A black text on a white background

Description automatically generated

**PROJECT REPORT**

**Project Title: Spam or Ham**

**Course: INFSYS-6862**

**Professor: Necdet Gurkan**

**11/10/2024**

**Team 3:**

Tim Burke

Jeremiah Fowler

Navya Rani Kanaka

Jin Bai

Contents

[Executive Summary 3](#_heading=h.gjdgxs)

[Introduction 3](#_heading=h.30j0zll)

[CRISP-DM Framework 5](#_heading=h.1fob9te)

[Phase 1: Business Understanding 5](#_heading=h.3znysh7)

[Phase 2: Data Understanding 6](#_heading=h.2et92p0)

[Phase 3: Data Preparation 7](#_heading=h.tyjcwt)

[Phase 4: Modeling 8](#_heading=h.3dy6vkm)

[TF-IDF 8](#_heading=h.1t3h5sf)

[Word2Vec 10](#_heading=h.4d34og8)

[BERT 12](#_heading=h.2s8eyo1)

[Phase 5: Evaluation 18](#_heading=h.17dp8vu)

[Phase 6: Implementation 18](#_heading=h.3rdcrjn)

[Conclusion 18](#_heading=h.26in1rg)

[References 19](#_heading=h.lnxbz9)

[Appendix (Optional) 19](#_heading=h.35nkun2)

# 

# Executive Summary

With the saturation of mobile use throughout the world, this has led to an increase in fraudulent SMS messages, often in the form of spam and phishing attacks, which can disrupt the user experience and pose a security risk. This project aims to enhance the customer experience by developing a robust machine learning model capable of accurately and precisely determining which SMS messages are fraudulent. We are leveraging technology such as Natural Language Processing (NLP) and Large Language Models (LLM) that can be adapted and implemented either on backbone nodes or directly on end-user devices. We will position this feature as a unique differentiator.

The scope of this project is focused on SMS text messaging. There is potential to expand upon this product with feature-rich text services and application for audio-based spam detection models. The primary goal is to significantly reduce the volume of unwanted communication. We are positioning this service as a unique offering to differentiate ourselves from our competitors. We believe this service will improve customer satisfaction, lower IT support costs, and enhance service reliability and security.

To accomplish our objective, we will need to acquire and preprocess collections of SMS text data. Exploratory data analysis (EDA) will be conducted to determine the patterns and characteristics of spam and non-spam messages, as well as determine what preprocessing needs to be done. This may include addressing missing values, removing punctuation, standardizing words, lemmatization, and generating key contextual features that can be used to further improve our model’s performance. Our approach will incorporate three different NLP/LLM models: Term Frequency – Inverse Document Frequency (TF-IDF), Word2Vec, and BERT embeddings. TF-IDF quantifies term importance in the dataset, Word2Vec captures semantic relationships between words, and BERT provides contextualized embeddings that represent the meaning of words within sentences. We will evaluate the performance of different models using classification metrics such as accuracy, precision, recall, and F1-score, with the goal of identifying the model that most effectively identifies spam messages while minimizing false classifications. We want to balance detection accuracy with minimal false positives for the best user experience and in doing so, we are positioning our product as a differentiator from the competition.

# Introduction

The rapid growth of digital communication in the 21st century has led to a significant increase in unwanted messages. “Spam frustrates users and exposes them to potential scams such as phishing attacks. This project aims to create and deploy a machine learning model for efficient classification of incoming text messages as “spam” or “ham”. The rise in phishing and scam messages places a significant burden on customers, both financially and emotionally, while also wasting their valuable time. In turn, drives up expenses for insurance and financial services that manage these risks. By developing our “Spam or Ham” detection mode, we can reduce fraud, protect customers, and reinforce our commitment to user safety while ensuring peace of mind when engaging with our services.

Automating the detection of spam messages helps reduce potential security threats and improves the overall quality and safety of our digital communication ecosphere. Implementing a robust model like ours will support both individual users and organization in their ability to filter and determine the legitimacy of SMS messages. This project follows the Cross-Industry Standard Process for Data Mining (CRISP-DM) framework with emphasis on the Data Preparation and Modeling phases.

The data preparation phase involves meticulously processing collections of text messages, addressing issues such as missing values, inconsistent formatting, special characters, and other factors that will impact the validity of our dataset. In addition to our preprocessing, we will also employ various text vectorization and embedding techniques to transform text into numerical representation that’s suitable for machine learning algorithms. The techniques we will be using in the scope of this project include Term Frequency-Inverse Document Frequency (TF-IDF), Word2Vec, and BERT models, each offering unique advantages for text analysis and classification. By using comparative methods between our three models, we aim to identify an ensemble architecture that provides the highest accuracy and reliability in detecting spam messages.

The successful implementation of this project will not only enhance customer experience by reducing spam and phishing messages but also provide a competitive edge for our organization. By offering a secure and trustworthy messaging experience, we can boost customer satisfaction, loyalty, and brand reputation. Ultimately, this project reinforces our commitment to user safety and positions our organization as a leader in providing reliable and secure communication services.

# CRISP-DM Framework

## Phase 1: Business Understanding

The goal of this project is to develop a machine learning model that can accurately and precisely categorize incoming text messages as either spam or ham. This solution to unwanted communication reduces the burden placed on individuals to discern the legitimacy of text messages and the security responsibility. Our robust model will be able to support both individual end-users and organizations in their specific use-cases for spam filtering.

Spam messages have a significant impact on businesses and individuals alike. Financial losses can occur when users fall victim to phishing scams or fraudulent messages, leading to compromised accounts or unauthorized transactions. Moreover, the time wasted on identifying and deleting spam messages reduces productivity and efficiency. Spam messages also pose potential security risks, as they may contain malicious links or attachments that can harm users' devices or steal sensitive information. This project aims to address the requirements of various industries, such as telecommunications, e-commerce, and financial services, where spam filtering is crucial for protecting customers and maintaining trust. Compliance with regulations like the CAN-SPAM Act in the United States and the General Data Protection Regulation (GDPR) in the European Union is essential to avoid legal penalties and reputational damage.

To meet our goal, we will use metrics and Key Performance Indicators (KPIs) to measure the performance of our model. We will focus heavily on precision and specificity to enhance user experience with minimal disruption to legitimate messages. Only valuable and legitimate messages will reach our customer. The KPIs that will measure our success includes:

1. High Precision (> 95%): High precision minimizes false positives, meaning legitimate messages are rarely flagged as spam. This preserves the flow of essential communication for our customers. The emphasis on minimizing false positives is crucial for maintaining customer trust and satisfaction.
2. High Specificity (>97%): Achieving high specificity allows our model to consistently identify ham messages as legitimate. This KPI helps ensure minimal interference with normal user communication, reinforcing a seamless user experience and aligning with our goal to offer a more reliable and non-disruptive service.
3. F1 Score (>90%): The F1 score balances precision and recall, providing a comprehensive measure of the model's effectiveness in capturing spam while minimizing errors. A high F1 score demonstrates the model's reliability in practical scenarios, supporting customer satisfaction and business reliability.

Additional KPIs such as ROC-AUC, Confusion Matrix, Recall, and Accuracy will help us further gauge the model’s performance. ROC-AUC scores will provide visualization of the model's true-positive rates, and the Confusion Matrix will provide insight into the specific performances of each classification. We will validate these KPIs by using real-world spam messages as a test to verify the model meets customer needs.

The dataset used for this project contains a total of 5,572 SMS messages, with 747 labeled as spam and 4,825 labeled as ham. The average message length is 80 characters, with a minimum of 2 characters and a maximum of 910 characters. Preprocessing steps include cleaning the text data by removing punctuation, converting it to lowercase, and tokenizing the messages.

## Phase 2: Data Understanding

The dataset we will be using to train our model contains real world data of SMS messages and their labels of either spam or ham. We outsourced real-world examples from Kaggle. We selected this dataset to train out NLP/LLM classification models because it provides real and diverse examples of spam. This dataset will train our models to accurately and precisely classify messages as spam or ham. The bias ratio of our spam-to-ham is an 87:13 split which corresponds with an 86:14 split verified by a recently published paper on "Securing Mobile Message Communications." We do not have suspicion that our model will favor either spam or ham. The model may perform less well when applied to messages outside of the context of the US English language since the messages in the dataset are pulled from North America.

Pros:

1. Pre-labeled: Ideal for supervised learning, labels are already assigned, enabling faster model training.
2. Balanced Lengths: SMS messages are brief, which aligns with NLP and LLM models using text classification algorithms.
3. Relevance to the real world: Aims to address spam filtering challenges, applicable to commercial SMS filtering systems.

Cons:

1. Outdated Data: The model's generalizability may be limited because the nature of spam messages may have changed since the dataset was collected (e.g., new sorts of scams).

The dataset contains a total of 5,572 SMS messages, with 747 (13.4%) labeled as spam and 4,825 (86.6%) labeled as ham. This distribution aligns with the expected ratio of spam to ham messages in real-world scenarios, providing an adequate representation for training and evaluating the model.

An initial EDA was performed on the dataset, revealing the following insights:

1. Message length distribution: The average message length is 71 characters, with a minimum of 2 characters and a maximum of 910 characters. Spam messages tend to be slightly longer than ham messages on average.
2. Common words and phrases: The most common words in spam messages include "free," "call," "prize," and "claim," while common words in ham messages include "ok," "thanks," and "see."
3. Class distribution: The dataset is slightly imbalanced, with 13.4% of messages labeled as spam and 86.6% labeled as ham.

The dataset will be split into training, validation, and test sets using a 70/15/15 ratio. This split allows for sufficient data to train the model (70%), tune hyperparameters and monitor overfitting (15% validation), and assess the model's performance on unseen data (15% test). The stratified sampling technique will be used to ensure that the class distribution is maintained across the splits.

## Phase 3: Data Preparation

To clean and prepare our raw dataset for ingestion, we use various techniques to standardize and prepare the text for analysis. The preprocessing steps include:

1. Lowercasing: Standardizing text by converting all characters to lowercase for uniformity.
2. Punctuation and Stop-word Removal: Removing non-essential words (e.g., "a," "the," "and") and punctuation marks that do not contribute to distinguishing spam from ham.
3. Tokenization: Breaking down text into individual words, allowing word-based analysis.
4. Stemming and Lemmatization: Reducing words to their root forms (e.g., "running" to "run"), improving word consistency across texts.

After our dataset has been processed, we perform feature extraction, which involves pulling specific information from raw text to create numerical representations. In the context of this project, we turn text in the English languages into numerical representation and vectors. Our model is only able to identify patterns and associations that distinguish neutral words (ham texts) from spam words through the ingestion of these numerical features and vectors. The feature extraction techniques we used includes:

1. Bag of Words: Counting word occurrences, helping the model focus on commonly repeated words typical of spam.
2. TF-IDF (Term Frequency-Inverse Document Frequency): Weighting words based on their rarity across all texts, emphasizing unusual words that may signal spam.
3. N-grams (Bigrams, Trigrams): Capturing key word pairs or triplets, such as "Click here," indicative of spam.
4. Word Embedding: Transforming words into vectors that represent semantic relationships. We may leverage pre-trained embeddings, if available, to save on training time.

The extracted features will be used as input for our machine learning algorithms during the model training process. The numerical representations and vectors obtained from feature extraction will be fed into the algorithms, allowing them to learn patterns and associations that distinguish spam from ham messages.

To enhance model precision, we will extract specific features that improve spam detection accuracy and precision. Such features include:

1. Term Frequency: Measures how frequently a term appears in a document to represent its importance.
2. Inverse Document Frequency: Highlights the uniqueness of a term by reducing the weights of common words.
3. Vectorization: Convert text into vectors that can represent semantic relationships and outputs contextualized vectors for an entire sequence of text.
4. Continuous-Bag-of-Words: Prediction of a target word based on its context.
5. N-grams: Using bi-grams, tri-grams to capture phrases.
6. Tokenization: Splitting text into separate tokens as part of the modeling process.
7. Word Embedding: Token of words are mapped to contextualized word embeddings, that captures meaning based on surround words
8. Pooling vectors: Combing or pooling all the tokens used for embedding sentence-level features.

## Phase 4: Modeling

### TF-IDF

We used term frequency to measure what words frequently appeared in our dataset. It detected common words like “to” or “the” that might appear more often but are not important in determining spam. Inverse Document Frequency is the opposite, it downplays the words that are common across our dataset. “Urgent”, “call” or “click”, could be words more often used in spam messages vs ham. IDF lets us weigh these words more relative to spam in our classification process.

Using TF-IDF lets us transform the text messages into vectors of numerical values, depicting the relevance of each word within spam and ham. The vectors are fed into a classifier, learning patterns correlated with spam and ham messages. TF-IDF is not as thorough as BERT but gives us a solid baseline model, providing understandable results and helping to identify spam according to the frequency and presence of keywords.

After the data was cleaned. I was able to apply vectorization to the cleansed data which improved our overall recall, F1-Score, and Accuracy. See the model below:

**Confusion Matrix: Before**

A screenshot of a computer

Description automatically generated

I was able to set the maximum document frequency which ignores terms that appear in this proportion, which helps remove common words that aren't informative. I was also able to include n-grams (unigrams and bigrams), and limiting the number of features which only keeps the top terms by frequency and importance. This led to improving the recall from 75 to 87. Our F1 Score improved from 85 to 93, meaning the model now correctly identifies a higher proportion of spam messages.

**Confusion Matrix: After**

A screenshot of a computer screen

Description automatically generated

### Word2Vec

Word2vec is not a single algorithm, instead it is a collection of models composed of different types of optimizations that are implemented to learn about word embeddings from large datasets.  Word embeddings have proven to have positive success on a variety of natural language processing tasks.  The two models utilized are continuous bag-of-words and continuous skip-gram.  Bag-of-words predicts the middle word based on the words around it to provide context for the middle word, the order of the words in this context is unimportant.  Skip-gram model is used to predict words within a certain range before and after the current word for providing context.

Bag-of-Words is a useful tool in natural language processing as it is simple and inexpensive to compute resources.  Bag-of-words change words into number representation which allows the computer to understand how frequently each word occurs within that sentence or document.  This is a crucial first step in natural language processing allowing for further modeling based on other more complex models.  Changing the words into a vector (read: a number) allows the model to quantify how often that word appears in a document. For classifying and sentiment analysis it is useful as word order does not matter, generally this is not an issue, however, word order can have much deeper implications.  Bag-of-Words does not understand the meaning of words surrounding them, which can lead this method to have issues factoring in word context.  This type of model always represents text in the same fashion, meaning the words are all the same length, a number.  The Bag-of-words model needs to know every word within the language it is interpreting to ensure that a number can be selected to prevent collisions. Also, of note that bag-of-words used is technically continuous bag-of-words (CBOW). This type of model relies on the surrounding words to enable the model to predict target words.  CBOW is based on a neural network with a single layer that learns via dense vector representations of words enabling this type of model to capture the semantic and syntactic properties of the word.

The other method that was utilized is skip-gram, which is based on a neural network architecture that predicts the words surrounding the target word.  This model you provide a word as an input, then assigns a heavier weight to the words that are closer to the target word than words farther away from the target word.  This model is utilized across many NLP tasks, such as attempting to determine the sentiment of the words, machine translation and is often used for spoken word tagging.  Compared to bag-of-words, skip-gram focuses on capturing the selected word which is highly useful for rare words.  After the skip-gram model is trained on the learned word embeddings based on vector space, words that are semantically and syntactically similar are located closer to each other, while words that are not will be farther apart.

To further illustrate words that were labeled as spam by the python extension word cloud was implemented which provides a great illustration.  The word cloud presents larger versions of the word and closer to the center of the screen the more frequently that a word is used and when viewing the spam, the consumer can see that the words that are aligned with spam texts.

A close-up of words

Description automatically generated

Along with the spam we also used the regular texts, which are referred to as ham messages in the word cloud.  Using the same illustrative rules as the spam word cloud the ham words that are generally found in actual messages.  Note that nearly none of the spam words are found in the ham word cloud, which is very interesting.

A close-up of words

Description automatically generated

### 

### BERT

**Introduction to BERT**

Bidirectional Encoder Representation from Transformers (BERT) is a powerful pre-trained transformer model used for NLP tasks. BERT has the capability to read entire sequences of words bidirectionally and capture the context of the words based on the surrounding words. This feature allows BERT to understand nuanced relationship between words, phrases, and whole sentences, making it ideal for tasks that require a deeper understand of text and semantics such as classifying fraudulent SMS messages.

BERT is pre-trained on many texts such as books and articles. The mass ingestion of information is what allows BERT to ascertain the patterns of various sentence structure, vocabulary, and nuance of the English language. Due to this training, the base BERT model can grasp rules of grammar and structures of different sentences with different syntax. It learns how words are commonly used together and how to determine contextual meaning based on the surrounding words.

**BERT Architecture**

To further elaborate on what a transformer model is, we’ll look at the encoder used by BERT and the Self-Attention mechanisms it uses for weight adjustment. BERT consists of multiple layers, the base model we’re using has 12 layers with 768 neurons. Each of these layers process the input from the previous layer and refines the understanding of nuanced textual relationships. BERT does this by first converting each sentence into tokens, and each token into embeddings. An embedding is a numerical representation of the token and its information and position in the sentence. The output of the forward pass uses the self-attention mechanism to adjust weights for the model to generate more and more refined embeddings. Each word gets assigned three vectors and the model will use the three vectors to calculate a score to understand how much attention each world should pay attention to other words. The model then uses the scores to create a weighted sum that tells it which words should be emphasized in that specific context.

BERT’s architecture consists of stacking encoder layers and self-attention mechanisms to enable it to capture deep non-linear relationship in text which is critical for NLP tasks like ours. It must understand the full context and semantics of each message.

**Fine-tuning BERT for Spam Detection**

To adapt the pre-trained BERT model for our specific task of spam detection, we considered various approaches. BERT must understand the full context and semantics of each message for accurate classification especially for classification of SMS messages which are irregular and non-conforming to traditional text and articles.

Initially, we prioritized the best performance. We considered training the BERT model ourselves but decided against it because of the lack of computational power. We attempted to train only the BERT header; however, that also proved unfeasible. Instead, we devised an alternative method to leverage BERT's capabilities while working within our computational constraints.

Our solution involved extracting the vectors from the BERT encoder rather than using the model's direct classification abilities. By saving these vectors to the runtime, we can then create a local neural network consisting of a few hidden layers and use the saved vector outputs of BERT as the input to our local and trainable neural network model. This allowed us to bypass the need for training the entire BERT model or the BERT header. By utilizing the pre-trained BERT model's vector representations, we could still benefit from its knowledge and context understanding while automating preprocessing and encoding.

We used the ADAM optimizer, which automatically adjusts the momentum based on the model's performance during training which allowed us to focus on tweaking our learning rate without worrying about missing the local minima. This approach enables us to harness the power of BERT while working within our resource constraints, ultimately leading to an effective spam detection model. To optimize our small neural network model for this project, we performed hyperparameter tuning. Initially, we had severe overfitting issues so to combat overfitting, we employed techniques such as high dropout rates for both layers and L2 regularization for both layers. These measures help prevent the model from memorizing the training data and improve its ability to generalize.

**Data Preparation for BERT**

We leveraged the functionality provided by the TensorFlow Hub for data preparation tasks. We imported the BERT preprocessor, tokenizer, and encoder directly from the hub, streamlining the data preprocessing workflow.

The BERT tokenizer was used to tokenize the SMS messages, breaking them down into individual words or sub-words. No additional preprocessing steps were applied to the dataset.

In this iteration of the model, we did not employ any data augmentation techniques; However, we plan to utilize more robust datasets and address class imbalance issues through undersampling and oversampling techniques in future iterations of this model. These methods will help adjust the spam-ham ratios and improve the model's performance on a more diverse range of messages. While data augmentation was not used in this version, it remains a promising avenue for future iterations to enhance the model's robustness and generalization capabilities.

**Training and Evaluation**

We divided our dataset into three subsets: 70% for training, 15% for validation, and 15% for testing. Initially, we started with 50 epochs but observed that the model's performance plateaued without reaching our predefined stopping patience count. We increased the number of epochs to 100, 200, and finally 300. With 300 epochs, the model converged in the 200s and exhibited minimal noise.

To assess the performance of our BERT-based spam detection model, we employed several evaluation metrics, including accuracy, precision, recall, and F1 score. These metrics provides insight to the model's effectiveness in correctly identifying spam and ham messages.

The final model evaluation on the test set yielded the following results:

* Test Accuracy: 97.97%
* Test Loss: 0.1277

The classification report further breaks down the model's performance for each class:

| Class | Precision | Recall | F1-score | Support |
| --- | --- | --- | --- | --- |
| Ham | 0.98 | 0.99 | 0.99 | 724 |
| Spam | 0.94 | 0.90 | 0.92 | 112 |

The confusion matrix analysis provides insights into the model's predictions:

* True Negatives (Ham correctly identified): 718
* False Positives (Ham incorrectly marked as spam): 6
* False Negatives (Spam incorrectly marked as ham): 11
* True Positives (Spam correctly identified): 101

Key metrics for spam detection:

* Precision (Spam prediction accuracy): 0.9439
* Recall (Spam detection rate): 0.9018
* F1 Score: 0.9224

These results signify the model’s strong performance in being able to generalize to unknown text messages and correctly and precisely identify them as spam or ham. The high precision and recall metrics are satisfactory with our expectation of providing a non-intrusive feature for customers due to minimized false positive and false negative rates.

**Interpretation and Analysis**

The BERT model performed exceptionally well as a spam detection filter. It achieved a high test accuracy score of 97% and a low test loss of 0.1277. We can confidently say that this model is able to effectively distinguish between spam and ham messages.

Taking a deeper look at the model’s ability to identify ham messages, we see that the model achieved a precision of 98%, which is a key indicator of the performance of our model due to the necessity of needing to reduce the possibility of legitimate text messages marked as spam. The model’s performance in determining spam messages yielded a precision of 94% and a recall of 90%. We can conclude that the model is effective at both spam and ham detection; however, there are signs that spam detection can be improved in later iterations.

Looking at our confusion matrix, see that there are 724 ham messages and 718 of them were correctly identified. Only 6 messages were misclassified as spam. Out of 112 spam messages, the model was able to detect 101 correctly while misclassifying 11.

Future improvements will focus on improving spam detection while maintaining current ham detection performance.

**Visualization**

A graph of a function

Description automatically generated with medium confidence

In the above image, we visualize our model’s accuracy and loss during training and validation. As we increased our epoch, the training accuracy improved, and our validation accuracy closely followed. This indicates that our model is effective at generalizing to unseen data. The model loss curve decreases rapidly for both training and validation and both seemingly converge, indicating that there is not overfitting.

A group of graphs showing different colors

Description automatically generated

In the above visualization, we can see a representation of our confusion matrix. Along with it, we have the ROC-AUC curve which helps demonstrate the model’s performance indicated by a score of 0.99. This means that this model has a high true positive rate and a low false positive rate. The precision-recall visual confirms our analysis on the validity of this model in minimizing false positive and false negative results. The prediction distribution visualization indicates that the model can confidently separate and classify messages as ham or spam with a very clear distinction between the two.

**W.I.P.**

## Phase 5: Evaluation

* Summarize the key outcomes from your initial model development, including evaluation metrics like accuracy, precision, recall, or F1 scores.
* Include visualizations like confusion matrices, ROC curves, or feature importance graphs.

## Phase 6: Implementation

* Interpret the preliminary results and provide insights into what they mean in the context of the problem.
* Mention challenges encountered and next steps for further refinement.

# Conclusion

* Summarize what you have achieved so far and outline the next phases (evaluation and deployment) for the final report.

*~~This project aims to develop a high-precision, reliable spam detecting model that~~*

*~~protects users from phishing and spam messages while enhancing overall~~*

*~~customer satisfaction. By reducing user engagement in the filtering process, we~~*

*~~streamline communication and minimize potential risks associated with fraud and~~*

*~~scams, safeguarding user’s private information and security.~~*

*~~Our model’s focus on key performance indicators like precision, specificity, and the~~*

*~~F1 score ensures that an approach that is technically sound and scalable while~~*

*~~aligning with the company’s business objective of delivering a more secure SMS~~*

*~~service. As the model evolves, it will allow us to offer customers a safer, distraction free communication experience, adding measurable value to our brand.~~*

*~~From a business perspective, this initiative has strong potential to become a profit~~*

*~~generator by reducing costs associated with fraud mitigation and by increasing~~*

*~~brand recognition and trust. Differentiating from our competitors will contribute to~~*

*~~customer loyalty and better position us in the marketplace as a trusted provider of~~*

*~~secure communications.~~*

*~~As we continue to refine the model based on real-world data and adapt to~~*

*~~emerging trends, this project will remain flexible, with ongoing enhancements that~~*

*~~ensures we meet current and future requirements. This proposal presents a~~*

*~~solution that not only mitigates spam and scam risks but also fortifies our business~~*

*~~as a trusted and innovative service provider.~~*

# References

* Include citations for any literature, datasets, or tools used, formatted according to your university’s citation style (e.g., APA, IEEE).

# Appendix (Optional)

* Include code snippets, more detailed charts, or supplemental information that supports the main report.