# **📊 Comprehensive Project Report**

## **📌 Project Title**

**Sentiment Analysis Model - Deploy using AWS Services or Hugging Face**

## **🎯 Objective**

To deploy a fine-tuned **DistilBERT-based Sentiment Analysis model** using **AWS services**, making it accessible to users via a web application built with **Streamlit** or **Gradio**.

## **🧩 Problem Statement**

The goal is to perform **entity-level sentiment analysis** on tweets. Given a tweet and an associated entity, the task involves determining whether the sentiment expressed is **Positive**, **Negative**, or **Neutral**.

## **📁 Dataset Overview**

* **Source**: Twitter entity-level sentiment analysis dataset
* **Link**: [Dataset](https://raw.githubusercontent.com/GuviMentor88/Training-Datasets/refs/heads/main/twitter_training.csv)
* **Classes**: Positive, Negative, Neutral
* **Features**:
  + **Tweet ID**: Unique identifier (optional).
  + **Entity**: Topic of the tweet.
  + **Sentiment**: Sentiment label (Positive/Negative/Neutral).
  + **Tweet Content**: Raw tweet text.

## **🔧 Technical Approach**

### **1. Data Preparation**

* Preprocessed the dataset.
* Fine-tuned the **DistilBERT-base-uncased** model using this data.

### **2. Infrastructure Setup**

* **AWS Services Used**:
  + **Amazon S3**: To store the trained model and app.py.
  + **Amazon EC2**: To host the application.
  + **Amazon RDS**: To log user interactions (text, sentiment, IP address).

### **3. Model Deployment**

* **Steps**:
  + Downloaded the model from **S3** into the EC2 instance.
  + Loaded the model using **Hugging Face Transformers**.
  + Created a sentiment analysis web application using **Gradio**.

### **4. Application Deployment**

* Deployed app.py on an EC2 instance.
* Configured EC2 security groups to allow traffic on ports **8501** (Streamlit/Gradio) and **3306** (RDS).

### **5. Logging User Data**

* Configured **Amazon RDS**:
  + Stored user details (input text, sentiment predictions, and IP address) in the sentiment\_logs table.

## **📈 Evaluation Metrics**

* **Accuracy**: How often predictions match true labels.
* **Precision**: Correctly predicted positive observations vs total predicted positive observations.
* **Recall**: Correctly predicted positive observations vs total actual positives.
* **F1-Score**: Harmonic mean of precision and recall.
* **Latency**: Time taken for predictions.

## **🖥️ Application Features**

* **User Interface**: Simple and intuitive Gradio web app.
* **Functionalities**:
  + Accepts user input.
  + Predicts sentiment and provides probabilities for Positive, Neutral, and Negative sentiments.
  + Logs user details securely in RDS.

## **🛠️ Project Implementation Code**

### **Key Components:**

* **AWS S3 Integration**: Downloads model files.
* **Model Inference**: Uses Hugging Face Transformers for predictions.
* **Gradio Interface**: Provides user-friendly web application.
* **RDS Logging**: Logs user input and predictions securely.

## **🔒 Security Measures**

* IAM roles with **AWS S3 Full Access** and restricted RDS access.
* Security groups to limit EC2 access to specific ports.

## **📜 Results & Deliverables**

### **Outcomes**

* Fully functional web application.
* Scalable deployment framework using **AWS services**.

### **Deliverables**

1. **Source Code**:
   * Fine-tuned model and app.py.
2. **Documentation**:
   * Project setup, deployment, and usage instructions.

## **🚀 Future Enhancements**

* **Scalability**: Use **Elastic Load Balancer (ELB)** for better handling of concurrent users.
* **Performance**: Optimize latency with AWS Lambda or containerization using Docker.
* **UI Improvements**: Enhance the Gradio interface for better user experience.

## **📌 Conclusion**

This project demonstrates the seamless deployment of an NLP model on AWS, leveraging the power of **Hugging Face Transformers** and **Gradio** for real-world applications. The pipeline ensures high availability, scalability, and security, providing users with an efficient and user-friendly sentiment analysis tool.