PROJECT REPORT

Master of Science in Quantitative Finance

AY2023-24 QF624 MACHINE LEARNING AND FINANCIAL APPLICATIONS

Date: 17-Jun-2024

Instructor:

Prof. LIU Peng

By:

LIM Eugene (01241520)

DENG Ke (01494971)

TAY Samuel (01473398)

XIA Tian (01476942)

Table of Contents

Part 1: Executive Summary

The motivation behind this project is to see if we could employ various machine learning techniques to accurately predict the movements of 3 diverse stock index prices. Our portfolio design strategically integrates three core components: it is composed of three ETFs, utilizes stacking models to forecast price trends of each ETF, effectively creating three distinct investment strategies, and employs Particle Swarm Optimization (PSO) to optimally allocate weights across these strategies. This approach ensures a robust alignment of predictive accuracy with dynamic weight adjustment, maximizing the overall risk-adjusted returns of the portfolio.

Part 2: Economic Significance

Accurate predictions of market movements and returns, effective risk management and robust portfolio management are critical for financial institutions, asset managers, and individual investors to make informed investment decisions and optimize their portfolios. This project aims to expand on a breadth of predictive models to specifically elucidate clearer insights into the mechanics of the commodity markets as well as give portfolio managers an idea on how to allocate their capital given these models.

Below, we list some ways in which we hope that market participants can benefit from our work.

1. **Investment Strategy Optimization**:
   - **Problem**: Investors can determine the best times to buy or sell ETFs to maximize returns and minimize risks.
   - **Solution**: By predicting whether an ETF will move up or down, the models can provide actionable insights for timing market entries and exits, thereby enhancing the overall investment strategy.
2. **Risk Management**:
   - **Problem**: Financial institutions need to manage and mitigate risks associated with market volatility and downturns.
   - **Solution**: Predictive models can identify potential downturns in ETF prices, allowing risk managers to implement hedging strategies or adjust asset allocations to protect portfolios from significant losses.
3. **Portfolio Allocation**:
   - **Problem**: Determining the optimal allocation of assets in a portfolio is challenging, especially in volatile markets.
   - **Solution**: Accurate predictions of ETF movements can inform dynamic asset allocation strategies, ensuring a balanced and diversified portfolio that adapts to market conditions.
4. **Trading Algorithms**:
   - **Problem**: Traders require automated systems that can execute trades based on real-time market data and predictions.
   - **Solution**: The models developed in this project can be integrated into trading algorithms to automatically execute buy or sell orders based on predicted market movements, improving trading efficiency and profitability.
5. **Economic Forecasting**:
   - **Problem**: Understanding the broader economic trends and their impact on financial markets is crucial for strategic planning.
   - **Solution**: By incorporating various economic indicators (e.g., unemployment rates, interest rates) into the predictive models, the project can provide insights into how these factors influence ETF performance, aiding in macroeconomic forecasting and strategic decision-making.

Also, below are the benefits and values to various stakeholders.

1. **Investors**:
   - **Benefit**: Improved decision-making regarding ETF investments, leading to higher returns and reduced risks.
   - **Value**: Enhanced portfolio performance and better alignment with investment goals.
2. **Financial Institutions**:
   - **Benefit**: More effective risk management and portfolio optimization strategies.
   - **Value**: Increased stability and profitability of managed funds and investment products.
3. **Traders**:
   - **Benefit**: Advanced trading algorithms that leverage predictive models for timely market actions.
   - **Value**: Higher trading efficiency, reduced manual intervention, and increased profitability.
4. **Economic Analysts**:
   - **Benefit**: Better understanding of the relationship between economic indicators and market performance.
   - **Value**: More accurate economic forecasts and informed strategic planning.

Part 3: Data Cleaning and Feature Engineering

3.1 Data Processing and Correlation Matrix

Our data methodology structure is as follows:

1. **Invesco DB Agricultural Funds (DBA)**: this is a fund comprising of one or more underlying commodities, and it is intended to reflect the economic performance of the agricultural sector. The fund pursues its investment objectives by investing in a portfolio of exchange-traded futures.
2. **SPDR Gold Shares (GLD)**: this Trust holds gold bars from time to time, issues Baskets in exchange for deposits of gold and distributes gold in connection with redemptions of Baskets. The investment objective of the Trust if for the Shares to reflect the performance of gold bullion price, less the Trust expenses.
3. **USO (United States Oil Funds)**: USO invests primarily in futures contracts for light, sweet crude oil, diesel-heating oil, gasoline, natural gas, and other petroleum-based fuels.

We obtained the price data for the period of January 01, 2008, to March 31, 2024, from Yahoo! Finance, giving us a total of 4,039 data points.

For the explanatory data, we used a diverse set of data ranging from general economic indicators, uncertainty indicators, and market volatility indices in our study. The economic and uncertainty indicators data are obtained through US Federal Reserve Economic Data (FRED) website, whereas the volatility indices and other asset prices are obtained through Yahoo! Finance,

A. **General Economic Data**
   1. Industrial production data
   2. Unemployment rates
   3. 3-month T-bill rates

B. **Uncertainty Indicators**
   4. US Economic Policy Uncertainty (EPU): this index is based on frequency of keyword searches in newspaper and media.
   5. US Equity Market Uncertainty (EMI): a measure from economic policy uncertainty websites.
   6. Infectious Disease Equity Market Volatility Tracker (EMV): a measure of market uncertainty due to infectious disease. And this indicator is useful as our data period includes COVID-19 period, where market could have entered a different regime.

C. **Market Volatility Indices**
   7. CBOE Volatility Index (VIX): an uncertainty indicator of US stock market based on S&P 500 index.
   8. CBOE Oil Price Volatility Index (OVX): a measure of oil market volatility using option prices on crude oil futures
   9. NASDAQ 100 Volatility Index (VXN): another uncertainty indicator on technology stock market volatility, similar to VIX but for NASDAQ.

D. **Other Asset Prices**
   10. iShares MSCI Peru ETF (EPU).
   11. ADX Energy Ltd.

The above data should give a general indication of the economy and market as a whole and should be useful to a certain extent in predicting the price movement of the 3 indices. One reason for predicting the price movements of the 3 indices is also to see if the movements are heavily dependent on a particular subset of explanatory data.

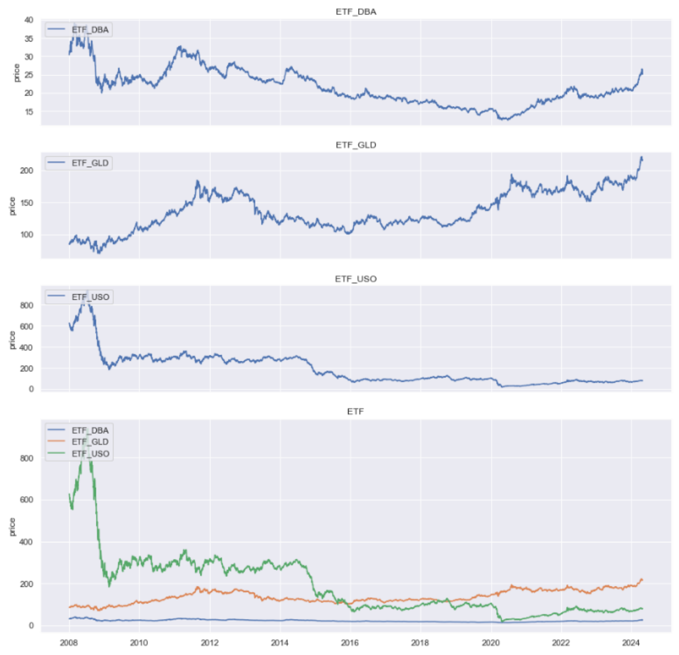


Figure 1(a): *Plots of the prices of the 3 subject indices, individual to show price trends and levels. The last subplot is the combined plot of the 3 indices to show their relative levels.*

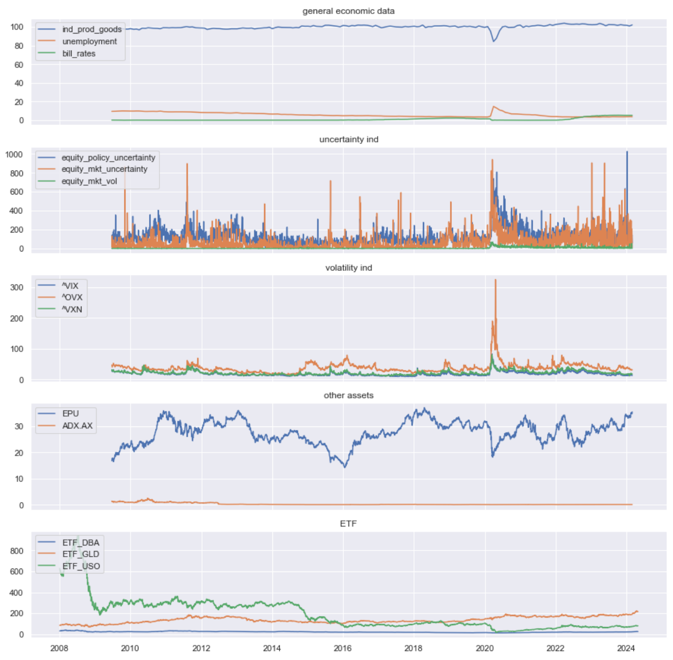


Figure 1(b): *Plots of explanatory data against time. First subplot contains the general economic data; second subplot contains the uncertainty indicators; and the third subplot contains the market volatility indices. The fourth subplot contains the prices of other assets. The fifth subplot contains the prices of the 3 subject indices.*

Other than market explanatory data, we obtained various technical indicators created from the index price data themselves. These should provide us with some insight as to any momentum in the price movements or volatilities:

A. **Lags**: 1- to 5-period lags
B. **Averages**
   i. Simple moving averages of 50-, and 200-period rolling windows,
   ii. Exponential moving averages of 10-, 30-, and 200-period rolling windows.
C. **Trending signals**: moving average convergence divergence (MACD), and relative strength indicator (RSI): 10-, 30-, and 200-period window.
D. **Stochastic oscillators**: 10-, 30-, and 200-period window.
E. **Rates of changes**: 10-, and 30-period window.

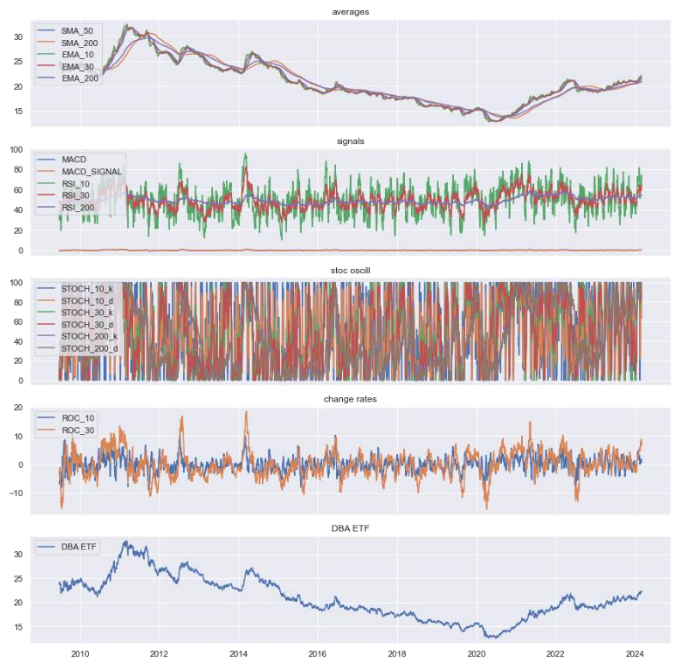


Figure 1(c): *Plots of various indicators for DBA ETF.*

After joining all the data into a single dataframe and generating all the technical indicators as listed in Section 1 above, we remove all the time points without data, and we are left with 3,594 datapoints for our study.

Before we proceed with Feature engineering, we generate the correlation heatmaps of the explanatory data and technical indicators for each of the ETF indices.

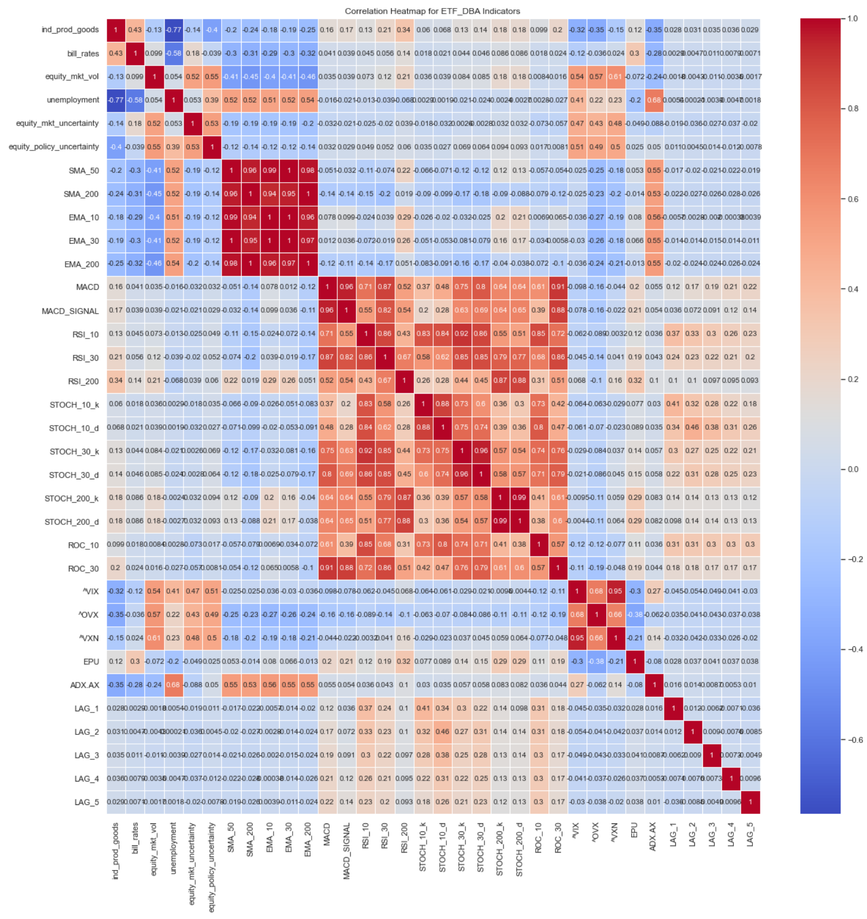


Figure 3(a): *Correlation heatmap of explanatory data and technical indicator of DBA.*

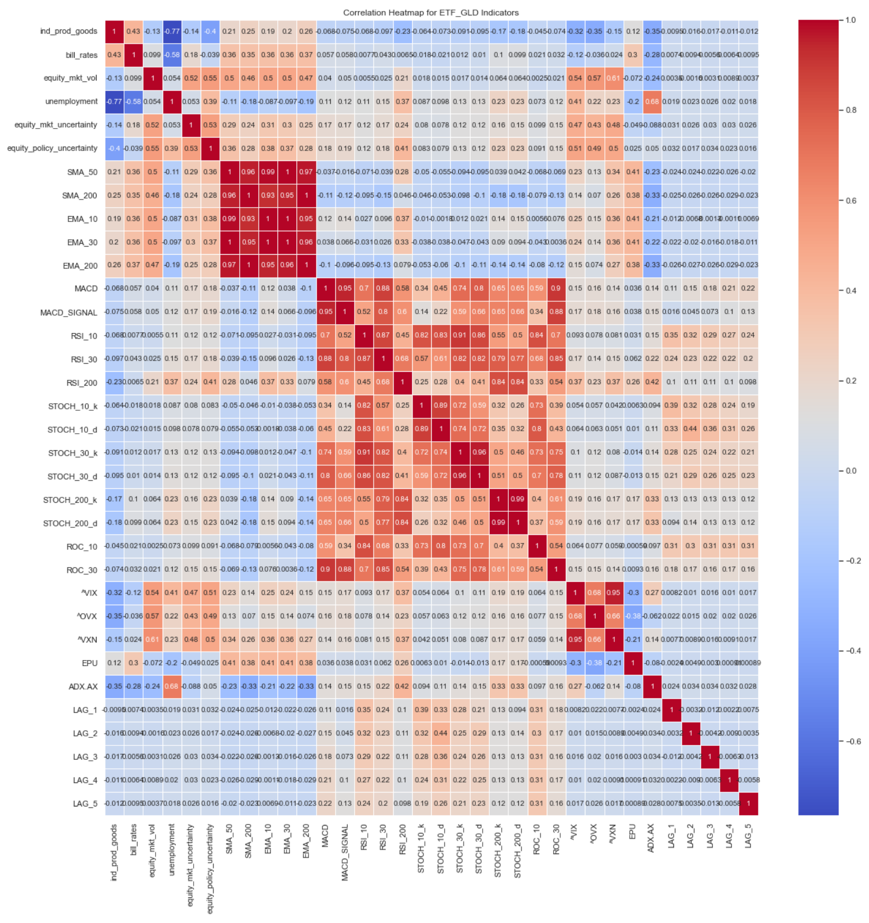


Figure 3(b): *Correlation heatmap of explanatory data and technical indicator of GLD.*

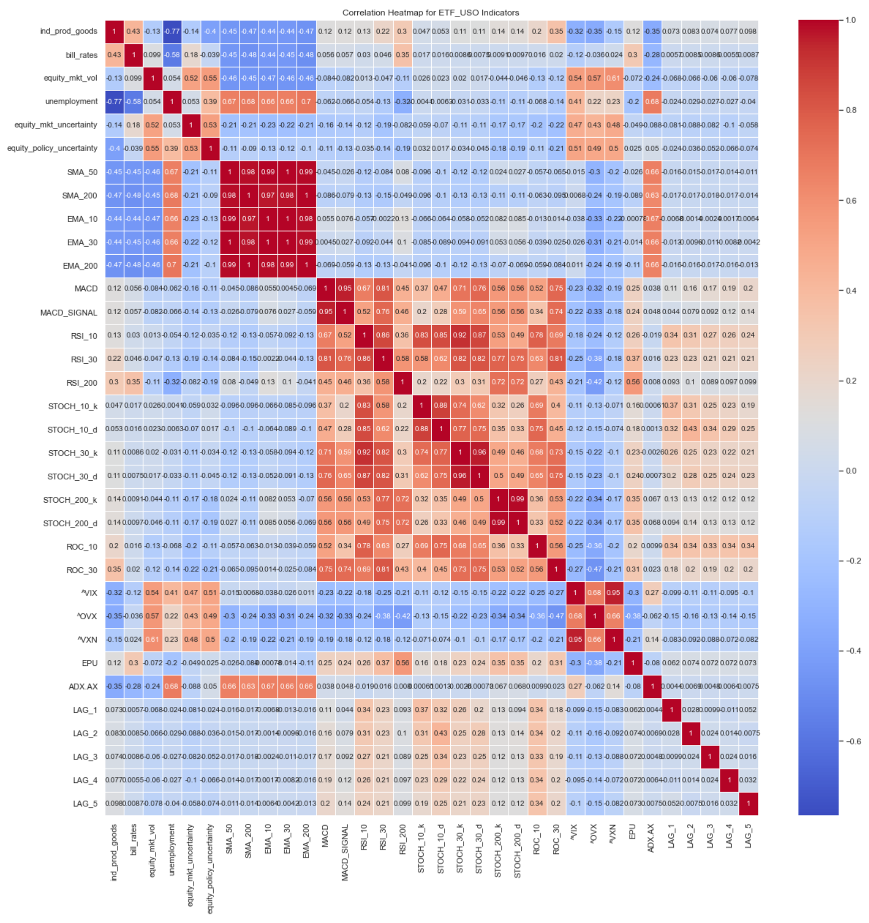


Figure 3(c): *Correlation heatmap of explanatory data and technical indicator of USO.*

In all the 3 correlation plots, we see strong positive correlation amongst the technical indicators of the same grouping, which is not surprising. And subsequently, during our feature selection process with Principal Component Analysis (PCA), most of the highly correlated features would be highlighted for elimination to prevent overfitting.

Also, looking converting the daily price movement to a binary output, i.e., 1 for positive price movements, and 0 for negative or no price movements, we first assess if the proportions of days with positive price movements and negative price movements.

DBA	
Positive Movement	1,867
Negative Movement	1,727

GLD	
Positive Movement	1,877
Negative Movement	1,717

USO	
Positive Movement	1,859
Negative Movement	1,735

Figure 3(c): *Tallies of data points with positive and negative price movement of the 3 indices.*

3.2 Feature Engineering

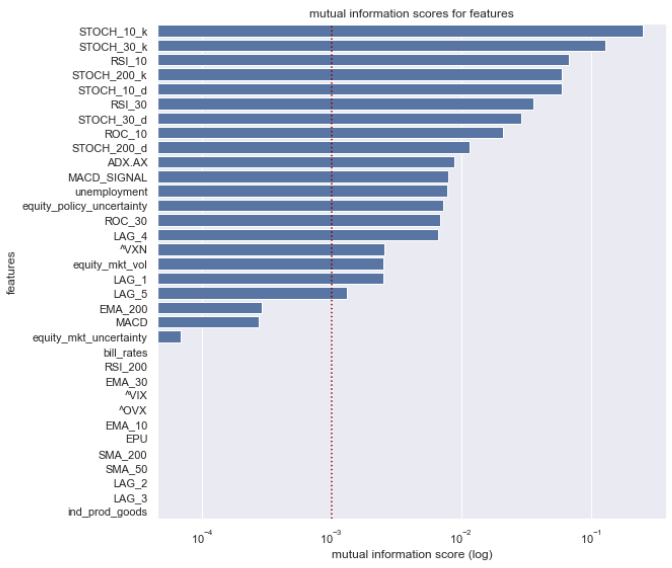
Principal Component Analysis (PCA) is a dimensionality reduction technique that transforms the data into a new coordinate system where the greatest variances by any projection of the data come to lie on the first coordinate (called the first principal component), the second greatest variances on the second coordinate, and so on. This is useful in reducing the number of features in the dataset while retaining most of the variation present in the original dataset. PCA helps in tackling issues related to multicollinearity and improving computational efficiency.

Mutual Information (MI) from the field of information theory measures the amount of information obtained about one random variable through another random variable. For feature selection, the Mutual Information Score (MIC) quantifies the dependency between each feature and the target variable. A higher MI score indicates a higher dependency and relevance of the feature to the target variable. This method is particularly useful for capturing non-linear relationships between features and the target.

In this study, we employed a systematic approach to feature selection utilizing both PCA and MIC. Initially, we prepared three datasets corresponding to target variables `'DBA'`, `'GLD'`, and `'USO'`, and divided each dataset into subsets before and after the application of PCA. The PCA was applied to reduce dimensionality, retaining 80% of the variance to preserve the essence of the original data. This resulted in transformed datasets with fewer features. Subsequently, we calculated the MIC for each feature in both the original and PCA-transformed datasets, setting a threshold of 0.001 to select relevant features.

Visualization of MIC scores on a logarithmic scale was conducted to highlight differences, resulting in six plots: three for the original datasets and three for the PCA-transformed datasets.

Analysis of the original datasets revealed that most features exhibited significant MIC scores, indicating high relevance to the target variables. For the DBA dataset, many features surpassed the threshold, making them crucial for model training. Similar patterns were observed in the GLD dataset, where several features had high MIC scores. The USO dataset displayed a mix of highly and moderately relevant features, with the threshold line distinguishing essential features. In the PCA-transformed datasets, the DBA PCA dataset showed substantial MIC scores for many features, confirming that PCA retained valuable information relevant to the target variable. The GLD PCA dataset's high MIC scores mirrored those of the original dataset, validating PCA's effectiveness. The USO PCA dataset also demonstrated significant MIC scores for several features, indicating that dimensionality reduction did not compromise essential data attributes.



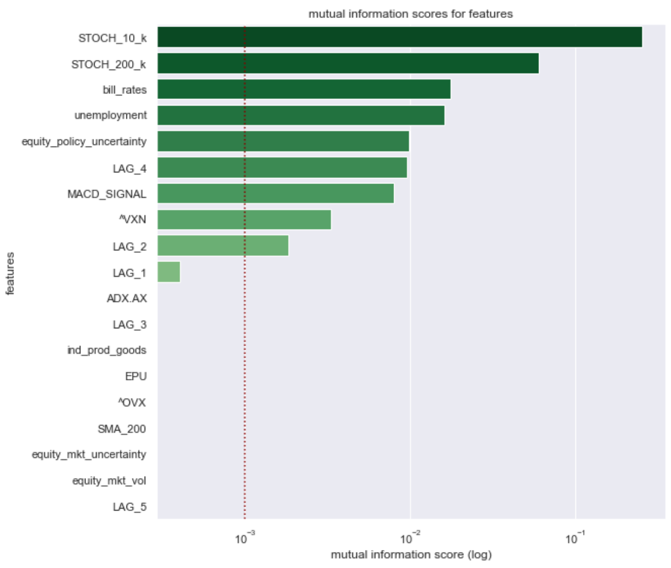
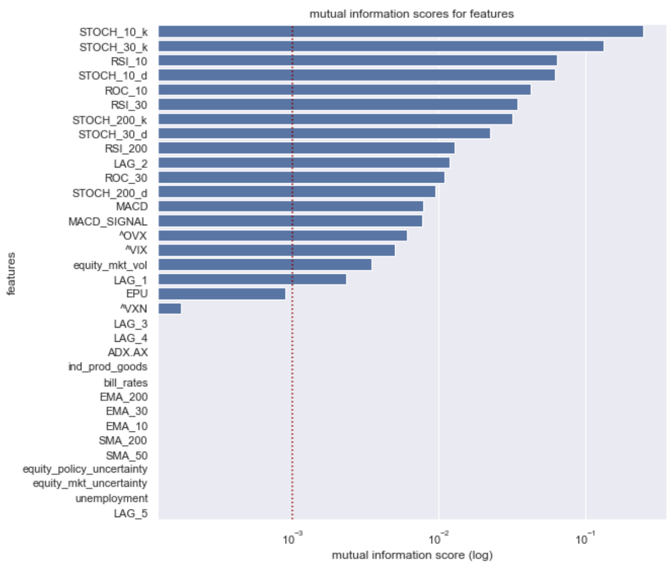


Figure 3(d): *Plots of MIC scores for all the features in predicting the price movement to DBA.*

Figure 3(e): *Plots of MIC scores for the features after removing those highly correlated ones as highlighted by PCA in predicting the price movement to DBA.*



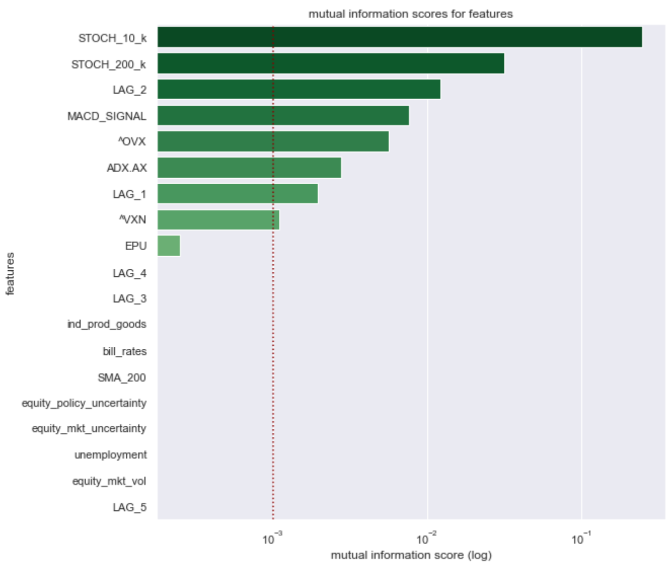
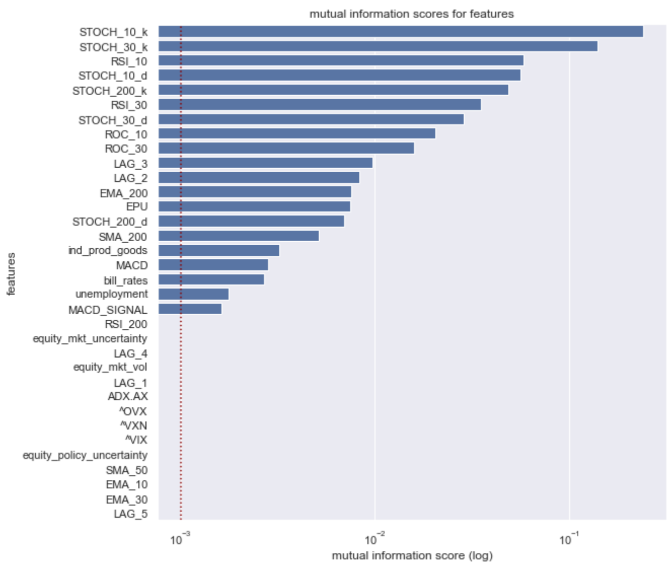


Figure 3(f): *Plots of MIC scores for all the features in predicting the price movement to GLD.*

Figure 3(g): *Plots of MIC scores for the features after removing those highly correlated ones as highlighted by PCA in predicting the price movement to GLD.*



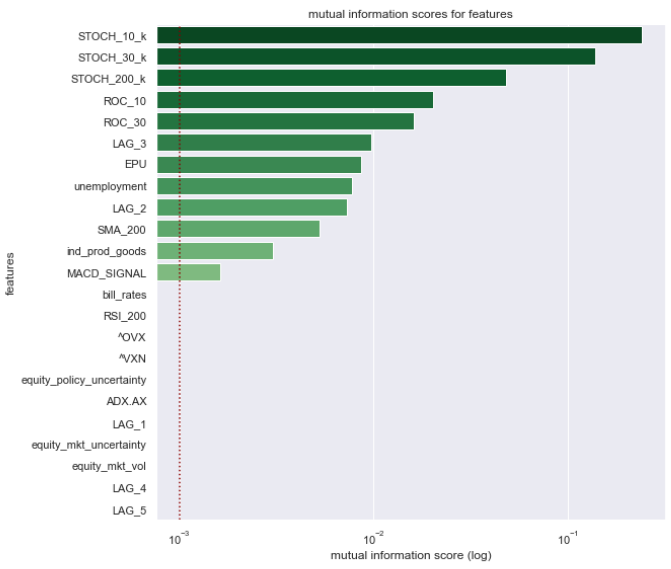


Figure 3(h): *Plots of MIC scores for all the features in predicting the price movement to USO.*

Figure 3(i): *Plots of MIC scores for the features after removing those highly correlated ones as highlighted by PCA in predicting the price movement to USO.*

Through the combined use of PCA and MIC, we effectively reduced the dimensionality of the datasets while preserving crucial information. The MIC scores provided clear indications of the most relevant features to the target variables. Focusing on these features enables streamlining the model training process and enhancing performance. The visualizations facilitated the easy identification of key features, supporting better decision-making for feature selection in machine learning models. This approach underscores the utility of PCA and MIC in feature selection, offering a balanced trade-off between dimensionality reduction and retention of informative features.

Part 4: Model Selection and Hyperparameter Tuning

4.1 Model Selection

4.1.1 Stacking Model

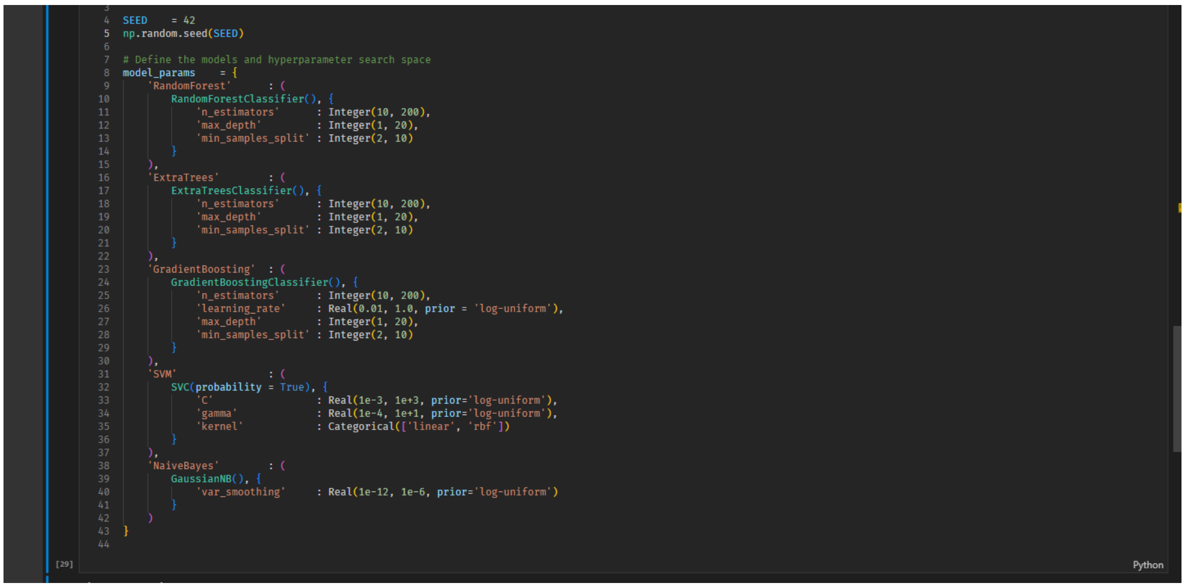
Stacking, or stacked generalization, is an ensemble learning technique that aims to improve the predictive performance by combining multiple machine learning models. It involves training various base models on the same dataset and then using their predictions as input features for a higher-level meta-model, which makes the final predictions. This hierarchical structure leverages the strengths of each model, thus mitigating individual model weaknesses and enhancing overall performance.

In this project, we implemented a stacking model framework to evaluate and enhance predictive performance across three datasets: DBA, GLD, and USO. The approach began by splitting each dataset into training, validation, and test sets. We trained five different base models: Random Forest Classifier (RF), Extra Trees Classifier (ET), Gradient Boosting Classifier (GB), Support Vector Machine (SVM), and Naive Bayes (NB). These base models were optimized using Bayesian optimization with cross-validation. Their predictions on the validation set were then used as input features for the meta-model. We chose AdaBoost Classifier as the meta-model, which was also tuned using Bayesian optimization to ensure the best performance.

4.1.2 Base Model

The base models selected for this project included:

1. **Random Forest Classifier (RF)**: an ensemble method that constructs multiple decision trees during training and outputs the mode of the classes (classification) or mean prediction (regression) of the individual trees.
2. **Extra Trees Classifier (ET)**: like Random Forest, but with a more random process of selecting cut-points, which typically leads to a reduction in overfitting.
3. **Gradient Boosting Classifier (GB)**: Builds an ensemble of trees sequentially, with each tree attempting to correct the errors of the previous one, thus improving the model's performance.
4. **Support Vector Machine (SVM)**: A powerful classification method that seeks the optimal hyperplane that maximizes the margin between classes.
5. **Naive Bayes (NB)**: A simple probabilistic classifier based on Bayes' theorem, assuming independence between features.



```python
SEED    = 42
np.random.seed(SEED)

# Define the models and hyperparameter search space
model_params    = {
    'RandomForest'      : (
        RandomForestClassifier(), {
            'n_estimators'      : Integer(10, 200),
            'max_depth'         : Integer(1, 20),
            'min_samples_split' : Integer(2, 10)
        }
    ),
    'ExtraTrees'        : (
        ExtraTreesClassifier(), {
            'n_estimators'      : Integer(10, 200),
            'max_depth'         : Integer(1, 20),
            'min_samples_split' : Integer(2, 10)
        }
    ),
    'GradientBoosting'  : (
        GradientBoostingClassifier(), {
            'n_estimators'      : Integer(10, 200),
            'learning_rate'     : Real(0.01, 1.0, prior = 'log-uniform'),
            'max_depth'         : Integer(1, 20),
            'min_samples_split' : Integer(2, 10)
        }
    ),
    'SVM'               : (
        SVC(probability = True), {
            'C'                 : Real(1e-3, 1e+3, prior='log-uniform'),
            'gamma'             : Real(1e-4, 1e+1, prior='log-uniform'),
            'kernel'            : Categorical(['linear', 'rbf'])
        }
    ),
    'NaiveBayes'        : (
        GaussianNB(), {
            'var_smoothing'     : Real(1e-12, 1e-6, prior='log-uniform')
        }
    )
}
```

Figure 4(a): *Parameter set up for the various base models used in machine learning in our study.*

4.1.3 Meta Model

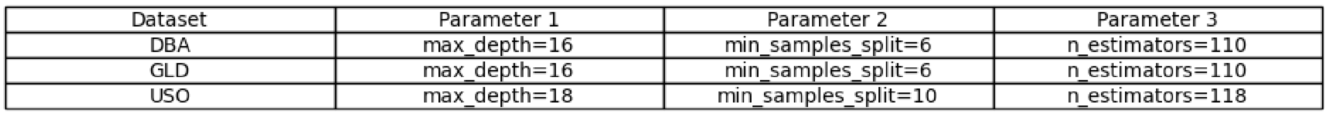
The meta-model in our stacking approach was the AdaBoost Classifier. AdaBoost, or Adaptive Boosting, combines multiple weak classifiers to create a strong classifier. It adjusts the weights of incorrectly classified instances, allowing subsequent classifiers to focus more on difficult cases, thereby improving the overall model accuracy.

The datasets were divided into three subsets: original features, PCA-transformed features, and MIC-selected features. Each subset was further split into training, validation, and test sets. We trained the base models on the training set and predicted on the validation set. The predictions were then used to train the meta-model. The final evaluation was performed on the test set, and the performance metrics, including accuracy, precision, recall, F1-score, and Cohen's kappa, were calculated.

4.2 Hyperparameter Tuning

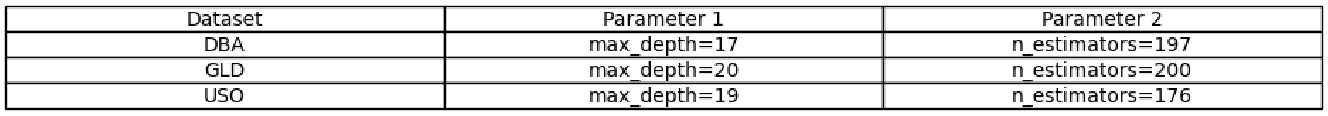
In this project, we used hyperparameter optimization to enhance the performance of several machine learning models. Hyperparameter optimization is crucial for improving model accuracy and robustness, as it involves fine-tuning model parameters to find the best combination that yields the highest performance. we employed Bayesian optimization via **`BayesSearchCV`** from the scikit-optimize library, which efficiently searches the hyperparameter space using Bayesian methods.

The data was split into training, validation, and test sets. we evaluated several models, including Random Forest, Extra Trees, Gradient Boosting, Support Vector Classifier, and Gaussian Naive Bayes. Each model was optimized using **`BayesSearchCV`** with 30 iterations of hyperparameter tuning and 3-fold cross-validation. The models were evaluated on the validation set using accuracy, F1 score, Cohen's kappa, precision, recall, and confusion matrix.



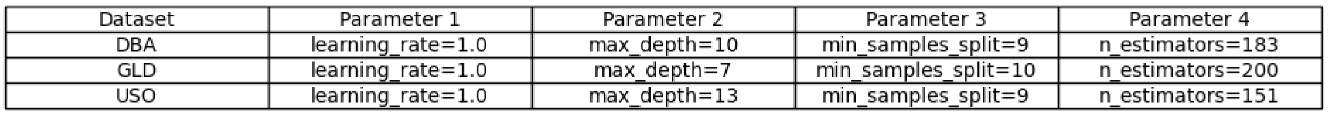
Dataset	Parameter 1	Parameter 2	Parameter 3
DBA	max_depth=16	min_samples_split=6	n_estimators=110
GLD	max_depth=16	min_samples_split=6	n_estimators=110
USO	max_depth=18	min_samples_split=10	n_estimators=118

Figure 4(b): *Optimal parameters for Random Forest in training phase for all 3 datasets.*



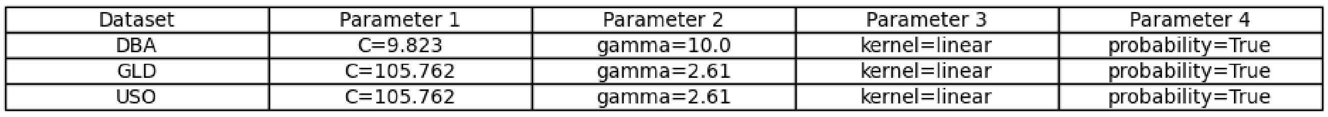
Dataset	Parameter 1	Parameter 2
DBA	max_depth=17	n_estimators=197
GLD	max_depth=20	n_estimators=200
USO	max_depth=19	n_estimators=176

Figure 4(c): *Optimal parameters for Extra Trees in training phase for all 3 datasets.*



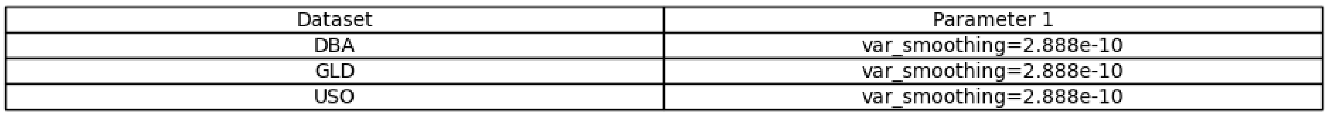
Dataset	Parameter 1	Parameter 2	Parameter 3	Parameter 4
DBA	learning_rate=1.0	max_depth=10	min_samples_split=9	n_estimators=183
GLD	learning_rate=1.0	max_depth=7	min_samples_split=10	n_estimators=200
USO	learning_rate=1.0	max_depth=13	min_samples_split=9	n_estimators=151

Figure 4(d): *Optimal parameters for Gradient Boosting in training phase for all 3 datasets.*



Dataset	Parameter 1	Parameter 2	Parameter 3	Parameter 4
DBA	C=9.823	gamma=10.0	kernel=linear	probability=True
GLD	C=105.762	gamma=2.61	kernel=linear	probability=True
USO	C=105.762	gamma=2.61	kernel=linear	probability=True

Figure 4(e): *Optimal parameters for Support Vector Classifier in training phase for all 3 datasets.*



Dataset	Parameter 1
DBA	var_smoothing=2.888e-10
GLD	var_smoothing=2.888e-10
USO	var_smoothing=2.888e-10

Figure 4(f): *Optimal parameters for Gaussian Naïve Bayes in training phase for all 3 datasets.*

The results of the hyperparameter optimization and model evaluation were processed and visualized for three datasets: DBA, GLD, and USO. Optimized models achieved varying degrees of performance improvements across different datasets. Gradient Boosting and Random Forest generally performed well, showing high accuracy and F1 scores. Support Vector Classifier also showed strong performance but was more sensitive to hyperparameter settings.

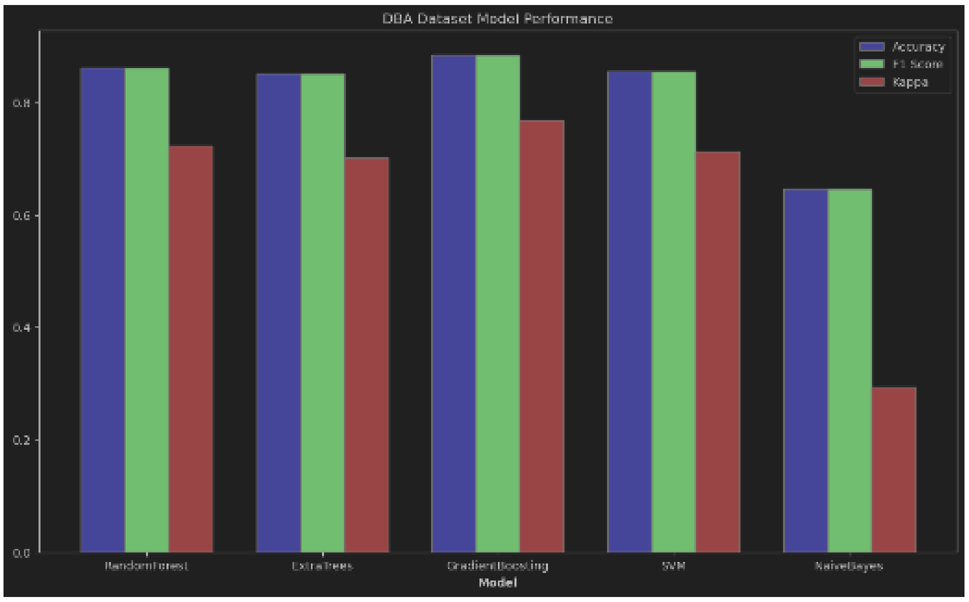


Figure 4(g): *Performance of various machine learning techniques in predicting DBA price movement in terms of accuracy, F1 score and kappa.*

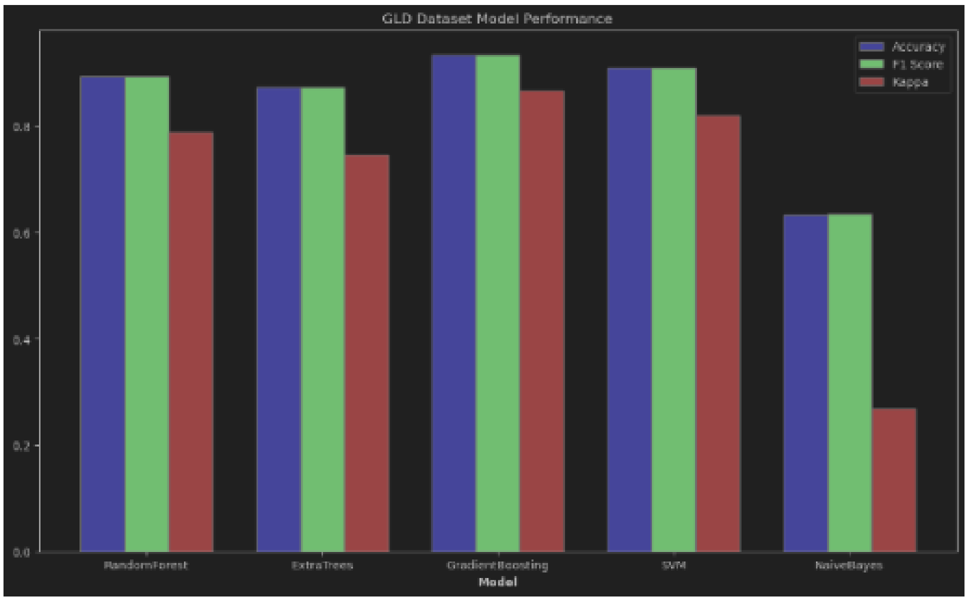


Figure 4(h): *Performance of various machine learning techniques in predicting GLD price movement in terms of accuracy, F1 score and kappa.*

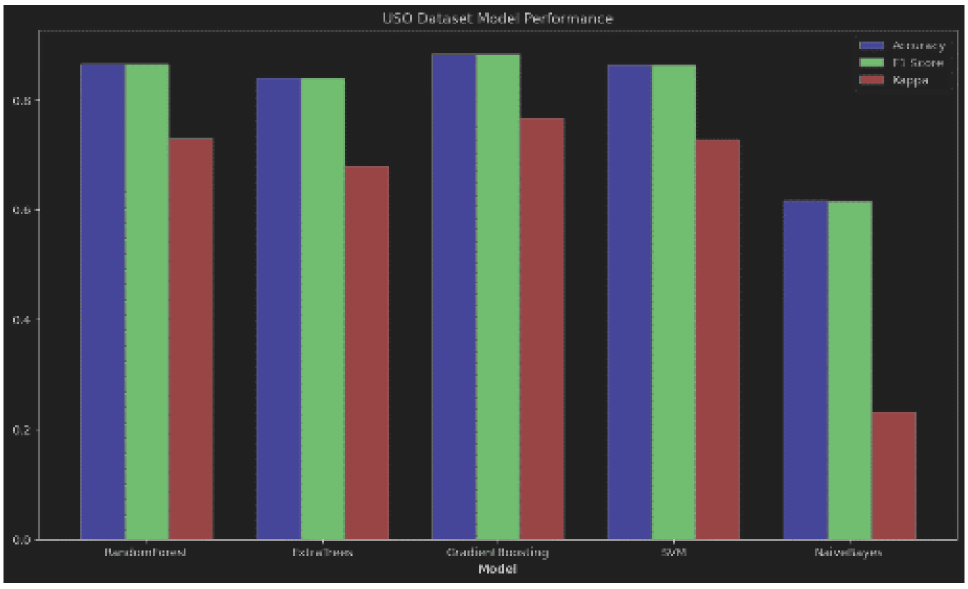


Figure 4(i): *Performance of various machine learning techniques in predicting USO price movement in terms of accuracy, F1 score and kappa.*

The hyperparameter optimization process significantly improved the performance of the machine learning models across different datasets. Bayesian optimization proved to be an efficient method for tuning hyperparameters, resulting in models with higher accuracy, precision, recall, F1 scores, and Cohen's kappa. The best models identified will be used for further analysis and application in the respective datasets.

Part 5: Results and Portfolio Optimization

Accuracy			
	DBA	GLD	USO
Random Forest	0.8757	0.8965	0.8583
Extra Trees	0.8574	0.8730	0.8417
Gradient Boosting	0.8817	0.9322	0.8878
SVM	0.8644	0.9148	0.8652
Naïve Bayes	0.6417	0.6270	0.6226
Ensemble Stacking	0.8846	0.9624	0.8804

Precision			
	DBA	GLD	USO
Random Forest	0.8757	0.8966	0.8588
Extra Trees	0.8574	0.8730	0.8427
Gradient Boosting	0.8818	0.9322	0.8884
SVM	0.8644	0.9148	0.8652
Naïve Bayes	0.6417	0.6285	0.6232
Ensemble Stacking	0.9036	0.9602	0.8929

Recall			
	DBA	GLD	USO
Random Forest	0.8757	0.8965	0.8583
Extra Trees	0.8574	0.8730	0.8417
Gradient Boosting	0.8817	0.9322	0.8878
SVM	0.8643	0.9148	0.8652
Naïve Bayes	0.6417	0.6270	0.6226
Ensemble Stacking	0.8547	0.9679	0.8737

The green row shows which model performed the best for a particular metric. While it might seem like an idea to simply pick the stacking model in future scenarios, we feel it is more prudent to select one of these simpler models, for ease of interpretation, faster computation, and less complexity in implementation and maintenance.

If a strategy were to be implemented based on only one technique, we feel that Gradient Boosting is the stronger candidate due its relatively high performance across all metrics while being relatively simpler to implement. In the real world, picking an ensemble of techniques or just one depends on circumstances like how mature a strategy or market is.

In our portfolio management strategy, we have adopted the Particle Swarm Optimization (PSO) algorithm to explore the optimal asset weight allocation for maximizing the Sharpe ratio of a portfolio.

Algorithm Description

Our implemented portfolio objective function calculates the expected return and volatility of the portfolio, thus deriving the negative Sharpe ratio. The goal in the PSO process is to minimize this negative value, which effectively maximizes the original Sharpe ratio.

PSO Algorithm Details

In the `pso_portfolio` function, we initialized a set of particles, each representing a set of asset weight allocations. These particles explore the multidimensional weight space, seeking to find a solution that optimizes the Sharpe ratio. The update of the particles' velocity and position depends on their own historical best position and the best position of the swarm, which guides the particles to explore new and potentially better regions.

Advantages Analysis

Particle Swarm Optimization offers significant advantages over traditional weight allocation methods:

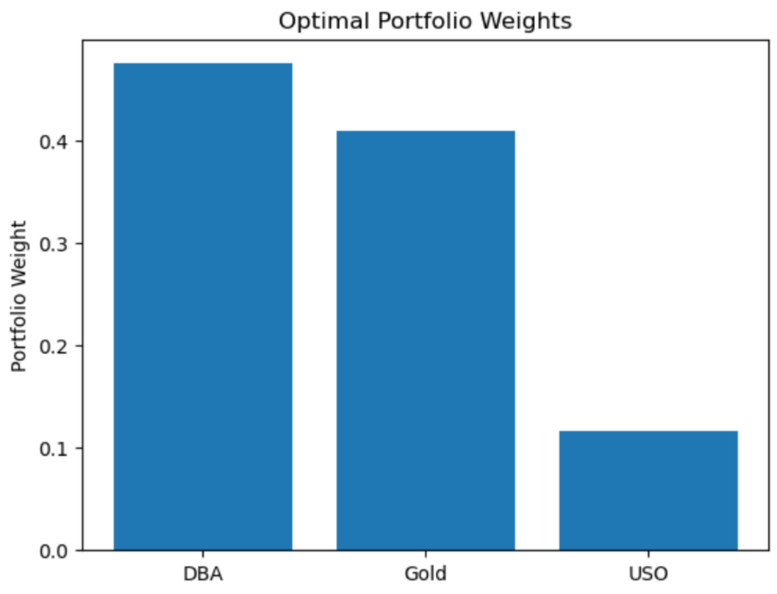
1. **Global Search Capability**: the collective intelligence of PSO enables it to avoid local optima and explore global optima, thus effectively improving the quality of the solution.
2. **Flexibility and Adaptability**: PSO does not rely on the derivatives of the objective function, making it effective in handling nonlinear optimization problems with complex constraints.
3. **Simplicity and Ease of Implementation**: the PSO algorithm is easy to implement, and its parameters are intuitive to adjust, making it easier to deploy and debug in real financial operations.
4. **Rapid Convergence**: PSO can reach a satisfactory solution within a fewer number of iterations, efficiently saving computational resources and time, enhancing decision-making speed.

Results and Display

Ultimately, the best weight configuration found by the algorithm, corresponding to the Sharpe ratio, is recorded and reported. We visually display the weight proportions of each asset in the optimal portfolio through a bar graph.

Finally, we concluded that the allocation ratios for the three asset strategies are 0.475, 0.409 and 0.116 respectively, and the Sharpe ratio of the final portfolio is 1.268

```
Portfolio Weights: [0.47501148 0.40910738 0.11588114]
Sharpe Ratio: 1.2681807403498422
```



Our ultimate goal is to predict the price trends of three types of assets based on stacking models, and then allocate weights to these three strategies. In general, we allocate the weights of the three strategies, not simply these three assets.

In our investment strategy update process, we selected the last 20% of our dataset as a new test set to forecast the performance of three ETFs using a stacking model. The results are following:

```
The return rate of DBA is 1.4264207963628712
```

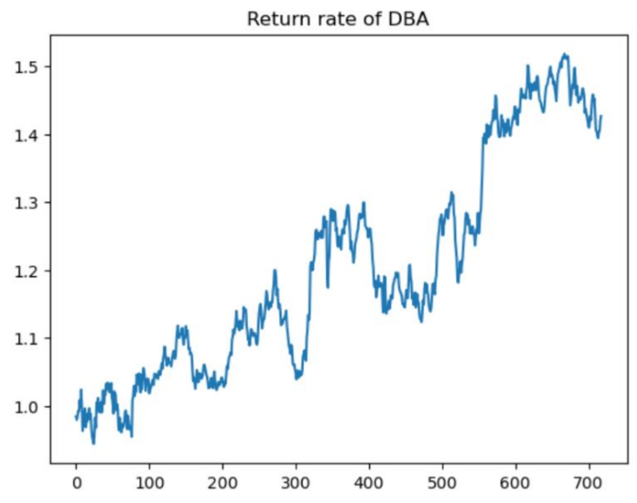
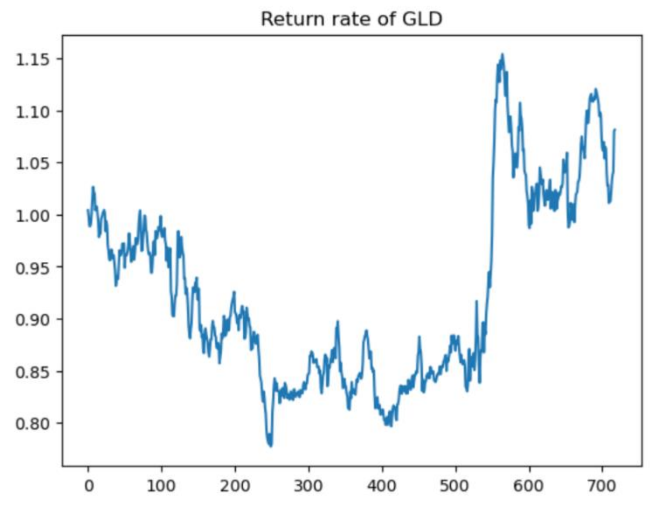


Figure 5(d): *Daily rate of return of DBA from prediction.*

Figure 5(e): *Daily rate of return of GLD from prediction.*

```
The return rate of GLD is 1.0813654406536823
```



```
The return rate of USO is 0.9352527350514022
```

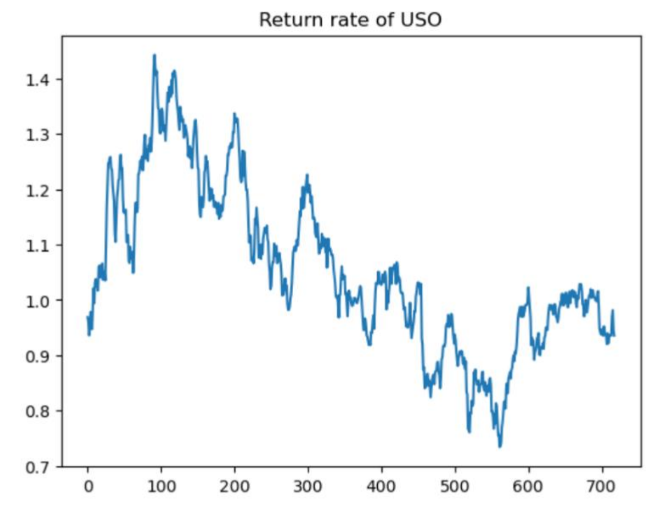


Figure 5(f): *Daily rate of return of USO from prediction.*

```
The final return rate is 1.2144715448914714
The final sharpe ratio is 0.5969486192097078
```

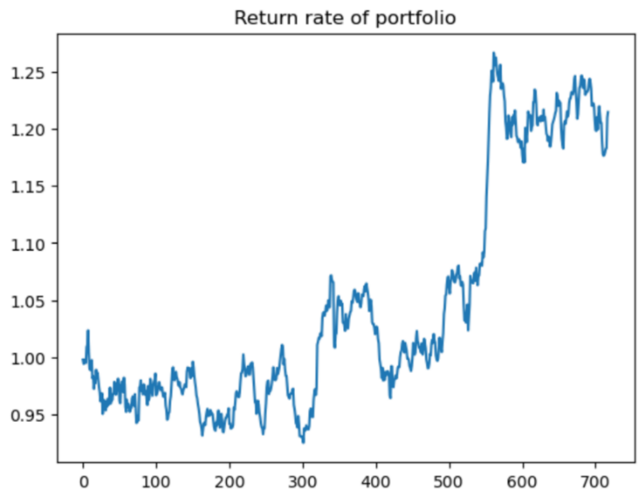


Figure 5(g): *Daily rate of return of constructed portfolio from prediction.*

Using these predictions and the PSO-weighted allocation, our final portfolio achieved a return of 1.214 and a Sharpe ratio of 0.597. This Sharpe ratio indicates a favorable risk-adjusted return, demonstrating that our portfolio has efficiently balanced return and risk per unit of volatility. This outcome not only underscores the practical application of advanced optimization techniques like PSO in asset management but also highlights the effectiveness of stacking models in forecasting financial returns.

The return rate graph shows an overall upward trend, demonstrating that the portfolio optimization strategy was effective in increasing the portfolio's value over time. The significant spike towards the end indicates that the strategy capitalized on favorable market conditions, leading to higher returns.

Part 6: Conclusion and Future Improvements

6.1 Conclusion

In conclusion, the project was aimed at developing and optimising a machine learning-driven trading strategy for a portfolio consisting of ETFs (DBA, GLD, USO). The focus was on maximizing the return rate and Sharpe ratio using various ML models. We found that Gradient Boosting and Ensemble Stacking were the top performers across all techniques. While our stacking model showed slightly higher performance in some cases but at the cost of increased complexity and computational overhead. Additionally, we used our ensemble strategy to calculate that the optimal portfolio weights of 0.475, 0.409 and 0.116.

6.2 Future Improvements

1. This project covers a whole breadth of Machine Learning models but does not focus too much on the portfolio optimization.
   - Future work could include implementing a dynamic rebalancing strategy to adjust the portfolio weights periodically (e.g., weekly, monthly) to account for changing market conditions and ensure optimal performance over time.
   - Experiment with other optimization algorithms such as Genetic Algorithms (GA), Differential Evolution (DE), or Gradient-Based Methods to compare performance and potentially improve optimization results.
   - Implement stress testing and scenario analysis to evaluate portfolio performance under extreme market conditions and ensure robustness against adverse events.
2. The optimization primarily focuses on maximizing the Sharpe ratio, which balances return and volatility. Things like risk management and transactions costs were not taken into account.
   - Integrate transaction cost into the ML optimization process to ensure that the portfolio performance accounts for real-world trading frictions, making the results more practical and applicable.
   - Incorporate additional risk management metrics such as Value at Risk (VaR), maximum drawdown or including a stop loss to provide a more comprehensive risk assessment and management strategy.
3. Portfolio optimization was done using actual day-to-day returns since our ML methods only predicted stock direction.
   - Predicting returns can be further explored to enable a more accurate representation of how our models would perform in the real-world markets.
4. Our project used only certain historical financial data and economic indicators.
   - Since the ETFs are related to commodities and a large volume of transactions are the big players trying to hedge their exposures with available instruments, certain industry-specific metrics can be used.
   - Examples include hedging pressure (The hedging pressure of commodity i is the number of futures contracts that commercial traders are short minus the number of futures contracts that are long, divided by the futures open interest) or Call-to-Put volume ratio.