IS YOUR PASSWORD SAFE?

How your choice of email provider, the platform you use and your habits have an influence on the strength of you passwords.

SIM219 - Economics of DATA
Laurie CIARAMELLA

CUVELIER Matteo LEPORCHER Vincent PUSKAS-JUHASZ Ruila

Index

Introduction

Abstract

Passwords are everywhere and often they are artificially protecting us because we don't make them complex enough. The idea here is to study the impact of geographical location, of the platform used and of email address providers impact on the password strength.

Key words

1. **Password**
2. **Strength**
3. **Vulnerability**
4. **Geographical Location**
5. **Email provider**

Introduction

There is a fair amount of already existing literature on the topic of password protection. There are two main categories : the literature analyses either the technical methods to crack passwords, such as algorithms and pseudo-algorithms, either individuals' and service providers' behavior in the choice of the passwords. Hence, there is no existing analysis of the factors influencing password strength *per se.*
This is why the choice of this subject appeared quite challenging. After our analysis, we will be able to rule on wether the password strength depends on particular factors or not. We will have a thorough analysis on the mechanisms influencing the choice of password. There is a lot at stake, since password leaks have had considerable consequences in the past, and it is likely that such leaks will happen in the future as well.
Thus, through this report, we will try to answer the question : What are the correlations between the type of platform, the e-mail providers, the geographical area and the necessary time to crack your password?
A first very telling datum in the analysis of password weakness, from an article in the New York Times [1], we learn that from leaked password lists, 20% of passwords are covered by a list of only 5000 common passwords. It means that a hacker can crack immediately 20% of every password database in the world, if he knows the 5000 most common passwords. It is clear that this ratio is not really favorable to the users.

I - State of the art

Most of the already existing literature focuses on **password cracking-methods** or on **human decision** according to platform requirements.

Numerous publications focus on an **empirical analysis of password strength**. Features of this topic include the analysis of **predictability of user-chosen passwords** and of dictionaries and **probabilistic models** enabling to crack passwords in a small amount of time. In a Conference in March 2010 [2], three researchers listed a few methods to efficiently crack passwords. Those methods are all the more efficient that they rely on the predictability of users, who tend to use passwords that are easy to remember and therefore easy to crack.

Other scientists have contributed to measure the strength of passwords, using **probabilistic and Markov models** [3]. Other publications have focused on the **individuals' behavior linked with the platform**. A research has shown that individuals' password choice varies according to the type of the platform (personal, organisational, etc) [4]. An interesting article about **online nudges** has made it possible to illustrate the difference between people and **risk aversion** [5], or people and **password reuse** [6]. The two latter show that all users are not equal when it comes to risk aversion, just like in finance for instance, some are more willing to take risks than others. But it is unsure wether this is risk aversion or simple ignorance of the risks.

The human factor in decision-making in general has been subject of other research topics [7] showing the diversity of possible **outcomes** when considering the event of choosing a password for an online platform. The possible outcomes, *ie*, the possible strength of the password, depends on factors such as the **type of platform**, the **age**, the **level of education**, the **revenue**, the **email provider** and the **geographical area**.

According to these remarks provided by the last source, it appears relevant to focus on some of these factors.

II - Data Description

II.1 - The idea behind the data

Why talking about passwords? Why now? These are the two questions that you should ask when reading this report. In the past decades, the world wide web has become ubiquitous and has totally changed all the economy and how we live our everyday life.

The point is that for you digital life, from buying things on Amazon to protecting sensible data of your company such as industrial or political secrets, **you protect everything with passwords**. Your phone is locked with a password, your bank account is protected with one too, same for your Facebook account, even your watch or your door if you have connected devices…

The passwords are so central that a **new type of digital systems** are emerging to help you **manage with all your passwords** not to forget them (such as 1Password…). And guess what? What is protecting these new services? A password!

Moreover, during the last years **uncountable leaks of passwords** happened in the biggest tech companies (Steam, Netflix, Facebook several times, Canadian Banks…) which leads to the following question : how to make my password secure enough to protect my data and which platforms are more dangerous than the others?

These mass leaks of passwords gave huge strength to a **black market of a new type**. Before when someone talked about the black market it was about organs, people or fake money… But now we can see the rise of a new **cyber black market** on which you can **buy stolen credit cards** number or cracked/stolen **passwords**.

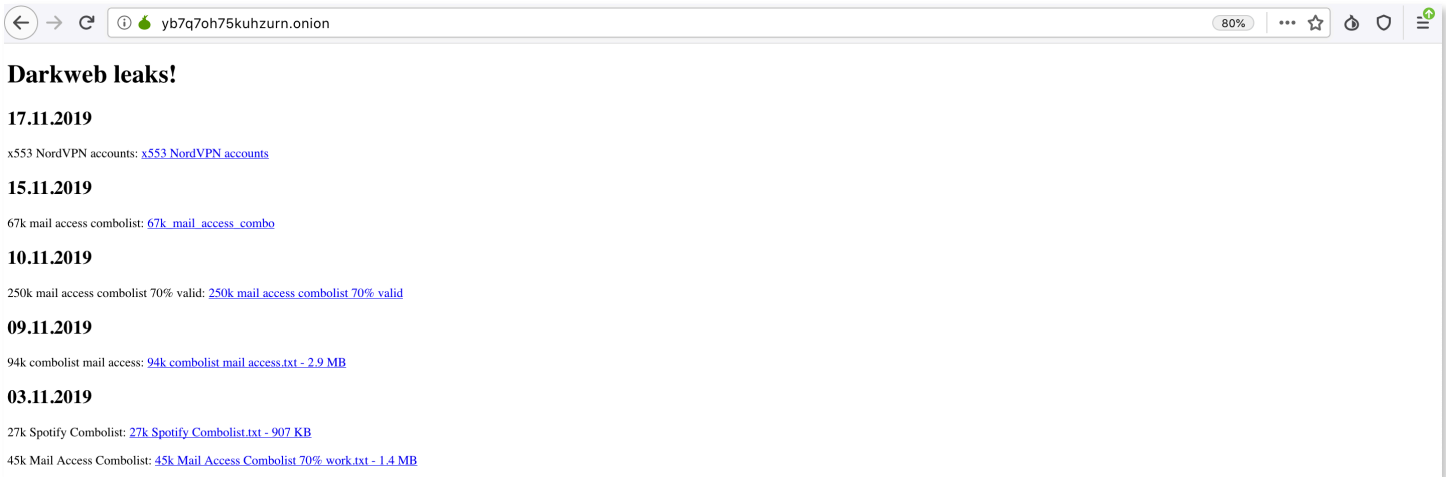
Believe me, this market I really **easy to access**. All you need is to download the Tor network and use clear keywords on the dark web equivalent of Google. As soon as the page is charged on your computer you can exchange some dollars for a PayPal Account, a Netflix Account, a credit card… Anonymously!

II.2 - Data scrapping

Considering all of this you can imagine that several **databases of leaked passwords are available on the dark web** (in fact it's only a really little part of the dark web but it is enough to find what we needed). We thus downloaded the **TOR network** to access these databases.

We searched a lot because there are an uncountable number of websites that offers to buy these passwords but really few that gives freely this data. We finally found a website (please don't try to pronounce this) yb7q7oh75kuhzurn.onion where we have been able to **download recent database** (less than a year) about **7 platforms** : Mail providers, Fortnite, Steam, Netflix, Minecraft, NordVPN and Spotify.

There was in total 11 text files regrouping couples of identifiers and passwords for these platforms, in total more than **300 000 couples**.



II.3 - Data processing

We are now with 11 text files, regrouping more than 300 000 couples of passwords and identifiers. Here is how we processed them to **make them exploitable**.

II.3.1 Exploiting text files

The very first thing to do was to **convert these raw text files to pandas DataFrame**, an exploitable format in python (see code at the end of the report).

For that we inspected the structure of the text files. For every line there is a couple of mail address (the identifier) and of a password separated by " : ".

Example of a line : *rianab@lantic.net:thinus*

To make them exploitable we followed these steps :

Use the open library of python to load documents
Create an empty list LIST
Treat iteratively each line of the text files by :

a. Split the line into two elements : the email address and the password
b. Add the platform and it's id (necessary to have digital data to do regressions) to the two elements not to loose any data
c. Add it to LIST
d. If the line includes a expiration date of the account (sometimes there is), delete it

Convert the list LIST into a pandas DataFrame
Name the columns of the DataFrame

Finally we have this DataFrame :

	login	password	platform	platform_id
0	joindiscordserver@gmail.com	https	fortnite	1
1	joindiscordserver@gmail.com	httpsfortnite@arcor.de:1fckoeln123	fortnite	1
2	fortnite@freenet.de	tiberius	fortnite	1

II.3.2 Processing email providers

With the simple data that we had with the platforms, the passwords and the email addresses we wanted to extract some more data to build our regression model. We thus created a variable for the email provider and also an id for it.

For that we followed these steps :

1. Extract the provider from the email address and add a corresponding column to the DataFrame (email_provider)
2. Count the number of occurrences and sort the providers id by order of appearance (1 is the most present provider)
3. Make a dictionary of the providers and their ids
4. Make a list of the contain the providers id (email_provider_id) corresponding to the login addresses
5. Add the column to the DataFrame

Finally we have this DataFrame :

	login	password	platform	platform_id	email_provider	email_provider_id
0	joindiscordserver@gmail.com	https	fortnite	1	gmail.com	2
1	joindiscordserver@gmail.com	httpsfortnite@arcor.de:1fckoeln123	fortnite	1	gmail.com	2
2	fortnite@freenet.de	tiberius	fortnite	1	freenet.de	4

II.3.3 Processing countries

Similarly we extracted the country with the extension (.com, .fr, .net, .de, .co.uk…) of the email provider and attributed an idea. The .com and .net extensions are considered international

For that we followed these steps :

1. Extract the extension from the provider and add a corresponding column to the DataFrame (country)
2. Count the number of occurrences and sort the extensions id by order of appearance (1 is the most present provider)
3. Make a dictionary of the extensions and their ids
4. Make a list of the contain the extensions id (country_id) corresponding to the login addresses
5. Add the column to the DataFrame

Finally we have this DataFrame :

login	password	platform	platform_id	email_provider	email_provider_id	country	country_id
joindiscordserver@gmail.com	https	fortnite	1	gmail.com	2	international	2
joindiscordserver@gmail.com	httpsfortnite@arcor.de:1fckoeln123	fortnite	1	gmail.com	2	international	2
fortnite@freenet.de	tiberius	fortnite	1	freenet.de	4	de	1

II.3.4 Adding password score and treating the different vulnerabilities

The next step was to compute the password score and to extract the vulnerabilities for each couple of data. The calculation process of the password strength will be discussed in II.4.

For that we followed these steps :

1. Install and import passwordmeter (a python library)
2. Create two empty lists (one for the score and one for the vulnerabilities)
3. Compute the score and vulnerabilities, line by line
4. Add the score column
5. Add the improvements column that contains the vulnerabilities, there are 6 types of vulnerabilities (cf. II.4.1)
6. Build a dictionary for the vulnerabilities and their id
7. Build binary variables for the 6 types of vulnerabilities and add the 6 columns

Finally we have this DataFrame :

	login	password	platform	platform_id	email_provider	email_provider_id	country
0	joindiscordserver@gmail.com	https	fortnite	1	gmail.com	2	international
1	joindiscordserver@gmail.com	httpsfortnite@arcor.de:1fckoeln123	fortnite	1	gmail.com	2	international
2	fortnite@freenet.de	tiberius	fortnite	1	freenet.de	4	de

country_id	score	improvements	length_suggestion	charmix_suggestion	casemix_suggestion	phrase_suggestion	notword_suggestion	variety_suggestion
2	0.199468	{'length': 'Increase the length of the passwor...	1	1	1	1	0	0
2	0.399384	{'casemix': 'Use a good mix of UPPER case and ...	0	0	1	1	0	0
1	0.111294	{'charmix': 'Use a good mix of numbers, letter...	0	1	1	1	1	0

II.3.5 Discrete score and time to crack

To extend more the data we have and have other possibilities of regressions we added a discrete score (possible values : 0,1,2,3,4) and a simulated necessary time to crack from another python library called zxcvbn.

For that we followed these steps :

1. Install and import zxcvbn (a python library)
2. Create four empty lists (score, time to crack, suggestions to strengthen the passwords, number of guesses to crack)
3. Compute the score, time to crack, number of guesses and suggestions, line by line
4. Add the four corresponding columns

Finally we have this DataFrame (here only the added columns) :

	score1_4	time_to_crack	nb_guesses_to_crack	suggestion
0	0	2.2E-9	22	[Add another word or two. Uncommon words are b...
4	4	3.0000000000000001E+19	3.0000000000000000010000000000E+29	[]
1	1	2.817E-7	2817	[Add another word or two. Uncommon words are b...

II.3.6 Exporting to csv

Finally, to make the data clear and at a unique place we exported it in a csv file. The database is thus exploitable for regressions and statistical study.

II.4 - The password strength

II.4.1 The different vulnerabilities

In the model we chose to compute the password strength (as a score between 0 and 1) we consider 6 vulnerabilities :

1. **length**: Increase the length of the password, because the necessary time to crack a password is expenentially dependent on its length

   Example *: a 3 character password only with small case letters represent* $26^3 = 17\ 576$ *possibilities while adding 1 character multiplies by 26 the number of possibilities :* $26^4 = 456\ 976$ *possibilities (this can apply to any char set, lower case, higher case, numbers, symbols…)*

2. **charmix**: Use a good mix of numbers, letters, and symbols

   i.e. *here we want to maximize the number of possible symbols in our char set (ie not only lower case letters that are only 26). For that if we consider lower case letters (26), numbers (10), symbols (at least 29) it makes it really complex to crack*

   Example *: a 3 character password with only lower case letters :* $26^3 = 17\ 576$ *possibilities, if also with higher case, numbers and symbols :* $65^3 = 274\ 625$ *possibilities, it is incredibly harder to crack*

3. **casemix**: Use a good mix of UPPER case and lower case letters
   i.e. *we also need higher case letters and often in passwords to make it more complex*
   Example *: similarly with 3 characters and no higher case :* $65^3 = 274\ 625$ *possibilities, with higher case :* $,91^3 = 753\ 571$ *possibilities, it is incredibly harder to crack*

4. **phrase**: Passphrases (e.g. an obfuscated sentence) are better than passwords
   Example *: I love to play 2 hours and send emails at 12:32 to cc@password.com becomes ILoveToPlay2HoursAndSendEmailsAt12:32Tocc@password.com and this password is easy to remember but extremely hard to crack because it follows the 3 previous rules*

5. **notword**: Avoid using one of the ten thousand most common passwords
   Example *: 123456 is definitely a bad password*

6. **variety**: Minimize character duplicates and repetitions
   Example *: Leo1Leo1Leo1Leo1 is a bad password even if it respects rules 1, 2 and 3 because it is only 4 characters long in reality (Leo1)*

II.4.2 Computation of the password strength

To compute the password strength (a float is in the range 0 to 1, inclusive, where 0 is extremely weak and 1 is extremely strong) the library passwordmeter uses several variables.

These variables are weights calculated with the passwords which are weighted and averaged (not all factors have the same weight, and so not the same importance.

These variables corresponds to the 6 vulnerabilities :

- **length**, there is a threshold to have a 1 over 1 score in length (for example if the length is superior to 13 you can have 1, if it is less the score is $\frac{length}{threshold}$)
- **charmix**: there are 3 things to have : numbers, letters and symbols
  - The calculation of this variable is the following : the mean between the number of things you have and the proportion of each you have (to have 1 score you must have the 3 things and in equal proportions)

$$charmix = \frac{(hasletters) + (hasnumbers) + (hassymbols)}{2 \times 3} + \frac{(maxdeviation - (1/3 - lettersproportion) - (1/3 - numbersproportion) - (1/3 - symbolsproportion))}{2 \times maxdeviation}$$

  - Example : for *Leo1@* we have the 3 (letters, symbols, numbers) and we have 20% of symbols, 20% of numbers and 60% of letters so a score of (2/3-(1/3-0,20)-(1/3-0,20)-(1/3-0,60)/(2/3))=0,20 (1 minus the sum of deviations to 33,3% for each thing, divided by the maximum deviation 0,667) : ie we do the mean between 1 and 0,20 => 0,6 is the charmix score
- **casemix**: it works similarly to charmix with 2 things to have : lower and higher case letters and then the proposition of each (here max deviation is 0,5 not 2/3)

$$casemix = \frac{(hashigher) + (haslower)}{2 \times 2} + \frac{(maxdeviation - (1/2 - higherproportion) - (1/2 - lowerproportion))}{2 \times maxdeviation}$$

passphrase : thanks to the structure of the password it detects if the password is a phrase (if it is its weights is 1, if not it's 0)
notword : if it is one of the 10.000 most know passwords its weight is 0 and if not the weight is 1
variety : deletes the repetitions and calculates a score with the five other variables to make a variety score.

I haven't been able to have the weights and thresholds of each variable in passwordmeter's code but all the weights are between 0 and 1 and sum to 1 such that the score is between 0 and 1.

II.5 - Final description of the database

Finally we have a database that contains 311536 lines with the following columns :

- **login** : email address
- **password** : the password
- **platform** : the name of the platform corresponding to the account
- **platform_id** : the id corresponding to the platform (not ordered)
- **email_provider** : the email provider extracted from the login
- **email_provider_id** : the email provider corresponding id (ordered by decreasing apparition count: 1 is the most present)
- **country** : extension of the email address
- **country_id** : extension corresponding id (ordered by decreasing apparition count: 1 is the most present)
- **score** : password strength score processed with passwordmeter
- **improvements** : password strength improvements processed with passwordmeter
- **length_suggestion** : binary variable for length vulnerability
- **charmix_suggestion** : binary variable for charmix vulnerability
- **casemix_suggestion** : binary variable for casemix vulnerability
- **phrase_suggestion** : binary variable for phrase vulnerability
- **notword_suggestion** : binary variable for notword vulnerability
- **variety_suggestion** : binary variable for variety vulnerability
- **score1_4** : discrete score processed with zxcvbn
- **time_to_crack** : estimated time to crack the password processed with zxcvbn
- **nb_guesses_to_crack** : estimated processed with zxcvbn
- **suggestion** : other suggestions to improve the password strength processed with zxcvbn

It gives the following DataFrame :

	login	password	platform	platform_id	email_provider	email_provider_id	country	country_id	score
0	joindiscordserver@gmail.com	https	fortnite	1	gmail.com	2	international	2	0.199468
1	joindiscordserver@gmail.com	httpsfortnite@arcor.de:1fckoeln123	fortnite	1	gmail.com	2	international	2	0.399384
2	fortnite@freenet.de	tiberius	fortnite	1	freenet.de	4	de	1	0.111294

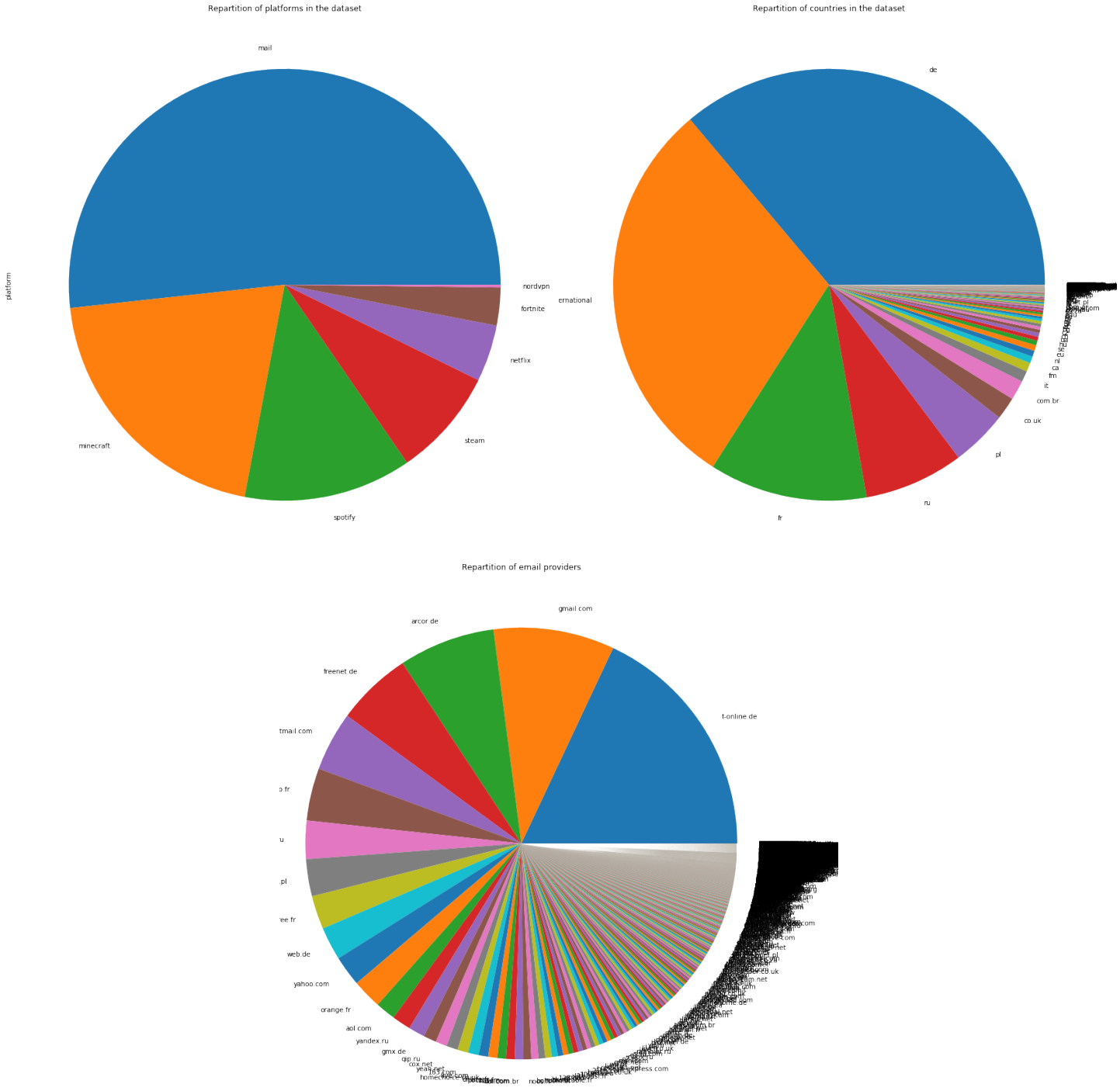
improvements	length_suggestion	charmix_suggestion	casemix_suggestion	phrase_suggestion	notword_suggestion	variety_suggestion	score1_4	time_to_crack	nb_guesses_to_crack	suggestion
{'length': 'Increase the length of the passwor...	1	1	1	1	0	0	0	2.2E-9	22	[Add another word or two. Uncommon words are b...
{'casemix': 'Use a good mix of UPPER case and ...	0	0	1	1	0	0	4	3.0000000000000001E+19	3.0000000000000001000000000E+29	[]
{'charmix': 'Use a good mix of numbers, letter...	0	1	1	1	1	0	1	2.817E-7	2817	[Add another word or two. Uncommon words are b...

III - Descriptive analysis data

III.1 Repartitions of platforms, countries and email providers

The first thing we want to show about our dataset is some **repartitions over some categories of data**. For that we computed the data in a python notebook available in the end of the report.

The first (top left) pie chart shows the repartition between the seven different platforms in the dataset. The second (top right) depicts the repartition of countries in the dataset. The third one (bottom) shows the repartition of email providers in the dataset.



Here we can notice that there are **a lot of mail accounts** (about half of the dataset). The most represented nationalities are **German** (de) and **International**. And finally there are really a lot of email providers, the most represented are t-online.de, gmail.com and arcor.de.

It is really important to know these repartitions to study the data, to **avoid missing a conclusion because of some asymmetries** in the categories.

III.2 Relevant informations on the data

	password
123456	758
123456789	244
schalke04	120
12345678	115
password	98
111111	94
hallo123	92
taishan2011	90
1q2w3e4r	88
internet	85
qwerty	77
12345	73
123123	71
zaq12wsx	69
000000	69
azerty	64
hassan123	63
passwort	63
1234567890	61
killer	59
daniel	57
benjamin	57
alexander	56
1q2w3e	54
doudou	52

While studying the content of our database some thing seemed important to show.

First of all, at the right, the most common passwords that you can find on the left. Here you have the top 25 passwords and their number of occurrences in the dataset (so over 311 536 lines).

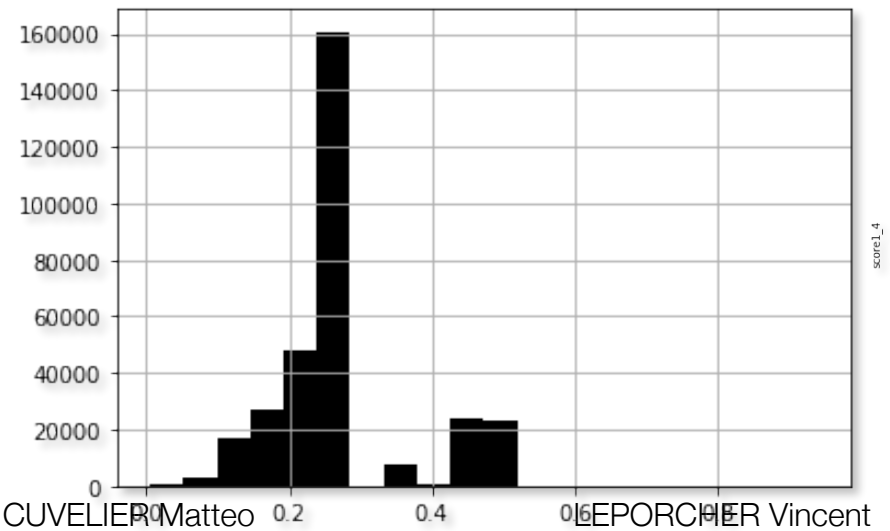
We can notice that the passwords that are a series of numbers are really common (123456, 123456789, 12345678, 12345,123123,1234567890…) and also common words such as password or its translation to German (passwort). In the pole position we also find the word internet, qwerty, azerty… And also some simple and common first names (Benjamin, Alexander, Daniel…). **All these passwords are extremely weak and are totally unable to protect any of your data.** This is the alarming observation we made thanks to the most common passwords.

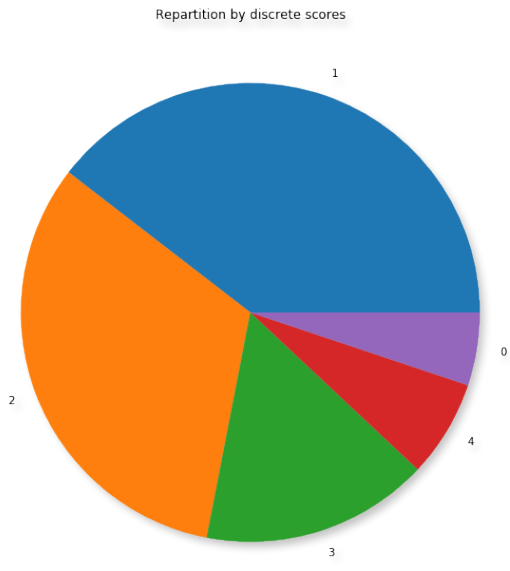
Then let's look at the scores. At the bottom of the page you will find two graphs. On the left an histogram of the passwords strengths and at the right a pie chart of the discrete score (see II.3.5).

On the histogram you can notice a peak around 0.3 which is really alarming. In an ideal case we should find a peak around 0.9 or 0.8, this would depict a good safety situation but in reality we notice here again the extreme weakness of the password database that we have.

The pie chart tells almost the same, even if there are very few 0 score, 1 and 2 scores are really highly represented (about 70% of 0, or 2 scores). Here again, in an ideal case it would be the 3 or 4 password scores that should have 70% of the pie…

Considering all this we wanted to know more about the different vulnerabilities of these passwords and to correlate them with some other variables such as the location (with the country column), the email provider or following the platform.





III.3 Basic statistics on the data

Here are some basic statistics about our database. This displays some qualitative facts about non digital columns and basic statistics for digital variables.

Here are the conclusions we can make from these information :

- **password** : there are only 175 712 unique passwords in more than 300 000 lines
- **platform** : the dominant platform here is mail with 161 133 accounts
- **email_provider** : the main email provider is t-online.de with 112 552 accounts
- **score** : the mean score is really low (0,27), 75% of passwords are below 0,265 in term of score, its standard deviation is of 0.10 so the dispersion is really low, almost all passwords seem to have a score around the mean
- **score1_4** : 75% have 2 or below, same observation as score
- **time_to_crack** : 75% of passwords can be cracked in less than a second
- **length_suggestion** : 23,7% of passwords are not long enough
- **charmix_suggestion** : 99,6% of passwords aren't mixing enough letters, numbers and symbols
- **casemix_suggestion** : 81,9% of passwords aren't mixing enough higher and lower case
- **phrase_suggestion** : 100% of passwords aren't passphrases
- **notword_suggestion** : 6,9% of passwords are in the 10 000 most used passwords (this is a good thing)
- **variety_suggestion** : 13,7% passwords need to avoid repetition
- **nombre_gesses_to_crack** : 75% of passwords can be cracked with 10^8 guesses or less which is not high at all

	password	platform	email_provider	country	score	score1_4	time_to_crack	length_suggestion	charmix_suggestion
count	311208	311536	311536	311536	311536.000000	311536.000000	3.115360e+05	311536.000000	311536.000000
unique	175712	7	4676	408	NaN	NaN	NaN	NaN	NaN
top	123456	mail	t-online.de	de	NaN	NaN	NaN	NaN	NaN
freq	758	161133	56101	112552	NaN	NaN	NaN	NaN	NaN
mean	NaN	NaN	NaN	NaN	0.269993	1.799484	1.330457e+47	0.236814	0.996235
std	NaN	NaN	NaN	NaN	0.100198	0.997752	5.907667e+49	0.425128	0.061246
min	NaN	NaN	NaN	NaN	0.005495	0.000000	0.000000e+00	0.000000	0.000000
25%	NaN	NaN	NaN	NaN	0.209912	1.000000	5.730000e-06	0.000000	1.000000
50%	NaN	NaN	NaN	NaN	0.262107	2.000000	1.990000e-04	0.000000	1.000000
75%	NaN	NaN	NaN	NaN	0.265260	2.000000	1.000000e-02	0.000000	1.000000
max	NaN	NaN	NaN	NaN	0.939885	4.000000	3.140824e+52	1.000000	1.000000

casemix_suggestion	phrase_suggestion	notword_suggestion	variety_suggestion	score1_4	time_to_crack	nb_guesses_to_crack
311536.000000	311536.0	311536.000000	311536.000000	311536.000000	3.115360e+05	3.115360e+05
NaN	NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN	NaN
0.818705	1.0	0.069693	0.136713	1.799484	1.330457e+47	1.330457e+57
0.385263	0.0	0.254630	0.343545	0.997752	5.907667e+49	5.907667e+59
0.000000	1.0	0.000000	0.000000	0.000000	0.000000e+00	0.000000e+00
1.000000	1.0	0.000000	0.000000	1.000000	5.730000e-06	5.730000e+04
1.000000	1.0	0.000000	0.000000	2.000000	1.990000e-04	1.990000e+06
1.000000	1.0	0.000000	0.000000	2.000000	1.000000e-02	1.000000e+08
1.000000	1.0	1.000000	1.000000	4.000000	3.140824e+52	3.140824e+62

III.4 Statistics following certain categories

Here we will present statistics of score in function of the regrouping categories that can be chosen (platform, email provider, country).

III.4.1 By platform

	score							
	count	mean	std	min	25%	50%	75%	max
platform								
fortnite	8828.0	0.272347	0.096281	0.065708	0.210793	0.262185	0.265618	0.913274
mail	161133.0	0.267033	0.097240	0.065708	0.209571	0.261949	0.265162	0.939424
minecraft	63243.0	0.261118	0.093315	0.065708	0.209132	0.261493	0.264820	0.934013
netflix	13226.0	0.282421	0.116229	0.065708	0.209860	0.262272	0.265723	0.927352
nordvpn	622.0	0.117149	0.139130	0.005495	0.005495	0.005495	0.178107	0.478778
spotify	39153.0	0.291891	0.109890	0.065708	0.254027	0.263171	0.266391	0.939885
steam	25331.0	0.273577	0.102476	0.065708	0.210327	0.262185	0.265618	0.932673

The classification by platform reveals that nordvpn is weaker in term of password score than the other platforms. The other results are quite close to each other.

III.4.2 By email provider

	score							
	count	mean	std	min	25%	50%	75%	max
email_provider								
0000.ru	1.0	0.212345	NaN	0.212345	0.212345	0.212345	0.212345	0.212345
01019freenet.de	4.0	0.238459	0.031112	0.211515	0.211515	0.238459	0.265402	0.265402
010fuss.com	1.0	0.256962	NaN	0.256962	0.256962	0.256962	0.256962	0.256962
02.pl	1.0	0.202518	NaN	0.202518	0.202518	0.202518	0.202518	0.202518
0bellsouth.net	1.0	0.205124	NaN	0.205124	0.205124	0.205124	0.205124	0.205124
0msn.com	1.0	0.208653	NaN	0.208653	0.208653	0.208653	0.208653	0.208653
101berlin.com	4.0	0.211280	0.000000	0.211280	0.211280	0.211280	0.211280	0.211280
123.com	3.0	0.195581	0.054124	0.164332	0.164332	0.164332	0.211205	0.258077
126.com	1394.0	0.216675	0.075727	0.065708	0.163264	0.210474	0.261860	0.769792
1278.se	3.0	0.161860	0.000000	0.161860	0.161860	0.161860	0.161860	0.161860
1337.no	11.0	0.290743	0.091002	0.209860	0.259804	0.260014	0.265423	0.477242
163.com	2522.0	0.219258	0.069636	0.065708	0.163729	0.211272	0.263004	0.823074
188.com	1.0	0.261858	NaN	0.261858	0.261858	0.261858	0.261858	0.261858
189.cn	46.0	0.227712	0.060565	0.134950	0.164180	0.261014	0.264553	0.468007
1999.ru	1.0	0.260237	NaN	0.260237	0.260237	0.260237	0.260237	0.260237
1ate3.com	2.0	0.209815	0.000000	0.209815	0.209815	0.209815	0.209815	0.209815
1nsk.ru	1.0	0.202082	NaN	0.202082	0.202082	0.202082	0.202082	0.202082
1thecity.biz	1.0	0.209571	NaN	0.209571	0.209571	0.209571	0.209571	0.209571
2.com	1.0	0.109252	NaN	0.109252	0.109252	0.109252	0.109252	0.109252
20000.ru	1.0	0.090657	NaN	0.090657	0.090657	0.090657	0.090657	0.090657
2000bk.ru	1.0	0.260858	NaN	0.260858	0.260858	0.260858	0.260858	0.260858
2010live.com	1.0	0.210474	NaN	0.210474	0.210474	0.210474	0.210474	0.210474
20hours.it	1.0	0.206628	NaN	0.206628	0.206628	0.206628	0.206628	0.206628
211.ru	1.0	0.267375	NaN	0.267375	0.267375	0.267375	0.267375	0.267375
21cn.com	1008.0	0.185985	0.046705	0.068527	0.162787	0.164844	0.208514	0.482222

Concerning the classification by email provider, we see that the mean scores are really different and so are the standard deviations, this is interesting to do a regression between the different providers and the score.

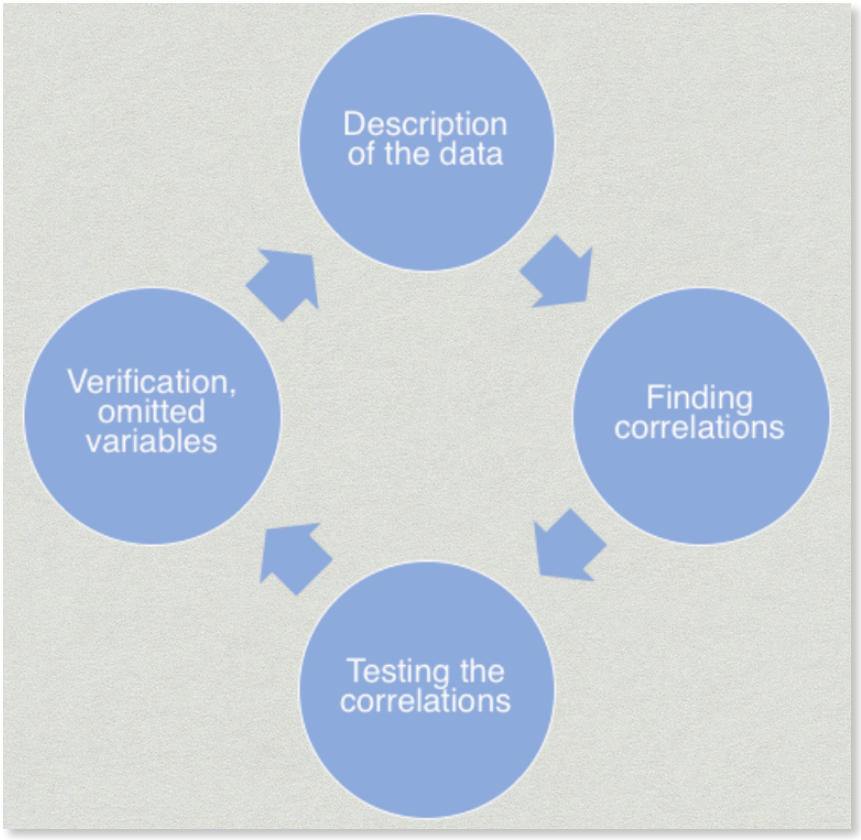
III.4.3 By country

	score							
	count	mean	std	min	25%	50%	75%	max
country								
CA	3.0	0.468270	0.006617	0.464449	0.464449	0.464449	0.470180	0.475910
CO.UK	1.0	0.265710	NaN	0.265710	0.265710	0.265710	0.265710	0.265710
COM	121.0	0.235691	0.111952	0.088433	0.163876	0.212501	0.262571	0.844989
COM.BR	1.0	0.255412	NaN	0.255412	0.255412	0.255412	0.255412	0.255412
COM.PL	1.0	0.264396	NaN	0.264396	0.264396	0.264396	0.264396	0.264396
Com	4.0	0.413440	0.237301	0.192042	0.247031	0.370803	0.537211	0.720110
DK	2.0	0.267321	0.000000	0.267321	0.267321	0.267321	0.267321	0.267321
EDU	6.0	0.200557	0.000000	0.200557	0.200557	0.200557	0.200557	0.200557
ES	2.0	0.166059	0.000000	0.166059	0.166059	0.166059	0.166059	0.166059
FR	10.0	0.244201	0.041579	0.165381	0.261491	0.262982	0.264751	0.266402
Fr	2.0	0.212704	0.071903	0.161860	0.187282	0.212704	0.238125	0.263547
IT	3.0	0.209167	0.046801	0.162787	0.185561	0.208335	0.232357	0.256379
NET	9.0	0.273071	0.154603	0.105915	0.167325	0.205124	0.465624	0.470854
Net	2.0	0.154875	0.067377	0.107232	0.131054	0.154875	0.178696	0.202518
PL	4.0	0.198538	0.051804	0.158363	0.158363	0.184419	0.224594	0.266952
RR.COM	1.0	0.212345	NaN	0.212345	0.212345	0.212345	0.212345	0.212345
RU	52.0	0.202975	0.073655	0.084233	0.099129	0.255693	0.264421	0.268511
RY	2.0	0.184393	0.026669	0.165535	0.174964	0.184393	0.193823	0.203252
Ru	1.0	0.258006	NaN	0.258006	0.258006	0.258006	0.258006	0.258006
Se	2.0	0.357206	0.134380	0.262185	0.309696	0.357206	0.404717	0.452227
UR	1.0	0.199292	NaN	0.199292	0.199292	0.199292	0.199292	0.199292
a2000.nl	1.0	0.264751	NaN	0.264751	0.264751	0.264751	0.264751	0.264751
abb.com	1.0	0.476418	NaN	0.476418	0.476418	0.476418	0.476418	0.476418
ac.in	1.0	0.109252	NaN	0.109252	0.109252	0.109252	0.109252	0.109252
ac.jp	1.0	0.471599	NaN	0.471599	0.471599	0.471599	0.471599	0.471599

Similarly, concerning the classification by country, we have really different mean scores and also standard deviations, this is interesting to do a regression between the different countries and the score.

IV - Empirical Strategy

As a general strategy, we will rely on four steps.



The first step of our empirical strategy is to **describe the data**. A thorough description has been made in the previous part.

Then, our strategy is to study three correlations:
- A correlation of the password strength (score) and the platform.
- A correlation of the score and the email provider.
- A correlation of the score and economic dependency, such as the GDP and the HDI (human development indice).

To have these correlations, we will plot the score in function of the dependent variable, when applicable, in order to have a first overview of the general behavior of the dependence of both variables. This will enable us to raise a hypothesis of potential correlations.

After studying these correlations, we will verify them by plotting the predicted score in function to the score. If there is a correlation, then the slope of the function should be close to 1. We can also use the stats model Python library to have the exact coefficients of the regression and the value of R-squared, indicating wether our correlation is truthful or not.

Last, we will verify if the omitted variables, such as the requirements of some platforms for password choice. This last verification would prompt us to review or not our original description. In particular, we might reconsider the way we compute the score given the platform's requirements.

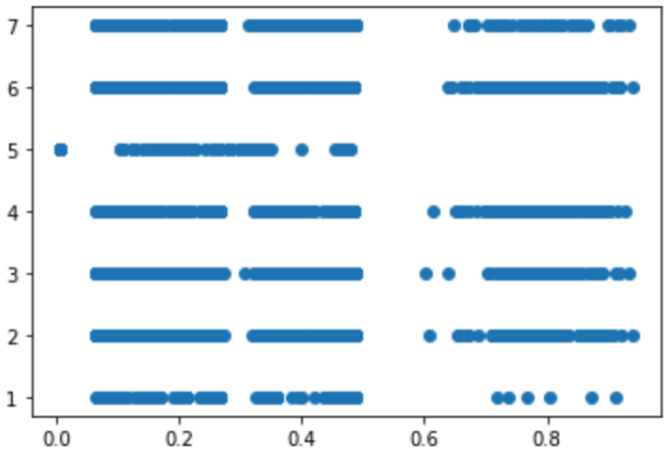
V - Results and tests

V.1 Platform

V.1.1. General overview

A first general indication of the dependence between the score and the platform is shown on the graph bellow :

1: Fortnite
2: Mail
3: Minecraft
4: Netflix
5: NordVPN
6: Spotify
7: Steam



It is striking to see that two platforms have particularly low-score passwords, Fortnite and NordVPN.
For Fortnite, we can say that it is because most of the users are young and therefore not sensitised to the reasons why they should use a high-score password.
For NordVPN, the need for safety is not very strong, since the service doesn't provide sensitive information.
Hence we can see a dependency between the choice of the platform and the choice of the password.
We can hypothesise that there is a correlation between the score and the platform.

V.1.2. Regression

We compute the OLS regression and we get the following results:

OLS Regression Results

Dep. Variable:	platform_id	**R-squared (uncentered):**	0.690
Model:	OLS	**Adj. R-squared (uncentered):**	0.690
Method:	Least Squares	**F-statistic:**	6.936e+05
Date:	Sun, 29 Mar 2020	**Prob (F-statistic):**	0.00
Time:	21:15:19	**Log-Likelihood:**	-6.6105e+05
No. Observations:	311536	**AIC:**	1.322e+06
Df Residuals:	311535	**BIC:**	1.322e+06
Df Model:	1		
Covariance Type:	nonrobust		

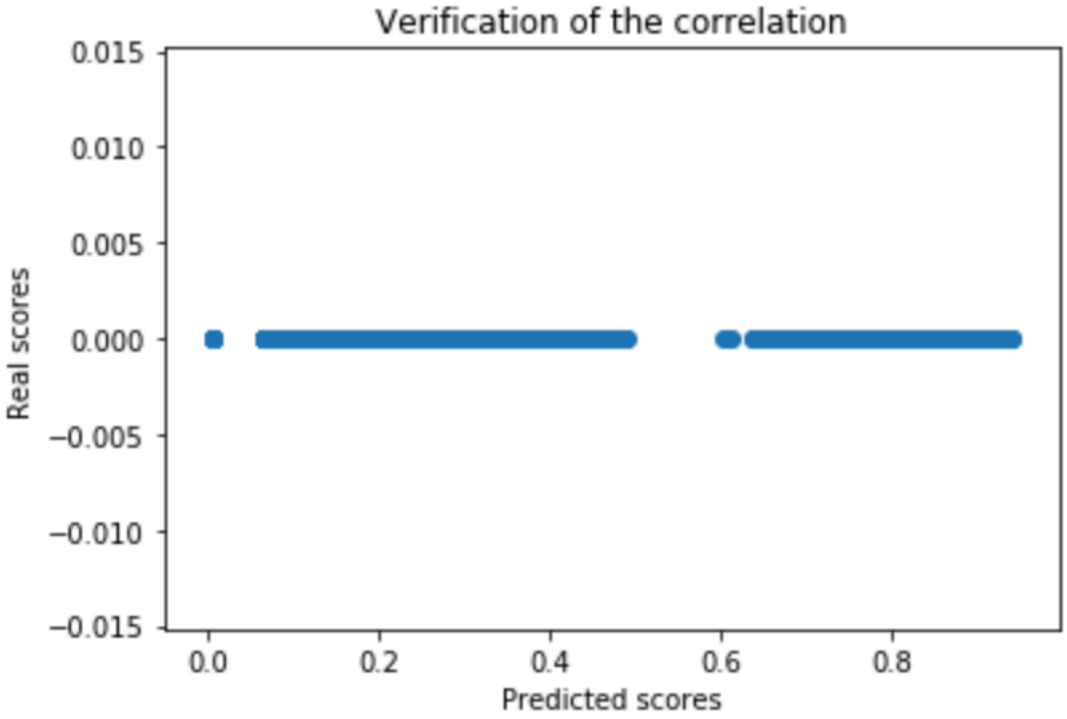
	coef	std err	t	P>|t|	[0.025	0.975]
score	10.4649	0.013	832.849	0.000	10.440	10.490

Omnibus:	17266.366	**Durbin-Watson:**	0.442
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	20306.890
Skew:	0.624	**Prob(JB):**	0.00
Kurtosis:	3.090	**Cond. No.**	1.00

We see that the correlation is not satisfactory. The R-squared is only equal to 0.690 whereas it should be close to 0.9 to be relevant, so our **hypothesis is invalidated**.

V.1.3. Testing the regression and results

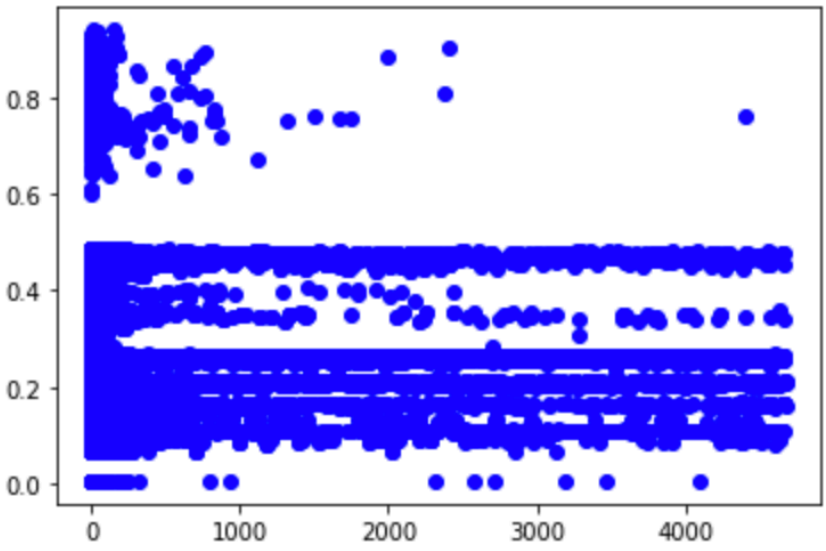
We compute the prediction score in function of the score.



There is obviously no correlation between the predicted score and the original score, since the slope of the curve is zero. There would have been a correlation if this slope was close to 1.
Hence, the original hypothesis that there is a correlation is invalidated.
Furthermore, it means that **the consideration of the platform's requirements as an omitted variable is not relevant**: since the score doesn't depend on the platform, the password-creation requirements do not play any role.

V.2 Email provider

V.2.1. General overview



We plot the score of the password in function of the email provider id.
We don't see any particular relationship on this graph.

Hence, we expect no correlation between the score and the email provider.

V.2.2. Regression

Afterwards, for more clarity, we focus on the **fourteen most represented email providers and regroup the other little providers in one binary variable** in the dataset. We compute the OLS estimator and we get the following results:

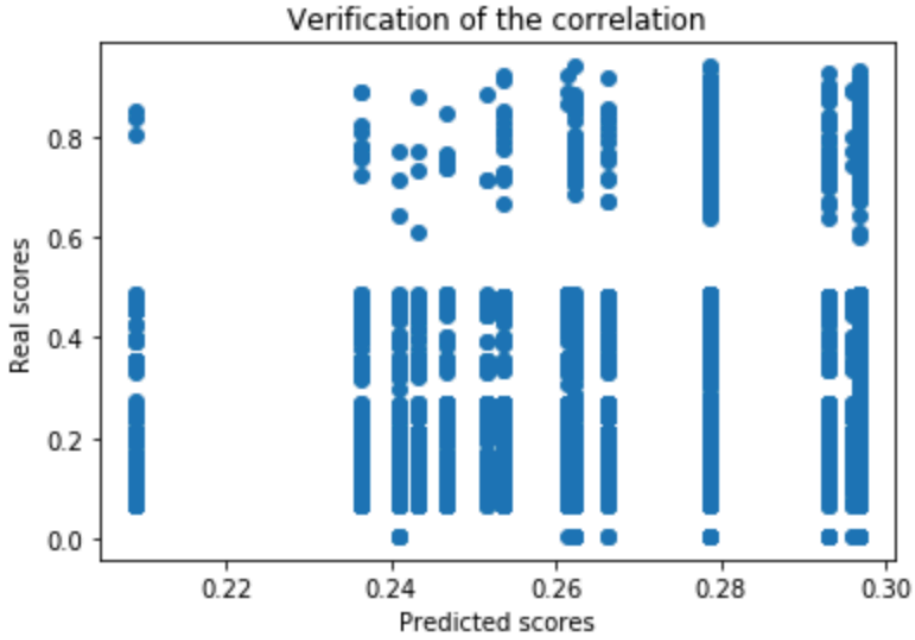
Dep. Variable:	score	**R-squared:**	0.039
Model:	OLS	**Adj. R-squared:**	0.039
Method:	Least Squares	**F-statistic:**	977.2
Date:	Sun, 29 Mar 2020	**Prob (F-statistic):**	0.00
Time:	21:26:28	**Log-Likelihood:**	2.8090e+05
No. Observations:	311536	**AIC:**	-5.618e+05
Df Residuals:	311522	**BIC:**	-5.616e+05
Df Model:	13		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0	0	nan	nan	0	0
x1	0.0180	0.001	28.355	0.000	0.017	0.019
x2	-0.0425	0.001	-60.760	0.000	-0.044	-0.041
x3	-0.0355	0.001	-45.582	0.000	-0.037	-0.034
x4	-0.0165	0.001	-18.994	0.000	-0.018	-0.015
x5	-0.0250	0.001	-27.124	0.000	-0.027	-0.023
x6	-0.0173	0.001	-16.289	0.000	-0.019	-0.015
x7	-0.0273	0.001	-25.160	0.000	-0.029	-0.025
x8	-0.0697	0.001	-61.237	0.000	-0.072	-0.067
x9	-0.0124	0.001	-10.895	0.000	-0.015	-0.010
x10	0.0143	0.001	12.458	0.000	0.012	0.016
x11	0.0172	0.001	14.257	0.000	0.015	0.020
x12	-0.0377	0.001	-25.880	0.000	-0.041	-0.035
x13	-0.0321	0.002	-21.324	0.000	-0.035	-0.029
x14	0.2788	0.000	1136.011	0.000	0.278	0.279

Omnibus:	47870.574	**Durbin-Watson:**	1.688
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	80386.574
Skew:	1.029	**Prob(JB):**	0.00
Kurtosis:	4.398	**Cond. No.**	inf

R-squared is close to zero, so this regression is not all satisfactory. The original hypothesis that there is no correlation between the score and the email provider is thus confirmed.

V.2.3. Testing the regression and results

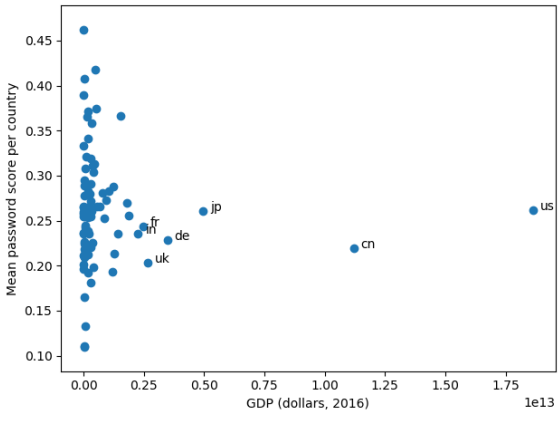


We can see that none of the slopes are close to 1. The results are displayed in the following table:

Hence, we verified the fact that there is no correlation between the score and the email provider and that **the hypothesis is confirmed.**

V.3 Gross Domestic Product (GDP)

V.3.1 General overview



In this section, we investigate the influence of the gross domestic product (GDP) on the password score. GDP is a benchmark economic indicator reflecting the total value of goods and services produced during a year. The most powerful countries such as China or the United States have the highest GDP. It reflects the economic development of a country, but omits all human considerations (education, life expectancy). It is not a sociological indicator, but a purely economic one.

We are using a dataset provided by the United Nations, with GNP values ranging from 1960 to 2016 for each country. We conducted the study with the most recent values, which are those for 2016. To link the GDP of a country with its Top Level Domain (TLD) extension (.fr, .de ...), the standard alpha-3 code is used. Two datasets are linked: the GDP according to the alpha-3 code of the country and the TLD extension according to the alpha-3 code.

Qualitatively, **we do not observe a clear link between GDP and the password score**.

V.3.2 Regression

Using a standard OLS regression, we obtain:

```
score = 0.262 + -1.25e-15 * GDP
```

The regression coefficient is very close to zero, but the GDP value are of the order of 1e13, so indeed it is not so close relatively to GDP values. However, it still doesn't seem significant. The established regression of the score has a **very low determination coefficient**, as expected: `R-squared = -0.12`.

V.3.3 Results

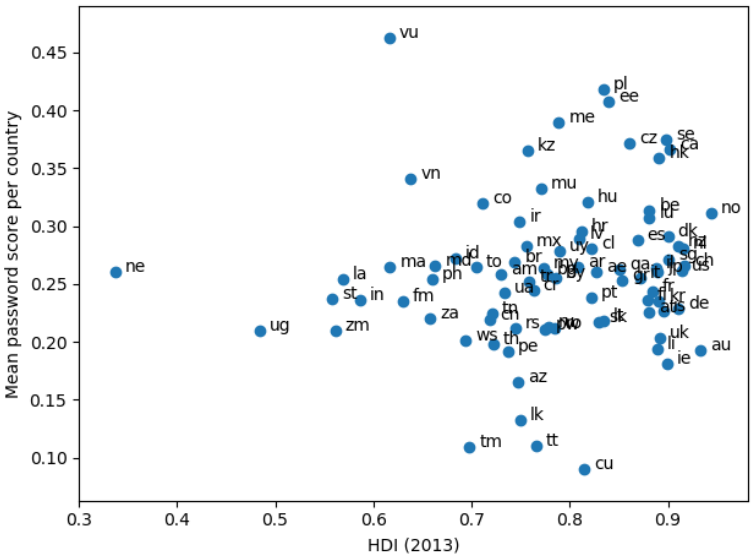
Having such a low determination coefficient, we can conclude that there's **no relationship between the password score and the country GDP**. However, it is interesting to notice that China has a particularly low score for its GDP, compared to the US (higher GDP) and Japan (lower GDP). Knowing that China is a less developed country than these two countries, it would be interesting to look at what happens with human development indicators. This is what we will do in the next section.

V.4 Human Development Index (HDI)

V.4.1 General overview

In this section, we investigate the influence of the **humain development index (HDI)** on the password score. The HDI is a composite indicator combining life expectancy, education and income. It reflects how well people are sheltered, fed, and healty. At first glance, one might think that a higher HDI means a higher general level of education, and thus an **awareness of the importance of the strength** of a stronger password.

We are using a dataset from the United Nations giving the HDI per country as of 2013.



V.4.2 Regression

Using a standard OLS regression, we obtain:

```
score = 0.208 + 0.0657 * HDI
```

Again, the regression coefficient do not seem significant. The **determination coefficient is insignificant**, and we can conclude without doubt that the linear model do not fit the data.

V.4.3 Results

The **HDI does not seem to influence the password score**, contrary to what one might have thought at first. However, we do not take into account the number of passwords from each country at all. Each country is considered the equal of another. For example, Vanuatu (.vu), a very small country, is very far from the point cloud. Thus, it reduces the coefficient of determination, even though it has a very low relative weight.

Conclusion

We investigated correlations between the time it takes to crack a password and various individual factors. First, we decided to quantify the time needed to crack a password, which depends on the machine and the method, by a numerical indicator that is the strength of the password (or score). Next, we gathered sensitive data from several well-known and diverse websites (Netflix, Spotify, Steam ...).
Our first findings were frightening: the vast majority of passwords are very insecure, and easy to crack. Three-quarters of all passwords can be cracked in less than a second. But this varies a little from platform to platform. For example, the score is generally higher on non-free platforms like Spotify or Netflix than on mail platforms, which are generally free. However, in the category of mail providers, there are great disparities.

Then, we've carried out statistical analysis. We have shown that there is no link between the password score and the service provider in the general case. We've found two exceptions (Spotify and Minecraft). Regarding the email providers, the analysis conducted proved that, unlike what had been intuited, they aren't correlated in any way with the password score. Finally, we've examined the gross domestic product (GDP) and humain development index (HDI). We've concluded that there was no significant link between the password strengh and neither GDP nor HDI.

This work demonstrate that there is no clear statistical link with the chosen indicators, and that password choice is an individual choise, which cannot be generalized across a user community or country. Further research should focus on people's identities: age, gender, level of education or hobbies.

Code

Code 1 - Processing data

In [1]:
```
import pandas as pd
import numpy as np
```

II.3.1 Exploiting text files

Transform password text files

In [2]:
```
def texttolist(file, expiration=False):
    f = open(file, "r")
    file = []
    for elt in f.readlines():
        [a,b]=elt.split(':', 1)
        if expiration :
            file.append([a,b[:-41]])
        else :
            file.append([a,b[:-1]])
    data = pd.DataFrame(file)
    data.columns = ['login','password']
    return data
```

In [3]:
```
fortnite = texttolist("./Fortnite 8k.txt")
fortnite['platform'] = 'fortnite'
fortnite['platform_id'] = 0+1

mail = texttolist("./Mail 161k.txt")
mail['platform'] = 'mail'
mail['platform_id'] = 1+1

minecraft = texttolist("./Minecraft 60k.txt")
minecraft['platform'] = 'minecraft'
minecraft['platform_id'] = 2+1

netflix = texttolist("./Netflix 12k.txt")
netflix['platform'] = 'netflix'
netflix['platform_id'] = 3+1

nordvpn = texttolist("./Nordvpn 633.txt", True)
nordvpn['platform'] = 'nordvpn'
nordvpn['platform_id'] = 4+1

spotify = texttolist("./Spotify 38k.txt")
spotify['platform'] = 'spotify'
spotify['platform_id'] = 5+1

steam = texttolist("./Steam 25k.txt")
steam['platform'] = 'steam'
steam['platform_id'] = 6+1
```

In [4]:
```
database = pd.concat([fortnite, mail, minecraft, netflix, nordvpn, spotify, steam], ignore_index=True)
nb_of_lines = database.shape[0]
database.head()
```

Out[4]:

	login	password	platform	platform_id
0	joindiscordserver@gmail.com	https	fortnite	1
1	joindiscordserver@gmail.com	httpsfortnite@arcor.de:1fckoeln123	fortnite	1
2	fortnite@freenet.de	tiberius	fortnite	1
3	fortnite@aliceadsl.fr	navarro	fortnite	1
4	fortnite@virgilio.it	211089	fortnite	1

II.3.2 Processing email providers

Extracting information and adding ids

In [5]:
```
database['email_provider']=[(database['login'][i]).split('@', 1)[1] for i in range(nb_of_lines)]
providers, count = np.unique(database['email_provider'].values, return_counts=True)

indexes = np.argsort(count, )
providers_sort = []

for ind in indexes:
    providers_sort.append(providers[ind])
providers_sort=np.array(providers_sort[::-1])
providers_sort
```

Out[5]:
```
array(['t-online.de', 'gmail.com', 'arcor.de', ..., 'mail.rd',
       'mail.qango.com', '0000.ru'], dtype='<U33')
```

In [6]:
```
nb_providers = (np.unique(database['email_provider'].values).shape)[0]

providers_dict = { providers_sort[i] : i+1 for i in range(nb_providers) }
email_provider_id = []
for i in range (nb_of_lines):
    email_provider_id.append(providers_dict[database['email_provider'][i]])
database['email_provider_id'] = email_provider_id
database.head()
```

Out[6]:

	login	password	platform	platform_id	email_provider	email_provider_id
0	joindiscordserver@gmail.com	https	fortnite	1	gmail.com	2
1	joindiscordserver@gmail.com	httpsfortnite@arcor.de:1fckoeln123	fortnite	1	gmail.com	2
2	fortnite@freenet.de	tiberius	fortnite	1	freenet.de	4
3	fortnite@aliceadsl.fr	navarro	fortnite	1	aliceadsl.fr	32
4	fortnite@virgilio.it	211089	fortnite	1	virgilio.it	139

II.3.3 Processing countries

In [7]:
```
ctry=[]
for i in range(nb_of_lines):
    if database['email_provider'][i].split('.', 1)[1]=='com':
        ctry.append('international')
    elif database['email_provider'][i].split('.', 1)[1]=='net':
        ctry.append('international')
    else :
        ctry.append(database['email_provider'][i].split('.', 1)[1])
database['country']=ctry

countries, ccount = np.unique(database['country'].values, return_counts=True)

indexes_country = np.argsort(ccount, )
country_sort = []

for ind in indexes_country:
    country_sort.append(countries[ind])
country_sort=np.array(country_sort[::-1])
```

In [8]:
```
nb_countries = (np.unique(database['country'].values).shape)[0]

countries_dict = { country_sort[i] : i+1 for i in range(nb_countries) }
country_id = []
for i in range (nb_of_lines):
    country_id.append(countries_dict[database['country'][i]])
database['country_id'] = country_id
database.head()
```

Out[8]:

	login	password	platform	platform_id	email_provider	email_provider_id	country	country_id
0	joindiscordserver@gmail.com	https	fortnite	1	gmail.com	2	international	2
1	joindiscordserver@gmail.com	httpsfortnite@arcor.de:1fckoeln123	fortnite	1	gmail.com	2	international	2
2	fortnite@freenet.de	tiberius	fortnite	1	freenet.de	4	de	1
3	fortnite@aliceadsl.fr	navarro	fortnite	1	aliceadsl.fr	32	fr	3
4	fortnite@virgilio.it	211089	fortnite	1	virgilio.it	139	it	8

II.3.4 Adding password score and treating the different vulnerabilities

First, password scores

In [9]:
```
import passwordmeter

score, advice = [], []

for i in range(nb_of_lines):
    strength, improvements = passwordmeter.test(database['password'].values[i])
    score.append(strength)
    advice.append(improvements)
```

In [10]:
```
database['score'] = score
database['improvements'] = advice
database.head()
```

Out[10]:

	login	password	platform	platform_id	email_provider	email_provider_id	country	country_id
0	joindiscordserver@gmail.com	https	fortnite	1	gmail.com	2	international	2
1	joindiscordserver@gmail.com	httpsfortnite@arcor.de:1fckoeln123	fortnite	1	gmail.com	2	international	2
2	fortnite@freenet.de	tiberius	fortnite	1	freenet.de	4	de	1
3	fortnite@aliceadsl.fr	navarro	fortnite	1	aliceadsl.fr	32	fr	3
4	fortnite@virgilio.it	211089	fortnite	1	virgilio.it	139	it	8

Then, adding binary variables for the 6 following improvement strategies

In [11]:
```
dictio = database['improvements'][0].copy()
for i in range(nb_of_lines):
    dictio.update(database['improvements'][i])
dictio
```

Out[11]:
```
{'length': 'Increase the length of the password',
 'charmix': 'Use a good mix of numbers, letters, and symbols',
 'casemix': 'Use a good mix of UPPER case and lower case letters',
 'phrase': 'Passphrases (e.g. an obfuscated sentence) are better than passwords',
 'notword': 'Avoid using one of the ten thousand most common passwords',
 'variety': 'Minimize character duplicates and repetitions'}
```

In [12]:
```
length_suggestion = []
charmix_suggestion = []
casemix_suggestion = []
phrase_suggestion = []
notword_suggestion = []
variety_suggestion = []

for i in range(nb_of_lines):
    if 'length' in database['improvements'][i]:
        length_suggestion.append(1)
    else :
        length_suggestion.append(0)
    if 'charmix' in database['improvements'][i]:
        charmix_suggestion.append(1)
    else :
        charmix_suggestion.append(0)
    if 'casemix' in database['improvements'][i]:
        casemix_suggestion.append(1)
    else :
        casemix_suggestion.append(0)
    if 'phrase' in database['improvements'][i]:
        phrase_suggestion.append(1)
    else :
        phrase_suggestion.append(0)
    if 'notword' in database['improvements'][i]:
        notword_suggestion.append(1)
    else :
        notword_suggestion.append(0)
    if 'variety' in database['improvements'][i]:
        variety_suggestion.append(1)
    else :
        variety_suggestion.append(0)

database['length_suggestion'] = length_suggestion
database['charmix_suggestion'] = charmix_suggestion
database['casemix_suggestion'] = casemix_suggestion
database['phrase_suggestion'] = phrase_suggestion
database['notword_suggestion'] = notword_suggestion
database['variety_suggestion'] = variety_suggestion
```

In [13]:
```
database.head()
```

Out[13]:

	login	password	platform	platform_id	email_provider	email_provider_id	country	country_i
0	joindiscordserver@gmail.com	https	fortnite	1	gmail.com	2	international	2
1	joindiscordserver@gmail.com	httpsfortnite@arcor.de:1fckoeln123	fortnite	1	gmail.com	2	international	2
2	fortnite@freenet.de	tiberius	fortnite	1	freenet.de	4	de	1
3	fortnite@aliceadsl.fr	navarro	fortnite	1	aliceadsl.fr	32	fr	3
4	fortnite@virgilio.it	211089	fortnite	1	virgilio.it	139	it	8

II.3.5 Discrete score and time to crack

In [14]:
```
from zxcvbn import zxcvbn

score1_4 = []
time_to_crack = []
nb_guesses_to_crack = []
suggestion = []

for i in range(311536):
    if database['password'].values[i]!='':
        results = zxcvbn(database['password'].values[i])
        score1_4.append(results['score'])
        time_to_crack.append(results['crack_times_seconds']['offline_fast_hashing_1e10_per_second'])
        nb_guesses_to_crack.append(results['guesses'])
        suggestion.append(results['feedback']['suggestions'])
    else :
        score1_4.append(0)
        time_to_crack.append(0)
        nb_guesses_to_crack.append(0)
        suggestion.append('')
```

In [15]:
```
database['score1_4'] = score1_4
database['time_to_crack'] = time_to_crack
database['nb_guesses_to_crack'] = nb_guesses_to_crack
database['suggestion'] = suggestion
```

In [16]:
```
database.head()
```

Out[16]:

	login	password	platform	platform_id	email_provider	email_provider_id	country	country_id
0	joindiscordserver@gmail.com	https	fortnite	1	gmail.com	2	international	2
1	joindiscordserver@gmail.com	httpsfortnite@arcor.de:1fckoeln123	fortnite	1	gmail.com	2	international	2
2	fortnite@freenet.de	tiberius	fortnite	1	freenet.de	4	de	1
3	fortnite@aliceadsl.fr	navarro	fortnite	1	aliceadsl.fr	32	fr	3
4	fortnite@virgilio.it	211089	fortnite	1	virgilio.it	139	it	8

In [17]:
```
#Example of all the data zxcvbn gives us
zxcvbn(database['password'].values[1])
```

Out[17]:

```
{'password': 'httpsfortnite@arcor.de:1fckoeln123',
 'guesses': Decimal('3.000000000000000010000000000E+29'),
 'guesses_log10': 29.477121254719663,
 'sequence': [{'pattern': 'dictionary',
   'i': 0,
   'j': 4,
   'token': 'https',
   'matched_word': 'https',
   'rank': 21,
   'dictionary_name': 'english_wikipedia',
   'reversed': False,
   'l33t': False,
   'base_guesses': 21,
   'uppercase_variations': 1,
   'l33t_variations': 1,
   'guesses': 50,
   'guesses_log10': 1.6989700043360185},
  {'pattern': 'dictionary',
   'i': 5,
   'j': 7,
   'token': 'for',
   'matched_word': 'for',
   'rank': 7,
   'dictionary_name': 'english_wikipedia',
   'reversed': False,
   'l33t': False,
   'base_guesses': 7,
   'uppercase_variations': 1,
   'l33t_variations': 1,
   'guesses': 50,
   'guesses_log10': 1.6989700043360185},
  {'pattern': 'bruteforce',
   'token': 'tnite@arcor.de:1fckoeln',
   'i': 8,
   'j': 30,
   'guesses': 10000000000000000000000,
   'guesses_log10': 22.999999999999996},
  {'pattern': 'sequence',
   'i': 31,
   'j': 33,
   'token': '123',
   'sequence_name': 'digits',
   'sequence_space': 10,
   'ascending': True,
   'guesses': 50,
   'guesses_log10': 1.6989700043360185}],
 'calc_time': datetime.timedelta(microseconds=7290),
 'crack_times_seconds': {'online_throttling_100_per_hour': Decimal('1.080000000000000063552043330E+31')
,
  'online_no_throttling_10_per_second': Decimal('3.000000000000000010000000000E+28'),
  'offline_slow_hashing_1e4_per_second': Decimal('3.000000000000000010000000E+25'),
  'offline_fast_hashing_1e10_per_second': Decimal('3.00000000000000001E+19')},
 'crack_times_display': {'online_throttling_100_per_hour': 'centuries',
  'online_no_throttling_10_per_second': 'centuries',
  'offline_slow_hashing_1e4_per_second': 'centuries',
  'offline_fast_hashing_1e10_per_second': 'centuries'},
 'score': 4,
 'feedback': {'warning': '', 'suggestions': []}}
```


II.3.6 Exporting to csv

In [18]:
```
database.to_csv('./database.csv', index=False)
```

Code 2 - Descriptive analysis data

Importing data

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:
```
database = pd.read_csv('./database.csv')
```

In [3]:
```
database.head()
```

Out[3]:

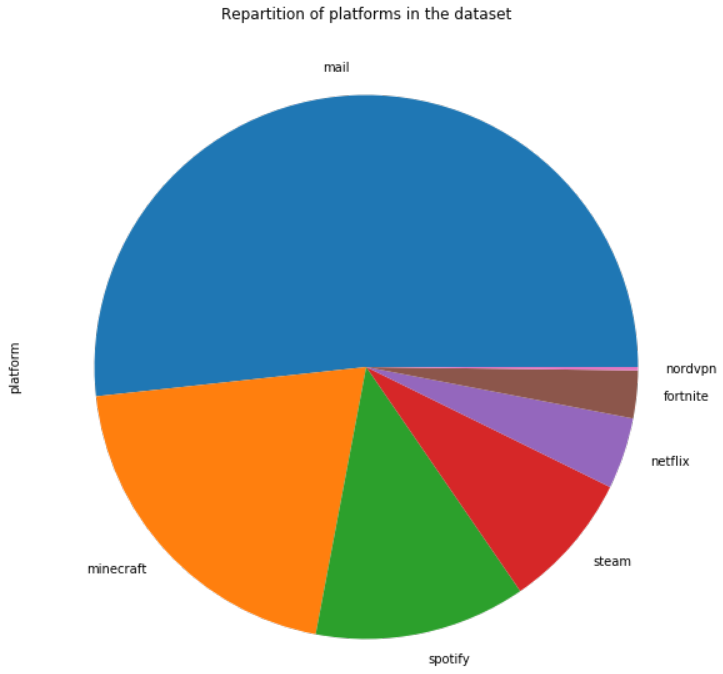
	login	password	platform	platform_id	email_provider	email_provider_id	country	country_id
0	joindiscordserver@gmail.com	https	fortnite	1	gmail.com	2	international	2
1	joindiscordserver@gmail.com	httpsfortnite@arcor.de:1fckoeln123	fortnite	1	gmail.com	2	international	2
2	fortnite@freenet.de	tiberius	fortnite	1	freenet.de	4	de	1
3	fortnite@aliceadsl.fr	navarro	fortnite	1	aliceadsl.fr	32	fr	3
4	fortnite@virgilio.it	211089	fortnite	1	virgilio.it	139	it	8

III.1 Repartitions in the dataset

Platforms

In [4]:
```
pie1 = database['platform'].value_counts(normalize=False, sort=True, ascending=False, bins=None, dropna=True)
pie1.plot.pie(title = 'Repartition of platforms in the dataset', figsize=(10,10))
```

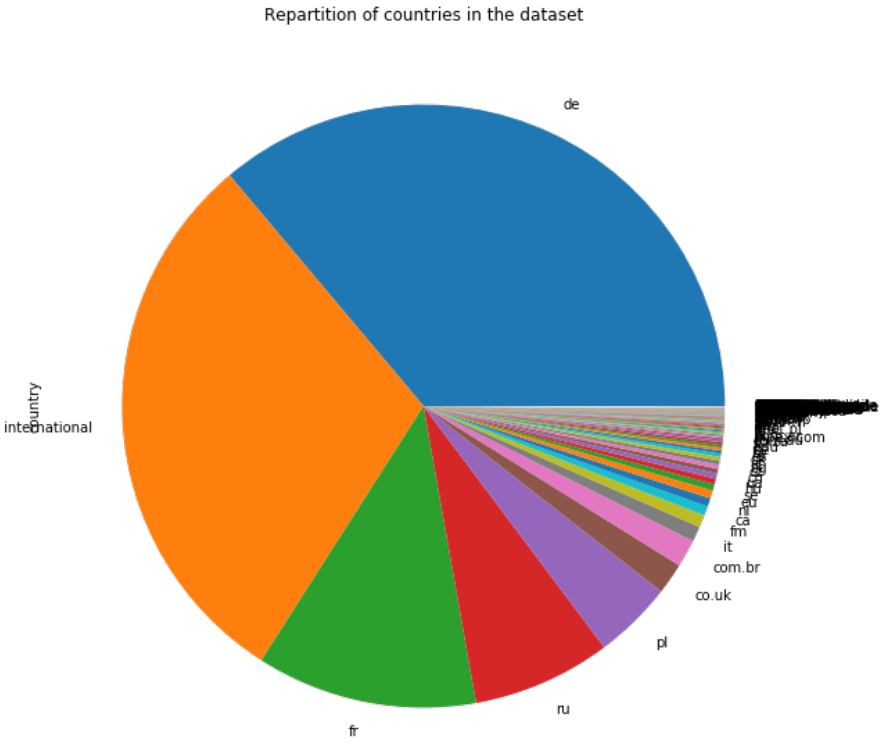
Out[4]: <matplotlib.axes._subplots.AxesSubplot at 0x11d3114a8>



Countries

```
In [5]: pie1 = database['country'].value_counts(normalize=False, sort=True, ascending=False, bins=None, dropna=True)
        pie1.plot.pie(title = 'Repartition of countries in the dataset', figsize=(10,10))
```

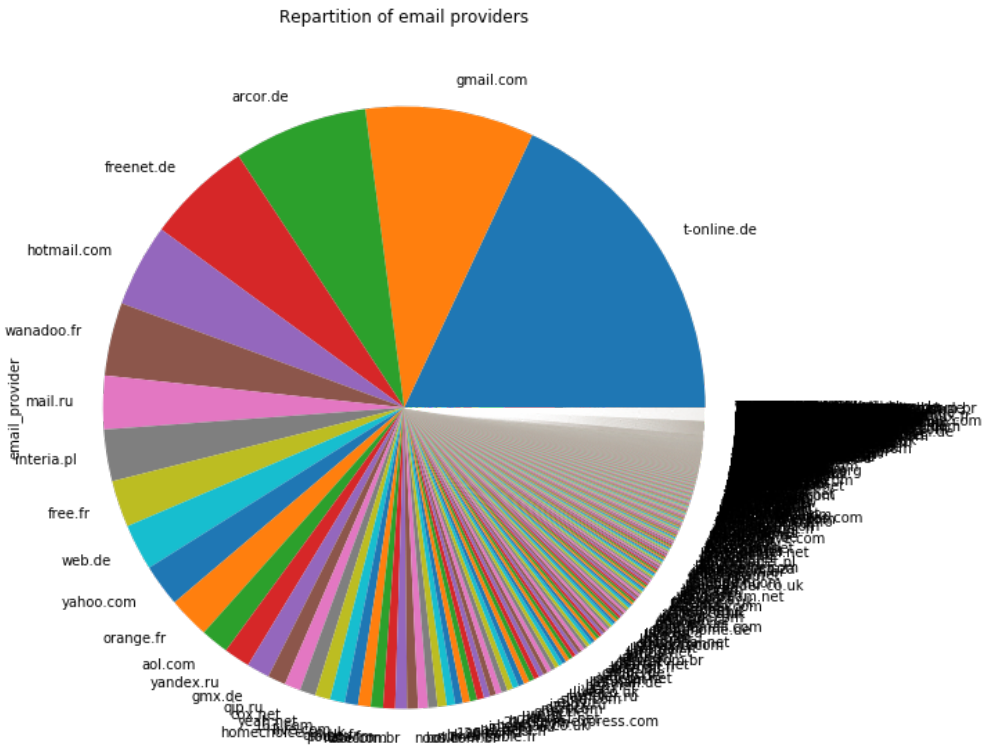
```
Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x10773cdd8>
```



Email providers

```
In [6]: pie2 = database['email_provider'].value_counts(normalize=False, sort=True, ascending=False, bins=None, dropna=True)
        pie2.plot.pie(title = 'Repartition of email providers', figsize=(10,10))
```

```
Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x1193d0358>
```



III.2 Relevant informations on the data

Top 25 of mostly used passwords

In [7]:
```
pd.DataFrame(database['password'].value_counts()).iloc[:25]
```

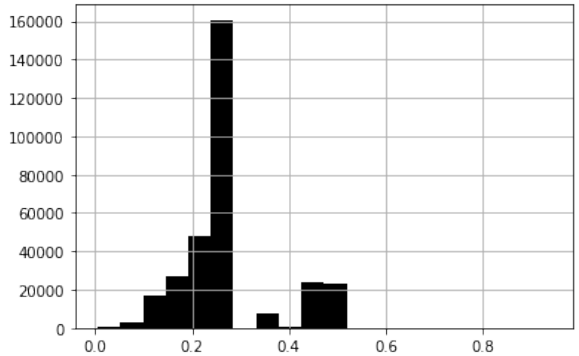
Out[7]:

	password
123456	758
123456789	244
schalke04	120
12345678	115
password	98
111111	94
hallo123	92
taishan2011	90
1q2w3e4r	88
internet	85
qwerty	77
12345	73
123123	71
000000	69
zaq12wsx	69
azerty	64
hassan123	63
passwort	63
1234567890	61
killer	59
daniel	57
benjamin	57
alexander	56
1q2w3e	54
doudou	52

Histogram of scores

In [8]:
```
database['score'].hist(bins = 20, color='black')
```

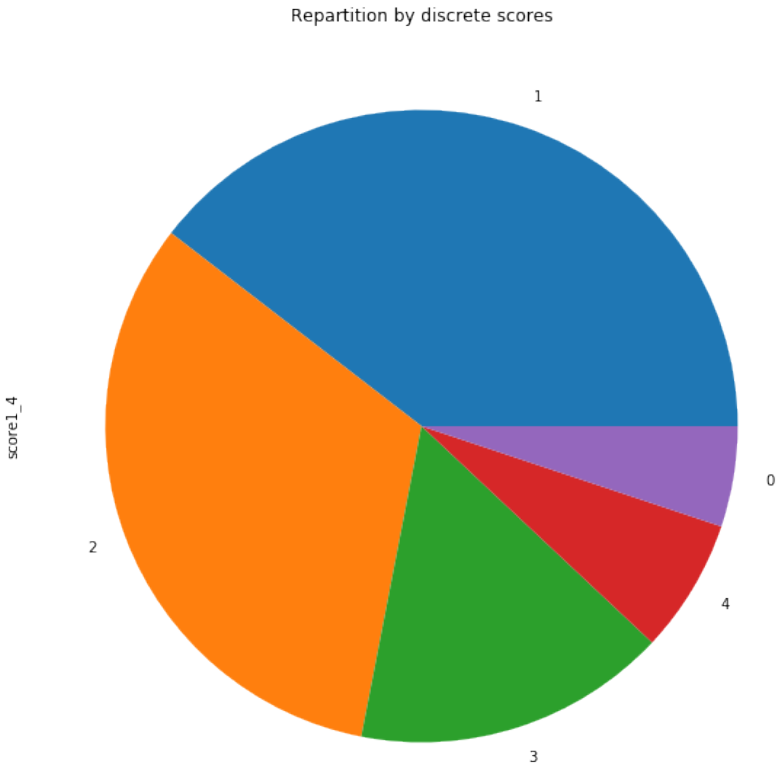
Out[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x121880780>
```



Discrete score repartition

In [9]:
```
pie3 = database['score1_4'].value_counts(normalize=False, sort=True, ascending=False, bins=None, dropna=True)
pie3.plot.pie(title = 'Repartition by discrete scores', figsize=(10,10))
```

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x12187cf28>



III.3 Basic statistics on the data

In [10]:
```
database[['password','platform','email_provider','country','score','score1_4','time_to_crack','length_suggestion','charmix_suggestion','casemix_suggestion','phrase_suggestion','notword_suggestion','variety_suggestion','score1_4','time_to_crack','nb_guesses_to_crack']].describe(include='all')
```

Out[10]:

	password	platform	email_provider	country	score	score1_4	time_to_crack	length_suggestion	charmix_suggestion	c
count	311208	311536	311536	311536	311536.000000	311536.000000	3.115360e+05	311536.000000	311536.000000	
unique	175712	7	4676	408	NaN	NaN	NaN	NaN	NaN	
top	123456	mail	t-online.de	de	NaN	NaN	NaN	NaN	NaN	
freq	758	161133	56101	112552	NaN	NaN	NaN	NaN	NaN	
mean	NaN	NaN	NaN	NaN	0.269993	1.799484	1.330457e+47	0.236814	0.996235	
std	NaN	NaN	NaN	NaN	0.100198	0.997752	5.907667e+49	0.425128	0.061246	
min	NaN	NaN	NaN	NaN	0.005495	0.000000	0.000000e+00	0.000000	0.000000	
25%	NaN	NaN	NaN	NaN	0.209912	1.000000	5.730000e-06	0.000000	1.000000	
50%	NaN	NaN	NaN	NaN	0.262107	2.000000	1.990000e-04	0.000000	1.000000	
75%	NaN	NaN	NaN	NaN	0.265260	2.000000	1.000000e-02	0.000000	1.000000	
max	NaN	NaN	NaN	NaN	0.939885	4.000000	3.140824e+52	1.000000	1.000000	

III.4 Statistics following certain categories

Vulnerabilities

In [11]:
```
database[['length_suggestion','charmix_suggestion','casemix_suggestion','phrase_suggestion','notword_suggestion','variety_suggestion']].describe(include='all')
```

Out[11]:

	length_suggestion	charmix_suggestion	casemix_suggestion	phrase_suggestion	notword_suggestion	variety_suggestion
count	311536.000000	311536.000000	311536.000000	311536.0	311536.000000	311536.000000
mean	0.236814	0.996235	0.818705	1.0	0.069693	0.136713
std	0.425128	0.061246	0.385263	0.0	0.254630	0.343545
min	0.000000	0.000000	0.000000	1.0	0.000000	0.000000
25%	0.000000	1.000000	1.000000	1.0	0.000000	0.000000
50%	0.000000	1.000000	1.000000	1.0	0.000000	0.000000
75%	0.000000	1.000000	1.000000	1.0	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.0	1.000000	1.000000

Score by country (25 mostly represented countries)

In [12]:
```
database[['country','score']].groupby(['country']).describe().iloc[:25]
```

Out[12]:

	score							
	count	mean	std	min	25%	50%	75%	max
country								
CA	3.0	0.468270	0.006617	0.464449	0.464449	0.464449	0.470180	0.475910
CO.UK	1.0	0.265710	NaN	0.265710	0.265710	0.265710	0.265710	0.265710
COM	121.0	0.235691	0.111952	0.088433	0.163876	0.212501	0.262571	0.844989
COM.BR	1.0	0.255412	NaN	0.255412	0.255412	0.255412	0.255412	0.255412
COM.PL	1.0	0.264396	NaN	0.264396	0.264396	0.264396	0.264396	0.264396
Com	4.0	0.413440	0.237301	0.192042	0.247031	0.370803	0.537211	0.720110
DK	2.0	0.267321	0.000000	0.267321	0.267321	0.267321	0.267321	0.267321
EDU	6.0	0.200557	0.000000	0.200557	0.200557	0.200557	0.200557	0.200557
ES	2.0	0.166059	0.000000	0.166059	0.166059	0.166059	0.166059	0.166059
FR	10.0	0.244201	0.041579	0.165381	0.261491	0.262982	0.264751	0.266402
Fr	2.0	0.212704	0.071903	0.161860	0.187282	0.212704	0.238125	0.263547
IT	3.0	0.209167	0.046801	0.162787	0.185561	0.208335	0.232357	0.256379
NET	9.0	0.273071	0.154603	0.105915	0.167325	0.205124	0.465624	0.470854
Net	2.0	0.154875	0.067377	0.107232	0.131054	0.154875	0.178696	0.202518
PL	4.0	0.198538	0.051804	0.158363	0.158363	0.184419	0.224594	0.266952
RR.COM	1.0	0.212345	NaN	0.212345	0.212345	0.212345	0.212345	0.212345
RU	52.0	0.202975	0.073655	0.084233	0.099129	0.255693	0.264421	0.268511
RY	2.0	0.184393	0.026669	0.165535	0.174964	0.184393	0.193823	0.203252
Ru	1.0	0.258006	NaN	0.258006	0.258006	0.258006	0.258006	0.258006
Se	2.0	0.357206	0.134380	0.262185	0.309696	0.357206	0.404717	0.452227
UR	1.0	0.199292	NaN	0.199292	0.199292	0.199292	0.199292	0.199292
a2000.nl	1.0	0.264751	NaN	0.264751	0.264751	0.264751	0.264751	0.264751
abb.com	1.0	0.476418	NaN	0.476418	0.476418	0.476418	0.476418	0.476418
ac.in	1.0	0.109252	NaN	0.109252	0.109252	0.109252	0.109252	0.109252
ac.jp	1.0	0.471599	NaN	0.471599	0.471599	0.471599	0.471599	0.471599

Score by platform

In [13]:
```
database[['platform','score']].groupby(['platform']).describe()
```

Out[13]:

	score							
	count	mean	std	min	25%	50%	75%	max
platform								
fortnite	8828.0	0.272347	0.096281	0.065708	0.210793	0.262185	0.265618	0.913274
mail	161133.0	0.267033	0.097240	0.065708	0.209571	0.261949	0.265162	0.939424
minecraft	63243.0	0.261118	0.093315	0.065708	0.209132	0.261493	0.264820	0.934013
netflix	13226.0	0.282421	0.116229	0.065708	0.209860	0.262272	0.265723	0.927352
nordvpn	622.0	0.117149	0.139130	0.005495	0.005495	0.005495	0.178107	0.478778
spotify	39153.0	0.291891	0.109890	0.065708	0.254027	0.263171	0.266391	0.939885
steam	25331.0	0.273577	0.102476	0.065708	0.210327	0.262185	0.265618	0.932673

Score by email provider (25 biggest providers)

In [14]:
```
database[['email_provider','score']].groupby(['email_provider']).describe().iloc[:25]
```

Out[14]:

	score							
	count	mean	std	min	25%	50%	75%	max
email_provider								
0000.ru	1.0	0.212345	NaN	0.212345	0.212345	0.212345	0.212345	0.212345
01019freenet.de	4.0	0.238459	0.031112	0.211515	0.211515	0.238459	0.265402	0.265402
010fuss.com	1.0	0.256962	NaN	0.256962	0.256962	0.256962	0.256962	0.256962
02.pl	1.0	0.202518	NaN	0.202518	0.202518	0.202518	0.202518	0.202518
0bellsouth.net	1.0	0.205124	NaN	0.205124	0.205124	0.205124	0.205124	0.205124
0msn.com	1.0	0.208653	NaN	0.208653	0.208653	0.208653	0.208653	0.208653
101berlin.com	4.0	0.211280	0.000000	0.211280	0.211280	0.211280	0.211280	0.211280
123.com	3.0	0.195581	0.054124	0.164332	0.164332	0.164332	0.211205	0.258077
126.com	1394.0	0.216675	0.075727	0.065708	0.163264	0.210474	0.261860	0.769792
1278.se	3.0	0.161860	0.000000	0.161860	0.161860	0.161860	0.161860	0.161860
1337.no	11.0	0.290743	0.091002	0.209860	0.259804	0.260014	0.265423	0.477242
163.com	2522.0	0.219258	0.069636	0.065708	0.163729	0.211272	0.263004	0.823074
188.com	1.0	0.261858	NaN	0.261858	0.261858	0.261858	0.261858	0.261858
189.cn	46.0	0.227712	0.060565	0.134950	0.164180	0.261014	0.264553	0.468007
1999.ru	1.0	0.260237	NaN	0.260237	0.260237	0.260237	0.260237	0.260237
1ate3.com	2.0	0.209815	0.000000	0.209815	0.209815	0.209815	0.209815	0.209815
1nsk.ru	1.0	0.202082	NaN	0.202082	0.202082	0.202082	0.202082	0.202082
1thecity.biz	1.0	0.209571	NaN	0.209571	0.209571	0.209571	0.209571	0.209571
2.com	1.0	0.109252	NaN	0.109252	0.109252	0.109252	0.109252	0.109252
20000.ru	1.0	0.090657	NaN	0.090657	0.090657	0.090657	0.090657	0.090657
2000bk.ru	1.0	0.260858	NaN	0.260858	0.260858	0.260858	0.260858	0.260858
2010live.com	1.0	0.210474	NaN	0.210474	0.210474	0.210474	0.210474	0.210474
20hours.it	1.0	0.206628	NaN	0.206628	0.206628	0.206628	0.206628	0.206628
211.ru	1.0	0.267375	NaN	0.267375	0.267375	0.267375	0.267375	0.267375
21cn.com	1008.0	0.185985	0.046705	0.068527	0.162787	0.164844	0.208514	0.482222

Platform and vulnerabilities

In [15]:
```
pd.set_option('display.max_columns', 500)
database[['platform','length_suggestion','charmix_suggestion','casemix_suggestion','phrase_suggestion','
notword_suggestion','variety_suggestion']].groupby(['platform']).describe()
```

Out[15]:

	length_suggestion								charmix_suggestion								casemix_su	
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max	count	ı
platform																		
fortnite	8828.0	0.232556	0.422485	0.0	0.0	0.0	0.0	1.0	8828.0	0.998641	0.036846	0.0	1.0	1.0	1.0	1.0	8828.0	(
mail	161133.0	0.257793	0.437421	0.0	0.0	0.0	1.0	1.0	161133.0	0.998796	0.034678	0.0	1.0	1.0	1.0	1.0	161133.0	(
minecraft	63243.0	0.255159	0.435954	0.0	0.0	0.0	1.0	1.0	63243.0	0.997723	0.047663	0.0	1.0	1.0	1.0	1.0	63243.0	(
netflix	13226.0	0.179571	0.383844	0.0	0.0	0.0	0.0	1.0	13226.0	0.979208	0.142694	0.0	1.0	1.0	1.0	1.0	13226.0	(
nordvpn	622.0	0.856913	0.350443	0.0	1.0	1.0	1.0	1.0	622.0	1.000000	0.000000	1.0	1.0	1.0	1.0	1.0	622.0	(
spotify	39153.0	0.143591	0.350679	0.0	0.0	0.0	0.0	1.0	39153.0	0.988149	0.108216	0.0	1.0	1.0	1.0	1.0	39153.0	(
steam	25331.0	0.217796	0.412756	0.0	0.0	0.0	0.0	1.0	25331.0	0.996684	0.057491	0.0	1.0	1.0	1.0	1.0	25331.0	(

Countries and vulnerabilities

In [16]:
```
pd.set_option('display.max_columns', 500)
database[['country','length_suggestion','charmix_suggestion','casemix_suggestion','phrase_suggestion','n
otword_suggestion','variety_suggestion']].groupby(['country']).describe().iloc[:25]
```

Out[16]:

	length_suggestion								charmix_suggestion								casemix_suggesti	
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max	count	mean
country																		
CA	3.0	0.000000	0.000000	0.0	0.00	0.0	0.00	0.0	3.0	1.000000	0.000000	1.0	1.00	1.0	1.0	1.0	3.0	0.000000
CO.UK	1.0	0.000000	NaN	0.0	0.00	0.0	0.00	0.0	1.0	1.000000	NaN	1.0	1.00	1.0	1.0	1.0	1.0	1.000000
COM	121.0	0.396694	0.491246	0.0	0.00	0.0	1.00	1.0	121.0	0.991736	0.090909	0.0	1.00	1.0	1.0	1.0	121.0	0.884298
COM.BR	1.0	1.000000	NaN	1.0	1.00	1.0	1.00	1.0	1.0	1.000000	NaN	1.0	1.00	1.0	1.0	1.0	1.0	1.000000
COM.PL	1.0	0.000000	NaN	0.0	0.00	0.0	0.00	0.0	1.0	1.000000	NaN	1.0	1.00	1.0	1.0	1.0	1.0	1.000000
Com	4.0	0.000000	0.000000	0.0	0.00	0.0	0.00	0.0	4.0	0.750000	0.500000	0.0	0.75	1.0	1.0	1.0	4.0	0.750000
DK	2.0	0.000000	0.000000	0.0	0.00	0.0	0.00	0.0	2.0	1.000000	0.000000	1.0	1.00	1.0	1.0	1.0	2.0	1.000000
EDU	6.0	1.000000	0.000000	1.0	1.00	1.0	1.00	1.0	6.0	1.000000	0.000000	1.0	1.00	1.0	1.0	1.0	6.0	1.000000
ES	2.0	0.000000	0.000000	0.0	0.00	0.0	0.00	0.0	2.0	1.000000	0.000000	1.0	1.00	1.0	1.0	1.0	2.0	1.000000
FR	10.0	0.200000	0.421637	0.0	0.00	0.0	0.00	1.0	10.0	1.000000	0.000000	1.0	1.00	1.0	1.0	1.0	10.0	1.000000
Fr	2.0	0.500000	0.707107	0.0	0.25	0.5	0.75	1.0	2.0	1.000000	0.000000	1.0	1.00	1.0	1.0	1.0	2.0	1.000000
IT	3.0	0.333333	0.577350	0.0	0.00	0.0	0.50	1.0	3.0	1.000000	0.000000	1.0	1.00	1.0	1.0	1.0	3.0	1.000000
NET	9.0	0.555556	0.527046	0.0	0.00	1.0	1.00	1.0	9.0	1.000000	0.000000	1.0	1.00	1.0	1.0	1.0	9.0	0.666667
Net	2.0	1.000000	0.000000	1.0	1.00	1.0	1.00	1.0	2.0	1.000000	0.000000	1.0	1.00	1.0	1.0	1.0	2.0	1.000000
PL	4.0	0.500000	0.577350	0.0	0.00	0.5	1.00	1.0	4.0	1.000000	0.000000	1.0	1.00	1.0	1.0	1.0	4.0	1.000000
RR.COM	1.0	0.000000	NaN	0.0	0.00	0.0	0.00	0.0	1.0	1.000000	NaN	1.0	1.00	1.0	1.0	1.0	1.0	1.000000
RU	52.0	0.115385	0.322603	0.0	0.00	0.0	0.00	1.0	52.0	1.000000	0.000000	1.0	1.00	1.0	1.0	1.0	52.0	1.000000
RY	2.0	0.000000	0.000000	0.0	0.00	0.0	0.00	0.0	2.0	1.000000	0.000000	1.0	1.00	1.0	1.0	1.0	2.0	1.000000
Ru	1.0	0.000000	NaN	0.0	0.00	0.0	0.00	0.0	1.0	1.000000	NaN	1.0	1.00	1.0	1.0	1.0	1.0	1.000000
Se	2.0	0.000000	0.000000	0.0	0.00	0.0	0.00	0.0	2.0	1.000000	0.000000	1.0	1.00	1.0	1.0	1.0	2.0	0.500000
UR	1.0	1.000000	NaN	1.0	1.00	1.0	1.00	1.0	1.0	1.000000	NaN	1.0	1.00	1.0	1.0	1.0	1.0	1.000000
a2000.nl	1.0	0.000000	NaN	0.0	0.00	0.0	0.00	0.0	1.0	1.000000	NaN	1.0	1.00	1.0	1.0	1.0	1.0	1.000000
abb.com	1.0	0.000000	NaN	0.0	0.00	0.0	0.00	0.0	1.0	1.000000	NaN	1.0	1.00	1.0	1.0	1.0	1.0	0.000000
ac.in	1.0	1.000000	NaN	1.0	1.00	1.0	1.00	1.0	1.0	1.000000	NaN	1.0	1.00	1.0	1.0	1.0	1.0	1.000000
ac.jp	1.0	0.000000	NaN	0.0	0.00	0.0	0.00	0.0	1.0	1.000000	NaN	1.0	1.00	1.0	1.0	1.0	1.0	0.000000

Email providers and vulnerabilities

In [17]:
```
pd.set_option('display.max_columns', 500)
database[['email_provider','length_suggestion','charmix_suggestion','casemix_suggestion','phrase_suggestion','notword_suggestion','variety_suggestion']].groupby(['email_provider']).describe().iloc[:25]
```

Out[17]:

	length_suggestion								charmix_suggestion								casemix_s	
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max	count	n
email_provider																		
0000.ru	1.0	0.000000	NaN	0.0	0.0	0.0	0.0	0.0	1.0	1.000000	NaN	1.0	1.0	1.0	1.0	1.0	1.0	1
01019freenet.de	4.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0	4.0	1.000000	0.000000	1.0	1.0	1.0	1.0	1.0	4.0	1
010fuss.com	1.0	1.000000	NaN	1.0	1.0	1.0	1.0	1.0	1.0	1.000000	NaN	1.0	1.0	1.0	1.0	1.0	1.0	1
02.pl	1.0	1.000000	NaN	1.0	1.0	1.0	1.0	1.0	1.0	1.000000	NaN	1.0	1.0	1.0	1.0	1.0	1.0	1
0bellsouth.net	1.0	1.000000	NaN	1.0	1.0	1.0	1.0	1.0	1.0	1.000000	NaN	1.0	1.0	1.0	1.0	1.0	1.0	1
0msn.com	1.0	0.000000	NaN	0.0	0.0	0.0	0.0	0.0	1.0	1.000000	NaN	1.0	1.0	1.0	1.0	1.0	1.0	1
101berlin.com	4.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0	4.0	1.000000	0.000000	1.0	1.0	1.0	1.0	1.0	4.0	1
123.com	3.0	1.000000	0.000000	1.0	1.0	1.0	1.0	1.0	3.0	1.000000	0.000000	1.0	1.0	1.0	1.0	1.0	3.0	1
126.com	1394.0	0.347202	0.476252	0.0	0.0	0.0	1.0	1.0	1394.0	0.998565	0.037864	0.0	1.0	1.0	1.0	1.0	1394.0	0
1278.se	3.0	1.000000	0.000000	1.0	1.0	1.0	1.0	1.0	3.0	1.000000	0.000000	1.0	1.0	1.0	1.0	1.0	3.0	1
1337.no	11.0	0.181818	0.404520	0.0	0.0	0.0	0.0	1.0	11.0	1.000000	0.000000	1.0	1.0	1.0	1.0	1.0	11.0	0
163.com	2522.0	0.322363	0.467474	0.0	0.0	0.0	1.0	1.0	2522.0	0.998810	0.034476	0.0	1.0	1.0	1.0	1.0	2522.0	0
188.com	1.0	0.000000	NaN	0.0	0.0	0.0	0.0	0.0	1.0	1.000000	NaN	1.0	1.0	1.0	1.0	1.0	1.0	1
189.cn	46.0	0.239130	0.431266	0.0	0.0	0.0	0.0	1.0	46.0	1.000000	0.000000	1.0	1.0	1.0	1.0	1.0	46.0	0
1999.ru	1.0	0.000000	NaN	0.0	0.0	0.0	0.0	0.0	1.0	1.000000	NaN	1.0	1.0	1.0	1.0	1.0	1.0	1
1ate3.com	2.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0	2.0	1.000000	0.000000	1.0	1.0	1.0	1.0	1.0	2.0	1
1nsk.ru	1.0	1.000000	NaN	1.0	1.0	1.0	1.0	1.0	1.0	1.000000	NaN	1.0	1.0	1.0	1.0	1.0	1.0	1
1thecity.biz	1.0	0.000000	NaN	0.0	0.0	0.0	0.0	0.0	1.0	1.000000	NaN	1.0	1.0	1.0	1.0	1.0	1.0	1
2.com	1.0	1.000000	NaN	1.0	1.0	1.0	1.0	1.0	1.0	1.000000	NaN	1.0	1.0	1.0	1.0	1.0	1.0	1
20000.ru	1.0	0.000000	NaN	0.0	0.0	0.0	0.0	0.0	1.0	1.000000	NaN	1.0	1.0	1.0	1.0	1.0	1.0	1
2000bk.ru	1.0	0.000000	NaN	0.0	0.0	0.0	0.0	0.0	1.0	1.000000	NaN	1.0	1.0	1.0	1.0	1.0	1.0	1
2010live.com	1.0	0.000000	NaN	0.0	0.0	0.0	0.0	0.0	1.0	1.000000	NaN	1.0	1.0	1.0	1.0	1.0	1.0	1
20hours.it	1.0	0.000000	NaN	0.0	0.0	0.0	0.0	0.0	1.0	1.000000	NaN	1.0	1.0	1.0	1.0	1.0	1.0	1
211.ru	1.0	0.000000	NaN	0.0	0.0	0.0	0.0	0.0	1.0	1.000000	NaN	1.0	1.0	1.0	1.0	1.0	1.0	1
21cn.com	1008.0	0.296627	0.456997	0.0	0.0	0.0	1.0	1.0	1008.0	1.000000	0.000000	1.0	1.0	1.0	1.0	1.0	1008.0	0

In [ ]:

Code 3 - Regressions

Regression score - email provider

```
#Variable binaire

tonline_bin = []
gmail_bin = []
arcor_bin = []
freenet_bin = []
hotmail_bin = []
wanadoo_bin = []
mail_bin = []
interia_bin = []
free_bin = []
web_bin = []
yahoo_bin = []
orange_bin = []
aol_bin = []
yandex_bin = []
littleprovider_bin = []

for i in range(nb_of_lines):
    taken = False
    if 'gmail.com' in database['email_provider'][i]:
        gmail_bin.append(1)
    else :
        gmail_bin.append(0)
    if 't_online.de' in database['email_provider'][i]:
        tonline_bin.append(1)
    else :
        tonline_bin.append(0)
    if 'arcor.de' in database['email_provider'][i]:
        arcor_bin.append(1)
    else :
        arcor_bin.append(0)
    if 'freenet.de' in database['email_provider'][i]:
        freenet_bin.append(1)
    else :
        freenet_bin.append(0)
    if 'hotmail.com' in database['email_provider'][i]:
        hotmail_bin.append(1)
    else :
        hotmail_bin.append(0)
    if 'wanadoo.fr' in database['email_provider'][i]:
        wanadoo_bin.append(1)
    else :
        wanadoo_bin.append(0)
    if 'mail.ru' in database['email_provider'][i]:
        mail_bin.append(1)
    else :
        mail_bin.append(0)
    if 'interia.pl' in database['email_provider'][i]:
        interia_bin.append(1)
    else :
        interia_bin.append(0)
    if 'free.fr' in database['email_provider'][i]:
        free_bin.append(1)
    else :
        free_bin.append(0)
    if 'web.de' in database['email_provider'][i]:
        web_bin.append(1)
    else :
        web_bin.append(0)
    if 'yahoo.com' in database['email_provider'][i]:
        yahoo_bin.append(1)
    else :
        yahoo_bin.append(0)
    if 'orange.fr' in database['email_provider'][i]:
        orange_bin.append(1)
    else :
        orange_bin.append(0)
    if 'aol.com' in database['email_provider'][i]:
        aol_bin.append(1)
    else :
        aol_bin.append(0)
    if 'yandex.ru' in database['email_provider'][i]:
        yandex_bin.append(1)
    else :
        yandex_bin.append(0)
    if not taken :
        littleprovider_bin.append(1)
    else :
        littleprovider_bin.append(0)


#Création de la variable binaire
Xmail = np.zeros((311536, 15))
Xmail[:,0] = tonline_bin
Xmail[:,1] = gmail_bin
Xmail[:,2] = arcor_bin
Xmail[:,3] = freenet_bin
Xmail[:,4] = hotmail_bin
Xmail[:,5] = wanadoo_bin
Xmail[:,6] = mail_bin
Xmail[:,7] = interia_bin
Xmail[:,8]= free_bin
Xmail[:,9]= web_bin
Xmail[:,10]= yahoo_bin
Xmail[:,11]= orange_bin
Xmail[:,12]= aol_bin
Xmail[:,13]= yandex_bin
Xmail[:,14] = littleprovider_bin
Xmail
```

```
#Regression
Y = database['score']
model_mail = sm.OLS(Y, Xmail).fit()
pred_mail = model_mail.predict(Xmail)
model_mail.summary()
```

Regression score - platform

```
fortnite_bin = []
mail_bin = []
minecraft_bin = []
netflix_bin = []
nordvpn_bin = []
spotify_bin = []
steam_bin = []

for i in range(nb_of_lines):
    if 'fortnite' in database['platform'][i]:
        fortnite_bin.append(1)
    else :
        fortnite_bin.append(0)
    if 'mail' in database['platform'][i]:
        mail_bin.append(1)
    else :
        mail_bin.append(0)
    if 'minecraft' in database['platform'][i]:
        minecraft_bin.append(1)
    else :
        minecraft_bin.append(0)
    if 'netflix' in database['platform'][i]:
        netflix_bin.append(1)
    else :
        netflix_bin.append(0)
    if 'nordvpn' in database['platform'][i]:
        nordvpn_bin.append(1)
    else :
        nordvpn_bin.append(0)
    if 'spotify' in database['platform'][i]:
        spotify_bin.append(1)
    else :
        spotify_bin.append(0)
    if 'steam' in database['platform'][i]:
        steam_bin.append(1)
    else :
        steam_bin.append(0)

database['fortnite_bin'] = fortnite_bin
database['mail_bin'] = mail_bin
database['minecraft_bin'] = minecraft_bin
database['netflix_bin'] = netflix_bin
database['nordvpn_bin'] = nordvpn_bin
database['spotify_bin'] = spotify_bin
database['steam_bin'] = steam_bin

#Variable binaire
X = np.zeros((311536, 7))
X[:,0] = fortnite_bin
X[:,1] = mail_bin
X[:,2] = minecraft_bin
X[:,3] = netflix_bin
X[:,4] = nordvpn_bin
X[:,5] = spotify_bin
X[:,6] = steam_bin
X
```

```
import statsmodels.api as sm
Y = database['score']
model_bin = sm.OLS(Y, X).fit()
pred_bin = model_bin.predict(X)
```

References

[1] https://www.nytimes.com/2010/01/21/technology/21password.html

[2] M. Dell' Amico, P. Michiardi and Y. Roudier, « Password Strength: An Empirical Analysis » *2010 Proceedings IEEE INFOCOM*, San Diego, CA, 2010, pp. 1-9.

[3] Claude Castelluccia, Markus Dürmuth, Daniele Perito, « Adaptive Password-Strength Meters from Markov Models », 2017, https://www.ndss-symposium.org/wp-content/uploads/2017/09/06_3.pdf

[4] https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3324907

[5] https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3142270

[6] https://dl.acm.org/doi/abs/10.1145/3054926

[7] https://www.researchgate.net/publication/305943216_PASSWORD_SECURITY_WHAT_FACTORS_INFLUENCE_GOOD_PASSWORD_PRACTICES