Project 1: Scam detection with naïve Bayes

Kevin Yu
Student ID: 1462539
The University of Melbourne

April 16, 2025

1 Supervised model training

The prior probability of **non-malicious class was 0.7995** (rounded to 4 d.p.), while the **scam class was 0.2005**. This indicates that the dataset is imbalanced, with a higher proportion of non-malicious instances. The distribution aligns well with real-world data, where non-malicious messages significantly outnumber spam.

Table 1 shows the top 10 most "probable" words for each class, based on the conditional probability $P(w \mid c)$. In contract, Table 2 shows the top 10 most "predictive" words for each class, calculated by the ratio, $P(w \mid c)/P(w \mid \neg c)$. Notably, the words in each table differ; although both measure the association of words with a particular class, they serve a slightly different purpose.

For example, most probable words include words such as `call` ($p = 0.0274$), `customer` ($p = 0.0092$) and `reply` ($p = 0.0080$). These words appear frequently in scam messages, which makes them probable; however, intuitively, we expect legitimate messages to frequently contain such words, especially those from customer service. In contrast, the most predictive words include terms such as `prize` ($p = 88.80$) and `claim` ($p = 41.21$). These words can appear in legitimate context, however, they are far more likely to be used in scam messages (*how often do you randomly win a prize?*).

For non-malicious messages, the most predictive words include slang such as `lor` ($p = 32.16$), `wat` ($p = 19.53$) and `lol` ($p = 17.80$). These casual words are more likely to be used in non-malicious messages, and are not commonly found in scam messages. This serves as a good example of how the model can learn to distinguish between the two classes based on word usage.

Overall, **the result is reasonable** and it seems feasible to distinguish between scam and non-malicious messages using a multinomial naïve Bayes model. The differences in both frequent and predictive vocabulary across classes suggest the model can effectively learn class-specific language patterns, despite the simplicity of its assumptions.

Scam Class		**Non-malicious Class**	
Word	**Probability**	**Word**	**Probability**
call	0.0274	go	0.0161
free	0.0137	get	0.0143
claim	0.0100	gt	0.0085
customer	0.0092	lt	0.0084
txt	0.0090	call	0.0083
ur	0.0085	ok	0.0078
text	0.0082	come	0.0075
stop	0.0082	ur	0.0075
reply	0.0080	know	0.0075
mobile	0.0078	love	0.0071

Table 1: Top 10 most probable words for each class

Scam Class		Non-malicious Class	
Word	$P(w \mid \textbf{scam})/P(w \mid \textbf{ham})$	**Word**	$P(w \mid \textbf{ham})/P(w \mid \textbf{scam})$
prize	88.80	gt	60.30
tone	57.46	lt	59.73
select	41.79	lor	32.16
claim	41.21	hope	27.57
50	34.82	ok	27.57
paytm	33.08	da	22.40
code	31.34	let	20.10
award	28.73	wat	19.53
won	27.86	oh	18.38
18	26.12	lol	17.80

Table 2: Top 10 most predictive words for each classe based on likelihood ratios

2 Supervised model evaluation

The classifier achieved an **accuracy of 0.9700**, with **precision = 0.9381**, **recall = 0.9100** and an **F1 score of 0.9239** (scam as positive class). The model's performance is quite good, indicating that it is effective in distinguishing between scam and non-malicious messages.

The confusion matrix in Figure 1 highlights a high count of true positives and true negatives, supporting the model's effectiveness.

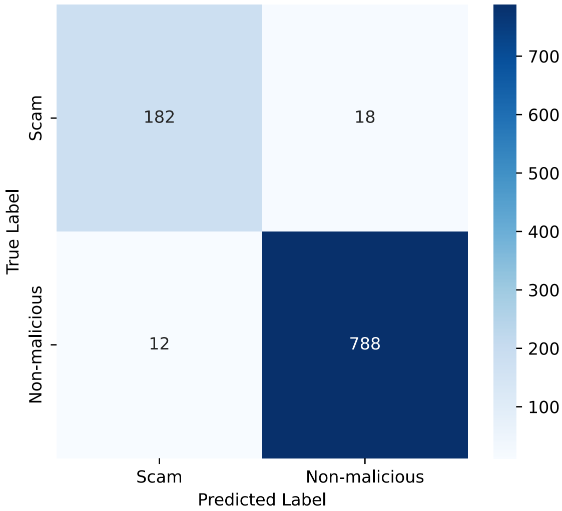


Figure 1: Confusion Matrix for Supervised Model.

Our model encountered **179 out-of-vocabulary words**, accounting for only 1.6320% of the total. These are likely rare or gibberish words, suggesting the vocabulary covers the dataset well.

Table 3 displays examples of high-confidence scam predictions with a measure of confidence, which is computed as the ratio of the posterior probabilities for each class. It often reference fake rewards from telecom companies and include words like `call`, `award` and `claim`—strong indicators per Table 2. Due to the naïve Bayes' multiplicative nature, confidence ratios can be very large, reaching up to 10^{17}.

Likewise, for high confidence non-malicious predictions, the messages are often casual, with words like `ok`, `good` and `wat`. It also includes systemetic transaction messages. While the use of formal language might intuitively align with the scam class, it is important to note that legitimate transactions are typically redacted. For example, the first text in High Confidence Non-Malicious Predictions has reference number, money and time redacted—unlike the scam messages. Additionally, such transations' raw text uses HTML entity encoding such as `&lt;`(<) and `&gt;`(>) to to enclose redacted content. As shown in Table 2, these tokens are highly predictive of the non-malicious class.

For boundary cases, it often contain neutral vocabulary that does not strongly indicate either class, or mix of words from both classes.

Overall, the **model's predictions are reasonable** and align with the expected patterns of scam and non-malicious messages.

High Confidence Scam Predictions	
Text	**Confidence Ratio**
Todays Vodafone numbers ending 5347 are selected to receive a Rs.2,00,000 award. If you have a match please call 6299257179 quoting claim code 2041 standard rates apply	1.87×10^{17}
Todays Vodafone numbers ending 3156 are selected to receive a Rs.2,00,000 award. If you have a match please call 7908807538 quoting claim code 9823 standard rates apply	1.40×10^{17}
Todays Voda numbers ending 5226 are selected to receive a ?350 award. If you have a match please call 08712300220 quoting claim code 1131 standard rates apply	1.07×10^{17}

Table 3: High Confidence Scam Predictions

High Confidence Non-Malicious Predictions	
Text	**Confidence Ratio**
NEFT Transaction with reference number <#> for Rs. <DECIMAL> has been credited to the beneficiary account on <#> at <TIME>: <#>	1.18×10^{-17}
no, i *didn't* mean to post it. I wrote it, and like so many other times i've written stuff to you, i let it sit there. It WAS what I was feeling at the time. I was angry...	2.34×10^{-16}
U wake up already? Wat u doing? U picking us up later rite? I'm taking sq825, reaching ard 7 smth 8 like dat. U can check e arrival time. C ya soon...	7.95×10^{-14}

Table 4: High Confidence Non-Malicious Predictions

Boundary Cases (Confidence Ratio $\approx$ 1)	
Text	**Confidence Ratio**
I've told him that I've returned it. That should I re-order it.	1.00
Glad to see your reply.	1.08
ALRITE SAM ITS NIC JUST CHECKIN THAT THIS IS UR NUMBER-SO IS IT?T.B*	0.90

Table 5: Boundary Cases (Confidence Ratio $\approx$ 1)

3 Extending the model: Active Learning

For this section, I have chose to implement **Option 2: Active Learning**, where 200 instances of the "unlabelled" data was carefully incorporated to improve the model.

The process began by passing through `sms_unlabelled.csv` to the model trained on the original `sms_supervised_train.csv`. For each message, the model generated a predicted class label and a corresponding confidence score. The method used to compute these scores is outlined in Section 2 and explained in greater detail in the Jupyter Notebook. Next, the original supervised data set was split into a training set (80%) and a validation set (20%). I then explored **three selection strategies** for choosing 200 unlabelled instances:

- **Random selection**: serves as a naive baseline, assuming all instances are equally informative.
- **Low-confidence selection**: selects instances near the decision boundary ($R \approx 1$).
- **High-confidence selection**: selects instances where the model is very certain (very high or very low R).

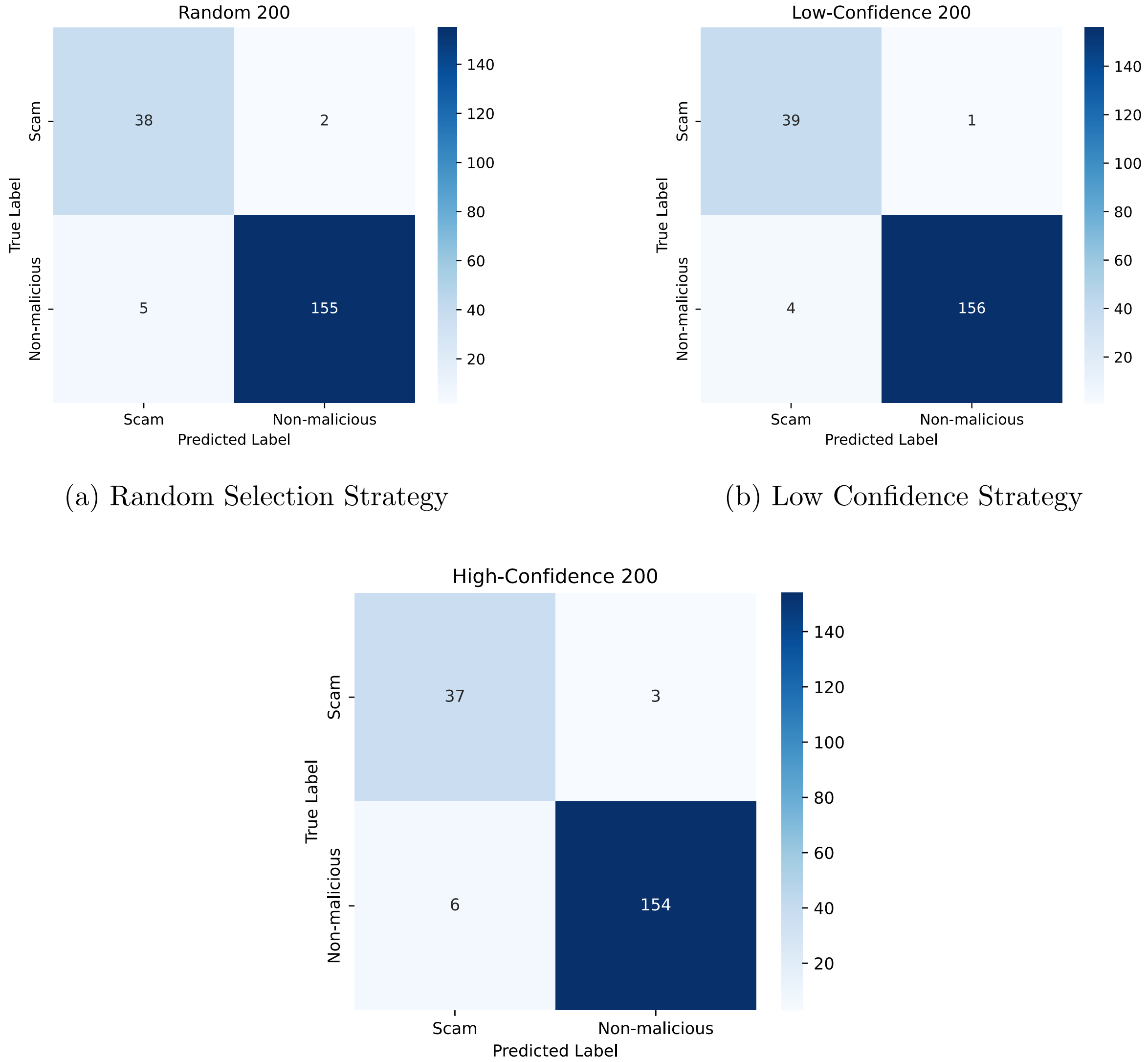
The lowest-confidence strategy was chosen to explore the core intuition behind active learning—when model is uncertain, it adjust to gain clarification. These ambiguous instances likely lie in regions of feature space that are underrepresented or misrepresented in the original training data, where previously misclassified messages are most likely to occur. Therefore, by including these samples, the model can refine its understanding of these boderline cases, increase its confidence in prediction; and, ideally, assign them to the correct class. Labelling such samples can help the model to adjust its boundaries more effectively and reduce generalisation error.

Conversely, the highest-confidence strategy was included to test a hypothesis: reinforcing highly confident prediction might shapen the model's learned representation. Whilst by amplifying the dominant pattern, it might result in **overfitting**, it was included to explore whether such reinforcement could still yield performance gains—and to provide a meaningful contrast to the low-confidence strategy.

The 200 instances was appended to the split training data set, and the evaluation result are presented in Table 6. The confusion matrices for each sampling strategy is shown in Figure 2.

Model	Accuracy	Precision	Recall	F1 Score	Error Rate
Random 200	0.9650	0.8837	0.9500	0.9157	0.0350
Low-Confidence 200	0.9750	0.9070	0.9750	0.9398	0.0250
High-Confidence 200	0.9550	0.8605	0.9250	0.8916	0.0450

Table 6: Evaluation metrics for different semi-supervised augmentation strategies.



(a) Random Selection Strategy

(b) Low Confidence Strategy

(c) High Confidence Strategy

Figure 2: Confusion Matrices for Each Sampling Strategy

The result aligned with our expectations: the low-confidence strategy outperformed both random and high-confidence selection, achieving an accuracy of 0.9750 and a F1 score of 0.9398. This suggests that the model effectively learned from the ambiguous instances, generalised better to unseen data, improving its performance. The confusion matrix in Figure 2b further supports this, showing the lowest number of classifications (5 in total), alluding that the model substentially benefited from incorporating uncertain instances. Notably, false negatives dropped to just 1.

The high-confidence strategy, however, performed worse than random selection. Unfortunetly, this indicates that reinforcing highly confident predictions may not be beneficial and could lead to overfitting. It did not generalise well to the validation set, as the model may have fitted too closely to the training data, acccumulating noise, resulting in poor performance.

Based on these findings, the low-confidence strategy not only offers the best quantitative performance, but also aligns with the theoretical motivations behind active learning. I therefore proceeded with this strategy to build the final model.

4 Semi-supervised model evaluation

The performance of the semi-supervised model based on the **low-confidence strategy** shows a clear improvement over the base (fully supervised) model when evaluated on the test set. Both models were trained on the same total number of instances; the semi-supervised model replaced the 200-instance validation set with 200 low-confidence labelled examples.

As shown in Table 7, all evaluation metrics improved, with precision showing the most significant gain—from **0.9381** to **0.9534**—indicating that the model is now better at correctly identifying scam messages and reducing false positives. This is particularly meaningful in the context of scam detection, where mistakenly flagging legitimate messages can lead to unnecessary alarm or inconvenience for users (*no one checks their spam folder unless they are sure there is something there!*). The **error rate** also dropped from **0.0300** to **0.0250**, which corresponds to an error rate reduction of **17%** ($(0.03-0.025)/0.03 = 0.17$). Whilst these improvement may seem marginal, they are actually quite significant, considering the base (supervised) model was already performing at a high level. Such gains highlight the value of carefully selected semi-supervised data.

The confusion matrix in Figure 3 further highlights the improvement, showing a reduction in both false negatives and false positives compared to the base model in Figure 1.

Metric	**Base**	**Low-Conf. 200**
Accuracy	0.9700	0.9750
Precision	0.9381	0.9534
Recall	0.9100	0.9200
F1 Score	0.9239	0.9364
Error Rate	0.0300	0.0250

Table 7: Base vs. Low-Confidence 200.

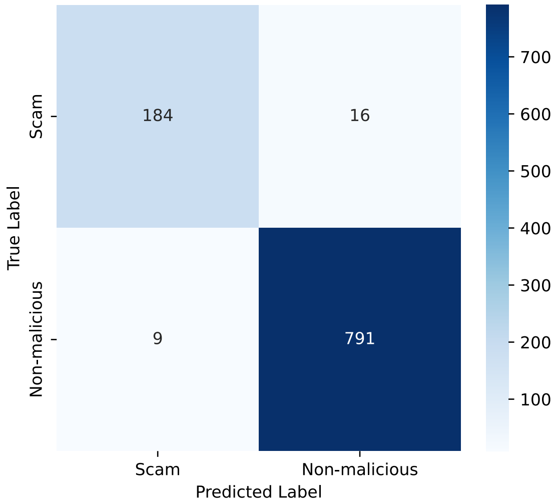


Figure 3: Confusion Matrix for Low-Conf. 200 on Test Set.

The model's representation has changed after the semi-supervised training, specifically, changes in which words the model considers most predictive of each class. This is illustrated in Table 8. Although most of the top 10 predictive words for both classes remain unchanged from Figure 2, their overall "predictiveness" has decreased. For example the top predictive word for scam class, `prize`, has dropped from 88.80 to 77.15. Similarly, for the non-malicious class, `gt`, has dropped from 60.30 to 56.00. The remaining top words follow a similar trend, with most experiencing a decline in their predictive power.

This aligns with our intuition, and it is a very natural and meaningful outcome. By incorporating ambiguous instances into training, the model has rebalanced to better reflect the true distribution of the data. It is now less reliant on a few highly predictive words, which may have been overfitting to the training data. Instead, the model has learnt to be more cautious, and to consider a wider range of words when making predictions. In other words, terms that has been highly predictive of the class have now been adjusted to be more balanced, while terms from the previously unrepresented "gray area", and therefore unpredictive, have became more influential. This is a positive outcome, as it indicates that the model is now more robust and avoid being overconfident based on a handful of domiant terms.

Scam Class		**Non-malicious**	
Word	$P(w \mid \textbf{scam})/P(w \mid \textbf{ham})$	**Word**	$P(w \mid \textbf{ham})/P(w \mid \textbf{scam})$
prize	77.15	gt	56.00
tone	54.86	lt	56.00
claim	54.00	ok	51.91
select	34.29	lor	29.75
paytm	32.57	da	19.83
cs	30.86	let	17.50
18	25.72	wat	17.50
code	25.72	lol	16.33
won	24.00	something	16.33
award	22.29	sorry	16.04

Table 8: Top 10 most predictive words for each class based on word likelihood ratios (Low-Confidence 200 strategy).

We can possibly further improve the model by incorporating more low-confidence instances, as these examples are likely to be more informative and in this case only accounted for 20% of the total training data. We can perhaps iteratively add low-confidence instances to the training set, retrain the model, add more low-confidence instances, and repeat the process until we reach a desired performance level. This approach supports our goal of active learning, where the model continuously learns from its own predictions and refines its understanding of the data.