Chapter 17. Privacy and Legal Requirements

Contributed by Catherine Nelson

Data privacy is becoming an important part of ML projects. There's an increasing push toward ethical AI and a growing number of legal requirements around data privacy. Many of the predictions made by ML models are based on personal data collected from users, so it's important to have an awareness of strategies to increase privacy in ML pipelines, as well as some knowledge of the laws and regulations in this area.

Before you even start building your ML pipelines, it's essential to be transparent with your users about what data you are collecting. You should ensure that you have consent from your users to use their data. And you should also minimize data collection to what's necessary to train your models. Once you have these fundamental principles in place, you can look at the privacy-preserving ML options we describe in this chapter to provide even greater privacy for your users.

At the time of this writing, there is always a cost to privacy: increasing privacy for our users comes with a cost in model accuracy, computation time, or both. At one extreme, collecting no data keeps an interaction completely private but is completely useless for ML. At the other extreme,

knowing all the details about a person might endanger that person's privacy, but it allows us to make very accurate ML models. We're starting to see the development of privacy-preserving ML, in which privacy can be increased without such a large trade-off in model accuracy.

In this chapter, we'll discuss some of the reasons why this is an important topic. We'll explain some of the legal considerations that may be important to your work, and we'll explain the difference between pseudonymization and anonymization. We'll also give you an overview of some of the methods you can use to increase privacy for your users when building ML models: these include differential privacy, federated learning, and encrypted ML. This chapter also includes a code example of differentially private ML using the TensorFlow Privacy (TFP) library.

Why Is Data Privacy Important?

Data privacy in ML pipelines may seem like an added complication, but it's an extremely important topic. Training data, prediction requests, or both can contain very sensitive information about people. For prediction requests, those people are your users. Privacy of sensitive data should be protected. Data privacy requires you to respect legal and regulatory requirements, as well as social norms and typical individual expectations. Consider putting safeguards in place to ensure each individual's privacy, including ML models that may remember or reveal aspects of the data they've been

exposed to. You may also need to take steps to ensure that users have adequate transparency and control of their data.

Before we discuss the legal requirements around privacy and some methods for keeping data private, we'll go through what kind of data needs to be kept private and discuss potential consequences if it is exposed.

What Data Needs to Be Kept Private?

You need to consider data privacy when you collect data from, for, or about people. There are two main ways of classifying this data: either as personal identifiable information (PII) or as sensitive data.

A major concern for PII is that it can be used to directly identify a single person. It includes data about any natural or legal person, living or dead, including their dependents, ascendants, and descendants, who might be identifiable through either direct or indirect relationships. PII includes features such as family names, patronyms, first names, maiden names, aliases, addresses, phone numbers, bank account details, credit cards, and tax ID numbers.

PII can appear in free text, such as feedback comments or customer service data, not just when users are directly asked for this data. Images of people may also be considered PII in some circumstances. There are often legal standards around this, which we will discuss in the next section. If your

company has a privacy team, it’s best to consult them before embarking on a project using this type of data.

Sensitive data also requires special care. This is often defined as data that could cause harm to someone if it were released, such as health data or proprietary company data (e.g., financial data). Ensure that this type of data is not leaked in the predictions of an ML model.

As a general rule, it’s best to err on the side of privacy and consider any personal information that you have in your data as sensitive. You should restrict access to it and keep it safe. Above all, you should think of it as the property of the person whose information it is, and honor their wishes.

Harms

Security and privacy are closely linked for some problems, or *harms*, in ML.

Informational harms are caused when information is allowed to leak from the model. There are at least three different types of informational harms:

Membership inference

An attacker can determine whether or not an individual’s data was included in a model’s training data.

Model inversion

An attacker is able to re-create the training data from the trained model.

Model extraction

An attacker steals the model or is able to re-create it exactly.

Behavioral harms are caused when an attacker is able to change the behavior of the model itself. They include the following:

Poisoning attacks

The attacker is able to insert malicious data into the training set.

Evasion attacks

The attacker makes small changes to prediction requests to cause the model to make bad predictions

In the next section, we'll give you an introduction to the legal requirements around data privacy as they apply to ML.

Only Collect What You Need

Privacy starts with the data you collect, for both training and inference. There is a tendency when collecting data to collect as much data as possible, with as many features as possible. Before you fall into that trap, consider the privacy implications. Do you really need the user's name and

email address? Could you just assign them an ID number instead, and keep them anonymous from the beginning? If you don’t have private information about a user in your dataset from the moment it’s created, it’s much easier to maintain that privacy going forward.

When possible, ask the user for their consent before collecting their data. For example, if you’re collecting the queries to a chatbot in order to create a new dataset, ask the user before collecting them, and save the user’s response. Above all, never collect a user’s data when they have refused permission.

GenAI Data Scraped from the Web and Other Sources

GenAI datasets tend to be very large, and they are often collected by scraping web pages or from other large sources. This raises privacy issues when that data contains personal information such as email addresses. It also raises questions about the ownership and fair use of the data. At the time of this writing, this is a very controversial subject, and the legal aspects of it are not settled. This is another area where it’s better to limit what you collect from the start, rather than collecting private information and managing the use and protection of it later. Avoiding collecting data that is not licensed for public use but is accessible from the internet is more difficult. Since the volume of data being collected is very large and the

sources of it are often complex and unstructured, it can be difficult to write scripts that can detect when data is not licensed for public use. We encourage you to research the current tools available for mitigating these issues before starting a new project.

Legal Requirements

There is also a legal side to practicing Responsible AI and protecting the privacy of your users. There are already legal requirements around data privacy in many countries and regions, and this trend is growing. Exposure to civil liability is another concern. So it’s important that you have an awareness of the laws that may apply to you when you’re building ML pipelines.

In this section, we’ll give an overview of two of the most impactful data privacy laws introduced in the past few years: the European Union’s General Data Protection Regulations (GDPR) and the California Consumer Privacy Act (CCPA). The GDPR in particular makes rigorous demands around data protection, and we’ll explain one of those in more detail. We also recommend that you consult your company’s legal team to find out what data privacy laws apply to your work.

The GDPR and the CCPA

The EU enacted the GDPR in 2018, and it became a model for many national laws outside the EU, including Chile, Japan, Brazil, South Korea, Argentina, and Kenya. The GDPR regulates data protection and privacy in both the EU and the European Economic Area (EEA). The GDPR gives individuals control over their personal data, and it requires that companies protect the data of their employees and their users. When data processing is based on consent, the data subject (usually an individual person) has the right to revoke their consent at any time. The GDPR sets out various rights of an individual, including the right to transparency around how their data is used, the right to access their data, the right to object to a decision that is made using their data, and the right to be forgotten, which we'll explain in more detail next.

The CCPA was modeled after the GDPR. It has similar goals, including enhancing the privacy rights and consumer protections for residents of California. It states that users have the right to know what personal data is being collected about them, including whether the personal data is sold or disclosed in some other way, who supplied their data, and who received their data. Users can access their personal data held by a company, block the sale of their data, and request a company to delete their data.

The GDPR's Right to Be Forgotten

One of the rights outlined in the GDPR is the right to be forgotten. This has implications for ML, so we'll go into it in more detail. As stated in Recitals 65 and 66 and in Article 17 of the GDPR:

> *The data subject shall have the right to obtain from the controller the erasure of personal data concerning him or her without undue delay and the controller shall have the obligation to erase personal data without undue delay.*

When the GDPR refers to a "data subject" it means a person, and when it refers to a "controller" it means a person or organization that has control over a dataset containing PII. A person can request the deletion of their data if they want to withdraw their consent to the use of the data. (However, in some cases an organization's right to process someone's data might override their right to be forgotten; for example, if the use of their data is in the public interest.)

If your company receives a valid request to have personal information deleted, you need to identify all the information related to the content requested to be removed. You also need to identify and delete all the metadata associated with that person. If you've run any analysis, the derived data and logs also must be deleted. The goal here is, as much as possible, to make it as if you never had their data.

There are two ways to delete data that will satisfy the requirements of the GDPR. First, you can anonymize the data, which will make it not personally identifiable under the terms of the GDPR. We'll explain this in more detail in the next section. Second, you can do a hard delete of the data, meaning actually delete that data, including any rows in your database that might contain it.

If you have an ML model that depends on some data that is deleted, it may be necessary to retrain that model. In this case, having good metadata and records of how that model was trained in the first place will be extremely useful.

In a database or any other similar relational datastore, deleting records can cause problems. User data is often referenced in multiple tables, so deleting those records breaks the connections. This can be difficult to repair, especially in large, complex databases. On the other hand, anonymization keeps the records and only anonymizes the fields containing PII, while still satisfying the requirements of the GDPR.

If deleting data isn't needed to conform with GDPR, but you still want to increase privacy for your users, you can look into the options we describe in the next four sections: pseudonymization and anonymization, differential privacy, federated learning, and encrypted ML.

Pseudonymization and Anonymization

Pseudonymization and anonymization are two of the most well-established ways of protecting privacy. The GDPR includes the legal definitions of many of the terms that it uses, including anonymization and pseudonymization. As shown in Figure 17-1, there's a spectrum of increasing privacy from pseudonymization to anonymization.

Figure 17-1. The anonymity spectrum

Pseudonymization means replacing PII with placeholder data in a way that's reversible. It's still possible for an attacker to identify individuals in pseudonymized data if they have additional data. Pseudonymization can be implemented with data masking, encryption, or tokenization. It relies on careful control of access to the additional identifying information. Pseudonymizing data may help you meet the data protection obligations of the GDPR, but it's best to consult your company's legal team to confirm this.

Data can be *de-identified* by deleting PII rather than replacing it with placeholder data. This provides a higher level of privacy than pseudonymization, but it may still be possible to re-identify individuals in your dataset with additional information.

The difference between de-identified data and pseudonymized data is not well-defined, and many discussions will group them together as one thing. Pseudonymized and de-identified data are at the lower end of the spectrum. They are indeed a way of preserving certain aspects of data privacy, but not to the level of truly anonymized data.

Anonymization removes PII from datasets so that the people who the data describes remain anonymous. In the GDPR, Recital 26 defines acceptable data anonymization as being:

- Irreversible
- Done in such a way that it is impossible to identify the person
- Impossible to derive insights or discrete information, even by the party responsible for anonymization

Once data has been acceptably anonymized, the GDPR no longer applies to that data.

If your ML model depends on PII to function, anonymization will severely reduce its performance. An example of this would be a recommendation system that uses an individual's gender as one of its features. However, there are situations where PII can be present in a model, but it's not essential to its function. An example is a large language model (LLM). LLMs are trained on large quantities of text data that may contain PII, but

they seek to generalize rather than use the PII for training data. Pseudonymization should not affect the performance of ML models.

Differential Privacy

Differential privacy (DP) is not mentioned in the GDPR, but it has a lot of potential for increasing privacy in ML pipelines while retaining good accuracy. It gives mathematical guarantees of privacy while still preserving the utility of data. It is a formalization of the idea that a query or a transformation of a dataset should not reveal whether a person is in that dataset. It gives a mathematical measure of the privacy loss that a person experiences by being included in a dataset and minimizes this privacy loss by adding noise.

To put it another way, a transformation of a dataset that respects privacy should not change if one person is removed from that dataset. In the case of ML models, if a model has been trained with privacy in mind, the predictions that a model makes should not change if one person is removed from the training set. DP is achieved by the addition of some form of noise or randomness to the transformation. A real-world example of the use of DP is documented in the Google Research blog post “Advances in Private Training for Production On-Device Language Models”.

To give a more concrete example, one of the simplest ways of achieving differential privacy is the concept of randomized response, as shown in Figure 17-2. This is useful in surveys that ask sensitive questions, such as "Have you ever been convicted of a crime?" To answer this question, the person being asked flips a coin. If it comes up heads, they answer truthfully. If it comes up tails, they flip again and answer "Yes" if the coin comes up heads and "No" if the coin comes up tails. Because we know the probabilities for a coin flip, if we ask a lot of people this question, we can calculate with reasonable accuracy the proportion of people who have been convicted of a crime. The accuracy of the calculation increases when larger numbers of people participate in the survey.

The important point here is the presence of randomness in the process. The survey participants can say that their answer was a random answer rather than a truthful answer, and this gives them privacy. Randomized transformations are the key to DP.

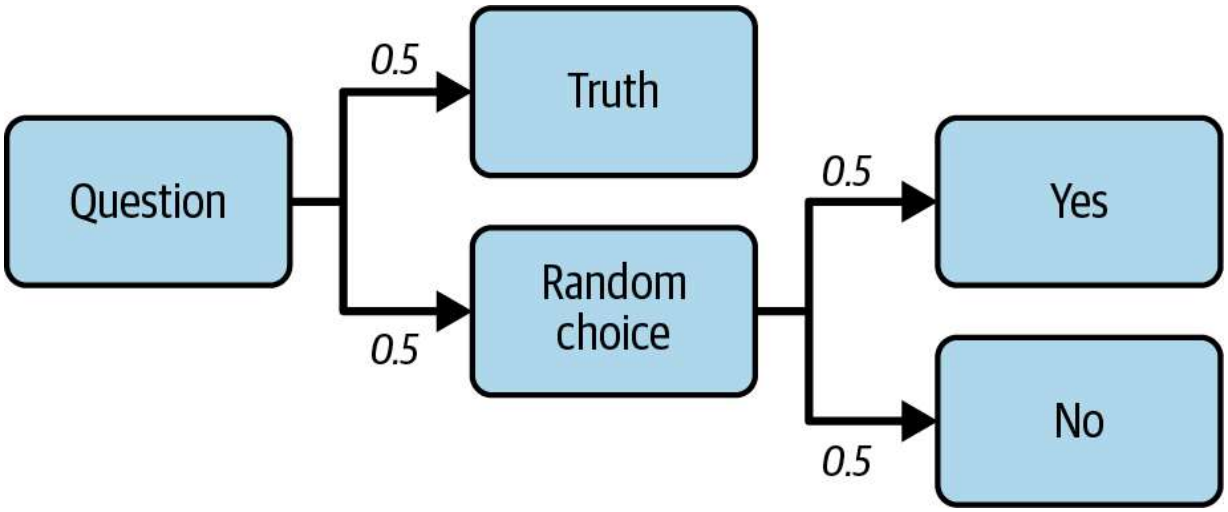

Figure 17-2. Randomized response flowchart

Local and Global DP

DP can be divided into two main methods: local DP and global DP. In *local DP*, noise or randomness is added at the individual level, as in the randomized response example earlier, so privacy is maintained between an individual and the collector of the data. In *global DP*, noise is added to a transformation on the entire dataset. The data collector is trusted with the raw data, but the result of the transformation does not reveal data about an individual.

Global DP requires us to add less noise compared to local DP. This requirement leads to a utility or accuracy improvement of the query for a similar privacy guarantee. The downside is that the data collector must be trusted for global DP, whereas for local DP only individual users see their own raw data.

Epsilon-Delta DP

Probably the most common way of implementing DP is to use ϵ - δ (the Epsilon-Delta framework). When comparing the result of a randomized transformation on a dataset that includes one specific person with another result that does not contain that person, e^{ϵ} describes the maximum difference between the outcomes of these transformations. So, if $\epsilon = 0$, both transformations return exactly the same result. If the value of ϵ is less than zero, the probability that our transformations will return the same result is greater. A lower value of ϵ is more private because ϵ measures the strength of the privacy guarantee.

In this framework, δ is the probability that ϵ does not hold, or the probability that an individual's data is exposed in the results of the randomized transformation. We generally set δ to be approximately the inverse of the population size: for a dataset containing 2,000 people, we would set δ to be 1/1,000. For more details on the math behind this, we recommend the paper "The Algorithmic Foundations of Differential Privacy" by Cynthia Dwork and Aaron Roth.

What value of epsilon should you choose? The ϵ allows us to compare the privacy of different algorithms and approaches, but the absolute value that gives us "sufficient" privacy depends on the use case. To decide on a value to use for ϵ, it can be helpful to look at the accuracy of the ML model as you decrease ϵ. Choose the most private parameters possible while

retaining acceptable data utility for the business problem. Alternatively, if the consequences of leaking data are very high, you may wish to set the acceptable values of ϵ and δ first, and then tune your other hyperparameters to get the best model accuracy possible.

Applying Differential Privacy to ML

In this section, we'll explain some of the ways that differential privacy can be applied to ML. This is just a brief overview, and, if you would like to learn more, we recommend the paper "How to DP-fy ML: A Practical Guide to Machine Learning with Differential Privacy" by Natalia Ponomareva and coauthors. In addition to the methods described here, DP can be included in a federated learning system (which we will explain in "Federated Learning"), and this can use either global or local DP.

Differentially Private Stochastic Gradient Descent

If an attacker is able to get a copy of a normally trained model, they can use the weights to extract private information. Differentially Private Stochastic Gradient Descent (DP-SGD), introduced by Martín Abadi and coauthors in 2016, eliminates that possibility by making the model training process differentially private. It does that by modifying the mini-batch stochastic optimization process by adding noise.

In detail, DP-SGD compares the gradient's updates with or without each individual data point and ensures that it is not possible to tell whether a specific data point was included in the gradient update. In addition, gradients are clipped so that they do not become too large, and this limits the contribution of any one training example. As a nice bonus, this also helps prevent overfitting. The result is a trained model that retains differential privacy, because of the postprocessing immunity property of differential privacy. Postprocessing immunity is a fundamental property of differential privacy: it means that regardless of how you process the model's predictions, you can't affect their privacy guarantees.

Private Aggregation of Teacher Ensembles

Private Aggregation of Teacher Ensembles (PATE) begins by dividing sensitive data into *k* partitions with no overlaps, and a separate "teacher model" is trained on each partition. Next, these models are queried to generate a new prediction on each example in the dataset. This query is differentially private so that you don't know which of the *k* models has made the prediction. The PATE framework shows how ϵ is being spent in this query.

The result of this query process is a new set of labeled data that maintains privacy. A new ("student") model is then trained from the new labels. The student model includes the information from the *k* hidden dataset partitions in such a way that it's not possible to learn about them. The student model

is the only model that gets deployed, and all the data and teacher models are discarded after training.

Confidential and Private Collaborative learning

Confidential and Private Collaborative (CaPC) learning enables multiple developers, using different data, to collaborate to improve their model accuracy without sharing information. This preserves both privacy and confidentiality. To do that, it applies techniques and principles from both cryptography and differential privacy. This includes using homomorphic encryption (HE) to encrypt the prediction requests that each collaborating model receives so that information in the prediction request is not leaked. It then uses PATE to add noise to the predictions from each of the collaborating models and uses voting to arrive at a final prediction, again without leaking information.

A great example of how CaPC learning can be used is to consider a group of hospitals that want to collaborate to improve one another's models and predictions. Because of health-care privacy laws, they can't share information directly, but using CaPC learning they can achieve better results while preserving the privacy and confidentiality of their patients.

TensorFlow Privacy Example

The TFP library adds DP to an optimizer during model training. The type of DP used in TFP is an example of global DP: noise is added during training so that private data is not exposed in a model's predictions. This lets us offer the strong DP guarantee that an individual's data has not been memorized while still maximizing model accuracy. You can install TFP with the following command:

```
$ pip install tensorflow-privacy
```

We start with a simple `tf.keras` binary classification example:

```
import tensorflow as tf
layers = [
tf.keras.layers.Dense(128, activation='relu'),
tf.keras.layers.Dense(128, activation='relu'),
tf.keras.layers.Dense(1, activation='sigmoid')
]
```

The differentially private model requires that we set two extra hyperparameters compared to a normal tf.keras model: the noise multiplier and the L2 norm clip. The noise multiplier hyperparameter controls the amount of random noise that is added to the gradients at each training step.

The optimizer also compares the gradients with or without each individual data point and ensures that it is not possible to tell whether a specific data point was included in the gradient update. The L2 norm clip hyperparameter clips the gradients so that they do not become too large, and this limits the contribution of any one training example. As a nice bonus, this also helps prevent overfitting.

It's best to tune the noise multiplier and the L2 norm clip to suit your dataset and measure their impact on ϵ:

```
NOISE_MULTIPLIER = 2
NUM_MICROBATCHES = 32
LEARNING_RATE = 0.01
L2_NORM_CLIP = 1.5
BATCH_SIZE = 32
EPOCHS = 70
```

The batch size must be exactly divisible by the number of microbatches. The learning rate, batch size, and epochs are unchanged from a normal training process.

Next, initialize the DPSequential model using the DP hyperparameters:

```
from tensorflow_privacy.privacy.keras_models.dp_l
model = DPSequential(
l2_norm_clip=L2_NORM_CLIP,
```

```
  noise_multiplier=NOISE_MULTIPLIER,
  num_microbatches=NUM_MICROBATCHES,
  layers=layers,
  )
```

The optimizer must be SGD. You can then compile the model as normal:

```
  optimizer = tf.keras.optimizers.SGD(learning_rate
  loss = tf.keras.losses.CategoricalCrossentropy(fr
  model.compile(optimizer=optimizer, loss=loss, met
```

Training the private model is just like training a normal tf.keras model:

```
  model.fit(
  X_train, y_train,
  epochs=EPOCHS,
  validation_data=(X_test, y_test),
  batch_size=BATCH_SIZE)
```

Now, we calculate the differential privacy parameters for our model and our choice of noise multiplier and gradient clip:

```
  from tensorflow_privacy.privacy.analysis.compute_
  import compute_dp_sgd_privacy_statement
  compute_dp_sgd_privacy_statement(number_of_exampl
```

```
  batch_size=BATCH_SIZE,
  num_epochs=EPOCHS,
  noise_multiplier=NOISE_MULTIPLIER,
  delta=1e-5)
```

The value of delta is set to 1/dataset size, rounded to the nearest order of magnitude. In this example, we've chosen 1e-5 because the dataset has 60,000 training points.

The final output of this calculation, the value of epsilon, tells us the strength of the privacy guarantee for our particular model. We can then explore how changing the L2 norm clip and noise multiplier hyperparameters discussed earlier affects both epsilon and our model accuracy. If the values of these two hyperparameters are increased, keeping all others fixed, epsilon will decrease (so the privacy guarantee becomes stronger). At some point, accuracy will begin to decrease and the model will stop being useful. This trade-off can be explored to get the strongest possible privacy guarantees while still maintaining useful model accuracy.

Federated Learning

Federated learning (FL) is another option for increasing privacy in an ML system. It is a protocol where model training is distributed across many different devices and the trained model is combined on a central server. The

raw data never leaves the separate devices and is never pooled in one place. This is very different from the traditional architecture of gathering a dataset in a central location and then training a model, and it improves privacy for the data owners because the data never leaves their device or system.

In an FL setup, each client receives the model architecture and some instructions for training. A model is trained on each client's device, and the weights are returned to a central server. This increases privacy slightly, in that it's more difficult for an interceptor to learn anything about a user from model weights than from raw data, but it doesn't provide any guarantee of privacy.

The step of distributing the model training also doesn't provide the user with any increased privacy from the company collecting the data, because the company can often work out what the raw data would have been with a knowledge of the model architecture and the weights. The key step that increases privacy in an FL setup is that the weights are securely aggregated into the central model.

FL is most useful in use cases that share the following characteristics, as described in research by Brendan McMahan and coauthors:

- The data required for the model can only be collected from distributed sources.
- The number of data sources is large.

- The data is sensitive in some way.
- The data does not require extra labeling—the labels are provided directly by the user and do not leave the source.
- Ideally, the data is drawn from close to identical distributions.

FL is often useful in the context of mobile phones with distributed data, or a user's browser. Google's Gboard keyboard for Android mobile phones is a great example of FL in production. Google is able to train a model to make better next-word predictions without learning anything about users' private messaging.

Another potential use case is in the sharing of sensitive data that is distributed across multiple data owners. For example, an AI startup may want to train a model to detect skin cancer. Images of skin cancer are owned by many hospitals, but they can't be centralized in one location due to privacy and legal concerns. FL lets the startup train a model without the data leaving the hospitals.

FL introduces many new considerations into the design of an ML system. For example, not all data sources may have collected new data between one training run and the next, not all mobile devices are powered on all the time, and so on. The data that is collected is often imbalanced and practically unique to each device. It's easiest to get sufficient data for each training run when the pool of devices is large. New secure infrastructure

must be developed for any project using FL. TensorFlow Federated is a useful library that lets you experiment with FL.

Encrypted ML

Encrypted ML is the final method of increasing privacy in ML that we want to introduce in this chapter. Like differential privacy, it seeks to increase privacy but retain model accuracy, and it's another useful technique to be aware of. It leans on technology and research from the cryptographic community and applies these techniques to ML. The major methods that have been adopted so far are HE and secure multiparty computation (SMPC). There are two ways to use these techniques: encrypting a model that has already been trained on plain-text data and encrypting an entire system (if the data must stay encrypted during training).

HE is similar to public-key encryption except that data does not have to be decrypted before a computation is applied to it. The computation (such as obtaining predictions from an ML model) can be performed on the encrypted data. A user can provide their data in its encrypted form using an encryption key that is stored locally and then receive the encrypted prediction, which they can then decrypt to get the prediction of the model on their data. This provides privacy to the user because their data is not shared with the party who has trained the model.

SMPC allows several parties to combine data, perform a computation on it, and see the results of the computation on their own data without knowing anything about the data from the other parties. This is achieved by secret sharing, a process where any single value is split into shares that are sent to separate parties. The original value can't be reconstructed from any share, but computations can still be carried out on each share individually. The result of the computations is meaningless until all the shares are recombined.

Both of these techniques come with a cost. At the time of this writing, HE is rarely used for training ML models: it causes several orders of magnitudes of slowdown in both training and predictions. SMPC also has an overhead in terms of networking time when the shares and the results are passed between parties, but it is significantly faster than HE. These techniques, along with FL, are useful for situations in which data can't be gathered in one place. However, they do not prevent models from memorizing sensitive data—DP is the best solution for that.

You can use the TF Encrypted library to try encrypted ML on your own models.

Conclusion

Data privacy is an important consideration when building ML pipelines. You should consider whether the data you're working with is PII or sensitive data, and what harms may occur if that data is exposed. You should also be aware of the legal requirements around this data, whether that's the GDPR, the CCPA, or any other regulations. If you need to comply with the GDPR, you should have a strategy for deleting or anonymizing data if a user exercises their right to be forgotten.

Methods that you can use to increase privacy include differentially private ML, federated learning, and encrypted ML. Differentially private ML is a good choice if a data scientist has access to raw data but the predictions from a model need to be kept private. Federated learning makes it possible to train a model without data leaving a user's personal device. Encrypted ML is useful when data needs to be kept private from the data scientist training the model or when two or more parties own data and want to train a model using all parties' data.

When you're working with personal or sensitive data, choose the data privacy solution that best fits your needs regarding who is trusted, what level of model performance is required, and what consent you have obtained from users. It's possible to increase privacy while still getting good accuracy from your ML model. The goals of data privacy and ML are

often well aligned, in that we want to learn about a whole population and make predictions that are equally good for everyone, rather than learning about only one individual. Adding privacy can stop a model from overfitting to one person's data. But there is always a substantial additional engineering effort involved in adding privacy to an ML pipeline.

To learn more about the topics in this chapter, we recommend *Practical Data Privacy* by Katharine Jarmul (O'Reilly). We also strongly encourage you to keep watching for new developments. We believe that it is always important to respect the privacy of your customers and to treat any PII that you have with great care.