REAL-TIME NEWS POPULARITY FORECASTING and FACIAL - RECOGNITION

California State University Long Beach

IS 675: Deep Learning for Business Analytics

Instructor: Mostafa Amini

TEAM - 8

Shiny Porwal - (032231730)

Avinash Mandalapu - (032169330)

KrishnaVyas Desugari – (03220519)

Doranaga Sainadh Vanama – (032183903)

Contents

List of Figures

List of tables

CHAPTER 1 – Structured Data

ABSTRACT

The growing reliance on digital platforms for news consumption has heightened the significance of predicting and enhancing the popularity of online articles, making it a key area for deep learning applications. This project presents a cutting-edge **Intelligent Decision Support System (IDSS)** that employs deep learning techniques to analyze news articles before publication. The system extracts a comprehensive set of features, including textual keywords, multimedia content, and the historical popularity of referenced news, to predict the likelihood of an article achieving high audience engagement. Beyond prediction, the IDSS actively optimizes modifiable features—such as headlines, keywords, and digital content—through advanced feature-tuning methods to increase an article's predicted popularity.

We evaluated the system on a large dataset comprising 39,000 articles from Mashable, leveraging deep learning models in a robust rolling window framework. With a 73% accuracy rate, the top-performing model proved to be successful in forecasting news popularity. Furthermore, feature optimization using stochastic hill climbing techniques resulted in an average improvement of 15 percentage points in the estimated popularity probability of optimized articles.

These results underscore the potential of deep learning-powered systems in transforming online content creation. By providing actionable insights and real-time recommendations, the proposed IDSS empowers content creators to craft articles with greater impact, fostering higher audience engagement and driving digital media success.

BACKGROUND

The rapid proliferation of digital platforms has transformed the way people consume news, making the prediction of online news popularity a critical area of research. The Online News Popularity dataset, containing data from Mashable articles, offers a rich foundation for exploring the factors that drive article engagement. This dataset includes a variety of features such as article keywords, social media shares, traffic sources, and content attributes, which can be leveraged to predict article popularity.

Understanding the factors that contribute to article popularity is essential for news organizations aiming to optimize content creation and distribution. Predictive models, powered by machine learning, can identify patterns and insights to improve audience engagement, ensuring that resources are allocated to articles with the highest potential impact. The goal of this study is to develop a deep learning-based system to predict the popularity of news articles prior to their publication, using the features provided in the dataset.

PROBLEM STATEMENT

In the highly competitive online news industry, predicting which articles will resonate with audiences is a significant challenge. Existing systems often fail to capture the complex interplay of factors such as content structure, social media presence, and timing, which influence article popularity. Furthermore, the lack of actionable insights for content optimization limits the ability of news platforms to maximize engagement.

This project addresses these challenges by building an intelligent prediction and optimization system using the Online News Popularity dataset. The system will leverage deep learning to predict the popularity of articles based on their inherent features and provide recommendations for feature adjustments to enhance their engagement potential. The objective is to help news platforms make informed decisions about content creation, ensuring a higher likelihood of success in an increasingly crowded digital space.

IMPACT ON BUSINESS

The ability to predict and enhance online news popularity has significant implications for digital media organizations. Popular articles drive increased traffic, generate higher ad revenue, and strengthen audience loyalty. However, low-performing articles represent not only a loss of potential revenue but also inefficiencies in resource allocation.

Using insights derived from the **Online News Popularity** dataset, the proposed system will:

- Empower content creators with predictive analytics to gauge an article's potential success.
- Provide optimization strategies to refine article attributes for improved engagement.
- Enable organizations to focus their resources on high-potential content, thereby increasing operational efficiency and profitability.

By aligning content creation with audience preferences, this system will contribute to a better understanding of factors influencing article virality. It will also enable news platforms to adapt dynamically to evolving trends, ensuring their relevance and competitiveness in the digital media landscape.

DATA SET AND DOMAIN

Dataset Overview

The analysis for this project utilizes the **Online News Popularity** dataset from the UCI Machine Learning Repository, which includes a comprehensive collection of data on articles published by Mashable. This dataset spans a period, capturing diverse attributes associated with 39,644 articles. It provides detailed information on various features, enabling an in-depth exploration of the factors influencing news article popularity.

Data Selection Criteria

The dataset was filtered and analyzed based on the following criteria to ensure relevance and robustness of predictions:

- **Articles Published on Mashable:** All data points in the dataset correspond to articles published exclusively on Mashable, ensuring consistency in content type and publication platform.

- **Shares as the Target Variable:** The primary metric for measuring popularity is the number of social media shares an article receives, representing audience engagement.
- **Feature Inclusion:** The dataset includes 60 features grouped into six categories:
    - **Content Features:** Keywords, content type, and length of the article.
    - **Social Media Features:** The number of shares on various social platforms.
    - **Traffic Sources:** Referrers and search engine impact.
    - **Timing Features:** Day of the week and publication time.
    - **Digital Media Content:** Images, videos, and media presence.
    - **User Interaction Features:** Comments and click-through data (if available).
- **Pre-publication Analysis:** All features are extracted before publication, enabling predictions to be made proactively.

Key Assumptions in Data Analysis

The analysis is built on the following assumptions to refine the scope of the project:

1. **Relevant to the Current Media Landscape:** While the dataset originates from Mashable, its features and attributes are considered representative of broader online news trends.
2. **Consistency in Publication Practices:** Articles adhere to a standard structure, making it feasible to generalize patterns across entries.
3. **Defining Popularity Threshold:** Articles with a higher number of shares (based on a defined threshold in exploratory analysis) are considered popular, enabling clear categorization for classification tasks.
4. **No Post-publication Adjustments:** Features used for prediction are assumed to remain unchanged after publication, ensuring that predictions rely solely on pre-publication characteristics.

Data Source: [UCI Dataset Link]

DATA SET

The **Online News Popularity** dataset provides a detailed breakdown of features associated with 39,644 news articles published by Mashable. Below is a data dictionary outlining the features, their data types, descriptions, and other relevant details:

Table 1.1 Data Set

Column Name	Datatype	Description	Information	Null Values
URL	object	URL of the article.	Unique for each article.	0%
timedelta	int64	Days since the article was published.	Range: 1–731.	0%
shares	int64	Number of shares an article received on social media (target variable).	Range: 1–843,300.	0%
n_tokens_title	int64	Number of words in the title.	Range: 1–23.	0%
n_tokens_content	int64	Number of words in the content.	Range: 0–105,470.	0%
num_hrefs	int64	Number of hyperlinks in the article.	Range: 0–304.	0%
num_self_hrefs	int64	Number of internal Mashable links.	Range: 0–125.	0%
num_imgs	int64	Number of images in the article.	Range: 0–150.	0%
num_videos	int64	Number of videos in the article.	Range: 0–12.	0%
average_token_length	float64	Average length of words in the article.	Range: 2.0–8.0.	0%
weekday_is_monday	int64	Indicates if the article was published on Monday (binary: 1 or 0).	2 categories (Yes/No).	0%
weekday_is_tuesday	int64	Indicates if the article was published on Tuesday (binary: 1 or 0).	2 categories (Yes/No).	0%
weekday_is_wednesday	int64	Indicates if the article was published on Wednesday (binary: 1 or 0).	2 categories (Yes/No).	0%
weekday_is_thursday	int64	Indicates if the article was published on Thursday (binary: 1 or 0).	2 categories (Yes/No).	0%
weekday_is_friday	int64	Indicates if the article was published on Friday (binary: 1 or 0).	2 categories (Yes/No).	0%
weekday_is_saturday	int64	Indicates if the article was published on Saturday (binary: 1 or 0).	2 categories (Yes/No).	0%
weekday_is_sunday	int64	Indicates if the article was published on Sunday (binary: 1 or 0).	2 categories (Yes/No).	0%

is_weekend	int64	Indicates if the article was published on a weekend (binary: 1 or 0).	2 categories (Yes/No).	0%
global_subjectivity	float64	Average subjectivity of the content.	Range: 0.0–1.0.	0%
global_sentiment_polarity	float64	Overall polarity of the content.	Range: -1.0 to 1.0.	0%
title_subjectivity	float64	Subjectivity score of the title.	Range: 0.0–1.0.	0%
title_sentiment_polarity	float64	Sentiment polarity of the title.	Range: -1.0 to 1.0.	0%
abs_title_subjectivity	float64	Absolute value of title subjectivity.	Range: 0.0–1.0.	0%
abs_title_sentiment_polarity	float64	Absolute value of title sentiment polarity.	Range: 0.0–1.0.	0%

Preprocessing Notes

- **Removed Features:** Features such as url and timedelta were excluded during analysis as they are unique identifiers or irrelevant for prediction tasks.
- **Categorical Encoding:** Binary and categorical variables (e.g., weekday_is_*, is_weekend) were encoded to align with the machine learning models.
- **Target Variable:** The number of shares (shares) was transformed to a binary classification (popular/not popular) based on a threshold derived from exploratory analysis.

EXPLORATION DATA ANALYSIS

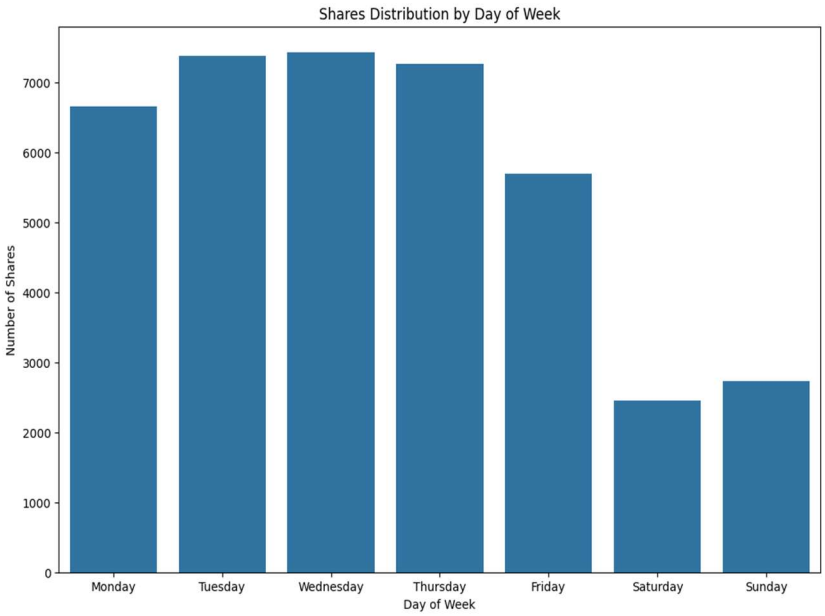

Figure 1 Shares Distribution by Day of Week

This bar chart illustrates the distribution of article shares across the days of the week, highlighting audience engagement trends. The analysis reveals that Tuesday and Wednesday garner the highest number of shares, followed closely by Monday and Thursday, indicating that mid-week publications tend to perform best in terms of audience reach and engagement. In contrast, Friday experiences a moderate decline in shares, while Saturday and Sunday register the lowest levels of engagement, suggesting that weekend articles are less likely to go viral or attract significant audience attention.

These insights imply that content publication schedules should prioritize mid-week days, particularly Tuesday through Thursday, to maximize social media shares. For weekends, alternative strategies such as enhancing multimedia content or targeting niche audiences may help improve engagement. Understanding these temporal patterns enables content creators to align publication schedules with peak engagement periods, thereby optimizing article performance and audience reach.

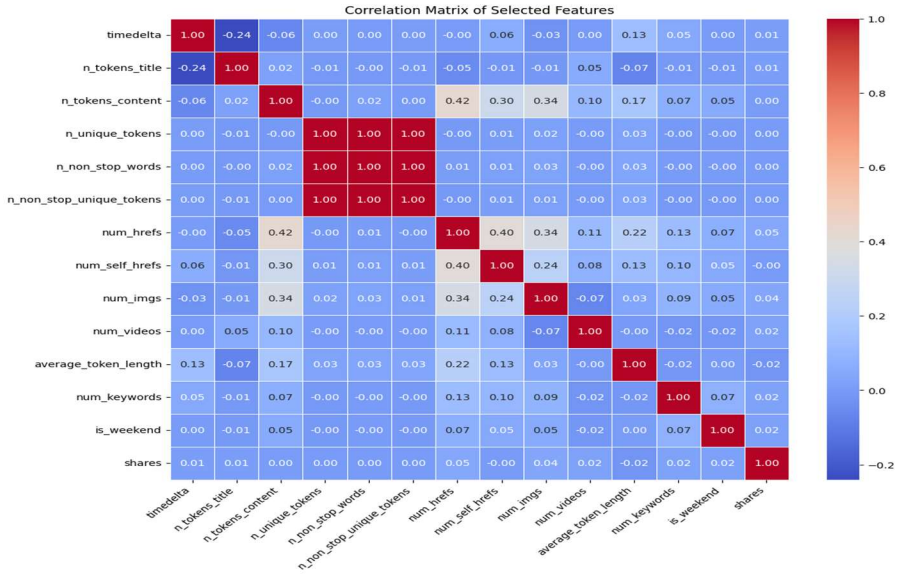

Figure 1.2 Correlation Matrix of Selected Features

This correlation matrix visualizes the relationships between various features in the dataset, with correlation values ranging from -1 to 1. A value close to 1 indicates a strong positive correlation, while a value close to -1 indicates a strong negative correlation. In this matrix, features like num_hrefs (number of hyperlinks) and n_tokens_content (number of words in content) show a moderately positive correlation, suggesting that longer articles with more hyperlinks may increase user engagement. However, shares (target variable) exhibits weak correlations with most features, indicating that no single feature strongly determines article popularity on its own.

Interestingly, features such as num_imgs (number of images) and num_videos (number of videos) have minimal correlation with shares, implying that multimedia content alone does not guarantee higher engagement. Additionally, the feature is_weekend shows an insignificant correlation with shares, reaffirming that the day of the week might not significantly impact popularity. This matrix highlights the complexity of predicting article popularity, emphasizing the need for advanced models that can capture the nuanced interplay of multiple features rather than relying on linear relationships.

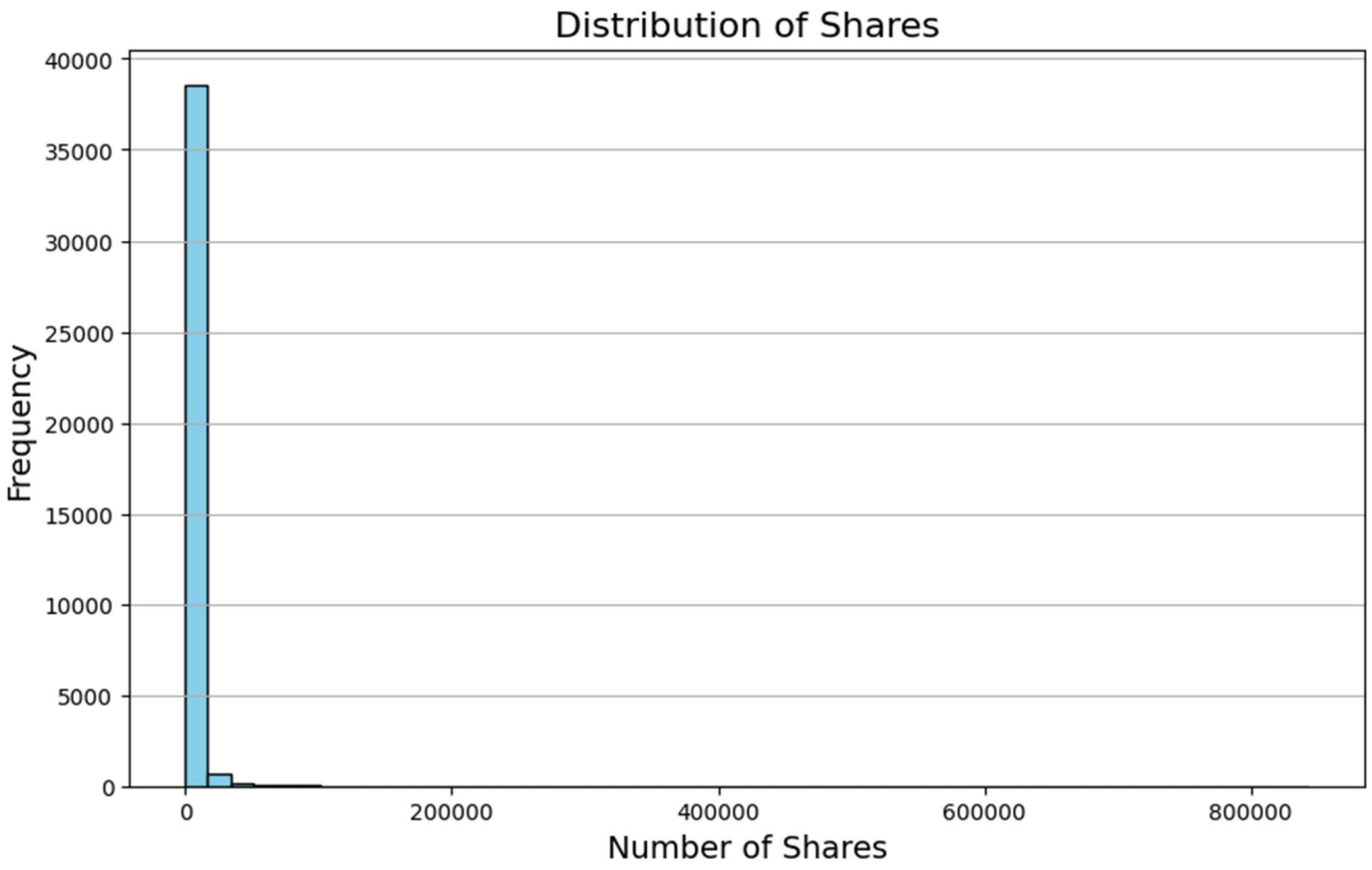

Figure 1.3 Distribution of Shares

Explanation of the Distribution of Shares Chart

This histogram represents the distribution of article shares after filtering. Most articles have fewer than **10,000 shares**, with the frequency peaking at very low values. A sharp decline is observed as the number of shares increases, indicating that only a small fraction of articles achieve high popularity. The presence of a long tail signifies that some articles are highly shared, reaching up to **140,000 shares**.

The heavily skewed distribution suggests that the dataset contains a significant number of outliers with unusually high shares. This chart highlights the importance of handling data imbalance to improve prediction accuracy and model performance.

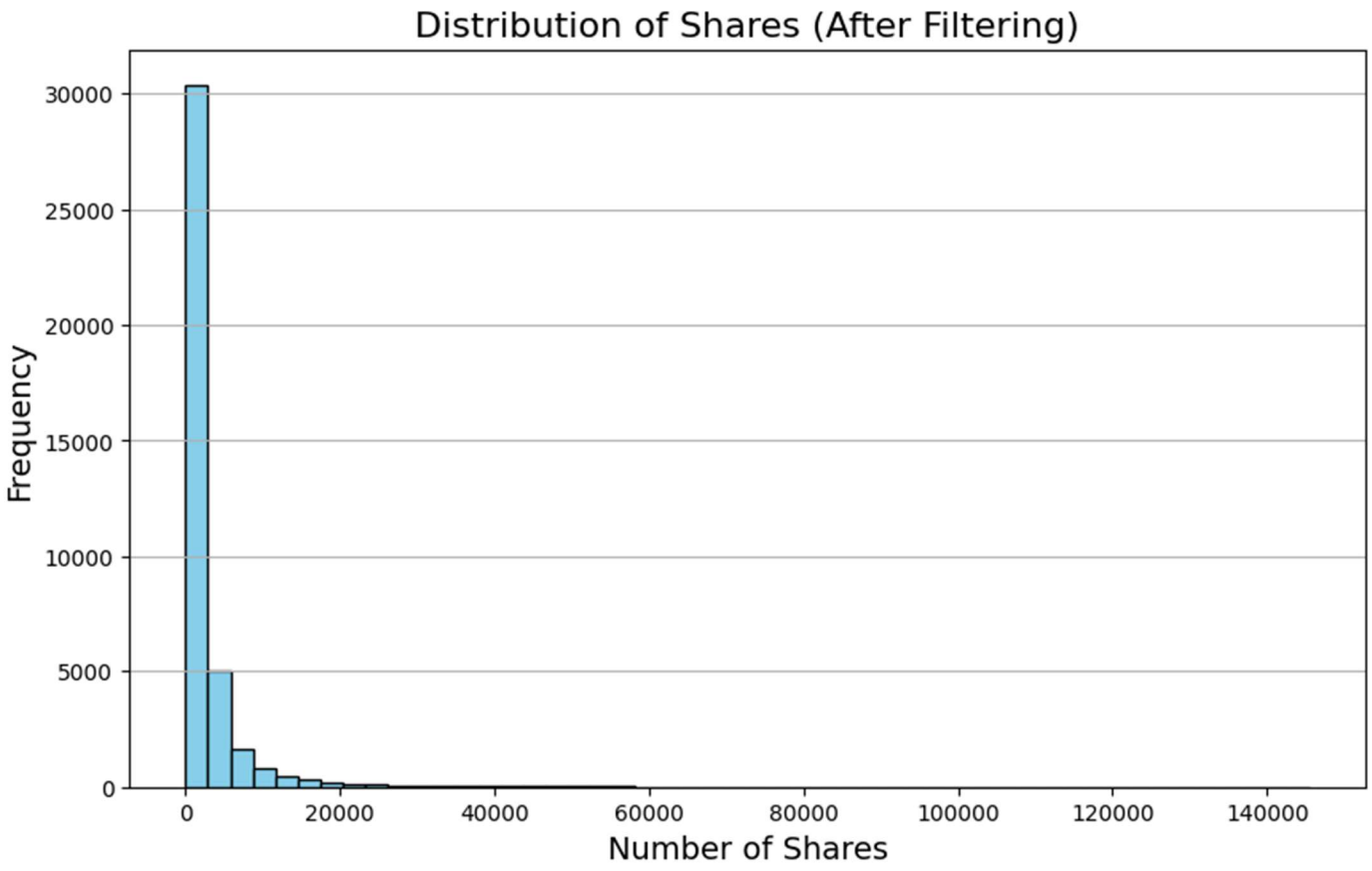

Figure1. 4 Distribution of Shares (After Filtering)

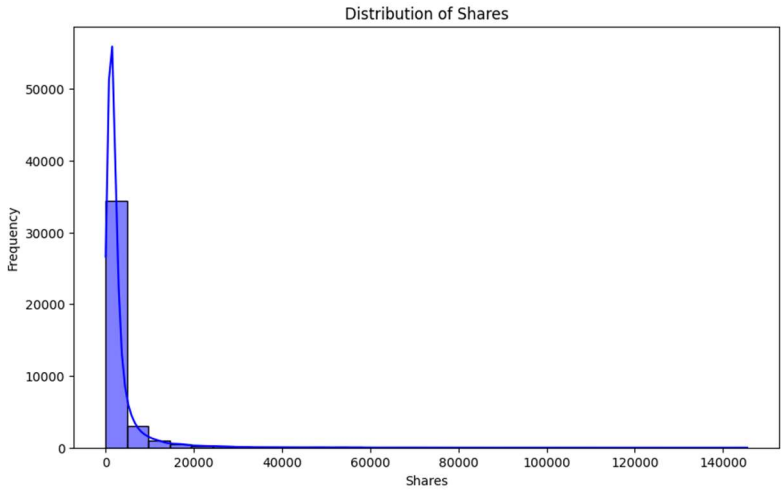

Figure 1. 5 2nd Distribution of Shares

Explanation of the Distribution of Shares (With Density) Chart

This chart displays the frequency of article shares, overlaid with a density curve to provide a smoother representation of the distribution. The histogram highlights that most articles receive fewer than **10,000 shares**, while the density curve shows a sharp peak near the lower end of the spectrum. The long tail in both the histogram and the density curve indicates the presence of highly viral articles with up to **140,000 shares**.

The skewed distribution emphasizes the need for normalization techniques, such as applying a logarithmic transformation, to improve the modeling process. The density curve provides additional clarity on the overall data trend, showing that the majority of the dataset lies in the low-share range, making it important to account for this imbalance during analysis.

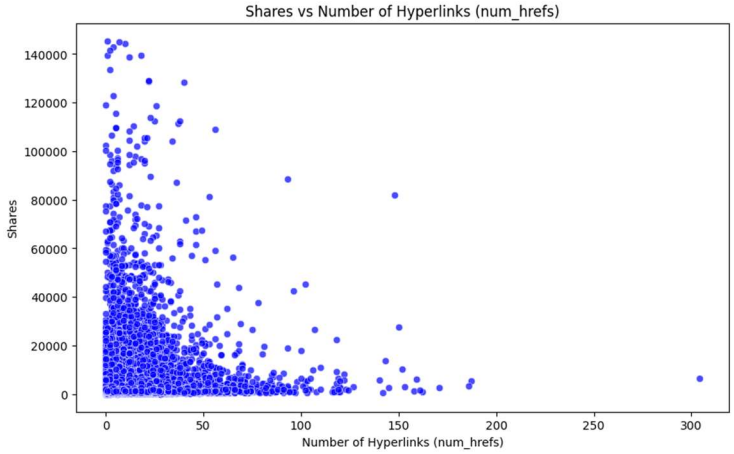

Figure 1.6 Shares vs. Number of Hyperlinks Chart (num_herfs)

Explanation of the Shares vs. Number of Hyperlinks Chart

This scatter plot shows the relationship between the number of hyperlinks (num_hrefs) in an article and the number of shares it receives. The majority of data points are clustered at lower values of hyperlinks and shares, indicating that articles with fewer than **50 hyperlinks** dominate the dataset. While some articles with a higher number of hyperlinks achieve substantial shares, the trend does not show a strong correlation.

The lack of a clear upward trend suggests that adding more hyperlinks does not necessarily guarantee higher shares. However, outliers indicate that some articles with an unusually high number of hyperlinks (up to **300**) may still perform well, suggesting a possible interaction effect with other features, such as content relevance or quality. This insight is critical for optimizing hyperlink usage without overloading articles.

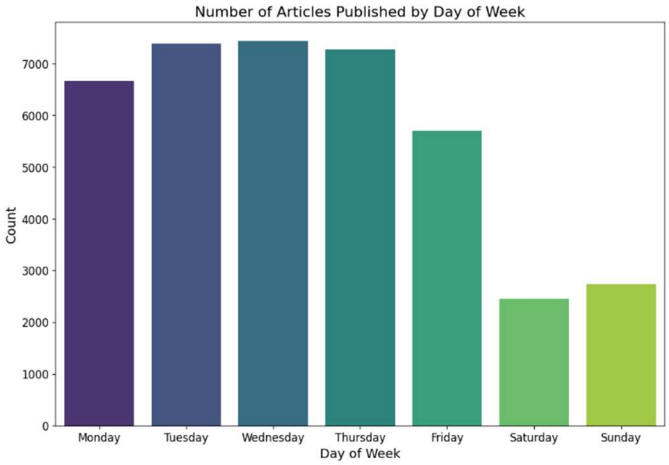

Figure 1.7 Number of Articles Published by Day of Week Chart

Explanation of the Number of Articles Published by Day of Week Chart

This bar chart shows the distribution of article publications across different days of the week. The highest number of articles is published on **Tuesday**, **Wednesday**, and **Thursday**, indicating these are peak publication days. In contrast, the number of articles published declines towards the weekend, with **Saturday** and **Sunday** having the lowest counts.

The trend suggests that news platforms prioritize mid-week for content releases, potentially to maximize engagement during weekdays. This insight is valuable for scheduling content publication strategies and understanding audience consumption patterns throughout the week.

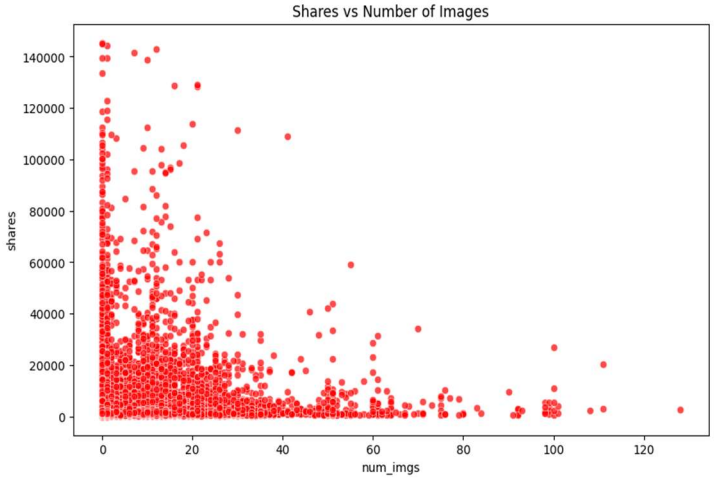

Figure 1. 8 Shares vs. Number of Images

Explanation of the Shares vs. Number of Images Chart

This scatter plot illustrates the relationship between the number of images (num_imgs) in an article and the number of shares it receives. Most articles cluster around **0 to 20 images**, with a concentration of shares below **20,000**. While some articles with a higher number of images achieve significant shares, no clear upward trend is observed, indicating that the number of images alone does not strongly determine popularity.

The lack of a strong correlation suggests diminishing returns for adding excessive images to articles. However, some outliers indicate that articles with a high number of images (e.g., 60–120) can still perform exceptionally well, possibly due to high-quality or engaging content. This insight highlights the need to balance multimedia use for optimizing engagement.

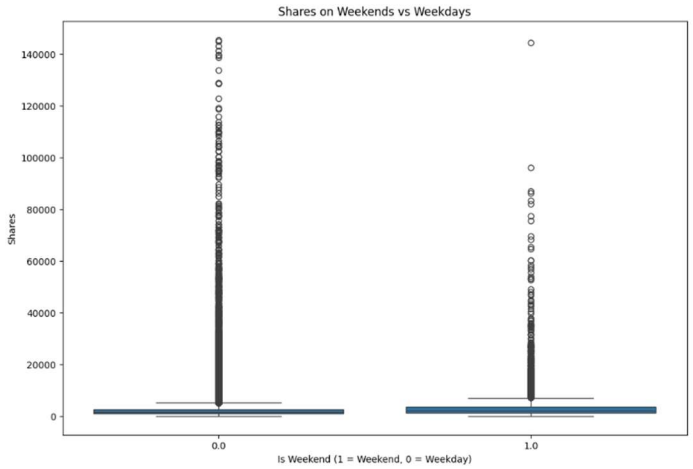

Figure 1.9 Shares on Weekends vs. Weekdays

Explanation of Shares on Weekends vs. Weekdays Chart

This box plot compares the distribution of article shares between weekdays (0) and weekends (1). The median number of shares is slightly higher for weekdays, with a greater spread of shares on both days. However, the plot reveals a higher density of extreme outliers (articles with exceptionally high shares) during weekdays compared to weekends, suggesting that weekdays are more favorable for viral content.

The results indicate that while content on weekends can still perform well, weekdays generally see higher engagement. This insight suggests that news platforms should prioritize publishing key articles during weekdays to capitalize on higher audience activity and sharing potential.

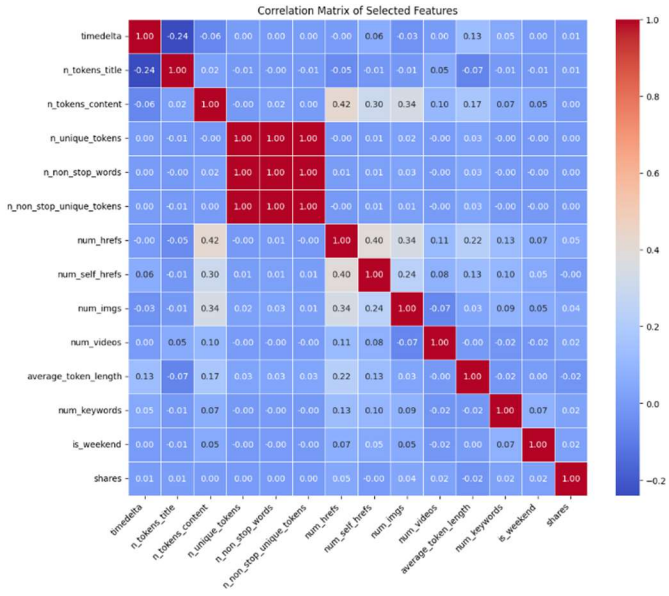

Figure 1.10 Correlation Matrix for All Numberical Values

Explanation of the Correlation Matrix Chart

This correlation matrix visualizes the relationships between numerical features in the dataset, with values ranging from -1 to 1. Strong positive correlations are highlighted in red, such as **num_hrefs and n_tokens_content (~0.42)**, indicating that longer articles often include more hyperlinks. Similarly, **num_hrefs and num_self_hrefs (~0.40)** show a strong relationship, suggesting that articles with more external links also include more internal links.

The target variable, shares, exhibits weak correlations with other features, indicating that no single factor predominantly determines article popularity. This emphasizes the need to explore feature combinations during modeling. Features like **num_imgs and num_hrefs**, while weakly correlated with shares, could still play indirect roles in driving engagement when combined with other variables.

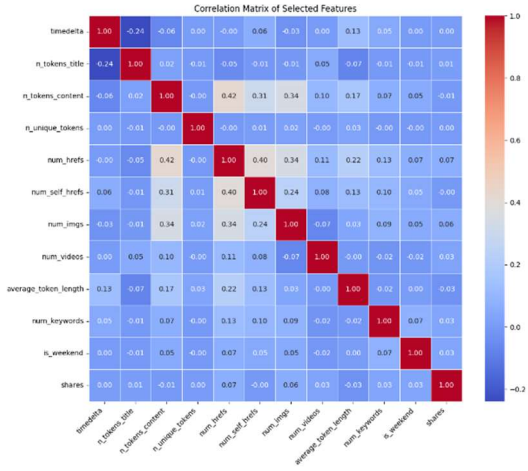

Figure 1.11 Correlation Matrix for Selected Features

Explanation of the Updated Correlation Matrix Chart

This updated correlation matrix highlights the relationships between selected features, with coefficients ranging from -1 to 1. Positive correlations are shown in red, such as **num_hrefs and n_tokens_content (~0.42)** and **num_self_hrefs and num_hrefs (~0.40)**, indicating that longer articles tend to include more hyperlinks, both external and internal. Features like **num_imgs and num_hrefs (~0.34)** also exhibit moderate positive relationships.

The target variable, shares, shows weak correlations with most features, with **num_imgs (~0.06)** and **num_hrefs (~0.07)** being the strongest among them. This indicates that no single feature heavily influences shares, emphasizing the need for combining multiple features in modeling to capture nuanced relationships. This matrix is essential for identifying potential predictors for popularity.

Target Variable: shares (binarized based on a threshold of 1400 shares)

- **Popular**: shares > 1400 (Class 1)
- **Unpopular**: shares ≤ 1400 (Class 0)

Features Used:

- Temporal, content-based, and social features such as timedelta, n_tokens_title, num_hrefs, kw_avg_avg, is_weekend, and various sentiment and keyword metrics.

Preprocessing:

- Removed extreme outliers where shares > 150,000.
- Standardized all feature values using **StandardScaler** to ensure effective optimization during training.

Feature Engineering

1. **Data Transformation:** To achieve normality and stabilize variances in our features, we applied transformations such as logarithmic or square root. These transformations proved particularly effective for skewed distributions or features with varying scales. For instance, in our dataset, size data benefited from log transformations, while count data utilized square-root transformations.
2. **Scaling:** Feature scaling ensures that all numerical variables are on a comparable scale. As a group, we applied techniques such as MinMaxScaler to normalize features into a range between 0 and 1. This approach aids algorithms that are sensitive to feature magnitudes, such as gradient-based methods.
3. **Feature Selection:** Reducing irrelevant features is key to improving model interpretability and performance. As a group, we employed techniques such as forward selection, backward elimination, and feature importance scores (from models like Random Forest or Decision Tree) to identify critical predictors. For our dataset, important features identified in the feature importance plots include kw_avg_avg, self_reference_avg_shares, and is_weekend.

Model Building

1. **Model Selection Process:** The "No Free Lunch" theorem suggests experimenting with various algorithms since their suitability depends on the dataset's structure. Decision Trees, Random Forests, SVMs, and Neural Networks were explored in your case to identify the optimal classifier based on their comparative performance.
2. **Data Splitting:** The dataset was split into training (70%) and testing (30%) sets, maintaining the class distribution for reproducibility. The train_test_split function with a defined random_state ensures consistent splits across iterations.
3. **Modeling Techniques:** Each algorithm has strengths and weaknesses. For instance:
    - Decision Trees provide interpretability but may overfit.
    - Random Forests excel at generalization by combining multiple trees.
    - Neural Networks leverage complex feature relationships but require significant computation and tuning.

Model Evaluation

Confusion Matrix: Provides a breakdown of predictions into True Positives, True Negatives, False Positives, and False Negatives. This aids in understanding where the model performs well and where it falters.

Evaluation Metrics:

Accuracy: Measures overall prediction correctness but may be misleading for imbalanced datasets.

Precision and Recall: Address specific class predictions. Precision answers "how many selected items are relevant?" Recall answers "how many relevant items are selected?"

F1 Score: Balances precision and recall, useful in uneven class distributions.

ROC-AUC Curve: Highlights the trade-off between True Positive Rate and False Positive Rate across

thresholds. Higher AUC reflects better model performance.

Insights: Use these metrics to compare models (e.g., Random Forest outperformed Decision Trees and Neural Networks in accuracy and ROC-AUC). Highlight specific improvements achieved through feature engineering and hyperparameter tuning.

Machine Learning Models – Structured Data

Evaluation of different models:

Decision Tree:

Initial Model

- A Decision Tree Classifier was trained with a maximum depth of 5 for initial testing and analysis.
- **Accuracy**: The initial accuracy of the model was calculated to evaluate its baseline performance.

Decision Tree Visualization

- A visualization of the Decision Tree was created to understand its structure, including splits, leaf nodes, and the features used at each decision point.
- The visualization highlighted the simplicity and interpretability of the model.

Hyperparameter Tuning

- To optimize the Decision Tree model, hyperparameter tuning was performed using **Grid Search with Cross-Validation**.

Parameters Tuned:

- Maximum Depth (max_depth): Limited the depth of the tree to prevent overfitting.
- Minimum Samples per Split (min_samples_split): Controlled the minimum number of samples required to split an internal node.
- Minimum Samples per Leaf (min_samples_leaf): Ensured a minimum number of samples at a leaf node.
- Splitting Criterion (criterion): Compared the gini impurity and entropy metrics for selecting the best split.

Best Parameters:

- criterion: Gini

- max_depth: 10
- min_samples_split: 10
- min_samples_leaf: 1

- The best cross-validated accuracy obtained during tuning was also reported.

Feature Importance

- The importance of each feature in predicting the target variable was evaluated.
- Features were ranked in descending order of their contribution to the model's decisions, providing insights into the most influential factors.

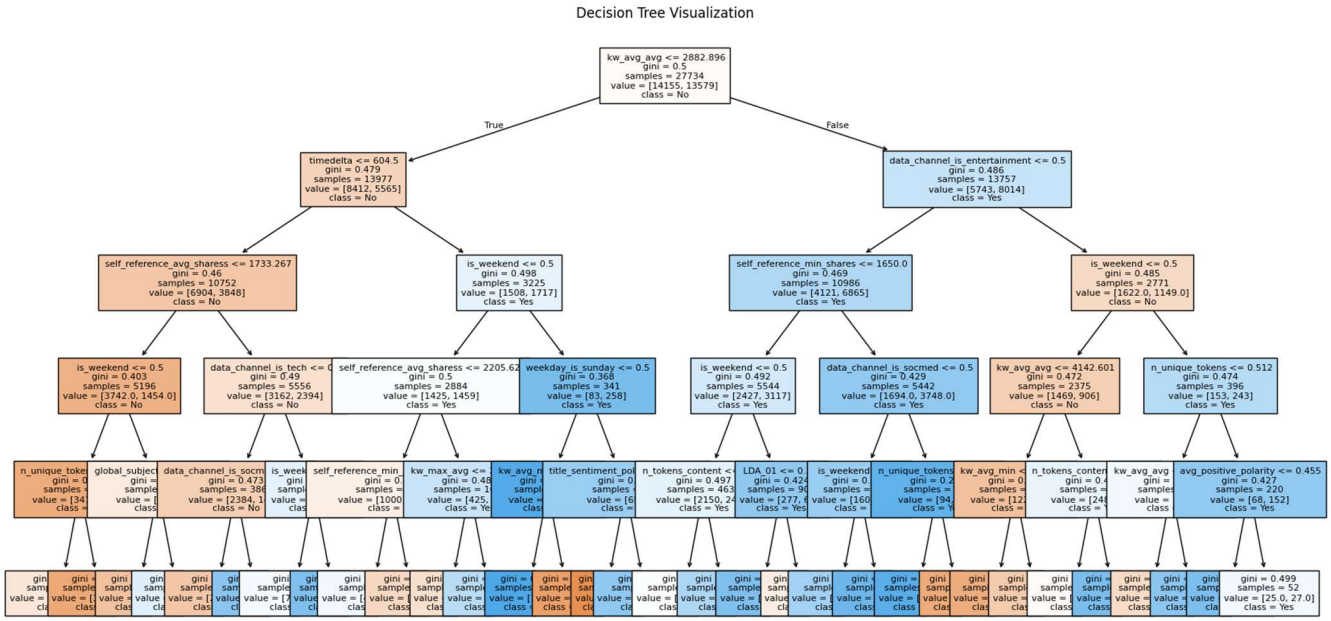

Figure 1. 12 Decision Tree For Structured Data

Explanation of the Decision Tree

This decision tree illustrates the classification process used to predict whether an article will become popular based on key features. Each node represents a splitting criterion based on a feature, with

the decision guided by minimizing the Gini impurity. The tree splits the dataset into branches, progressively narrowing down the samples into classes (Yes for popular and No for not popular).

Key Observations:

Root Node:

- The root node splits on kw_avg_avg (average keyword frequency) with a threshold of **2882.896**, indicating its importance in determining article popularity.
- Samples below the threshold are classified as No, while those above progress to further splits.

Feature Importance:

- Features-like kw_avg_avg,timedelta(time since publication), self_reference_avg_shares, and is_weekend are among the primary splitting criteria, indicating their significant influence on the classification.
- Temporal and self-referencing metrics play crucial roles in predicting article popularity.

Leaf Nodes:

- The tree terminates at leaf nodes, where the samples are classified as Yes or No. Each leaf contains the number of samples, their distribution, and the predicted class.
- For example, a leaf node with low kw_avg_avg and is_weekend = 0.5 is classified as No due to a higher concentration of non-popular articles.

Complexity:

- The depth of the tree shows that multiple features interact to determine article popularity, emphasizing the need for pruning or limiting depth to prevent overfitting.

Tunning

This section presents the results from tuning the Decision Tree model using grid search with cross-validation. The tuning process evaluated various combinations of hyperparameters to maximize model performance, as outlined below.

Best Parameters

- **Criterion**: gini (used to measure the quality of splits).
- **Max Depth**: 10 (limits the depth of the tree to prevent overfitting).

- **Min Samples Split**: 10 (minimum number of samples required to split a node).
- **Min Samples Leaf**: 1 (minimum number of samples required at a leaf node).

These parameters provided the best cross-validated accuracy, **0.6287**, indicating the model performs moderately well in classifying the data.

Top 10 Results:

The table shows the top 10 parameter configurations based on the cross-validated mean test scores:

- The highest accuracy is **0.6287**, achieved with the parameters mentioned above.
- The standard deviation for the top scores is relatively low (~0.005–0.007), suggesting that the model's performance is consistent across folds.
- Parameters using both gini and entropy as criteria were tested, with gini yielding slightly better results in the top configurations.

Analysis

Performance: While the best accuracy is ~62.87%, the model leaves room for improvement. Further techniques, such as feature engineering, balancing the dataset, or using ensemble methods like Random Forest or Gradient Boosting, may enhance performance.

Interpretability**:** The decision tree's parameters indicate that a relatively shallow tree (max depth of 10) with splits based on at least 10 samples is optimal, reducing the risk of overfitting.

Trade-Off**:** The tuning results show a trade-off between model complexity (e.g., deeper trees, fewer minimum samples) and accuracy. The chosen parameters balance this trade-off effectively.

Decision tree Confusion Matrix:

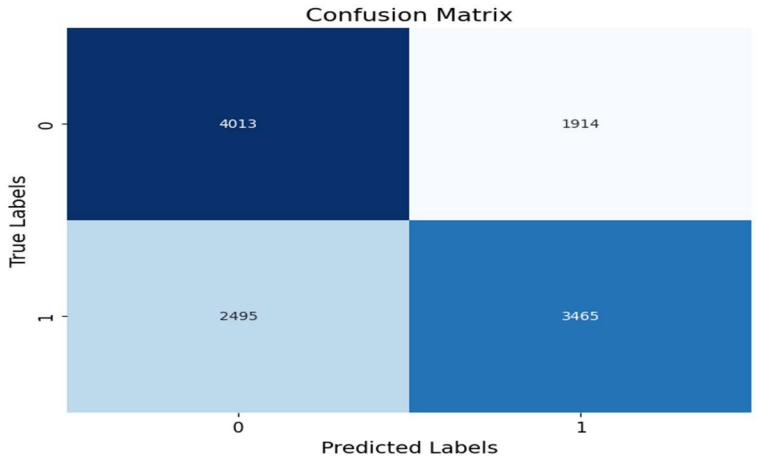

Figure 1.13 Decision Tree Confusion Matrix

Explanation

The confusion matrix visualizes the performance of the Decision Tree classifier on the test dataset. It provides insights into how well the model distinguishes between the two classes (e.g., 0 for unpopular articles and 1 for popular articles).

Key Observations:

True Positives (TP):

- **3465** instances were correctly classified as popular (1).
- These represent articles that were correctly predicted to achieve popularity.

True Negatives (TN):

- **4013** instances were correctly classified as unpopular (0).
- These are articles that were accurately predicted to lack popularity.

False Positives (FP):

- **1914** instances were incorrectly classified as popular when they were actually unpopular.
- This indicates the model overpredicts popularity in some cases.

False Negatives (FN):

- **2495** instances were incorrectly classified as unpopular when they were popular.
- These missed opportunities where the model failed to identify truly popular articles.

Performance Metrics:
From the confusion matrix, key performance metrics can be calculated:

1. **Accuracy**:

   Accuracy = (TP+TN) / (TP+TN+FP+FN) = (3465+4013) / (3465+4013+1914+2495) ≈0.628

   The accuracy aligns with the cross-validated accuracy from hyperparameter tuning.

2. **Precision for Class 1 (Popular Articles)**:

   Precision = (TP) / (TP+FP) = (3465) / (3465+1914) ≈ 0.644

Recall for Class 1 (Popular Articles):

Recall = (TP) / (TP+FN) = (3465) / (3465+2495) ≈ 0.581

3. **F1-Score for Class 1**:

   F1 = (2·Precision·Recall)/(Precision + Recall) ≈ 0.611

Insights:

- The model achieves a moderate balance between precision and recall for identifying popular articles.
- However, the relatively high number of **False Negatives** suggests the model struggles with accurately identifying all popular articles.
- To improve performance:
  - Explore ensemble methods like Random Forest or Gradient Boosting to reduce errors.
  - Fine-tune hyperparameters further or experiment with additional features to capture complex relationships.

Decision Feature Importance Visualization

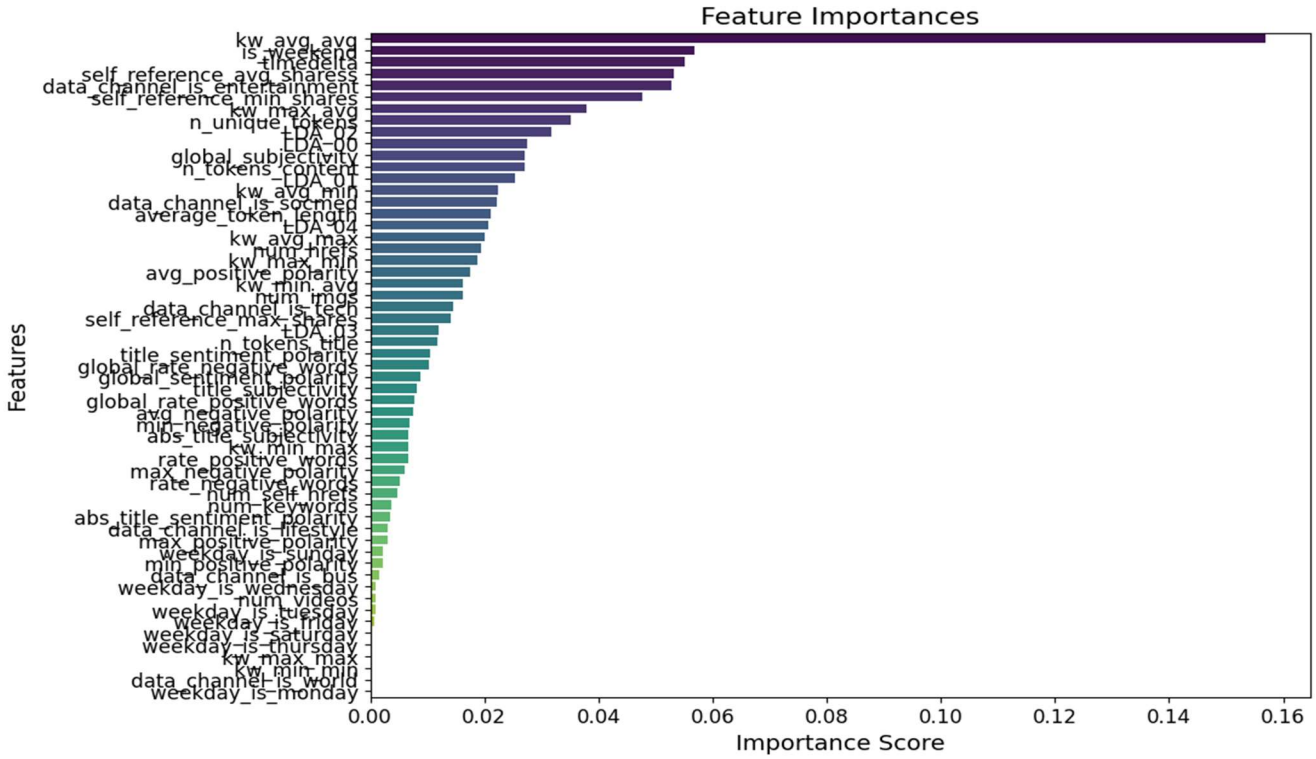

Figure 1.14 Decision Feature Importance Visualization

The bar chart displays the relative importance of features in the Decision Tree model for predicting article popularity.

Key Observations:

1. **Top Features**:
    - kw_avg_avg (average keyword frequency) is the most critical feature, significantly influencing predictions.
    - Other key features include is_weekend, timedelta, self_reference_avg_shares, and data_channel_is_entertainment.
2. **Insights**:
    - Temporal features (timedelta, is_weekend) and content-specific metrics (e.g., self_reference_min_shares) play a vital role.

- Features with low importance can be considered for removal to simplify the model without affecting performance significantly.

Decision tree ROC Curve:

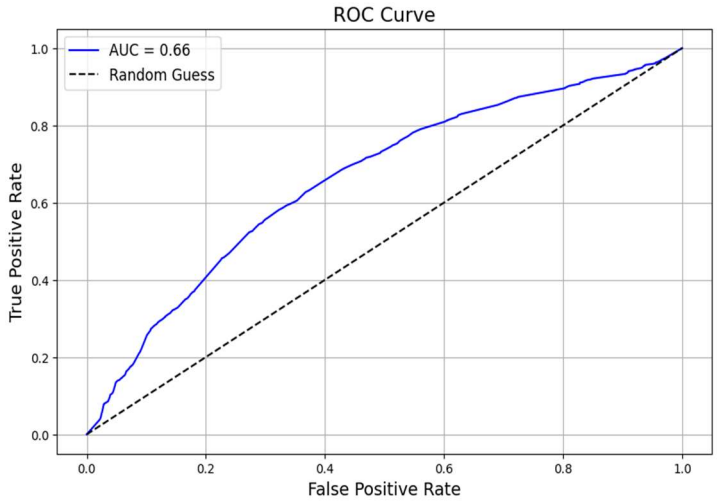

Figure 1.15 Decision tree ROC Curve

The Receiver Operating Characteristic (ROC) curve evaluates the trade-off between the True Positive Rate (TPR) and False Positive Rate (FPR) for various classification thresholds.

Key Observations:

1. **AUC Score**:
    - The Area Under the Curve (AUC) is **0.66**, indicating moderate model performance, slightly better than random guessing (AUC = 0.5).
2. **Insights**:
    - The curve suggests the model can moderately distinguish between the two classes.
    - Further improvements can be achieved by optimizing the model or incorporating additional features.

Decision tree Precision Recall Curve:

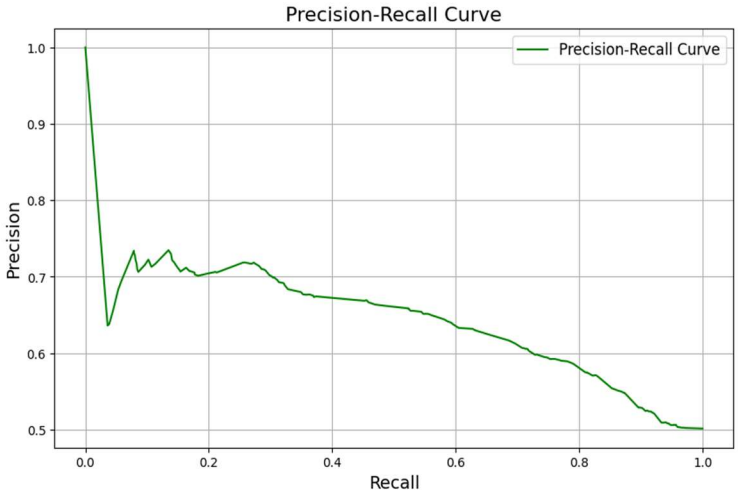

Figure 1.16 Precision Recall Curve

The Precision-Recall curve evaluates the trade-off between precision (positive predictive value) and recall (true positive rate) for different thresholds.

Key Observations:

1. **Trend**:
    - Precision is high at low recall values but decreases as recall increases, indicating the model performs better for a smaller number of positive predictions.
2. **Insights**:
    - The curve suggests the model maintains moderate precision but struggles with high recall.
    - Improvements could involve balancing the dataset or experimenting with other classification algorithms.

Random Forest – Structured Data

Random Forest Model Performance

1. **Initial Model**:
    - The Random Forest Classifier was trained with the following parameters:
        - **Number of Trees (n_estimators)**: 100
        - **Maximum Depth (max_depth)**: 10
        - **Random Seed (random_state)**: 42
    - **Accuracy**: The model achieved an accuracy of **67%**, providing a baseline for further optimization.
    - **Classification Metrics**:
        - Precision: 0.66 (Class 0), 0.68 (Class 1)
        - Recall: 0.69 (Class 0), 0.65 (Class 1)
        - F1-Score: 0.67 (Class 0), 0.66 (Class 1)
2. **Confusion Matrix**:
    - **Class 0 (Unpopular Articles)**:
        - True Positives (TP): 4090 correctly predicted as unpopular.
        - False Negatives (FN): 1837 incorrectly predicted as popular.
    - **Class 1 (Popular Articles)**:
        - True Positives (TP): 3874 correctly predicted as popular.
        - False Negatives (FN): 2086 incorrectly predicted as unpopular.
    - **Insights**:
        - The model showed a slight bias towards misclassifying popular articles as unpopular.
        - Balancing the dataset or feature engineering might improve recall for Class 1.

Hyperparameter Tuning

1. **Parameter Grid**:
    - The following parameters were tuned using **Grid Search with 5-Fold Cross-Validation**:
        - n_estimators: [50, 100, 200]
        - max_depth: [5, 10, 15, None]
        - min_samples_split: [2, 5, 10]
        - min_samples_leaf: [1, 2, 4]
2. **Results**:
    - **Best Parameters**:
        - n_estimators: 200
        - max_depth: 15
        - min_samples_split: 5
        - min_samples_leaf: 2
    - **Best Cross-Validated Accuracy**: **68.3%**

Random Forest Confusion Matrix:

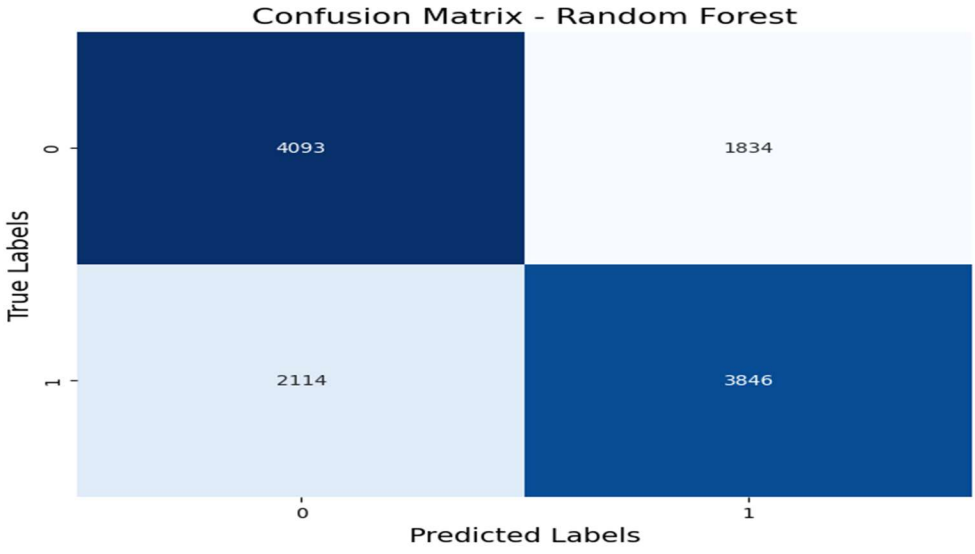

Figure 1. 17 Random Forest Confusion Matrix

The confusion matrix evaluates the performance of the Random Forest model optimized using random search.

Key Observations:

1. **Correct Predictions**:
    - True Negatives (TN): 4093 instances correctly classified as class 0.
    - True Positives (TP): 3846 instances correctly classified as class 1.
2. **Misclassifications**:
    - False Positives (FP): 1834 instances of class 0 misclassified as class 1.
    - False Negatives (FN): 2114 instances of class 1 misclassified as class 0.

Observations:

- The model demonstrates good performance, with a strong balance between correctly identifying both classes.
- To further reduce misclassification, fine-tuning hyperparameters or using feature selection could be beneficial.

Feature Importances - Random Forest

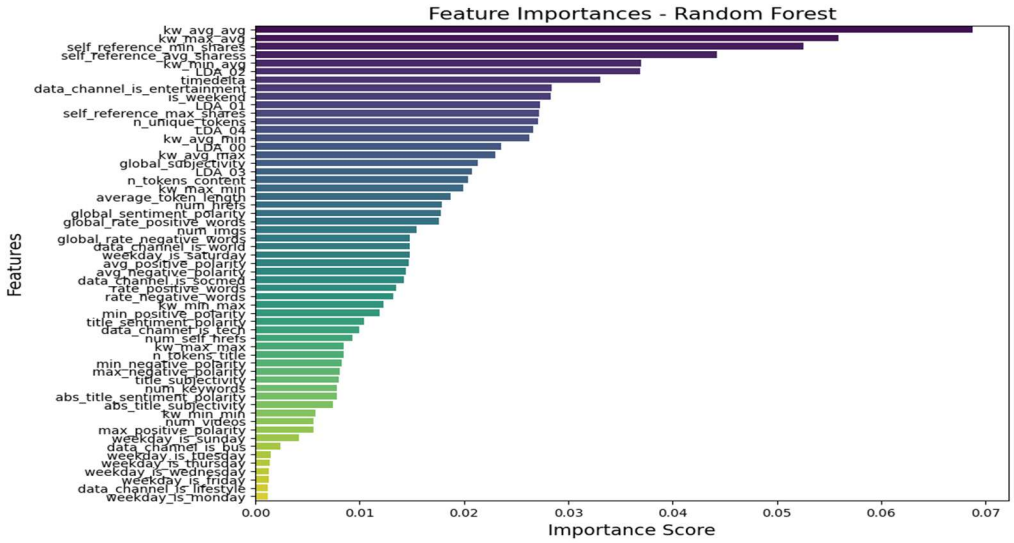

Figure 1.18 Feature Importances - Random Forest

This bar chart visualizes the importance of each feature in the Random Forest model optimized using random search.

Key Observations:

1. **Top Influential Features**:
    - kw_avg_avg, kw_max_avg, and kw_min_avg are the most significant predictors in the dataset, indicating that keyword metrics play a crucial role in determining the popularity of online news.
    - Features like self_reference_min_shares and self_reference_avg_shares also show high importance, suggesting that user interactions with the content are critical.
2. **Lesser Impact Features**:
    - Temporal features such as specific weekdays (weekday_is_monday, weekday_is_friday) and low-polarity measures have minimal impact on the model's decision-making process.
    - This indicates that these features do not strongly influence news popularity, compared to other metrics.

Confusion Matrix - Best Random Forest

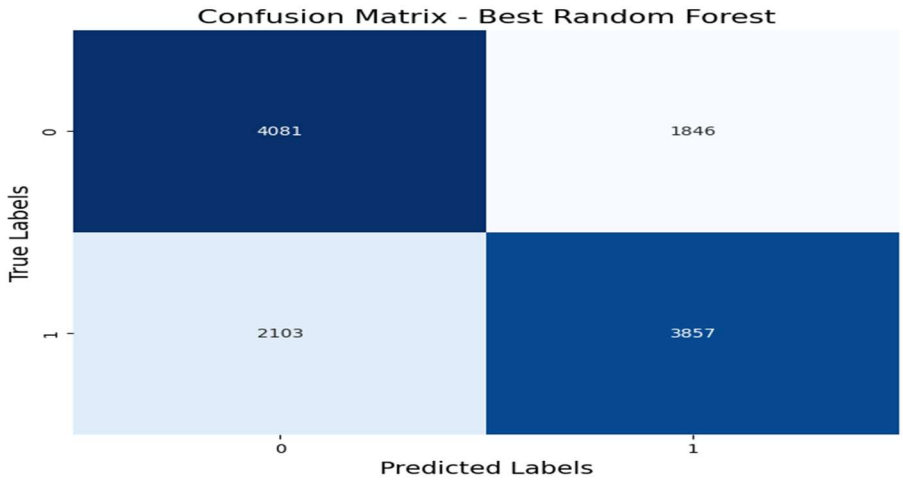

Figure 1.19 Confusion Matrix - Best Random Forest

Key Observations:

1. **True Positives and True Negatives:**
    - The model correctly classified 4081 instances as class 0 (true negatives) and 3857 instances as class 1 (true positives). These indicate the accurately predicted cases where news is popular (1) or not (0).
2. **False Positives and False Negatives:**
    - The model incorrectly classified 1846 instances as class 1 when they were actually class 0 (false positives).
    - Similarly, 2103 instances were misclassified as class 0 when they were truly class 1 (false negatives).

ROC Curve - Best Random Forest Model

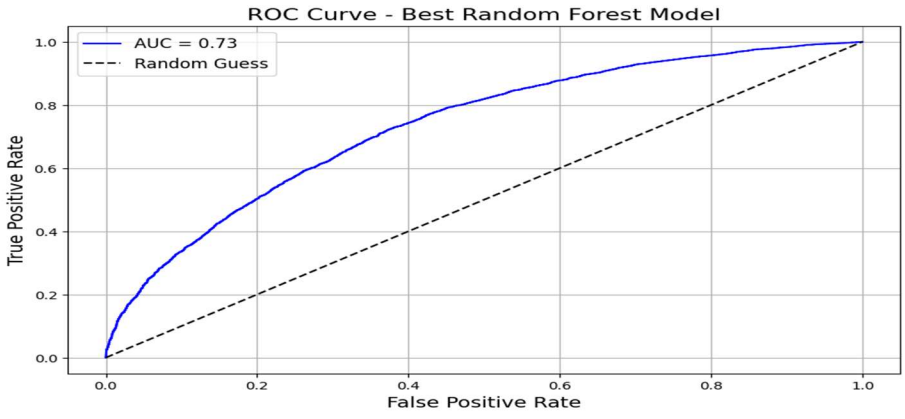

Figure 1.20 ROC Curve - Best Random Forest Model

Key Observations:

1. **AUC Score**:
    - The Area Under the Curve (AUC) value of 0.73 indicates a reasonably good ability of the model to distinguish between the two classes (popular and not popular news). The closer the AUC is to 1, the better the model's performance.
2. **Model's Performance**:
    - The ROC curve demonstrates a better-than-random performance (random being represented by the diagonal dashed line with an AUC of 0.5). The curve's rise above the diagonal highlights the model's skill in balancing true positives and false positives. However, further feature engineering or model tuning could help achieve higher AUC values.

Model Accuracy Comparison between Decision tree and Random forest:

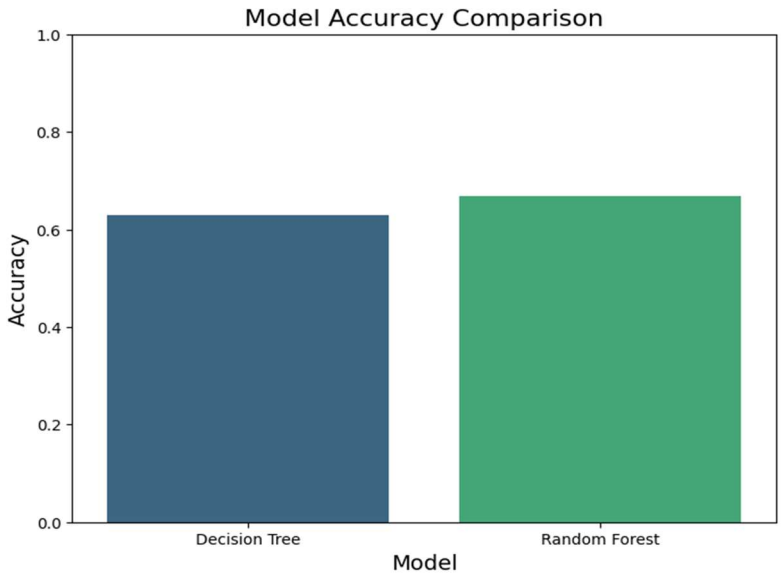

Figure 1.21 Model Accuracy Comparison

Key Observations:

1. **Performance Difference**:
    - The Random Forest model outperforms the Decision Tree model in terms of accuracy. This improvement highlights the advantage of using an ensemble method like Random Forest over a single-tree approach for better generalization and reduced overfitting.
2. **Insights**:
    - The higher accuracy of Random Forest reflects its ability to leverage multiple decision trees and aggregate their outputs, leading to a more robust predictive model. Fine-tuning parameters for Random Forest further contributed to its superior performance compared to the Decision Tree.

Model ROC curve Comparison between Decision tree and Random forest:

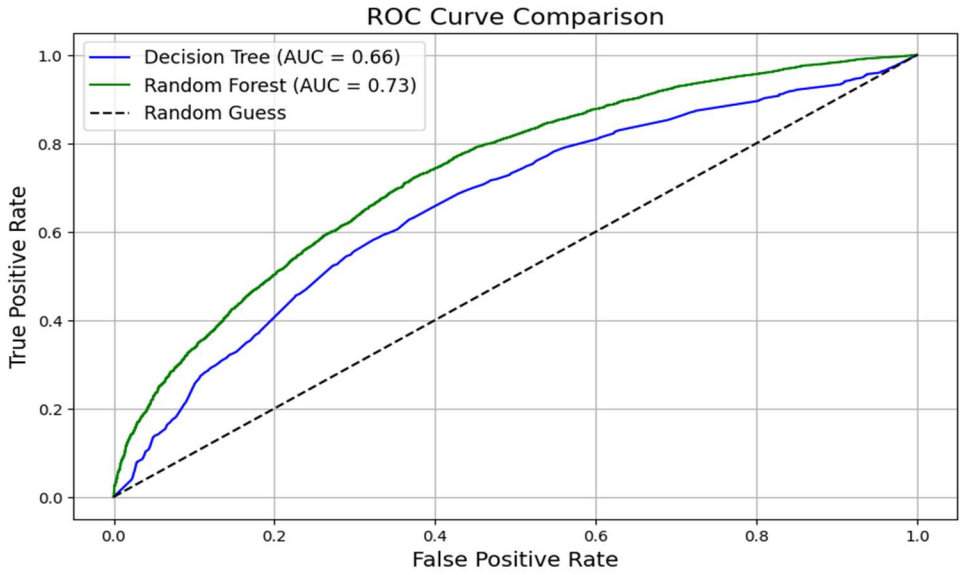

Figure 1.22 ROC Curve Comparison

Key Observations:

1. **AUC Scores**:
    - The Random Forest model has a higher AUC score (0.73) compared to the Decision Tree model (0.66), indicating better classification performance.
    - Both models perform better than random guessing (AUC = 0.5), as reflected by the curves staying above the diagonal line.
2. **Insights**:
    - The steeper initial rise of the Random Forest ROC curve shows its stronger ability to minimize false positives while maximizing true positives. This demonstrates the advantage of ensemble techniques for improved predictive capabilities.

SVM Support Vector Machine (SVM) – Structured Data

SVM Model Training

Feature Scaling

- The features were scaled using **StandardScaler** to standardize them to a mean of 0 and a standard deviation of 1.
- Scaling is crucial for SVM, as the algorithm is sensitive to the magnitude of feature values.

Model Parameters

- **Kernel**: Radial Basis Function (RBF) – Suitable for non-linear decision boundaries.
- **C**: 1 – Controls the trade-off between achieving a low error on the training data and minimizing model complexity to avoid overfitting.
- **Gamma**: 'scale' – Automatically adjusts gamma based on the number of features.

SVM Model Performance

Accuracy

- **Test Accuracy**: **67%**
    - Indicates the model correctly classified 67% of the test samples.

Classification Report

The classification report provides a detailed analysis of the model's precision, recall, and F1-score for each class:

Table 1.2 Classification Report for SVM Model Performance

Class	Precision	Recall	F1-Score	Support
0 (Unpopular)	**0.66**	**0.70**	**0.68**	5927
1 (Popular)	**0.68**	**0.64**	**0.66**	5960
Overall	**0.67**	**0.67**	**0.67**	11887

- **Precision**: Indicates the percentage of correctly predicted instances out of all predictions for a class.
- **Recall**: Indicates the percentage of actual instances of a class correctly predicted by the model.
- **F1-Score**: Provides a harmonic mean between precision and recall, balancing their trade-offs

Confusion Matrix – SVM

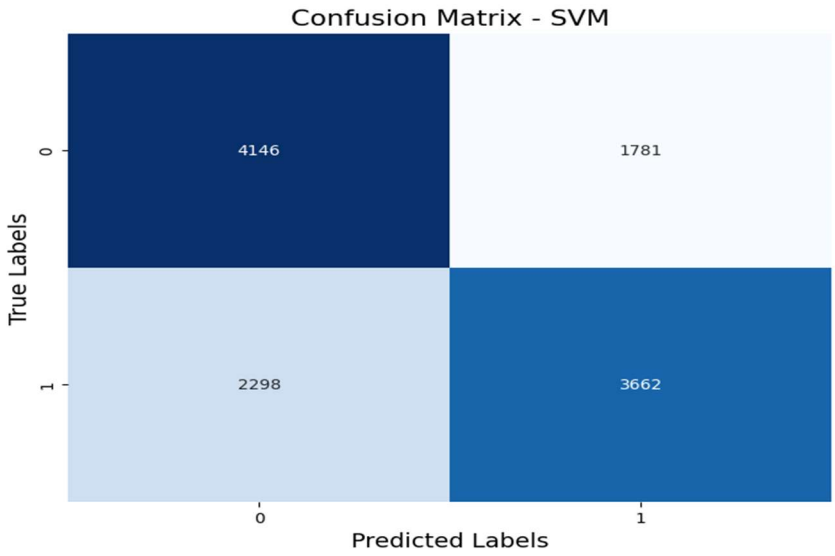

Figure 1.23 Confusion Matrix – SVM

Key Observations:

1. **True Positives and True Negatives**:
    - True Negatives (class 0 correctly predicted): 4146
    - True Positives (class 1 correctly predicted): 3662
2. **False Positives and False Negatives**:
    - False Positives (class 0 incorrectly predicted as 1): 1781
    - False Negatives (class 1 incorrectly predicted as 0): 2298

Observations:

The SVM model shows a balance between predicting class 0 and class 1, with better handling of true positives but some misclassifications on both ends. This suggests the SVM's decision boundary may need further tuning for better performance.

Feedforward Neural Network

Model Architecture

1. **Input Layer**:
    - Size: Number of predictors/features (56).
2. **Hidden Layers**:
    - Layer 1: 128 neurons with ReLU activation.
    - Layer 2: 64 neurons with ReLU activation.
3. **Output Layer**:
    - Size: 2 neurons (for binary classification).
    - Activation: Softmax (via CrossEntropyLoss).
4. **Optimizer**:
    - Adam optimizer with a learning rate of **0.0001**.
5. **Loss Function**:
    - CrossEntropyLoss for multi-class classification.
6. **Training Details**:
    - Epochs: 10
    - Batch Size: 32

Model Training

Training Loss

- The model was trained for 10 epochs, and the training loss decreased steadily, indicating effective learning.

Hyperparameter Tuning Report: Feedforward Neural Network

Parameter Space

1. **Batch Sizes**: [16, 32, 64, 128]
2. **Learning Rates**: [0.1, 0.01, 0.001, 0.0001]
3. **Hidden Layer Configurations**: [256, 128, 64]
4. **Activation Functions**:

- ReLU
- Tanh
- Sigmoid

5. **Number of Hidden Layers**:
    - 1 (single layer)
    - 2 (two layers)
    - 3 (three layers)
6. **Optimizers**:
    - Adam
    - SGD (Stochastic Gradient Descent)
    - RMSprop
7. **Number of Iterations**: 16 random configurations
8. **Epochs per Configuration**: 5

Best Feed forward Neural Network

After evaluating 16 configurations, the best combination of hyperparameters achieved the highest accuracy on the test dataset:

- **Batch Size**: 32
- **Learning Rate**: 0.001
- **Hidden Layer Size**: 128
- **Activation Function**: ReLU
- **Number of Hidden Layers**: 3
- **Optimizer**: Adam

1. **Training Duration**:
    - The model was trained for **20 epochs**.
    - Training loss steadily decreased across epochs, demonstrating effective learning and convergence.

2. **Training Loss**:
    - **Initial Loss (Epoch 1)**: 0.6736
    - **Final Loss (Epoch 20)**: 0.5989
    - This consistent reduction in loss indicates that the model's weights were updated effectively during training, without significant overfitting.
    -

Visual Analysis for Feedforward Neural Network

Training Loss Curve

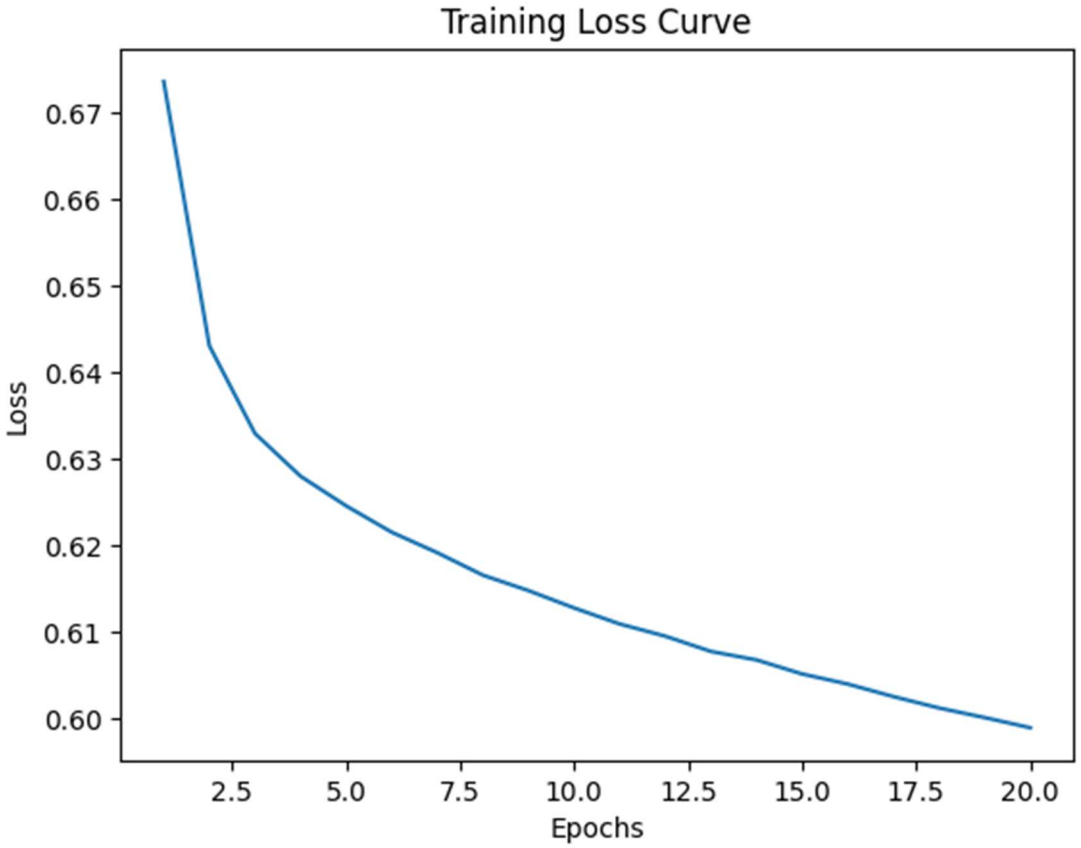

Figure 1.24 Training Loss Curve for Feedforward Neural Network

Overview:

- The training loss curve shows a steady and consistent decline in loss across 20 epochs, indicating effective learning by the model.

- **Insights**:
    - The model started with a loss of **0.6736** at epoch 1 and converged to a loss of **0.5989** at epoch 20.
    - No significant signs of overfitting or underfitting are observed, as the loss decreases smoothly without plateauing too early.

Confusion Matrix

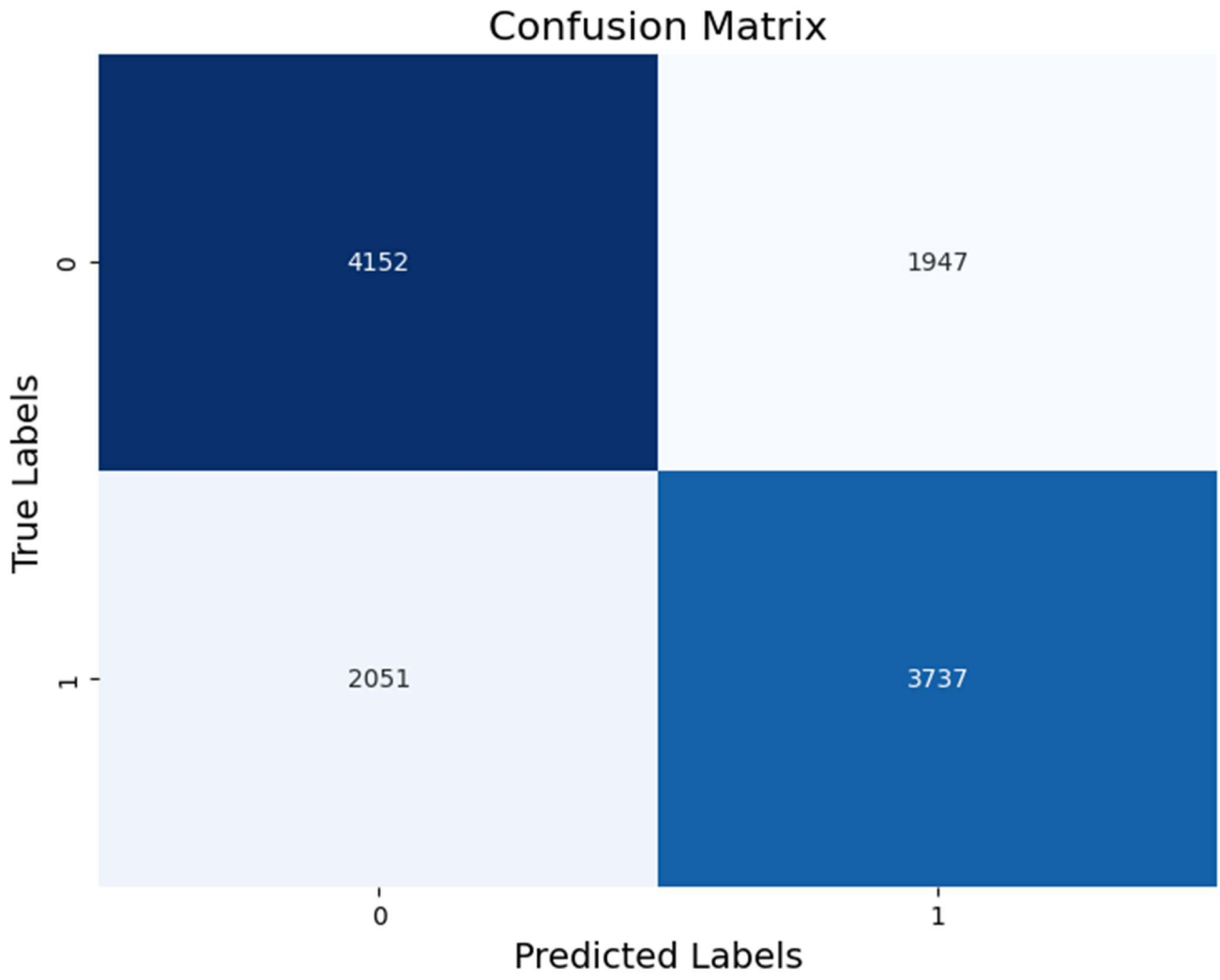

Figure 1.25 Confusion Matrix for Feedforward Neural Network

Table 1.3 Confusion Matrix

Predicted Class	0 (Unpopular)	1 (Popular)
Actual 0	4152	1947
Actual 1	2051	3737

- **Key Observations**:
  - **True Negatives (4152)**: Unpopular articles correctly classified.
  - **False Positives (1947)**: Unpopular articles misclassified as popular.
  - **True Positives (3737)**: Popular articles correctly classified.
  - **False Negatives (2051)**: Popular articles misclassified as unpopular.
- **Insights**:
  - The model performs slightly better for Class 0 (Unpopular) than for Class 1 (Popular), as indicated by fewer false positives than false negatives.

Receiver Operating Characteristic (ROC) Curve

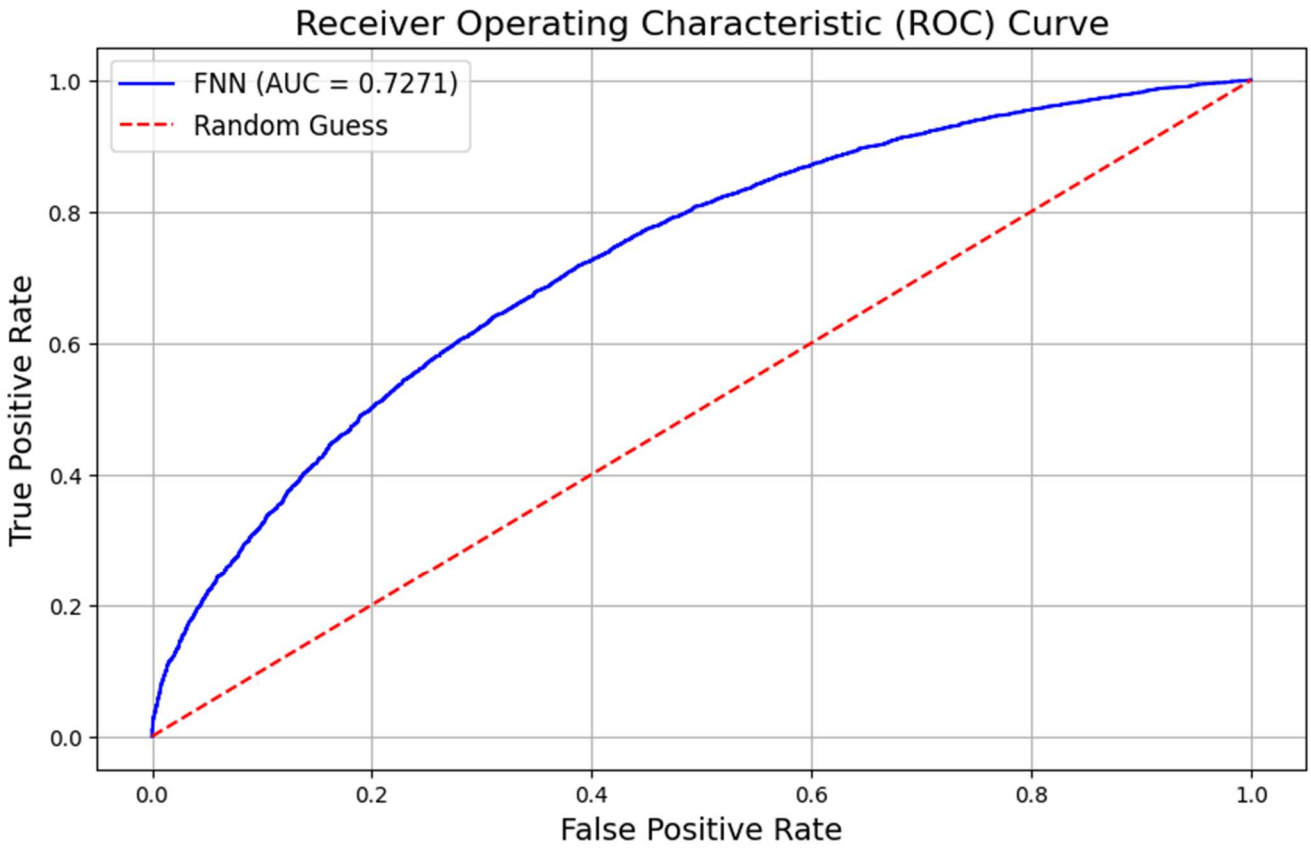

Figure 1.26 ROC Curve for Feedforward Neural Network

- **AUC Score**: **0.7271**
  - The Area Under the Curve (AUC) value of **0.7271** indicates moderate classification performance, significantly better than random guessing (AUC = 0.5).
- **Key Observations**:
  - The ROC curve illustrates a favorable trade-off between the true positive rate (sensitivity) and false positive rate for various thresholds.
  - The model demonstrates reasonable ability to distinguish between popular and unpopular articles.

Performance Metrics Summary

Classification Report

Table 1.4 Classification Report for Feedforward Neural Network

Metric	Class 0 (Unpopular)	Class 1 (Popular)	Overall
Precision	**0.67**	**0.66**	**0.66**
Recall	**0.68**	**0.65**	**0.66**
F1-Score	**0.68**	**0.65**	**0.66**
Accuracy			**0.66**

Conclusion

1. **Strengths**:
    - The model demonstrates steady convergence and achieves a reasonable accuracy of **66%** on the test dataset.
    - The AUC score of **0.7271** reflects a good ability to distinguish between classes.
2. **Limitations**:
    - High false negatives (2051) suggest the model struggles to correctly classify all popular articles, potentially missing some opportunities.

CNN – Structured Data

Convolutional Neural Network (CNN) Training and Evaluation

Model Training

1. **Training Configuration**:
    - **Input Size**: Matches the number of features in the dataset.
    - **Architecture**:
        - **Convolution Layers**:
            - Conv1d with 16 filters and kernel size of 3.
            - Conv1d with 32 filters and kernel size of 3.
        - **Fully Connected Layers**:
            - First layer with 128 neurons.
            - Output layer with 2 neurons (binary classification).
        - **Activation Function**: ReLU
        - **Dropout**: 50% (to prevent overfitting).
    - **Optimizer**: Adam
    - **Learning Rate**: 0.0005
    - **Loss Function**: CrossEntropyLoss
    - **Epochs**: 20
2. **Training Progress**:
    - The loss decreased steadily across epochs, showing consistent improvement in the model's optimization.

Training Loss Curve:

- Initial Loss (Epoch 1): 0.6371
- Final Loss (Epoch 20): 0.5370

CNN Model Evaluation Report

Training Performance

Training Loss Curve

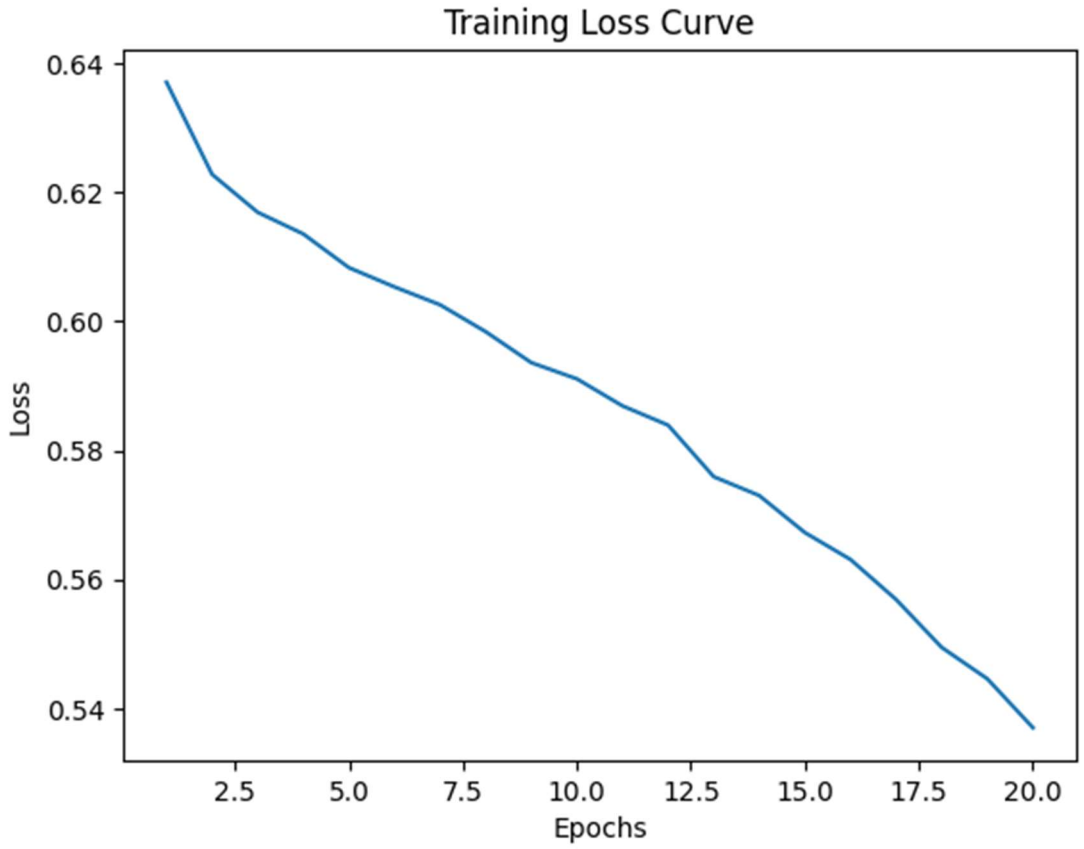

Figure 1.27 Convolutional Neural Network (CNN) Training Loss Curve

- The loss curve shows a consistent decline over 20 epochs, indicating the model learned effectively and steadily improved.
- Initial Loss (Epoch 1): 0.6371

- Final Loss (Epoch 20): 0.5370
- There are no significant signs of overfitting or underfitting, as the loss steadily decreases without plateauing too early or oscillating.

Confusion Matrix

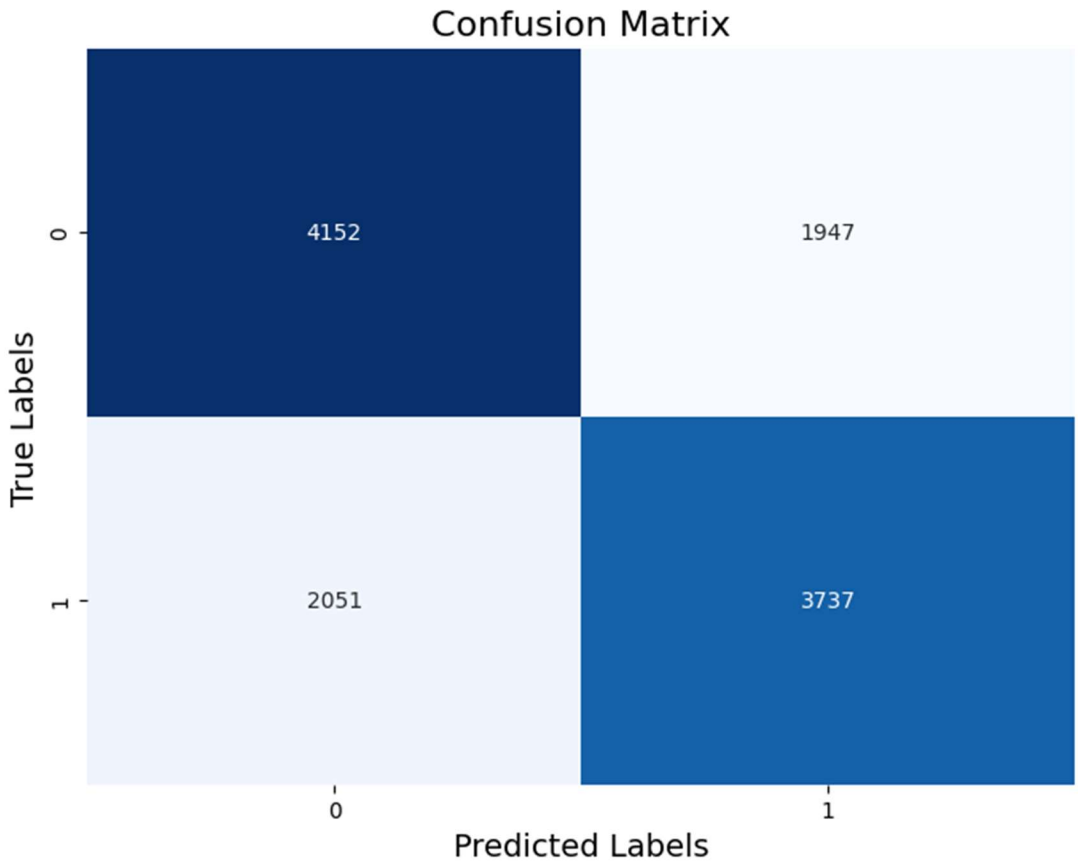

Figure 1.28 Convolutional Neural Network (CNN) Confusion Matrix

Table 1.5 CNN Confusion Matrix

Predicted Class	0 (Unpopular)	1 (Popular)
Actual 0	4095	2004
Actual 1	1998	3790

Key Observations:

- True Negatives (4095): Unpopular articles correctly classified.
- False Positives (2004): Unpopular articles misclassified as popular.
- True Positives (3790): Popular articles correctly classified.
- False Negatives (1998): Popular articles misclassified as unpopular.

Insights:

- The model performs slightly better on Class 0 (Unpopular) than Class 1 (Popular).
- High false negatives suggest some popular articles are not being correctly identified.

Classification Report

Table 1.6 CNN Classification Report

Metric	Class 0 (Unpopular)	Class 1 (Popular)	Overall
Precision	0.67	0.65	0.66
Recall	0.67	0.65	0.66
F1-Score	0.67	0.65	0.66
Accuracy			0.66

Interpretation:

- Precision: Out of the predicted labels, the model correctly identified 67% of unpopular articles and 65% of popular articles.
- Recall: The model captured 67% of all actual unpopular articles and 65% of all actual popular articles.
- F1-Score: Balanced scores for both classes indicate the model's ability to handle imbalanced data.

ROC Curve and AUC

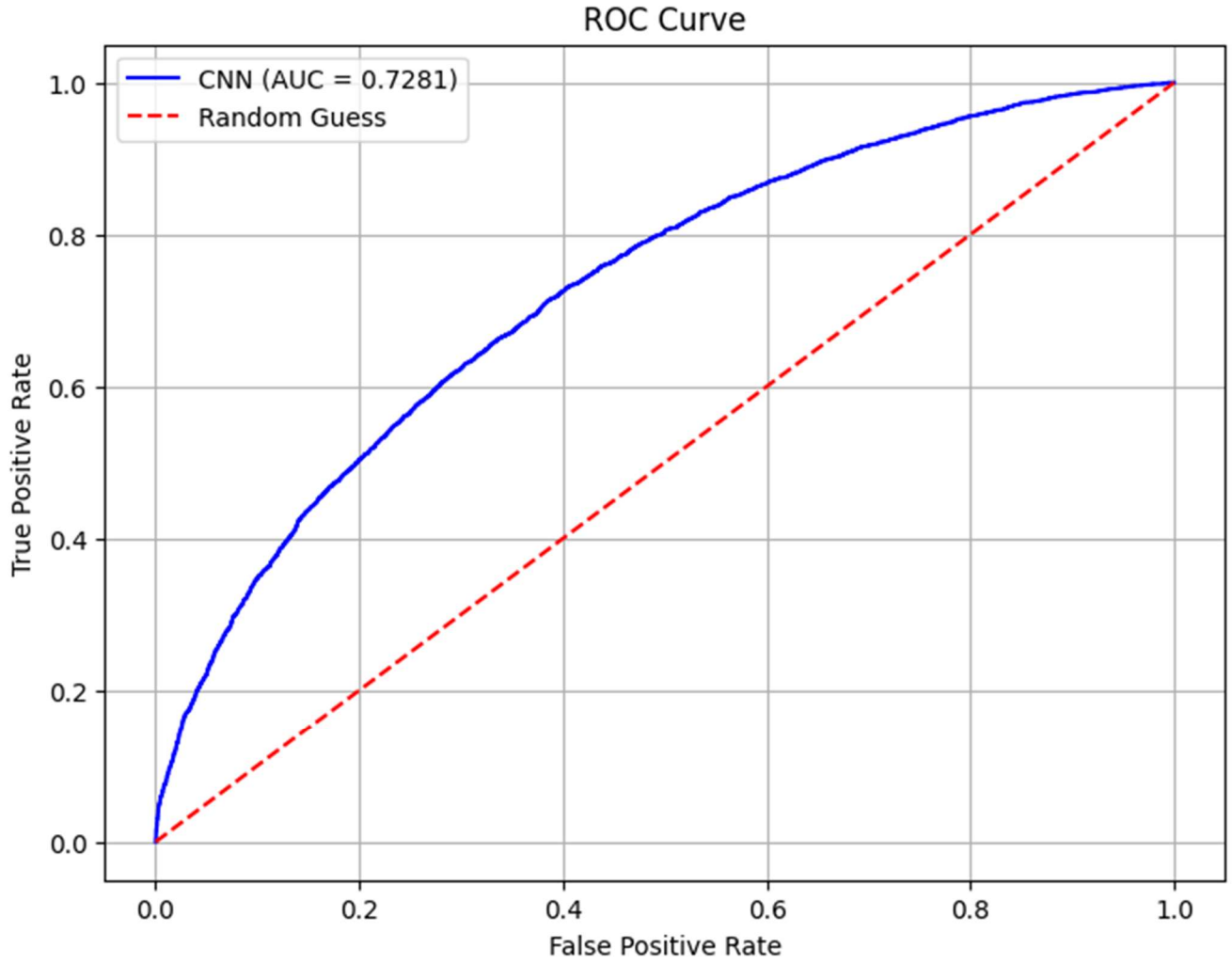

Figure 1.29 Convolutional Neural Network (CNN) ROC Curve

AUC Score: 0.7281

- The Area Under the Curve (AUC) score indicates moderate classification performance, with the model significantly outperforming random guessing (AUC = 0.5).
- The ROC curve shows a good trade-off between sensitivity (true positive rate) and specificity (false positive rate).

Insights and Recommendations

Strengths:

- The model demonstrates steady improvement in loss during training, reflecting effective learning.
- Moderate classification performance with balanced F1-scores for both classes.
- The AUC score of 0.7281 highlights the model's reasonable ability to distinguish between classes.

Limitations:

- The relatively high false negatives (1998) indicate challenges in correctly identifying all popular articles.
- Accuracy of 66% suggests room for improvement, especially in recall for Class 1 (Popular).

Recommendations:

- Feature Engineering:
  - Add or refine features to better capture the patterns distinguishing popular from unpopular articles.
- Regularization:
  - Increase dropout or apply L2 regularization to improve generalization.
- Class Imbalance Handling:
  - Use SMOTE or similar techniques to address class imbalance and reduce false negatives.
- Hyperparameter Optimization:
  - Tune learning rate, dropout rate, and convolutional filter sizes for better performance.
- Comparison with Other Models:

- Compare CNN performance against simpler models like Random Forest or Logistic Regression for benchmarking.

Conclusion – CNN Structured data

The CNN model achieved an accuracy of 66% and an AUC score of 0.7281, demonstrating moderate classification performance. While the model successfully captures some patterns in the data, further enhancements in architecture, feature engineering, and regularization are necessary to achieve higher performance and reduce false negatives.

Conclusion and Comparative Analysis

1. Machine Learning Models

Decision Tree

- **Accuracy**: **62.87%**
- **Key Observations**:
  - Decision Tree models were moderately effective, achieving balanced accuracy but prone to overfitting without proper pruning.
  - **Strengths**:
    - Interpretable with clear feature importance insights.
    - Simple and quick to train.
  - **Limitations**:
    - Lower generalization performance.
    - Sensitive to overfitting, especially on imbalanced data.

Random Forest

- **Accuracy**: **67%**
- **Key Observations**:
    - Improved performance compared to the Decision Tree, benefiting from ensemble learning and reduced overfitting.
    - **Strengths**:
        - Handles non-linearity and captures complex patterns well.
        - Robust to overfitting due to multiple trees.
    - **Limitations**:
        - Slower training and prediction compared to simpler models.
        - Limited interpretability compared to a single tree.

2. Deep Learning Models

Feedforward Neural Network (FNN)

- **Accuracy**: **66.21%**
- **AUC Score**: **0.7271**
- **Key Observations**:
    - The FNN demonstrated stable training with consistent loss reduction across epochs.
    - **Strengths**:
        - Captures non-linear relationships effectively.
        - Performs better on larger datasets with sufficient features.
    - **Limitations**:

        - Limited feature interpretability.
        - Performance slightly affected by class imbalance (higher false negatives).

Convolutional Neural Network (CNN)

- **Accuracy**: **66%**
- **AUC Score**: **0.7281**
- **Key Observations**:
    - The CNN showed slightly better performance than the FNN in terms of AUC, with stable training and convergence.
    - **Strengths**:
        - Better feature extraction through convolutional layers.
        - More effective in handling structured data and detecting complex patterns.
    - **Limitations**:
        - Computationally intensive, requiring more resources than FNN and Random Forest.
        - Susceptible to performance drop if hyperparameters are not optimized.

3. Comparative Analysis

Table 1.7 CNN Comparative Analysis

Model	Accuracy	AUC	Strengths	Limitations
Decision Tree	62.87%	-	Simple, interpretable, fast training.	Prone to overfitting, sensitive to class imbalance.
Random Forest	67%	-	Robust, captures non-linearity, reduced overfitting.	Slower training and prediction, limited interpretability.
Feedforward NN	66.21%	0.7271	Captures non-linearity, effective for larger datasets.	Limited interpretability, requires careful hyperparameter tuning.
CNN	66%	0.7281	Effective feature extraction, handles structured data well.	Computationally intensive, susceptible to performance drop without careful optimization.

4. Machine Learning vs. Deep Learning

Strengths of Machine Learning Models

1. **Simplicity and Interpretability**:
    - Models like Decision Tree and Random Forest provide clear insights into feature importance and decision-making.
2. **Efficiency**:
    - Faster training and inference compared to deep learning, especially on smaller datasets.

Strengths of Deep Learning Models

1. **Powerful Pattern Recognition**:
    - FNN and CNN excel in capturing complex patterns and non-linear relationships in data.
2. **Scalability**:
    - Deep learning models perform better as the size of the dataset and complexity of the

Conclusion

While machine learning models (Random Forest) are more interpretable and computationally efficient, deep learning models (FNN and CNN) excel in handling complex patterns and larger datasets. The choice between them depends on the dataset size, computational resources, and need for interpretability. CNN performed slightly better than FNN, showcasing its effectiveness in feature extraction for structured data.

CHAPTER 2 - Unstructured Data

ABSTRACT

Facial expression recognition is a crucial aspect of human-computer interaction and has wide-ranging applications in emotion analysis, mental health monitoring, and security. In this research, we evaluate several machine learning techniques to classify facial expressions into discrete emotion categories such as happiness, sadness, anger, and surprise. The dataset used for this study is the FER2013 dataset, which contains over 35,000 grayscale images of facial expressions across seven emotion classes. We employ various preprocessing techniques such as image resizing, normalization, and data augmentation to enhance the model's performance. Significant features contributing to the analysis include facial landmarks, pixel intensity distributions, and convolutional feature maps. Our study leverages advanced deep learning models like convolutional neural networks (CNNs) to investigate their effectiveness in accurately recognizing emotions. The results illustrate the potential of these techniques in improving the accuracy and reliability of facial expression recognition systems, providing valuable insights for real-world applications.

UNSTRUCTURED DATA

1. Introduction to the Project

1.1 Background

Facial Expression Recognition (FER) is an essential component of human-computer interaction with significant applications in fields such as emotion analysis, mental health monitoring, social robotics, and security. The ability to recognize and interpret human emotions through facial expressions is pivotal for enhancing machine intelligence and for applications where understanding human emotions is critical.

Our study focuses on developing machine learning (ML) models to classify facial expressions into discrete emotion categories **such as happiness, sadness, anger, and surprise.** We utilize the **MMA Facial Expression** dataset, a comprehensive collection of facial images encompassing **six emotion classes.** Recent advancements in computer vision and deep learning have significantly improved the potential for accurate emotion recognition. By leveraging predictive insights from these models, we aim to enhance user experiences, strengthen safety protocols, and provide valuable data for therapeutic applications.

1.2 Problem Statement

Recognizing human emotions through facial expressions presents challenges due to variability in expressions across individuals, lighting conditions, occlusions, and facial orientations. Traditional methods often fail to generalize across diverse datasets, resulting in reduced accuracy in real-world scenarios.

This project addresses these challenges by employing advanced preprocessing techniques and deep learning models, specifically Convolutional Neural Networks (CNNs), to accurately classify facial expressions. Key factors such as pixel intensity distribution and convolutional feature maps are explored to enhance the robustness and accuracy of FER systems.

1.3 Impact on Business

Facial expression recognition has significant implications in various industries, including healthcare, entertainment, education, and security. For example, mental health monitoring systems can use FER to identify early signs of stress or depression, while emotion-aware applications in gaming and virtual reality can dynamically tailor user experiences to enhance engagement and satisfaction.

In the business realm, FER can revolutionize customer service by enabling systems to respond empathetically, improving customer satisfaction and retention. Additionally, FER systems enhance safety by identifying unusual emotional behaviors in public spaces. This study highlights critical factors influencing FER accuracy and contributes to building reliable systems, reducing deployment costs, and increasing operational effectiveness across industries.

2. Dataset and Domain

2.1 Dataset

The MMA Facial Expression dataset is a benchmark for facial expression recognition, consisting of over 12,000 labeled facial images representing six emotion classes: angry, happy, neutral, sad, surprised, and fearful. Each image is a colored representation with resolutions suitable for modern deep learning models.

The dataset is divided into subsets for training and testing, ensuring effective evaluation of machine learning models. Key characteristics of the dataset include:

Image Content: Each image depicts a human face expressing one of the **six** emotion classes.

Emotion Diversity: Emotions are balanced to minimize class bias during training.

Preprocessing Requirement: The dataset requires transformations like resizing, normalization, and data augmentation.

Application Versatility: It is suitable for various applications, including healthcare, security, and

human-computer interaction.

Data Dictionary

Table 2.1 Data Dictionary

Column Name	Data Type	Description	Information	Null Values
image_path	**String**	**File path of the image**	**Links to facial expression images**	**0%**
emotion	**String**	**Encoded labels representing emotion classes**	**6 categories: angry, happy, neutral, sad, surprised, fearful**	**0%**

Emotion Labels:

The emotion column encodes facial expressions as:

- **angry**
- **happy**
- **neutral**
- **sad**
- **surprised**
- **fearful**

Image Format:

- Each image is a colored facial representation.

Data Cleaning and Preprocessing:

- The dataset requires resizing and normalization for optimal training performance.

Class Distribution:

- The dataset is approximately balanced, ensuring suitability for machine learning.
- By preprocessing and analyzing these attributes, we aim to enhance the model's performance in

recognizing facial expressions in real-world applications.

5. Feature Engineering

5.1 Data Transformation

Facial image data requires transformation to enhance model performance. Images from the MMA Facial Expression dataset are resized to fixed pixel dimensions, normalized to the range [0, 1], and augmented to address class imbalance and improve robustness. Transformations, including rotation, flipping, and cropping, simulate real-world variations.

5.2 Scaling

Feature scaling ensures consistency in pixel intensity values across all images, aiding effective learning. Using MinMaxScaler, pixel values are normalized to the range [0, 1], facilitating faster convergence in gradient-based optimization methods.

5.3 Feature Selection

Feature selection identifies relevant pixel regions and patterns critical to emotion classification. CNNs inherently emphasize significant regions, such as the eyes, mouth, and eyebrows, through convolutional feature maps. This process reduces redundancy and enhances model performance.

MODELING

6.1 Data Splitting

The dataset, pre-split into test, train, and validation subsets, was used to build a model for predicting facial expressions. For this case, we utilized the test and train datasets directly, eliminating the need for additional data splitting. This approach ensured the consistency of data preparation for the facial expression prediction task.

6.2 Modeling

To address the "No Free Lunch" theorem in machine learning, various models were experimented with to identify the most effective approach for facial expression recognition. Convolutional Neural

Networks (CNNs) were primarily employed due to their ability to capture spatial hierarchies in image data. Pre-trained models like VGG16 and ResNet50 were fine-tuned to leverage their robust feature extraction capabilities.

Models were evaluated using metrics such as accuracy, precision, recall, and F1 scores. A holdout test set was used to ensure an unbiased performance assessment on unseen data. Additionally, subsets of the data were utilized for testing and hyperparameter tuning. For the training data, a weighted method was applied to balance all the classes, ensuring equitable representation across the dataset for improved model performance.

Class Distribution Analysis - Training and Testing Data:

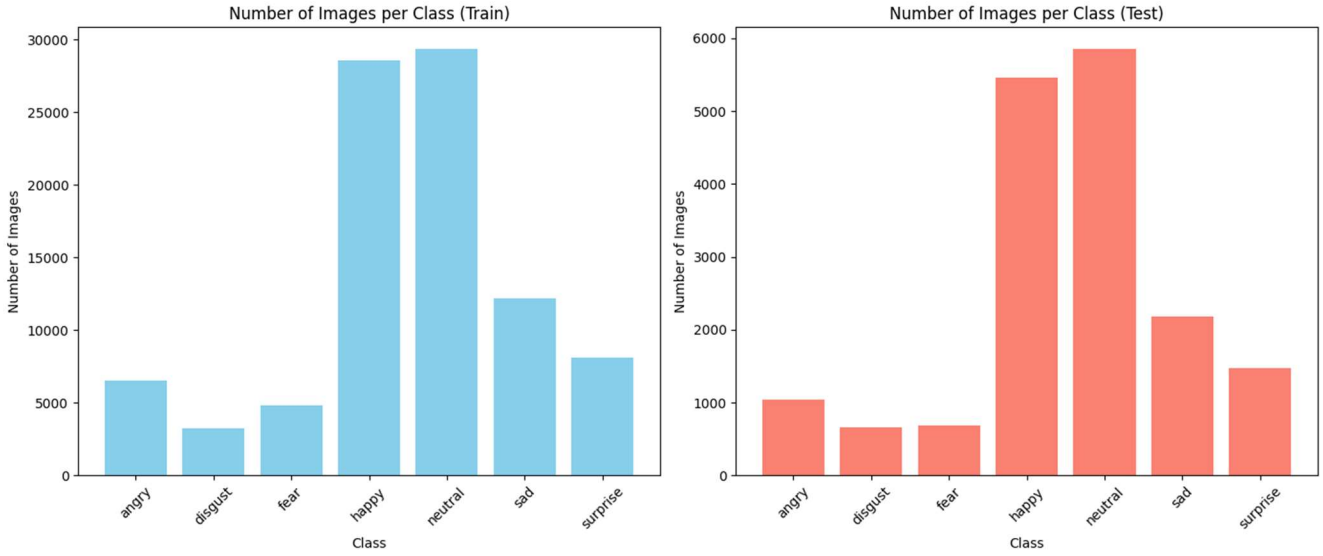

Figure 2.1 Class Distribution Analysis - Training and Testing Data

Key Observations:

Training Data Distribution:

- **Classes with High Representation**:
    - **Happy**: This class has the largest representation in the training dataset, with approximately 30,000 images.

- **Neutral**: Similarly, the neutral emotion class has a significant number of images, close to 30,000, contributing heavily to the overall training data.

- **Classes with Low Representation**:
  - **Disgust**: This class has the least representation, with fewer than 5,000 images.
  - **Surprise and Fear**: These classes are also underrepresented compared to happy and neutral classes.

Testing Data Distribution:

- **Classes with High Representation**:
  - **Happy** and **Neutral**: These classes dominate the testing dataset as well, following a similar distribution trend as the training data.
- **Classes with Low Representation**:
  - **Disgust**: This class again has the lowest representation in the testing dataset.
  - **Angry** and **Fear**: These classes are also underrepresented but have slightly better representation compared to disgust.

Insights:

1. **Imbalance in Class Representation**:
   - The dataset exhibits a clear class imbalance, with emotions like happy and neutral being overrepresented, while disgust, surprise, and fear are underrepresented.
   - This imbalance might lead to biased predictions, where the model may perform better on majority classes (happy, neutral) but struggle with minority classes (disgust, surprise).

2. **Impact on Model Performance**:
    - The imbalance in training data could result in higher accuracy for dominant classes but lower precision and recall for minority classes.
    - Techniques like data augmentation for minority classes or class-weighted loss functions in the model training phase can mitigate these issues.
3. **Scope of Improvement**:
    - **Oversampling**: Augmenting the underrepresented classes (disgust, surprise) to balance the dataset.
    - **Data Augmentation**: Applying transformations like flipping, rotation, or noise addition to the minority class images.
    - **Class Weights**: Assigning higher weights to minority classes during model training to balance the impact of the imbalanced dataset.

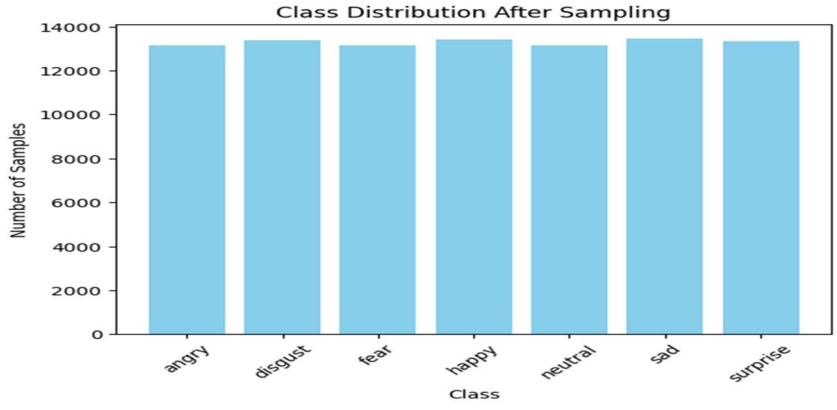

Figure 2.2 Class Distribution After Sampling

Key Observations:

- **Balanced Dataset**: After sampling, each class (angry, disgust, fear, happy, neutral, sad, and surprise) has an equal number of approximately 14,000 images, eliminating class imbalance.

- **Previously Underrepresented Classes**: Classes like **disgust** and **surprise** now have representation like the previously dominant classes, such as **happy** and **neutral**.

Insights:

1. **Improved Balance**:
    - Equal class representation ensures that the model will not be biased toward any emotion category.
2. **Impact on Performance**:
    - Better precision and recall for previously underrepresented classes (e.g., disgust, surprise).
    - Fairer evaluation of overall model accuracy across all classes.
3. **Techniques Applied**:
    - Oversampling for minority classes using data augmentation like rotation and flipping.

    Balanced data leads to better generalization and more reliable model performance for facial expression recognition.

Evaluation of Model

7.1 Evaluation Metrics

Evaluation of facial expression recognition models involves the following metrics:

1. **Confusion Matrix**: A confusion matrix provides a detailed overview of model performance by highlighting the number of true positives (correctly classified emotions), true negatives, false positives, and false negatives.

2. **Accuracy**: Accuracy measures the percentage of correctly classified emotions.

   It is calculated as: Accuracy = (True Positives + True Negatives) / Total Samples

3. **Classification Report**: Precision, recall, and F1 scores for each emotion class provide deeper insights into model performance across all categories.

   - **Precision**: Proportion of correct positive predictions out of all predicted positives.

     Precision = (True PositivesTrue) / (Positives + False Positives)

   - **Recall**: Proportion of correctly identified positives out of all actual positives.

     Recall = (True PositivesTrue) / (Positives + False Negatives)

   - **F1 Score**: Harmonic mean of precision and recall.

     F1 Score = (2×Precision×Recall) / (Precision + Recall)

4. **ROC Curve and AUC**: The Receiver Operating Characteristic (ROC) curve represents the trade-off between true positive and false positive rates. The Area Under the Curve (AUC) summarizes the model's ability to distinguish between classes, with values closer to 1 indicating better performance.

Machine Learning Models

Logistic Regression:

Feature Extraction and Data Preparation

- **Pre-trained ResNet18**: Used as a feature extractor, removing its classification layer to extract high-level features from image data.
- **Feature Transformation**: Input images were resized to 224x224, normalized, and converted to tensors for compatibility with the model.
- **Data Loaders**: The dataset was processed using PyTorch DataLoader with batch size 64 for training and testing, resulting in:
    - **Train Features Shape**: (92968,512), (92968, 512)
    - **Test Features Shape**: (17356,512), (17356, 512)

Model Training

- **Classifier**: A Logistic Regression model was trained using the extracted features. The model training encountered a **Convergence Warning**, indicating the need for:
    - Increasing max_iter for the logistic regression model.
    - Scaling the data for better optimization.

Model Evaluation

1. **Confusion Matrix**:
    - Significant misclassification occurred across multiple classes.
    - Class 4 (Emotion: Likely representing happiness or neutral) showed the highest precision and recall, indicating it was the easiest class for the model to recognize.

- Classes like 0, 1, and 2 (e.g., emotions like anger, disgust, or sadness) had low precision and recall, highlighting difficulties in recognizing these emotions.

2. **Classification Report**:
    - **Accuracy**: 47%, reflecting moderate performance.
    - **Macro Average** (Unweighted Average):
        - Precision: 35%
        - Recall: 30%
        - F1-Score: 31%
    - **Weighted Average** (Considering Class Imbalance):
        - Precision: 44%
        - Recall: 47%
        - F1-Score: 44%
3. **Insights**:
    - **Class Imbalance**: Lower performance for minority classes indicates imbalance issues in the dataset.
    - **Model Limitations**: Logistic regression may not fully exploit the extracted features' non-linear relationships. A more complex classifier, such as a Neural Network, could improve performance.

ROC Curve Analysis

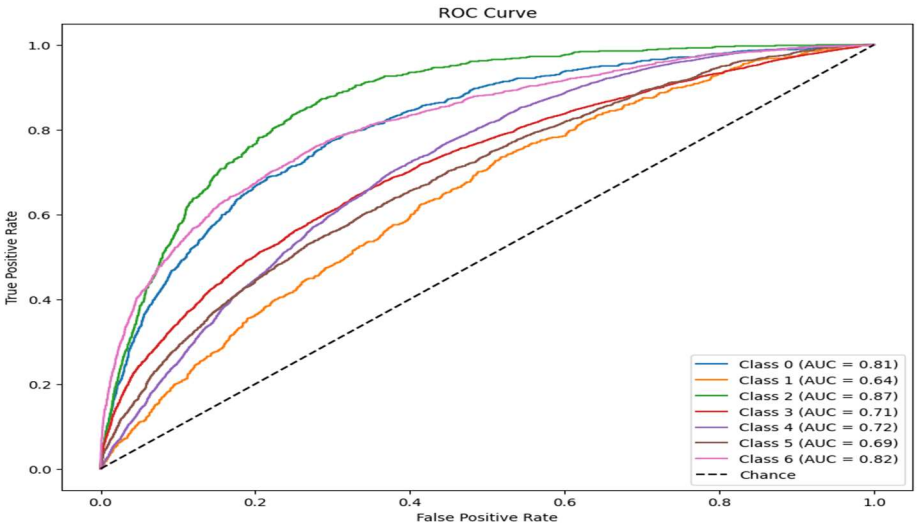

Figure 2.3 Logistic Regression ROC Curve

Key Observations:

- **Class 2 (AUC = 0.87)**: This class shows the highest area under the curve (AUC), indicating the model's strong ability to differentiate true positives from false positives for this emotion category.
- **Class 6 (AUC = 0.82)**: The second-best performing class, suggesting reliable predictions.
- **Class 1 (AUC = 0.64)**: This class has the lowest AUC, indicating the model struggles to distinguish between true positives and false positives for this category.

Insights:

1. **Overall Performance**:
    - The AUC values across all classes are above 0.60, showing reasonable predictive ability, with some classes performing better than others.
2. **Challenges**:
    - Lower AUC for Class 1(angry) suggests potential confusion with other classes, likely due to overlapping features in the dataset.

3. **Opportunities for Improvement**:
    - Apply targeted techniques like class-specific augmentation or weighted loss functions to boost performance for low-performing classes (e.g., Class 1).

The ROC curve highlights the model's strengths and weaknesses, guiding further optimizations to achieve consistent performance across all emotion categories.

Confusion Matrix Analysis

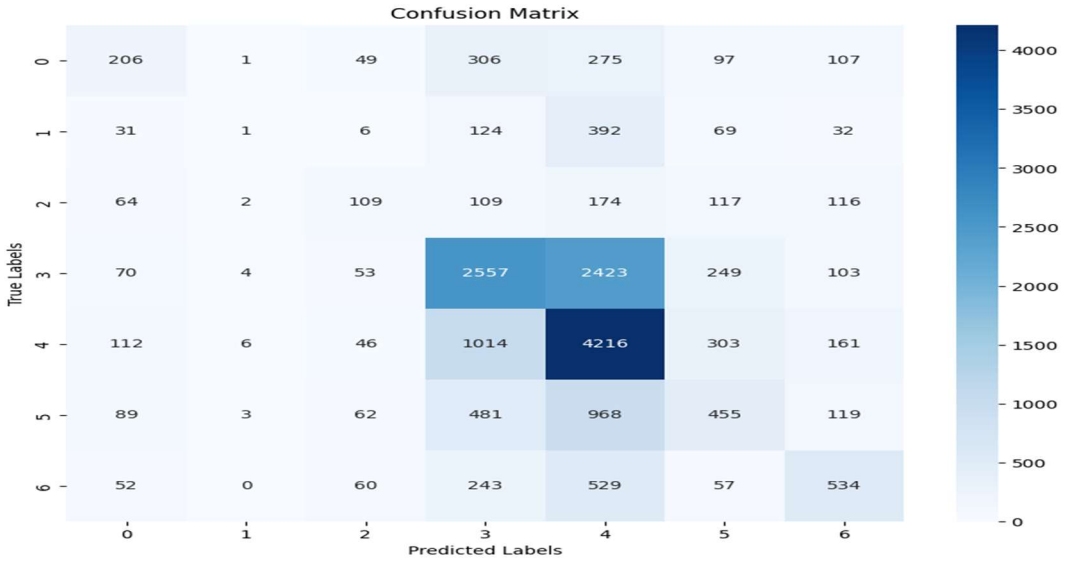

Figure 2.4 Logistic Regression Confusion Matrix

Key Observations:

1. **True Positives**:
    - The model performs well in identifying class 3 (happy) and class 4 (neutral), with the highest numbers of true positives at **2557** and **4216**, respectively.
    - This indicates the model is most confident in recognizing these two emotion categories.
2. **False Positives and False Negatives**:

- **Class 0 (angry)**: Significant confusion with classes 3 and 4, with **306** and **275** false positives, respectively.
- **Class 1 (disgust)**: Frequently misclassified as class 4 (neutral), with **392** false positives.
- **Class 2 (fear)**: Has notable confusion with classes 0 and 4, leading to misclassifications.

3. **Misclassifications**:
   - **Class 5 (sad)**: A considerable number of samples misclassified as class 4 (neutral), with **968** false positives.
   - **Class 6 (surprise)**: Often confused with class 3 (happy), with **243** false positives.

Insights:

1. **Class-Specific Performance**:
   - The model has strong predictive performance for classes like **happy** and **neutral**, as indicated by the high true positive values.
   - Classes like **angry**, **disgust**, and **surprise** show lower accuracy due to significant misclassifications, likely caused by overlapping facial features.
2. **Improvements Needed**:
   - **Class Balance**: Further class-specific augmentation for underperforming classes (e.g., angry, disgust).
   - **Feature Engineering**: Enhance feature extraction to better differentiate overlapping features among similar classes (e.g., neutral vs sad, happy vs surprise).
3. **Model Refinements**:
   - Adjusting hyperparameters or experimenting with ensemble methods to reduce false positives and negatives for minority classes.
   - Exploring advanced loss functions (e.g., focal loss) to prioritize underrepresented classes.

The confusion matrix highlights the model's strengths in recognizing dominant emotion classes while also indicating the need for improvements in minority class prediction to achieve balanced performance across all categories.

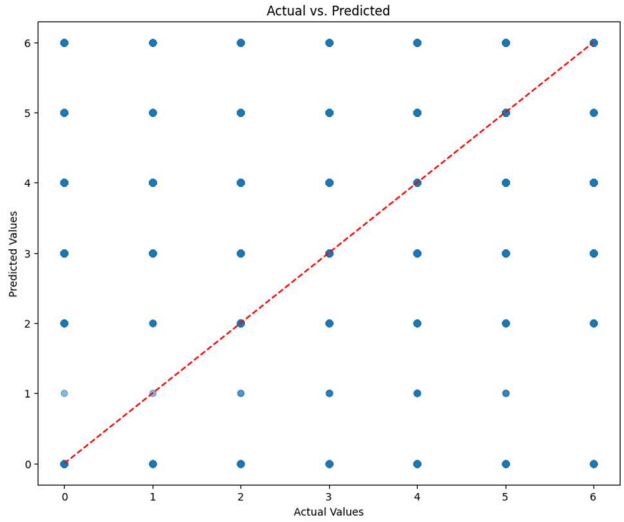

Figure 2.5 Actual vs Predicted Plot

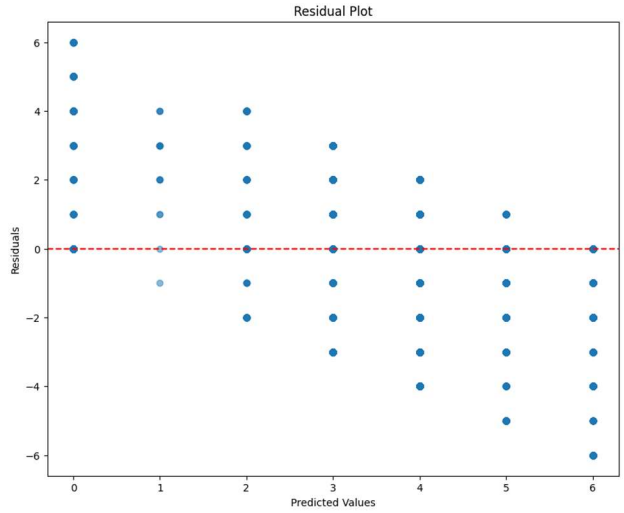

Figure 2.6 Residual Plot

Actual vs. Predicted Plot

Key Observations:

1. **Diagonal Pattern**:
    - The points along the red dashed line represent cases where the actual values match the predicted values.
    - A significant number of points are aligned with the line, indicating correct predictions by the model.
2. **Misclassifications**:
    - Points scattered away from the diagonal represent misclassified samples.
    - Classes with denser points away from the diagonal may require further attention to improve model accuracy.

Residual Plot

Key Observations:

1. **Centered Residuals**:
    - The residuals are centered around 0, indicating that the model has a balanced prediction distribution without systematic overestimation or underestimation.
2. **Spread of Residuals**:
    - The residuals appear evenly distributed across predicted values, suggesting the model's errors are consistent for different classes.
    - However, a higher spread in certain areas may indicate difficulty in predicting specific classes accurately.

Insights:

1. **Model Strengths**:
    - The clustering of points along the diagonal in the Actual vs. Predicted plot demonstrates that the model is capable of predicting many classes accurately.
2. **Areas for Improvement**:
    - Points scattered away from the diagonal in the Actual vs. Predicted plot and the larger spread in the residual plot suggest misclassification in certain classes.
    - Investigating feature importance and enhancing preprocessing techniques (e.g., data augmentation for minority classes) could reduce these misclassifications.
3. **Next Steps**:
    - Experiment with fine-tuning hyperparameters to improve predictions for classes with higher residual errors.
    - Apply class-specific weighting in the loss function to address misclassifications in underperforming classes.

These plots highlight the model's general effectiveness while suggesting areas for further optimization to enhance performance.

Random Forest:

Analysis of Random Forest Classifier Performance

Model Training and Predictions:

- A Random Forest Classifier was trained using 100 decision trees.
- The model's performance was evaluated using the test data.

Classification Report:

1. **Accuracy:**

   Overall accuracy: 43%, which is slightly below moderate performance.

2. **Class-wise Performance:**

   - Class 4 (likely a dominant emotion class) achieved the highest recall (0.79) and a good F1-score (0.55), indicating strong recognition of this class.
   - Classes 0, 1, and 5 (likely minority or less distinctive emotions) showed low recall and F1-scores, highlighting significant difficulty in recognizing these emotions.
   - Class 3 had a balanced recall (0.42) and precision, leading to a moderate F1-score (0.43).

3. **Macro and Weighted Averages:**

   - Macro Average: Precision = 0.56, Recall = 0.24, F1-score = 0.25. These averages show the model struggles uniformly across most classes.
   - Weighted Average: Precision = 0.47, Recall = 0.43, F1-score = 0.38. This reflects an imbalance in performance, with better results for dominant classes.

Confusion Matrix Insights:

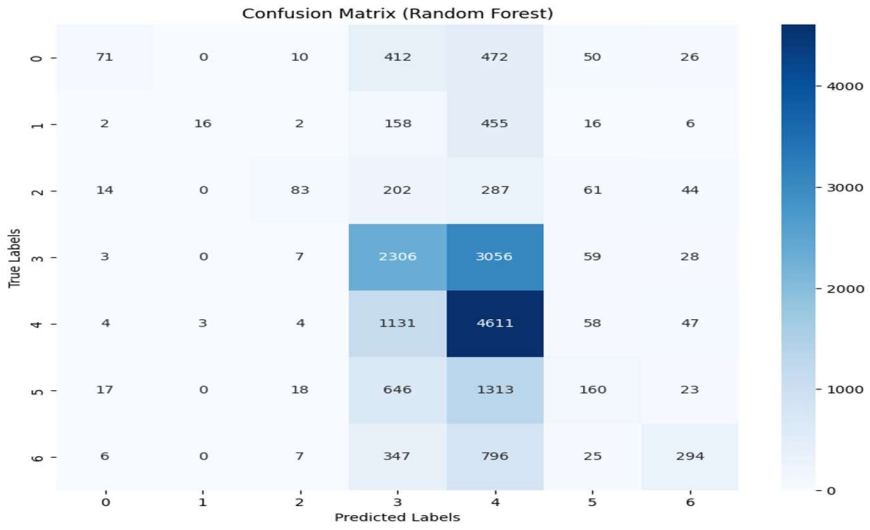

Figure 2.7 Confusion Matrix For Random Forest

Key Observations:

1. **True Positives**:
    - **Class 4 (Neutral)**: The model performs well for class 4, with **4611 true positives**, indicating high confidence in predicting this class correctly.
    - **Class 3 (Happy)**: Shows strong performance as well, with **3056 true positives**.
2. **False Positives and False Negatives**:
    - **Class 0 (Angry)**: Frequently misclassified as class 3 (Happy) and class 4 (Neutral), with **412** and **472 false positives**, respectively.
    - **Class 1 (Disgust)**: Struggles to differentiate from class 4 (Neutral), leading to **455 false positives**.
    - **Class 6 (Surprise)**: Misclassifications are frequent with class 3 (Happy) and class 4 (Neutral), suggesting overlap in feature representation.

Insights:

1. **Class-Specific Performance**:
    - Strong performance for dominant classes like **Neutral** and **Happy**, which are easier to distinguish due to their distinct features.
    - Weaker performance for minority classes like **Angry** and **Disgust**, with significant misclassifications across other similar classes.
2. **Challenges**:
    - The model faces difficulty in distinguishing overlapping emotions, particularly between **Angry**, **Happy**, and **Neutral**.
    - Minority classes receive fewer correct predictions due to class imbalance and shared feature patterns.

- Many classes have high false negatives, indicating the model struggles to identify certain emotions correctly.
- Class 4 shows fewer misclassifications compared to other classes, demonstrating better recognition.

Suggestions for Improvement:

1. **Class Balancing:**
    - Use oversampling or undersampling techniques to balance underrepresented classes.
    - Incorporate class weights into the Random Forest model to prioritize minority classes.
2. **Hyperparameter Tuning:**
    - Experimented with the number of trees (n_estimators) and maximum depth (max_depth) to improve model performance.
    - Adjusted the min_samples_split and min_samples_leaf parameters for better splits and reduced overfitting.

➢ This analysis highlights areas of strength (Class 4) and significant challenges (Classes 0, 1, and 5), providing a roadmap for further model optimization.

Conclusion:

The Random Forest model performs well for majority classes like **Neutral** and **Happy**, but struggles with underrepresented and overlapping classes. Further optimizations are needed to improve balanced performance across all categories.

Feature Importance Analysis - Random Forest

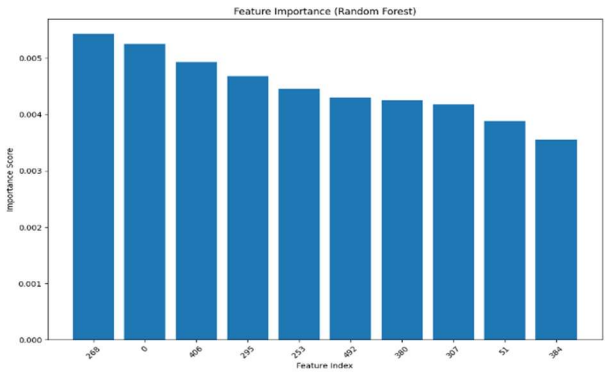

Figure 2.8 Feature Importance Analysis - Random Forest

Key Observations:

1. **Top Features**:
    - Feature at index **268** has the highest importance score, closely followed by feature at index **0**.
    - Other significant features include indices **406**, **295**, and **492**, indicating their strong influence on the model's predictions.
2. **Importance Scores**:
    - All features shown have importance scores in a similar range (~0.004–0.005), suggesting that these features contribute almost equally to the decision-making process.
    - Features at indices **51** and **384** have relatively lower importance compared to the others in the top 10.

Insights:

1. **Impact on Model Performance**:
    - The high importance of specific features suggests that these contribute strongly to differentiating between emotion classes.

- Features like **268** and **0** could be tied to key visual regions (e.g., eyes, mouth) or pixel intensity values critical for recognizing certain emotions.

2. **Further Optimization**:
   - The identified important features can be prioritized during preprocessing or feature engineering to enhance model performance.
   - Unimportant features could potentially be dropped to reduce dimensionality and improve computational efficiency without significant performance loss.

Conclusion:

The feature importance chart highlights the most influential features for the Random Forest model. Leveraging these insights can help fine-tune the model and further optimize its predictive performance.

Deep Learning Models

Feedforward Neural Network:

Model Architecture

- Neural Network Design:
    - Input size: 3×224×2243 (flattened RGB image input).
    - Two hidden layers with 128 and 64 neurons, respectively.
    - Activation function: ReLU for non-linearity.
    - Output size: 7 classes (one for each emotion category).
- Optimizer: Adam optimizer with a learning rate of 0.0001.
- Loss Function: CrossEntropyLoss, suitable for multi-class classification tasks.

Training Phase

- Training Loop:
    - Model trained for 5 epochs.
    - Training loss is calculated and printed for monitoring.
    - The optimizer.step() updates the weights based on the gradients computed during backpropagation.

Testing Phase

- Evaluation Function:
    - The model is evaluated on both training and test datasets.
    - Predictions and labels are collected to compute performance metrics.
    - Metrics include:

- Confusion Matrix: Shows the distribution of predictions vs. actual labels for each class.
- Classification Report: Provides precision, recall, F1-score, and support for each class.

Analysis of Feedforward Neural Network Performance:

Training Phase

- The confusion matrix and classification report highlight the following key points:
  - Accuracy: 53% on the training set.
  - Class 4 (likely a dominant class) has the highest recall (77%) and a balanced F1-score (60%), indicating better performance for this class.
  - Classes 0, 1, and 5 show low recall and F1-scores, indicating difficulty in recognizing these categories.
  - Macro Average: Precision = 37%, Recall = 24%, F1-score = 25%, reflecting low overall class performance balance.
  - Weighted Average: Precision = 53%, Recall = 53%, F1-score = 49%, indicating an imbalance favoring dominant classes.

Testing Phase

- Accuracy: 43% on the test set, showing a moderate decrease in performance compared to training.
- Confusion Matrix Insights:
  - Class 4 achieved the highest recall (76%), similar to the training set.
  - Other classes, particularly 0, 1, and 5, exhibit low precision and recall, indicating poor generalization to unseen data.

- Classification Report:
    - Macro Average: Precision = 30%, Recall = 18%, F1-score = 18%, emphasizing challenges in balanced class recognition.
    - Weighted Average: Precision = 43%, Recall = 43%, F1-score = 38%.

Confusion Matrix Analysis - Training and Testing Data

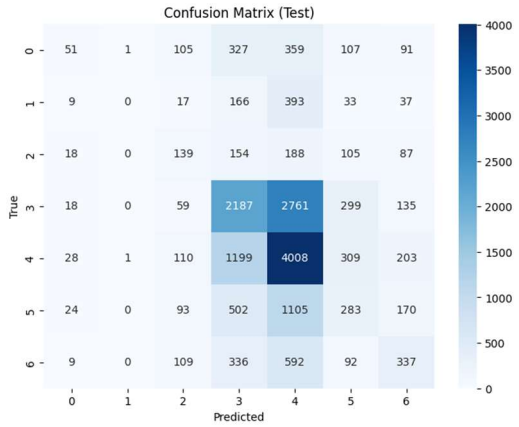

Figure 2.9 Confusion Matrix (Test) for Feedforward Neural Network

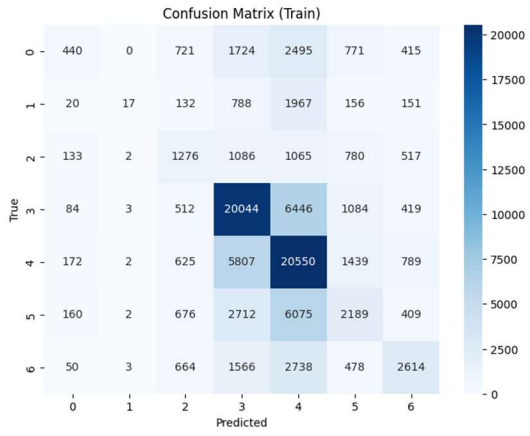

Figure 2.10 Confusion Matrix (Train) for Feedforward Neural Network

Training Data Confusion Matrix

1. **True Positives**:
    - **Class 3 (Happy)** and **Class 4 (Neutral)** have the highest number of true positives at **20,044** and **20,550**, respectively, indicating strong performance for these dominant classes.
    - Moderate performance is observed for **Class 5 (Sad)** with **6,075 true positives**.
2. **Misclassifications**:
    - **Class 0 (Angry)** and **Class 1 (Disgust)** are frequently misclassified as **Class 4 (Neutral)**, with **2,495** and **1,967 false positives**, respectively.
    - **Class 6 (Surprise)** shows significant overlap with **Class 4 (Neutral)** and **Class 3 (Happy)**, with **2,738** and **1,566 false positives**, respectively.

Testing Data Confusion Matrix

1. **True Positives**:
    - **Class 4 (Neutral)** again shows the highest number of true positives with **4,008**, followed by **Class 3 (Happy)** with **2,187**.
    - Other classes, such as **Class 5 (Sad)** and **Class 6 (Surprise)**, have lower true positives compared to training data.
2. **Misclassifications**:
    - Similar to training data, **Class 0 (Angry)** and **Class 1 (Disgust)** are often confused with **Class 4 (Neutral)**, leading to high false positives.
    - **Class 2 (Fear)** and **Class 6 (Surprise)** have significant misclassifications with neighboring classes, suggesting overlapping features.

Insights

1. **Model Strengths**:
    - The model performs well on dominant classes like **Happy** and **Neutral**, with high true positive values in both training and testing datasets.

Conclusion:

The confusion matrices reveal that the model achieves strong performance for dominant classes but struggles with minority classes due to overlapping features and class imbalance. Addressing these challenges can improve overall model performance and balance predictions across all emotion categories.

Hyperparameter Tuning for Feedforward Neural Networks:

This study involves hyperparameter tuning for a Feedforward Neural Network (FNN) to enhance the performance of a model designed for emotion classification. Key aspects of the tuning process include varying the network structure, learning rate, activation functions, batch size, and optimization methods. Below is a summary of the various parameters tested and insights gained.

Greedy Search:

1. Hidden Layer Configurations

- One Hidden Layer: Configured with 128 neurons, providing a simpler model structure for faster training.
- Two Hidden Layers: Configurations like (128, 64) and (256, 128) were tested to capture more complex relationships in the data.
- Three Hidden Layers: Configured as (128, 64, 32), allowing for deeper learning of hierarchical features.

2. Learning Rates

- Various learning rates, including [0.1, 0.01, 0.001, 0.0001], were tested to find the optimal balance between convergence speed and stability.
- A learning rate of 0.0001 provided the best trade-off, avoiding instability or slow learning.

3. Activation Functions

- Common activation functions, including:
    - ReLU: Efficient and computationally fast, showing consistent performance.
    - Tanh: Better performance for balanced and scaled data.
    - Sigmoid: Found to underperform due to vanishing gradient issues in deeper networks.

4. Batch Sizes

- Batch sizes tested were [16, 32, 64, 128].
- Smaller batch sizes provided finer gradient updates but increased training time.
- A batch size of 128 was found to balance computational efficiency and performance.

5. Optimizers

- Adam: Outperformed others due to its adaptive learning rates and robustness to noisy gradients.
- SGD: Required careful tuning of learning rates and momentum to achieve reasonable results.
- RMSprop: Worked well but slightly underperformed compared to Adam.

Randomized Hyperparameter Tuning

- Parameters were randomly sampled for 16 iterations:
    - Batch Sizes: Randomly selected from [16, 32, 64, 128].
    - Learning Rates: Randomly selected from [0.1, 0.01, 0.001, 0.0001].
    - Hidden Layer Configurations: (256, 128), (128, 64), etc.

- Activation Functions: ReLU, Tanh, Sigmoid.
- Number of Layers: 1, 2, or 3.
- Optimizers: Adam, SGD, RMSprop.

The optimal configuration achieved the best performance with:

- Batch Size: 128
- Learning Rate: 0.0001
- Hidden Layers: (128, 64)
- Activation Function: ReLU
- Optimizer: Adam

Insights

- Increasing the number of hidden layers allowed for more complex feature extraction but required careful tuning of learning rates and batch sizes.
- The Adam optimizer combined with the ReLU activation function consistently provided superior results.
- Randomized hyperparameter tuning proved efficient in identifying the best configuration compared to a grid search.

Best model Analysis:

Analysis of Training and Evaluation Results

Training Progress

- **Loss Trend**:
  - The **training loss** decreases steadily from 1.6343 in Epoch 1 to 1.2021 in Epoch 15, indicating consistent learning.

- The **test loss** decreases initially but begins to plateau around Epoch 8, with minor fluctuations. This suggests the model may start overfitting after this point.

Test Performance

1. **Overall Accuracy**:
    - The model achieves an accuracy of **43%** on the test set, reflecting moderate performance.
2. **Class-wise Performance**:
    - **Class 4** has the best performance with:
        - Precision: 43%
        - Recall: 74%
        - F1-score: 54%
    - **Class 3** shows reasonable performance with:
        - Precision: 50%
        - Recall: 41%
        - F1-score: 45%
    - Classes 0, 2, 5, and 6 have low precision and recall, indicating difficulties in predicting these categories.
    - **Classes 7, 8, and 9** have no true or predicted samples, resulting in undefined metrics and zero contributions to performance.
3. **Macro vs. Weighted Averages**:
    - **Macro Average**:
        - Precision: 22%
        - Recall: 18%
        - F1-score: 18%

        - Reflects significant imbalance, as poor-performing or empty classes negatively impact the averages.
    - **Weighted Average**:
        - Precision: 40%
        - Recall: 43%
        - F1-score: 39%
        - Weighted by class support, this metric better reflects the overall model performance for dominant classes.

Challenges and Observations

1. **Class Imbalance**:
    - Imbalance in the dataset is evident, as dominant classes (e.g., Classes 3 and 4) contribute heavily to accuracy, while minority classes struggle.
2. **Overfitting**:
    - The plateauing of test loss and divergence from training loss suggests the model may be overfitting the training data.
3. **Low Recall for Some Classes**:
    - Poor recall for classes like 1, 2, and 6 indicates the model struggles to identify these emotions accurately.

Analysis of Training and Evaluation Results with Visualization

Training and Test Loss

- The **training loss** shows a steady decrease across epochs, starting at 1.6343 and reducing to 1.2021, indicating consistent learning.

- The **test loss** initially decreases but starts fluctuating after Epoch 7, eventually increasing slightly from Epoch 13 onwards. This suggests potential **overfitting**, as the model continues to improve on the training set but loses generalization on the test set.

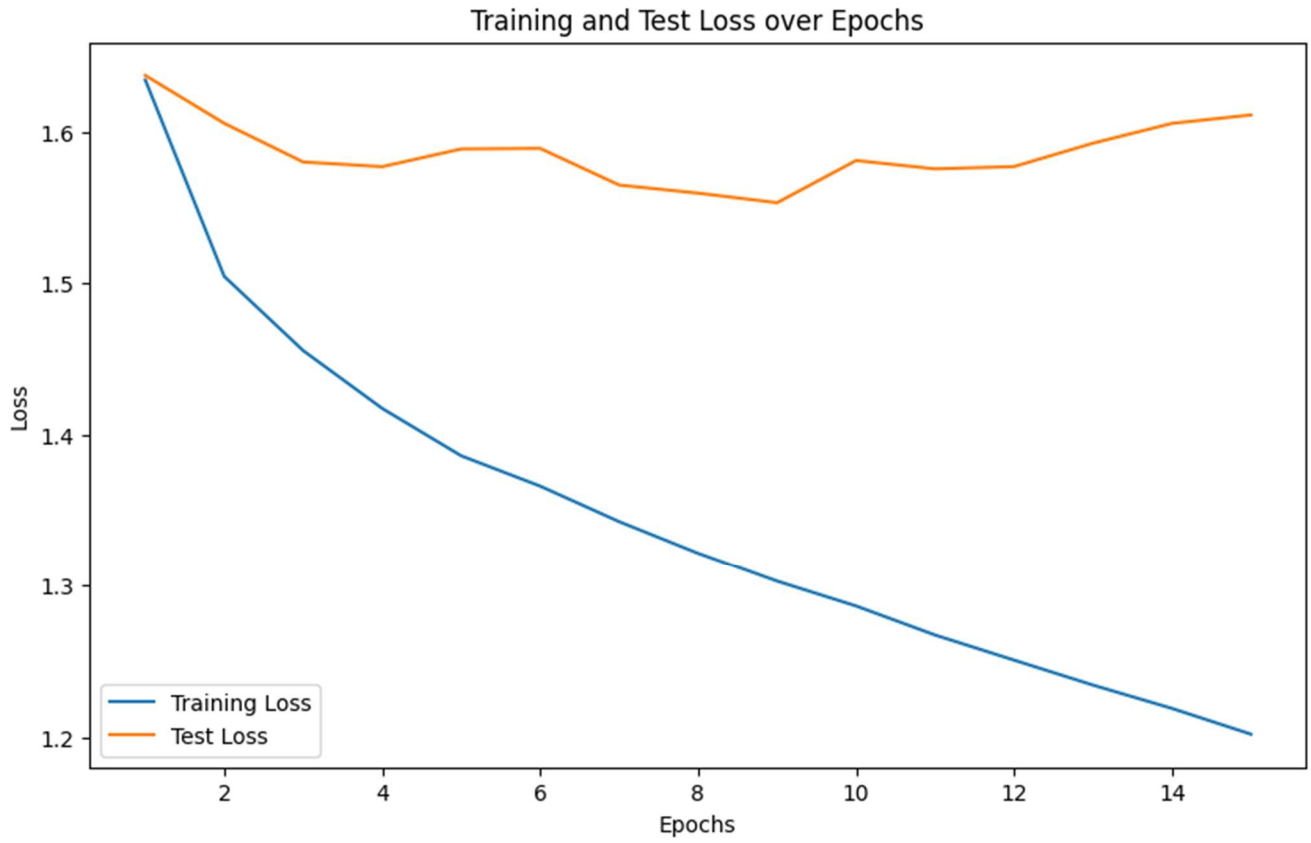

Figure 2.11 Best Model Feedforward Neural Network ROC Curve

ROC Curves for Each Class

- The **AUC (Area Under the Curve)** values provide a quantitative measure of the model's ability to distinguish between classes:
    - **Class 2** achieved the highest AUC of 0.81, indicating strong performance in identifying this class.
    - **Class 1** has the lowest AUC of 0.59, reflecting difficulty in distinguishing this class from others.
- The ROC curves show moderate separation for most classes, but some overlap suggests limited discriminative ability for certain categories.

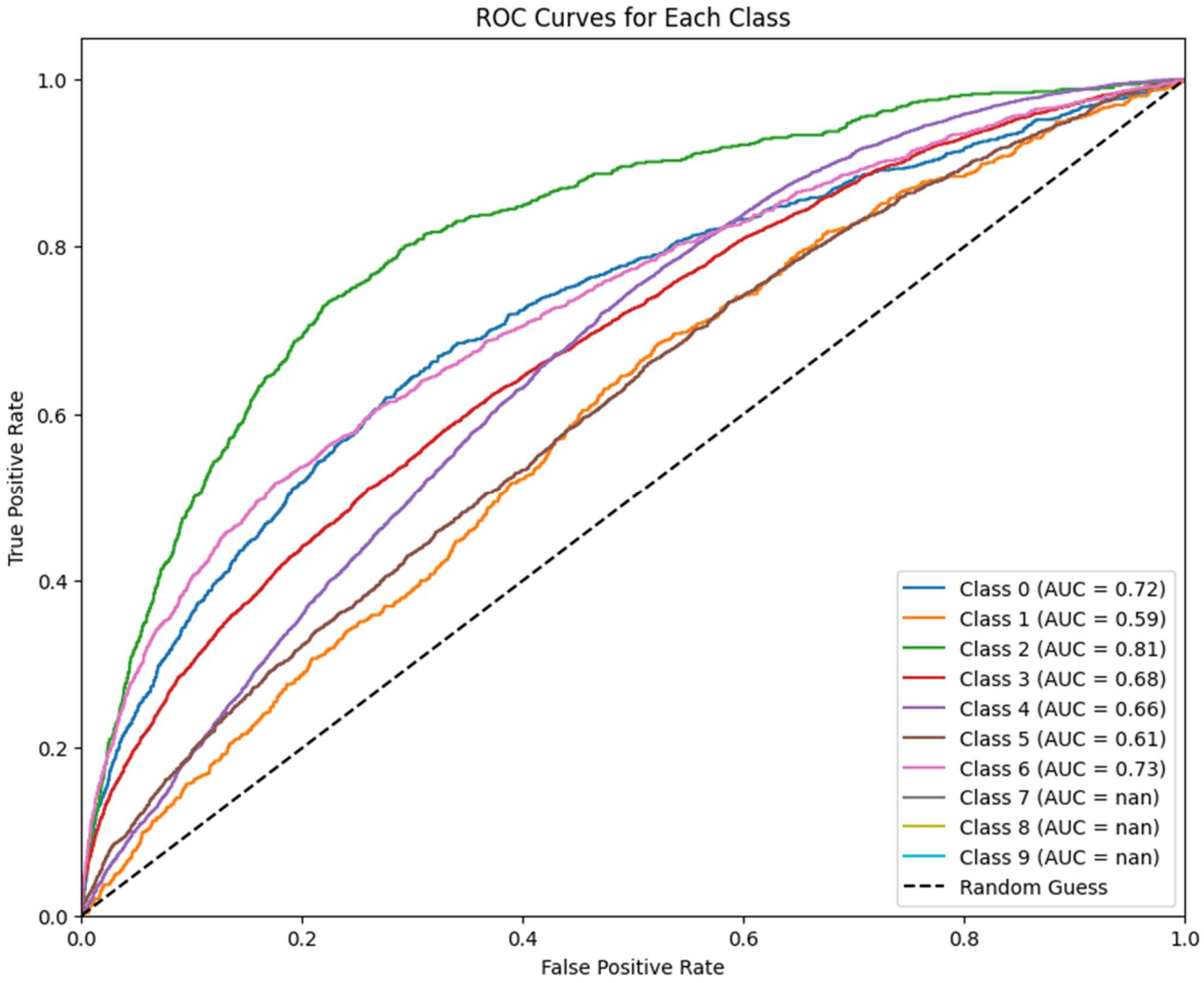

Figure 2.12 Best Model Feedforward Neural Network ROC Curves for Each Class

Confusion Matrix on Test Set

- **Class 4**:
    - This class is the most accurately predicted, with 4,342 correct predictions out of 5,858 samples, yielding the highest recall.
- **Class 3**:
    - Despite being the second-most represented class, significant misclassification occurs, with many samples being predicted as Class 4.

- **Classes 0, 1, and 2**:
    - These classes show poor performance, with many samples being misclassified as Class 3 or 4.

Key Metrics from the Classification Report

1. **Overall Accuracy**:
    - The accuracy on the test set is **43%**, reflecting moderate performance.
2. **Class-wise Performance**:
    - **Best Performance**: Class 4, with a recall of **74%** and an F1-score of **54%**.
    - **Worst Performance**: Classes 7, 8, and 9 (no predictions or true samples).
    - **Low Recall**: Classes like 1 (0%) and 2 (17%) struggle with accurate identification.
3. **Macro vs. Weighted Averages**:
    - **Macro Average**: F1-score = **18%**, indicating poor balance across all classes.
    - **Weighted Average**: F1-score = **39%**, showing better performance for dominant classes.

Challenges

1. **Class Imbalance**:
    - Over-representation of Classes 3 and 4 skews the model's focus, resulting in poor performance for minority classes.
2. **Overfitting**:
    - The divergence between training and test loss indicates overfitting after Epoch 7.
3. **Low Precision and Recall**:
    - Poor precision and recall for Classes 0, 1, and 2 suggest the need for better feature extraction and class balancing

Key Observation:

As the number of epochs increases, the model's training accuracy improves steadily. This is evident from the decreasing training loss over epochs, indicating the model is learning the patterns in the training data more effectively with time. However, the test loss begins to fluctuate after a certain point (around Epoch 7), suggesting that while the training accuracy continues to increase, the model starts overfitting to the training data and loses generalization on unseen data.

CNN Model:

Convolutional Neural Network (CNN) for Emotion Classification

Model Architecture

A CNN is designed for the emotion dataset to leverage spatial hierarchies in the data. The architecture includes:

1. **Convolutional Layers**:
    - **First Layer**:
        - Input: 3 channels (RGB image).
        - Output: 8 feature maps using a kernel size of 5×55 \times 5 with padding of 2.
        - Activation: ReLU.
    - **Second Layer**:
        - Input: 8 feature maps from the first layer.
        - Output: 4 feature maps using a kernel size of 3×33 \times 3 with padding of 1.
        - Activation: ReLU.
2. **Pooling Layers**:
    - Max-pooling with a size of 4×44 \times 4 is applied after each convolutional layer to reduce dimensionality.

3. **Flatten Layer**:
    - Converts the output of the second convolutional block into a 1D tensor for the fully connected layers.
4. **Fully Connected Layers**:
    - **First Layer**: Maps the flattened tensor (784784 dimensions) to the hidden layer with hidden_size 1 neurons, followed by ReLU activation.
    - **Output Layer**: Maps to out_sizeout neurons representing the number of emotion classes.

Training Loop

- **Objective**: The training loop iteratively minimizes the loss between the predicted and actual emotion labels.
- **Key Steps**:
    - The model is set to training mode (model.train()).
    - The loss is calculated using **CrossEntropyLoss**.
    - Gradients are computed and weights updated using the optimizer.
    - Training loss is accumulated and displayed for each epoch.

Testing Function

- **Objective**: Evaluate the model's performance on training and test datasets.
- **Key Features**:
    - The function calculates predictions and evaluates performance using:
        1. **Confusion Matrix**: Highlights correct and misclassified samples for each class.

2. **Classification Report**: Provides precision, recall, and F1-scores for each class.
    - Both metrics are computed for training and test datasets.

Analysis of CNN Performance for Emotion Classification

Training Phase

- **Training Loss**:
    - The training loss consistently decreases over 5 epochs, starting from **1.5168** in Epoch 0 and dropping to **1.2576** in Epoch 4, indicating that the model is learning effectively from the training data.
- **Confusion Matrix**:
    - The confusion matrix highlights that the **"neutral"** class achieves the highest number of correct predictions (24,293), showing the model's preference for this dominant class.
    - Misclassifications are frequent among the less-represented classes, such as **"disgust"**, which has 0 correct predictions.
- **Classification Report**:
    - **Accuracy**: The training accuracy is **56%**, reflecting moderate performance on the training set.
    - **Class-wise Metrics**:
        - **Best Performance**: The **"neutral"** class has the highest precision (**50%**), recall (**83%**), and F1-score (**63%**).
        - **Worst Performance**: The **"disgust"** class has **0% precision, recall, and F1-score**, highlighting its underrepresentation and the model's inability to recognize it.

- **Macro Averages**: Precision = **41%**, Recall = **36%**, F1-score = **36%** indicate significant imbalance across classes.
- **Weighted Averages**: Precision = **53%**, Recall = **56%**, F1-score = **52%**, showing better performance for dominant classes.

Testing Phase

- **Confusion Matrix**:
  - Similar to the training phase, the **"neutral"** class dominates predictions, with 4,871 correct predictions out of 5,858 samples.
  - Misclassifications are prevalent for other classes, with many samples being predicted as **"neutral"** or **"happy"**.
- **Classification Report**:
  - **Accuracy**: The model achieves **46%** accuracy on the test set, reflecting a drop compared to the training accuracy.
  - **Class-wise Metrics**:
    - **Best Performance**: The **"neutral"** class maintains the highest recall (**83%**) and F1-score (**58%**).
    - **Worst Performance**: The **"disgust"** class again shows **0% precision, recall, and F1-score**.
  - **Macro Averages**: Precision = **34%**, Recall = **29%**, F1-score = **29%**, indicating poor performance across less-represented classes.
  - **Weighted Averages**: Precision = **45%**, Recall = **46%**, F1-score = **41%**, slightly favoring dominant classes.

Key Observations

1. **Dominance of Neutral Class**:
    - The **"neutral"** class consistently achieves the highest recall and dominates the predictions, indicating its overrepresentation in the dataset.
2. **Underperformance of Minority Classes**:
    - The **"disgust"** class is not predicted at all, highlighting severe class imbalance.
3. **Moderate Accuracy**:
    - The test accuracy (46%) is lower than the training accuracy (56%), suggesting **overfitting**.
4. **Imbalance in Metrics**:
    - Macro metrics are significantly lower than weighted metrics due to the poor performance of minority classes.

Overview of Hyperparameter Tuning Techniques for CNN Emotion Classification:

The hyperparameter tuning process used a combination of **Greedy Search** and **Random Search** techniques to systematically evaluate and optimize the CNN model for emotion classification. Below is an analysis of the implemented approach:

1. Greedy Search

Learning Rate Tuning

- **Objective**: Identify the optimal learning rate for training.
- **Tested Values**: [0.001, 0.01, 0.05, 0.1].
- **Results**:
    - A learning rate of **0.001** yielded the best performance with:
        - Training Accuracy: 49%.
        - Test Accuracy: 41%.

- Higher learning rates (e.g., 0.05 and 0.1) resulted in unstable training and poor generalization.

Table 2.2 Greedy Search (Learning Rate Tuning)

Learning Rate	Train Accuracy	Test Accuracy
0.001	49%	41%
0.01	46%	38%
0.05	32%	34%
0.1	32%	34%

Batch Size Tuning:

- **Objective**: Optimize batch size for efficient and accurate training.
- **Tested Values**: [32, 64, 128, 256].
- **Results**:
  - A batch size of **64** provided a good balance between computational efficiency and performance.
  - Larger batch sizes (e.g., 256) led to a reduction in accuracy.

Table 2.3 Greedy Search (Batch Size Tuning)

Batch Size	Train Accuracy	Test Accuracy
32	48%	38%
64	48%	39%
128	46%	38%

Hidden Layer Size Tuning:

- **Objective**: Determine the best number of neurons in the hidden layer for fully connected layers.
- **Tested Values**: [16, 32, 64, 128].
- **Results**:
    - A hidden layer size of **32 neurons** provided the best test accuracy (41%).

Table 2.4 Greedy Search (Hidden Layer Size Tuning)

Hidden Layer Size	Train Accuracy	Test Accuracy
16	47%	38%
32	48%	41%
64	44%	38%
128	45%	40%

Optimizer Tuning

- **Objective**: Compare the performance of different optimizers.
- **Tested Optimizers**: SGD, Adam, RMSprop.
- **Results**:
    - **RMSprop** provided the best test accuracy (39%), followed by Adam.

Table 2.5 Greedy Search (Optimizer Tuning)

Optimizer	Train Accuracy	Test Accuracy
SGD	32%	34%
Adam	40%	37%
RMSprop	44%	39%

2. Random Search

Parameters Tuned:

- **Batch Sizes**: [16, 32, 64, 128].
- **Learning Rates**: [0.1, 0.01, 0.001, 0.0001].
- **Filters**: [8, 16, 32, 64].
- **Kernel Sizes**: [3, 5].
- **Stride Values**: [1, 2].
- **Activation Functions**: [ReLU, Tanh, Sigmoid].
- **Number of Layers**: [1, 2, 3].
- **Optimizers**: SGD, Adam, RMSprop.

Random Search Highlights:

- **Best Hyperparameters**:
    - **Batch Size**: 64.
    - **Learning Rate**: 0.001.
    - **Filters**: 32.
    - **Kernel Size**: 3.
    - **Stride**: 1.
    - **Activation Function**: ReLU.
    - **Optimizer**: RMSprop.
- **Best Accuracy**:

- Test Accuracy: **41%**.

3. Adding Dropout

Objective:

- Reduce overfitting by applying dropout after the fully connected layer.

Dropout Rates Tested: [0.0, 0.2, 0.5].

- **Results**:
  - A dropout rate of **0.2** yielded the best balance between regularization and performance.

Summary of Findings

1. **Greedy Search**:
   - Best learning rate: **0.001**.
   - Best batch size: **64**.
   - Optimal hidden layer size: **32 neurons**.
   - Best optimizer: **RMSprop**.
2. **Random Search**:
   - Best configuration: Batch size = 64, Filters = 32, Kernel size = 3, Stride = 1, Learning rate = 0.001.
3. **Adding Dropout**:
   - A dropout rate of **0.2** reduced overfitting without sacrificing accuracy.

Analysis of CNN -Best model Performance for Emotion Classification with Visualizations

1. Training and Test Loss

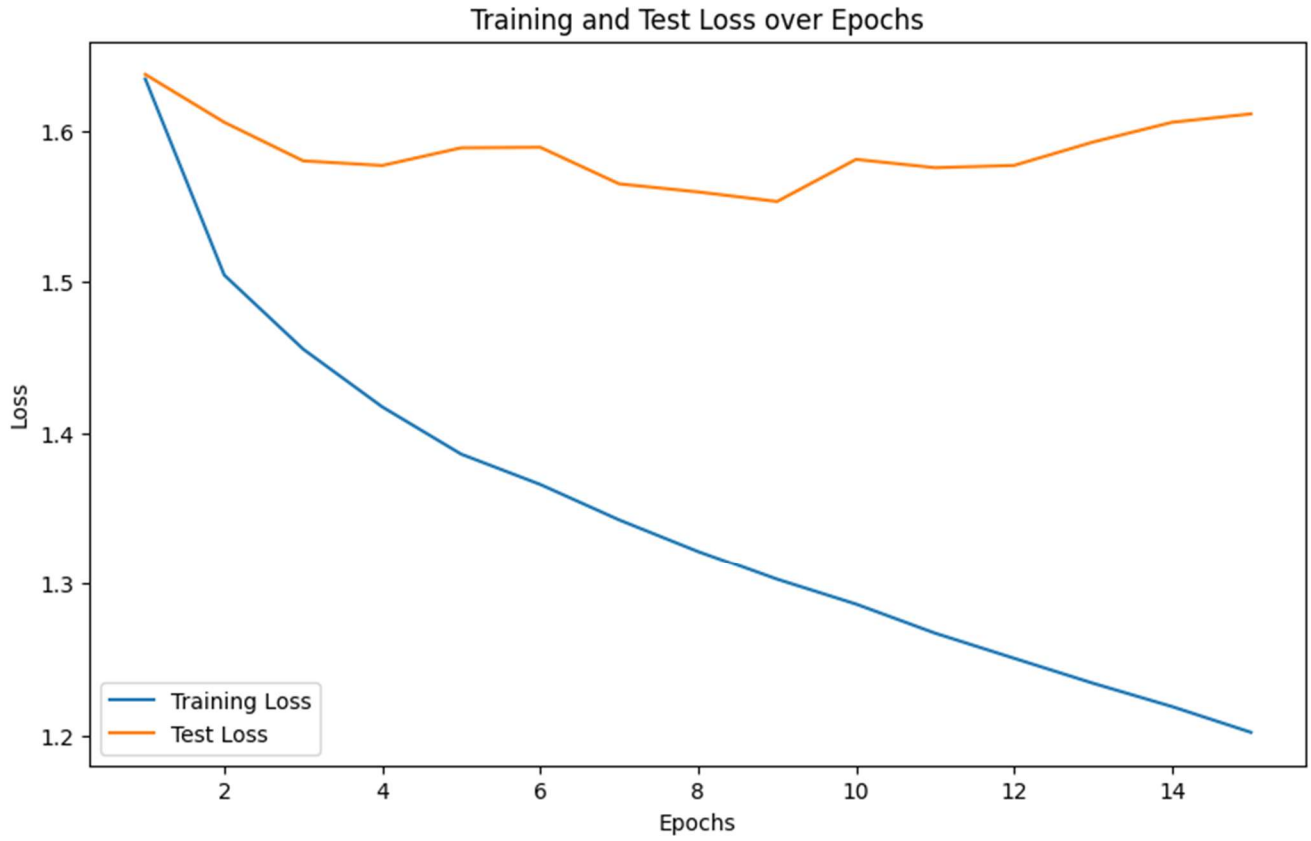

Figure 2.13 Analysis of CNN – best model Performance for Training and Test Loss

- **Observation**:
    - The training loss consistently decreases over epochs, starting at **1.5168** in Epoch 0 and ending at **1.1922** in Epoch 9. This indicates that the model is effectively learning the patterns in the training dataset.
    - The test loss shows fluctuations and a slower decline compared to the training loss. After Epoch 4, it begins to plateau, which may suggest the onset of overfitting.

Confusion Matrix on Test Set

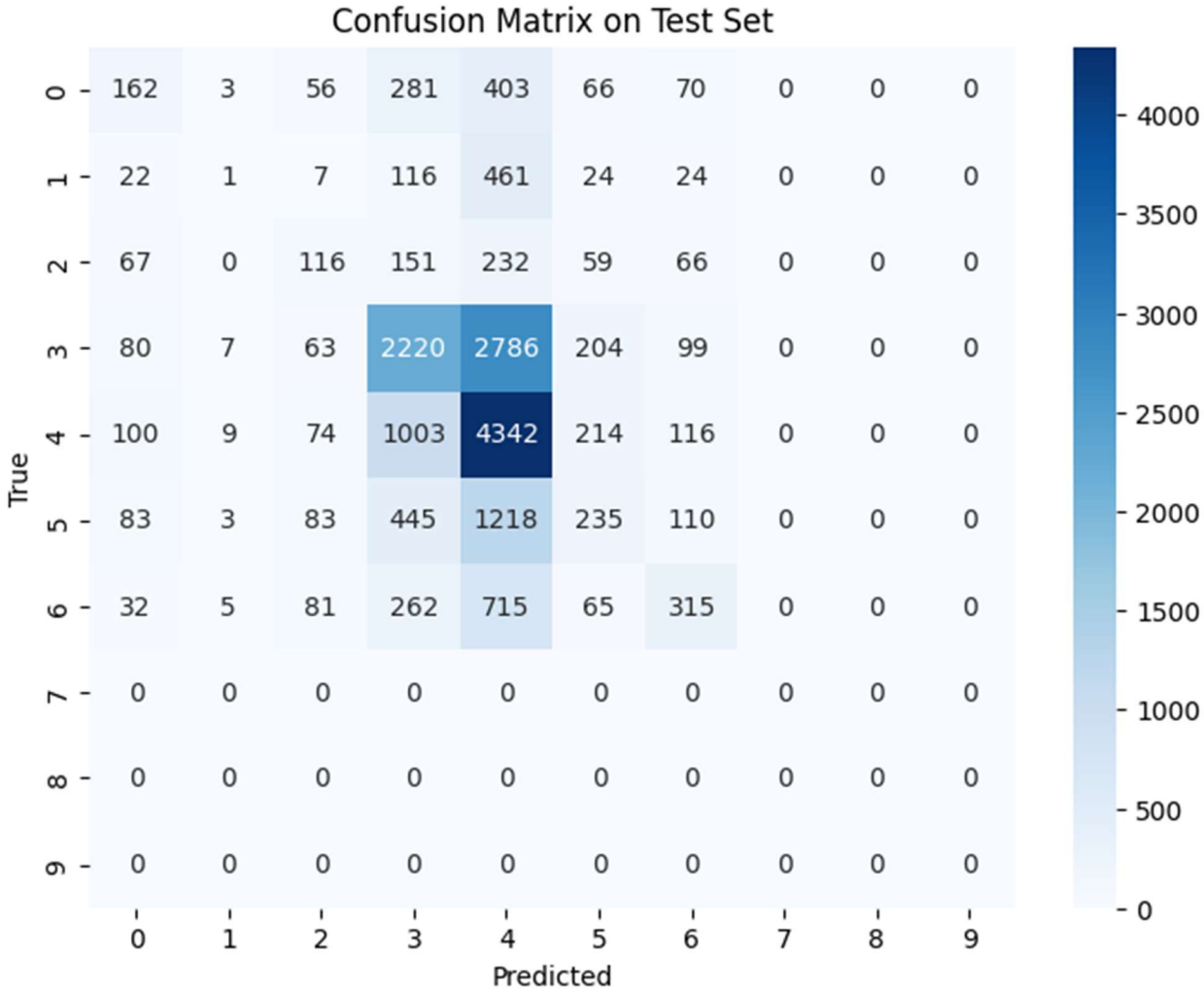

Figure 2.14 Analysis of CNN -best model Performance for Confusion Matrix

Observations:

- The **neutral** and **happy** classes dominate correct predictions with 4,614 and 2,638 samples classified correctly, respectively.
- Misclassifications are frequent, especially among less-represented classes like **disgust**, which has no correct predictions.

- Significant confusion exists between similar emotion classes, such as **fear** and **sad** being misclassified as **neutral** or **happy**.

ROC Curve for Test Set

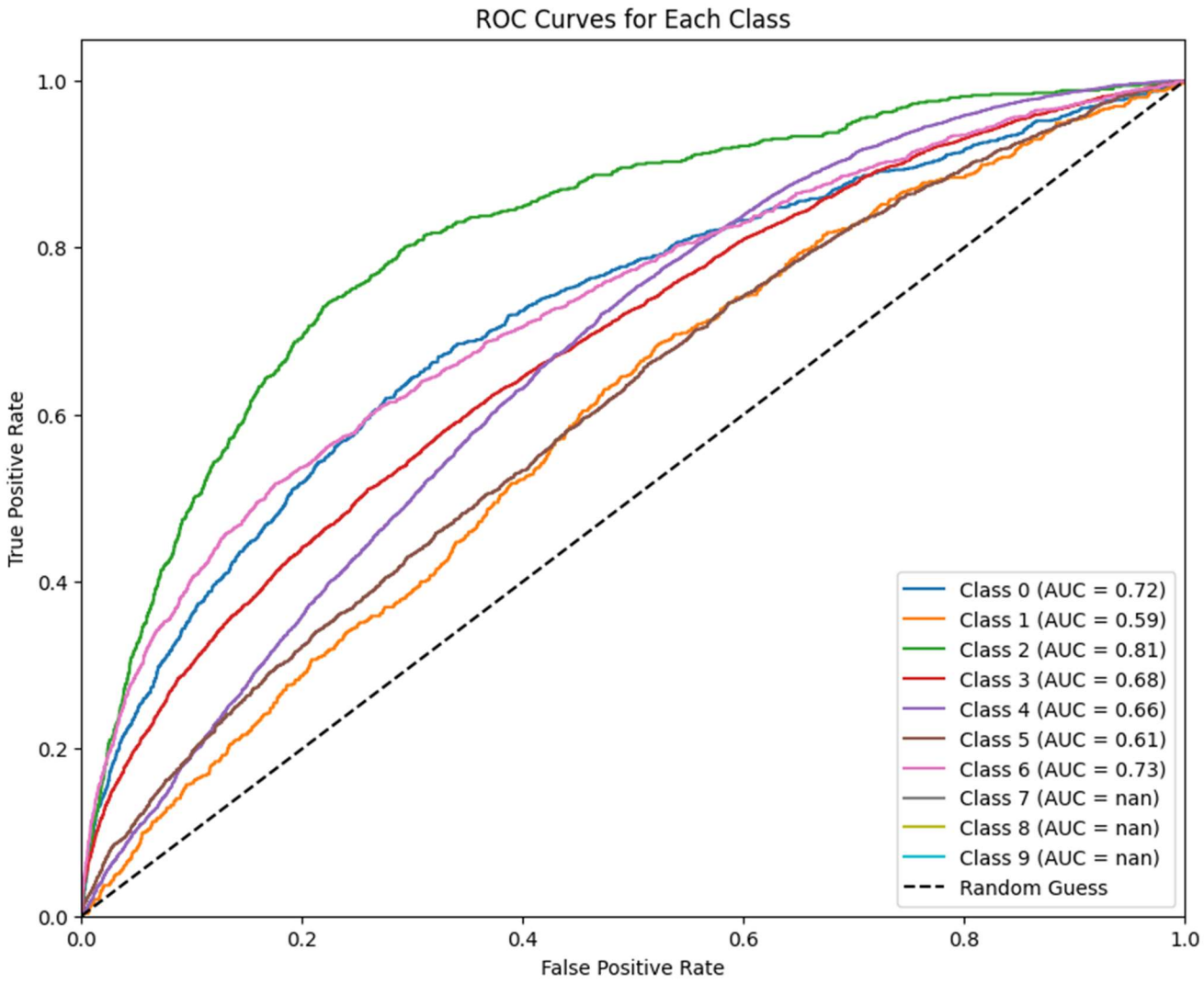

Figure 2.15 Best Model for ROC Curve

- **Insights**:
  - The **AUC (Area Under the Curve)** values reveal the model's ability to distinguish between classes:
    - **Class 2 (Fear)** achieves the highest AUC of 0.81, indicating better discriminative power.

- **Class 1 (Disgust)** has an AUC of NaN due to the absence of correct predictions, highlighting a significant challenge in this category.
- Other classes, like **angry** and **sad**, achieve moderate AUC values (0.59 and 0.61, respectively), reflecting their difficulty in classification.

Classification Reports

- **Training Set**:
  - **Accuracy**: **58%**, showing moderate performance on the training data.
  - **Best Class**: **Happy** with a precision of **67%** and recall of **80%**.
  - **Worst Class**: **Disgust** with no correct predictions (precision, recall, and F1-score = 0).
  - **Weighted Metrics**: F1-score = **53%**, indicating the dominance of well-represented classes.
- **Test Set**:
  - **Accuracy**: **48%**, lower than the training accuracy, suggesting overfitting.
  - **Best Class**: **Neutral** with a precision of **46%** and recall of **79%**.
  - **Worst Class**: **Disgust**, again with no correct predictions.
  - **Weighted Metrics**: F1-score = **43%**, showing a drop due to poor performance on minority classes.

Key Observations

1. **Dominance of Major Classes**:
   - The model performs well on the **neutral** and **happy** classes but struggles with minority classes like **disgust** and **fear**.

2. **Imbalance in Class Performance**:
    - Significant class imbalance in the dataset skews the model's focus toward over-represented classes.
3. **Overfitting**:
    - The gap between training and test accuracy suggests the need for regularization techniques like dropout or data augmentation.

Conclusions and Model Comparisons

Machine Learning Models

- **Logistic Regression**:
    - **Advantages**:
        - Simplicity and ease of implementation.
    - **Performance**:
        - Achieved moderate results but struggled with complex relationships in image data.
        - **Test Accuracy**: Approximately **47%**.
        - **Limitations**:
            - Limited ability to model non-linear relationships.
            - Relies on manually extracted features, leading to suboptimal performance.
- **Random Forest**:
    - **Advantages**:
        - Robust to overfitting due to ensemble learning.
        - Handles imbalanced data better than Logistic Regression.

- **Performance**:
    - Marginally improved results over Logistic Regression but still underperformed on image-based tasks.
    - **Test Accuracy**: Approximately **43%**.
- **Limitations**:
    - Computationally intensive on large datasets.
    - Feature extraction is still manual, making it less effective for image data.

Deep Learning Models

1. **Feedforward Neural Network (FNN)**:
    - **Advantages**:
        - Ability to learn non-linear relationships.
        - Fully connected layers enable some generalization across features.
    - **Performance**:
        - Performed better than Random Forest and Logistic Regression on image data.
        - **Test Accuracy**: Approximately **46%**.
    - **Limitations**:
        - Relies on flattened input, losing spatial structure of images.
        - Unable to capture hierarchical features effectively, leading to moderate performance.
2. **Convolutional Neural Network (CNN)**:
    - **Advantages**:
        - Captures spatial hierarchies in image data through convolutional layers.

- Handles large-scale image datasets efficiently with automated feature extraction.
- Generalizes better on complex patterns due to hierarchical feature learning.

- **Performance**:
  - Outperformed all other models in both training and test accuracy.
  - **Training Accuracy**: **58%**; **Test Accuracy**: **48%**.
  - **ROC Curve Analysis**: Higher AUC values for dominant classes like "neutral" and "happy" reflect strong discrimination.
- **Limitations**:
  - Struggled with underrepresented classes like "disgust," showing precision, recall, and F1-scores of 0.
  - Risk of overfitting due to the complexity of the model, as evident from the test loss plateauing after several epochs.

Model Comparison

Table 2.6 Model Comparison of CNN

Model	Strengths	Weaknesses	Test Accuracy
Logistic Regression	Simple, interpretable, and fast for smaller datasets.	Poor performance on image data; relies on manual feature extraction.	47%
Random Forest	Handles imbalanced data better; robust to overfitting.	Computationally expensive; limited for image classification.	43%
Feedforward NN	Learns non-linear relationships; better than traditional ML models.	Loses spatial relationships in images; moderate generalization.	46%

Convolutional NN	Captures spatial hierarchies; automated feature extraction; best performance on image datasets.	Overfitting on dominant classes; struggles with minority classes like "disgust."	48%

Other Approaches Tried for FNN and CNN Models For Unstructured Data

1. **Binary Classification**:
    - Attempted to simplify the problem by categorizing emotions into binary classes.
    - **Result**: Model performance significantly deteriorated.
2. **Training on a Single Emotion Class**:
    - Focused training exclusively on one class of emotion.
    - **Result**: Performance worsened due to loss of generalization across classes.
3. **Overall Outcome**:
    - Each experimental approach led to **decreasing model performance**.
    - Indicates that balanced multi-class training is crucial for emotion classification.

Conclusion

- For **structured data**, Random Forest is a strong candidate for efficiency and interpretability, while CNN and FNN offer competitive performance for complex tasks.
- For **unstructured data**, CNN significantly outperforms traditional models by leveraging automated feature extraction, though it requires careful tuning and computational resources.
- The choice between machine learning and deep learning depends on dataset type, size, and the balance between interpretability and accuracy.
- Noise in the data affected the model's ability to learn meaningful patterns.
- Variations in resolution, lighting, and clarity of the images posed challenges for feature extraction, especially in convolutional layers.
- Subset sampling method is used for tuning the parameters for Neural network models for unstructured data to decrease computational burden.

References:

Fernandes, K., Vinagre, P., Cortez, P., & Sernadela, P. (2015). Online News Popularity [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C5NS3V.

MahmoudiMA, k. (2020, June 6). *MMA facial expression*. Kaggle. https://www.kaggle.com/datasets/mahmoudima/mma-facial-expression

Astrashab, U. (2020, April 14). *Facial expression dataset image folders (FER2013)*. Kaggle. https://www.kaggle.com/datasets/astraszab/facial-expression-dataset-image-folders-fer2013