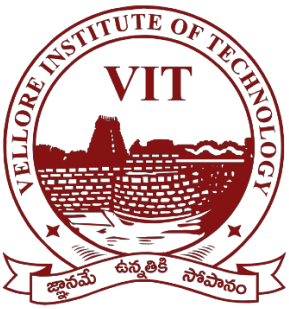


VIT-AP UNIVERSITY

A STUDY ON

Password Strength Classification

Submitted by

Abhay Chaudhary -19BCE7290
Gurram Maurya -19BCD7178
Yash Kumar Panse -19BCE7289
Sudhir Kumar Singh -19BCE7447

UNDER THE GUIDANCE

OF

Dr Nitesh Asaramji Funde
(Professor)

In Partial Fulfillment For The Award Of Degree Of

Computer Science And Engineering

2019-2023

DECLARATION

I, at this moment, declare that this project entitled *"Password Strength Classification"* was submitted by me during the period of DECEMBER In Partial Fulfillment For The Award Of Degree Of Computer Science And Engineering At Vellore Institute Of Technology.

I also declare that this project is the result of my team effort. Therefore, it was not submitted earlier to any other person/university to award any other degree or diploma, either in whole or partial.

PLACE: VIT-AP

DATE: May 16, 2022

Abhay Chaudhary -19BCE7290
Gurram Maurya -19BCD7178
Yash Kumar Panse -19BCE7289
Sudhir Kumar Singh -19BCE7447

SUMMARY

We used various machine learning models to classify the passwords depending on their strength in our dataset. We can reasonably claim that SVM (radial) is the best method, with the highest accuracy and F1 score, followed by SVM and the Naive Bayes classifier. On the other hand, random Forest lags because it successfully classifies two classes but fails to categorize one. One of the reasons for this is that we utilize the party random forest library, which reduces the dataset to run compared to the standard libraries.

The same concept may be used for any group of passwords. We've also built a function that accepts the user's input and calculates the strength using these techniques, bolstering our claims.

TABLE OF CONTENTS

Chapter 1

Introduction

Since the emergence of technology, passwords have become a significant element of people's lives. Since the dawn of technology, they have played a critical role in safeguarding our digital lives online. In a world where information security is becoming increasingly important, every organization's requirement for user access control is vital. Verifying a user's identity and granting appropriate access is known as authentication. Authentication may be accomplished using various techniques, such as cards or biometric characteristics such as iris scanners and fingerprint scanners. The difficulty with these approaches is that smartcards and other similar items might be lost or stolen, and biometric scanners are expensive and require additional resources that aren't always accessible, particularly in houses. Password authentication, the most prevalent type of authentication, is generally what a person knows. Passwords are the most frequent authentication method since they are free and easy to use, but they may also be the most dangerous if not set up correctly. Most passwords do not necessitate using a separate hardware device. Thus there is no cost to the company in purchasing one. Physical hardware devices may be utilized by anybody who has them, making them more likely to be misplaced or stolen. Password systems also do not need much hardware or computing power to function. Because users are already accustomed to using passwords to access, these authentication methods seldom require substantial training.

Passwords, unlike biometric information, may be quickly changed if they are hacked. However, if implemented with suitable rules and processes, passwords may give significant security benefits to both companies and people. A password is a secret that is remembered. Usually, a string of characters to verify a user's identity. A party retains the secret termed the claimant, while the party validating the claimant's identity is called the verifier, according to the NIST Digital Identity Guidelines. The verifier can infer the claimant's identity if the claimant successfully shows password knowledge to the verifier using an established authentication methodology. The efficiency of a password in repelling guessing and brute-force assaults is measured by its password strength. Therefore, the length and complexity of a password are crucial. The size of a password is merely the number of individual characters used in its construction, whereas complexity refers to the number of characters used in its production.

The absolute risk of a security compromise is reduced when strong passwords are used.

Chapter 2

Background Study

Significant research has been done with passwords, their security, authentication methods and options beyond passwords. More secure alternatives to passwords exist. But as Herley at al stated in their paper, there are a number of barriers to moving beyond passwords, such as diversity of requirements, user reluctance and usability, individual control of end user systems etc. As of today, alphanumeric passwords are still the most common mode of authentication; hence the focus rests on improving the security of passwords and their authentication. Halderman et al bypass the need to remember multiple passwords for different accounts by using a strengthened hash function to generate high entropy passwords when they are needed. These passwords are protected by a single short master password. Udi Manber implemented a scheme with two salts to prevent guessing attacks on passwords protected with one way functions. So far, most of the existing research focuses on secure management and storage of passwords.

Password Security: An Analysis of Password Strengths and Vulnerabilities Keith at al presented an empirical study based on the usability of passphrases after a 12 week long experiment. Campbell et al proved that enforcing good password composition rules does not discourage users from setting strong and meaningful passwords. Alain et al used Persuasive Technology as a method to help users chose memorable passwords without forgoing security. Schechter et al propose a method to strengthen user passwords by setting a minimum acceptable false positive rate to prevent statistical guessing attacks. Duggan et al analyzed the password goals for different groups such as students, administrative staff and scientists and observed how password security was related to the sensitivity of their tasks. Kharod et al proposed a new technique that involves the use of hashing, salting and differential masking with a low time complexity to strengthen passwords. Bailey at al studies the fact that users pick passwords of different strengths for different categories of websites; financial accounts have significantly stronger passwords and analyzes the implications of this fact on password research. Despite research on strengthening passwords, data continues to be compromised on a regular basis, prompting the need for better vigilance and stronger passwords from both users as well as organizations. This paper focuses on how organizations as well as individual users can safeguard their data better against malevolent attacks.

Passwords have long been used in the IT sector. They've been employed for various objectives, the most common of which safeguard someone's identity or data. Many research articles have been published on this subject, and research has been ongoing for quite some time. Every study focused on a method for improving current research. For example, "Password Strength Prediction Using Supervised Machine Learning Techniques" and "A New Multimodal Approach for Password Strength Estimation—Part II: Experimental Evaluation" are two articles published in IEEE on this topic. To get better outcomes, these articles employed a variety of strategies.

The dataset we chose can be found on Kaggle.com. The credentials utilized in the research came from the internet breach of 000webhost. The password's strength was determined using a tool developed by Georgia Tech University called PARS, which includes all commercial password metres. All of the passwords were supplied to the device, and it generated new files for each commercial password strength metre. The passwords were stored in files with an additional column indicating their strength as determined by commercial password strength metres. Twitter, Microsoft, and battle are among the commercial password strength algorithms employed. Rather than rules, it is solely dependent on machine learning; only credentials that have been classified as weak, medium, or firm by all three strength metres were stored. This indicates that all passwords were weak, medium, or firm.

There were 3 million passwords, but only 0.7 million passwords remained after the intersection of all commercial metre classes. Because only those passwords were utilized that were highlighted in a specific category by all three algorithms, the decrease was achieved.

Chapter 3

Problem

Definition

The most important words or phrases used to protect someone's identity and data are passwords. They serve as your first line of defence against illegal access to your accounts and personal data. In addition, your computer will be safer from hackers and lousy malware if you use a strong password.

However, judging the strength of a password simply by glancing at it is challenging. Furthermore, determining a password's strength based on some guidelines is not recommended since it might be deceptive because various password metres employ different criteria to determine its strength. As a result, we used a database of existing passwords with specified strengths based on settings provided by commercial password metres. Based on this data, we constructed a machine learning model to predict if a password is weafounddium, or firm. Instead of using rules, we used machine learning to make our forecast.

Then, to determine the optimal ML algorithm, we compared numerous ML methods. We have included a feature that allows users to enter their passwords and retrieve the strength based on these algorithms.

Chapter 4

Objective

The main goal of our research is to determine the strength of a password and classify it as Strong, Weak, or Medium using machine learning methods (2,1,0). We compared the most prominent machine learning techniques for our dataset in addition to this classification. We've also introduced a feature that allows users to input a password and check it for strength using these methods. Finally, we must fulfil specific secondary objectives to achieve our core objectives, including preprocessing, comprehending the data, applying the appropriate models, and assessing the findings.

Chapter 5

Methodology/Procedure

Gathering data → Pre-processing → Creating ML models → Analyzing the results → Conclusion

1) Gathering Data:

Kaggle.com was used to obtain the data. The passwords we utilized in our investigation came from the internet breach of 000webhost. The strength of the passwords in the dataset is pre-determined using a technology called PARS developed by Georgia Tech University, which includes all commercial password metres.

Twitter, Microsoft, and battle are among the commercial password strength algorithms employed.

2) a. There are eight columns in the dataset:

3) b. The first column is called 'Id,' and it is unique.

4) c. The passwords are in the second column.

5) d. The password length is listed in the third column.

6) e. The percentage of capital, lowercase, numerals, and special characters in the password is represented in the fourth, fifth, sixth, and seventh columns, respectively.

7) f. The password's strength is shown in the last column, ranging from 0-Weak to 1-Medium to 2-Strong.

2) Preprocessing:

There were roughly 7 lakh rows and eight columns in our dataset. We chose a sample of 2 lakh rows for our study because the number of rows was enormous, and we didn't have the physical capacity to analyze the entire dataset. The Sr No column and the column holding the passwords themselves were deleted from our dataset. We deleted the password column since the type and quantity of characters in the password, not itself, determines our strength. Our second difficulty was that our dataset had no missing values. We added a function that generated missing values in the dataset at random. We have about 10% missing values in our dataset after that. We chose to mean to handle missing data since it gave us the best results, and we used mode for the Strength (Class) column because mean couldn't be used because it would establish a new class on its own. For our algorithms to work, we needed to modify the type of columns.

We changed the Strength column to factor for the algorithms and the length column to numeric from Integer. The next step was to see if any of the columns had a high association. We looked for a heatmap for this and couldn't locate any. After that, we moved on to the outliers. For this, we utilized the outlier's library. Outliers were deleted from all of the columns. The data has to be normalized next. Before normalization, we separated the data into train and test data in a 75:25 ratio. We then used the scale function to standardize the data. Finally, the caTools library was used to divide the data.

3) ML Models:

We started with a linear kernel Support Vector Machine (SVM). Then, we utilized the e1071 library. To analyze the data, we created a confusion matrix. After that, we switched to SVM with a radial kernel. The passwords were then categorized using a nave Bayes classifier. Then we moved on to Random Forest, for which we created a decision tree and a confusion matrix using the party library. The data were then compared and analyzed.

- Support Vector Machine (both linear and radial kernels): SVMs have supervised learning models that examine classification and regression analysis data. They come with related learning algorithms.

- Naïve Bayes Classifier: The Bayes' Theorem is used to create a set of classification algorithms known as Naive Bayes classifiers. It is a family of algorithms that share a similar idea, namely that each pair of characteristics being categorized is independent of the others.

- Random Forest: A random forest is a meta estimator that employs averaging to increase predicted accuracy and control over-fitting by fitting many decision tree classifiers on various sub-samples of the dataset. After that, we visualized our confusion matrix to make it easier to comprehend. We also designed a tool for the end-user to input a password and rapidly checked its strength.

Chapter 6

Results and Discussion

```
> str(datakaggle)
'data.frame':   75997 obs. of  2 variables:
 $ password: chr  "kzde5577" "kino3434" "visi7k1yr" "megzy123" ...
 $ strength: chr  "1" "1" "1" "1" ...

'data.frame':   669640 obs. of  8 variables:
 $ X                   : int  0 1 2 3 4 5 6 7 8 9 ...
 $ Password            : chr  "kzde5577" "kino3434" "visi7k1yr" "megzy123" ...
 $ Length              : int  8 8 9 8 11 16 8 8 12 8 ...
 $ X.No.of.Upper.Case: num  0 0 0 0 0 ...
 $ X.No.of.Lower.Case: num  0.5 0.5 0.778 0.625 0.909 ...
 $ X.No.of.Numbers    : num  0.5 0.5 0.2222 0.375 0.0909 ...
 $ X.No.of.Spl.Chars : num  0 0 0 0 0 0 0 0 0 0 ...
 $ Strength            : int  1 1 1 1 1 2 1 1 1 1 ...
```

Fig 1: Structure of the original dataset and pre-processed data

	X	Password	Length	X.No.of.Upper.Case	X.No.of.Lower.Case	X.No.of.Numbers	X.No.of.Spl.Chars	Strength
1	0	kzde5577	8	0.0000	0.5000000	0.50000000	0	1
2	1	kino3434	8	0.0000	0.5000000	0.50000000	0	1
3	2	visi7k1yr	9	0.0000	0.7777778	0.22222222	0	1
4	3	megzy123	8	0.0000	0.6250000	0.37500000	0	1
5	4	lamborghin1	11	0.0000	0.9090909	0.09090909	0	1
6	5	AVYq1lDE4MgAZfNt	16	0.5625	0.3125000	0.12500000	0	2

Fig 2: First 6 rows of the original dataset.

X	Password	Length	X.No.of.Upper.Case	X.No.of.Lower.Case	X.No.of.Numbers	X.No.of.Spl.Chars	Strength
Min. : 0	Length:669640	Min. : 1.00	Min. :0.00	Min. :0.00	Min. :0.00	Min. :0	Min. :0.00
1st Qu.:167322	Class :character	1st Qu.: 8.00	1st Qu.:0.00	1st Qu.:0.50	1st Qu.:0.19	1st Qu.:0	1st Qu.:1.00
Median :334895	Mode :character	Median : 9.00	Median :0.00	Median :0.67	Median :0.30	Median :0	Median :1.00
Mean :334779		Mean : 9.99	Mean :0.05	Mean :0.61	Mean :0.33	Mean :0	Mean :0.99
3rd Qu.:502185		3rd Qu.: 11.00	3rd Qu.:0.00	3rd Qu.:0.78	3rd Qu.:0.40	3rd Qu.:0	3rd Qu.:1.00
Max. :669639		Max. :220.00	Max. :1.00	Max. :1.00	Max. :1.00	Max. :1	Max. :2.00
NA's :67419		NA's :66887	NA's :66859	NA's :66938	NA's :66698	NA's :66840	NA's :67500

Fig 3: Summary of the dataset after creating Na's

	X	Password	Length	X.No.of.Upper.Case	X.No.of.Lower.Case	X.No.of.Numbers	X.No.of.Spl.Chars	Strength
1	0	kzde5577	8	0.0000	0.5000000	0.50000000	0	1
2	NA	kino3434	NA	0.0000	0.5000000	0.50000000	0	1
3	2	<NA>	9	0.0000	0.7777778	0.22222222	0	1
4	3	megzy123	NA	0.0000	0.6250000	0.37500000	0	1
5	4	lamborghin1	11	NA	0.9090909	0.09090909	0	1
6	5	AVYq1lDE4MgAZfNt	16	0.5625	0.3125000	0.12500000	0	NA

Fig 4: First six rows after inserting NA's

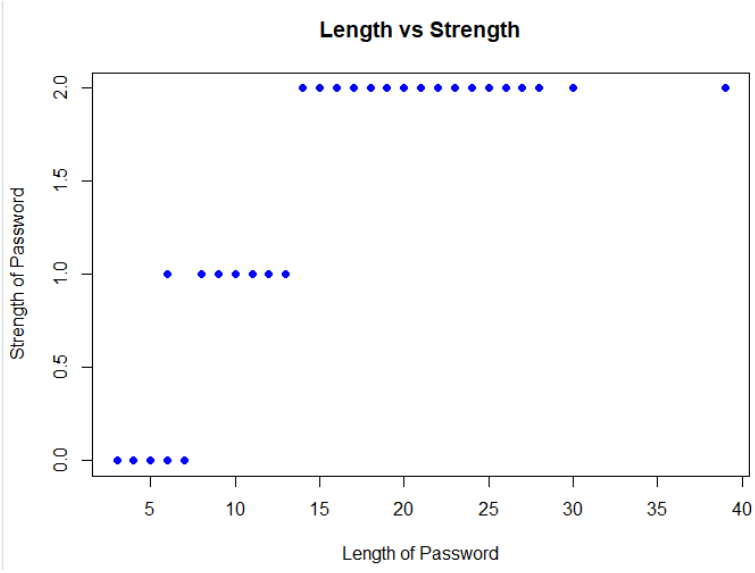


Fig 5: Basic Analysis from Dataset- Length vs Strength of the Password

We can see that as the password length increases, the strength also tends to increase. Also, our dataset has more passwords in the substantial region lengthwise.

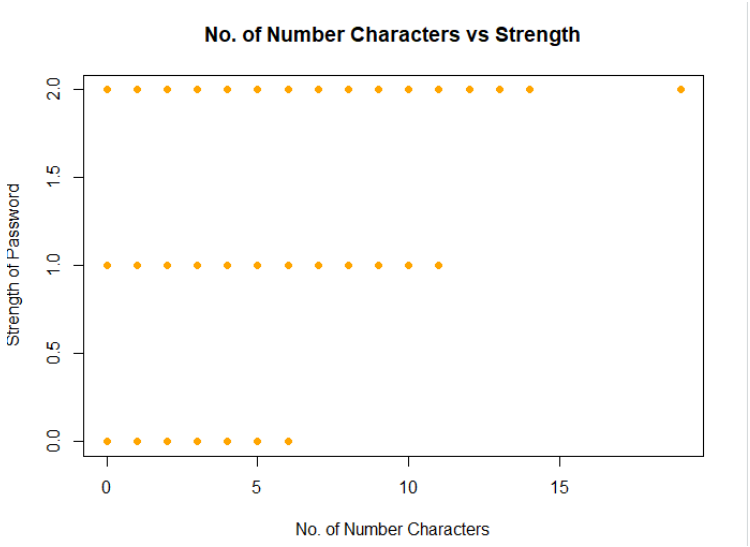


Fig 6: Basic Analysis from Dataset - No. of Number Characters vs Strength of the Password

The number of characters does not play as significant a part in determining strength as the length, but as the number of characters grows, so does the strength.

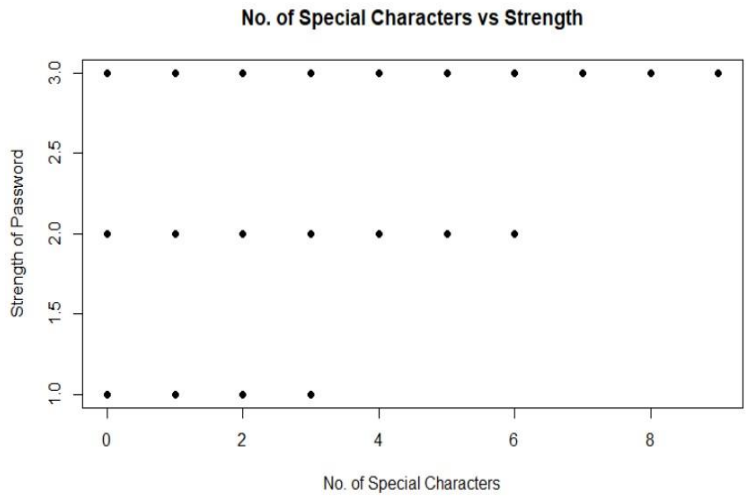


Fig 7: Basic Analysis from Dataset – Number of Special Characters vs Strength of the Password

An increase in special characters also increases the strength of the password.

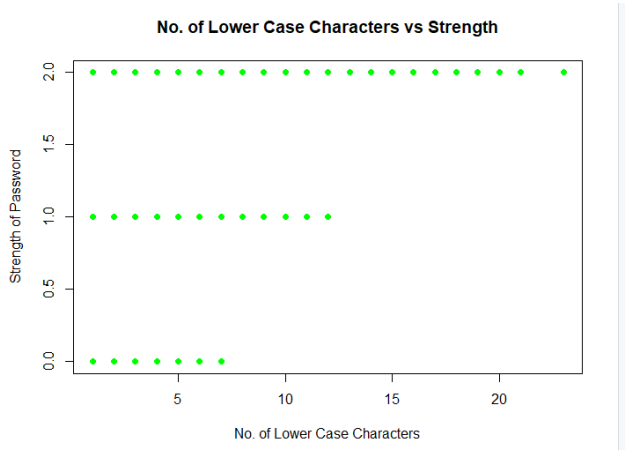


Fig 8: Basic Analysis from Dataset – No. of Lower Case Characters vs Strength of the Password

The more the number of lower case characters, the higher the strength.

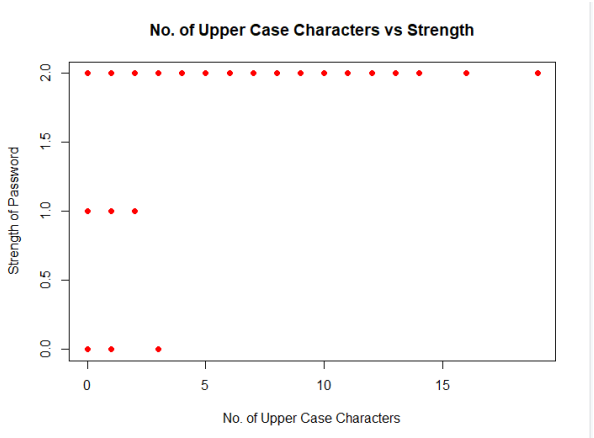


Fig 9: Basic Analysis from Dataset – No. of Upper Case Characters vs Strength of the Password

The collection contains many more items with a large number of upper-case characters. As the number of people grows, so does their power.

Also, we discovered that length is not the only criterion for determining strength since many passwords with more minor special characters, upper case, lower case, and other characters have high strength. We can deduce that both the length and the combination of different characters are equally important.

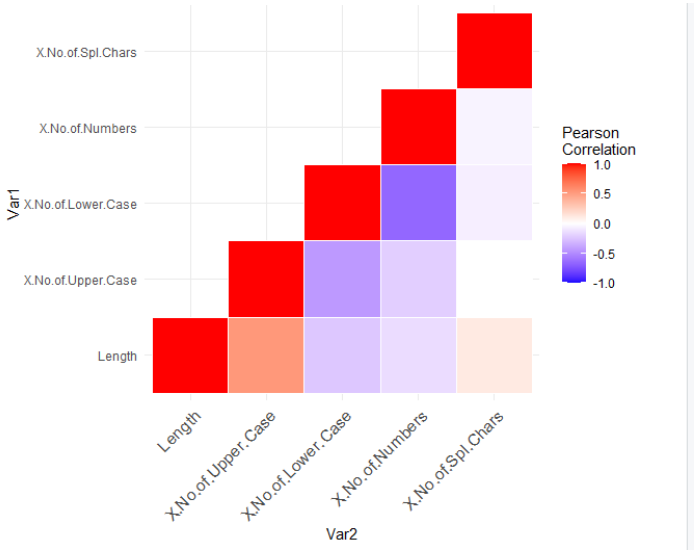


Fig 10: Correlation Analysis

No two columns are highly correlated with each other.

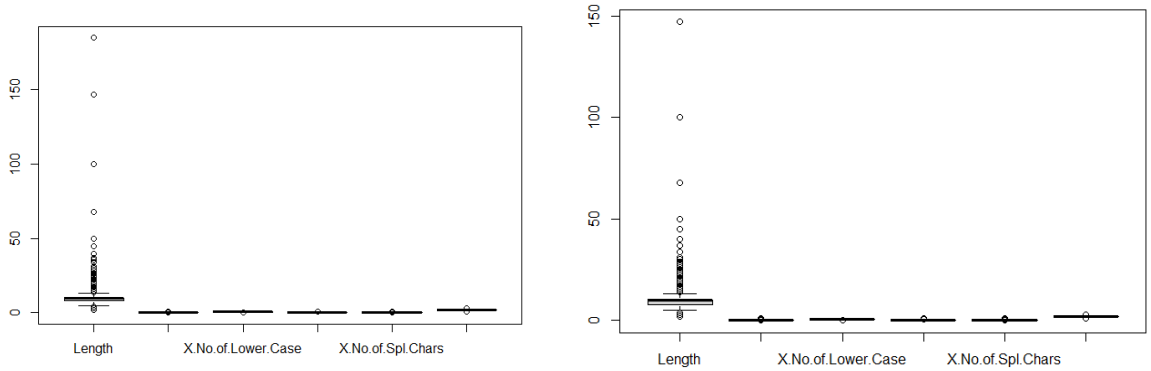


Fig 11: Boxplot Before and After Outliers

Outliers Detected and Removed

Length	1
Upper case	4
Lower Case	3329
No of numbers	11
Special characters	1

Analysis from ML models

Confusion Matrix SVM:

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1     2
         0  5261   611     0
         1   648 36501   594
         2     7   586  4955

Overall Statistics

               Accuracy : 0.9502
                 95% CI : (0.9483, 0.9522)
    No Information Rate : 0.7668
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.8705

 Mcnemar's Test P-Value : 0.04317

Statistics by Class:

                     Class: 0 Class: 1 Class: 2
Sensitivity            0.8893   0.9682   0.8930
Specificity            0.9859   0.8917   0.9864
Pos Pred Value         0.8959   0.9671   0.8931
Neg Pred Value         0.9849   0.8952   0.9864
Prevalence             0.1203   0.7668   0.1129
Detection Rate         0.1070   0.7424   0.1008
Detection Prevalence   0.1194   0.7677   0.1128
Balanced Accuracy      0.9376   0.9300   0.9397
```

Confusion Matrix SVM (Radial):

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1     2
         0  5211   594     0
         1   702 36487    88
         2     3   617  5461

Overall Statistics

               Accuracy : 0.9592
                 95% CI : (0.9575, 0.961)
    No Information Rate : 0.7668
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8956

 Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

                     Class: 0 Class: 1 Class: 2
Sensitivity            0.8808   0.9679   0.9841
Specificity            0.9863   0.9311   0.9858
Pos Pred Value         0.8977   0.9788   0.8980
Neg Pred Value         0.9837   0.8981   0.9980
Prevalence             0.1203   0.7668   0.1129
Detection Rate         0.1060   0.7422   0.1111
Detection Prevalence   0.1181   0.7582   0.1237
Balanced Accuracy      0.9335   0.9495   0.9850
```

Confusion Matrix Naïve Bayes:

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1     2
         0  4808   552     1
         1  1005 36247   376
         2   103   899  5172

Overall Statistics

               Accuracy : 0.9403
                 95% CI : (0.9381, 0.9424)
    No Information Rate : 0.7668
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8452

 Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

                     Class: 0 Class: 1 Class: 2
Sensitivity            0.8127   0.9615   0.9321
Specificity            0.9872   0.8795   0.9770
Pos Pred Value         0.8968   0.9633   0.8377
Neg Pred Value         0.9747   0.8742   0.9912
Prevalence             0.1203   0.7668   0.1129
Detection Rate         0.0978   0.7373   0.1052
Detection Prevalence   0.1090   0.7654   0.1256
Balanced Accuracy      0.9000   0.9205   0.9545
```

Confusion Matrix Random Forest:

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1     2
         0  2612   438     3
         1  3131 33681    27
         2   173  3579  5519

Overall Statistics

               Accuracy : 0.8505
                 95% CI : (0.8473, 0.8536)
    No Information Rate : 0.7668
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.623

 Mcnemar's Test P-Value : < 2.2e-16

Statistics by Class:

                     Class: 0 Class: 1 Class: 2
Sensitivity           0.44151   0.8934   0.9946
Specificity           0.98980   0.7246   0.9140
Pos Pred Value        0.85555   0.9143   0.5953
Neg Pred Value        0.92835   0.6741   0.9992
Prevalence            0.12033   0.7668   0.1129
Detection Rate        0.05313   0.6851   0.1123
Detection Prevalence  0.06210   0.7493   0.1886
Balanced Accuracy     0.71566   0.8090   0.9543
```

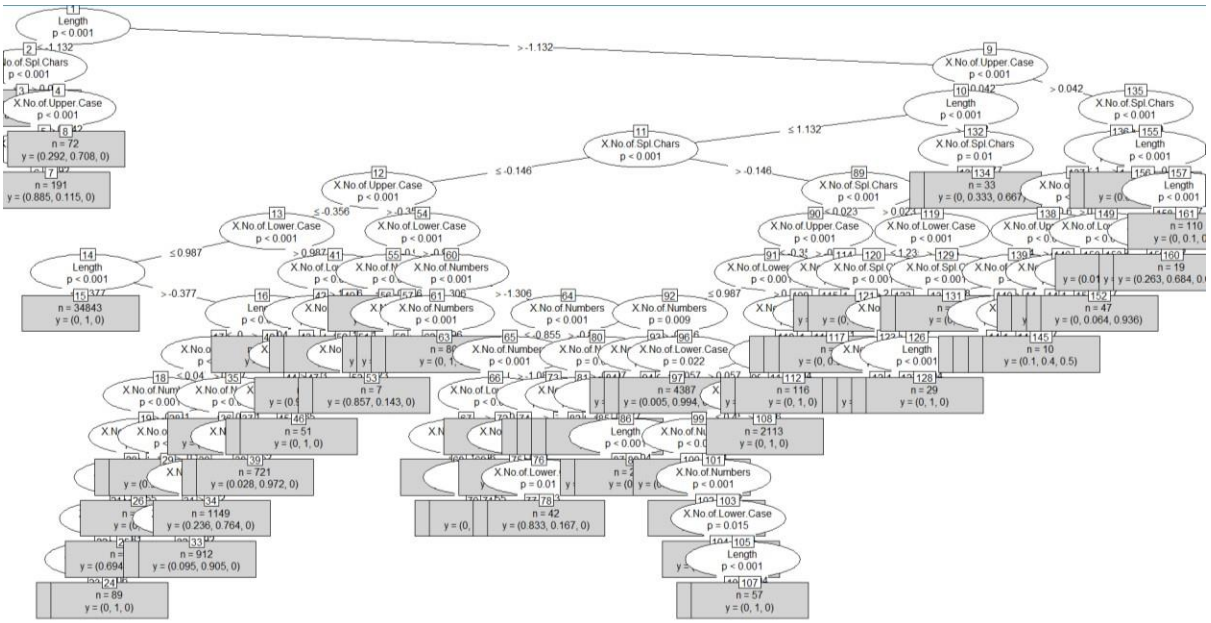
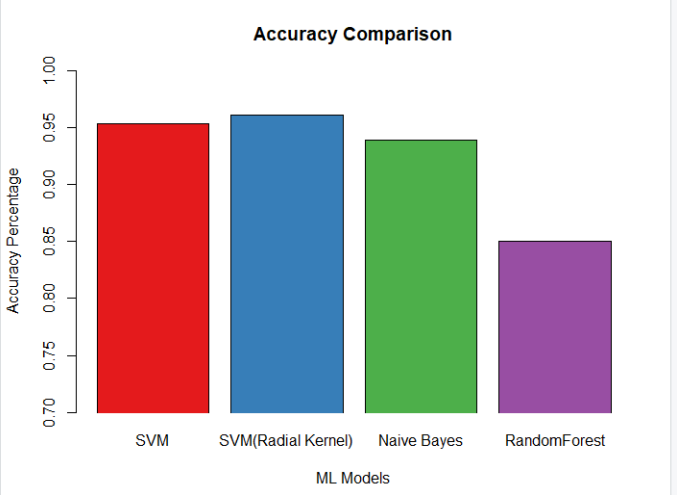


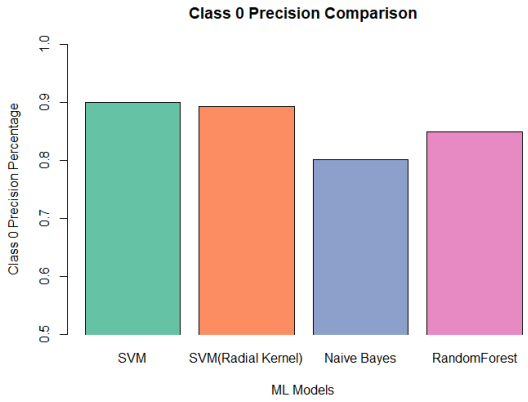
Fig 12: Decision Tree Using Random Forest

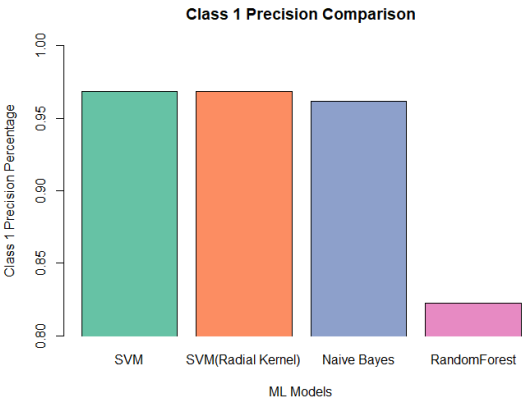
Accuracy:

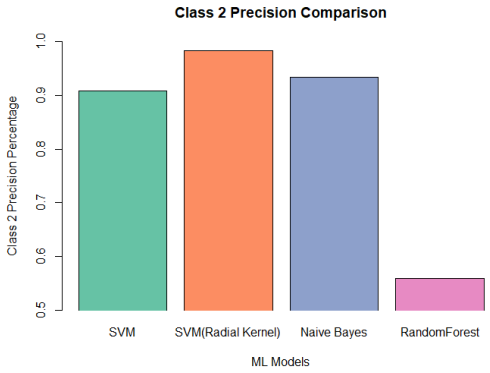


When we compare the accuracy of all the methods, we find that SVM – radial has the best accuracy, followed by SVM and nave Bayes, all of which are close to 95 per cent. The accuracy of random forest is 85 per cent. The SVM model should be used in this case.

Precision:



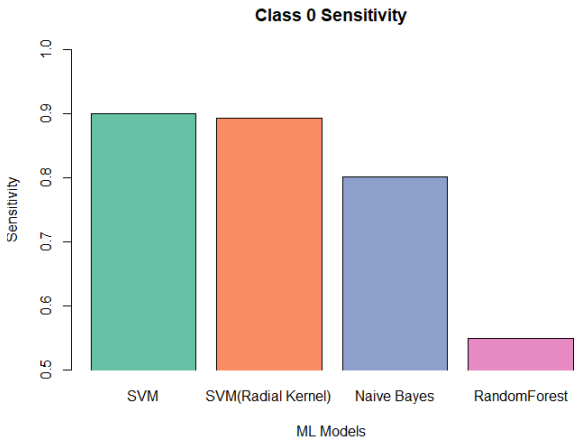


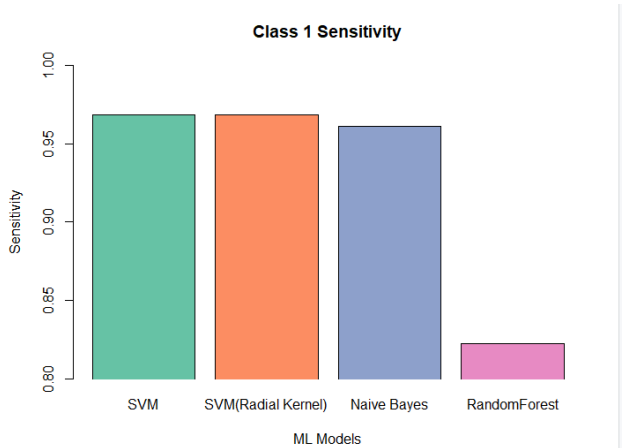


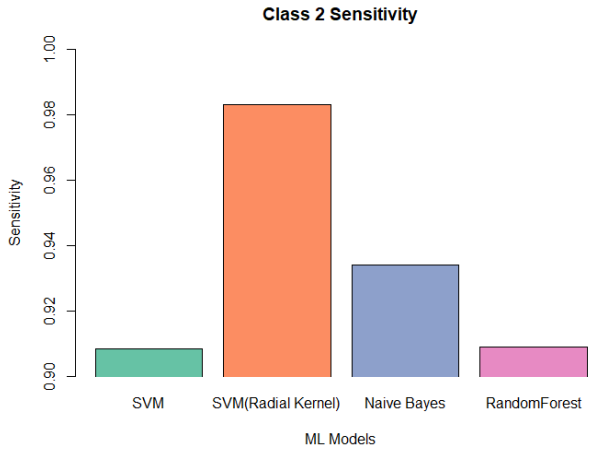
Precision (also known as positive predictive value) refers to the percentage of relevant results older than a certain age. For all of the classes, we can observe that SVM (both) have a good level of accuracy. For classes 1 and 2, Nave Bayes has high accuracy. However, for class 0, it fails. Only class 0 has a high level of accuracy, whereas the rest of the classes fail.

All of the algorithms have good accuracy, except for random forest for class 2, which has a lot of false positives.

Recall:



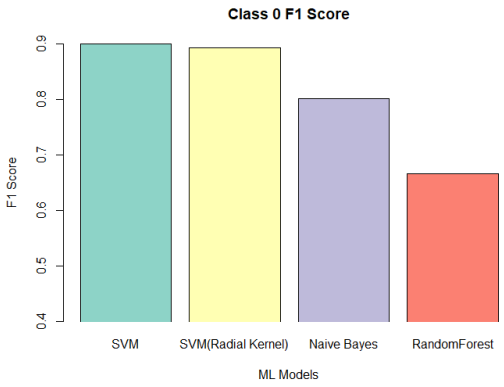


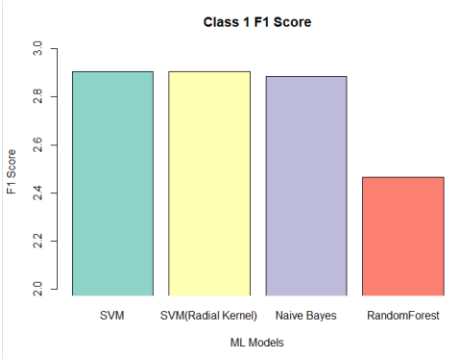


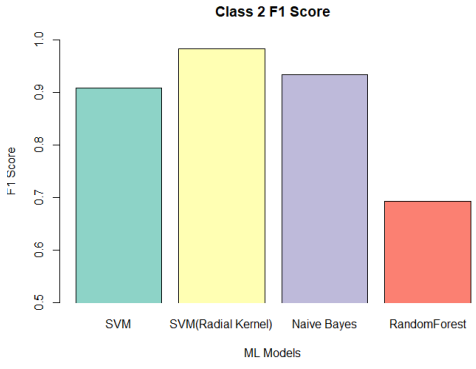
Recall (also called Sensitivity) refers to the %age of total relevant results correctly classified by the algorithm.

SVM radial is better than the rest as it shows high recall for all the classes. SVM linear shows consistent recall for all classes at 0.9. Naïve Bayes and random forest show comparatively low recall than others.

F1 Score

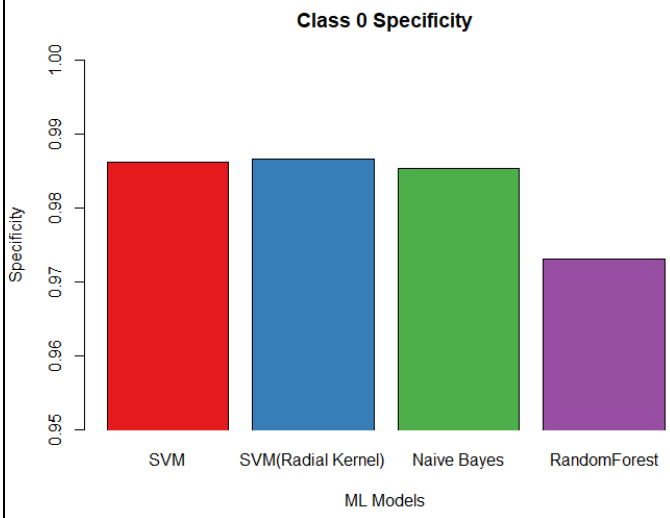


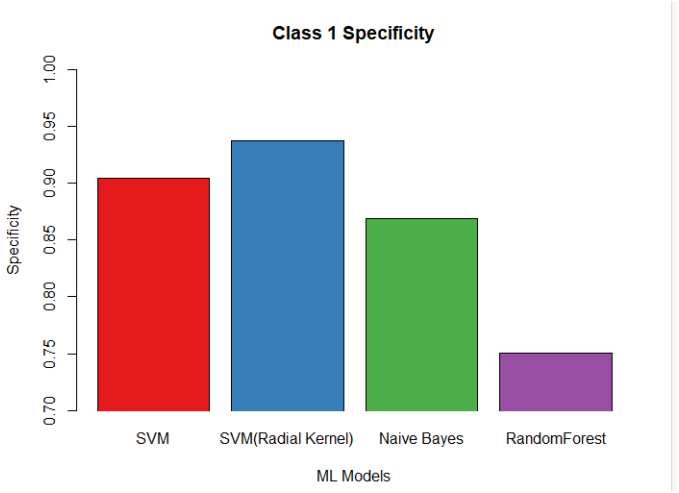


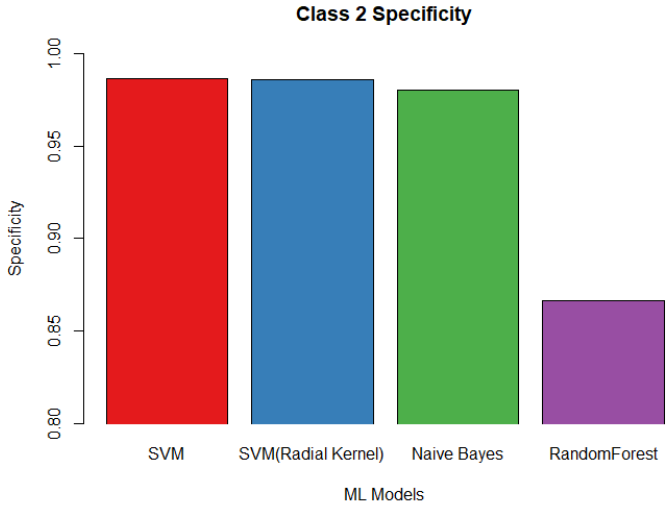


It is the harmonic mean of Precision and Recall and gives a better measure of the incorrectly classified cases than the Accuracy Metric.

Since we have imbalanced classes, the F1 score is a better metric. Now we can clearly say that SVM(radial) is better. SVM linear is better for binary classification and fails for multiclass


Class 0 Specificity
Specificity
1.00
0.99
0.98
0.97
0.96
0.95
SVM
SVM(Radial Kernel)
Naive Bayes
RandomForest
ML Models


Class 1 Specificity
Specificity
1.00
0.95
0.90
0.85
0.80
0.75
0.70
SVM
SVM(Radial Kernel)
Naive Bayes
RandomForest
ML Models


Class 2 Specificity
Specificity
1.00
0.95
0.90
0.85
0.80
SVM
SVM(Radial Kernel)
Naive Bayes
RandomForest
ML Models

User Prediction Function:

```
> length = nchar(str)
> upp=str_count(str, "[A-Z]")
> loo=str_count(str, "[a-z]")
> num=str_count(str, "[0-9]")
> spec=length-(upp+loo+num)
> vec<-c(length,upp/length,loo/length,num/length,spec/length)
> new_df=data.frame(vec)
> new_df=transpose(new_df)
> names(new_df)[1] <- "Length"
> names(new_df)[2] <- "X.No.of.Upper.Case"
> names(new_df)[3] <- "X.No.of.Lower.Case"
> names(new_df)[4] <- "X.No.of.Numbers"
> names(new_df)[5] <- "X.No.of.Spl.Chars"
> new_df[] = scale(new_df)
> print("Prediction form SVM")
[1] "Prediction form SVM"
> y_pred = predict(classifier_svm, newdata = new_df)
> print(y_pred)
factor(0)
Levels: 0 1 2
> print("Prediction form Kernel SVM")
[1] "Prediction form Kernel SVM"
> y_pred = predict(classifier_ksvm, newdata = new_df)
> print(y_pred)
factor(0)
Levels: 0 1 2
> print("Prediction form Naive Bayes")
[1] "Prediction form Naive Bayes"
> y_pred = predict(classifier_nb, newdata = new_df)
> print(y_pred)
[1] 1
Levels: 0 1 2
> print("Prediction form Random Forest")
[1] "Prediction form Random Forest"
> y_pred = predict(rf, newdata = new_df)
> print(y_pred)
[1] 1
Levels: 0 1 2
```

For password “abhay@123”

Strength given by SVM:0

SVM(radial) :0

Naïve Bayes :1

Random Forest :1

Chapter -7

Conclusion and Future Scope

On our dataset, we used various machine learning models to classify the passwords depending on their strength. We can reasonably claim that SVM (radial) is the best method, with the highest accuracy and F1 score, followed by SVM and the Naive Bayes classifier. On the other hand, random Forest lags because it successfully classifies two classes but fails to categorize one. One of the reasons for this is that we utilize the party random forest library, which reduces the dataset to run compared to the standard libraries.

The same concept may be used for any group of passwords. We've also built a function that accepts the user's input and calculates the strength using these techniques, bolstering our claims. This is an area where further research can be done. More categorization methods can be used, and the results can be compared. Newer strategies, such as ensemble, can help us achieve more accuracy, precision, recall, and, as a result, a higher F1 score. These approaches can also replace rule-based password metres because they are far more practical. We can avoid data breaches and assaults and make the world safer if we can make our passwords stronger.

References

[1] https://www.kaggle.com/bhavikbb/password-strength-classifier-dataset

[2] http://en.wikipedia.org/wiki/Password

[3] F.Bergadano, B.Crispo, G.Ruffo, “Proactive password checking with decision trees”,Proc. of the 4th ACM conference on computer and communications security, Zurich, Switzerland, 1997, pp 67-77.

[4] Giancarlo Ruffo, Francesco Bergadano, “EnFilter : A Password Enforcement and Filter Tool Based on Pattern Recognition Techniques”, Springer Berlin / Heidelberg, 1611-3349 (Online), Volume 3617/2005.

[5] Vijaya MS, Jamuna KS, Karpagavalli S,”Password Strength Prediction using Supervised Machine Learning Techniques”, IEEE, 978-1-4244-5321-4,pp 401-405, 2009.

[6] lan H. Witten, Eibe Frank, “Data Mining – Practical Mahine Learning Tools and Techniques,” 2nd Edition, Elsevier, 2005.

[7] John Shawe-Taylor, Nello Cristianini, “Support Vector Machines and other kernel-based learning methods”, 2000, Cambridge University Press, UK.

[8] Vapnik V.N,”Statistical Learning Theory”, J.Wiley & Sons, Inc., 1998, New York.

[9] Soman K.P, Loganathan R, Ajay V, “ Machine Learning

[10] with SVM and other Kernel Methods”, 2009, PHI, India.

[11] https://medium.com/analytics-vidhya/accuracy-vs-f1-score-6258237beca2

[12] https://ieeexplore.ieee.org/abstract/document/5376606

[13] https://www.ukessays.com/essays/information-technology/security-benefits-of- passwords-information-technology-essay.php

[14] https://medium.com/@faizann20/machine-learning-based-password-strength- classification-7b2a3c84b1a6

Appendix – A

Team Work and Work Management

A team is a collection of people united by a shared goal. Teams are especially well suited to completing activities with a high level of complexity and several corresponding subtasks. A team is a group of persons involved in the same task or activity. Members of a good team create an environment that allows everyone to go past their limits.

Teamwork is defined as the collaboration of people with varying levels of technical competence who work together in a group to achieve a shared goal or aim. To run efficiently and productively, each organization or enterprise requires collaboration. An organization's success is directly proportionate to the quality of its collaboration. Following the establishment of any team, it is critical to divide and manage the work that the Team must complete to achieve a goal. A solid team and effective work management within the Team may assist in completing any assignment swiftly.

Fig 13: Steps involved in team building

Appendix – B

Coding

```
library(ggplot2)
library(caret)
library(dplyr)

#initial dataset
datakaggle<-Kaggle.Password
dim(datakaggle)
str(datakaggle)
head(datakaggle)

#Missing Values
sum(is.na(datakaggle))

createNAs <- function (x, pctNA = 0.1) {
  n <- nrow(x)
  p <- ncol(x)
  NAloc <- rep(FALSE, n * p)
  NAloc[sample.int(n * p, floor(n * p * pctNA))] <- TRUE
  x[matrix(NAloc, nrow = n, ncol = p)] <- NA
  return(x)
}
datakaggle=createNAs(datakaggle)
summary(datakaggle)
sum(is.na(datakaggle))
head(datakaggle)

#processed dataset
dataset<-Password
dim(dataset)
str(dataset)
head(dataset)

datasetset=dataset[c(-1,-2)]
#Frequency of Length of password
ggplot(count(dataset,Length),aes(x=Length,y=n))+geom_bar(stat="identity")
ggplot(count(dataset[dataset$Length<25,],Length),aes(x=Length,y=n))+geom_bar(stat="identity")
#correlation between length and strength
cor(dataset[,c("Length","Strength")])
#boxplot
boxplot(dataset[,c(3:8)])
#removing outlier in Length
Q1 <- quantile(dataset$Length,0.25)
Q3 <- quantile(dataset$Length,0.75)
IQR <- IQR(dataset$Length)
dataset <- subset(dataset, dataset$Length> (Q1 - 1.5*IQR) & dataset$Length< (Q3 + 1.5*IQR))
Q1 <- quantile(dataset$Length,0.25)
Q3 <- quantile(dataset$Length,0.75)
IQR <- IQR(dataset$Length)
dataset <- subset(dataset, dataset$Length> (Q1 - 1.5*IQR) & dataset$Length< (Q3 + 1.5*IQR))

#removing outlier in X.No.of.Lower.Case
Q1 <- quantile(dataset$X.No.of.Lower.Case,0.25)
Q3 <- quantile(dataset$X.No.of.Lower.Case,0.75)
IQR <- IQR(dataset$X.No.of.Lower.Case)
dataset <- subset(dataset, dataset$X.No.of.Lower.Case> (Q1 - 1.5*IQR) & dataset$X.No.of.Lower.Case< (Q3 +
1.5*IQR))
Q1 <- quantile(dataset$X.No.of.Lower.Case,0.25)
Q3 <- quantile(dataset$X.No.of.Lower.Case,0.75)
IQR <- IQR(dataset$X.No.of.Lower.Case)
dataset <- subset(dataset, dataset$X.No.of.Lower.Case> (Q1 - 1.5*IQR) & dataset$X.No.of.Lower.Case< (Q3 +
```

```
1.5*IQR))

#Scatterplot
ggplot(dataset,aes(x=Length,y=Strength))+geom_point()
#heatmap
ggplot(dataset,aes(x=Length,y=Strength))+geom_tile()
#training datasetset
library(caTools)
set.seed(123)
split=sample.split(dataset,SplitRatio = 0.75)
train=subset(dataset,split==TRUE)
test=subset(dataset,split==FALSE)
#Normalizaton
train[-6] = scale(train[-6])
test[-6] = scale(test[-6])

#Missing Values
sum(is.na(dataset))

createNAs <- function (x, pctNA = 0.1) {
  n <- nrow(x)
  p <- ncol(x)
  NAloc <- rep(FALSE, n * p)
  NAloc[sample.int(n * p, floor(n * p * pctNA))] <- TRUE
  x[matrix(NAloc, nrow = n, ncol = p)] <- NA
  return(x)
}
dataset=createNAs(dataset)
summary(dataset)
sum(is.na(dataset))
head(dataset)

#Reducing the size of the dataset to improve the speed
dataset=dataset[sample(c(1:669640), size=200000, replace = FALSE, prob = NULL),]
dim(dataset)
str(dataset)
sum(is.na(dataset))
#Removing Unnecessay coloumns

dataset=dataset[c(-1,-2)]

#Dealing with missing values

mode <- function (x, na.rm) {
  xtab <- table(x)
  xmode <- names(which(xtab == max(xtab)))
  if (length(xmode) > 1) xmode <- ">1 mode"
  return(xmode)
}

dataset$Length = ifelse(is.na(dataset$Length),ave(dataset$Length, FUN = function(x) mean(x, na.rm =
'TRUE')),dataset$Length)
dataset$X.No.of.Upper.Case = ifelse(is.na(dataset$X.No.of.Upper.Case),ave(dataset$X.No.of.Upper.Case, FUN
= function(x) mean(x, na.rm = 'TRUE')),dataset$X.No.of.Upper.Case)
dataset$X.No.of.Lower.Case = ifelse(is.na(dataset$X.No.of.Lower.Case),ave(dataset$X.No.of.Lower.Case, FUN
= function(x) mean(x, na.rm = 'TRUE')),dataset$X.No.of.Lower.Case)
dataset$X.No.of.Numbers = ifelse(is.na(dataset$X.No.of.Numbers),ave(dataset$X.No.of.Numbers, FUN =
function(x) mean(x, na.rm = 'TRUE')),dataset$X.No.of.Numbers)
dataset$X.No.of.Spl.Chars = ifelse(is.na(dataset$X.No.of.Spl.Chars),ave(dataset$X.No.of.Spl.Chars, FUN =
function(x) mean(x, na.rm = 'TRUE')),dataset$X.No.of.Spl.Chars)
dataset$Strength = ifelse(is.na(dataset$Strength),ave(dataset$Strength, FUN = function(x) mode(x, na.rm =
'TRUE')),dataset$Strength)
as.data.frame(table(dataset$Strength))
sum(is.na(dataset))
sum(is.na(dataset$Strength))
dataset$Strength=as.factor(dataset$Strength)
dataset$Length=as.numeric(dataset$Length)
```

```
#Heatmap
d1=dataset[,c(1,2,3,4,5)]
res <- cor(d1)
cormat= round(res, 2)
library(reshape2)
melted_cormat <- melt(cormat)

get_lower_tri<-function(cormat){
  cormat[upper.tri(cormat)] <- NA
  return(cormat)
}

get_upper_tri <- function(cormat){
  cormat[lower.tri(cormat)]<- NA
  return(cormat)
}

upper_tri <- get_upper_tri(cormat)
melted_cormat <- melt(upper_tri, na.rm = TRUE)
ggplot(data = melted_cormat, aes(Var2, Var1, fill = value))+
  geom_tile(color = "white")+
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                       midpoint = 0, limit = c(-1,1), space = "Lab",
                       name="Pearson\nCorrelation") +
  theme_minimal()+
  theme(axis.text.x = element_text(angle = 45, vjust = 1,
                                   size = 12, hjust = 1))+
  coord_fixed()




#Detecting Outliers

boxplot(dataset)
#install.packages("outliers")
library(outliers)

outlier_tf = outlier(dataset$Length,logical=TRUE)
sum(outlier_tf)
#What were the outliers
find_outlier = which(outlier_tf==TRUE,arr.ind=TRUE)
#Removing the outliers
dataset = dataset[-find_outlier,]
nrow(dataset)

outlier_tf = outlier(dataset$X.No.of.Upper.Case,logical=TRUE)
sum(outlier_tf)
#What were the outliers
find_outlier = which(outlier_tf==TRUE,arr.ind=TRUE)
#Removing the outliers
dataset = dataset[-find_outlier,]
nrow(dataset)

outlier_tf = outlier(dataset$X.No.of.Lower.Case,logical=TRUE)
sum(outlier_tf)
#What were the outliers
find_outlier = which(outlier_tf==TRUE,arr.ind=TRUE)
#Removing the outliers
dataset = dataset[-find_outlier,]
nrow(dataset)

outlier_tf = outlier(dataset$X.No.of.Numbers,logical=TRUE)
sum(outlier_tf)
#What were the outliers
```

```
find_outlier = which(outlier_tf==TRUE,arr.ind=TRUE)
#Removing the outliers
dataset = dataset[-find_outlier,]
nrow(dataset)

outlier_tf = outlier(dataset$X.No.of.Spl.Chars,logical=TRUE)
sum(outlier_tf)
#What were the outliers
find_outlier = which(outlier_tf==TRUE,arr.ind=TRUE)
#Removing the outliers
dataset = dataset[-find_outlier,]
nrow(dataset)

dim(dataset)
str(dataset)
boxplot(dataset)
# Splitting the dataset into the Training set and Test set

# install.packages('caTools')
library(caTools)
set.seed(123)
split = sample.split(dataset$Strength, SplitRatio = 0.75)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)
#Normalizaton
training_set[-6] = scale(training_set[-6])
test_set[-6] = scale(test_set[-6])


#Applying Machine Learning Models


#SVM
# Fitting SVM to the Training set
# install.packages('e1071')
library(e1071)
classifier_svm = svm(formula = Strength ~ .,
                 data = training_set,
                 type = 'C-classification',
                 kernel = 'linear')

# Predicting the Test set results
y_pred = predict(classifier_svm, newdata = test_set[-6])

# Making the Confusion Matrix
confusionMatrix((y_pred),test_set[,6])


#Kernel SVM
# Fitting Kernel SVM to the Training set
# install.packages('e1071')
library(e1071)
classifier_ksvm = svm(formula = Strength ~ .,
                 data = training_set,
                 type = 'C-classification',
                 kernel = 'radial')

# Predicting the Test set results
y_pred = predict(classifier_ksvm, newdata = test_set[-6])

# Making the Confusion Matrix
confusionMatrix((y_pred),test_set[,6])


#naiveBayes
# Fitting naiveBayes to the Training set
# install.packages('e1071')
```

```
library(e1071)
classifier_nb = naiveBayes(x = training_set[-6],
                           y = training_set$Strength)

# Predicting the Test set results
y_pred = predict(classifier_nb, newdata = test_set[-6])

# Making the Confusion Matrix
confusionMatrix((y_pred),test_set[,6])


#RandomForest
#install.packages("randomForest")
#install.packages("party")
library(party)
library(randomForest)

#rf = randomForest(Strength
~Length+X.No.of.Upper.Case+X.No.of.Lower.Case+X.No.of.Numbers+X.No.of.Spl.Chars,data=training_set)
rf <- ctree(Strength ~Length+X.No.of.Upper.Case+X.No.of.Lower.Case+X.No.of.Numbers+X.No.of.Spl.Chars,
            data=training_set)

y_pred = predict(rf, newdata=test_set[-6])
confusionMatrix((y_pred),test_set[,6])
plot(rf,type="simple")


#Single prediction

#install.packages('data.table')
library(stringr)
library(data.table)

str=readline(prompt = "Enter Password to find strength: ");
length = nchar(str)
upp=str_count(str, "[A-Z]")
loo=str_count(str, "[a-z]")
num=str_count(str, "[0-9]")
spec=length-(upp+loo+num)

vec<-c(length,upp/length,loo/length,num/length,spec/length)

new_df=data.frame(vec)
new_df=transpose(new_df)

names(new_df)[1] <- "Length"
names(new_df)[2] <- "X.No.of.Upper.Case"
names(new_df)[3] <- "X.No.of.Lower.Case"
names(new_df)[4] <- "X.No.of.Numbers"
names(new_df)[5] <- "X.No.of.Spl.Chars"
new_df[] = scale(new_df)
print("Prediction form SVM")
y_pred = predict(classifier_svm, newdata = new_df)
print(y_pred)

print("Prediction form Kernel SVM")
y_pred = predict(classifier_ksvm, newdata = new_df)
print(y_pred)

print("Prediction form Naive Bayes")
y_pred = predict(classifier_nb, newdata = new_df)
print(y_pred)

print("Prediction form Random Forest")
y_pred = predict(rf, newdata = new_df)
print(y_pred)
```

```
#Stats
#install.packages('caret')


#Accuracy Graph
library(RColorBrewer)
coul <- brewer.pal(4, "Set2")

barplot(c(0.9533 , 0.9612 , 0.939,0.85 ),
      main = "Accuracy Comparison",
      xlab = "ML Models",
      ylab = "Accuracy Percentage",
      names.arg = c("SVM", "SVM(Radial Kernel)", "Naive Bayes","RandomForest"), ylim = c(0.70,1.0),
xpd=FALSE,col=coul)


#Precision Graph

barplot(c(0.8999, 0.8933, 0.8011,0.85),
      main = "Class 0 Precision Comparison",
      xlab = "ML Models",
      ylab = "Class 0 Precision Percentage",
      names.arg = c("SVM", "SVM(Radial Kernel)", "Naive Bayes","RandomForest"), ylim = c(0.50,1.0),
xpd=FALSE,col=coul)

barplot(c(0.9684, 0.9685, 0.9614,0.8222),
      main = "Class 1 Precision Comparison",
      xlab = "ML Models",
      ylab = "Class 1 Precision Percentage",
      names.arg = c("SVM", "SVM(Radial Kernel)", "Naive Bayes","RandomForest"), ylim = c(0.80,1.0),
xpd=FALSE,col=coul)

barplot(c(0.9083, 0.9831, 0.9338,0.5588),
      main = "Class 2 Precision Comparison",
      xlab = "ML Models",
      ylab = "Class 2 Precision Percentage",
      names.arg = c("SVM", "SVM(Radial Kernel)", "Naive Bayes","RandomForest"), ylim = c(0.5,1),
xpd=FALSE,col=coul)




#Sensitivity Graph

coul <- brewer.pal(5, "Set2")

barplot(c(c(0.8999),c(0.8933),c(0.80119),0.54852),
      main = "Class 0 Sensitivity",
      xlab = "ML Models",
      ylab = "Sensitivity",
      names.arg = c("SVM", "SVM(Radial Kernel)", "Naive Bayes","RandomForest"), ylim = c(0.50,1.0),
xpd=FALSE,col=coul)

barplot(c(c(0.9685),c(0.9686),c(0.9614),0.8222),
      main = "Class 1 Sensitivity",
      xlab = "ML Models",
      ylab = "Sensitivity",
      names.arg = c("SVM", "SVM(Radial Kernel)", "Naive Bayes","RandomForest"), ylim = c(0.80,1.0),
xpd=FALSE,col=coul)

barplot(c(c(0.9084),c(0.9832),c(0.9339),0.9089),
      main = "Class 2 Sensitivity",
      xlab = "ML Models",
```

```
      ylab = "Sensitivity",
      names.arg = c("SVM", "SVM(Radial Kernel)", "Naive Bayes","RandomForest"), ylim = c(0.90,1.0),
xpd=FALSE,col=coul)




#Specificity Graph
coul <- brewer.pal(5, "Set1")
barplot(c(c(0.9862),c(0.9866),c(0.98536),0.97308),
      main = "Class 0 Specificity",
      xlab = "ML Models",
      ylab = "Specificity",
      names.arg = c("SVM", "SVM(Radial Kernel)", "Naive Bayes","RandomForest"), ylim = c(0.95,1.0),
xpd=FALSE,col=coul)

barplot(c(c(0.9043),c(0.9375),c(0.8688),0.7507),
      main = "Class 1 Specificity",
      xlab = "ML Models",
      ylab = "Specificity",
      names.arg = c("SVM", "SVM(Radial Kernel)", "Naive Bayes","RandomForest"), ylim = c(0.70,1.0),
xpd=FALSE,col=coul)

barplot(c(c(0.9863),c(0.9861),c(0.9805),0.8662),
      main = "Class 2 Specificity",
      xlab = "ML Models",
      ylab = "Specificity",
      names.arg = c("SVM", "SVM(Radial Kernel)", "Naive Bayes","RandomForest"), ylim = c(0.80,1.0),
xpd=FALSE,col=coul)



#F1 Score
coul <- brewer.pal(4, "Set3")
barplot(c(c((2*0.8999),(2*0.8933),(2*0.8011),(2*0.85))*c(0.8999, 0.8933,
0.80119,0.54852))/(c((0.8999),(0.8933),(0.8011),(0.85))+c(0.8999, 0.8933, 0.80119,0.54852)),
      main = "Class 0 F1 Score",
      xlab = "ML Models",
      ylab = "F1 Score",
      names.arg = c("SVM", "SVM(Radial Kernel)", "Naive Bayes","RandomForest"), ylim = c(0.4,.9),
xpd=FALSE,col=coul)

barplot(c(c((2*0.9684),(2*0.9685),(2*0.9614),2*0.8222)*c(0.9685,0.9686,0.9614,0.8222)/c((0.9684),(0.9685),(0.
9614),0.8222)+c(0.9685,0.9686,0.9614,0.8222)),
      main = "Class 1 F1 Score",
      xlab = "ML Models",
      ylab = "F1 Score",
      names.arg = c("SVM", "SVM(Radial Kernel)", "Naive Bayes","RandomForest"), ylim = c(1.975,3),
xpd=FALSE,col=coul)

barplot(c(c((2*0.9083),(2*0.9831),(2*0.9338),2*0.5588)*c(0.9084,0.9832,0.9339,0.9089))/(c((0.9083),(0.9831),(
0.9338),0.5588)+c(0.9084,0.9832,0.9339,0.9089)),
      main = "Class 2 F1 Score",
      xlab = "ML Models",
      ylab = "F1 Score",
      names.arg = c("SVM", "SVM(Radial Kernel)", "Naive Bayes","RandomForest"), ylim = c(0.5,1),
xpd=FALSE,col=coul)




#Ploting Dataset Coloumns

#Modifying Dataset for Plotting Graphs
```

```
for (i in 1:nrow(dataset)) {
  dataset$X.No.of.Upper.Case[i]=dataset$X.No.of.Upper.Case[i]*dataset$Length[i]
  dataset$X.No.of.Lower.Case[i]=dataset$X.No.of.Lower.Case[i]*dataset$Length[i]
  dataset$X.No.of.Numbers[i]=dataset$X.No.of.Numbers[i]*dataset$Length[i]
  dataset$X.No.of.Spl.Chars[i]=dataset$X.No.of.Spl.Chars[i]*dataset$Length[i]}


plot(dataset$Length, dataset$Strength, main = "Length vs Strength",
     xlab = "Length of Password", ylab = "Strength of Password",
     pch = 19)

plot(dataset$X.No.of.Upper.Case, dataset$Strength, main = "No. of Upper Case Characters vs Strength",
     xlab = "No. of Upper Case Characters", ylab = "Strength of Password",
     pch = 19)

plot(dataset$X.No.of.Lower.Case, dataset$Strength, main = "No. of Lower Case Characters vs Strength",
     xlab = "No. of Lower Case Characters", ylab = "Strength of Password",
     pch = 19)

plot(dataset$X.No.of.Numbers, dataset$Strength, main = "No. of Number Characters vs Strength",
     xlab = "No. of Number Characters", ylab = "Strength of Password",
     pch = 19)

plot(dataset$X.No.of.Spl.Chars, dataset$Strength, main = "No. of Special Characters vs Strength",
     xlab = "No. of Special Characters", ylab = "Strength of Password",
     pch = 19)
```