# Data Mining: Final Project

*Andrew Resnikoff*

*aresniko*

## Introduction

### Background

Many popular websites use advertisements to generate revenue for the site. Some users would prefer to use these sites without having to look at these advertisements, so they employ an "ad-blocker" to remove these advertisements from the page. Through this analysis, we aim to build a model that can successfully determine which images on a page are ads. Using this model, we can predict whether or not an image is an advertisement (and an actual ad blocking service could then actually block the image on the site.)

### Data

To build this model, we are given a training data set of 1359 images. Each of these observations has values for variables such as the image width, height, name and, most importantly, whether or not it is an advertisement.

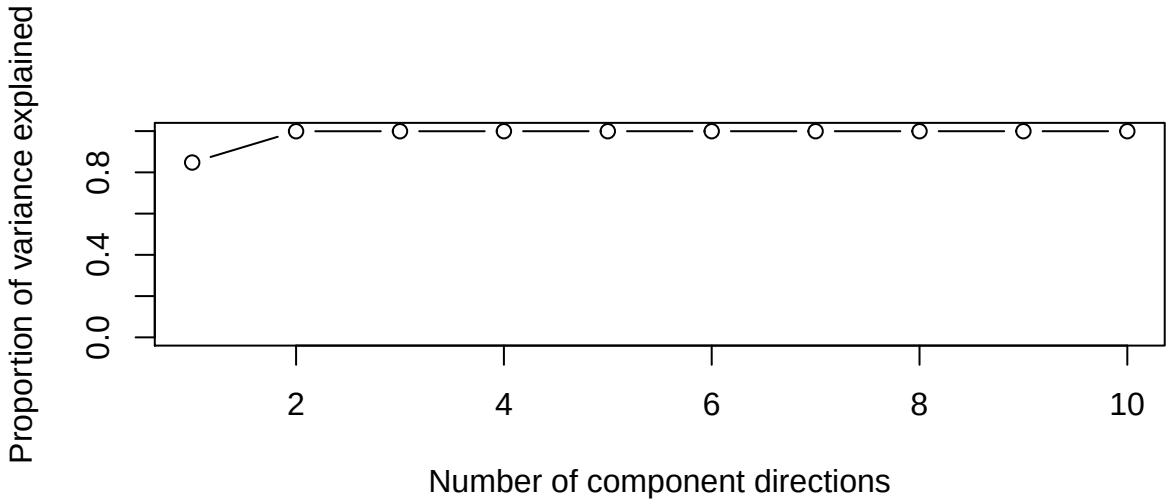
## Unsupervised Analysis

We begin our investigation by performing unsupervised analysis. This analysis will give us a better understanding of the underlying structure of the data.

### Choosing $k$

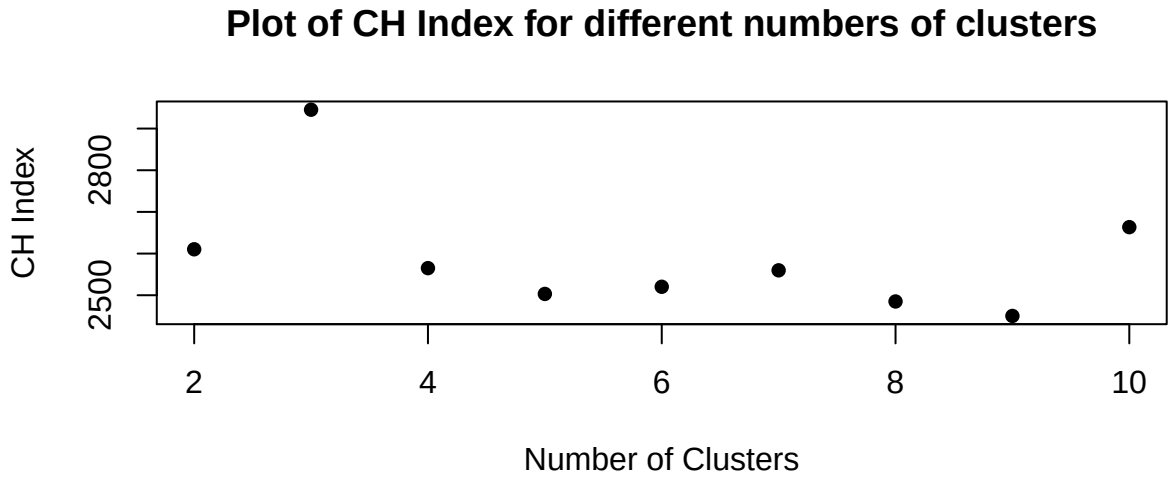
The first step in unsupervised analysis is to choose a value for $k$, the number of groupings of the data we would like to see.

We can use Principal Components Analysis to view the proportion of variance explained in the image data for varied $k$, the number of component directions.



Notice that essentially all the variance is explained for the image data by two factors. This suggests that $k = 2$.

To confirm this value of $k$, we can look at the CH index.




This confirms our choice of $k = 2$. It seems reasonable to assume that these two clusters represent ads and non ads, as opposed to two other random factors.

### k-means

Now that we have a set number of groups to divide our data, we can try splitting it into these groups.

Using the k-means clustering algorithm, we can divide the training data into two clusters.

```
km<-kmeans(x = X,centers = 2,iter.max = 20,nstart = 2,algorithm = "Lloyd")
```

After running k-means and separating the data into two clusters, we are left with a classification rate (on the test data) of 11%. In this clustering, 1195 data points were labeled as non ads an 164 data points were labeled as ads.

We can use the centers of these two clusters to classify future data.

### Hierarchical Agglomerative Clustering

In addition to running the k-means algorithm, we can also perform hierarchical agglomerative clustering on the data set.

```
d = dist(X)
tree.s<-hclust(d,method="single")
tree.c<-hclust(d,method="complete")
tree.avg<-hclust(d,method="average")
```

Printed below are the classification rates for $k$-means and then hierarchical clusters using single, complete and average linkage respectively, for $k$ in the range 2 to 11. For $k \neq 2$, every single combination of clustering the $k$ clusters into 2 clusters was considered, and the best classification error (and cluster) was used.

Table 1: Misclassification Rates for Unsupervised Methods

| | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| k-means | 11.0 | 10.4 | 10.4 | 10.1 | 10.2 | 10.1 | 9.2 | 10.0 | 10.1 | 10.1 |
| single linkage | 16.3 | 16.2 | 16.2 | 16.2 | 16.2 | 16.2 | 16.2 | 16.2 | 16.2 | 16.2 |
| complete linkage | 16.8 | 10.4 | 10.4 | 10.2 | 10.2 | 9.4 | 9.4 | 9.4 | 9.5 | 9.5 |
| average linkage | 16.3 | 10.2 | 10.2 | 10.2 | 10.1 | 10.1 | 9.3 | 9.3 | 9.3 | 9.3 |

Notice that we get the best rate (even better than k-means clustering) for hierarchical clustering with average linkage and $k = 8$ or 9 or 10.

It is possible that within the two groups (ad and non-ad), there are 8 (or 9 or 10) subgroups that can be used to even better identify which images are ads. Because we do not have any more information about ads (like whether there are 4 different types of ads and 4 different types of regular images), it seems unwise to try to incorporate this finding into our model. It is possible this is a structure that exists in our training set but not in the set of all images.

### Summary

Ultimately, the most important takeaway from our unsupervised analysis is that there are 2 main subgroups of our training data (hopefully, these groups are ads and non ads.) While we may later try to use some of the clustering from this analysis, it is more likely that supervised analysis will be much better at predicting whether an image is an ad.

## Supervised Analysis

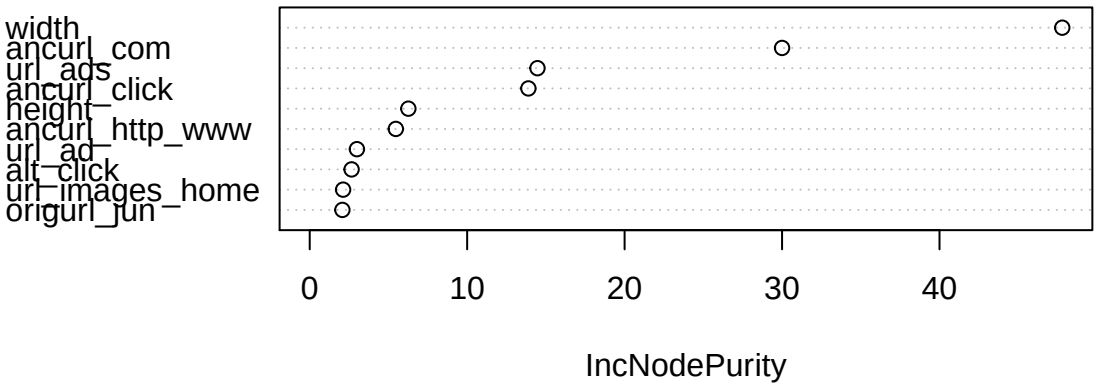
Supervised analysis allows us to incorporate the labels (ad and non ad) from our training set and build models that can predict whether or not an image is an ad. We will use cross validation for all of our models to get the

### Variable Importance

We start our supervised analysis by determining which predictor variables are the most important in predicting whether an image is an ad.

To find these predictor variables, we create a random forest using the training data

```
forest<-randomForest(ad~.,data=images.train)
varImpPlot(forest,n.var=10)
```




We can see the 10 most important variables in this plot. The names of these variables are listed below in order of importance. Based on the variable importance as given by the random forest the variables

```
width, ancurl_com, url_ads, ancurl_click, height, ancurl_http_www, url_ad, alt_click,
url_images_home, origurl_jun
```

are the most important. We can use these variables to build models that do not depend on all of the variables. This could help us prevent over-fitting.

From now on, we can refer to models built only on these predictor variables as the limited model, and models built on all of the predictor variables as the full model.

### $k$-nearest neighbors

Our first predictive model will use the $k$-nearest neighbors algorithm.

We can run this algorithm on our two different models and classify observations by the

Table 2: Misclassification Errors for varied k (KNN)

| | k = 1 | k = 2 | k = 3 | k = 4 | k = 5 | k = 6 | k = 7 | k = 8 | k = 9 | k = 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Limited model | 0.057 | 0.076 | 0.077 | 0.088 | 0.087 | 0.084 | 0.084 | 0.091 | 0.084 | 0.082 |
| Full model | 0.052 | 0.068 | 0.071 | 0.077 | 0.078 | 0.090 | 0.084 | 0.085 | 0.086 | 0.082 |

Since 1-nearest neighbors has the best cross validated error (0.0566611% for the limited model and 0.0522439% for the full model), we will use this in our final prediction.

### Logistic Regression

Using logistic regression, we can predict probabilities for images to be ads. We can then classify observations as ads or not based on how large these predicted probabilities are. We build Below is a table of the misclassification rates a logistic model over varied classification boundaries.

Table 3: Misclassification Errors for varied classification boundary (logistic regression)

| | p > .5 | p > .6 | p > .7 | p > .8 | p > .9 |
|---|---|---|---|---|---|
| misclassification error | 0.058 | 0.057 | 0.053 | 0.052 | 0.059 |

From the table, we see that using a probability boundary of .8, we get the best misclassification error (0.0522276%). While this implies that we should classify something as ad if we are at least 80% sure that it is an ad, we must be careful of over-fitting. It is possible that having to be 80% sure (instead of 50%) works on the training data because most of the observations in the training data are not ads.

We also can try building different models using the limited and full data and allowing the predict function to classify each observation. In these models, we can use the value of $\lambda$ one standard deviation over from the cross validated $\lambda$.

The misclassification rate for the full model when we predict class instead of probabilities is 0.0264901%.

The misclassification rate for the limited model when we predict class instead of probabilities is 0.0676968%.

### Boosting

Our final predictive models will be generated using boosting. We can create boosted models for the full and limited data set and then predict using each of them.

The misclassification rate for the full model when we use boosting is 0.0073584%.

The misclassification rate for the limited model when we use boosting is 0.0161884%.

## Predictions

Now that we have finished building our models, it is time to make predictions from them.

Based on the cross validated misclassification rates, the best three models were the full boosting model, the limited boosting model and full logistic model from best to worst. Since these three models have much lower misclassification rates than most of the other models, will use these three models to make predictions for our test set.

We can use these three sets of predictions to predict whether or not each observation is an ad or not. We will do this by comparing the predictions between the three models. The algorithm used is explained in the commented function below.

```
# ARGUMENTS: model1, model2, model3, The best model, second best model and third best model
# RETURNS: a binary vector of predictions indicating which observations are ads
predict.ad<-function(model1,model2,model3){
  ad<-rep(NA,length(model1))
  for (i in 1:length(model1)){
    # all models agree
    if (model1[i] == model2[i] & model2[i] == model3[i] ){
      ad[i] = model1[i]
    }
    # model1 and model2 agree
    else if (model1[i] == model2[i]){
      ad[i] = model1[i]
    }
```

```
    # model1 and model3 agree
    else if (model1[i] == model3[i]){
      ad[i] = model1[i]
    }
    # model 2 and model 3 agree
    else if (model2[i] == model3[i]){
      ad[i] = model2[i]
    }
    # Default case: return best model prediction
    else{
      ad[i] = model1[i]
    }
  }
  return(ad)
}
```

Even though the limited boosting model had a better misclassification rate than the full logistic model, since the misclassification rates were close it seems like a good idea to use the logistic model as our second best model. This might help stabilize our predictions because it is possible there is an error in both boosting models.

The first 70 predictions of our final prediction for the 1000 images in the test set are printed below.

```
##  [1] 0 1 0 1 0 1 0 1 0 1 0 0 0 0 1 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [36] 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0
```

We can compare these predictions to the actual classification of each observation.

```
# correctly predicted
sum(ifelse(ad == test.ans,1,0))
```

```
## [1] 974
```

We correctly predicted 974 of the 1000 images observed.

The misclassification rate of our prediction rule is thus .026%.

```
# correctly predicted for limited boosting model
sum(ifelse(cluster.boost == test.ans,1,0))
```

```
## [1] 976
```

```
# correctly predicted for full boosting model
sum(ifelse(cluster.boost.full == test.ans,1,0))
```

```
## [1] 975
```

```
# correctly predicted for full logistic model
sum(ifelse(cluster.glmnet.full == test.ans,1,0))
```

```
## [1] 964
```

Looking at the number of images correctly predicted by the three models we used alone, notice that we did better than the logistic regression but not better than the full or limited boosting model, although it was very close. A .026% misclassification rate demonstrates that our prediction rule was a success.