ML-Assignment-3 - Gradient Boosting

PrabhTalwar

2023-01-09

```
library(dplyr) # for general data wrangling needs
# Modeling packages
library(rsample)      # data splitting
library(caret)
library(recipes)
library(gbm) # original implementation of regular & stochastic GBMs
library(xgboost) # for fitting extreme gradient boosting
```

```
iris_data <- iris
summary(iris_data)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

```
dim(iris_data)
```

```
## [1] 150   5
```

```
set.seed(123)
iris_split <- initial_split(iris_data, prop = 0.7)
iris_train <- training(iris_split)
iris_test  <- testing(iris_split)
```

Gradient Boosting

```
set.seed(123) # for reproducibility
gb_model <- gbm(Species ~., data = iris_train,
                n.trees = 1000,
                shrinkage = 0.01,
                interaction.depth = 3,
                n.minobsinnode = 10,
                cv.folds = 10)
```

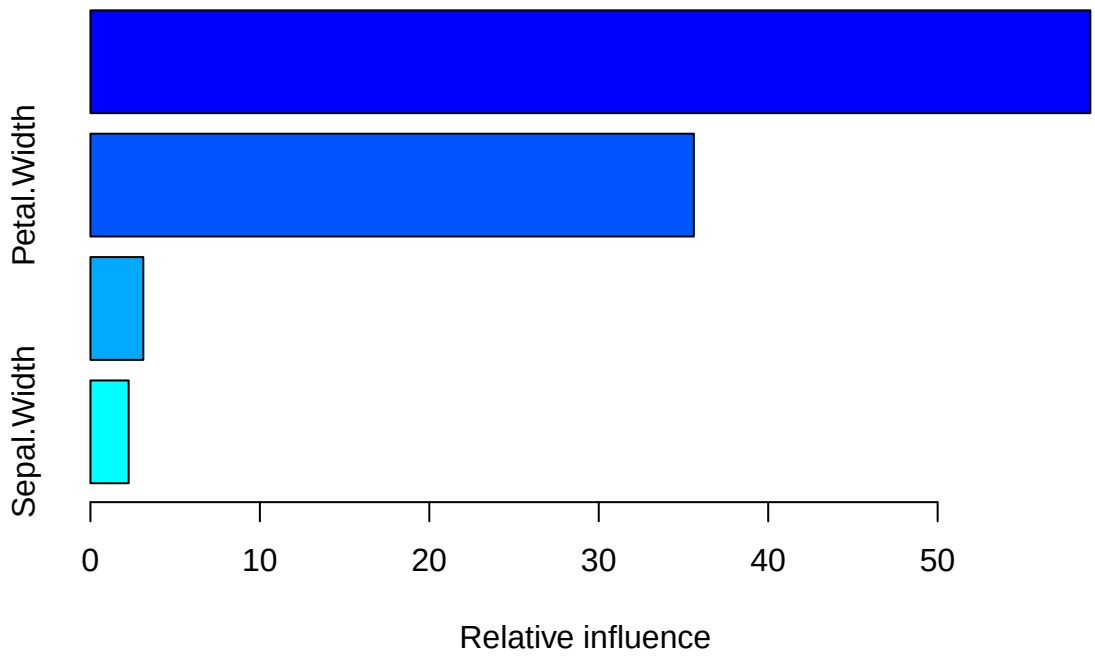
```
## Distribution not specified, assuming multinomial ...

## Warning: Setting ‘distribution = "multinomial"‘ is ill-advised as it is
## currently broken. It exists only for backwards compatibility. Use at your own
## risk.
```

```
print(gb_model)
```

```
## gbm(formula = Species ~ ., data = iris_train, n.trees = 1000,
##     interaction.depth = 3, n.minobsinnode = 10, shrinkage = 0.01,
##     cv.folds = 10)
## A gradient boosted model with multinomial loss function.
## 1000 iterations were performed.
## The best cross-validation iteration was 238.
## There were 4 predictors of which 4 had non-zero influence.
```

```
summary(gb_model)
```



```
##                        var   rel.inf
## Petal.Length Petal.Length 59.012405
## Petal.Width   Petal.Width 35.611081
## Sepal.Length Sepal.Length  3.120293
## Sepal.Width   Sepal.Width  2.256221
```

we can see that Petal.Length and Petal.Width are by far the most important variables in our gbm model.

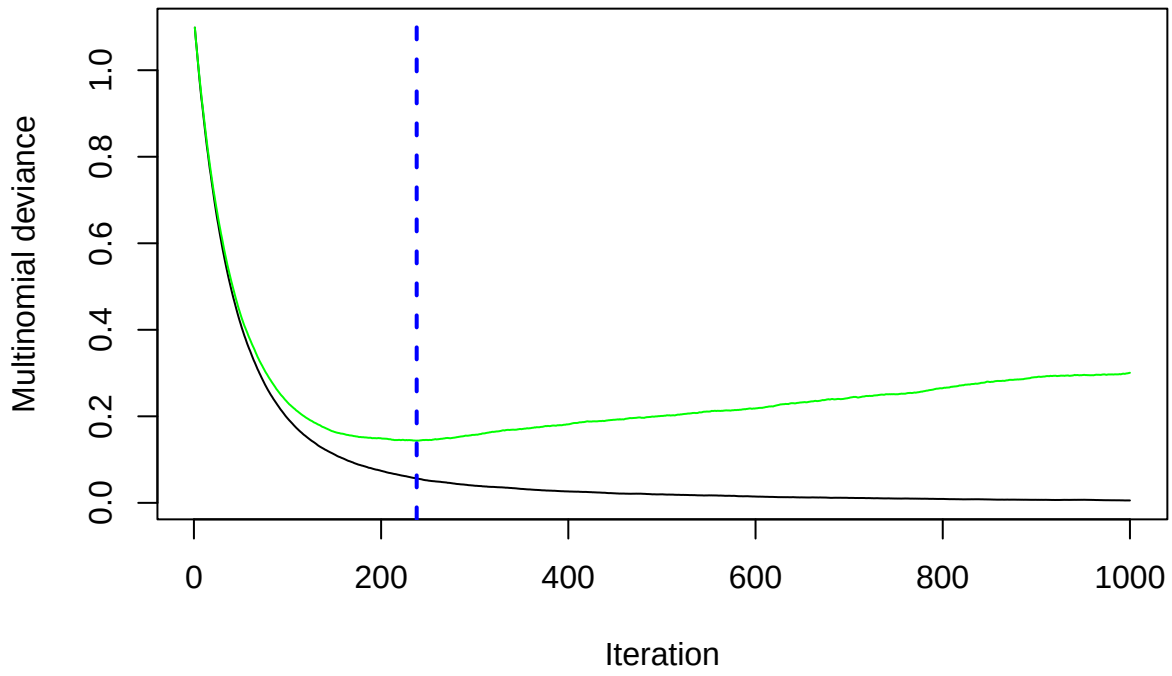
```
# find index for number trees with minimum CV error
best <- which.min(gb_model$cv.error)
best
```

```
## [1] 238
```

```
# get MSE and compute RMSE
sqrt(gb_model$cv.error[best])
```

```
## [1] 0.3791915
```

```
# plot error curve
gbm.perf(gb_model, method = "cv")
```




```
## [1] 238
```

Our results show a cross-validated SSE of 0.3770093 which was achieved with 238 trees.

```
pred_test = predict.gbm(object = gb_model,
                  newdata = iris_test,
                  n.trees = 1000,
                  type = "response")
```

```
# Give class names to the highest prediction value.
class_names = colnames(pred_test)[apply(pred_test, 1, which.max)]
result = data.frame(iris_test$Species, class_names)
```

```
confusion_matrix <- confusionMatrix(iris_test$Species, as.factor(class_names))
print(confusion_matrix)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         14          0         0
##   versicolor      0         17         1
##   virginica       0          0        13
##
## Overall Statistics
##
##                Accuracy : 0.9778
##                  95% CI : (0.8823, 0.9994)
##     No Information Rate : 0.3778
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9664
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            1.0000           0.9286
## Specificity                 1.0000            0.9643           1.0000
## Pos Pred Value              1.0000            0.9444           1.0000
## Neg Pred Value              1.0000            1.0000           0.9688
## Prevalence                  0.3111            0.3778           0.3111
## Detection Rate              0.3111            0.3778           0.2889
## Detection Prevalence        0.3111            0.4000           0.2889
## Balanced Accuracy           1.0000            0.9821           0.9643
```

XGBoost

```
xgb_prep <- recipe(Species ~ ., data = iris_train) %>%
 step_integer(all_nominal()) %>%
 prep(training = iris_train, retain = TRUE) %>%
```

```
 juice()
X <- as.matrix(xgb_prep[setdiff(names(xgb_prep), "Species")])
Y <- xgb_prep$Species
```

```
set.seed(123)
ames_xgb <- xgb.cv(data = X, label = Y,
                   nrounds = 1000,
                   n.minobsinnode = 10,
                   early_stopping_rounds = 50,
                   nfold = 10,
                   params = list(eta = 0.01,
                                 max_depth = 3,
                                 min_child_weight = 3,
                                 subsample = 0.5,
                                 colsample_bytree = 1.0),
                   verbose = 0)

# minimum test CV RMSE
min(ames_xgb$evaluation_log$test_rmse_mean)
```

```
## [1] 0.1857639
```