Caret-ML_R

```
library(tidyverse)
library(caret)
library(mlbench)

## load data
data("PimaIndiansDiabetes")

df <- PimaIndiansDiabetes
# check null
mean(complete.cases(df))
```

```
> # check null
> mean(complete.cases(df))
[1] 1
> head(df,10)
   pregnant glucose pressure triceps insulin mass pedigree age diabetes
1         6     148       72      35       0 33.6    0.627  50      pos
2         1      85       66      29       0 26.6    0.351  31      neg
3         8     183       64       0       0 23.3    0.672  32      pos
4         1      89       66      23      94 28.1    0.167  21      neg
5         0     137       40      35     168 43.1    2.288  33      pos
6         5     116       74       0       0 25.6    0.201  30      neg
7         3      78       50      32      88 31.0    0.248  26      pos
8        10     115        0       0       0 35.3    0.134  29      neg
9         2     197       70      45     543 30.5    0.158  53      pos
10        8     125       96       0       0  0.0    0.232  54      pos
>
```

```
## train model rpart
## recursive partitioning (decision tree)
ctrl <- trainControl(
  method = "cv",
  number = 5,
  verboseIter = TRUE,
  classProbs = TRUE, # we can change threshold 0.5
  summaryFunction = twoClassSummary
)
```

```
tree_model <- train(
  diabetes ~ glucose + pressure + insulin + mass + age,
  data = df,
  method = "rpart",
  metric = "ROC",
  trControl = ctrl
)
```

```
> tree_model
CART

768 samples
  5 predictor
  2 classes: 'neg', 'pos'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 614, 614, 615, 615, 614
Resampling results across tuning parameters:

  cp          ROC        Sens   Spec
  0.01741294  0.7558354  0.828  0.6153739
  0.10447761  0.6856003  0.856  0.5043326
  0.24253731  0.6344493  0.874  0.3948987

ROC was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.01741294.
```

```
## prediction
predict(tree_model, df, type = "prob")[1:10, ]
```

```
> ## prediction
> predict(tree_model, df, type = "prob")[1:10, ]
         neg       pos
1  0.2753623 0.7246377
2  0.8061856 0.1938144
3  0.6842105 0.3157895
4  0.8061856 0.1938144
5  0.2753623 0.7246377
6  0.8061856 0.1938144
7  0.8061856 0.1938144
8  0.8061856 0.1938144
9  0.2753623 0.7246377
10 0.8061856 0.1938144
>
```

```
## prediction
predict(tree_model, df, type = "prob")[1:10, ]

## change threshold
probs <- predict(tree_model, df, type = "prob")

p_class <- ifelse(probs$pos >= 0.5, "pos", "neg")
```

```
> head(probs,10)
         neg       pos
1  0.2753623 0.7246377
2  0.8061856 0.1938144
3  0.6842105 0.3157895
4  0.8061856 0.1938144
5  0.2753623 0.7246377
6  0.8061856 0.1938144
7  0.8061856 0.1938144
8  0.8061856 0.1938144
9  0.2753623 0.7246377
10 0.8061856 0.1938144
> head(p_class,10)
 [1] "pos" "neg" "neg" "neg" "pos" "neg" "neg" "neg" "pos" "neg"
>
```

```
## confusion matrix for cal Accuracy,Precision,Recall,F1
table(df$diabetes, p_class)
```

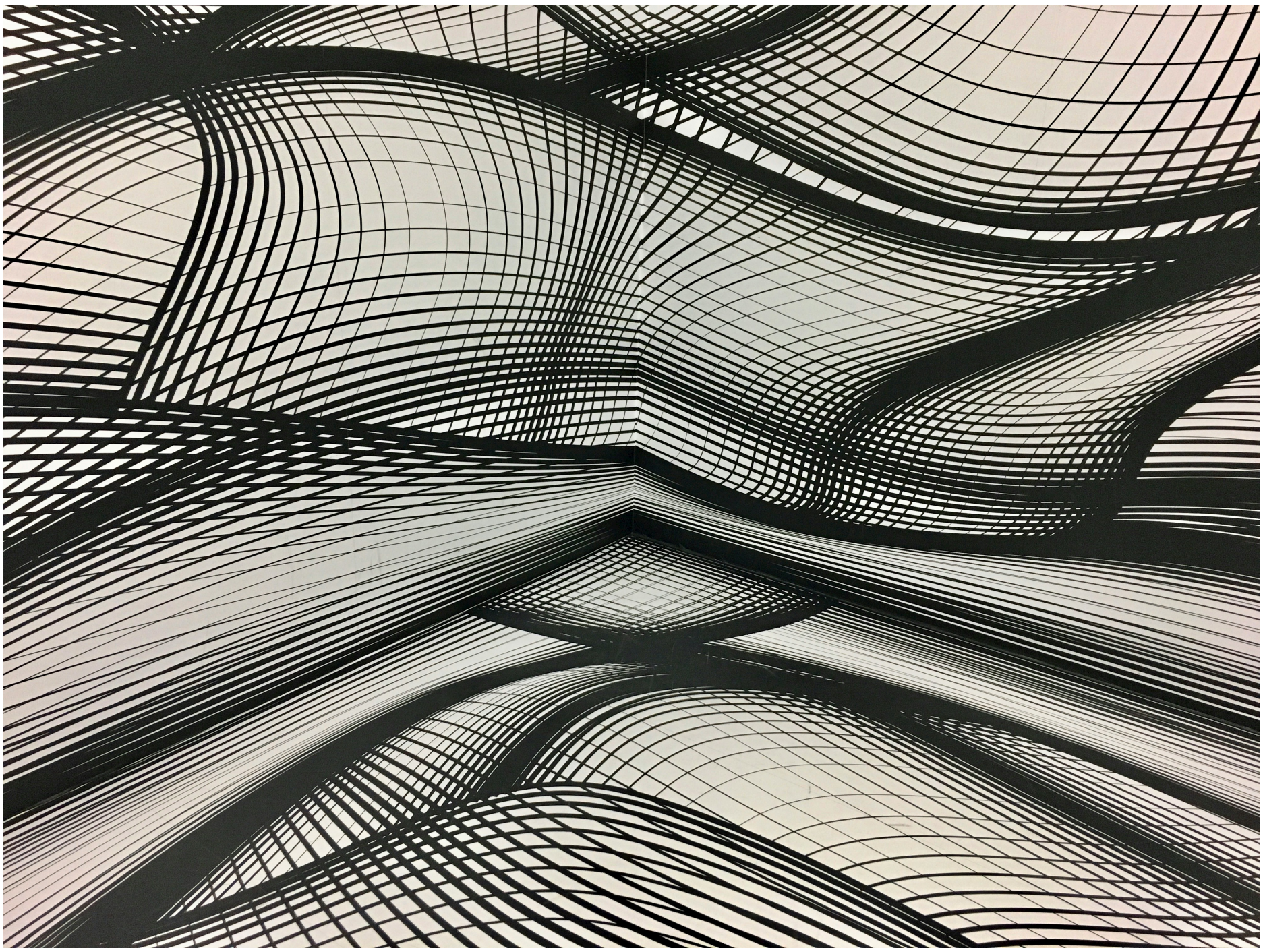
```
> table(df$diabetes, p_class)
     p_class
      neg pos
  neg 443  57
  pos 118 150
>
```

```
## random forest (bagging)  Random Forest helps reduce overfitting
## RPART
```

```
ctrl <- trainControl(
  method = "cv",
  number = 5,
  verboseIter = TRUE
)
## Random Forest
rf_model <- train(
  diabetes ~ .,
  data = df,
  method = "rf",
  metric = "Accuracy",
  tuneGrid = data.frame(mtry = c(2,3,4)),
  trControl = ctrl
)
```

```
> rf_model
Random Forest

768 samples
  8 predictor
  2 classes: 'neg', 'pos'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 614, 614, 615, 615, 614
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
  2     0.7616926  0.4572797
  3     0.7655802  0.4662425
  4     0.7591036  0.4561094

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 3.
>
```

RF > Decision Tree



```
## Ridge vs. Lasso Regression
## Regularization is a technique to reduce overfitting
## Ridge(L2) => beta will be lower, but not zero
## Lasso(L1) => beta can be zero (feature selection)


## ElasticNet = Mixed betweem Ridge+ Lasso
##alpha=1 Lasso , alpha=0 Ridge
glmnet_model <- train(
  diabetes ~ .,
  data = df,
  method = "glmnet",
  metric = "Accuracy",
  tuneGrid = expand.grid(
    alpha = 0:1,
    lambda = c(0.004, 0.04, 0.08)
  ),
  trControl = ctrl
)

## save model
##saveRDS(glmnet_model, "ridge_lasso_reg.RDS")
## call to use
##model <- readRDS("ridge_lasso_reg.RDS")
```

```
> glmnet_model
glmnet

768 samples
  8 predictor
  2 classes: 'neg', 'pos'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 614, 615, 614, 615, 614
Resampling results across tuning parameters:

  alpha  lambda  Accuracy   Kappa
  0      0.004   0.7734063  0.4712504
  0      0.040   0.7694848  0.4586636
  0      0.080   0.7720567  0.4571163
  1      0.004   0.7746796  0.4755669
  1      0.040   0.7668449  0.4448035
  1      0.080   0.7499533  0.3782315

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were alpha = 1 and lambda = 0.004.
>
```

##Choosing Lasso

Ridge Regression (L2)

$$RSS = \sum (\hat{y} - y)^2$$

Normal RSS from Linear Regression

$$Ridge\ RSS = \sum (\hat{y} - y)^2 + \lambda \sum \beta^2$$

Ridge add this term to the error function

Lasso Regression (L1)

$$RSS = \sum (\hat{y} - y)^2$$

Normal RSS from Linear Regression

$$Lasso\ RSS = \sum (\hat{y} - y)^2 + \lambda \sum |\beta|$$

Lasso add this term to the error function

##pic from datarookie slide