CNN in Market Movement Prediction | Data Prep

aiquants Research | © Dayton Nyamai

Application of CNNs in Market Direction Prediction

Summary

This project focuses on the application of deep neural networks, specifically Convolutional Neural Networks (CNNs), in the financial market. It involves the implementation of a Python code that utilizes historical data to predict the direction of price movement for a financial instrument. The code encompasses various stages such as data preprocessing, feature engineering, and visualization, all of which are crucial in training and optimizing the deep neural network models.

Import the necessary libraries

[1]:
```python
import pandas as pd
import numpy as np
```

[2]:
```python
from pylab import mpl, plt
plt.style.use('seaborn-v0_8')
mpl.rcParams['font.family'] = 'serif'
%matplotlib inline
```

The Data

[3]:
```python
# Load the historical data and drop any row with missing values
url = 'https://raw.githubusercontent.com/dayton-nyamai/MarketDLModels/main/data/
 ↪historical_data.csv'
raw = pd.read_csv(url, index_col=0, parse_dates=True).dropna()
raw.info() #the raw data meta information
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 3535 entries, 2010-01-01 to 2023-07-28
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   EURUSD=X  3535 non-null   float64
 1   GBPUSD=X  3535 non-null   float64
 2   AUDUSD=X  3535 non-null   float64
 3   NZDUSD=X  3535 non-null   float64
 4   JPY=X     3535 non-null   float64
```

```
 5   EURJPY=X  3535 non-null   float64
dtypes: float64(6)
memory usage: 193.3 KB
```

Select the symbol and create a DataFrame

[4]:
```
symbol = ['EURUSD=X']
data = pd.DataFrame(raw[symbol])
```

Align dates and rename the column containing the price data to 'price'.

[5]:
```
start_date = data.index.min()
end_date = data.index.max()
data = data.loc[start_date:end_date]
data.rename(columns={'EURUSD=X': 'price'}, inplace=True)
```

Calculate log returns and create direction column

[6]:
```
data['returns'] = np.log(data['price'] / data['price'].shift(1))
data.dropna(inplace=True)
data['direction'] = np.where(data['returns'] > 0, 1, 0)
data.round(4).head()
```

[6]:
```
             price  returns  direction
2010-01-04  1.4424   0.0024          1
2010-01-05  1.4366  -0.0040          0
2010-01-06  1.4404   0.0026          1
2010-01-07  1.4318  -0.0060          0
2010-01-08  1.4411   0.0065          1
```

A histogram providing visual representation of the EUR log returns distribution

[7]:
```
data['returns'].hist(bins=35, figsize=(10, 6));
# Add figure caption
plt.figtext(0.5, -0.01, 'Fig. 1.1 A histogram  showing the distribution of EUR␣
 ↪log returns ', style='italic',ha='center')

# Show the plot
plt.show()
```

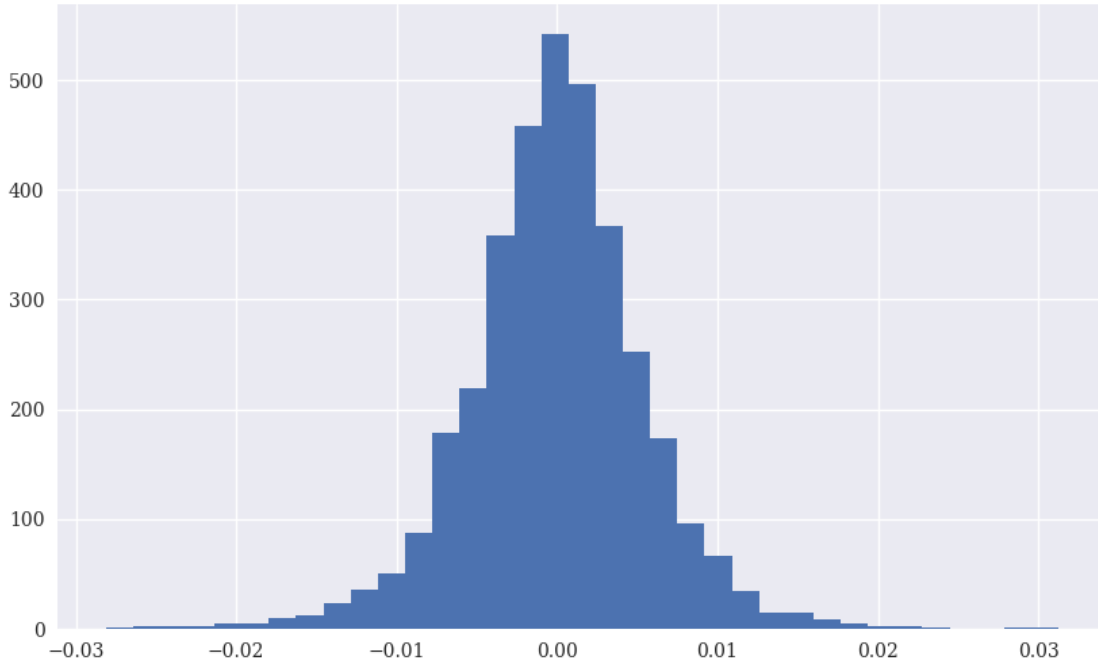

Fig. 1.1 A histogram showing the distribution of EUR log returns

Second, create the features data by lagging the log returns and visualize it in combination with the returns data. We can use various visualization techniques such as scatter plots or line plots to compare the lagged log returns with the returns data.

Create lagged columns

[8]:
```
lags = 5

def create_lags(data):
    global cols
    cols = []
    for lag in range(1, lags + 1):
        col = 'lag_{}'.format(lag)
        data[col] = data['returns'].shift(lag)
        cols.append(col)
create_lags(data)
data.round(4).tail(4)
```

[8]:

	price	returns	direction	lag_1	lag_2	lag_3	lag_4	lag_5
2023-07-25	1.1063	-0.0056	0	-0.0011	-0.0061	-0.0021	-0.0008	0.0009
2023-07-26	1.1050	-0.0011	0	-0.0056	-0.0011	-0.0061	-0.0021	-0.0008
2023-07-27	1.1078	0.0025	1	-0.0011	-0.0056	-0.0011	-0.0061	-0.0021
2023-07-28	1.0979	-0.0090	0	0.0025	-0.0011	-0.0056	-0.0011	-0.0061

Scatter plot based on features and labels data

[9]:
```
data.plot.scatter(x='lag_1', y='lag_2', c='returns',
                  cmap='coolwarm', figsize=(10, 6), colorbar=True)

# Adding vertical and horizontal lines at 0
plt.axvline(0, c='r', ls='--')
plt.axhline(0, c='r', ls='--')

# Add figure caption
plt.figtext(0.4, -0.03,
            'Fig. 1.2 A scatter plot based on features and labels data',
            style='italic',ha='center')

# Show the plot
plt.show()
```

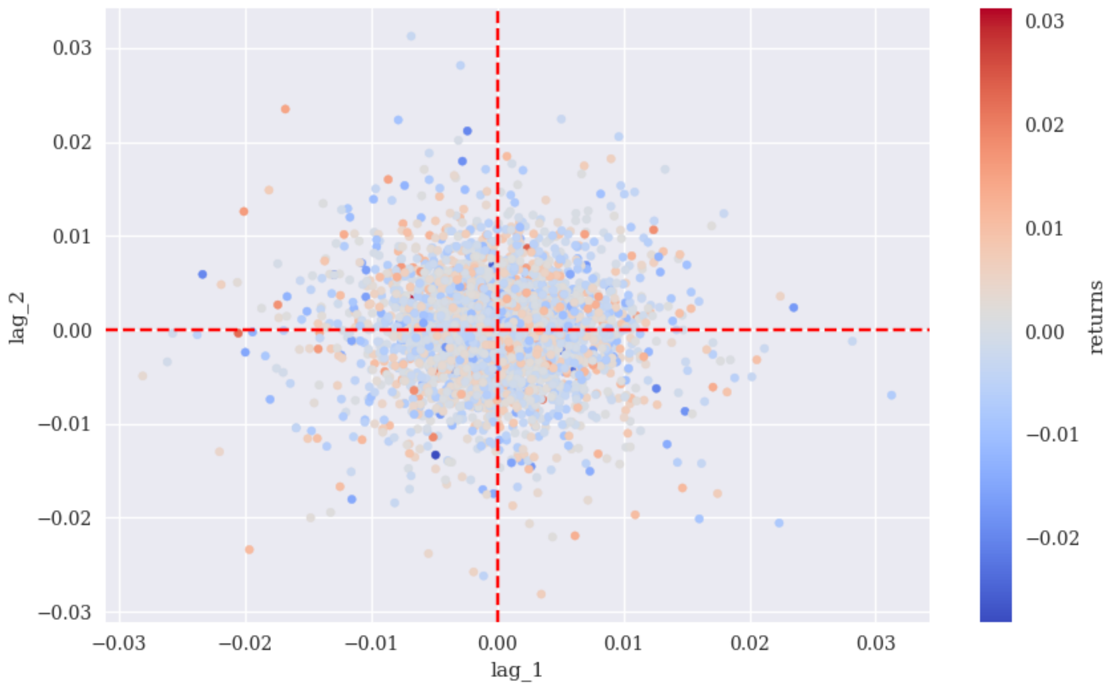

Fig. 1.2 A scatter plot based on features and labels data

With the dataset fully prepared, various deep learning techniques can be employed to forecast market movements based on the provided features. Additionally, these predictions can be utilized to rigorously backtest a trading strategy.