CS412 Report

**Classification:**

In this task, I worked on a dataset with an imbalanced class distribution and limited samples. To address this, I created synthetic data to improve the dataset size and balance. For the model input, I combined the captions of posts with the biography of the accounts, feeding this combined text into the model for classification. To represent the text, I used TF-IDF vectorization, which initially resulted in a sparse feature matrix with 5000 columns. To address sparsity while maintaining efficiency, I applied TruncatedSVD, reducing the dimensions to 2000 while preserving 98% of the variance.

During synthetic data generation, I first fit and transformed the TF-IDF vectorizer using only the captions, and then incorporated the bio text, transforming the data again to effectively double the dataset. This helped fix the limited data issue and improved the model's robustness.

Initially, separate models like SVM, Random Forest, and Logistic Regression did not perform well individually. To improve performance, I combined these models into an ensemble using a soft voting approach. To ensure balanced performance across underrepresented classes, I used stratified splitting for training and employed stratified cross-validation for evaluation.

For hyperparameter optimization, I used the Optuna library, which utilizes Bayesian optimization to efficiently explore the hyperparameter space. The results of the optimization process were saved in a database file within the repository.

**Regression:**

For the regression task, I utilized the entire dataset of 180,000 samples. Key features included std\_like\_count, max\_like\_count, min\_like\_count, avg\_like\_count, follower\_count, following\_count, comments\_count, and like\_per\_follower. The target variable like count had a skewed distribution, so I applied a log transformation to normalize it before training.

I started with a prototype Random Forest model to establish a baseline. Again I used Optuna to extensively tune my hyper parameters. Afterward, I analyzed the error distribution across different bins of predictions. This analysis revealed patterns in the errors, so I decided to introduce a custom weighting to penalize bins with higher errors. Subsequently, I retrained the model using this weights and used Adaptive Boosting which resulted in halving the log sum error and significantly improving predictive accuracy.