# Mini Project
## UCS2604 Principles of Machine Learning

# Sentiment Analysis of Airline Tweets

Janani Hariharakrishnan 3122 22 5001 046, CSE A, 3rd year
Harsh Pratap Singh 3122 22 5001 038, CSE A, 3rd year
Diya Seshan 3122 22 5001 030, CSE A, 3rd year

**Abstract**

The growing presence of airline related content on social media platforms like Twitter has made real-time sentiment analysis a crucial tool for customer service and brand reputation management. This study aims to develop a machine learning-based sentiment analysis system to classify tweets about U.S. airlines into positive, neutral, and negative sentiments. Additionally, we also aim to detect hints of sarcasm- a perspective that is often overlooked by traditional sentiment analysis algorithms. By leveraging supervised learning techniques, the proposed model enables airlines to monitor public opinions, address customer complaints efficiently, and aim to improve overall passenger satisfaction.

## 1 Introduction

Airlines receive a vast amount of customer feedback on Twitter, covering complaints about delays, poor service, or lost baggage, as well as praises for smooth experiences. Manually analyzing this data is time-consuming and inefficient, making it difficult for airlines to respond promptly and enhance customer satisfaction. By leveraging machine learning and deep learning techniques, this study aims to automate sentiment classification, enabling real-time monitoring of customer sentiments. This dataset presents a mix of numerical and textual features, offering a rich foundation for sentiment analysis. The goal is to develop a predictive model that accurately determines the sentiment of a tweet based on various attributes, such as the text content, posting details, and user engagement metrics. Automating this process allows airlines to gain actionable insights, enhance customer service, and address concerns proactively.

## 2 Literature Survey

Sentiment analysis plays a vital role in gauging public opinion on social media. Over the years, various approaches have been successful in this domain, ranging from traditional machine learning techniques to more recent deep learning methods. Initially, models like Support Vector Machines (SVMs) and Naive Bayes were widely used for sentiment classification, leveraging handcrafted features

such as term frequency and n-grams (5). However, the advent of deep learning has taken sentiment analysis to a much higher level, with Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks improving the ability to capture sequential dependencies in text (4).

More recently, transformer-based models like BERT and its multilingual variants, such as mBERT, have set new benchmarks in sentiment analysis (2). Transformer-based studies have revolutionized sentiment analysis by using contextual embeddings to enhance sentiment classification tasks. Studies have demonstrated that fine-tuning BERT on airline-specific tweets leads to substantial improvements in sentiment detection accuracy.(1)

### 2.1 Identified Gaps

Despite significant advancements in sentiment analysis, several challenges remain that hinder its effectiveness in real-world applications. **Contextual ambiguity** continues to be a major issue, as models struggle with expressions where sentiment depends on broader discourse or external knowledge. **Sarcasm and irony detection** also pose difficulties since these require an advanced understanding of tone, intent, and cultural nuances beyond lexical cues. Additionally, sentiment models often perform poorly on **domain-specific texts**, particularly when applied to new datasets without extensive fine-tuning, as slang, jargon, and informal language can vary widely.Addressing these gaps requires further research into hybrid approaches that integrate linguistic and contextual knowledge, improved sarcasm detection mechanisms, and lightweight transformer variants optimized for real-time applications.

## 3 Problem Statement

Airlines receive a large volume of customer feedback through Twitter, ranging from complaints about delays to praises for exceptional service. However, manually analyzing this data is inefficient, and existing systems often struggle to accurately interpret sentiments, especially sarcasm, leading to delayed responses and missed opportunities for service improvement. This study proposes a real-time sentiment analysis system that not only classifies and monitors customer sentiments automatically but also detects sarcasm in feedback, ensuring more accurate interpretation of customer emotions.

## 4 Novelty

Our study aims to address the challenge of **sarcasm detection and contextual ambiguity** in airline sentiment analysis. While most sentiment analysis models struggle with detecting sarcasm—where a negative sentiment is expressed using positive words—our approach integrates context-aware sentiment analysis using a transformer-based model, namely RoBERTa, which has been fine-tuned on sarcasm-labeled datasets. Additionally, we incorporate both text

features and user engagement metrics (such as likes and retweets) to enhance the classification accuracy.

By integrating these techniques, our model aims to provide more accurate sentiment classification, allowing airlines to better understand customer concerns and improve their services accordingly.

# 5 System Design Architecture

The proposed system consists of the following components:

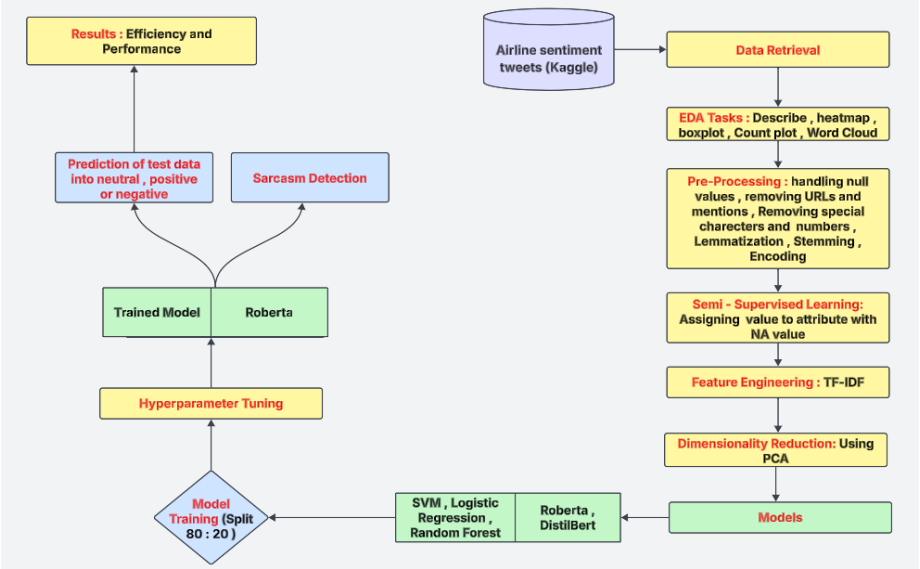



Figure 1: System Architecture Diagram

# 6 Implementation and Experiments

## 6.1 Development Environment

Below is a description of the tools and platforms used to build, test, and deploy the application. This includes cloud services, programming languages, frameworks, and deployment utilities that support end-to-end development.

## 6.2 Dataset Description

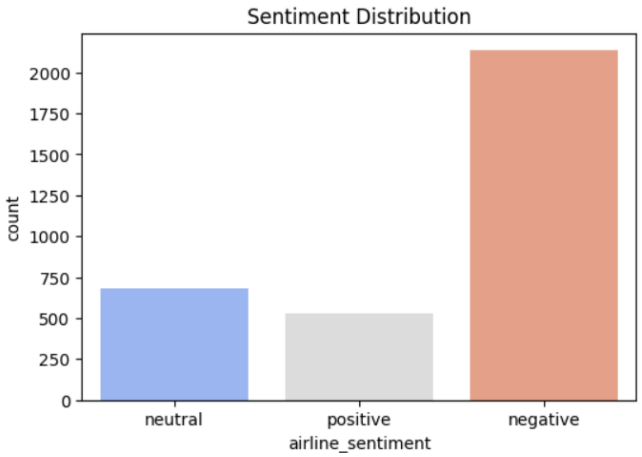
The dataset consists of airline tweets labeled as positive, neutral, or negative. The total dataset consists of :

| Component | Tech Stack | Tools/Services |
|---|---|---|
| **Model Training** | Python (Colab + GPU) | Scikit-learn, TensorFlow/Keras, PyTorch, Hugging Face Transformers |
| **Frontend** | HTML, CSS, Jinja2, Bootstrap | - |
| **Backend** | Flask (REST API) | Gunicorn |
| **Deployment** | Google Cloud | GCP, gcloud CLI |
| **Version Control** | Git | GitHub |

Table 1: Tech Stack and Tools/Services

| Dataset Name | Dataset Link |
|---|---|
| Airline Sentiment Tweets | Kaggle Sentiment Dataset |

- 3339 unique entries
- 12 columns




## 6.3 Implementation Steps

- **Step 1: Exploratory Data Analysis (EDA)**
  - Bar Graph , Box plot
  - Count plot
  - Heat map , Word cloud
- **Step 2: Data Preprocessing**
  - Removing stop words, special characters, and URLs
  - Tokenizing text

| Description | Number of Features | Type of Models Used | Model Names | Classification |
|---|---|---|---|---|
| • Includes text-based tweets, sentiment labels, and metadata such as airline names and tweet sources.<br>• Used for sentiment analysis, NLP model training, and understanding customer feedback in the airline industry. | 12 (tweet_id, airline_sentiment, airline_sentiment_confidence, name, retweet_count, text, tweet_created, user_timezone, airline, etc.) | Machine Learning-Supervised | SVM, Logistic Regression, Random Forest | • Positive<br>• Negative<br>• Neutral |

Table 2: Dataset Information and Models Used

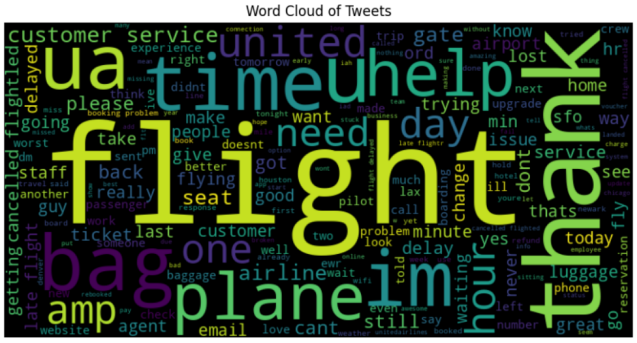
  - Lemmatization and stemming

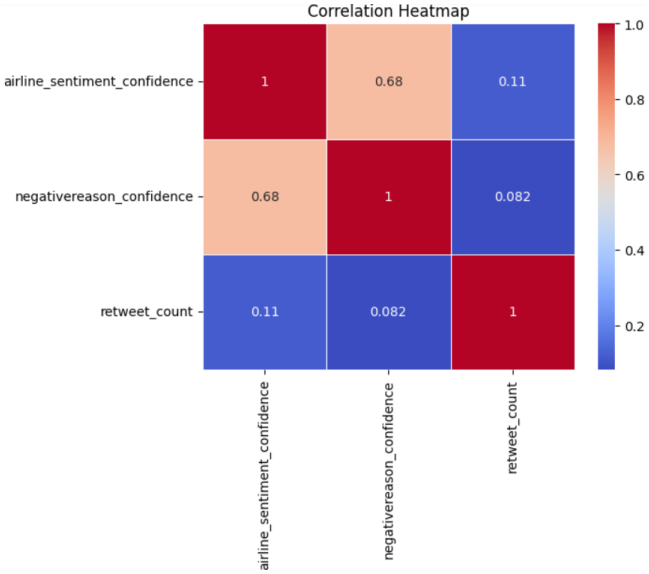
- **Step 3: Handling NaN values**

  Our dataset contained a large number of NaN values- nearly 50% of the data was lacking the target label, hence leading to poor model performance due to data sparsity. Since dropping these entries would result in a major loss of valuable data, we employed a **Semi-Supervised Self-Training approach (Iterative Pseudo-Labeling)** to handle missing values in the negativereason column without discarding data. Initially, a Random Forest Classifier was trained on labeled data using TF-IDF features, and its performance was validated. The model then iteratively predicted labels for missing values, selecting only high-confidence predictions ($\geq 90\%$) to be added as pseudo-labels to the training set. This process was repeated for 5 iterations, progressively improving label assignment. Finally, the NaN values were filled using the model's predictions, ensuring a semi-supervised approach to infer missing categories while maintaining data integrity.

- **Step 4: Feature Engineering**

  - TF-IDF Vectorization




  - Word Embeddings (Word2Vec, GloVe)

- **Step 4:Experimenting with Machine Learning Models**
  - Logistic Regression
  - Support Vector Machine (SVM)
  - Random Forest Classifier
  - Deep Learning (LSTM)
- **Step 5: Model Evaluation**
  - Accuracy, Precision, Recall, F1-score
  - Confusion Matrix for performance analysis
- **Step 6: Deployment**

- Deploy the trained model on AWS/Azure/GCP
- Use streaming services like Kafka or Flask API for real-time monitoring

## 6.4 Transformer Models

- **Step 1: Dataset Preparation**
  - Load the dataset and select relevant columns (`text`, `airline_sentiment`).
  - Drop missing values and encode sentiment labels (Negative $\rightarrow$ 0, Neutral $\rightarrow$ 1, Positive $\rightarrow$ 2).
  - Split into training (85%) and testing (15%) sets.
- **Step 2: Tokenization and Data Loading**
  - Tokenize text using the RoBERTa tokenizer with `max_length=64`.
  - Convert to PyTorch tensors and create `DataLoader` with batch size 16.
- **Step 3: Model Setup**
  - Load the pre-trained RoBERTa model for sequence classification (3 output labels).
  - Use `AdamW` optimizer, Cross-Entropy Loss, and a linear learning rate scheduler.
- **Step 4: Mixed Precision Training**
  - Utilize `torch.cuda.amp` for faster training with automatic mixed precision.
  - Train for 3 epochs, tracking progress with `tqdm`.
- **Step 5: Model Saving and Evaluation**
  - Save the trained model as a `pickle` file.
  - Reload the model, perform inference, and compute accuracy, precision, recall, and F1-score.

## 6.5 Python Code Implementation

### 6.5.1 Assignment 1 Implementation- Basic Models and Sarcasm Detection

```
import re
import nltk
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
import seaborn as sns
from wordcloud import WordCloud
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
from transformers import pipeline

# Download necessary NLTK resources
nltk.download('stopwords')
nltk.download('wordnet')

# Load Dataset
df = pd.read_csv('airline_train.csv')  # Update path if necessary

# Data Preprocessing
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

def preprocess_text(text):
    if not isinstance(text, str):  # Handle NaN values
        return ""
    text = text.lower()
    text = re.sub(r'http\S+', '', text)  # Remove URLs
    text = re.sub(r'@[A-Za-z0-9_]+', '', text)  # Remove mentions
    text = re.sub(r'[^a-zA-Z\s]', '', text)  # Remove special
        characters & numbers
    words = text.split()
    words = [lemmatizer.lemmatize(word) for word in words if word
        not in stop_words]
    return " ".join(words)

df['clean_text'] = df['text'].apply(preprocess_text)

# Exploratory Data Analysis
plt.figure(figsize=(6, 4))
sns.countplot(x=df['airline_sentiment'], palette="coolwarm")
plt.title("Sentiment Distribution")
plt.show()

wordcloud = WordCloud(width=800, height=400, background_color="
    black").generate(" ".join(df['clean_text']))
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.title("Word Cloud of Tweets")
plt.show()

# Feature Engineering (TF-IDF)
tfidf_vectorizer = TfidfVectorizer(max_features=5000)
X = tfidf_vectorizer.fit_transform(df['clean_text'])

sentiment_mapping = {'positive': 1, 'neutral': 0, 'negative': -1}
```

```
df['sentiment_label'] = df['airline_sentiment'].map(
    sentiment_mapping)
y = df['sentiment_label']

# Split Data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
    =0.2, random_state=42)

# Train ML Models
models = {
    "Logistic Regression": LogisticRegression(),
    "SVM": SVC(kernel='linear'),
    "Random Forest": RandomForestClassifier(n_estimators=100)
}

for name, model in models.items():
    print(f"\nTraining {name}...")
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(f"\n{name} Performance:\n")
    print(classification_report(y_test, y_pred))
    print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))

# Sarcasm Detection Using RoBERTa
sarcasm_detector = pipeline("text-classification", model="
    cardiffnlp/twitter-roberta-base-sentiment")

def detect_sarcasm(text):
    prediction = sarcasm_detector(text)[0]
    return prediction['label'] == "LABEL_2"

df['sarcasm_detected'] = df['clean_text'].apply(detect_sarcasm)

# Evaluate Sarcasm Detection
sarcasm_count = df['sarcasm_detected'].sum()
print(f"\nTotal Sarcastic Tweets Detected: {sarcasm_count}")

# Model Deployment

def predict_sentiment(tweet):
    processed_tweet = preprocess_text(tweet)
    features = tfidf_vectorizer.transform([processed_tweet])
    sentiment = models["Logistic Regression"].predict(features)[0]
    sarcasm = detect_sarcasm(tweet)
    sentiment_map = {1: "Positive", 0: "Neutral", -1: "Negative"}
    sentiment_result = sentiment_map[sentiment]
    if sarcasm:
        sentiment_result += " (Sarcasm detected)"
    return sentiment_result
def best_airline(df):
    airline_sentiment_counts = df.groupby('airline')['
        airline_sentiment'].value_counts().unstack(fill_value=0)
    airline_sentiment_counts['positive_ratio'] =
        airline_sentiment_counts['positive'] / (
        airline_sentiment_counts['negative'] + 1)  # Avoid division
         by zero
```

```
    best_airline = airline_sentiment_counts['positive_ratio'].
        idxmax()
    return best_airline


# Test the model
test_tweet = "@airline Great job on the 5-hour delay. Best
    experience ever! "
print("\nSample Tweet Analysis:")
print(f"Tweet: {test_tweet}")
print(f"Predicted Sentiment: {predict_sentiment(test_tweet)}")
best_airline_name = best_airline(df)
print(f"\  n     Best Airline Based on Sentiment Analysis: {
    best_airline_name}")
```

### 6.5.2 Assignment 2 Implementation- Self Training to handle NaN Values

```
from sklearn.model_selection import GridSearchCV

df = airline

labeled_data = df.dropna(subset=['negativereason'])
unlabeled_data = df[df['negativereason'].isna()]

X_labeled = labeled_data['text']
y_labeled = labeled_data['negativereason']
X_unlabeled = unlabeled_data['text'].fillna('')  # Fill NaNs in
    text

# Train-test split for validation
X_train, X_test, y_train, y_test = train_test_split(X_labeled,
    y_labeled, test_size=0.2, random_state=42)

# TF-IDF Vectorization
vectorizer = TfidfVectorizer(max_features=5000)
X_train_tfidf = vectorizer.fit_transform(X_train)  #     Fit on
    labeled data
X_test_tfidf = vectorizer.transform(X_test)
X_unlabeled_tfidf = vectorizer.transform(X_unlabeled)

# Initialize Classifier
clf = RandomForestClassifier(n_estimators=200, random_state=42)
clf.fit(X_train_tfidf, y_train)

y_pred = clf.predict(X_test_tfidf)
print(f"Initial Model Accuracy: {accuracy_score(y_test, y_pred):.2f
    }")

# HANDLING NAN VALUES
# Self-training Loop (Iterative Pseudo-Labeling), with 90%
    confidence
CONFIDENCE_THRESHOLD = 0.9
MAX_ITER = 5

```

```
for i in range(MAX_ITER):
    print(f"\  n     Iteration {i+1} - Self-Training")

    # Predict probabilities for unlabeled data
    y_proba = clf.predict_proba(X_unlabeled_tfidf)
    y_pseudo = clf.classes_[np.argmax(y_proba, axis=1)]  # Get
        class with highest probability
    max_proba = np.max(y_proba, axis=1)  # Get max probability for
        each prediction

    # Select high-confidence predictions
    high_confidence_mask = max_proba >= CONFIDENCE_THRESHOLD
    newly_labeled = X_unlabeled_tfidf[high_confidence_mask]
    new_labels = y_pseudo[high_confidence_mask]

    if newly_labeled.shape[0] == 0:
        print("No new confident samples. Stopping self-training.")
        break

    # Add new pseudo-labeled samples to training set
    X_train_tfidf = vstack([X_train_tfidf, newly_labeled])  #
        Maintain sparse format
    y_train = np.hstack([y_train, new_labels])

    # Retrain classifier
    clf.fit(X_train_tfidf, y_train)
    print(f"Added {newly_labeled.shape[0]} pseudo-labeled samples."
        )

# Assign final labels to dataset
final_predictions = clf.predict(X_unlabeled_tfidf)
df.loc[df['negativereason'].isna(), 'negativereason'] =
    final_predictions

print("Self-training complete. Labels assigned.")
```

### 6.5.3 Assignment 2 Implementation- Hyperparameter Tuning

```
df['clean_text'] = df['text'].apply(preprocess_text)

# Feature Engineering (TF-IDF)
tfidf_vectorizer = TfidfVectorizer(max_features=5000)
X = tfidf_vectorizer.fit_transform(df['clean_text'])

# Encode Sentiment Labels
sentiment_mapping = {'positive': 1, 'neutral': 0, 'negative': -1}
df['sentiment_label'] = df['airline_sentiment'].map(
    sentiment_mapping)

y = df['sentiment_label']

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
    =0.2, random_state=42)

# Hyperparameter Tuning
```

```
param_grid = {
    "Logistic Regression": {
        "model": LogisticRegression(),
        "params": {"C": [0.01, 0.1, 1, 10], "solver": ['liblinear',
            'lbfgs']}
    },
    "SVM": {
        "model": SVC(),
        "params": {"C": [0.1, 1, 10], "kernel": ['linear', 'rbf']}
    },
    "Random Forest": {
        "model": RandomForestClassifier(),
        "params": {"n_estimators": [50, 100, 200], "max_depth": [
            None, 10, 20]}
    }
}

best_models = {}
for name, cfg in param_grid.items():
    print(f"\ n     Tuning {name}...")
    grid_search = GridSearchCV(cfg["model"], cfg["params"], cv=3,
        scoring='accuracy', n_jobs=-1)
    grid_search.fit(X_train, y_train)
    best_models[name] = grid_search.best_estimator_

    print(f"\ n   Best {name} Model: {grid_search.best_params_}")
    print(f"Training Accuracy: {grid_search.best_estimator_.score(
        X_train, y_train)}")
    print(f"Testing Accuracy: {grid_search.best_estimator_.score(
        X_test, y_test)}")
    y_pred = best_models[name].predict(X_test)
    print(classification_report(y_test, y_pred))

# Save best parameters for fine-tuning
best_params = {name: model.get_params() for name, model in
    best_models.items()}
pd.DataFrame(best_params).to_csv("/content/best_hyperparameters.csv
    ", index=False)
```

### 6.5.4 Assignment 2 Implementation- Dimensionality Reduction using PCA

```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
# Load Dataset
df = airline

df['clean_text'] = df['text'].apply(preprocess_text)

# Feature Engineering (TF-IDF)
tfidf_vectorizer = TfidfVectorizer(max_features=5000)
X_tfidf = tfidf_vectorizer.fit_transform(df['clean_text']).toarray
    ()  # Convert sparse to dense

```

```
# Standardize features before PCA
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_tfidf)
print(f"\nOriginal shape: {X_scaled.shape}")
# Determine best PCA variance
variance_options = [0.90, 0.95, 0.99]
best_n_components = None
best_accuracy = 0

for var in variance_options:
    pca = PCA(n_components=var)
    X_pca = pca.fit_transform(X_scaled)
    X_train, X_test, y_train, y_test = train_test_split(X_pca, df['
        airline_sentiment'].map({'positive': 1, 'neutral': 0, '
        negative': -1}), test_size=0.2, random_state=42)

    model = LogisticRegression()
    model.fit(X_train, y_train)
    acc = accuracy_score(y_test, model.predict(X_test))
    print(f"PCA Variance {var*100:.0f}% -> Accuracy: {acc:.4f}")

    if acc > best_accuracy:
        best_accuracy = acc
        best_n_components = pca.n_components_

print(f"\ n   Best PCA n_components: {best_n_components}")

# Apply best PCA
pca = PCA(n_components=best_n_components)
X_pca = pca.fit_transform(X_scaled)
X_train, X_test, y_train, y_test = train_test_split(X_pca, df['
    airline_sentiment'].map({'positive': 1, 'neutral': 0, 'negative
    ': -1}), test_size=0.2, random_state=42)

print(f"\nReduced shape: {X_pca.shape}")

# Hyperparameter Tuning
param_grid = {
    "Logistic Regression": {
        "model": LogisticRegression(),
        "params": {"C": [0.01, 0.1, 1, 10], "solver": ['liblinear',
             'lbfgs']}
    },
    "SVM": {
        "model": SVC(),
        "params": {"C": [0.1, 1, 10], "kernel": ['linear', 'rbf']}
    },
    "Random Forest": {
        "model": RandomForestClassifier(),
        "params": {"n_estimators": [50, 100, 200], "max_depth": [
            None, 10, 20]}
    }
}

best_models = {}
for name, cfg in param_grid.items():
    print(f"\  n     Tuning {name}...")
```

```
    grid_search = GridSearchCV(cfg["model"], cfg["params"], cv=3,
        scoring='accuracy', n_jobs=-1)
    grid_search.fit(X_train, y_train)
    best_models[name] = grid_search.best_estimator_

    print(f"\ n   Best {name} Model: {grid_search.best_params_}")
    print(f"Training Accuracy: {grid_search.best_estimator_.score(
        X_train, y_train)}")
    print(f"Testing Accuracy: {grid_search.best_estimator_.score(
        X_test, y_test)}")
    y_pred = best_models[name].predict(X_test)
    print(classification_report(y_test, y_pred))

# Save best parameters
best_params = {name: model.get_params() for name, model in
    best_models.items()}
pd.DataFrame(best_params).to_csv("/content/best_hyperparameters.csv
    ", index=False)
```

### 6.5.5 Assignment 2 Implementation- Grid Search

```
from sklearn.model_selection import GridSearchCV

# Define hyperparameter grid
param_grid = {
    'C': [0.1, 1, 10],
    'gamma': [1, 0.1, 0.01],
    'kernel': ['rbf']
}

grid = GridSearchCV(SVC(), param_grid, cv=5, verbose=2)
grid.fit(X_pca, y_train)

print(f"Best Parameters: {grid.best_params_}")
```

### 6.5.6 Novelty- Sarcasm Detection

```
#Sarcasm Detection Using RoBERTa
sarcasm_detector = pipeline("text-classification", model="
    cardiffnlp/twitter-roberta-base-sentiment")

def detect_sarcasm(text):
    prediction = sarcasm_detector(text)[0]
    return prediction['label'] == "LABEL_2"  # RoBERTa's "LABEL_2"
        often represents sarcasm

df['sarcasm_detected'] = df['clean_text'].apply(detect_sarcasm)

sarcasm_count = df['sarcasm_detected'].sum()
print(f"\nTotal Sarcastic Tweets Detected: {sarcasm_count}")

def predict_sentiment(tweet):
    processed_tweet = preprocess_text(tweet)
```

```
15
16      # Convert to TF-IDF representation
17      features_tfidf = tfidf_vectorizer.transform([processed_tweet]).
            toarray()  # Ensure it's a dense array
18
19      # Apply the same PCA transformation as done during training
20      features_pca = pca.transform(features_tfidf)
21
22      # Retrieve the trained Logistic Regression model
23      best_logreg = best_models.get("Logistic Regression")
24
25      if best_logreg is None:
26          raise ValueError("    Error: Best Logistic Regression model
                not found. Ensure GridSearchCV was run correctly.")
27
28      # Use the trained model for prediction
29      sentiment = best_logreg.predict(features_pca)[0]  # Use best
            trained model
30      sarcasm = detect_sarcasm(tweet)
31
32      sentiment_map = {1: "Positive", 0: "Neutral", -1: "Negative"}
33      sentiment_result = sentiment_map[sentiment]
34
35      if sarcasm:
36          sentiment_result += " (Sarcasm detected)"
37
38      return sentiment_result
39
40  # Test the model
41  test_tweet = "@airline Great job on the 5-hour delay. Best
        experience ever!"
42  print("\nSample Tweet Analysis:")
43  print(f"Tweet: {test_tweet}")
44  print(f"Predicted Sentiment: {predict_sentiment(test_tweet)}")
```

## 6.6 Output Screenshots

```
Value Count For:  airline_sentiment
negative    2135
neutral      679
positive     525
Name: count, dtype: int64
```

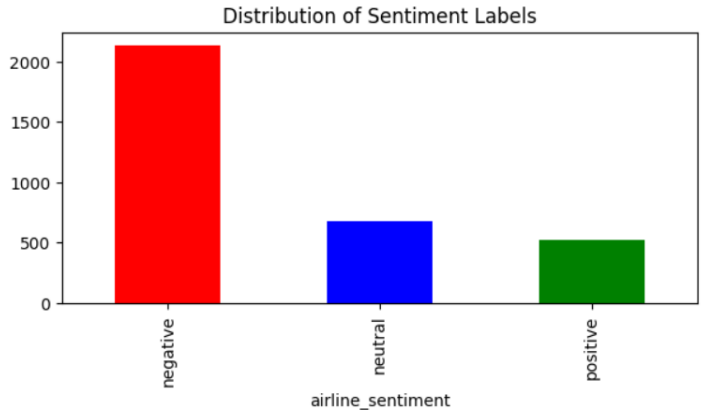



Figure 2: Distribution of Sentiment Analysis Labels

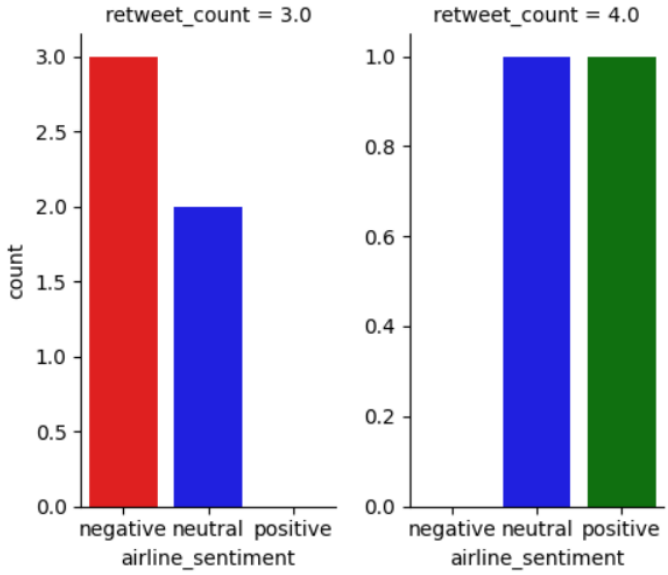



Figure 3: Retweets Count

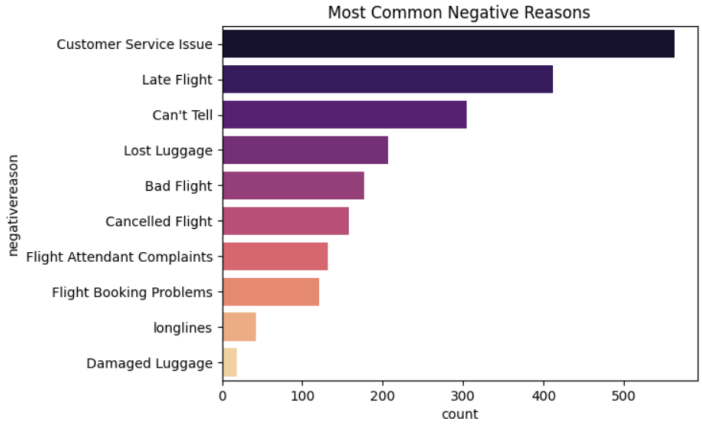



Figure 4: Most Common Negative reasons

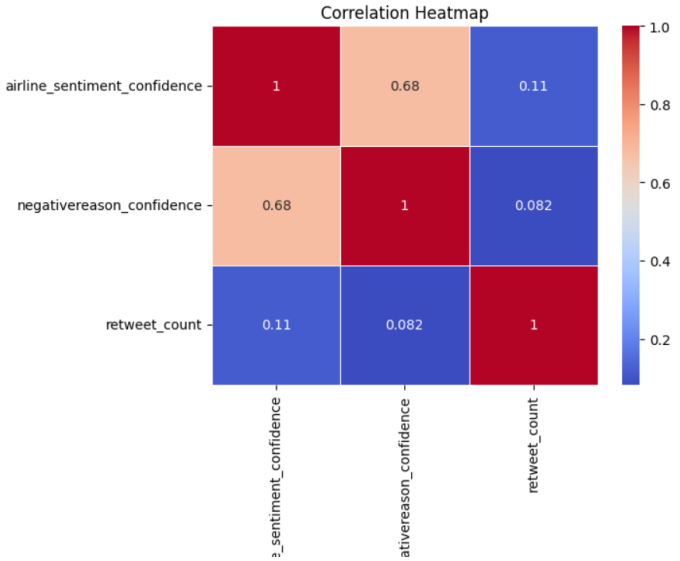



Figure 5: Corelation Heatmap

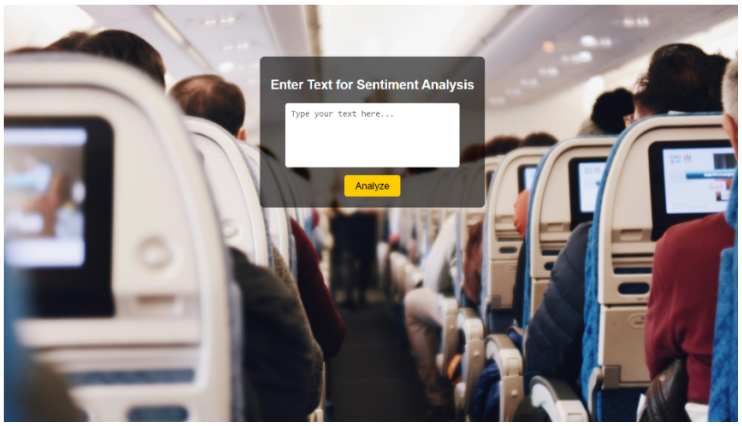



Figure 6: Flask Frontend

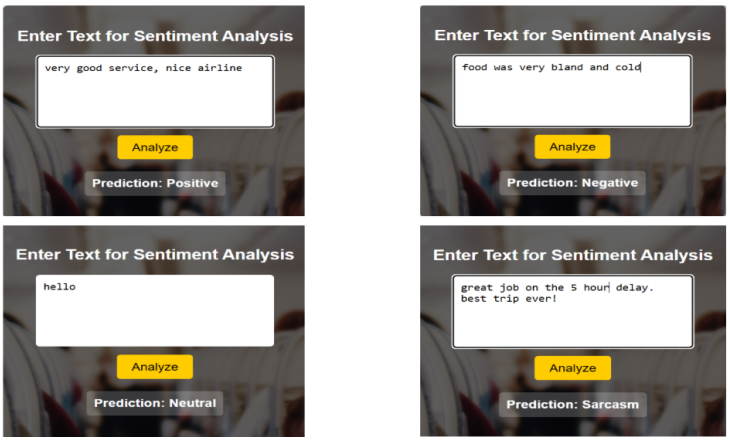



Figure 7: Sentiment Analysis Predictions

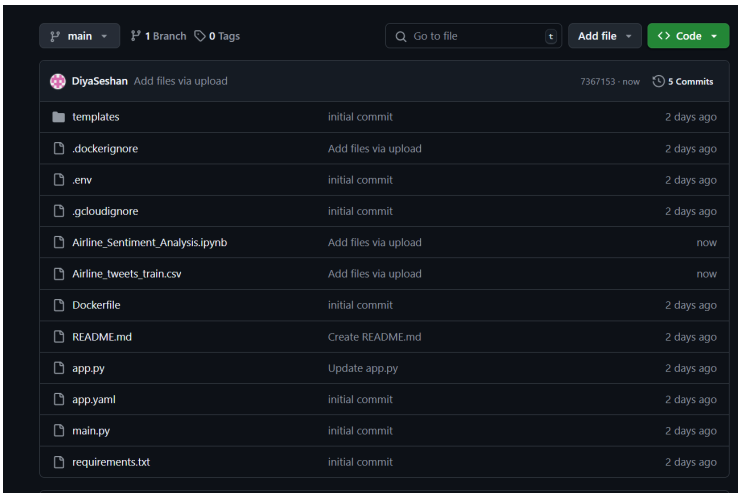



Figure 8: Git Deployment

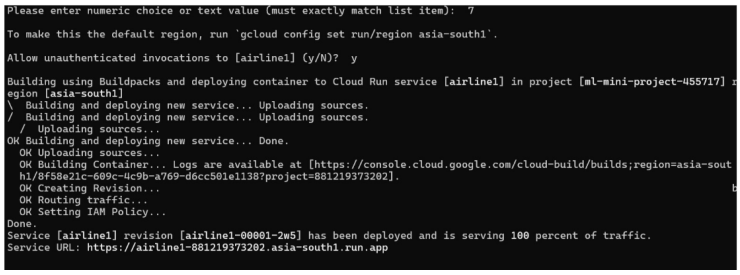



Figure 9: Google Cloud Deployment (GCP)

# 7 Results and Discussions

Our sentiment analysis model employs fundamental machine learning techniques, including SVM, Logistic Regression, and Random Forest. The models were trained with and without dimensionality reduction and hyperparameter tuning techniques to assess their impact on performance. Among them, Logistic Regression performed the best, achieving the highest accuracy of 75% when PCA and grid search were applied for dimensionality reduction and hyperparameter tuning respectively.

**RoBERTa Accuracy:** 85.98%
**DistilBERT Accuracy:** 84.66%

| Model Name | Accuracy | Precision (-1, 0, 1) | Recall (-1, 0, 1) | F1-score (-1, 0, 1) |
|---|---|---|---|---|
| Logistic Regression | 0.71 | (0.70, 0.62, 0.89) | (0.98, 0.18, 0.37) | (0.82, 0.28, 0.52) |
| SVM | 0.74 | (0.73, 0.62, 0.85) | (0.96, 0.25, 0.51) | (0.83, 0.35, 0.64) |
| Random Forest | 0.71 | (0.72, 0.56, 0.76) | (0.95, 0.22, 0.46) | (0.82, 0.31, 0.57) |

Table 3: Performance Metrics of Different Models

| Model Name | Accuracy | Precision (-1, 0, 1) | Recall (-1, 0, 1) | F1-score (-1, 0, 1) |
|---|---|---|---|---|
| Logistic Regression | 0.70 | (0.77, 0.37, 0.78) | (0.88, 0.34, 0.42) | (0.82, 0.35, 0.55) |
| SVM | 0.70 | (0.72, 0.47, 0.87) | (0.91, 0.29, 0.36) | (0.81, 0.36, 0.51) |
| Random Forest | 0.68 | (0.68, 0.47, 0.67) | (0.99, 0.07, 0.13) | (0.81, 0.13, 0.21) |

Table 4: Performance Metrics of Different Models with Dimensionality Reduction

## 7.1 Model Performance Analysis

- **Lack of Contextual Understanding**: These models treat words as isolated tokens, failing to capture the relationships between them.
- **Difficulty with Long and Complex Sentences**: Without structural comprehension, the models struggle with nuanced sentiments.
- **Inability to Handle Synonyms Effectively**: Similar sentiments expressed with different words may be classified inconsistently due to the lack of semantic understanding.
- **Issues with Negation, Word Order, and Sarcasm**: These models often misinterpret sentences where meaning depends on negation (e.g., *"not bad"*), sarcasm, or word positioning.

These factors contribute to the observed accuracy of 75%, indicating that more advanced techniques, such as deep learning-based transformers, may be required for further improvement.

## 7.2 Impact of the Project on Human, Societal, Ethical, and Sustainable Development

- **Human Impact:** Helps airlines improve customer service by addressing issues promptly.
- **Societal Impact:** Enhances passenger experience by providing real-time issue resolution.
- **Ethical Considerations:** Ensures unbiased and fair sentiment analysis by handling data responsibly.
- **Sustainability:** Reduces manual efforts, enabling efficient resource allocation for better airline services.

| Model Name | Accuracy | Precision (-1, 0, 1) | Recall (-1, 0, 1) | F1-score (-1, 0, 1) |
|---|---|---|---|---|
| Logistic Regression | 0.75 | (0.78, 0.52, 0.84) | (0.93, 0.35, 0.53) | (0.85, 0.42, 0.65) |
| SVM | 0.74 | (0.75, 0.55, 0.87) | (0.95, 0.23, 0.52) | (0.84, 0.32, 0.65) |
| Random Forest | 0.74 | (0.75, 0.54, 0.78) | (0.95, 0.20, 0.50) | (0.84, 0.29, 0.61) |

Table 5: Performance Metrics of Different Models with Dimensionality Reduction and Hyperparameter Tuning

| Model Name | Best Parameters |
|---|---|
| Logistic Regression | Solver: 'liblinear', C: 10 |
| SVM | Kernel: 'linear', C: 1.0 |
| Random Forest | Number of Estimators: 100, Max Depth: None |

Table 6: Best Parameters for Each Model

# 8 Conclusion and Future Work

This study successfully implemented a sentiment analysis system for U.S. airline tweets using machine learning techniques, including Logistic Regression, Support Vector Machines (SVM), and Random Forest. Various techniques, such as feature selection, dimensionality reduction using Principal Component Analysis (PCA), and hyperparameter tuning, were explored to enhance model performance.

Our experiments revealed that while hyperparameter tuning improved the predictive capabilities of certain models, PCA did not prove to be a very beneficial approach. Instead of improving accuracy, PCA led to a decline in performance, indicating that reducing dimensionality may have resulted in the loss of crucial information. This suggests that for sentiment analysis tasks, where textual features play a significant role, preserving the original feature space is more effective than compressing it.

To address sarcasm detection, RoBERTa, a transformer-based deep learning model, was leveraged, significantly improving the ability to recognize sarcastic remarks within the dataset. This addition demonstrated the importance of contextual embeddings for understanding sentiment beyond surface-level text analysis.

The study also highlighted key challenges faced by traditional machine learning models in sentiment analysis, including their difficulty in handling sarcasm, negations, complex sentence structures, and context-dependent sentiments. Despite achieving a competitive accuracy, these limitations emphasize the need for more advanced techniques such as deep learning-based models that can capture contextual meaning more effectively.

Despite achieving a competitive accuracy, the study highlights the need for more advanced techniques, such as transformer-based models (e.g., BERT), which can provide better semantic understanding. The insights gained from this study provide a strong foundation for future enhancements in sentiment analysis systems.

### 8.1 Future Work

Several avenues for improvement and expansion remain for this sentiment analysis framework:

- **Multilingual sentiment analysis:** Expanding the system to analyze tweets in multiple languages can increase its applicability across global airlines.
- **Real-time sentiment analysis:** Extending the system to work in a real-time setting, where tweets are analyzed as they are posted, can help airlines respond to customer feedback more efficiently.
- **Integration with airline feedback systems:** Deploying the model within airline customer support systems can provide automated insights and improve customer experience.
- **Multimodal Sentiment Analysis:** Expanding the system to analyze not just text but also images, videos, and emojis in tweets can provide a more holistic understanding of customer sentiment.
- **Cross-Domain Adaptability:** Extending the sentiment analysis model to other industries like hospitality, retail, and public services can validate its robustness and generalizability.

By addressing these areas, future iterations of this sentiment analysis system can provide more accurate, context-aware, and actionable insights, thereby contributing to improved airline customer experience and operational efficiency.

## References

[1] Adoma, A. F., Henry, N.-M., and Chen, W. (2020). Comparative analyses of bert, roberta, distilbert, and xlnet for text-based emotion recognition. In *2020 17th international computer conference on wavelet active media technology and information processing (ICCWAMTIP)*, pages 117–121. IEEE.

[2] Krasitskii, M., Kolesnikova, O., Hernandez, L. C., Sidorov, G., and Gelbukh, A. (2025). Comparative approaches to sentiment analysis using datasets in major european and arabic languages. In *Artificial Intelligence and Big Data Trends 2025*, AIBD, page 137–150. Academy Industry Research Collaboration Center.

[3] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

[4] Srinivas, A., Satyanarayana, C., Divakar, C., and Sirisha, K. (2021). Sentiment analysis using neural network and lstm. *IOP Conference Series: Materials Science and Engineering*, 1074:012007.

[5] Sugitomo, J. C., Kevin, N., Jannatri, N., and Suhartono, D. (2021). Sentiment analysis using svm and naïve bayes classifiers on restaurant review dataset. In *2021 1st International Conference on Computer Science and Artificial Intelligence (ICCSAI)*, volume 1, pages 100–108.

[6] Tan, K. L., Lee, C. P., Anbananthen, K. S. M., and Lim, K. M. (2022). Roberta-lstm: a hybrid model for sentiment analysis with transformer and recurrent neural network. *IEEE Access*, 10:21517–21525.

[7] Zupan, J. (1994). Introduction to artificial neural network (ann) methods: what they are and how to use them. *Acta Chimica Slovenica*, 41(3):327.

## Assets

[1] GitHub Repository. *Airline Sentiment Analysis Project*. Available at: https://github.com/DiyaSeshan/Airline-Tweets-Sentiment-Analysis

[2] Dataset Source. *Airline Twitter Sentiment Dataset*. Available at: https://www.kaggle.com/datasets/tango911/airline-sentiment-tweets/data

[3] Google Colab Notebook. *Model Training and Inference*. Available at: https://colab.research.google.com/drive/1hQhv5dfYXD2oFyN24fKlEZZI8w63Zvo_?usp=sharing