Project II Report

1.Introduction

The report details a sophisticated project designed to forecast stock market returns using advanced machine learning techniques. It emphasizes the key steps required for effective data analysis, the application of various machine learning models, and the utilization of prominent tools such as BackTrader for rigorously back-testing trading strategies. A significant portion of the study is devoted to examining different trading models, assessing their strengths and weaknesses within the financial market landscape. This comprehensive approach aims to provide insights into the potential of machine learning in enhancing stock market predictions.

2.Algorithm Overview

The algorithm developed in "Zhijiang_Chen_Project2.py" is constructed around several key components, each serving a distinct purpose in the overall machine learning and data analysis framework:

- DataPreparation: A crucial module that deals with the acquisition and preprocessing of financial market data. It leverages libraries like pandas and yfinance to fetch and prepare data for subsequent analysis. This component is vital for ensuring data quality and readiness for modeling.

- FeatureEngineering: This segment is dedicated to transforming raw financial data into informative features suitable for machine learning models. It involves techniques like scaling and normalization, crucial for enhancing model performance and accuracy in predicting stock market trends.

- ModelBuilding: A core part of the project, focusing on constructing and training various machine learning models. It employs a range of algorithms from libraries such as scikit-learn, XGBoost, and RandomForestClassifier, highlighting the project's commitment to exploring and evaluating diverse modeling approaches.

- ModelEvaluation: This component is tasked with assessing the performance of the machine learning models. It utilizes metrics like accuracy and precision scores, providing a quantitative basis for model comparison and selection.

- StrategyBackTesting: Integrating BackTrader, QuantStats, this module facilitates the backtesting of trading strategies informed by machine learning predictions. It's a critical phase for validating the effectiveness of the models in a simulated trading environment.

Overall, the algorithm is a comprehensive system designed to harness machine learning for insightful financial market analysis, encompassing everything from data handling to strategy evaluation.

3. Implementation Details

(1) Data Manipulation

The Python script implements data manipulation using pandas and yfinance. Historical data for specific stock tickers are fetched from Yahoo Finance within user-defined start and end dates. To maintain data continuity and integrity, forward filling is employed to address missing values, and a standard deviation method is used for outlier detection. Each column (except the last one) is scrutinized for outliers, defined as data points lying beyond three standard deviations from the mean. Finally, rows containing NaN values are removed, ensuring the dataset's quality.

(2) Features Generation

This project emphasizes creating meaningful features from raw data. It includes:

- Price Change Features: Calculating the difference between opening and closing prices ('open-close') and the range between the high and low prices ('high-low').
- Moving Averages and Ratios: Computing Exponential Moving Average (EMA) and Simple Moving Average (SMA) ratios for intervals like 5, 30, 60, and 252 days.
- Volume Features: Including normalized volume ('norm_vol') and the percentage change in volume from the previous day ('vol_change').
- Technical Indicators: Incorporating the Relative Strength Index (RSI) and Moving Average Convergence Divergence (MACD) to capture momentum and trend-following aspects.

(3) Model selection

The use of XGBoost and Random Forest models for predicting stock price direction—up or down—stems from their robust performance in classification problems, particularly in complex domains like financial markets. These markets are characterized by volatile and non-linear patterns, which require models that can capture intricate dependencies and subtle signals amidst market noise and random fluctuations.

XGBoost: is a powerful machine learning algorithm that has gained popularity in the field due to its speed and performance. It is an ensemble method that builds multiple decision trees and combines them to improve accuracy and control overfitting. The use of gradient boosting techniques allows for the optimization of arbitrary differentiable loss functions, making it suitable for the binary classification problem of predicting stock price movements. The hyperparameters for the XGBoost model were chosen to build a robust model that is sensitive enough to detect subtle patterns but resistant to overfitting:

- `max_depth`: Controls the depth of the trees and helps in capturing interactions among features.
- `n_estimators`: The number of trees in the ensemble, which can improve the model's accuracy.
- `learning_rate`: Determines the step size at each iteration while moving toward a minimum of a loss function, preventing overfitting by making the model training more conservative.
- `gamma`: A regularization parameter that encourages simpler models, to control overfitting.
- `subsample` and `colsample_bytree`: These ensure that each tree only considers a random subset of the features and the data, adding to the model's robustness.
xgb_model = xgb.XGBClassifier(objective='binary:logistic', eval_metric='logloss', max_depth=6, # Adjusted max depth n_estimators=200, # Increased number of trees learning_rate=0.05, # Lower learning rate gamma=0.1, # Adjusted gamma subsample=0.8, # Adjusted subsample ratio colsample_bytree=0.8 # Adjusted subsample ratio of columns)

Random Forest is an ensemble of decision trees, typically used for its ability to run efficiently on large databases and its effectiveness in handling thousands of input variables without variable deletion. It is less likely to overfit than a single decision tree and is very easy to measure the relative importance of each feature on the prediction. Hyperparameters for the Random Forest model were chosen to ensure diversity among the trees and to provide a balance between the model's bias and variance:

- `n_estimators`: More trees can lead to a more accurate model, but also to longer computation time.
- `max_depth`: Prevents the trees from becoming too deep, which can lead to overfitting on the training data.
- `min_samples_split` and `min_samples_leaf`: Define the conditions for further splitting the trees, ensuring that the trees do not become overly complex and can generalize well.
rf_model = RandomForestClassifier(n_estimators=150, # Increased number of trees max_depth=10, # Maximum depth of each tree min_samples_split=4, # Minimum number of samples required to split an internal node min_samples_leaf=2, # Minimum number of samples required to be at a leaf node random_state=42 # Random state for reproducibility)

The starting expectation was that these models would effectively discriminate between upward and downward movements in stock prices by learning from historical data. Given the chaotic nature of the financial markets, the models were not expected to

perfectly predict every movement but to capture significant trends and patterns that could be leveraged for a trading advantage. The chosen hyperparameters were expected to contribute to a model complex enough to understand financial data but also generalize well to new, unseen data.

(4) Trading Strategies
The script introduces a custom Strategy class for BackTrader, based on machine learning predictions. This strategy involves buying stocks at the opening price if an upward trend is predicted and selling if a downward trend is forecasted. It also includes detailed logging for various order statuses.

(5) Backtesting Framework
Backtesting is executed using BackTrader. The `run_backtest()` function takes a model and ticker, processes the data, and integrates it into the BackTrader environment. Key performance indicators like Sharpe Ratio and DrawDown are extracted, and trading reports are generated using QuantStats for a comprehensive strategy evaluation.

(6) Trading report
Trading report of 10 best performer are in the folder “top10_trading_report”. All of the reports are in HTML format.
The 10 best performers and their metircs are:

top_10_combined_accuracy

Ticker	Combined_Accuracy	RF_Accuracy	XGB_Accuracy
LKOR	0.7777777777777780	0.7777777777777780	0.7777777777777780
DTYL	0.7727272727272730	0.7272727272727270	0.8181818181818180
AMTD	0.7170542635658910	0.7209302325581400	0.7131782945736440
TAPR	0.6779661016949150	0.6779661016949150	0.6779661016949150
RDIB	0.6685082872928180	0.6850828729281770	0.6519337016574590
APDN	0.6679184549356220	0.6759656652360520	0.6598712446351930
CNFR	0.65625	0.625	0.6875
FMHI	0.652061855670103	0.6701030927835050	0.634020618556701
BKYI	0.6514550264550270	0.6587301587301590	0.6441798941798940
HMNY	0.6485294117647060	0.6705882352941180	0.6264705882352940

4. Model Evaluation and Conclusion

XGBoost Model Accuracy	0.5037614331330844
XGBoost Model Precision	0.5198220064724919

Random Forest Model Accuracy	0.5111760567191643
Random Forest Model Precision	0.5147762013012464

In evaluating the performance of the XGBoost and Random Forest models, we see a nuanced distinction in their effectiveness. The XGBoost model achieved a precision of approximately 0.5198, indicating its strength in correctly predicting positive stock movements, which is crucial for minimizing false positives in trading strategies. On the other hand, the Random Forest model's accuracy of about 0.5112 points to its slightly superior ability to classify both positive and negative instances correctly.

Advantages of XGBoost include its efficient handling of sparse data and its gradient boosting framework, which excels in situations where the relationship between the data points is complex and non-linear. However, a potential disadvantage of XGBoost is its propensity to overfit, especially when the data is not abundant or sufficiently diverse, despite hyperparameter tuning aimed at regularization.

Random Forest's advantage lies in its inherent nature of reducing overfitting through its ensemble approach, making it robust to noise and capable of generalizing well to new data. However, its performance can be limited if the individual decision trees are not well-tuned, and the model can become computationally expensive as the number of trees increases.

Both models outperformed the established benchmarks, which suggests that they are effective tools for predicting stock market trends. The slight advantage in precision for XGBoost may be beneficial in scenarios where the cost of false positives is high, whereas the advantage in accuracy for the Random Forest model could be more desirable in scenarios where the overall rate of correct predictions is paramount.

These findings underscore the importance of choosing a model based on the specific needs and constraints of the application, with an understanding that each model's strengths can be leveraged to suit particular aspects of financial data analysis.

5. Additional consideration

a. The models achieved modest accuracies of approximately 0.5038 for XGBoost and 0.5112 for Random Forest. While these figures exceed the benchmark values set for the project, they are close to the threshold and suggest that the models perform only slightly better than a random guess in the complex task of predicting stock market trends. Therefore, starting to trade based on these models without further refinement and validation could be premature and risky.

To enhance the realism and potential practical application of the analysis, several steps could be taken:

Feature Engineering: Further development of more sophisticated features that could capture market sentiment, economic indicators, or inter-market relationships might provide the models with more predictive power.

Additional Data: Incorporating a more extensive dataset with a longer historical period or higher frequency data, such as minute-by-minute price changes, might improve the model's learning capability.

Model Complexity: Adjusting the complexity of the model to ensure it captures relevant patterns without overfitting. This can involve tuning hyperparameters or even exploring simpler models that might perform better.

Ensemble Methods: Combining the predictions of multiple models to potentially smooth out errors and improve overall prediction accuracy.

Risk Management: Implementing stop-loss orders, position sizing, and other risk management techniques to ensure that any trading strategy based on the model does not result in significant losses.

b. Regarding overfitting, my model is still possible for being overfitting because I didn’t use Cross-Validation because of the time limitation. Also, we only used 10 stock data to train the models.

The steps I have done are:

Regularization: Hyperparameters such as gamma in XGBoost and constraints on the depth of trees in Random Forest are examples of regularization techniques that help prevent the model from becoming too complex and memorizing the training data.

Pruning: In tree-based models, limiting the depth of the trees or pruning them to remove branches that have little power in predicting the target variable can reduce overfitting.

Overall, while the models show promise, they should be further refined and subjected to rigorous validation before any real trading is considered. Additional data, careful feature selection, and robust validation techniques are essential to develop a model that can be confidently used in trading decisions.