

“PASSWORD STRENGTH PREDICTION USING MACHINE LEARNING”

Minor Project Report

*Submitted in Partial Fulfillment of the
Requirements for the Degree of*

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

By

**Parth Parikh
(20BEC081)**



**Department of Electronics and Communication Engineering,
Institute of Technology,
Nirma University,
Ahmedabad 382 481**

December 2023

CERTIFICATE

This is to certify that the Minor Project Report entitled “Password Strength Prediction using Machine Learning” submitted by Ms Mrudani Vishal Hada (20BEC071) towards the partial fulfillment of the requirements for the award of degree in Bachelor of Technology in the field of Electronics & Communication Engineering of Nirma University is the record of work carried out by her under our supervision and guidance. The work submitted has in our opinion reached a level required for being accepted for examination. The results embodied in this minor project work to the best of our knowledge have not been submitted to any other University or Institution for award of any degree or diploma.

Name of the Guide**(Amisha P. Naik)**

Department of Electronics &
Communication Engineering,
Institute of Technology,
Nirma University,
Ahmedabad 382 481

Head of Department

Department of Electronics &
Communication Engineering,
Institute of Technology,
Nirma University,
Ahmedabad 382 481

Date: 2/12/23

CERTIFICATE

This is to certify that the Minor Project Report entitled “Password Strength Prediction using Machine Learning” submitted by Mr Parth Parikh (20BEC081) towards the partial fulfillment of the requirements for the award of degree in Bachelor of Technology in the field of Electronics & Communication Engineering of Nirma University is the record of work carried out by him under our supervision and guidance. The work submitted has in our opinion reached a level required for being accepted for examination. The results embodied in this minor project work to the best of our knowledge have not been submitted to any other University or Institution for award of any degree or diploma.

Name of the Guide**(Amisha P. Naik)**

Department of Electronics &
Communication Engineering,
Institute of Technology,
Nirma University,
Ahmedabad 382 481

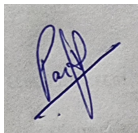
Head of Department

Department of Electronics &
Communication Engineering,
Institute of Technology,
Nirma University,
Ahmedabad 382 481

Date: 2/12/23

Undertaking for Originality of the Work

We, Mrudani Hada & Parth Parikh, Roll Nos 20BEC071 & 20BEC081, give undertaking that the Minor Project entitled “Password Strength Prediction using Machine Learning” submitted by us, towards the partial fulfillment of the requirements for the degree of Bachelor of Technology in Electronics and Communication of Nirma University, Ahmedabad 382 481, is the original work carried out by us and We give assurance that no attempt of plagiarism has been made. We understand that in the event of any similarity found subsequently with any other published work or any project report elsewhere; it will result in severe disciplinary action.



Signature of the Student

Date: 2/12/23

Place: Ahmedabad

Endorsed by:

(Signature of the Guide)

Acknowledgement

A journey is easier when you travel together. Interdependence is certainly more valuable than independence. This thesis is the result of work whereby we have been accompanied and supported by many people. It is a pleasant aspect that we have now the opportunity to express my gratitude for all of them.

With immense pleasure we express my sincere gratitude, regards and thanks to my supervisor Dr. Amisha P. Naik for her excellent guidance, invaluable suggestions and continuous encouragement at all the stages of my research work. Her interest and confidence in us were the reason for all the success we have made. We have been fortunate to have her as our guide as she has been a great influence on us, both as a person and as a professional.

It was a pleasure to be associated with the Software Laboratory of Nirma University, and we would like to thank the entire Laboratory Staff. To acknowledge help taken from friends is always a joy. I take this opportunity to convey sincere thanks to my class fellows for their smiles and friendship making the life at Nirma University enjoyable and memorable.

The chain of our gratitude would be definitely incomplete if we would forget to thank the first cause of this chain, using Aristotle's words, The Prime Mover for showering Her blessings on us always.

Mrudani Hada & Parth Parikh

Abstract

In this era of digitalization, one has to set passwords every now and then. Humans have a tendency to inculcate sensitive personal information while choosing their passwords. A weak password is prone to cyber-attacks as it paves the way for hackers to crack it easily. With ever-increasing security concerns due to unauthorized intrusion into private data it has become necessary to keep strong passwords.

In today's scenario, passwords serve an important mechanism for authentication and protection of devices. Myriads methods such as biometrics including face and fingerprint recognition have been introduced in recent times, but none could completely eliminate the use of passwords. Normal passwords drawback is that a human has to memorize different passwords which proves to be challenging and inconvenient. Password cracking tools are rapidly increasing on the internet thus an organization should ensure strong passwords. For similar reasons, they should include password strength predictors.

Passwords are required for numerous purposes like logging into computer accounts, retrieving email, transferring funds, accessing databases and websites. The need for selecting and using unbreachable passwords are increasing in leaps and bounds. For enhancing security concerns, one should use variety of passwords for different purposes. Users generally choose simple and guessable passwords which can easily be cracked by data thieves and can be misused against non-techosavy masses. Evolution of the internet and e-commerce in the field of information technology has impacted the lives of the people in a great way. Along with it, alarming security issues have come into limelight. The strength of a strong password depends on two main factors namely complexity and length of the password. Complexity is delineated by amalgamation of multiple characters- both in uppercase and lowercase, numbers and special characters. Whereas, the length elucidates count of the characters used for making the password. To minimize the risk of foreign attack, lengthy and diversified password is crucial.

Thus, it is necessary to develop a machine learning approach to scrutinize the strength of the password and to provide a possible protection against the exploitation of vulnerabilities.

INDEX

Chapter No.	Title	Page No.
	Acknowledgement	i
	Abstract	ii
	Index	iii
	List of Figures	iv
	List of Tables	Vi
1	Introduction	1
	1.1 Introduction/ Prologue/Background	1
	1.2 Motivation	2
	1.3 Objective	3
	1.4 Problem Statement	5
	1.5 Approach	6
	1.6 Scope of the Project	10
	1.7 Gantt Chart (Timeline of your work)	11
2	Literature Review	12
	2.1 Paper 1 - Password Strength prediction Using Supervised Machine Learning	12
	2.2 Paper 2 – A Password Strength Evaluation Algorithm based on Sensitive Personal Information	12
	2.3 Paper 3 - Password Strength Analysis and its Classification by Applying Machine Learning Based Techniques	13
3	Software Design	14
4	Experimental/Simulation Results and Discussion	16
5	Conclusions and Future Scope	21
	References	23
	Appendix	

LIST OF FIGURES

Figure No.	Title	Page No.
1.	Data of Time Taken by AI to Crack Password	3
2.	Data Visualization using Bar Chart	4
3.	Data Visualization using Line Chart	4
4.	Flow of Our Approach	6
5.	Logistic Regression Curve	9
6.	Novelty Introduced	9
7.	Time Complexity Comparison of Simple Python Code and ML	16
8.	Reading the Data	16
9.	Classification of Data into Strengths	17
10.	Prediction for MP7_two@111	18
11.	Prediction for the Test Data	18
12.	Confusion Matrix of The Implemented Code	18
13.	Final Report Summary for the ML Approach	19
14.	Comparison Based on Accuracy	19
15.	Comparison Based on Training Time	20

LIST OF TABLES

Table No.	Title	Page No.
1	TF Function	7
2	IDF Function	7

Chapter 1

Introduction

1.1 Prologue

In an era dominated by digital connectivity and information exchange, the safeguarding of personal and sensitive data has become paramount. One of the primary gateways to secure digital environments is through the use of passwords. As technology advances, so do the methods employed by malicious actors to compromise these passwords, highlighting the critical need for robust authentication mechanisms.

This research paper delves into the realm of cybersecurity, focusing on the pivotal role of password strength in fortifying digital defenses. Traditional password policies often fall short in the face of sophisticated attacks, prompting the exploration of innovative approaches. Machine learning, with its ability to discern patterns and adapt to evolving threats, emerges as a promising tool for predicting and enhancing password strength.

The primary objective of this study is to develop a reliable password strength prediction model using machine learning algorithms. By analyzing vast datasets of historical password information and incorporating features that encapsulate various aspects of password complexity, our research aims to create a predictive model capable of evaluating the strength of a given password. This predictive capability not only aids in real-time authentication decisions but also serves as a proactive defense against potential security breaches.

In the following sections, we will explore the current landscape of password security, shedding light on the limitations of traditional approaches and the pressing need for more intelligent solutions. Subsequently, we will delve into the theoretical foundations of machine learning and its application in cybersecurity, providing a comprehensive understanding of how this technology can be harnessed to predict password strength effectively.

By the conclusion of this research, we anticipate contributing valuable insights to the ongoing discourse surrounding cybersecurity, particularly in the context of password protection. The

potential implications of this work extend beyond individual users to encompass organizations, institutions, and entire digital ecosystems, emphasizing the broader significance of bolstering authentication mechanisms in the face of an ever-evolving threat landscape.

1.2 Motivation

The motivation for selecting the topic of "Password Strength Prediction Using Machine Learning" for research stems from the increasingly critical role of passwords as a primary means of securing digital assets and sensitive information. As technology continues to advance, the methods employed by malicious actors to compromise passwords become more sophisticated, necessitating innovative approaches to enhance cybersecurity.

Several key motivations underpin the choice of this research topic such as Ubiquity of Passwords, Human Factor and Password Weakness, Continuous Threat Landscape, Machine Learning's Predictive Power, Real-time Authentication Enhancement, Relevance to Cybersecurity Challenges and Practical Applications.

In summary, the motivation behind researching password strength prediction using machine learning lies in the recognition of passwords as a critical link in the cybersecurity chain and the belief that leveraging machine learning can offer innovative solutions to address the evolving challenges in this domain. The goal is to contribute to the development of more resilient and adaptive authentication mechanisms that align with the contemporary digital landscape.

1.3 Objective

The time taken by AI to crack passwords depends on several factors, including the length of the password, the character set used (letters, numbers, symbols), and the overall complexity of the password.

A password has to be protected through all types of attacks that AI can perform including Brute Force Attacks where it tries systematically every possible combination until the correct password is found, Dictionary Attacks where it tries a predefined list of common passwords or words and some advanced techniques like rainbow tables which are precomputed tables of password hashes.



Fig 1. Data of Time Taken by AI to Crack Passwords

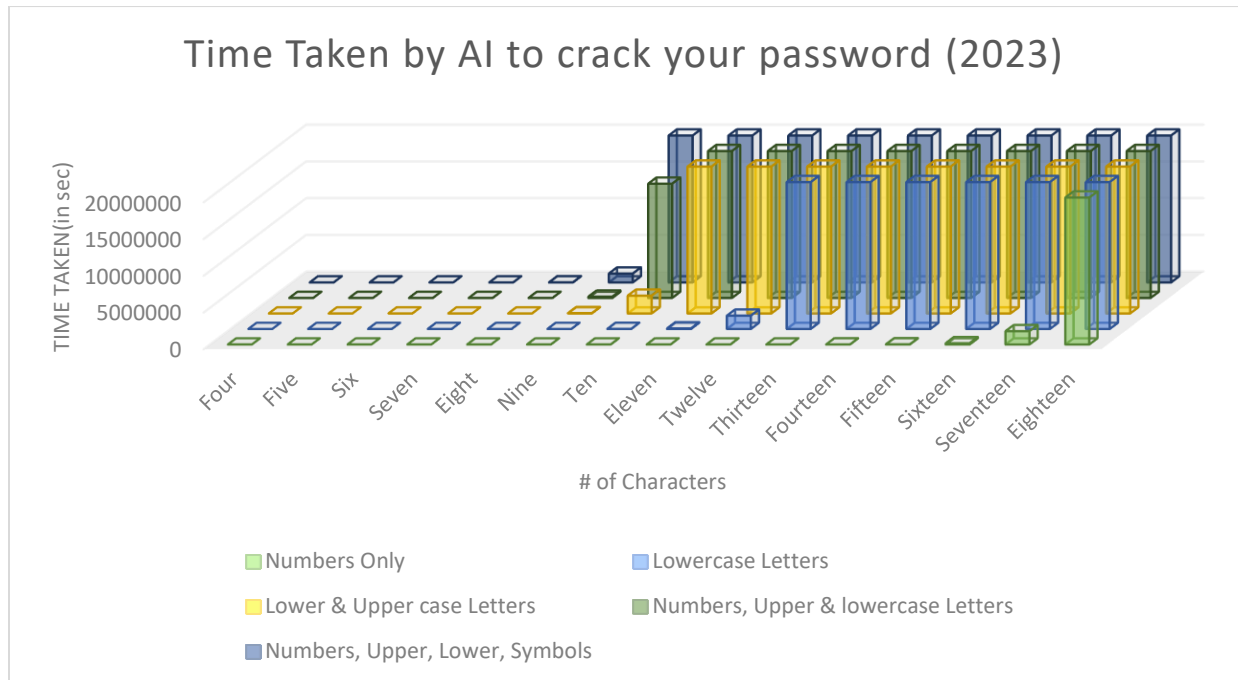


Fig 2. Data Visualization using Bar Chart

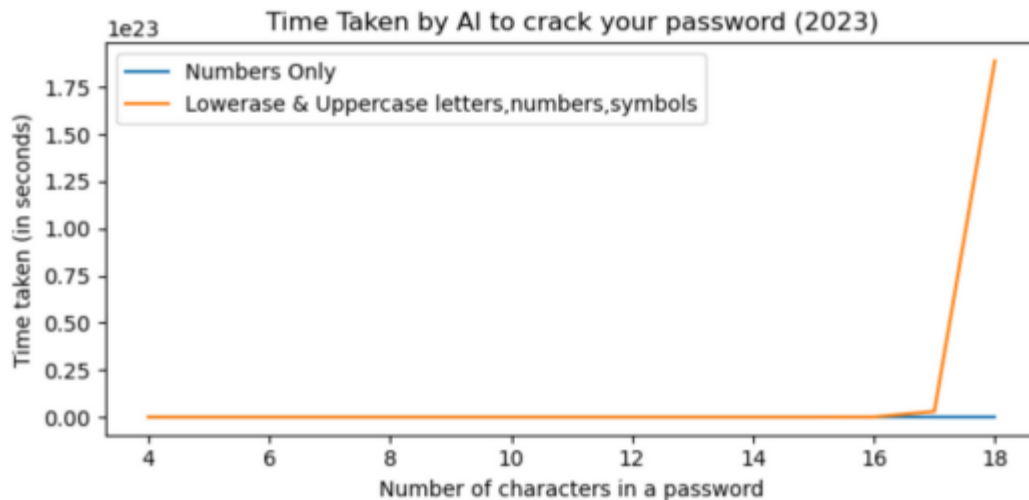


Fig 3. Data Visualization using Line Chart

Thus, for prime applications of digitization such as Email accounts, Online Banking, Workplace systems, e-commerce accounts, government portals, etc. passwords play an important role and hence from the above visualizations it was taken into consideration that longer and more complex passwords take significantly more time to crack than shorter and simpler ones.

1.4 Problem Statement

The problem at hand is the inadequacy of existing password strength assessment mechanisms to proactively adapt to emerging threats. Conventional approaches rely on static rules and heuristics, often unable to contend with the dynamic nature of cyber threats. Moreover, the human factor in password creation introduces a level of unpredictability that necessitates a more intelligent and adaptive solution. Users tend to create passwords that are easily guessable, using familiar words, sequences, or common patterns, thereby weakening the overall security posture.

A normal python code requires if-else statements to execute password correcting measures which needs multiple iterations to run each time. Whereas a machine learning model when trained once, takes negligible amount of time to execute and produce outcomes. Machine learning and deep learning models to automatically extract, classify, and label elements of text data and then assign a statistical likelihood to each possible meaning of those elements for 6,69,640 entries in dataset.

Addressing this problem requires a paradigm shift towards leveraging machine learning algorithms for predictive password strength assessment. Machine learning, with its ability to analyze large datasets, discern complex patterns, and adapt to evolving trends, presents an opportunity to enhance the accuracy and agility of password strength prediction models. The challenge lies in developing a robust machine learning model that not only accurately evaluates the strength of passwords but also adapts to emerging threats in real-time, providing a proactive defense against unauthorized access.

This research aims to bridge the gap in current password strength prediction methodologies by harnessing the power of machine learning. By exploring and implementing advanced algorithms capable of learning from historical password data, the objective is to create a predictive model that elevates the overall security of digital authentication systems. Through this, the research endeavors to contribute to the development of intelligent, adaptive, and resilient password protection mechanisms, addressing the pressing cybersecurity challenges of the digital era.

1.5 Approach

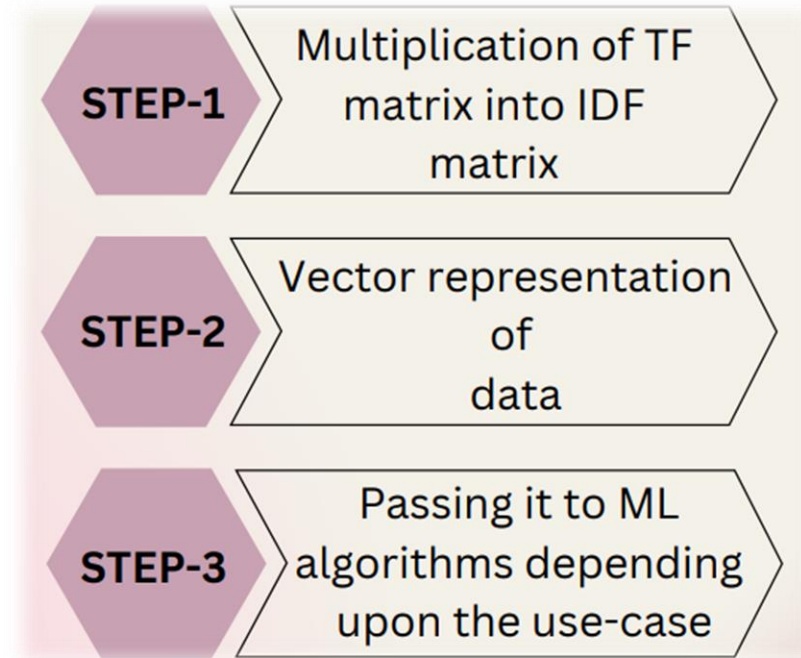


Fig 4. Flow of Our Approach

I. TF - IDF Vectorization:

As the ML model does not understand the string data so this algorithm is used to convert it into numerical format.

The formula for finding the TF (Term Frequency):

$$\frac{\text{no. of times a word occurs in a document}}{\text{total no. of words in that document}}$$

The IDF (Inverse Document Frequency) matrix formula is:

$$\log\left(\frac{\text{total no of documents}}{\text{no. of documents containing that word}}\right)$$

Moreover, in this algorithm it multiplies these two matrices to get the importance of the word in all the documents. Furthermore, this matrix is used after removing all the null values and splitting the data into lists.

Explaining the concept with an example, let's take two doc strings and then make a vector of the available string data as a part of data preprocessing.

DOC 1: Saturday is the first review of minor project

DOC 2: Amisha Ma'am is our minor project guide

For the Term Frequency matrix, the table for the two docs is as follows where the mentioned formula is applied on each and every word present in the two strings.

TF FUNCTION <i>no. of times a word occurs in a document</i> <i>total no. of words in that document</i>		
Words	Doc 1	Doc 2
Saturday	1/8	0
Is	1/8	1/7
The	1/8	0
First	1/8	0
Review	1/8	0
Of	1/8	0
minor	1/8	1/7
Project	1/8	1/7
Amisha	0	1/7
Ma'am	0	1/7
Our	0	1/7
Guide	0	1/7

Table 1. TF Function

For the Inverse Document Frequency matrix, the table is as follows combined into a single column with the formula applied on each word

IDF FUNCTION <i>Log</i> $\left(\frac{\text{Total no. of documents}}{\text{no. of documents containing the word}} \right)$	
Words	IDF
Saturday	Log 2 = 0.301
Is	Log 1 = 0
The	Log 2 = 0.301
First	Log 2 = 0.301
Review	Log 2 = 0.301
Of	Log 2 = 0.301
minor	Log 1 = 0
Project	Log 1 = 0
Amisha	Log 2 = 0.301
Ma'am	Log 2 = 0.301
Our	log 2 = 0.301
Guide	Log 2 = 0.301

Table 2. IDF Function

Now, multiplying the two functions to get a final vector

DOC 1: Saturday is the first review of minor project

$$TF * IDF == \frac{1}{8} * \log 2 + \frac{1}{8} * 0 + \frac{1}{8} * \log 2 + \frac{1}{8} * \log 2 + \frac{1}{8} * \log 2 + \frac{1}{8} * \log 2 + \frac{1}{8} * 0 + \frac{1}{8} * 0 + \frac{5}{8} * \log 2 = 0.188125$$

DOC 2: Amisha Maam is our minor project guide

$$TF * IDF == \frac{1}{7} * \log 2 + \frac{1}{7} * \log 2 + \frac{1}{7} * 0 + \frac{1}{7} * \log 2 + \frac{1}{7} * 0 + \frac{1}{7} * 0 + \frac{1}{7} * \log 2 + \frac{4}{7} * \log 2 = 0.172$$

II. Logistic Regression:

Logistic Regression is a statistical method used for binary classification, which means it's designed to predict the probability that an instance belongs to a particular class. Despite its name, logistic regression is employed for classification rather than regression tasks.

The basic idea of Logistic Regression models the relationship between a binary dependent variable and one or more independent variables. The dependent variable represents the outcome, which is typically coded as 0 or 1 (e.g., "yes" or "no," "spam" or "not spam").

Sigmoid (Logistic) Function: The logistic regression model uses the logistic function (also called the sigmoid function) to transform a linear combination of input features into a value between 0 and 1. The sigmoid function is defined as:

$$f(z) = \frac{1}{1 + e^{-z}}$$

where z is the linear combination of input features and model parameters. In this example the final TF- IDF features.

Training the Model: The model is trained using a process called maximum likelihood estimation. The goal is to find the values of the parameters that maximize the likelihood of observing the given set of outcomes in the training data.

Decision Boundary: The decision boundary is the line or hyperplane that separates the instances of one class from another. It is determined by the coefficients learned during training.

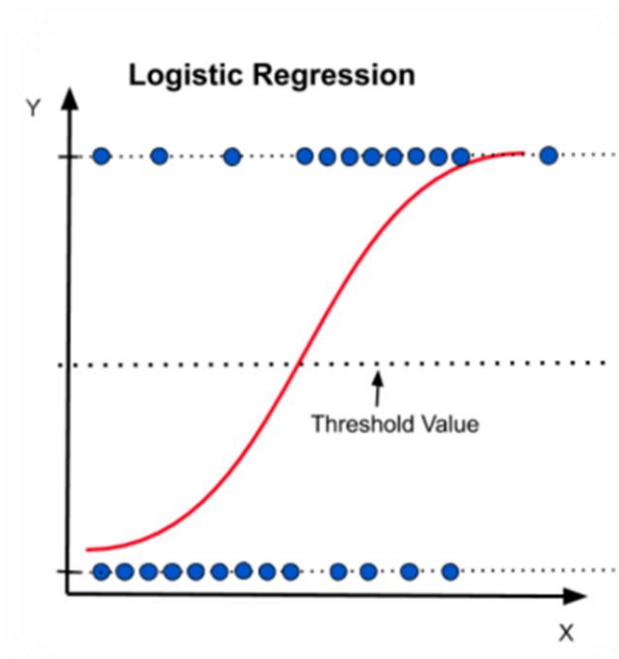


Fig 5. Logistic Regression Curve

Passing this preprocessed data into Logistic Regression Classifier in order to classify them as ‘weak’, ‘good’ and ‘strong’ passwords.

We use multinomial logistic regression as having classification of more than two values i.e. 0,1 and 2.

III. Novelty:

We have introduced a method which predicts the password strength and quantizes it into the strength of 0,1 and 2. This algorithm checks the length which should be greater than 8, uppercase, lowercase, number, special character and it finally checks that no two adjacent characters should repeat.

For the given password, the output is shown below:

```
dt=np.array(['MP7_two@111'])
```

```
(0, ['Password has consecutive occurrences, Kindly update with new characters'])
```

Fig 6. Novelty Introduced

1.6 Scope of the Project

The scope of a password strength predictor using machine learning is expansive, and its application extends across various domains due to the growing emphasis on robust cybersecurity measures. Here are key aspects that define the scope of a password strength predictor utilizing machine learning:

1. **Real-Time Authentication Enhancement:** Integration with user authentication systems to provide real-time feedback to users during the password creation process, guiding them towards choosing stronger and more secure passwords.
2. **Customized Password Policies:** Implementation of machine learning-based predictors to customize password policies based on user behavior, roles, and specific security requirements within an organization. This allows for adaptive and context-aware policies.
3. **Adaptive Threat Detection:** Machine learning models can adapt to emerging threats by analyzing patterns in password usage over time. This enables the system to dynamically adjust its predictions and policies in response to evolving cybersecurity risks.
4. **Continuous Learning:** The scope includes the ability for the password strength predictor to continuously learn from new data and adapt to changes in user behavior, emerging attack patterns, and evolving best practices in password security.
5. **Behavioral Biometrics Integration:** Integrating behavioral biometrics and machine learning to assess user-specific patterns in password creation, enhancing the ability to distinguish between legitimate users and potential attackers.
6. **Scalability and Performance:** Designing password strength predictors that are scalable and capable of handling large datasets efficiently, ensuring high performance even as user bases expand.

The evolving nature of cybersecurity threats necessitates the continuous evolution and adaptation of password strength predictors, making the scope of machine learning applications in this domain dynamic and far-reaching.

1.7 Gant Chart



Chapter 2

Literature Review

2.1 Paper 1: Password Strength prediction Using Supervised Machine Learning

Algorithms

1. Abstract: With ever-increasing security concerns due to unauthorized intrusion into private data it has become necessary to keep strong passwords. Thus, it is necessary to develop a machine learning approach to scrutinize the strength of the password and to provide a possible protection against the exploitation of vulnerabilities.
2. Approach: Widely used supervised machine learning techniques namely C 4.5 Decision tree classifier, Multilayer Perceptron, Naïve Bayes Classifier and Support Vector Machine were used for learning the model. The results of the models were compared and observed that SVM performs well.
3. Conclusion: The presented work proposes classification as the machine learning approach for accomplishing with the password strength prediction. The dataset consists of 10000 entries belonging to different categories. The performance of the model was evaluated using 10-fold cross validation and observed that SVM classifier trained with RBF kernel performs well.

2.2 Paper 2: A Password Strength Evaluation Algorithm based on Sensitive Personal Information

1. Abstract: In this era of digitalization, one has to set passwords every now and then. Humans have a tendency to inculcate sensitive personal information while choosing their passwords. A weak password is prone to cyber-attacks as it paves the way for hackers to crack it easily. To get rid of them, an algorithm for evaluating passwords based on private data is modelled known as bidirectional matching algorithm.
2. Approach: In this paper, we use the structure segmentation algorithm and the bidirectional matching algorithm to investigate how users' personal information is used in passwords. It proposes three steps: preprocessing, prediction dictionary generation and password strength evaluation.

3. Conclusion: This work defines the coverage ratio of personal information in the password strength predictor and further calculates the password strength to suggest the user to select a quantified password for better security.

2.3 Paper 3: Password Strength Analysis and its Classification by Applying Machine Learning Based Techniques

1. Abstract: In today's scenario, passwords serve an important mechanism for authentication and protection of devices. Myriads methods such as biometrics including face and fingerprint recognition have been introduced in recent times, but none could completely eliminate the use of passwords. Usually, an individual chooses a password comprising of alphanumeric as well as special characters. Its drawback is that a homosocial has to memorize different passwords which proves to be challenging and inconvenient. Password cracking tools are rapidly increasing on the internet thus an organization should ensure strong passwords. For similar reasons, they should include password strength predictors, like one proposed in this paper, on their servers.
2. Approach: In this proposed work, prediction of the strength of a password has been modeled as a classification task with the aid of multiple supervised machine learning algorithms which were used for the learning purpose. The novelty of this work lies in the fact, that two new algorithms namely, XGBoost and Multilayer Perceptron have been implemented to determine the strength and the corresponding category of a password which has not been evaluated in similar other works.
3. Conclusion: In this algorithm, passwords are broadly categorized into three main classes namely Weak, Medium and Strong passwords. Weak passwords are further classified into four sub-parts: numbers only, small letters only, block letters only and special characters only. First type of medium passwords are: numeric and small letters, numeric and block letters, numeric and special characters, block and small letters, symbols and block letters as well as symbols and small case letters. Strong password consists of a combination of uppercase and lowercase letters, special characters and numeric values having length larger than 8 characters.

Chapter 3

Software Design

Python, a versatile and dynamically-typed programming language, has become a cornerstone in the world of software development. Known for its simplicity, readability, and vast ecosystem of libraries, Python caters to a broad range of applications, from web development and data science to artificial intelligence.

One of Python's key strengths lies in its elegant syntax, fostering a clean and concise coding style. This simplicity accelerates development cycles, making it an ideal choice for both beginners and experienced developers. Python's extensive standard library and third-party packages further contribute to its popularity, providing solutions for diverse tasks without the need to reinvent the wheel.

Python's flexibility shines in web frameworks like Django and Flask, enabling rapid development of robust web applications. Additionally, its prominence in data science and machine learning is evident through libraries such as NumPy, pandas, and TensorFlow, empowering researchers and engineers to analyze and model complex datasets.

In essence, Python's widespread adoption stems from its balance of simplicity, power, and community support. As an open-source language, Python continues to evolve, embracing emerging technologies and reinforcing its position as a go-to language for solving real-world problems across industries.

Libraries Used:

1. Pandas is an open-source library in Python that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series.
2. NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions.
3. Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

4. Warnings are provided to warn the developer of situations that aren't necessarily exceptions. Usually, a warning occurs when there is some obsolete of certain programming elements, such as keyword, function or class, etc.
5. Python Random module is an in-built module of Python that is used to generate random numbers in Python. These are pseudo-random numbers means they are not truly random.
6. `sklearn.feature_extraction.text.TfidfVectorizer` to Convert a collection of raw documents to a matrix of TF-IDF features. scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines.
7. In Python, date and time are not data types of their own, but a module named `DateTime` can be imported to work with the date as well as time. Python `Datetime` module comes built into Python, so there is no need to install it externally.
8. A regular expression, sometimes referred to as rational expression, is a sequence of characters that specifies a match pattern in text.

Chapter 4

Experimental Simulations and Results

```
clf.fit(X_train,y_train) #fit the data

* LogisticRegression
LogisticRegression(multi_class='multinomial', random_state=0)

from datetime import datetime
start = datetime.now()

doing prediction for specific custom data

dt=np.array(['Parth@99999']) #consider an array of test password
pred=vectorizer.transform(dt) #vectorizer for converting string to numeric
clf.predict(pred) #to predict the strength of the password

array([1])

end=datetime.now()
td=(end-start).total_seconds()*10**3
print(f"The time of execution is : {td:.03f}ms")

The time of execution is : 20.175ms
```

```
from datetime import datetime

# record current timestamp
start = datetime.now()
MIN_PASSWORD_LENGTH = 6
MAX_PASSWORD_LENGTH = 16

def password_func(ls):
    password = input("Please enter your password: ")
    if len(password) < MIN_PASSWORD_LENGTH or len(password) > MAX_PASSWORD_LENGTH:
        message = "Password should have more than 6 but less 16 characters"
        print(message)
        ls.append(message)
        return password_func(ls)
    elif password.isalpha():
        message = "password weak - contains only letters"
        print(message)
        ls.append(message)
        return password_func(ls)
    elif password.isnumeric():
        message = "password weak - contains only numbers"
        print(message)
        ls.append(message)
        return password_func(ls)
    else:
        return ls.append("strong")

ls = []
password_func(ls)
print("")
print("error messages")
print("ls,sep='\n')
end = datetime.now()

# find difference loop start and end time and display
td = (end - start).total_seconds() * 10**3
print(f"The time of execution of above program is : {td:.03f}ms")

Please enter your password: Parth@99999

error messages
strong
The time of execution of above program is : 9935.321ms
```

Fig 7. Time Complexity Comparison of Simple Python Code and ML Approach

1. **Reading the Data set:** Here, our data set is a CSV file consisting of 6,00,000 data. On_bad_lines= 'skip' means to skip the lines consisting parser (erroneous lines). Furthermore, we would find the unique values in the data of strength.

	password	strength
0	kzde5577	1
1	kino3434	1
2	visi7k1yr	1
3	megzy123	1
4	lamborghini1	1

Fig 8. Reading the Data

2. **Check all the missing values in the dataset:** Firstly, count the null values in a dataset. Furthermore, drop the null values. Lastly plot the countplot of the strength data.

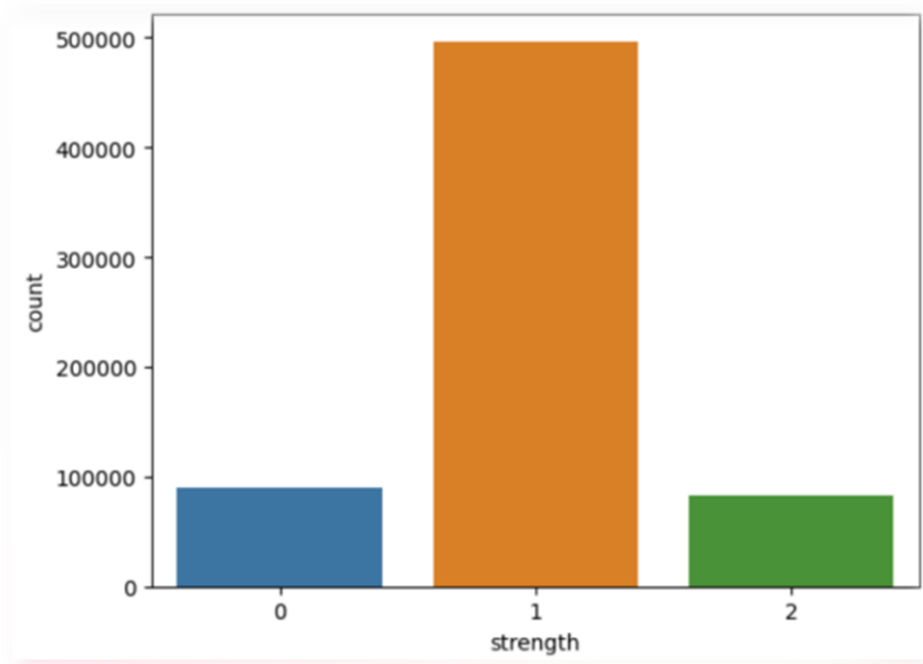


Fig 9. Classification of Data into Strengths

3. **Shuffling randomly for robustness & create a custom function to split input into characters of list Apply TF-IDF vectorizer on data:** Here, we create a function named `word_divide_char` which split the password into a list of character
4. **Apply TF-IDF vectorizer on data:** Here, the TF-IDF function segregates all the features from all the features. Furthermore, a matrix is created suggesting the importance of every feature. Lastly, sort this importance in descending order.
5. **Split data into train & test:** Apply Logistic on data as use-cas is Classification Doing prediction for specific custom data Firstly, split the data into train and test. Furthermore, define the `test_size=0.2` that is 20% of the data is considered for test and 80% for training purpose.
6. **Apply Logistic on data as use-cas is Classification:** Apply multinomial classification as there are more then 2 predictors that is 0,1 and 2.

7. **Doing prediction for specific custom data:** Firstly, consider an array of test password. Furthermore, apply vectorizer for converting string to numeric. Lastly, predict the strength by applying the trained logistic regression model.

```
array([1])
```

Fig 10. Prediction for MP7_two@111

8. **Doing prediction on X-Test data:** Check Accuracy of your model using confusion_matrix, accuracy_score Here, the 20% data set kept for testing purpose is predicted by the model. Finally, an array of strength was created as the output.

```
array([1, 1, 1, ..., 1, 1, 2])
```

Fig 11. Prediction for the Test Data

9. **Check Accuracy of your model using confusion_matrix, accuracy_score:** Here, the 20% data set kept for testing purpose is predicted by the model. Finally an array of strength was created as the output Check both the dataset of test and predicted and create a confusion matrix for the same. Here, the diagonal elements are the true predicted values and all others are false predictions. Finally, check the accuracy.

```
[[ 5245 12876    23]
 [ 3781 92887 2536]
 [    38  5097 11445]]
0.8181784242279434
```

Fig 12. Confusion Matrix of The Implemented Code

10. **Create report of your model:** Create a final report of the model with the precision, accuracy, recall, f1-score for every strength 0,1 and 2.

	precision	recall	f1-score	support
0	0.58	0.29	0.39	18144
1	0.84	0.94	0.88	99204
2	0.82	0.69	0.75	16580
accuracy			0.82	133928
macro avg	0.74	0.64	0.67	133928
weighted avg	0.80	0.82	0.80	133928

Fig 13. Final Report Summary for the ML Approach

While comparing various machine learning models namely XG Boost, MLP, Random Forest, Decision tree, Logistic regression, SVM, KNN, K means and Naïve Bayes based on accuracy we come to a conclusion that XG Boost performs best in terms of accuracy.

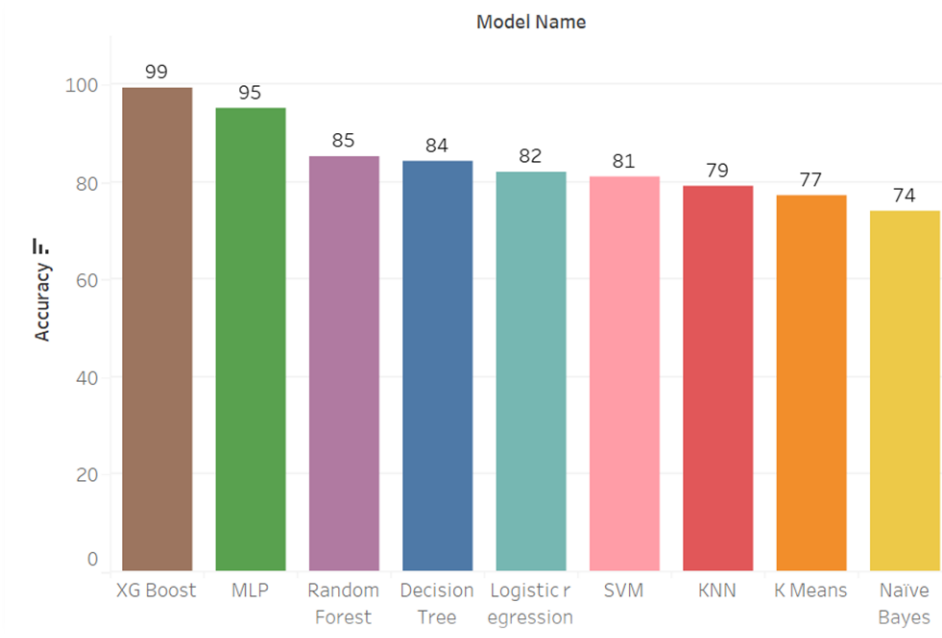


Fig 14. Comparison Based on Accuracy

While comparing various machine learning models namely XG Boost, MLP, Random Forest, Decision tree, Logistic regression, SVM, KNN, K means and Naïve Bayes based on training time we come to a conclusion that naïve bayes performs best in terms of training time.

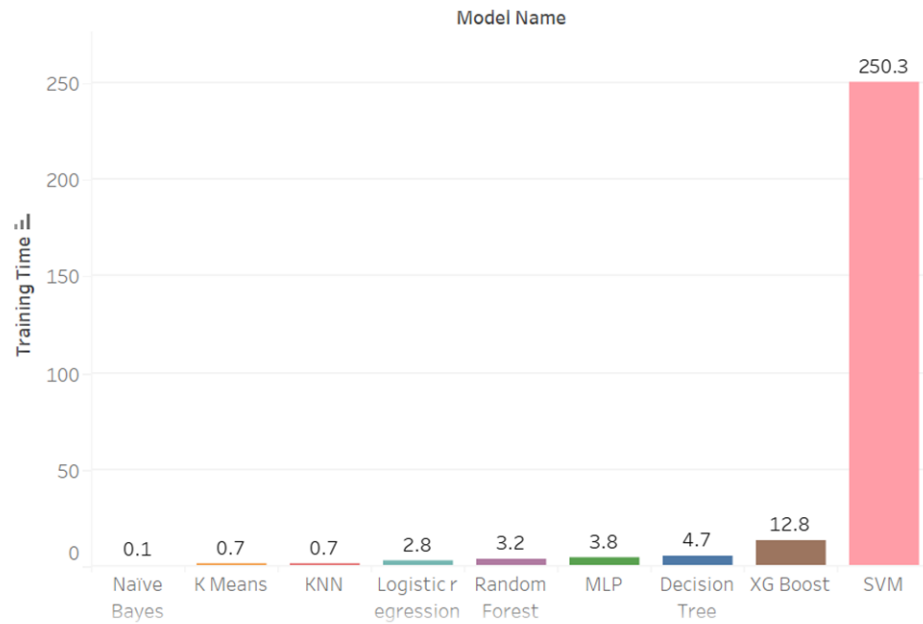


Fig 15. Comparison Based on Training Time

We chose logistic regression as our machine learning model as it is an amalgamation of a decent accuracy with a less training time.

Chapter 5

Conclusion and Future Scope

5.1 Conclusion

In conclusion, our project on password strength prediction using machine learning has successfully explored the effectiveness of various algorithms, ultimately revealing the strengths and nuances of each. Leveraging the TF-IDF vectorizer for feature extraction and employing logistic regression as a baseline model, we established a foundation for comparison.

Introducing a novel approach, the suggestive password algorithm demonstrated promising potential in enhancing user experience by providing personalized password suggestions based on individual patterns. This innovative feature adds a layer of user-friendly guidance to the security paradigm.

In our comprehensive model comparison, XG Boost emerged as the top performer in terms of accuracy, showcasing its robust capabilities in handling complex patterns within the password dataset. Meanwhile, Naive Bayes exhibited impressive efficiency during training, making it an attractive option for scenarios where computational resources are a critical consideration.

The deliberate choice of logistic regression was rooted in its interpretability and simplicity. While it may not outperform some advanced models in terms of accuracy, its efficiency and ease of understanding make it a pragmatic choice for applications where transparency and resource efficiency are paramount.

As we conclude this project, we recognize that the dynamic landscape of cybersecurity requires continual exploration and adaptation. The insights gained from this endeavor pave the way for future advancements in password security, offering a blend of accuracy, efficiency, and user-centric features to fortify digital defenses.

5.2 Future Scope

The project on password strength prediction using machine learning presents a promising avenue for future exploration and expansion in the ever-evolving field of cybersecurity. Several potential areas of future scope can be identified:

1. **Dynamic Password Policies:** Exploration of adaptive password policies that evolve based on the continuously changing threat landscape and user behavior. This could involve the incorporation of reinforcement learning techniques to dynamically adjust password requirements in response to emerging risks.
2. **Hybrid Models and Ensemble Learning:** Investigation into the effectiveness of hybrid models and ensemble learning techniques that combine the strengths of multiple machine learning algorithms. This can potentially improve accuracy, robustness, and the model's ability to handle diverse password patterns.
3. **Explainable AI in Password Security:** Integration of explainable AI techniques to enhance the interpretability of the model's predictions. Providing clear insights into why a password is deemed strong or weak can aid users and cybersecurity professionals in better understanding and reinforcing security practices.
4. **Continuous Monitoring and Anomaly Detection:** Implementation of continuous monitoring mechanisms that can detect anomalous patterns in password creation and usage. Machine learning models can play a crucial role in identifying suspicious activities and triggering security alerts for potential breaches.
5. **User Education and Awareness:** Development of educational features within the project to inform and educate users about the importance of strong passwords and best practices in cybersecurity. Integration with training modules and awareness campaigns can contribute to a more security-conscious user base.
6. **Benchmarking Against Evolving Threats:** Regular benchmarking and testing of the password strength predictor against evolving cybersecurity threats. This involves incorporating the latest attack vectors and staying abreast of advancements in password cracking techniques to ensure the model remains resilient.

References:

- [1] Vijaya MS, Jamuna KS, Karpavali S - Password Strength Prediction using Supervised Machine Learning Techniques
- [2] Xinchun Chui, Xueqing Li, Yiming Qin - A Password Strength Evaluation Algorithm based on Sensitive Personal Information
- [3] Sakya Sarkar, Mauparna Nandan - Password Strength Analysis and its Classification by Applying Machine Learning Based Techniques
- [4] Predicting Password Strength Based on Natural Language Processing Technique (daffodilvarsity.edu.bd)
- [5] <https://medium.com/@aagarwal691/password-strength-classifier-using-ml-algorithms-31080dbccd77>
- [6] <https://teampassword.com/blog/ai-password-cracking>
- [7] <https://medium.com/analytics-vidhya/data-science-for-cybersecurity-password-strength-meter-b933b96bff32>
- [8] <https://thecleverprogrammer.com/2022/08/22/password-strength-checker-with-machine-learning/>
- [9] <https://machinelearningmastery.com/compare-machine-learning-algorithms-python-scikit-learn/>
- [10] Darbutaitė, E.; Stefanovič, P.; Ramanauskaitė, S. Machine-Learning-Based Password-Strength-Estimation Approach for Passwords of Lithuanian Context. *Appl. Sci.* **2023**, *13*, 7811. <https://doi.org/10.3390/app13137811>

Appendix

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

Reading dataset

```
In [2]: data=pd.read_csv('data.csv',on_bad_lines='skip') #to skip the lines consisting
data.head()
```

Out[2]:

	password	strength
0	kzde5577	1
1	kino3434	1
2	visi7k1yr	1
3	megzy123	1
4	lamborghini1	1

```
In [3]: data['strength'].unique()
```

Out[3]: array([1, 2, 0], dtype=int64)

```
In [4]: data.shape
```

Out[4]: (669640, 2)

code to check all the missing values in my dataset

```
In [5]: data.isna().sum() #to count null values in a dataset
```

Out[5]: password 1
strength 0
dtype: int64

```
In [6]: data[data['password'].isnull()] # null value in password column
```

Out[6]:

	password	strength
367579	NaN	0

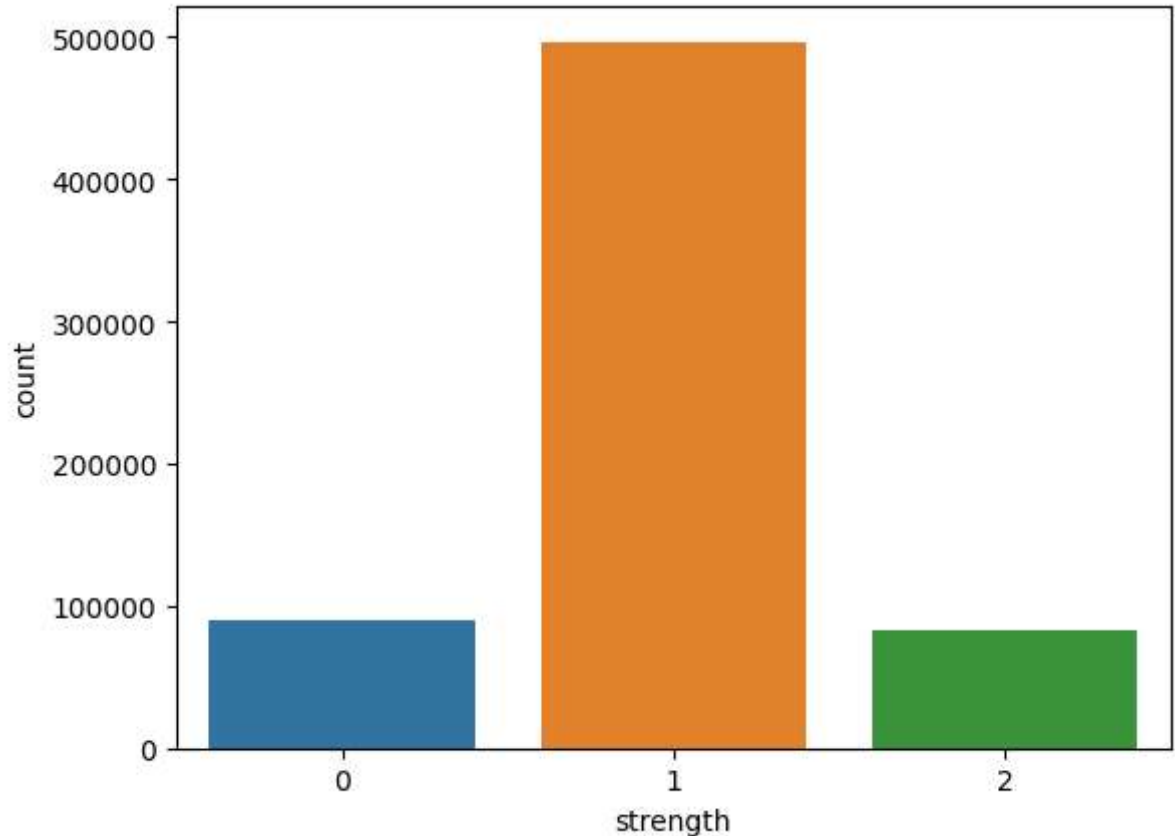
```
In [7]: data.dropna(inplace=True) # drop null value
```

```
In [8]: data.isnull().sum() #check the null value
```

```
Out[8]: password    0  
strength    0  
dtype: int64
```

```
In [9]: sns.countplot(x=data["strength"]) #count of different strengths
```

```
Out[9]: <Axes: xlabel='strength', ylabel='count'>
```



```
In [10]: password_tuple=np.array(data) #data representation in the form of an array
```

```
In [11]: password_tuple #displaying the array
```

```
Out[11]: array([[ 'kzde5577', 1],  
                [ 'kino3434', 1],  
                [ 'visi7k1yr', 1],  
                ...,  
                [ '184520socram', 1],  
                [ 'marken22a', 1],  
                [ 'fxx4pw4g', 1]], dtype=object)
```

```
In [ ]:
```

shuffling randomly for robustness

```
In [12]: import random
random.shuffle(password_tuple) #shuffle array
```

```
In [13]: x=[labels[0] for labels in password_tuple] #0 is index of password
y=[labels[1] for labels in password_tuple] # 1 is index of strength for list co
# y contains dependent data
```

```
In [14]: x
```

```
Out[14]: ['kzde5577',
          'kzde5577',
          'kino3434',
          'visi7k1yr',
          'lamborghini1',
          'AVYq1lDE4MgAZfNt',
          'AVYq1lDE4MgAZfNt',
          'megzy123',
          'kzde5577',
          'as326159',
          'megzy123',
          'kino3434',
          'megzy123',
          'kzde5577',
          '612035180tok',
          'jytifok873',
          'g067057895',
          'idof0673',
          'as326159',
          'lamborghini1']
```

```
In [ ]:
```

create a custom function to split input into characters of list

```
In [15]: def word_divide_char(inputs): # function to split the letters into list of a pa
          character=[]
          for i in inputs:
              character.append(i)
          return character
```

```
In [16]: word_divide_char('kzde5577') # example of the function
```

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

```
In [ ]:
```

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

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



```
In [24]: df=pd.DataFrame(first_document_vector.T.todense(),index=vectorizer.get_feature_
df.sort_values(by=['TF-IDF'],ascending=False) #sort the data on basis of import
```

Out[24]:

	TF-IDF
7	0.590990
5	0.567453
z	0.336194
k	0.291590
d	0.285791
...	...
>	0.000000
=	0.000000
<	0.000000
;	0.000000
,	0.000000

127 rows × 1 columns

In []:

split data into train & test

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

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

```
In [26]: X_train, X_test, y_train, y_test=train_test_split(X,y,test_size=0.2)#test_size=
```

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

Out[27]: (535711, 127)

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

In []:

Apply Logistic on data as use-cas is Classification

```
In [29]: clf=LogisticRegression(random_state=0,multi_class='multinomial')#multinomial be
```

```
In [30]: clf.fit(X_train,y_train) #fit the data
```

```
Out[30]: LogisticRegression
LogisticRegression(multi_class='multinomial', random_state=0)
```

```
In [31]: from datetime import datetime
start = datetime.now()
```

doing prediction for specific custom data

```
In [32]: dt=np.array(['Parth@99999']) #consider an array of test password
pred=vectorizer.transform(dt) #vectorizer for converting string to numeric
clf.predict(pred) #to predict the strength of the password
```

```
Out[32]: array([1])
```

```
In [33]: end=datetime.now()
td=(end-start).total_seconds()*10**3
print(f"The time of execution is : {td:.03f}ms")
```

The time of execution is : 20.175ms

doing prediction on X-Test data

```
In [34]: y_pred=clf.predict(X_test) #for the 20% data
y_pred #20% data predicted array
```

```
Out[34]: array([1, 1, 1, ..., 1, 1, 2])
```

```
In [ ]:
```

check Accuracy of your model using confusion_matrix,accuracy_score

```
In [35]: from sklearn.metrics import confusion_matrix,accuracy_score
```

```
In [36]: cm=confusion_matrix(y_test,y_pred) # both data of test and pred
print(cm)
print(accuracy_score(y_test,y_pred)) #same arrays
```

```
[[ 5245 12876    23]
 [ 3781 92887  2536]
 [    38  5097 11445]]
0.8181784242279434
```

In [37]: *# 5318, 92752, 11370 are true prediction wheras others are false pred. therefor*

create report of your model

In [38]: `from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))`

	precision	recall	f1-score	support
0	0.58	0.29	0.39	18144
1	0.84	0.94	0.88	99204
2	0.82	0.69	0.75	16580
accuracy			0.82	133928
macro avg	0.74	0.64	0.67	133928
weighted avg	0.80	0.82	0.80	133928

In [39]: *# model is good if we use advance classifier can increase the accuracy or by hy*

```

In [40]: #Suggestive password maker
import re
def check_password_strength(password):
    score = 0
    suggestions = []

    # check Length
    if len(password) >= 8:
        score += 1
    else:
        suggestions.append("Password should be at least 8 characters long")

    # check for uppercase Letter
    if re.search(r"[A-Z]", password):
        score += 1
    else:
        suggestions.append("Password should contain at least one uppercase lett

    # check for Lowercase Letter
    if re.search(r"[a-z]", password):
        score += 1
    else:
        suggestions.append("Password should contain at least one lowercase lett

    # check for numeric digit
    if re.search(r"\d", password):
        score += 1
    else:
        suggestions.append("Password should contain at least one numeric digit"

    # check for special character
    if re.search(r"[!@#$%^&*,.?\"':{}|<>_]", password):
        score += 1
    else:
        suggestions.append("Password should contain at least one special charac

    #consecutive Letter check
    if any(c1 == c2 for c1, c2 in zip(password, password[1:])):
        suggestions.append("Password has consecutive occurrences, Kindly update
    else:
        score += 1
    if score == 0 or 1 or 2 or 3:
        score = 0
    elif score == 4 or 5:
        score = 1
    else:
        score = 2
    return score, suggestions
password = np.array_str(dt)
print(check_password_strength(password))

```

```

(0, ['Password has consecutive occurrences, Kindly update with new character
s'])

```



```

In [42]: from datetime import datetime

# record current timestamp
start = datetime.now()
MIN_PASSWORD_LENGTH = 6
MAX_PASSWORD_LENGTH = 16
def password_func(ls):
    password = input("Please enter your password: ")
    if len(password) < MIN_PASSWORD_LENGTH or len(password) > MAX_PASSWORD_LENGTH:
        message = "Password should have more than 6 but less 10 characters"
        print(message)
        ls.append(message)
        return password_func(ls)
    elif password.isalpha():
        message = "password weak - contains only letters"
        print(message)
        ls.append(message)
        return password_func(ls)
    elif password.isnumeric():
        message = "password weak - contains only numbers"
        print(message)
        ls.append(message)
        return password_func(ls)
    else:
        return ls.append("strong")
ls = []
password_func(ls)
print("")
print("error messages")
print(*ls, sep='\n')
end = datetime.now()

# find difference loop start and end time and display
td = (end - start).total_seconds() * 10**3
print(f"The time of execution of above program is : {td:.03f}ms")

```

Please enter your password: Parth@999999

error messages

strong

The time of execution of above program is : 9935.321ms

In []:

```
In [1]: import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings("ignore")
```

```
In [2]: dataset=pd.read_csv('data.csv',on_bad_lines='skip')
```

```
In [3]: ##Lets check the head of our data
dataset.head()
```

```
Out[3]:
```

	password	strength
0	kzde5577	1
1	kino3434	1
2	visi7k1yr	1
3	megzy123	1
4	lamborghini1	1

```
In [4]: ##Lets check the shape of data
dataset.shape
```

```
Out[4]: (669640, 2)
```

```
In [5]: ##Checking the unique values in strength
dataset["strength"].unique()
```

```
Out[5]: array([1, 2, 0], dtype=int64)
```

There are only 3 classes in strength

0->Password is weak

1->Normal password

2->Strong password

In [6]: `dataset.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 669640 entries, 0 to 669639
Data columns (total 2 columns):
#   Column      Non-Null Count  Dtype
---  -
0   password    669639 non-null object
1   strength    669640 non-null int64
dtypes: int64(1), object(1)
memory usage: 10.2+ MB
```

In [7]: *##Lets see if there are any null values in dataset*
`dataset.isnull().sum()`

Out[7]: password 1
strength 0
dtype: int64

There is 1 null value in password

In [8]: *##Lets check the record where our value is null*
`dataset[dataset["password"].isnull()]`

Out[8]:

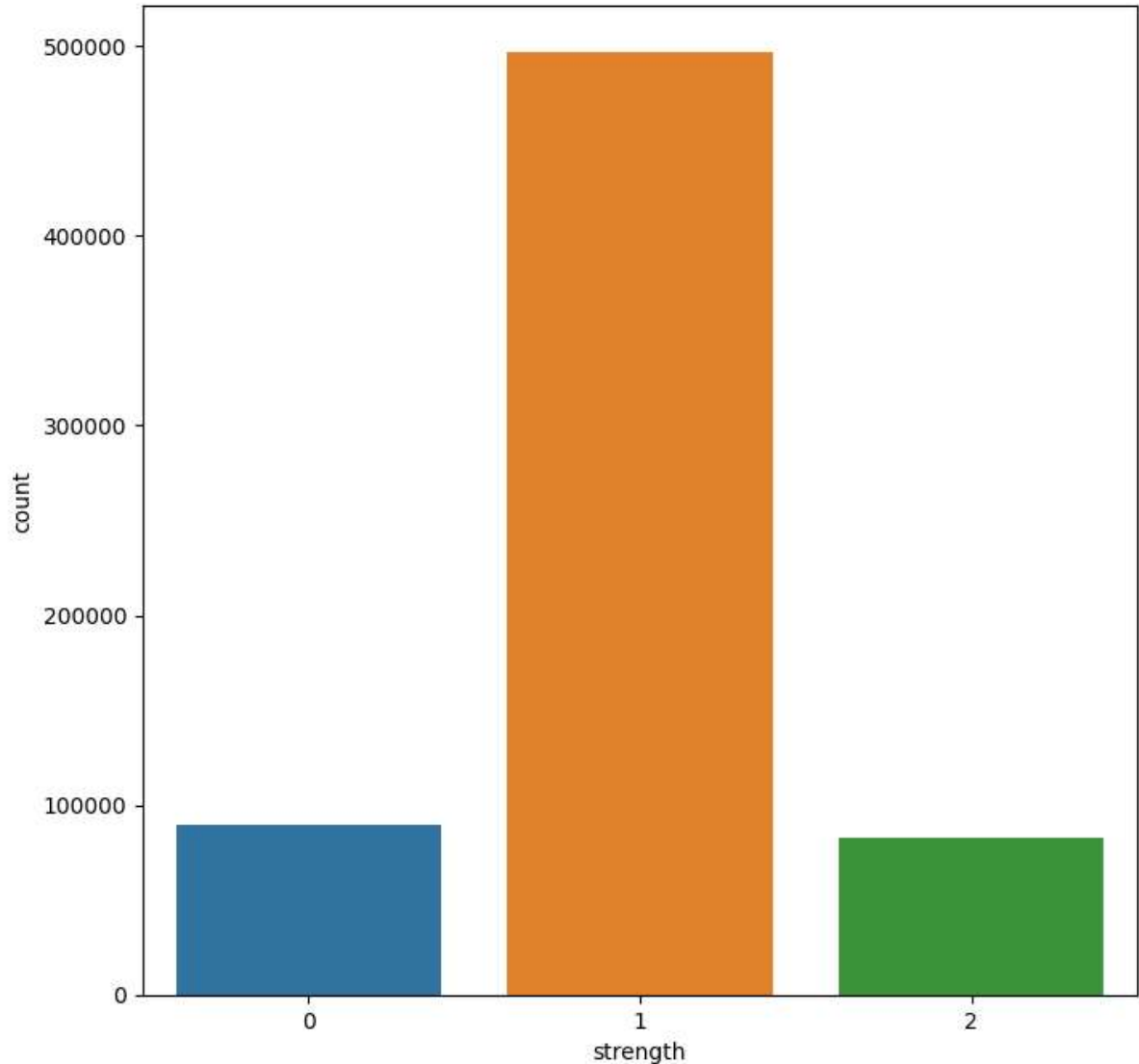
	password	strength
367579	NaN	0

As there is only 1 record we can drop our nan values

In [9]: `dataset.dropna(inplace=True)`

```
In [10]: plt.figure(figsize=(8,8))
sns.countplot(x=dataset["strength"])
```

```
Out[10]: <Axes: xlabel='strength', ylabel='count'>
```



As we can see count of class 1 is every high compared to class 0 and class 2

```
In [11]: ##I'll convert my data to array
password_=np.array(dataset)
password_
```

```
Out[11]: array([[ 'kzde5577', 1],
                [ 'kino3434', 1],
                [ 'visi7k1yr', 1],
                ...,
                [ '184520socram', 1],
                [ 'marken22a', 1],
                [ 'fxx4pw4g', 1]], dtype=object)
```

```
In [12]: type(password_)
```

```
Out[12]: numpy.ndarray
```

```
In [13]: password_[0]
```

```
Out[13]: array(['kzde5577', 1], dtype=object)
```

```
In [14]: ##Extracting the password from data  
password_[0][0]
```

```
Out[14]: 'kzde5577'
```

```
In [15]: import random  
random.shuffle(password_)
```

```
In [16]: password_
```

```
Out[16]: array([[ 'kzde5577', 1],  
                [ 'kino3434', 1],  
                [ 'visi7k1yr', 1],  
                ...,  
                [ 'ckilpc8', 0],  
                [ 'bodafyw548', 1],  
                [ 'piesek1', 0]], dtype=object)
```

```
In [17]: ##Create my dependant and independant feature  
X=[passwords[0] for passwords in password_]  
y=[passwords[1] for passwords in password_]
```

```
In [18]: ##Lets check passwords  
X
```

```
Out[18]: ['kzde5577',  
          'kino3434',  
          'visi7k1yr',  
          'kzde5577',  
          'kino3434',  
          'AVYq1lDE4MgAZfNt',  
          'visi7k1yr',  
          'lamborghini1',  
          'AVYq1lDE4MgAZfNt',  
          'v1118714',  
          'kino3434',  
          'u6c8vhow',  
          '612035180tok',  
          'jytifok873',  
          'lamborghini1',  
          'jytifok873',  
          'kino3434',  
          'asv5o9yu',  
          'visi7k1yr',  
          'v1118714']
```

```
In [19]: type(X)
```

```
Out[19]: list
```

```
In [20]: ##Convert words into characters  
def make_chars(inputs):  
    characters=[]  
    for letter in inputs:  
        characters.append(letter)  
    return characters
```

```
In [21]: make_chars("MinorProject")
```

```
Out[21]: ['M', 'i', 'n', 'o', 'r', 'P', 'r', 'o', 'j', 'e', 'c', 't']
```

```
In [22]: vectorizer=TfidfVectorizer(tokenizer=make_chars)
```

```
In [23]: X_=vectorizer.fit_transform(X)
```

```
In [24]: X_.shape
```

```
Out[24]: (669639, 133)
```

```
In [25]: vectorizer.get_feature_names_out()
```

```
Out[25]: array(['\x04', '\x05', '\x06', '\x08', '\x0e', '\x10', '\x12', '\x13',  
                '\x16', '\x17', '\x19', '\x1b', '\x1c', '\x1d', '\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',  
                '\x81', '\xa0', 'ı', 'ç', 'æ', 'ı', '«', '°', '±', '²', '³', '´',  
                'µ', '¶', '·', '¸', '¹', 'º', '»', '¼', '½', '¾', 'À', 'Á', 'Â', 'Ã', 'Ä',  
                'Å', 'Æ', 'Ç', 'È', 'É', 'Ê', 'Ë', 'Ì', 'Í', 'Î', 'Ï', 'Ð', 'Ñ', 'Ò', 'Ó', 'Ô',  
                'Õ', 'Ö', '÷', 'Ù', 'Ú', 'Û', 'Ü', 'Ý', 'Þ', 'ß', 'à', 'á', 'â', 'ä', 'å',  
                'ö', 'ø', '÷', 'ù', 'ú', 'û', 'ü', 'ý', 'þ', 'ÿ', '-', '†', '›'],  
               dtype=object)
```

```
In [26]: X_[0]
```

```
Out[26]: <1x133 sparse matrix of type '<class 'numpy.float64'>'  
         with 6 stored elements in Compressed Sparse Row format>
```

```
In [27]: first_=X_[0].T.todense()
```

```
In [28]: vec=pd.DataFrame(first_,index=vectorizer.get_feature_names_out(),columns=['tfidf
```

In [29]:

```
vec
```

Out[29]:

	tfidf
	0.0
	0.0
	0.0
	0.0
	0.0
...	...
p	0.0
ÿ	0.0
—	0.0
‡	0.0
›	0.0

133 rows × 1 columns

In [30]:

```
vec.sort_values(by=['tfidf'],ascending=False)
```

Out[30]:

	tfidf
7	0.592039
5	0.566540
z	0.335797
k	0.291835
d	0.285592
...	...
;	0.000000
9	0.000000
8	0.000000
6	0.000000
›	0.000000

133 rows × 1 columns

In [31]:

```
x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.27,random_state=42)
```

In [32]:

```
x_train.shape,x_test.shape
```

Out[32]: ((488836, 133), (180803, 133))

```
In [33]: #Model
from sklearn.linear_model import LogisticRegression
import xgboost as xgb
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.cluster import KMeans
from sklearn import neighbors
from sklearn import svm
```

```
In [34]: classifier=[]
classifier.append(LogisticRegression(multi_class='multinomial',solver='newton-cg'))
classifier.append(xgb.XGBClassifier(n_jobs=-1))
classifier.append(RandomForestClassifier(n_estimators=10,criterion='entropy',random_state=0))
classifier.append(svm.SVC(kernel='linear', C = 1.0))
classifier.append(KMeans(n_clusters = 3, random_state = 0, n_init='auto'))
classifier.append(neighbors.KNeighborsRegressor(n_neighbors = 3))
classifier.append(MultinomialNB())
```

```
In [35]: classifier
```

```
Out[35]: [LogisticRegression(multi_class='multinomial',n_jobs=-1, solver='newton-cg'),
          XGBClassifier(base_score=None, booster=None, callbacks=None,
                        colsample_bylevel=None, colsample_bynode=None,
                        colsample_bytree=None, early_stopping_rounds=None,
                        enable_categorical=False, eval_metric=None, feature_types=None,
                        gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
                        interaction_constraints=None, learning_rate=None, max_bin=None,
                        max_cat_threshold=None, max_cat_to_onehot=None,
                        max_delta_step=None, max_depth=None, max_leaves=None,
                        min_child_weight=None, missing=nan, monotone_constraints=None,
                        n_estimators=100, n_jobs=-1, num_parallel_tree=None,
                        predictor=None, random_state=None, ...),
          RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=0),
          SVC(kernel='linear'),
          KMeans(n_clusters=3, n_init='auto', random_state=0),
          KNeighborsRegressor(n_neighbors=3),
          MultinomialNB())]
```

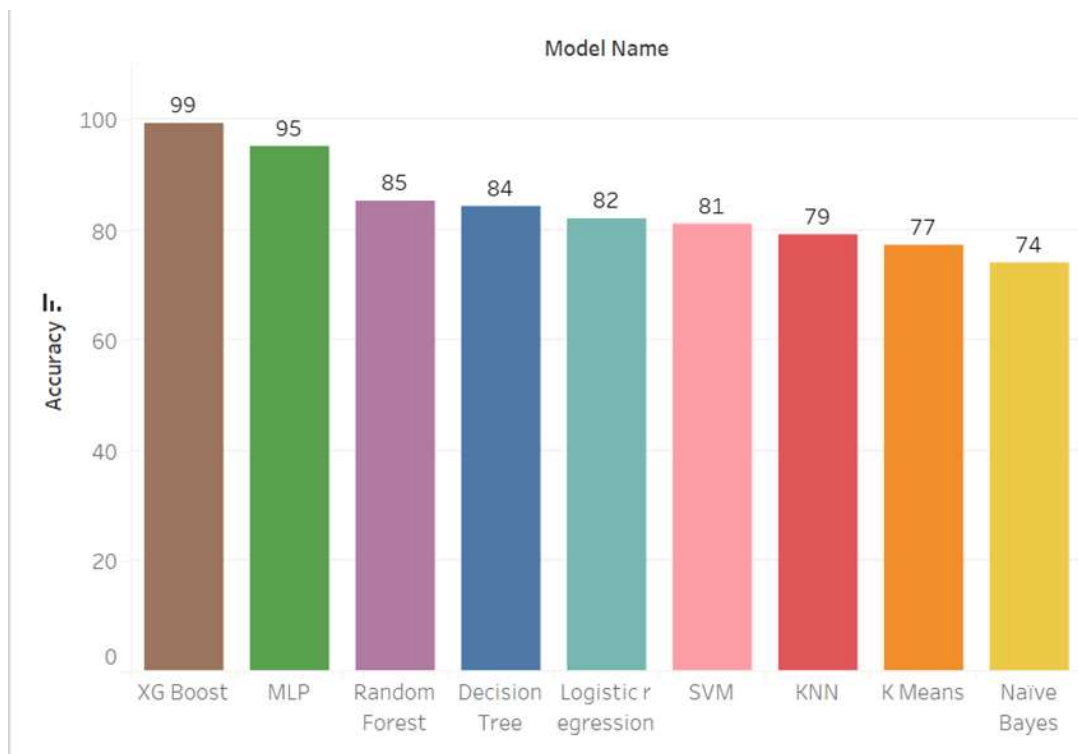
```
In [ ]: result=[]
for model in classifier:
    a=model.fit(x_train,y_train)
    result.append(a.score(x_test,y_test))
```

```
In [ ]: result1=pd.DataFrame({'score':result,
                             'algorithms':['logistic Regression','xgboost','Random For
```



```
In [ ]: result1
```

```
In [ ]: plt.figure(figsize=(8,8))
a=sns.barplot(x='score',y='algorithms',data=result1)
a.set_label('accuracy')
```



As we can see that XGBoost performs good for that given data

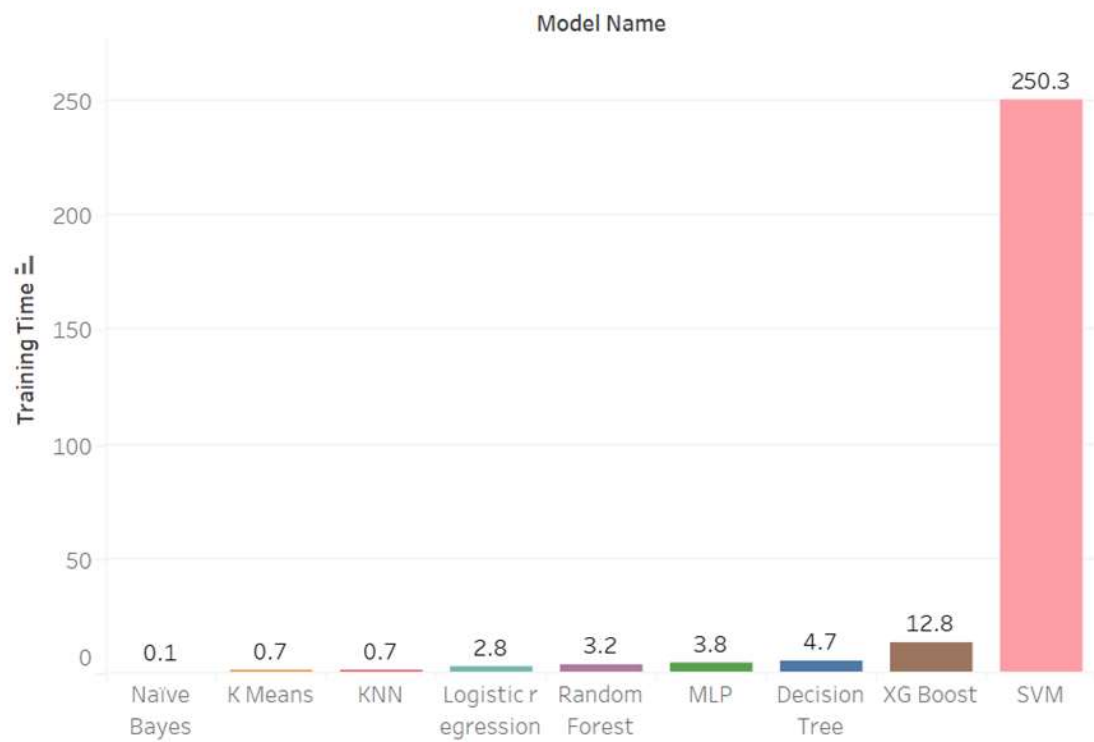
```
In [ ]: import time
for model in classifier:
    model_name = classifier.__class__.__name__

    start_time = time.time()
    model.fit(X_train, y_train)
    end_time = time.time()
    training_time = end_time - start_time

    print(f"{model_name} Training Time: {training_time} seconds")
```

```
In [ ]: result2=pd.DataFrame({'score':result,
                             'algorithms':['logistic Regression','xgboost','Random For
```

```
In [ ]: plt.figure(figsize=(8,8))
a=sns.barplot(x='time',y='algorithms',data=result2)
a.set_label('time')
```



In []: