Best Hyperparameter: {n\_components: 10, n\_iter: 200, tol:0.5}

Metrics:

* Accuracy: 0.671721311475409
* Balanced Accuracy: 0.633610282429531
* Precision: 0.7325293948460894
* Recall: 0.6717213114754098
* F1 Score: 0.6847787712176521

For the Hidden Markov Model, a MultinomialHMM was built for each family (21 families). Once the HMM was trained, each test case was scored with respect to each model with the method *score* (see the documentation [here](https://hmmlearn.readthedocs.io/en/latest/api.html?highlight=predict#hmmlearn.base._BaseHMM.score)). This will return the log-likelihood of the sequence with respect to the model you call it upon. The test case is then classified in the class of the model which returns the highest likelihood result.

For our experiments, we tested out a wide range of hyperparameters for amount of hidden states (n\_components), maximum amount of iterations (n\_iter), and convergence threshold (tol). RandomizedSearchCV was used to fine-tune the hyperparameters. We found the optimal model used 10 hidden states,200 max iterations, and convergence threshold of 0.5.

For Bagged HMM, we built 5 HMM models with each model built using a subsample (with replacement) of the training dataset dataset, which had size of 60% of the training dataset. Afterwards, we implemented a voting system where each model had equal weights on the test set. In the event of a tie, we added the individual scores (log-likelihood) for each model of its top prediction to deduce the final outcome.

For Boosted HMM, we built 5 HMM models with each model again built with a subsample that had a size 60% of the original dataset. For each subsequent model, the errors of the previous ensemble were automatically inputted into the subsample dataset to train this particular model. We also implemented voting system similar to that of the BaggedHMM.