GENERAL ASSEMBLY

DSI

Feature Selection and Predicting the Madelon Dataset

Ed Prentice

November 13, 2017

Abstract

For this project we worked with the Madelon dataset. Madelon is an artificial dataset with many noisy features. Our main goal was to significantly reduce the number of features while maintaining a high accuracy score. In order to reduce non-contributing features, "leave one out regression" effectively identifies features within data set with the most impact. By removing a single category and cross referencing the R-squared value, we are able to identify the largest contributing features. We then removed more features based on high correlations. For the UCI dataset, we were able to reduce down to 8 features with an accuracy score of 89.6%. For the instructors dataset, we were able to reduce down to 11 features with an accuracy score of 85.63%. Ideally this method would be effective on any data set composed of a large number of features of unknown importance, helping to eliminate bias in feature reduction.

Introduction

Madelon is an artificial dataset created as a challenge for feature selection. It is a two-class classification problem with highly non-linear data. Our main goal with this project is to demonstrate our ability to identify relevant features when there are hundreds (or thousands) features and to classify with a relatively high degree of accuracy. There are several challenges that we expect along the way. One is that we are forcing ourselves to perform our analysis on a AWS t2.Micro (1 GB RAM). We do this to simulate engineering problems we could run into with "big" data. Also there is no context involved with the features, so we cannot make any intuitive conclusions while we move through our analysis.

We will be working with two different Madelon datasets. One from the University of California Irvine (UCI), and the other is from a database created by our instructors that is much larger. We will analyze the UCI dataset first (since it is significantly smaller) and develop a strategy that would scale well to the instructor's larger dataset.

Description of UCI Data

The UCI dataset has 500 different features and 4000 observations (we combined the training and validation datasets). There are 20 "true" features (5 informative and the other 15 are linear combinations of the "real" 5). The target values are either -1 or 1. From the 4000 observations we will get a representative sample to make it easier on our t2.micro. We would like a 95% confidence interval ($z = 1.96$) with a margin of error of 2%

$$n = \left(\frac{Z_{\alpha/2} \cdot \sigma}{\text{margin of error}} \right)^2$$

$$\Rightarrow n = 1500$$

We will use a sample of 1500 observations.

Exploratory Analysis (UCI)

There is not a whole lot of EDA we can do (especially when there is no context in the features). However, we we still want some idea of the data we are working with. We find that there are no missing values and everything is the correct data type. Below is a histogram of 12 random features.

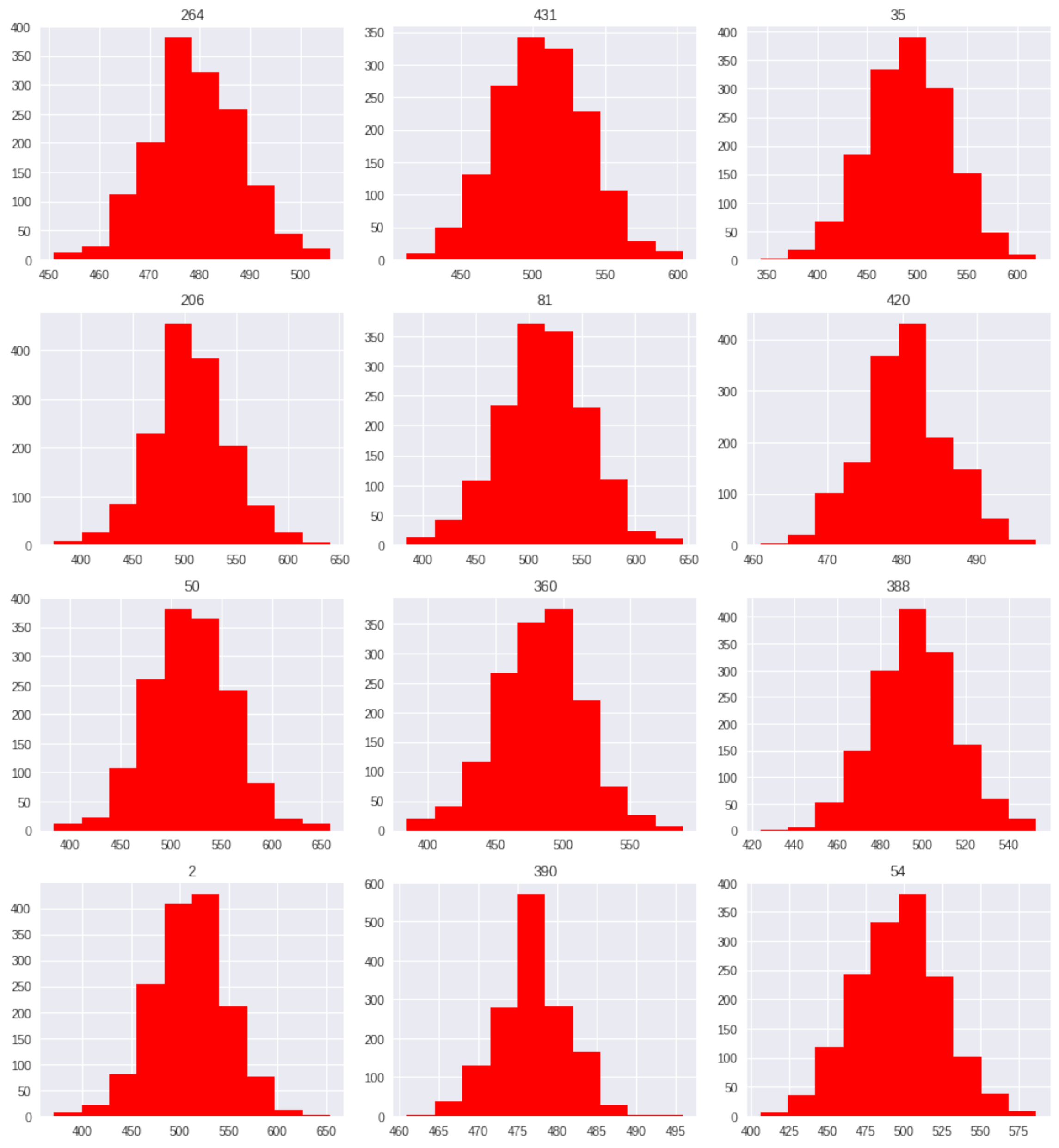

Figure 1: Histogram of 12 random features

All of the features are continuous (numerical) and appear to be fairly normal. We will explore this more after a significant feature reduction.

Reducing Dimmensions (UCI)

Our first task is to significantly reduce our features. We will look at two different methods for this. Since they are numerical we will first look at the ANOVA F-value of the features (vs χ^2 for categorical variables). The ANOVA F-value is a supervised learning method where the features are tested against the target variable and a corresponding p-value is calculated (the smaller the p-value the more significant the feature). Below is a bar plot of the 23 most important features.

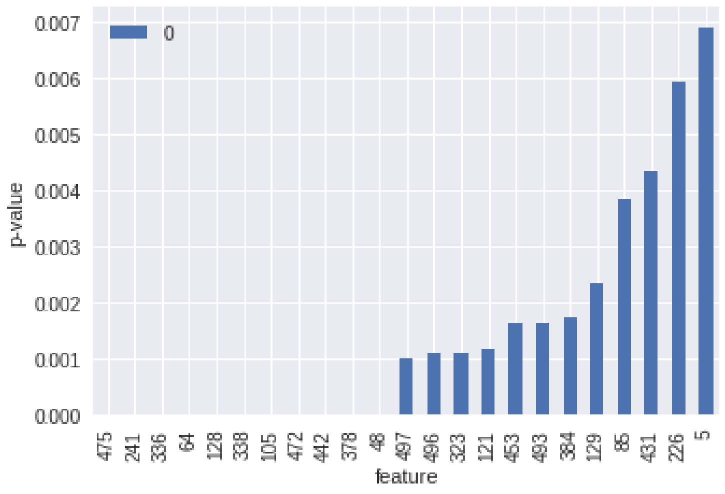

Figure 2: Bar plot of the p-values of each feature (the smaller the bar the better)

A second approach is an unsupervised method, meaning we don't consider the target variable in our analysis. Since we know that there are 20 relevant features, and most of them are linear combinations of each other, we will look at which features "explain" each other. To do this, we will use a "leave one out" regression model (credit Josh Cook). This method takes one feature out (to use as the target) and then uses the remaining features as the explanatory variables in a regression model. We then evaluate the importance of a feature based on the R^2 value from the model. For this particular dataset we used a K-Nearest-Neighbor Regressor since the data is non-linear. Below is a bar plot of the 20 best features based on R^2

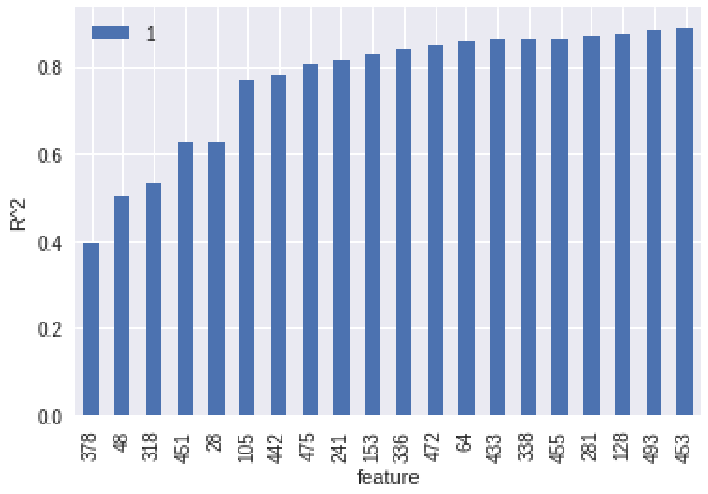

Figure 3: Bar plot of the R^2 of each feature

Let's compare the two method's final features.

Table 1: Comparing features of each method

Method	**Features**
F-value	5, 48, 64, 85, 105, 121, 128, 129, 226, 241, 323, 336, 338, 378, 384, 431, 442, 453, 472, 475, 493, 496, 497
R^2	28, 48, 64, 105, 128, 153, 241, 281, 318, 336, 338, 378, 433, 442, 451, 453, 455, 472, 475, 493

For the most part, both methods end up with the same features. Since the "leave one out" regression method makes intuitive sense, we will move forward with those 20 features.

EDA on the 20 Important Features

Now that we have significantly reduced the number of features, let's do some exploratory analysis. Below is a histogram of the 20 features.

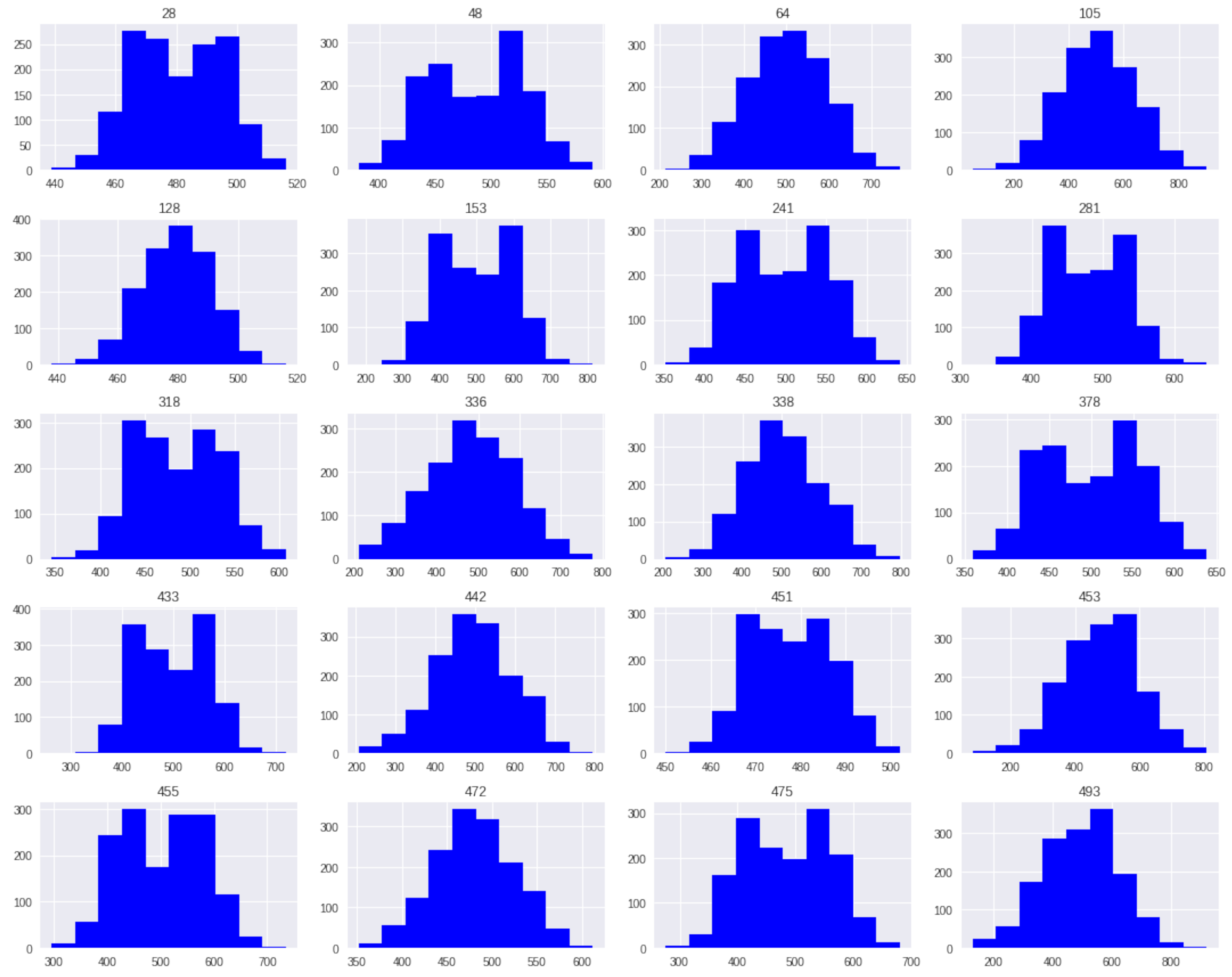

Figure 4: Histograms of the 20 Features

We see that a couple of features are normal; however, a majority of them are either skewed or bimodal. For the skewed features, we will perform a box-cox transformation. Given more time, we would have explored different methods for making the bimodal features normal (possibly splitting them into two different features). We will eventually perform Principal Component Analysis, which does require Gaussian data.

Below is a scatter plot of the 20 features vs a random important feature.

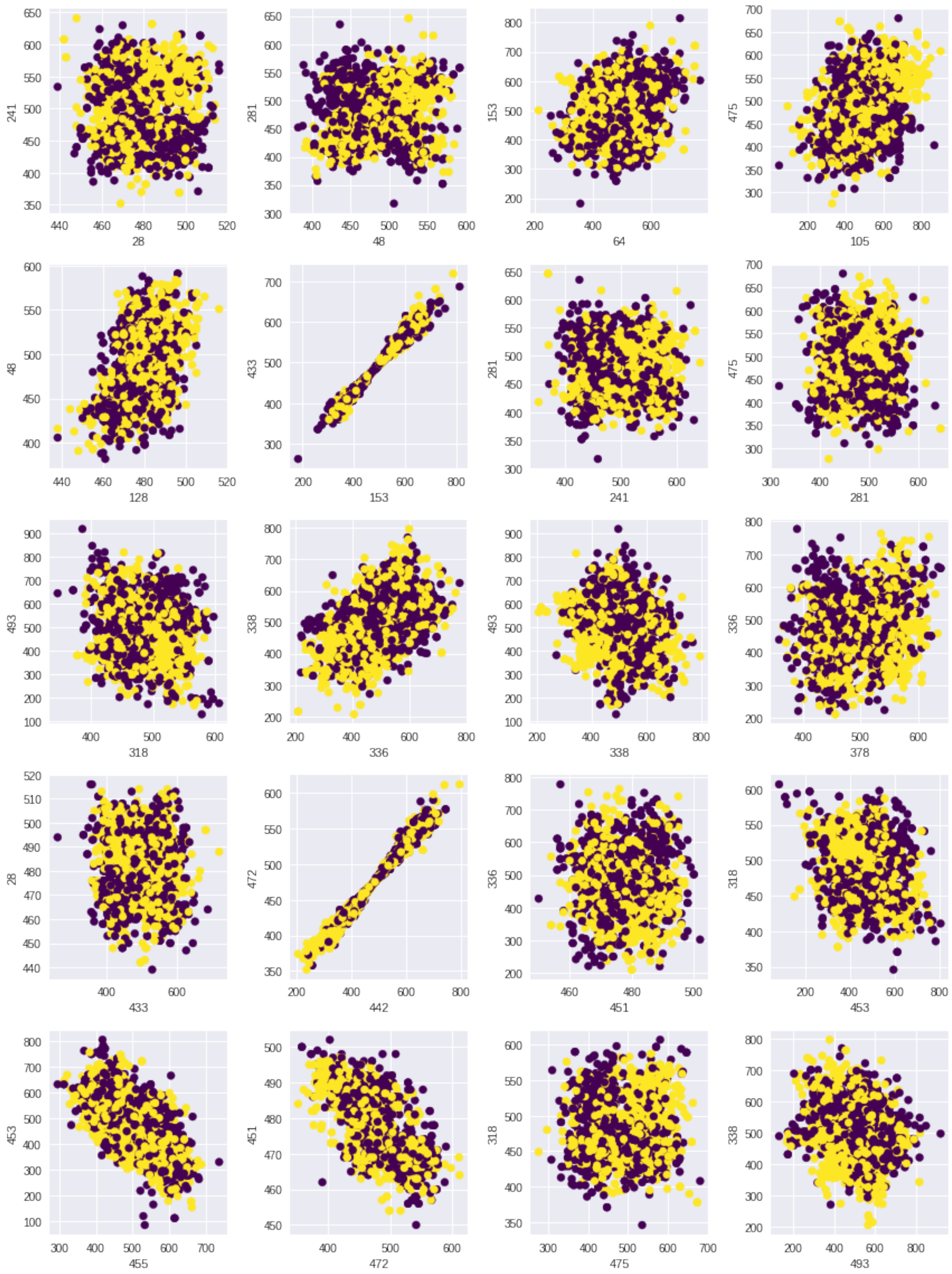

Figure 5: Scatter plot of the 20 important features vs a random important feature (on the y-axis). Different colors represent the classes

This gives us a general idea that we will be able to develop a model that can classify the data fairly well. Also we see that some of the features are highly correlated with each other. We will explore this later on.

Benchmarking

We will look at three different models as our benchmark (Logistic Regression, K-Nearest-Neighbors Classifier, and a Decision Tree Classifier). For benchmarking we will use naive parameters (low/no regularization) Also, since the data is highly non-linear, we expect Logistic Regression to perform poorly. Below is a table comparing our different benchmarking models.

Table 2: Comparing models (considering 20 scaled features)

Method	**Parameters**	**Train Accuracy**	**Test Accuracy**
LogReg	$C = 100.0$	0.6302	0.5547
KNN	$k = 5$	0.9324	0.8907
DecisionTree	max depth = `None`	1.000	0.8240

As expected Logistic Regression did not perform well. K-Nearest-Neighbors with $K = 5$ performed the best with a 89.07% accuracy. Moving forward, we will explore two different methods: further feature reduction, and Principal Component Analysis on these 20 features.

Further Feature reduction

As we saw before in the scatter plots (figure 5), some of the features were highly correlated with each other. Below is a heat map where the features with a correlation higher than ±0.8 are shown.

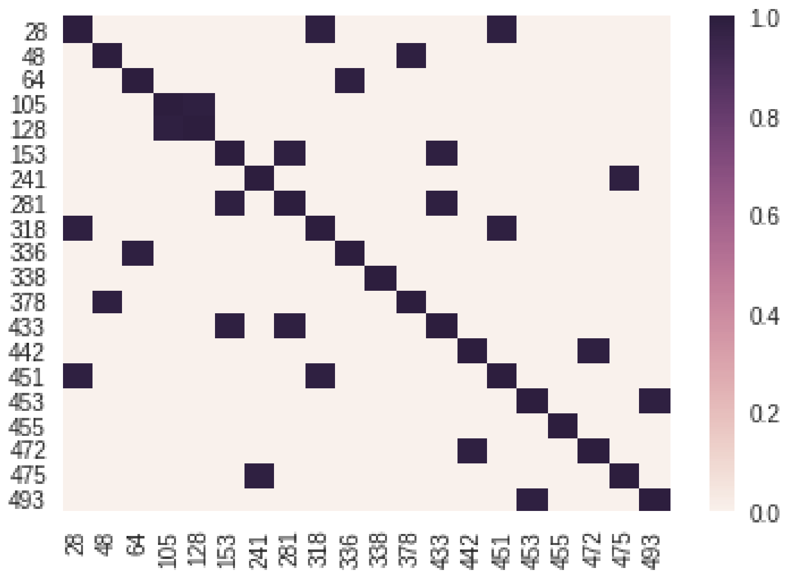

Figure 6: Heat map of highly correlated features

Since we know that 15 of the important features are linear combinations of the "real" 5 features, we manually remove the highly correlated features. This leaves us with 8 final features (shown below).

Table 3: Final 8 features after removing highly correlated features

Features
336, 378, 338, 433, 451, 455, 472, 475

Principal Component Analysis

Another way to deal with the highly correlated features is to use Principal Component Analysis. PCA is the Eigen-decompostion of the covariance matrix where the principal components are the eigenvectors and the explained variance is the eigenvalues. In simple terms is a method of feature engineering/reduction that creates new features that explains a certain amount of the data. The eigenvalues and eigenvectors are ordered and paired (i.e. the ith eigenvalue corresponds to the ith eigenvector).

$$\text{Eigenvalues: } \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$$

$$\text{Eigenvectors: } \bar{e}_1, \bar{e}_2, \ldots, \bar{e}_p$$

Now the Principal Components (orthogonal transformation) are:

$$Y_1 = e_1^T X, \quad Y_2 = e_2^T X, \ldots, Y_p = e_p^T X,$$

So basically we have 20 features and we can reduce these to 5 principal components that "explains" more than 99% of the data. First, before we perform PCA, we need to make the data as normal as possible. We do this using a box-cox transformation. Below is a histogram of the 20 features after being transformed.

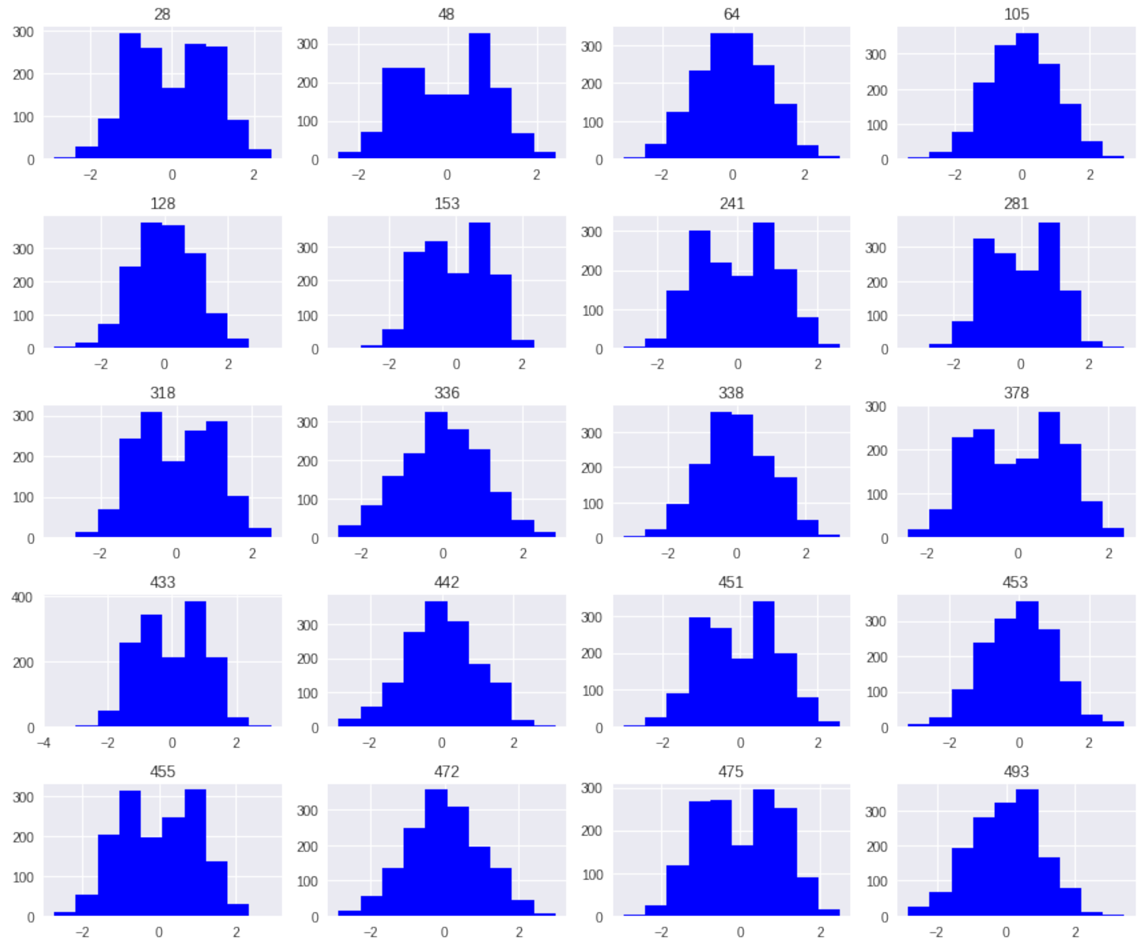

Figure 7: Histogram of the 20 features after being transformed

We see that the data is more normal now (with the exception of the bimodal features). After applying PCA, we have our explained variance ratio below.

Table 4: 5 Principal Components Explained Variance Ratio

PC1	**PC2**	**PC3**	**PC4**	**PC5**	
30.7%	23.8%	20.3%	13.8%	10.6%	

We see that 99.2% of the variance is explained by these 5 Principal Components.

Final Model and Results (UCI)

Moving towards our final model, we have two different feature matrices:

1. 5 Principal Components of the 20 features
2. 8 final features: 336, 378, 338, 433, 451, 455, 472, 475

We will grid search both K-Nearest-Neighbors and Random Forest with these two feature matrices and come up with the best parameters for our final model. Below are the results.

Table 5: Comparing final models

Method	**Feature Matrix**	**Parameters**	**Train Acc**	**Test Acc**
KNN	8 features	$K = 3$	0.9547	0.8960
RandomForest	8 features	max depth = None		
		n estimators = 100	1.000	0.8880
KNN	5 Principal Components	$K = 3$	.9520	0.8960
RandomForest	5 Principal Components	max depth = None		
		n estimators = 50	1.000	0.8747

We see that our best performing model is K-Nearest-Neighbors for both feature matrices with the number of neighbors = 3. The accuracy for both of these is 89.60% which is slightly better than our benchmark accuracy of 89.07%. Below is the confusion matrix for KNN on our 8 most important features.

Table 6: Confusion Matrix for KNN

	True class (1)	**True class (-1)**
Predicted (1)	0.421	0.059
Predicted (-1)	0.045	0.475

We can also see we had a 87.7% true positive rate and a 91.3% true negative rate.

Instructor's Dataset

Now that we have developed an effective method for tackling Madelon, we will apply the same methods on a much larger Madelon dataset (provided by our GA instructors). This dataset has 1000 feature columns and 200,000 observations. Below is a reminder of how we tackled the first dataset.

Collect the data $\Rightarrow$ Reduce to a reasonable amount of features $\Rightarrow$ Benchmark $\Rightarrow$ Further reduction of features/PCA

$\Rightarrow$ GridSearch final model

Collecting the Data

With such a large dataset and such a small computer (1 GB RAM), we have to get creative with how we import the data. With 200,000 observations, we will first gather a sample of 4000 (99% confidence and 2% margin of error). After we significantly reduce the number of features, we will then be able to gather more observations (100,000) to train and test our models.

Reducing Features

We will use the same "leave one out" regression method that we used on the first dataset. Below is a bar plot comparing the features and there R^2.

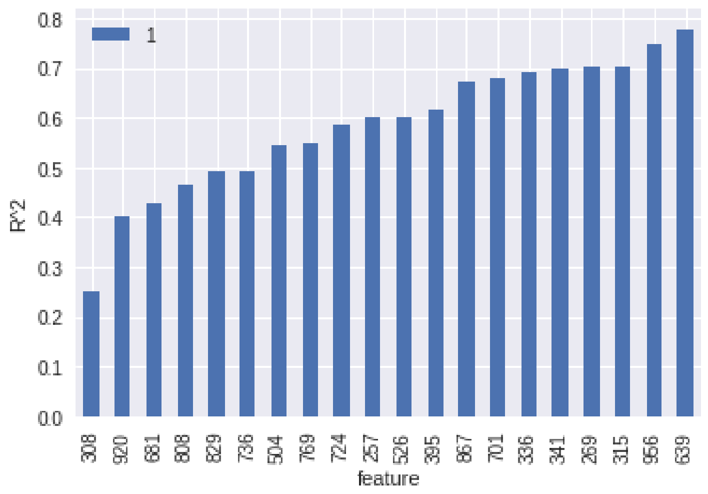

Figure 8: Bar plot of the R^2 of each feature

Again we end up with 20 important features.

Table 7: Comparing features of each method

Method	Features
R^2	257, 269, 308, 315, 336, 341, 395, 504, 526, 639, 681, 701, 724, 736, 769, 808, 829, 867, 920, 956

We will move forward in our analysis with these 20 features.

EDA on the 20 Features

Below is a histogram of the 20 features.

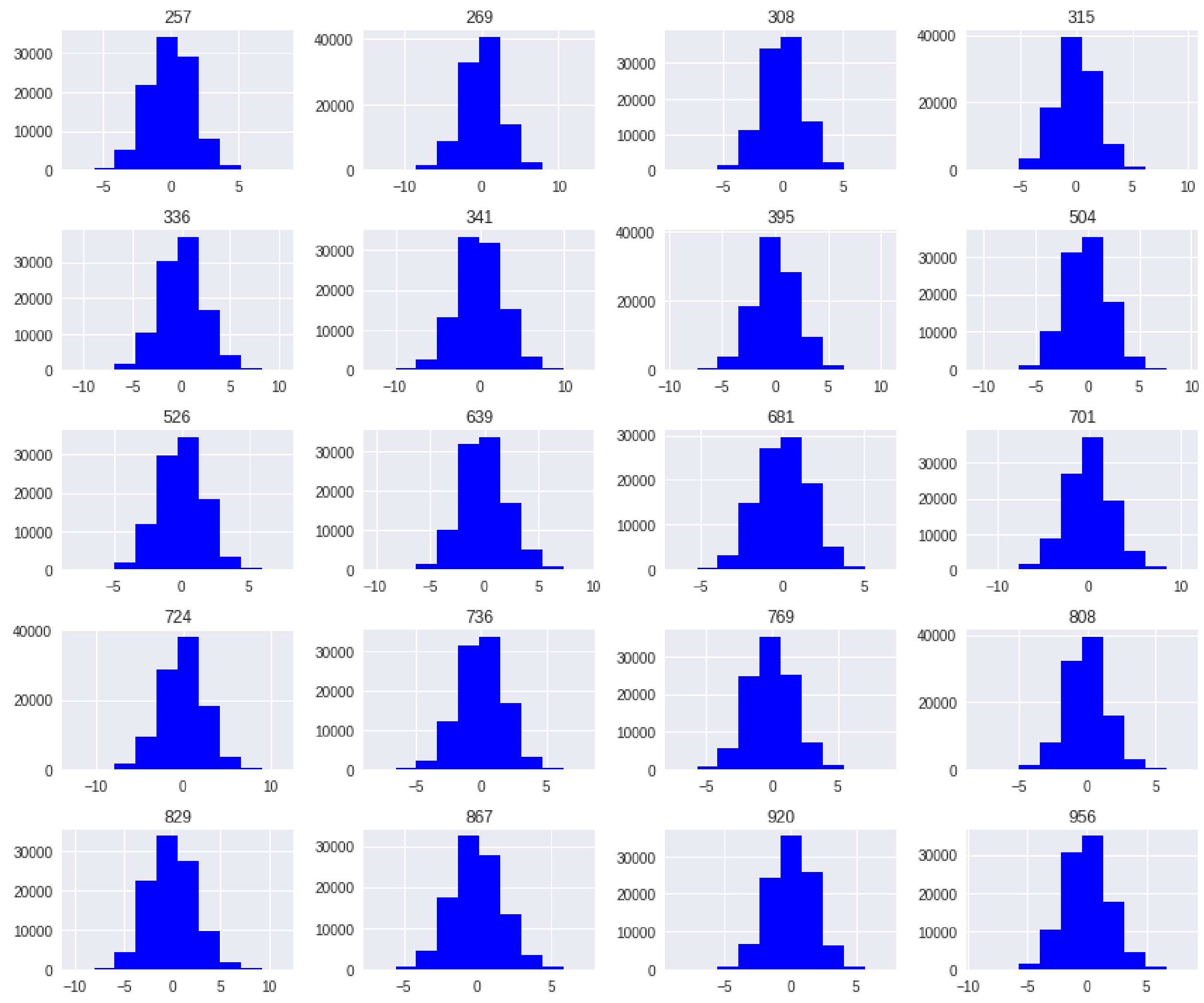
Figure 9: Histograms of the 20 features

We see that all of the data is fairly normal, and there are no bimodal distributions. This means we will not be doing a box-cox transformation before applying PCA. Also the data is already scaled (everything is centered around 0). Now let's take a look at the correlations.

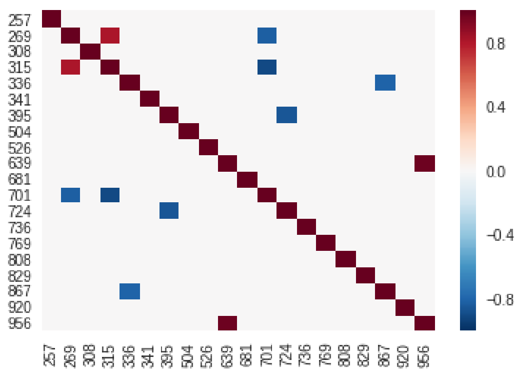
Figure 10: Heat map of the features with the highest correlations

We see that there is also less high correlations in this dataset compared to the first one. So when we remove the features with the highest correlations, we are left with the 11 most important features

Features
257, 308, 341, 504, 526, 681, 736, 769, 808, 829, 920

Benchmarking

Again we will naively fit Logistic Regression, KNN, and a Decision Tree to come up with our benchmark.

Table 8: Comparing models (considering 20 scaled features)

Method	Parameters	Train Accuracy	Test Accuracy
LogReg	$C = 100.0$	0.6058	0.6023
KNN	$k = 5$	0.8953	0.8472
DecisionTree	max depth = `None`	1.000	0.7848

We see that KNN performed the best again with a 84.72% accuracy.

Principal Component Analysis

We will perform PCA with 5 components. Below is the explained variance ratio of each of the 5 components.

Table 9: 5 Principal Components Explained Variance Ratio

PC1	PC2	PC3	PC4	PC5	
36.6%	32.0%	13.7%	10.2%	7.54%	

With these 5 components, we see that 99.9% of the variance is explained.

Building Final Model

Now to build our final model on this dataset. We will use two different feature matrices:

1. 5 Principal Components of the 20 features
2. 11 final features: 257, 308, 341, 504, 526, 681, 736, 769, 808, 829, 920

We will grid search both K-Nearest-Neighbors and Random Forest with these two feature matrices and come up with the best parameters for our final model. Below are the results.

Table 10: Comparing final models

Method	Feature Matrix	Parameters	Train Acc	Test Acc
KNN	11 features	$K = 9$	0.8810	0.8494
RandomForest	11 features	max depth = None		
		n estimators = 100	1.000	0.8563
KNN	5 Principal Components	$K = 7$	.8853	0.8486
RandomForest	5 Principal Components	max depth = None		
		n estimators = 50	1.000	0.8523

The best performing model with this dataset was a Random Forest Classifier with a max depth of None, and a number of estimators = 100. The models also performed slightly better with the 11 important features. Below is a confusion matrix for Random Forest with the 11 features.

Table 11: Confusion Matrix for Random Forest

	True class (1)	**True class (−1)**
Predicted (1)	0.430	0.071
Predicted (−1)	0.073	0.426

We can also see that we have 85.9% true positive rate and a 85.3% true negative rate.

Conclusion

The best model for the UCI dataset ended up being a K-Nearest-Neighbors with $K = 3$ with 8 features with a testing accuracy of 89.60%. The best model for the instructors dataset was a Random Forest with 11 features. Below is a table of the best results.

Table 12: Comparing final models

Method	**Feature Matrix**	**Parameters**	**Train Acc**	**Test Acc**
KNN	8 features (UCI)	$K = 3$	0.9547	0.8960
RandomForest	11 features (Instructors)	max depth = None n estimators = 100	1.000	0.8563

Table 13: Comparing final features of each dataset

Dataset	**Features**
UCI	336, 378, 338, 433, 451, 455, 472, 475
Instructors	257, 308, 341, 504, 526, 681, 736, 769, 808, 829, 920

Ideally we would have liked to reduce down to the "real" five features in each dataset. However, we lost too much accuracy when we tried to reduce it down further. The "leave one out regression" combined with removing the highly correlated features of those 20 features proved to be effective with the Madelon dataset. Principal Component Analysis was also effective.

Given more time, we would have explored more options in our analysis. For the UCI dataset, we would explore how to handle bimodal data. Along with that, we would have explored more classification models (possibly a Bayesian classifier). Also, it's possible recursive feature elimination would be more effective in reaching our goal of 5 final features.