Business Value Case Study

The purpose of this tutorial is to show the business value that analytical work adds to an insurance carrier. Reading this will not necessarily help you be a better competitor in this competition. Our hope is to put the work you are doing for the hackathon in context so you gain an understanding of why this type of work is beneficial.

Let's fit a couple of models, one which will be better than the other.

Load Packages

```
In [1]: library(gbm)
```

```
Loading required package: survival
Loading required package: lattice
Loading required package: splines
Loading required package: parallel
Loaded gbm 2.1.1
```

```
In [2]: library(ggplot2)
```

```
Warning message:
: package 'ggplot2' was built under R version 3.2.3
```

Set Random Seed

Set a random seed to allow reproducability of these results.

```
In [3]: set.seed(8675309)
```

Load Data

Make sure this is analysis is being done from within the directory where the data is located. We will only use the training data for this illustration as a response variable is required for measurement.

```
In [4]: train = read.csv('train.csv')
```

Error Metric

R-Bloggers has a nice article on explaining the Log Loss function which can be read here (http://www.r-bloggers.com/making-sense-of-logarithmic-loss/).

In [5]:

```
LogLossBinary = function(actual, predicted, eps = 1e-15) {
    predicted = pmin(pmax(predicted, eps), 1-eps)
    - (sum(actual * log(predicted) + (1 - actual) * log(1 - predicted))) / length(actual)
}
```

Create Data Subset

In [6]:

```
dataSubsetProportion = .4
randomRows = sample(1:nrow(train), floor(nrow(train) * dataSubsetProportion))
trainingHoldoutSet = train[randomRows, ]
trainingNonHoldoutSet = train[!(1:nrow(train) %in% randomRows), ]
```

Fit Worse Model

In [7]:

```
glmModel = glm(Response ~ Var1 + Var2 + NVVar1, data = trainingNonHoldoutSet, family = "binomial")
```

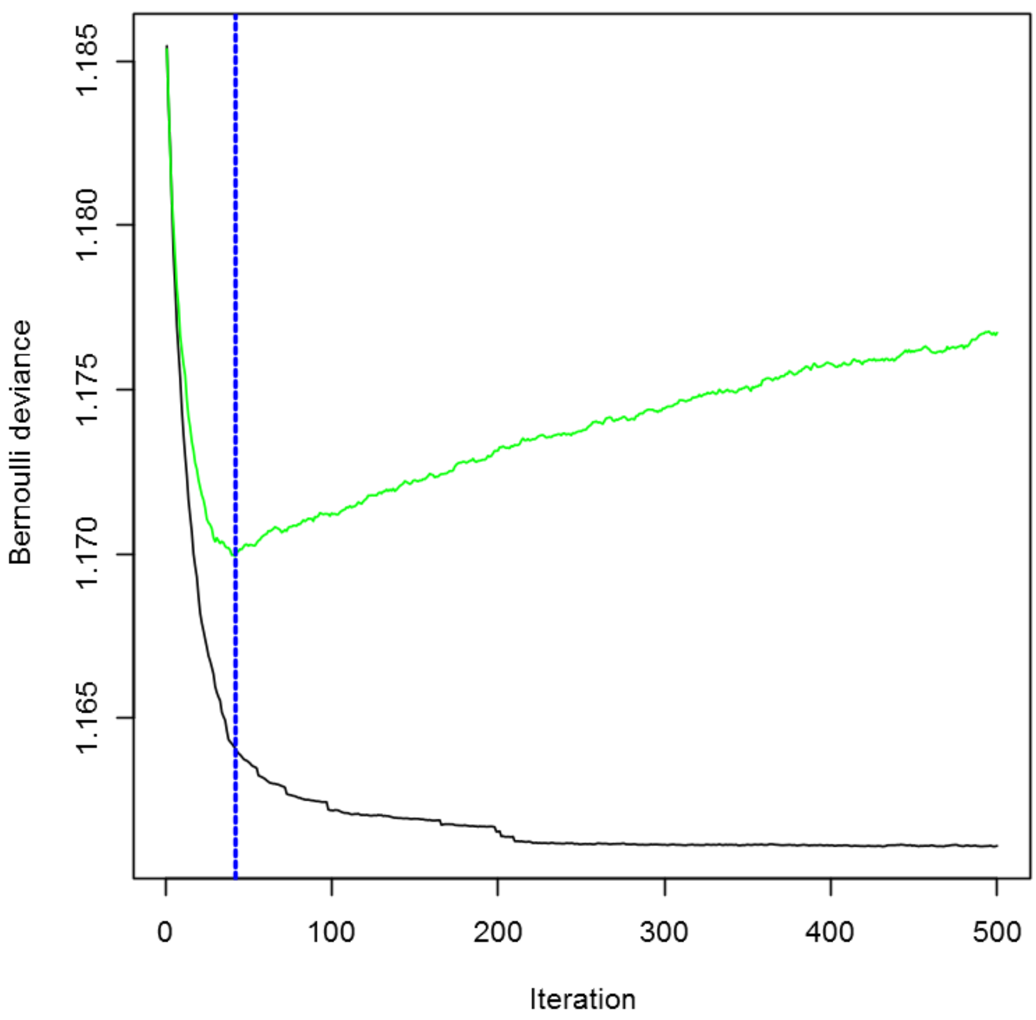
Fit Better Model

Also find the best number of trees for predictions via cross validation.

In [8]:

```
gbmModel = gbm(formula = Response ~ . - RowID - ModelYear,
               distribution = "bernoulli",
               data = trainingNonHoldoutSet,
               n.trees = 500,
               shrinkage = .1,
               n.minobsinnode = 20,
               cv.folds = 5,
               n.cores = 1) # n.cores = 1 forces reproducible results wi
thout having to set gbm seed
gbmTrees = gbm.perf(gbmModel)
```

```
Using cv method...
```



Make Predictions

Find predictions for both models

In [9]:

```
glmPredictions = predict(object = glmModel,
                         newdata = trainingHoldoutSet,
                         type = "response")

gbmPredictions = predict(object = gbmModel,
                         newdata = trainingHoldoutSet,
                         n.trees = gbmTrees,
                         type = "response")
```

Assess Model Performance

In [10]:

```
LogLossBinary(trainingHoldoutSet$Response, glmPredictions)
LogLossBinary(trainingHoldoutSet$Response, gbmPredictions)
```

Out[10]:

```
0.591403827220903
```

Out[10]:

```
0.583614229340114
```

The GBM had a lower log loss, meaning it performed better according to this metric.

Inspect Spread in Predictions by Model

In [11]:

```
allLevels = lapply(1:10, function(x){rep(x, nrow(trainingHoldoutSet) / 10)})
decileNumber = do.call(c, allLevels)
response = trainingHoldoutSet$Response
```

Split the glm and gbm predictions into 10 sorted sets of data and view their means.

In [12]:
```
data.frame(glmDeciles = as.numeric(tapply(response[order(glmPrediction
s)], as.factor(decileNumber), mean)),
           gbmDeciles = as.numeric(tapply(response[order(gbmPrediction
s)], as.factor(decileNumber), mean)))
```

Out[12]:

	glmDeciles	gbmDeciles
1	0.22225	0.19175
2	0.23325	0.22275
3	0.2435	0.225
4	0.26325	0.23675
5	0.2865	0.25525
6	0.27425	0.27675
7	0.29675	0.28775
8	0.33525	0.31925
9	0.33575	0.37625
10	0.327	0.42625

Since we have a binary response in this setting, models that are performing better will predict closer to 0 for low deciles and closer to 1 for high deciles. This is another way of showing the GBM is outperforming the GLM.

Business Example

Insurance costs to the insurer are the product of the probility of having a claim (or often the predicted number of claims) and the expected amount of that claim.

Suppose that, on average, claims cost the insurance carrier $2000.

Insurance carriers often gross their estimates of cost up for expenses and desired profit margin to determine price of insurance.

In practice, we can ignore this grossing up for expenses and desired profit margin and pretend an insurance company would charge the prediction of claims costs resulting from the model (the model's prediction times the average claim cost of $2000)

Suppose there are only 2 insurance companies in the marketplace and the 40K vehilces in our holdout set we created are the only vehicles in the entire market place.

The two insurance carriers determine their price for each vehicle insured by multiplying expected claim cost ($2000) by the predictions of the probility of having a claim from the glm (for company 1) and gbm (for company 2) models above.

Suppose also that each vehicle's owner decides to buy their insurance from the company with the lower price. Then company 1 will write the business when the glm's prediction is lower and company 2 will write it when the gbm's prediction is lower.

Which insurance company will have better financial performance?

In [13]:

```
companySelected <- as.numeric(glmPredictions > gbmPredictions) + 1
dataFrameProfitabilityAnalysis <- data.frame(company1Premium = glmPredic
tions * 2000,
                                             company2Premium = gbmPredic
tions * 2000,
                                             costs = trainingHoldoutSet
$Response * 2000)
```

In [14]:

```
table(companySelected)
```

Out[14]:

```
companySelected
    1     2 
18607 21393 
```

In [15]:

```
colnamesForLooping = c("company1Premium", "company2Premium")
lossRatios = sapply(1:2, function(i){
    indices = which(companySelected == i)
    columnForPremium = colnamesForLooping[i]
    sum(dataFrameProfitabilityAnalysis$costs[indices]) /
        sum(dataFrameProfitabilityAnalysis[[columnForPremium]][indices])
})
names(lossRatios) = c("lossRatioCompany1", "lossRatioCompany2")
print(lossRatios)
```

```
lossRatioCompany1 lossRatioCompany2 
        1.1989288         0.9709553 
```

Company 1 with the worse model loses about 20 cents on every dollar of premium they collect (relative to their targeted profit margin)

Company 2 with the better model hits their targeted profit margin (their loss ratio is not greater than 1)

Even a seemingly small difference between models can have a huge business impact!