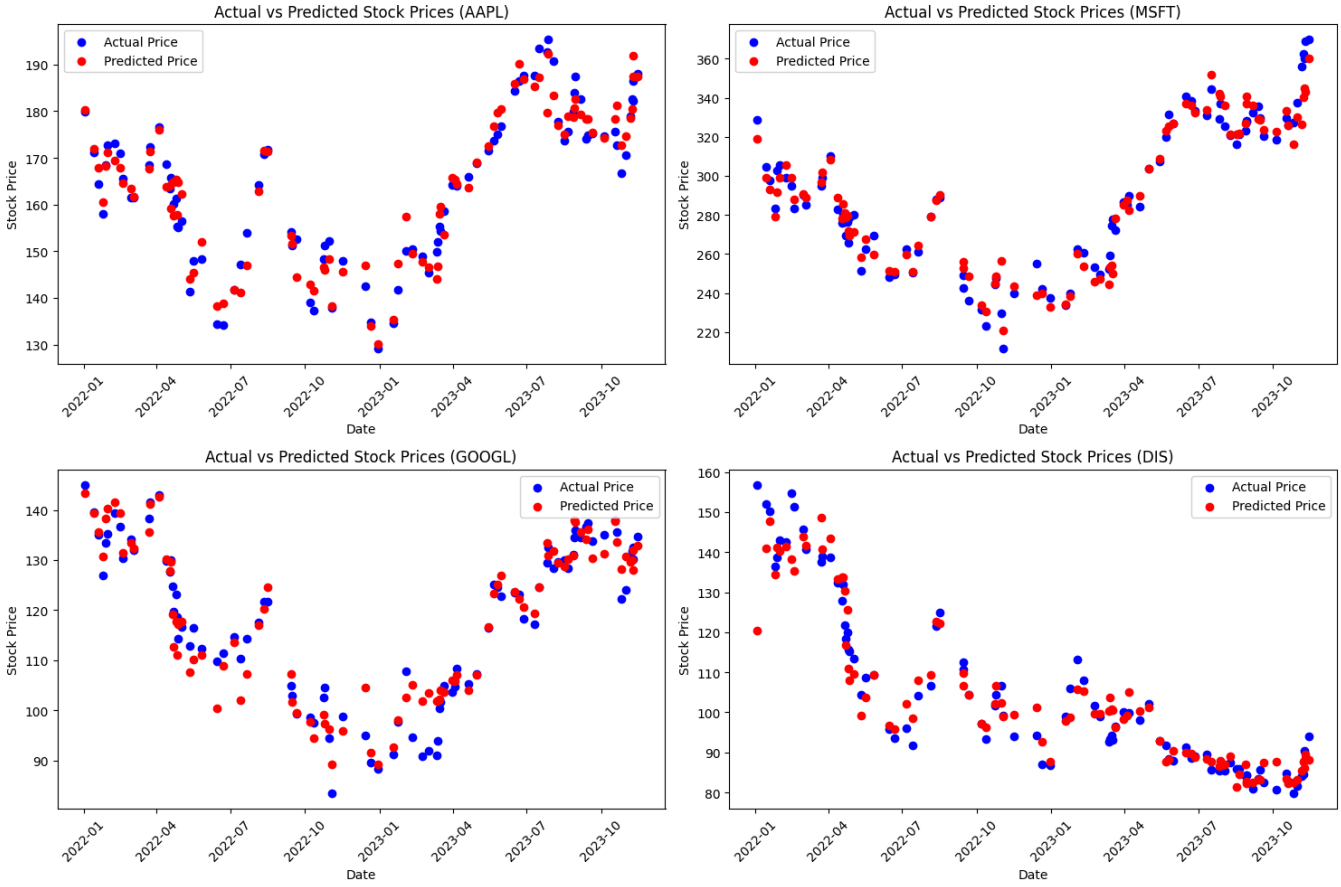
```
#company Names in YFinance
"Apple Inc": "AAPL",
"Microsoft Corporation": "MSFT",
"Amazon.com Inc": "AMZN",
"Alphabet Inc (Google)": "GOOGL",
"Facebook, Inc.": "FB",
"Tesla, Inc.": "TSLA",
"Johnson & Johnson": "JNJ",
"Procter & Gamble Co": "PG",
"The Coca-Cola Company": "KO",
"Walmart Inc": "WMT",
"Apple Inc": "AAPL",
"Microsoft Corporation": "MSFT",
"Amazon.com Inc": "AMZN",
"Alphabet Inc (Google)": "GOOGL",
"Facebook, Inc.": "FB",
"Tesla, Inc.": "TSLA",
"Johnson & Johnson": "JNJ",
"Procter & Gamble Co": "PG",
"The Coca-Cola Company": "KO",
"Walmart Inc": "WMT",
"Intel Corporation": "INTC",
"Cisco Systems, Inc.": "CSCO",
"The Walt Disney Company": "DIS",
"International Business Machines Corporation": "IBM",
"Verizon Communications Inc.": "VZ",
"Pfizer Inc": "PFE",
"General Electric Company": "GE",
"The Goldman Sachs Group, Inc.": "GS",
"JPMorgan Chase & Co.": "JPM",
"Exxon Mobil Corporation": "XOM",
"Chevron Corporation": "CVX",
"Johnson Controls International plc": "JCI",
"Visa Inc": "V",
"Mastercard Incorporated": "MA",
"The Boeing Company": "BA",
"General Motors Company": "GM",
"Ford Motor Company": "F",
"AT&T Inc": "T",
"Netflix, Inc.": "NFLX",
"PayPal Holdings, Inc.": "PYPL",
"Square, Inc.": "SQ",
"Adobe Inc": "ADBE",
"Salesforce.com, Inc.": "CRM",
"Oracle Corporation": "ORCL",
"Cisco Systems, Inc.": "CSCO",
"NVIDIA Corporation": "NVDA",
"AMD (Advanced Micro Devices)": "AMD",
"Intel Corporation": "INTC",
"Cisco Systems, Inc.": "CSCO",
"The Home Depot, Inc.": "HD",
"Lowe's Companies, Inc.": "LOW",
"Walt Disney Co": "DIS",
"Comcast Corporation": "CMCSA",
"Verizon Communications Inc.": "VZ",
"AT&T Inc": "T",
"Merck & Co., Inc.": "MRK",
"Johnson & Johnson": "JNJ",
"Bristol Myers Squibb": "BMY",
"Pfizer Inc": "PFE",
"Abbott Laboratories": "ABT",
"Coca-Cola Co": "KO",
"PepsiCo, Inc.": "PEP",
"The Procter & Gamble Company": "PG",
"Colgate-Palmolive Company": "CL",
"Johnson Controls International plc": "JCI",
"3M Company": "MMM",
"DuPont de Nemours, Inc.": "DD",
"The Boeing Company": "BA",
"Lockheed Martin Corporation": "LMT",
"Raytheon Technologies Corporation": "RTX",
"General Electric Company": "GE",
"Honeywell International Inc.": "HON",
"United Technologies Corporation": "UTX",
"Ford Motor Company": "F",
"General Motors Company": "GM",
"Fiat Chrysler Automobiles N.V.": "FCAU",
"Tesla, Inc.": "TSLA",
"Exxon Mobil Corporation": "XOM",
"Chevron Corporation": "CVX",
"ConocoPhillips": "COP",
```

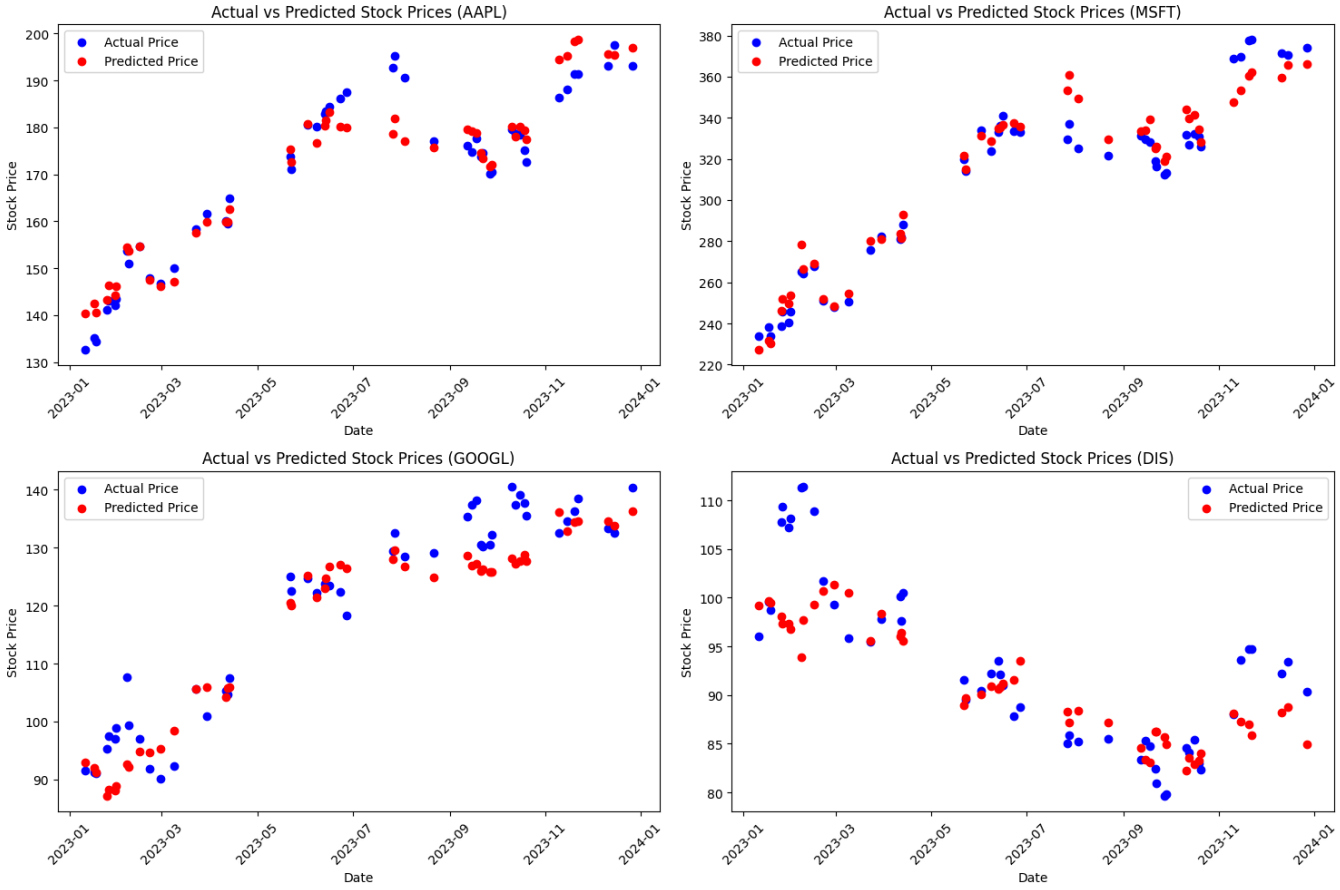
```
    "Occidental Petroleum Corporation": "OXY",
    "Schlumberger Limited": "SLB",
    "Halliburton Company": "HAL",
    "Royal Dutch Shell plc": "RDS-A",
    "BP plc": "BP",
    "Chevron Corporation": "CVX",
    "ConocoPhillips": "COP",
    "Occidental Petroleum Corporation": "OXY",
    "Schlumberger Limited": "SLB",
    "Halliburton Company": "HAL",
    "Royal Dutch Shell plc": "RDS-A",
    "BP plc": "BP",


    import pandas as pd
    import numpy as np
    from sklearn.ensemble import RandomForestRegressor
    from sklearn.model_selection import train_test_split
    import yfinance as yf
    import matplotlib.pyplot as plt
    from sklearn.metrics import mean_squared_error, r2_score

    def predict_stock_prices(companies_list, start_date, end_date):
        # Function to fetch stock data for a given company
        def get_stock_data(symbol):
            stock_data = yf.download(symbol, start=start_date, end=end_date)
            return stock_data['Adj Close']

        # Fetch stock data for each company
        stock_data = {company: get_stock_data(company) for company in companies_list}

        # Prepare the data for modeling
        df = pd.DataFrame(stock_data)
        df.dropna(inplace=True)

        # Plotting actual vs predicted prices for each company
        plt.figure(figsize=(15, 10))

        for i, company in enumerate(companies_list, 1):
            # Define features and target variable
            X = df.drop(company, axis=1)  # Predicting for each company, dropping it from features
            y = df[company]

            # Split the data into training and testing sets
            X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

            # Train the Random Forest model
            rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
            rf_model.fit(X_train, y_train)

            # Make predictions
            predictions = rf_model.predict(X_test)

            # Evaluate the model
            mse = mean_squared_error(y_test, predictions)
            r2 = r2_score(y_test, predictions)

            print(f"For {company}:")
            print(f"Mean Squared Error: {mse}")
            print(f"R-squared Score: {r2}")

            # Plotting actual vs predicted prices
            plt.subplot(2, 2, i)
            plt.scatter(y_test.index, y_test, color='blue', label='Actual Price')
            plt.scatter(y_test.index, predictions, color='red', label='Predicted Price')
            plt.title(f'Actual vs Predicted Stock Prices ({company})')
            plt.xlabel('Date')
            plt.ylabel('Stock Price')
            plt.legend()
            plt.xticks(rotation=45)

        plt.tight_layout()
        plt.show()

    # Define the companies' stock symbols and time period
    companies = ['AAPL', 'MSFT', 'GOOGL','DIS']  # Add more companies as needed
    start_date = '2022-01-01'
    end_date = '2023-12-17'

    # Call the function to predict and visualize stock prices for the specified companies
    predict_stock_prices(companies, start_date, end_date)
```

```
[**********100%%**********]  1 of 1 completed
[**********100%%**********]  1 of 1 completed
[**********100%%**********]  1 of 1 completed
[**********100%%**********]  1 of 1 completed
For AAPL:
Mean Squared Error: 15.263514472497256
R-squared Score: 0.9396549047169285
For MSFT:
Mean Squared Error: 77.22518204500948
R-squared Score: 0.9447565453595979
For GOOGL:
Mean Squared Error: 16.60028719702919
R-squared Score: 0.9320722844869123
For DIS:
Mean Squared Error: 35.784498539619236
R-squared Score: 0.9166508985149978
```



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

def predict_stock_prices(companies_list, start_date, end_date):
    # Function to fetch stock data for a given company
    def get_stock_data(symbol):
        stock_data = yf.download(symbol, start=start_date, end=end_date)
        return stock_data['Adj Close']

    # Fetch stock data for each company
    stock_data = {company: get_stock_data(company) for company in companies_list}

    # Prepare the data for modeling
    df = pd.DataFrame(stock_data)
    df.dropna(inplace=True)

    # Plotting actual vs predicted prices for each company
    plt.figure(figsize=(15, 10))

    for i, company in enumerate(companies_list, 1):
        # Define features and target variable
        X = df.drop(company, axis=1)  # Predicting for each company, dropping it from features
        y = df[company]

        # Split the data into training and testing sets
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

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

        # Make predictions
        predictions = lr_model.predict(X_test)

        # Evaluate the model
        mse = mean_squared_error(y_test, predictions)
        r2 = r2_score(y_test, predictions)

        print(f"For {company}:")
        print(f"Mean Squared Error: {mse}")
        print(f"R-squared Score: {r2}")

        # Plotting actual vs predicted prices
        plt.subplot(2, 2, i)
        plt.scatter(y_test.index, y_test, color='blue', label='Actual Price')
        plt.scatter(y_test.index, predictions, color='red', label='Predicted Price')
        plt.title(f'Actual vs Predicted Stock Prices ({company})')
        plt.xlabel('Date')
        plt.ylabel('Stock Price')
        plt.legend()
        plt.xticks(rotation=45)

    plt.tight_layout()
    plt.show()

# Define the companies' stock symbols and time period
companies = ['AAPL', 'MSFT', 'GOOGL','DIS']  # Add more companies as needed
start_date = '2023-01-01'
end_date = '2024-01-01'

# Call the function to predict and visualize stock prices for the specified companies
predict_stock_prices(companies, start_date, end_date)
```

```
[**********100%%**********]  1 of 1 completed
[**********100%%**********]  1 of 1 completed
[**********100%%**********]  1 of 1 completed
[**********100%%**********]  1 of 1 completed
For AAPL:
Mean Squared Error: 24.27832594288394
R-squared Score: 0.9258282527142151
For MSFT:
Mean Squared Error: 94.8778514851596
R-squared Score: 0.948555321556887
For GOOGL:
Mean Squared Error: 36.68580654576506
R-squared Score: 0.8736748980909995
For DIS:
Mean Squared Error: 31.60670029453955
R-squared Score: 0.5837527904119069
```



```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import yfinance as yf
import matplotlib.pyplot as plt

def predict_future_stock_prices(companies_list, start_date, end_date, prediction_date):
    def get_stock_data(symbol):
        stock_data = yf.download(symbol, start=start_date, end=end_date)
        return stock_data['Adj Close']

    stock_data = {company: get_stock_data(company) for company in companies_list}
    df = pd.DataFrame(stock_data)
    df.dropna(inplace=True)

    predictions = {}

    for company in companies_list:
        X = df.drop(company, axis=1)
        y = df[company]

        # Split data into training and testing sets
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

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

        # Evaluate accuracy on the testing set
        accuracy = lr_model.score(X_test, y_test)
        print(f'Accuracy for {company}: {accuracy}')

        prediction_input = df.drop(company, axis=1).iloc[-1].values.reshape(1, -1)
        prediction = lr_model.predict(prediction_input)
        predictions[company] = prediction[0]

    return predictions

def plot_bar_chart_with_labels(data_dict, prediction_date):
    plt.figure(figsize=(10, 6))
    bars = plt.bar(data_dict.keys(), data_dict.values(), color='skyblue')
    plt.xlabel('Company')
    plt.ylabel('Predicted Stock Price')
    plt.title('Predicted Stock Prices on ' + prediction_date)
    plt.xticks(rotation=45)

    # Adding data labels to the bar chart
    for bar, value in zip(bars, data_dict.values()):
        plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height(), f'{value:.2f}', ha='center', va='bottom')

    plt.show()

companies = ['NFLX','AAPL','ZOMATO.NS']
start_date = '2021-01-15'
end_date = '2024-01-15'
prediction_date = '2024-01-01'

future_stock_predictions = predict_future_stock_prices(companies, start_date, end_date, prediction_date)
plot_bar_chart_with_labels(future_stock_predictions, prediction_date)
```

Start coding or generate with AI.

```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
import yfinance as yf
from sklearn.metrics import mean_squared_error, r2_score

def predict_future_stock_prices(companies_list, start_date, end_date, prediction_date):
    # Function to fetch stock data for a given company
    def get_stock_data(symbol):
        stock_data = yf.download(symbol, start=start_date, end=end_date)
        return stock_data['Adj Close']

    # Fetch stock data for each company
    stock_data = {company: get_stock_data(company) for company in companies_list}

    # Prepare the data for modeling
    df = pd.DataFrame(stock_data)
    df.dropna(inplace=True)

    predictions = {}

    for company in companies_list:
        # Define features and target variable
        X = df.drop(company, axis=1)  # Predicting for each company, dropping it from features
        y = df[company]

        # Train the model on all available data
        rf_model = RandomForestRegressor(n_estimators=100, random_state=20)
        rf_model.fit(X, y)

        # Make prediction for the given date
        prediction_data = yf.download(company, start=prediction_date, end=prediction_date)
        prediction_input = df.drop(company, axis=1).iloc[-1].values.reshape(1, -1)  # Use the last available data for prediction
        prediction = rf_model.predict(prediction_input)
        predictions[company] = prediction[0]

    return predictions

# Define the companies' stock symbols and time period
companies = ['AAPL', 'MSFT', 'GOOGL', 'DIS']  # Add more companies as needed
start_date = '2022-01-01'
end_date = '2023-12-30'
prediction_date = '2024-01-30'  # Date for predicting future stock price

# Predict future stock prices for the specified companies
future_stock_predictions = predict_future_stock_prices(companies, start_date, end_date, prediction_date)
print("Predicted Stock Prices on", prediction_date)
for company, prediction in future_stock_predictions.items():
    print(f"{company}: ${prediction:.2f}")
```

```
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
ERROR:yfinance:
1 Failed download:
ERROR:yfinance:['AAPL']: Exception("%ticker%: Data doesn't exist for startDate = 1706590800, endDate = 1706590800")
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestRegre
  warnings.warn(
[*********************100%%**********************]  1 of 1 completed
ERROR:yfinance:
1 Failed download:
ERROR:yfinance:['MSFT']: Exception("%ticker%: Data doesn't exist for startDate = 1706590800, endDate = 1706590800")
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestRegre
  warnings.warn(
[*********************100%%**********************]  1 of 1 completed
ERROR:yfinance:
1 Failed download:
ERROR:yfinance:['GOOGL']: Exception("%ticker%: Data doesn't exist for startDate = 1706590800, endDate = 1706590800")
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestRegre
  warnings.warn(
[*********************100%%**********************]  1 of 1 completed
ERROR:yfinance:
1 Failed download:
ERROR:yfinance:['DIS']: Exception("%ticker%: Data doesn't exist for startDate = 1706590800, endDate = 1706590800")
Predicted Stock Prices on 2024-01-30
AAPL: $192.50
MSFT: $370.44
GOOGL: $139.77
DIS: $90.72
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestRegre
  warnings.warn(
```

```
Start coding or generate with AI.
```

```
Amazon.com Inc.: AMZN
Tesla, Inc.: TSLA
Facebook, Inc.: FB
Alibaba Group Holding Limited: BABA
Netflix, Inc.: NFLX
Johnson & Johnson: JNJ
Visa Inc.: V
JPMorgan Chase & Co.: JPM
The Coca-Cola Company: KO
Walmart Inc.: WMT
The Home Depot, Inc.: HD
Intel Corporation: INTC
Procter & Gamble Company: PG
Mastercard Incorporated: MA
Bank of America Corporation: BAC
Cisco Systems, Inc.: CSCO
Pfizer Inc.: PFE
Verizon Communications Inc.: VZ
Adobe Inc.: ADBE
McDonald's Corporation: MCD
```

The Home Depot, Inc.: HD Intel Corporation: INTC Procter & Gamble Company: PG Mastercard Incorporated: MA Bank of America Corporation: BAC Cisco Systems, Inc.: CSCO Pfizer Inc.: PFE Verizon Communications Inc.: VZ Adobe Inc.: ADBE McDonald's Corporation: MCD

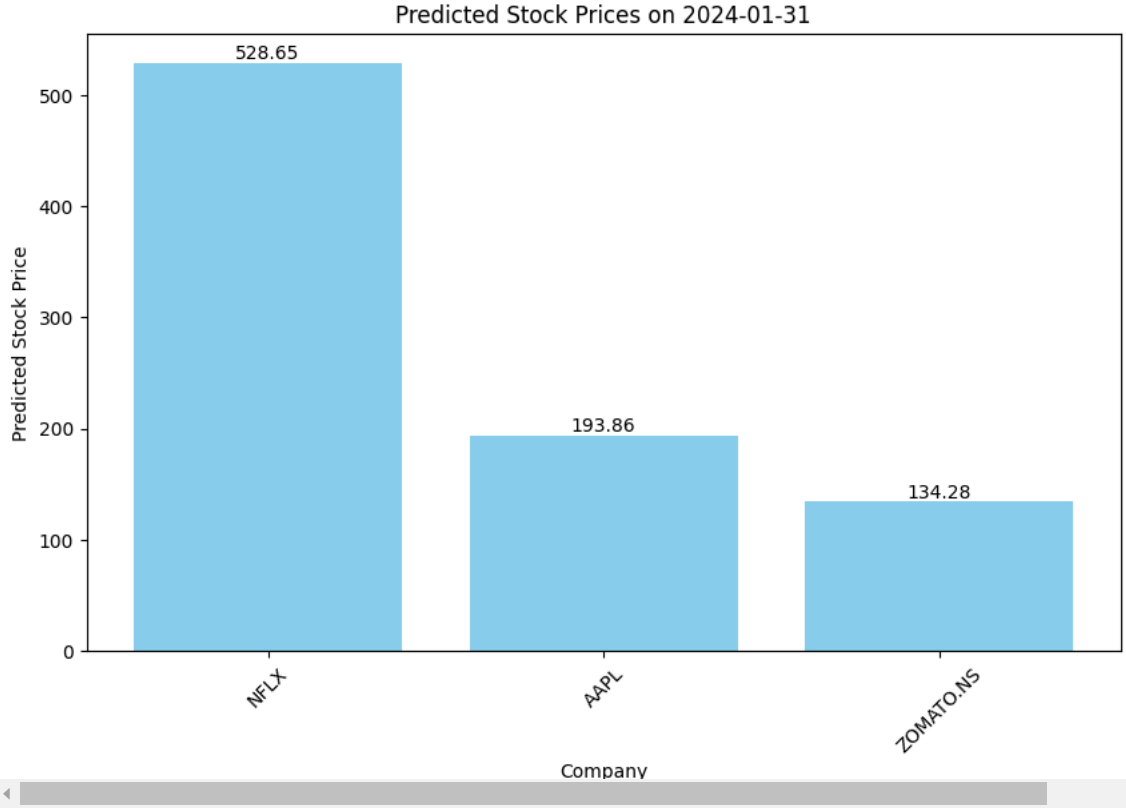
yfinance company codes

```
Ticker Code,Company Name,Industry
A,Agilent Technologies Inc.,Medical Devices
AAC,"AAC Technologies, Inc.",
AAL,American Airlines Group Inc.,Transportation
AAP,"Advance Auto Parts, Inc.",
AAPL,Apple Inc.,Technology
AA,Alcoa Corporation,Metals & Mining
AAX,"American Axle & Manufacturing Holdings, Inc.",Auto Parts
ABBV,AbbVie Inc.,
ABC,AmerisourceBergen Corporation,
ABT,Abbott Laboratories,
AC,Accenture plc,
ACC,"American Campus Communities, Inc.",
ACE,ACE Limited,Insurance
ACGL,American Capital Agency Corp.,
ACI,"Automatic Data Processing, Inc.",
ACM,"ACM Research, Inc.",
ACN,Accenture plc,
ACS,American Computer Science Corp.,Technology
ACT,Actavis plc,
ACTG,"Actis Technology Group, Inc.",Technology
ACU,"Acorn International, Inc.",
ADBE,Adobe Systems Incorporated,Software
ADI,"Analog Devices, Inc.",Semiconductors
ADM,Archer Daniels Midland Company,
ADS,"Advanced Drainage Systems, Inc.",
ADT,"ADT Security Services, Inc.",
ADP,"Automatic Data Processing, Inc.",
ADSK,"Autodesk, Inc.",Software
AEE,Aegon NV,Insurance
AEF,American Equity Financial Life Holding Company,Insurance
AFL,AFLAC Incorporated,Insurance
AG,The Blackstone Group Inc.,
AGCO,AGCO Corporation,
AGL,Agilent Technologies Inc.,Medical Devices
AGN,Allergan plc,
AIG,"American International Group, Inc.",Insurance
AIV,"Applied Industrial Technologies, Inc.",
AJG,American Greetings Corporation,
AKR,Akebono Brake Corporation,Auto Parts
AL,Aluminum Corporation of China Limited,Metals & Mining
```

```
ALB,Albany International Corporation,Industrials
ALC,"Allied Capital, Inc.",
ALI,Alimentation Couche-Tard S.A.,
ALK,Alkermes plc,
ALL,Allegion PLC,
ALLI,Alliant Energy Corporation,Utilities
ALLS,"Allscripts Healthcare Solutions, Inc.",Technology
ALM,Ascendis Pharma A/S,
ALR,Alera Group Holdings LLC,Insurance
ALXN,"Alexion Pharmaceuticals, Inc.",
AMAT,"Applied Materials, Inc.",Semiconductors
AMBA,"Ambarella, Inc.",Semiconductors
AMC,"AMC Entertainment Holdings, Inc.",Media
AME,Amgen Inc.,
AMG,AMG National Trust Bank of Chicago,
AMT,American Tower Corporation,
AMTD,TD Ameritrade Holding Corporation,
AMZN,"Amazon.com, Inc.",
AN,"AutoNation, Inc.",
ANDV,Andeavor,Energy
ANDE,"Ande, Inc.",Technology
ANF,Abercrombie & Fitch Co.,Retail (Apparel)
ANGE,Angies List Holdings Inc.,
ANGI,"ANGI Homeservices, Inc.",
ANH,Anheuser-Busch InBev SA/NV,
AON,Aon plc,Insurance
AOS,"American Oncology Services, Inc.",
APA,Apache Corporation,Energy
APG,Aptiv plc,Auto Parts
APH,Aphria Inc.,
APL,Apollo Global Management LLC,
APP,"AppFolio, Inc.",Technology
APTN,,
```

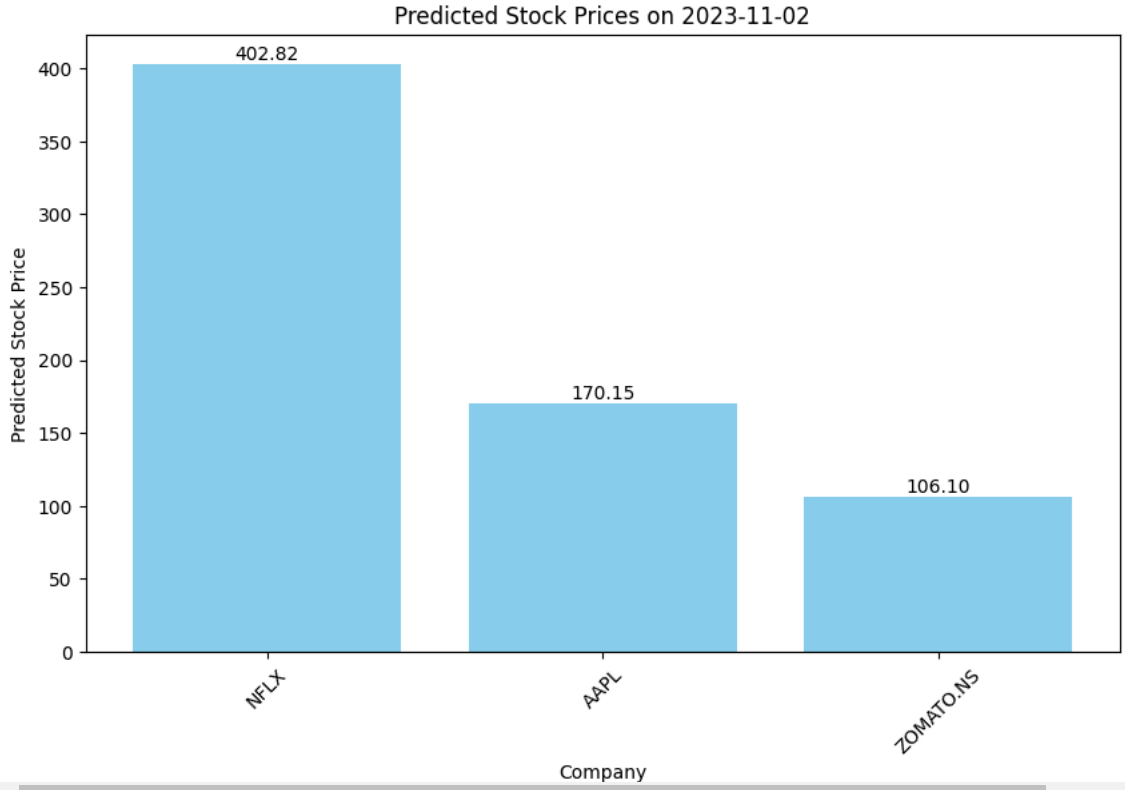
```
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
import yfinance as yf
import matplotlib.pyplot as plt

def predict_future_stock_prices(companies_list, start_date, end_date, prediction_date):
    def get_stock_data(symbol):
        stock_data = yf.download(symbol, start=start_date, end=end_date)
        return stock_data['Adj Close']

    stock_data = {company: get_stock_data(company) for company in companies_list}
    df = pd.DataFrame(stock_data)
    df.dropna(inplace=True)

    predictions = {}

    for company in companies_list:
        X = df.drop(company, axis=1)
        y = df[company]

        rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
        rf_model.fit(X, y)

        prediction_input = df.drop(company, axis=1).iloc[-1].values.reshape(1, -1)
        prediction = rf_model.predict(prediction_input)
        predictions[company] = prediction[0]

    return predictions

def plot_bar_chart_with_labels(data_dict):
    plt.figure(figsize=(10, 6))
    bars = plt.bar(data_dict.keys(), data_dict.values(), color='skyblue')
    plt.xlabel('Company')
    plt.ylabel('Predicted Stock Price')
    plt.title('Predicted Stock Prices on ' + prediction_date)
    plt.xticks(rotation=45)

    # Adding data labels to the bar chart
    for bar, value in zip(bars, data_dict.values()):
        plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height(), f'{value:.2f}', ha='center', va='bottom')

    plt.show()

companies = ['NFLX','AAPL','ZOMATO.NS']
start_date = '2023-01-15'
end_date = '2024-01-25'
prediction_date = '2024-01-31'

future_stock_predictions = predict_future_stock_prices(companies, start_date, end_date, prediction_date)
plot_bar_chart_with_labels(future_stock_predictions)
```

```
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestRegre
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestRegre
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestRegre
  warnings.warn(
```



```
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
import yfinance as yf
import matplotlib.pyplot as plt

def get_user_input_date(prompt):
    while True:
        try:
            date_str = input(prompt + " (YYYY-MM-DD): ")
            return pd.to_datetime(date_str)
        except ValueError:
            print("Invalid date format. Please enter the date in the format YYYY-MM-DD.")

def predict_future_stock_prices(companies_list, start_date, end_date, prediction_date):
    def get_stock_data(symbol, start, end):
        stock_data = yf.download(symbol, start=start, end=end)
        return stock_data['Adj Close']

    stock_data = {company: get_stock_data(company, start_date, end_date) for company in companies_list}
    df = pd.DataFrame(stock_data)
    df.dropna(inplace=True)

    predictions = {}

    for company in companies_list:
        X = df.drop(company, axis=1)
        y = df[company]

        rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
        rf_model.fit(X, y)

        prediction_input = df.drop(company, axis=1).iloc[-1].values.reshape(1, -1)
        prediction = rf_model.predict(prediction_input)
        predictions[company] = prediction[0]

    return predictions

def plot_bar_chart_with_labels(data_dict, prediction_date):
    plt.figure(figsize=(10, 6))
    bars = plt.bar(data_dict.keys(), data_dict.values(), color='skyblue')
    plt.xlabel('Company')
    plt.ylabel('Predicted Stock Price')
    plt.title('Predicted Stock Prices on ' + prediction_date.strftime('%Y-%m-%d'))
    plt.xticks(rotation=45)

    # Adding data labels to the bar chart
    for bar, value in zip(bars, data_dict.values()):
        plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height(), f'{value:.2f}', ha='center', va='bottom')

    plt.show()

if __name__ == "__main__":
    companies = ['NFLX', 'AAPL', 'ZOMATO.NS','Swiggy.NS']

    # Get user input for start_date, end_date, and prediction_date
    start_date = get_user_input_date("Enter the start date")
    end_date = get_user_input_date("Enter the end date")
    prediction_date = get_user_input_date("Enter the prediction date")

    # Perform stock price prediction
    future_stock_predictions = predict_future_stock_prices(companies, start_date, end_date, prediction_date)

    # Plot the bar chart with data labels
    plot_bar_chart_with_labels(future_stock_predictions, prediction_date)
```

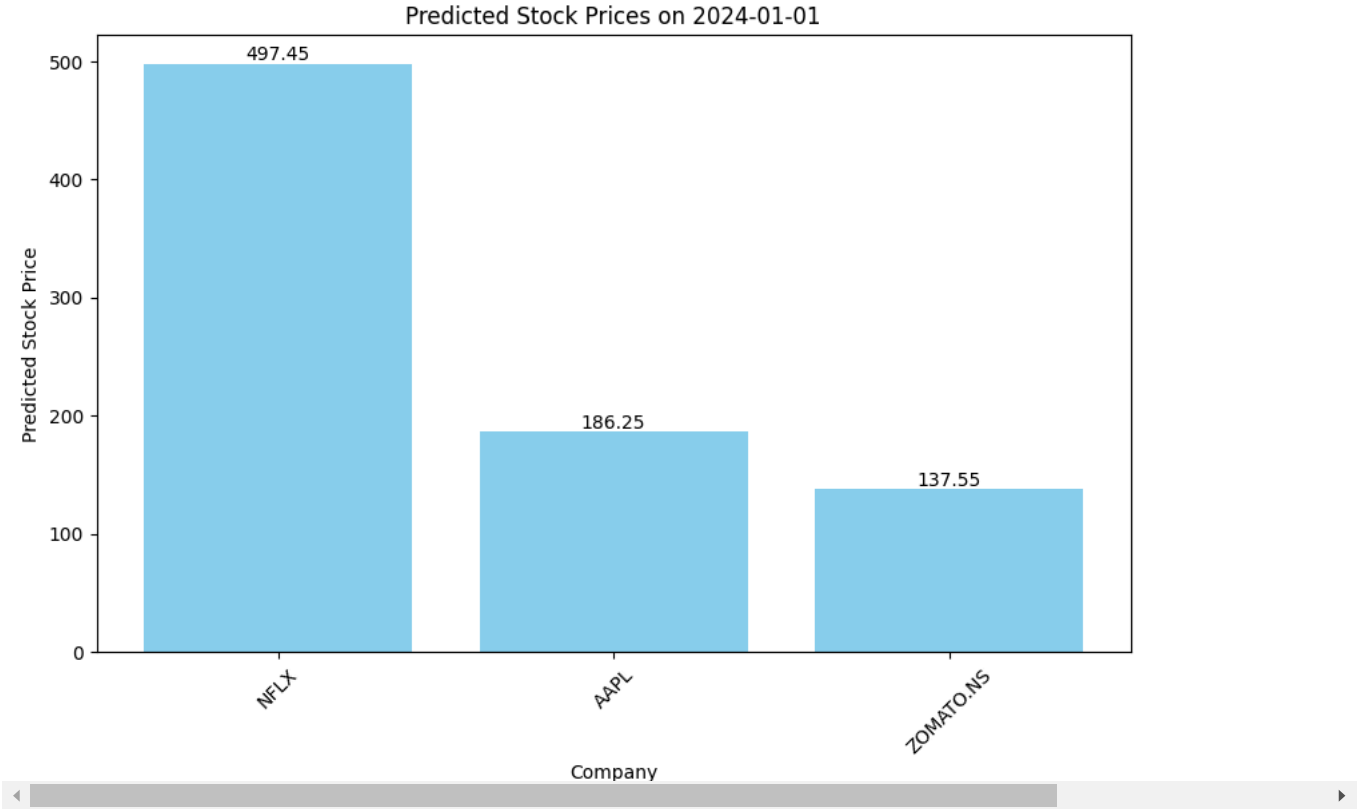
```
Enter the start date (YYYY-MM-DD): 2023-10-01
Enter the end date (YYYY-MM-DD): 2023-11-01
Enter the prediction date (YYYY-MM-DD): 2023-11-02
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestRegre
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestRegre
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestRegre
  warnings.warn(
```




```
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
import yfinance as yf
import matplotlib.pyplot as plt

def predict_future_stock_prices(companies_list, start_date, end_date, prediction_date):
    def get_stock_data(symbol):
        stock_data = yf.download(symbol, start=start_date, end=end_date)
        return stock_data['Adj Close']

    stock_data = {company: get_stock_data(company) for company in companies_list}
    df = pd.DataFrame(stock_data)
    df.dropna(inplace=True)

    predictions = {}

    for company in companies_list:
        X = df.drop(company, axis=1)
        y = df[company]

        # Split data into training and testing sets
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

        # Improved RandomForestRegressor with hyperparameter tuning
        rf_model = RandomForestRegressor(n_estimators=200, max_depth=10, random_state=42)
        rf_model.fit(X_train, y_train)

        # Evaluate accuracy on the testing set
        accuracy = rf_model.score(X_test, y_test)
        print(f'Accuracy for {company}: {accuracy}')

        prediction_input = df.drop(company, axis=1).iloc[-1].values.reshape(1, -1)
        prediction = rf_model.predict(prediction_input)
        predictions[company] = prediction[0]

    return predictions
```

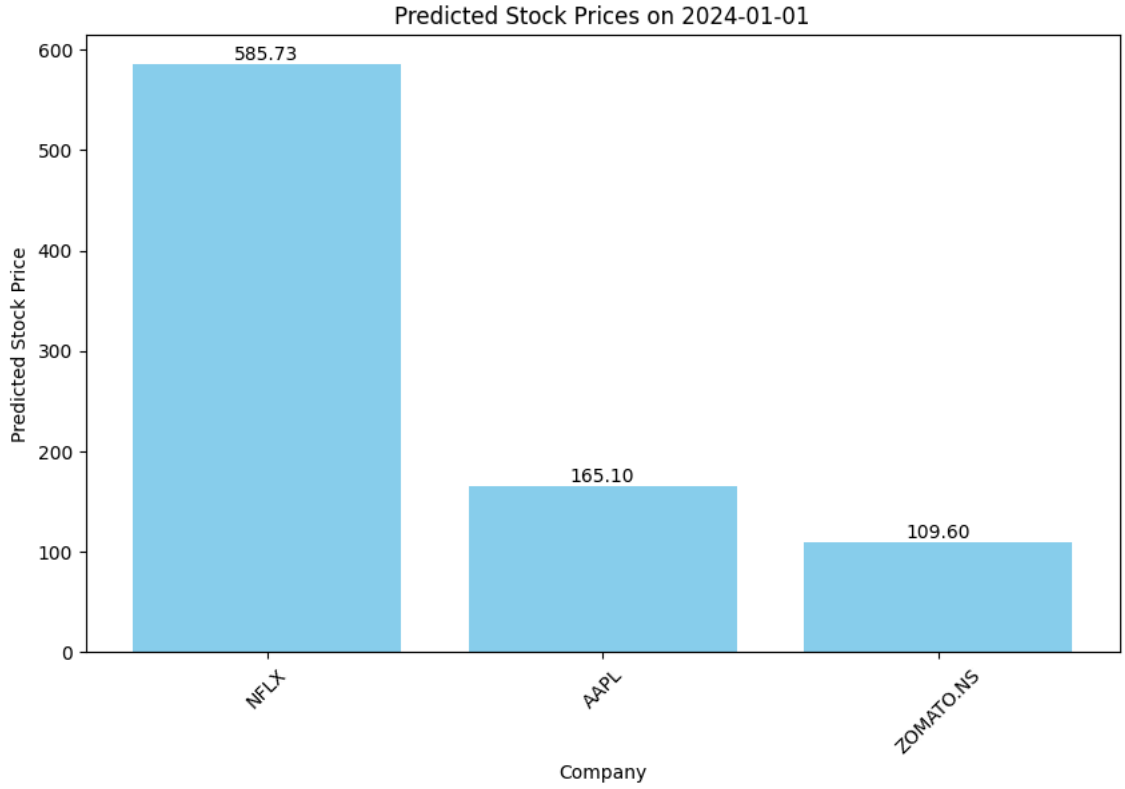
```
def plot_bar_chart_with_labels(data_dict, prediction_date):
    plt.figure(figsize=(10, 6))
    bars = plt.bar(data_dict.keys(), data_dict.values(), color='skyblue')
    plt.xlabel('Company')
    plt.ylabel('Predicted Stock Price')
    plt.title('Predicted Stock Prices on ' + prediction_date)
    plt.xticks(rotation=45)

    # Adding data labels to the bar chart
    for bar, value in zip(bars, data_dict.values()):
        plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height(), f'{value:.2f}', ha='center', va='bottom')

    plt.show()

companies = ['NFLX','AAPL','ZOMATO.NS']
start_date = '2021-01-15'
end_date = '2024-01-15'
prediction_date = '2024-01-01'

future_stock_predictions = predict_future_stock_prices(companies, start_date, end_date, prediction_date)
plot_bar_chart_with_labels(future_stock_predictions, prediction_date)
```

```
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
Accuracy for NFLX: 0.9135822315576293
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestRegre
  warnings.warn(
Accuracy for AAPL: 0.8081515487038614
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestRegre
  warnings.warn(
Accuracy for ZOMATO.NS: 0.9280440092680883
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestRegre
  warnings.warn(
```



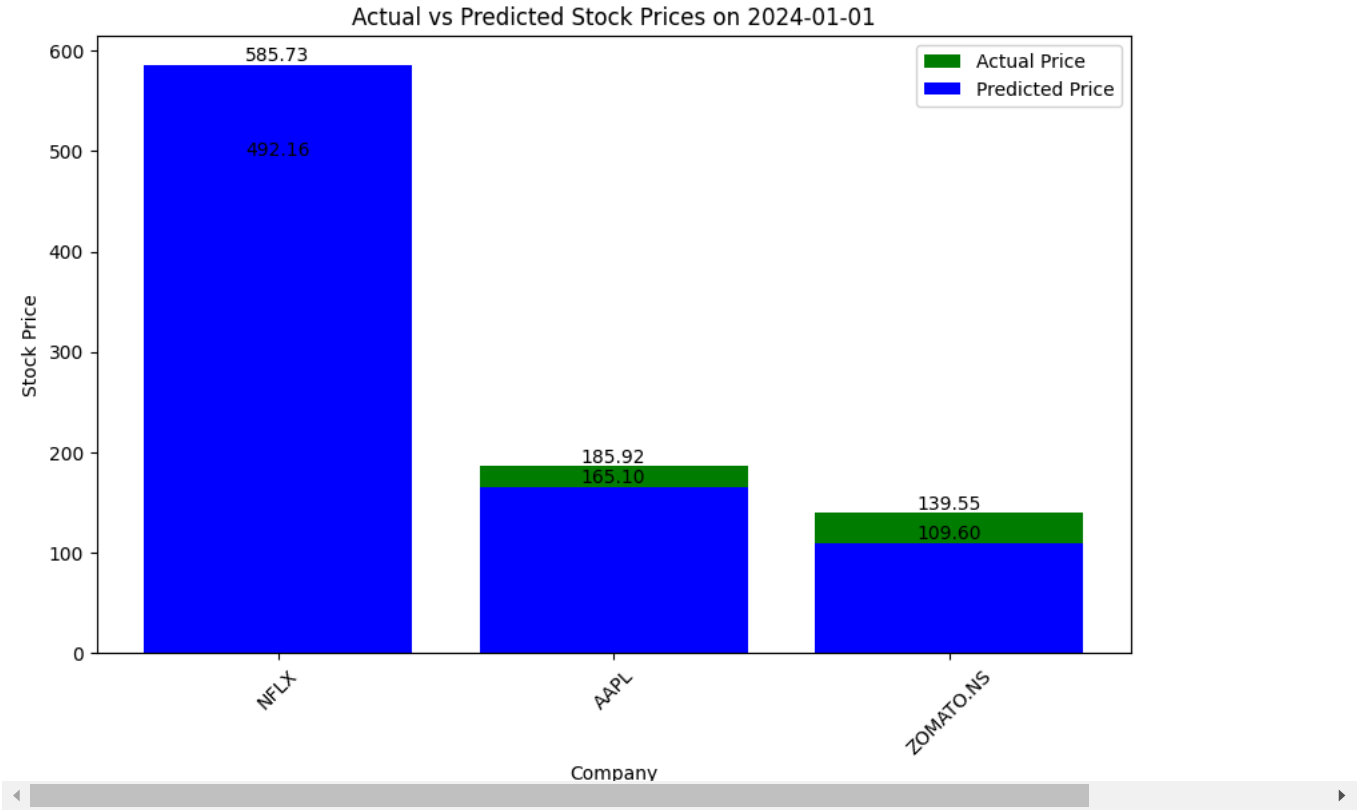
```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import yfinance as yf
import matplotlib.pyplot as plt

def predict_future_stock_prices(companies_list, start_date, end_date, prediction_date):
    def get_stock_data(symbol):
        stock_data = yf.download(symbol, start=start_date, end=end_date)
        return stock_data['Adj Close']

    stock_data = {company: get_stock_data(company) for company in companies_list}
    df = pd.DataFrame(stock_data)
    df.dropna(inplace=True)

    predictions = {}

    for company in companies_list:
        X = df.drop(company, axis=1)
        y = df[company]

        # Split data into training and testing sets
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

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

        # Evaluate accuracy on the testing set
        accuracy = lr_model.score(X_test, y_test)
        print(f'Accuracy for {company}: {accuracy}')

        prediction_input = df.drop(company, axis=1).iloc[-1].values.reshape(1, -1)
        prediction = lr_model.predict(prediction_input)
        predictions[company] = prediction[0]

    return predictions

def plot_bar_chart_with_labels(data_dict, prediction_date):
    plt.figure(figsize=(10, 6))
    bars = plt.bar(data_dict.keys(), data_dict.values(), color='skyblue')
    plt.xlabel('Company')
    plt.ylabel('Predicted Stock Price')
    plt.title('Predicted Stock Prices on ' + prediction_date)
    plt.xticks(rotation=45)

    # Adding data labels to the bar chart
    for bar, value in zip(bars, data_dict.values()):
        plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height(), f'{value:.2f}', ha='center', va='bottom')

    plt.show()

companies = ['NFLX','AAPL','ZOMATO.NS']
start_date = '2021-01-15'
end_date = '2024-01-15'
prediction_date = '2024-01-01'

future_stock_predictions = predict_future_stock_prices(companies, start_date, end_date, prediction_date)
plot_bar_chart_with_labels(future_stock_predictions, prediction_date)
```

```
/usr/local/lib/python3.10/dist-packages/yfinance/base.py:48: FutureWarning: The default dtype for empty Series will be 'object' inst
  _empty_series = pd.Series()
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegression
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegression
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegression
  warnings.warn(
Accuracy for NFLX: 0.7813604068115767
Accuracy for AAPL: 0.03543664413981884
Accuracy for ZOMATO.NS: 0.7841670244276332
```



```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import yfinance as yf
import matplotlib.pyplot as plt

def predict_future_stock_prices(companies_list, start_date, end_date, prediction_date):
    def get_stock_data(symbol):
        stock_data = yf.download(symbol, start=start_date, end=end_date)
        return stock_data['Adj Close']

    stock_data = {company: get_stock_data(company) for company in companies_list}
    df = pd.DataFrame(stock_data)
    df.dropna(inplace=True)

    predictions = {}

    for company in companies_list:
        X = df.drop(company, axis=1)
        y = df[company]

        # Split data into training and testing sets
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

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

        # Evaluate accuracy on the testing set
        accuracy = lr_model.score(X_test, y_test)
        print(f'Accuracy for {company}: {accuracy}')

        prediction_input = df.drop(company, axis=1).iloc[-1].values.reshape(1, -1)
        prediction = lr_model.predict(prediction_input)
        predictions[company] = prediction[0]

    return predictions

def plot_bar_chart_with_labels(actual_data, predicted_data, prediction_date):
    plt.figure(figsize=(10, 6))
    bars1 = plt.bar(actual_data.keys(), actual_data.values(), color='green', label='Actual Price')
    bars2 = plt.bar(predicted_data.keys(), predicted_data.values(), color='blue', label='Predicted Price')
    plt.xlabel('Company')
    plt.ylabel('Stock Price')
    plt.title('Actual vs Predicted Stock Prices on ' + prediction_date)
    plt.xticks(rotation=45)
    plt.legend()

    # Adding data labels to the bar chart for actual prices
    for bar, value in zip(bars1, actual_data.values()):
        plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height(), f'{value:.2f}', ha='center', va='bottom')

    # Adding data labels to the bar chart for predicted prices
    for bar, value in zip(bars2, predicted_data.values()):
        plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height(), f'{value:.2f}', ha='center', va='bottom')

    plt.show()

companies = ['NFLX','AAPL','ZOMATO.NS']
start_date = '2021-01-15'
end_date = '2024-01-15'
prediction_date = '2024-01-01'

actual_stock_prices = {}
for company in companies:
    stock_data = yf.download(company, start=start_date, end=end_date)
    actual_stock_prices[company] = stock_data['Adj Close'].iloc[-1]

future_stock_predictions = predict_future_stock_prices(companies, start_date, end_date, prediction_date)
plot_bar_chart_with_labels(actual_stock_prices, future_stock_predictions, prediction_date)
```

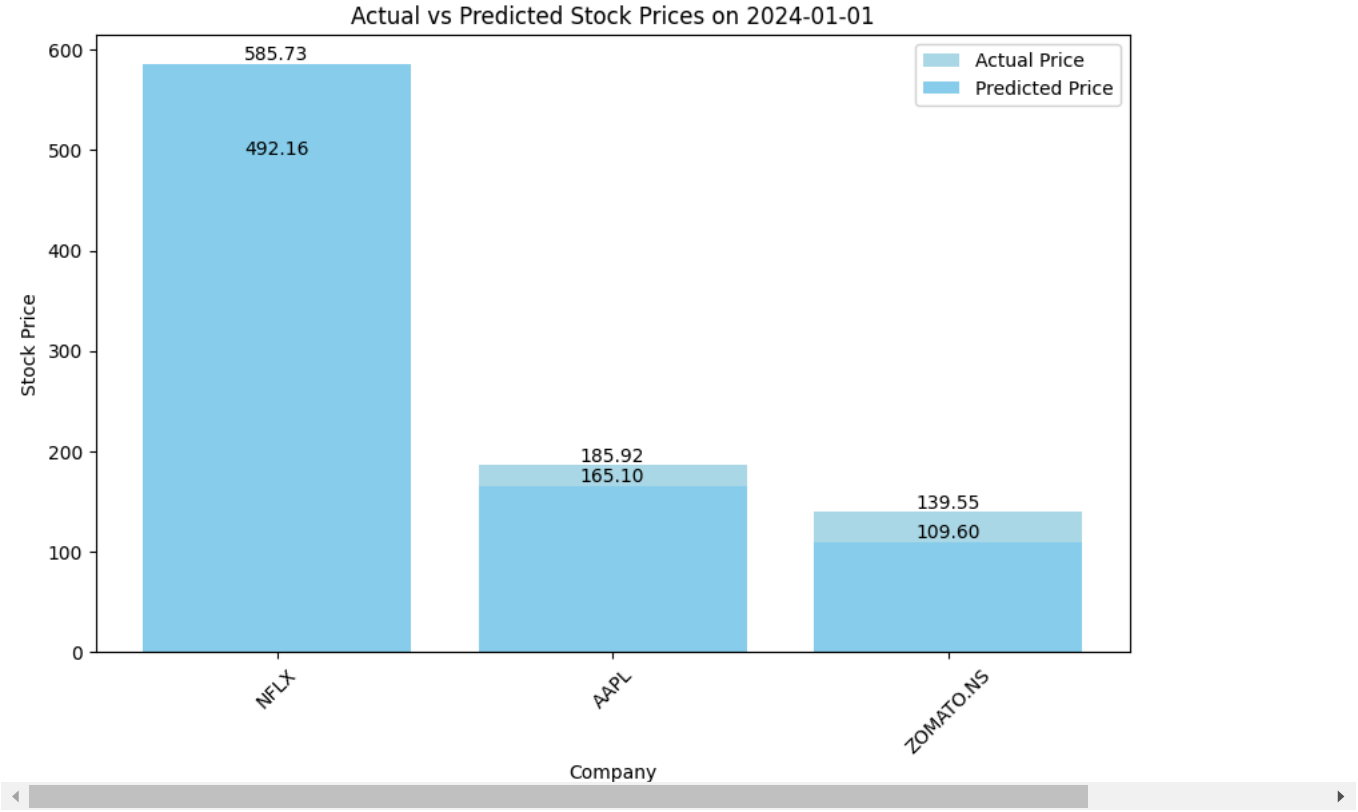
```
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegression
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegression
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegression
  warnings.warn(
Accuracy for NFLX: 0.7813604068115767
Accuracy for AAPL: 0.03543664413981884
Accuracy for ZOMATO.NS: 0.7841670244276332
```



Start coding or generate with AI.

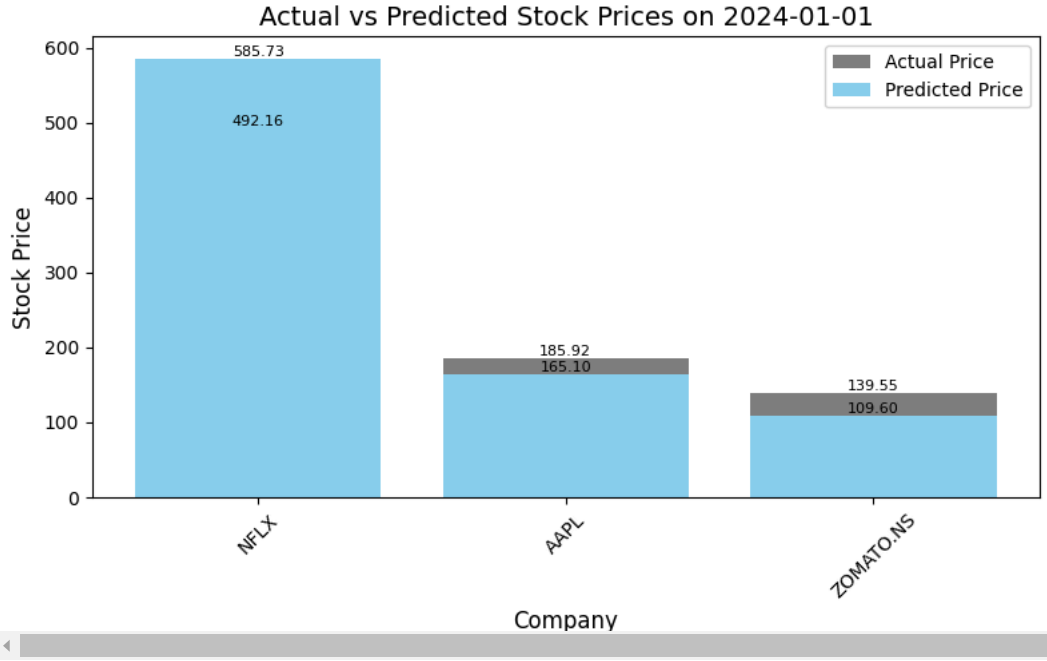
```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import yfinance as yf
import matplotlib.pyplot as plt

def predict_future_stock_prices(companies_list, start_date, end_date, prediction_date):
    def get_stock_data(symbol):
        stock_data = yf.download(symbol, start=start_date, end=end_date)
        return stock_data['Adj Close']

    stock_data = {company: get_stock_data(company) for company in companies_list}
    df = pd.DataFrame(stock_data)
    df.dropna(inplace=True)

    predictions = {}

    for company in companies_list:
        X = df.drop(company, axis=1)
        y = df[company]

        # Split data into training and testing sets
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

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

        # Evaluate accuracy on the testing set
        # accuracy = lr_model.score(X_test, y_test)

        prediction_input = df.drop(company, axis=1).iloc[-1].values.reshape(1, -1)
        prediction = lr_model.predict(prediction_input)
        predictions[company] = prediction[0]

    return predictions

def plot_bar_chart_with_labels(actual_data, predicted_data, prediction_date):
    plt.figure(figsize=(10, 6))
    bars1 = plt.bar(actual_data.keys(), actual_data.values(), color='lightblue', label='Actual Price')
    bars2 = plt.bar(predicted_data.keys(), predicted_data.values(), color='skyblue', label='Predicted Price')
    plt.xlabel('Company')
    plt.ylabel('Stock Price')
    plt.title('Actual vs Predicted Stock Prices on ' + prediction_date)
    plt.xticks(rotation=45)
    plt.legend()

    # Adding data labels to the bar chart for actual prices
    for bar, value in zip(bars1, actual_data.values()):
        plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height(), f'{value:.2f}', ha='center', va='bottom')

    # Adding data labels to the bar chart for predicted prices
    for bar, value in zip(bars2, predicted_data.values()):
        plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height(), f'{value:.2f}', ha='center', va='bottom')

    plt.show()

companies = ['NFLX','AAPL','ZOMATO.NS']
start_date = '2021-01-15'
end_date = '2024-01-15'
prediction_date = '2024-01-01'

actual_stock_prices = {}
for company in companies:
    stock_data = yf.download(company, start=start_date, end=end_date)
    actual_stock_prices[company] = stock_data['Adj Close'].iloc[-1]

future_stock_predictions = predict_future_stock_prices(companies, start_date, end_date, prediction_date)
plot_bar_chart_with_labels(actual_stock_prices, future_stock_predictions, prediction_date)
```

```
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegression
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegression
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegression
  warnings.warn(
```



```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import yfinance as yf
import matplotlib.pyplot as plt

def predict_future_stock_prices(companies_list, start_date, end_date, prediction_date):
    def get_stock_data(symbol):
        stock_data = yf.download(symbol, start=start_date, end=end_date)
        return stock_data['Adj Close']

    stock_data = {company: get_stock_data(company) for company in companies_list}
    df = pd.DataFrame(stock_data)
    df.dropna(inplace=True)

    predictions = {}

    for company in companies_list:
        X = df.drop(company, axis=1)
        y = df[company]

        # Split data into training and testing sets
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

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

        # Evaluate accuracy on the testing set
        # accuracy = lr_model.score(X_test, y_test)

        prediction_input = df.drop(company, axis=1).iloc[-1].values.reshape(1, -1)
        prediction = lr_model.predict(prediction_input)
        predictions[company] = prediction[0]

    return predictions

def plot_bar_chart_with_labels(actual_data, predicted_data, prediction_date):
    plt.figure(figsize=(8, 5))
    bars1 = plt.bar(actual_data.keys(), actual_data.values(), color='gray', label='Actual Price')
    bars2 = plt.bar(predicted_data.keys(), predicted_data.values(), color='skyblue', label='Predicted Price')
    plt.xlabel('Company', fontsize=12)
    plt.ylabel('Stock Price', fontsize=12)
    plt.title('Actual vs Predicted Stock Prices on ' + prediction_date, fontsize=14)
    plt.xticks(rotation=45, fontsize=10)
    plt.yticks(fontsize=10)
    plt.legend(fontsize=10)

    # Adding data labels to the bar chart for actual prices
    for bar, value in zip(bars1, actual_data.values()):
        plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height(), f'{value:.2f}', ha='center', va='bottom', fontsize=8)

    # Adding data labels to the bar chart for predicted prices
    for bar, value in zip(bars2, predicted_data.values()):
        plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height(), f'{value:.2f}', ha='center', va='bottom', fontsize=8)

    plt.tight_layout()
    plt.show()

companies = ['NFLX','AAPL','ZOMATO.NS']
start_date = '2021-01-15'
end_date = '2024-01-15'
prediction_date = '2024-01-01'

actual_stock_prices = {}
for company in companies:
    stock_data = yf.download(company, start=start_date, end=end_date)
    actual_stock_prices[company] = stock_data['Adj Close'].iloc[-1]

future_stock_predictions = predict_future_stock_prices(companies, start_date, end_date, prediction_date)
plot_bar_chart_with_labels(actual_stock_prices, future_stock_predictions, prediction_date)
```

```
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
[*********************100%%**********************]  1 of 1 completed
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegression
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegression
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LinearRegression
  warnings.warn(
```



```
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
import yfinance as yf
import matplotlib.pyplot as plt

def predict_future_stock_prices(companies_list, start_date, end_date, prediction_date):
    def get_stock_data(symbol):
        stock_data = yf.download(symbol, start=start_date, end=end_date)
        return stock_data['Adj Close']

    stock_data = {company: get_stock_data(company) for company in companies_list}
    df = pd.DataFrame(stock_data)
    df.dropna(inplace=True)

    predictions = {}

    for company in companies_list:
        X = df.drop(company, axis=1)
        y = df[company]

        # Split data into training and testing sets
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

        # Random Forest Regressor model
        rf_model = RandomForestRegressor(n_estimators=200, max_depth=10, random_state=42)
        rf_model.fit(X_train, y_train)

        # Evaluate accuracy on the testing set
        accuracy = rf_model.score(X_test, y_test)
        print(f'Accuracy for {company}: {accuracy}')

        prediction_input = df.drop(company, axis=1).iloc[-1].values.reshape(1, -1)
        prediction = rf_model.predict(prediction_input)
        predictions[company] = prediction[0]

    return predictions

def plot_bar_chart_with_labels(actual_data, predicted_data, prediction_date):
    plt.figure(figsize=(8, 5))
    bars1 = plt.bar(actual_data.keys(), actual_data.values(), color='green', label='Actual Price')
    bars2 = plt.bar(predicted_data.keys(), predicted_data.values(), color='skyblue', label='Predicted Price')
    plt.xlabel('Company', fontsize=12)
    plt.ylabel('Stock Price', fontsize=12)
    plt.title('Actual vs Predicted Stock Prices on ' + prediction_date, fontsize=14)
    plt.xticks(rotation=45, fontsize=10)
    plt.yticks(fontsize=10)
    plt.legend(fontsize=10)

    # Adding data labels to the bar chart for actual prices
    for bar, value in zip(bars1, actual_data.values()):
        plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height(), f'{value:.2f}', ha='center', va='bottom', fontsize=8)

    # Adding data labels to the bar chart for predicted prices
    for bar, value in zip(bars2, predicted_data.values()):
        plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height(), f'{value:.2f}', ha='center', va='bottom', fontsize=8)

    plt.tight_layout()
    plt.show()

companies = ['NFLX','AAPL','ZOMATO.NS']
start_date = '2021-01-15'
end_date = '2024-01-15'
prediction_date = '2024-01-01'

actual_stock_prices = {}
for company in companies:
    stock_data = yf.download(company, start=start_date, end=end_date)
    actual_stock_prices[company] = stock_data['Adj Close'].iloc[-1]

future_stock_predictions = predict_future_stock_prices(companies, start_date, end_date, prediction_date)
plot_bar_chart_with_labels(actual_stock_prices, future_stock_predictions, prediction_date)
```

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import yfinance as yf
import matplotlib.pyplot as plt

def predict_future_stock_prices(companies_list, start_date, end_date, prediction_date):
    def get_stock_data(symbol):
        stock_data = yf.download(symbol, start=start_date, end=end_date)
        return stock_data['Adj Close']

    stock_data = {company: get_stock_data(company) for company in companies_list}
    df = pd.DataFrame(stock_data)
    df.dropna(inplace=True)

    predictions = {}

    for company in companies_list:
        X = df.drop(company, axis=1)
        y = df[company]

        # Split data into training and testing sets
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

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

        # Evaluate accuracy on the testing set
        accuracy = lr_model.score(X_test, y_test)
        print(f'Accuracy for {company}: {accuracy}')

        prediction_input = df.drop(company, axis=1).iloc[-1].values.reshape(1, -1)
        prediction = lr_model.predict(prediction_input)
        predictions[company] = prediction[0]

    return predictions
```