AML Milestone 1

Group 1: Aaditya Gupta and Harjeet Singh Yadav

Experimentation on 0 noise:

a) Separated ‘era’, ‘target_5’, and ‘target_10’ columns from the dataset
b) Digitized the ‘era’ columns to create a new label column from 0 to 11.
c) Trained a sequential MLP to train:

```
# Splitting dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, Y_new, test_size=0.2, random_state=42)

# Define the MLP model
model = Sequential([
    Dense(128, input_shape=(26,), activation='relu'),
    Dense(64, activation='relu'),
    Dense(12, activation='softmax')  # Output layer with softmax activation for multiclass classification
])

# Compile the model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

d)
e) Zero noise loss and accuracy :

```
Test Loss: [0.3394385278224945, 0.8814102411270142]
```

Experimentation on feature noise (taking ‘eras’ as the label)

1) **MLPclassifier**
   hyperparameters (hidden layers = (200, 150, 50, 25))
   Epoch = 10,000
   Learning rate = 0.01
   activation = relu and Softmax at the last layer
   - Zero noise:- training accuracy = 0.84 and testing accuracy = 0.85
   - Low Noise:- training and testing both = 0.99 for 5000 epochs (Grokking observed)
   - High Noise:- training accuracy = 0.58 and testing accuracy = 0.57
2) We applied **L1 and L2 regularization** with MLP classification and saw some improvements in the accuracy of test data (1 or 2 %).
3) **Ensemble Learning (RandomForestClassifier, Gradient Boosting classifier)**
   - Zero noise:- Training accuracy = 0.99, test accuracy = 0.96
   - Low Noise:- Training accuracy = 0.99, test accuracy = 0.83
   - High Noise:- Training accuracy = 0.1, test accuracy = 0.6
4) MLP classification with noise attention
   - Low Noise:- Training accuracy = 0.6 and testing accuracy = 0. 3
   - High Noise:- Training accuracy = 0.5 and testing accuracy = 0.25

Experimentation using feature and label noise, (predict 'target_10_val,"):
Exp 1:
We tried the following method on the assumption that NOT ALL labels are noisy:

1. Assumed each data point (without the label) as a vector
2. Found cosine similarity of every vector with every other vector.
3. Assigned to each vector that label, which vectors most similar to it have: basically "Majority Voting" but to create new labels.
4. Then, using these "new_labels," the MLP was trained and tested on the dataset modified similarly.

Observations:

1. This operation was very costly, hence only applied this on "random" 20000 train and "random" 20000 test data points.
2. Almost every data point was assigned the label "1" out of the four possible values.
3. Training and testing hence could not give accurate results.

Exp 2:
Using Autoencoders on high noise and low noise data:

1. Obtained embeddings for high-noise data by feeding into an autoencoder:

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 26)]              0

 dense (Dense)               (None, 256)               6912

 dense_1 (Dense)             (None, 64)                16448

 dense_2 (Dense)             (None, 5)                 325

 dense_3 (Dense)             (None, 64)                384

 dense_4 (Dense)             (None, 256)               16640

 dense_5 (Dense)             (None, 26)                6682

_________________________________________________________________
```

2.
3. These embeddings were then used as input to the MLP
4. Train accuracy = 26% and Test accuracy = 25.8%