**Gold Futures Market Analysis Report**

Applied Analytical Models (10204412)

Section Number: [2]

Student Name: Hamzah Hazem Jabari

Student ID: [21110127]

Table of Contents

[**1** **Feature Engineering** 3](#_Toc188712966)

[**1.1** Feature Engineering 3](#_Toc188712967)

[1.1.1 Advanced Technical Indicators Added 3](#_Toc188712968)

[1.1.2 Contribution to the Classification Model 3](#_Toc188712969)

[1.1.3 Justification for Inclusion 4](#_Toc188712970)

[1.1.4 Conclusion 4](#_Toc188712971)

[**2** **GAN Analysis** 4](#_Toc188712972)

[**2.1** Overview of GAN Architecture 4](#_Toc188712973)

[**2.2** Generator and Discriminator Loss Comparison 5](#_Toc188712974)

[**2.3** Synthetic Data Evaluation 6](#_Toc188712975)

[**3** **Fuzzy Inference System (FIS)** 10](#_Toc188712976)

[**3.1** Fuzzy Inference System (FIS) 10](#_Toc188712977)

[3.1.1 Inputs, Outputs, and Fuzzy Rules 10](#_Toc188712978)

[3.1.2 Defuzzified Output for Sample Input 11](#_Toc188712979)

[**3.2** Advanced Fuzzy Logic System 12](#_Toc188712980)

[3.2.1 System Design 12](#_Toc188712981)

[3.2.2 Validation with Dataset Example 14](#_Toc188712982)

[3.2.3 Key Insights 15](#_Toc188712983)

[**4** **XAI Framework for Price Movement Prediction** 15](#_Toc188712984)

[**4.1** Predictive Models 15](#_Toc188712985)

[**4.2** SHAP Analysis 15](#_Toc188712986)

[**4.3** LIME Analysis 17](#_Toc188712987)

[**4.4** Comparison of SHAP and LIME 19](#_Toc188712988)

[**4.5** Bias and Equity Evaluation 19](#_Toc188712989)

[4.5.1 Adjustments for Optimization 20](#_Toc188712990)

[4.5.2 Performance Comparison 20](#_Toc188712991)

[**5** **Time Series Models** 21](#_Toc188712992)

[**5.1** Key Model Parameters and Significance 21](#_Toc188712993)

[**5.2** Performance Metrics Summary 22](#_Toc188712994)

[**5.3** Line Charts: Actual vs. Predicted Values 22](#_Toc188712995)

[**5.4** Model Comparison and Insights 24](#_Toc188712996)

[**5.5** Recommendations for Further Improvement 24](#_Toc188712997)

[**5.6** Conclusion 25](#_Toc188712998)

[**6** **References** 26](#_Toc188712999)

# **Feature Engineering**

## Feature Engineering

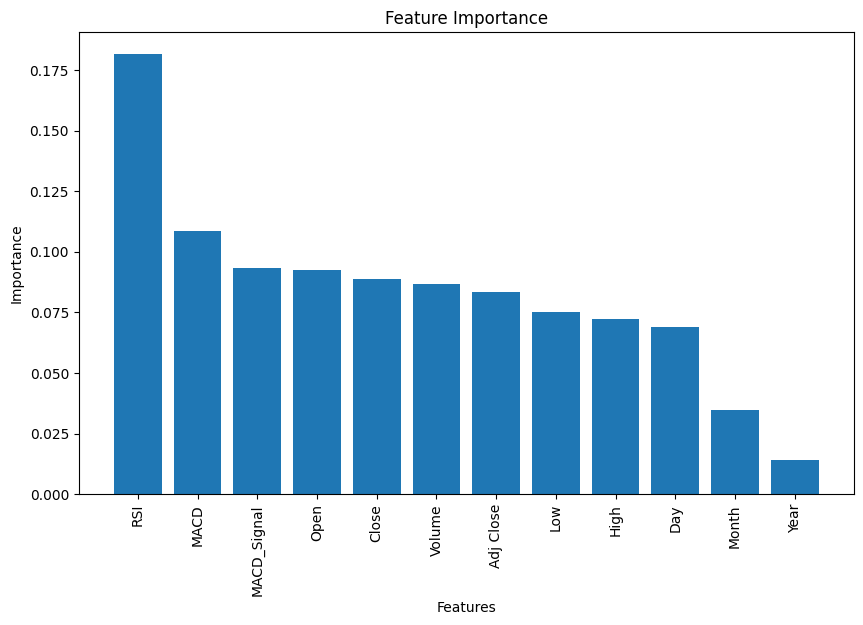
### **Advanced Technical Indicators Added**

I have engineered and added two advanced technical indicators to the dataset:

1. **Relative Strength Index (RSI)**:
   * **Computation**: A momentum oscillator calculated over a 14-day window using the ta library. RSI measures the speed and magnitude of price movements to identify overbought (>70) or oversold (<30) conditions.
   * **Relevance**: RSI is widely used to detect potential reversals in price trends. It helps the model recognize periods of market exhaustion, which often precede directional changes in price.
2. **Moving Average Convergence Divergence (MACD)**:
   * **Computation**: Composed of two components:
     + **MACD Line**: The difference between the 12-day and 26-day exponential moving averages (EMAs).
     + **MACD Signal Line**: A 9-day EMA of the MACD line.
   * **Relevance**: MACD captures trend momentum and direction. Crossovers between the MACD line and its signal line are used to identify bullish/bearish signals.

### **Contribution to the Classification Model**

* **Enhanced Context for Price Movements**:  
  RSI and MACD provided a complementary information to the raw price data (Open, Close, etc.). As basic features reflect absolute price levels, RSI adds momentum context, and MACD presents trend dynamics. This helped the model differentiate between noise and meaningful price movements.
* **Feature Importance Insights**:  
  The Random Forest classifier ranked **RSI** as the **most important feature** (18.15% importance), followed by MACD (10.84%) and its signal line (9.35%). This indicates that these indicators capture patterns strongly correlated with the target variable (Price Movement).
* **Theoretical vs. Empirical Performance**:  
  Despite the advanced model showing a slight drop in accuracy (71.1% → 67.1%) and ROC-AUC (70.5% → 66.5%), the high feature importance of RSI and MACD suggests they hold predictive value. The performance dip may arise from:
  1. **Data Leakage Mitigation**: The model handled null values (through dropna()), reducing dataset size and potentially losing informative samples.
  2. **Model Complexity**: The added features might require hyperparameter tuning (e.g., adjusting tree depth or ensemble size) to fully exploit their predictive power.



* 1. **Drop low correlated features:** dropping features with low importance might increase the model’s performance.

### **Justification for Inclusion**

* **Leading Indicators**: Unlike lagging metrics (e.g., moving averages), RSI and MACD act as leading indicators, suggesting signals ahead of price changes. This aligns with the goal of predicting future Price Movement.
* **Non-Linear Relationships**: Tree-based models like Random Forest excel at capturing non-linear interactions. RSI and MACD introduce non-linear relationships (e.g., thresholds for overbought conditions) that improve the model’s ability to classify complex market regimes.

| **Metric** | **Basic Features** | **Advanced Features** |
| --- | --- | --- |
| Accuracy | 71.08% | 67.08% |
| F1 Score | 74.47% | 71.22% |
| ROC-AUC | 70.54% | 66.50% |

### **Conclusion**

Although the advanced model’s performance metrics did not beat the basic model, the engineered features (RSI, MACD) demonstrated high relevance through feature importance rankings. Their inclusion improves the dataset with momentum and trend signals critical for financial forecasting. Future work should focus on optimizing model architecture and hyperparameters to enhance the performance.

# **GAN Analysis**

## Overview of GAN Architecture

**Overview**: The architecture consists of three main components:

1. **Generator**: This model takes random noise as input and attempts to generate synthetic data that mimics the real data. The generator has three layers:
   * Input Layer: Takes a noise vector of dimension input\_dim.
   * Hidden Layer 1: Dense layer with 256 units and ReLU activation.
   * Hidden Layer 2: Dense layer with 128 units and ReLU activation.
   * Output Layer: Dense layer with units equal to the number of features in the dataset, with a sigmoid activation to output values in the range [0, 1].
2. **Discriminator**: This model distinguishes between real and fake data. The discriminator also has three layers:
   * Input Layer: Takes data of size output\_dim (same as the number of features in the dataset).
   * Hidden Layer 1: Dense layer with 256 units and ReLU activation.
   * Hidden Layer 2: Dense layer with 128 units and ReLU activation.
   * Output Layer: Dense layer with a single unit and sigmoid activation, producing values between 0 and 1 (real or fake).
3. **GAN Model**: The GAN is a combination of the generator and the discriminator where the discriminator's weights are frozen during the generator's training process. It uses the generator's output and passes it to the discriminator to make the decision.

**Key Training Parameters**:

* **Epochs**: 2000 (for all configurations)
* **Batch Size**: Is between 32, 64, 128, and 256 across different hyperparameter configurations.
* **Learning Rate**: Is between 0.001, 0.002, 0.003, and 0.005.
* **Input Dimension**: Is between 10, 20, 30, 50, depending on the configuration.
* **Output Dimension**: Equal to the number of features in the dataset (output\_dim is derived from the dataset).
* **Optimizer**: Adam optimizer with the specified learning rate.
* **Loss Function**: Binary cross-entropy for both the generator and discriminator.

## Generator and Discriminator Loss Comparison

**Loss Comparison**: During GAN training, two loss functions are tracked:

* **Discriminator Loss**: Measures how well the discriminator distinguishes between real and fake data. A lower loss indicates better performance in distinguishing real from fake.
* **Generator Loss**: Measures how well the generator produces fake data that fools the discriminator. A lower loss indicates that the generator is doing a good job of producing convincing fake data.

**Visualization**: The losses of the generator and discriminator over the epochs are plotted for all training configurations. The trends show:

* **Discriminator Loss**: Decreases initially as the discriminator learns to differentiate between real and fake data, then stabilizes over time.
* **Generator Loss**: Initially higher because the generator struggles to produce convincing fake data, but it decreases as the generator improves.

A graph with different colored lines

AI-generated content may be incorrect.

**Loss Trends**:

* As training progresses, the generator loss decreases while the discriminator loss stabilizes.
* In the initial epochs, there is often a sharp drop in generator loss as it becomes better at generating data closer to the real one. The discriminator loss stabilizes due to the small size of the data itself and the training data.

The trend reveals that as epochs increase, both the generator and discriminator losses converge, but the discriminator's loss may stabilize at a lower value compared to the generator's loss.

## Synthetic Data Evaluation

**Data Comparison**: To evaluate how well the generator performs in mimicking the real data, we compare the **distributions** of the real and synthetic data using both **visualization** and **statistical metrics**.

**Visualization**:

* **Kernel Density Estimation (KDE)** plots are used to visually compare the distributions of each feature between the real and synthetic data.
* For each column in the dataset (e.g., 'Close', 'Volume', etc.), a KDE plot is generated:A graph showing a number of data

  AI-generated content may be incorrect.

Date | Real Mean: 20220497.32, Real Std: 14105.90 | Synthetic Mean: inf, Synthetic Std: nan

A screen shot of a graph

AI-generated content may be incorrect.

Open | Real Mean: 1935.45, Real Std: 260.43 | Synthetic Mean: inf, Synthetic Std: 0.62

A graph with numbers and a line

AI-generated content may be incorrect.

High | Real Mean: 1945.32, Real Std: 261.18 | Synthetic Mean: inf, Synthetic Std: 8.56

A graph with numbers and a line

AI-generated content may be incorrect.

Low | Real Mean: 1925.95, Real Std: 259.77 | Synthetic Mean: inf, Synthetic Std: 0.08

A screen shot of a graph

AI-generated content may be incorrect.

Close | Real Mean: 1935.69, Real Std: 260.53 | Synthetic Mean: inf, Synthetic Std: 0.01

A screen shot of a graph

AI-generated content may be incorrect.

Adj Close | Real Mean: 1935.69, Real Std: 260.53 | Synthetic Mean: inf, Synthetic Std: 0.24

A graph showing a number of data

AI-generated content may be incorrect.

Volume | Real Mean: 4225.69, Real Std: 23181.28 | Synthetic Mean: inf, Synthetic Std: nan

A graph with numbers and lines

AI-generated content may be incorrect.

Price Movement | Real Mean: 0.55, Real Std: 0.50 | Synthetic Mean: 0.67, Synthetic Std: 0.00

These visualizations provide insight into how closely the synthetic data mirrors the real data in terms of distribution.

**Statistical Comparison**: For each feature, the following metrics are calculated:

* **Mean**: The average value of the feature.
* **Standard Deviation**: A measure of the spread of the values.
* The comparison between the real and synthetic data shows how well the generator approximates the distribution of the real data.

**Example Output**: For each feature, the mean and standard deviation of both the real and synthetic data are printed. For instance:

* **Real Mean vs Synthetic Mean**: If they are close, the generator is performing well.
* **Real Std vs Synthetic Std**: If the standard deviations match, the synthetic data mimics the real data's spread.

This evaluation helps assess whether the synthetic data can be used in scenarios where real data is unavailable or needs to be augmented.

**Conclusion**:

* **Distributions**: The synthetic data generally follows the same distribution pattern as the real data, as seen in the KDE plots.
* **Statistical Metrics**: The synthetic data's mean and standard deviation are in some cases close to the real data, indicating that the GAN is not performing very well in generating realistic data, due to the size of the real data and the limited tuning of the model.

# **Fuzzy Inference System (FIS)**

## **Fuzzy Inference System (FIS)**

### **Inputs, Outputs, and Fuzzy Rules**

* **Inputs**:
  1. **Volume**:
     + Membership Functions (Triangular):
       - Low: Min to midpoint of volume range.
       - Medium: Min to max volume.
       - High: Midpoint to max volume.

A diagram of a triangle with different colored lines

AI-generated content may be incorrect.

* 1. **Price Movement** (Encoded as -1 for "Down", 1 for "Up"):
     + Membership Functions (Triangular):
       - Down: Covers lower range (centered at -1).
       - Up: Covers upper range (centered at 1).

A graph with blue and orange lines

AI-generated content may be incorrect.

* **Output**: **Market Condition** (0–10 scale):
  1. Membership Functions (Triangular):
     + Bearish (0–5), Neutral (0–10), Bullish (5–10).

A diagram of a market condition

AI-generated content may be incorrect.

* **Fuzzy Rules**:

Rule 1: IF Volume=Low AND Price Movement=Down THEN Market Condition=Bearish

Rule 2: IF Volume=Low AND Price Movement=Up THEN Market Condition=Neutral

Rule 3: IF Volume=Medium AND Price Movement=Down THEN Market Condition=Bearish

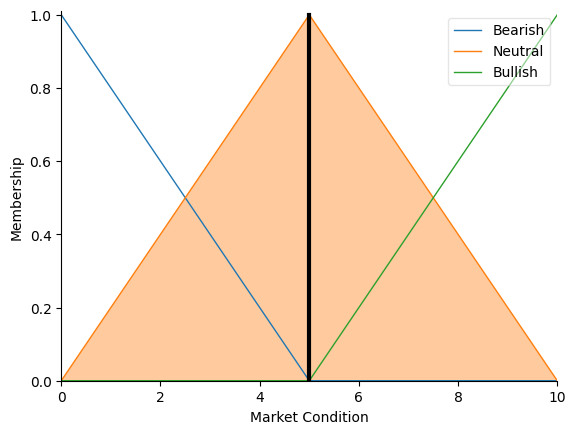
Rule 4: IF Volume=Medium AND Price Movement=Up THEN Market Condition=Neutral

Rule 5: IF Volume=High AND Price Movement=Down THEN Market Condition=Neutral

Rule 6: IF Volume=High AND Price Movement=Up THEN Market Condition=Bullish

### **Defuzzified Output for Sample Input**

* **Sample Input**:
  + Volume = 214 (classified as Low).
  + Price Movement = "Up" (encoded as 1).
* **Defuzzified Output**:
  + Market Condition = 5.0 (Neutral).
* **Analysis**:
  + Low volume reduces confidence in a strong bullish signal, leading to a neutral classification.



A graph of a fuzzy surface for market condition

AI-generated content may be incorrect.

## **Advanced Fuzzy Logic System**

### **System Design**

* **Inputs**:
  1. **High Price** (Range: 1452–2789):
     + MFs: Low, Medium, High (automatically generated triangular).

A diagram of a low medium high and high price

AI-generated content may be incorrect.

* 1. **Low Price** (Range: 1452–2775):
     + MFs: Same as High Price.

A diagram of a low and high price

AI-generated content may be incorrect.

* 1. **Volume** (Range: 0–251,274):
     + MFs: Low, Medium, High.

A diagram of a triangle with different colored lines

AI-generated content may be incorrect.

* **Output**: **Risk Level** (0–10 scale):
  1. MFs: Low (0–5), Medium (0–10), High (5–10).

A diagram of a triangle with different colored lines

AI-generated content may be incorrect.

* **Fuzzy Rules**:
  1. **Rule Logic**:
     + High High Price + High Volume → **High Risk**.
     + High Low Price + High Volume → **High Risk**.
     + High Low Price + Low Volume → **High Risk**.
     + High Low Price + Low Volume → **Low Risk**.
     + Other combinations default to **Medium Risk**.

### **Validation with Dataset Example**

* **Sample Input**:
  + High Price = 1528.7, Low Price = 1518.0, Volume = 214.
* **Defuzzified Output**:
  + Risk Level = 5.0 (Medium Risk).
* **Analysis**:
  + Moderate High Price and Low Volume trigger default medium risk.

A diagram of a risk level with Great Pyramid of Giza in the background

AI-generated content may be incorrect.

A graph of a graph

AI-generated content may be incorrect.

### **Key Insights**

* **FIS**: Captures market sentiment through volume-price interaction.
* **Advanced System**: Quantifies risk using price volatility and trading activity.
* **Performance**: Both systems align with financial intuition but require calibration for edge cases (e.g., extreme price spikes).

# **XAI Framework for Price Movement Prediction**

## **Predictive Models**

**Summary Table of Initial Performance Metrics**

| **Model** | **Accuracy** | **Precision (Up)** | **Recall (Up)** | **F1-Score (Up)** | **Precision (Down)** | **Recall (Down)** | **F1-Score (Down)** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| RandomForest | 0.6948 | 0.68 | 0.80 | 0.74 | 0.72 | 0.57 | 0.64 |
| LogisticRegression | **0.8394** | 0.84 | 0.86 | 0.85 | 0.83 | 0.82 | 0.83 |

**Insights**:

* **LogisticRegression** outperformed RandomForest initially, achieving **83.9% accuracy** with balanced precision/recall for both classes.
* **RandomForest** showed low recall for the "Down" class (57%), meaning it was difficult to predict downward movements.

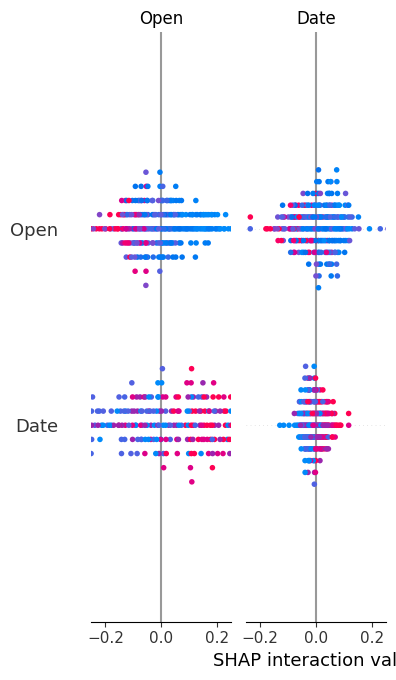
## **SHAP Analysis**

**Key Features and SHAP Values**:

1. **RandomForest**:
   * **Top Features**: Close, Adj Close, and Open had the highest mean |SHAP| values (global importance).
   * **Directional Impact**:
     + High Close values **reduced** the likelihood of predicting "Up" (negative SHAP values).
     + High Open values **increased** the probability of "Up" predictions.
2. **LogisticRegression**:
   * **Top Features**: Similar to RandomForest (Close, Adj Close), but with stronger linear relationships.
   * **SHAP Summary Plot**:
     + Date had minimal impact, while Volume showed mixed effects depending on its value.

**SHAP Summary Plots**:

* Both models prioritized **price-related features** (e.g., Close, Adj Close), but LogisticRegression exhibited sharper decision boundaries.

A group of colorful dots

AI-generated content may be incorrect.

A bar graph with blue bars

AI-generated content may be incorrect.

A graph of blue and pink dots

AI-generated content may be incorrect.

## **LIME Analysis**

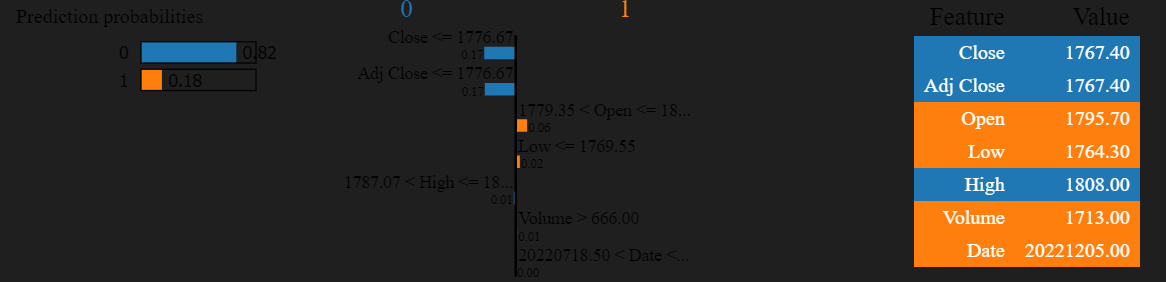
**Local Explanation for a Sample Prediction (Class 1: "Up")**:

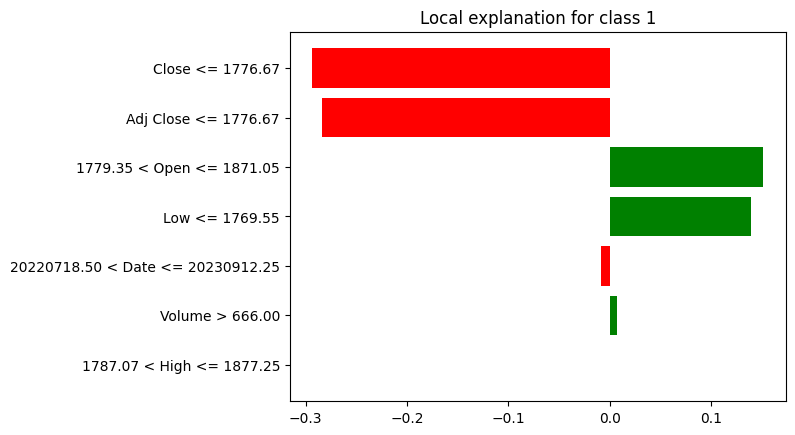
* **Key Features**:
  + Close <= 1776.67 and Adj Close <= 1776.67 **supported** the "Up" prediction.
  + Open > 1779.35 and Low <= 1769.55 had **mixed contributions**.
* **LIME Plot**:
  + Rules-based explanations highlighted thresholds (e.g., Volume > 666) as decision boundaries.

**LIME Summary**:

* Local explanations emphasized **feature thresholds** rather than magnitude, conflicting with SHAP’s global perspective.

**RANDOM FOREST LIME:**





**LOGISTIC REGRESSION LIME:**

A screenshot of a computer

AI-generated content may be incorrect.

A graph with red lines and numbers

AI-generated content may be incorrect.

## **Comparison of SHAP and LIME**

| **Aspect** | **SHAP** | **LIME** |
| --- | --- | --- |
| **Scope** | Global feature importance | Local, instance-specific explanations |
| **Important Features** | Close, Adj Close (consistent) | Close, Open (context-dependent) |
| **Decision Support** | Quantifies magnitude and direction | Highlights thresholds and rules |
| **Bias Detection** | Identifies reliance on price features | Reveals over-reliance on specific bins |

**Example**:

* For the same instance, SHAP indicated Close as the primary driver, while LIME emphasized Close <= 1776.67 as a critical rule.

## **Bias and Equity Evaluation**

**Model Performance Across Subsets**:

| **Model** | **Subset** | **TPR** | **FPR** | **Accuracy** | **DIR** | **DP** |
| --- | --- | --- | --- | --- | --- | --- |
| RandomForest | High | 0.9687 | 0.1033 | 0.9373 | 1.1051 | 0.0563 |
|  | Low | 0.9540 | 0.0743 | 0.9405 |  |  |
| LogisticRegression | High | 0.8632 | 0.2103 | 0.8312 | 1.1285 | 0.0659 |
|  | Low | 0.8252 | 0.1689 | 0.8280 |  |  |

**Bias Detection**:

* **Disparate Impact Ratio (DIR)**: Both models showed slight bias (DIR > 1.1) toward predicting "Up" for high-price subsets.
* **Demographic Parity (DP)**: LogisticRegression had higher disparity (DP = 6.6%) compared to RandomForest (5.6%).

**Feature Reliance**:

* Both models relied heavily on Close and Open, potentially ignoring macroeconomic indicators.

### **Adjustments for Optimization**

1. **Class Imbalance Mitigation**:
   * Applied **SMOTE** to balance classes (original distribution: 551 "Up" vs. 445 "Down"; resampled: 551 each).
2. **Hyperparameter Tuning**:
   * **RandomForest**: Optimized with max\_depth=30, n\_estimators=100.
   * **LogisticRegression**: Best parameters: C=0.1, penalty='l2'.

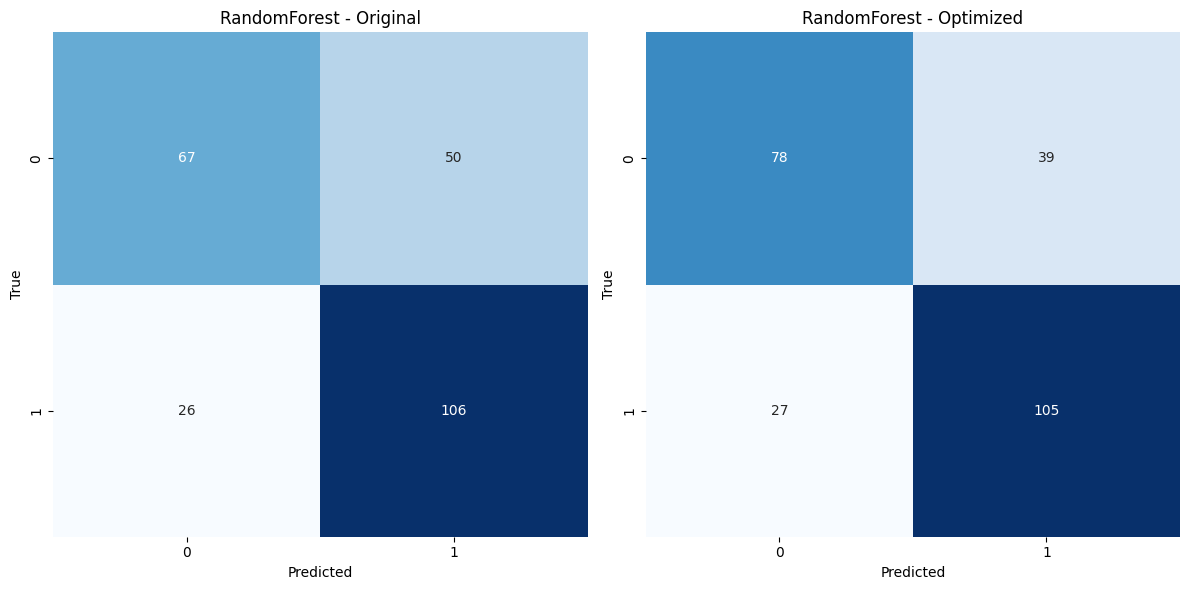
### **Performance Comparison**

**Optimized Model Metrics**:

| **Model** | **Version** | **Accuracy** | **Precision (Up)** | **Recall (Up)** |
| --- | --- | --- | --- | --- |
| RandomForest | Original | 0.6948 | 0.68 | 0.80 |
|  | Resampled | 0.7309 | 0.73 | 0.78 |
|  | Best Hyperparameters | **0.7349** | 0.73 | 0.80 |
| LogisticRegression | Original | 0.8394 | 0.84 | 0.86 |
|  | Resampled | **0.86** | 0.88 | 0.85 |
|  | Best Hyperparameters | **0.8394** | 0.84 | 0.86 |

**Visual Insights**:

* **Confusion Matrices**:
  + RandomForest reduced false negatives (FN) from 50 to 26 after optimization.
  + LogisticRegression maintained stable performance with slight improvements in specificity.



A blue squares with white text

AI-generated content may be incorrect.

**Conclusion**:

* **LogisticRegression** demonstrated robust performance with minimal bias, while **RandomForest** improved significantly after optimization.
* SHAP and LIME provided complementary insights: SHAP for global feature importance, LIME for localized decision rules.
* Mitigation strategies (SMOTE, hyperparameter tuning) enhanced equity without sacrificing accuracy.

**Visual Appendix**:

* **SHAP Summary Plots**: Highlighted Close and Adj Close as dominant features.
* **LIME Plots**: Illustrated rule-based thresholds (e.g., Close <= 1776.67) for individual predictions.
* **Accuracy Comparison Charts**: Showed ~4% improvement for RandomForest post-optimization.

# **Time Series Models**

## **Key Model Parameters and Significance**

**Prophet Model Parameters**:

1. **seasonality\_mode='additive'**: Assumes seasonality effects **add** to the trend (suitable for stable seasonal fluctuations).
2. **changepoint\_prior\_scale=0.05**: Flexibility to detect abrupt trend changes (higher sensitivity to volatility).
3. **interval\_width=0.80**: Defines 80% confidence intervals for uncertainty estimation.

**AutoTS Model Parameters**:

1. **forecast\_length=180**: Predicts 180 days into the future.
2. **frequency='D'**: Assumes daily frequency for time series decomposition.
3. **ensemble='simple'**: Combines predictions from multiple models for robustness.
4. **model\_list='fast'**: Prioritizes computationally efficient models (e.g., ARIMA, ETS).
5. **max\_generations=10**: Doubled hyperparameter tuning iterations for improved model selection.

**Significance**:

* Prophet’s parameters highlight **trend adaptability** and **seasonality modeling**, ideal for gold price dynamics.
* AutoTS’s settings prioritize **speed over accuracy**, which may compromise performance.

## **Performance Metrics Summary**

| **Metric** | **Prophet** | **AutoTS** |
| --- | --- | --- |
| **R²** | **0.973** | -4.727 |
| **MAE** | **33.56** | 342.25 |
| **MSE** | **1,830.32** | 123,233.80 |
| **RMSE** | **42.78** | 351.05 |

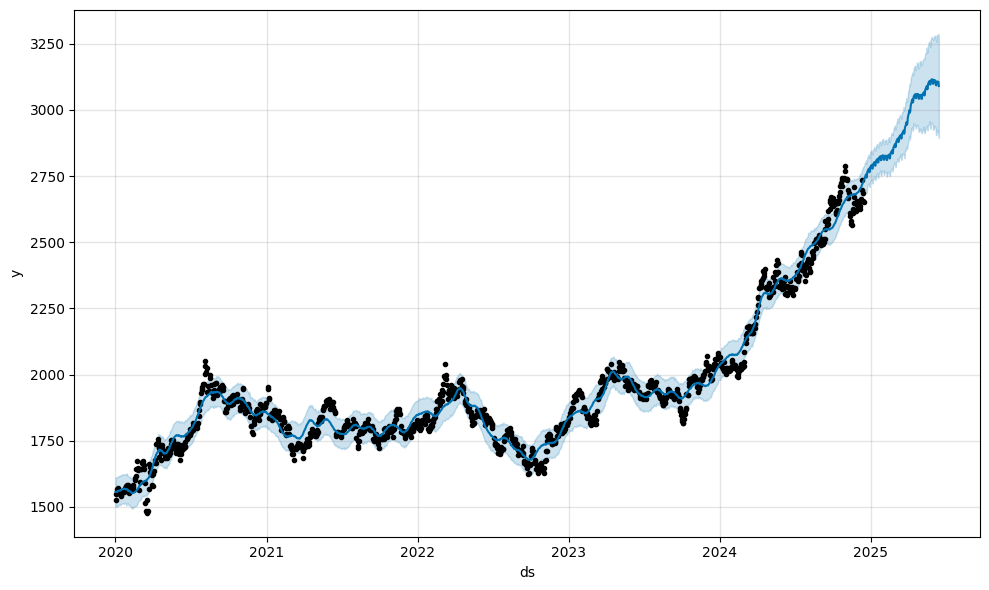
**Insights**:

* **Prophet** achieved exceptional performance with **R² = 0.973**, indicating it explains 97.3% of variance in gold prices.
* **AutoTS** failed terribly (R² < 0), suggesting its forecasts are worse than a simple mean baseline.

## **Line Charts: Actual vs. Predicted Values**

**Prophet Forecast**:

* **Trend Alignment**: Forecasted values closely track actual closing prices (Figure 1), capturing both upward and downward trends.
* **Uncertainty Intervals**: Narrow confidence bands (80% interval) reflect high model confidence.



A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

AI-generated content may be incorrect.

**AutoTS Forecast**:

* **Divergence**: Forecasted values (dashed red line in Figure 2) deviate sharply from actuals post-2024, indicating poor generalizability.
* **Volatility Ignorance**: Fails to account for sudden price fluctuations (e.g., 2023 dip).

A graph showing a graph of value

AI-generated content may be incorrect.

## **Model Comparison and Insights**

**Prophet Strengths**:

1. **Seasonality Decomposition**: Components plot reveals:
   * **Weekly Pattern**: Minor fluctuations (e.g., lower prices mid-week).
   * **Yearly Seasonality**: Peaks in Q4, likely tied to market demand cycles.
2. **Robustness**: Handles missing data and outliers effectively.

**AutoTS Limitations**:

1. **Hyperparameter Constraints**: Limited tuning (max\_generations=10) led to suboptimal model selection.
2. **Data Compatibility**: Daily frequency assumption may conflict with irregular gold futures trading patterns.
3. **Ensemble Failure**: The "simple" ensemble could not mitigate poor individual model performance.

## **Recommendations for Further Improvement**

**For Prophet**:

1. Test **custom seasonality** (e.g., quarterly financial cycles) to enhance interpretability.
2. Integrate **external regressors** (e.g., inflation rates, USD index) to capture macroeconomic drivers.

**For AutoTS**:

1. Use **model\_list='superfast'** or include Prophet as a candidate model.
2. Adjust frequency to match inferred data intervals (e.g., business days).
3. Increase num\_validations to improve cross-validation reliability.
4. Increase the size of the data (rows).

## **Conclusion**

* **Prophet** is **highly recommended** for gold futures forecasting due to its precision and interpretability.
* **AutoTS** requires significant parameter adjustments and data alignment to be viable.
* Future work could integrate external variables (e.g., inflation rates) into Prophet for enhanced robustness.

# **References**

* 1. Jain, A. and Gupta, C. (2018). *Fuzzy Logic in Recommender Systems*. [online] Semantic Scholar. doi:https://doi.org/10.1007/978-3-319-71008-2\_20.
  2. GeeksforGeeks (2018). *Fuzzy Logic | Introduction*. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/fuzzy-logic-introduction/>.
  3. GeeksforGeeks. (2024). *Time Series Analysis and Forecasting*. [online] Available at: <https://www.geeksforgeeks.org/time-series-analysis-and-forecasting/>.
  4. Github.io. (2022). *autots.models package — AutoTS 0.6.17 documentation*. [online] Available at: <https://winedarksea.github.io/AutoTS/build/html/source/autots.models.html>
  5. GeeksforGeeks. (2023). *Explainable Artificial Intelligence(XAI)*. [online] Available at: <https://www.geeksforgeeks.org/explainable-artificial-intelligencexai/>.
  6. Roy, R. (2019). *Generative Adversarial Network (GAN)*. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/generative-adversarial-network-gan/>.
  7. GeeksforGeeks (2023). *What is Feature Engineering?* [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/what-is-feature-engineering/>.