w2 Data_Preprocessing_and_Visualization

February 18, 2024

1 Week 02

[3]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
/var/folders/hl/6492kk1939x_mv_8fr9qfz_40000gn/T/ipykernel_40836/2151744951.py:1
: DeprecationWarning:
Pyarrow will become a required dependency of pandas in the next major release of
pandas (pandas 3.0),
(to allow more performant data types, such as the Arrow string type, and better
interoperability with other libraries)
but was not found to be installed on your system.
If this would cause problems for you,
please provide us feedback at https://github.com/pandas-dev/pandas/issues/54466

  import pandas as pd
```

1.1 Check and handle empty data

In stock analysis tasks, using fitted values to fill in empty data is harmful to accurate prediction results, so empty data entries should be deleted.

[23]:
```python
# Load the data
file_path = '../../data/row/NVDA_5_years_daily_data.csv'
data = pd.read_csv(file_path, index_col='date', parse_dates=True)

# Ensure the data is sorted by date for accurate calculations
data.sort_index(inplace=True)

data.isnull().values.any()
```

[23]:
```
False
```

1.2 Data merging and time alignment

The time accuracy of the 5-year historical data we obtained is uneven. You need to merge all historical data together so that all data have the same time accuracy. > Difficulty: How to deal with Volumes of different time scales?

1.3 Auxiliary data broadcast

In addition to historical stock prices, other data such as cash flow and assets and liabilities need to be integrated into your features. But the data we obtain are quarterly or even annual. Because these data are disclosed in financial reports released every quarter, and are manually entered as data by some third-party projects (such as yfinance)

As data for auxiliary forecasting, even if there are only 8 financial reports in a two-year time scale, the financial report data still need to be broadcast to each quarter's time period. The data for the past 3-5 years may need to be obtained by querying the company's financial reports, and Prepare to simplify data dimensions and explore their correlations through subsequent feature engineering.

1.4 Historical volatility

According to the logarithmic price change method formula:

$$X_i = \ln \frac{P_{i+1}}{P_i} = \ln P_{i+1} - \ln P_i$$

$$\bar{X} = \frac{1}{N} \sum X_i$$

$$\sigma = \sqrt{\frac{\sum \left(X_i - \bar{X}\right)^2}{N-1}}$$

Calculate all historical volatility with 1d accuracy and 2 years, and draw it as a plot together with the box chart of historical price data.

Find whether NVDA (i.e. NVIDIA) has corresponding news/event support when volatility occurs abnormally.

[24]:
```python
def get_last_n_years_data(n):
    # Filter for the last n years
    now = pd.Timestamp.now()
    start_date = now - pd.DateOffset(years=n)
    return data.loc[start_date:]['5. adjusted close']

data_2_years = get_last_n_years_data(n=2)
data_3_years = get_last_n_years_data(n=3)

# Calculate daily logarithmic returns
log_returns = np.log(data_3_years / data_3_years.shift(1))

# Calculate the historical volatility
volatility = log_returns.rolling(window=252).std() * np.sqrt(252)

# Calculate the rate of change in volatility
volatility_change = volatility.pct_change() * 100  # Convert to percentage

# Identify dates with abnormal changes in volatility (> 3%)
abnormal_volatility_changes = volatility_change[abs(volatility_change) > 1.5]
```

Save processed data

```
[26]: # Save the log returns, volatility, and its changes to a CSV file
save_data = pd.DataFrame({
    'Adjusted Close Pirce': data_3_years,
    'Log Returns': log_returns,
    'Volatility': volatility,
    'Volatility Change (%)': volatility_change
})
save_data_path = '../../data/processed/NVDA_volatility_and_changes.csv'
save_data.to_csv(save_data_path)
# Provide the path to the saved data
save_data_path
```

```
[26]: '../../data/processed/NVDA_volatility_and_changes.csv'
```

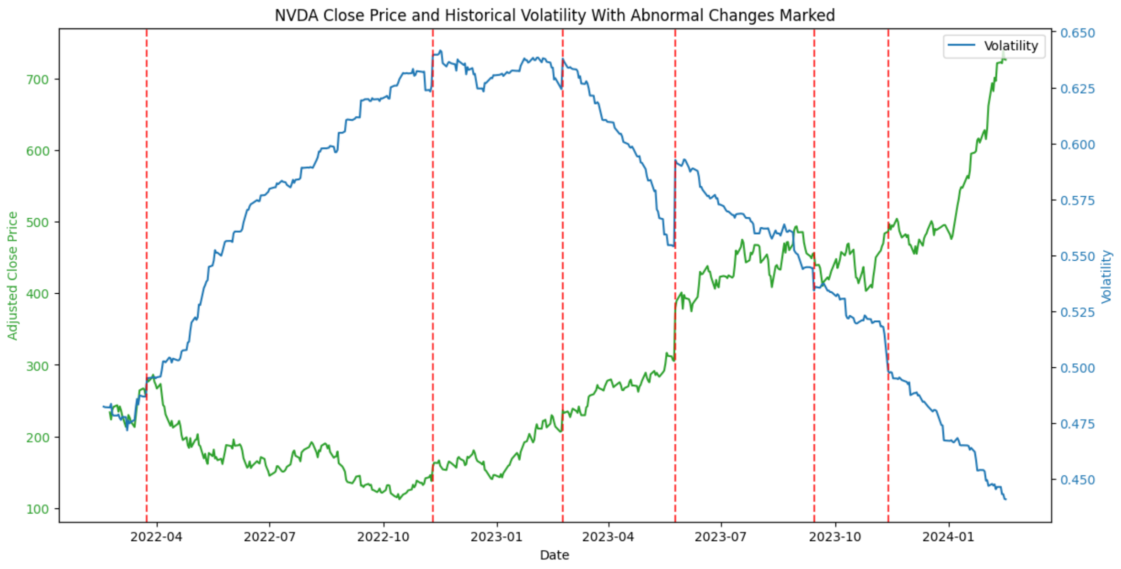
```
[28]: # Plotting with abnormalities marked
fig, ax1 = plt.subplots(figsize=(14, 7))

# Plot close prices
color = 'tab:green'
ax1.set_xlabel('Date')
ax1.set_ylabel('Adjusted Close Price', color=color)
ax1.plot(data_2_years.index, data_2_years,
         color=color, label='Adjusted Close Price')
ax1.tick_params(axis='y', labelcolor=color)

# Plot volatility
ax2 = ax1.twinx()
color = 'tab:blue'
ax2.set_ylabel('Volatility', color=color)
ax2.plot(volatility.index, volatility, color=color, label='Volatility')
ax2.tick_params(axis='y', labelcolor=color)

# Mark abnormalities
for date in abnormal_volatility_changes.index:
    ax2.axvline(x=date, color='r', linestyle='--', alpha=0.75)

plt.title('NVDA Close Price and Historical Volatility With Abnormal Changes␣
 ↪Marked')
plt.legend()
plt.show()
```



[22]:
```
# Print dates and changes of abnormal volatility changes
print("Dates and Changes of Abnormal Volatility Changes:")
print(abnormal_volatility_changes)
```

```
Dates and Changes of Abnormal Volatility Changes:
date
2022-03-24    1.696451
2022-11-10    2.198042
2023-02-23    2.187866
2023-05-25    6.957041
2023-09-14   -1.857797
2023-11-13   -3.214232
Name: 5. adjusted close, dtype: float64
```

- 2022-03-24:
  - Nvidia Would Consider Using Intel as a Foundry, CEO Says
- 2022-11-10:
  - Nvidia will reportedly sell new chips to China that still meet U.S. rules
- 2023-02-23
  - Nvidia shares up 14% on earnings and bullish outlook on A.I.
- 2023-05-25
  - Nvidia shares surge to record close with 24% rally
- 2023-09-14
  - Nvidia and Capital One back Databricks at $43 billion valuation in latest funding round
- 2023-11-13
  - Nvidia unveils H200, its newest high-end chip for training AI models

Faced with data that is unevenly distributed over time (the longer the data, the smaller the amount of data), how should we deal with it when performing feature engineering and window sampling? What is the relationship between data proportion?