**Final Lab Report**

**Amazon Employee Access Kaggle Competition**

**Nolan Cochoran (ndc466)**

**Jeremy Shahan (jrs5695)**

Initially, we began playing with the code separately, with the plan to join together at the end to see what we came up with and try combining things. That’s essentially what we ended up doing. While I (Jeremy) documented my findings primarily on this word document, Nolan worked with Jupyter notebook, documenting most of his work along with his code (also in the repository).

I began by fiddling with the data and trying to find any relationships. I first did this by trying train.corr()[“ACTION”] to find any linear correlations between the columns:

ACTION 1.000000

RESOURCE 0.000185

MGR\_ID -0.005167

ROLE\_ROLLUP\_1 -0.013702

ROLE\_ROLLUP\_2 0.005179

ROLE\_DEPTNAME 0.001025

ROLE\_TITLE -0.010169

ROLE\_FAMILY\_DESC 0.003565

ROLE\_FAMILY 0.000502

ROLE\_CODE 0.017147

This told me that there are not any linear correlations of interest, and no column that we may want to individually delve deeper into at this time. Additionally, the nonlinear nature of the data compelled me to stray away from a linear regression model. In the meantime, Nolan tried an SVM classifier and obtained a 0.70 score.

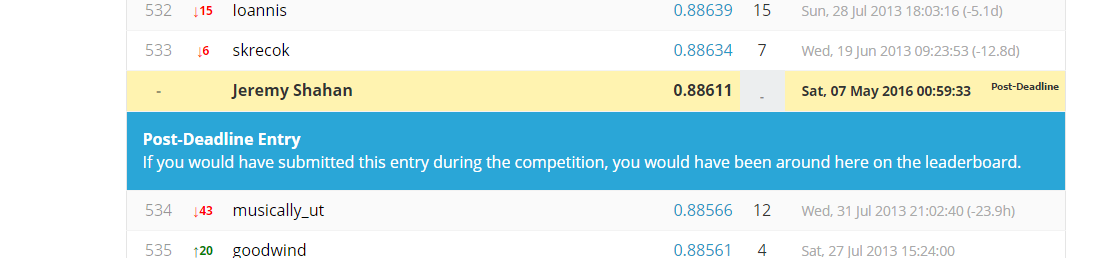
Messing around on the forums I stumbled onto the starter code that we eventually found in class, so I immediately began using this as a basis for my models. Interestingly, the code used a linear regression model and received a very high (almost 0.90) score, debunking my original thought that it wouldn’t be a good model. I realized then that it probably would have been a good idea to one-hot-encode or binarize the data before I searched for a correlation, since the categorical values were not very useful. Still, I wanted to try other models to see how they would score for the dataset. I kept the cross-validation method used and went off to test other classifiers.

I first tried using a LinearSVM, which yielded me a 0.71 private score, landing me somewhere in the lower thousands in ranking. Tweaking the C error value and other parameters, I was not able to achieve a significantly higher score than this. Nolan seemed to reach the same conclusion with his SVM. (file: /src/svm.py, /results/svm.csv)

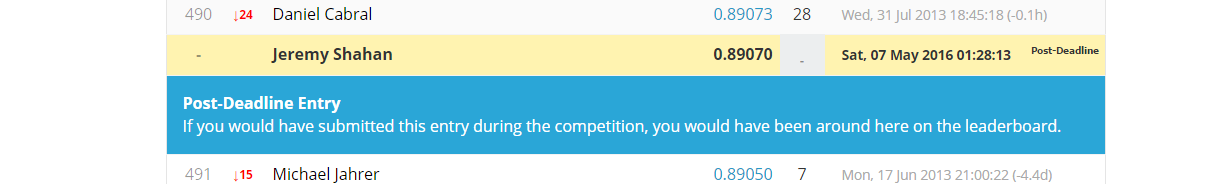
Next up I tried using a K-Nearest Neighbors classifier. I liked that it could give me decimal values for action like the regression model, whereas the SVM was only able to assign binary values. However, not even the small n-values (number of trees) were able to run on my computers prehistoric memory capability. On to the next one. (file: /src/KNC.py)

Since that failed, I tried Random Forest classifier, hoping to adjust for possible overfitting and improve my accuracy. This time, I was able to get a 0.85 private score. (file: /src/RFC.py, /results/RFC.csv)

Using the provided classifier with adjusted parameters, I began manipulating the results from the Random Forest Classifier and the results from the linear logistic regression models to see if I could get a higher score. Using Weighted-Mean AUC, I was able to get up to the low 500s in rank with a private score of 0.88611, giving the classifier a 0.92 weight to the RFC’s 0.08.



I was pretty happy with this score. Since this was effective, I revisited later and tried running the RFC again with as high of a number of estimators as my machine would allow – 20. With a 0.6 – 0.4 weighted average AUC split, I reached the coveted 0.89 mark, and broke the 500 ranking. Woohoo.



Lastly, we decided to try the same methodology with a logistic regression approach. Several people had expressed their preference with a logistic regression so why not try? Wouldn’t you know, the resulting score was pretty dang good, 0.898. Didn’t quite reach the 0.9 mark but we’re well pleased. This code was put into Jupyter Notebook src/finalSubmission.ipynb and the final submission csv file was: results/finalTestSubmission.csv

