Stock Price Prediction

Goal: Use machine learning to predict future stock prices (Apple Inc. - AAPL) using historical trends and technical indicators.

Tools: Python, yfinance, pandas, matplotlib, seaborn, scikit-learn

Complete Code:

```
# stock_price_prediction.py

import yfinance as yf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
import seaborn as sns

# Load AAPL Stock Data
df = yf.download('AAPL', start='2018-01-01', end='2024-01-01')
df.reset_index(inplace=True)

# Feature Engineering
df['Close_lag1'] = df['Close'].shift(1)
df['Close_lag2'] = df['Close'].shift(2)
df['MA10'] = df['Close'].rolling(window=10).mean()
df['MA20'] = df['Close'].rolling(window=20).mean()
df['Return'] = df['Close'].pct_change()

# Drop rows with missing values
df.dropna(inplace=True)

# Define Features and Target
X = df[['Close_lag1', 'Close_lag2', 'MA10', 'MA20', 'Return']]
y = df['Close']

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)

print("X shape:", X.shape)
print("y shape:", y.shape)
```

```python
y = df['Close']  # Overwrite y properly
print(type(y))
print(y.shape)

# Train Linear Regression Model
model = LinearRegression()
model.fit(X_train, y_train)

# Predict and Evaluate
y_pred = model.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)

print(f"Root Mean Squared Error: {rmse:.2f}")
print(f"R² Score: {r2:.2f}")

# Actual vs Predicted Visualization
plt.figure(figsize=(10, 5))
plt.plot(y_test.values, label='Actual Price', alpha=0.7)
plt.plot(y_pred, label='Predicted Price', alpha=0.7)
plt.title("Actual vs Predicted AAPL Closing Price")
plt.xlabel("Test Data Index")
plt.ylabel("Price ($)")
plt.legend()
plt.tight_layout()
plt.show()

errors = y_test - y_pred
plt.figure(figsize=(8, 5))
sns.histplot(errors, bins=50, kde=True, color='crimson')
plt.title("Prediction Error Distribution")
plt.xlabel("Prediction Error ($)")
plt.ylabel("Frequency")
plt.tight_layout()
plt.show()
```

```python
plt.figure(figsize=(8, 6))
sns.heatmap(df[['Close', 'Close_lag1', 'Close_lag2', 'MA10', 'MA20', 'Return']].corr(),
            annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Matrix of Features")
plt.tight_layout()
plt.show()

# Predict Next Day Price (Real-Time Inference)
latest_features = X.tail(1)
predicted_next = float(model.predict(latest_features)[0])
print(f"Predicted next close price: ${predicted_next:.2f}")
```

Snippetwise Details:

1.Import libraries and loading data:

```
import yfinance as yf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

```
df = yf.download('AAPL', start='2018-01-01', end='2024-01-01')
df.reset_index(inplace=True)
```

```
C:\Users\91866\AppData\Local\Temp\ipykernel_11180\3601102785.py:1: FutureWarning: YF.downlo
ad() has changed argument auto_adjust default to True
  df = yf.download('AAPL', start='2018-01-01', end='2024-01-01')
[*********************100%***********************]  1 of 1 completed
```

- Libraries used for data loading, manipulation, modelling, and evaluation.
- Uses yfinance to retrieve Apple stock data over 6 years.
- Data includes open, close, high, low, volume, and date.

2. Feature engineering , Define features and target

- **Definition:** The process of transforming raw data into meaningful features that improve model performance.
- **Key Points:**
  - Includes selecting, modifying, or creating new features from the original data.
  - Techniques: encoding categorical variables, scaling, normalization, binning, polynomial features, etc.
  - Helps the model learn better patterns and reduces overfitting.
  - Essential for enhancing the predictive power of machine learning algorithms.

- X contains engineered indicators.
- y is the target variable: next-day closing price.

```
df['Close_lag1'] = df['Close'].shift(1)
df['Close_lag2'] = df['Close'].shift(2)
df['MA10'] = df['Close'].rolling(window=10).mean()
df['MA20'] = df['Close'].rolling(window=20).mean()
df['Return'] = df['Close'].pct_change()
df.dropna(inplace=True)
```

```
X = df[['Close_lag1', 'Close_lag2', 'MA10', 'MA20', 'Return']]
y = df['Close']
```

3.Train/test split , Train model ,Make predictions and evaluate:

- Split into training (80%) and testing (20%) while preserving distribution.
- Linear model fits data to minimize prediction error using least squares.
- RMSE measures average error (lower is better).
- R^2 measures how much variance is explained by the model (closer to 1 is better).

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

```
model = LinearRegression()
model.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print("RMSE:", rmse)
print("R² Score:", r2)
```

```
RMSE: 0.9248716791845651
R² Score: 0.9996889879386754
```

4.Predict future price:

- Simulates using the model to predict the next day's price.
- Useful for practical applications like trading signals.

```
latest_features = X.tail(1)
predicted_next_close = float(model.predict(latest_features)[0])
print(f"Predicted next close price: ${predicted_next_close:.2f}")
```

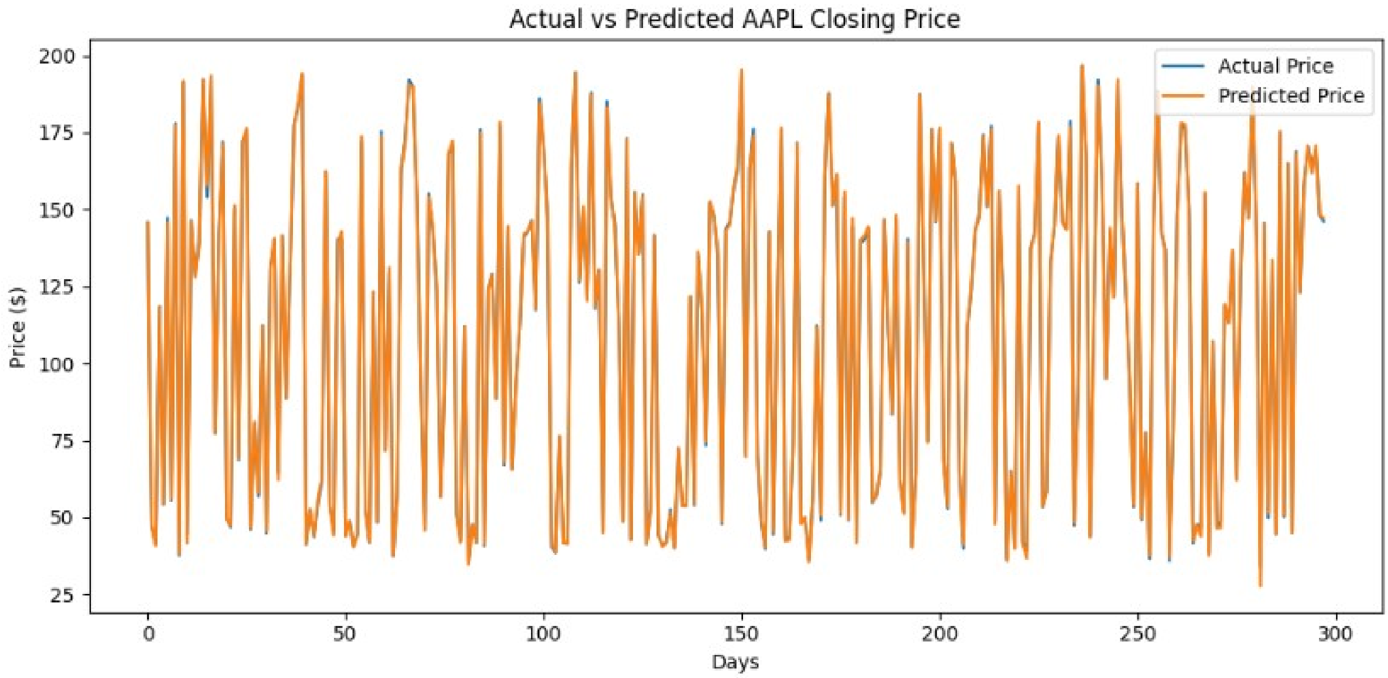
```
Predicted next close price: $191.76
```

5.Visualize:

A. Actual VS Predicted AAPL Closing Price :

- Shows model's ability to track price movement.
- Visual confirmation of prediction quality.

```
plt.figure(figsize=(10, 5))
plt.plot(y_test.values, label='Actual Price')
plt.plot(y_pred, label='Predicted Price')
plt.title("Actual vs Predicted AAPL Closing Price")
plt.xlabel("Days")
plt.ylabel("Price ($)")
plt.legend()
plt.tight_layout()
plt.show()
```



B .Prediction Error Distribution :

- This plot shows how far off the model's predictions are from actual stock prices.
- The error is calculated as:
  ➤ error = actual − predicted

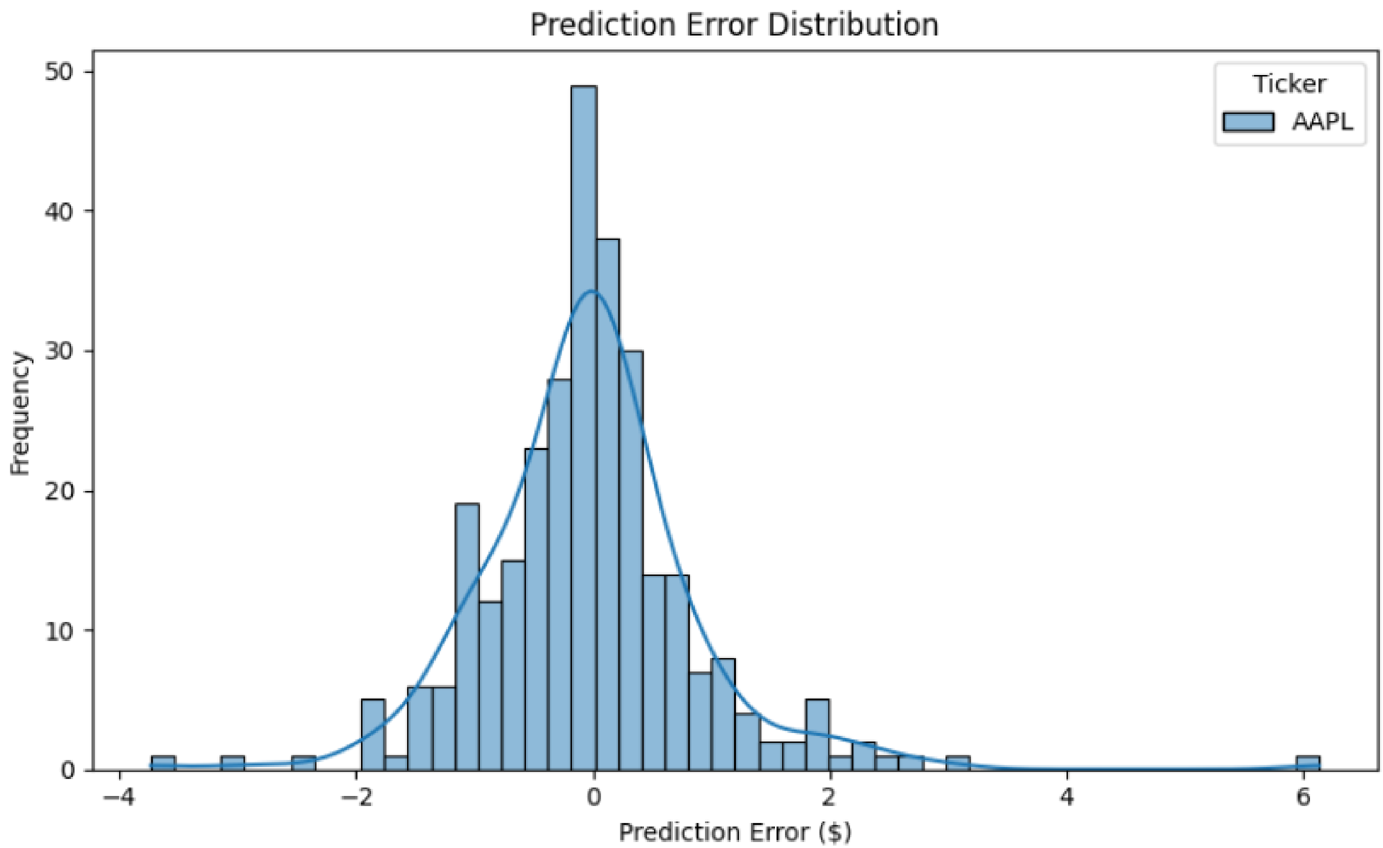
- A normal-like bell curve centered around 0 means the model makes unbiased predictions.
- Wider spread indicates larger average error or inconsistency.

Why it's useful:

- Helps detect bias in predictions.
- Indicates whether your model consistently overestimates or underestimates prices.
- Great for checking real-world usability of the model.

```
errors = y_test - y_pred

plt.figure(figsize=(8, 5))
sns.histplot(errors, bins=50, kde=True, color='crimson')
plt.title("Prediction Error Distribution")
plt.xlabel("Prediction Error ($)")
plt.ylabel("Frequency")
plt.tight_layout()
plt.show()
```



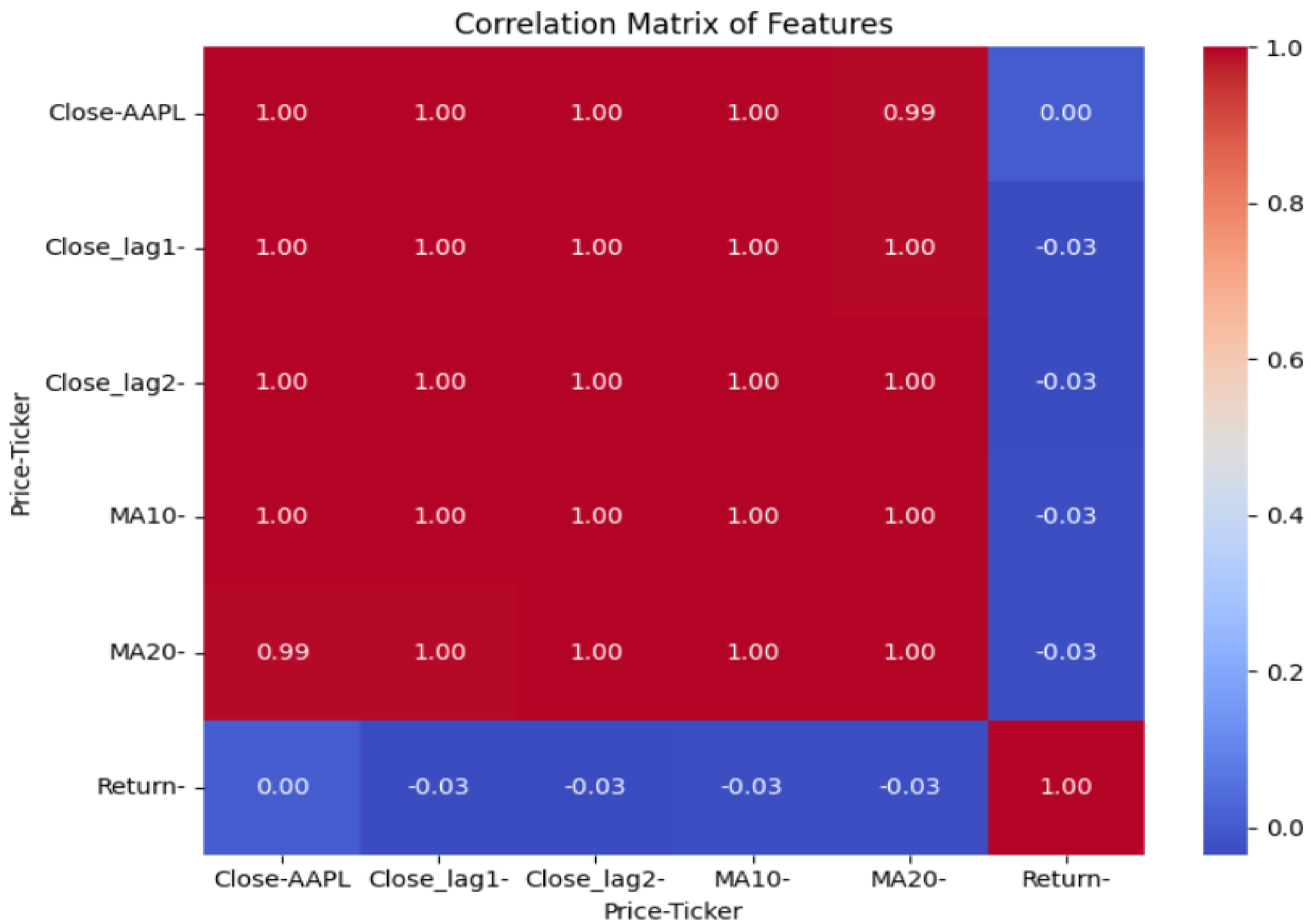
C. Correlation Matrix:

- Shows the strength of linear relationships between features.
- Values range from:
  - ➤ +1 = strong positive correlation
  - ➤ -1 = strong negative correlation
  - ➤ 0 = no correlation
- Diagonal is always 1 (perfect correlation with itself).

Why it's useful:

- Helps assess multicollinearity (overlapping info between features).
- Aids feature selection or dimensionality reduction decisions.
- Shows data understanding and not just model-building.

```
plt.figure(figsize=(8, 6))
sns.heatmap(df[['Close', 'Close_lag1', 'Close_lag2', 'MA10', 'MA20', 'Return']].corr(),
            annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Matrix of Features")
plt.tight_layout()
plt.show()
```



Importance of project:

Real-World Relevance :

- This project simulates a simplified version of what quantitative analysts and financial firms like Fidelity do to assess future price movements.
- Predicting future prices can support:
    - Algorithmic trading
    - Portfolio management strategies
    - Risk management decisions

Technical Relevance :

- Demonstrates my ability to work with time-series financial data.
- Shows how to engineer meaningful features from raw market data (e.g., lags, moving averages, returns).
- Shows practical use of regression for forecasting — a key ML application in finance.

Business Relevance :

- Models like this help determine entry/exit points in trading.
- Understanding price trends based on past behaviour is a core component of market analysis.
- Shows initiative toward data-driven decision-making — a skill highly valued by fintech and investment firms.

Analytical Depth :

- Visualized error distribution
- Interpreted feature importance
- Explored feature correlation
- Evaluated performance with RMSE and R^2