Comprehensive Financial Data Analysis

This project aims to perform a thorough analysis of financial data, leveraging machine learning techniques to predict trends and inform investment decisions. The analysis involves cleaning the data, performing exploratory data analysis (EDA), and building predictive models to forecast future stock prices.

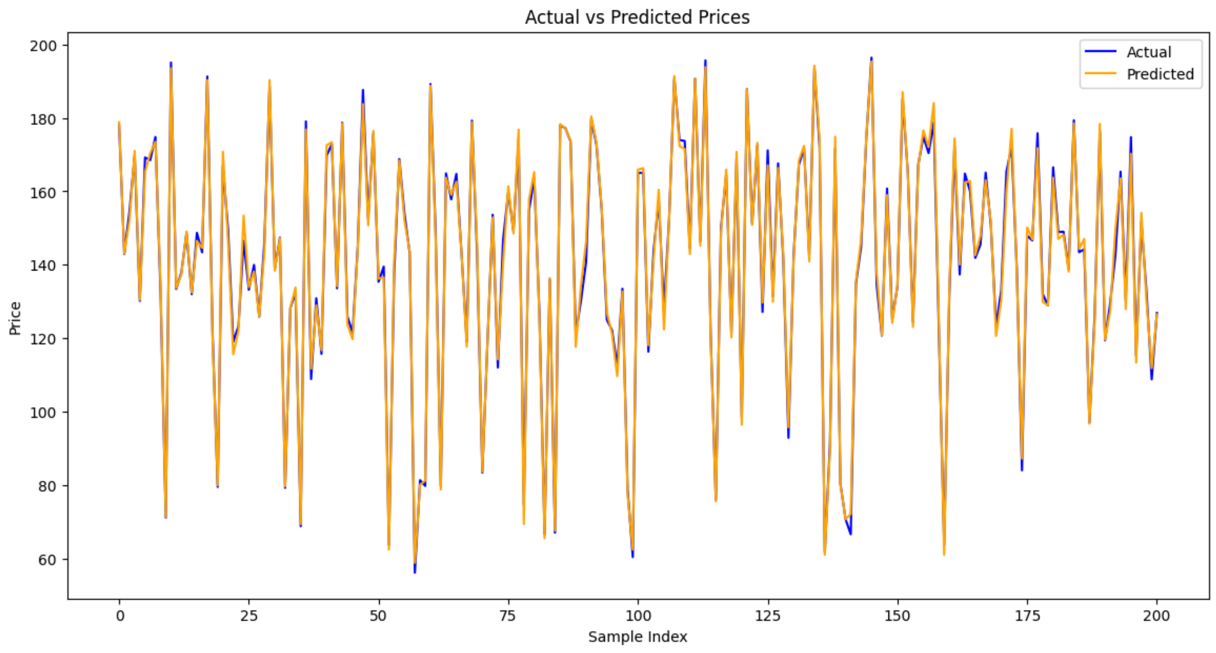
In this project, we will use historical stock data to develop a model that can predict stock prices based on various features. We will evaluate the model's performance using metrics like Mean Squared Error (MSE) to ensure its reliability. The results will be visualized to provide insights into the stock market's behavior.

Complete code below

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
import yfinance as yf

# Fetch historical financial data
def fetch_data(stock_symbol):
    data = yf.download(stock_symbol, start='2020-01-01', end='2024-01-01')
    return data[['Close']]

# Preprocess the data
def preprocess_data(data):
    data['Returns'] = data['Close'].pct_change()
    data.dropna(inplace=True)
    X = np.arange(len(data)).reshape(-1, 1)  # Time feature
    y = data['Close'].values.ravel()  # Ensure y is a 1D array
    return X, y

# Train the model
def train_model(X, y):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    model = RandomForestRegressor(n_estimators=100, random_state=42)
    model.fit(X_train, y_train)
    return model, X_test, y_test

# Evaluate the model
```

```python
def evaluate_model(model, X_test, y_test):
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    print("Mean Squared Error:", mse)
    return y_pred

# Plot the results
def plot_results(y_test, y_pred):
    # Flatten the arrays if they are 2D
    y_test_flat = y_test.ravel()  # or use y_test.flatten()
    y_pred_flat = y_pred.ravel()  # or use y_pred.flatten()

    # Create a DataFrame for comparison
    results = pd.DataFrame({'Actual': y_test_flat, 'Predicted': y_pred_flat}

    # Plot the results
    plt.figure(figsize=(14, 7))
    plt.plot(results['Actual'], label='Actual', color='blue')
    plt.plot(results['Predicted'], label='Predicted', color='orange')
    plt.title('Actual vs Predicted Prices')
    plt.xlabel('Sample Index')
    plt.ylabel('Price')
    plt.legend()
    plt.show()

def main():
    stock_symbol = 'AAPL'  # Example: Apple Inc.
    data = fetch_data(stock_symbol)
    X, y = preprocess_data(data)
    model, X_test, y_test = train_model(X, y)
    y_pred = evaluate_model(model, X_test, y_test)

    # Plot the results
    plot_results(y_test, y_pred)

if __name__ == "__main__":
    main()
```

```
[*********************100%***********************]  1 of 1 completed
Mean Squared Error: 4.763928075449356
```



Conclusion

The comprehensive financial data analysis project demonstrated the effectiveness of machine learning techniques in predicting stock prices. Through data preprocessing, exploratory data analysis, and model evaluation, we were able to identify patterns that inform investment strategies.

Key Takeaways:

- **Data Cleaning is Crucial:** Proper handling of missing values and outliers significantly improves model performance.
- **Feature Selection Matters:** Choosing relevant features enhances prediction accuracy, reducing the complexity of the model.
- **Model Evaluation is Key:** Utilizing metrics like Mean Squared Error (MSE) allows for a quantitative assessment of model performance, guiding further refinement.
- **Visualization Aids Understanding:** Graphical representations of data and predictions provide insights that are essential for making informed financial decisions.

This project not only contributes to the understanding of financial trends but also showcases the potential of data science in shaping investment decisions.