RNNs in Market Movement Prediction | Data Prep

aiquants Research | © Dayton Nyamai

Application of RNNs in Market Direction Prediction

Summary

This project focuses on the application of Recurrent Neural Networks (RNNs) in the 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.

Import the necessary libraries

[10]:
```
import pandas as pd
import numpy as np
```

[11]:
```
from pylab import mpl, plt
plt.style.use('seaborn-v0_8')
mpl.rcParams['font.family'] = 'serif'
%matplotlib inline
```

The Data

[12]:
```
# Load the historical data and drop any row with missing values
url = 'https://raw.githubusercontent.com/dayton-nyamai/MarketDLModels/main/data/
↪data.csv'

raw = pd.read_csv(url,index_col=0, parse_dates=True).dropna()
#raw.info()
```

Select the symbol and create a DataFrame

[13]:
```
symbol = ['EUR=']
data = pd.DataFrame(raw[symbol])
```

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

[14]:
```
start_date = data.index.min()
end_date = data.index.max()
data = data.loc[start_date:end_date]
data.rename(columns={'EUR=': 'price'}, inplace=True)
```

Calculate log returns and create direction column

[20]:
```
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()
```

[20]:

Date	price	returns	direction	lag_1	lag_2	lag_3	lag_4	lag_5
2010-01-12	1.4494	-0.0013	0	0.0070	0.0065	-0.0065	0.0031	-0.0030
2010-01-13	1.4510	0.0011	1	-0.0013	0.0070	0.0065	-0.0065	0.0031
2010-01-14	1.4502	-0.0006	0	0.0011	-0.0013	0.0070	0.0065	-0.0065
2010-01-15	1.4382	-0.0083	0	-0.0006	0.0011	-0.0013	0.0070	0.0065
2010-01-19	1.4298	-0.0059	0	-0.0083	-0.0006	0.0011	-0.0013	0.0070

A histogram providing visual representation of the EUR log returns distribution

[16]:
```
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()
```

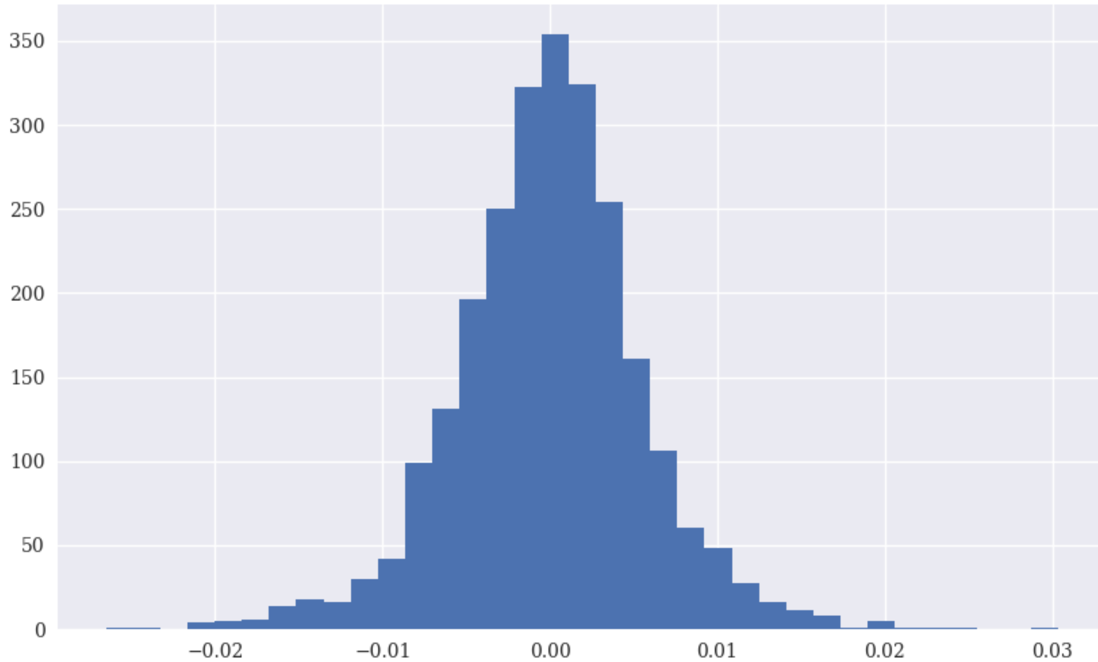

Fig. 1.1 A histogram showing the distribution of EUR log returns

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

[19]:
```
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)
```

[19]:

Date	price	returns	direction	lag_1	lag_2	lag_3	lag_4	lag_5
2019-12-26	1.1096	0.0008	1	0.0001	0.0007	-0.0038	0.0008	-0.0034
2019-12-27	1.1175	0.0071	1	0.0008	0.0001	0.0007	-0.0038	0.0008
2019-12-30	1.1197	0.0020	1	0.0071	0.0008	0.0001	0.0007	-0.0038
2019-12-31	1.1210	0.0012	1	0.0020	0.0071	0.0008	0.0001	0.0007

Scatter plot based on features and labels data

[18]:
```
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()
```

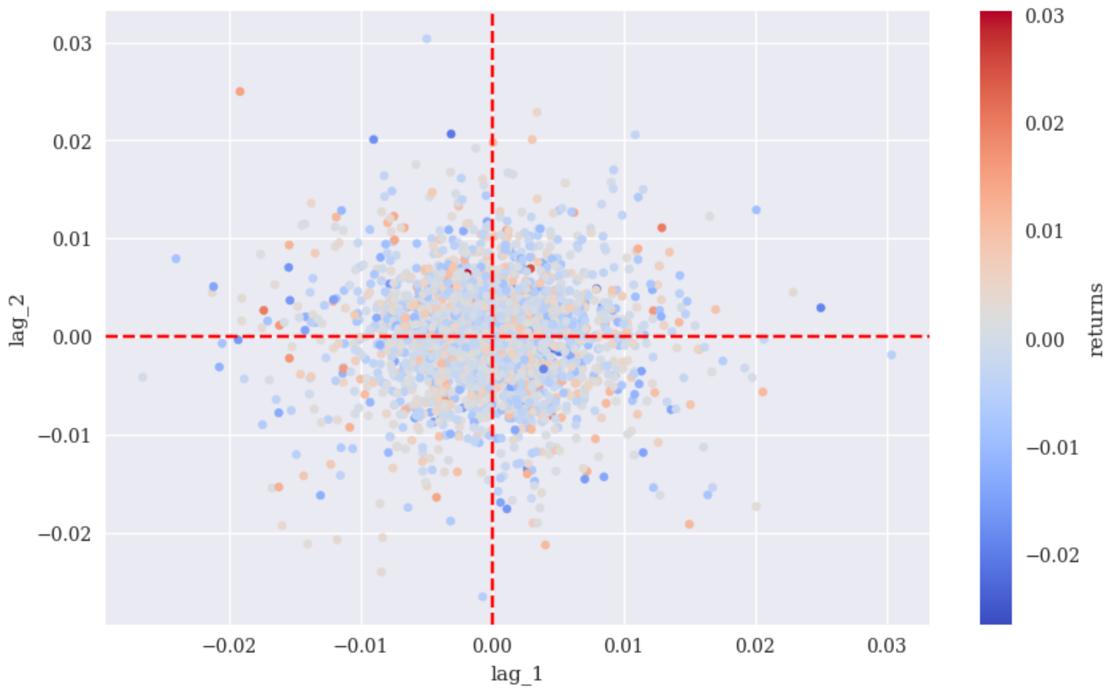

Fig. 1.2 A scatter plot based on features and labels data

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.