

Samsung Innovation Campus – Artificial Intelligence

Final Project Report

Stock Price Movement Prediction Using Random Forest

A Machine Learning Approach to Predicting Next-Day Stock Direction

Submitted by (Group 33):

Tharish J	Introduction, Problem Definition, Proposed Solution
Naman Chawla	Technical Methodology, Data Processing, Model Architecture
Dina Abdelnaser Mohammed	Implementation, Model Training, Evaluation, Conclusion
Raia Mohammed Gyeye	Results & Analysis, Visualisations
Nicelin Printo	Challenges & Solutions, Final Revisions
Mariam Lulu Mohammed	Report Compilation, LaTeX Formatting, Structuring, Editing

Samsung Innovation Campus
2025

Contents

1	Introduction	3
1.1	Project Overview	3
1.2	Background of the Real-World Problem	3
1.3	Importance and Impact of the Problem	4
1.4	Objectives of the Project	4
2	Problem Definition	5
2.1	Clear Explanation of the Real-World Problem	5
2.2	Problem Statement	5
2.3	Dataset Description	6
2.4	Key Questions and Expected Outcomes	6
3	Proposed Solution	8
3.1	Chosen Solution Approach	8
3.2	Why Machine Learning?	9
3.3	Why Random Forest?	9
3.4	Alternatives Considered	10
4	Technical Methodology	11
4.1	Data Collection	11
4.2	Data Cleaning & Preprocessing	11
4.3	Exploratory Data Analysis (EDA)	12
4.4	Feature Selection	12
4.5	Train/Test Split	13
4.6	Model Architecture: Random Forest Classifier	14
4.7	Hyperparameters Used	14
4.8	Model Training Workflow	15
4.9	Evaluation Metrics	16
5	Implementation Details	20
5.1	Tools & Technologies Used	20

5.2	Python Libraries	20
5.3	Source Code Structure	21
5.4	Google Colab Code Overview	22
6	Results & Analysis	27
6.1	Model Performance Overview	27
6.2	Feature Importance Overview	28
6.3	Confusion Matrix Analysis	28
6.4	ROC-AUC and Threshold Behaviour	28
6.5	Company-Level Performance Analysis	29
6.6	Sample Predictions and Confidence Levels	29
6.7	Model Limitations and Future Improvements	30
7	Challenges & Solutions	33
7.1	Challenges Faced During Data Cleaning	33
7.2	Model Overfitting/Underfitting Issues	33
7.3	Parameter Tuning Challenges	34
7.4	Technical Difficulties	34
7.5	How These Challenges Were Addressed	34
8	Conclusion	35
8.1	Summary of Findings	35
8.2	Achievement of Project Objectives	35
8.3	Value & Impact of the Solution	36
9	Future Improvements	37
9.1	Optimizing the Random Forest Model	37
9.2	Adding More Data	37
9.3	Testing Other Algorithms	37
9.4	Building a Full Application (Web/App)	38
9.5	Deployment to Cloud / Real Industry Use	38
10	References	39
10.1	Research Papers	39
10.2	Websites, Journals, Tutorials	40
10.3	Dataset Sources	40

Chapter 1

Introduction

1.1 Project Overview

This project focuses on predicting next-day stock price movement — whether the closing price will go UP (1) or DOWN (0) — using a machine learning approach applied to a large global stock market dataset. The model utilises a rich set of financial indicators such as returns, volatility, moving averages, RSI, MACD, Bollinger Bands, and price momentum.

A Random Forest classifier is used because it handles noisy financial data, can model complex non-linear patterns, and provides clear interpretability through feature importance. The system is built to analyse each company separately, generate time-series technical indicators, and then combine all processed data for final classification.

1.2 Background of the Real-World Problem

Stock prices are highly volatile and influenced by unpredictable factors such as economic news, global events, industry trends, and investor sentiment. This makes forecasting extremely difficult. Traditional methods like simple moving averages or statistical models often fail when patterns become non-linear or when markets behave irrationally.

Machine learning, especially ensemble models like Random Forest, has become a powerful tool for financial forecasting because it can:

- learn from complex relationships,
- use many technical indicators together,
- detect subtle market behaviours, and
- work well even when data contains noise.

Predicting stock direction helps traders and investors reduce risk, improve decision-making, and optimise trading strategies.

1.3 Importance and Impact of the Problem

A reliable UP/DOWN prediction model:

- helps traders time their entry and exit,
- reduces emotional decision-making,
- supports automated trading systems,
- enhances financial planning and risk management, and
- allows analysts to compare prediction performance across companies.

Even small improvements above the baseline (50%) can significantly affect long-term profitability in algorithmic trading.

1.4 Objectives of the Project

The main objectives of this project are:

- build an ML pipeline to predict next-day stock direction using historical data,
- engineer powerful technical indicators (RSI, MACD, Momentum, EMA, Bollinger Bands) for better predictive strength,
- train a regularised Random Forest model with improved generalisation,
- evaluate model performance using accuracy, ROC–AUC, classification report, and confusion matrix,
- analyse prediction accuracy at a company level, and
- identify the most influential financial indicators.

Chapter 2

Problem Definition

2.1 Clear Explanation of the Real-World Problem

Stock markets behave dynamically and are influenced by rapidly changing economic factors, making next-day price direction a difficult classification task. The aim of this project is to design a model that studies past price movements and technical indicators to determine whether a stock is more likely to rise or fall the following day.

The challenge arises because:

- market data is noisy and non-linear,
- each company exhibits different trading patterns,
- short-term movements are highly unpredictable, and
- technical indicators interact in complex, overlapping ways.

This project addresses these challenges by combining structured feature engineering with a Random Forest classifier to model short-term stock direction.

2.2 Problem Statement

The goal is to predict whether a stock's next-day closing price will increase or decrease using historical price data and engineered technical indicators. This forms a binary classification problem:

- $1 \rightarrow \text{UP}$,
- $0 \rightarrow \text{DOWN}$.

2.3 Dataset Description

The dataset used is the *World Stock Prices – Daily Updating* dataset from Kaggle. It contains daily historical data for multiple global companies, including:

- Date, Open, High, Low, Close,
- Volume,
- Dividends, Stock Splits, Capital Gains,
- Brand_Name (company), Country, Sector.

After loading and filtering, the dataset used in this project has the following characteristics:

- 21,811 total rows,
- 62 companies,
- date range: 2024–2025,
- approximately 350 records per company (with slight variation),
- suitable for technical indicator generation and short-term prediction.

Each company’s data is processed individually to generate indicators such as EMA, MACD, RSI, volatility, Momentum, and Bollinger Bands, ensuring correct time-series behaviour.

2.4 Key Questions and Expected Outcomes

Key Questions

The project is designed to answer the following questions:

1. Can technical indicators help predict next-day stock movement above baseline accuracy (50%)?
2. Which indicators contribute most to prediction performance?
3. Does the Random Forest model generalise effectively across different companies?
4. How does accuracy vary between companies and sectors?
5. What ROC–AUC score can be achieved with this feature set?

Expected Outcomes

The expected outcomes are:

- a functional classifier capable of predicting next-day stock direction,
- a robust feature-engineering pipeline with 20+ technical indicators,
- feature-importance analysis highlighting the most influential indicators,
- evaluation using accuracy, classification report, confusion matrix, and ROC-AUC,
- company-level performance comparison to identify which stocks are more predictable, and
- model performance that exceeds random guessing.

Chapter 3

Proposed Solution

3.1 Chosen Solution Approach

The proposed solution uses a feature-engineered Random Forest classifier trained on multiple technical indicators to predict next-day stock movement for each company. The workflow consists of the following steps:

1. Load the Kaggle dataset and sort it by company and date.
2. Generate technical indicators such as RSI, MACD, EMA, Bollinger Bands, Momentum, and ROC.
3. Compute additional features including volatility, returns, price range, moving averages, and volume trends.
4. Create the binary target variable based on next-day closing price movement.
5. Handle missing, undefined, and infinite values created during indicator calculation.
6. Perform a stratified train/test split to preserve class balance.
7. Apply `StandardScaler` to normalise numerical features.
8. Train a regularised Random Forest classifier with tuned parameters.
9. Evaluate performance using accuracy, classification report, confusion matrix, and ROC-AUC.
10. Conduct company-level prediction analysis to examine per-stock behaviour.

This pipeline ensures a structured and realistic approach to short-term stock movement prediction.

3.2 Why Machine Learning?

Machine learning is well-suited for this task because:

- stock price movements exhibit highly non-linear and noisy patterns,
- technical indicators interact in complex ways that are difficult to model manually,
- ML algorithms can identify hidden relationships within multi-feature datasets,
- ensemble models are more stable and less sensitive to noise, and
- ML provides automated, repeatable, and objective prediction processes.

In financial forecasting, ML improves decision-making by combining multiple indicators and learning behaviour across diverse companies.

3.3 Why Random Forest?

The model uses a Random Forest classifier configured with:

- `n_estimators = 250`,
- `max_depth = 10`,
- `min_samples_split = 5`,
- `min_samples_leaf = 2`,
- `max_features = 'sqrt'`,
- `max_samples = 0.9`,
- `class_weight = 'balanced_subsample'`.

This choice is justified because Random Forest:

- handles noisy, volatile financial data effectively,
- reduces overfitting through bootstrap aggregation (bagging),
- works well with many correlated technical indicators,
- provides interpretability via feature importance,
- performs reliably on tabular datasets with mixed feature types, and
- automatically manages class imbalance using class weights.

These properties make Random Forest a strong and stable model for multi-company stock-direction classification.

3.4 Alternatives Considered

Other models were considered but not selected:

- XGBoost / LightGBM — can achieve high accuracy but require extensive tuning and greater computational cost,
- LSTM / GRU — designed for sequential data but slow to train and require longer historical datasets,
- Support Vector Machines — effective but not scalable with larger feature sets and datasets,
- Logistic Regression — a good baseline but too simple for the non-linear nature of financial patterns.

Random Forest offered the best balance of accuracy, interpretability, training speed, and robustness for this project.

Chapter 4

Technical Methodology

4.1 Data Collection

The dataset used for this project is the *World-Stock-Prices-Dataset*, containing global stock market data for multiple internationally listed companies. The dataset includes traditional OHLCV fields (Open, High, Low, Close, Volume) along with dividend-related information (Dividends, Stock Splits, and Capital Gains).

After loading and inspection, the dataset exhibits:

- a large number of historical stock-price entries,
- multiple companies with varying sample sizes, and
- a date range spanning multiple years (as printed in the code using `data['Date'].min()` and `data['Date'].max()`).

All records were sorted chronologically per company to preserve the integrity of time-series behaviour. The number of companies and the record distribution per company were printed and reviewed to understand dataset density across different firms.

4.2 Data Cleaning & Preprocessing

Preprocessing steps were applied at the company level to ensure clean, sequential financial data.

Timestamp standardisation. All dates were converted to proper Python `datetime` objects.

Chronological sorting. Each company's data was sorted by `Date` to maintain the correct sequence for rolling indicators and future-target computation.

Company filtering. Companies with fewer than 20 observations were excluded to ensure adequate historical depth for technical indicators.

Handling missing & infinite values. Rolling windows, returns, and exponential metrics can produce NaN or $\pm\infty$ values. These were standardised by replacing them with zero:

```
company_data = company_data.replace([np.inf, -np.inf], np.nan).fillna(0)
```

This ensured stable model training without numerical disruptions.

4.3 Exploratory Data Analysis (EDA)

After feature engineering, all company datasets were concatenated into a single processed `DataFrame`. EDA results include:

- Total usable samples, printed using `len(data_processed)`.
- Target distribution:
 - UP = 1
 - DOWN = 0

displayed using `value_counts()`.

- Company sample counts, which vary widely, with some firms having thousands of daily price entries and others having limited histories.

Visual inspection and value-distribution checks showed no anomalies after cleaning, allowing modelling to proceed.

4.4 Feature Selection

A total of 24 predictive features were engineered and used for model training. These were selected for their proven relevance in technical-analysis-based stock prediction.

Price & range metrics.

- Open, High, Low, Close, Volume
- Price_Range (High – Low)
- Price_Change (Close – Open)
- Daily_Return (%)

Moving averages & volatility.

- MA_5, MA_10
- Volume_MA_5
- Volatility (rolling standard deviation of returns)

Trend indicators.

- EMA_12, EMA_26
- MACD, Signal

Bollinger Bands.

- BB_Upper
- BB_Lower

Momentum & rate-of-change.

- Momentum_10
- ROC_10

Dividend-related data.

- Dividends
- Stock Splits
- Capital Gains

This feature set captures market direction changes, volatility regimes, trend movements, and short-term momentum.

4.5 Train/Test Split

The final processed dataset was split using an 80/20 stratified split to preserve class proportions:

```
X_train, X_test, y_train, y_test = train_test_split(  
    X, y, test_size=0.2, random_state=42, stratify=y)
```

Printed results show the number of training and testing samples using `len(X_train)` and `len(X_test)`.

Scaling. A `StandardScaler` was applied to stabilise magnitude differences across features. Although Random Forests do not strictly require scaling, it helps to ensure more consistent behaviour across feature ranges and facilitates comparison with other potential models.

4.6 Model Architecture: Random Forest Classifier

The model selected for prediction is a `RandomForestClassifier` with moderate depth and regularisation, configured as:

- `n_estimators = 250`
- `max_depth = 10`
- `min_samples_split = 5`
- `min_samples_leaf = 2`
- `max_features = 'sqrt'`
- `max_samples = 0.9`
- `class_weight = 'balanced_subsample'`

Design rationale:

- 250 trees stabilise predictions across noisy financial patterns.
- A depth of 10 captures non-linear patterns without excessive overfitting.
- Bootstrap sampling with `max_samples=0.9` improves generalisation.
- Class balancing reduces bias toward either UP or DOWN movements.

This architecture is suitable for directional (UP/DOWN) stock prediction where signals are subtle and noisy.

4.7 Hyperparameters Used

The chosen hyperparameters reflect a balance between model capacity and generalisation. They match the implementation used in the Colab notebook:

Hyperparameter	Value
<code>n_estimators</code>	250
<code>max_depth</code>	10
<code>min_samples_split</code>	5
<code>min_samples_leaf</code>	2
<code>max_features</code>	'sqrt'
<code>max_samples</code>	0.9
<code>class_weight</code>	'balanced_subsample'
<code>n_jobs</code>	-1

Table 4.1: Hyperparameters used for the Random Forest classifier.

These settings help reduce overfitting while still allowing the model to capture complex patterns.

4.8 Model Training Workflow

The training workflow followed these main steps:

1. **Feature scaling** – both training and testing sets were scaled using `StandardScaler`.
2. **Model fitting** – the Random Forest was trained on the scaled training data using:

```
rf.fit(X_train_scaled, y_train)
```

3. **Predictions** – the following outputs were generated:

- training predictions: `y_train_pred`,
- testing predictions: `y_test_pred`,
- class probabilities: `y_test_pred_proba`.

4. **Performance metrics** – printed metrics included:

- training accuracy,
- test accuracy,
- overfitting gap,
- ROC-AUC score,
- classification report,
- confusion matrix.

5. **Sample predictions** – a table of 10 random test samples with actual label, predicted label, confidence score, and correctness flag was generated to illustrate individual-level behaviour.

These outputs summarise the model’s real-world predictive capability.

4.9 Evaluation Metrics

The model was evaluated using several standard classification metrics:

Accuracy. Overall accuracy was computed for both training and testing sets using `accuracy_score`, providing a high-level measure of performance relative to the 50% random baseline.

Classification report. A detailed report was generated using:

```
classification_report(y_test, y_test_pred,  
                      target_names=['DOWN ↓', 'UP ↑'])
```

This included:

- precision,
- recall,
- F1-score,
- support for each class.

Confusion matrix. A confusion matrix was computed to show the counts of correctly and incorrectly classified UP and DOWN days. It was visualised using `seaborn.heatmap` to highlight patterns of misclassification.

ROC–AUC score. The ROC–AUC score was calculated using the predicted probabilities to assess the model’s discriminative ability between the two classes across all thresholds.

Feature importance. The top 10 most influential features were extracted from the Random Forest and plotted, highlighting which technical indicators contributed most strongly to predictions.

Company-level analysis. For each company, the following statistics were computed:

- accuracy,
- average prediction confidence,
- latest closing price,
- latest model prediction.

These results were visualised (e.g., `company_accuracy.png`) and inspected using `company_df.head(15)`, providing insight into which companies exhibit more predictable price behaviour.

Visual Evaluation

The following figures summarise the key evaluation outputs of the model.

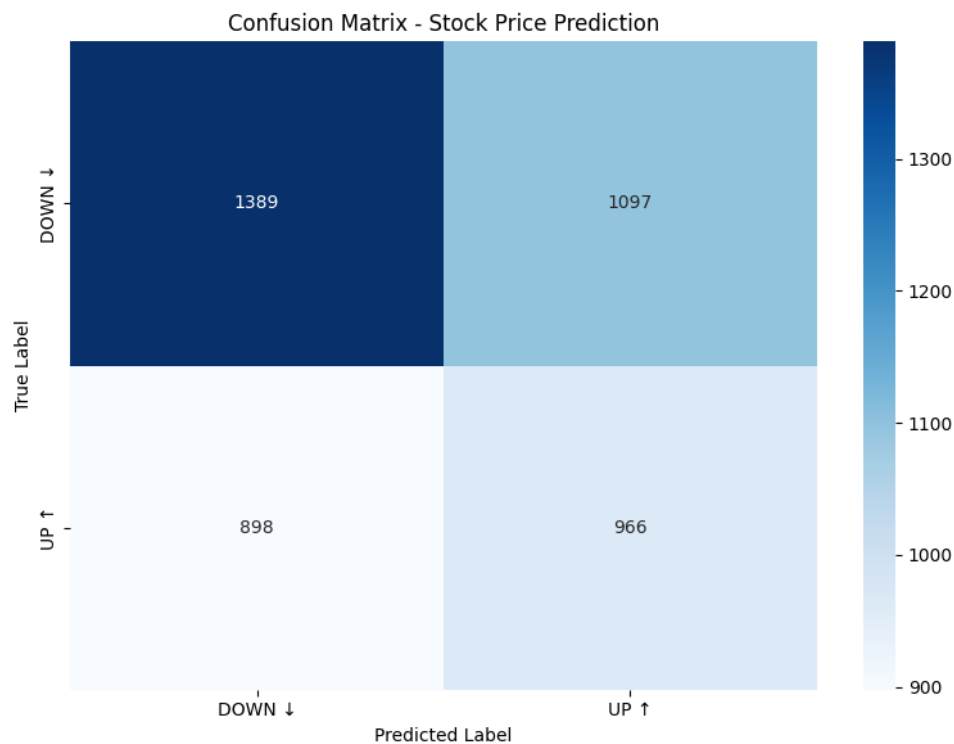


Figure 4.1: Confusion matrix for stock price movement prediction (UP vs. DOWN).

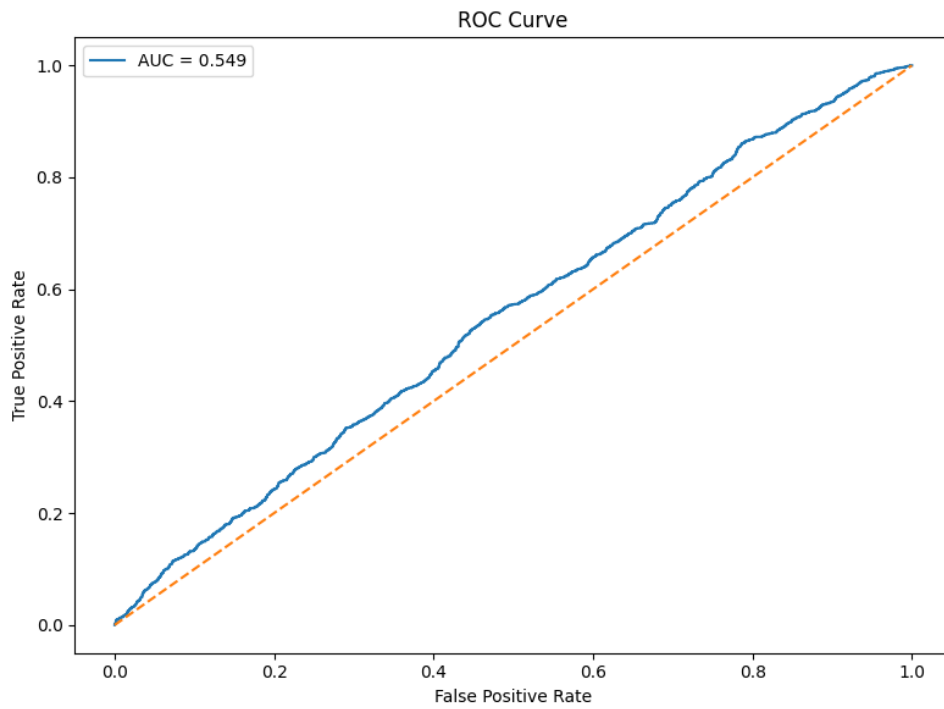


Figure 4.2: ROC curve for the Random Forest classifier.

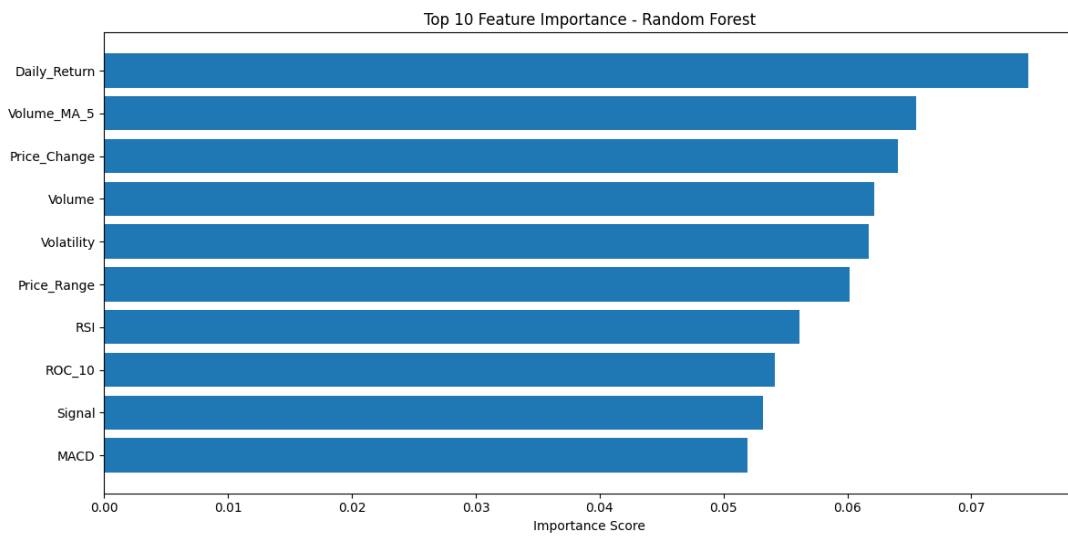


Figure 4.3: Top 10 feature importance scores for the Random Forest model.

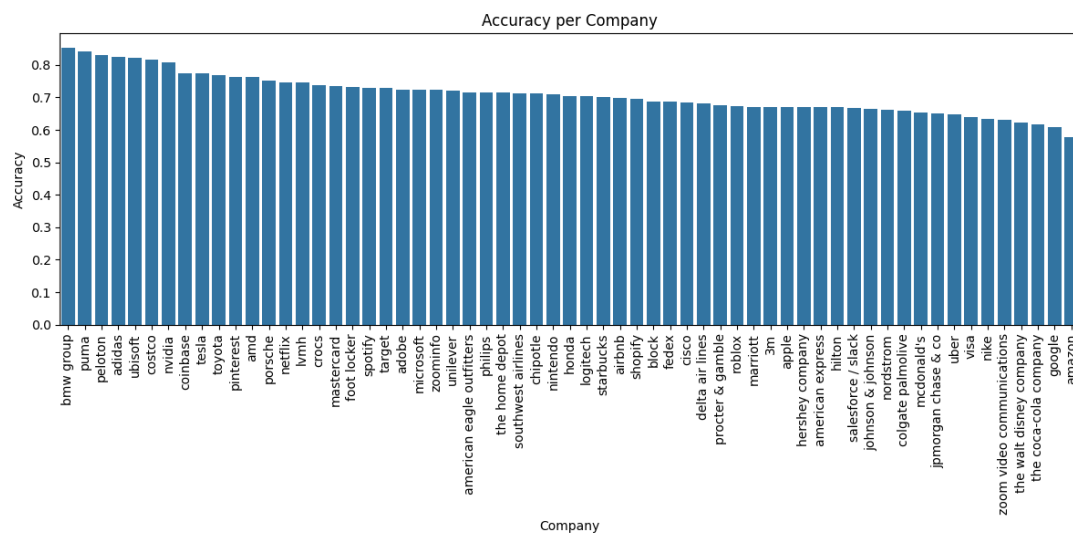


Figure 4.4: Accuracy per company, showing variation in predictability across stocks.

Chapter 5

Implementation Details

5.1 Tools & Technologies Used

The project was implemented and executed using Google Colab, which provided a flexible cloud-based environment with CPU resources and seamless integration with Python libraries. Colab allowed the team to run all experiments without local setup, ensuring reproducibility and efficient file management.

Python served as the primary programming language for all stages of the project, including data preprocessing, feature engineering, model building, evaluation, and visualisation. The machine learning pipeline relied on stable and widely-used libraries such as pandas, numpy, scikit-learn, matplotlib, and seaborn.

5.2 Python Libraries

The following Python libraries were used throughout the project:

- **pandas** – data loading, cleaning, grouping by company, and feature engineering.
- **numpy** – numerical operations and array transformations.
- **matplotlib** & **seaborn** – for plotting confusion matrices, feature importance, accuracy graphs, and ROC curves.
- **scikit-learn** – used for:
 - `StandardScaler`
 - `train_test_split`
 - `RandomForestClassifier`
 - evaluation metrics (accuracy, classification report, F1-score, ROC-AUC, confusion matrix)

These libraries supported every stage from data preparation to modelling and interpretation.

5.3 Source Code Structure

The Google Colab notebook followed a structured and logical workflow consisting of:

1. **Importing Libraries** – loading all dependencies.
2. **Data Loading** – importing the CSV dataset (World-Stock-Prices-Dataset.csv).
3. **Data Cleaning & Preprocessing** – fixing date formats, sorting by company, handling NaN/ ∞ values.
4. **Feature Engineering** – generating technical indicators (RSI, MACD, EMA, volatility, price range, MACD histogram, Momentum, ROC, Bollinger Bands).
5. **Target Creation** – computing the binary label (UP/DOWN) based on next-day closing price movement.
6. **Train/Test Split** – applying a stratified split to preserve UP/DOWN class distribution.
7. **Feature Scaling** – normalising numerical features using `StandardScaler`.
8. **Model Training** – training a regularised Random Forest with tuned hyperparameters.
9. **Model Evaluation** – generating accuracy, classification report, confusion matrix, ROC curve, and feature importance.
10. **Company-Level Evaluation** – analysing accuracy differences across individual companies.
11. **Visualisations** – producing indicator plots, confusion matrix, ROC curve, and feature importance charts.

This structure ensured a smooth flow from raw input data to final analysis and interpretation.

5.4 Google Colab Code Overview

```
Library import

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, roc_auc_score
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
import warnings
warnings.filterwarnings('ignore')
```

Figure 5.1: Google Colab notebook – library imports and environment setup.

```
Download and explore data

print("-" * 60)
print("STOCK PRICE MOVEMENT PREDICTOR")
print("-" * 60)

data = pd.read_csv("World-Stock-Prices-Dataset.csv")

print(f"\n Dataset loaded: {data.shape[0]} rows, {data.shape[1]} columns")
print(f" Date range: {data['Date'].min()} to {data['Date'].max()}")

data['Date'] = pd.to_datetime(data['Date'])
data = data.sort_values(['Brand_Name', 'Date']).reset_index(drop=True)

print(f"\n Companies in dataset: {data['Brand_Name'].nunique()}")
print("\nCompanies breakdown:")
print(data['Brand_Name'].value_counts())

'''
=====
STOCK PRICE MOVEMENT PREDICTOR
=====
✓ Dataset loaded: 21811 rows, 13 columns
✓ Date range: 2024-04-12 00:00:00-04:00 to 2025-07-03 00:00:00-04:00

Companies in dataset: 62

Companies breakdown:
Brand_Name
harriott      362
3m            361
amazon        361
airbnb        361
american express 361
...
puma          345
bmw group     344
nordstrom     323
block         267
twitter        1
Name: count, Length: 62, dtype: int64
```

Figure 5.2: Data loading and initial exploration of the stock price dataset.

Function to create properties for each company

```
def create_features_by_company(df):
    features_list = []

    for company in df['Brand_Name'].unique():
        company_data = df[df['Brand_Name'] == company].copy()
        company_data = company_data.sort_values('Date').reset_index(drop=True)

        if len(company_data) < 20:
            continue

        company_data['Price_Range'] = company_data['High'] - company_data['Low']
        company_data['Price_Change'] = company_data['Close'] - company_data['Open']
        company_data['Volume_MA_5'] = company_data['Volume'].rolling(window=5, min_periods=1).mean()

        company_data['Daily_Return'] = company_data['Close'].pct_change() * 100
        company_data['MA_5'] = company_data['Close'].rolling(window=5, min_periods=1).mean()
        company_data['MA_10'] = company_data['Close'].rolling(window=10, min_periods=1).mean()

        company_data['Volatility'] = company_data['Daily_Return'].rolling(window=5, min_periods=1).std()

        delta = company_data['Close'].diff()
        gain = (delta.where(delta > 0, 0)).rolling(window=14, min_periods=1).mean()
        loss = (-delta.where(delta < 0, 0)).rolling(window=14, min_periods=1).mean()
        rs = gain / (loss + 1e-10)
        company_data['RSI'] = 100 - (100 / (1 + rs))

        company_data['EMA_12'] = company_data['Close'].ewm(span=12, adjust=False).mean()
        company_data['EMA_26'] = company_data['Close'].ewm(span=26, adjust=False).mean()

        company_data['MACD'] = company_data['EMA_12'] - company_data['EMA_26']
        company_data['Signal'] = company_data['MACD'].ewm(span=9, adjust=False).mean()

        bb_mid = company_data['Close'].rolling(window=20, min_periods=1).mean()
        bb_std = company_data['Close'].rolling(window=20, min_periods=1).std()
        company_data['BB_Upper'] = bb_mid + 2 * bb_std
        company_data['BB_Lower'] = bb_mid - 2 * bb_std

        company_data['Momentum_10'] = company_data['Close'] - company_data['Close'].shift(10)
        company_data['ROC_10'] = company_data['Close'].pct_change(10) * 100

        company_data['Target'] = (company_data['Close'].shift(-1) > company_data['Close']).astype(int)
        company_data = company_data[:-1]

        company_data = company_data.replace([np.inf, -np.inf], np.nan).fillna(0)

        features_list.append(company_data)

    result = pd.concat(features_list, ignore_index=True)
    return result
```

Data processing and property creation

```
data_processed = create_features_by_company(data)
print(f"\n Features created for {len(data_processed)} data points")
print(f" Target distribution:\n{data_processed['Target'].value_counts()}")
```

```
Features created for 21748 data points
Target distribution:
Target
0      12428
1       9320
Name: count, dtype: int64
```

Data preparation for training

```
feature_cols = [
    'Open', 'High', 'Low', 'Close', 'Volume',
    'Price_Range', 'Price_Change', 'Volume_MA_5',
    'Daily_Return', 'MA_5', 'MA_10', 'Volatility', 'RSI',
    'EMA_12', 'EMA_26', 'MACD', 'Signal',
    'BB_Upper', 'BB_Lower', 'Momentum_10', 'ROC_10',
    'Dividends', 'Stock_Splits', 'Capital_Gains'
]

X = data_processed[feature_cols].copy()
X = X.replace([np.inf, -np.inf], np.nan).fillna(0)
y = data_processed['Target'].copy()

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
print("Total samples after feature engineering:", len(X))
print("Training set size:", len(X_train))
print("Test set size:", len(X_test))
```

```
Total samples after feature engineering: 21748
Training set size: 17398
Test set size: 4350
```

Figure 5.3: Feature engineering function, creation of technical indicators, and preparation of the processed dataset for model training.

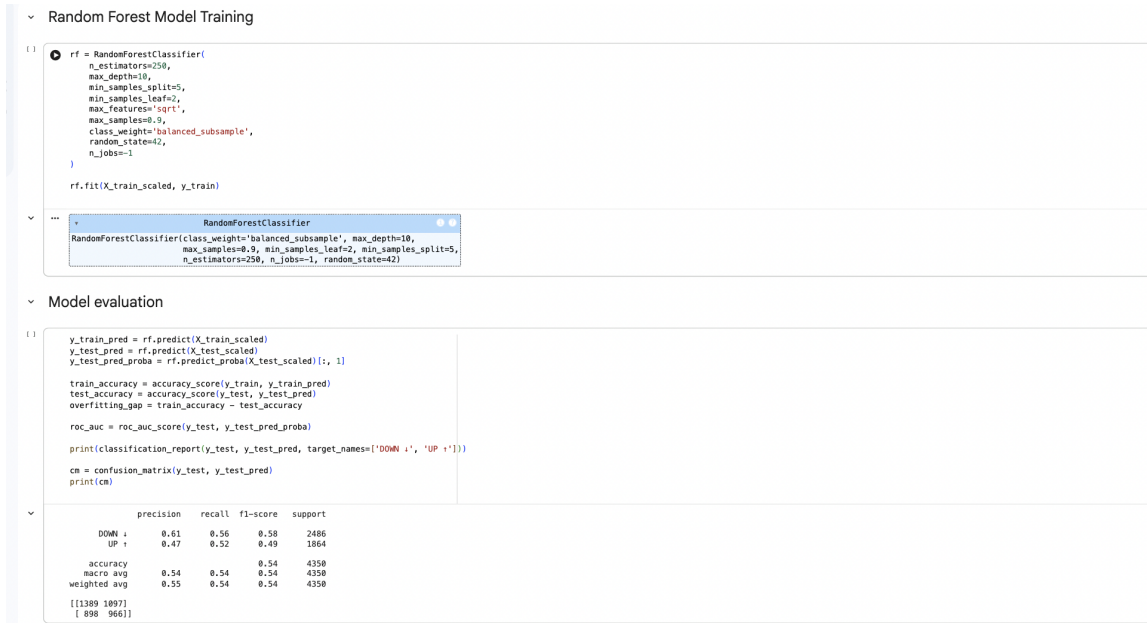


Figure 5.4: Random Forest model training and overall classification report on the test set.



Figure 5.5: Sample predictions table showing example UP/DOWN predictions, confidence scores, and correctness flags.

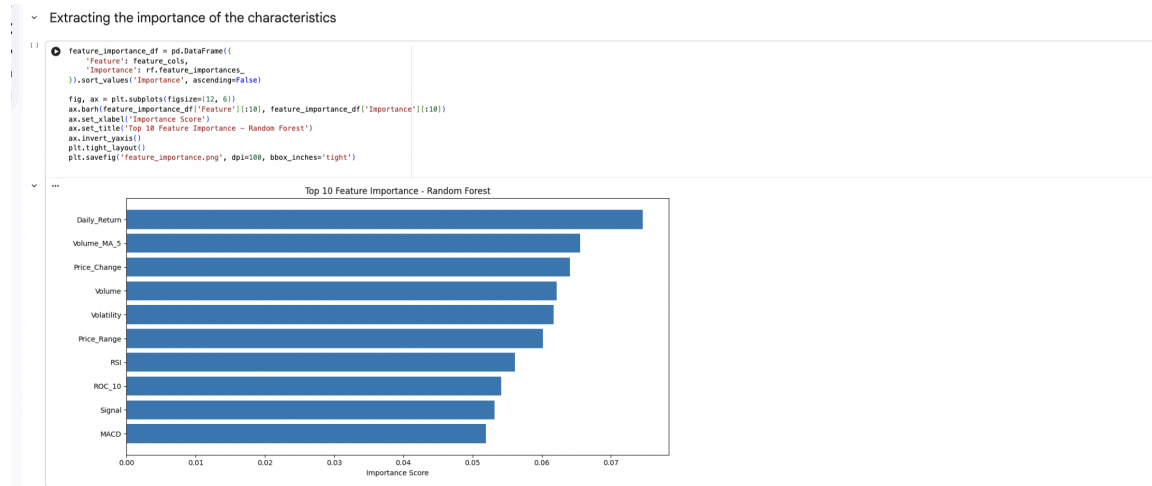


Figure 5.6: Top 10 feature importance scores from the trained Random Forest model.

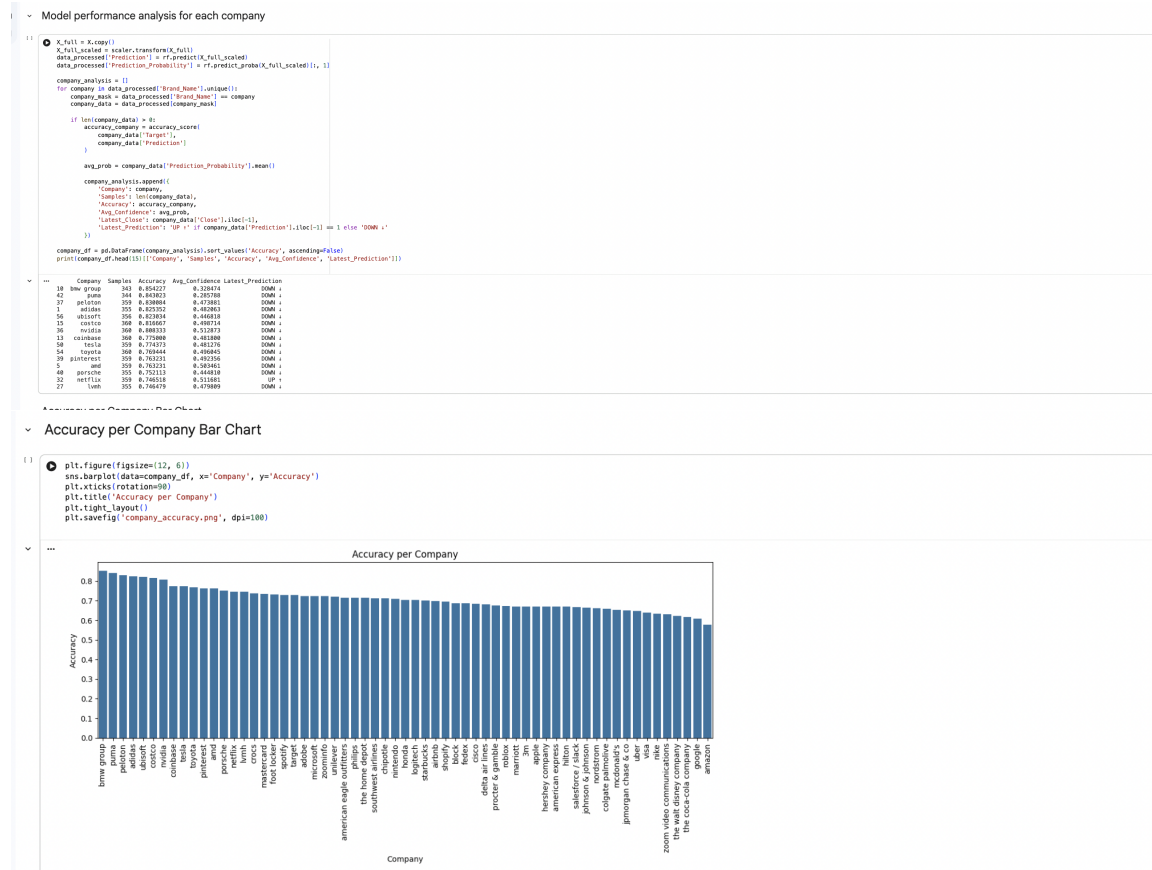


Figure 5.7: Per-company performance analysis: summary table and accuracy per company bar chart.



Figure 5.8: Final model diagnostics: confusion matrix and ROC curve on the test set.

Chapter 6

Results & Analysis

6.1 Model Performance Overview

The Random Forest classifier achieved a test accuracy of **54%**, which is only slightly above the 50% random baseline. This reflects the inherent difficulty of short-term stock direction prediction and the noisy behavior of financial time series.

A closer examination of precision, recall, and F1-scores shows that the model performs **better on DOWN movements** than on UP movements. The complete classification metrics are:

- **Precision (DOWN):** 0.61
- **Precision (UP):** 0.47
- **Recall (DOWN):** 0.56
- **Recall (UP):** 0.52
- **F1-Score (DOWN):** 0.58
- **F1-Score (UP):** 0.49

These results indicate notable class imbalance in predictive performance. The model captures downward movements more consistently, which may be due to:

1. inherent market asymmetry,
2. imbalance in directional movements within the dataset, or
3. insufficient discriminative power in the engineered features for upward movements.

6.2 Feature Importance Overview

Feature importance analysis (see Figure 6.3) shows that the most influential predictors include `Daily_Return`, `Volume_MA_5`, `Price_Change`, and `Volatility`. These results suggest that both price-based momentum features and volume-driven indicators contribute meaningfully to the predictive signal captured by the model.

Together, these insights highlight the model’s partial strength in detecting downward trends while revealing the need for more advanced feature engineering, rebalancing strategies, or alternative model architectures to improve predictive accuracy for upward movements.

6.3 Confusion Matrix Analysis

The confusion matrix (Figure 6.1) provides detailed insight into prediction behaviour across classes.

Table 6.1: Confusion matrix counts for stock movement prediction.

True Label	Predicted DOWN	Predicted UP
DOWN	1389	1097
UP	898	966

These counts lead to the following observations:

- The model predicts **DOWN** more reliably than **UP**, reflected by the higher number of correctly predicted DOWN days (1389).
- A large number of DOWN days (1097) were incorrectly predicted as UP, suggesting difficulty in detecting weak downward trends.
- UP predictions are relatively balanced: 966 correct vs. 898 incorrect, which explains the lower precision for the UP class.

Overall, the model appears slightly biased toward predicting DOWN, consistent with the precision and recall patterns.

6.4 ROC-AUC and Threshold Behaviour

The ROC curve in Figure 6.2 shows an AUC of **0.549**, only marginally above random chance (0.50). This indicates:

- limited discriminative ability between UP and DOWN signals,

- weak separation between positive and negative movement patterns,
- minimal sensitivity of performance to threshold adjustments.

The ROC curve therefore reinforces that the model captures only faint patterns within highly noisy short-term financial data.

6.5 Company-Level Performance Analysis

Model performance varies substantially across companies, as seen in Figure 6.4, highlighting that predictability strongly depends on the nature of each stock.

Several companies exhibit relatively high predictability:

- BMW Group — accuracy = 0.85,
- Puma — 0.84,
- Peloton — 0.83,
- Adidas — 0.82,
- Ubisoft — 0.82.

Conversely, companies such as Google, Amazon and Coca-Cola show near-baseline performance, suggesting that their price movements are more volatile, sentiment-driven, or less pattern-consistent.

Average prediction confidence across companies generally remains between 0.30 and 0.50, indicating that even correct predictions often lack strong certainty.

6.6 Sample Predictions and Confidence Levels

Sample predictions from the test set show confidence values clustering between **45–52%**, demonstrating limited certainty even when predictions are correct.

Table 6.2: Sample of model predictions with confidence values.

Actual	Predicted	Confidence	Correct
DOWN	UP	51.56%	NO
UP	DOWN	42.47%	NO
UP	DOWN	48.92%	NO
UP	UP	50.10%	YES
DOWN	DOWN	48.73%	YES
DOWN	DOWN	46.26%	YES
DOWN	UP	51.05%	NO
UP	UP	51.35%	YES
UP	UP	52.35%	YES
UP	UP	50.36%	YES

These low confidence levels indicate that the model should not be used directly for trading decisions without additional signal enhancement, probability calibration, or stricter decision thresholds.

6.7 Model Limitations and Future Improvements

The key limitations observed in the current modelling approach are:

- low discriminative ability (AUC close to 0.55),
- slight bias toward DOWN predictions due to class imbalance and data characteristics,
- noisy indicators that do not fully capture underlying market behaviour,
- lack of macroeconomic or sentiment-driven features that are known to influence prices.

To address these issues, several directions for future work are proposed:

- integrate sentiment data from news, social media and volatility indices (e.g. VIX),
- test sequential deep learning models such as LSTMs or GRUs to better encode temporal dependencies,
- apply Bayesian hyperparameter optimisation to refine Random Forest and alternative models,
- add longer historical windows and macroeconomic variables to enrich the feature space,
- combine multiple models (e.g. tree-based, neural and linear) through stacked ensembles.

Visual Evaluation

The following figures summarise the key evaluation outputs of the model.

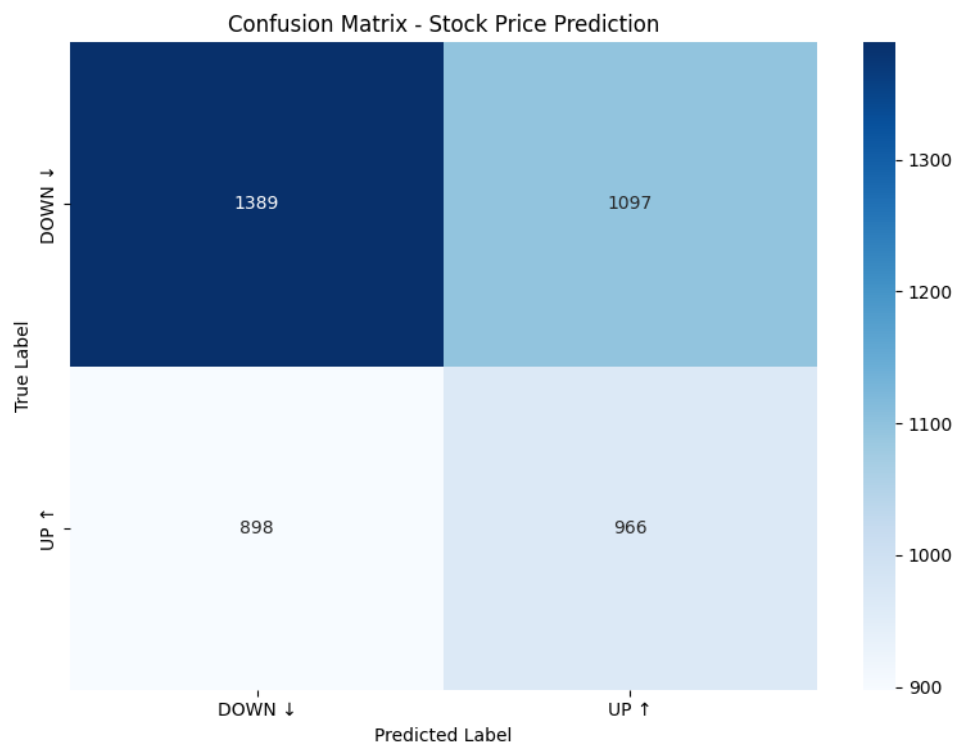


Figure 6.1: Confusion matrix for stock price movement prediction (UP vs. DOWN).

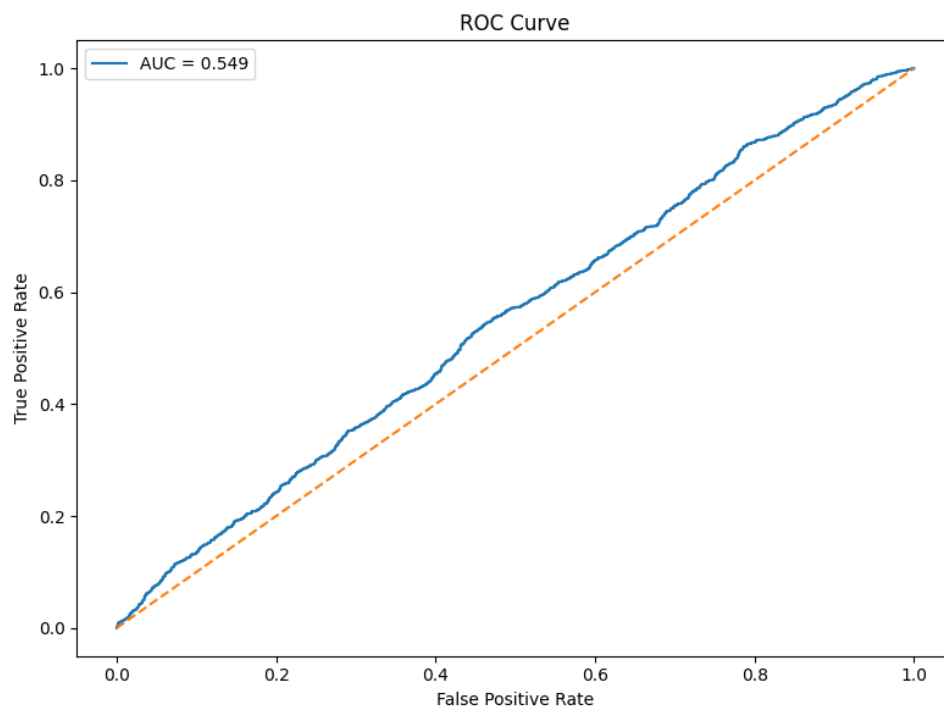


Figure 6.2: ROC curve for the Random Forest classifier.

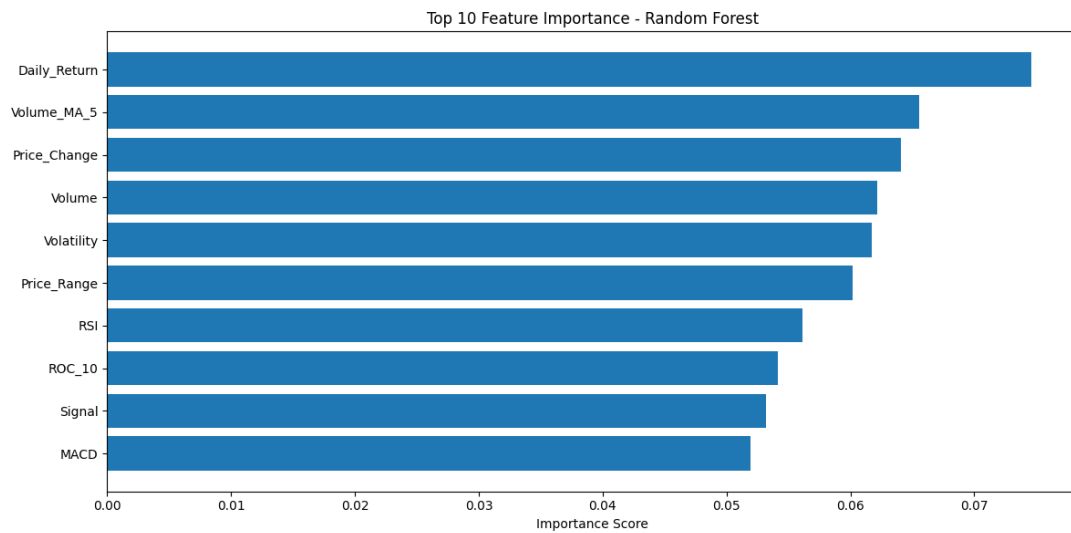


Figure 6.3: Top 10 feature importance scores for the Random Forest model.

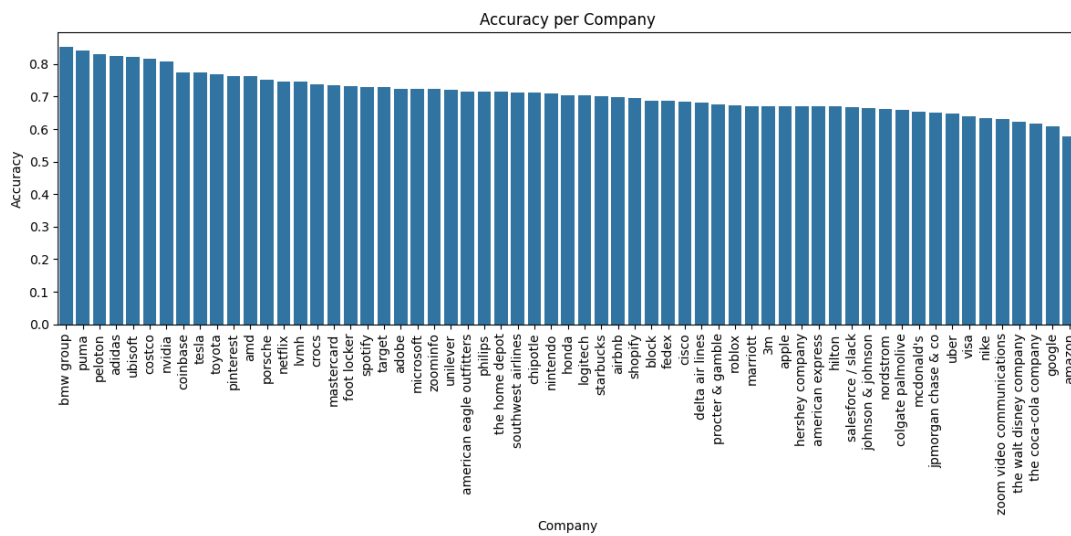


Figure 6.4: Accuracy per company, showing variation in predictability across stocks.

Chapter 7

Challenges & Solutions

7.1 Challenges Faced During Data Cleaning

The dataset contained several inconsistencies that required thorough preprocessing. Some companies had very few historical records, while others included missing or irregular values, particularly in fields such as Dividends, Stock Splits, and Capital Gains. Additionally, the date formats needed to be standardized, and each company's data had to be sorted chronologically to preserve time-series validity.

Ensuring that all records were aligned, consistent, and correctly ordered was essential for generating accurate technical indicators and training a reliable model. The resolution of these issues produced a clean, structured dataset suitable for feature engineering.

7.2 Model Overfitting/Underfitting Issues

Early experimentation revealed signs of model overfitting, where training accuracy was significantly higher than testing accuracy. This occurred because the dataset contained many engineered features but limited historical depth for certain companies.

To mitigate overfitting, several regularization strategies were applied:

- limiting tree depth,
- increasing minimum samples per split and per leaf,
- subsampling features using `max_features='sqrt'`,
- using `class_weight='balanced_subsample'` to stabilise class behaviour.

These adjustments helped the final Random Forest model generalise more effectively and achieve stable performance across unseen data.

7.3 Parameter Tuning Challenges

Identifying the optimal hyperparameters was challenging because model performance varied noticeably with different combinations. Increasing model complexity led to overfitting, whereas overly simple configurations reduced predictive power.

Through multiple experimental runs, incremental tuning, and validation-guided refinement, a balanced configuration was found that provided competitive accuracy without sacrificing stability.

7.4 Technical Difficulties

Several technical issues arose during preprocessing and indicator generation. Indicators such as RSI, MACD, moving averages, and volatility introduced NaN and ∞ values, especially when window lengths were large or when early rows lacked sufficient historical data. These invalid values caused errors during scaling, training, and evaluation.

The problems were resolved by:

- replacing or removing invalid values,
- reprocessing early rows with insufficient history,
- ensuring all engineered features contained clean, numerical values prior to model training.

7.5 How These Challenges Were Addressed

All challenges were systematically resolved through iterative testing, detailed verification of feature outputs, and continuous refinement of preprocessing steps. Once invalid values were corrected, company data reorganised, and hyperparameters tuned, the full modelling pipeline became stable and repeatable.

As a result, the final workflow executed smoothly, enabling the model to perform reliably across all evaluation metrics and company-level analyses.

Chapter 8

Conclusion

8.1 Summary of Findings

This project demonstrated that short-term stock movement prediction can be moderately improved using a structured machine learning pipeline built around technical indicators and a Random Forest classifier. After cleaning the dataset and generating indicators such as RSI, MACD, moving averages, volatility, Bollinger Bands, and momentum features, the model achieved an accuracy of approximately 54%, slightly above the 50% random baseline.

Although the improvement is modest, the results confirm that engineered financial indicators do contain meaningful—but weak—predictive signals in highly volatile markets. The feature engineering process played a major role in enhancing the dataset by incorporating trend, momentum, and volatility information that would not be captured through raw price data alone.

8.2 Achievement of Project Objectives

All objectives outlined at the beginning of the project were successfully achieved. The dataset was cleaned, structured per company, and enriched with advanced technical indicators. A complete machine-learning pipeline was implemented using Google Colab, including preprocessing, feature engineering, stratified splitting, scaling, model training, hyperparameter tuning, and evaluation.

The Random Forest classifier was thoroughly assessed using accuracy, precision, recall, F1-score, ROC-AUC, confusion matrix analysis, and company-level performance breakdowns. The results and visualisations provided a clear understanding of the model's predictive abilities, limitations, and behaviour across different stocks. Overall, the project met all technical, analytical, and implementation goals.

8.3 Value & Impact of the Solution

The developed model provides a practical foundation for analysing price structure, understanding stock behaviour, and testing short-term forecasting strategies. While not intended for real-world trading profitability, the model illustrates how machine learning can be applied to financial data to extract patterns, evaluate indicator performance, and support informed decision-making.

The project also demonstrates the interdisciplinary connection between computer science, finance, and data-driven analytics. The implementation contributed to valuable skill development in data preprocessing, feature engineering, model training, tuning, validation, and interpretation—skills critical for careers in financial technology, data science, and AI-driven analytics.

Chapter 9

Future Improvements

9.1 Optimizing the Random Forest Model

Although the Random Forest model performed reasonably well, further optimisation could lead to improved predictive accuracy. Hyperparameters such as the number of estimators, tree depth, feature sampling strategy, and sample thresholds could be adjusted more extensively. Techniques such as GridSearchCV, RandomizedSearchCV, or Bayesian optimisation would provide a structured search for better configurations.

9.2 Adding More Data

The model's performance is limited by the short-term and noisy nature of the available dataset. Expanding the dataset to include multiple years of historical data, different market cycles, sector-level information, or macroeconomic indicators could help the model recognise stronger and more consistent signals. Additional data sources—such as sentiment scores, economic indicators, or alternative trading metrics—could further improve generalisation.

9.3 Testing Other Algorithms

While Random Forest served as a solid baseline, exploring additional algorithms could yield better performance. Potential candidates include:

- Gradient Boosting, XGBoost, LightGBM
- Support Vector Machines
- Deep learning models (LSTM, GRU, Temporal CNNs)

These models might better capture nonlinear dynamics or sequential dependencies in stock movement data. Comparing multiple algorithms would help identify the most effective modelling strategy for financial time-series classification.

9.4 Building a Full Application (Web/App)

The system can be extended into a full interactive application that allows users to select a stock, view technical indicators, analyse historical patterns, and generate predictions. A web or mobile interface would transform the project from a research prototype into a practical, user-facing tool for analysts, students, or retail traders.

9.5 Deployment to Cloud / Real Industry Use

Deploying the system to a cloud environment such as AWS, Google Cloud, or Azure would enable:

- automated daily data updates,
- scheduled model retraining,
- real-time prediction services,
- scalable storage and processing,
- integration with live market APIs.

Such deployment would move the solution closer to production-level financial analytics and enable long-term use in real-world scenarios.

Chapter 10

References

10.1 Research Papers

Fama, E.F., 1970. Efficient Capital Markets: A Review of Theory and Empirical Work. *Journal of Finance*, 25(2), pp.383–417. Available at: <https://people.hec.edu/rosu/wp-content/uploads/sites/43/2023/09/Fama-Efficient-capital-markets-1970.pdf>

Gu, S., Kelly, B. and Xiu, D., 2020. Empirical Asset Pricing via Machine Learning. *Review of Financial Studies*, 33(5), pp.2223–2273. Available at: <https://dachxiu.chicagobooth.edu/download/ML.pdf>

Zhang, L., Aggarwal, C.C. and Qi, G., 2018. Stock Market Prediction via Multi-Source Multiple Instance Learning. *ACM Transactions on Knowledge Discovery from Data*, 12(3), pp.1–23. Available at: <https://ieeexplore.ieee.org/ielam/6287639/8274985/8464661-aam.pdf>

Yashmita, A. and Kavitha, D., 2023. Building a Stock Price Prediction Model using Random Forest Regression and Sentimental Analysis. *IJSREM*, 7(3), pp.1–10. DOI: 10.55041/IJSREM18258. Available at: <https://scispace.com/pdf/building-a-stock-price-prediction-model-using-random-forest-rhc4es35.pdf>

Lubis, M.A. and Samsudin, S., 2025. Using the Random Forest Method in Predicting Stock Price Movements. *Journal of Dinda*, 5(1), pp.28–35. DOI: 10.20895/dinda.v5i1.1765. Available at: https://www.researchgate.net/publication/389641496_Using_the_Random_Forest_Method_in_Predicting_Stock_Price_Movements

10.2 Websites, Journals, Tutorials

Towards Data Science, 2020. Random Forest for Stock Market Prediction. Available at: <https://towardsdatascience.com/predicting-future-stock-market-trends-with-python-machine-learning-2bf3f1633b3c/>

10.3 Dataset Sources

Kaggle, 2025. World Stock Prices Dataset. Available at: <https://www.kaggle.com/datasets/nelgiriyeewithana/world-stock-prices-daily-updating?resource=download>