Homework 1

Kelsey Hawkins

Analysis

The first steps that I took was loading in the dataset and exploring the features, attributes, data types, and data structures. By printing out the dataset, it can be confirmed that all independent variables are numerical and the dependent variable is categorical to represent a group assignment. There was no need for further cleaning as there were no null or missing values.

The data has a total of one thousand rows. When grouped by the categorical dependent variable, "Group", it can be broken down into an 'A' and 'B' group. Group A has 405 observations and group B has 595 observations. Given group B has a slightly higher proportion of observations, about 20%, it will be considered when analyzing the model's performance for class imbalances. There are eight independent variables, all with generic naming conventions, but when analyzed in a correlation matrix, we can see some variables having slight correlations with each other. The highest value is between X1 and X7 of -.272, showing a negative correlation between the two variables.

Methods

The first model built is a hyperparameter tuned SVM. It consists of a pipeline that handles normalization as well as preprocessing while including GridSearchCV to tune the hyperparameters and refit the model. With a SVM, I needed to tune the values of the Kernel, C, and gamma.

The C parameter is a number that controls the error or slack of the support vector classifier function. A large value makes the model prone to overfitting as well as narrow margins with fewer support vectors, and a small value is the opposite of those three facts. The hyperparameter tuner was given the options between 0.001 and 50.

Gamma is related to the kernel function in the sense that it acts as an exponentiating factor in the radial bias function. The higher the value of gamma, the more points are going to be attracted to the decision boundaries of the hyperplane and the model may become overfit as the kernel function is going to pay closer attention to the data closest to the separating line. As gamma gets smaller, the kernel function is going to be less attentive to the data close to the boundary and it will possibly be underfit to the data points as the boundary can be very large and over-predicting. The hyperparameter tuner was given the options between 0.001 and 5.

The kernel hyperparameter will either be set to linear or RBF (Radial Bias Function). The radial bias function will project the data into infinite dimensions, allowing for the model to rely on kernels and taylor series to utilize a polynomial function that best fits the data. This allows the model to create separating hyperplanes regardless of the data having a linear relationship or not. Utilizing the radial bias function kernal is a popular and successful function to maximize the SVM algorithm and clustering. The linear function is going to assume the data is linearly separable and a straight hyperplane will be utilized to classify the data. This function is not ideal due to naturally occuring data not always having a linear relationship; a dataset that is non linearly separable may perform poorly with a linear function kernel.

The second model built is a simple Logistic Regression, with no hyperparameter tuning. A pipeline is used to preprocess and streamline the model for simplicity, accuracy, and clean prediction.

The third model built is a K-Neighbors Classifier, and utilizes GridSearchCV to tune the n-neighbors hyperparameter. By changing the n-neighbors, the model is going to look at n items around each data point in order to classify the data point. When n-neighbors increase, the model may have large decision boundaries or clusters and, depending on the data, the model may underfit or miss key features and have low accuracy scores. N-Neighbors is critical to a successful kNN model, as the model should have a well chosen number to classify data points correctly to make accuracy predictions. The tuner was given a range of values between 1 and 20.

Results

The SVM had an expected performance, and it is proven through the tuner's hyperparameter choices and output metrics. The kernel chosen was a RBF, C was 25, and gamma was 0.01. Both training and testing scores were within the range of each other which means the model is predicting unseen data with the optimal amount of bias and variance. Both scores were high, with 78% accuracy and 85% ROC/AUC. This model was able to classify unseen data with high success rates, and the confusion matrices show a balanced tradeoff between accuracy and precision. In the training data confusion matrix, the data shows a high amount of misclassified data points in relation to the group A. This may be because of group B having a higher class proportion than group A, so the SVM formula was not able to successfully find the boundary between the two groups. When looking into the data utilizing a scatter plot of all the variables grouped by their predicted group, the data points are very mixed in with eachother. This makes it impossible to plot a hyperplane it in a two dimensional space.

Through exploration, I can not find a simple answer to why the training confusion matrix is predicting many A's incorrectly.

The logistic regression model performed the same as the SVM. There was no significant difference between the training and testing scores which indicates a sufficient tradeoff between bias and variance. The accuracy score was 78% and the ROC/AUC was 85%. This model has a slightly better training accuracy than the SVM, according to the confusion matrix. The Logistic regression was able to correctly predict the groups more than the SVM, as it fell short in predicting the A group correctly. This may be because of the kernel function of the SVM allowing more slack to the points around the hyperplane's boundaries, on top of the class imbalance of B group, the model may have allowed the hyperplane to classify some of those points incorrectly.

The KNN model performed much worse than the previous two models, with clear overfitting present. The tuner chose a K of 1, which is alarmingly low for this dataset, and in general for a KNN model. There is clear evidence of overfitting due to the testing metrics being much worse than the training metrics. Both the accuracy and ROC/AUC have training scores of 1, which is a negative indication for the model's performance as it has fit the clusters perfectly around the training data. The testing accuracy was 69%, and the test ROC/AUC was 67%. With those lower metrics, the model did not perform well in classifying the groups of the dataset and was unable to determine the difference between groups on unseen data. I believe changing the n-neighbors parameter may improve the model, however the GridSearchCV was used to choose the parameters and chose a small value due to the scoring being on accuracy.

In production, I would choose the logistic regression model. Even though the SVM had almost the exact same metrics, the confusion matrices made me choose the SVM. The SVM was able to get high metrics, even though it was classifying many A groups as B groups in the training data. In production, I would want the most accurate predictions even if the metrics are the same as the other model. When looking at the math in this problem, it seemed that the SVM handled the messy data the best by utilizing the kernel trick, and I would want to scale this to production.

Reflection

In this assignment, I did not come across any struggles or problems with the code or analysis. However I did refer back to the math of the models to justify my answers and think about why the models performed the way they did. It was interesting to be able to quantify them with a simple classification problem and think about how each model handles class imbalances in the slightest way, by about 20%. In the future, I would approach it the same way as I did here, as it seemed to work well.