META STOCK PRICE PREDICTION

UTILIZING MACHINE LEARNING MODELS

THAO VY NGUYEN (JULIE)

PROJECT OVERVIEW

Objective

Predict META stock returns using machine learning models based on a mix of stock data, technical indicators, and macroeconomic factors.

Key Models

Random Forest, XGBoost, and Support Vector Machine (SVM) were selected to handle the non-linear relationships in the stock data.

Key Focus

- Choosing relevant features
- Exploring non-linear relationships between features
- Exploring the implementation of non-linear regression algorithms
- Model implementation following time sequence

DATA COLLECTION

Stock data was collected from **‘yfinance’ (Yahoo Finance)**, and macroeconomic data from the **FRED API (Federal Reserve Economic Data)**.

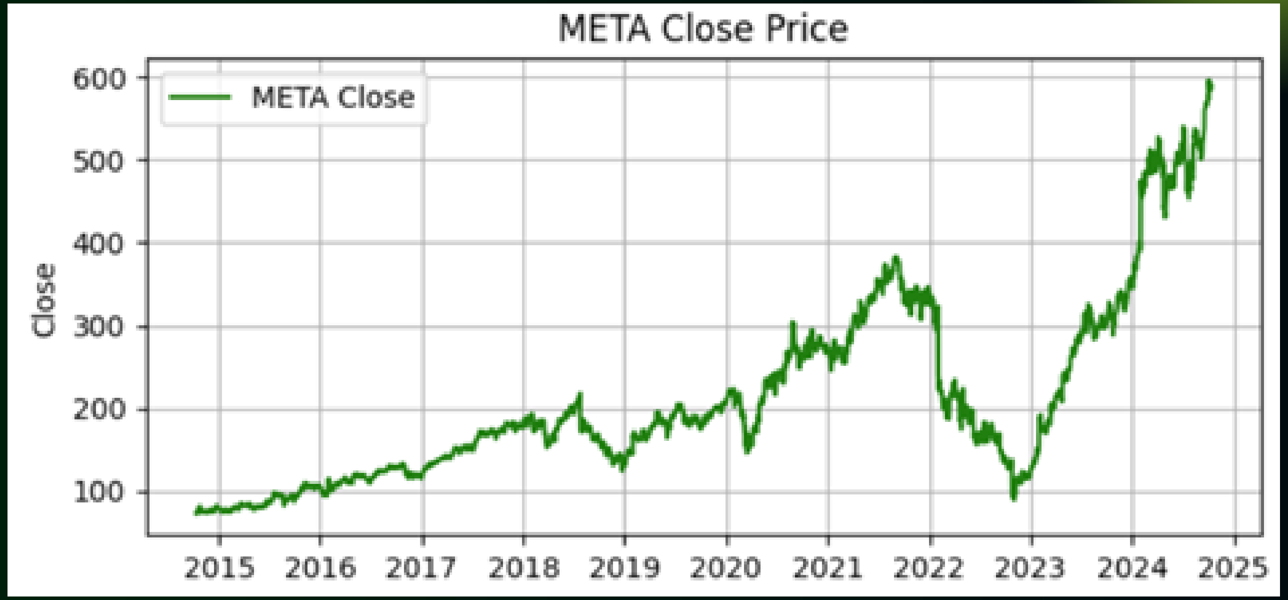
Target

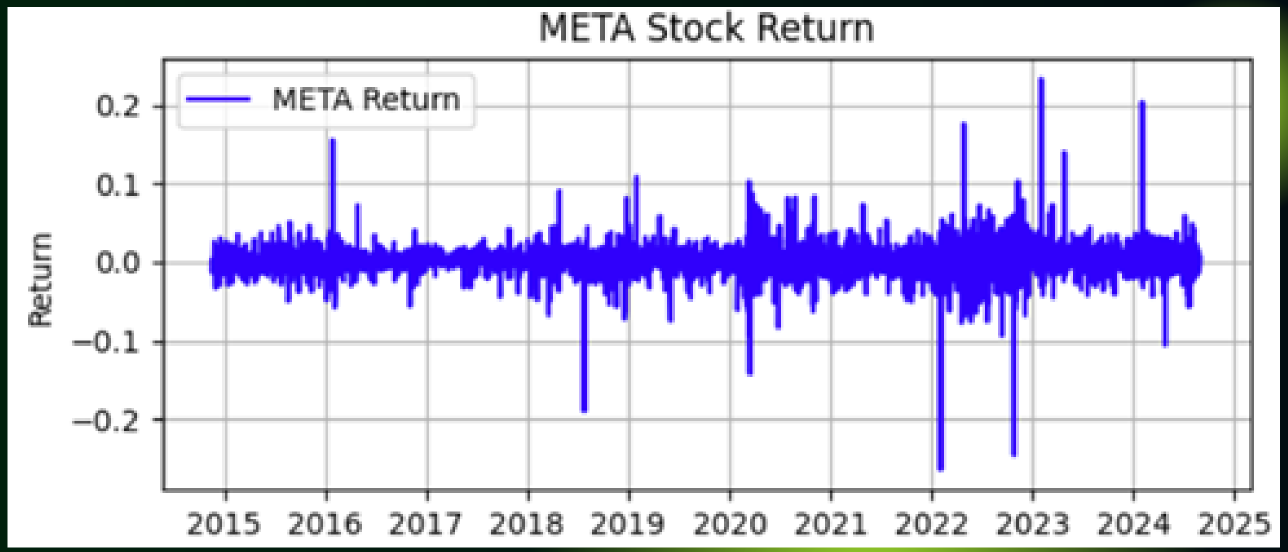
‘Return’ (pct. change in the stock’s closing price)

- Normalized performance
- More stationary

Key Features

- META stock data: ‘Return’ and its lagged feature
- Technical indicators: RSI, MACD, Bollinger Bands, and Stochastic Oscillator values
- Macroeconomic indicators: Federal Funds Rate (Interest Rate) and S&P 500 returns





DATA PREPROCESSING

01 Handling Missing Values

- Forward-filling techniques for time-series data
- Dropping rows

02 Train-Val-Test Splitting and Scaling

- Ratio: 70-15-15
- Follow time sequence
- StandardScaler

MODEL SELECTION

All models were designed to capture complex, non-linear relationships between stock returns and various features.

RANDOM FOREST

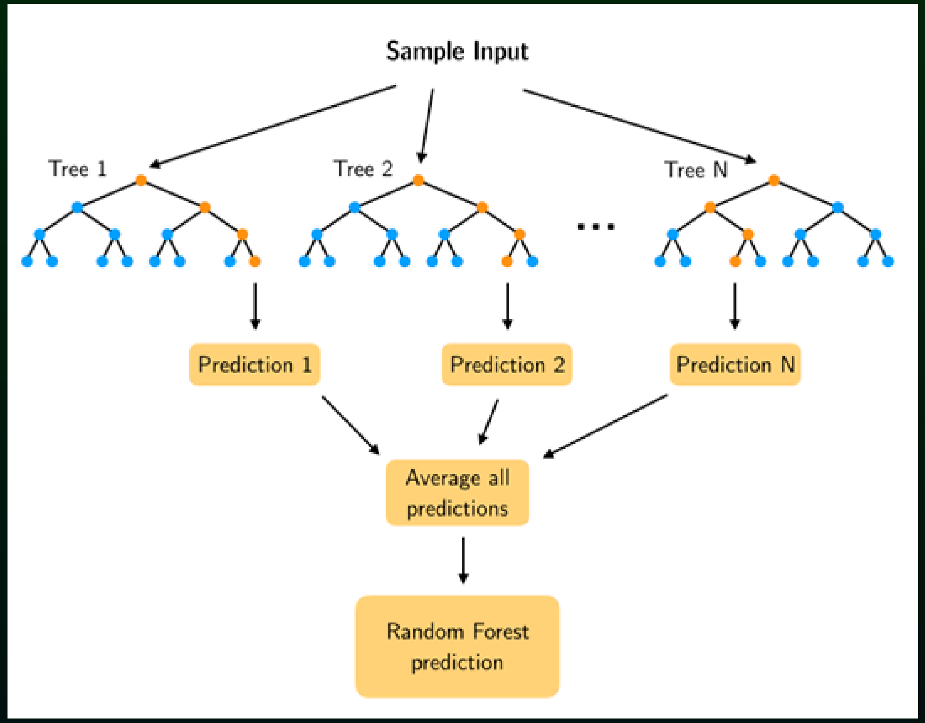
Combines multiple decision trees to reduce variance

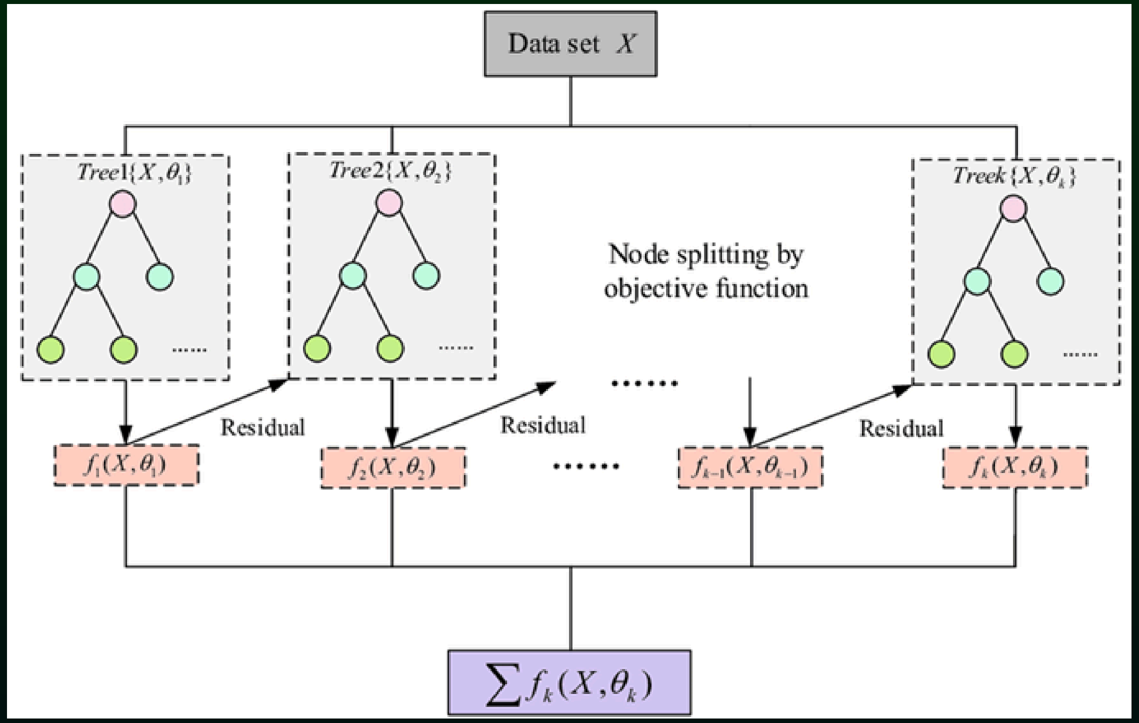
XGBOOST

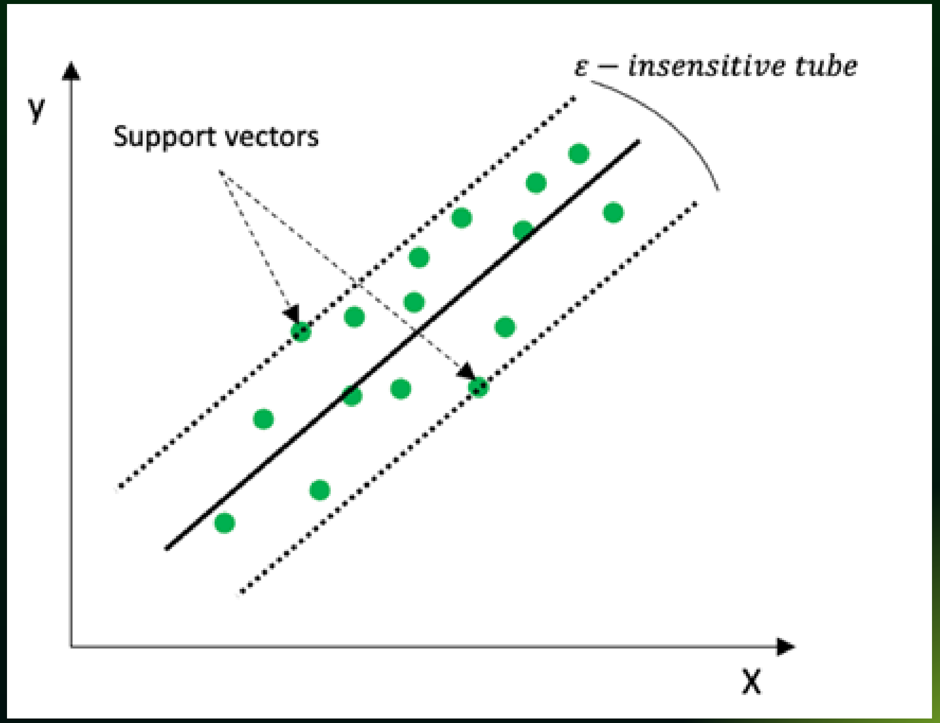
Sequential boosting method optimizing prediction residuals.

SUPPORT VECTOR MACHINE

Uses kernel functions to handle non-linear data.







LOSS FUNCTION

RANDOM FOREST   XGBOOST

Minimize Mean Squared Error to focus on reducing large errors.

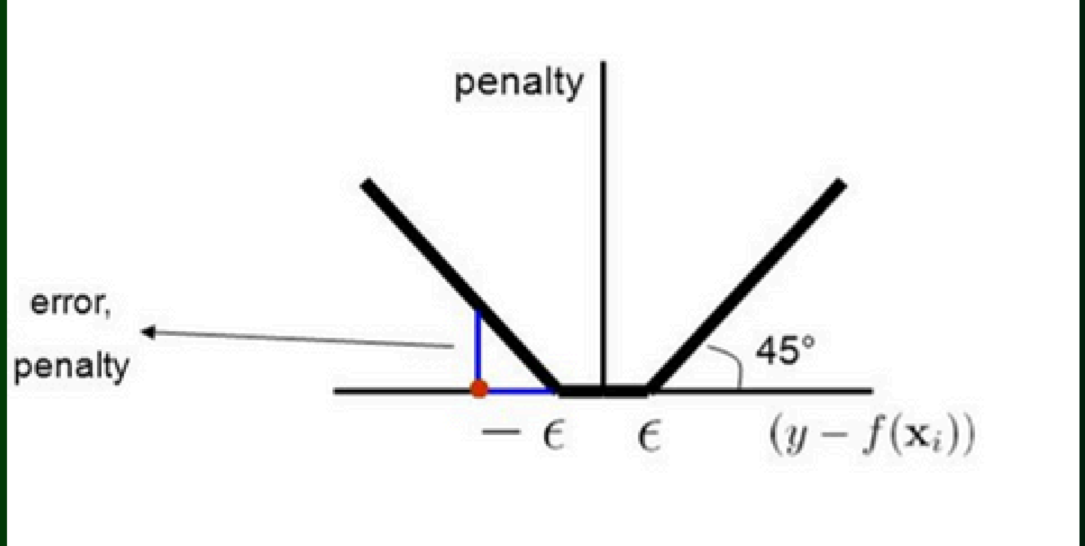
- Penalize large errors more than smaller ones
- Computationally efficient

$$\text{MSE} = \overset{\text{Mean}}{\boxed{\frac{1}{n}\sum_{i=1}^{n}}}\overset{\text{Error}}{\boxed{(Y_i - \hat{Y}_i)}}^{\overset{\text{Squared}}{\boxed{2}}}$$

- y_i represents the actual value.
- $\hat{y}_i$ represents the predicted value.
- n is the number of observations.

SUPPORT VECTOR MACHINE

Epsilon-insensitive loss: Ignoring small deviations and emphasizing significant prediction errors



$$|y_i - f(\mathbf{x}_i)|_\epsilon \equiv \max\{0, |y_i - f(\mathbf{x}_i)| - \epsilon\}$$

MODEL IMPLEMENTATION

01 BASELINE MODEL

Use mean of the training set returns as the prediction for all future returns

02 INITIAL TRAINING

Training the models with default hyperparameters

03 HYPERPARAMETER TUNING

- Selecting the best combination of hyperparameters
- Grid Search with TimeSeriesSplit cross-validation (n_splits=3)

MODEL IMPLEMENTATION - RANDOM FOREST

		Default	**Tuned** (Best from GridSearch)
Configuration	n_estimators	100	500
	max_depth	None	10
	min_samples_split	2	5
	min_samples_leaf	1	2
Performance	RMSE	0.011	**0.012**
	MAE	0.001	**0.001**
	R2	0.915	**0.902**

- The tuned version performed slightly worse than the default
- Became too constrained, with reduced tree depth and higher split/leaf sample requirements, leading to underfitting and worse generalization.

MODEL IMPLEMENTATION - XGBOOST

		Default	**Tuned** (Best from GridSearch)
Configuration	n_estimators	100	500
	max_depth	6	5
	learning_rate	0.01	0.03
	subsample	1	0.8
	regularization		lambda=2 alpha=0.5
Performance	RMSE	0.013	**0.016**
	MAE	0.002	**0.003**
	R2	0.875	**0.820**

- The tuned version performed slightly worse than the default
- The lower learning rate combined with a higher number of boosting rounds may have led to overfitting, even with regularization.
- It may also have underfit due to the more gradual learning.

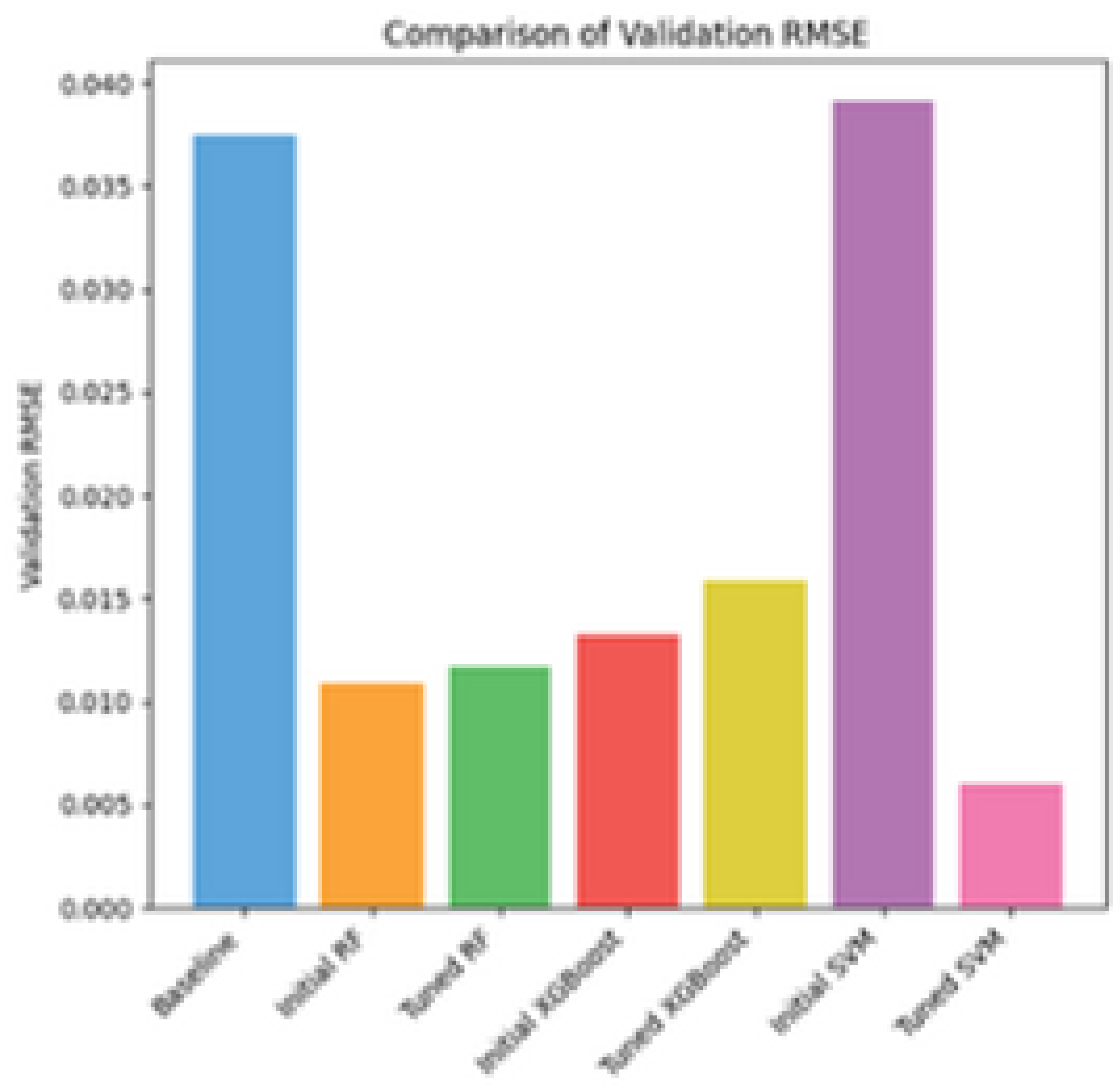
MODEL IMPLEMENTATION - SVM

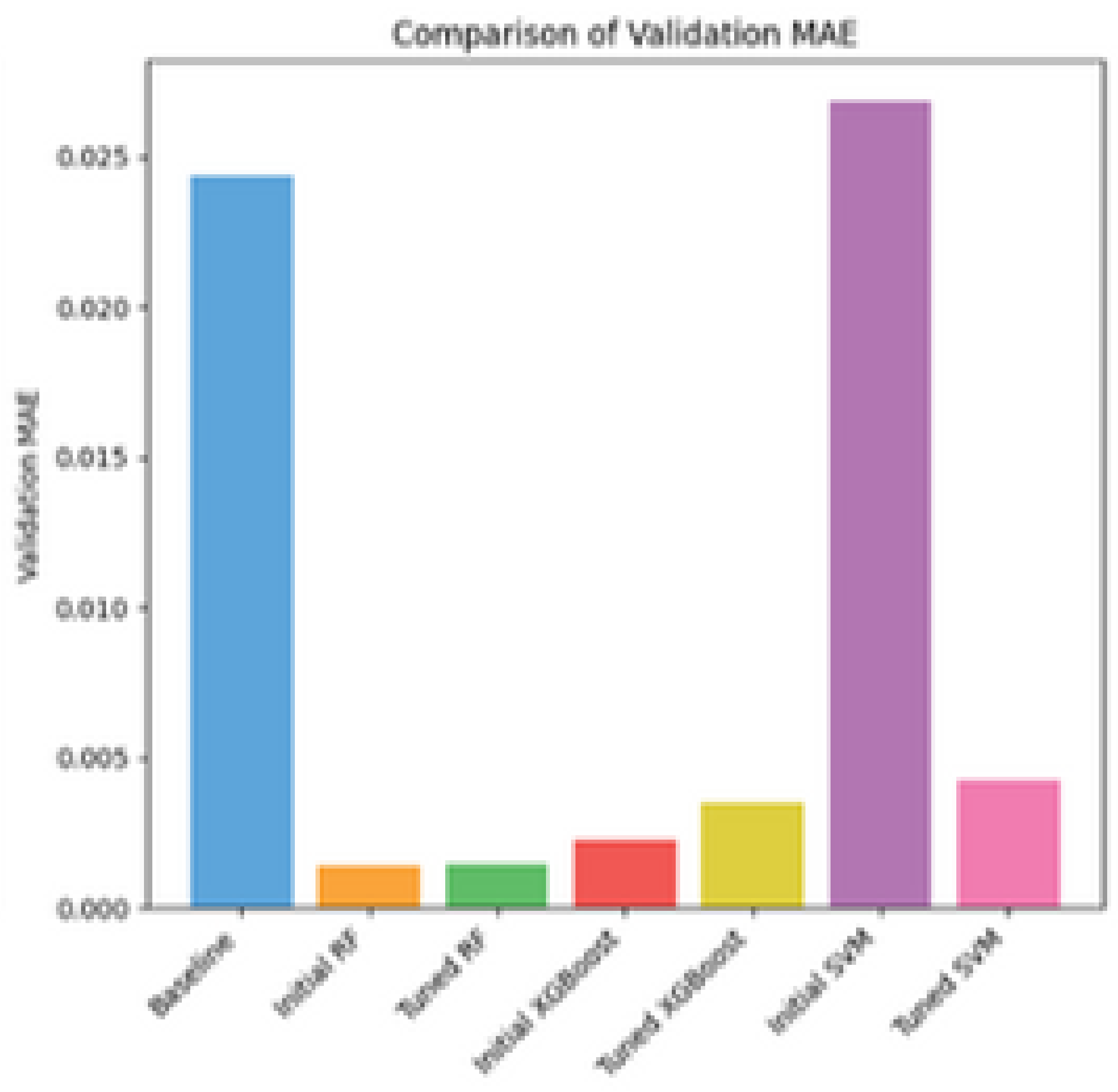
		Default	**Tuned** (Best from GridSearch)
Configuration	C	1	20
	epsilon	0.1	0.01
	kernel	'rbf'	'poly'
	gamma		‘auto’
Performance	RMSE	0.039	**0.006**
	MAE	0.027	**0.004**
	R2	-0.094	**0.974**

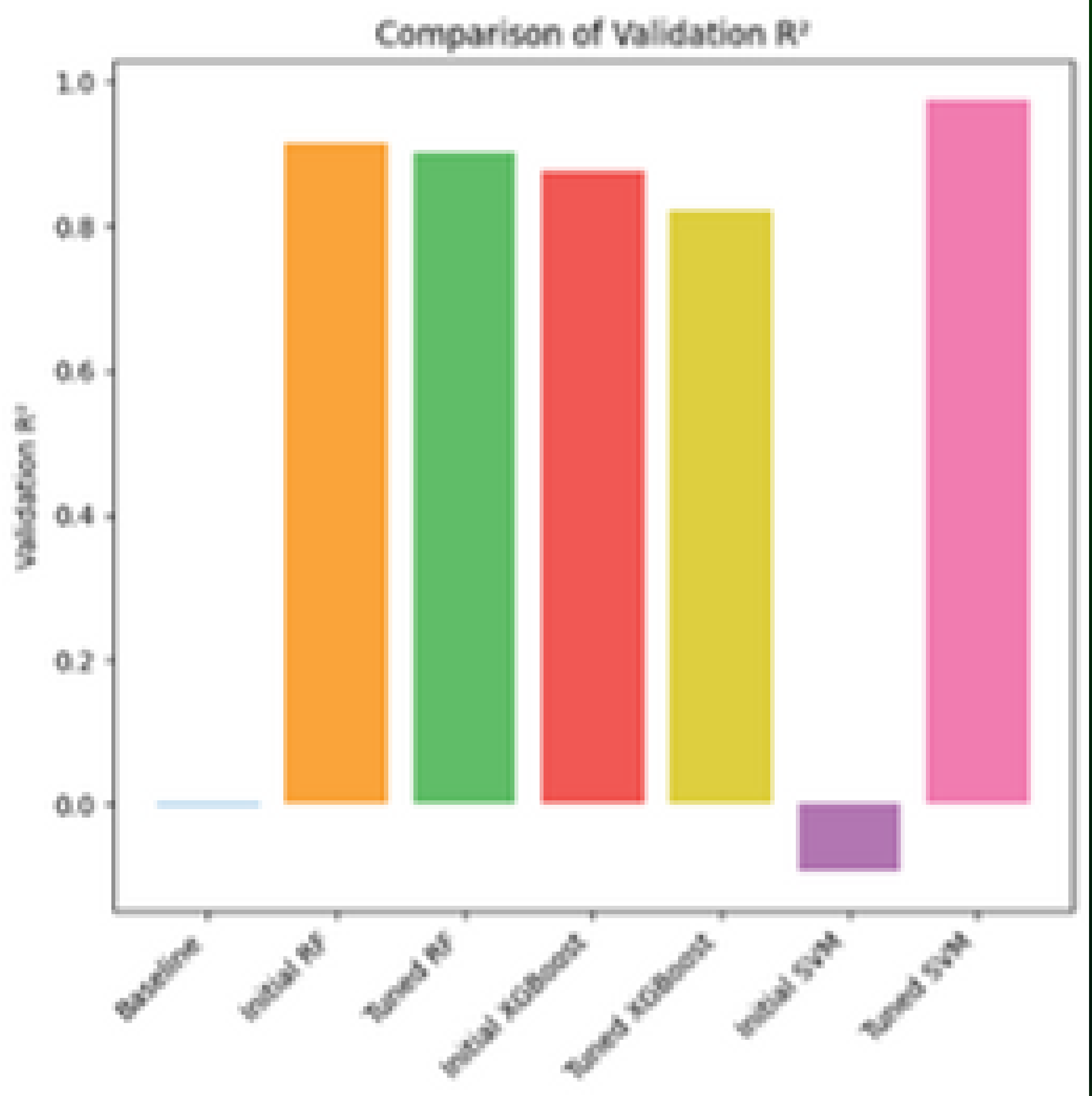
- The default SVM model struggled with performance, as the RBF kernel and wider epsilon margin did not capture enough complexity in the stock return data.
- The tuned SVM model performed much better. The polynomial kernel, higher regularization (C), and smaller epsilon helped the model generalize well, capturing complex, non-linear relationships in the data and focusing on significant errors.

MODEL EVALUATION

Tuned SVM model outperformed all others across most metrics on validation set:





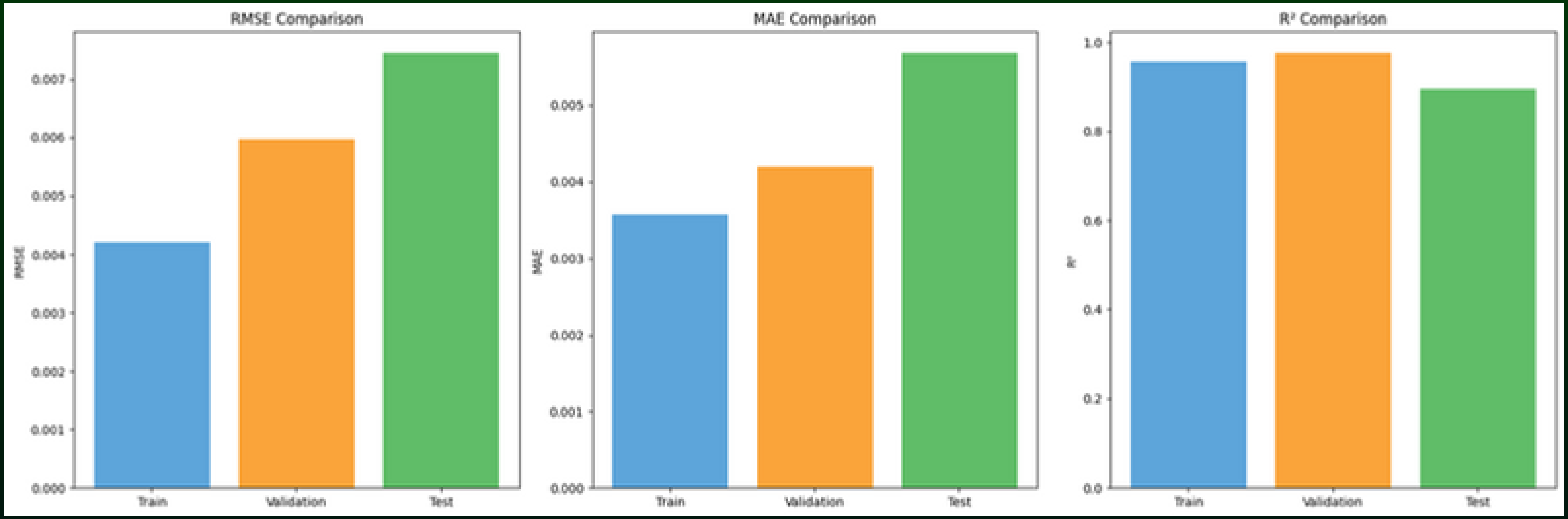


Tuned SVM had the lowest RMSE, indicating it had the smallest average prediction error.

While the SVM's MAE was slightly higher than that of Random Forest and XGBoost, it still performed very well.

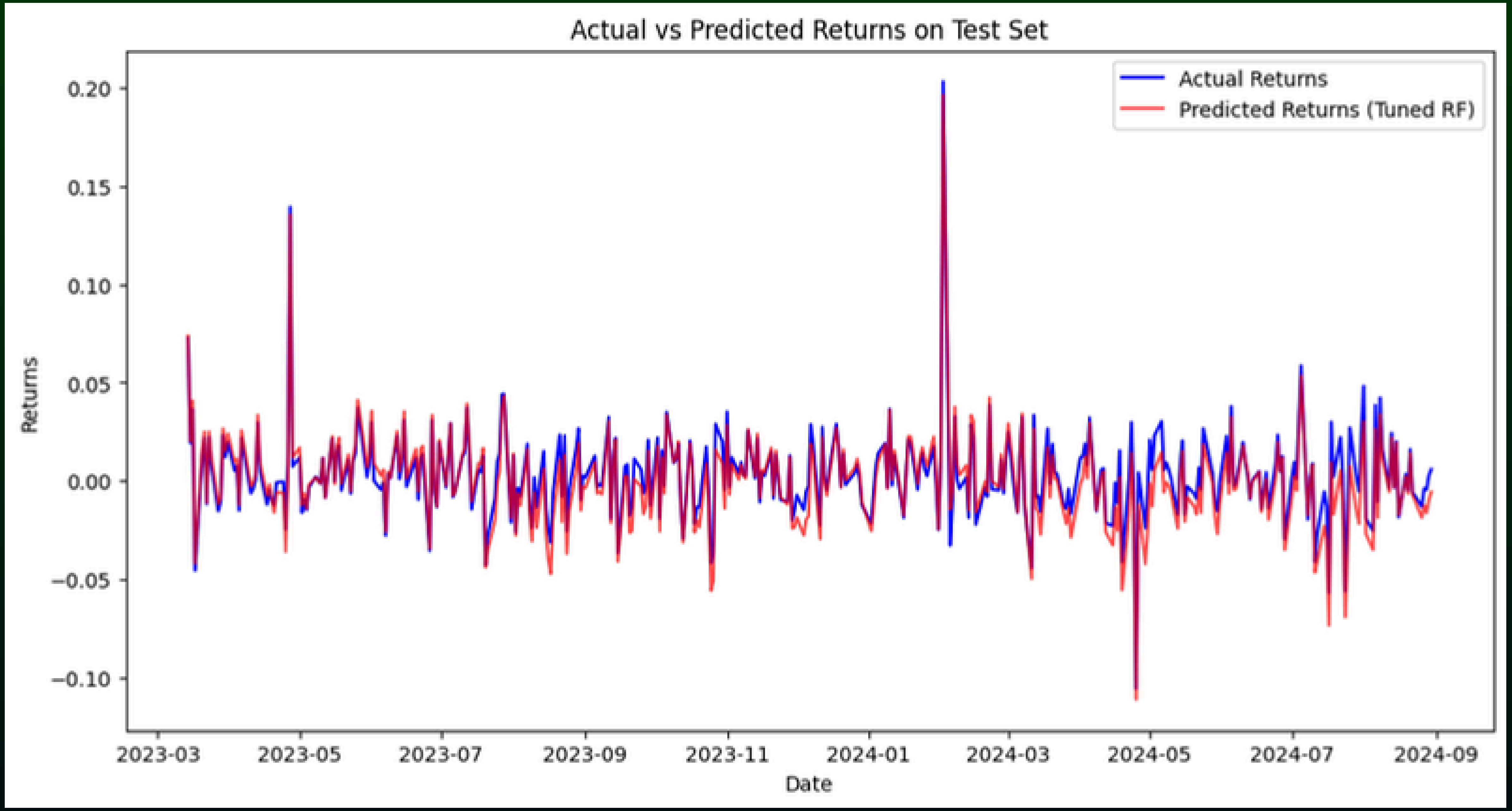
With an R^2 of 0.974, SVM explained the most variance, indicating it captured the complex relationships in the data most effectively.

BEST MODEL PERFORMANCE (SVM) ON TEST SET



While there is a slight performance degradation when moving from training to unseen data (validation and test sets), the model still provides accurate predictions, as indicated by the relatively low RMSE and MAE, and high R^2 values.

BEST MODEL PERFORMANCE (SVM) ON TEST SET



The model is able to track the actual returns reasonably well, even some sharp spikes and dips.