Name of the student: *MADINENI VENKATA SAI TEJA*
Roll Number:*DA24B031*

I used Softmax+Bagging,XGB,XGB+BaggedSoftmax

Softmax+Bagging

softmax + bagging. For each MNIST image, every model gives 10 scores (for digits 0–9). Softmax turns these scores into probabilities between 0 and 1, and all of them add up to 1. The digit with the highest probability is the prediction. With bagging, I train many softmax models on different samples of the data. Each model gives its own probability vector, and then I average all the probabilities. This soft-voting makes the final prediction more stable and more accurate.

Table 1: BaggedSoftmax Hyperparameter Tuning Results

n_models	sample_ratio	lr	epochs	Time (s)	Accuracy	Precision	Recall	F1	Bias	Variance
20	0.7	0.05	200	44.05	0.9020	0.9011	0.9006	0.9005	0.12384	0.00061
25	0.7	0.05	250	67.08	0.9043	0.9033	0.9029	0.9027	0.12206	0.00069
25	0.6	0.10	250	58.91	0.9081	0.90698	0.90689	0.90665	0.11799	0.00110
30	0.8	0.05	400	148.30	0.9081	0.90710	0.90680	0.90662	0.11891	0.00077
30	0.8	0.10	400	140.84	0.9115	0.91030	0.91018	0.90996	0.11585	0.00109

we got best as 0.9115 Accuracy and 0.91 F1 score We shall improve this

XGB

XGBoost with One-vs-Rest for MNIST. This trains 10 XGB models, one for each digit from 0 to 9. For each digit, it makes the labels binary: if the image is that digit $\rightarrow$ 1, else $\rightarrow$ 0. Then it trains an XGB model only for that digit. When predicting, all 10 models give a probability for their own digit. Then we simply pick the digit that has the highest probability. So basically 10 XGB models check the image, each one says "I think it is my digit", and the one with the highest probability wins.

Table 2: OneVsRestXGB Hyperparameter Tuning Results

n_estimators	lr	max_depth	reg_lambda	Time (s)	Accuracy	Precision	F1	Bias	Variance
20	0.5	4	1.0	117.72	0.9426	0.94215	0.94204	0.0133	0.0441
20	0.5	5	1.0	187.18	0.9482	0.94784	0.94777	0.0027	0.0491
30	0.5	4	1.0	169.26	0.9482	0.94775	0.94763	0.0023	0.0495
30	0.5	4	0.0	167.91	0.9319	0.93120	0.93114	0.0097	0.0584
30	0.3	5	1.0	313.32	0.9506	0.95036	0.95023	0.0033	0.0461
30	0.5	5	1.0	287.73	0.9524	0.95205	0.95198	0.0003	0.0473
30	0.5	5	0.0	257.86	0.9301	0.92950	0.92942	0.0052	0.0647

we got 0.9524 Accuracy and 0.95F1 score

XGB+BaggedSoftmax

My model trains two systems: one-vs-rest XGBoost and bagged softmax. XGBoost trains 10 separate models, each one checking if the image belongs to its digit or not, and each gives a probability. Bagged softmax trains many softmax models on random samples and averages their probabilities. In the end, I combine both outputs using weights (0.7 XGB and 0.3 softmax). Then I take the digit that has the highest final probability. So basically both models give their confidence, I mix them, and the strongest probability wins.

Exp	**XGB (ne, lr, d)**	**Softmax (n, r, lr, ep)**	**Weights**	**Acc**	**F1**	**Bias**	**Var**
1	20, 0.5, 5	20, 0.7, 0.05, 250	0.7 / 0.3	0.9363	0.9356	0.0033	0.0604
2	20, 0.5, 5	20, 0.7, 0.1, 200	0.8 / 0.2	0.9362	0.9356	0.0033	0.0605
3	25, 0.5, 5	15, 0.5, 0.1, 100	0.8 / 0.2	0.9342	0.9335	0.0033	0.0625
4	30, 0.5, 5	10, 0.7, 0.1, 100	0.8 / 0.2	0.9359	0.9352	0.0033	0.0608
5	30, 0.5, 5	15, 0.5, 0.1, 100	0.9 / 0.1	**0.9402**	**0.9397**	0.0033	**0.0565**

Table 3: Full comparison of XGB + Bagged Softmax hybrid experiments with all parameters, metrics, bias and variance.

Sadly our model didnt improve from previous we can see the some of the experiments tested below

so our XGB only gave us good results than our XGB+Baggedsoftmax

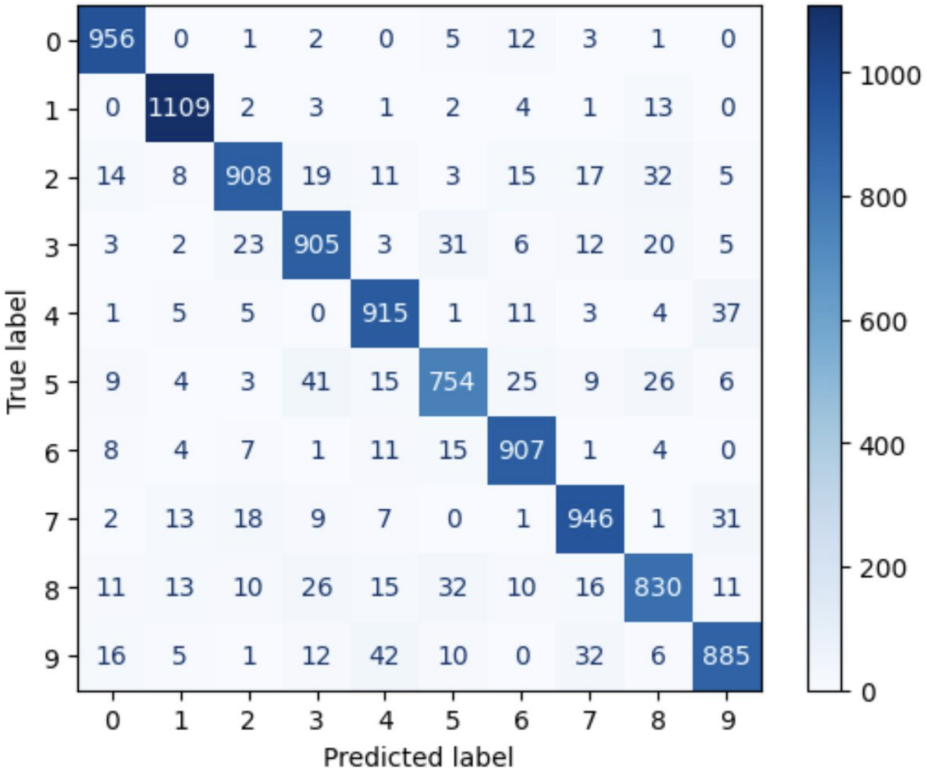

Figure 1: Enter Caption

CONFUSION MATRIX FOR THE TOP MOST F1 Score of BaggedSoftmax

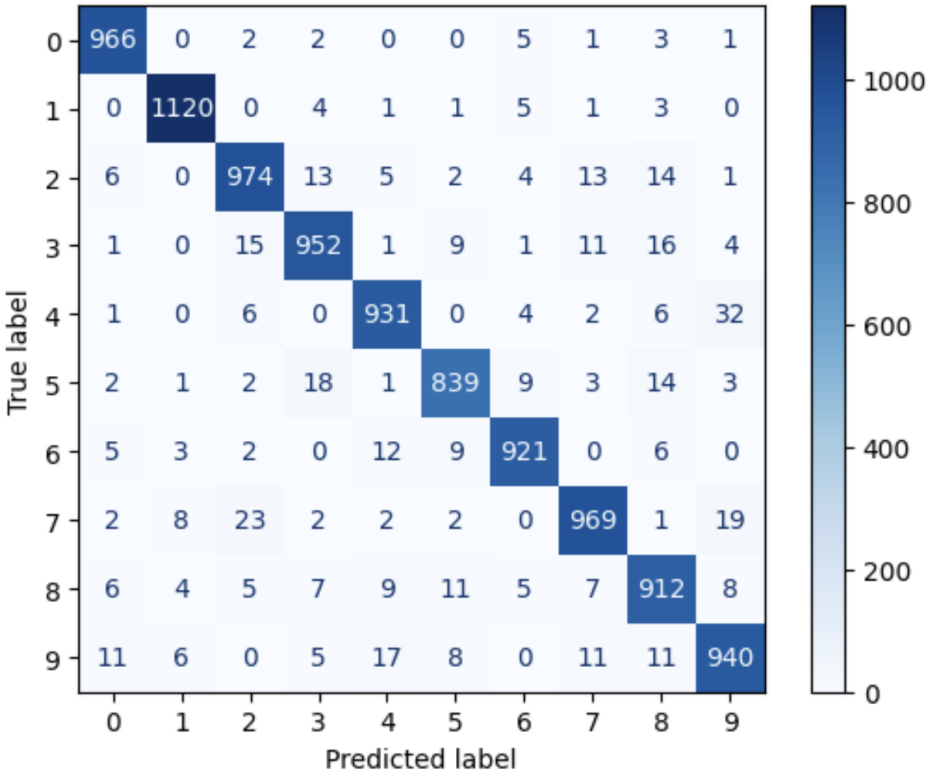

Figure 2: Enter Caption

CONFUSION MATRIX FOR THE TOP MOST F1 Score of BaggedSoftmax

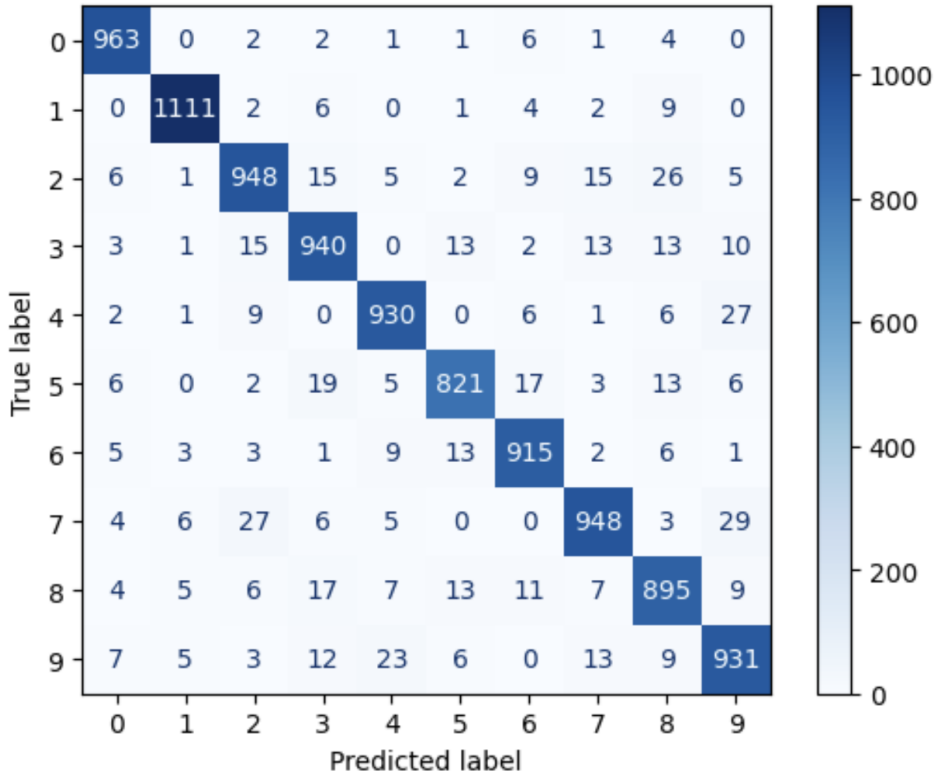

Figure 3: Enter Caption

CONFUSION MATRIX FOR THE TOP MOST F1 Score of XGB+BaggedSoftmax

Steps Taken to Optimize System Performance and Limit Run Time

To make the system run fast and still get good accuracy, I applied a few practical steps. For BaggedSoftmax, I trained many small softmax models instead of one large one. Each model was trained on a random sample of the data, which reduced variance and made the predictions more stable without increasing the run time too much. I also averaged the probabilities from all models so that the final output stayed smooth and reliable.

For XGBoost, I kept the tree depth small (4 or 5), used around 20–30 trees, and set a reasonable learning rate. Since XGB trains ten separate models for digits 0–9, controlling these values helped manage the overall training time. Even with these limits, XGB still achieved strong accuracy.

For the combined model (XGB + BaggedSoftmax), I did not train any new model. I simply mixed the probability outputs from XGB and the bagged softmax ensemble using weights like 0.7/0.3, 0.8/0.2, and 0.9/0.1. This step takes almost no time, and it allowed me to quickly test if combining both models gives better performance. As I increased the XGB weight (for example 0.9 for XGB and 0.1 for softmax), the overall F1 score improved, showing that XGB is the stronger model in the ensemble.

Overall, the goal was to keep all models lightweight and fast while still maintaining good accuracy on the MNIST dataset.

Evaluation Results on Validation Dataset

BaggedSoftmax:

The best accuracy I got was around 0.9115, and the F1 score was also about 0.91. The results were very stable because of bagging, but the accuracy was limited since softmax is a simple linear model and cannot capture complex patterns.

XGBoost:

XGB gave the best performance out of all the models I tested. The highest accuracy I got was 0.9524, and the F1 score was around 0.95. The model learned the MNIST data very well and gave strong results on the validation set.

XGB + BaggedSoftmax:

The combined model reached around 0.9402 accuracy and 0.9397 F1 score. The predictions were smoother, but the hybrid model still did not beat the pure XGB model. When I increased the XGB weight to 0.9 and reduced the softmax weight to 0.1, the F1 score improved. This clearly shows that giving more weight to XGB increases the final performance because XGB is the stronger model in the ensemble. So as the XGB weight increases, the final F1 score also increases.

SUMMARY From this whole exercise, I understood how different types of models behave on the same dataset, and how combining them can help or sometimes hurt performance. BaggedSoftmax is very stable but limited. It cannot learn complex patterns, so even after bagging and tuning, it stays around 91% accuracy. XGB was the strongest model. It learns nonlinear relationships, handles each digit separately using the one-vs-rest setup, and gives very confident predictions. It performed the best with around 95% accuracy. The hybrid XGB + BaggedSoftmax idea was to mix a strong model with a stable one. However, in this case, the softmax output softened the strong predictions from XGB, so the final result did not beat pure XGB. The hybrid was smoother but slightly weaker. The main takeaway is that sometimes the single best model already captures everything, and adding more models does not always improve results. In this project, XGB alone was enough to give the best performance. Overall, this exercise helped me understand model behaviour, bias-variance trade-offs, and how both accuracy and runtime matter when building a full classification system.