Feature Importance Ranking with Neural Networks and Permutation Feature Importance

COMP-4730 Project 2

1st Aran Abrahamlingam
dept. Computer Science
University of Windsor
Windsor, Ontario
abrah11l@uwindsor.ca

2nd David Dagati
dept. Computer Science
University of Windsor
Windsor, Ontario
dagati@uwindsor.ca

3rd Peter Zoura
dept. Computer Science
University of Windsor
Windsor, Ontario
zourap@uwindsor.ca

***Abstract*—In today's connected world, cyber security is of the utmost importance, and at the very base of this, is the password. Having a strong password provides users with a safe experience whether online, or on a local computer. Determining what characteristics of a password make one stronger than another would allow people to better their password practices. Together we have used a dataset of passwords with a rank of weak, normal, or strong, to create features representing the characteristics of passwords. From these features, we then wanted to learn how to rank them, determining which characteristics were most important to a strong password. We next wanted to learn what would happen when we started introducing all of the other features. To accomplish this we used many machine learning models such as decision tree and their ensemble random forest, a neural network mode, and through the Permutation Feature Importance method we used Gaussian Naive Bayes, Logistic Regression, and K-Nearest Neighbours as well. From our decision tree's feature importance rankings and the neural network's weight distribution, we found that the three most important features were "Length," "Numbers," and "Uppers." These findings were also enforced by Permutation Feature Importance, which showed us that the most important characteristic of a password was its length. We also learned from Permutation Feature Importance that the second and third-place features could differ slightly depending on which model was used. Together these questions acted as both a way to learn about the importance of creating strong passwords and through machine learning about feature importance ranking between multiple models.**

***Index Terms*—Decision Tree, Random Forest, Neural Network, Permutation Feature Importance, Feature Ranking, Password Strength**

I. Introduction

In the field of machine learning, data science researchers deal with millions of data samples to train powerful AI. The number of features that each data sample records can be anywhere from twenty to three million or more. Having access to millions of features of real data is beneficial to the discovery of prediction methods for real life problems. Each feature that has some correlation to the target variable provides a little bit of info that machine learning models can make use of to determine an accurate prediction of the target value. Some features being totally uncorrelated could throw off a machine learning model or cause the training time to increase so that the model can learn which features provide no information about the target. Cutting down on the number of features could greatly improve machine learning model prediction accuracy and training times as well as the time it takes to predict on new data. Analysing the importance of each feature will allow us to be informed about which features should be removed to improve AI model accuracy by identifying which features benefit or detriment the prediction. Feature importance analysis can also help us learn about the real world by identifying important factors in a problem domain from the important features. One use case of feature selection is identifying which subset of genes can be used to predict cancer diagnosis (Guyon et al.). The number of features can be cut down using Principle Component Analysis, which projects the data into a lower dimensionality. This makes the data easier to work with, however it means that each feature in the new featureset does not directly correspond to the features in the original featureset. Therefore this method does not work for feature importance ranking. Rather we want to determine the value of each real feature which corresponds to real world data.

This paper will research a variety of methods to determine which features in a dataset are most important and whether the less-important features provide any help or hindrance to the model's prediction accuracy. Although we are only testing with one dataset, the research process used here can be used in future research to determine which method is most effective in determining important features and worthless features if there are any. We chose a simple dataset that has one input feature, a password, and three classes of output, which are weak, normal, and strong. In order to work with this dataset, the passwords need to be analysed and features must be extracted. We generated fourteen features (or twenty after one-hot encoding), and tested feature importance by using Decision Trees, Neural Networks, Gaussian Naive Bayes, Logistic Regression, and K-Nearest Neighbours.

Decision Trees use feature ranking when fitting the data, so by extracting this ranking we are able to see which features were valued more and which were less valued. Neural

Networks were used by extracting the weight vectors of the first layer and taking note of which features had a higher weight coefficient given to it. Logistic Regression, K-Nearest Neighbours, Decision Trees, and Naive Bayes classifiers were also used with the permutation feature importance method, which is done by fitting a classifier to data, then iterating through the features, shuffling the feature, and refitting the classifier to see how much the accuracy drops as a result of ruining that feature's data by shuffling it.

II. State of the Art

In 2002, four researchers, Isabelle Guyon, Jason Weston, Stephen Barnhill of Barnhill Bioinformatics, and Vladimir Vapnik of AT&T labs (Guyon et al.) made use of feature importance analysis and DNA micro-array data to determine the activity of certain genes in cancer tissue. Their study used Recursive Feature Elimination and Support Vector Machines to analyse the data. Recursive Feature Elimination works by analysing the importance (either by weights or other importance metrics) of each feature in the data that was fit to the classifier and it prunes the least important features, then retrains the classifier and repeats. They used this method on DNA micro-arrays, which is an analysis method that allows scientists to determine which genes are on or off in a given cell. They found that using SVMs with Recursive Feature Elimination was effective in discovering genes that are related to cancer.

In 2019 the permutation feature importance measure proposed by Richard Breiman for random forests research in 2001 was expanded by researchers Aaron Fisher at Takeda Pharaceuticals, Cynthia Rudin at Duke University in the department of Computer Science, and Francesca Dominici at Harvard in the department of Biostatistics (Fisher et al.). They see that permutation importance does differ in the results it produces based on what kind of classifier it is used with, as different classifiers may value particular features differently. They propose Model Class Reliance as a measure of feature importance for a range of machine learning models. MCR is defined as "the highest and lowest degree to which any well-performing model within a given class may rely on a variable of interest for prediction accuracy". In other words, their metric is defined as an upper and lower bound on the reliance of any effective machine learning model on a particular variable, with the added stipulation that the potential models used are within the same class. They discover that MCR is a better measure of importance than traditional methods of feature importance ranking.

In 2020, Maksymilian A.Wojtas a PHD student at the University of Manchester and Ke Chen an academic staff member of the School of Computer Science at the University of Manchester (Wojtas and Chen) developed a new method of feature importance ranking that uses two Neural Networks. One Neural network is called the operator, this network is trained for supervised learning with potentially optimal feature subsets. The subsets are decided on by the other neural network, which is called the selector. While this happens, the selector learns to predict the performance of the operator. By training the selector network, the best subset can be discovered and the ranking of those features in the subset can also be discovered. In their paper, Wojtas and Chen showed that their method outperforms other feature selection methods such as Conditional Covariance Minimization, Least Angle Regression, and Genetic Algorithm methods for feature selection. However, one drawback is the heavy computation cost of the training of two neural networks.

In June of this year, a paper was published in the Soft Computing journal detailing the research of two faculty members of the Department of Computer Science at Minia University in El-Minia, Egypt. Heba Farghaly and Tarek El-Hafeez (Farghaly and El-Hafeez) used an association analysis method to select features that are highly related to the target variable and to eliminate redundant features. Their motivation was to find a feature selection algorithm that can be used with large amounts of data while remaining efficient and to find a provide another method of dimensionality reduction, because many fields of huge data processing suffer from having too high dimensionality in the data. The method they propose is able to extract features that are correlated and strongly related to the target variable using an association analysis method. They found that the association analysis method they developed was very effective in reducing the number of features used for prediction and maintained quality and efficiency. In the data they tested with, which was an SMS spam collection database, they were able to identify a minimal subset of features that produces an equal prediction accuracy in the classifier, using only 7% of the total features in the data.

III. Creating the Dataset

A. Maintaining the Integrity of the Specifications

The dataset contained 600 000 passwords, we decided to randomly sample this to make a dataset of 25 000 passwords for the sake of decreasing the training time. After randomly sampling the passwords, the 25 000 password subset is input to a feature generator program which passes through the data once, generating features for each password. The program is written in Python and makes use of the Natural Language Toolkit to compare password contents with a list of English words, as well as the Pandas library to contain and manipulate data. The program uses counters and a for loop that passes through the data once to determine the password's

1) Length = Length of the password
2) #Uppers = Number of uppercase letters
3) #Lowers = Number of lowercase letters
4) #Numbers = Number of digits
5) #Symbols = Number of symbols
6) SLG_uppers = Size of largest group of uppercase letters

7) SLG_lowers = Size of largest group of lowercase letters,
8) SLG_numbers = Size of largest group of numbers
9) SLG_symbols = Size of largest group of symbols

The following 8 features are one-hot encodings.

10) FCT_Up = 1 if the first character is an uppercase letter
11) FCT_Lo = 1 if the first character is a lowercase letter
12) FCT_Nu = 1 if the first character is a number
13) FCT_Sy = 1 if the first character is a symbol

14) LCT_Up = 1 if the last character is an uppercase letter
15) LCT_Lo = 1 if the last character is a lowercase letter
16) LCT_Nu = 1 if the last character is a number
17) LCT_Sy = 1 if the last character is a symbol

18) #words = Number of English words in the password
19) ave_size_of_words = Average size of those words
20) palindrome = 1 if the password is a palindrome

The features representing the type of the first character and last character are one-hot encoded. This means that instead of representing the type of the first character as one of four labels, it is represented as a vector of 4 binary digits [0 0 1 0], with each component of the vector representing one of four labels. The vector will contain three 0s and a single 1 to distinguish which label is set. Since this adds 6 more features to the data, there are 20 features generated.

There are two reasons for using one-hot encoding. The first reason is that if we represent the different labels as strings which name the label (e.g. label "u" for uppercase, or "s" for the symbol) then the machine learning models which use mathematics and statistics will not be able to interpret these features. The simplest option is to encode each label as a number (i.e. label 0 for uppercase, 1 for lowercase and so on) however this will cause the machine learning models to treat lowercase letters as a higher number value than uppercase letters when in reality there is no ordinal relationship between those labels. For those two reasons, one-hot encoding is the best option, as it makes the labels interpretable and equally values each label.

After processing, the feature generator saves the new data into a file. The data is stored in a tab-separated value file (.tsv) because some of the passwords contain commas.

IV. Methods

A. Decision Tree and Random Forest

Methodology: Decision tree is a classification model of supervised machine learning. This model is presented as a tree, consisting of root nodes, branches, decision nodes, and leaf nodes. The base of a decision tree is the root node, this is the beginning of the model's journey to classification. From the root node down, branches are created to flow down left and right if the condition found in the above node resulted in true or false. These internal nodes and branches continue until they end on a leaf node. The leaf node is final, classifying the data point as one of the defined classes. Along the way in developing a decision tree, entropy, the measure of uncertainty in the dataset, and information gain, the reduction of uncertainty, are used to help determine what feature should be the root node, and what features should follow after.

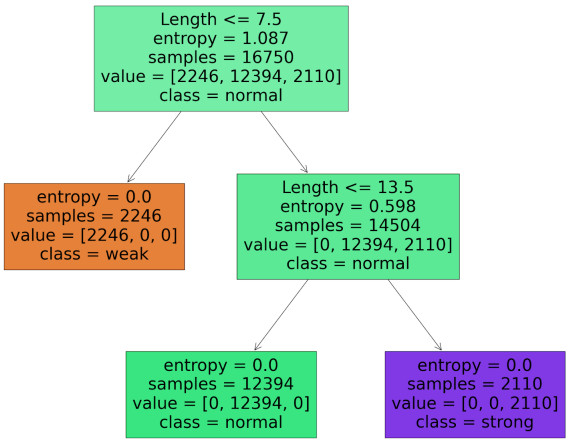
Random forest is the ensemble model of decision tree classifiers. It uses the concept of bagging which is creating a subset of training data based on random training data, then testing this 'bag', and creating another random bag to test that includes the data points that tested poorly in the previous bags. In random forest, the bagging is taken further through the use of feature bagging. This is when only a small random sample of features enforces a low correlation between all decision trees in the random forest model.

To solve our research questions we decided to use a decision tree to compare the results of the root node, and the top 2 decision nodes, with the results of a neural network. This would show us the top 3 features from our created dataset. Another method we used is looking at the feature_importances_ attribute recorded by the DecisionTreeClassifier in the sci-kit-learn library. This is an array containing the importance value for each feature, this importance value is called gini importance, and is defined as the normalized total reduction of the impurity brought about by choosing that feature. Our second question looks more at how the decision tree model handles an excess of information. Once we determine our best 3 features, does the model improve when given the next 17 features as well? If it does, then by how much, and what do we see from the visual point of view of the decision tree image? We also used the introduction of random forest to confirm our results using an ensemble method. This allowed for us to test more than just a single decision tree, but upwards of thousands of trees each comparing for the best 3 features and what difference the other 17 make to the process. Using the ensemble mode random forest, we also add in a minor third research question. We were curious to see if our dataset would return the same results as our decision tree, and if we would notice the use of feature bagging when searching for our top 3 features. This may also interact with the 17 other features being introduced and how they are in our final tree.

Results: We begin, we started with the decision tree. We trained this model on our dataset with the hyperparameters: random_state= 400 and criterion= entropy. This made the decision tree use the entropy formula to determine the information gain. This information gain was then used to help measure the quality of the split. With these hyperparameters and our dataset, we were able to achieve 100% accuracy. The output can be found in the figure DTF1.

Here we observe from top to bottom our top features of this decision tree. The length of the password is found to be the only feature. No other features were required to achieve 100%.

Fig. 1. Optimal Decision Tree



From this first decision tree, we learned how quickly we can maximize classifying the passwords with our created dataset. // After training a set of decision trees with six different random seeds, the feature_importance_ array was looked at to find the average rankings. We found that the Length feature was given an importance value of 1, and the other features were given importance of 0. The reason for this will be explained in the Permutation Feature Importance section. In order to get a ranking for the other features, we forced the model to use them by dropping the Length feature from the dataset. After doing this, the first four features listed by the six trees in descending order of importance are the same on average: Length, #Uppers, #Numbers, #Lowers. This shows that despite having different random seeds, the output did not change. The table showing the top six features of each tree is seen below.

random_state:	0	1	2
First	Length	Length	Length
Second	#Uppers	#Uppers	#Uppers
Third	#Numbers	#Numbers	#Numbers
Fourth	#Lowers	#Lowers	#Lowers
Fifth	SLG_lowers	#Symbols	SLG_symbols
Sixth	SLG_symbols	SLG_symbols	#Symbols

3	4	5	6
Length	Length	Length	Length
SLG_uppers	SLG_uppers	SLG_uppers	#uppers
#Numbers	#Numbers	#Numbers	#Numbers
#Lowers	#Lowers	#Lowers	#Lowers
SLG_lowers	SLG_symbols	SLG_lowers	SLG_lowers
#Symbols	SLG_numbers	SLG_symbols	SLG_symbols

TABLE I
RESULTS OF SUCCESSIVELY RANDOM SEEDED DECISION TREES

Following the success of decision trees, we next turned to random forest. This next step allowed us to create a full ensemble of decision trees to learn from one another through bagging. In order to determine the best hyperparameters with random forest we used the sci-kit learn's Grid Search. This grid search allowed us to set a dictionary of random forest parameters as the keys and a list of possible values for that parameter. This grid search can be seen in the table below.

These parameters then generated many different random forest models, each creating a great number of decision trees. When this completed running we had it export the best hyperparameters. These were:

With these hyperparameters we were able to achieve 100% accuracy in the classification of our passwords dataset. This resulted in the final decision tree from the optimized random forest seen in Figure DTF2.

This decision tree is quickly able to classify a password as weak with one simple decision (root) node. From there this tree is able to break down a password's features to determine if it is either normal or strong. From this decision tree, and all others we observed through this process, we were able to answer our research questions

TABLE II
RANDOM FOREST SETS OF INPUT HYPERPARAMETERS

Feature	**Set 1**	**Set 2**	**Set 3**	**Set 4**	**Set 5**
max_depth	3	5	7	10	None
n_estimators	10	100	200	1000	
max_features	1	3	5	7	
min_samples_leaf	1	2	3		
min_samples_split	1	2	3		
max_leaf_nodes	1	2	3	4	5

Feature	**Set 1**	**Set 2**	**Set 3**	**Set 4**	**Set 5**
max_depth		**5**			
n_estimators		**100**			
max_features			**5**		
min_samples_leaf	**1**				
min_samples_split		**2**			
max_leaf_nodes					**5**

TABLE III
RANDOM FOREST BEST HYPERPARAMETERS

Research Question 1: Determine the Best 3 Features: From our most optimal decision tree created by the random forest model, we were able to look top-down, finding the top three features. This was possible because the order in which they were used in the decision tree is based on the information gain and gini index values. The top three features determined by the random forest were: Length, SLG_uppers, and #Uppers. The most optimized tree also uses the features #Lowers, and SLG_lowers to determine if each password is either normal strength or strong. These were however not included in the top three features to reach a rate of 100% classification. The top three features determined from the basic decision trees by looking at the feature_importances_ array are Length, #Uppers, and #Numbers or SLG_uppers in third place. The two methods we used to analyze decision trees are in agreement that Length is the most important and #Uppers is either the second or third most important, but they disagreed on the third feature being either SLG_lowers or #Numbers.

Research Question 2: Is the Classifier Helped or Hindered by Including the Other Features?: When running a modified

random forest with all features included, there is no difference in accuracy. This was due to the 100% accuracy rate already achieved with the original optimized random forest model. When creating plain decision trees, we found that the Length attribute alone and 2 nodes were enough to fully classify all the passwords with 99% or more accuracy. In general, only the Length feature is necessary, and including the other attributes neither improves nor impairs the prediction accuracy. From this, we can determine that when people are creating their passwords and they want a simple way to ensure they are protected, when in doubt always make the password longer. Our model considered anything over a length of 13.5 characters to be extremely strong, meaning a 14-character password should suffice in difficulty and complexity to create a safe and protected presence online or on a local computer.

B. Neural Network

Methodology: We used a neural network in the form of a Multi-Layer Perceptron (MLP) classifier to examine the predictive power of the features in our dataset. Our methodology involved configuring the MLP with a specific architecture: three hidden layers with descending neuron counts of 100, 50, and 30, respectively. Tanh, an activation function well-known for its non-linear characteristics and backpropagation efficiency, was selected. We used the 'adam' solver for weight optimization due to its adaptive learning rate capabilities, and we incorporated a regularization term (alpha = 0.001) to mitigate the risk of overfitting.

Early stopping of the training process was applied in order to accelerate the training phase and further prevent overfitting. We used a rigorous methodology to make sure that every feature had an equal chance of influencing the model's decisions by normalizing them all. To ensure the validity of our conclusions, we employed a train-test split to assess model performance on omitted data.

Results: The machine learning model demonstrated remarkable predictive performance on the test set with a high accuracy rate of approximately 99.87% after training. We used permutation importance analysis, a method that assesses the drop in model accuracy when the data of each feature is randomly permuted, to determine the significance of each feature. The most significant feature was found to be "Length," followed by "#Numbers" and "#Uppers," demonstrating the disproportionate influence of certain features. We also looked at top 3 features according to highest weight given in the first layer. Since there are 100 neurons in the first layer, it is necessary to take an average. We found that according to this method the top 3 features are #Lowers, #words, and LCT_Up. We tested NN prediction accuracy with both subsets and with all the features.

TABLE IV
FEATURE RANKING FOR NEURAL NETWORKS

	Importance	Std
Length	0.3292969696969697	0.0036313972820077828
#Numbers	0.10800404040404045	0.0019204844132649729
#Uppers	0.0786464646464647	0.001766839628613014

Features	Avg Weight In First Layer
#Lowers	0.026919058776304485
#words	0.018230552061529966
LCT_Up	0.016525415344952245
LCT_Sy	0.015882309851143896
ave_size_of_words	0.012963004073114863

TABLE V
TOP FEATURES ACCORDING TO AVERAGE WEIGHT IN FIRST LAYER

Accuracy of all features	0.9986666666666667
Accuracy of top 3 features by PI	0.9993939393939394
Accuracy of top 3 features by average weight	0.7881212121212121

TABLE VI
ACCURACY OF NEURAL NETWORK WITH AND WITHOUT THE LESS IMPORTANT FEATURES

From these results we see that the feature subset determined by looking at the top 3 average weighted features in the first layer did not produce a good neural network when we retrained using only those features. Instead, the accuracy dropped by about 20%. However, the neural network that was trained only on the top 3 features identified by the permutation importance method had scored an equal accuracy, while using only 15% of the features. From this data we can conclude that analysing the weights given to each feature by the neurons in the first layer of a neural network can not accurately determine the importance of features.

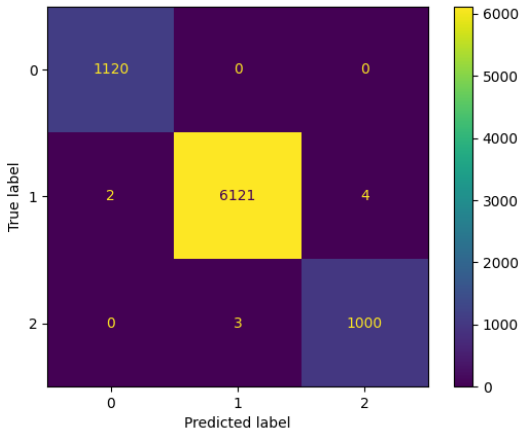


Fig. 2. Figure. Neural Network Confusion Matrix.

Research Question 1: Determine the Best 3 Features: When using neural networks, the permutation feature importance method of feature importance ranking is very effective. However this method does not necessarily need to be used with neural networks, with Logistic Regression we found the same 3 features (Length, #Numbers, and #Uppers) to be most important by the permutation feature importance method. We found that the method was significantly better than weight vector analysis. When using average weight ranking to determine the top 3 features, a neural network trained on those 3 features had 20% lower prediction accuracy.

Research Question 2: Is the Classifier Helped or Hindered by Including the Other Features?: The second research question explored the impact of excluding the less significant 17 features from the model. We saw an accuracy of about 99.94% when we retrained the neural network using just the three important features, which was higher than the accuracy we got using the entire feature set. This finding implies that the features that were eliminated were either less informative or redundant. Therefore, by eliminating them, the model became more streamlined and effective while slightly improving predictive accuracy. This result highlights how important feature selection is for enhancing model performance and cutting down on computational complexity.

C. Dual-Net

Methodology: Two neural networks are used, one is called the operator, and the other is called the selector. The selector network will choose feature subsets for the operator to train with. The operator will try to learn the features and predict the target. The selector network will try to learn the operator's accuracy with a given subset, and the importance ranking of each feature in the subset. The method uses deep neural nets and aims to find a fixed-size optimal feature subset that maximizes the prediction accuracy of deep neural networks on the data. The code for this method was made available on Git Hub by the researchers who developed it at the University of Manchester. We used this repository and plugged in our dataset to the dual-net feature importance ranking program. The method takes quite some time longer than the other methods discussed in this paper, as each iteration is training the operator deep neural network with a different subset of the features. The operator network is created with hidden layer sizes of 60, then 30, then 20, and one node in the output layer. The selector network is created with hidden layer sizes of 100, then 50, then 10, and 1 node in the output layer. Since there are three classes to predict in our data, the final layer of the operator network should actually have 3 nodes. However the dual-net program would not run in that configuration, so for that reason, we did not achieve the optimal setup.

Results: The network produced a different optimal subset of features each time it was run. It was not successful in accurately predicting the importance of each feature in the data. One potential reason for this may be the final layer of the operator network. Since there are 3 classes to predict in our data the operator network should have had 3 output nodes. It could be that having only one node meant that the operator could not properly produce a prediction, then the selector network would not be able to accurately learn the prediction accuracy given a subset.

Research Question 1: Determine the Best 3 Features: We found that the dual-net system was not able to discover the top 3 most important features. In the research paper detailing the dual-net method (Wojtas and Chen) the researchers found success using their model on artificial and real-world data. We conclude that our set-up was not optimal and does not accurately reflect the effectiveness of the dual-net feature importance ranking method.

D. Permutation Feature Importance

Methodology: Permutation Feature importance is a method for evaluating how important each feature in a dataset is to the prediction accuracy of the model. This is accomplished by iterating through the features, training and testing the data in each iteration, but shuffling the feature which corresponds to the current iteration in order to break its correlation with the target variable. The method can be used with any classifier model. We used the sci-kit learn Python library to run this algorithm using the following classifiers: Decision Tree, Gaussian Naive Bayes, K-Nearest Neighbors, and Logistic Regression.

The algorithm works by taking a classifier that has been fit to data and testing it another F times (F is the number of input features). In each test, one feature is shuffled. Since the feature no longer has the same relationship to the output variable, the classifier will likely lose accuracy. This accuracy loss is measured for each feature that was shuffled, and this value is then used to rank the features. Whichever feature caused the classifier to lose the most accuracy when the feature was shuffled, is considered the most important.
Using the sci-kit-learn library function permutation_importance() we tested on the training data first, and then we tested on the testing data. When we use the permutation importance method on testing data, we get the added info of knowing which features are most important in generalizing the model to unknown data. In each test, the parameters used are:

1) estimator = DecisionTree(random_state=47, criterion = "entropy")
   X = X_train
   y = y_train
   n_repeats=10
   random_state=72
2) estimator = LogisticRegression(random_state=472, solver = "newton-cholesky")
   X = X_train
   y = y_train
   n_repeats=10
   random_state=72
3) estimator = GaussianNB() X = X_train
   y = y_train
   n_repeats=10
   random_state=72
4) estimator = KNeighborsCLassifier(n_neighbors=2)

These 4 tests were then repeated using X = X_test, y = y_test

In each test, the classifier was created and fit before being passed to the permutation_importance() function.

In the tests for the decision tree, the Length attribute was omitted to force the test to produce results for the other features, because when the Length feature was an option, the decision tree would classify Length as the only important feature and assign the others an importance value of 0. The reason for this is that when the decision tree has access to the Length attribute, it picks this attribute because it is the most useful one, and since this attribute is enough to fully predict the password strength, removing the other features or shuffling them will have no effect on the accuracy, so the importance of every other attribute is 0.

TABLE VII
RESULTS OF PERMUTATION FEATURE IMPORTANCE TESTING ON PREDICTING TRAINING DATA

	DT	LR	GNB	KNN
first	Length	Length	#Uppers	Length
second	#Numbers	#Numbers	FCT_Up	#Lowers
third	#Lowers	#Uppers	LCT_Up	#Numbers
fourth	#Uppers	#Lowers	Length	SLG_lowers
fifth	SLG_lowers	SLG_uppers	SLG_uppers	FCT_Up
sixth	SLG_symbols	SLG_lowers	#Symbols	SLG_numbers

	DT	LR	GNB	KNN
first	Length	Length	#Uppers	Length
second	#Numbers	#Numbers	FCT_Up	#Lowers
third	#Lowers	#Uppers	LCT_Up	#Numbers
fourth	#Uppers	#Lowers	SLG_uppers	SLG_lowers
fifth	SLG_lowers	SLG_uppers	#Symbols	FCT_Up
sixth	SLG_symbols	SLG_lowers	Length	SLG_numbers

TABLE VIII
RESULTS OF PERMUTATION FEATURE IMPORTANCE TESTING ON PREDICTING TESTING DATA

Results: We found that the top 3 most important features were not agreed upon by each classifier. But by considering each classifier's given top features as a vote, we find that the top 3 most important features in descending order of importance are Length, #Numbers, and a 4-way tie for the third best feature. Each classifier used different features out of the less important ones. Since some of the methods such as Logistic Regression and Decision Tree use thresholds, and Naive Bayes uses probability and statistical analysis, we see that there is a difference between how each feature provides value to a particular type of classifier.

TABLE IX
RESULTS OF TESTING EACH CLASSIFIER WITH AND WITHOUT THE UNIMPORTANT FEATURES INCLUDED

	DT	LR	GNB	KNN
score w/ all features	100%	99.04%	84.12%	99.30%
score w/ top 3 features	100%	99.08%	83.77%	99.90%

When fitting the classifiers to data containing only the top 3 features identified by permutation importance ranking, every classifier was able to maintain an accuracy within 1% of the accuracy produced by fitting to data with all 20 features. This shows that the permutation importance method was correct in identifying features that strongly relate to the target variable, and that these classifiers do not need the 17 less important features in identifying strong passwords.

Research Question 1: Determine the Best 3 Features: The permutation importance method of feature importance ranking was very effective in determining which features were most important for each classifier. We found consistent results when testing with different random seeds and different random sampling of the test/training data. We successfully used this method to find the important features to each classifier and discovered that classifiers generally agree on the single most important feature, but can differ on which features are in the middle range of importance. We found that the most important feature was Length of the password. This is what we would expect since security experts have been teaching for a long time that we need to have longer passwords to be more secure. The second most important feature was not agreed upon but depended on the method of classification. So we were not able to determine the second and third most important features for password classification in general.

1) Research Question 2: Is the Classifier Helped or Hindered by Including the Other Features?: There were 3 top features identified for each classifier. Running the classifier without the 17 less important features did not result in a significant increase or decrease of prediction accuracy. This means that using the permutation feature importance method, we successfully reduced the data by using only 15% of the features and predicting the strength of passwords with an equal amount of accuracy.

V. CONCLUSION AND FUTURE WORK

To address our research questions, we employed Decision Trees/Random Forest, Neural Network models, and Permutation Feature Importance methodology. Our main goal was to identify the three most important features (out of twenty) for neural network training. We also aimed to investigate whether adding the seventeen less important features would improve the prediction ability and to compare the performance of a single decision tree versus an ensemble of Random Forests, as well as the conventional advantages like reducing overfitting and improving accuracy.

According to the decision tree's feature importance rankings and the neural network's weight distribution, our findings verified that the three most important features were "Length," "Numbers," and "Uppers." Prediction accuracy remained at its peak when only the best features were used, and the addition of the seventeen extra features had no discernible impact. This demonstrates the negligible or redundant nature of the lesser features in our specific dataset.

We found that although both the Random Forest ensemble and the single decision tree achieved high accuracy, the ensemble approach did not significantly outperform the single tree. This may be explained by the single tree's already high accuracy, which leaves little opportunity for improvement.

The Random Forest did, however, provide information about the significance of features and the function of bagging in predictive modelling.

Our findings essentially confirm the significance of feature selection in machine learning. We can streamline models for improved performance and efficiency by identifying and using only the most influential features. This idea can be applied to a variety of data science endeavours. Further research endeavours may involve implementing these approaches on a wider range of datasets and machine learning techniques, such as unsupervised techniques like Support Vector Machines and Clustering, and regression models, in order to corroborate the feature importance analysis's universal fit.

REFERENCES

[1] Guyon, Isabelle, et al. "Gene Selection for Cancer Classification using Support Vector Machines." Machine Learning, vol. 46, pp. 389-422. Springer, 2002.

[2] Wojtas, Maksymilian A. and Ke Chen. "Feature Importance Ranking for Deep Learning." NeurIPS, https://proceedings.neurips.cc/paper/2020/hash/36ac8e558ac7690b6f44e2cb5ef93322-Abstract.html. Accessed 14 December 2023.

[3] Farghaly, Heba M. and Tarek Abd El-Hafeez, "A high-quality feature selection method based on frequent and correlated items for text classification" Soft Computing, vol. 27, pp.11259-11274. Springer, 2023.

[4] Fisher, Aaron, Cynthia Rudin, Francesca Dominici, "All Models are Wrong, but Many are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously" Journal of Machine Learning Research, vol 20, num 177, pp. 1-81.

APPENDIX

Fig. 3. Figure all-20-features.

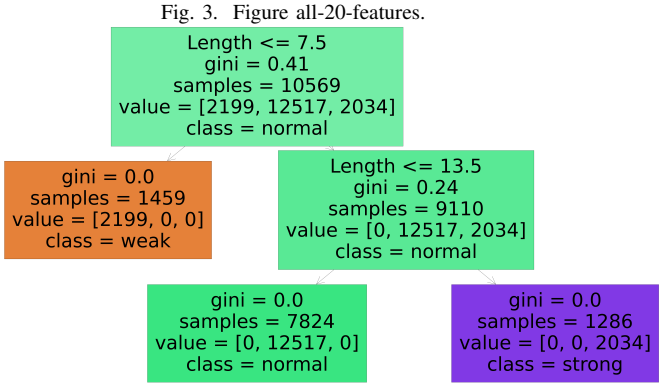


Fig. 4. Optimized Decision Tree

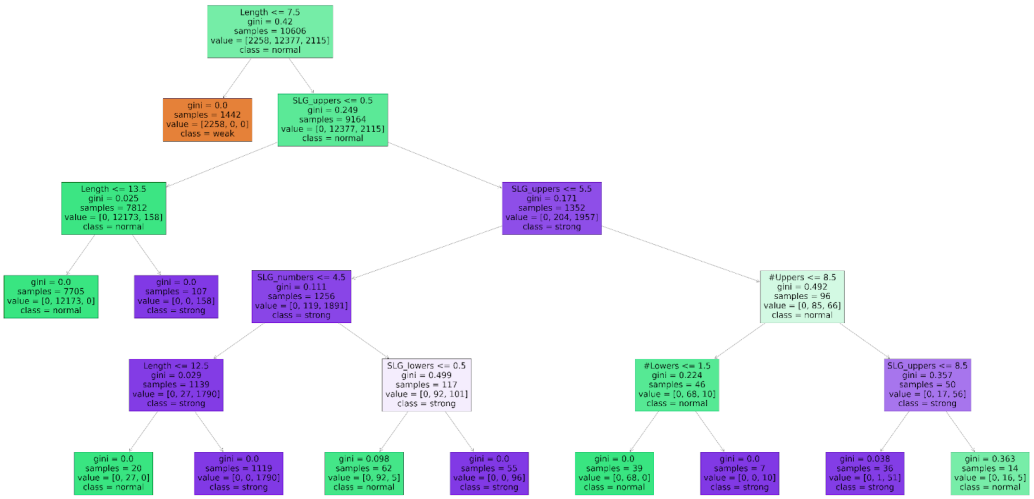


Fig. 5. Figure optimized-rf.

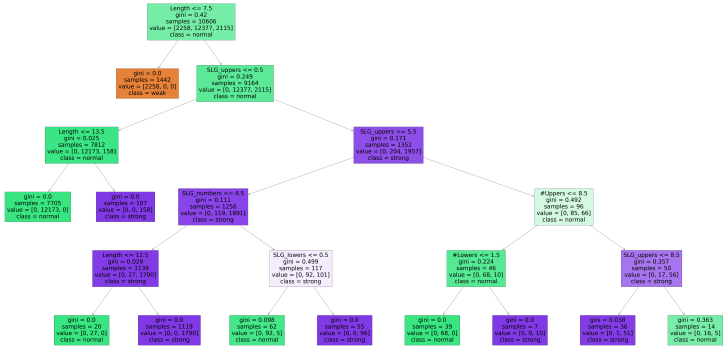


Fig. 6. Figure output-0.

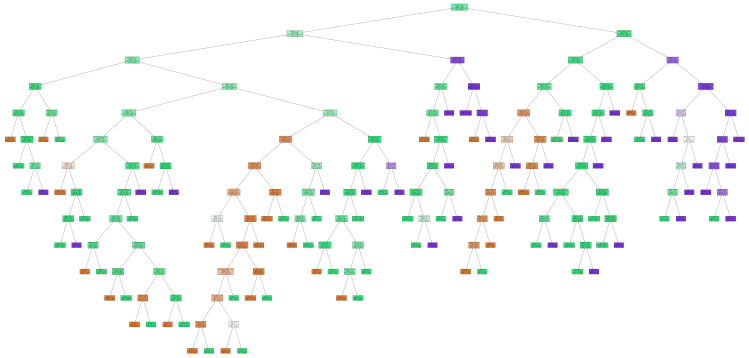


Fig. 7. Figure output-1.

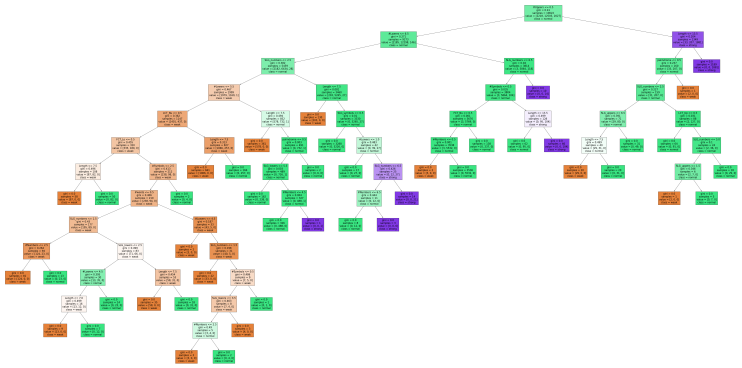


Fig. 8. Figure output-100.