Predict Password Strength Using Natural Language Processing

(June -July 2021)

Submitted by:-

Shivansh Garg

INTRODUCTION

Abstaract:-

In nowadays password is a very vital component in living, so to ensure that password is strong and strength full, so we will have to check that, so in this project, I'm using a machine learning algorithm and a few data analyses on the data site and make the model that gives the output as the strength of the password or it gives how strong the password is.

Introduction;-

Password strength prediction is very good for making you safe and secure. There are many areas where the password is to be confidential and unique. In this project, I use various types of data analytics techniques and various machine learning techniques that help to predict password strength.

In this project, I use approx. 6 lakh password datasets to ensure that my model is to be trained perfectly to any scenario because password strength is very necessary in the digital world.

This project is sub-divided into the following sections:

1. Importing necessary libraries of python.
2. Importing password strength data sets from a CSV file.
3. Then we checked that value in data sets
4. Data pre-processing
5. Use of TF-IDF Vactorizer on data
6. Split the data into train and test sets.
7. Apply logistic regression on data as a use case in classification
8. Check the accuracy of the model
9. Report creation of the model.

Dataset

The source of the dataset is KAGGLE. The password in the dataset, that we analyzed are from 000 WebHost leak, which is available online.

This file contains the passwords with their strengths based on the commercial password strength meter.

The commercial password strength algorithms I used are of Twitter, Microsoft, and battle. How is this algorithm different from these strength meters? First of all, it is entirely based on machine learning rather than on rules. Secondly, I only kept those passwords that were flagged weak, medium, and strong by all three strength meters. This means that all the passwords were indeed either weak, medium, or strong.

It has a total of 3 million passwords but after taking the intersection of all classifications of commercial meters, I was left with 0.7 million passwords. The reduction was because I only used passwords that were flagged in a particular category by all three algorithms.

Dataset URL: https://www.kaggle.com/bhavikbb/password-strength-classifier-dataset

Colors :

1. Passwords:- contains numerous types of passwords.
2. Strength:- it shows the strength of the password according to the commercial password strength algorithms.

METHODOLOGY, ANALYSIS, AND RESULTS

• Data Pre-processing -

The final data fed to our models have to be created by cleaning the dataset. From observing our dataset, we find that our pollutants have NaN concentration values. We will remove these values.

code to check all the missing values in my dataset

```
In [38]: data.isna().sum()
Out[38]: password    1
         strength    0
         dtype: int64

In [39]: data[data['password'].isnull()]
Out[39]:
```

	password	strength
367579	NaN	0

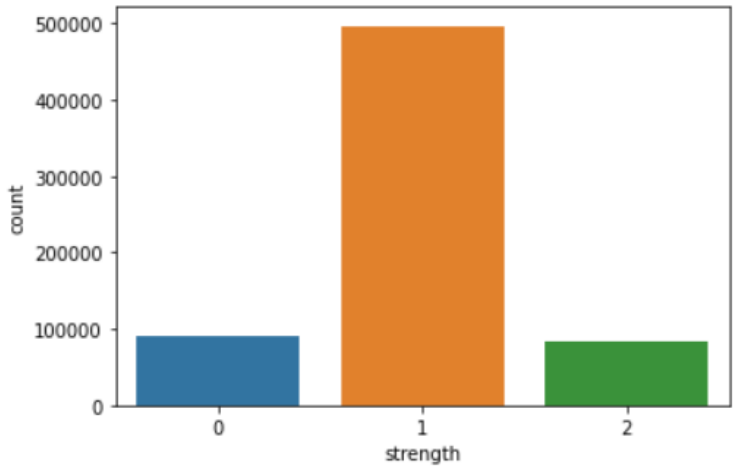
```
In [40]: data.dropna(inplace=True)

In [41]: data.isnull().sum()
Out[41]: password    0
         strength    0
         dtype: int64
```

Data Visualization -

We first plot the password strength plot to find the no of the level of passwords.

```
In [42]: sns.countplot(data['strength'])
Out[42]: <AxesSubplot:xlabel='strength', ylabel='count'>
```



Here, we see that the 1 strength password has very high occurrences in the dataset.

Data Manipulation

Then, we convert the data in the form of a list to further analyze it it will make the dependent list.

And then separate them in their labels

```
In [46]: x=[labels[0] for labels in password_tuple]
         y=[labels[1] for labels in password_tuple]

In [47]: x
```

```
'universe2908',
'as326159',
'lamborghin1',
'kzde5577',
'AVYq1lDE4MgAZfNt',
'AVYq1lDE4MgAZfNt',
'AVYq1lDE4MgAZfNt',
'6975038lp',
'6975038lp',
'asv5o9yu',
'klara-tershina3H',
'kzde5577',
'612035180tok',
'lamborghin1',
'jytifok873',
'kzde5577',
'asv5o9yu',
'idofo673',
'u6c8vhow',
'g067057895',
```

Then, we convert the password into characters because machine learning Algo cant take a string as an input so we write the function to break it into characters.

create a custom function to split input into characters of list

```
In [48]: def word_divide_char(inputs):
             character=[]
             for i in inputs:
                 character.append(i)
             return character

In [49]: word_divide_char('kzde5577')
Out[49]: ['k', 'z', 'd', 'e', '5', '5', '7', '7']
```

Machine LEarning algorithm

import TF-IDF vectorizer to convert String data into numerical data

```
In [50]: from sklearn.feature_extraction.text import TfidfVectorizer

In [51]: vectorizer=TfidfVectorizer(tokenizer=word_divide_char)
```

apply TF-IDF vectorizer on data

```
In [52]: X=vectorizer.fit_transform(x)

In [53]: X.shape
Out[53]: (669639, 128)

In [54]: vectorizer.get_feature_names()
Out[54]: ['\x05',
          '\x06',
          '\x08',
          '\x10',
          '\x16',
          '\x17',
          '\x19',
          '\x1b',
          '\x1c',
          '\x1e',
          ' ',
          '!',
          '"',
          '#',
          '$',
```

Apply logistic regression on data

Apply Logistic on data as use-cas is Classification

```
In [62]: clf=LogisticRegression(random_state=0,multi_class='multinomial')

In [63]: clf.fit(X_train,y_train)
Out[63]: LogisticRegression(multi_class='multinomial', random_state=0)

In [ ]:
```

doing prediction for specific custom data

```
In [64]: dt=np.array(['%@123abcd'])
         pred=vectorizer.transform(dt)
         clf.predict(pred)
Out[64]: array([2])

In [ ]:
```

Then we check the accuracy of the model at last

doing prediction on X-Test data

In [65]:
```
y_pred=clf.predict(X_test)
y_pred
```

Out[65]:
```
array([1, 1, 1, ..., 2, 1, 2])
```

In [ ]:
```

```

check Accuracy of your model using confusion_matrix,accuracy_score

In [66]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
```

In [67]:
```
cm=confusion_matrix(y_test,y_pred)
print(cm)
print(accuracy_score(y_test,y_pred))
```

```
[[ 5387 12562    12]
 [ 3776 92913  2646]
 [   42  5084 11506]]
0.8198882981900723
```

CONCLUSIONS

1. We used different Python libraries like Pandas, NumPy, etc. to manipulate and analyze data.
2. This model predicts the password strong various ml algo and data analytics.

REFERENCES

1.https://www.javatpoint.com/k-nearest-neighboralgorithm-for-machine-learning

2.https://www.javatpoint.com/machine-learning-randomforest-algorithm

3.https://www.geeksforgeeks.org/linear-regressionpython-implementation/

4. https://datascience.stackexchange.com/questions/9228/ decision-tree-vs-knn

5. https://forum.airnowtech.org/t/the-aqi-equation/169

6.https://medium.com/@yrnigam/how-to-write-a-datascience-report-181bd49d8f4d

7.https://app.cpcbccr.com/ccr_docs/How_AQI_Calculated