CISC 372 Assignment 1

Model Development

To get a feel for sklearn and how to use it, I began by going through the sample script and understanding each step and what it was doing. Once I felt confident with it, I began experimenting with changing the possible values of the hyperparameters of the XGBClassifier that gridsearchcv could choose from. This hyperparameter tuning seemed to only improve the accuracy slightly, so I decided to implement some other models and compare their initial performances before continuing to tune the hyperparameters.

Using their default parameter values, I tried the sklearn RandomForestClassifier as well as the sklearn MLPClassifier. I picked these models as they both work well with categorical and numerical values and can pick up on linear and non-linear relationships between features, which is ideal when we don't know the type of relationships that exist. These models are computationally more expensive than others, however, so it is expected that the computation times will be relatively long. The results of the RandomForestClassifier, MLPClassifier, and the XGBClassifier showed very similar accuracies as they were all between 69% and 70%. To try and increase the performance of these models, more data was added by increasing the number of features in the training data. I evaluated the features and decided which attributes would most likely hold the most information in context with this problem and added those features. The total number of features in the training data fed into the models was increased from 7 to 26. This increase in data also means an increase in information for the models to make predictions. Therefore, it was predicted that this would increase the models' performances. It only resulted in an accuracy increase of about 1% to 2%, which was lower than expected, however it did still increase performance, even though it was only a small amount.

The next step to try and improve the performance was to implement an ensemble method, as it was hoped that these three models, which achieved mediocre performance separately, would be more powerful together. These three models; XGBClassifier, RandomForestClassifier, and MLPClassifier, were put into a VotingClassifier to use bagging. The VotingClassifier takes the votes of each of the models for each record and creates a final label of the class with the most votes. This technique increased the accuracy of the model significantly, up to about 83%. When the test data predictions were submitted, the accuracy was only 72.955%. To combat this overfitting, the next ideal step would be to collect more data for the training set, to try and make it more generalizable to the test set.

Instead, tuning the models hyperparameters was experimented with in an attempt to improve the model's performance. For each model, only the most important parameters were tuned as they have the most power in improving a model. One parameter was given 3 to 5 options for GridSearchCV to try, while all other parameters were held constant. For numeric parameters, GridSearchCV would give the best value for that parameter, and then another 3 to 5 options were given for that same parameter, with two greater than and two less than the previous ideal value. Each time, the range of values would get smaller as we reached closer to the optimal value. Once the optimal value was found, that parameter was held constant with that optimal value and the same approach would be used for the next parameter. For non-numeric parameters, such as the activation function for MLPClassifier, the option with the best result was simply chosen as its value. This hyperparameter tuning increased each individual model's accuracy by roughly 1%. When these results were inputted into the VotingClassifier,

the ensemble model achieved an accuracy of 85.85%. This accuracy is over 15% better than each of the models individually. When the test predictions were submitted, however, the accuracy was only 74.475%. This means that this ensemble model is overfitting. To prevent this, more data would need to be collected for the training data to make the model more generalizable to the testing data.

Questions

1. We should limit the number of trials per day as it forces users to consider the changes to their model with more thought. If you could test each time you make a change, this doesn't force you to think at a high level whether your changes will make a positive or negative impact, you are just waiting to see what the results tell you. Limiting the number of trials ensures us to follow the data science life cycle more carefully, which leads to better results.
2. The private leaderboard will only be used after the submission deadline for evaluation so that people continue to tune their models to try and make them as robust and generalizable as possible. Just because you are first on the public leaderboard does not mean that your model will also have the best performance on the private leaderboard. Also, the test set that the public leaderboard uses is like a validation set, while the test set that the private leaderboard uses acts like a true test set. A validation set is used to evaluate the trained models, and we pick the one that achieves the best performance, which is what Kaggle does for us when it saves your predictions that give the highest accuracy. The testing set is used is used minimally to evaluate the chosen model and reports its performance.
3. One of the models used was Random Forest. The flexibility was controlled through various hyperparameters. To make the model more flexible, the number of trees was increased in the forest from the default of 100 to 500. To make it less flexible, the maximum depth was set at 60. This kept the trees from growing too much and becoming to too specific and not generalizable.