Cyberbullying Detection

Classifying texts from various social media platforms into cyberbullying/non-cyberbullying

Group 6
Veronika Junková 113300372
經濟三 111208094 鄭琬蓁
創國三 111ZU1056 蔡珮元

Agenda

01. Defining the Problem and Context

02. Data Collection & Processing

03. Black-Box Design and Performance

04. Visualization and Interpretation of Results

TOPIC: Classifying texts from various social media platforms into cyberbullying/non-cyberbullying

Goal

Creating a tool to classify comments on social media and detecting cases of cyberbullying, offering support for navigating the complexities of online communication.

The Problem

Social Media Immersion + Ambiguous Comments + Unintentional Offense ▶ Tool to classify comments and detect bullying

Related information

Pew Research Center, 2021: Around 41% of Americans have experienced online harassment, and 66% have witnessed others being harassed online.

EF English Proficiency Index, 2022: Over 72% of internet users communicate in a non-native language, often leading to unintentional misunderstandings or insults.

3 kaggle data sets
+ 2 extra data sets: Web scraping + pre-trained model
= Combination to perform the next step of balancing data and test-train split

Data Collection

First Dataset Collection

Kaggle

Mix insults + Youtube + Twitter

Label 0 (non - cyberbullying) : 122497
Label 1 (cyberbullying) : 21894

Total: 144391

Second Dataset Collection (Just collect label 1)

Web Scraping + Using pre-trained model to label (From YT)

APT + 911 Attack

Label 0 (non - cyberbullying) : 122497
Label 1 (cyberbullying) : 22615 (21894+721)

Total: 145112

This code shows how we make the label.

```
# We are scraping comments from Youtube and then using a text-classification model from transformers package to label the rows to then
# This model is intended to be used on emotion classification and not bullying, so it is not perfect. We are not going to be using

from transformers import pipeline
import pandas as pd

classifier = pipeline("text-classification", model="j-hartmann/emotion-english-distilroberta-base", truncation=True, padding=True)

bullying_keywords = [
        'anger', 'disgust', 'hate', 'rage', 'frustration', 'envy',
        'annoyed', 'irritation', 'resentment', 'dislike', 'contempt', 'revenge', 'hostility', 'spite',
        'bitter', 'jealous', 'malice', 'vengeance', 'cruelty',
        'attack', 'violence', 'hostile', 'aggression', 'offense', 'hurtful', 'vile', 'abominable', 'intense',
        'mock', 'bully', 'shame', 'insulting', 'demeaning', 'provoking', 'ridicule', 'scorn', 'mockery',
        "dick", "bitch", "cunt", "shit", "homo", "faggot", "fuck", "stupid", "idiot", "scumbag",
]

bullying_labels = ['insult', 'threat', 'abuse', 'humiliation', 'harassment', 'intimidation',
                                'cyberbullying', 'offensive', 'bullying', 'exclusion', 'shaming', 'provocation',
                                'mockery', 'degradation', 'verbal_abuse', 'threatening', 'defamation', 'scorn',
                                'derogatory', 'humiliation', 'discrimination', 'denigration', 'exploitation', 'malice']
```

Data Processing

To address the unbalanced dataset with more non-bullying than bullying instances, we are down-sampling the non-bullying class using sklearn.model_selection to improve balance and performance.

Balancing data

Down-sampling the non-bullying to balance the data more

Test-train split

Using sklearn.model_selection to improve balance and performance.

Data Processing

Balancing data

Test-train split

Step 1

Separate data by label and Randomly sample 25,000 rows from label 0

Step 2

Combine all label 1 data with the sampled label 0 data

Step 3

Shuffle the dataset to mix the rows

Step 4

Verify the label distribution

label	
0	25000
1	22592

Data Processing

Balancing data

Test-train split

Step 1

Split the Dataset

- finetune_df (80%)
- test_df (20%)

Step 2

Downsample for Balance and Performance

- finetune_df: 5,000
- test_df: 1,000

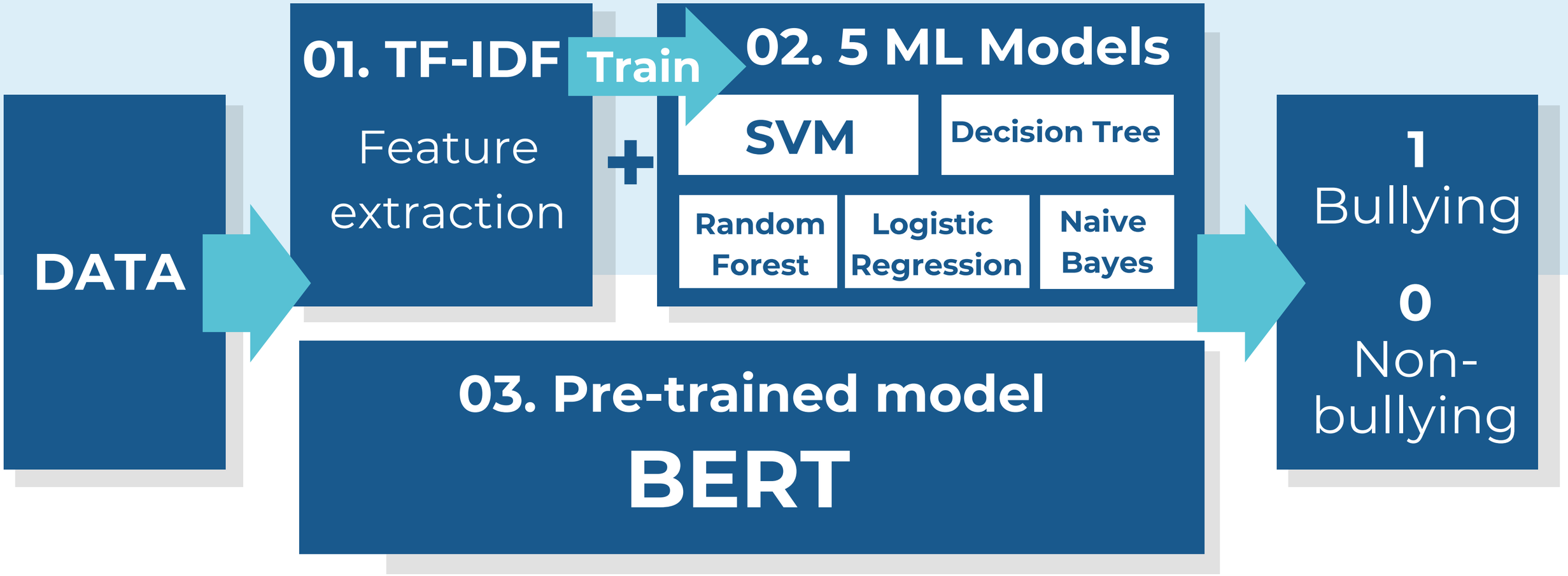
Step 3

Convert Dataset

Step 4

Verify the Data

Black-Box Design and Performance



01 TFIDF + **02 ML Models**

How TFIDF trains classifiers

01 Use TfidfVectorizer to transform text into numerical features.

```
# Transform text data into numerical format
vectorizer = TfidfVectorizer(max_features=2000)    # Limit to 2000 features to reduce dimensionality
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)
```

02 Then all classifiers were trained using this TF-IDF transformed feature matrix.

```
# Train classifiers
trained_classifiers = {}
for name, clf in classifiers.items():
        clf.fit(X_train_tfidf, y_train)
        trained_classifiers[name] = clf
```

03 Each trained classifier are used to predict labels for the testing data, then the predictions will be stored.

```
for name, clf in trained_classifiers.items():
        y_pred = clf.predict(X_test_tfidf)
```

01 TFIDF

Importing Libraries and Download NLTK Resource and What is TFIDF

Importing Libraries and Download NLTK Resources

NLTK: A library for natural language processing (NLP).
We're importing tools to:

- **Tokenize**: Split sentences into words.
- **Stem/Lemmatize**: Reduce words to their root forms (optional for TF-IDF but often used in preprocessing).

By the way, NLTK needs additional data to work properly:

- punkt: Helps with splitting sentences into words.
- wordnet: A database for finding word relationships, needed for lemmatization.

01 TFIDF

Importing Libraries and Download NLTK Resource and What is TFIDF

What is TFIDF?

TF-IDF stands for Term Frequency-Inverse Document Frequency. It is a statistical measure used in text analysis to evaluate the importance of a word in a document relative to a collection of documents (called the corpus).

TF (Term Frequency): How often a word appears in a document.

- Example: If "data" appears 5 times in a document with 100 words:
- TF=5/100=0.05

IDF (Inverse Document Frequency): How rare a word is across all documents.

- Example: If "data" appears in 1,000 out of 10,000 documents:
- IDF=log(10,000/1,000)=1

TF-IDF Score: Multiply the two to see how important a word is.

- Example:TF−IDF=0.05×1=0.05

02 ML Models

Used models like Logistic Regression, Random Forest, Decision Tree, Naive Bayes, and SVM to evaluate performance.

Model	Strengths	Weaknesses	Example Use
Logistic Regression	Simple, interpretable	Struggles with non-linear data	Email classification
Random Forest	Handles complex data, reduces overfitting	Slower with large datasets	Customer predictions
Decision Tree	Easy to understand	Overfits on noisy data	Loan approval
Naive Bayes	Fast, works well for text	Assumes feature independence	Sentiment analysis
SVM	Works well with high dimensions	Slower with large datasets	Image recognition

03 Pre-trained model

What is BERT and How it works.

What is BERT?

BERT (Bidirectional Encoder Representations from Transformers) is a popular pre-trained model developed by Google. It's specifically designed to understand the meaning of words in context by analyzing the text in both directions (left-to-right and right-to-left).

Key Features of BERT

Bidirectional Understanding:

- BERT looks at the entire sentence at once, both from left-to-right and right-to-left, to understand words better.
- Example: Sentence: "The bank on the river is beautiful."

Pre-trained on Large Data:

- BERT has already been trained on tons of text (like Wikipedia) and knows a lot about language.

Fine-tuning:

- After BERT is pre-trained, you can adjust it for specific tasks (like classifying reviews as "positive" or "negative") using your own data.

03 Pre-trained model

What is BERT and How it works.

How Fine-tuning Works

Step 1 Start with BERT, which already understands language.

Step 2 Add a small layer to teach it your specific task (e.g., sentiment analysis).

Step 3 Train it on your data (e.g., “bullying" or "non-bullying" reviews).

Step 4 BERT becomes great at specific task!

Visualization and Interpretation of Results

01. Model Performance Metrics Visualization

02. TFIDF Visualization

03. Error Analysis of BERT's Performance

04. Final Summary

Performance Evaluation Using 4 Metrics

01 Accuracy

Indicates the overall correctness of predictions.

02 Precision

Measures the accuracy of toxic comment predictions.

03 Recall

Assesses the model's ability to identify toxic comments

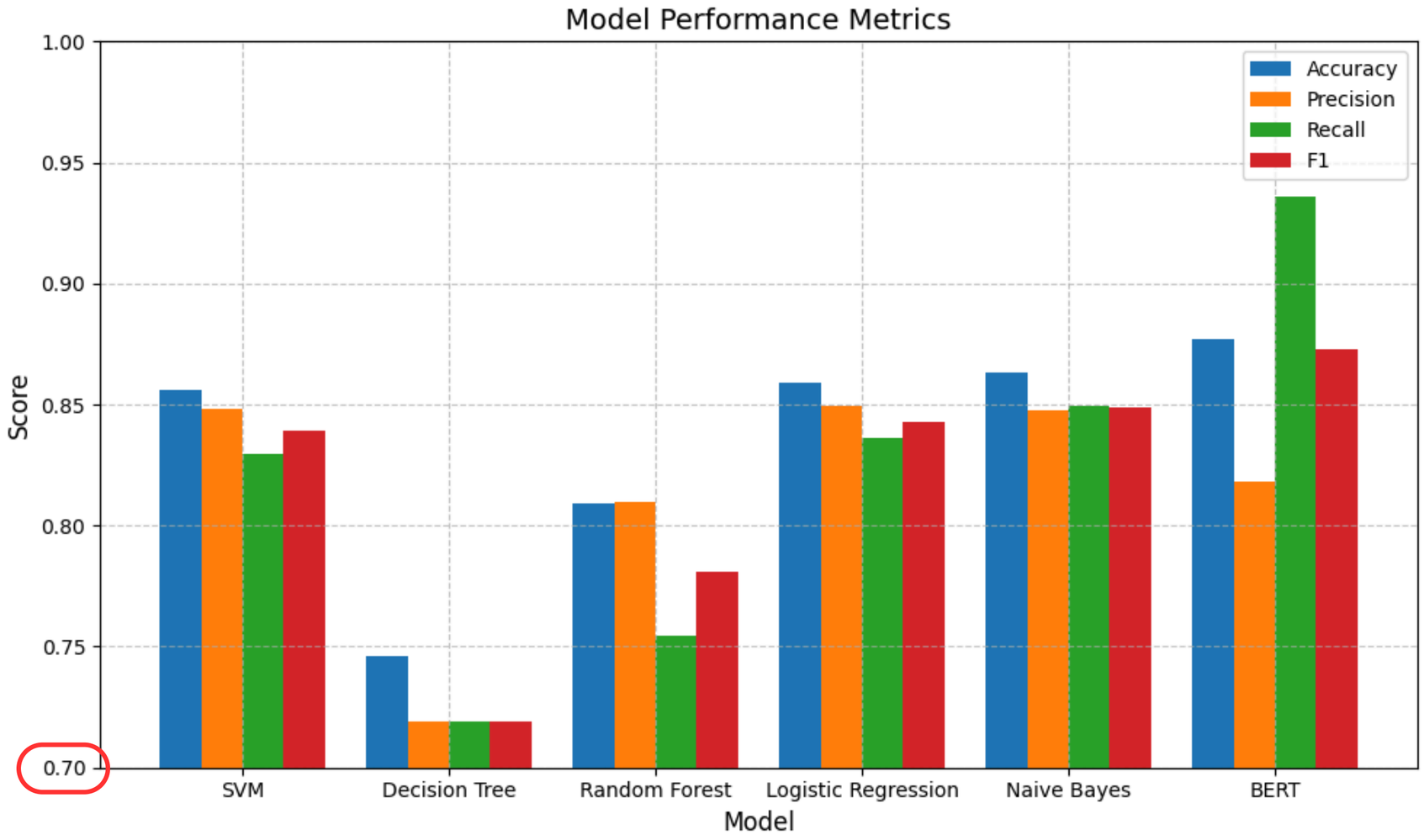
04 F1 Score

Balances Precision and Recall, especially crucial for imbalanced datasets.

01.

Model Performance Metrics Visualization

BERT outperformed all other models, while Decision Tree performed the worst.



01.

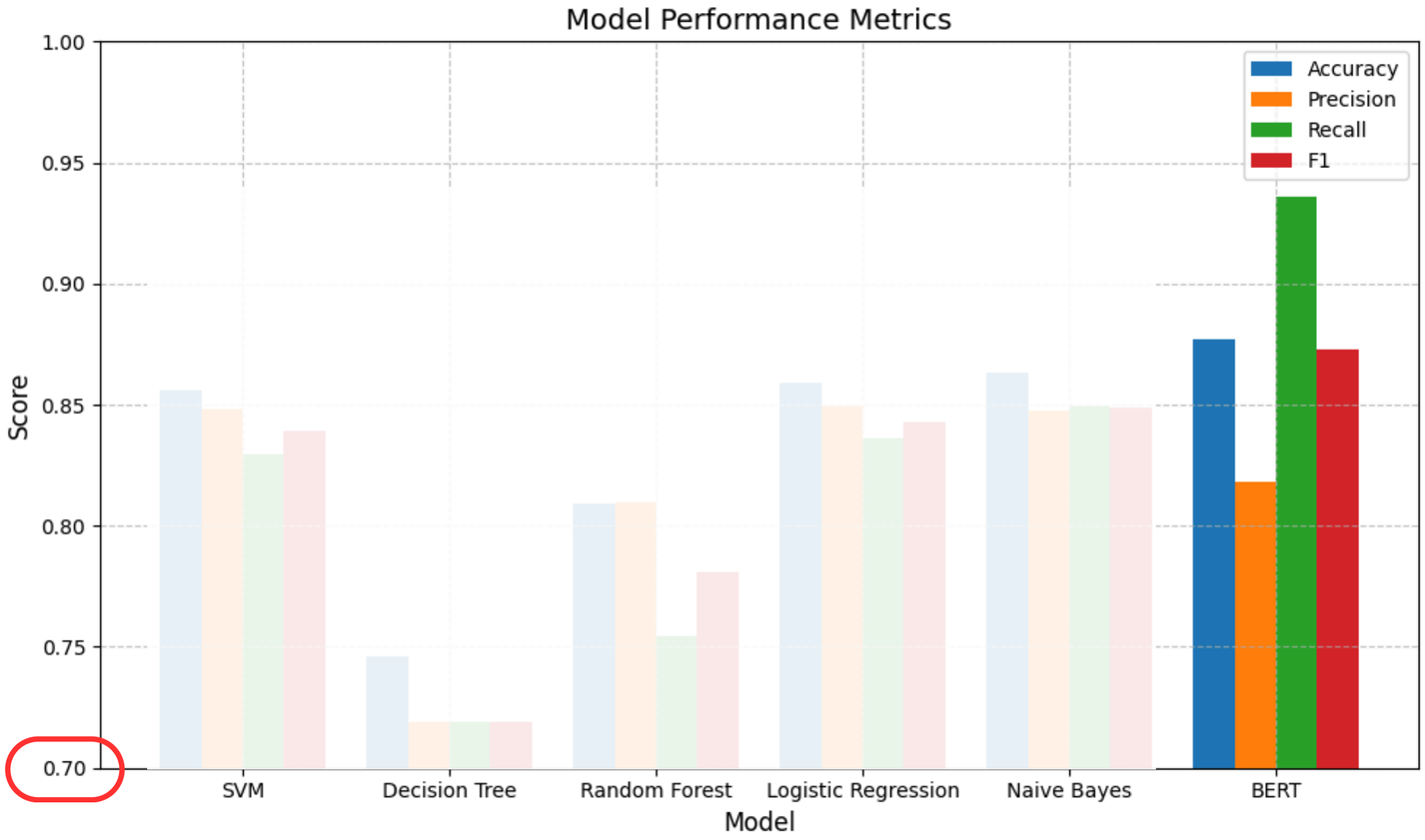
Model Performance Metrics Visualization

BERT outperformed all other models, while Decision Tree performed the worst.

	Accuracy	Precision	Recall	F1
SVM	0.856	0.848416	0.829646	0.838926
Decision Tree	0.746	0.719027	0.719027	0.719027
Random Forest	0.809	0.809976	0.754425	0.781214
Logistic Regression	0.859	0.849438	0.836283	0.842809
Naive Bayes	0.863	0.847682	0.849558	0.848619
BERT	0.877	0.818182	0.935841	0.873065

01. Model Performance Metrics Visualization

BERT outperformed all other models, while Decision Tree performed the worst.

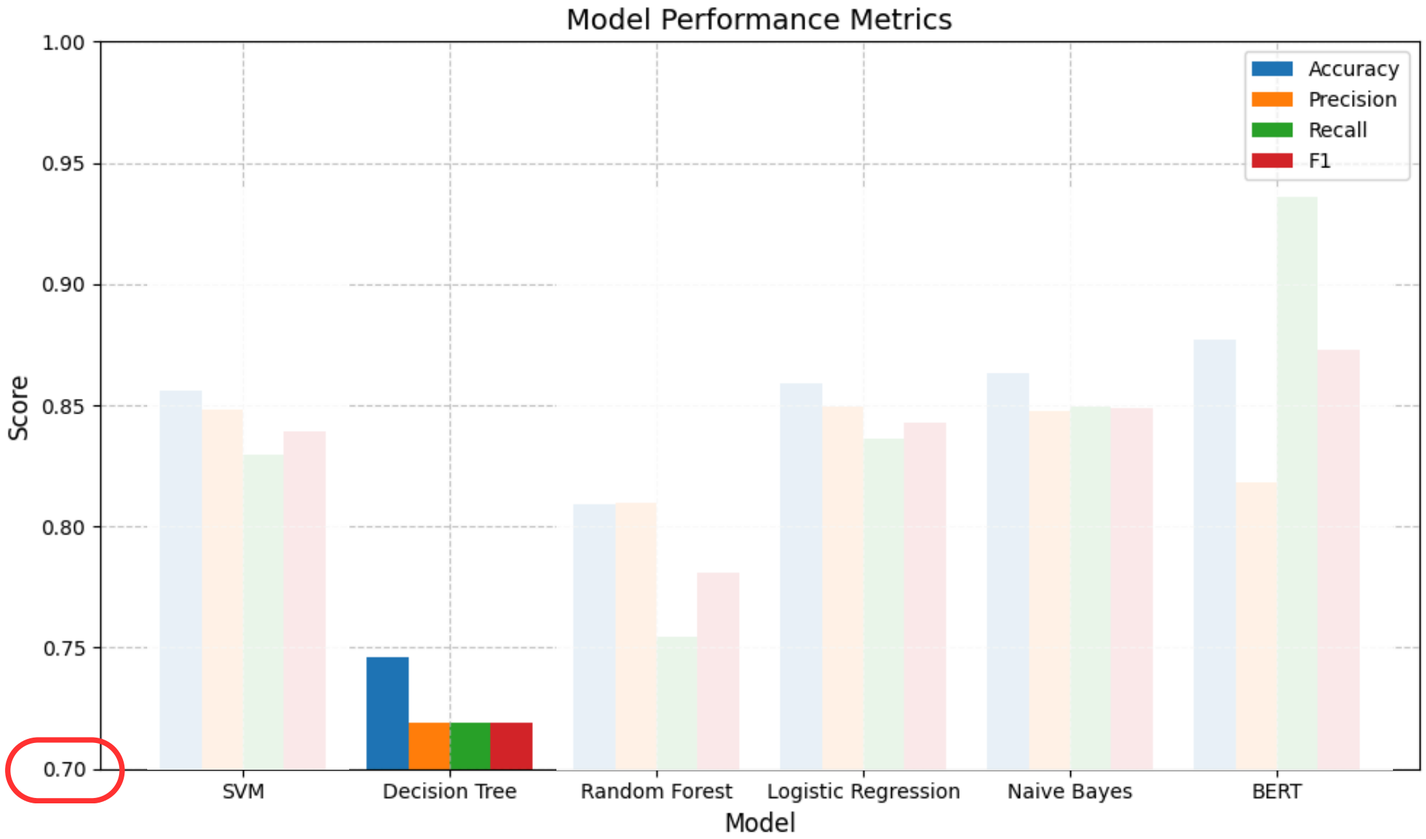


BERT

- Outperformed all other models, achieving the highest scores across all metrics.
- Particularly excelled in Recall and F1 scores.
- Its performance is attributed to its contextual understanding of language and pre-training on large datasets, making it highly effective in identifying toxic comments.

01. Model Performance Metrics Visualization

BERT outperformed all other models, while Decision Tree performed the worst.

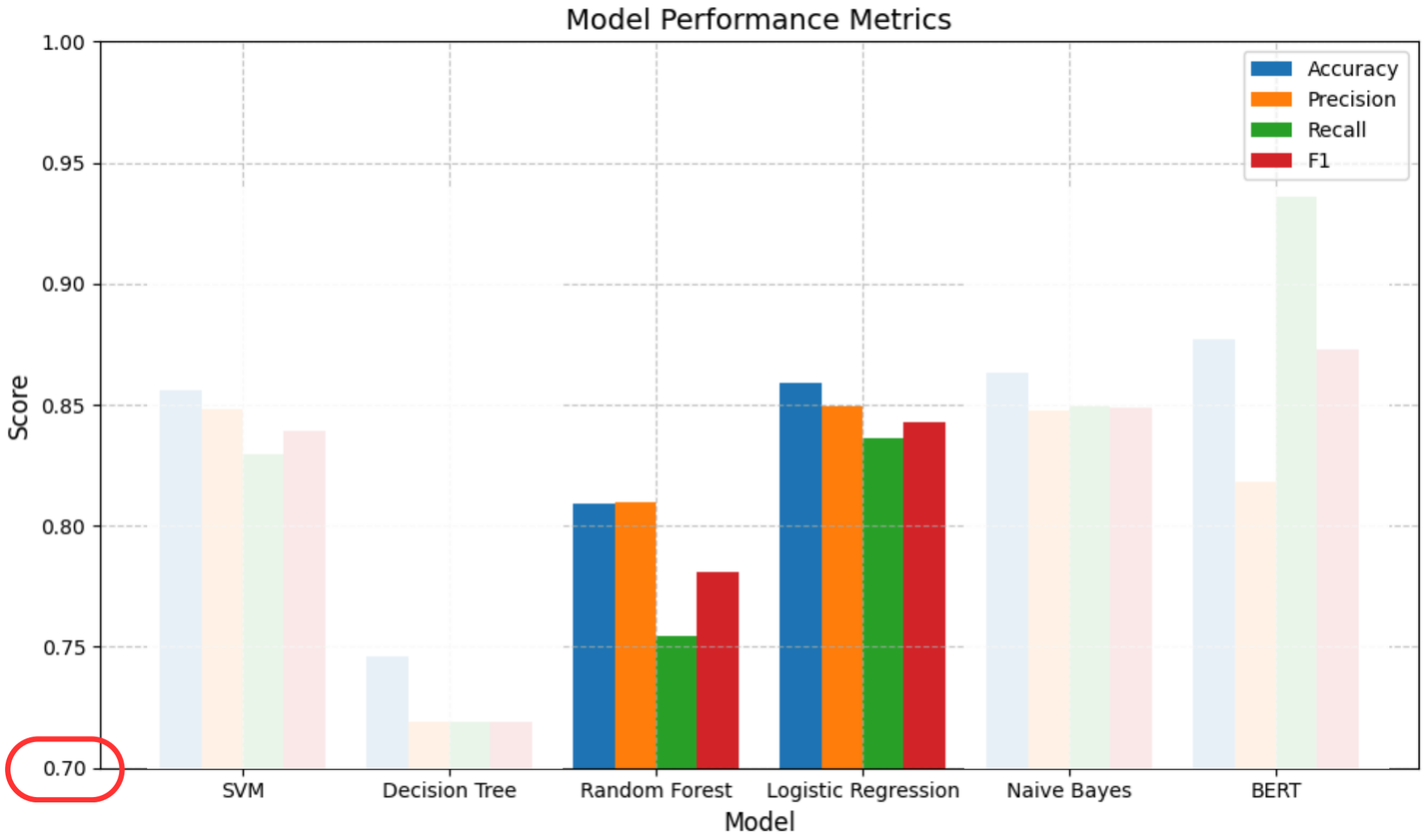


Decision Tree

- Performed the worst among the models.
- Exhibited low Precision and Recall, indicating difficulty in generalizing patterns.
- Likely suffered from overfitting on noisy data.

01. Model Performance Metrics Visualization

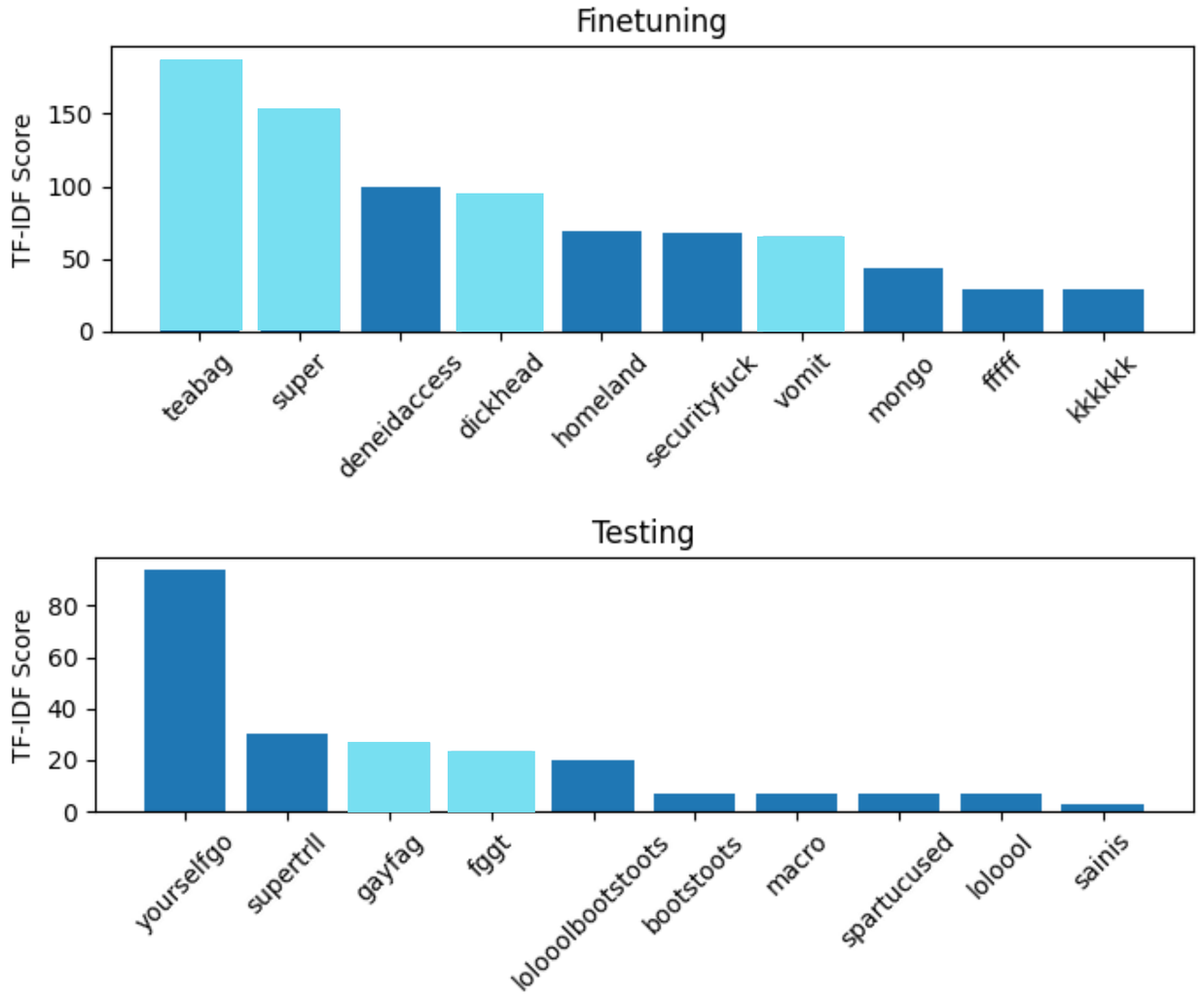
BERT outperformed all other models, while Decision Tree performed the worst.



Traditional Models (Random Forest & Logistic Regression)

- Delivered decent performance but scores were lower than BERT.
- Demonstrated limitations compared to advanced language models like BERT.

02. TFIDF Visualization



02. TFIDF Visualization

This is how we cleaned the data

```
[22] # Cleaning the text

     import re

     def clean_text(text):
         text = re.sub(r"http\S+", "", text)  # Remove URLs
         text = re.sub(r"@\w+", "", text)  # Remove usernames starting with @
         text = re.sub(r"[^a-zA-Z\s]", "", text)  # Remove non-alphabetic characters
         text = re.sub(r"\s+", " ", text)  # Replace multiple spaces, newlines, and tabs with a single space
         text = text.lower().strip()  # Convert to lowercase and strip leading/trailing spaces
         return text

     data["text"] = data["text"].apply(clean_text)
```

02. TFIDF Visualization

Potential problems:

- **Highly Specific Words**
  - The dataset contains highly specific words, possibly originating from the same individual or used as nicknames in web-scraped examples.
- **Incomplete Data Cleansing**
  - Efforts were made to remove the most obvious cases, but fully cleansing the dataset would require significantly more time.
- **Potential Weakness**
  - These issues represent a potential weakness of the model, as the remaining noise might affect its performance.

03. Error Analysis of BERT's Performance

We want to find items with different labels and prediction results, and find a few more special prediction error data for further analysis.
From the picture, we can see that the values of label and predict are not the same.

```
# BERT being the best performing model - let's look at error analysis of his results:

bert_result = test_df[test_df['label'] != test_df['predict']]
bert_result.head(3)
```

	text	label	predict
39001	wow thats alot	0	1
35611	excellent command of the english language once...	1	0
40597	you currently appear to be engaged in an edit ...	0	1

03. Error Analysis of BERT's Performance

Next, we identified three specific data points for analysis: 46987, 2297, and 8158, which were flagged due to aggressive language, hate-associated terms, or misinterpretation of nuanced phrases.

Index	Text	Label	Predicted
46987	"I'd jam with you if you ever kicked some nobodies off your damn friends list lol"	0	1
8158	That's what sucks: I don't actually get to do it. I just had to prove I could plan it out. -KayCee	0	1
2297	.... He became notorious on campus for wearing a Nazi uniform, Ku Klux Klan apparel, and swastika paraphernalia while picketing and holding parties on the anniversary of the birth of Adolf Hitler. The year of his graduation, he was elected Grand Wizard of the Knights of the Ku Klux Klan. ...(skip)	0	1

03. Error Analysis of BERT's Performance

Next, we identified three specific data points for analysis: 46987, 2297, and 8158, which were flagged due to aggressive language, hate-associated terms, or misinterpretation of nuanced phrases.

Index	Analyze
46987	The model likely flagged this as bullying due to aggressive-sounding words like "kicked," "damn," and "nobodies."
8158	The model might have misinterpreted the phrase "that's what sucks" as negative or hostile language.
2297	The text contains terms often associated with hate speech or bullying, such as "Nazism," "swastika," and "Ku Klux Klan." The model likely flagged these terms as bullying without accounting for the historical and informational context of the passage.

04. Final summary

Conclusion: BERT excels at handling nuanced text, demonstrating strong performance. Incorporating TF-IDF insights and refining the dataset could further enhance its reliability.

- **BERT's Performance**
- **TF-IDF Insights**
- **Challenging Data**
- **Dataset Noise**

04. Final summary

Conclusion: BERT excels at handling nuanced text, demonstrating strong performance. Incorporating TF-IDF insights and refining the dataset could further enhance its reliability.

BERT's Performance

Challenging Data

TF-IDF Insights

Dataset Noise

- Achieved an F1 score close to 1.0, showcasing its ability to balance Precision and Recall.
- Highly suitable for tasks with high costs of false negatives, such as detecting bullying.

04. Final summary

Conclusion: BERT excels at handling nuanced text, demonstrating strong performance. Incorporating TF-IDF insights and refining the dataset could further enhance its reliability.

BERT's Performance

Challenging Data

TF-IDF Insights

Dataset Noise

- Difficult-to-classify examples included sarcastic comments, mixed-language text, and ambiguous phrases with subtle undertones (e.g., "you're amazing, no really...", "nice try, buddy").
- BERT's higher Recall score highlights its strength in capturing these subtleties.

04. Final summary

Conclusion: BERT excels at handling nuanced text, demonstrating strong performance. Incorporating TF-IDF insights and refining the dataset could further enhance its reliability.

- BERT's Performance
- Challenging Data
- **TF-IDF Insights**
- Dataset Noise

TF-IDF analysis highlighted the importance of specific terms in distinguishing bullying from non-bullying comments.

04. Final summary

Conclusion: BERT excels at handling nuanced text, demonstrating strong performance. Incorporating TF-IDF insights and refining the dataset could further enhance its reliability.

BERT's Performance

Challenging Data

TF-IDF Insights

Dataset Noise

- Presence of highly specific words (e.g., nicknames or repetitive terms from web-scraped data) adds noise.
- While partial cleansing was done, fully cleaning the dataset would require substantially more time, representing a potential weakness of the model.

Thank you!

Any Questions?