Filtering the Family Group Chat: Using Machine Learning to Determine Importance

Sarah Putnam Ling spl2@williams.edu
Computer Science and Classics

1. Introduction

My phone is on me almost always and the majority of vibrations come from one single group chat: the family WhatsApp. It is the stream of my mom, dad, sister's and my thoughts and important communications. From links to funny dog videos on Instagram, updates on social lives, what needs to be bought at the grocery store, who needs to be picked up from the train, to the deaths of our friends and family members: our family chat has everything. I love my family chat; however, some texts are good to review when the day is done whereas others need to be seen in the moment. Due to the vast amount of texts, it is hard to see what is actually important. Ideally, a ML model would be able to read texts that come in and determine whether or not it is important enough for a notification. So, I trained a model on the data set of my family texts to determine whether or not a specific text is important: 0 or 1. There are a total of 6,803 texts ranging from January 26 to November 7, 2023 that I ranked.

For Apple devices, the current way of determining message importance is simply time. If 5 iMessages are received from the same sender in a short duration of time, the iPhone will offer to put that individual conversation on mute. This is a very naive solution that does not consider the content or the importance of each text. As for current methodologies of determining importance, Roy et al. (2020) uses Deep Learning to successfully detect spam in SMS messages. They categorized their data with common categories of English language such as nouns, adjectives, and verbs as well as more complex analyses such as the Flesh Reading Score (ranking ease to read), Dale Challe Score (ranking complexity of words), and Wrong words (ranking number of misspelled words) (Roy et al., 2020). After running their dataset on all variation of ML models, Roy et al. (2020) decided on using a 3 fold Convolutional Neural Network (CNN) with drop out and 10 fold cross validation which resulted in a .9944 accuracy score. Roy et al. (2020) made a 2D matrix for the words and values in each data point, which lends itself to the CNN since a CNN is an analogous form of Neural Networks which specializes in 2D image-based data (O'Shea and Nash, 2015). They claim that the hidden layers of a CNN model help extract context-dependent in the text that other models cannot do as successfully.

Though Roy et al. (2020)'s work relates to my project because we are both aiming to filter SMS messages, their work focuses on eliminating spam which often has notable characteristics such as links, poor language, and questionable demands. My project's problem is more nuanced since all texts are from the same four humans speaking in colloquial English

and instead of determining unimportance I am deciding importance, which is based on if a text is thought to be "significant" as opposed to a spam being determined as "sketchy".

Due to the nature of my goal to measure significance in a text, I processed my data and chose my parameters to all the words mentioned over 75 times. The value of each theta is the number of times a text uses the word. My base model is set to determine importance if it includes a "?" character, which accounted for a 0.614 accuracy. The highest accuracy I achieved was a decision tree of depth 3, which received an accuracy of 0.631. This lack of improvement I presume can be further fixed by further cleaning my data, choosing better values for thetas, and by minimizing bias perhaps through boosting.

2. Preliminaries

The two different models I used in this project are Logistic regression and Classification Trees:

Logistic regression uses a number of features which each have an assigned weight given by θ. The dot product of the weights θ with the value of each individual data point we call x gives us the final prediction we call Y (Peng et al., 2002).

$$Y = \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_d x_d.$$

In order to improve the weights θ to predict a more accurate Y value, we deploy a loss function that utilizes gradient descent. Gradient Descent uses the first-order derivative to determine the slope (or the speed of change) when a small step in place step is taken at the current weights. It is very similar to how a derivative of distance gives us the speed in a physics problem. In an ideal function, using the gradient descent of a Loss function will cause the function to hit a global minimum.

In this project, an L1 regularization is used to determine the loss. An example of L1 loss is the Mean Absolute Deviation (MAD), which measures loss by the sum of all the absolute value differences between the predicted values and the actual values.

$$MAD = \frac{1}{n} \sum_{i=1}^{n} |y_{predicted} - y_{actual}|$$

L2 regularization is another frequently used loss function that uses the squared difference as its benchmark. Though good for many projects, L2 regularization penalizes wrong answers too much for a project such like this. Since very similar-looking texts could easily be important or unimportant, it is better to go with the loss function that punishes error less: L1 regularization.

Together, with the gradient descent and the L1 regularization, the loss function is as follows:

$$L(\theta^{(i+1)}) \leq L(\theta^{(i)}) - \frac{\alpha}{2} ||\nabla L(\theta^{(i)})||^2$$

where α is the learning rate. This model is implemented using Sklearn's Logistic Regression Linear Model package with the specific "liblinear" solver which allows for L1 regularization. Liblinear works well for large classification problems given that it uses L1

regularization and a coordinate descent algorithm, which is similar to gradient descent except calculated in the context of coordinate directions. Logistic regression guarantees the same outcome each time since there is no randomization of variables used in determining the predicted Y values.

Classification Trees is a machine learning technique that returns a binary output based on a series of decisions as denoted by junctions in a tree, where you begin at the root and end at a leaf (Loh, 2011). Each decision is determined by the θ of most impact. One way to calculate the θ of most impact is through entropy, which is the way that I implement the classification trees. The entropy is the weighted average between a potential split decision, and the entropy with the number closest to 1 is the best split, which is given by this equation where p_n are all the possible splits:

$$Entropy(Data) = -\Sigma_c p_c log_2(p_c)$$

Classification Trees depend on their depth of tree for their fitness to the data. On a clear set of data, a decision stump (a tree with depth = 1) normally is a model that overgeneralizes or, in other words, has high bias. A tree of unlimited depth should overfit the data and result in a rigid model that doesn't generalize enough. This is a variance issue. The bias-variance trade-off is that either extreme does not make for a robust model, so finding an in-between depth is desirable so it avoids the faults of either.

These two different models have different strengths. The logistic regression is helpful as a baseline for any ML model, but also achieved a higher accuracy when the data was shuffled. The classification Tree is more consistent even when more parameters are added and sticks closely to specific words being used and makes decisions solely based on keywords.

3. Data

As for the data, WhatsApp has a export function where formats all the texts in a conversation into a .txt file. At that phase there were about 7,000 texts. Images and documents show up as empty texts, so I removed this empty data making the texts' numbers 6,803. I then converted it into a .csv which had columns date, time, sender, and text message. I added a Y column where I manually ranked the texts as important or not important.

What were the guidelines for myself in determining if a text was important? Realistically, I had no good methodology. I started out ranking the texts based on impulse: would I need to read this or not. As I was ranking the texts I realized that most the texts I rated as important (1) were questions, simply because if someone in the group chat was asking a question, a majority of times the question was important. Most of our family questions have to do with who is picking up whom from the train, which is pertinent and time-sensitive. However, not all questions are urgent. Another factor my impulse was based on was if the text had celebratory or interesting information in it, like when my sister got a full-time job after graduation. However, this information is not as pertinent as the questions; it just happens that it is breaking news.

While ranking the data, I quickly realized it would be impossible to keep the data completely independent and identically distributed (IID). Inherently when I ranked the texts, I was reading them in chronological order. I realized while I was deep in the ranking

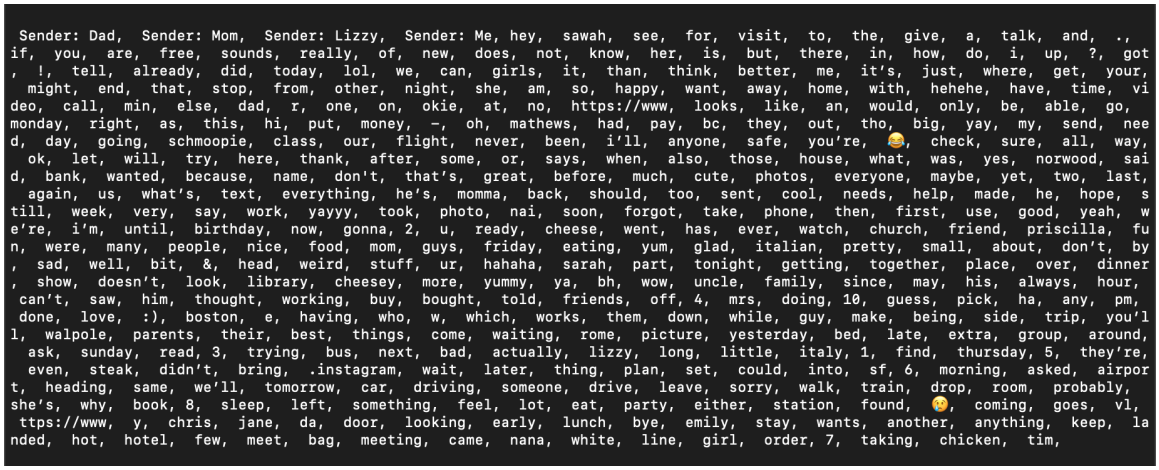

Figure 1: Words used over 18 times

process that there was no way for the machine to know if "ok" was an important text or not. When it came to the text "ok", I ranked it as important if it was an answer to a question of imminence, which arbitrarily is important and gets scored 1. But there were many other instances in the sentiment of agreement where "ok" was not deemed as important and got scored 0. This means that there are many data points in my data set that have the same input—text="ok"—where the y could be either 0 or 1. This dependence on the context shows me the faults of my data set. Also while I was ranking the data, I realized I was biased towards myself and tended to rank my own texts more frequently as important than others in my family.

Once I rated all the texts, I coded a counting method where I inputted a .txt file of the texts all together stripped from elements such as sender, date, and time. Originally, I took the words mentioned over 75 times. Then in my second rendition of the model, I took any words mentioned over 18 times and made these most frequently used words my parameters. I did not take out articles or seemingly unimportant words since I wanted to know what the ML model would take into consideration even if these words add no significance. Each of these words was a column in the .csv file. Each text had a corresponding row where the number in each location corresponded with the count of how many times that word was used in the text. I then stripped away the columns that denote the date and the time sent. As for sender name, I replaced with numbers.

A way to make the data more IID would be to group texts that are sent within the span of 70 seconds apart from each other. This would simplify the issue with texts like "ok". I also noticed that if there was an important conversation topic going on, I would rate all texts related to that conversation as important even if the individual text was not. By grouping together texts in similar time frames, the model would be able to learn more systematically what is considered "important".

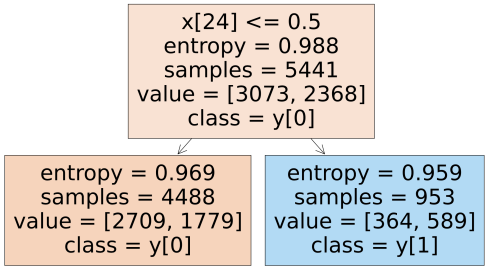

Figure 2: Accuracy of 0.58 decision stump. x[24] = "!"

4. Training And Validation Of Models

There is no base approximation established for what I am training my machine to do because of the personal nature of my project. Also, I found that it was more common to measure spam or unimportance of texts than importance. To create a baseline approximation for myself, I used the "?" as the standard. Logically speaking, if someone asks a question, that means they need an answer and hence that text is considered important, because you might need to answer the question. Considering only the "?" resulted in an accuracy of 0.613.

As for the logistic regression model, an intercept column of ones was added and L1 regularization was used. The intercept column of ones lets us have data points of all zeros without affecting entropy and weight calculations.

As for the classification trees used, I used a decision stump (a depth of 1), a tree of depth 3 for the model with words used more than 75 times, a tree of depth 4 for the model with words used more than 18 times, and a max depth tree. Each of the trees used an entropy calculation to determine best split.

5. Results

All numerical results can be found in the table below. The data was split into a .80 training and .20 testing. I made two different rounds of models: the first with words mentioned over 75 times, the second 18 times. When I switched models to include more word parameters, the Logistic Regression improved slightly and the decision trees remained mostly the same. In the discussion of results, when the Logistic Regression is discussed I will largely reference the second round model, whereas for the trees I will reference the first round models for their greater simplicity.

When using the words mentioned greater than 75 times (WM>75), Logistic Regression and the Max Depth Decision Tree did worse than the baseline "?", except the decision stump and the Decision Depth 3. Ironically the decision stump used "!" as the feature denoting the most importance. This is probably in a text of excitement. Colloquially, it is normal to include many exclamation points. For example: "I got a job!!!!!" has many exclamation points and is considered important.

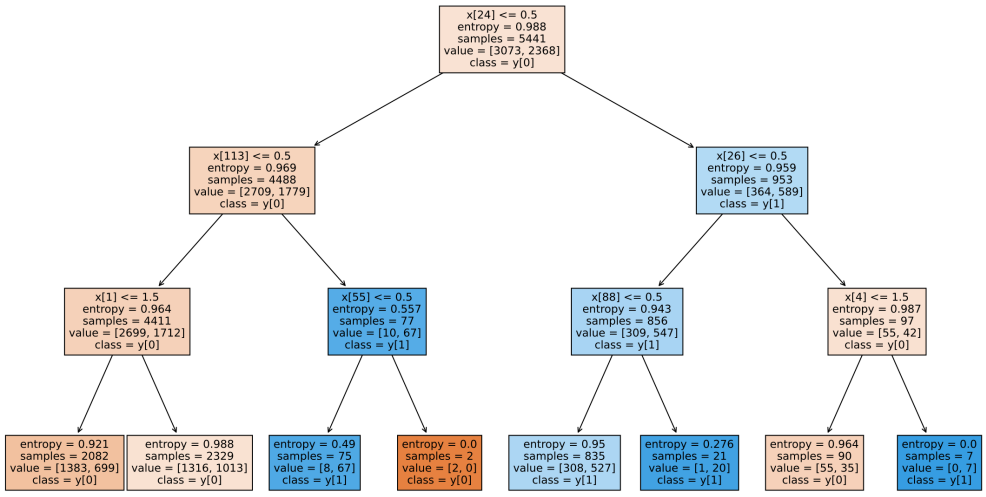

Figure 3: Accuracy of 0.631 tree of depth 3. x[24] = "!", x[113] = , x[26] = "?", x[1] = sender, x[55] = "no", x[88] = "us", x[4] = "to"

For the WM>75, Logistic regression did slightly better than the Baseline, but the Precision, Recall, and F1 are strikingly low for such a small increase in accuracy in comparison to the baseline. However, when more parameters were added in the WM>18 model, Logistic Regression improved to be better than the Baseline. The increase of parameters gives the Linear Regression more options to place weight distribution. The words used fewer times still can act as important keywords in the Logistic Regression model, whereas in a Decision tree of limited depth, smaller features will get completely ignored. Also for the WM>18 version, I decided to use L1 regularization instead of the L2 regularization that was used in the WM>75 edition. L1 simply works better for the data that I create since it punishes the errors less harshly. Now the Logistic Regression can be compared to the WM>75 Decision Tree Depth 3 to have similar strength if not better since the Regression's Recall and F1 score are higher as well. I will talk more about Logistic regression in the ablation study.

The Depth 3 did the best until it met its superior: the WM>18 Depth 4 Tree. The Depth 3 uses factors like "!", "train", "?", sender, "no", "us", and "to" which gained it a 0.631 accuracy. However, the precision-recall balance for the Depth 3 tree is not great. Since we are determining importance, it is better to have a higher precision, which is preferable to false positives. A false positive in this project is when the predicted y value is 1 (flagged as important) when the reality is that the text is a 0 (not important). In other words, it is better to have a text labeled as important even if it isn't (false positives) than the phone silence a text thought to be unimportant when it actually is (a false negative).

The WM>18 Depth 4 Tree did slightly better than the Depth 3 Tree in Accuracy, but also had a general improvement in its Precision, Recall, and F1 score. This makes it the

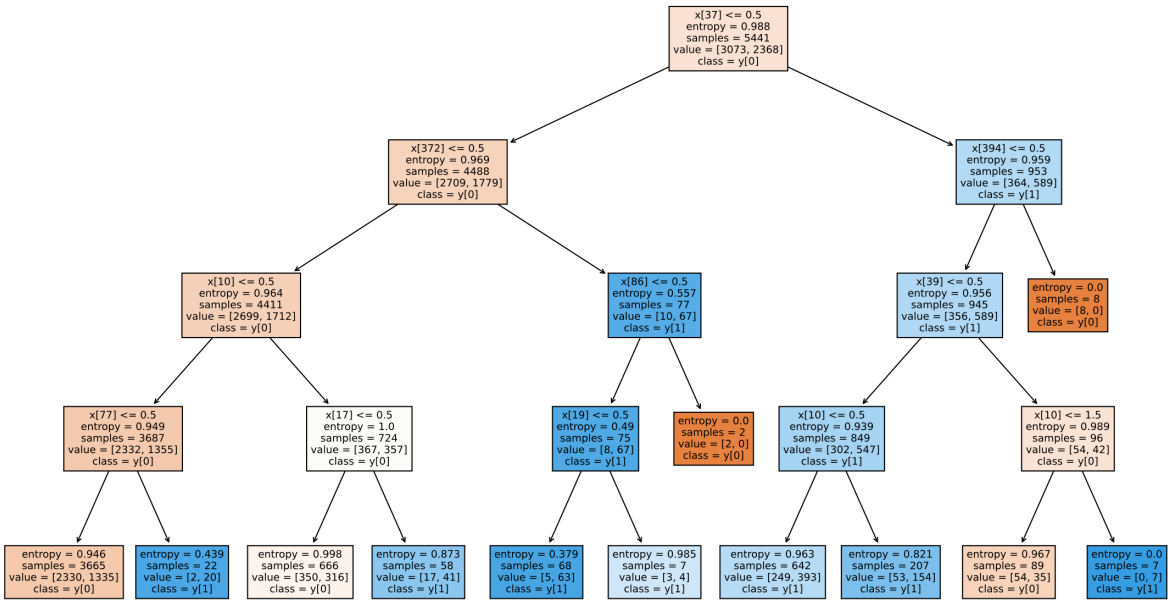

Figure 4: Accuracy of 0.639 tree of depth 4. x[37] = "?", x[372] = "train", x[77] = "call", x[17] "if", x[86] = "no", x[19] = "are", x[394] = "ttps://www", x[39] = "!", and the x[10] "to" in three different instances

best model in this set. What this means is that there was a feature in the expanded set of parameters that split entropy slightly better than the parameters in the Depth 3. This tree used "?", "train", "to", "call", "if", "no", "are", "ttps://www", "!", and the word "to" in three different instances. I will also mention now, that when originally cleaning out the data's texts that used the enter key, that new line needed to be removed in order to be compatible with the .csv environment. However, since I did this on a grand scale, the best solution I could find forced me to delete the first character of the word following the line break. This is why "ttps://www" and "https://www" are two separate parameters. "Call" and "ttps://www" are the only words that are not available to the Depth 3 set that are used.

I find it surprising that the Decision Max Depth Tree did not overfit the data, which is what decision trees are prone to do. This must mean that there is a lot of noise in the data and that the data is not consistent. Consistency and noise will be further discussed in the ablation study section of this paper.

Round	Model	Accuracy	Precision	Recall	F1
	Baseline: "?"	0.613	0.596	0.613	0.572
WM>75	Logistic Regression	0.614	0.463	0.328	0.384
WM>18	Logistic Regression	0.631	0.497	0.386	0.435
WM>75	Decision Stump	0.623	0.473	0.224	0.304
WM>18	Decision Stump	0.623	0.473	0.224	0.304
WM>75	Decision Depth 3	0.631	0.496	0.228	0.312
WM>18	Decision Depth 4	0.639	0.518	0.260	0.346
WM>75	Decision Max Depth	0.58	0.418	0.362	0.388
WM>18	Decision Max Depth	0.586	0.437	0.436	0.436

6. Ablation Study

I did two variations of an Ablation Study. I compared the Logistic Models with and without the "ok" column, and secondly, I shuffled the data.

First I discarded the "ok" column. And the only difference was that the Logistic regression accuracy for WM>75 became 0.61 and the WM>18 became 0.627. The Trees did not change much since if "ok" was not a factor used in prior trees, then deleting it would not make a difference. This does not prove anything about noise since the "ok" column turns out to not have weighed too heavily in the end result. If anything, the "ok" column did provide some accuracy even if it is partially misleading.

The second variation I did was shuffling my data. Ideally, I did not want to shuffle my data since the model should be able to "generalize" onto different months, which of course have different topics of discussion and ways of life. So shuffling the data actually makes it a worse model if I were to extend it beyond this dataset. However, the result of the shuffled data was as follows.

Round	Model	Accuracy	Precision	Recall	F1
	Baseline: ?	0.589	0.61	0.589	0.541
WM>75	Logistic Regression	0.628	0.624	0.358	0.455
WM>18	Logistic Regression	0.652 - 0.675	0.439	0.638	0.52
WM>75	Decision Stump	0.599	0.594	0.236	0.337
WM>75	Decision Depth 3	0.611	0.62	0.268	0.374
WM>75	Decision Depth 11	0.644	0.447	0.562	0.372
WM>75	Decision Max Depth	0.589	0.469	0.532	0.419

With the shuffling of the data, logistic regression vastly improved and the most accurate tree became the Decision Depth 11 Tree.

The reason why Depth 11 is better must be similar to the idea of finding the sweet spot between too basic versus too in the weeds. Since the shuffled data takes away some element

of the unknown in the test set, the Depth 11 tree's splits can be more accurate to what it will be tested on.

Logistic Regression's improvement makes sense since the shuffled data samples help the parameters be more congruent with the test data that has yet to be seen. However, this makes the integrity of the data set questionable since these values portray a false generalizability in the model. I would also like to note, that in the WM>18 Regression data, I added a range of accuracies that on a given shuffle would perform within a .02 percent difference. This also proves that the model I have made is rudimentary, since the cut of data affects the model so greatly. A good model would remain more consistent throughout shuffles; however, some variation is bound to happen when a random variable is introduced.

I am including this as an ablation study and not as my real results because I do not believe that these models should be used since the shuffled nature takes away from the integrity of the model.

7. Discussion and Conclusion

Clearly, the accuracies are not high enough to consider implementing a real-time notification silencer on my phone. I started out with the question of whether a basic ML program could determine which texts in my family group chat are important, because it is not easy for a human like me to determine these things instantaneously either. I then cleaned and sorted the data, and chose to use logistic regression and decision trees given that the parameters I was feeding the model were specific words. Intuitively, I thought that if word x is in the text, then either yes or no, this text is important. Through testing I found that a Classification Tree of depth 4 on words mentioned more than 18 times is indeed better than logistic regression and determination by use of question marks; however, not by much. I then briefly looked into how the accuracy of each model changes when the data is shuffled which improves the accuracy of logistic regression and some decision trees while the accuracy decreases for the question mark set and the other decision trees. As for areas of improvement, categorizing group texts by time frame instead of individual texts, trying a different ML model, and correcting spelling mistakes could all decrease noise and bias in my current models.

Overall, ML models should do better than a query of "if question marks are included" since they are given more information. They did, but not as well as would be expected for the amount of (albeit noisy) information given to them. Throughout the process of this project, I have actually become proud of the low accuracy scores. I originally decided on this topic because I knew how random, wild, and surprising my family group chat is. The low score is proof of both my beginner level of machine learning skills but also at the unpredictability and humanity of my family's life experience through text.

References

Wei-Yin Loh. Classification and regression trees. *WIREs Data Mining and Knowledge Discovery*, 1(1):14–23, 2011. doi: https://doi.org/10.1002/widm.8. URL `https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.8`.

Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks, 2015.

Chao-Ying Joanne Peng, Kuk Lida Lee, and Gary M Ingersoll. An introduction to logistic regression analysis and reporting. *The journal of educational research*, 96(1):3–14, 2002.

Pradeep Kumar Roy, Jyoti Prakash Singh, and Snehasish Banerjee. Deep learning to filter sms spam. *Future Generation Computer Systems*, 102:524–533, 2020.