## Lab 6-01: Responsible AI for Text Classification with Bias Detection and Explainability

|  |
| --- |
| **Introduction:**  This lab introduces the concept of Responsible AI by showing how to build a simple text classifier, evaluate it for fairness, and explain its predictions. Using the SMS Spam Collection dataset, a Naive Bayes model is trained to distinguish between spam and non-spam messages. Beyond accuracy, the lab highlights bias detection by comparing performance on short versus long messages and applies LIME to explain which words most influence predictions. This demonstrates how fairness checks, interpretability, and governance practices can be integrated into a machine learning workflow.  **Problem Scenario:**  An organization plans to use AI for automating spam detection in incoming text messages, but is concerned about risks such as model bias, lack of transparency, and governance gaps. The leadership team needs assurance that the model performs consistently across different message types, provides explanations that users can understand, and generates outputs that can be documented for compliance. To address these concerns, this lab demonstrates how bias monitoring, explainability, and reporting practices can be applied in practice to ensure the classifier remains ethical, transparent, and trustworthy.  **Solution:**  **Step 1: Set up Environment**   1. Install Python (>=3.9).      1. Install required libraries:   **pip install scikit-learn pandas matplotlib seaborn lime**    **Step 2: Prepare Dataset**   1. Use a simple open dataset (e.g., SMS Spam Collection). Download from: [SMS Spam Dataset UCI](https://archive.ics.uci.edu/ml/datasets/sms+spam+collection?utm_source=chatgpt.com).     **Step 3: Create Python Script**   1. In your Generative AI Leader Lab folder, create a new file: responsible\_ai\_lab.py      1. Copy-paste the full script below into that file:   **# responsible\_ai\_lab.py**  **# Lab: Responsible AI Text Classifier with Bias and Explainability**  **import pandas as pd**  **from sklearn.model\_selection import train\_test\_split**  **from sklearn.feature\_extraction.text import CountVectorizer**  **from sklearn.naive\_bayes import MultinomialNB**  **from sklearn.metrics import classification\_report**  **# =============================**  **# Step 1: Load Dataset**  **# =============================**  **print("Loading dataset...")**  **data = pd.read\_csv("SMSSpamCollection", sep="\t", names=["label", "text"])**  **print(data.head())**  **# =============================**  **# Step 2: Train-Test Split**  **# =============================**  **X\_train, X\_test, y\_train, y\_test = train\_test\_split(**  **data["text"], data["label"], test\_size=0.2, random\_state=42**  **)**  **# =============================**  **# Step 3: Vectorize Text**  **# =============================**  **vectorizer = CountVectorizer()**  **X\_train\_vec = vectorizer.fit\_transform(X\_train)**  **X\_test\_vec = vectorizer.transform(X\_test)**  **# =============================**  **# Step 4: Train Model**  **# =============================**  **model = MultinomialNB()**  **model.fit(X\_train\_vec, y\_train)**  **# =============================**  **# Step 5: Evaluate Model**  **# =============================**  **y\_pred = model.predict(X\_test\_vec)**  **print("\nClassification Report:\n")**  **print(classification\_report(y\_test, y\_pred))**  **# =============================**  **# Step 6: Bias Detection**  **# =============================**  **data["length"] = data["text"].apply(len)**  **short\_msgs = data[data["length"] < 50]**  **long\_msgs = data[data["length"] >= 50]**  **short\_acc = model.score(vectorizer.transform(short\_msgs["text"]), short\_msgs["label"])**  **long\_acc = model.score(vectorizer.transform(long\_msgs["text"]), long\_msgs["label"])**  **print("\nBias Detection Results:")**  **print(f"Short Messages Accuracy: {short\_acc:.2f}")**  **print(f"Long Messages Accuracy: {long\_acc:.2f}")**  **# =============================**  **# Step 7: Explainability with LIME**  **# =============================**  **from lime.lime\_text import LimeTextExplainer**  **class\_names = ["ham", "spam"]**  **explainer = LimeTextExplainer(class\_names=class\_names)**  **# Pick an example from test set**  **i = 10**  **sample\_text = X\_test.iloc[i]**  **print(f"\nExplaining prediction for message:\n{sample\_text}\n")**  **exp = explainer.explain\_instance(**  **sample\_text,**  **model.predict\_proba,**  **num\_features=6,**  **labels=[0, 1],**  **vectorizer=vectorizer**  **)**  **# Show explanation in console**  **print("Top features influencing prediction:")**  **for feature, weight in exp.as\_list(label=1):**  **print(f"{feature} => {weight:.3f}")**  **# =============================**  **# Step 8: Governance Notes**  **# =============================**  **print("\nGovernance Notes:")**  **print("- Record overall accuracy")**  **print("- Note subgroup accuracy (short vs long messages)")**  **print("- Save explainability results")**  **print("- Suggest mitigation like retraining with balanced dataset")**    **Step 4: Run the Lab**   1. Open a terminal in your project folder and run the script:   **python responsible\_ai\_lab.py**   1. You should see:    * First few rows of the dataset    * Classification report (precision, recall, F1-score)    * Bias detection (accuracy for short vs long messages)    * Example explanation from LIME (which words influenced spam vs ham)       **Step 5: Governance and Reporting**   1. Save outputs into a text file:   **python responsible\_ai\_lab.py > lab\_results.txt** |