US Oil Markets Volatility Forecasting to Price Options and Make Market Decisions

Michael Obajemu

May 09, 2025

Abstract

Accurate prediction of volatility is critical to effectively pricing the US oil markets. This study explores how to reliably predict volatility change in the US crude oil market by developing a classification model trained on historical data, multiple volatility estimators, and statistical indicators specific to the West Texas Intermediate (WTI) futures contracts(ticker: CL=F). Among various algorithms tested, XGBoost emerged as the best performing model, achieving a high accuracy of approximately 71 percent, effectively predicting volatility increases and decreases with great generalization capabilities. This report contributes to existing literature by demonstrating that incorporating comprehensive volatility indicators and metrics as features into a boosting algorithm yields a reliable forecasting tool for investors and traders to make informed market decisions.

1 Introduction

Volatility forecasting is an important component of asset risk management and derivatives pricing in todays financial markets. It serves as a useful gauge for assessing how much risk an investor is willing to tolerate and how assets can be combined to diversify a portfolio and maximize returns [1].

The decision problem this paper addresses is the accurate classification of future volatility in the US oil markets, specifically predicting whether the realized volatility of the West Texas Intermediate (WTI) index will increase or decrease in the near term. This issue is critical to investors, asset managers, and other stakeholders involved in the pricing of the financial markets. Since volatility influences portfolio risk tolerance and option pricing models, asset managers can use such models to adjust their exposure to navigate turbulent market conditions or capitalize on arbitrage opportunities.

The dataset used in this study consists of historical data for the West Texas Intermediate futures contract (ticker symbol: CL=F), covering the 2024 calendar year and sourced directly from the Yahoo Finance API. The dataset includes daily open, high, low, close, and volume (OHLCV) values. From these inputs, we compute features such as log returns, the mean and standard deviation of returns, price range volatility estimators, and volume dynamics. One of feature variable, log returns, is constructed by calculating realized volatility over a defined window and labeling each observation based on whether volatility increased or decreased in subsequent periods.

A significant body of literature and industry leaders continue to explore and refine models for forecasting volatility in economic markets. One such model is GARCH (Generalized Auto regressive Conditional Heteroskedasticity), which estimates volatility by capturing clustering behavior and the persistence of volatility following certain market events. GARCH produces a time series of variance forecasts that can be updated using only the prior prediction and residual data. Although powerful, GARCH assumes linearity and places greater weight on recent data, which increases the risk of overfitting, especially in environments where non-linear factors influence volatility [1]. Its sensitivity to outliers also makes it less ideal for the experimental purposes of this study.

Other notable methods for estimating volatility are Parkinsons, Garman-Klass, Rogers-Satchell, and Yang-Zhang.

Parkinsons uses the difference between the days high and low prices to estimate volatility. A key limitation in it is the assumption of continuous trading and it doesn't consider information from the overnight gap in price changes, which can lead to an underestimation of actual volatility [1].

$$\sigma_{\text{Parkinson}} = \sqrt{\frac{1}{4\ln(2)}\,\frac{1}{n}\sum_{i=1}^{n}\bigl[\ln(H_i/L_i)\bigr]^2}\,. \tag{1}$$

The Garman-Klass estimator is a slight variation from the Parkinsons estimators. If the opening price is unavailable, the model uses the prior day's closing price in place of it. Just like Parkinsons, it normally ignores overnight returns and underestimates volatility [1].

$$\sigma_{\text{GK}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\Bigl[\,\tfrac{1}{2}\bigl(\ln(H_i/L_i)\bigr)^2 \;-\; (2\ln(2)-1)\bigl(\ln(C_i/O_i)\bigr)^2\Bigr]}\,. \tag{2}$$

The Rogers-Satchell estimator uses the same OHLC data, assumes a nonzero return and includes a drift in the model, making it more complex. However, like Parkinson's and Garmen-Klass, it still ignores the overnight price change, making it underestimate volatility [1].

$$\sigma_{\text{RS}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\Bigl[\ln\bigl(\tfrac{H_i}{C_i}\bigr)\,\ln\bigl(\tfrac{H_i}{O_i}\bigr) + \ln\bigl(\tfrac{L_i}{C_i}\bigr)\,\ln\bigl(\tfrac{L_i}{O_i}\bigr)\Bigr]}\,. \tag{3}$$

The Yang Zhang estimator tries to address all the prior models limitations by combining overnight volatility with a weighted average of open-to-close and Rogers-Satchell volatility. This results in a more robust estimator than the ones previously mentioned [1].

$$\sigma_{\text{YZ}} = \sqrt{\sigma_o^2 \;+\; k\,\sigma_c^2 \;+\; (1-k)\,\sigma_{\text{RS}}^2}\,. \tag{4}$$

This report aims to develop a more reliable and robust methodology for forecasting volatility, compared to the models mentioned above, by incorporating them all into the predictor of the classification model. In addition, the following predictors are also considered:

1. Realized volatility of log returns over the past window of annualized trading day. Which indicates how bumpy the price has been on an annual basis
2. Rolling mean, an average log returns over the last window of days. It is a short-term volatility indicator
3. Rolling standard deviation, the standard deviation of log returns over the last window
4. Volume change, the day-over-day percentage change in trading volume. Often a indicator or confirmation of price moves

5. The prior days log return and volume to add predictive power in the time series model
6. Five and ten day moving averages of log returns as a short term trend indicator
7. Five and ten day un-annualized volatility
8. The change in log return over five days to track momentum
9. Five day rolling standard deviation of log returns as a short-term volatility measure

For the target variable. Future volatility is calculated from the realized annualized volatility windows. The target is then labeled zero or one; one if future volatility is greater than the current realized volatility and zero if otherwise.

These predictors and target variables will be used to train and develop the predictive model to evaluate how effectively past price action and volume shifts can forecast future increases in volatility.

This report is organized as follows:

1. The Methodology section discusses the data processing, feature engineering, model construction and implementation.
2. The Discussion interprets the results, discusses the findings, and offers suggestions for potential improvement in future implementations
3. The Appendix contains all reference materials and literature

2 Methodology

2.1 Preparing the developer environment

To begin, a Jupyter notebook was prepared running on Python 3.10.9 within an Anaconda environment. This version of Python provides the latest and greatest features, including pre-built data science packages. It was run in an isolated virtual environment to avoid package conflicts.

The following libraries are required and must be installed to proceed:

- **Pandas (version: 1.5.3)** provides the ability to read and manipulate Data Frames.
- **NumPy (version: 1.23.5)** enables numerical computations.
- **Matplotlib**offers data visualization tools, such as scatter plots, sigmoid curves, and confusion matrices
- **scikit-learn (sklearn)** provides a range of tools for building and evaluating models:
  - DecisionTreeClassifier, KNeighborsClassifier, LogisticRegression, GaussianNB
  - LinearDiscriminantAnalysis, SVC and RandomForestClassifier for training
  - StandardScaler to ensure all features contribute equally
  - train_test_split to divide data into training and test sets,
  - learning_curve to generate learning curves
  - cross_val_score to assess performance
  - GridSearchCV to tune each model

Also, from sklearn.metrics we get performance evaluators such as:

- roc_curve (true vs false positive rates)
- auc (area under the ROC curve)
- ConfusionMatrixDisplay (plot true vs predicted labels)
- TimeSeriesSplit for time-series-aware splits.

xgboost and **lightgbm** to build gradient boosted models with XGBClassifier and LGBMClassifier respectively

2.2 Loading the data and preparing the predictors and target

The CL=F dataset is imported into a data frame using pandas. Then all the predictors stated above in the introduction are calculated and stored in associated columns in the data frame.

Because there are so many predictor features, not all will be useful in predicting volatility, in fact some might be extra noise for the algorithms to parse through. So, a Logistic regression model is trained to measure which predictors contribute the greatest to predicting volatility. First, the dataset is trained, tested and split, with 20 percent of the data remaining for testing. Random state was set to 42 to ensure the split reproducibility each time the code is executed. Shuffle is set to false since we want pure rolling splits, and the dataset is time aware. Since logistic regression is sensitive to feature scaling, the training data was standardized with StandardScaler() and fitted on the training data. The logistic regression baseline model Is then trained and fitted.

The coefficients (which weighs the influence of each feature on predicting) of the model features are then outputted and ranked. Positive coefficient means the feature increases the likelihood of the target predicting volatility goes up and inverse for negative. The higher the absolute coefficient the stronger the influence over the model.

The features with the highest coefficients will be considered to retrain the model, and all other features would be dropped.

2.3 Algorithm information

To conduct the classification of volatility in the US oil markets, seven models were considered: Decision tree, random forest classifier, logistic regression, k-nearest neighbors, support vector machines, XGBoost, and LightGBM.

Decision Tree

Decision trees is an algorithm that splits the dataset into subsets based on feature values, creating a tree like structure. It is robust in the presence of outlier data and is a nonlinear model. However, it is prone to overfitting and can be unstable when exposed to noise in the dataset . The algorithm partitions the feature space by selecting splits that maximize information gain [5].

$$\text{Gini}(t) = 1 - \sum_{i=1}^{C} p_i^2. \tag{5}$$

Random Forest

The random forest algorithm is an expansion of the decision tree. It takes different parts of the dataset to train each tree and then combines the results by averaging them. This approach helps improve its accuracy compared to decision tree. It also reduces the overfitting of decision tree at the expense of a higher computation workload [6].

$$\hat{y} = \text{mode}\big\{\, h_b(x) \big\}_{b=1}^{B}. \tag{6}$$

Logistic Regression

Logistic regression is a classification tool to predict the probability that an instance belongs to a given class or not by analyzing the relationship between two data factors. Logistic regression performs well when the dataset is linearly separable, however, its key limitation is its assumption of linearity between the dependent and independent variables, which is rarely the case in real-world datasets. Logistic regression is used for binary classification with the use of the sigmoid function. Which takes the input as independent variables and produces a probability value between 0 and 1 to which class it belongs to [2, 3].

$$p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}. \tag{7}$$

K-Nearest Neighbors (KNN)

The KNN algorithm is used for classification where the input is classified by its nearest neighbors in the feature space, working off the assumption that similar points can be found near one another. In most cases, distance is measured using Euclidean distance by measuring the straight line between each query point, but other distances can be used. The k value in the k-NN algorithm defines how many neighbors will be used for classification. Choosing k is a balancing act, as too small a value can lead to noisy and overfitted data, and too large a value can lead to over-smoothing and underfitting [4].

$$\mathcal{R}_R\big(C_n^{\text{wnn}}\big) - \mathcal{R}_R\big(C^{\text{Bayes}}\big) = \big(B_1\, s_n^2 + B_2\, t_n^2\big)\big[1 + o(1)\big]. \tag{8}$$

Support Vector Machines (SVM)

SVM is a margin based algorithm that finds the hyperplane in an n-dimension space, maximizing the separation between the closet point of different classes, making it ideal for classification. It works well in high dimensional spaces and can handle nonlinear boundaries well. However, It can be memory intensive and slow on large datasets [7]. SVM solves a convex optimization problem to maximize the margin while penalizing misclassifications through slack variables [8].

$$f(x) = \text{sign}\big(w^\top x + b\big). \tag{9}$$

XGBoost

XGBoost or extreme gradient boosting is a gradient boosting algorithm that builds strong learners by sequentially adding decision tree models to correct previous errors. It offers high predictive accuracy and regularization to prevent overfitting [9].

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathcal{F}. \tag{10}$$

LightGBM

LightGBM is another gradient boosting algorithm that uses the decision tree algorithm for classification. It is optimized for speed and memory efficiency and can handle large scale data and categorical features with minimal preprocessing [10].

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i). \tag{11}$$

It is anticipated that the non-linear models will perform better in classifying the data as the data is very nonlinear, but all are considered to perform well especially given their positive attributes.

2.4 Training the algorithms

Since each classification tool have different parameters to optimize for, it is best to approach the experimentation with which parameter maximize classification performance.

To accomplish this, a grid of parameter combinations was created. Firstly, the possible parameters were specified in a grid and then GridSearchCV was used to take every possible parameter combination from the grid and evaluates it to maximize its area under curve score. Since the data is time sensitive (meaning the sequence of the data is important in the classification), a time series split was used to respect this ordering for the cross validation evaluation.

Note: Five folds were specified for the cross validation evaluation. A fold refers to a split subset of the dataset. In k-fold cross-validation, the dataset is divided into k equal groups, where one group serves as the test set while the remaining k-1 groups are used for training. This process repeats until each group has served as a test set once. For example, in 5-fold cross-validation, 4 folds are used for training while 1 is reserved for testing in each iteration.

Once all combinations have been evaluated and the best scoring parameter combinations are determined. GridSearchCV stores the optimal configuration of each model into the best_models dictionary under its algorithms name. Some of the parameters considered were:

- **Decision tree**: Max depth to control the depth of the tree generated; this controls the complexity of the tree, shallow depths mean high bias, deeper depths mean more variance
- **Random forest and the boosting model (XGBoost and LightGBM)**: N-estimators to balance bias and variance

- **Logistic regression and SVM**: Regularization strength C to control the tradeoff between fitting the training data well and keeping the model weights small. The larger the C value the more accurate but complex the model
- **KNN**: The number of neighbors. It was capped at the smallest training fold size to avoid requesting more neighbors than data points available
- **Boosting models**: The learning rate to set the speed of the learning, higher the value the faster the learning but with a greater risk for overshooting optimal value
- **LightGBM**: The number of leaves to sets the cap on the complexity of each tree generated

All seven models are then trained and fitted using their best parameter. Two more models are instantiated and fitted as a baseline to compare against the tuned seven models, naïve bayes (bayes) and linear discriminant analysis (LDA).

Naïve bayes

Naïve bayes is a classifier based on Bayes theorem. It assumes conditional independence between features. It is extremely fast to train and predict, handles high-dimensional data well, and requires comparatively little training data. Its simplicity, low variance, and speed make it an excellent baseline model against which to compare more sophisticated classifiers [11].

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} P(y) \prod_{i=1}^{n} P(x_i \mid y). \tag{12}$$

Linear discriminant analysis

Linear discriminant analysis is a generative linear classifier that uses bayes theory like naïve bayes. LDA works by identifying a linear combination of features that separates two or more classes of objects. LDA does this by projecting data with two or more dimensions into one dimension so that it can be more easily classified. The algorithm estimates class priors, class-specific means, and a shared covariance matrix, then computes a linear discriminant score for each class and assigns each sample to the class with the highest score [12].

$$\delta_k(x) = x^\top \Sigma^{-1} \mu_k - \tfrac{1}{2} \mu_k^\top \Sigma^{-1} \mu_k + \log \pi_k. \tag{13}$$

2.5 Evaluating the algorithms

After each model is trained, it needs to be scored and evaluated to assess its performance for the classification task. Three functions were created to evaluate each model.

- **getScore**: Plots a confusion matrix and build a classification report on each model
- **plotAuc**: Plots all models on an ROC AUC to compare their performance to one another
- **getFit**: Score the models on a fitting graph and output its cross validation per fold, mean, and standard deviation scores

Confusion matrices and classification metrics offer detailed insights into each model's performance. The confusion matrix visually demonstrates true positives, true negatives, false positives, and false negatives, aiding easy identification of misclassifications. Classification metrics such as accuracy, precision, recall, and F1-score quantitatively evaluate model performance:

- **Accuracy**: Indicates overall correctness
- **Precision**: Measures the accuracy of positive predictions, reflecting a models reliability
- **Recall**: Reflects the model's sensitivity in identifying all actual instances of a class
- **F1-score**: Provides a balanced measure between recall and precision, especially useful in scenarios with imbalanced datasets

The **ROC AUC plot** curve is created by plotting the true positive rate against the false positive rate at different threshold levels. The shape of the ROC curve provides insight into the performance of each model, and the area under this curve (AUC) summarizes this performance.

The **AUC value** is interpreted as the probability that a randomly chosen positive instance is ranked higher than a randomly chosen negative instance, which is a useful metric for understanding model performance. An AUC of 0.5 suggests that the model has no classification power, while an AUC of 1.0 indicates perfect classification. Values between these extremes indicate varying levels of performance, with higher values representing better model performance [13].

A **fitting graph** visualizes the performance of the models cross validation score. It compares the training data to the holdout data (testing data) across each fold. If training and holdout scores are close and high, the model generalizes well. If they are both low, the model under-fits. If training accuracy is high but holdout is low, the model overfits. The mean and the standard deviation across scores are calculated as well. High variance in fold score means the model performance is unstable. Consistently high scores signify a reliable model. A low mean score indicates a poor performing model.

For selection of which model to utilize, only the area under curve and accuracy score will be considered as they are the best separators comparing the models.

2.6 Using chosen algorithm to make market decision

Once a model has been selected. The model will then be tested on new data of the crude oil futures ticker from January through April 2025. And the full capabilities of the functions getScore, and getFit will be utilized to model its performance and make a market decision.

3 Results

3.1 Ranking coefficients

From the executed code to rank the coefficients: the features with the greatest predictive coefficients were Rolling standard deviation, Yang Zhang volatility estimator, five and ten moving averages of log returns, and five day rolling standard deviation of log returns.

Yang-Zhang volatility estimator being one of the highest contributors is in line with expectations as its key feature is addressing the shortcoming of the Parkinson, Garman-Klass, and Roger-Satchell estimator of not having overnight volatility as part of its computation

Feature	Coefficient	Absolute Coefficient
std_10	0.818186	0.818186
rolling_std	-0.814477	0.814477
std_5	-0.814477	0.814477
volatility_5	-0.814477	0.814477
yang_zhang	0.788453	0.788453
ma_10	0.293861	0.293861
log_return	0.196096	0.196096
momentum_5	0.182764	0.182764
garman_klass	0.171577	0.171577
parkinson_vol	-0.152569	0.152569
volume_lag1	0.150323	0.150323
ma_5	-0.106306	0.106306
rolling_mean	-0.106306	0.106306
rogers_satchell	-0.068182	0.068182
volume_change	0.053894	0.053894
return_lag1	0.039189	0.039189

Table 1: Feature coefficients and their absolute values

volatility_5, rolling_std, and std_5 are the most important features, suggesting that short-term volatility is most critical for predicting future volatility.

Features like log_return and return_lag1 have smaller coefficients, indicating they have less influence on the model's predictions.

3.2 Best parameters for the seven primary models

From the executed code to choose the best parameter for each model is as follows:

Model	Best Parameters
Decision Tree	{max_depth: 3}
Random Forest	{n_estimators: 200}
Logistic Regression	{C: 1}
KNN	{n_neighbors: 10}
SVM	{C: 1, gamma: 0.01}
XGBoost	{learning_rate: 0.1, max_depth: 3, n_estimators: 50}
LightGBM	{learning_rate: 0.01, n_estimators: 500, num_leaves: 31}

Table 2: Best parameters for each model

The selected parameters from the grid search suggests that the default setting for models performed best on the given dataset. For the Decision Tree, a shallow depth of 3 was best, indicating that a small number of splits was sufficient and deeper trees likely led to overfitting. The Random Forest performed best with the highest number of estimators tested (200), more trees typically reduce overfitting. Logistic Regression model performed best with default regularization (C=1). The K-Nearest Neighbors chose 10 neighbors, showing that mores neighbors helped reduce sensitivity to noise. For the Support Vector Machine, the default regularization parameter (C=1) combined with a low gamma (0.01) resulted in a smoother decision boundary. The dataset is likely reasonably linear or have simple non-linear separation. XGBoost performed best with a small tree depth (3), a moderate learning rate (0.1), and fewer boosting rounds (50). LightGBM selected a very slow learning rate (0.01), more boosting rounds (500), and a smaller number of leaves (31), meaning that incremental learning with simple structures generalized better. Overall, the results point to a dataset with enough data to classify well.

Below is the resultant of the ROC AUC plot:

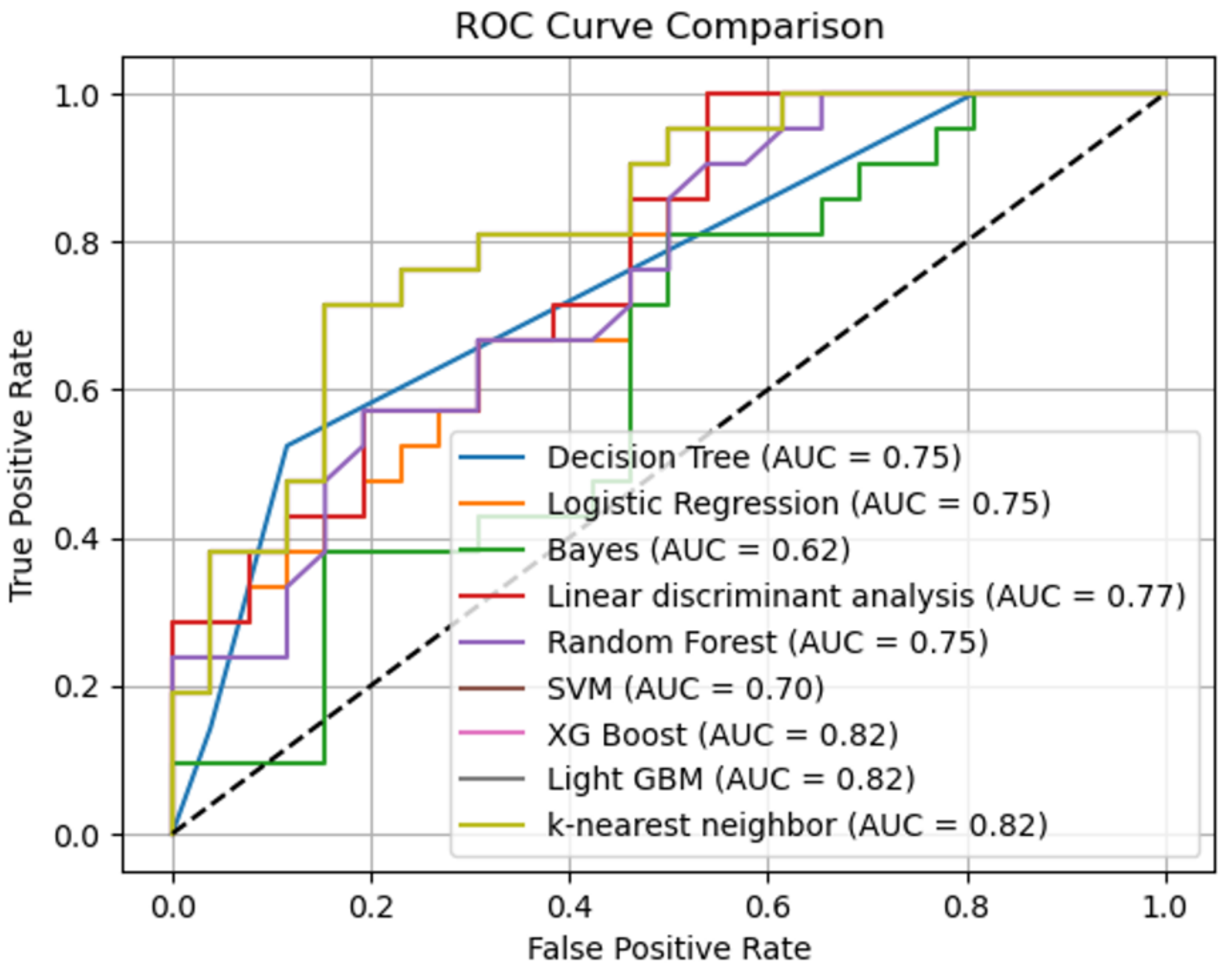


Figure 1: ROC AUC plot

3.3 ROC AUC results

The AUC scores reflect the models ability to distinguish between classes, with 1.0 being perfect and 0.5 being no better than random guessing. From the plot we can infer that: The best performing models are KNN (AUC: 0.822), XGBoost (0.82), and LightGBM (0.82). These scores indicate suggest these models are capturing important patterns in the data. Next up is LDA (0.769), logistic regression (0.755) and the decision tree (0.748). They also performed reasonably well, indicating that simpler models can classify well. Random Forest (0.746) is the worst performer of the tree based models which is surprising as it is typically robust and meant to outperform the likes of Decision Tree. Support Vector Machine (0.70) shows moderate performance. Lastly, Naive Bayes (0.625) lags significantly behind, which is expected as it is the simplest of all the models, if a model weren't able to trump the Bayes score it suggest an extremely poorly tuned model. Overall, KNN and the boosting models performed the classification the best.

3.4 Classification score summaries

To then choose from three models, it is best to consider their scores to find which performed the best. The scores of the eliminated models will not be discussed in this section but are available in the appendices below [15]. The scoring metric used for the cross validation score is balanced accuracy. Balanced accuracy is used for imbalanced datasets to account for class distributions [14].

Model	Accuracy	Macro Precision	Macro Recall
KNN	0.55	0.28	0.50
LightGBM	0.70	0.70	0.70
XGBoost	0.74	0.74	0.74

Table 3: Summary of model performance scores

Model	Mean Balanced Accuracy	Std Dev	Cross-Validation Scores
KNN	0.6505	0.0246	0.6184, 0.6711, 0.6228, 0.6725, 0.6681
LightGBM	0.6768	0.0536	0.6199, 0.7529, 0.7295, 0.6433, 0.6389
XGBoost	0.7123	0.1033	0.5424, 0.7529, 0.7032, 0.7003, 0.8626

Table 4: Cross-validation scores with mean balanced accuracy and standard deviation

KNN significantly underperformed, especially with a poor accuracy score, despite its decent AUC score. In addition, though it has a low standard deviation across cross validation scores, it consistently underperformed across each of its folds. LightGBM provided a stronger balances between precision and recall, indicating better generalization and more reliable classification for both classes. LightGBM has a slightly better mean balanced accuracy than KNN and is stable making it a safer model choice. XGBoost performed the best overall, with the highest precision, recall, and accuracy. It effectively handles class differences better and appears the leading model to use to perform the classification. XGBoost has the highest mean balanced accuracy, indicating it correctly identifies both classes. Though it has a higher standard deviation, it churned out better accuracy. The cross validation results showed a mean balanced accuracy of 0.71, with some variability across folds, suggesting that while performance can fluctuate slightly depending on the data split, the model remains generally strong.

The fitting graph reinforces this conclusion: although there is a clear gap between the high training accuracy and the lower holdout accuracy there's an upward trend in holdout performance as sample size increases. This suggests the model's generalization improves with more data. Overall, the test outcomes confirms that XGBoost is a solid and dependable choice for forecasting volatility.

3.5 Using model on new data to make a prediction

Once XGBoost was solidified as the model of choice it was then tested on the data from January through April 2025 to see how well it can predict volatility, and the results are as following.

The confusion matrix shows that the model performs well in separating between volatility down and volatility up. Out of 66 total predictions, it correctly classified 27 instances of volatility down and 18 instances of volatility up. There were 10 cases where volatility down was misclassified as up and 11 where volatility up was incorrectly predicted as down.

The model demonstrates slightly better precision and recall when predicting volatility down compared to volatility up. However, the number of misclassifications is fairly even between both classes. On the January to April 2025 new data, it achieved an average accuracy, precision, and recall of 0.68, demonstrating some reliability in generalization.

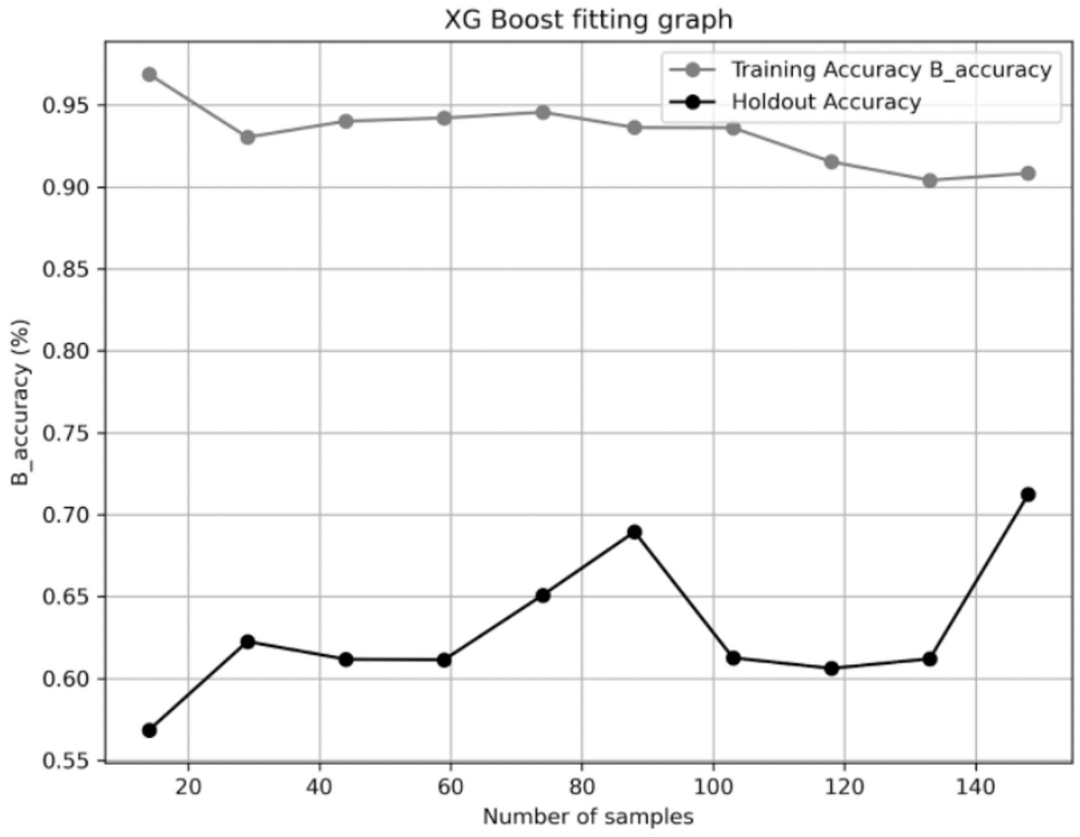


Figure 2: XGBoost Fitting graph, balanced accuracy scoring

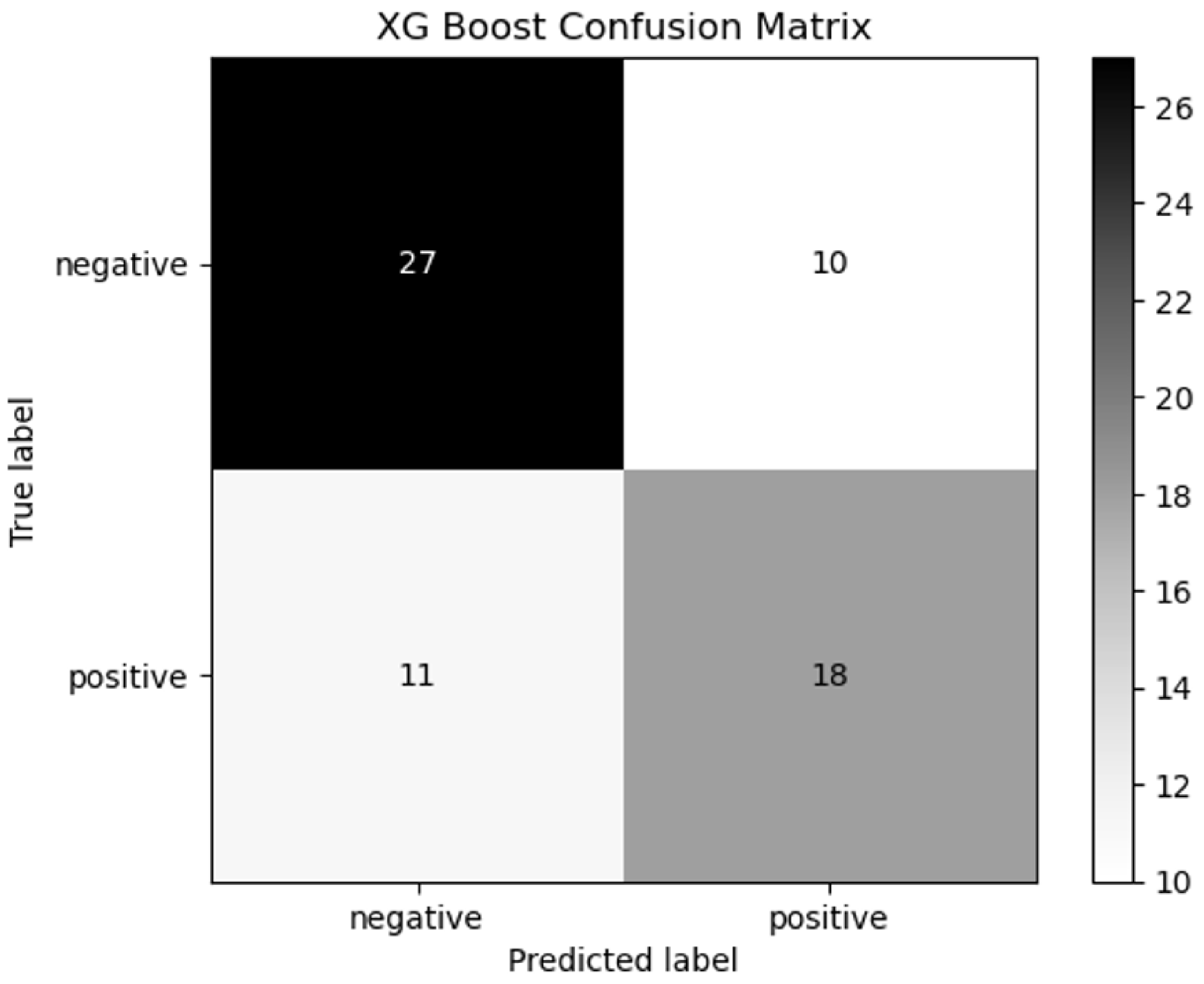


Figure 3: XG confusion matrix

Class	Precision	Recall	F1-score
Volatility down	0.71	0.73	0.72
Volatility up	0.64	0.62	0.63
Overall Accuracy		0.68	
Macro Precision		0.68	
Macro Recall		0.68	

Table 5: Classification report: XG Boost classfication report

In summary, the confusion matrix and classification report reflect a balanced and generally reliable performance in predicting volatility shifts.

4 Discussion

The dominance of short-term volatility features, specifically the 5-day and 10-day rolling standard deviations and the Yang-Zhang estimator as top predictors, confirms that capturing both intraday price swings and overnight gaps is critical for forecasting near-term volatility in WTI futures. This aligns with the expectation that combining multiple volatility estimators into a machine learning framework would outperform any single estimator alone. In addition, the superior performance of nonlinear boosting models (XGBoost and LightGBM) validates the hypothesis that flexible learners better capture complex market dynamics than linear models such as logistic regression or the GARCH-style approaches. Accurately forecasting volatility in oil markets holds critical value for many financial and industry stakeholders. For investors and asset managers, a volatility predictor enables dynamic adjustment of hedge ratios and improved risk-adjusted returns when timing options trades. For derivative traders, this offers a data-driven approach to options pricing. Beyond the financial market, volatility forecasting using a data-driven model can help oil suppliers identify price-arbitrage opportunities between the US market and other regions such as OPEC. In sum, this work bridges the gap between traditional volatility estimators and modern machine-learning techniques.

To demonstrate the practical deployment of the machine-learning algorithm, consider the following scenario. It is December 2024, and a small hedge fund wants to adjust its options exposure to West Texas Intermediate (WTI) by monitoring the corresponding futures contract. The fund plans to make daily trades with a maximum budget of $100,000, scaled according to the confidence of the volatility prediction. Specifically, the fund will go long (+1) or short (−1) if the predicted probability of a volatility increase exceeds a certain threshold (see equation below).

$$\text{signal}_t = \begin{cases} +1, & \text{if } P(\uparrow) \geq \tau_0, \\ -1, & \text{if } P(\downarrow) \geq \tau_1. \end{cases} \tag{14}$$

The institution has also decided to scale its order size in proportion to the confidence level of the volatility signal, using the Kelly-type sizing (see equations below). With these parameters, the hedge fund can determine whether to buy, sell, or hold its position each day. Please see the appendix for the output of these simulated trades [16].

$$f_t \propto \frac{P(\uparrow) - \big(1 - P(\uparrow)\big)}{\text{odds}}. \tag{15}$$

While the XGBoost-based model performed well given the features and parameters on which it was trained, there are several avenues that could further enhance forecasting accuracy and scope. Additional features such as US crude-oil inventory levels, OPEC production cuts, and oil-market sentiment metrics could be incorporated to improve the model's robustness. The analysis could also be extended to high-frequency data (intra-day and hourly price movements) to capture volatility throughout the trading day. By pursuing these improvements in future work, the model could become a more robust, timely, and actionable volatility estimator.

References

[1] Salt Financial. *The Layman's Guide to Volatility Forecasting.* New York: Salt Financial, May 2021. Available: `https://saltfinancial.com/static/uploads/2021/05/The%20Laymans%20Guide%20to%20Volatility%20Forecasting.pdf`.

[2] Akshay Jain. "Advantages and Disadvantages of Logistic Regression in Machine Learning." *Medium*, July 22, 2020. Available: `https://medium.com/@akshayjain_757396/advantages-and-disadvantages-of-logistic-regression-in-machine-learning-a6a247e42b20`. [Accessed: March 25, 2025].

[3] GeeksforGeeks. "Understanding Logistic Regression." Available: `https://www.geeksforgeeks.org/understanding-logistic-regression/`. [Accessed: March 25, 2025].

[4] IBM. "What Is the k-Nearest Neighbors (KNN) Algorithm?" *IBM Think Blog.* Available: `https://www.ibm.com/think/topics/knn`. [Accessed: March 25, 2025].

[5] Inside Learning Machines. "Advantages and Disadvantages of Decision Trees." Available: `https://insidelearningmachines.com/advantages_and_disadvantages_of_decision_trees/`. [Accessed: March 25, 2025].

[6] GeeksforGeeks. "Random Forest Algorithm in Machine Learning." Last updated January 16, 2025. Available: `https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/`. [Accessed: March 25, 2025].

[7] GeeksforGeeks. "Support Vector Machine (SVM) Algorithm." Last updated January 27, 2025. Available: `https://www.geeksforgeeks.org/support-vector-machine-algorithm/`. [Accessed: March 25, 2025].

[8] Analytics Vidhya. "The Mathematics Behind SVM." October 2020. Available: `https://www.analyticsvidhya.com/blog/2020/10/the-mathematics-behind-svm/`. [Accessed: March 25, 2025].

[9] NVIDIA. "XGBoost – What Is It and Why Does It Matter?" July 28, 2023. Available: `https://www.nvidia.com/en-us/glossary/xgboost/`. [Accessed: March 25, 2025].

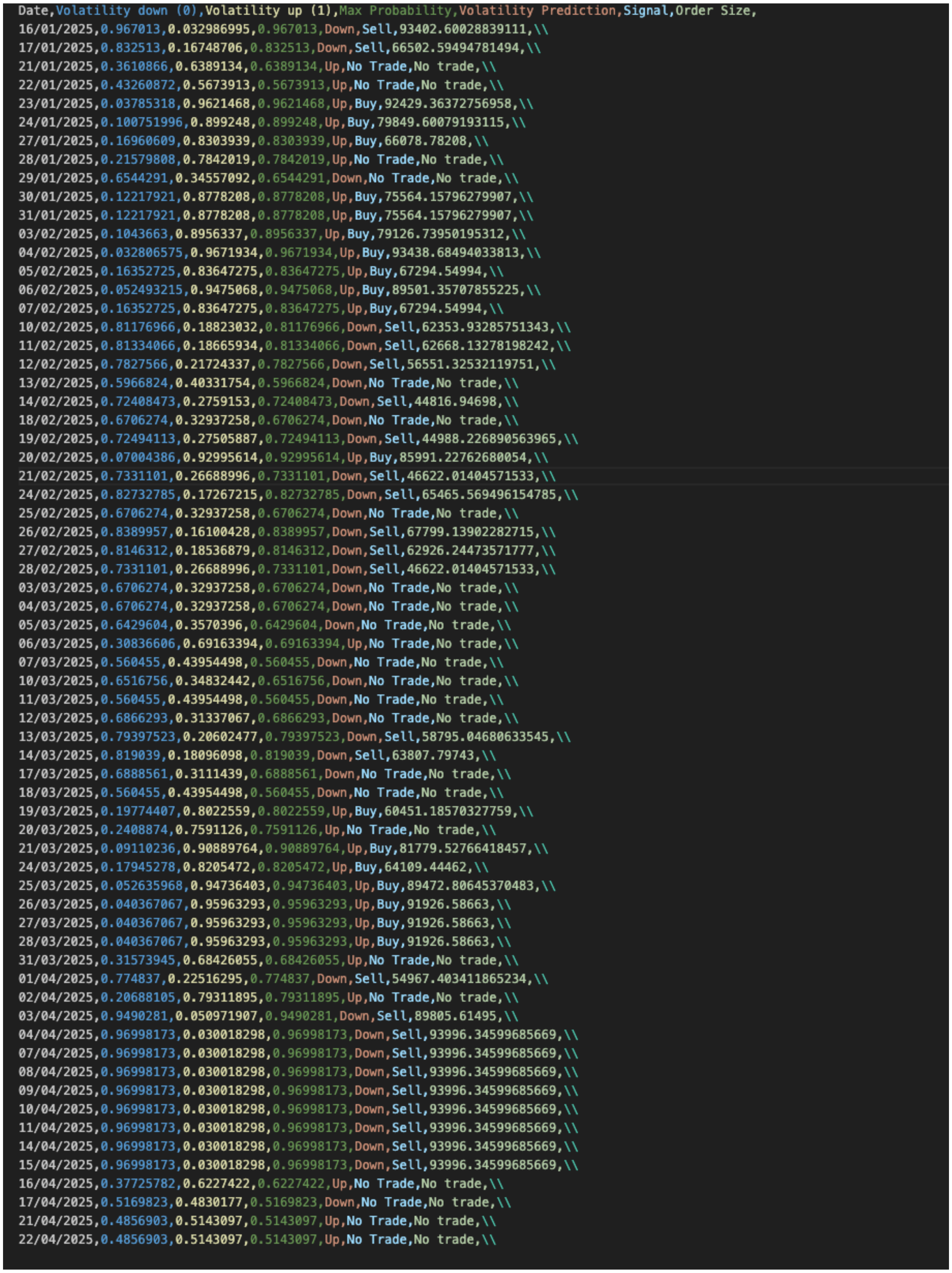
[10] Data Science Team. "What Is LightGBM?" December 28, 2019. Available: `https://datascience.eu/machine-learning/1-what-is-light-gbm/`. [Accessed: March 25, 2025].

[11] Selva Prabhakaran. "How Naive Bayes Algorithm Works?" *Machine Learning Plus*, December 3, 2017. Available: `https://www.machinelearningplus.com/predictive-modeling/how-naive-bayes-algorithm-works-with-example-and-full-code/`. [Accessed: March 25, 2025].

[12] IBM. "What Is Linear Discriminant Analysis (LDA)?" *IBM Think*, November 27, 2023. Available: `https://www.ibm.com/think/topics/linear-discriminant-analysis`. [Accessed: March 25, 2025].

[13] Learn Statistics Easily. "What Is: Area Under Curve – A Comprehensive Guide." 2025. Available: `https://statisticseasily.com/glossario/what-is-area-under-curve-a-comprehensive-guide/`. [Accessed: March 25, 2025].

[14] Neptune.ai. "Balanced Accuracy." October 22, 2024. Available: `https://neptune.ai/blog/balanced-accuracy`. [Accessed: March 25, 2025].



```
Date,Volatility down (0),Volatility up (1),Max Probability,Volatility Prediction,Signal,Order Size,
16/01/2025,0.967013,0.032986995,0.967013,Down,Sell,93402.60028839111,\\
17/01/2025,0.832513,0.16748706,0.832513,Down,Sell,66502.59494781494,\\
21/01/2025,0.3610866,0.6389134,0.6389134,Up,No Trade,No trade,\\
22/01/2025,0.43260872,0.5673913,0.5673913,Up,No Trade,No trade,\\
23/01/2025,0.03785318,0.9621468,0.9621468,Up,Buy,92429.36372756958,\\
24/01/2025,0.100751996,0.899248,0.899248,Up,Buy,79849.60079193115,\\
27/01/2025,0.16960609,0.8303939,0.8303939,Up,Buy,66078.78208,\\
28/01/2025,0.21579808,0.7842019,0.7842019,Up,No Trade,No trade,\\
29/01/2025,0.6544291,0.34557092,0.6544291,Down,No Trade,No trade,\\
30/01/2025,0.12217921,0.8778208,0.8778208,Up,Buy,75564.15796279907,\\
31/01/2025,0.12217921,0.8778208,0.8778208,Up,Buy,75564.15796279907,\\
03/02/2025,0.1043663,0.8956337,0.8956337,Up,Buy,79126.73950195312,\\
04/02/2025,0.032806575,0.9671934,0.9671934,Up,Buy,93438.68494033813,\\
05/02/2025,0.16352725,0.83647275,0.83647275,Up,Buy,67294.54994,\\
06/02/2025,0.052493215,0.9475068,0.9475068,Up,Buy,89501.35707855225,\\
07/02/2025,0.16352725,0.83647275,0.83647275,Up,Buy,67294.54994,\\
10/02/2025,0.81176966,0.18823032,0.81176966,Down,Sell,62353.93285751343,\\
11/02/2025,0.81334066,0.18665934,0.81334066,Down,Sell,62668.13278198242,\\
12/02/2025,0.7827566,0.21724337,0.7827566,Down,Sell,56551.32532119751,\\
13/02/2025,0.5966824,0.40331754,0.5966824,Down,No Trade,No trade,\\
14/02/2025,0.72408473,0.2759153,0.72408473,Down,Sell,44816.94698,\\
18/02/2025,0.6706274,0.32937258,0.6706274,Down,No Trade,No trade,\\
19/02/2025,0.72494113,0.27505887,0.72494113,Down,Sell,44988.226890563965,\\
20/02/2025,0.07004386,0.92995614,0.92995614,Up,Buy,85991.22762680054,\\
21/02/2025,0.7331101,0.26688996,0.7331101,Down,Sell,46622.01404571533,\\
24/02/2025,0.82732785,0.17267215,0.82732785,Down,Sell,65465.569496154785,\\
25/02/2025,0.6706274,0.32937258,0.6706274,Down,No Trade,No trade,\\
26/02/2025,0.8389957,0.16100428,0.8389957,Down,Sell,67799.13902282715,\\
27/02/2025,0.8146312,0.18536879,0.8146312,Down,Sell,62926.24473571777,\\
28/02/2025,0.7331101,0.26688996,0.7331101,Down,Sell,46622.01404571533,\\
03/03/2025,0.6706274,0.32937258,0.6706274,Down,No Trade,No trade,\\
04/03/2025,0.6706274,0.32937258,0.6706274,Down,No Trade,No trade,\\
05/03/2025,0.6429604,0.3570396,0.6429604,Down,No Trade,No trade,\\
06/03/2025,0.30836606,0.69163394,0.69163394,Up,No Trade,No trade,\\
07/03/2025,0.560455,0.43954498,0.560455,Down,No Trade,No trade,\\
10/03/2025,0.6516756,0.34832442,0.6516756,Down,No Trade,No trade,\\
11/03/2025,0.560455,0.43954498,0.560455,Down,No Trade,No trade,\\
12/03/2025,0.6866293,0.31337067,0.6866293,Down,No Trade,No trade,\\
13/03/2025,0.79397523,0.20602477,0.79397523,Down,Sell,58795.04680633545,\\
14/03/2025,0.819039,0.18096098,0.819039,Down,Sell,63807.79743,\\
17/03/2025,0.6888561,0.3111439,0.6888561,Down,No Trade,No trade,\\
18/03/2025,0.560455,0.43954498,0.560455,Down,No Trade,No trade,\\
19/03/2025,0.19774407,0.8022559,0.8022559,Up,Buy,60451.18570327759,\\
20/03/2025,0.2408874,0.7591126,0.7591126,Up,No Trade,No trade,\\
21/03/2025,0.09110236,0.90889764,0.90889764,Up,Buy,81779.52766418457,\\
24/03/2025,0.17945278,0.8205472,0.8205472,Up,Buy,64109.44462,\\
25/03/2025,0.052635968,0.94736403,0.94736403,Up,Buy,89472.80645370483,\\
26/03/2025,0.040367067,0.95963293,0.95963293,Up,Buy,91926.58663,\\
27/03/2025,0.040367067,0.95963293,0.95963293,Up,Buy,91926.58663,\\
28/03/2025,0.040367067,0.95963293,0.95963293,Up,Buy,91926.58663,\\
31/03/2025,0.31573945,0.68426055,0.68426055,Up,No Trade,No trade,\\
01/04/2025,0.774837,0.22516295,0.774837,Down,Sell,54967.403411865234,\\
02/04/2025,0.20688105,0.79311895,0.79311895,Up,No Trade,No trade,\\
03/04/2025,0.9490281,0.050971907,0.9490281,Down,Sell,89805.61495,\\
04/04/2025,0.96998173,0.030018298,0.96998173,Down,Sell,93996.34599685669,\\
07/04/2025,0.96998173,0.030018298,0.96998173,Down,Sell,93996.34599685669,\\
08/04/2025,0.96998173,0.030018298,0.96998173,Down,Sell,93996.34599685669,\\
09/04/2025,0.96998173,0.030018298,0.96998173,Down,Sell,93996.34599685669,\\
10/04/2025,0.96998173,0.030018298,0.96998173,Down,Sell,93996.34599685669,\\
11/04/2025,0.96998173,0.030018298,0.96998173,Down,Sell,93996.34599685669,\\
14/04/2025,0.96998173,0.030018298,0.96998173,Down,Sell,93996.34599685669,\\
15/04/2025,0.96998173,0.030018298,0.96998173,Down,Sell,93996.34599685669,\\
16/04/2025,0.37725782,0.6227422,0.6227422,Up,No Trade,No trade,\\
17/04/2025,0.5169823,0.4830177,0.5169823,Down,No Trade,No trade,\\
21/04/2025,0.4856903,0.5143097,0.5143097,Up,No Trade,No trade,\\
22/04/2025,0.4856903,0.5143097,0.5143097,Up,No Trade,No trade,\\
```

Figure 4: Simulated trades table