Following The Smart Money: Applying Machine Learning on Insider & Congressional Trades

Trabajo De Grado

Sergio Alvarado Cediel

Bogotá - Colombia
2025

Following The Smart Money: Applying Machine Learning on Insider & Congressional Trades

Trabajo De Grado

Sergio Alvarado Cediel

Administración de Negocios Internacionales

Ignacio Anguita Espadaler

Escuela de Administración

Universidad del Rosario

Bogotá - Colombia
2025

Declaración de originalidad y autonomía

Declaro bajo la gravedad del juramento, que he escrito el documento de título "Following The Smart Money: Applying Machine Learning on Insider & Congressional Trades", en la opción de grado de convenio de doble titulación con EADA Business School y que, por lo tanto, su contenido es original.

Declaro que he indicado clara y precisamente todas las fuentes directas e indirectas de información y que este trabajo únicamente ha sido entregado únicamente a EADA Business School y la Universidad del Rosario con fines de calificación o publicación.

Sergio Fernando Alvarado Cediel

Declaración de exoneración de responsabilidad

Declaro que la responsabilidad intelectual del presente trabajo es exclusivamente de su autor. La Universidad del Rosario no se hace responsable de contenidos, opiniones o ideologías expresadas total o parcialmente en él.

Sergio Fernando Alvarado Cediel

Tabla de Contenido

Glossary

Insider Trading: The buying or selling of securities by corporate executives, directors, or employees who have access to material, non-public information about their company.

Congressional Trading: Financial transactions executed by members of the U.S. Congress, who may possess privileged information related to legislation, regulation, or public policy.

Disclosure Time Lag: The time delay between the execution of a trade and the moment it becomes publicly disclosed, which is a key factor in predictive trading strategies.

Machine Learning: A branch of artificial intelligence that enables systems to learn patterns from historical data and make predictions without explicit programming.

Supervised Learning: A type of machine learning in which models are trained using labeled data, consisting of input features and known output labels.

Binary Classification: A supervised learning task in which observations are classified into one of two possible categories, such as profit or loss.

Random Forest: An ensemble machine learning algorithm that builds multiple decision trees independently and aggregates their predictions to improve accuracy and robustness.

XGBoost (Extreme Gradient Boosting): An advanced gradient boosting algorithm that builds decision trees sequentially, correcting previous errors while incorporating regularization to prevent overfitting.

Gradient Boosting: A machine learning technique that constructs models iteratively, where each new model attempts to correct the errors of the previous one.

Overfitting: A modeling error that occurs when a model learns the training data too closely, including noise, resulting in poor generalization to new data.

Underfitting: A condition in which a model is too simple to capture the underlying patterns in the data, leading to poor performance on both training and test sets.

Feature Engineering: The process of creating, transforming, or selecting relevant variables to improve a model's predictive performance.

Backtesting: The evaluation of an investment strategy using historical data to assess its potential performance before real-world implementation.

Sharpe Ratio: A risk-adjusted performance metric that measures excess return per unit of total risk, compared to a risk-free asset.

Portfolio Optimization: The process of allocating asset weights in a portfolio to maximize expected returns for a given level of risk.

Modern Portfolio Theory (MPT): A financial theory introduced by Harry Markowitz that emphasizes diversification to optimize the risk-return tradeoff.

Efficient Frontier: A curve representing the set of optimal portfolios that provide the highest expected return for each level of risk.

Accuracy: A performance metric that measures the proportion of correct predictions made by a classification model.

Precision: The proportion of true positive predictions among all positive predictions made by the model.

Recall: The proportion of actual positive cases that are correctly identified by the model.

F2 Score: A composite metric that combines precision and recall while placing greater emphasis on recall, prioritizing the detection of positive cases.

ROC Curve (Receiver Operating Characteristic Curve): A graphical representation showing the relationship between the true positive rate and false positive rate across different classification thresholds.

AUC (Area Under the Curve): A scalar value representing the area under the ROC curve, indicating the model's ability to discriminate between classes.

Hyperparameters: Configuration variables set prior to model training that control the learning process and model complexity.

Optuna: An automated hyperparameter optimization framework used to identify optimal model configurations through iterative trials.

Regularization: A technique used to penalize model complexity in order to reduce overfitting and improve generalization.

Trade Size: The monetary value or number of shares involved in a transaction, often interpreted as a signal of trader confidence.

Sector Sensitivity: The degree to which specific economic sectors respond to insider or political trading activity.

Predictive Signal: A statistically significant pattern or indicator that suggests a potential future movement in asset prices.

Risk-Adjusted Returns: Investment returns evaluated relative to the level of risk assumed, rather than in absolute terms.

Abstract and Key Words

This project explores the development of investment strategies based on insider and congressional trading data, leveraging historical transaction disclosures to identify profitable market and investment signals. Conducted in collaboration with B-Quant, a financial education and analytics company, the project aims to transform privileged trading behavior, such as trades by corporate insiders and U.S. Congress members, into actionable investment insights. The core objective is to extract statistically significant patterns from these datasets, account for the disclosure time lag, and design empirical strategies that optimize both profitability and risk-adjusted returns.

The analysis is grounded in two proprietary datasets provided by B-Quant: the Insider Trading database, containing detailed information on trades made by corporate executives and insiders, and the Congressional Trading database, which includes transaction records from U.S. congressmen and women. Through rigorous data preprocessing, exploratory analysis, and the application of statistical and machine learning methods, the project identifies recurrent behaviors, such as timing advantages, individual track records, trade sizes, and sector-level preferences, that correlate with subsequent market movements. Based on these insights, a series of quantitative trading strategies were developed, incorporating rules for trade selection, entry and exit timing, and risk management. These strategies were evaluated through backtesting using historical market data, assessing performance across key metrics including returns, Sharpe ratios, and drawdowns.

Key Words: Insider Trading, Congressional Trading, Disclosure Time Lag, Machine Learning, Supervised Learning, Binary Classification, Random Forest, XGBoost (Extreme Gradient Boosting), Gradient Boosting, Overfitting, Underfitting, Feature Engineering, Backtesting, Sharpe Ratio, Portfolio Optimization, Modern Portfolio Theory (MPT), Efficient Frontier, Accuracy, Precision, Recall, F2 Score, ROC Curve (Receiver Operating Characteristic Curve), AUC (Area Under the Curve), Hyperparameters, Optuna, Regularization, Trade Size, Sector Sensitivity, Predictive Signal, Risk-Adjusted Returns.

Resumen y palabras clave

Este proyecto explora el desarrollo de estrategias de inversión basadas en datos sobre operaciones bursátiles de personas con información privilegiada y miembros del Congreso, aprovechando la información histórica sobre transacciones para identificar señales rentables del mercado y de inversión. Realizado en colaboración con B-Quant, una empresa de educación y análisis financiero, el proyecto tiene como objetivo transformar el comportamiento privilegiado en las operaciones bursátiles, como las realizadas por personas con información privilegiada de empresas y miembros del Congreso de los Estados Unidos, en conocimientos de inversión aplicables. El objetivo principal es extraer patrones estadísticamente significativos de estos conjuntos de datos, tener en cuenta el desfase temporal en la divulgación y diseñar estrategias empíricas que optimicen tanto la rentabilidad como los rendimientos ajustados al riesgo.

El análisis se basa en dos conjuntos de datos propios proporcionados por B-Quant: la base de datos Insider Trading, que contiene información detallada sobre las operaciones realizadas por ejecutivos y personas con información privilegiada de empresas, y la base de datos Congressional Trading, que incluye registros de transacciones de congresistas estadounidenses. Mediante un riguroso preprocesamiento de datos, análisis exploratorios y la aplicación de métodos estadísticos y de aprendizaje automático, el proyecto identifica comportamientos recurrentes, como ventajas temporales, historiales individuales, volúmenes de operaciones y preferencias a nivel sectorial, que se correlacionan con los movimientos posteriores del mercado. A partir de estas conclusiones, se desarrollaron una serie de estrategias de negociación cuantitativas que incorporaban reglas para la selección de operaciones, el momento de entrada y salida, y la gestión de riesgos. Estas estrategias se evaluaron mediante

backtesting utilizando datos históricos del mercado, valorando el rendimiento en función de métricas clave como la rentabilidad, los ratios de Sharpe y las caídas.

Palabras Clave: Insider Trading, Operaciones del Congreso, Retraso en la Divulgación, Aprendizaje Automático, Aprendizaje Supervisado, Clasificación Binaria, Bosque Aleatorio, XGBoost (Extreme Gradient Boosting), Potenciación por Gradiente, Sobreajuste, Subajuste, Ingeniería de Características, Backtesting, Ratio de Sharpe, Optimización de Portafolio, Teoría Moderna de Portafolio (MPT), Frontera Eficiente, Exactitud, Precisión, Recall, Puntaje F2, Curva ROC (Receiver Operating Characteristic), AUC (Área Bajo la Curva), Hiperparámetros, Optuna, Regularización, Tamaño de la Operación, Sensibilidad Sectorial, Señal Predictiva, Rendimientos Ajustados por Riesgo.

Following The Smart Money: Applying Machine Learning on Insider & Congressional Trades - Introduction and context

Description of the company

Company: BQUANT

Objective: Develop investment strategies based on insider trading and congressional trading data. The project will focus on extracting valuable insights from these datasets, accounting for the time lag between transaction execution and public disclosure. The goal is to design and test strategies that leverage this information effectively, optimizing their profitability and risk-adjusted returns.

B-Quant is a company constituted by a group of professionals with experience in the financial sector and financial markets. Formed by teachers of vocation, who have passed through various universities and training centers, the company seeks to offer the knowledge of those who founded it to a wider audience.

In the B-Quant program, the company is committed to offer its clients the tools and knowledge developed during years of professional and teaching experience. The objective is that its users acquire the necessary skills to develop successfully in the professional world and in the analysis of financial markets. The courses offered by B-Quant are designed for students with different levels of skill and knowledge, from beginners to more experienced profiles.

Aim of the Project

This project aims to develop a statistical-mathematical model that predicts when to buy and sell stocks in the financial market. The project is carried out in collaboration with B-Quant Finance, the company which has provided two central databases to build the statistical model,

one which is called “insider database” and the other “congressional data”. An analysis of these datasets will be developed, identifying specific investing patterns and trends, and developing investment strategies based on these two-trading data. The project will focus on extracting valuable insights from these datasets, accounting for the time lag between transaction execution and public disclosure. The goal is to design and test strategies that leverage this information effectively, optimizing their profitability and risk-adjusted returns.

Scope of the Project

The scope of the project consists in analyzing insider and congressional trading data for investment strategy development, identifying specific investing patterns or trends, and develop a series of investing strategies suggestions based on the insights drawn from the datasets.

Included tasks (In-Scope):

1. Data Acquisition and Preprocessing

Data acquisition and preprocessing step involves gathering the datasets, in this case, "insider database" and "congressional data", and transforming them into a format suitable for analysis. Data cleaning ensures the removal of inconsistencies and missing values, while preprocessing aligns the datasets in terms of structure and timing, laying the foundation for accurate insights.

- Gather and import the "insider database" and "congressional data" into a suitable analytical environment, in this case, a CSV format that can be read by Python with Pandas to then process its data and analyze the information.
- Clean and preprocess the data:
    - Handle missing values. (deletion or imputation)

    - Standardize data formats.
    - Ensure data consistency.
    - Handle the time lag between transaction execution and public disclosure.
- Merge or relate the datasets as needed for comprehensive analysis.

2. Exploratory Data Analysis:

Once the data is cleaned and structured, exploratory data analysis helps obtaining initial insights. This involves statistical and visual techniques to identify patterns, trends, and anomalies in trading behavior. This data analysis provides a clearer understanding of how insiders and congressional members interact with the stock market, highlighting activity levels, sector preferences, and possible responses to public news.

- Perform statistical analysis to identify trends and patterns in insider and congressional trading activity.
- Visualize data to reveal insights into trading behavior (e.g., volume, frequency, sector preferences).
- Analyze the timing of trades relative to market events and public disclosures.
- Identify the most active traders, and the most traded sectors.

3. Pattern and Trend Identification:

This step focuses on identifying and quantifying trading behaviors that could indicate predictive potential. Analytical techniques are used to find important relationships between

trading actions and subsequent stock movements, potentially uncovering possible patterns in the data.

- Develop algorithms or methods to identify statistically significant trading patterns.
- Analyze the correlation between insider/congressional trades and subsequent stock price movements.
- Identify recurring trading behaviors and potential predictive signals.
- Analyze the trend of the trades, if they are usually buys or sells.

4. Investment Strategy Development:

With patterns and trends identified, the next step is to translate these insights into empirical investment strategies. This involves defining entry and exit criteria, handling the disclosure delay, and managing portfolio risks.

- Design quantitative investment strategies based on the identified patterns and trends.
- Incorporate the time lag between transaction execution and public disclosure into the strategy design.
- Develop rules for entry and exit points, position sizing, and risk management.
- Create different strategies, and compare the results.

5. Backtesting and Performance Evaluation:

To validate the efficacy and success of the proposed strategies, backtesting is performed using historical data. This helps evaluate potential profitability in the future, assess risk metrics, and test strength under different market conditions.

- Implement the developed strategies using historical data.
- Backtest the strategies to evaluate their performance (e.g., profitability, risk-adjusted returns, drawdown).
- Analyze the robustness and sensitivity of the strategies to different market conditions.
- Provide statistical analysis of the performance of the strategies.

6. Documentation and Reporting:

This final step involves recording all analytical procedures, results, and strategic conclusions, providing credibility and rigor to the entire research process.

- Document the data analysis process, methodology, and findings.
- Prepare a comprehensive report summarizing the results of the backtesting and performance evaluation.
- Present the investment strategies and their potential implications.[1]

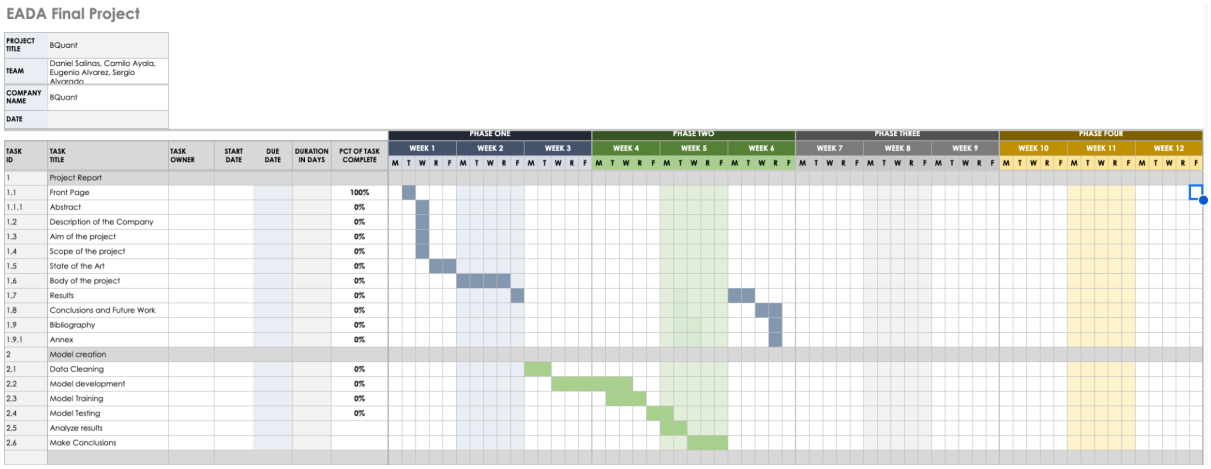
Excluded tasks (Out-of-Scope)

1. Real-Time Trading Implementation:
    - This project focuses on developing and backtesting strategies, not on deploying them for live trading.
    - Automated real-time execution of the strategies is outside the scope.
2. Legal and Regulatory Compliance:
    - While awareness of regulatory aspects is essential, this project will not provide legal or compliance advice.
    - The project will not include the research of the legal ramifications of insider trading.
3. Fundamental Analysis of the financial statements:
    - The project focuses on analyzing trading data, not on performing an exhaustive fundamental analysis of the companies involved.
    - Analysis of company earnings reports, or balance sheets.
4. External Data Integration Beyond Provided Datasets:
    - The analysis will be limited to the provided "insider database" and "congressional data."
    - Integration of opinion news, social media data, or other external data sources is excluded.
5. Development of a User Interface or Software Application:
    - The project will produce analysis and reports, not a software application for end-users.
    - There will not be the creation of a website, or mobile application.

- Methodology (gantt chart)

Figure 1

Gantt Chart EADA



EADA Final Project

PROJECT TITLE	BQuant
TEAM	Daniel Salinas, Camilo Ayala, Eugenio Alvarez, Sergio Alvarado
COMPANY NAME	BQuant
DATE	

TASK ID	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION IN DAYS	PCT OF TASK COMPLETE
1	Project Report					
1.1	Front Page					100%
1.1.1	Abstract					0%
1.2	Description of the Company					0%
1.3	Aim of the project					0%
1.4	Scope of the project					0%
1.5	State of the Art					0%
1.6	Body of the project					0%
1.7	Results					0%
1.8	Conclusions and Future Work					0%
1.9	Bibliography					0%
1.9.1	Annex					0%
2	Model creation					
2.1	Data Cleaning					0%
2.2	Model development					0%
2.3	Model Training					0%
2.4	Model Testing					0%
2.5	Analyze results					
2.6	Make Conclusions					

PHASE ONE: WEEK 1, WEEK 2, WEEK 3
PHASE TWO: WEEK 4, WEEK 5, WEEK 6
PHASE THREE: WEEK 7, WEEK 8, WEEK 9
PHASE FOUR: WEEK 10, WEEK 11, WEEK 12

Source: own elaboration

State of the Art

1. Introduction to Insider & Congressional Trading

Insider trading refers to the buying or selling of a company's stocks by individuals who have access to non-public, material information (often referred as privileged information) about the company. Typically, these individuals are corporate executives, board members, or key employees who hold access to sensitive information that could significantly impact stock prices once disclosed, for example, a CFO who perfectly knows the P&L of the company and decides to buy puts since he knows the company is not doing well.

Insider trading is legal when appropriately disclosed and filed with the Securities and Exchange Commission (SEC), for example, a CEO of a publicly traded company can purchase 10,000 shares of his own company's stock because he believes in its long-term growth prospects, however, he needs to submit the required Form 4 to the U.S. Securities and Exchange

Commission (SEC) within two business days of the transaction, as it's required by law. The trade is then made public through the SEC's EDGAR database, allowing investors to see that an insider has bought shares.

Insider trading becomes illegal when trades are based on confidential information that is not yet public and are not reported in accordance with regulatory requirements.

On the other side congressional trading refers to the buying or selling of stocks and other securities by members of the U.S. Congress. Due to their job, legislators often have access to sensitive, non-public information related to upcoming legislation, regulatory changes, or national security matters that could significantly influence financial markets. For example, a senator who is privately briefed about the likelihood of upcoming defense spending increases might purchase shares of military contractors like Lockheed Martin or Raytheon before the legislation is publicly announced, anticipating that the companies' stock prices will rise once the bill is passed.

While members of Congress are legally allowed to trade securities, they are required to disclose their transactions under the STOCK Act (Stop Trading on Congressional Knowledge Act) of 2012. This law mandates that any trade made by a member of Congress or their immediate family must be reported within 45 days of the transaction. These disclosures are publicly available and aim to promote transparency and prevent the misuse of privileged information.

However, congressional trading becomes controversial or potentially illegal when lawmakers use non-public information for personal financial gain before that information is

available to the public, especially if the disclosure is delayed or omitted. Although enforcement has been limited, such behavior raises ethical concerns and questions about fairness in the market.

2. Theoretical background

Current Techniques in Predictive Modeling

Traditional statistical models provide the foundation for creating an analysis that identifies behavioral patterns to forecast investment behavior. One of the most widely used techniques today is logistic regression and multiple regression models. The logistic regression models are useful for binary classification tasks, such as predicting whether an insider will sell or buy stocks based on historical data. Multiple regression is a statistical technique that models the relationship between a dependent variable (y) and multiple independent variables (x). This method estimates the probability of corporate insiders purchasing or selling stocks based on historical data, allowing them to identify trends that can help predict future behaviors.

Machine Learning and Deep Learning

Machine Learning is a branch of artificial intelligence that allows machines to learn automatically from data. Instead of following specific instructions, a machine learning system analyzes patterns and relationships within databases and uses the acquired knowledge to make better and more accurate predictions and provide more solid decision-making processes.

The training datasets are an important part of machine learning because the better the data is, the better it can learn. The model is the math that learns from the data. The training algorithm is also very important because it helps the model make fewer mistakes and be more

accurate. After training, the model should be able to make accurate predictions about data it has never seen before.

Currently, various models use artificial intelligence and employ more sophisticated techniques to analyze financial data, which can improve accuracy. One of the most effective machine learning models used for stock market predictions is Random Forest and XGBoost. Both models use tree based algorithms to identify patterns within datasets. These models are particularly useful for predicting buy and sell decisions based on historical trading activity and insider transactions.

The Random Forest forecast is useful for predicting stock movements based on insider trading data, as it shows the relationship between variables. XGBoost is a method used to improve the Random Forest by using gradient boosting, where trees are trained sequentially, each correcting the mistakes of the previous one. XGBoost is used in stock market forecasting since it has high accuracy, speed, and the ability to handle missing data.

Supervised Learning

Supervised learning is a branch of machine learning in which a labeled database is used to train the model, which means that during the training process, the model receives both the inputs (features) and the expected outputs (labels) during the training process. This enables the model to discover a relationship between the two inputs. Predicting a continuous value from a set of independent variables is the primary goal of supervised regression models, such as multiple regression or linear regression.

In supervised regression models, such as linear regression, the model learns by adjusting a mathematical function that minimizes the error between the predictions and the actual observed values. In order to minimize the cost function during training, the dependent and independent variables are the parameters that are adjusted. As a result, the model can make predictions on previously unseen data and generalize its knowledge. It is important that the data must have a distribution that is comparable to the training set in order for the model to function correctly. Furthermore, if the model fits the data too closely, it may overfit and lose its ability to generalize.

Overfitting occurs when a model learns the training data too well, including the noise, not important information, or irrelevant particularities. This indicates that the model loses its capacity to generalize to new and unseen data because it fits the training data too closely. On the other hand, underfitting occurs when a model does not learn enough from the training data. Because of this, it is unable to identify the underlying relationships or structure in the data. As a result, it has high error on both the training set and the test set. Some causes of underfitting include the model being too simple, using too few relevant features, too little training time, and data that is too complex and does not match the model's needs. It is important in machine learning to find a proper balance between both extremes, overfitting and underfitting, in order to achieve a model that generalizes adequately.

Machine learning models

In order to develop the predictive investment strategy, it is very important to select which machine learning model to use. Some relevant things to take into consideration are balance accuracy, robustness, and the ability to generalize from historical data. In this project,

we looked at tree-based methods like: Random Forest and XGBoost. Both are commonly used in classification tasks and are known for their effectiveness in capturing non-linear relationships in structured data.

Random Forest

Random Forest is a method that creates many decision trees, not just one. Each tree looks at a random part of the data and uses different variables[1]. Once all the trees make their predictions, the algorithm combines them to decide the final result. This helps reduce mistakes and prevents the model from focusing too much on just one part of the data. In this project, Random Forest helped our Machine Learning model to find patterns between features like trade size, who made the trade, and how many days passed before the trade became public. However, one of its downsides is that it treats all trees the same and doesn't try to improve over time.

Random Forest is an ensemble of T decision trees trained independently, and it combines their predictions through averaging (for regression) or majority vote (for classification).

The prediction for an input x is the average of all tree outputs:

$$\hat{y}(x) = \frac{1}{T}\sum_{t=1}^{T} f_t(x) \qquad [1]$$

Each tree gives a class prediction. The final class is the majority vote:

$$\hat{y}(x) = \text{mode}\left(f_1(x), f_2(x), \ldots, f_T(x)\right) \quad [2]$$

where:

- fx(t) is the prediction from tree t.

Each tree is trained on a bootstrap sample (random sample with replacement) from the training data.

Each tree grows by splitting nodes to minimize impurity:

For Classification (Minimize Gini or Entropy):

$$\text{Gini}(D) = 1 - \sum_{k=1}^{K} p_k^2 \quad [3]$$

where:

- pk: fraction of class k samples in the node.

Classification: majority vote

$$\hat{y}(x) = \arg\max_y \sum_{t=1}^{T} 1\{f_t(x) = y\} \quad [4]$$

XG Boost

XGBoost stands for Extreme Gradient Boosting. This is a machine learning library commonly used in stock market forecasting, and well known for its speed, efficiency, and good performance in handling large datasets and ability to handle missing data. It also builds decision trees[2], but in a smarter way. Each tree is trained one after another, and every new tree learns

from the mistakes made by the previous ones. This step-by-step learning makes XGBoost very powerful, and hence the decision for selecting it in this study. It also includes features to avoid overfitting, which means it can make good predictions even when the data is noisy or limited.

Logistic regression and Random forest were two models previously tested than XG Boost, and both of them showed an overall lower performance in terms of accuracy, precision, recall and F2 Score. With accuracy percentages of 57% and 68% respectively ordered, XG Boost surpassed these results with a total accuracy percentage of 71%. Because it performed better than the other models in terms of accuracy and control, XGBoost was selected as the main algorithm for this project.

The mathematical model of XGBoost is based on gradient boosting decision trees, and it optimizes an objective function by adding new trees that correct the errors of previous trees.

In order to minimize a total objective function made up of a regularization term (to avoid overfitting) and a loss term (the distance between predictions and true values), it builds a new tree at each step. To effectively optimize each tree, XGBoost calculates gradients and Hessians (first and second derivatives of the loss) using a second-order Taylor expansion. Each tree divides the data into leaves, gives each leaf a score, and uses the gradient and Hessian sums to get the best score for each leaf. The total of all the decision trees outputs determines the final forecast, and regularization makes sure the model stays straightforward and has good generalization.

Predict an output yi, for each input xi, by adding decision trees step by step;

$$\hat{y}=\sum_{t=1}^{T} \quad f_t(x_i) \qquad [5]$$

Minimize the following equation:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) \quad [6]$$

where:

- l: loss function
- Ω(ft): regularization term to control overfitting.
- T: number of leaves in the tree.
- wj: the score of leaf j.

Approximate the loss using Taylor expansion:

$$\mathcal{L}^{(t)} \approx \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t(x_i)^2 \right] + \Omega(f_t) \quad [7]$$

where:

- gi: first derivative, gradient.
- hi: second derivative, hessian.

Each tree splits inputs into leaves j, and assigns a leaf score j:

- For leaf j, define:

$$G_j = \sum_{i \in I_j} g_i, \quad H_j = \sum_{i \in I_j} h_i \quad [8]$$

- Optimal score for that leaf:

$$w_j^* = -\frac{G_j}{H_j + \lambda} \quad [9]$$

- Contribution to the objective:

$$\mathcal{L}^{(t)} = -\frac{1}{2} \sum_j \frac{G_j^2}{H_j + \lambda} + \gamma T \quad [10]$$

- After training T number of trees:

$$\hat{y}_i = \sum_{t=1}^{T} f_t(x_i) \quad [11]$$

This constitutes the final prediction of the model.

Logistic Regression

Logistic Regression is one of the most commonly used models for binary classification tasks. Despite its name, it is not used for predicting continuous values, but rather for estimating the probability that an input belongs to one of two classes. In this project, it was tested to predict whether a trade would be profitable or not, based on features such as the trader's identity, the sector, or the time between purchase and disclosure.

It works by combining all input variables through a linear equation and then passing the result through a sigmoid function, which transforms it into a value between 0 and 1. If the output is above 0.5, the model predicts the positive class for example "successful trade"; otherwise, it predicts the negative class or non-successful trade. The core function used is:

$$p(x) = \frac{1}{1 + e^{-z}}, \quad \text{where } z = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n \quad [12]$$

The model is trained using binary cross-entropy loss, which evaluates how close the predictions are to the actual labels:

$$L = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)] \quad [13]$$

While logistic regression is efficient and easy to interpret, it assumes a linear relationship between variables and the output, which makes it less effective in capturing complex patterns.

Performance Metrics

Accuracy, precision, recall, and F2 score are all performance metrics used to evaluate the effectiveness of a classification model. Accuracy measures the overall correctness of the model's predictions. Precision focuses on the correctness of positive predictions, while recall measures the model's ability to find all positive instances. The F2 score emphasizes recall over precision, giving it more weight in the overall evaluation.

Accuracy: Determines the percentage of cases that are correctly classified out of all instances. It is a broad indicator of how often the model is right.

$$\boldsymbol{Accuracy} = TP + TN \,/\, TP + TN + FP + FN \quad [14]$$

where:

- TP: True positives, correct predictions of positive class.
- TN: True negatives, correct predictions of negative class.
- FP: False positives, incorrect predictions of positive class.
- FN: False negatives, incorrect predictions of negative class.

Precision: Determines the percentage of actual positive forecasts among all positive predictions. "Of all the items the model identified as positive, how many were actually positive?" is the question it addresses.

$$Precision = TP \,/\, TP \, + \, FP \qquad [15]$$

where:

- TP: True positives, correct predictions of positive class.
- FP: False positives, incorrect predictions of positive class.

Recall: Determines the percentage of real positive occurrences that are accurate positive predictions. In other words, "Of all the actual positive instances, how many did the model correctly identify?".

$$Recall = TP \,/\, TP \, + \, FN \qquad [16]$$

where:

- TP: True positives, correct predictions of positive class.
- FN: False negatives, incorrect predictions of negative class.

F2 Score: A metric that combines recall and precision, but gives recall a higher weight. When reducing false negatives is more crucial than reducing false positives, the F2 score can be helpful. For instance, identifying every possible case of a disease (high recall) may be more important in medical diagnosis, even if doing so results in some healthy people being mistakenly labeled (poor precision).

$$F2\ Score = 5 \cdot ((Precision \cdot Recall)/(4 * Precision) + Recall) \quad [17]$$

Portfolio management

When managing a portfolio, the main goal is to have a perfect balance between risk and return. This can be done by combining in the most optimal way the amount invested in stocks, bonds, or other investments. When a portfolio is optimized, it gives you the best possible outcome for the amount of risk you're comfortable with.

According to Modern Portfolio Theory, introduced by Harry Markowitz in 1953[3], one of the smartest things you can do is diversify. That means to spread your investments across different assets that don't move together in the same direction. For example, investing in very different industries so that when one is affected, it doesn't drag the other one down. By doing this, you lower the overall risk without necessarily sacrificing potential returns.

One concept often heard when talking about portfolio management is the efficient frontier. This is like a curve that shows the best return you can get for a given level of risk. If your portfolio sits on that curve, you're making the most efficient use of your money, getting the highest possible return without taking on extra risk.

To understand how the portfolio is performing, you'll want to track a few key metrics:

- Return: This tells how much money has been gained or lost over a period of time.
- Return and Risk Matrix: A chart that helps compare expected returns and volatility of different asset combinations.

- Variance: This measures how much return fluctuates from the average. The higher the variance, the riskier the asset.
- Covariance: This tells whether two assets tend to move together or in opposite directions.
- Correlation: A simplified version of covariance, ranging from -1 to +1. If two assets have a correlation of 1, they move in sync, if it's -1, they move in completely opposite ways.
- Standard Deviation: This is a commonly used risk metric that tells how much the returns tend to deviate from the average.
- Sharpe Ratio: This helps evaluate how much extra return is being earned for the risk being taken, compared to a risk-free investment.

With this in mind once the model is developed, we will try to create the most optimal portfolio by maximizing sharpe ratio, so we don't only stick with which stocks to buy or not but know the ideal amount of money to allocate for each stock and maximize our gap between the risk taken and the possible return.

Historical Context and High-Profile Cases

Insider and congressional trading have played controversial roles in financial markets, with several high profile cases drawing public attention and regulatory scrutiny. Regarding insider trading it has cases from the range of celebrities like Martha Stewart, who sold shares of ImClone Systems in 2001[4] after receiving a tip from her broker about an impending FDA

decision resulting in her making great financial returns, to prominent company founders like Raj Rajaratnam, founder of the Galleon Group hedge fund[5], who in 2011 was convicted of orchestrating one of the largest insider trading schemes in history, using insider tips to make tens of millions of dollars in profits.

On the congressional side, there are also a lot of historic events that have happened. For example, during the pandemic in early 2020 a major controversy emerged where several U.S. senators, including Richard Burr and Kelly Loeffler, were reported to have sold big amounts of stock shortly after attending private briefings about the severity of the virus[6], well before the public fully understood its economic implications. Although formal charges were not brought, the incident provoked criticism and social disapproval, making people call for stricter regulations and enforcement regarding stock exchanges.

These are just a couple of examples of public cases that not only have concerned society and raised ethical and legal questions, it have also sparked interest among investors and researchers who see the potential of predictive power using insider information.

One of the most important research studies regarding this topic is a paper by Lakonishok and Lee called Are Insider Trades Informative? (2001)[7]. They showed that insider purchases, particularly in small-cap firms, tended to give abnormal positive returns. Similarly, in 1998, Nejat Seyhun found in his book Investment Intelligence from Insider Trading[8] that when you look at the buying and selling activities of a group of company insiders, their trades often go against what most other investors do or think that will happen. And surprisingly, the insiders end up being right more often.

Moving into congressional trading, this segment has also gained some momentum, particularly with the increased availability of data. Today, more than ever, access to data is easier, and studies have shown that whenever certain politicians or parties consistently outperform the market, there is some coincidence with key legislative developments, like it is suggested in the paper Capitol Losses: The Mediocre Performance of Congressional Stock Portfolios.[9] The emergence of retail platforms like QuiverQuant and Unusual Whales has helped democratize access to this data, enabling both retail and institutional investors to build strategies around political trades.

Publications from *The Economist* and *Bloomberg* have also highlighted examples of political trades that seemed to happen just before major events, like buying pharma stocks before public health news or defense stocks ahead of military spending increases. Some platforms, like QuiverQuant, even track and mimic these trades in portfolios that, in backtests, have sometimes outperformed the S&P 500.

With all this information, we can assume that people like company insiders and members of Congress often seem to make better-than-average trades. This challenges the idea that markets are always efficient. According to the Efficient Market Hypothesis (EMH)[10], stock prices already reflect all public information, so it should be very hard to consistently beat the market. But when insiders or politicians keep making trades that perform well, it suggests they might have access to information that the general public doesn't or that markets don't always react as efficiently as we expect.

These kinds of trades can move the market by affecting how investors feel, how much a stock is traded, and how its price moves. When insiders buy, people often see it as a good sign, and the stock goes up. Insider sales are harder to interpret, since it can just be a CEO wanting to have more personal liquidity to buy a house or something else personal, but they

can create negative reactions if they seem like a signal of bad news since it can be interpreted as they don't trust their company enough or see some underperforming number. On the other hand, political trades tied to upcoming laws or policies can move entire sectors.

From an investment perspective, these behaviors suggest that there are a lot of opportunities to gain valuable information from certain signals. However, the strength or accuracy of these signals depends on several variables. Here are some:

- **Timing:** Trades disclosed quickly after being made tend to perform better.
- **Trade size:** Bigger trades often show more confidence and have stronger predictive power.
- **Sector sensitivity:** Some industries react more strongly to these trades like tech, healthcare, or defense.
- **Who is trading:** Not all insiders or politicians are equally effective; some demonstrate a repeated ability to time markets well.

Once we understand these factors, we can start to build rules for investing based on them. By analyzing who traded, when, how much, and in which sectors, we can identify strong signals to use in a systematic investment strategy.

Investment strategy

After analyzing all the previous data, we've come up with a strategy that will look at the top 15 politicians that will be filtered by the ones with the highest gain/loss percentage and that have made at least 100 trades. Then we will buy the same stock on or just after the date that the trade becomes public and hold it for 30 days. This holding period gives the market time to react to the news, which has shown in the past to sometimes result in short-term price increases.

To decide whether to buy or not a stock, we will use our machine learning model that has learned on past congressional trading data. This model, since it is a binary one, will just predict whether it's worth buying or not, and we will only include the decision if it has 70% or more probability of success.

Finally, with the stocks we get we will build a portfolio and optimize it by assigning weights to each selected stock in a way that maximizes the Sharpe ratio (a measure of return adjusted for risk). This will help us create a well-balanced portfolio each month that aims to achieve the best possible performance with controlled risk.

Body of the report

In this section we will explain the full development process where we transformed raw congressional and insider trading data into a predictive investment strategy model. Starting with data cleaning, exploratory analysis, labeling, model development and personal information.

Data description

The first database contains Insider information, which includes details about transactions made by senior executives, directors, and other employees with privileged access within their own companies. This type of operation is insider trading, which provides insight into the executives' confidence regarding the company's future performance. The database includes records such as transaction dates, the executive's name, title, type of operation (buy or sell), the number of shares traded, the share price, and the percentage change in share ownership.

The second database, called Congress Trading, contains information about transactions made by congress members and public officials who, due to their positions, may have privileged information regarding policies and regulations which allows them to have early access to events that may affect the stock market, which could be crucial in identifying patterns that indicate significant market movements.

Both databases contain historical information on relevant transactions that can be analyzed through advanced statistical techniques to identify trends, correlations, and most importantly, predictive patterns. The objective is to build a model that can accurately determine

when it is convenient to buy or sell stocks, thereby maximizing profits based on the buying and selling activities of insiders and congress members.

Insider Trading

Databases explanation

Insiders Trade Database

- Number of rows: 753,196 rows.
- Number of columns: 19 columns
- 12 columns have exactly 4,336 rows with missing data.
- Columns information:
    - Column #1 “Trade Date”: date when the insider transaction took place.
    - Column #2 “Filling date”: date and time when the transaction to the SEC (Securities and Exchange Commission).
    - Column #3 “Symbol”: stock ticker symbol of the company.
    - Column #4 “Insider Name”: name of the individual who made the transaction.
    - Column #5 “Title”: corporate title of the insider (ex: CEO, VP)
    - Column #6 “Trade type”: type of financial operation (ex: sale)
    - Column #7 “Price”: price per share at which the trade occurred.
    - Column #8 “Qty”: quantity of shares traded.
    - Column #9 “Owned”: number of shares the insider owns after the transaction.
    - Column #10 “ΔOwn”: percentage of change in ownership due to the trade.

- Column #11 “Value”: total dollar value of the transaction.
- Column #12 “SeñalC”: a binary indicator (1 or 0) indicating a buying signal.
- Column #13 “SeñalV”: a binary indicator signifying a selling signal.
- Column #14 “Value_C”: dollar value of the trade if it is a buy.
- Column #15 “Value_V”: dollar value of the trade if it is a sale.
- Column #16 “Year”: year of the trade.
- Column #17 “Month”: month of the trade.
- Column #18 “Quarter”: financial quarter of the trade.
- Column #19 “is_sp500”: indicates whether the company is part of the S&P 500 index (True/False).

Databases transformations

Insiders Trade Database

- Initially there are 753,196 rows and 19 columns.
- 12 columns have exactly 4336 rows with missing data, for this reason the dropna function is used.
- The database is left with 748,824 rows and 19 columns with 0 missing data.
- 99.42% of the data is retained.

Feature engineering

In order to create the machine learning models that accurately predict the change in the price of an action, some additional processing and work in the database was needed. This additional work was focused on not only using the columns and data received from the original

database, but also adapting these original columns, as well as creating new ones that would benefit the predicting ability of the machine learning model.

As stated before, the original database already came with very detailed columns that were ready to be used for the model; however, there were some small changes to apply to a specific column, Column #10 "ΔOwn." This column had a small issue that needed to be corrected: it wasn't a numeric value, and it displayed the percentage in the wrong format. Changing this value into a numeric value is a small change, but it is crucial for future processes.

In addition to changing original columns, some columns were created in order to help the model predict better into the future. The columns created were:

- **Price on Filed Date**: Numeric value that displays the price on the Filing Date.

- **Success Percentage**: Numeric value that displays the percentage of trades that the insider has increased his "Owned" value over the total number of transactions.

- **Days between Filed Date and Purchased Date**: Numeric value that displays the number of days between the day that the insider made the transaction and the day that the transaction was published.

- **Price on Filed Date vs Purchase Date Ratio**: Numeric value that displays the ratio of the price on filing date over the price on the purchase date.

- **Politician Activity Frequency**: Numeric value that displays the number of times the insider made a transaction.

- **Symbol Activity Frequency**: Numeric value that displays the number of times a symbol appears on the database.

All these columns were created in order to make the model take more and more information in order to have better predictability. In the same way, these columns show different types of trends. For example, the column “Days between Filed Date and Purchased Date” shows the delay between the two dates. This is a simple way to identify trades that have a larger delay, which could show transactions for which the insider had access to information before the price changes drastically.

In addition to the column changes there was another key step in the process of developing this model, the final dataset had 25 columns, 748,824 rows and 0 missing values; this presented a completely different level of computing power if the model was based on the complete database. Therefore, the database was cutdown from all the insiders into the top 250 insiders by a various of different metrics which included:

- Average “Change +30”: This cuts down the list to the most “successful” trades in the database while also protecting the metric from outliers, as the first idea was to use the total sum of the same metric.
- Number of appearances: This was a hidden threshold used in order to use only the insiders that had a higher number of appearances as it could show a lot more information than one singular insider with one transaction that had a very high change in price. Threshold: 100 appearances
- Weighted Score: This feature is created by multiplying the Average Change +30 with the number of appearances, benefiting the insiders that are more consistent over a larger number of transactions.

With these metrics the database was cut down from 748,824 to 5,592 rows, to ease up the computing power needed.

Model Development

For the model creation process we passed through a whole journey, the initial idea was to make the model as simple as possible without leaving behind the accuracy metrics that we wanted to achieve. The first attempt started as a simple linear regression model which immediately proved it was too simple for us to use in this exercise as it had too many variables to take into account which made it difficult for the model to predict accurately, therefore we started our journey to discover the machine learning model that could give us the best accuracy scores we could get. As we kept on trying different models passing by models like Random Forest Classifiers, Gradient Boost Classifiers, XGBoosts classifiers and more; in the end the XGBoosts Classifier was the one that presented the best results however it could be improved, applying technology like hyper parameters, auto stopping and splits in the data made the model more and more precise in predicting the change in the price.

At the end of all our machine learning creation process we found different issues and things to consider before continuing our project and testing our model we had to have the discussion to define if our models where good enough to be tested with real data, as it takes a considerable amount of computing time to develop this models and to test them, we decided that to consider as an effective predictive model it should show an accuracy of at least 70% and as you can see none of the models that we tried to develop with this database surpassed this threshold therefore this models were not tested with unseen data.

Table 1

Comparison between machine learning models

Metric	Random Forest	Gradient Boost	XG Boost
Accuracy	54%	55%	60%
Precision (Positive)	55%	54%	63%
Recall (Positive)	59%	88%	59%
F2 Score	59%	78%	59%

Source: own elaboration

As shown in the table, the results from the machine learning models did not reach the 70% accuracy threshold that was set at the beginning of the project. However, the closest we came was with the XGBoost model, which included several modifications. Given its importance and better performance, and the adjustments made to improve it, this model will now be explained in more detail to highlight the specific changes and their impact.

Base Model:

The base model that was created was an XGBoost model, as stated before this model works by creating a series of decision trees in a sequential manner and correcting and improving each tree for the classification task, the key characteristics that made us choose this model were its feature of regularization which reduces the general overfitting as well as its over parallel processing which helps to optimize the computing power and time.

Database Split:

This allows to separate the database into 3 sections, Train - Validation - Test, the training set is composed from 60% of the data and it is used for the learning process of the model; The Validation set is composed from 20% of the data at is used during the Optuna Training Process; the process to evaluate different model configurations and identify the optimal hyperparameters. And finally the last 20% of the dataset for the test set, which is the last set used on the model, completely separate from the previous steps, used to evaluate the model with unseen data and give metrics on its performance. The key characteristic of this step is that it includes the parameter, *Stratify,* which makes sure that each set contains a balanced number of gain and losses in order to decrease the effect of overfitting and is crucial to use when dealing with unbalanced data sets.

Model Tuning with Optuna:

Optuna Tuning is the python library used to find the best hyperparameters to apply in the model by creating a wide number of trials, each one changing the hyperparameters to find the best combination of these parameters for the model to use. In this specific project there were three static parameters, *Objective* that was set to "Binary" in order to make the XG Boost model a classification model, the second static parameter is the *eval_metric* that was set to AUC in order to output a accuracy score to be evaluated and lastly was the *random_state* which was set to "42" being a common practice across programming projects as well as being able to replicate the results. In the other hand there were seven tunable parameters for the Optuna tuning process to change this were the following:

Table 2

Hyperparameters Explanation

Hyperparameter	What is does?	Value	Why was it used?

n_estimators	Sets the number of decision trees the model is going to create	200 to 1000 (in steps of 100) Final Value: 800	Having a higher value makes the model more complex and accurate, in countersight having a higher number also makes the model slow and harder to compute.
max_depth	Sets the maximum depth of every decision tree	3 to 8 Final Value: 8	The depth is the parameter that allows the model to learn patterns. It needs to be high enough to detect the patterns and lower enough that it can overfit the data.
learning_rate	It controls how much each new tree contributes to the final prediction.	0.01 to 0.2 (log scale) Final Value: 0.039	The lower the learning rate the less prone to overfitting, however it requires a larger number of trees to train.
subsample	The percentage of rows to be randomly selected for building each tree.	0.6 to 1.0 Final Value: 0.682	Not using the 100% of data per tree makes the model less prone to overfitting.
colsample_bytree	The percentage of columns (features) to be randomly selected for building each tree.	0.6 to 1.0 Final Value: 0.866	Similar to subsample, this introduces randomness by limiting the features each tree can see.

reg_alpha	It encourages sparsity by pushing less important feature weights towards zero.	1e-8 to 1.0 (log scale) Final Value: 1.753e-05	This was a key component of the model as the model itself selects the features that show a better predictive power
reg_lambda	It penalizes larger weights to make the model's predictions less sensitive to any single feature.	1.0 to 10.0 (log scale) Final Value: 8.822	This parameter complements the reg_alpha work in order to avoid only one feature being selected.

Source: own elaboration

It can be seen that the model created was complicated enough to handle the dataset, even so in the end with the selected parameters and several iterations the model had a lot of issues and didn't show an accuracy high enough to surpass the threshold for backtesting; along the list of issued the model presented there is one worth mentioning. The main issue that appeared on the model was the high correlation between some features being used which damages the interpretability and reliability of the model, at the end these features were removed in order to decrease this negative effect.

Threshold Optimization and Evaluation:

In order to determine the output probability given by the model into the classification of Gain or Loss, an optimal threshold needs to be identified and it was found by using the validation set. This process involved evaluating the range of potential values, for each possible value an accuracy score was calculated by using the real data against the predicted data; the

threshold value that showed the highest accuracy score was selected as optimal and therefore being the one used on the test set, this way it can be assured that the model was calibrated to maximize its predictive power and accuracy. After completion the model decided the optimal threshold was 0.5157.

Results

In the following section the results and evaluation metrics given by the model will be shown and detailed for a further understanding of the project, as well as explaining the issues encountered during the process and the changes made to correct them.

- Accuracy Metrics

The model was evaluated under three main accuracy metrics; Accuracy, AUC Score and F2 Score; these three metrics provide a holistic view of the model as well as complementing each metric flaw in order to present an overall complete and robust evaluation. The first metric, Accuracy, is simple and straightforward as it evaluates the number of times the model was correct on predicting a gain or a loss against the real data. The second metric chosen was the AUC Score, which shows the model's capability of distinguishing a gain or a loss with an independent threshold. Finally, the F2 Score was specifically chosen to evaluate the model's practical utility for this financial task, as it places twice as much importance on Recall (finding all actual gains) than on Precision. This is crucial because, in this context, the penalty for missing a potential profit opportunity (a false negative) is considered higher than incorrectly predicting one (a false positive), and the F2 Score directly reflects this priority. The final accuracy metrics were:

Table 3

Accuracy Metrics of Machine Learning Model

Metrics	Precisión	Recall	F1-Score
No Gain	0.57	0.61	0.59

Gain	0.62	0.58	0.60
Accuracy	N/A	N/A	0.59
Macro Avg	0.59	0.59	0.59
Weighted Avg	0.59	0.59	0.59

Source: own elaboration

Table 4

Accuracy Scores of Machine Learning Model

Metric	Value
Threshold	0.5157
Accuracy	0.5934
AUC Score	0.6402
F2 Score	0.5881

Source: own elaboration

- Insider Analysis

Instead of relying on a single overall score, its main purpose is to evaluate how accurately the model predicted the trading outcomes for each individual insider. It essentially creates a performance report card for every person, using the unseen test data to provide a fair and

unbiased assessment. This performance report includes metrics such as Accuracy, F1 Score and precision as well as creating a full on False Negative matrix for each insider.

This detailed breakdown is crucial because it provides a nuanced evaluation of the model's reliability and risk-management capabilities, moving beyond a single accuracy score. By deconstructing each prediction into components like True Positives and False Positives. The final ranked list makes it easy to identify for which individuals the model is most effective, offering a deep and granular understanding of its specific strengths and weaknesses in a real-world context. The insider analysis evaluation was:

Table 5

Accuracy Metrics per Insiders

Insider Name	Total Trades	Accuracy	Precision	Recall	F1_Score
Lechleiter Richard A	32	71.88	0.73	0.69	0.71
Defeo Ronald M	43	67.44	0.73	0.67	0.7
Spurio Chris	28	53.57	0.56	0.67	0.61
Eberwein Jeffrey E.	57	66.67	0.71	0.54	0.61
Frost Phillip Md Et Al	300	57.33	0.62	0.58	0.6
Bunka Christopher	27	55.56	0.73	0.47	0.57
Perceptive Advisors LLC	28	57.14	0.47	0.64	0.54
Flynn James E	25	52	0.33	0.2	0.25

Source: own elaboration

Table 6

True Positive Evaluation per Insiders

Insider Name	TP	FP	TN	FN	Total_Correct	Total_Errors
Lechleiter Richard A	11	4	12	5	23	9
Defeo Ronald M	16	6	13	8	29	14
Spurio Chris	10	8	5	5	15	13
Eberwein Jeffrey E.	15	6	23	13	38	19
Frost Phillip Md Et Al	95	59	77	69	172	128
Bunka Christopher	8	3	7	9	15	12
Perceptive Advisors LLC	7	8	9	4	16	12
Flynn James E	2	4	11	8	13	12

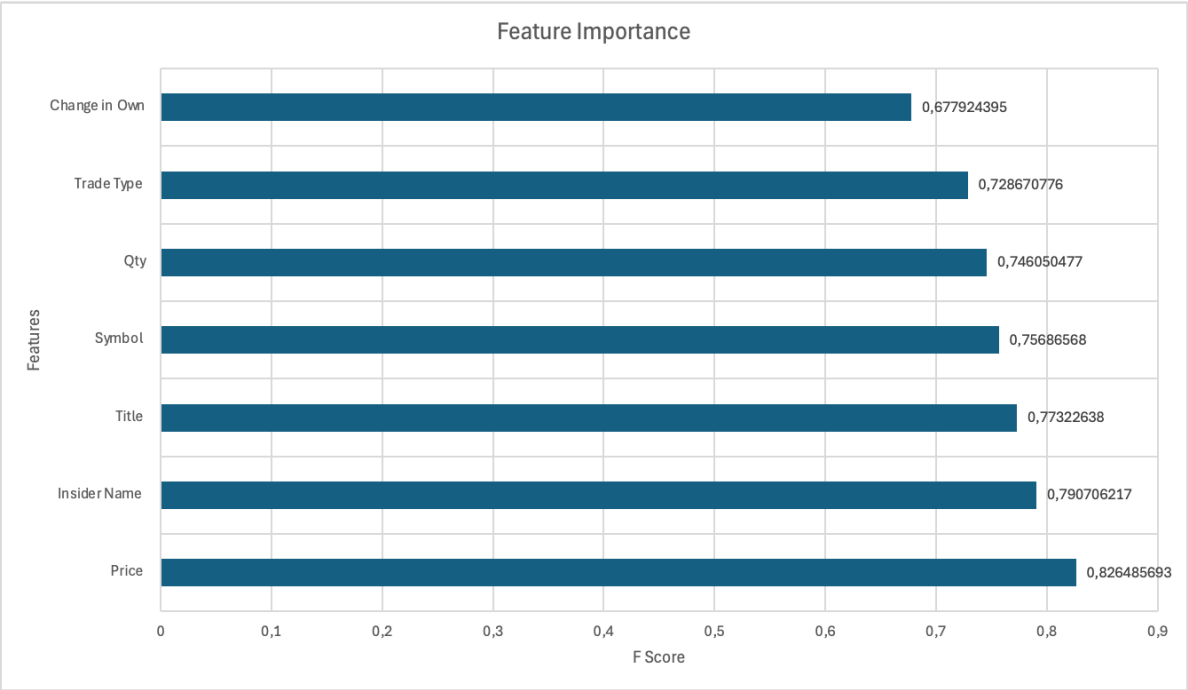
Source: own elaboration

- Feature Importance

As stated before, the model itself is the one selecting the features that affect the prediction, therefore a weight or F Score for each feature representing its importance on the predictive process for each transaction. The importance here is to understand which feature the model is selecting as more important in order to predict the change in price, as it is the key to understand how these transactions are made and why do the insiders make this decision along with the impact it has on the market. The final feature importance combination was:

Graph 1

Feature Importance



Source: own elaboration

Some of this results are very logical as the model aims to understand the impact an insider transaction has on the market, there is no reason why the insider name and price are the top two features in importance as the insider name is the feature that allows the model to identify impact and the price is the feature that gives the starting point for the model to predict the change of price after 30 days.

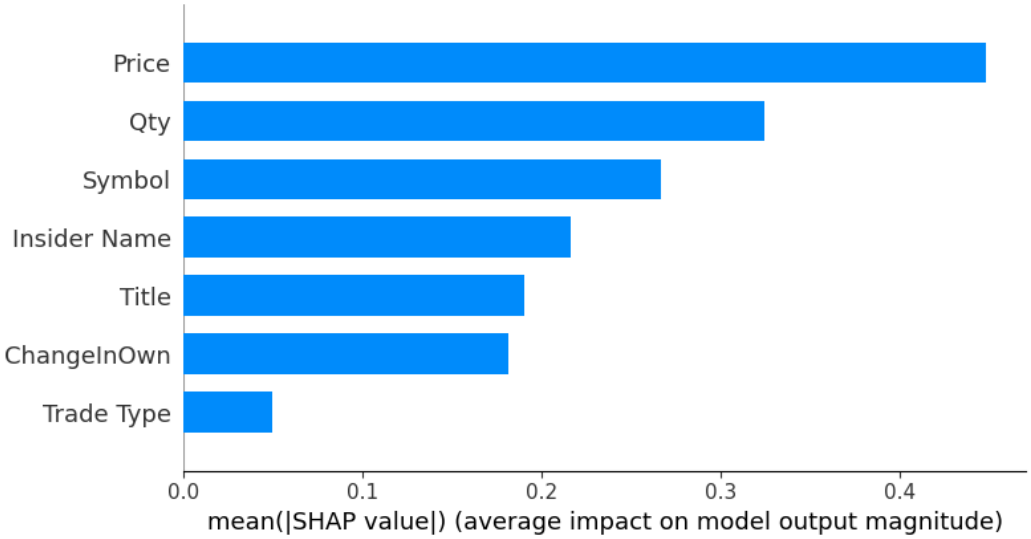
- SHAP Value

In order to have more metrics to consider a SHAP Value evaluation was done in addition to the first feature importance evaluation. SHAP refers to “SHapley Additive exPlanations” which is a powerful and popular method used to explain the predictions of a machine learning model on a case-by-case basis. Instead of showing what features are important overall, a SHAP

evaluation breaks down a single, individual prediction to show exactly how each feature contributed to that specific outcome. With this being said, the SHAP Value evaluation gives two important graphics:

Graph 2

SHAP Value for Features

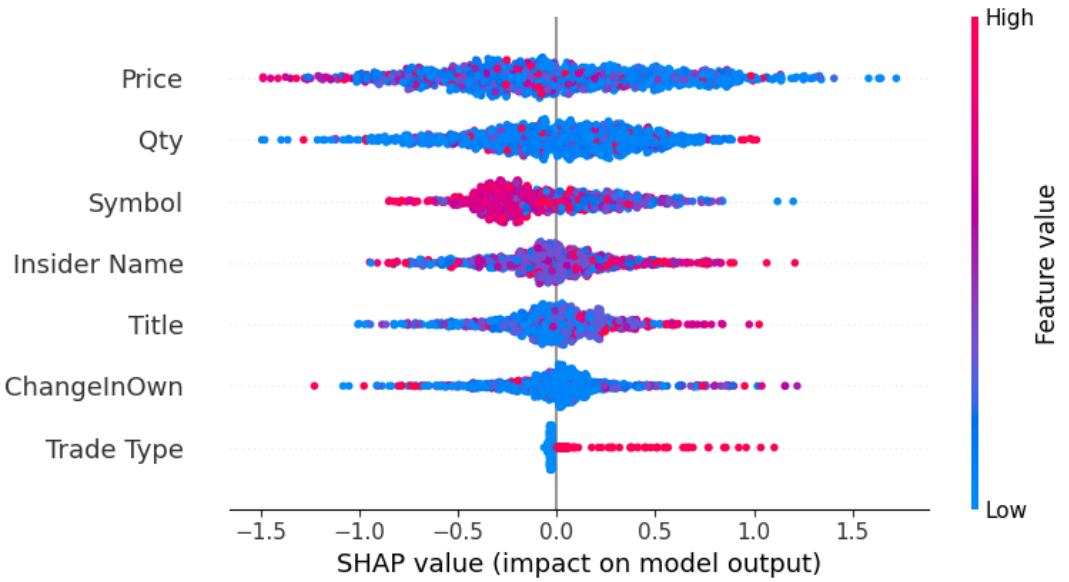


Source: own elaboration

The first graph shows the average impact of each feature in the predictive decisions, this complements the initial features selected by the model as it continues to show a similar trend of the feature importance in the model.

Graph 3

SHAP Summary Plot for Features



Source: own elaboration

The second graph shows a SHAP summary plot; it provides a high-level overview of the prediction process of the model, it shows both the importance and impact of each feature. Each dot on the plot represents a single prediction, the horizontal scale shows if the feature pushed the prediction to a positive (Gain) or a negative (Loss) value and its color indicates the features own value meaning high (Red) or low (Blue). A great example of this is the feature *Price* which clearly shows that higher prices, red dots, have a positive SHAP value making the prediction higher, whereas low prices, blue dots, push the prediction the other way.

- Correlation Matrix

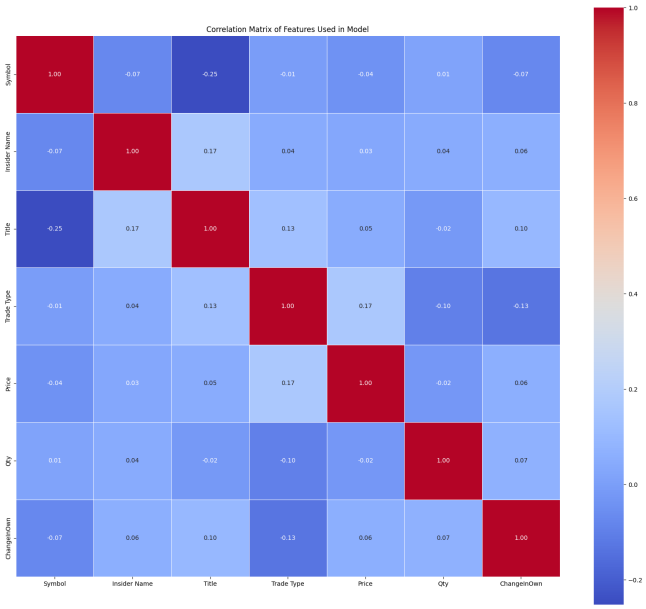
In the process of developing a machine learning model, it is crucial to minimize the correlation among features, regardless of whether that correlation is positive or negative. Ideally, the correlation coefficients of features should be close to zero, making the features independent from one another. When features operate independently from one another, it

simplifies the interpretation of the model's findings and clarifies the individual impact of each feature on the overall prediction. Increased correlation can result in model instability and render the assessments of feature importance unreliable, as the model finds it difficult to differentiate the distinct contributions of various features.

With this theory in mind, the correlation matrix was key in our evaluation process as the project aimed to be as precise and reliable as possible. As stated before, in this step is where a lot of issues appear as most of the features created to complement the model ended up having very high positive correlations, meaning they where affecting the reliability of the model as well as damaging the overall accuracy and that's why most of the features created didn't end up on the final feature importance selection. In the end, the correlation matrix between the features was:

Graph 4

Correlation Matrix Between Features



Source: own elaboration

As it can be seen in the graph there is not a single combination of features that has a high correlation value, positive or negative. Having the vast majority of the correlation values between the -0.2 and 0.2 range is a great sign of reliable and positive results for the model; there is only one correlation between features that surpasses this -0.2 - 0.2 range with a value of -0,25 is the relationship between the features *Symbol* and *Title*, this features refer to the Ticker or name the stock is being traded in the market and the title refers to the position in which the insider is within the company. This correlation can be explained by understanding that most of the insider transactions made on the database were focused on insiders trading their own company stocks.

Congress trading

In the past years there has been an increase in the interest in the analysis of the financial behaviour of the congress members of the US, since they might have access to some privileged information about politics economics and some strategic sectors regulations, and their behaviour over their investments could represent some signals that could be used in the financial market. Many people try to follow the same strategy patterns when investing their money depending on the politician's transactions, creating some direct impact on the market. Taking this into consideration, we raise the question of whether it is possible to create an investment strategy based on the public transactions done by the US Congress members of the US?

To find the solution to this problem, a complete code architecture with analysis and different machine learning models was designed, starting from data cleaning and transformation to machine learning classification and financial backtesting.

Databases explanation

Congress Trading Database

- Number of rows: 21,211
- Number of columns: 17
- Columns information:
    - Column #1 “Unnamed”: auto generated index column, not used for forward analysis.
    - Column #2 “Symbol”: stock ticker symbol of the traded company.
    - Column #3 “Action”: type of transaction, either “Purchase” or “Sale”.
    - Column #4 “Filled date”: date when the trade was officially reported.
    - Column #5 “Purchase date”: actual date when the trade occurred.
    - Column #6 “Description”: additional details (mostly empty in this sample).
    - Column #7 “Gain/Loss”: percentage of change between purchase date and filed date.
    - Column #8 “Politician”: name of the Congress member who made the trade.
    - Column #9 “Transaction ID”: unique identifier for the trade (including chamber and member ID).
    - Column #10 “Company”: full name of the company involved in the trade.
    - Column #11 “Security”: type of financial security (ex: stock).
    - Column #12 “Amount range”: disclosed range of the transaction value (used for transparency).
    - Column #13 “Chamber”: the house of congress the politician belongs to (ex: “House”)

- Column #14 “Party”: political party of the reporting member (ex: “D” for Democrat)
- Column #15 “Industry”: industry sector of the traded company.
- Column #16 “Amount”: approximate numerical value of the traded (estimated).
- Column #17 “Filed Date +1”: the day after the filing date, used for timing analysis.

Database Cleaning

Congress Trading Database Cleaning:

- Initially, there are 21,211 rows and 17 columns.
- In the ‘Description’ column, the blank spaces or NaN rows are dropped because it does not contribute to the model.
- In the ‘Industry’ column, the black spaces or NaN rows are dropped because it does not contribute to the model.
- All the rows that do not have information in the ‘Symbol’ column are dropped because they do not contribute to the model.
- Dropped all the rows that have no information in the ‘Filled Date’ column, as they are not useful for the model.
- Dropped all the rows that have no information in the ‘Security’ column, as they are not useful for the model.
- All rows with no information in the ‘Gain/Loss’ column are dropped because they are not useful for the model.
- 80.49% of the data is kept.

Once the information is clean, it was decided to cut the database to only include the top 15 congress members. These were selected based on a weighted average calculated by multiplying

the number of transactions by their average gain/loss. This approach allowed the model only to focus on the most active and profitable congress members, by doing this, the database had only the politicians who had the highest market participation and consistently strong investment outcomes.

Limiting the dataset to the top 15 performers helped the models reduce the risk of overfitting and ensured that the models are trained over the most relevant information for predicting future trading success.

The top 15 congressmen that are left on the database are the following:

Table 7

Top 15 Performers

Politician	Average Gain/Loss	Number of Appearances	Weighted Score (Total Gain/Loss)
Sheldon Whitehouse	219.442312	454	99626.809582
Thomas R. Suozzi	62.190193	507	31530.427804
Josh Gottheimer	11.713083	2429	28451.079533
Mark E. Green	45.288361	346	15669.772902
Ron Wyden	72.613859	208	15103.682697
Pete Sessions	45.715975	304	13897.656511
Nancy Pelosi	84.745349	131	11101.640715
Suzan K. DelBene	67.727437	125	8465.929570
Greg Landsman	26.066907	110	2867.359815
Markwayne Mullin	1.944778	171	332.557064
Michael C. Burgess	-1.862768	124	-230.983282
Kathy E. Manning	-0.670848	682	-457.518576
Marjorie Taylor Greene	-3.695915	195	-720.703449
Jerry Moran	-5.429966	178	-966.533885
John James	-5.621748	496	-2788.387181

s

Source: own elaboration

Feature Engineering

In order for the model to obtain a higher accuracy, Feature engineering was used to obtain the changes between the same variables obtained from the database. These engineered features allowed the model to better interpret the congress trading behaviour and market context, which boosted the accuracy and precision.

Some of these variables are the following:

- Price_on_Filing_Date: Market price of the stock on the filing date (via Yahoo Finance)
- Success_Percentage: Win rate of past trades by the insider
- price_ratio_filed_vs_purchase: Ratio of filing price to purchase price (indicates early opportunity)
- Days_Between_Filed_and_Transaction: Delay in reporting the trade
- Insider_Activity_Frequency: Number of trades made by the insider
- Symbol_Activity_Frequency: Number of trades made across all insiders for that symbol

The target variable was named Gain_Label, which was transformed with the label encoder in a binary classification:

- 1 If the stock price is higher than the filing date in 4, 15, and 30 days after the filing date.
- 0 If the stock price is lower than the filed date in 4, 15, and 30 days after the filing date.

In this model it was analyzed the short term to midterm reaction post disclosure.

Database Splits:

The databases were split into different subsets:

- Train 70% (4,522 Transactions), used to train the model.
- Validation 10% (646 Transactions), used to tune hyperparameters and validate the model´s performance during training.
- Test 20% (1292 Transactions), used to evaluate the final model´s performance.

Model Development:

Three different machine learning models were used to find out which one worked best for the prediction task. These were:

- Logistic Regression: A basic and easy-to-understand model that worked as a starting point for comparison.
- Random Forest: A more advanced model that combines many decision trees. It's better at handling complex patterns in the data.
- XGBoost: A powerful and fast model that often performs well when working with structured data.

Below is a summary of the performance of each model on the test set:

Table 8

Summary of the performance per model

Metric	Logistic Regression	Random Forest	XG Boost
Accuracy	57%	68%	71%
Precision (Positive)	57%	79%	75%

Recall (Positive)	90%	58%	77%
F2 Score	60%	61%	76%

Source: own elaboration

These results gave us some important insights:

- Logistic Regression had a high recall (90%), meaning it found most of the winning trades. But its precision was low, meaning it also picked many bad trades by mistake. In finance, this is risky because we want to avoid false positives.
- Random Forest was better at precision, meaning it picked fewer bad trades. But it missed many good ones, which lowered its overall performance.
- XGBoost gave us the best results. It had the highest accuracy (71%) and the best F2 score (76%). This is important because we wanted a model that catches as many good trades as possible without including too many bad ones. XGBoost found a good balance.

After all the models that were tested, the XGBoost was selected as the optimal and final model. Based on the balance, it shows by catching the most winning trades as well as not selecting a high number of losing trades. In this type of financial project and context, it is better to have a False Negative rather than a False Positive, as the false positives would end up in situations where money would be lost. In addition to the model selection, a deep dive and understanding of how this model makes its decisions, for this deep dive, the following metrics were selected:

- SHAP values, which helped us see which features have a higher impact in the model.

- ROC curves and precision-recall graphs, to help choose the best threshold for the model to classify the trade should be marked as a “Gain” or a “Loss”
- And some confusion matrices for each politician, to check if the model was being fair, or was undergoing overfitting or underfitting.

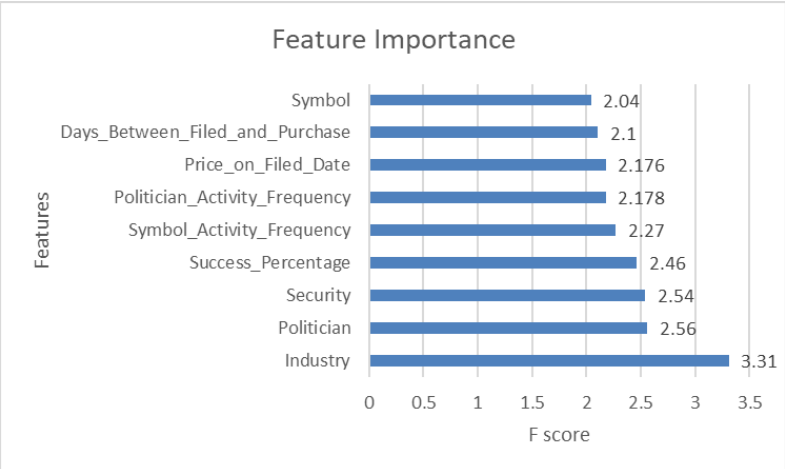
RFECV:

The model uses the Recursive Feature Elimination with Cross Validation (RFECV) inside the XGBoost Model. The RFECV is crucial for feature selection in Machine Learning, and it helps to automatically select the most relevant variables by optimizing the area under the curve in order to improve the model performance and reduce overfitting. The RFCEV tests how well the model performs with the current subsets of features, and this process iterates continuously by eliminating one feature at a time until the model reaches the best performance, in this case, the AUC, with the optimal number of features[11].

The most relevant features obtained with the RFCEV are the following:

Table 9

Feature Importance with RFCEV



Source: own elaboration

Correlation Between Imput Variables:

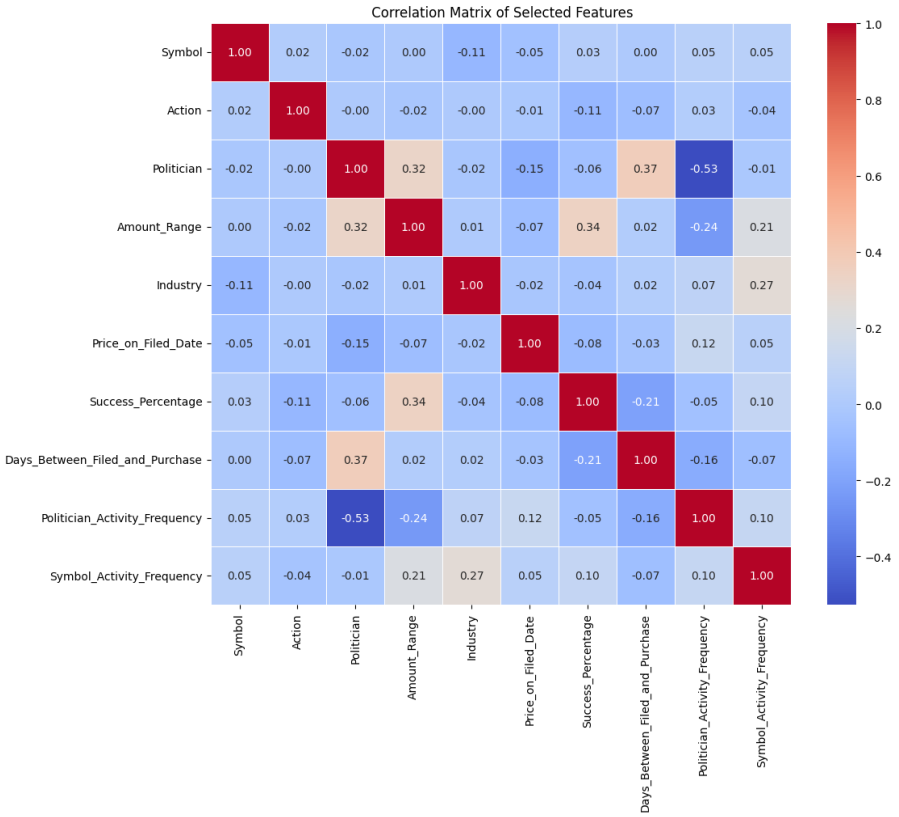
Once the model is completed, a Pearson correlation matrix of variables that were chosen from the RFCEV was performed, since it is important not to have a correlation between the variables to increase the performance and interpretability of the model. If two or more variables are correlated, they would give the same information to the model and in consequently, more importance to some of the variables. Many correlated variables increase the risk of overfitting. When a highly correlated variable is found, it is important to eliminate it, since the model would be trained with repeated information, this way, the model only uses the ones that have a higher impact and helps to generalize better. The XGBoost Model is not affected by the correlated variables since it is a tree-based ensemble approach, and it allows it to effectively learn from such data without a significant negative impact. The correlation matrix would only help to make a decision to see if some variables must be removed to reduce the noise in the model.[12]

The Pearson Correlation matrix between two variables is interpreted as +1, -1, and 0. When the values are closer to 1 means that the variable has a high positive correlation, if the values are closer to -1, it represents a strong negative correlation, and when the values are closer to 0, it means that they don't have any correlation. If the variables have a correlation of +0.8 or -0.8, one of the variables must be eliminated.[13]

The correlation matrix obtained from the variables chosen by the RFCEV are the following ones:

Graph 5

Correlation matrix between variables.



Source: own elaboration

In the correlation matrix, some independent variables have some correlation:

Politician vs Amount Range: (Correlation of 0.32) Some politicians work with similar amount ranges of investments.

Politician vs Days between Filed and Purchase: (Correlation of 0.37) Some politicians can have patterns of the window time with the filed and purchase date.

Amount Range vs. Success Percentage: (Correlation of 0.34) The results show that the politicians with the highest success rate operate with higher amounts of money, and could also represent that they might have more information about their investment decisions.

Politician vs. Politician Activity Frequency: (Correlation of -0.53) This is an artificial negative correlation, since the politician variable is encoded and the other one is its frequency, they don't have a linear simple relation since the encoded doesn't reflect hierarchy

Symbol Activity Frequency vs. Industry: (Correlation of .27) Some sectors can be more popular in the investments of the politicians

All the variables are accepted in the model since they are inside of the range of +0.8 and -0.8.

SMOTE

In machine learning, classification is a task that aims to assign a category or label to data points. In the case of this dataset, it is unbalanced, meaning that one or more classes have significantly fewer instances. In the case of the database of the congress trading, it has more wins than losses in the Gain Loss feature. It is important to balance the databases because when training a Machine Learning Model, when the data is unbalanced, it could lead to models biased toward the majority class. After all, there are more samples to learn from. Meaning that even a bad model can achieve a high overall accuracy if the majority class dominates the datasets, which wouldn't be correct for this type of analysis, and even more so when talking about investments.[14]

SMOTE was used to preprocess the data before training the classification model since it is a data argumentation that helps to balance the database by generating examples that are

similar to the ones in the database for the minority class instances, this ones are chosen by using the nearest neighbors from the minority class which would improve the reliability of the model by being more effective in learning of underlying patterns and decision boundaries. It also helps to reduce the overfitting in imbalanced datasets and create a more generalizable model by generating new data. In the following table there is a comparison of the before and after of the database for the Dependent Variable (Gain_Label)

Table 10

Before and After SMOTE.

Classification	Before SMOTE	After SMOTE
Winning Class (1)	2692	2692
Losing Class (0)	1830	2692

Source: own elaboration

Cross Validation:

Cross validation is a technique used to check how well a machine learning model performs on unseen data. When the cross validation is used, it splits the database into several parts, trains the model on some parts, and the remaining part is used to test the results It then iterates following that process, and the results of the validation step are averaged to produce a more accurate model. By using cross validation it also helps the model to prevent overfitting, and this way you can be sure that the model is not just memorizing the information, but learning patterns and increasing the prediction accuracy. In this model it is being used the stratified cross validation is used since it is a technique used to ensure that each fold of the cross

validation process maintains the same distribution as the entire dataset and it is essential when dealing with classification problems and it is crucial for the model to generalize well the unseen data.[15]

The dataset is divided into K folds and maintains the proportions of the classes in each fold. In each iteration done by the cross validation, one-fold is used to test, and the other folds are used for the training part.

In the model, we are using 3 folds for each of 36 candidates, totaling 108 fits. The three folds represent how many times the data was divided. In this case, the model is trained with two parts, and the third one is used for validation. This is repeated for every combination of hyperparameters. The 36 candidates mean that the GridSearch is evaluating 36 different combinations of hyperparameters, for every one of the 36 combinations, 3 trainings are made (one for each fold,) totaling 108 fits.

GridSearchCV:

When doing a machine learning model, it is important to take into consideration finding the best parameters to achieve the best performance possible. This process is usually time consuming and complex, but by using the tool GridsearchCV it helps to automate the process.

GridsearchCV is a method commonly used in machine learning to find the best hyperparameters. The hyperparameters are the settings that are configured before training the model. The tool GridSearchCV configures the grid search with the model, parameter grid, and cross validation strategy and scoring metric. This way, it can evaluate the best model found on the test data using different metrics to evaluate the model.[16]

By using the cross validation in the model, it lets you find the combination in which the hyperparameters maximize the AUC. By doing this, it evaluates the different configurations

with cross validation to decrease the risk of overfitting. The hyperparameters that are being used are the following:

- Learning rate: Controls how the model is adjusted for every tree. A higher value means that it will learn faster, but it increases the risk of overfitting.
- Max_depth: Maximum depth of the tree. If there is more depth, the model will be more complex.
- n_estimators: Total number of trees. If the number increases, the better the adjustment it will have.
- Subsample: Represents the percentage of the samples used per tree. If the value is under 1.0 increases the randomness and helps to reduce overfitting.

The best hyperparameters that were found with GridsearchCV for the different models are the following:

Table 11

Best hyperparameters found with GridsearchCV

Model Hyperparameters	Prediction 4 Days after the filing date.	Prediction 15 Days after the filing date.	Prediction 30 Days after the filing date.
Learning_rate	0.1	0.1	0.1
Max_depth	7	7	7
n_estimators	200	200	200
subsample	0.8	1	0.8

Source: own elaboration

The same model architecture across all three prediction horizons, the learning rate, max depth and number of estimations stayed the same in the 3 models, the subsample parameter varied slightly for the prediction 15 days after filling date, in which the model used the full sample, while in the other models they are using only the 80% of the samples used per tree, this change might have reduced the regularization in the prediction of 15 days after filling date.

Confusion Matrix

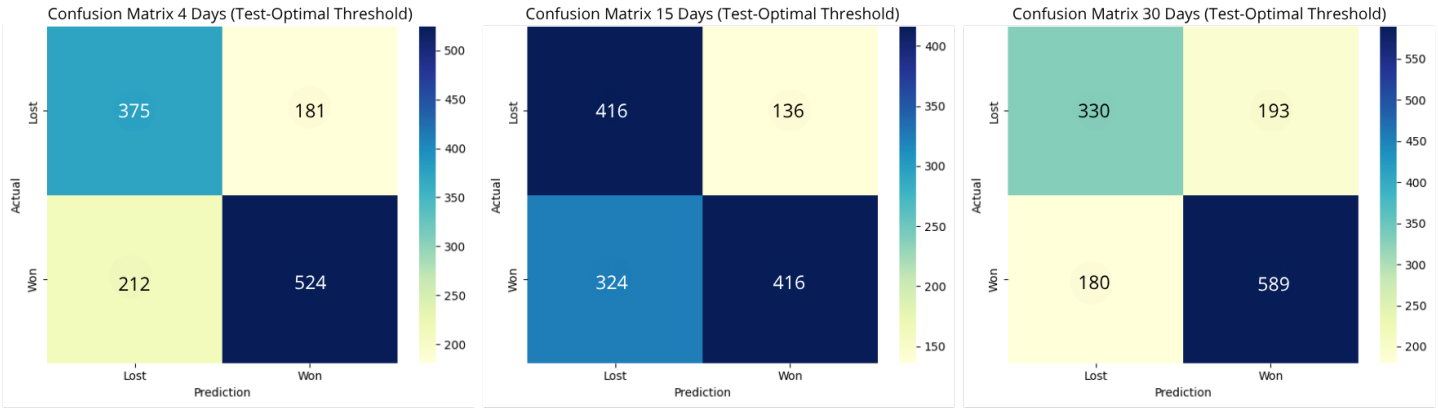
The confusion matrix is a table used to evaluate the performance of a classification model. It provides a detailed breakdown of the model´s prediction compared to the actual outcomes. Since the model is used to solve a binary classification problem, a confusion matrix was used to evaluate the performance with the following four categories:

- True Positives: The model correctly predicts the positive outcome.
- True Negatives: The model correctly predicts the negative outcome.
- False Positives: The model incorrectly predicts the positive outcome.
- False Negatives: The model incorrectly predicts the negative outcome.

The results of the confusion matrix for the predictions at 4, 15, and 30 days after the filing date are the following:

Graph 6

Confusion matrix 4, 15, and 30 Days after filling date.



Source: own elaboration

The same XGBoost classification model was trained using the different dependent variables representing the stock price movement. The confusion matrix shows how the model´s behaviour shifts depending on the prediction horizon.

The model of 4 days after the filing date maintains a balanced performance with both good precision and recall. At 15 days, the model becomes more conservative, lowering the false positives at the cost of missing more true positive signals, meaning that the predictive power of the congressional trades begins to fade over time. However, at 30 days after the filing date, the model becomes more aggressive in identifying positive movements, increasing the true positives, and in consequence, it also increases the false positives. These results might represent the tradeoff between precision and recall as the prediction window increases and might reflect the decreasing informational advantage or market impact of congressional disclosures over time.

The change in the hyperparameters in the prediction for the 15 days after the filing date had slightly reduced the regularization, possibly contributing to its more conservative behaviour seen in the confusion matrix, since it has fewer false positives and more missed opportunities. This also shows how this slight change in the hyperparameters can significantly change the results of the model performance, affecting how the model balances the bias and variance depending on the time horizon.

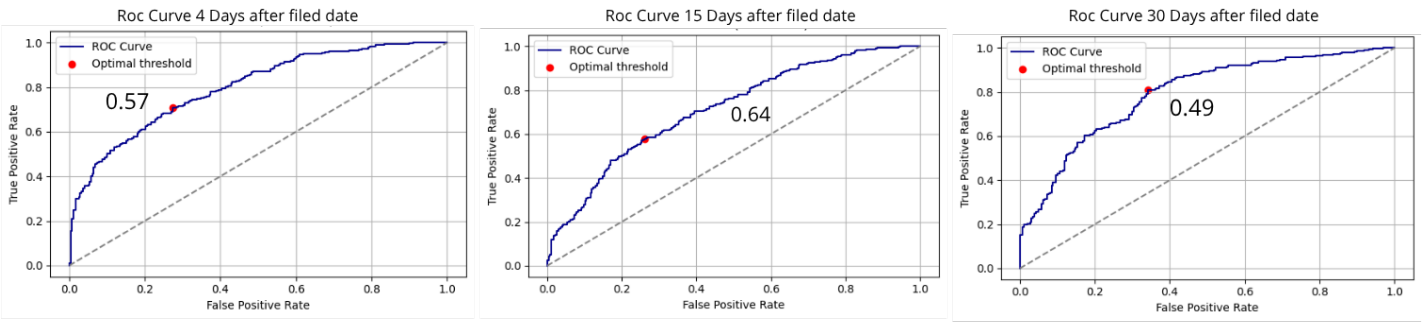
ROC Curve

The Receiver Operating Characteristic (ROC) Curve is a graphical representation that shows the diagnostic ability of binary classifiers. It shows the relationship between the true positive rate (sensitivity) and the false positive rate (1-specificity) across the different threshold settings. The more that the ROC Curve is closer to the top left corner, the better the model does at classifying the data into categories. A model with an AUC equal to 0.5 would be a perfectly diagonal line, and this represents that the model is not better than a model that makes random classifications[17]

The ROC Curve with its optimal thresholds obtained with the different models predicting for 4, 15, and 30 days after the filing date are the following:

Graph 7

ROC Curve for 4, 15, and 30 Days after the filing date.



Source: own elaboration

With these results, it can be concluded that the ROC curve for the 4 days model is closer to the upper left corner, which means it has a high True Positive Rate and a low False Positive Rate. The model has good discriminative power in the short term, and the threshold of 0.57 indicates that the model needs more confidence to predict a win.

In the model of 15 days after the filing date, making the predictions becomes harder, and this is reflected by the need to have a high confidence to predict a win, the highest threshold represents that the model is more conservative. The ROC curve is still over the baseline, which represents that it works better than the random ones.

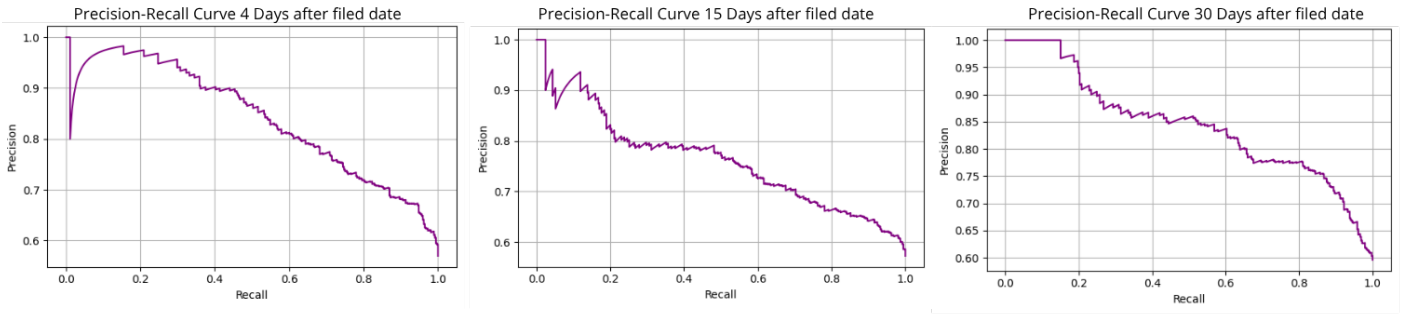
The 30 days after the filing date model shows a lower predictive power, since the threshold is closer to 0.5, which means that the predictions are balanced. The ROC curve is over the baseline, which represents that it works better than the random ones.

Precision-Recall Curve:

The PR Curve is a graphical representation to evaluate the performance of a binary classification model. The plot includes the Precision on the y-axis and Recall on the x-axis for the different probability thresholds. A model that works correctly will be represented by a curve that bows towards the coordinate of (1,1), while a no skill classifier will be shown as a horizontal line with a precision proportional to the number of positive examples in the dataset, The ROC Curve also provides insights about the model capability to classify positive instances.[18]

Graph 8

Precision-Recall Curve for 4, 15 & 30 days



Source: own elaboration

The PR curve in model 4 days after the field date has a high precision at the beginning with low recall. As time goes by, the recall increases and the precision gradually goes down, which means that it is a conservative model and precise when detecting a positive case. It shows a good curve when the priority is precision, it avoids false positives.

The model of 15 days after the filing date has some noise at the beginning, the precision decreases faster when the recall increases, this curve shows lower precision when increasing the recall, this is not an optimal curve since it shows less stability and a worse performance in general.

The model of 30 days after the filing date at the beginning shows high precision, and it stays stable between 0.85 and 0.9 until the recall reaches 0.6, after that, the precision goes down. This model has a better prediction model and more stability, obtaining a better relationship between precision and recall.

Final Models:

Once the model was completed by doing the data cleaning, feature engineering, and training of the model it was analyzed the performance of the three models of XGBoost trains for the different time horizons of 4. 15 and 30 days after the filing date the final results where the following ones on the test set over the key evaluation metrics:

Table 12

Summary of model results

Performance Metrics	XGBoost 4 days after the filing date	XGBoost 15 days after the filing date	XGBoost 30 days after the filing date
Accuracy	0.70	0.64	0.71
AUC	0.76	0.74	0.77
F2 Score	0.72	0.59	0.76
Recall positive class	0.71	0.56	0.77
Recall negative class	0.67	0.75	0.63
Precision positive class	0.74	0.75	0.75
Precision negative class	0.64	0.56	0.65
F1- Score positive class	0.73	0.64	0.76
F1- Score negative class	0.66	0.64	0.64

Optimal Threshold	0.57	0.64	0.49

Source: own elaboration

The model XGBoost for 30 days after the filed date showed a better general performance, obtaining an accuracy of 71%, an AUC of 77% and an F2 Score of 76%, which shows a good capacity to identify correctly the positive trades. The model also shows a positive recall of 77% and an F1 score of 76% which are better than the other ones.

The model of 4 days after the filing date also showed good performance metrics such as an accuracy of 70%, an F2 Score of 0.72% and the AUC of 76%, the only problem with this model is its investment horizon because it is too short, which limits the reaction time after the filing date. On the other hand, the model of 15 days after the fling date is the least efficient, with an accuracy of 64% and an F2 score of 59%, which shows less equilibrium between the identification of opportunities and the reduction of false positives.

For these reasons, the model of 30 days after the filing date is used as the final model to create an investment strategy. This model has the best predictive performance, and it has a real life application because of the larger reaction time to buy the stock.

Results

After finalizing the model and measuring its performance it was time for a real test, The results that were obtained from the model, the stocks that where chosen are the ones appearing in the following Table:

Table 13

Results From the Model

Politician	Symbol	Action	Industry	Amount_Range	Predicted_Probabili	Predicted_Label	Actual_La	Change_%_+30_Days	Filed_Date	Purchase_D
John James	DFS	Purchase	Financials	$1,001 - $15,000	0.50918263	1	0	-0.005792065	02/09/2024 00:00	18/07/2024
John James	DFS	Purchase	Financials	$1,001 - $15,000	0.50918263	1	0	-0.005792065	02/09/2024 00:00	18/07/2024
John James	TMO	Purchase	Health Care	$1,001 - $15,000	0.60445035	1	1	0.010372178	02/09/2024 00:00	06/03/2024
John James	WAB	Purchase	Industrials	$1,001 - $15,000	0.917833	1	1	0.1041920545941993	02/09/2024 00:00	14/02/2024
John James	ADP	Purchase	Information T	$1,001 - $15,000	0.92841345	1	1	0.029032843	02/09/2024 00:00	10/11/2023
John James	CCI	Purchase	Real Estate	$1,001 - $15,000	0.874867	1	1	0.021438024	02/09/2024 00:00	02/05/2024
John James	NVDA	Purchase	Information T	$1,001 - $15,000	0.9525765	1	1	0.1004629629629629	02/09/2024 00:00	13/03/2024
John James	NVDA	Purchase	Information T	$1,001 - $15,000	0.9589157	1	1	0.1004629629629629	02/09/2024 00:00	09/02/2024

Source: own elaboration

Backtesting

After training the model, the next step was to test how it might have performed in the real world. This was done through backtesting, which means simulating what would have happened if the model had been used in the past. It helps check whether the strategy is useful, risky, or just lucky. To keep it consistent, a few simple rules were followed:

- Only include trades made by the top 15 politicians who had a high number of trades and a history of good results.
- Only use purchase trades, not sales.
- Only take a trade if the model predicted success with more than 70% confidence.
- Assume the stock was bought the day it was filed or the day after, and held for 30 days before being sold.

Summary of Returns and Risks

This table shows the basic performance of each stock that appeared most often in high-confidence trades. It includes their monthly and annual returns, along with their variance and volatility.

It is important because it helps understand the risk–return profile of each stock. For example, NVDA had the highest return but also the highest volatility, while CCI had negative returns, showing that not all stocks from congressional trades are good picks.

Table 14

Summary of Returns & Risks

Summary of Return and Risk						
Stock Name	APD	CCI	DFS	NVDA	TMO	WAB
Average Return per Month	0.48%	-0.37%	1.87%	5.03%	0.59%	2.36%
Annual Return	5.89%	-4.38%	24.91%	80.15%	7.33%	32.36%
Variance per Month	0.55%	0.71%	0.98%	2.46%	0.43%	0.52%
Annual Variance of Returns	2.19%	2.84%	3.92%	9.82%	1.71%	2.07%
Standard Deviation of Returns	14.81%	16.85%	19.80%	31.34%	13.08%	14.37%
Monthly Volatility	7%	8%	10%	16%	7%	7%

Source: own elaboration

Variance-Covariance Matrix

This table shows how the returns of each pair of stocks move together. If two stocks go up and down at the same time, they have a high covariance. If they move in different directions, the covariance is low or negative.

Understanding covariance is key when building a portfolio because if all the stocks move the same way the risk is higher, but if they have different behaviour, having them combined can reduce risk.

Table 15

Variance-Covariance Matrix

Variance-Covariance Matrix						
Stock Name	APD	CCI	DFS	NVDA	TMO	WAB
APD	0.55%	0.29%	0.06%	0.22%	0.22%	0.22%
CCI	0.29%	0.71%	0.29%	0.40%	0.34%	0.27%
DFS	0.06%	0.29%	0.98%	0.32%	0.09%	0.36%
NVDA	0.22%	0.40%	0.32%	2.46%	0.43%	0.41%
TMO	0.22%	0.34%	0.09%	0.43%	0.43%	0.21%
WAB	0.22%	0.27%	0.36%	0.41%	0.21%	0.52%

Source: own elaboration

Correlation Matrix

This matrix shows the correlation between stock returns, the values range from -1 to 1 which means that a value close to 1 indicates 2 stocks moving in the same direction, while a value near to 0 means they are mostly unrelated and a value near -1 would mean they move in opposite directions which in this case didn't happen

Having stocks with low to medium correlation is helpful in a portfolio because it reduces overall risk. This table confirms that the selected stocks don't all behave the same way, which is good for diversification.

Table 16

Variance-Covariance Matrix

Correlation Matrix						
	APD	CCI	DFS	NVDA	TMO	WAB
APD	**1.00**	**0.46**	**0.08**	**0.19**	**0.45**	**0.40**
CCI	**0.46**	**1.00**	**0.35**	**0.30**	**0.61**	**0.44**
DFS	**0.**	**0**	**1**	**0**		**0**

	08	**.35**	**.00**	**.21**	**0.14**	**.50**
NVDA	**0.19**	**0.30**	**0.21**	**1.00**	**0.42**	**0.36**
TMO	**0.45**	**0.61**	**0.14**	**0.42**	**1.00**	**0.45**
WAB	**0.40**	**0.44**	**0.50**	**0.36**	**0.45**	**1.00**

Source: own elaboration

Weight Distribution:

This table shows the final investment weights given to each stock after running the portfolio optimization strategy. Since the goal was to maximize the Sharpe ratio, we aimed to get the best return for the lowest amount of risk.

In this case, CCI was given 0% weight because its returns were negative. DFS, NVDA, and WAB were each assigned 25%, showing strong performance. TMO and APD were given smaller weights, offering moderate returns with lower risk.

Having this information on hand can be useful because it shows how by optimizing the portfolio, we can get the most efficient combination rather than just spreading the money equally.

Table 17

Weight distribution

Weight Distribution		
Stock Name	Annual Returns	Weights
APD	5.8872%	16.67%
CCI	-4.3833%	0.00%
DFS	24.9092%	25.00%
NVDA	80.1539%	25.00%
TMO	7.3276%	8.33%
WAB	32.3641%	25.00%

Source: own elaboration

Distribution of weight to maximize Sharpe Ratio and Returns

This table compares 2 optimized portfolio strategies, one that focuses on getting the "Max Return" which is the highest possible return and the other one that focuses on getting the highest Sharpe Ratio, which means better returns for the amount of risk taken.

This comparison helps show the trade-off between going for maximum profit and building a safer, more balanced portfolio. Although both strategies performed well, the Sharpe-focused one spread risk more effectively while keeping returns almost the same.

Table 18

Distribution of weights to maximize Sharpe Ratio and Returns

Distribution of weights to maximize Sharpe Ratio and Returns								
	Return	Sharpe	APD	CCI	DFS	NVDA	TMO	WAB
Max Return	36.19%	4.5454	0.00%	0.00%	25.00%	25.00 %	25.00 %	25.00 %
Max Sharpe Ratio	35.95%	4.6154	16.67 %	0.00%	25.00%	25.00 %	8.33%	25.00 %

Source: own elaboration

As part of the backtesting analysis, the monthly returns of the selected stocks were calculated. This was done to understand how each stock performed on a short-term basis, since the strategy involved holding each position for 30 days.

Table 19

Actual Results From stock Market

Actual Results From stock Market	
Buy Date	02/09/2024
Sale Date	02/10/2024
TICKER	% Change
APD	6.01%
CCI	3.50%
DFS	-0.58%
NVDA	10.06%
TMO	1.10%
WAB	10.42%

Source: own elaboration

Here are some key insights from the table:

- NVDA had the highest average monthly return at 5.03%, which helped explain why it received a high weight in the optimized portfolios. However, it also came with higher risk.
- DFS and WAB also performed well, with 1.87% and 2.36% respectively, offering a solid return with a more stable profile.

- APD and TMO had modest but positive returns, making them good candidates for balance and diversification.
- CCI was the only stock with a negative monthly return (-0.37%), which is why it was excluded from the final portfolio allocations.

After cleaning the data, picking the right features, and training the models and testing the model, the results gave us a good idea of what worked, what didn't, and what could maybe be improved later on.

First of all, the best model was XGBoost. It wasn't the first one tried, but it ended up being the most solid. It got about 71% accuracy. That might not sound amazing at first, but in finance, that's actually pretty good, especially when trying to avoid bad trades.

Additionally other metrics like precision and recall were considered. Precision tells how many of the trades the model said were good actually turned out to be good. XGBoost got 75% precision, so most of the trades it picked were winners. And it also had 77% recall, which means it caught a lot of the real winners too.

The model also scored 0.76 on the F2 metric, which was one of the main things that cared. F2 gives more weight to recall than precision, and that's good because it's important to catch as many good trades as possible, even when having a few bad ones in between. It's better to find the big winners than to avoid every single loser.

The model looked at how each politician performed. Some were much more predictable than others. For example, Thomas R. Suozzi and Kathy E. Manning had really strong results,

the model predicted their trades with high precision and recall. Others like Sheldon Whitehouse were a bit more all over the place and didn't perform as well. That helped figure out which politicians might be more useful to follow and which ones to maybe ignore or treat with caution.

Once the model was ready and started seeing good results from the back testing, it was ready to go one step further. Instead of just picking trades and checking if they'd go up or down, the following question was raised: what if a portfolio using the stocks that came up the most was built?

So exactly that was done. Six stocks that showed up often in successful congressional trades were chosen: APD, CCI, DFS, NVDA, TMO, and WAB. Then, instead of allocating equal capital to each stock, a portfolio optimization approach was applied to determine the most efficient weight distribution.

More specifically, to maximize the Sharpe ratio. This ratio is a common way to measure how good an investment is when also taking into account the risk. The higher the Sharpe, the better the return per unit of risk, which is exactly what most investors want.

Past monthly returns from each stock and calculated were used:

- The average return per month
- The total return for the year
- The variance and standard deviation were used to understand how risky each stock was

What was found was interesting:

- NVDA stood out by far. It had a monthly return of over 5% and an annual return above 80%, but it also came with high risk. However, it was a time when the microchips were booming, which can be a bit biased.
- DFS and WAB also gave solid returns with a decent balance of risk.
- CCI, on the other hand, actually had negative returns, so not every stock that appears in congressional trades is worth betting on.

Using optimization, the model assigned more weight to the higher-performing stocks (like NVDA and WAB) and less to the underperformers. This way, the portfolio became more efficient, meaning it aimed to get better returns without taking on unnecessary risk.

This part of the project helped move from just evaluating single trades to thinking like actual investors. It also gave something to compare the machine learning strategy to. Combining both, the ML predictions and a well-balanced portfolio, gave a smarter and more realistic approach to investing.

With everything considered, the results showed that using this kind of model, not just blindly copying politicians, can really help make smarter decisions. It’s not perfect, but it gives a solid edge.

Conclusions and Future Work

Looking back at the full process, this project was able to answer the main question: can congressional trades be used to build a real investment strategy? Based on the results, the answer is yes.

At the beginning, the dataset from Congress was messy, incomplete, and sometimes felt random. But after cleaning it and creating new features like success percentage and price ratio, clearer patterns started to appear. After testing different models, XGBoost stood out as the most balanced one, reaching 71% accuracy and an F2 score of 0.76, which was important since the goal was to catch as many winning trades as possible.

Once the model was working, a backtest was performed to simulate real-world results. The test only included trades from the top 15 most active and profitable politicians, and predictions were only used if they had at least 70% confidence. This helped focus the results on quality trades, and the model handled that filter well.

After that, the project moved one step further by trying to build a real portfolio using the stocks that came up most often. Monthly returns from companies like NVDA, DFS, and WAB were used, and a portfolio optimization process was applied to maximize the Sharpe ratio, which measures return adjusted for risk.

The findings were clear:

- NVDA had a very high monthly return (over 5%) and an annual return above 80%, but with high risk.
- DFS and WAB gave strong results with a better balance between risk and return.

- CCI had negative returns, showing that not all stocks appearing in congressional trades are good investments.

The optimization gave more weight to the best-performing stocks and reduced exposure to the weaker ones. This made the portfolio more efficient and realistic for real-world investing. This part of the project showed that it's possible to go beyond model predictions and apply the logic of real investors. By combining machine learning with portfolio thinking, the strategy became stronger and more useful.

There is still room for improvement. Some ideas for future work include:

- Trying different time horizons, such as 15 or 60-day returns.
- Adding insider trading data, which could complement the signals from Congress.
- Creating a real-time dashboard that shows new trades as they are filed and suggests actions.
- Improving the portfolio management by testing dynamic rebalancing, adjusting positions based on confidence, or other risk-based techniques.
- Including external data like market news or economic indicators to improve decision-making.
- Deploying the strategy in a sandbox or paper trading environment to track performance in live conditions.

Overall, this project showed how public government data, combined with machine learning and basic investment tools, can be used to build a smart and working strategy. It was not just about analyzing data, it was about creating something that could actually be used in the real world.

Bibliography

Baeldung. (2023). How to handle unbalanced data with SMOTE? Baeldung on Computer Science. https://www.baeldung.com/cs/smote-unbalanced-data

Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32. https://doi.org/10.1023/A:1010933404324

Brownlee, J. (2023). ROC curves and precision-recall curves for imbalanced classification. Machine Learning Mastery. https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-imbalanced-classification/

Capitol losses: The mediocre performance of congressional stock portfolios. (2013). The Journal of Politics, 75(2). https://www.journals.uchicago.edu/doi/abs/10.1017/S0022381613000194

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 785–794). ACM. https://doi.org/10.1145/2939672.2939785

Displayr. (2024). What is a ROC curve – How to interpret ROC curves. https://www.displayr.com/what-is-a-roc-curve/

Downey, L. (n.d.). Efficient market hypothesis (EMH): Definition and critique. Investopedia. https://www.investopedia.com/terms/e/efficientmarkethypothesis.asp

GeeksforGeeks. (2024). Cross validation in machine learning. https://www.geeksforgeeks.org/cross-validation-in-machine-learning/

Hagar-zakaria. (2022). Understanding GridSearchCV in machine learning from A to Z. GitHub. https://github.com/Hagar-zakaria/Understanding-GridSearchCV-in-Machine-Learning-from-A-to-Z

Heckmann, J., Jacobs, H., & Schwarz, P. (2024). Synthesizing information-driven insider trade signals. SSRN Electronic Journal. https://ssrn.com/abstract=4537187

Federal Bureau of Investigation. (2011). Hedge fund billionaire Raj Rajaratnam found guilty in Manhattan federal court of insider trading charges. https://archives.fbi.gov/archives/newyork/press-releases/2011/hedge-fund-billionaire-raj-rajaratnam-found-guilty-in-manhattan-federal-court-of-insider-trading-charges

MIT Press. (n.d.). Investment intelligence from insider trading. https://direct.mit.edu/books/book/2765/Investment-Intelligence-from-Insider-Trading

Investopedia. (2024). Portfolio management. https://www.investopedia.com/terms/p/portfoliomanagement.asp

Investopedia. (2024). Sharpe ratio. https://www.investopedia.com/terms/s/sharperatio.asp

Markowitz, H. (1952). Portfolio selection. The Journal of Finance, 7(1), 77–91. https://doi.org/10.2307/2975974

Martha Stewart's insider trading scandal. (n.d.). https://harbert.auburn.edu/binaries/documents/center-for-ethical-organizational-cultures/cases/martha-stewart.pdf

New York Times. (2020, March 20). Kelly Loeffler and Richard Burr were briefed on coronavirus. Then they sold stocks. What now? https://www.nytimes.com/2020/03/20/us/politics/kelly-loeffler-richard-burr-insider-trading.html

OpenInsider. (n.d.). SEC Form 4 insider trading screener. http://openinsider.com/

Ortega, C. (2023). Análisis de regresión: Qué es, tipos y cómo realizarlo. QuestionPro. https://www.questionpro.com/blog/es/analisis-de-regresion/

Probabilidad y Estadística. (2023). Regresión lineal múltiple. https://www.probabilidadyestadistica.net/regresion-lineal-multiple/

Programa único en finanzas cuantitativas y estrategias de inversión. (2024). BQuant. https://bquantfinance.com/

Rodriguez, J. (2017). Portfolio optimization with R: Part I. RPubs. https://rpubs.com/Joaquin_AR/242707

ScienceDirect. (2024). Pearson correlation – An overview. https://www.sciencedirect.com/topics/engineering/pearson-correlation

Scikit-learn. (2024). RFECV — scikit-learn 1.7.0 documentation. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFECV.html

XGBoosting. (2023). XGBoost robust to correlated input features (multicollinearity). https://xgboosting.com/xgboost-multicollinearity/

Annex

Google Collab Links:

CONGRESS TRADING AND BACKTESTING.ipynb

Final_Project_Insiders_Trading.ipynb

Spreadsheet link:

https://docs.google.com/spreadsheets/d/1JhFoUk8eNrlfuBcqJM4eE0oRRkZ4PsfA/edit?usp=sharing&ouid=103885713984821760785&rtpof=true&sd=true