Gold ETF Price Prediction

A Comparative Project of Baseline, Linear Models, a Random Forest, and an Ensemble of All Methods

by Luca Reina, Chuting Xu, Abdulgani Muhammedsani, Tawakalt Okunola
22nd December, 2024

1 Introduction

The **ultimate goal** of this project is to *build a predictive model for Gold ETF prices.* Rather than relying on classical time-series methodologies (e.g., *ARIMA*), we embed *temporal structure* by *shifting* each predictor (yesterday's values) into the current day's feature set. In other words, we treat *today's gold price as the response* and *all previous-day variables as predictors*, allowing us to apply **traditional supervised ML** techniques (Regression, Random Forest, etc.) to this "lagged" dataset.

Although such an approach is *not* uncommon in financial machine learning analysis, it does *not* fully address typical time-series issues such as stationarity, seasonality, or the dependence between consecutive days. Future work could adopt more specialized time-series frameworks if we wish to capture *long-term* cyclic patterns or ensure strict temporal independence. We discuss the data pre-processing steps in a later section.

To achieve these objectives, this work will:

- **Undertake a Data Exploration & Preliminary Analysis** of the raw dataset.
- **Execute Data Preprocessing** steps (e.g., rolling means, shifting predictors, removal of redundant columns).
- **Compare multiple models**—Baseline, Linear Regression (including subset selection), Lasso, Ridge, and a tuned Random Forest.
- **Explore a final Ensemble approach** to see if averaging predictions furthers predictive gains.

2 Data Exploration & Preliminary Analysis

2.1 Data Source (Kaggle)

This dataset is publicly available on Kaggle at: `Gold Price Prediction Dataset`. We downloaded it locally as `FINAL_USO.csv` for this project.

2.2 Dataset Structure

The raw CSV initially contained **1,718 rows (days)** spanning roughly **2011–2019**, with **80 columns** in total.

- Many columns were 0/1 "trend" indicators; after removing them, we ended up with **72 columns**.
- Weekends and about 120 business days were missing (likely holidays), which is typical for financial time series.

Gold ETF Columns

- **Date**: The trading date.
- **Open, High, Low, Close**: Daily open, high, low, and closing prices for the Gold ETF.
- **Adjusted Close**: The closing price adjusted for dividends, stock splits, etc.

- **Volume**: The number of shares traded.

Since *Adjusted Close* reflects real valuation changes after corporate actions, we use it as our primary forecasting target rather than the raw *Close*.

Additional Economic Indices Beyond the Gold ETF columns, the dataset provides multiple related market indices, such as:

- **S&P 500** (e.g., `SP_close`),
- **Dow Jones** (`DJ_close`),
- **Oil ETFs**,
- **Silver**,
- **Platinum**,
- **Palladium**,
- **Rhodium**,
- **US Dollar Index**,
- **EUR/USD**, etc.

Each typically has `Open, High, Low, Close, Volume` columns (some also had "trend" indicators). These indices can serve as additional predictors (features) for Gold ETF price modeling.

2.3 Selected Variables for Modeling

To reduce redundancy and avoid excessive *multicollinearity*, we focus on **Close** or **Price** columns only. Specifically:

- **Gold ETF Adjusted Close (Target)**: Reflects the post-split/dividend value of the ETF.
- **SP_close, DJ_close**: Major U.S. equity indices; gold may move inversely if market sentiment changes.
- **USDI_Price, EU_Price**: Currency measures (Dollar Index, EUR/USD); often inversely related to gold.
- **GDX_Close**: Gold Miners ETF; closely correlated with gold.
- **SF_Price**: Silver futures, another precious metal that often tracks gold.
- **PLT_Price, PLD_Price, RHO_PRICE**: Platinum, Palladium, Rhodium to cover broader precious metals.
- **USO_Close**: Oil ETF; energy prices can influence inflation expectations.
- **OF_Price, OS_Price**: Brent Crude, WTI Crude metrics; shifts in oil may affect broader commodity markets.

Reason for Selection:

- **Limited Redundancy:** Only using `Close/Price` avoids duplication (no `Open, High, Low`).
- **Market Diversity:** We include equity (S&P, Dow), currency (Dollar Index, EUR/USD), and other commodities (oil, silver, platinum, etc.).
- **Historical Correlations:** Literature shows gold often correlates (positively or negatively) with these assets.

2.4 Preliminary Findings

Descriptive Statistics

- We computed mean, std, min/max, median, and quartiles for the chosen columns.
- The Gold ETF's Adjusted Close ranged from about \$100 to \$173 over 2011–2019.

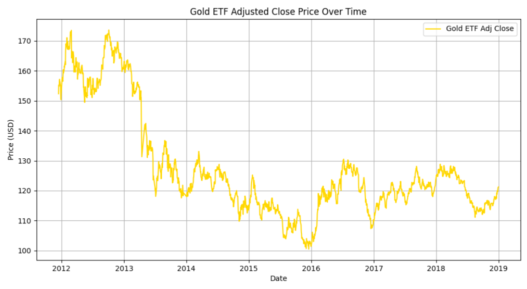

Figure 1: Time series of the Gold ETF's Adjusted Close (2011–2019).

Correlation Analysis

- Many features showed **strong correlations** (e.g., `GDX_Close` with `Adj Close`, `USDI_Price` inversely with `EU_Price`).
- This indicates *multicollinearity*, motivating subset selection or regularization for linear models.

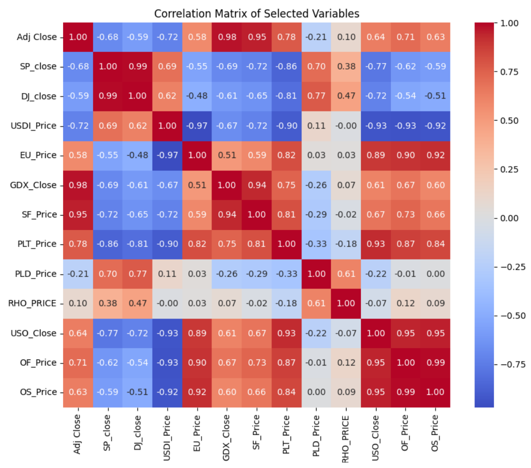

Figure 2: Correlation heatmap for selected variables, illustrating strong relationships.

These observations guided our data preprocessing steps and informed the final set of predictors for modeling.

2.5 Data Preprocessing

Creating Rolling Means For each date t, we compute two rolling averages of the Gold ETF's `Adj Close`:

- A 7-day rolling mean (using the current day and the previous 6 days).

- A 30-day rolling mean (using the current day and the previous 29 days).

These rolling means help capture short-term (7-day) fluctuations and smooth out monthly-scale (30-day) noise. Any rows that lack enough historical data (e.g., the first 6 or 29 days, respectively) are removed.

Shifting Features by One Record After computing rolling means, we shift all predictors (including the new rolling averages) so each row contains "yesterday's" data. For instance, if `Adj Close` on day t was \$120.00, then the row for day $t+1$ will store `Adj_Close_prev` = \$120.00. This ensures each row uses only prior information when predicting the current day's `Adj Close`.

Resulting Dataset Shape Once the necessary rows with insufficient historical data are dropped and all features are shifted by one record, the final dataset ends up with **1,688 rows** and **16 columns**. Each row includes:

Adj Close (target), Adj_Close_prev, SP_close_prev, DJ_close_prev, USDI_Price_prev, EU_Price_prev, GDX_Close_prev, SF_Price_prev, PLT_Price_prev, PLD_Price_prev, RHO_PRICE_prev, USO_Close_prev, OF_Price_prev, OS_Price_prev, Adj_Close_7d_prev, Adj_Close_30d_prev.

3 Modeling Approach

3.1 Evaluation Metric: Mean Squared Error

Throughout the experiments, we use **Mean Squared Error (MSE)**. For predictions $\hat{y}_i$ and actual values y_i:

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

where a smaller MSE implies closer predictions to observed data.

3.2 Cross-Validation Setup (10-Fold)

We apply **10-fold cross-validation** to estimate generalization:

- The dataset is split into 10 roughly equal subsets (folds).
- For fold j, we train on the other 9 folds and measure MSE on fold j.
- The final estimate is the average of all fold MSEs:

$$\text{CV\_MSE} = \frac{1}{10}\sum_{j=1}^{10}\text{MSE}_j.$$

- We do *not* use a separate hold-out set, since the data is limited; training on more data is advantageous. However, this can make our final MSE estimate slightly optimistic.
- All `random_state=42` settings (KFold splits, RandomForest, Lasso, Ridge) ensure reproducibility.
- **Important Note on Time-Series Data:** 10-fold CV is *not* typically recommended for *true* time-series analysis because it can inadvertently use future data to predict the past. However, for this project, we treat each day's record as an *artificially* independent observation after shifting features by one day. If we intended to respect strict temporal ordering (e.g., no future leakage), a rolling or walk-forward validation scheme would be more appropriate.

3.3 Baseline Model

A **naive forecast** sets today's `Adj Close` to *yesterday's price*:

$$\hat{y}_t = y_{t-1}.$$

This baseline provides a reference: any model failing to beat it in MSE likely doesn't justify its added complexity.

3.4 Random Forest Model Selection

We utilize a **Random Forest** regressor to handle nonlinearity and potential multicollinearity. Through `GridSearchCV`, we tested:

```
{
  'n_estimators': [100, 200, 300, 500],
  'max_depth': [None, 15, 25, 35],
  'max_features': ['sqrt', 0.5, 1.0],
  'min_samples_split': [2, 5, 10],
  'min_samples_leaf': [1, 2],
  'bootstrap': [True]
}
```

We also tried alternative grids (including larger `n_estimators`) but found the best set unchanged. Our final parameters:

$$\begin{aligned} \text{bootstrap} &= \text{True}, \\ \text{max\_depth} &= \text{None}, \\ \text{max\_features} &= \text{`sqrt'}, \\ \text{min\_samples\_leaf} &= 1, \\ \text{min\_samples\_split} &= 2, \\ \text{n\_estimators} &= 100. \end{aligned}$$

Hence, we fix these hyperparameters for the Random Forest in our study.

3.5 Best Subset (Exhaustive) Linear Regression

After preprocessing, we have **16** columns: 1 target + 15 potential predictors (_prev plus rolling means). We use `ExhaustiveFeatureSelector` (EFS) to evaluate *every possible* subset:

$$2^{15} - 1 = 32{,}767 \text{ subsets.}$$

Though computationally expensive, it ensures we find the subset with the lowest cross-validated MSE. EFS selected:

- Indices: `(0, 1, 2, 3, 5, 12)`
- Names: {`Adj Close_prev`, `SP_close_prev`, `DJ_close_prev`, `USDI_Price_prev`, `GDX_Close_prev`, `OS_Price_prev`}

Hence, our **Best Subset LR** uses these six lagged features and achieves the best MSE among subsets tested.

3.6 Linear Regression with Lasso & Ridge (with Scaling)

To address *multicollinearity* and reduce overfitting, we tested **Lasso** (L1) and **Ridge** (L2), each with α tuned via `GridSearchCV`. We **standardize** features first (`StandardScaler`), so that all predictors have comparable scales. Each model solves:

$$\min_{\beta} \sum (y_i - \hat{y}_i)^2 \; + \; \alpha \, R(\beta),$$

where $R(\beta) = \|\beta\|_1$ for Lasso and $\|\beta\|_2^2$ for Ridge. We sampled α from 10^{-5} to 10^2 in a `logspace` of 20 values, training and validating in a 10-fold manner. Lasso can zero out some coefficients (feature selection), whereas Ridge shrinks but rarely eliminates them. After finding each best α, we retrain final Lasso/Ridge. Scaling ensures consistent penalty application across features of differing magnitudes.

3.7 Ensemble (Averaging) of RF, LR, Lasso, & Ridge

Finally, we blend the predictions from **Random Forest**, **Linear Regression** (best subset), **Lasso**, and **Ridge** by averaging:

$$\hat{y}_t^{(\text{ensemble})} = \frac{1}{4}\Big[\hat{y}_t^{(\text{RF})} + \hat{y}_t^{(\text{LR})} + \hat{y}_t^{(\text{Lasso})} + \hat{y}_t^{(\text{Ridge})}\Big].$$

This capitalizes on each model's distinct error patterns, often yielding improved accuracy. We compute this *ensemble* MSE via the same 10-fold approach.

4 Results & Conclusion

4.1 Final MSE Comparison

We now compare each method's mean squared error (**MSE**) under 10-fold CV:

Baseline (Prev Day) :	1.5458,
Random Forest (Fixed) :	1.4878,
Best Subset LR :	1.5255,
Lasso ($\alpha = 0.02069$) :	1.5404,
Ridge ($\alpha = 0.11288$) :	1.5405,
Ensemble (Avg of 4) :	1.4404.

- **All main models** (RF, Best Subset LR, Lasso, Ridge) surpass the baseline MSE of 1.5458, indicating that more sophisticated techniques outperform the naive "yesterday's price."
- **Random Forest** leads individually (1.4878).
- **Best Subset LR** (1.5255) also improves over baseline but still ranks below RF.
- **Lasso (1.5404)** vs. **Ridge (1.5405)** yield nearly identical results, suggesting the data does not strongly favor L1-based zeroing; both forms of shrinkage produce comparable outcomes.
- **Ensemble (1.4404)** achieves the best MSE overall, demonstrating that combining these models can produce additional gains when their error patterns are not perfectly correlated.

4.2 Feature Importance

We evaluated each predictor's contribution using two approaches:

1. **Random Forest Importance:** We retrieve the built-in `feature_importances_` from our Random Forest model, indicating which variables reduce node impurity the most across all trees.
2. **Model Coefficients (Linear Methods):** For Linear Regression, Lasso, and Ridge, we re-train each model (after hyperparameter tuning or best-subset selection) on the entire dataset and inspect its coefficients. Sorting coefficients by absolute value reveals which features have the greatest effect on predictions.

Tables 1–4 list the top predictors identified by these methods, providing a concise view of which variables most strongly influence our models.

Random Forest Feature Importances.

Table 1: Random Forest: Top Feature Importances (Truncated)

Feature	**Importance**
`Adj_Close_7d_prev`	0.1835
`Adj_Close_30d_prev`	0.1762
`SP_close_prev`	0.1390
`GDX_Close_prev`	0.1239
`Adj Close_prev`	0.1214
`DJ_close_prev`	0.0984

Linear Regression (Best Subset) Coefficients.

Table 2: Best Subset LR: Top Coefficients (Truncated)

Feature	**Coefficient**	**|Coef.|**
`Adj Close_prev`	0.9392	0.9392
`GDX_Close_prev`	0.0773	0.0773
`USDI_Price_prev`	-0.0316	0.0316
`SP_close_prev`	-0.0297	0.0297

Lasso Coefficients (L1).

Table 3: Lasso: Top Nonzero Coefficients (Truncated)

Feature	**Coefficient**	**|Coef.|**
`Adj Close_prev`	16.0600	16.0600
`Adj_Close_7d_prev`	0.5675	0.5675
`GDX_Close_prev`	0.4471	0.4471

Ridge Coefficients (L2).

Table 4: Ridge: Top Coefficients (Truncated)

Feature	**Coefficient**	**|Coef.|**
`Adj Close_prev`	15.1861	15.1861
`SP_close_prev`	-1.2118	1.2118
`Adj_Close_7d_prev`	1.1689	1.1689

Overall Summary

- As seen in Table 2, linear models (Linear Regression, Lasso, Ridge) consistently assign the largest coefficient to the previous day's `Adj Close`, making it the strongest indicator of today's price. This reflects the fact that linear models tend to favor predictors that exhibit high correlation with the response.
- From Table 1, Random Forest places more emphasis on rolling averages (`Adj_Close_7d_prev`, `Adj_Close_30d_prev`) alongside `Adj Close_prev`, likely exploiting nonlinear relationships in the data.
- Other features, such as `SP_close_prev` and `GDX_Close_prev`, offer additional but smaller predictive value across all models (Tables 3 and 4).

4.3 Discussion & Concluding Remarks

All four advanced models consistently outperform the baseline, indicating that added complexity and thoughtful feature selection indeed pay off. Among the individual models, **Random Forest** achieves the lowest MSE, likely due to its ability to capture nonlinear interactions. Meanwhile, **Lasso** and **Ridge** perform similarly, suggesting that in this dataset, L1-based feature selection does not substantially outperform L2 shrinkage.

However, the **Ensemble** model produces the overall best MSE (1.4404), highlighting the advantage of combining predictions when models exhibit complementary error patterns. This result underscores the value of ensembling strategies in predictive analytics.

Time-Series vs. Traditional ML. In this project, we created lagging features (yesterday's values) and treated each day as an independent observation, which is a common but *simplified* approach in financial machine learning. While this design allows us to use traditional supervised learning techniques (including 10-fold cross-validation), it does not fully address stationarity or seasonality, which are often crucial in time-series forecasting. In a strict time-series setting, a rolling (walk-forward) validation method is typically preferred to avoid any look-ahead bias.

Because our dataset is relatively small, we did *not* maintain a separate hold-out set; instead, we relied on 10-fold CV for both hyperparameter tuning and performance estimation. This practice can introduce slight optimism in the reported errors, but is commonly used when data is limited. All random states (`random_state=42`) were fixed for reproducibility.

Potential Next Steps.

- **Extended Data**: Incorporate additional macroeconomic indicators or alternative features to provide the models with more predictive signals.
- **Enhanced Validation**: Use nested cross-validation or introduce a dedicated hold-out set if more data becomes available, potentially improving the reliability of performance estimates.
- **Advanced Time-Series Methods**: Investigate specialized time-series frameworks (e.g., ARIMA, SARIMA, or state-space models) or adopt a strict rolling-window validation scheme to respect temporal ordering.
- **Ensemble Refinements**: Explore weighted or stacked ensemble methods and conduct more extensive hyperparameter searches to further boost accuracy.

In summary, our results confirm that lagging features, rolling averages, and ensemble-based strategies are promising for daily Gold ETF forecasting. Future work could refine the temporal structure and introduce a broader range of economic indicators to bolster model performance in dynamic market environments.