**IT3389 Applied ai PROJECT**

**PROJECT PROPOSAL**

|  |  |
| --- | --- |
| **Submitted By:** | Gangula Karthik (223715Y)**\*** |
|  | Ng Jun Ming (220080D) |
|  | Choy Wei Jun (221814H) |
|  | Seah Pin Shien (220273H) |
| **Team Name:** | Team 3 |
| **Tutor:** | Dr Veronica Lim |
| **Submission:** | 16 February 2025 |

# **Table of Contents**

[Introduction 4](#_Toc190625733)

[Solution 5](#_Toc190625734)

[Model Performance 6](#_Toc190625735)

[**Hateful Meme Classification** 6](#_Toc190625736)

[**Text Classification Approach: BERT** 7](#_Toc190625737)

[**Image Classification Approach: ResNet** 7](#_Toc190625738)

[**Multimodal Classification Approach: CLIP** 8](#_Toc190625739)

[**Multimodal Classification Approach: MemeCLIP Architecture** 12](#_Toc190625740)

[**Evaluation using Test Set** 13](#_Toc190625741)

[**Deep Fake Detection** 17](#_Toc190625742)

[**Xception Model** 18](#_Toc190625743)

[**Evaluation using Test Set** 20](#_Toc190625744)

[**Explainability** 21](#_Toc190625745)

[**Fake News Detection** 22](#_Toc190625746)

[**Base Models** 23](#_Toc190625747)

[**One-Class Neural Network** 23](#_Toc190625748)

[**Deep Learning Base Model Findings** 23](#_Toc190625749)

[**Fine-Tuned Neural Networks** 24](#_Toc190625750)

[**Evaluation using Test Set** 26](#_Toc190625751)

[**Explainability** 28](#_Toc190625752)

[**Mental Health Classifier** 28](#_Toc190625753)

[**Classical Models** 28](#_Toc190625754)

[**Deep Learning Models** 29](#_Toc190625755)

[**Evaluation using Test Set** 31](#_Toc190625756)

[Additional Features 32](#_Toc190625757)

[**Harmful Meme Classifier** 32](#_Toc190625758)

[**Deep Fake Detection** 33](#_Toc190625759)

[**Fake News Detection** 34](#_Toc190625760)

[**Mental Health Classifier** 34](#_Toc190625761)

[User Experience 34](#_Toc190625762)

[**Hateful Meme Classification** 36](#_Toc190625763)

[**Deep Fake Detection** 38](#_Toc190625764)

[**Fake News Detection** 39](#_Toc190625765)

[**Mental Health Classifier** 41](#_Toc190625766)

[Reflection 43](#_Toc190625767)

[**Group Reflection** 43](#_Toc190625768)

[**Individual Reflection** 43](#_Toc190625769)

[Appendix 46](#_Toc190625770)

# **Introduction**

As a global leader in digital adoption with 93% internet penetration, Singapore faces critical challenges in balancing rapid technological growth with online safety. Despite frameworks like the Online Safety Act and the SG$50 million Centre for Advanced Technologies in Online Safety (CATOS), harmful content exposure persists: 60% of children experienced online harm in 2023, while two-thirds of users encountered toxic content such as cyberbullying and deepfakes. Only 25% of incidents are reported, highlighting systemic gaps in detection and user empowerment. Addressing these challenges is of the utmost importance to Singapore’s commitment to sustainable digital development, aligning with UN SDGs 11 (Sustainable Cities) and 16 (Peaceful Societies) while reducing risks to social cohesion and economic stability. The 4 interconnected challenges which have been addressed include:

|  |  |  |
| --- | --- | --- |
| Topic | Problem Statement | Solution |
| Hateful Meme Classification (Karthik) | * Memes combining text and imagery are increasingly weaponized to spread divisive rhetoric, often amplifying racial, religious, or political tensions. For instance, memes targeting marginalized groups have been linked to real-world violence, such as racially motivated attacks fueled by online radicalization. | * Advanced multimodal AI models analyze both visual and textual elements to detect harmful/hateful memes. |
| Mental Health Classification (Pin Shien) | * Exposure to toxic content on social media correlates with rising anxiety, depression, and suicidal ideation among youths. Vulnerable groups, particularly young women, face heightened risks due to cyberbullying and algorithmic amplification of harmful content. | * Deep learning models trained on social media data identify high-risk content and user behavior. |
| Deepfake Classification (Wei Jun) | * + Deepfakes are increasingly used for financial scams, political manipulation, and identity theft. Recent incidents include synthetic media impersonating public figures like Prime Minister Lee Hsien Loong to promote fraudulent schemes. | * CNN-based classifiers to detect synthetic media by analyzing inconsistencies in the deepfake images. |
| Fake News Classification (Jun Ming) | * Misinformation spreads rapidly through manipulated narratives, eroding public trust. For example, fabricated claims during elections and health crises have fueled panic and polarization. | * Deep NLP models trained on fake news datasets (e.g., LIAR2 and PolitiFact) to classify claims using contextual analysis. |

# **Solution**

The core solution involves a web application that allows users to test the different online safety features before purchasing/integrating them into their workflows and daily lives.

The core technologies used for the solution include:

1. **Frontend:** NextJS and ReactJS were chosen for a fast and SEO-friendly frontend.
2. **Backend:** FastAPI was used for its high performance and automatic generation of documentation.
3. **Models:** TensorFlow was used to develop efficient deep learning models with a familiar and easy to use API.

The solution that was developed is deployed on google cloud with the microservice architecture instead of the traditional monolithic architecture. In this, the frontend and backend were deployed separately and within the backend, which consisted of many services, such as the harmful meme service, deepfake service, fake news service, etc., were also deployed separately.

A diagram of a service

AI-generated content may be incorrect.

*Figure 1 – architecture diagram of deployed solution*

1. **Client Applications:** 
   1. **Poseidon Playground (Web Application):** A React and NextJS based web application for users to experiment and try out different security features.
2. **API Gateway:** 
   1. The API Gateway acts as an intermediary between client applications and backend microservices. It allows for more security and better API management.
3. **Microservices:** 
   1. Adopting a microservices approach helps to ensure scalability, fault isolation, and independent deployments. Even if one of the services fails, it allows the other services to continue operation without affecting the others. Additionally, this architecture allows for faster development cycles and easier maintenance.

Below is the summary of the cloud run deployment that was done on google cloud:

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 2 – summary of cloud run deployments*

All the microservices are online and successfully able to serve the requests. Below is the requests and the time taken to serve these requests, which is in real-time / near real time.

A graph with blue lines

AI-generated content may be incorrect.

*Figure 3 – Showing request count over time, instances are able to successfully serve requests*

# **Model Performance**

## **Hateful Meme Classification**

To understand which is the most suitable model, several models have been experimented with along with a diverse set of parameters. Roughly there are all the approaches can be grouped into 3 categories that have been explored:

1. **Text Classification Approach:** This approach uses both a frozen and unfrozen BERT classifier on meme captions / meme descriptions (cases where there is no text on the meme).
2. **Image Classification Approach:** This approach uses only the meme image for the model training. The meme text from the meme has been inpainted over to prevent the model from looking at the textual information. A frozen and unfrozen resnet 50 model was used.
3. **Multimodal Classification Approach:** This approach uses both the image and text for classification. The CLIP embeddings were used with the frozen CLIP embeddings and a fine tuned CLIP embeddings and model based on the MemeCLIP architecture (only difference is that a cosine classifier was not used) [1].

### **Text Classification Approach: BERT**

|  |  |
| --- | --- |
| A screenshot of a computer  AI-generated content may be incorrect. | A screenshot of a computer  AI-generated content may be incorrect. |
| *Figure 4 – Bert frozen model classification report* | *Figure 5 – Bert unfrozen model classification report* |

* The unfrozen model exhibits severe class imbalance, achieving perfect recall (1.00) but low precision (0.55) for class 0, while completely failing to identify class 1 (precision/recall = 0.00). This suggests overfitting to dominant class patterns during fine-tuning, potentially exacerbated by an imbalanced dataset.
* Compared to the frozen model, the unfrozen model shows worse generalization for class 1, despite higher overall accuracy (55% vs. 48%). The frozen model, though weaker overall, maintains a more balanced performance across classes, indicating it may be less overfit and better at preserving learned representations.

### **Image Classification Approach: ResNet**

Two ResNet50 models were evaluated: one with all layers frozen (pretrained on ImageNet, excluding the classification head) and another with all layers unfrozen for fine-tuning. Both models incorporate a custom classification head consisting of a **GlobalAveragePooling2D** layer, a **Dense(1024, ReLU)** layer for additional learning capacity, and a final **Dense(1, sigmoid)** layer for binary classification.

|  |  |
| --- | --- |
| A screenshot of a black screen  AI-generated content may be incorrect. | A screenshot of a computer screen  AI-generated content may be incorrect. |
| *Figure 6 – Frozen RESNET classification report* | *Figure 7 – Frozen RESNET classification report* |

* Both models achieve nearly identical performance (~64% accuracy), with the frozen model showing slightly better recall for class 1, while the unfrozen model marginally improves precision for class 0. However, these differences are minor, indicating no clear advantage of fine-tuning in this scenario.
* The lack of significant improvement in the unfrozen model suggests that fine-tuning the entire ResNet50 backbone does not necessarily enhance performance, possibly due to overfitting, insufficient training data, or suboptimal fine-tuning strategies.
* This also shows that the image component contains much more information in predicting how harmful a meme is compared to the text component.

### **Multimodal Classification Approach: CLIP**

CLIP embeddings (512-dimensional) are extracted from a pretrained model for both image and text inputs. These embeddings are fused via vector multiplication to create a unified representation, which is then used as input for a neural network trained to make predictions.

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 8 – simple neural network trained on clip embeddings*

A graph of a line and a line

AI-generated content may be incorrect.

*Figure 9 – loss and accuracy curves*

* The model exhibits increasing validation loss while training loss continues to decrease, accompanied by a widening gap between training and validation accuracy across epochs, indicating poor generalization.
* The instability in validation loss and the model's tendency to overfit suggest that the architecture may be too simplistic, requiring regularization techniques, additional data, or a more sophisticated fusion strategy to improve generalization.

To mitigate overfitting, **batch normalization**, **dropout (0.3)**, and **L2 regularization** were incorporated into the Dense layers, aiming to improve generalization and stabilize training. The number hidden layers also increased to 4 from 2.

A screenshot of a computer program

AI-generated content may be incorrect.

*Figure 10 – updated neural network*

A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

AI-generated content may be incorrect.

*Figure 11 – accuracy and loss curves for updated neural network*

* While the loss curves now decrease at a more stable rate, the gap between training and validation performance continues to widen, indicating persistent overfitting.
* Despite the applied regularization techniques, the model still struggles with generalization, suggesting that further adjustments, such as data augmentation, stronger regularization, or a more complex architecture, may be necessary.

Increased **dropout** from 0.3 to 0.6 and introduced **skip/residual connections** to improve regularization and gradient flow, aiming to enhance model stability and generalization.

A screen shot of a computer

AI-generated content may be incorrect.

*Figure 12 – updated neural network with skip connection*

A graph of a graph of a graph

AI-generated content may be incorrect.

*Figure 13 – updated loss and accuracy curves with skip connections*

* Loss curves now show steady decline, but validation accuracy has become more erratic, suggesting potential instability in optimization or sensitivity to hyperparameter choices.
* Planning further refinements by switching the optimizer from **Adam to AdamW** and experimenting with the **learning rate** to improve convergence and generalization.

A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

AI-generated content may be incorrect.

*Figure 14 – accuracy and loss curves with adamW optimizer*

*Some additional parameters that were*

A screenshot of a computer screen

AI-generated content may be incorrect.

*Figure 15 – model validation results with adamw optimizer*

* The model achieves an overall accuracy of 74%, with a macro F1-score of 0.74, indicating balanced performance across both classes.
* Recall for the harmful class (1) is 0.61, meaning the model correctly identifies 61% of harmful cases but misses 39%, which may be a concern depending on the application. The precision for class 1 is 0.78, suggesting that when the model does predict harm, it is fairly confident in its decision.
* In contrast, class 0 (non-harmful) has a higher recall of 0.86, meaning the model is much better at correctly identifying non-harmful cases, but this comes at the cost of missing more harmful instances.
* If recall for class 1 is the key priority, the model still needs to be adjusted to improve sensitivity.

### **Multimodal Classification Approach: MemeCLIP Architecture**

In the memeCLIP architecture [1], the CLIP encoder remains frozen, while linear projection layers (for both image and text) and feature adapters with ReLU activation are trained. These layers refine embeddings by mapping them into a common space and enhancing feature representations. The overall architecture and hyperparameters remain consistent with prior experiments.

**A diagram of a process

AI-generated content may be incorrect.**

*Figure 16 – meme clip architecture (clip embeddings are being finetuned)*

* Performance improves as the learning rate decreases, but once it drops below 1e-6, accuracy plateaus at 50%, indicating diminishing returns from further reductions.
* At 1e-5 learning rate, smaller batch sizes yield better performance, suggesting that lower batch sizes may help stabilize gradient updates and improve generalization.

A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

AI-generated content may be incorrect.

*Figure 17 – accuracy vs loss curves for meme clip*

* Increasing the number of hidden layers reduces loss but does not meaningfully improve accuracy, implying that the model’s capacity is not the primary bottleneck and that further refinements may be needed elsewhere (e.g., optimization strategy or data representation).

### **Evaluation using Test Set**

A blue squares with white text

AI-generated content may be incorrect.

*Figure 18 – confusion matrix*

* The model correctly identifies 135 true negatives and 176 true positives, but it also produces 115 false positives and 74 false negatives, indicating challenges in precision and recall when distinguishing between harmful and harmless memes.
* The high number of false positives suggests a tendency to over-flag harmless content, which may lead to unnecessary moderation. However, the lower number of false negatives indicates a better ability to catch harmful memes, which is preferable from a safety perspective.

A blue line graph with white lines

AI-generated content may be incorrect.

*Figure 19 – ROC curve*

* The model achieves an **AUC of 0.62,** indicating that its ability to distinguish between harmful and harmless content is only **marginally better than random guessing**, which is suboptimal for real-world deployment. While the performance is not ideal, it remains **acceptable given the resource constraints** and in comparison to other models evaluated, making it a **trade-off between efficiency and accuracy.**

**A graph with lines and symbols

AI-generated content may be incorrect.**

*Figure 20 – precision vs recall curve*

* The precision-recall curve highlights a trade-off: as the model captures more harmful memes (higher recall), its precision declines, leading to more false positives.
* This suggests that tuning the decision threshold is crucial, depending on whether the priority is maximizing recall to catch more harmful content or improving precision to reduce false alarms. For now it seems that a threshold between 0.5 is ideal.

**A graph of a number of graphs

AI-generated content may be incorrect.**

*Figure 21 – class separation probability distribution*

* In an ideal classifier, there would be minimal overlap between the predicted probability distributions of harmful and harmless memes, with harmful memes clustering near 1 and harmless memes near 0.
* While some overlap exists in the **0.75 to 1.0 range**, the majority of **harmless predictions cluster around 0** and **harmful predictions around 1**, suggesting that the model is performing reasonably well at distinguishing between the two classes.

A graph with blue lines

AI-generated content may be incorrect.

*Figure 22 – calibration plot*

* The model demonstrates high confidence in predicting harmful memes (0.9-1.0 probability), but the actual occurrence of harm is lower than expected, indicating overconfidence in certain predictions. Additionally, the model shows fluctuating confidence around mid-range probabilities (0.4-0.8), likely due to challenges in integrating multimodal data, which results in inconsistent predictions.

A graph with a line and a line

AI-generated content may be incorrect.

*Figure 23 – cross validation across different folds*

A screenshot of a black background with white text

AI-generated content may be incorrect.

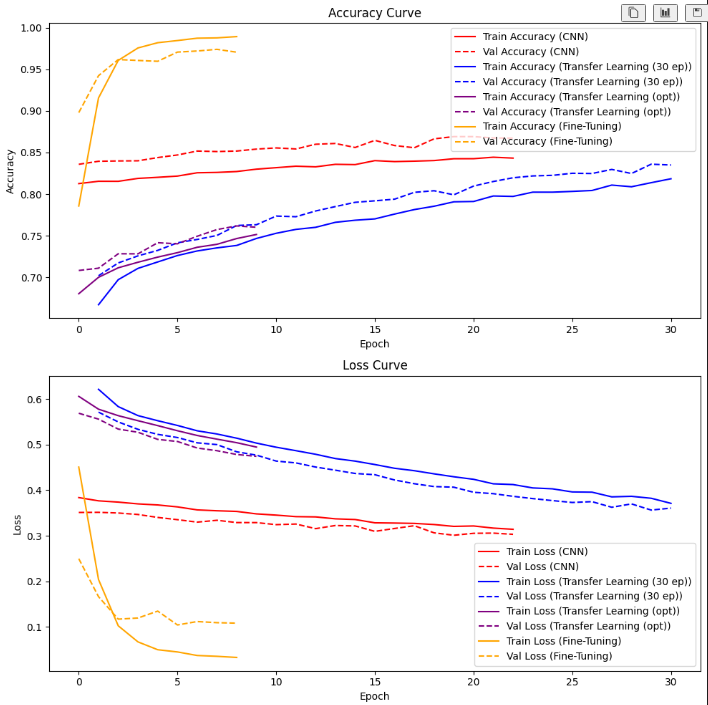
*Figure 24 – Average result from different folds in cross validation*

* Model performance for precision, recall and F1-score, is quite consistent across the different folds with an average 0.76 in recall for the harmful class. This suggests that the model is able to generalize effectively across the dataset.

## 

## **Deep Fake Detection**

These are the best models from training.

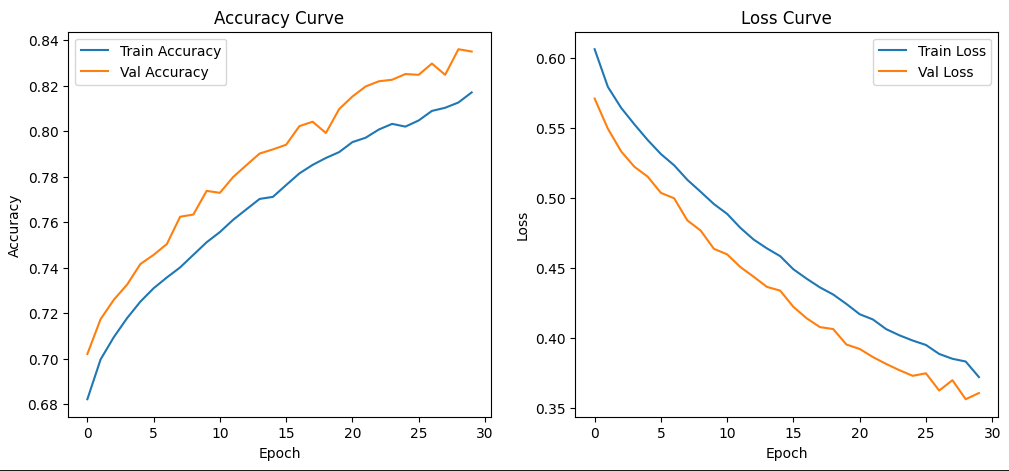


*Figure 25 – accuracy and loss curves summary*

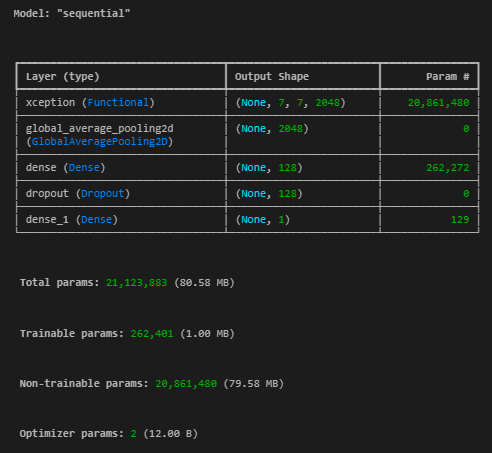
I chose the Transfer Learning model trained for 30 epochs over the CNN and Transfer Learning with optimization due to its smaller size and better real-time performance. At 80MB, it’s the smallest model, while the Transfer Learning with optimization is 240MB, and the CNN from scratch is 450MB. Larger models are slower and more resource-intensive, making them less suitable for real-time detection. The 30-epoch Transfer Learning model strikes the best balance between size, efficiency, and speed for deployment, and still hitting the target 80% accuracy.

Early stopping was used hence why some models trained for less epochs compared to others

Looking at the training results from the model, transfer learning (30 epoch), the model is generalizing well to the data, the model performing better on the validation data compared to the training. Validation accuracy is higher, and validation loss is lower compared to the training set.



*Figure 26 – accuracy and loss curves for transfer learning*



*Figure 27 – transfer learning on xception net*

### **Xception Model**

**Parameters**

1. **Base Model**: Xception (pre-trained on ImageNet)
   1. include\_top=False: Excludes the fully connected layers of the model.
   2. weights='imagenet': Uses pre-trained weights from ImageNet.
   3. input\_shape=(224, 224, 3): Input image size is set to 224x224 with 3 color channels (RGB).
2. **Base Model Freezing**:
   1. base\_model.trainable = False: Freezes the weights of the base model so that they are not updated during training.
3. **Custom Layers**:
   1. GlobalAveragePooling2D(): A pooling layer that reduces each feature map to a single value (average) to flatten the output.
   2. Dense(128, activation='relu'): Fully connected layer with 128 units and ReLU activation.
   3. Dropout(0.5): Dropout layer to prevent overfitting, with a 50% dropout rate.
   4. Dense(1, activation='sigmoid'): Output layer for binary classification (1 output unit with sigmoid activation).

**Compilation Parameters**

* **Optimizer**: adam: Adaptive moment estimation optimizer, commonly used in deep learning for faster convergence.
* **Loss Function**: binary\_crossentropy: Used for binary classification tasks.
* **Metrics**: accuracy: Tracks the accuracy of the model during training.

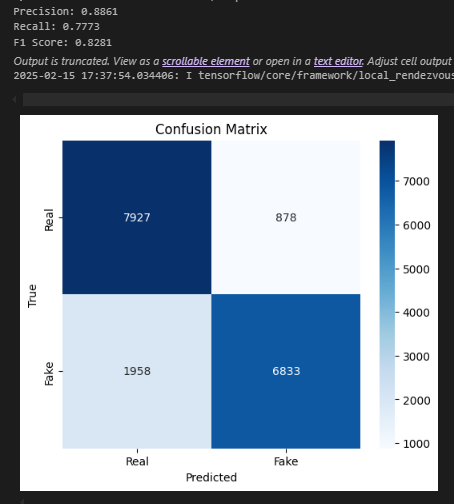
**Callbacks**

1. **EarlyStopping**:
   1. monitor="val\_loss": Monitors validation loss for stopping criteria.
   2. patience=3: Waits for 3 epochs before stopping training if no improvement is seen.
   3. restore\_best\_weights=True: Restores the weights from the epoch with the best validation loss.
2. **CSVLogger**:
   1. CSVLogger("training\_log\_X\_V2.csv", append=False): Logs the training history to a CSV file, overwriting previous logs.

**Others**

* **Epochs**: 30: The model is set to train for a maximum of 30 epochs.
* **Batch Size**: 64.

### **Evaluation using Test Set**



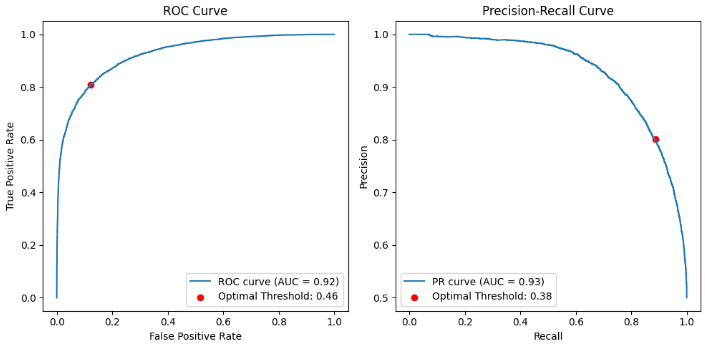
*Figure 28 – confusion matrix and other metrics*

*\*Note - Threshold used is 0.5*

* The **high precision** (88.61%) means the model is reliable when it says something is "Real," but there’s a **slight room for improvement in recall** (77.73%) to ensure fewer "Real" instances are missed.
* The **F1-score** is quite good at 0.8281, reflecting a solid balance between precision and recall. However, the **false negatives** (1958) could be a point of focus to increase recall and, as a result, boost the F1-score further.

To improve the model's performance:

* **Increase recall**: Focus on reducing false negatives. This could be achieved by adjusting the decision threshold or using methods like oversampling the minority class.
* **Improve model robustness**: Try to fine-tune the model to minimize both false positives and false negatives for a more balanced outcome.



*Figure 29 – ROC and PR-Curves*

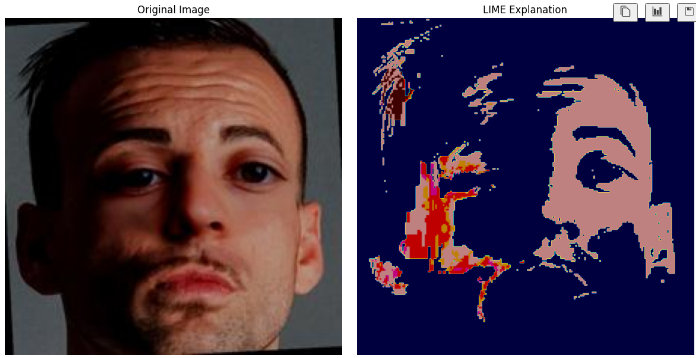
* **ROC Curve (Threshold = 0.46)**
  + The optimal threshold here is chosen to **balance True Positive Rate (TPR) and False Positive Rate (FPR)**.
  + Since ROC is good at assessing general classification performance, this threshold **optimizes the distinction between positive and negative classes**.
* **PR Curve (Threshold = 0.38)**
  + The Precision-Recall Curve is more useful when dealing with **imbalanced datasets**.
  + The optimal threshold here is chosen to **balance Precision and Recall**, which is different from balancing TPR and FPR.
  + Since recall is high at lower thresholds, the PR curve often suggests a **lower optimal threshold** than ROC.

As I want a balanced model, I will be using the ROC-based threshold (0.46).

### **Explainability**

The image below is a deepfake.

**Dark Red Areas:** These areas highlight the regions that have the most *negative* influence towards the "fake" class. In simpler terms, these are the parts of the image that the model "sees" as the strongest indicators of the image being a deepfake.



## **Fake News Detection**

During the Exploratory Data Analysis (EDA), it was observed that the dataset suffered from class imbalance. To address this issue, two techniques were employed:

* **Weighted Class:** Adjusting class weights during model training.
* **SMOTE (Synthetic Minority Over-sampling Technique):** Oversampling the minority class to balance the dataset.

A variety of machine learning and deep learning models were tested, including:

**Classical Machine Learning Models**

* Logistic Regression
* Naïve Bayes
* Support Vector Machine (SVM)
* Random Forest
* Gradient Boosting
* Simple Neural Network

**Deep Learning Models**

* Long Short-Term Memory (LSTM)
* Convolutional Neural Network (CNN)
* Feedforward Neural Network
* One-Class Neural Network

Additionally, various encoding techniques were explored:

* **TF-IDF (Term Frequency-Inverse Document Frequency)**
* **GloVe (Global Vectors for Word Representation) – glove.6B.300d**
* **BERT (Bidirectional Encoder Representations from Transformers) – DistilBERT**
* **Word2Vec**

### **Base Models**

Initial experiments with classical models revealed that Random Forest outperformed others in terms of F1 score (Macro F1 Score: 0.72). However, a simple Neural Network performed poorly in comparison when using TF-IDF with the top 5000 features.

Upon further optimization, the **Random Forest model performed well**, but a **Feedforward Neural Network (FNN) eventually surpassed it (Macro F1 Score: 0.75)**. Given that Neural Networks, especially **Recurrent Neural Networks (RNNs), are well-suited for handling sequential data**, the focus shifted towards deep learning models with advanced embeddings like BERT and GloVe for improved generalization.

### **One-Class Neural Network**

An anomaly detection approach was tested using a One-Class Neural Network. The model was trained solely on the majority class (Fake News) and classified uncertain instances as True News. However, this method performed poorly (Macro F1 Score: 0.49) and was discarded after testing with both GloVe and Word2Vec embeddings.

### **Deep Learning Base Model Findings**

**Adam vs Stochastic Gradient Descent (SGD)**

A comparative study was conducted between Adam and Stochastic Gradient Descent (SGD) optimizers. Results indicated that **SGD consistently outperformed Adam** by an F1-score margin of 0.05 when using GloVe embeddings (Macro F1 Score: 0.75 for SGD, 0.70 for Adam). Similar trends were observed for Word2Vec, leading to the adoption of SGD for further experiments.

**Word2Vec, GloVe, and BERT Findings**

* **Word2Vec:** Underperformed compared to other embedding techniques (Macro F1 Score: 0.60).
* **GloVe:** Performed well and provided competitive results (Macro F1 Score: 0.75).
* **BERT:** Scored close to GloVe (Macro F1 Score: 0.72).
* **CNN models were dropped** due to subpar performance.

### **Fine-Tuned Neural Networks**

**GloVe-Based Model**

A **Feedforward Neural Network with oversampling** was optimized and achieved strong performance (Macro F1 Score: 0.78):

* **Model Type:** Sequential Feedforward Neural Network
* **Handling Imbalance:** Oversampling applied

**BERT-Based Model**

For **BERT-based models**, various architectures were tested:

* The **best-performing model** utilized **5 hidden layers with a learning rate of 0.001** (Macro F1 Score: 0.74).
* A **Recurrent Neural Network (RNN) with 2 LSTM layers and 3 Feedforward layers** closely followed in performance (Macro F1 Score: 0.73).
* Transfer learning with frozen DistilBERT embeddings and a custom LSTM-based classifier, consisting of one 64-unit LSTM layer and a 64-unit dense layer, achieved an F1 macro score of 0.71.

After an extensive search, it was found that combining GloVe embeddings with a feed-forward neural network using the following hyperparameters yielded the best results:

A screenshot of a computer code

AI-generated content may be incorrect.**Best Parameters:**

* **Use L2 Regularization:** Yes
* **Use L1 Regularization:** No
* **Early Stopping Patience:** 10 epochs
* **Momentum:** 0.95
* **Learning Rate:** 0.001
* **Layer Sizes:** [512, 256, 128, 64, 128]
* **L2 Regularization Rate:** 0.001
* **L1 Regularization Rate:** 0.001
* **Number of Epochs:** 50
* **Dropout Rate:** 50%
* **Batch Size:** 32

With these settings, the model achieved an F1 score of 0.78, precision of 0.78, and recall of 0.77, surpassing the target of 0.75 for all metrics outlined in the proposal.

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 30 – Model Architectures*

A screenshot of a computer

Description automatically generated

*Figure 31 – Model performance classification report*

**Class 0 (Fake News)**

* **Precision:** 0.92 → When the model predicts class 0, it is correct 92% of the time.
* **Recall:** 0.93 → The model correctly identifies 93% of all actual class 0 instances.
* **F1-score:** 0.93 → A strong balance between precision and recall.

**Class 1 (True News)**

* **Precision:** 0.64 → When the model predicts class 1, it is correct 64% of the time.
* **Recall:** 0.61 → The model only correctly identifies 61% of actual class 1 instances.
* **F1-score:** 0.62 → A moderate performance, but significantly lower than for class 0

### **Evaluation using Test Set**

The macro average calculates the unweighted mean of precision, recall, and F1-score across both classes, treating them equally despite class imbalance. This helps assess the model’s performance on minority classes without bias toward the majority class.

* **Macro Avg (0.78 F1-score)** → The unweighted average of both classes’ performance, showing moderate balance.

**ROC and PRC Analysis**

To determine the optimal classification threshold, **ROC and PRC curves** were analyzed. The results suggested two threshold values: **0.02 and 0.59**. Further evaluation indicated the **best threshold based on SHAP analysis**

. A graph with a line and a point

Description automatically generated

*Figure 32 – Model performance ROC Curve*

A graph with a line going up

Description automatically generated

*Figure 33 – Model performance Precision-Recall Curve*

### **Explainability**

A screenshot of a graph

Description automatically generated

*Figure 34 – Model Explainability SHAP graph*

Feature importance was examined using **SHAP (SHapley Additive exPlanations)**. Key observations:

* Words like **"photo"** and **"show"** were strongly associated with Fake News.
* Reporting-oriented words like **"find"** were indicative of True News.
* **SHAP-based thresholds provided the most stable and interpretable results**

## **Mental Health Classifier**

### **Classical Models**

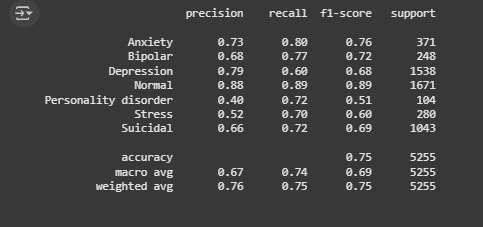
**Logistic Regression**

After preprocessing the text data by eliminating stop words, punctuation, and non-English words, the cleaned text is converted into numerical features using TF-IDF (Term Frequency-Inverse Document Frequency). TF-IDF is selected because it highlights less common yet significant words while reducing the weight of frequently occurring terms, making it well-suited for classical models such as Logistic Regression. The TfidfVectorizer is applied with max\_features=2000 to constrain the feature space, ensuring computational efficiency while preserving critical information for classification. This method allows the model to concentrate on meaningful features, enhancing classification accuracy by minimizing noise and emphasizing distinctive terms.



*Figure 35 – Logistic regression model*

The class\_weight='balanced' parameter was used in the Logistic Regression model to address class imbalance. It adjusts weights inversely proportional to class frequencies, ensuring the minority class receives adequate attention during training. This improves the model's ability to generalize and make more accurate predictions.



*Figure 36 – classification report of classical models*

The model achieved an overall accuracy of 0.75, with precision, recall, and F1-scores indicating its ability to balance identifying true positives and minimizing false positives. For instance, the "Normal" class had the highest performance, with an F1-score of 0.89, reflecting strong precision and recall. Similarly, "Anxiety" and "Bipolar" classes showed solid performance, with F1-scores of 0.76 and 0.72, respectively.

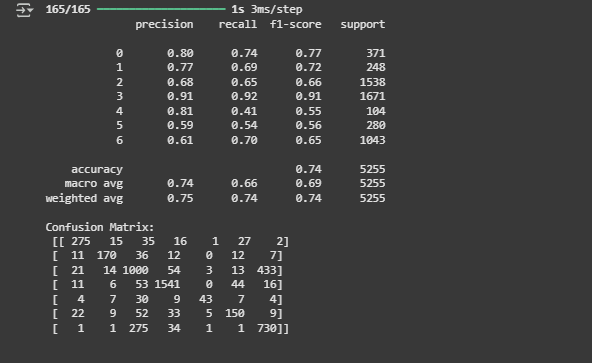
However, the model struggled with certain classes, such as "Personality disorder" and "Stress," which had lower F1-scores of 0.51 and 0.60, respectively. This suggests challenges in accurately identifying these conditions, possibly due to class imbalance or overlapping features with other categories. Despite these limitations, the model demonstrated reasonable performance for "Depression" and "Suicidal" classes, with F1-scores of 0.68 and 0.69, respectively. Overall, while the Logistic Regression model performs well for some classes, there is room for improvement, particularly in addressing the challenges posed by minority classes and enhancing precision and recall for underrepresented conditions.

### **Deep Learning Models**

For Deep Learning Models, I chose not to remove stop words or other commonly used words to enable the model to grasp the contextual meaning of the text. In deep learning models, particularly those utilizing word embeddings, every word plays a role in understanding sentence structure and context. Eliminating stop words could lead to the loss of crucial contextual relationships necessary for accurate predictions.

Unlike traditional machine learning methods, which often remove stop words to reduce dimensionality and noise, deep learning models like LSTMs, GRUs, and transformers depend on capturing dependencies between words across long sequences. By retaining all words, including frequently occurring ones, the model can better learn semantic representations and understand the relationships between words within the given context.

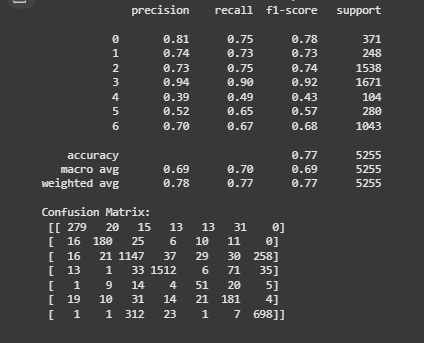
**CNN Model**



*Figure 37 – Classification report + confusion matrix of CNN model*

The CNN model demonstrated solid performance, achieving an overall accuracy of 0.75. It performed particularly well on the "Normal" class, with an F1-score of 0.91, indicating strong precision and recall. However, it struggled with minority classes like "Personality disorder" and "Stress," which had lower F1-scores of 0.55 and 0.56, respectively. This suggests that while the CNN effectively captured local patterns in the text, it faced challenges with imbalanced data and overlapping features in certain classes.

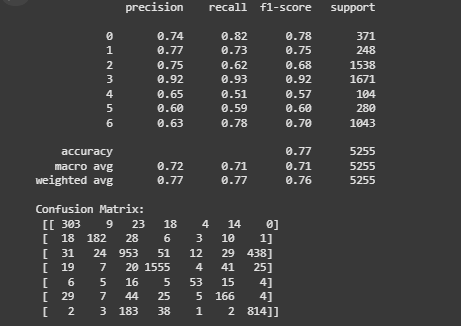
**LSTM Model (Best Model)**



*Figure 38 – Classification report + confusion matrix of LSTM*

The LSTM model outperformed the others, achieving the highest overall accuracy of 0.77. It excelled in classifying the "Normal" class, with an F1-score of 0.92, and showed consistent performance across other classes, such as "Anxiety" (F1-score: 0.78) and "Bipolar" (F1-score: 0.75). Its ability to capture long-term dependencies in text data likely contributed to its superior performance, especially in handling complex and nuanced mental health conditions.

**GRU Model**



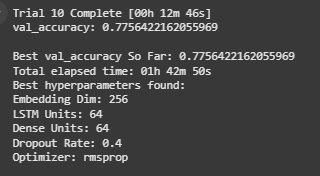
*Figure 39 – classification report + confusion matrix of GRU model*

The GRU model achieved an accuracy of 0.77, matching the LSTM in overall performance but with slightly lower F1-scores for some classes. For example, it performed well on the "Normal" class (F1-score: 0.92) but had lower scores for "Personality disorder" (F1-score: 0.57) and "Stress" (F1-score: 0.60). While the GRU's efficiency in processing sequential data was evident, it showed similar limitations as the CNN and LSTM in handling minority classes.

### **Evaluation using Test Set**

All three models—CNN, LSTM, and GRU—achieved comparable overall accuracy (~0.75–0.77), with the LSTM slightly outperforming the others. The LSTM's ability to capture long-term dependencies made it the best performer, particularly for complex text data. The CNN, while effective in extracting local patterns, struggled more with imbalanced classes. The GRU, though efficient, showed similar limitations to the LSTM but with marginally lower performance in some classes. Overall, the LSTM emerged as the most robust model for this classification task, balancing accuracy and generalization across all classes.

**Best Parameters**



*Figure 40 – Hyperparameter tuning*

The best hyperparameters found through tuning are:

* Embedding Dim: 256
* LSIM Units: 64
* Dense Units: 64
* Dropout Rate: 0.4
* Optimizer: RMSPROP

These parameters suggest that a relatively large embedding dimension and moderate dropout rate help in preventing overfitting while capturing sufficient contextual information.

# **Additional Features**

## **Harmful Meme Classifier**

The Harmful Meme Classification System is designed to detect and analyze harmful memes using a combination of a custom deep learning model and a Judge Vision-Language Model (VLM) powered by OpenAI's GPT-4 API. The system ensures high accuracy by incorporating a validation mechanism where the predictions of both models are logged and analyzed for continuous improvement. Below is a feedback loop that is being used.

**A diagram of a service

AI-generated content may be incorrect.**

*Figure 41 – model retraining loop with JudgeLLM evaluation*

This system proactively mitigates model bias through a multi-layered feedback loop that integrates cross-model validation, bias auditing, and continuous learning. By leveraging a dual-model approach—combining a custom deep learning model and Judge VLM (GPT-4)—it detects biases by flagging discrepancies, particularly in sarcasm or culturally specific contexts. Continuous monitoring ensures that when bias thresholds are breached, retraining pipelines activate to adapt the model to evolving meme trends. This framework aligns with best practices in fairness-aware AI, adversarial training, and proactive bias mitigation, ensuring equitable and accurate classification over time.

Below is the picture of the judgeLLM output and the model predictions being logged for human review in a PostgreSQL database.

A screenshot of a black screen

AI-generated content may be incorrect.

*Figure 42 – Logging predictions in database*

A card has also been included on the frontend to show the number of misclassifications:

A screenshot of a graph

AI-generated content may be incorrect.

*Figure 43 – Picture of JudgeLLM Validation Card*

## **Deep Fake Detection**

In deep fake detection, additional feature is implemented to increase the accuracy and efficiency of detection.

1. **Face Extraction:**

* Faces from image(s) are extracted using Google’s face detection API
* Facial features are the easiest way to determine if something is a deepfake, which helps improve accuracy
* Using an external API enhances speed and lowers costs on our end

## **Fake News Detection**

In fake news detection, two additional features are implemented to enhance the accuracy and efficiency of information retrieval, helping users make well-informed decisions:

1. **Article Retrieval:**
   * Relevant articles are sourced from two primary channels.
   * The first source is **NewsAPI**, which retrieves articles from reputable news outlets.
   * If no relevant articles are found through NewsAPI, a fallback mechanism utilizes **web scraping** from bing’s news tab upon searching the query.
   * Once retrieved, the articles are sorted and filtered based on **relevance**, determined by the **cosine similarity** score between the given statement and the first few paragraphs of each article.
2. **Article Summarization:**
   * Each retrieved article includes its **title, URL, and full text**.
   * To enhance readability and efficiency, the Gemini API leverages prompt engineering to generate concise summaries.
   * This feature provides users with a streamlined, user-friendly interface for **quick cross-referencing** and deeper verification of claims.

## **Mental Health Classifier**

In mental health classifier, each detected mental health state will have a tailored response message to provide emotional support and resources:

1. Customized Support Messages

* Based on the mental health status, a customised support message will be displayed
* Additionally, links to helplines, articles and support groups are also displayed
* This enhances the overall user experience by fostering a supportive and compassionate environment while encouraging individuals to seek help when needed.

# **User Experience**

The platform was built using React.js with Next.js, combining Shadcn UI for modular, customizable components and Hero UI (formerly Next UI) for dynamic animations and modern design elements. This architecture prioritizes user experience through clean interfaces while optimizing SEO via Next.js’ hybrid rendering (SSR/SSG) for fast load times and search engine visibility.

The website has a landing page with an interactive carousel showing the different features and a floating navbar for easy and aesthetic navigation.

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 44 – Image of the website landing page*

The website is also mobile responsive.

A screenshot of a phone

AI-generated content may be incorrect.

*Figure 45 – Image of the website landing page (mobile responsive)*

The frontend is deployed separately from the backend and is hosted on the Google Cloud Platform.

## **Hateful Meme Classification**

The hateful meme classifier page allows the user to upload an image where it will be processed and a label will be returned mentioning whether the meme is hateful or not.

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 46 – Image of the harmful meme classifier service*

This is how the website looks like initially when the user has not uploaded any meme. When the user uploads the meme, it returns the results as below:

|  |  |
| --- | --- |
|  |  |
| *Figure 47 - Result of hate meme* | *Figure 48 - Result of non-hate meme* |

To explain exactly what the user is looking at:

* **Label:** This indicates whether the meme is a hateful/non hateful meme. It is color coded with icons for more expressiveness to the user.
* **Confidence Score:** Shows the user how confident the model is in identifying whether the meme is harmful or not. The closer the number is to 1, the more likely the meme is hateful.
* **Processing Time:** This indicates how long it takes to return the result in seconds. Shows the user how fast the model is.

Besides the scores and results an additional feature has been implemented which will show the user an explanation on why the meme is harmful. This is done using a vision language model and has been described in greater depth in the additional features section.

|  |  |
| --- | --- |
| A black and white text on a black background  AI-generated content may be incorrect. | A screenshot of a graph  AI-generated content may be incorrect. |
| *Figure 49 – Judge vision language model explanation* | *Figure 50 – Judge vision language model validation* |

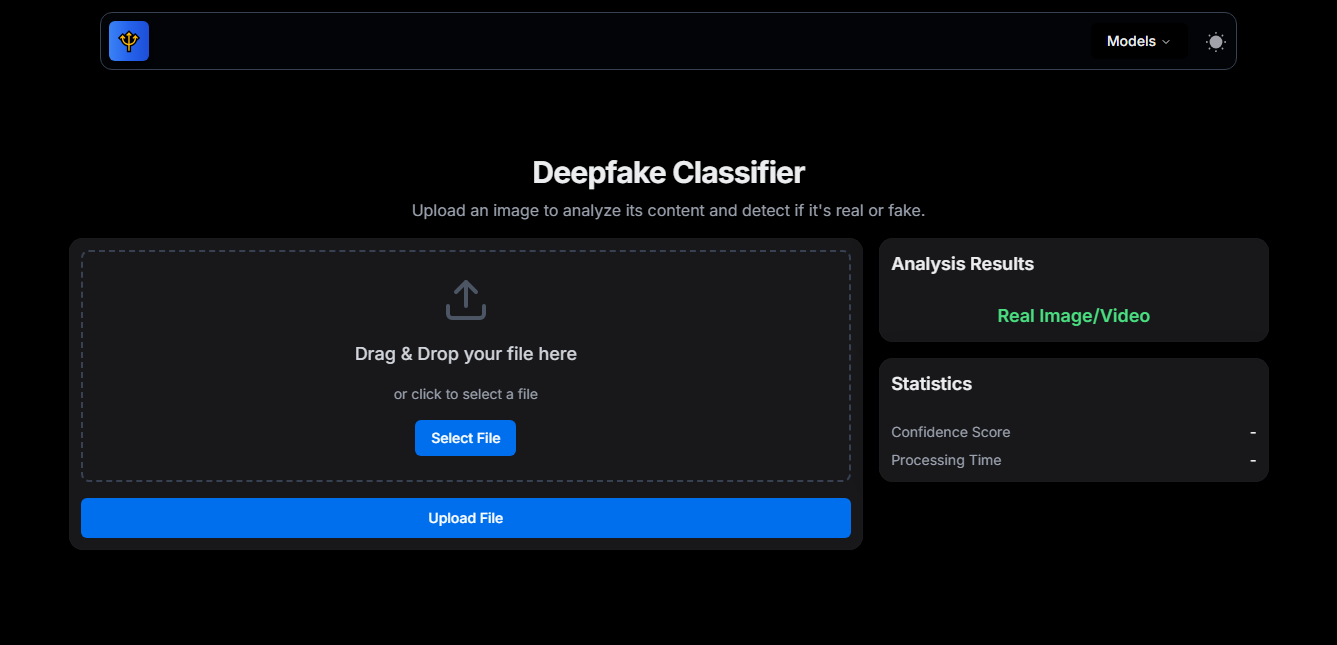
Additionally, since the model contains some biases and misclassifications, due to the training data set, which is used, it is essential to include a disclaimer on the website informing the users of this. This is to prevent misplaced reliance on AI outputs, mitigate risks of unintended harm from algorithmic biases, and ensure compliance with ethical AI governance frameworks.

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 51 – Disclaimer for service*

## **Deep Fake Detection**



*Figure 52 – Deepfake classifier service*

To explain exactly what the user is looking at:

* **Label:** This indicates whether an image is a deepfake. It is color coded with icons for more expressiveness to the user.
* **Confidence Score:** Shows the user how confident the model is in identifying whether the meme is harmful or not. The closer the number is to 1, the more likely the image is a deepfake.

The additional feature added is that googles API will automatically extract the face from the image and send it to the model for analysis.

|  |  |
| --- | --- |
|  |  |
| *Figure 53 – Result of real image* | *Figure 54 – Result of real image* |

A simple, clean UI ensures easy navigation and usability, making the system accessible to all users without unnecessary complexity.

As the model only processes images without storing them due to privacy reasons, it does not provide explanations for how it arrives at its conclusions. Because the input is purely an image, the model lacks interpretability beyond its prediction. The results should be taken as a suggestion rather than a definitive judgment, and users are encouraged to verify the outcome through additional means if needed.

## 

## **Fake News Detection**

A screenshot of a black screen

AI-generated content may be incorrect.

*Figure 55.1 – Result of fake news detection service*

This UI appears to be a **fake news detection interface** with a dark theme. It consists of three main sections:

1. **Text Input Section (Left Panel)**
   * A large text box where users can input a claim or news snippet.
   * A blue "Classify Text" button below the text box to submit the input for analysis.
2. **Classification Result (Top Right Panel)**
   * Displays the model's prediction (e.g., "Fake").
   * Shows a confidence score (e.g., 6.71%), indicating how certain the model is about its classification.
3. **Related Articles (Bottom Right Panel)**
   * A list of articles with their similarity scores, ranked by relevance.
   * Each article has a clickable blue title and a similarity percentage, indicating how closely related it is to the input text.

A black background with white text

AI-generated content may be incorrect.

*Figure 55.2 – Result of fake news detection service*

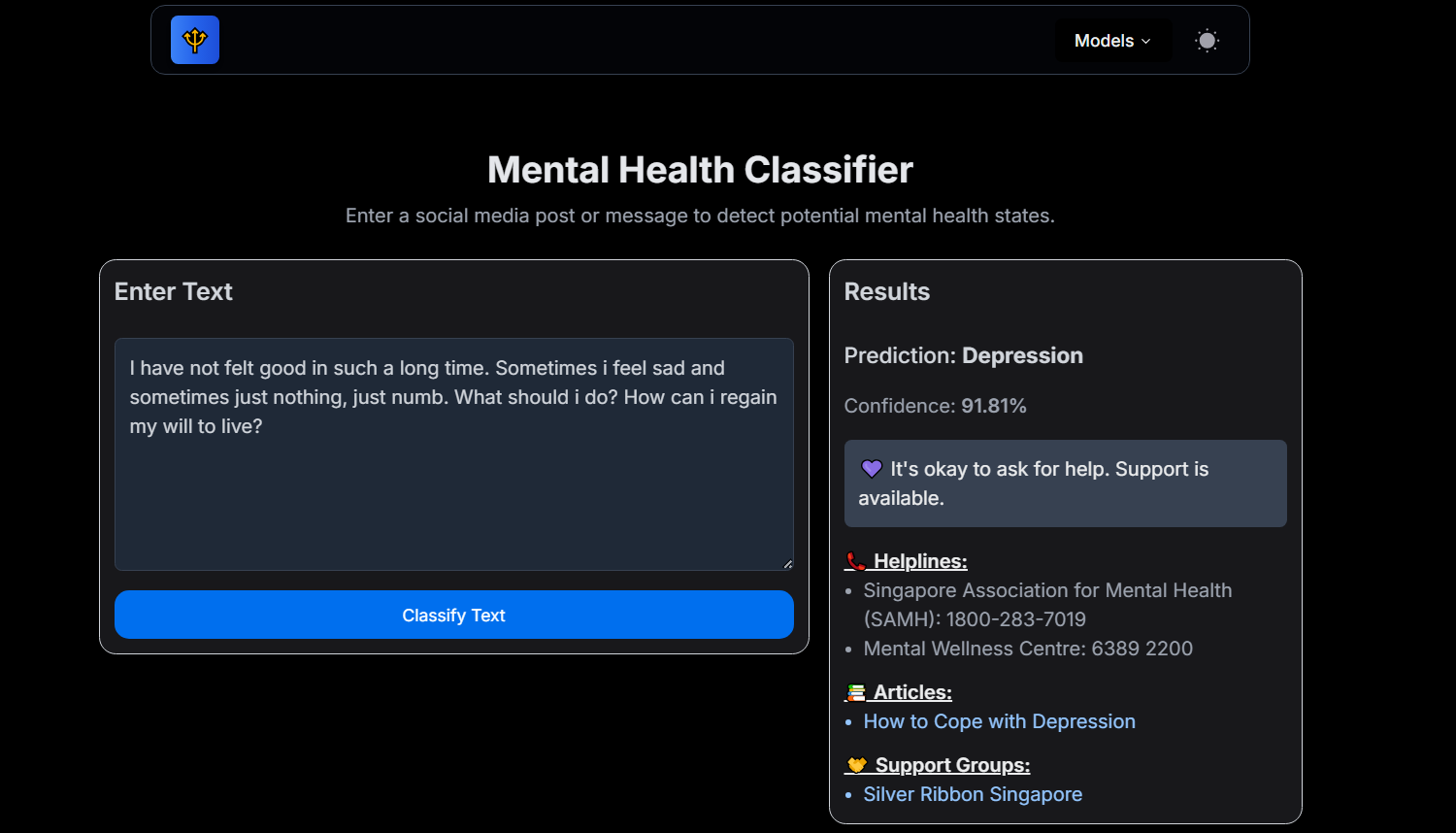
This UI appears to be a **fake news analysis results page**, displaying a **conclusion** based on the classification model’s output. It consists of two main sections:

1. **Summary Section**
   * Provides a brief overview of relevant information related to the analysed statement.
   * Mentions a Pakistani kheer vendor resembling Donald Trump and highlights other news items.
   * Appears to give context to the claim by listing related or trending news topics.
2. **Analysis Section**
   * Explains the model’s classification decision.
   * States that the model predicted the statement as **Fake News** but acknowledges the possibility of truth based on some evidence.
   * Discusses the credibility of sources, lack of independent corroboration, and the potential for exaggeration.
   * Concludes that further verification from reliable sources is needed before making a definitive judgment.

In conclusion, this UI enhances user experience by providing a clean, structured layout with clear explanations and relevant sources. Instead of just labeling content as fake or true, it offers an **Analysis** section that explains the reasoning behind the classification. The **Related Articles** section provides additional context, allowing users to verify claims independently. By combining transparency, clarity, and supporting evidence, the UI fosters trust and critical thinking, making the tool both informative and user-friendly.

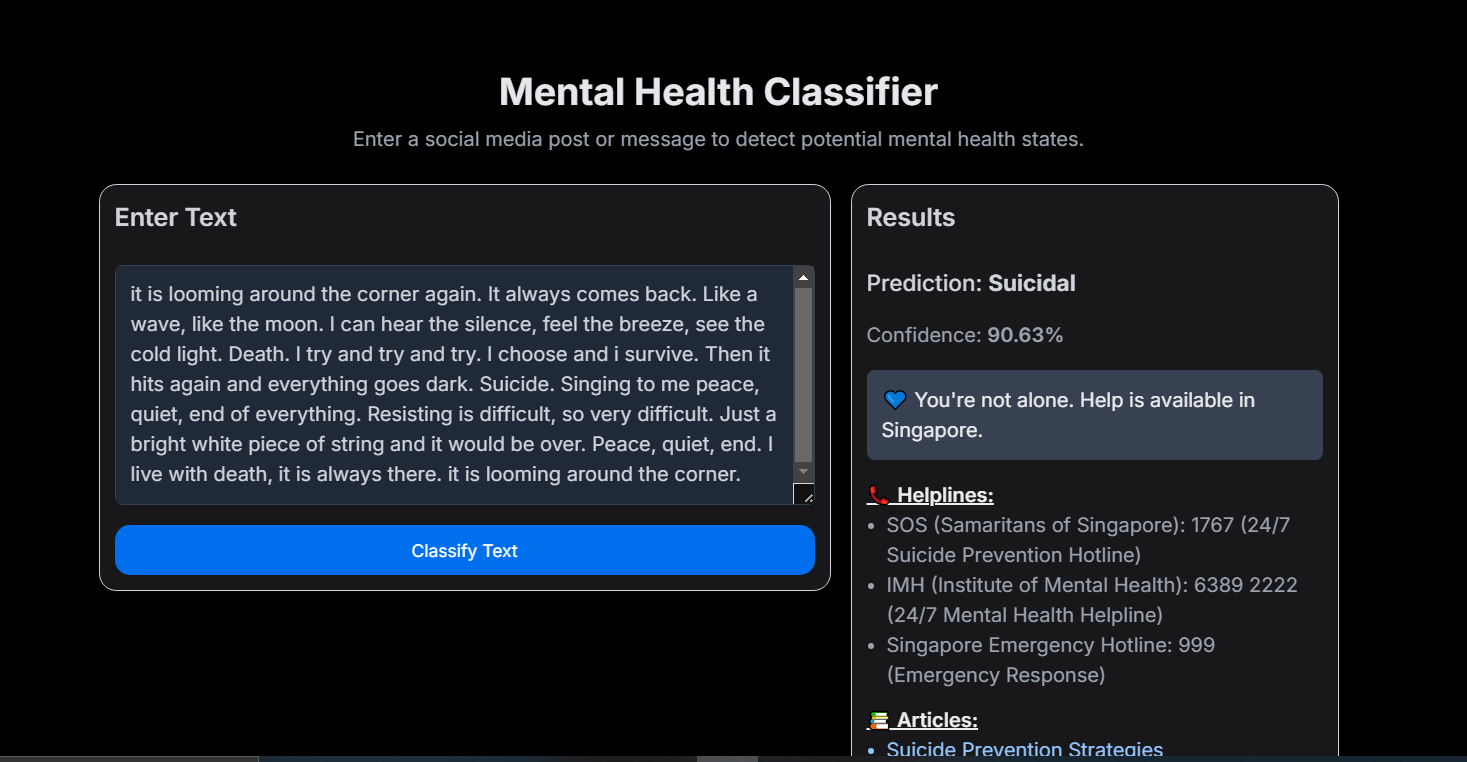
## **Mental Health Classifier**

The mental health classifier enables users to input text, which is then analyzed to determine and display a corresponding mental health status based on the content.



*Figure 56 – Result of mental health detection service (depression output)*

This is the response that would be displayed if the user's input is classified as "Depression."



*Figure 57 – Result of fake news detection service* *(suicidal output)*

This is the response that would be displayed if the user’s input is classified as “Suicidal”.

A total of **seven** possible mental health statuses can be displayed:

* **Anxiety**
* **Bipolar**
* **Depression**
* **Normal**
* **Personality Disorder**
* **Stress**
* **Suicidal**

In the second example, the classifier provides a prediction regarding the nature of the text. In this case, it predicts "Suicidal" with a confidence level of 90.63%. The confidence level indicates the algorithm's certainty about its prediction, which is relatively high in this instance. Upon detecting suicidal content, the classifier offers immediate support information, including helplines and emergency contacts. It lists specific helplines such as SOS (Samaritans of Singapore), IMH (Institute of Mental Health), and the Singapore Emergency Hotline.

The interface is designed to be user-friendly, presenting the classification results and support options in a clear and accessible manner. It reassures the user that they are not alone, and that help is available, which is crucial for individuals in distress.

Ultimately, users need to understand that the tool should be used as a supportive resource rather than a definitive diagnostic or decision-making tool.

# **Reflection**

## **Group Reflection**

* **What Went Well:** The project was executed collaboratively with effective communication, particularly through the use of GitHub, which allowed us to work efficiently and stay organized. Team members supported each other when challenges arose, particularly when we encountered issues with limited credits, ensuring that we met deadlines and maintained progress. Additionally, everyone actively contributed to the project, particularly in the creation of the slides, and each member pulled their weight, fostering a strong sense of teamwork.
* **Challenges Encountered:** One of the challenges faced during the project was managing the limited number of credits, which at times affected our ability to execute certain tasks as quickly as we would have liked. This led to some delays in certain stages of the project, though we were able to overcome this through collaboration and support. Overall, all the project milestones were submitted on time with no delays.
* **Opportunities for Improvement:** Looking ahead, we could improve by planning better for resource limitations, such as credits, to minimize delays. A more detailed contingency plan could help us avoid disruptions. Additionally, while communication was generally effective, allocating regular time slots for group discussions could enhance clarity and reduce misunderstandings. Lastly, better distribution of tasks in earlier stages could ensure a more even workload throughout the project. Furthermore, while the overall contribution from team members was strong, task allocation could be better optimized in the initial stages. A more strategic division of work, perhaps based on individual strengths and expertise, would help ensure that tasks are distributed in a way that maintains momentum and prevents bottlenecks.

## **Individual Reflection**

|  |  |
| --- | --- |
| Team Member | Reflection |
| Karthik | I’m particularly proud of the research I conducted on the problem at hand, as it allowed me to gain a deep understanding of the core challenges. I applied design thinking to identify the key pain points and tailored my solutions specifically to address them. Additionally, I was able to bring new ideas to the table, such as the "judgeVLM," where I incorporated concepts from the MLOps module and used my knowledge of generative AI to implement a unique feature. This allowed me to merge theoretical knowledge with practical application, which I found particularly fulfilling.  Additionally, other concepts and technologies I learned during this process was deployment to Google Cloud, CI/CD, docker and system design strategies, such as the comparison between monorepo and microservices architectures. Understanding these approaches has been valuable in creating scalable and efficient solutions. This also aligns with my future aspiration of becoming a machine learning engineer, where deployment is one of the key skillsets.  If given the opportunity to redo the project, I would prioritize collaborating with domain experts from the very beginning. Their insights would enhance the accuracy and contextual relevance of our deep learning models. |
| Wei Jun | In this deepfake classifier project, I gained hands-on experience in managing image data, converting it into machine-learnable formats, and optimizing batch sizes based on available VRAM. I’m particularly proud of deploying the model using microservices on Google Cloud Run, where I integrated Docker and CI/CD principles to automate testing and deployment. This was my first exposure to these technologies, and it was rewarding to apply my ML Ops knowledge. I also learned TypeScript for the frontend, and AI chatbots were key in boosting productivity throughout the process. Additionally, using Google API for face detection was an exciting part of the project, as I got to apply cutting-edge technology in real-world scenarios.  For improvement, I would add video support by extracting frames and performing face analysis, flagging the entire video as deepfake if more than 10% of the faces are flagged. Additionally, with more resources, I would implement transfer learning to optimize model performance and size. Finally, I’d introduce a feedback loop where users can submit mislabeled content, helping to finetune the model over time. Additionally, if given more time, I will try quantization on larger models to make the model smaller and be able to run faster while minimizing performance loss.  Overall, this project enhanced my technical skills, especially in cloud deployment, Docker, and CI/CD, while also giving me a deeper understanding of model management. Looking ahead, I see opportunities to expand the system’s capabilities and refine it through user feedback and advanced model optimization. |
| Jun Ming | One aspect I am particularly proud of is successfully applying and integrating concepts from various modules into this WIU. These include BERT embeddings, LSTM networks, and deep learning principles from the Applied Deep Learning module. These foundational concepts provided a strong basis for experimentation, allowing me to combine them with techniques I had previously learned, such as random search for hyperparameter tuning. This approach was especially useful in addressing the challenges of deep learning, such as long training times, and in selecting an efficient optimization strategy. Additionally, we incorporated AI explainability, a concept covered in the Applied Machine Learning module, to enhance the interpretability of our model’s decisions.  Another key concept I applied was API integration, which I learned from the AI Services in Analytics module. This was instrumental in implementing text generation as an additional feature, helping to generate meaningful conclusions based on detected news articles. Beyond model development, I also extended my learning to MLOps, where we utilized a CI/CD framework to continuously deploy updates via GitHub and Google Cloud. This not only streamlined our development process but also ensured our model remained scalable, adaptable, and production ready.  One area for improvement is the optimization of the backend. I realized that I incurred higher-than-expected computational costs when deploying the backend, likely due to unnecessary package installations and storage-heavy dependencies. Additionally, I downloaded a sentence transformer model to compute cosine similarity scores, which further increased resource usage. Given more time, I would refine the code by exploring alternative solutions to optimize efficiency and reduce costs without compromising performance. |
| Pin Shien | Working on this project was a deeply insightful experience that blended technical learning with ethical reflection. I developed valuable skills in natural language processing, from preprocessing text data to evaluating model performance using key metrics like precision and recall. Beyond the technical aspects, I gained a deeper understanding of how language patterns and behaviors can serve as indicators of mental health. This reinforced the importance of handling sensitive data with both accuracy and empathy, ensuring that technological solutions are not only effective but also responsible.    To further enhance the solution, integrating a hybrid approach that combines rule-based sentiment analysis with machine learning models could improve both interpretability and accuracy. While deep learning models offer high performance, incorporating linguistics-based rules can help mitigate biases and provide more transparent reasoning behind predictions.  Additionally, enforcing a human-in-the-loop system—where mental health professionals can review and validate flagged cases—could enhance the tool’s reliability while preventing misdiagnoses. This collaborative approach would ensure that AI serves as an assistive tool rather than an autonomous decision-maker, maintaining ethical integrity and trustworthiness. |

# **Appendix**

1. <https://arxiv.org/pdf/2409.14703>