


Machine Learning- based Password Strength Checker




Introduction/Project's Purpose

The project aimed to develop a machine learning-based approach for password strength checking, departing from traditional rule-based methods. The goal was to create a dynamic and adaptable system that leverages machine learning algorithms to assess password strength, learning from the classifications of multiple commercial password strength meters.



Importance of Password Strength in Security

- Protection Against Unauthorized Access
- Prevention of Unauthorized Transactions
- Confidentiality of Personal Data
- Mitigation of Security Risks
- Compliance with Security Standards
- User Accountability
- Prevention of Identity Theft



Current challenges in password strength evaluation

1. User Behavior and Education
2. Password Reuse
3. Dynamic Threat Landscape
4. Usability and User Experience
5. Emergence of Advanced Attacks
6. Password Recovery and Reset Processes
7. Technological Advances

Addressing these challenges requires a holistic approach that combines technological solutions, user education, and ongoing adaptation to the evolving threat landscape. Machine learning-based password strength checking, as discussed in your project, is one avenue to explore in improving the effectiveness of password security measures.



Project Goals

Objectives of the Project:

- Develop a Machine Learning-Based Password Strength Checker
- Data Collection and Preparation
- Feature Selection and Tokenization
- Model Training and Evaluation
- Generalization and Robustness
- Usability and Integration
- Documentation and Sharing of Results

Expected Outcomes:

- Machine Learning-Based Password Strength Checker
- Improved Understanding of Password Strength Factors
- Model Performance Evaluation
- Generalization Across Password Datasets
- Usability and Practical Applicability
- Contribution to Password Security Research
- Potential for Future Extensions



Data gathering

Passwords were taken from 000webhost leak

Using PARS by Georgia Tech University we checked the passwords against 3 commercial password strength meters which were Twitter, Microsoft, and Battle.

We only used the passwords that were flagged as weak, medium, or strong by all three commercial strength meters



Libraries used

`import pandas as pd` - Data manipulation and analysis library for working with structured data.

`from sklearn.model_selection import KFold` - Cross-validation method for assessing model performance.

`from sklearn.feature_extraction.text import TfidfVectorizer` - Converts a collection of raw documents to a matrix of TF-IDF features.

`from sklearn.linear_model import LogisticRegression` - Implements logisistical regression, a classification algorithm.

`from sklearn.metrics import accuracy_score, classification_report, confusion_matrix` - Evaluation metrics for assessing model performance.

`import numpy as np` - Numerical computing library for efficient array operations.

`import matplotlib.pyplot as plt` - plotting library for creating visualizations.

`import seaborn as sns` - Data visualization library based on matplotlib, providing additional functionalities.

`import io` - input/output tools for working with streams and file-like objects.

`import csv` - Module for reading and writing CSV files.

Explanation of source code/how it works

```
15 def fix_csv(file_path):
16     corrected_data = []
17     with open(file_path, 'r', encoding='utf-8') as file:
18         reader = csv.reader(file)
19         headers = next(reader)
20         corrected_data.append(headers)
21         for row in reader:
22             password = ','.join(row[:-1])
23             strength = row[-1]
24             corrected_data.append([password, strength])
25     return corrected_data
26
27 file_path = '/Users/JPL/Downloads/Project/data.csv'
28
29 corrected_lines = fix_csv(file_path)
30 buffer = io.StringIO()
31 csv.writer(buffer).writerows(corrected_lines)
32 buffer.seek(0)
33 data = pd.read_csv(buffer, header=0)
34
35 x = data['password'].fillna('')
36 y = data['strength'].astype(int)
```


This section of the code reads the csv file and sorts the data into two arrays

Challenge faced:

- Commas in the passwords

Solution:

- Last column is password and everything before that is combined with a comma



Explanation of source code/how it works

Cont.

```
38 vectorizer = TfidfVectorizer(analyzer='char', lowercase=False)
39 X = vectorizer.fit_transform(X)
40
41 # KFold Cross-Validation setup
42 kf = KFold(n_splits=5, shuffle=True, random_state=42)
43
44 # Lists to store results of each fold
45 accuracies = []
46 confusion_matrices = []
47
48 for train_index, test_index in kf.split(X):
49     X_train, X_test = X[train_index], X[test_index]
50     y_train, y_test = y[train_index], y[test_index]
51
52     clf = LogisticRegression()
53     clf.fit(X_train, y_train)
54
55     predictions = clf.predict(X_test)
56     accuracies.append(accuracy_score(y_test, predictions))
57     confusion_matrices.append(confusion_matrix(y_test, predictions))
```


This section transforms password strings into numerical data that can be used for machine learning classification

Challenge faced:

- Converting password strings into a format suitable for machine learning.
- Ensuring the model is not biased

Solution:

- Used TfidfVectorizer and K-Fold Cross-Validation



Explanation of source code/how it works

Cont.

```
59 # Print overall results
60 print(f"Average Accuracy: {np.mean(accuracies):.2f}")
61 print("\nClassification Report for last fold: ")
62 print(classification_report(y_test, predictions))
63
64 # Visualize the confusion matrix for the last fold
65 cm = confusion_matrices[-1]
66 plt.figure(figsize=(8, 6))
67 sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
68             xticklabels=clf.classes_,
69             yticklabels=clf.classes_)
70 plt.xlabel('Predicted')
71 plt.ylabel('Actual')
72 plt.title('Confusion Matrix for Last Fold')
73 plt.show()
```

This section outputs the results of the machine learning model

Challenge faced:

- Aggregating the results from multiple validation folds to understand the model's performance.
- Visualizing the model's performance for easier interpretation.

Solution:

- Calculated the mean accuracy, used classification report, utilized a heatmap to represent the confusion matrix,

Results

Average Accuracy: 0.86

Classification Report for last fold:

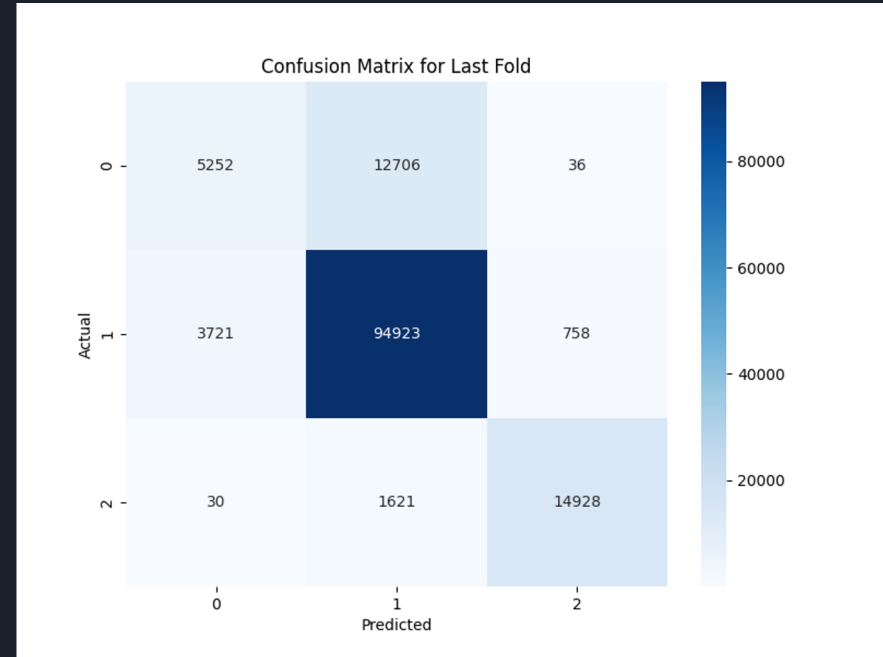
	precision	recall	f1-score	support
0	0.58	0.29	0.39	17994
1	0.87	0.95	0.91	99402
2	0.95	0.90	0.92	16579
accuracy			0.86	133975
macro avg	0.80	0.72	0.74	133975
weighted avg	0.84	0.86	0.84	133975

□

Here we have the accuracy table and the confusion matrix.

Accuracy Table: Evaluating the performance of the classification model.

Confusion Matrix: Summary of predictions against actual values.





Conclusion: Did it work?

- Project Goal: Developing a machine learning model that could assess password strength
- Summary: average accuracy of 0.86; high level of precision in classifying password strengths
- Analysis: Robust performance for medium and strong passwords
- Passwords: first line of defense against unauthorized access
- Relevance: traditional methods rely heavily on predefined rules and heuristics
- Cyber attackers use rainbow tables, dictionary attack and social engineering nowadays
- Using actual data, the model maybe can adapt to changing tactics of attackers
- Cybersecurity needs to be more proactive, preemptive and dynamic to respond to new threats



Thanks!!

