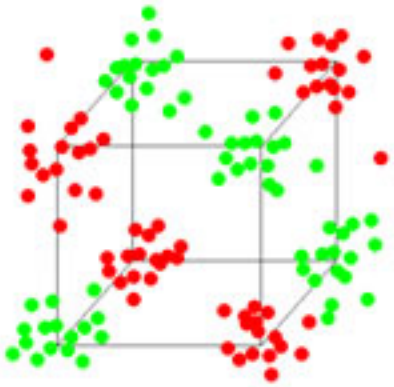
Exploring the Madelon Data Set

Using Logistic Regression and K-Nearest Neighbors

Alex Cheng



Description

The Madelon data set is from the UCI Machine Learning Repository. The data is completely artificial, randomly labeled +1 or -1 with 500 features. Of these 500 features, 20 of them are "real" features and the remaining 480 are "probes" which add noise to the data. Of the 20 real features, there are 5 informative features and the remaining 15 are linear combinations of the 5 informative features to add additional noise. The order of the features and patterns are randomized.

Problem Statement

Predicting a synthetic data set such as the UCI's Madelon data set can be difficult because the features are anonymous. The goal of the project is to use statistical models to perform both feature extraction and classification.

Solution Statement

To predict the labels (1 or -1) of this data set, four models will be built. First, a naive logistic regression (no regularization penalty) will be made as the benchmark for our classification score. Second, a logistic regression model with a L1 normalization penalty to regularize the data set's features. Third, a logistic regression model whose parameters are tuned through a Grid Search and Cross-Validation. Finally, a K-Nearest Neighbor Classifier with parameters tuned through Grid Search will be used in case the data set is non-linear.

Metric

I will be measuring "accuracy", based on the percentage of the model label the data points correctly. I will look at a confusion matrix to identify the false positives and the Area Under Curve (AUC). In addition, KNN's accuracy can be calculated by the standard Minkowski metric. However, the results of the KNN will be comparable to the logistic regression through AUC and confusion matrix analysis.

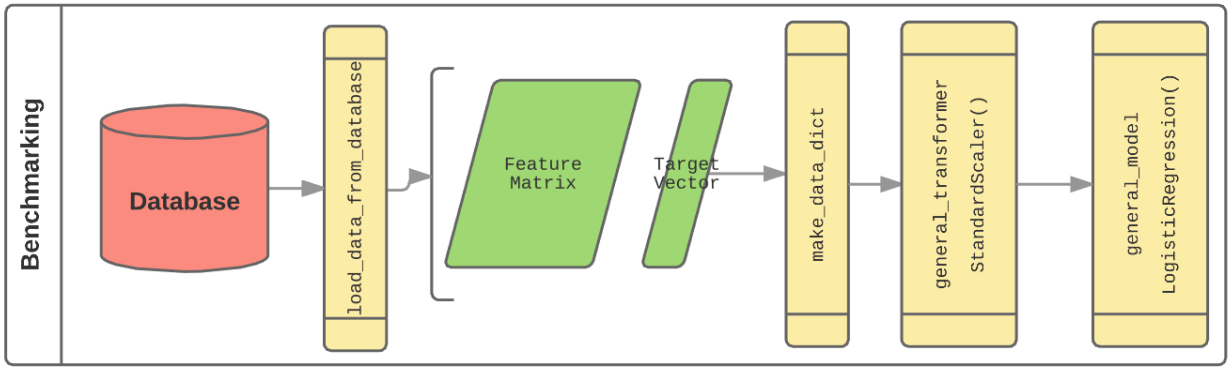
Benchmark

The benchmark will be a naive Logistic Regression (with no regularization penalty) for our classification score.

Methods

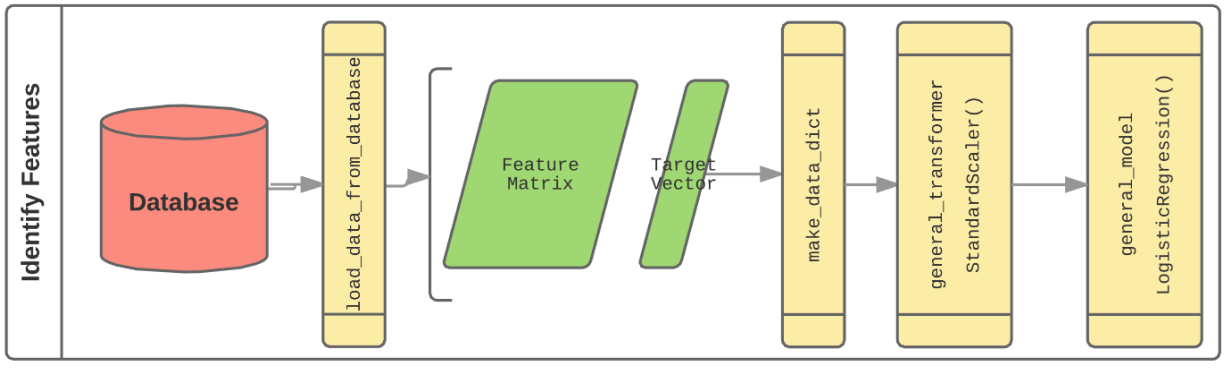
Building the Benchmark / Logistic Regression Model

The benchmark will be a naive Logistic Regression (with no regularization penalty) for our classification score.



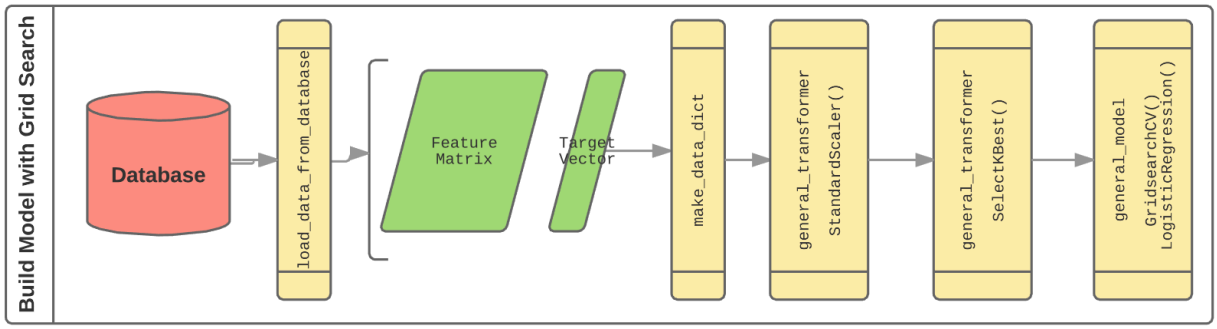
Feature Selection using Logistic Regression and SelectKBest

This schema uses logistic regression to select import features, similar to using the L1 norm logistic regression model. Rather than predicting on the test set, a table of coefficients for each feature is obtained, with larger coefficients being more meaningful than smaller coefficients.



Grid Search with Logistic Regression and KNN

The schema below for building this model also uses SelectKBest, but the feature matrix outputted from SelectKBest goes into the GridSearchCV. Using the GridSearchCV allows me to tune my model's parameters for the logistic regression to try different penalties and regularization features.



Results

Feature Selection from L1 Norm Regularization and SelectKBest

SelectKBest (above) found these top 10 features that scored relatively high.

	Selected Features	Coefficients
1	feat_048	0.5497
2	feat_493	0.3923
3	feat_338	0.3395
4	feat_248	0.2687
5	feat_053	0.2641
6	feat_384	0.2260
7	feat_403	0.2234
8	feat_431	0.2130
9	feat_477	0.1944
10	feat_073	0.1882

Logistic Regression and KNN Classification

Scoring by each Model

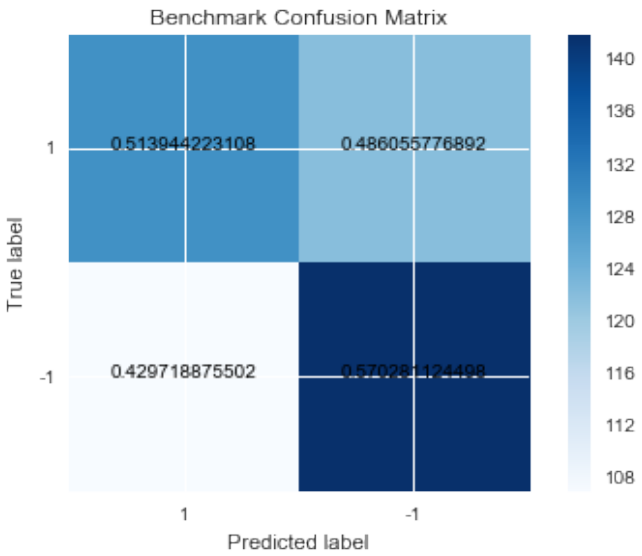
	Best Parameters	Train Score	Test Score	AUC
Benchmark Logistic Regression	penalty=l2, C=1.0, solver=liblinear	0.7913	0.5420	0.5421
Logistic Regression with L1 Penalty	penalty=l1, C=1.0, solver=liblinear	0.7933	0.5480	0.5483
Grid Search Logistic Regression	penalty=l1, C=1.718, solver=liblinear	0.622	0.608	0.6071
Grid Search KNN	n_neighbors=31, weights=distance	1.0	0.67	0.6709

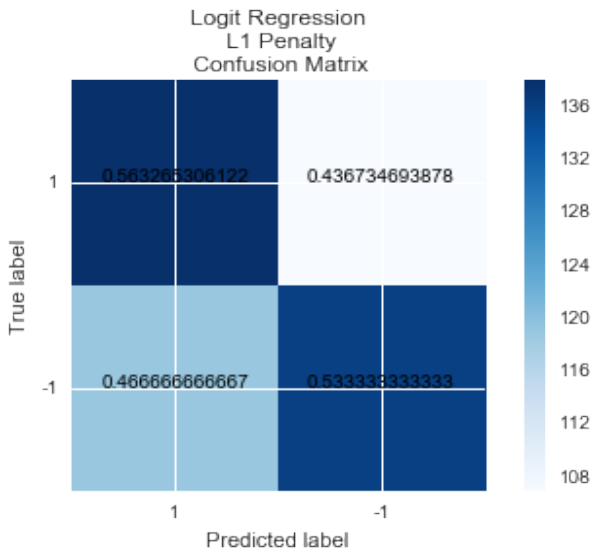
The table shows how the models performed, best parameters that were run, their training score, test score, and area under the ROC curve. The benchmark's train score was fairly high relative to its test score, indicating that the model is overfit, same with the Logistic Regression with the L1 penalty. When grid searching over the logistic regression parameters, the grid search only fared a little bit better, indicating that the features in the Madelon data set are very nonlinear, and logistic regression may be limited in some capacity during this analysis.

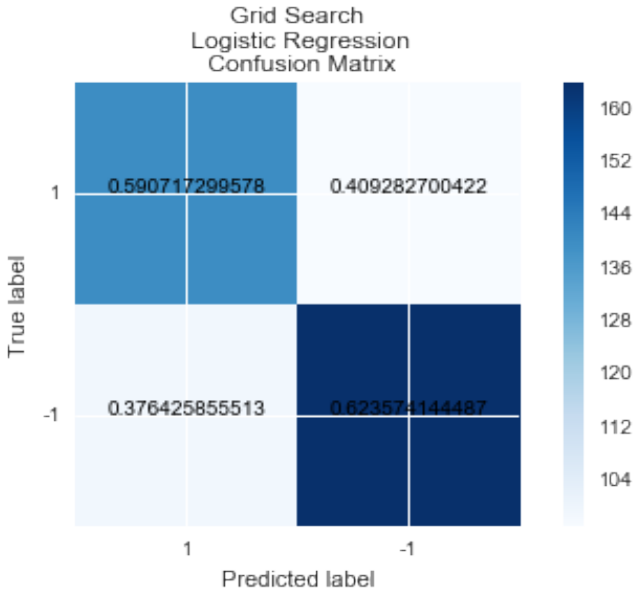
When using the SelectKBest to select 10 best features and then grid searching over weights and how many neighbors to make the classification, KNN had a perfect training score 1.0 and a test score of 0.67. While the perfect training score should cause some suspicion with how overfit the data is, the test score is also relatively higher as well.

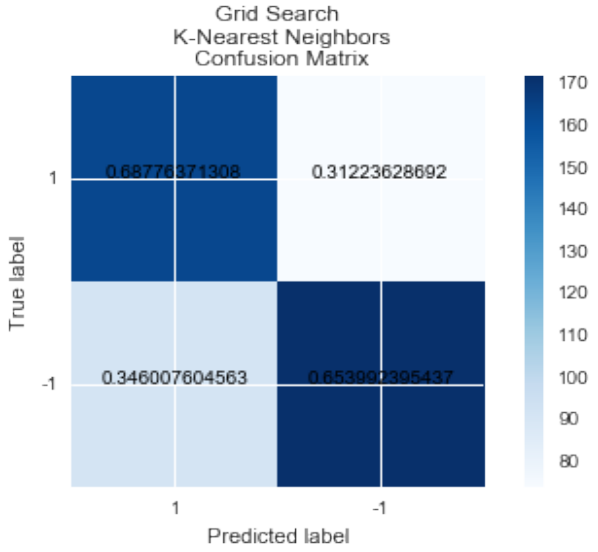
Looking at the confusion matrices (below), the color bar represents the true number of samples in each part of the confusion matrix whereas the normalized True Positive, True Negative, False Positive, and False Negative are shown in the middle of each square. What's most noticeable is how KNN has the highest True Positive and True Negative Rate.

Confusion Matrices









Conclusion

Given the results, Logistic Regression had some difficulty predicting labels in the Madelon data set. SelectKBest, when selecting 10 features, did a good job in finding the most important features. Furthermore, KNN did the best in predicting the labels, with a True Positive Rate of 0.688 and True Negative Rate of 0.654. In the future, using Recursive Feature Elimination may also help minimize features while maintaining the high prediction score, whereas using a Random Forest Classifier may help as well.