Predicting Significant Price Increases in the Chinese Stock Market

Authors: Cyrus Kurd, Anouksha Rajesh, Niranjan Sundararajan, Clarence Chen, Jacob Zhang

Introduction

The Chinese stock market presents unique opportunities and challenges for quantitative trading strategies. One particularly lucrative yet risky phenomenon is the "limit-up" event, where a stock's price reaches the daily maximum allowable increase, typically a 10% rise. These events are rare and often followed by sharp corrections, making them high-risk, high-reward opportunities. Our project aims to predict significant price increases, or "blow-ups," in the Chinese stock market by leveraging machine learning techniques, specifically using XGBoost and technical analysis indicators. By accurately predicting these events, investors can potentially capitalize on substantial short-term gains.

Data Description and Preprocessing

We utilized minute-level stock data aggregated into daily records for multiple stocks in the Chinese market. The dataset includes key features such as opening price, closing price, high, low, volume, and other market indicators.

- **Feature Selection and Cleaning**: Rows with null values after feature engineering were removed due to their minimal impact on the overall data volume (~.1% of total).
- **Target Variable Creation**: The target variable y was defined based on our objective to predict limit-up events. We considered stocks that increased by at least 9.5% (just below the 10% cap to account for price denominations) over a specified period as positive cases.

Exploratory Data Analysis

- The distribution of stock prices follows a log-normal pattern with large tails.
- No significant relationship was found between price variability and closing price.
- A high degree of positive correlation exists among many stocks, indicating market-wide trends.

Feature Variability:

- Standard deviation analysis highlighted the necessity for feature normalization.
- Principal Component Analysis (PCA) revealed that seven principal components explain the majority of variance, suggesting that dimensionality reduction could be beneficial.

Class Imbalance

Limit-up events are inherently rare, leading to a highly imbalanced dataset where positive cases constitute less than 0.1% of the data. To address this, we:

- **Adjusted the Threshold of Price Increase**: Set a flexible threshold (e.g., 9.5%) to include near limit-up events.
- **Tuned the Buying Period**: Set the number of days preceding the maxout to 10, with linspace between 0 and 1 to incentivize buying across all days, especially favoring the days closest to the maxout.
- **Defined the Number of Consecutive Limit-Up Days**: Focused on stocks with at least three consecutive limit-up days to capture sustained upward movements.
- **Handled Imbalance in Modeling**: Used the scale_pos_weight parameter in XGBoost and weighted loss training in RNN to balance the positive and negative classes during training.

Feature Engineering with Technical Indicators

To enhance predictive power, we incorporated several technical analysis indicators known for capturing stock trends, and momentum shifts, and market volatility:

- **Trend Indicators**: Simple Moving Average (SMA), Exponential Moving Average (EMA), Moving Average Convergence Divergence (MACD).
- **Momentum Indicators**: Relative Strength Index (RSI), Stochastic Oscillator (STOCH), Rate of Change (ROC), Momentum.
- **Volatility Indicators**: Bollinger Bands (BB), Average True Range (ATR).
- **Volume Indicators**: On-Balance Volume (OBV), Volume Weighted Average Price (VWAP), Chaikin Money Flow (CMF).
- **Other Indicators**: Commodity Channel Index (CCI).

After feature engineering, the F1 score of the models improved significantly, in one of the cases from 0.07 to 0.45.

Model Selection and Training

- **Multilayer Perceptron (MLP)**: Used as a baseline to compare performance.
- **XGBoost Classifier**: Chosen for its robustness and ability to handle complex, non-linear patterns in data. We fine-tuned hyperparameters like n_estimators, max_depth, and learning_rate. To address class imbalance, we set the scale_pos_weight parameter in XGBoost to the ratio of negative to positive samples in the training set.
- **Recurrent Neural Network (RNN)**: Used from its predictive power in sequential data, which is useful in modelling stocks and financial data. We fine-tuned how many previous days we wanted to give as a reference to the model to predict the limit up conditions.

Model Evaluation Metrics:

- **Accuracy**: Proportion of correctly predicted instances.
- **F1 Score**: Harmonic mean of precision and recall, suitable for imbalanced datasets.
- **ROC AUC Score**: Measures the ability of the classifier to distinguish between classes.
- **Confusion Matrix**: Provides detailed insight into true positives, false positives, true negatives, and false negatives.

Best MLP Model Performance:

- **Accuracy**: 96.84% **F1 Score**: 0.49 **Recall** 0.00
- The model was biased to the majority class and completely neglected the minority class. This necessitated the need for better models or sampling techniques to handle the class imbalance.

Best XGBoost Model Performance:

- **Accuracy**: 54.82% **F1 Score**: 0.55 **Recall** *0.65* **ROC AUC Score**: *0.6380*
- The model demonstrated high recall for the positive class, crucial for not missing potential limit-up events. Hence, this was our best model. This used scale_pos_weights.

RNN Model Performance:

- **Accuracy**: 90.79% **F1 Score**: 0.56 **Recall** 0.30 **ROC AUC Score**: 0.6224
- The model demonstrated better trends than the MLP, but worse than XGBoost. This was because the recall was lower for the minority class, which is our most important metric.

Sampling Methods for Class Imbalance

Since there was a huge imbalance in our dataset, we used methods like Random Oversampling, Random Undersampling, and SMOTE. However, the results with these models were very similar to the results with our scale_pos_weighted XGBoost model.

Backtesting Strategy and Results

To validate the practical applicability of our model, we implemented a backtesting module simulating trading over a historical period. We did this for our best XGBoost model. It contains:

- **A Custom Data Feeder**: Provided data up to the current date during the simulation, mimicking real-world trading conditions.
- **Portfolio Management**: Managed cash and positions, executed trades based on model predictions.
- **Purchase Strategy**: Used an Equal Allocation Strategy to distribute capital equally among stocks generating buy signals.

Trading Rules:

- **Buy Signals**: Generated when the model predicted a probability above a threshold (e.g., 0.5) for a stock to experience a significant price increase.
- **Sell Strategy**: Positions were held for a fixed period (e.g., 10 days) before being sold, regardless of performance.

Backtesting Results:

- **Initial Capital**: $1,000,000 **Final Balance**: $1,083,993.71
- **Total Return**: 8.40% **Annualized Return**: 230.54%
- **Volatility**: 25.03% **Sharpe Ratio**: 9.21
- **Best Performing Stock**: Encoded ticker 300072 with a profit of $650.88
- **Worst Performing Stock**: Encoded ticker 600070 with a loss of $47.84

The high Sharpe ratio indicates that the strategy yielded high returns relative to its volatility, suggesting strong risk-adjusted performance.

Model Strengths and Limitations

- The XGBoost model effectively handled class imbalance and captured complex patterns leading up to limit-up events.
- Incorporation of technical indicators enhanced the model's ability to predict significant price movements.
- The rarity of limit-up events limited the number of positive samples, potentially affecting model generalizability.
- The backtesting assumed perfect execution and did not account for transaction costs, slippage, or market impact.

Conclusion

Our project demonstrates the potential of using machine learning, specifically XGBoost, with technical analysis indicators, to predict significant price increases in the Chinese stock market. Despite the challenges posed by class imbalance and the rarity of limit-up events, the model showed promising results both in predictive performance and in simulated trading scenarios.
NOTE: for plots and graphs, kindly refer to the outputs of our Jupyter Files (space limitation).