CSE 655 – Project Proposal

Student: Merve DÖNMEZ (244201001016)

Project Topic

Privacy-Preserving Credit Default Prediction using Homomorphic Encryption (SEAL CKKS) and Stacking Ensemble

Motivation

Financial datasets contain sensitive personal and transactional information. Organizations often face regulatory and confidentiality constraints that limit data sharing and model deployment across parties. This project explores privacy-preserving machine learning using approximate homomorphic encryption (CKKS) to protect data during processing while maintaining competitive predictive performance. We target credit default prediction as a representative and high-impact use case and evaluate the trade-offs between privacy, accuracy, runtime, and memory.

Key motivations:

- Maintain data confidentiality during model training/inference via homomorphic encryption.
- Quantify the overhead introduced by encryption and identify practical optimization strategies (batching, parameter tuning).
- Provide a reproducible pipeline combining strong tabular models through stacking with HE-based data handling.

Available Datasets

- UCI Default of Credit Card Clients dataset (UCI repository, ucimlrepo id=350). The dataset includes demographic and payment history attributes to predict credit default probability.
  - Usage plan: feature engineering (ratios, interactions, summary stats), class rebalancing with SMOTE, scaling with MinMax, train/test split.
- Optional alternatives for robustness checks (time permitting): UCI German Credit, public Lending Club snapshots.

Links/references:

- UCI Default of Credit Card Clients (ucimlrepo id=350)

Relation to Your Graduate Work

The project aligns with privacy-preserving and trustworthy AI research directions, focusing on encrypted computation and secure analytics. It contributes empirical evidence on the feasibility of CKKS-based encrypted pipelines for tabular credit risk modeling and informs thesis-level exploration of trade-offs (accuracy vs. compute/memory vs. privacy), parameter tuning (scale, modulus, poly degree), and deployment considerations.

Methodology and Planned System

1) Data Pipeline

- Feature engineering: ratio, product, difference, row-wise sums/means/std; feature selection with RandomForest importances (top-10).
- Imbalance handling with SMOTE; MinMax scaling; train/test split.

2) Model

- Stacking ensemble with base learners: RandomForest, XGBoost, LightGBM; meta-learner: LogisticRegression.
- Parallelism via n_jobs=-1 where supported; optional GPU for gradient-boosting (time permitting).

3) Privacy (Homomorphic Encryption)

- Microsoft SEAL (CKKS) for approximate arithmetic on encrypted real-valued vectors.
- Parameters (initial): poly_modulus_degree=4096; CoeffModulus.Create(4096, [40,20,40]); scale=2^35.
- Batch encrypt/decrypt utilities; batch size tuning (initial default 256) to balance throughput and memory.

4) Evaluation

- Metrics: ROC-AUC, accuracy, classification report; optimal probability threshold via argmax(tpr − fpr) on ROC.
- Performance: measure runtime and peak memory for plain vs. encrypted flows to quantify overhead.
- Ablations (time permitting): batch size (128/256/512), CKKS parameters, early stopping for boosters, float32 vs float64.

Hardware Requirements

- CPU with multi-core support (recommended 8+ cores). HE operations are CPU-intensive.
- Memory: 16 GB RAM recommended to comfortably handle batch-based encryption and model training.
- Optional GPU (e.g., NVIDIA CUDA) for accelerated XGBoost/LightGBM training; not strictly required.

Related Literature

- Cheon, Jung Hee, et al. "CKKS: An Approximate Homomorphic Encryption Scheme for Real Numbers." (Original CKKS scheme introducing approximate arithmetic on ciphertexts.)

- Microsoft SEAL Documentation. Homomorphic encryption library with CKKS and BFV schemes and implementation best practices.
- CryptoNets and follow-up works on privacy-preserving ML using HE for inference/training, demonstrating feasibility and trade-offs in practical settings.
- Stacking ensemble literature (Wolpert's stacking and subsequent applications to tabular ML) showing performance gains from meta-learning over diverse base learners.

Please include formal citations in the final PDF (IEEE/ACM/APA) as required by course policy.

Project Plan and Milestones

- Week 1–2: Data ingestion, feature engineering, baseline stacking (plain). Establish metrics.
- Week 3: Integrate CKKS pipeline (encrypt/decrypt), parameterize batch size, measure overhead.
- Week 4: Optimization passes (batch size sweep, early stopping, float32), ablation runs, documentation.

Expected Outcomes

- A working privacy-preserving credit default prediction pipeline with reproducible experiments.
- Quantitative comparison of plain vs. encrypted execution (accuracy, AUC, runtime, memory).
- Recommendations for practical CKKS configurations and batching strategies in tabular credit risk tasks.

Repository

- Source code and experiment artifacts will be maintained at: https://github.com/MerveDnmz/SealCreditDefaultClassifierWithStackingEnsemble

Submission

- Deliverable: 2–3 page PDF with the sections above, figures/tables as needed, submitted by October 14, 2025, 11:59 PM.

References

- Cheon, J. H., Kim, A., Kim, M., & Song, Y. (2017). Homomorphic encryption for arithmetic of approximate numbers. In ASIACRYPT 2017 (pp. 409–437). Springer. https://doi.org/10.1007/978-3-319-70694-8_15

- Microsoft SEAL (release v4.x) documentation. (n.d.). https://github.com/microsoft/SEAL and https://github.com/microsoft/SEAL/tree/main/native/src/seal
- Gilad-Bachrach, R., Dowlin, N., Laine, K., Lauter, K., Naehrig, M., & Wernsing, J. (2016). CryptoNets: Applying neural networks to encrypted data with high throughput and accuracy. In ICML 2016 (pp. 201–210). http://proceedings.mlr.press/v48/gilad-bachrach16.html
- Wolpert, D. H. (1992). Stacked generalization. Neural Networks, 5(2), 241–259. https://doi.org/10.1016/S0893-6080(05)80023-1
- Breiman, L. (1996). Stacked regressions. Machine Learning, 24(1), 49–64. https://doi.org/10.1007/BF00117832
- Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In KDD 2016 (pp. 785–794). https://doi.org/10.1145/2939672.2939785
- Ke, G., et al. (2017). LightGBM: A highly efficient gradient boosting decision tree. In NeurIPS 2017. https://papers.nips.cc/paper_files/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html