Justin Morera
METIL Quickstart Internship - Project Report

Exploring CKKS Fully Homomorphic Encrypted Inferencing for Post-Quantum Resistant Machine Learning

Justin Morera

December 29, 2024

Abstract

This project examines the performance and runtime of a linear regression machine learning (ML) model on simulated manufacturing data to predict quality rating in comparison to a fully homomorphic encrypted (FHE) inferencing algorithm using the same test data and the trained model's weights as regression coefficients to show that FHE can be used to secure ML models against post-Quantum attacks while maintaining sufficient performance on a given task. A linear regression ML model was trained and evaluated on a synthetic dataset based on manufacturing process optimization tasks. The trained coefficients and intercept were taken from the model and used to perform encrypted inferencing via the CKKS encryption scheme. Predictions made from both the plaintext model and the encrypted inference were compared quantitatively using several metrics, as well as runtime and memory usage analyses. Experimental results show that the encrypted inference was just as effective as the non-encrypted model at the given task; however, the additional security provided by FHE comes at a cost of significantly higher runtimes due to preprocessing and computational overhead, as well as additional storage space. The findings serve as proof-of-concept for the use of Choen-Kim-Kim-Song (CKKS) FHE in Post-Quantum security for ML. The potential benefits and tradeoffs offered by CKKS FHE are discussed at the end of this paper.

1. Introduction

The goal of this project is to examine the effectiveness of Fully Homomorphic Encryption (FHE) as a potential tool for post-Quantum defense, specifically in regard to securing machine learning (ML) models. The role of ML is growing exponentially in the modern world as tasks become increasingly more complex and automation becomes more necessary to maintaining the status quo. The United States Department of Energy (DoE) offers guidelines on how to safely, ethically, and securely use artificial intelligence (AI) and ML tools in the DOE AI Risk Management Playbook (AIRMP) but does not explicitly address post-Quantum defensibility or the use of homomorphic encryption to such ends (n.d.). The findings presented here will hopefully add to the admission of FHE to the DoE's recommendations, as well as the department's overall practice as the world readies itself for widespread Quantum computing.

If the DoE can gain confidence in its ability to protect its models from adversaries, or even from internal misuse, then the department can make great strides into the modern technological age by automating tasks across the board using secure ML models to accomplish jobs otherwise done manually. The example introduced in this project is just one simplified example representing a single use case for ML in a real-world production scenario. It is presumable that more complex tasks requiring more sophisticated ML techniques will require more effort to encrypt and secure; but with these initial findings, it can be shown that such a feat is within reach, and can accelerate production and maintenance of the country's nuclear energy implementations with assurance for the future. As such, FHE, particularly the CKKS scheme, has been suggested to be post-Quantum resistant, as in Yu et al.'s 2021 research article, *Privacy-preserving computation in the post-quantum era*.

2. Methods

2.1 Dataset

A publicly available dataset from *Kaggle*, "Manufacturing Data for Polynomial Regression", uploaded by Ruken Missonier was used for this project (n.d.). The dataset consisted of synthetic data representing features measured from some manufacturing output. Each of the 3'957 input rows consisted of the following feature columns: Temperature in degrees Celsius, Pressure in kilopascals, an interaction column of the multiplication of Temperature and Pressure, Material Fusion Metric, and Material Transformation Metric. The latter two columns represent polynomial combinations of temperature and pressure, giving insight into fusion and transformation of manufacturing materials during production. The final column, the target for this experiment, is the Quality Rating, ranging from 0 to 1, which represents the overall percent quality of the produced material. Two other datasets were originally downloaded, but ultimately were not used due to a lack of applicable theming toward DoE operations; however, it is worth noting that the code can be tweaked slightly to allow for similar examinations of the data in these two datasets, and additional models can be trained to perform different tasks on each dataset.

2.2 Data Preprocessing

The input data was first preprocessed by analyzing the correlation coefficients of each feature column with the target column, Quality Rating, to understand each feature's direct correlation with the target value; all features except for Pressure (roughly 1% correlation) were found to have a significant relationship with the target column. A RobustScaler was applied to the data to reduce the effects of outliers on performance. To allow for the capture of non-linear patterns between features and the target column, as well as interactions between feature columns, polynomial features were added for each feature column pair, up to a degree of two; this brought the number of features up to 21 from the original 5.

2.3 Plaintext Model

A linear regression model was trained on 80% of the dataset (3'166 rows) and tested on the remaining 791 rows. The model was then evaluated on mean absolute error (MAE), mean squared error (MSE), root mean squared error (RMSE) and R^2-score (R2). Additionally, the runtime of all sections, including data preprocessing were tracked. Memory usage of the trained coefficients, intercept, and all plaintext input rows was also measured. Finally, the model's final residuals (the difference between predicted and actual values, were graphed and examined.

2.4 Encrypted Inference

Originally, it was planned that an entirely separate model was going to be trained and tested on encrypted data. Fortunately, after further research, it was discovered that the use of a machine learning model to accomplish regression tasks could be bypassed altogether during the encryption stage. The alternative method introduced is known as encrypted inference, and the benefits of using this technique are discussed later in the Discussion section. In this case, only the trained coefficients and intercept from the plaintext model (which could still be trained on synthetic data minimizing the patterns of a given DoE task) would need to be encrypted and stored for inferencing. Then, whenever a prediction needs to be made, a user would use the same private key to encrypt the input data, which likely contains sensitive information, and send it to the inferencing program. The encrypted output would be calculated using the trained coefficients, intercept, and input, and returned to the user. This result can then be decrypted with the private key.

Rather than training a separate model for a linear regression task, the coefficients and intercept trained from the original plaintext model were simply saved and encrypted using CKKS FHE. The same inputs used for the plaintext evaluation were also encrypted using the same encryption context. In a real-world scenario, this would be done in a separate, secure environment, and the private key generated from this context would be stored secretly. the context used a polynomial modulus degree of 8'192, coefficient modulo bit sizes of 60, 40, 40, and 60, and a global scale of 2^{40}. These proved to be sufficiently large values to allow for a proper noise budget and exceptional precision for the values used in this experiment without sacrificing security or increasing computational overhead unnecessarily. Since the plaintext model was a linear regressor, the inferencing step simply involved taking the dot product of the encrypted coefficients and the input features, then adding the encrypted intercept as a scalar. After inferencing, the predictions were then decrypted and the output values were evaluated on the same metrics as the plaintext model: MAE, MSE, RMSE, and R2. Additionally, the runtime of each step was tracked, including additional preprocessing such as context setup, encryption, and decryption time, and memory usage for the encrypted coefficients, intercept, and input rows was measured. Finally, the residuals between the actual and expected predictions were graphed and compared to the plaintext model's.

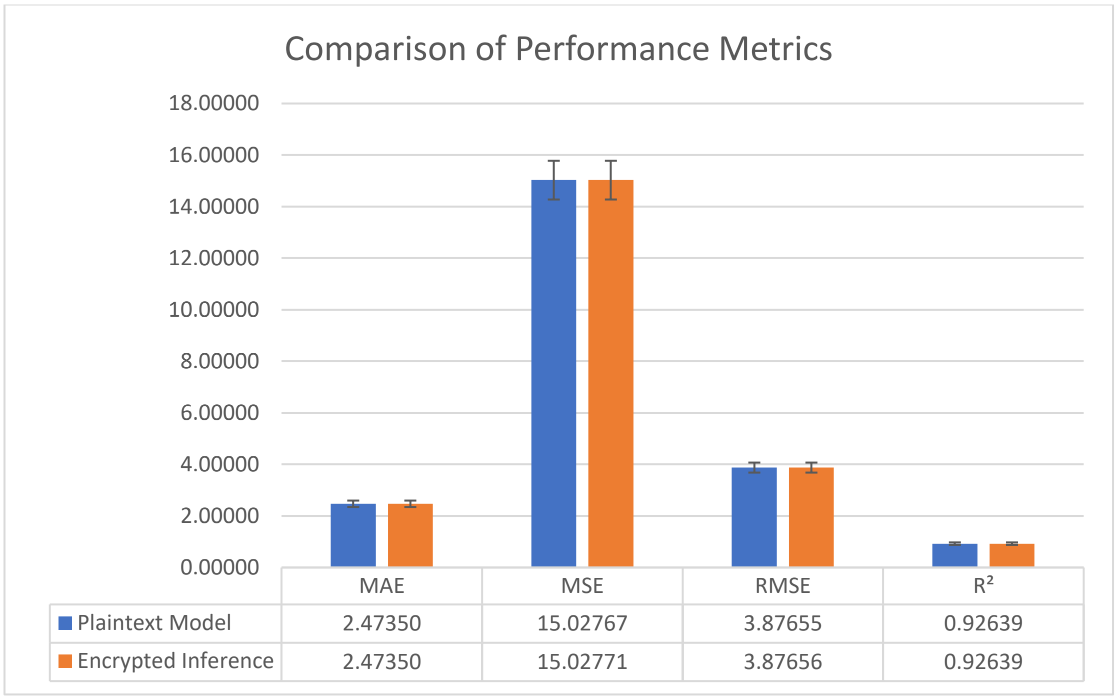
3. Results

3.1 Performance Metrics

The performance metrics for both methods could not have been closer. Both methods were able to explain roughly 92% of the variance in Quality Rating due to changes in the input features with R2 scores of 0.92. Errors for both methods were only off by about 2.5% on average, and the mean squared error and root mean squared error were ~15% and ~3.9%, respectively, for both methods. Metrics for both the plaintext model and encrypted inference differed by no more than a factor of 10^5, or 0.00028%. Table 1 shows the exact measurements for each metric. Figures 2-5 show the residuals measured from both methods, which are observably identical.

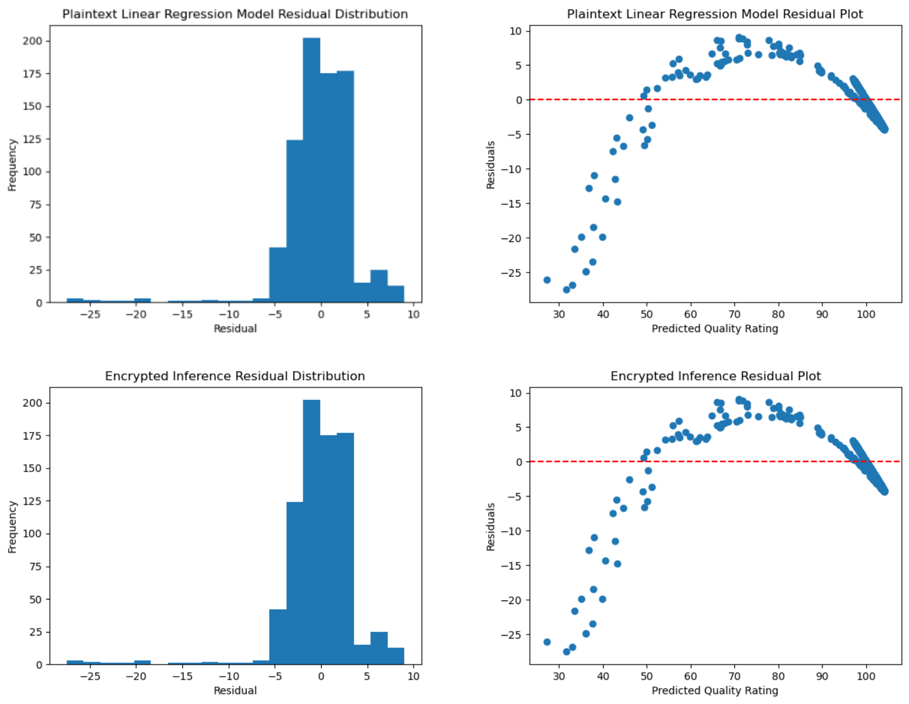
Performance Metrics				
Metric	**Plaintext Model**	**Encrypted Inference**	**Absolute Difference**	**Relative Difference**
MAE	2.47350	2.47350	1.11952E-06	0.00005%
MSE	15.02767	15.02771	4.27826E-05	0.00028%
RMSE	3.87655	3.87656	5.51812E-06	0.00014%
R^2	0.92639	0.92639	2.09564E-07	0.00002%

Table 1: Final MAE, MSE, RMSE, and R2 readings of both methods. The absolute difference measures the difference between each metric for each method, while the relative difference divides that amount by the plaintext model's value for that metric, which treats the plaintext model as the baseline.



	MAE	MSE	RMSE	R^2
🟦 Plaintext Model	2.47350	15.02767	3.87655	0.92639
🟧 Encrypted Inference	2.47350	15.02771	3.87656	0.92639

Figure 1: Side-by-side comparison of each metric for both methods via bar graph. Error bars are percentage error.



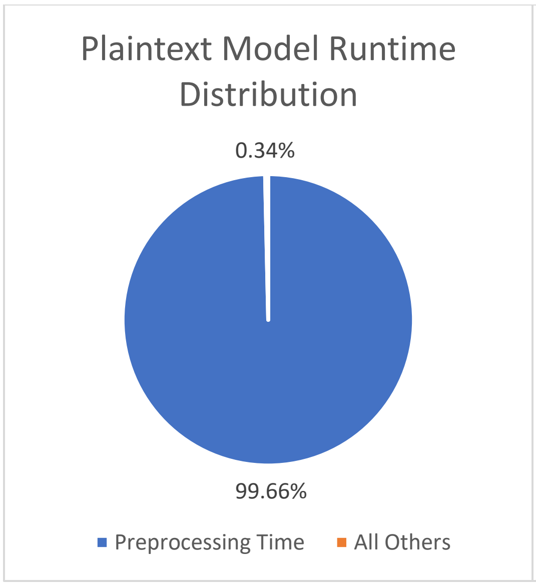
Figures 2-5: Histograms (left) and Scatterplots (right) of recorded residuals of plaintext model (top) and encrypted inference (bottom). Both respective graphs for each model appear identical due to the closeness of predicted values.

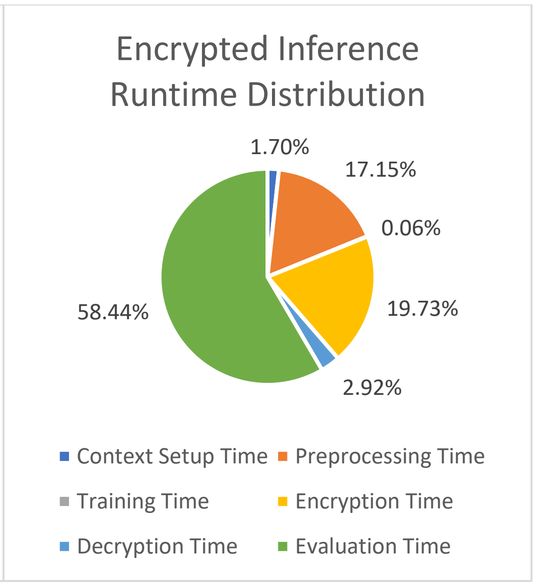
3.2 Runtime Analysis

The plaintext model had a total runtime of 4.16 seconds while encrypted inference took 24.17 seconds. The total runtime overhead caused by the additional preprocessing and computational overhead from FHE was roughly 6, meaning the encrypted inferencing method was at about 6 times slower than plaintext linear regression. When only accounting for the necessary stages for each method (training and testing for plaintext regression, and encryption, decryption, context setup, and the inferencing step for encrypted inference), the overhead was 1’405 slower. Preprocessing took 99.66% of the runtime for linear regression without encryption, but only 17.15% of the total runtime for encrypted inference. The inferencing step was the dominant stage in the encrypted pipeline, occupying 58.44% of the total runtime and 70.59% of the runtime when only accounting for stages unique to encrypted inferencing. Table 2 illustrates the full analysis of runtime values while Figures 6-8 portray the runtime distributions for both methods.

Runtime Analysis						
	Runtime (seconds)		Runtime (%)		Runtime Overhead	Runtime w/o Pre (%)
Step	Plaintext Model	Encrypted Inference	Plaintext Model	Encrypted Inference		Inference only
Context Setup Time		0.4109	0.00%	1.70%	N/A	2.05%
Preprocessing Time	4.1475	4.1475	99.66%	17.15%	1.0000	N/A
Training Time	0.0138	0.0138	0.33%	0.06%	1.0000	N/A
Encryption Time		4.7699	0.00%	19.73%	N/A	23.83%
Decryption Time		0.7062	0.00%	2.92%	N/A	3.53%
Evaluation Time	0.0004	14.1302	0.01%	58.44%	N/A	70.59%
Additional Overhead Time		5.8869	0.00%	24.35%	N/A	29.41%
Total Preprocessing Time	4.1475	10.0344	99.66%	41.50%	2.4194	29.41%
Total Without Initial Preprocessing	0.0142	20.0172	0.34%	82.79%	1405.6726	29.41%
Total Runtime	4.1617	24.1785	100.00%	100.00%	5.8097	100.00%

Table 2: Final runtime values for both methods. First six rows are individual stages of each pipeline while next Additional Overhead Time accounts for the sum of decryption, encryption, and context setup; Total Preprocessing Time accounts for the sum of initial preprocessing, feature selection, and, in the case of encrypted inference, Additional Overhead Time; and Total Without Initial Preprocessing accounts for only training and evaluation for plaintext method, and Additional Overhead Time plus inferencing step for encrypted method.





Figures 6 & 7: Pie charts showcasing runtime distributions for each stage for both methods. For the plaintext linear regressor (Figure 6), All Others includes training and evaluation. For Encrypted Inference (Figure 7), Training Time and Preprocessing Time are the same values as the plaintext linear regressor's while Evaluation Time is the encrypted inferencing step.

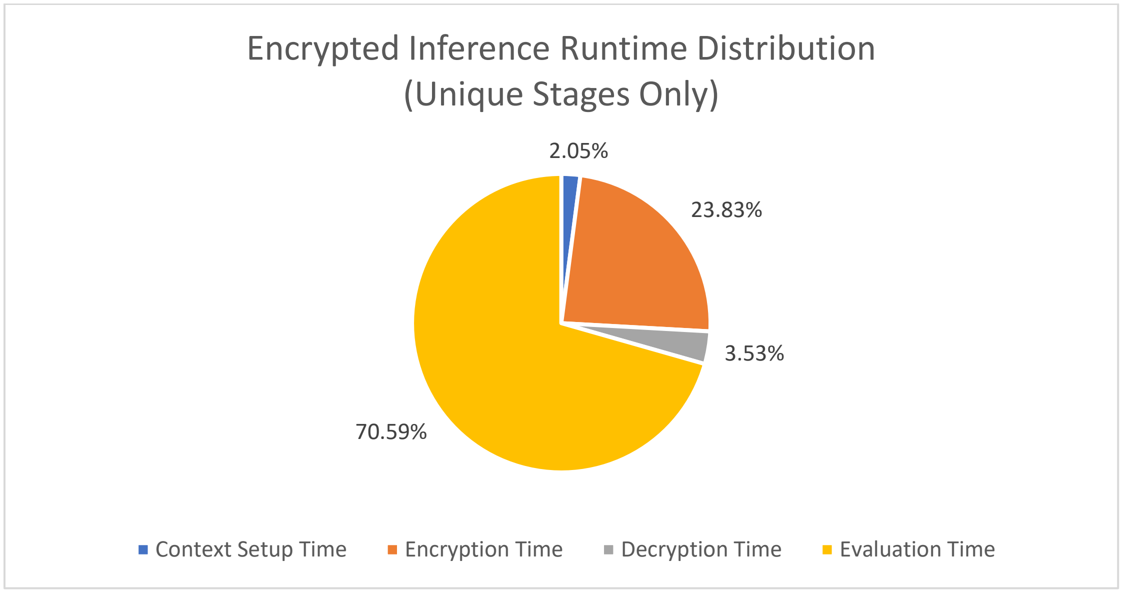


Figure 8: Pie chart showcasing the runtime distribution of only the stages novel to the encrypted inference method to show which unique stages dominate the runtime when not accounting for shared (and unavoidable) stages.

3.3 Memory Usage

The total memory usage for the coefficients, intercept, and inputs was measured to be 0.13 MB for the plaintext data and 253.06 MB for the encrypted data. Of these amounts, almost 100% of the size was due to the >3’000 inputs of 21 features each, as is to be expected. It is worth noting that encryption increased memory usage by almost 200’000% for the input features and the coefficients, but increased memory usage by 736’000% for the intercept; this discrepancy is likely due to the encryption overhead for each encrypted item, such as stored encryption keys and metadata. Even though the percent increase for the intercept, a scalar value, appears much larger, the overall distribution of memory sizes remained the same after encryption. Table 3 shows the full values of each data type and the graphed memory distribution can be examined in Figure 9.

Memory Usage					
	Size (MB)		% Increase	Size Distribution (%)	
Type	Plaintext Model	Encrypted Inference		Plaintext Model	Encrypted Inference
Coefficients	0.0002	0.3187	198840%	0.13%	0.13%
Intercept	0.0000	0.2245	735503%	0.02%	0.09%
Features	0.1269	252.5212	198905%	99.85%	99.79%
Total	0.1271	253.0644	199033%	100.00%	100.00%

Table 3: Final memory size measurements for both methods, in megabytes. Percent increases are extraordinarily large, but size distribution is relatively the same for both.

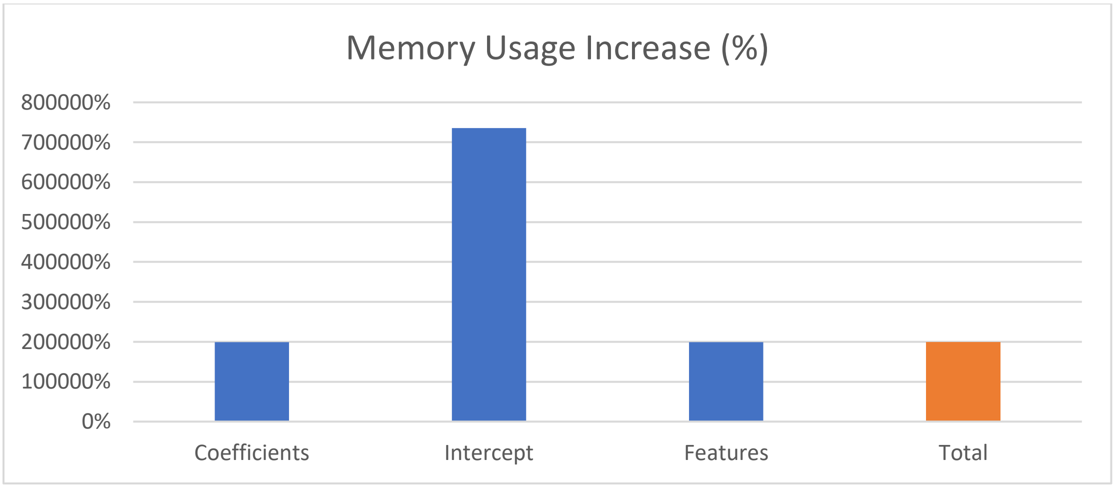


Figure 9: Bar graph showing percent increases of each data type. Intercept increase likely larger due to flat increase from encryption overhead compared to the negligible original size.

3.4 Regression coefficients

The trained regression coefficients of the model hold valuable insight into the patterns discovered by the linear regressor during training. Given the fidelity of predictions between encrypted inferencing and the original model, it is feasible to assume the coefficients remained largely untrained, but Table 4 shows that, without a doubt, the coefficients remain almost identical, differing by a factor of no more than 10^9. Additionally, it is helpful to understand that in this specific scenario, Material Transformation Metric, Temperature, and the interaction between those two features held the highest influence on the final predictions; likewise, the square of Temperature x Pressure, the interaction between Pressure and Temperature x Pressure, and the interaction between Temperature x Pressure and Pressure Material Fusion Metric held the least influence. This is helpful in real-world scenarios as it offers insight into the most sensitive features for the target variable, as well as help identify features to consider removing, likely to save space and time during computation, as seen above.

Regression Coefficients				
Feature	**Plaintext Coefficient**	**Encrypted Coefficient**	**Absolute Difference**	**Relative Difference**
Material Transformation Metric	-17854.32705	-17854.32705	1.36E-09	0.0000000000%
Temperature (°C)	14042.59546	14042.59546	1.91E-10	0.0000000000%
Temperature (°C) Material Transformation Metric	5513.877039	5513.877039	1.02E-09	0.0000000000%
Temperature x Pressure	4701.117549	4701.117549	3.89E-10	0.0000000000%
Temperature (°C)^2	4363.940246	4363.940246	1.93E-10	0.0000000000%
Pressure (kPa)	-4350.725523	-4350.725523	2.64E-10	0.0000000000%
Temperature (°C) Pressure (kPa)	-2201.599339	-2201.599339	2.51E-10	0.0000000000%
Material Transformation Metric^2	-1364.822336	-1364.822336	1.37E-09	0.0000000001%
Pressure (kPa) Material Transformation Metric	-427.6614786	-427.6614786	3.23E-09	0.0000000008%
Material Fusion Metric	-321.2333688	-321.2333688	8.45E-10	0.0000000003%
Material Fusion Metric^2	-201.7694551	-201.7694551	1.39E-09	0.0000000007%
Material Fusion Metric Material Transformation Metric	201.2629544	201.2629544	5.73E-10	0.0000000003%
Temperature (°C) Material Fusion Metric	185.0718711	185.0718711	1.57E-09	0.0000000008%
Pressure (kPa) Material Fusion Metric	151.5601083	151.5601083	4.48E-10	0.0000000003%
Temperature (°C) Temperature x Pressure	143.0183858	143.0183858	8.70E-11	0.0000000001%
Temperature x Pressure Material Transformation Metric	114.7607662	114.7607662	3.16E-11	0.0000000000%
Pressure (kPa)^2	10.59737628	10.59737628	2.66E-09	0.0000000251%
Temperature x Pressure Material Fusion Metric	-7.089485867	-7.089485867	2.17E-10	0.0000000031%
Pressure (kPa) Temperature x Pressure	6.209684236	6.209684235	9.99E-10	0.0000000161%
Temperature x Pressure^2	-0.292616816	-0.292616817	6.43E-10	0.0000002197%

Table 4: Table of the 21 coefficients, what feature patterns they are weighted toward, and the absolute and relative differences between both methods. The values are sorted by absolute value of coefficient in descending order, thus the most influential feature patterns are at the top and the least are on the bottom.

4. Discussion

As shown from the results of this experiment, FHE is more than capable of performing the same task as a trained ML model on a real-world task; however, it is clear that the increased security comes with some tradeoffs. With significant expansion of memory usage, larger and more complex tasks become more difficult or less worthwhile to perform with FHE. This demonstrates the need to manage data and analyze patterns thoroughly before encryption is utilized on a broader scale to lessen overhead. It should also be noted that FHE is a relatively new technique, and is constantly being improved as its capabilities, and our understanding of them, grow (FHE.org, n.d.). Given the importance of security in an ever-changing world, it is necessary to adapt to using these up-and-coming strategies in order to keep up with potential adversaries. Of course, this comes with the caveat of ensuring that the new technologies are proven to be as secure as they are theorized to be.

Similarly, it cannot be overlooked that ML and FHE can only be utilized effectively in conjunction with proper processing of data. Initially, the linear regression model used third degree polynomial features and interactions, which allowed the performance, particularly the R2 score, to improve even further; however, when attempting to perform encrypted inference with the same features, not only was the overhead much higher given the exponential increase in features, but the simplistic linear regression formula failed to capture these patterns and yielded atrocious results. It is possible that multiple encrypted inferencing equations could have been used for each degree of features to maximize predictions, but the easier, and ultimately most beneficial, solution was to remove the cubic features and stick with the linear regression inference, a dot product followed by a scalar addition. Even though the loss of cubic features resulted in a minor drop in the plaintext model's performance, it allowed the encrypted inference to mimic that performance flawlessly, as well as significantly lowered memory usage and runtime due to computational overhead.

5. Conclusion

As always, further research is required to fully understand the extent of FHE's ability to secure data in a post-Quantum world; fortunately, the scientific community is presently pursuing goals related to the continuous development of FHE as a groundbreaking security tool. Institutions such as IBM are already projecting how influential the different FHE schemes will become (n.d.), and the applications FHE has on ML alone cannot be overstated. As both ML and FHE techniques continue to develop, so does the complexity of issues faced by nations; thus, it is important, and cost-effective, to proactively begin utilizing these revolutionary technologies, where applicable.

It is also feasible that other nations are currently looking into the most secure and efficient ways to utilize ML and AI for complex military and industrial tasks, and it is likely that they hold FHE in high regard for this purpose, given the promise it holds for flexible and robust security. But even when only considering the scope of applying FHE to ML, the benefits far outweigh the added costs of applying this security scheme. There is room for ML models to excel in tasks far beyond linear regression, freeing time and resources for other tasks that cannot yet be automated without sacrificing security. If models are allowed to be developed on government systems without compromising national security, even if these models are implemented in a gradual manner with minimal risk, there is likely to be a marked increase in productivity and cost savings across multiple departments. Let this project stand as one of the first steps toward exemplifying how such methods can accomplish this goal.

6. Implementation

6.1 Code

GitHub: https://github.com/JustinMorera/METIL-Quickstart-Internship

6.2 Dependencies

- Python 3.11.15
- Python packages: (pip install _____)
  1. pandas
  2. sklearn
  3. matplotlib
- Tenseal Python package:

  https://github.com/OpenMined/TenSEAL?tab=readme-ov-file#installation

  Installation:

  pip install tenseal

  python import tenseal as ts

7. References

El Kharoua, R. (n.d.). Predicting manufacturing defects dataset [Dataset]. Kaggle. https://www.kaggle.com/datasets/rabieelkharoua/predicting-manufacturing-defects-dataset

Ely, N. (n.d.). Predictive modeling for defect reduction [Computer code]. Kaggle. https://www.kaggle.com/code/nileshely/predictive-modeling-for-defect-reduction

FHE.org. (n.d.). History of fully homomorphic encryption. FHE.org. https://fhe.org/history/

IBM Research. (n.d.). FHE: The future of cloud security. IBM Research Blog. https://research.ibm.com/blog/fhe-cloud-security-hE4cloud

OpenMined. (2021). TenSEAL: A library for doing homomorphic encryption operations on tensors [GitHub Repository]. GitHub. https://github.com/OpenMined/TenSEAL

Missonnier, R. (n.d.). Manufacturing data for polynomial regression [Dataset]. Kaggle. https://www.kaggle.com/datasets/rukenmissonnier/manufacturing-data-for-polynomial-regression

U.S. Department of Energy. (n.d.). DOE AI Risk Management Playbook (AIRMP). U.S.Department of Energy. https://www.energy.gov/ai/doe-ai-risk-management-playbook-airmp

Yu, Y., & Xie, X. (2021). Privacy-preserving computation in the post-quantum era. National Science Review, 8(9), nwab115.