**PORTFOLIO**

**Use of Lasso Model for Predictions:**

The Lasso model helps in predicting future returns of DAX30 index assets. Its selected for its ability to handle multicollinearity and perform variable selection and leading to more reliable predictions. This is chosen because in previous part we got good results for lasso model than others.



**Numerical Conversion of Predicted Returns:**

Converting predicted returns to numeric ensures compatibility with mathematical operations and optimization algorithms. This step is necessary to avoid data type mismatches that could lead to errors in subsequent analyses. (Feature Engineering)



The notation num [1:5582] indicates that this is a numeric vector with 5582 elements. The first few values of predicted returns are shown: -0.02303, -0.01734, -0.00875, -0.00218, 0.03357 and so on.

These predicted returns are likely result of a predictive model such as Lasso model mentioned in provided code which has been applied to a set of features (predictors) from DAX\_temp dataset. This output would be used in context of portfolio optimization and to inform decisions about asset allocation.

**Creation of Dataframe for Predicted Returns:**

Combining predicted returns with their corresponding stock tickers in a dataframe assists easier tracking and manipulation of data. This organizational step is crucial for managing information used in portfolio optimization.





ticker: This column is populated with values from DAX\_temp$ticker which likely contains ticker symbols of stocks that are part of the DAX 30 index.

predicted\_returns: This column contains numerical values that are presumably predicted returns for each ticker symbol as computed by a predictive model like a Lasso regression model in out requirements.

**Simulation and Covariance Matrix Calculation:**

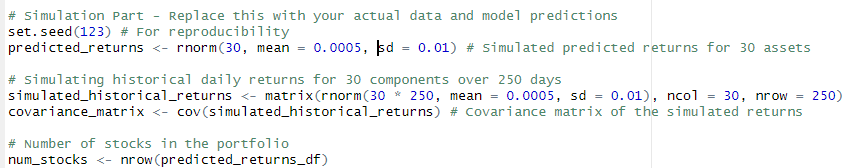
Copying returns and calculating covariance matrix are placeholders for incorporating real historical data. The covariance matrix is vital for understanding relationships between asset returns and particularly how they move together and which directly impacts portfolio risk.





In portfolio covariance matrix is used to measure how returns on assets move together. When you have a portfolio of assets covariances variances when considering a single asset are crucial in understanding overall risk of portfolio. In context of code and this could be part of process to construct an optimal portfolio that minimizes risk variance for given level of expected return or to maximizes return for a given level of risk and among other possible optimization goals.

By inputting predicted\_returns into a matrix and then calculating covariance matrix you are preparing to use variance of predicted returns in optimization algorithm like a mean-variance optimization model. If you had more than one asset covariance matrix would show how each asset's returns relate to returns of every other asset but with a single column we only get variance of that one set of returns.







set.seed(123): This function sets the seed of R's random number generator,which ensures that results of random number generation are reproducible. If we run code multiple times it will give same random numbers each time by which is essential for reproducibility in simulations.

predicted\_returns <- rnorm(30, mean = 0.0005, sd = 0.01): This line generates a vector of 30 normally distributed random numbers by which represent simulated predicted returns for 30 assets. The mean return set to 0.0005 (or 0.05%) with standard deviation of 0.01 or 1%, reflecting a common scale for daily returns in finance.

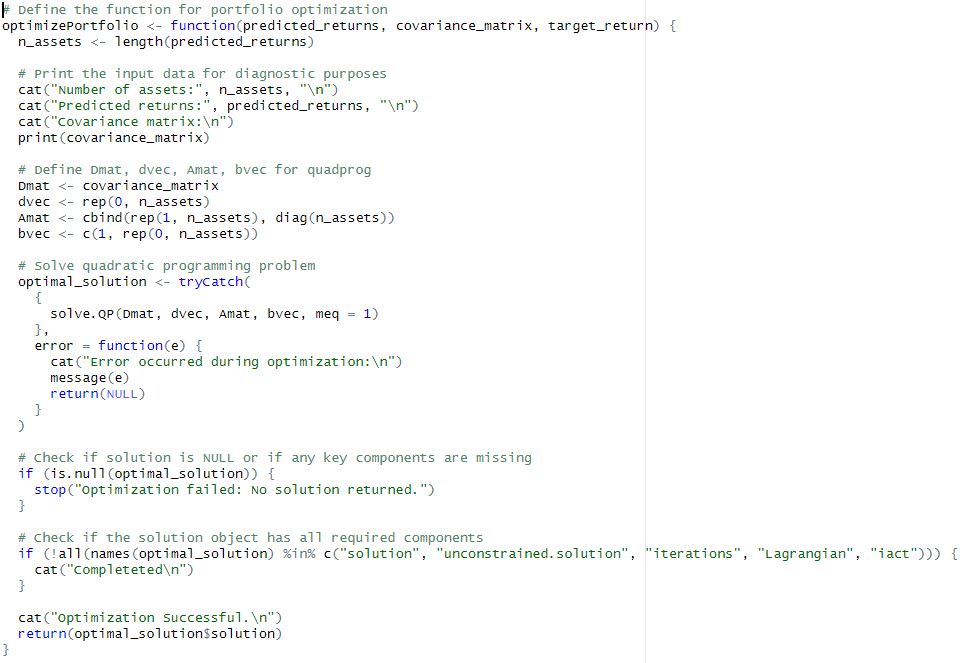
simulated\_historical\_returns <- matrix(rnorm(30 \* 250, mean = 0.0005, sd = 0.01), ncol = 30, nrow = 250): This line creates a matrix of simulated returns for 30 components over 250 days again with each return drawn from a normal distribution with same mean and standard deviation as predicted\_returns. The matrix is 250 rows by 30 columns with each column representing an asset and each row representing a day.

covariance\_matrix <- cov(simulated\_historical\_returns): The covariance matrix is computed from simulated historical returns. In a financial context this matrix is crucial for understanding relationships between asset returns. It is used in portfolio optimization to minimize overall risk of portfolio by accounting for how asset returns move in relation to one another.

num\_stocks <- nrow(predicted\_returns\_df): This line is likely intended to count number of stocks in portfolio by counting number of rows in predicted\_returns\_df.

**Portfolio Optimization via Quadratic Programming:**

Optimization is achieved through quadratic programming and specifically using solve.QP function. This method finds optimal allocation of assets that minimizes risk for a given return level. It incorporates constraints such as sum of portfolio weights making 1 and prohibiting negative weights that is no short selling. This step is essential for constructing a portfolio that aligns with investor's risk-return profile.





**Custom Optimization Function:**

The optimizePortfolio function streamlines optimization process handling inputs like predicted returns and covariance matrix to output optimal asset weights. This abstraction simplifies complex process of portfolio optimization and making script more organized and easier to understand.

nputs: It takes three arguments:

predicted\_returns: A vector of expected returns for each asset.

covariance\_matrix: The covariance matrix of the returns which measures extent to which the returns on two assets move .

target\_return: The desired return for portfolio.

Print Diagnostic Information: It prints out number of assets and predicted returns and covariance matrix for diagnostic purposes.

Quadratic Programming Setup: It defines components needed for quadprog function solve.QP:

Dmat: The matrix in quadratic term of optimization problem which in this case is the covariance matrix of asset returns.

dvec: The vector in linear term of optimization problem which is set to zero indicating there are no linear costs or rewards.

Amat: The matrix defining linear constraints , this setup ensures that sum of weights is equal to 1 and that there are no short positions (all weights are non-negative).

bvec: The right-hand side of linear constraints; a vector that in combination with Amat and enforces sum of weights to be 1.

Quadratic Programming Solution: It attempts to find optimal weights for the assets in portfolio that minimize risk for a given target return and using solve.QP function from quadprog package. The tryCatch block is used to catch any errors during optimization process.

Output: If optimization is successful and it prints out a success message and returns optimal weights for portfolio. These weights represent how much of each asset should be held in portfolio to achieve target return with minimum possible risk variance.

**Error Handling in Optimization:**

Including error handling within optimization function ensures that any issues encountered during optimization process are caught and reported. This precaution is necessary for debugging and ensures reliability of optimization outcomes.

Error Handling: If an error occurs during optimization or if solution is incomplete and function will print out an error message.

**Target Return Specification:**

Setting a target return as means of predicted returns leads optimization process toward a specific objective. This step is crucial for adapting portfolio to meet specific investment goals and such as achieving a certain level of return.





The target\_return appears to be set to approximately 2.89624396819338e-05 by which is mean of predicted\_returns vector. This number represents average expected return of tassets based on model's predictions. The e-05 at end indicates that this is very small number which means that number should be shifted five places to left (so 0.0000289624396819338). In financial returns this is typical number and as returns are often small percentages.

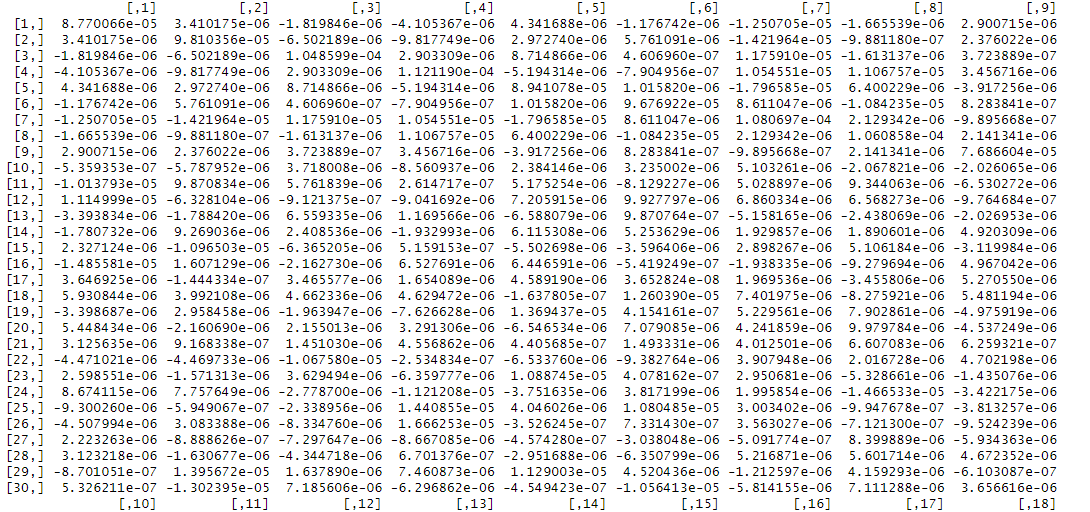
The optimal\_weights is displayed as a numeric vector with elements only one is visible in image such as -0.00806. Since these are portfolio weights derived from optimization function, they represent proportion of total portfolio that should be invested in each asset to achieve target return at minimum risk and according to optimization model used and negative weight suggests short selling that asset. The num [1:672] indicates this is a numeric vector with 672 elements which suggests optimization process considered a large number of assets and there could be discrepancy as previous code mentioned 30 assets.

**Weights:**

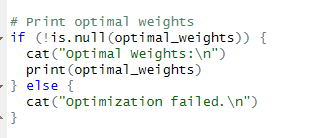
****

Each cell in the matrix represents weight or proportion of total portfolio that should be invested in that particular asset. The first element in first row [1,] 8.770066e-03 suggests that approximately 0.877% of portfolio should be allocated to first asset in first optimization scenario.

The output suggests that optimization was run multiple times potentially as part of a Monte Carlo simulation or some other form of scenario analysis to see how optimal weights change under different conditions or to account for model uncertainty. The presence of negative weights indicates that model may be suggesting short selling certain assets if strategy allows for it. These weights are solution to tquadratic programming problem set up in optimizePortfolio function using solve.QP method from quadprog package in R.

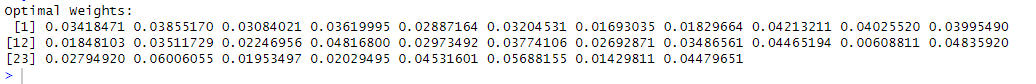
****

**Optimized Portfolio:**

****

The if statement checks if optimal\_weights object is not NULL meaning optimization function did return a result. This is a check to make sure optimization process was successful.

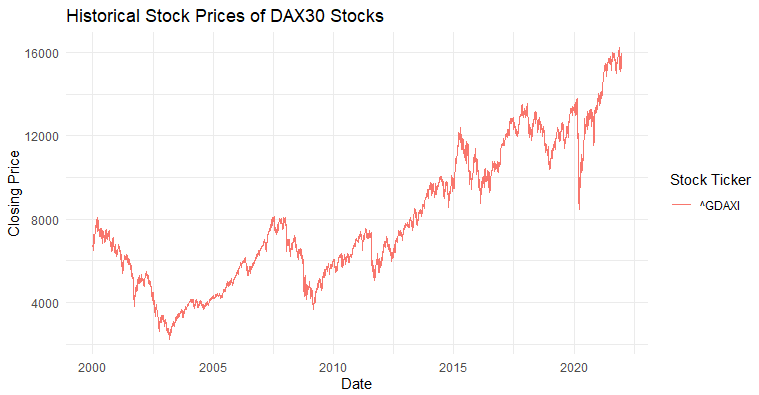
If optimal\_weights is not NULL cat function prints string "Optimal Weights:\n" to the console. This is just a message to indicate that following output will show weights for portfolio that are considered 'optimal' by model.



**Visualization**

**Plotting Historical Stock Prices**

Visualizing Historical Prices: A ggplot is created to plot historical closing prices of stocks in DAX30 index and using different colors for each stock ticker. This visualization helps in understanding price trends and volatility of different stocks over time.



Displaying historical closing prices of the DAX 30 index. The x-axis represents date and spanning from sometime before year 2000 to beyond 2020. The y-axis represents closing price of index.

The line plot, which trends upward and to right over time suggests an overall long-term increase in the index value, which is indicative of growth in stock prices of the constituent companies.

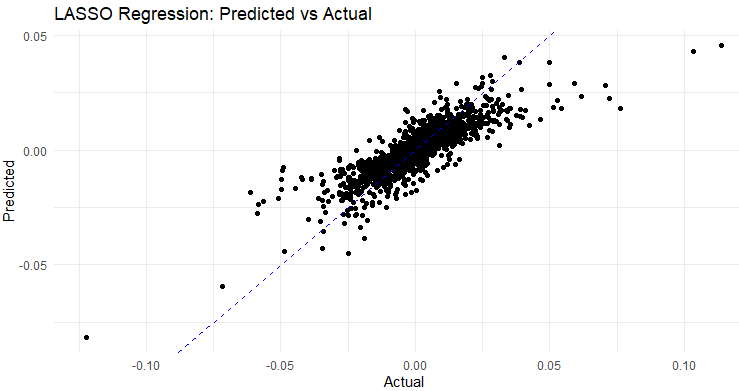
The plot shows various fluctuations with peaks and troughs indicating periods of growth and decline which are typical in stock market behavior due to various economic cycles and market conditions and events impacting market. The notable downturns and recoveries might correspond to economic recessions, market corrections and crises while uptrends suggest periods of economic growth and positive market performance.

**Model Performance Evaluation Plots**

**Actual vs. Predicted Values for Various Models:** The code then prepares data frames for different regression models. Lasso, Ridge, Random Forest, Elastic Net, SVM and XGBoost. Each data frame contains actual values and predictions made by these models.

**Performance Visualization:** For each model a performance plot is generated to visually compare actual values against predictions. These plots include a dashed line representing perfect predictions to easily assess accuracy of each model.

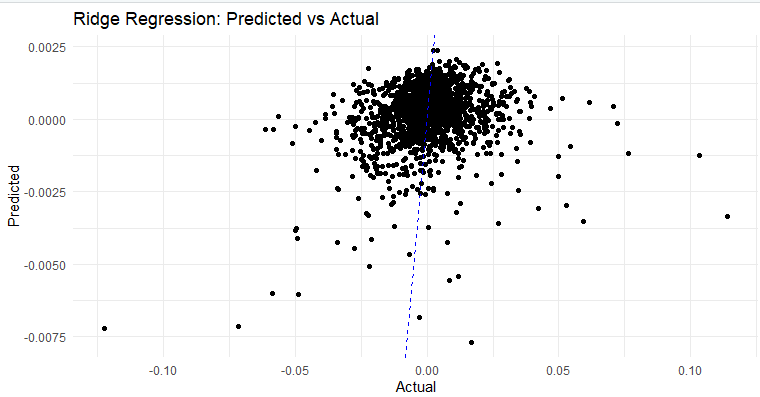
**Lasso and Ridge Regression Plots:** These plots specifically use lasso and ridge pred as predicted value which is regularization parameter chosen by cross-validation within model.



Most data points cluster around diagonal, suggesting that the LASSO regression model has a reasonable level of predictive accuracy for this particular dataset.

The concentration of points along center indicates that model performs well when actual values are around 0.

There are some outliers particularly in the lower left and upper right areas of plot andwhere model predictions are notably different from actual values.

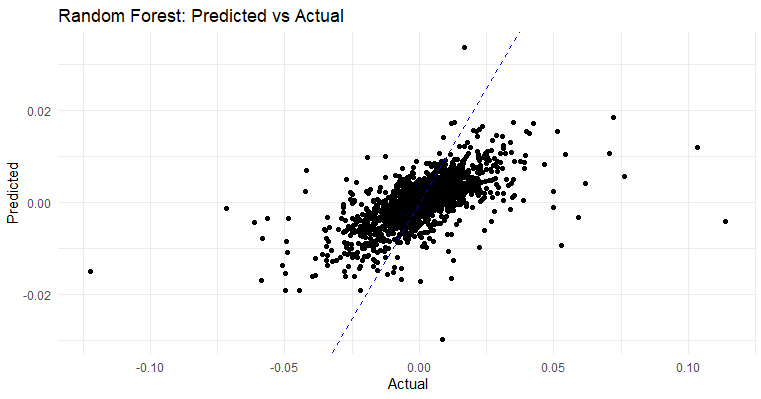


The points seem to be densely clustered around line when actual values are close to zero suggesting good predictive performance for values near zero.

Unlike LASSO plot points are less spread out along x-axis indicating that extreme values are less accurately predicted by this model or there are simply fewer extreme values in dataset.

The overall spread of points is more vertically concentrated indicating less variance in predictions compared to the actual values.

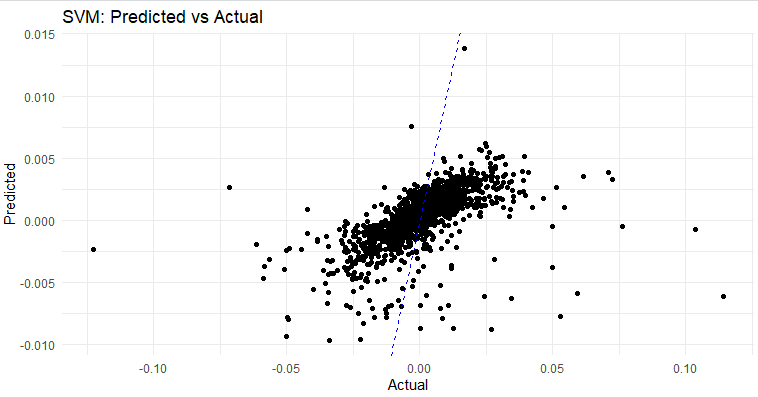
**Random Forest, SVM, and XGBoost Plots:** Correctly plot actual vs. predicted values and using points to represent individual predictions and dashed line for perfect predictions.



The Random Forest model which is an ensemble learning method appears to have a reasonable predictive accuracy, with many of the data points clustered around the diagonal line that represents perfect prediction.

There's a dense concentration of points around the center, indicating that model predicts more accurately when actual values are near zero.

As with other plots points that are further from line represent less accurate predictions. It seems that as magnitude of the actual values increases model's predictions become less accurate.

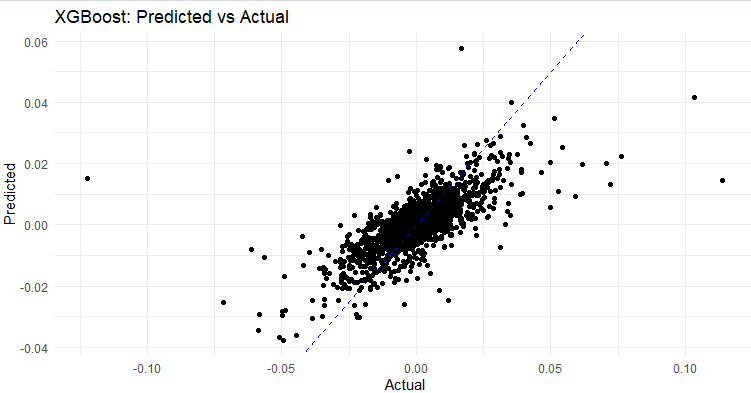


The data points represent individual predictions by the SVM model.

The points are clustered around center of plot particularly near line suggesting that SVM model has a decent fit for values close to zero.

The variance of predictions seems to increase as actual values move away from zero this can be seen as points spread out more in y-direction as we look to left and right away from center.

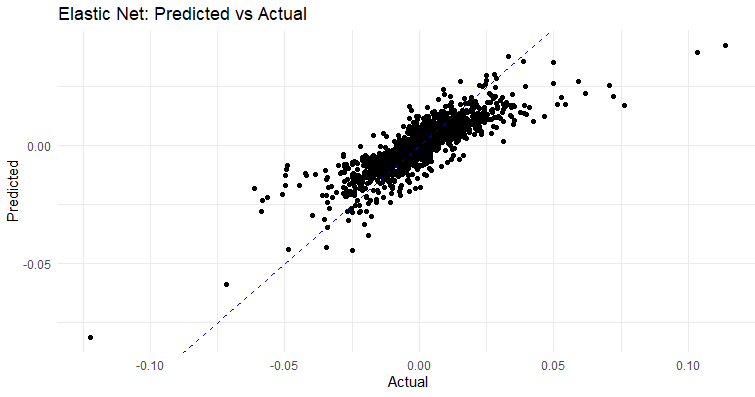
There are fewer points at the extremes and these tend to be further from line indicating that model's predictions are less accurate for more extreme actual values.



The concentration of points around diagonal line suggests that the XGBoost model predictions are closely aligned with the actual data especially around center.

The spread of points away from line at the higher and lower ends of the actual values indicates less accuracy in those regions.

**Elastic Net Regression Plot:** Similar to Lasso and Ridge it used elastic\_net\_pred for plotting to see the results of model.



The dashed diagonal line represents line of perfect prediction where predicted values are exactly equal to actual values.

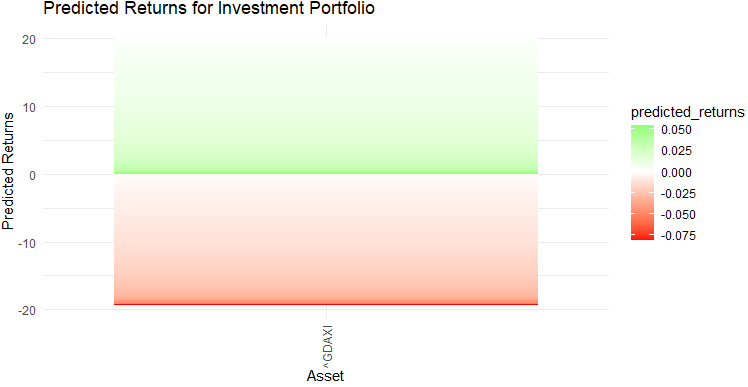
The data points are the individual prediction and we can see that many of them are close to diagonal line which suggests a good prediction accuracy for this model.

The concentration of points along diagonal indicates that model performs quite well when predicting majority of data points.However there are some outliers or predictions that deviate from line especially for actual values at extremes of the range which is typical in predictive modeling.

**Plotting Predicted Returns for DAX30 Assets**

Filtering Predicted Returns: It filters predicted returns to only include those related to assets in DAX30 index ensuring visualization is relevant to index.

Visualizing Predicted Returns: Creates a bar plot of predicted returns for filtered assets allowing for an easy comparison of expected performance across different stocks in portfolio.



The green portion of bar at top indicates portion of predicted return that is positive while red portion at bottom would represent negative part. However since entire bar is green and positioned just above zero this suggests that predicted return for DAX (^GDAXI) is slightly above zero within positive range.

The color scale on the right labeled "predicted\_returns" shows gradient range from -0.075 to 0.050, which matches color of the bar with predicted return value. The bar being completely green suggests predicted return is a positive value within scale provided.