Sensitive Data Discovery

[**Motivation** 1](#_heading=h.30j0zll)

[**Requirement** 2](#_heading=h.1fob9te)

[Assumptions 2](#_heading=h.3znysh7)

[Use-case 2](#_heading=h.qsh70q)

[**Sensitive Information** 2](#_heading=h.2et92p0)

[Personally Identifiable Information (PII) 2](#_heading=h.tyjcwt)

[Sensitive personal information (SPI) 3](#_heading=h.3dy6vkm)

[Personally identifiable financial information (PIFI) 3](#_heading=h.1t3h5sf)

[Protected Health Information PHI 3](#_heading=h.4d34og8)

[Organization/business specific Sensitive Information 3](#_heading=h.2s8eyo1)

[**Solution** 3](#_heading=h.17dp8vu)

[Data 4](#_heading=h.3rdcrjn)

[Pre-trained ML models 5](#_heading=h.35nkun2)

[Scan and collect data from the file system 5](#_heading=h.3as4poj)

[Categorizer 5](#_heading=h.1ksv4uv)

[Pre-trained models 6](#_heading=h.1pxezwc)

[Text Analysis 7](#_heading=h.26in1rg)

[**References** 7](#_heading=h.2jxsxqh)

[**Appendix** 8](#_heading=h.z337ya)

[ElasticSearch Stack 8](#_heading=h.3j2qqm3)

[Retrieve data 8](#_heading=h.1y810tw)

[RegEx based classifier/categorizer 10](#_heading=h.4i7ojhp)

[Google Data Loss Prevention (DLP) API 11](#_heading=h.2xcytpi)

[Passwords 11](#_heading=h.1ci93xb)

[File types 12](#_heading=h.2bn6wsx)

## **Motivation**

Typically, an organization would like to identify systems or users in possession of files with potentially sensitive/critical data (like financial data like accounts, PII docs, legal docs, code etc.) that could cause problems if leaked outside the organization. Does a user have access to sensitive data or data itself that he/she is not supposed (according to their role) to have access to? The security tool is expected to flag such files on any system of a user in the organization.

## **Requirement**

Give a ML framework that an organization can use to train a set of labelled (categorized) documents and generate a trained ML Model (or a set of trained ML Models). The categories and the data/file formats are specific to an organization. The trained ML Model (or the trained set of ML Models) will then be used to categorize an unseen/unlabeled document (if it falls in the set of identified categories). The trained ML Model will be run on all the files of the systems in the organization to identify if any of the files on the systems fall into the above identified categories.

### Assumptions

1. The data is in English language
2. Emails are not checked

### Characteristics of the problem

1. Text analysis/ NLP
2. Multi-Classification problem
3. Do not have all the samples during learning. Samples continuously come in the deployment
4. Do not have samples for negative class i.e. samples not belonging to any class we know
5. Samples are not uniformly distributed across the classes.
6. Since the data is sensitive the data cannot be persisted in the database

### Requirements

* Collect and analyze the **data** (metadata + content) on a system. Data: files, what else???
* Tag data to identify the category to which it belongs
* Rank the data in terms of its criticality (wrt the organization)
* For each **user**, identify the mismatch in their role and kind of data they have access to.
* Define a pipeline architecture to check various categories. Check for each category can be implemented differently. Allow possibility to add or remove category plugins.???

## **Sensitive Information**

Sensitive information is a piece of data owned by entities or individuals, which, if potentially lost, damaged or compromised, bares significant financial and integrity damages to the enterprise or person of interest.

1. **Personally Identifiable Information (PII):** PII is data that could potentially identify a specific individual. PII, according to the U.S. Office of Management and Budget, is any information that can be used to uniquely identify, contact or locate an individual, or can be used with other sources to uniquely identify a person.

*Name, Date & place of birth, Mother’s maiden name, Phone Number, Addresses, email ids, Unique ids - Social Security Numbers (SSNs), National ID, Passport numbers, Biometric records, Driver’s license numbers, tax-related data,* [*passwords*](#_heading=h.4i7ojhp)

1. **Sensitive personal information (SPI):** SPI refers to information that does not identify an individual, but is related to an individual, and communicates information that is private or could potentially harm an individual should it be made public.

*Biometric data, genetic information, sex, trade union membership, sexual orientation, etc.*

1. **Personally identifiable financial information (PIFI)**: PIFI is any type of [PII](https://searchfinancialsecurity.techtarget.com/definition/personally-identifiable-information) that is linked to that person's finances.

*Credit card information, bank account numbers, PCI - Payment card information*

1. **Protected Health Information PHI**: Sensitive patient health data includes insurance information, medical information and patient data such as Social Security numbers and other sensitive information that is not in the public domain.

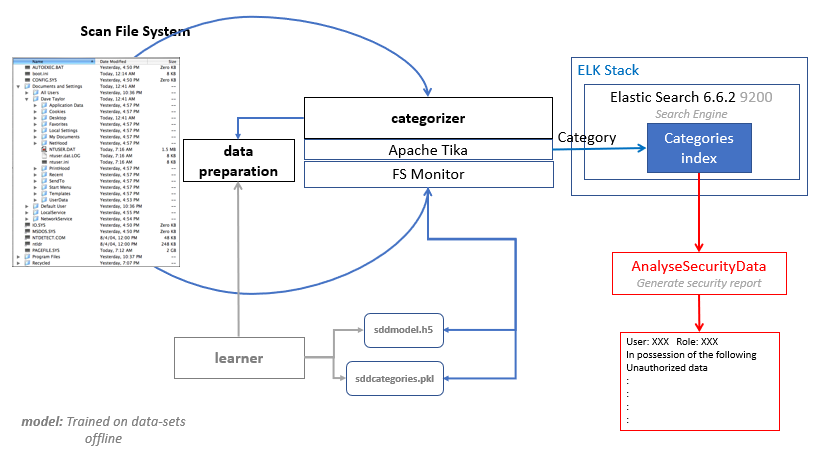
*Medical information, health and insurance records, Health Data, International Classification of Diseases - ICD, National drug codes*

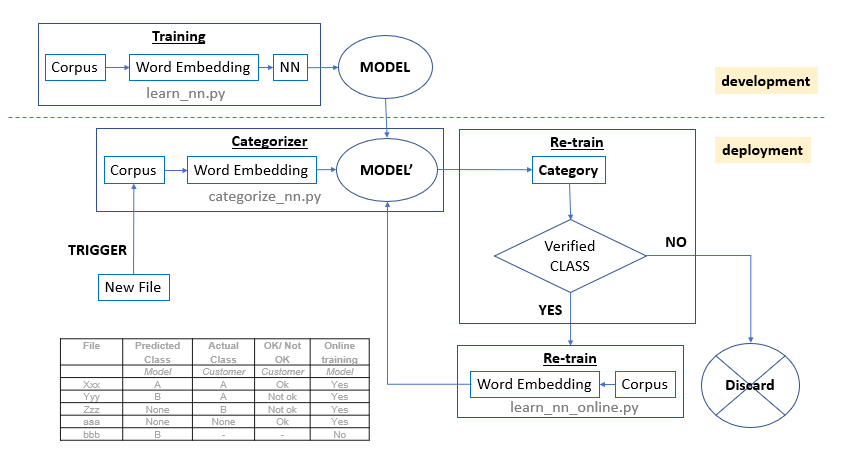
1. **Organization/business specific Sensitive Information**: Confidential business information, and other domain-specific sensitive information.

*Business engagement contracts, customer facing materials, technical solution documents, sales data, business intelligence data, management reports, trade secrets, research and customer information and public newsletters, source code, IpAddresses, Legal, intellectual property, Security certificate*

## **Solution**

**Text data** -- **NLP** -- **Topic Modelling** – **Classification**





Scan/Parse data on the system → Preprocess data → Identify topics/clusters → Update ElasticSearch

### Data

Organization should identify the various categories of data it wants to identify and secure. For example - Recon files, source code etc. It needs to prepare data-sets for each category for an ML model to be trained on.

Data can be in many [formats](#_heading=h.2bn6wsx) - files, database and the files can be of different types - word doc, excel, pdf, text files. The files can be with correct/incorrect or no extension.

Further the content of the files can be different

* **Structured**: Files as recon files or customer data will mostly contain continuous numerical values (amount, date etc.) and innumerable discrete values (customer, company names etc.). Most of the structured data will be found in excel, csv files.

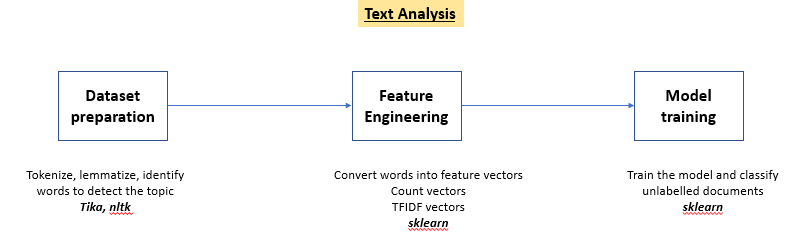
**Challenges:**

If data (that is inherently structured – series of unique values) is treated as text data, the number of features (vocabulary) will be very high and there will be no meaningful set of features. Also, the validation set might have totally different values and may not match to the model’s feature set. So, these kinds of documents cannot be treated as NLP problem. One option is to convert the continuous values to words (type of values + size etc) and treat it as an NLP problem.

* **Unstructured**: Files like Business proposals, Technical documents, source code contain dominantly free flowing text with a reasonably finite vocabulary. These can be treated as an NLP problem.

### Text Analysis

We treat the data-set as text data and treat the problem at hand as NLP problem.



1. **Dataset Preparation:** Tokenize + Lemmatize + Remove stop words
2. **Feature Engineering:** Extract features from text data.  **Vectorization** is the general process of converting a collection of text documents into numerical feature vectors
   1. word/token count (CountVectorizer)
   2. term frequency- inverse document frequency (TfidfVectorizer)
   3. represent (embed) words in a continuous vector space where semantically similar words are mapped to nearby points ('are embedded nearby each other') (WordEmbedding or Word2Vec)

Note: The above 2 steps should be done for training set, validation set and for the data set on which prediction will be done.

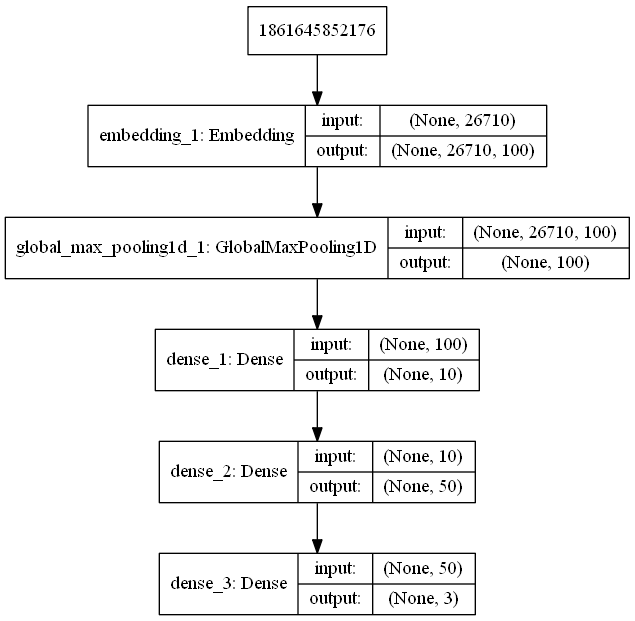
**To be considered**:

* **Option:** Named-entity recognition (NER) (also known as entity identification, entity chunking and entity extraction) is a subtask of [information extraction](https://en.wikipedia.org/wiki/Information_extraction) that seeks to locate and classify [named entity](https://en.wikipedia.org/wiki/Named_entity) mentions in [unstructured text](https://en.wikipedia.org/wiki/Unstructured_data) into pre-defined categories such as the person names, organizations, locations, [medical codes](https://en.wikipedia.org/wiki/Medical_classification), time expressions, quantities, monetary values, percentages, etc. NER is used in structured data (csv, xlsx) to replace continuous values to <NERType>+<Size>
* Can we use [Google Data Loss Prevention (DLP) API](#_heading=h.2xcytpi) to identify the entities (phone number, SSN number etc.) and use them as the tokens?
* Add additional features - number of words/tokens in a file, meta-data, number of spelling mistakes, number of complex words, number of parts-of-speech, Indianized words, Counts of types of numbers (dates, ints, floats, currency, with units) and not the exact numbers themselves, (useful/domain related) words

### Training ML model

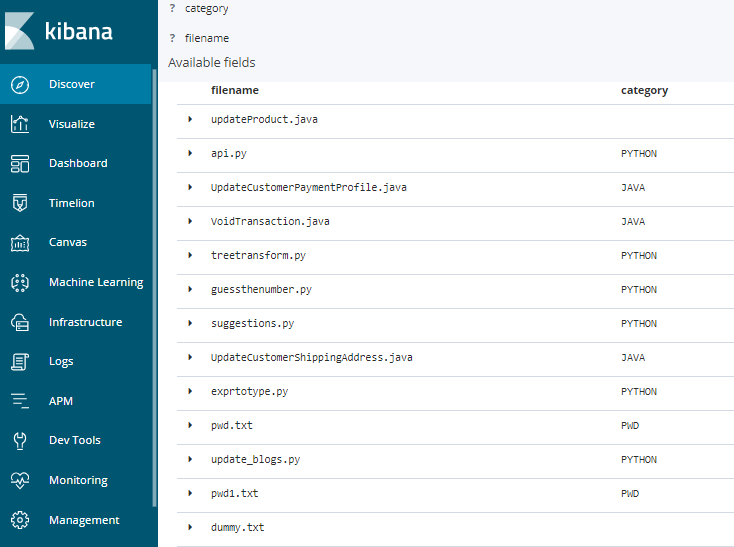
#### Neural Networks – Deep Learning

Keras offers an [Embedding](https://keras.io/layers/embeddings/#embedding) layer that can be used for neural networks on text data. It requires that the input data be integer encoded, so that each word is represented by a unique integer.



### Categorizer

The categorizer contains a set of **pre-trained ML model**. The categorizer reads the content of each file and runs the ML model to classify the file (> 0.7). If it identifies a category, then it is a sensitive file. Else it is not a sensitive file. It then updates the Categories Index (in Elastic Search) - the file name, location and the category.



**Alternate solution:** [RegEx based classifier](#_heading=h.49x2ik5)

**Full scan:** For the first time, all the files should be categorized.

**Subsequently:** Identify only changed files in the file system using FSMonitor and categorize only the changed files.

### Phases

In the production environment, the following phases exist

1. **Prediction** phase: When the model is run, it predicts the category of a file.
2. **Verification** phase: The customer checks the predicted categories and marks it as correct or incorrect
3. **Retraining** phase: The model trains on all the files that were marked by the customer. The model does not train on the files that were not marked by the customer. The retraining can be done as and when the customer marks each file or can do it in batches. If the retraining accuracy is beyond a certain threshold, the model will be considered/saved. Else it would be discarded.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **File** | **Predicted Class** | **Actual Class** | **OK/Not OK** | **Re-training/ Online training** |
|  | *Model* | *Customer* | *Customer* | *Model* |
| Xxx | A | A | Ok | Yes |
| Yyy | B | A | Not ok | Yes |
| Zzz | None | B | Not ok | Yes |
| aaa | None | None | Ok | Yes |
| bbb | B | - | - | No |

**Open**: What would the verification dataset be? Currently this is a static data-set. This should also be changing. k-fold cross-validation

### Next Steps

#### Online learning *todo*

**Online learning** is about learning from one sample at a time (one data point at a time), but you're still updating all the weights of the neural network. Also known as - Realtime learning, continual learning

* Online learning is data efficient because once data has been consumed it is no longer required. Technically, this means you don’t have to store your data.
* Online learning is adaptable because it makes no assumption about the distribution of your data. As your data distribution morphs or drifts, due to say changing customer behavior, the model can adapt on-the-fly to keep pace with trends in real-time.

Online learning is prone to [*catastrophic interference*](https://en.wikipedia.org/wiki/Catastrophic_interference)*—*more so than most other techniques*.*

#### Others

* Categorize on different files
* Learn/train on more samples/files
* How do you see it working at customers – different customers different files
* Implement in tensorflow
* Attempt to extract tables from pdf files.
* Text Analysis + Machine Learning feature of Elastic Search
* CNN model

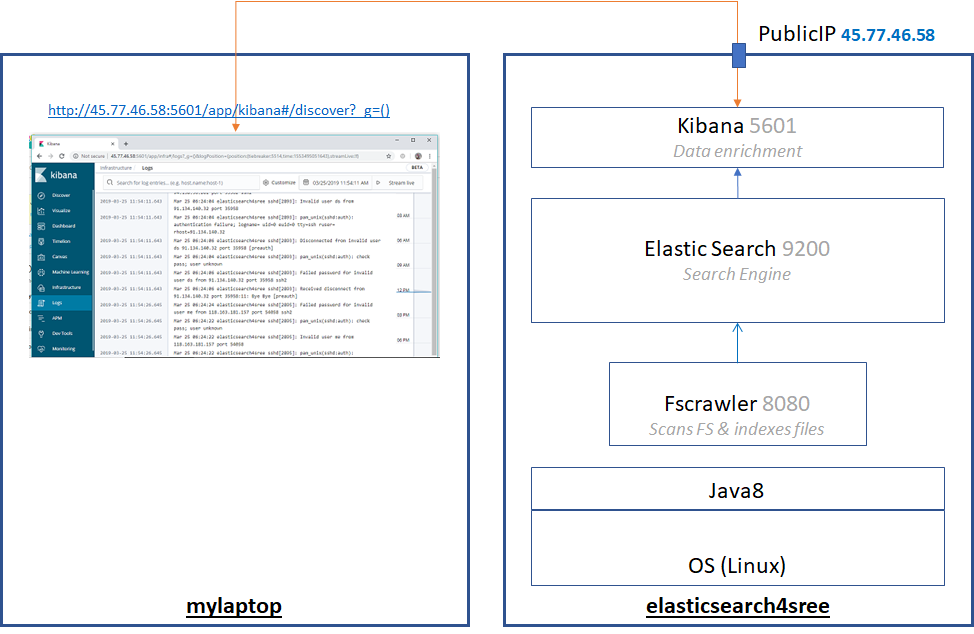
## **References**

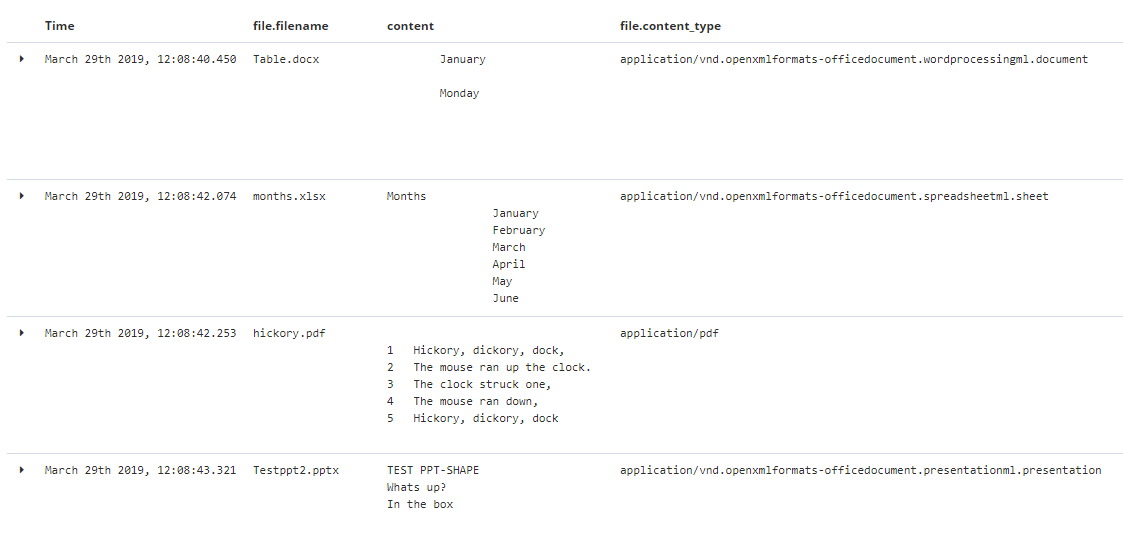
* <https://digitalguardian.com/blog/what-data-discovery>
* <https://researcher.watson.ibm.com/researcher/files/us-mqiao/CLOUD2017.pdf>
* <https://www.groundlabs.com/enterprise-recon/>

## **Appendix**

### ElasticSearch Stack

Stores data in JSON format and provides RESTful APIs to manage - **Retrieve, Store & Analyze data**

**a**

****

### FSCrawler

**Fscrawler** scans the file system and indexes the content of all the (pdf, doc, ppt, xls) files in a temp index (in Elastic Search).

**Issues**:

* Couldn't make it work as a service. But can run it once and get all the contents of the files indexed into ElasticSearch. See [issue](https://discuss.elastic.co/t/fscrawler-is-not-indexing-consistently/174267/4)
* fscrawler is unable to identify updates in the files (content added, deleted and modified) and folders (files added, deleted and modified)

### Retrieve data

**FileBeat**: Collect, parse, and visualize common log formats

|  |  |
| --- | --- |
| **Can do** | **Can’t do** |
| Can harvest multiple **text** files - log or custom non-log text files | Each line comes as a single line. Can parts of the line be configured as fields?  Prospectors - fields - fields\_under\_root |
|  | Had some incorrect processing |
|  | Could not recognise a line added on the top |

**Other Options**

1. Use ElasticSearch and Ingest Attachment Processor Plugin: The forums suggest it is not possible to use IAP on a file directly.
2. Develop a custom component that

* Converts the content of the file to base64 encoding
* Add this base64 encoded content to the index using ES+IAP.

This means we would have to keep checking all the directories for changes and add the new/updated data to the index

**TODO**

**Modules**: simplify the collection, parsing, and visualization of common log formats

**Prospectors**: locate and process files

**Processors**: filter and enhance the data before sending events to the configured output.

**Ingest Attachment Processor Plugin:** extract file attachments in common formats (such as PPT, XLS, and PDF)

**Geo-IP Processor** (default): adds information about the geographical location of IP addresses

**User-agent Processor** (default): extracts details from the User-Agent header value

Mapper Annotated Text Plugin

**Pipeline:** Series of processors that are to be executed

**Mapping:** Defining how a document and the fields it contains are stored and indexed

**Analyser**:

### Model per category using OneSVM

**Pretrained models:** For each category of data (as identified by the organization), an ML model is trained. Each ML model is pre-trained offline on datasets of various categories - bank, pwd, source code etc.

The problem at hand is a classification problem.

Since we have only the positive class(es) dataset and do not have a comprehensive dataset for the negative class (files or data that is not sensitive), **OneClassSVM** is the most appropriate and is chosen as the classifier.

**Alternate algorithms**:

* Convolutional Neural networks:
  + **Step1**: Generate word2vec word embeddings. This will be the input layer of the CNN
  + *You could have a separate channels for different word embeddings (*[*word2vec*](https://code.google.com/p/word2vec/) *and* [*GloVe*](http://nlp.stanford.edu/projects/glove/) *for example)*
* Decision tree/Random Forest, Logistic Regression, Naïve Bayes - gaussian, multinomial, bernoulli
* PU-learning, one-class learning, semi-supervised learning

[Metrics to evaluate algorithms](#_heading=h.3whwml4)

**Categorizer:** The categorizer contains a set of **pre-trained ML models**. The categorizer reads the content of each file from the Temp index and runs each of the ML models to identify the category to which the file belongs. If it matches any of these, then it is a sensitive file. Else it is not a sensitive file. It then updates the Categories Index (in Elastic Search) - the file name, location and the category.

### Topic Modelling

Topic Modelling is a process to automatically identify topics present in a text object and to derive hidden patterns exhibited by a text corpus. Topic Modelling is different from rule-based text mining approaches that use regular expressions or dictionary-based keyword searching techniques. It is an unsupervised approach used for finding and observing the bunch of words (called “topics”) in large clusters of texts.

Topics can be defined as “a repeating pattern of co-occurring terms in a corpus”. A good topic model should result in – “health”, “doctor”, “patient”, “hospital” for a topic – Healthcare, and “farm”, “crops”, “wheat” for a topic – “Farming”.

Latent Dirichlet Allocation (LDA) is a generative probabilistic model which can ﬁnd latent semantic topics within a collection of documents. LDA is a matrix factorization technique.

### RegEx based classifier/categorizer

Looks (regex search) for PII (Personally Identifiable Information) (credit card numbers, ssns, email-ids, phone numbers, passports etc. ) in the files. Possible to add other PII plugin.

The density of PII or financial data will be quite high in the documents. If name, address, phone number, credit card kind of data occur together in a document, then it's relevance and criticality increases

**Challenges**:

* PII data can be in different formats

4916 9766 5240 6237 → Spaces in between the 4 digit tuples

4916976652406237 → No spaces between the 4 digit tuples

credit-card:4916 9766 5240 6237, (4916 9766 5240 6237) → Preceded/Followed by a string w/o spaces

12345491697665240623712345 → Preceded/Followed by numbers w/o spaces

Archive/\*.xls files: 607532XXXXXXXXXX → This is clearly a credit card number as it has a label. Not sure about the rest though.

21012018/visaout\* files: 060040339835XXXXXXXX000

21012018/CTF\*.txt files: 0505468019055XXXXXXXX000079323535600

* **ssn**
* **phonenumber:** Mostly erroneous as they can contain any number and can be in any 10-digit format immediately preceded/followed by text or numbers. Not reliable.
* **emailid**: Mostly works. Cannot verify if it is a valid (existing) email id.
* **Birthdate**: All dates will be picked up. Cannot distinguish between the type of dates.
* Since there is no single place where we can get all the valid ccn formats, I will include the ones we know now (BINmaster.xls) and the rest will have to be added/updated (as a regular expression in a python dictionary data structure) as and when we learn/discover new formats.
* If there are files of specific format that need to be searched, then the search can be customized for that. But if the file format can be generic, then it is difficult to make the regular expression based search work 100% accurately in all cases. Will the search be only in the type of files (that were shared by Sreedhar)?
* **Passwords**: Stuff such as AWS/Azure keys or Social media account passwords or passwords to other databases or repositories that can get them into other systems and cause further hacking. How to identify these?
* How to verify that it is identifying all the cases?

|  |  |
| --- | --- |
| **Library** | **Remarks** |
| CommonRegex | The results are not accurate. Shows lot of bad matches |
| piianalyzer | Depends on Common Regex. |
| Apache tika | Evaluated this for file type identification. Identifies using the extension. Not useful for our case. |

### Identification of type of files

Again not 100% accurate. The magic library that I am using is able to identify .c,.cpp,.java, .py files accurately (even if the extension is incorrect). But is unable to identify files like .js, .ts,.R files (even with the extensions). There is Apache tika library that identifies the type of file but using the extensions.

[Comparison of magic and tika libraries](https://docs.google.com/spreadsheets/d/1rHaaTwQtEZiXc3IioyrtdvdcjVWTF2bmKUrkvJC0e34/edit#gid=207227525)

### Google Data Loss Prevention (DLP) API

[API](https://cloud.google.com/dlp/) -- [Demo](https://cloud.google.com/dlp/demo/#!/)

The DLP API gives GCP users access to a classification engine that includes over 40 predefined content templates for credit card numbers, social security numbers, phone numbers and other sensitive data. Users send the API textual data or images and get back metadata such as likelihood and offsets (for text) and bounding boxes (for images)

Like all ML algorithms, the DLP API generates a probability score used to classify the likelihood that a given set of data matches one of the protected data types. Of course, the algorithms work on text, as is best seen by trying this demo application, but what makes Google's system so impressive is that it also works with images, for example, numbers from a scanned credit card as it showed in this presentation. The DLP API currently works with data stored in Google Cloud Storage (object) BigQuery (data warehouse), and Cloud Datastore (NoSQL database), but since it's a REST API, can also be integrated with external data sources

Amazon Macie automatically scours the data and classifies it by file and content type, using both file metadata and the raw content.

### Passwords

From <https://www.spirion.com/howitworks/identities/passwords/>

Passwords are not typically recognized as part of an individual’s identity. However, the increasing number of capabilities offered through online institutions allows users with passwords to transfer money from bank accounts, purchase goods, and many other highly private activities. As a result, passwords are highly sensitive pieces of information and are considered one of the easiest forms of identity theft.

Organizations looking to secure sensitive data and stop data loss should absolutely include passwords within their scope.

**Passwords are often stored in files such as text documents and Microsoft Office documents** as well as in system areas like the Windows Registry. Sometimes users intentionally store this information to help them remember their logins. Other times, applications store this information unbeknownst to the user.

Regardless of reason, hackers seek this information to help them get access to secure locations. In addition, Spirion automatically finds password using its AnyFind technology. Once found, user can clean this sensitive information to prevent a data breach from ever occurring. Spirion first starts with deep, deep data analysis. When Spirion comes across data that could potentially be a match, it applies proprietary validation techniques developed by our data experts to determine whether the data is likely to be a Password. Beyond data analysis, Spirion also performs contextual analysis to increase the accuracy of returning relevant information. Doing this helps Spirion ignore random strings and increase the accuracy of finding Passwords.

### Metrics to evaluate algorithms

1. Many metrics (apart from accuracy are available) wrt classification models. Will record and analyze those. - Sensitivity, Specificity, CM, ROC, AUC, Gain, Lift, Youden’s index, Gini Criteria (in case of DT)
2. Features/words understanding using plots is needed
3. Visualize
4. Train error, Validation error, Test error
5. Plot Learning curves → To decide if getting more data will help or not
6. Manually check in which cases the algorithm is making errors
7. PCA - Reduce feature dimension for performance and visualization
8. (Hyper)Parameter tuning using sklearn grid search
9. Start by building a very simple algorithm that you can implement quickly. implement it and then test it on my cross-validation data.
10. Plot learning curves of the training and test errors to figure out if you're learning algorithm maybe suffering from high bias or high variance. Decide if having more data, more features, and so on are likely to help
11. Error analysis: Manually check the examples from CV set where the algorithm has made errors on. See it you can spot any systematic patterns in what type of examples it is making errors on.
12. Use a single real number evaluation metric. This can then be a vehicle for you to try out different ideas and quickly see if the different ideas you're trying out are improving the performance of your algorithm.

### File types

|  |  |
| --- | --- |
|  |  |
| **Application Logs** |  |
| **Database** |  |
| [**HyperText Markup Language**](https://tika.apache.org/1.20/formats.html#HyperText_Markup_Language) | HTML |
| [**XML and derived formats**](https://tika.apache.org/1.20/formats.html#XML_and_derived_formats) | XML |
| [**Microsoft Office document formats**](https://tika.apache.org/1.20/formats.html#Microsoft_Office_document_formats) | xls, doc, ppt |
| [**OpenDocument Format**](https://tika.apache.org/1.20/formats.html#OpenDocument_Format) |  |
| [**iWorks document formats**](https://tika.apache.org/1.20/formats.html#iWorks_document_formats) |  |
| [**WordPerfect document formats**](https://tika.apache.org/1.20/formats.html#WordPerfect_document_formats) |  |
| [**Portable Document Format**](https://tika.apache.org/1.20/formats.html#Portable_Document_Format) | pdf |
| [**Electronic Publication Format**](https://tika.apache.org/1.20/formats.html#Electronic_Publication_Format) |  |
| [**Rich Text Format**](https://tika.apache.org/1.20/formats.html#Rich_Text_Format) | rtf |
| [**Compression and packaging formats**](https://tika.apache.org/1.20/formats.html#Compression_and_packaging_formats) |  |
| [**Text formats**](https://tika.apache.org/1.20/formats.html#Text_formats) | Tar, AR, ARJ, CPIO, Dump, Zip, 7Zip, Gzip, BZip2, XZ, LZMA, Z and Pack200 |
| [**Feed and Syndication formats**](https://tika.apache.org/1.20/formats.html#Feed_and_Syndication_formats) |  |
| [**Help formats**](https://tika.apache.org/1.20/formats.html#Help_formats) |  |
| [**Audio formats**](https://tika.apache.org/1.20/formats.html#Audio_formats) | mp3, mp4 |
| [**Image formats**](https://tika.apache.org/1.20/formats.html#Image_formats) | PNG, GIF and BMP, Jpeg and Tiff . |
| [**Video formats**](https://tika.apache.org/1.20/formats.html#Video_formats) | MP4, Quicktime, 3GPP |
| [**Java class files and archives**](https://tika.apache.org/1.20/formats.html#Java_class_files_and_archives) |  |
| [**Source code**](https://tika.apache.org/1.20/formats.html#Source_code) |  |
| [**Mail formats**](https://tika.apache.org/1.20/formats.html#Mail_formats) |  |
| [**CAD formats**](https://tika.apache.org/1.20/formats.html#CAD_formats) |  |
| [**Font formats**](https://tika.apache.org/1.20/formats.html#Font_formats) |  |
| [**Scientific formats**](https://tika.apache.org/1.20/formats.html#Scientific_formats) |  |
| [**Executable programs and libraries**](https://tika.apache.org/1.20/formats.html#Executable_programs_and_libraries) |  |
| [**Crypto formats**](https://tika.apache.org/1.20/formats.html#Crypto_formats) |  |
| [**Database formats**](https://tika.apache.org/1.20/formats.html#Database_formats) |  |
| [**Natural Language Processing**](https://tika.apache.org/1.20/formats.html#Natural_Language_Processing) |  |
| [**Image and Video object recognition**](https://tika.apache.org/1.20/formats.html#Image_and_Video_object_recognition) |  |