COMS W4721
Machine Learning for Data Science
Team DAMAGE

Daniel Nachajon (dn2387)
Manu Singh (ms5129)
Gerardo Sierra (gs2842)

May 4, 2016

1 Outline of approach

To build the best possible predictive model, we perform several different 10-fold cross validations to evaluate the standalone merits of various binary classifiers. It became apparent rather quickly that tree-based methods did particularly well and so Team DAMAGE chose to employ a voting scheme over choice permutations of these methods. Our final classifier was a majority vote over an AdaBoosted Decision Tree and two different random forests. Each approach was trained over a different processing of the data.

The approach we followed can be summarized in the following steps:

- Feature space transformation parameters definition
- Model selection
- Tuning model parameters
- Voting

All our analyses were completed using the permitted Python packages and libraries.

2 Feature space transformation parameters

In the process of exploring and manipulating the training data, we identified two key issues which might complicate the training of a classifier. These issues were:

- **Low variability features**.- A few Features exhibited no change in value whatsoever. Other Features offered such small variance that it is unclear whether they add any relevant or extractable information to a predictive model.
- **Categorical features with high cardinality**.- Python cannot readily handle categorical variables, which must instead be expanded into a dummy-variabel representation. Certain categorical Features had distinct values in the 1000s and this would result in a considerable expansion of the feature space and therefore impact our running times.

To address these issues, we defined two transformations of the data that are functions of parameter thresholds: *homogeneity* and *cardinality*.

2.1 Low variability features .- *Homogeneity*

The first step to detect variability was to measure the *homogeneity*. For each feature column, we measured the highest percent of total observations that has the same value. We called this percentage the *homogeneity* of each feature. Features with 100% *homogeneity* were automatically excluded.

Once the values with no variability were removed, we kept the *homogeneity* value for each parameter as a parameter for future use and would remove certain features if their *homogeneity* was too close to 100%.

2.2 Categorical features high cardinality.- *Cardinality*

Categorical feature columns with large (*cardinality*) of values, might be problematic for two reasons. The first reason is computational complexity. The number of potential permutations between large number of categories grows rapidly, and models can become very complicated. This in turn could affect feature importance selection. The second reason we considered was that some categorical values will show up only a few times, and this could lead the model to have some unwanted variance in fitting these observations. To address this issue, we decided to convert the categorical values that are above a chosen *Cardinality* threshold to a meaningful numerical value.

In order to convert the values of a categorical feature X to meaningful numerical values $\pi(X)$, we decided to replace each value $X = x$ by an estimate $\pi(X = x)$ of the *prior* conditional probability of having $label = 1$ given that particular value. In other words, $\hat{Pr}(Y = 1|X = x)$

To estimate this number for each categorical value, we computed the relative number of observations in which each categorical value had a $label = 1$ using the following formula:

$$\pi(x) = \hat{Pr}(Y = 1|X = x) = \frac{\sum_{i=1}^{n} \mathbb{1}(x_i = x, y_i = 1)}{\sum_{i=1}^{n} \mathbb{1}(x_i = x)}$$

We then replaced each categoric value by its associated value $\pi(x)$, and repeated this for every column in which there were more than a certain *cardinality* threshold of different categorical values. We keep this threshold as a parameter for future use.

It is important to note that this procedure generates missing values when dealing with "new" values in the testing dataset (i.e., the denominator of our estimate is 0. For this cases (around 300 in total), we set $\pi(x)$ equal to the proportion of cases in the whole training dataset with $label = 1$. An avenue for improving our algorithm would be figuring better ways to impute such missing values, such as nearest neighbours to predict their values with "known" values from the training set. Another benefit of this transormation is that it let us use high variance features which later turned out to be quite predictive, whereas otherwise we might have prematurely discarded them.

2.3 Other features

Numerical and binary features were not transformed. Categorical features with sufficiently low homogeneity and low cardinality were transformed into dummy variables

3 Models selection

To select which models to include on our voting, we decided to do 10-fold Cross Validation using different techniques listed below:

We tested the following models in no particular order:

- Random forests
- Adaboost (Weak Learner: decision tree)
- Decision trees
- Nearest neighbours
- Ridge regression
- Logistic Regression
- LDA / QDA
- Averaged Perceptron

The feature transformation parameters we used to select the model were:

- Feature transformation parameters set 1
  - Homogeneity threshold = 98
  - Cardinality threshold = 4

Meaning that we excluded any feature with over 98% homogeneity, and any categorical feature with over 4 different possible values. We decided to use these parameters so that the feature space was relatively small to perform CV. We had initially hoped to run an SVM but that proved far too time-intensive to really compare. We also attempted a degree-2 polynomial expansion of the feature space but this did not offer better CV accuracy rates. There was a certain sequential nature to model selection process. For instance, we first discovered that a simple Decision tree was a good model and then later quanitified the benefits of boosting and bootstrapping.

Our initial explorations revealed that LDA, QDA, Ridge Regression and Perceptron seemed doomed to sub-optimal performance. This is unsurprising as some of these impose too strong of a structural assumption not justified by the data and others required reasonably (soft) linear separation which also was not the case. These findings suggested that perhaps the ability of logistic regression to work with conditional probabilities and non-parametric methods' power to settle on non-linear boundaries might do well. Indeed, the best 10-fold CV accuracy rates by model family were:

- Decision Tree of depth 25 - 92.23%
- First Nearest Neighbour - 90.91%

- Logistic Regression - 88.25%

Unfortunately, kNN cannot be readily boosted in Python and when we tried to boost Logistic Regression the results were not hopeful. We therefore decided to further our model selection with tree-based approaches. From this process, we found out that **Random Forest** and **Adaboost** were the best single model predictors, and in-truth part of the power of our voting scheme reflects and a tuning of the parameters for these algorithms.

4 Tuning model parameters

Once we determined we would use Adaboost and Random Forests, we proceeded to tune the models' parameters

4.1 Feature transformation parameters set

While we used the small (Cardinality = 4, Homogeneity = 98%) feature space to select our models, we decided to use a larger feature space transformation to capture as much information from the dataset as possible. Thus, the other set of feature transformation parameters we used was:

- Feature transformation parameters set 2
  - Homogeneity = 100%
  - Cardinality = 1000

It is important to note that Adaboost ran for too long to fit feature space 2, which was just reserved for our Random Forest voters.

4.2 Feature prioritization

To reduce computation speed without losing predictive power, we applied a Recursive Feature Elimination algorithm (RFE) to select the most important variables in our "small" feature spaces (Cardinality = 4, Homogeneity = 98). We did not do a feature prioritization for the "large" feature space 2 described above (Cardinality = 1000, Homogeneity = 100) because we wanted to capture information from as many features as possible.

We selected the number of prioritized features using Cross Validation. For Adaboost, we selected 25 variables, and for Random Forest we selected 30.

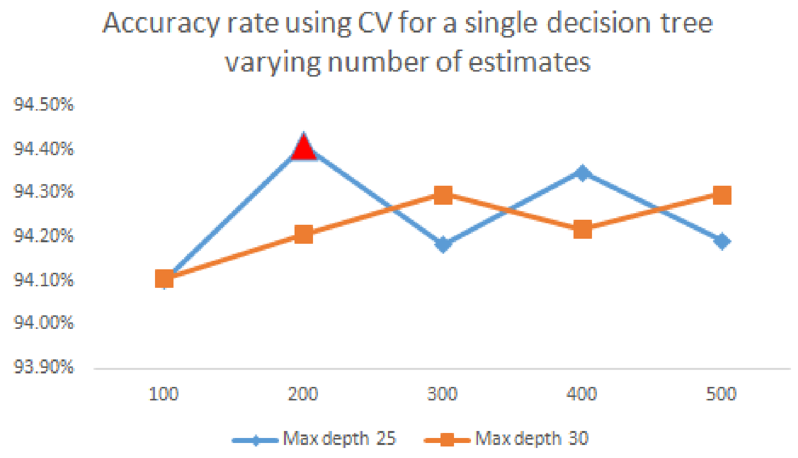
This procedure was especially useful for Adaboost; dramatically improving run time once we reduced the feature space to under 30 variables.

4.3 Individual model parameters

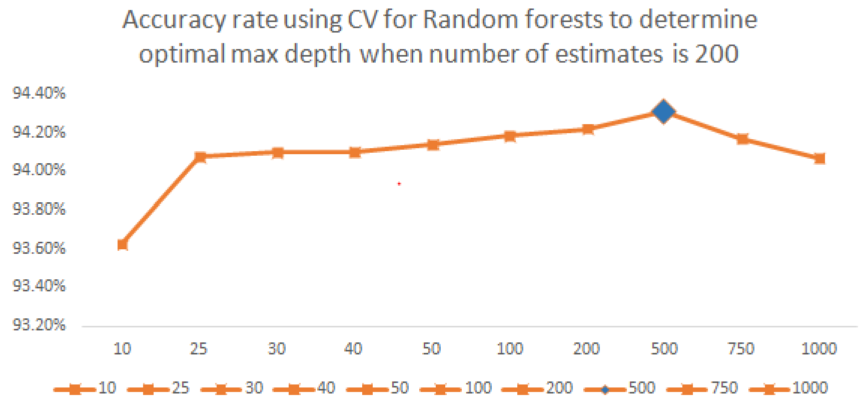
The last step in this section was to define the individual model parameters. For this, we used Cross Validations to determine a good combination of parameters.

The most efficient way of determining this was looking at the variation in model accuracy with different parameters.

In the case of random forest, we tuned two parameters: number of estimates, and max depth of tree. In the following figure, the cross-validated test accuracies (y axis) were plotted against increasing number of estimates (x axis) at a constant depth (two cases illustrated here show depth of 25 and 30). The plot of the accuracies shows the optimum number of estimates is 200.



Now, holding the number of estimates fixed at 200, we selected the best value of max depth. In the figure ahead, the cross validated accuracies are plotted against increasing tree depth. The plot of accuracies shows maximum efficiency being achieved at depth 500 for a tree with 200 estimates.



Ideally such a selection procedure should have been iterated a few times over until no significant increase in efficiency is observable (ie finding the optimal number of estimates holding max depth fixed at 500, and so on). But keeping in mind the short time frame, we decided to keep the above parameters.

4.4 Final models

In sum, the final 3 models we used for our voting were:

Voter 1: Adaboost (25 variables)

- Feature space
  - Homogeneity = 98%
  - Cardinality = 4
  - RFE feature selection = 25
- Model parameters
  - 200 estimates
  - Max depth (WL) = 500

Voter 2: "Small" Random forest (30 variables)

- Feature space
  - Homogeneity = 98%
  - Cardinality = 4
  - RFE feature selection = 30
- Model parameters
  - 500 trees
  - Max depth = None

Voter 3: "Large" Random forest (700+ variables)

- Feature space
  - Homogeneity = 100%
  - Cardinality = 1000
  - RFE feature selection = None
- Model parameters
  - 500 trees
  - Max depth = 200

5 Voting

Once we generated the predictions with our three final models, we proceeded to vote.

Voting consists simply in choosing as our optimal prediction, the label that was chosen by at least two of our three final models.

In other words:

$$Prediction = sign(Voter_1 + Voter_2 + Voter_3)$$

6 Potential further improvements

Current methodology could yield even better predictions by (for example):

- Improving the missing data imputation process (see Section 2.2)
- Further fine tuning feature space
- Further fine tuning model selection