CISC 372
Assignment 1
Tyler Mainguy
20023142

Process

The process in developing my model was not overly complex, but it did take a fair amount of time to tune parameters and to discover which aspects of my model were performing poorly, and why.

I began my process with the initial configuration that was provided in the sample assignment. I start off by attempting to alter some of the hyperparameters in an attempt to improve from the baseline score. I started by adding some additional parameters to my gridsearch method, beginning with colsample_bytree (how many columns will be sampled per tree). I had read online that this was an important parameter when tuning the XGBoost model, so I found some common values and ran them through the grid search. I tried ranges from 0.5, 0.7, and 0.9, and didn't see any substantial improvements in my model. I then added some additional columns as features for my dataset. This gave me an approximate 1% increase in the quality of my models predictions, which I didn't consider substantial (but according to the leaderboard it was!). After this, I decided that I would attempt to use a different model. I spent a few hours reviewing tensorflow syntax, and configured a deep neural network in an attempt to classify the data. Due to the small dataset, even this deep neural network had pretty poor base predictions (62% accuracy), so after some tuning of hyperparameters I decided to go back to an ensemble based method.

After researching various popular ensemble methods (such as XGBoost), I settled on LightGBM from Microsoft. This model seemed to be as robust as XGBoost on average, while also training at a much more efficient rate. I passed in the same parameter list as I had given to the XGBoost models and used the default LightGBM model (no parameter tuning) to gauge how the base model performed. It's base was higher than that of XGBoost, but not by a substantial amount. After this stage, I decided I would spend more time tuning parameters in an attempt to gain better accuracy. I looked online to find what parameters tend to be the most important when tuning LightGBM, and I settled on a set that I would test first; number of leaves, number of estimators, max depth, learning rate, and minimum data in a leaf. I began with fairly small and well distributed values for each of these (approximately 3 unique values per parameter), and ran 3-fold cross-validation with these parameters. The best model at that time was one with fairly small values for each of these parameters, and the accuracy of my model had increased to approximately 72%. I decided to use smaller values for both the max depth and the minimum data in leaf, as there wasn't overfitting to the training set (test set accuracy was similar), so by allowing more general rules I was hoping to get a higher accuracy. Further tuning of these parameters (and other parameters, such as gamma) resulted in little to no improvement in my model. I decided at this point to spend more time preprocessing and feature engineering in order to improve my models accuracy. I spent more time reviewing the statistical properties of the numerical attributes, and tried to get an intuitive idea of which categorical attributes may be useful. I then decided to include variables; latitude, longitude, review scores (and other review attributes). My model (performing the same cross-validation with the same parameter lists) then performed up to 73% accuracy on the training and testing set, which was a sign that feeding the model more data was effective in increasing its ability to correctly identify data. I then went on to add further data that I believed would assist in the classification process (such as minimum and maximum nights, availability, number of reviews, etc.).

Answers

1. You should limit the number of trials per day as the more attempts you get, the more likely you are to overfit to the public test set. This will result in models that aren't necessarily the most robust, but just reflect the subset of test data that's on the public leaderboard.
2. It's designed like this for a similar reason as above; if the same set was used for public and private, users would overfit their models to the public testing set. By separating the public and private datasets, the final test set that the private leaderboard uses will be unused, therefore preventing users from overfitting on that set. This will result in the most accurate measurement of the final models submitted, as this is truly data they've never seen before.
3. One of the models I used was LightGBM, which is a variant of gradient boosting. I was able to control it's flexibility through a variety of parameters I used to tune the model. I controlled it's flexibility by altering the learning rate of the model, and by changing some of the features of the tree, such as max depth. By increasing the learning rate, the model becomes more flexible, as newer data has a significantly higher impact on the models predictions. Similar properties hold for increasing the depth of the tree, as allowing for further branching allows the model to more specifically model the data (at the risk of overfitting). I tended to alter the control of my model after I had attempted to test it on the testing set. This gave me a good indication as to whether or not my model was robust. In cases where the testing accuracy was much lower than the training, I would reduce the maximum depth of the tree and the learning rate, which in turn reduced the flexibility of the model. This iterative process aided in my ability to determine optimal parameters for bias and variance.