Introduction

Introduction
Machine learning is now used in many areas, including financial markets.
However, Stock data is hard to work with because it changes quickly and there is a huge amount of information to look at. A previous study by Alzaman used data from the Toronto Stock Exchange to test if machine-learning models could predict whether a stock would rise or fall. Their results showed that the method worked for that market.
In this project, I used the same idea but with a different dataset. I collected S&P 500 stock data using the yfinance library and built a similar machine-learning model to predict stock movement. I explored the data, created the features, and trained the model following a workflow. The main goal of this study is to see how well these methods work on S&P 500 data and whether they can be used to understand general market behavior.

Machine learning techniques have increasingly become an appropriate measure across various fields. In modern, daily financial markets, there is much value in prediction, although predicting stock developments is more complicated than it seems. Prices change due to microtransactions and terabytes of data create new patterns every day. For example, research done by Alzaman found that machine learning techniques predict stock movement based on Toronto Stock Exchange information, confirming that financial sets are appropriate to use this technique since there's relevant patterning.

In this research, I used a similar idea through different datasets. S&P 500 historical data was acquired through the **yfinance** library, and using closing price as a predictive measure seeks prediction based on minimal change relative to prediction. The main goal of this study is to see how well these methods work on S&P 500 data and whether they can be used to understand general market behavior.

🧰 Methods

Methods
• Data Source: Daily closing prices for Apple (AAPL), Tesla (TSLA), Amazon (AMZN), Visa (V), and Microsoft (MSFT) were downloaded from Yahoo Finance using the yfinance API for the period 2017-01-01 to 2019-12-31.
• Data Cleaning: Missing values were removed and the time index was converted into a clean Date index.
• Feature Engineering: For the target stock (Apple), the prediction target was defined as the next day’s price. Input features included today’s price and 1- to 5-day percentage returns.

• Train–Test Split: Data was split into 80% training and 20% testing, with the split visualized on a time-series plot.
• Models: Two regression models were trained and evaluated:
  – Linear Regression
  – Random Forest Regressor (300 trees, random_state = 42)
• Evaluation Metrics: Model quality was assessed using Mean Absolute Error (MAE) and R^2 score, and by visually comparing predicted vs. actual prices on the test set.

🎯 Study Objective / Aim

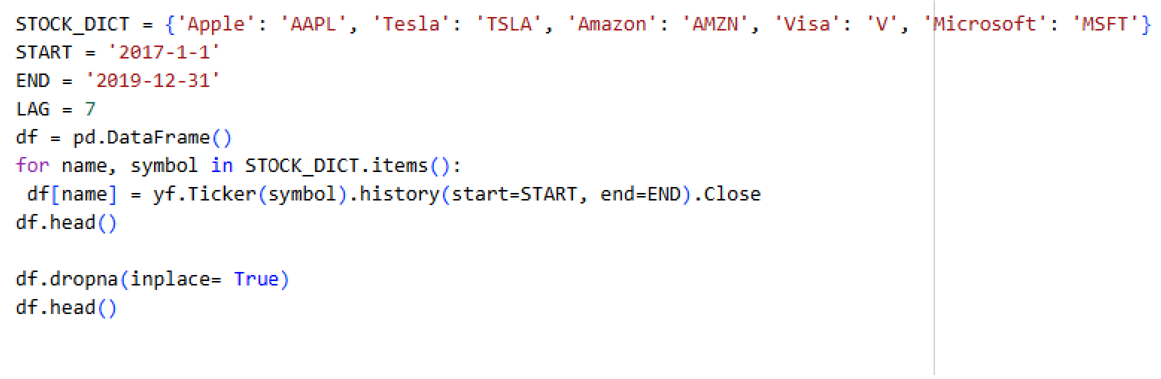
Study Objective/Aim
apply machine-learning methods to analyze historical S&P 500 stock prices.
visualize sector distribution, price trends, and correlations among five major companies.
build models that predict next-day Apple prices using return features.
Compare Linear Regression and Random Forest for accuracy.
evaluate strengths and limitations of models to understand stock market behavior.

📊 Findings (Main Block)

1️⃣ Price Trends (2017–2019)

1. Price Trends (2017–2019)
All five companies showed an overall upward trend in closing prices during the study period. Apple and Microsoft displayed relatively smoother and more consistent growth, while Tesla exhibited higher volatility. These trends highlight both the long-term growth of large technology firms and the short-term fluctuations that make prediction challenging.

Overall, for the length of the research period, the closing prices for all five companies trended higher. Apple's and Microsoft's trajectory is more stable and consistent than Tesla's fluctuations. This suggests a long-term outlook of big tech companies and a more short-term possibility that complicates predictions.



```
STOCK_DICT = {'Apple': 'AAPL', 'Tesla': 'TSLA', 'Amazon': 'AMZN', 'Visa': 'V', 'Microsoft': 'MSFT'}
START = '2017-1-1'
END = '2019-12-31'
LAG = 7
df = pd.DataFrame()
for name, symbol in STOCK_DICT.items():
 df[name] = yf.Ticker(symbol).history(start=START, end=END).Close
df.head()

df.dropna(inplace= True)
df.head()
```

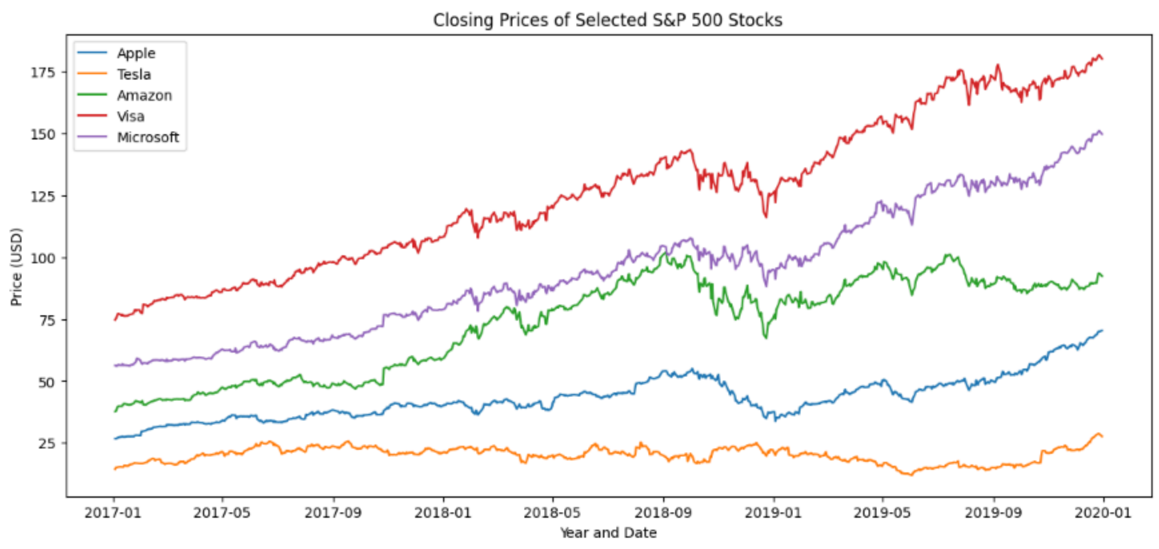
...

Date	Apple	Tesla	Amazon	Visa	Microsoft
2017-01-03 00:00:00-05:00	26.770882	14.466000	37.683498	74.684151	56.299316
2017-01-04 00:00:00-05:00	26.740915	15.132667	37.859001	75.294792	56.047413
2017-01-05 00:00:00-05:00	26.876902	15.116667	39.022499	76.177818	56.047413
2017-01-06 00:00:00-05:00	27.176542	15.267333	39.799500	77.229996	56.533222
2017-01-09 00:00:00-05:00	27.425463	15.418667	39.846001	76.797844	56.353298

```
# remove time from the index
df.reset_index(inplace=True) # Date becomes a column
df['Date'] = df.Date.dt.date # Apply dt.date to Date column
df.set_index('Date', inplace=True)
df.head()
```

Date	Apple	Tesla	Amazon	Visa	Microsoft
2017-01-03	26.770882	14.466000	37.683498	74.684151	56.299316
2017-01-04	26.740915	15.132667	37.859001	75.294792	56.047413
2017-01-05	26.876902	15.116667	39.022499	76.177818	56.047413
2017-01-06	27.176542	15.267333	39.799500	77.229996	56.533222
2017-01-09	27.425463	15.418667	39.846001	76.797844	56.353298

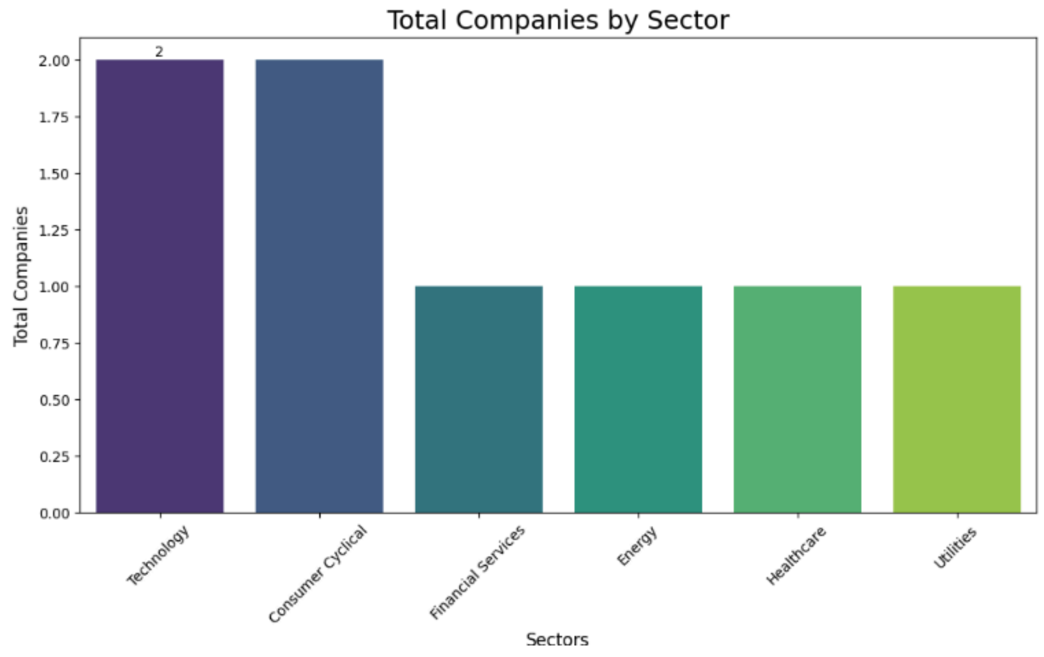
```
plt.figure(figsize=(14,6))
for col in df.columns:
    plt.plot(df[col], label=col)

plt.title("Closing Prices of Selected S&P 500 Stocks")
plt.xlabel("Year and Date")
plt.ylabel("Price (USD)")
plt.legend()
plt.show()
```

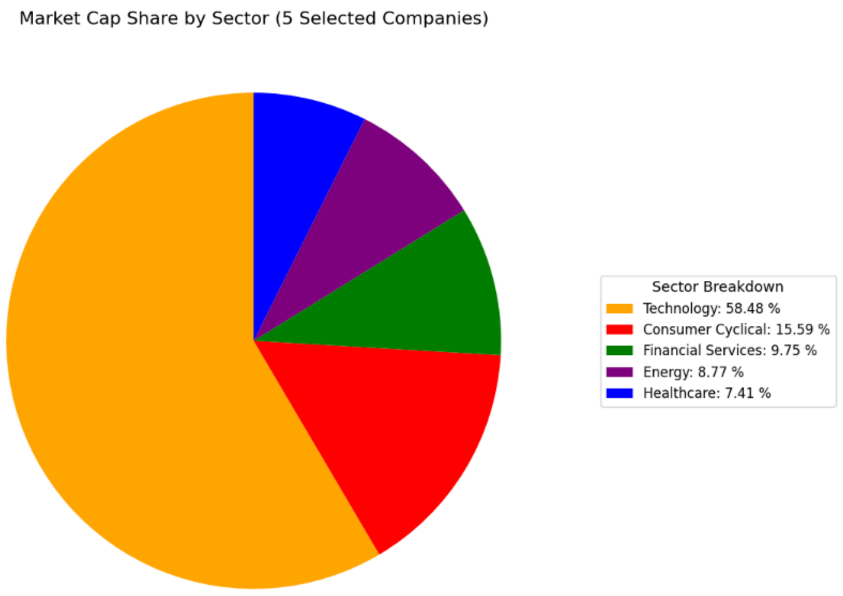


2 Sector Distribution and Market Capitalization

	Symbol	Company	Sector	Marketcap
0	AAPL	Apple	Technology	3.000000e+12
1	TSLA	Tesla	Consumer Cyclical	8.000000e+11
2	AMZN	Amazon	Consumer Cyclical	1.500000e+12
3	V	Visa	Financial Services	5.000000e+11
4	MSFT	Microsoft	Technology	3.200000e+12
5	XOM	ExxonMobil	Energy	4.500000e+11
6	JNJ	Johnson & Johnson	Healthcare	3.800000e+11
7	NEE	NextEra Energy	Utilities	1.700000e+11

2. Sector Distribution and Market Capitalization
The selected stocks span Technology 58.48%, Consumer Cyclical15.59%, Financial Services9.75%, Energy8.77%, Healthcare7.4%, and Utilities. Technology companies (Apple and Microsoft) contributed the largest share of total market capitalization in the sample, illustrating the dominant role of large tech firms in the modern S&P 500.
A sector countplot and market-cap pie and bar chart visually summarized how concentrated the sample is in technology-related sectors.





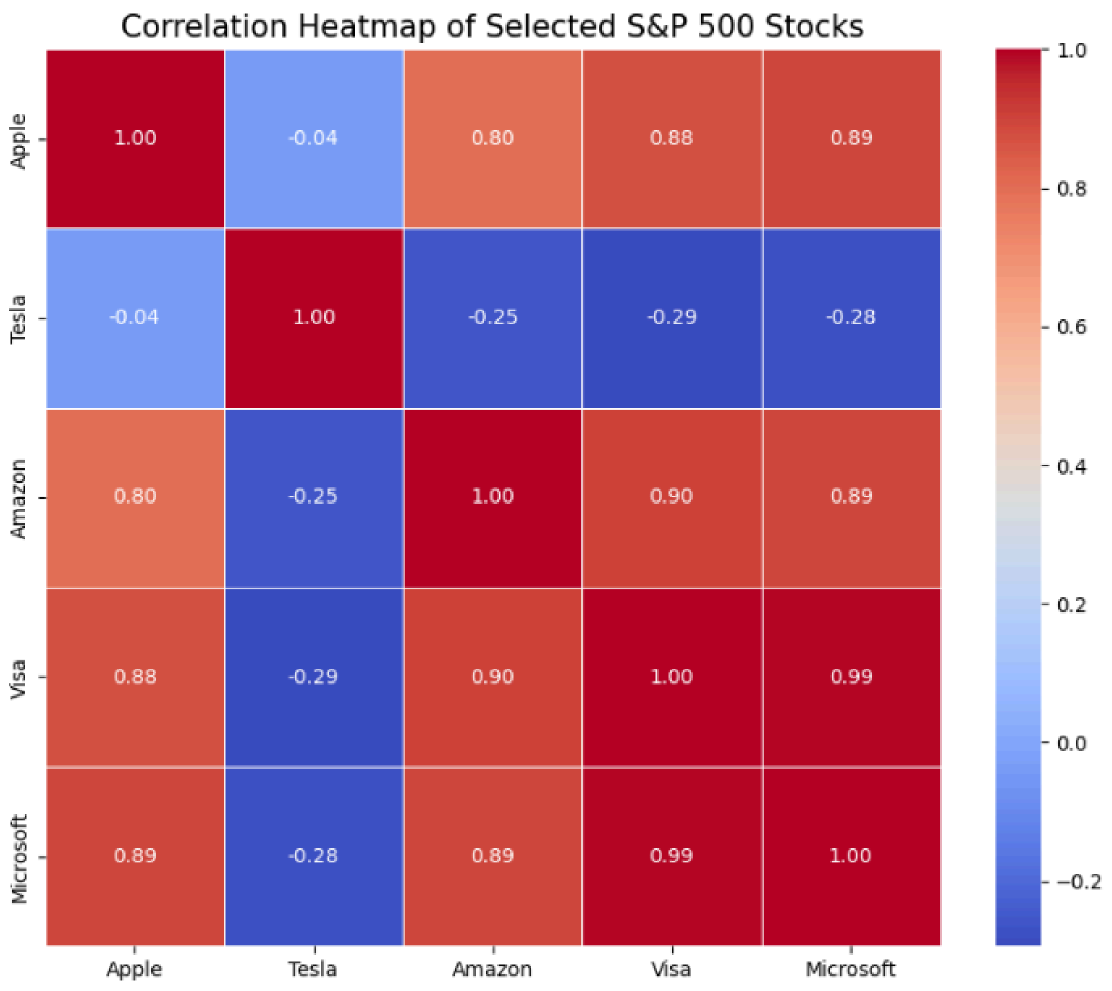
3️⃣ Correlation Among Selected Stocks

3. Correlation Among Selected Stocks
A correlation heatmap of daily closing prices showed strong positive correlation between Apple and Microsoft, and moderate correlations between the other large-cap stocks. Visa and Tesla were less correlated with the core tech names, reflecting differences in industry and risk profile. These correlations suggest that some stocks tend to move together, which is important for diversification and portfolio design.

The correlation heat map of adjusted daily closing prices showed the greatest positive correlation between Apple and Microsoft as well as a mediocre correlation among the other large-cap stocks with the exception of Visa and Tesla, who were less correlated to this group of major tech company names; this makes sense for industry and risk.These correlations suggest that some stocks tend to move together, which is important for diversification and portfolio design.

```python
df.columns = ['Apple', 'Tesla', 'Amazon', 'Visa', 'Microsoft']
import matplotlib.pyplot as plt
import seaborn as sns

# Correlation of your 5 stocks
corr = df.corr()

plt.figure(figsize=(10, 8))
sns.heatmap(
    corr,
    annot=True,
    cmap='coolwarm',
    fmt='.2f',
    linewidths=0.5,
    square=True
)

plt.title("Correlation Heatmap of Selected S&P 500 Stocks", fontsize=15)
plt.show()
```

...

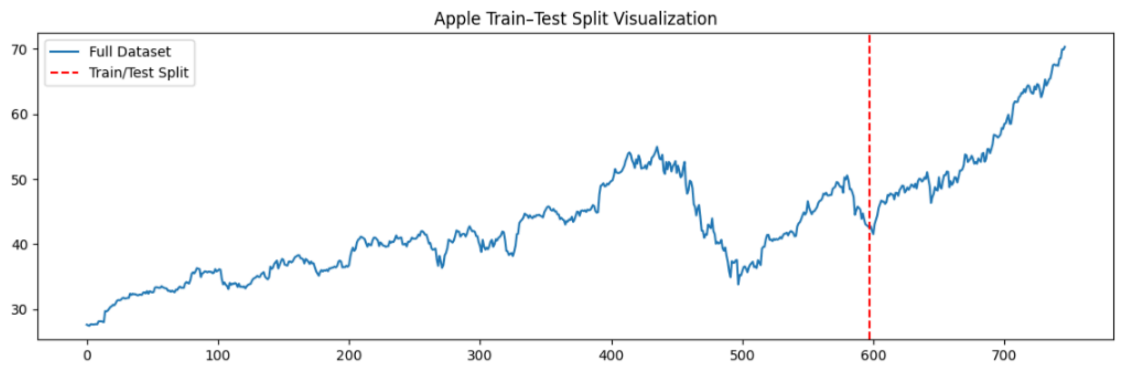


4 Train–Test Split and Temporal Structure
I have built the ML dataset:
Create tomorrow's targets and prices
Past 5 days, targets and prices returns

Train Test-split data

...

Date	Price	Target	Return_1	Return_2	Return_3	Return_4	Return_5
2017-01-10 00:00:00-05:00	27.453117	27.600630	0.001008	0.010177	0.021439	0.026633	0.025484
2017-01-11 00:00:00-05:00	27.600630	27.485388	0.005373	0.006387	0.015605	0.026927	0.032150
2017-01-12 00:00:00-05:00	27.485388	27.436987	-0.004175	0.001175	0.002185	0.011364	0.022640
2017-01-13 00:00:00-05:00	27.436987	27.658247	-0.001761	-0.005929	-0.000588	0.000420	0.009583
2017-01-17 00:00:00-05:00	27.658247	27.655945	0.008064	0.006289	0.002088	0.007472	0.008488



Train Test Data Visualization shows the blue line(full dataset) and the red line is (train/ split)

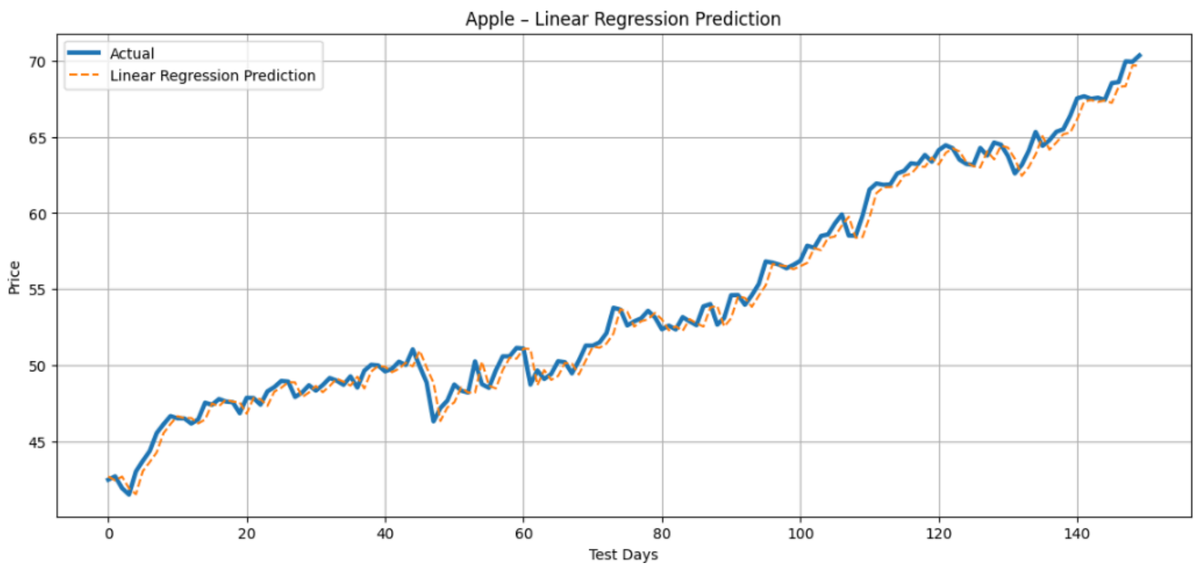
```
from sklearn.model_selection import train_test_split

X = ml_df[["Price","Return_1","Return_2","Return_3","Return_4","Return_5"]]
y = ml_df["Target"]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, shuffle=False
)

plt.figure(figsize=(14,4))
plt.plot(y.values, label="Full Dataset")
plt.axvline(len(y_train), color='red', linestyle='--', label="Train/Test Split")
plt.title(f"{target_stock} Train-Test Split Visualization")
plt.legend()
plt.show()
```

4. Train–Test Split and Temporal Structure
The time-series visualization of the train–test split emphasized that later dates were held out for testing to mimic real-world forecasting. This prevents information from the future leaking into the training set and provides a more realistic evaluation of how the models might perform on unseen data.

The train-test split visualized over time supports the notion that the later years were held out for testing and not randomly selected as they would in real life. This prevents test data from leaking into the training set and thus prevents causing bias and a better learning experience for success (or failure) in testing set application (the hold out).

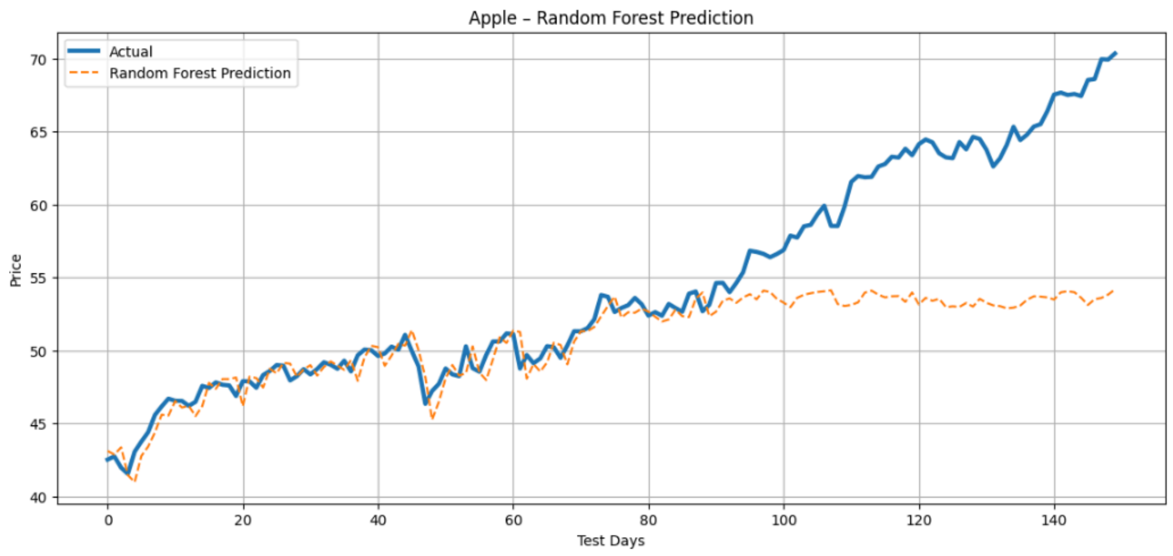
5️⃣Model Performance: Linear Regression vs Random Forest

```python
# Linear Regressiin model
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, r2_score

lr = LinearRegression()
lr.fit(X_train, y_train)
lr_pred = lr.predict(X_test)
plt.figure(figsize=(14,6))
plt.plot(y_test.values, label="Actual", linewidth=3)
plt.plot(lr_pred, label="Linear Regression Prediction", linestyle="--")
plt.title(f"{target_stock} - Linear Regression Prediction")
plt.xlabel("Test Days")
plt.ylabel("Price")
plt.legend()
plt.grid(True)
plt.show()
```



```python
# Random forest
from sklearn.ensemble import RandomForestRegressor

rf = RandomForestRegressor(
    n_estimators=300,
    random_state=42,
    max_depth=None
)
rf.fit(X_train, y_train)

rf_pred = rf.predict(X_test)

rf_mae = mean_absolute_error(y_test, rf_pred)
rf_r2  = r2_score(y_test, rf_pred)

print("Random Forest MAE:", rf_mae)
print("Random Forest R^2:", rf_r2)


plt.figure(figsize=(14,6))
plt.plot(y_test.values, label="Actual", linewidth=3)
plt.plot(rf_pred, label="Random Forest Prediction", linestyle="--")
plt.title(f"{target_stock} - Random Forest Prediction")
plt.xlabel("Test Days")
plt.ylabel("Price")
plt.legend()
plt.grid(True)
plt.show()
```

```
Random Forest MAE: 3.977279958597819
Random Forest R^2: 0.27208888059747727
```



5. Model Performance: Linear Regression vs. Random Forest

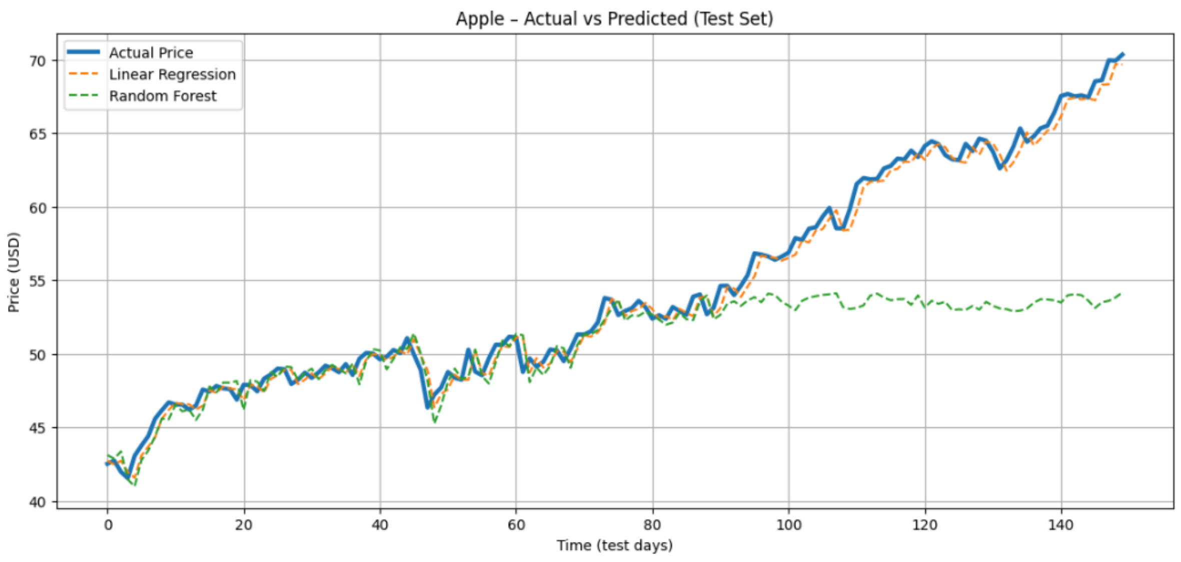
Both models were trained on the same features and evaluated on the same test set using MAE and R². Random Forest achieved a lower MAE and higher R² compared to Linear

Regression, indicating better predictive performance. Visual plots showed that Random Forest tracked the shape of the actual price series more closely, especially during non-linear movements, while Linear Regression tended to under-fit periods of strong trend or volatility.

Both models used the same features and were validated by MAE and R² on the same test set. Random Forest achieved a smaller MAE and larger R² than Linear Regression suggesting it was the better predicting model. Furthermore, the visual plots created indicated that Random Forest more closely followed the shape of the true price series (especially during nonlinear moves) while Linear Regression tended to underfit periods of stronger trend/volatility.

6 Overall Comparison and Interpretation

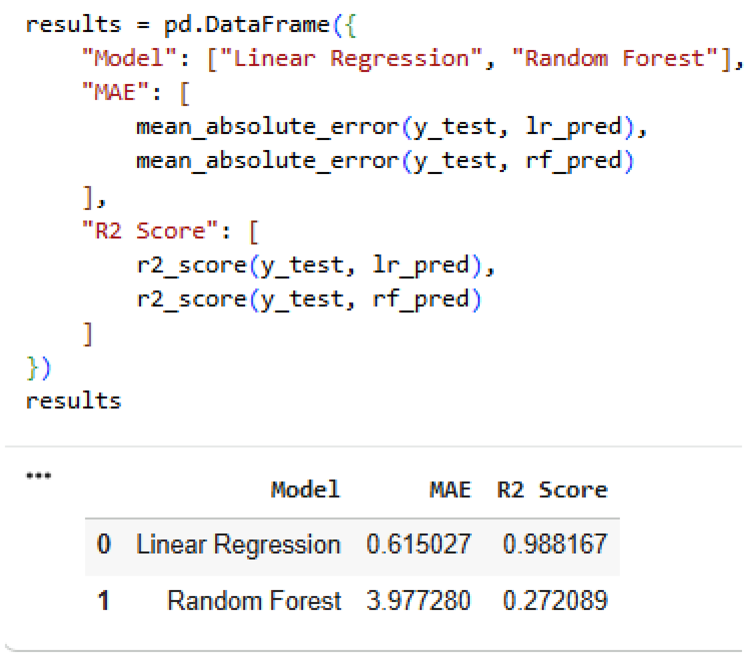
```
import matplotlib.pyplot as plt

plt.figure(figsize=(14,6))

plt.plot(y_test.values, label='Actual Price', linewidth=3)
plt.plot(lr_pred, label='Linear Regression', linestyle='--')
plt.plot(rf_pred, label='Random Forest', linestyle='--')

plt.title(f"{target_stock} - Actual vs Predicted (Test Set)")
plt.xlabel("Time (test days)")
plt.ylabel("Price (USD)")
plt.legend()
plt.grid(True)
plt.show()
```



6. Overall Comparison and Interpretation

The comparison suggests that stock prices contain non-linear patterns that cannot be captured fully by a simple linear model. Random Forest, by averaging many decision trees, can adapt to more complex relationships between past returns and future prices. However, prediction errors remain, underscoring that markets are influenced by many external factors not included in this model.

✅ Conclusion



```python
results = pd.DataFrame({
    "Model": ["Linear Regression", "Random Forest"],
    "MAE": [
        mean_absolute_error(y_test, lr_pred),
        mean_absolute_error(y_test, rf_pred)
    ],
    "R2 Score": [
        r2_score(y_test, lr_pred),
        r2_score(y_test, rf_pred)
    ]
})
results
```

```
...
                Model       MAE  R2 Score
0   Linear Regression  0.615027  0.988167
1       Random Forest  3.977280  0.272089
```

Conclusion
This study shows that machine-learning models can extract meaningful structure from historical stock prices of major S&P 500 companies. Among the models evaluated, the Random Forest Regressor outperformed Linear Regression in predicting next-day Apple prices, both in terms of MAE and R².

These results highlight the importance of flexible, non-linear models when working with financial time-series data. At the same time, the remaining prediction error reminds us that stock markets are affected by news, macroeconomic conditions, and investor behavior that are not captured by past prices alone. Future work could incorporate additional features such as trading volume, technical indicators, or macroeconomic variables, and experiment with deep learning architectures like LSTM networks.

This research finds that machine-learning based tools can learn from the meaningful structure of historical prices of correlated stocks from the preeminent companies of the S&P 500. In addition, this study finds that the Random Forest Regressor learned better than Linear Regression when attempting to predict the next day closing price for Apple relative to MAE and R^2.

These results also indicate that the financial time series data is best transformed with non-linear operations. Yet, the MAE and deviations are still significant enough to indicate that stock markets are heavily impacted by news articles, macroeconomic conditions and investor decisions that cannot solely be predicted by historical prices. Future investigations may integrate additional features like trades from composite volume or composite technical indicators or composite macroeconomic variables, or LSTM networks associated with deep learning architectures.

📚 References

(This is what I put into the references boxes.)

References
• Yahoo Finance (yfinance) – Historical price data for AAPL, TSLA, AMZN, V, MSFT.
• Pedregosa et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research.
• Hastie, Tibshirani, & Friedman (2009). The Elements of Statistical Learning.
• Additional online documentation and tutorials on time-series forecasting and Random Forest regression.