1 Introduction [20 points]

Team Members
Matt Riker and Spencer Schneider

Team Name
Jawn Heads

Private Leaderboard Standings
Part 1: 61
Part 2: 40

Private AUC Score
Part 1: 0.78126
Part 2: 0.77718

Division of Labor
Matt: Model Architecture, Cross Validation, Figures, Report
Spencer: Model Architecture, Feature Exploration, Model Research, Report

2 Overview [20 points]

Neural Networks
We tried using both TensorFlow's and sklearn's version of a neural network. We tried different amounts of layers, regularization strengths, activation functions, and hidden nodes.

K Nearest Neighbors
We used sklearn's implementation of K nearest neighbors with various numbers of neighbors and various different distance metrics. We also tried boostrap aggregation.

Linear Models
We tried logistic regression, lasso and ridge linear regression, and support vector machines. All using the sklearn library

Adaboost
We tried using sklearn's version of the Adaboost classifier.

Decision Trees
Several different types of depths of decision trees were tried. Gini impurity was used.

Random Forest
We used sklearn's random forest classifier and varied the number of estimators, max depth, and number of samples required to split a node. Gini impurity was used. The extra random forest classifier was also tried

Ensemble Methods
In addition to bagging, we averaged different models together to get the final predictions.

Timeline

February 8th: Preliminary observations and brain storming
February 9th: First model creations. Experimentation with neural networks and logistic regression.
February 10th: Experimentation with linear regression and random forest models.
February 11th: Parameter Selection, condensing input features, experimentation with SVM, k nearest neighbors, adaboost, and ensemble methods.
February 12th: Final parameter selection and refinement of ensemble methods
February 13th: Report
February 14th: Report

3 Approach [20 points]

Data

At first we trained using just the raw data. Our models performed reasonably well but we figured that manipulating the data could help improve performance and reduce bias. We then decided to cut the input space in half using only the most influential features according to our random forest model because we thought the features with lower weights could be finding noise. We used the sklearn Select From Model implementation to do this. However, this actually led to a decrease in model accuracy. So, we then decided to look at what each column actually represented and realized there was a possibility of data being considered twice. For example, city and state were recorded twice essentially. Thus, we manually went through and noted which features were redundant and got rid of them by parsing the data and removing the desired columns.

Models

Neural Networks

We started trying neural networks because it was the most recently implemented technique in class and they work well with large input spaces. However, despite the different parameters we used the neural nets always predicted all 0's with the relu activation function or all .25's with sigmoid and tanh activation functions. We were never able to resolve this issue so we went on to diferent models.

K Nearest Neighbors

We tried K nearest neighbors because it is generally good at classification problems. Furthermore, conceptually, if the idea is that people with similar responses to the survey will vote similarly then K nearest neighbors makes sense. This classifier trained quickly and is relatively inexpensive to compute, but ultimately none of the k nearest neighbors models outperformed even simple linear models.

Linear Models

We used all sorts of linear models including lasso regression, ridge regression, logistic regression and SVM. The logic behind this was that there should be some very important features that create a semi linear decision boundary as well as unimportant features which can be thrown out by SVM and lasso regression. Logistic regression simply output .25 as the probability someone would vote for all voters. SVM is far too computationally expensive and never finished running. Lasso regression performed very well and was our second best performing model after random forests. We used cross validation to choose the best regularization strength. Lasso regression outperformed ridge regression probably due to the fact that Lasso regression sends unimportant feature weights all the way to 0 while ridge regression weights are always non zero.

Adaboost

Adaboost is good in binary classification problems so we thought we could use it in this case. However, it isn't good at outputting probabilities for binary classification problems so the result was all 1's and 0's which wasnt useful in this context so we went away from adaboost.

Decision Trees

Decision trees are good at classification and we tried trees of different depths and of different minimum samples to split an internal node. In the end it was too difficult to balance under and over fitting so our model topped out at an accuracy of about .7.

Random Forest

Random forests are very good for classification problems which is why we tried them. This ended up being our best performing non ensemble model. We used cross validation to select parameters and ending up getting a model that was accurate and that converged quickly.

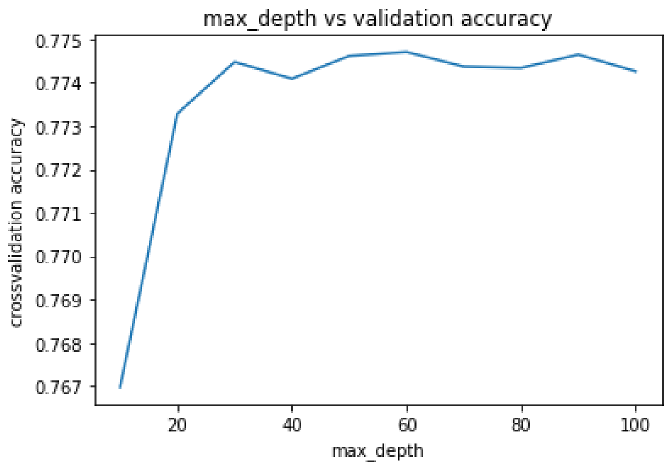
Ensemble Methods

Ensemble methods are probably the most popular for complex machine learning problems. The Netflix competition and many kaggle competitions are won by ensemble models. We tried averaging the probabilities of all our our relatively successful models but ultimately the model with just random forest and lasso regression preformed the best. This model was a little more computationally expensive, but its significant increase in accuracy made it worth it. This was our best performing model.

4 Model Selection [20 points]

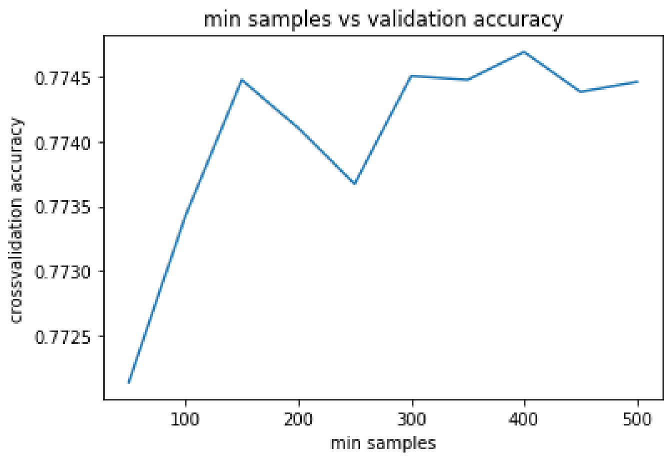
The scoring we used for our models was testing accuracy through cross validation. We used accuracy because loss isn't really as relevant in our classification problem. The model that scored the highest was when we took the average from the lasso linear regression and the random forest. For the Lasso linear regression, we used an alpha of 0.0001 and maximum number of iterations as 10000. See details below for the random forest parameters.

We used 5fold cross validation to test our models and find the optimal parameters to use for our random forest. We first wanted to find the maximum depth to use so we ran 5fold cross validation using depths from 10 to 100, with number of estimators at 250 and min samples at 80 and got the following results:



We concluded the optimal depth to use 55. We also saw that for depths greater than 250 the cross validation accuracy continued to increase, higher than it was at 55, but this was more prone to overfitting so we used 55.

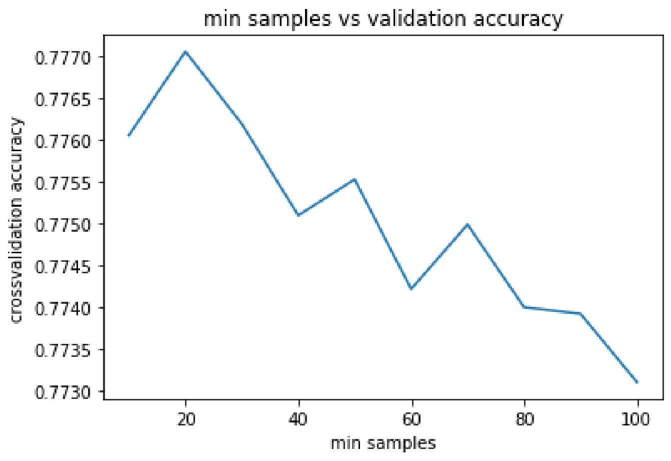
We then ran 5fold cross validation on the number of estimators from 50 to 500 using a maximum depth of 55 and 80 as our number of samples which produced the following:



Note: the x label should be number of estimators

We concluded the optimal number of estimators to use was 150 a it produced the highest cross validation accuracy other than very large values that could have produced over fitting and require much more time and computational power for little, if any, gain.

Lastly, to optimize the minimum number of samples we ran 5fold cross validation using 150 as out number of estimators and 55 as our maximum depth. We produced the following:



This allowed us to conclude the optimal minimum number of samples to use given the other two parameters was 20.

5 Conclusion [20 points]

The features that have the most influence on the target for the random forest classifier are: household characteristics, a non interview reason, household identifier, household identifier, filtered for retired, looked for work in the last 12 months, household identifier, household identifier, check item 4, and check item 3 in increasing order.
The features that have the most influence on the target for the Lasso linear regression are: interview status, household identifier, a non interview reason, household identifier, on layoff last week, household type, did you have a job last week, filter for passive job seekers, 35+ hour work week, and worked full time last week if hours were available, in increasing order.

One thing we could have done better with the features is to use cross validation to check which columns add a lot of noise, and we could use this information to not account for those features heavily in our model, or just not consider them at all. I think a reason why our model didn't perform as well as others is because of this, and the model was taking into consideration some features that were too noisy and not giving us accurate predictions.
The AUC appears to be the more appropriate metric to use here because this is a good metric to use for skewed data and for binary classifications, which have here. This matches closest to the data that we are dealing with here, so it does not appear there is a better metric to use for this project. Furthermore, the AUC predictor can range from 0 to 1, which is also what we were dealing with in our predictions.

Parallelism was not present in our project, but through our random forest classifier it could have been. For example, in our random forest classifier, one of the possible inputs is number of jobs. We did not specify this input with our parameters, so it was assumed to be 1. However, if we made this number greater than 1, we would have had multiple jobs run at once in parallel for both the fit and predict that we used for this classifier. The number for that input represents the number of concurrently running jobs, and would have lead to parallelism.

One thing we learned from this project that we didn't experience in the homework was a dying relu in action. Our first model attempt was a neural net using a relu, and when we did this all of the projections were 0. Overall, it was a great learning experience in trying out the different models and seeing results we get based on the models we use. With homeworks and lectures, we are often catered to use a specific model for a specific problem. Here, we were giving no sense of direction on where to begin and made us see models that do and do not work for this specific problem. Another thing we learned was how much noise can affect the model, and the weights the model produced. We learned how there are often times when we need to do particular operations to manage this noise.

The main challenge that we faced was dealing with a real life dataset that is imbalanced and noisy. Getting various types of models to work on a dataset of this type is a challenge without overfitting. Using cross validation to choose parameters was also a challenge on laptops as these models take a while to run and can cause a laptop to crash if they require too much RAM. For example, holding two of three parameters constant while testing 5fold cross validation while varying the third parameter the random forest took a lot of time to compute. This process had to be repeated for each one of the parameters. Overall, it was a big jump from training models in sets and doing it in real life with fewer contrived learning situations.