Practical Machine Learning - Week4

Saul Lugo

January 31, 2016

Reguralized Regression

Model Selection Approach: split samples

1. Divide data into training/test/validation sets
2. Treat validation as a test data, train all competing models on the train data and pick the best one on validation.
3. To appropriately assess performance on new data apply the model to test set
4. You might re-split and reperform steps 1-3

Combining Predictors

The idea is to combine different predictor models in order to improve the accuracy. The following example combines a GLM with a Random Forest predictor:

```
library(ISLR); data(Wage); library(ggplot2); library(caret);
Wage <- subset(Wage, select=-c(logwage))

set.seed(1234)
#Splitting the data into training, testing and validation sets
inBuild <- createDataPartition(y=Wage$wage,p=0.7,list=FALSE)
validation <- Wage[-inBuild,]; buildData <- Wage[inBuild,]
inTrain <- createDataPartition(y=buildData$wage,p=0.7,list=FALSE)
training <- buildData[inTrain,]; testing <- buildData[-inTrain,]

dim(training); dim(testing); dim(validation);
```

```
## [1] 1474   11
```

```
## [1] 628  11
```

```
## [1] 898  11
```

```
#Now, fit the two models
mod1 <- train(wage ~.,method="glm",data=training)
mod2 <- train(wage ~.,method="rf",data=training,trControl=trainControl(method="cv",number=3))

mod1
```

```
## Generalized Linear Model
##
## 1474 samples
##   10 predictor
```

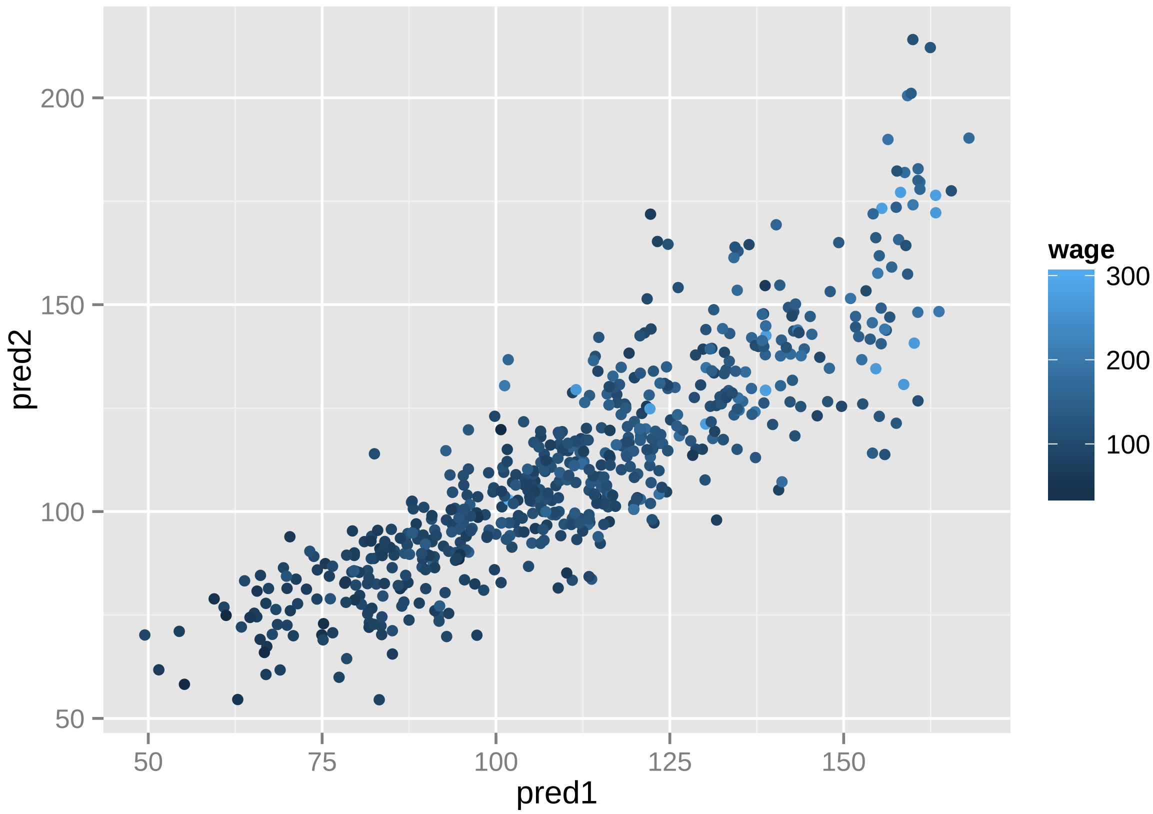
```
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 1474, 1474, 1474, 1474, 1474, 1474, ...
## Resampling results
##
##   RMSE     Rsquared   RMSE SD   Rsquared SD
##   35.0092  0.3204793  1.762416  0.02920338
##
##
```

```r
mod2
```

```
## Random Forest
##
## 1474 samples
##   10 predictor
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 982, 983, 983
## Resampling results across tuning parameters:
##
##   mtry  RMSE      Rsquared   RMSE SD   Rsquared SD
##    2    38.16546  0.3123716  3.278744  0.02009053
##   13    37.05130  0.2644913  1.636132  0.03221683
##   25    38.47024  0.2321406  1.272379  0.03441396
##
## RMSE was used to select the optimal model using  the smallest value.
## The final value used for the model was mtry = 13.
```

```r
#plotting mod1 vs mod2

pred1 <- predict(mod1,testing); pred2 <- predict(mod2,testing);
qplot(pred1,pred2,colour=wage,data=testing)
```




```r
#Fit a model that combines both predictors
predDF <- data.frame(pred1,pred2,wage=testing$wage)
combModFit <- train(wage ~.,method="gam",data=predDF)
combPred <- predict(combModFit,predDF)
combModFit
```

```
## Generalized Additive Model using Splines
##
## 628 samples
##   2 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 628, 628, 628, 628, 628, 628, ...
## Resampling results across tuning parameters:
##
##   select  RMSE      Rsquared   RMSE SD   Rsquared SD
##   FALSE   33.64839  0.3406428  2.142237  0.04548650
##    TRUE   33.68495  0.3392327  2.145042  0.04566066
##
## Tuning parameter 'method' was held constant at a value of GCV.Cp
## RMSE was used to select the optimal model using  the smallest value.
## The final values used for the model were select = FALSE and method
##  = GCV.Cp.
```

```r
#Comparing the Root Squared Errors between mod1, mod2 and the combined model
sqrt(sum(pred1-testing$wage)^2)
```

```
## [1] 566.8475
```

```r
sqrt(sum(pred2-testing$wage)^2)
```

```
## [1] 528.6399
```

```r
sqrt(sum(combPred-testing$wage)^2)
```

```
## [1] 1.336176e-10
```

```r
#Checking the model on the validation set
pred1V <- predict(mod1,validation); pred2V <- predict(mod2,validation)
predVDF <- data.frame(pred1=pred1V,pred2=pred2V)
combPredV <- predict(combModFit,predVDF)

#Checking the RSE on the validation set
sqrt(sum(pred1V-validation$wage)^2)
```

```
## [1] 342.1366
```

```r
sqrt(sum(pred2V-validation$wage)^2)
```

```
## [1] 285.9747
```

```r
sqrt(sum(combPredV-validation$wage)^2)
```

```
## [1] 656.5295
```

Unsupervised Prediction

When performing unsupervised prediction one does not know the labels of the outcome beforehand.

The following example use clustering technique on the IRIS dataset. We ignore the Species variable in order to simulate that we don't know the outcome. We use clustering by **k-means** in order to build the clusters, then we build a prediciton model using the clusters in the training dataset as the outcome variable:

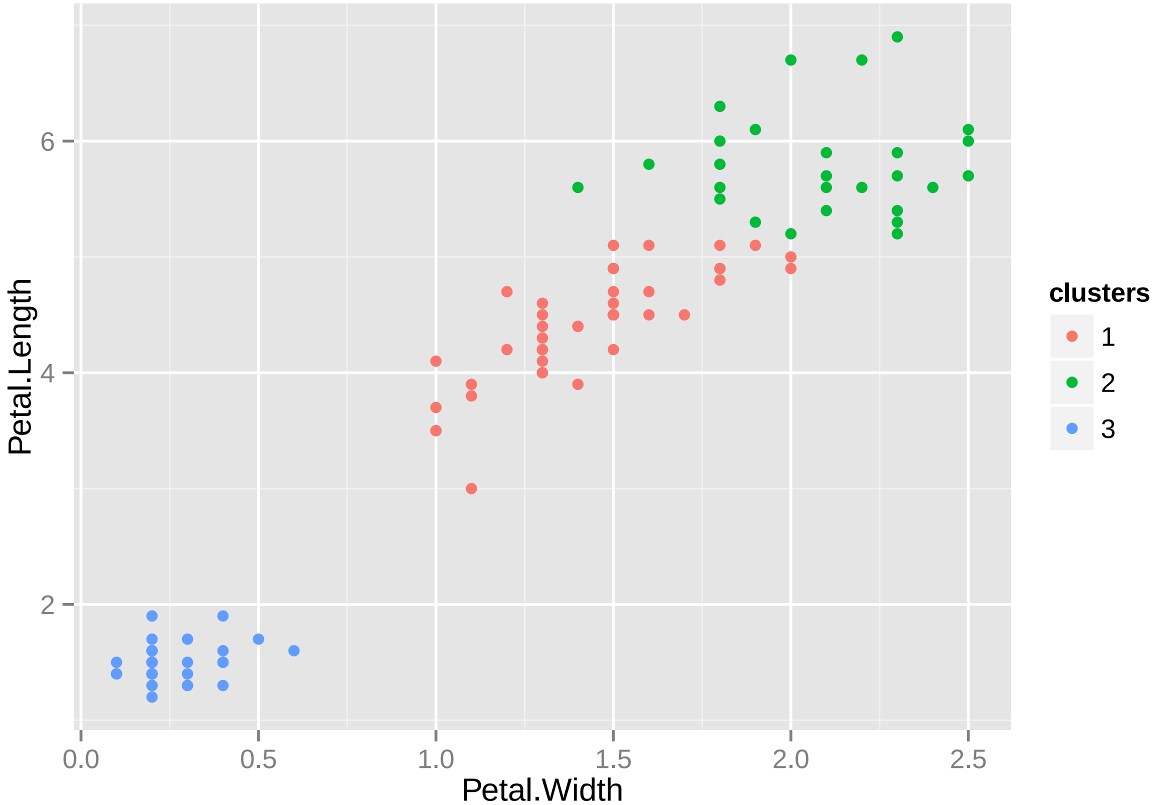
```r
data(iris)
library(ggplot2)
library(caret)

inTrain <- createDataPartition(y=iris$Species,p=.7,list=FALSE)
training <- iris[inTrain,]
testing <- iris[-inTrain,]
dim(training); dim(testing)
```

```
## [1] 105   5

## [1] 45  5
```

```r
#Building the clusters
kMeans1 <- kmeans(subset(training,select=-c(Species)),centers=3)
training$clusters <- as.factor(kMeans1$cluster)
qplot(Petal.Width,Petal.Length,colour=clusters,data=training)
```




```r
table(kMeans1$cluster,training$Species)
```

```
##
##     setosa versicolor virginica
##   1      0         34         8
##   2      0          1        27
##   3     35          0         0
```

```r
#Fitting the model using the clusters
modFit <- train(clusters ~.,data=subset(training,select=-c(Species)),method="rpart")
table(predict(modFit,training),training$Species)
```

```
##
##     setosa versicolor virginica
##   1      0         35         8
##   2      0          0        27
##   3     35          0         0
```

```r
#Apply the model on the test set
testClusterPred <- predict(modFit,testing)
table(testClusterPred,testing$Species)
```

```
##
## testClusterPred setosa versicolor virginica
##               1      0         15         8
##               2      0          0         7
##               3     15          0         0
```