# P8106_Midterm

jck2183_Chia-wen Kao

2021/3/26

```r
library(tidyverse)
library(caret)
library(glmnet)
library(mlbench)
library(pROC) #generate ROC curve and calculate AUC
library(pdp) #partial dependent plot
library(vip) #variable importance plot: global impact on different predictor
library(AppliedPredictiveModeling) # for visualization purpose
library(corrplot)
library(RColorBrewer)
library(RANN)
library(visdat)
library(mgcv)
```

## Introduction:

According to the World Health Organization (WHO) stroke is the 2nd leading cause of death globally, responsible for approximately 11% of total deaths. This dataset is used to predict whether a patient is likely to get stroke based on the input parameters like gender, age, various diseases, and smoking status. Each row in the data provides relavant information about the patient.

Data Source: https://www.kaggle.com/fedesoriano/stroke-prediction-dataset

All the features we had:

- id: unique identifier
- gender: “Male”, “Female” or “Other”
- age: age of the patient
- hypertension: 0 if the patient doesn’t have hypertension, 1 if the patient has hypertension
- heart_disease: 0 if the patient doesn’t have any heart diseases, 1 if the patient has a heart disease
- ever_married: “No” or “Yes”
- work_type: “children”, “Govt_jov”, “Never_worked”, “Private” or “Self-employed”
- Residence_type: “Rural” or “Urban”
- avg_glucose_level: average glucose level in blood
- bmi: body mass index
- smoking_status: “formerly smoked”, “never smoked”, “smokes” or “Unknown”*
- stroke: 1 if the patient had a stroke or 0 if not *Note: “Unknown” in smoking_status means that the information is unavailable for this patient

**Import Data**

```r
stroke_df = read.csv("./data/healthcare-dataset-stroke-data.csv")
# head(stroke_df)

stroke_df$stroke = as.factor(stroke_df$stroke)
stroke_df$gender = factor(stroke_df$gender) %>% as.numeric()
stroke_df$ever_married = factor(stroke_df$ever_married) %>% as.numeric()
stroke_df$work_type = factor(stroke_df$work_type) %>% as.numeric()
stroke_df$Residence_type = factor(stroke_df$Residence_type) %>% as.numeric()
stroke_df$smoking_status = factor(stroke_df$smoking_status) %>% as.numeric()
stroke_df$heart_disease = factor(stroke_df$heart_disease) %>% as.numeric()
stroke_df$hypertension = as.numeric(factor(stroke_df$hypertension))
stroke_df$work_type = as.factor(stroke_df$work_type) %>% as.numeric()
stroke_df$bmi = as.numeric(stroke_df$bmi)
```

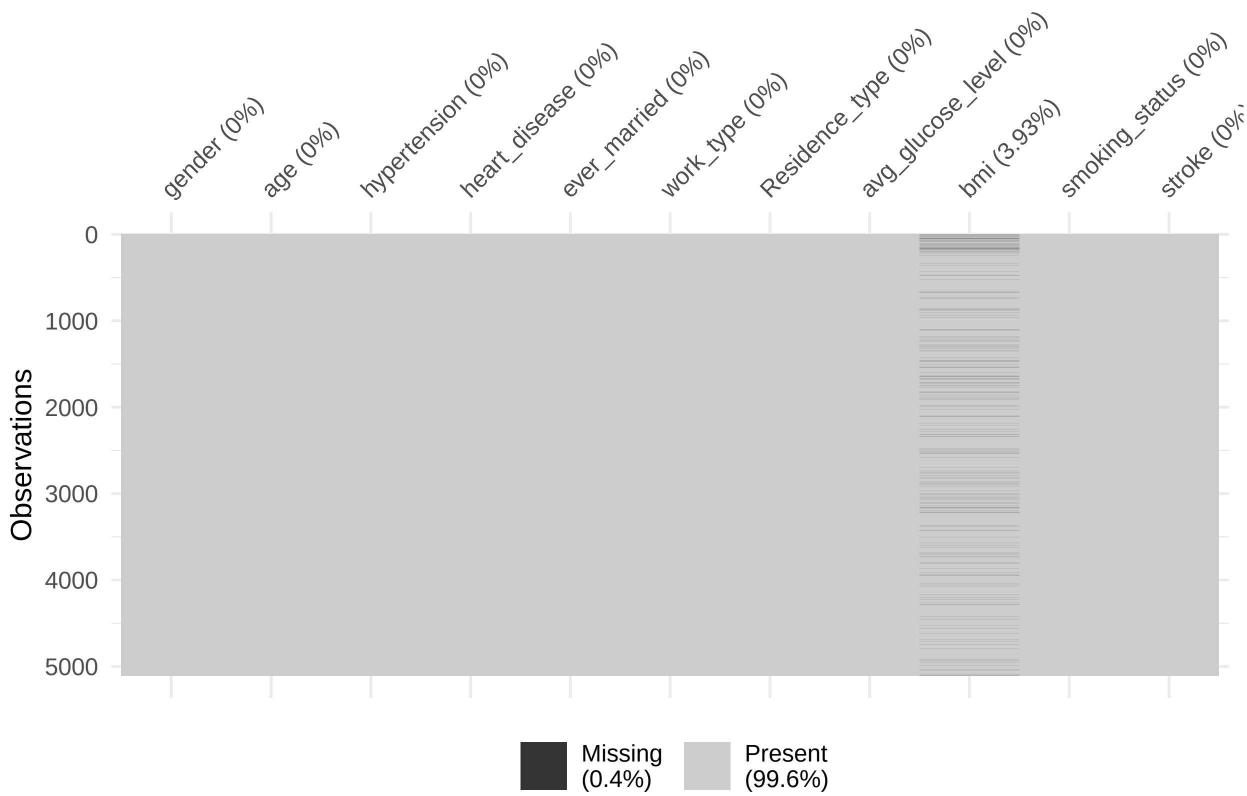
```
## Warning: NAs introduced by coercion
```

```r
stroke_df = stroke_df[, -1] %>%
    mutate(stroke = recode(stroke,
                           `0` = "No",
                           `1` = "Yes"),
           stroke = factor(stroke)) %>%
    filter(gender < 3)


summary(stroke_df)
```

```
##      gender           age         hypertension   heart_disease
##   Min.   :1.000   Min.   : 0.08   Min.   :1.000   Min.   :1.000
##   1st Qu.:1.000   1st Qu.:25.00   1st Qu.:1.000   1st Qu.:1.000
##   Median :1.000   Median :45.00   Median :1.000   Median :1.000
##   Mean   :1.414   Mean   :43.23   Mean   :1.097   Mean   :1.054
##   3rd Qu.:2.000   3rd Qu.:61.00   3rd Qu.:1.000   3rd Qu.:1.000
##   Max.   :2.000   Max.   :82.00   Max.   :2.000   Max.   :2.000
##
##   ever_married     work_type     Residence_type  avg_glucose_level
##   Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   : 55.12
##   1st Qu.:1.000   1st Qu.:2.000   1st Qu.:1.000   1st Qu.: 77.24
##   Median :2.000   Median :4.000   Median :2.000   Median : 91.88
##   Mean   :1.656   Mean   :3.495   Mean   :1.508   Mean   :106.14
##   3rd Qu.:2.000   3rd Qu.:4.000   3rd Qu.:2.000   3rd Qu.:114.09
##   Max.   :2.000   Max.   :5.000   Max.   :2.000   Max.   :271.74
##
##       bmi         smoking_status  stroke
##   Min.   :10.30   Min.   :1.000   No :4860
##   1st Qu.:23.50   1st Qu.:2.000   Yes: 249
##   Median :28.10   Median :2.000
##   Mean   :28.89   Mean   :2.586
##   3rd Qu.:33.10   3rd Qu.:4.000
##   Max.   :97.60   Max.   :4.000
##   NA's   :201
```

```r
vis_miss(stroke_df)
```




The imported dataset has 5110 observations in total. Excluding the id, we only gave ten features and one binary outcome variable-stroke (0:no stroke, 1:stroke). We found that the stroke outcome distribution is imbalanced with 4861 observations have no stroke while 249 observations have a stroke.

We find out there are 201 observations with missing values in BMI. Among these missing values, 40 observations have a stroke while 161 observations without stroke. We will then apply preprocess imputation in the caret train function to address the imputation problem. We also have 1544 unknown in smoke status, will treat those who answered unknown as a variable so no need to impute them.

Our main task is to find out the appropriate models that have a better performance on prediction by comparing several models' performance.

First, we have to convert character variables into factors to add them into our model and proceed with the analysis. Plus, we will also examine if there is any correlation among features. Meanwhile, we also found there is an observation who identified their gender as "Other". We decide to omit this single subject so that we can proceed with our analysis.

Next, the characteristics of features will help us determine which model would be proper. As the outcome is binary, and the features are mixtures of continuous and categorical variables. We also have to decide how to partition the train and test data, which cross-validation method to use. Evaluation metrics should be used and set up a reasonable tuning grid corresponding to the tuning parameter.

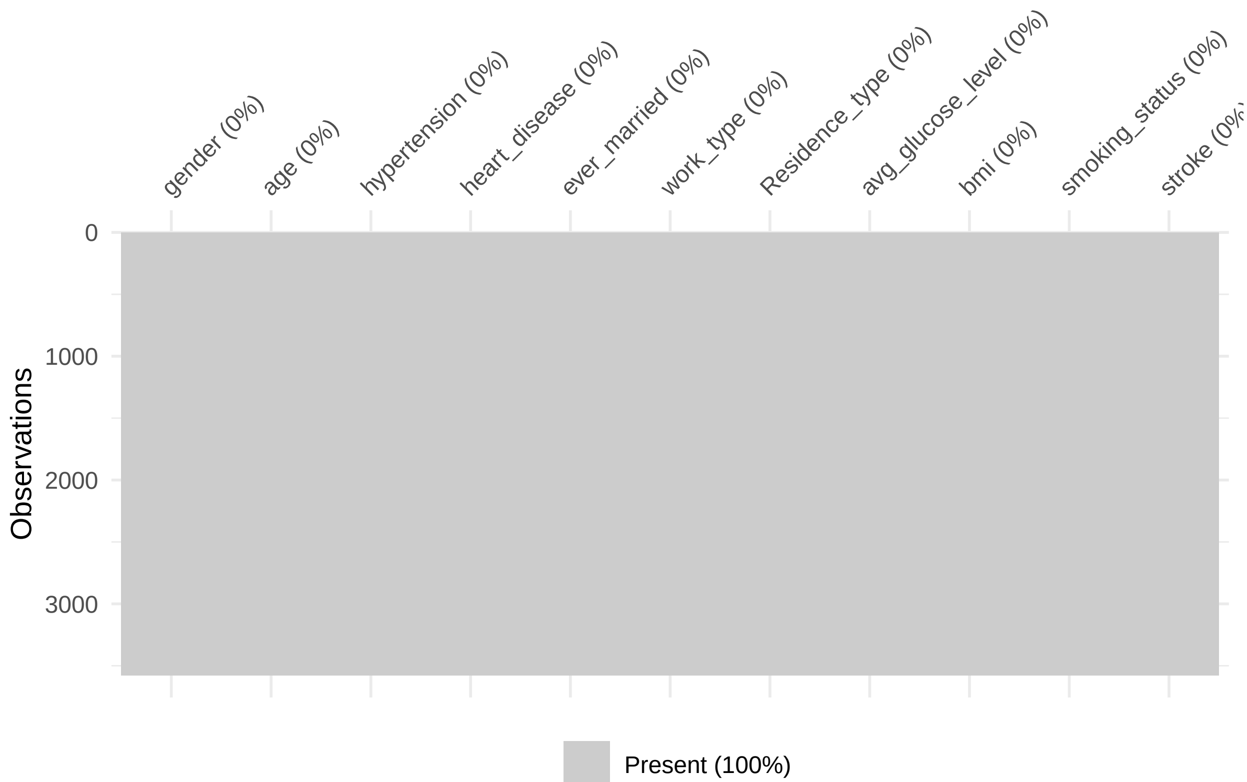
## Exploratory Data Analysis

Partition the dataset, I will use 70% as training data and 30% as test data.

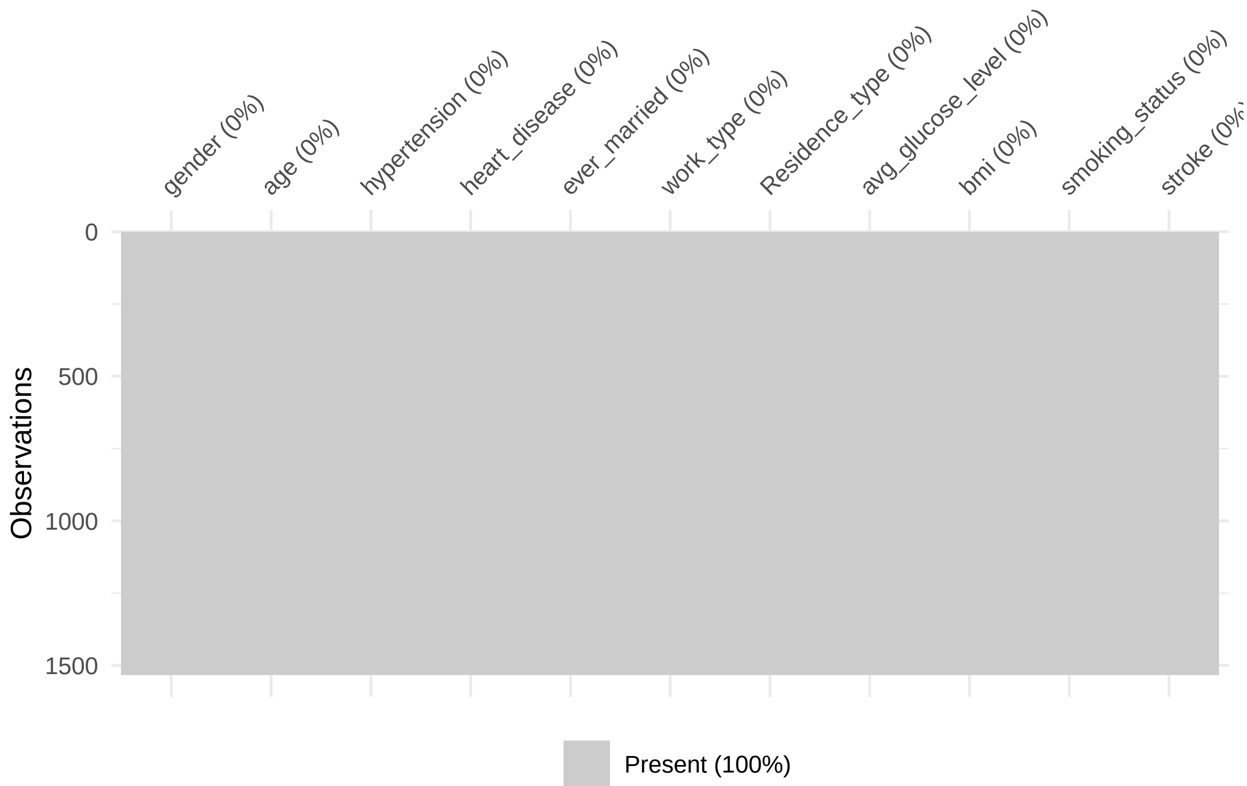
```r
set.seed(123)
trRow = createDataPartition(y = stroke_df$stroke, p = 0.7, list = F)
train.data = stroke_df[trRow, ]
test.data = stroke_df[-trRow, ]
```

Try imputation with `preProcess()`

```r
knnImp = preProcess(train.data, method = "knnImpute", k = 3)
train.data = predict(knnImp, train.data)
vis_miss(train.data)
```




```r
test.data = predict(knnImp,test.data)
vis_miss(test.data)
```




Try following models to see which algorithm fits the best because our outcome is binary and it would better to proceed with which classification performs the best. We will have accuracy and ROC/AUC as our evaluation metrics.

**Logistic Regression**

```
set.seed(123)
ctrl = trainControl(method = "cv",
                    summaryFunction = twoClassSummary,
                    classProbs = TRUE)

ctrl1 = trainControl(method = "cv",
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE, sampling = 'smote')

lm.fit = train( x = train.data[, c(1:10)],
                y = train.data$stroke,
                method = "glm",
                metric = "ROC",
                trControl = ctrl)

lm.fit1 = train( x = train.data[, c(1:10)],
                 y = train.data$stroke,
                 method = "glm",
                 metric = "ROC",
```

```r
                   trControl = ctrl1)
```

```
## Loading required package: grid

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```r
lm.pred = predict(lm.fit, newdata = test.data,
                          type = "prob")

lm.prob = ifelse(lm.pred$Yes > 0.5, "Yes", "No")



confusionMatrix(data = as.factor(lm.prob),
                reference = test.data$stroke,
                positive = "Yes")
```

```
## Warning in confusionMatrix.default(data = as.factor(lm.prob), reference =
## test.data$stroke, : Levels are not in the same order for reference and data.
## Refactoring data to match.

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  1458   74
##        Yes    0    0
##
##                Accuracy : 0.9517
##                  95% CI : (0.9397, 0.9619)
##     No Information Rate : 0.9517
##     P-Value [Acc > NIR] : 0.5309
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.0000
##             Specificity : 1.0000
##          Pos Pred Value :    NaN
##          Neg Pred Value : 0.9517
##              Prevalence : 0.0483
##          Detection Rate : 0.0000
##    Detection Prevalence : 0.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : Yes
##
```

```
roc.lm = roc(test.data$stroke, lm.pred[,2])
```

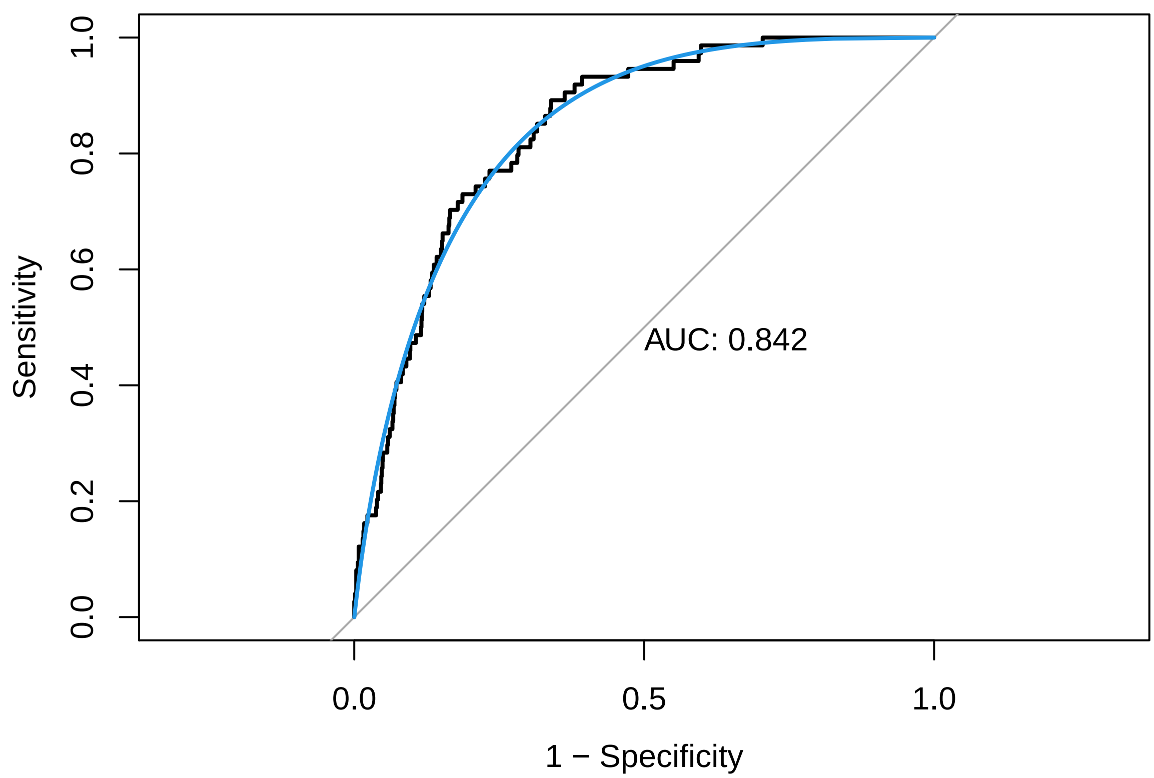
```
## Setting levels: control = No, case = Yes
```

```
## Setting direction: controls < cases
```

```
auc.lm = roc.lm$auc[1]
auc.lm
```

```
## [1] 0.8423516
```

```
plot(roc.lm, legacy.axes = TRUE, print.auc = TRUE)
plot(smooth(roc.lm), col = 4, add = TRUE)
```




#### Penalized logistic regression

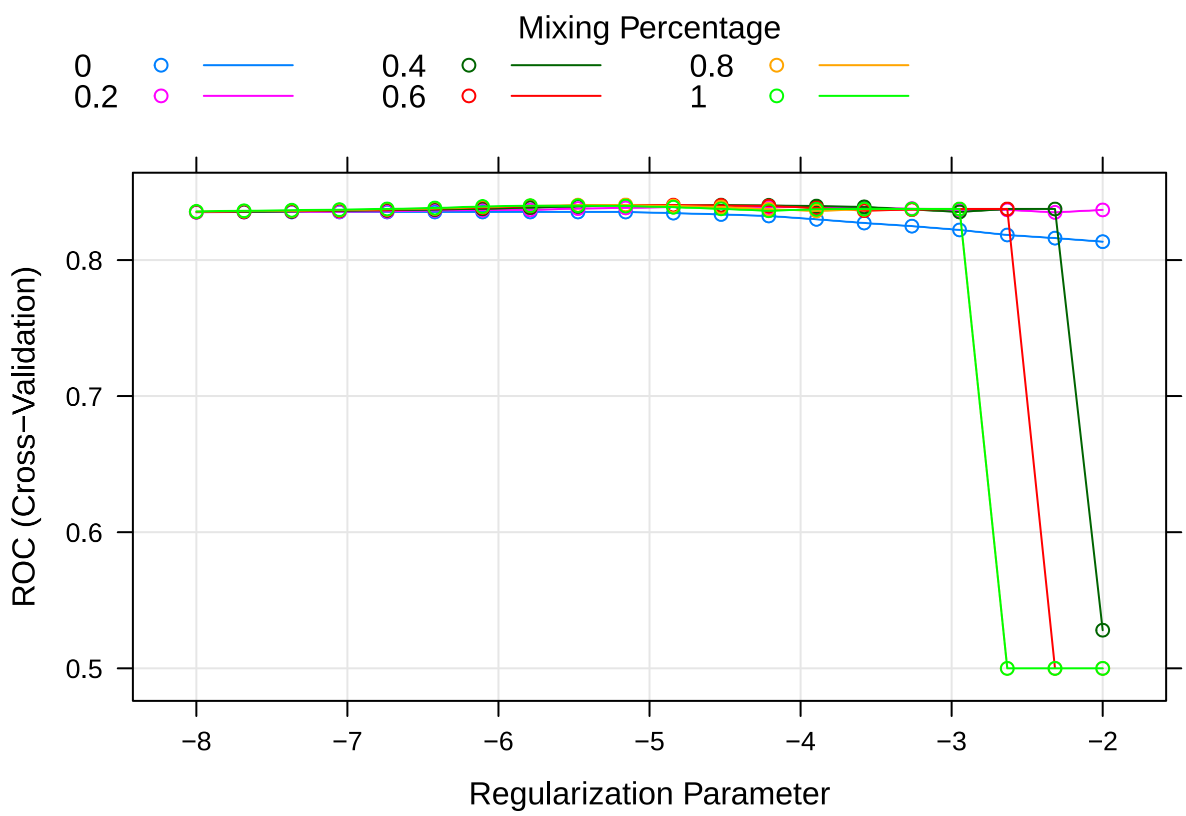
To add penalty to our loss, we can shrink the coefficients of correlated predictors towards each other by tuning alpha and lambda.

```
set.seed(123)
glmnGrid = expand.grid(.alpha = seq(0, 1, length = 6),
                       .lambda = exp(seq(-8, -2, length = 20)))
```

```r
model.glmn = train( x = train.data[,c(1:10)],
                    y = train.data$stroke,
                    method = "glmnet",
                    tuneGrid = glmnGrid,
                    metric = "ROC",
                    trControl = ctrl)

model.glmn1 = train( x = train.data[,c(1:10)],
                    y = train.data$stroke,
                    method = "glmnet",
                    tuneGrid = glmnGrid,
                    metric = "ROC",
                    trControl = ctrl1)

plot(model.glmn, xTrans = function(x) log(x))
```




```r
model.glmn$bestTune
```

```
##    alpha     lambda
## 70   0.6 0.0057538
```

```r
glmn.pred = predict(model.glmn, newdata = test.data, type = "prob")
glmn.prob = ifelse(glmn.pred$Yes > 0.5, "Yes", "No")
confusionMatrix(data = as.factor(glmn.prob),
```

```r
               reference = test.data$stroke,
               positive = "Yes")
```

```
## Warning in confusionMatrix.default(data = as.factor(glmn.prob), reference =
## test.data$stroke, : Levels are not in the same order for reference and data.
## Refactoring data to match.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  1458   74
##        Yes    0    0
##
##                Accuracy : 0.9517
##                  95% CI : (0.9397, 0.9619)
##     No Information Rate : 0.9517
##     P-Value [Acc > NIR] : 0.5309
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.0000
##             Specificity : 1.0000
##          Pos Pred Value :    NaN
##          Neg Pred Value : 0.9517
##              Prevalence : 0.0483
##          Detection Rate : 0.0000
##    Detection Prevalence : 0.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : Yes
##
```

```r
roc.glmn = roc(test.data$stroke, glmn.pred[,2])
```

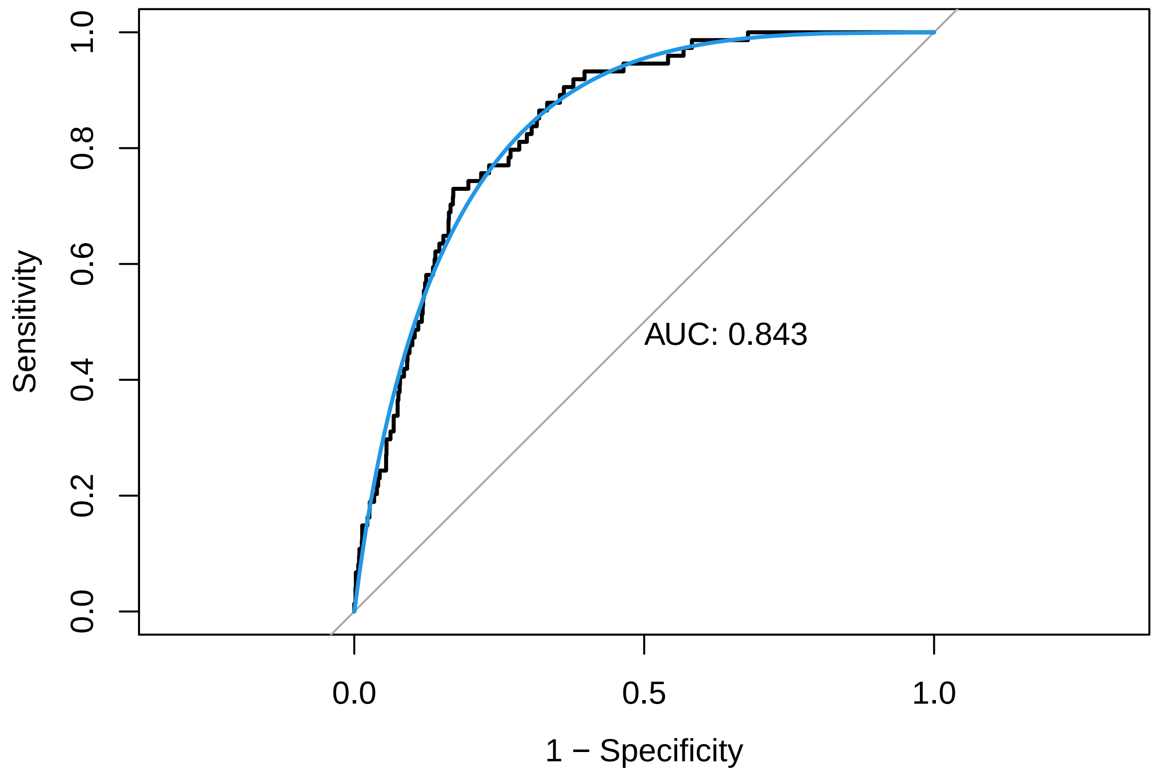
```
## Setting levels: control = No, case = Yes
```

```
## Setting direction: controls < cases
```

```r
auc.glmn = roc.glmn$auc[1]
auc.glmn
```

```
## [1] 0.8434824
```

```r
plot(roc.glmn, legacy.axes = TRUE, print.auc = TRUE)
plot(smooth(roc.glmn), col = 4, add = TRUE)
```




**Ridge Regression**

Ridge can also help us shrink the coefficients of correlated predictors towards each other by tuning only lambda.

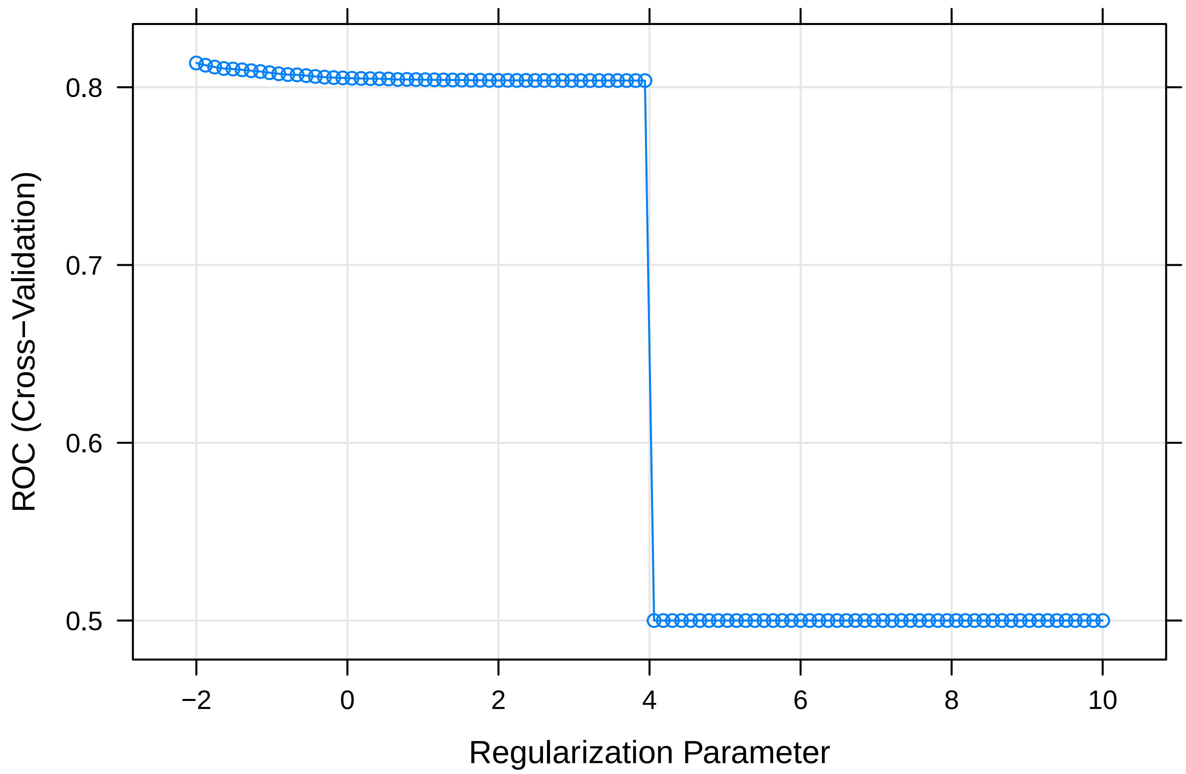
```
set.seed(123)
ridge.fit = train( x = train.data[,c(1:10)],
                   y = train.data$stroke,
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha = 0, #ridge
                                          lambda = exp(seq(10, -2, length=100))),
                   preProc = c("center", "scale"),
                   trControl = ctrl)
```

```
## Warning in train.default(x = train.data[, c(1:10)], y = train.data$stroke, : The
## metric "Accuracy" was not in the result set. ROC will be used instead.
```

```
ridge.fit1 = train( x = train.data[,c(1:10)],
                   y = train.data$stroke,
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha = 0, #ridge
                                          lambda = exp(seq(10, -2, length=100))),
                   preProc = c("center", "scale"),
                   trControl = ctrl1)
```

```
## Warning in train.default(x = train.data[, c(1:10)], y = train.data$stroke, : The
## metric "Accuracy" was not in the result set. ROC will be used instead.
```

```
#need to specify 2 tunning parameters.
plot(ridge.fit, xTrans = log)
```




```
ridge.pred = predict(ridge.fit, newdata = test.data, type = "prob")
ridge.prob = ifelse(ridge.pred$Yes > 0.5, "Yes", "No")
confusionMatrix(data = as.factor(ridge.prob),
                reference = test.data$stroke,
                positive = "Yes")
```

```
## Warning in confusionMatrix.default(data = as.factor(ridge.prob), reference =
## test.data$stroke, : Levels are not in the same order for reference and data.
## Refactoring data to match.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  1458   74
##        Yes    0    0
##
##                Accuracy : 0.9517
##                  95% CI : (0.9397, 0.9619)
```

```
##      No Information Rate : 0.9517
##      P-Value [Acc > NIR] : 0.5309
##
##                    Kappa : 0
##
##   Mcnemar's Test P-Value : <2e-16
##
##              Sensitivity : 0.0000
##              Specificity : 1.0000
##           Pos Pred Value :    NaN
##           Neg Pred Value : 0.9517
##               Prevalence : 0.0483
##           Detection Rate : 0.0000
##     Detection Prevalence : 0.0000
##        Balanced Accuracy : 0.5000
##
##         'Positive' Class : Yes
##
```

```
roc.ridge = roc(test.data$stroke, ridge.pred[,2])
```

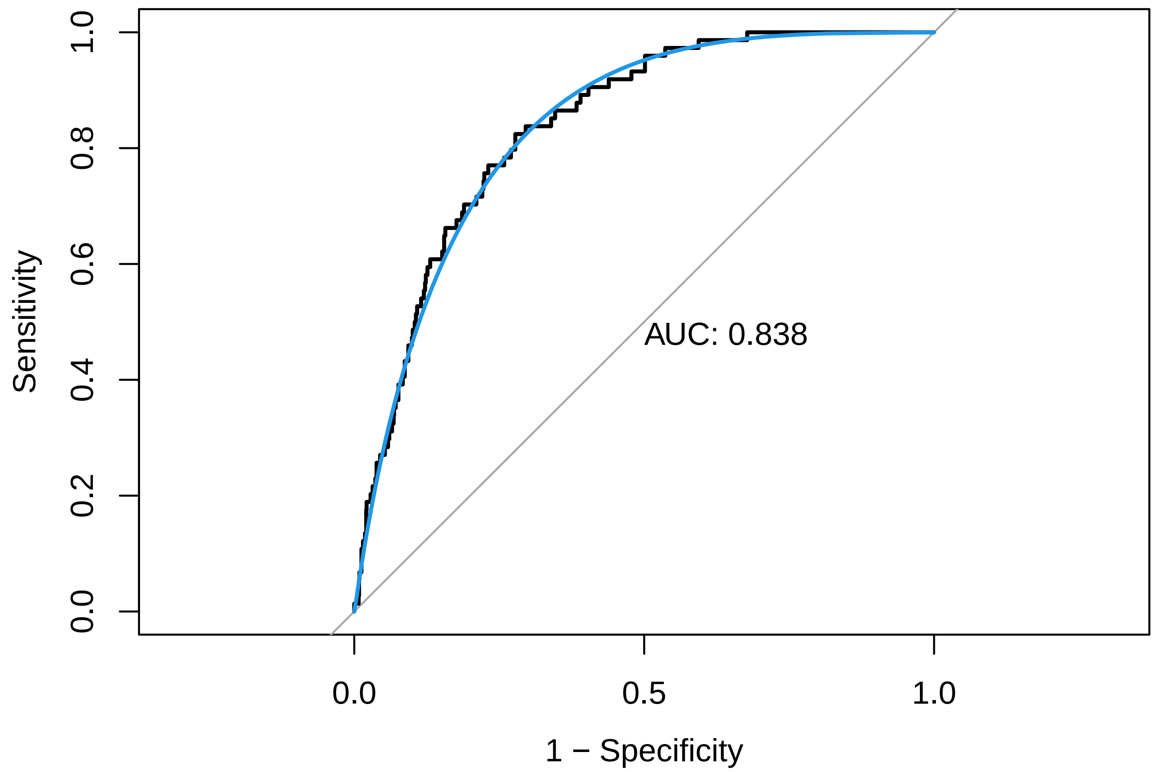
```
## Setting levels: control = No, case = Yes
```

```
## Setting direction: controls < cases
```

```
auc.ridge = roc.ridge$auc[1]
auc.ridge
```

```
## [1] 0.8384681
```

```
plot(roc.ridge, legacy.axes = TRUE, print.auc = TRUE)
plot(smooth(roc.ridge), col = 4, add = TRUE)
```




### LDA If we want to use LDA we have to make the assumption that the predictors have Gaussian-alike distribution.

```
set.seed(123)
lda.fit = train(   x = train.data[,c(1:10)],
                   y = train.data$stroke,
                   method = "lda",
                   metric = "ROC",
                   trControl = ctrl)

lda.fit1 = train(   x = train.data[,c(1:10)],
                   y = train.data$stroke,
                   method = "lda",
                   metric = "ROC",
                   trControl = ctrl1)

lda.pred =
    predict(lda.fit, newdata = test.data, type = "prob")

lda.pred
```

```
##              No         Yes
## 3     0.6941017 0.305898333
## 4     0.9749334 0.025066623
## 11    0.7788112 0.221188829
## 12    0.8548643 0.145135651
```

```
## 15   0.5219243 0.478075719
## 16   0.9573593 0.042640739
## 21   0.8068017 0.193198267
## 25   0.9214322 0.078567771
## 32   0.9884552 0.011544794
## 33   0.6041185 0.395881506
## 37   0.8572564 0.142743564
## 40   0.9860736 0.013926358
## 46   0.6756568 0.324343195
## 49   0.8862736 0.113726367
## 53   0.6583932 0.341606790
## 60   0.8902666 0.109733421
## 67   0.9067830 0.093217041
## 68   0.8530151 0.146984927
## 69   0.9696920 0.030307997
## 72   0.8424262 0.157573823
## 74   0.9086190 0.091381013
## 83   0.7120203 0.287979698
## 84   0.7816196 0.218380395
## 85   0.9718163 0.028183744
## 86   0.9754253 0.024574678
## 88   0.8646973 0.135302663
## 91   0.5126721 0.487327944
## 100  0.9127399 0.087260074
## 101  0.8182436 0.181756420
## 106  0.9683164 0.031683595
## 116  0.2145450 0.785454973
## 122  0.9587116 0.041288362
## 123  0.7255176 0.274482378
## 124  0.8956673 0.104332671
## 130  0.8753368 0.124663210
## 131  0.7919669 0.208033150
## 132  0.4688704 0.531129598
## 133  0.6578244 0.342175641
## 140  0.8179667 0.182033327
## 143  0.8243166 0.175683410
## 144  0.5712073 0.428792702
## 145  0.7296912 0.270308770
## 147  0.9651158 0.034884199
## 152  0.7188059 0.281194143
## 153  0.9068290 0.093171030
## 161  0.9146979 0.085302134
## 165  0.9041737 0.095826279
## 174  0.8282982 0.171701847
## 175  0.9265321 0.073467862
## 182  0.9666924 0.033307587
## 183  0.9953590 0.004640986
## 191  0.9393479 0.060652142
## 193  0.9716322 0.028367823
## 197  0.8185583 0.181441746
## 199  0.8708179 0.129182115
## 201  0.8493886 0.150611366
## 205  0.9689974 0.031002574
## 206  0.8908870 0.109112980
```

```
## 208  0.7606217 0.239378271
## 211  0.9913342 0.008665814
## 212  0.9732851 0.026714873
## 213  0.9186343 0.081365697
## 216  0.2998877 0.700112261
## 221  0.2580618 0.741938176
## 222  0.8928711 0.107128947
## 226  0.8261431 0.173856908
## 228  0.9273631 0.072636914
## 229  0.9921973 0.007802717
## 234  0.7219963 0.278003717
## 236  0.9028495 0.097150509
## 237  0.9697677 0.030232306
## 240  0.9047631 0.095236945
## 247  0.9324235 0.067576460
## 248  0.8826825 0.117317455
## 255  0.9799230 0.020076983
## 257  0.5999733 0.400026697
## 258  0.9955803 0.004419693
## 263  0.9937009 0.006299106
## 265  0.9857151 0.014284881
## 271  0.9658491 0.034150912
## 272  0.9867899 0.013210118
## 273  0.8703044 0.129695596
## 277  0.9784280 0.021571962
## 278  0.9545975 0.045402519
## 287  0.9926730 0.007326996
## 296  0.9774830 0.022516953
## 297  0.9331762 0.066823772
## 299  0.9128341 0.087165942
## 300  0.9666632 0.033336804
## 302  0.7573869 0.242613108
## 306  0.9844643 0.015535697
## 311  0.9862204 0.013779590
## 313  0.9954877 0.004512293
## 315  0.6677108 0.332289194
## 322  0.9927353 0.007264660
## 330  0.9952680 0.004732021
## 333  0.9877339 0.012266094
## 334  0.9898997 0.010100266
## 335  0.9832440 0.016755977
## 337  0.9821080 0.017891965
## 340  0.9739477 0.026052326
## 342  0.9962658 0.003734179
## 343  0.9812613 0.018738709
## 347  0.9293622 0.070637817
## 350  0.9222082 0.077791811
## 354  0.9669729 0.033027062
## 355  0.9936642 0.006335826
## 359  0.9929986 0.007001401
## 360  0.9960780 0.003921994
## 361  0.7635946 0.236405442
## 364  0.9968433 0.003156723
## 366  0.8765238 0.123476180
```

```
## 368  0.9347564 0.065243586
## 375  0.9748254 0.025174553
## 376  0.9692026 0.030797419
## 377  0.9961482 0.003851783
## 379  0.9922555 0.007744489
## 384  0.9950031 0.004996936
## 394  0.9872758 0.012724230
## 395  0.9937265 0.006273476
## 398  0.9421986 0.057801440
## 405  0.9609190 0.039081021
## 406  0.6411573 0.358842713
## 407  0.9761723 0.023827676
## 417  0.9957498 0.004250234
## 419  0.9933783 0.006621741
## 420  0.9910296 0.008970436
## 423  0.9589307 0.041069262
## 425  0.9332012 0.066798811
## 426  0.9946039 0.005396061
## 435  0.9366854 0.063314577
## 437  0.9490163 0.050983726
## 438  0.9959783 0.004021735
## 439  0.9767790 0.023221003
## 445  0.8371425 0.162857467
## 448  0.9642821 0.035717936
## 454  0.9697931 0.030206948
## 457  0.4043744 0.595625596
## 459  0.9856302 0.014369792
## 468  0.9027982 0.097201840
## 470  0.9281573 0.071842707
## 474  0.9862649 0.013735100
## 475  0.9844304 0.015569608
## 476  0.7535493 0.246450690
## 477  0.9888632 0.011136754
## 478  0.9138134 0.086186562
## 479  0.9210741 0.078925907
## 482  0.9806852 0.019314777
## 484  0.9936338 0.006366235
## 485  0.9842278 0.015772228
## 486  0.9951520 0.004847986
## 493  0.8974744 0.102525556
## 495  0.9803905 0.019609508
## 497  0.9960524 0.003947622
## 503  0.9688665 0.031133508
## 506  0.9635182 0.036481761
## 510  0.9934923 0.006507695
## 514  0.9897371 0.010262935
## 522  0.9539252 0.046074793
## 523  0.9907922 0.009207815
## 527  0.9726144 0.027385609
## 534  0.9809998 0.019000174
## 537  0.9691452 0.030854837
## 547  0.9915252 0.008474766
## 549  0.9753553 0.024644721
## 554  0.9809814 0.019018588
```

```
## 557  0.9834286 0.016571398
## 561  0.9921420 0.007857971
## 562  0.7881580 0.211841957
## 570  0.9922354 0.007764633
## 573  0.9771730 0.022826973
## 574  0.9937322 0.006267756
## 579  0.9944996 0.005500400
## 581  0.9792095 0.020790456
## 583  0.9962707 0.003729344
## 585  0.9924934 0.007506595
## 588  0.9798572 0.020142796
## 591  0.9863102 0.013689835
## 592  0.9848194 0.015180644
## 594  0.9958646 0.004135356
## 597  0.9830130 0.016986963
## 598  0.9968338 0.003166200
## 605  0.9942408 0.005759158
## 609  0.9878265 0.012173528
## 616  0.9895464 0.010453640
## 630  0.9826694 0.017330640
## 634  0.9832068 0.016793229
## 638  0.9842721 0.015727885
## 640  0.9832866 0.016713351
## 642  0.9921921 0.007807940
## 646  0.9807425 0.019257486
## 648  0.9362742 0.063725762
## 651  0.9760432 0.023956798
## 652  0.8403904 0.159609645
## 653  0.9832217 0.016778273
## 654  0.8809859 0.119014117
## 659  0.9863580 0.013641952
## 660  0.9139583 0.086041683
## 661  0.9569006 0.043099438
## 665  0.9744282 0.025571750
## 666  0.8898805 0.110119493
## 673  0.9977241 0.002275914
## 674  0.9611120 0.038887975
## 675  0.9933692 0.006630831
## 679  0.9971204 0.002879626
## 686  0.9860307 0.013969252
## 687  0.9867752 0.013224786
## 688  0.9915227 0.008477331
## 689  0.9661183 0.033881699
## 695  0.9866266 0.013373436
## 696  0.9955427 0.004457250
## 699  0.9948630 0.005137024
## 701  0.7781520 0.221847956
## 703  0.9869621 0.013037856
## 710  0.9937792 0.006220750
## 712  0.9780205 0.021979505
## 714  0.8571079 0.142892120
## 717  0.9913819 0.008618075
## 724  0.9812979 0.018702134
## 725  0.9365960 0.063404019
```

```
## 727  0.9859072 0.014092802
## 729  0.9945395 0.005460527
## 732  0.9833082 0.016691826
## 735  0.9963760 0.003624030
## 744  0.9913560 0.008644007
## 748  0.9495548 0.050445240
## 750  0.9717687 0.028231317
## 756  0.8797577 0.120242306
## 757  0.9920373 0.007962679
## 758  0.9951723 0.004827726
## 759  0.9798245 0.020175485
## 760  0.8819265 0.118073536
## 763  0.9913669 0.008633099
## 764  0.9984247 0.001575278
## 765  0.9940655 0.005934467
## 768  0.8255921 0.174407910
## 769  0.9952174 0.004782619
## 770  0.9865126 0.013487446
## 780  0.9933595 0.006640458
## 783  0.9877127 0.012287348
## 787  0.9322340 0.067765978
## 789  0.9304357 0.069564342
## 795  0.9885173 0.011482673
## 796  0.9920495 0.007950471
## 799  0.9884702 0.011529806
## 803  0.8926316 0.107368355
## 805  0.9882192 0.011780785
## 806  0.9906502 0.009349790
## 808  0.9097626 0.090237354
## 809  0.9802614 0.019738635
## 811  0.9751230 0.024876966
## 812  0.9899163 0.010083664
## 814  0.9587613 0.041238660
## 815  0.9941877 0.005812293
## 817  0.9510267 0.048973329
## 818  0.9939364 0.006063615
## 820  0.9699098 0.030090198
## 822  0.8660205 0.133979504
## 824  0.9052200 0.094780041
## 825  0.9900523 0.009947728
## 828  0.9926399 0.007360070
## 833  0.9929052 0.007094754
## 838  0.9657132 0.034286779
## 840  0.9976365 0.002363505
## 841  0.9938302 0.006169802
## 842  0.9545401 0.045459929
## 843  0.9943730 0.005626977
## 844  0.9948698 0.005130192
## 845  0.9965776 0.003422381
## 846  0.9873638 0.012636206
## 847  0.8358878 0.164112181
## 849  0.9390876 0.060912375
## 853  0.9958758 0.004124245
## 856  0.9946177 0.005382305
```

```
## 857  0.9969311 0.003068857
## 858  0.9919186 0.008081364
## 860  0.9868213 0.013178739
## 865  0.9586904 0.041309570
## 868  0.6404603 0.359539699
## 871  0.9840362 0.015963756
## 874  0.9972269 0.002773125
## 877  0.9920433 0.007956668
## 879  0.9904967 0.009503280
## 880  0.9356473 0.064352717
## 881  0.9203924 0.079607576
## 886  0.9917102 0.008289807
## 889  0.9264050 0.073595008
## 890  0.9112656 0.088734442
## 894  0.9961490 0.003851000
## 897  0.9966622 0.003337774
## 899  0.9724009 0.027599148
## 901  0.8215309 0.178469110
## 902  0.9867512 0.013248771
## 903  0.9867945 0.013205456
## 906  0.9964406 0.003559421
## 928  0.9410382 0.058961850
## 930  0.6640958 0.335904164
## 934  0.9064155 0.093584525
## 936  0.9816994 0.018300641
## 940  0.9935175 0.006482467
## 944  0.9310033 0.068996714
## 947  0.9811336 0.018866423
## 949  0.9650873 0.034912730
## 958  0.9957758 0.004224219
## 965  0.9158988 0.084101165
## 971  0.4987610 0.501238986
## 973  0.9946625 0.005337476
## 983  0.9946620 0.005338002
## 985  0.9977110 0.002288988
## 987  0.9938482 0.006151782
## 993  0.7256658 0.274334188
## 998  0.9815580 0.018442049
## 1000 0.9956540 0.004346013
## 1002 0.9917627 0.008237327
## 1012 0.8702449 0.129755126
## 1017 0.9855843 0.014415662
## 1019 0.9593883 0.040611707
## 1020 0.9386040 0.061395968
## 1024 0.9433373 0.056662698
## 1025 0.9929811 0.007018893
## 1030 0.8780052 0.121994802
## 1033 0.9944726 0.005527386
## 1036 0.9779492 0.022050825
## 1041 0.9966316 0.003368423
## 1044 0.9615733 0.038426704
## 1047 0.9866054 0.013394646
## 1049 0.9246824 0.075317571
## 1050 0.9884646 0.011535394
```

```
## 1055 0.8553976 0.144602376
## 1057 0.9935755 0.006424492
## 1060 0.9439510 0.056049017
## 1072 0.9778943 0.022105659
## 1073 0.9280976 0.071902439
## 1075 0.9928931 0.007106910
## 1082 0.9956620 0.004338008
## 1087 0.9612537 0.038746325
## 1093 0.8969483 0.103051674
## 1097 0.9922996 0.007700416
## 1098 0.9626417 0.037358285
## 1101 0.9599508 0.040049191
## 1102 0.9967056 0.003294440
## 1106 0.9837867 0.016213272
## 1112 0.9938427 0.006157328
## 1113 0.9795177 0.020482335
## 1117 0.9224580 0.077542031
## 1123 0.9949100 0.005090030
## 1125 0.9902578 0.009742213
## 1129 0.9732711 0.026728943
## 1137 0.9321606 0.067839357
## 1140 0.9959804 0.004019594
## 1141 0.9392831 0.060716926
## 1144 0.9961803 0.003819697
## 1146 0.9939473 0.006052701
## 1151 0.9961983 0.003801714
## 1157 0.9872081 0.012791897
## 1158 0.9447401 0.055259857
## 1160 0.9952237 0.004776334
## 1167 0.8691314 0.130868581
## 1170 0.9974186 0.002581398
## 1171 0.8702621 0.129737880
## 1173 0.9731501 0.026849879
## 1176 0.9922329 0.007767123
## 1183 0.9953564 0.004643570
## 1184 0.8471217 0.152878345
## 1186 0.9946833 0.005316748
## 1189 0.9254368 0.074563185
## 1190 0.5087948 0.491205212
## 1193 0.9935058 0.006494218
## 1196 0.9951427 0.004857291
## 1201 0.8406215 0.159378526
## 1206 0.9937646 0.006235367
## 1210 0.9933148 0.006685177
## 1211 0.9585496 0.041450410
## 1212 0.8676455 0.132354511
## 1217 0.9556022 0.044397785
## 1220 0.9813785 0.018621483
## 1222 0.9923797 0.007620286
## 1243 0.9964754 0.003524600
## 1244 0.9959898 0.004010181
## 1245 0.9968940 0.003105969
## 1247 0.9962395 0.003760497
## 1249 0.9834835 0.016516547
```

```
## 1258 0.9764005 0.023599452
## 1261 0.9917293 0.008270676
## 1262 0.9669538 0.033046162
## 1265 0.9950957 0.004904275
## 1267 0.9643110 0.035689024
## 1269 0.9954819 0.004518131
## 1270 0.9732246 0.026775367
## 1272 0.9833045 0.016695524
## 1276 0.9949978 0.005002229
## 1280 0.8488762 0.151123802
## 1284 0.7870694 0.212930564
## 1287 0.6761030 0.323896956
## 1288 0.9940865 0.005913494
## 1294 0.9748782 0.025121762
## 1295 0.8373500 0.162650002
## 1300 0.9851147 0.014885340
## 1302 0.9931930 0.006806966
## 1303 0.9364128 0.063587208
## 1308 0.9908238 0.009176184
## 1312 0.9663394 0.033660605
## 1316 0.9727612 0.027238812
## 1326 0.8538381 0.146161934
## 1333 0.9764551 0.023544911
## 1334 0.9349624 0.065037594
## 1335 0.9966383 0.003361703
## 1339 0.9947161 0.005283870
## 1341 0.9820529 0.017947101
## 1344 0.7795943 0.220405742
## 1346 0.9804175 0.019582462
## 1349 0.9101618 0.089838248
## 1351 0.7419646 0.258035383
## 1353 0.9414917 0.058508344
## 1361 0.9731364 0.026863615
## 1364 0.9560908 0.043909191
## 1373 0.9883775 0.011622478
## 1374 0.9636714 0.036328610
## 1379 0.9866092 0.013390845
## 1381 0.9915829 0.008417128
## 1387 0.9940448 0.005955213
## 1392 0.9207756 0.079224424
## 1394 0.9756656 0.024334413
## 1396 0.9768573 0.023142745
## 1397 0.9930619 0.006938062
## 1399 0.7745426 0.225457427
## 1408 0.8880474 0.111952630
## 1421 0.9937216 0.006278421
## 1423 0.9939606 0.006039361
## 1424 0.9497586 0.050241387
## 1426 0.9964736 0.003526357
## 1430 0.9916232 0.008376770
## 1431 0.9961151 0.003884873
## 1436 0.9270593 0.072940745
## 1449 0.8107874 0.189212566
## 1455 0.9542418 0.045758165
```

```
## 1458 0.9828122 0.017187769
## 1462 0.9928955 0.007104547
## 1465 0.9940266 0.005973371
## 1467 0.9649007 0.035099338
## 1469 0.9800273 0.019972670
## 1472 0.9940522 0.005947849
## 1474 0.9543418 0.045658224
## 1477 0.9969812 0.003018817
## 1481 0.9067520 0.093247961
## 1487 0.9949714 0.005028646
## 1489 0.9932959 0.006704051
## 1495 0.9932872 0.006712757
## 1497 0.9291431 0.070856904
## 1498 0.9337619 0.066238116
## 1499 0.9816957 0.018304263
## 1512 0.6346530 0.365347049
## 1516 0.8130381 0.186961933
## 1518 0.8504073 0.149592700
## 1519 0.9859201 0.014079943
## 1520 0.9896414 0.010358571
## 1531 0.9786827 0.021317290
## 1533 0.9842993 0.015700670
## 1536 0.9589502 0.041049750
## 1542 0.9754710 0.024529045
## 1546 0.9900939 0.009906127
## 1547 0.8910705 0.108929499
## 1551 0.9630055 0.036994519
## 1552 0.9885572 0.011442792
## 1554 0.8833540 0.116646016
## 1559 0.9922126 0.007787416
## 1560 0.9914818 0.008518238
## 1567 0.7492621 0.250737888
## 1569 0.9521717 0.047828271
## 1570 0.9797986 0.020201398
## 1571 0.9874129 0.012587149
## 1572 0.9554361 0.044563861
## 1573 0.9926577 0.007342299
## 1578 0.9765620 0.023438017
## 1579 0.9851184 0.014881605
## 1585 0.9971674 0.002832648
## 1587 0.9838542 0.016145824
## 1589 0.6615715 0.338428541
## 1592 0.9972042 0.002795844
## 1595 0.9610031 0.038996887
## 1597 0.9123689 0.087631130
## 1600 0.9308346 0.069165431
## 1610 0.9945393 0.005460696
## 1623 0.9916119 0.008388127
## 1624 0.9963964 0.003603633
## 1625 0.9819004 0.018099622
## 1627 0.9802352 0.019764774
## 1631 0.8490172 0.150982779
## 1632 0.9943029 0.005697129
## 1637 0.9964316 0.003568395
```

```
## 1638 0.9551592 0.044840841
## 1639 0.9477338 0.052266237
## 1640 0.9936206 0.006379430
## 1642 0.9957457 0.004254270
## 1645 0.9895113 0.010488684
## 1650 0.9851581 0.014841934
## 1656 0.9559756 0.044024421
## 1657 0.9842402 0.015759784
## 1658 0.9744281 0.025571874
## 1662 0.9536254 0.046374627
## 1663 0.9793580 0.020642040
## 1667 0.9961271 0.003872942
## 1670 0.9613334 0.038666634
## 1671 0.9601915 0.039808481
## 1683 0.9687652 0.031234785
## 1689 0.9966349 0.003365092
## 1690 0.9944367 0.005563303
## 1692 0.8221348 0.177865174
## 1700 0.9790529 0.020947071
## 1703 0.9961876 0.003812402
## 1706 0.9949058 0.005094201
## 1708 0.9907329 0.009267058
## 1709 0.9916449 0.008355122
## 1710 0.9930418 0.006958158
## 1711 0.9880355 0.011964476
## 1712 0.9280108 0.071989235
## 1715 0.9952519 0.004748108
## 1719 0.8163097 0.183690326
## 1722 0.9690993 0.030900698
## 1724 0.9296487 0.070351290
## 1725 0.9934825 0.006517461
## 1726 0.9928134 0.007186565
## 1727 0.9949721 0.005027858
## 1728 0.9617857 0.038214289
## 1733 0.8919403 0.108059676
## 1736 0.9559644 0.044035558
## 1740 0.9954513 0.004548726
## 1744 0.9923503 0.007649737
## 1745 0.9932826 0.006717411
## 1746 0.9938301 0.006169897
## 1749 0.9539713 0.046028733
## 1752 0.9524243 0.047575699
## 1753 0.5786972 0.421302769
## 1754 0.9620588 0.037941188
## 1756 0.9516533 0.048346701
## 1757 0.9799779 0.020022090
## 1758 0.9946088 0.005391191
## 1760 0.9790795 0.020920491
## 1763 0.9811123 0.018887653
## 1764 0.9954373 0.004562731
## 1769 0.9920843 0.007915690
## 1772 0.9864978 0.013502218
## 1773 0.9946557 0.005344313
## 1774 0.9932547 0.006745271
```

```
## 1777 0.8076986 0.192301383
## 1778 0.9953247 0.004675324
## 1779 0.6744109 0.325589067
## 1782 0.9929818 0.007018225
## 1784 0.9964131 0.003586910
## 1789 0.9732300 0.026769965
## 1795 0.9963287 0.003671319
## 1799 0.9641280 0.035872008
## 1804 0.9929974 0.007002571
## 1807 0.9944828 0.005517248
## 1809 0.9966576 0.003342440
## 1812 0.9909668 0.009033155
## 1816 0.9608815 0.039118543
## 1819 0.9208969 0.079103112
## 1825 0.9935203 0.006479680
## 1826 0.9895322 0.010467815
## 1827 0.9949421 0.005057858
## 1832 0.9941656 0.005834352
## 1839 0.9892798 0.010720198
## 1840 0.7446914 0.255308591
## 1841 0.9041337 0.095866313
## 1844 0.8006960 0.199303981
## 1845 0.9910335 0.008966470
## 1850 0.9950181 0.004981949
## 1851 0.9048278 0.095172163
## 1852 0.9941171 0.005882905
## 1854 0.9316859 0.068314085
## 1865 0.9682341 0.031765893
## 1872 0.9188761 0.081123885
## 1873 0.9769404 0.023059561
## 1874 0.9557198 0.044280181
## 1875 0.9859014 0.014098607
## 1877 0.9831444 0.016855575
## 1883 0.9958064 0.004193611
## 1891 0.9810964 0.018903560
## 1898 0.9901828 0.009817230
## 1903 0.9939455 0.006054528
## 1908 0.9854658 0.014534182
## 1909 0.9837129 0.016287125
## 1914 0.9849617 0.015038310
## 1919 0.7614421 0.238557903
## 1920 0.9951497 0.004850305
## 1925 0.9736372 0.026362847
## 1926 0.9909305 0.009069461
## 1930 0.9961212 0.003878770
## 1936 0.9729024 0.027097630
## 1937 0.9921218 0.007878154
## 1943 0.9944982 0.005501769
## 1944 0.9888323 0.011167667
## 1945 0.9512228 0.048777212
## 1948 0.9939642 0.006035784
## 1953 0.9394274 0.060572633
## 1964 0.9832674 0.016732645
## 1967 0.9911338 0.008866162
```

```
## 1968 0.9103335 0.089666545
## 1977 0.9967417 0.003258325
## 1980 0.9647142 0.035285838
## 1991 0.9951436 0.004856427
## 1994 0.9163948 0.083605160
## 1998 0.9941891 0.005810897
## 2003 0.9060569 0.093943058
## 2005 0.9687098 0.031290209
## 2006 0.9936123 0.006387654
## 2008 0.9606259 0.039374135
## 2012 0.9513522 0.048647815
## 2013 0.9962723 0.003727726
## 2015 0.9573807 0.042619260
## 2018 0.9812791 0.018720864
## 2020 0.9936958 0.006304199
## 2021 0.9946309 0.005369110
## 2024 0.9850852 0.014914841
## 2027 0.9950686 0.004931408
## 2029 0.9909793 0.009020669
## 2035 0.9390170 0.060983022
## 2037 0.9932232 0.006776798
## 2049 0.9774701 0.022529899
## 2050 0.9768476 0.023152356
## 2060 0.8733575 0.126642546
## 2062 0.8576548 0.142345194
## 2065 0.9231899 0.076810097
## 2066 0.9939999 0.006000098
## 2067 0.9957232 0.004276754
## 2068 0.9916399 0.008360113
## 2074 0.9942009 0.005799122
## 2076 0.9901245 0.009875468
## 2083 0.8643861 0.135613859
## 2086 0.9457084 0.054291553
## 2089 0.8442670 0.155732969
## 2092 0.9931275 0.006872543
## 2097 0.9032396 0.096760424
## 2107 0.9960928 0.003907191
## 2115 0.9947873 0.005212750
## 2116 0.9799104 0.020089564
## 2117 0.9955695 0.004430499
## 2119 0.9407913 0.059208666
## 2120 0.9921042 0.007895772
## 2121 0.9325944 0.067405568
## 2123 0.7720464 0.227953601
## 2128 0.9905933 0.009406704
## 2139 0.9160325 0.083967471
## 2140 0.9938153 0.006184695
## 2142 0.9963890 0.003610988
## 2146 0.9838483 0.016151662
## 2150 0.9860660 0.013933988
## 2151 0.9882722 0.011727798
## 2152 0.9963450 0.003654997
## 2159 0.8881858 0.111814213
## 2162 0.8806644 0.119335596
```

```
## 2167 0.9930690 0.006931018
## 2168 0.9926427 0.007357278
## 2171 0.8681433 0.131856704
## 2173 0.9255090 0.074491006
## 2182 0.9949888 0.005011160
## 2187 0.9928004 0.007199590
## 2189 0.6507716 0.349228413
## 2195 0.9951794 0.004820565
## 2196 0.9929200 0.007080034
## 2199 0.9818048 0.018195186
## 2200 0.9198985 0.080101521
## 2208 0.9922047 0.007795309
## 2216 0.8356872 0.164312780
## 2217 0.9885862 0.011413785
## 2220 0.9916532 0.008346790
## 2223 0.9949953 0.005004676
## 2224 0.9770739 0.022926144
## 2226 0.9264111 0.073588899
## 2227 0.9667958 0.033204243
## 2228 0.9555640 0.044436019
## 2232 0.9598469 0.040153115
## 2234 0.8892912 0.110708765
## 2235 0.9905139 0.009486145
## 2238 0.9941430 0.005857014
## 2240 0.9881902 0.011809770
## 2244 0.9856529 0.014347125
## 2247 0.8935809 0.106419053
## 2249 0.9946567 0.005343308
## 2252 0.9770183 0.022981737
## 2256 0.8594301 0.140569928
## 2258 0.8327377 0.167262317
## 2259 0.9933180 0.006682011
## 2260 0.9919038 0.008096246
## 2263 0.9836839 0.016316091
## 2266 0.9571504 0.042849637
## 2267 0.9707475 0.029252539
## 2268 0.9748793 0.025120694
## 2269 0.9942604 0.005739591
## 2272 0.9912022 0.008797831
## 2278 0.9968403 0.003159681
## 2281 0.9324944 0.067505609
## 2288 0.9104939 0.089506079
## 2289 0.9941072 0.005892801
## 2297 0.9842107 0.015789314
## 2299 0.9550193 0.044980682
## 2300 0.9959776 0.004022398
## 2302 0.9267551 0.073244918
## 2303 0.9943838 0.005616168
## 2305 0.9952715 0.004728526
## 2307 0.9868010 0.013199023
## 2310 0.9851939 0.014806090
## 2313 0.9090489 0.090951113
## 2315 0.8308217 0.169178327
## 2319 0.9442256 0.055774377
```

```
## 2322 0.6356359 0.364364148
## 2324 0.9951465 0.004853537
## 2328 0.9519545 0.048045531
## 2332 0.9939269 0.006073056
## 2339 0.9604354 0.039564583
## 2340 0.9852002 0.014799812
## 2341 0.9939326 0.006067378
## 2344 0.8457217 0.154278279
## 2346 0.9854754 0.014524609
## 2347 0.9922237 0.007776341
## 2354 0.9934776 0.006522436
## 2359 0.9955800 0.004419965
## 2363 0.6757496 0.324250399
## 2365 0.9939939 0.006006141
## 2373 0.9935874 0.006412556
## 2375 0.8929480 0.107052046
## 2377 0.9756021 0.024397948
## 2382 0.9971535 0.002846506
## 2383 0.9966824 0.003317559
## 2397 0.9869082 0.013091821
## 2398 0.9955046 0.004495444
## 2401 0.9941464 0.005853585
## 2406 0.8591223 0.140877690
## 2407 0.9699768 0.030023231
## 2408 0.9680964 0.031903565
## 2410 0.9852255 0.014774515
## 2416 0.9941193 0.005880693
## 2423 0.9193290 0.080670959
## 2426 0.9920871 0.007912872
## 2428 0.9714860 0.028513962
## 2431 0.8280975 0.171902486
## 2434 0.3347467 0.665253314
## 2436 0.9890964 0.010903576
## 2437 0.9941008 0.005899242
## 2438 0.9930765 0.006923507
## 2443 0.9943231 0.005676863
## 2445 0.8713489 0.128651111
## 2446 0.9863741 0.013625908
## 2456 0.9899721 0.010027945
## 2463 0.7127699 0.287230135
## 2465 0.9926344 0.007365608
## 2467 0.8963652 0.103634782
## 2469 0.8954224 0.104577609
## 2470 0.7875114 0.212488558
## 2472 0.9940758 0.005924177
## 2479 0.9909646 0.009035433
## 2481 0.9915465 0.008453504
## 2483 0.9958858 0.004114151
## 2484 0.9823017 0.017698304
## 2490 0.9942861 0.005713852
## 2496 0.9834056 0.016594395
## 2497 0.9755953 0.024404746
## 2502 0.9819087 0.018091278
## 2503 0.7920781 0.207921868
```

```
## 2508 0.9213218 0.078678211
## 2511 0.9895838 0.010416241
## 2513 0.9654156 0.034584372
## 2520 0.8153907 0.184609340
## 2526 0.9934969 0.006503119
## 2530 0.8563085 0.143691450
## 2531 0.9899014 0.010098586
## 2532 0.9962637 0.003736289
## 2537 0.9946350 0.005365032
## 2538 0.9534689 0.046531134
## 2542 0.7155536 0.284446413
## 2546 0.9683444 0.031655562
## 2547 0.9819463 0.018053747
## 2554 0.8095567 0.190443284
## 2557 0.9688278 0.031172223
## 2559 0.9953986 0.004601432
## 2565 0.9882370 0.011763013
## 2567 0.9939207 0.006079314
## 2569 0.8945524 0.105447644
## 2572 0.9861660 0.013833959
## 2573 0.8803633 0.119636690
## 2575 0.9911674 0.008832556
## 2576 0.9928329 0.007167142
## 2582 0.9932789 0.006721056
## 2589 0.9950350 0.004964996
## 2598 0.9938990 0.006101002
## 2601 0.9707474 0.029252592
## 2602 0.8770472 0.122952773
## 2610 0.9133860 0.086613995
## 2611 0.9357591 0.064240850
## 2613 0.9376747 0.062325267
## 2617 0.9810156 0.018984351
## 2623 0.9904055 0.009594480
## 2624 0.9601999 0.039800070
## 2626 0.9943821 0.005617896
## 2631 0.9954236 0.004576353
## 2641 0.9826651 0.017334924
## 2642 0.9510423 0.048957717
## 2643 0.9606563 0.039343740
## 2647 0.9939162 0.006083753
## 2651 0.9312533 0.068746746
## 2652 0.9470904 0.052909602
## 2657 0.9675105 0.032489517
## 2658 0.9888904 0.011109555
## 2660 0.9955132 0.004486830
## 2663 0.9927520 0.007247966
## 2665 0.9870217 0.012978311
## 2667 0.9048702 0.095129779
## 2668 0.9963725 0.003627525
## 2673 0.9902737 0.009726331
## 2675 0.9779229 0.022077104
## 2676 0.7433917 0.256608318
## 2678 0.9242313 0.075768746
## 2679 0.9868758 0.013124167
```

```
## 2685 0.9739001 0.026099856
## 2692 0.9665671 0.033432910
## 2694 0.7791424 0.220857621
## 2696 0.9813241 0.018675862
## 2697 0.9755869 0.024413096
## 2701 0.9762478 0.023752221
## 2705 0.9930863 0.006913714
## 2707 0.9725294 0.027470557
## 2711 0.8740948 0.125905224
## 2713 0.9915896 0.008410447
## 2715 0.9928533 0.007146715
## 2716 0.8186998 0.181300247
## 2720 0.9495158 0.050484209
## 2725 0.9765686 0.023431391
## 2730 0.9888000 0.011199951
## 2731 0.9907079 0.009292090
## 2735 0.9120107 0.087989307
## 2736 0.9861818 0.013818166
## 2739 0.9877975 0.012202539
## 2740 0.9952043 0.004795684
## 2746 0.9939300 0.006069966
## 2747 0.9554158 0.044584166
## 2748 0.9916414 0.008358609
## 2750 0.9847722 0.015227825
## 2758 0.8778558 0.122144175
## 2764 0.9892810 0.010718966
## 2765 0.9987329 0.001267133
## 2766 0.7854790 0.214520956
## 2768 0.9963124 0.003687558
## 2773 0.9805753 0.019424732
## 2775 0.9807752 0.019224791
## 2778 0.9592997 0.040700260
## 2780 0.9951581 0.004841879
## 2782 0.9943079 0.005692128
## 2784 0.9904188 0.009581242
## 2791 0.9938366 0.006163382
## 2793 0.9814285 0.018571529
## 2797 0.9484294 0.051570582
## 2798 0.9121077 0.087892258
## 2799 0.9576339 0.042366068
## 2803 0.9818108 0.018189211
## 2808 0.9850531 0.014946929
## 2809 0.9966091 0.003390940
## 2812 0.9938513 0.006148737
## 2814 0.9899407 0.010059300
## 2815 0.9630875 0.036912498
## 2817 0.9869063 0.013093694
## 2818 0.9657587 0.034241251
## 2823 0.9813545 0.018645475
## 2824 0.8633902 0.136609755
## 2827 0.8877203 0.112279678
## 2830 0.9848672 0.015132773
## 2831 0.9808417 0.019158264
## 2833 0.9932343 0.006765722
```

```
## 2843 0.9881978 0.011802215
## 2844 0.9920780 0.007922023
## 2851 0.9445428 0.055457159
## 2856 0.7127018 0.287298247
## 2859 0.9926391 0.007360877
## 2862 0.9960332 0.003966814
## 2864 0.9709374 0.029062563
## 2867 0.8346712 0.165328772
## 2872 0.9938600 0.006139962
## 2877 0.8811349 0.118865095
## 2892 0.7812016 0.218798429
## 2893 0.9919706 0.008029419
## 2894 0.9949095 0.005090476
## 2903 0.5704234 0.429576563
## 2906 0.9583986 0.041601385
## 2909 0.9418387 0.058161314
## 2915 0.9689646 0.031035412
## 2920 0.9919502 0.008049784
## 2921 0.9864041 0.013595882
## 2922 0.9945343 0.005465662
## 2930 0.9947745 0.005225495
## 2934 0.9821993 0.017800744
## 2935 0.7038684 0.296131557
## 2940 0.9879727 0.012027283
## 2945 0.9927381 0.007261905
## 2949 0.9453868 0.054613206
## 2950 0.9925533 0.007446739
## 2951 0.9944305 0.005569549
## 2952 0.9097851 0.090214897
## 2953 0.9730643 0.026935651
## 2956 0.9808431 0.019156872
## 2962 0.8828866 0.117113393
## 2964 0.9837269 0.016273123
## 2969 0.9606218 0.039378173
## 2974 0.9899774 0.010022561
## 2980 0.7550690 0.244930964
## 2984 0.9955071 0.004492881
## 2985 0.9938682 0.006131757
## 2986 0.7535476 0.246452357
## 2988 0.9946638 0.005336204
## 2995 0.9844306 0.015569422
## 2998 0.6976005 0.302399539
## 2999 0.9793482 0.020651838
## 3004 0.9837512 0.016248808
## 3005 0.9933840 0.006616010
## 3012 0.9172026 0.082797369
## 3017 0.9907175 0.009282539
## 3018 0.8887253 0.111274685
## 3021 0.9948445 0.005155468
## 3023 0.9952072 0.004792815
## 3024 0.9948640 0.005136017
## 3031 0.9799080 0.020092021
## 3034 0.8549819 0.145018137
## 3035 0.9916942 0.008305847
```

```
## 3040 0.9965782 0.003421789
## 3043 0.9955084 0.004491603
## 3045 0.9944034 0.005596599
## 3052 0.9953838 0.004616214
## 3054 0.9960512 0.003948775
## 3055 0.3904660 0.609533984
## 3056 0.9450144 0.054985616
## 3057 0.8547145 0.145285526
## 3062 0.9938856 0.006114376
## 3063 0.9743661 0.025633916
## 3066 0.9520103 0.047989667
## 3067 0.9747430 0.025256955
## 3068 0.9911221 0.008877936
## 3070 0.7046354 0.295364621
## 3073 0.9915980 0.008402024
## 3074 0.9761833 0.023816741
## 3077 0.9953700 0.004629972
## 3078 0.9717360 0.028263984
## 3079 0.9845697 0.015430310
## 3081 0.8972977 0.102702265
## 3085 0.9719364 0.028063632
## 3091 0.9674572 0.032542827
## 3096 0.9847931 0.015206861
## 3097 0.9935456 0.006454436
## 3100 0.9894574 0.010542610
## 3101 0.9727047 0.027295318
## 3102 0.9818789 0.018121121
## 3105 0.9904841 0.009515883
## 3106 0.9383364 0.061663645
## 3107 0.9826869 0.017313110
## 3109 0.6339518 0.366048195
## 3110 0.9796987 0.020301251
## 3115 0.9496034 0.050396561
## 3119 0.9905470 0.009453009
## 3122 0.9641482 0.035851780
## 3132 0.9938779 0.006122101
## 3133 0.9953811 0.004618898
## 3140 0.9796800 0.020320006
## 3142 0.9841040 0.015895976
## 3146 0.8662450 0.133755005
## 3151 0.9526846 0.047315366
## 3153 0.9676399 0.032360070
## 3161 0.9832425 0.016757531
## 3164 0.8494012 0.150598833
## 3171 0.9941603 0.005839687
## 3176 0.9921369 0.007863129
## 3178 0.9946860 0.005314012
## 3179 0.9962075 0.003792514
## 3181 0.9016058 0.098394195
## 3182 0.8377635 0.162236458
## 3192 0.5349594 0.465040581
## 3193 0.9879313 0.012068690
## 3206 0.9342998 0.065700207
## 3208 0.9947079 0.005292067
```

```
## 3212 0.9592205 0.040779544
## 3213 0.7323463 0.267653734
## 3220 0.9120895 0.087910500
## 3233 0.9961476 0.003852387
## 3245 0.8151612 0.184838803
## 3251 0.9957227 0.004277330
## 3252 0.9806599 0.019340091
## 3259 0.9961665 0.003833479
## 3260 0.8908616 0.109138356
## 3262 0.9829208 0.017079178
## 3265 0.9715251 0.028474920
## 3267 0.9726362 0.027363801
## 3272 0.9855640 0.014436001
## 3274 0.9066126 0.093387350
## 3277 0.9958965 0.004103502
## 3279 0.9941167 0.005883253
## 3280 0.9931045 0.006895486
## 3282 0.9949160 0.005083955
## 3283 0.9888452 0.011154774
## 3285 0.9584372 0.041562834
## 3288 0.9409017 0.059098258
## 3293 0.9731434 0.026856608
## 3303 0.9904733 0.009526736
## 3306 0.9058023 0.094197676
## 3307 0.8402405 0.159759458
## 3308 0.9405544 0.059445599
## 3315 0.9956196 0.004380389
## 3319 0.9932186 0.006781373
## 3321 0.9954927 0.004507326
## 3322 0.9209695 0.079030456
## 3323 0.9937858 0.006214236
## 3328 0.8311680 0.168832029
## 3336 0.9869985 0.013001536
## 3339 0.9542063 0.045793723
## 3342 0.9119679 0.088032060
## 3345 0.5724925 0.427507515
## 3347 0.7685030 0.231497002
## 3354 0.9929671 0.007032937
## 3357 0.9895636 0.010436405
## 3359 0.9925312 0.007468788
## 3365 0.9791384 0.020861579
## 3366 0.9973390 0.002661016
## 3368 0.9926507 0.007349300
## 3369 0.9421896 0.057810449
## 3385 0.9936867 0.006313341
## 3386 0.9894297 0.010570324
## 3391 0.9049612 0.095038790
## 3392 0.9958594 0.004140557
## 3400 0.5295250 0.470474994
## 3402 0.9934826 0.006517394
## 3403 0.9914635 0.008536488
## 3405 0.9875285 0.012471476
## 3406 0.9876516 0.012348407
## 3409 0.9877618 0.012238203
```

```
## 3415 0.9894903 0.010509721
## 3421 0.9763074 0.023692560
## 3423 0.9948718 0.005128212
## 3427 0.9551801 0.044819906
## 3428 0.9036392 0.096360794
## 3430 0.9901823 0.009817721
## 3434 0.9958905 0.004109538
## 3439 0.8832468 0.116753193
## 3440 0.9955277 0.004472299
## 3446 0.9920719 0.007928107
## 3452 0.9825899 0.017410087
## 3454 0.9617689 0.038231121
## 3463 0.9953210 0.004679000
## 3467 0.9969862 0.003013799
## 3470 0.9758264 0.024173562
## 3475 0.9810959 0.018904054
## 3476 0.9604682 0.039531799
## 3484 0.9909274 0.009072625
## 3487 0.9748612 0.025138818
## 3491 0.9923075 0.007692500
## 3494 0.9265057 0.073494263
## 3497 0.9903678 0.009632195
## 3499 0.9971470 0.002852982
## 3504 0.9427048 0.057295238
## 3514 0.9846767 0.015323349
## 3521 0.9785246 0.021475407
## 3522 0.7028724 0.297127577
## 3525 0.9656923 0.034307722
## 3527 0.9923128 0.007687182
## 3528 0.9577872 0.042212778
## 3529 0.9854302 0.014569791
## 3532 0.9962279 0.003772068
## 3534 0.9469701 0.053029874
## 3538 0.9848038 0.015196177
## 3539 0.9934378 0.006562181
## 3541 0.9932823 0.006717666
## 3542 0.9892582 0.010741790
## 3545 0.9917329 0.008267051
## 3547 0.9073863 0.092613669
## 3553 0.8137140 0.186285985
## 3556 0.9904509 0.009549143
## 3561 0.9910556 0.008944356
## 3566 0.9965634 0.003436647
## 3571 0.9698526 0.030147430
## 3579 0.8630886 0.136911388
## 3580 0.9939342 0.006065790
## 3592 0.9957499 0.004250078
## 3595 0.9715547 0.028445256
## 3596 0.9940766 0.005923447
## 3604 0.9957488 0.004251204
## 3606 0.9687529 0.031247120
## 3609 0.9952862 0.004713790
## 3617 0.9798809 0.020119098
## 3618 0.9955762 0.004423848
```

```
## 3631 0.9661872 0.033812803
## 3632 0.7947645 0.205235496
## 3633 0.9815730 0.018426990
## 3634 0.8677214 0.132278641
## 3635 0.9829468 0.017053197
## 3638 0.9886472 0.011352817
## 3646 0.9761091 0.023890899
## 3649 0.9359329 0.064067147
## 3651 0.9939680 0.006032017
## 3657 0.9952035 0.004796496
## 3660 0.9847517 0.015248340
## 3661 0.9926578 0.007342153
## 3663 0.9340165 0.065983493
## 3667 0.9915219 0.008478107
## 3678 0.9942711 0.005728896
## 3679 0.9877201 0.012279862
## 3681 0.9941531 0.005846943
## 3682 0.9424726 0.057527433
## 3691 0.9433154 0.056684558
## 3700 0.9566025 0.043397485
## 3702 0.9927784 0.007221581
## 3706 0.9943520 0.005648005
## 3707 0.9839330 0.016067045
## 3708 0.9921489 0.007851111
## 3712 0.9936272 0.006372769
## 3719 0.9910444 0.008955576
## 3720 0.9556658 0.044334184
## 3723 0.9533171 0.046682918
## 3729 0.9863007 0.013699289
## 3730 0.9643924 0.035607584
## 3731 0.9929618 0.007038163
## 3735 0.9425599 0.057440084
## 3741 0.9912536 0.008746437
## 3742 0.9136772 0.086322754
## 3746 0.9937580 0.006241998
## 3747 0.9968510 0.003149019
## 3752 0.9449573 0.055042656
## 3753 0.3785199 0.621480138
## 3757 0.9824194 0.017580648
## 3759 0.9506336 0.049366395
## 3760 0.9906550 0.009345003
## 3764 0.9821576 0.017842446
## 3766 0.9863804 0.013619591
## 3767 0.9889178 0.011082187
## 3768 0.9664756 0.033524430
## 3771 0.7505099 0.249490139
## 3772 0.8345044 0.165495553
## 3773 0.9873739 0.012626060
## 3775 0.9439409 0.056059096
## 3776 0.9915322 0.008467753
## 3781 0.5160836 0.483916383
## 3791 0.9860564 0.013943605
## 3794 0.8897175 0.110282499
## 3800 0.9937472 0.006252763
```

```
## 3802 0.9506436 0.049356409
## 3806 0.9936491 0.006350881
## 3807 0.9968768 0.003123157
## 3809 0.9948160 0.005184026
## 3812 0.9884364 0.011563568
## 3814 0.9964414 0.003558643
## 3815 0.9846761 0.015323883
## 3818 0.9810624 0.018937635
## 3821 0.9169612 0.083038812
## 3822 0.9954761 0.004523892
## 3824 0.9870670 0.012933030
## 3826 0.8708927 0.129107321
## 3829 0.9941238 0.005876188
## 3837 0.9931101 0.006889860
## 3838 0.9783862 0.021613807
## 3840 0.9962321 0.003767903
## 3846 0.9934371 0.006562852
## 3850 0.9924687 0.007531322
## 3852 0.9914757 0.008524273
## 3854 0.9114927 0.088507287
## 3857 0.9917811 0.008218863
## 3863 0.9211133 0.078886656
## 3867 0.9908728 0.009127211
## 3868 0.9962748 0.003725173
## 3869 0.9633454 0.036654609
## 3873 0.9906368 0.009363247
## 3876 0.9948253 0.005174717
## 3877 0.9925054 0.007494613
## 3880 0.9920125 0.007987531
## 3883 0.9963247 0.003675284
## 3886 0.9844631 0.015536948
## 3887 0.9935819 0.006418075
## 3889 0.8341104 0.165889575
## 3894 0.9963806 0.003619380
## 3895 0.9938031 0.006196915
## 3899 0.9448254 0.055174628
## 3909 0.9776320 0.022367974
## 3913 0.8143377 0.185662281
## 3916 0.9489421 0.051057931
## 3919 0.9370424 0.062957583
## 3929 0.9851035 0.014896541
## 3930 0.9802505 0.019749530
## 3934 0.9912853 0.008714652
## 3941 0.9950044 0.004995647
## 3944 0.9420655 0.057934524
## 3949 0.9963274 0.003672576
## 3951 0.7907735 0.209226481
## 3953 0.9949059 0.005094063
## 3957 0.9927020 0.007297989
## 3958 0.9931702 0.006829810
## 3963 0.9051389 0.094861122
## 3969 0.8893159 0.110684069
## 3972 0.9933121 0.006687946
## 3974 0.9655217 0.034478325
```

```
## 3975 0.9675612 0.032438761
## 3977 0.9930235 0.006976501
## 3979 0.9475763 0.052423663
## 3983 0.9791926 0.020807357
## 3984 0.9835771 0.016422862
## 3986 0.9926670 0.007332967
## 3987 0.9753259 0.024674072
## 3991 0.9717312 0.028268754
## 3997 0.8748668 0.125133223
## 3998 0.9958820 0.004118010
## 4003 0.9858638 0.014136218
## 4005 0.9880844 0.011915579
## 4012 0.9938604 0.006139635
## 4013 0.9756853 0.024314733
## 4019 0.9929990 0.007000981
## 4020 0.9939767 0.006023327
## 4022 0.7182912 0.281708834
## 4026 0.9544529 0.045547073
## 4039 0.9908485 0.009151501
## 4040 0.9933638 0.006636203
## 4041 0.9956558 0.004344153
## 4043 0.9933221 0.006677922
## 4045 0.9867831 0.013216919
## 4052 0.9332909 0.066709059
## 4054 0.9087435 0.091256490
## 4055 0.8283952 0.171604762
## 4069 0.9953594 0.004640599
## 4070 0.9597371 0.040262906
## 4073 0.9266185 0.073381461
## 4080 0.9957171 0.004282945
## 4081 0.9922047 0.007795312
## 4090 0.9867101 0.013289868
## 4093 0.9960428 0.003957234
## 4095 0.9732594 0.026740612
## 4098 0.9852100 0.014789975
## 4101 0.9679894 0.032010601
## 4114 0.9902438 0.009756240
## 4117 0.9911746 0.008825395
## 4120 0.9887712 0.011228840
## 4122 0.9660693 0.033930727
## 4125 0.9461877 0.053812257
## 4129 0.8455628 0.154437233
## 4131 0.9921507 0.007849254
## 4133 0.9947569 0.005243080
## 4138 0.9073507 0.092649307
## 4139 0.9880390 0.011961044
## 4143 0.9815020 0.018497954
## 4151 0.9920338 0.007966164
## 4155 0.9647175 0.035282481
## 4162 0.7724005 0.227599487
## 4163 0.9768722 0.023127777
## 4165 0.8046895 0.195310510
## 4173 0.9951737 0.004826339
## 4174 0.9928502 0.007149790
```

```
## 4176 0.9718588 0.028141204
## 4178 0.8663477 0.133652268
## 4179 0.9742255 0.025774527
## 4180 0.9960896 0.003910364
## 4182 0.9871664 0.012833596
## 4189 0.9899658 0.010034245
## 4193 0.9849657 0.015034326
## 4197 0.9412352 0.058764767
## 4203 0.9958071 0.004192911
## 4209 0.9972676 0.002732366
## 4215 0.9924646 0.007535440
## 4216 0.9951806 0.004819352
## 4220 0.9839082 0.016091799
## 4223 0.9908634 0.009136636
## 4225 0.9962561 0.003743910
## 4227 0.9964683 0.003531748
## 4228 0.9489154 0.051084576
## 4231 0.9453828 0.054617169
## 4232 0.9950248 0.004975229
## 4236 0.9927660 0.007233956
## 4246 0.9930920 0.006908031
## 4247 0.9914325 0.008567543
## 4248 0.9843830 0.015616994
## 4253 0.9031070 0.096892965
## 4255 0.9940686 0.005931367
## 4267 0.9937465 0.006253539
## 4269 0.9874542 0.012545817
## 4271 0.9951194 0.004880603
## 4273 0.9841303 0.015869699
## 4275 0.9950573 0.004942702
## 4281 0.9974140 0.002585994
## 4283 0.8460871 0.153912930
## 4285 0.9030941 0.096905885
## 4288 0.9814498 0.018550178
## 4289 0.9933099 0.006690051
## 4297 0.7290147 0.270985252
## 4302 0.9892419 0.010758067
## 4306 0.9903035 0.009696471
## 4311 0.9337963 0.066203710
## 4313 0.9577163 0.042283714
## 4317 0.9959803 0.004019719
## 4322 0.9948514 0.005148631
## 4324 0.6005893 0.399410690
## 4327 0.8926937 0.107306296
## 4330 0.9769246 0.023075440
## 4331 0.9637406 0.036259363
## 4335 0.9864234 0.013576608
## 4337 0.9749437 0.025056323
## 4352 0.8583438 0.141656213
## 4356 0.9800297 0.019970269
## 4357 0.9412303 0.058769676
## 4360 0.8262962 0.173703802
## 4362 0.9540033 0.045996747
## 4370 0.9926119 0.007388094
```

```
## 4372 0.9787974 0.021202644
## 4376 0.8755814 0.124418565
## 4379 0.9929579 0.007042051
## 4387 0.9874401 0.012559865
## 4388 0.9910581 0.008941920
## 4391 0.9946740 0.005326048
## 4394 0.9972585 0.002741534
## 4395 0.8973041 0.102695927
## 4396 0.9915578 0.008442160
## 4400 0.9928341 0.007165926
## 4402 0.9952083 0.004791661
## 4406 0.9873681 0.012631940
## 4407 0.9928775 0.007122531
## 4410 0.9930858 0.006914168
## 4411 0.9159566 0.084043377
## 4417 0.9849933 0.015006653
## 4423 0.9942809 0.005719079
## 4425 0.9213421 0.078657892
## 4427 0.9525785 0.047421486
## 4428 0.9868575 0.013142466
## 4430 0.8193516 0.180648434
## 4438 0.9821815 0.017818487
## 4442 0.9814545 0.018545473
## 4443 0.8326218 0.167378176
## 4445 0.8664534 0.133546603
## 4448 0.8822100 0.117789967
## 4450 0.9675684 0.032431563
## 4457 0.8880983 0.111901739
## 4460 0.9690451 0.030954926
## 4465 0.9961818 0.003818238
## 4468 0.9719879 0.028012143
## 4473 0.9703842 0.029615796
## 4474 0.9950082 0.004991842
## 4478 0.9936660 0.006334023
## 4484 0.9887804 0.011219619
## 4489 0.9932971 0.006702937
## 4490 0.9926031 0.007396897
## 4491 0.9783209 0.021679062
## 4499 0.9808002 0.019199826
## 4503 0.9760652 0.023934822
## 4506 0.9891954 0.010804607
## 4507 0.9794318 0.020568166
## 4509 0.9946436 0.005356355
## 4511 0.8142701 0.185729899
## 4518 0.9929327 0.007067292
## 4519 0.9845211 0.015478946
## 4525 0.9905240 0.009476010
## 4528 0.9880675 0.011932466
## 4532 0.9919106 0.008089390
## 4534 0.9645637 0.035436290
## 4541 0.9958599 0.004140054
## 4544 0.9943046 0.005695378
## 4545 0.9912805 0.008719517
## 4547 0.9775768 0.022423186
```

```
## 4559 0.9779244 0.022075606
## 4564 0.9805561 0.019443855
## 4565 0.9791603 0.020839727
## 4566 0.9891346 0.010865370
## 4567 0.9940352 0.005964805
## 4569 0.9951632 0.004836764
## 4571 0.9939309 0.006069107
## 4572 0.9950832 0.004916795
## 4574 0.9933542 0.006645831
## 4578 0.9220039 0.077996063
## 4584 0.9909904 0.009009583
## 4585 0.9941006 0.005899434
## 4589 0.9893175 0.010682508
## 4590 0.5183566 0.481643369
## 4593 0.9961141 0.003885859
## 4595 0.9869732 0.013026826
## 4598 0.9309107 0.069089267
## 4603 0.9838157 0.016184298
## 4608 0.9723375 0.027662476
## 4613 0.9933280 0.006671985
## 4618 0.9823851 0.017614901
## 4619 0.9946430 0.005356992
## 4624 0.9881254 0.011874616
## 4631 0.9653289 0.034671088
## 4632 0.9795583 0.020441667
## 4634 0.8877910 0.112209016
## 4641 0.9843457 0.015654277
## 4644 0.9955502 0.004449765
## 4647 0.9441872 0.055812783
## 4649 0.9919513 0.008048748
## 4657 0.9959941 0.004005911
## 4658 0.9613546 0.038645357
## 4662 0.9967568 0.003243191
## 4665 0.9068132 0.093186787
## 4671 0.9953042 0.004695816
## 4672 0.9580436 0.041956352
## 4677 0.9600861 0.039913948
## 4678 0.9643144 0.035685617
## 4684 0.4151926 0.584807369
## 4687 0.9731868 0.026813239
## 4689 0.9941239 0.005876095
## 4691 0.9870326 0.012967409
## 4695 0.9754341 0.024565945
## 4701 0.9789747 0.021025303
## 4702 0.9245885 0.075411531
## 4704 0.9866769 0.013323068
## 4706 0.8816184 0.118381628
## 4707 0.9685571 0.031442851
## 4709 0.9937529 0.006247070
## 4713 0.8457064 0.154293590
## 4715 0.9758244 0.024175610
## 4716 0.6489490 0.351051012
## 4720 0.9912402 0.008759823
## 4729 0.9954855 0.004514483
```

```
## 4732 0.9883574 0.011642645
## 4733 0.9471502 0.052849789
## 4734 0.9285893 0.071410678
## 4736 0.9926649 0.007335114
## 4744 0.9934092 0.006590838
## 4746 0.9375949 0.062405113
## 4752 0.9965752 0.003424838
## 4754 0.9836877 0.016312265
## 4757 0.9961122 0.003887786
## 4760 0.9858055 0.014194484
## 4761 0.9938130 0.006186987
## 4764 0.9957597 0.004240341
## 4769 0.9276730 0.072326979
## 4770 0.9935157 0.006484307
## 4777 0.8961240 0.103875980
## 4778 0.9761634 0.023836584
## 4780 0.9810457 0.018954262
## 4781 0.9928734 0.007126631
## 4785 0.9944841 0.005515883
## 4786 0.9928273 0.007172653
## 4787 0.9914265 0.008573466
## 4788 0.9937674 0.006232570
## 4792 0.9913089 0.008691108
## 4794 0.9926826 0.007317397
## 4796 0.6568517 0.343148296
## 4798 0.9939163 0.006083741
## 4802 0.9947227 0.005277277
## 4803 0.9881318 0.011868192
## 4804 0.9807077 0.019292308
## 4810 0.9671021 0.032897888
## 4813 0.9941163 0.005883678
## 4815 0.9923977 0.007602334
## 4823 0.7948989 0.205101060
## 4824 0.8770439 0.122956068
## 4828 0.9924485 0.007551453
## 4831 0.9965829 0.003417096
## 4832 0.9936070 0.006392986
## 4834 0.9818435 0.018156495
## 4835 0.9707562 0.029243763
## 4836 0.7889931 0.211006901
## 4841 0.9953693 0.004630743
## 4844 0.9954736 0.004526387
## 4845 0.9792707 0.020729307
## 4846 0.9929224 0.007077563
## 4847 0.9972103 0.002789724
## 4848 0.9926435 0.007356510
## 4853 0.9927239 0.007276092
## 4860 0.9955571 0.004442865
## 4870 0.9928920 0.007108005
## 4876 0.9814109 0.018589136
## 4877 0.9719076 0.028092427
## 4879 0.9956170 0.004383000
## 4881 0.7686721 0.231327939
## 4883 0.8277195 0.172280478
```

```
## 4885 0.9930470 0.006952989
## 4891 0.9962774 0.003722635
## 4894 0.9860464 0.013953581
## 4897 0.9063820 0.093617958
## 4898 0.9905410 0.009459033
## 4902 0.9940665 0.005933528
## 4904 0.9960700 0.003929970
## 4907 0.9813099 0.018690120
## 4910 0.9959424 0.004057590
## 4914 0.9855806 0.014419422
## 4916 0.9739070 0.026092956
## 4918 0.8712945 0.128705550
## 4922 0.9765857 0.023414269
## 4923 0.9970995 0.002900507
## 4924 0.9812961 0.018703904
## 4926 0.9911839 0.008816114
## 4937 0.9323347 0.067665330
## 4944 0.9912949 0.008705132
## 4945 0.9947254 0.005274591
## 4946 0.9794777 0.020522271
## 4950 0.9967416 0.003258394
## 4951 0.9893907 0.010609349
## 4952 0.9675140 0.032485953
## 4953 0.9823244 0.017675582
## 4960 0.9869189 0.013081078
## 4962 0.9826108 0.017389157
## 4965 0.9924261 0.007573947
## 4966 0.9745444 0.025455611
## 4967 0.9625213 0.037478667
## 4968 0.9958207 0.004179289
## 4971 0.9706542 0.029345778
## 4978 0.9954384 0.004561575
## 4979 0.8888633 0.111136702
## 4987 0.8523736 0.147626412
## 4988 0.8141138 0.185886212
## 4990 0.9951553 0.004844713
## 4992 0.9897453 0.010254656
## 5004 0.9949405 0.005059498
## 5005 0.9938500 0.006149973
## 5011 0.9770868 0.022913157
## 5014 0.9897437 0.010256293
## 5015 0.9855015 0.014498519
## 5017 0.9590995 0.040900476
## 5019 0.9929667 0.007033276
## 5020 0.9944293 0.005570658
## 5027 0.9970182 0.002981793
## 5030 0.9976606 0.002339441
## 5037 0.9729897 0.027010255
## 5038 0.9658298 0.034170240
## 5043 0.9730097 0.026990344
## 5045 0.9871371 0.012862872
## 5052 0.9948850 0.005115020
## 5053 0.9838145 0.016185478
## 5054 0.9842896 0.015710373
```

```
## 5055 0.9899112 0.010088770
## 5064 0.8796320 0.120367954
## 5068 0.8621056 0.137894378
## 5069 0.9933951 0.006604944
## 5073 0.9909931 0.009006868
## 5075 0.9316870 0.068312971
## 5080 0.9963407 0.003659328
## 5081 0.9735405 0.026459490
## 5084 0.9381804 0.061819594
## 5085 0.9343553 0.065644675
## 5091 0.9387212 0.061278782
## 5100 0.7967956 0.203204355
## 5102 0.9698980 0.030101986
## 5104 0.9909520 0.009048003
## 5107 0.9953449 0.004655148
## 5109 0.9804782 0.019521845
```

```
lda.prob = ifelse(lda.pred$Yes > 0.5, "Yes", "No")

confusionMatrix(data = as.factor(lda.prob),
                reference = test.data$stroke,
                positive = "Yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  1452   70
##        Yes    6    4
##
##                Accuracy : 0.9504
##                  95% CI : (0.9383, 0.9607)
##     No Information Rate : 0.9517
##     P-Value [Acc > NIR] : 0.6233
##
##                   Kappa : 0.0847
##
##  Mcnemar's Test P-Value : 4.953e-13
##
##             Sensitivity : 0.054054
##             Specificity : 0.995885
##          Pos Pred Value : 0.400000
##          Neg Pred Value : 0.954008
##              Prevalence : 0.048303
##          Detection Rate : 0.002611
##    Detection Prevalence : 0.006527
##       Balanced Accuracy : 0.524969
##
##        'Positive' Class : Yes
##
```

```
roc.lda = roc(test.data$stroke, lda.pred[,2])
```

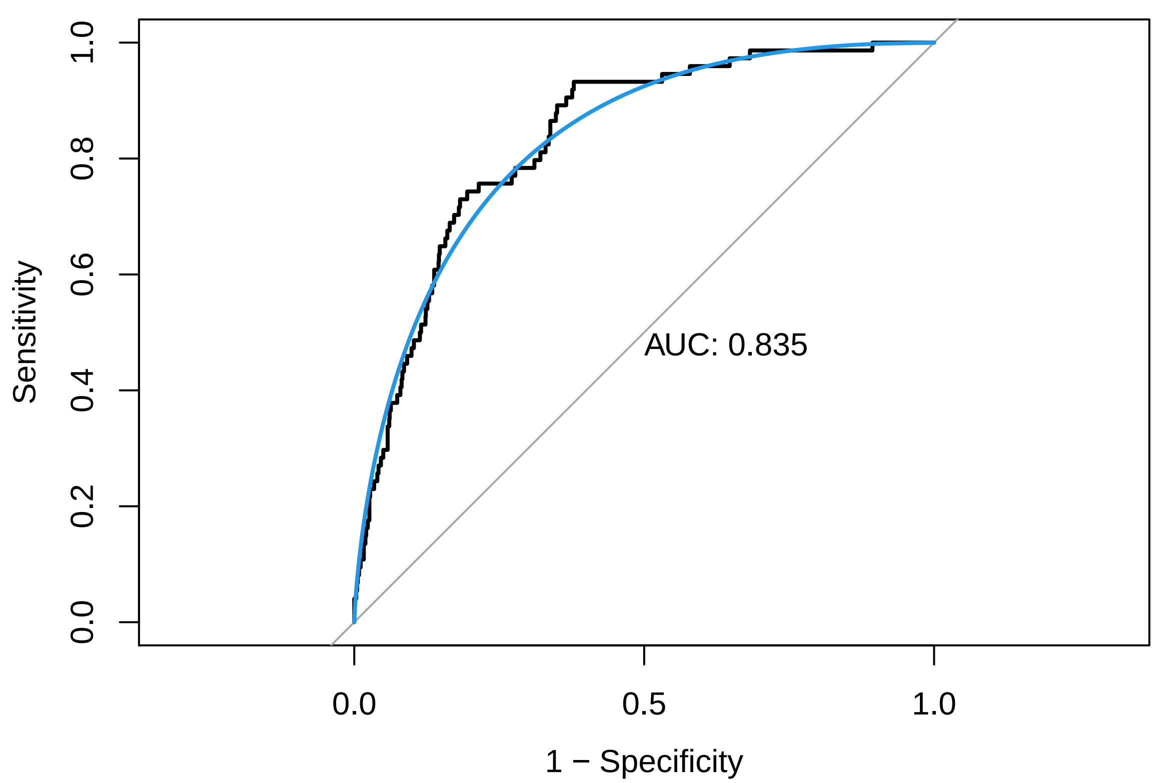
```
## Setting levels: control = No, case = Yes
```

```
## Setting direction: controls < cases
```

```r
auc.lda = roc.lda$auc[1]
auc.lda
```

```
## [1] 0.8354002
```

```r
plot(roc.lda, legacy.axes = TRUE, print.auc = TRUE)
plot(smooth(roc.lda), col = 4, add = TRUE)
```




```r
varImp(lda.fit)
```

```
## ROC curve variable importance
##
##                    Importance
## age                   100.000
## ever_married           32.881
## work_type              29.346
## avg_glucose_level      28.665
## smoking_status         21.892
## hypertension           20.993
## heart_disease          17.829
## bmi                    15.836
## Residence_type          1.466
## gender                  0.000
```

### KNN

```r
set.seed(123)
knn.fit = train(   x = train.data[, c(1:10)],
                   y = train.data$stroke,
                   method = "knn",
                   preProcess = c("center","scale"),
                   tuneGrid = data.frame(k = seq(1,200,by=5)),
                   trControl = ctrl)
```

```
## Warning in train.default(x = train.data[, c(1:10)], y = train.data$stroke, : The
## metric "Accuracy" was not in the result set. ROC will be used instead.
```

```r
knn.fit1 = train(  x = train.data[, c(1:10)],
                   y = train.data$stroke,
                   method = "knn",
                   preProcess = c("center","scale"),
                   tuneGrid = data.frame(k = seq(1,200,by=5)),
                   trControl = ctrl1)
```

```
## Warning in train.default(x = train.data[, c(1:10)], y = train.data$stroke, : The
## metric "Accuracy" was not in the result set. ROC will be used instead.
```

```r
knn.fit$finalModel
```

```
## 196-nearest neighbor model
## Training set outcome distribution:
##
##   No  Yes
## 3402  175
```

```r
knn.pred =
    predict(knn.fit, newdata = test.data, type = "prob")

knn.prob = ifelse(knn.pred$Yes > 0.5, "Yes", "No")

confusionMatrix(data = as.factor(knn.prob),
                reference = test.data$stroke,
                positive = "Yes")
```

```
## Warning in confusionMatrix.default(data = as.factor(knn.prob), reference =
## test.data$stroke, : Levels are not in the same order for reference and data.
## Refactoring data to match.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  1458   74
##        Yes    0    0
```

```
##
##                Accuracy : 0.9517
##                  95% CI : (0.9397, 0.9619)
##     No Information Rate : 0.9517
##     P-Value [Acc > NIR] : 0.5309
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.0000
##             Specificity : 1.0000
##          Pos Pred Value :    NaN
##          Neg Pred Value : 0.9517
##              Prevalence : 0.0483
##          Detection Rate : 0.0000
##    Detection Prevalence : 0.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : Yes
##
```

```
roc.knn = roc(test.data$stroke, knn.pred[,2])
```

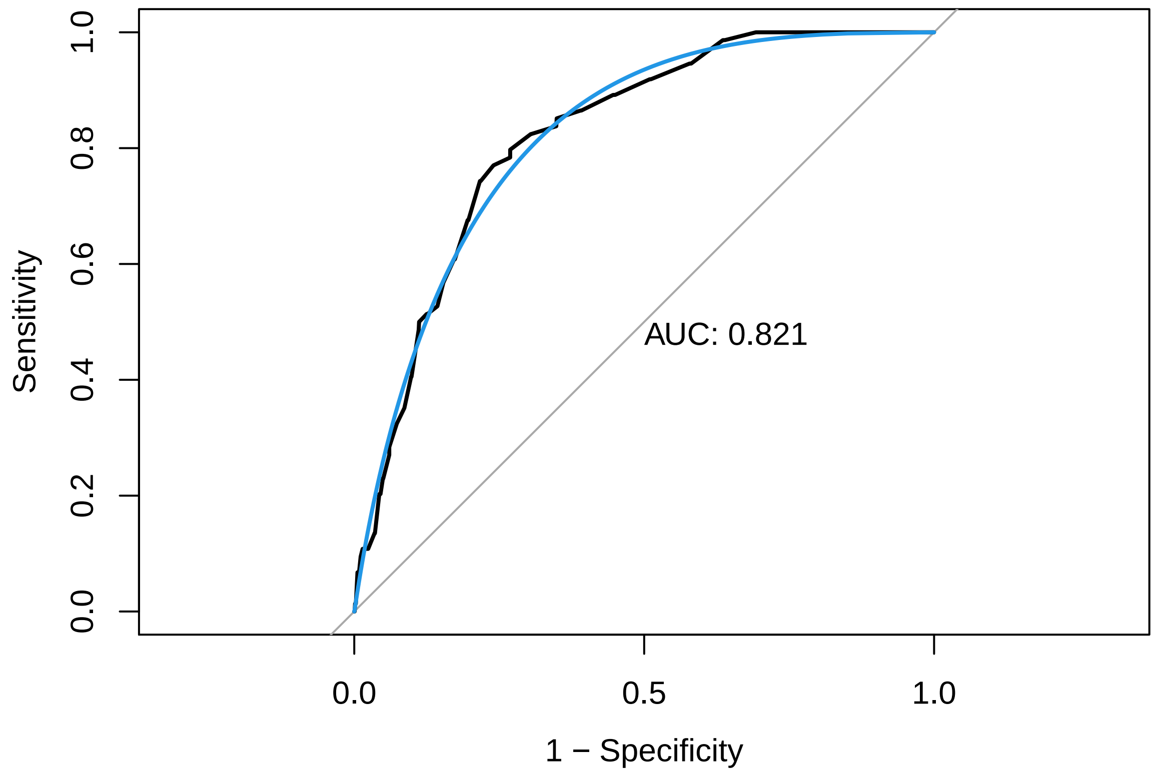
```
## Setting levels: control = No, case = Yes
```

```
## Setting direction: controls < cases
```

```
auc.knn = roc.knn$auc[1]
auc.knn
```

```
## [1] 0.821363
```

```
plot(roc.knn, legacy.axes = TRUE, print.auc = TRUE)
plot(smooth(roc.knn), col = 4, add = TRUE)
```



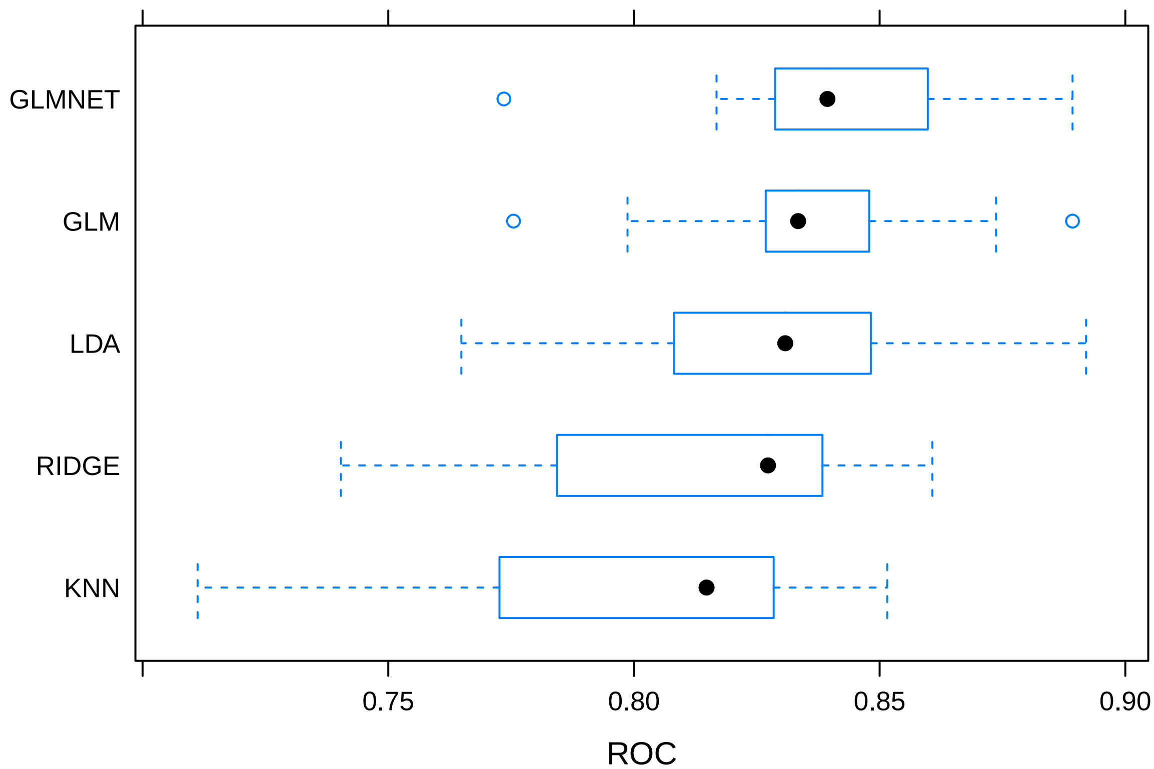

Evaluation the ROC by resampling these models(with normal sampling method).

```
res = resamples(list(GLM = lm.fit,
                      GLMNET = model.glmn,
                      RIDGE = ridge.fit,
                      LDA = lda.fit,
                      KNN = knn.fit))
summary(res)
```

```
##
## Call:
## summary.resamples(object = res)
##
## Models: GLM, GLMNET, RIDGE, LDA, KNN
## Number of resamples: 10
##
## ROC
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## GLM    0.7754902 0.8268599 0.8334198 0.8349111 0.8467729 0.8892531    0
## GLMNET 0.7735294 0.8298010 0.8393791 0.8405305 0.8569204 0.8892531    0
## RIDGE  0.7403595 0.7909336 0.8272876 0.8136173 0.8361697 0.8607266    0
## LDA    0.7648693 0.8104695 0.8307958 0.8278133 0.8455474 0.8920131    0
## KNN    0.7111928 0.7796197 0.8147780 0.7988977 0.8277982 0.8515571    0
##
## Sens
##             Min.   1st Qu.  Median      Mean   3rd Qu. Max. NA's
```

```
## GLM     1.0000000 1.0000000 1.00000 1.0000000 1.0000000    1    0
## GLMNET  1.0000000 1.0000000 1.00000 1.0000000 1.0000000    1    0
## RIDGE   1.0000000 1.0000000 1.00000 1.0000000 1.0000000    1    0
## LDA     0.9882353 0.9911765 0.99266 0.9929438 0.9941176    1    0
## KNN     1.0000000 1.0000000 1.00000 1.0000000 1.0000000    1    0
##
## Spec
##         Min. 1st Qu.     Median       Mean    3rd Qu.       Max. NA's
## GLM        0       0 0.00000000 0.00000000 0.00000000 0.00000000    0
## GLMNET     0       0 0.00000000 0.00000000 0.00000000 0.00000000    0
## RIDGE      0       0 0.00000000 0.00000000 0.00000000 0.00000000    0
## LDA        0       0 0.05555556 0.03431373 0.05800654 0.05882353    0
## KNN        0       0 0.00000000 0.00000000 0.00000000 0.00000000    0
```

```
bwplot(res, metric = "ROC")
```



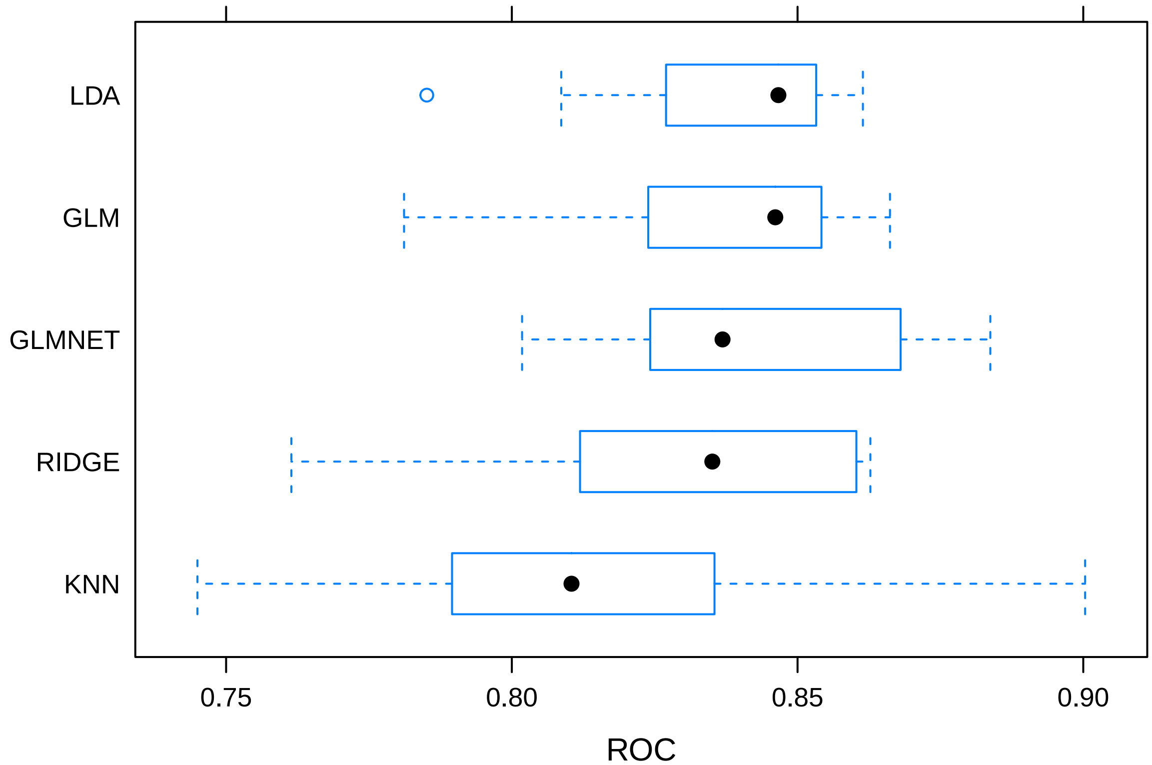

Evaluation the ROC by resampling these models(with smote sampling method).

```
res1 = resamples(list(GLM = lm.fit1,
                      GLMNET = model.glmn1,
                      RIDGE = ridge.fit1,
                      LDA = lda.fit1,
                      KNN = knn.fit1))
summary(res1)
```

```
##
```

```
## Call:
## summary.resamples(object = res1)
##
## Models: GLM, GLMNET, RIDGE, LDA, KNN
## Number of resamples: 10
##
## ROC
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## GLM    0.7811419 0.8243496 0.8461073 0.8358150 0.8535488 0.8661765    0
## GLMNET 0.8017974 0.8254758 0.8368752 0.8409525 0.8635264 0.8837330    0
## RIDGE  0.7614187 0.8144906 0.8350875 0.8311311 0.8581564 0.8627451    0
## LDA    0.7851211 0.8280221 0.8466480 0.8371647 0.8527778 0.8614379    0
## KNN    0.7449827 0.7942979 0.8104440 0.8165459 0.8326257 0.9003268    0
##
## Sens
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## GLM    0.7205882 0.7698529 0.7852941 0.7821899 0.7930999 0.8294118    0
## GLMNET 0.7323529 0.7670778 0.7856089 0.7857176 0.8110294 0.8264706    0
## RIDGE  0.7323529 0.7942686 0.8002889 0.7977652 0.8088235 0.8352941    0
## LDA    0.7088235 0.7639706 0.7738658 0.7692565 0.7794118 0.8058824    0
## KNN    0.7323529 0.7441176 0.7632353 0.7639434 0.7740275 0.8181818    0
##
## Spec
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## GLM    0.5882353 0.6764706 0.7140523 0.7248366 0.7982026 0.8333333    0
## GLMNET 0.5294118 0.7058824 0.7140523 0.7078431 0.7222222 0.8333333    0
## RIDGE  0.5294118 0.6029412 0.7140523 0.6901961 0.7540850 0.8888889    0
## LDA    0.6470588 0.7058824 0.7500000 0.7532680 0.7777778 0.8888889    0
## KNN    0.4705882 0.6805556 0.7434641 0.7362745 0.8235294 0.8888889    0
```

```
bwplot(res1, metric = "ROC")
```




### Variable Importance

```
varImp(model.glmn)
```

```
## glmnet variable importance
##
##                   Overall
## age               100.000
## avg_glucose_level   9.872
## heart_disease       5.792
## hypertension        5.713
## work_type           0.000
## gender              0.000
## Residence_type      0.000
## bmi                 0.000
## smoking_status      0.000
## ever_married        0.000
```

We can see that from the best performance model: Penalized Regression Model, Age is the most important variable in determining having outcome of interest.

## Conclusion and Model limitation

Based on the Exploratory Data Analysis, we can see that only 5% of all the people in the dataset had a stroke at some point. This means that our baseline dummy model has an accuracy of 95%. I tried to use smote sampling to adjust the unbalanced dataset and found that the Penalized Logistic Regression outperformed Ridge Regression, LDA, Logistic Regression, and KNN. However, when I removed the smote sampling from the train control function, the rank fluctuated. Penalized Logistic Regression still performed the best, while logistic regression came the next, with LDA, Ridge Regression, and KNN followed. Ridge Regression performed the best between the two sampling methods, and KNN performed the worst. It is because Logistic regression is popular for classification when K= 2(stroke vs. non-stroke). In contrast, Ridge regression is to shrink the coefficients of correlated predictors towards each other. On the other hand, LDA is more appropriate when the sample size is small, or the classes are well separated, and Gaussian assumptions are reasonable, as well as when the class greater than two categories. KNN is easier to be affected by outliers.

I will use the evaluation based on the normal sampling way to avoid the biased prediction results because of oversampling diseased reality distribution/prevalence.