Final Project

Zack Nadrich

12/2/2019

Data Description

The dataset used for this project, Climate Model Simulations Crashes, is from the UCI Machine Learning Repository. The goal for this dataset is determine which combinations of parameter inputs for the climate model simulation causes the simulation to later crash.

In the frame of scientific computing, we are aiming to reduce wasted computation time for expensive supercomputers as well as reduce debugging time that programmers have to spend in fixing the climate model simulations.

The dataset includes 180 simulations under 3 latin hypercube studies, for a total of 540 observations. The target variable, `outcome`, is coded as `1=success, 0=failure` with an average success rate of 0.915. In addition, there are 18 model parameters that are each scaled to an interval of `[0, 1]`.

Below are the names of each parameter - note that definitions are available in the paper 'Failure Analysis of Parameter-Induced Simulation Crashes in Climate Models'

```
names(sim)
```

```
##  [1] "Study"                "Run"                  "vconst_corr"
##  [4] "vconst_2"             "vconst_3"             "vconst_4"
##  [7] "vconst_5"             "vconst_7"             "ah_corr"
## [10] "ah_bolus"             "slm_corr"             "efficiency_factor"
## [13] "tidal_mix_max"        "vertical_decay_scale" "convect_corr"
## [16] "bckgrnd_vdc1"         "bckgrnd_vdc_ban"      "bckgrnd_vdc_eq"
## [19] "bckgrnd_vdc_psim"     "Prandtl"              "outcome"
```

Classification Algorithms

A total of 5 clasification algorithms were evaluated for this dataset

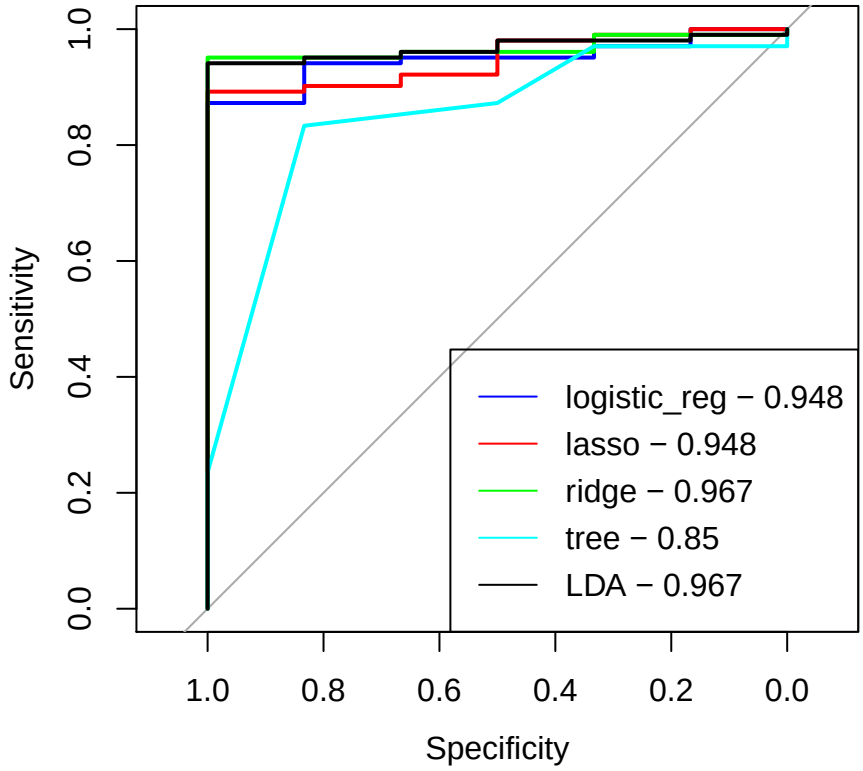
- Logistic regression
- Ridge regression
  - Lambda parameter chosen by CV
- Lasso
  - Lambda parameter chosen by CV
- Linear Discriminant Analysis (LDA)
- Classification tree
  - CV was performed to determine best tree complexity

In preparation to evaluate the algorithms against each other, the data was split into a 80%/20% train/validation split and evaluation was based on the AUC metric.

Below we can see that each algorithm performs similarly, except decision trees.

```
auc <- sapply(fits, function(x) round(x$eval, 3))
plot(fits$logistic_reg$roc, col='blue')
plot(fits$lasso$roc, col='red', add=TRUE)
plot(fits$ridge$roc, col='green', add=TRUE)
plot(fits$tree$roc, col='cyan', add=TRUE)
plot(fits$LDA$roc, col='black', add=TRUE)
legend(
```

```
  'bottomright',
  paste(names(fits), auc, sep=' - '),
  col=c('blue', 'red', 'green', 'cyan', 'black'),
  lty=1
)
```



Model Inference

Logistic Regression

One benefit of logistic regression is the ease in statistical inference, such as the ability to understand the effect each predictor has on the outcome and statistical testing for predictor significance. In our case, we would like to detect which of the input parameters lead to higher chance of a simulation crash, so our needs are met.

The drawbacks of logistic regression would be that we are assuming a linear relationship of the log-odds and predictors. Should this assumption not hold true, our model will not be flexible enough to detect the true relationship.

As we will see in more clearly in both Lasso and Ridge regression, only a few of the climate model parameter inputs are considered statistically significant in the logistic regression. Based on the performance of Ridge and Lasso, indeed these models will be preferable as they will be more intepretable and as performant as including every climate model parameter.

```
summary(fits$logistic_reg$fit)
```

```
##
## Call:
## outcome ~ . - Run - Study
```

```
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.5852   0.0007   0.0077   0.0769   1.5165
##
## Coefficients:
##                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)            14.15003    3.38029   4.186 2.84e-05 ***
## vconst_corr           -12.27530    2.45648  -4.997 5.82e-07 ***
## vconst_2              -11.87397    2.42282  -4.901 9.54e-07 ***
## vconst_3               -1.76845    1.20233  -1.471  0.14133
## vconst_4                1.59687    1.13249   1.410  0.15852
## vconst_5                4.50234    1.37347   3.278  0.00105 **
## vconst_7                2.83448    1.22119   2.321  0.02028 *
## ah_corr                 0.74027    1.14547   0.646  0.51811
## ah_bolus                1.18240    1.16662   1.014  0.31081
## slm_corr                1.44478    1.20840   1.196  0.23185
## efficiency_factor      -2.24264    1.20757  -1.857  0.06329 .
## tidal_mix_max          -0.31666    1.11504  -0.284  0.77641
## vertical_decay_scale   -2.28795    1.21514  -1.883  0.05972 .
## convect_corr           -6.61255    1.62455  -4.070 4.69e-05 ***
## bckgrnd_vdc1            6.20253    1.54499   4.015 5.95e-05 ***
## bckgrnd_vdc_ban         1.25626    1.20895   1.039  0.29874
## bckgrnd_vdc_eq          3.48899    1.29107   2.702  0.00688 **
## bckgrnd_vdc_psim        3.22615    1.38255   2.333  0.01962 *
## Prandtl                -0.01799    1.04683  -0.017  0.98629
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 266.540  on 431  degrees of freedom
## Residual deviance:  76.169  on 413  degrees of freedom
## AIC: 114.17
##
## Number of Fisher Scoring iterations: 9
```

Ridge Regression

In ridge regression we are sacrificing more bias for less variance in model fit. This sacrifice in bias also depends on the relationship being linear, just as with logistic regression. Note that does not shrink coefficient values to 0, so we will not be performing variable selection. We shall see in Lasso that indeed only a few input paramters should be necessary to determine model simulation crashes.

```
fits$ridge$params$coefs
```

```
##                                1
## (Intercept)          2.283223e+00
## vconst_corr         -1.197338e-03
## vconst_2            -1.204089e-03
## vconst_3            -6.363599e-05
## vconst_4             2.424593e-04
## vconst_5             3.467661e-04
## vconst_7             2.839284e-04
## ah_corr             -2.650430e-05
```

```
## ah_bolus              7.296650e-05
## slm_corr              1.745738e-04
## efficiency_factor    -1.968280e-04
## tidal_mix_max        -4.104329e-05
## vertical_decay_scale -2.162420e-04
## convect_corr         -8.747783e-04
## bckgrnd_vdc1          7.110031e-04
## bckgrnd_vdc_ban      -4.751810e-05
## bckgrnd_vdc_eq        2.786100e-04
## bckgrnd_vdc_psim      2.568245e-04
## Prandtl              -1.557624e-04
```

Lasso

Unlike ridge regression, Lasso does indeed shrink coefficient values to 0. Effectively, this results in a feature selection algorithm that will output which input parameters have the most influence on simulation crashes.

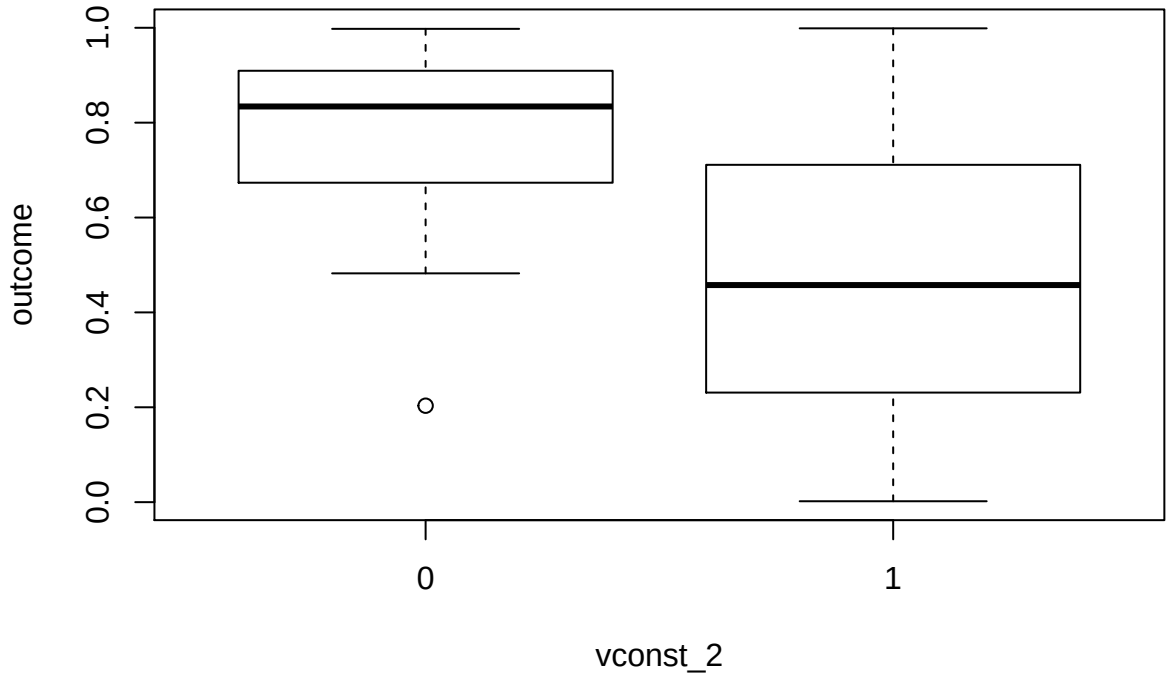
Out of all models evaluated, my preferred is Lasso due to the output below. Simplicity is key here - only two model input parameters determine simulation crashes! Based on definition from the original publications, these are both variable viscosity parameters.

```
fits$lasso$params$coefs
```

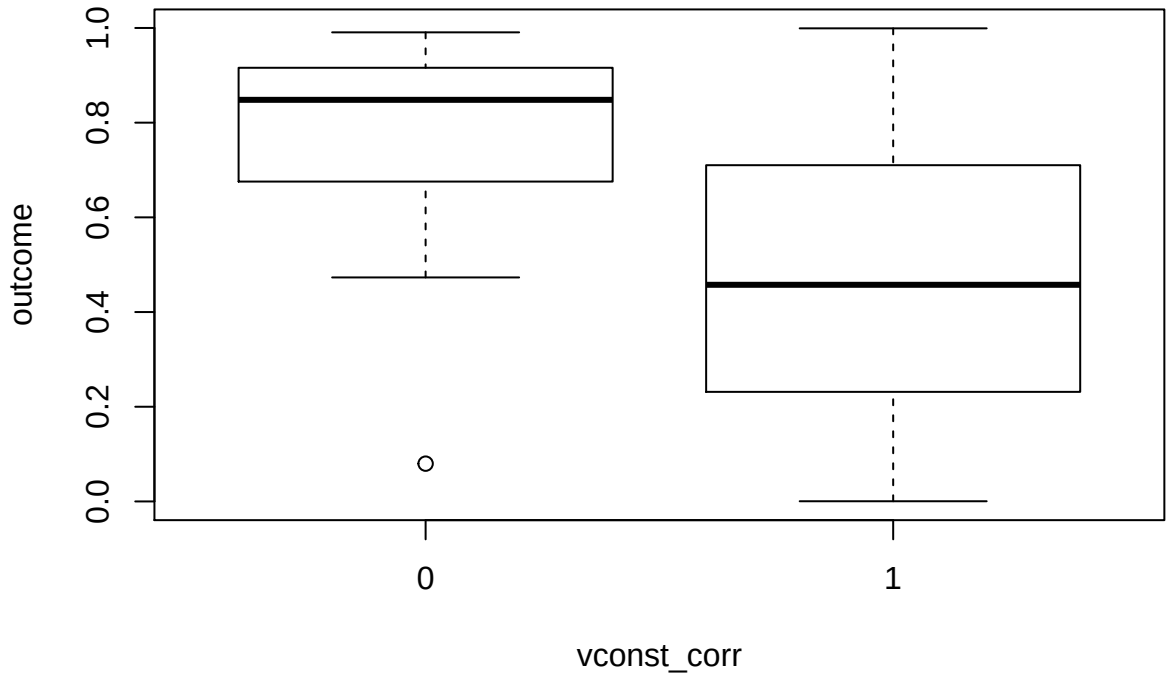
```
##                             1
## (Intercept)          2.9224476
## vconst_corr         -0.5993460
## vconst_2            -0.6173287
## vconst_3             0.0000000
## vconst_4             0.0000000
## vconst_5             0.0000000
## vconst_7             0.0000000
## ah_corr              0.0000000
## ah_bolus             0.0000000
## slm_corr             0.0000000
## efficiency_factor    0.0000000
## tidal_mix_max        0.0000000
## vertical_decay_scale 0.0000000
## convect_corr         0.0000000
## bckgrnd_vdc1         0.0000000
## bckgrnd_vdc_ban      0.0000000
## bckgrnd_vdc_eq       0.0000000
## bckgrnd_vdc_psim     0.0000000
## Prandtl              0.0000000
```

Indeed below we can see that the distribution for the two parameters are different by simulation outcome, where a higher value is more likely to cause a simulation crash.

```
plot(sim$outcome, sim$vconst_2, ylab='outcome', xlab='vconst_2')
```




```
plot(sim$outcome, sim$vconst_corr, ylab='outcome', xlab='vconst_corr')
```



Linear Discriminate Analysis

One benefit of LDA is that the parameterization is intuitive and easy to understand. On the otherhand, due to this parameterization we are making several strong assumptions.

LDA assumptions

- Features are distributed as multivariate normal for each class
- Covariance matrices are homogenous across classes

Given that these assumptions hold true, we will be able to make inference based on the mean vector for each class.

As we have seen before in Lasso, `vconst_corr` and `vconst_2` have a large amount of weight placed on them in the linear discriminant coefficients. Additionally, we can see that these two variables have drastically different means between outcome.

```
fits$LDA$fit
```

```
## Call:
## outcome ~ . - Run - Study
##
## Prior probabilities of groups:
##          0          1
## 0.09259259 0.90740741
##
## Group means:
##   vconst_corr  vconst_2  vconst_3  vconst_4  vconst_5  vconst_7   ah_corr
## 0   0.7930669 0.7826211 0.5199245 0.4359020 0.4279348 0.4399434 0.5072566
```

```
## 1   0.4819447 0.4697452 0.5033899 0.4989096 0.5180383 0.5137232 0.5003660
##    ah_bolus  slm_corr efficiency_factor tidal_mix_max vertical_decay_scale
## 0 0.4834388 0.4530363         0.5463350     0.5021271            0.5551992
## 1 0.5024001 0.4983952         0.4951834     0.4914606            0.4990096
##   convect_corr bckgrnd_vdc1 bckgrnd_vdc_ban bckgrnd_vdc_eq
## 0    0.7068259    0.3255917       0.5195261       0.434498
## 1    0.4795190    0.5103404       0.5071780       0.506890
##   bckgrnd_vdc_psim   Prandtl
## 0        0.4550142 0.5435159
## 1        0.5217489 0.5030389
##
## Coefficients of linear discriminants:
##                                LD1
## vconst_corr          -2.178716674
## vconst_2             -2.216403595
## vconst_3             -0.130536433
## vconst_4              0.311333758
## vconst_5              0.675433268
## vconst_7              0.468932941
## ah_corr               0.009843911
## ah_bolus              0.101453677
## slm_corr              0.390382029
## efficiency_factor    -0.202231269
## tidal_mix_max        -0.028590793
## vertical_decay_scale -0.398824676
## convect_corr         -1.605766100
## bckgrnd_vdc1          1.346901707
## bckgrnd_vdc_ban      -0.074041549
## bckgrnd_vdc_eq        0.586507141
## bckgrnd_vdc_psim      0.463602322
## Prandtl              -0.217668237
```

Classification tree

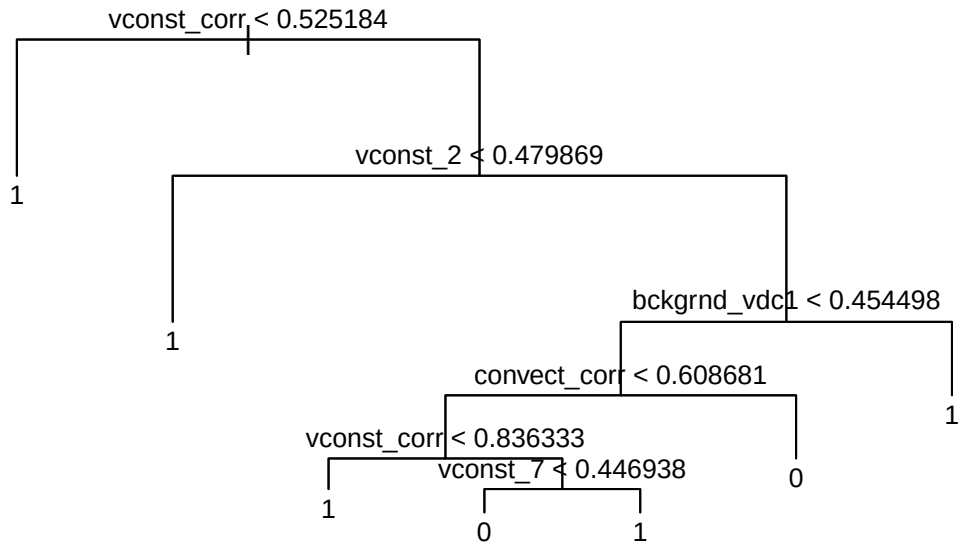
Classification trees offer a non-parametric model with no assumptions on the data. This flexibility, however, will come at a cost in higher variance. As additional benefits, classification trees will offer variable selection due to the greedy split strategy and a great visualization for model inference.

Of all the algorithms used, classification trees had the worst AUC. This is not to say that the tree is not worthy of being examined as it has the easiest interpretation. Indeed `vconst_corr` and `vconst_2` are picked up as important predictors, along with some other parameters. From the diagram we can see that the tree has segmented simulation crashes as a result of `vconst_corr` > 0.525

```
summary(fits$tree$fit)
```

```
##
## Classification tree:
## outcome ~ . - Run - Study
## Variables actually used in tree construction:
## [1] "vconst_corr"  "vconst_2"     "bckgrnd_vdc1" "convect_corr"
## [5] "vconst_7"
## Number of terminal nodes:  7
## Residual mean deviance:  0.201 = 85.44 / 425
## Misclassification error rate: 0.03009 = 13 / 432
```

```
plot(fits$tree$fit)
text(fits$tree$fit, cex=.8)
```



Conclusion

Failures of simulation models can be costly from both a time and computation perspective. Climate models are some of the most demanding simulations out there and are often run on expensive supercomputers. This case study provides an opportunity to address simulation crashes as a result of invalid or buggy input parameters.

Due to the nature of this problem, an easy to understand solution is preferrable. As a result, our most preferred method would be Lasso due to the shrinkage penalty. The Lasso model fit reduces our feature set to 2, `vconst_corr` and `vconst_2`. With this information, the computational scientists would be able to troubleshoot the simulation code or place restrictions on input parameters in order to avoid costly simulation crashes.