Domain and Data

Domain: Our firm is bidding on a large project that involves working with many features. Since the usual feature selection methods will not aid us in this case, we must come up with a better way to select salient features.

Data: Our dataset is the MADELON dataset. It is a synthetic, artificial dataset, with 500 features and one target variable, labeled either +1 or -1. Of the 500 features, 5 are actually informative, 15 are linear combinations of the 5 informative features, and the rest (480) are basically noise (i.e. distractors).

Problem Statement

We must come up with a solution to select relevant features from a set of numerous features, many of which are random noise/distractors.

Solution Statement

The solution to our problem will be to build our own machine learning pipeline. This pipeline will load the data, split the data into target and feature matrix, split the data into training and test sets, transform the data, and finally, model our transformed data.

Metric

We will use 2 metrics. One will be the number of salient features we can identify. The other will be our accuracy, which is our test score.

Benchmark

For the number of salient features, we have been instructed that hitting 30 or less features should be considered a win.

For our accuracy, we have set up a benchmark accuracy score, using a naive Logistic Regression, in our "Step 1 - Benchmarking" Jupyter Notebook. We will present those results shortly in our section entitled "Results: Step 1 - Benchmarking"

Results: Step 1 - Benchmarking

In our first pipeline, our goal was to set a starting benchmark from which to compare the relative successes or failures of our subsequent pipelines. Our test score from our naive Logistic Regression, which utilized a high C value of

1000000, was approximately 57%. Our C value was intentionally set high for the purposes of minimizing regularization (i.e. setting a high C value, such as 1000000, will make our penalty very small, since the penalty is, in fact, 1/C).

Results: Step 2 - Identify Salient Features Using L1 (Lasso) Penalty

In our second pipeline, we decided to use Lasso regularization with Logistic Regression. The goal in using the L1 penalty was to reduce the number of features by driving the irrelevant feature's coefficients to zero. Unfortunately, out of the 500 features, we were only successful in eliminating 37 features, as we were left with 463 features whose coefficients were not driven to zero. It is important to note that we did not modify any other parameters of the Logistic Regression model (e.g. the C value). Finding optimal parameters, such as the C value, would be accomplished in our Step 3 pipeline.

Results: Step 3 - Build Model

In our third pipeline, we used a different transformer, SelectKBest, than our previous pipelines, which used StandardScaler. With this transformer, we ran a Logistic Regression and a KNeighborsClassifier. Both performed better than our benchmark. But what is more interesting is that our KNeighborsClassifier model posted a significantly higher test score than our Logistic Regression model, with the former scoring 84% versus the latter's 62%.

As for feature selection, our Logistic Regression with SelectKBest did a much better job than our Logistic Regression with Lasso (from our Step 2) in identifying relevant features: the SelectKBest/Logistic Regression combination had 4 features remaining, versus the Lasso's 463 features remaining. Of course, as mentioned in our Step 3 notebook's results section, the accuracy score was still only at 62%, so it seemed that our model erroneously eliminated informative features as well.

Next Steps - Recommendations

Though we have invested a lot of time into building our machine learning pipelines, there are improvements that can be made, areas that we haven't addressed. For example, our pipelines are designed to select relevant features by eliminating less relevant features, but what about the case when feature creation is necessary? There may be cases when we need to create features from our data, but our pipelines do not address this issue currently.