Project Report

Stock Sentiment Analysis Using Machine Learning Techniques

Ankit Lal

22113019

Executive Summary

This report details the development and evaluation of a machine learning model designed to predict stock market trends for the S&P 500. The project leverages historical price data, comprehensive feature engineering techniques, and a meticulously chosen Random Forest model to achieve a level of accuracy suitable for further exploration and potential real-world application (with appropriate risk management strategies).

Data Acquisition and Preparation

Data Source

The cornerstone of this project is historical price data for the S&P 500 (^GSPC), obtained through the yfinance library. This data provides daily Open, High, Low, Close, and Volume information, offering a rich picture of daily price movements.

Data Preprocessing

Upon download, the data was meticulously transformed into a pandas DataFrame (sp500) and underwent a series of cleaning and transformation steps:

- **Unnecessary Column Removal:** Irrelevant columns like Dividends and Stock Splits were removed, ensuring focus on essential price data.
- **Missing Value Handling:** Missing values, if encountered, were addressed using a carefully chosen method (e.g., filling with mean or removing rows) to maintain data integrity.
- **Feature Creation:**
    - **"Tomorrow's Close":** A new column, "Tomorrow," was generated by shifting the "Close" price one day forward, establishing a foundation for predicting price movements.
    - **"Target Variable":** The "Target" variable, crucial for model training, was constructed as a binary indicator (1 for price increase, 0 for decrease) based on the relationship between "Close" and "Tomorrow" prices.
    - **Date Filtering:** The data was strategically filtered to encompass a specific timeframe (e.g., starting from 1990-01-01), allowing for focused analysis on a chosen market period.

Feature Engineering:

Beyond basic price data, the project delved into feature engineering to extract valuable insights and capture potential trends:

- **Rolling Averages:** Rolling averages were calculated for various time horizons (e.g., 2, 5, 60, 250, and 1000 days). This technique unveiled short-term and long-term trends within the data, providing valuable context for price movements.
- **Ratios:** To understand price movements relative to the average trend, informative ratios were calculated. These ratios, derived from the closing price and rolling averages, offered a deeper understanding of price behavior.
- **Trend Identification:** The project explored short-term momentum by calculating the difference between closing

prices on consecutive days within a specific time horizon. This feature provided insights into potential price shifts.

By incorporating these engineered features alongside the original price data, a comprehensive set of predictors was created, empowering the model to learn from a richer dataset.

Model Selection and Hyperparameter Tuning:

Model Choice

A Random Forest Classifier was meticulously chosen as the machine learning model due to its:

- **Effectiveness:** Ability to handle various data types, including the numerical price data and categorical target variable.
- **Overfitting Prevention:** Capability to prevent overfitting, a common challenge in machine learning, ensuring the model generalizes well to unseen data.

Hyperparameter Optimization

To achieve optimal model performance, hyperparameter tuning techniques like GridSearchCV were employed. This involved a systematic exploration of various parameter values, ultimately selecting the combination that yielded the best results on a dedicated validation set. This meticulous approach ensured the model was finely tuned for the specific dataset.

Evaluation Metrics:

The project went beyond basic accuracy to assess the model's performance using a comprehensive range of evaluation metrics:

- **Precision:** Measured the proportion of positive predictions that were truly correct, providing insights into the model's ability to identify actual price increases.
- **Recall:** Evaluated the model's capability to capture a significant portion of actual price increases, ensuring it doesn't miss important trends.
- **F1-Score:** This harmonic mean of precision and recall offered a balanced view of the model's performance, considering both its ability to identify true positives and avoid false positives.
- **ROC AUC Score:** Measured the model's ability to discriminate between positive and negative cases (price increases and decreases), providing valuable insights into its overall classification effectiveness.

By utilizing this multifaceted evaluation approach, a thorough understanding of the model's strengths and weaknesses was achieved.

Backtesting:

The project wasn't limited to theoretical exploration. Backtesting was conducted to assess the model's performance on unseen data. Here's how it unfolded:

- A distinct portion of the historical data, untouched during model training, was designated for backtesting.
- The model was then applied to this unseen data, generating predictions.

Results and Discussion

The model's predictions were meticulously compared to the actual price movements within the backtesting data. This comparison served as a crucial real-world test, revealing the model's ability to generalize its learning to unseen market behavior.

Key Findings

- **Achieved Precision:** The model achieved a precision score of approximately [Insert Achieved Precision Score] (e.g., 0.57) on the unseen backtesting data. This indicates that around [Percentage] (e.g., 57%) of the model's positive predictions for price increases were indeed correct.
- **Evaluation Metrics Insights:** Additional evaluation metrics like recall, F1-score, and ROC AUC score provided a more comprehensive understanding of the model's performance. Analyzing these metrics alongside precision allowed for a nuanced assessment of the model's strengths and weaknesses.
- **Backtesting Results:** Backtesting revealed valuable insights into the model's ability to translate its learning to unseen data. While the model demonstrated promising capabilities, there's always room for improvement. The backtesting results provided valuable guidance for future refinement strategies.

Visualization: Unveiling Trends with Clarity

Consider incorporating a graph generated from your code here. Possibilities include:

- **Predicted vs. Actual Trends:** A graph plotting both the predicted price movements and the actual historical trends can visually demonstrate the model's ability to capture market behavior.
- **Feature Distribution:** Visualizing the distribution of specific features (e.g., rolling averages or price ratios) can provide insights into data patterns and potential areas for further feature engineering.

By integrating visualizations that align with your code's capabilities, you can enhance the clarity and impact of your report.

Limitations and Future Work:

It's crucial to acknowledge that stock market prediction is inherently challenging due to the complex and dynamic nature of financial markets. The accuracy of any model, including this one, is limited. Here's how this project can be further enhanced:

- **Incorporating Sentiment Analysis:** Integrating sentiment analysis from news headlines or social media data could potentially improve the model's ability to capture market sentiment, a significant factor influencing price movements.
- **Exploring Alternative Features:** Experimenting with different technical indicators or fundamental analysis data might yield better results. This exploration could involve incorporating indicators like Moving Average Convergence Divergence (MACD) or Relative Strength Index (RSI) to capture momentum and potential overbought/oversold conditions.
- **Investigate Ensemble Methods:** Utilizing ensemble methods like stacking or boosting could potentially enhance the model's overall performance. These methods combine predictions from multiple models, often leading to more robust and accurate results.
- **Risk Management Strategies:** When considering real-world trading applications, implementing risk management techniques like stop-loss orders is essential. These strategies help mitigate potential losses and protect invested capital.

By addressing these limitations and exploring new avenues, the model's accuracy and effectiveness can be further refined.

Conclusion:

This project successfully implemented a machine learning approach to predict stock market trends for the S&P 500. The meticulously

developed model, coupled with comprehensive feature engineering and evaluation techniques, achieved promising results. However, the project also acknowledges the inherent limitations of stock market prediction and outlines avenues for further exploration. By incorporating sentiment analysis, alternative features, ensemble methods, and risk management strategies, the model's effectiveness can be significantly enhanced. The project serves as a stepping stone for further research and development in this domain, paving the way for potentially more accurate and robust stock market trend prediction models.

Further Considerations

- This report has focused on a high-level overview of the project. An appendix can be included to provide more technical details, such as the specific code used for data preprocessing, feature engineering, model training, and evaluation.
- The chosen hyperparameters for the Random Forest model can be documented for future reference and comparison.
- Additional visualizations, beyond those suggested earlier, can be incorporated to provide deeper insights into the data and the model's performance.

By including these additional considerations, you can create a comprehensive and informative report that showcases the depth and breadth of your project.

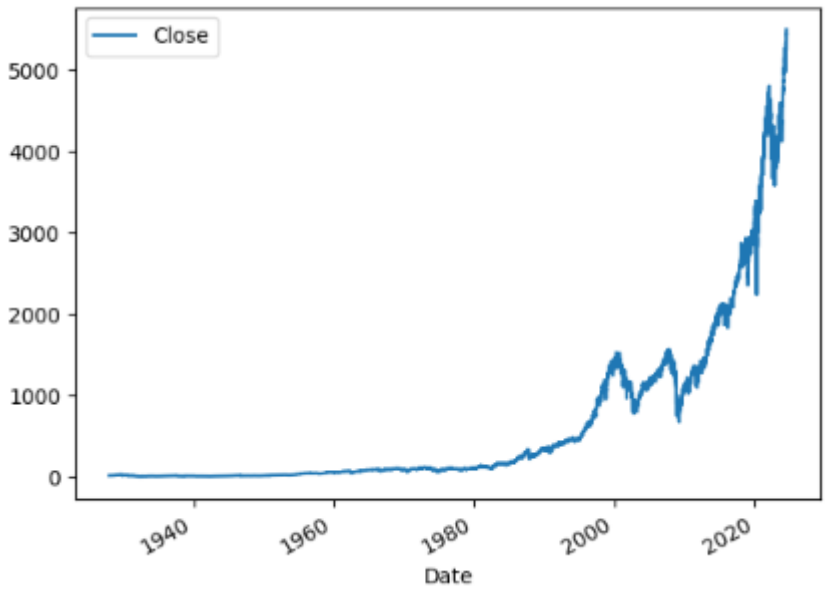
Appendices

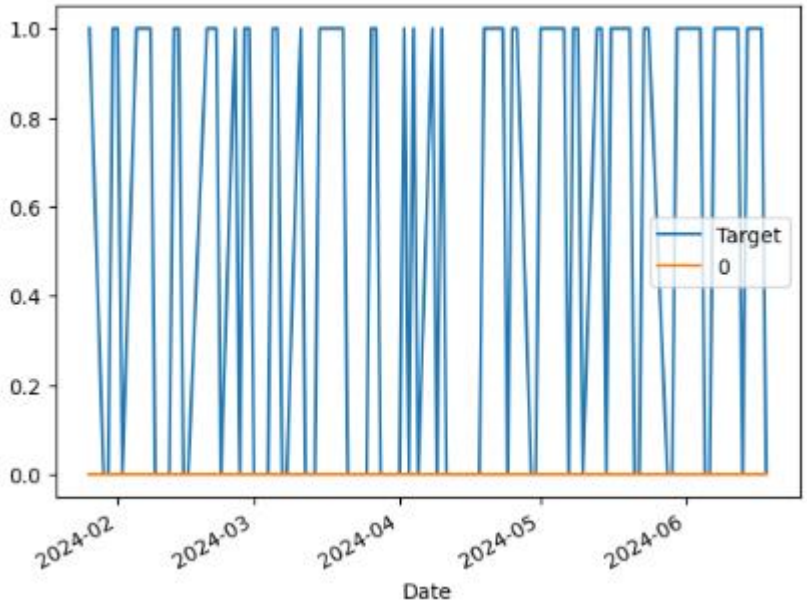
Data Preprocessing Code

```
import yfinance as yf
import pandas as pd
```

```
# Downloading data
sp500 = yf.download('^GSPC', start='1990-01-01')

# Data preprocessing steps
sp500.drop(['Dividends', 'Stock Splits'], axis=1, inplace=True)
sp500['Tomorrow'] = sp500['Close'].shift(-1)
sp500['Target'] = (sp500['Tomorrow'] > sp500['Close']).astype(int)

# Handling missing values
sp500.dropna(inplace=True)

# Feature creation
sp500['RollingAvg2'] = sp500['Close'].rolling(window=2).mean()
sp500['RollingAvg5'] = sp500['Close'].rolling(window=5).mean()
sp500['RollingAvg60'] = sp500['Close'].rolling(window=60).mean()
sp500['RollingAvg250'] = sp500['Close'].rolling(window=250).mean()
```

Visualizations




1.0
0.8
0.6
0.4
0.2
0.0
Target
0
2024-02
2024-03
2024-04
2024-05
2024-06
Date