OST Experiential Learning Assignment

13th March 2025

Name: Aparna Iyer

PRN: 22070126017

Branch: AI-ML A1

Batch: 2022-2026

ML Model using Shell Scripting:

1. Step 1: touch mlmodel.sh
2. Step 2: nano mlmodel.sh

```
aparna@aparna-VMware-Virtual-Platform:~/Documents$ nano mlmodel.sh
```

3. Step 3: Add Bash Code to the opened Shell script file (mlmodel.sh).
4. Step 4: Save the File and Exit.
5. Step 5: Grant 'write' access to mlmodel.sh using:
   chmod +x mlmodel.sh
6. Step 6: Execute the file.

   ./mlmodel.sh

Bash Script for mlmodel.sh:

```
  GNU nano 7.2
#!/bin/bash

# Directories
DATA_DIR="C:\Users\Aparna Iyer\Documents\IRIS.csv"
MODEL_DIR="C:\Users\Aparna Iyer\Documents\iris_classification_model.pkl"
LOG_FILE="C:\Users\Aparna Iyer\Documents\training_log.txt"

# Model versioning
TIMESTAMP=$(date +"%Y%m%d%H%M%S")
NEW_MODEL_PATH="$MODEL_DIR/model_$TIMESTAMP.pkl"

# Function to train the model
train_model() {
    echo "[INFO] Training new model..."
    python train_model.py --data_dir "$DATA_DIR" --output "$NEW_MODEL_PATH"
}

# Function to evaluate the model
evaluate_model() {
    echo "[INFO] Evaluating model..."
    NEW_MODEL_SCORE=$(python evaluate_model.py --model "$NEW_MODEL_PATH")
    DEPLOYED_MODEL_SCORE=$(python evaluate_model.py --model "$MODEL_DIR/model.pkl")

    echo "New Model Score: $NEW_MODEL_SCORE"
    echo "Deployed Model Score: $DEPLOYED_MODEL_SCORE"

    if (( $(echo "$NEW_MODEL_SCORE > $DEPLOYED_MODEL_SCORE" | bc -l) )); then
        return 0 # New model is better
    else
        return 1 # New model is not better
    fi
}
```

```
# Function to deploy the model
deploy_model() {
    echo "[INFO] Deploying new model..."
    cp "$NEW_MODEL_PATH" "$DEPLOY_DIR/model.pkl"
    echo "[INFO] Deployment successful."
}

# Check for new data
if [ "$(ls -A $DATA_DIR)" ]; then
    echo "[INFO] New data found. Starting training pipeline..." | tee -a "$LOG_FILE"
    train_model
    evaluate_model
    if [ $? -eq 0 ]; then
        deploy_model
        echo "[INFO] Model deployed successfully at $TIMESTAMP" | tee -a "$LOG_FILE"
    else
        echo "[INFO] New model did not outperform the deployed model. Skipping deployment." | tee -a "$LOG_FILE"
    fi
else
    echo "[INFO] No new data found. Exiting..." | tee -a "$LOG_FILE"
fi
```

Code Screenshots for Supporting Files:

1. IRIS.csv:

	A	B	C	D	E
1	sepal_leng	sepal_widt	petal_leng	petal_widt	species
2	5.1	3.5	1.4	0.2	Iris-setosa
3	4.9	3	1.4	0.2	Iris-setosa
4	4.7	3.2	1.3	0.2	Iris-setosa
5	4.6	3.1	1.5	0.2	Iris-setosa
6	5	3.6	1.4	0.2	Iris-setosa
7	5.4	3.9	1.7	0.4	Iris-setosa
8	4.6	3.4	1.4	0.3	Iris-setosa
9	5	3.4	1.5	0.2	Iris-setosa
10	4.4	2.9	1.4	0.2	Iris-setosa
11	4.9	3.1	1.5	0.1	Iris-setosa
12	5.4	3.7	1.5	0.2	Iris-setosa
13	4.8	3.4	1.6	0.2	Iris-setosa
14	4.8	3	1.4	0.1	Iris-setosa
15	4.3	3	1.1	0.1	Iris-setosa
16	5.8	4	1.2	0.2	Iris-setosa
17	5.7	4.4	1.5	0.4	Iris-setosa
18	5.4	3.9	1.3	0.4	Iris-setosa
19	5.1	3.5	1.4	0.3	Iris-setosa
20	5.7	3.8	1.7	0.3	Iris-setosa
21	5.1	3.8	1.5	0.3	Iris-setosa
22	5.4	3.4	1.7	0.2	Iris-setosa
23	5.1	3.7	1.5	0.4	Iris-setosa
24	4.6	3.6	1	0.2	Iris-setosa
25	5.1	3.3	1.7	0.5	Iris-setosa
26	4.8	3.4	1.9	0.2	Iris-setosa

2. Iris_classification_model.pkl:

Created from Python code run on Google Collab.

```
[1]  import pickle
     import pandas as pd
     from sklearn.ensemble import RandomForestClassifier
     from sklearn.model_selection import train_test_split
```

```
[4]  # Load sample data
     data = pd.read_csv("/content/IRIS.csv") #Iris Dataset

     # Split data into features (X) and target (y)
     X = data.drop(columns=["species"]) # Features

     y = data["species"]  # Target

     # Split into training and testing sets
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

     # Train a simple RandomForest model
     model = RandomForestClassifier(n_estimators=100, random_state=42)
     model.fit(X_train, y_train)
```

```
RandomForestClassifier(random_state=42)
```

```
[13]
# Define the path to save the model within Google Drive
model_path = "/content/drive/My Drive/iris_classification_model.pkl"

# Save the trained model
with open(model_path, "wb") as file: # Changed to "wb" to write in binary mode
    pickle.dump(model, file)

# Load the trained model
with open(model_path, "rb") as file:
    model = pickle.load(file)
```

3. Log File: training_log.txt

```
[2025-03-13 09:30:15] INFO: New data found. Training started.
[2025-03-13 09:31:00] INFO: Model trained successfully.
[2025-03-13 09:31:05] INFO: Model evaluation score: 0.89
[2025-03-13 09:31:10] INFO: Model deployed successfully.
```