ETHICAL CONSIDERATIONS FOR FRAUDENTIAL EMAIL DETECTION

Advaith Shyamsundar Rao asr209
Falgun Malhotra fm466
Hsiao-Chun Hung hh617
Vanshita Gupta vg422

ABSTRACT

In today's data-driven landscape, the detection of fraudulent emails within corporate communications is critical. With email communication still being the most used mode of communication in organizations, hackers over time have found creative ways to bypass several security layers. In 2022 alone, email-based scams have led to losses of over $2.7 billion.

Over the last few years, Machine Learning as well as Deep Learning models, specifically Transformer-based models have enabled remarkable advancements in Natural Language Understanding, making them a great choice for tasks such as text classification and generation. These modeling techniques have the potential to enable high-scale and high-efficacy Fraud Detection. However, with deeper neural network-based architectures and models pre-trained on huge amounts of data, privacy concerns loom larger, making it imperative to ensure data protection while maintaining the integrity of the analysis.

1 DATASET DESCRIPTION

The project makes use of a rich source of public email communication, the Enron email dataset. In 2000, Enron was one of the largest companies in the United States. By 2002, it had collapsed into bankruptcy due to widespread corporate fraud. The data has been made public and presents a diverse set of email information ranging from internal, marketing emails to spam and fraud attempts.

In the early 2000s, Leslie Kaelbling at MIT purchased the dataset and noted that, though the dataset contained scam emails, it also had several integrity problems. The dataset was updated later, but it becomes key to ensure privacy in the data while it is used to train a deep neural network model.

Though the Enron Email Dataset contains over 500K emails, one of the problems with the dataset is the availability of labeled frauds in the dataset. Label annotation is done to detect an umbrella of fraud emails accurately. Since, fraud emails fall into several types such as Phishing, Financial, Romance, Subscription, and Nigerian Prince scams, there have to be multiple heuristics used to effectively label all types of fraudulent emails.

To tackle this problem, the following heuristics have been used to label the Enron data corpus using email signals, and automated labeling has been performed using simple ML models on other smaller email datasets available online. These fraud annotation techniques are discussed in detail in section 3 below.

To perform fraud annotation on the Enron dataset as well as provide more fraud examples for modeling, two more fraud data sources have been used,
1) Phishing Email Dataset: https://www.kaggle.com/dsv/6090437
2) Social Engineering Dataset: http://aclweb.org/aclwiki

To perform high-quality testing of the fraud detector, two gold label sets have been used,
1) **Gold Fraud Set**: Contains 1000 curated fraud emails from the phishing and social engineering dataset. The testing results successfully answer the question - how many fraud emails does the model miss out on?
2) **Sanity Set**: Contains 250000 curated internal email communication emails between employees at Enron. The testing results successfully answer the question - how many times does the model pick up any non-fraud email that it is not supposed to flag as fraud?

	Fraud	Non-Fraud
Enron Dataset	2327	445090
Phishing Dataset	4976	12515
Social Engineering Dataset	4160	6475

Table 1: Table represents the breakdown of the fraud vs non-fraud labels for each dataset

Above is a short data summary of the label distribution across different sources (on the x-axis) and labels (on the y-axis).

To tackle data imbalance between the fraud and non-fraud classes, Data Augmentation has been performed, creating 9 synthetic emails for every 1 fraud email. The augmentation process has made use of techniques such as,
1) Synonym Replacement
2) Stopword Removal
3) Swapping Noun Phrases

2 DATA PREPROCESSING

As part of preprocessing the project performed 4 types of data cleanup operations,
1) **Link Removal**: Several types of links were removed from the email body such as URLs and href tags and were replaced with a common <URL> token.
2) **HTML Replacement**: The HTML in the email body was replaced with the text inside the HTML content. Usually, HTML tags were observed to be present in the email body when images are shared.
3) **New Lines and Unicode Removal**: New line characters, emojis, and Unicode characters were removed from the email text.
4) **Remove Specific Patterns**: Specific patterns such as Boilerplate, Signatures, and Greetings were removed from our email text.

3 LABEL ANNOTATION

To label the Enron email dataset two signals are used to filter suspicious emails and label them into fraud and non-fraud classes.
1) Automated ML labeling
2) Email Signals

3.1 AUTOMATED ML LABELING

The following heuristics are used to annotate labels for Enron email data using the other two data sources,
1) **Phishing Model Annotation**:A high-precision SVM model trained on the Phishing mails dataset, which is used to annotate Phishing Label on the Enron Dataset.
2) **Social Engineering Model Annotation**: A high-precision SVM model trained on the Social Engineering mails dataset, which is used to annotate Social Engineering Label on the Enron Dataset.

The two ML Annotator models use Term Frequency Inverse Document Frequency (TF-IDF) to embed the input text and make use of SVM models with Gaussian Kernel.

If either of the model predicted that an email is fraud, the mail metadata was checked for several email signals. If these heuristics met the requirements of a high-probability fraud email, we label it as a fraud email.

3.2 EMAIL SIGNALS

Email Signal-based heuristics are used to filter and target suspicious emails for fraud labeling specifically. The signals used were,
1) **Person Of Interest**: There is a publicly available list of email addresses of employees who were liable for the massive data leak at Enron. These user mailboxes have a higher chance of containing quality fraud emails.
2) **Suspicious Folders**: The Enron data is dumped into several folders for every employee. Folders consist of inbox, deleted-items, junk, calendar, etc. A set of folders that have a higher chance of containing fraud emails, such as Deleted Items and Junk.
3) **Sender Type**: The sender type was categorized as 'Internal' and 'External' based on their email address.
4) **Low Communication**: A threshold of 4 emails based on the table below was used to define Low Communication. A user qualifies as a Low-Comm sender if their sent emails are less than this threshold. Mails sent from low-comm senders have been assigned with a high probability of being a fraud.
5) **Contains Replies and Forwards**: If an email contains forwards or replies, a low probability was assigned for it to be a fraud email.

count	20131
mean	12.3
std	104.9
min	1
25%	1
50%	1
75%	4
max	5486

Table 2: Table represents the distribution of the length of email bodies in terms of words.

3.3 MANUAL INSPECTION

To ensure high-quality labels, the mismatch examples from ML Annotation have been manually inspected for Enron dataset relabeling.

4 BASE MODELING

Fraud Modeling was performed in this project utilizing several Machine Learning(ML) and Deep Learning (DL) models. These models were trained on the training dataset without any ethical considerations, enabling the benchmarking of fraud modeling performance on the email dataset.

The ML models used in this project include.
1) (Baseline) Support Vector Machine (SVM)
2) Random Forest
3) Logistic Regression
4) Naive Bayes

The DL models used in this project include,
1) DistilBERT
2) RoBERTa

4.1 MACHINE LEARNING

For the ML models, Word2Vec embeddings have been used to encode the text data into a numeric space. The Word2Vec embeddings have been trained on the Google News dataset of around 100 Billion words. Each word in each email text is encoded into a 300-dimensional vector space, and these embeddings are averaged across the entire fraud email.

Hyperparameter Optimization is performed to pick the best version of each of the ML models dataset, ensuring a low False Positive rate in the Sanity set and a low False Negative rate in the Gold Fraud set.

After successfully performing Hyperparameter Optimization, the project notes the best parameters for the ML models to be as follows,

SVM

Kernel	Gaussian
Regularization Parameter (C)	1
Class Weights	Balanced

Table 3: SVM Hyperparameters

Random Forest

Criterion	Gini
Number of Estimators (C)	100

Table 4: Random Forest Hyperparameters

Logistic Regression

Penalty	L2
Solver	lbfgs
Tolerance Level	1e-4

Table 5: Logistic Regression Hyperparameters

Naive Bayes

Variance Smoothing	1e-9

Table 6: Naive Bayes Hyperparameters

4.2 DEEP LEARNING

Since the project makes use of a large dataset, fraud modeling using a deeper neural network-based architecture is also explored. Pre-trained DistilBERT model is used, which is a lightweight version of the BERT transformer model. Though lighter in size compared to the larger BERT model, distilBERT is benchmarked to perform almost as well as the BERT model on a text classification task.

To segment the input email text and encode it into a vector space, a DistilBERT tokenizer is used. To specifically run the model on the downstream task of text classification, a classification head is added to the model logits, which consists of 2 linear layers and a Rectified Linear Unit(ReLU) layer

Similarly, pre-trained RoBERTa-large, a larger variant of the BERT model is also explored. The model makes use of dynamic masking, enabling better attention computations.

To segment the input email text and encode it into a vector space, a RoBERTa tokenizer is used. Again a classification head is added to the model logits, which consists of 2 linear layers and a Rectified Linear Unit(ReLU) layer.

One of the constraints faced in performing the RoBERTa model training is computation. With multiple trainable layers and computationally heavy backpropagation, the RoBERTa training requires GPUs as well as takes a significant amount of time to train.

The configuration of hyperparameters used for the deep learning modeling experiments are as follows,

Learning Rate	2e-5
Optimizer	Adam W
Epochs	40
Batch Size	32

Table 7: DL Experiment Hyperparameters

5 ETHICAL MODELING

5.1 DIFFERENTIAL PRIVACY

Differential privacy offers a rigorous mathematical framework for quantifying and bounding the privacy risks associated with releasing statistical information about a dataset.

5.1.1 MACHINE LEARNING

Differential privacy was implemented using Machine Learning on the following models, 1) Random Forest
2) Logistic Regression
3) Naive Bayes

The application of noise has been done on the model and not on the training data. This successfully enables Global Differential Privacy. To implement it, the IBM Differential Privacy Library Diffprivlib [https://github.com/IBM/differential-privacy-library] was used.

5.1.2 DEEP LEARNING

A Differentially Private Fraud Model was also implemented on a BERT model using the "Opacus" framework [https://opacus.ai/].

The experiment was implemented by adding noise to the parameters of our BERT Model.

Throughout the training process, noise was introduced into the gradients of deep neural networks via differentially private stochastic gradient descent (DP-SGD). The goal of this strategy was to reduce the model's ability to memorize specific input instances. For this privacy problem, Gaussian noise was added to model parameters. In DP-SGD, adding noise during each iteration can accumulate and potentially degrade the model's performance over time. Thus, gradient clipping was introduced before adding the noise, noise scaling, and learning rate adjustment.

For the differentially private model experiment, the following parameter values were used,

Sigma (Noise Multiplier)	0.37
C (Max Grad Norm)	0.1
Epsilon	1e-08
Target Epsilon	7.5
Delta	1/len(training$_data$) $= 3.4e-05$
Epochs	40

Table 8: Differential Privacy - DL Experiment Hyperparameters

5.1.3 PRIVACY-ACCURACY TRADE-OFF

The following observations were made on the Differentially Private Machine Learning modeling performed using the Logistic Regression model. Smaller values of Epsilon provide a stronger pri-

vacy guarantee but may reduce the model accuracy. The best Evaluation F1-Score seen was 0.919 for Epsilon equals 20.

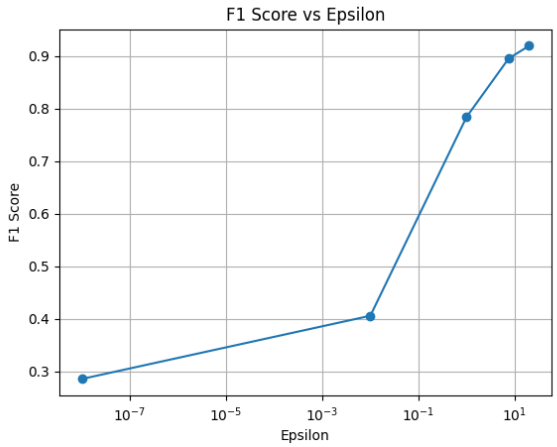

Figure 1: Visualizing Privacy Budget alongside F1-Score for Logistic Regression Model

5.2 HOMOMORPHIC ENCRYPTION

There are several benefits of utilizing Homomorphic Encryption (HE) to add privacy into Fraud Modeling. The benefits include,
1) It enables secure sharing and evaluation of private data without compromising privacy.
2) It Allows mathematical operations on encrypted data without revealing the actual data.
3) Addition of random noise addition to data before encryption to ensure security.
4) Noise in HE grows with each operation, potentially overflowing the actual data.

Homomorphic encryption is classified into three categories. These categories are distinguished based on the kind and frequency of mathematical computations that can be executed on the ciphertext.

	Actions	Number of Operations
Partially Homomorphic Encryption (PHE)	One (Addition or multiplication)	Unlimited
Somewhat Homomorphic Encryption (SHE)	Two (Addition and multiplication)	Limited
Fully Homomorphic Encryption (FHE)	Two (Addition and multiplication)	Unlimited

Figure 2: Types of Homomorphic Encryption

As part of the project, Fully Homomorphic encryption was implemented on the following ML models,
1) Logistic Regression
2) Random Forest

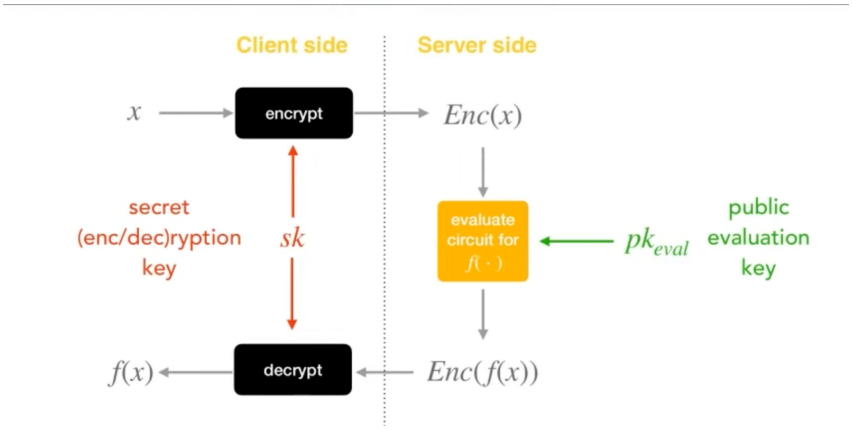

Figure 3: Fully Homomorphic Encryption Architecture

Approaches in Fully Homomorphic Encryption (FHE) include,
1) "Leveled" approach (e.g. BGV, CKKS): Minimizes computation to prevent noise overflow. "Bootstrapped" approach (e.g. TFHE): Uses bootstrapping operation to reduce noise, resulting in a tradeoff between limited fast operations and unlimited slower operations.
2) BGV and CKKS support additions and multiplications, requiring approximations for non-linear functions, suitable for statistical computing (e.g., machine learning).
3) TFHE supports boolean gates, enabling exact computation for any function. Exactness is crucial for applications like healthcare data or financial transactions, provided by TFHE.

Fully Homomorphic Encryption was implemented using the Concrete-ML Framework. The Concrete-ML Framework features a variant of TFHE supporting leveled and fast bootstrapped operations.

	BGV	CKKS	TFHE-LIB	TFHE-CONCRETE
Operations	Leveled	Leveled	Bootstrapped	Leveled + Bootstrapped
Non-linear functions	Approximate	Approximate	Exact	Exact or Approximate
Data Types	Integers	Reals	Boolean	Boolean + Integers

Figure 4: Fully Homomorphic Encryption Approaches

6 RESULTS

The above modeling experiments were run and results were noted. To benchmark the base modeling, differential privacy, and homomorphic encryption experiments, the following metrics were used.
1) False Positives in Traning Dataset
2) False Negatives in Training Dataset
3) False Positives in Sanity Dataset
4) False Negatives in Gold Fraud Dataset

6.1 MODELING

The following results were observed for the base modeling experiments. Through the model testing run, Random Forest model displayed the best results. The relative counts of training FPs and FNs

between the Random Forest model and the other models were significant. There were still a bit of missed FNs in the gold fraud set.

Model	Train FP	Train FN	Sanity FP	Gold Fraud FN
SVM	2066	17	2403	63
Random Forest	9	4	767	160
Logistic Regression	9822	8406	9514	86
Naive Bayes	60560	15327	72373	177
DistilBERT	792	645	583	82
RoBERTa	7562	4326	6885	169

Table 9: Base Model Results

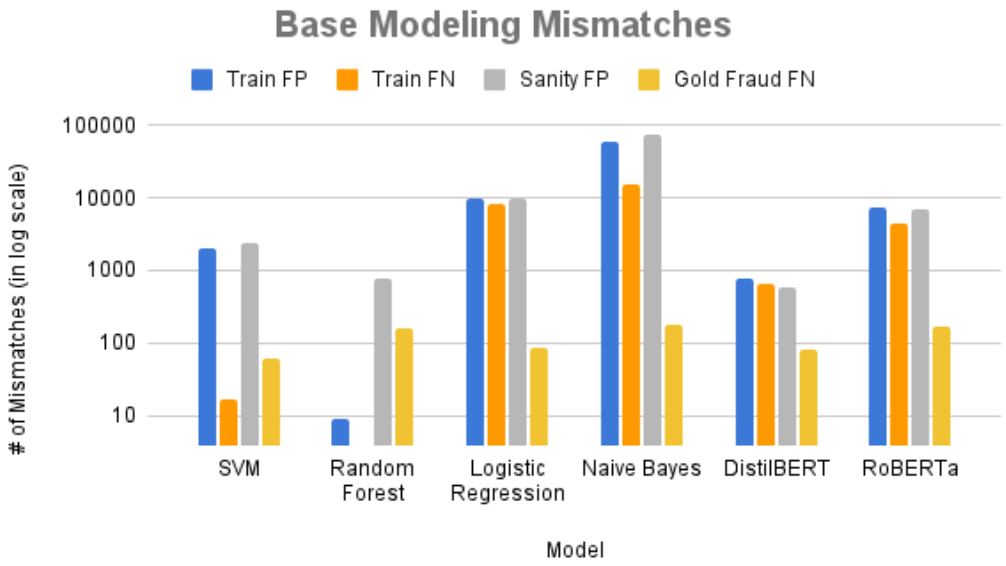

Figure 5: Comparison of Base Model Results

6.2 DIFFERENTIAL PRIVACY

The following results were observed for the differentially private modeling experiments. Through the model testing run, Logistic Regression model displayed the best results. Though the Gold Fraud FN count was low, there was still a decrease in the model performance after the application of Differential Privacy. Though the accuracy of the model decreased, there was a significant addition of privacy to the model.

Model	Train FP	Train FN	Sanity FP	Gold Fraud FN
Random Forest	25023	10132	33542	143
Logistic Regression	10023	8713	10193	88
Naive Bayes	37295	34280	34486	342
BERT	127201	439	243788	8

Table 10: Differential Privacy Results

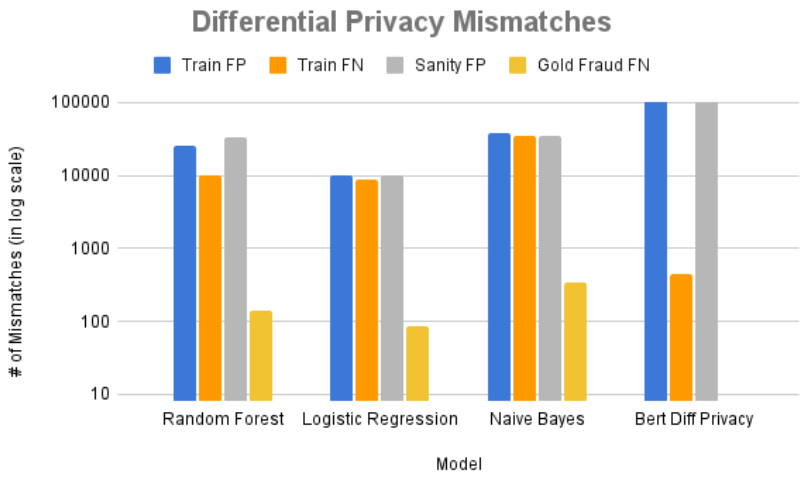

Figure 6: Comparison of Differential Privacy Results

6.3 HOMOMORPHIC ENCRYPTION

The following results were observed for the homomorphic encryption modeling experiments. Through the model testing run, Logistic Regression model displayed the best results.

Model	Train FP	Train FN	Sanity FP	Gold Fraud FN
Random Forest	56524	71666	17733	251
Logistic Regression	20136	31572	16721	179

Table 11: Homomorphic Encryption Results

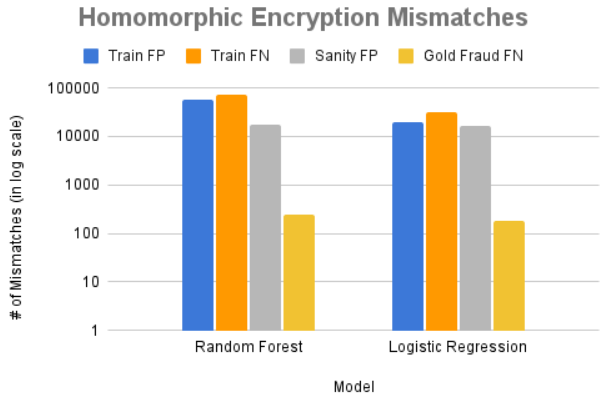

Figure 7: Comparison of Homomorphic Encryption Results

6.4 CONCLUSION

Through the modeling experiment results, it was observed that ethical techniques such as Differential Privacy and Homomorphic Encryption in Fraud Modeling using Machine Learning and Deep Learning technique help make more privacy-preserving models, but the privacy gains come at the expense of model accuracy.

In the performed Ethics-driven Fraud Detection experiments, the two best performing models were Logistic Regression and Random Forest. Though the Random Forest fraud model performed better without the addition of ethical elements, the model did not sustain great performance on the addition of privacy to the model. Whereas the Logistic Regression fraud model saw near-consistent performance throughout the base modeling as well as privacy-preserving modeling experiments. This shows that the Logistic Regression based fraud model was able to keep the accuracy of the model, while adding more noise to the data.

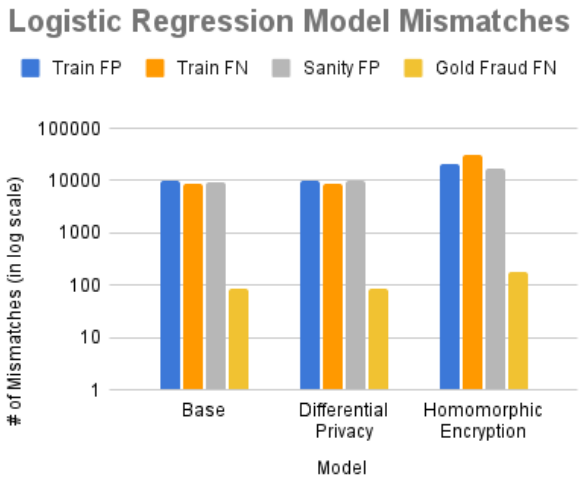

Figure 8: Comparison of Logistic Regression Model results

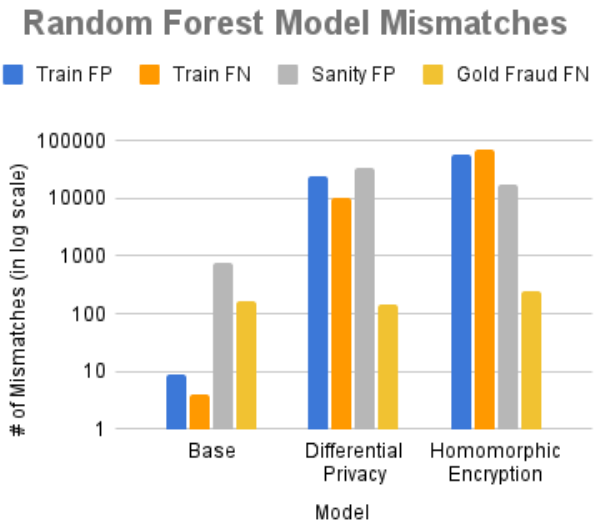

Figure 9: Comparison of Random Forest Model results

There has been great development in the field of Ethical Machine Learning and Deep Learning, where research explores novel techniques such as Federated Learning and Secure Multi-Party Computation to deal with the issue of Privacy. The project can be extended to more such experiments to

derive more conclusive evidence of the performance of different ML and DL models in being able to perform Ethics-Driven Fraud Detection. Making use of these privacy preserving techniques with Large Language Models (LLMs), that harness the power of huge amounts of internet information can also potentially help develop highly accurate yet highly private fraud detection models.

REFERENCES

1) Enron Email Dataset - https://www.cs.cmu.edu/~enron/
2) Phishing Email Dataset: https://www.kaggle.com/dsv/6090437
3) Social Engineering Dataset: http://aclweb.org/aclwiki
4) Weights and Biases for model tracking - https://wandb.ai/site
5) Differential Privacy - https://arxiv.org/abs/1412.7584
6) Homomorphic Encryption - https://arxiv.org/abs/1704.03578
7) Support Vector Machines - https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
8) Random Forest - https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
9) Logistic Regression - https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
10) Gaussian Naive Bayes - https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html
11) DistilBERT - https://huggingface.co/docs/transformers/model_doc/distilbert
12) RoBERTa - https://huggingface.co/docs/transformers/model_doc/roberta
13) IBM Diffprivlib for Differentially Private Machine Learning Models - https://github.com/IBM/differential-privacy-library
14) Opacus for Differentially Private Deep Learning Models - https://opacus.ai/
15) Concrete ML for Homomorphic Encryption - https://github.com/zama-ai/concrete-ml

A APPENDIX

1) The code implementation for the Fraud Detector can be found at https://github.com/advaithsrao/Fraud-Detector
2) The model experiment tracking can be found at https://wandb.ai/regressors/Fraud-Detector?workspace=user-advaithrao
3) Setup Instructions as well as documentation for the project can be found at https://github.com/advaithsrao/Fraud-Detector/wiki