Data Science Case Study

Overcoming the 'Curse of Dimensionality' on High Dimensional Datasets

Ahmad Khan

Problem Overview

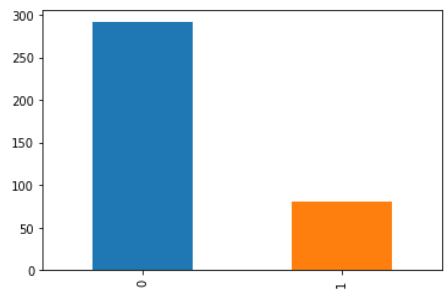
A classic academic problem with Machine Learning in high dimensional datasets is the "Curse of Dimensionality" where traditional supervised learning methods end up overfitting, and don't perform as well, when the dimensionality of a dataset increases dramatically such that the number of dimensions M is magnitudes greater than the number of training examples N. In this paper machine learning models were trained to predict a binary 'response' variable using an extremely high dimensional dataset with 16K anonymous features but only 500+ training examples using various supervised and unsupervised learning techniques including feature selection/wrapping, dimensionality reduction, and clustering to see how effective each were in overcoming the "Curse of Dimensionality" problem.

Part 1 - Analysis of Model Performance on High Dimensional Data

We start the analysis by first splitting the data randomly between train and test. Here train was 70% of the data, and 30% held out as test data to be used final model evaluation in the end. Instead of creating a separate validation dataset, cross validation was used because first by repeatedly sampling and creating multiple folds we can achieve a better measure of how well the model generalizes, and we can reduce the impact of seeding when splitting the data. Finally since this data is extremely small (500+ rows) it doesn't take much time to train and so repeated sampling and training is fast making CV a time efficient exercise and without making us lose valuable training examples to the validation set. Since the data is fully anonymous in terms of feature descriptions, we cannot assume much about the kind of data we are using (other than the fact that it is binary). As such, we will focus more on optimizing dimensionality to help overcome the Curse of Dimensionality instead of deriving new features through feature engineering using prior domain knowledge.

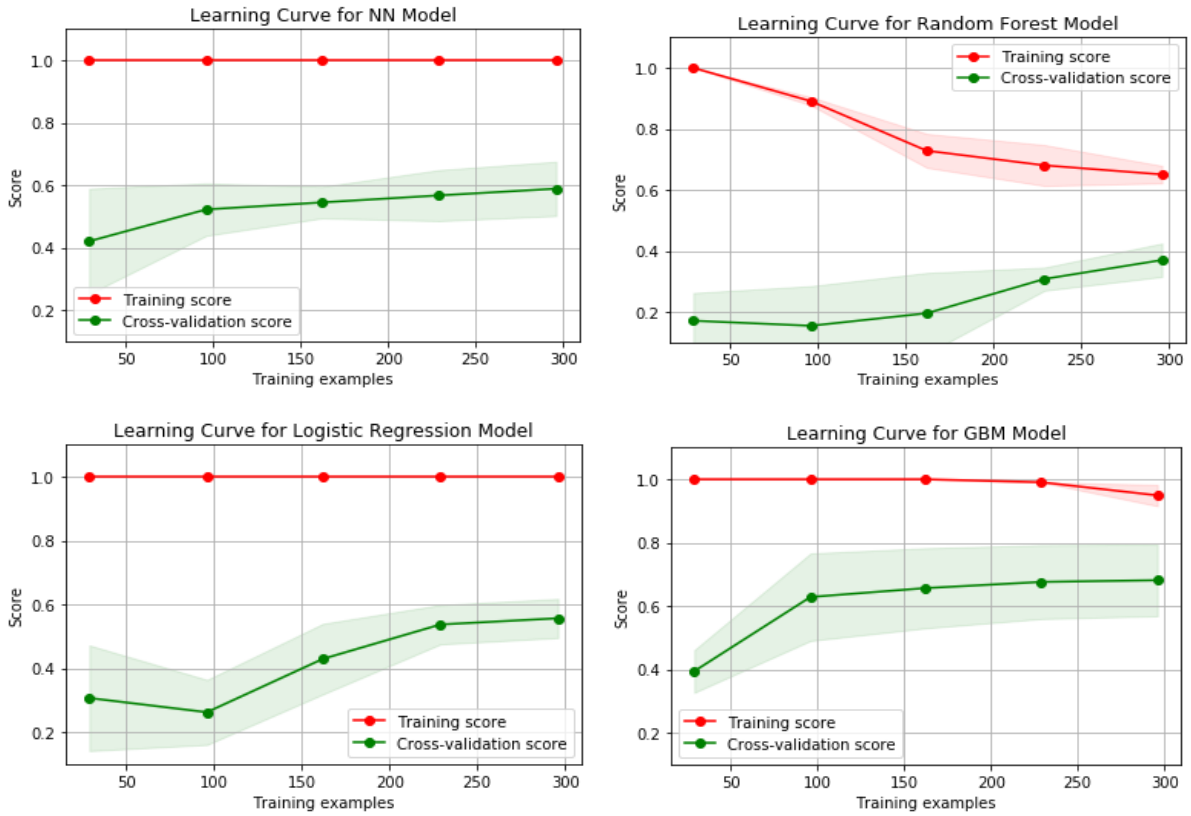
First a random baseline which every algorithm should do better on is established. Each training label is predicted randomly but proportional to the weight of each class occurrence in the target label. Shown below is the distribution of the positive and negative "responses" in the data. We can see that the positive class with label = 1 occurs approximately 25% of the time, and negative response the remaining 75%. The performance of this random baseline where 1s and 0s are predicted according to their weight is as follows: Accuracy 0.654, F1 Score: 0.247, Precision 0.233 and Recall 0.2625. Since this dataset's target classes are somewhat imbalanced accuracy can be a misleading metric to optimize on. For this reason F1 Score will be used as the primary metric of evaluation but Precision, Recall and Accuracy will still be looked at for sanity checks on model performance.

Occurence of Positive and Negative Responses in Training Data



Next, we establish a more rigorous baseline where the following algorithms Random Forest, Logistic Regression (with L2 or L1 regularization), Gradient Boosting, and a simple one layer Neural Network are all trained on the entire dataset with 16K features. The choice of these algorithms is intentional. Log Regression can do well with less data, but all the other algorithms generally require

large volumes of training data (especially Neural Networks) to perform well which allows us to set up a nice contrast. Second, most of these models focus on reducing Error by decreasing Variance either through some form of regularization (Log Reg/Neural Nets) or through repeated sampling or “bagging” (Random Forests) which makes them ideal suited to a problem where we are likely to have high Variance because of the “Curse of Dimensionality” and the tiny amount of training samples (500+ examples). GBM is included as a contrast to the other 3 because it mainly attacks Bias. Results for each algorithm are shown by plotting Learning Curves on the training set. The learning curves simply show how both the training performance (F1 score) and 5 fold cross-validation performance (F1 score) varies for the model as the number of training examples N grows. They also allow us to better diagnose model performance and error in terms of its bias and variance.



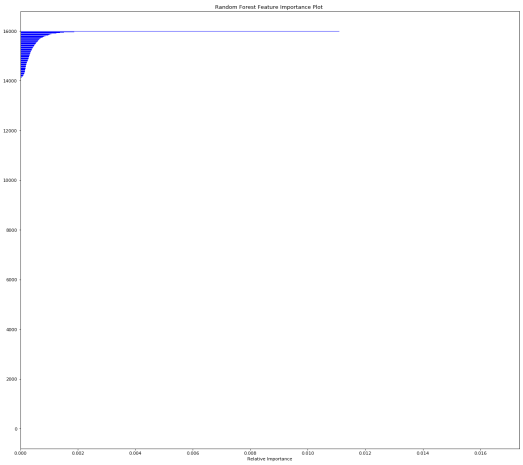
From the above learning curves we can observe two things, First, each model easily beats the random baseline of 0.247 F1 Score. At the same time however, all models suffer from extremely high variance as seen from the huge gap between their training performance and their cross validation performance even as the number of samples N grows. Finally even though variance is huge bias is low or non existent as most models except the Random Forest are able to learn perfectly on the training set with zero or low training error. This illustrates the initial ‘curse of dimensionality’ suspicion we had as all models are severely overfitting. That is the models have large Variance but low Bias which means the model learns the noise in the dataset as opposed to the true underlying hypothesis function and therefore is extremely sensitive to even slight changes in out of sample data. Variance and consequently overfitting however can be overcome by 1) adding more training data 2) reducing the dimensionality of the dataset 3) reducing model complexity. More training examples allow the model to generalize better but since we cannot add more training examples to our dataset and since the choice of our parameters for each model were already on the low complexity side (only 100 trees for Random Forest, 32 hidden nodes for Neural Net etc) we will instead now focus on the part which we can control, reducing the dimensionality of the dataset either through 1) Feature Selection 2) Dimensionality Reduction or 3) Clustering.

Part 2 - Analysis of Algorithm Performance on Reduced Data

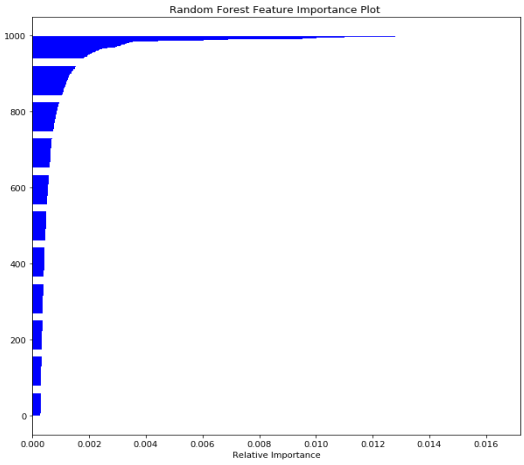
1. Feature Selection Methods

We start this section by first reducing the dimensionality of the dataset by reducing the number of features we use. The key assumption behind this approach is that most features contribute very little in information gain during training either due to being highly correlated with other features in the dataset or simply being useless or not predictive. There are many feature selection methods (Lasso Regression for example) but in this paper we will focus on one of the simplest ones: Random Forest Feature Selection. When training Random Forests we can see which feature contributes the most to “information gain” when the individual decision trees are created. These are simply the features which each decision tree in the random forest chooses the most (and usually at the top most levels) of the tree for splitting the data because they lead to most reduction in entropy (as inferred from the gini coefficient) for the resulting split dataset. Features that get selected the most by extension can also be considered more predictive. Below is the feature importance plot of the random forest for all features (left) and then the top 1000 features (right) with each feature on the y axis and its importance (gini coefficient) on the x axis.

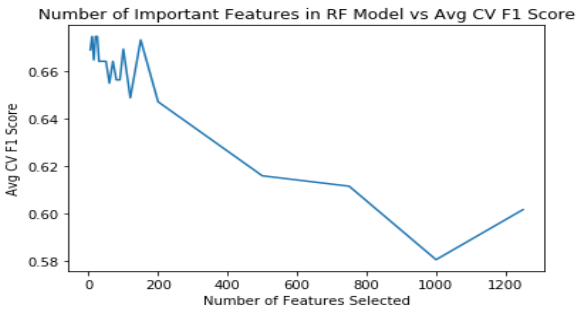
Random Forest Feature Plot (All 16K Features)



Random Forest Feature Plot (Only top 1000 Features)



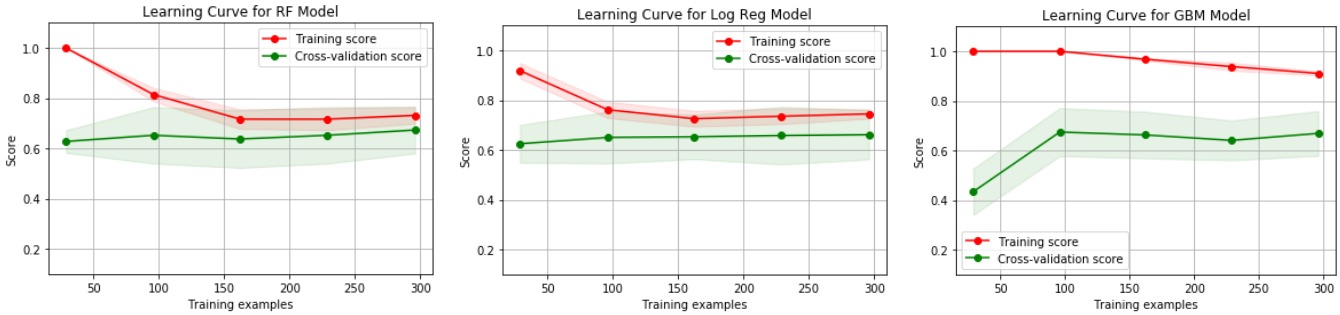
From the two plots we can conclude the following. After the first 2K or so top features there is close to zero contribution to information gain in the model. As such having these features is close to useless as they add little in predictive power while worsen the model in terms of complexity. Second even when filtering for only the top 1000 features (right chart) wee see that there is a sharp drop off in the gini coefficient of these features after just 50 or so features. So even out of the important features which have some positive values only 100 or so show a lot in terms of gain after which the contribution of the remaining features in improving model performance is limited. To find the “optimal model” the random forest was trained/evaluated using different number of features and performance noted as seen below.



	Number of Top Features	CV F1 Score for RF
4	25	0.674462
3	20	0.674462
1	10	0.674462
14	150	0.673004
12	100	0.669128

From the above plots we can see that CV F1 Score is high when the number of features is small up to 150 features (and even smaller combination of features with N=10 and N=25 achieve even better performance). More importantly, the performance of the Random Forest with 150 features is 0.67 F1 Score which is much better than before easily beating the previous benchmark of 0.40 F1 Score for the Random Forest in the RF All Features Model in Part 1. However, after 150 features performance of the Random Forest starts degrading. This subselection of only the top 150 top features was also tried on the other algorithms Neural Network, Log Regression, and GBM model with new learning curves plotted.

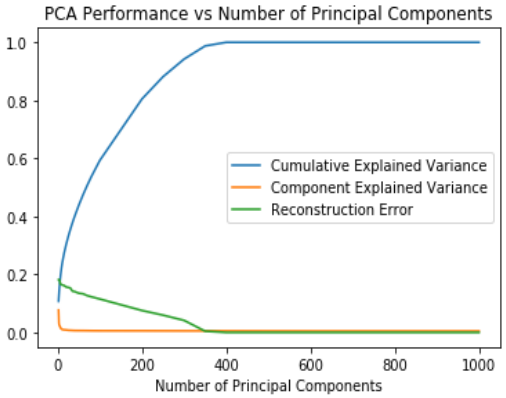
Learning Curves for each model after applying Random Forest Feature Selection



We can see that the variance and subsequent in overfitting in the models has been reduced after applying feature selection for Logistic Regression and Random Forests as the gap between the training error and CV error has shrunk (although at the expense of increased Bias). Second, the out of sample CV F1 Score performance is greater at around 0.65 for each vs the 0.40-0.58 F1 range before. Therefore by doing simple feature selection and reducing the dataset to 150 features, overfitting from high variance was reduced for at least the logistic regression and random forest models. However, for the GBM model the performance is similar which makes some sense especially for GBMs as Gradient Boosting aims to reduce bias as opposed to attacking variance and so will be less sensitive to feature reduction.

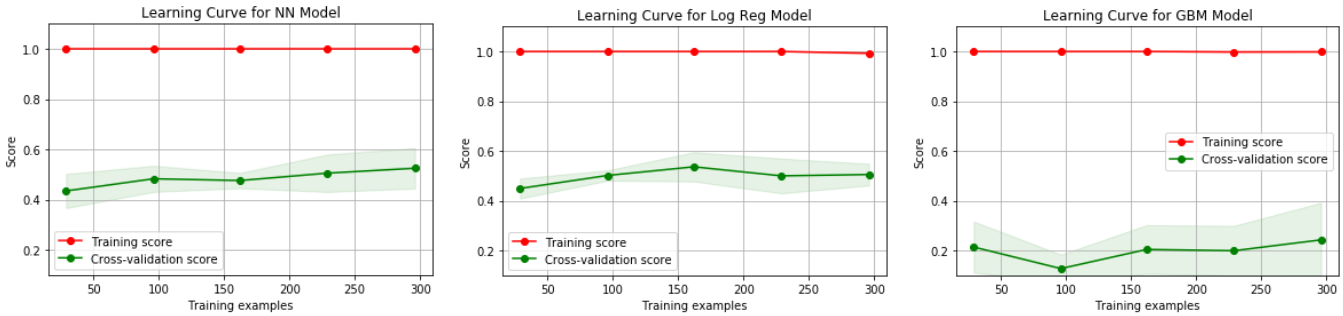
2. Dimensionality Reduction Methods on Full High Dimensional Data

Next, we focus on dimensionality reduction (DR) methods. Unlike feature selection, DR methods aim to find a reduced linear combination of the feature set such that most information is retained. We will look at 3 different methods here. Principal Components Analysis (PCA), Independent Components Analysis (ICA) and Linear Discriminant Analysis (LDA). PCA aims to reduce the feature set by reducing the dataset through linear decomposition such that the reduced “principle components” are eigenvectors which maximize the variance in the data. By maximizing variance in the data as the criterion to reduce the dataset, PCA effectively captures most information about the high dimensional dataset but in a reduced dimensional space. ICA is similar to PCA but instead of maximizing variance it aims to find the components which maximize independence. The key idea being that the data is generated by multiple independent non-gaussian signals which can be decomposed linearly into independent projections. Finally LDA, achieves dimensionality reduction by accounting for class label information such that the linear combination of the reduced feature set maximizes the separation between the class labels thus allowing not only a highly reduced dataset but also one which tries to maintain class separability. To run PCA on the



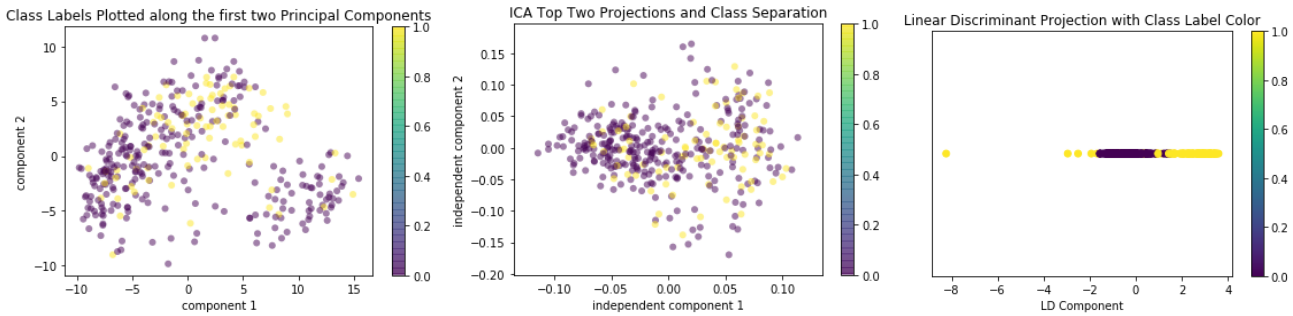
16K feature set, first the optimal numbers of components was observed by looking at the point where most of the variance (>=90%) was explained. This comes out to be around 300 components (See curve above). Then PCA with 300 components was run on the full dataset and the learning curves plotted for the models to see how model performance changes after PCA. Unfortunately from the learning curves above it seems that PCA did not seem to improve model performance and it actually worsened F1 Score.

Learning Curves for each model after applying PCA to the High Dimensional Dataset



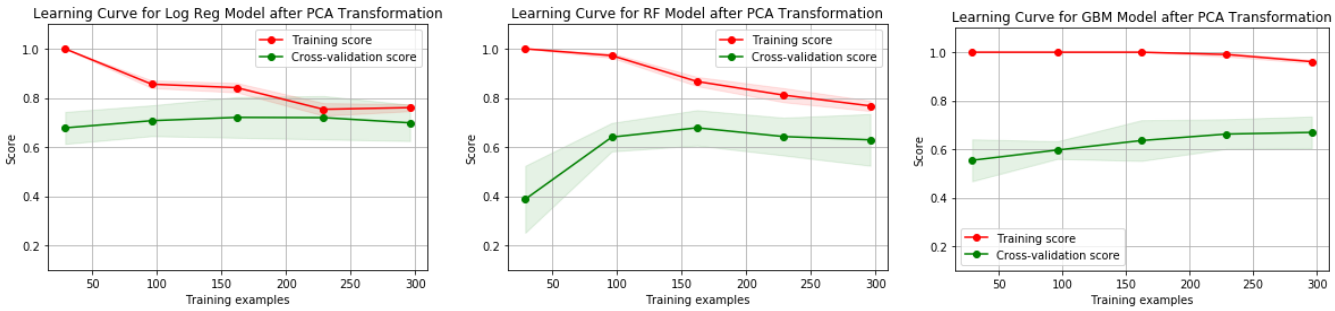
This could be because while PCA is great at reducing the dataset it might lose information related to class separability during the decomposition. For example the left most plot below shows the top two Principal Components and the location of each training point in the reduced 2D space with their colors corresponding to the class label (purple = 0, yellow = 1). We can see that after doing PCA the top two components have little separation in the true class labels as both colors overlap heavily and t is hard to draw a clean line through the 2D surface and achieve great separability of classes. Similarly after running ICA we see the same issue with the top two components not able to achieve clean separability. And when LDA is run we can see that the purple (negative class) is unfortunately sandwiched between.

Class Separation among top two components after applying PCA (left), ICA(middle), LDA(right)



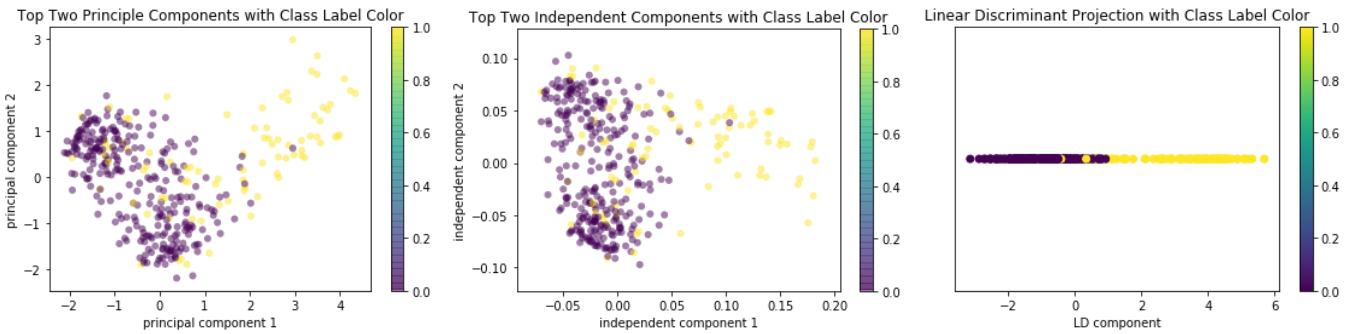
3. Dimensionality Reduction Methods after Feature Selection

The above results don't necessarily imply that Dimensionality reduction cannot work in improving model performance. As noted the dimensionality of the dataset is huge with most features completely redundant. However, it is possible to guide DR methods towards better performance by first running feature selection and eliminating redundant features and then applying Dimensionality Reduction. This approach can help because it simplifies the feature hyperplane space by eliminating clutter in which the reduced linear combination of features is found. Hence, each DR method was now run again after feature selection done using Random Forests earlier (keeping the top 150 features) and performance of each algorithm noted. 50 components were kept this time as it is the point where 80% of the variance is explained. Learning Curve of the Log Regression, Random Forest and GBM were plotted again below. This time the results are more encouraging.



We can see that the error from variance has decreased vs the all features model in the beginning. Although in terms of F1 Score it is almost the same as the Feature Selected model because bias (from increased training error) has increased as well. Even though the model performance as measured by F1 Score doesn't beat the Feature Selected Model it has still improved by >10% over the original All Features model for Random Forests/Logistic Regression and performance has remained somewhat similar for GBM and Neural Nets despite massive dimensionality reduction from 16K features to just 50 features now. Visualizing the datasets in reduced 2D principal components space below we can see why; PCA, ICA, LDA achieve much cleaner separations in class labels once they are applied to the reduced feature-selected data of 150 features as opposed to when they were applied to the huge 16K dataset.

Class Separation among top two components after applying Feature Selection First and then PCA (left), ICA(middle), LDA(right)

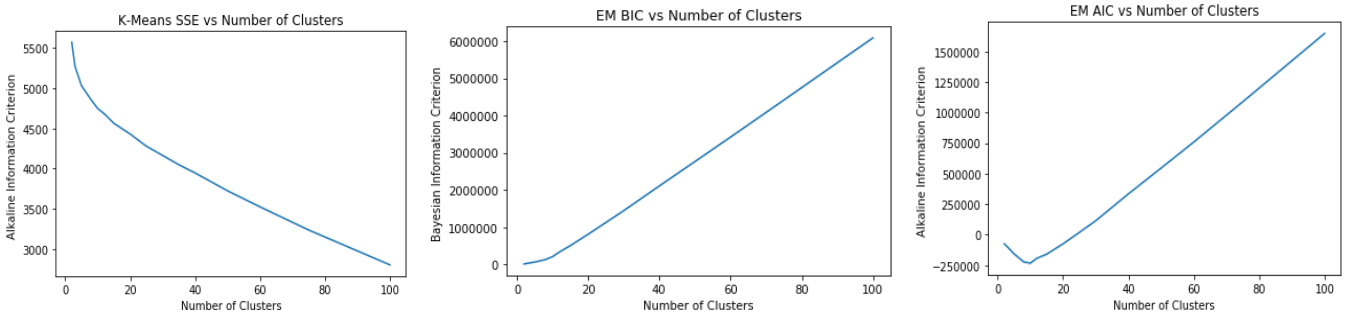


4. Clustering Methods after Feature Selection

So far we have utilized Dimensionality reduction to achieve better supervised learning performance but clustering techniques can also be effective techniques in reducing the dimensionality of the data and also as feature augmentation methods where the assigned cluster of a data point is treated as a "feature" and added back to the original training dataset. Two different clustering methods are utilized, K Means and Expectation Maximization. K Means finds hard decision boundaries by iteratively assigning points to clusters such that the inter cluster distance is minimized but the intra cluster distance is maximized until there is no more improvement. Expectation maximization assigns soft probabilities of belonging to a cluster. Before running clustering on the dataset, the optimal number of clusters needed to be determined. For K Means the 'elbow point' where SSE shows a marginal decrease in SSE reduction is used as the optimal cluster point. For EM, it can be the point where the Bayesian Information Criterion (BIC) or Alkaline Information Criterion (AIC) is minimized. First clustering was done only on the feature selected dataset to speed convergence. Plots below show the optimal number of K Means clusters to be approximately 8 and the optimal number of EM clusters to be also approximately 5. First, cluster Labels were used as individual features to train on but this led to severe degradation in CV F1 Score and the

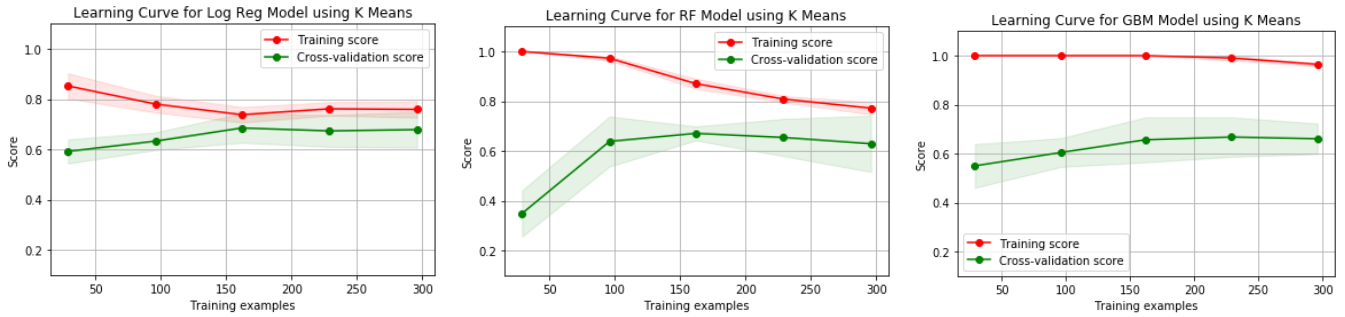
approach discarded. Next, each of the 4 algorithms were re-trained with their K Means assignment or the EM cluster probabilities vector appended to the 50 principle component vector feature set from earlier.

K Means Clustering SSE vs Number of Clusters **EM AIC and BIC vs Number of Clusters**

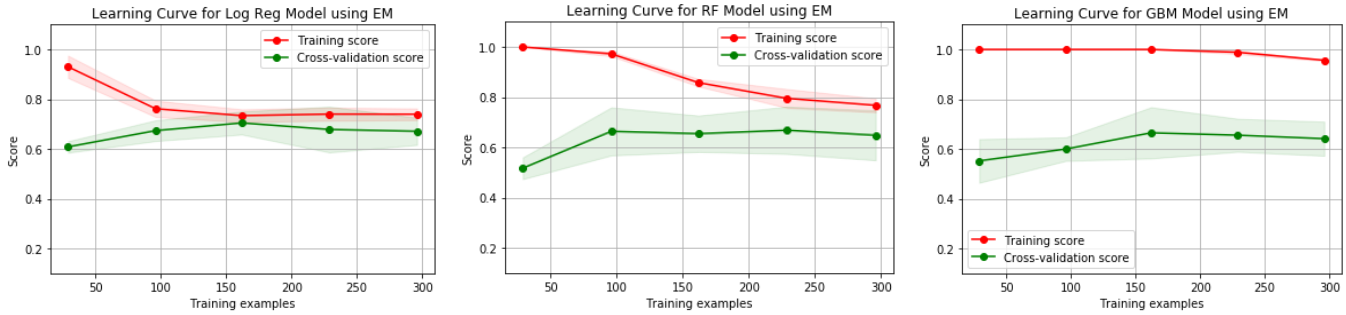


Learning curves for the models using cluster assignments as features are plotted below. However, after “feature augmentation” of cluster labels to PCA components the resulting performance of each model was almost the same as before when running the Models with only Feature Selection + PCA .

Learning Curves using K Means Cluster Assignments only as Input Features (Neural Net not shown due to lack of space).



Learning Curves using Expectation Maximization Cluster Probabilities only as Input Features



Part 3 - Final Evaluation of Models on Hidden Test Data

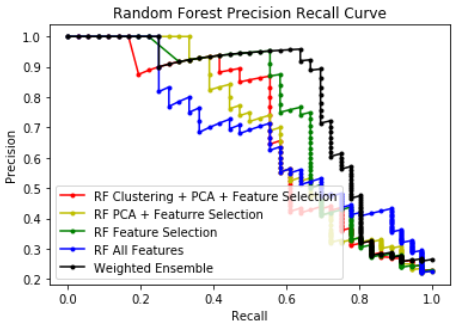
So far all models have been trained on the 70% of data that was randomly split into train and model performance evaluated using Cross Validation using the same train data. In addition, all DR, Clustering, Feature Selection was also done on the training data. While Cross Validation ensures that one does not ‘cheat’ by looking at the class labels itself it nonetheless can lead to overfitting to the training data. Second, while unsupervised learning methods like PCA and Clustering do not use class labels explicitly it is entirely possible to “peek” into the true labels themselves if components or clusters are strongly correlated with class labels and the DR/Clustering method is fitted on the same data used for CV (ideally we would have trained the PCA and Clustering algos on a different set of data). For this reason to evaluate final model performance (after doing hyperparameter grid search tuning for each model) all

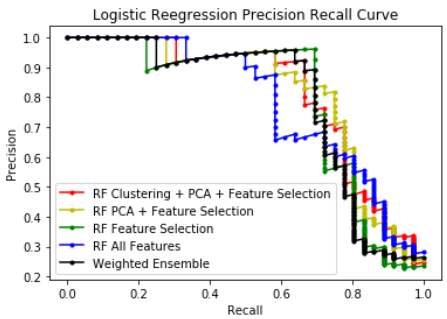
models were now tested on hidden test data which had not been utilized thus far (for DR only PCA is shown as ICA/LDA perform similarly) with performance summarized in the table below.

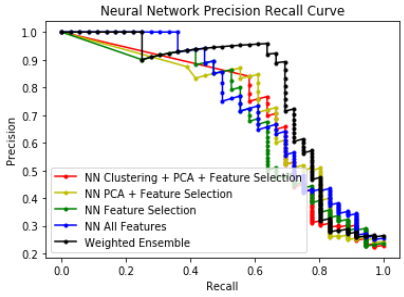
Performance of all models on test data ranked from worst performing to best (Note: Hyperparameter grid search results are in code)

	Model	F1	Precision	Recall	Accuracy
3	Random Forest All Features	0.415	0.647	0.306	0.805
11	Random Forest: Feature Selection + Dimensionality Reduction (PCA)	0.538	0.875	0.389	0.849
14	GBM: Feature Selection + PCA + Clustering (K Means)	0.538	0.875	0.389	0.849
10	GBM: Feature Selection + Dimensionality Reduction (PCA)	0.566	0.882	0.417	0.855
1	Neural Network All Features	0.610	0.783	0.500	0.855
15	Random Forest: Feature Selection + PCA + Clustering (K Means)	0.632	0.857	0.500	0.868
5	Neural Network: Only Feature Selection	0.638	0.667	0.611	0.843
9	Neural Network: Feature Selection + Dimensionality Reduction (PCA)	0.649	0.632	0.667	0.836
0	Log Regression All Features	0.655	0.947	0.500	0.881
4	Log Regression: Only Feature Selection	0.655	0.947	0.500	0.881
13	Neural Network: Feature Selection + PCA + Clustering (K Means)	0.667	0.641	0.694	0.843
2	Gradient Boosting All Features	0.689	0.840	0.583	0.881
6	GBM: Only Feature Selection	0.689	0.840	0.583	0.881
7	Random Forest: Only Feature Selection	0.689	0.840	0.583	0.881
8	Log Regression: Feature Selection + Dimensionality Reduction (PCA)	0.712	0.913	0.583	0.893
16	Weighted Ensemble (Best 4 Models)	0.724	0.955	0.583	0.899
12	Log Regression: Feature Selection + PCA + Clustering (K Means)	0.762	0.889	0.667	0.906

Results support the observations we have seen earlier. The “All features” models for each algorithm generally do poorest with the Random Forest All Features model the worst with an out of Sample test F1 Score of only 0.41. The best models on the other hand this time tend to be all models which had dimensionality reduction done either through Feature Selection + either Principal Components or Clustering or both. The one exception is the Gradient Boosting Model which seems to hold up with or without any dimensionality reduction and feature selection. Finally an Ensemble model was also created which is just the average prediction of the best model for each algorithm combined. Rationale for building an ensemble was that the correlations in predictions for each algorithm was low (between 0.65-0.80) so by combining multiple individual algorithms we could again do pseudo-“bagging” much akin to a Random Forest and so possibly achieve better generalization. This model performed second best. Precision Recall Curves for RF, Log Regression, and Neural Nets also support the observation that the best models tend to be those with lower dimensionality as their curves are all shifted to the right more.

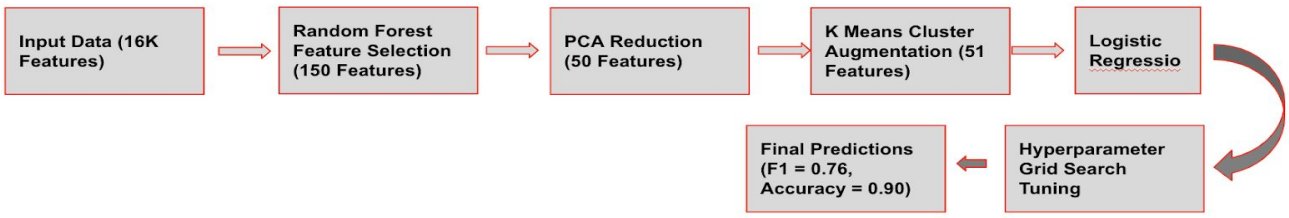






In fact the overall best performing model in our experiments is the Logistic Regression with everything applied to it and with key parameter values (after grid search) as penalty for regularization = L2 with C = 0.01. First, feature selection through random forest plots to choose the best 150 features, then Principle Components applied on this 150 dimensional feature subspace and finally K means Clustering labels augmented to the 50 principle components. This generates an F1 Score of 0.762 and Accuracy of 0.906 on test data. In this final model, the input data final is only 51 features wide (50 principle components + 1 K Means cluster label) and beats the All 16K features Logistic Regression by a huge margin of >=20% (and the worst performing model the all features Random Forest by >=70%).

Final Best Performing Logistic Regression Model Architecture



Conclusion

In this paper a high dimensional dataset was used to predict a given 'response' variable. Due to the extremely small number of training samples and huge number of dimensions, extreme overfitting from high variance was observed when different supervised learning algorithms (Neural Net, GBM, Random Forest and Logistic Regression) were trained. To overcome this "curse of dimensionality", first Feature Selection using Random Forest Importance plots was used to eliminate useless features, then dimensionality reduction through methods like PCA was used on the feature subspace to further reduce overfitting, followed lastly by feature augmentation of the principal components using clustering. The final models were greatly reduced in dimensionality and were then tested on hidden test data to see if performance generalized to unseen data. The best performing model on hidden test data was indeed the Logistic Regression with Feature Selection + PCA + K Means clustering augmentation applied iteratively. Results from this exercise therefore show that very high dimensional datasets can overcome the 'curse of dimensionality' when appropriate unsupervised learning methods for reducing dimensionality are used.

Areas of Improvement:

Given more time the following could all also have been tried to create more robust results

1) This analysis was seeded for reproducibility. Different seeds tend to change the ordering of the best performing model. Due to the extremely small number of training examples the impact of a seed can be huge in terms of performance and large variation in performance was noted depending on seed. However, generally speaking the All Features (16K) models almost always were the worst performers compared to their Dimensionally reduced counterparts. If more time was present we would want to repeat each experiment N times over different seeds and take the average results for a more robust and consistent result.
2) Second, more rigorous parameter tuning could have done which could have impacted some of the model results and ordering of model performance.
3) PCA and Clustering would have been better done on a different dataset as opposed to the train as explained earlier