A Model for Automated Feature Selection

Rojina Deuja
Department of Computer Science
University of Nebraska-Lincoln
Lincoln, USA
rdeuja2@huskers.unl.edu

Anup Adhikari
Department of Computer Science
University of Nebraska-Lincoln
Lincoln, USA
aadhikari2@huskers.unl.edu

***Abstract*—In this report, we investigate multiple techniques and purpose a model for dealing with the feature selection which have been the focus in areas of application for datasets with hundreds and thousands of variables. To this end we utilize a dataset which was prepared artificially for examining feature selection techniques. Our investigation clearly shows that feature selection in datasets where variables do not have significance can enhance the predictive ability of a machine learning algorithm.**

Keywords—feature selection, dimensionality,

I. Introduction

Feature selection is a very common problem when dealing with datasets where you can have overwhelming number of variables associated. This becomes more acute in cases where the numbers of features are over hundreds. It is further complicated if the features have no correlation or doesn't belong to some natural groupings. Furthermore, the complexity of various machine learning algorithms depends on the dimension of the dataset and data samples available. Hence the layman technique of using all the features would be computationally expensive and the algorithm would learn useless information from these features. Feature extraction algorithms could be one of the solutions but with application of such algorithm we lose some sense of interpretability which is often very important in many domains such as medical etc. Feature selection would not have that issue as we would be able to original features.

II. Problem Statement

While doing feature selection there are a lot of factors to consider. The domain knowledge can be a great insight but often the person handling the data may have no idea about it. The data itself maybe too complex to analyze even for the experts of that domain. Although it is possible to select the features using some prior knowledge and other manual techniques, it is not always feasible. Many times, when carrying out feature selection we might be biased with our assumptions while selecting a feature or there may be too many features in a dataset to be conveniently filtered out. The usual layman way of dropping the features based on correlation would be very tedious. Feature selectin is very important in many supervised learning problems for variety of reasons such as generalization performance, computational constrains, run time requirements, and interpretational issues by the problem itself.

In classification problems we are given *n* numbers of data points with *d* dimensions labeled *y* drawn i.i.d from a probability distribution *P (x, y).* We select a subset of features while preserving or improving the discriminative ability of a classifier. One of the ways to obtain such subset would to brute force through the powerset but one needs to consider the computational expense of any given algorithm. []

We derive our motivation for choosing this problem through the "**Occam's razor (or Ockham's razor)**" principle. It is a problem-solving principle from philosophy which says that if there exist two explanations for an occurrence, the one that needs the smallest number of assumptions is usually correct. i.e. The more assumptions that you must make, the higher the chances that it is an unlikely explanation.

III. Research Questions

The Madelon dataset was designed for the NIPS 2003 feature selection challenge. Challenges like this showcase the importance of feature selection in the Machine Learning paradigm. The complexity of this dataset (large features and non-linear data) puts forwards many research questions: How to approach feature selection for a very large feature space? What machine learning algorithms work best for high dimensional data? How to reduce the dimension space and tackle the 'Curse of dimensionality'? How to optimize feature selection to work well with non-linear dataset? and so on. In our research, we aim to answer these questions by exploring various conventional and non-conventional feature selection mechanisms and algorithms.
Our research will be centered around the Madelon dataset and we will try to achieve the highest accuracy with the numbers of features as close to the number of true features in the dataset (i.e. 5 features).

IV. Data Summary

The Madelon Data Set is obtained from the UCI Machine Learning Repository [Link]. It is an artificial dataset created by NIPS. This is a two -class classification problem with continuous input variables. It contains data points grouped in 32 clusters placed on the vertices of a five-dimensional hypercube and randomly labeled +1 or -1. The five dimensions constitute 5 informative features. 15 linear combinations of those features were added to form a ser of 20 (redundant) informative features. Based on those 20 features one must separate the examples into the 2 classes (corresponding to ‘1’ or ‘2’ numeric labels) Numbers of distractive features having no predictive power called ‘probes’ is 480.

There are a total of 2600 data samples. The data samples are equally divided into the two classes. Each class has a total of 1200 samples. Hence, the dataset is not skewed. The data samples are divided into train set and test. The training set includes 2080 data samples and the test set includes 520 data samples. The statistics of each variable can be summarized as below:

TABLE I SUMMARY OF WHITE WINE DATA

	V1	**V2**	**V3**	**V4**	**V5**	**V6**
mean	481.74	483.55	509.33	483.36	500.92	479.35
Std.	6.31	30.33	38.79	8.94	41.35	6.89
40.42	462.0	377.0	370.0	453.0	360.0	459.0
0.25	478.0	464.0	484.0	477.0	474.0	475.0
0.50	482.0	484.0	509.0	483.0	500.0	479.0
0.75	486.0	504.0	535.0	489.0	527.0	484.0
max	503.0	610.0	654.0	519.0	688.0	505.0

V. Data Pre-processing

The dataset was processed before using to build our model. The data collected may have various inconsistencies that need to be removed. Similarly, we need to perform an analysis of the correlations present in the dataset between various features. This can help us identify any redundant, insignificant and unnecessary features present in the dataset. Such analysis can help reduce the dimensions without loss of much information. It also helps in making the model much faster by significantly decreasing the training time and efficiency since many features can cause overfitting.

A. Data Cleaning

Data cleaning is performed in order trace any missing, inconsistent or incorrect data from the dataset. If the number of inconsistencies in the data is very high, it may affect the accuracy of our model. So before beginning with any other process, we carefully look at our dataset. The dataset does not have any missing values and NULL values. So the need to handle such inconsistencies were not seen. Hence, we concluded the data to be well-fitted for further processing without requiring significant data cleaning.

B. Standardization (Feature Scaling)

The input features in the dataset varies through various scale. If the model is built on unscaled data, variable while larger scale is overrepresented and it will affect the predictions more than variables with smaller scales. To avoid such variation, we carry out Standardization of the data. Even though the magnitudes of the values do not vary a lot in our dataset, we carry out Standardization since it further ensures consistency. We used the Gaussian standardization for scaling each input variable. The Gaussian standardization is given by the formula:

$$z = (x - \bar{x}) / \sigma$$

where,
z= z-score (number of standard deviations from the mean a data point)
x= data point
$\bar{x}$ = mean
σ= standard deviation

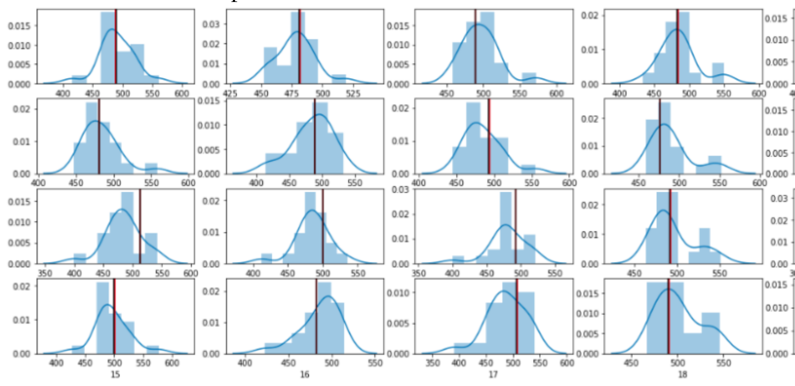
VI. Exploratory Data Analysis

We carry out various exploratory data analysis on our dataset to better understand the relations within the dataset. We look at the data samples count, class labels, class count, skewedness, distribution and the collinearity in our dataset to determine the best approach for further data processing.

A. *Data Distributions*

Before applying any machine learning or statistical methods, it is crucial to understand the underlying distribution of the data. The distribution of the data helps us clearly state our assumptions and base our models upon them. We visualize the distribution among the attributes by using histograms. Since the number of features is very high, we will plot the histograms by sampling 20 random features out of the 500 as shown in the fig. The observations made from the visualization are as follows:

1. Some of the features are skewed (i.e. the mean and the median are not the same)
2. Most of the features have a Normal distribution.
3. Some of the features have a Bimodal distribution.
4. Outliers are present in the data.



B. *Correlation Analysis*

Since correlation analysis on all 500 features is tedious, we generated a list showing the highly correlated with the target columns. Some features that are highly correlated with the target column are shown in Table II. Here, dropping features based on correlation wouldn't be wise for highly dimensional data as such a simple baseline approach might not be sufficient to remove all the undesired features. So, we move forward with carrying out a covariance analysis. In this method, we view the relationships present in between the features. We use Pearson correlation which is a measure of the linear correlation between two variables X and Y. Here, the two variable X and Y are two features selected and processed in a pair-wise fashion. The Pearson correlation gives us a value between +1 and –1, where 1 is total positive linear correlation, 0 is no linear correlation, and −1 is total negative linear correlation.

For our study, we use a value of 0.9 as our threshold correlation. Any feature pairs having correlations greater than the given threshold are considered to be highly correlated. From our analysis, we find that there are 18 features which have correlation value greater than the given threshold. We also used a heatmap to visualize these features as shown in fig. This leads us to the finding that our dataset has highly correlated features.

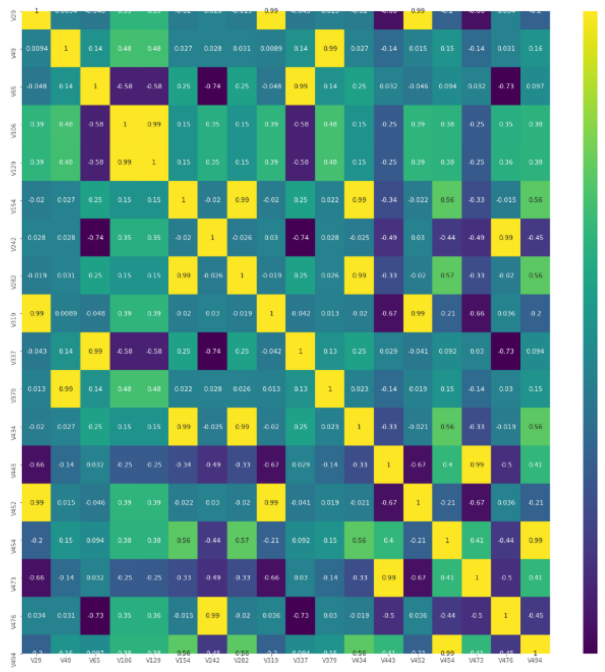


TABLE II LIST OF MOST CORRELATED FEATURES

Features	Absolute correlation with 'Class'
Class	1.000000
V337	0.159557
V65	0.158897
V339	0.129799
V443	0.107368

C. *Linearity Check*

Our dataset has a large number of dimensions. Such a dataset is mostly non-linear due to it's propert of high dimensionaity. Ie. A linear decision boundary cannot suffice to separate such data. With this assumption in mind, we picked random features from the dataset and used a scatter plot to observe the distribution of the data points. Since we have 500 features, we visualise 15 features and their data relationship. For each unique visualization, we see that the data is highly non-linear (ie. A linear decision boundary cannot be used to separate them). To further validate our observation, we use a purely linear classfier, Logistic Regression Classifier to make predictions on our dataset. The Logisitic Regression Classifier cannot perform well on the given non-linear dataset and gives us an accuracy of just 55.96%. The incapability of this classifier to create a decision boundary for our dataset establishes the assumption that our dataset is not linearly separable.

VII. Evaluation Metrics

For the purpose of evaluating our dataset, we will need to choose a classifier to be used in all our predictions. Since this model is primarily based on the goal of feature selection, the benchmark classifier will be used to assess the efficiency of our model. The benchmark model should be able to predict well on high dimensional, non-linear dataset and also be able to resist the vast noise present. We enumerate 4 classifiers for this purpose, use each one to predict on the raw dataset without feature selection and then pick the one that gives us the highest accuracy as the benchmark classifier. We have brief reviews of each classifier performance below:

1. Logistic Regression

The logistic regression is a linear classifier as such it draws a linear decision boundary to separate the datasets. So, for our non-linear dataset we expect it to perform poorly and the observed accuracy scores confirm this. Logistic Regression can classify non-linear data only when data is augmented by projecting the features on a high-dimensional space.

2. KNeighbors Classifier

The K Nearest Neighbors classifier from sklearn is a non-linear classifier. It can classify non-linear dataset by creating nonlinear decision boundary. In the K-NN model there is no learning, thus we only tune the hyperparameters to create the optimal model. With no hyperparameter tuning, the model can't be expected to perform well. Moreover, our dataset has 500 dimension and KNN suffers from the curse of dimensionality. The curse of dimensionality affects the K-NN model in four ways:

- In high-dimension all neighbors are far away (outliers)

- Distance calculation is expensive in high-dimension

- Number of required training data increases exponentially

- Large number of irrelevant features in high-dimensional data

3. Decision Tree

Decision Tree Classifier from sklearn predicts the value of a target variable by using data observations to extract simple rules about the data. Decision trees can be fairly robust to noises as it can entirely ignore irrelevant features and only make decisions based on features significant in the classification. In all subsets, the accuracy scores were higher than other models.

4. Support Vector Classifier

In the SVC function from sklearn we use the kernel trick to project the data into a higher dimension. Since the data is non-linear, we use the Radial Basis Function (RBF) kernel. It suffers from a similar issue as the KNN Classifier--all features are maintained in the subset of instances used as support vectors, so the features that are noisy decrease the accuracy.

Out of the above four classifiers, the Decision Tree Classifier performed the best with the highest training as well as testing accuracy. Hence, we will use Decision Tree Classifier as our benchmark model for all future predictions.

VIII. METHODS

For our feature selection model, we use a combination of methods to get the most influential feature set. The methods used as described below. Before starting on our feature selection model, we also use the sklearn.feature_selection_SelectKBest model as a baseline. We evaluate the performance of this library on the dataset with k= 20, since we know that there are 20 combinational useful features. The accuracies of the model to predict the final set of feature set are noted with the help of our benchmark model i.e. Decision Tree Classifier.

A. SelectKBest

The SelectKBest removes all but the k highest scoring features by scoring the features sing some function. In our model we use f_regression as the scoring function. f_regression is the linear model for testing the individual effort of each of many regressors. It calculates the correlation between each regressor and the target computed, then it is converted to an F score then to p-value.

We use k =20 which is the number of 20 features to be selected. The limitation of this is technique is the need to guess ‘k’ and its unknown which scoring function to use. Since it uses linear function to calculate the score and our data is highly non-linear the accuracy obtained is very low.

B. Stepwise Regression

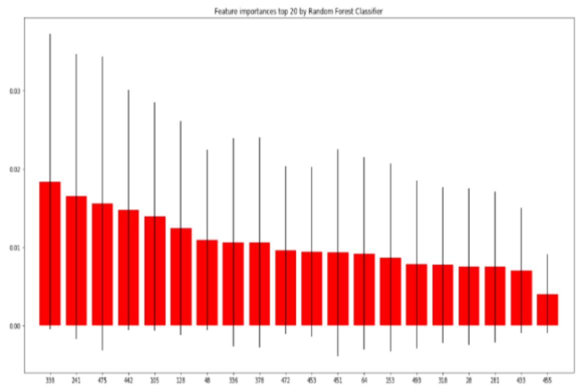
In this method, we take all the features as candidate features. They are checked over each iteration to observe their influence on the target variable. For the penalty, we have used the Lasso regularization (L1 regularization). The L1 regularize adds the absolute value of the magnitude of regularization coefficient as penalty to the loss function. It shrinks the values of irrelevant features to zero and thus we can simply remove them from the model. Thus, if the feature is found to be of a significance lower than our tolerance level, it will not be selected further for creating the model. The strength of the regularization is controlled by the hyperparameter C. The higher the value of C, the less the model is regularized.

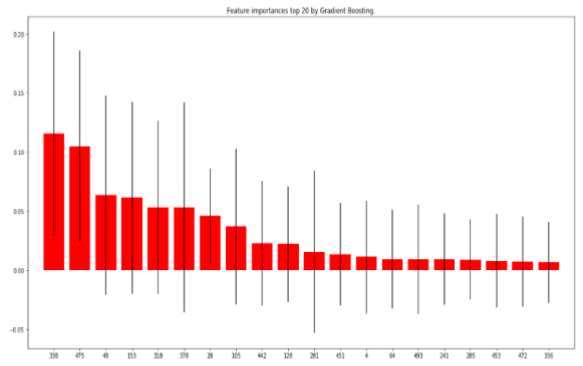
For the selection of the value of C, we carry out hyperparameter tuning to find out the most optimal parameter (maximum accuracy). The values of C used for the hyperparameter tuning are values starting from 0.0003 to 0.0005 with a step of 0.00001. Since we require a very strict regularization for the large number of features, we have used very small values of C. This gives us a list of 8 selected features.

C. Tree-based Selection

We use Random Forest Classifier for tree-based feature selection. Random forests consist of hundreds of decision trees, each of them built over a random extraction of the observations from the data samples and features which ensures that the trees are de-correlated and therefore less prone to overfitting. Each node of the tree is associated with some decision yes-no questions and divided into two nodes, each of them having different from the ones in the other node. Therefore, the importance of each feature is derived from how ‘pure’ each node is. Based on this we get the top most features which have high predictive capability.

In our model we used n estimators = 100 which means the number of trees in the random forest would be equal to 100. The more the tree the better the prediction but it is also computationally expensive. We also use random sate = 0, so that the randomness of the bootstrapping of the samples used while building trees and the sampling of the features to consider when looking for the best split at each node. The feature importance given by the Random forest classifier can be seen in the fig. below the Additionally, we also used gradient boosting for obtaining feature importance on the basis of their learning capability to predict the target variable. The n estimators which represent the number of boosting stages was set to 100 and random state was set to 0. The feature importance given by the Gradient Boosting can be seen in the fig. Below. Finally, we add both the metrics to see combined effect verdict of the both techniques and sort the highest 20 best features to perform further analysis.





D. Sequential Forward Selection

The Sequential Forward Selection (SFS) algorithm takes in all of the d-dimensions in the input set. It gives the output as 'k' unique features, where the value of 'k' must be specified beforehand and k < d. In the first step, it initializes the selected feature set as an empty set Ø so that k = 0. [https://sebastianraschka.com/pdf/software/mlxtend-latest.pdf] Then, the performance of the classifier (Accuracy) is evaluated with respect to each feature in the dataset. A feature X' is added to the feature subset if the feature maximizes the criterion function (Accuracy). Then, it moves on to the next feature. This process is repeated in a sequential manner, until the criteria for termination is satisfied (size of subset = k).

For our model, we set the value of k = 20 since we want to find the set of useful features in our dataset. For scoring, we have used 'accuracy' as the metric and we run this algorithm for 4 folds cross-validation. We use random state = 42 to ensure the reproducibility of our dataset used in training and testing the model. Similarly, for the classification we used our benchmark classifier i.e.. Decision Tree Classifier.

E. Exhaustive Forward Selection

The Exhaustive Forward Selection (EFS) samples and evaluates all possible feature combinations within a given range. It is essentially a wrapper method that carried out a brute-force evaluation on various combinations of features. After it goes through each possible combination of given features, it chooses the best subset as per the criterion function provided given any arbitrary classifier. Since it tries out each combination of inputs, it is computationally very expensive. So, we only use this approach at the very end, after we have listed the minimum number of selected features from one or more methods listed above.

In our implementation, we apply the EFS method on a subset of 9 features that are selected as a union from Tree-based and Sequential Forward Selection method. We set the number of max_features =5, so that the EFS gives us the list of 5 most useful features as an output. Similar to the other algorithms, we have used Decision Tree Classifier, accuracy as scoring and random state = 42.

IX. Results

The final feature selection model was built using a combination of two best performing methods: Random Forest Tree with Gradient Boosting and Sequential Forward Selection (SFS). The common features from these models were extracted which can be listed as following:

Combined_features = ['V49', 'V129', 'V282', 'V319', 'V339', 'V379', 'V452', 'V473', 'V476']

From this set of features, we further filter out to only 5 features. The final 5 features are selected using an Exhaustive Forward Selection (EFS) algorithm. The EFS gives us the set of five most useful features as below:

Selected_features = ['V129', 'V282', 'V319', 'V379', 'V46']

The final five features were used in the benchmark model i.e. DecisionTree Classifier to predict on the test set. An accuracy of 82.69% was achieved on the test set which is much higher than the accuracy from the unprocessed dataset ie.74.03%. Thus, with the help of various feature selection strategies, we were able to significantly increase the accuracy of the prediction model by reducing the feature set to a minimum. The performance summary of the final classification model is as follows:

TABLE VI PERFORMANCE MEASURES

Measure	Value
Accuracy	0.8269
Precision	0.8036
Recall	0.8599
F1 Score	0.8308
Confusion Matrix	[221 36] [54 209]

If we evaluate the performance measures for different k values, we see that the model performs well, given the F1 score of around 0.84. During the selection of optimal parameters as shown in table V, we note that the performance of our model peaks at k=11. When we use a value of k=1, the model tends to overfit the data and gives us a training accuracy of 1.0. Inorder to avoid this phenomenon, we utilized the inverse distance function.

X. Conclusion and Future Work

We created a model for feature selection from a dataset with 500 features. The ratio of true features (i.e. 5) to the total features (i.e. 500) was very low. Picking out the true features from all this noise was a challenging task in itself. Another challenge was to design a suitable model for the non-linear characteristic of the chosen dataset. We were able to cover prominent territory in the course of working on this project and gained useful insight in the process of optimizing feature selection.

Although various approaches have been developed to uncover the true features in the Madelon dataset, we believe that we have put forward a strong effort in this domain. Moving forward, we can try additional techniques like Univariate Feature Selection, MLP based feature selection, Kernel methods etc. We could also try out various sklearn.feature_selection libraries like SelectFromModel, RFE, RFECV to do the heavy lifting for us.

References

[1] Weston, J., Mukherjee, S., Chapelle, O., Pontil, M., Poggio, T., & Vapnik, V. (2001). Feature selection for SVMs. In *Advances in neural information processing systems* (pp. 668-674)