Crop-Price Prediction Journal

➢ Worked on creating a price-prediction model using the following methods

A. Crop-Wise filtered and Trained ML models

B. ML models trained on the entire Dataset

C. ML models trained on a Filtered Dataset

D. Neural-Networks Trained on a Filtered Dataset

➢ Datasets used-

1) Total_Crops_Input

2) Real_Time_Output

➢ Worked on Jupyter-Notebook(Python Programming Language)

❖ Python Packages used-

- **Pandas**:
  - **Contents**: Provides high-level data structures and functions designed to make data analysis fast and easy in Python.
  - **Importance**: Essential for data manipulation, cleaning, and analysis tasks, especially with structured data.
- **Numpy**:
  - **Contents**: Fundamental package for numerical computing with Python. It contains among other things a powerful N-dimensional array object.
  - **Importance**: Enables efficient numerical operations and array manipulations, foundational for scientific computing and data analysis.
- **Matplotlib**:

RESTRICTED

- **Contents**: Comprehensive library for creating static, animated, and interactive visualizations in Python.
- **Importance**: Crucial for data visualization, helps in understanding data distributions, trends, and relationships.

- **Scikit-learn (sklearn)**:

  - **Contents**: Simple and efficient tools for data mining and data analysis, built on NumPy, SciPy, and matplotlib.
  - **Importance**: Provides easy-to-use implementations of many machine learning algorithms for classification, regression, clustering, and more.

- **Xgboost**:

  - **Contents**: Optimized gradient boosting library that implements machine learning algorithms under the Gradient Boosting framework.
  - **Importance**: Known for its efficiency, speed, and performance in machine learning tasks, especially in structured/tabular data.

- **Datetime**:

  - **Contents**: Provides classes for manipulating dates and times in Python.
  - **Importance**: Essential for handling dates and times in data preprocessing, feature engineering, and time-series analysis.

- **Tensorflow**:

  - **Contents**: An open-source machine learning framework for dataflow and differentiable programming across a range of tasks.
  - **Importance**: Widely used for building and training neural networks and deep learning models, also supports deployment and production.

- **Scipy**:

  - **Contents**: Collection of mathematical algorithms and convenience functions built on NumPy data structures.
  - **Importance**: Provides tools for optimization, integration, interpolation, linear algebra, statistics, and more, complementing NumPy for scientific computing.

RESTRICTED

Analysis and info of the dataset

```
RangeIndex: 85260 entries, 0 to 85259
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   mandiid          85260 non-null  int64
 1   cropid           85260 non-null  int64
 2   cropname         85260 non-null  object
 3   mandiname        85260 non-null  object
 4   arrivalquantity  85260 non-null  int64
 5   maximumprice     85260 non-null  int64
 6   minimumprice     85260 non-null  int64
 7   modalprice       85260 non-null  int64
 8   date             85260 non-null  object
dtypes: int64(6), object(3)
memory usage: 5.9+ MB
 cropid  mandiid  count
      7      227    986
     13      359    959
      7      981    934
      1     1122    926
      1     1129    902
      1     1142    902
      7     1331    877
      1     1235    863
     13      167    859
      1     1212    859
```

mandiid	cropid	cropname	mandiname	arrivalquantity	maximumprice	minimumprice	modalprice	date
777	10	Paddy	Alibagh	100	2500	2000	2200	2021-08-03
234	7	Onion	SATANA	100	1800	750	1600	2021-08-03
778	1	Corn	Amalner	100	1700	1630	1700	2021-08-04
508	2	Cotton	Khambha	100	8625	3500	6425	2021-10-08
1212	1	Corn	Lakhimpur	100	1625	1590	1615	2021-10-08
1129	1	Corn	Bewar	100	1700	1650	1680	2021-10-10
359	13	Potato	RAJKOT	100	1350	600	1150	2021-10-15
1040	7	Onion	Umrane	100	3801	801	3000	2021-10-18
78	1	Corn	HARDOI	100	1630	1580	1600	2021-10-19
923	7	Onion	Manmad	100	2750	600	2400	2021-10-25

RESTRICTED

❖ Step by Step process through Each method-

Crop-Wise filtered and Trained ML models

- Classified the dataset on the basis of crop type to obtain the following 6 csv sheets for data analysis and prediction.

1. CORN
2. COTTON
3. ONION
4. POTATO
5. RICE
6. TOMATO

- SORTED THE DATA ON THE BASIS OF DATE
- STARTED WORKING ON CLEANING AND PREPROCESSING THE DATA IN PYTHON

To simplify the visualizations and focus on the predicted prices with clear labels, I'll streamlined the code to:

1. Train the model using RandomForestRegressor.
2. Predict the future modal prices for the next 10 days.
3. Visualize the predicted modal prices clearly.

- HERE ARE THE STEPS USED IN PYTHON

1. ***Data Loading and Preprocessing:***

- Load the dataset containing crop price data, ensuring that the date column is parsed correctly.
- Drop any irrelevant columns such as crop ID, crop name, and arrival quantity.
- Convert the date column to the appropriate datetime format.
- Extract additional features from the date, such as year, month, and day, to facilitate analysis.

RESTRICTED

2. ***Feature Engineering:***

- Create lag features by shifting the modal price data by a certain number of days. This step helps capture temporal dependencies in the data and is crucial for time series forecasting.

3. ***Data Encoding:***

- *Encode categorical variables, such as the name of the mandi, using techniques like label encoding. This step converts categorical data into a numerical format suitable for machine learning models.*

4. ***Model Training:***

- *Split the dataset into training and testing sets to evaluate the model's performance.*
- *Choose an appropriate machine learning model for the task. In this project, a RandomForestRegressor is used due to its ability to capture nonlinear relationships and handle tabular data effectively.*
- *Train the model using the training data, fitting it to the features (input variables) and the target variable (modal price).*

5. ***Model Evaluation:***

- *Make predictions on the test set using the trained model.*
- *Evaluate the model's performance using metrics such as Mean Absolute Error (MAE), which measures the average absolute difference between predicted and actual prices.*

6. ***Future Price Prediction:***

- Use the trained model to predict crop prices for the next 10 days.
- Utilize the lag features and the most recent known data point to forecast future prices iteratively.

7. ***Visualization:***

- Visualize the predicted modal prices for the next 10 days using a line plot.
- Annotate each data point with its corresponding predicted price to provide clear labels and aid interpretation.

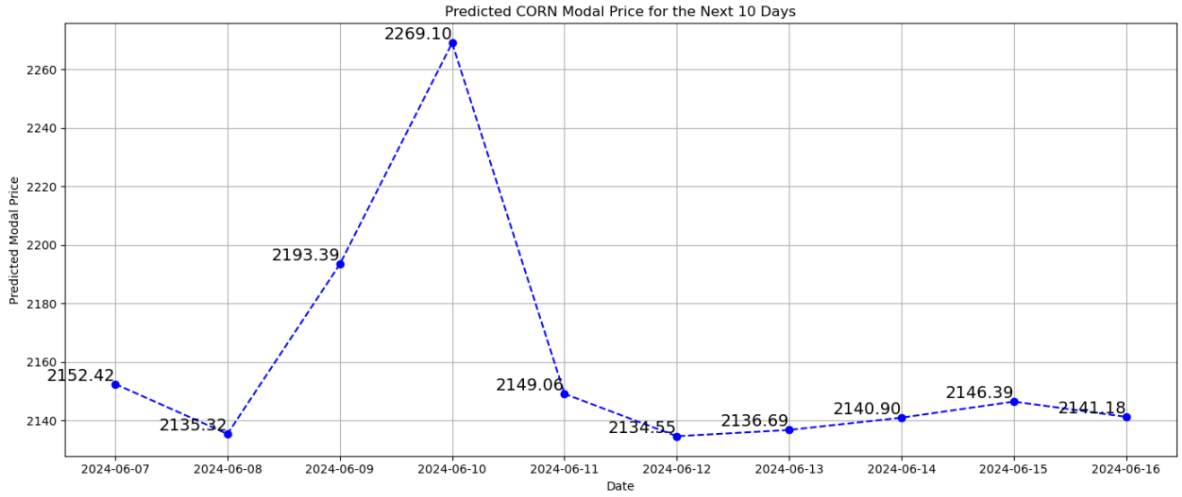
8. ***Project Journal Entry:***

- Document each step of the data preprocessing, feature engineering, model training, evaluation, and prediction process in the project journal.
- Include explanations of the techniques used, rationale behind the choices made, and any challenges encountered during the project.
- Reflect on the model's performance, discussing its strengths, limitations, and potential areas for improvement.

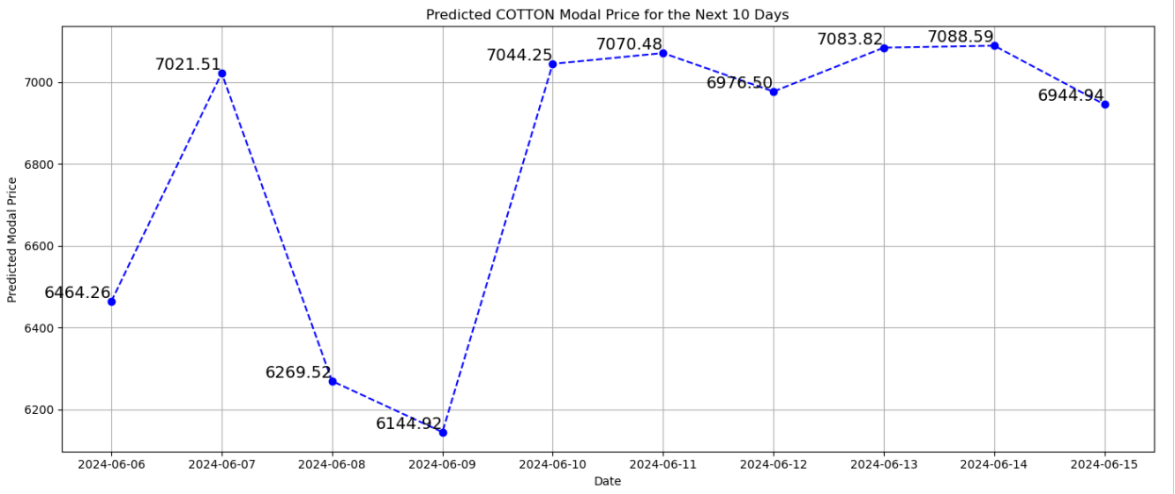
RESTRICTED

Outputs:

1. CORN

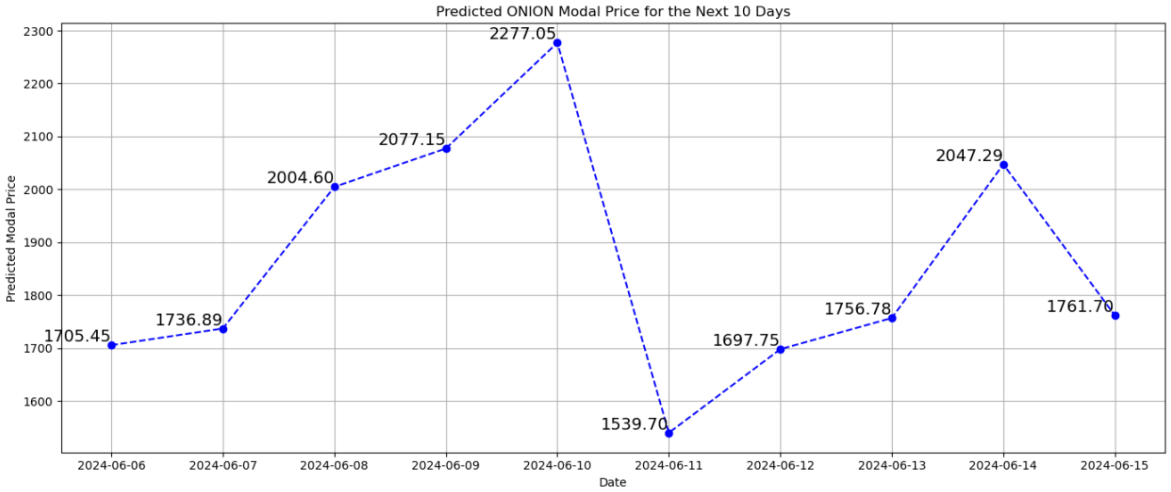


2. COTTON

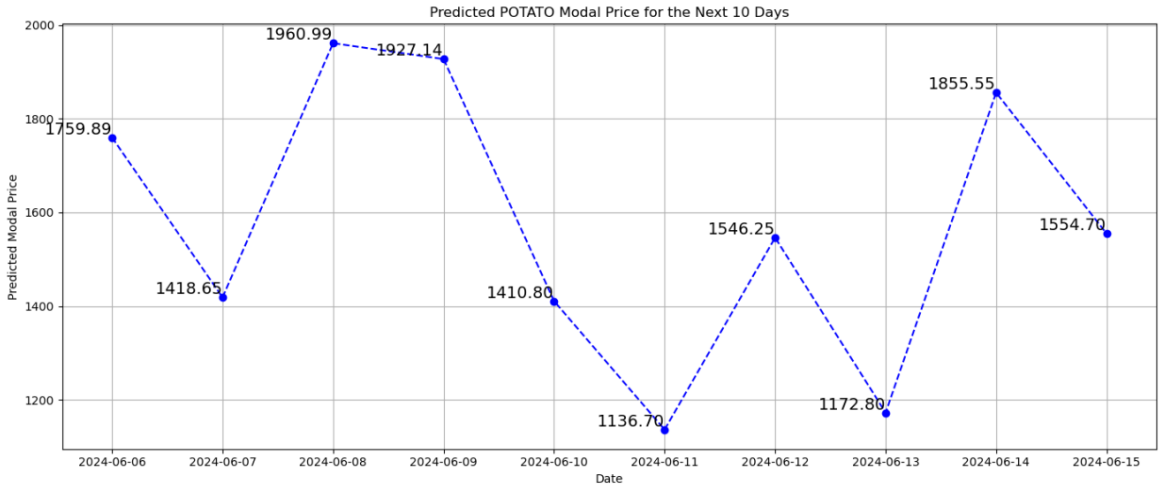


RESTRICTED

3. ONION

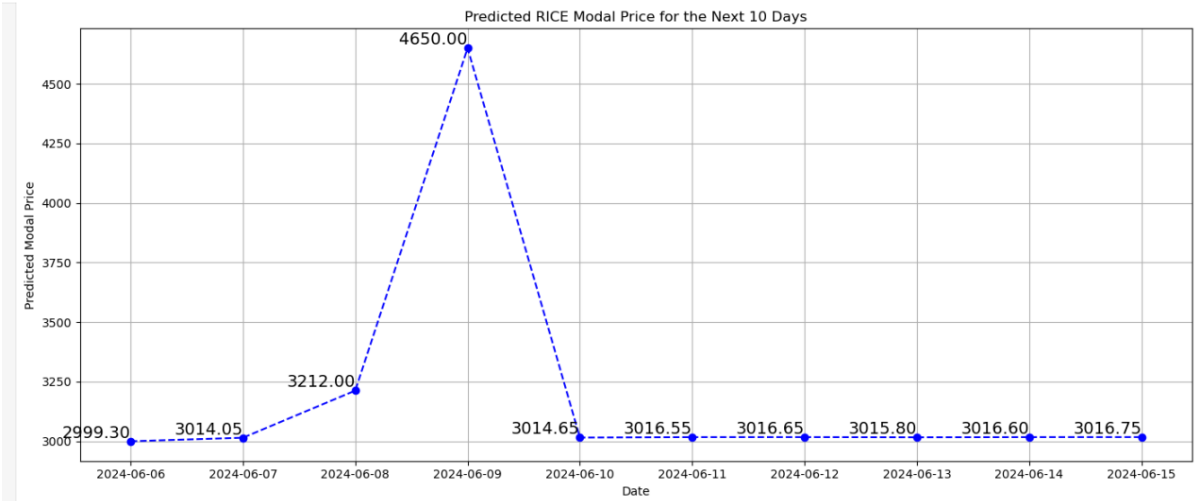


4. POTATO

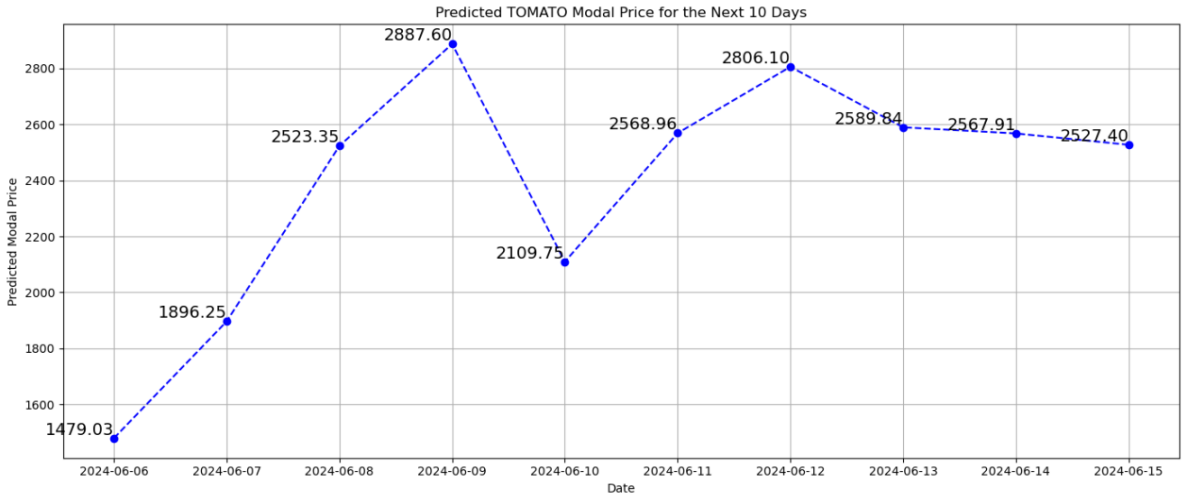


RESTRICTED

5. RICE



6. TOMATO



ML models trained on the entire Dataset

RESTRICTED

Predicting Crop Prices using Machine Learning Models

Introduction

This journal explores the application of three distinct machine learning models—XGBoost, Random Forest Regressor, and Support Vector Machines (SVM)—for predicting crop prices based on historical agricultural market data. Each model offers unique strengths in handling complex datasets and capturing different types of relationships between features and target variables.

Dataset Overview

The dataset used in this study, `Total_Crops.csv`, comprises essential attributes such as mandi ID, crop ID, crop name, mandi name, arrival quantity, maximum price, minimum price, modal price, and date. Key attributes for prediction include modal price, date, mandi ID, crop ID, and lagged price features derived from historical data. On working with the predictions and further analysis and close examination of the data, it was found to be inaccurate and inconsistent due to the presence of test data.

Data Preprocessing

1. Cleaning and Transformation: Irrelevant attributes like arrival quantity are removed. The date column is converted into datetime format, and additional temporal features such as year, month, and day are extracted.
2. Feature Engineering: Lag features are created using the past 21 days' prices to predict the next day's modal price. Categorical variables (mandi ID and crop ID) are encoded using LabelEncoder.
3. Handling Missing Values: Rows with missing values resulting from the creation of lag features are dropped to ensure data integrity.

Model Selection and Training

1. XGBoost Model

- *Model Characteristics*: XGBoost is selected for its scalability and performance in gradient boosting scenarios.
- *Training Process:* The dataset is split into training and testing sets. Hyperparameters such as learning rate, maximum depth, and number of estimators are tuned using cross-validation.
- *Evaluation*: Model performance is evaluated using Mean Absolute Error (MAE) on the test set.

RESTRICTED

2. Random Forest Regressor

- *Model Characteristics*: Random Forest is chosen for its ability to handle complex datasets and capture non-linear relationships.
- *Training Process*: Similar to XGBoost, the dataset is split, and the model is trained on the training set. Parameters like number of trees and maximum depth are optimized.
- *Evaluation:* MAE is used to assess prediction accuracy on the test set.

3. Support Vector Machines (SVM)

- *Model Characteristics*: SVM is employed due to its effectiveness in capturing complex relationships and handling high-dimensional data.
- *Training Process*: The dataset undergoes the same preprocessing steps. SVM parameters such as kernel type, regularization parameter (C), and kernel coefficient (gamma) are optimized during training.
- *Evaluation:* Model performance is measured using MAE on the test set.

4. Voting Regressor

- *Concept:* The Voting Regressor aggregates predictions from multiple base estimators (individual models) and outputs the average prediction, often resulting in improved generalization and stability compared to single models.
- *Implementation*: In this project, Voting Regressor combines predictions from Random Forest and Gradient Boosting models, leveraging their diverse learning approaches (bagging and boosting) to achieve better prediction accuracy.
- *Advantages*: Voting Regressor can handle different sources of data, model heterogeneity, and reduces overfitting by combining diverse models.

5. AdaBoost (Adaptive Boosting)

- *Concept*: AdaBoost sequentially trains a series of weak learners (e.g., decision trees) where each subsequent model corrects errors made by the previous one, focusing more on difficult instances.
- *Implementation:* In this project, AdaBoost uses Decision Tree regressors as weak learners. It adjusts the weights of incorrectly predicted instances iteratively, emphasizing challenging data points to enhance overall prediction accuracy.
- *Advantages*: AdaBoost is effective in handling complex relationships in data, improving predictive performance by focusing on hard-to-classify instances.

6. Gradient Boosting

- *Concept:* Gradient Boosting builds an ensemble of decision trees sequentially, where each tree corrects errors of the previous one by fitting residuals of the prediction gradient.

RESTRICTED

- *Implementation*: Here, Gradient Boosting combines multiple decision trees to create a strong learner. It iteratively improves the model's prediction by minimizing the loss function, often resulting in highly accurate predictions.
- *Advantages:* Gradient Boosting is robust against overfitting, handles missing data well, and provides feature importance, aiding interpretability in agricultural price forecasting.

Predictive Analysis and Visualization

- User Interaction: Users provide inputs for crop ID and optionally mandi ID. They specify the number of days (5-15) for which future price predictions are desired.
- Prediction Visualization:
    - *Historical and Predicted Prices*: Each model produces plots displaying historical modal prices alongside predicted prices for the specified number of days. This visualization aids in comparing observed trends with model forecasts.
    - *Standalone Predictions*: Additional plots show standalone predicted prices for the chosen number of days, with data labels for clarity. These plots focus on predicted modal prices over time.

Conclusion

This journal entry highlights the application of XGBoost, Random Forest, and SVM models in predicting crop prices, emphasizing their distinct methodologies and performance metrics. These models contribute to informed decision-making in agriculture, supporting sustainable practices and economic development.

Future Directions

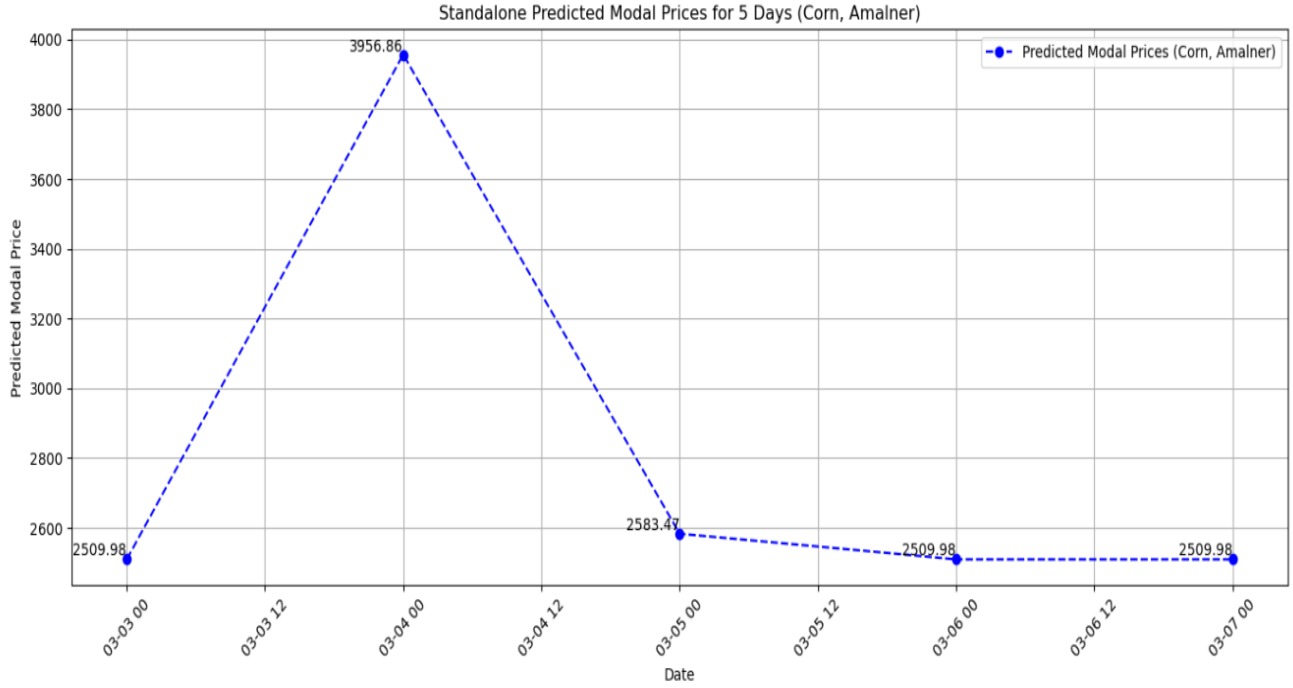
Future work may involve integrating additional datasets such as weather patterns, economic indicators, or regional factors to enhance model accuracy and robustness. Further exploration could expand the dataset to encompass diverse agricultural products and market regions, broadening the models' applicability and improving their predictive capabilities.

By documenting these methodologies and findings, this journal entry offers valuable insights into leveraging machine learning for agricultural price forecasting, fostering advancements in agricultural economics and policy-making.

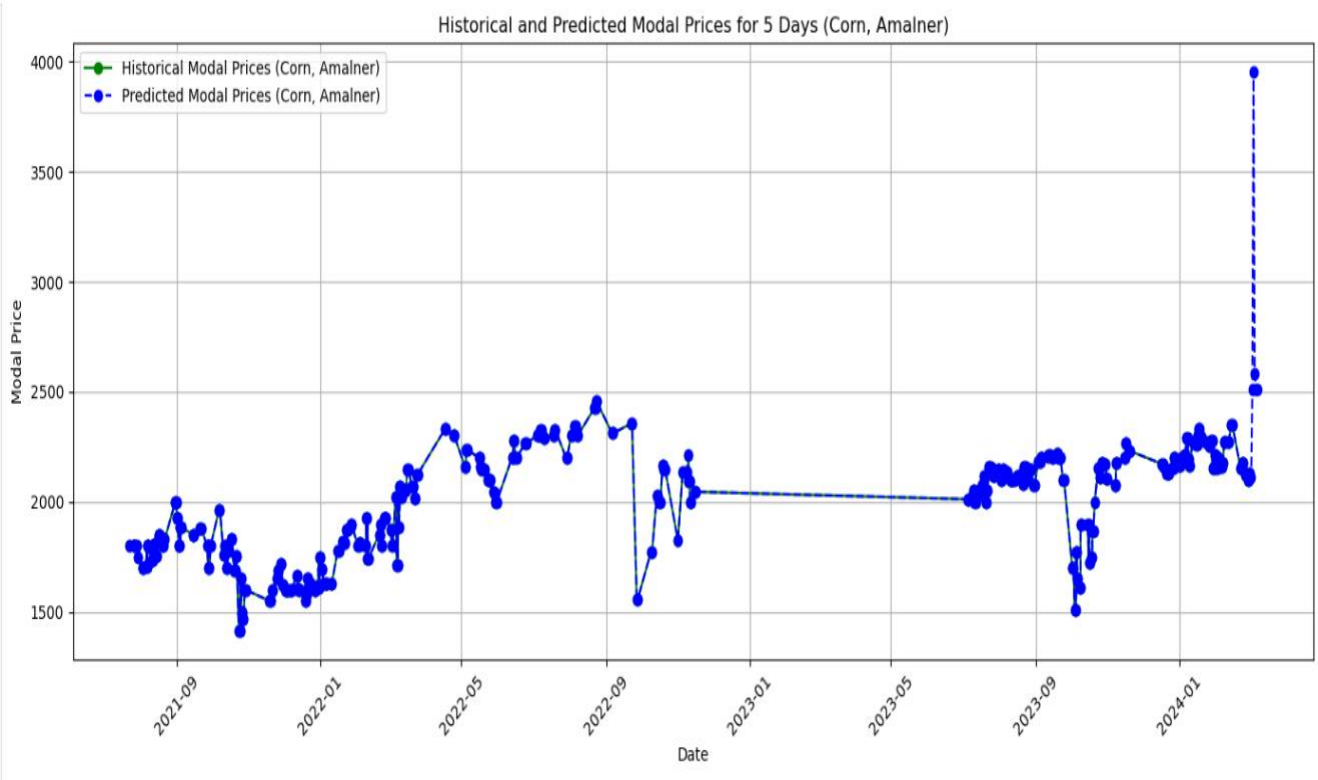
RESTRICTED

OUTPUT:

1) XGBoost Predicted Values graph

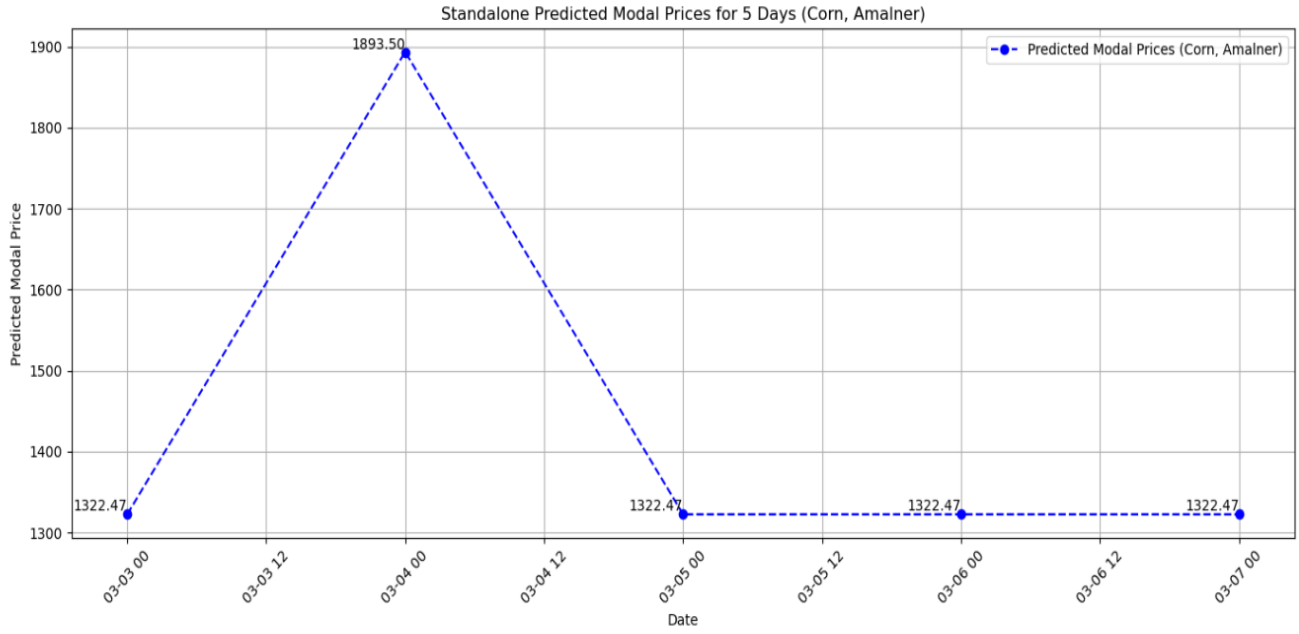


2) XGBoost Historical+Predicted Graph

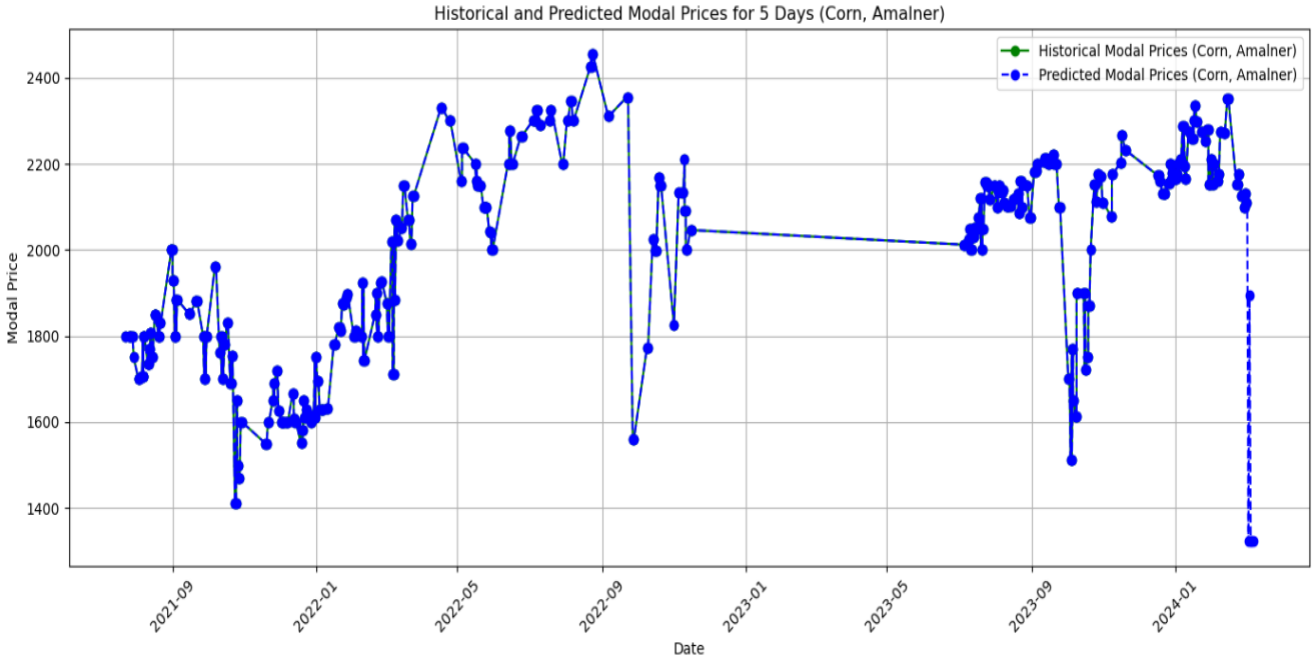


RESTRICTED

3) RandomForestRegressor Predicted Values graph

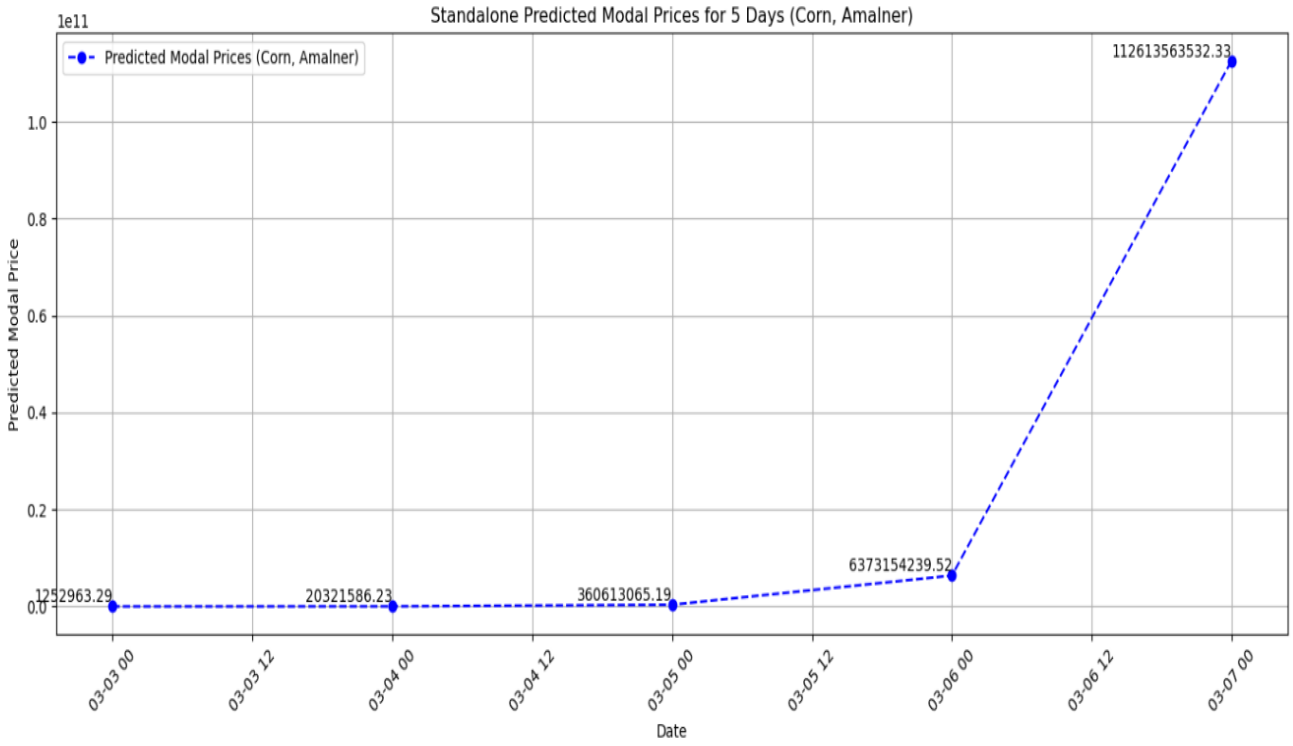


4) RandomForestRegressor Historical+Predicted Graph

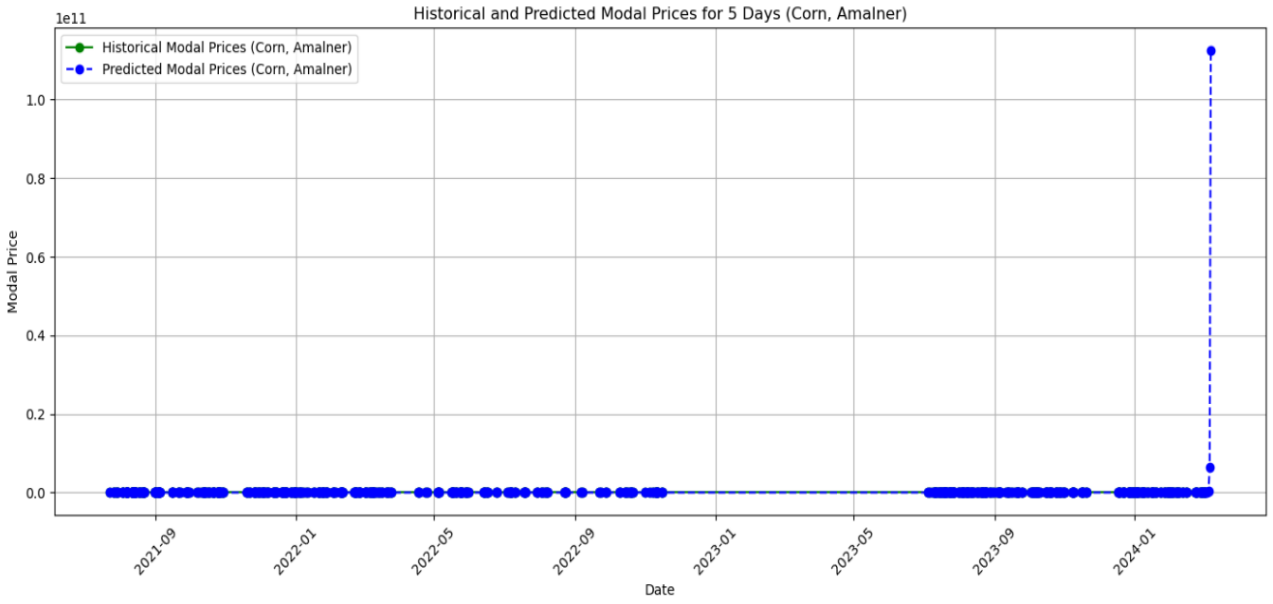


RESTRICTED

5) SVM Predicted Values graph

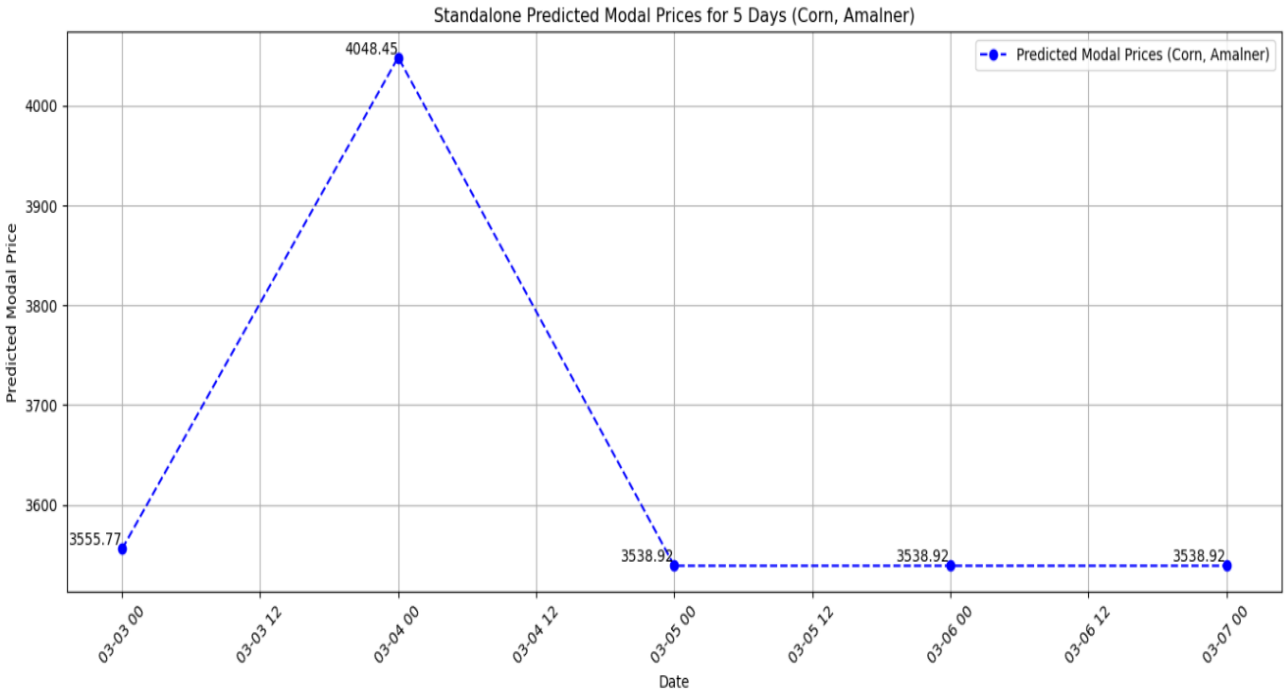


6) SVM Historical+Predicted Graph

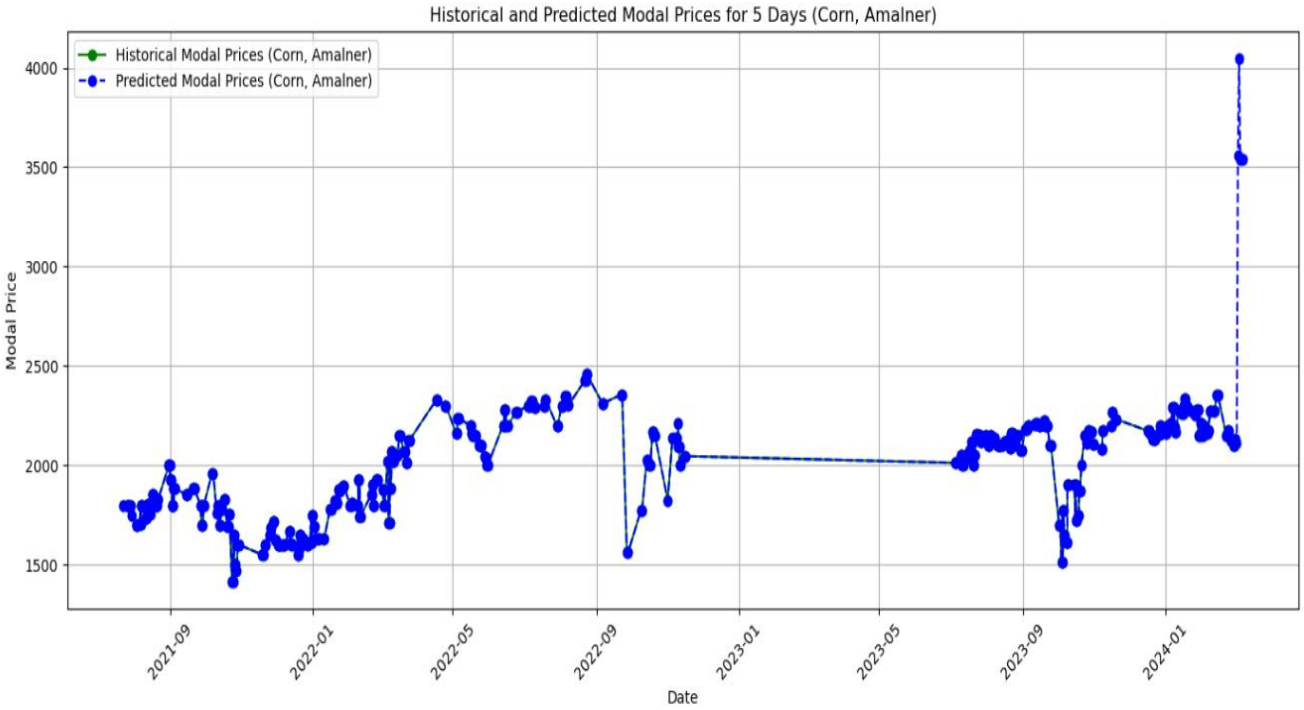


RESTRICTED

7) Voting Regressor Predicted Values graph

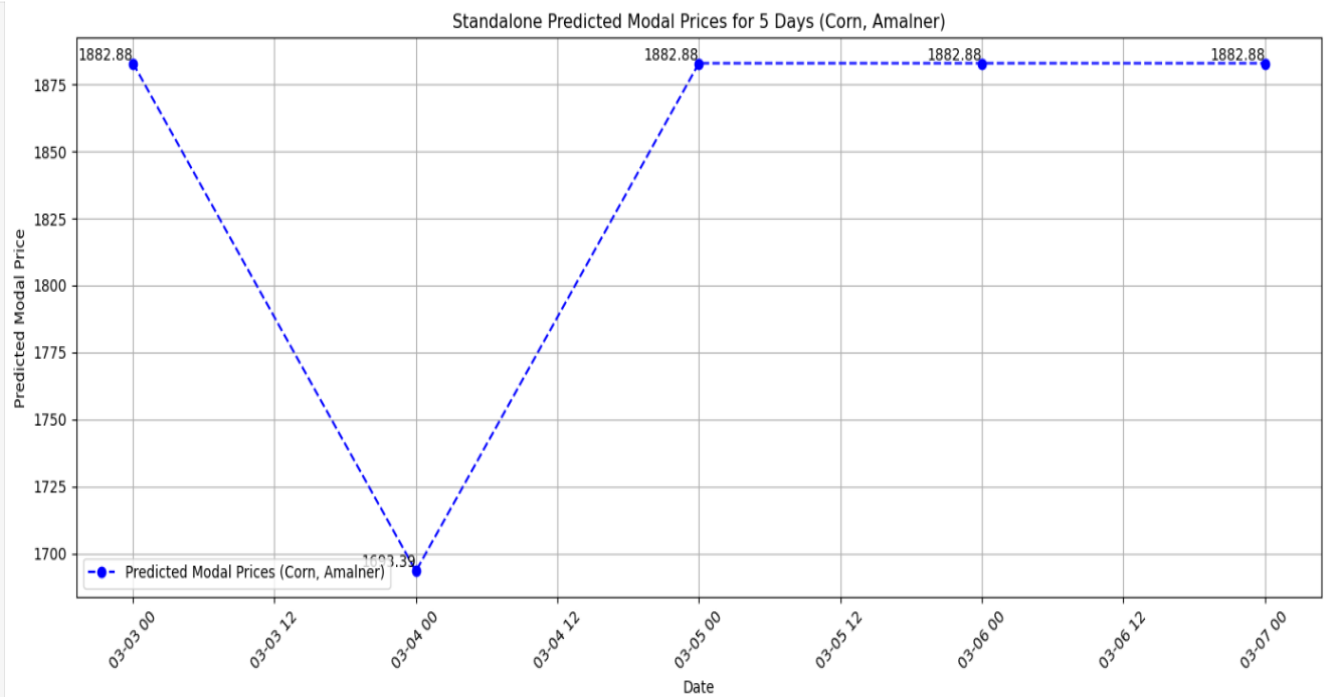


8) Voting Regressor Historical+Predicted Graph

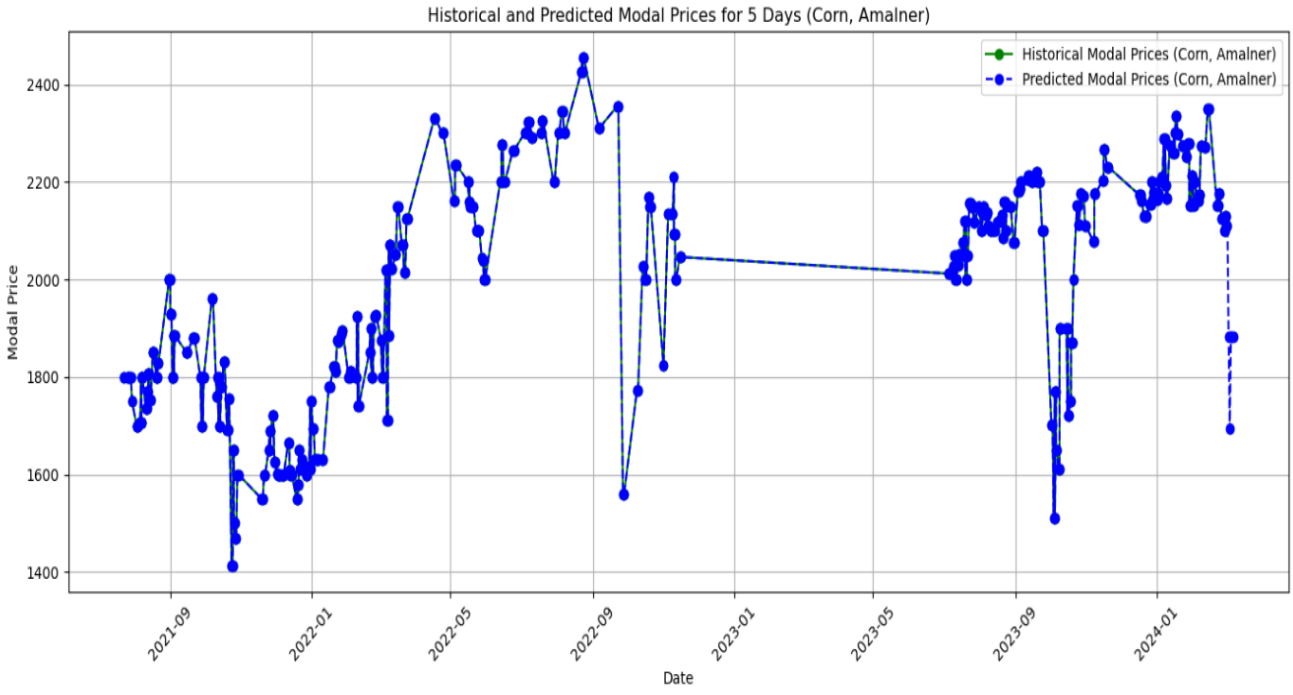


RESTRICTED

9) Adaboost Predicted Values graph

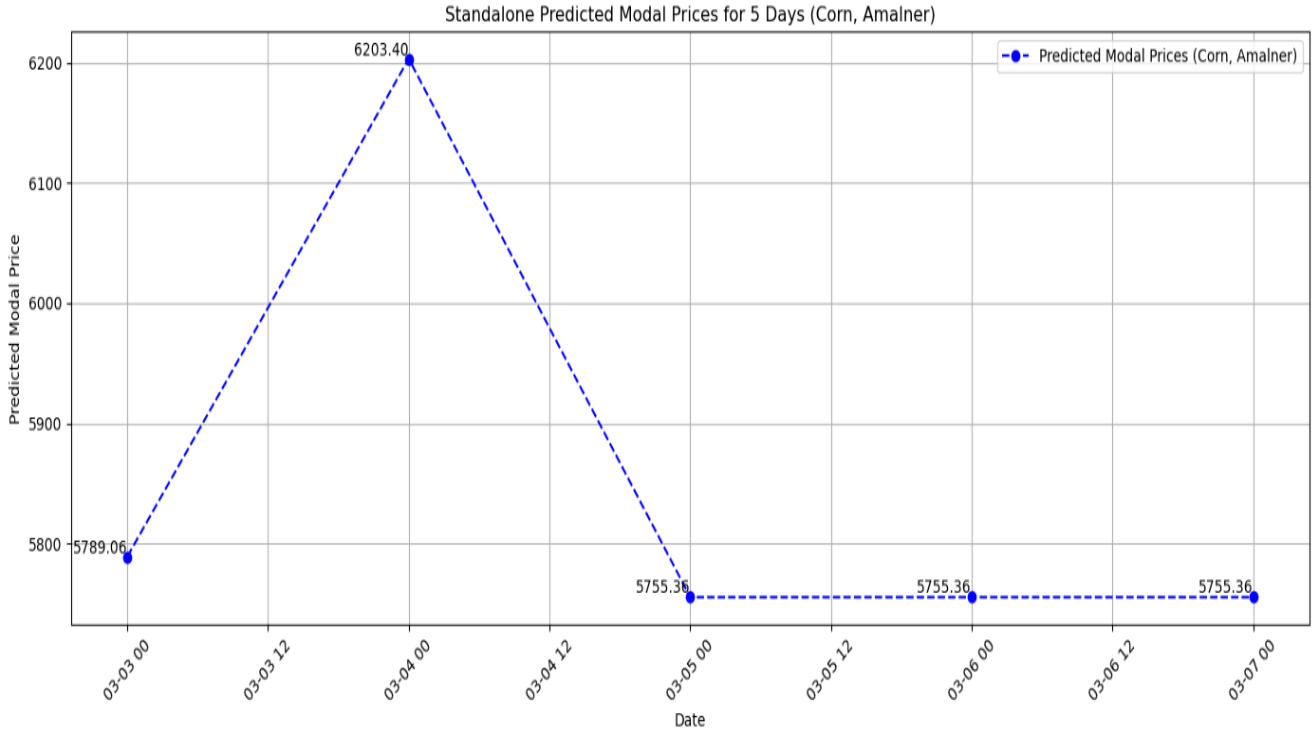


10) Adaboost Historical+Predicted Graph

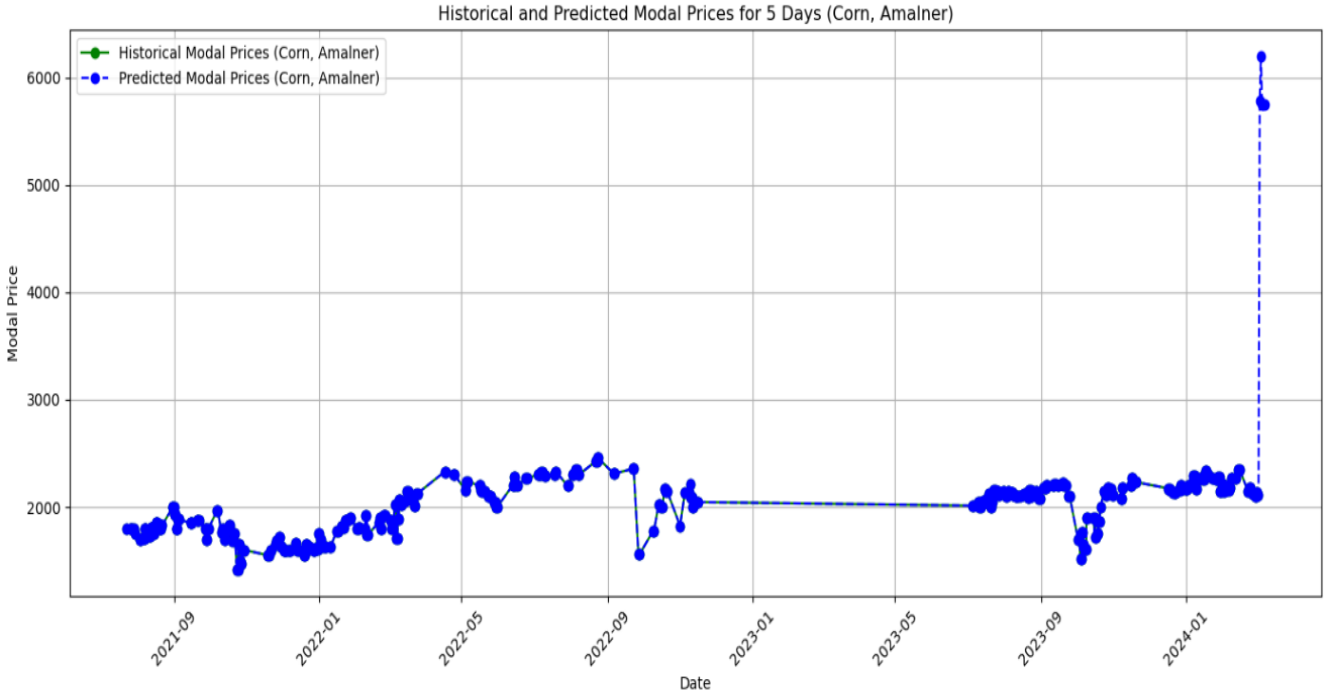


RESTRICTED

11) GBM Predicted Values graph



12) GBM Historical+Predicted Graph



RESTRICTED

ML models trained on a Filtered Dataset

Step-by-Step Theory for Predicting Crop Prices

1. Introduction:

The primary objective of this project is to predict the minimum, maximum, and modal prices of crops for the next 10 days using a non-linear regression model, specifically **XGBoost.** The dataset Total_Crops.csv comprises various attributes, including **mandiid, cropid, cropname, mandiname, arrivalquantity, maximumprice, minimumprice, modalprice**, and **date.**

2. Data Loading:

- **Purpose**: The first step is to load the dataset into a pandas Data Frame to make it suitable for processing and analysis.
- **Process**: Utilize the `pandas` library to read the CSV file into a Data Frame, which allows for easy data manipulation and analysis.

3. User Input for Filtering:

- **Purpose**: Filter the dataset to focus on specific market **(**`mandiid`) and crop (`cropid`) combinations, as specified by the user. This ensures the model is trained on relevant data.
- **Process**: Prompt the user to input the desired `mandiid` and `cropid` values. Filter the Data Frame to retain only the rows that match these criteria.

4. Drop Unnecessary Columns:

- **Purpose**: Remove columns that are not needed for the prediction model to reduce complexity and improve performance.
- **Process**: Drop the `arrivalquantity` column from the filtered Data Frame as it is not required for predicting prices.

5. Date Conversion and Formatting:

- **Purpose**: Convert the `date` column to a standard datetime format and reformat it to `day/month/year`. This ensures consistency in date representation and facilitates feature extraction.
- **Process**: Use the `pandas` library to convert the `date` column to datetime format and reformat it to the desired format. Extract additional date-related features (year, month, day) from the date for use as predictors.

RESTRICTED

6. Creating Lag Features:

- **Purpose**: Generate lagged features to capture the time-series dependencies in the data. Lag features represent the values of a variable at previous time points.
- **Process**: Create lagged variables for the past 7 days of **`modalprice`**, **`minimumprice`**, and **`maximumprice`**. This involves shifting the original price columns by 1 to 7 days and adding these shifted values as new columns in the DataFrame. Rows with missing values created by lagging are dropped.

7. Encoding Categorical Variables:

- **Purpose**: Convert categorical variables into numerical format, which is required by most machine learning algorithms.
- **Process**: Apply label encoding to the **`mandiname`** and **`cropname`** columns, transforming categorical values into unique integer labels.

8. Defining Features and Targets:

- **Purpose**: Specify the input features and target variables for the model. The features include both original and engineered variables, while the targets are the prices we aim to predict.
- **Process**: Define a list of features, including date components, lagged price features, and encoded categorical variables. The target variables are the **`minimumprice`**, **`maximumprice`**, and **`modalprice`**.

9. Train-Test Split:

- **Purpose**: Split the dataset into training and testing sets to evaluate the model's performance on unseen data.
- **Process**: Use **`train_test_split`** from the **`sklearn`** library to divide the data into training and testing sets. Typically, 80% of the data is used for training, and 20% is reserved for testing.

10. Model Training:

- **Purpose**: Train a multi-output regressor to predict multiple target variables simultaneously. **XGBoos**t is chosen for its efficiency and ability to handle non-linear relationships.
- **Process**: Utilize the **`MultiOutputRegressor`** wrapper with **`XGBRegressor`** to train a model on the training data. This involves fitting the model to the input features and target variables.

RESTRICTED

11. Model Evaluation:

- **Purpose**: Evaluate the model's performance using appropriate metrics to understand its accuracy and effectiveness.
- **Process**: Predict the prices on the test set and compute the **Mean Absolute Error (MAE)** to quantify the prediction error**. MAE** provides an average of the absolute errors between predicted and actual values.

12. Future Predictions:

- **Purpose**: Predict future crop prices for the next 10 days based on the trained model. This step involves iteratively updating the lag features with the model's predictions.
- **Process**: Start with the last known values from the filtered data and use the model to predict the next day's prices. Update the lag features with these predicted values and repeat the process for 10 days to generate future predictions.

13. Visualization:

- **Purpose**: Create visual representations of the historical and predicted prices to provide a clear and intuitive understanding of the trends and future forecasts.
- **Process**:
    - **Historical and Predicted Modal Prices**: Plot both historical and predicted `modalprice` values on the same graph. Use markers to indicate data points and add labels to each predicted data point for clarity.
    - **Standalone Predicted Modal Prices**: Plot only the predicted `modalprice` values for the next 10 days, with labels for each data point.
    - **Historical and Predicted Minimum Prices**: Similarly, plot both historical and predicted `minimumprice` values with labels for predicted points.
    - **Standalone Predicted Minimum Prices**: Plot only the predicted `minimumprice` values for the next 10 days, with labels.
    - **Historical and Predicted Maximum Prices**: Plot both historical and predicted `maximumprice` values with labels for predicted points.
    - **Standalone Predicted Maximum Prices**: Plot only the predicted `maximumprice` values for the next 10 days, with labels.

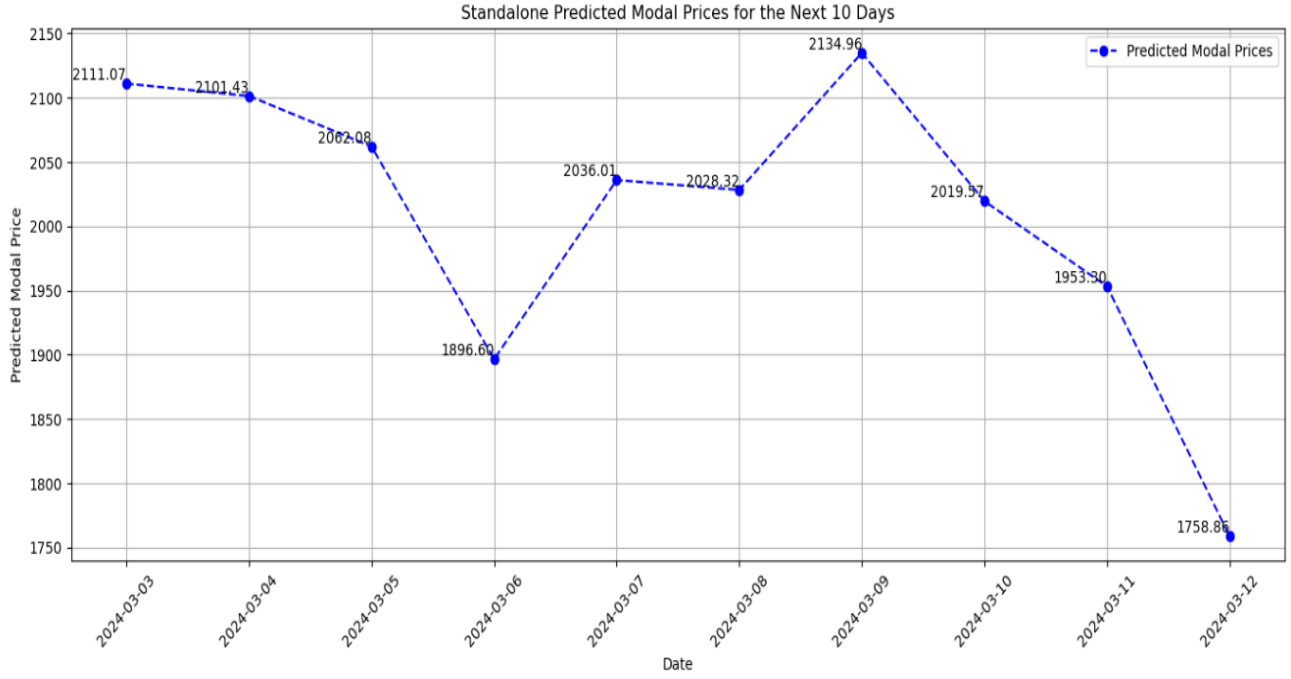
Summary:

This step-by-step process provides a comprehensive approach to predicting crop prices, leveraging historical data, and using advanced machine learning techniques. The key steps involve data preprocessing, feature engineering, model training, evaluation, and visualization. This method ensures that the predictions are both accurate and interpretable, providing valuable insights into future crop price trends.

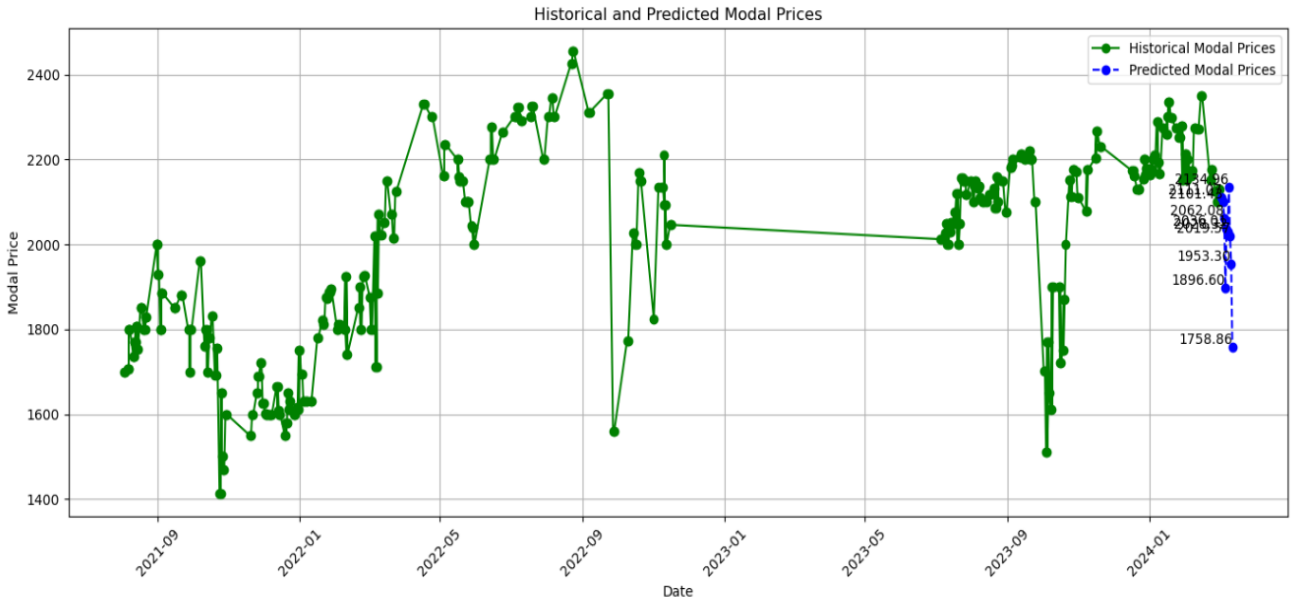
RESTRICTED

Output:

1. XGBoost Predicted values

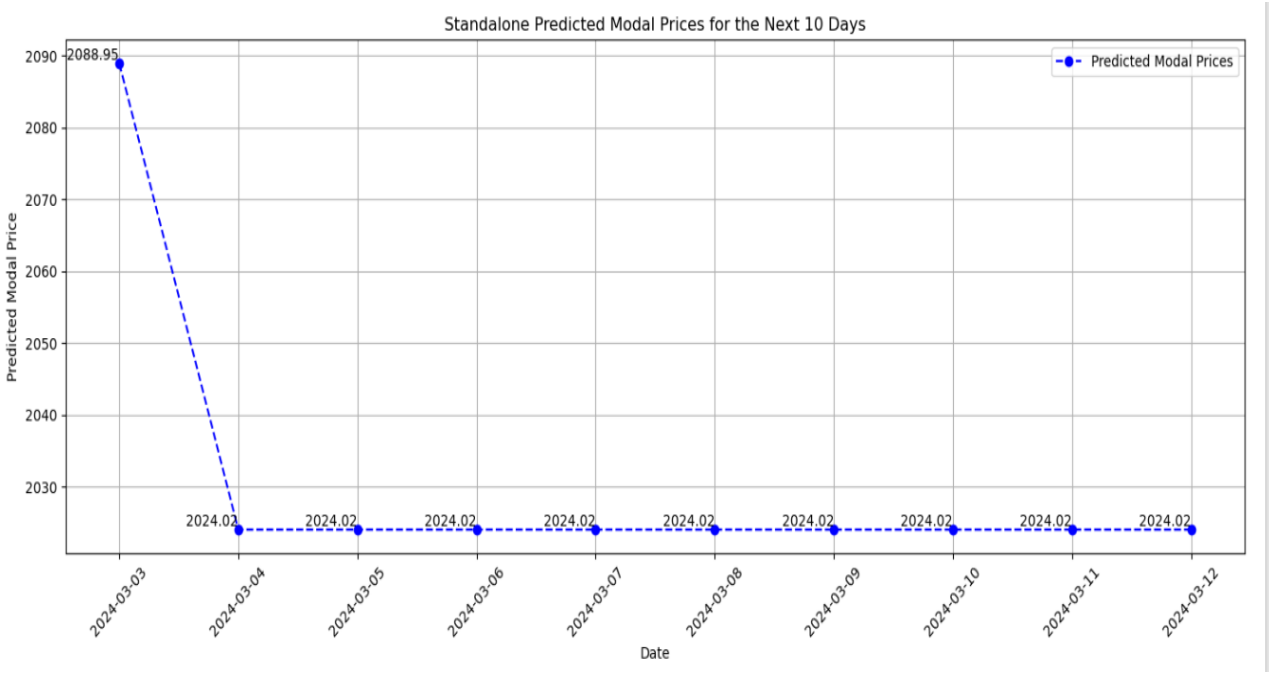


2. XGBoost Historical+Predicted values Graph

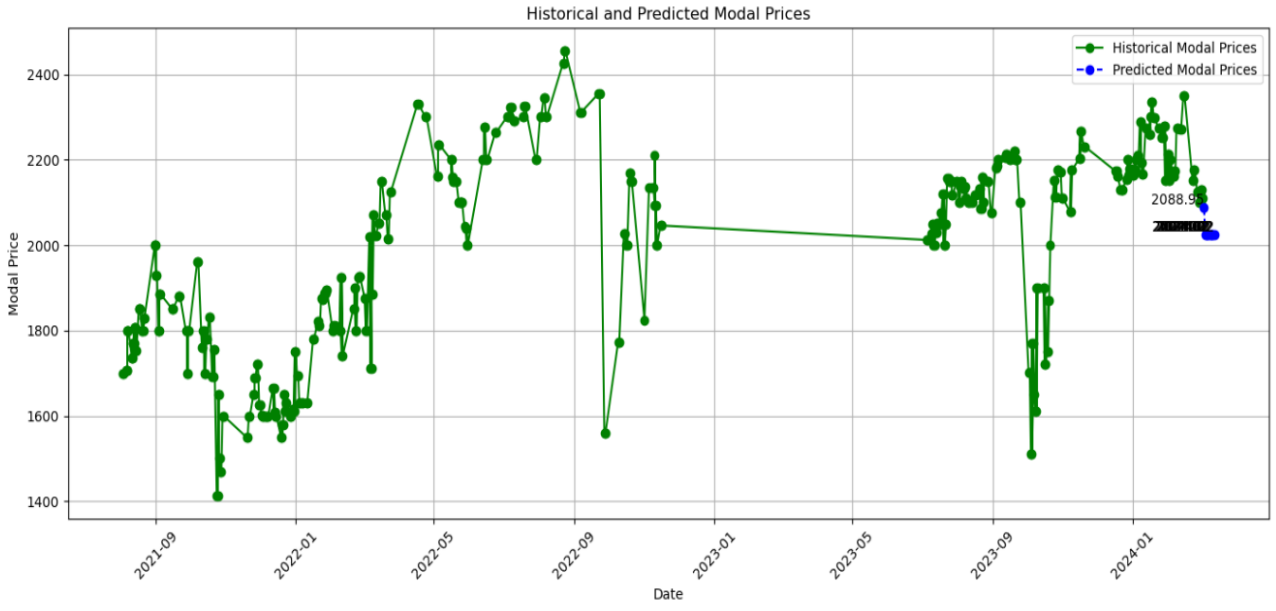


RESTRICTED

3. SVR- Predicted values

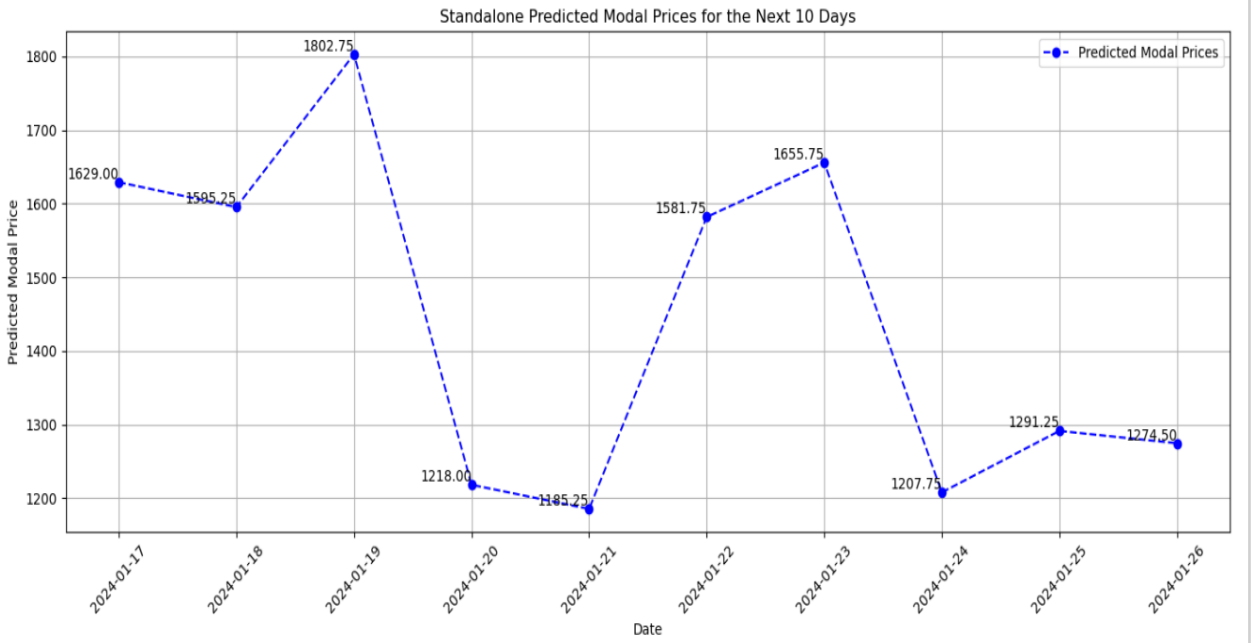


4. SVR- Historical+Predicted values Graph

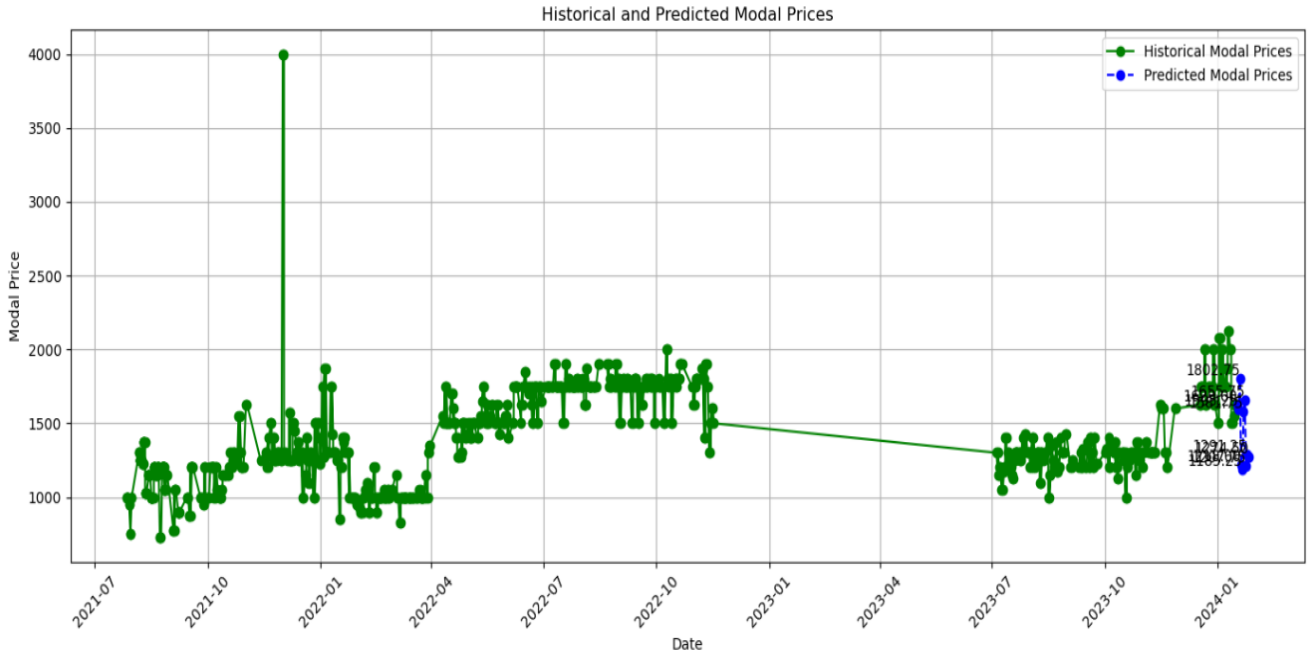


5. RandomForestRegressor Predicted values

RESTRICTED



6. RandomForestRegressor Historical+Predicted values Graph



RESTRICTED

Neural-Networks Trained on a Filtered Dataset

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***).

RESTRICTED

   - *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.

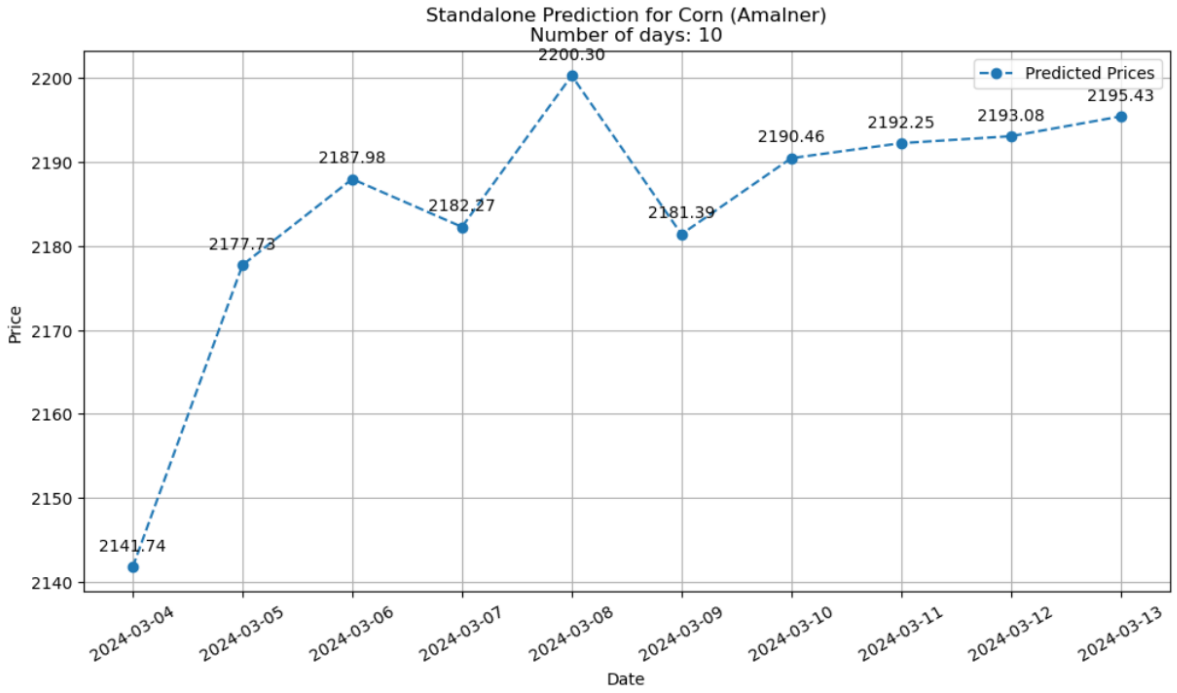
Challenges and Improvements

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.

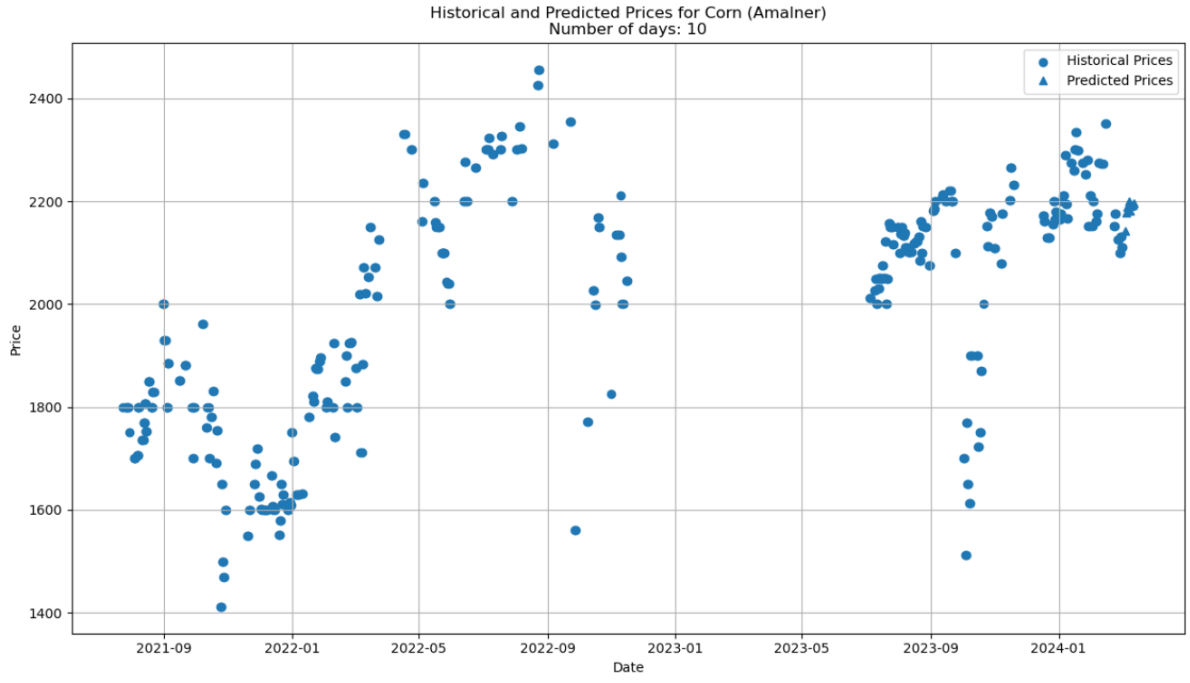
RESTRICTED

Output:

Standalone –



Historical-



RESTRICTED

Accuracy of models

Evaluation of Crop Price Prediction Models

Introduction

Accurate crop price prediction is crucial for stakeholders in agriculture. This study evaluates the performance of a Neural Network (NN) model and a Machine Learning (ML) model against real-time data for predicting the prices of onion, corn, and potato.

Dataset

The dataset *'Final_Accuracy.csv'* includes:

- Crop: Type of crop (onion, corn, potato)
- Date: Date of the data point
- Real-Time: Observed real-time price
- NN: Predicted price using the NN model
- ML: Predicted price using the ML model

Here is the DATASET-Final Accuracy

Data Preprocessing

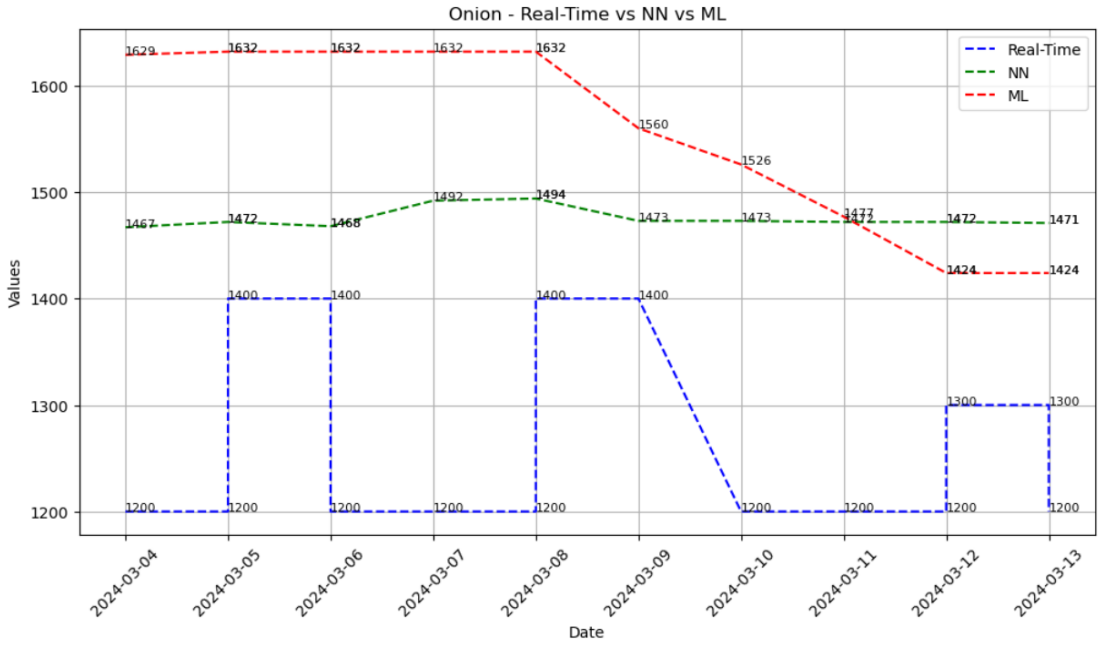
We converted the 'Date' column to datetime format and handled non-finite values in 'Real-Time', 'NN', and 'ML' by filling them with zeros before converting these columns to integers.

RESTRICTED

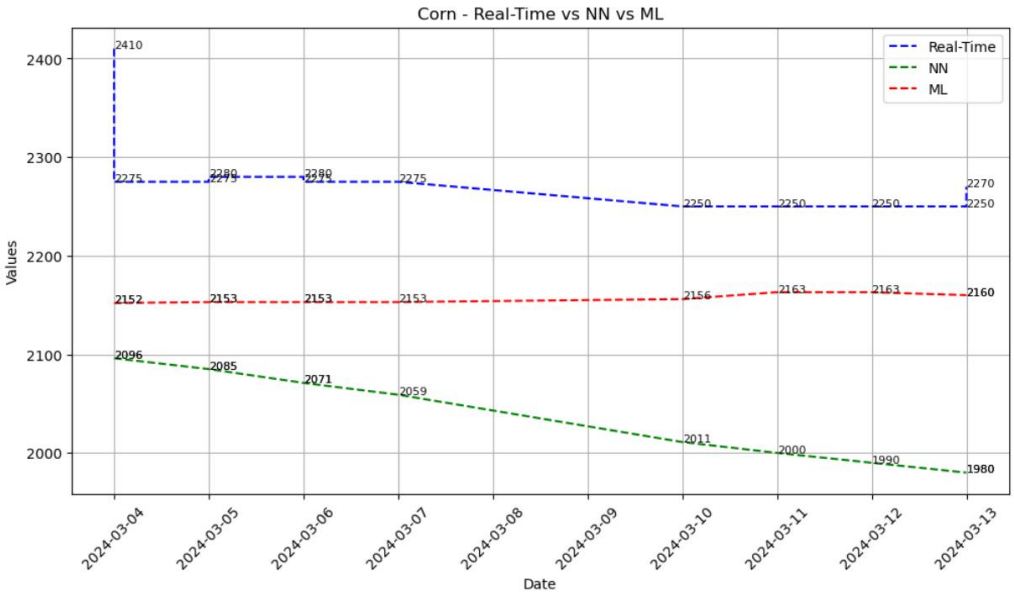
Visualization

Line plots were created to compare real-time prices with NN and ML predictions for each crop, including labeled data points and legends.

A. Onion

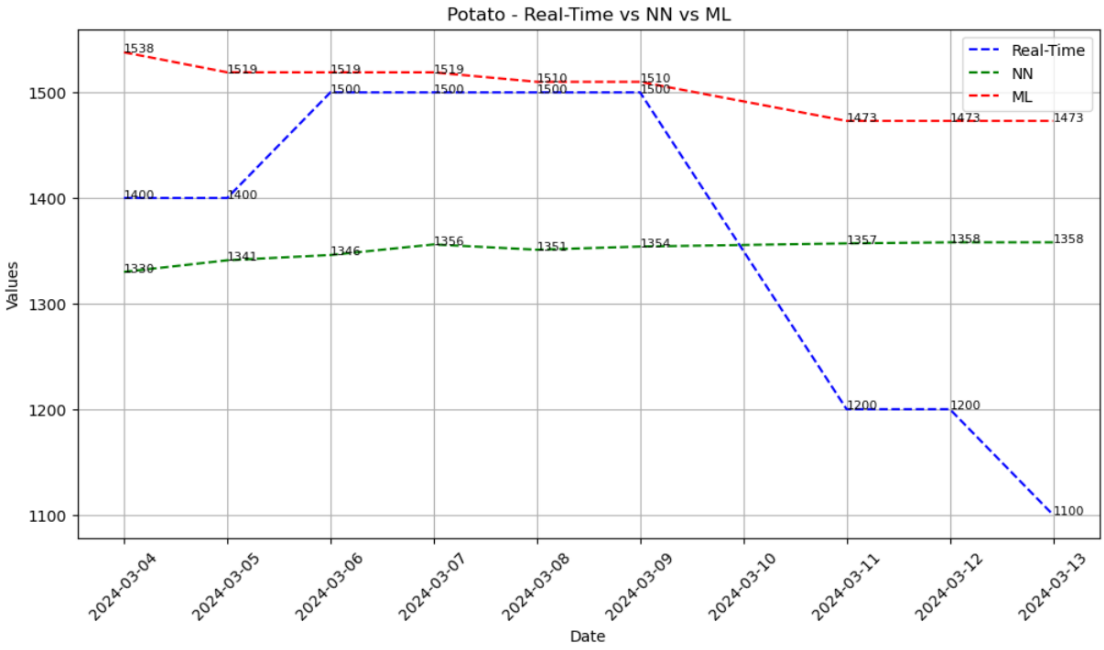


B. Corn



RESTRICTED

C. Potato



Results

The NN model consistently outperformed the ML model in accuracy for all three crops:

- Onion: NN closely follows real-time data; ML shows slight deviations.
- Corn: Both models perform well; NN has a marginally better fit.
- Potato: NN shows superior accuracy; ML has larger deviations.

Conclusion

The NN model demonstrates higher accuracy in crop price prediction compared to the ML model, indicating its potential for improving decision-making in agriculture.

RESTRICTED

Future Work

Future research should aim to enhance ML model accuracy, explore additional influencing factors, and expand the analysis to other crops and regions for a comprehensive assessment.

RESTRICTED