Comparative Study: MLP vs. LSTM for Crop Yield Forecasting

This report focuses on building two data-driven deep learning models—a Panel Long Short-Term Memory (LSTM) and a Multilayer Perceptron (MLP)—to forecast annual crop yields by country and crop type. Both models utilize historical agro-climatic, land cover, and production data to capture spatial and temporal yield patterns.

The objective is to evaluate and compare these architectures in their ability to predict crop yields for the year 2023. The report outlines the full pipeline: from data integration and preprocessing to model design, training, evaluation, and forecasting.

A. Performance

Multilayer Perceptron (MLP) Model: The MLP model demonstrated strong performance metrics after 100 epochs of training. Its training Mean Squared Error (MSE) was 0.1237, with a corresponding validation MSE of 0.1584. In terms of R-squared (R^2), which assesses the goodness of fit, the MLP achieved a final training R^2 score of 0.8726 and a validation R^2 score of 0.8582. These metrics were computed at each epoch using the entire training or validation sets, respectively. The total number of data instances used for the MLP originated from the 'df' DataFrame (a copy of 'merged_cols' created after all data source merging and before the MLP's train/test split), corresponding to (67963, 31) instances as per the notebook's processing steps. Principle component analysis(PCA) removed insignificant columns reducing the dataset size to (67963, 16). For training and evaluation, this data was divided, allocating 80% to the training set and 20% to the validation set, using Scikit-learn's train_test_split function with a random_state of 42 to ensure reproducibility.

Long Short-Term Memory (LSTM) Model: The LSTM model, after 100 epochs, yielded a training MSE of 0.4931 and a validation MSE of 0.4960. The R^2 scores were 0.5088 for the training set and 0.4978 for the validation set. Similar to the MLP, these metrics were calculated at each epoch. The LSTM model was trained on 51,613 sequences, which was the first dimension of the input tensor X (shape (51613, 3, 32)) generated by the 'make_panel_sequences_default' function. These sequences were then split, with 80% used for training and 20% for validation, employing train_test_split with random_state=42 and shuffle=True.

Comparative Performance Notes: A direct comparison of the validation R^2 scores reveals that the MLP model, achieving 0.8582, significantly outperformed the LSTM model, which scored 0.4978. This indicates that, for this specific dataset and the configurations implemented in the notebook, the MLP model demonstrated a superior ability to generalize to unseen data and provide a better fit for the crop yield forecasting task.

B. Model

Multilayer Perceptron (MLP) Model: The MLP model architecture is designed for structured, tabular data. It begins by utilizing embedding layers to handle the categorical features of 'country' and 'item'. Earlier LabelEncoding and OneHotEncoding was explored but the model wasn't converging to a suitable loss value. These learned embeddings, each of 8 dimensions, are then concatenated with the numerical features, which have previously undergone Principal Component Analysis (PCA). This combined feature vector forms the input to a sequence of two dense hidden layers. The first hidden layer consists of 128 units, followed by a ReLU activation function and a Dropout layer with a rate of 0.3. The second hidden layer has 64 units, also followed by ReLU activation and 0.3 Dropout. Finally, a single linear output unit predicts the scaled yield value. Key hyperparameters include the Adam optimizer with a learning rate of 0.001, MSELoss as the loss function, a batch size of 64, and training over 100 epochs. To mitigate overfitting, the model incorporates Dropout layers, monitors performance on a validation set, utilizes PCA for dimensionality reduction of input features, and employs batch processing during training. The input provided here were the average of the features from 2020 to 2022 followed bythe model architecture to produce 2023 yield value.

Long Short-Term Memory (LSTM) Model: The LSTM model, implemented as PanelLSTM in the notebook, is tailored for sequential data. Its core is a 2-layer stacked LSTM with a hidden state size of 64 units and an internal dropout rate of 0.2 applied between LSTM layers. This LSTM network processes input sequences, where each sequence has a length of 3 time steps (seq_len=3) and each time step comprises 32 features. The batch_first=True parameter ensures the batch dimension comes first in the input tensor. The output from the final time step of the LSTM layer is then passed to a fully connected linear layer (self.fc), which maps the LSTM's hidden state to a single output value representing the predicted scaled yield. For final forecasting of crop for 2023 a window from 2020 to 2022 does the job.

Conceptually, while LSTMs are distinct from simple MLPs due to their recurrent nature and memory capabilities, they are built upon similar foundational principles. The internal computations of an LSTM cell, particularly for its gating mechanisms (input, forget, and output gates), involve operations akin to those in MLP neurons: inputs are multiplied by learned weights, summed, and passed through non-linear activation functions (like sigmoid or tanh). Each gate can be seen as a small, specialized neural network learning to regulate information flow. Furthermore, if an LSTM processing a sequence is "unrolled" through time, it can be visualized as a very deep feedforward network where parameters are shared across time steps (layers in the unrolled view). This unrolled perspective, though specialized, highlights a layered computational flow. Often, as seen in this notebook's PanelLSTM, the LSTM's output is then fed into a standard feedforward layer (an MLP component) for the final prediction, explicitly linking the two architectures. Thus, an LSTM can be considered a sophisticated evolution of MLP concepts, adapted for the complexities of sequential data.

The LSTM model's hyperparameters include its input feature size of 32 per time step, a sequence length of 3, an LSTM hidden size of 64 across 2 layers with 0.2 dropout, the Adam optimizer with a learning rate of 1e-3, and a StepLR learning rate scheduler (with step_size=5 and gamma=0.5). It also uses MSELoss, a batch size of 64, and is trained for 100 epochs. Overfitting prevention relies on the LSTM's internal dropout, validation set performance monitoring, and the adaptive learning rate provided by the scheduler.

Comparative Model Notes: The MLP is structured for static feature sets, effectively using embeddings for categorical data. In contrast, the LSTM is designed to capture temporal dependencies within sequential data. Architecturally, the LSTM is generally more complex due to its recurrent connections and internal memory cells. Both models utilize Adam optimization and MSE loss, but the LSTM benefits from an additional learning rate scheduler to fine-tune the optimization process. The conceptual link shows LSTMs leveraging MLP-like computations internally while extending them for sequence modelling. One of the reasons why the LSTM model performance could have been better would be the use of embeddings as done in case of MLP. But here we use LabelEncoded ‘items’ only making it a generalised model for forecasting.

C. Features & Labels

The common feature columns included here are `'Avg_CanopInt_inst',`

```
 'Avg_ESoil_tavg',
 'Land_cover_percent_class_1',
 'Land_cover_percent_class_2',
 'Land_cover_percent_class_3',
 'Land_cover_percent_classh_4',
 'Land_cover_percent_class_5',
 'Land_cover_percent_class_6',
 'Land_cover_percent_class_7',
 'Land_cover_percent_class_8',
 'Land_cover_percent_class_9',
 'Land_cover_percent_class_10',
 'Land_cover_percent_class_11',
 'Land_cover_percent_class_12',
 'Land_cover_percent_class_13',
 'Land_cover_percent_class_14',
 'Land_cover_percent_class_15',
 'Land_cover_percent_class_16',
 'Land_cover_percent_class_17',
 'Avg_Rainf_tavg',
 'Avg_Snowf_tavg',
 'Avg_SoilMoi0_10',
 'Avg_SoilMoi10_40',
 'Avg_SoilMoi100_200',
 'Avg_SoilMoi40_100',
 'Avg_SoilTMP0_10',
 'Avg_SoilTMP10_40',
 'Avg_SoilTMP100_200',
 'Avg_SoilTMP40_100',
 'Avg_TWS_inst',
 'Avg_TVeg_tavg'
```

Multilayer Perceptron (MLP) Model: The input features for the MLP model are twofold. Monthly value of the numerical features were averaged out to get a yearly trend. Numerical features consist of annually averaged values of various agro-climatic variables (such as canopy interception, soil properties, land cover types, rainfall, etc.), which are first aggregated by country and year, then scaled using StandardScaler, and finally dimensionality-reduced via PCA, retaining 95% of the variance. The PCA keeps 16 features. Categorical features include 'country_id' and 'item_id', which are label-encoded versions of 'country' and 'Item' respectively, and these are fed into dedicated embedding layers. The output label is the 'yield_value', scaled using StandardScaler, sourced from the Yield_and_Production_data.csv file (specifically where the 'Element' column is 'Yield'). The rationale is to use a comprehensive set of environmental drivers as features, allow the model to learn representations for countries and items through embeddings, and use PCA to manage feature dimensionality, all aimed at predicting the direct measure of crop productivity.

Long Short-Term Memory (LSTM) Model: The LSTM model takes sequences of 3 time steps as input, where each time step contains 32 features. These features include various scaled agro-climatic variables and the 'item_id' (label-encoded 'Item'), which is treated as a numerical feature within the sequence. These sequences are generated by the make_panel_sequences_default function, which processes data previously grouped by 'country' and 'item_id' and applies StandardScaler to all features within the sequences. The output label is 'Value_scaled', representing the StandardScaler transformed 'yield_value' for the year immediately following the input sequence. The design is intended to leverage the LSTM's capability to learn from temporal patterns in the data, with 'item_id' included in the sequence to potentially capture item-specific temporal behaviours.

Comparative Features & Labels Notes: The fundamental difference lies in how features are structured and utilized. The MLP model uses a "snapshot" of features, including learned embeddings, for each country-item-year instance to make a prediction. In contrast, the LSTM model processes an explicit sequence of yearly feature sets to understand temporal dynamics leading to a prediction. The MLP's features undergo aggregation and PCA, while the LSTM's features are maintained in their time-step resolved (though scaled) form within the generated sequences. Both models ultimately aim to predict the same underlying 'yield_value', albeit using differently preprocessed versions of this target during training.

D. Preprocessing

The ‘merged_cols’ DataFrame consolidates diverse environmental, geographic, and agricultural datasets into a unified structure for crop yield forecasting. Its construction involves the following key stages:

1. Loading and Preprocessing Raw Data

Multiple datasets are loaded, including:

- Monthly environmental variables: canopy interception, soil moisture, evaporation, temperature, precipitation, vegetation transpiration, and water storage.
- Static data: land cover percentages and country centroid metadata.
- Agricultural records: crop yield and production statistics from FAO.

2. Temporal Aggregation

Monthly environmental measurements are averaged annually per (latitude, longitude, year), resulting in features like Avg_CanopInt_inst. This step reduces temporal granularity while retaining seasonal trends. Land cover data, already annual, is merged as-is.

3. Dataset Merging

The processed environmental and land cover datasets are merged incrementally on shared spatial-temporal keys (longitude, latitude, year). Inner joins ensure alignment and consistency. Any resulting missing values are removed.

4. Geolocation to Country Mapping

Grid cells are assigned to countries using a nearest-neighbor approach via a cKDTree, built from cleaned country centroid data. This method offers greater geographic coverage (194 countries) than bounding box matching(150 countries) and ensures each (lat, lon) point is mapped to the most geographically appropriate country.

5. Spatial Aggregation to Country-Year Level

Once geolocated, spatial columns are dropped. Data is aggregated to the (country, year) level by computing mean values for all numeric features, resulting in a single annual environmental profile per country.

6. Merging Crop Yield Data

Yield data is filtered for entries with "Element" == "Yield" and merged with the environmental dataset using country and year as keys. Column names are standardized, and the final merged structure includes:

- Annual environmental/climatic indicators
- Land cover statistics
- Country and crop identifiers
- Reported crop yield values (yield_value)

Multilayer Perceptron (MLP) Model: The preprocessing pipeline for the MLP model involves several key stages. Initially, missing values are handled by dropping rows with any NaN values after the primary merging of feature datasets. Categorical variables 'country' and 'Item' are converted to numerical IDs via label encoding. Subsequently, all numerical input features and the target 'yield_value' are scaled using StandardScaler. A significant step is the application of Principal Component Analysis (PCA) to the scaled numerical features, where components explaining 95% of the variance are retained. Finally, all processed data is converted into PyTorch tensors and managed using DataLoader for efficient batching during training. The rationale behind these steps is to create a clean, numerical, and dimensionality-managed static feature set suitable for the MLP, while also normalizing feature ranges for optimal training.

Long Short-Term Memory (LSTM) Model: For the LSTM model, preprocessing begins with grouping the data by 'country' and 'item_id' to form panels. The 'Item' categorical feature is label-encoded to 'item_id' and subsequently treated as a numerical feature. Both these input features and the target 'yield_value' are scaled using StandardScaler. The most distinctive preprocessing step for the LSTM is the creation of sequences using the make_panel_sequences_default function. This function constructs sliding windows from the time-ordered data within each panel, where three years of scaled features form an input sequence, and the scaled yield of the fourth year serves as the target. As with the MLP, the data is then converted to PyTorch tensors and fed into DataLoader instances, which handle batching and shuffling for the training set. The overall aim is to transform the initial tabular data into a sequential format that LSTMs are designed to process, while also normalizing data ranges.

Comparative Preprocessing Notes: Both models share foundational preprocessing activities such as label encoding for the 'Item' feature, scaling of numerical data (features and target) using StandardScaler, conversion to PyTorch tensors, and the use of DataLoader. The critical divergence in preprocessing is the MLP's use of PCA for feature dimensionality reduction on its static, aggregated feature set, versus the LSTM's intensive sequence generation process, which fundamentally reshapes the data to capture temporal relationships. The MLP operates on a feature vector representing a single point in time (or an aggregation), while the LSTM explicitly models a series of feature vectors over time.

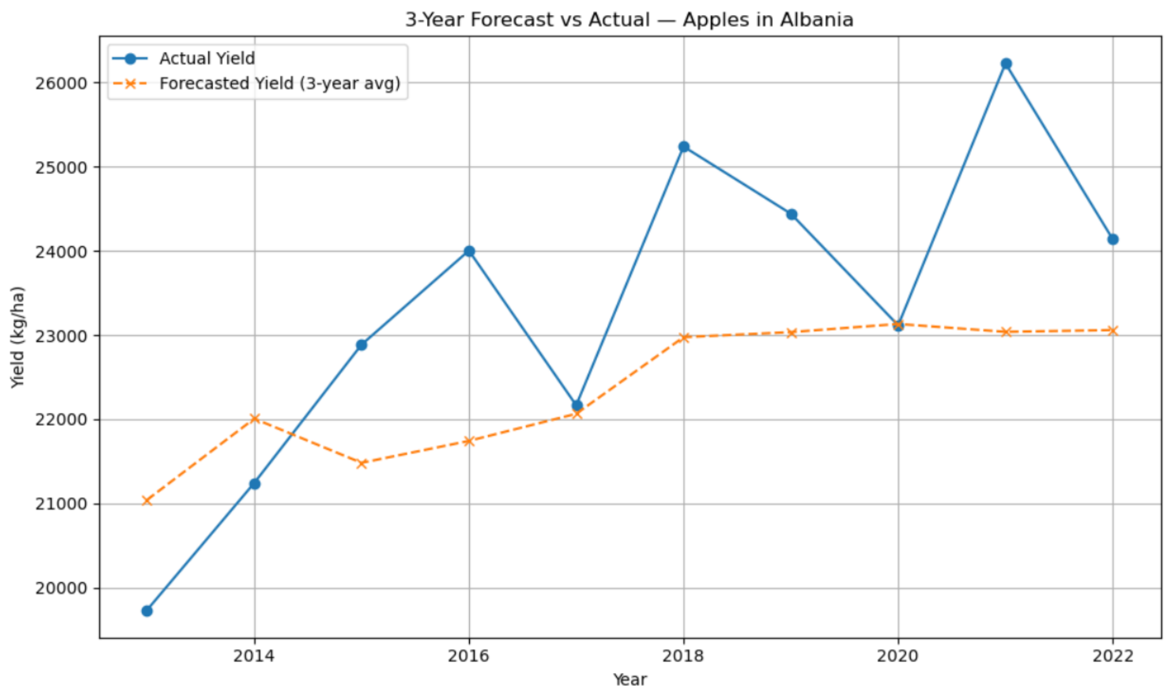

Fig. 1 – Forecasting vs Prediction Function included in notebook

Overall Comparative Conclusion

While LSTMs are theoretically well-suited for time-dependent data, in this case, the **MLP emerged as both more effective and more practical**. Its superior performance indicates that the problem may be more **spatial-aggregated** or **feature-driven** than inherently temporal, or that the LSTM architecture requires further tuning (e.g., longer sequences, attention mechanisms) to be competitive.

Aspect	Description
Model Type	Feedforward Neural Network with **Embedding Layers**
Input Structure	Static feature vector per instance (country-item-year). Numerical features **PCA-transformed**.
Categorical Handling	**Embedding layers** for **'country_id' (dim=8)** and **'item_id' (dim=8)**.
Core Architecture	**2 Hidden Dense Layers (128 units -> ReLU -> Dropout(0.3) ; 64 units -> ReLU -> Dropout(0.3))**, 1 Output Unit.
Key Hyperparameters	**Embedding Dim: 8**, **Hidden Dims: [128, 64]**, **Dropout: 0.3**, **LR: 0.001 (Adam)**, **Batch Size: 64**, **Epochs: 100**.
Preprocessing Highlights	Data aggregation, **Label Encoding**, **StandardScaler**, **PCA (95% variance)**.
Validation R^2 (Epoch 100)	**0.8582**
Overfitting Prevention	**Dropout**, Validation Set, **PCA**.

Table 1: Multilayer Perceptron(MLP) Model Summary

Aspect	**Description**
Model Type	**Recurrent Neural Network (*PanelLSTM*)**
Input Structure	**Sequences of feature vectors (3 time steps, 32 features per step). 'item_id' included as a feature.**
Categorical Handling	**'item_id' as a scaled numerical feature** within sequences. **'country' used for panel grouping** before sequencing.
Core Architecture	**2-Layer Stacked LSTM (Hidden Size: 64, Dropout: 0.2)**, 1 Linear Output Unit.
Key Hyperparameters	**Seq Length: 3, LSTM Hidden: 64, LSTM Layers: 2, LSTM Dropout: 0.2, LR: 1e-3 (Adam), Scheduler, Batch: 64, Epochs: 100.**
Preprocessing Highlights	Data grouping by country/item, **Label Encoding ('Item'), StandardScaler, Sequence Generation** (*make_panel_sequences_default*).
Validation R^2 (Epoch 100)	**0.4978**
Overfitting Prevention	**LSTM Dropout**, Validation Set, **Learning Rate Scheduler.**

Table 2: Long Short-Term Memory (LSTM) Model Summary