CLASSIFICATION OF PASSWORD STRENGTH

MSc. Data Science - 2023/24
Fundamentals of Data Science
Final project report

Rebecca Conti, Antonella Mei, Jacopo Orsini, Lorenzo Pannacci

December 27, 2023

Abstract

The proposed task is the classification of passwords based on their strength. The dataset we used gave us a high degree of liberty on how to extract features and we decided to do a wide-range study by using four different kinds of features and four different machine learning algorithms.

The unnaturally high results we obtained show the task on this specific dataset was very easy as the variance could almost totally be explained by only the password length, however alternative approaches seems to suggest a strong correlation between password strength and other behaviors in password creation.

Introduction

Passwords have been and still continue to be the most diffused way of authentication throughout the internet, often guarding the access to sensitive material as personal data or payment information. It should be imperative for everyone to have passwords difficult to guess with brute-force or from easily accessible personal informations, approaches often used by malicious users. While people competent with the digital world may understand how this type of attacks work and can act accordingly, those who aren't as knowledgeable are more exposed to this risk.

From this scenario our proposed task is to create a machine learning algorithm capable of classify passwords' strength.

Related works

Given that we used a dataset publicly available on Kaggle we found projects made by different people regarding our same task. While our results are on par with the what others got, it seems that nobody questioned the incredibly high performance of the models and tried to understand the abnormal results (we will focus about it later in the report). Given that many used the same approaches we cite the following as the main related works: This used decision tree and logistic regression models, this used neural networks and this one used TF-IDF for feature extraction.

Work division

We decided to split the work in this way: Rebecca cleaned the dataset and implemented the decision tree model, Antonella create the TF-IDF dataset and implemented the k-NN, Jacopo created the statistic datasets and implemented the logistic regression and Lorenzo created the Per-Character dataset and implemented the neural network.

Datasets

Since the dataset contains passwords as strings and their classification we have total freedom over what features to use. We decided to use three different kind of features:

1. **Statistical features:** This approach extract some useful statistics about the password, as its length, the count of certain character types and their frequency over the total. It also contains boolean values as the presence of special characters or more sophisticated information as the presence of existing words or dates.
2. **TF-IDF features:** This approach uses TF-IDF over the password to give to each character its importance based on both its frequency on the password and the relative frequency over the whole dataset.
3. **Per-Character features:** This approach aims to minimize the loss of information by using one-hot encoding to categorize the character at each position in the password string (lowercase, uppercase, number, symbol, empty).

Given the overwhelming importance we'll see the length will have on the classification we want to create a derivative dataset from the Statistical features where we remove every feature that contains information about the password length, as the count of the character types.

Proposed Methods

The study addresses the analysis of four models applied to the previously described datasets. Those models are:

1. **k-Nearest Neighbors (k-NN):** This is a non-parametric, supervised classifier. It works on the assumption that points with the same class can be found one near another in the feature space. It has the advantage of being a very simple model with only one hyper-parameter but it scales bad with the size of the dataset.
2. **Decision Tree:** This model uses a tree-like structure to make decisions. Each node represent a condition on a feature, each branch a different response to the condition and leafs are the outcome of the prediction. The tree is built by trying to separate classes using as few conditions as possible. This model doesn't work well with outliers and scales bad on large datasets.
3. **Logistic Regression:** This model is usually used for binary classification and is particularly important for its statistical meaning. Since we are dealing with a multi-class classification problem we are using a "one vs. rest" strategy, where the problem is divided into multiple binary classification sub-problems.
4. **Neural Network:** Neural Networks are a very important concept in machine learning as they are a "universal function approximator", meaning that they are capable of model arbitrary complex relations in data. In particular we are using a MLP (Multi-Layer Perceptron) Classifier, where each layer is fully connected with the previous one and the last one has one node for each class. We also decided to limit the network size to only two layers.

Benchmark

As the Kaggle page states the dataset has been created from a real password leak of a website. To obtain the classes it has been used a tool from Georgia Tech University which includes various password strength meters and a password has been inserted into the dataset if certain three meters got the same result. The dataset has 669'880 samples but after some data cleaning operation we got 669'806.

We noticed a great imbalance of the dataset towards the medium strength. From this we can set as baseline the model that always classify a password as medium and that would have around 74% accuracy. Despite that we observed that balancing the dataset only brought minimal decreases in overall performances.

We divided the data into 20% test set and 80% train set, of which 20% is used for the validation of hyper-parameters.

Experimental results

Before discussing the results it's important highlight a critical problem in the data that seems to have passed unnoticed by who used the data before us: the length of the password is capable alone to explain almost all the variance of the dataset, rendering the classification trivial when it is used in some way. Of the datasets we created "Statistical" explicitly contains it and "Per-Character" contains it implicitly by construction. It can be seen from both our results and those of anyone who used the dataset before us that every model that uses the length reach a striking accuracy of over 99%. We therefore decided to consider meaningful only the results obtained on the remaining two datasets:

The **k-Nearest Neighbors (k-NN)** models have been the slowest to compute. While the training is fast, as it is only memorization of the data points of the training set, it has been repeated multiple times together with the inference over the validation data during the hyper-parameter tuning phase we implemented to identify the optimal 'k', the one that yields the best results in terms of performances. This process ensure the best possible model performance given the data, but significantly increases the time to create the model.

The model performed well on all datasets except TF-IDF, that got results only slightly higher than baseline, around 79%. We thought that this could have been caused by the imbalance towards the medium strength, however by balancing we noticed an increase in performances for the other two classes but an overall decrease in accuracy to around 70%.

The **Decision tree** seems to be a good model both in performances and running time. We used the model built in sklearn, applied it on each dataset and studied the attribute '**feature_importances_**' to highlight the important features:

We can see that password length is the most important feature of all, in the Statistical dataset it has an importance of over 0.99, magnitudes orders greater than the second, that as an importance value of only 0.00007. In the same way for Per-Characters features the two most important features are by far "7=empty" and "13=empty", the model probably understood those two lengths as good separators between the three classes.

However we noticed Decision Tree performed very well also on the two datasets without information about length, reaching for both over 92% accuracy. For TF-IDF the two most important features are "q" and "1": probably the presence of the first is a good sign, as it is an unusual letter for real words while the second is negative, as it can by symptom of a lazy number combination like "123456" or the start of a birth year, as "1991". Finally the Statistical without length gives great importance to the uppercase frequency but is otherwise balanced across all the other features.

The **Logistic regression** has proven not to be a good model for our data, as the two meaningful datasets yields an accuracy around 81-84%. Probably the reason of this performances is that the generalization ability of this model is limited to only linear relationships between features. However this model further proves the importance of length in the classification, as models trained on data that contains it still manage to get near-perfect scores.

Given the good results yielded by the previous models the use of **Neural Networks** may seem unnecessary, however we got some meaningful results: while Statistical without length has around the same score as the decision tree model TF-IDF reaches a surprising near-perfect score of almost 99% accuracy.

Conclusions

Every model that uses data with information about length has near-perfect results, but why is it so if by first-hand experience we know that length isn't the only requirement for website password checkers? Our hypothesis is that taking the consensus of three checkers may have rendered the problem too easy as this selection may have removed many factors of the classification, given that every website has different policies on password security. Another fact to consider is that while some checkers may have requirements for a password to be accepted there is no specified middle-ground boundaries, that could easily been set to be only about length.

While the variability of the target label could almost completely be explained by the password length, as it is in most cases in real scenarios, given that the best defense against brute-force approaches to password stealing is having a long password, it is interesting to see that even without this information we managed to create near-perfect models. Those results could be explained by a correlation between password length and other features: people who create long passwords are more likely to be conscious about security and might employ other good practises in password creation.

Another important result we reached with our analysis has a more general character and is the importance of the study of what the model really learned: many before us studied the same problem on the same data, extracted features about length and never questioned the reason of such incredibly high performances, leading to the construction models that simply apply thresholds to the length of the password to determine its strength.