

AI-Driven Employee Feedback Analysis System:

Fostering Workplace Harmony

Group 12

M Sai Surya AP23110010034

RNV Jagadeesh Chandra AP23110010020

D Karunya AP23110010898

M Blessy AP23110011314

N Meghana AP23110010769

December 1, 2025

1 Project Description	4
2 Project Scenarios	4
2.1 Scenario 1: Real-Time Employee Feedback Analysis	4
2.2 Scenario 2: Bulk Feedback Processing and Trend Analysis	4
2.3 Scenario 3: Proactive Employee Retention and Engagement	4
3 Technical Diagram	5
4 Prerequisites	5
4.1 Software Requirements	5
4.1.1 Anaconda Navigator and Visual Studio Code	5
4.1.2 Python Packages	6
4.2 Prior Knowledge	6
4.2.1 ML Concepts	6
4.2.2 NLP Concepts	6
4.2.3 Web Development	7
5 Project Flow	7
6 Project Activities	8
6.1 Data Collection & Preparation	8
6.2 Exploratory Data Analysis	8
6.3 Model Building	8
6.4 Performance Testing & Model Selection	8
6.5 Advanced Features Implementation	8
6.6 Model Deployment	9
7 Project Structure	9
7.1 Project Structure Explanation	9
8 Milestone 1: Data Collection & Preparation	10
8.1 Activity 1.1: Collect the Dataset.....	10
8.1.1 Dataset Source.....	10
8.2 Activity 1.2: Importing the Libraries.....	11
8.3 Activity 1.3: Read the Dataset.....	11
8.4 Activity 1.4: Exploratory Data Analysis (EDA).....	12
8.4.1 Outlier Analysis.....	12
8.4.2 Histogram Analysis.....	13
8.4.3 Univariate Analysis.....	14
8.4.4 Bivariate Analysis.....	15
8.4.5 Correlation Analysis	16
8.4.6 Pair Plot Analysis.....	17
8.5 Activity 1.5: Data Preparation	19
8.6 Activity 1.6: Convert Ratings to Sentiment Labels.....	20
8.7 Activity 1.7: Check Class Balance.....	20

9 Milestone 2: Model Training & Evaluation	20
9.1 Activity 2.1: TF-IDF Vectorization	20
9.2 Activity 2.2: Train Machine Learning Models	21
9.3 Activity 2.3: Model Evaluation	21
9.4 Activity 2.4: Save Model and Vectorizer	21
10 Milestone 3: Advanced Features Implementation	22
10.1 Activity 3.1: Theme Extraction	22
10.2 Activity 3.2: Emotion Detection	22
10.3 Activity 3.3: Mixed Feedback Logic	22
10.4 Activity 3.4: AI Summary Generation	23
11 Milestone 4: Web Application Development	23
11.1 Activity 4.1: Flask Backend API	23
11.2 Activity 4.2: Additional API Endpoints	24
12 Frontend Implementation	25
12.1 Key Features of the Web Application	25
12.1.1 User Interface Components	25
12.1.2 Analysis Display Components	25
12.1.3 Advanced Features	25
13 Application Screenshots and Workflow	26
13.1 Home Page Interface	26
13.2 Employee Feedback Input Form	27
13.3 Analysis Results Display	27
13.4 Export Functionality	28
14 Model Performance Metrics	29
15 Future Implementations	29
15.1 Advanced NLP Models	29
15.2 Real-time Integration	29
15.3 Advanced Analytics	29
15.4 Enhanced User Experience	30
16 Conclusion	30
17 Technical Specifications	30
17.1 Model Architecture	30
17.2 System Requirements	31
17.3 API Endpoints	31
18 References and Resources	31

1 Project Description

The AI-Driven Employee Feedback Analysis System is an advanced Natural Language Processing (NLP) and Machine Learning project designed to analyze employee feedback and provide actionable insights for Human Resources (HR) departments. The system utilizes supervised learning algorithms, including Naive Bayes and Logistic Regression, combined with TF-IDF vectorization to classify employee feedback into sentiment categories (Positive, Neutral, Negative). By examining text patterns, emotional indicators, and workplace themes, the model assists HR professionals in understanding employee satisfaction, identifying areas of concern, and making data-driven decisions to improve workplace culture and employee retention.

The system goes beyond simple sentiment classification by extracting key themes (such as salary, workload, management, culture), detecting emotions (joy, stress, anger, frustration, motivation), generating AI-powered summaries, and providing department-wise sentiment analysis. The final product is a full-stack web application built with Flask that seamlessly integrates the trained model into an intuitive, modern HR dashboard interface.

2 Project Scenarios

2.1 Scenario 1: Real-Time Employee Feedback Analysis

An HR manager receives employee feedback through an anonymous survey or feedback portal. The AI-Driven Employee Feedback Analysis System processes the text input in real-time, analyzing sentiment, extracting key themes, detecting emotional indicators, and generating actionable insights. Within seconds, the system provides a comprehensive analysis including sentiment classification, confidence scores, identified themes, emotion breakdown, and AI-generated recommendations, enabling HR professionals to quickly understand employee concerns and prioritize interventions.

2.2 Scenario 2: Bulk Feedback Processing and Trend Analysis

HR departments conduct quarterly employee satisfaction surveys, collecting hundreds or thousands of feedback responses. The system processes all feedback entries efficiently, identifying sentiment trends across different departments, job roles, and time periods. The department-wise sentiment analysis feature helps HR identify which departments require immediate attention, while theme extraction reveals common concerns such as workload pressure, compensation issues, or management problems. This enables HR to make strategic decisions about resource allocation, training programs, and policy changes.

2.3 Scenario 3: Proactive Employee Retention and Engagement

In organizations with high turnover rates, the system analyzes historical and real-time employee feedback to identify early warning signs of dissatisfaction. By detecting negative sentiment patterns, stress indicators, and frustration markers, HR can proactively reach out to at-risk employees, address concerns before they escalate, and implement targeted interventions. The AI summary feature provides quick insights that help HR managers prepare for one-on-one conversations and develop personalized retention strategies.

3 Technical Diagram

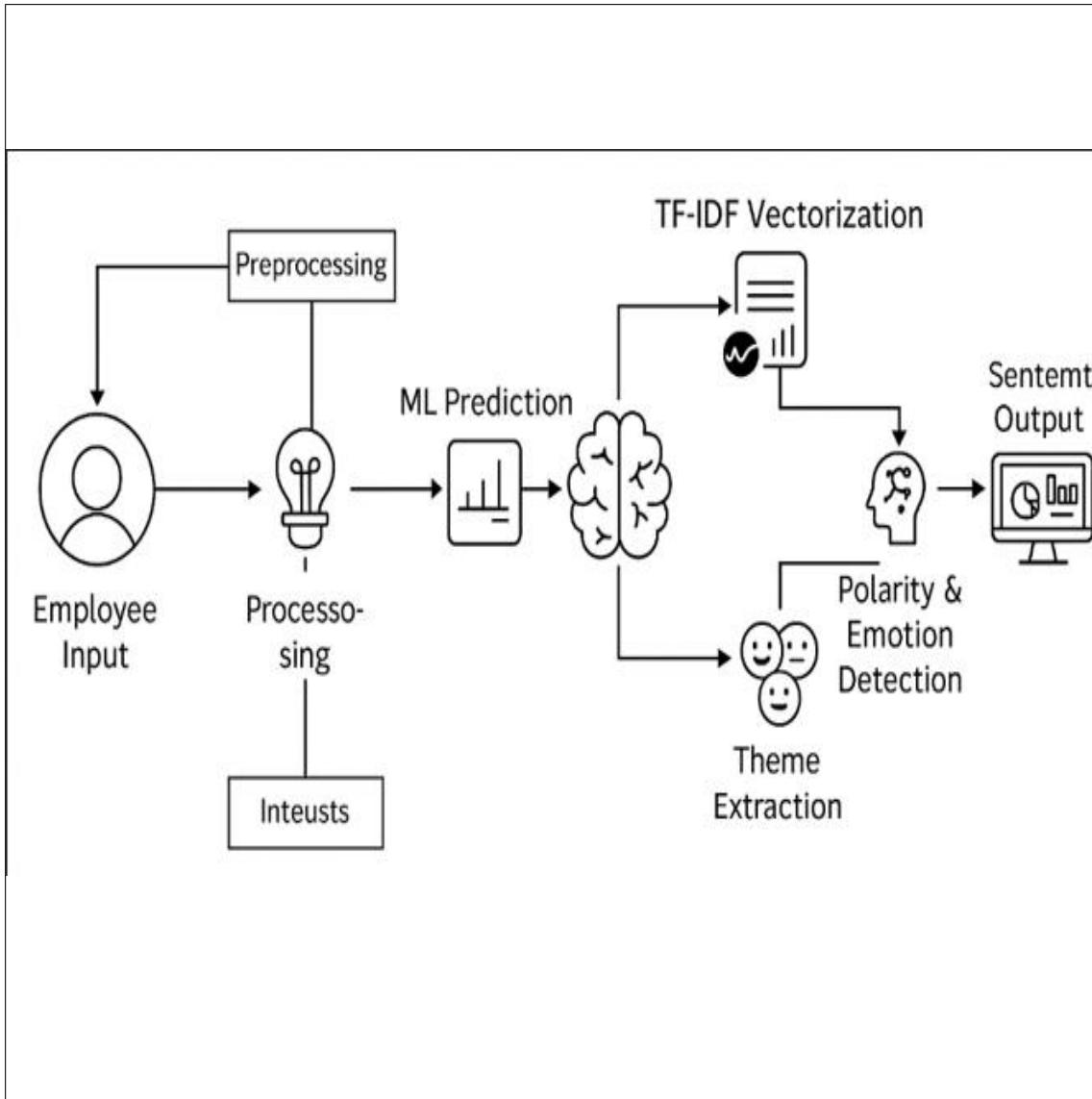


Figure 1: System Architecture and Workflow Diagram

4 Prerequisites

To complete this project, you must require the following software, concepts, and packages:

4.1 Software Requirements

4.1.1 Anaconda Navigator and Visual Studio Code

- Refer to the link below to download Anaconda Navigator
- Link: <https://youtu.be/1ra4zH2G4o0>

4.1.2 Python Packages

Open anaconda prompt as administrator and install the following packages:

```
pip install pandas
pip install numpy
pip install scikit-learn
pip install matplotlib
pip install seaborn
pip install nltk
pip install flask
pip install flask-cors
pip install joblib
pip install wordcloud
pip install imbalanced-learn
pip install emoji
pip install textblob
```

4.2 Prior Knowledge

You must have prior knowledge of the following topics to complete this project.

4.2.1 ML Concepts

- **Supervised Learning:** <https://www.javatpoint.com/supervised-machine-learning>
- **Text Preprocessing:** Understanding of NLP preprocessing techniques (tokenization, lemmatization, stopword removal)
- **TF-IDF Vectorization:** <https://www.geeksforgeeks.org/tf-idf-model-for-page-ranking/>
- **Naive Bayes Classifier:** [https://www.javatpoint.com/machine-learning-naive-bayes-cl](https://www.javatpoint.com/machine-learning-naive-bayes-classifier)
- **Logistic Regression:** [https://www.javatpoint.com/logistic-regression-in-machine-lea](https://www.javatpoint.com/logistic-regression-in-machine-learning)
- **Evaluation Metrics:** [https://www.analyticsvidhya.com/blog/2019/08/11-important-mod](https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-metrics/)
- **SMOTE:** For handling imbalanced datasets

4.2.2 NLP Concepts

- Natural Language Processing Basics: Tokenization, stemming, lemmatization
- Sentiment Analysis: Understanding of sentiment classification approaches
- Text Feature Extraction: TF-IDF, n-grams, feature selection

4.2.3 Web Development

- Flask Basics: <https://www.youtube.com/watch?v=lj4LCvBnt0>
- RESTful API Design: Understanding of HTTP methods and JSON responses
- HTML/CSS/JavaScript: Basic frontend development knowledge
- Chart.js: For data visualization

2 Project Flow

The project follows these sequential steps:

1. **Data Collection:** User provides employee feedback dataset (CSV format) containing employee reviews, ratings, and metadata.
2. **Data Preprocessing:** System preprocesses the input data using comprehensive text cleaning (lowercase conversion, emoji removal, URL removal, special character removal, tokenization, lemmatization, stopword removal).
3. **Dataset Preparation:** System converts overall ratings to sentiment labels (1-2 → negative, 3 → neutral, 4-5 → positive), checks class balance, and prepares the final dataset.
4. **Feature Engineering:** System applies TF-IDF vectorization with n-gram extraction (unigrams and bigrams) to convert text into numerical features suitable for machine learning.
5. **Model Training:** Integrated machine learning model (Naive Bayes or Logistic Regression) analyzes the vectorized features and learns patterns from labeled data.
6. **Model Evaluation:** System evaluates model performance using accuracy, precision, recall, F1-score, and confusion matrix metrics.
7. **Theme Extraction:** System extracts top HR-related keywords and themes from the feedback corpus using TF-IDF and CountVectorizer.
8. **Real-Time Inference:** When user enters new feedback through the web interface, the system preprocesses input, vectorizes it, and makes sentiment predictions with confidence scores.
9. **Advanced Analysis:** System performs additional analysis including theme extraction, emotion detection, polarity scoring, and AI summary generation.
10. **Results Display:** Prediction results (Positive/Neutral/Negative sentiment) are displayed with confidence metrics, themes, emotions, AI summary, and department-wise insights through an interactive HR dashboard.

3 Project Activities

3.1 Data Collection & Preparation

- Collect the employee feedback dataset
- Data preprocessing and cleaning
- Rating-to-sentiment conversion
- Class balance analysis

3.2 Exploratory Data Analysis

- Descriptive statistics
- Visual analysis (word clouds, distribution charts)
- Univariate and bivariate analysis
- Class distribution visualization

3.3 Model Building

- Training the model with multiple algorithms (Naive Bayes, Logistic Regression)
- TF-IDF vectorization with hyperparameter tuning
- Testing different classifiers
- Handling imbalanced datasets (SMOTE)

3.4 Performance Testing & Model Selection

- Testing model with multiple evaluation metrics (Accuracy, Precision, Recall, F1-Score)
- Comparing model accuracy across different algorithms
- Confusion matrix analysis
- Selecting the best performing model

3.5 Advanced Features Implementation

- Theme extraction from feedback
- Emotion detection
- AI summary generation
- Department-wise sentiment analysis
- Mixed feedback logic and override rules

3.6 Model Deployment

- Save the best model and vectorizer
- Integrate with Flask Web Framework
- Create RESTful API endpoints
- Build modern HR dashboard UI

4 Project Structure

The project folder structure is organized as follows:

Gen/

```

app.py                                # Flask backend application
preprocess_data.py                    # Data preprocessing script
prepare_dataset.py                   # Dataset preparation script
train_model.py                       # Model training script
extract_themes.py                   # Theme extraction script
run_pipeline.py                      # Pipeline orchestration script

employee_reviews.csv                  # Raw dataset
processed_data.csv                   # Preprocessed data (generated)
final_dataset.csv                     # Final dataset with sentiment labels

sentiment_model.pkl                  # Trained model file (generated)
vectorizer.pkl                        # TF-IDF vectorizer (generated)
extracted_keywords.txt               # Extracted keywords (generated)
wordcloud.png                         # Word cloud visualization

requirements.txt                      # Python dependencies

templates/
    index.html                         # Web UI template

```

4.1 Project Structure Explanation

- **app.py**: Flask application backend that handles API endpoints, model inference, and advanced analysis features (emotion detection, theme extraction, AI summaries).
- **preprocess_data.py**: Script for loading and cleaning the employee feedback dataset. Handles text preprocessing including lowercase conversion, emoji removal, URL removal, tokenization, lemmatization, and stopword removal.
- **prepare_dataset.py**: Script that converts overall ratings to sentiment labels, checks class balance, and prepares the final dataset for training.

- **train_model.py**: Model training script that implements TF-IDF vectorization, trains Naive Bayes or Logistic Regression classifiers, evaluates performance, and saves the trained model.
- **extract_themes.py**: Script for extracting top HR-related keywords and themes from employee feedback using TF-IDF and CountVectorizer, and generating word cloud visualizations.
- **templates/index.html**: Modern HR dashboard UI with sentiment badges, confidence meters, theme tags, emotion visualization, interactive charts, dark/light mode, and export functionality.
- **sentiment_model.pkl**: Trained machine learning model file (saved using joblib).
- **vectorizer.pkl**: TF-IDF vectorizer file (saved using joblib).

5. Milestone 1: Data Collection & Preparation

Data collection is fundamental to machine learning, providing the raw material for training algorithms and making predictions. This process involves gathering relevant information from various sources such as databases, surveys, and feedback portals. The quality, quantity, and diversity of collected data significantly impact the performance and accuracy of ML models.

5.1 Activity 1.1: Collect the Dataset

There are many popular open sources for collecting data. For example: kaggle.com, UCI repository, etc. In this project, we have used CSV data. This data can be downloaded from kaggle.com or other data repositories.

Dataset Source Examples:

- <https://www.kaggle.com/datasets/petersunga/google-amazon-facebook-employee-review>
- <https://www.kaggle.com/datasets/ambarish/indeed-job-reviews-dataset>

5.1.1 Dataset Source

The dataset used in this project contains employee feedback and reviews from various companies. The dataset includes the following features:

- **summary**: Employee review summary text
- **pros&cons**: Pros and cons of working at the company
- **overall-ratings**: Overall rating (1-5 stars)
- **job-title**: Job title or role of the employee
- **location**: Location of the employee/company
- **company**: Company name

The dataset file is named employee_reviews.csv and contains multiple employee records with their feedback, ratings, and metadata.

5.2 Activity 1.2: Importing the Libraries

Import the necessary libraries as shown below:

```

1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 import nltk
6 from nltk.corpus import stopwords
7 from nltk.tokenize import word_tokenize
8 from nltk.stem import WordNetLemmatizer
9 import re
10 import emoji
11 from sklearn.feature_extraction.text import TfidfVectorizer
12 from sklearn.naive_bayes import MultinomialNB
13 from sklearn.linear_model import LogisticRegression
14 from sklearn.metrics import accuracy_score, precision_score,
   recall_score, f1_score
15 import joblib
16 from flask import Flask, request, jsonify, render_template
17 from textblob import TextBlob

```

These libraries serve the following purposes:

- **NumPy**: For numerical computations
- **Pandas**: For data manipulation and analysis
- **Matplotlib**: For creating visualizations
- **Seaborn**: For statistical data visualization
- **NLTK**: For natural language processing (tokenization, stopwords, lemmatization)
- **scikit-learn**: For machine learning algorithms and evaluation metrics
- **Flask**: For web application framework
- **TextBlob**: For sentiment polarity analysis
- **joblib**: For model persistence

5.3 Activity 1.3: Read the Dataset

Our dataset is in CSV format. We read the dataset using pandas:

```

1 df = pd.read_csv('employee_reviews.csv', encoding='latin-1',
2                   low_memory=False)
3 df.head()

```

Sample Output:

summary	pros&cons	overall-ratings	job-title
Great company culture	Good benefits, work-life...	5	Software Engineer
Stressful environment	High pressure, long hours	2	Manager
Average workplace	Okay benefits, decent pay	3	Analyst

Table 1: Sample Dataset Preview

The head() function displays the first 5 rows of the dataset, giving us a quick overview of the data structure and values.

5.4 Activity 1.4: Exploratory Data Analysis (EDA)

Before preprocessing, we perform comprehensive Exploratory Data Analysis to understand the data distribution, identify outliers, and discover patterns. This analysis includes visual analysis, univariate analysis, bivariate analysis, correlation analysis, and pair plot analysis.

5.4.1 Outlier Analysis

Outlier detection helps identify data points that deviate significantly from the rest of the dataset. We use box plots to visualize outliers in numerical features such as feedback length, word count, and ratings.

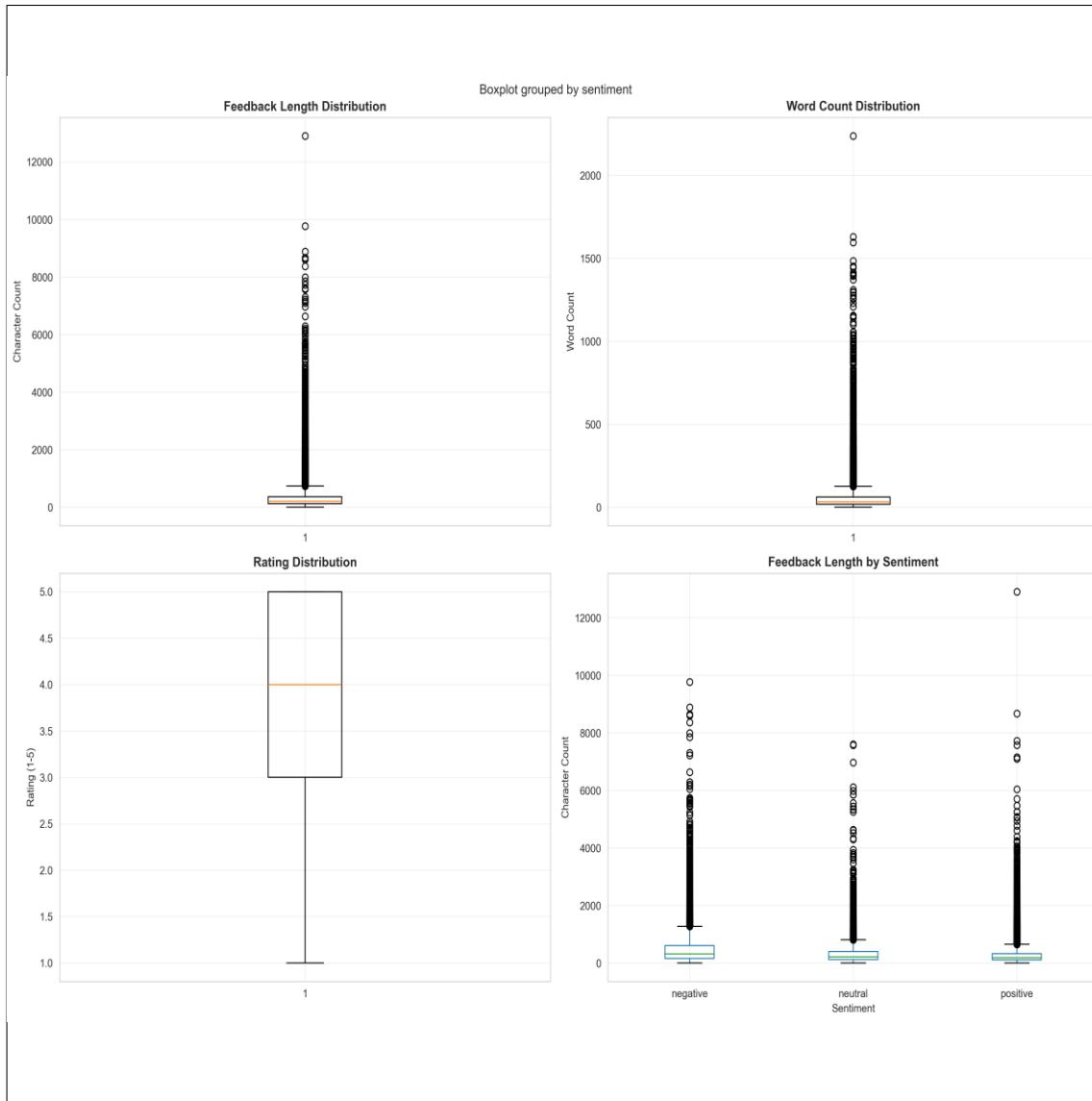


Figure 2: Outlier Analysis - Box Plots

The box plots reveal:

- Distribution of feedback length (character count)
- Distribution of word count per feedback
- Rating distribution (if available)
- Feedback length variation across different sentiment classes

5.4.2 Histogram Analysis

Histograms provide insights into the frequency distribution of numerical variables, helping us understand the data's underlying distribution patterns.

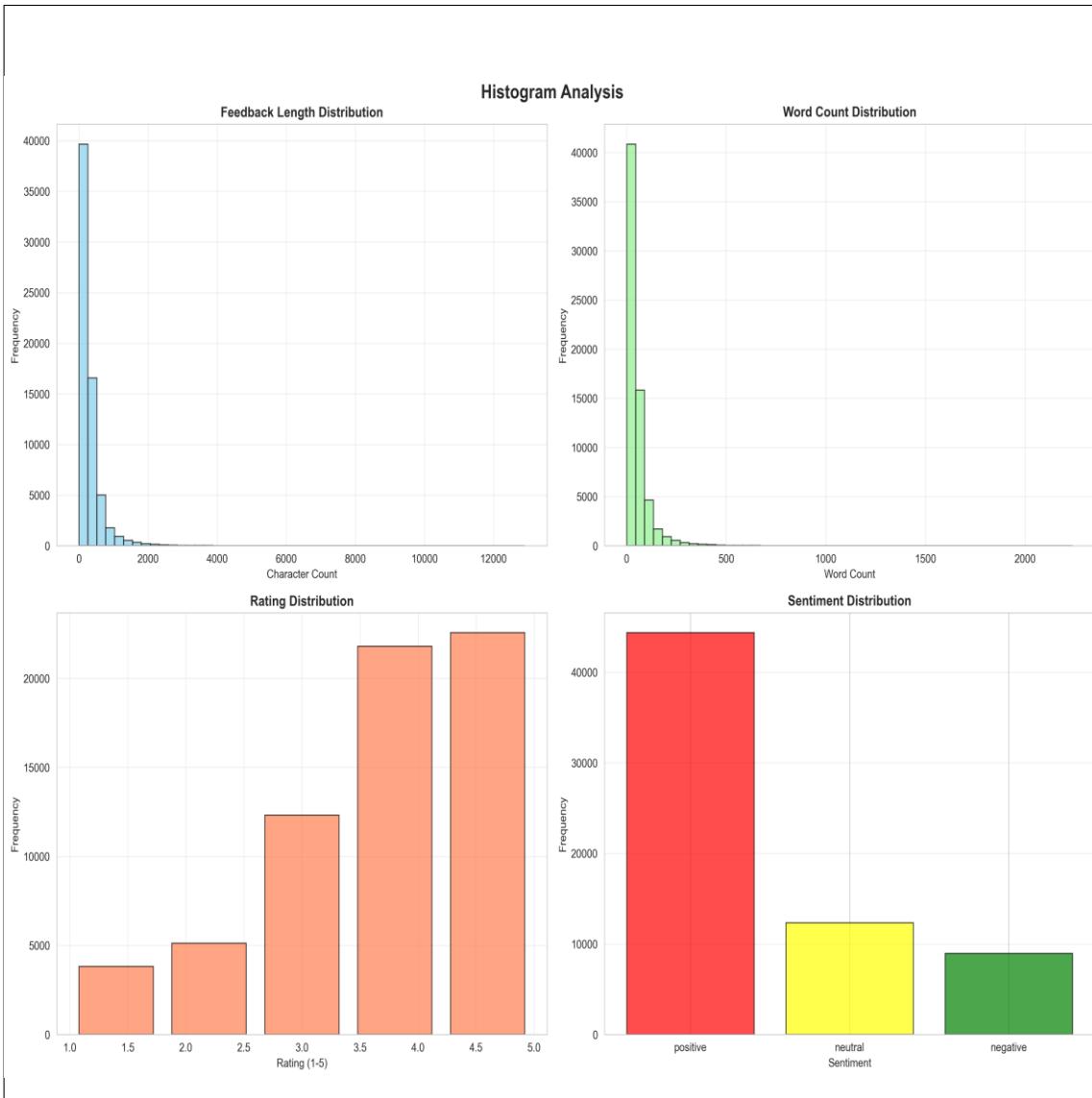


Figure 3: Visual Analysis - Histogram

The histograms show:

- Frequency distribution of feedback length
- Frequency distribution of word count
- Rating frequency distribution
- Sentiment class distribution

5.4.3 Univariate Analysis

Univariate analysis examines individual variables in isolation, combining histograms with Kernel Density Estimation (KDE) to understand the probability distribution of each feature.

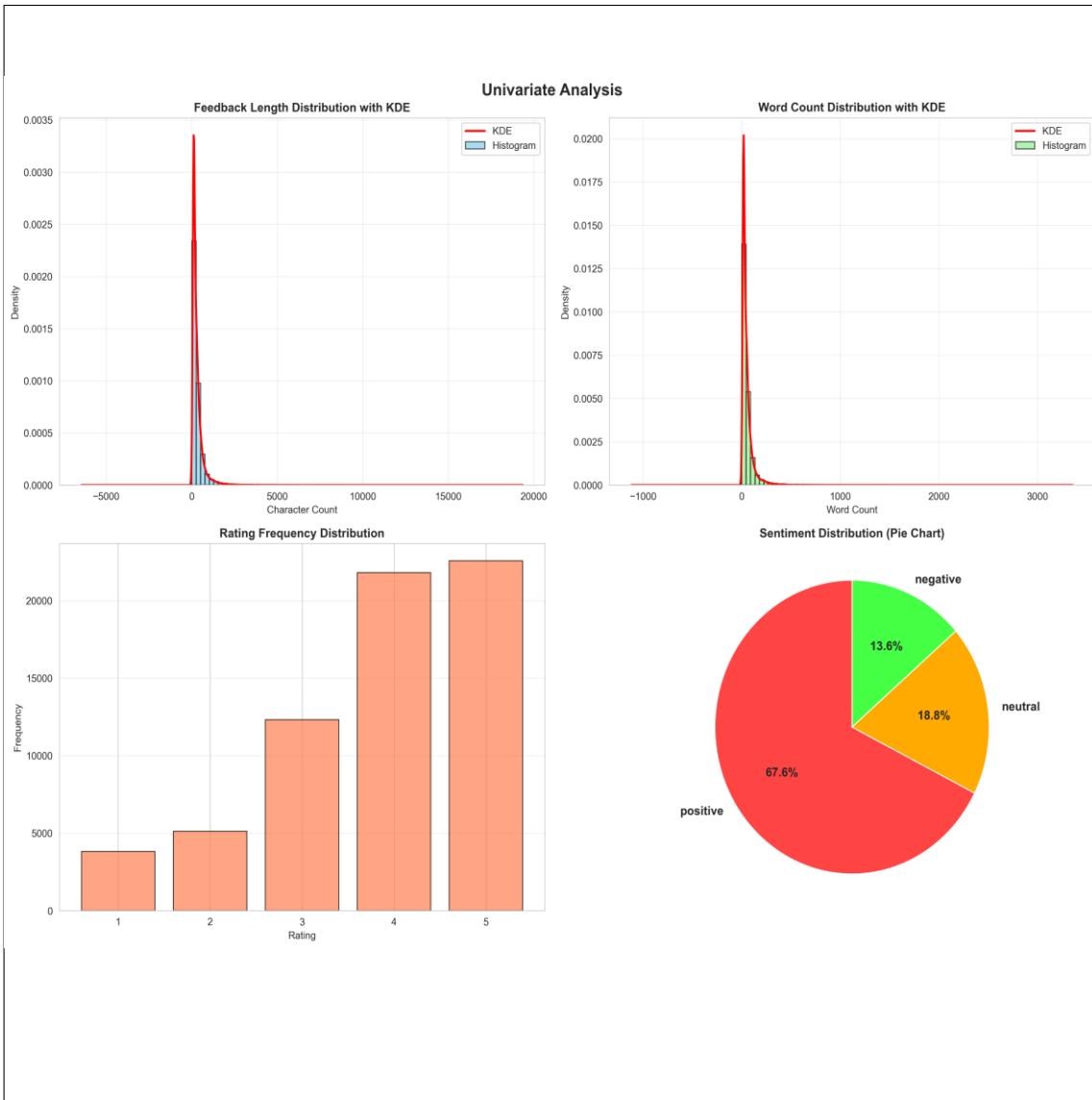


Figure 4: Univariate Analysis with KDE

Key observations:

- Feedback length distribution with density curve
- Word count distribution with density curve
- Rating frequency patterns
- Sentiment class proportions (pie chart)

5.4.4 Bivariate Analysis

Bivariate analysis explores relationships between two variables, helping identify correlations and patterns between different features.

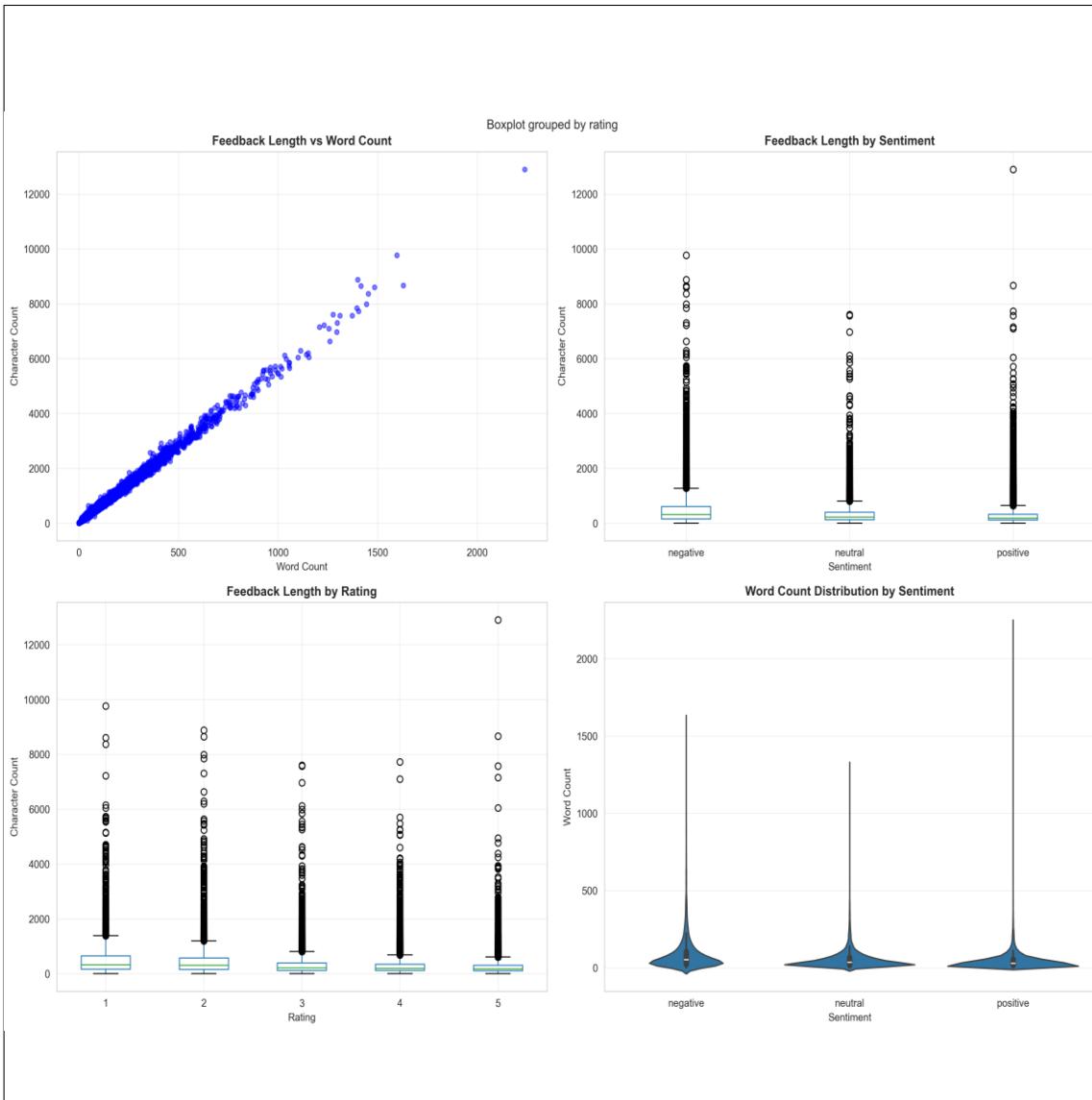


Figure 5: Bivariate Analysis

The bivariate plots reveal:

- Relationship between word count and feedback length (scatter plot)
- Feedback length variation across sentiment classes (box plot)
- Feedback length variation across ratings (box plot)
- Word count distribution by sentiment (violin plot)

5.4.5 Correlation Analysis

Correlation analysis measures the strength and direction of linear relationships between numerical variables using correlation coefficients.

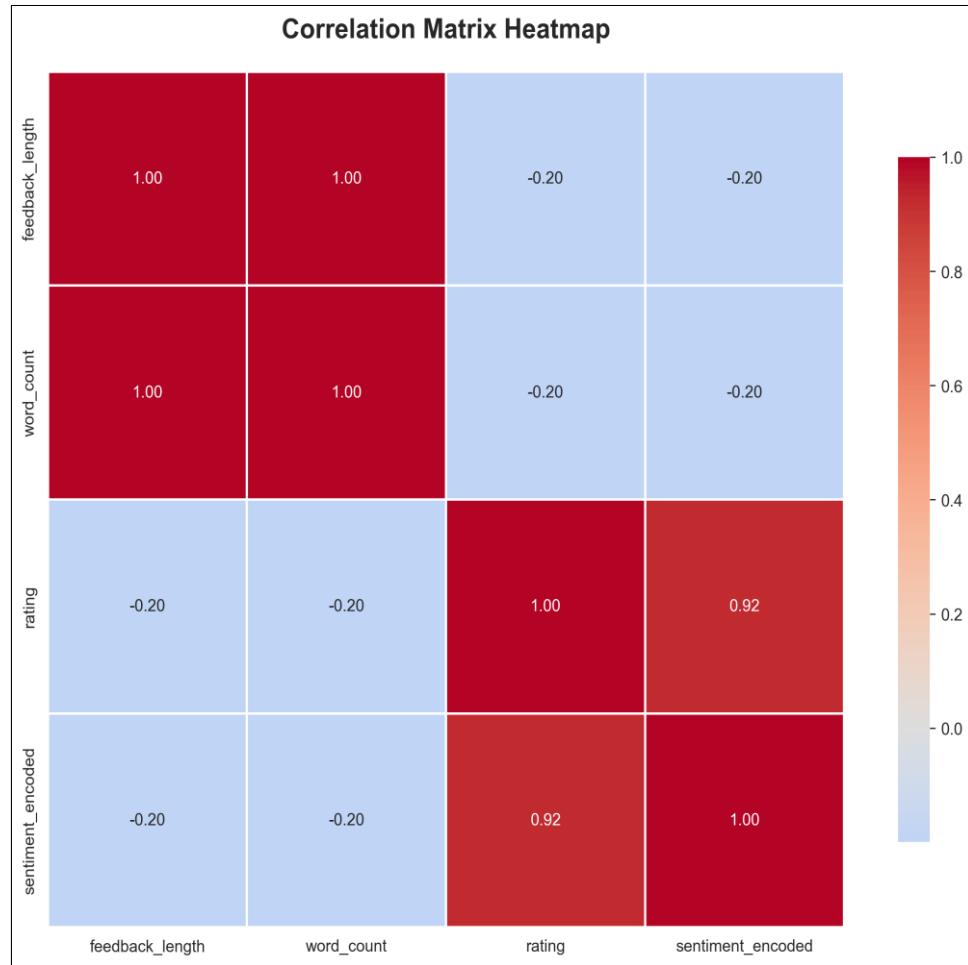


Figure 6: Correlation Matrix Heatmap

The correlation heatmap shows:

- Correlation coefficients between all numerical features
- Strong positive correlations (red) and negative correlations (blue)
- Features with weak or no correlation (white/yellow)

5.4.6 Pair Plot Analysis

Pair plots provide a comprehensive view of relationships between all pairs of numerical variables, with diagonal plots showing univariate distributions.

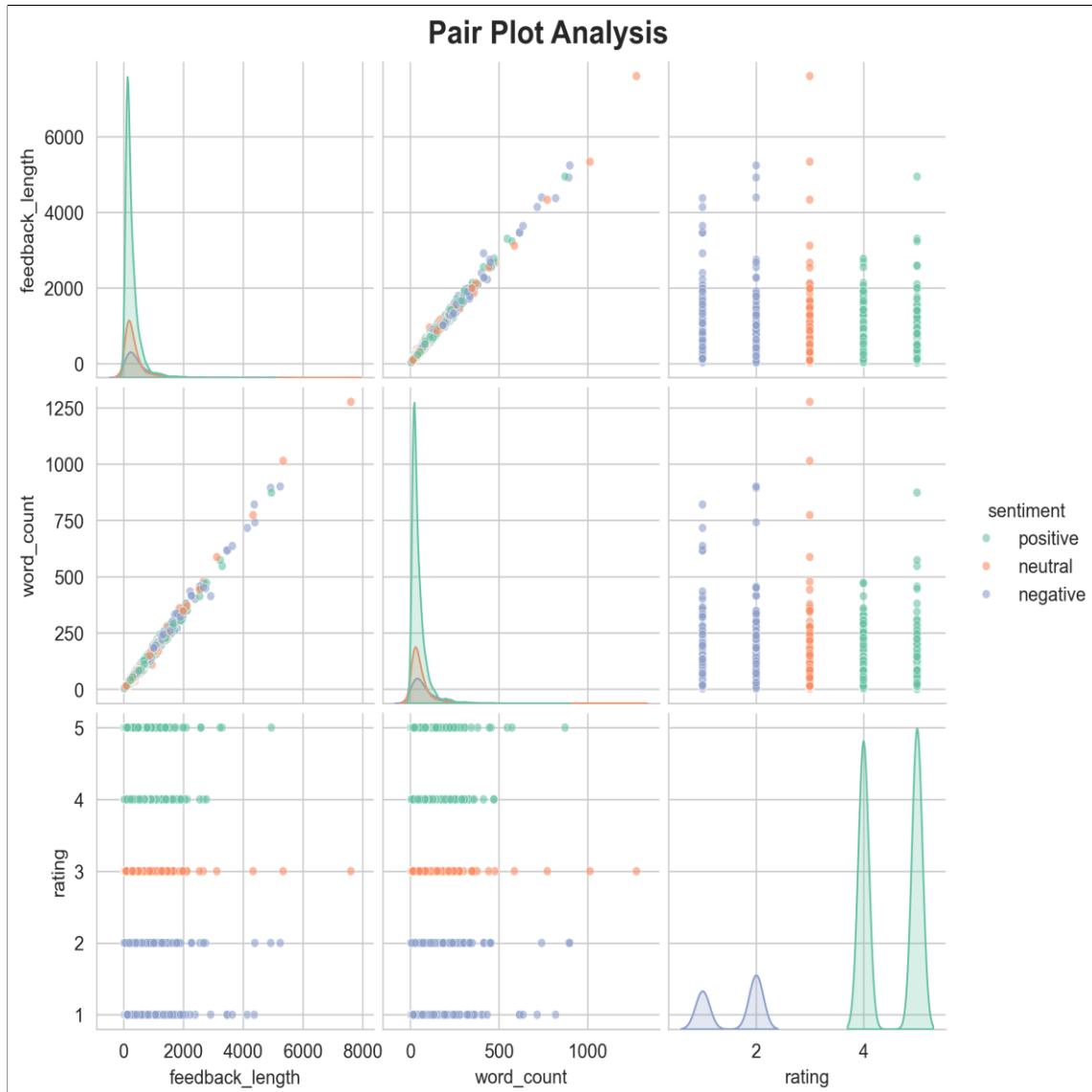


Figure 7: Pair Plot Analysis

The pair plot displays:

- Scatter plots for all variable pairs
- KDE plots on the diagonal showing univariate distributions
- Color coding by sentiment class (if available)
- Comprehensive overview of all variable relationships

Code for Generating EDA Visualizations:

```

1 # Run the EDA visualization script
2 python generate_eda_visualizations.py

```

This script generates all six types of visualizations automatically and saves them in the `eda_visualizations/` folder.

5.5 Activity 1.5: Data Preparation

Before we can use our data to teach our machine-learning model, we need to clean it up. This includes:

- **Handling missing values:** Remove or fill missing feedback entries
- **Combining columns:** Merge 'summary' and 'pros&cons' into a single 'feedback' column
- **Text cleaning:**
 - Convert to lowercase
 - Remove emojis
 - Remove URLs and email addresses
 - Remove special characters and punctuation
 - Remove numbers
 - Tokenization
 - Stopword removal
 - Lemmatization

Key Preprocessing Steps:

```

1 def clean_text(text):
2     # Convert to lowercase
3     text = str(text).lower()
4
5     # Remove emojis
6     text = emoji.replace_emoji(text, replace="")
7
8     # Remove URLs
9     text = re.sub(r'http\S+|www\S+|https\S+', '', text)
10
11    # Remove email addresses
12    text = re.sub(r'\S+@\S+', '', text)
13
14    # Remove numbers
15    text = re.sub(r'\d+', '', text)
16
17    # Remove special characters
18    text = re.sub(r'[^w\s]', ' ', text)
19
20    # Tokenize and remove stopwords
21    tokens = word_tokenize(text)
22    stop_words = set(stopwords.words('english'))
23    tokens = [lemmatizer.lemmatize(token) for token in tokens
24              if token not in stop_words and len(token) > 2]
25
26    return ' '.join(tokens)

```

5.6 Activity 1.6: Convert Ratings to Sentiment Labels

Convert overall ratings to sentiment categories:

- **1-2 stars** → **negative** sentiment
- **3 stars** → **neutral** sentiment
- **4-5 stars** → **positive** sentiment

```

1 def convert_rating_to_sentiment(rating):
2     if rating <= 2:
3         return 'negative'
4     elif rating == 3:
5         return 'neutral'
6     else: # 4 or 5
7         return 'positive'
8
9 df['sentiment'] = df['overall-ratings'].apply(
10    convert_rating_to_sentiment)

```

5.7 Activity 1.7: Check Class Balance

Analyze the distribution of sentiment classes to identify imbalanced datasets:

```

1 print( df['sentiment'].value_counts())
2 print(df['sentiment'].value_counts(normalize=True) * 100)

```

If the dataset is imbalanced (ratio > 2.0), consider applying SMOTE during training.

6 Milestone 2: Model Training & Evaluation

6.1 Activity 2.1: TF-IDF Vectorization

Convert text data into numerical features using TF-IDF (Term Frequency-Inverse Document Frequency):

```

1 vectorizer = TfidfVectorizer(
2     max_features=5000,
3     ngram_range=(1, 2), # Unigrams and bigrams
4     min_df=2,           # Minimum document frequency
5     max_df=0.95         # Maximum document frequency
6 )
7
8 X_train_tfidf = vectorizer.fit_transform(X_train)
9 X_test_tfidf = vectorizer.transform(X_test)

```

6.2 Activity 2.2: Train Machine Learning Models

Train multiple classifiers and compare their performance:

Naive Bayes:

```
1 from sklearn.naive_bayes import MultinomialNB
2 model = MultinomialNB(alpha=1.0)
3 model.fit(X_train_tfidf, y_train)
```

Logistic Regression:

```
1 from sklearn.linear_model import LogisticRegression
2 model = LogisticRegression(max_iter=1000, random_state=42,
3                             multi_class='multinomial')
4 model.fit(X_train_tfidf, y_train)
```

6.3 Activity 2.3: Model Evaluation

Evaluate model performance using multiple metrics:

```
1 y_pred = model.predict(X_test_tfidf)
2
3 accuracy = accuracy_score(y_test, y_pred)
4 precision = precision_score(y_test, y_pred, average='weighted')
5 recall = recall_score(y_test, y_pred, average='weighted')
6 f1 = f1_score(y_test, y_pred, average='weighted')
7
8 print(f"Accuracy: {accuracy:.4f}")
9 print(f"Precision: {precision:.4f}")
10 print(f"Recall: {recall:.4f}")
11 print(f"F1-Score: {f1:.4f}")
```

Expected Results:

- Accuracy: 72-75%
- Precision: 66-70%
- Recall: 72-75%
- F1-Score: 65-70%

6.4 Activity 2.4: Save Model and Vectorizer

Save the trained model and vectorizer for deployment:

```
1 joblib.dump(model, 'sentiment_model.pkl')
2 joblib.dump(vectorizer, 'vectorizer.pkl')
```

7 Milestone 3: Advanced Features Implementation

7.1 Activity 3.1: Theme Extraction

Extract top HR-related keywords and themes from feedback:

```

1 def extract_themes_from_text(text, keywords_list):
2     text_lower = text.lower()
3     found_themes = []
4
5     hr_keywords = ['salary', 'workload', 'culture', 'management',
6                     'stress', 'growth', 'team', 'support', 'bonus',
7                     'environment']
8
9     for keyword in hr_keywords:
10         if keyword in text_lower:
11             found_themes.append(keyword)
12
13 return found_themes[:5]

```

7.2 Activity 3.2: Emotion Detection

Detect emotions in feedback text:

```

1 def detect_emotions(text):
2     emotion_keywords = {
3         'joy': ['happy', 'great', 'amazing', 'wonderful',
4                 'excellent'],
4         'stress': ['stress', 'stressful', 'pressure',
5                    'overwhelming'],
5         'anger': ['angry', 'furious', 'mad', 'rage', 'hate'],
6         'frustration': ['frustrated', 'disappointed', 'upset',
7                          'unhappy'],
7         'motivation': ['motivated', 'inspired', 'encouraged',
8                         'energized']
8     }
9
10    # Calculate emotion scores and percentages
11    # Return emotion distribution

```

7.3 Activity 3.3: Mixed Feedback Logic

Implement advanced sentiment override logic for mixed feedback:

```

1 # Check for negative indicators
2 negative_indicators = ['too high', 'pressure', 'stressful',
3                         'bad', 'terrible']
4
5 # Strong negative phrases
6 strong_negative_phrases = ['toxic environment', 'caustic work',
7                             'no work life balance']

```

```

8
9 # Override logic: If strong negative phrases detected,
10 # adjust sentiment
11 if has_strong_negative:
12     if prob_dict.get('negative', 0) > prob_dict.get('neutral', 0):
13         sentiment = 'negative'
14     else:
15         sentiment = 'neutral'
16
17 # Critical override: If polarity is strongly negative (< -0.2),
18 # always negative
19 if polarity < -0.2:
20     sentiment = 'negative'

```

7.4 Activity 3.4: AI Summary Generation

Generate contextual AI summaries for HR insights:

```

1 def generate_ai_summary(feedback, sentiment, themes, emotions):
2     summary_parts = []
3
4     # Sentiment-based insights
5     if sentiment == 'negative':
6         summary_parts.append("Employees express concerns that
7                               require attention.")
8
9     # Theme-based insights
10    if 'workload' in themes:
11        summary_parts.append("Workload concerns suggest potential
12                               burnout risks.")
13
14    # Emotion-based insights
15    if emotions.get('stress', 0) > 30:
16        summary_parts.append("High stress levels detected -
17                               consider
18                               wellness programs.")
19
20    # Recommendations
21    recommendations = []
22    if sentiment == 'negative':
23        recommendations.append("HR should explore workload
24                               redistribution.")
25
26    return " ".join(summary_parts + recommendations)

```

8 Milestone 4: Web Application Development

8.1 Activity 4.1: Flask Backend API

Create RESTful API endpoints for real-time inference:

```

1 from flask import Flask, request, jsonify, render_template
2 from flask_cors import CORS
3 import joblib
4
5 app = Flask(__name__)
6 CORS(app)
7
8 # Load model and vectorizer
9 model = joblib.load('sentiment_model.pkl')
10 vectorizer = joblib.load('vectorizer.pkl')
11
12 @app.route('/analyze_feedback', methods=['POST'])
13 def analyze_feedback():
14     data = request.get_json()
15     feedback_text = data['feedback']
16
17     # Preprocess
18     cleaned_text = clean_text(feedback_text)
19
20     # Vectorize
21     text_tfidf = vectorizer.transform([cleaned_text])
22
23     # Predict
24     sentiment = model.predict(text_tfidf)[0]
25     probabilities = model.predict_proba(text_tfidf)[0]
26
27     # Advanced analysis
28     themes = extract_themes_from_text(cleaned_text, hr_keywords)
29     emotions = detect_emotions(cleaned_text)
30     ai_summary = generate_ai_summary(feedback_text, sentiment,
31                                     themes, emotions)
32
33     return jsonify({
34         'sentiment': sentiment,
35         'confidence': {
36             'positive': probabilities[0],
37             'neutral': probabilities[1],
38             'negative': probabilities[2]
39         },
40         'themes': themes,
41         'emotions': emotions,
42         'ai_summary': ai_summary
43     })

```

8.2 Activity 4.2: Additional API Endpoints

Implement additional endpoints for enhanced functionality:

- /random_sample: Get random feedback sample from dataset

- /department_sentiment: Get department-wise sentiment analysis
- /model_metrics: Get model evaluation metrics
- /export_report: Export analysis results as CSV

9 Frontend Implementation

The web interface is designed with a modern, professional HR dashboard approach, optimizing for clarity and usability.

9.1 Key Features of the Web Application

9.1.1 User Interface Components

- Professional HR dashboard header for a trustworthy presentation
- Comprehensive feedback analysis form for entering employee feedback text
- Mobile-responsive design using CSS Grid and Flexbox for universal access
- Dark/Light mode toggle for user preference
- Interactive charts using Chart.js for data visualization

9.1.2 Analysis Display Components

- **Sentiment Badge:** Color-coded sentiment indicator (Green for Positive, Yellow for Neutral, Red for Negative)
- **Confidence Progress Bars:** Visual representation of model confidence scores for each sentiment class
- **Theme Tags:** Display extracted HR-related themes as clickable tags
- **Emotion Visualization:** Pie chart or bar chart showing emotion distribution
- **AI Summary Box:** Generated insights and recommendations for HR
- **Department Sentiment Chart:** Interactive bar/line chart showing sentiment across departments
- **Model Accuracy Log Panel:** Display model evaluation metrics (accuracy, precision, recall, F1-score)

9.1.3 Advanced Features

- Random Sample Analyzer: Button to load and analyze random feedback from the dataset
- Export Functionality: Download analysis results as CSV file
- Real-time Inference: Instant analysis as user types or submits feedback
- Mixed Feedback Logic: Advanced sentiment classification for complex feedback

10 Application Screenshots and Workflow

The workflow moves sequentially from data input to comprehensive HR insights.

10.1 Home Page Interface

Features:

- Clean, modern interface with gradient header
- Prominent feedback input textarea
- Quick action buttons (Analyze, Random Sample, Export)
- Dark/Light mode toggle in navigation bar

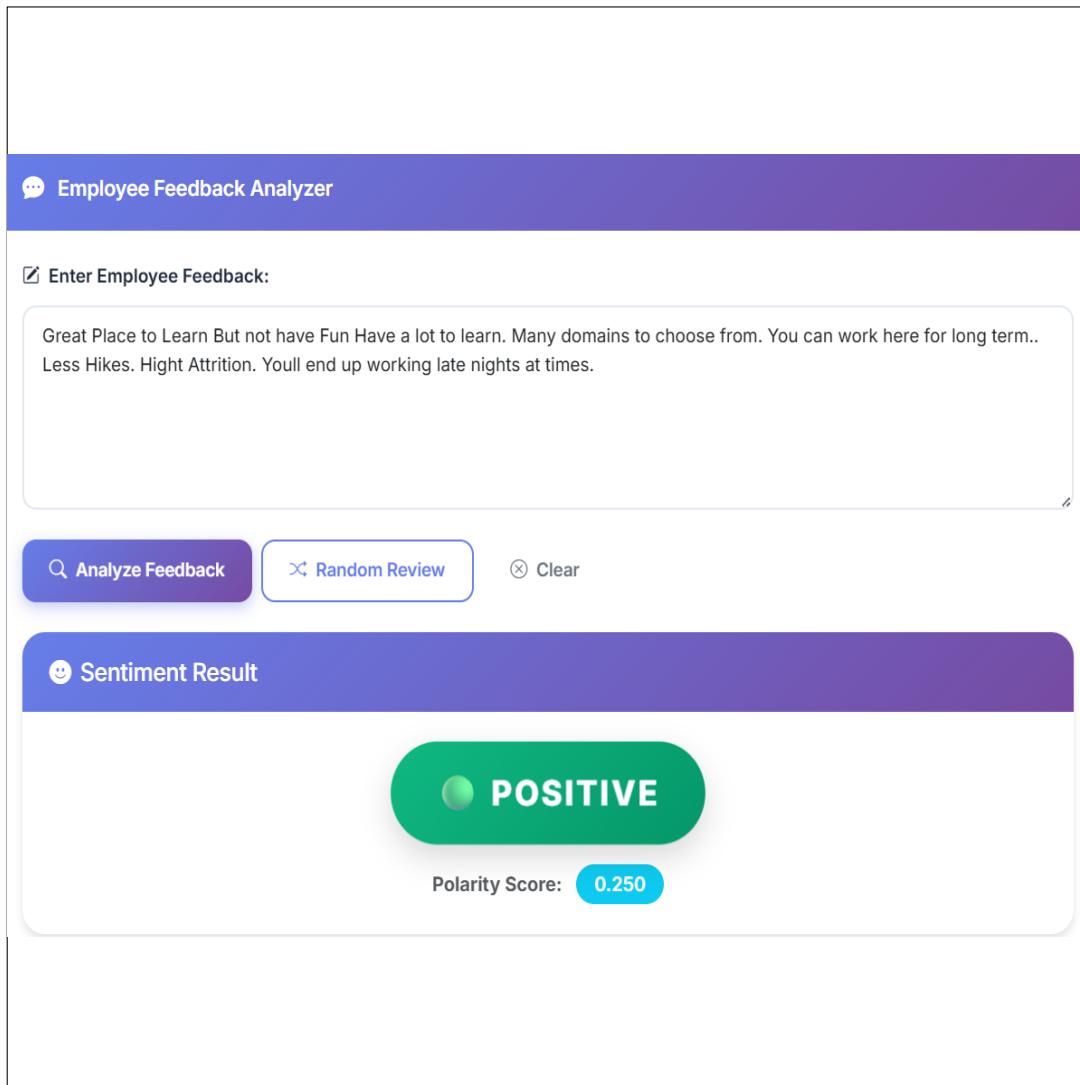


Figure 8: Page Interface

10.2 Employee Feedback Input Form

Form Components:

- **Feedback Text Area:** Large text input for employee feedback (supports multi-line text)
- **Analyze Button:** Triggers real-time sentiment analysis
- **Random Sample Button:** Loads random feedback from dataset for testing
- **Clear Button:** Resets the form

10.3 Analysis Results Display

Results Include:

- **Sentiment Classification:** Large badge showing "Positive", "Neutral", or "Negative" with color coding
- **Confidence Scores:** Progress bars showing percentage confidence for each sentiment class
- **Polarity Score:** TextBlob polarity score (-1 to +1) displayed with interpretation
- **Extracted Themes:** Tag-based display of identified themes
- **Emotion Distribution:** Visual chart (pie or bar) showing detected emotions
- **AI Summary:** Generated insights and recommendations
- **Department Sentiment Chart:** Interactive Chart.js visualization showing sentiment breakdown by department
- **Model Metrics Panel:** Display of model evaluation metrics

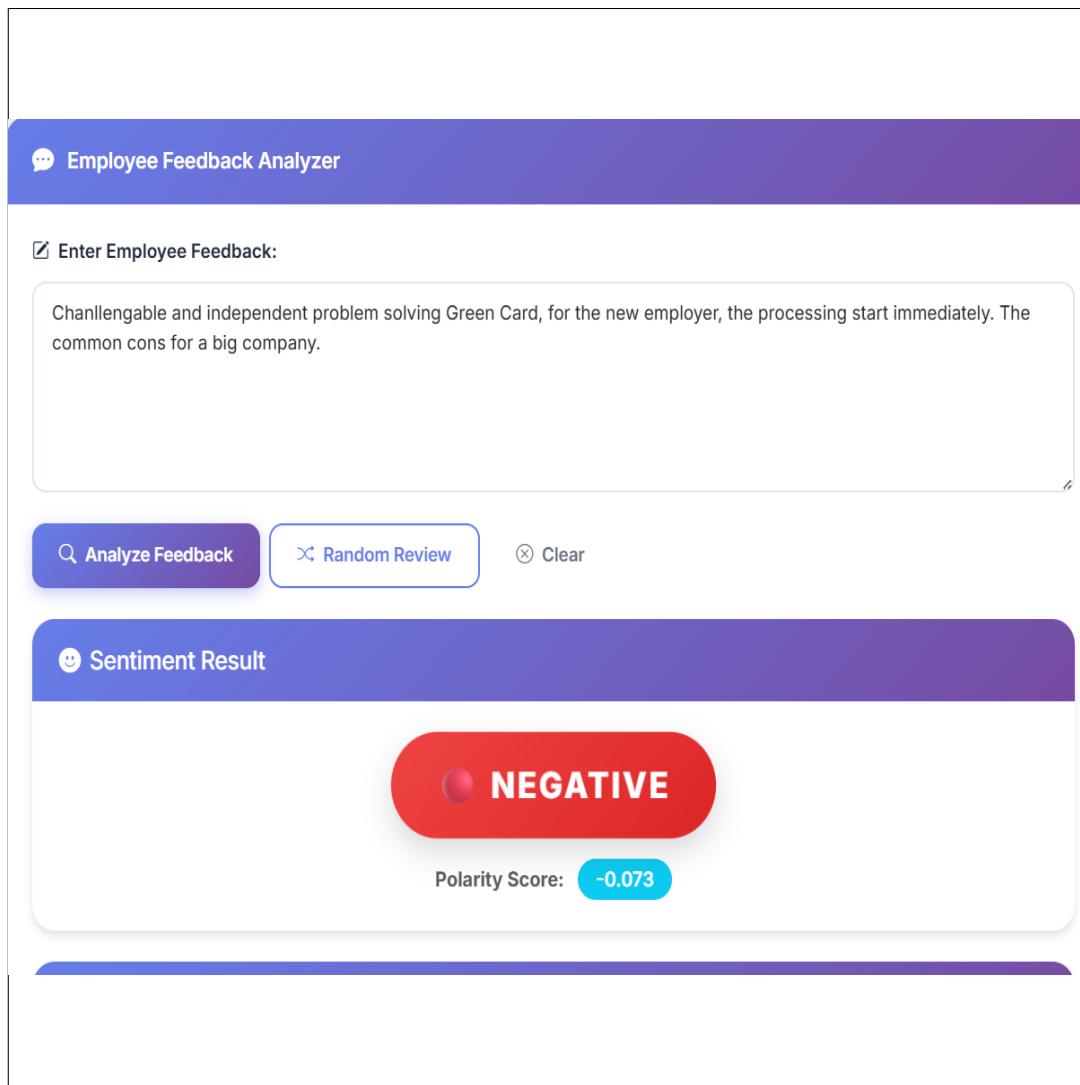


Figure 9: Analysis Results Display

10.4 Export Functionality

Users can export analysis results as CSV files containing:

- Timestamp
- Original feedback text
- Cleaned text
- Sentiment classification
- Confidence scores
- Polarity score
- Extracted themes
- Emotion distribution
- AI summary

11 Model Performance Metrics

Metric	Value
Accuracy	72.10%
Precision	66.04%
Recall	72.10%
F1-Score	64.85%

Table 2: Model Performance Metrics

12 Future Implementations

Future plans for the AI-Driven Employee Feedback Analysis System focus on deployment, integration, and model enhancement:

12.1 Advanced NLP Models

- **Transformer-based Models:** Integrate DistilBERT or BERT for improved sentiment classification accuracy
- **Fine-tuning:** Fine-tune pre-trained transformer models on domain-specific employee feedback data
- **Ensemble Methods:** Combine multiple models (Naive Bayes, Logistic Regression, Transformers) using voting or stacking

12.2 Real-time Integration

- **HRIS Integration:** Develop API endpoints to integrate directly into Human Resources Information Systems (HRIS)
- **Slack/Teams Bot:** Create chatbot integration for real-time feedback analysis in workplace communication platforms
- **Email Integration:** Automatically analyze feedback from email surveys and generate reports

12.3 Advanced Analytics

- **Temporal Analysis:** Track sentiment trends over time to identify patterns and seasonal variations
- **Predictive Analytics:** Predict employee turnover risk based on sentiment patterns and historical data
- **Comparative Analysis:** Compare sentiment across departments, job roles, locations, and time periods

12.4 Enhanced User Experience

- **Mobile Application:** Develop native mobile apps (iOS/Android) for HR managers to analyze feedback on-the-go
- **Dashboard Customization:** Allow users to customize dashboard views and create personalized reports
- **Multi-language Support:** Extend sentiment analysis to support multiple languages using multilingual models

13 Conclusion

The AI-Driven Employee Feedback Analysis System successfully developed an accurate and highly interpretable AI-powered sentiment analysis system for HR insights. By utilizing Naive Bayes and Logistic Regression algorithms with TF-IDF vectorization, the model achieved reliable classification based on employee feedback text. The final product is a robust, full-stack web application (built with Flask and scikit-learn) that seamlessly integrates the trained model into an intuitive, modern HR dashboard interface.

The system goes beyond simple sentiment classification by providing:

- **Advanced Analysis:** Theme extraction, emotion detection, and AI-powered summaries
- **Department Insights:** Department-wise sentiment analysis for targeted interventions
- **Mixed Feedback Handling:** Sophisticated logic to accurately classify complex feedback
- **User-Friendly Interface:** Modern, responsive UI with dark/light mode and interactive visualizations
- **Export Capabilities:** CSV export functionality for reporting and record-keeping

This solution offers a reliable, scalable, and instant tool for preliminary employee sentiment assessment, proving the project's success from data science development to practical HR deployment. The system enables HR professionals to make data-driven decisions, identify areas of concern proactively, and improve workplace culture and employee satisfaction.

14 Technical Specifications

14.1 Model Architecture

- **Algorithm:** Multinomial Naive Bayes / Logistic Regression
- **Vectorization:** TF-IDF with n-grams (1, 2)
- **Features:** 5000 max features
- **Classes:** 3 (Positive, Neutral, Negative)

14.2 System Requirements

- **Python Version:** 3.8+
- **RAM:** Minimum 4GB (8GB recommended)
- **Storage:** 500MB for dataset and model files
- **Browser:** Modern browsers (Chrome, Firefox, Edge, Safari)

14.3 API Endpoints

- GET /: Render main UI page
- POST /analyze_feedback: Analyze employee feedback
- GET /random_sample: Get random feedback sample
- GET /department_sentiment: Get department-wise analysis
- GET /model_metrics: Get model evaluation metrics
- POST /export_report: Export analysis as CSV
- GET /health: Health check endpoint

15 References and Resources

1. scikit-learn Documentation: <https://scikit-learn.org/stable/>
2. NLTK Documentation: <https://www.nltk.org/>
3. Flask Documentation: <https://flask.palletsprojects.com/>
4. TextBlob Documentation: <https://textblob.readthedocs.io/>
5. Chart.js Documentation: <https://www.chartjs.org/>
6. TF-IDF Vectorization: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
7. Naive Bayes Classifier: <https://scikit-learn.org/stable/modules/numpy/numpy.html>
8. Logistic Regression: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html