In [1]:
```
import pandas as pd
import numpy as np
import datetime as dt
import yfinance as yf
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.ar_model import AutoReg, ar_select_order

# Download NVIDIA historical data
start_date = '2021-01-01'
end_date = '2024-12-31'

# Download data using yfinance
nvda_data = yf.download('NVDA', start=start_date, end=end_date)

# Let's first check what columns we actually have
print("Available columns:", nvda_data.columns)

# Prepare the dataframe - using 'Close' instead of 'Adj Close'
df = nvda_data[['Close']].copy()
df = df.asfreq('d')  # Change frequency to daily
df = df.fillna(method='ffill')  # Fill missing values with forward fill

# Set style for seaborn plot
sns.set_style('darkgrid')
pd.plotting.register_matplotlib_converters()

# Find optimal lag parameters
lags = ar_select_order(df['Close'], maxlag=30)

# Split data into training and testing sets
# Use 80% for training, 20% for testing
train_size = int(len(df) * 0.8)
train_df = df.iloc[:train_size]
test_df = df.iloc[train_size:]

# Create and train the model
# Using 250 trading days (approximately 1 year) for the AR model
train_model = AutoReg(df['Close'], 250).fit(cov_type="HC0")

# Make predictions on test set
start = len(train_df)
end = len(df) - 1
prediction = train_model.predict(start=start, end=end, dynamic=True)

# Generate future dates for prediction
last_date = df.index[-1]
future_dates = pd.date_range(start=last_date + pd.Timedelta(days=1),
                           end='2025-04-30',
                           freq='B')  # 'B' for business days

# Predict future values
forecast = train_model.predict(start=end + 1,
                             end=end + len(future_dates),
                             dynamic=True)
forecast.index = future_dates

# Create single plot with both historical data and forecast
```

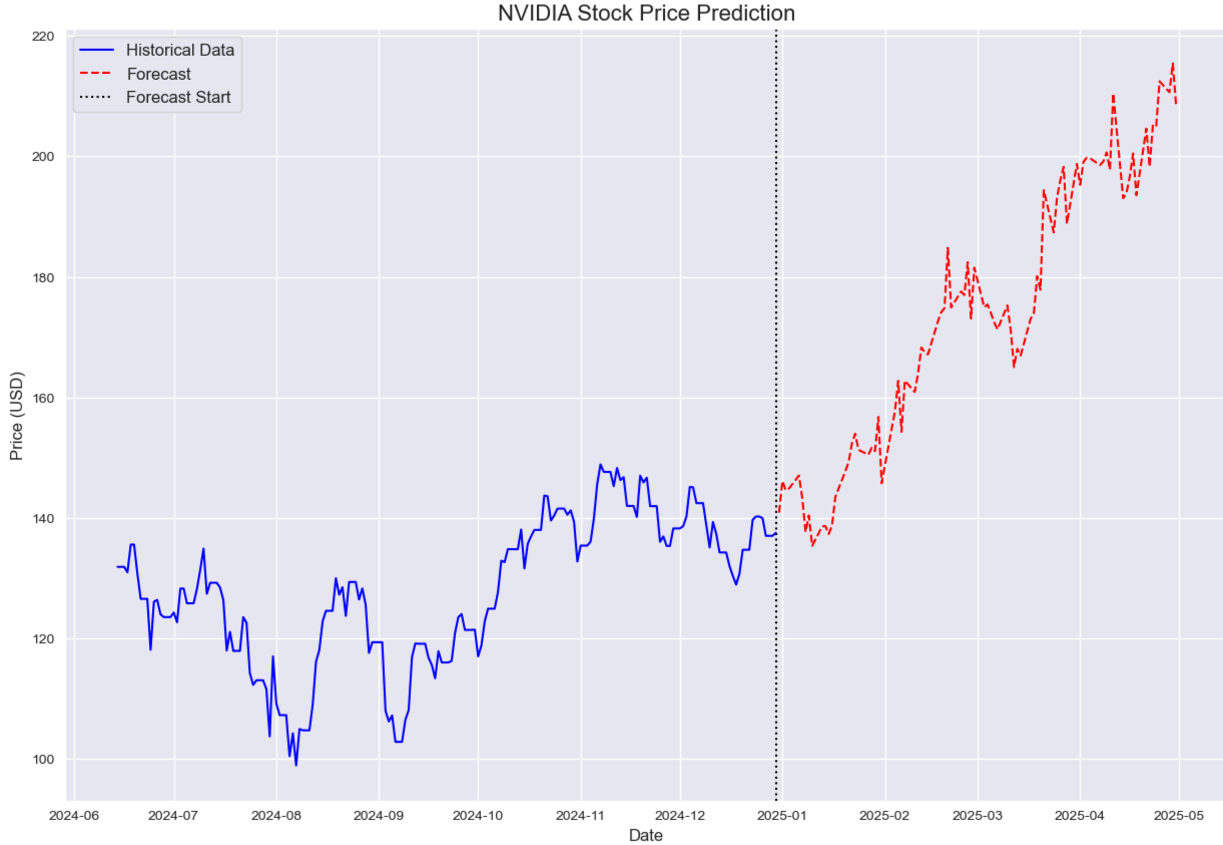
```python
plt.figure(figsize=(15, 10))
plt.title('NVIDIA Stock Price Prediction', fontsize=16)
plt.plot(df.index[-200:], df['Close'][-200:], label='Historical Data', color='blue')
plt.plot(forecast.index, forecast, label='Forecast', color='red', linestyle='--')
plt.axvline(x=last_date, color='black', linestyle=':', label='Forecast Start')
plt.xlabel('Date', fontsize=12)
plt.ylabel('Price (USD)', fontsize=12)
plt.legend(fontsize=12)
plt.grid(True)
plt.show()

# Calculate some statistics about the forecast
forecast_stats = {
    'Predicted Price at End of April 2025': f"${forecast[-1]:.2f}",
    'Average Predicted Price': f"${forecast.mean():.2f}",
    'Predicted Price Range': f"${forecast.min():.2f} - ${forecast.max():.2f}"
}

# Print forecast statistics
print("\nForecast Statistics for Q1 2025:")
for stat, value in forecast_stats.items():
    print(f"{stat}: {value}")

# Calculate predicted monthly averages for Q1 2025
monthly_averages = forecast.resample('M').mean()
print("\nPredicted Monthly Averages:")
for date, price in monthly_averages.items():
    print(f"{date.strftime('%B %Y')}: ${price:.2f}")
```

```
[*********************100%***********************]  1 of 1 completed
Available columns: MultiIndex([( 'Close', 'NVDA'),
            (  'High', 'NVDA'),
            (   'Low', 'NVDA'),
            (  'Open', 'NVDA'),
            ('Volume', 'NVDA')],
           names=['Price', 'Ticker'])
```




```
Forecast Statistics for Q1 2025:
Predicted Price at End of April 2025: $208.45
Average Predicted Price: $173.50
Predicted Price Range: $135.34 - $215.42

Predicted Monthly Averages:
December 2024: $140.87
January 2025: $145.49
February 2025: $169.64
March 2025: $179.82
April 2025: $201.74
```

In [ ]: