Utilising Differential Privacy and Synthetic Data to Protect Against Adversarial Attacks in Machine Learning

SEBASTIAN KOBLER

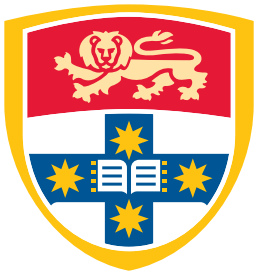
SID: 480377661

Supervisor: Dr Clement Canonne

This thesis is submitted in partial fulfillment of
the requirements for the degree of
Bachelor of Information Technology (Honours)

School of Computer Science
The University of Sydney
Australia

20 February 2023



THE UNIVERSITY OF
SYDNEY

Student Plagiarism: Compliance Statement

I certify that:

I have read and understood the University of Sydney Student Plagiarism: Coursework Policy and Procedure;

I understand that failure to comply with the Student Plagiarism: Coursework Policy and Procedure can lead to the University commencing proceedings against me for potential student misconduct under Chapter 8 of the University of Sydney By-Law 1999 (as amended);

This Work is substantially my own, and to the extent that any part of this Work is not my own I have indicated that it is not my own by Acknowledging the Source of that part or those parts of the Work.

Name: Sebastian Kobler

Signature: **Date**:

Abstract

Protecting individuals privacy has never been more relevant, as a trillion fold increase in computing power in recent years has resulted in a dependence on machine learning models to learn patterns in individuals data and make predictions on future inputs. The data used to train these models is often not available to individuals who attempt to access them, however using well-crafted attacks, an adversary can extract information from this training set which may contain private and sensitive information.

In recent years, significant defences have been proposed against these attacks, but often come at the cost of model accuracy. One promising area of research is the generation of synthetic data, that is, data that contains only 'fake' records, yet maintains the statistical properties of the original dataset.

Furthermore, with government bodies requiring by law that individuals can revoke their participation in a data set at any point in time, the dependence on a reliable means for data privacy is more relevant than ever.

This research presents a novel approach at solving these issues; we utilise a combination of both differential privacy and synthetic data generation to combat the issues that arise in both machine learning and unlearning. The process of generating this private, synthetic data is examined with the natural trade off between privacy and statistical accuracy shown. We run a variety of attacks on models trained with and without differential privacy, demonstrating experimentally just how easy extracting private information can be. Further, we examine how useful this new data is at both solving the aforementioned issues and representing sensitive data in machine learning models.

Acknowledgements

A number of individuals and organisations both within the University of Sydney and outside have contributed to my completion of this research.

First, I would like to express gratitude to my supervisor Dr Clément Canonne for sacrificing his time to supervise me as his Honours student. His passion for both learning and teaching is inspiring and motivates me to continue research into the future. Clément consistently provided me with helpful insights and assisted with any issues I had throughout my Honours year.

I would like to thank my family for supporting me throughout my research, the assistance they gave me throughout the difficult year made the process smooth and less stressful.

Finally, I am indebted to the team at Gretel.ai for providing me free access to their services for experimentation. Without the kindness of the team, my experiments would have been significantly slower and less effective at showing my hypotheses. Such kindness is what motivates research, benefiting society as a whole.

CONTENTS

Student Plagiarism: Compliance Statement ii

Abstract iii

Acknowledgements iv

List of Figures viii

List of Tables x

Chapter 1 Introduction 1

Chapter 2 Literature Review 3

2.1 Machine Learning Background . . . 3

2.2 Adversarial Machine Learning . . . 4

2.2.1 Shadow Models . . . 4

2.2.2 Membership Inference Attack . . . 4

2.2.3 Poisoning Attack . . . 7

2.2.4 Attribute Inference Attack . . . 8

2.2.5 Reconstruction Attack . . . 10

2.2.6 Summarising Privacy Attacks . . . 10

2.3 Machine Unlearning . . . 11

2.3.1 Background and Need . . . 11

2.3.2 Issues with Machine Unlearning . . . 11

2.4 Defence Mechanisms . . . 12

2.4.1 Adversarial Training . . . 13

2.4.2 Differential Privacy . . . 13

2.4.3 Private Machine Learning . . . 16

2.5 Synthetic Data . . . 18

2.5.1 Generative Models . . . 18

2.5.2 Private Generative Models 20

Chapter 3 Evaluation **21**

3.1 Experiments and Hypotheses 21

3.2 Aim 22

3.3 Contributions 22

Chapter 4 Experiments **24**

4.1 Attacking Machine Learning Models 24

4.1.1 Datasets 24

4.1.2 Models and Experimental Setup 24

4.1.3 Conducting the Membership Inference Attack 26

4.1.4 Measuring the Privacy Leakage of a Membership Inference Attack 27

4.1.5 Measuring the Value of ϵ 28

4.2 Generating Synthetic Data 29

4.2.1 Datasets 29

4.2.2 Models and Experimental Setup 30

4.2.3 Process 31

4.2.4 Evaluating the Performance of Synthetic Data 31

4.3 Generating Differentially-Private Synthetic Data 35

4.3.1 Dataset 35

4.3.2 Gretel.ai 35

4.3.3 Models and Experimental Setup 36

4.3.4 Generation Process and Collecting Results 36

Chapter 5 Results **38**

5.1 Attacking Machine Learning Models 39

5.1.1 Vulnerability per Epoch 39

5.1.2 Privacy vs Utility 41

5.1.3 ROC Curves 41

5.1.4 Discussion 42

5.2 Generating Synthetic Data 45

5.2.1 Metrics 45

5.2.2 Discussion 48

5.3 Generating Differentially-Private Synthetic Data 53
5.3.1 Metrics 53
5.3.2 Discussion 57

Chapter 6 Conclusion **61**
6.1 Future Work 62

Chapter 7 Appendix **63**

Bibliography **81**

List of Figures

2.1 The process of training k shadow models based off the input and outputs to the target model 5

2.2 Performance of *Carlini et al's.* Membership Inference Attack against previous attacks in literature. 5

2.3 Poisoning attack separating the loss distribution of members and non members of the training set. The amount of poisoning dictates how many mislabelled copies of a data point are inserted 7

2.4 Success of an attribute inference attack in the presence of a poisoned dataset 9

2.5 Misclassification of an image of a Panda by a model once noise is injected Image Credit : Explaining and Harnessing Adversarial Examples by Goodfellow et al 13

2.6 Local and central DP, the addition of noise occurs at different times. Image credit: Bennett Cyphers 15

2.7 The process of the Differentially Private Stochastic Gradient Descent algorithm Image Credit: NIST - Deploying Machine Learning with Differential Privacy 17

2.8 The process of a Generative Adversarial Network Image Credit - O'Reilly 19

4.1 MIA model architecture 25

4.2 Vehicle Dataset sample 29

4.3 Twitter Sentiment Analysis sample 30

4.4 Tiktok Popular Songs 2022 sample 30

5.1 MIA Vulnerability per Epoch high noise 38

5.2 MIA Vulnerability per Epoch low noise + no DP 39

5.3 MIA Privacy vs Utility high noise 40

5.4 MIA Privacy vs Utility low noise 40

5.5 ROC curves with varying noise 42

5.6 Vehicle Distribution per Feature 46

5.7 TikTok Distribution per Feature 47

5.8 First 5 rows of the Synthetic Twitter Dataset 50

5.9 Vehicle Distribution per Feature, noise multiplier = 0.001 54

5.10 Vehicle Distribution per Feature, noise multiplier = 0.01 55

5.11 Vehicle Distribution per Feature, noise multiplier = 0.1 56

5.12 First 5 rows of Synthetic Vehicle dataset trained with 0.1 DP 58

7.1 Code Snippet of the PrivacyMetrics class adapted from TensorFlow tutorial 64

7.2 Code Snippet of the training process followed by collecting auc and adv for a single model and plotting ROC curve 65

7.3 Custom TensorFlow callback to record time 65

7.4 Helper function to create percentage frequencies for values in each attribute 65

7.5 Absolute Log Mean and STDs of Vehicle numeric data 66

7.6 Vehicle Cumulative Sums per feature 67

7.7 Vehicle Confusion Matrix 67

7.8 Absolute Log Mean and STDs of TikTok numeric data 68

7.9 TikTok Cumulative Sums per feature 69

7.10 TikTok Confusion Matrix 70

7.11 Absolute Log Mean and STDs of Vehicle numeric data, noise multiplier = 0.001 70

7.12 Vehicle Cumulative Sums per feature, noise multiplier = 0.001 71

7.13 Vehicle Confusion Matrix, noise multiplier = 0.001 71

7.14 Absolute Log Mean and STDs of Vehicle numeric data, noise multiplier = 0.01 72

7.15 Vehicle Cumulative Sums per feature, noise multiplier = 0.01 73

7.16 Vehicle Confusion Matrix, noise multiplier = 0.01 73

7.17 Absolute Log Mean and STDs of Vehicle numeric data, noise multiplier = 0.1 74

7.18 Vehicle Cumulative Sums per feature, noise multiplier = 0.1 75

7.19 Vehicle Confusion Matrix, noise multiplier = 0.1 75

List of Tables

2.1 Summarising Privacy Attacks 11

2.2 Comparing Differential Privacy models 16

4.1 Python and library versions utilised 25

5.1 Membership Inference Attack Results 40

5.2 Performance Metrics for each Synthetic Dataset when compared against its corresponding Real Dataset 48

5.3 Performance Metrics for the Synthetic Vehicle datasets trained with varying levels of noise against the real dataset 57

CHAPTER 1

Introduction

Over recent years, there has been significant progress in the field of Machine Learning. Many applications in this field involve analysing large datasets that contribute to the training of various models to predict patterns in future data inputs. Despite the usefulness of these models, there has been increasing need for privacy measures to protect an individuals data that contributed to the training of these specific models as this data may contain sensitive information such as medical records or personal emails. It is well known that machine learning models often unintentionally leak information about the data used to train them, often facilitated through a number of attacks specifically aimed at revealing aspects of this sensitive training data or worse, entirely reconstructing it.

One privacy technique in use to combat these attacks is "Differential Privacy" [DMNS06], whereby a bound is placed on the influence on any individual data point in a model's training. In addition to this, researchers have begun experimenting with generating synthetic data, with the hope of creating entire datasets that maintain the statistical properties of the original data, but contain only fake records.

In addition to the requirement for practical applications of machine learning models to be inherently private, legislation has been introduced recently which require by law that any individual can maintain the *"right to be forgotten"*. That is, at any time, a user can request their data be removed from a dataset and further, their data should be "forgotten" from any model trained on this data. This led to the notion of "Machine Unlearning", whereby the goal is to remove some data from a machine learning model as if the model were never trained with that data in the first place.

This requirement has resulted in a number of frameworks being developed [TCMK21, BCCC$^+$19]. However, the frameworks proposed rarely account for the presence of an adversary and current literature [GGMV22, CZW$^+$21] shows that privacy attacks are perhaps even easier after the unlearning process has occurred, highlighting the need for defence mechanisms to be in place for real-world machine unlearning.

To demonstrate the necessity of defence mechanisms, we run state of the art adversarial attacks on machine learning models trained both with and without *Differentially Private Stochastic Gradient Descent*, an adaptation to the typical training of machine learning models whereby the privacy of the training set is protected. The ease at which private information is leaked without sufficient defence is obvious. We vary the level of privacy added to demonstrate the trade off between privacy and accuracy that naturally occurs.

In addition to this, we propose the usage of *Differentially Private Synthetic Data* for applications in the real world. Theoretically, using a completely synthetic data set that also has differential privacy guarantees on the original data set would allow for model usage without the privacy concerns typically involved. Furthermore, the need for *"the right to be forgotten"* would be unnecessary, considering the training data set of any model would only contain fake records. As a result, we would be solving the computational costs of machine unlearning.

Obviously, the process of generating this private, synthetic data is not so simple, we examine the costs involved as well as the trade off between privacy and statistical accuracy of the resulting data set. The aim is to generate this data with two guarantees. One being that the underlying properties of the original data set are maintained, meaning any resulting analysis on the two data sets would be approximately the same. Second, we require that an analysis of the data can not infer if any single individual was present or not in the data set (differential privacy guarantee). Theoretically, data with both these properties would solve the variety of issues involved with not only data privacy within machine learning, but also machine unlearning.

CHAPTER 2

Literature Review

This literature review analyses the various adversarial attacks against machine learning models that occur in literature, examines the successes and downfalls of current machine unlearning frameworks and reveals the effect of differential privacy in a machine learning and unlearning context. In addition to this, current examples of synthetic data generation are explored and examined, with its ability to preserve privacy being criticised.

2.1 Machine Learning Background

A classifier $f_\theta : \mathcal{X} \to [0,1]^n$ is a learned function that takes an input sample $x \in \mathcal{X}$ and maps it to a probability vector over n classes. Given some dataset sampled from some underlying distribution $\mathcal{D} \leftarrow \mathbb{D}$, we write $f_\theta \leftarrow \mathcal{T}(\mathcal{D})$ to denote that some neural network f parameterized with weights θ is learned by running the training algorithm $\mathcal{T}$ on the training set $\mathcal{D}$. Neural networks are trained through a process called stochastic gradient descent [LBBH98] to minimize some loss function l:

$$\theta_{i+1} \leftarrow \theta_i - \eta \sum_{(x,y)\in B} \nabla_\theta \ell\left(f_{\theta_i}(x), y\right)$$

B is a batch of random training examples from the training set D, and η is the learning rate. When conducting classification tasks, the most common loss function to use is cross-entropy loss:

$$\ell\left(f_\theta(x), y\right) = -\log\left(f_\theta(x)_y\right) \quad x \in \mathcal{X}, y \in [n]$$

Running the gradient descent algorithm above on the training set will eventually yield 100% training accuracy, however the more difficult yet desired outcome is to be able to generalise to an unseen test set $\mathcal{D}_{test} \leftarrow \mathbb{D}$ drawn from the same distribution as the training set.

2.2 Adversarial Machine Learning

In recent years, significant developments [RG20] in adversarial attacks have occurred as a result of the need for heightened security and privacy. Adversarial attacks are techniques aimed at exploiting models by using obtainable information to lower the performance of classifiers. Once the performance is lowered, these adversaries aim to extract private information on aspects such as training data. In the following section, we examine privacy attacks on models, how they are conducted and what information the adversary can extract from the model. For each attack, the adversary is assumed to only have black-box access to the model, that is no information about the model structure or parameters is known to the attacker. The adversary only has the ability to input into the model and observe the output. In real-life scenarios, the adversary often has the ability to leverage information known about the models structure and as such, black-box access is assumed to demonstrate the minimum capabilities of an adversary.

2.2.1 Shadow Models

Recent advances in adversarial machine learning have included the use of **shadow models** in the creation of attacks [SSSS16]. These are essentially a collection of models trained by the adversary to attempt to mimic the target model.

The difference here is that the data used to train shadow models is known (and thus the ground truth label of each individual point). Then, an **attack model** is trained based on the inputs and outputs of each respective shadow model. *Shokri et al.* describe two methods to generate data to train shadow models assuming black box access to the target model. Here, it is required that the training sets of both the shadow and target models are disjoint, but each respective shadow model may have overlapping data with one another. Once the attack model has been trained, we can attempt to construct various attacks as seen in the next sections.

2.2.2 Membership Inference Attack

To determine the amount of information leaked about individual data records within the training set of a model, a membership inference attack (MIA) is utilised [SSSS16, CCN$^+$21]. Given a machine learning model and a record, we ask the question wether or not this record was utilised in the training of the model. *Shokri et al. (2017)* limit the access of an adversary by assuming only **black-box** model access.

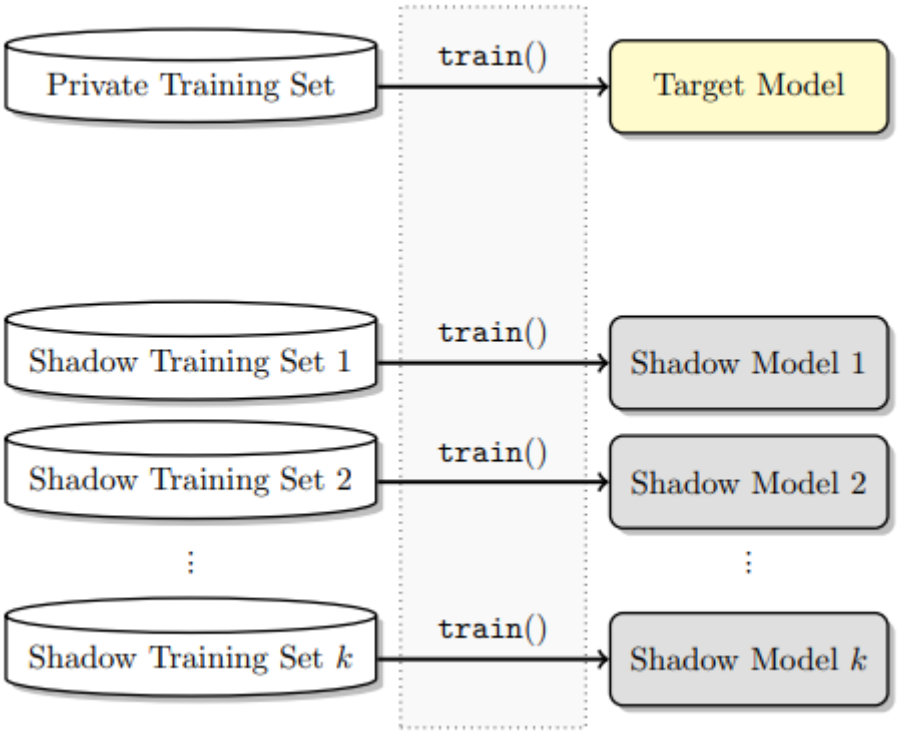


FIGURE 2.1: The process of training k shadow models based off the input and outputs to the target model

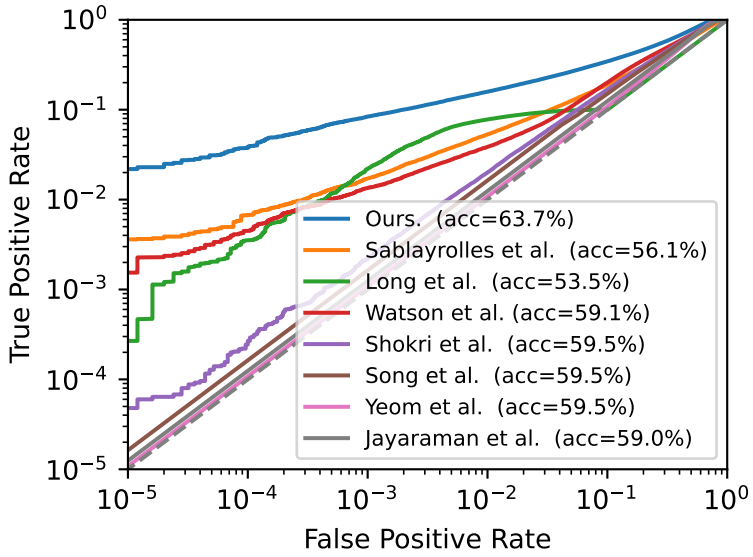


FIGURE 2.2: Performance of *Carlini et al's.* Membership Inference Attack against previous attacks in literature.

Carlini et al. (2021) improve upon the inference attacks from previous works, creating an attack method that is 10x as accurate as shown in Figure 2.2. This work is considered the state of the art attack and should be utilised when extending these attacks into other areas such as machine unlearning. The authors mention MIA's as the simplest and most widely deployed attack to measure training data privacy. That is, if a model leaks significant information through a MIA, the model itself must be unintentionally leaking information.

To formalise a membership inference attack, *Jayaraman et al*. further define a security game as follows.

Game 1 [JWK$^+$21]

The game proceeds between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:

(1) *The challenger samples a training dataset D from $\mathbb{D}$ and trains a model $f_\theta \leftarrow \mathcal{M}(D)$ on the dataset D.*
(2) *The challenger flips a coin, if the coin lands heads, they sample a fresh point (x,y) from the underlying distribution $\mathbb{D}$ such that (x,y) was not already in the dataset D. If the coin lands tails, the challenger selects (x,y) from the training set D.*
(3) *The challenger sends this point (x, y) to the adversary.*
(4) *The adversary gets query access to the distribution $\mathbb{D}$, and to the model f_θ, and outputs a prediction on the sample (x,y).*
(5) *Output 1 if this prediction is heads and 0 otherwise.*

This game specifically highlights the process of a membership inference attack, where the adversary outputs the prediction on the sample (x,y) as $\mathcal{A}(x, y)$ assuming the adversary is given access to the underlying training data distribution $\mathbb{D}$. Now, Carlini et al. reference both an online and offline attack. Both attacks utilise the training of shadow models on random samples from the data distribution. A gaussian is fit to the confidences of these shadow models after half are trained on the target sample (x,y) and the other half are not. A likelihood ratio-rest is utilised in the online case whereas a one-sided hypothesis test is used in the offline model comparing the target models confidence on the sample (x,y) in comparison to the tests on the trained shadow models. This attack leverages the fact that examples with lower loss are on average more likely to be members of the training data. Further details in [CCN$^+$21].

This attack is of particular relevance to the research topic at hand. As typically, when determining the amount of privacy leakage incurred, we examine the effect of a membership inference attack on the

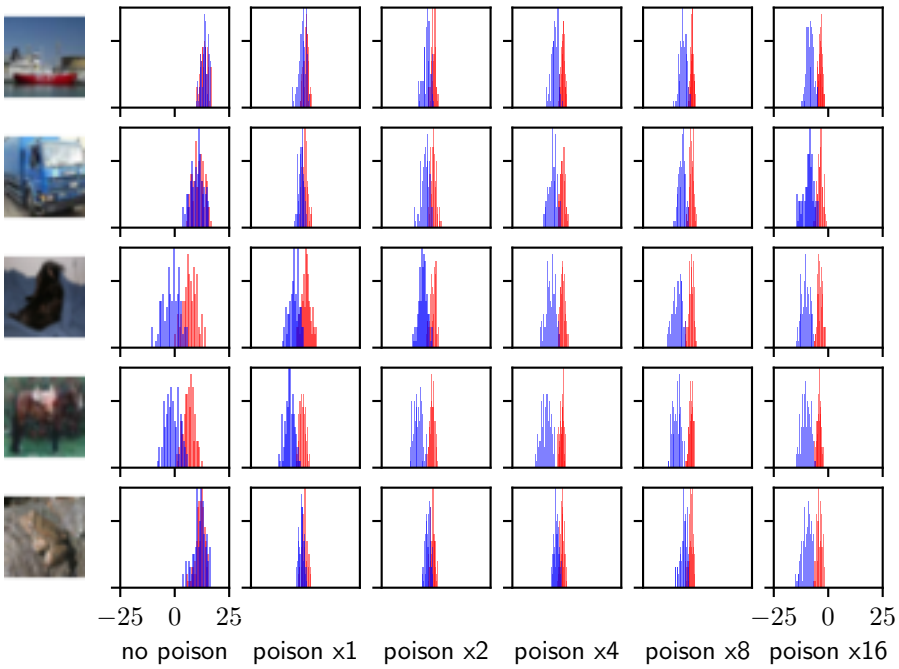


FIGURE 2.3: Poisoning attack separating the loss distribution of members and non members of the training set. The amount of poisoning dictates how many mislabelled copies of a data point are inserted

model. In general, if deploying defence mechanisms such as differential privacy hinders the effect of a MIA, it must inherently be making the model more private.

2.2.3 Poisoning Attack

We now highlight another attack on machine learning models, being **poisoning attacks**. As opposed to a MIA where privacy of individuals is harmed, poisoning attacks target the **integrity** of a model, degrading the performance of the classifier [MXLM20]. *Ma et al.* define two types of poisoning attacks, being *targeted attacks* and *reliability attacks*. When an adversary conducts a targeted attack, they seek to insert carefully constructed malicious data into the input space of a model without being detected to cause effects such as shifts in the decision boundary of a classifier. In a reliability attack, the adversary aims to make the model unreliable by maximising the overall prediction error of the model. The paper further presents a general attack method where poisoned copies of data are inserted into the original training data set with the goal of causing a target sample x to be misclassified. Thus here, the goal is not to compromise the privacy of the data, but the accuracy of the model.

Further work is done on extending this attack in the paper of [TSJ+22] whereby a model is initially poisoned with mislabelled copies of a target sample x, which causes significantly easier membership inference attacks on the model. The target sample becomes an outlier in the model due to the existence of several miss labeled copies of the data point, which then separates the loss distributions of members and non members as shown in Figure 2.3. *Tramer et al.* mention that a MIA is significantly easier on a point that is an outlier, as it has a heightened influence on the model when present in the initial training set.

This motivates the need for a game definition similar to that in Section 2.2.2, though the game is extended to account for a poisoning attack by *Tramer et al.* as follows:

Game 2 [TSJ+22]

The game proceeds between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. The assumption is both have access to a distribution $\mathbb{D}$, know the universe $\mathcal{U}$ and training algorithm $\mathcal{T}$:

(1) *The challenger samples a training dataset $D \leftarrow \mathbb{D}$ and a target $z \leftarrow \mathcal{U}$ from the universe such that $D \cap \mathbb{U} = \emptyset$.*
(2) *The adversary send a poisoned data set D_{adv} of size N_{adv} to the challenger.*
(3) *The challenger trains a model $f_\theta \leftarrow \mathcal{T}(D \cup D_{adv} \cup \{z\})$ on the poisoned dataset $D \cup D_{adv}$ and target z.*
(4) *The challenger gives the adversary query access to f_θ and the adversary emits a guess $\hat{z} \in \mathbb{U}$*
(5) *The adversary wins the game if $\hat{z} = z$.*

In summary, the paper aims to poison a model such that any target sample x is treated as an outlier by the model, then it conducts the membership inference attack of *Carlini et al.* as highlighted in Section 2.2.2. This attack in theory should be significantly harder in a differentially private environment, as the influence of any single data point on the output of a model is bounded, meaning it would be harder to leverage changes in the loss distributions of members and non members.

2.2.4 Attribute Inference Attack

The work of *Tramer et al.* also extends past inferring only membership and looks at inferring actual data known as an *attribute inference attack* (AIA). Utilising a state-of-the-art model inversion attribute inference attack [MDK+22] from *Mehnaz et al.* In this attack, the adversary attempts to infer missing

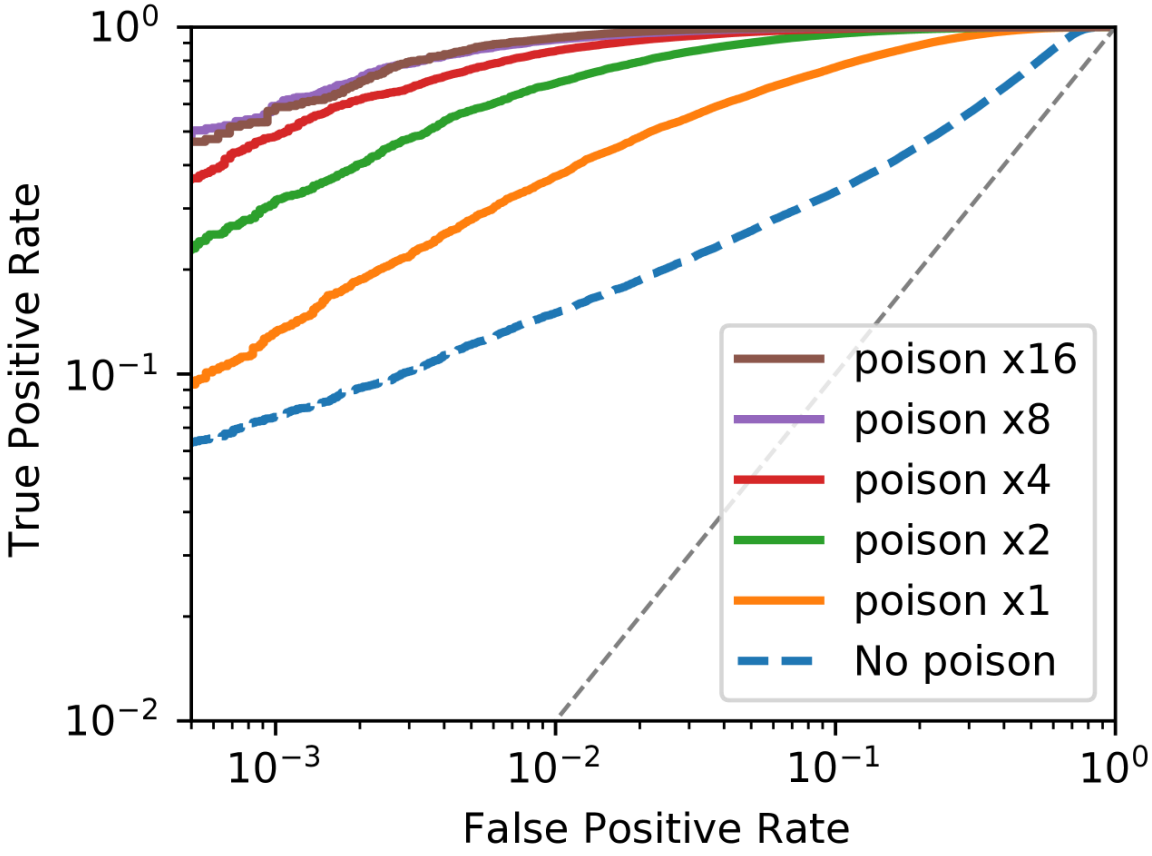


FIGURE 2.4: Success of an attribute inference attack in the presence of a poisoned dataset

attributes of a partially known record within the training data set. Previous research has shown attribute inference to be a much harder attack [ZAC+21], as if a model is susceptible to a MIA, it is not inherently susceptible to an AIA.

However, the presence of inserting poisoned samples into the training set as shown in Section 2.2.3 is shown to significantly improve the success rate of an attribute inference attack. To highlight the effect of this attack, *Tramer et al.* utilised the Adult dataset, which involves 14 input variables in the sample x such as Age and Occupation to predict an individuals income (the label y). They ran the attack assuming the unknown input variable was 'Marital Status'. This is highlighted in Figure 2.4, where the insertion of poisoned copies of the sample x with the carefully constructed labels results in a significant increase in the True Positive rate (at low false positive rates).

Clearly, if it is possible to construct attributes of some partially known data record through a poisoning attack, it may be possible to completely reconstruct some data that was deleted in the process of machine

unlearning provided partial knowledge of the deleted record. This presents significant privacy risks as it questions if the deletion actually caused a worsening of privacy. This extends naturally into the next section, where a worse case privacy leakage is examined.

2.2.5 Reconstruction Attack

Reconstruction attacks attempt to fully or partially recreate one or more training samples from a model [RG20]. As such, they are considered a stronger attack than Membership Inference attacks as the recreation of a training sample implies that sample was originally in the training set. One interesting fact [YGFJ17] is that models with higher predictive power are more susceptible to reconstruction attacks, which is atypical as one would assume more accurate models would be robust against adversarial attacks.

Limited work has been conducted on reconstruction attacks in a black-box setting, as it is extremely difficult for an adversary to fully recreate a training sample whilst only being able to query the model and obtain an output. However, in a machine unlearning context, the additional power to query the model both before and after unlearning and compare results has lead to the development of a "deletion reconstruction attack" [GGMV22] whereby the goal is to extract information about features of a deleted instance, although some assumptions are made on the adversary and the deleted sample (x, y).

The work of *Gao et al.* leads heavily into the goal of assessing the privacy leakage of machine unlearning and utilising defence mechanisms to prevent adversarial attacks at low accuracy cost. If machine unlearning leads to easier reconstruction attacks, it is vital we implement strategies to protect the privacy of individuals.

2.2.6 Summarising Privacy Attacks

The privacy attacks mentioned are only a select few of the significant amount that exist in literature [RG20]. Despite this, these are the most relevant and applicable to the research area of interest. The goal is to extend these attacks into machine unlearning, whereby the privacy leakage can be measured via a membership inference attack. If any of these attacks are more successful due to the machine unlearning process, it reveals the process as being unsafe and suggests defences are necessary to protect an individuals privacy.

Attack	*State-of the Art Paper*	*Assumptions*	*Goal*	*Target*
Membership Inference	[CCN+21]	Black box, access to training data distribution, training technique known	Given a machine learning model and a record (x, y), determine if the record was utilised in the training of the model.	Privacy
Poisoning	[TSJ+22]	Black box, adversary can insert data into training set, access to same distribution $\mathbb{D}$ as model owner, knows the universe of data $\mathcal{U}$, knows training algorithm $\mathcal{T}$	*Targeted:* insert malicious data into the input space to cause model weakness. *Reliability:* make the model unreliable by maximising the prediction error of the model	Integrity
Attribute Inference	[MDK+22]	MIA assumptions, partial knowledge of a data record (x, y)	Given a machine learning model and partial knowledge of a data record (x, y) determine the missing attributes.	Privacy
Reconstruction	[GGMV22]	MIA assumptions, depending on level of reconstruction some existing knowledge of (x, y)	Given a machine learning model, reconstruct records (fully or partially) utilised in the training procedure.	Privacy

TABLE 2.1: Summarising Privacy Attacks

2.3 Machine Unlearning

2.3.1 Background and Need

Inspired by the need for increased levels of privacy as the use of machine learning models drastically increases, machine unlearning is the process of a model 'forgetting' some subset of the data initially used to train it [CY15]. The introduction of recent laws such as the GDPR in the EU, the California Consumer Privacy Act in the US and the PIPEDA privacy legislation in Canada require that any individual participating in some dataset have the "right to be forgotten". That is, they are able to request their data be deleted from a model at any time (under some mild conditions) [BCCC+19]. *Bourtoule et al.* define the necessity of machine unlearning in the current time and explain the challenge in achieving it. The authors formalise the problem of machine unlearning as a game between two entities, an honest service provider $\mathcal{S}$ and a user population $\mathcal{U}$. The service provider uses data from a dataset $\mathcal{D}$ to train a machine learning model M. Now, any user $u \in \mathcal{U}$ can issue a deletion request of their data $d_u \subset \mathcal{U}$. The requirement is that the service provider must now convert any trained models M into $M_{\neg d_u}$, where $M_{\neg d_u}$ is sufficiently close to any model that was not trained with that data in the first place.

2.3.2 Issues with Machine Unlearning

The paper of *Bourtoule et al.* mentions the key issues as:

- Limited understanding of how a individual data point affects the model

- Stochasticity within training methods.
- Training is incremental
- Stochasticity in learning methods.

Clearly, there exists a naive method of unlearning which is to simply retrain the model on the dataset $\mathcal{D}^{'} = \mathcal{D} \setminus d_u$ where $\mathcal{D}$ is the original dataset and d_u is the requested data to be deleted. This itself is an issue, as retraining from scratch is a computationally inefficient solution. Not only this, but it also requires the model owner to keep the full dataset forever for retraining purposes. As data breaches are not uncommon, this poses significant security risks. As a result of these issues, a variety of machine unlearning frameworks have been developed. [CY15, BCCC$^+$19, TCMK21, CTMK22]. These frameworks all improve upon retraining time, however some act upon additional constraints such as the inability to look at the data to be deleted once the request has been issued.

From a privacy standpoint, we ask the question whether the ability for an adversary to query both M and $M_{\neg d_u}$ unintentionally leaks additional private information about the users data d_u. Previous works have shown this to be the case [CZW$^+$21, GGMV22, MRA21]. The work of *Chen et al.* proposes a novel membership inference attack that demonstrates the increased privacy leakage that results due to an adversaries ability to query both models. On a similar note, the work of *Gao et al.* utilise both inference and reconstruction attacks to highlight the downfall of the machine unlearning process. *Merchant et al.* propose a poisoning attack that specifically targets model integrity which forces the unlearning process to be as inefficient as the naive method of retraining.

Due to these extensive attacks against the unlearning process, our work instead proposes the usage of differential privacy and synthetic data to render the unlearning process unnecessary as shown in the next section.

2.4 Defence Mechanisms

In the following subsections, we review various defence mechanisms against the privacy attacks mentioned in Section 2.2. Due to the increasing usage of machine learning models in everyday life, there has been significant research into adversarial attacks [RZQL20] as well as defences to protect individuals privacy.

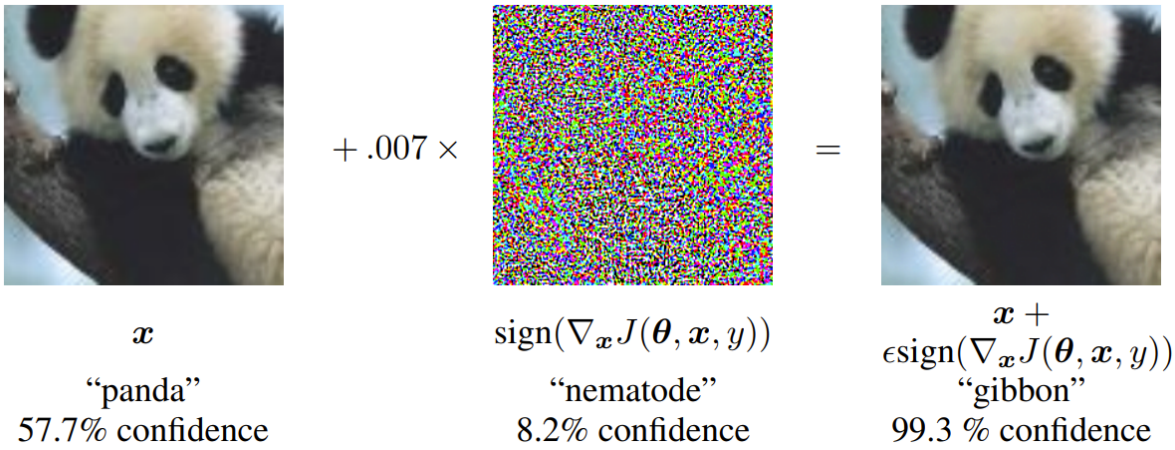


FIGURE 2.5: Misclassification of an image of a Panda by a model once noise is injected
Image Credit : Explaining and Harnessing Adversarial Examples by Goodfellow et al

To protect against adversarial attacks, a variety of defence methods are utilised [CAD+21]. Although the primary focus is on differential privacy, it is important to examine the other potential defences against adversarial attacks as these may indicate shortcomings of differential privacy as a standalone defence mechanism.

2.4.1 Adversarial Training

The primary focus of adversarial training is to inject the training set of a model with adversarial examples to increase the robustness of that model [GSS15]. Adversarial examples are perturbed examples inserted as inputs into a model and designed to fool the model into producing an incorrect result. When injecting these examples into a model during training time, the model becomes significantly more robust to even white-box attacks, where the adversary knows the model parameters and structure. Despite the success of adversarial training, the model is not entirely protected from some adversarial attacks, even in a black box setting [TKP+17].

2.4.2 Differential Privacy

Differential privacy has arisen in recent years as a way to protect private data [DMNS06]. In basic terms, this ensures that any individual who decides to participate in a statistical dataset should incur no risk. That is, any mechanism to obtain some result from a dataset D should be identical to the result obtained from a dataset $D_{\neg d_u}$ [Dwo08] where d_u denotes the data on some specific individual u.

Definition (ϵ - Differential Privacy [DMNS06]) Let ϵ be a real number and $\mathcal{M} : \mathcal{X}^n \to \mathcal{T}$ be some randomised algorithm that takes a dataset $\mathcal{D}$ as input. The algorithm $\mathcal{M}$ is (ϵ, δ) - *differentially private* if for any pair of neighbouring datasets $\mathcal{D}_1$ and $\mathcal{D}_2$ and all subsets $S \subset \mathcal{T}$,

$$\mathbb{P}[\mathcal{M}(\mathcal{D}_1) \in S] \leq e^{\varepsilon} \cdot \mathbb{P}\left[\mathcal{M}\left(\mathcal{D}_2\right) \in S\right] + \delta$$

In this case, neighbouring datasets refers to any datasets that differ by exactly one record.

Property (Post-Processing [DMNS06]) Let $\mathcal{M}$ be an ϵ-differentially private mechanism and g be an arbitrary mapping from the set of possible outputs to an arbitrary set. Then, $g \circ M$ is ϵ-differentially private.

This property states that any differentially private output can be arbitrarily transformed, using some data-independent function g, without impacting its privacy guarantees.

Essentially, differential privacy acts to bound the influence on any individual data point by some privacy guarantee ϵ. Higher values of ϵ indicate less private but also more accurate results, lower values however correspond to higher levels of noise, making it harder for attackers but also cause unreliability in the results. It is worth noting, that for small values of ϵ, $e^{\epsilon} \approx 1 + \epsilon$, giving the intuition that each output happens with a probability within a factor $\approx 1 + \epsilon$.

The value of δ corresponds to the probability of the privacy guarantee not holding. That is, it defines the probability of information accidentally being leaked.

2.4.2.1 Differential Privacy Models

In terms of achieving differential privacy, there are three main methods, being local differential privacy, central (global) differential privacy as well as shuffle privacy. All three methods achieve the privacy goal, however come with differing advantages and disadvantages.

In **local** differential privacy [KLN$^+$08], each user permutes their data before sending it to an untrusted aggregator as shown in figure 2.6. The aggregator can then run any mechanism on the new dataset, including sending this data to other servers. After the initial addition of noise by each individual, any mechanism on the data is differentially private forever.

In **central** differential privacy [DMNS06], the data is sent to a trusted central aggregator as shown in figure 2.6, who then transforms the data using a differentially private algorithm.

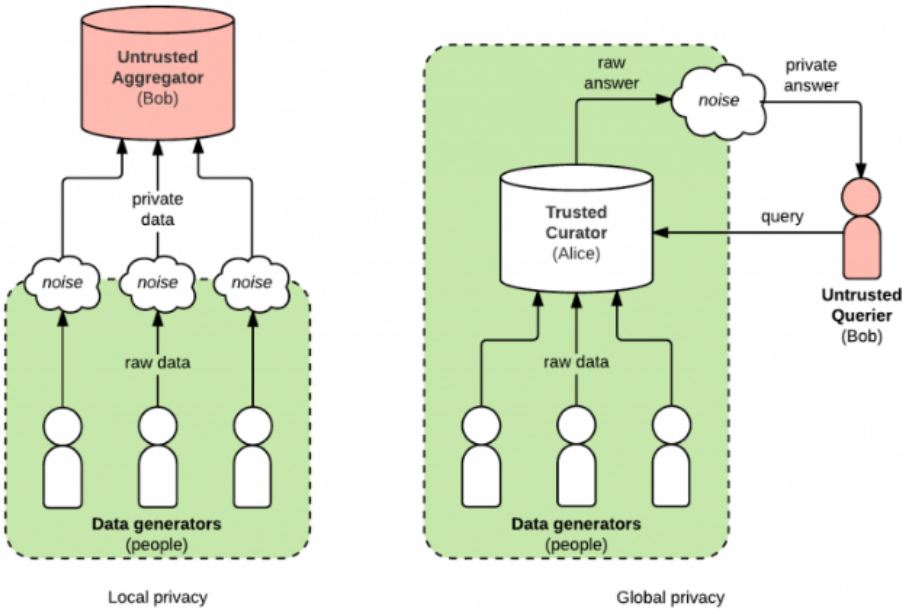


FIGURE 2.6: Local and central DP, the addition of noise occurs at different times. Image credit: Bennett Cyphers

Albert Cheu et al. extended these two methods to create a model known as the **shuffle** model [CSU+19]. The aim to is create a differentially private mechanism that combines the advantages of both the local and central models. It is composed of the following:

- Several users, who encrypts their own data and pass the resulting message to the shuffler
- A shuffler, who collects all the messages from the users and anonymises (or shuffles) them.
- An analyser who processes the data and computes some desired output.

Since it is known that the "trusted" aggregators in the central model are vulnerable to attacks, removing the need for them in the shuffle model is a significant advantage. Another positive is the improvement over accuracy when compared to the local model. The local model results in a much larger level of noise (as each individual permutes their data separately). As such, we end up with a more reliable differentially private mechanism in the shuffle model. There is one pitfall however, as we require the shuffler to be trusted. This is relatively easy to ensure as the shufflers job only involves a single, simple process that can be done securely and thus can be trusted more easily.

For best performance of the shuffle model, it is important users privatise their data in some way before sending it. It is well known that shuffling or anonymisation alone is insufficient in guaranteeing privacy,

DP- Model	*Advantages*	*Disadvantages*
Local	Data aggregator does not know actual data, privacy much easier protected. No requirement for the data curator to be trusted.	Noise is added by each individual, resulting in much higher total noise. Lower accuracy as a result. Typically results in higher values of ϵ.
Central	High accuracy since noise is relatively low (only added by the trusted curator).	Requirement for aggregator to be trusted, a significant downfall in practice.
Shuffle	Does not rely on trusted aggregator. Accuracy still high as total noise is kept at a low level.	Needs to only trust the shuffler. More components to the DP process, therefore higher computational cost.

TABLE 2.2: Comparing Differential Privacy models

however small additions of noise or privatisation followed by the anonymisation process has been shown to be extremely effective in protecting privacy.

2.4.3 Private Machine Learning

Recent work has been done to apply the differential privacy defence mechanism into machine learning training [ACG+16]. The need for differentially private learning has emerged due to the rapid progression in neural networks requiring large, representative and sometimes private data. *Abadi et al.* implement a state of the art differentially private stochastic gradient descent algorithm that works as follows:

(1) Compute the gradient for a random subset of examples.
(2) Clip the ℓ_2 norm of each gradient.
(3) Compute the average and add some amount of noise into the calculation.
(4) Take a step in the opposite direction of this new, noisy gradient.

This approach ensures the model itself is differentially private, however the authors results indicate some losses in model accuracy as a result of the added noise. The approach of *Abadi et al.* essentially works to bound the size of gradients of individual examples, meaning the influence any single data point can have on the model is bounded by some value. This leads to another important research question **What is the least amount of noise we can add to ensure the privacy of all individuals whose data is present in a model are protected?** The hope is to extend this to machine unlearning, where a minimal amount of noise is added such that a no additional privacy leakage will occur in the unlearned model. We ask the

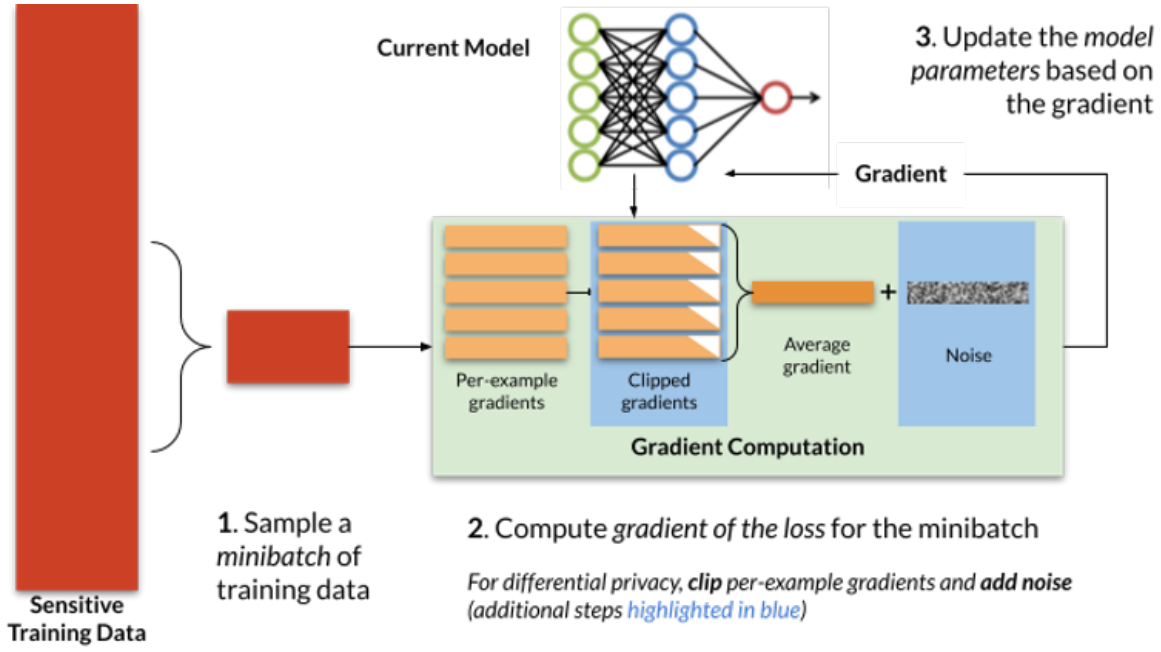


FIGURE 2.7: The process of the Differentially Private Stochastic Gradient Descent algorithm Image Credit: NIST - Deploying Machine Learning with Differential Privacy

question **Is machine unlearning possible without defence mechanisms such as differential privacy?** Recent work by *Gao et al.* indicate this not to be the case and we aim to show this in a rigorous manner.

In the work of *Tramer et al.*, rather than bounding the *size of gradients* of individual examples as done in the work of *Abadi et al.*, they decide to bound the *losses* of individual examples. This is much more computationally efficient as it only requires losses to be scaled before the backpropogation process in training. *Tramer et al.* notice this defence significantly dampens their proposed attack. This is due to the attack relying on the effect of outliers naturally having a higher influence on a model compared to an inlier, so by bounding the effect that *any* data point can have on the model, we reduce the effectiveness of the combined poisoning and MI attacks.

2.4.3.1 Where is it achieved in practice?

Recently, there has been implementations of differential privacy ranging from malware detecting in Google Chrome, Microsoft Windows' statistics collection and even the 2020 US Census [DKM19]. *C. Dwork* mentions the difficulty in choosing the best ϵ in implementations of differential privacy, highlighting the tradeoff between performance and privacy. It is also mentioned that there is a significant

need for shared learning within the DP community, as current users can learn off eachothers implementations. As such, formulating an efficient and privacy-preserving implementation of DP in the specific case of machine unlearning may be a difficult task. Judging by successful uses of DP in practice, the size of the dataset is an important factor. We see the best implementations are those with extremely large datasets, where the addition of noise from the DP process has less of an effect in impacting performance.

2.5 Synthetic Data

In this section, we present the current research into the generation of synthetic data. There are a variety of methods currently utilised to generate this data, all with varying use cases.

2.5.1 Generative Models

In recent times, there have been significant developments in the field of generative modelling. These models have been able to develop highly realistic data types such as synthetic images that are indifferentiable from real images to the naked eye. As a result, there has been increasing desire to use these models to generate 'fake' data for use in other machine learning models, allowing for datasets that contain entirely synthetic records.

2.5.1.1 Generative Adversarial Networks

Generative Adversarial Networks [GPAM+14] consists of two neural networks competing against each other in a zero-sum game. That is, one neural networks gain is the other networks loss. One of these networks is called the *Generator* whilst the other is called the *Discriminator*. The generator generates new data instances while the discriminator evaluates them to determine whether or not they were synthetically generated by the generator, or part of the original training set. The end goal of the generator is to fool the discriminator into believing the generated data is real.

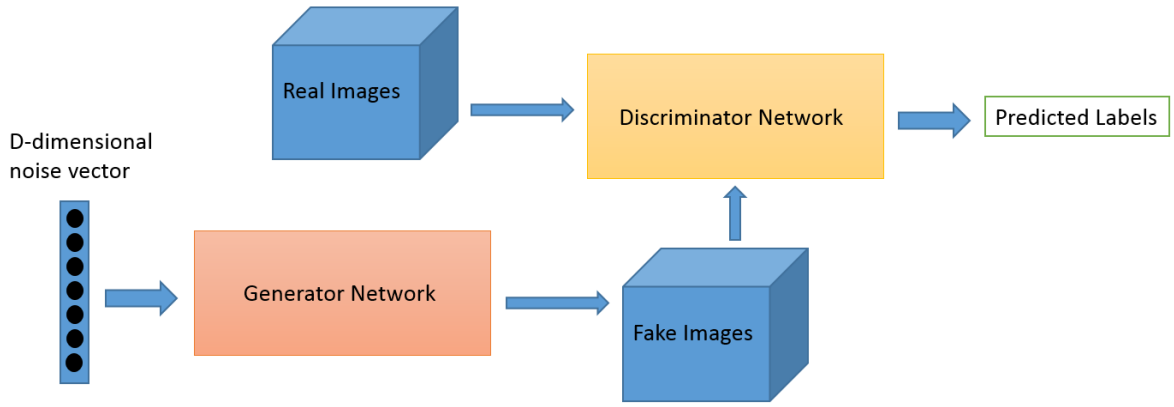


FIGURE 2.8: The process of a Generative Adversarial Network Image Credit - O'Reilly

Data generated from the Generator network is often, following the training process completely indifferentiable from training data (to a human observer). As such, this data has the potential for use in other machine learning models, where the desire is that no actual individuals are present within the training dataset.

2.5.1.2 CTGAN

CTGAN's [XSCIV19], first proposed by *Xu et al* in 2019 aimed at solving the problem of creating generative models for tabular data. Adapting GAN's for use in tabular data was a non-trivial task, considering tabular data often contains a mix of both discrete and continuous data. As a result, pre-existing statistical and deep neural network models failed to properly model this data type. The main advantage over a regular GAN is the use of mode-specific normalisation techniques to deal with attributes that do not correspond with Gaussian or multimodal distributions. In addition, in order to combat class imbalance issues that are often present in tabular data, a conditional generator and training-by-sampling methods are used.

2.5.1.3 Language Models

Language models attempt to learn the underlying probability distribution of training data. These models include Recurrent Neural Networks (RNN's) and Transformers that assign probabilities to a sequence of words. These models can predict both short and long sequences of words and as such, have the power

of generating sentences following training on large amounts of data. One important type of RNN is Long Short-Term Memory (LSTM) [HS97] , used for processing sequential data. In particular, they are capable of learning long-term dependencies, an important factor when processing inputs such as written natural language. LSTM's are a popular method of data generation as they are capable of producing new sequences based off a single seed input.

2.5.2 Private Generative Models

One promising area of study is the generation of differentially private synthetic data for public use [TMH$^+$21, ND21, Nik19]. This solves various privacy concerns in particular "the right to be forgotten". If an individuals data was used once, in the process of generating synthetic data and then deleted afterwards, there would be no need for machine unlearning to occur, meaning the privacy concerns highlighted previously would not arise. *Y. Tao et al* proposes that the generation of a synthetic data set is creating some set that "looks like" the original dataset from the perspective of an analyst, however contains only fake records. The benefit here is that once this synthetic data has been created, it remains private forever, there is no action that can "undo" this privacy as each record in the new database is not a real record. *J. Near and D. Darais* mention that many techniques to generate synthetic data work in practice, however do not satisfy the condition of differential privacy, meaning they are ineffective as adversaries could produce a reconstruction attack to create data that was originally in the training set. Although differentially private synthetic data would solve many issues with current privacy requirements, the accuracy loss as a result of utilising DP mechanisms is often large, so it is an active area of research to provide a private mechanism of generating data that still maintains a usable level of accuracy.

J. Near and D. Darais describe the process of generating data as building a probabilistic model based on the underlying distribution. Then, this new model is used to generate "fake" data that maintains the properties of the original dataset. The challenge here is in building the model and one of the more promising solutions is to utilise generative adversarial networks to generate DP-data. To protect individual data, the GAN must be trained using DP-learning as proposed by *Abadi et al.*

CHAPTER 3

Evaluation

From the literature, we know a capable adversary has the potential to attack machine learning models and extract private information from the training set. In addition to this, recall that machine unlearning has become an increasingly prevalent issue in the past few years, as laws aimed at protecting individuals privacy have been introduced that place increased pressure upon model makers to either include machine unlearning frameworks or significant privacy protection methods. Our experiments show first that machine learning models are not inherently private and also expensive to protect, demonstrating the need for other solutions to these problems. This leads to the exciting new field of synthetic data generation. If we can generate data that is both accurate and private, we simultaneously solve the issues of adversarial attacks and machine unlearning as the new data would contain no real individuals. It is already known that synthetic data generation is a promising new field as it has the potential to create as much data as desired. In a world where the need for data is exponentially increasing, generating high quality private, synthetic data is of significant importance.

3.1 Experiments and Hypotheses

This research involves 3 primary experiments to explain the practicality of generating private, synthetic data for deployment in the real world:

- **Attacking Machine Learning Models:** In this experiment, we run-state-of-the-art membership-inference attacks on neural network models with varying levels of privacy protection. We analyse the tradeoff between accuracy and privacy that occurs when varying the level of ϵ. We predict that lowering the privacy budget ϵ will have a significant effect on both model accuracy and training times and thus, worsen the utility of the machine learning model. As a result of this, we propose the use of synthetic data as an alternative method of privacy protection.

- **Generating Synthetic Data:** In this experiment, we utilise CTGAN's, a form of *generative adversarial network* used to generate synthetic tabular data. We generate synthetic data for a variety of tabular datasets and analyse its performance against the corresponding real dataset using a variety of accuracy and privacy metrics and from this, determine its potential to fix the issues of adversarial attacks and machine unlearning. We predict that synthetic data will perform well in accuracy metrics, however will fail to provide any significant means of privacy protection and as such, we move towards training generative models with differential privacy in mind.
- **Generating Differentially Private Synthetic Data:** Here, we extend upon the previous experiment by training the generative model with varying levels of privacy protection. We conduct the same metrics and compare the utility of synthetic datasets trained with differing levels of ϵ in the DPSGD process. Furthermore, we determine if this increase in privacy is sufficient enough to solve adversarial attack and machine unlearning concerns. It is predicted that as the level of ϵ lowers (i.e as the privacy guarantee increases), the quality of the synthetic data also lowers.

3.2 Aim

This research aims to contribute to the field of adversarial machine learning and data privacy. It does this by analysing the practicality of deploying private, synthetic data for use in the real world, particularly for machine learning. First the aim is to demonstrate the importance of implementing private training methods in the construction of neural network models. As a side goal, we intend to demonstrate the cost in implementing these privacy precautions and verify if other methods of protection such as data generation should be sought after if these costs are too detrimental to the learning process. Next, we aim to analyse the ability for real data to be replaced by synthetic data for use in the real world. For this to be possible, we are required to measure its effectiveness not only in maintaining accuracy against the original dataset, but protecting the privacy of individuals who contributed to the real dataset.

3.3 Contributions

This research broadly contributes to the field of adversarial machine learning and data privacy by not only determining if synthetic data has the capability to replace real data in accuracy metrics, but also

determines its effectiveness at preventing a capable adversary from extracting private information from the original dataset. Furthermore, we determine the ability for synthetic data to act as a solution to the issues of machine unlearning, which include (but are not limited to) privacy and model retraining costs. Although significant work has been done on comparing synthetic and real data through statistical metrics, this paper determines if it is possible to maintain high accuracy in the synthetic dataset while simultaneously providing individuals in the real dataset privacy guarantees through the use of differentially-private training methods. We examine the relationship between accuracy and privacy and thus, determine if this synthetic data is feasible for real world use.

CHAPTER 4

Experiments

In this chapter, we explain the set-up for the three experiments described in Section 3 as well as how the results for each are collected.

4.1 Attacking Machine Learning Models

In this experiment, we conduct two types of membership inference attacks on models trained with varying levels of privacy protection. The aim is to show the trade off between privacy and accuracy that occurs when training models utilising Differentially Private Stochastic Gradient Descent (DPSGD). The expectation is that as you decrease ϵ, corresponding to a lower privacy budget (less privacy leakage allowed), the model accuracy suffers and training time increases. We ask if there is some value of ϵ where accuracy and training time are affected as little as possible while still providing significant protection against a membership inference attack.

4.1.1 Datasets

For this experiment, CIFAR-10 [Kri09] was the dataset of choice. CIFAR-10 consists of 60000 images of size 32x32 of which there are 10 classes, 6000 per class. There are 50000 images for training and 10000 images for testing. This dataset is simple in nature allowing for the construction of small models that still maintain high enough accuracy for testing purposes.

4.1.2 Models and Experimental Setup

For this task, a variety of simple image classification models were trained on the CIFAR-10 dataset as above with varying levels of privacy. To vary this privacy, as mentioned in section 2.4.3 the level of noise introduced throughout the training process was varied. As the noise multiplier is changed, so to

Name	Version
Python	3.7.13
Tensorflow	2.9.1
Tensorflow Privacy	0.8.4

TABLE 4.1: Python and library versions utilised

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 30, 30, 32)        896

 max_pooling2d_3 (MaxPooling  (None, 15, 15, 32)       0
 2D)

 conv2d_4 (Conv2D)           (None, 13, 13, 32)        9248

 max_pooling2d_4 (MaxPooling  (None, 6, 6, 32)         0
 2D)

 conv2d_5 (Conv2D)           (None, 4, 4, 32)          9248

 max_pooling2d_5 (MaxPooling  (None, 2, 2, 32)         0
 2D)

 flatten_1 (Flatten)         (None, 128)               0

 dense_2 (Dense)             (None, 64)                8256

 dense_3 (Dense)             (None, 10)                650

=================================================================
Total params: 28,298
Trainable params: 28,298
Non-trainable params: 0
_________________________________________________________________
```

FIGURE 4.1: MIA model architecture

does the value of ϵ, the privacy budget. Model parameters are kept consistent throughout each training run, utilising ReLU as the activation function, a batch size of 50 and a learning rate of 0.001. All models are trained for a total of 50 epochs.

To construct, train and test each model, we utilised the Tensorflow and Tensorflow Privacy libraries within Python. The versions are listed in Table 4.1:

These experiments are done using Nvidia Tesla P100 GPU's with 16gb memory as provided by Google Collab Pro. Four Intel(R) Xeon(R) CPU's @ 2.20GHz are utilised providing a total of 24gb of Ram.

To get a summary of model architecture, TensorFlow provides a method model.summary() with output shown in Figure 4.1. Since the model architecture is unchanged throughout the experiment, the output is the structure of all trained models presented in the results section.

```
model.compile(
    loss=tf.keras.losses.CategoricalCrossentropy(from_logits=True),
    optimizer=tensorflow_privacy.DPKerasAdamOptimizer(l2_norm_clip=1.0,
    noise_multiplier=0.05, num_microbatches=1,learning_rate=lr),
    #optimizer=tf.keras.optimizers.Adam(learning_rate=lr),
    metrics=['accuracy'])
```

As shown in the code snippet above, each model is compiled using Cross Entropy loss and the Adam optimiser. When utilising differential privacy, $tensorflow\_privacy.DPKerasAdamOptimizer()$ replaces the usual $tf.keras.optimizers.Adam()$ optimiser. This optimiser takes in the required DP parameters, being *l2_norm_clip* and the *noise_multiplier* as mentioned in section 2.4.3 as well as the *num_microbatches* which is the number of microbatches into which each minibatch is split, although this is kept at 1 during our experiments.

4.1.3 Conducting the Membership Inference Attack

To conduct a MIA, we use two methods

(1) **Logistic Regression:** We train a shadow model based on the outputs of the target model as per section 2.2.1. In this particular experiment, logistic regression is the training method, however it is also possible to use multilayer perceptrons, random forests or k-nearest-neighbour models.

(2) **Threshold Attack:** We also conduct a threshold attack, whereby a shadow model is not necessary. In this attack, for some given threshold amount, the attack will determine how many samples present in both the training and testing set that have membership probability higher than this given threshold. This attack utilises the fact that training samples typically have lower loss than samples from the test set. As a result, it predicts samples that have lower loss than this threshold as being apart of the original training set.

With this is mind, we train a total of 6 models with identical architecture and parameters as per above for 50 epochs each. Each model is trained with a varying *noise multiplier* levels between 0.5 and 0.001 and

an ℓ_2 *norm clipping rate* of 1.0. On each model, we conduct both a *logistic regression* and *threshold attack* during the training process. To keep track of both accuracy and privacy metrics over time, we utilise a TensorFlow custom callback called *PrivacyMetrics* which hooks into the training process of each model, collects the desired metrics every 2 epochs and runs the specific membership inference attacks. As the model trains, the callback records test and train accuracy, epoch number, AUC and attacker advantage. The source code of this specific class is available under an Apache 2.0 license from `https://github.com/tensorflow/privacy/blob/master/g3doc/tutorials/privacy_report.ipynb`. Additionally, we incorporate an additional TensorFlow custom callback called *TimeCallback* found in the code snippet 7.3 which records the cumulative training time per epoch. Once all models have trained, we combine all the attack results into a single AttackResultsCollection. From here, it is possible to collect a number of results and even plot things like the ROC curves (defined below) of the best attacks within each model.

4.1.4 Measuring the Privacy Leakage of a Membership Inference Attack

To measure privacy leakage, there are two important metrics to consider:

(1) The first is Area Under the Receiving Operating Characteristics curve (AUC). The Receiving Operating Characteristics (ROC) Curve plots the true positive rate against the false positive rate. These two values are calculated using precision [1] and recall. [2]. Once plotted, the curve presents how accurately an adversary can predict whether or not some data point was present in the training set or not. Since the ROC is a probabilistic curve, we instead plot the AUC, which measures how well the attacker can distinguish between the membership classes. Hence, an AUC value of 0.5 would correspond to the attacker not being able to distinguish between membership classes. As this value increases, the attacker is better able to distinguish data being in the training set or not and therefore, corresponds to a higher privacy leakage.

(2) The second is attacker advantage, loosely based on the work of [YGFJ17] whereby a *membership advantage* is defined and formalised. Here, we measure how well the attacker can distinguish between a data point being from the training set or not. Essentially, the membership advantage is the difference between the models true and false positive rates.

[1] Precision is calculated by dividing true positives by all positives predicted.

[2] Recall is calculated as the true positive rate, that is dividing the true positives by all positives (even if reported as a negative).

We present four plots in total to accurately measure privacy leakage. The first set plots vulnerability changes over time through epochs as the independent variable. This allows us to examine exactly how privacy leakage changes over the entire training process. The second set plots privacy against utility, using validation accuracy as the independent variable. This allows us to visualise the natural trade off between privacy and accuracy.

4.1.5 Measuring the Value of ϵ

The privacy budget ϵ is important to calculate for each model as it effectively determines how "private" each is. In order to do so, we can use TensorFlow privacy's *compute_dp_sgd_privacy* tool. After model training, the function is run with five parameters:

(1) The size of the training data
(2) The batch size
(3) The noise multiplier
(4) The number of training epochs
(5) The value of δ as seen in Section 2.4.2

The value of δ is often set to be less than the inverse of the size of the training set and so is set to 10^{-5} in this specific experiment [3] as CIFAR-10 has 50000 training points.

An example is as below:

```
compute_dp_sgd_privacy.compute_dp_sgd_privacy(n=x_train.shape[0],
                                              batch_size=50,
                                              noise_multiplier=0.5,
                                              epochs=50,
                                              delta=1e-5)


DP-SGD with sampling rate = 0.1% and noise_multiplier = 0.5 iterated over
50000 steps satisfies differential privacy with eps = 10.7
and delta = 1e-05.
The optimal RDP order is 2.5.
(10.736487423262826, 2.5)
```

[3] a natural and typical setting of parameters, as $\delta > \frac{1}{|dataset\_size|}$ would allow for undesirable algorithms which could leak entirely a record with small probability. This is clearly an undesirable outcome, as it violates any intuitive notion of privacy.

	name	year	selling_price	km_driven	mileage (kmpl)	engine (CC)	max_power (bhp)	seats
0	Maruti Swift Dzire VDI	2014	450000	145500	23.400000	1248	74.000000	5
1	Skoda Rapid 1.5 TDI Ambition	2014	370000	120000	21.140000	1498	103.520000	5
2	Honda City 2017-2020 EXi	2006	158000	140000	17.700000	1497	78.000000	5

FIGURE 4.2: Vehicle Dataset sample

4.2 Generating Synthetic Data

In this experiment, we specifically examine the usefulness of synthetic data, that is data that contains the same properties as the original data, but contains entirely fake records. The aim here is to determine if this synthetic data is sufficient enough to solve the issues of both adversarial attacks and machine unlearning. We expect that this synthetic data would provide some levels of protection against a capable adversary, however based on current literature, we predict this data will still reveal significant information about the original data set. In addition, it is expected that synthetic data will have trouble dealing with structured sentences and correlations between columns.

4.2.1 Datasets

For this experiment, we restrict the type of data being used to tabular data, both for ease of training and simplicity. The three data sets utilised are Vehicle Dataset, Tiktok Popular Songs 2022 and Twitter Sentiment Analysis all found on Kaggle.

- The **Vehicle** dataset contains 7819 rows of specific vehicles along with their model year, selling price, current kilometres driven, mileage, engine capacity, max power and seats. Typically, the dataset is used to predict, given other attributes, the selling price of a vehicle.
- The **TikTok** dataset is a collection of the top 263 most popular songs in the app. The dataset contains 15 features particular to the song such as the key, tempo and duration.
- The **Twitter** dataset contains 12447 tweets about certain games with a particular sentiment. This dataset is used to train models such that given a particular tweet, the subject and the sentiment is easily predicted. As such, the features of this dataset are heavily correlated and relate to one another.

Samples of each dataset are given in Figures 4.2, 4.3 and 4.4 respectively. Despite all being tabular datasets, they vary in properties such as size and dimensionality and hence are sufficient choices for synthetic data generation. Where the vehicle dataset contains a low amount of mostly numerical attributes,

	ID	Game	Sentiment	Tweet
12444	9198	Nvidia	Negative	Nvidia really delayed the 3070 2 weeks .
12445	9199	Nvidia	Positive	Let no elim go unnoticed. . . . NVIDIA Highlights automatically records your best moments in @FortniteGame on GFN!. . Share them with
12446	9200	Nvidia	Positive	Just realized the windows partition of my Mac is like 6 years behind on Nvidia drivers and I have no idea how I didn't notice

FIGURE 4.3: Twitter Sentiment Analysis sample

	track_name	artist_name	artist_pop	album	track_pop	danceability	energy	loudness	mode	key	speechiness	acousticness	instrumentalness	liveness	valence	tempo	time_signature	duration_ms
260	Jimmy Cooks (feat. 21 Savage)	Drake	95	Honestly, Nevermind	92	0.529000	0.673000	-4.711000	1	0	0.175000	0.000307	0.000002	0.093000	0.366000	165.921000	4	218365
261	Good Looking	Suki Waterhouse	64	Good Looking	80	0.377000	0.558000	-9.076000	1	4	0.029900	0.078900	0.000342	0.125000	0.267000	149.971000	3	214800
262	INFERNO	Sub Urban	67	INFERNO	71	0.820000	0.611000	-5.020000	0	9	0.122000	0.076600	0.000025	0.068400	0.637000	127.883000	4	133134

FIGURE 4.4: Tiktok Popular Songs 2022 sample

the Tiktok dataset is high dimensional yet only contains 263 records. The addition of the Twitter dataset is important to determine how well synthetic data generation handles structured sentences and correlations between attributes.

4.2.1.1 Data Preprocessing

In order to achieve adequate model performance, both the Vehicle and Twitter datasets were cleaned and preprocessed before training began. The "fuel", "seller_type", "transmission" and "owner" attributes were removed from the Vehicle dataset to reduce training time and rows that included null values were removed entirely. Duplicate tweet ID's were removed from the Twitter Sentiment Analysis dataset to reduce the size of the dataset from 74682 records to 12447 (a 6x reduction) and hence improve model training time.

4.2.2 Models and Experimental Setup

The structure of the CTGAN model that is trained on each respective dataset is found in *Xu et al's* [XSCIV19] paper from Section 2.5.1.2. Both the generator and discriminator contain two fully connected hidden layers. The generator utilises batch normalisation and Relu activation. The discriminator uses leaky ReLU activation and a dropout rate of 0.2. The models are trained utilising Wasserstein Gradient Penalty Loss [GAA$^+$17] and utilise the Adam optimiser with a weight decay of 1e-6. The CTGAN's are trained with a learning rate of 2e-4, a batch size of 500 and run for a total of 300 epochs each.

For training and testing, the physical equipment used is identical to that found in Section 4.1.2.

For these experiments, we utilise the Synthetic Data Vault (SDV) [PWV16] which gives us access to the pre-built CTGAN model and the ability to sample synthetic data from the fit model.

4.2.3 Process

To begin generating synthetic data, we fit a CTGAN as mentioned in Section 2.5.1.2 to each data set.

```
from sdv.tabular import CTGAN
model = CTGAN()
model.fit(data)
```

Once fit, we can simply call the CTGAN.sample method to generate a new data frame with identical size to the original dataset and save it as a csv file using pandas.

```
import pandas as pd
new_data = model.sample(num_rows=len(data))
new_data.to_csv("synthetic_data.csv")
```

4.2.4 Evaluating the Performance of Synthetic Data

To conduct an evaluation of the similarity between the synthetic and real data, we utilise the TableEvaluator package from pypi.

```
from table_evaluator import load_data, TableEvaluator
real, fake = load_data('real.csv', 'fake.csv')
table_evaluator = TableEvaluator(real, fake)
table_evaluator.visual_evaluation()
```

This provides a visual presentation of a variety of metrics such as the mean and std's of the real and fake data, cumulative sums per feature, the distributions of each feature and a confusion matrix for the real and synthetic datasets.

As previously mentioned, it is necessary that whilst containing entirely "fake" records, the synthetic data must also maintain the same properties as the original dataset. In order to determine how well these properties are maintained, we additionally conduct a variety of tests using metrics that compare the real dataset against the new, synthetic dataset. These tests are as follows:

4.2.4.1 Statistical Metrics

To compare the statistical properties of the datasets, we use two goodness-of-fit hypothesis tests on each attribute to compare the original dataset to the new synthetic data. Both the Kolmogorov-Smirnov

(KS) test [Mas51] on the numerical attributes and the Chi-Squared (CS) test [Pea00] on the categorical attributes are utilised. Where the KS test compares the distribution of continuous columns, the CS test instead measures the probability that two columns were sampled from the same distribution. This gives an effective measure as to comparing the similarity of each individual attribute, the aim is for the synthetic data to be as statistically close to the original dataset as possible.

```
from scipy.stats import ks_2samp
return 1 - ks_2samp(real_data, synthetic_data)

from scipy.stats import chisquare
return chisquare(f_observed, f_expected)
```

As shown above, we can import both the 2-sample KS test and the chi-square test from scipy.stats. However, the chisquare test requires us to compute the observed and expected frequencies for each real categorical value. To compute this, we use a helper function from SDV, found in Figure 7.4 of the Appendix. The output will be a value between 0 and 1 for both tests, with scores closer to 1 indicating higher statistical similarities.

4.2.4.2 Likelihood Metrics

Here, we aim to measure the potential for each row in the synthetic data to belong to a distribution learned from the real data. This is done by fitting a Bayesian Network [4] to the original dataset and evaluating the average likelihood that any row belongs to it. The aim is to determine how likely the synthetic data is drawn from the same distribution as the original dataset. The distribution in this case is learned from fitting the Bayesian Network to the dataset. To conduct this test, we utilise the BayesianNetwork class from the *pomegranate* package in Python. This particular implementation of a Bayesian Network to compare two datasets is implemented by SDV and is run as shown below. The output here is a value between 0 and 1, indicating the average probability that a row from the synthetic data is drawn from the same distribution as the original data. As such, values closer to 1 would show higher likelihood for the synthetic data to belong to the learned distribution.

```
from sdv.metrics.tabular import BNLikelihood
BNLikelihood.compute(real_data, synthetic_data)
```

[4] A Bayesian Network can be visualised as a Directed Acyclic Graph that represents a set of variables and their conditional probability distributions.

4.2.4.3 Detection Metrics

Detection metrics aim to determine how easy it is to predict if a random data example is synthetic or real. This is done by attempting to train a machine learning model on the shuffled, combined data, where each row is given a boolean label as to wether it is real or synthetic. Clearly, if a machine learning model is able to easily determine the source of data, the synthetic dataset must fail at representing the original dataset sufficiently enough. Hence, we aim for any model attempting to detect the source of data to have accuracy no better than chance (50%). Here, we train the model using Logistic Regression from the *scikit-learn* Python package. Once again, we utilise SDV's implementation of this metric as shown below. The output of this metric is the average ROC AUC score for the learned classification model. As such, values closer to 1 would indicate accuracy roughly equal to chance.

```
from sdv.metrics.tabular import LogisticDetection
LogisticDetection.compute(real_data, synthetic_data)
```

4.2.4.4 Machine Learning Efficacy Metrics

Next, we measure how effective the synthetic data is at replacing the original dataset in machine learning problems. That is, we formulate a machine learning problem where we aim to predict a column given a set of other columns in the synthetic data and extrapolate the model to the original dataset. The aim here is for the synthetic data to be a sufficient enough replacement of the original dataset for a machine learning problem. One important factor to consider is the machine learning problem considered must be solvable in the original dataset, or else this metric is irrelevant considering it would not be possible for the model to become more accurate on the synthetic data. Considering the aim of this experiment is to determine how suitable synthetic data is at replacing original data, this metric is vital in determining its overall success. We choose the type of machine learning model here to be a *Multilayer Perceptron* from *sklearn.neural_network* to handle the cases of both binary and multi-class classification. As before, we implement the metric from SDV shown below.

```
from sdv.metrics.tabular import BinaryMLPClassifier, MulticlassMLPClassifier
target = "selling_price" #assuming this column is private
BinaryMLPClassifier.compute(real_data, synthetic_data, target=target)
MulticlassMLPClassifier.compute(real_data, synthetic_data, target=target)
```

The output here is a value between 0 and 1 indicating both the success of the models performance on the real data after having trained on the synthetic data and the difficulty of the machine learning problem established.

```
train = real_data.sample(int(len(real_data) * 0.75))
test = real_data[~real_data.index.isin(train.index)]
LinearRegression.compute(test, train, target='selling_price')
```

We also measure the difficulty of the machine learning problem by splitting the real dataset into train and test sets as shown in the code snippet above. This allows us to accurately measure the efficacy metric by determining the influence the difficulty of the machine learning problem has on the result. We present this value alongside the original efficacy metric as a means of comparison.

Whilst this measure seems similar to the Detection metric above, it varies in that here we attempt to train on the *synthetic data* and predict or classify on the *real data*. If we were to both train **and** test on the synthetic dataset and compare to the same process on the real dataset as before, it would just be another detection test, as if the outputs were to be different, they would generally be from opposite (real vs synthetic) datasets, giving an easy form of detection.

4.2.4.5 Privacy Metrics

To determine the extent at which the synthetic data reveals sensitive information about the original dataset, we explore a similar problem to the efficacy metric. We assume one attribute of the data to be considered "private", with the intention of preventing this column being leaked to a capable adversary and fit an adversarial attack model on the synthetic data. Here, the adversary has full access to the synthetic dataset and trains a model with the intention of extending it to the real data and extracting personal information by predicting the private attribute. As such, we evaluate how well the adversarial model performs on the original dataset, whereby the aim is that given non-private attributes, the model is not able to predict the sensitive column. The difficulty here is implementing adversarial models on a set of columns that contain both categorical and numerical attributes. For simplicity, we conduct this metric only on numerical columns, utilising again a *Multilayer Perceptron* for this task. The code is run as shown below, with the key fields and sensitive fields being columns from the Vehicle dataset.

```
from sdv.metrics.tabular import NumericalMLP
NumericalMLP.compute(
              real_data,
```

```
synthetic_data,
key_fields=['km_driven', 'mileage␣(kmpl)', 'engine␣(CC)', 'max_power␣(bhp)'],
sensitive_fields=['selling_price']
)
```

The output here is 1 minus the success rate of the adversarial model on the real data. Hence, values closer to 1 are ideal, indicating that the adversarial model has increased difficulty in predicting the sensitive field in the real data.

4.3 Generating Differentially-Private Synthetic Data

Due to some of the results from the previous section, the need for Synthetic Data that also satisfies concerns for privacy is evident. We already know (from the results presented in Section 5.2) that synthetic data has the capability to replace real data from an accuracy standpoint, however fails to provide any meaningful privacy guarantees on individuals in the original dataset. Here, we aim to showcase the potential for private synthetic data to solve the issues of both adversarial attacks and machine unlearning and thus, demonstrate the tradeoff that occurs between protecting privacy and maintaining accuracy against the original dataset.

4.3.1 Dataset

For this experiment, we use the Vehicle dataset. The details of this dataset are mentioned in Section 4.2.1. This dataset was chosen as it performed the best in the accuracy metrics in the previous experiment when generating synthetic (non-private) data.

4.3.2 Gretel.ai

To train generative models with differential privacy, we utilise Gretel.ai [5], which is a new platform designed for generating synthetic data with privacy guarantees in mind. The gretel.ai dashboard simplifies the process of developing differentially-private synthetic data by making the process as straightforward as creating a model, providing a dataset and editing a config file for specific use. Once trained, the model can be used to generate private, synthetic data of any size. Alongside this, gretel.ai also provides a report with a synthetic data quality score and a privacy protection level, giving a general idea as to its

[5] https://gretel.ai/

success. Gretel provides users with 30 free credits per month, however additional credits were provided at no charge to assist with our extensive experimental process.

4.3.3 Models and Experimental Setup

The generative model type in this experiment differs to that found in the previous experiment. Instead, we utilise a Long Short-Term Memory (LSTM) model to both compare its performance to the CTGAN used previously and to allow for an easier Differentially-Private training process. Since Gretel.ai have implemented a DP-LSTM model that allows for simple parameter tuning, it was an obvious choice to integrate it into our experiments. A custom config was created to suit our needs and is found in the Appendix Figure 7.1. The LSTM model has 256 RNN units, and trains for 50 epochs with a batch size of 4. To keep consistency with the CTGAN parameters, we also utilise a dropout rate of 0.2 and a learning rate of 2e-4.

The training process is conducted on the Gretel.ai servers, whereas the equipment used to test the model is found in Section 4.1.2.

4.3.4 Generation Process and Collecting Results

Gretel.ai provide both video and Google Collab tutorials as to how to generate differentially private synthetic data with varying noise levels. At a high level, an LSTM is trained using Differentially-Private Stochastic Gradient Descent on the real dataset, then given some seed column, can generate fake records that maintain properties of the original dataset. Once the LSTM is trained, the gretel console can be used to generate a dataset of any size. For the purposes of collecting results, the synthetic dataset generated is of the same size as the original dataset.

Alongside Differential Privacy, Gretel also adds the ability to add two privacy filters to the generation process, namely similarity filters and outlier filters. Similarity filters ensure that no generated record is overly similar to a duplicate of a training record. Here we specifically chose a level of medium which prevents exact duplicates between the synthetic and real dataset. Outlier filters ensure that no synthetic record is an outlier with respect to the original dataset. Recall that outliers have the potential to reveal sensitive information through Membership Inference Attacks. Here, we choose to filter out records that have a high likelihood of being an outlier. These add an additional dimension of privacy whereby an adversaries capability to extract sensitive information is significantly diminished. Adding these two

filters can only improve privacy, and cannot degrade any of the DP guarantees due to the postprocessing process of DP.

To test and analyse the success of the synthetic data, we use the same metrics as those found in Section 4.2.4, along with TableEvaluator figures to provide a visual overview of data performance. As Gretel.ai also provides its on synthetic quality report, we can reference its scores to determine areas where the data performed well or perhaps failed. These scores can be found in the sample Gretel Synthetic Report in the appendix.

CHAPTER 5

Results

This section first presents the results of conducting membership inference attacks on machine learning models trained on the CIFAR-10 dataset. The results are presented primarily through two important metrics mentioned in Section 4 being AUC-ROC and Attacker Advantage.

Next, we explore how effective synthetic data is at replacing real data. The results are presented by examining a variety of metrics explained in Section 4.2.4.

Finally, we examine the effectiveness of differentially-private synthetic data in replacing real data through both privacy and accuracy metrics, where the trade off between these two measures is analysed.

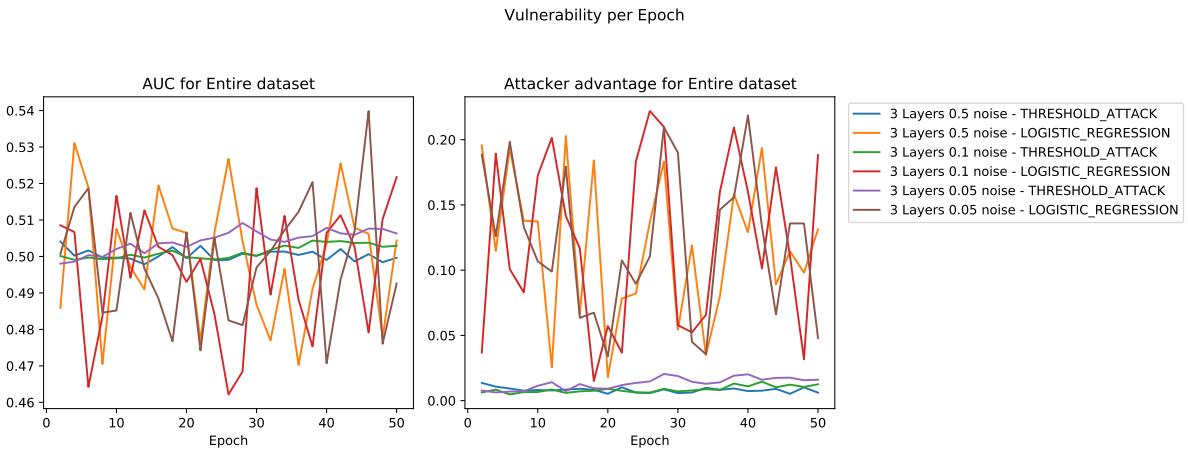


FIGURE 5.1: MIA Vulnerability per Epoch high noise

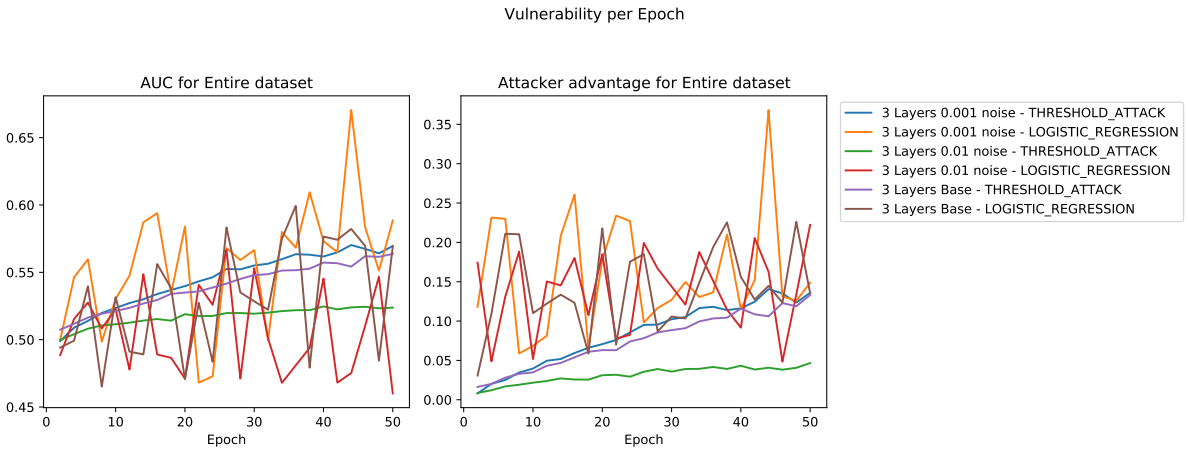


FIGURE 5.2: MIA Vulnerability per Epoch low noise + no DP

5.1 Attacking Machine Learning Models

5.1.1 Vulnerability per Epoch

The first metric we examine is the change in privacy leakage over time. This is shown in both Figures 5.1 and 5.2. As models with higher noise are trained, we see no clear increase in vulnerability, however in both low-noise environments and the base model, it is clear that vulnerability trends upwards over time. This is true for both the threshold and logistic regression attacks. Considering that MIA's exploit the loss difference between members and non members, as the model trains, this difference becomes more clear and hence privacy leakage is expected to increase over time. The models *3 Layers Base* and *3 Layers 0.001 noise* have a very clear linear relationship between privacy leakage and epochs and perform very similarly specifically in the threshold attack, however a general upwards trend is also visible in the logistic regression attack on both models. The general instability of the logistic regression attack can be attributed to its nature. Since the attack relies on training a shadow model based off the outputs of the target model, we can expect some instability in its performance, however it is clear to see that for models trained with little or no privacy protection, the higher values of these plots occur towards the end of training. When analysing these unstable logistic regression curves, we consider the peak values, as they represent the worst-case scenario privacy leakage of the model. In Figure 5.1 as expected, vulnerability does not increase over time at all, attributed to the effectiveness of the DPSGD process and its ability to stop the MIA from leveraging differences in losses to make membership predictions.

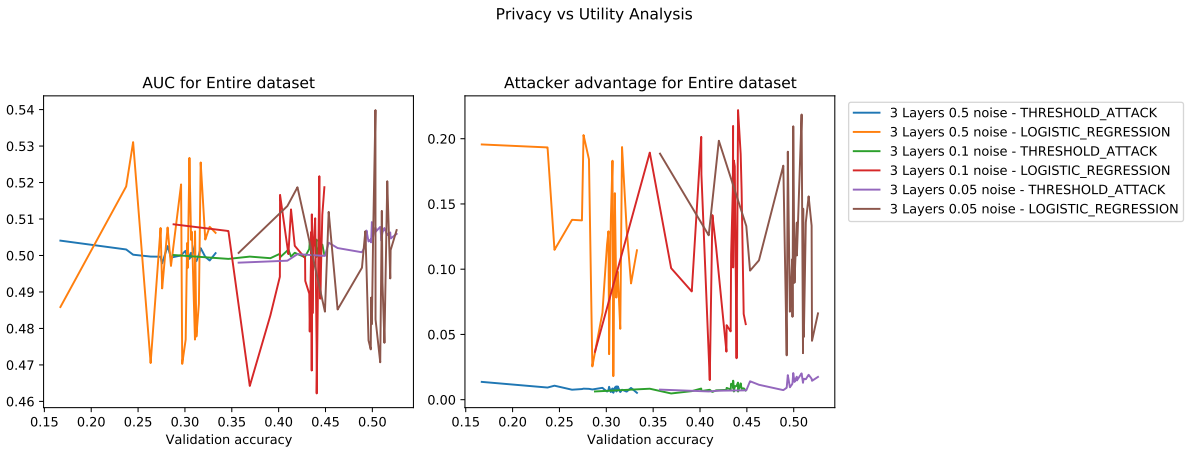


FIGURE 5.3: MIA Privacy vs Utility high noise

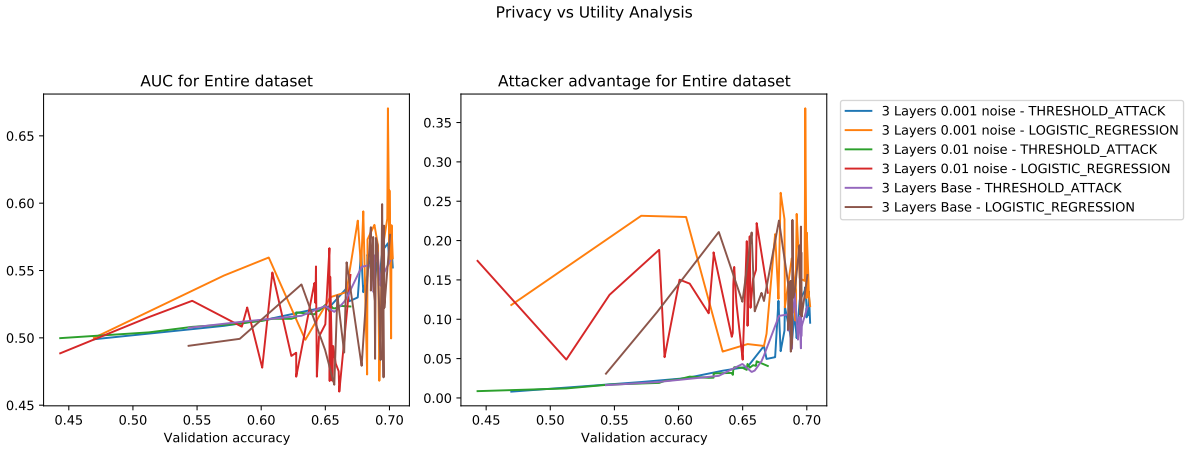


FIGURE 5.4: MIA Privacy vs Utility low noise

Model Name	Training Time (s)	ϵ	Test Accuracy (%)	Max AUC	Max Attacker Adv
3 Layers 0.5 noise	1003	10.7	0.33	0.56	0.26
3 Layers 0.1 noise	991	1398288	0.45	0.57	0.26
3 Layers 0.05 noise	986	10773104	0.53	0.60	0.29
3 Layers 0.01 noise	971	310773104	0.67	0.66	0.37
3 Layers 0.001 noise	968	31248273104	0.70	0.69	0.42
3 Layers Base	832	∞	0.70	0.69	0.42

TABLE 5.1: Membership Inference Attack Results

5.1.2 Privacy vs Utility

The second metric examined is the trade off between model performance and privacy. Figures 5.3 and 5.4 demonstrate this performance. As the value of ϵ is lower corresponding to a lower privacy budget, model performance significantly lowers, becoming as low as 0.33 for our strictest model *3 Layers 0.5 noise*. However, the introduction of noise has also resulted in significantly less privacy leakage as the gradient of the threshold attack plots in Figure 5.3 is flat. For models trained with lower noise values, this gradient becomes higher as validation accuracy increases, meaning that small increases in model performance are met with what appears to be exponentially higher increases in privacy leakage. Figure 5.4 demonstrates this phenomenon as model accuracy passes approximately 0.65, whereby the gradient passed these accuracy values becomes much larger in the threshold attacks. This is also the case for the logistic regression attack, where the higher peak values are seen towards higher validation accuracy's in both the AUC and Attacker Advantage plots when little or no privacy protection at all was used to train the model.

Table 5.1 further demonstrates the trade off between privacy and utility. As the value of ϵ increases, we see an increase in three values, being model accuracy, max AUC and max Attacker Advantage, demonstrating that as ϵ lowers, not only does model performance lower, but an adversaries ability to conduct a MIA becomes significantly harder than before. Hence, we seek to find a natural middle ground where we see large improvements in privacy leakage with a relatively small hit to model accuracy.

5.1.3 ROC Curves

Another important metric in examining the effectiveness of differential privacy in preventing privacy leakage within machine learning models is the Receiver Operating Characteristic Curve and the corresponding area under the curve. As we vary the level of noise in the DPSGD process, we can examine the ROC curves for the best attacks on each respective model. The process of plotting these ROC curves is seen in the code snippet 7.2. As shown in Figure 5.5, as the noise level is increased from left to right, the ROC curve moves closer towards the centre line, meaning the attacker has increased difficulty in predicting the membership of a data sample.

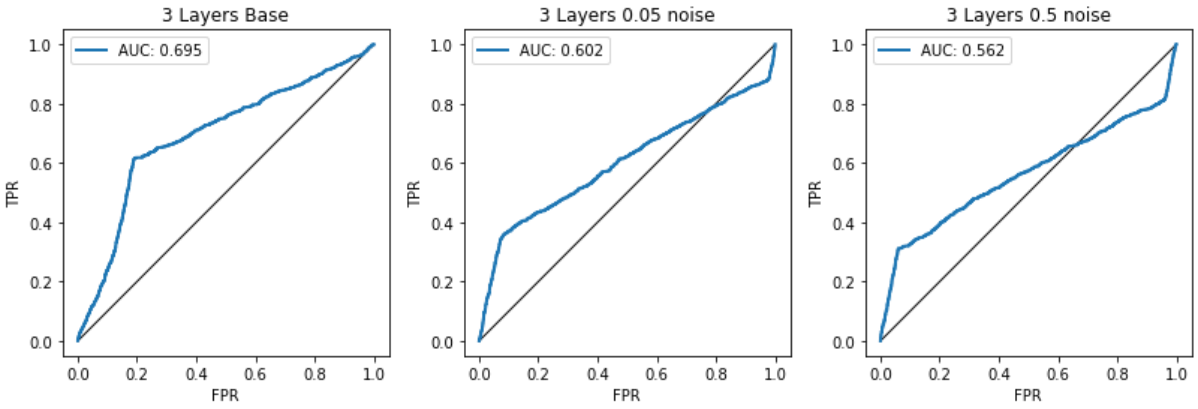


FIGURE 5.5: ROC curves with varying noise

5.1.4 Discussion

From the mentioned metrics above, it is obvious that differential privacy has a significant effect on both model performance and privacy. If we only consider the base model trained with no privacy protection, it is clear that an adversary can easily attack the model and discover information about the training dataset which may be sensitive. This means that any machine learning models created should be trained using *Differentially Private Stochastic Gradient Descent* with a sufficiently low ϵ value to protect against adversaries. However one may ask exactly what this ϵ value should be? As expected, this value would vary significantly between model types and datasets, as it is easier to protect against models who have less of a tendency to overfit their training data. Thus, models trained with regularisation techniques such as dropout will perform much better against membership inference attacks then those trained without. Furthermore, models trained on datasets with a higher number of features will naturally have higher levels of overfitting and as a result, metrics such as the AUC-ROC and Attacker Advantage will be larger. Hence, to find an adequate ϵ for a specific use case, individual experiments should be conducted.

5.1.4.1 Best Model

The best performing model of the collection is one that maintains relatively high accuracy while having significantly lower levels of both AUC and Attacker Advantage when compared to the base model. From figures 5.2 and 5.4, we can see the model *3 Layers 0.01 noise* has a very flat gradient in the threshold attack plot and relatively low peaks in the logistic regression plots, meaning the introduction of noise has resulted in quite a large level of protection against privacy leakage. However, it is worth noting that Table 5.1 shows a max AUC and max Attacker Advantage of 0.66 and 0.37 respectively which is

not far off the base values of 0.69 and 0.42. But this can be attributed to the variance of performance in the shadow model corresponding to the logistic regression attacks and examining both the threshold and logistic regression attacks is important in criticising the privacy leakage of a model. The model suffers only a 0.03 reduction in validation accuracy, which is an extremely small price to pay for the increase in privacy protection. From this, we know the best performing model is one that has the highest validation accuracy whilst maintaining a low gradient in the threshold plot and low peaks in the logistic regression plot. *When the gradient becomes too high, we know the model is gaining small increases in performance at the cost of exponentially increasing costs in privacy and hence, we seek to avoid models with this property.*

5.1.4.2 Training Times

As seen in Table 5.1, there is two clear conclusions that can be drawn from the training time results. The first is that the DPSGD process itself contributes quite extensively to training time, as there is a jump of 136 seconds between the models *3 Layers Base* and *3 Layers 0.001 noise* with identical accuracy and privacy results. This is a 15% increase in training time with no meaningful positive outcome. When training deeper models, such an increase in training time can be extremely costly in terms of not only time, but money too. The second conclusion is the relationship between the level of noise added and the training time. *As the value of ϵ lowers, the training time increases meaning the more private the DPSGD process is, the more expensive it is to train.* This proves to be an issue for model creators, as not only do they need to pay accuracy costs when conducting privacy protection measures, but also resources costs.

5.1.4.3 Implications

The results of this specific experiment have considerable implications for both the users and creators of machine learning models. Any user who participates in a private dataset that is used to train a machine learning model should expect that an adversary is unable to extract their private data through adversarial attacks. Any creator of a machine learning model should hold the responsibility of protecting the privacy of individuals who contributed to the training of that model. The problem here is in training with such protection as to not significantly inhibit the performance of these models, which could have serious implications particularly in areas such as medicine if the model can not be relied on to make accurate predictions. If it is not possible to make well-performing models that also protect the privacy of individuals in its training set, then perhaps other solutions must be examined.

Since it is known that machine unlearning has the potential to worsen the privacy of models without protective training methods, this demonstrates further the necessity of utilising DPSGD in the training process. However as shown, DPSGD is an expensive process as it both lowers model accuracy and increases training times. This leads to even further pressure placed on model owners, either forcing the use of expensive protection methods in model training or looking towards other potential solutions such as synthetic data.

5.1.4.4 Improvements to Experiment Design

As is the case with machine learning, resource and time constraints pose a significant issue in conducting a valid and effective experiment. As a result, a variety of improvements could be made if the experiments were to be repeated with increased equipment or a longer time frame.

- **Models:** With increased model accuracy comes increased model complexity and training time. One of the top performing CNN's for the CIFAR-10 dataset is EfficientNetV2 [TL21], with 121 million training parameters. Obviously incorporating the DPSGD process into a state-of-the-art model such as this would result in a clearer trend between privacy and accuracy that is perhaps more relevant considering these models are at the forefront of machine learning research, however considering the equipment necessary to train these models, a much more efficient method is to create relatively simpler models that although do not perform as well, still clearly show the tradeoff between ϵ and model performance. Although, further experimentation could be conducted with deeper models that are still relatively easy to train. Further, a repeat of this experiment could involve training the models with various regularisation and optimisation techniques to better improve both model accuracy and prevent over fitting, which inherently worsens the success rate of membership inference attacks against the model. Adjusting model parameters such as learning rate and batch size may also have some impacts upon model performance.
- **Datasets:** Expanding the experiment to include other datasets such as MNIST [1][Den12] or CIFAR-100 [2] [Kri09] would benefit in showing privacy-accuracy trends across a broad range of datasets. It is known that datasets with a higher number of features such as CIFAR-100 have a higher tendancy to overfit the training data and hence, a membership attack against models trained in this case would show higher AUC and Attacker Advantage values. Viewing trends

[1] A popular dataset consisting of 60000 handwritten digits.

[2] Identical to CIFAR-10, but has 100 classes with 600 images each instead.

across these datasets that have differing properties would be extremely valuable, as the effect of the particular properties of each dataset (e.g MNIST's simplicity) could be investigated.

- **Equipment:** More powerful GPU's and CPU's would result in both reduced training time and increased memory, allowing for the training of deeper and more complex models that achieve much higher validation accuracy. However due to resource constraints, these models were trained using Google Collab Pro and the provided equipment. This lead to considerable issues particularly with memory, as training more than two models in the same session was not possible with the given memory amount. As a result, it was only possible to use models with simple architecture, although this was still sufficient enough to demonstrate the effect of differential privacy on the privacy and utility of a machine learning model.

5.2 Generating Synthetic Data

5.2.1 Metrics

In this section, the specific distribution per features graphs are presented in figures 5.6 and 5.7 for the Vehicle and TikTok dataset. The other TableEvaluator metrics for each dataset are found in Figures 7.5-7.10 of the Appendix. Note that the Twitter Dataset metrics are missing. This is due to the dataset containing structured sentences, and is explained further in section 5.2.2.2.

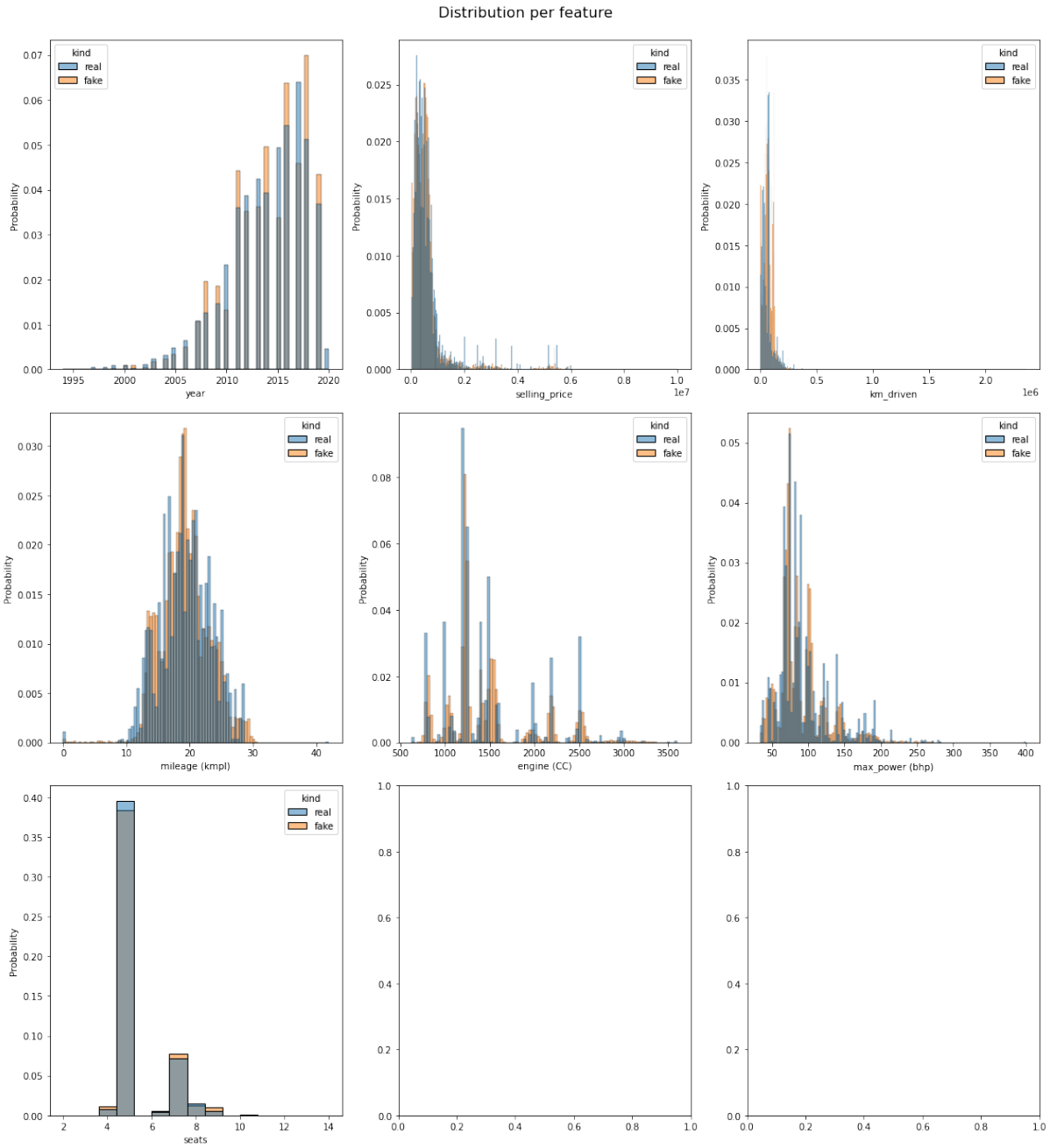


FIGURE 5.6: Vehicle Distribution per Feature

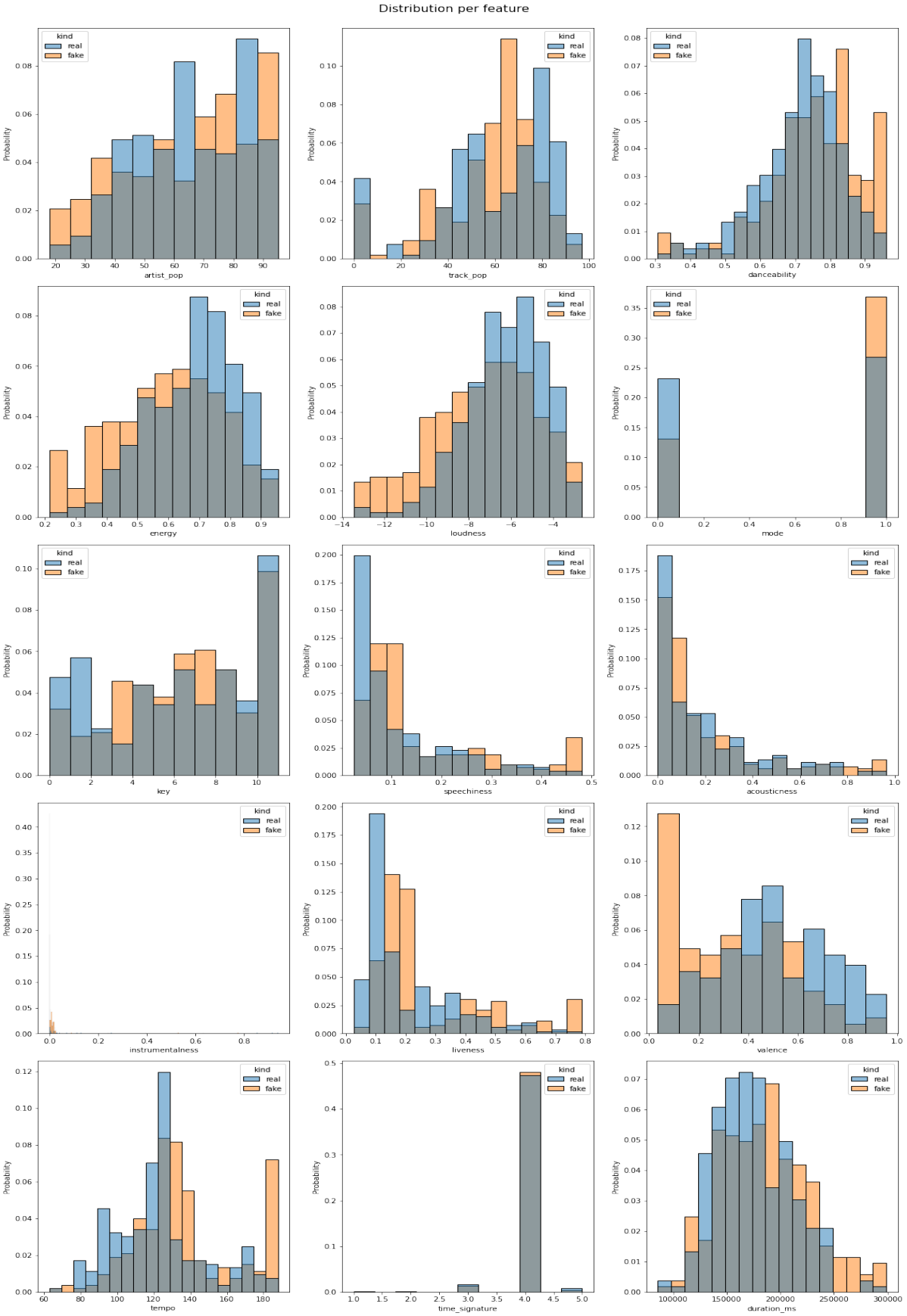


FIGURE 5.7: TikTok Distribution per Feature

Dataset	Statistical	Likelihood	Detection	Efficacy	Privacy	Training Time (min)
Vehicle	0.926 (KS)	0.99	0.92	0.49 / 0.64	0.07	42
TikTok	0.72 (KS)	0.99	0.71	0 / 0.08	0.24	11
Twitter	0.99 (CS) 0.94 (KS)	0.99	0.70	0.29 / 0.54	0.20	243

TABLE 5.2: Performance Metrics for each Synthetic Dataset when compared against its corresponding Real Dataset

5.2.2 Discussion

5.2.2.1 Quality of the Vehicle and TikTok Datasets

To visually analyse the performance of the Vehicle and TikTok datasets, we separate each individual feature column and plot their respective distributions for both the real and fake data. This is shown in Figure 5.6 and 5.7. In addition to this, we conducted a variety of metric testing for each dataset, the results of this are presented in Table 5.2. These metrics are described in detail in Section 4.2.4

In the **Vehicle** dataset it is clear to see that the CTGAN has been able to learn the individual distributions of each feature as the general shape of the synthetic distribution is roughly equivalent to its real counterpart. However, interestingly the CTGAN seems to produce rows spread out around certain peak values rather than imitating the real data. An example of this is in the engine(CC) distribution, there are a large number of synthetic rows spread around 1500, 2000, 2500 which are large peak values in the real data. This indicates that the CTGAN has some variability in its learning or generative processes which may in fact benefit the privacy of future models trained on the synthetic set as it lowers the potential for memorisation.

As shown in table 5.2, the **Vehicle** dataset performs extremely well in four out of the five metrics being statistical, likelihood, detection and efficacy. This can be attributed to the structure of the dataset, being well suited for a CTGAN. However, when running privacy tests, the metric returned a value of 0.07, indicating the success of fitting an adversarial attack model on the synthetic data and using it to predict sensitive columns in the real data.

On the contrary, in the **TikTok** dataset it is clear there are differences in the distributions of each attribute. However this can be attributed to the low size of the dataset. Although the distributions have these clear differences, the overall shape is still somewhat preserved when comparing the real and fake data. This

demonstrates the power of CTGAN's even when learning on smaller datasets. However, due to the lower size of the dataset, it is expected the fake data distributions would fluctuate greatly if the data were to be generated again.

The **TikTok** dataset performs relatively well in Statistical, Likelihood and Detection metrics, achieving scores of 0.72, 0.99 and 0.71 respectively, however fails in the Efficacy metric as the lack of data results in an unsolvable machine learning problem. This demonstrates the need for large amounts of data if the intention is to solve machine learning problems with said data. Despite this, the ability for the CTGAN in this case to generate data that maintains statistical properties is promising. The data also achieved a relatively high privacy result of 0.24, but this can be attributed to a variety of factors.

Oftentimes there is a trade off between the quality of the synthetic data and the privacy of the original dataset, as the adversarial model can maintain higher accuracy on the original dataset if the synthetic dataset is of adequate quality. This is obvious when comparing the privacy scores for the Vehicle and TikTok datasets of 0.07 and 0.24 respectively. One would assume upon first glance that the TikTok data is more private, this may be true but is misleading as the poorer quality synthetic data has resulted in lower adversarial attack performance, leading to increases in privacy. As a result, the privacy metric is better viewed alongside the other metrics, giving a clearer view of how successful the synthetic data is at protecting against adversarial attacks.

5.2.2.2 Quality of the Twitter Dataset

Initially the Twitter dataset was chosen to determine how well a generative model (namely a CTGAN) could learn dependencies between attributes, considering the feature columns of "Game" and "Sentiment" are completely dependant on the actual tweet. Despite achieving good scores in statistical, likelihood and detection metrics as shown in table 5.2, the generative model failed to produce rows of data with valid column combinations. This is seen clearly in Figure 5.8. The fifth rows tweet clearly references the game *Pubg*, however the row was incorrectly generated with the game *Hearthstone*. Furthermore, the first rows sentiment is *Irrelevant* but the tweet is clearly positive. As a result, the CTGAN must have failed to learn the cross-dependencies between attributes. In addition to this, it is obvious that the CTGAN has memorised the tweets in the original dataset, as the synthetic data contains primarily sentences that are present in the original dataset. Obviously this is a privacy issue, as if we considered this attribute to be private, the CTGAN has done nothing to protect the original information and instead

	ID	Game	Sentiment	Tweet
0	0	Fortnite	Irrelevant	great stream, today. thank you all for coming ...
1	1	TomClancysGhostRecon	Negative	CS:GO: Adds new bench on mirage. . Smileybs (o...
2	2	Overwatch	Positive	This is really interesting for indie RPGs with...
3	3	PlayerUnknownsBattlegrounds(PUBG)	Negative	RT @richardturrin: Amazon and Goldman partner....
4	4	Hearthstone	Positive	Miss U Pubg . . . .

FIGURE 5.8: First 5 rows of the Synthetic Twitter Dataset

has memorised it in the learning process. Recall that any time a machine learning model memorises information, it is easy for a capable adversary to extract that information.

Clearly, this synthetic data would be irrelevant in training a model to the same effect as the original dataset due to two primary concerns:

- Model's trained on the synthetic dataset have significantly lower **accuracy**. Recall that this dataset is used to train models that predict the sentiment given a certain tweet. If the synthetic data is invalid, it would be impossible for a model to learn correlations between tweets and sentiments. This reduction in accuracy is seen in the Efficacy metric of 0.29 in table 5.2. Here, the model performs approximately at chance prediction, since there are 3 possible values for the *Sentiment* attribute. However, it performs significantly better at 0.54 when training exclusively on the original dataset. This shows how the machine learning problem becomes unsolvable when replacing the original dataset with the synthetic dataset.
- The rows in the synthetic dataset are vulnerable to **privacy** attacks. As mentioned, this is due to model memorisation. If rows from the original dataset are extracted from the synthetic dataset easily, the process of training and generating this new data would be useless.

Due to the inability to solve accuracy and privacy issues, this synthetic data would also be unsuitable in solving the problems of *machine unlearning*. As a result, we should look towards other methods of generating synthetic data that better deal with structured sentence data such as language models.

This experiment is of particular importance as a significant amount of tabular data is of this form, where certain attributes are completely dependant on another. Here, the *Game* and *Sentiment* attributes are derived exclusively from the original tweet and as such, a generative model that only draws from a distribution based on columns being independent is insufficient to solve our problems.

5.2.2.3 Training Times

Fitting a CTGAN to a dataset is an exhaustive process, with larger or more complex datasets resulting in considerable training times. Even with significant GPU power provided by Google Collab Pro +, the Twitter dataset had to be trimmed in order for quicker training, yet still resulted in training times of approximately 4 hours. This can be attributed to both the structured sentence data causing difficulties in learning distributions and the size of the dataset. The less complex datasets Vehicle and TikTok achieved lower training times of 42 and 11 minutes respectively, as they contained unstructured, mostly numeric data and were lower in size.

In addition to training a CTGAN, a variety of models were trained to conduct the detection, efficacy and privacy metrics presented in Table 5.2. The details of these models are explained throughout Section 4.2.4. The training times of each model scaled with dataset size and complexity as expected, with some training times taking even longer than the CTGAN fitting process. This is due to the necessity of formulating a machine learning problem for each dataset in order to conduct the efficacy and privacy metrics. Depending on the difficulty of this problem, training times would vary heavily, resulting in these specific performance metrics taking a considerable amount of time to conduct.

5.2.2.4 Implications

The results presented have significant implications on the use of synthetic data in the real world. We examine how well the synthetic data does at replacing real data through both accuracy and privacy metrics. Specifically, the goal here is to use this data to solve the issues associated with both adversarial attacks and machine unlearning. As shown, depending on the properties of the data, the CTGAN has varying performance. When generating synthetic data, the corresponding real dataset should be adequately sized and should not consist of attributes that contain structured data such as sentences. Oftentimes however, datasets dealt with in the real world are both extremely large and complex such as census data, posing issues not only with accuracy but with training times also. On a positive note, once the model is trained it can be saved and loaded at any point, preventing the need for significant retraining.

Although it is possible to achieve high similarity between the real and fake data, the low values recorded for the privacy metric are worrying. Perhaps indicating the necessity of additional privacy measures in the training of the CTGAN. If an adversary has the capability of extracting information contained in the real dataset from the synthetic dataset, we would not be solving the issues of both adversarial attacks

and machine unlearning. In fact, the process of generating this synthetic data would be useless as there are no privacy guarantees for individuals in the original dataset.

5.2.2.5 Improvements to Experiment Design

Despite achieving good results particularly in the accuracy metrics, a number of experimental design changes could be made were the experiments to be repeated in order to both improve results and reduce training time.

- **Models:** In order to generate synthetic data with maximum quality, other methods of generation should be explored. For example, when dealing with structured sentence data, perhaps models typically applied to NLP problems such as Variational Autoencoders [KW13] or transformer based architectures such as GPT-3 [BMR+20] would be better suited. Were these experiments to be repeated, testing how effective these models are at generating synthetic data in the presence of structured or more complex data such as sentences would be highly useful. Furthermore, experimentation could have been done on changing the CTGAN's parameters to determine the effect on both generation quality and training time. A list of these hyperparameters can be found in the SDV user guide. Due to time and resource constraints, exploring the capabilities of these other models and performing additional hyperparameter tuning would have proved too difficult and thus, we instead pose this as future work.
- **Datasets:** Tabular data proved to be an efficient way of experimenting with the usefulness of synthetic data. However, obviously this data type is not reflective of the different data used in real world applications. Image data is extremely popular, particularly in fields such as facial recognition. Demonstrating the capability for synthetic image data to replace real image data particularly in addressing the concerns of adversarial attacks and machine unlearning would be a beneficial area of future work. Obviously, different data types would require the use of generative models applicable to the type of data such as GAN's for images.

5.3 Generating Differentially-Private Synthetic Data

5.3.1 Metrics

In this section, we provide the specific distribution per features graphs for the Synthetic Vehicle datasets trained with varying noise levels in figures 5.9, 5.10 and 5.11. Additional visual results are presented in the Appendix in Figures 7.11 - 7.19.

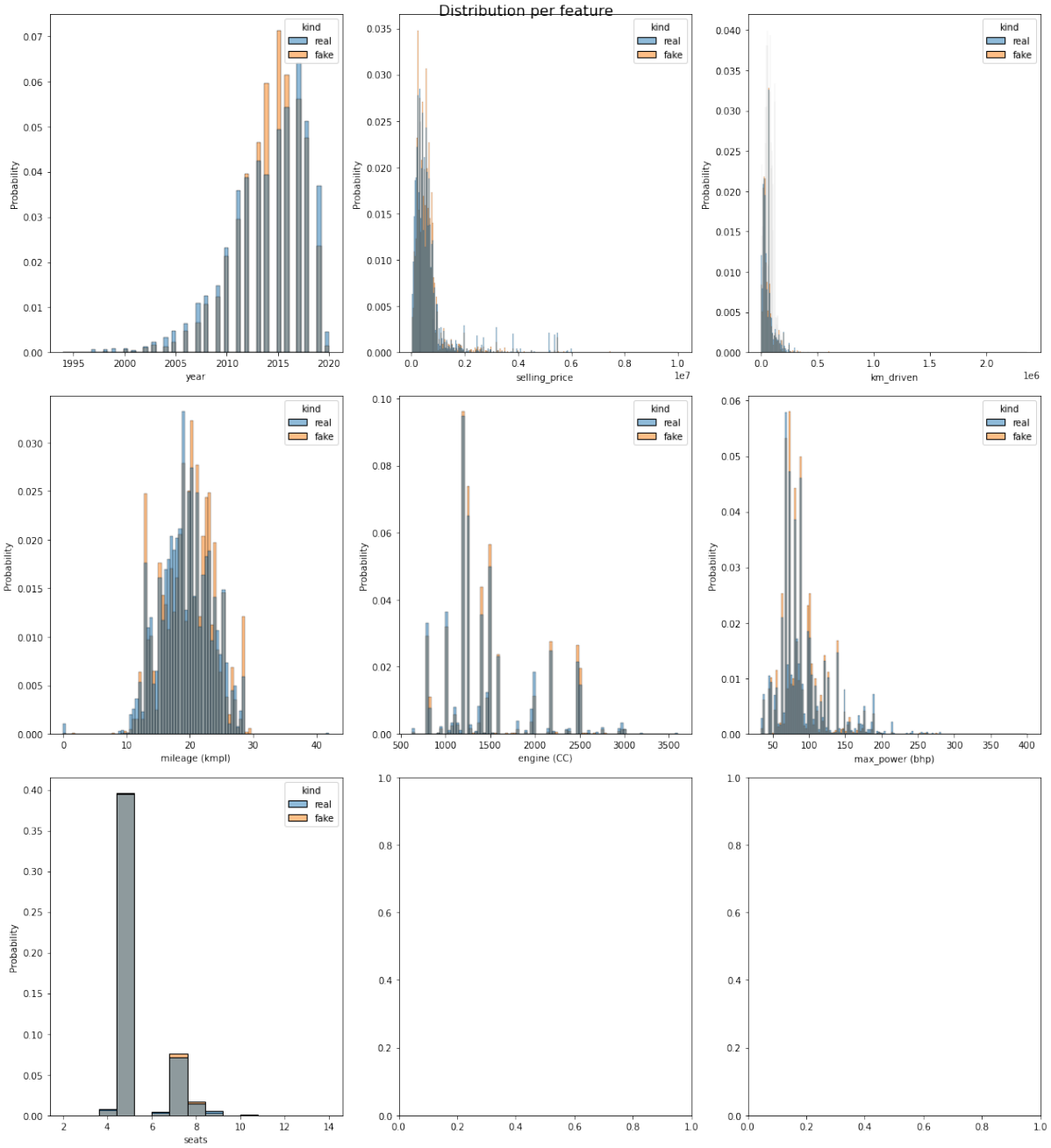


FIGURE 5.9: Vehicle Distribution per Feature, noise multiplier = 0.001

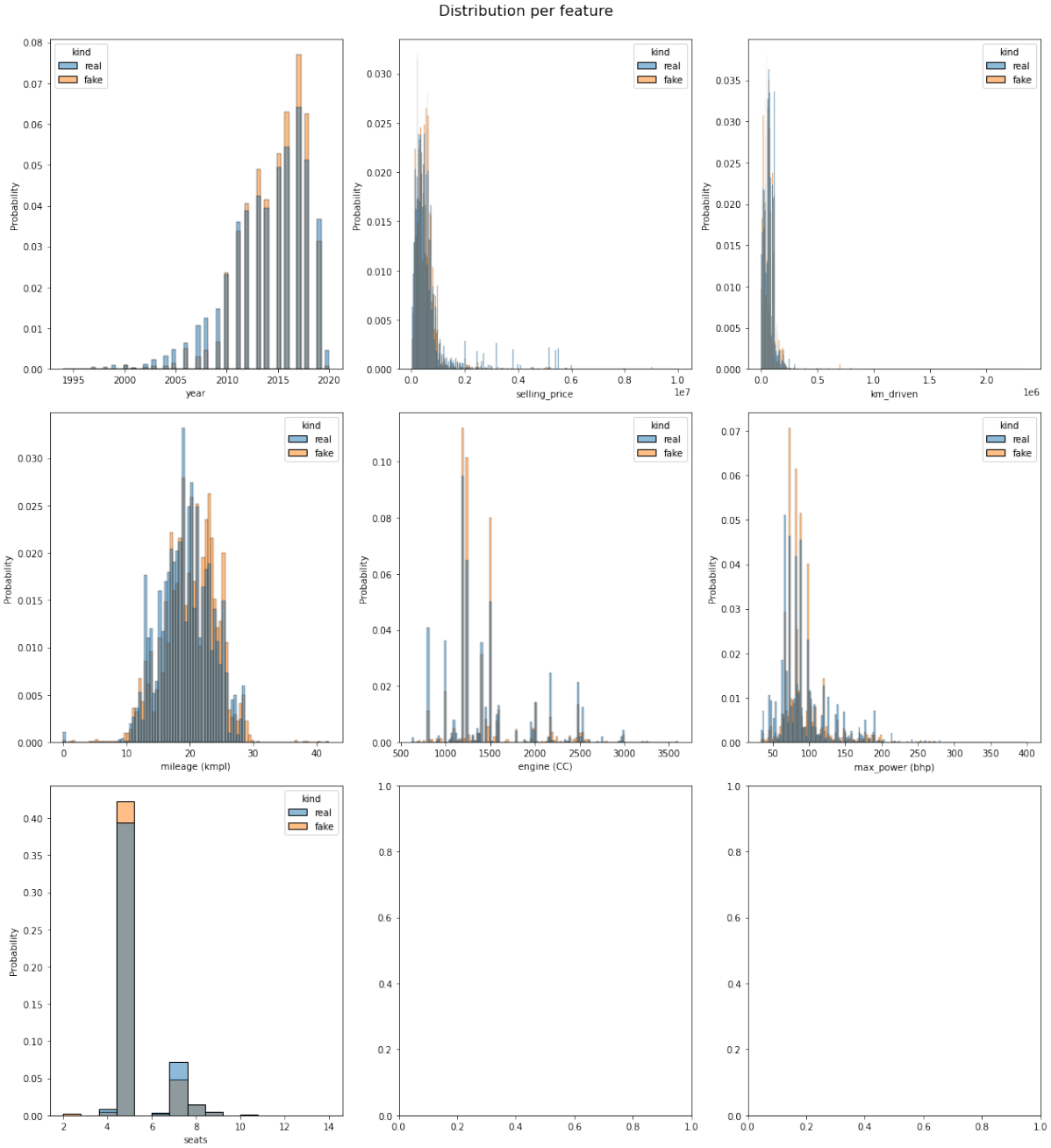


FIGURE 5.10: Vehicle Distribution per Feature, noise multiplier = 0.01

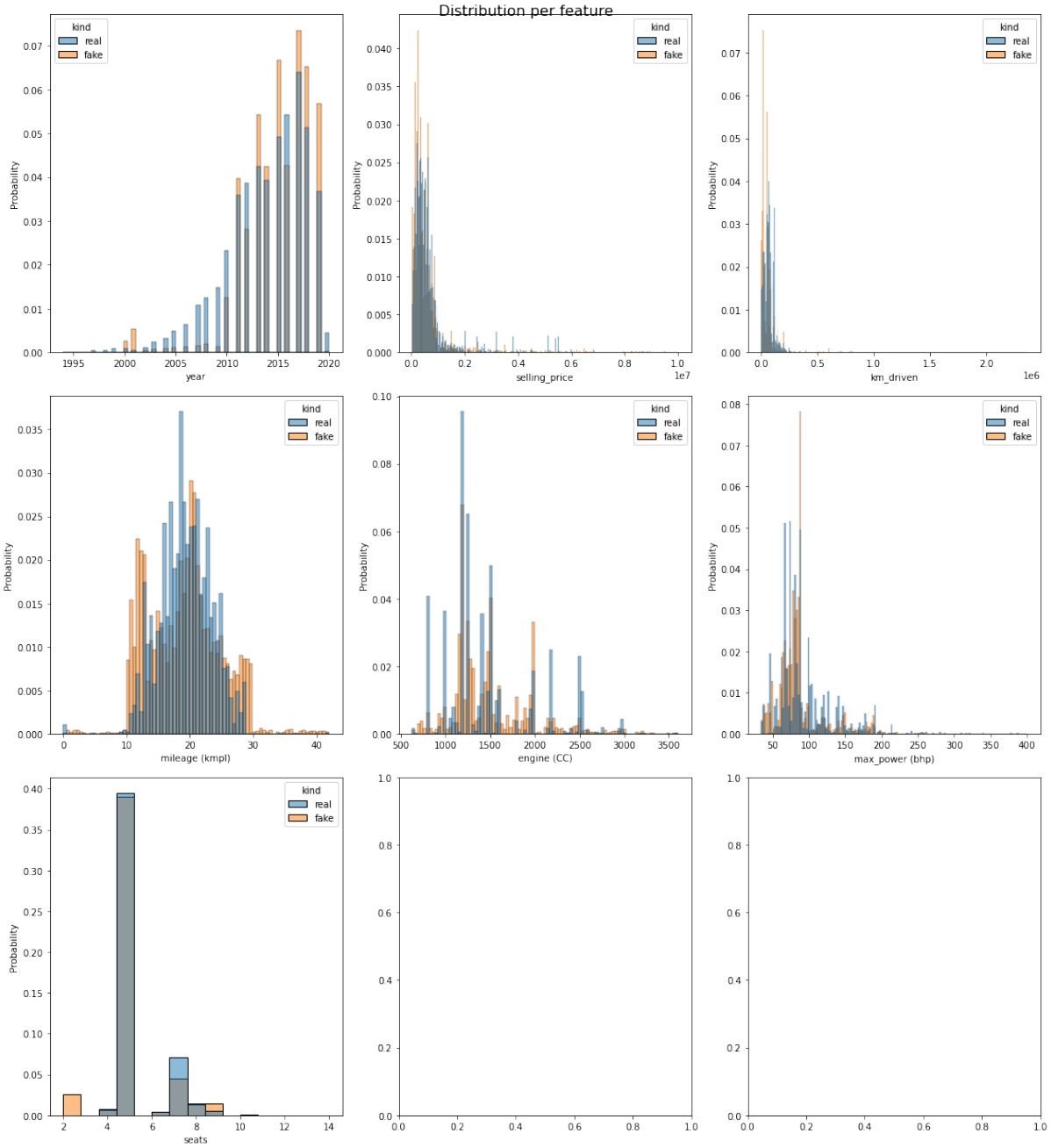


FIGURE 5.11: Vehicle Distribution per Feature, noise multiplier = 0.1

Dataset	Statistical	Likelihood	Detection	Efficacy	Privacy	Training Time (min)
0.001 DP	0.95 (KS)	0.99	0.89	0.61 / 0.64	0.09	15
0.01 DP	0.91 (KS)	0.99	0.78	0.34 / 0.64	0.12	16
0.1 DP	0.86 (KS)	0.99	0.72	0.1 / 0.64	0.14	18

TABLE 5.3: Performance Metrics for the Synthetic Vehicle datasets trained with varying levels of noise against the real dataset

5.3.2 Discussion

5.3.2.1 Effect of ϵ on Dataset Quality

Recall that as the level of noise increases, the privacy budget ϵ lowers, meaning higher noise corresponds to more private datasets. It is obvious from viewing the distribution per features figures 5.9, 5.10 and 5.11 that as we increase the amount of noise, the quality of the synthetic data lowers as expected. Especially in figure 5.11, the distributions of some features seem vastly different which may have detrimental effects when extending the data to real-world use. The extent at which privacy protecting methods impact data quality is further seen in Table 5.3, where there is a general decrease in dataset quality across statistical, detection and efficacy metrics.

One concerning measure is the significant decrease in efficacy scores between the 0.001 and 0.1 DP datasets. This indicates that the addition of noise detriments the datasets ability to solve machine learning problems. It is expected that the noise added during the training process adds enough randomness to the data such that correlations between features are unable to be learnt.

One interesting comparison is between Table 5.3 and the non-DP Synthetic Vehicle Dataset in Table 5.2. The LSTM trained with 0.001 DP performs even better than the CTGAN in statistical, detection and efficacy metrics. This can be attributed to two factors, first being that an LSTM may be better suited for the structure of this data, secondly the injection of noise into the training process can act as a very strong form of regularisation [DFH$^+$15], which improves model performance as it improves generalisation. *This is an exciting result, as even small amounts of noise in the training process can provide strong improvements not only in data privacy, but accuracy also.*

	name	year	selling_price	km_driven	mileage (kmpl)	engine (CC)	max_power (bhp)	seats
0	K.t izleR XCpr	2017.0	1000000	50000.0	12.1	2179.0	79.0	5.0
1	VfgengNole Sag VX	2012.0	450000	10000.0	11.2	1428.0	72.4	5.0
2	CritXrga 1jGpro VSeS	2015.0	155000	15000.0	23.0	1449.0	81.6	5.0
3	Adslirziamrlrc Bl Vp	2011.0	370000	100000.0	23.0	1198.0	63.2	5.0
4	ycisf)dorNel Elvutle 1	2012.0	620000	15000.0	2.0	799.0	120.0	7.0

FIGURE 5.12: First 5 rows of Synthetic Vehicle dataset trained with 0.1 DP

5.3.2.2 Dataset Privacy

As shown in Table 5.3, there is a slight improvement to the privacy metric as the level of noise added during the training process increases. However, this is not the only indicator of privacy as the filters added during the generation process also hinder the capability for an adversary to extract private information. For example, since the amount of outliers in the dataset is limited, the potential for membership inference attacks is lowered. Furthermore, we ensure there are no real records in the synthetic dataset through the use of similarity filters.

One extra added level of privacy is the level of noise added to columns typically considered private such as names or addresses. Oftentimes, as this data is highly unique, generative models will memorise these fields in the training process and repeat them when generating synthetic records. Clearly, this leads to significant privacy leakages as the synthetic data contains exact duplicates of these private fields. As shown in Figure 5.12, the vehicle names are completely unrecognisable from their original names. This has the added benefit of protecting private information in the real dataset as reversing these columns without access to the original data would be difficult. However, it is insufficient to rely on this as an effective form of data privacy as it is unclear at what noise level a capable adversary can begin to reverse the "noisy" attributes and find information present in the original dataset. Instead, we can treat this as an added privacy bonus when training generative models with differential privacy.

5.3.2.3 Training Times

As expected, training time increases as the privacy budget ϵ lowers. This is seen in the general upwards trend in Table 5.3. The fact that these times are significantly lower than those found for experiment 2 can be attributed to two main factors.

- The training process of an LSTM is vastly different to that of a CTGAN. In this case, it may be that the LSTM trains faster due to the structure of data.
- Due to Gretel.ai kindly providing access to their servers with free credits, the physical hardware would be different to that used in experiment 2, resulting in differing training times.

5.3.2.4 Implications

The results in this section have considerable implications for the practicality of this data in the real world. We showed that low noise levels in the DPSGD process with added privacy methods such as similarity and outlier filters are promising. However, the addition of excessive noise results in significant degradation of data quality, particularly in the efficacy field as the ability to solve machine learning problems is diminished. As a result, choosing a noise level that results in both accurate and private data with respect to the original dataset is vital. Furthermore, since the DPSGD process is resource intensive, choosing lower noise levels would result in lower costs for model creators.

One fact to consider is that this data is essentially infinite, once the generative model is trained it can be queried to generate as much data as required. As such, platforms like Gretel.ai are an exciting form of synthetic data creation as they allow for privacy protection methods both in model training and data generation. Our goal was to determine if Differentially Private Synthetic data could solve the issues associated with both adversarial attacks and machine unlearning, we showed for our specific case that this generation method can provide benefits to both accuracy and privacy when compared to standard synthetic data. As a result, it should be obvious to extend this generative method for practical use, at least with tabular data and provided the computational resources to do so.

From a *Machine Unlearning* standpoint, since we are guaranteed the synthetic data contains no real records, we are only required to ensure that an adversary can not extract private information from the original dataset. If this is possible, there would be no need for unlearning to occur, saving model owners the extensive resource costs as mentioned in Section 2.3. This is an exciting prospect as the laws surrounding Machine Unlearning restrict model owners capabilities at making well-performing models that can benefit society in meaningful ways. It should be noted that once data is used to train a generative model, it must be securely deleted or hidden such that no individual can gain access to the real dataset, this ensures model owners comply with the laws surrounding the "Right to Forget" (as addressed by Machine Unlearning).

5.3.2.5 Improvements to Experiment Design

There are a number of improvements to be made to experimental design such that the success of Differentially Private Synthetic Data can be better measured. Since the design of the experiment is relatively similar to experiment 2, the experiment improvements mentioned in Section 5.2.2.5 are still applicable which include utilising different model types and also generating data from both tabular datasets with different structure and non tabular datasets such as images. However, we also propose further improvements involving Gretel.ai in particular.

- **Model Parameters:** Experimenting with varying parameter values for LSTM training in the config file could have been beneficial in improving both the accuracy and privacy of the synthetic data. This includes parameters such as RNN units or dropout rate, where changing these values can have significant effects on the performance of the model.
- **Privacy Filters:** Experimenting with the effect of the inbuilt privacy filters in Gretel.ai on data quality would be valuable. As we utilised only medium filter settings in this experiment, changing filter settings as well as removing the filters entirely and determining the overall effect on privacy would be an effective study. We propose this as an area of future work in Section 6 as this process would involve further complex experimentation.
- **Further Privacy Analysis:** Considering that privacy is a multifaceted measurement, the privacy metric recorded in this experiment is not the only indicator as to how private the data is. Particularly if individuals do not have access to any of the original data, we instead measure privacy by the extent at which information from the original dataset is leaked following potential attacks from a capable adversary who has access only to the Differentially Private Synthetic dataset. As such, if this experiment were to be repeated, it would be beneficial to record performance in other areas of privacy also. This is proposed as future work in Section 6 as it involves training attack models on the synthetic dataset to attempt to reconstruct the original data.

CHAPTER 6

Conclusion

This research provides an analysis of the practicality of both Synthetic Data and Differentially Private Synthetic Data at solving issues of adversarial attacks and machine unlearning. We demonstrate how a capable adversary can easily construct membership inference attacks on models which have the potential to reveal sensitive information about the training dataset. Further, we explore the effectiveness of DP as a defence mechanism against these attacks and demonstrate how in practice, DP can significantly hinder a models performance both in training times and accuracy. The results of this experiment combined with the introduction of new laws requiring model creators to implement *Machine Unlearning*, which has also been discovered to worsen model privacy has resulted in the need for other solutions to these problems.

The most exciting solution is the generation of synthetic data that maintains the properties of the original dataset, yet only contains "fake" records. We analysed the performance of a CTGAN trained on a variety of datasets and determined its quality through both accuracy and privacy metrics. It was discovered that whilst it was possible to generate high quality data from an accuracy standpoint, significant privacy concerns appeared which detrimented its ability to be applicable for real world use.

To fix some of these privacy concerns, we experimented with training generative models with DP by implementing various levels of noise in the SGD process. Alongside DPSGD, we utilised privacy filters in the generation process to ensure the synthetic data contained no real records or significant outliers. This lead to promising results, where improvements in both accuracy and privacy with low noise amounts was recorded, however higher noise levels significantly degraded the quality of the dataset. As this data is infinite, training with low noise levels and adding filters to generation provided no obvious downsides provided the computational resources to do so. Although further experimentation is required with different data structures and types, applying this data for real world use seems to be the natural next step forward.

6.1 Future Work

The results of these experiments open up exciting new research areas in the field of data privacy and machine learning. As an extension to our experiment, considering data privacy is a multifaceted measurement, it would be valuable to record the maximum potential privacy leakage when the public has access to only the differentially private synthetic dataset. If the data can successfully defend against a capable adversary in the worst case, the data can also be developed for practical use as there are no real individuals to *unlearn* from the model, solving the considerable issues associated with machine unlearning.

Furthermore, experimenting with privacy mechanisms in the generation process such as the similarity and outlier filters used in our experiment would be valuable future work. A combination of both training and generation defence may be the most effective solution for use in the real world.

CHAPTER 7

Appendix

```
    schema_version: "1.0"
name: default-config
models:
  - synthetics:
      data_source:
        - gretel_37427187899142beaae6134765cf222f_Copy of Car Data - Car details
          v3.csv
      ref_data: {}
      params:
        field_delimiter: null
        epochs: 50
        batch_size: 4
        vocab_size: 0
        reset_states: false
        learning_rate: 0.002
        rnn_units: 256
        dropout_rate: 0.2
        overwrite: true
        early_stopping: true
        gen_temp: 1
        predict_batch_size: 64
        validation_split: false
        dp: true
        dp_noise_multiplier: 0.1
        dp_l2_norm_clip: 1.5
        dp_microbatches: 1
        early_stopping_min_delta: null
        max_training_time_seconds: 3000
        field_cluster_size: 20
        average_record_length_threshold: 250
        data_upsample_limit: 0
      validators:
        in_set_count: 10
        pattern_count: 10
        datetime_formats: infer
        use_numeric_iqr: false
        allow_empty_values: true
        open_close_chars: null
      generate:
        num_records: 5000
        max_invalid: 100000
      task: null
      debug:
        invalid_record_cutoff: 40
      data_checks:
        strategy: log
notifications: null
label_predictors: null
```

```
class PrivacyMetrics(tf.keras.callbacks.Callback):
  def __init__(self, epochs_per_report, model_name):
    self.epochs_per_report = epochs_per_report
    self.model_name = model_name
    self.attack_results = []

  def on_epoch_end(self, epoch, logs=None):
    epoch = epoch+1

    if epoch % self.epochs_per_report != 0:
      return

    print(f'\nRunning privacy report for epoch: {epoch}\n')

    logits_train = self.model.predict(x_train, batch_size=batch_size)
    logits_test = self.model.predict(x_test, batch_size=batch_size)

    prob_train = special.softmax(logits_train, axis=1)
    prob_test = special.softmax(logits_test, axis=1)

    # Add metadata to generate a privacy report.
    privacy_report_metadata = PrivacyReportMetadata(
        # Show the validation accuracy on the plot
        # It's what you send to train_accuracy that gets plotted.
        accuracy_train=logs['val_accuracy'],
        accuracy_test=logs['val_accuracy'],
        epoch_num=epoch,
        model_variant_label=self.model_name)

    attack_results = mia.run_attacks(
        AttackInputData(
            labels_train=y_train_indices[:, 0],
            labels_test=y_test_indices[:, 0],
            probs_train=prob_train,
            probs_test=prob_test),
        SlicingSpec(entire_dataset=True, by_class=True),
        attack_types=(AttackType.THRESHOLD_ATTACK,
                      AttackType.LOGISTIC_REGRESSION),
        privacy_report_metadata=privacy_report_metadata)

    self.attack_results.append(attack_results)
```

FIGURE 7.1: Code Snippet of the PrivacyMetrics class adapted from TensorFlow tutorial

```
import tensorflow_privacy.privacy.membership_inference_attack.plotting as plotting
all_reports = []
callback = PrivacyMetrics(epochs_per_report, "3␣Layers␣0.05␣noise")
history = model_3layers.fit(
      x_train,
      y_train,
      batch_size=batch_size,
      epochs=total_epochs,
      validation_data=(x_test, y_test),
      callbacks=[callback],
      shuffle=True)

all_reports.extend(callback.attack_results)
max_auc = all_reports.get_result_with_max_auc().get_auc()
max_adv = all_reports.get_result_with_max_attacker_advantage().get_attacker_advantage()
plotting.plot_roc_curve(best_auc.get_result_with_max_auc().roc_curve)
```

FIGURE 7.2: Code Snippet of the training process followed by collecting auc and adv for a single model and plotting ROC curve

```
class TimeCallback(tf.keras.callbacks.Callback):
    def __init__(self):
        self.times = []
        # use this value as reference to calculate cummulative time taken
        self.timetaken = time.clock()
    def on_epoch_end(self,epoch,logs = {}):
        self.times.append((epoch,time.clock() - self.timetaken))
```

FIGURE 7.3: Custom TensorFlow callback to record time

```
    import numpy as np
    import pandas as pd
    from collections import Counter

    f_obs, f_exp = [], []
    real, synthetic = Counter(real_data), Counter(synthetic_data)
    for x in real_data:
        f_obs.append(synthetic[x] / sum(synthetic.values()))
        f_exp.append(real[x] / sum(real.values()))
```

FIGURE 7.4: Helper function to create percentage frequencies for values in each attribute

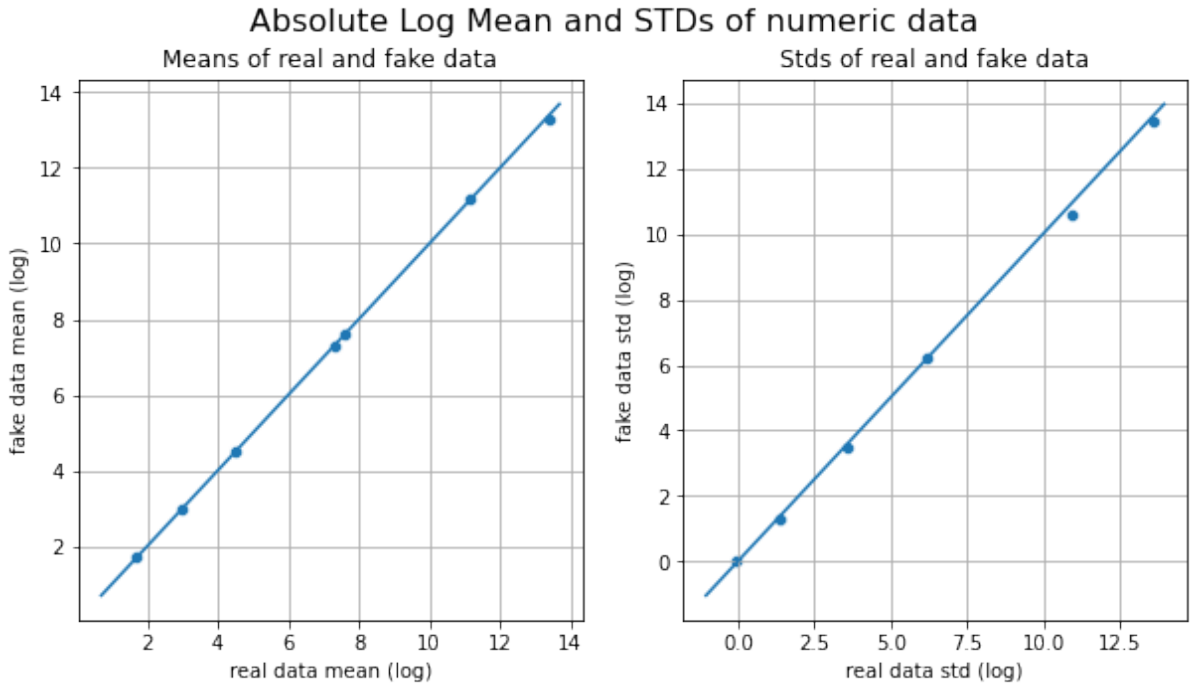


FIGURE 7.5: Absolute Log Mean and STDs of Vehicle numeric data

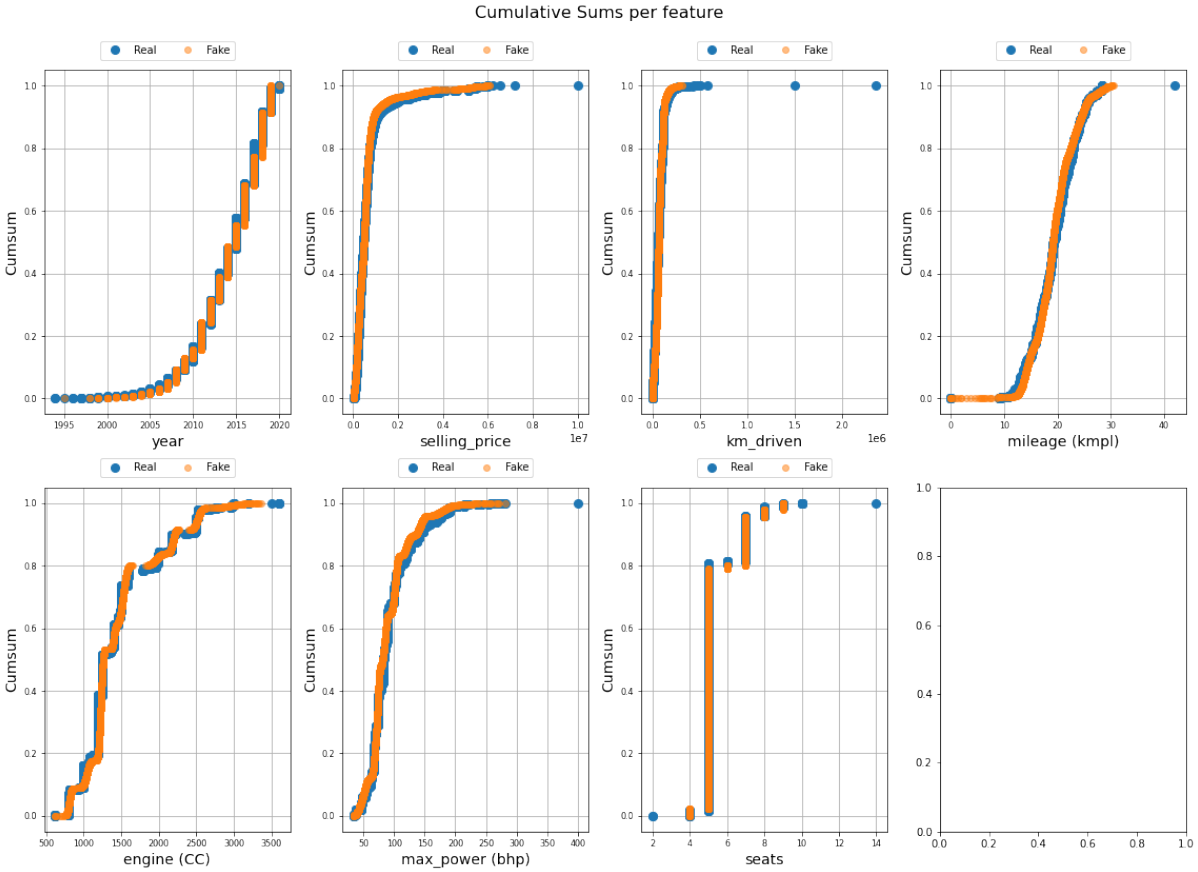


FIGURE 7.6: Vehicle Cumulative Sums per feature

FIGURE 7.7: Vehicle Confusion Matrix

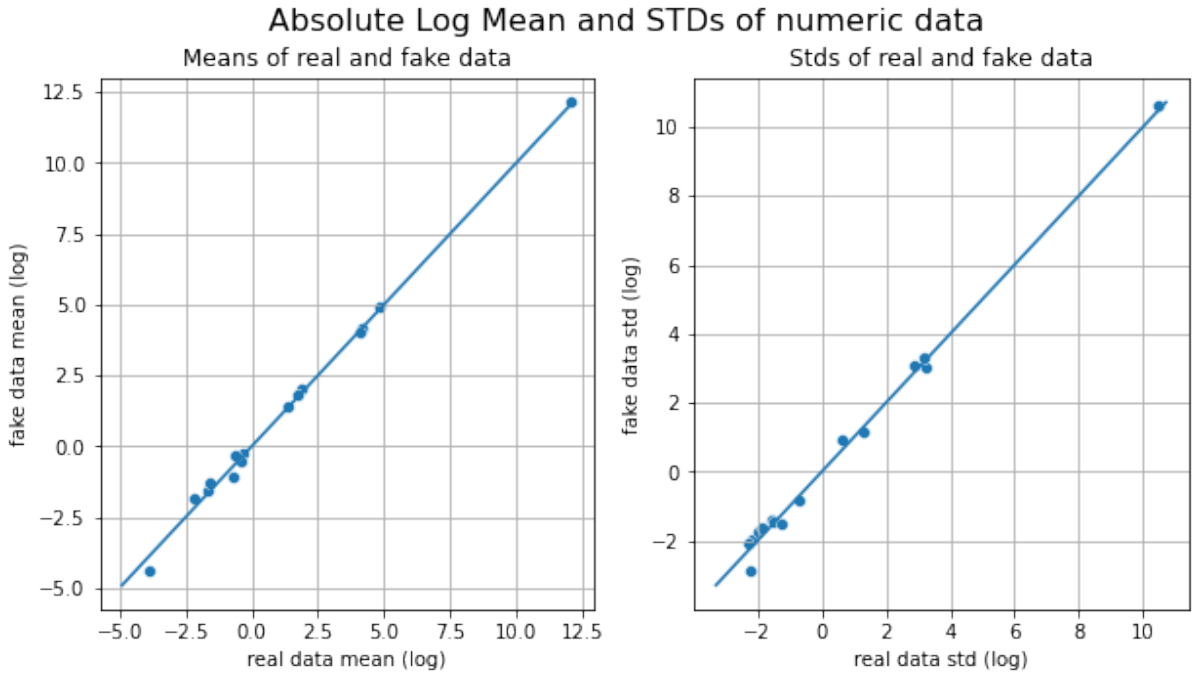


FIGURE 7.8: Absolute Log Mean and STDs of TikTok numeric data

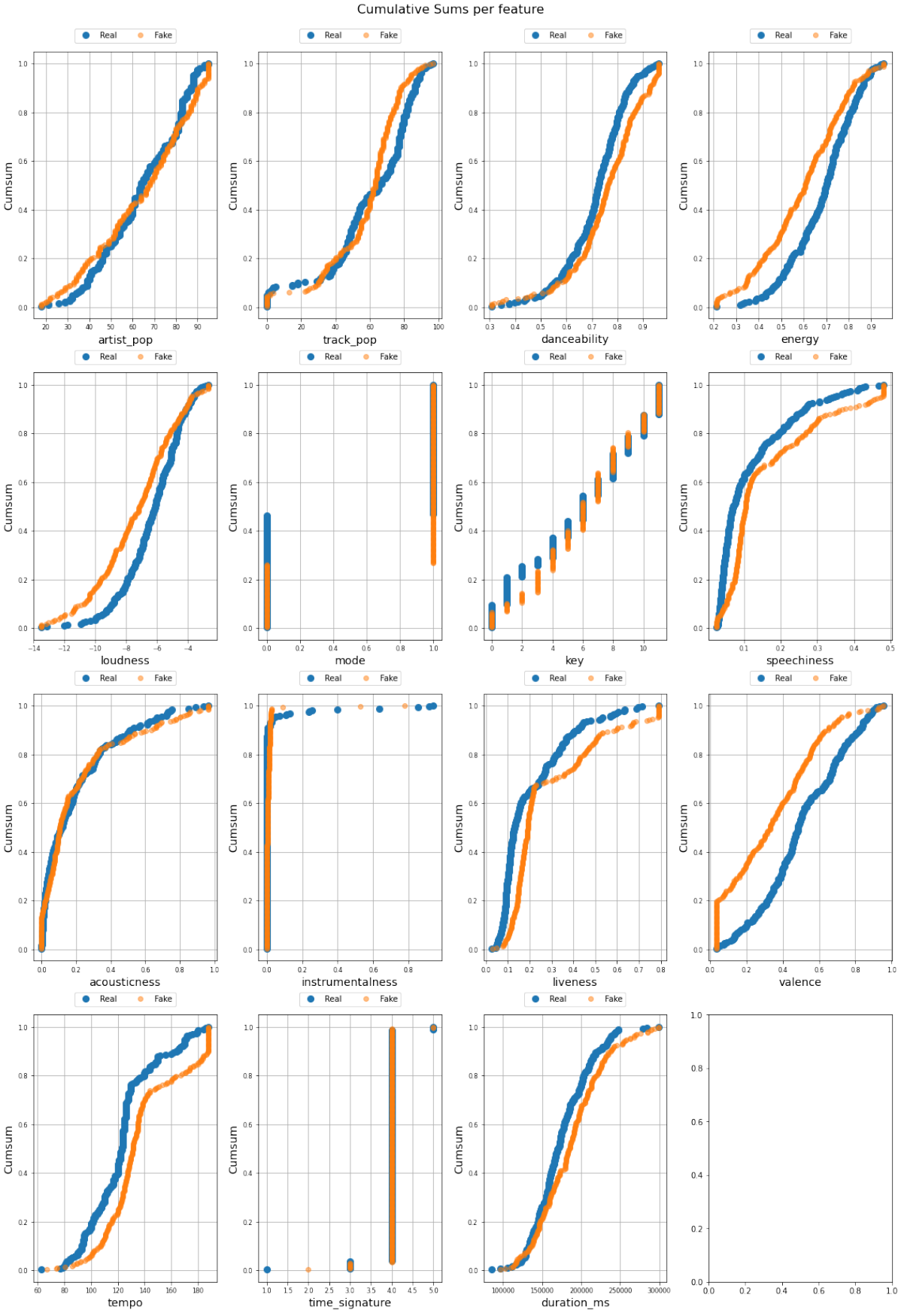


FIGURE 7.9: TikTok Cumulative Sums per feature

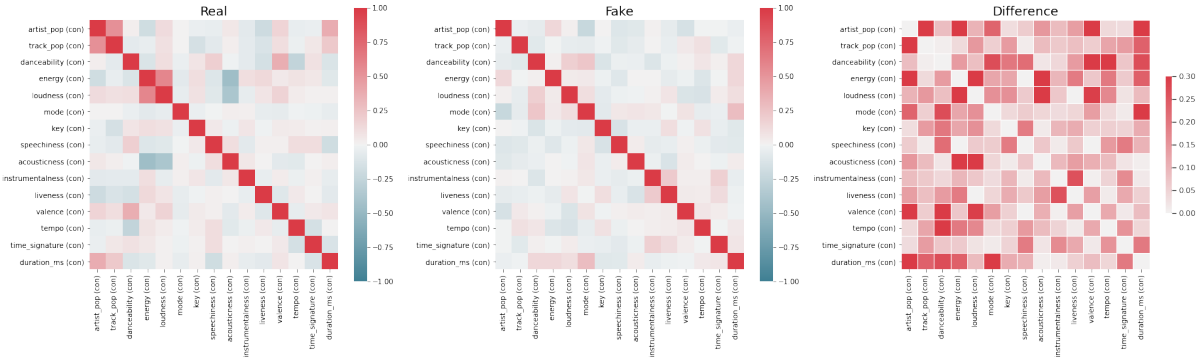


FIGURE 7.10: TikTok Confusion Matrix

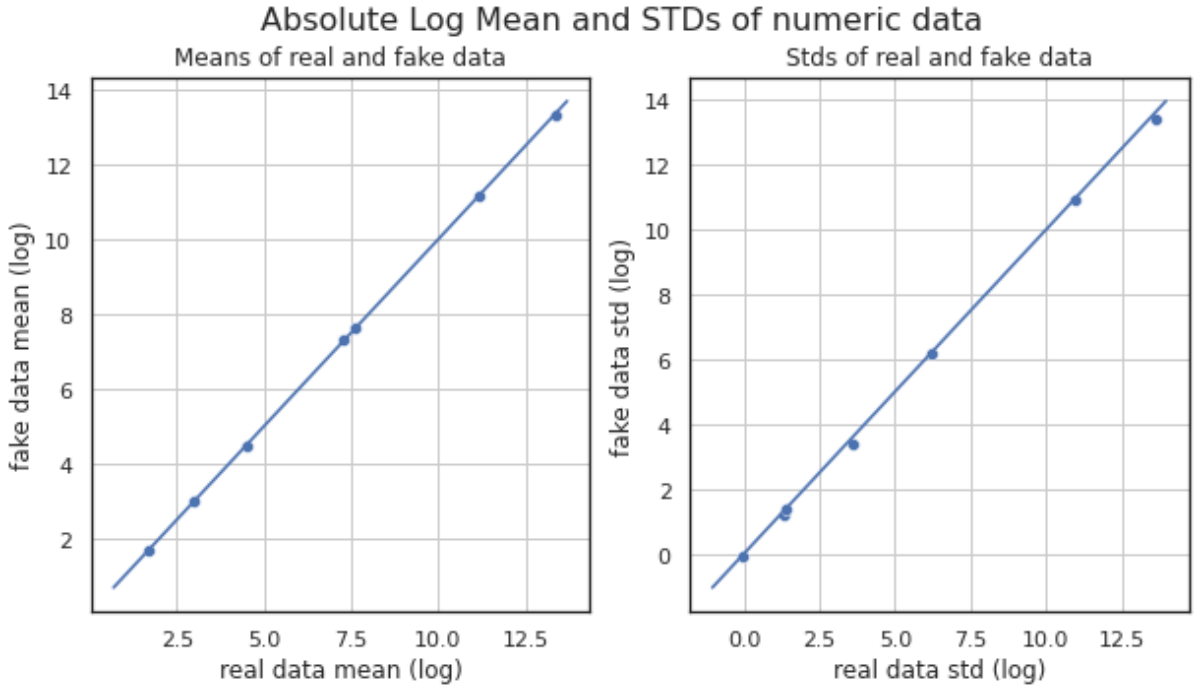


FIGURE 7.11: Absolute Log Mean and STDs of Vehicle numeric data, noise multiplier = 0.001

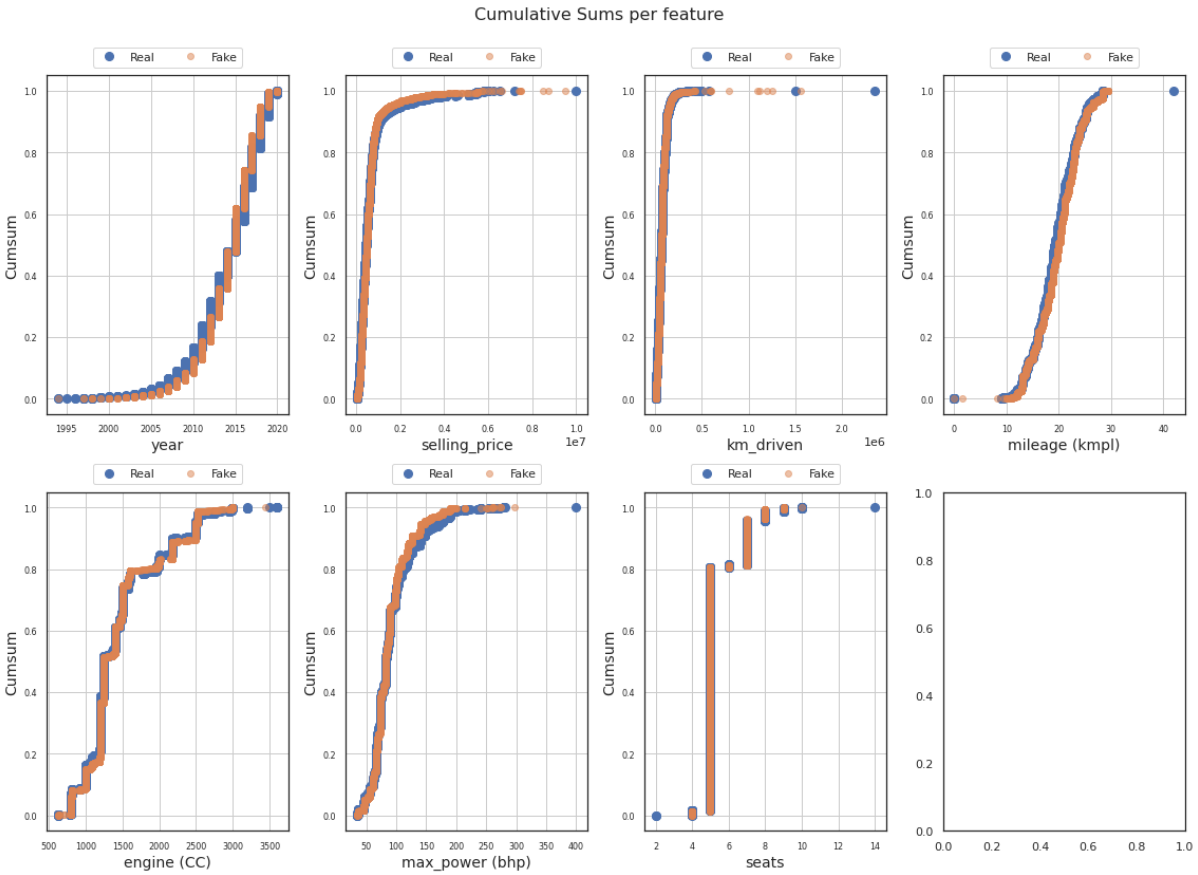


FIGURE 7.12: Vehicle Cumulative Sums per feature, noise multiplier = 0.001

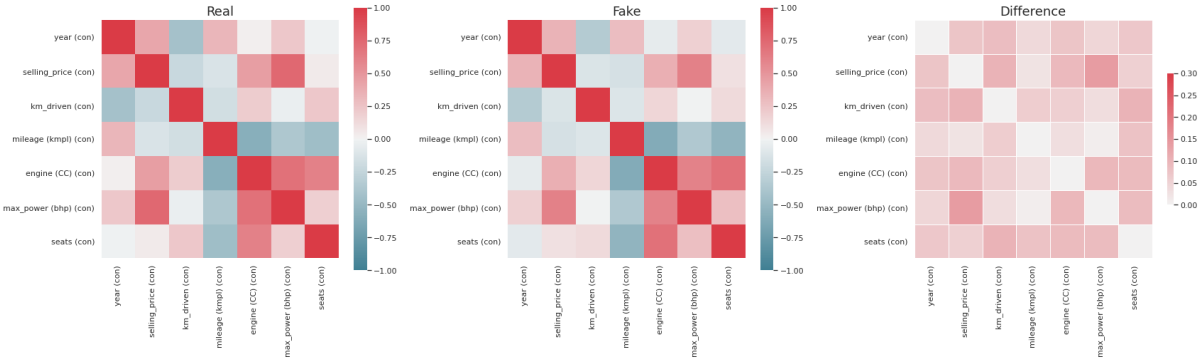


FIGURE 7.13: Vehicle Confusion Matrix, noise multiplier = 0.001

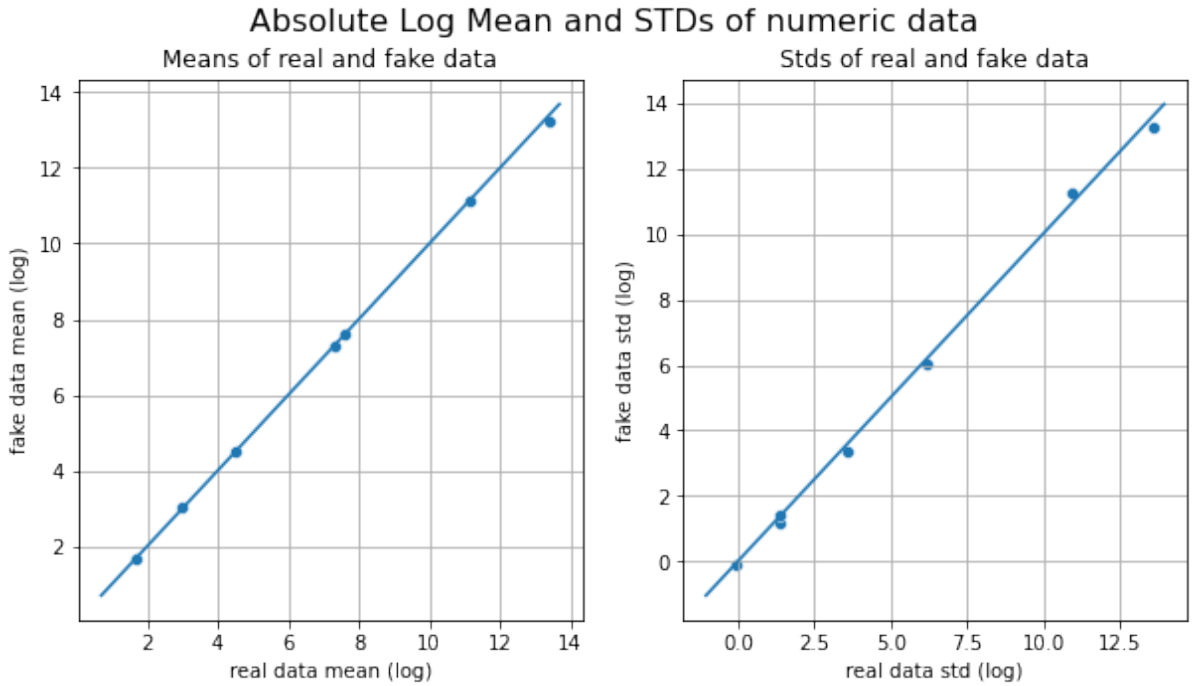


FIGURE 7.14: Absolute Log Mean and STDs of Vehicle numeric data, noise multiplier = 0.01

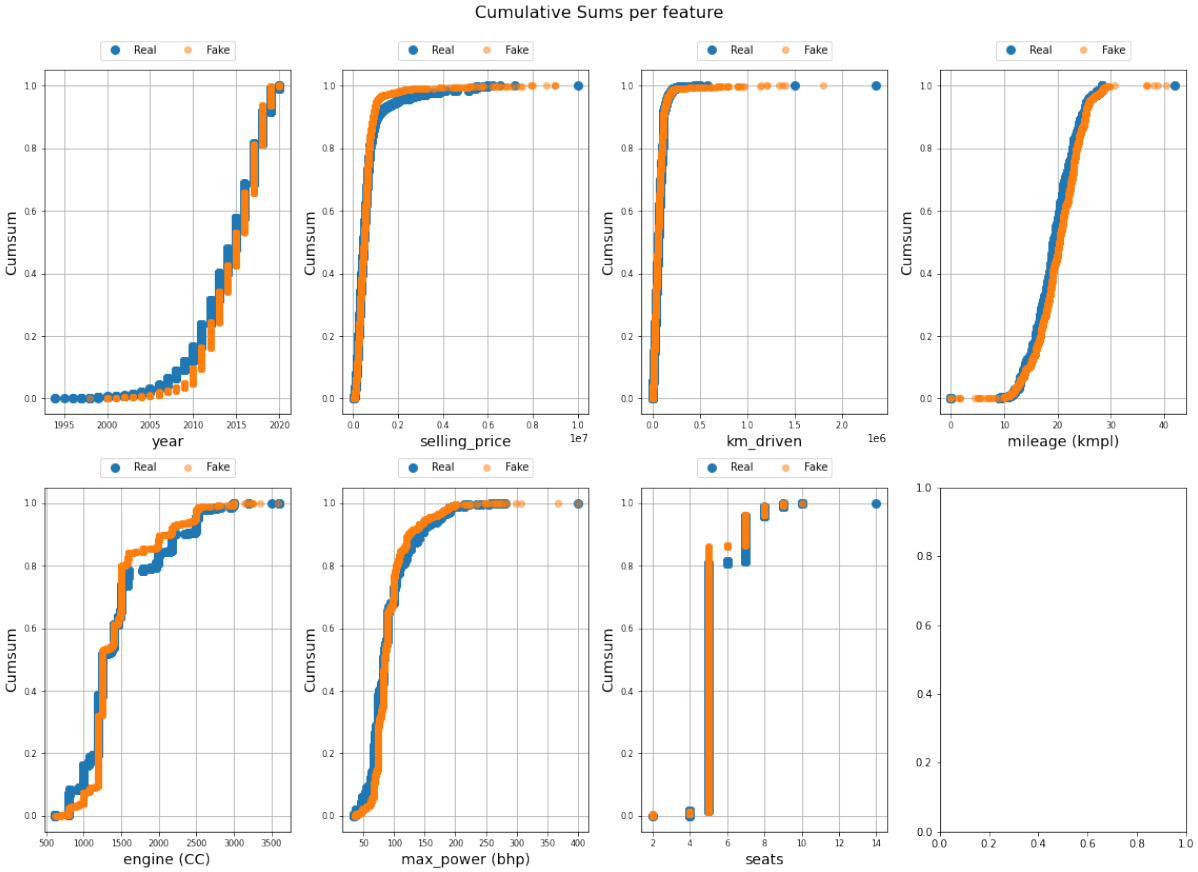


FIGURE 7.15: Vehicle Cumulative Sums per feature, noise multiplier = 0.01

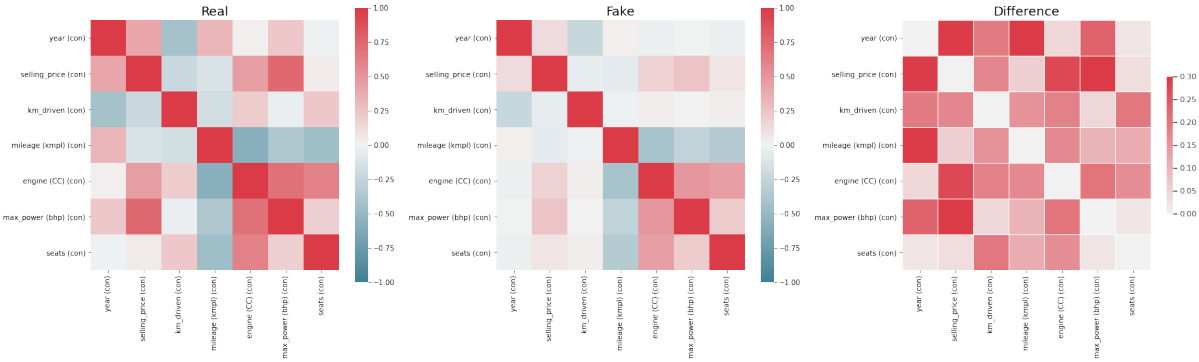


FIGURE 7.16: Vehicle Confusion Matrix, noise multiplier = 0.01

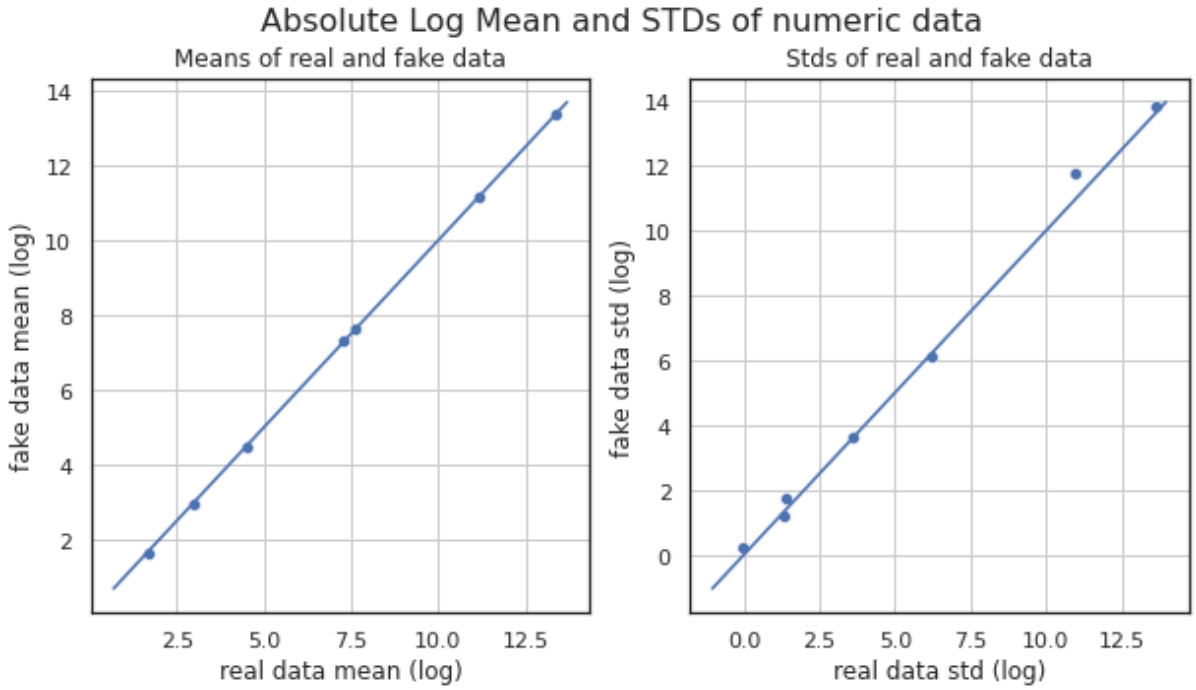


FIGURE 7.17: Absolute Log Mean and STDs of Vehicle numeric data, noise multiplier = 0.1

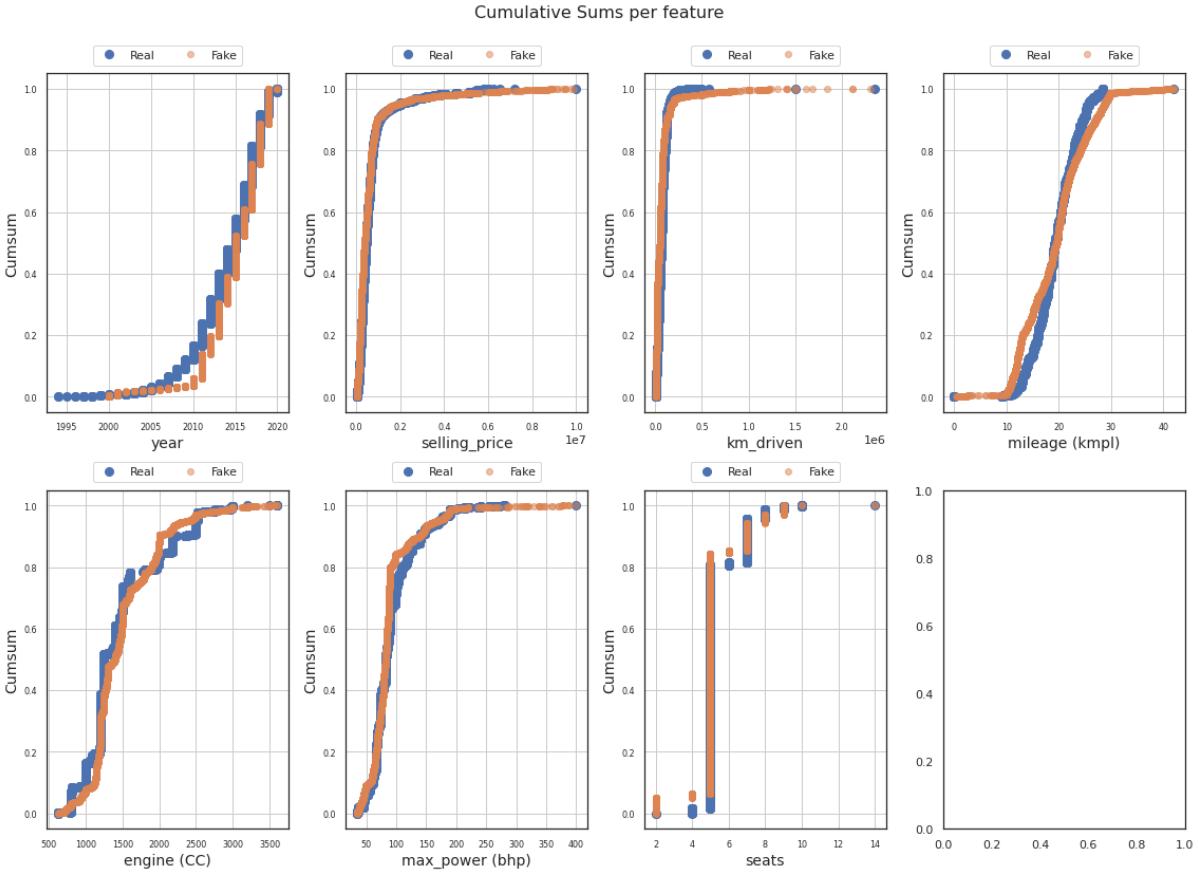


FIGURE 7.18: Vehicle Cumulative Sums per feature, noise multiplier = 0.1

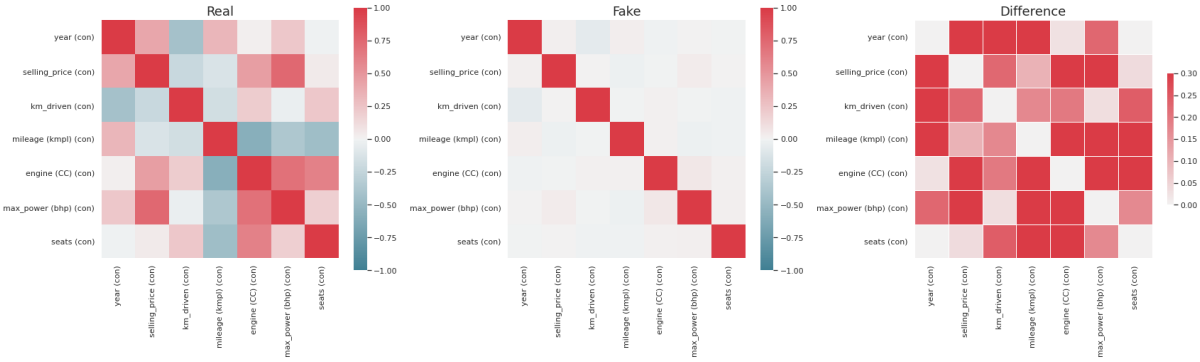
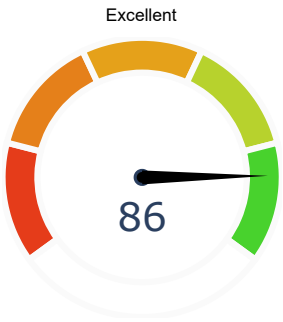


FIGURE 7.19: Vehicle Confusion Matrix, noise multiplier = 0.1

Gretel Synthetic Report

Generated 09/27/2022, 05:13



Synthetic Data Quality Score

Privacy Protection Level

Data Summary Statistics

Field Correlation Stability

Deep Structure Stability

Field Distribution Stability

	Training Data	Synthetic Data
Row Count	5000	5000
Column Count	8	8
Training Lines Duplicated	--	0

What do these values mean?

Privacy Protection Summary

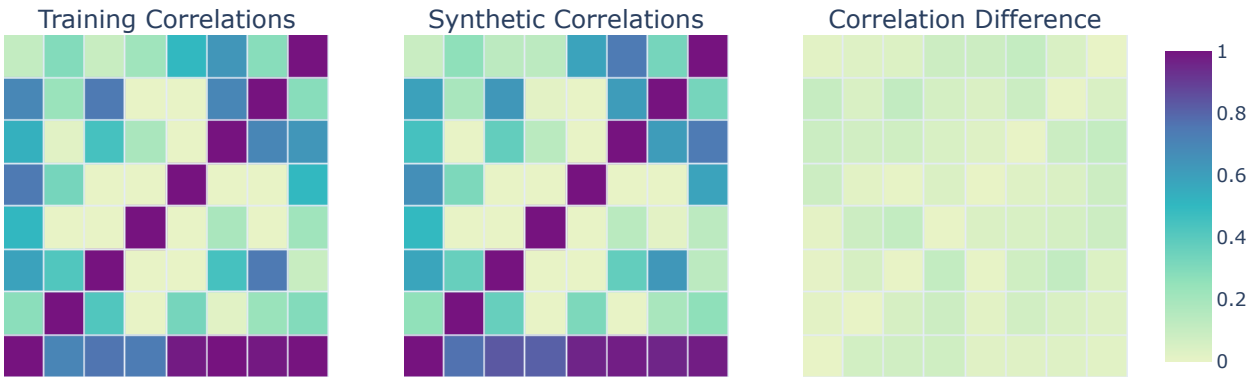
Default Privacy Protections			Advanced Protections
Outlier Filter Medium	Similarity Filter Medium	Overfitting Prevention Enabled	Differential Privacy Enabled

Training field overview

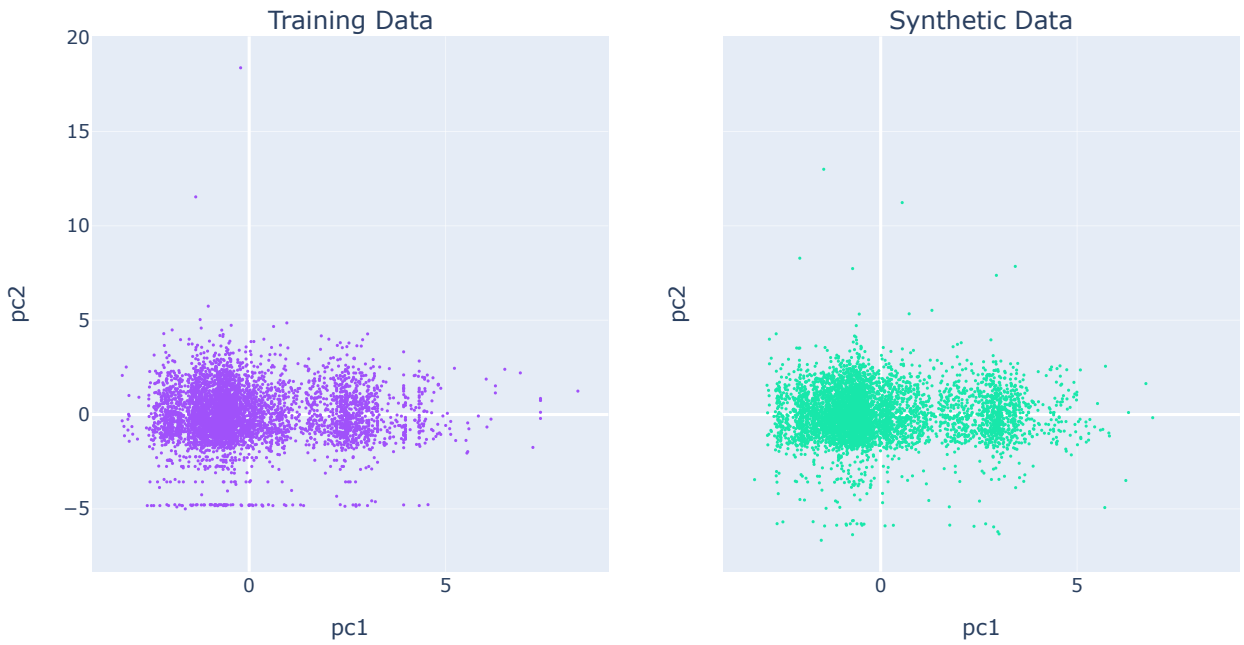
Field	Unique	Missing	Ave. Length	Type	Distribution Stability
name	1615	0	25.21	Other	N/A
selling_price	545	0	6.08	Numeric	Excellent
mileage (kmpl)	337	0	4.41	Numeric	Excellent
max_power (bhp)	279	0	4.51	Numeric	Excellent
engine (CC)	112	0	3.84	Numeric	Excellent
year	26	0	4.00	Numeric	Excellent

km_driven	669	0	5.19	Numeric	Excellent
seats	9	0	1.00	Categorical	Excellent

Training and Synthetic Data Correlation



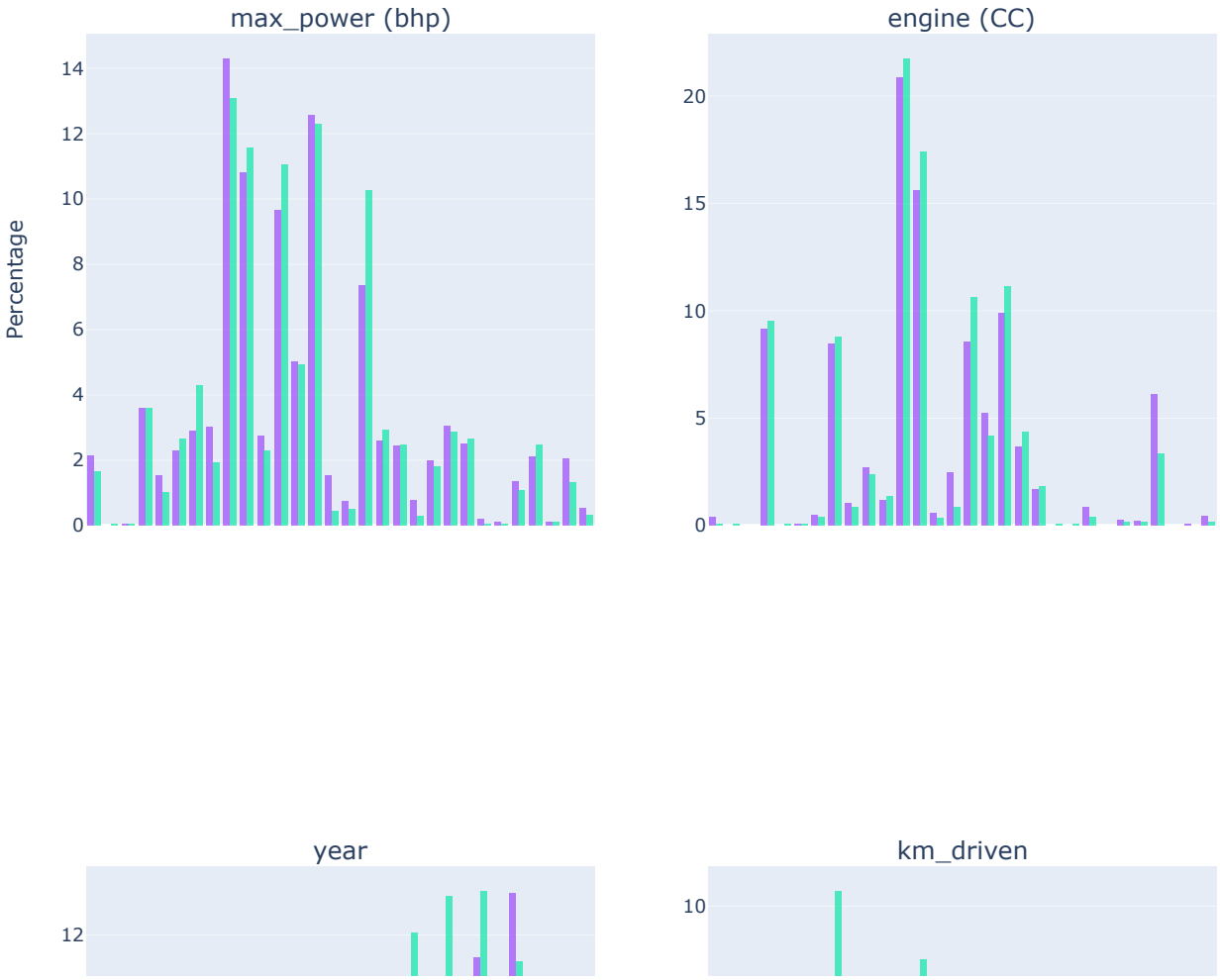
Principal Component Analysis



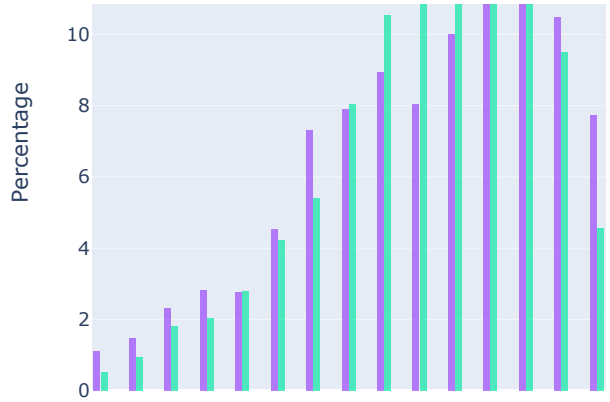
Field Distribution Comparisons

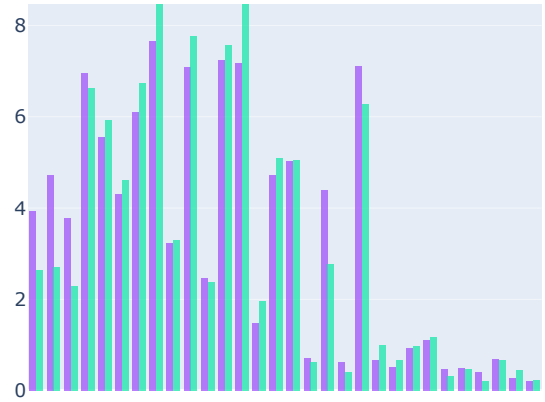
() ()

() ()

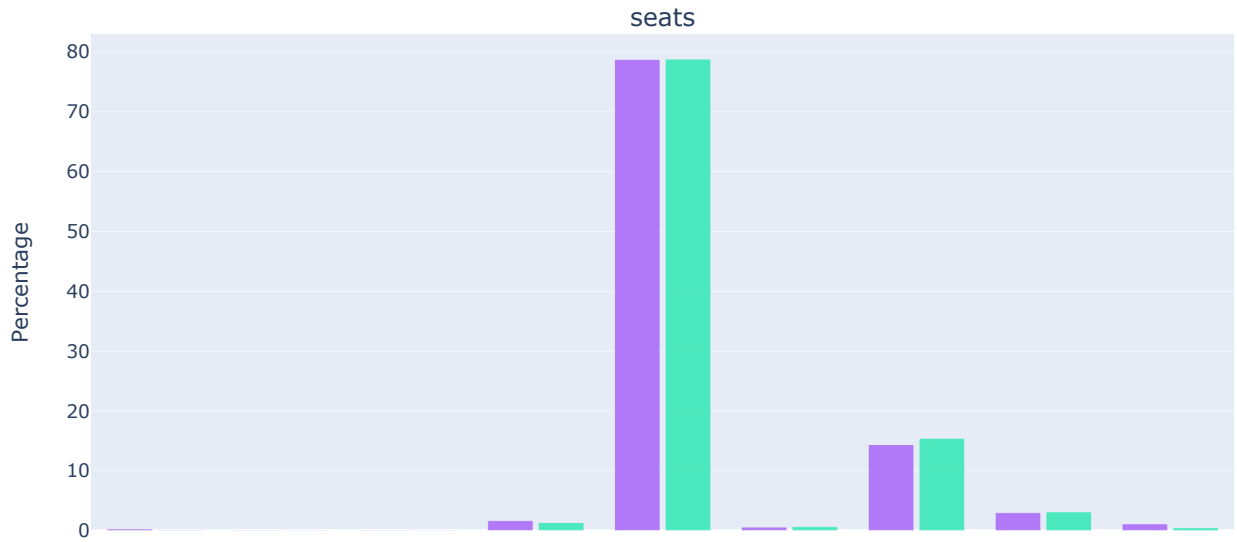


() ()





()



Bibliography

[ACG+16] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, oct 2016.

[BCCC+19] Lucas Bourtoule, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning, 2019.

[BMR+20] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[CAD+21] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. A survey on adversarial attacks and defences. *CAAI Transactions on Intelligence Technology*, 6(1):25–45, 2021.

[CCN+21] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. Membership inference attacks from first principles, 2021.

[CSU+19] Albert Cheu, Adam Smith, Jonathan Ullman, David Zeber, and Maxim Zhilyaev. Distributed differential privacy via shuffling. In *Advances in Cryptology – EUROCRYPT 2019*, pages 375–403. Springer International Publishing, 2019.

[CTMK22] Vikram S Chundawat, Ayush K Tarun, Murari Mandal, and Mohan Kankanhalli. Zero-shot machine unlearning, 2022.

[CY15] Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE Symposium on Security and Privacy*, pages 463–480, 2015.

[CZW+21] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. When machine unlearning jeopardizes privacy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. ACM, nov 2021.

[Den12] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[DFH+15] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. Generalization in adaptive data analysis and holdout reuse. *CoRR*,

abs/1506.02629, 2015.

[DKM19] Cynthia Dwork, Nitin Kohli, and Deirdre Mulligan. Differential privacy in practice: Expose your epsilons! *Journal of Privacy and Confidentiality*, 9(2), Oct. 2019.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography*, pages 265–284, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[Dwo08] Cynthia Dwork. Differential privacy: A survey of results. In Manindra Agrawal, Dingzhu Du, Zhenhua Duan, and Angsheng Li, editors, *Theory and Applications of Models of Computation*, pages 1–19, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[GAA+17] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans, 2017.

[GGMV22] Ji Gao, Sanjam Garg, Mohammad Mahmoody, and Prashant Nalini Vasudevan. Deletion inference, reconstruction, and compliance in machine (un)learning, 2022.

[GPAM+14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[GSS15] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[JWK+21] Bargav Jayaraman, Lingxiao Wang, Katherine Knipmeyer, Quanquan Gu, and David Evans. Revisiting membership inference under realistic assumptions. *Proceedings on Privacy Enhancing Technologies*, 2021:348–368, 04 2021.

[KLN+08] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? 2008.

[Kri09] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[KW13] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the Institute of Radio Engineers*, 86(11):2278–2323, 1998.

[Mas51] Frank J. Massey. The kolmogorov-smirnov test for goodness of fit. *Journal of the American Statistical Association*, 46(253):68–78, 1951.

[MDK+22] Shagufta Mehnaz, Sayanton V. Dibbo, Ehsanul Kabir, Ninghui Li, and Elisa Bertino. Are your sensitive attributes private? novel model inversion attribute inference attacks on classification models, 2022.

[MRA21] Neil G. Marchant, Benjamin I. P. Rubinstein, and Scott Alfeld. Hard to forget: Poisoning attacks on certified machine unlearning, 2021.

[MXLM20] Yuxin Ma, Tiankai Xie, Jundong Li, and Ross Maciejewski. Explaining vulnerabilities to adversarial machine learning through visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):1075–1085, jan 2020.

[ND21] Joseph Near and David Darais. Differentially private synthetic data, May 2021.

[Nik19] Sergey I. Nikolenko. Synthetic data for deep learning, 2019.

[Pea00] Karl Pearson. X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 50(302):157–175, July 1900.

[PWV16] N. Patki, R. Wedge, and K. Veeramachaneni. The synthetic data vault. In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 399–410, Oct 2016.

[RG20] Maria Rigaki and Sebastian Garcia. A survey of privacy attacks in machine learning, 2020.

[RZQL20] Kui Ren, Tianhang Zheng, Zhan Qin, and Xue Liu. Adversarial attacks and defenses in deep learning. *Engineering*, 6(3):346–360, 2020.

[SSSS16] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models, 2016.

[TCMK21] Ayush K Tarun, Vikram S Chundawat, Murari Mandal, and Mohan Kankanhalli. Fast yet effective machine unlearning, 2021.

[TKP+17] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses, 2017.

[TL21] Mingxing Tan and Quoc V. Le. Efficientnetv2: Smaller models and faster training. 2021.

[TMH+21] Yuchao Tao, Ryan McKenna, Michael Hay, Ashwin Machanavajjhala, and Gerome Miklau. Benchmarking differentially private synthetic data generation algorithms, 2021.

[TSJ+22] Florian Tramèr, Reza Shokri, Ayrton San Joaquin, Hoang Le, Matthew Jagielski, Sanghyun Hong, and Nicholas Carlini. Truth serum: Poisoning machine learning models to reveal their secrets, 2022.

[XSCIV19] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan, 2019.

[YGFJ17] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting, 2017.

[ZAC+21] Benjamin Zi Hao Zhao, Aviral Agrawal, Catisha Coburn, Hassan Jameel Asghar, Raghav Bhaskar, Mohamed Ali Kaafar, Darren Webb, and Peter Dickinson. On the (in)feasibility of attribute inference attacks on machine learning models, 2021.