Conner Yin
W1614583
May 11, 2023

COEN 140 - Program 2 Report

I. Introduction

Program 2 is a regression problem that requires utilizing various dimensionality reduction and regression techniques in order to generate a model to predict target values. We're given a training dataset that contains 76,329 samples with 479 features as well as a target feature and our goal is to predict the values of the target feature for all 19,083 samples in a test dataset. The accuracy of the model we choose to use will be measured by Root Squared Mean Error (RSME), essentially the differences between each of our predicted values and the actual values. Initially, we will use dimensionality reduction techniques in order to select and/or extract features into relevant numerical values. Then, we will use sci-kit learn libraries in order to test various methods (SVD, PCA, LLE, etc), finding the optimal methods to generate a regression model, and use our model to predict values for the target feature for the test dataset.

II. Methodology

Firstly, we will import the training and test data, and store them in multiple pandas dataframes using the pandas read to csv function. We will also store the target feature from the training dataset into a numpy array/dataframe of its own.
In the feature selection process, it's important to only select values that have are relevant for regression. This means that features containing nominal values likely should not be included in the process as there are differences, but the differences between them are not meaningfully relevant. In order to determine what type of variable a feature is, we look through dictionary.txt's description of each feature. While this process can be automated, I found the description formats to be too varied and opted to sort the type of variable manually, settling on four categories: numerical, yes/no, x/blank, irrelevant. These values will need to be converted to meaningful numeric values or not be selected.
Numerical features will not need to be changed in the pre-processing operation and include counts (e.g. POP901: number of persons) and percentages (e.g. ETH1: percent white). The next two categories are both binary features since there are only two possibilities. Examples include COLLECT1 and NOEXCH which are either Y/N or X/_ respectively. As part of pre-processing, I have replaced the values of Y and X to 1, and N and _ to 0. Finally, irrelevant values include dates (e.g. ODATEDW: Origin Date in YYMM format), arbitrary codes (e.g. TCODE: Donor Title Code), and numeric but purely nominal values (e.g. ZIP: Zipcode). While these features may or may not influence the target variable, the regression model will not be able to accurately utilize these features in a meaningful way for prediction, thus they will be excluded in the pre-processing operation.
Following processing the above feature categories, our linear regression functions require all values in the dataframe to be numerical. Even after pre-processing, there will be several stray empty strings or null values present in our dataframe. We can use the fillna function to replace all null values with 0, and cast our entire dataframe as a float64 type using the astype function. This is the final step in pre-processing and needs to be applied to both the training and test datasets.

At this point, we can run the dataframe through a multitude of dimensionality reduction techniques. Specifically, I tested Principle Component Analysis (PCA), Truncated Singular Value Decomposition (SVD), Gaussian Random Projection (RP), and Locally Linear Embedding (LLE). All of these functions can be imported from the sklearn library. Each of them have a goal of compressing the data into a smaller amount of features, which we can customize using the n_components parameter. I will further elaborate in the experimentation and optimization section how I determined the best model to use. Implementing the method, we will fit_transform the train and test datasets, which will return a modified version of the training and test datasets with n components. The modified version can increase the accuracy of our model slightly, as well as decreasing computational power when predicting.
Finally, we will call the Linear Regression model (also imported from sklearn) and fit it to our modified training dataset and target values. We can then use this model to predict the target values given the modified test dataset. Then, the prediction is printed to the prediction.txt file for submission, and the model's accuracy will be determined by the random mean squared error (RMSE) between the prediction for the target values and the actual target values for the test dataset.

III. Experimentation and Optimization

In order to derive the optimum model, I split off the last 3000 lines of the train data to act as a temporary test data. This way, we know what the actual target values should be and can calculate the RSME of our prediction compared to the actual values. I tested the previously mentioned methods: Principle Component Analysis (PCA), Truncated Singular Value Decomposition (SVD), Gaussian Random Projection (RP), and Locally Linear Embedding (LLE). We also want to test different number of components, and I tested each of the four methods with n_components from 1 to 30. After evaluating the RMSE score between each model's prediction and real target values, I found that all the methods yielded very similar results but a low n_components value (1-3) gave the best results (lowest RMSE). Since the errors were so similar on the last 3000 lines, I decided to just try multiple predictions for the CLP. The best result was Singular Value Decomposition (SVD) with n_components = 1. The exact score will be discussed in the next section.

IV. Evaluation and Results

At the time of writing this report, my model is in 2nd place on the CLP with an RMSE of 3.8471%. Looking at the class distribution, it's apparent that many other students used a very similar methodology to mine. The difference between 1st place and 10th place at the time of writing is 0.0824%. This indicates that my solution is likely close to optimal at least under the scope of the assignment. The minute differences may be due to different choices in n_components or processing method, or it could be due to several extra features added or removed in the pre-processing. However, I'm very happy with my result and hope that my model can remain in the top 3.