Data Mining Final Report

For our project our group did an analyzation of the Portuguese “Vinho verde” which translates to green wine. To understand how to analyze the data we first had to evaluate the data. In evaluating the data we had to make sure that it met the criteria for the project. Once we understood the data we had to import all of the necessary libraries. While doing different processes we generated various visualizations to show how the data evolves as we did the work. As the work continued all of the work done led us to our final conclusion.

The first step in the process is to get a proper understanding of the data. The data consists of two datasets, red and white vinho verde wine samples from Portugal, to model the wine quality based off of chemical testing. Both of the data sets meet the requirements for the project. In the two datasets there are twelve attributes. All the features of the data are all continuous, and the labels can be considered categorical and continuous since they range from three to nine. The attributes or input variables that were based on the chemicals as follows fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sugars, total sulfur dioxide, density, pH, sulfates, alcohol, and finally the output variable quality. The quality was scored from zero to ten, zero being the worst and ten being the best quality. In the dataset all the attributes are clearly labeled so there was no guessing to their meanings. Also there were no missing values so there was no need to fill in missing values. Now that we had the understanding of the data we could begin the process of analyzation.

To begin we first had to import the needed libraries. The first import was cross validation to assess the predictive performance of the models and judging how well they perform. The cross validation function is from sklearn.model_selection. Logistic Regression, a

classification algorithm, was used for the prediction of the probability of a categorical dependent variable was imported. Also imported, on the same line, was Logistic Regression for the assignment of 1 for yes and 0 for no to the dependent variable. Both Linear Regression and Logistic Regression are from sklearn.linear_model. To make sure that the features we selected had the highest k score we imported SelectKBest from sklearn.feature_selection. To test the individual effects of each regressor we imported f_regression that is a scoring function and cannot be used as a stand alone. Finally to be able to read the csv for each dataset we import pandas as pd, json, random, warnings. Warnings.filterwarnings(action='ignore') is used in the preprocessing.

With the libraries fully imported we can begin to work directly with the data. Since we used data that consisted of two csv the files had to be merged. The merging of the two datasets was done with pd.concat. To read the two individual csv were read with pd.read_csv. Where the two individual csv were known separately as white_data and red_data the now new combined dataset will now be named wine_data. With the joining completed the pre-processing of the data can be done.

To start off with preprocessing, we plotted the data and there was a large imbalance with the data. The graph told us that we had to balance out our data. We knew that we would have to over sample the minority classes and under sample the majority classes. Then we performed PCA on the data for visualization. For the features, we showed the distribution of each feature, along with their boxplots so visualize their outliers. To resample the data, we used SMOTE for the oversampling of the minority classes and NearMiss for the undersampling of the majority classes. Oversampling by a constant is much better than altering the ratio of the original dataset.

It is as if we have the original dataset but with more information. We were able to show the PCA visualization for the resampled dataset. Then we used heatmap to visualize the correlation of the original data and the correlation of the resampled data. The correlation values, specifically the target (quality), became a little stronger with the resampled data. Using the resampled data, we show the graphs with the distributions of the features. For feature selection, we used the correlation values and dropped the features that were less than the magnitude of three. Before the feature selection there were twelve features, and after the drop there were four remaining: volatile acidity, chlorides, density, and alcohol. Then we reduced the original and resampled dataset by splitting into training and testing sets, so we can evaluate the algorithm's model on feature-selected original data. The reduced resampled dataset was then visualized using PCA with two components.

After all that preprocessing, we ran cross-validation on multiple algorithms to find the one with the best accuracy. Of course we had to import the libraries for the algorithms we were testing, like linear regression, random forest classifier, k-nearest classifier, and decision trees. We showed the confusion matrices and classification reports for each algorithm between the original and the reduced datasets. After comparing the accuracies in the reduced datasets, we concluded that random forests classifier yield the best results. Also, the performance on the original data is better and has better metrics with unreduced oversampled data than the reduced, feature-scaled oversampled data.

EDA (plots)

Upon initial inspection, we noticed that the entire data has 6,497 samples and 12 features. The target/output is a closed set of numerical values, so a regression and classification approach could be taken.

We checked the dataset for null values and any potential anomalies. Describing the data, we took note of the min, max, mean and standard deviation of each feature.

After ensuring that the data was ready to analyze, we checked the class distribution of the dataset. The 2 majority classes, 6 and 5, had 2,836 samples and 2,138 samples respectively. The 2 minority classes, 3 and 9, had 30 samples and 5 samples, respectively. Clearly there was an issue with an uneven, skewed class distribution, so the logical next step was to research a little more about class imbalance. This will be discussed further in the next section.

Given the dimensions, we attempted to plot the entire dataset by reducing the dimensionality using PCA. Conventionally, plotting PCA plots require scaled data, so we used the MinMax scaler to scale the features of the data. Color-coded by class, the PCA plot showed no noticeable relationship between any features. The plot seemed to be scattered all over the place, overall useless. Despite this, the information retained from the two PCA components summed up to be about 55.5%. This isn't great, but it is decent given the dimensions.

Another important part of the EDA process was using boxplots to detect and visualize outliers of the features. Just about every single feature contained outliers. However it is important to note that after transforming the data to fit a normal distribution for each feature, there were little to no outliers.

The last of the plots we used for EDA was seaborn's distribution plot. Visualizing the distributions can be key to determining what changes must be made to the data in the

preprocessing phase. Our observation of the distribution plots are that most of the features of the data did not follow a Gaussian distribution.

Preprocessing (fixing class imbalance)

One important detail about our approach to testing various preprocessing methods is that for each method, we tested its impact with a random algorithm, usually LogisticRegression, to see if said method was useful or not.

Scaling

Although there was nothing special about the description of the dataset, we realized that one approach we could take was standardizing and normalizing the entire data as a result of the aforementioned distribution observations. After standardizing and normalizing using the PowerTransformer within sklearn.preprocessing, we compared the results of a random algorithm on the original data and the transformed data. The performance actually decreased with the transformed data. We also tried feature scaling using the MinMax scaler and the Standard scaler, and the same thing happened. There was no improvement. It turns out that these two scalers were only useful for reducing the dimensions of the data to plot using PCA components.

Resampling

(Spoiler Alert!) Resampling was the bread and butter of our approach to this project. Without resampling, We're not sure that we would have found a way to get the good results that we did.

Part of our research on class imbalance was finding out that resampling, and all other preprocessing methods, should be done on training data, not the entire data (training and testing).

The purpose of training a machine learning model is to train on a sample of data and validate on completely unseen data.

Our approach to resampling was to construct a new dataset with synthetic samples of the original dataset, train models on the resampled data, and validate on the original data. Before going further into the resampling process, it is important to note that our models may not be as accurate as we think because some of the samples used in validating (original data) may have been included in the training set (resampled data). To resample, we used the SMOTE (Synthetic Minority Over-sampling Technique) from the imbalanced-learn library. The idea was to increase each class total by a constant amount so that the ratio of classes was preserved. Also worth noting is that samples weren't copied directly when using SMOTE. Instead, synthetic samples were created using a neighboring algorithm.

After resampling the original data, constructing the new dataset, we compared the correlation values of both. We observed that the correlation values of the target variable "quality" increased. To observe the correlation of both the original data's correlation values and the resampled data's correlation values, we plotted them side-by-side using seaborn's heatmap.

We also plotted both datasets' feature distributions side-by-side to see if there was any change. Unfortunately there wasn't much of a change in the distributions for each.

Feature selection (by correlation)

For feature selection, we used SelectKBest and f_regression from sklearn's feature selection module. Because this approach select features based on the k-best correlations, we found out what those correlations were using python's dictionary data structure and the json library. We constructed a new dataset for the feature-scaled (reduced) resampled data. However

if we were going to train a model on the reduced resampled data, we had to scale down the original data the same way.

Cross-validation

In our presentation, we mentioned that we applied cross-validation with various regression and classification algorithms. However this was before we realized that resampling should be done during the training phase and not on the entire data (training and testing). Due to our approach of constructing a new dataset for resampled data, training on the resampled data and validating on the original data, cross-validation could not be performed.

Applying different algorithms

We split up the application of different algorithms by classification and regression. The classification algorithms we tested out on our preprocessed data are K-Nearest Neighbors, Logistic Regression, Naive Bayes, Decision Tree Classifier and Random Forest Classifier. The regression algorithms we tested out on our preprocessed data are Linear Regression, Decision Tree Regressor and Random Forest Regressor.

The RandomForestClassifier performed the best among the classifiers, and RandomForestRegressor performed the best among the regressors.

Performance/evaluation

The metrics we used to evaluate the classifiers were accuracy, precision and recall. Part of research mentioned that accuracy may not be a great approach to measuring performance on imbalanced data. For the regressors, we measured performance using Root Mean Squared Error and accuracy score.

Final model

Our final model was the Random Forest Classifier, as it performed the best of all the algorithms we tested. In the end, we ended up with 95% weighted average for precision, 95% weighted average for recall and 95% accuracy.

Ways to improve performance

Some possible ways to improve the performance would be to bin the targets, normalize the features to match a gaussian distribution and change the random_state of the splits used in training the model.