Comprehensive Evaluation of Sentiment Analysis Classification Methods

Aaditya Bansal, Anmol Yadav, Om Sharma, Meet Titala, Rudra Gupta

Department of Computer Science and Engineering, IIT Jodhpur
{b23cs1083, b23cs1004, b23cs1048, b23cs1036, b23cs1098}@iitj.ac.in

Abstract

This report presents a comprehensive comparison of various machine learning approaches for sentiment analysis classification. We evaluate traditional algorithms including Naive Bayes, Decision Trees, Random Forests, Support Vector Machines, and K-Nearest Neighbors, along with more advanced techniques such as Neural Networks and unsupervised clustering methods. Each approach is assessed using both Bag-of-Words (BoW) and TF-IDF feature extraction methods. We analyze accuracy, precision, recall, F1-scores, and the impact of key hyperparameters on model performance. We also identify that the dataset's class imbalance significantly impacts model performance, particularly for minority classes. This compilation serves as a reference for selecting appropriate sentiment analysis methods based on specific requirements.

Keywords: sentiment analysis, text classification, machine learning, feature extraction, natural language processing

Contents

1 Introduction

Sentiment analysis, the task of classifying text according to the expressed sentiment, has become increasingly important in natural language processing. This report compiles findings from various approaches to sentiment classification, focusing on tweets and similar short text formats.

The classification task involves categorizing text into sentiment classes (typically positive, negative, and neutral). This presents several challenges, including handling linguistic nuances, dealing with class imbalance, and selecting appropriate feature representations.

Through this report, we compare traditional machine learning approaches (Naive Bayes, Decision Trees, Random Forests, SVM, KNN) and complex models (ANNs). We also assess the impact of different text representation methods (primarily Bag-of-Words and TF-IDF) and dimensionality reduction techniques like SVD on model performance.

The structure of this report is as follows:

- Section 2 describes the approaches tested
- Section 3 details experiments and results for each method
- Section 4 provides a comparative analysis across all methods
- Section 5 summarizes key findings and best practices

2 Approaches Tried

We explored various approaches to sentiment classification, each with distinct characteristics and performance profiles. The following subsections briefly outline each method.

2.1 Text Representation Methods

Two primary text vectorization techniques were employed across all models:

1. **Bag of Words (BoW)**: A simple representation that counts word occurrences in documents, disregarding grammar and word order but retaining multiplicity.
2. **Term Frequency-Inverse Document Frequency (TF-IDF)**: A numerical statistic reflecting the importance of a word in a document relative to a corpus, combining term frequency with inverse document frequency to reduce the impact of common words.

Additionally, we explored dimensionality reduction techniques including Singular Value Decomposition (SVD) and feature selection using methods like SelectKBest with chi-square test.

2.2 Classification Algorithms

2.2.1 Naive Bayes

A probabilistic classifier based on applying Bayes' theorem with strong independence assumptions between features. We implemented both:

- Custom Naive Bayes with token-level log-likelihoods
- Multinomial Naive Bayes with TF-IDF features

2.2.2 Linear Regression

A regression-based approach applied to sentiment classification by predicting numerical sentiment labels. We implemented:

- Linear Regression trained on TF-IDF-transformed textual data
- Predictions rounded and clipped to the nearest valid class to simulate classification behavior

2.2.3 Decision Trees and Random Forests

Decision Trees create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. Random Forests extend this by creating an ensemble of decision trees, typically trained via bagging methods.

2.2.4 Support Vector Machines (SVM)

SVMs find the hyperplane that best separates classes in the feature space. We explored different regularization parameters and assessed performance with both BoW and TF-IDF features.

2.2.5 K-Nearest Neighbors (KNN)

A non-parametric method where the input consists of the k closest training examples in the feature space. We tested various distance metrics (Euclidean and Manhattan) and values of k.

2.2.6 Logistic Regression

A statistical model that uses a logistic function to model a binary dependent variable. We extended this to multi-class classification using one-vs-rest strategy.

2.2.7 Artificial Neural Networks (ANN)

We implemented both shallow and deep neural network architectures for comparison, with proper preprocessing and dimensionality reduction.

2.2.8 Unsupervised Clustering

We explored clustering algorithms including K-Means, Agglomerative Clustering, and DBSCAN to identify natural groupings in the data without predetermined labels.

3 Experiments and Results

3.1 Dataset Characteristics

The dataset consists of tweets labeled as positive, negative, or neutral. A significant class imbalance was observed:

- Positive: approximately 77%
- Negative: approximately 23%
- Neutral: approximately 0.098%

Data Analysis Visualizations

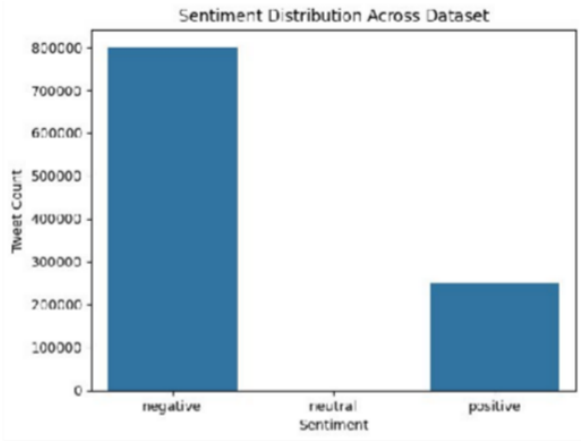


Figure 1: overall distribution after cleaning.

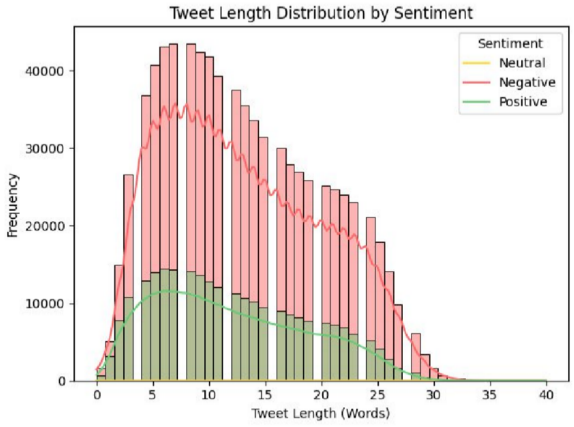


Figure 2: Tweet Length Analysis.

Figure 1: Visualizations related to tweet dataset .

This imbalance affected performance across all models, particularly for the neutral class.

3.2 Naive Bayes Results

3.2.1 Custom Naive Bayes Implementation

The custom implementation using token frequency achieved:

- Validation Accuracy: 81.34%
- Test Accuracy: 81.46%

Class-specific F1-scores:

- Negative: 0.88
- Positive: 0.53
- Neutral: 0.00

3.2.2 TF-IDF with Multinomial Naive Bayes

The TF-IDF implementation achieved:

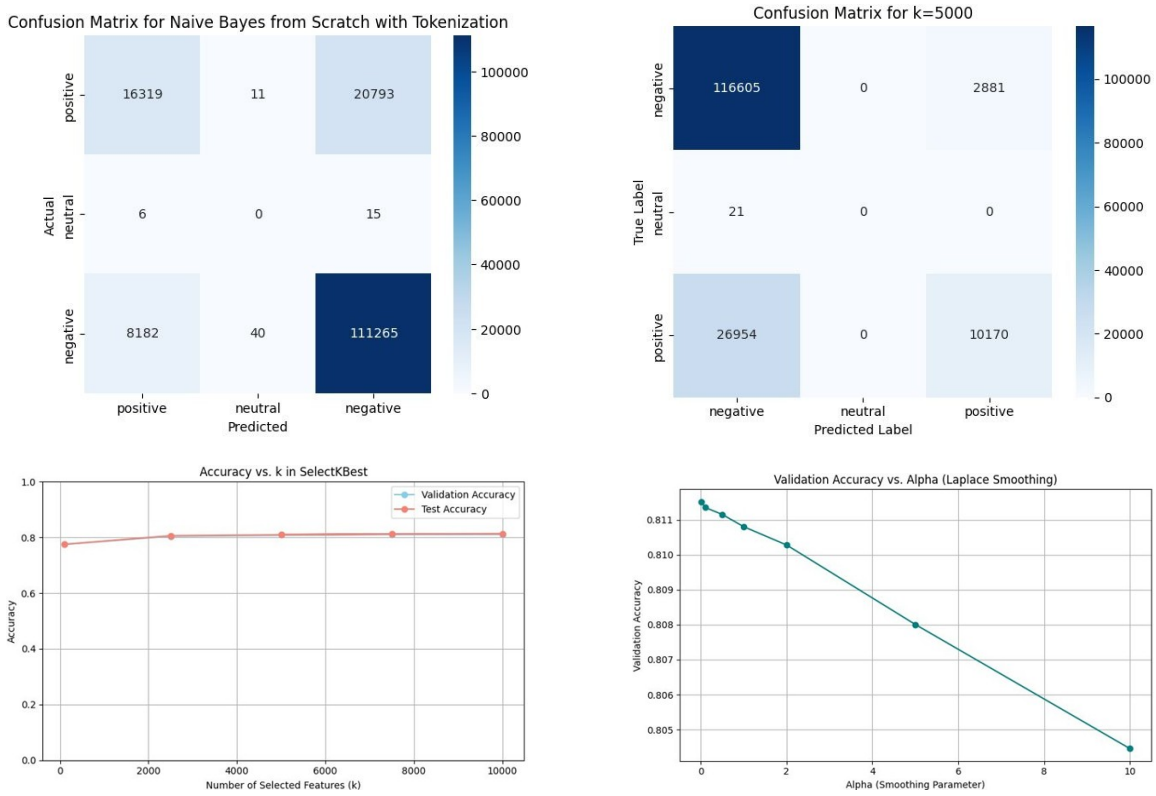
- Validation Accuracy: 81.08%
- Test Accuracy: 81.33%

Hyperparameter tuning showed alpha=1 and n-gram range=2 as optimal settings.

3.2.3 Feature Selection with SelectKBest

- k=100: 77.55% test accuracy
- k=2500: 80.63% test accuracy
- k=5000: 81.11% test accuracy
- k=7500: 81.30% test accuracy
- k=10000: 81.33% test accuracy

Visualizations for Naive Bayes



Experiments with SelectKBest showed that k=7500 provided optimal accuracy

3.3 Linear Regression Results

The Linear Regression model was applied using TF-IDF features and label-encoded sentiment classes. Since regression models predict continuous values, predictions were rounded and clipped to the nearest valid sentiment class:

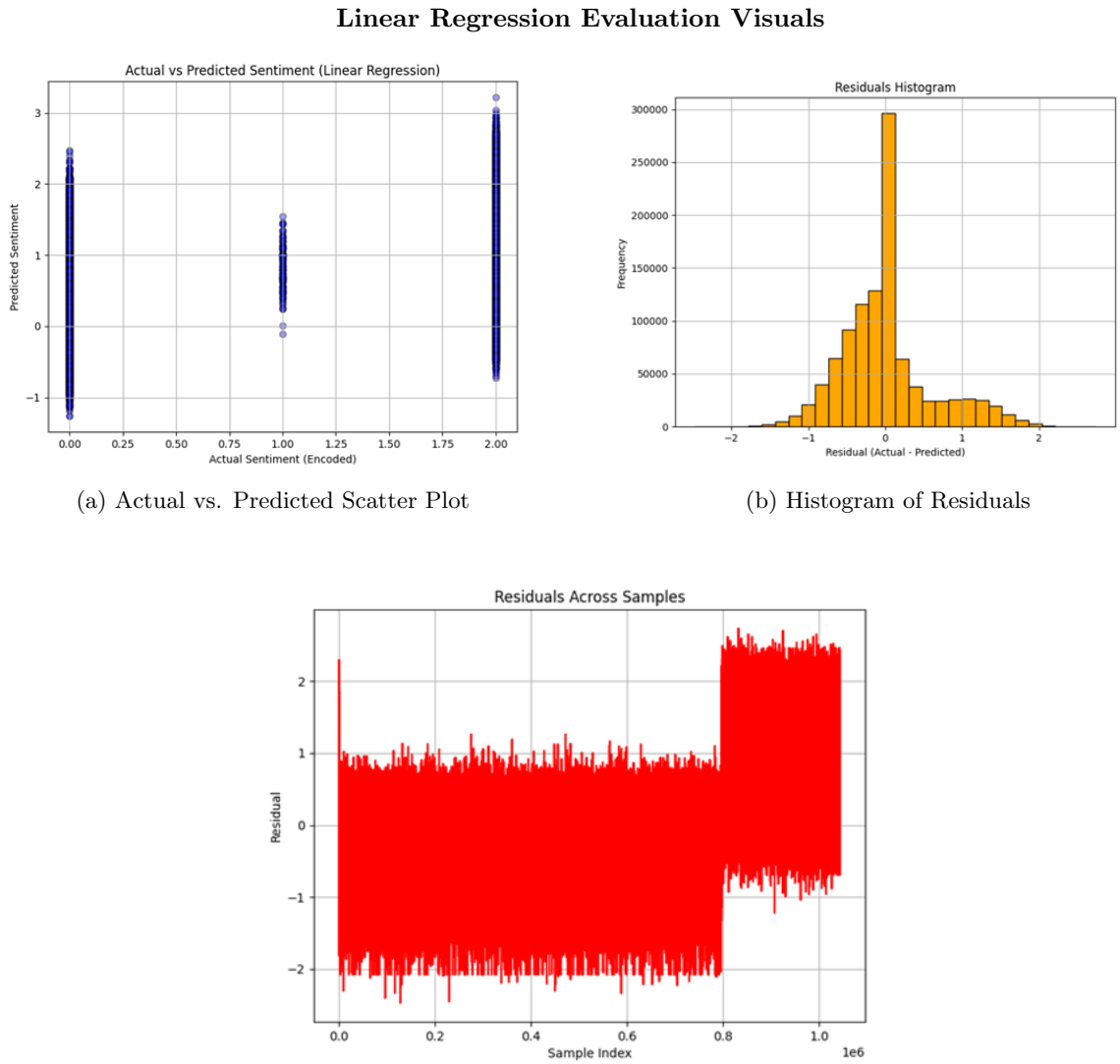
```
0 = Negative, 1 = Neutral, 2 = Positive
```

A standard Linear Regression model was trained using the encoded features and evaluated as a classifier by post-processing the outputs.

3.3.1 Performance Metrics

- **Mean Squared Error (MSE):** ≈ 0.3496
- **R^2 Score:** ≈ 0.5167
- **Approximate Classification Accuracy:** $\approx 68.42\%$

3.3.2 Visualizations



(a) Actual vs. Predicted Scatter Plot

(b) Histogram of Residuals

(c) Line Plot of Residuals

Figure 3: Visualizations for Linear Regression model performance on sentiment classification.

3.4 Decision Tree and Random Forest Results

3.4.1 Without SVD

Decision Tree with TF-IDF Features

- Accuracy: 77.19%

Decision Tree with BoW Features

- Accuracy: 77.06%

Random Forest with TF-IDF Features

- Accuracy: 82.01%

Random Forest with BoW Features

- Accuracy: 81.57%

3.4.2 With SVD

Decision Tree with TF-IDF Features

- Accuracy: 57.69%

Decision Tree with BoW Features

- Accuracy: 57.69%

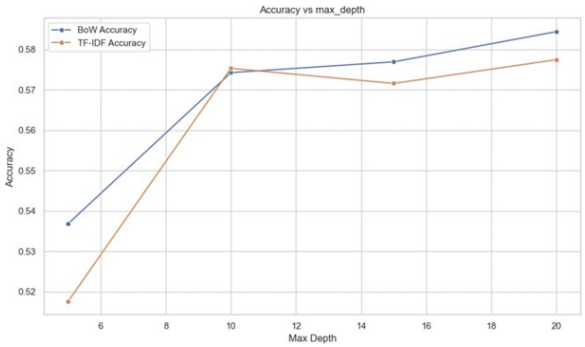
Random Forest with TF-IDF Features

- Accuracy: 57.69%

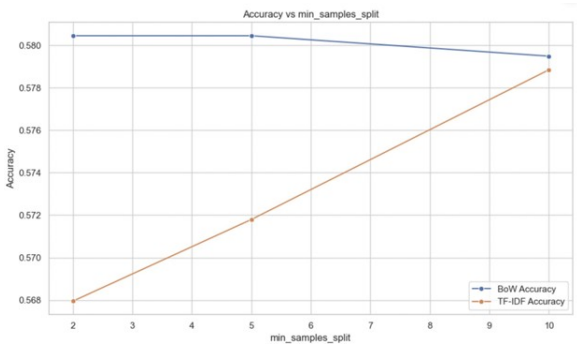
Random Forest with BoW Features

- Accuracy: 68.27%

Accuracy vs. Max Depth Accuracy initially increases with depth for both BoW and TF-IDF. BoW consistently outperforms TF-IDF across all depth values. Beyond a depth of 10, both lines plateau, with BoW still slightly ahead.



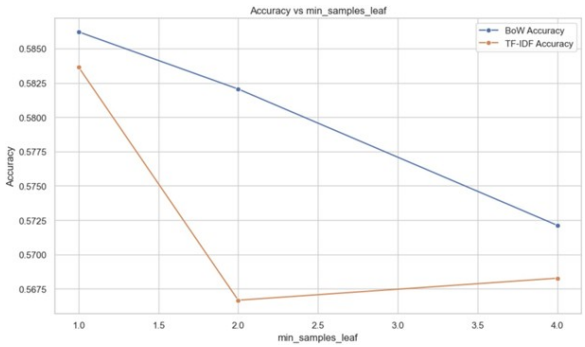
(a) Accuracy vs. Max Depth



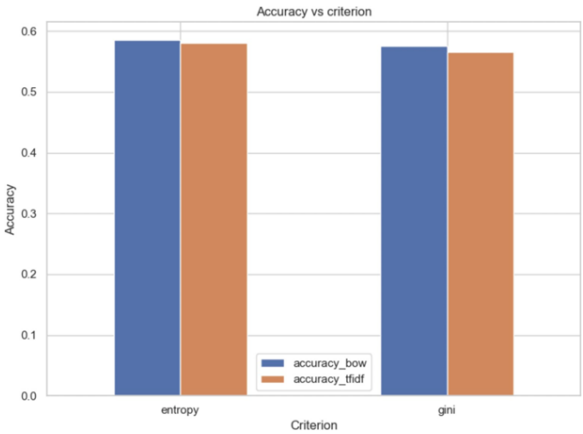
(b) Accuracy vs. Min Samples Split

Accuracy vs. Min Samples Split BoW starts with higher accuracy and remains stable. TF-IDF starts low but improves with increased `min_samples_split`. Small splits increase model complexity and risk overfitting. BoW performs well at small splits; TF-IDF improves with regularization.

Accuracy vs. Min Samples Leaf BoW peaks at `min_samples_leaf = 1` and declines steadily. TF-IDF shows a sharp drop at 2, then stabilizes at a lower level. BoW thrives with minimal leaf constraints due to frequent, strong features. TF-IDF over-regularizes easily, losing performance.

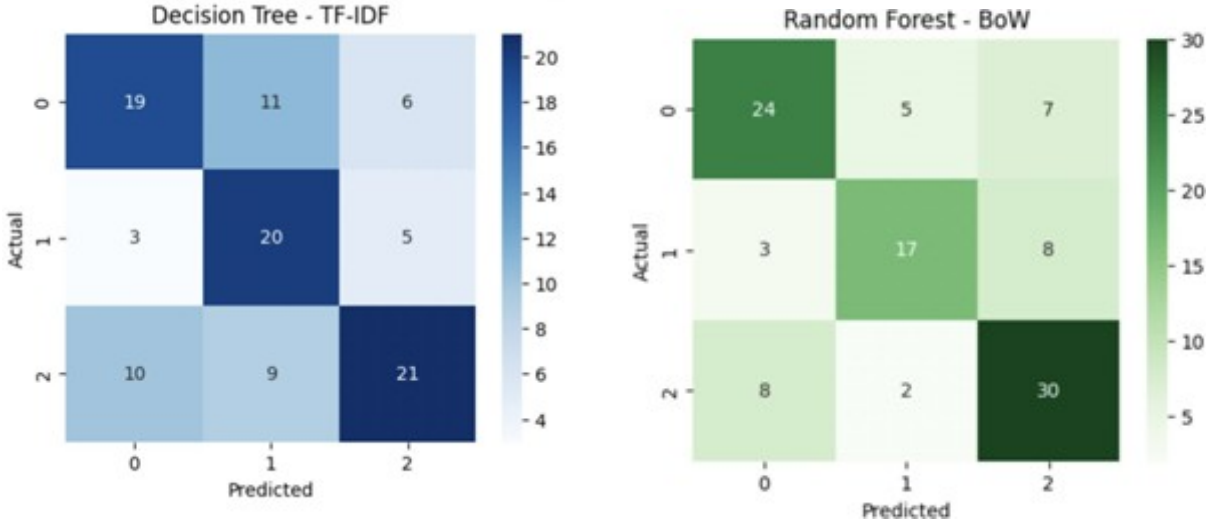


(a) Accuracy vs. Min Samples Leaf



(b) Accuracy vs. Criterion

Accuracy vs. Criterion (Entropy vs. Gini) Entropy consistently outperforms Gini across both feature sets. BoW outperforms TF-IDF for both criteria. Entropy captures subtle splits in sparse data better. Gini is faster but slightly less precise.



(a) Decision Tree Confusion Matrix

(b) Random Forest Confusion Matrix (BoW)

Figure 6: Additional Visualizations for Evaluation

3.5 Support Vector Machine Results

3.5.1 SVM with BoW

The SVM with BoW features achieved:

- Accuracy: 0.596
- Class-specific performance:
  - Class 0: 44.4% correct (16/36)
  - Class 1: 71.4% correct (20/28)
  - Class 2: 65.0% correct (26/40)
- Precision: 0.42 to 0.80
- Recall: 0.44 to 0.71
- F1-scores: 0.53 to 0.68

3.5.2 SVM with TF-IDF

The SVM with TF-IDF features achieved:

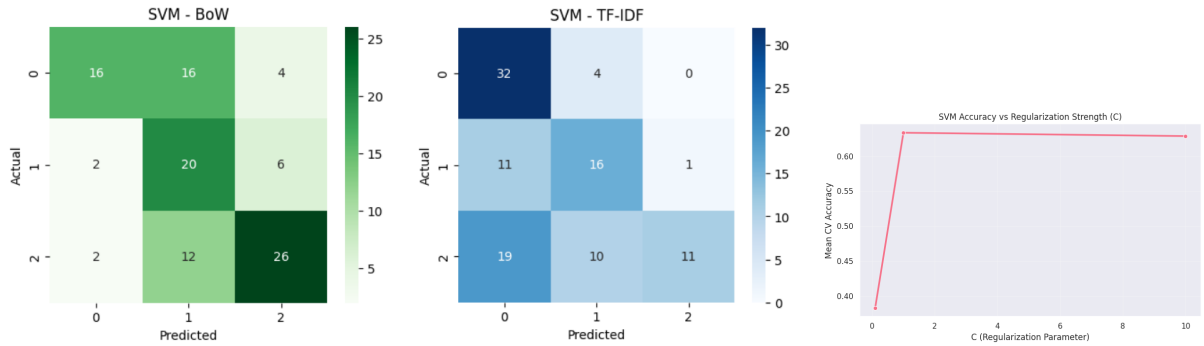
- Accuracy: 0.567
- Class-specific performance:
  - Class 0: 88.9% correct (32/36)
  - Class 1: 57.1% correct (16/28)
  - Class 2: 27.5% correct (11/40)
- Precision: 0.52 to 0.92
- Recall: 0.28 to 0.89
- F1-scores: 0.42 to 0.65

3.5.3 Regularization Effect

Regularization strength (C parameter) significantly impacted performance:

- Low C values (near 0.1) resulted in poor performance (0.38 accuracy)
- Sharp increase between C=0.1 and C=1.0 (up to 0.64 accuracy)
- Performance stabilized between C=1.0 and C=10.0
- Optimal performance at C=1.0 (0.64 accuracy)

SVM Visualizations



3.6 Logistic Regression Results

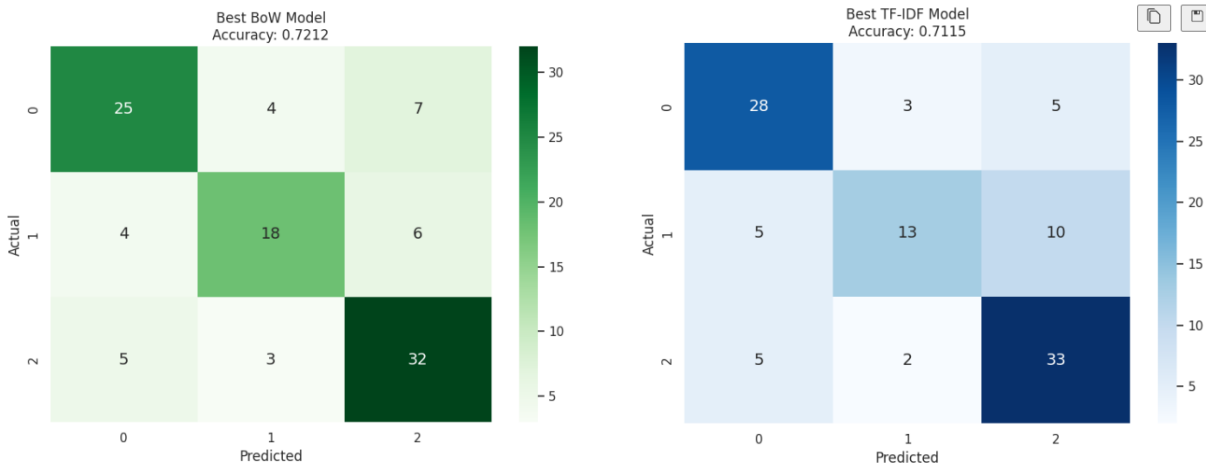
3.6.1 Overall Performance

BoW consistently outperformed TF-IDF with average accuracy of 0.59 versus 0.54.

3.6.2 Class Weighting Effect

Class weighting had minimal impact:

- BoW: 0.60 with balanced weights vs. 0.59 without
- TF-IDF: 0.54 with balanced weights vs. 0.52 without

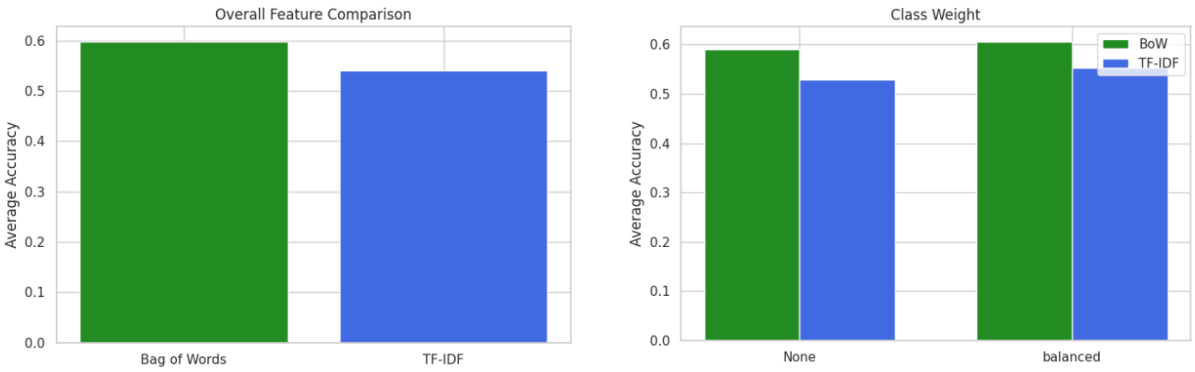


3.6.3 Regularization Effect

Performance improved with increasing C from 10^{-3} to 10^{1}:

- At C=10^{-3}: BoW=0.44, TF-IDF=0.38
- Optimal at C=10^{1}: BoW=0.71, TF-IDF=0.69

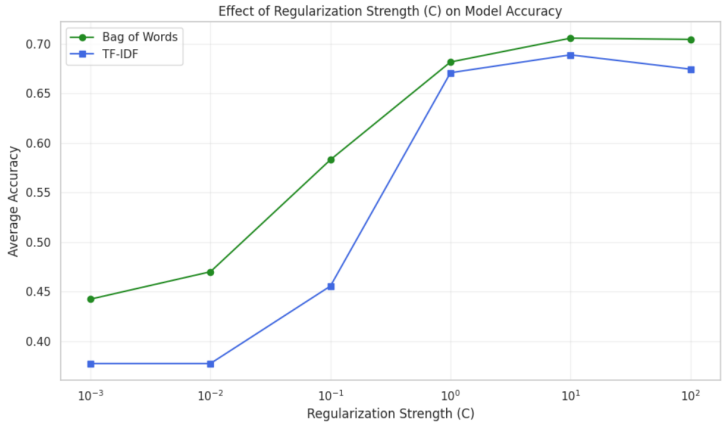
- At C=10^2: BoW plateaued, TF-IDF slightly decreased



3.6.4 Optimal Configuration

Best performance (0.7212 accuracy) achieved with:

- BoW features
- C=10.0
- L1 penalty
- Liblinear solver
- No class weighting



3.7 K-Nearest Neighbors Results

3.7.1 KNN with TF-IDF Features

Euclidean Distance

- Accuracy: 78.99%
- Best k: 30

Manhattan Distance

- Accuracy: 77.89%
- Best k: 28

3.7.2 KNN with BoW Features

Euclidean Distance

- Accuracy: 79.54%
- Best k: 30

Manhattan Distance

- Accuracy: 79.68%
- Best k: 30

subcaption

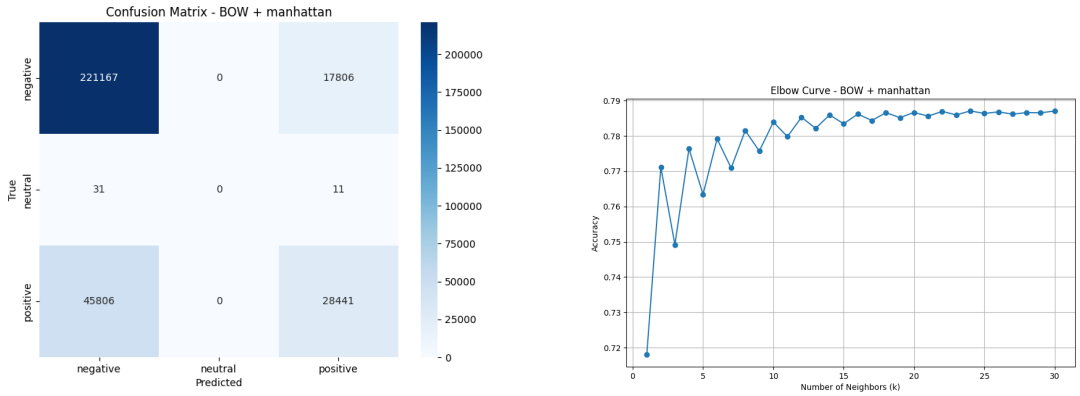


Figure 10: confusion matrix and elbow curve for best combination for knn.

3.7.3 Elbow Curve Analysis

Analysis of elbow curves showed:

- Accuracy stabilized after k=26
- No significant improvement beyond this point

3.7.4 Confusion Matrix Analysis

Key observations:

- Similar accuracy (78-80%) across all configurations
- Zero predictions for the neutral class due to imbalance
- Misclassifications mainly between positive and negative classes

3.8 Artificial Neural Networks Results

3.8.1 Preprocessing and Feature Selection

SVD component selection experiments found:

- Explained variance reached 89% at 3000 components
- Diminishing returns beyond 2000 components
- Optimal performance around 2400 components

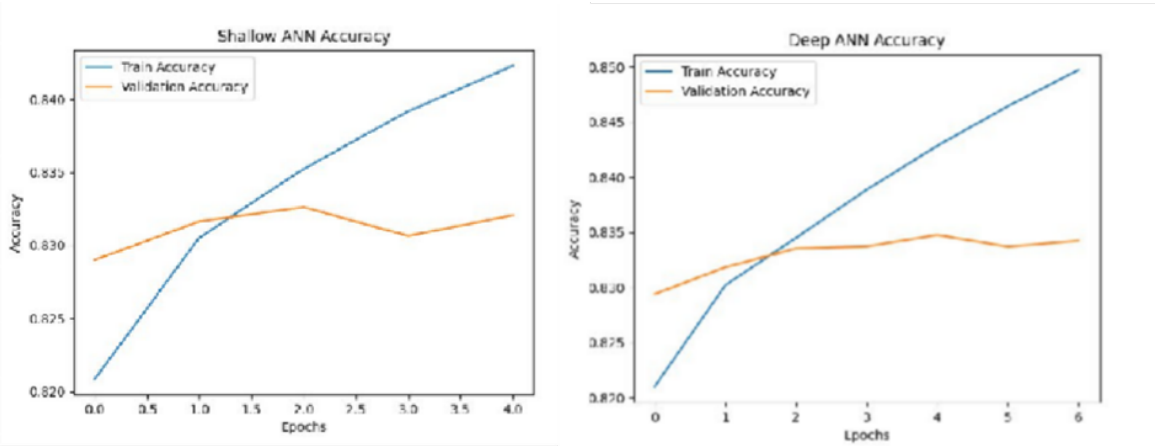


Figure 11: Graph of accuracy v/s epoch for both deep and shallow ANN.

3.8.2 Shallow vs. Deep ANN

Performance comparison:

- Deep ANN: 83.48% accuracy, 82.35% F1-score
- Shallow ANN: 83.26% accuracy, 81.93% F1-score

Class-specific performance (Deep ANN):

- Negative class: F1=0.90
- Positive class: F1=0.59
- Neutral class: F1=0.00

3.9 Unsupervised Clustering Results

3.9.1 Optimal K Selection

The elbow method indicated that k = 2 is optimal for the clustering of K-Means.

Visualizations for Clustering

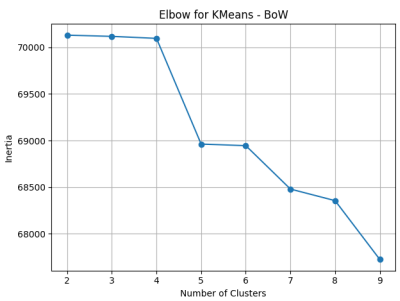


Figure 1: elbow curve for best K.

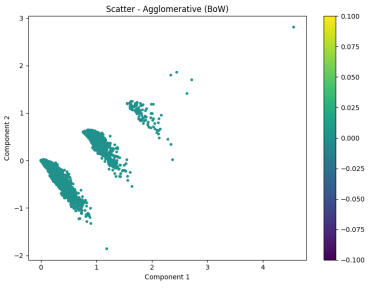


Figure 2: cluster visualization using agglomerative.

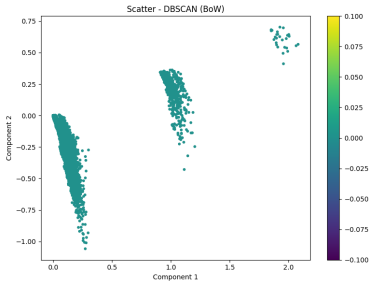


Figure 3: Cluster Visualization using DBSCAN.

3.9.2 Clustering Performance

All clustering models showed:

- Accuracy around 76% across all configurations
- Heavy bias toward the negative class
- Poor identification of neutral and positive classes

Comparison of algorithms:

- K-Means with BoW/TF-IDF: 76.3% accuracy
- Agglomerative with BoW/TF-IDF: 76.4% accuracy
- DBSCAN with BoW: 75.7% accuracy
- DBSCAN with TF-IDF: 76.4% accuracy

4 Comparative Analysis

4.1 Overall Performance Comparison

Table 1: Accuracy Comparison Across Methods

Method	**BoW Accuracy**	**TF-IDF Accuracy**
Naive Bayes	81.46%	81.33%
Decision Tree	57.69%	57.69%
Random Forest	68.27%	57.69%
SVM	59.60%	56.70%
Logistic Regression	72.12%	71.15%
KNN (Euclidean)	79.54%	78.99%
KNN (Manhattan)	79.68%	77.89%
Deep ANN	81.59%	83.48%
Clustering (K-Means)	76.30%	76.30%

4.2 Feature Representation Impact

4.2.1 BoW vs. TF-IDF

- BoW generally outperformed TF-IDF for most classical algorithms
- Exceptions: ANNs performed better with TF-IDF features
- The performance gap was most significant for Random Forest (10.58%)
- For Decision Trees, both methods performed identically

4.2.2 Why BoW Often Outperforms TF-IDF

For sentiment analysis, BoW may be more effective because:

- Common sentiment-indicative words (e.g., "good", "bad", "love", "hate") are crucial
- TF-IDF down-weights these frequent terms, reducing their impact
- Simple frequency counts in BoW preserve the importance of these sentiment markers

4.3 Class Imbalance Impact

All models struggled with the severe class imbalance:

- Strong performance on the majority class (negative)
- Moderate performance on the positive class
- Nearly zero F1-score for the neutral class across all models

4.4 Hyperparameter Sensitivity

- Tree-based methods: Optimal depth around 10-15
- SVM and Logistic Regression: Optimal C around 1.0 to 10.0
- KNN: Optimal k around 28-30
- ANN: Dimensionality reduction to 2400 components optimal

5 Summary

This report has presented a comprehensive analysis of various classification approaches for sentiment analysis. The key findings are:

1. **Best Performing Methods:**
   - Artificial Neural Networks achieved the highest accuracy (83.48%)
   - Naive Bayes methods were close behind (81.46%)
   - KNN with Manhattan distance and BoW features also performed well (79.68%)
2. **Feature Representation:**
   - BoW generally outperformed TF-IDF for traditional algorithms
   - Neural networks performed better with TF-IDF features
   - The choice of feature representation had more impact than clustering algorithm selection
3. **Class Imbalance:**
   - All models struggled with the severe class imbalance
   - The neutral class was particularly problematic, with near-zero F1 scores
   - Overall accuracy was heavily influenced by performance on the majority class
4. **Recommendations:**
   - For balanced performance: KNN with BoW features and Manhattan distance
   - For highest accuracy: Deep ANN with TF-IDF features
   - For computational efficiency: Naive Bayes with token frequency
   - Address class imbalance through resampling or weighted loss functions

This analysis demonstrates that the choice of classification algorithm and feature representation significantly impacts sentiment analysis performance. Future work should focus on addressing the class imbalance problem, particularly for the neutral class, and exploring more sophisticated feature engineering approaches.

A Cloud-Based Deployment and Model Management

In this project, **Google Cloud Run** was utilized to deploy the backend for the sentiment prediction simulation. This deployment allowed us to run a **containerized FastAPI application** that handled inference requests from various machine learning models, such as **Artificial Neural Networks (ANN)**, **Decision Trees**, and **Support Vector Machines (SVM)**.

To efficiently manage large model files and training data, we integrated **Google Cloud Storage (GCS)**. GCS was used to store essential assets like **trained TensorFlow models**, **pickled vectorizers**, and **SVD components**. These files were dynamically loaded into the backend during runtime, ensuring that the container remained lightweight. This design enabled faster and more scalable deployments, ultimately improving the overall performance of the system.

This cloud-based architecture not only enhanced the system's efficiency but also simplified model management, making the entire process more streamlined and scalable.

B Contribution of each member

1. Aaditya Bansal (B23CS1083) : Implemented Naive Bayes Model. Prepared report and video.
2. Anmol Yadav (B23CS1004): Implemented KNN and clustering. Prepared report.
3. Meet Tilala (B23CS1036) : Implemented Logistic Regression and SVM. Prepared video and presentation.
4. Om Sharma (B23CS1048) : Implemented ANN. Prepared Backend.
5. Rudra Gupta (B23CS1098) : Implemented Decision tree and random forest. Prepared frontend.

C References

- **K-Nearest Neighbors (KNN)**: scikit-learn KNN Documentation
- **Clustering Techniques (K-Means, Agglomerative, DBSCAN):**
  scikit-learn Clustering Documentation
  A Clustering-based Approach on Sentiment Analysis
- **Decision Tree Classifier**: scikit-learn Decision Trees
- **Artificial Neural Networks (ANN)**: Keras Sequential Model Guide
- **Naive Bayes Classifier**: scikit-learn Naive Bayes Documentation, Medium, Vectorizer
- **PRML course slides**