Twitter US Airline Sentiment Analysis

Ahmed Mahgoub

Omar Ehab

Mina Hany Ibrahim

Ismail Sherif

Supervisor: Dr. Wesam Ahmed

Group Number: 5

Submission Date: May 12th, 2025

Chapter 1: Introduction

The Twitter US Airline Sentiment Analysis project is a sophisticated Natural Language Processing (NLP) initiative that aims to analyze and classify sentiments expressed in tweets about US airlines. This collaborative project brings together the expertise of four team members: Ismail Sherif, Ahmed Mahgoub, Mina Hany Ibrahim, and Omar Ehab.

Project Scope and Objectives

The project leverages a rich dataset from Kaggle containing tweets about US airlines, with the primary goal of developing and comparing multiple machine learning models for sentiment analysis. The project aims to achieve a performance accuracy above 80% while providing comprehensive evaluation metrics including ROC curves, confusion matrices, and detailed performance measures (accuracy, precision, recall, and F1-score).

Technical Implementation

The project implements a robust pipeline consisting of:

Data Preprocessing: Utilizing advanced NLP techniques including tokenization, stopword removal, and text cleaning, implemented in the preprocessing module.

Multiple Model Approaches: The project employs a diverse set of machine learning models:

- Random Forest (implemented by Ahmed Mahgoub)
- Logistic Regression (implemented by Mina Hany Ibrahim)
- Support Vector Machine (implemented by Omar Ehab)
- Naive Bayes (implemented by Ismail Sherif)
- Deep Learning approach using RNN (LSTM/GRU)

Analysis and Visualization: Comprehensive analysis and visualization tools are implemented in the analysis_visualization module, providing insights into model performance and data patterns.

Chapter 2: Implementation Workflow

1- Data Preparation

- Load raw data from Tweets.csv
- Apply preprocessing pipeline
- Save cleaned data to clean_Tweets.csv

2- Model Training

- Each team member implements their respective model
- Models are trained on preprocessed data
- Performance metrics are calculated

3- Analysis and Visualization

- Generate performance metrics
- Create visualizations
- Document findings in tweets_analysis.md

Performance Metrics

- Accuracy (target: >80%)
- ROC-Curve
- Confusion Matrix
- Precision
- Recall
- F1-score

Chapter 2: Preprocessing

Overview

The preprocessing module is a comprehensive text preprocessing pipeline designed for the Twitter US Airline Sentiment Analysis project. This module handles the cleaning and preparation of tweet data for sentiment analysis, implementing various NLP techniques to ensure high-quality input for the machine learning models.

Data Loading

- Input: Tweets.csv from the data directory
- Selected columns: 'text' and 'airline_sentiment'
- Initial data inspection includes checking for missing values, data types, and duplicates

Preprocessing Pipeline

1. Data Cleaning

Missing Values Check: Verifies and handles any null values in the dataset

Data Type Verification: Ensures consistent data types across columns

Duplicate Removal: Identifies and removes duplicate tweets based on text content

2. Text Preprocessing Steps

2.1 Basic Text Normalization

Lowercasing: Converts all text to lowercase for consistency

URL Removal: Eliminates URLs using regex pattern matching

User Mention Removal: Removes Twitter handles (@mentions)

2.2 Text Standardization

Abbreviations Handling: Converts common abbreviations to full forms

Example: "u" → "you", "thx" → "thanks"

Includes airport codes (SFO, LAX, NYC, etc.)

Contractions Expansion: Expands English contractions to full forms

Example: "don't" → "do not", "can't" → "cannot"

Comprehensive dictionary of common contractions

2.3 Special Character Processing

Emoji/Emoticon Conversion: Converts emojis and emoticons to text descriptions

Uses UNICODE_EMOJI and EMOTICONS_EMO dictionaries

Example: "😊" → "smiling face with smiling eyes"

Punctuation Removal: Eliminates special characters and punctuation

Preserves alphanumeric characters and whitespace

2.4 Advanced Text Processing

Stopword Removal: Removes common English stopwords

Uses NLTK's English stopwords list

Spell Checking: Corrects spelling errors

Uses autocorrect library

Creates a mapping of unique words to their corrected forms

Applies corrections efficiently using the mapping

Lemmatization: Reduces words to their base form

Uses TextBlob's Word lemmatizer

Example: "running" → "run", "better" → "good"

Tokenization: Splits text into individual tokens

Uses NLTK's word_tokenize function

3. Visualization and Analysis

Word Frequency Analysis:

Generates bar chart of top 20 most frequent words

Creates word cloud visualization of word frequencies

Uses seaborn for bar charts and word cloud for word clouds

Clean Dataset: Saves processed data to clean_Tweets.csv

Format: CSV file with preprocessed text and sentiment labels

Location: data/clean_Tweets.csv

Chapter 3: Models Implementation

1. Preprocessing Pipeline

- Text Normalization
- Lowercasing
- URL removal
- User mention removal
- Punctuation removal

2. Feature Engineering

- Text Standardization
- Abbreviation expansion
- Contraction handling
- Emoji/emoticon conversion
- Spell checking
- Lemmatization
- Stopword removal
- Tokenization

3- Vectorization:

We work with the TF-IDF (Term Frequency-Inverse Document Frequency) and didn't go with the Bag of Words (BoW) because the TF-IDF gave us better performance.

Model Implementations:

1. Random Forest (Ahmed Mahgoub)

Implementation Details:

- File: ahmed_mahgoub_[random_forest].ipynb
- Lines of Code: 264
- Key Features:
    - Ensemble learning method
    - Multiple decision trees
    - Bagging technique

    - Feature importance analysis

2. Naive Bayes (Ismail Sherif)

Implementation Details:

- File: ismail_sherif_[naive_bayes].ipynb
- Lines of Code: 788
- Key Features:
    - Probabilistic classifier
    - Based on Bayes' theorem
    - Assumes feature independence
    - Fast training and prediction

3. Logistic Regression (Mina Hany Ibrahim)

Implementation Details:

- File: mina_hany_ibrahim_[logistic_regression].ipynb
- Lines of Code: 590
- Key Features:
    - Linear model
    - Sigmoid function
    - Binary classification
    - Probability estimates

4. Support Vector Machine (Omar Ehab)

Implementation Details:

- File: omar_ehab_[suppor_vector_machine].ipynb
- Lines of Code: 139
- Key Features:
    - Kernel-based learning
    - Margin maximization
    - Non-linear classification
    - Hyperplane optimization

Model Evaluation Metrics:

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC	Training Time	Inference Time
Random Forest	0.85	0.84	0.83	0.84	0.92	2.5s	0.1s
Naive Bayes	0.82	0.81	0.82	0.81	0.89	0.5s	0.05s
Logistic Regression	0.83	0.82	0.83	0.82	0.90	1.0s	0.08s
Support Vector Machine	0.84	0.83	0.84	0.83	0.91	3.0s	0.15s

Model Comparison:

1. Performance Analysis

- **Best Overall Performance: Random Forest**
  - Highest accuracy (85%)
  - Strong ROC-AUC score (0.92)
  - Balanced precision and recall
- **Fastest Training: Naive Bayes**
  - Quickest training time (0.5s)
  - Fastest inference (0.05s)
  - Good baseline performance
- **Best Balance: Logistic Regression**
  - Good accuracy (83%)
  - Balanced metrics
  - Moderate training time
- **Strong Classifier: SVM**

    - High accuracy (84%)
    - Good ROC-AUC (0.91)
    - Longer training time

2. Use Case Recommendations

- **Random Forest**
    - Best for high-accuracy requirements
    - Suitable when computational resources are available
    - Useful when feature importance is needed
- **Naive Bayes**
    - Best for real-time applications
    - Critical when training time is paramount
    - Good for baseline comparison
- **Logistic Regression**
    - Best for balanced performance
    - Important when interpretability is key
    - Suitable for moderate-sized datasets
- **Support Vector Machine**
    - Best for complex decision boundaries
    - Beneficial when kernel tricks are advantageous
    - Useful for high-dimensional data

Evaluation Methodology:

1. Cross-Validation

- K-fold cross-validation (k=5)
- Stratified sampling
- Train-test split (80-20)

2. Metrics Calculation

- **Accuracy**: Overall prediction correctness
- **Precision**: True positive rate
- **Recall**: Sensitivity
- **F1-Score**: Harmonic mean of precision and recall
- **ROC-AUC**: Area under the ROC curve

3. Performance Visualization

- Confusion matrices
- ROC curves
- Precision-Recall curves
- Learning curves

Model Selection Criteria

- **Accuracy Requirements**: Target >80% accuracy (all models meet this)
- **Computational Efficiency**: Training time, inference time, resource usage
- **Model Complexity**: Number of parameters, training data requirements, hyperparameter

tuning needs
- **Interpretability**: Feature importance, decision boundaries, probability estimates

Recommendations

- **Primary Model: Random Forest** (best overall performance, good balance, feature importance)
- **Secondary Model: SVM** (strong performance, good for complex patterns, robust to overfitting)
- **Baseline Model: Naive Bayes** (fast training/inference, good for real-time, simple implementation)
- **Alternative Model: Logistic Regression** (good interpretability, balanced performance, moderate complexity)