Task 4 Report

Primary: Anuk Centellas **Secondary:** Katelynn Carlson, Cooper Cox, Nina Ervin

1 Methods

We used Scikit-learn to test various model types for this dataset. Since the dataset is large and this is a classification task, a neural network (NN) was the first model we experimented with. We used MLPClassifier to create a neural network and manually tuned the hyperparameters, saving the trained models that performed best in joblib files. We also created a linear model using LogisticRegression because it was simple to implement and gave us another baseline. The linear model uses StandardScalar from Scikit-learn to preprocess the data. We also experimented with random forest models using RandomForestClassifier because the input data is low dimensional, and although the dataset is large, we wanted to explore different model types. We also created an ensemble model with some of our trained NNs, experimenting with the number of NNs and the number of votes that each NN is given. To create a baseline, we calculated the accuracy when the most common class was predicted for all classes. We did this for both the training data and the dev set.

2 Model Details

After tuning various neural networks, we created an ensemble model consisting of the twelve best performing NNs. All of the NNs in the ensemble model use early stopping to prevent overfitting, ReLU as the activation function, and Adam as the optimizer. NNs 7 and 11 use an adaptive learning rate while the rest use a constant learning rate. The ensemble model uses probabilistic voting to predict the class for each input through the predict_proba method in Scikit-learn. The four best performing neural networks within the ensemble model are given three votes each while the other NNs are given one vote each. The hyperparameters for each of the twelve NNs (ordered from best to worst performance) are presented in Table 1.

Table 1: Hyperparameters of NNs in Ensemble Model

NN	Num Hidden Layers	Num Hidden Units	Minibatch Size	Learning Rate
NN 1	5	64	128	0.0001
NN 2	5	64	128	0.0001
NN 3	5	64	128	0.0001
NN 4	5	64	128	0.0001
NN 5	5	64	128	0.0001
NN 6	8	64	64	0.00001
NN 7	7	64	64	0.00001
NN 8	5	64	128	0.0001
NN 9	3	256	64	0.000001
NN 10	3	256	64	0.00001
NN 11	5	64	64	0.00001
NN 12	3	256	64	0.001

3 Results

Table 2: Accuracy for Baselines and Models

Model Type	Accuracy (Dev)
Baseline (Avg Mode Train)	0.127
Baseline (Avg Mode Dev)	0.142
Linear Model	0.406
Neural Network (Best)	0.449
Random Forest	0.292
Ensemble Model (NNs)	0.456

Table 2 shows the majority class baselines and the results from the trained models after hyperprameter tuning. All models do significantly better than the baselines, with the ensemble model having the best performance overall. While the random forest does better than the baselines, it performs poorly when compared to all other models. This is likely due to insufficient hyperparameter tuning, which is difficult given the size of the dataset. For the neural networks, we found that 5 hidden layers gives the best performance while being less computationally expensive, since the performance does not improve with more than 5 layers. We also found that 64 hidden units generally provides the best performance with 5 layers, and that minibatch sizes ranging from 16 to 512 have a minimal impact on performance. We tried different learning rates and observed that when set below 0.001, the NN performs worse, and when set above 0.001, the NN usually performs marginally better than when set to 0.001. When creating the ensemble model, we experimented with giving the top four NNs more votes than the rest of the NNs. We found that three votes is the optimal number, as shown in Figure 1. Although the performance difference is minimal, the best result is with three votes. We include twelve NNs because the performance increases as we add more, shown in Figure 2, but the performance of the individual NN after the twelfth best one decreases more significantly.

Figure 1

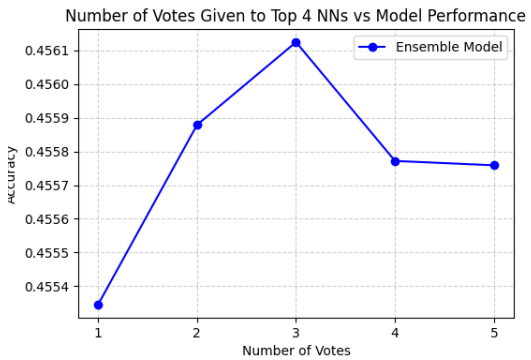
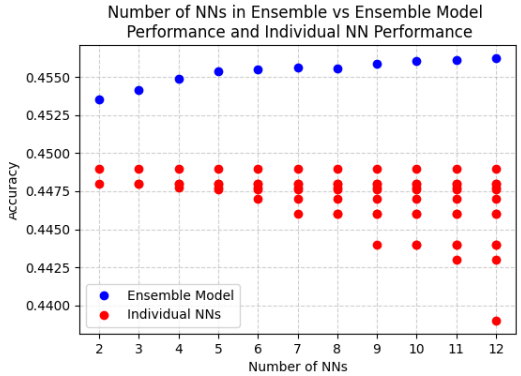


Figure 2



4 Distribution Of Work

As a group we collaborated on ideas and modeling suggestions. All code and results for task 4 were made by Anuk Centellas. The task 4 report was written by Anuk Centellas, with review and editing by all other group members. The results table formatting was written by Nina Ervin.