MACHINE LEARNING APPLICATIONS IN PORTFOLIO OPTIMISATION

by

Shaan Patankar

This project centres on exploring the integration of machine learning algorithms in portfolio optimisation strategies, see the Jupyter Notebook for more info

Contents

1 Background and Motivation

Portfolio optimisation is a crucial aspect of modern finance, aiming to construct an investment portfolio that maximises returns while minimising risk. Traditional methods like Mean-Variance Optimisation (MVO) and the Capital Asset Pricing Model ($CAPM$) have been widely used for decades. However, the financial markets are complex and dynamic, and classical methods may not fully capture the intricate relationships between assets.

Machine learning algorithms have shown promise in various fields, including finance, for their ability to capture non-linear patterns and make data-driven decisions. This project explores the integration of machine learning algorithms in portfolio optimisation strategies to enhance portfolio performance and risk management.

1.1 Research Objectives

The main objectives of this project are as follows:

1. To compare the performance of machine learning-based portfolio optimisation strategies with classical MVO and $CAPM$ in terms of risk-adjusted returns.
2. To identify machine learning algorithms that are most effective in capturing non-linear relationships between asset returns in a portfolio context.
3. To investigate the impact of transaction costs, market liquidity, and model complexity on the feasibility of implementing machine learning-based strategies in real-world trading scenarios.

2 Data Retrieval and Preprocessing

Data retrieval and preprocessing ensure we have relevant and high-quality data for our project. This is vital given the financial implications of decisions based on this data. We'll utilise the *Yahoo Finance API* for fetching historical stock price data, ensuring we have a reliable and comprehensive data source.

For our analysis, we're more interested in the daily percentage change in stock prices (returns) rather than the actual prices. This gives us insights into the stocks performance and volatility. Note that raw financial data can have imperfections, as such, we'll address missing values and handle potential outliers by removing any empty rows to ensure data integrity and consistency in our analysis.

2.1 Exploratory Data Analysis (EDA)

After processing through all the data we can plot a stock price visualisation graph, a distribution of returns histogram, and a correlation heatmap (explained below). The stocks could be arbitrary, but for this project the stocks I've opted to look at are: $AAPL$ - Apple; $AMZN$ - Amazon; $GOOGL$ - Alphabet (*Google*); $MSFT$ - Microsoft; $META$ - Meta (*Facebook*).

Plotting stock price trends, studying the distribution of returns, and assess asset correlations provides insight and guides subsequent analyses.

1. The stock price visualisation plot showcases the historical stock prices for the five assets over the specified time frame. The x-axis represents time, while the y-axis indicates the stock price. The plot provides a clear view of the price trends and allows for easy comparison between the stock performances.

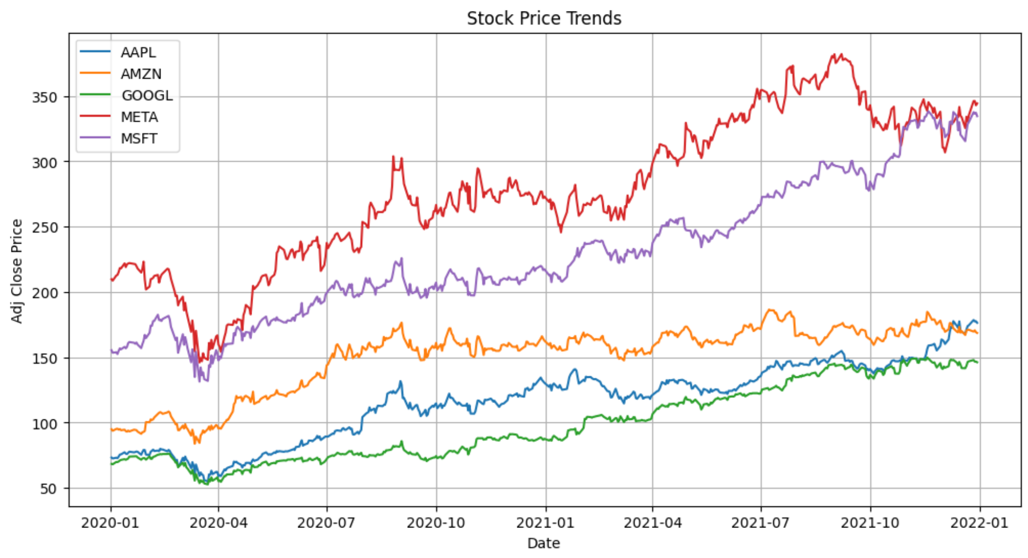


Figure 1: Stock price trend (*Yahoo Finance*)

2. The histogram visualises the distribution of daily returns for the stocks. It offers insights into the frequency of certain return ranges, helping to understand the volatility and typical performance of each asset.

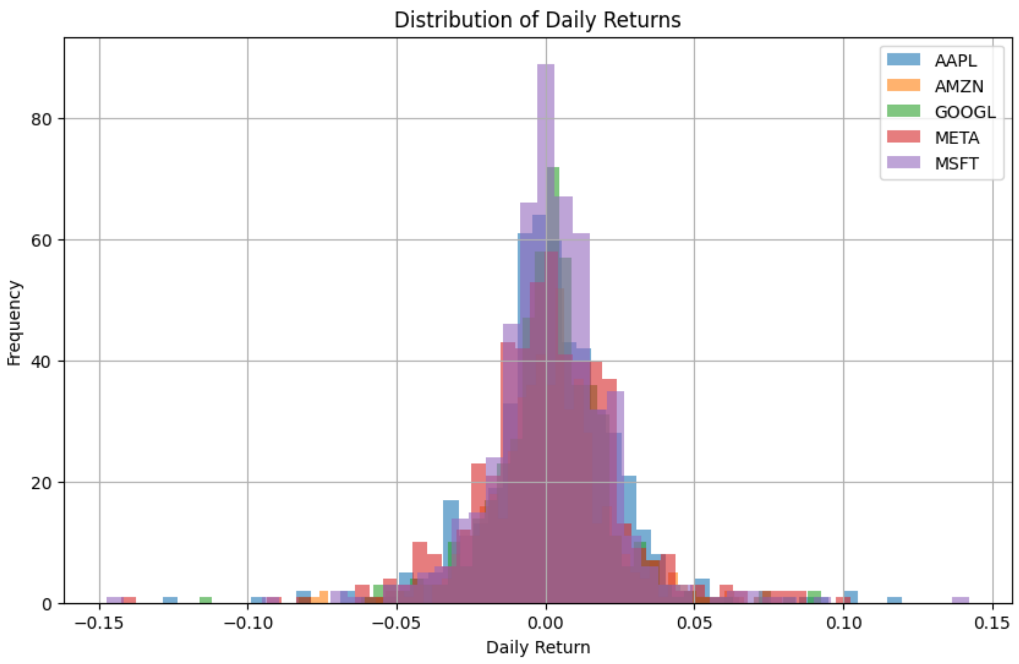


Figure 2: Histogram of daily returns for each stock

3. The heatmap displays the correlation coefficients between the daily returns of the stocks. A value close to 1 indicates a strong positive correlation, while a value close to -1 shows a strong negative correlation. This visualisation aids in understanding how different assets move in relation to each other.

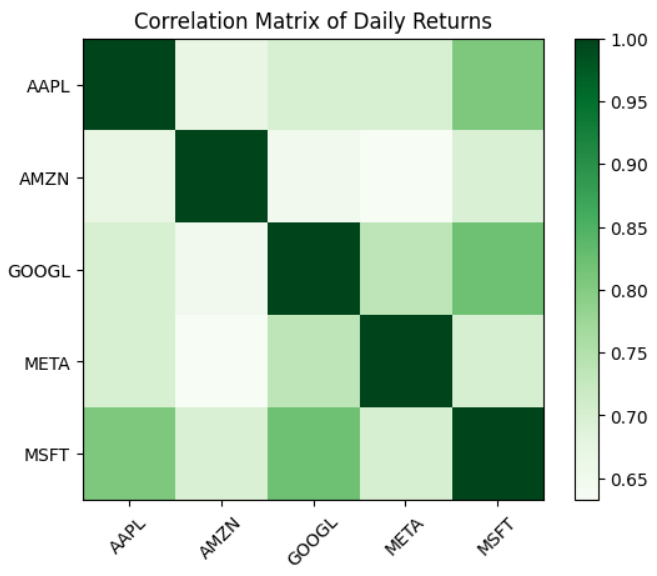


Figure 3: Heatmap correlation matrix of daily return

Python Code

Further details regarding the data collection and coding process can be found on the **Jupyter Notebook** file.

3 Capital Asset Pricing Model (CAPM)

Before delving into the application of machine learning, it is essential to understand the fundamentals of portfolio optimisation. The following sections will provide an overview for some of the traditional portfolio optimisation methods, such as Mean-Variance Optimisation (MVO) and the Capital Asset Pricing Model ($CAPM$).

The Capital Asset Pricing Model ($CAPM$) is a fundamental tool in finance that relates an asset's expected return to its systematic risk. We will explore the concepts of beta, the security market line (SML), and the risk-free rate in the context of $CAPM$. The $CAPM$ uses the formula,

$$E(R_i) = R_f + \beta_i(E(R_m) - R_f),$$

where $E(R_m)$ and $E(R_i)$ are the expected return of the market/asset i respectively, R_f is the risk-free rate, β_i is the beta of asset i - the volatility in relation to the market. Also, note that:

$$\beta_i = \frac{\text{Cov}(R_i, R_m)}{\text{Var}(R_m)},$$

where for discrete random variables X and Y with X_i and Y_i denoting the i^{th} sample of X and Y respectively, $\text{Var}(X) = \frac{S_{X_iX_i}}{N}$ and $\text{Cov}(X,Y) = \frac{S_{X_iY_i}}{N}$. Recall that $S_{X_iY_i} = \sum_{i=0}^{N} X_iY_i - N\bar{X}_i\bar{Y}_i$. β_i can thus be written as $\beta_i = \frac{S_{R_iR_m}}{S_{R_mR_m}}$, which is just the gradient of the regression line y on x. We can use RSS (or some other metric) to calculate how closely the data fits a linear model.

3.1 Security Market Line (SML)

The Security Market Line (SML) graph visually represents the relationship between the expected return of an asset and its systematic risk (beta). On the graph below, the y-axis represents the expected return of an asset, while the x-axis represents its beta. The slope of the SML is determined by the market risk premium. Assets that plot on the SML are considered fairly priced. If an asset plots above the SML, it indicates that it provides a higher return for its level of risk than the market portfolio, suggesting it might be undervalued. Conversely, assets below the SML might be overvalued, as they offer lower returns for their level of risk.

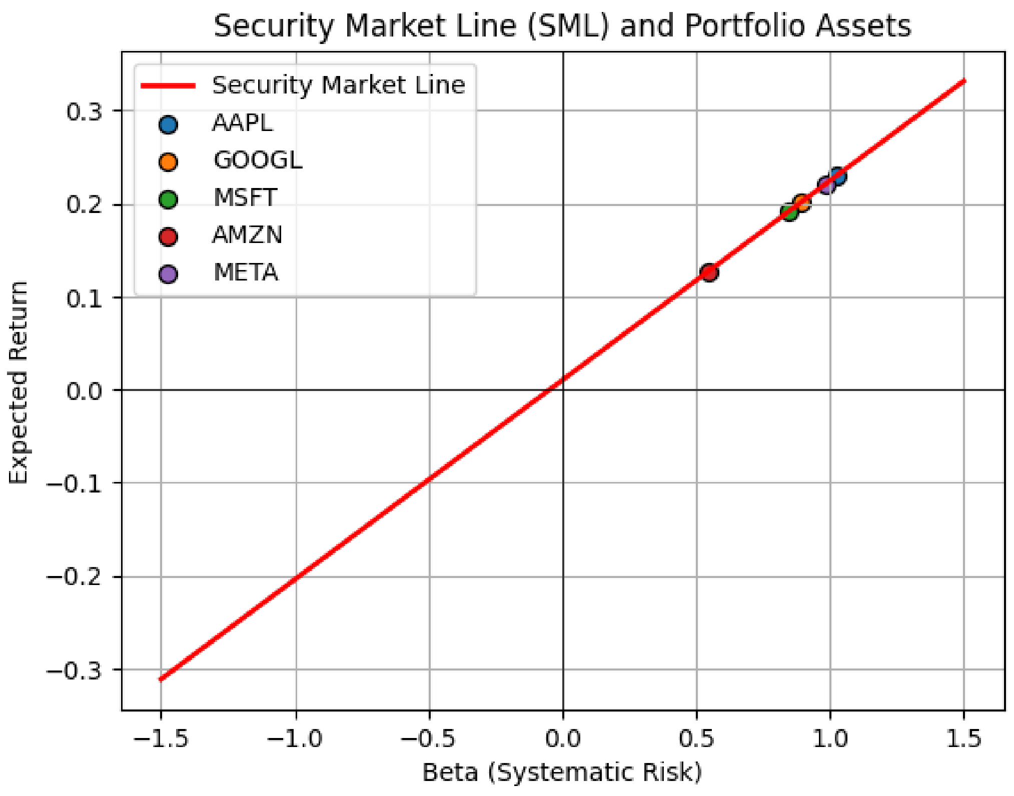


Figure 4: SML and portfolio assets , $S\&P$ 500 is used as the market benchmark

In this plot we've taken market data from each of the assets and the $S\&P$ 500 which acts as the market benchmark. By taking the `risk_free_rate` at 1% (details in the Notebook), calculating the `expected_stock_return` and `expected_market_return`, we can use this to find the `market_risk_premium` which is the slope of the SML (red line) using the $CAPM$ formula, `market_risk_premium = expected_market_return - risk_free_rate`. Multiplying the market risk premium by β_i and adding the risk free rate will give us the expected return for each of the assets (see below).

3.2 Asset Beta and Expected Returns

A beta value greater than 1 suggests that the asset's returns are more volatile than the market's. In Apple's case, its beta, 1.0264, suggests that it's roughly 2.64% more volatile than the market. The $CAPM$ suggests that, given Apple's beta and the prevailing market conditions, an investor should expect a return of approximately 22.99% for holding Apple, given the risk associated.

Google's beta (0.8918) suggests that it's about 10.82% less volatile than the market. For the risk associated with holding Google (which is slightly less than the market risk), an investor might expect a return of 20.11%. Microsoft is about 15.18% less volatile than the market having a beta of 0.8482. Given this beta value, we'd expect a return of 19.17% for holding it.

Amazon's low beta 0.5479 suggests it's approximately 45.21% less volatile than the market. This might seem surprising given Amazon's growth, but it could be influenced by time frame or some of the assumptions/limitations of $CAPM$, we will discuss some of these drawbacks later in the section. Amazon's expected return is the lowest among the assets (12.74%), reflecting its lower beta.

Meta's beta value of 0.9852 suggests its returns are roughly in line with the market, being just 1.48% less volatile, with expected returns similar to Apple of 22.11%.

Some of these expected returns are unreasonably high, and comparing with the actual historical returns for each asset we can see that the models predictions are not very accurate. Next, we will explore possible reasons as to why the model is predicting higher expected returns.

3.3 Assumptions and Limitations of CAPM

Here are some of the assumptions made when using the Capital Asset Pricing Model, these assumptions simplify our model - making it easier to use; however, this comes at the cost of making the model unrealistic/impractical in real-world scenarios.

1. Investors hold diversified portfolios: this implies that investors will only require a return for the systematic risk of their investments.
2. No transaction costs: investors can buy and sell securities without incurring any costs, which isn't the case in real-world scenarios.
3. Investors can lend and borrow at the risk-free rate: this is often not feasible in real markets.
4. All investors have the same expectations: investors are assumed to agree on the expected returns, volatilities, and correlations of all assets.

Here are a list of the limitations of the Capital Asset Pricing Model, these limitations are a direct result of some of the simplification/assumptions we have made when using the model. These limitations naturally suggests to us to use machine learning models which we will explore in the later sections.

1. Beta's predictive power: the beta of $CAPM$, which measures systematic risk, may not always be a complete measure of risk.
2. Market portfolio: the true market portfolio cannot be observed in reality, so proxies like the $S\&P$ 500 are used, which might not capture the entire market.
3. Static nature: $CAPM$ is a single-period model and doesn't capture changes over multiple periods.

Again, machine learning can help in capturing dynamic relationships, understanding non-linear patterns, and accounting for multiple factors that might affect stock returns beyond just the market return.

4 Mean-Variance Optimisation (MVO)

Mean-Variance Optimisation (MVO) is a classical portfolio optimisation technique that seeks to maximise the portfolio's expected return while minimising its variance or risk. There is always a trade-off between expected returns and risk, and our goal is aiming to achieve the most optimal balance. We will discuss the underlying assumptions and limitations of MVO.

The key principles of MVO to remember are the risk and return trade-off - every increase in potential return comes with an increase in risk. MVO uses this principle to find portfolios that offer the maximum expected return for a given level of risk. The second principle involves diversification, by combining assets that aren't perfectly correlated, MVO will help in constructing portfolios that reduce unsystematic risk.

Given a set of assets with known expected returns, the expected return of a portfolio is the weighted sum of the expected returns of the individual assets:

$$E(R_p) = \sum_{i=1}^{N} w_i E(R_i).$$

$E(R_p)$ and $E(R_i)$ are the expected returns of the portfolio and asset i respectively, and w_i is the weight of asset i in the portfolio. Note that there are a total of N assets in the portfolio, and we require that the weights sum to 1, that is $\sum_{i=1}^{N} w_i = 1$. We can compute the variance of the portfolio based on the variances of the individual assets, their weights, and correlation, ρ_{ij}, between them:

$$\text{Var}(R_p) = \sum_{i=1}^{N}\sum_{j=1}^{N} w_i w_j \text{Cov}(E(R_i), E(R_j)).$$

$\sigma_i = \sqrt{\text{Var}(R_i)}$ is the standard deviation of asset i and $-1 \leq \rho_{ij} \leq 1$ is the correlation coefficient defined as, $\rho_{ij} = \frac{\text{Cov}(r_i, r_j)}{\sigma_i \sigma_j} = \frac{S_{r_i r_j}}{\sqrt{S_{r_i r_i} S_{r_j r_j}}}$. This result follows just from using the definition of the product moment correlation coefficient ($PMCC$), where we use $r_i := E(R_i)$ to compact notation.

4.1 Assumptions and Limitations of MVO

These are some of the assumptions made when using Mean-Variance Optimisation, these assumptions simplify our model - making it easier to use; however, this comes at the cost of making the model unrealistic/impractical in real-world scenarios.

1. Investors are rational and avoid risk: investors aim to maximise their utility, which can be achieved with a high expected return and low portfolio risk.

2. Investments are limited to a set of risky assets: MVO doesn't consider the inclusion of risk-free assets.

3. Returns are (jointly) normally distributed: this is often not the case in real financial markets, leading to potential underestimation of risk.

4. Investors have access to the same information: all investors will estimate the expected returns, standard deviations, and correlation coefficients in the same way.

Similar to the $CAPM$, here are a list of the limitations of MVO, these limitations are a direct result of some of the simplification/assumptions we have made when using the model. These limitations naturally suggests to us to use machine learning models which we will explore in the later sections.

1. Estimation errors: MVO is highly sensitive to changes in input parameters, which can lead to vastly different portfolios.
2. Normal distribution assumption: financial returns are often not normally distributed. They can have fat tails, which means there's a higher probability of extreme returns.
3. Static model: MVO doesn't account for changes in the market over time.

Machine learning techniques can address some of these limitations. For example, they can handle non-linear relationships, adapt to changing market conditions, and account for non-normal distributions of returns.

That said, machine learning while offering advanced capabilities, it's not without its' challenges. ML-based models can be data-hungry, require significant computational power, and can sometimes act as black boxes, making their decisions hard to interpret. We will explore these sort of limitations in later chapters.

4.2 Sharpe Ratio

The Sharpe Ratio

The Sharpe ratio is a measure used to understand the average return earned in excess of the risk-free rate per unit of volatility or total risk. It provides a tool to assess the risk-adjusted performance of an investment or a portfolio.

The formula for the Sharpe ratio is:

$$\text{Sharpe Ratio} = \frac{E(R_p) - R_f}{\sigma_p}$$

$E(R_p)$ and R_f take the same form as before and $\sigma_p = \sqrt{\text{Var}(R_p)}$ is the standard deviation (volatility) of returns of the portfolio; representing its' total risk.

- A higher Sharpe ratio indicates a better risk-adjusted performance of the investment or portfolio. Essentially, for every unit of risk (volatility) taken, the investment returns a certain excess amount of return over the risk-free rate.
- A positive Sharpe ratio means the investment return is higher than the risk-free rate, whereas a negative Sharpe ratio indicates the investment return is below the risk-free rate.
- A Sharpe ratio of zero suggests that the investment has a return exactly equal to the risk-free rate.

For our purposes, the Sharpe ratio is used as a criterion in portfolio optimisation, where the goal is to maximise the Sharpe ratio, thereby seeking the portfolio with the best risk-adjusted return.

4.3 Efficient Frontier

The efficient frontier visualisation helps investors understand the risk-return trade-off and identify portfolios that maximise returns for a given risk level. When selecting portfolios, those on the efficient frontier curve are preferable, with the red-starred portfolio being the most optimal, and the green-starred being the safest in terms of risk-adjusted returns.

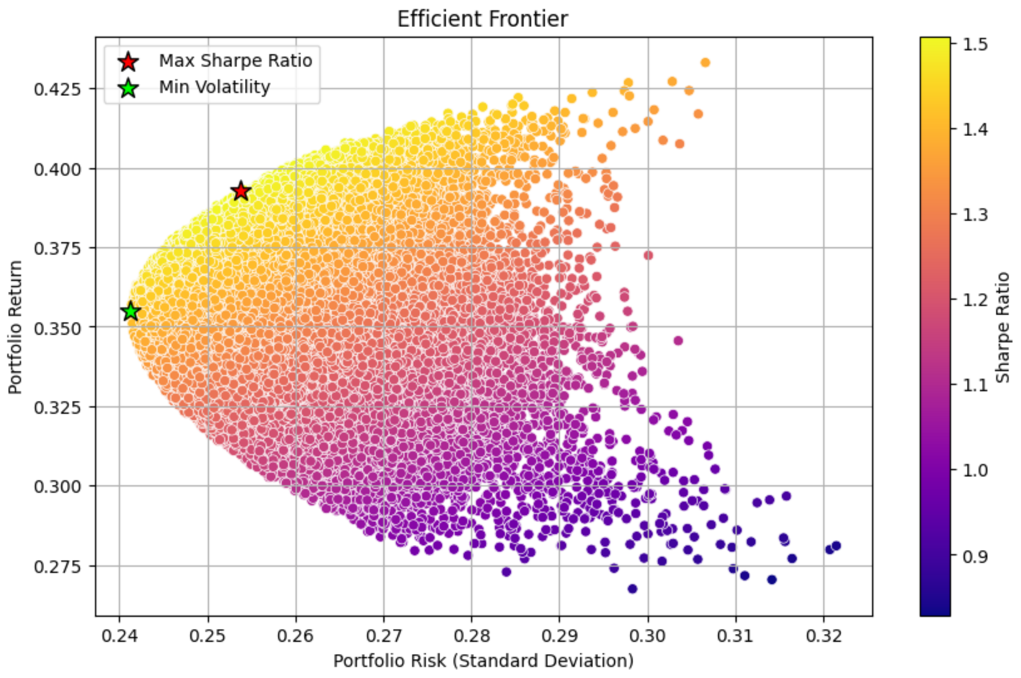


Figure 5: Efficient frontier with highlighted optimal portfolios

Each point on the scatter plot represents a potential portfolio composed of the assets considered. The position of a point is determined by its risk (standard deviation on the x-axis) and expected return (on the y-axis).

The colour of each point corresponds to its Sharpe ratio, which measures the risk-adjusted return of the portfolio. A higher Sharpe ratio suggests a more favorable risk-to-reward trade-off. In this visualisation, the color gradient is shown using the plasma colormap, with warmer colours indicating higher Sharpe ratios.

A red star marks the portfolio with the maximum Sharpe ratio. It's the most desirable portfolio because it offers the highest return for a given level of risk. This point is often referred to as the *tangency portfolio* since it's where the Capital Market Line or CML (a line drawn from the risk-free rate tangent to the efficient frontier) touches the frontier.

Conversely, a lime star represents the portfolio with the minimum volatility (or risk). It's the safest portfolio because it has the lowest standard deviation, and thus, the lowest risk.

Lastly, the curve formed by the upper boundary of the scatter plot is what represents the efficient frontier. This boundary denotes the set of optimal portfolios that offer the highest expected return for a defined level of risk. Portfolios below this boundary are sub-optimal because they either provide less return for the same level of risk or have more risk for the same expected return.

4.4 Optimised Portfolio Analysis

Max Sharpe Ratio vs Min Risk

The table below gives asset allocations for portfolios corresponding to the max Sharpe ratio and min risk. The max Sharpe ratio portfolio aims to give the highest risk-adjusted return, and the min risk portfolio, as the name suggests, aims to mitigate risk.

Stock Ticker	Weight (Max Sharpe Ratio)	Weight (Min Risk)
$AAPL$	0.214641	0.003440
$GOOGL$	0.025050	0.339532
$MSFT$	0.363010	0.350740
$AMZN$	0.000673	0.007854
$META$	0.396626	0.298435

As for the max Sharpe ratio portfolio, $META$ has the highest allocation of 39.66%, followed by $MSFT$ (36.3%) and then $AAPL$ (21.46%). $GOOGL$ and $AMZN$ on the other hand, have much lower allocations of 2.51% and 0.07% respectively.

For the min risk portfolio, $MSFT$ has the highest allocation at 35.07%, closely followed by $GOOGL$ (33.95%) and $META$ (29.84%). $AAPL$ and $AMZN$ have minimal allocations, at 0.34% and 0.78% respectively.

The asset allocation between the portfolios is quite different, $META$ is heavily favoured in the max Sharpe ratio portfolio, but has a reduced weight in the min risk portfolio. Conversely, $GOOGL$, which has a small presence in the max Sharpe ratio portfolio, has a significant weight in min risk. $MSFT$ maintains a strong presence in both portfolios. $AMZN$ is minimally represented in both portfolios, suggesting that, it might not be contributing substantially to either maximising the Sharpe ratio or minimising risk.

Metric	Max Sharpe Ratio	Min Risk
Portfolio Return	0.396189	0.353511
Portfolio Risk	0.256094	0.241258
Sharpe Ratio	1.546296	1.464490

The max Sharpe ratio portfolio has a ratio of 1.5463, the portfolio return is 39.62%, while the standard deviation is 25.61%. Comparing this with the min risk portfolio which has a risk of 24.13%, a Sharpe ratio of 1.4645 - just 0.0818 difference, and a return of 35.35%. It is clear that the max Sharpe ratio portfolio offers a higher expected return. This is consistent with the risk-return trade-off; generally, portfolios with higher expected returns are also expected to exhibit higher volatility (risk).

Similarly, the max Sharpe ratio portfolio has a slightly higher Sharpe ratio than the min risk portfolio, suggesting that it provides a better risk-adjusted return. Lastly, for risk, as the name suggests, the min risk portfolio has a lower volatility compared to the max Sharpe ratio portfolio, that said the difference in risk (1.48%) is not that substantial.

In summary, if maximising returns for a given level of risk is the primary objective, then the max Sharpe ratio portfolio would be preferred. However, if minimising risk is more crucial, even at the expense of higher returns, then min risk would be more suitable.

Using the Sequential Least Squares Quadratic Programming ($SLSQP$) method, we are able to deduce the optimal allocation of capital across the five stocks to maximise the Sharpe ratio. More details in the **Jupyter Notebook**.

Apple accounts for approximately 22.07% of the portfolio. Where as Google makes up 4.97% of the portfolio. $SLSQP$ allocates roughly 34.78% of the portfolio to Microsoft and 34.78% to Meta (*Facebook*). This means, based on the data and optimisation objective - Amazon's optimal weight is 0.00% meaning it's not ideal to hold Amazon in the portfolio to achieve the maximum Sharpe ratio.

These weights suggest that the model finds the most value (in terms of risk-adjusted returns) in allocating capital to Meta and Microsoft. Apple also gets a significant allocation, while Google gets a smaller portion; and Amazon doesn't get any allocation under these parameters.

5 Machine Learning in Portfolio Optimisation

Portfolio optimisation has traditionally relied on mathematical models and techniques, such as the Capital Asset Pricing Model and Mean-Variance Optimisation, to determine the best allocation of assets in a portfolio. These traditional methods are based on certain assumptions about market behavior, which may not always hold true. This is where machine learning offers a fresh perspective.

Machine learning, with its ability to model and predict complex, non-linear relationships, provides tools to capture patterns in data that might be overlooked by traditional models. By analysing vast amounts of historical data, machine learning algorithms can identify intricate relationships between assets, potentially leading to better portfolio performance.

However, like all models, machine learning-based approaches have their limitations. They require large datasets to train on, can sometimes be opaque or hard to interpret (a challenge known as the *black box* problem), and are sensitive to the quality of the input data.

5.1 Machine Learning Algorithms Overview

The three main machine learning models we'll be taking a look at are:

1. Decision Trees: they map out decisions and their possible consequences. In portfolio management, they can be used to decide on asset allocations based on certain criteria.
2. Random Forests: an ensemble of Decision Trees, often trained with the bagging method. They can reduce overfitting compared to a single Decision Tree and give better accuracy.
3. Neural Networks: comprise layers of interconnected nodes or neurons. They are especially good at capturing non-linear relationships in data, which can be beneficial for predicting stock prices. (This model will be included in the **Jupyter Notebook**).

To evaluate and compare the potential performance of different asset combinations, we'll simulate multiple portfolios with random asset allocations. This simulation will give us insights into the possible returns and volatilities we can expect from different portfolio structures. We'll simulate portfolios using historical data, and for each portfolio, we'll calculate its expected return and volatility based on the random asset weights. This will provide further insight into the risk-reward trade-off for our simulations. Then, in the following chapters, we will use the data gathered to train our ML models.

5.2 Decision Tree

Decision Trees split the data into subsets based on certain decision rules - mapping out these decisions and their possible consequences. They are intuitive and easy to visualise but can sometimes overfit to the training data. A Decision Tree can be employed to predict the future return of a stock or a portfolio based on historical data. The tree makes decisions based on various factors, such as past returns, trading volumes, and other relevant financial indicators.

For our portfolio optimisation task, it's looking at relationships between asset weights, volatility, and returns to predict portfolio returns. The metrics we've computed provide insights into how well the Decision Tree has learned these relationships:

- MAE (Mean Absolute Error): this metric tells us, on average, how much our predictions deviate from the actual returns. A lower MAE is better.
- $RMSE$ (Root Mean Squared Error): this metric gives more weight to larger errors, making it more sensitive to outliers than MAE. A lower $RMSE$ indicates a better fit to the data. Note MSE is just the (positive) square root of the $RMSE$. We will explore MSE later.
- R-squared: represents the proportion of the variance in the dependent variable that is predictable from the independent variables. A value closer to 1 indicates that the model explains a higher proportion of the variance.

The process of hyperparameter tuning ensures that you're not just fitting your model to the training data but that it can also generalise well to unseen data. The hyperparameters are set prior to training a machine learning model. They are not learned from the data but can significantly impact model performance. For example, certain hyperparameters can affect the speed and efficiency of training and can influence the complexity of the model leading to optimal model complexity.

In Decision Trees, the depth of the tree (`max_depth`) can determine how well the model fits to the data. Too deep, and it may overfit; too shallow, and it may underfit. Some hyperparameters introduce regularisation (like the `min_samples_leaf` in Decision Trees), which can prevent overfitting by adding constraints to the model.

k-fold cross-validation involves partitioning the original training dataset into k equal subsets or folds. Then, a model is trained using $k-1$ of the folds and validated on the remaining fold. This process is repeated k times, with each fold serving as the validation set exactly once. The performance metric is then averaged over all k trials to provide a more stable estimate of model performance. By integrating k-fold cross-validation in our Decision Tree training process, we ensure a more robust assessment of the model's capability.

In grid search, for each hyperparameter, we specify a range of values. The algorithm then systematically trains the model for every combination of hyperparameters. This ensures that we explore a wide range of configurations to find the best one.

To summarise, for each combination of hyperparameters, k-fold cross-validation trains the model on different subsets (or folds) of the data and validates it on the remaining data. This process is repeated k times. This provides a more stable and robust estimate of the model's performance, ensuring that the chosen hyperparameters generalise well across different subsets of data.

Decision Tree Model Analysis

This is an explaination for the physical interpretations for each of the metrics calculated using the Decision Tree machine learning algorithm:

Model	MAE	RMSE	R-squared	MSE
Decision Tree	0.001363	0.001804	0.967676	0.000003

1. MAE (Mean Absolute Error): $\sim 1.36 \times 10^{-3}$ - this metric tells us that, on average, the model's predictions are off by approximately $\sim 1.36 \times 10^{-3}$. Given that this is a small value, it indicates the models predictions are quite close to the actual returns.

2. $RMSE$ (Root Mean Squared Error): $\sim 1.80 \times 10^{-3}$ - $RMSE$ gives more weight to larger errors. This means if you have more outliers or predictions that were way off, this value will increase. The $RMSE$ being slightly higher than the MAE suggests that there might be a few predictions where the model didn't perform as well. Still, given the small value, the model is doing a good job overall.

3. R-squared: $\sim 9.68 \times 10^{-1}$ - this is a proportion between $0 \leq R^2 \leq 1$, and it tells you the percentage of the variance in the dependent variable (portfolio returns) that the independent variables explain. An R-squared value of $\sim 9.68 \times 10^{-1}$ means that $\sim 96.8\%$ of the variation in portfolio returns can be explained by the model. This is a strong R-squared value, indicating a high level of predictive power.

4. MSE (Mean Squared Error): $\sim 3.00 \times 10^{-6}$ - this metric represents the average squared difference between the observed actual out-turn values and the values predicted by the model. The small MSE value indicates that the model's predicitve capabilities are fairly accurate.

5.3 Random Forest

Random Forests are an ensemble learning method that builds upon the Decision Tree algorithm. Instead of relying on a single Decision Tree, Random Forests consist of multiple Decision Trees. A Random Forest uses a collection (or forest) of Decision Trees to make predictions, each tree is trained on a random subset of the data with replacement and makes its own predictions. This technique called bootstrapping, and means that some samples may be used multiple times in a single tree, while others may not be used at all. The Random Forest then aggregates these predictions to produce a final result which is typically an average of the predictions from all trees in the forest. This ensemble approach generally improves performance and reduces overfitting providing a more robust model compared to a single Decision Tree.

The benefits of using Random Forests in portfolio optimisation include the ability to capture complex, non-linear relationships between assets, as well as providing a measure of feature importance. This can be particularly useful in identifying which financial indicators or asset relationships are most influential in determining portfolio returns.

As with the Decision Tree model, we will evaluate the Random Forest's performance using the following metrics:

- MAE: indicates the average error of the Random Forest model's predictions. Comparing with Decision Tree's MAE can give insight into the benefit of using ensemble.

- RMSE: helps understand the error distribution and the impact of outliers on the Random Forest model's predictions. (We will cover MSE later in the section).
- R-squared: a higher R-squared value would suggest that the ensemble of trees in the Random Forest can explain more variance in the portfolio returns.

`RandomizedSearchCV` is used for hyperparameter tuning, this is different from `GridSearchCV` which was used in the Decision Tree model. The key difference being, traditional grid search methods evaluate every possible combination of hyperparameters, which can be computationally expensive. `RandomizedSearchCV`, on the other hand, samples a fixed number of hyperparameter combinations from the specified distributions. This approach is faster and can lead to better results, especially when the number of hyperparameters is large (as is the case).

Random Forest Model Analysis

This is an explaination for the physical interpretations for each of the metrics calculated using the Random Forest machine learning algorithm:

Model	MAE	RMSE	R-squared	MSE
Random Forest	0.000536	0.000763	0.994220	5.820780×10^{-7}

1. MAE (Mean Absolute Error): $\sim 5.36 \times 10^{-4}$ - the MAE for the Random Forest model is approximately $\sim 5.36 \times 10^{-4}$. This metric conveys that, on average, the model's predictions deviate by about $\sim 5.36 \times 10^{-4}$ from the actual values. Given the diminutive nature of this value, it suggests that the Random Forest model's predictions are more precise compared to the Decision Tree model, emphasising its heightened accuracy in forecasting returns.
2. $RMSE$ (Root Mean Squared Error): $\sim 7.63 \times 10^{-4}$ - the $RMSE$ value is approximately $\sim 7.63 \times 10^{-4}$. Like with the Decision Tree, the $RMSE$ metric prioritises larger errors more. The fact that this value is slightly larger than the MAE, though still quite small, indicates there might be occasional predictions where the model deviated a bit more. However, in general, the model's performance is quite good.
3. R-squared: $\sim 9.94 \times 10^{-1}$ - again, this metric represents the proportion of variance in the dependent variable that can be attributed to the independent variables. An R-squared value of $\sim 9.94 \times 10^{-1}$ denotes that around $\sim 99.4\%$ of the variation in portfolio returns is elucidated by the model. This is an stronger R-squared value than the Decision Tree, showcasing the Random Forest's superior predictive prowess.
4. MSE (Mean Squared Error): $\sim 5.82 \times 10^{-7}$ - again, the MSE indicates the average squared discrepancy between the observed actual outcomes and the values the model predicted. For the Random Forest, this value is about $\sim 5.82 \times 10^{-7}$, which is significantly lower than that of the Decision Tree. This underlines the Random Forest model's enhanced precision and suggests that the model has a strong capacity to predict portfolio returns accurately.

In summary, based on the provided metrics, the Random Forest model *appears* to have a stronger predictive capability than the Decision Tree, with higher accuracy and precision in its predictions, we'll explore this further in the next section.

6 Evaluating Machine Learning Model Performance

The Random Forest model has a lower MAE compared to Decision Tree, suggesting it makes, on average, smaller prediction errors compared to the Decision Tree model. Again, the Random Forest model has a lower $RMSE$, indicating it has fewer large prediction errors compared to the Decision Tree model. The R^2 value, often called the *coefficient of determination*, is larger for the Random Forest model, meaning it can explain a larger portion of the variance in the returns compared to the Decision Tree model. Lastly, Random Forest outperforms the Decision Tree model with a significantly lower MSE indicating better predictive accuracy. Based on these metrics and the rationale behind each of them, it is clear that the Random Forest model consistently outperforms the Decision Tree model.

However, it's essential to consider other factors like model interpretability, training time, and computational cost. While Random Forests might give better predictions, Decision Trees are simpler and more interpretable.

In summary, the Random Forest model, which utilises an ensemble of Decision Trees and aggregates their predictions, offers a more accurate and reliable prediction for portfolio returns in this context.

In the following sections, we will evaluate the performance of the ML models by comparing the predicted against actual returns data, and investigate the impact of external factors on the feasibility of implementing ML-based strategies in a real-world scenario.

6.1 Impact of External Factors on ML Strategies

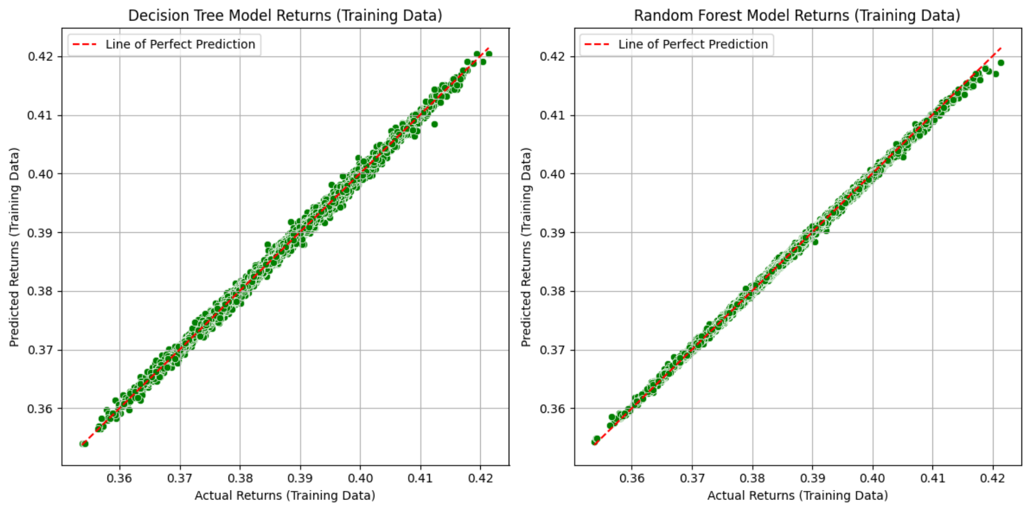
Feasibility of ML-based Models

We will briefly outline and explore how external factors affect the performance and feasibility of machine learning-based portfolio optimisation strategies.

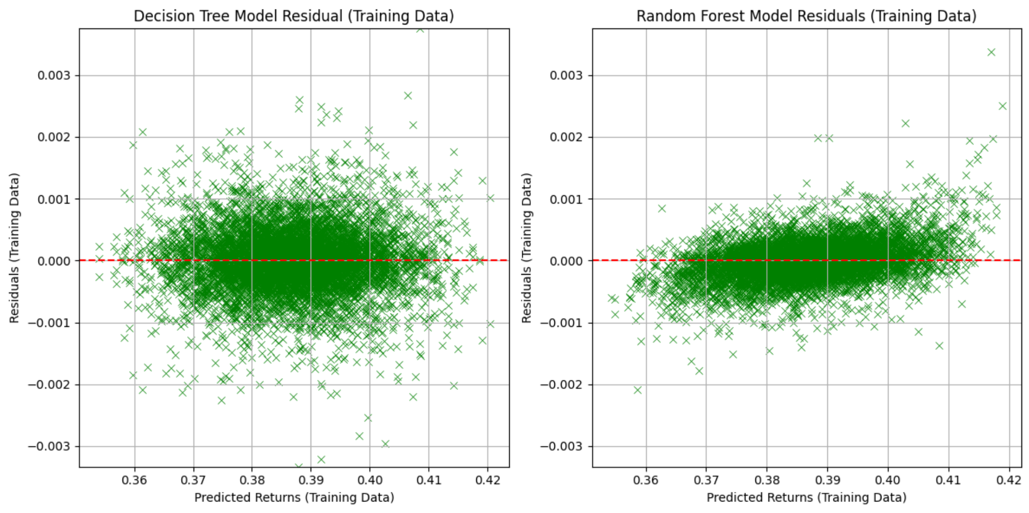
- Transaction costs: high-frequency trading strategies based on machine learning can incur significant transaction costs, which can erode profits. By simulating a trading strategy with and without transaction costs, you can determine the impact on net returns.
- Market liquidity: machine learning models might identify arbitrage opportunities in illiquid stocks. However, due to low liquidity, executing large trades can be challenging. Assessing the liquidity of assets in the portfolio and comparing the model's performance on liquid verse illiquid assets can provide insights.
- Model complexity: while deep learning models can capture complex patterns, they require extensive computational resources and can be prone to overfitting. Comparing the performance of a deep neural network (DNN) with a simpler model like linear regression or a Decision Tree can shed light on the trade-off between model complexity and performance.

It's essential to consider transaction costs, market liquidity, and model complexity, as these can influence the feasibility and effectiveness of machine learning models in a real-world scenario.

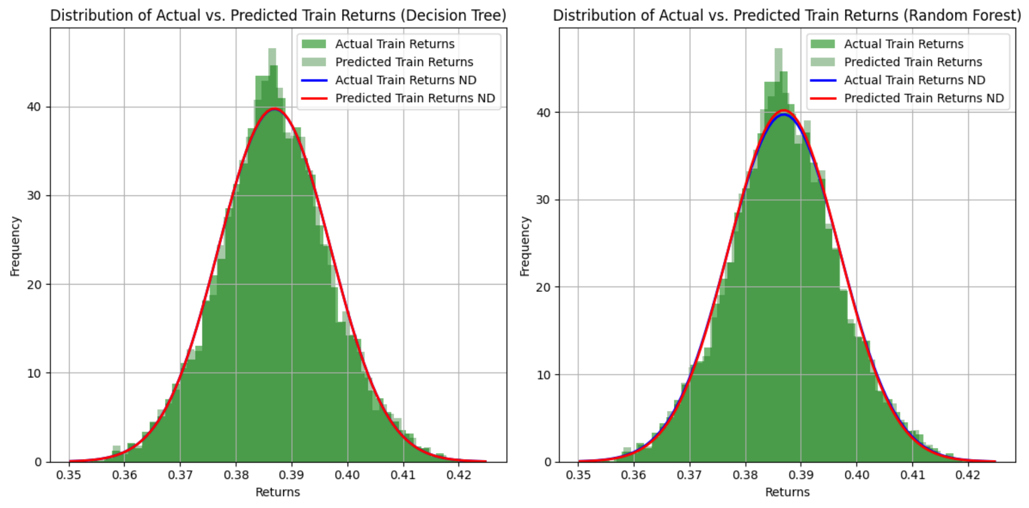
6.2 Training Data Analysis



(a) Predicted vs Actual Returns for the Training Dataset



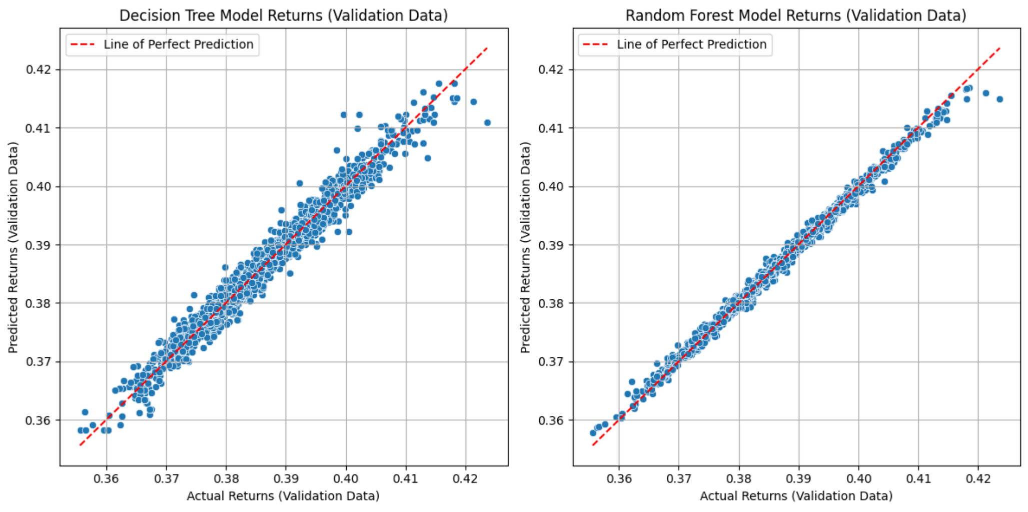
(b) Residuals for DT and RF Training Dataset



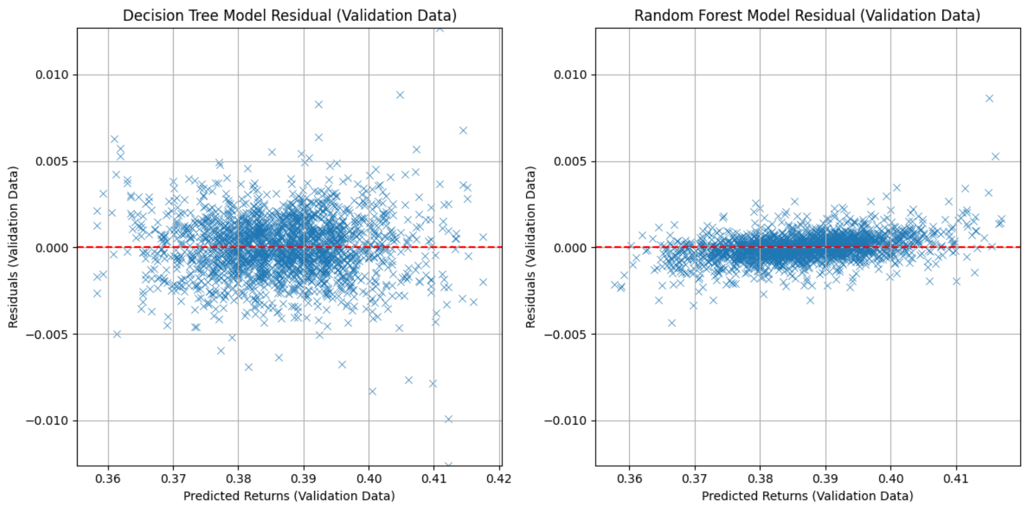
(c) Distribution for Actual vs Predicted Returns for the Training Dataset

Figure 6: Analysis of the Training Dataset for Decision Tree and Random Forest

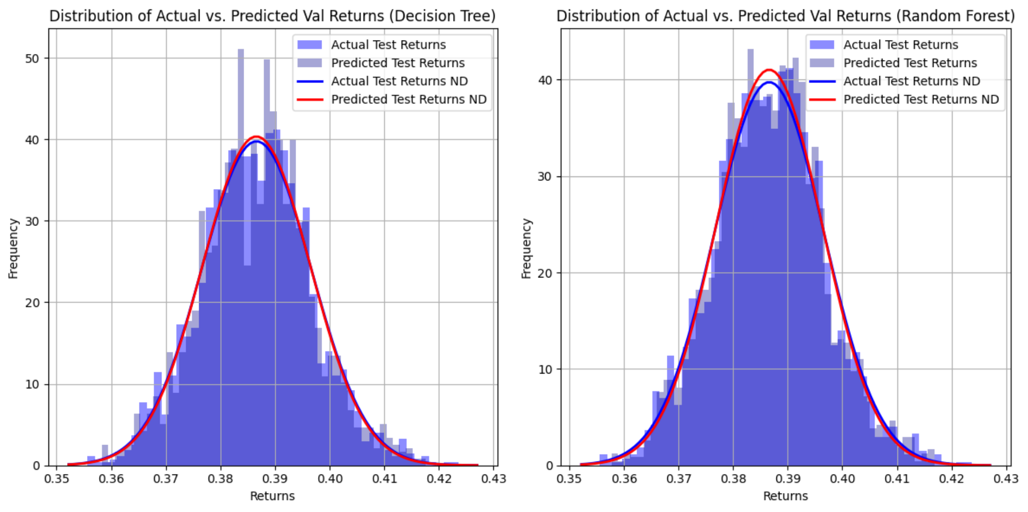
6.3 Validation Set Analysis



(a) Predicted vs Actual Returns for the Validation Dataset



(b) Residuals for DT and RF Validation Dataset



(c) Distribution for Actual vs Predicted Returns for the Validation Dataset

Figure 7: Analysis of the Validation Dataset for Decision Tree and Random Forest

7 Future Research

While this project explored traditional machine learning models like Decision Trees and Random Forests, the application of deep learning methods, such as recurrent neural networks (RNNs) or long short-term memory networks ($LSTM$s), could be valuable. These models have proven adept at capturing complex temporal patterns in data, which may be particularly relevant for financial time series.

Machine learning models such as Neural Networks are complex structures inspired by human brain function - they consist of layers of interconnected nodes or neurons. In fact, while we haven't talked much about Neural Networks, a ML model has been implemented in the Jupyter Notebook

Incorporating alternative data, such as social media sentiment, macroeconomic indicators, or news analytics, could provide richer models. Understanding how these diverse data streams influence asset prices and portfolio performance can unveil novel investment strategies.

Lastly, rather than treating classical and machine learning models as separate entities, a hybrid approach that synergistically combines the strengths of methods like $CAPM$ or MVO with machine learning could yield innovative strategies. An example might be, instead of simulating portfolios with random asset weighting for Random Forests, we can use the optimised asset weights which were deduced using MVO.

8 Conclusion

This project embarked on an exploratory journey to understand the merging of portfolio optimisation and machine learning techniques, aiming to decipher whether machine learning offers a tangible advantage over traditional methods.

Our analysis revealed that machine learning algorithms, particularly Random Forests, demonstrated a notable capability in capturing the non-linear relationships inherent in financial datasets. These models were able to unearth intricate patterns and dependencies which classical models might overlook.

Comparatively, while classical methods like Mean-Variance Optimisation and the Capital Asset Pricing Model offer a robust theoretical foundation, they come with underlying assumptions that may not always hold true in the chaotic world of financial markets. Whereas, machine learning models, with their data-driven approach, showed potential in navigating this chaos.

However, the integration of machine learning into portfolio management is not without its challenges. Considerations like transaction costs, market liquidity, and model complexity introduce layers of difficulty. Moreover, the black-box nature of many machine learning algorithms necessitates rigorous validation to ensure that they don't merely overfit historical data but genuinely offer predictive power.

This project underscores that machine learning, when wielded judiciously, can be a potent tool; and how it's essential to approach this intersection of finance and technology with a balanced perspective, leveraging the strengths of both traditional financial models and advanced machine learning algorithms to achieve optimal portfolio performance.

References

[1] David H. Bailey, Jonathan M. Borwein, Marcos López de Prado, and Qiji J. Zhu. Pseudo-mathematics and financial charlatanism: The effects of backtest overfitting on out-of-sample performance. In *Notices of the AMS*, volume 61, pages 458–471, 2014.

[2] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. In *IEEE Transactions on Neural Networks*, volume 5, pages 157–166, 1994.

[3] Zvi Bodie, Alex Kane, and Alan J. Marcus. *Investments.* McGraw-Hill/Irwin, 2005.

[4] John Y. Campbell, Andrew W. Lo, and A. Craig MacKinlay. *The Econometrics of Financial Markets.* Princeton University Press, 1997.

[5] Eugene F. Fama and Kenneth R. French. The cross-section of expected stock returns. *The Journal of Finance*, 47(2):427–465, 1992.

[6] Shihao Gu, Bryan Xiong, Hongyang Zhang, and Zhening Zhu. A machine learning approach to portfolio optimization. In *Proceedings of the 2020 Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[7] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. The elements of statistical learning: Data mining, inference, and prediction. *Springer Series in Statistics*, 2009.

[8] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: with Applications in R.* Springer, 2013.

[9] Harry Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952.

[10] John Moody, Lizhen Wu, Yuansong Liao, and Matthew Saffell. Performance functions and reinforcement learning for trading systems and portfolios. In *Journal of Forecasting*, volume 17, pages 441–470, 1998.

[11] William F. Sharpe. Capital asset prices: A theory of market equilibrium under conditions of risk. *The Journal of Finance*, 19(3):425–442, 1964.

[12] Ruey S. Tsay. Analysis of financial time series. *Wiley Series in Probability and Statistics*, 2005.