# Text Classification

## Using SVM with Various Vectorization Techniques

Ahmed Khwaja
Amrita Ajay Sagar
Hassan Sayed
Manisha K

# Project Overview

- **Objective:** Comparing different approaches to classify text into different categories
- **Dataset:** Contains various text samples with corresponding categories.
- Techniques Used:
    - Bag of Words
    - TF -IDF
    - Word2Vec
    - SVM Model for classification
    - Deep Learning with BERT & other transformer embeddings

# Data Preprocessing

- Steps taken:
  - Convert text to lowercase
  - Remove URLs, emails, and hashtags
  - Remove special characters and numbers
  - Tokenization
  - Part-of-Speech (POS) Tagging
  - Stopword removal
  - Lemmatization
- Libraries used : NLTK, re, sklearn

## Data Splitting

- Initial Stratified Split:
    - 98% Training Data
    - 2% Test Data
- Further Splitting:
    - 80% Train
    - 20% Validation

# Vectorization Techniques

1. **Bag of Words (BoW)**
   - Converts text into frequency-based numerical representation.
   - Uses CountVectorizer from sklearn.
2. **TF-IDF (Term Frequency-Inverse Document Frequency)**
   - Assigns weights to words based on importance.
   - Uses TfidfVectorizer from sklearn.
3. **Word2Vec**
   - Generates dense word embeddings.
   - Uses Word2Vec from gensim.

# BERT-training Approach

- **Model Used:** BERT-base-uncased
- **Training:** Trained from scratch for 50 epochs
- **Results:**
  - Accuracy: **~80%**
  - Issues: Computationally expensive, longer training time

# Embeddings from Deep Learning Models

**Sentence Transformer Models Tested:**

- **MiniLM+ Xgboost**: Accuracy **~62.05%**
- **MPNet+ Xgboost**: Accuracy **~65.2%**
- **BERT+ XgBoost:** Accuracy: **~53.96%**

**Conclusion:** Deeplearning embeddings performed poorly.

# Classical ML - SVM + TF-IDF

**Feature Extraction:** TF-IDF (Term Frequency-Inverse Document Frequency)

**Classifier Used:** SVM (Support Vector Machine)

**Why TF-IDF + SVM?**

- Simpler and computationally efficient
- No need for GPU
- Performed best for our dataset

# Model Training - SVM

- **Model Used**: Support Vector Machine (SVM)
- **Hyperparameters**:
  - Kernel: Linear
  - Probability: True
  - Random Seed: 42
- **Libraries Used**: sklearn.svm.SVC

# Alternative approaches

- **Multinomial NaiveBayes**: Accuracy **~68**
- **Random Forest:** Accuracy **~77**
- **KNN:** Accuracy: **~66**

# Model Comparison

| Algorithm | Accuracy | Precision | F1-Score | Recall |
|---|---|---|---|---|
| MiniLM+ Xgboost | 62.05 | 63 | 62 | 62 |
| MPNet+ Xgboost | 65.20 | 64 | 64 | 64 |
| BERT+ XgBoost | 53.96 | 54 | 53 | 54 |
| Multinomial NaiveBayes | 68 | 66 | 66 | 68 |
| Random Forest | 77 | 76 | 75 | 77 |
| KNN | 66 | 65 | 65 | 66 |
| SVM | 81 | 80 | 81 | 80 |

# Final Model Selection

**Final Choice: SVM with TF-IDF**

**Reason for Selection:**

Outperformed BERT-based approaches and other ML approaches

Faster training and inference

Bag of Words was competitive but slightly less effective

Word2Vec also underperformed

**Best Accuracy & F1-Score** on validation data

# Conclusion

**Key Findings:**

- Deep learning models did not significantly outperform traditional ML for this dataset.
- **SVM + TF-IDF was the optimal choice** due to its efficiency and accuracy.