# CS412 – Machine Learning Spring 2025 Project Report

**Group Members:**

- Alper Çamlı 30858
- Korhan Erdoğdu 30838
- Merve Bilgi 29117
- Deniz Gürleyen 30919
- Nazlı Dilanur Keleş 31313

**Colab Notebook Link:**

- **Logistic Regression:** CS412_LogReg_TFIDF.ipynb
- **Naive Bayes:** CS412_NaiveBayes_TFIDF.ipynb
- **ANN:** CS412_Group36_ANN.ipynb
- **BERT:** CS412_Group36_BERT.ipynb

# Table of Contents

## 1. Introduction

This project addresses the task of binary sentiment classification using the IMDB movie reviews dataset. Our objective is to predict whether a given movie review expresses a positive or negative sentiment. We explored and evaluated four models: Naive Bayes, Logistic Regression, Artificial Neural Networks (ANN), and BERT-based transformers. These models span from traditional machine learning techniques to deep learning and pre-trained transformer architectures.

Our main goal was to compare the performance of these approaches using macro F1 score as the primary evaluation metric, supported by additional metrics like accuracy, precision, and recall.

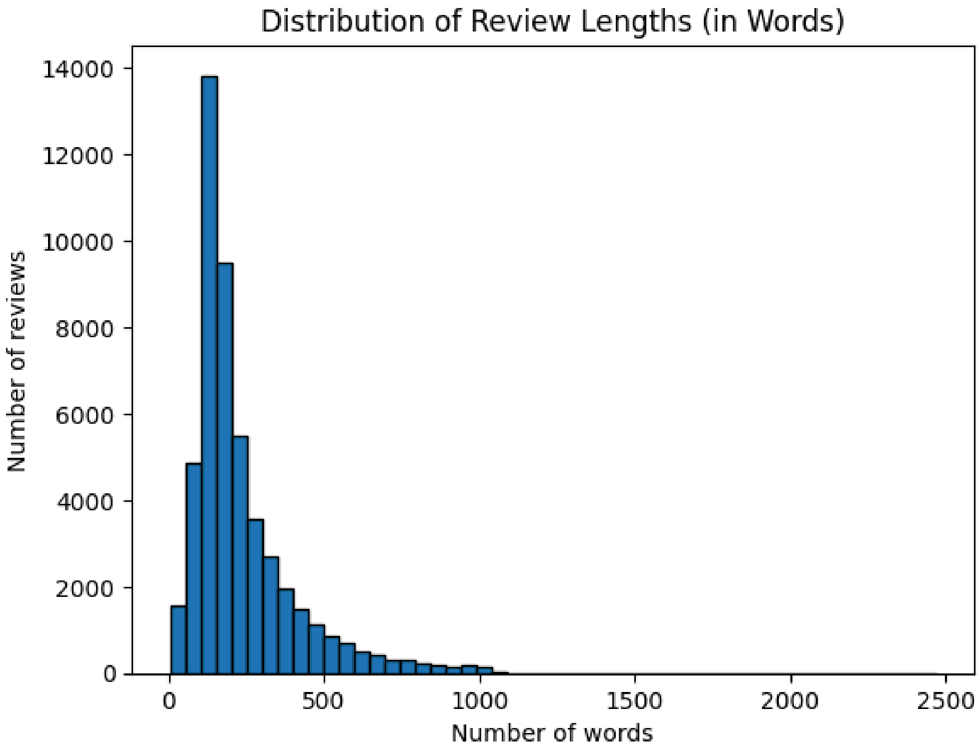
## 2. Problem Description

This project is formulated as a **binary classification problem**, where each movie review must be labeled as either **positive** or **negative**. The IMDB dataset contains 50,000 labeled reviews (25,000 positive and 25,000 negative), and the task involves learning a mapping from review text to sentiment class using machine learning algorithms.

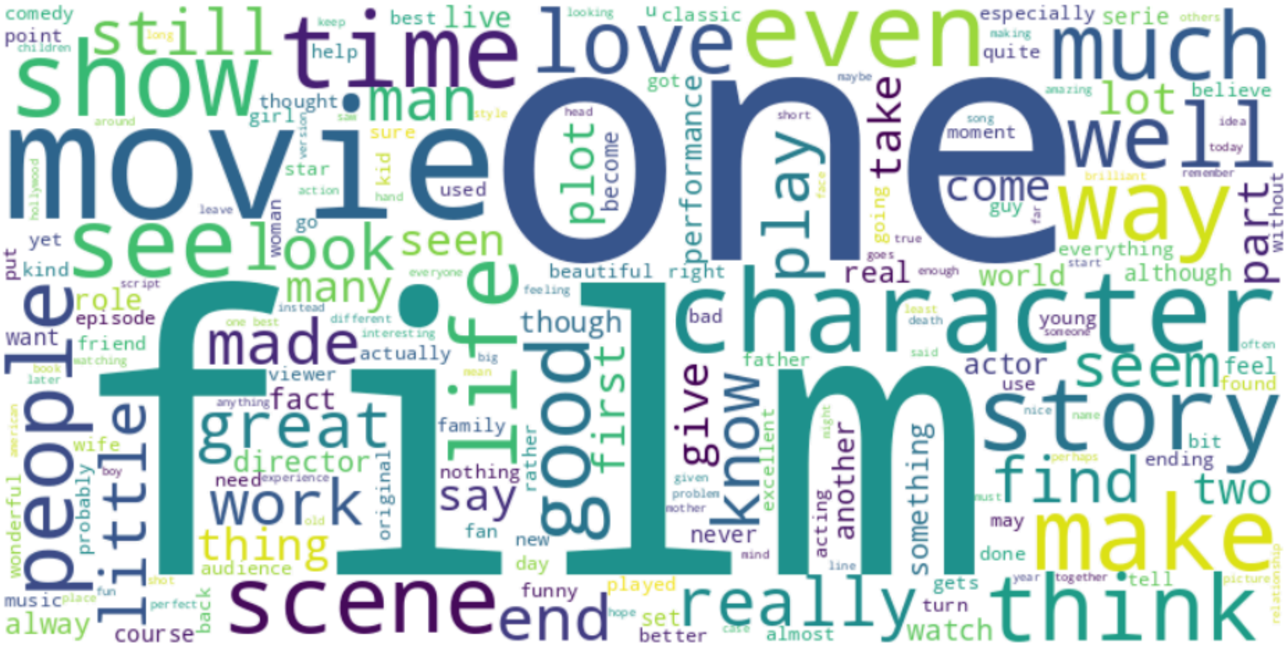
## 3. Methods

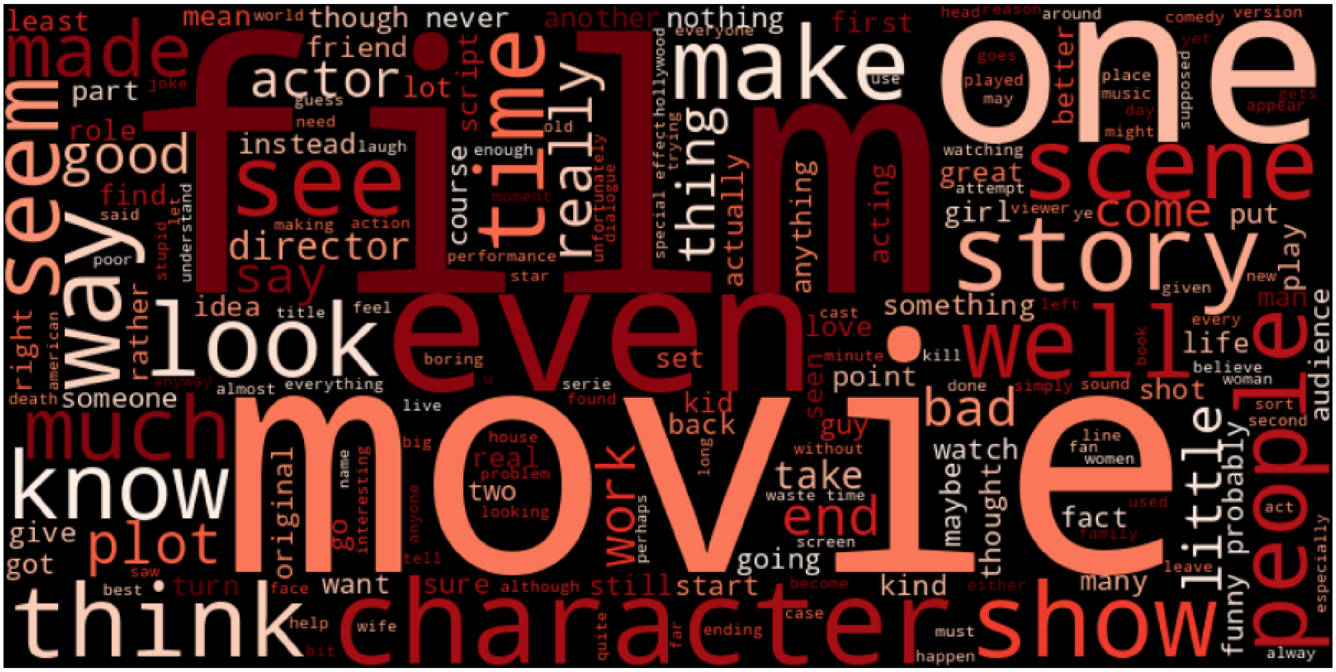
### 3.1 Statistical Analysis of the Dataset

- Number of Total Reviews: 50,000
- Class distribution: 50% positive, 50% negative which is balanced
- Average review length: **231.15** words with the standard deviation of **171.34** words, which is very high and, here is the Distribution of Review Lengths in Words:

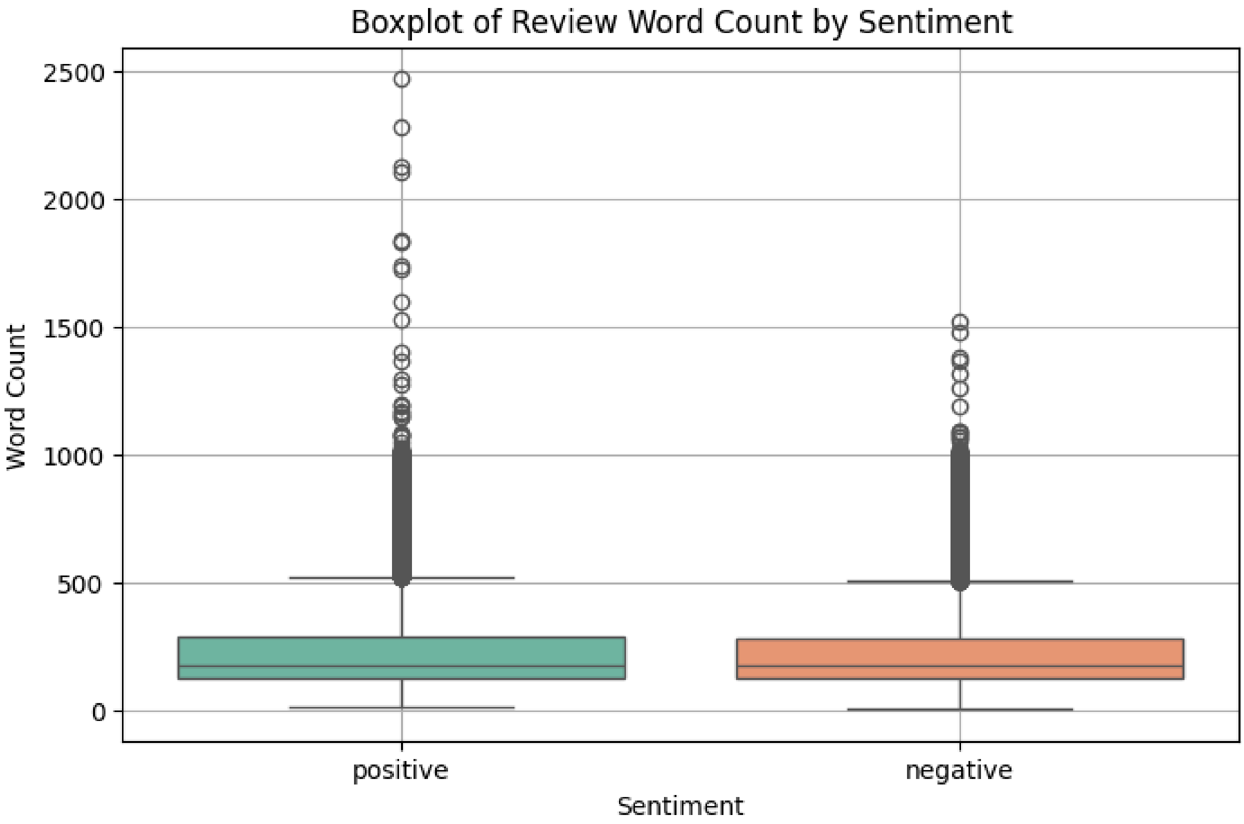


- Some of the most used words; one, film, movie, even, well, character, story, people, show, time, think, etc. Additionally a word cloud below graph for each class:


WordCloud - Positive Reviews


WordCloud - Negative Reviews

- Maximum review length observed is **2470** words and minimum is **4** words. Also the bar graph below shows that anomalistic amount of words used more often in positive reviews compared to negative reviews. However, on average positive reviews have only 3.2 words more than negative.



### 3.2 Preprocessing Steps

- **Naive Bayes & Logistic Regression**: Text lower-cased, HTML tags removed, punctuation stripped, vectorized.

- **ANN**: Preprocessed similarly, followed by tokenization and padding with zeros using Keras. Max vocabulary size limited by 10000 unique words, and max length of a review in words limited by 300 to eliminate exceptional cases. In more detail, a Keras Tokenizer was used to convert text into integer sequences. The vocabulary was limited to the 10,000 most frequent words (MAX_VOCAB=10000) to reduce memory usage and avoid overfitting. An out-of-vocabulary token (<OOV>) was introduced to handle unseen words.

  After tokenization, the resulting sequences were padded to a fixed length of 300 tokens (MAX_LEN=300) using post-padding and truncation. This ensured uniform input dimensions for the ANN model.

- **BERT**: Similarly data is cleaned.Then, used Hugging Face’s **bert-base-uncased** tokenizer with padding and truncation.

### 3.3 Description of Models and Hyperparameters

- **Naive Bayes**:
    - A simple and fast probabilistic model used for text classification.
    - Type: MultinomialNB
    - Input: TF-IDF vectors

- **Logistic Regression**:
    - Solver: **liblinear**
    - Input: TF-IDF vectors
    - C = 2.0, max_iter = 200

- **ANN**:
    - Architecture: Embedding → Dense(256) → Dropout(0.5) → Dense(128) → Dropout(0.4)→ Dense(64) → Dropout(0.3) → Dense(1, sigmoid)
    - Optimizer: Adam (learning rate = 0.0005)
    - Loss: Binary cross-entropy
    - Epochs: 15, Batch size: 128

- **BERT**:
    - Model: **bert-base-uncased**
    - Fine-tuning: Added classification head with dropout and linear layer
    - Optimizer: AdamW
    - Epochs: 3, Batch size: 8

## 4. Results and Discussion

| Model | Accuracy | Precision | Recall | F1 Score (Macro) |
|---|---|---|---|---|
| Naive Bayes | 0.87 | 0.87 | 0.87 | 0.87 |
| Logistic Regression | 0.90 | 0.90 | 0.90 | 0.90 |
| ANN | 0.86 | 0.87 | 0.86 | 0.86 |
| BERT | 0.931 | 0.931 | 0.931 | 0.931 |

Our experimental results highlight a clear trend: while all models performed reasonably well on the IMDB sentiment classification task, models with deeper semantic understanding of the text generally achieved higher performance.

- **Naive Bayes** served as a simple yet surprisingly effective baseline, achieving an F1 score of **0.87**. This model works well when strong assumptions (e.g., feature independence) roughly hold true. However, its inability to model word order or contextual nuances limits its effectiveness for complex sentence structures.

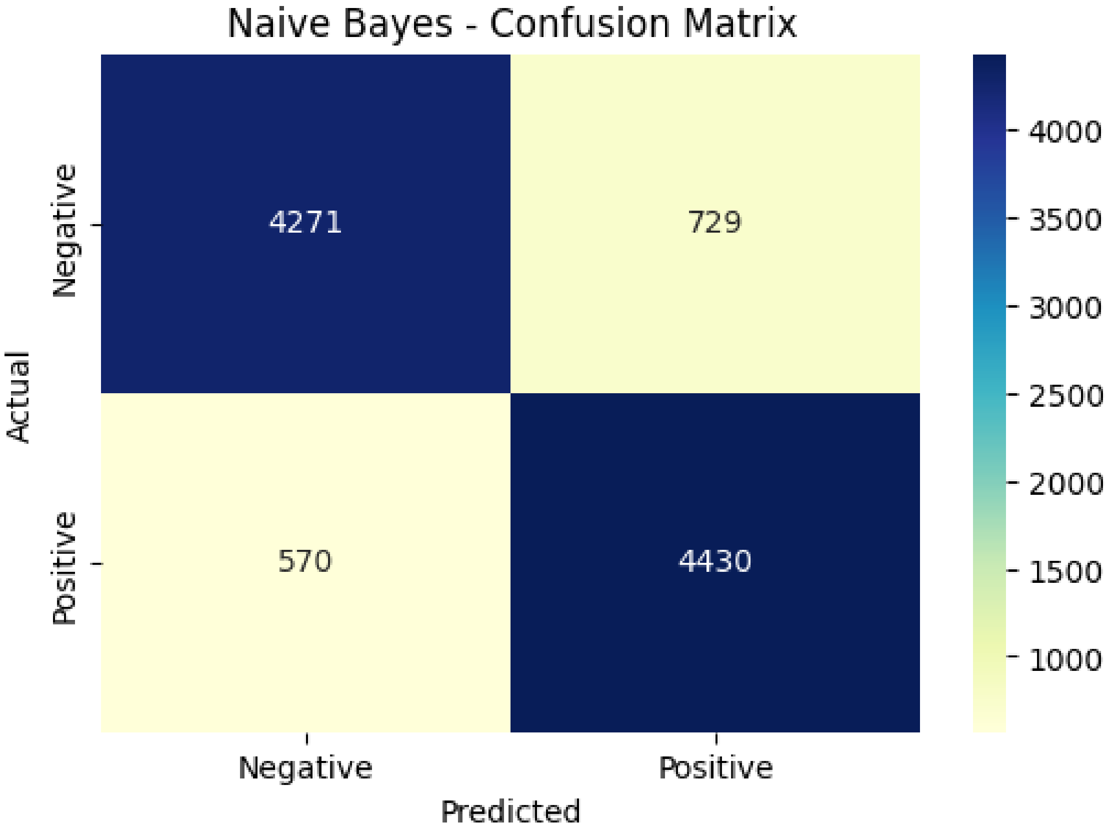
- **Logistic Regression** outperformed Naive Bayes slightly across all metrics, with an F1 score of **0.90**. Its performance improvement can be attributed to better handling of linear relationships in TF-IDF features. It also benefited from the high-dimensional sparse feature representation that captures word frequency patterns without making naive independence assumptions.

- **ANN** delivered an F1 score of **0.86**, which was slightly lower than both classical ML models. While it was able to capture some non-linear relationships in the data, its performance was likely impacted by its inability to capture long-range dependencies in text and reliance on word-level embeddings without contextual information. This suggests that simple feedforward architectures may not be sufficient for tasks requiring deeper language understanding.

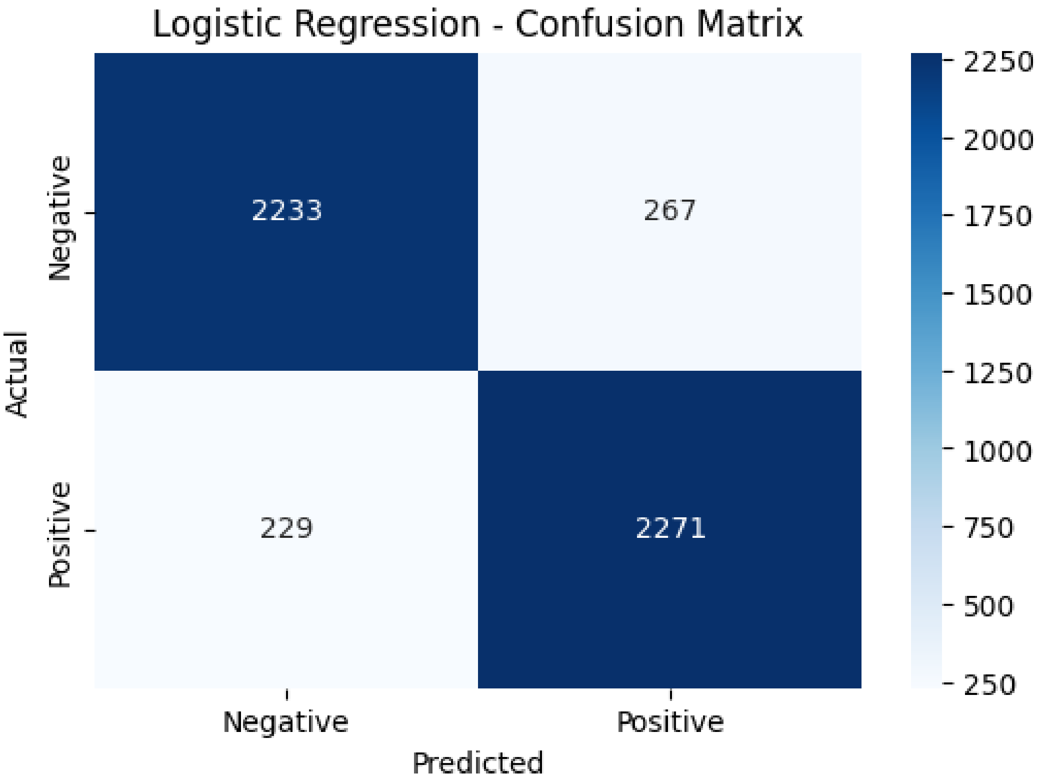
- **BERT** clearly outperformed all other models, achieving the highest F1 score of **0.931**. It,s superior performance is due to its pre-training on large corpora and its ability to model deep contextual representations. BERT processes the full sequence of tokens bidirectionally, which allows it to capture semantic and syntactic information that traditional models and shallow neural networks miss. However, this came at the cost of longer training times and greater computational resource requirements. We tried 2 versions: 2 epochs and 3 epochs. They both have similar results but the 3 epoch training results with a much higher train accuracy and a slightly higher validation accuracy compared to 2 epoch. But the difference between train and validation accuracy is so much more with 3 epochs so it results in slight overfitting. 2 epochs is recommended for future implementations.

**Confusion Matrices**

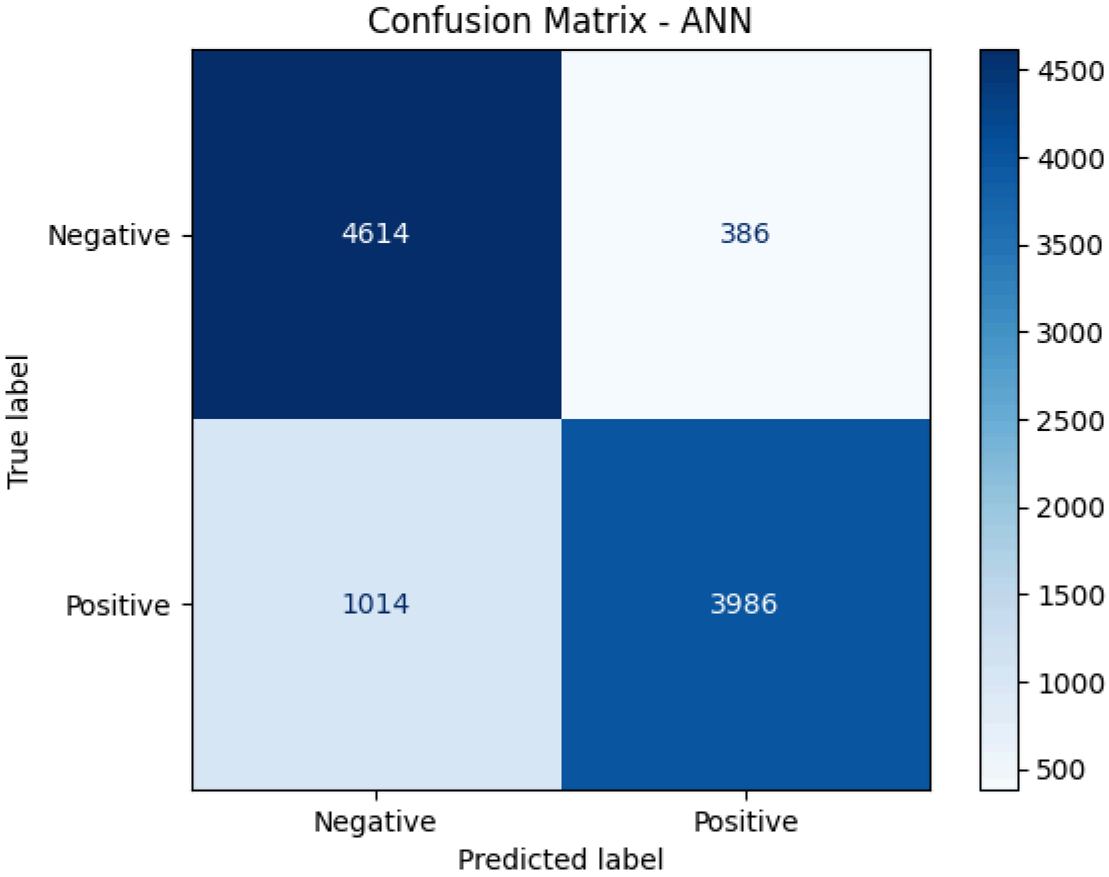
- **Naive Bayes**



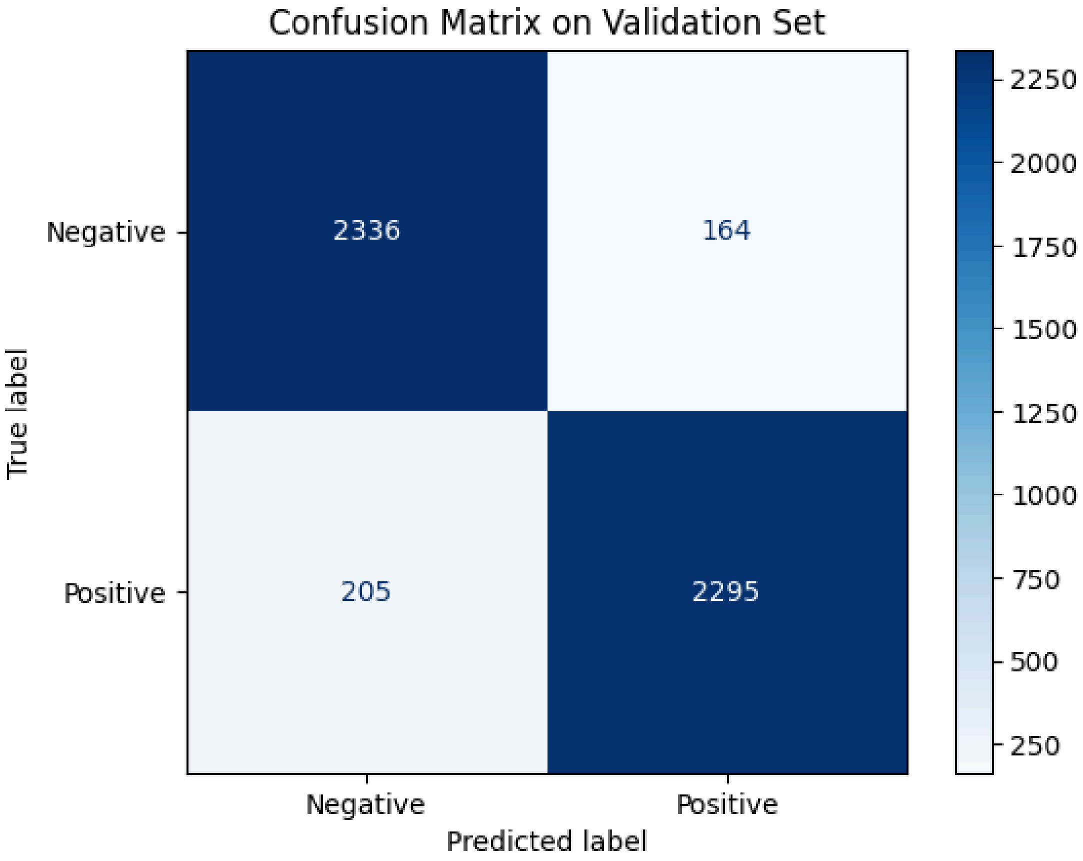
- **Logistic Regression**



- **ANN**

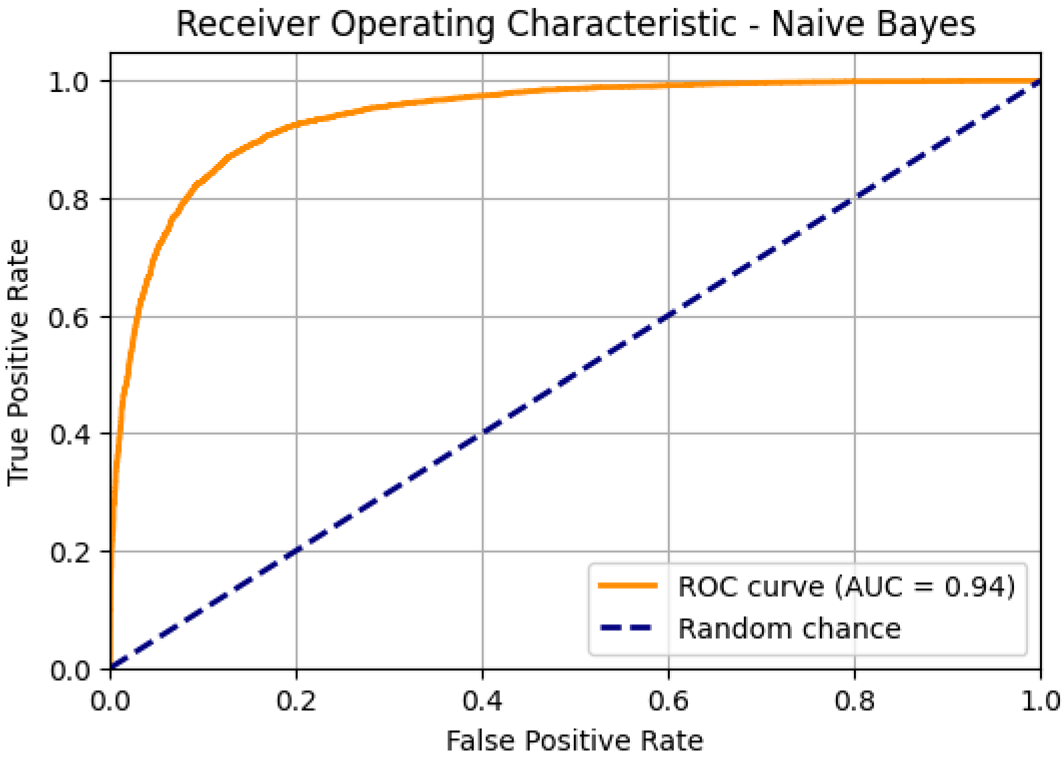

Confusion Matrix - ANN
True label
Negative
Positive
4614
386
1014
3986
Predicted label
Negative
Positive
4500
4000
3500
3000
2500
2000
1500
1000
500

- **BERT**

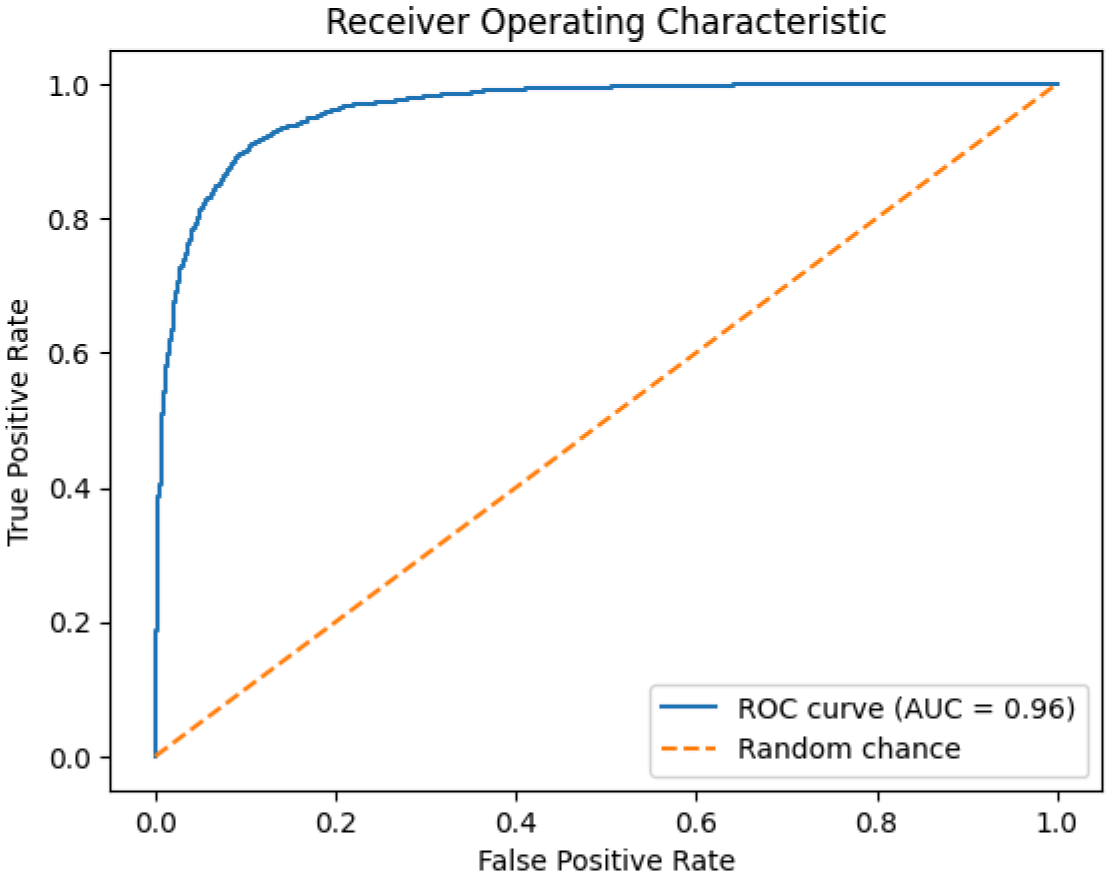

Confusion Matrix on Validation Set
True label
Negative
Positive
2336
164
205
2295
Negative
Positive
Predicted label
2250
2000
1750
1500
1250
1000
750
500
250

**Precision-Recall Curves**

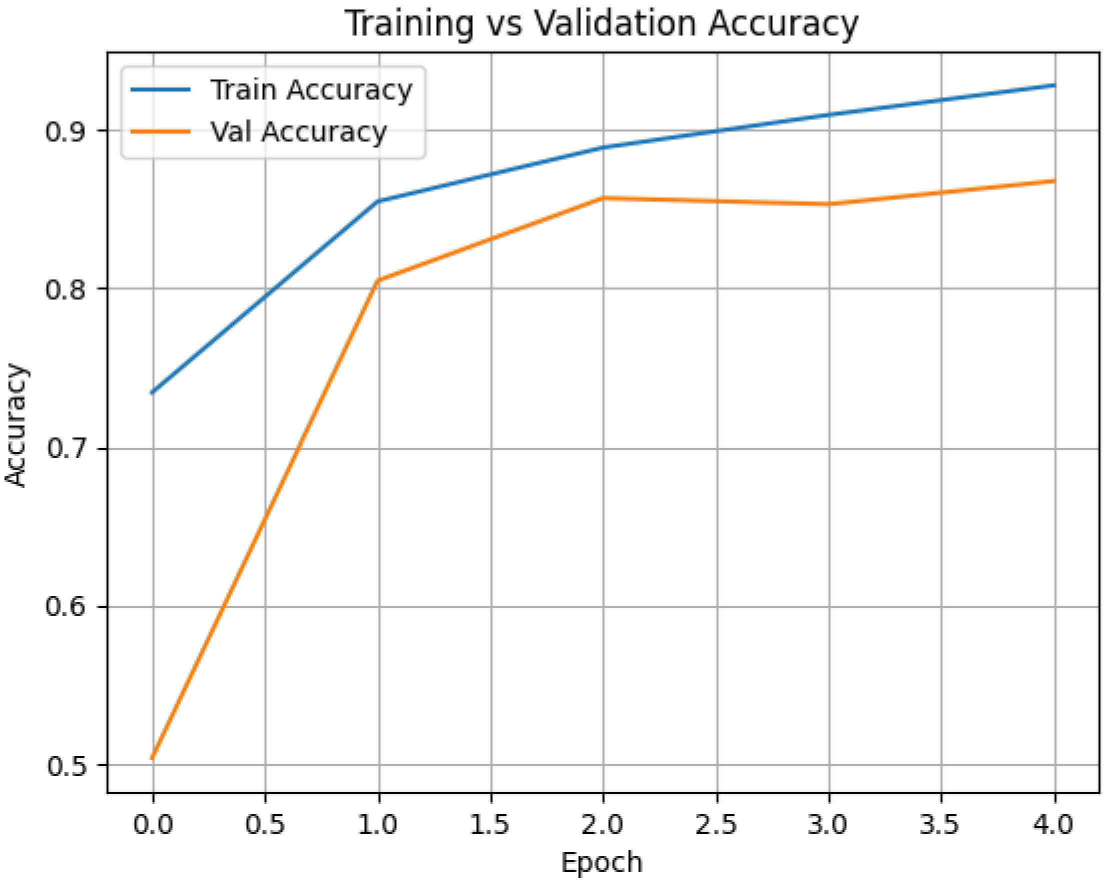
- **Naive Bayes**

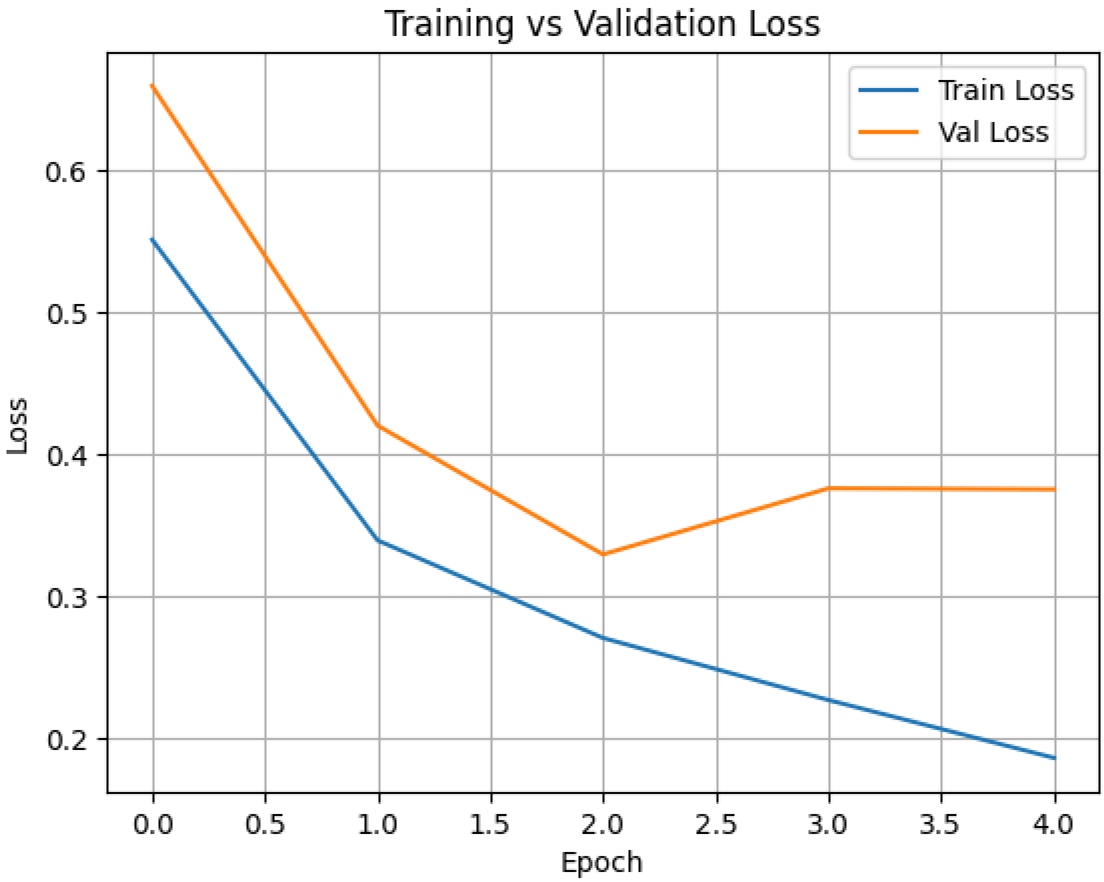


- **Logistic Regression**

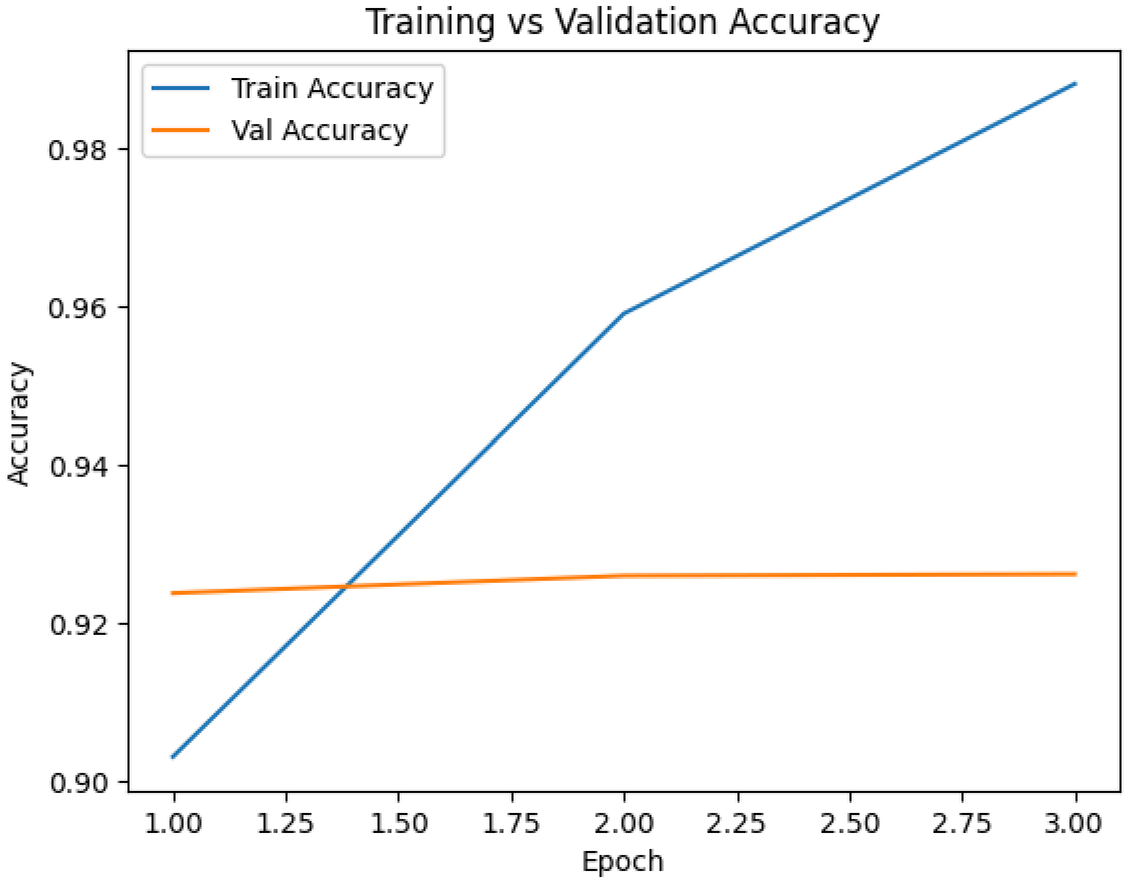


- **ANN**


Training vs Validation Accuracy
Train Accuracy
Val Accuracy
Accuracy
0.5
0.6
0.7
0.8
0.9
Epoch
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


Training vs Validation Loss
Train Loss
Val Loss
Loss
0.6
0.5
0.4
0.3
0.2
Epoch
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

- **BERT**



## 5. Conclusion

This project explored binary sentiment classification using four different models: Naive Bayes, Logistic Regression, Artificial Neural Networks (ANN), and BERT. Each model demonstrated varying capabilities in capturing sentiment from textual data, highlighting the trade-offs between interpretability, computational efficiency, and classification performance.

The **Naive Bayes** model served as a simple and effective baseline, performing well given its assumptions, but ultimately limited by its inability to model contextual dependencies. **Logistic Regression** improved upon this by effectively leveraging high-dimensional TF-IDF representations, making it a strong candidate for tasks requiring quick deployment and low computational cost.

The **ANN** model introduced non-linearity and offered greater flexibility in learning patterns from token sequences. However, its architecture lacked the depth to handle complex syntactic and semantic structures, which likely contributed to its slightly lower performance compared to traditional models.

**BERT**, a transformer-based model fine-tuned for our task, achieved the highest performance across all evaluation metrics. Its contextualized word representations and deep bidirectional encoding allowed it to capture the intricacies of sentiment far more effectively. Despite requiring significant computational resources, BERT proved to be the most powerful model for this task.

Overall, the results reinforce the value of using **pretrained transformer models like BERT** for nuanced natural language understanding tasks. However, for resource-constrained environments or when interpretability is critical, **Logistic Regression** offers a highly competitive alternative.

## 6. Appendix

**Group Contributions:**

- Merve: I implemented both the Naive Bayes and Logistic Regression models used for sentiment classification in this project. I handled all steps of the TF-IDF-based preprocessing, including cleaning the text data (lowercasing, removing punctuation, and stopwords) and transforming the reviews into numerical feature vectors suitable for machine learning. For both model, I managed the setup and training process using Scikit-learn, ensured proper stratified train-test splitting, and tuned key parameters to improve performance. I also conducted evaluations using metrics like accuracy, macro F1-score, and confusion matrices. To make the results more interpretable, I generated visualizations such as ROC curves and confusion matrix heatmaps. Additionally, I ensured that the Jupyter notebooks remained well-structured and readable, in line with the CS412 project requirements.

- Korhan: Korhan was responsible for the complete implementation of the Artificial Neural Network (ANN) model used in this project. His work began with exploratory data analysis, where he examined the class distribution of sentiments, analyzed review lengths, and generated visualizations such as histograms and word clouds to better understand the structure and characteristics of the IMDB dataset. Following this, Korhan carried out all the necessary preprocessing steps, including cleaning the text (removing HTML tags, punctuation, and lowercasing), tokenizing the reviews, converting them into fixed-length padded sequences, and encoding the sentiment labels for binary classification. He then designed and implemented both the baseline and an improved ANN architecture. The improved model included several enhancements to improve performance and prevent overfitting, such as the use of dropout layers, batch normalization, L2 regularization, and multiple dense layers with ReLU activation. Korhan also fine-tuned hyperparameters such as learning rate, batch size, and optimizer configuration to maximize validation accuracy. In addition to his technical contributions, Korhan also wrote several parts of the final report, particularly those detailing the ANN model design, implementation process, and evaluation results.

- Deniz was responsible for the implementation and fine-tuning of the BERT model for sentiment analysis using the Hugging Face Transformers library. This involved preprocessing and cleaning the IMDB dataset, conducting exploratory data analysis, and performing a detailed statistical summary of the data. Deniz designed and applied the train/validation/test split to ensure robust and unbiased model evaluation. Throughout the training process, Deniz tracked and visualized training and validation accuracies, created relevant performance plots, and evaluated the model's performance using classification reports and confusion matrices for both the validation and test sets. Additionally, Deniz contributed by writing and organizing the Results section of the project report.

- Alper/Nazlı: Wrote the report and helped with the planning.

**Supplementary Material:**

- Tokenizers: Keras Tokenizer, BERT Tokenizer (transformers)
- Hardware: [Colab GPU / Local Machine]