KDD Dataset Practical Assignment DDS

Vincent Buijtendijk

2023-06-12

Introduction

The goals I have set myself for this assignment are the following

- Learn basics of R programming
- Improve similarity to real world by
  - Using a different initial dataset
  - Use KFold cross-validation
- Test different models on the dataset and compare their accuracy
- Optimize number of trees used in Random Forest to reduce computational strain

REFERENCES:

List of references:

- Source data KDD Cup 1999 Data Data Set
- A Detailed Analysis of the KDD CUP 99 Data Set
- KDD download archive
- Kaggle comunity notebooks with KDD CUP 99 data set.

Based on the above analysis in the second reference we download the NSL-KDD dataset that offers a more realistic and improved Train and Test set (removed duplicates, removed erroneous lines, a more representative selection of records) from https://www.kaggle.com/datasets/hassan06/nslkdd?resource=download. We will load the KDDTestPlus.csv and KDDTrainPlus.csv datasets. Column names have been added, same names as Book1.csv and Book2.csv to avoid compatibility issues. The aim of this is to improve measuring the effectiveness of each model.

We have enabled parallel computing in the setup section to improve speed as some models, such as Random Forest, are computationally very intensive.

The following section sets up the data in different folds, which is a technique used in cross-validation this is a method for assessing the performance of a machine learning model. Cross-validation helps to estimate how well a model will generalize to unseen data by evaluating its performance on different subsets of the training data. It provides a better estimation of accuracy of each model.

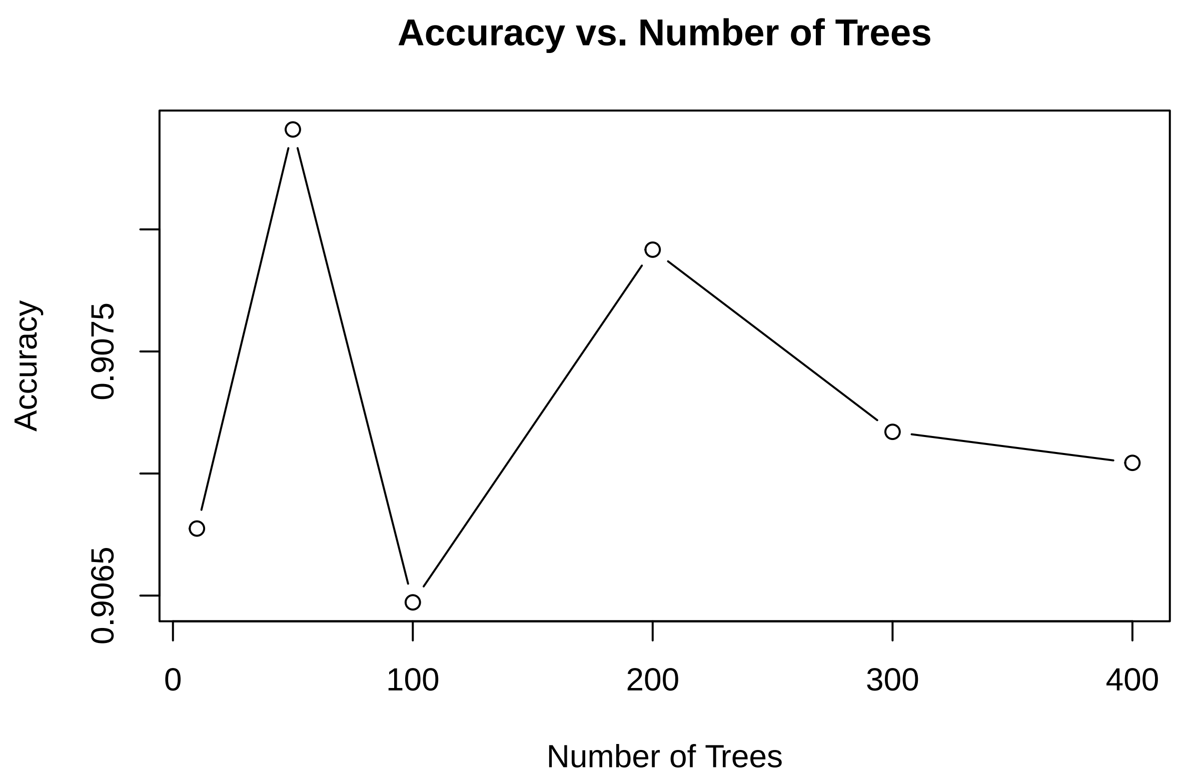
```r
# Set the seed for reproducibility
set.seed(123)

#Set the number of folds
k <- 10
```

```r
# Create a cross-validation object
cv <- trainControl(method = "cv", number = k)
```

In the following section we are trying to determine the optimum number of trees to be used in RandomForest. As this takes very long to execute (10 folds) and causes memory problems on higher numbers of trees we added a switch to run or not. If set to TRUE it will run, if not it will present earlier calculated results for the purpose of this document.

```
## The code in this chunk is not being executed, however presenting earlier calculated data below:
```



```
## Number of Trees Accuracy
## 10              0.9067746
## 50              0.9084092
## 100              0.9064723
## 200              0.9079169
## 300              0.9071708
## 400              0.9070437
```

Apply the LRM model

```r
# Fit the Logistic Regression model using cross-validation
logistic_regression_model <- train(AttackBoolean ~ ., data = dataLRM, method = "glm", family = "binomial
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

Apply the Decision Tree Model

```
# Fit the Decision Tree model using cross-validation
decision_tree_model <- train(Attack ~ ., data = data, method = "rpart", trControl = cv)
```

Apply Randomforest model. Based on earlier calculations the optimal number of trees is 50, so this will be applied here.

```
# Create a cross-validation object with verbose output
cv <- trainControl(method = "cv", number = k, verboseIter = TRUE)

# Fit the Random Forest model using cross-validation and verbose output
random_forest_model <- train(Attack ~ ., data = data, method = "rf", trControl = cv, tuneGrid = data.fra
```

```
## Aggregating results
## Fitting final model on full training set
```

The following section prints the results of each of the models.

```
## Logistic Regression Model:

## Generalized Linear Model
##
## 125973 samples
##     10 predictor
##      2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 113375, 113376, 113376, 113376, 113376, 113376, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9087662  0.8160952

## Accuracy: 0.9087662

## Logistic Regression Precision: 0.8984037

## Logistic Regression Recall: 0.9351974

## Logistic Regression F1-score: 0.9164314

## Decision Tree Model:

## CART
##
## 125973 samples
##     10 predictor
##     23 classes: 'back', 'buffer_overflow', 'ftp_write', 'guess_passwd', 'imap', 'ipsweep', 'land', 'l
##
```

```
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 113376, 113378, 113375, 113378, 113375, 113376, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.01720109  0.8717096  0.7659366
##   0.03003582  0.8477057  0.7167895
##   0.65531298  0.7477454  0.4915878
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.01720109.
```

```
## Accuracy: 0.8717096
```

```
## Decision Tree Precision: 0.8746794
```

```
## Decision Tree Recall: 0.08587519
```

```
## Decision Tree F1-score: 0.9259309
```

```
## Random Forest Model:
```

```
## Random Forest
##
## 125973 samples
##     10 predictor
##     23 classes: 'back', 'buffer_overflow', 'ftp_write', 'guess_passwd', 'imap', 'ipsweep', 'land', 'l
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 113378, 113373, 113375, 113375, 113376, 113374, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.9052173  0.8319871
##
## Tuning parameter 'mtry' was held constant at a value of 2
```

```
## Accuracy: 0.9052173
```

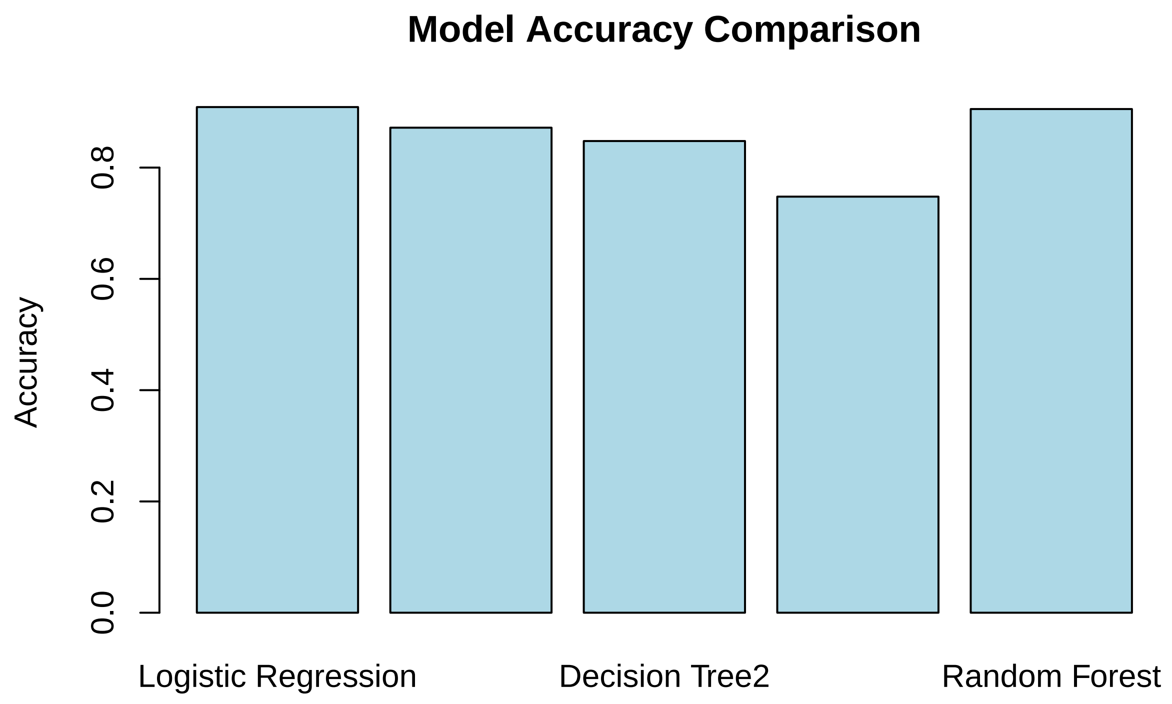
```
## Random Forest Precision: 0.9482311
```

```
## Random Forest Recall: 0.1662249
```

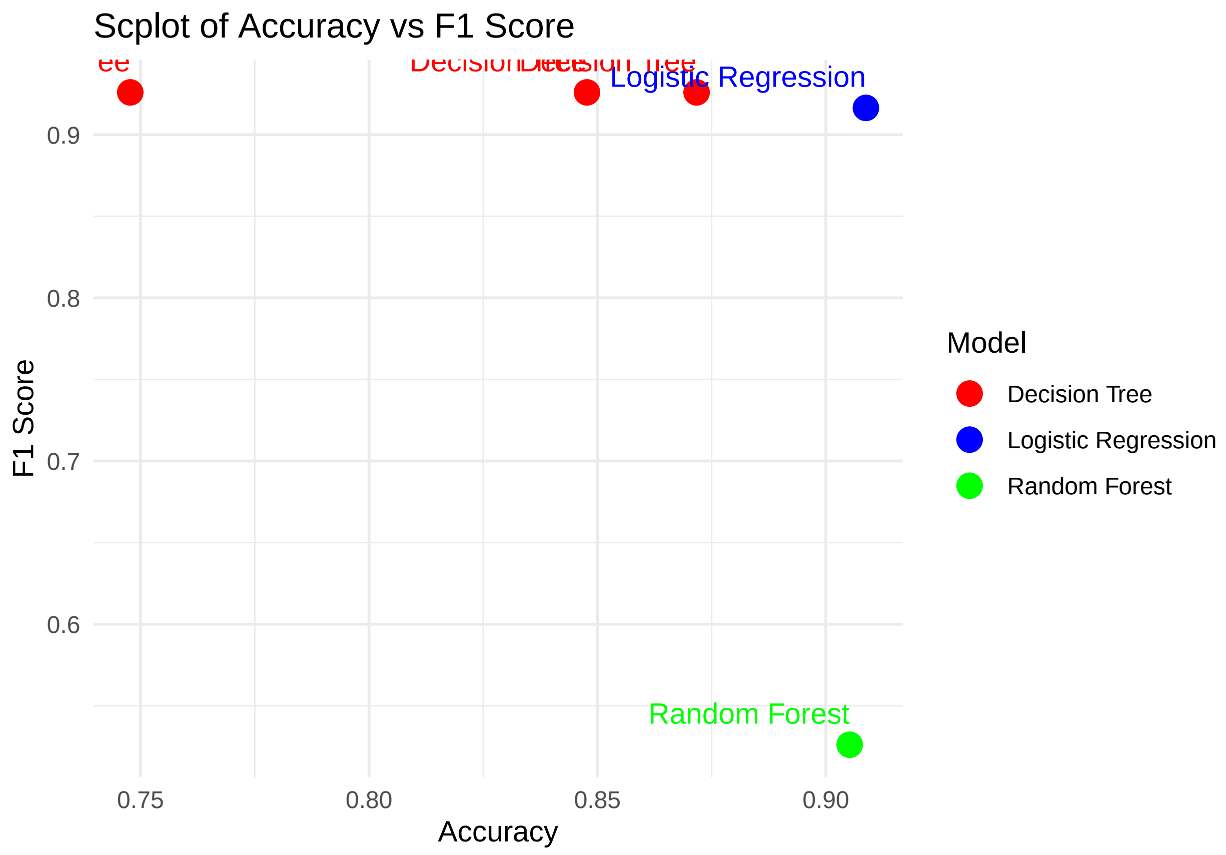
```
## Random Forest F1-score: 0.5257677
```

Now determine the model with the best score based on the data set supplied:

```
## Best Model: Logistic Regression
## Best Model Accuracy: 0.9087662
```



Create a scatterplot with the two most important scores on X and Y axis, respectively accuracy and f1 score, to compare the models.



Conclusions

We compared 3 different models. Looking at both the accuracy and F1 scores it looks as though LRM is a good model both for accuracy and F1 score on this dataset, and computationally relatively less expensive. Some points remain open for future such as determining if more Trees in RandomForest lead to better F1 results. Also other models could be added and tweak more parameters in current models.