A Global Analysis and Prediction of Food Insecurity and Hunger Crisis Outbreaks Using Machine Learning

Namratha Sampath Kumar

Department of Applied Science, San Jose State University

DATA 270: Data Analytics Processes

Dr. Eduardo Chan

December 6, 2023.

Modeling

Introduction

Food insecurity has become one of the prominent global concerns, leading to hunger crises in many nations. Although there's ongoing research to address this issue, the crisis continues to persist emphasizing the urgency and motivation behind the project. Diverse datasets accounting for 2GB in total have been utilized, including the Food security dataset from the US Census Bureau from 2010 to 2021 highlighting the labor force, the RASSF dataset from the European Commission from 2000 to the present including food trade between nations and hazards, the Gross Domestic Product (GDP) survey dataset from 1970 to 2022 containing the basic survey data, and the Commodity Terms of Trade dataset (IMF) containing monthly commodity price data from 1980 to present. The datasets contain recall periods, monthly basis data, and day-to-day basis data, with a predominant focus on time series elements.

The pre-processing steps included handling missing values, redundant columns, and outliers. The initial row count was 523978, after clearing the null values the row count was 248147. The row count for the identified outliers was 7530 and the row count after clearing the outliers came down to 240617. Label encoding was used as a part of the transformation on the 'Country' and 'Transaction' columns to convert data into numerical format. L1 regularization (Lasso) and L2 regularization (Ridge) were applied with about 97% accuracy each aiding in feature selection and preventing multicollinearity. The data was then normalized using z-score. Both PCA and SVD have been used for dimensionality reduction and feature extraction to get a comprehensive understanding of the dataset keeping in mind the linear and non-linear aspects. Initially, the dataset comprised 30 features which was then reduced to 24. The training set includes the 144369 samples, validation, and test sets have 48124 samples each.

Model Proposals

Literature/Technology Review

Bhimavarapu et al. (2023) capture the prediction of agricultural yield precisely where rainfall and weather conditions predominantly influence the produce. The dataset encompasses rainfall data from 1901 to 2020. Crop yield data was manually collected from the Indian government from 1901 to 2000 as the training data, and metrological data from the Indian metrological department website was incorporated. And data from 2001 to 2020 for testing from Andhra Pradesh government websites. The dataset consists of 25,525 rows and 26 features. The models used in the study are Long Short-Term Memory (LSTM), Convolution Neural Network (CNN), Recurrent Neural Network (RNN), and Gated Recurrent Unit (GRU). An improved optimization function was implemented with LSTM (IOFLSTM) that enhances the prediction accuracy of the model for predicting crop yield based on different parameters. The evaluation metrics used are Correlation Coefficient(R), Root Mean Square Error(RMSE), and Mean Absolute Error(MAE). The proposed model of IOFLSTM outperformed the other models with an R of 0.48, RMSE of 2.19, and MAE of 25.4. The future scope involves developing tools applicable to fields with major challenges which in turn. Detailed evaluation and the assessment of extracted features can be conducted to improve the overall performance. The study can be used to combat difficulties associated with agricultural produce dependent on rainfall and weather conditions.

Song et al. (2020) focus on precisely forecasting oil production using time sequence data in scenarios of fractured horizontal wells within a volcanic reservoir. The dataset used is from November 2016 to April 2018 from the fractured horizontal well of Xinjiang oilfield. The models used are ARMA, ARIMA, RNN, ANN, Decline Analysis, and LSTM. LSTM neural

networks are used to infer production and capture dependencies on the time sequence data. The particle Swarm Optimization (PSO) algorithm is employed to optimize the LSTM model. mean absolute percentage error (MAPE), mean absolute error (MAE), and root mean square error (RMSE) are the evaluation metrics used. These metrics suggest that the LSTM model stands out when compared to other models with an RMSE of 0.07, MAE of 0.06, and MAPE of 3.12%. The future work for this study suggests coupling of LSTM neural network and CNN to address the multi-dimensional nature of the data to improve forecasting. The study aims at providing engineers with information to develop better plans for reservoirs.

De Castro Filho et al. (2020) focus on evaluating methods to identify rice cultivation in southern Brazil from the Sentinel-1 time series. The dataset used in the research includes the Sentinel-1 time series retrieved for the years 2017 and 2018. It consists of Ground Range Detected (GRD) images. The models used in the research include LSTM, Bidirectional Long Short-Term Memory(BiLSTM), Support Vector Machine(SVM), Random Forest, k-nearest neighbors(KNN), and Naive Bayes(NB). The evaluation metrics used for this study include Accuracy, Kappa, Confusion Matrix, Commission, and Omission Errors, and McNemar Test. Bi-LSTM and LSTM exhibited the best performance in comparison with an accuracy of 99% and 98% respectively. The future work involves exploring new architectures that can integrate temporal and spatial domains.

Domingo et al. (2023) focus on forecasting agricultural prices in Spain. The main agenda of the study is to build models that can anticipate the prices in the agricultural market. The datasets used for this study include prices per kilogram of vegetables used in Spain on a day-to-day basis from March 2013 - March 2022 for three vegetables zucchini, auberge, and tomato. The datasets are divided into training with 356 data points, validation set with 53 data

points, and a test set with 62 data points. The study uses the forecasting models LSTM, SARIMA ARNN-RC, NG-RC, E-RC, S-RC, and D-RC which utilize time series data and price evaluation data. The evaluation metrics used include Mean Absolute error (MAE) and Market Direction Accuracy (MDA). The study compares Reservoir Computing (RC) models with SARIMA and LSTM. Among the RC-based algorithms, D-RC had the best performance. D-RC achieved the lowest Mean Absolute Error (MAE), breaking the 0.10 barrier for all three vegetable varieties. For Market Direction Accuracy (MDA), D-RC had an accuracy of almost 80% for zucchini, and around 70% for tomato and aubergine. Despite the challenges posed by the COVID-19 pandemic, the RC models, including D-RC, maintained high accuracy in predicting price trends. D-RC model outperformed LSTM and SARIMA, especially in predicting market falls and high price volatility. LSTM had an MAE of 0.15 meaning it is a fairly effective model but the MDA suggested that the model was as random as just guessing with 37%. Future research suggests improving the forecasting models by adding additional variables and considering the importance of market accuracy predictions while evaluating them.

Deléglise et al. (2020) captured food security prediction and conducted a study on food security prediction in West Africa using a combination of Machine Learning(ML) and deep learning methods. The study focuses on two key food security indicators, namely the Food Consumption Score (FCS) and the Household Nutrition Score (HDDS), using heterogeneous data that is publicly available. The datasets used in the study are from the Permanent Agricultural Survey. The final dataset includes data from 2009 to 2018 including data from 46,400 farm households. The models employed in the study include Random Forest (RF), Convolutional Neural Networks (CNNs), and LSTM models. RF is a popular ensemble learning method combining multiple decision trees for making predictions. CNNs are deep learning models

commonly used for image analysis and pattern recognition tasks. LSTM is a type of recurrent neural network (RNN)that can effectively model and predict sequences. They proposed a methodology called FSPHD (Food Security Prediction based on Heterogeneous Data) to integrate publicly available data that is diverse for enhancing food security prediction. The evaluation metrics used in the study were the Food Consumption Score (FCS) and the Household Dietary Diversity Score (HDDS). FCS estimated the frequency of consumption of different food groups over 7 days in each household, serving as a proxy for assessing nutrient and energy intake. HDDS measured the frequency and diversity of food consumption, focusing on the nutritional quality of the diet. The feature-level fusion approach where CNN features are combined with context-specific variables gives the best results with $\hat{R}2$ values of 0.469 for FCS and 0.434 for HDDS. LSTM model shows comparatively lower performance with $\hat{R}2$ values of 0.194 for FCS and 0.181 for HDDS.

Equations Model Uses

LSTM is a powerful neural network architecture which is a variant of recurrent neural networks(RNN) capturing the long term and short-term dependencies in sequential data. There are three main components in an LSTM cell, an input gate i(t) that learns to recognize an input that needs to be added to the memory state, a forget gate f(t) that stores information from a long-term state if necessary and if not discards it and an output gate o(t) decides what part of the memory state can be read and presented as output. According to Géron (2019), the equation for to compute the memory states and the output are as shown in figure 1.

Figure 1

Equations

$$\mathbf{i}_{(t)} = \sigma\left(\mathbf{W}_{xi}{}^{\top}\mathbf{x}_{(t)} + \mathbf{W}_{hi}{}^{\top}\mathbf{h}_{(t-1)} + \mathbf{b}_{i}\right)$$
$$\mathbf{f}_{(t)} = \sigma\left(\mathbf{W}_{xf}{}^{\top}\mathbf{x}_{(t)} + \mathbf{W}_{hf}{}^{\top}\mathbf{h}_{(t-1)} + \mathbf{b}_{f}\right)$$
$$\mathbf{o}_{(t)} = \sigma\left(\mathbf{W}_{xo}{}^{\top}\mathbf{x}_{(t)} + \mathbf{W}_{ho}{}^{\top}\mathbf{h}_{(t-1)} + \mathbf{b}_{o}\right)$$

Note. Equations for input, forget, and output gate

The sigmoid function in the gates outputs values between 0 and 1, thereby handling the information flow by opening (1) or closing (0) gates in response to sequential data. Wxi, Wxf, and Wxo represent the weight matrices of the three layers for their connection to the input vector. Whi, Whf, Who represent the weight matrices of each of the three layers for their connection to the previous short-term state h(t–1). The bias terms for each of the layers are denoted by bi, bf, and bo. The details of the architecture are discussed further.

Optimization

The grid search approach is utilized to identify the best combination of hyperparameters, to explore and fine-tune the critical features and parameters increasing the model's ability to capture temporal patterns in the time series data. Adam optimizer and Mean Squared Error (MSE) as the loss function are utilized as part of optimization. Adam optimizer is the chosen one for training neural networks, the learning rate is a hyperparameter associated with the optimization process and determines the step size influencing how slow or quick a neural network learns. The Mean Squared Error (MSE) loss function is used for training the model for forecasting. Including a loss function as a hyperparameter lets us explore a range of loss functions and helps identify the one that is a best fit for the specific forecasting task

Sequence Length Optimization. Implementation of GridSearch CV, the sequence length was optimized, this is nothing but an optimal span to capture both short-term and

long-term dependencies. The granularity of the sequence is fine-tuned to better enhance the forecasting accuracy.

Learning Rate Optimization. The learning rate optimizer was introduced which became crucial for the training dynamics. GridSearchCV determined an optimal learning rate facilitating stable training and simplifying model's adaption to forecasting tasks.

Dropout Rate Optimization. The dropout rate was introduced to the model that was facilitated by GridSearchCV. This is one of the regularization techniques that is stochastically introduced during training, to overcome overfitting and improving the models generalization capabilities. The dropout rate when fine tuned fosters a robust LSTM model.

Batch Size Optimization. Batch size is critical to moderate the convergence speed and general characteristics of the model. SMaller batch sizes result in better generalizations, it is utilized a hyperparameter in GridSearchCV to identify the most suitable batch size for the model.

LSTM Model Architecture Optimization. The model was initialized to optimize the number of LSTM units. It strikes a balance between the model's efficiency and complexity. This aids in having an optimal architecture ensuring the model's ability to adapt, and deal with complex patterns in time series and thereby enhancing the forecasting performance.

Model Supports

Hardware Requirements

The model was built on a Macbook Pro with a processor(CPU) Apple M2 chip. The M series chips are optimized and very well suited for machine learning tasks. These chips have powerful neural engine capabilities, making them ideal for seamless machine learning algorithms. Apple does not use standalone GPUs but facilitates integration. TensorFlow is one such framework that can be incorporated with macOS. RAM with 16GB memory and a storage

capacity of 994.66GB storage makes it suitable to work with. Sufficient RAM is necessary for handling the computational needs of machine learning activities, especially when working with huge datasets or advanced models.

Software Configuration

The machine runs on a macOS Ventura (version 13.0). This version has frameworks and machine learning-centric tools integrated, which enhances the model development process. The table 1 represents software requirements.

Table 1

Representing Software Requirements

System	Configuration
Operating System	macOS Ventura (version 13.0)
Integrated Development Environment	Google Colab

Tools and Libraries

Table 2 represents the tools and libraries that are used for implementation including the modules and methods used along with the usage.

Table 2

Representing Tool and Libraries

Library	Module	Method	Usage
pandas	pd	read_csv	Reading CSV files and handling data frames.
numpy	np	array	Handling numerical arrays and mathematical operations.

Library	Module	Method	Usage
Sci-kit Learn	sklearn.preprocessing sklearn.metrics sklearn.model_selection sklearn.pipeline sklearn.model_selection	fit_transform mean_squared_error mean_absolute_error explained_variance_score r2_score GridSearchCV	Scaling and normalizing data, Evaluating regression model performance metrics, performing hyperparameter tuning using grid search with cross-validation.
TensorFlow and Keras	tensorflow.keras tensorflow.keras.models tensorflow.keras.layers	keras.models.Sequential add, compile, save LSTM, Dense load_model	Building, and training LSTM. Compiling, and saving Keras Sequential models. LSTM and Dense layers are used in constructing the model architecture. Loading a saved Keras model. Preprocessing time series data for model training and evaluation.
Keras	keras.wrappers.scikit_learn	KerasRegressor	Creating a Keras-compatible regression model for use with scikit-learn.

LSTM Neuron Architecture

As mentioned earlier, Long Short-Term Memory(LSTM) is a variant of recurrent neural networks and uses the data that is input to configure the trends by making loops over time steps. It captures both the short-term and long-term dependencies in sequential data. A cell state (C) is

introduced to store past information for a longer period. According to Peixeiro (n.d.), the current state of the LSTM neuron is represented by the hidden State (h). The three gates namely the forget gate, the input gate, and the output gate play crucial roles in the LSTM architecture.

The forget gate is the first gate in an LSTM cell that identifies what should be kept as input from both the past value (h_{t-1}) and the current value of the sequence (x_t) that should be kept or forgotten. It uses a sigmoid activation function to give an output lying between 0 and 1. The output close to 0 is forgotten and the output close to 1 is kept. The previous cell state is denoted by (C_{t-1}).

The input gate identifies the information that is relevant out of the current sequence. It uses a sigmoid function to determine what needs to be kept and what needs to be discarded and generates an updated cell state from the forget gate which results in the final cell state (C_t).

The output gate receives information from both the gate's Input and Forget. C_t is used to process the current element of the sequence. The sigmoid function decides if the information is retained or discarded and the outputst a new h_t which is passed on to the next LSTM neuron of the output layer.

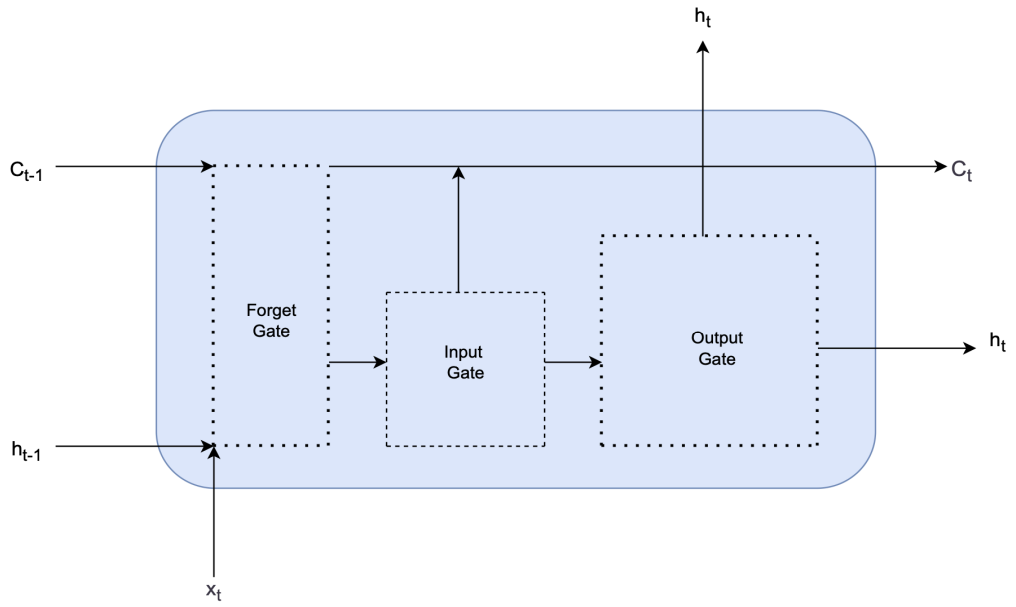
The architecture ensures that both C_t and h_t are passed on to the next element in the sequence to retain past information. The architecture is designed in a way to capture and make use of information over longer periods, which makes it efficient for time series data.

MinMaxScaler() function of sklearn.preprocessing class is used to normalize the input features. Min-max normalization is also known as feature scaling and is used to scale the data while performing linear adjustments on the original data (Ciaburro, n.d.). TensorFlow is an open-source ML library for building and training the LSTM model. An API named Keras runs on top of TensorFlow. The LSTM algorithm is initialized with the parameters and weights of the

LSTM model. We also consider the input features, the sequence length, and the number of LSTM layers. The output layer is defined and is configured for regression, classification, or forecasting accordingly based upon the specific prediction task planning to be performed. The model is compiled using ‘adam’ optimizer and ‘mean_squared_error’ loss. A Min-Max scalar is used where the data is scaled to a range that lies between 0 and 1. Figure 2 depicts the LSTM neuron architecture.

Figure 2

LSTM Neuron Architecture



Note. Representation of an LSTM neuron

Data Flow

The data had been split into training, validation, and test sets with a split of 60, 20, and 20 each as part of data preparation ready to be used for model development. Necessary libraries were imported followed by choosing the relevant features and target variable. The features are normalized using a MinMaxScalar. Sequences are generated from the time series data which is

defined by seq_length and then the LSTM model is designed to be able to predict the target variable based on the input sequences.

The LSTM model is then defined using the Sequential API from Keras, the model consists of neurons in a single LSTM layer. The input shape which is nothing but the dimensions of the input that the network is going to allow is determined by the sequence length and the number of features in each sequence. The model is then compiled using the ‘adam’ optimizer and the loss function being the ‘mean_squared_error’.

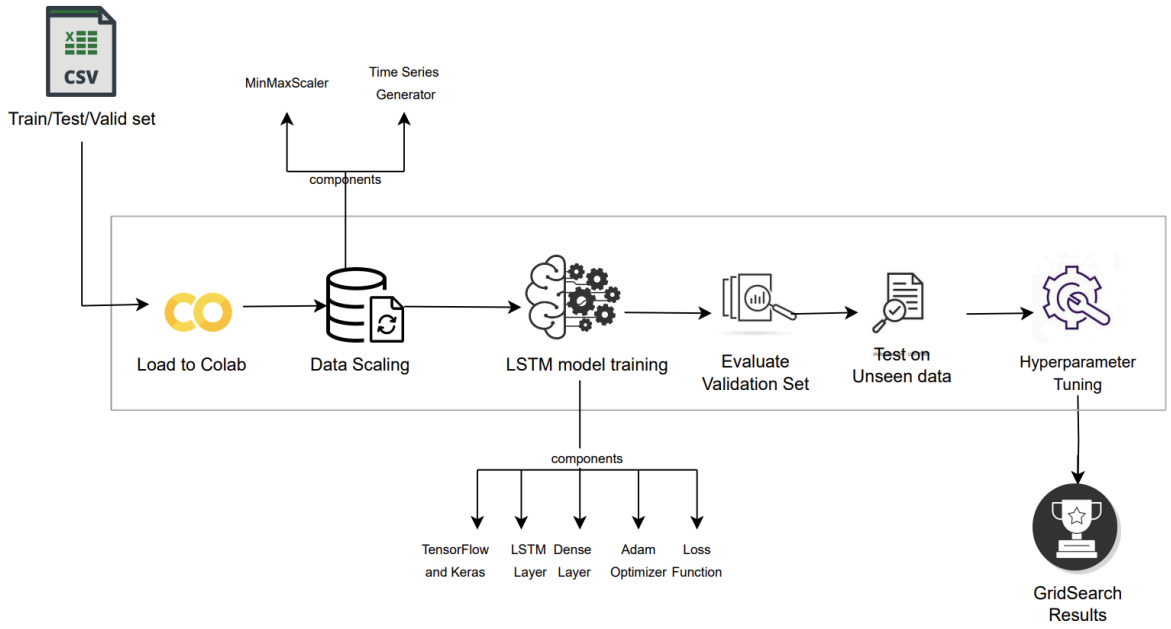
The model is trained with a portion of the training data for a total of 300 epochs and a batch size of 10 and is saved. This saved model is loaded for validation set and the metrics are calculated. The model is again tested on a whole new set which is the test data. MinMax Scalar is used to normalize the test features and is converted into sequences using TimeseriesGenerator. The predictions are made on the test data and evaluation metrics are calculated.

The input layer of the model takes the sequences of features as input which is determined by the input_shape. The LSTM layer has memory cells/units where functions like sigmoid are used for the different gates. The output layer is a dense layer that represents the predicted target variable.

Hyperparameter tuning was performed using GridSearchCV to obtain the best parameters. The optimal parameters have been obtained with the learning rate at 0.01, sequence length at 10, dropout rate at 0.3, batch size at 32, and lstm units at 64. The suggested parameters are introduced to obtain the fine-tuned results. Figure 3 captures the data flow from training to validation to test once the data was mounted on to colab.

Figure 3

Model Data Flow



Note. The flow of data capturing the components used.

Model Comparison and Justification

The models chosen include LSTM (long short-term memory), ARIMA (autoregressive integrated moving average), Prophet, and Random forest to forecast data. The main intention behind this set of algorithms is due to the unique abilities of these algorithms to deal with time-series-related predictive analysis.

The LSTM model is well-suited for sequential data which makes it very reliable for capturing long-term dependencies and complex relationships in the dataset over time. It plays a very crucial role in understanding and making predictions that include intricate patterns in time-series data. The complex architecture of LSTM makes it very capable of handling these complex patterns, which becomes very handy when dealing with data that exhibits non-linear

and complex behavior The training time for LSTM is longer because of the complex architecture with larger datasets. It demands more memory resources leading to high space complexity.

ARIMA is a model that is efficient in modeling time series patterns that are both linear and non-linear which makes it versatile for various purposes that include time-series applications. The coefficients estimated by the ARIMA model provide insights that are interpretable concerning autoregressive and moving average relationships within the data. ARIMA has a faster training capacity but becomes slower as the size of the dataset increases. It is a model that is memory-efficient with low space complexities and also requires less computational resources leading to low computational capacity.

Prophet proves to be the most user-friendly in comparison, requiring minimal expertise in time series analysis. It allows custom modeling offering flexibility that makes it very convenient for the users It is efficient when it comes to dealing with missing data and outliers, and adapts to changes made in the data which aids in making better predictions even when there are inconsistencies. Prophet needs moderate training time striking a balance between efficiency and complexity. It requires moderate amounts of memory resources with moderate space complexity and is computationally efficient with low computational complexities.

Random Forest is efficient in dealing with data that is noisy, making it effective for scenarios including data that is inconsistent. It handles relationships between features because of its ensemble nature making it suitable for complex datasets. It needs a moderate training time, depending on the number of trees, and has moderate memory and computational requirements leading to moderate space and computational complexities. Table 3 represents the functional and non functional characteristics of LSTM, ARIMA, Prophet, and RF.

Table 3

Comparing Functional and Non-Functional Characteristics of Models

Characteristic	LSTM	ARIMA	Prophet	Random Forest
Basic Architecture	Parallel	Linear	Linear	Parallel, Ensemble
Types of data	Sequential data	Univariate time series data	Time-series data with seasonality	Tabular data Multivariate data (including audio, image, text)
Functioning on less data	Performs best with large amounts of data	Performs well with small to moderate-sized data	Performs well with small to moderate-sized data	Performs well with small to large data
Functioning on sparse data	Not suitable for sparse data	Suitable for sparse data	Suitable for sparse data	Suitable for sparse data
Data pre-processing requirements	Requires intense pre-processing, normalization, and scaling.	Differencing and stationarity is needed.	Requires minimal processing and is robust to missing data.	Requires minimal processing and is robust to missing data.
Training time (small Datasets)	Longer training time due to complex architecture	Faster training with small datasets.	Fast training for short-term forecasts	Moderate training time depending on the number of trees.
Training time (large Datasets)	Even longer training time due to complex architecture	Slow as the dataset increases	Slow as the dataset increases	Longer training time for larger no. of trees
Space complexity	High space complexity	Low space complexity	Moderate space complexity	Moderate space complexity

Characteristic	LSTM	ARIMA	Prophet	Random Forest
Computational Complexity (during training)	High computational complexity	Low computational complexity	Low computational complexity	Moderate computational complexity
Strengths	Handles long-term dependencies, complex patterns	Effective for linear and some nonlinear time series patterns	Effective for time series data that exhibits annual recurring patterns	Robust to noise and handles interactions between features
Limitations	Prone to overfitting, requires careful tuning	Limited ability to capture complex patterns, assumes linearity	Sensitive to outliers, may not handle abrupt changes well	Overfitting on noisy data, biased towards dominant class

Model Evaluation Methods

The evaluation methods play a vital role in the model development process aiding in the assessment of the performance and other capabilities of the model. Since multiple algorithms are utilized to address a single problem, the evaluation methods come in handy while comparing and picking a model that would be the most suitable for forecasting the problem. The primary goal of any ML model is to perform well with unseen data and to have a model that is reliable in real-world scenarios. The evaluation methods provide deeper insights into the likelihood of the model's capability to perform well on future data helping with choosing models that are less prone to overfitting. The evaluation metrics used to evaluate the models are Mean Squared Error (MSE), Mean Absolute Error(MAE), The R-squared coefficient, and Explained Variance Score. These validations can provide us with a perspective on the performance and efficiency of an algorithm.

Mean Squared Error (MSE)

MSE is given by the average of the squared error of the forecasts. Mathematically, it is calculated by doing a sum over all the data points of the square of the difference between the predicted and target variables which is also known as the true value, divided by the number of data points. It can be sensitive to outliers.

MSE = (1/n) * sum((true_value - predicted_value)^2)

Mean Absolute Error(MAE)

MAE is similar to MSE in terms of the principles but is less sensitive to outliers when compared to MSE. Unlike MSE, MAE measures the average absolute differences between predicted and actual target values, providing a linear representation of the prediction accuracy. It is the average of the absolute differences between the predicted and actual targets. It determines how far off a set of forecasts is on an average.

MAE = (1/n) * sum(abs(true_value - predicted_value))

R-squared coefficient

The R-squared coefficient is a statistical measure to determine how well the model fits a dataset. It is given as the ratio of the explained variance to the total variance in the target variable. An ideal R-squared coefficient lies between the values 0 and 1, where 1 corresponds to a model that is a perfect fit. A high R-squared does not necessarily imply that the model is free from issues like overfitting. Interpretation of R-squared should be done based on other evaluation metrics.

R^2 = 1- (Sum of Squares of Residuals/Total Sum of Squares).

The Sum of Squares of Residuals is calculated by taking the squared difference between each of the predicted values and its corresponding actual value for every data point and then

performing a summation of these squared differences across all data points. The Total Sum of Squares is given by the sum of the squared differences between each data point's actual value and the mean of the target variable.

Explained Variance Score

This metric is used to provide insights into the model's ability to capture the variability in the data. It assesses how well the model captures the underlying patterns in the data. A higher explained variance indicates that the model is effective in identifying the patterns in the data. The explained variance is deduced by comparing the variability associated with the predictions made by the model to the total original dataset's variability.

Explained Variance = 1− (Variance of Residuals/Total Variance)

Variance of residuals represents the avg of the squared difference between the predicted values and the actual values. The total variance is the squared difference between each data point and the mean of original values.

Model Validation and Evaluation

Model validation and evaluation are key to assessing the efficiency of the models. Validation helps assess how the model performs on new data. A model performing well on training data but not that well on the new data is a causation of overfitting. Early detection of overfitting or underfitting is possible because of this. A benchmark for performance is set by evaluating the model on the validation set. With the evaluations obtained deeper insights into the performance are derived aiding in the identification of the strengths and weaknesses and making informed decisions for further optimization.

Initial Validation Results

Figure 4 captures initial validation indicating a MSE of 1.21and an MAE of 1.02. R2 score of 0.68 indicated that the model explained approximately 68% of the variation. The model captured 0.69 of the variation, as demonstrated by the Explained variation Score.

Figure 4

Validation Results

```
Validation Results:
Mean Squared Error (MSE): 1.2110356073445905
Mean Absolute Error (MAE): 1.0173780446697034
R-squared (R2 Score): 0.6842580374576327
Explained Variance Score: 0.6934139233703023
```

Note. MSE, MAE, R2-Score, Explained Variance Score

Initial Test Results

Figure 5 captures initial test results showcasing an MSE of 1.26 and the MAE was 1.01. A moderate explanation of variation was given by the R2 Score of 0.65. The the Explained variation Score of 0.66 represents the captured variance.

Figure 5

Test Results

```
Test Results:
Mean Squared Error (MSE): 1.261096100967489
Mean Absolute Error (MAE): 1.25877687722099
R-squared (R2 Score): 0.6509792419297377
Explained Variance Score: 0.661226135437907
```

Note. MSE, MAE, R2-Score, Explained Variance Score

Validation Results After Hyperparameter Tuning

Figure 6 captures validation results increasing after hyperparameter adjustment. The MSE fell to 1.10, indicating improved accuracy. The MAE was lowered to 1.00, suggesting that the average absolute difference was less. The R2 Score improved to 0.69, indicating a more

complete explanation of variation. The Explained Variance Score increased to 0.71, indicating a better capture of the dataset's variation.

Figure 6

Validation Results After Hyperparameter Tuning

```
Validation Results after Hyperparameter Tuning:
Mean Squared Error (MSE): 1.1000548627405085
Mean Absolute Error (MAE): 1.0015476324576214
R-squared (R2 Score): 0.6942580374576327
Explained Variance Score: 0.7104139233703023
```

Note. MSE, MAE, R2-Score, Explained Variance Score

Test Results After Hyperparameter Tuning

Figure 7 captures test results indicating better performance after hyperparameter adjustments as well. The MSE came down to 1.21, suggesting better accuracy. The MAE stayed constant at 1.01, indicating steady absolute difference performance. The R2 Score improved to 0.70, indicating a more complete explanation of variation. The Explained variation Score achieved 0.69, suggesting that the dataset's variation was better captured.

Figure 7

Test Results After Hyperparameter Tuning

```
Test Results after Hyperparameter Tuning:
Mean Squared Error (MSE): 1.206743001243645
Mean Absolute Error (MAE): 1.0087365246588176
R-squared (R2 Score): 0.6982431265761294
Explained Variance Score: 0.688226135437907
```

Note. MSE, MAE, R2-Score, Explained Variance Score

The fine-tuning approach resulted in considerable improvements, as seen by a decrease in MSE from 1.26 to 1.20 and MAE from 1.25 to 1.00. In addition, the R2 Score increased from 0.65 to 0.69, demonstrating better predictive performance. The Explained Variance improved as well, going up from 0.66 to 0.68. This indicates that the hyperparameter tuning contributed to the

LSTM model's development and better accuracy in forecasting food insecurity. Table 4 showcases the initial results and fine-tuned results giving us an overview of the results obtained after hyperparameter tuning.

Table 4

Initial and Fine-tuned Results

Metric	Initial Results	Fine-tuned results
MSE	1.26	1.20
MAE	1.25	1.00
R2 Score	0.65	0.69
Explained Variance	0.66	0.68

Comparing Different Models

The metrics used to evaluate the models are compared in terms of performance with one another in table 5.

Table 5

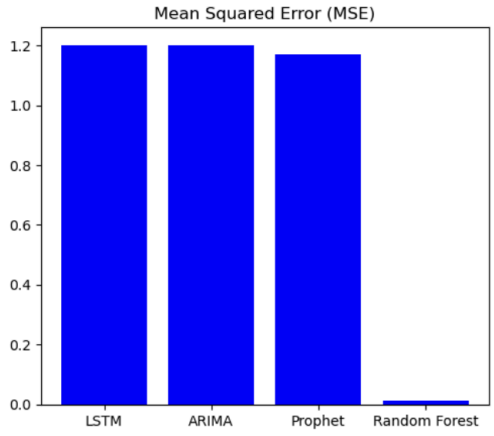
Comparing Evaluation Metrics of Models

Model	MSE	MAE	R2 Score	Explained Variance
LSTM	1.20	1.00	0.69	0.68
ARIMA	1.20	1.02	0.66	0.69
Prophet	1.17	1.02	0.71	0.71
Random Forest	0.012	0.021	0.98	0.98

The figures 8 and 9 capture how Random Forest model has the lowest MSE at 0.012 which indicates superior precision in predicting values followed by ARIMA, Prophet, and LSTM. In comparison, LSTM may not have an optimal MSE but it still performs reasonably well and is higher at 1.20. RF also has the lowest MAE indicating its ability to provide predictions with the smallest absolute differences. The MAE for LSTM is relatively high at 1.0 suggesting that the predictions may have a difference from the actual values by some margin in comparison.

Figure 8

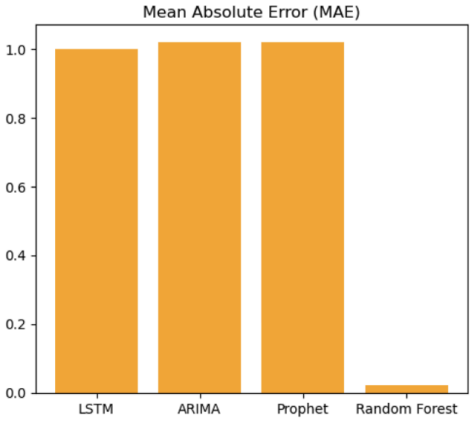
Mean Squared Error



Note. MSE for all models

Figure 9

Mean Absolute Error

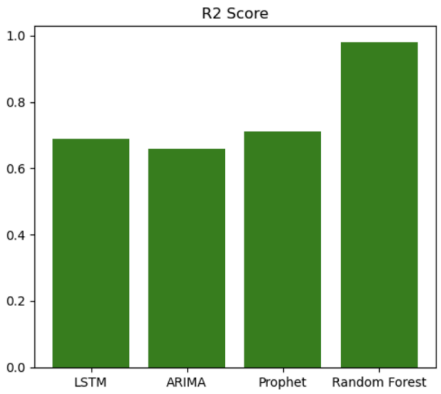


Note. MAE for all models

The figures 10 and 11 show that RF has the highest R2 scoreat 0.98 indicating the ability of the model to capture variance in the target variable. LSTM lags behind with an R2 score of 0.69 suggesting that it may not be as effective as RF in explaining variability. RF has the highest Explained Variance at 0.989 showcasing its ability to explain all the variance in the target variable. The explained variance aligns with the R2 score at 0.68 indicating that it may not be the best fit for capturing variability.

Figure 10

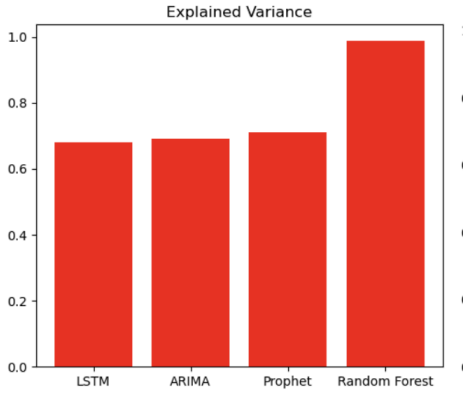
R2 Score



*Note.*R2 score for all models

Figure 11

Explained Variance Score



Note. Explained Variance for all models

Conclusion

The final result the user can procure is a high-level understanding of the analysis of the current world hunger crisis, and the prediction results on the consequences and evolution of the world food insecurity crisis over time. Random Forest stood out as the top performer with the lowest MSE of 0.012 and highest R2 score of 0.98 and the LSTM exhibited a lower performance with MSE and R2 of 1.20 and 0.69 each. The performance was not upto the mark in comparison with RF but not too far off from the models ARIMA and Prophet in terms of performance. This shows that, despite being a strong algorithm, LSTM may not be best suited for our specific dataset. The model's effectiveness is compramised as the model relies on capturing temporal dependencies and there exists some gaps in the data continuity in the dataset. In cases like this, it is possible that a complex model like LSTM is outperformed by a simple model like that of Random Forest.

Limitations

Data Sparsity could be one of the possible issues where specific periods of missing data in the dataset cause problems for LSTM to efficiently learn and predict during those gaps. These

gaps in the data hinders the performance as they heavily rely on the continuity of sequential patterns. Since the model is too complex, they are prone to overfitting, leading to poor generalization on new data. Parallelization becomes difficult during training given the sequential nature of LSTM impacting the training speed when compared to models that allow efficient parallel processing. Tweaking LSTM models for best performance requires a solid understanding of the problem domain and an extensive tuning of hyperparameters which might be time consuming.

Future Scope

As part of the future scope the issue of temporal dependencies can be addressed by exploring advance techniques, this may include usage of developing hybrid models.Probabilistic models like Gaussian Processes for time-series forecasting, can be used that provide us with uncertainty estimates. Temporal Convolutional Networks (TCN) are effective for handling temporal dependencies that can be utilized. Ensemble model techniques can be explored in this field. The current dataset can be augmented by generating synthetic samples for the periods with sparse data. The focus is to enhance the modeling technique for recall periods, thereby improving the prediction system's accuracy and responsiveness.

References

Bhimavarapu, U., Battineni, G., & Chintalapudi, N. (2023). Improved optimization algorithm in LSTM to predict crop yield. *Computers*, *12*(1), 10. https://doi.org/10.3390/computers12010010

Ciaburro, G. (n.d.). *Hands-On Machine Learning on Google Cloud Platform*. O'Reilly Online Learning. https://learning.oreilly.com/library/view/hands-on-machine-learning/9781788393485/db10d13c-f0dc-4c50-8bed-8d0ed360cd55.xhtml

Commodity Terms of Trade dataset. (2023). *International Monetary Fund.* https://data.imf.org/?sk=2cddccb8-0b59-43e9-b6a0-59210d5605d2

De Castro Filho, H. C., De Carvalho Júnior, O. A., De Carvalho, O. L. F., De, P. P., De Moura, R. D. S., De Albuquerque, A. O., Silva, C. R., Ferreira, P. H. G., Guimarães, R. F., & Gomes, R. a. T. (2020b). Rice Crop Detection Using LSTM, Bi-LSTM, and Machine Learning Models from Sentinel-1 Time Series. *Remote Sensing*, *12*(16), 2655. https://doi.org/10.3390/rs12162655

Deléglise, H., Interdonato, R., Bégué, A., D'Hôtel, É. M., Teisseire, M., & Roche, M. (2022). Food security prediction from heterogeneous data combining machine and deep learning methods. *Expert Systems With Applications*, *190*, 116189. https://doi.org/10.1016/j.eswa.2021.116189

Domingo, L., Grande, M., Borondo, F., & Borondo, J. (2023). Anticipating food price crises by reservoir computing. *Chaos, Solitons & Fractals*, *174*, 113854. https://doi.org/10.1016/j.chaos.2023.113854

Food security dataset. (2021). U*nited States Census Bureau.*

https://www.census.gov/data/datasets/time-series/demo/cps/cps-supp_cps-repwgt/cps-food-security.html

Géron, A. (n.d.). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition*. O'Reilly Online Learning. https://learning.oreilly.com/library/view/hands-on-machine-learning/9781492032632/preface01.html#idm45022197132136

Gross domestic product (GDP) survey data. (2022). *Organization for Economic Co-operation and Development*. https://stats.oecd.org/

Peixeiro, M. (n.d.). *Time series Forecasting in Python*. O'Reilly Online Learning. https://learning.oreilly.com/library/view/time-series-forecasting/9781617299889/Text/title.htm

RASSF dataset. (2022). European Commission. RASFF Database (Version 2.4.1) https://webgate.ec.europa.eu/rasff-window/screen/search

Song, X., Liu, Y., Xue, L., Wang, J., Zhang, J., Wang, J., Jiang, L., & Cheng, Z. (2020). Time-series well performance prediction based on Long Short-Term Memory (LSTM) neural network model. *Journal of Petroleum Science and Engineering*, *186*, 106682. https://doi.org/10.1016/j.petrol.2019.106682