Advanced Programming 2025

Forecasting Gold Prices Using Random Forest Regressor and Long Short-Term Memory Models

Final Project Report

Amalia Naqib
Amalia.Naqib@unil.ch
Student ID: 22323356

January 11, 2026

Abstract

This project investigates the medium-term forecasting of gold prices using both machine learning and deep learning models driven by financial market indicators. The aim of the paper is to predict daily gold prices 63 days ahead using historical gold prices and four explanatory variables: USD Index, S&P 500 index, Crude Oil prices (WTI), and the CBOE Volatility Index. A naive persistence forecast model is used as a baseline, and will be compared to a Random Forest Regression Algorithm (RF) model and a Long Short-Term Memory (LSTM) model. Hyperparameter tuning for both the RF and LSTM was performed, using a Random Search and a Grid Search respectively. The results show that the LSTM model achieved the best performance across all the metrics, with an RMSE of 94.69 and MAE of 78.28, demonstrating the advantage of sequence based deep learning models in learning medium-term temporal patterns.

Keywords: Data science, Python, time-series forecasting, gold price prediction, financial indicators, machine learning, deep learning, feature engineering, lagged features, Naive Forecast, Random Forest Regression (RF), Long Short-Term Memory (LSTM), hyperparameter tuning, random search, grid search

Contents

1 Introduction

Gold is considered as a safe investment and as a hedge against inflation and tail risks, since it is able to hold or even increase its value, provide liquidity and deliver long-term returns, unlike fiat currencies. Serving as a financial asset, demand for gold surges at times of economic uncertainty, caused by crises, geopolitical tensions and stock market volatility, pushing its price up (Jarvis G, 2025).

Central banks around the world are currently increasing their reserves of gold bullions, representing 20% of global demand with the remaining 80% driven by individual investors, jewellery manufacturers, and medical, industrial and technological applications, due to its durability, malleability and conductivity. (Bruggen A et al, 2025). Gen Z, the generation born between 1997 and 2012, is showing a great interest in gold when it comes to diversifying their investment portfolio, especially as digital platforms are enabling ease of access and purchase (Saputra I, 2025). Gold price has more than doubled in the last three years, from around USD 1,900 per ounce in late 2022 to USD4,400 in late 2025 and is expected to increase to USD 5,000 in 2026 (JP Morgan, 2025).

Forecasting gold price trends are crucial for better-informed asset management and investment strategies by the different stakeholders. Predicting gold price behaviour presents a challenge as it is influenced by various financial, macroeconomic, political and social factors (Taneva-Angelova G et al, 2025).

The goal of this project is to forecast spot gold prices, based on historical data and external financial indicators and comparing the performance of the two methods in uncovering trends. The models selected in this study are the Random Forest regressor machine learning algorithm (RF) and a Long Short Term Memory (LSTM) deep learning model, as they are known for handling large datasets with multiple features and can capture non-linear relationships between variables in a financial time series. They are favored over traditional statistical forecasting methods such as AutoRegressive Integrated Moving Average (ARIMA) which have limitations in dealing with the noisy non-linear and non-stationary behaviour of financial time series data such as gold price data (Gong W 2024, Wang W 2024).

This study will explore whether Random Forest or LSTM will perform better in modelling gold prices in the medium term, in light of financial indicators. The expectation is that the LSTM model will outperform the Random Forest one as it should handle the long sequential, time-dependent structure better.

2 Literature Review / Related Work

Guo (2024) used a LSTM network to predict the future price of gold based on a 10-year data sample of gold prices, including the closing, opening, highest, lowest prices, the volume and daily percentage change. His model performed well, although he recommends that adding more quantitative indicators reflecting the global political and economic climate would allow the model to generalize better. Nagata et al. (2024) also found that LSTM can predict gold price movements effectively, especially on daily and weekly timeframes.

Chen (2025) did a comparative study of LSTM and Random forest models and concluded that both are both adequate in predicting Chinese gold prices, however the LSTM performed better as it captured temporal patterns. In contrast, Pan (2024) found that Random Forest had a better predictive accuracy than LSTM when forecasting the price of gold in the long term. However, his study had limitations as it only used historical data without other predictors, and the LSTM model might have been optimized better with further hyperameter tuning. Chai et al. (2021) studied the dynamic relationships between gold price returns, and found that Crude

oil and VIX are positively correlated, while the U.S dollar index is negatively correlated with gold price returns.

Cohen and Aiche (2023) used the machine learning models Random Forest (RF), Gradient Boosted Decision Trees (GBDT) and Extreme Gradient Boosting (XGBoost) to predict future gold prices with financial indicators. Lagged features, up to 10 days, of global stock indices with S&P500 and VIX index, as well as key commodity futures and 10-year bond yields were used to improve prediction. The study found that a one-day lagged VIX was the most significant indicator in predicting gold prices, meaning economic uncertainty drives up the demand for gold. Overall, GBDT and XGBoost were seen as the most valuable for gold investment decision making.

The paper by Zhang (2024) used the machine learning models Random Forest (RF), Extreme Gradient Boosting (XGBoost) and Support Vector Regression (SVR) to forecast gold returns. Oil prices, S&P500, USD index and volatility index features were implemented to improve the predictions of the models. The grid search method was applied to find the best parameters for the models. Overall, SVR and Random Forest had the best performance.

Chew, Yi, and Ong (2023) conducted a Bidirectional LSTM considering SPX500, USD Index, Crude Oil Prices and CPI as factors to forecast gold prices. A rising CPI signals inflation, increasing the attractiveness of gold and hence its price. They used a random search tuner to optimize their model and identify the best hyperparameters. Using the MSE, MAW, RMSE, MAPE and Rsquared evaluation metrics, they found that their deep learning model could provide valuable insights and improve the accuracy of predicting gold prices.

3 Methodology

3.1 Data Description

The international benchmark for physical gold trading is the spot or current market price of one troy of 24-karat gold, which is set and traded in US dollars across major exchanges like the Commodity Exchange Inc COMEX in New York or the London Bullion Market Association LBMA (www.goldavenue.com).

The time series dataset variables are the following: The target variable of interest is the daily spot gold price, in US dollars per ounce of gold (XAUUSD). The four financial indicators as explanatory variables are the daily USD Index (DXY), daily Crude Oil Prices: West Texas Intermediate (WTI), daily S&P 500 (SP500) and daily CBOE Volatility Index: VIX (VIX).

The data sources are Investing.com for XAUUSD and DXY and FRED.com for WTI, SP500 and VIX. The closing prices from the data above will be used. All the data will span over a 10 year period, specifically from the start of January 2014 to January 2024, creating a sample size of around 2600 raw observations for each daily variable, considering a financial week of 5 days, resulting in a total of around 13,000 observations for the entire data set.

The relationship between the target variable and the financial variables chosen is as follows: The USD Index measures the U.S dollar's strength relative to a basket of six other key world currencies. When the U.S dollar strengthens, gold becomes more expensive to non-U.S buyers, which reduces its demand thereby reducing gold prices. Conversely, when the U.S dollar weakens, gold becomes cheaper and more attractive for investment than purchasing interest-bearing assets, meaning the price of gold increases from higher demand. Hence, gold and USD index have an inverse relationship. Crude oil reflects similar commodity market conditions and often has similar price movements with gold prices, given that rising oil prices lead to higher inflation, increasing the demand for gold as a hedge against it. The relationship between the S&P 500 Index and

gold is complex and dynamic and can be either an inverse one in times of high market volatility or economic crisis, or a positive one when they are both driven by high inflation or a weakening USD. The Volatility index (VIX) reflects market uncertainty, and moves in the same direction as gold. When VIX increases, signalling uncertainty, the demand for gold increases due to its safe-haven status, causing prices to increase. When VIX decreases, signalling confidence, the demand for gold and hence prices decrease. A correlation matrix, illustrated as a heatmap, of the gold variable and the financial indicators to further visualise the relationships between each other can be found in Appendix A as Figure A1.

Data quality issues may arise due to the characteristics of a typical macroeconomic and financial time series, including strong temporal dependence and non-stationary price levels. Moreover, differences in data availability across the data sets may result in missing observations and varying market holidays. These issues highlight the need for temporal alignment and suitable indicator transformations to support the time series analysis, which will be addressed in the data preprocessing stage in Section 3.2 below.

3.2 Approach

3.2.1 Algorithms used and Model Architectures

The aim of the study is to forecast gold prices (XAUUSD) with financial indicators over a fixed horizon, using a combination of a baseline, machine learning and deep learning model. The forecasting horizon chosen will be medium-term, i.e. between 30 and 90 days, and has been set for the next financial quarter, with H=63, which represents 3 financial months totalling to 63 days. For temporal context, a fixed lookback window of also 63 days, with T=63, will be used and adapted for all the models.

For the algorithms used, the naive forecast will be used as a baseline/benchmark model, where only historical data is applied, the Random Forest Regression Algorithm (RF) will be used as a machine learning model and a Long short-term memory (LSTM) will be used as a deep learning model.

Naive Forecast

A naive persistence model will be implemented as a baseline for performance comparison. The naive predicts each future price observation the same as the last observed value within the input window. Let yt represent the gold price at time t. Given the input window T of length 63 and the forecast horizon H of length 63, the true target is defined as $y_{t+T+H-1} = y_{t+T-1}$.
The naive forecast model is structured as follows:

$$y_{t+T+H-1} = y_{t+T-1}.$$

This equation corresponds to a random walk with no drift, and is a strong baseline for a financial time series, where price levels often have high persistence. This naive model is temporally aligned with the LSTM and RF setup, ensuring a fair and consistent comparison across all the models.

Random Forest Regression (RF)

The Random Forest (RF) Regressor Algorithm is an ensemble supervised machine learning model, and is a combination of multiple decision trees which are trained on different randomly sampled subsets of the data. The latter are created by a step in RF called Bootstrap Aggregating or Bagging The final output is determined by the average of all the predictions achieved by the individual trees. Another step is feature bagging which assigns a unique randomly selected subset of features to each tree. This machine learning model is a good choice for financial datasets as it handles multiple features, it can capture non-linear relationships between variables and it is also known to be robust against noise and overfitting.

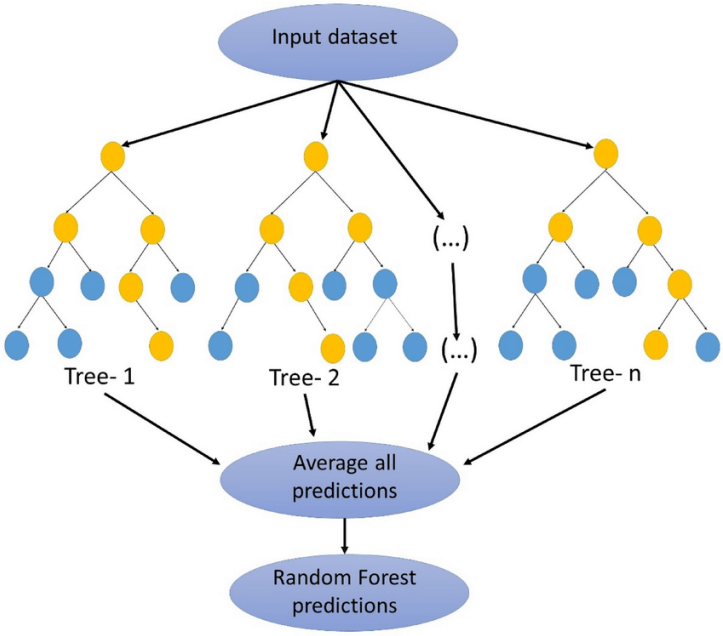

Figure 1: Diagram of Random Forest Regressor (Sahour H et al, 2021)

If there are B trees in the Random Forest and each tree gives a prediction $h_b(x)$, then the final prediction $\hat{y}$ for an input x is:

$$\hat{y} = \frac{1}{B} \sum_{b=1}^{B} h_b(x)$$

Where $h_b(x)$ is the prediction from the b-th tree, B is the total number of trees in the forest, and $\hat{y}$ is the final predicted value (the average of all trees) (Mukherjee S, 2025).

Random Forest Hyperparameter tuning

A random search method, using the RandomizedSearchCV function from the scikit-learn library, will be used to test 20 different hyperparameter combinations (n_iter = 20) and find the best candidate. This method with parallel processing enabled (n_jobs = -1) was chosen because it is more computationally efficient for this model. A time series cross validation (k-fold) strategy will be carried out to evaluate the models performance and prevent overfitting by temporally splitting the dataset into k subsets of equal size, then training on k-1 folds and testing on the last fold. The TimeSeriesSplit function will be used to ensure the training data precedes the validation data in time, to avoid any look-ahead bias. A five-fold time series cross validation (TimeSeriesSplit with cv = 5) will be used for the model. The model will also use bagging with the default bootstrap sampling (bootstrap=True), meaning sampling is done with replacement. The criterion used for splitting the nodes will be the default MSE (criterion='squared_error'). The model performance will be evaluated using the MSE, specifically its negative form as scikit-learn uses a maximum based scoring framework (scoring = "neg_mean_squared_error"), to find the best combination of hyperparameters.

The number of features considered for the best split (max_features) will be fixed to 'sqrt', rather than the default 'auto' for RF regressions, as it reduces overfitting, can handle strongly correlated features, which is needed given the lagged features, and improves generalization for noisy financial time series data.

The following parameters will be varied for hyperparameter tuning. The number of trees (n_estimators) that are built in the forest, which help improve accuracy of the model at the expense of computation time, will range between 200 and 800. The maximum depth of the tree (max_depth), which increases the complexity of the model by capturing more complex patterns at the expense of overfitting, will range between 10 and 30. The minimum number of samples needed to split an internal node (min_samples_split), which is used to prevent overfitting by creating branches with few samples at the expense of learning less and missing complex patterns from the data, will range between 20 and 80. The minimum number of samples required to actually be a leaf node (min_samples_leaf), which is used to reduce overfitting at the expense of more detailed splitting, will range between 10 and 80 (Sharma, 2024).

Long Short-Term Memory (LSTM)

Recurrent neural networks (RNNs) allow sequence learning, as input is captured from the beginning. However with long sequences, vanilla RNNs struggle in practice to connect distant, previous information to the present, as the gradient progressively decays with each time step and this is known as the vanishing gradient problem. Hochreiter & Schmidhuber (1997) introduced the solution to this by designing the Long Short-Term Memory (LSTM) model that is capable of learning long-term dependencies which could be critical to predictions, incorporating memory cells and gates which control the flow of information (Liu Y, 2024).

Similarly to RNNs, LSTMs follow a chain-like structure over time but with a more sophisticated internal memory system. The LSTM consists of two internal memory vectors: the cell state C_t, which stores long-term information, and the hidden state h_t, which represents short-term information and the actual output of the model at each time step. At each time step, the LSTM receives the current input x_t as well as the previous hidden state h_{t-1}, which summarises what has been learned so far.

The internal structure of the LSTM is centred on a memory cell C_t regulated by three information gates: the forget gate f_t, the input gate i_t, and the output gate o_t (Olah, 2015).

Memory Cell: It is at the core of the LSTM unit and it stores information throughout long sequences.

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

Forget Gate: It controls which information is discarded from the memory cell.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

Input Gate: It controls which information is added to the memory cell from the current input or the previous hidden state.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

Output Gate: It controls which information is used to produce the output from the current hidden state.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

All three gates are controlled by a sigmoid function σ, which produces a value between 0 and 1 to determine whether information is removed, added, or revealed. The weights W determine how strongly the inputs h_{t-1} and x_t affect the gate, while the bias b provides the offset. The tanh layer generates candidate memory values $\tilde{C}_t$ that can update the current cell state C_t.

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

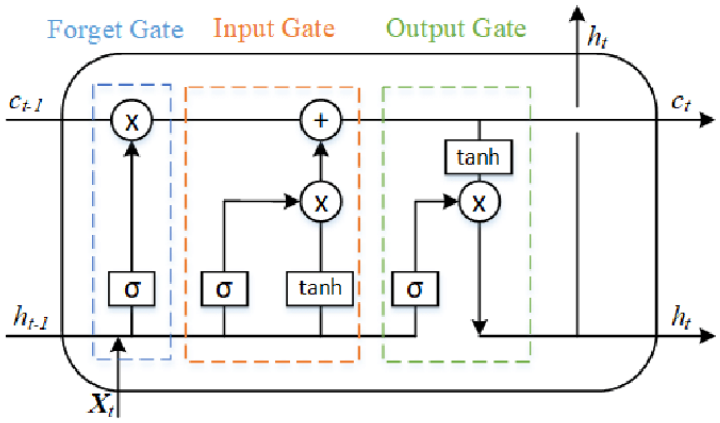

Figure 2: Diagram of LSTM cell (Mittendorf M et al, 2022)

Figure 3 illustrates what happens at each time step in the LSTM model. The input data x_t first passes through the forget gate f_t, which also uses the previous hidden state h_{t-1} to decide which parts of the previous cell state C_{t-1} should be removed. The input x_t then passes through the input gate i_t, which determines whether to include new information from the candidate state $\tilde{C}_t$ generated by the tanh function. Finally, the input x_t passes through the output gate o_t, which determines which parts of the updated memory cell should be exposed, producing the output that becomes the new hidden state h_t.

$$h_t = o_t \odot \tanh(C_t)$$

LSTM Hyperparameter tuning

A grid search will be used to test different hyperparameter combinations and find the best candidate. The LSTM architecture will be fixed with one hidden layer as it has a strong balance between capturing complex patterns and preventing overfitting. The loss function will be the MSE to find the combination of parameters with the best performance. For the optimiser selection, which is used to adjust the weights and biases of the parameters to minimise the loss function, the Adam Optimiser was chosen. The Adam Optimiser, which is the default in TensorFlow Keras, is a gradient-descent based optimiser that has a strong balance between training speed and accuracy with its adaptive learning rates. The learning rate (step-size) parameter, which controls for how quickly or slowly the model trains by determining the size of weight updates at each optimization step, will be fixed at 0.001 (lr = 0.001) because it is the default and recommended speed for the Adam optimizer. The dropout rate, which randomly drops neurons to prevent dependency on certain neurons, hence overfitting, will be fixed at 0.2 between layers.

The following parameters will be tuned. The number of neurons, which determines the capacity of the model, will be u = 32, 64, 128), and the batch size, which determines the number of samples processed before updating the weights, will be BS = 16, 32, 64, 128. These sizes range from the binary system of base 2. The number of epochs, which is the number of times the entire dataset is looked at, will be determined using the Keras Earlystopping callback, with a fixed patience of 10, as it is a useful tool that helps prevent overfitting.

3.2.2 Data pre-processing steps

Pre-processing the data is important to generate features that enhance the prediction of the target variable. The data pre-processing steps used in this paper will be forward filling for the financial indicators, leaving gold prices untouched, and dropping NaN values to handle missing

observations and different holidays across the different datasets. All datasets are now aligned to the same daily, financial frequency.

Feature engineering

Given that the LSTM model already uses memory through its RNN structure, meaning it is already capable of learning the temporal dependencies from the sequential data through its sliding window, feature engineering will be excluded from its dataset as it will train using a sliding window of fixed length 63 to represent a financial quarter. In contrast, Random Forests do not automatically account for the sequential nature of the time series data. To address this, the multivariate RF time series will be transformed into a supervised learning though lagged features of 0-62 days. This transformation flattens the sequential structure into a flattened tabular form, allowing the RF to access the same historical information as the LSTM.

Additionally, all the financial indicators (DXY, SP500, WTI and VIX) will be transformed into percentage changes rather than raw price levels to handle non-stationarity and reduce noise in the gold price predictions, improving the model's ability to learn useful relationships under varying market conditions.

The LSTM model will use scaling to account for the incompatible scales between the datasets. This is especially important to consider when models like LSTMs are trained via gradient descent, meaning without scaling larger variables will cause gradients to explode, while smaller variables will cause the gradients to vanish. The standardization (Z-score normalization) method will be applied, using the StandardScaler function from scikit-learn library, to stabilize the training data set and prevent features with large values from disproportionately influencing the model's prediction. All the LSTM features are rescaled to a range between 0 and 1.

Data Splitting

For the validation of the model, temporal splits will be used, training on unscaled data from 2014-2020 and testing on 2021-2024. Selecting the predictor features as variable x and the gold price as the target value variable y, we temporally split x and y values into Train and Test Sets, with 70% x_Train, 30% x_Test and 70% y_Train, 30% y_Test.

3.2.3 Evaluation Metrics

In order to evaluate the performance and accuracy of the regression prediction models, the following metrics are used: Mean Standard Error (MSE): This measures the squared difference between the predicted and the actual values.

Mean Squared Error (MSE): This measures the squared difference between the predicted and the actual values.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Root Mean Squared Error (RMSE): This is the square root of the MSE and measures the magnitude of the difference between the predicted and the actual values, expressed in the same units as the data.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Mean Absolute Error (MAE). This measures the absolute difference between the predicted and the actual values.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

3.3 Implementation

This project was implemented using the Python language. The data loading, cleaning and preprocessing were carried out using pandas and Numpy libraries. The Random Forest regressor model and hyperparameter tuning procedures used the scikit-learn library, while the LSTM model was built and trained using TensorFlow with the KerasAPI. The model performance and forecasts were visualized using matplotlib, and the correlation matrix or exploratory analysis used the seaborn library.

For the system architecture, the main workflow is coordinated in main.py. The source code includes data_loader.py, which handles data loading and preprocessing steps with features and train and test splitting, models.py which mainly handles the model training and hyperparameter tuning are handled in models.py, and evaluation.py which deals with the the evaluation metrics and visualisations of the models. All the raw financial datasets were directly downloaded as CSV files from external sources (FRED.com and Investing.com) and stored locally in the data/raw folder. Given that each variable was downloaded separately, using standardized date formats and punctuation, and only keeping closing prices when merging the datasets was handled in the data preprocessing steps, as well as forward filling financial indicator variables only to account for missing observations and dropping remaining NaN values. The dataset was split temporally into a training and test set before scaling to avoid look-ahead bias.

Example code snippet:

```
def load_and_clean(data_folder):

    # All dates will be transformed into the YYYY-MM-DD format (ISO 8601)
    # Daily Gold Prices (XAUUSD) (Investing.com)
    XAUUSD = pd.read_csv(data_folder/"XAUUSD.csv", parse_dates=["Date"],
                         date_format="%m/%d/%Y")
    # Keeping only the Date and Price (Closing Price) columns
    XAUUSD = XAUUSD[["Date", "Price"]].rename(columns={
                                    "Price": "XAUUSD"})
    # Cleaning commas from numbers and converting values to numeric
    XAUUSD["XAUUSD"] = (XAUUSD["XAUUSD"].astype(str).str
        .replace(",", "", regex=False))
    XAUUSD["XAUUSD"] = pd.to_numeric(XAUUSD["XAUUSD"], errors="coerce")
    # Similarly for DXY, SP500, VIX and WTI
    ...
    # Merging all datasets and sorting by date
    Merged_Raw_Data_Sets = (XAUUSD.merge(DXY, on = "Date", how = "outer")
                          .merge(SP500, on = "Date", how = "outer")
                          .merge(VIX, on = "Date", how = "outer")
                          .merge(WTI, on = "Date", how = "outer")
                         .sort_values("Date"))
    # Forward-fill all financial indicators
    cols_to_fill = [col for col in Merged_Raw_Data_Sets.columns
                    if col not in ["Date", "XAUUSD"]]
    Merged_Raw_Data_Sets[cols_to_fill] = (
        Merged_Raw_Data_Sets[cols_to_fill].ffill())
    # Dropping rows with NaN values and creating a fresh index
    Cleaned_Data_Set = Merged_Raw_Data_Sets.dropna().reset_index(drop=True)

    return Cleaned_Data_Set
```

Listing 1: Example of loading the gold price csv dataset and cleaning

4 Results

4.1 Experimental Setup

All experiments were implemented using Python 3.12 within the Nuvolos cloud-based operating system. All computations used a CPU-based system with 1 virtual CPU and 4 GB of memory, which was sufficient to handle the datasets and computational requirements of the models. A fixed random seed of 42 was used throughout the experiment to ensure reproducibility.

The choice of hyperparameters play a crucial role in maximizing the performance and learning capacity of both machine learning and deep learning models, as well as the training time of the models, hence it is important to find the best combination of parameters. After the tuning procedures mentioned in Section 3.2 we have the following best combinations per model. The final Random Forest regressor combination consists of 764 trees, a maximum depth of 23, a minimum of 18 samples per leaf and a minimum of 45 samples required to split an internal node. The top 5 RF configurations are reported in Appendix A as Table A1. The final LSTM combinations consist of 128 LSTM units, a batch size of 16 and training stopped after 10 epochs based on the early stopping criteria. The top five LSTM configurations are reported in Appendix A as Table A2.

4.2 Performance Evaluation

Model	MSE	RMSE	MAE
Naive Forecast (Benchmark)	12115.33	110.07	91.04
Random Forest Regression	10809.31	103.97	90.1
Long Short-Term Memory	8966.35	94.69	78.28

Table 1: Performance evaluation metrics (MSE, RMSE, MAE) per model.

4.3 Visualizations

The following figures plot the predicted vs actual gold price data to see how each model performs.

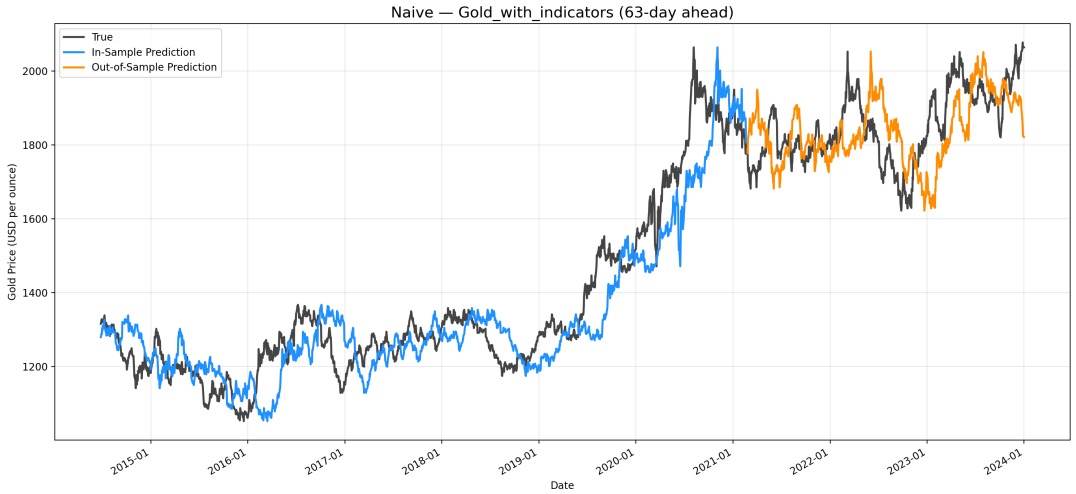

Figure 3: Naive Forecast (Benchmark) model performance in the medium-term (63 day ahead)

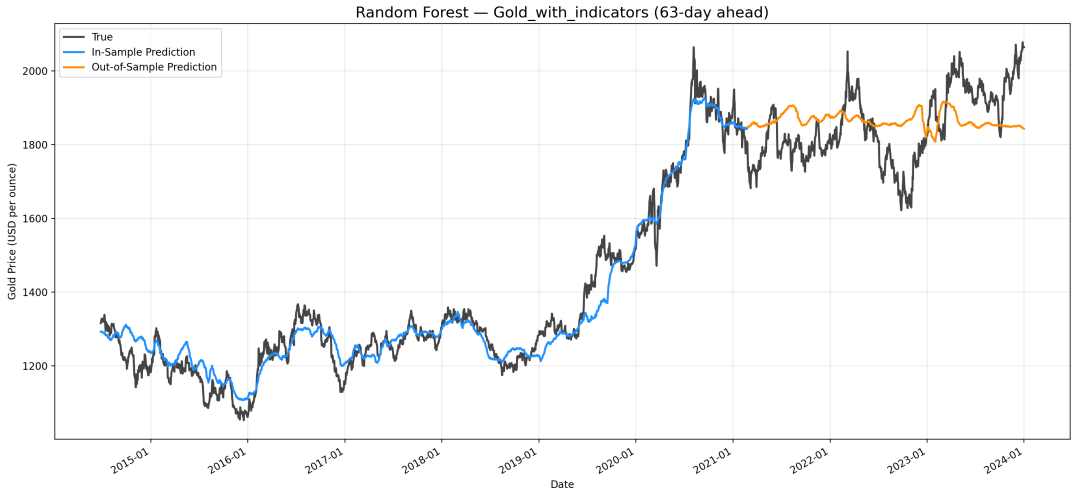

Figure 4: Random Forest Regressor model performance in the medium-term (63 day ahead)

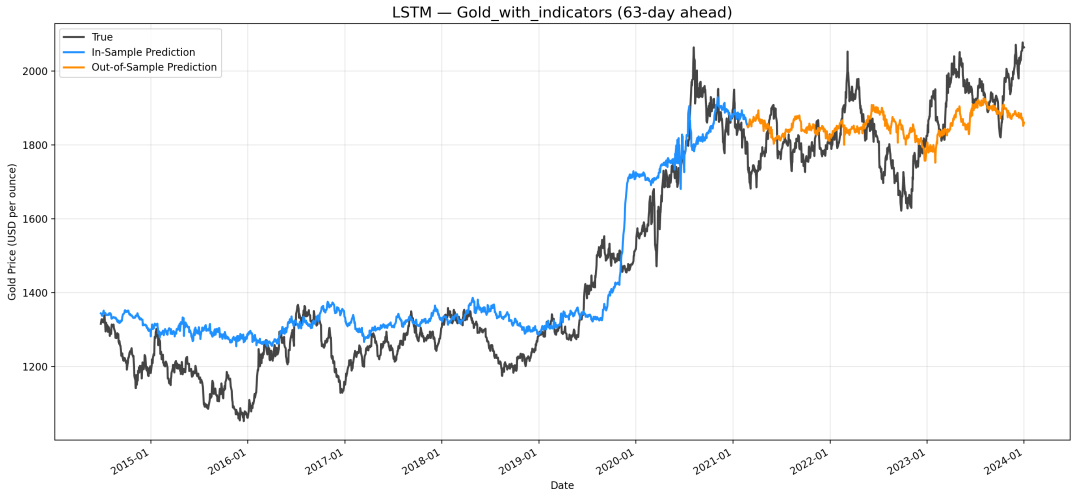

Figure 5: Long Short-Term Memory model performance in the medium-term (63 day ahead)

5 Discussion

The results in Table 1 show clear differences between the models for forecasting gold prices over a 63 day medium-term horizon. The LSTM model achieved the best out-of-sample performance in predicting gold prices in USD per ounce across all of the evaluation metrics, with a MSE of 8966.35, RMSE of 94.69 and MAE of 78.28. The RMSE means that on average the model's quarterly, 63 day horizon, forecasts deviate from the true prices by about 94.69 USD per ounce. Given the longer forecasting horizon, the level of error is consistent with financial medium-term forecasting uncertainty and reflects the increasing difficulty in maintaining accuracy over large horizons, meaning the performance remains informative in analysing gold price dynamics. The LSTM performing the best was expected as it reflects its ability to effectively use its 63 day lookback window to learn sequential patterns and retain useful temporal information from historical data of gold and financial indicators to forecast medium-term trends.

The Random Forest regressor performed better than the naive forecast benchmark, showing that machine learning models add additional predictive value from the nonlinear relationships between gold and the financial indicators in the medium-term horizon. However, despite the

RF achieving a strong in-sample performance, its out-of-sample forecasts were considerably smoother, reverting to more average values rather than capturing any spikes as seen in Figure 4, showing a key challenge with the model. Despite extensive hyperparameter tuning using a time series cross validation with the regularization of parameters, such as constraining the max tree depth and min leaf size to prevent overfitting, the RF still has a large gap between in-sample and out-of-sample performance. While tuning helped reduce excessive in-sample variability, it also resulted with worsened out-of-sample predictions. This suggests that the weaker performance of the RF model could reflect a structural limitation of the tree-based ensemble model itself in capturing temporal dynamics over a 63 day medium-term forecasting horizon.

The naive forecast benchmark remained competitive, highlighting the difficulty of forecasting financial prices even with a medium-term horizon. Its performance shows the importance of using strong baselines when evaluating more complex, machine learning and deep learning models.

Overall, the results align with the expectations, as the models that account for temporal structure with the RF model using lagged features and especially the LSTM model through its built in sequences, are better suited for a financial time series setting and for forecasting medium-term prices, where trends and temporal patterns play an important role. Regarding the limitations of this approach, the analysis focused on a single forecasting horizon and had a fixed lookback window, meaning the results may differ when forecasting with a short-term and long-term horizon. Additionally, the random forest did not receive any additional features, such as moving averages and rolling volatiles, which could have helped strengthen it in capturing temporal dependencies given the time-series setting.

6 Conclusion and Future Work

6.1 Summary

Machine and Deep learning models such as Random Forest and LSTM have a proven value in forecasting gold price and other financial time series data in the medium-term, as they were both able to beat the Naive forecast benchmark. The stronger performance of the LSTM suggests that this model may be useful in supporting medium-term gold investment decisions, as it can handle non-linear interactions and complex temporal dependencies between the variables in the input data. Random Forest does not deal with temporal dependencies as efficiently when only given lagged features, and due to the approach used in this paper the model also experiences overfitting issues.

6.2 Future Directions

Financial markets remain volatile and complex, and they are constantly evolving, meaning that ensuring the generalisability of the models for long-term forecasting remains a challenge. Incorporating financial or macroeconomic factors into the models enhances their applicability to the real world. Future work can explore considering more explanatory factors including other macroeconomic indicators, technical indicators and news sentiment data. Other models could also be explored such as a Bi-directional LSTM or a hybrid model which combines LSTM with ARIMA or with GARCH to handle market volatility forecasting, or LSTM with CNN (Convolutional Neural Network) to improve the extraction of relevant features and reduce dimensionality and noise.

References

Bruggen A et al (2025). Gold Demand: The Role of the Official Sector and Geopolitics. European Central Bank. https://www.ecb.europa.eu/press/other-publications/ire/focus/html/ecb.irebox202506_01~f93400a4aa.en.html

Chai J, Zhao C, Hu Y, and Zhang ZG (2021). Structural analysis and forecast of gold price returns. Journal of Management Science and Engineering, 6(2), 135–145. https://doi.org/10.1016/j.jmse.2021.02.011

Chew L, Yi S, and Ong Y (2023). Gold prices forecasting using bidirectional LSTM model based on SPX500 index, USD index, crude oil prices and CPI. Proceedings of the 2023 International Conference on Information and Communication Technology (ICoICT), 539–544. https://doi.org/10.1109/ICoICT58202.2023.10262481

Cohen G and Aiche A (2023). Forecasting gold price using machine learning methodologies. Chaos, Solitons & Fractals, 175, 114079. https://doi.org/10.1016/j.chaos.2023.114079

Federal Reserve Bank of St. Louis (n.d.). CBOE Volatility Index (VIX) [VIXCLS]. Federal Reserve Economic Data (FRED). https://fred.stlouisfed.org/series/VIXCLS

Federal Reserve Bank of St. Louis (n.d.). Crude Oil Prices: West Texas Intermediate (WTI) [DCOILWTICO]. Federal Reserve Economic Data (FRED). https://fred.stlouisfed.org/series/DCOILWTICO

Federal Reserve Bank of St. Louis (n.d.). S&P 500 [SP500]. Federal Reserve Economic Data (FRED). https://fred.stlouisfed.org/series/SP500

Gold Avenue (n.d.). Understanding the price of gold. https://www.goldavenue.com/en/precious-metals-guide/a-beginner-s-guide-to-gold

Gong W (2024). Research on gold price forecasting based on LSTM and linear regression. SHS Web of Conferences, 181. https://doi.org/10.1051/shsconf/202418102005

Guo Y (2024). Research on the application of gold price prediction based on LSTM model. Information Systems and Economics, 5(4). https://doi.org/10.23977/infse.2024.050414

Hochreiter S and Schmidhuber J (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Investing.com (n.d.). Gold spot price (XAU/USD) historical data. https://www.investing.com/currencies/xau-usd-historical-data

Investing.com (n.d.). U.S. Dollar Index historical data. https://www.investing.com/indices/usdollar-historical-data

Jarvis G (2025). Why central banks are turning to gold. World Finance. https://www.worldfinance.com/special-reports/why-central-banks-are-turning-to-gold

JP Morgan (2025). Gold prices. https://www.jpmorgan.com/insights/global-research/commodities/gold-prices

Liu Y (2024). Python Machine Learning by Example (4th ed.), Chapter 12.

Mittendorf M et al (2022). The prediction of sea state parameters by deep learning techniques using ship motion data. 7th World Maritime Technology Conference.

Mukherjee S (2025). A practical guide to random forests in machine learning. https://www.digitalocean.com/community/tutorials/random-forest-in-machine-learning

Nagata AB et al (2025). Predicting gold price movement using long short-term memory model. Journal of Applied Intelligent System, 9(1), 19–28. https://doi.org/10.62411/jais.v9i1.10305

Olah C (2015). Understanding LSTM networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Pan X (2024). The comparison between random forest and LSTM models based on the gold price prediction. Advances in Economics Management and Political Sciences, 94(1), 123–129. https://doi.org/10.54254/2754-1169/94/2024OX0150

Sahour H et al (2021). Random forest and extreme gradient boosting algorithms for streamflow modelling. Environmental Earth Sciences, 80(22). https://doi.org/10.1007/s12665-021-10054-5

Saputra I (2025). Gold investment choice for Gen Z in 2025. Proceedings of the International Conference on Business and Economics, 3(1), 42–45. https://doi.org/10.56444/icbe-untagsmg.v3i1.2699

Sharma K (2024). Mastering random forest hyperparameter tuning for enhanced machine learning models. Medium. https://medium.com/@kalpit.sharma/mastering-random-forest-hyperparameter-tuning-for-enhanced-machine-learning-models-2d1a8c6c426f

Taneva-Angelova G et al (2025). A framework for gold price prediction combining classical and intelligent methods with financial, economic, and sentiment data fusion. International Journal of Financial Studies, 13(2), 102. https://doi.org/10.3390/ijfs13020102

Wang W (2024). Comparative analysis of ARIMA, random forest, and LSTM models for Mercedes-Benz stock price prediction. Advances in Economics Management and Political Sciences, 134(1), 1–8. https://doi.org/10.54254/2754-1169/2024.18718

Zhang R (2024). Gold price relative return prediction with machine learning models. Proceedings of the 2nd International Conference on Data Analysis and Machine Learning (DAML), 579–584. https://doi.org/10.5220/0013528700004619

Use of AI Tools:

ChatGPT (OpenAI) was used as an AI tool to assist in clarifying concepts and supporting the development of the code.

OpenAI (2025). ChatGPT (version GPT-5.2) [Large language model]. https://chat.openai.com/

A Additional Figures

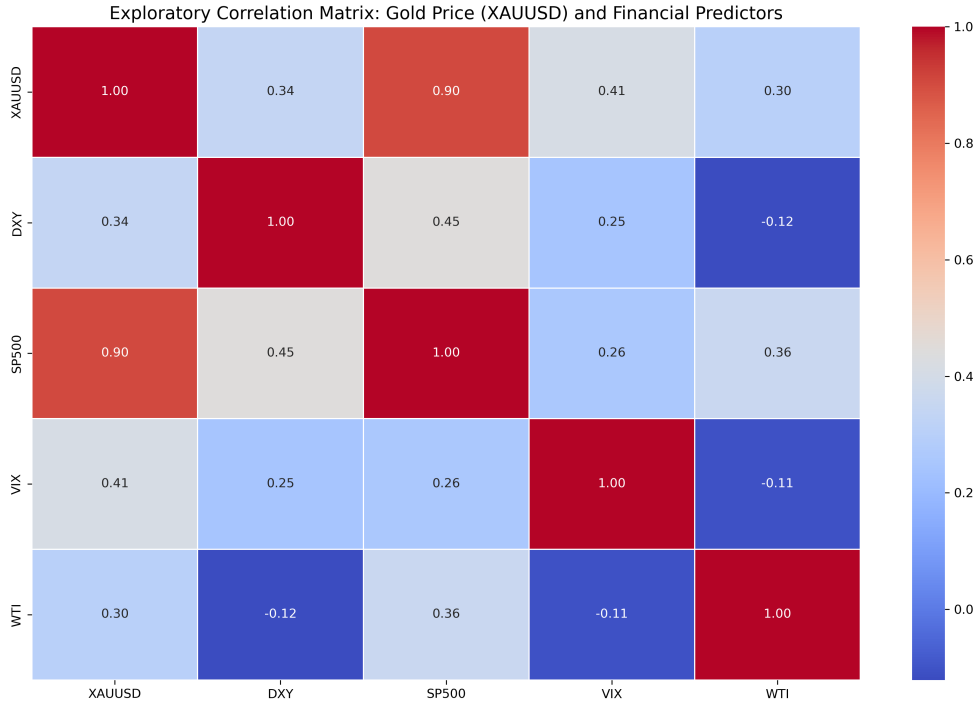

Figure A1: Correlation Matrix of Gold Price (XAUUSD), DXY, S&P500, VIX and WTI

Based on the results in Figure A1, we see that gold prices and S&P 500 have the strongest correlation, with a positive coefficient of 0.90, meaning that increases in the US stock index increases are generally associated with increases in gold prices. Moderate correlations are seen between gold and DXY, VIX and WTI, with correlation coefficients of 0.34, 0.41, 0.30 and 0.38, respectively. This means that gold prices are positively associated with all these variables.

Rank	n_estimators	max_depth	min_samples_leaf	min_samples_split	mean_val_MSE
1	764	23	18	45	29243.95
2	661	25	24	27	29963.6
3	508	28	33	22	30086.43
4	305	11	15	73	30156.93
5	585	11	39	57	30408.36

Table A1: Random Forest Regressor top 5 hyperparameter combinations.

Rank	lstm_units	batch_size	best_epoch	best_val_mse
1	128	16	10	7548.96
2	32	16	10	16692.81
3	128	32	18	17755.13
4	128	128	20	19343.03
5	32	32	11	19561.99

Table A2: Long Short-Term Memory top 5 hyperparameter combinations.

B Code Repository

GitHub Repository: `https://github.com/amaliamn/gold-price-forecasting`

This repository contains all the source code, raw datasets, data processing scripts, and experiment outputs needed to reproduce the results presented in this report.

Repository Structure

The project is organized as follows:

```
my-project/
├── main.py              # Main entry point
├── data/                # Data directory
│   ├── raw/             # Raw CSV files downloaded from data sources
├── src/                 # Source code
│   ├── data_loader.py   # Data loading and preprocessing
│   ├── models.py        # Model training and hyperparameter tuning
│   └── evaluation.py    # Evaluation metrics
├── results/             # Generated outputs
│   ├── metrics/         # Evaluation metrics table
│   ├── figures/         # Prediction plots
│   └── appendix/        # Correlation matrix and hyperparameter tuning top 5 results
└── environment.yml      # Software dependencies
```

Installation Instructions

Create and activate the Conda environment

```
conda env create -f environment.yml
conda activate gold_forecasting
```

Reproducing the Results

To reproduce all experiments, figures, and evaluation tables, set the project directory and run:

```
python main.py
```