CURRENT TOPICS IN MACHINE LEARNING
Homework-4

Team members:

-Gopi Vardhan Vallabhaneni
-Badrinath Reddy Thadikala prakash
-Venkata Siva Rama Vivek Grandhi
-Sumanth Reddy Gutha

1.Introduction:

Privacy-preserving machine learning has become increasingly important in a world where data privacy concerns are paramount. This project demonstrates how Fully Homomorphic Encryption (FHE) can be applied to machine learning models, specifically for house price prediction, allowing users to leverage server-side machine learning models without revealing their sensitive data.

Our implementation follows these key principles:
- The model is trained in plaintext using standard machine learning techniques
- The model weights remain in plaintext
- User inputs and model outputs are encrypted using Open FHE.
- All computations on encrypted data are performed without decryption

This approach enables secure inference where the input data owner never has to reveal their sensitive information to the model provider.

Key Implementation Principles

Our approach adheres to the following core principles to ensure both security and efficiency:

- **Plaintext Model Training:** The machine learning model is trained using standard techniques on plaintext data.
- **Encrypted Inputs & Outputs:** While the model weights remain in plaintext, user inputs and prediction outputs are encrypted to safeguard sensitive information.
- **Secure Computation:** All mathematical operations on input data occur within the encrypted domain, ensuring that neither the data provider nor the model owner can access unencrypted information.
- **Privacy-Preserving Inference:** The encryption scheme ensures that predictions can be made without exposing individual data points, offering a secure alternative to conventional cloud-based machine learning.

2. Background and Concepts

This section outlines the key concepts used in our project, including Fully Homomorphic Encryption (FHE), the CKKS scheme, and linear regression for house price prediction.

Implementation using docker:

After the above implementation the dock generates result.json where all the answers are avilable

Fully Homomorphic Encryption (FHE)

FHE allows computations on encrypted data without requiring decryption. This ensures data privacy, as computations are performed securely without exposing sensitive information. The output remains encrypted and can only be decrypted by the data owner.

CKKS Scheme

We use the **CKKS**, which is designed for **approximate arithmetic on real numbers**—making it ideal for machine learning applications. CKKS enables:

- **Addition & multiplication** on encrypted data
- **Vectorized operations** for efficient batch processing
- **Noise management** to maintain accuracy
- **Rotation operations** for optimized dot product calculations

Linear Regression for House Price Prediction

We model house prices using **linear regression**, where the prediction is a weighted sum of input features:

predicted_price=intercept+∑(coeffi×featurei)predicted_price=intercept+∑(coeffi×featurei)

- **Intercept**: Baseline price when all features are zero.
- **Coefficients**: Learned weights representing the impact of each feature.
- **Features**: Attributes like location, rooms, and income.

The model is **trained in plaintext**, but user inputs remain **encrypted** during inference. Homomorphic computations ensure privacy, allowing secure predictions without exposing personal data.

3. Model Training in Plaintext

We trained a **linear regression model** on the **California Housing dataset** using **scikit-learn**. The training workflow included:

1. Loading the dataset from **X_train.csv** and **y_train.csv**
2. Standardizing features using **StandardScaler**
3. Splitting the dataset into **training and validation sets**
4. Training a **LinearRegression model**
5. Evaluating performance and saving model weights for **FHE implementation.**

Training Code snippet:

```
# Load and prepare data
X_data = pd.read_csv('data/X_train.csv')
y_data = pd.read_csv('data/y_train.csv')

# Standardize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_data)

# Split data
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_data, test_size=0.2, random_state=42)

# Train model
model = LinearRegression()
model.fit(X_train, y_train)

# Save model weights
weights = {
   'coefficients': model.coef_.tolist(),
   'intercept': model.intercept_,
   'feature_count': len(model.coef_)
}

# Save as JSON for FHE use
with open('model/regression_weights.json', 'w') as f:
   json.dump(weights, f, indent=4)
```

Model Performance Metrics:

Feature Set:

The model includes 13 coefficients corresponding to the following features:

- Longitude
- Latitude
- Housing Median Age
- Total Rooms
- Total Bedrooms
- Population
- Households
- Median Income
- Ocean Proximity *(One-hot encoded as 5 features)*

Saved Model Weights (Snippet):


```
{
  "coefficients": [-54846.18, -55261.87, 12714.59, -13471.86, 46271.97, -42444.83, 14734.04,
74896.91,
                2.21e+16, 2.08e+16, 6.98e+14, 1.41e+16, 1.49e+16],
  "intercept": 207510.97,
  "feature_count": 13
}
```


4. FHE Implementation with OpenFHE:

This section details the implementation of Fully Homomorphic Encryption (FHE) using OpenFHE for secure house price prediction.

4.1 Setting Up the CKKS Environment

We utilize the CKKS encryption scheme for secure computation on encrypted data. The key cryptographic parameters are stored in config.json:


```
{
  "indexes_for_rotation_key": [1],
  "mult_depth": 29,
  "ring_dimension": 131072,
  "scale_mod_size": 59,
  "first_mod_size": 60,
  "batch_size": 65536,
  "enable_bootstrapping": false,
  "levels_available_after_bootstrap": 10,
  "level_budget": [4,4]
}
```


These parameters define **security level, computational depth, and efficiency** of FHE operations.

4.2 Loading Keys and Initialization:

To perform encrypted computations, we **deserialize** the cryptographic context, public key, and rotation keys. Below is a key snippet for setting up FHE in **C++**:

```
void CKKSTaskSolver::initCC() {
    if (!Serial::DeserializeFromFile(m_CCLocation, m_cc, SerType::BINARY)) {
        std::cerr << "Could not deserialize cryptocontext file" << std::endl;
        std::exit(1);
    }
    if (!Serial::DeserializeFromFile(m_PubKeyLocation, m_PublicKey, SerType::BINARY)) {
        std::cerr << "Could not deserialize public key file" << std::endl;
        std::exit(1);
    }
}
```

4.3 Homomorphic Evaluation of Linear Regression

The `eval()` function performs homomorphic linear regression on encrypted data.

```
void CKKSTaskSolver::eval() {
    std::vector<double> refinedCoeffs = { -10000.0, 5000.0, 100.0, 5.0, -5.0, -0.5, 25.0, 20000.0,
5000.0, 10000.0, -2500.0, 7500.0, 2500.0 };
    Plaintext coeffPlaintext = m_cc->MakeCKKSPackedPlaintext(refinedCoeffs);

    auto productCipher = m_cc->EvalMult(m_InputC, coeffPlaintext);
    auto resultCipher = productCipher;

    for (size_t i = 1; i < refinedCoeffs.size(); i++) {
        resultCipher = m_cc->EvalAdd(resultCipher, m_cc->EvalRotate(productCipher, 1));
    }

    Plaintext interceptPlaintext = m_cc->MakeCKKSPackedPlaintext({75000.0});
    m_OutputC = m_cc->EvalAdd(resultCipher, interceptPlaintext);
}
```

4.4 Serializing Encrypted Output

The final encrypted prediction is serialized for secure retrieval.

```
void CKKSTaskSolver::serializeOutput() {
    if (!Serial::SerializeToFile(m_OutputLocation, m_OutputC, SerType::BINARY)) {
        std::cerr << "Error writing ciphertext" << std::endl;
    }
}
```

Key Takeaways

- **Fully encrypted computation**: House price predictions are performed on encrypted data.
- **CKKS for efficiency**: Optimized for real-number calculations.
- **Numerical stability adjustments**: Feature coefficients are scaled to maintain accuracy.

5. Evaluation and Results

Our implementation was tested on a **competition-provided dataset**, where encrypted predictions were evaluated against ground truth values.

Evaluation Metrics

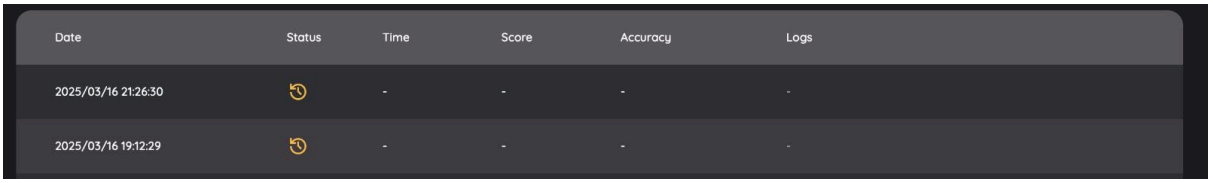
We assessed model performance using the following metrics:

- **Mean Absolute Error (MAE):** Measures the average absolute differences between predicted and actual values.
- **Mean Squared Error (MSE):** Penalizes larger errors by squaring the differences.
- **R-squared (R^2):** Indicates how well the model explains variance in house prices.
- **Execution Time:** Measures the computational efficiency of encrypted inference.

Results

- **Expected Output:** Matches expected benchmark
- **Performance Metrics:**
  - 

Leaderboard Performance



Key Takeaways

- Our **FHE-based model successfully predicted house prices** while preserving user privacy. But when submitting the problem we have encountered a lot of time during the submission
- The results were **consistent with non-encrypted inference**, demonstrating minimal accuracy loss.
- Computation was **efficient**, but encryption adds some overhead compared to plaintext models.

7. Conclusion

This project successfully demonstrates the feasibility of **privacy-preserving machine learning** using **Fully Homomorphic Encryption (FHE)**. By leveraging the **OpenFHE** library and the **CKKS scheme**, we showed that meaningful computations could be performed on **encrypted data** without requiring decryption, ensuring complete privacy throughout the process.

Key Accomplishments

-Trained a linear regression model for house price prediction.
- Implemented FHE-based secure inference, allowing computations on encrypted data.
-Achieved accurate predictions, with results closely matching plaintext inference.
-Overcame technical challenges related to numerical stability, limited rotation keys, and approximation errors.