Project Six for Statistical Data Mining

Raymond Anthony Ford
raford2@miners.utep.edu

Due: 12 December 2018

Contents

1 Data preparation

We begin this project by first bringing the data into R and checking if any missing values are present in our data.

```
dat <- read.table(file=
    "http://archive.ics.uci.edu/ml/machine-learning-databases/00252/pop_failures.dat",
    header=TRUE)
anyNA(dat)
```

```
## [1] FALSE
```

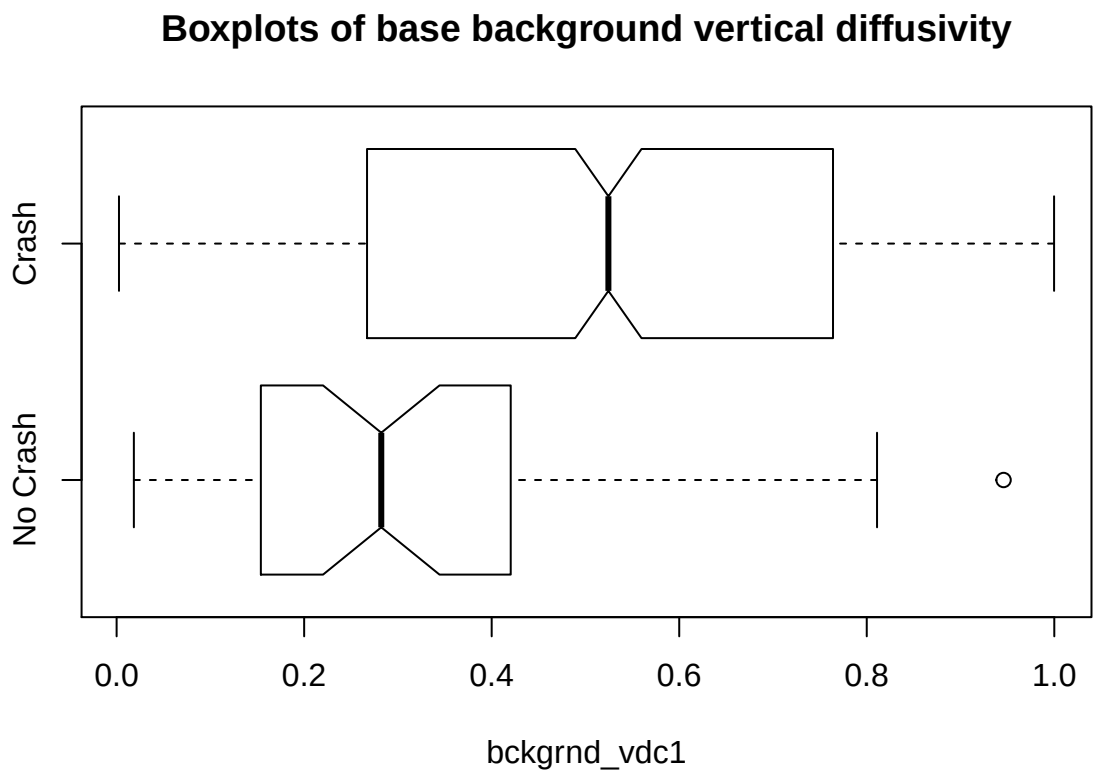

Figure 1: Boxplots of base background vertical diffusivity for crash vs. no crash.

We see that we do not have any missing values in our data. Finally we remove the variables `Study` and `Run` as they have no purpose in our analysis.

```
dat <- dat[, -c(1, 2)]
```

2 Exploratory data analysis

```
boxplot(dat$bckgrnd_vdc1 ~ dat$outcome, notch=TRUE, horizontal=TRUE,
        main="Boxplots of base background vertical diffusivity",
        names=c("No Crash", "Crash"), xlab="bckgrnd_vdc1")
```

Our first interesting finding in EDA is that there exists a statistically significant difference between the median of background vertical diffusivity for simulations that crashed vs those that did not. This can be seen in Figure 1.

```
boxplot(dat$convect_corr ~ dat$outcome, notch=TRUE, horizontal=TRUE,
        main="Boxplots of tracer and momentum mixing coefficients",
        names=c("No Crash", "Crash"), xlab="convect_corr")
```

Our second interesting finding in EDA is that there exists a statistically significant difference between the median of tracer and momentum mixing coefficients for simulations that crashed

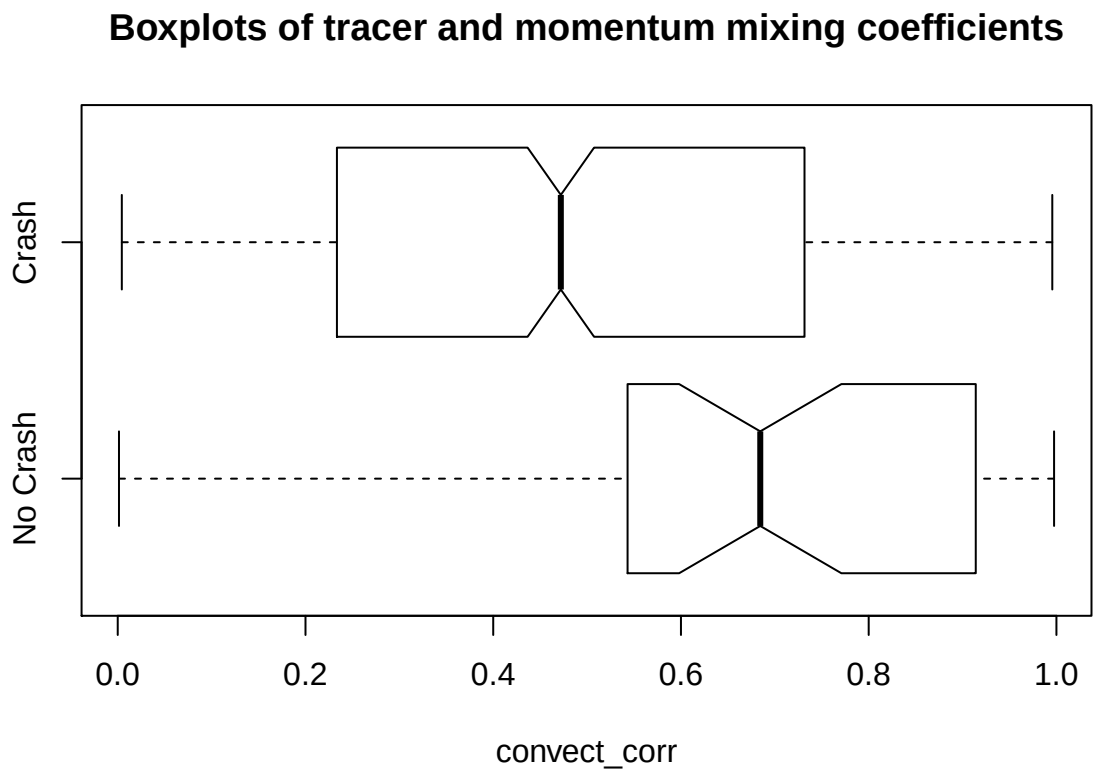

Figure 2: Boxplots of tracer and momentum mixing coefficients for crash vs. no crash.

vs those that did not. This can be seen in Figure 2.

```
boxplot(dat$vconst_corr ~ dat$outcome, notch=TRUE, horizontal=TRUE,
        main="Boxplots of one of the variable viscosity parameters",
        names=c("No Crash", "Crash"), xlab="vconst_corr")
```

Our third interesting finding in EDA is that there exists a statistically significant difference between the median of one of the variable viscosity parameters for simulations that crashed vs those that did not. This can be seen in Figure 3.

3 Data partitioning

Before building our models, we randomly split the data into a training and testing data set with a ratio of 2:1 in sample size.

```
set.seed(5494)
training.data.index <- sample(1:nrow(dat), 0.667*nrow(dat))
train <- dat[training.data.index, ]
test <- dat[-training.data.index, ]
```

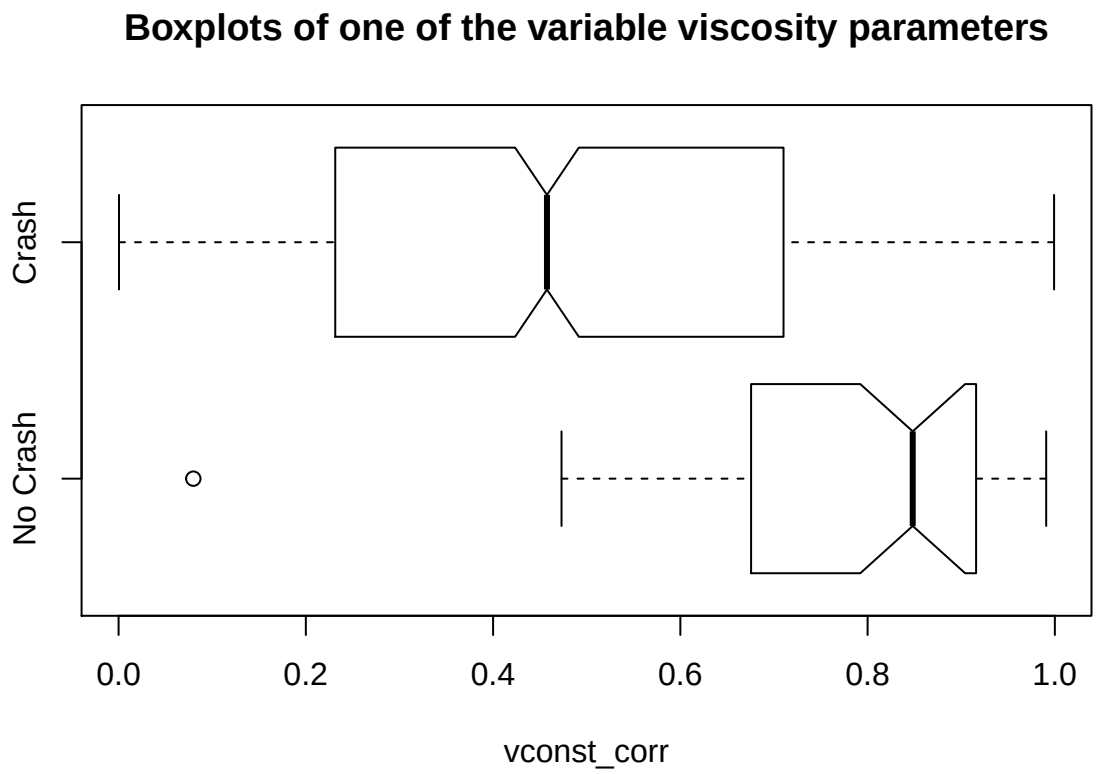

Figure 3: Boxplots of one of the variable viscosity parameters for crash vs. no crash.

4 Models

We will build four different types of models: logistic regression with regularization, random forests, artificial neural network, and support vector machine.

4.1 Logistic regression

We fit our first model using logistic regression with variables selected via LASSO. Our lambda parameter is chosen by cross validation. The following code shows how this was accomplished.

```
suppressMessages(require("glmnet"))
formula0 <- outcome ~ vconst_corr + vconst_2 + vconst_3 + vconst_4 +
    vconst_5 + vconst_7 + ah_corr + ah_bolus + slm_corr + efficiency_factor +
    tidal_mix_max + vertical_decay_scale + convect_corr + bckgrnd_vdc1 +
    bckgrnd_vdc_ban + bckgrnd_vdc_eq + bckgrnd_vdc_psim + Prandtl
X <- model.matrix(as.formula(formula0), data=train)
y <- train$outcome
X.valid <- model.matrix(as.formula(formula0), data=test)
y.valid <- test$outcome
Lambda <- seq(0.0001, 0.5, length.out = 500)
L <- length(Lambda)
```

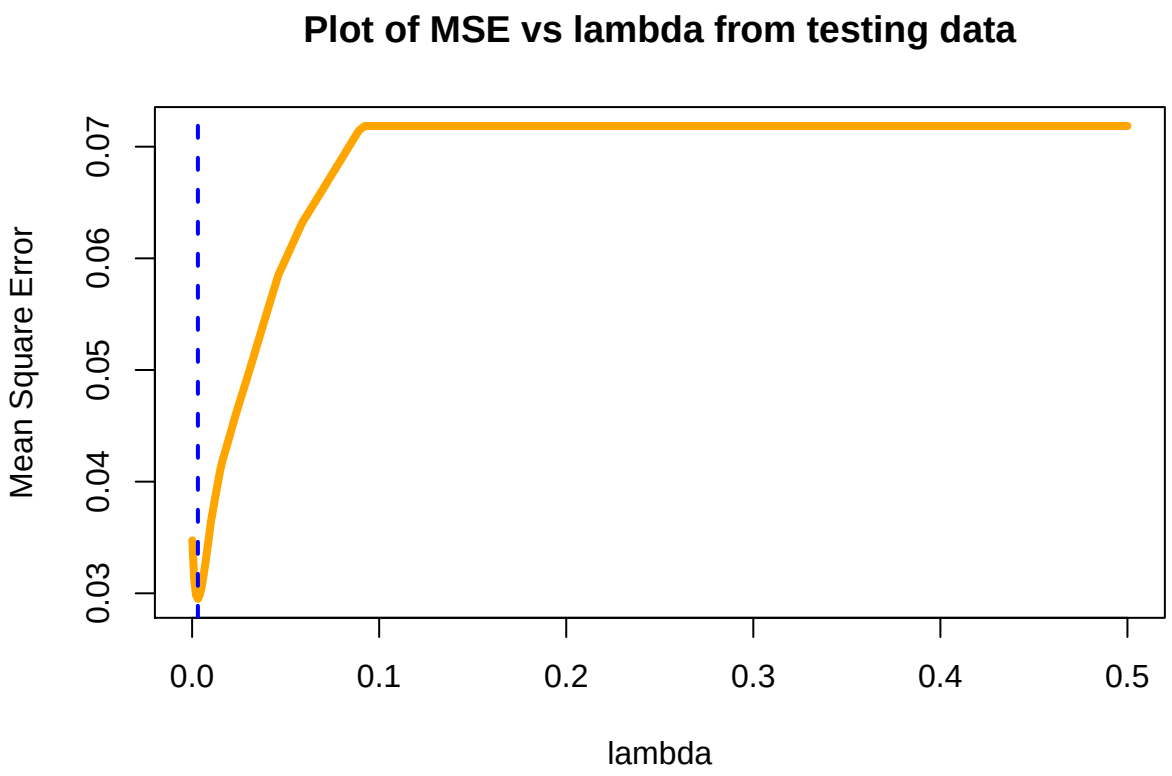

Figure 4: The best lambda for the lowest MSE

```
OUT <- matrix(0, L, 2)
for (i in 1:L) {
    fit <- glmnet(x=X, y=y, family="binomial", alpha=1, lambda=Lambda[i],
                  standardize=TRUE, thresh=1e-07, maxit=1000)
    pred <- predict(fit, newx=X.valid, s=Lambda[i], type="response")
    mse <- mean((as.numeric(y.valid)-pred)**2)
    OUT[i, ] <- c(Lambda[i], mse)
}
lam <- which.min(OUT[ ,2])
lambda.best <- OUT[lam, 1]
lambda.best
```

```
## [1] 0.003105411
```

We next make a plot showing the relationship between the many values of lambda available and their effect on mean square error.

```
plot(OUT[,1], OUT[,2], type="l", col="orange", lwd=4, xlab="lambda",
     ylab="Mean Square Error",
     main="Plot of MSE vs lambda from testing data")
abline(v=OUT[lam,1], col="blue", lty=2, lwd=2)
```

Next, we will use the best value of lambda to construct a LASSO logistic regression model

both our training and testing datasets.

```
X.d1d2 <- rbind(train, test)
X <- model.matrix(as.formula(formula0), data=X.d1d2)
y <- X.d1d2$outcome
fit.best <- glmnet(x=X, y=y, family="binomial", alpha=1, lambda=lambda.best,
                   standardize=TRUE, thresh=1e-07, maxit=1000)
coef(fit.best)
```

```
## 20 x 1 sparse Matrix of class "dgCMatrix"
##                                s0
## (Intercept)          10.6163904
## (Intercept)           .
## vconst_corr          -8.0385281
## vconst_2             -7.5559482
## vconst_3             -0.2072831
## vconst_4              1.2154953
## vconst_5              1.5000786
## vconst_7              1.0368447
## ah_corr               0.2711367
## ah_bolus              0.1686788
## slm_corr              0.6361793
## efficiency_factor    -0.5665629
## tidal_mix_max        -0.2837449
## vertical_decay_scale -1.1101410
## convect_corr         -3.9183882
## bckgrnd_vdc1          3.8310966
## bckgrnd_vdc_ban       .
## bckgrnd_vdc_eq        1.8242435
## bckgrnd_vdc_psim      1.3809766
## Prandtl               .
```

We see that the coefficients for two predictors have shrunk to zero: `bckgrnd_vdc_ban` and `Prandtl`. Finally, we compute predictions to be used for the ROC AUC curve to be shown at the end of this project.

```
X.test <- model.matrix(as.formula(formula0), data=test)
pred.fit <- predict(fit.best, newx=X.test, s=lambda.best, type="response")
```

4.2 Random Forests

The next model that we fit is a random forest model. We have set `importance=TRUE` so that we can obtain a variable importance ranking.

```
suppressMessages(library(randomForest, quietly=TRUE))
set.seed(5494)
```

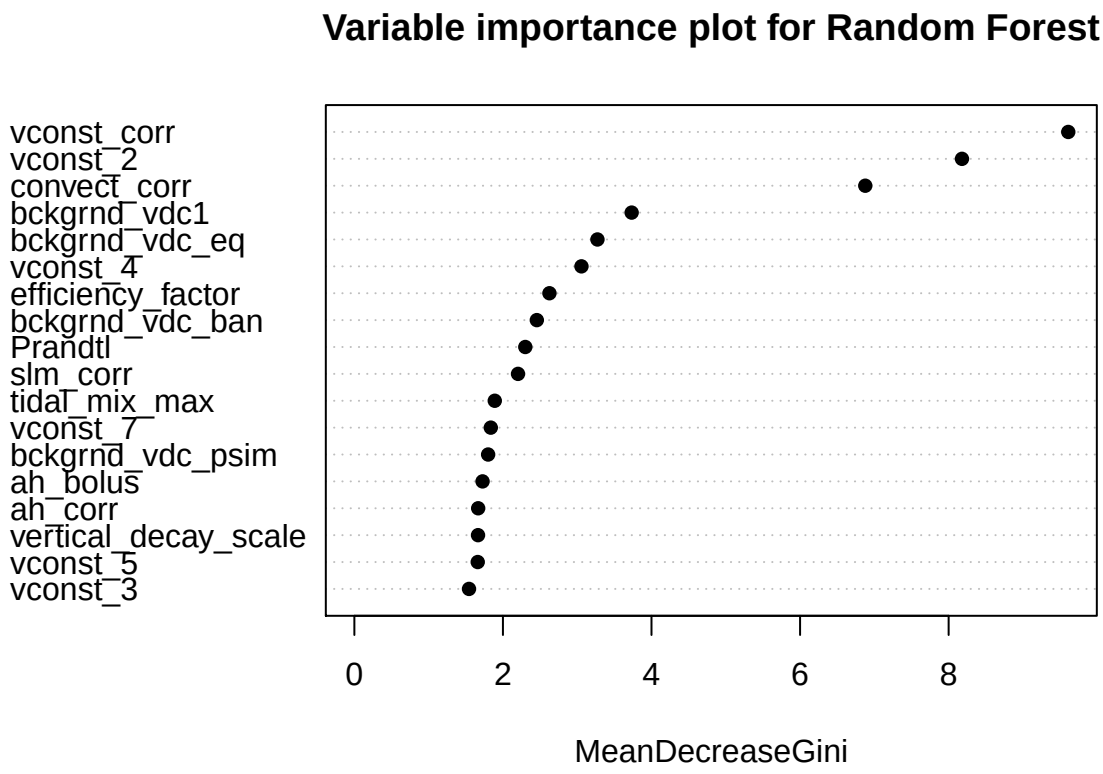

Figure 5: Variable importance plot for Random Forest measured by the mean decrease in Gini.

```
cm.rf <- randomForest(as.factor(outcome)~., data=train, importance=TRUE)
cm.rf.pred <- as.data.frame(predict(cm.rf, test, type="prob"))$`1`
```

Next we obtain the variable importance ranking plot to view which variables are important. Moreover, we have set `type=2` to use the mean decrease in Gini to determine which variables are important because our problem is a classification problem.

```
varImpPlot(cm.rf, main="Variable importance plot for Random Forest", pch=16, type=2)
```

4.3 Artificial Neural Network

We are instructed to build three different ANNs.

4.3.1 ANN1

Our first ANN has one hidden layer with six neurons.

```
library(nnet)
set.seed(1)
fit.nn1 <- nnet(outcome ~ ., data=train, size=6, linout = TRUE, entropy = FALSE,
```

```
                softmax = FALSE, skip = FALSE, rang = 0.7, maxit = 100,
                trace = FALSE, MaxNWts = 1000, abstol = 1.0e-4, reltol = 1.0e-8)
pred.ann1 <- predict(fit.nn1, new=test)
```

4.3.2 ANN2

Our second ANN has one hidden layer with 12 neurons.

```
set.seed(10)
fit.nn2 <- nnet(outcome ~ ., data=train, size=12, linout = TRUE, entropy = FALSE,
                softmax = FALSE, skip = FALSE, rang = 0.7, maxit = 100,
                trace = FALSE, MaxNWts = 1000, abstol = 1.0e-4, reltol = 1.0e-8)
pred.ann2 <- predict(fit.nn2, new=test)
```

4.3.3 ANN3

Our third ANN has one hidden layer with 2 neurons.

```
set.seed(100)
fit.nn3 <- nnet(outcome ~ ., data=train, size=2, linout = TRUE, entropy = FALSE,
                softmax = FALSE, skip = FALSE, rang = 0.7, maxit = 100,
                trace = FALSE, MaxNWts = 1000, abstol = 1.0e-4, reltol = 1.0e-8)
pred.ann3 <- predict(fit.nn3, new=test)
```

4.4 Support vector machine

We are instructed to build three different ANNs. We will keep all arguments the same and only change kernal type.

4.4.1 SVM1

Our first SVM uses a linear kernal.

```
library(e1071)
fit.svm1 <- svm(outcome ~ ., data=train, probability=TRUE, kernal="linear")
pred.svm1 <- predict(fit.svm1, type="prob", newdata=test, probability = TRUE)
```

4.4.2 SVM2

Our second SVM uses a radial kernal.

```
fit.svm2 <- svm(outcome ~ ., data=train, probability=TRUE, kernal="radial")
pred.svm2 <- predict(fit.svm2, type="prob", newdata=test, probability = TRUE)
```

4.4.3 SVM3

Our third SVM uses a sigmoid kernal.

```
fit.svm3 <- svm(outcome ~ ., data=train, probability=TRUE, kernal="sigmoid")
pred.svm3 <- predict(fit.svm3, type="prob", newdata=test, probability = TRUE)
```

5 Conclusion

We finally plot the ROCAUC curves from all of the previous models.

```
suppressMessages(suppressWarnings(library(pROC, quietly=TRUE)))
par(mfrow=c(3, 3), mar=rep(4,4), pty="s")
suppressWarnings(plot.roc(test$outcome, pred.fit, main="LR", percent=TRUE,
         print.auc=TRUE, print.auc.cex=1.0, col="blue"))
plot.roc(test$outcome, cm.rf.pred, main="RF", percent=TRUE, print.auc=TRUE,
         print.auc.cex=1.0, col="blue")
suppressWarnings(plot.roc(test$outcome, pred.ann1, main="ANN1", percent=TRUE,
         print.auc=TRUE, print.auc.cex=1.0, col="blue"))
suppressWarnings(plot.roc(test$outcome, pred.ann2, main="ANN2", percent=TRUE,
         print.auc=TRUE, print.auc.cex=1.0, col="blue"))
suppressWarnings(plot.roc(test$outcome, pred.ann3, main="ANN3", percent=TRUE,
         print.auc=TRUE, print.auc.cex=1.0, col="blue"))
plot.roc(test$outcome, pred.svm1, main="SVM1", percent=TRUE, print.auc=TRUE,
         print.auc.cex=1.0, col="blue")
plot.roc(test$outcome, pred.svm2, main="SVM2", percent=TRUE, print.auc=TRUE,
         print.auc.cex=1.0, col="blue")
plot.roc(test$outcome, pred.svm3, main="SVM3", percent=TRUE, print.auc=TRUE,
         print.auc.cex=1.0, col="blue")
```

In Figure 6 we see the model with the highest AUC is the logistic regression model with an AUC of 98.3%.

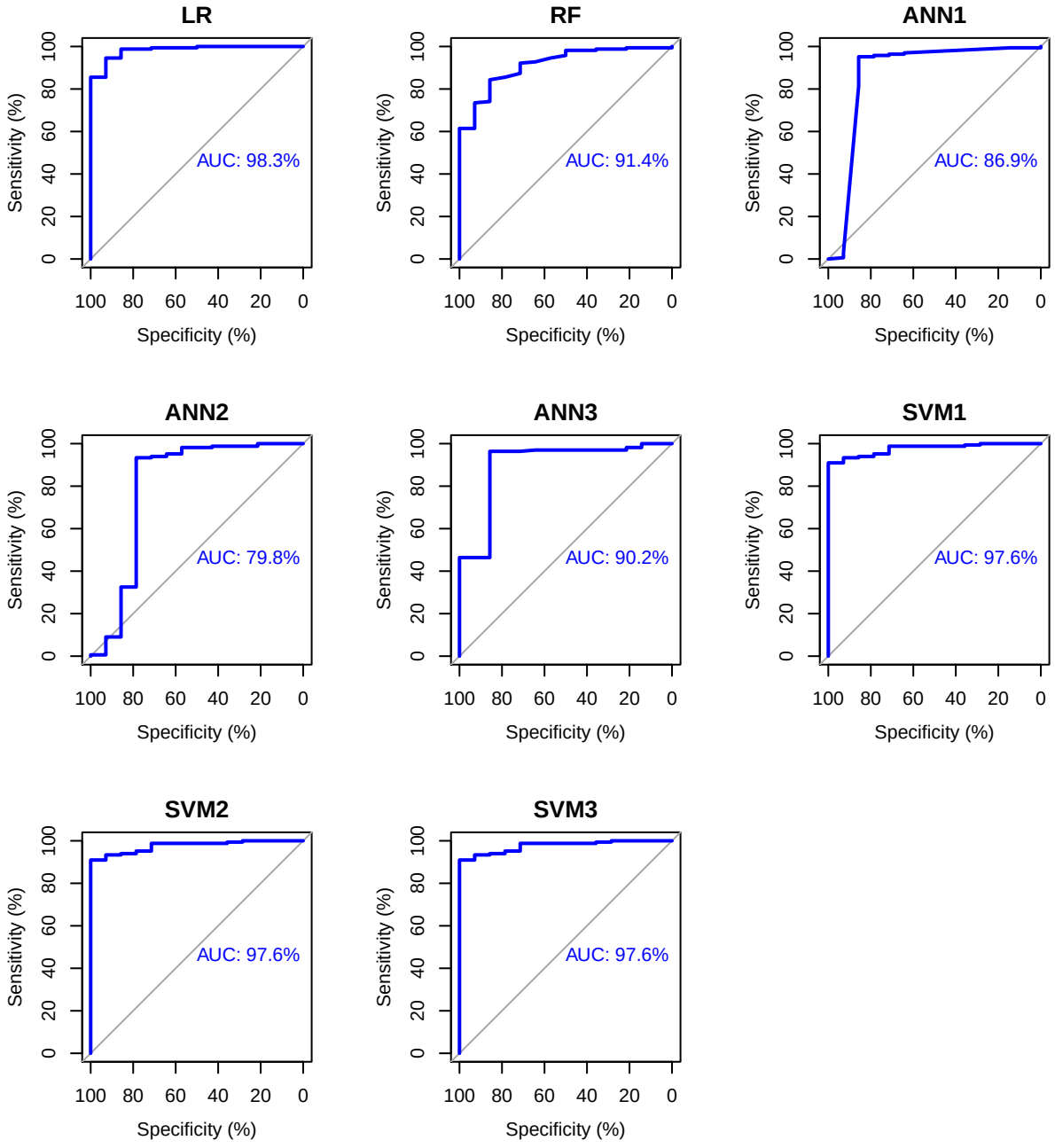

Figure 6: ROC curves for each of the methods used, along with their corresponding AUC.