**Dimensionality Reduction:**

**Principal Component Analysis**: It is a means of recognizing patterns and expressing data in a way that highlights their similarities and differences[[1]](file:///C:\Users\aksha\OneDrive\Documents\GitHub\who-is-more-influential\principal_components.pdf). We use it to extract a low dimensional set of features from the existing high dimensional feature set with the aim of capturing as much information as possible[[2]](https://www.analyticsvidhya.com/blog/2016/03/practical-guide-principal-component-analysis-python/). We use the first six principal components generated by PCA since they cover about 90% of the variance of the original dataset.

**(Add along with feature selection)**

***k*-fold Cross Validation:** In order the analyze the performance of the various models we use *20-*fold cross validation. In this technique, we divide the dataset into *k*-subsets. In each iteration one of the *k*-subsets is used as the test set and the *k-*1 subsets form the training set. The average error is then computed across all the *k*-trials[[5]](https://www.cs.cmu.edu/~schneide/tut5/node42.html).

**Methods:**

**Logistic Regression:** After visualizing the data it seems feasible that the data is linearly separable. Thus, we begin by modelling the data using a linear classifier. We use the Python Scikit-Learn library’s implementation of Logistic Regression[[3]](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html) to do the analysis on various data transforms discussed in the section (add number of the section).

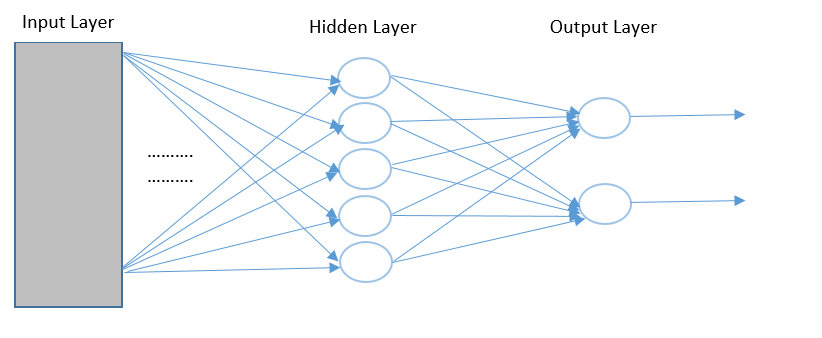
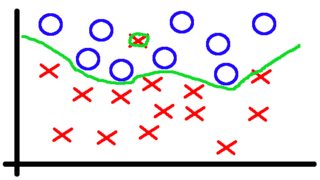
The high accuracy obtained with this model supports our hypothesis that data is indeed linearly separable. To boost the performance of the model and understand the influence of each feature on the accuracy, we re-run the model with a smaller set of features and capture the difference in the performance.

**Support Vector Machines (SVM):** Another classical model for linear classification is the support vector machines. We use it on all the transforms discussed in the section (add number of the section). We evaluate the performance of the model by using various kernels provided in the Python Scikit-Learn library’s implementation of the SVM[[4]](http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html).

The performance of non-linear kernels is not as good as that of the linear kernel. This also supports our hypothesis of linear separability of the data.

**Neural Network**: Although non-linear SVM did not perform as well as expected but we still wanted to try another non-linear classifier for this problem. We decided to go with neural network which might be a good choice because of its high capacity.

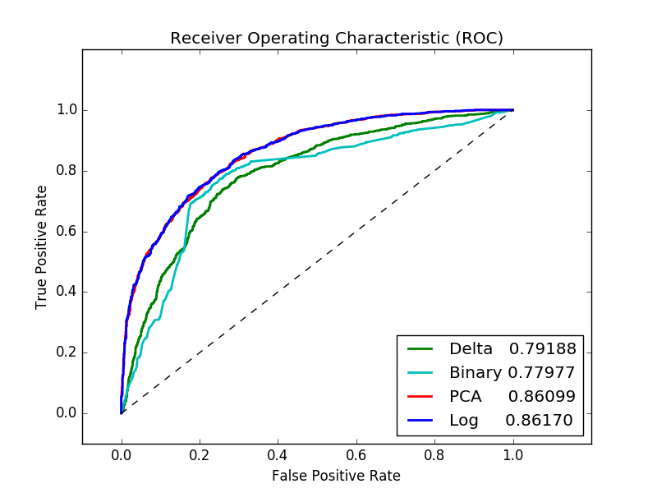
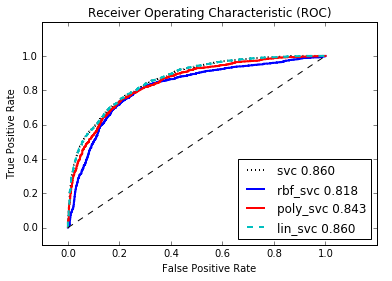
We use Python Scikit-Learn Library’s Multi-Layer Perceptron algorithm which is designed for classification problems. We are using a single hidden layer neural network where the hidden layer has 5 perceptrons. The output layer has 2 perceptrons. The network structure is shown in figure # with an “Identity” activation function in the output layer. The activation of the hidden layer is customizable and we will try out different activation for the hidden layer and compare their performances.

**Gradient Boosting:** Experiments with non-linear kernels in SVM and Neural Networks with non-linear activations indicated that overfitting was affecting non-linear models. Hence, we used Scikit-Learn Library’s Gradient Boosting which is an ensemble model that combines many weak learners to form a strong learning hypothesis and is known for its robustness to overfitting. Boosting algorithms treat the learning problem as an optimization problem where the objective function is the error function and the goal is to minimize it. Although there hasn’t been any theoretical explanation to why Gradient Boosting does not overfit, empirical results show that due to extremely high level of non-linearity the decision boundaries around the outlying points become so narrow that no other point can fit into that. Hence it does not lead the model to incorrect predictions.

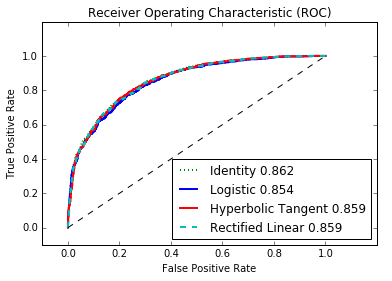
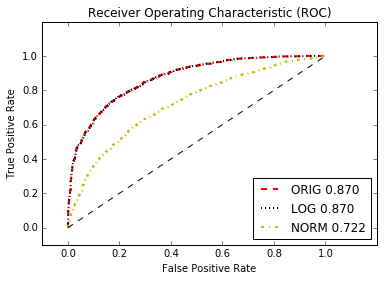
**Experiments:**

**Logistic Regression:** We begin by evaluating the performance of the performance of model on various data transforms. From the ROC curves in Figure # , we see that the log transform and the first six principal components from PCA performs the best with an accuracy of 86.17% and 86.099% respectively. The Normalized and the binary transform fail to give fruitful results because of the reduced variance in these transforms. Moreover, these transforms fail to attribute for the scale of the data.



**Support Vector Machines:** The performance of the SVM with a linear kernel gave very similar results as compared to Logistic Regression. It yielded an accuracy of 86% with the log transform. To improve the accuracy of the SVM, we tried to map our input feature set to a high dimensional space. We used the radial bias function or the Gaussian kernel and a 3-degree polynomial kernel for doing this[[6]](http://scikit-learn.org/stable/modules/svm.html#kernel-functions). However, this did not improve the accuracy of the model since the original feature set has around 11 features. Mapping these further to a high dimensional space could not improve performance[[7]](https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf). The ROC curves of the SVM indicate this. We prefer the linear kernel over the non-linear kernels since it is a simpler model and is less prone to overfitting.

**Neural Network:** In this model, we tried four different activations for the hidden layer – Identity, Sigmoid, Rectified Linear and Hyperbolic Tangent. Our experiment shows that the model performs best with Identity activation which effectively makes the Neural Network a linear classifier and we are getting an accuracy of 86.2%. The non-linear activations are also performing quite well with an accuracy of around 85.6%. The comparison is shown in the ROC curve of figure #. We also tried with different transforms and among them the Logarithmic transform performed the best.

 ****

**Gradient Boosting:**We used Gradient Boosting with the original data, the logarithmic transform and the normalized transform and we got the best results of 87% accuracy using the former two transforms. We used a model with 100 layers of decision tree which produced an extremely non-linear boundary and yet did not overfit owing to the property of Gradient Boosing. The comparison of the results can be seen from the ROC curve of figure #.