Project Five for Statistical Data Mining

Raymond Anthony Ford[*]

Due: 26 November 2018

Contents

1 Preliminaries

1.1 Data preparation

We begin this project by bringing the data into `R` and ordering the salary.

```
hr <- read.csv("~/Dropbox/Fall2018/STAT5494/Project5/HR_comma_sep.csv", sep=",",
               header=TRUE)
hr$salary <- factor(hr$salary, levels=c("low", "medium", "high"), ordered=TRUE)
```

Next we check the data for missing values.[1]

```
##   col.number                vname      mode n.levels ncomplete miss.prop
## 1          1   satisfaction_level    double       92     14999         0
## 2          2      last_evaluation    double       65     14999         0
## 3          3       number_project   integer        6     14999         0
## 4          4 average_montly_hours   integer      215     14999         0
## 5          5   time_spend_company   integer        8     14999         0
## 6          6         Work_accident    binary        2     14999         0
```

[*]raford2@miners.utep.edu

[1]The function used to perform this task is in the appendix.

```
## 7           7                   left    binary        2     14999         0
## 8           8 promotion_last_5years    binary        2     14999         0
## 9           9                  sales character       10     14999         0
## 10         10                 salary character        3     14999         0
```

As we can see, there are no missing values present in the data.

1.2 Exploratory data analysis

While there exist many graphical techniques for EDA demonstrated on different websites for this dataset. For this project however, I wanted to explore the relationship between the categorical predictior amongst themselves and with the target `left`.

Firstly, I think it would be interesting to see if there is a statistically significant relationship between `salary` and `sales` (because there are so many type of sales and their numbers look similar). Thus, we have the following χ^2 test and results.

```
chisq.test(table(hr$salary, hr$sales))
```

```
##
##  Pearson's Chi-squared test
##
## data:  table(hr$salary, hr$sales)
## X-squared = 700.92, df = 18, p-value < 2.2e-16
```

From the above output we obtain a significant result for any reasonable level of statistical significance.

Secondly, I think it would be interesting to see if there is a statistically significant relationship between `salary` and `time_spend_company`. Thus, we have the following χ^2 test and results.

```
chisq.test(table(hr$salary, hr$time_spend_company))
```

```
##
##  Pearson's Chi-squared test
##
## data:  table(hr$salary, hr$time_spend_company)
## X-squared = 290.04, df = 14, p-value < 2.2e-16
```

From the above output we obtain a significant result for any reasonable level of statistical significance.

Finally, I think it would be interesting to see if there is a statistically significant relationship between `salary` and `left`. Thus, we have the following χ^2 test and results.

```
chisq.test(table(hr$salary, hr$left))
```

```
##
##  Pearson's Chi-squared test
##
## data:  table(hr$salary, hr$left)
## X-squared = 381.23, df = 2, p-value < 2.2e-16
```

From the above output we obtain a significant result for any reasonable level of statistical significance.

1.3 Data partitioning

Before constructing our models, we need to partition the data into a training and testing dataset. This is accomplished with the following code.

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

2 Classification methods

We will fit five models for this data set and then evaluate the performance of each model. The five models that we will fit are logistic regression, random forests, a generalized additive model, a multivariate adaptive regression splines model, and a project pursuit regression model.

2.1 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 <- left ~ satisfaction_level + last_evaluation + number_project +
    average_montly_hours + time_spend_company + Work_accident +
    promotion_last_5years + sales + salary
X <- model.matrix(as.formula(formula0), data=train)
y <- train$left
X.valid <- model.matrix(as.formula(formula0), data=test)
y.valid <- test$left
Lambda <- seq(0.0001, 0.5, length.out = 500)
L <- length(Lambda)
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] 0.002103607
```

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

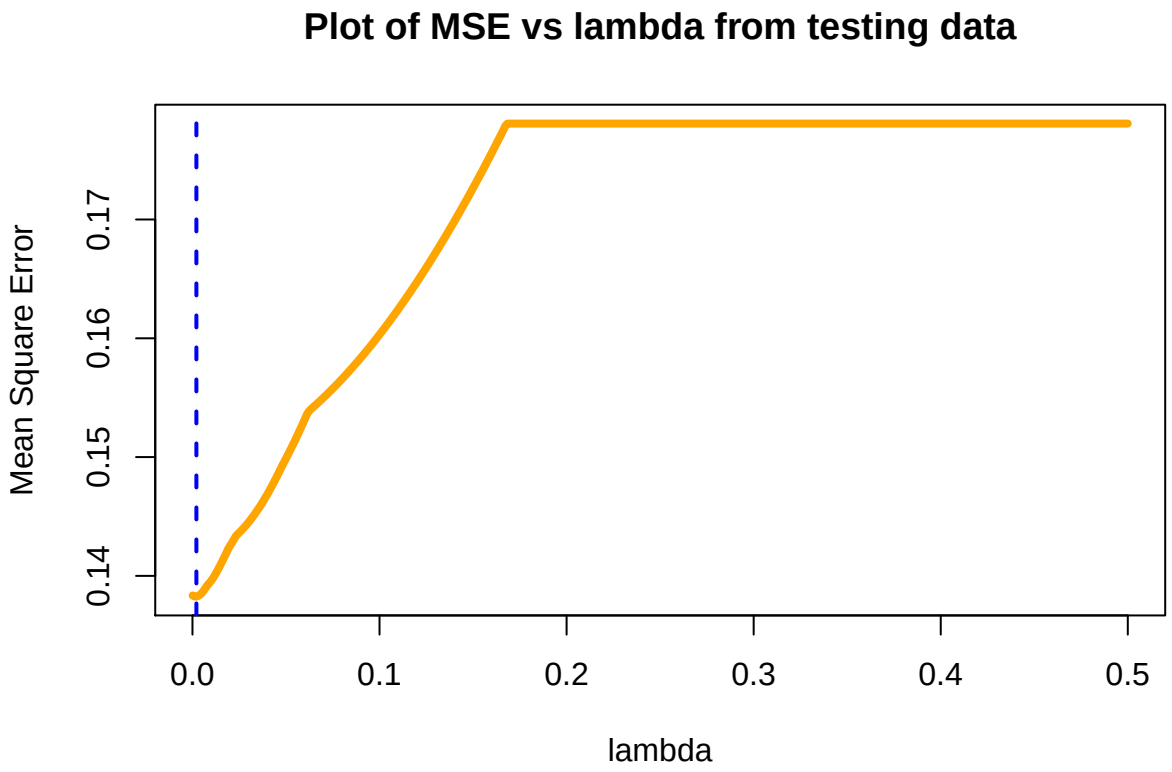

Figure 1: The best lambda for the lowest MSE

```
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$left
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)          -0.251849962
## (Intercept)           .
## satisfaction_level   -3.991617134
## last_evaluation       0.582501581
## number_project       -0.274683264
## average_montly_hours  0.003923193
## time_spend_company    0.247886213
```

```
## Work_accident          -1.428966338
## promotion_last_5years -1.160124544
## saleshr                 0.203814474
## salesIT                -0.095979611
## salesmanagement        -0.351715798
## salesmarketing          .
## salesproduct_mng       -0.064070513
## salesRandD             -0.469536554
## salessales              .
## salessupport            0.040668911
## salestechnical          0.062995470
## salary.L               -1.188969415
## salary.Q               -0.243377128
```

We see that the coefficients for two predictors have shrunk to zero: `salesmarketing` and `salessales`. 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")
```

2.2 Random forest

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)
hr.rf <- randomForest(as.factor(left)~., data=train, importance=TRUE)
hr.rf.pred <- as.data.frame(predict(hr.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(hr.rf, main="Variable importance plot for Random Forest", pch=16, type=2)
```

From Figure 2 we see that the top four important variables are `satisfaction_level`, `time_spend_company`, `number_project`, and `average_montly_hours`. These seem pretty reasonable, and we can learn about their contribution with partial dependence plots.

Now we obtain the partial dependence plots for those four important variables.

```
suppressMessages(library(interpretR, quietly=TRUE))
par(mfrow=c(2, 2))
parDepPlot(x.name="satisfaction_level", hr.rf, data=train,
    main="satisfaction_level", col="blue", lwd=2, xlab="",
    ylab="Partial dependence")
parDepPlot(x.name="time_spend_company", hr.rf, data=train,
    main="time_spend_company", col="blue", lwd=2, xlab="", ylab="")
```

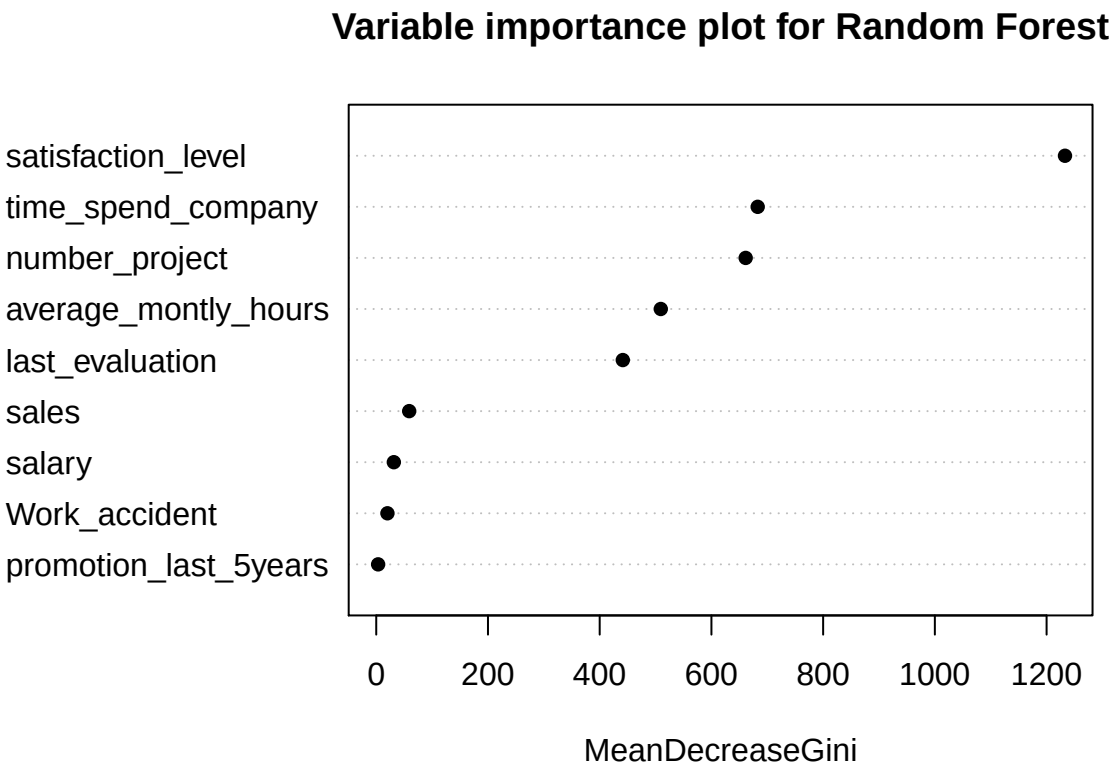

Figure 2: Variable importance plot for Random Forest measured by the mean decrease in Gini.

```
parDepPlot(x.name="number_project", hr.rf, data=train,
    main="number_project", col="blue", lwd=2, xlab="",
    ylab="Partial dependence")
parDepPlot(x.name="average_montly_hours", hr.rf, data=train,
    main="average_montly_hours", col="blue", lwd=2, xlab="", ylab="")
```

From these plots in Figure 3 we note a few observations. As satification level increases it seems less likely that an employee will leave. The longer an employee has been at a company, the more likely they are to leave. This makes sense as employees after a certain number of years need to move onto a new employer if they would like their wages to at least keep up with inflation; most raises do not actually keep up inflation. Finally, employees working too few hours and too many hours seem more likely to leave.

2.3 Generalized additive model (GAM)

Next model is a GAM. Since I was having issues with this model, I just used the default parameter settings. I think the other implementations, say with cross validation, do not work with categorical predictors. So maybe this type of model would not be a wise choice.

```
suppressMessages(library(gam))
fit.gam <- gam(left~., family=binomial, data=train)
pred.gam <- predict(fit.gam, newdata=test, type="response")
```

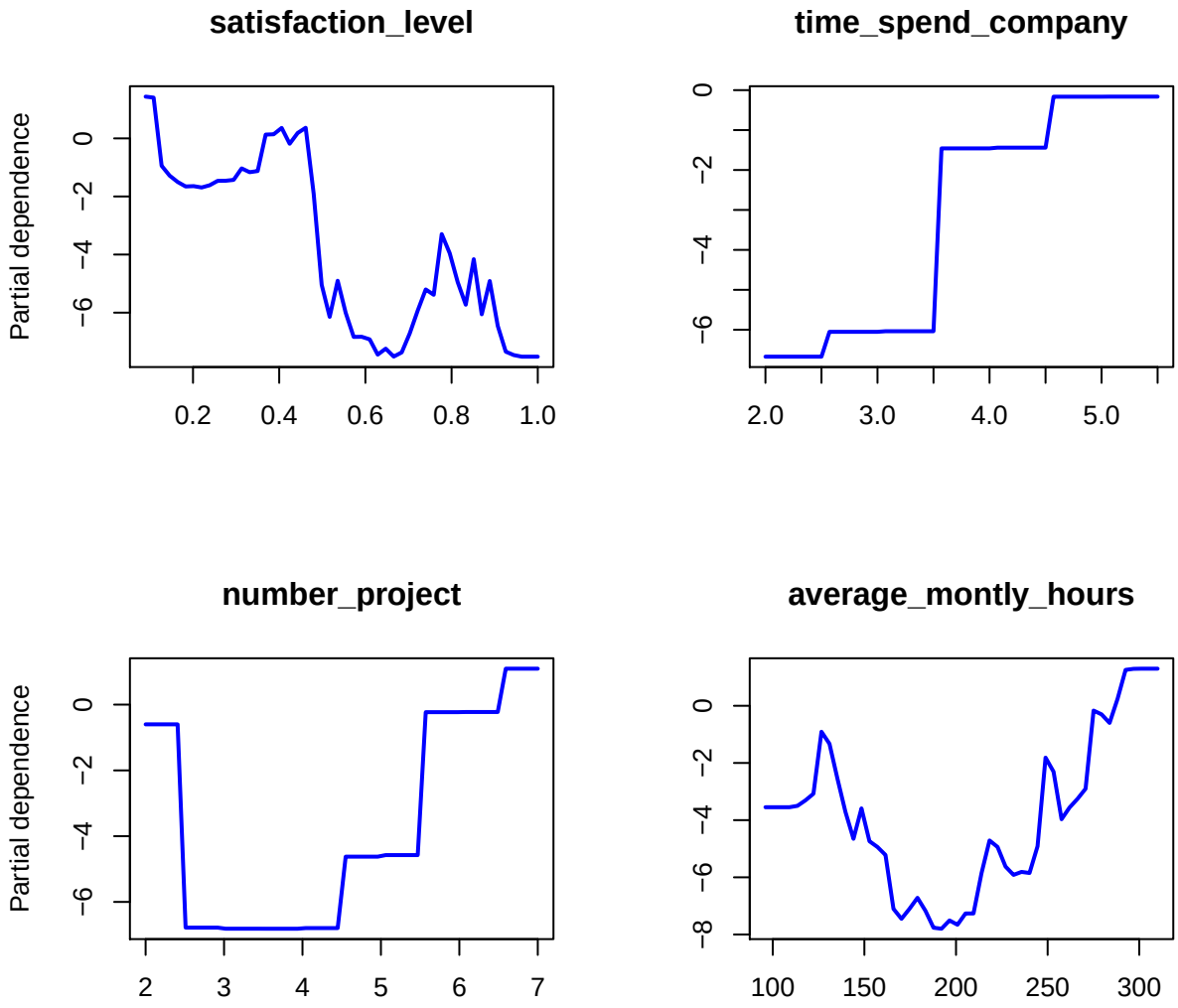

Figure 3: Partial dependence plots for the top four important variables determined by mean decrease in Gini.

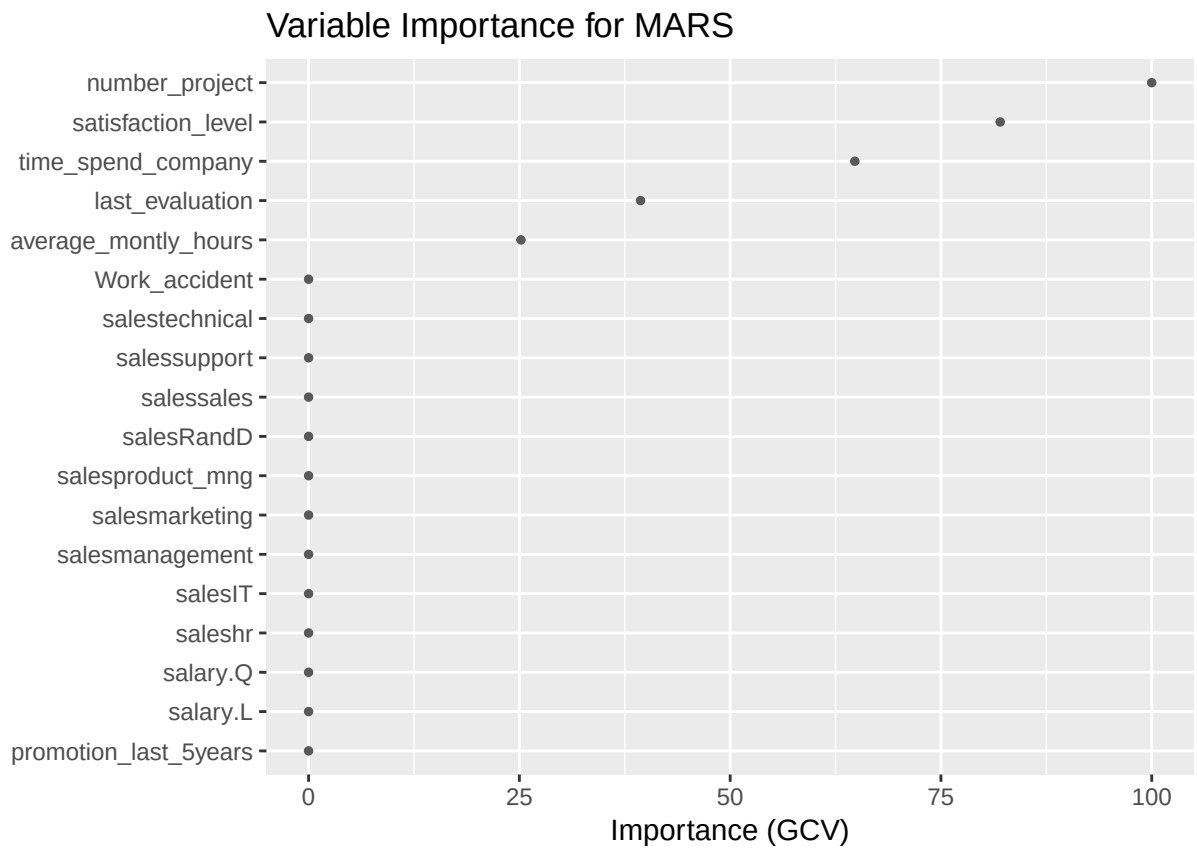

Figure 4: Variable importance plot for MARS

2.4 Multivariate adaptive regression splines (MARS)

The next model that we use is MARS. We fit the model with the following code and use crossvalidation

```
suppressMessages(library(earth, quietly = TRUE))
fit.mars <- suppressWarnings(earth(left ~ .,  data=train, degree=1,
            glm=list(family=binomial(link = "logit")), pmethod="cv", nfold=10))
```

We can plot a variable importance plot with the following code.

```
suppressMessages(library(vip, quietly = TRUE))      # variable importance
suppressMessages(library(ggplot2))
vip(fit.mars, num_features = 40, bar = FALSE) +
    ggtitle("Variable Importance for MARS")
```

We see in Figure 4 that the top four predictors from the random forest variable importance ranking are in the top predictors for MARS.

Next we will plot partial dependence plots for the only two continuous predictors: `satisfation_level` and `last_evaluation`.

```
suppressMessages(library(pdp, quietly = TRUE))
par(mfrow=c(1, 2), mar=rep(4,4), pty="s")
```

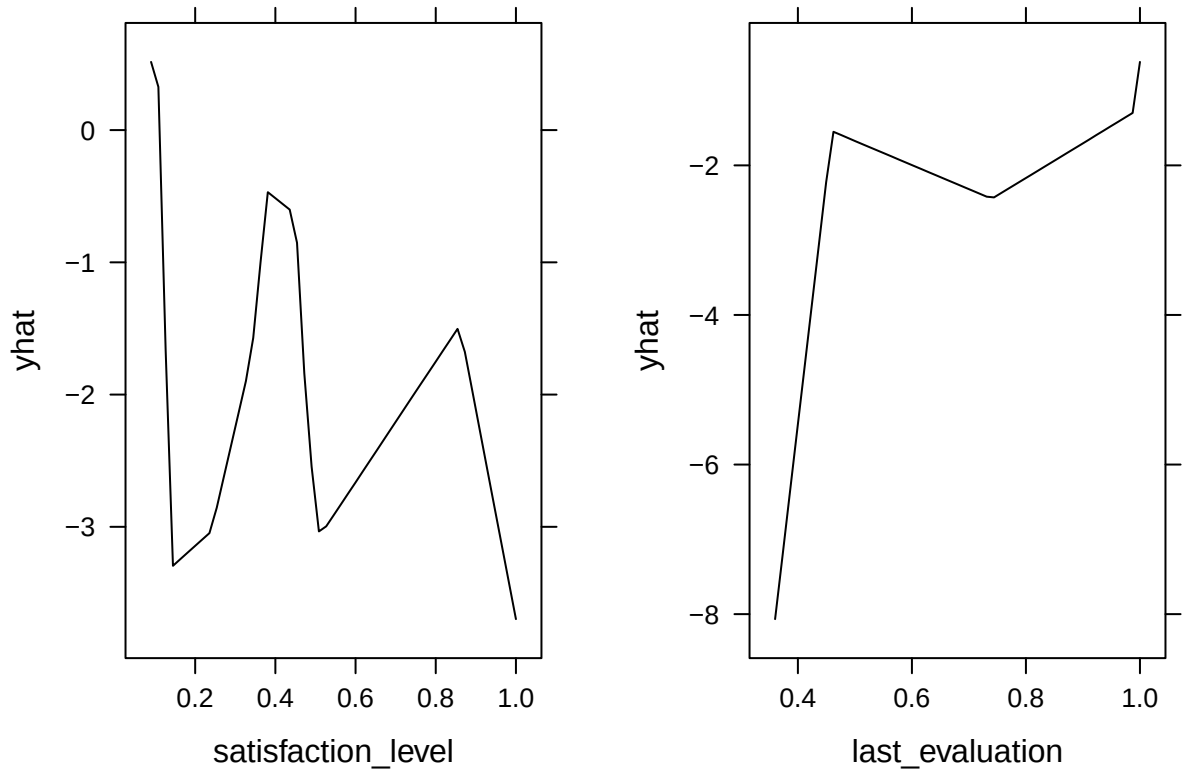

Figure 5: Partial dependence plots for continuous predictors used in MARS

```
plot1 <- partial(fit.mars, pred.var="satisfaction_level", plot=TRUE, trim.outliers=TRUE)
plot2 <- partial(fit.mars, pred.var="last_evaluation", plot=TRUE, trim.outliers=TRUE)
grid.arrange(plot1, plot2, ncol=2)
```

In Figure 5 we see that as satisfaction level increases, the less likely an employee is to leave the company; however, we see that as the last evaluation level increases, the more likely the employee is to leave the company.

```
pred.mars <- predict(fit.mars, newdata=test, type="response")
```

2.5 Project pursuit regression (PPR)

Finally, we build a PPR model and calculate the predictions for the next section.

```
fit.ppr <- ppr(left~., sm.method="supsmu", data=train, nterms=2, max.terms=10, bass=3)
pred.ppr <- predict(fit.ppr, new=test)
```

2.6 Results/conclusion

In Figure 6 we see that Random Forest performed the best with an AUC of 99.1% and MARS was second with an AUC of 97.4%.

For this task I believe that Random Forest and MARS would be good models to use because not only did they have the highest AUCs, but both the Variable Importance Plots and Partial Dependence Plots can go a long way in explaining key results to decision makers. Employee turnover is an expensive problem, and I would assume that management would like more insight into what could be "causing" it other than `That's what my neural network said` would be appreciated.

```
suppressMessages(suppressWarnings(library(pROC, quietly=TRUE)))
par(mfrow=c(3, 2), mar=rep(4,4), pty="s")
plot.roc(test$left, pred.fit, main="LR", percent=TRUE,
         print.auc=TRUE, print.auc.cex=1.0, col="blue")
```

```
## Warning in roc.default(x, predictor, plot = TRUE, ...): Deprecated use
## a matrix as predictor. Unexpected results may be produced, please pass a
## numeric vector.
```

```
plot.roc(test$left, hr.rf.pred, main="RF", percent=TRUE,
         print.auc=TRUE, print.auc.cex=1.0, col="blue")
plot.roc(test$left, pred.gam, main="GAM", percent=TRUE,
         print.auc=TRUE, print.auc.cex=1.0, col="blue")
plot.roc(test$left, pred.mars, main="MARS", percent=TRUE,
         print.auc=TRUE, print.auc.cex=1.0, col="blue")
```

```
## Warning in roc.default(x, predictor, plot = TRUE, ...): Deprecated use
## a matrix as predictor. Unexpected results may be produced, please pass a
## numeric vector.
```

```
plot.roc(test$left, pred.ppr, main="PPR", percent=TRUE,
         print.auc=TRUE, print.auc.cex=1.0, col="blue")
```

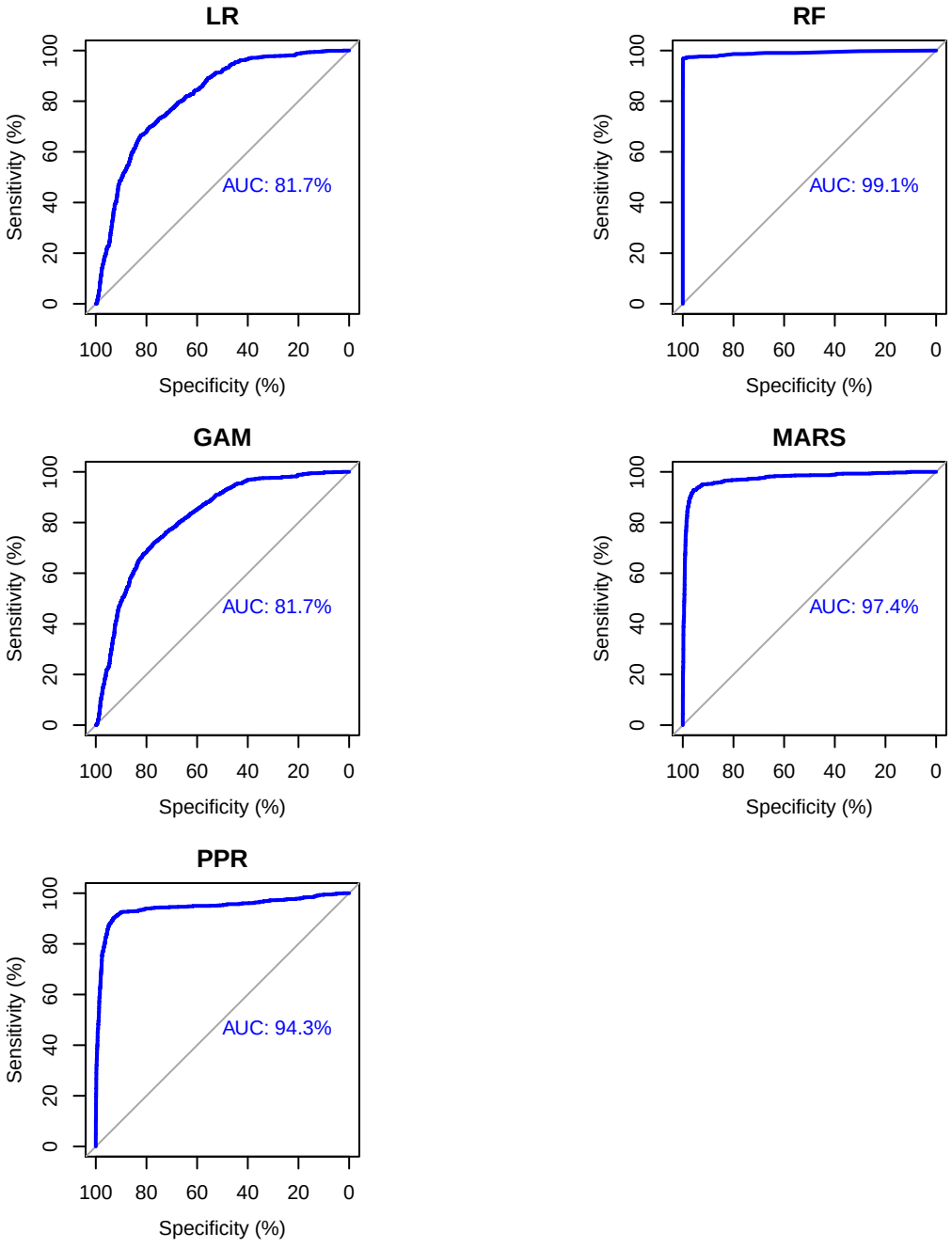

Figure 6: ROC curves for each of the methods used, along with their corresponding AUC.

3 Appendix

3.1 Checking for missing values

```
n <- nrow(hr)
out <- NULL
for (k in 1:ncol(hr)) {
    vname <- colnames(hr)[k]
    x <- as.vector(hr[,k])
    n1 <- sum(is.na(x), na.rm=TRUE)
    n2 <- sum(x=="NA", na.rm=TRUE)
    n3 <- sum(x=='', na.rm=TRUE)
    n4 <- sum(x=="?", na.rm=TRUE)
    n5 <- sum(x=="*", na.rm=TRUE)
    n6 <- sum(x==".", na.rm=TRUE)
    nmiss <- n1 + n2 + n3 + n4 + n5 + n6
    ncomplete <- n - nmiss
    var.type <- typeof(x)
    if (var.type == "integer") {
        if (length(unique(x)) == 2) {
            out <- rbind(out, c(col.number=k, vname=vname, mode="binary",
                                n.levels=length(unique(x)), ncomplete=ncomplete,
                                miss.prop=round(nmiss/n, digits=4)))
        } else {
            out <- rbind(out, c(col.number=k, vname=vname, mode=typeof(x),
                                n.levels=length(unique(x)), ncomplete=ncomplete,
                                miss.prop=round(nmiss/n, digits=4)))
        }
    } else {
        out <- rbind(out, c(col.number=k, vname=vname, mode=typeof(x),
                            n.levels=length(unique(x)), ncomplete=ncomplete,
                            miss.prop=round(nmiss/n, digits=4)))
    }
}
out <- as.data.frame(out)
row.names(out) <- NULL
out
```