# 📘 Documentation: Stock Price Prediction using ML (Random Forest & XGBoost)

## 🔍 Overview

This document provides deep explanations behind every major decision in the machine learning workflow used for predicting Tesla (TSLA) stock returns using Random Forest and XGBoost. It covers model design, feature engineering, evaluation, and why specific methods were chosen.

## 📦 Data & Preprocessing

### 1. Why use shuffle=False in train\_test\_split?

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, shuffle=False)

* ✅ Required for **time-series problems**.
* **Stock prices depend on temporal order**. Today’s return depends on yesterday’s.
* Shuffling would leak future data into the past — creating impossible and misleading results.
* Example of incorrect split: training on May data, testing on Feb → breaks causality.

### 2. Why only scale X (features), not y (target)?

scaler = StandardScaler()  
X\_train\_scaled = scaler.fit\_transform(X\_train)

* X (Lag features) can vary more than y, so we scale them.
* y (Return) is already:
  + Small: mostly in the range -0.05 to +0.05
  + Centered: around zero (mean ≈ 0)
  + Intuitive: as % change
* ❌ Scaling y adds complexity → need to inverse transform after prediction.

### 3. Why is y = Return already small?

Return = (Today\_Close - Yesterday\_Close) / Yesterday\_Close

* Returns are by definition % changes.
* Daily stock moves are rarely over 5%.
* That’s why return values like 0.01, -0.003, etc., are already normalized.

### 4. What is Feature Scaling (for X)?

* Even though all lag features are returns, they might:
  + Vary in scale slightly
  + Have different volatility/distributions
* **StandardScaler** gives each feature:
  + Mean = 0
  + Std dev = 1
* ✅ Helps models treat all inputs equally.

## 🌲 Why Random Forest?

### Why 300 Trees (n\_estimators=300)?

* Random Forest = Ensemble of decision trees.
* Trade-off:

| Trees | Effect |
| --- | --- |
| <100 | Too unstable |
| 300 | ✅ Balanced performance |
| >500 | Marginal improvement |

* 300 is a **sensible default** for good bias-variance tradeoff.

### Why RandomForest over Linear/Ridge/Lasso/SVR?

| Model | Why We Didn’t Use It |
| --- | --- |
| **Linear Regression** | Only captures linear relationships; stock returns are highly non-linear |
| **Ridge / Lasso** | Add regularization (L2 / L1), but still linear → insufficient for return prediction |
| **SVR (Support Vector Regression)** | Good for small datasets but scales poorly and is sensitive to parameter tuning |
| **Decision Trees (single)** | Prone to overfitting and unstable on noisy data |
| ✅ **Random Forest** | ✔ Captures non-linearities, ✔ Robust to noise, ✔ Fast to train, ✔ Interpretable baseline |

## ⚡ Why XGBoost?

### What is XGBoost?

* **Extreme Gradient Boosting** → Trees built sequentially.
* Each new tree fixes errors of the previous ones.
* **Usually outperforms Random Forest** on tabular data.

### Hyperparameters Used:

| Parameter | Meaning |
| --- | --- |
| n\_estimators=500 | Number of trees |
| learning\_rate=0.03 | Slower learning = better generalization |
| max\_depth=5 | Shallower trees prevent overfitting |
| subsample=0.8 | Train each tree on 80% of data rows |
| colsample\_bytree=0.8 | Train each tree on 80% of features (columns) |
| objective='reg:squarederror' | Standard regression loss function |

✅ These settings introduce randomness → helps generalization.

## 📏 Evaluation Metrics

### Why these metrics?

| Metric | Formula | Meaning |
| --- | --- | --- |
| **MAE** | mean(abs(y\_true - y\_pred)) | Measures average error magnitude |
| **RMSE** | sqrt(mean((y\_true - y\_pred)^2)) | Penalizes large errors more |
| **R²** | 1 - (model error / baseline error) | Proportion of variance explained |

### What do results mean?

| Model | MAE | RMSE | R² | Interpretation |
| --- | --- | --- | --- | --- |
| Random Forest | 0.0316 | 0.00191 | -0.0201 | ✅ Lowest MAE. R² is weak → data is noisy |
| XGBoost | 0.0329 | 0.00208 | -0.1084 | ❌ Slightly worse performance |

### Negative R²?

* Means model is **worse than predicting the mean**.
* Common in stock returns → extremely hard to predict.

## 🔊 Why Stock Return Prediction is Hard

| Reason | Explanation |
| --- | --- |
| Noise | Stock prices react to irrational or unpredictable events |
| Non-stationary | Patterns change with time (volatility, regime shifts) |
| Spikes | Sudden movements due to earnings, news, rumors |

Even the best models often fail to outperform simple baselines.

## 🔮 Real-world Application

### Why Not Just Use These Models to Trade?

* Prediction ≠ strategy.
* Use model as a **signal** within a **trading rule**:

if predicted\_return > 0.005:  
 buy  
else:  
 hold

* Combine with:
  + Stop loss
  + Risk management
  + Technical indicators

## 🧼 What’s Missing? (and What’s Next?)

### Extra Features That Could Help:

* Volume
* Volatility (std dev of returns)
* RSI, MACD, Bollinger Bands
* Sentiment from news/Twitter

### De-noising Techniques (Advanced):

* Moving Averages
* Wavelet Transforms
* PCA

⚠️ Use carefully: might remove signal along with noise.

## ✅ Summary

* Used Random Forest & XGBoost to predict TSLA returns
* Minimal features: Lag\_1, Lag\_2, Lag\_3
* Data split carefully with time-order
* Scaled only inputs (X), not output (y)
* Evaluated using MAE, RMSE, R²
* Understood limitations: noisy target, limited R²
* Built a solid baseline for stock prediction workflows