Stats 202 Kaggle Competition

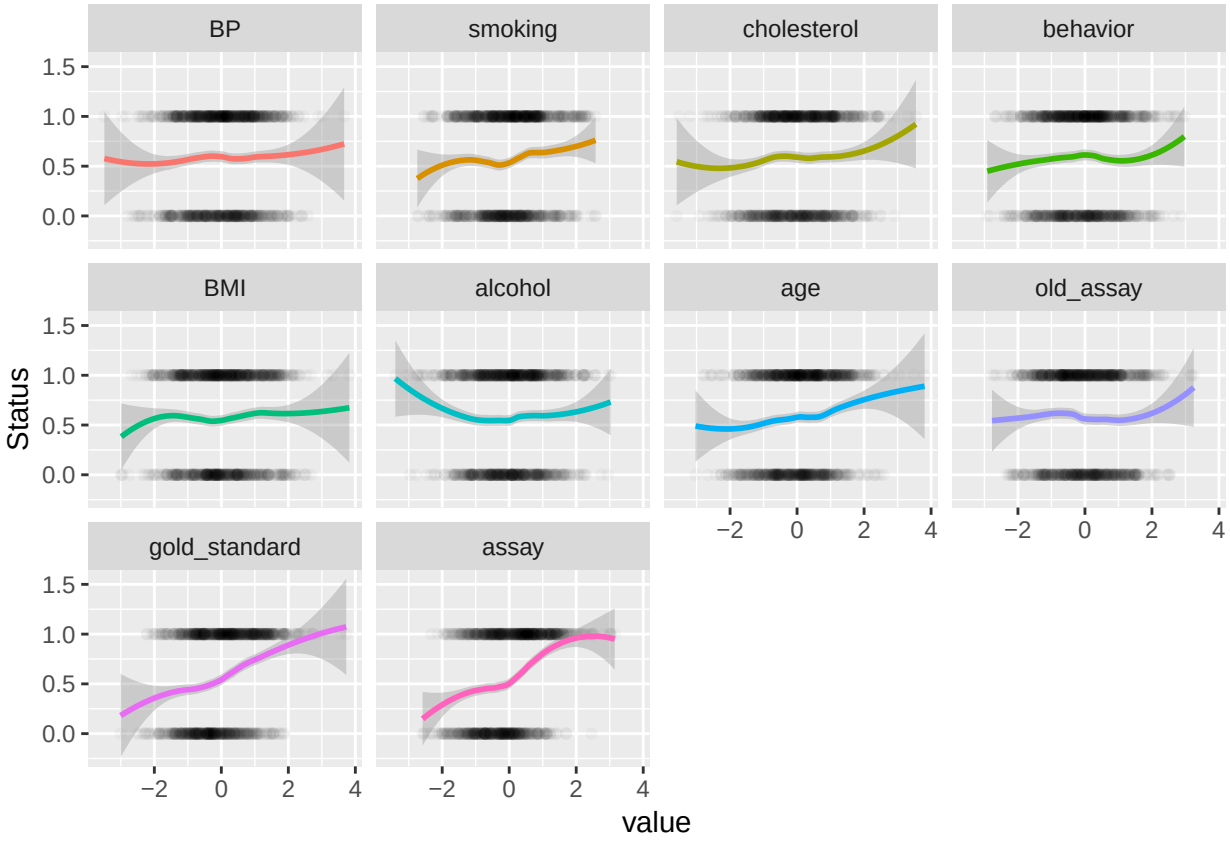
Nicholas Allen (Kaggle ID naallen)

February 19, 2019

My first attempt at tackling the problem involved fitting a GAM to the data. To determine which predictors were likely the most important, I imported the data and visualized each predictor against the smoothed Status percentage:

```
library(reshape2)
library(ggplot2)

train.data = read.csv("train_data.csv", header = TRUE)
test.data = read.csv("test_data.csv", header = TRUE)
melted = melt(train.data, id.vars = "Status")
melted = melted[melted$variable != "Id",]
ggplot(data = melted, aes(y=Status, x=value)) +
  geom_point(alpha=0.03) +
  stat_smooth(aes(color = variable), method = "loess", show.legend = FALSE) +
  facet_wrap(variable ~ . )
```



Noting which predictors appeared to be correlated to the data and which didn't, I built up a GAM with splines for predictors that appeared non-linear, and linear fits for factors that appeared linear. I also used ANOVA to evaluate the significance of each variable added to the model, and to justify omitting the variables that I omitted, and tuned the degrees of freedom of each spline using 10-fold CV. I ended up with the following fit:

```
library(gam)
```

```
## Loading required package: splines
```

```
## Loading required package: foreach
```

```
## Loaded gam 1.16
```

```
fit.gam = gam(Status~BP+s(smoking,7)+behavior+age+gold_standard+s(assay,7)+s(alcohol,3),
              family="binomial", data=train.data)
pred.train = predict(fit.gam, train.data, type="response") > 0.5
print(sprintf("Training error: %s", mean(pred.train != train.data$Status)))
```

```
## [1] "Training error: 0.31"
```

```
pred.test = predict(fit.gam, test.data, type="response") > 0.5
predictions.gam = data.frame(test.data$Id, as.logical(pred.test))
names(predictions.gam) = c("Id", "Category")
write.csv(predictions.gam, "predictions_gam.csv", row.names = FALSE, quote = FALSE)
```

However, this fit performed fairly badly in the Kaggle evaluation, and had a rather bad training error. Since the response is class-based, I decided to move on to using an SVM. I fit linear, polynomial, and radial SVMs to the data, and tuned the cost of each, and degree of the polynomial SVM, using the tune function. The best model I obtained using this method was with the radial SVM using a cost of 0.7, the code for which is given as follows:

```
library(e1071)
fit.radsvm = svm(as.factor(Status)~., data=train.data, kernel="radial", cost=0.7)
pred.train = predict(fit.radsvm, train.data)
print(sprintf("Training error: %s", mean(pred.train != train.data$Status)))
```

```
## [1] "Training error: 0.257"
```

```
pred.test = predict(fit.radsvm, test.data)
predictions.rad = data.frame(test.data$Id, pred.test == 1)
names(predictions.rad) = c("Id", "Category")
write.csv(predictions.rad, "predictions_rad.csv", row.names = FALSE, quote = FALSE)
```

I also fit and tuned a polynomial SVM, and while it had a good CV error, the overall training error was worse than the radial SVM when I calculated total training error.

```
library(e1071)
fit.polysvm = svm(as.factor(Status)~., data=train.data, kernel="polynomial", cost=0.1,
                  degree=1)
pred.train = predict(fit.polysvm, train.data)
print(sprintf("Training error: %s", mean(pred.train != train.data$Status)))
```

```
## [1] "Training error: 0.335"
```

```
pred.test = predict(fit.polysvm, test.data)
predictions.poly = data.frame(test.data$Id, pred.test == 1)
names(predictions.poly) = c("Id", "Category")
write.csv(predictions.poly, "predictions_poly.csv", row.names = FALSE, quote = FALSE)
```

I ended up going with the radial SVM, with cost=0.7, as my final submission, attaining the following score:

predictions_rad.csv 0.68333

a minute ago by Nicholas Allen

add submission details

#