# Considerations for Sensitive Data within Machine Learning Datasets

When you are developing a machine learning (ML) program, it's important to balance data access within your company against the security implications of that access. You want insights contained in the raw dataset to guide ML training even as access to sensitive data is limited. To achieve both goals, it's useful to train ML systems on a subset of the raw data, or on the entire dataset after partial application of any number of aggregation or obfuscation techniques.

For example, you might want your data engineers to train an ML model to weigh customer feedback on a product, but you don't want them to know who submitted the feedback. However, information such as delivery address and purchase history is critically important for training the ML model. After the data is provided to the data engineers, they will need to query it for data exploration purposes, so it is important to protect your sensitive data fields before making it available. This type of dilemma is also common in ML models that involve recommendation engines. To create a model that returns user-specific results, you typically need access to user-specific data.

Fortunately, there are techniques you can use to remove some sensitive data from your datasets while still training effective ML models. This article aims to highlight some strategies for identifying and protecting sensitive information, and processes to help address security concerns you might have with your ML data.

## Handling sensitive information

Sensitive information is any data that you and your legal counsel want to protect with additional security measures such as restricted access or with encryption. For example, fields such as name, email address, billing information, or information that could allow a data engineer or malicious actor to indirectly deduce the sensitive information are often considered sensitive.

Standards such as HIPAA and PCI-DSS specify a set of best practices for protecting sensitive data, while also informing customers about the way their sensitive data is supposed to be handled. These certifications allow customers to make informed decisions about the security of their information.

Handling sensitive data in machine learning datasets can be difficult for the following reasons:

- Most role-based security is targeted towards the concept of ownership, which means a user can view and/or edit their own data but can't access data that doesn't belong to them. The concept of ownership breaks down with ML datasets that are an aggregate of data from many users. Essentially, data engineers need to be granted view-access to an entire set of data in order to effectively use the dataset.
- Encrypting or reducing the resolution of sensitive fields is often used as a preventive measure, but isn't always sufficient for an ML dataset. The aggregate dataset itself often provides the means of breaking the encryption through frequency analysis attacks (https://wikipedia.org/wiki/Frequency_analysis).
- Random tokenization, suppression, or removal of the sensitive fields from the dataset can degrade effective ML model training by obscuring necessary data, resulting in poor performance of your predictions.

Organizations often develop tools and a set of best practices in order to strike an appropriate balance between security and utility. To help protect sensitive data in ML datasets, keep in mind the following three goals, which are discussed in the rest of this document:

- Identify sensitive data in the dataset with a high level of confidence.
- Protect sensitive data without adversely impacting the project. This can be accomplished by removing, masking, or coarsening the data you have determined to be sensitive.
- Create a governance plan and best practices documentation. This allows your data engineers as well as customers to make appropriate decisions about your sensitive data, particularly those scenarios where the sensitive data cannot be identified reliably, masked, or removed.

These three goals are discussed in detail in the following sections, which focus on scenarios where your datasets remain private within your company. This article does not cover scenarios where the datasets are meant to be shared publicly.

## Identifying sensitive data

Sensitive data might exist in your environment in several scenarios. The following sections cover five of the most common scenarios, and present methods you can use to identify sensitive data in each.

### Sensitive data in columns

Sensitive data can be restricted to specific columns in structured datasets. For example, you might have a set of columns containing a user's first name, last name, and mailing address. In this case, you identify which columns have sensitive data, decide how to secure them, and document these decisions.

#### Sensitive data in unstructured text-based datasets

Sensitive data can be part of an unstructured text-based dataset, and it can often be detected using known patterns. For example, credit card numbers in chat transcripts can be reliably detected using a common regular expression pattern for credit card numbers. Regular expression detection errors, leading to misclassification, can be minimized using more complex tools like the Cloud Data Loss Prevention API (/dlp) (DLP).

#### Sensitive data in free-form unstructured data

Sensitive data can exist in free-form unstructured data such as text reports, audio recordings, photographs, or scanned receipts. These datasets make it considerably more difficult to identify your sensitive data, but there are many tools available to help you.

- For free-text documents, you might use a natural language processing system such as the Cloud Natural Language API (/natural-language) to identify entities, email addresses, and other sensitive data.
- For audio recordings, you can use a speech-to-text service such as the Cloud Speech API (/speech), and subsequently apply the natural language processor.
- For images, you can use a text-detection service such as the Cloud Vision API (/vision) to yield raw text from the image and isolate the location of that text within the image. The Vision API can provide the coordinates for locations of some targeted items within images, and you might use this information for example, to mask all faces from images of a cash register line before training a machine learning model to estimate average customer wait times.
- For videos, you can parse each video into individual picture frames and treat them as image files, or you can use a video processing tool such as the Cloud Video Intelligence API (/video-intelligence) along with the Cloud Speech API to process the audio.

These techniques are still subject to the review and approval of your own legal counsel and depend on how well your systems are able to process free text, transcribe audio, understand images, and segment video in order to identify potential sensitive data. The Google APIs listed above, as well as Cloud DLP, are powerful tools you can incorporate into

your preprocessing pipeline. However, these automated methods are imperfect, and you will want to consider maintaining a governance policy to deal with any sensitive information that remains after scrubbing.

### Sensitive data in a combination of fields

Sensitive data can exist as a combination of fields, or manifest from a trend in a protected field over time. For example, a standard practice to reduce the likelihood of identifying a user is to blur the last two zip-code digits, reducing the zip code from five to three ("zip3"). However, a combination of zip3 associated with work and a zip3 associated with a home address might be enough to identify users with unusual home-work combinations. Similarly, a zip3 home address trend over time might be enough to identify an individual who has moved several times.

Identifying whether a dataset is truly protected in the face of a frequency analysis attack requires statistical expertise. Any scenario dependent upon human experts presents scalability challenges, and can paradoxically require the same data engineer scrubbing the data to inspect the raw data for potential problems. Ideally, you would create automated ways to identify and quantify this risk, a task beyond the scope of this article.

Regardless, you should work with your legal counsel and data engineers to assess your exposure to risk in these scenarios.

### Sensitive data in unstructured content

Sensitive data sometimes exists in unstructured content because of embedded contextual information. For example, a chat transcript might include the phrase "I called yesterday from my office. I had to go to the eighteenth floor lobby by the Cafe Deluxe Espresso, because the fourth floor has poor mobile reception."

Based on the context and scope of your training data and advice of your legal counsel, you might want to filter aspects of this content. Due to the unstructured nature and large set of complex combinations of phrases that could enable similar inferences, this is a difficult scenario to address with programmatic tools, but it is worth considering tighter governance around access to the entire unstructured dataset.

For model development it is often effective to take a subsample of this data that has been scrubbed and reviewed by a trusted person and make it available for model development. You would then be able to use security restrictions and software automation to process the full dataset through the production model training process.

# Protecting sensitive data

After you have identified your sensitive data, you must determine how to protect it.

## Removing sensitive data

If user-specific information is not necessary for your project, consider deleting all that information from the dataset before it is provided to the data engineers building your ML model. However, as discussed earlier, there are cases where removing the sensitive data dramatically reduces the value of the dataset and in these cases, sensitive data should be masked using one or more of the techniques discussed in the Masking Sensitive Data section.

Depending on the structure of the dataset, removing sensitive data requires different approaches:

- When data is restricted to specific columns in structured datasets, you can create a view that doesn't provide access to the columns in question. The data engineers cannot view the data, but at the same time the data is "live," and doesn't require human intervention to de-id it for continuous training.
- When sensitive data is part of unstructured content, but it's identifiable using known patterns, it can be automatically removed and replaced by a generic string. This is how the Cloud DLP (/dlp) addresses this challenge.
- When sensitive data exists within images, videos, audio, or unstructured free-form data, you can extend the tools you've deployed to identify the sensitive data to mask or remove it.
- When sensitive data exists because of a combination of fields, and you have incorporated automated tools or manual data analysis steps to quantify the risk posed by each column, your data engineers can make informed decisions about retaining or removing any relevant column.

## Masking sensitive data

When you can't remove sensitive data fields, it might still be possible for your data engineers to train effective models with the data in a masked format. If your data engineers determine that some or all of the sensitive data fields can be masked without impacting the ML training, several techniques can be used to mask the data.

- The most common approach is to use a substitution cipher, which involves replacing all occurrences of a plain-text identifier by its hashed and/or encrypted value. It is generally accepted as a best practice to use a strong cryptographic hash such as SHA-256 (https://wikipedia.org/wiki/SHA-2), or a strong encryption algorithm such as AES-256 (https://wikipedia.org/wiki/Advanced_Encryption_Standard) to store all sensitive fields. It is important to remember that using a salt with your encryption will not create repeatable values, and it is detrimental to ML training.
- Tokenization is a masking technique that substitutes an unrelated dummy value for the real value stored in each sensitive field. The mapping of the dummy value to the real value is encrypted/hashed in a completely different and presumably more secure database. It is worth noting that this method works for ML datasets only if the same token value is reused for identical values. In this case, it is akin to a substitution cipher and is vulnerable to frequency analysis attacks. The primary difference is that tokenization adds an additional layer of protection by pushing the encrypted values into a separate database.
- Another method of protecting data with multiple columns uses Principal Components Analysis (https://wikipedia.org/wiki/Principal_component_analysis) (PCA) or other dimension-reducing techniques to combine several features and then carry out ML training only on the resulting PCA vectors. For example, given three different fields, *age*, *smoker* (represented as 1 or 0), and *body-weight*, the data might get condensed into a single PCA column that uses the following equation: *1.5age+30smoker+0.2*body-weight.* Somebody who is 20 years old, smokes, and weighs 140 pounds generates a value of 88. This is the same value generated by someone who is 30 years old, doesn't smoke, and weighs 215 pounds.

  This method can be quite robust because even if one identifies individuals who are unique in some way, it is hard to determine without an explanation of the PCA vector formula what makes them unique. However, all PCA processing reduces the data distribution, and trades accuracy for security.

As previously noted, it is sometimes possible to break a substitution cipher using *a priori* knowledge of the frequency with which different identifiers occur "in the wild," and deriving inferences from the actual occurrence of the various encrypted identifiers. For example, the distribution of first names in a public dataset of baby names (/bigquery/public-data/usa-names) can be used to infer the likely set of names for a particular encrypted identifier. Given that bad actors might have access to the complete dataset, encryption, hashing, and tokenization are vulnerable to frequency analysis attacks. Generalization and quantization (https://wikipedia.org/wiki/Quantization) use a many-to-one mapping in their substitution, and the corresponding inference is slightly weaker but still vulnerable to a frequency analysis attack. Because machine learning datasets have a number of corresponding variables, the

frequency analysis attack can use joint-probabilities of occurrence (https://wikipedia.org/wiki/Joint_probability_distribution), potentially making the cipher much easier to crack.

Therefore, all masking methods have to be combined with an effective auditing and governance mechanism to restrict access to all machine learning datasets that could potentially contain sensitive data. This includes datasets where all the sensitive fields have been suppressed, encrypted, quantized, or generalized.

## Coarsening sensitive data

Coarsening is another technique used to decrease the precision or granularity of data in order to make it more difficult to identify sensitive data within the dataset, while still giving you comparable benefits versus training your model with the pre-coarsened data. The following fields are particularly well-suited to this approach:

- **Locations**. Population density varies across the globe, and there is no easy answer to how much you should round off location coordinates. For example, decimal-based latitudes and longitudes, rounded off to single-digit precision (e.g. -90.3, approximately within 10 km), might be sufficient to pinpoint residents of rural areas with large farms. When rounding is insufficient for coordinates, you can use location identifiers such as city, state or zipcode. These cover much larger areas, and hence make it harder to distinguish one single individual. Choose a large enough bucket size to adequately obfuscate the unique characteristics of any one row.
- **Zip Codes**. US zip codes in a 5+4 form can identify a household, but can be coarsened to include just the first three digits ("zip3"). This limits the ability to identify a specific user by putting many users into the same bucket. Again, you might want to quantify this risk since extremely large datasets enable increasingly sophisticated attacks.
- **Numeric quantities**. Numbers can be binned to make them less likely to identify an individual. For example, an exact birthday is often not required, just the decade or month a user was born. Therefore, ages, birthdays, and similar numeric fields can be coarsened by substituting ranges.
- **IP addresses** IP addresses are often part of any machine learning workflow that involves application logs, and they are often treated like physical addresses in terms of sensitivity. A good coarsening technique is to zero out the last octet of IPv4 addresses (the last 80 bits if using IPv6). This has the same function as rounding off the latitude/longitude or reducing a street address to a zip code, trading geographic accuracy for more protection. Engage in IP address coarsening as early in the pipeline

as possible: you might even be able to modify your logging software to mask or suppress IP addresses before writing them to disk.

## Establishing a governance policy

If your datasets have any amount of sensitive data, it is recommended that you consult legal counsel to establish some sort of governance policy and best practices documentation. The details of your policy are left to you, and there are many resources available, such as the PCI Security Standards Council's Best Practices for Maintaining PCI DSS Compliance (https://www.pcisecuritystandards.org/documents/PCI_DSS_V3.0_Best_Practices_for_Maintaining_PCI_DSS_Compliance.pdf), and the ISO/IEC 27001:2013 security techniques requirements available to preview here (https://www.iso.org/obp/ui/). The following list also contains a number of common concepts you can consider when establishing your policy framework:

- Establish a secure location for governance documentation.
- Exclude encryption keys, hash functions, or other tools from your documentation.
- Document all known sources of incoming sensitive data.
- Document all known locations of stored sensitive data along with what type of data is present. Include all remediation steps that have been taken to protect it.
- Document known sensitive data locations where remediation steps are difficult, inconsistent, or impossible. This covers situations where it is suspected that frequency analysis attacks could be used.
- Establish a process to continually scan for and identify new sources of sensitive data.
- Document the roles and (possibly) individual employee names who have been granted temporary or permanent access to sensitive data. Include information as to why they required the access.
- Document the processes by which employees request access to sensitive data. Specify where they can access sensitive data, if, how, and where they can copy it, and any other restrictions associated with access.
- Establish a process to regularly review who has access to what sensitive data and determine whether access is still required. Outline what to do when employees leave or change roles as part of an off-boarding process.

- Establish a process to communicate the policies, enforce them, and regularly review them.

## Next steps

- Review Creating a PCI-DSS-Compliant Environment on GCP (/solutions/pci-dss)
- Learn more about the Data Loss Prevention (/dlp)
- Learn more about the Cloud Natural Language API (/natural-language)
- Learn more about the Cloud Speech API (/speech)
- Learn more about the Cloud Video Intelligence API (/video-intelligence)
- Learn more about the Cloud Vision API (/vision)
- Try out other Google Cloud features for yourself. Have a look at our tutorials (/docs/tutorials).



Last updated 2020-01-02 UTC.