Machine Learning Project: Suicide Detection

Ivan Gorbachev / איוון גורבצ’וב / 326960333
לידור בורופקר / lidor borofker / 326065646

Dataset:
Suicide and Depression Detection

Code:
Google Colab
GitHub

Dataset overview:
The dataset is a collection of text posts collected from the subreddits r/SuicideWatch (Dec 16, 2008 - Jan 2, 2021), and r/teenagers. All posts from r/SuicideWatch are labeled as “suicide” while all posts from r/teenagers are labeled as “non-suicide”.

The dataset contains a total of 232,074 posts split equally between the two classes (116,037 each), giving us a perfectly balanced dataset.

The dataset was originally published alongside the paper Suicide Ideation Detection in Social Media Forums

Dataset features:

# (unused) - the index of the entry
text (input) - the textual contents of the post
class (target) - suicide / non-suicide

The aim of our work:
We aim to solve a binary classification problem: determining whether a post indicates suicidal ideation or not, based on its textual content. In addition to using the text directly, we also aim to extract supplementary features from the text, such as inferred age and gender, which we hypothesize may exhibit meaningful patterns associated with suicide risk which can be used in conjunction with the text when making our prediction.

We test four different classification algorithms: Support Vector Machines (SVM), Logistic Regression, Random Forest, and Fully Connected Neural Network (FCNN)) along with several variations and enhancements, such as adjusted decision thresholds, different text embedding techniques (TF-IDF and USE), and the aforementioned supplementary features. All models are evaluated on a 100,000 sample subset (For the sake of performance) using an 80/20 train-test split and a fixed random seed (42) to ensure consistency and reproducibility.

Performance Evaluation Metrics:

To evaluate the performance of our models, we focus on the following key metrics:

- Recall - Measures the model's ability to correctly identify true positives (suicidal posts). This is the most critical metric in our context, as failing to detect a suicidal post (a false negative) could have serious consequences.

- Precision - Measures the proportion of positive identifications (posts classified as suicidal) that are actually correct. High precision ensures that we are not misclassifying too many non-suicidal posts as suicidal.

- F2 Score - A weighted harmonic mean of precision and recall, with more emphasis on recall. It is especially useful when false negatives are more costly than false positives, which fits our case. The F2 score gives us a better picture of the model's performance in life-critical situations like suicide prevention.

- Accuracy - The proportion of total correct predictions. While accuracy is commonly used, it is less informative in imbalanced or high-risk classification tasks like this one.

In suicide detection, the cost of a false negative (misclassifying a suicidal post as non-suicidal) is far greater than that of a false positive (flagging a non-suicidal post as at-risk). Failing to identify someone in crisis could delay life-saving interventions, while false alarms may strain mental-health resources unnecessarily.

To address this, we prioritize recall (sensitivity) to maximize the detection of true suicidal posts. However, an overly sensitive model with low precision could overwhelm support systems with false alerts, diverting attention from genuine cases

To balance these concerns, we optimize the F2 score, a weighted harmonic mean of precision and recall that emphasizes recall twice as heavily as precision ($\beta=2$). This metric aligns with our goal:

- High recall: Minimize missed suicidal posts (false negatives).

- Reasonable precision: Limit false positives to avoid resource mismanagement.

By tuning the classification threshold (Lowering it to 0.3 from 0.5 in our case), we can further bias the model toward recall while monitoring precision to ensure it remains actionable. This approach reflects real-world priorities: catching every possible crisis while maintaining a practically usable system.

<u>Questions About dataset:</u>

As part of our analysis, we aim to explore the following questions:

- Can we extract supplementary features, such as age and gender, directly from the text in a reliable way?

- Do certain age groups or genders appear more frequently in suicidal posts compared to non-suicidal ones?

- Are there linguistic patterns or writing styles that correlate with suicidal ideation?

- Can these supplementary features, when combined with the text data, enhance the performance of our classification models?

- What is the nature of our dataset? Is our data linearly separable?

By investigating these questions, we hope to better understand not only how to classify suicidal posts, but also the underlying behavioral or demographic patterns that might help us make better informed classifications.

<u>Reflection On The Dataset:</u>

While the dataset used in the project is limited to Reddit posts, we believe it offers a strong foundation for training suicide ideation detection models. Unlike other platforms such as X (Formally Twitter) or Facebook where labeling suicidal intent entails extensive manual validation, since it requires verifying that the person behind the post actually has suicidal ideation. Reddit offers structured communities such as r/SuicideWatch and r/teenagers that naturally separate suicidal and non-suicidal content.

Although not every post on r/SuicideWatch guarantees clinical suicidal ideation, the subreddit is a relatively reliable proxy due to its support-focused nature and the sensitive topics discussed. Similarly, r/teenagers offers a consistent stream of casual, non-suicidal content.

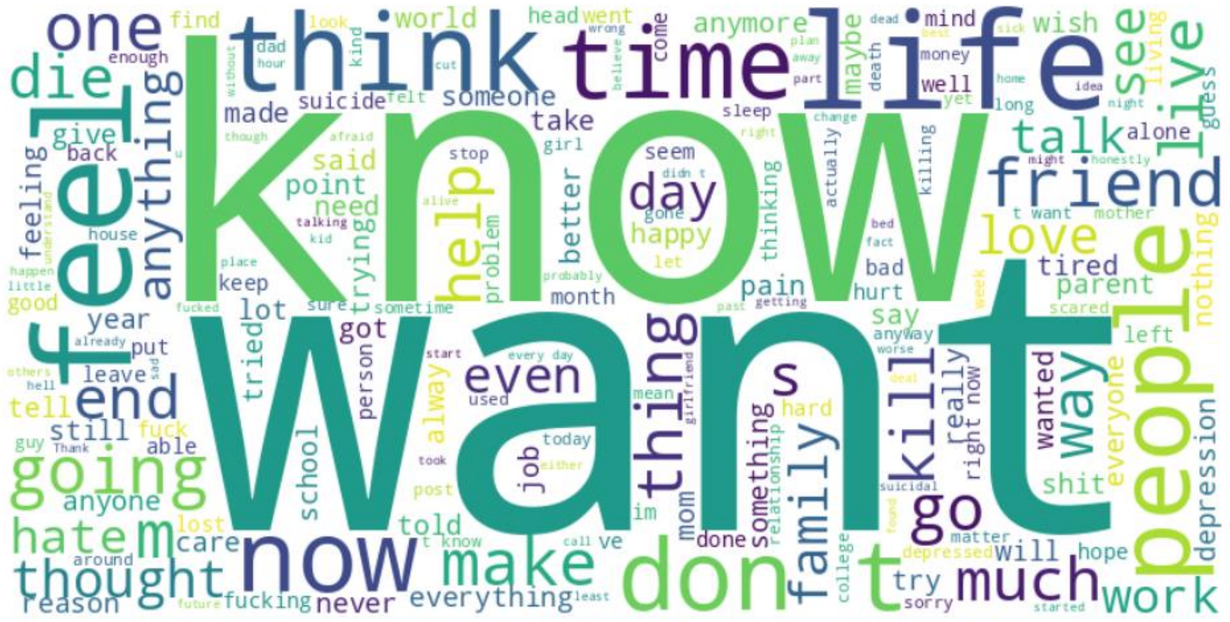
While the language used on Reddit is specific to its user base and internet culture, which limits generalization, this dataset is still more credible than alternatives without verified context.

In an ideal scenario, a platform like Facebook with its more personal and identity-linked posts might yield more verifiable labels, but data access and ethical concerns make that route far more difficult. In practice, Reddit may be one of the best openly available sources for this task.

<u>Dataset Analysis:</u>

Word Clouds:

Word Cloud - Suicide Posts

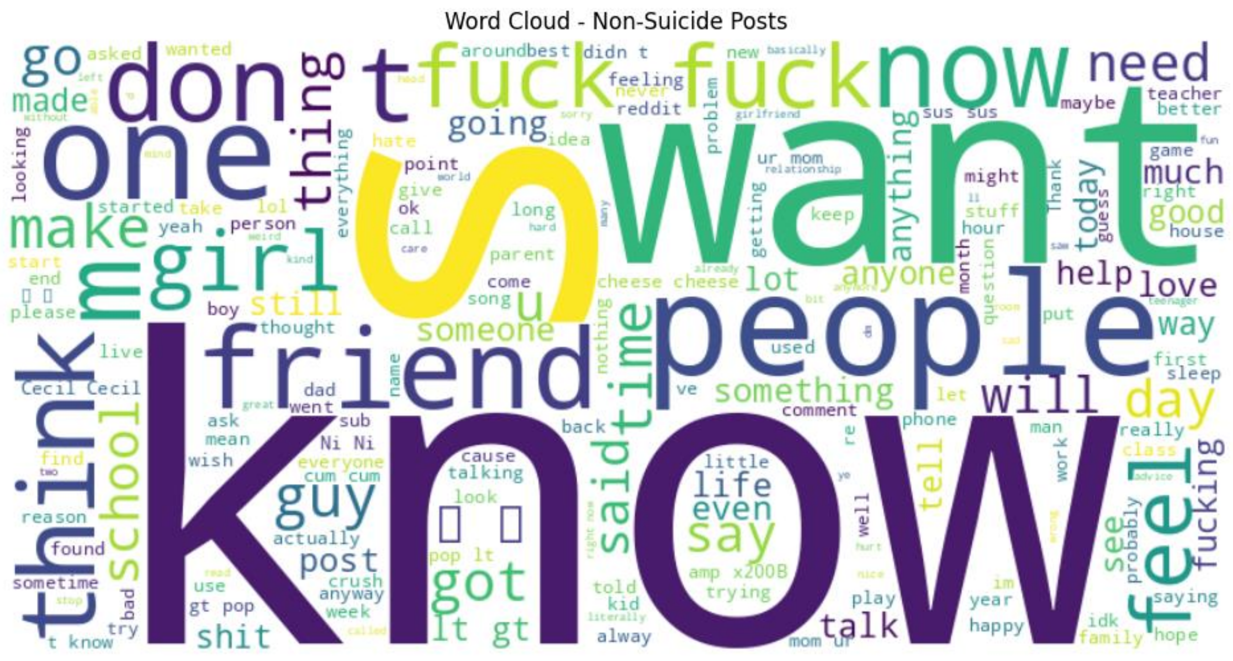


As we can see, the language used in suicide posts is emotionally charged and deals with existential struggles.

Words like “kill”, “die”, “suicide”, “depression”, “pain”, “hurt”, “lost”, “alone”, “leave”, “help”, “tired”, and “bad” are frequently used, with some directly referencing suicide ideation.

The use of words such as “life”, “school”, “job”, ”college”, “work”, “family”, “friend”, “girlfriend”, “money”, “parent”, and “future” point to stressors that posters experience which may be the cause of their suicide ideation.

Most posts are written in the first person as evidenced by the use of words like “know”, “want”, “feel”, “go”, “thought”, “don’t”, “I’m”, “try”, and “think”, and display strong emotions through the use of words like “love”, “hate”, “happy”, “hurt”, “need”, “sad”, “pain”. This suggests that the posters are reflecting on their life in a process of introspection and experiencing strong emotions in doing so.

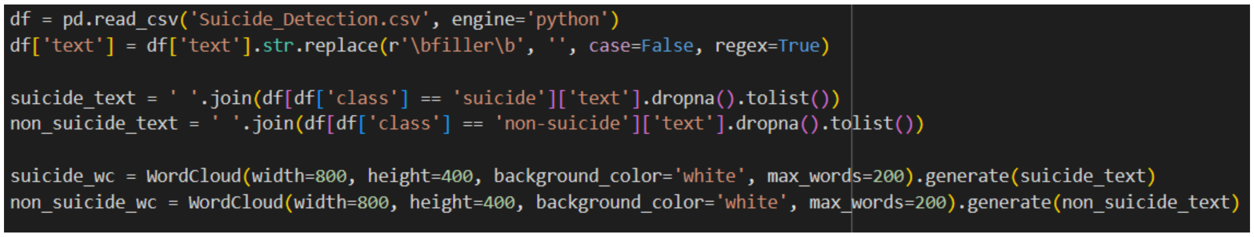


On the opposite end, the language used in non-suicide posts is much more relaxed, unserious, and imbued with in-jokes, expletives and slang.

“ur mom”, “pop it”, “cheese”, “lol”, “Cecil”, “sus”, and common expletives are all regular components of average good spirited teenage banter.

Words such as “teacher”, “school”, “dad”, “crush”, and “family” are representative of the common teenager experience.

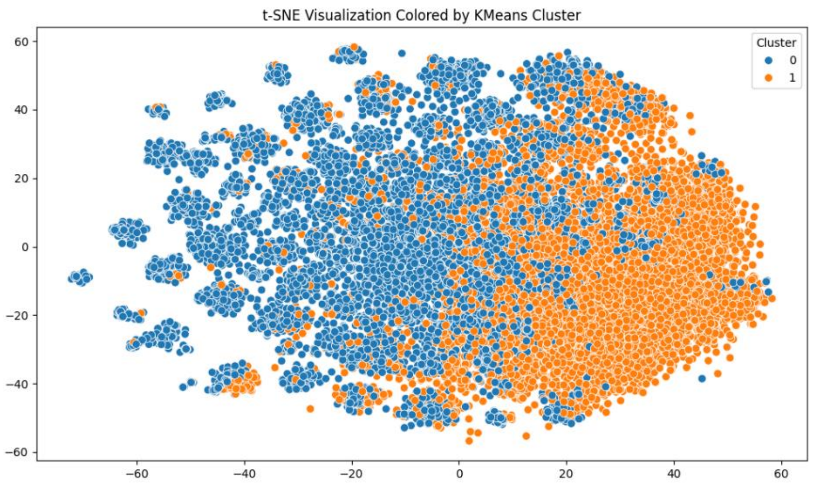
Overall, we can see a sharp contrast between the language of suicide posts and the language of non-suicide posts. This highlights the existence of a clear distinction between these types of posts which supports our ability to differentiate between them using language based features. We also observed that words like “know” and “want” are common in both suicidal and non-suicidal posts. However, we chose not to filter them out during preprocessing, as they are often used in emotional and introspective language, and may carry important semantic weight.

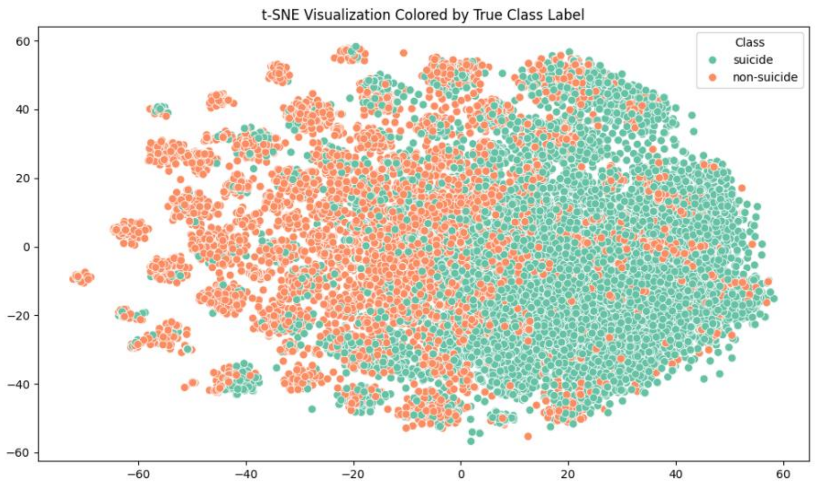


```
df = pd.read_csv('Suicide_Detection.csv', engine='python')
df['text'] = df['text'].str.replace(r'\bfiller\b', '', case=False, regex=True)

suicide_text = ' '.join(df[df['class'] == 'suicide']['text'].dropna().tolist())
non_suicide_text = ' '.join(df[df['class'] == 'non-suicide']['text'].dropna().tolist())

suicide_wc = WordCloud(width=800, height=400, background_color='white', max_words=200).generate(suicide_text)
non_suicide_wc = WordCloud(width=800, height=400, background_color='white', max_words=200).generate(non_suicide_text)
```

We generated these word clouds using wordcloud. The word “filler” was removed because it appeared artificially and frequently in non-suicide posts, likely as a placeholder or artifact introduced during dataset creation, and did not carry meaningful information.

KMeans Analysis TF-IDF:





Cluster ID/ Label	**Non-Suicide**	**Suicide**
0	**0.736252**	**0.263748**
1	**0.111247**	**0.888753**

Cluster 0 samples:

“Am I weird I don't get affected by compliments if it's coming from someone I know irl but I feel really good when internet strangers do it” [non-suicide]

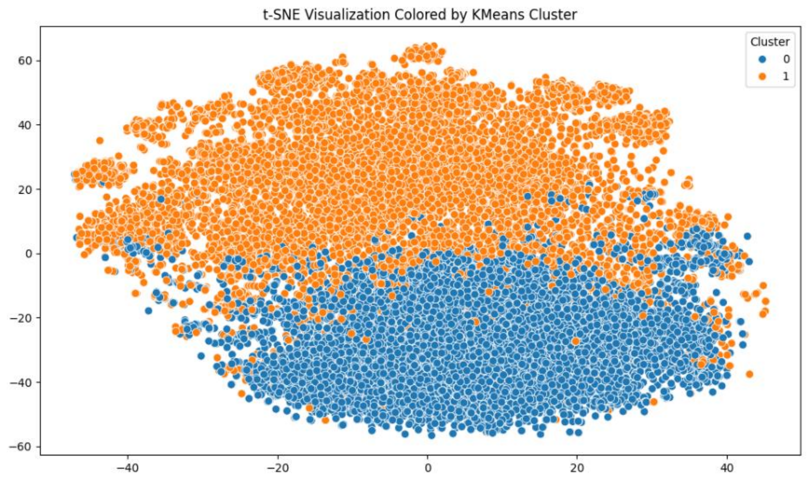
Cluster 1 samples:

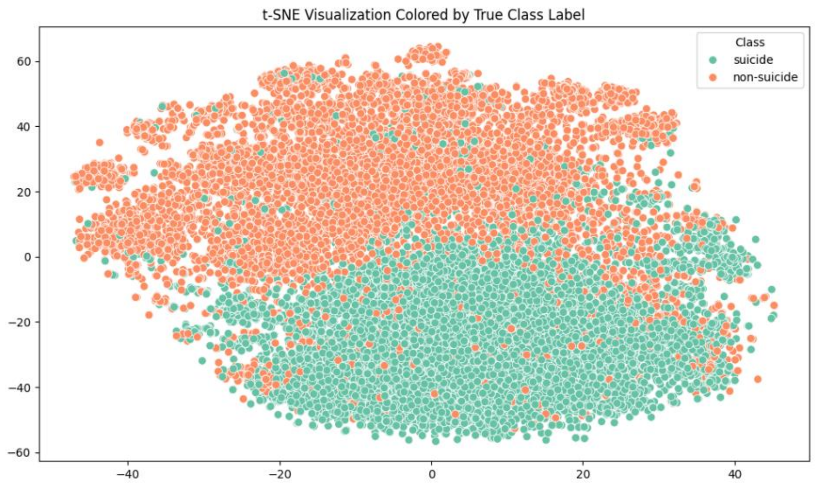
“I’m so lostHello, my name is Adam (16) and I’ve been struggling for years and I’m afraid. Through these past years...” (Very long post) [suicide]

Clusters formed using TF-IDF vectorization can be differentiated by the length, themes, language, and formatting of the posts. TF-IDF creates sparse high-dimensional vectors, where the magnitude and distance between samples are influenced by word frequency and document length because longer texts tend to introduce more unique n-grams. As a result, longer, more articulate posts with detailed introspection which are common in suicide-related content, often cluster together due to their distinct linguistic richness. In contrast, shorter and rougher posts, more aligned with teenage banter, form a separate group. Although TF-IDF does not capture semantic meaning, it proves effective at separating texts based on structure and vocabulary usage patterns alone.

The relatively high purity observed in both clusters suggests that the data is linearly separable to a meaningful extent, further supported by the strong performance of linear models like SVM. Additionally, t-SNE visualizations show a clear separation of labels, with each class forming distinct regions in the reduced space.

KMeans Analysis USE:





Cluster ID/ Label	**Non-Suicide**	**Suicide**
0	**0.085705**	**0.914295**
1	**0.878708**	**0.121292**

Cluster 0 samples:

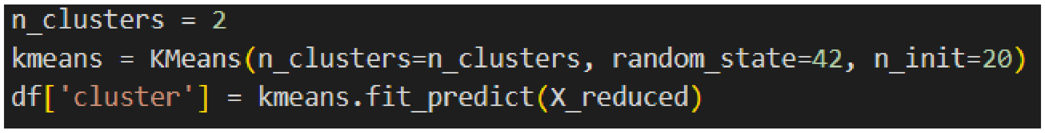
“My life is over at 20 years oldHello all. I am a 20 year old balding male. My hairline is trash...” [suicide]

Cluster 1 sample:

“Everyone wants to be "edgy" and it's making me self conscious I feel like I don't stand out..” (Very long post) [non-suicide]

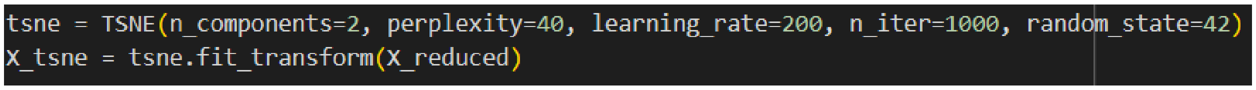
Clusters formed using Universal Sentence Encoder (USE) show a clear semantic separation between suicidal and non-suicidal posts. Cluster 0 contains over 91% suicide-labeled posts, while cluster 1 is made up of nearly 88% non-suicidal content. Unlike TF-IDF, USE captures the semantic meaning of entire sentences, allowing it to group together posts with similar emotional and contextual tone, even if they don't share many common words. The result is a more conceptually driven clustering, where introspective, emotionally charged, and self-reflective content gravitates toward one cluster, while lighthearted or off-topic posts are grouped into the other. This makes USE particularly powerful for nuanced psychological text analysis like suicide ideation detection.

The relatively high purity observed in both clusters suggests that the data is linearly separable to a meaningful extent, further supported by the strong performance of linear models like SVM. Additionally, t-SNE visualizations show a clear separation of labels, with each class forming distinct regions in the reduced space.



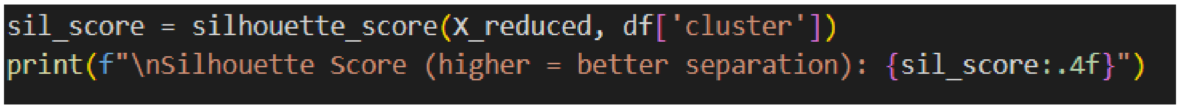
```python
n_clusters = 2
kmeans = KMeans(n_clusters=n_clusters, random_state=42, n_init=20)
df['cluster'] = kmeans.fit_predict(X_reduced)
```

For clustering analysis, we applied KMeans with two clusters (n_clusters=2) to explore natural groupings in the data.



```python
tsne = TSNE(n_components=2, perplexity=40, learning_rate=200, n_iter=1000, random_state=42)
X_tsne = tsne.fit_transform(X_reduced)
```

to inspect how well the vectorized representations separate suicidal from non-suicidal posts in a lower-dimensional space we used t-SNE with two output dimensions, a perplexity of 40, and a learning rate of 200.

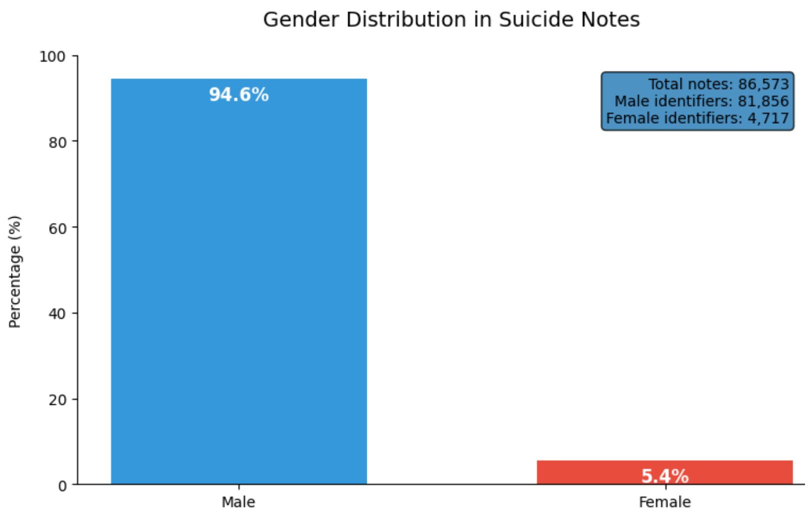


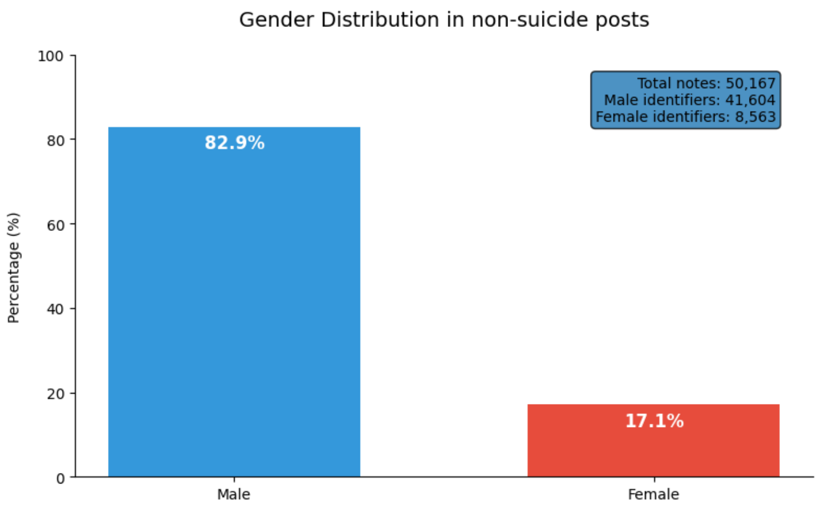
```python
sil_score = silhouette_score(X_reduced, df['cluster'])
print(f"\nSilhouette Score (higher = better separation): {sil_score:.4f}")
```

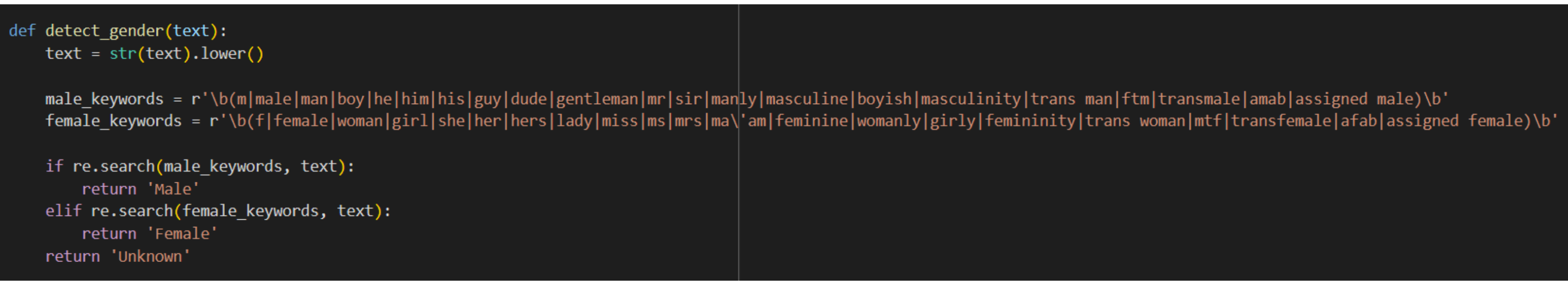
We also calculated the Silhouette Score to evaluate clustering quality. Both TF-IDF and USE embeddings yielded positive but low scores (0.0621 and 0.0995, respectively), suggesting partial separation between suicidal and non-suicidal posts. While the clusters are not strongly distinct, USE's slightly higher score further supports its ability to capture underlying semantic differences.

As an exploratory step, we briefly experimented with using KMeans as a classifier by assigning cluster labels as predictions, but this approach did not yield significant results or improvements compared to supervised models.

Gender distribution:







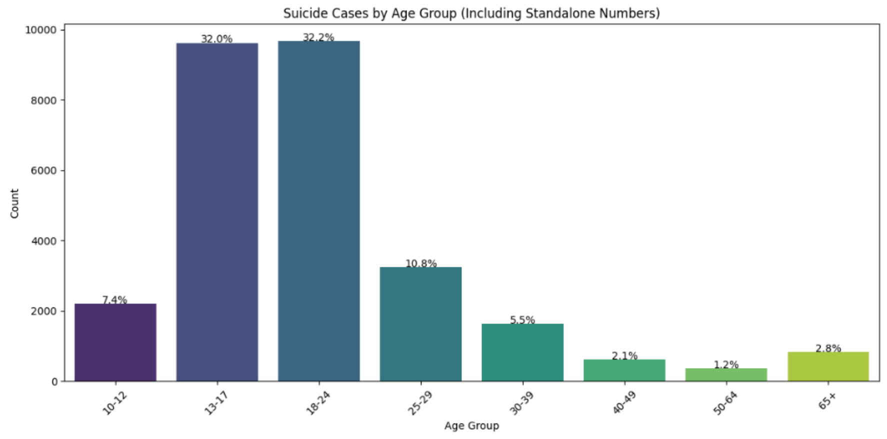
```python
def detect_gender(text):
    text = str(text).lower()

    male_keywords = r'\b(m|male|man|boy|he|him|his|guy|dude|gentleman|mr|sir|manly|masculine|boyish|masculinity|trans man|ftm|transmale|amab|assigned male)\b'
    female_keywords = r'\b(f|female|woman|girl|she|her|hers|lady|miss|ms|mrs|ma\'am|feminine|womanly|girly|femininity|trans woman|mtf|transfemale|afab|assigned female)\b'

    if re.search(male_keywords, text):
        return 'Male'
    elif re.search(female_keywords, text):
        return 'Female'
    return 'Unknown'
```

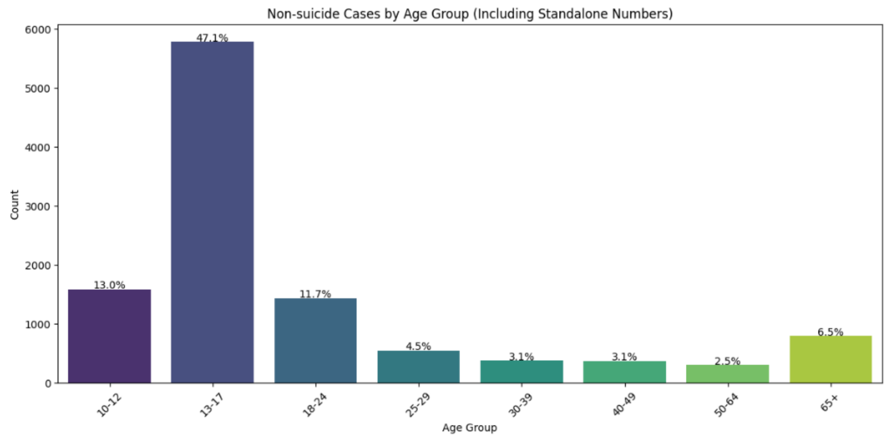
We used rule-based pattern matching to infer the gender of the poster by scanning each post for keywords and descriptors such as “m”, “f”, “male”, “woman”, and related pronouns. The matching was case-insensitive, and posts without identifiable gender indicators were labeled as "Unknown" and excluded from gender-based analysis. Interestingly, we found that a larger proportion of suicide posts contained identifiable gender information compared to non-suicide posts. This is likely due to the introspective and personal nature of suicide-related content, where individuals often reflect on their identity, life story, personal struggles, and sometimes explicitly mentioning their gender as part of their narrative. In contrast, non-suicide posts were typically shorter, casual, and less self-descriptive.

Although both classes showed more male posters overall, it’s likely due to Reddit’s general male dominated user base.

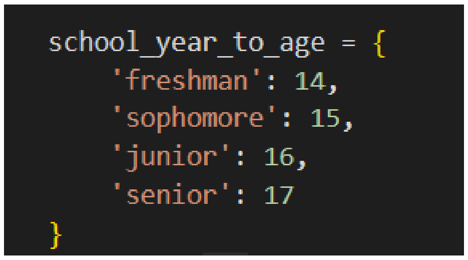
The overrepresentation of gender in suicide posts made it a useful auxiliary feature when combined with textual data. Gender alone did not yield strong predictive performance, but when fused with language-based features, it noticeably improved classification results.

Age Distribution:

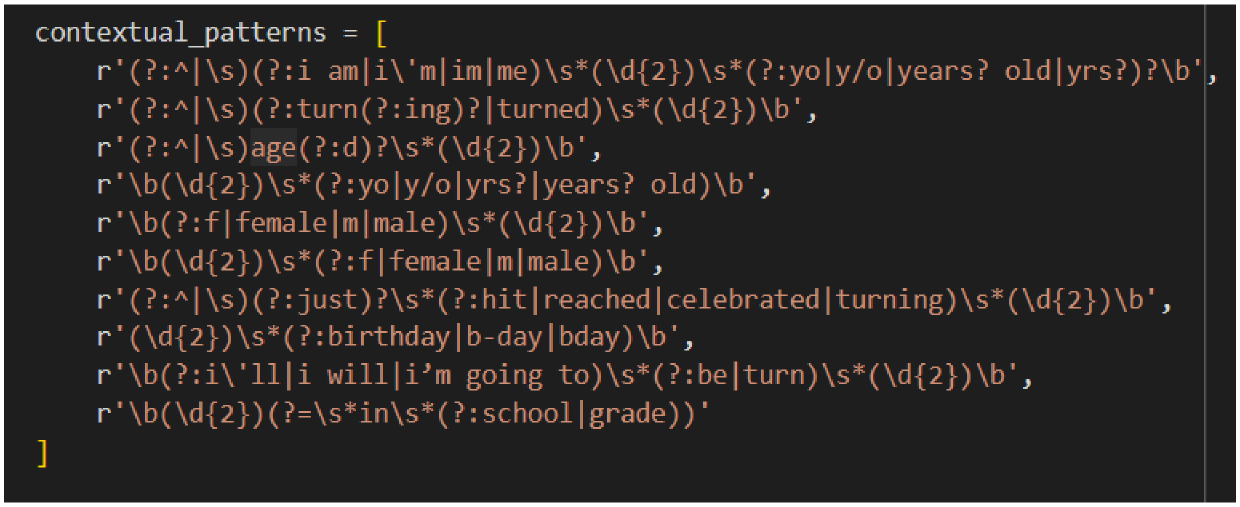




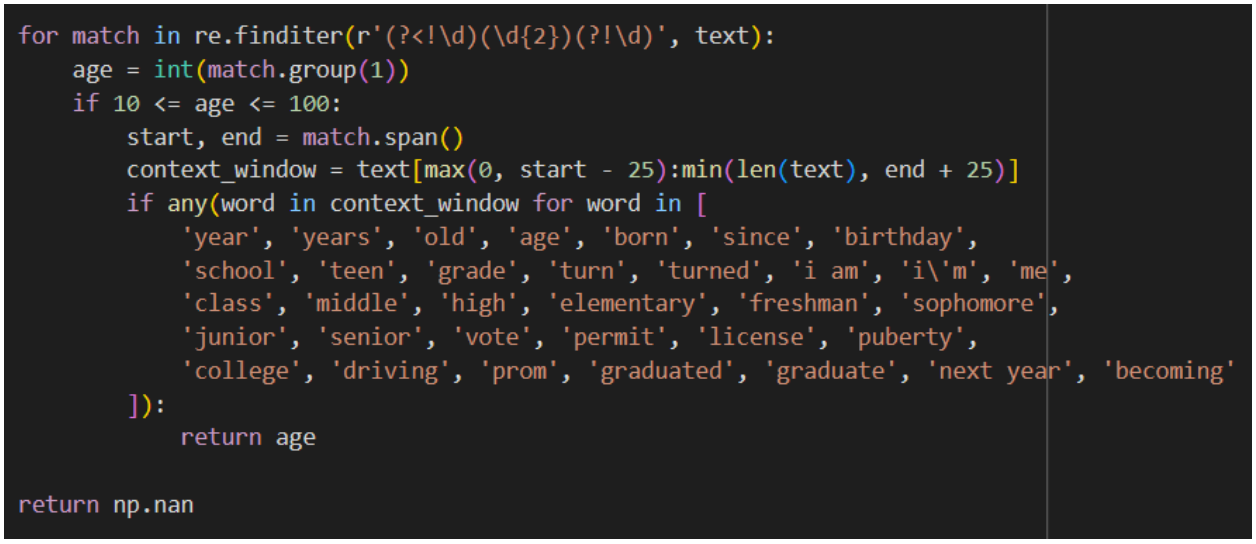
We used rule-based pattern matching to infer the age of the poster similarly to how we inferred the gender of the poster. While age distribution among suicide posts roughly aligned with general Reddit userbase demographics, the age distribution in non-suicide posts was overwhelmingly skewed toward teenagers. However, this is expected given the fact that the non-suicide posts were sourced from r/teenagers. As such, age doesn't serve as a meaningful predictor of suicidal ideation as it more so reflects the posts rather than any useful signs of suicidal ideation. Thus, we elected to exclude age as a feature in our classification.



```
school_year_to_age = {
    'freshman': 14,
    'sophomore': 15,
    'junior': 16,
    'senior': 17
}
```



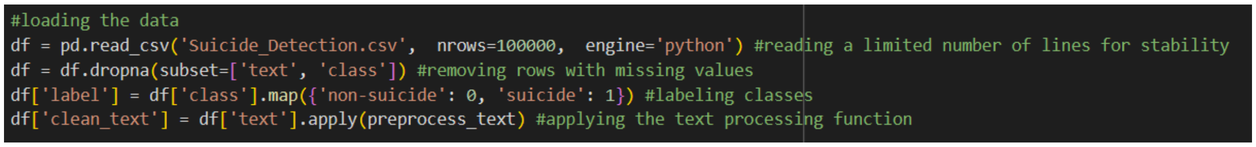
```
contextual_patterns = [
    r'(?:^|\s)(?:i am|i\'m|im|me)\s*(\d{2})\s*(?:yo|y/o|years? old|yrs?)?\b',
    r'(?:^|\s)(?:turn(?:ing)?|turned)\s*(\d{2})\b',
    r'(?:^|\s)age(?:d)?\s*(\d{2})\b',
    r'\b(\d{2})\s*(?:yo|y/o|yrs?|years? old)\b',
    r'\b(?:f|female|m|male)\s*(\d{2})\b',
    r'\b(\d{2})\s*(?:f|female|m|male)\b',
    r'(?:^|\s)(?:just)?\s*(?:hit|reached|celebrated|turning)\s*(\d{2})\b',
    r'(\d{2})\s*(?:birthday|b-day|bday)\b',
    r'\b(?:i\'ll|i will|i’m going to)\s*(?:be|turn)\s*(\d{2})\b',
    r'\b(\d{2})(?=\s*in\s*(?:school|grade))'
]
```



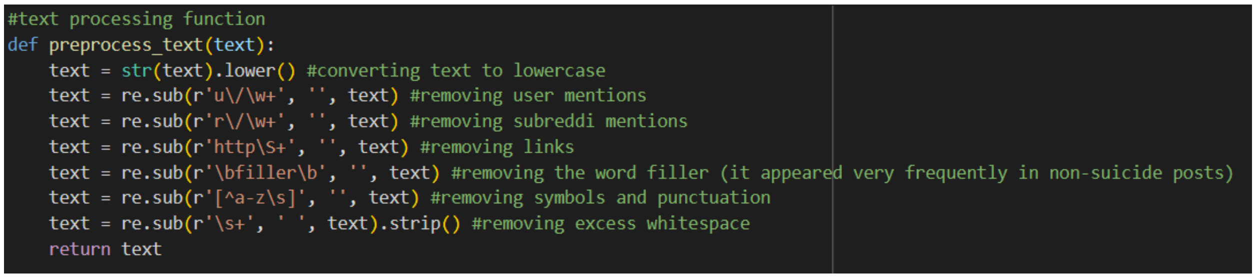
```
for match in re.finditer(r'(?<!\d)(\d{2})(?!\d)', text):
    age = int(match.group(1))
    if 10 <= age <= 100:
        start, end = match.span()
        context_window = text[max(0, start - 25):min(len(text), end + 25)]
        if any(word in context_window for word in [
            'year', 'years', 'old', 'age', 'born', 'since', 'birthday',
            'school', 'teen', 'grade', 'turn', 'turned', 'i am', 'i\'m', 'me',
            'class', 'middle', 'high', 'elementary', 'freshman', 'sophomore',
            'junior', 'senior', 'vote', 'permit', 'license', 'puberty',
            'college', 'driving', 'prom', 'graduated', 'graduate', 'next year', 'becoming'
        ]):
            return age

return np.nan
```

To extract age from posts, we developed a rule-based extract_age function that uses keyword heuristics and regular expressions to identify numerical age mentions. The method first maps common U.S. school year terms (freshman, senior) to approximate ages. It then searches for explicit age expressions using patterns like “I’m 17” or “turning 18.” If no match is found, the function looks for standalone two-digit numbers and checks for surrounding age-related context words such as birthday, teen, or school to reduce false positives. This layered approach balances precision and recall in noisy, informal text data.

Data Loading & Preprocessing

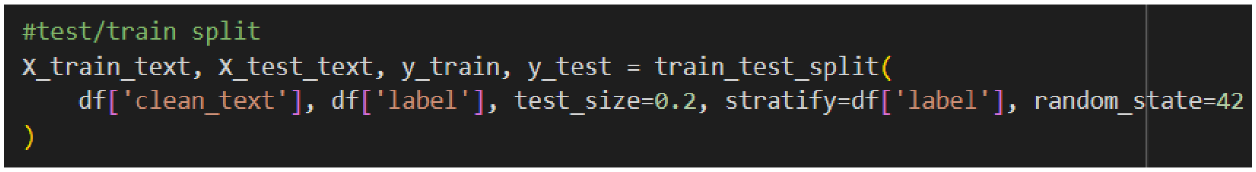


```
#loading the data
df = pd.read_csv('Suicide_Detection.csv',  nrows=100000,  engine='python') #reading a limited number of lines for stability
df = df.dropna(subset=['text', 'class']) #removing rows with missing values
df['label'] = df['class'].map({'non-suicide': 0, 'suicide': 1}) #labeling classes
df['clean_text'] = df['text'].apply(preprocess_text) #applying the text processing function
```



```
#text processing function
def preprocess_text(text):
    text = str(text).lower() #converting text to lowercase
    text = re.sub(r'u\/\w+', '', text) #removing user mentions
    text = re.sub(r'r\/\w+', '', text) #removing subreddi mentions
    text = re.sub(r'http\S+', '', text) #removing links
    text = re.sub(r'\bfiller\b', '', text) #removing the word filler (it appeared very frequently in non-suicide posts)
    text = re.sub(r'[^a-z\s]', '', text) #removing symbols and punctuation
    text = re.sub(r'\s+', ' ', text).strip() #removing excess whitespace
    return text
```

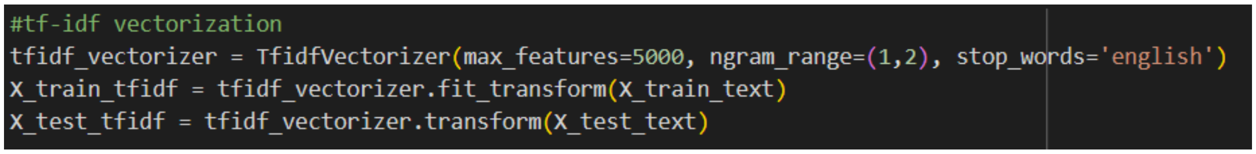
We began by loading the Reddit dataset, removing entries with missing values, mapping our class labels (non-suicide=0, suicide=1), and applying a rule-based text preprocessing function that standardized the data by converting all text to lowercase, removing user and subreddit mentions (u/ or r/), hyperlinks, and irrelevant tokens such as the word “filler.” We also stripped away non-alphabetic characters and normalized white spaces. This step is crucial to eliminate noise and ensure that the downstream vectorizers focus on meaningful linguistic content rather than artifacts or formatting inconsistencies.



```
#test/train split
X_train_text, X_test_text, y_train, y_test = train_test_split(
    df['clean_text'], df['label'], test_size=0.2, stratify=df['label'], random_state=42
)
```

Finally, we split the data into test/train sets at a constant random state to ensure consistent performance throughout all algorithm runs.

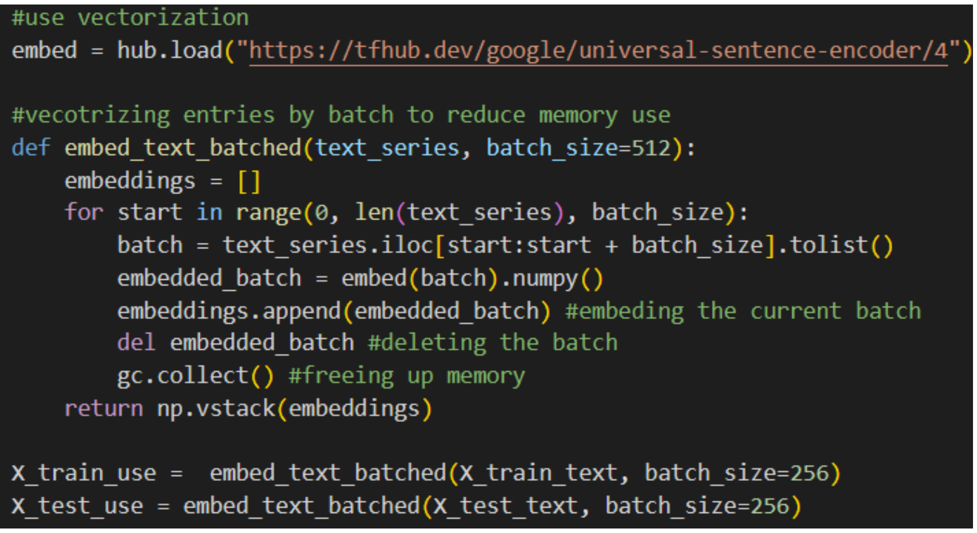
TF-IDF vectorizer



```
#tf-idf vectorization
tfidf_vectorizer = TfidfVectorizer(max_features=5000, ngram_range=(1,2), stop_words='english')
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train_text)
X_test_tfidf = tfidf_vectorizer.transform(X_test_text)
```

The Term Frequency-Inverse Document Frequency (TF-IDF) vectorizer converts the preprocessed text into a sparse numerical matrix that reflects the importance of each word or phrase across the dataset. This approach balances dimensionality and expressiveness by emphasizing words that are distinctive to each post while down-weighting common, less informative terms. The resulting vectors are suitable for linear classifiers such as SVMs.

USE vectorizer



```
#use vectorization
embed = hub.load("https://tfhub.dev/google/universal-sentence-encoder/4")

#vecotrizing entries by batch to reduce memory use
def embed_text_batched(text_series, batch_size=512):
    embeddings = []
    for start in range(0, len(text_series), batch_size):
        batch = text_series.iloc[start:start + batch_size].tolist()
        embedded_batch = embed(batch).numpy()
        embeddings.append(embedded_batch) #embeding the current batch
        del embedded_batch #deleting the batch
        gc.collect() #freeing up memory
    return np.vstack(embeddings)

X_train_use =  embed_text_batched(X_train_text, batch_size=256)
X_test_use = embed_text_batched(X_test_text, batch_size=256)
```

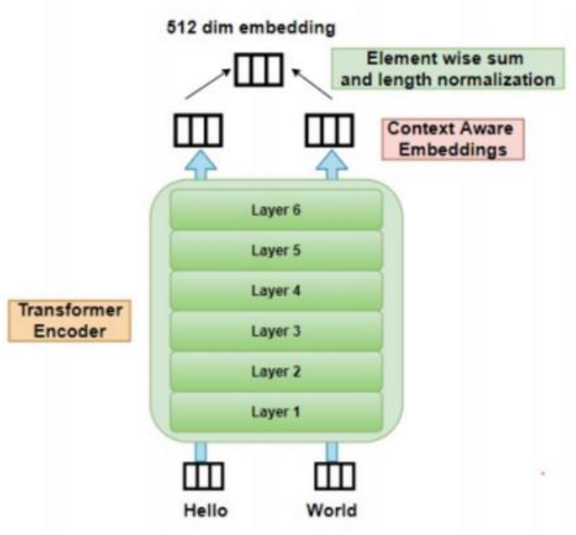
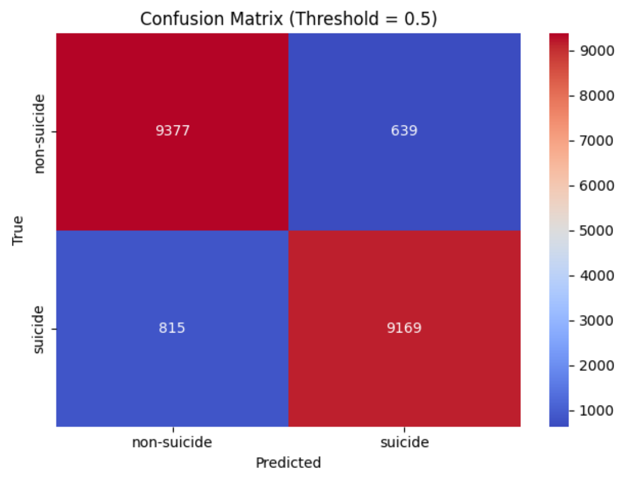


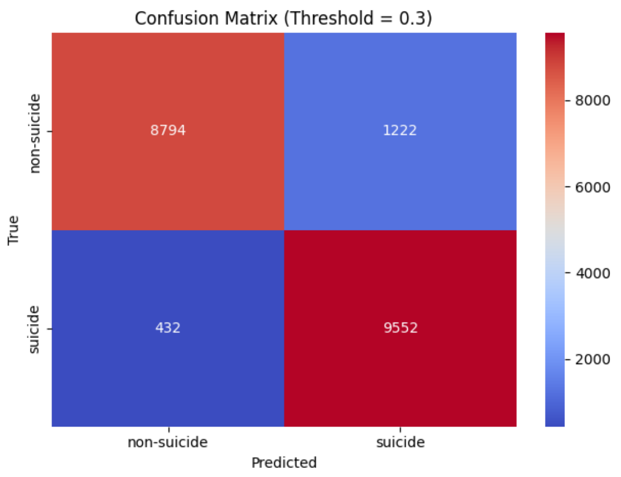
Figure 2. Universal Sentence Encoder Architecture

USE is a deep learning-based model that transforms each post into a fixed-size 512-dimensional dense vector. Unlike TF-IDF, USE captures the **semantic meaning** of entire sentences rather than just word frequency. This allows the model to understand context, such as the difference between “I want to die” and “I don’t want to die,” which share similar words but convey opposite meanings. USE achieves this by leveraging deep transformer or LSTM-based encoders trained on a variety of natural language tasks, enabling it to embed not just words, but the relationships between them. The 512-dimensional size is a design choice that balances expressive capacity with computational efficiency, allowing the vector to encode a rich variety of linguistic features while remaining compact enough for fast downstream processing. The resulting embeddings were then used as input features for classifiers like SVMs and neural networks.

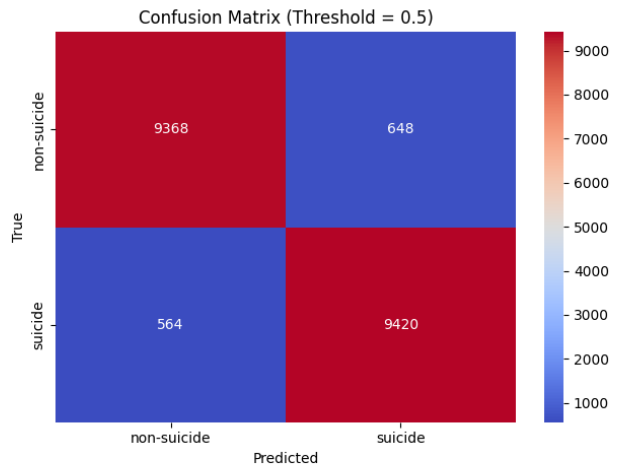
Algorithm 001: Support Vector Machine (SVM)

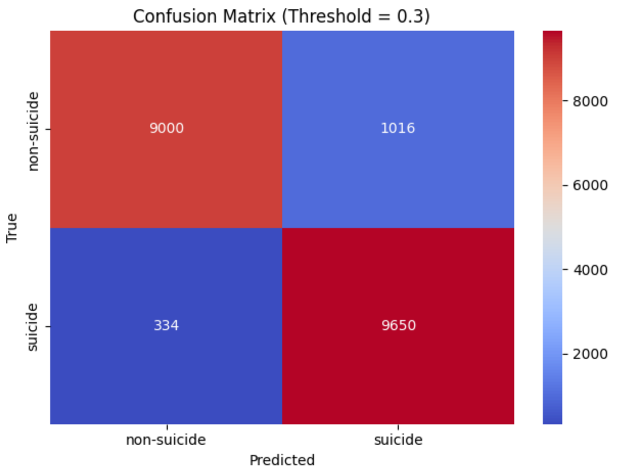
TF-IDF:





USE:



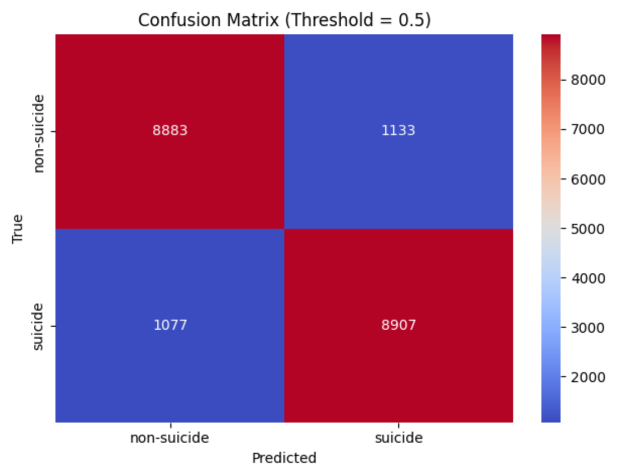


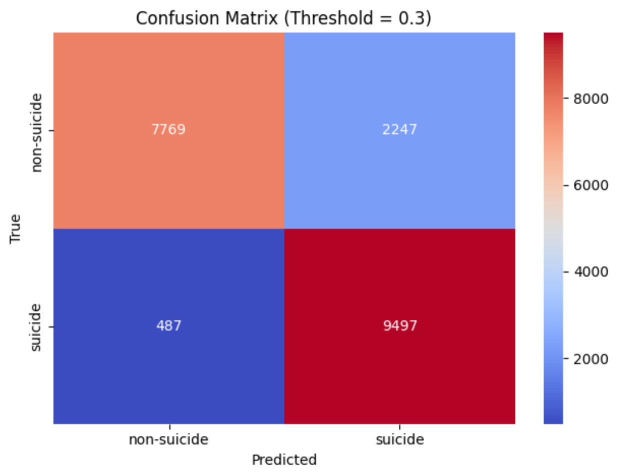
Support Vector Machines are linear classifiers that aim to find the optimal hyperplane that separates data points of different classes with the maximum margin. In this project, we used a linear kernel, which assumes that the data is linearly separable in its current feature space. This choice proved effective, as both TF-IDF and USE representations are high-dimensional and tend to naturally spread out the data, allowing for a linear decision boundary to form between classes. With TF-IDF, the SVM achieved an F2 score of 0.94 at threshold 0.3 and 0.92 at threshold 0.5, and slightly higher scores under USE, peaking at an F2 of 0.95. The model benefits from margin-based separation and works particularly well in high-dimensional vector spaces, which suits both TF-IDF and USE embeddings. Despite being simple and efficient, its performance is competitive with more complex models.

```
base_svm = LinearSVC(class_weight='balanced', random_state=42)
svm = CalibratedClassifierCV(base_svm, cv=2)
svm.fit(X_train_use, y_train)
```

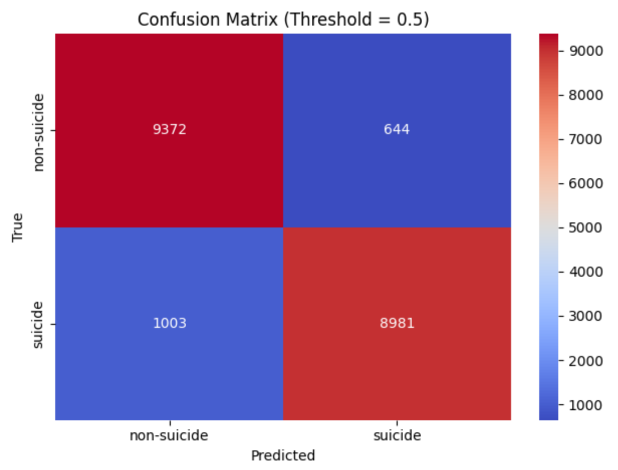
Algorithm 002: Random Forest

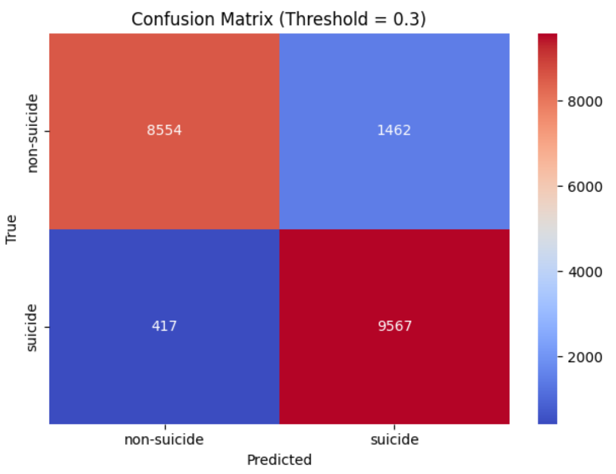
TF-IDF:



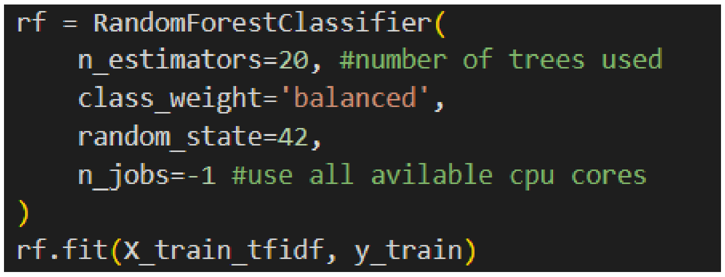


USE:





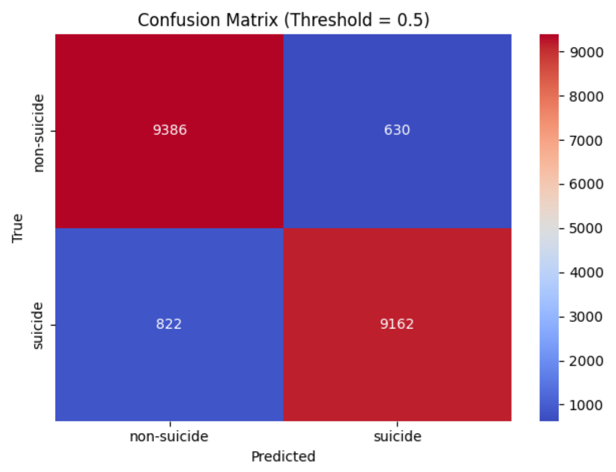
Random Forest is an ensemble of decision trees that makes predictions based on the majority vote across many randomly trained trees. While it performed decently across both TF-IDF and USE, it was generally weaker than the other models, especially in precision. For example, using TF-IDF at threshold 0.3, it had a recall of 0.95 but only a precision of 0.80, indicating a tendency to over-predict the positive class (suicide). This high recall may be useful for maximizing detection, but the lower precision comes at the cost of more false positives, making it less ideal for sensitive domains like suicide prevention. Its weakness can be attributed to the fact that text vectorizations like TF-IDF create thousands of features, most of which are zero (sparse), which makes it hard for the model to find informative splits.

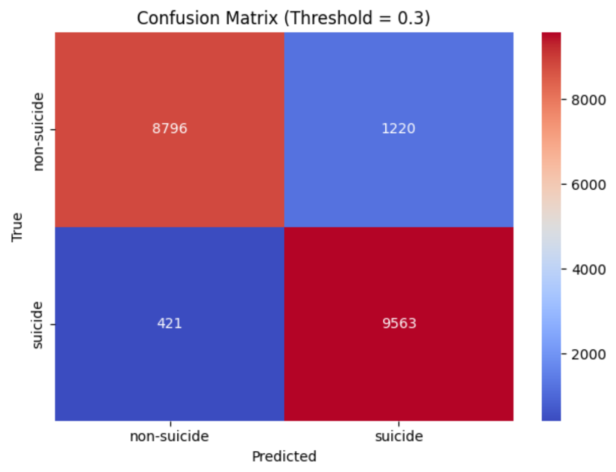


```
rf = RandomForestClassifier(
    n_estimators=20, #number of trees used
    class_weight='balanced',
    random_state=42,
    n_jobs=-1 #use all avilable cpu cores
)
rf.fit(X_train_tfidf, y_train)
```

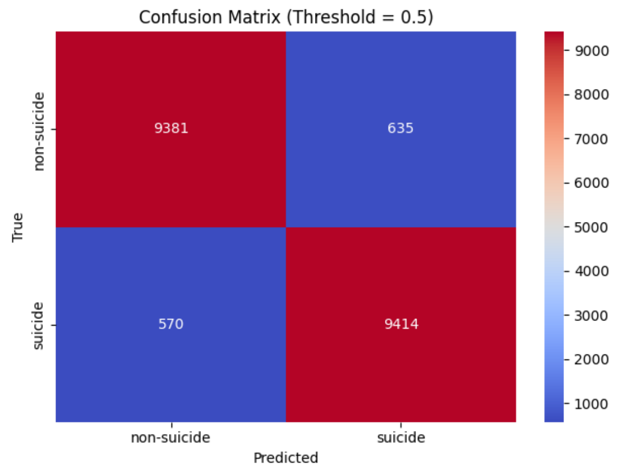
Algorithm 003: Logistic Regression

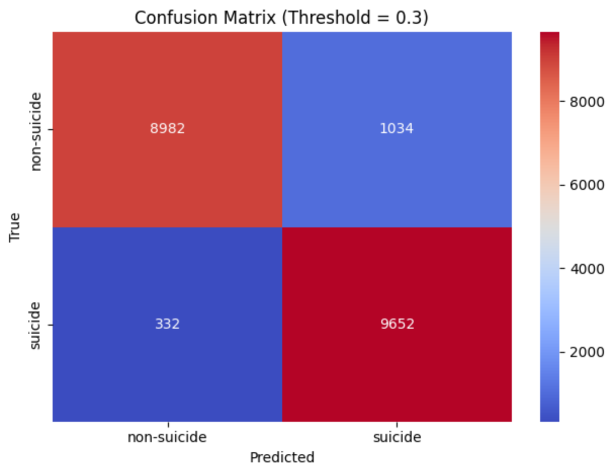
TF-IDF:





USE:





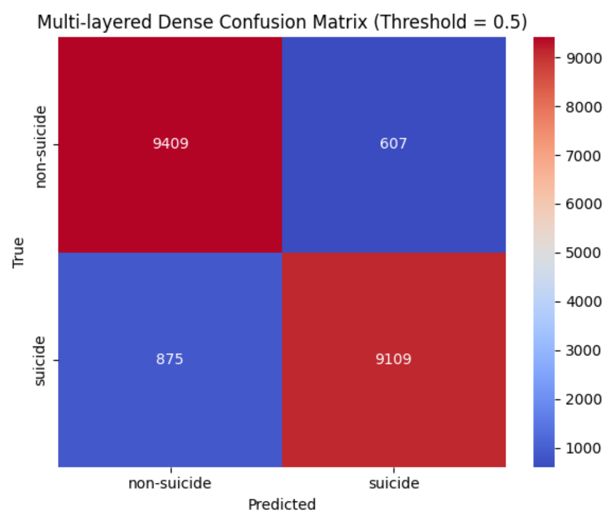
Logistic Regression is a simple linear model that estimates probabilities using a sigmoid function. Despite its simplicity, it performed nearly identically to the SVM across all representations and thresholds. This was especially true at threshold 0.3, where its F2 score reached 0.94 (TF-IDF) and 0.95 (USE), both emphasizing high recall with solid precision. Its strong performance demonstrates that when combined with rich features like USE or TF-IDF vectors, even basic models can be highly effective, suggesting that the feature quality has a larger impact than model complexity in this task.
However, unlike SVM, Logistic Regression lacks the concept of a margin and is more sensitive to overlapping data points, which may explain its slightly lower consistency in separating borderline cases.

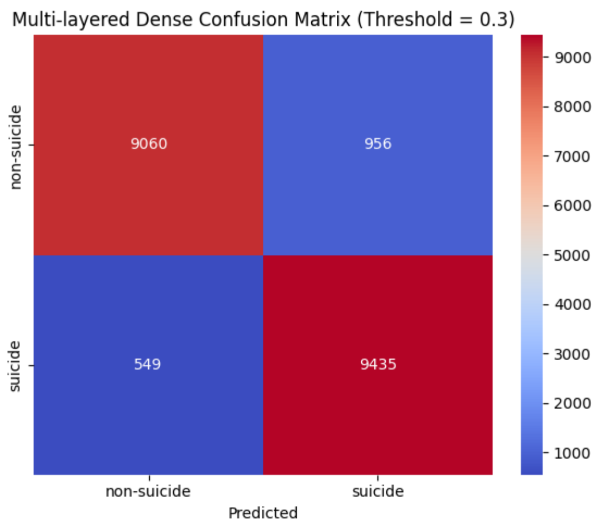
```
logreg = LogisticRegression(class_weight='balanced', max_iter=2000, random_state=42)
logreg.fit(X_train_tfidf, y_train)
```

Here we set the number of maximum iterations and fit it on our TF-IDF vectors or USE vectors

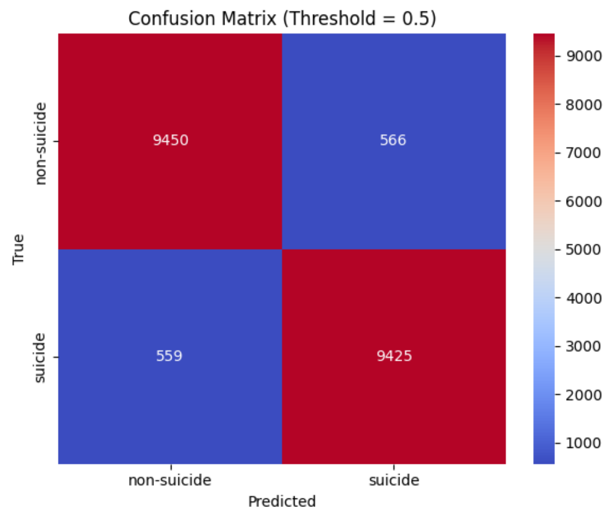
Algorithm 004: Fully Connected Neural Networks (FCNN)

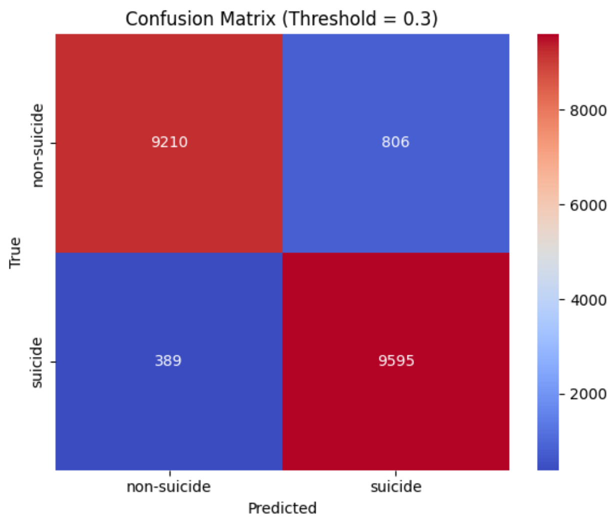
TF-IDF:





USE:

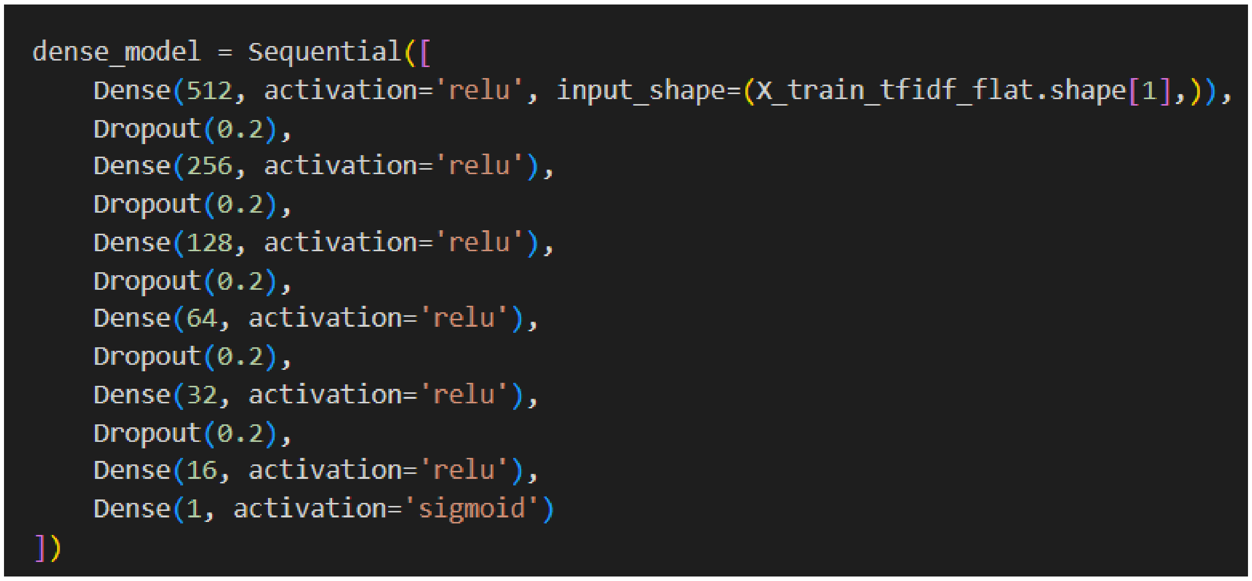




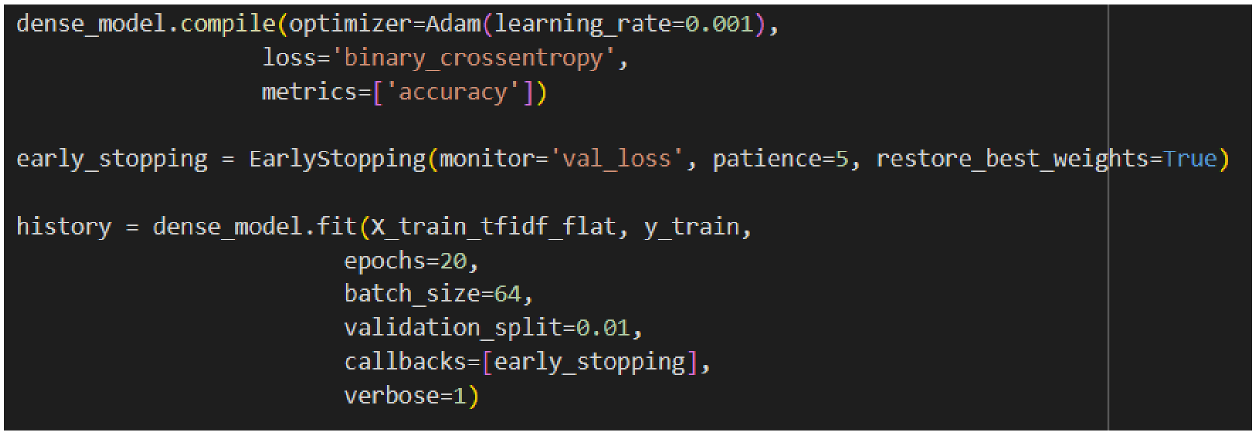
We implemented a deep fully connected neural network (FCNN) to classify suicide ideation based on the vectorized text inputs. The architecture begins with a 512-neuron dense layer and successively halves the number of units down to 16, followed by a final sigmoid output neuron for binary classification. Each dense layer uses ReLU activation. This activation function helps highlight meaningful positive signals in the input representations (TF-IDF and USE), which are already high-dimensional and contain many zero or near-zero values, while ignoring the rest. After each hidden layer, dropout layers with a 20% rate are inserted to prevent overfitting.

Because USE produces 512-dimensional representations and TF-IDF performed best at 512 neurons, we were able to reuse the same architecture across both input types, making the model both performant and computationally efficient.

We compiled the model with the Adam optimizer (learning rate = 0.001) and trained using binary cross entropy loss. Early stopping is employed with a patience of 5 epochs to avoid overfitting and restore the best-performing weights.



```
dense_model = Sequential([
    Dense(512, activation='relu', input_shape=(X_train_tfidf_flat.shape[1],)),
    Dropout(0.2),
    Dense(256, activation='relu'),
    Dropout(0.2),
    Dense(128, activation='relu'),
    Dropout(0.2),
    Dense(64, activation='relu'),
    Dropout(0.2),
    Dense(32, activation='relu'),
    Dropout(0.2),
    Dense(16, activation='relu'),
    Dense(1, activation='sigmoid')
])
```



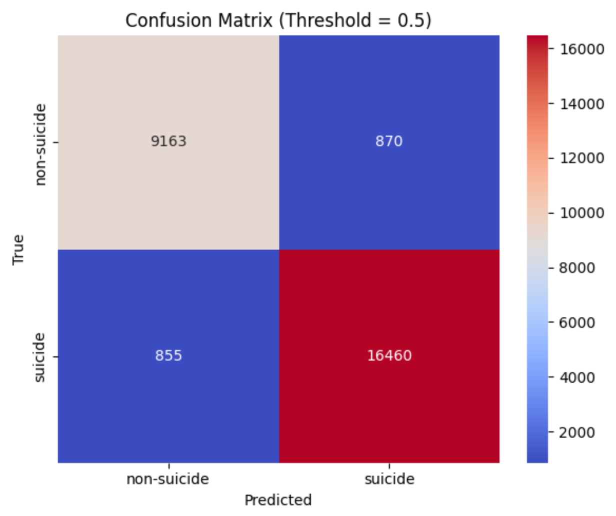
```
dense_model.compile(optimizer=Adam(learning_rate=0.001),
                    loss='binary_crossentropy',
                    metrics=['accuracy'])

early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

history = dense_model.fit(X_train_tfidf_flat, y_train,
                          epochs=20,
                          batch_size=64,
                          validation_split=0.01,
                          callbacks=[early_stopping],
                          verbose=1)
```

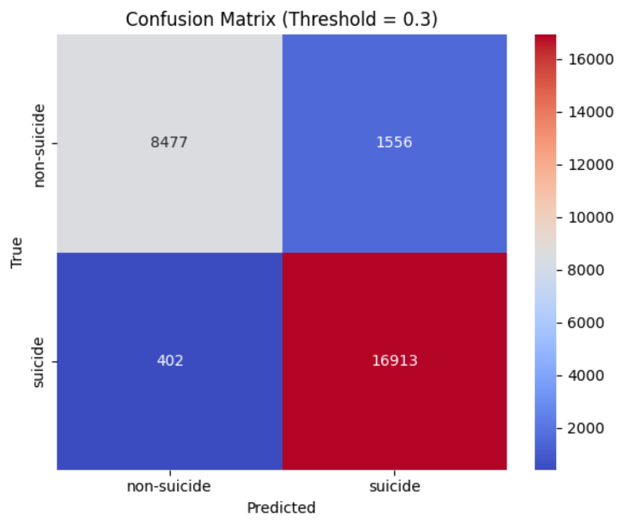
FCNN performed competitively: achieving an F2 score of 0.91 (TF-IDF, threshold = 0.5) and 0.93 (threshold = 0.3), and 0.94 across USE variants.

While the FCNN effectively captured patterns in the input representations, its lack of margin-based decision boundaries and sensitivity to sparse or redundant input dimensions may have limited its precision compared to the more focused separation achieved by the SVM.

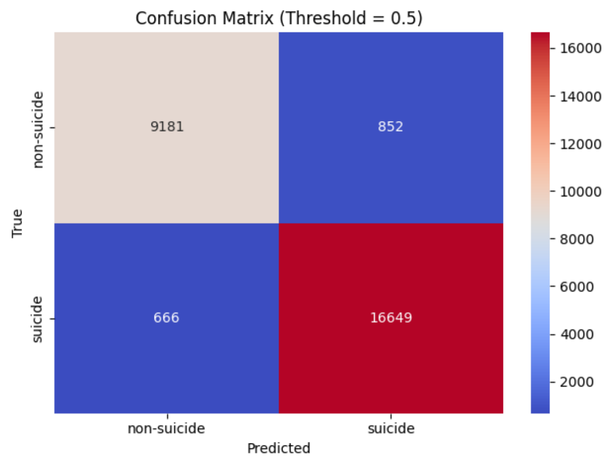
Algorithm 005: Support Vector Machine (SVM) + Gender Feature

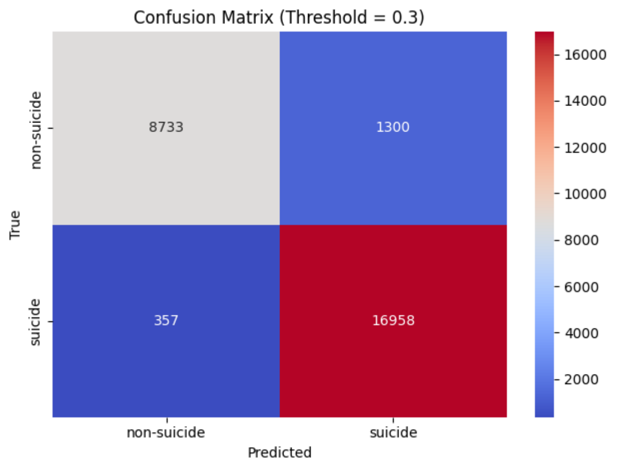
TF-IDF:





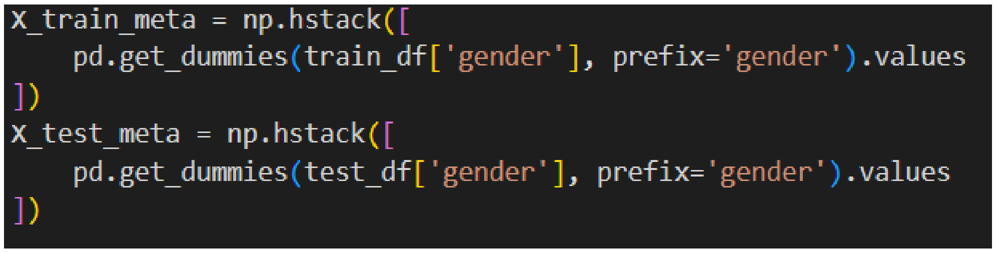
USE:





Incorporating gender as a structured feature alongside TF-IDF or USE vectors produced the best performance overall, peaking at an F2 score of 0.96 across both embedding types (at threshold 0.3). This result highlights the value of even minimal demographic cues when combined with linguistic features. The model outperformed all others across recall, precision, and accuracy. This is notable because gender on its own is not highly predictive, but when paired with text content, it helps sharpen the classifier's understanding of user context likely because suicide posts often include personal and reflective gendered language.

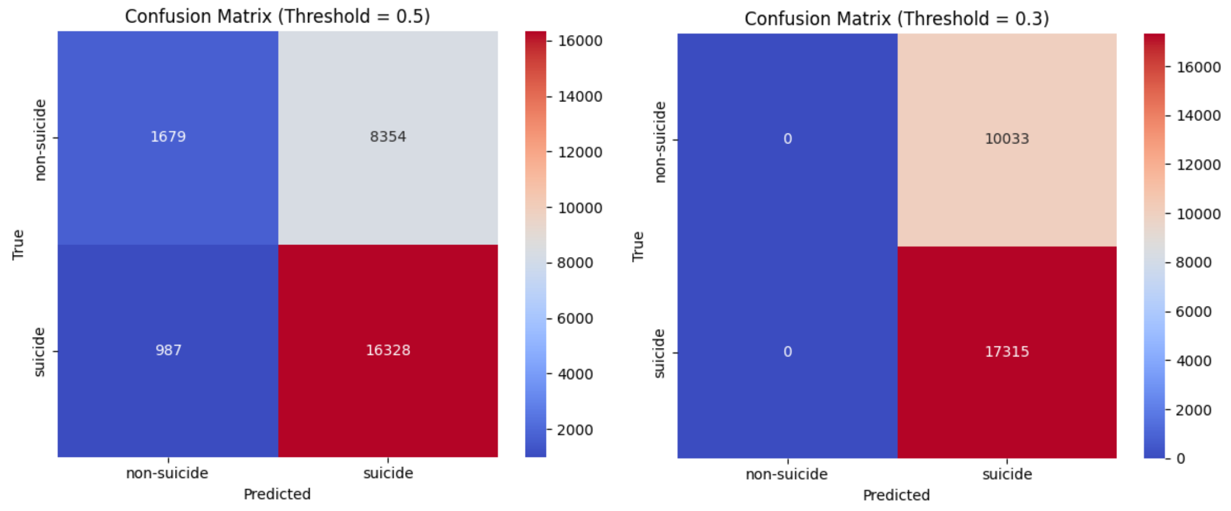
It's important to note that this result was obtained by evaluating only the subset of posts where gender could be reliably inferred. Posts without identifiable gender indicators were excluded from this analysis to better isolate and emphasize the impact of gender as a feature.



```
X_train_meta = np.hstack([
    pd.get_dummies(train_df['gender'], prefix='gender').values
])
X_test_meta = np.hstack([
    pd.get_dummies(test_df['gender'], prefix='gender').values
])
```

We used rule-based pattern matching to infer the gender of the poster as seen in our data analysis section and inserted it into our data as an indicator.

SVM Gender-Only Model:



To assess the standalone predictive power of gender, we trained an SVM using only gender information, after removing all entries without identifiable gender. The model achieved an F2 score of 0.89 at threshold 0.3, with perfect recall (1.00) but low precision (0.63). This means that while gender alone can identify nearly all suicide cases in the gendered subset, it misclassifies many non-suicide posts as suicidal. While not viable as a standalone predictor, this result supports the earlier finding that gender adds meaningful signal when used alongside linguistic features.

<u>Results:</u>

We evaluated model performance using two classification thresholds: the standard 0.5, and a more recall-oriented threshold of 0.3, allowing us to compare balanced predictions against those that prioritize reducing false negatives.

TF-IDF (threshold = 0.5)

Model / Metric	F2	Recall	Precision	Accuracy
SVM	0.92	0.91	0.93	0.93
Random Forest	0.89	0.89	0.88	0.89
Logistic Regression	0.92	0.91	0.93	0.93
FCNN	0.91	0.91	0.93	0.93
SVM + Gender	0.95	0.95	0.94	0.94

TF-IDF (threshold = 0.3)

Model / Metric	F2	Recall	Precision	Accuracy
SVM	0.94	0.95	0.88	0.92
Random Forest	0.91	0.95	0.80	0.86
Logistic Regression	0.94	0.95	0.88	0.92
FCNN	0.93	0.94	0.90	0.92
SVM + Gender	0.96	0.97	0.91	0.93

USE (threshold = 0.5)

Model / Metric	F2	Recall	Precision	Accuracy
SVM	0.94	0.94	0.93	0.94
Random Forest	0.90	0.89	0.93	0.92
Logistic Regression	0.94	0.94	0.93	0.94
FCNN	0.94	0.94	0.94	0.94
SVM + Gender	0.95	0.96	0.95	0.94

USE (threshold = 0.3)

Model / Metric	F2	Recall	Precision	Accuracy
SVM	0.95	0.96	0.90	0.93
Random Forest	0.93	0.95	0.86	0.91
Logistic Regression	0.95	0.96	0.90	0.93
FCNN	0.95	0.96	0.92	0.94
SVM + Gender	0.96	0.97	0.92	0.94

SVM only on gender

Threshold / Metric	F2	Recall	Precision	Accuracy
Threshold = 0.5	0.86	0.94	0.66	0.66
Threshold = 0.3	0.89	1	0.63	0.63

Model Performance Summary:

SVM (Support Vector Machine):
SVM proved to be the most successful model by far, thanks to its ability to handle high-dimensional, linearly separable data. Its simplicity and margin-based optimization help it generalize well. Combined with the use of a linear kernel, it performed strongly on both TF-IDF and USE embeddings, which were shown to be linearly separable in our KMeans clustering analysis. The only limitation is its reduced flexibility compared to non-linear models, which may hinder performance in more complex or nonlinear feature spaces.

Random Forest:
Random Forest struggled with the high dimensionality and sparsity of TF-IDF and USE vectors. It tends to overfit on noisy features and can produce too many false positives, particularly in imbalanced or sparse settings. Its strength on structured tabular data does not translate well to text-based representations without careful feature selection or dimensionality reduction. This weakness is the reason why we have decided to avoid other tree based models and focus our attention on other models that benefit more from the linearly separable nature of our embeddings.

Logistic Regression:
Logistic Regression performed nearly identically to SVM, showing that simple linear models can be highly effective when paired with rich features. It is efficient and interpretable, but it lacks the robustness of margin-based optimization and may be more sensitive to noise or borderline cases in the data.

Fully Connected Neural Network (FCNN):
The FCNN excelled at learning from both sparse and dense representations, capturing more complex patterns in the data thanks to its deep, layered structure. It was particularly effective with USE embeddings. However, it requires more computational power and is more prone to overfitting, especially without large-scale data or heavy regularization.

SVM + Gender Feature:
Incorporating gender alongside text significantly improved performance. While gender alone was not a reliable predictor, it added context that enhanced the classifier's understanding when used in combination with linguistic features. This hybrid approach proved especially useful in cases where gendered language correlated with introspective, personal expression. However, this improvement is limited to samples where gender could be confidently detected.

Prediction Examples And Error Analysis:

To better understand the behavior and limitations of our best performing model, SVM on USE embeddings, we examined individual examples of correct and incorrect predictions.

True Positive (Correctly predicted suicide)

- “people dont have empathy for you when youre wealthy... ive been struggling with depression for years... why do i feel like less of a person for having money... i cant take this any more”

This example contains multiple direct signals of distress and depression, making it an easier and more confident classification for the model.

True Negative (Correctly predicted non-suicide)

- “playlist review i have this spotify playlist consisting of pink guy and joji would like to hear your opinions and song recommendations”

This is a typical example of casual, benign teenage discourse that do not indicate mental health concerns, allowing the model to correctly classify them as non-suicidal.

False Positive (Non-suicide predicted as suicide)

- “my grooming story im a home groomer i have a ranch house on a full basement groom shop... client shows up husband drove she comes downstairs with me and dog and she is terrified husband had a confrontation with some teenagers on the way here shes scared crying and it becomes clear to me shes in an abusive relationship and still she leaves me with the... client comes back is ok but still scared…”

This post was likely misclassified due to the emotionally intense content, mentions of fear, abuse, and relationship trauma, and the presence of first person narration. Phrases like “she’s terrified” and “abusive relationship” resemble the language used in posts about personal crises or mental health struggles.

False Negatives (Suicide posts predicted as non-suicide)

- “what are the songs that comfort you you know the ones that just resonate with your feelings when you feel down i need more songs to cry to”
- “i silenced my thoughts but its no better than it was before i no longer have the voices in my head but this infinite silence isnt better”

Although both posts are in the first person, owing to the introspective nature of suicide posts, the language, phrasing, and subject of the posts mask their emotional content, leading to misclassification. The first post could be misconstrued as a casual question asking for playlist suggestions, while the second post is very poetic and eccentric, making it difficult for the model to interpret.

Conclusion:

In our project, we compared a variety of machine learning and deep learning models aimed at detecting suicidal ideation using two text vectorization methods: TF-IDF and USE. The latter generally performed better, as it captures the semantic meaning of full sentences rather than relying solely on word frequency.

We evaluated model performance at two classification thresholds: the standard 0.5 and a lowered threshold of 0.3. The lower threshold was used to prioritize recall and improve the F2 score, since in this context, identifying true suicidal posts is more important than avoiding false positives.

SVM consistently delivered strong results, often outperforming other models, with logistic regression following closely. One key insight was the role of supplementary features like gender. While gender alone was not a strong predictor, it contributed valuable context when combined with textual input, helping our SVM model achieve the best overall performance.

Our results show that high-quality textual representations, whether simple or deep, are critical for effective suicide ideation classification. Additional features such as gender can enhance performance when used thoughtfully, but are insufficient on their own.

An analysis of correct and incorrect predictions revealed that our best performing model, SVM + USE embedding, generally performs well on emotionally direct posts but can be misled by ambiguous or poetic language, emphasizing the challenges of nuanced language understanding in sensitive contexts like suicide detection.

Future work:

In future iterations of this project, we would like to explore more reliable and diverse data sources beyond Reddit, such as identity-linked platforms like Facebook, where suicidal intent could be more verifiable. Additionally, with access to more powerful hardware, implementing transformer-based models like BERT could improve contextual understanding beyond what TF-IDF or USE offer.

Lastly, we aim to extract and incorporate more structured features from the text, similar to how gender was used in this project, to provide richer context and potentially enhance classification accuracy.