Chapter 19: Interactive Notebook for Students

Ram Gopal, Dan Philps, and Tillman Weyde

2022

Contents

Load libraries

```
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
library(rattle)
```

```
## Loading required package: tibble

## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
if(!require("pacman")){
  install.packages("pacman")
}
```

```
## Loading required package: pacman
```

```
pacman::p_load("remotes", "RnavGraphImageData", "magrittr", "nnet", "data.table",
               "caret", "rpart", "tictoc", "naivebayes", "RSNNS", "mvtnorm", "patchwork")
```

Get the data

This is an interesting and somewhat larger dataset. This dataset contains grayscale images of faces of 40 people. The each image has has 64 x 64 pixels, with values in the range [0,1]. The data depicts 10 images for each person with a total of 40 individuals in the dataset. To keep the computations manageable, only 50 columns of data are read.

Obtain the data from the source

- The following code reads a few libraries and standardizes the data.

```
if(!require("snedata")){
  remotes::install_github("jlmelville/snedata")
}
```

```
## Loading required package: snedata
```

```
library(snedata)
library(RnavGraphImageData)
```

```
oli <- as.data.frame(snedata::olivetti_faces())[1:50]
oli <- as.data.table(snedata::olivetti_faces())[1:50]
y <- oli[, .(Label)] %>%
  .[, Label := factor(Label)]
X <- as.data.table(scale(oli[, !c("Label")]))
y <- oli[, .(Label)] %>%
  .[, Label := factor(Label)]
X <- as.data.table(scale(oli[, !c("Label")]))
oli_sc <- cbind(y, X)
```

Directly read the data

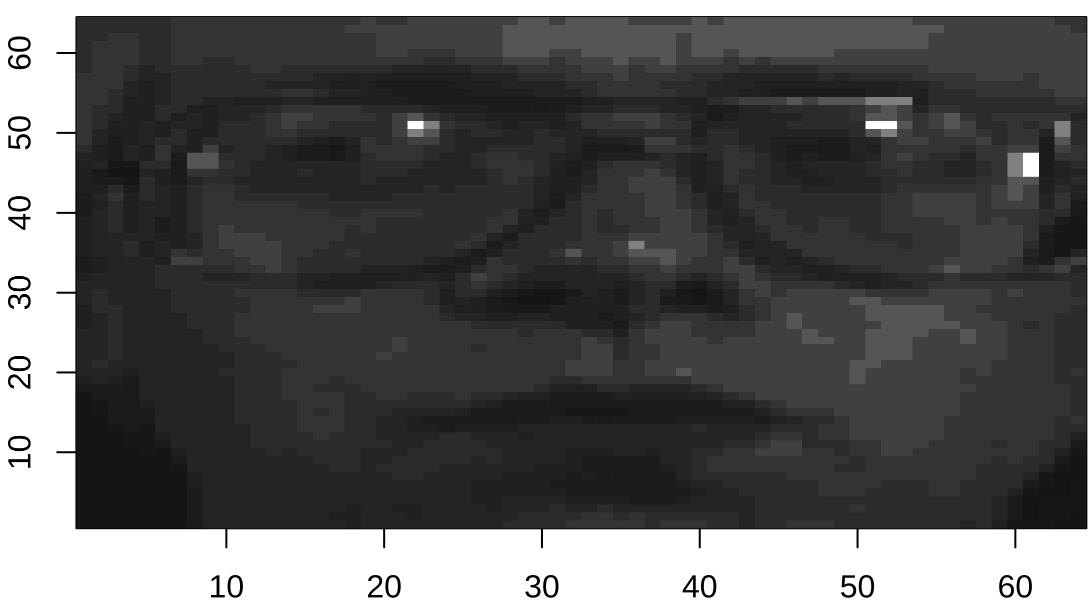
- You can skip the above steps and directly read the data from the online repository.

```
oli_sc = read.csv("oli_sc.csv")
```

View the images

- You can view the images by changing the last two parameters below which are the individual and image number for that individual.

```
show_olivetti_face(snedata::olivetti_faces(), 2, 3)
```



Modeling ## Create Partition

```r
set.seed(123456)
oli_sc$Label = as.factor(oli_sc$Label)
index <- createDataPartition(oli_sc$Label,p=0.8,list=FALSE)
train <- oli_sc[index,]
test <- oli_sc[-index,]
trControl <- trainControl(method  = "cv", number  = 2)
```

Run Models

Multinomial Regression

- Functions tic and toc from the tictoc package track the compute time for each of the algorithms. Useful to experiment with it across algorithms.
- The model below takes a while to run.
- Note the notation caret::train(). This indicates the function train() from the caret package. This comes in useful as at times the same function name be used by different packages.

Decision Tree, Naive Bayes, Multi Layer Perceptron algorithms

```r
for (mdl in c("rpart","naive_bayes","mlp"))
{
     mdl_model <- caret::train(Label ~ .,
             method=mdl,
             trControl  = trControl,
             data = train,
             metric     = "Accuracy")
     mdl_pred_test= predict(mdl_model,test)
     mdl_pred_train= predict(mdl_model,train)
    train_accuracy = caret::confusionMatrix(mdl_pred_train,train$Label)$overall[1]
    test_accuracy = caret::confusionMatrix(mdl_pred_test,test$Label)$overall[1]
    print(paste("Model ", mdl, " Accuracy: ",
                "Training = ", train_accuracy,
                " Test = ",test_accuracy))
    print(caret::confusionMatrix(mdl_pred_test,test$Label))
}
```

```
## [1] "Model  rpart  Accuracy:  Training =  0.6  Test =  0.6"
## Confusion Matrix and Statistics
##
##             Reference
## Prediction 1 2 3 4 5
##          1 2 0 0 2 1
##          2 0 2 0 0 0
##          3 0 0 2 0 1
##          4 0 0 0 0 0
##          5 0 0 0 0 0
##
## Overall Statistics
##
##                Accuracy : 0.6
```

```
##                   95% CI : (0.2624, 0.8784)
##     No Information Rate : 0.2
##     P-Value [Acc > NIR] : 0.006369
##
##                   Kappa : 0.5
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity            1.0000      1.0   1.0000      0.0      0.0
## Specificity            0.6250      1.0   0.8750      1.0      1.0
## Pos Pred Value         0.4000      1.0   0.6667      NaN      NaN
## Neg Pred Value         1.0000      1.0   1.0000      0.8      0.8
## Prevalence             0.2000      0.2   0.2000      0.2      0.2
## Detection Rate         0.2000      0.2   0.2000      0.0      0.0
## Detection Prevalence   0.5000      0.2   0.3000      0.0      0.0
## Balanced Accuracy      0.8125      1.0   0.9375      0.5      0.5
## [1] "Model  naive_bayes  Accuracy:  Training =  1  Test =  1"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 1 2 3 4 5
##          1 2 0 0 0 0
##          2 0 2 0 0 0
##          3 0 0 2 0 0
##          4 0 0 0 2 0
##          5 0 0 0 0 2
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.6915, 1)
##     No Information Rate : 0.2
##     P-Value [Acc > NIR] : 1.024e-07
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity               1.0      1.0      1.0      1.0      1.0
## Specificity               1.0      1.0      1.0      1.0      1.0
## Pos Pred Value            1.0      1.0      1.0      1.0      1.0
## Neg Pred Value            1.0      1.0      1.0      1.0      1.0
## Prevalence                0.2      0.2      0.2      0.2      0.2
## Detection Rate            0.2      0.2      0.2      0.2      0.2
## Detection Prevalence      0.2      0.2      0.2      0.2      0.2
## Balanced Accuracy         1.0      1.0      1.0      1.0      1.0
## [1] "Model  mlp  Accuracy:  Training =  0.925  Test =  0.7"
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction 1 2 3 4 5
##          1 2 0 0 0 0
##          2 0 2 0 1 0
##          3 0 0 2 0 0
##          4 0 0 0 0 1
##          5 0 0 0 1 1
##
## Overall Statistics
##
##                Accuracy : 0.7
##                  95% CI : (0.3475, 0.9333)
##     No Information Rate : 0.2
##     P-Value [Acc > NIR] : 0.0008644
##
##                   Kappa : 0.625
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5
## Sensitivity               1.0   1.0000      1.0   0.0000   0.5000
## Specificity               1.0   0.8750      1.0   0.8750   0.8750
## Pos Pred Value            1.0   0.6667      1.0   0.0000   0.5000
## Neg Pred Value            1.0   1.0000      1.0   0.7778   0.8750
## Prevalence                0.2   0.2000      0.2   0.2000   0.2000
## Detection Rate            0.2   0.2000      0.2   0.0000   0.1000
## Detection Prevalence      0.2   0.3000      0.2   0.1000   0.2000
## Balanced Accuracy         1.0   0.9375      1.0   0.4375   0.6875
```

Scaling

```
train1 = as.data.frame(scale(train[,-1]))
test1 = as.data.frame(scale(test[,-1]))
train1$Label = train$Label
test1$Label = test$Label

trControl <- trainControl(method  = "cv", number  = 2)
for (mdl in c("naive_bayes","mlp"))
{
     mdl_model <- caret::train(Label ~ .,
             method=mdl,
             trControl  = trControl,
             data = train1,
             metric     = "Accuracy")
     mdl_pred_test= predict(mdl_model,test1)
     mdl_pred_train= predict(mdl_model,train1)
    train_accuracy = caret::confusionMatrix(mdl_pred_train,train1$Label)$overall[1]
    test_accuracy = caret::confusionMatrix(mdl_pred_test,test1$Label)$overall[1]
    print(paste("Model ", mdl, " Accuracy: ",
```

```
                 "Training = ", train_accuracy,
                 " Test = ",test_accuracy))
}
```

```
## [1] "Model  naive_bayes  Accuracy:  Training =  1  Test =  1"
## [1] "Model  mlp  Accuracy:  Training =  1  Test =  1"
```

Iris Dataset

```
index <- caret::createDataPartition(iris$Species,p=0.5,list=FALSE)
train_iris <- iris[index,]
test_iris <- iris[-index,]
trControl <- trainControl(method  = "cv", number  = 2)
for (mdl in c("multinom","rpart","naive_bayes","mlp"))
{
     mdl_model <- caret::train(Species ~ .,
             method=mdl,
             trControl  = trControl,
             data = train_iris,
             metric     = "Accuracy")
     mdl_pred_test= predict(mdl_model,test_iris)
     mdl_pred_train= predict(mdl_model,train_iris)
    train_accuracy =
      caret::confusionMatrix(mdl_pred_train,train_iris$Species)$overall[1]
    test_accuracy =
      caret::confusionMatrix(mdl_pred_test,test_iris$Species)$overall[1]
    print(paste("Model ", mdl, " Accuracy: ",
                 "Training = ", train_accuracy,
                 " Test = ",test_accuracy))
}
```

```
## # weights:  18 (10 variable)
## initial  value 40.648655
## iter  10 value 3.666237
## iter  20 value 0.053687
## iter  30 value 0.011145
## iter  40 value 0.007880
## iter  50 value 0.002872
## iter  60 value 0.000200
## iter  70 value 0.000182
## iter  80 value 0.000133
## final  value 0.000097
## converged
## # weights:  18 (10 variable)
## initial  value 40.648655
## iter  10 value 10.663145
## iter  20 value 10.400859
## final  value 10.400800
## converged
## # weights:  18 (10 variable)
```

```
## initial  value 40.648655
## iter  10 value 3.687801
## iter  20 value 0.712625
## iter  30 value 0.562126
## iter  40 value 0.548687
## iter  50 value 0.517861
## iter  60 value 0.509088
## iter  70 value 0.507534
## iter  80 value 0.506853
## iter  90 value 0.505573
## iter 100 value 0.505018
## final  value 0.505018
## stopped after 100 iterations
## # weights:  18 (10 variable)
## initial  value 41.747267
## iter  10 value 3.589957
## iter  20 value 0.018922
## iter  30 value 0.003226
## iter  40 value 0.002057
## iter  50 value 0.001511
## iter  60 value 0.001352
## iter  70 value 0.001294
## iter  80 value 0.000493
## iter  90 value 0.000439
## iter 100 value 0.000362
## final  value 0.000362
## stopped after 100 iterations
## # weights:  18 (10 variable)
## initial  value 41.747267
## iter  10 value 11.588698
## iter  20 value 11.453868
## final  value 11.453849
## converged
## # weights:  18 (10 variable)
## initial  value 41.747267
## iter  10 value 3.613628
## iter  20 value 0.613770
## iter  30 value 0.499080
## iter  40 value 0.461368
## iter  50 value 0.442449
## iter  60 value 0.412064
## iter  70 value 0.397030
## iter  80 value 0.388383
## iter  90 value 0.379896
## iter 100 value 0.377604
## final  value 0.377604
## stopped after 100 iterations
## # weights:  18 (10 variable)
## initial  value 82.395922
## iter  10 value 18.563554
## iter  20 value 17.147657
## iter  30 value 17.145437
## iter  30 value 17.145437
## iter  30 value 17.145437
```

```
## final  value 17.145437
## converged
## [1] "Model  multinom  Accuracy:  Training =  0.973333333333333  Test =  0.973333333333333"
## [1] "Model  rpart  Accuracy:  Training =  0.973333333333333  Test =  0.933333333333333"
## [1] "Model  naive_bayes  Accuracy:  Training =  0.986666666666667  Test =  0.92"
## [1] "Model  mlp  Accuracy:  Training =  0.986666666666667  Test =  0.986666666666667"
```

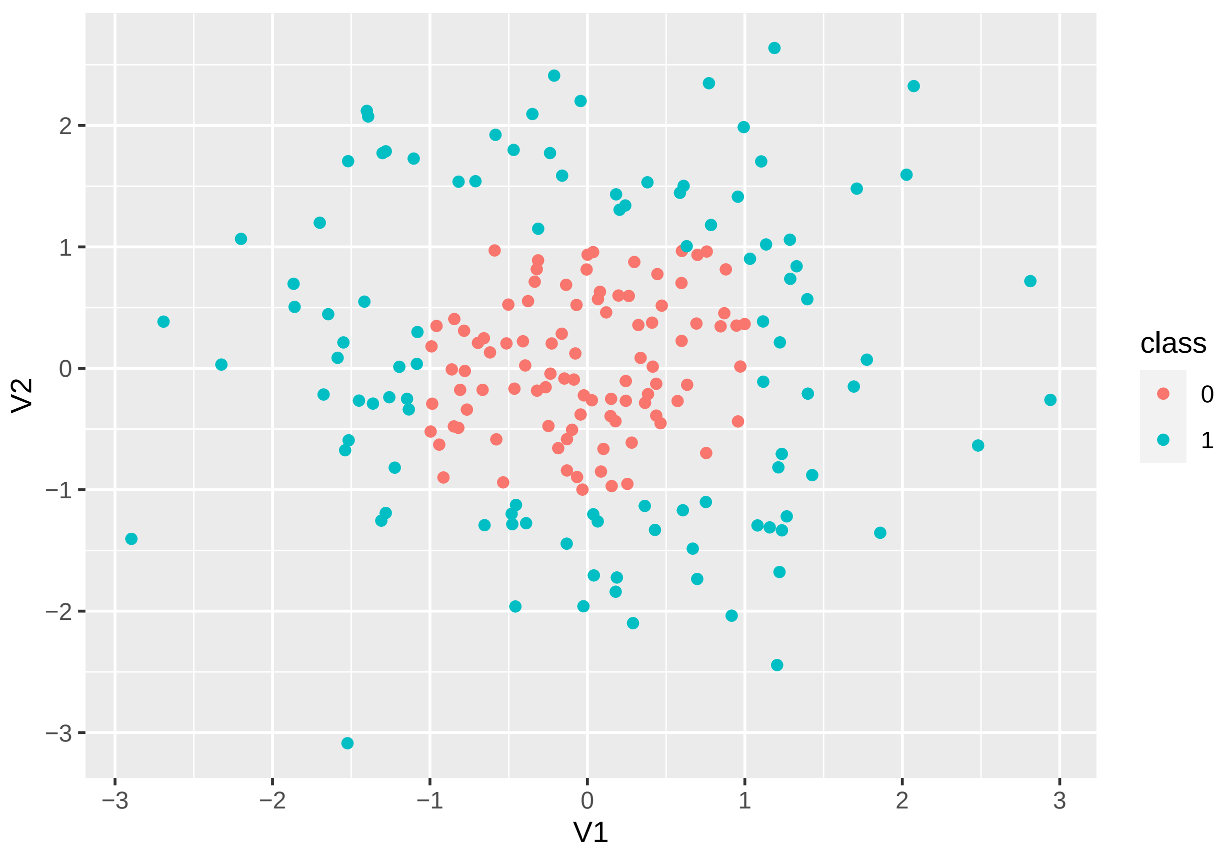
Synthetic Data 1

Data creation

```
x <- as.data.frame(rmvnorm(n=400, sigma=matrix(c(1,0,0,1), ncol=2)))
x$class =  factor(ifelse((x$V1 > -1 & x$V1 < 1 )&(x$V2 > -1 & x$V2 < 1 ), 0, 1))
```

Partition and visualize

```
index <- caret::createDataPartition(x$class,p=0.5,list=FALSE)
train_x <- x[index,]
test_x <- x[-index,]
ggplot(train_x) +
  geom_point(aes(V1, V2, color = class))
```



Multinomial

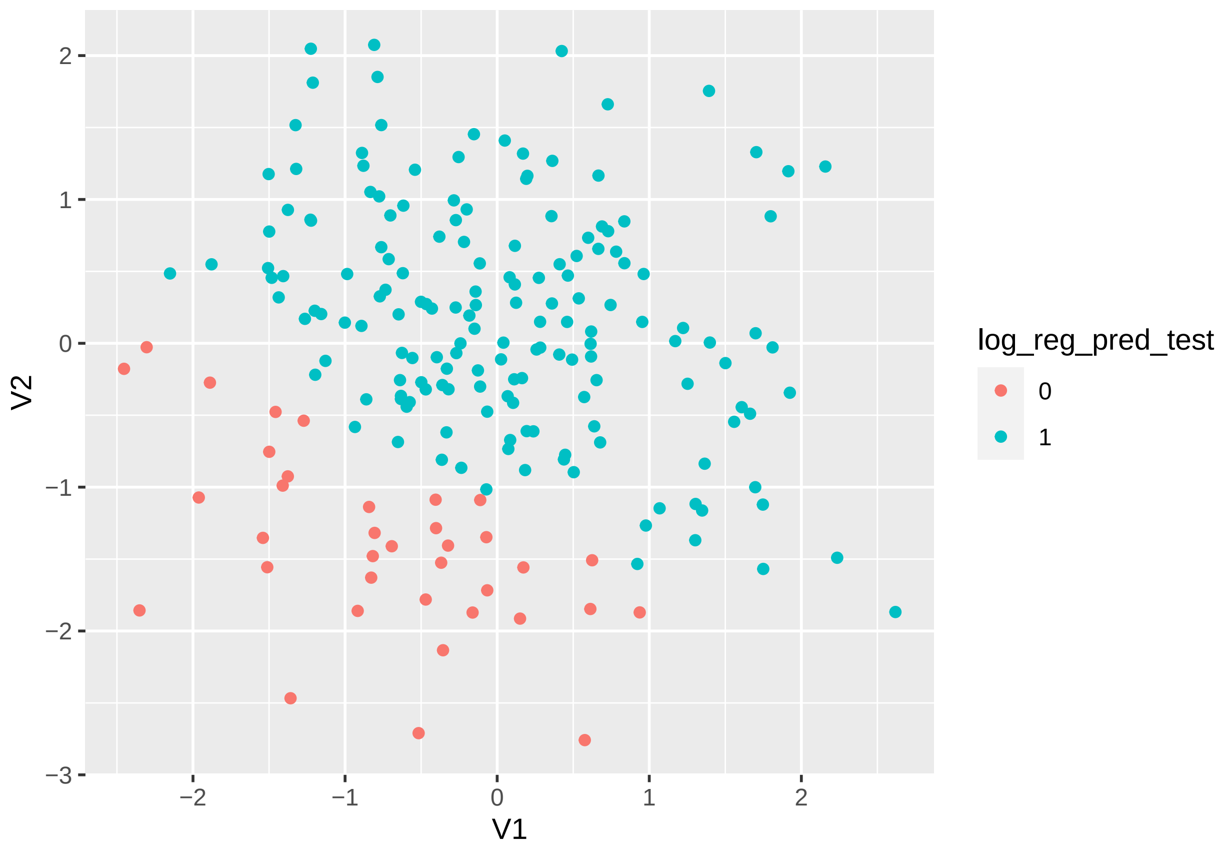
```r
trControl <- trainControl(method  = "cv", number  = 2)
log_reg = caret::train(class ~ .,   method='multinom',
             trControl  = trControl,
             data = train_x,
             metric = "Accuracy",
              MaxNWts = 10000000,
             maxit = 10)
```

```
## # weights:  4 (3 variable)
## initial  value 69.314718
## final  value 67.889467
## converged
## # weights:  4 (3 variable)
## initial  value 69.314718
## final  value 67.903065
## converged
## # weights:  4 (3 variable)
## initial  value 69.314718
## final  value 67.889481
## converged
## # weights:  4 (3 variable)
## initial  value 69.314718
## final  value 68.962546
## converged
## # weights:  4 (3 variable)
## initial  value 69.314718
## final  value 68.965065
## converged
## # weights:  4 (3 variable)
## initial  value 69.314718
## final  value 68.962548
## converged
## # weights:  4 (3 variable)
## initial  value 138.629436
## final  value 138.315454
## converged
```

```r
log_reg_pred_train = predict(log_reg, train_x)
log_reg_pred_test = predict(log_reg, test_x)
train_accuracy =
  caret::confusionMatrix(log_reg_pred_train,train_x$class)$overall[1]
test_accuracy =
  caret::confusionMatrix(log_reg_pred_test,test_x$class)$overall[1]
print(paste("Multinomial Regression Accuracy: ",
                "Training = ", train_accuracy,
                " Test = ",test_accuracy))
```

```
## [1] "Multinomial Regression Accuracy:  Training =  0.4  Test =  0.34"
```

```r
ggplot(test_x) +
  geom_point(aes(V1, V2, color = log_reg_pred_test))
```

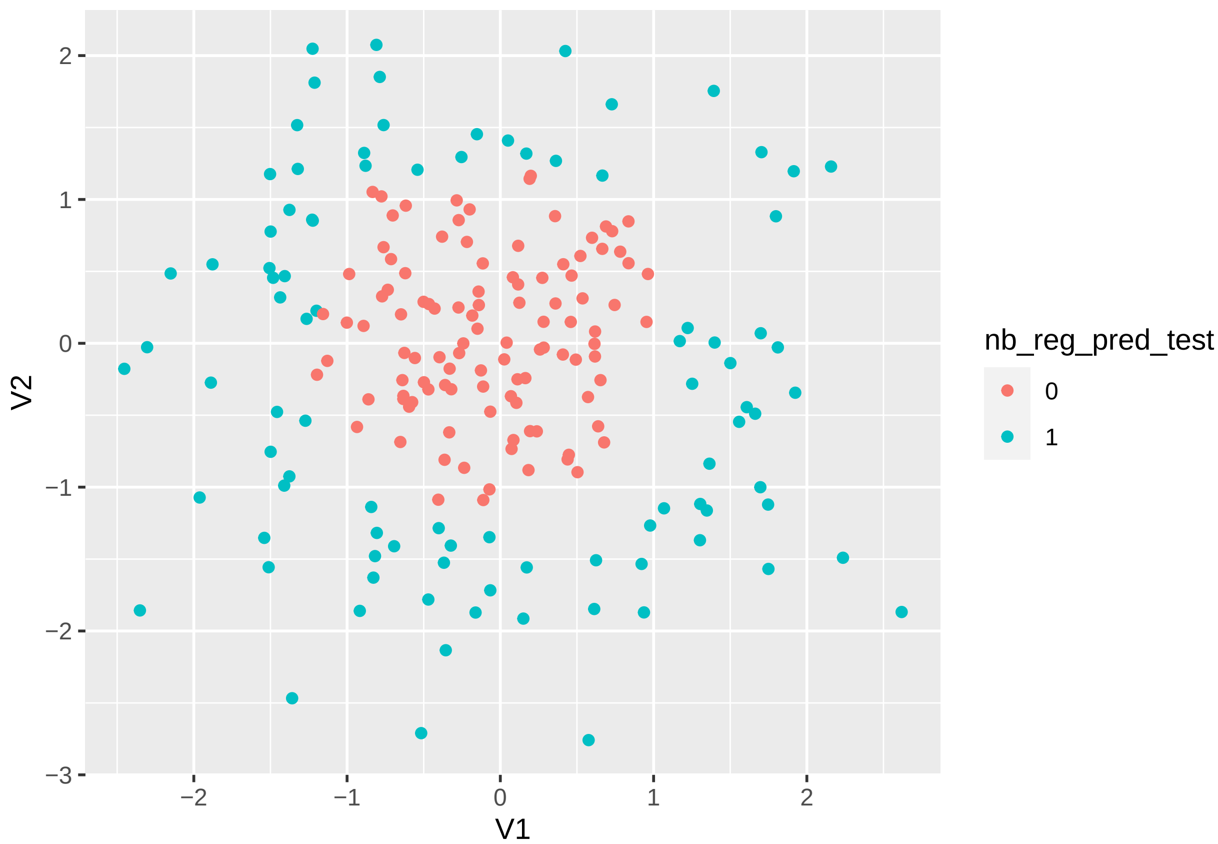


##Naive Bayes

```
trControl <- trainControl(method  = "cv", number  = 2)
nb_reg = caret::train(class ~ .,   method='naive_bayes',
             trControl  = trControl,
             data = train_x,
             metric = "Accuracy")
nb_reg_pred_train = predict(nb_reg, train_x)
nb_reg_pred_test = predict(nb_reg, test_x)
train_accuracy =
  caret::confusionMatrix(nb_reg_pred_train,train_x$class)$overall[1]
test_accuracy =
  caret::confusionMatrix(nb_reg_pred_test,test_x$class)$overall[1]
print(paste("Naive Bayes: ",
                "Training = ", train_accuracy,
                " Test = ",test_accuracy))
```

```
## [1] "Naive Bayes:  Training =  0.955  Test =  0.945"
```

```
ggplot(test_x) +
  geom_point(aes(V1, V2, color = nb_reg_pred_test))
```

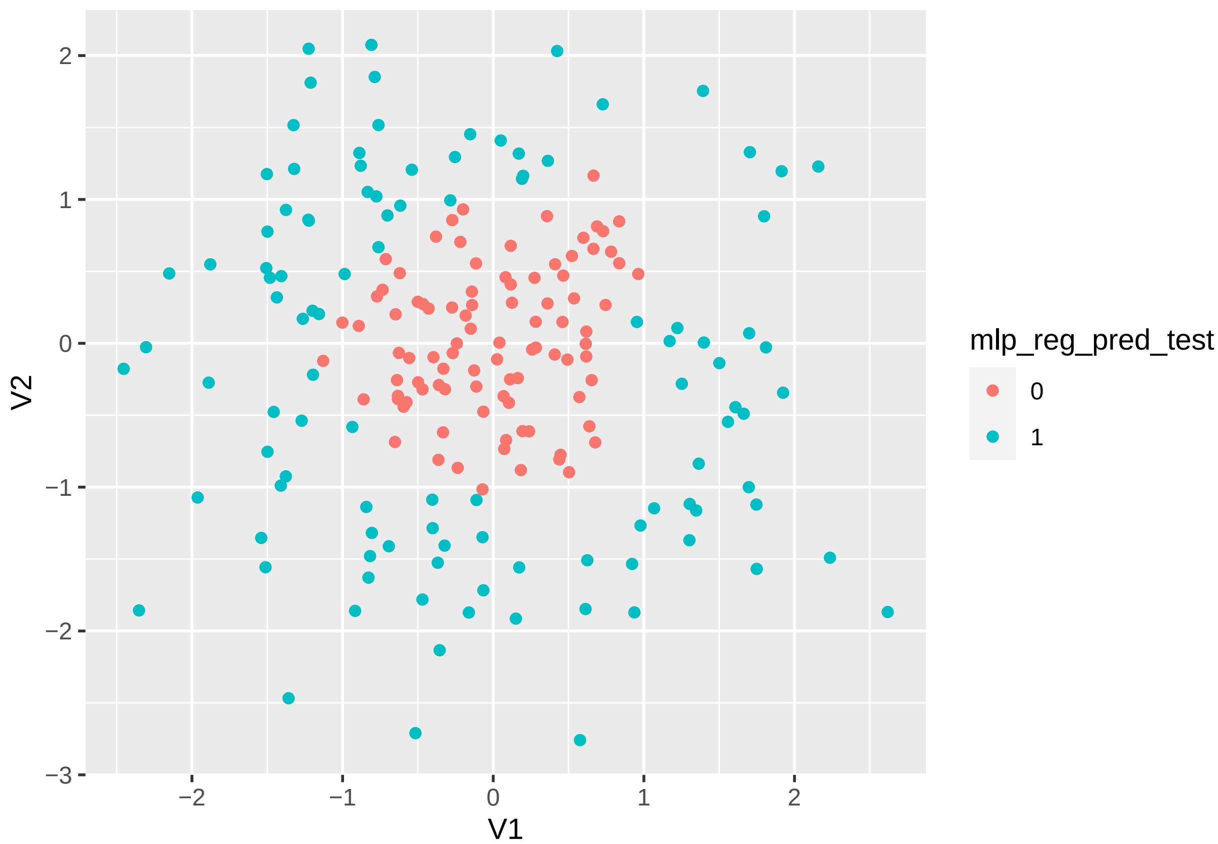


Multi Layer Perceptron

```
trControl <- trainControl(method  = "cv", number  = 2)
mlp_reg = caret::train(class ~ .,   method='mlp',
             trControl  = trControl,
             data = train_x,
             metric = "Accuracy")
mlp_reg_pred_train = predict(mlp_reg, train_x)
mlp_reg_pred_test = predict(mlp_reg, test_x)
train_accuracy =
  caret::confusionMatrix(mlp_reg_pred_train,train_x$class)$overall[1]
test_accuracy =
  caret::confusionMatrix(mlp_reg_pred_test,test_x$class)$overall[1]
print(paste("MLP: ",
                "Training = ", train_accuracy,
                " Test = ",test_accuracy))
```

```
## [1] "MLP:  Training =  0.935  Test =  0.945"
```

```
ggplot(test_x) +
  geom_point(aes(V1, V2, color = mlp_reg_pred_test))
```



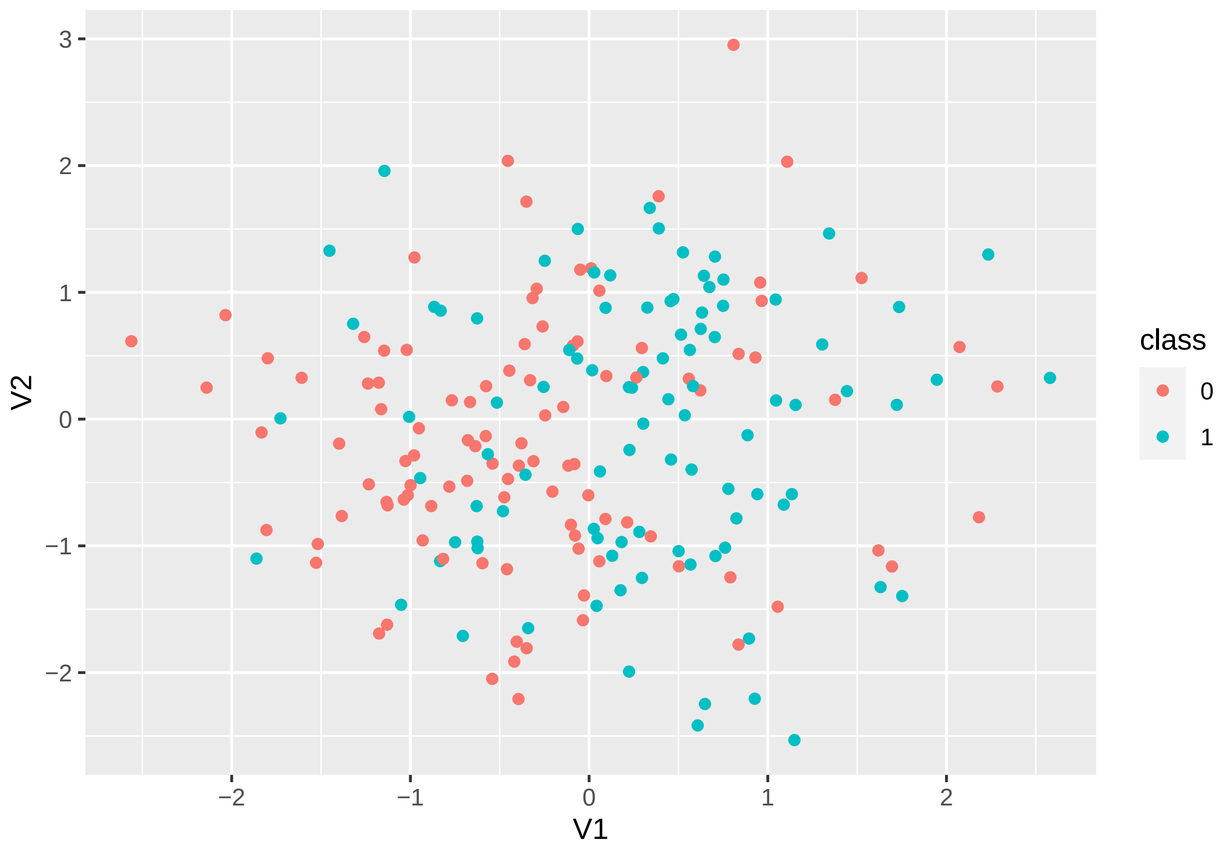
Synthetic Data 2

Data creation

```
x <- as.data.frame(rmvnorm(n=400, sigma=matrix(c(1,0,0,1), ncol=2)))
x$class1 = ifelse(x$V1<0,0,1)
x$class2 = ifelse(x$class1==0,1,0)
x$flip = ifelse(runif(nrow(x))<0.3,1,0)
x$class = factor(ifelse(x$flip==1,x$class2,x$class1))
x = x[,c(1,2,6)]
```

Partition and visualize

```
index <- caret::createDataPartition(x$class,p=0.5,list=FALSE)
train_x <- x[index,]
test_x <- x[-index,]
ggplot(train_x) +
  geom_point(aes(V1, V2, color = class))
```



Logistic Regression

```
trControl <- trainControl(method  = "cv", number  = 2)
log_reg = caret::train(class ~ .,   method='multinom',
             trControl   = trControl,
             data = train_x,
             metric = "Accuracy",
              MaxNWts = 10000000,
             maxit = 10)
```

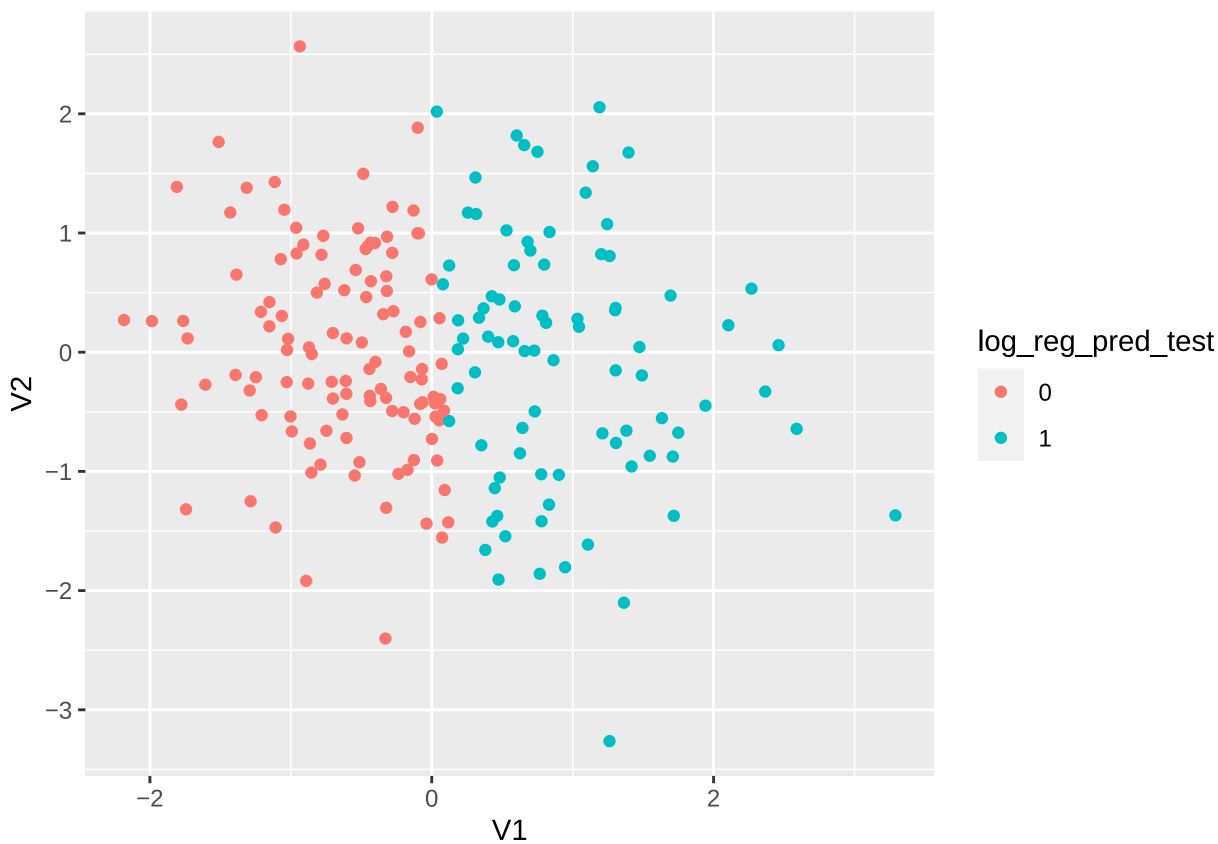
```
## # weights:  4 (3 variable)
## initial  value 69.314718
## final  value 64.355972
## converged
## # weights:  4 (3 variable)
## initial  value 69.314718
## final  value 64.419891
## converged
## # weights:  4 (3 variable)
## initial  value 69.314718
## final  value 64.356037
## converged
## # weights:  4 (3 variable)
## initial  value 69.314718
## final  value 62.469516
## converged
## # weights:  4 (3 variable)
```

```
## initial  value 69.314718
## final  value 62.535079
## converged
## # weights:  4 (3 variable)
## initial  value 69.314718
## final  value 62.469583
## converged
## # weights:  4 (3 variable)
## initial  value 138.629436
## final  value 127.366385
## converged
```

```
log_reg_pred_train = predict(log_reg, train_x)
log_reg_pred_test = predict(log_reg, test_x)
train_accuracy =
  caret::confusionMatrix(log_reg_pred_train,train_x$class)$overall[1]
test_accuracy =
  caret::confusionMatrix(log_reg_pred_test,test_x$class)$overall[1]
print(paste("Multinomial Regression Accuracy: ",
                "Training = ", train_accuracy,
                " Test = ",test_accuracy))
```

```
## [1] "Multinomial Regression Accuracy:  Training =  0.705  Test =  0.635"
```

```
ggplot(test_x) +
  geom_point(aes(V1, V2, color = log_reg_pred_test))
```

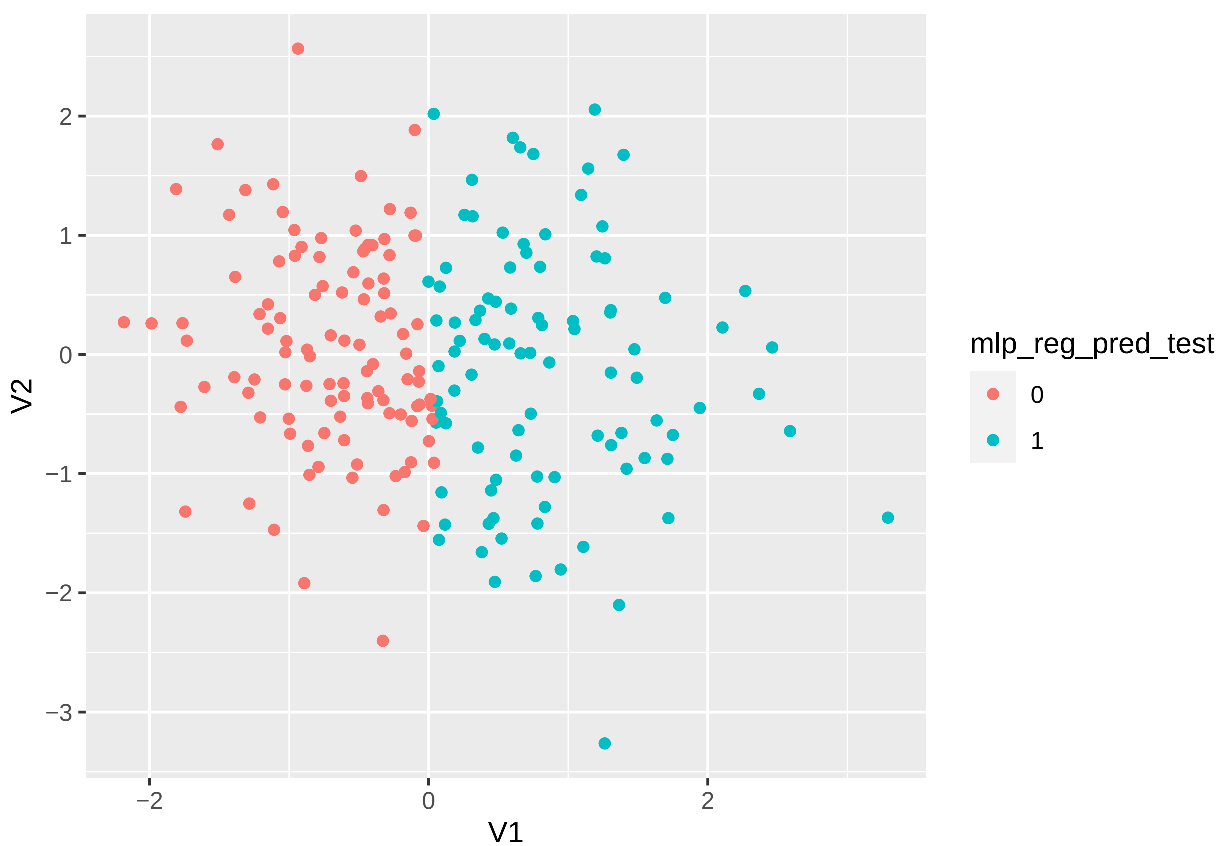


Multi Layer Perceptron

```
trControl <- trainControl(method  = "cv", number  = 2)
mlp_reg = caret::train(class ~ .,   method='mlp',
             trControl  = trControl,
             data = train_x,
             metric = "Accuracy")
mlp_reg_pred_train = predict(mlp_reg, train_x)
mlp_reg_pred_test = predict(mlp_reg, test_x)
train_accuracy =
  caret::confusionMatrix(mlp_reg_pred_train,train_x$class)$overall[1]
test_accuracy =
  caret::confusionMatrix(mlp_reg_pred_test,test_x$class)$overall[1]
print(paste("MLP: ",
                "Training = ", train_accuracy,
                " Test = ",test_accuracy))
```

```
## [1] "MLP:  Training =  0.705  Test =  0.67"
```

```
ggplot(test_x) +
  geom_point(aes(V1, V2, color = mlp_reg_pred_test))
```



L2 Regularization

```
trControl <- trainControl(method  = "cv", number  = 2)
mlp_reg = caret::train(class ~ .,   method = 'mlp',
                       learnFuncParams=c(0.01),
             trControl  = trControl,
             data = train_x,
             metric = "Accuracy")
mlp_reg_pred_train = predict(mlp_reg, train_x)
mlp_reg_pred_test = predict(mlp_reg, test_x)
train_accuracy =
  caret::confusionMatrix(mlp_reg_pred_train,train_x$class)$overall[1]
test_accuracy =
  caret::confusionMatrix(mlp_reg_pred_test,test_x$class)$overall[1]
print(paste("MLP: ",
                "Training = ", train_accuracy,
                " Test = ",test_accuracy))
```

```
## [1] "MLP:  Training =  0.69  Test =  0.635"
```

```
ggplot(test_x) +
  geom_point(aes(V1, V2, color = mlp_reg_pred_test))
```