ELEMENTS OF CRYPTOGRAPHY

Credit Card Fraud detection using ML and HE

Group Members

Meet Bhatt (202001267)
Vraj Chaudhari (202003006)
Jay Patel (202003019)
Kunjera Shreyansh (202003031)

CONTRIBUTION:

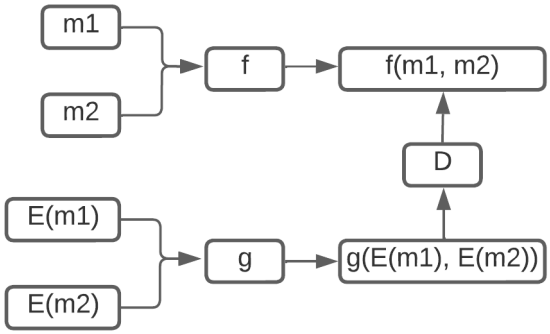
- **Meet Bhatt:** Gathering the information related to our encryption scheme, Analysis on Dataset, Creating SRS.
- **Chaudhari Vraj:** Analysis on Dataset, Implementation of code, Creating PPT, Report and Video.
- **Jay Patel:** Analysis of encryption scheme, Implementation of code, Creating PPT, Report and Video.
- **Shreyansh Kunjera:** Analysis on Dataset, Implementation of code, Creating PPT, Report and Video.

1 Introduction

In the financial sector, detecting credit card fraud is a major issue that may be solved in a number of ways, one of which is by using machine learning algorithms. The need to secure sensitive financial data from unauthorised access is a barrier when applying machine learning algorithms for fraud detection. A cryptographic method named as homomorphic encryption enables us to do computations on encrypted data without having to first decrypt it, protecting the confidentiality and security of the sensitive data.

2 Homomorphic Encryption

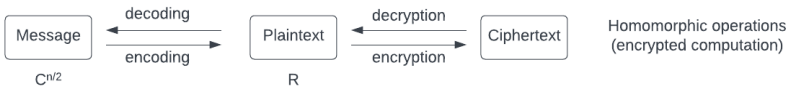
Homomorphic encryption(HE) is an encryption scheme which allows us to perform mathematical operations on encrypted data. In other words, HE allows computation on ciphertext, while maintaining the confidentiality of the plaintext (unencrypted data).



Homomorphic encryption can be applied in various scenarios, such as secure computation in cloud computing, privacy-preserving data analysis, and secure multiparty computation. It has the potential to enable secure computation on sensitive data without the need to reveal the data in plaintext, which is particularly useful for protecting user privacy.
There are various types of HE encryption techniques, such as fully homomorphic encryption (FHE) and partially homomorphic encryption (PHE). FHE allows arbitrary computation on encrypted data, while PHE only enables certain types of computations, such as addition or multiplication. However, FHE is currently less efficient and more computationally expensive than PHE.
HE can be used in machine learning to enable secure computation on sensitive data while preserving the confidentiality of the user data. This is particularly useful in scenarios where multiple parties want to collaboratively train a ML model without revealing their users private data to each other.

3 CKKS

CKKS is not a perfect HE(Homomorphic Encryption) scheme. It is an Approximate HE. The goal of ckks is to find the efficient approximation of computation on HE. The key concept is to interpret encryption noise as an error that can happen when performing approximate computations. In other words, if a message is encrypted using the secret key sk, the decryption structure will be of the type $\langle c, s_k \rangle = m + e (\text{mod } q)$, where e is a minor error added to ensure the security of the hardness assumptions. If e is sufficiently small in relation to the message, this noise is unlikely to affect the significant figures of m, and the approximation arithmetic value $m' = m + e$ can take the place of the original message. Before encryption, one can multiply the message by Δ(scaling factor) to decrease the precision loss caused by encryption noise.



In order to ensure that the result is smaller than q(ciphertext modulus) for homomorphic operations, we keep our decryption structure small enough in relation to the q. The fact that a message's bit size increases exponentially with circuit depth without being rounded remains a problem, nevertheless. We propose a novel ciphertext message manipulation method termed as rescaling to solve this issue.

We have used Microsoft's Tenseal library for Homomorphic Encryption whose brief introduction is given below

3.1 Microsoft Tenseal Library

We have used TenSEAL to implement CKKS encryption. TenSEAL is used to perform HE operations on tensors. It is constructed on top of Microsoft Tenseal, a C++ library that implements the HE algorithms BFV and CKKS. TenSEAL is a C++ library with a Python interface that offers simplicity through a Python API while maintaining efficiency by carrying out the majority of its operations in C++.

3.2 Why to use CKKS ?

One must manage enormous amount of data collections because a lot of data is generated in a cloud computing environment. Our technique may be a useful tool for data analysis, due to its ability to encrypt a huge amount of data into a single ciphertext and parallelize both computation and space. For instance, we used a batching technique to enhance the evaluation of the logistic function.

3.3 Encoding and Decoding

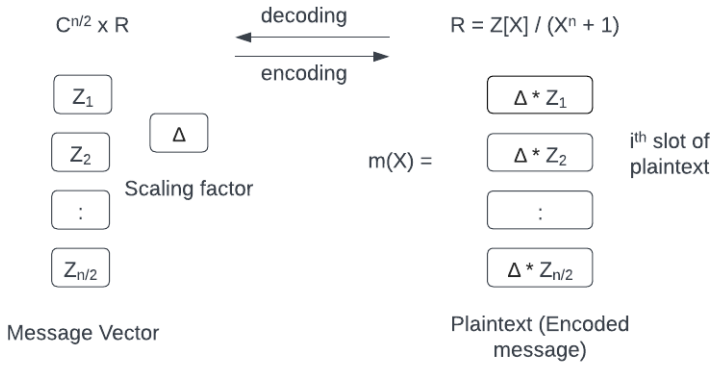
For effective homomorphic computation, a vector of several plaintexts must inevitably be encrypted in a single ciphertext. A ring isomorphism could be used to decode a plaintext polynomial as a vector of plaintext values into a product of finite fields. After performing homomorphic operations, an inserted error is positioned apart from the plaintext space so it can be eliminated using the plaintext characteristic.

The plaintext of our approach, on the other hand, is a component of a cyclotomic ring with the property zero, and it includes a minor error that is either added during encryption to assure security or occurs during approximation arithmetic. Therefore, we adapt the complex canonical embedding map, an isometric ring homomorphism. In order to prevent a little inaccuracy in a plaintext polynomial from having a big impact on the encoding and decoding processes, it maintains the size of polynomials.

Ecd$(z; \Delta)$. Let $z = (z_i)_{i \in T}$ be a $(N/2)$-dimensional vector of complex numbers, the encoding procedure expands it into the vector $\pi^{-1}(z) \in \mathbb{H}$ and computes its discretization to $\sigma(\mathcal{R})$ after multiplying Δ (scaling factor). Return integral polynomial

$$m(X) = \sigma^{-1}\left(\lfloor \Delta \cdot \pi^{-1}(z) \rfloor_{\sigma(\mathcal{R})}\right) \in \mathcal{R}$$

Dcd$(m; \Delta)$. Let $m \in \mathcal{R}$ be an input polynomial, return the vector $z = \pi \circ \sigma\left(\Delta^{-1} \cdot m\right)$, i.e., the j^{th} entry in z is $\in T$ is $z_j = \Delta^{-1} \cdot m\left(\zeta_M^j\right)$.



We can encrypt several messages in a single ciphertext using the batching mechanism in the HE system, which also allows for parallel processing. In real world, we use it as an advantage to parallelize computations and decrease their complexity and memory requirements.

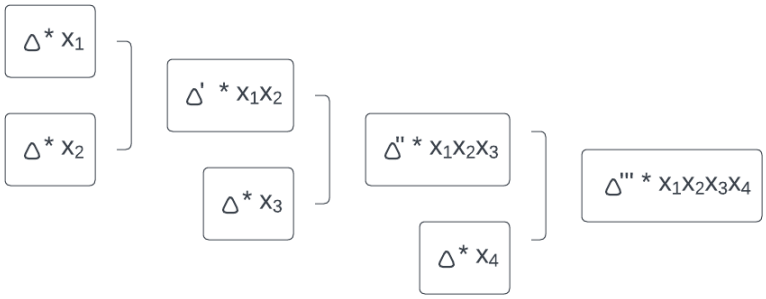
3.4 Encrypt and Decrypt

Constructing a leveled HE scheme for approximate arithmetic is the purpose of this section. For convenience, we fix a modulus q_0 and a base $0 < p$, and let $q_\ell = p^\ell \cdot q_0$ for $0 < \ell \leq L$. p is used as a base for scaling. For a security parameter λ, we choose a parameter $M = M(\lambda, q_L)$ for cyclotomic polynomial. For a level $0 \leq \ell \leq L$, a ciphertext of level ℓ is a vector in $\mathcal{R}_{q\ell}^k$ for a fixed integer k. Our cryptographic scheme has five algorithms (Addition, Multiplication, KeyGeneration, Encryption, Decryption). We will be describing a HE scheme over polynomial ring $\mathcal{R} = \mathbb{Z}[X]/(\Phi_M(X))$

- **KeyGeneration** $\left(1^\lambda\right)$. Creates a secret key s_k, a public key p_k, and a evaluation key evk.
- **Encryption**$_{pk}(m)$. For a given polynomial $m \in \mathcal{R}$, return a ciphertext $c \in \mathcal{R}_{q_L}^k$. An encryption c of m will satisfy $\langle c, sk\rangle = m + e \,(\mathrm{mod} q_L)$ for some small e.
- **Decryption**$_{sk}(c)$. For a ciphertext c at level ℓ, output a polynomial $m' \leftarrow \langle c, sk\rangle \,(\mathrm{mod} q\ell)$ for s_k.
- **Addition** (c_1, c_2). are encryptions of m_1 and m_2, output an encryption of $m_1 + m_2$.
- **ReScaling**$_{\ell\to\ell'}(\mathbf{c})$. Let $c \in \mathcal{R}_{q\ell}^k$ be a ciphertext, at level ℓ and ℓ'a lower level, output $c' \leftarrow \left\lfloor \frac{q\ell'}{q_\ell} c \right\rceil$ in $\mathcal{R}_{q\ell'}^k$, i.e., c' is a ciphertext obtained by scaling $\frac{q_{\ell'}}{q_\ell}$ to the entries of c and rounding the coefficients to the closest integers. We will omit the subscript $\ell \to \ell'$ when $\ell' = \ell - 1$.

3.5 Rescaling

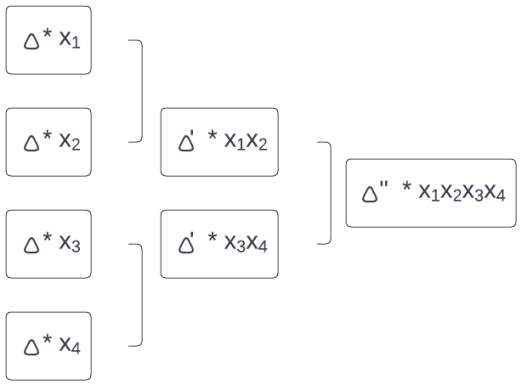
The rescaling process produces a ciphertext $\left\lfloor p^{-1} \right\rceil * c \,(\mathrm{mod} \frac{q}{p})$, which is a valid encryption of $\frac{m}{p}$ with noise approximately $\frac{e}{p}$, for an encryption c of m such that $\langle c, s_k\rangle = m + e \,(\mathrm{mod}\ q)$. Similar to the rounding step in fixed/floating-point arithmetic, it decreases the size of the ciphertext modulus and thus eliminates the error found in the LSBs of messages, practically maintaining the precision of plaintexts.



3.6 Leveled HE

Ciphertexts might be at different levels while doing Homomorphic Operations. Before performing homomorphic operation, we should reduce a ciphertext's level from a bigger level l to a smaller level l' when the encryptions c, c' of given messages m, m' belong to distinct levels l and l'. There are two contenders: the ReScaling technique and simple modular reduction. Since the ReScaling procedure changes m to $\frac{q_{l'}}{q_l}m$, while the modular reduction $c \mapsto c \pmod{q_{l'}}$ preserves the plaintext, it should be chosen very carefully. Unless otherwise mentioned, we compute on ciphertexts at various levels using simple modulus reduction to the smaller modulus throughout this study.

Instead of doing linear multiplication of encrypted data we should perform the multiplication in a tree based structure. In linear multiplication we will need to perform $(n-1)$ rescaling operations for n multiplications on encrypted data. Instead we can perform same number of multiplications in $(\log n + 1)$ rescaling operations using tree based structure.



4 Credit Card Fraud Detection using HE

4.1 About Dataset

In our credit card dataset, we have total 284808 rows and 31 columns from which, only transaction and amount columns are original, due of privacy concerns, all additional columns are scaled. The transactions aren't very big, really. The average of all the mounts produced is about 88. There are only 492 instances of Fraud transactions that is 0.17%, all other transactions are Non-Fraud transactions that is 99.83%. That also means that dataset is heavily imbalanced. Scaling has done by using PCA transformation(Dimensionality Reduciton Technique).

V10	...	V22	V23	V24	V25	V26	V27	V28	Class	scaled_amount	scaled_time
0.090794	...	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053	0	1.783274	-0.994983
-0.166974	...	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724	0	-0.269825	-0.994983
0.207643	...	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752	0	4.983721	-0.994972
-0.054952	...	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.061458	0	1.418291	-0.994972
0.753074	...	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.215153	0	0.670579	-0.994960
...	...	...	...	...	...	...	...	...	...	...	...
4.356170	...	0.111864	1.014480	-0.509348	1.436807	0.250034	0.943651	0.823731	0	-0.296653	1.034951
-0.975926	...	0.924384	0.012463	-1.016226	-0.606624	-0.395255	0.068472	-0.053527	0	0.038986	1.034963
-0.484782	...	0.578229	-0.037501	0.640134	0.265745	-0.087371	0.004455	-0.026561	0	0.641096	1.034975
-0.399126	...	0.800049	-0.163298	0.123205	-0.569159	0.546668	0.108821	0.104533	0	-0.167680	1.034975

4.2 Preprocessing

Since our dataset is very imbalanced, if we use this dataset for model training, it will not give very accurate results and can get many errors. Since most of them are non-fraud, Our model will operate on the presumption that the majority of transactions are non-fraud, thus it won't be able to spot the patterns that indicate a transaction is fraudulent.

4.2.1 Scaling:

We'll start by scaling the time and money column, which needs to be scaled similarly to the other columns. Because these values have a large range of values due to which the prediction will not be accurate so we need to scailing them in small range. Below is the code for scailing.

CODE:

4.2.2 Distributing:

We will create a sub-sample on which we can do model training to avoid overfitting, wrong correlations and imbalance. For that we will take 5:5 ratio of (Same amount) both class).

```
rob_scaler = RobustScaler()

data['scaled_amount'] = rob_scaler.fit_transform(data['Amount'].values.reshape(-1,1))
data['scaled_time'] = rob_scaler.fit_transform(data['Time'].values.reshape(-1,1))

data.drop(['Time','Amount'], axis=1, inplace=True)
data
```

CODE:

```
[ ]  print('Distribution of the Classes in the subsample dataset')
     print(new_data['Class'].value_counts()/len(new_data))
```

4.2.3 Random Under Sampling:

To create a sub sample data frame, we will use random under sampling, in which we will choose the class which have less instances and will take the same amount of instances of other class.
So we will create a new sub-sample by concat 500 cases of 1st class and 500 transaction of 2nd class and then will shuffle the data. **Risk: Information Loss.**
CODE:

```
[ ]  fraud_data = data.loc[data['Class'] == 1]
     non_fraud_data = data.loc[data['Class'] == 0][:492]

     normal_distributed_data = pd.concat([fraud_data, non_fraud_data])

     new_data = normal_distributed_data.sample(frac=1, random_state=42)

     new_data.head()
```

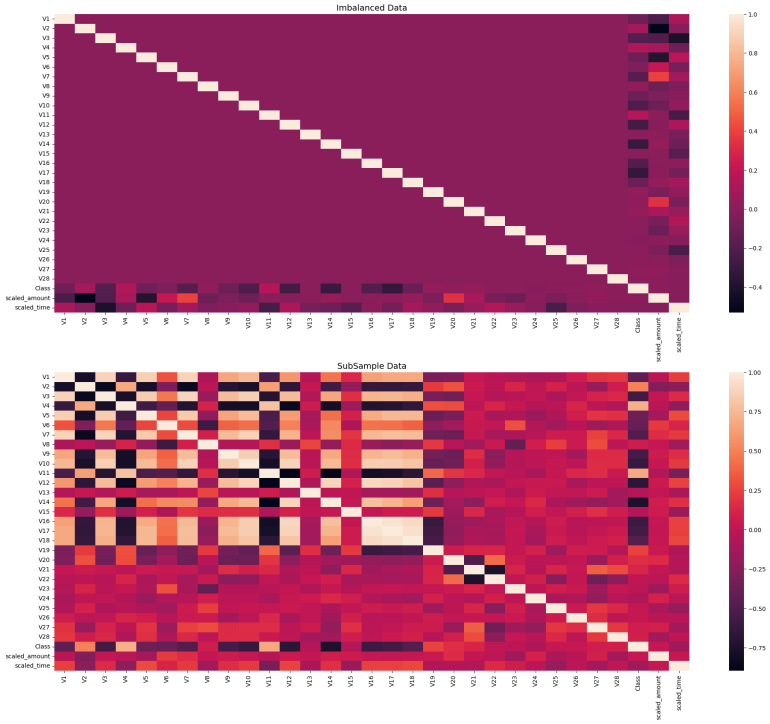
4.3 Analysis (After Preprocessing)

After preprocessing, we will observe the correlation between features and will also see the boxplots of theirs.

4.3.1 Correlation Matrices:

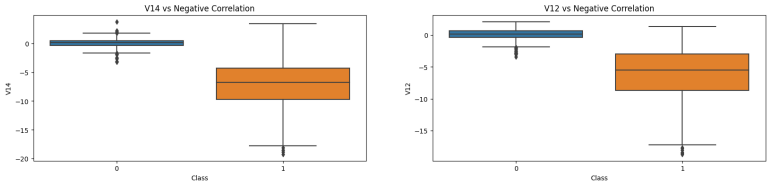
By using correlation matrices, we can observe that which features are heavily related with transaction being fraud. So we will use this matrix in sub-sample data frame and will see which features are positively correlated with fraud transactions and which are negatively correlated with fraud transactions.
We have observed that features V14 and V12's correlation is negative with fraud transactions so as the value of these features will be lower, the higher chance that transaction will be fraud.

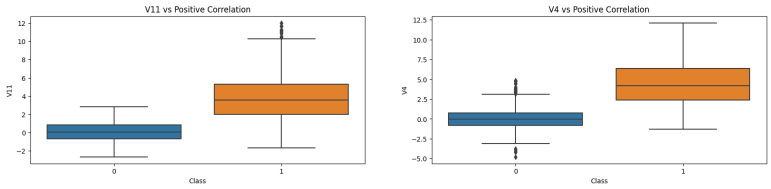
We have also observed that V4 and V11's correlation is positive with fraud transactions so as the value of these features will be higher, the higher chance that transactions will be fraud.



4.3.2 Box Plots:

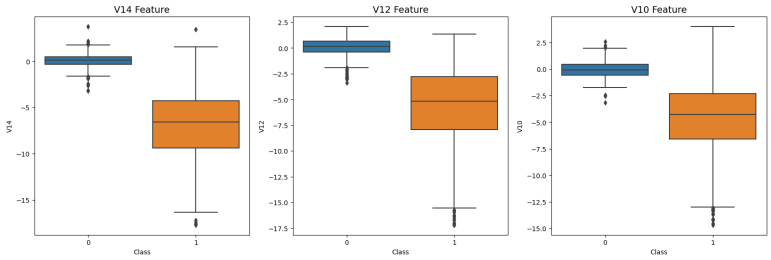
Box plots help us comprehend the features and how they are distributed in both fraudulent and legitimate transactions.
Here are some of the box plots of some features with class negative and class positive correlation.





4.3.3 Outlier Removal:

We need to remove the outliers from all the features which have very great amount of positive or negative correlation with fraud and non-fraud classes. By using this we can improve our accuracy of the model. For this we can use IQR to calculate the 25th and 75th percentile and can remove the transactions which are beyond the 25th and 75th percentile as an extreme outliers.



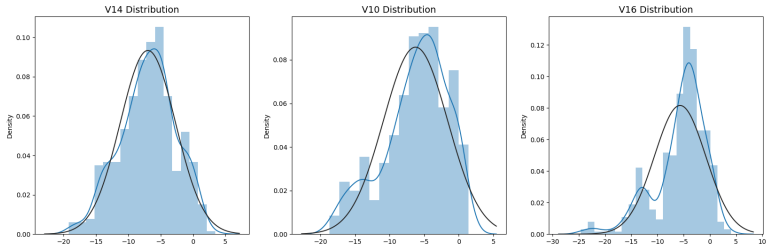
Tradeoff is that We must be careful when deciding how far to raise the outlier removal threshold. The threshold is calculated by multiplying a value (for example, 3/2) by the inter-quartile range.On the other hand if multiplied by a larger value, such as three, it will discover fewer outliers; conversely, if this threshold is lower, it will detect more outliers and if it will detect more outliers, it might be risky because by doing that we can also lose some important pieces of information and by doing that we will have lower accuracy.

4.3.4 Distribution plot:

The distribution of the characteristic we will use to weed out some of the outliers is first visualised. Only feature V14 shows a Gaussian distribution in comparison to features V12 and V10.

4.3.5 Threshold:

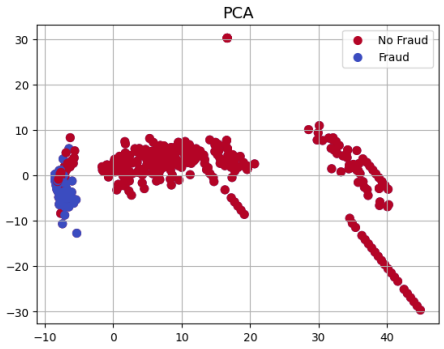
The upper thresholds will be determined by adding the 75th quantile and the lower thresholds will be determined by subtracting 25th quantile after choosing the value that will be multiplied by the iqr.
We have also plot the graph that shows that after doing this, the number of outliers have decreased with considerable amount.



We have also observed that our accuracy has also i by 3% though some outliers have distort the accuracy of out model.

4.3.6 Clustering and PCA:

We will use PCA (Dimensionality Reduction) to reduce the dimension of our data and then we will use clustering to see how well our model will separate two classes(fraud and non-fraud).
We have observed that PCA has done very good clustering of both classes in our dataset. Even though the sub-sample was pretty small. So we can also tell that further models that we will apply will also perform well.



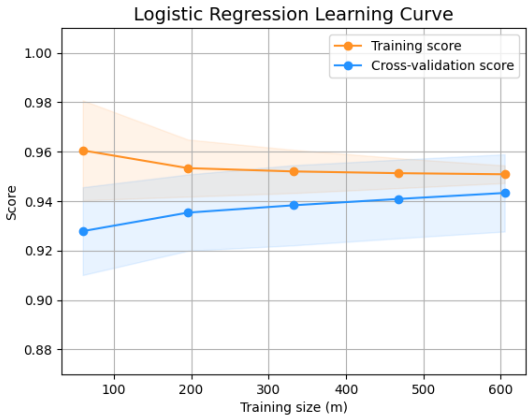
4.4 Model Training (Logistic Regression):

For training out data set, we have used Logistic Regression model.Before that,Additionally, the dataset has been divided into training and testing sets. In order to anticipate it, we have additionally separated the labels from other features.

4.4.1 Observation:

We have observed that, our model is more likely to be overfitted (have a high variance) if the training score and the cross-validation score differ significantly. Our model is underfitting (high bias) if the score is low in both the training and cross-validation sets.

The results of logistic regression are excellent in both the cross-validating as well as training sets.
Following is the plot of the **Learning curve of Logistic regression.**



4.4.2 Confusion matrix:

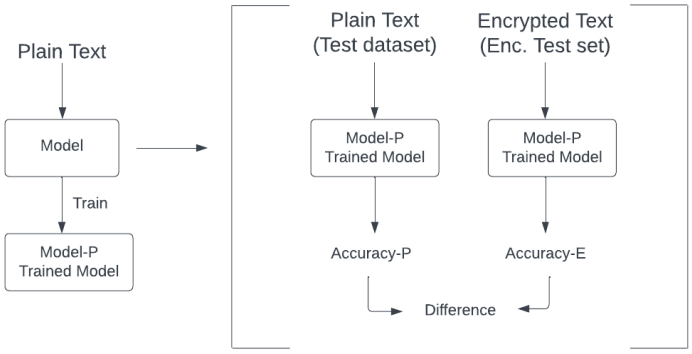
True Positive(tp): Transactions Classified As Fraud Correctly
False Positive(fp): Incorrectly Classified Fraud Transactions
True Negative(tn): Transactions Classified Non-Fraud Correctly
False Negative(fn): Incorrectly Classified Non-Fraud Transactions
Precision: tp / (tp + fp)
Recall: tp / (tp + fn)

Precision and Recall Tradeoff: Our model will find fewer examples as it becomes more Precise(selective). Example: Let's assume that the model has a 95% accuracy rate, let's say that there are only 5 instances of fraud for which the model has an accuracy rate of 95% or higher. If we lower the precision, our model will be able to discover more cases, such as the additional 5 cases that our model believes to be 90% fraud cases.

5 Encryption Process

5.1 Flowchart of model using plain data

The below image shows the whole process of how and where the model is trained and where we are doing the encryption.



5.2 Logistic Model on Plain Train_Set

- First we have trained our model on plain text 5 times using Logistic Regression Gradient Descent.
- After that we have calculated the loss at each iteration and observed that after each iteration the loss was decreasing because each time when we were getting parameters, we were using it as initial value for next iteration.

```
Loss at epoch 1: 0.7951291799545288
Loss at epoch 2: 0.30758434534072876
Loss at epoch 3: 0.2514585256576538
Loss at epoch 4: 0.2131306231021881
Loss at epoch 5: 0.18765997886657715
```

- Then after training the model we have calculated the accuracy on plain test set.

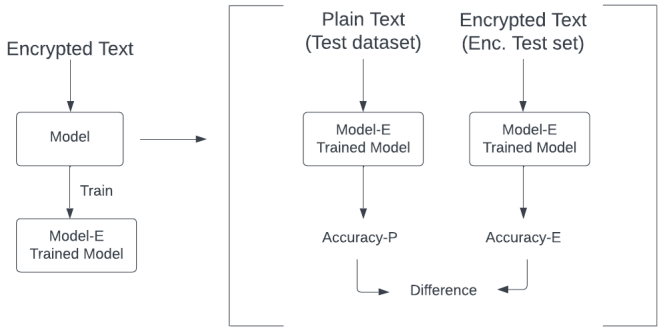
```
Accuracy on plain test_set: 0.9197080135345459
```

- Now we have encrypted the test data and calculated the accuracy on cipher test set

```
Accuracy: 235/274 = 0.8576642335766423
```

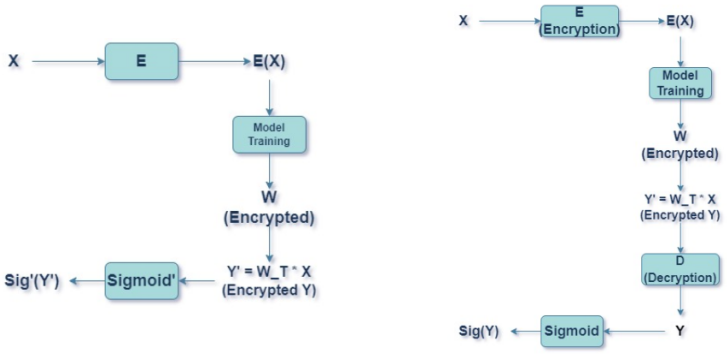
- Difference between the accuracy of plain and cipher test set

```
Difference between plain and encrypted accuracies: 0.06204378604888916
```



5.3 Flowchart of Model using Encrypted Data

5.4 Two types of process happening in Model



- Here first of all we have encrypted our train set.
- Then we have trained our model on cipher train set 5 times using Logistic Regression & Gradient Descent.
- After getting the optimal parameter values we have calculated the accuracy of cipher test set using different sigmoid function and then using the parameters we have predicted the class on plain test set and decrypted the result because those are encrypted results
- And after decrypting the result we have calculated the accuracy and found the difference between the accuracies

```
Difference between plain and encrypted accuracies: 0.470802903175354
```

Here are the references (Nug22, ug22) , (CKKS17, KKS17)

References

Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23*, pages 409–437. Springer, 2017.

David Nugent. Privacy-preserving credit card fraud detection using homomorphic encryption. *arXiv preprint arXiv:2211.06675*, 2022.