Project 3 - Report

November 13, 2017

0.1 Problem Statement

Predicting the classes of a synthetic data set is challenging due to lack of feature context. The goal of this endeavor was to use unsupervised and supervised approaches to identify important features in the data set and predict a binary class.

0.2 The Data

UCI's Madelon: This data set was provide in 5 separate chunks. There was a data and labels pairing for both 'training' and 'validation' data. The fifth set was a 'test' set with no labels. I combined the training and validation data so I could conduct my own cross validation. I did not use the test data. The resulting data set:

2000 rows, 501 columns

It is provided that this data set has 20 important features, and 480 noisey features. The classes (1 or -1) exist in 16 clusters per class in a hypercube, clustering at different corners of said hypercube.

Josh Cook Data: This data set was stored on a SQL server. Unlike the Madelon, all the data was in one database.

200,000 rows, 1,001 columns

It is provided that this data set has some number of important features, where the rest are noisey features. The classes (1 or 0) exist in 16 clusters per class in a hypercube, clustering at different corners of said hypercube.

0.3 Overall Approach

I first conducted my analysis on a subset of each data set. I took a 10% sample of UCI's Madelon data and a 3.3% sample of the Josh Cook data. I then took my feature selection and modeling approaches and attempted to apply them to each full data set. Subsequently, the report is split into four sections: UCI's Madelon Subset, UCI's Madelon Full Set, Josh Cook Data Subset, and Josh Cook Data Full Set.

The following steps were taken for each of those four sections:

Benchmarking

In order to establish a baseline to measure the models against, I conducted 4 naive tests: Logistic Regression, Decision Tree Classifier, K Nearest Neighbor, and Support Vector Classifier.

When applicable, I kept regularization strength low and used predominately default hyperparameters. No feature selection, preprocessing, or hyperparameter tuning was used in this step.

Note: I did not conduct a separate benchmarking analysis for a subset of the UCI Madelon data. The report below will show identical results for UCI Subset and UCI Full Set only on the benchmarking section.

Feature Selection

Given what I know about the data and the classes, I first performed unsupervised approaches to try and determine important features. Since I know that there are only 20 important features out of 500, I thought I could check to see if there were correlations across features. By using my `.corr()` method, I was able to discern 20 features with significant correlation to at least 1 other feature. I've provided the section of code used to do this:

```
predictors_corr = predictors.corr() hi_corrs = predictors_corr.abs()
> .5 hi_count = predictors_corr[hi_corrs].count() > 1 top_corrs =
list(predictors_corr[hi_count].index)
```

Essentially, this is measuring the correlation of each feature with every other feature in the data set (including itself). The code above is checking to look for correlations greater than .5. If a feature is correlated with more than 1 feature at that threshold (because every feature will have a correlation of 1.0 with itself), it gets flagged as being a top correlated feature. This resulted in 20 important features in both UCI's Madelon data set and the Josh Cook data set.

I also applied several supervised feature selection approaches, but given the dimensionality of the data, they did not provide consistent results across them. I decided to go with the 20 features achieved in the correlation analysis.

Modeling

For the modeling, I used the same four models that were used in the benchmarking. However, this time around I included only my 20 important features. Additionally, I applied preprocessing and hyperparameter tuning to the model. Each section in the extended report lists those steps in building my models.

What I discovered was that **K Nearest Neighbors** best predicted my classes. Given that we know the data exists in a multi-dimensional hypercube (as explained in the data section above), it makes sense that this would be the strongest approach. The reason is that K Nearest Neighbors assigns class based on distance between data points.

On the full UCI Madelon data, I was able to predict at a rate of 89% accuracy the classes for each record. On the full Josh Cook data, I was able to predict at a rate of 76% accuracy.

0.4 Next Steps

This section might only make sense to someone with data science expertise. To improve prediction accuracy, there are two things that I'd do next. 1. Stratify y in my train test split in order to ensure class balance: This makes sure that when I build my training and test data sets, I help to mitigate the risk of class imbalance.

2. Further tune hyperparameter in my K Nearest Neighbor: I want to test a larger range of `n_neighbors`.

0.5 Extended Report Overview

Below I've included details on the steps taken in each of the four sections I mentioned above. Those four sections are organized below in the following order:

1. UCI's Madelon Subset

2. UCI's Madelon Full Set
3. Josh Cook's Subset
4. Josh Cook's Full Set

In each of those four sections, you'll see details on the benchmarking, feature selection, and modeling steps taken in the Jupyter notebooks that pertain to each section and stored in this repository.

---

1 Extended Report

1.1 UCI's Madelon Data Set - Subset

1.1.1 Benchmarking

Description of Approach:
Built naive models to get a prediction benchmark for our classification problem. The models used are:

1. Logistic Regression
2. Decision Tree Classifier
3. K Nearest Neighbors
4. Support Vector Classifier

Decision Tree Classifier performed best with ROC AUC Score of .754 and Log Loss of 8.502.

1.1.2 Benchmarking Scores

Logistic Regression:
ROC AUC Score: 0.520
Log Loss: 16.605
Decision Tree Classifier:
ROC AUC Score: 0.754
Log Loss: 8.502
K Nearest Neighbors:
ROC AUC Score: 0.696
Log Loss: 10.495
Support Vector Classifier:
ROC AUC Score: 0.500
Log Loss: 17.469

1.1.3 Feature Selection

*** Description of Approach: ***
In this approach, I conduct several methods of feature selection. These methods include both supervised and unsupervised approaches.

Contents: 1. Data Import 2. Unsupervised Approach - Looked at correlation (`.corr()`) between features and returned those features that are highly correlated (> 0.5) with at least one other feature. - The result was 20 features: - 28, 48, 64, 105, 128, 153, 241, 281, 318, 336, 338, 378, 433, 442, 451, 453, 455, 472, 475, 493 - Visualizations of those 20 features 3. Supervised Approaches - Select from Model w/ Logistic Regression (L2 penalty) - Select from Model w/ Logistic Regression (L1 penalty) - SelectKBest w/ k = 20 - RFE w/ RandomForestClassifier as estimator

These approaches all gave me various feature importances and coefficients with little consistency across them.

Results: I decided to use the 20 features gained from the correlation approach. Given that I know the madelon data has 20 important (or redundant) features and 480 noise, I felt comfortable with the approach that produced exactly 20 features.

1.1.4 Modeling

Description of Approach:

In alignment with the benchmarking approach (for this workflow, the benchmarking was done on the full data set and not the sample used for modeling), I modeled the sample on these four models: Logistic Regression, Decision Tree, K Nearest Neighbors, and Support Vector Classifier.

The full steps of modeling: 1. Train Test Split 2. Min Max Scaler 3. Deskewing (Boxcox) 4. PCA (5 components) 5. Standard Scaler 6. Model

Steps 5 and 6 were built into a pipeline and gridsearched on to tune hyperparameters.

The best performing model was K Nearest Neighbors with an ROC AUC Score of .740 and Log Loss of 8.635.

1.1.5 Modeling Scores

Logistic Regression:
ROC AUC Score: 0.501
Log Loss: 16.605
Decision Tree:
ROC AUC Score: 0.618
Log Loss: 12.620
K Nearest Neighbors:
ROC AUC Score: 0.740
Log Loss: 8.635
Support Vector Classifier:
ROC AUC Score: 0.458
Log Loss: 17.934

1.2 UCI's Madelon Data Set - Full Set

1.2.1 Benchmarking

Description of Approach:

Built naive models to get a prediction benchmark for our classification problem. The models used are:

1. Logistic Regression
2. Decision Tree Classifier

3. K Nearest Neighbors
4. Support Vector Classifier

Decision Tree Classifier performed best with ROC AUC Score of .754 and Log Loss of 8.502.

1.2.2 Benchmarking Scores

Logistic Regression:
ROC AUC Score: 0.520
Log Loss: 16.605
Decision Tree Classifier:
ROC AUC Score: 0.754
Log Loss: 8.502
K Nearest Neighbors:
ROC AUC Score: 0.696
Log Loss: 10.495
Support Vector Classifier:
ROC AUC Score: 0.500
Log Loss: 17.469

1.2.3 Feature Selection

*** Description of Approach: ***
In this approach, I conduct several methods of feature selection. These methods include both supervised and unsupervised approaches. I've run these methods on the full UCI Madelon data set.

Contents: 1. Data Import 2. Unsupervised Approach - Looked at correlation (`.corr()`) between features and returned those features that are highly correlated (> 0.5) with at least one other feature. - The result was 20 features: - 28, 48, 64, 105, 128, 153, 241, 281, 318, 336, 338, 378, 433, 442, 451, 453, 455, 472, 475, 493 - Visualizations of those 20 features 3. Supervised Approaches - Select from Model w/ Logistic Regression (L2 penalty) - Select from Model w/ Logistic Regression (L1 penalty) - SelectKBest w/ k = 20 - RFE w/ RandomForestClassifier as estimator - These approaches all gave me various feature importances and coefficients with little consistency across them.

Results: I decided to use the 20 features gained from the correlation approach. Given that I know the madelon data has 20 important (or redundant) features and 480 noise, I felt comfortable with the approach that produced exactly 20 features.

1.2.4 Modeling

Description of approach:
In alignment with the benchmarking approach, I modeled the sample on these four models: Logistic Regression, Decision Tree, K Nearest Neighbors, and Support Vector Classifier.

The full steps of modeling: 1. Train Test Split 2. Min Max Scaler 3. Deskewing (Boxcox) 4. PCA (5 components) 5. Standard Scaler 6. Model

Steps 5 and 6 were built into a pipeline and gridsearched on to tune hyperparameters.

The best performing model was K Nearest Neighbors with a ROC AUC Score of .889 and Log Loss of 3.852.

1.2.5 Modeling Scores

Logistic Regression:
ROC AUC Score: 0.619
Log Loss: 13.151
Decision Tree:
ROC AUC Score: 0.773
Log Loss: 7.838
K Nearest Neighbors:
ROC AUC Score: 0.889
Log Loss: 3.852
Support Vector Classifier:
ROC AUC Score: 0.800
Log Loss: 6.907

1.3 Josh Cook Data Set - Subset

1.3.1 Benchmarking

Description of Approach:
Built naive models to get a prediction benchmark for our classification problem. The models used are:

1. Logistic Regression
2. Decision Tree Classifier
3. K Nearest Neighbors
4. Support Vector Classifier

The best performing model was a close race between Decision Tree Classifier and Support Vector Classifier.
Decision Tree Classifier had an ROC AUC Score of .629 and Log Loss of 12.821.
Support Vector Classifier had an ROC AUC Score of .624 and Log Loss of 12.978.

1.3.2 Benchmarking Scores

Logistic Regression:
ROC AUC Score: 0.539
Log Loss: 15.909
Decision Tree Classifier:
ROC AUC Score: 0.629
Log Loss: 12.821
K Nearest Neighbors:
ROC AUC Score: 0.600
Log Loss: 13.816
Support Vector Classifier:
ROC AUC Score: 0.624
Log Loss: 12.978

1.3.3 Feature Selection

*** Description of Notebook: ***

In this approach, I conduct several methods of feature selection. These methods include both supervised and unsupervised approaches.

Contents: 1. Data Import 2. Unsupervised Approach - Looked at correlation (.corr()) between features and returned those features that are highly correlated (> 0.5) with at least one other feature. - The result was 20 features: - 'feat_257', 'feat_269', 'feat_308', 'feat_315', 'feat_336', 'feat_341', 'feat_395', 'feat_504', 'feat_526', 'feat_639', 'feat_681', 'feat_701', 'feat_724', 'feat_736', 'feat_769', 'feat_808', 'feat_829', 'feat_867', 'feat_920', 'feat_956' - Visualizations of those 20 features 3. Supervised Approaches - Select from Model w/ Logistic Regression (L2 penalty) - Select from Model w/ Logistic Regression (L1 penalty) - SelectKBest w/ k = 20 - RFE w/ RandomForestClassifier as estimator - These approaches all gave me various feature importances and coefficients with little consistency across them.

Results: I decided to use the 20 features gained from the correlation approach. Given that I know the madelon data has 20 important (or redundant) features and 480 noise, I felt comfortable with the approach that produced exactly 20 features.

1.3.4 Modeling

Description of notebook:

In alignment with the benchmarking approach, I modeled the sample on these four models: Logistic Regression, Decision Tree, K Nearest Neighbors, and Support Vector Classifier.

The full steps of modeling: 1. Train Test Split 2. Min Max Scaler 3. Deskewing (Boxcox) 4. PCA (5 components) 5. Standard Scaler 6. Model

Steps 5 and 6 were built into a pipeline and gridsearched on to tune hyperparameters.

The best performing model was K Nearest Neighbors with an ROC AUC Score of .744 and Log Loss of 8.844.

1.3.5 Modeling Results

Logistic Regression:

ROC AUC Score: 0.588
Log Loss: 14.208
Decision Tree:
ROC AUC Score: 0.658
Log Loss: 11.800
K Nearest Neighbors:
ROC AUC Score: 0.744
Log Loss: 8.844
Support Vector Classifier:
ROC AUC Score: 0.711
Log Loss: 9.969

1.4 Josh Cook Data Set - Full Set

1.4.1 Benchmarking

Description of Approach:
Built naive models to get a prediction benchmark for our classification problem. The models used are:

1. Logistic Regression
2. Decision Tree Classifier
3. K Nearest Neighbors
4. Support Vector Classifier

The best performing model was a close race between Decision Tree Classifier and Support Vector Classifier.
Decision Tree Classifier had an ROC AUC Score of .644 and Log Loss of 12.298.
Support Vector Classifier had an ROC AUC Score of .624 and Log Loss of 12.978.

1.4.2 Benchmarking Scores

Logistic Regression:
ROC AUC Score: 0.539
Log Loss: 15.909
Decision Tree Classifier:
ROC AUC Score: 0.644
Log Loss: 12.298
K Nearest Neighbors:
ROC AUC Score: 0.600
Log Loss: 13.816
Support Vector Classifier:
ROC AUC Score: 0.624
Log Loss: 12.978

1.4.3 Feature Selection

*** Description of approach: ***
In this approach, I conduct several methods of feature selection. These methods include both supervised and unsupervised approaches. I've used only a sample of the data to conduct feature selection, as the dataset is too large to run this methods on the full 200,000 rows.
Contents: 1. Data Import 2. Unsupervised Approach - Looked at correlation (.corr()) between features and returned those features that are highly correlated (> 0.5) with at least one other feature. - The result was 20 features: - 'feat_257', 'feat_269', 'feat_308', 'feat_315', 'feat_336', 'feat_341', 'feat_395', 'feat_504', 'feat_526', 'feat_639', 'feat_681', 'feat_701', 'feat_724', 'feat_736', 'feat_769', 'feat_808', 'feat_829', 'feat_867', 'feat_920', 'feat_956' - Visualizations of those 20 features 3. Supervised Approaches - Select from Model w/ Logistic Regression (L2 penalty) - Select from Model w/ Logistic Regression (L1 penalty) - SelectKBest w/ k = 20 - RFE w/ RandomForestClassifier as estimator - These approaches all gave me various feature importances and coefficients with little consistency across them.

Results: I decided to use the 20 features gained from the correlation approach. Given that I know the madelon data has 20 important (or redundant) features and 480 noise, I felt comfortable with the approach that produced exactly 20 features.

1.4.4 Modeling

Description of notebook:

I ran my best model fit and scored from the sample of the data from Josh Cook's database on the full dataset.

The full steps of modeling: 1. Train Test Split 2. Min Max Scaler 3. Deskewing (Boxcox) 4. PCA (5 components) 5. Standard Scaler 6. Model

Steps 5 and 6 were built into a pipeline and gridsearched on to tune hyperparameters.

As mentioned, K Nearest Neighbors performed best. Results below:

1.4.5 Results

K Nearest Neighbors:

ROC AUC Score: 0.757

Log Loss: 8.389