially if comparing to or using the random forest method with other machine learning methods.

Sample code for creating a Random Forest in R:

Table 1: Sample Random Forest Probabilities

Row	Yes	No
1	0.136842105	0.863157895
2	0.595041322	0.404958678
3	0.990654206	0.009345794

Row	Yes	No

Analysis and Results

Data and Visualization

Data Source: Diabetes Health Indicators Dataset (Kaggle)

This dataset focuses on diabetes, a chronic health condition affecting millions of people worldwide. Diabetes can cause serious complications and lower quality of life; therefore, there is a need to try to predict an individual's risk of developing diabetes in the hopes that preventive measures can be taken to improve outcomes (Teboul 2021).

This dataset is a subset of the The Behavioral Risk Factor Surveillance System (BRFSS) dataset. The BRFSS is a survey conducted by the CDC to examine Americans' responses in the areas of health behaviors and conditions. The diabetes dataset includes 21 feature variables and 70,692 survey responses. The response variable is whether or not an individual has diabetes (indicated by 0 for no or 1 for pre-diabetes or diabetes). This dataset has a 50-50 binary split, meaning the data is evenly split between having diabetes or not (Teboul 2021).

Table 2: Data Fields and Description

Data Field	Description of Data	Range of Values
HighBP	Told by a healthcare professional they have high blood pressure.	(0, 1)
HighChol	Told by a healthcare professional they have high cholesterol.	(0, 1)
CholCheck	Cholesterol check within past 5 years.	(0, 1)
BMI	Body Mass Index	(12, 98)
Smoker	Smoked at least 100 cigarettes lifetime.	(0, 1)
Stroke	Told by a healthcare professional they have had a stroke.	(0, 1)
HeartDiseaseorAttack	Reported having heart disease or heart attack.	(0, 1)
PhysActivity	Physical activity/exercise in past 30 days.	(0, 1)
Fruits	Consume fruit 1+ times daily.	(0, 1)
Veggies	Consume vegetables 1+ times daily.	(0, 1)
HvyAlcoholConsump	Heavy drinkers (adult men having more than 14 drinks per week and adult women having more than 7 drinks per week)	(0, 1)
AnyHealthcare	Reported having any kind of health insurance/healthcare coverage.	(0, 1)

Data Field	Description of Data	Range of Values
NoDocbcCost	Reported needing to see a doctor in the past year, but unable because of cost.	(0, 1)
GenHlth	Statement of general health.	(1, 5)
MentHlth	Statement of mental health in past 30 days (not good).	(0, 30)
PhysHlth	Statement of physical health in past 30 days (not good).	(0, 30)
DiffWalk	Difficulty walking up or down stairs.	(0, 1)
Sex	Sex/gender	(0, 1)
Age	Age group	(1, 13)
Education	Highest grade completed.	(1, 6)
Income	Annual household income.	(1, 8)

```
library(tidyverse)
#library(dplyr)
library(rpart)
library(rpart.plot)
library(caTools)
library(caret)
library(gtsummary)
library(ggplot2)
#library(ggcorrplot)
library(gridExtra)
#library(visreg)
library(splitTools)
library(ranger)
library(vip)
library(lessR)
```

```
randseed <- 123456
set.seed(randseed)

#data types for data frame columns
coltypes <- c("f",#Diabetes_012
              "f",#HighBP
              "f",#HighChol
              "f",#CholCheck
              "n",#BMI
```

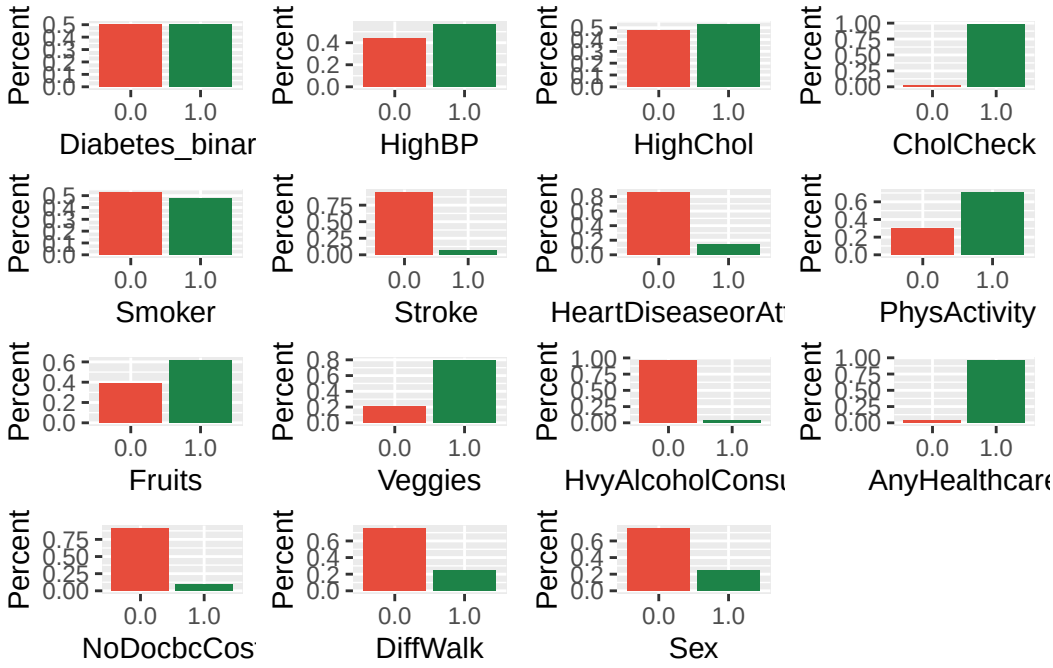
```
                "f",#Smoker
                "f",#Stroke
                "f",#HeartDiseaseorAttack
                "f",#PhysActivity
                "f",#Fruits
                "f",#Veggies
                "f",#HvyAlcoholConsump
                "f",#AnyHealthcare
                "f",#NoDocbcCost
                "f",#GenHlth
                "n",#MentHlth
                "n",#PhysHlth
                "f",#DiffWalk
                "f",#Sex
                "f",#Age
                "f",#Education
                "f" #Income
                )
coltypes_collapsed <- paste(coltypes, collapse="")

#read the full dataset into a dataframe - 50/50 split dataset
df <-
 ↪  read_csv("diabetes_binary_5050split_health_indicators_BRFSS2015.zip",
 ↪  col_types = coltypes_collapsed)

#set factor level order for factor variables
df$Age <- factor(df$Age, levels = c("1.0", "2.0", "3.0", "4.0", "5.0",
 ↪  "6.0", "7.0", "8.0", "9.0", "10.0", "11.0", "12.0", "13.0"))
df$GenHlth <- factor(df$GenHlth, levels = c("1.0", "2.0", "3.0", "4.0",
 ↪  "5.0"))
df$Education <- factor(df$Education, levels = c("1.0", "2.0", "3.0",
 ↪  "4.0", "5.0", "6.0"))
df$Income <- factor(df$Income, levels = c("1.0", "2.0", "3.0", "4.0",
 ↪  "5.0", "6.0", "7.0", "8.0"))

binaryLevels <- c("0.0", "1.0")
#set all binary factor fields up the same way
df$Diabetes_binary <- factor(df$Diabetes_binary, levels = binaryLevels)
df$HighBP <- factor(df$HighBP, levels = binaryLevels)
df$HighChol <- factor(df$HighChol, levels = binaryLevels)
df$CholCheck <- factor(df$CholCheck, levels = binaryLevels)
```

```r
df$Smoker <- factor(df$Smoker, levels = binaryLevels)
df$Stroke <- factor(df$Stroke, levels = binaryLevels)
df$HeartDiseaseorAttack <- factor(df$HeartDiseaseorAttack, levels =
 ↪  binaryLevels)
df$PhysActivity <- factor(df$PhysActivity, levels = binaryLevels)
df$Fruits <- factor(df$Fruits, levels = binaryLevels)
df$Veggies <- factor(df$Veggies, levels = binaryLevels)
df$HvyAlcoholConsump <- factor(df$HvyAlcoholConsump, levels =
 ↪  binaryLevels)
df$AnyHealthcare <- factor(df$AnyHealthcare, levels = binaryLevels)
df$NoDocbcCost <- factor(df$NoDocbcCost, levels = binaryLevels)
df$DiffWalk <- factor(df$DiffWalk, levels = binaryLevels)
df$Sex <- factor(df$DiffWalk, levels = binaryLevels)
```

```r
#print gtsummary
smr <-
  tbl_summary(
    df,
    include = c("Diabetes_binary", "HighBP", "HighChol","CholCheck",
     ↪  "BMI", "Smoker", "Stroke", "HeartDiseaseorAttack",
     ↪  "PhysActivity", "Fruits", "Veggies","HvyAlcoholConsump",
     ↪  "AnyHealthcare", "NoDocbcCost", "GenHlth", "MentHlth",
     ↪  "PhysHlth", "DiffWalk","Sex", "Age", "Education", "Income"),
    type = list(BMI ~ "continuous2", MentHlth  ~ "continuous2", PhysHlth
     ↪  ~ "continuous2"),
    statistic = all_continuous() ~ c("{median} ({p25}, {p75})", "{min},
     ↪  {max}")
  ) %>%
  add_n() %>% # add column with total number of non-missing observations
  bold_labels()
#smr

#use the summary table data to output bar charts
#create lists for charts
binary_chart_list <- list()
multiple_chart_list <- list()

#loop field stats from internals of summary data
for (field in smr$meta_data$df_stats) {
  #print(field)
```

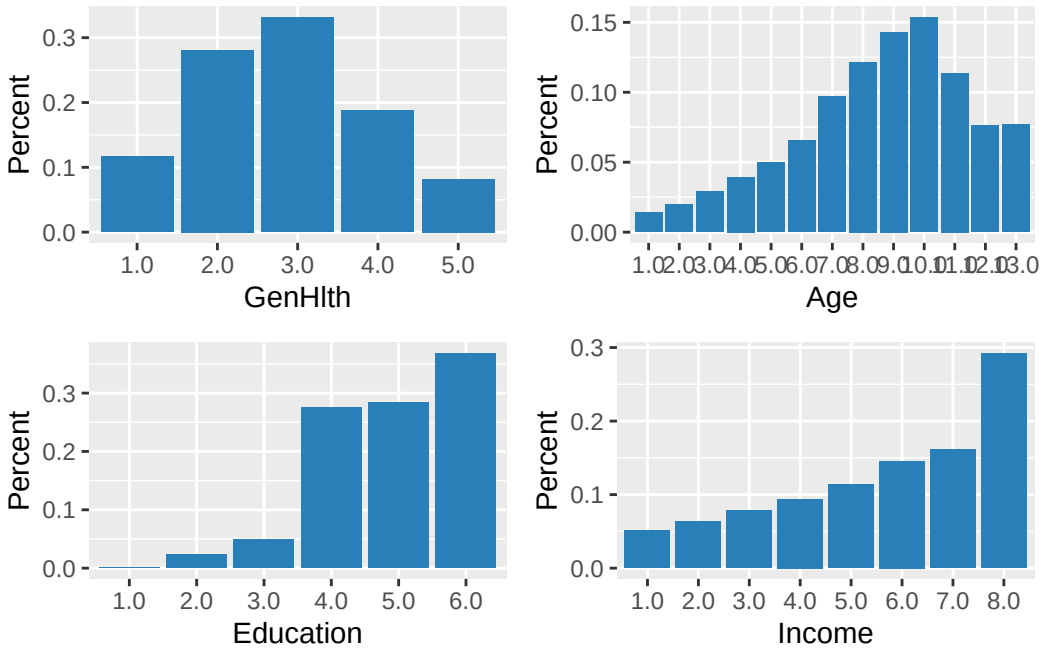
```
  #if this is a field with a probability field
  if ("p" %in% names(field)) {
    #add chart to correct list
    if (nrow(field) == 2) {
      #create ggplot for field with binary values
      field_chart <- ggplot(field, aes(x = label, y = p, fill = label))
↪  +
        geom_col() +
        xlab(field$variable[1]) + ylab("Percent") +
        theme(legend.position = "none") +
        scale_fill_manual(values=c("#E74C3C","#1D8348")) +
        scale_x_discrete(limits = field$label)

      binary_chart_list[[field$variable[1]]] <- field_chart
    } else {
      #create ggplot for field with multiple values
      field_chart <- ggplot(field, aes(x = label, y = p,
↪  fill="#2980B9")) +
        geom_col() +
        xlab(field$variable[1]) + ylab("Percent") +
        theme(legend.position = "none") +
        scale_fill_manual(values="#2980B9") +
        scale_x_discrete(limits = field$label)

      multiple_chart_list[[field$variable[1]]] <- field_chart
    }
  }
}
```

```
#print charts in grid
grid.arrange(grobs=binary_chart_list, ncol=4)
```



Summary of the binary variables in the data set.

```
grid.arrange(grobs=multiple_chart_list, ncol=2)
```



Summary of the multi-valued variables in the data set.

Statistical Modeling

The goal of this analysis is to gauge the level of predictive accuracy of the variables in the data set while using the random forest method. As part of the analysis we will be utilizing the importance measures output to determine the most important questions in the aforementioned Behavioral Risk Factor Surveillance System (BRFSS) questionnaire. Once we have the variables ranked by importance, we will create a reduced version of the BRFSS questionnaire with less questions and gauge accuracy of the reduced version of the BRFSS questionnaire. With this information we can increase participation by having a shorter time commitment to complete the survey.

Our initial random forest analysis will use the default parameters. The ranger package in R implements a fast version of random forest. We are classifying the Diabetes Binary field using all of the remaining fields in the data set. We are also outputting the variable importance using Gini impurity as the importance measure.

```
#create actual random forest with full list of variables for analysis
diabetes.forest <- ranger(Diabetes_binary ~ ., data=df, importance =
 ↪  'impurity')
```

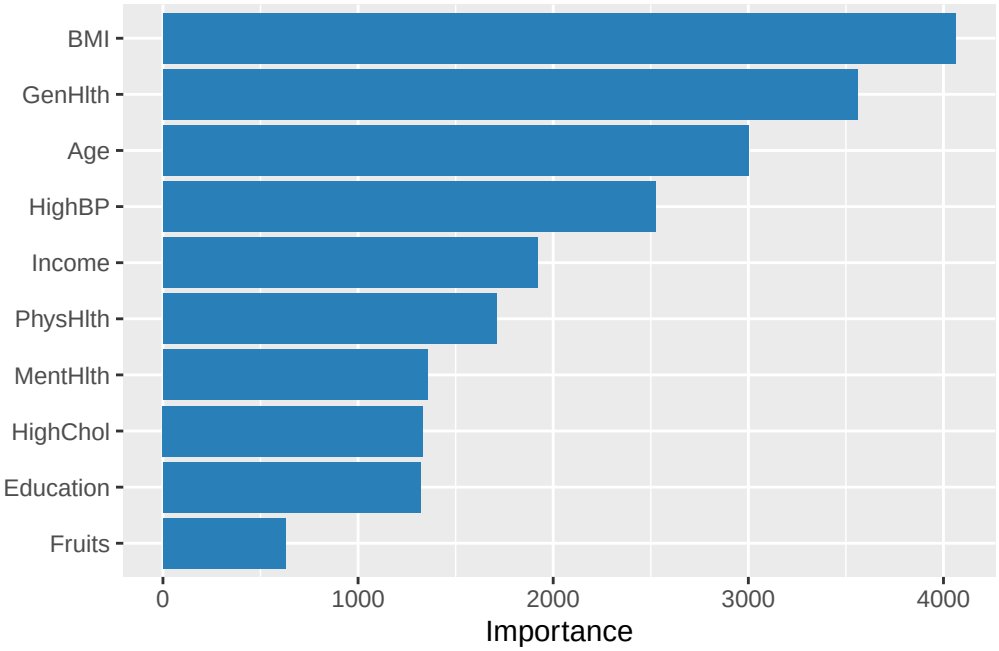
```r
print(diabetes.forest)
```

```
Ranger result

Call:
 ranger(Diabetes_binary ~ ., data = df, importance = "impurity")

Type:                             Classification
Number of trees:                  500
Sample size:                      70692
Number of independent variables:  21
Mtry:                             4
Target node size:                 1
Variable importance mode:         impurity
Splitrule:                        gini
OOB prediction error:             25.36 %
```

This shows that our estimated prediction error is around 25.6%, for an accuracy rate of around 74.4%.

We can check the variable importance to determine the most important variables contributing to the classification.

```r
#create vi object for variable importance graph
vi_values <- vi(diabetes.forest)
#create variable importance graph
vip(vi_values, aesthetics = list(fill="#2980B9"))
```



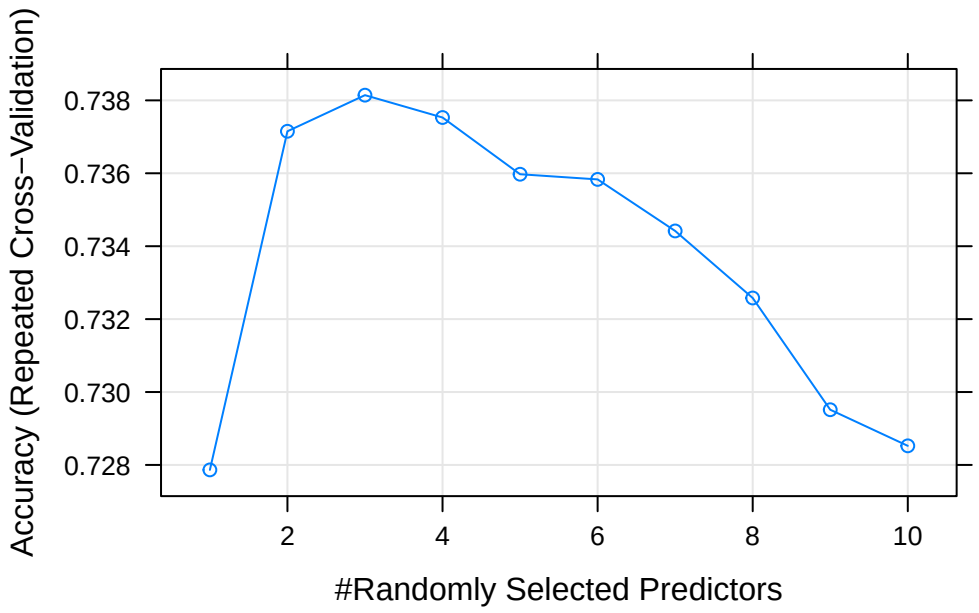
The most important variables impacting Diabetes were found to be BMI, General Health, High Blood Pressure, and Age.

Hyperparameter Tuning

Random Forest and other machine learning methods require hyperparameter tuning to obtain optimal settings for variables. Hyperparameters are parameters that can vary and different values can have a large effect on the model. In this example a grid search is used to find the optimal number of variables to use for the split, which uses the "mtry" parameter. For speed and efficiency, a stratified sample of 10% of the dataset was used for this algorithm. Stratified means that an equal number of the target class (e.g. Diabetes_binary) are present in the sample.

```
#split data and stratify on class label
parts <- splitTools::partition(df$Diabetes_binary, p=c(sample = 0.1,
 ↪  remainder = 0.9))
df.sample <- df[parts$sample,]

#set the metric to be used for tuning
metric <- "Accuracy"
```

```
#setup control settings for caret train(tune) call
control <- trainControl(method="repeatedcv", number=5, repeats=3,
 ↪  search="grid")
#set up train (tune) parameters
tunegrid <- expand.grid(.mtry=c(1:10), .splitrule='gini',
 ↪  .min.node.size=1)
#call to caret train(tune) using sample of data
rf_gridsearch <- train(Diabetes_binary~., data=df.sample,
 ↪  method="ranger", metric=metric, tuneGrid=tunegrid,
 ↪  trControl=control)

#plot results for parameter tuning
plot(rf_gridsearch)
```



This plot shows that the optimal value for number of variables to split at each node peaks at 3 and declines from there. Using this value should yield the optimal accuracy rate. The number of trees was tuned manually and it was found that about 2000 trees is optimal.

```
#now try with specific parameters found by tuning
diabetes.forest <- ranger(Diabetes_binary ~ ., data=df,
```

```
                            mtry=3, num.trees=2000, verbose = FALSE)
print(diabetes.forest)
```

```
Ranger result

Call:
 ranger(Diabetes_binary ~ ., data = df, mtry = 3, num.trees = 2000,      verbose = FALSE)

Type:                             Classification
Number of trees:                  2000
Sample size:                      70692
Number of independent variables:  21
Mtry:                             3
Target node size:                 1
Variable importance mode:         none
Splitrule:                        gini
OOB prediction error:             25.12 %
```

The tuned model has a prediction error of 25.1% for a higher accuracy of 74.9%.

Questionnaire Reduction

The goal of the analysis is to reduce the number of questions needed on the Behavioral Risk Factor Surveillance System (BRFSS) and still have high accuracy. Here we have selected the top 4 most important questions and re-run the classification to gauge accuracy.

```
#use reduced list of parameters found by assessing variable importance
diabetes.forest.reduced <- ranger(Diabetes_binary ~ BMI + GenHlth +
 ↪  HighBP + Age, data=df)
print(diabetes.forest.reduced)
```

```
Ranger result

Call:
 ranger(Diabetes_binary ~ BMI + GenHlth + HighBP + Age, data = df)

Type:                             Classification
Number of trees:                  500
Sample size:                      70692
Number of independent variables:  4
```

```
Mtry:                           2
Target node size:               1
Variable importance mode:       none
Splitrule:                      gini
OOB prediction error:           26.00 %
```

From this result we can see that using a questionnaire with only 4 questions will allow us to still have 74.0% accuracy (prediction error 26.0%). Amazing!

Random Forest Prediction

Next we want to illustrate how to use prediction with random forest. For prediction the first thing we want to do is to split the data into train (70%) and test (30%) sets.

Next the model is fit on the train set, leaving the test set as "unseen" data. Finally the model is used to predict on the test set and the true value of the test set is compared to the predicted value.

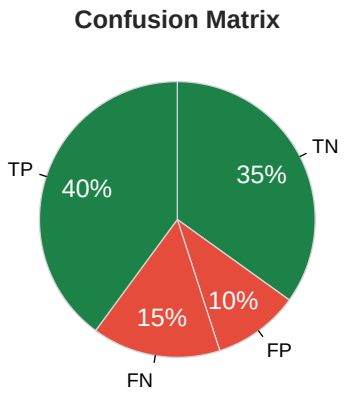
```
#split data and stratify on class label
parts <- splitTools::partition(df$Diabetes_binary, p=c(train = 0.7, test
 ↪  = 0.3))
df.train <- df[parts$train,]
df.test <- df[parts$test,]

#train rf with train set
diabetes.forest <- ranger(Diabetes_binary ~ ., data=df.train,
                          mtry=3, num.trees=2000, verbose = FALSE)

#predict using test set
preds <- predict(diabetes.forest, data = df.test, seed=randseed)
```

Now we compare the prediction with the true value to determine the model accuracy.

```
#create confusion matrix of prediction vs actual on test set
confusion.test <- confusionMatrix(data=df.test$Diabetes_binary,
 ↪  reference=preds$predictions)

#create dataframe to use for the chart
cf <- data.frame(cfm = confusion.test$table)
#add percentage field
cf <- cf %>%  mutate(cfm.Percent = cfm.Freq / sum(cfm.Freq))
#add labels for TN, FN, TP, FP
```

```
cf <- cf %>%  mutate(cfm.Label =
                       case_when(cfm.Prediction == "0.0" & cfm.Reference
                        ↪  == "0.0" ~ "TN",
                                 cfm.Prediction == "0.0" & cfm.Reference
 ↪  == "1.0" ~ "FN",
                                 cfm.Prediction == "1.0" & cfm.Reference
 ↪  == "1.0" ~ "TP",
                                 cfm.Prediction == "1.0" & cfm.Reference
 ↪  == "0.0" ~ "FP"))

#add color field to go with the labels - green is good and red is bad
cf <- cf %>%  mutate(cfm.Label.Color =
                       case_when(cfm.Prediction == "0.0" & cfm.Reference
                        ↪  == "0.0" ~ "#1D8348",
                                 cfm.Prediction == "0.0" & cfm.Reference
 ↪  == "1.0" ~ "#E74C3C",
                                 cfm.Prediction == "1.0" & cfm.Reference
 ↪  == "1.0" ~ "#1D8348",
                                 cfm.Prediction == "1.0" & cfm.Reference
 ↪  == "0.0" ~ "#E74C3C"))
#tried to set it to output in the order I wanted but it seems to have
 ↪  it's own order
#this did nothing for the sort order of the output chart
#cf <- cf %>% mutate(cfm.Label = factor(cfm.Label, levels = c("TN",
 ↪  "FN", "TP", "FP")))

#output the chart
PieChart(x = cfm.Label, y = cfm.Percent, hole = 0, values = "%", data =
 ↪  cf,
         clockwise = T, quiet = T, fill = cf$cfm.Label.Color,
          ↪  values_size = 1.2,
         main = "Confusion Matrix")
```

A confusion matrix calculates True Positive, True Negative, False Positive, and False Negative values. True Positive (TP) is a sample classified as positive which is in fact positive. True Negative (TN) refers to a sample classified as negative which indeed is negative. False Positive (FP) refers to a sample incorrectly classified as positive when it is in fact negative. False Negative (FN) refers to a sample incorrectly classified as negative when it is in fact positive. From these values, the accuracy can be obtained as:



$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

(Kundu 2022)

This allows us a deeper understanding of the model, as we can easily see the model's accuracy as it relates to positive and negative classifications. In this case our model's accuracy is approximately 75% as illustrated in the pie chart above.

Conclusion

We were able to use the Random Forest method to classify the data into Diabetic and Non-Diabetic with approximately 74.9% accuracy (prediction error 25.1%). Additionally, we showed how to use the random forest classifier to rank the importance measures of the variables to determine the most important variables contributing to the classification. Using this information we were able to select the top variables for a reduced version of the Behavioral Risk Factor Surveillance System (BRFSS) of 4 questions compared to the original 20 questions. The classifier with the reduced number of questions is able to classify the data with approximately 74.0% accuracy (prediction error 26.0%).

We demonstrated hyperparameter tuning for Random Forest for the mtry parameter and manually tuned the number of trees. We also demonstrated using random forest for prediction of unseen data and showed the method of gauging prediction accuracy comparing the true value with the predicted value on the unseen test data. We appreciate this opportunity to provide an introduction to the purpose and usage of the random forest method for those who are not familiar with it.

References

Beaulac, Cédric, and Jeffrey S. Rosenthal. 2019. "Predicting University Students' Academic Success and Major Using Random Forests." *Research in Higher Education* 60 (7): 1048–64. http://www.jstor.org/stable/45217777.

Boulesteix, Anne-Laure, Silke Janitza, Jochen Kruppa, and Inke R. König. 2012. "Overview of Random Forest Methodology and Practical Guidance with Emphasis on Computational Biology and Bioinformatics." *WIREs Data Mining and Knowledge Discovery* 2 (6): 493–507. https://doi.org/10.1002/widm.1072.

Breiman, Leo. 1996. "Bagging Predictors." *Machine Learning* 123–140.

———. 2001. "Random Forests." *Machine Learning* 45: 5–32.

———. 2004. "Consistency for a Simple Model of Random Forests: Technical Report 670." Statistics Department, University of California at Berkeley.

de Ville, Barry. 2013. "Decision Trees." *WIREs Computational Statistics* 5 (6): 448–55. https://doi.org/10.1002/wics.1278.

Dietterich, Thomas G. 2000. "An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization." *Machine Learning* 40: 139–57.

Dimitriadis, S. I., and D. Liparas. 2018. "Alzheimer's Disease Neuroimaging Initiative. How Random Is the Random Forest? Random Forest Algorithm on the Service of Structural Imaging Biomarkers for Alzheimer's Disease: From Alzheimer's Disease Neuroimaging Initiative (ADNI) Database." *Neural Regeneration Research* 13 (6): 962–70. https://doi.org/10.4103/1673-5374.233433.

Fawagreh, K., M. Gaber, and E. Elyan. 2014. "Random Forests: From Early Developments to Recent Advancements." *Systems Science & Control Engineering* 2 (9): 602–9.

Gardner, Charles, and Dan Chia-Tien Lo. 2021. "PCA Embedded Random Forest," 1–6. https://doi.org/10.1109/SoutheastCon45413.2021.9401949.

Kundu, Rohit. 2022. "Confusion Matrix: How to Use It & Interpret Results." https://www.v7labs.com/blog/confusion-matrix-guide.

Lin, Shaofu, Wei Ji, and Jiangtao Pei. 2019. "A Method for Selecting Diabetes Features Based on Random Forest." *Journal of Physics: Conference Series* 1237 (2). https://www.proquest.com/docview/2566217454.

Lin, Yi, and Yongho Jeon. 2006. "Random Forests and Adaptive Nearest Neighbors." *Journal of the American Statistical Association* 101: 578–90. https://doi.org/10.1198/016214505000001230.

Moore, P. J., T. J. Lyons, and J. Gallacher. 2019. "Random Forest Prediction of Alzheimer's Disease Using Pairwise Selection from Time Series Data." *PLoS ONE* 14 (2). https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0211558.

Robnik-Šikonja, Marko. 2004. "Improving Random Forests." In *Machine Learning: ECML 2004*, 359–70. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-30115-8_34.

Rodriguez, Juan J., Ludmilla I. Kuncheva, and Carlos J. Alonso. 2006. "Rotation Forest: A New Classifier Ensemble Method." *IEEE Transactions On Pattern Analysis And Machine*

Intelligence 28 (10): 1619–30.

Teboul, Alex. 2021. "Diabetes Health Indicators Dataset." https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset/.

Verikas, A., A. Gelzinis, and M. Bacauskiene. 2011. "Mining Data with Random Forests: A Survey and Results of New Tests." *Pattern Recognition* 44 (2): 330–49. https://doi.org/10.1016/j.patcog.2010.08.011.