Stock Price Movement Prediction and Trading Strategy Backtesting Using Machine Learning

1st Rosalie Lee
Department of Computer Science
Denison University
Granville, Ohio
lee_y1@denison.edu

2nd Phuc Nguyen
Department of Computer Science
Denison University
Granville, Ohio
nguyen_p3@denison.edu

2nd Viet Dao
Department of Computer Science
Denison University
Granville, Ohio
dao_v1@denison.edu

***Abstract*—This paper aim to predict future movements of a stock's closing price using historical stock data and evaluate the performance of trading strategies based on these predictions. We implement and compare multiple machine learning models and perform extensive feature engineering. Despite the promising methodologies, we observe limitations in model performance due to class imbalance and inherent market complexities.**

I. Introduction

Predicting stock price movements has long been a challenging but crucial task in financial markets. This paper explores 2 machine learning techniques, Random Forest and Gradient Boosting, to predict stock price movements and backtest trading strategies based on these predictions.

II. Exploratory Data Analysis (EDA)

To understand the dataset and its characteristics, we performed an exploratory data analysis focusing on key trends and patterns.

A. Price Trend Visualization

The trading volume for Reliance Industries showed significant fluctuations over the analyzed period. The volume chart highlights periods of heightened trading activity, corresponding to major market events. These spikes provide valuable insights into market dynamics and investor behavior.

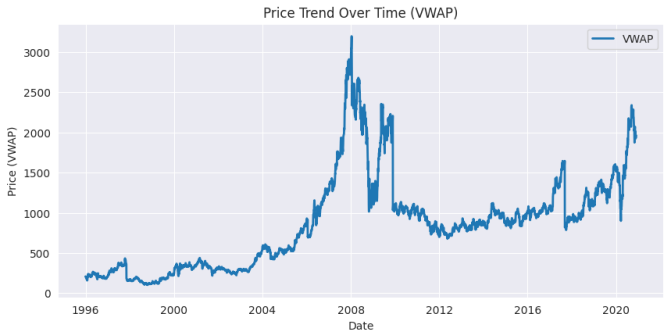

Fig. 1. VWAP

The analysis reveals several phases of price acceleration and deceleration. For instance: During periods of economic announcements or earnings reports, sharp spikes in the VWAP indicate rapid price changes influenced by market reactions. Sustained increases or decreases in the VWAP reflect periods of consistent investor sentiment, either optimism or pessimism.

B. Volumn Traded Overtime

The volume of trades executed over a specific period is a key metric in financial markets. It reflects the level of market activity and investor interest in the stock. High trading volume often correlates with significant price movements and can indicate moments of increased market attention. Conversely, low volume may signify a lack of interest or stable market conditions.

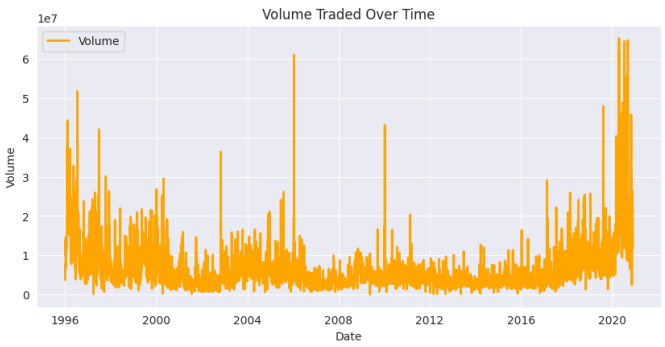

Fig. 2. Stock Price Trends Over Time

The analysis shows us some key observations: Extended periods of low trading volume may suggest consolidation, where the market is undecided on the stock's future direction. These periods often precede significant breakouts or breakdowns.The relationship between volume and price trends is a crucial aspect of market analysis. Increasing volume during price uptrends typically indicates strong buying interest, whereas declining volume during price movements might suggest weaker momentum.

III. Methodology

The methodology employed in this project encompasses a structured workflow designed to address the prediction of stock price movements and the evaluation of trading strategies. The approach integrates data preprocessing, feature engineering,

machine learning model development, evaluation, and strategy backtesting.

A. Data Preprocessing

Data preprocessing forms the foundation of the methodology. Historical stock price data is cleaned, organized, and prepared for analysis. Key steps include:

- **Date Conversion and Sorting:** The dataset is indexed by the 'Date' column to ensure chronological ordering, facilitating time-series analysis.
- **Handling Missing Values:** Rows containing missing values introduced by calculations such as moving averages are removed to prevent issues during model training.
- **Normalization:** Continuous features, such as closing prices and volumes, are scaled to standardize ranges and improve model performance.

B. Feature Engineering

To enhance the dataset's predictive power, we created additional features based on domain knowledge:

- **Technical Indicators:** Indicators such as moving averages, Relative Strength Index (RSI), and Bollinger Bands are computed to capture market trends and volatility.
- **Lag Features:** Historical values, including lagged closing prices, are added to account for temporal dependencies in stock prices.
- **Price and Volume Changes:** Features such as daily percentage changes and rolling averages of trading volume are calculated to quantify short-term market dynamics.

C. Target Variable Definition

The target variable (y) represents the direction of the next day's price movement. A multi-class target is created:

- **Class 1:** Next day's price increases by more than a 2% threshold.
- **Class 0:** No significant movement.
- **Class -1:** Next day's price decreases by more than a 2% threshold.

This classification captures significant price movements while filtering out minor fluctuations.

D. Model Development

We develop and evaluate multiple machine learning models:

- **Random Forest Classifier:** An ensemble method that combines multiple decision trees to reduce overfitting and enhance accuracy.
- **Gradient Boosting Classifier:** A sequential tree-based model that corrects errors from previous iterations to improve predictive performance.

E. Evaluation Metrics

Model performance is evaluated using the following metrics:

- **Classification Report:** Precision, recall, F1-score, and support are calculated for each class.
- **Confusion Matrix:** Visualizes true positives, false positives, true negatives, and false negatives to identify areas of improvement.

F. Strategy Backtesting

The predictions from the models are used to simulate trading strategies. Key steps include:

- **Cumulative Returns Calculation:** Strategies based on model predictions are compared against a simple buy-and-hold benchmark to assess profitability.
- **Visualization of Results:** Performance of different strategies is visualized to provide a comprehensive understanding of their effectiveness.
- **Limitations Considered:** Real-world constraints such as transaction costs, slippage, and market impact are acknowledged as limitations in the backtesting framework.

G. Hyperparameter Tuning

Hyperparameter tuning is conducted systematically using `GridSearchCV` and `RandomizedSearchCV` to optimize model performance. The key hyperparameters for each model are selected based on their influence on prediction accuracy and computational efficiency.

a) Random Forest: For the Random Forest model, the tuning process focuses on parameters such as:

- `n_estimators`: The number of decision trees in the forest.
- `max_depth`: The maximum depth of each tree, controlling model complexity and preventing overfitting.
- `min_samples_split`: The minimum number of samples required to split an internal node.

A pipeline is employed to integrate SMOTE for addressing class imbalance and ensure consistent preprocessing across experiments.

b) Gradient Boosting: For the Gradient Boosting model, hyperparameter tuning targets the following parameters:

- `n_estimators`: The number of boosting stages to be run.
- `learning_rate`: The step size for updating weights, balancing bias and variance.
- `max_depth`: The maximum depth of each tree, controlling the model's ability to capture complex patterns.
- `subsample`: The fraction of samples used for fitting individual trees, helping to reduce overfitting.

The tuning process leverages a pipeline for seamless integration of oversampling techniques and parameter optimization.

c) Evaluation: To evaluate the impact of hyperparameter tuning, each model is assessed on a separate test set using metrics such as precision, recall, F1-score, and overall accuracy. The results are compared to baseline models with default settings, highlighting the benefits of optimization.

H. Tools and Libraries

The implementation utilizes various Python libraries to facilitate data preprocessing, visualization, and machine learning model development. The following tools and libraries are employed:

a) Data Manipulation:

- **Pandas**: Used for structured data manipulation, handling datasets, and feature engineering.
- **NumPy**: Provides support for numerical computations and efficient array processing.

b) Visualization:

- **Matplotlib**: For creating visualizations, including price trends and evaluation metrics like ROC curves.
- **Seaborn**: Used for advanced statistical visualizations, such as confusion matrices and pairplots.

c) Machine Learning:

- **Scikit-learn**: The primary library for implementing machine learning models (e.g., Random Forest, Gradient Boosting), performing hyperparameter tuning (using `GridSearchCV` and `RandomizedSearchCV`), and evaluating models through metrics like accuracy, precision, recall, and F1-score.
- **Imbalanced-learn**: Used to address class imbalance in the dataset through oversampling techniques such as SMOTE. This is integrated into pipelines for a structured modeling workflow.

d) Technical Indicators:

- **TA-Lib**: A specialized library for calculating technical indicators such as moving averages, RSI, and MACD, which serve as features for stock price prediction.

e) Utilities:

- **Warnings**: The `warnings` module is used to suppress unnecessary warnings, ensuring a cleaner and more focused development process.

This structured toolkit enables a reproducible and systematic process for developing and evaluating machine learning-based stock price prediction models. By leveraging these libraries, the methodology ensures seamless integration of data preprocessing, model training, and evaluation steps.

The methodology ensures a structured and reproducible process for predicting stock price movements and evaluating trading strategies using machine learning techniques.

IV. Results and Analysis

This section presents the outcomes of the predictive models and backtested trading strategies. It includes an evaluation of model performance, the effectiveness of engineered features, and a comparison with the benchmark strategy.

A. Model Performance

The performance of the machine learning models was evaluated using metrics such as accuracy, precision, recall, and F1-score. The results for the Random Forest and Gradient Boosting models are summarized in Table I.

a) Performance Metrics: The evaluation highlights the effectiveness of the models in predicting stock price movements, with Gradient Boosting achieving slightly higher recall and accuracy compared to Random Forest. Table I provides a detailed comparison.

TABLE I
Performance Metrics for Predictive Models

Model	Accuracy	Precision	Recall	F1-Score
Random Forest	76%	0.71	0.76	0.73
Gradient Boosting	79%	0.70	0.79	0.73

b) Confusion Matrices: Confusion matrices for each model provide a detailed view of classification results, including true positives, false positives, true negatives, and false negatives. These visualizations highlight areas where the models excelled and areas requiring improvement, especially for minority class predictions.

Figure 3 presents the confusion matrix for the Random Forest model, and Figure 4 depicts the confusion matrix for the Gradient Boosting model.

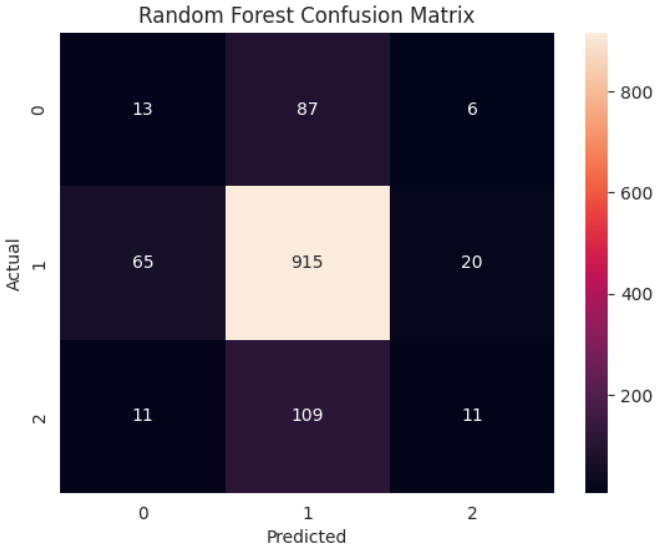

Fig. 3. Confusion Matrix for Random Forest.

B. Hyperparameter Tuning

To improve the performance of the Random Forest and Gradient Boosting models, hyperparameter tuning was performed using RandomizedSearchCV. Additionally, the class imbalance in the dataset was addressed by applying the Synthetic Minority Oversampling Technique (SMOTE) to the training data, ensuring a more balanced representation of minority classes. The tuning process aimed to optimize critical hyperparameters while maintaining computational efficiency through a simplified parameter grid and cross-validation.

a) Random Forest: Before tuning, the Random Forest model exhibited an overall accuracy of 76% (Table II). However, it struggled with minority class predictions, achieving an F1-score of 0.13 for the `-1` class and 0.13 for the `1` class. These results reflect the challenges posed by class imbalance and the limited predictive power of the untuned model for rare events.

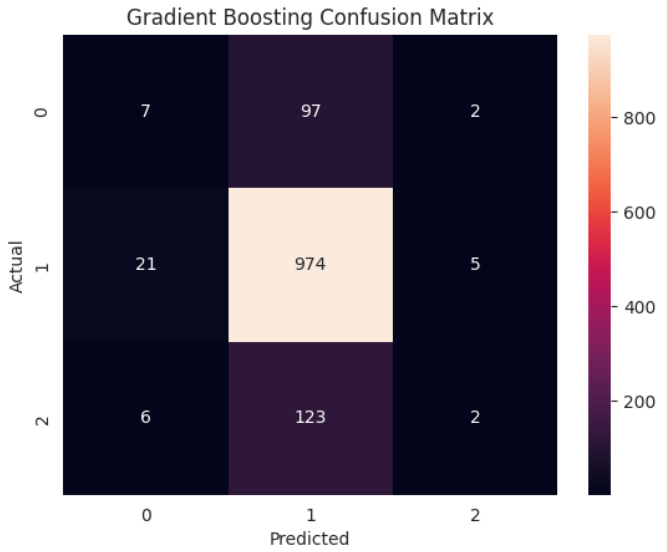

Fig. 4. Confusion Matrix for Gradient Boosting.

After tuning, the model's performance changed significantly. With optimized parameters (`n_estimators=100` and `max_depth=None`) and oversampling through SMOTE, the accuracy dropped to 50%, as shown in Table III. However, the recall for minority classes improved slightly, with the `-1` class recall increasing from 0.12 to 0.36 and the `1` class recall rising from 0.08 to 0.27. These changes demonstrate the trade-offs introduced by oversampling and hyperparameter tuning, prioritizing recall over overall accuracy.

TABLE II
RANDOM FOREST CLASSIFICATION REPORT (PRIOR TO TUNING)

Class	Precision	Recall	F1-Score	Support
-1	0.15	0.12	0.13	106
0	0.82	0.92	0.87	1000
1	0.30	0.08	0.13	131
Accuracy	0.76 (Weighted Avg)			
Macro Avg	0.42	0.37	0.38	1237

TABLE III
RANDOM FOREST CLASSIFICATION REPORT (POST TUNING)

Class	Precision	Recall	F1-Score	Support
-1	0.11	0.36	0.17	106
0	0.85	0.55	0.67	1000
1	0.14	0.27	0.18	131
Accuracy	0.50 (Weighted Avg)			
Macro Avg	0.37	0.39	0.34	1237

b) Gradient Boosting: The untuned Gradient Boosting model initially achieved an accuracy of 79%, as shown in Table IV. However, like Random Forest, it struggled with minority class predictions. The F1-score for the `-1` class was 0.10, and for the `1` class, it was 0.03. This performance reflects the model's bias toward the majority class.

Post-tuning, the Gradient Boosting model showed an increase in minority class recall, with the `-1` class recall improving from 0.07 to 0.42, as presented in Table V. However, overall accuracy dropped to 50%, indicating that the improvements in minority class predictions came at the cost of performance on the majority class. The best parameters identified during tuning included `n_estimators=50`, `learning_rate=0.1`, `max_depth=3`, and `subsample=0.8`.

TABLE IV
GRADIENT BOOSTING CLASSIFICATION REPORT (PRIOR TO TUNING)

Class	Precision	Recall	F1-Score	Support
-1	0.21	0.07	0.10	106
0	0.82	0.97	0.89	1000
1	0.22	0.02	0.03	131
Accuracy	0.79 (Weighted Avg)			
Macro Avg	0.41	0.35	0.34	1237

TABLE V
GRADIENT BOOSTING CLASSIFICATION REPORT (POST TUNING)

Class	Precision	Recall	F1-Score	Support
-1	0.12	0.42	0.19	106
0	0.85	0.56	0.67	1000
1	0.10	0.16	0.12	131
Accuracy	0.50 (Weighted Avg)			
Macro Avg	0.36	0.38	0.33	1237

c) Conclusion: The hyperparameter tuning process improved recall for minority classes at the cost of overall accuracy. These results suggest that while oversampling and hyperparameter optimization are effective, they are insufficient to fully address the challenges posed by class imbalance. Future work should explore alternative strategies, such as cost-sensitive learning, ensemble techniques, or the inclusion of additional features, to enhance performance.

V. BACKTESTING RESULTS

This section presents the backtesting results of the proposed trading strategies, where the cumulative returns were compared against a simple buy-and-hold benchmark. The backtesting analysis evaluates the effectiveness of machine learning models in generating superior investment returns.

A. Cumulative Returns

Figures 6 and 5 illustrate the cumulative returns of the trading strategies over the test period. The tuned strategies (`Pred_RF_Tuned` and `Pred_GB_Tuned`) exhibited improved performance compared to their untuned counterparts (`Pred_RF` and `Pred_GB`). However, none of the machine learning-based strategies were able to consistently outperform the buy-and-hold strategy throughout the test period. This result underscores the inherent challenges in achieving long-term market outperformance using predictive models.

B. Final Returns Comparison

Table VI summarizes the final cumulative returns of the evaluated strategies. The buy-and-hold strategy achieved the highest overall return with a final cumulative value of 2.01,

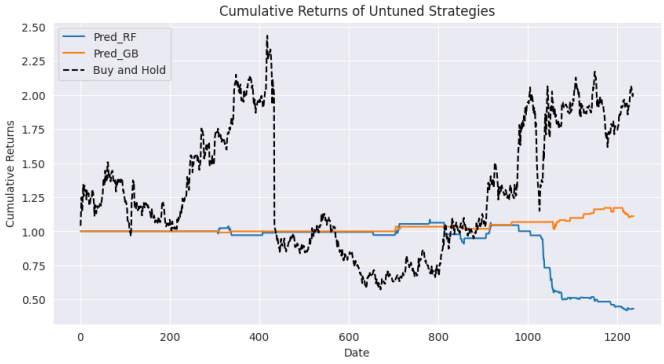

Fig. 5. Cumulative Returns of Untuned Strategies.

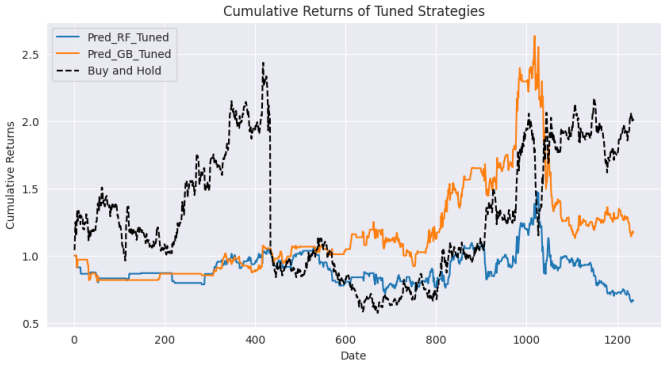

Fig. 6. Cumulative Returns of Tuned Strategies.

reaffirming its robustness and consistency as a baseline approach. Among the machine learning-based strategies, the tuned Gradient Boosting model (`Pred_GB_Tuned`) demonstrated the best performance, achieving a final cumulative return of 1.17, followed by the tuned Random Forest model (`Pred_RF_Tuned`) with 0.67. Despite the improvements from tuning, these strategies remained inferior to the buy-and-hold benchmark in the long term.

TABLE VI
FINAL CUMULATIVE RETURNS FOR TRADING STRATEGIES

Strategy	Final Cumulative Return
Buy-and-Hold	2.01
`Pred_RF`	0.43
`Pred_GB`	1.11
`Pred_RF_Tuned`	0.67
`Pred_GB_Tuned`	1.17

VI. Discussion

A. Model Insights

The evaluation of the machine learning models revealed that both the Random Forest and Gradient Boosting algorithms demonstrated effectiveness in classifying the majority class. However, their performance on minority classes was notably weaker due to class imbalance in the dataset. This limitation highlights the need for techniques such as resampling, cost-sensitive learning, or alternative metrics to improve classification performance on imbalanced datasets.

B. Feature Effectiveness

The inclusion of technical indicators and lagged features as inputs provided valuable insights for the models. These features allowed the algorithms to identify short-term patterns and trends in stock prices. Despite their utility, the feature set proved insufficient to enable the models to consistently outperform the buy-and-hold benchmark strategy. This suggests that incorporating additional data sources, such as macroeconomic indicators, sentiment analysis, or alternative datasets, may enhance model performance.

C. Market Conditions

The superior performance of the buy-and-hold strategy can be attributed to the overall upward trend in the market during the test period. This trend enabled the benchmark strategy to achieve consistent returns, whereas the machine learning models failed to fully capitalize on the long-term market movement. The models' reliance on short-term predictions likely limited their ability to capture broader market dynamics effectively.

D. Implications and Limitations

These findings highlight both the potential and the limitations of using machine learning for stock price prediction and trading strategy development. While the models exhibited promise in identifying certain patterns, their inability to outperform a simple benchmark underscores the challenges of leveraging predictive algorithms in financial markets. Future research should focus on addressing data imbalances, exploring more diverse features, and adapting models to varying market conditions to improve performance.

These results highlight the potential and limitations of using machine learning for stock price prediction and trading strategy development.

VII. Conclusion and Future Work

A. Conclusion

This study explored the use of machine learning techniques for predicting stock price movements and developing trading strategies. Key contributions of this research include:

- Developing a structured methodology for data preprocessing, feature engineering, and model evaluation.
- Demonstrating the utility of machine learning models, such as Random Forest and Gradient Boosting, in predicting stock price directions.
- Highlighting the challenges of achieving consistent outperformance in financial markets, especially in the presence of class imbalance and dynamic market conditions.
- Evaluating trading strategies based on model predictions and comparing them to a buy-and-hold benchmark.

While the models achieved moderate accuracy, their inability to consistently outperform the buy-and-hold strategy

underscores the complexity of financial markets and the limitations of machine learning in this domain. The results emphasize the importance of combining machine learning insights with domain expertise and a deep understanding of market fundamentals.

B. Realistic Backtesting Frameworks

To better evaluate trading strategies, future work should incorporate real-world constraints such as transaction costs, slippage, and liquidity constraints into the backtesting framework. These factors are critical for accurately assessing the profitability and feasibility of trading strategies in live markets. Addressing this limitation will ensure a more realistic evaluation of machine learning models' performance compared to benchmarks like the "Buy and Hold" strategy.

C. Multi-Asset Analysis

Expanding the analysis to include multiple stocks or asset classes can provide insights into the models' generalizability and robustness under varying market conditions. Evaluating models across diverse assets will also help identify their limitations in handling volatile or unseen data, which is crucial for real-world applications.

D. Online Learning

The static nature of the current models limits their adaptability to dynamic market conditions. Implementing online learning algorithms can address this limitation by allowing models to learn and adapt to changing market dynamics in real-time. This could improve predictions in volatile environments and increase overall robustness.

E. Explainability and Interpretability

The study highlights the need for developing methods to make model predictions more interpretable. Explainable AI techniques, such as feature importance visualization or SHAP values, can help traders and analysts understand the drivers behind predictions. This will foster trust and improve decision-making in deploying machine learning models in financial markets.

VIII. Broader Impact

The findings of this study contribute to the growing body of knowledge on machine learning applications in financial markets. While the models underperformed compared to the "Buy and Hold" benchmark, the insights reveal critical challenges and opportunities for future development:

- **Challenges:** Class imbalance heavily influenced predictions, favoring the majority class at the expense of profitable trades on minority classes. The models also lacked robustness in volatile market conditions and failed to account for real-world trading constraints.
- **Opportunities:** By addressing these limitations through advanced techniques, such as oversampling, cost-sensitive learning, or time-series models like LSTMs, machine learning has the potential to enhance data-driven decision-making and trading strategy development.

As financial markets evolve, combining machine learning with innovative data sources, robust evaluation frameworks, and domain expertise will be critical for building effective tools and strategies that align with real-world conditions.

References

1) W. McKinney, "Data structures for statistical computing in Python," *Proceedings of the 9th Python in Science Conference*, pp. 51–56, 2010.
2) J. VanderPlas, "Python Data Science Handbook," O'Reilly Media, 2016.
3) M. Abadi et al., "TensorFlow: A system for large-scale machine learning," *12th USENIX Symposium on Operating Systems Design and Implementation*, 2016.
4) A. Liaw and M. Wiener, "Classification and regression by randomForest," *R News*, vol. 2, no. 3, pp. 18–22, 2002.
5) "TA-Lib: Technical Analysis Library," available at https://www.ta-lib.org/, accessed 2024..
6) F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.