ML Final Report: Predicting Stock Returns and Portfolio Construction using ML and Congressional Stock Transactions

Chung-Mou Pan (NetID: N14124164)

December 21, 2024

Quad Chart

Motivation	**ML Solution and Tools**
- Predicting stock returns is challenging due to market noise. - Improve portfolio decisions and risk management. - Integrate congressional sentiment with a curated set of technical indicators.	- Chose Random Forest for non-linearity and feature importance. - Conducted experiments with both regression and classification tasks. - GridSearchCV + TimeSeriesSplit for robust validation.
Contributions	**Results and Future Directions**
- Achieved Sharpe Ratio > 1.4 using selective technical indicators and congressional signals. - Leveraged ML predictions to rank stocks.	- Regression model: low R^2 ($\sim$0.02) but enabled profitable portfolio construction using mean-variance optimization. - Future: Test XGBoost, reduce technical noise, integrate alternative data.

Table 1: Quad Chart Executive Summary

My Contributions

- Developed a comprehensive feature set initially combining congressional sentiment and a broad range of technical indicators. - Experimented with classification approaches (predicting "up" or "down" movements), achieving around 50% accuracy, indicating the difficulty of directional prediction. - Conducted two key experiments: 1. **With many technical indicators:** Found that a large set of technical indicators tended to overshadow the congressional sentiment signal, making it hard to extract meaningful patterns from sentiment data. 2. **With fewer technical indicators:** By omitting a large portion of technical features and focusing mainly on congressional sentiment signals (with a minimal set of technical features), the model was able to better leverage the unique sentiment data. - Employed TimeSeriesSplit to avoid look-ahead bias. - Despite low R^2 (around 0.02), leveraged ML predictions to rank stocks. - Constructed a portfolio using the Markowitz optimization approach, achieving a high Sharpe ratio (1.4+), demonstrating that even weak signals, if properly isolated, can be turned into strong returns when combined with robust optimization.

1 Introduction

Congressional stock transactions have been a topic of heated debate, with critics arguing that members of Congress may have access to privileged information, potentially offering them an unfair trading advantage. For example, a portfolio that replicates the stock purchases of House Members outperforms the market by 55 basis points per month, equivalent to roughly 6% on an annual basis [1]. Further, high-profile cases, such as Nancy Pelosi's trading activities, have drawn media attention. Under the STOCK Act, these transactions must be disclosed within 30 to 45 days, providing a unique dataset for analysis.

In this project, I leverage congressional stock trading data to predict short-horizon returns and construct a profitable portfolio using Machine Learning methods. While predicting returns directly is challenging due to market noise and complexity, the goal is to uncover small but exploitable signals, especially from congressional sentiment, and then combine them with robust portfolio optimization [2].

2 Data and Preprocessing

Data Sources: - Historical stock prices from *Alpaca API*. - Congressional trading disclosures from *Quiver Quantitative* paid subscription.

Preprocessing: - Aligned and filtered data from 2016-01-01 onwards. - Computed future returns over a 5-day prediction horizon. - Merged congressional sentiment features with standard technical indicators.

3 Feature Engineering

Sentiment Features: Congressional transaction data was transformed into meaningful features to capture sentiment dynamics:

- **Net Sentiment:** Aggregates the net number of buys (+1) and sells (-1).
- **Weighted Sentiment:** Sentiment adjusted by trade size to emphasize the impact of larger trades.
- **Party-specific Sentiment:** Separate sentiment metrics for Democratic and Republican trades.

These features were aggregated daily at the ticker level, resulting in metrics such as:

- 3-day lagged sentiment values (e.g., *net_sentiment_lag3*).
- 30-day rolling means for net and weighted sentiment signals.
- 30-day rolling sums for trade activity, reflecting congressional trading trends.

Technical Indicators (Initial Set): Initially, a broad range of technical indicators was included:

- Momentum (5-day, 20-day), Volatility (20-day rolling standard deviation),
- Moving Averages (20-day), Relative Strength Index (RSI, 14-day),
- MACD, Signal line, Bollinger Bands, and ATR.

Refining the Feature Set: Preliminary experiments showed that an abundance of technical indicators overshadowed the congressional sentiment signals, leading to suboptimal model performance. The feature set was refined by:

- Reducing technical indicators to only **Momentum (5-day, 20-day)** and **Moving Average (20-day)**.
- Emphasizing sentiment features, such as rolling averages, lags, and party-specific sentiment metrics.

Final Features: The final feature set integrated congressional sentiment and select technical indicators:

- Sentiment: *net_sentiment*, *weighted_sentiment*, lagged and rolling versions of sentiment signals.
- Technical: *mom_5d*, *mom_20d*, 20-day volatility (*vol_20d*), and 20-day moving average (*ma_20d*).

Feature Target Alignment: Future returns (*Future_Return*) over a 5-day horizon were calculated and aligned with the feature set.

4 Model Training

The approach and key findings are outlined below:

Data Preparation: The dataset was split into training and testing subsets to avoid look-ahead bias:

- **Cutoff Date:** December 31, 2022.
- **Training Data:** All trades prior to the cutoff date.
- **Testing Data:** Trades on or after the cutoff date.

The target variable, *Future_Return*, represented the 5-day forward return:

$$\text{Future\_Return} = \frac{\text{Price}_{t+5} - \text{Price}_t}{\text{Price}_t}$$

Models Tested:

- **Regression with Random Forest:** The model was trained to predict future returns directly. Hyperparameters such as *n_estimators*, *max_depth*, and *min_samples_split* were tuned using **GridSearchCV**.
- **Classification (Up/Down):** An alternative approach tested whether predicting return direction (up/down) might simplify the problem. However, the accuracy hovered around 50%, indicating no strong directional edge.

Validation: To ensure robustness, a **TimeSeriesSplit** cross-validation strategy was employed. This method preserved the chronological order of data to prevent look-ahead bias and ensured realistic performance estimates.

Hyperparameter Tuning: The Random Forest model was tuned using **GridSearchCV** with the following search grid:

$$\begin{aligned}
\text{n\_estimators: } & [100, 200], \\
\text{max\_depth: } & [5, 10], \\
\text{min\_samples\_leaf: } & [3, 5, 7], \\
\text{max\_features: } & [\text{“sqrt”}, \text{None}].
\end{aligned}$$

Feature Importances: The following table summarizes the feature importances derived from the trained model (Experiment with fewer technical indicators):

Feature	**Importance**
net_sentiment	0.0089
weighted_sentiment	0.0183
net_sentiment_D	0.0187
net_sentiment_R	0.0056
total_trades	0.0068
avg_trade_size	0.0176
mom_5d	0.1973
mom_20d	0.1895
vol_20d	0.1701
ma_20d	0.1076
net_sentiment_lag3	0.0123
weighted_sentiment_lag3	0.0195
net_sentiment_rolling30	0.0404
weighted_sentiment_rolling30	0.0695
total_trades_rolling30	0.0426
net_sentiment_D_rolling30	0.0382
net_sentiment_R_rolling30	0.0372

Table 2: Feature Importances from the Model

Model Performance: Despite a low regression accuracy ($R^2 \sim 0.02$) (0.02 good in the context of stock prediction. Even Hedge Funds struggle to train models with scores higher than 0.5) and classification accuracy close to 50%, the regression model demonstrated a useful ranking capability.

Key Insights:

- Reducing technical indicators allowed congressional sentiment metrics to play a more dominant role in the model, as seen in Table 2.
- While absolute predictions were weak, the model's ranking capability proved valuable for downstream tasks such as portfolio optimization.

5 Portfolio Construction

Approach: The portfolio construction followed a systematic methodology leveraging regression-based predicted returns and Markowitz mean-variance optimization to maximize the Sharpe ratio

[3]. The steps are outlined below:

1. **Predict Returns for the Test Set:** The regression model was applied to the test set to generate predicted returns for all stocks. The predictions were averaged by ticker to produce a mean predicted return per stock. The top-k stocks (where $k = 30$) were selected based on these rankings:

$$\text{Top-k stocks} = \text{Sort}(\text{Mean Predicted Returns}, \text{Descending})$$

2. **Compute Historical Returns and Covariance:** For the selected top-k stocks, historical daily returns were computed using percentage changes in adjusted closing prices. The covariance matrix of returns was then estimated to quantify the risk:

$$\text{Covariance Matrix} = \text{historical\_returns.cov()}$$

3. **Portfolio Optimization:** Using the Markowitz mean-variance framework, portfolio weights were optimized to maximize the Sharpe ratio, subject to the following constraints:

   - Weights sum to 1 (portfolio fully invested).
   - Minimum weight of 2% per stock to enforce diversification [4].
   - Individual weights are bounded between 0% and 100%.

   The Sharpe ratio is defined as:

$$\text{Sharpe Ratio} = \frac{\text{Portfolio Return} - \text{Risk-Free Rate}}{\text{Portfolio Volatility}}$$

   where:

$$\text{Portfolio Return} = w^T \cdot \mu, \quad \text{Portfolio Volatility} = \sqrt{w^T \cdot \Sigma \cdot w}$$

   Here, w represents the weights, μ the expected returns, and Σ the covariance matrix.

4. **Portfolio Summary:** With the optimal weights obtained, the portfolio return, volatility, and Sharpe ratio were computed:

$$\text{Portfolio Return} = w^T \cdot \mu, \quad \text{Portfolio Volatility} = \sqrt{w^T \cdot \Sigma \cdot w}, \quad \text{Sharpe Ratio} = \frac{\text{Portfolio Return}}{\text{Portfolio Volatility}}$$

Results: By carefully reducing technical indicators and relying more on congressional sentiment signals, the constructed portfolio achieved a Sharpe ratio exceeding 1.4 on test set. This result highlights the following:

- Even weak predictive signals, such as those from congressional sentiment, can be effectively leveraged for profitable portfolio strategies.
- The optimization framework successfully balanced return and risk, assigning optimal weights under diversification constraints.
- Figure 1 shows that the optimized portfolio achieves a higher annualized return compared to the S&P 500 benchmark, with annual returns of 18.06% and 11.09%, respectively. This represents a 7% outperformance, surpassing the 6.55% portfolio construction method proposed in [1].

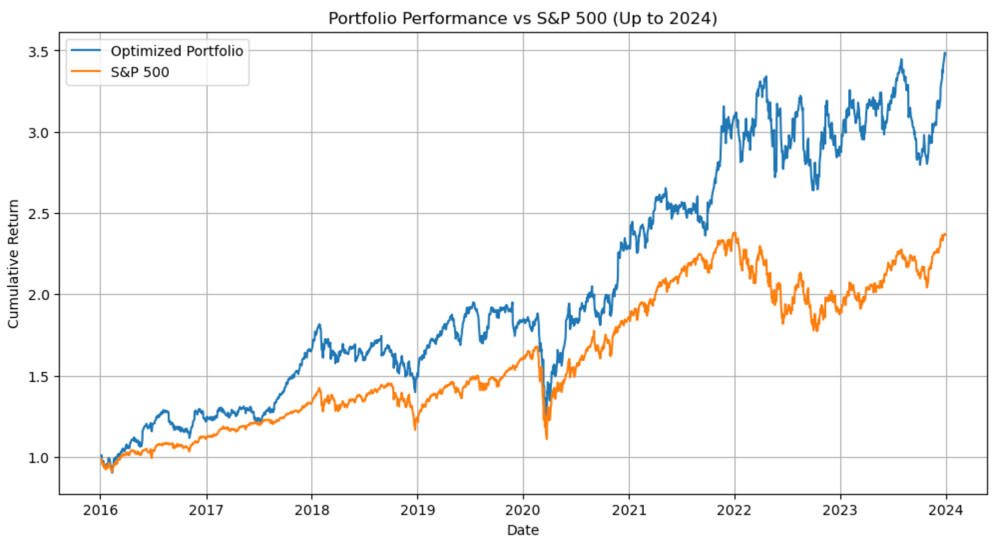

Figure 1: Cumulative returns of the optimized portfolio vs. S&P 500.

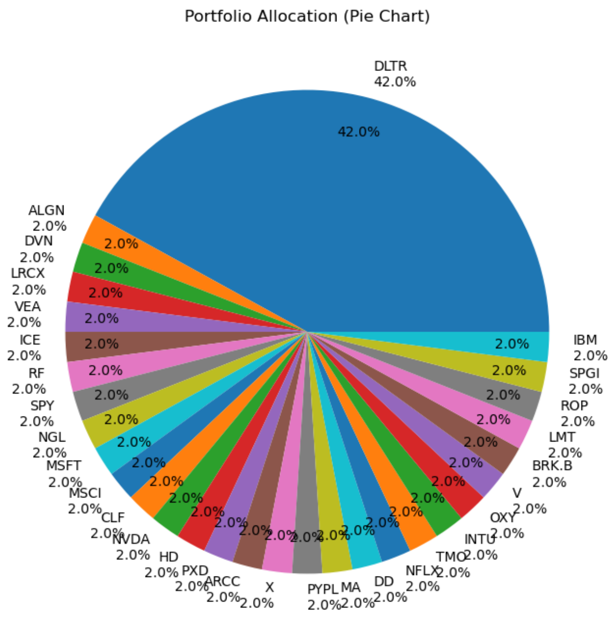

Figure 2: Portfolio Weights

6 Discussion and Future Work

The core lesson is that while the model's direct predictive power on returns or direction might be low, as reflected by the R^2 score—a common challenge even for hedge fund models due to the inherent noise and volatility of financial markets—it can still effectively rank securities. By focusing on a less cluttered feature set—reducing technical noise to highlight congressional sentiment—and then applying robust portfolio optimization, the outcome can be significantly improved.

Future Directions: - Explore neural network or reinforcement learning methods (e.g., PPO). - Integrate more alternative datasets (e.g., macroeconomic indicators, social media sentiment, insider trading information) to complement congressional signals. - Further refine feature selection to ensure congressional sentiment is not overshadowed.

Lessons Learned: - More features is not always better; too many technical indicators can drown out unique sentiment signals. - Even signals with very low predictive power (low R^2 or 50% classification accuracy) can lead to profitable strategies if effectively combined with portfolio optimization. - Rigorous validation methods (TimeSeriesSplit) and careful feature engineering are essential.

7 Code Printout

Below is a snippet of the Python code used in this project. It demonstrates the overall process: data preparation, feature engineering, model training, and portfolio construction. For full code printout, please refer to *project.ipynb* or *full_code_printout.pdf*.

Listing 1: Code Snippet

```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import TimeSeriesSplit, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score

# ... load data, merge sentiment, compute technical indicators...

X = merged[final_feature_cols]  # Reduced set of technical indicators
y = merged[target_col]

tscv = TimeSeriesSplit(n_splits=5)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [5, 10],
    'min_samples_leaf': [3, 5],
    'min_samples_split': [2, 5],
    'max_features': ['sqrt', None],
    'bootstrap': [True, False]
}

model = RandomForestRegressor(random_state=42)
grid_search = GridSearchCV(model, param_grid, scoring='r2', cv=tscv,
    n_jobs=-1)
grid_search.fit(X_scaled, y)

final_model = RandomForestRegressor(**grid_search.best_params_,
    random_state=42)
final_model.fit(X_train_scaled, y_train)
y_pred = final_model.predict(X_test_scaled)
```

```
test_r2 = r2_score(y_test, y_pred)
```

8 Data Availability and Reproducibility

Project File Structure:

```
- congress-trading-all.xlsx
- daily_sentiment.csv
- final_model.pkl
- historical_price_data.csv
- historical_price_data.pkl
- merged_data.csv
- pivoted_close.csv
- project.ipynb
- full_code_printout.pdf
- project_report.pdf
- requirements.txt
```

Data Sources:

- **Congressional Trading Data**: The primary data source *(congress-trading-all.xlsx)* was obtained from Quiver Quantitative's paid subscription platform (https://www.quiverquant.com/). **Note:** To comply with copyright laws, the file *congress-trading-all.xlsx* included in the provided zip archive will contain no data. Users wishing to replicate the results must independently obtain access to the dataset through a valid subscription to Quiver Quantitative.

- **Historical Price Data**: The historical price data (*historical_price_data.csv* and *historical_price_data.pkl*) was downloaded from Alpaca's API. Alpaca provides free access to certain historical market data. Anyone can sign up for an Alpaca account and replicate the data download process using code included in the *project.ipynb* with their obtained API Key. (https://alpaca.markets/)

Derived Files:

- **Merged Dataset** (*merged_data.csv*): Created by joining the congressional trading dataset with Alpaca price data, and incorporating engineered features.

- **Pivoted Close Data** (*pivoted_close.csv*): A pivoted format of historical prices for streamlined feature engineering and model training.

- **Daily Sentiment** (*daily_sentiment.csv*): Aggregated sentiment metrics derived from the congressional trading data on a daily basis.

Model and Code:

- **Trained Model** (*final_model.pkl*): The trained Random Forest model is provided. To replicate training, run the code in *project.ipynb*.

- **Code** (*project.ipynb*): The Jupyter notebook includes code that generates result.

- **Code Printout** (*full_code_printout.pdf*): Full code printout in pdf format.

Reproducibility Steps:

1. **Data Access:** To replicate, obtain the *congress-trading-all.xlsx* file from Quiver Quantitative (with subscription). Next, obtain an API Key from Alpaca and download historical price data using the provided code or adapt it as needed.
2. **Run the Notebook:** With the raw data in place, run *project.ipynb.* This will:
   - Load and merge the congressional and price data.
   - Generate features and sentiment metrics.
   - Train the Random Forest model.
   - Produce predictions and perform portfolio optimization and visualization.
3. **Environment:** Install Python package dependencies via *requirements.txt.* No special proprietary software is needed, aside from the data subscription.

9 References

References

[1] A. J. Ziobrowski, J. W. Boyd, P. Cheng, and B. J. Ziobrowski, "Abnormal Returns from the Common Stock Investments of Members of the U.S. House of Representatives," *Business and Politics*, vol. 13, no. 1, pp. 1-23, 2011.

[2] S. Gu, B. Kelly, and D. Xiu, "Empirical Asset Pricing via Machine Learning," *The Review of Financial Studies*, vol. 33, no. 5, pp. 2223-2273, 2020.

[3] H. Markowitz, "Portfolio Selection," *The Journal of Finance*, vol. 7, no. 1, pp. 77-91, 1952.

[4] R. Jagannathan and T. Ma, "Risk Reduction in Large Portfolios: A Role for Portfolio Weight Constraints," *The Journal of Finance*, vol. 58, no. 4, pp. 1651-1683, 2003.