**Competitive Project Mid-term report**

Analysis of Stochastic Gradient Descent Classification Implementation for Income Prediction

In this report, I examine a sophisticated implementation of a Stochastic Gradient Descent (SGD) classifier designed for income prediction. My analysis covers model architecture, preprocessing methodology, and potential improvements for the rest of the semester. The implementation demonstrates the practical application of modern machine learning techniques that I have learned while highlighting areas for future experiments.

The foundation of my implementation relies on the SGD Classifier from scikit-learn. It has incorporated selected hyperparameters to optimize performance and mitigate overfitting risks. It has achieved a score of 0.81565, which is the highest among all the models that I have created for this project. Though it still has room to improve, I think it can offer a good benchmark for me to compare more sophisticated models for the rest of the semester. The following details are:

**Core Model Components**

* Loss Function Selection and Optimization:

The implementation utilizes a 'modified\_huber' loss function, a sophisticated approach that combines the best aspects of logistic regression and hinge loss. Unlike standard hinge loss which is linear beyond the margin, modified Huber loss is quadratic for small margin violations and linear for larger ones, making it more robust to outliers. This quadratic-linear hybrid nature provides probability estimates while maintaining robustness against outliers, enabling more nuanced predictions compared to standard hinge loss and proving particularly effective for binary classification tasks with noisy data. The modified Huber loss allows for smooth gradient updates during training, enhancing the overall stability of the learning process, especially when dealing with mislabeled training examples.

* Regularization Strategy:

Our model incorporates L2 regularization (Ridge) with alpha=0.0001, which prevents overfitting by penalizing large coefficients in the model. This promotes model generalization across different data distributions while maintaining model stability during training iterations.

* Learning Rate Optimization:

The learning process implements adaptive learning rate adjustment with an initial learning rate (\eta0) configured at 0.1. This is complemented by an early stopping mechanism with 5-iteration patience and a validation fraction of 0.1 for monitoring convergence. The learning rate undergoes dynamic adjustment based on training performance, ensuring optimal convergence.

* Training Configuration

The training process incorporates several sophisticated elements: (1) Maximum iteration limit set to 1000 with early stopping capabilities. (2) Convergence criterion tolerance of 1e-3. (3) Reservation of 10% training data for validation. (4) Implementation of 5-fold cross-validation for robust evaluation. (5) Balanced handling of class weights to address potential imbalances

* Model Evaluation Strategy

The evaluation framework includes: (1) Multiple performance metrics including accuracy, precision, and recall. (2) Cross-validation to ensure robust performance estimation. (3) Separate validation set for unbiased performance assessment. (4) Detailed classification reports for comprehensive analysis.

**Data Preprocessing Implementation**

The preprocessing pipeline demonstrates a comprehensive approach to data preparation, crucial for optimal model performance.

* Missing Value Management

1. Categorical Feature Handling:

The preprocessing pipeline implements a dedicated function handle\_missing\_values() that systematically addresses missing data marked as '?' in the dataset. The function creates separate copies of both training and test datasets to prevent data leakage. For three specific categorical features - 'workclass', 'occupation', and 'native.country' - the function calculates the mode (most frequent value) from the training data, excluding the missing values ('?'). This mode value is then used to replace missing entries in both training and test datasets. Importantly, the mode calculation is performed only on the training data and applied to both datasets, maintaining proper machine learning practices by preventing test data information from influencing the imputation strategy.

1. Numeric Feature Processing:

For numeric features, the pipeline implements standard scaling for normalization, incorporating robust scaling techniques for handling outliers. This approach preserves relationships while normalizing scales, ensuring consistent application across both training and testing phases of the model development.

* Target Variable Processing

The target variable undergoes specific processing: (1) Label encoding for binary income classification. (2) Maintenance of class balance awareness. (3) Consistency checks between training and prediction. (4) Proper handling of class weights in model training

**Potential Improvements and Future Directions**

Besides SGD classification, I have tried both traditional linear regression, logistic regression, decision tree with gini impurity test, and adaboosting methods. So far, SGD has provided the highest cross-validation score. There are still room to improve in terms model prediction score. Hence, several strategic enhancements could further optimize the model's performance and reliability. They are sketches for now. This is not meant to be a comprehensive list that I will explore for the final project.

* Feature Engineering Extensions

The feature selection process could be enhanced through implementation of recursive feature elimination and incorporation of feature importance analysis. This would be complemented by selection of optimal feature subsets and appropriate dimensionality reduction techniques.

* Model Optimization Opportunities

1. Hyperparameter Tuning:

Future optimization should implement grid search for hyperparameter optimization, exploring alternative loss functions and fine-tuning learning rate schedules. This process should include optimization of regularization parameters to maximize model performance while maintaining generalization capabilities.

1. Ensemble Integration:

Model performance could be enhanced through consideration of bagging and boosting techniques, integration with other model types, and implementation of weighted ensemble approaches. This would be supported by diversity in base model selection to capture different aspects of the underlying data patterns.

**Conclusion**

While the current implementation demonstrates solid fundamental practices in machine learning, the suggested improvements could significantly enhance both model performance and reliability.

For reference, I have created a public repository for this project. All components of the project are uploaded to it. GitHub link: