Project Report: NLP Engineer Assignment

1. Problem Statement

The goal of this project is to develop an NLP model capable of identifying emotions in tweets directed toward specific products. Key objectives include:

- **Exploratory Data Analysis (EDA)** to understand data patterns.
- **Dataset Enhancement** for improved model training.
- **Data Augmentation** to enrich the dataset and generalize the model.
- **Model Fine-Tuning and Evaluation** using a transformer architecture.
- **Deployment** in a production-ready environment with post-deployment monitoring metrics.

This assignment emphasizes the approach, solution quality, and deployment readiness rather than raw model performance.

2. Solution Approach

The solution consists of the following stages:

2.1 Data Preprocessing and Exploration

- **Initial Exploration**: Assessed the dataset structure and identified missing values.
- **EDA**: Analyzed patterns in emotion distribution, tweet lengths, and word frequency.

2.2 Dataset Enhancement

- **Class Balancing**: Used oversampling for underrepresented classes to ensure a balanced distribution of emotions.
- **Duplicate Removal**: Removed duplicates and noisy entries to maintain data quality.

2.3 Data Augmentation

- **Synonym Replacement**: Introduced variation by replacing words with synonyms.
- **Back Translation**: Added linguistic diversity by translating tweets to another language and back.
- **Random Insertion**: Randomly inserted synonyms to introduce new sentence structures.

2.4 Model Training

- **Model Selection**: Chose `bert-base-uncased` for its effectiveness in text classification.
- **Fine-Tuning**: Optimized the model on the balanced and augmented dataset, focusing on emotion and product classification.

2.5 Model Evaluation

- Evaluated performance using precision, recall, and F1 score, emphasizing both per-class and overall metrics.

2.6 Deployment

- **Flask API**: Created a `/predict` endpoint to serve model predictions.
- **Deployment Configuration**: Provided Docker and AWS Elastic Beanstalk setup instructions.

2.7 Post-Deployment Monitoring

Defined key metrics (latency, throughput, error rate, model drift, and resource usage) to ensure the model's reliability and performance in production.

3. Code Explanation

The code implementation is divided as follows:

3.1 Data Loading and Exploration

- Loaded data from `Train.csv` and `Test.csv`, checked for null values, and inspected initial records for format and content validation.

3.2 Exploratory Data Analysis (EDA)

- **Class Distribution**: Visualized emotion distribution to assess class imbalance.
- **Text Length Analysis**: Examined tweet length patterns to identify preprocessing needs.
- **Word Cloud**: Created a word cloud to observe common words associated with each emotion.

3.3 Dataset Improvement

- **Oversampling**: Balanced classes through oversampling of minority classes.
- **Duplicate Removal**: Removed redundant tweets to improve data quality.

3.4 Data Augmentation

- **Synonym Replacement**: Replaced words with synonyms to increase variability.
- **Back Translation**: Translated tweets to a secondary language and back to enhance diversity.
- **Random Insertion**: Added synonyms at random positions to generate new sentence structures.

3.5 Text Cleaning

- Removed URLs, mentions, and special characters.
- Converted text to lowercase to maintain consistency.

3.6 Model Training

- Loaded `bert-base-uncased` transformer model.
- Tokenized and fine-tuned the model on the processed dataset, adjusting for optimal batch size, learning rate, and number of epochs.

3.7 Model Evaluation

- Used `classification_report` to calculate precision, recall, and F1 scores per class and overall.
- Produced a confusion matrix to evaluate performance across different emotions and targets.

3.8 Deployment with Flask API

- Implemented a Flask API with a `/predict` endpoint for model inference.
- Configured Docker and AWS Elastic Beanstalk for scalable deployment.

4. Dataset Enhancement Suggestions

- **Class Balancing**: Utilize oversampling for minority classes to achieve balanced training.
- **Duplicate and Noise Removal**: Remove duplicate tweets and filter out noisy data, such as tweets with only emojis or hashtags.
- **Data Augmentation Techniques**: Enhance data variety through synonym replacement, back translation, and random insertion.
- **External Datasets**: Integrate additional emotion or sentiment datasets with similar labels for expanded training.

5. Deployment Requirements

5.1 Environment Setup

Deploy the model using Docker for a consistent production environment:

Dockerfile:

```
# Use an official Python runtime as a parent image
FROM python:3.8-slim

# Set the working directory in the container
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Install dependencies
RUN pip install --no-cache-dir -r requirements.txt

# Expose the port that Flask will run on
EXPOSE 5000

# Run the Flask application
CMD ["python", "app.py"]
```

Requirements.txt:

```
transformers
torch
Flask
pandas
matplotlib
seaborn
scikit-learn
wordcloud
textblob
googletrans==4.0.0-rc1
```

5.2 Deployment on AWS Elastic Beanstalk

Steps for deployment on AWS Elastic Beanstalk:

1.Initialize EB:

```
eb init -p docker flask-sentiment-api
```

2.Create an Environment:

```
eb create flask-sentiment-env
```

3.Deploy Application:

```
eb deploy
```

4.Open the Application:

```
eb open
```

6. Post-Deployment Monitoring Metrics

To ensure the model's performance and reliability post-deployment, monitor the following metrics:

- Latency: Measures request processing time. Low latency is crucial for user experience.
- Throughput: Tracks the number of requests over time, providing insight into API usage and scaling needs.
- Error Rate: Monitors the percentage of failed requests. High error rates may indicate API or model issues.
- Model Drift: Regularly check model accuracy on a sample of recent data to identify data drift or performance degradation.
- Resource Usage: Monitor CPU, memory, and GPU usage for resource optimization and server scaling.

7. Conclusion

This project successfully develops, evaluates, and deploys an NLP model to detect emotions directed toward specific products in tweets. Key accomplishments include:

- Conducting EDA and dataset enhancement to optimize data quality.
- Fine-tuning a transformer-based model for multi-class classification.
- Deploying a Flask API on AWS Elastic Beanstalk with Docker, making the solution scalable and reliable.
- Establishing monitoring metrics for post-deployment to ensure ongoing performance and usability.

This end-to-end approach enables the model to deliver actionable insights in real-world environments, with mechanisms for continuous monitoring and improvement.