Journal for Crop Price Prediction Using LSTM

Introduction

Predicting crop prices is essential for farmers, traders, and policymakers to make informed decisions. In this project, we utilized a Long Short-Term Memory (LSTM) neural network to predict future crop prices based on historical data. This document provides an overview of the process, including data preparation, model training, prediction, and visualization.

Data Preparation

1. Loading the Dataset: The dataset used in this project was loaded from a CSV file named ***Total_Crops.csv.*** This file contains historical data on various crops across different mandis (markets). The columns ***include mandiid, cropid, cropname, mandiname, arrivalquantity, maximumprice, minimumprice, modalprice***, and ***date.***
2. User Input for Filtering Data: Users were prompted to input the ***cropid*** and optionally the ***mandiid*** to filter the data. If the ***mandiid*** was left blank, the data for all mandis was selected.
3. Data Filtering: The dataset was filtered based on the user's input. If no data was found for the given ***cropid*** and ***mandiid***, an appropriate message was displayed.
4. Normalization: The target variable ***modalprice*** was extracted, reshaped, and normalized using the ***MinMaxScaler*** from ***sklearn***.
5. Data Splitting: The data was split into training and testing sets with a 70%-30% ratio.
6. Dataset Creation: A helper function was used to create the dataset for the LSTM model. This function generated sequences of data with a specified look-back period.

Model Training

1. LSTM Network Architecture:

    - *Input Layer*: The model took sequences of shape ***(1, look_back***).
    - *LSTM Layers*: Two LSTM layers with 100 units each, followed by dropout layers to prevent overfitting.
    - *Output Layer*: A dense layer with a single neuron for the price prediction.

2. Early Stopping: Early stopping was used to monitor the validation loss and stop training if it did not improve for a specified number of epochs, thereby preventing overfitting.
3. Training: The model was trained using the ***adam*** optimizer and ***mean_squared_error*** loss function for 200 epochs, with a batch size of 32.

Prediction

1. User Input for Prediction Duration: The user was prompted to input the number of days for which the price prediction was required.
2. Future Prediction Function: A function was created to generate future predictions based on the trained model. This function used the last sequence from the test set as the starting point and generated predictions iteratively.
3. Fixed Start Date: All future predictions were set to start from 4th March 2024.

Visualization

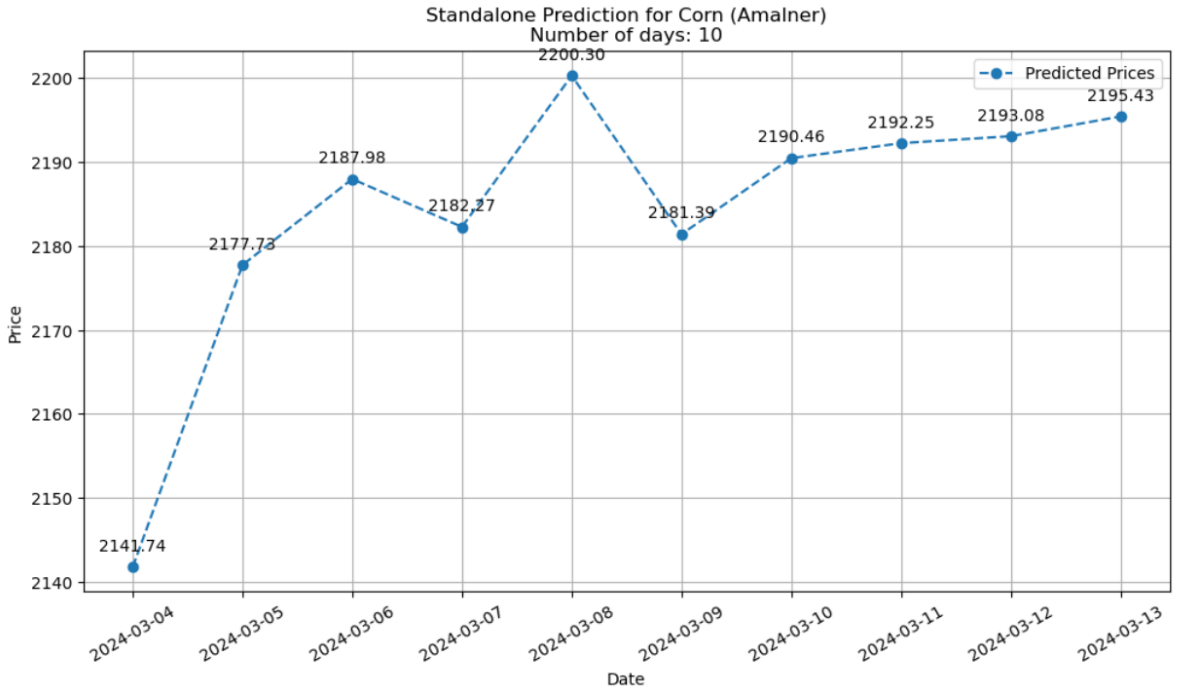
1. Standalone Prediction Plot: A standalone plot of the predicted prices was generated, with individual data points labeled and annotated.
2. Combined Plot: A combined plot of historical and predicted prices was created, displaying the historical prices as a scatter plot and the predicted prices as a line plot with annotated data points.

Challenges and Improvements

1. Overfitting and Underfitting: To tackle overfitting and underfitting, dropout layers and early stopping were used. These techniques helped in improving the model's generalization.
2. Accuracy: The look-back period was increased to consider more past days, which helped in capturing the temporal dependencies better. However, the model still exhibited some hyperbolic behavior towards the end of the prediction. Future improvements could include experimenting with different network architectures and hyperparameters.
3. Input Shape Warning: An initial warning related to input shapes was addressed by using an Input layer instead of directly specifying the ***input_shape*** in the ***LSTM*** layers.

Output:

Standalone –



Historical-