
UNIVERSITÀ
DI PARMA

# Model for password security

**Matteo Gianvenuti**

**Donato Bruno**

Academic year 2023/2024

# Summary

Problem description ..... 3
Technologies used ..... 3
Dataset ..... 3
Stratification ..... 3
Features ..... 4
Accuracy ..... 4
Bibliography ..... 5

# Problem description

Identify and create the best classifier model to determine the strength of a password. In practice, the model must establish which class the password belongs to with a good accuracy, choosing between five classes: "Too weak", "Weak", "Moderate", "Strong", "Very strong".

## Technologies used/tested

We used the scikit-learn library, an efficient tool for machine learning in Python. We tested different classifier models from the library to understand which is the most efficient to choose.

- Gradient Boosting Classifier (ensemble): It is an additive model, which adds decision trees one at a time, trying to improve the model step by step. It is based on negative gradient of the loss function.[1] (An ensemble model combines predictions from multiple models to improve overall performance over that achieved with a single model).
- Random Forest Classifier (ensemble): It is based on a set of independent decision trees, trained on different subsets of the training set. It uses averaging to improve the predictive accuracy and control over-fitting.[2]
- K Neighbors Classifier (neighbors): A classifier implementing the k-nearest neighbors vote. The classification is computed from a simple majority vote of the nearest neighbors of each point: a query point is assigned to the data class that has the most representatives within the nearest neighbors of the point.[3]
- Support Vector Classifier (svm): It is a technique based on the Support Vector Machines, which tries to find a hyperplane that best separates classes in a dataset.[4]

## Dataset

We decided to split our dataset in the classical two parts with the following dimension: 70% training set and 30% test set. This is because the other main option, 80% training set and 20% test set, caused overfitting in our model. Furthermore, a test set that is too small may not be representative for a reliable performance evaluation, especially since our dataset consists of only 195 examples. All other options appeared too unbalanced based on experimental results.

### Stratification

This is one of the most important parts in the data handling. The *stratification* consists of keeping the dataset balanced also after the split. This means maintaining the same number of examples for each class in both the training set and the evaluation set. This prevents the model from learning more about one class than the others.

## Features

As features for the model, we identified seven password characteristics:

- The password length.
- The presence of uppercase characters.
- The presence of digits.
- The presence of special characters as “!”, “@”, “#”, “%” and so on.
- The presence of sequences of digits e.g. “1234”.
- The number of unique characters.
- The presence of common words/phrases.

## Accuracy

We have tested and evaluated all these models to find the best one for our task. The following table also shows the importance of the stratification.

| Model | Acc. with stratification | Acc. without stratification |
|---|---|---|
| K Neighbors Classifier | 74.57% | 71.18% |
| Gradient Boosting Classifier | 74.57% | 67.79% |
| Random Forest Classifier | 74.57% | 64.40% |
| Support Vector Classifier | 66.10% | 64.40% |

We decided to use the Random Forest Classifier for the logical simplicity and the high accuracy. To improve accuracy, we also experimented with the “GridSearchCV” class from scikit-learn. This class allows us to systematically search through multiple hyperparameter combinations by splitting the dataset into several folds and training the model on each fold. It then evaluates the performance of each model configuration to find the best one. This approach was unsuccessful because the dataset was too small, which led to a decrease in performance.

# Bibliography

[1] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html#sklearn.ensemble.GradientBoostingClassifier.

[2] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

[3] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html.

[4] https://scikit-learn.org/stable/modules/svm.html#svm-classification.