S&P 500 Buy Signal Classifier

Group 2

Project 2

Rob, Phillip, Tony

July 9, 2024

Executive Summary

- Goal: Accurately predict S&P 500 one-day in the future.
- Analyze SP-500 index from 1927 to present day.
- Create a buy signal target variable.
- Classify buy signal success based upon next day's return.
- Achieved a 93% balanced accuracy score, but with caveats.
- 49% minority precision resulted 2X false positive trades.
- 100% recall on both majority and minority classes.
- Transactions costs and impact of high number of false positive trades need further examination.
- Not recommend to trade on this signal in its current form.

Exploratory Data Analysis

- Collection – Simple fetch from Yahoo Finance ticker function
- Clean Up – Dropped nulls, dividends, stock splits, volume, and close. Sorted by date in ascending order.
- Feature Engineering – Created simple moving averages, RSI, MACD, Bollinger Bands, Signal, and Return columns
- Signal Feature – Close price less than Bollinger Mid, RSI < 50, and MACD < 0.
- Target Variable – Signal > 0 and Return >0.

Exploratory Data Analysis Continued

- Winsorization – Experimented with different winsorization parameters to improve accuracy. Settled on 5% for upper and lower settings. Marginal accuracy effect.
- Random Sampling – Experimented with original data, oversampling, and undersampling to address imbalanced data. Settled on undersampling as it had a significant accuracy effect.
- Scaling – Experimented with standard scaling and min-max scaling. Settled on standard scaling. Marginal accuracy effect.

Methodical 10-Step Approach

1. Create an install package list.
2. Read SP-500 index using the YahooFinance (yf) ticker function.
3. Clean data set by dropping columns and null values.
4. Perform Feature Engineering:
    - Moving Averages
    - RSI
    - MACD
    - Bollinger Bands
    - Signal
    - Return

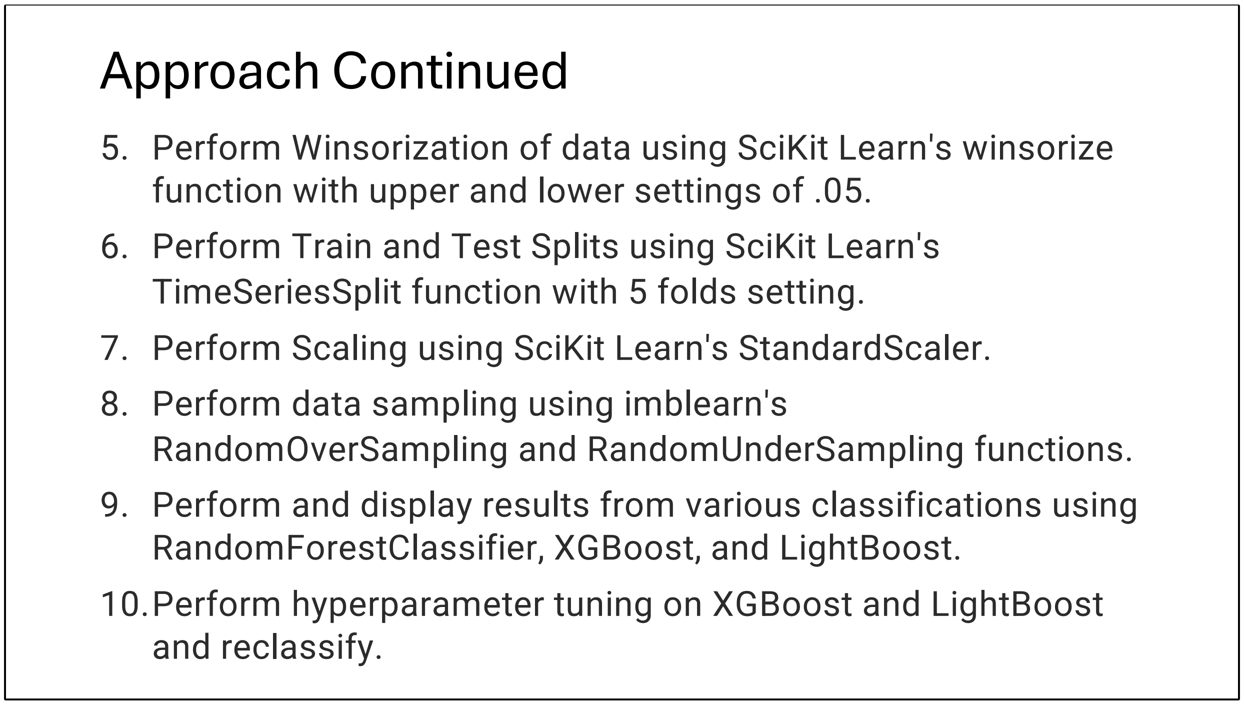
• Create an install package list.
This script automates the process of checking for the presence of certain Python packages (yfinance, sklearn, xgboost, lightgbm) and installs them if they are not already installed. It uses Python's subprocess module to execute pip install commands and sys module to determine the appropriate Python interpreter. This approach ensures that the required packages are available for use in subsequent parts of your code or environment. It also speeds up the default pip install process by skipping the verification process if the code is already installed.

• Read SP-500 index using the YahooFinance (yf) ticker function
The code retrieves historical price data of the S&P 500 index (^GSPC) using the yfinance library. It initializes a Ticker object for ^GSPC and then uses its history method to fetch all available historical data up to the current date.

• Clean data set by dropping columns and null values.
It adjusts the datetime index (sp500) by resetting it, removing timezone information, and setting 'Date' as the index.
It creates a new DataFrame (stock_df) from the modified sp500, drops unnecessary columns ('Volume', 'Dividends', 'Stock Splits'), and sorts the data by date in

ascending order. This prepares the data for further analysis or visualization, focusing on the stock price data without irrelevant columns and ensuring chronological order.

- Perform Feature Engineering:
    - Moving Averages - the creation of 10, 20, 30, 50, 100, & 200 day moving averages. Created to use in any data analysis.
    - RSI - the Relative Strength Index (RSI) is a valuable tool for technical analysts and traders seeking insights into the strength and direction of price movements in financial markets.
    - MACD - MACD is a versatile and powerful tool that is used in providing insights into the momentum and direction of price movements in financial markets.
    - Bollinger Bands - are a versatile tool used to analyze volatility, identify potential trading signals, and determine overbought or oversold conditions in financial markets.
    - Signal - often used in combinations, signals are used to help flag conditions that meet trading strategies.
    - Return -calculated from the previous day's closing, it's the % of change between closing prices.



Approach Continued

5. Perform Winsorization of data using SciKit Learn's winsorize function with upper and lower settings of .05.
6. Perform Train and Test Splits using SciKit Learn's TimeSeriesSplit function with 5 folds setting.
7. Perform Scaling using SciKit Learn's StandardScaler.
8. Perform data sampling using imblearn's RandomOverSampling and RandomUnderSampling functions.
9. Perform and display results from various classifications using RandomForestClassifier, XGBoost, and LightBoost.
10. Perform hyperparameter tuning on XGBoost and LightBoost and reclassify.

- Perform Winsorization of data using SciKit Learn's winsorize function with upper and lower settings of .05

This process clips the highs and lows that can be considered anomalies in the data.

- Perform Train and Test Splits using SciKit Learn's TimeSeriesSplit function with 5 folds setting using TimeSeriesSplit with 5 folds in SciKit Learn allows you to:
    - **Maintain Temporal Order**: Ensure that the model is trained only on past data and tested on future data.
    - **Progressive Training**: Train on progressively larger sets of data, which is useful for time series forecasting where more recent data can be more relevant.
    - **Cross-Validation**: Evaluate model performance on different segments of the data to get a more robust estimate of its performance.
    - This method is particularly useful for time series data where the order of observations is critical and traditional cross-validation techniques (which might shuffle the data) are not appropriate.

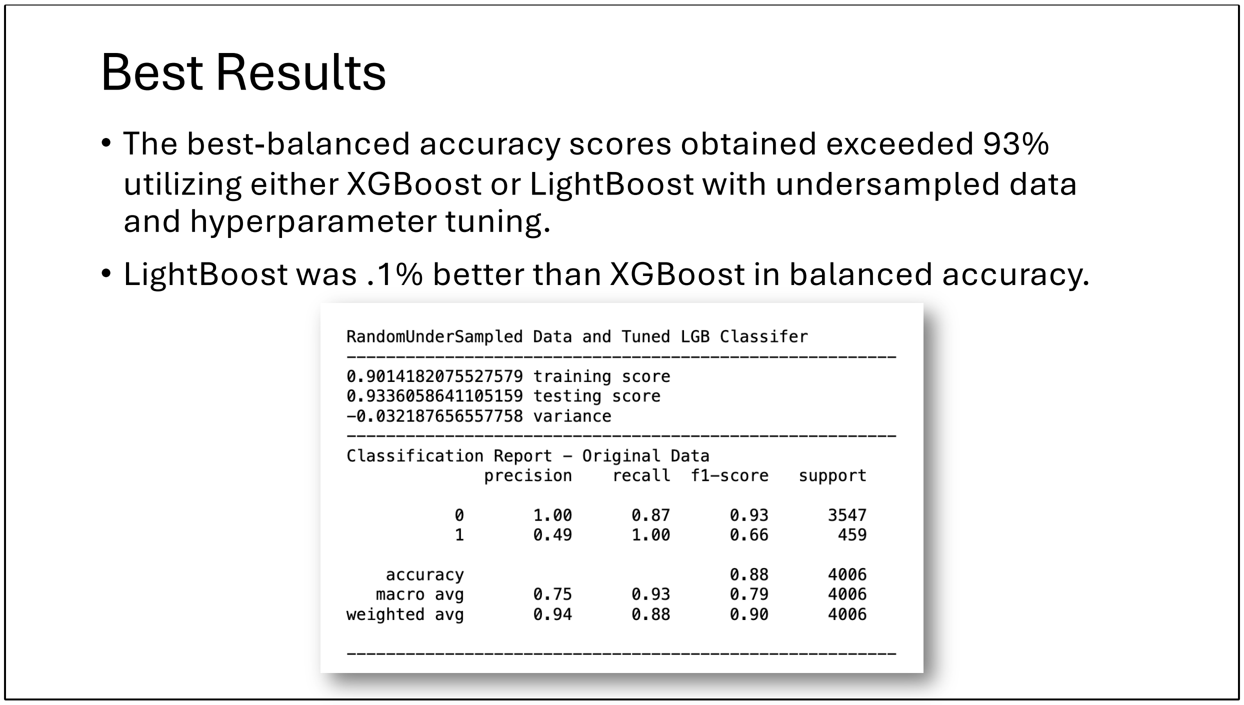
- Perform Scaling using SciKit Learn's StandardScaler
    - Scaling, when done correctly, can significantly improve the

performance and robustness of your time series models.

- Perform data sampling using imblearn's RandomOverSampling and RandomUnderSampling functions
- Temporal Order: Ensure that the temporal order of the time series data is not violated by random sampling techniques.Avoid Data Leakage: Apply sampling only to the training data after scaling, without influencing the test data.Sampling Techniques: Consider the impact of oversampling or undersampling on the temporal patterns in your data.
- If preserving the temporal structure is critical, you might want to explore alternative methods for handling imbalanced data, such as using time series-specific resampling techniques or weighting the classes in your model.

- Perform and display results from various classifications using RandomForestClassifier, XGBoost, and LightBoost
    - **RandomForestClassifier**: A versatile classifier that fits multiple decision trees on various sub-samples of the dataset and uses averaging to improve predictive accuracy and control over-fitting.
    - **XGBoost**: An optimized distributed gradient boosting library designed to be highly efficient, flexible, and portable.
    - **LightGBM**: A highly efficient gradient boosting framework that uses tree-based learning algorithms and is designed for faster training speed and higher efficiency.

- Perform hyperparameter tuning on XGBoost and LightBoost and reclassify
    - **Parameter Grids: We define the parameter grids for XGBoost and LightGBM. These grids include various hyperparameters like n_estimators, learning_rate, max_depth, subsample, and num_leaves for LightGBM.**
    - GridSearchCV: We use GridSearchCV with 3-fold cross-validation (cv=3) to find the best hyperparameters. The scoring metric is accuracy (scoring='accuracy').
    - **Best Parameters and Accuracy**: After tuning, we print the best parameters found and the corresponding accuracy on the test set.



Best Results

- The best-balanced accuracy scores obtained exceeded 93% utilizing either XGBoost or LightBoost with undersampled data and hyperparameter tuning.
- LightBoost was .1% better than XGBoost in balanced accuracy.

```
RandomUnderSampled Data and Tuned LGB Classifer
-------------------------------------------------------
0.9014182075527579 training score
0.9336058641105159 testing score
-0.032187656557758 variance
-------------------------------------------------------
Classification Report - Original Data
              precision    recall  f1-score   support

           0       1.00      0.87      0.93      3547
           1       0.49      1.00      0.66       459

    accuracy                           0.88      4006
   macro avg       0.75      0.93      0.79      4006
weighted avg       0.94      0.88      0.90      4006

-------------------------------------------------------
```

The classification report above shows that the model performs well on the training data and generalizes well on unseen data. The negative variance may indicate a favorable test set or slight underfitting. This would need further examination.

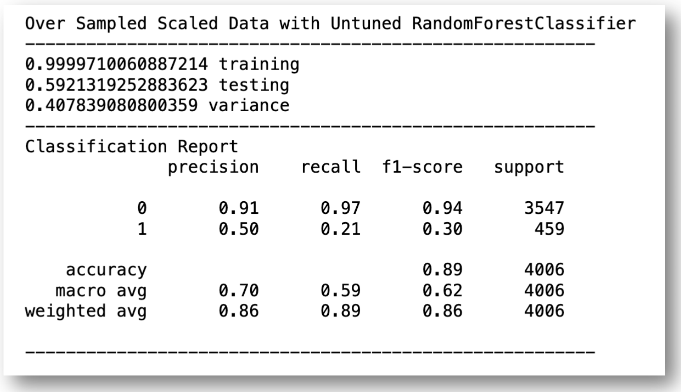
All unprofitable trades were correctly predicted as explained by the 100 % precision score in the majority class with only 13% false negatives.

The precision score on the minority class (buy signal) was only 49% while the recall was 100%. Such scores indicate that all actual buy signals were identified correctly in the testing set, but for every correct identification there was a false positive trade too.

Given the high number of false positives and the expected transaction costs of actually trading on this signal, it is possible this application does not produce a profitable trading strategy. Further analysis is required.

Worst Results

- The worst-balanced accuracy score obtained was 59% utilizing oversampled data and an untuned RandomForestClassifier.



```
Over Sampled Scaled Data with Untuned RandomForestClassifier
--------------------------------------------------------
0.9999710060887214 training
0.5921319252883623 testing
0.407839080800359 variance
--------------------------------------------------------
Classification Report
              precision    recall  f1-score   support

           0       0.91      0.97      0.94      3547
           1       0.50      0.21      0.30       459

    accuracy                           0.89      4006
   macro avg       0.70      0.59      0.62      4006
weighted avg       0.86      0.89      0.86      4006

--------------------------------------------------------
```

The classification report above shows that this model performed poorly well on the testing data.

All unprofitable trades were correctly predicted as explained by the 91% precision score in the majority class with only 3% false negatives.

The precision score on the minority class (buy signal) was only 50% while the recall was 21%. Only 1 in 5 profitable trades were identified.

Additional Questions and Next Steps

- The model clearly needs to improve the precision of the minority class. This may involve additional feature engineering with different technical indicators.
- Experiment with ensemble voting, bagging, and AdaBoost in accordance with Shareef, S. A., Kumar, C. T., Harika, K., & Yellisetti, S. (2023).
- Experiment with the buy signal generation could be done.
- Test the model as a short signal classifier.
- Also, the negative variance should be examined as well as the profitability of the model once transaction costs are added.

Conclusions

- Don’t quit your day job.
- Transactions costs and impact of high number of false positive trades need further examination.
- To be viable, minority precision must be increased all else being equal.

Questions?

Appendix

Technical Indicators

- The application utilizes the following technical indicators to enable machine learning and to generate a buy signal:
- Simple Moving Averages of 10, 20, 30, 50, 100, and 200 days. Moving averages are used to identify trends.
- Moving Average Convergence Divergence (MACD) shows the relationship between the two exponential moving averages (EMA) of 26-day EMA and 12-day EMA. The signal line is 9 EMA.
- Relative Strength Indicator oscillates between zero and 100. Readings above 50 indicate positive and uptrend momentum while readings below 50 show negative and downtrend momentum. Readings above 70 indicate overbought conditions and readings below 30 indicate oversold conditions.
- Bollinger Bands, developed by John Bollinger, gauge volatility to determine if an asset is over or undervalued. The center line is the 20-day SMA while the upper and lower bands are two standard deviations above and below the midline. The lines contract when volatility is low and expand when volatility is high.

Appendix

Buy Signal

- Buy low and sell high is the goal.
- This application generates a buy signal target variable of 1 when all the following are true:
    - The Relative Strength Indicator (RSI) is less than 50.
    - The MACD is less than zero.
    - The last closing price is less than the Bollinger mid line.
    - For training, the next day's percent return must be greater than zero.
- When these conditions are false, the target variable is 0.

Appendix

Winsorization

- Winsorizing or winsorization is the transformation of statistics by limiting extreme values in the statistical data to reduce the effect of possibly spurious outliers. It is named after the engineer-turned-biostatistician Charles P. Winsor (1895–1951). The effect is the same as clipping in signal processing.

References

Shareef, S. A., Kumar, C. T., Harika, K., & Yellisetti, S. (2023). Predicting the Stock Market Prices Using Ensemble and Fbprophet Model. In Information and Communication Technology for Competitive Strategies (ICTCS 2022) Intelligent Strategies for ICT (pp. 201-212). Singapore: Springer Nature Singapore.

https://www.investopedia.com/terms/m/movingaverage.asp

https://www.investopedia.com/terms/b/bollingerbands.asp

https://blog.elearnmarkets.com/top-5-momentum-indicators/

https://xgboost.readthedocs.io/en/stable/parameter.html

https://lightgbm.readthedocs.io/en/latest/Parameters-Tuning.html

https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.mstats.winsorize.html

https://pypi.org/project/yfinance/

https://en.wikipedia.org/wiki/Winsorizing