MA679 Midterm Exam

Ryan O'Dea

3/27/2021

Context

A company wants to hire data scientists from pool of people enrolled in the courses conducted by the company. The company wants to know which of these candidates are looking to change their job. Information related to demographics, education, experience are in hands from candidates signup and enrollment. In this exam, your goal is to predict if the candidate is looking for a new job or will work for the current company.

- uid : Unique ID for candidate
- city: City code
- city_dev_index : Development index of the city (scaled)
- gender: Gender of candidate
- relevant_experience: Relevant experience of candidate
- enrolled_university: Type of University course enrolled if any
- education_level: Education level of candidate
- major_discipline :Education major discipline of candidate
- experience_years: Candidate total experience in years
- company_size: No of employees in current employer's company
- company_type : Type of current employer
- lastnewjob: Difference in years between previous job and current job
- training_hours: training hours completed
- change_job: 0 – Not looking for job change, 1 – Looking for a job change

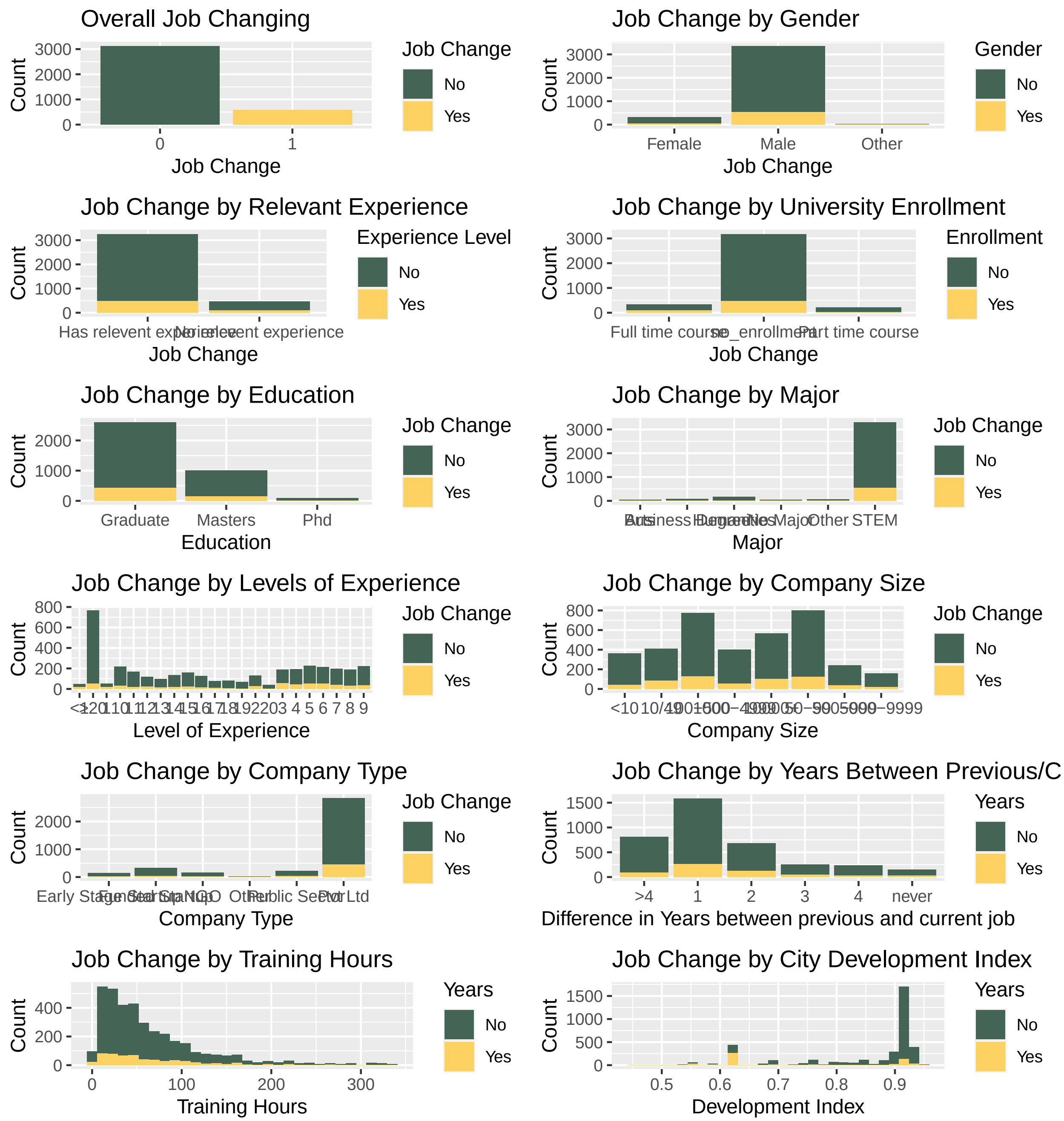
Introduction

Understanding that this is a classification problem given a set of 10 factored and one continuous variable - not considering the candidate ID and city code (I figured city code is included within the continuous City Development Index), a classification tree is likely to be the best approach. In using a classification tree, we determine how "well" the model preforms via the error rate and accuracy (James et al, 311). Additionally, there are some advantages to using trees for clarification purposes like being easily interpretable, similar/mirroring human decision making and easily handle qualitative variables (315). There are certain drawbacks to using trees as they can be very non robust and perform poorly when applied to other data sets which can be handled through boosting among other methods.

The approach I've taken is to use a gradient boosting machine (GBM). Boosting works through growing trees sequentially, meaning that each tree is grown from the previous one (321). This helps in the classification by fitting continuously smaller trees to the previously made residuals. The problem and challenge with using a GBM in this approach is the missing values in this data set. Out of the 8000 training values, there are 4,282 incomplete rows ecompassing about 54% of the data!

Data EDA/Visualization

Unfortunately, it appears that our data is very uneven once we remove NANs rows! Additionally, it appears when grouping by certain variables, we still have a large amount unevenness in job change, with a significantly higher rate of "No" responses. Because of this uneveness-way more people declining jobs compared to accepting them, I would assume more problems in the tree with model specificity when we approach the modeling process.

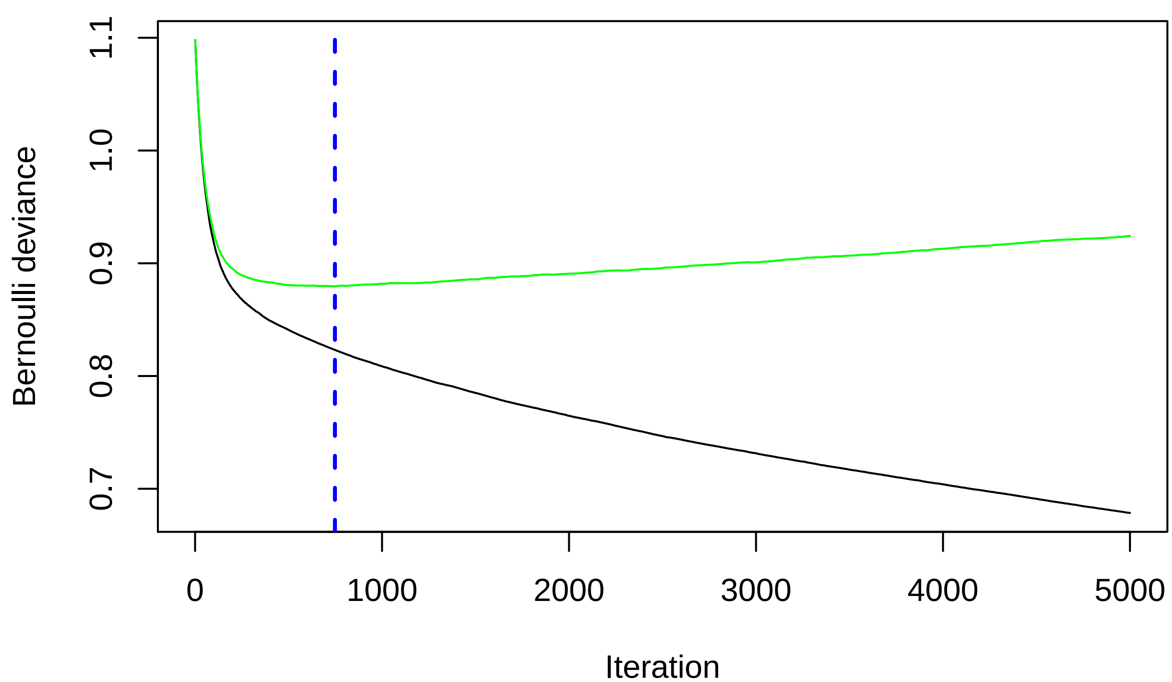


Dealing with NANs

Because of the unevenness in the data and the large amount of missing values, it is likely beneficial to include them somehow. We have several methods for approaching the NA problem; We could "input" missing values simply as the overall mean of the predictor - this would somewhat balance our data towards the mean, create a dummy variable for "missing" and feed it into the model, or create surrogate variables (Hastie et al, 311). I decided to fabricate the variables from the mean of the overall predictor set as a method to balance the observations and allow parsing to library(gbm). The new generated data EDA shows in Appendix A.

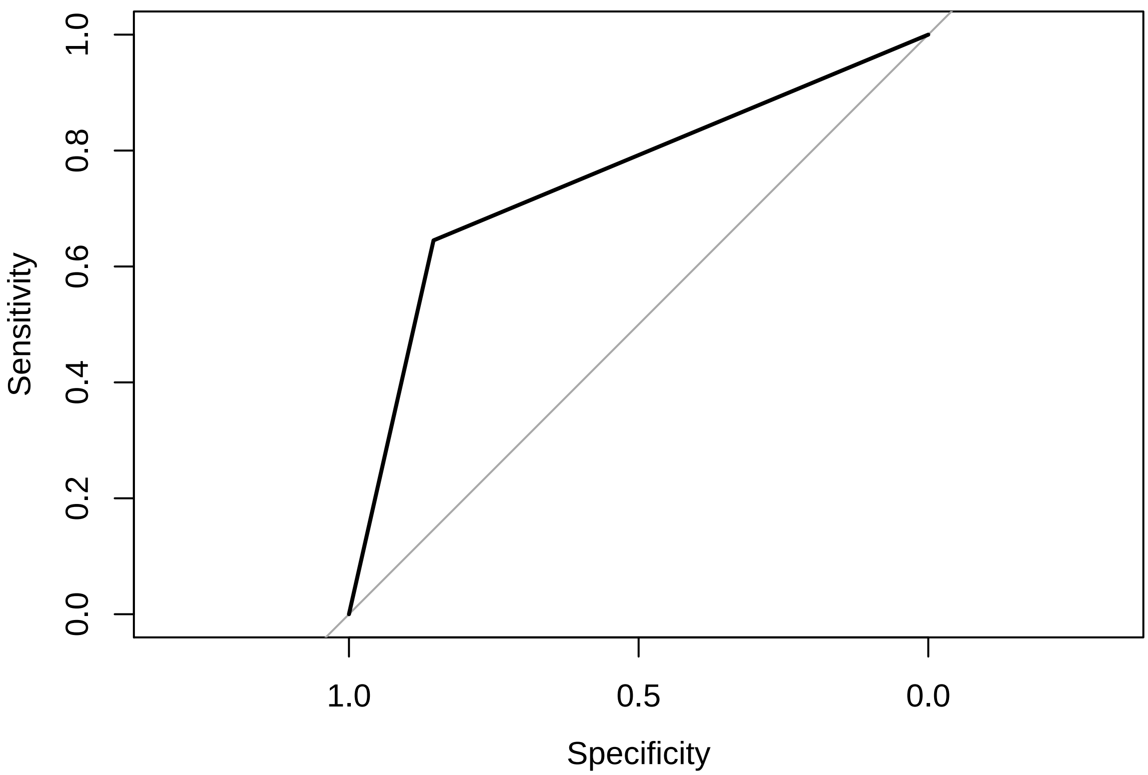
Model Selection

Originally, I was considering filling the NA values with a $N-(\mu, \sigma^2)$ to add some variation and minimize drawing predictors to their mean; however, I turned to just the mean-fill method to avoid adding further noise. Using the GBM Classification Tree as described above, I fit a model of 5000 trees which, as expected, turned out to be too many (Seen Below)! To deal with this and avoid overfitting to the training set, I used gbm::gbm.pref to return the estimated optimal number of iterations using the cv method (748) - found by determining at what iteration count the trees have the lowest MSE compared to the training error; below we can see the error on test (green line) and the error on train (black line) over increasing trees. The relative influence can be found in Appendix B.



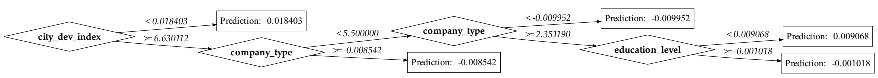
Model Validation / Evaluation

To validate the model, I split the given train into half and trained the model on the first half, then tested it on the second to generated a confusion matrix (Appendix C). Overall the model preformed well, with an accuracy of 80.25% - 95% confidence intervals being (78.98% and 81.47%) and test MSE of 3.92. Furthermore, as expected, the sensitivity preformed well, and the specificity of the model preformed poorly. For further validation, we can look at the ROC curve (Below) and test the AUC (.7495). As a part of evaluation, I also wanted to test if the model would have preformed better if I had dropped the NA values entirely. The NA dropped model (Appendix D) drew roughly the same variable relative influences, but had worse accuracy, specificity and MSE when applied to the held test set.



Discussion

As previously mentioned this model has the shortfall of not being able to intake the 54% incomplete rows in the data and instead intakes the mean value of the variable, which artificially stabilizes the data around the mean. Furthermore, the selected model has a difficult time classifying false positives, as conjecture, this is because of the overall imbalance in the predicted variable. However, the model preformed much better than the NA removed model in overall accuracy and specificity/sensitivity. Special thanks to Justin Singh-M from UC Santa Barbara (https://gist.github.com/program--) for the visualization of tree function!



References

Baptiste Auguie (2017). gridExtra: Miscellaneous Functions for "Grid" Graphics. R package version 2.3. https://CRAN.R-project.org/package=gridExtra

Matt Dowle and Arun Srinivasan (2020). data.table: Extension of `data.frame`. R package version 1.13.0. https://CRAN.R-project.org/package=data.table

Brandon Greenwell, Bradley Boehmke, Jay Cunningham and GBM Developers (2020). gbm: Generalized Boosted Regression Models. R package version 2.1.8. https://CRAN.R-project.org/package=gbm

Trevor Hastie et al., The Elements of Statistical Learning: Data Mining, Inference, and Prediction.

Gareth James et al., An Introduction to Statistical Learning with Applications in R.

Max Kuhn (2020). caret: Classification and Regression Training. R package version 6.0-86. https://CRAN.R-project.org/package=caret

Stefan Milton Bache and Hadley Wickham (2014). magrittr: A Forward-Pipe Operator for R. R package version 1.5. https://CRAN.R-project.org/package=magrittr

Erich Neuwirth (2014). RColorBrewer: ColorBrewer Palettes. R package version 1.1-2. https://CRAN.R-project.org/package=RColorBrewer

Karthik Ram and Hadley Wickham (2018). wesanderson: A Wes Anderson Palette Generator. R package version 0.3.6. https://CRAN.R-project.org/package=wesanderson

Xavier Robin, Natacha Turck, Alexandre Hainard, Natalia Tiberti, Frédérique Lisacek, Jean-

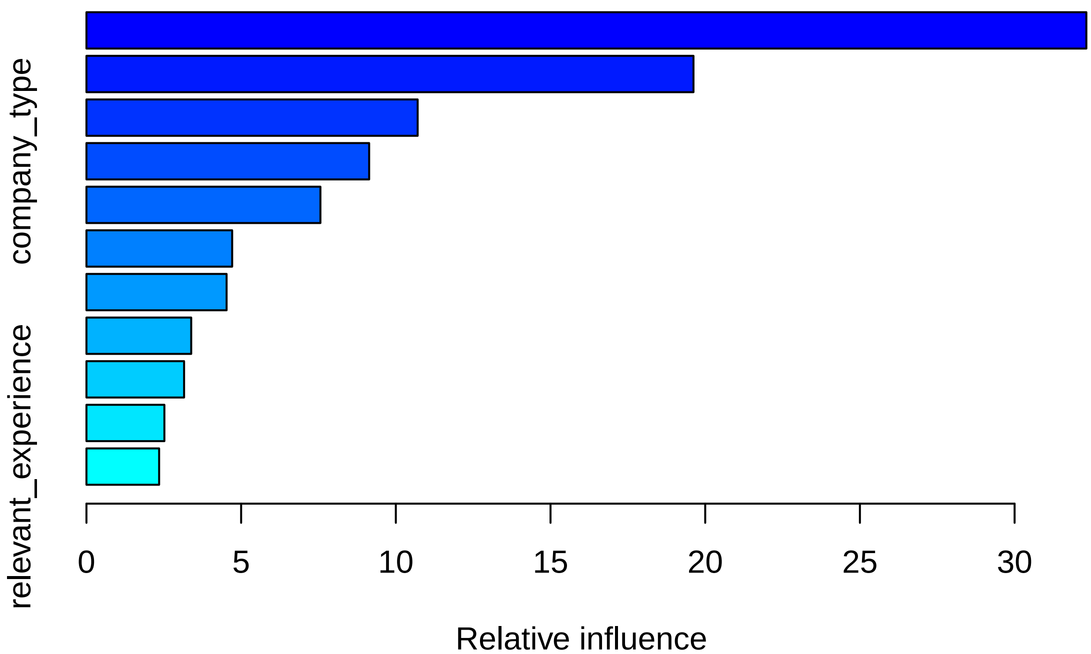
Charles Sanchez and Markus Müller (2011). pROC: an open-source package for R and S+ to analyze and compare ROC curves. BMC Bioinformatics, 12, p. 77. DOI: 10.1186/1471-2105-12-77 http://www.biomedcentral.com/1471-2105/12/77/

Wickham et al., (2019). Welcome to the tidyverse. Journal of Open Source Software, 4(43), 1686, https://doi.org/10.21105/joss.01686

Appendix

A

B



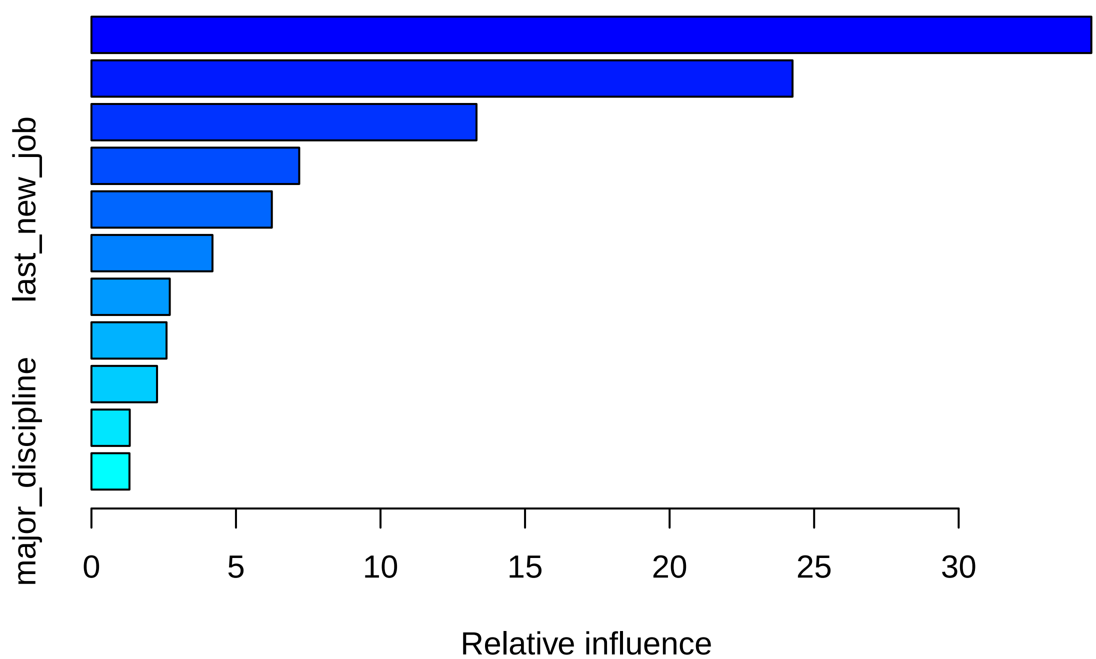

```
##                                       var   rel.inf
## city_dev_index           city_dev_index 32.319667
## training_hours           training_hours 19.620727
## experience_years       experience_years 10.706254
## company_type               company_type  9.137948
## company_size               company_size  7.562379
## education_level         education_level  4.710258
## last_new_job               last_new_job  4.530599
## major_discipline       major_discipline  3.386707
## enrolled_university enrolled_university  3.156699
## gender                           gender  2.519015
## relevant_experience relevant_experience  2.349747
```

C

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2574  350
##          1  440  636
##
##                Accuracy : 0.8025
##                  95% CI : (0.7898, 0.8147)
##     No Information Rate : 0.7535
```

```
##     P-Value [Acc > NIR] : 1.013e-13
##
##                   Kappa : 0.4842
##
##  Mcnemar's Test P-Value : 0.001543
##
##             Sensitivity : 0.8540
##             Specificity : 0.6450
##          Pos Pred Value : 0.8803
##          Neg Pred Value : 0.5911
##              Prevalence : 0.7535
##          Detection Rate : 0.6435
##    Detection Prevalence : 0.7310
##       Balanced Accuracy : 0.7495
##
##        'Positive' Class : 0
##
```

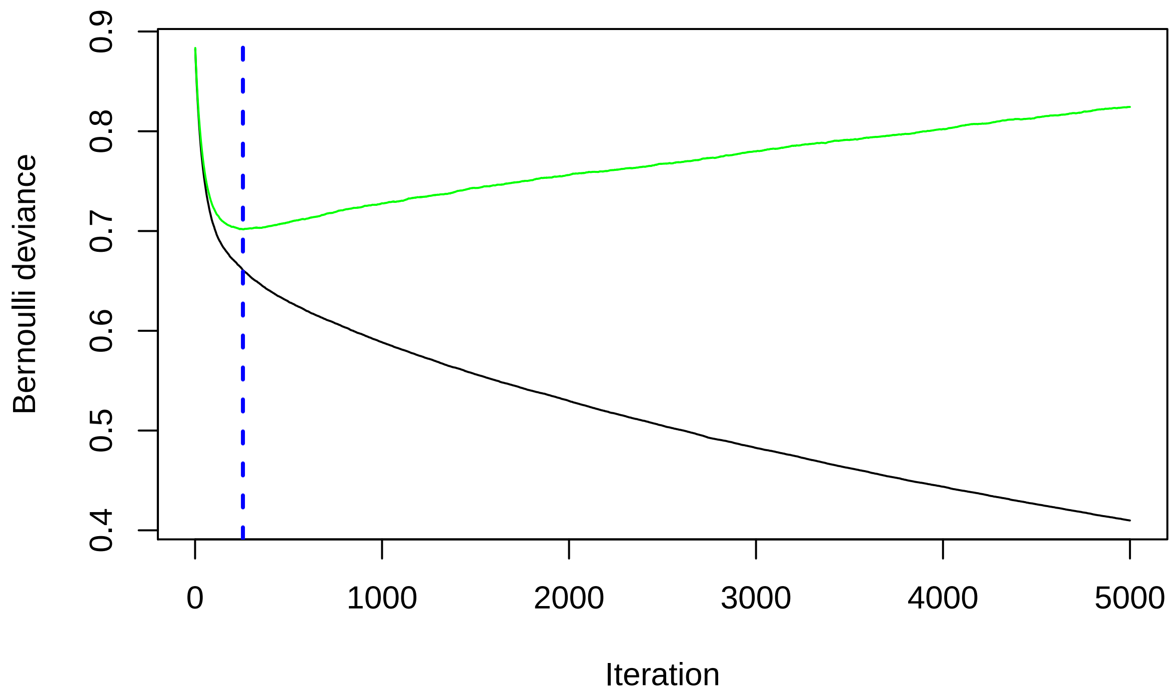
D




```
##                                      var   rel.inf
## city_dev_index           city_dev_index 34.590496
## training_hours           training_hours 24.253311
## experience_years       experience_years 13.320469
## company_size               company_size  7.188066
## last_new_job               last_new_job  6.240360
```

```
## company_type                      company_type  4.187622
## education_level                education_level  2.710540
## enrolled_university enrolled_university  2.598123
## relevant_experience relevant_experience  2.269638
## gender                                  gender  1.325470
## major_discipline              major_discipline  1.315906
```

```
## [1] 8.232002
```




```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1454  131
##          1  116  158
##
##                Accuracy : 0.8671
##                  95% CI : (0.8509, 0.8822)
##     No Information Rate : 0.8445
##     P-Value [Acc > NIR] : 0.003423
##
##                   Kappa : 0.4831
##
##  Mcnemar's Test P-Value : 0.373037
##
##             Sensitivity : 0.9261
##             Specificity : 0.5467
```

```
##          Pos Pred Value : 0.9174
##          Neg Pred Value : 0.5766
##              Prevalence : 0.8445
##          Detection Rate : 0.7821
##    Detection Prevalence : 0.8526
##       Balanced Accuracy : 0.7364
##
##        'Positive' Class : 0
##
```