TNSA Household Wealth Index

Ömer Şahin

Introduction

2013 Turkey Demographic and Health Survey (TDHS-2013), fertility levels and trends, infant and child mortality, family planning and maternal and child health issues a sample survey at the national level designed to provide information on. And also, gives information about the wealth index of the household. In this work, the wealth index analysis and prediction according to other information about a household are evaluated.

Data Analyze

Libraries

Required libraries for data preparation and analyze:

```
library(readr)    # Data loading
library(tidyr)    # Seperate data column
library(ggplot2)  # Visualize
```

Load Household Dataset

Simplified household data of the TNSA dataset is loaded.

```
household <- read.table(file="household.csv", sep=";", header = TRUE, na.strings = c("" , " ","\t ", "Mi
```

Each household row has "case_id" field. These values are unique for each sample. The ID column is dropped due to has no contribution to the training.

```
household <- household[, -1]  # drop case id
```

In house ownership column has only one "Other" feature. Therefore, this row is evaluated as outliers and dropped.

```
household <- household[-which(household$house_ownership == "Other"), ]
household$house_ownership <- factor(household$house_ownership)
```

In this stage, the number of samples:

```
nrow(household)
```

```
## [1] 11793
```

The dataset has some rows with unknown attributes. The number of rows with missing values is:

```
sum(!complete.cases(household))
```

```
## [1] 178
```

The number of rows that are not complete can be ignored compared to the total number of samples, and these samples are dropped.

```
household <- na.omit(household)
```

At the end of the data clearing, the number of samples:

```
nrow(household)
```

```
## [1] 11615
```

Combined Region

The household data contain the combined region field that consists of a combination of a cardinal direction, region, and settlement. These fields are separated into three.

```
head(household$region_combined)
```

```
## [1] West - Istanbul - Urban/Metropol West - Istanbul - Urban/Metropol
## [3] West - Istanbul - Urban/Metropol West - Istanbul - Urban/Metropol
## [5] West - Istanbul - Urban/Metropol West - Istanbul - Urban/Metropol
## 35 Levels: Central - Aegean - Rural ... West - West Marmara - Urban
```

```
household <- separate(household, region_combined, c("cardinal_direction", "region", "settlement"), sep =
household$cardinal_direction <- as.factor(household$cardinal_direction)
household$region <- as.factor(household$region)
household$settlement <- as.factor(household$settlement)

head(household[, 2:4])
```

```
##   cardinal_direction     region      settlement
## 1               West   Istanbul   Urban/Metropol
## 2               West   Istanbul   Urban/Metropol
## 3               West   Istanbul   Urban/Metropol
## 4               West   Istanbul   Urban/Metropol
## 5               West   Istanbul   Urban/Metropol
## 6               West   Istanbul   Urban/Metropol
```

Wealth Index

The aim of this project is predicting the wealth index of the household. There is an order between wealth index values. Therefore, wealth index factor is releveled.

```
# Refactor levels of wealth index
household$wealth_index <- factor(household$wealth_index,
                                 levels = c("Poorest", "Poorer", "Middle", "Richer", "Richest"))
levels(household$wealth_index)
```

```
## [1] "Poorest" "Poorer"  "Middle"  "Richer"  "Richest"
```

Attribute Relation

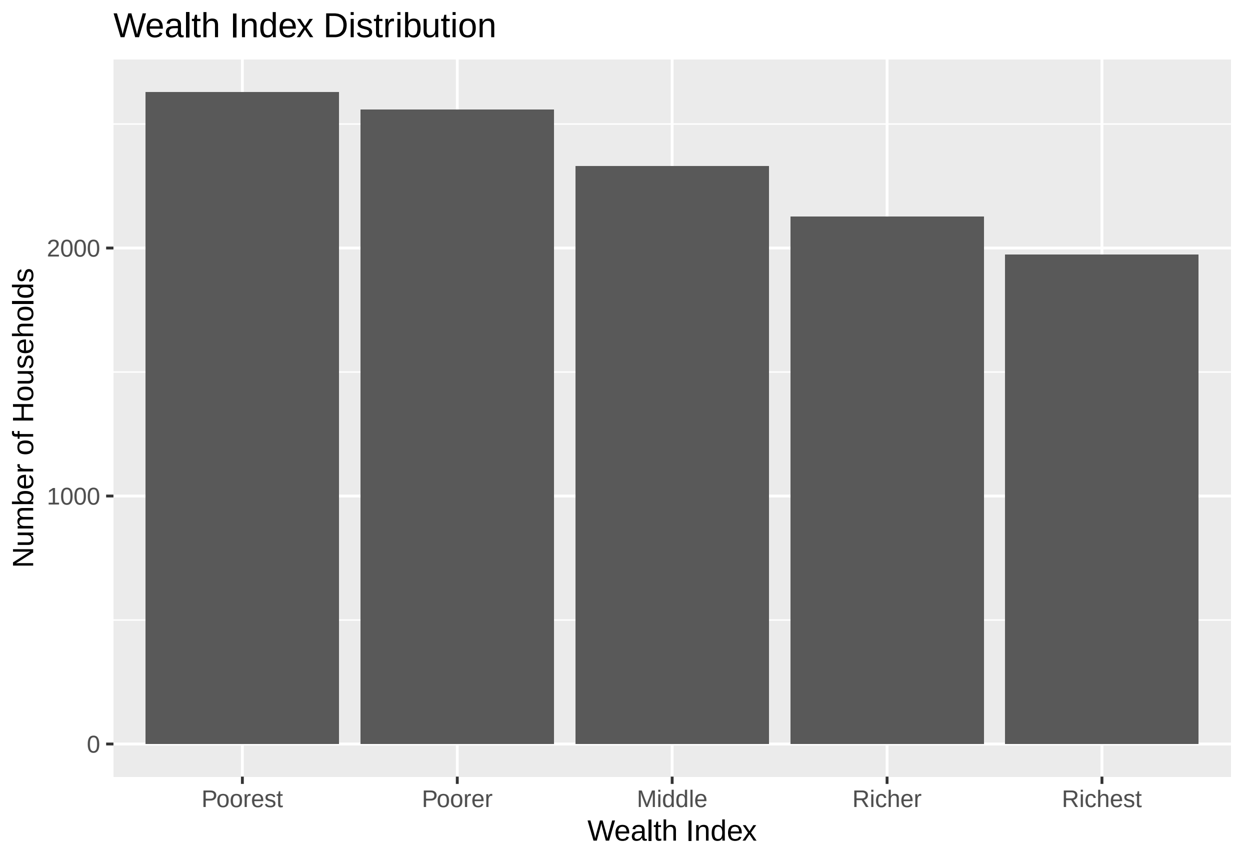
Some attributes have a relation to each other. These relations can be represented as a ratio between them. In this way, all household samples will have attributes that in the same range even they are in the natural number range. After extending columns with the rate of related ones, duplicate columns are dropped.

```
# Rate of related attributes
household$man_member_rate = (household$household_member - household$woman_member) / household$household_
household$woman_member_rate = household$woman_member / household$household_member
household$child_member_rate = household$children_under_5 / household$household_member
household$bedroom_rate = household$bedroom_number / household$rooms_number

# Drop duplicated columns with rate values
household <- household[ , -which(names(household) %in% c("woman_member",  # woman_member_rate
                                                          "children_under_5", # children_member_rate
                                                          "bedroom_number"))]  # bedroom_rate
```

Data Distribution

The wealth index of a household is the target that is tried to predict. The distribution of the wealth index is quite balanced. It helps to ensure the model does not tend to a class when the target class is balanced.

```r
ggplot(household, aes(x = wealth_index)) + geom_bar() +
        xlab("Wealth Index") + ylab("Number of Households") +
        ggtitle(label = "Wealth Index Distribution")
```



Distributions of the some of binary attributes about households are too imbalanced. There are not enough counter samples for these attributes. Therefore, these attributes can misguide the model when predicting the wealth index. Imbalanced attributes:

```r
summary(household[, c("refrigerator", "garbage_grinder", "washing_machine", "washer_dryer",
                      "home_theather", "mobile_phone", "taxi_minibus", "tractor", "motorcycle")])
```

```
##  refrigerator garbage_grinder washing_machine washer_dryer home_theather
##  No :  175    No :11544       No :  515       No :11411    No :11282
##  Yes:11440    Yes:   71       Yes:11100       Yes:  204    Yes:  333
##  mobile_phone taxi_minibus tractor     motorcycle
##  No :  565    No :11126    No :10642   No :10806
##  Yes:11050    Yes:  489    Yes:  973   Yes:  809
```

According to results, nearly all households have a refrigerator, washing machine, and a mobile phone. On the other hand, nearly none of the households have the garbage grinder, washer dryer or home theatre. As a result, these attributes are not used for training the model.

```r
household <- household[ , -which(names(household) %in% c("refrigerator", "garbage_grinder",
                                                          "washing_machine", "washer_dryer",
                                                          "home_theather", "mobile_phone",
                                                          "taxi_minibus", "tractor", "motorcycle"))]
```

Province column in the dataset is also imbalanced and has too many different classes due to there are 81 cities in Turkey. Besides, the Random Forest model cannot handle too many categories. Therefore, the province column is dropped to evaluate other models in equal conditions.

```
head(summary(household$province))
```

```
##    Adana Adiyaman    Afyon     Agri  Aksaray   Amasya
##      441       84       35      162       75       39
```

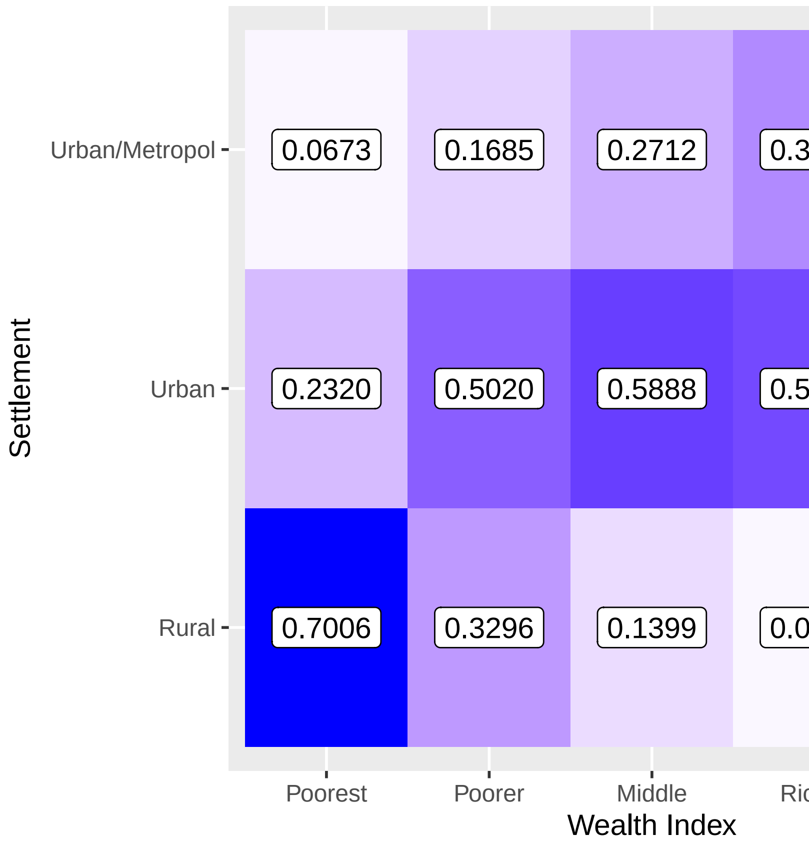
```
household <- household[, -which(names(household) == "province")]
```

Outliers

TNSA Household dataset was collected under controlled by qualified observers. Therefore, Obtained information about households is accepted as truth. Information that does not have enough observations or balanced distribution is not used in the analysis to make a proper evaluation.

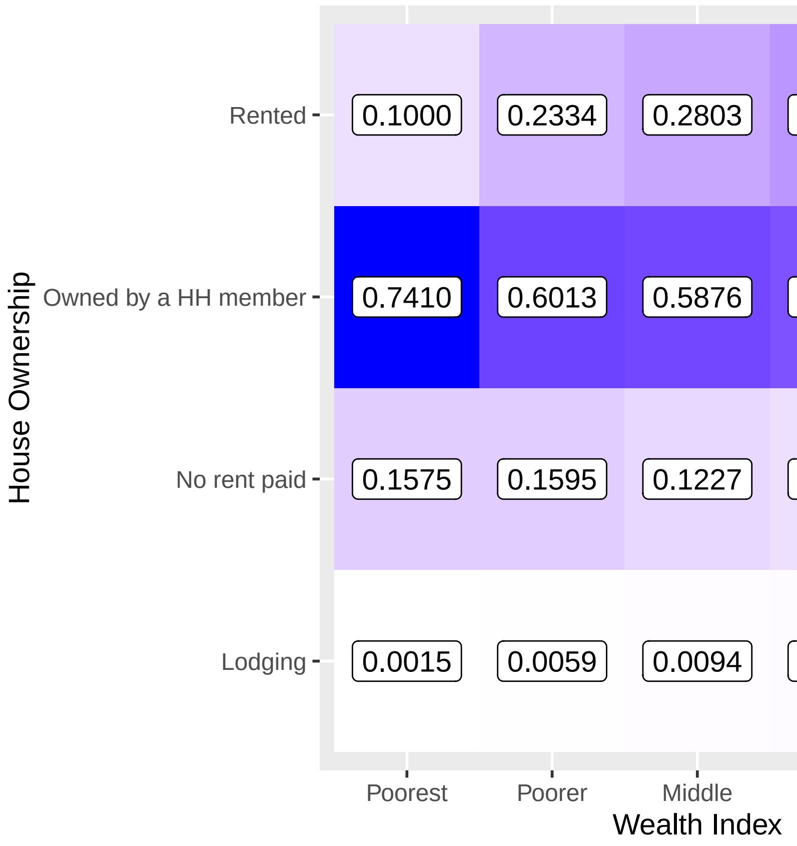
Analyze

At the end of data preparation, data analyzing by visualizing is done over some information about the household dataset.



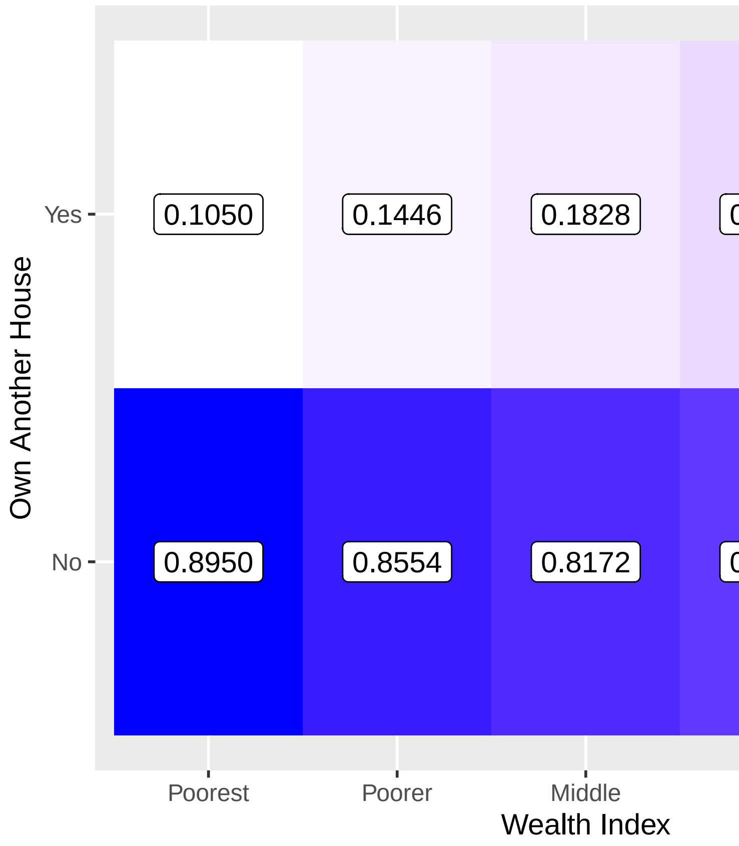
The relation between wealth index and settlement area:

Most of the people live in urban areas. On the other hand, when the wealth index is reducing, people tend to live in rural areas or people that live in rural areas are poorer than the others.



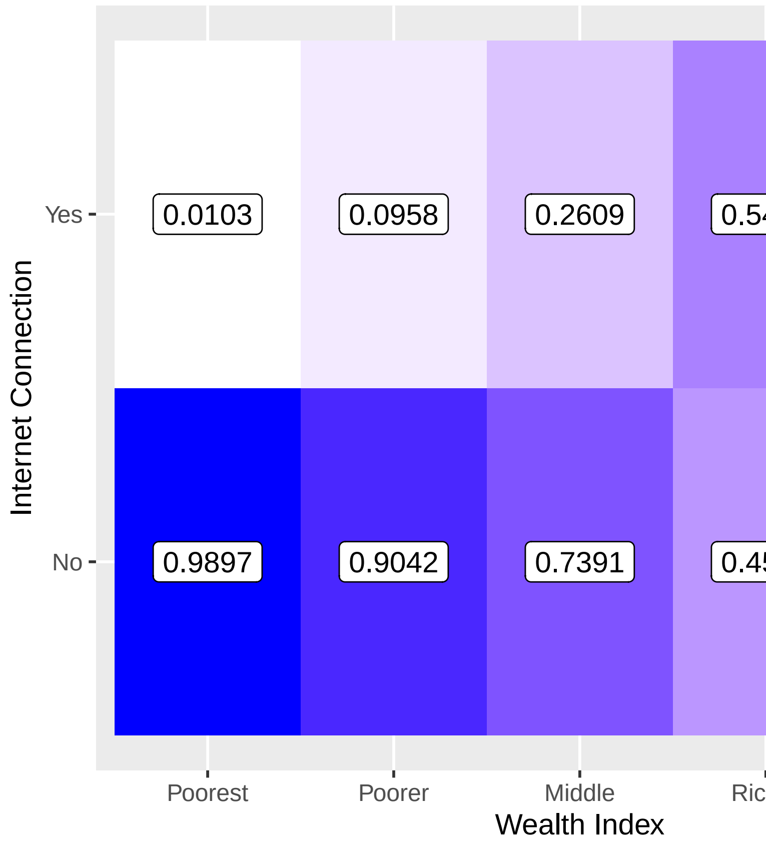
The relation between wealth index and house ownership:

Contrary to expectation, poor households live in their own houses and richer households may prefer to live in a rented house.



The relation between wealth index and owning another house:

According to the data, people do not much interest in having another house. Still, rich households much more have another house than the poor ones as expected.



The relation between internet connection and wealth index:

According to 2013 data, the internet connection is still not very prevalent and seen as a luxury feature.

Classification

Each sample row contains 34 features about household and wealth index. The wealth index of the household is the target value that is tried to predict by the classifier. Different types of classifiers are compared with others and itself that uses a different parameter set.

Libraries

Required libraries for data classification and performace evaluation:

```
library(performanceEstimation)   # Performance estimation
library(caret)         # Confusion matrix
library(CORElearn)     # Feature extraction
library(DMwR2)         # Decision tree
library(rpart)         # Decision tree
library(rpart.plot)    # Decision tree visualize
library(e1071)         # Support vector machine
library(randomForest)  # Random forest
library(adabag)        # Adabag
library(h2o)           # Neural network
```

Split Train/Test Set

Randomly selected 20 percent of the dataset is used as a test and the rest of them are used for training models.

```r
set.seed(1024)
samples <- sample(1:nrow(household), nrow(household)*0.8)
train <- household[samples, ]
test <- household[-samples, ]
```

Analyze Decision Boundary

By using a simple decision tree, tried to understand the decision boundary for classification household dataset according to the wealth index.

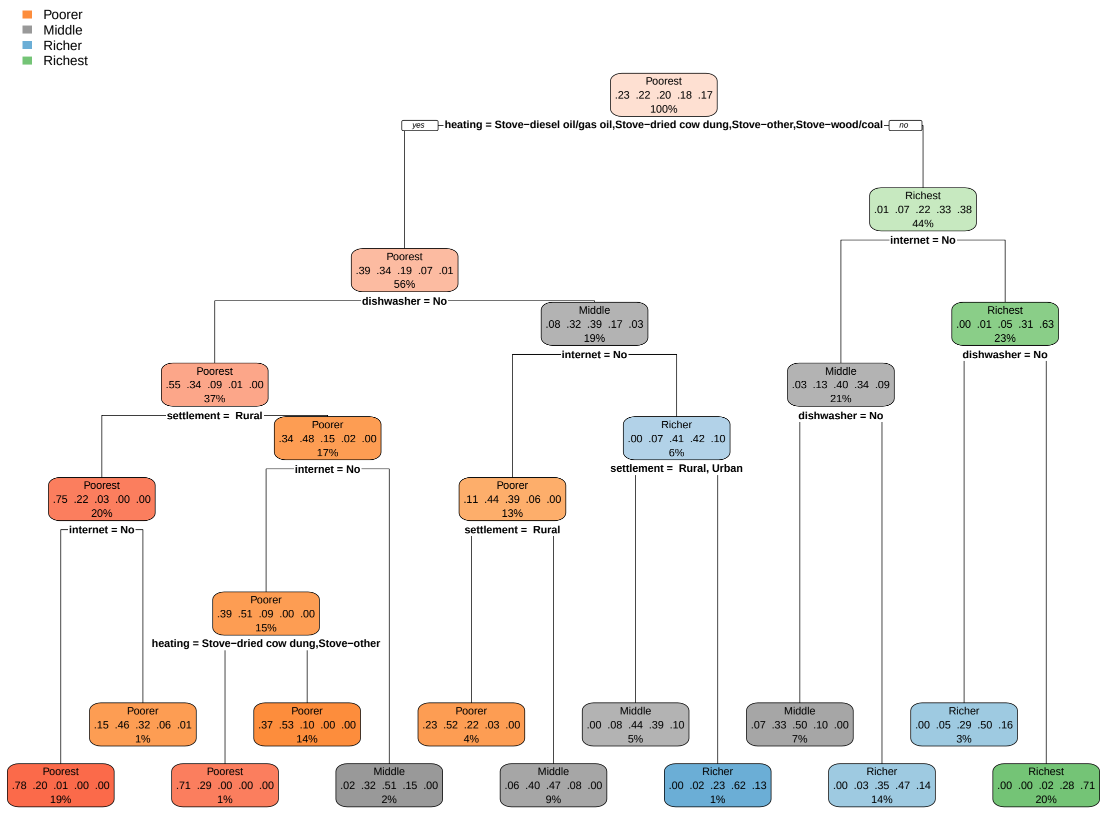
```r
decision_tree <- rpartXse(wealth_index ~ .,
                          train[,c("wealth_index", "heating", "dishwasher",
                                   "internet", "settlement")], se = 1)

predicted <- predict(decision_tree, test, type = "class")

conf_matrix <- confusionMatrix(predicted, test$wealth_index)
conf_matrix$table
```

```
##             Reference
## Prediction Poorest Poorer Middle Richer Richest
##    Poorest     366     97      6      0       0
##    Poorer      139    259     72      6       0
##    Middle       24    160    225     95       9
##    Richer        1     12    125    227      59
##    Richest       0      0      5    117     319
```

```r
conf_matrix$overall["Accuracy"]
```

```
##  Accuracy
## 0.6009471
```

```r
rpart.plot(decision_tree)
```


Poorer
Middle
Richer
Richest
Poorest .23 .22 .20 .18 .17 100%
yes
heating = Stove−diesel oil/gas oil,Stove−dried cow dung,Stove−other,Stove−wood/coal
no
Poorest .39 .34 .19 .07 .01 56%
dishwasher = No
Richest .01 .07 .22 .33 .38 44%
internet = No
Middle .08 .32 .39 .17 .03 19%
internet = No
Richest .00 .01 .05 .31 .63 23%
dishwasher = No
Poorest .55 .34 .09 .01 .00 37%
settlement = Rural
Middle .03 .13 .40 .34 .09 21%
dishwasher = No
Poorer .34 .48 .15 .02 .00 17%
internet = No
Richer .00 .07 .41 .42 .10 6%
settlement = Rural, Urban
Poorest .75 .22 .03 .00 .00 20%
internet = No
Poorer .11 .44 .39 .06 .00 13%
settlement = Rural
Poorer .39 .51 .09 .00 .00 15%
heating = Stove−dried cow dung,Stove−other
Poorer .15 .46 .32 .06 .01 1%
Poorer .37 .53 .10 .00 .00 14%
Poorer .23 .52 .22 .03 .00 4%
Middle .00 .08 .44 .39 .10 5%
Middle .07 .33 .50 .10 .00 7%
Richer .00 .05 .29 .50 .16 3%
Poorest .78 .20 .01 .00 .00 19%
Poorest .71 .29 .00 .00 .00 1%
Middle .02 .32 .51 .15 .00 2%
Middle .06 .40 .47 .08 .00 9%
Richer .00 .02 .23 .62 .13 1%
Richer .00 .03 .35 .47 .14 14%
Richest .00 .00 .02 .28 .71 20%

Asking a household member about settlement area that they are living, their heating method, whether they have dishwasher and internet connection gives an idea of the wealth index of the household by mostly misclassified only one level different. After all, a better classification model is required for more success. Even attributes that have high information gain are selected, the wealth index cannot be predicted directly by using some partition of attributes.

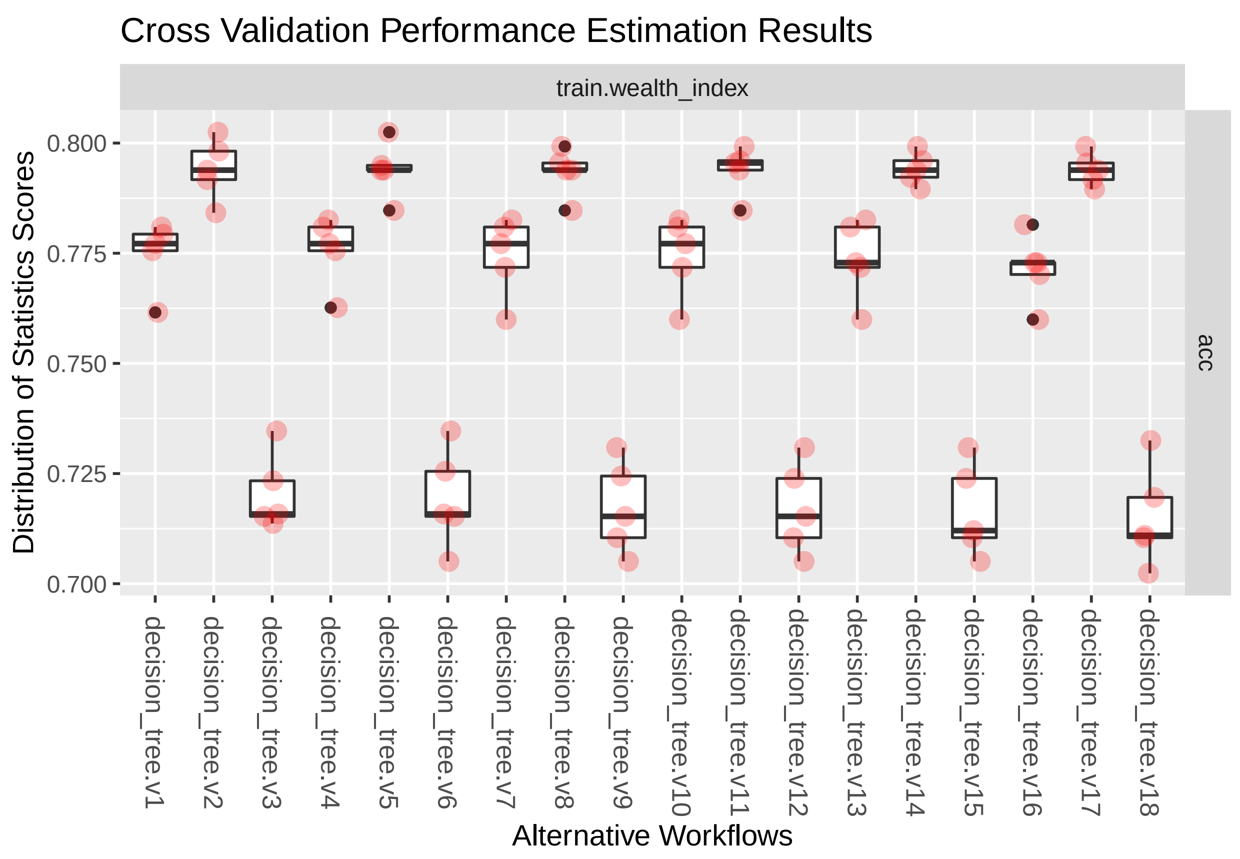
Decision Tree

The first model is the decision tree. Optimum parameters are selected by using a performance estimator. The model is trained with this parameter set and the test set is predicted by the model. Parameters are information gain threshold and prune value. Features are sorted by their information gain and thresholded by a value to reduce the number of features. Features are kept if their information gain value is greater than the threshold. Decision tree pruning is evaluated by given parameters.

Parameter set:

```r
inf_gain <- c(0.0, 0.1, 0.2)
prune_se <- c(0.0, 0.2, 0.4, 0.6, 0.8, 1.0)
```

```r
load("decisiontree.perf")  # Load pre-estimated performances
plot(perfEst)    # Plot performance estimation results
```



```r
# Best parameter set
work_flow <- getWorkflow(topPerformers(perfEst, maxs = TRUE)[[1]][1,1], perfEst)

inf_gain <- work_flow@pars[["inf_gain"]]
prune_se <- work_flow@pars[["prune_se"]]

cat("Best Parameter set:",
    "\n  inf_gain : ", inf_gain,
    "\n  prune_se : ", prune_se)
```

```
## Best Parameter set:
##   inf_gain :  0.1
##   prune_se :  0.8
```

```r
# Train model with best parameter set for testing
information_gain <- attrEval(wealth_index ~ ., train, estimator = "InfGain")
features <- names(information_gain)[information_gain>inf_gain]

model <- rpartXse(wealth_index ~ ., train[,c("wealth_index", features)], se = prune_se)

predicted <- predict(model, test, type = "class")

conf_matrix <- confusionMatrix(predicted, test$wealth_index)
conf_matrix$table
```

```
##             Reference
## Prediction Poorest Poorer Middle Richer Richest
##     Poorest     470     56       0       0        0
##     Poorer       59    408      58       2        0
##     Middle        1     62     317      93        1
##     Richer        0      2      58     300       49
##     Richest       0      0       0      50      337
```

```
results["Decision Tree"] <- conf_matrix$overall["Accuracy"]
results["Decision Tree"]
```

```
## Decision Tree
##     0.7886354
```

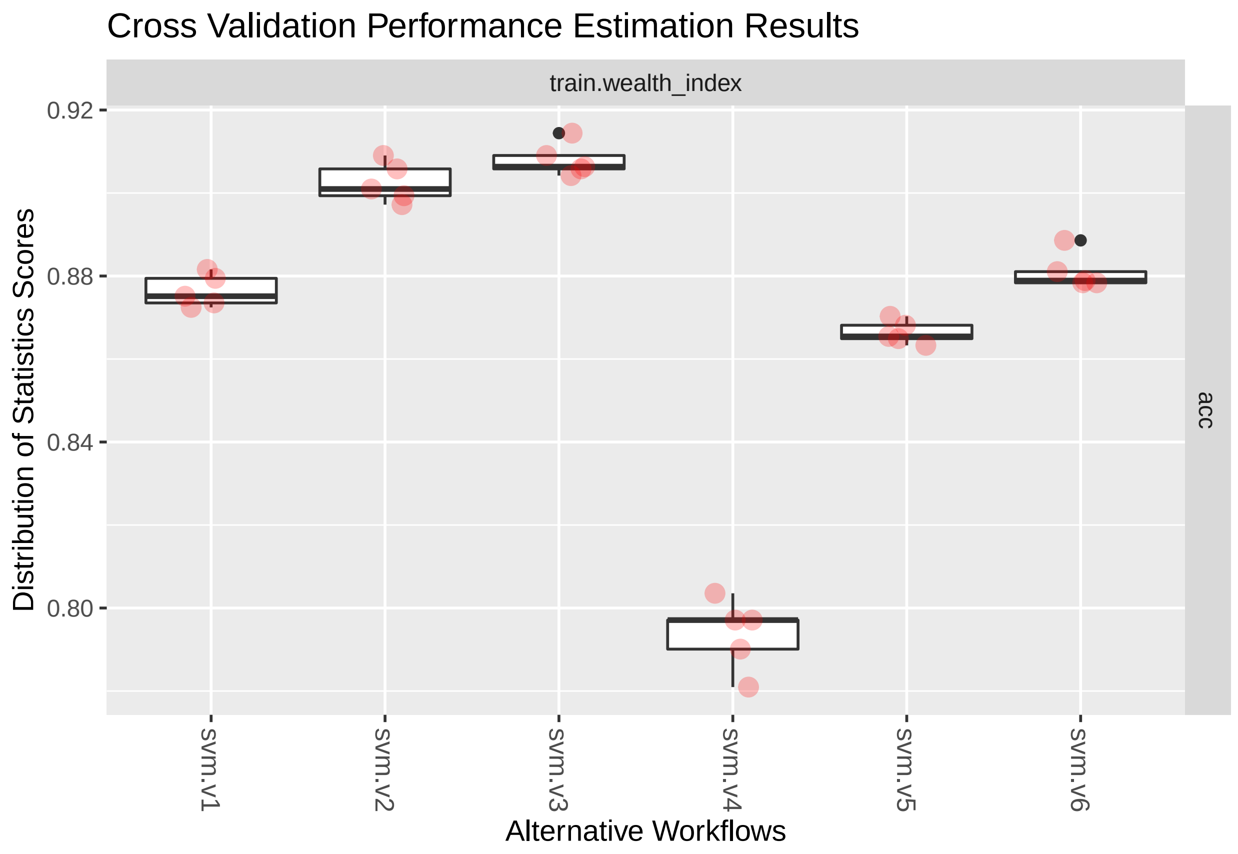
Support Vector Machine

SVM creates a decision boundary between classes. Cost and gamma parameters are evaluated by performance estimation. The test model is trained with the optimum parameter set for the classification.

Parameter set:

```
cost <- c(1, 5, 10)
gamma <- c(0.01, 0.001)
```

```
load("svm.perf")  # Load pre-estimated performances
plot(perfEst)    # Plot performance estimation results
```



```
# Best parameter set
work_flow <- getWorkflow(topPerformers(perfEst, maxs = TRUE)[[1]][1,1], perfEst)

cost <- work_flow@pars[["learner.pars"]][["cost"]]
```

```r
gamma <- work_flow@pars[["learner.pars"]][["gamma"]]

cat("Best Parameter set:",
    "\n  cost  : ", cost,
    "\n  gamma : ", gamma)
```

```
## Best Parameter set:
##   cost  :  10
##   gamma :  0.01
```

```r
# Train model with best parameter set for testing
model <- svm(wealth_index ~ ., train, cost = cost, gamma = gamma)

predicted <- predict(model, test, type = "class")

conf_matrix <- confusionMatrix(predicted, test$wealth_index)
conf_matrix$table
```

```
##           Reference
## Prediction Poorest Poorer Middle Richer Richest
##    Poorest     503     30      0      0       0
##    Poorer       27    463     20      0       0
##    Middle        0     35    390     33       0
##    Richer        0      0     23    394      11
##    Richest       0      0      0     18     376
```

```r
results["Support Vector Machine"] <- conf_matrix$overall["Accuracy"]
results["Support Vector Machine"]
```

```
## Support Vector Machine
##              0.9151959
```

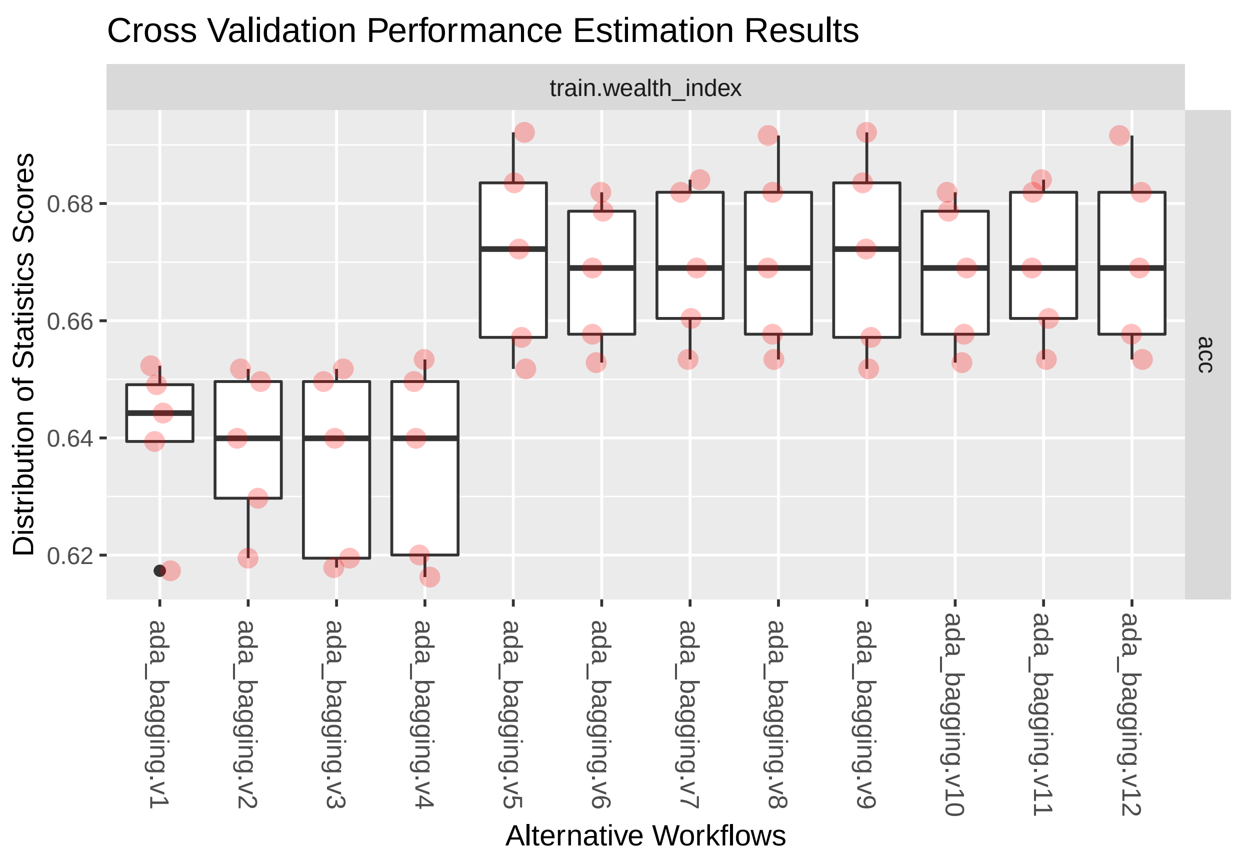
Adabag

Bootstrap Aggregation (Bagging) is an ensemble method. The number of iterations for boosting run and max depth is evaluated by using the performance estimator.

Parameter set:

```r
mfinal = c(20, 80, 120, 200)
maxdepth = c(5, 10, 20)
```

```r
load("adabag.perf")  # Load pre-estimated performances
plot(perfEst)
```



```
# Best parameter set
work_flow <- getWorkflow(topPerformers(perfEst, maxs = TRUE)[[1]][1,1], perfEst)

mfinal <- work_flow@pars[["mfinal"]]
maxdepth <- work_flow@pars[["maxdepth"]]

cat("Best Parameter set:",
    "\n  mfinal   : ", mfinal,
    "\n  maxdepth : ", maxdepth)
```

```
## Best Parameter set:
##   mfinal   :  20
##   maxdepth :  10
```

```
# Train model with best parameter set for testing
model <- bagging(wealth_index ~ ., train, mfinal = mfinal, control = rpart.control(maxdepth=maxdepth))

predicted <- predict(model, test, type = "class")

conf_matrix <- confusionMatrix(as.factor(predicted$class), test$wealth_index)
```

```
## Warning in confusionMatrix.default(as.factor(predicted$class),
## test$wealth_index): Levels are not in the same order for reference and
## data. Refactoring data to match.
```

```
conf_matrix$table
```

```
##            Reference
## Prediction Poorest Poorer Middle Richer Richest
##    Poorest     435     85      7      2       1
##    Poorer       80    307     56      3       0
##    Middle       15    123    306    133       2
##    Richer        0     13     61    251     118
##    Richest       0      0      3     56     266
```

```
results["Adabag"] <- conf_matrix$overall["Accuracy"]
results["Adabag"]
```

```
##    Adabag
## 0.6736978
```

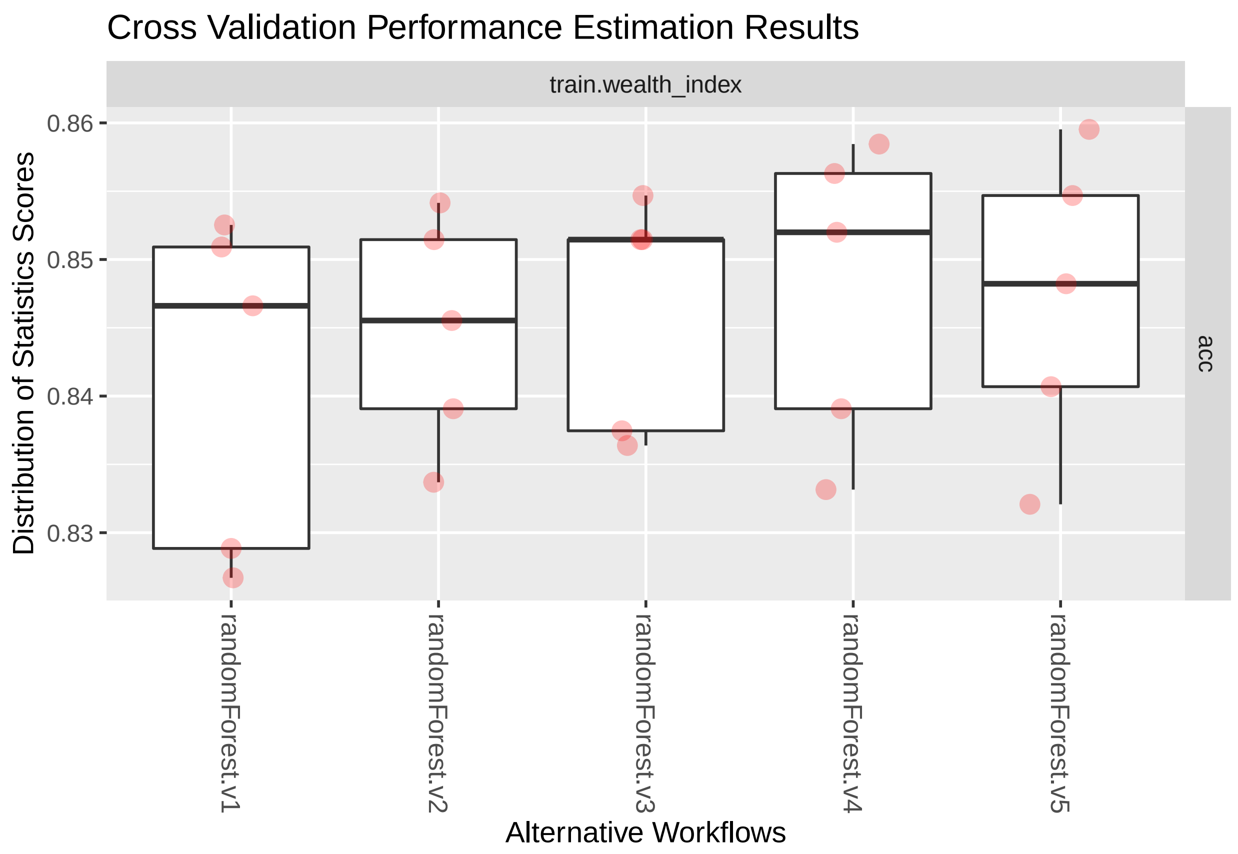
Random Forest

Random Forest is a combination of the multiple decision trees to prevent overfitting that comes with decision trees. The number of trees is evaluated by using the performance estimator.

Parameter set:

```
ntree = c(100, 200, 300, 400, 500)
```

```
load("randomforest.perf")  # Load pre-estimated performances
plot(perfEst)
```



```
# Best parameter set
work_flow <- getWorkflow(topPerformers(perfEst, maxs = TRUE)[[1]][1,1], perfEst)

ntree <- work_flow@pars[["learner.pars"]][["ntree"]]
```

```r
cat("Best Parameter set:",
    "\n  ntree  : ", ntree)
```

```
## Best Parameter set:
##   ntree  :  400
```

```r
# Train model with best parameter set for testing
model <- randomForest(wealth_index ~ ., train, ntree = ntree)

predicted <- predict(model, test, type = "class")

conf_matrix <- confusionMatrix(predicted, test$wealth_index)
conf_matrix$table
```

```
##            Reference
## Prediction Poorest Poorer Middle Richer Richest
##    Poorest     496     39      0      0       0
##    Poorer       34    447     54      1       0
##    Middle        0     42    340     81       0
##    Richer        0      0     39    337      41
##    Richest       0      0      0     26     346
```

```r
results["Random Forest"] <- conf_matrix$overall["Accuracy"]
results["Random Forest"]
```

```
## Random Forest
##     0.8463194
```

Neural Network

A neural network is designed that is suitable for the classification problem. The architecture of the neural network consists of three hidden layers. After each hidden layer dropout is applied to prevent overfitting and ReLu is used as an activation function.

```r
train_frame <- as.h2o(train, "train_frame")
test_frame <- as.h2o(test, "test_frame")

model <- h2o.deeplearning(x=names(household[, -1]),
                          y=c("wealth_index"),
                          training_frame=train_frame,
                          hidden = c(196, 256, 128),
                          hidden_dropout_ratios = c(0.4, 0.4, 0.4),
                          activation = "RectifierWithDropout",
                          epochs = 500,
                          shuffle_training_data=TRUE)

predicted <- as.data.frame(h2o.predict(model, test_frame))
```

```r
predicted$predict <- factor(predicted$predict,
                            levels = c("Poorest", "Poorer", "Middle", "Richer", "Richest"))

predicted$target <- test$wealth_index

conf_matrix <- confusionMatrix(predicted$predict, predicted$target)
conf_matrix$table
```

```
##             Reference
## Prediction Poorest Poorer Middle Richer Richest
##    Poorest     499     28      0      0       0
##    Poorer       31    478     23      0       0
##    Middle        0     22    399     42       0
##    Richer        0      0     11    383       8
##    Richest       0      0      0     20     379
```

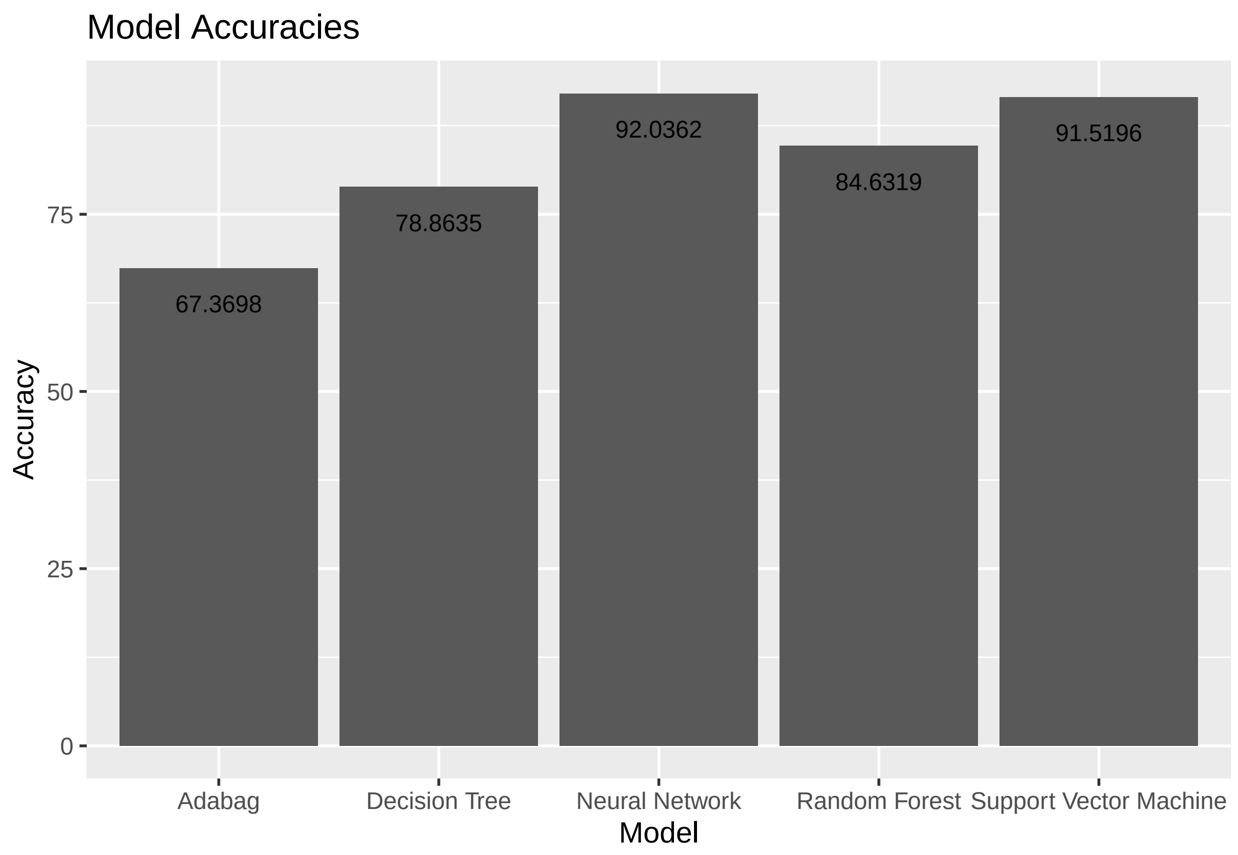
```
results["Neural Network"] <- conf_matrix$overall["Accuracy"]
results["Neural Network"]
```

```
## Neural Network
##      0.9203616
```

Results

The success of models that are trained with optimal parameter sets are given in the following bar plot:

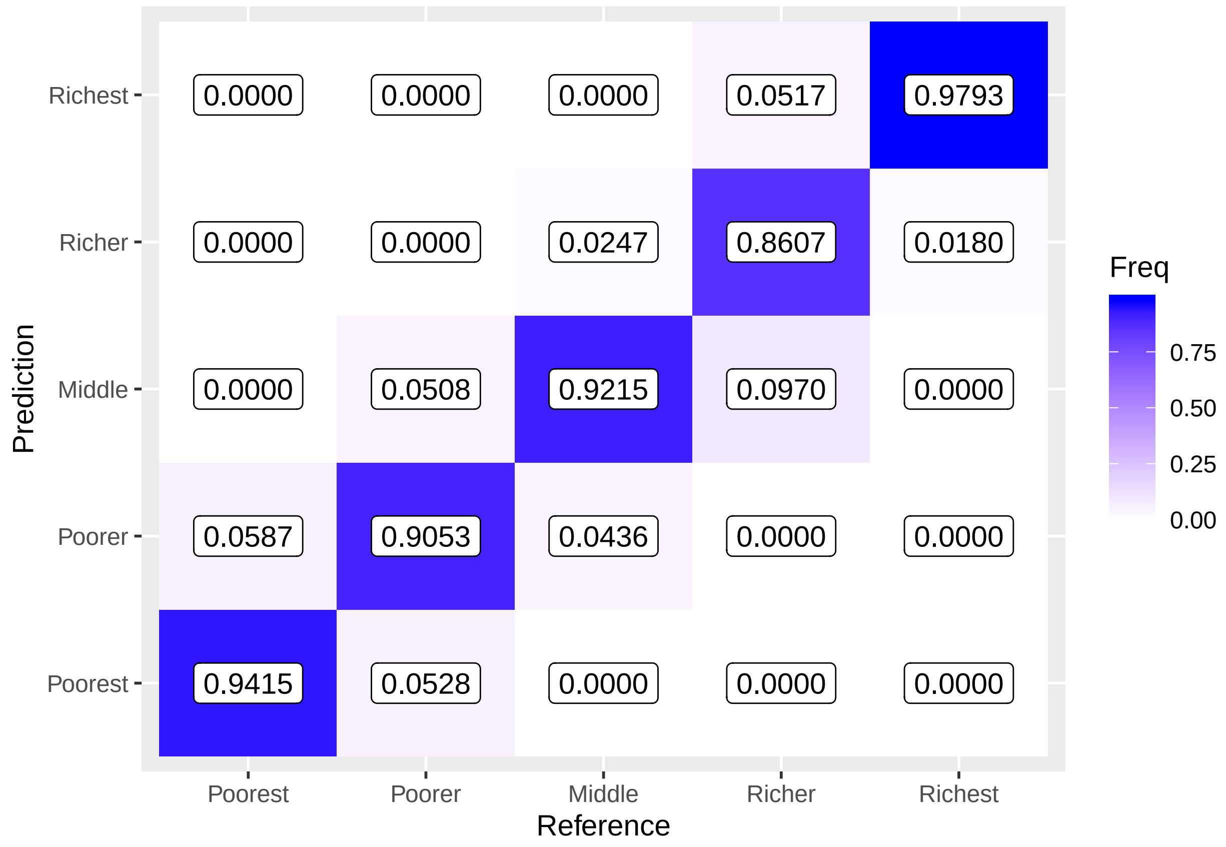
```
df = data.frame(Model = names(results),
                Accuracy = results*100)
ggplot(df, aes(x = Model, weight=Accuracy)) + geom_bar() +
        xlab("Model") + ylab("Accuracy") +
        ggtitle(label = "Model Accuracies") +
        geom_text(aes(label = sprintf("%0.4f", round(Accuracy, digits = 4)),
                      y = Accuracy-5), size = 3)
```



According to results, Adabag achieved the worst score, decision tree, and random forest got more success than bagging. SVM and neural network models achieved a similar score on classification TNSA household data for the wealth index. The neural network model is the best in evaluated ones. Besides, misclassification is done mostly in favor of the close class such as richer to richest.

The normalized confusion matrix of the neural network that is the best model is given at the following plot:

```r
normalized_conf_matrix <- conf_matrix$table / colSums(conf_matrix$table)
normalized_conf_matrix <- as.data.frame(normalized_conf_matrix)

ggplot(data =  normalized_conf_matrix, mapping = aes(x = Reference, y = Prediction)) +
       xlab("Reference") + ylab("Prediction") +
       geom_tile(aes(fill = Freq)) +
       geom_label(aes(label = sprintf("%0.4f", Freq)), vjust = 0.5) +
       scale_fill_gradient(low = "white",
                           high = "blue")
```



According to the confusion matrix, the neural network model is more successful in predicting the extreme classes than the predicting to wealth indexes in the middle.