Predicting Password Strength

Importing Libraries

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

Import Dataset

In [2]:
```
# Lines with too many fields will by default cause an exception to be raised, and no DataFrame will be returned.
# These "bad lines" will be dropped from the DataFrame that is returned.

data=pd.read_csv('data.csv',error_bad_lines=False)
data.head()
```

```
b'Skipping line 2810: expected 2 fields, saw 5\nSkipping line 4641: expected 2 fields, saw 5\nSkipping line 7171: expected 2 fields, saw 5\nSkipping line 11220: expected 2 fields, saw 5\nSkipping line 13809: expected 2 fields, saw 5\nSkipping line 14132: expected 2 fields, saw 5\nSkipping line 14293: expected 2 fields, saw 5\nSkipping line 14865: expected 2 fields, saw 5\nSkipping line 17419: expected 2 fields, saw 5\nSkipping line 22801: expected 2 fields, saw 5\nSkipping line 25001: expected 2 fields, saw 5\nSkipping line 26603: expected 2 fields, saw 5\nSkipping line 26742: expected 2 fields, saw 5\nSkipping line 29702: expected 2 fields, saw 5\nSkipping line 32767: expected 2 fields, saw 5\nSkipping line 32878: expected 2 fields, saw 5\nSkipping line 35643: expected 2 fields, saw 5\nSkipping line 36550: expected 2 fields, saw 5\nSkipping line 38732: expected 2 fields, saw 5\nSkipping line 40567: expected 2 fields, saw 5\nSkipping line 40576: expected 2 fields, saw 5\nSkipping line 41864: expected 2 fields, saw 5\nSkipping line 46861: expected 2 fields, saw 5\nSkipping line 47939: expected 2 fields, saw 5\nSkipping line 48628: expected 2 fields, saw 5\nSkipping line 48908: expected 2 fields, saw 5\nSkipping line 57582: expected 2 fields, saw 5\nSkipping line 58782: expected 2 fields, saw 5\nSkipping line 58984: expected 2 fields, saw 5\nSkipping line 61518: expected 2 fields, saw 5\nSkipping line 63451: expected 2 fields, saw 5\nSkipping line 68141: expected 2 fields, saw 5\nSkipping line 72083: expected 2 fields, saw 5\nSkipping line 74027: expected 2 fields, saw 5\nSkipping line 77811: expected 2 fields, saw 5\nSkipping line 83958: expected 2 fields, saw 5\nSkipping line 85295: expected 2 fields, saw 5\nSkipping line 88665: expected 2 fields, saw 5\nSkipping line 89198: expected 2 fields, saw 5\nSkipping line 92499: expected 2 fields, saw 5\nSkipping line 92751: expected 2 fields, saw 5\nSkipping line 93689: expected 2 fields, saw 5\nSkipping line 94776: expected 2 fields, saw 5\nSkipping line 97334: expected 2 fields, saw 5\nSkipping line 102316: expected 2 fields, saw 5\nSkipping line 103421: expected 2 fields, saw 5\nSkipping line 106872: expected 2 fields, saw 5\nSkipping line 109363: expected 2 fields, saw 5\nSkipping line 110117: expected 2 fields, saw 5\nSkipping line 110465: expected 2 fields, saw 5\nSkipping line 113843: expected 2 fields, saw 5\nSkipping line 115634: expected 2 fields, saw 5\nSkipping line 121518: expected 2 fields, saw 5\nSkipping line 123692: expected 2 fields, saw 5\nSkipping line 124708: expected 2 fields, saw 5\nSkipping line 129608: expected 2 fields, saw 5\nSkipping line 133176: expected 2 fields, saw 5\nSkipping line 135532: expected 2 fields, saw 5\nSkipping line 138042: expected 2 fields, saw 5\nSkipping line 139485: expected 2 fields, saw 5\nSkipping line 140401: expected 2 fields, saw 5\nSkipping line 144093: expected 2 fields, saw 5\nSkipping line 149850: expected 2 fields, saw 5\nSkipping line 151831: expected 2 fields, saw 5\nSkipping line 158014: expected 2 fields, saw 5\nSkipping line 162047: expected 2 fields, saw 5\nSkipping line 164515: expected 2 fields, saw 5\nSkipping line 170313: expected 2 fields, saw 5\nSkipping line 171325: expected 2 fields, saw 5\nSkipping line 171424: expected 2 fields, saw 5\nSkipping line 175920: expected 2 fields, saw 5\nSkipping line 176210: expected 2 fields, saw 5\nSkipping line 183603: expected 2 fields, saw 5\nSkipping line 190264: expected 2 fields, saw 5\nSkipping line 191683: expected 2 fields, saw 5\nSkipping line 191988: expected 2 fields, saw 5\nSkipping line 195450: expected 2 fields, saw 5\nSkipping line 195754: expected 2 fields, saw 5\nSkipping line 197124: expected 2 fields, saw 5\nSkipping line 199263: expected 2 fields, saw 5\nSkipping line 202603: expected 2 fields, saw 5\nSkipping line 209960: expected 2 fields, saw 5\nSkipping line 213218: expected 2 fields, saw 5\nSkipping line 217060: expected 2 fields, saw 5\nSkipping line 220121: expected 2 fields, saw 5\nSkipping line 223518: expected 2 fields, saw 5\nSkipping line 226293: expected 2 fields, saw 5\nSkipping line 227035: expected 2 fields, saw 7\nSkipping line 227341: expected 2 fields, saw 5\nSkipping line 227808: expected 2 fields, saw 5\nSkipping line 228516: expected 2 fields, saw 5\nSkipping line 228733: expected 2 fields, saw 5\nSkipping line 232043: expected 2 fields, saw 5\nSkipping line 232426: expected 2 fields, saw 5\nSkipping line 234490: expected 2 fields, saw 5\nSkipping line 239626: expected 2 fields, saw 5\nSkipping line 240461: expected 2 fields, saw 5\nSkipping line 244518: expected 2 fields, saw 5\nSkipping line 245395: expected 2 fields, saw 5\nSkipping line 246168: expected 2 fields, saw 5\nSkipping line 246655: expected 2 fields, saw 5\nSkipping line 246752: expected 2 fields, saw 5\nSkipping line 247189: expected 2 fields, saw 5\nSkipping line 250276: expected 2 fields, saw 5\nSkipping line 255327: expected 2 fields, saw 5\nSkipping line 257094: expected 2 fields, saw 5\n'
b'Skipping line 264626: expected 2 fields, saw 5\nSkipping line 265028: expected 2 fields, saw 5\nSkipping line 269150: expected 2 fields, saw 5\nSkipping line 271360: expected 2 fields, saw 5\nSkipping line 273975: expected 2 fields, saw 5\nSkipping line 274742: expected 2 fields, saw 5\nSkipping line 276227: expected 2 fields, saw 5\nSkipping line 279807: expected 2 fields, saw 5\nSkipping line 283425: expected 2 fields, saw 5\nSkipping line 287468: expected 2 fields, saw 5\nSkipping line 292995: expected 2 fields, saw 5\nSkipping line 293496: expected 2 fields, saw 5\nSkipping line 293735: expected 2 fields, saw 5\nSkipping line 295060: expected 2 fields, saw 5\nSkipping line 296643: expected 2 fields, saw 5\nSkipping line 296848: expected 2 fields, saw 5\nSkipping line 308926: expected 2 fields, saw 5\nSkipping line 310360: expected 2 fields, saw 5\nSkipping line 317004: expected 2 fields, saw 5\nSkipping line 318207: expected 2 fields, saw 5\nSkipping line 331783: expected 2 fields, saw 5\nSkipping line 333864: expected 2 fields, saw 5\nSkipping line 335958: expected 2 fields, saw 5\nSkipping line 336290: expected 2 fields, saw 5\nSkipping line 343526: expected 2 fields, saw 5\nSkipping line 343857: expected 2 fields, saw 5\nSkipping line 344059: expected 2 fields, saw 5\nSkipping line 348691: expected 2 fields, saw 5\nSkipping line 353446: expected 2 fields, saw 5\nSkipping line 357073: expected 2 fields, saw 5\nSkipping line 359753: expected 2 fields, saw 5\nSkipping line 359974: expected 2 fields, saw 5\nSkipping line 366534: expected 2 fields, saw 5\nSkipping line 369514: expected 2 fields, saw 5\nSkipping line 377759: expected 2 fields, saw 5\nSkipping line 379327: expected 2 fields, saw 5\nSkipping line 380769: expected 2 fields, saw 5\nSkipping line 381073: expected 2 fields, saw 5\nSkipping line 381489: expected 2 fields, saw 5\nSkipping line 386304: expected 2 fields, saw 5\nSkipping line 387635: expected 2 fields, saw 5\nSkipping line 389613: expected 2 fields, saw 5\nSkipping line 392604: expected 2 fields, saw 5\nSkipping line 393184: expected 2 fields, saw 5\nSkipping line 395530: expected 2 fields, saw 5\nSkipping line 396939: expected 2 fields, saw 5\nSkipping line 397385: expected 2 fields, saw 5\nSkipping line 397509: expected 2 fields, saw 5\nSkipping line 402902: expected 2 fields, saw 5\nSkipping line 405187: expected 2 fields, saw 5\nSkipping line 408412: expected 2 fields, saw 5\nSkipping line 419423: expected 2 fields, saw 5\nSkipping line 420962: expected 2 fields, saw 5\nSkipping line 425965: expected 2 fields, saw 5\nSkipping line 427496: expected 2 fields, saw 5\nSkipping line 438881: expected 2 fields, saw 5\nSkipping line 439776: expected 2 fields, saw 5\nSkipping line 440345: expected 2 fields, saw 5\nSkipping line 445507: expected 2 fields, saw 5\nSkipping line 445548: expected 2 fields, saw 5\nSkipping line 447184: expected 2 fields, saw 5\nSkipping line 448603: expected 2 fields, saw 5\nSkipping line 451732: expected 2 fields, saw 5\nSkipping line 458249: expected 2 fields, saw 5\nSkipping line 460274: expected 2 fields, saw 5\nSkipping line 467630: expected 2 fields, saw 5\nSkipping line 473961: expected 2 fields, saw 5\nSkipping line 476281: expected 2 fields, saw 5\nSkipping line 478010: expected 2 fields, saw 5\nSkipping line 478322: expected 2 fields, saw 5\nSkipping line 479999: expected 2 fields, saw 5\nSkipping line 480898: expected 2 fields, saw 5\nSkipping line 481688: expected 2 fields, saw 5\nSkipping line 485193: expected 2 fields, saw 5\nSkipping line 485519: expected 2 fields, saw 5\nSkipping line 486000: expected 2 fields, saw 5\nSkipping line 489063: expected 2 fields, saw 5\nSkipping line 494525: expected 2 fields, saw 5\nSkipping line 495009: expected 2 fields, saw 5\nSkipping line 501954: expected 2 fields, saw 5\nSkipping line 508035: expected 2 fields, saw 5\nSkipping line 508828: expected 2 fields, saw 5\nSkipping line 509833: expected 2 fields, saw 5\nSkipping line 510410: expected 2 fields, saw 5\nSkipping line 518229: expected 2 fields, saw 5\nSkipping line 520302: expected 2 fields, saw 5\nSkipping line 520340: expected 2 fields, saw 5\n'
b'Skipping line 525174: expected 2 fields, saw 5\nSkipping line 526251: expected 2 fields, saw 5\nSkipping line 529611: expected 2 fields, saw 5\nSkipping line 531398: expected 2 fields, saw 5\nSkipping line 534146: expected 2 fields, saw 5\nSkipping line 544954: expected 2 fields, saw 5\nSkipping line 553002: expected 2 fields, saw 5\nSkipping line 553883: expected 2 fields, saw 5\nSkipping line 553887: expected 2 fields, saw 5\nSkipping line 553915: expected 2 fields, saw 5\nSkipping line 554172: expected 2 fields, saw 5\nSkipping line 563534: expected 2 fields, saw 5\nSkipping line 565191: expected 2 fields, saw 5\nSkipping line 574108: expected 2 fields, saw 5\nSkipping line 574412: expected 2 fields, saw 5\nSkipping line 575985: expected 2 fields, saw 5\nSkipping line 580091: expected 2 fields, saw 5\nSkipping line 582682: expected 2 fields, saw 5\nSkipping line 585885: expected 2 fields, saw 5\nSkipping line 590171: expected 2 fields, saw 5\nSkipping line 591924: expected 2 fields, saw 5\nSkipping line 592515: expected 2 fields, saw 5\nSkipping line 593888: expected 2 fields, saw 5\nSkipping line 596245: expected 2 fields, saw 5\nSkipping line 607344: expected 2 fields, saw 5\nSkipping line 607633: expected 2 fields, saw 5\nSkipping line 610939: expected 2 fields, saw 5\nSkipping line 613638: expected 2 fields, saw 5\nSkipping line 615643: expected 2 fields, saw 5\nSkipping line 615901: expected 2 fields, saw 5\nSkipping line 617389: expected 2 fields, saw 5\nSkipping line 634641: expected 2 fields, saw 5\nSkipping line 635755: expected 2 fields, saw 5\nSkipping line 646243: expected 2 fields, saw 5\nSkipping line 647165: expected 2 fields, saw 5\nSkipping line 648610: expected 2 fields, saw 5\nSkipping line 648772: expected 2 fields, saw 5\nSkipping line 651833: expected 2 fields, saw 5\nSkipping line 653663: expected 2 fields, saw 5\nSkipping line 656233: expected 2 fields, saw 5\nSkipping line 656694: expected 2 fields, saw 5\nSkipping line 659783: expected 2 fields, saw 5\nSkipping line 660478: expected 2 fields, saw 5\nSkipping line 661133: expected 2 fields, saw 5\nSkipping line 661736: expected 2 fields, saw 5\nSkipping line 669827: expected 2 fields, saw 5\n'
```

Out[2]:

	password	strength
0	kzde5577	1
1	kino3434	1
2	visi7k1yr	1
3	megzy123	1
4	lamborghin1	1

In [3]:
```
# Unique values in strength column i.e., target variable

data['strength'].unique()
```

Out[3]:
```
array([1, 2, 0])
```

Password Strength

0 : Bad

1 : Average

2 : Good

In [4]:
```
# Check for any missing value

data.isna().sum()
```

Out[4]:
```
password    1
strength    0
dtype: int64
```

In [5]:
```
# Missing value

data[data['password'].isnull()]
```

Out[5]:

	password	strength
367579	NaN	0

In [6]:
```
# Drop the rows with missing values

data.dropna(inplace=True)
```

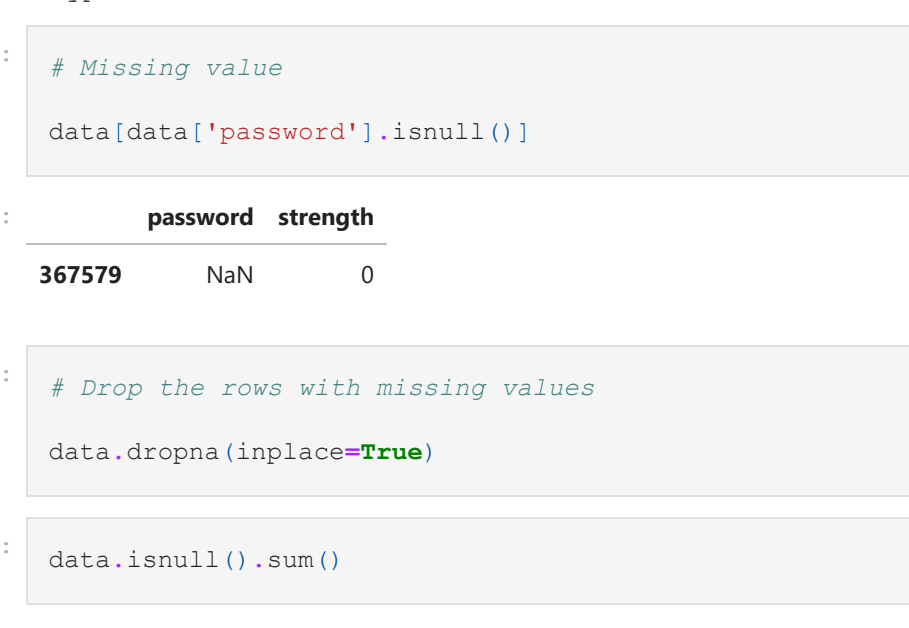
In [7]:
```
data.isnull().sum()
```

Out[7]:
```
password    0
strength    0
dtype: int64
```

In [8]:
```
# Check distribution of target variable

sns.countplot(data['strength'])
```

Out[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff038b3fad0>
```



Observation:

1. Majority of the password strength is 1 i.e., Average.
2. It is an imbalanced dataset

Seperate dependent and independent variables

In [9]:
```
# Converting DataFrame to numpy array

password_tuple = np.array(data)
```

In [10]:
```
password_tuple
```

Out[10]:
```
array([['kzde5577', 1],
       ['kino3434', 1],
       ['visi7k1yr', 1],
       ...,
       ['184520socram', 1],
       ['marken22a', 1],
       ['fxx4pw4g', 1]], dtype=object)
```

In [11]:
```
# Shuffling randomly for robustness in my model
import random

random.shuffle(password_tuple)
```

In [12]:
```
x=[labels[0] for labels in password_tuple]
y=[labels[1] for labels in password_tuple]
```

In [13]:
```
x
```

```
 1,
 0,
 1,
 1,
 1,
 1,
 1,
 0,
 1,
 1,
 0,
 1,
 2,
 0,
 1,
 1,
 1,
 1,
 1,
 0,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 1,
 0,
 0,
 1,
 1,
 0,
 1,
 ...]
```

Create a custom function to split input into characters of list

In [15]:
```python
def word_divide_char(inputs):
    character = []
    for char in inputs:
        character.append(char)
    return character
```

In [16]:
```python
# Example to explain working of custom function i.e., tokenizer

word_divide_char('kzde5577')
```

Out[16]: `['k', 'z', 'd', 'e', '5', '5', '7', '7']`

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

In [17]:
```python
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [18]:
```python
vectorizer = TfidfVectorizer(tokenizer = word_divide_char)
```

Apply TF-IDF vectorizer on data

In [19]:
```python
X = vectorizer.fit_transform(x)
```

In [20]:
```python
X.shape
```

Out[20]: `(669639, 133)`

After applying TF-IDF vectorizer we now have 128 unique features.

In [21]:
```python
vectorizer.get_feature_names()
```

Out[21]:
```
['\x01',
 '\x04',
 '\x05',
 '\x06',
 '\x08',
 '\x0e',
 '\x10',
 '\x11',
 '\x12',
 '\x16',
 '\x17',
 '\x19',
 '\x1b',
 '\x1c',
 '\x1e',
 ' ',
 '!',
 '"',
 '#',
 '$',
 '%',
 '&',
 '(',
 ')',
 '*',
 '+',
 '-',
 '.',
 '/',
 '0',
 '1',
 '2',
 '3',
 '4',
 '5',
 '6',
 '7',
 '8',
 '9',
 ';',
 '<',
 '=',
 '>',
 '?',
 '@',
 '[',
 '\\',
 ']',
 '^',
 '_',
 '`',
 'a',
 'b',
 'c',
 'd',
 'e',
 'f',
 'g',
 'h',
 'i',
 'j',
 'k',
 'l',
 'm',
 'n',
 'o',
 'p',
 'q',
 'r',
 's',
 't',
 'u',
 'v',
 'w',
 'x',
 'y',
 'z',
 '{',
 '|',
 '}',
 '~',
 '\x7f',
 '\x8d',
 '\xa0',
 '¡',
 '¤',
 '¨',
 '«',
 '¯',
 '°',
 '±',
 '²',
 '³',
 'µ',
 '·',
 'º',
 '¼',
 '¾',
 '¿',
 '×',
 'ß',
 'à',
 'á',
 'â',
 'ä',
 'å',
 'æ',
 'ç',
 'è',
 'é',
 'ê',
 'í',
 'î',
 'ð',
 'ñ',
 'ò',
 'ó',
 'ô',
 'õ',
 '÷',
 'ú',
 'û',
 'ü',
 'ý',
 'þ',
 'ÿ',
 'œ',
 '–',
 '‚',
 '‡',
 '‹',
 '›',
 '™']
```

In [22]:
```python
# Object of Sparse matrix
# the vector returned by tfidf vectorizer contains a large number of zero-valued elements
# sparse matrices only store non-zero elements. It storing non-zero elements with triples - (Row, Column, valu
# By using sparse matrix we save a significant amount of memory and speed up the processing of that data

first_document_vector = X[0]
first_document_vector
```

Out[22]:
```
<1x133 sparse matrix of type '<class 'numpy.float64'>'
	with 6 stored elements in Compressed Sparse Row format>
```

In [23]:
```python
# Transforming sparse to dense representation

first_document_vector.T.todense()
```

Out[23]:
```
matrix([[0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.56693833],
        [0.        ],
        [0.59141273],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.28580163],
        [0.22132987],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.29203803],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.33594081],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ],
        [0.        ]])
```

In [24]:
```python
# Create DataFrame with TF_IDF values as columns and feature names as index

df = pd.DataFrame(first_document_vector.T.todense(), index=vectorizer.get_feature_names(),columns=["TF-IDF"])
df.sort_values(by=["TF-IDF"],ascending=False)
```

Out[24]:

	TF-IDF
7	0.591413
5	0.566938
z	0.335941
k	0.292038
d	0.285802
...	...
;	0.000000
9	0.000000
8	0.000000
6	0.000000
™	0.000000

133 rows × 1 columns

Split Data into Train and Test set

```
train ---> To learn the relationship within data.
test -->  To do predictions, and this testing data will be unseen to my model.
```

In [25]:
```python
from sklearn.model_selection import train_test_split
```

In [26]:
```python
# Split into 80% Training data and 20% Testing data

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2)
```

In [27]:
```python
X_train.shape
```

Out[27]: `(535711, 133)`

Model

Logistic Regression

In [28]:
```python
from sklearn.linear_model import LogisticRegression
```

In [29]:
```python
model_lr = LogisticRegression(random_state=100, multi_class='multinomial')
```

In [30]:
```python
model_lr.fit(X_train,y_train)
```

Out[30]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='multinomial', n_jobs=None, penalty='l2',
                   random_state=100, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

Prediction for specific custom data

In [151…
```python
password = input('Enter which password you want to predict: ')
```

```
Enter which password you want to predict: asap121@16
```

In [150…
```python
def predict(model, password):
    # Convert str to numpy array
    password_pred = np.array([password])
    # apply TF-IDF vectorizer which has been fit with train data
    pred = vectorizer.transform(password_pred)
    # predict strength of the password
    y_predicted = model.predict(pred)
    # Print strength of the password
    if y_predicted == np.array([0]):
        print('Your password "{}" is weak.'.format(password))
    elif y_predicted == np.array([1]):
        print('Your password "{}" is Medium.'.format(password))
    else:
        print('Your password "{}" is Strong.'.format(password))
```

In [152…
```python
predict(model_lr, password)
```

```
Your password "asap121@16" is Medium.
```

Checking Performance of your model using confusion_matrix and accuracy_score

In [34]:
```python
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
```

In [130…
```python
y_predicted = model_lr.predict(X_test)
y_predicted
```

Out[130…: `array([1, 1, 1, ..., 1, 1, 1])`

In [131…
```python
cm = confusion_matrix(y_test,y_predicted)
print(cm)
print("\nAccuracy:",accuracy_score(y_test,y_predicted))
```

```
[[ 5479 12510    15]
 [ 3888 92964  2505]
 [   40  5106 11421]]

Accuracy: 0.820321366704498
```

In [133…
```python
class_acc = list(cm.diagonal()/cm.sum(axis=1))
print("Class wise accuracy: ")
for i in range(len(class_acc)):
    print(i, class_acc[i])
```

```
Class wise accuracy:
0 0.3043212619417907
1 0.9356562698149099
2 0.6893825073942175
```

In [132…
```python
# Create report of yout model

print(classification_report(y_test,y_predicted))
```

```
              precision    recall  f1-score   support

           0       0.58      0.30      0.40     18004
           1       0.84      0.94      0.89     99357
           2       0.82      0.69      0.75     16567

    accuracy                           0.82    133928
   macro avg       0.75      0.64      0.68    133928
weighted avg       0.80      0.82      0.80    133928
```

Random Forest Classifier

In [135…
```python
from sklearn.ensemble import RandomForestClassifier
```

In [134…
```python
clf = RandomForestClassifier(n_jobs=-1)
clf.fit(X_train, y_train)
y_predicted = clf.predict(X_test)
```

In [136…
```python
cm = confusion_matrix(y_test,y_predicted)
print(cm)
print("\nAccuracy:",accuracy_score(y_test,y_predicted))
```

```
[[16896  1106     2]
 [  275 98905   177]
 [   16   472 16079]]

Accuracy: 0.9847082014216594
```

In [138…
```python
class_acc = list(cm.diagonal()/cm.sum(axis=1))
print("Class wise accuracy: ")
for i in range(len(class_acc)):
    print(i, class_acc[i])
```

```
Class wise accuracy:
0 0.938458120417685
1 0.9954507483116438
2 0.9705438522363735
```

In [137…
```python
print(classification_report(y_test,y_predicted))
```

```
              precision    recall  f1-score   support

           0       0.98      0.94      0.96     18004
           1       0.98      1.00      0.99     99357
           2       0.99      0.97      0.98     16567

    accuracy                           0.98    133928
   macro avg       0.99      0.97      0.98    133928
weighted avg       0.98      0.98      0.98    133928
```