Nick Stephenson
Yulong Liang
Zhimeng Pan

CS 6140 – Data Mining
Intermediate Report
Due Monday, 3/26

For data mining and machine learning tasks, data preprocessing is the most important and time-consuming process before the deployment of algorithms. It is a classical topic that how to select relevant features to fit to the algorithm. The goal for the project is to compare between the modern **feature selection techniques** in data preprocessing.

## Current Progress

Currently, our task is using the diabetes data we found on the UCI repository to predict the probability of readmission by a patient to the hospital within 30 days. As far as progress towards this task, we started by cleaning the data. Initially the data had 50 variables (features). To reduce the number of variables that we will be using, we started by removing features that will have no effect on our desired outcome, including the "encounter id" and "patient number" that essentially act as indices. As we looked at each variable, we also noticed some missing data. The variables that had missing data were categorical, so we imputed the missing values with the mode of the non-missing values. This can introduce some bias into the data, but it is still better than removing the observations with missing information. Our dataset is quite large (101,766 observations), so doing a complete-case analysis was considered. However, we thought that the best model was to preserve all of the observations, while reducing the number of features as much as possible.

Along with the removal of unnecessary variables and imputing missing data, most of the variables needed a mapping of some sort to assign each level of the categorical information to a number. Some mappings were already provided in the folder where the dataset was found.

It should be noted the all these categorical should actually be convert to "on-hot encoding", i.e., create a new binary attribute for every category this variable can take. But for preliminary test we just mapped categories to integers arbitrarily. To keep all variables in the same scale, we performed a min-max scaling(normalization) for every variable. Then the cleaned and transformed data set was split into a training set and a test set on a ratio of 9:1 chosen arbitrarily.

In order to evaluate the choice of features, we need some classifiers as "backend" algorithms. We initially use all 39 features as input. To our surprise, none of the standard algorithms for classification (logistic regression, decision tree, adaboost) we tested looked promising or workable on this data set, i.e., all these classifiers didn't work better than a dump classifier that always say no for every input. Specifically, all these classifiers achieved a test set accuracy about 88.9%, which seemed good. But note that in this data set, the portion of positive result(yes) is exactly 11.1%, which means a dummy classifier could get accuracy of 88.9%. In order to properly measure the performance of classifier, we instead resort to measure the recall rate, which is more suitable in this setting, as we main goal is to identify all patients that are highly possible to be readmitted, other than focusing on those are not. Then all above tested algorithms got similar low recall rate about 0.015. This may because that there are too many irrelevant features so that the true related features are buried in huge noise.

Then our planning forks to two path. One is to continue looking for suitable classifier for this data set and problem setting. The other is to perform feature selection algorithms that are not depended on a learning algorithm and then use these selected features to train model.

**Future Plan**

We want to experiment with different feature selection techniques to start seeing which features would be significant. We will try to improve the prediction accuracy and recall rate as we discover all the possible feature selection techniques.

- **Embedded Methods**
  Some techniques like Regularized Logistic Regression, Regularized Linear Regression (Lasso, Ridge, ElasticNet), and Decision Tree with max-depth have their own embedded approaches to implement feature selection while fitting the model to the data. We are not going to implement them but treat them as references.
- **Filter Methods**
  Filter Methods refer to approaches which use statistical metrics to evaluate each feature/feature combinations. Common measures include the mutual information, Pearson correlation coefficient, inter/intra class distance or the scores of significance tests. These approaches not only can reflect the significance of a feature but easy to compute as well. Those approaches are supported by most of the data processing and statistical libraries and are widely used by data scientists, we are not going to implement them but treat them as references.
- **Wrapper Methods**

Wrapper Methods are less computationally efficient since they recursively run the data mining/machine learning algorithms with different feature sets, especially for much more complicated models like Kernel SVM, and Artificial Neural Networks. However, they produce a feature set which is tuned to a specific type of predictive model.
There are three most popular methods – Forward Feature Selection, Backward Feature Elimination and Recursive Feature elimination. Forward Feature Selection starts with an empty set and iteratively add the feature which best improves the performance. Backward Feature Elimination is the reverse version of Forward Feature Selection which starts with a set with all the features. Recursive Feature elimination is a greedy algorithm which aims to find the best performing feature subset. We are going to implement at least one the them.

- **Feature Clustering**
  Clustering is always being used to group data points into different clusters. However, we can also treat each feature as a data point in a feature space. We want to group closed feature subspaces together to perform feature selection.
  For assignment-based clustering (k-means, k-medians, k-mediods), users have to provide the number of clusters, which is not applicable in feature selection.
  For density-based clustering (DBSCAN), outliers will be ignored by the algorithm. For feature selection, an outlier should be kept because it represents a feature that cannot be represented by other features.
  For hierarchical clustering, it follows the strategy that points are grouped so that within each group there are no points that are very far from each other. Moreover, it will not let the user to specify the number of clusters at first. These properties fit the feature selection objective very well. The famous machine learning toolkit *sciki-learn* has also added a method `sklearn.cluster.FeatureAgglomeration`, which use the idea of hierarchical clustering. Thus, we are going to implement hierarchical clustering for feature selection.

With the help with those techniques, hopefully we can improve the prediction accuracy with full feature space.