COS30018 – Intelligent System

Option B: Stock Price Prediction

Report v0.5

Name Phong Tran
Student Id:104334842

Class: 1-5
Tutor: Tuan Dung Lai

Table of Contents

Dataset

```
[2] # load data
    df = pd.read_csv('/content/AMZN_2020-01-01_2023-01-01.csv')
```

Taken from B.4

	Unnamed: 0	open	high	low	close	adjclose	volume	ticker
0	2020-01-02	93.750000	94.900497	93.207497	94.900497	94.900497	80580000	AMZN
1	2020-01-03	93.224998	94.309998	93.224998	93.748497	93.748497	75288000	AMZN
2	2020-01-06	93.000000	95.184502	93.000000	95.143997	95.143997	81236000	AMZN
3	2020-01-07	95.224998	95.694504	94.601997	95.343002	95.343002	80898000	AMZN
4	2020-01-08	94.902000	95.550003	94.321999	94.598503	94.598503	70160000	AMZN
...	...	...	...	...	...	...	...	...
751	2022-12-23	83.250000	85.779999	82.930000	85.250000	85.250000	57433700	AMZN
752	2022-12-27	84.970001	85.349998	83.000000	83.040001	83.040001	57284000	AMZN
753	2022-12-28	82.800003	83.480003	81.690002	81.820000	81.820000	58228600	AMZN
754	2022-12-29	82.870003	84.550003	82.550003	84.180000	84.180000	54995900	AMZN
755	2022-12-30	83.120003	84.050003	82.470001	84.000000	84.000000	62401200	AMZN

756 rows × 8 columns

Display part of the table at csv file

Multistep

This used to address for display future days prediction with specify variable.

prepare_multistep_data

```python
def prepare_multistep_data(data, features, target_column, n_steps, k):

    feature_data = data[features].values
    target_data = data[[target_column]].values
    feature_scaler = MinMaxScaler(feature_range=(0, 1))
    target_scaler = MinMaxScaler(feature_range=(0, 1))

    feature_data_scaled = feature_scaler.fit_transform(feature_data)
    target_data_scaled = target_scaler.fit_transform(target_data)

    X, y = [], []

    for i in range(len(feature_data_scaled) - n_steps - k + 1):
        X.append(feature_data_scaled[i:i + n_steps, :])  # Use all features
        y.append(target_data_scaled[i + n_steps:i + n_steps + k, 0])  # Use only the target column

    return np.array(X), np.array(y), feature_scaler, target_scaler
```

This function takes the following parameters:

data, dataframe: uses for loading as mentioned in the dataset

features, array: taken all the features column in the dataset target_column, str: target column that we want to use

n_steps, int: How many days that we want to look back in the past

k, int: Days to look in the future

How the function operate:

- The data will used all the feature column to the target column, which is the 'close' data. After that, it will use to create the sequence data in scaled features and target column.

Window Slicing: an algorithm for predict the value k (predict) and its variable needs to check n_steps (past look back days), for example:

X[-15, ...0]: get the days that we want to look back at features

Y[0,.... 15]: predict output days in the future.

As both increase, they will get until reach to the target as X will up to 0, and y will increase to the target future days.

build_multistep_model

```
#build multistep model
def build_multistep_model(n_steps, k):
    model = Sequential()
    model.add(Input(shape=(n_steps, 1)))
    model.add(LSTM(50, activation='relu'))
    model.add(Dense(k))
    model.compile(optimizer='adam', loss='mse')
    return model
```

Parameters

- n_steps, int: number of past days look back
- k, int: number of future days to predict

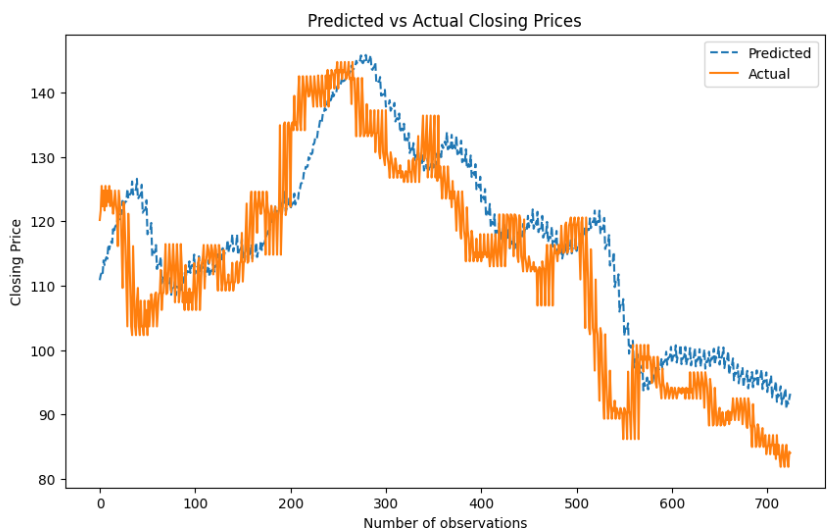
How the function operates:

- Build a LSTM sequential model for shaping the look back days, before the last Dense layer unit is same as future prediction days.

	Predicted	Actual
0	110.857689	120.209503
1	111.632980	121.683998
2	112.035896	125.511002
3	111.727699	122.349998
4	113.920219	124.790001

The result of observation table

The result display of the prediction and actual column as



The plot demonstrates that the actual yellow color has several of days duplicate, like 2 days – 2 times on day 2 or could be 2 times on day 3 and 4.

Multivariate

preprocess_multivariate_single_step

```
def preprocess_multivariate_single_step(data, features, target_column, n_steps):
    # Normalize features
    scaler = MinMaxScaler(feature_range=(0, 1))
    data_scaled = scaler.fit_transform(data[features].values)

    X, y = [], []

    for i in range(len(data_scaled) - n_steps):
        X.append(data_scaled[i:i+n_steps])  # Past n_steps of features
        y.append(data_scaled[i+n_steps, features.index(target_column)])

    X = np.array(X)
    y = np.array(y)
    return X, y, scaler
```

Parameters:

- data, dataframe: taken from dataset
- features, array: list of features we want to specify, for example like open, high, low, so on.
- target_column, str: the column that we want to target
- n_steps, int: the number of lookback days.

How the function work:

- Same as multistep, it needs to normalise to 0 and 1 before applying window slicing. Then apply the array in numpy format for both historical data and targets variable before returning.

build_multivariate_single_step_model

```
def build_multivariate_single_step_model(n_steps, n_features):
    model = Sequential()
    model.add(Input(shape=(n_steps, n_features)))
    model.add(LSTM(50, activation='relu'))
    model.add(Dense(1))
    model.compile(optimizer='adam', loss='mse')
    return model

n_steps = 30
n_features = len(features)  # Number of features used in prediction
multivariate_single_step_model = build_multivariate_single_step_model(n_steps, n_features)
```

Parameters

- n_steps, int: look back days in the past
- n_features, int: future days

How the function works:

- Same as multistep, this also use LSTM for neural network Sequential. However, in the last Dense layer, there only one for just one day prediction.

	Day	Predicted	Actual
0	2022-06-03	122.256982	122.349998
1	2022-06-06	123.775901	124.790001
2	2022-06-07	125.614434	123.000000
3	2022-06-08	125.576198	121.180000
4	2022-06-09	125.319476	116.150002

Display first 5 rows to display the predicted with actual days.

	Day	Predicted	Actual
141	2022-12-23	87.945330	85.250000
142	2022-12-27	87.491677	83.040001
143	2022-12-28	87.170328	81.820000
144	2022-12-29	86.726596	84.180000
145	2022-12-30	86.773059	84.000000

Display last 5 row, the last one shows the predicted day as the multivariate specify.

The plot shows the predicted and actual price in multivariate way.

Combine Multistep and Multivariate

preprocess_multivariate_multistep

```
def preprocess_multivariate_multistep(data, features, target_column, n_steps, k):
    # Separate scalers for features and target column
    feature_scaler = MinMaxScaler(feature_range=(0, 1))
    target_scaler = MinMaxScaler(feature_range=(0, 1))

    features_scaled = feature_scaler.fit_transform(data[features].values)
    target_scaled = target_scaler.fit_transform(data[[target_column]].values)

    X, y = [], []

    for i in range(len(features_scaled) - n_steps - k + 1):
        X.append(features_scaled[i:i + n_steps])
        y.append(target_scaled[i + n_steps:i + n_steps + k, 0])

    X = np.array(X)
    y = np.array(y)

    return X, y, feature_scaler, target_scaler
```

Parameters:

- data, dataframe: used for dataset
- features, array: constant features that we want to use
- target_column, str: target column we want to use
- n_steps, int: days to lookback in the past
- k, int: days to look in the future

Observations:

- This will used for scaling in both feature and target, alongside using window steps with multivariate sequence, before applying in the numpy array assign to X for inputting look back days and y in prediction days in the future.

build_multivariate_multistep_model

```
def build_multivariate_multistep_model(n_steps, n_features, k):

    model = Sequential()
    model.add(Input(shape=(n_steps, n_features)))
    model.add(LSTM(50, activation='relu'))
    model.add(Dense(k))
    model.compile(optimizer='adam', loss='mse')
    return model
```

Parameters

- n_steps, int: days to look back
- n_features, int: number of features combined that we have selected in the array before
- k, int: days prediction

Observation

- Build a LSTM sequential model, with last layer is use same as number of future days prediction.

	Predicted	Actual
0	113.309616	122.081829
1	115.153893	127.347649
2	113.500099	121.362357
3	114.135025	124.790001
4	116.183464	125.922496
...	...	...
575	90.932915	84.180000
576	88.928856	83.299858
577	91.161514	83.480003
578	90.329964	83.634182
579	90.330139	84.000000

580 rows × 2 columns

The table result of combination of multistep and multivariate

The plot of combination in multistep and multivariate function.

References:

https://stackoverflow.com/questions/69785891/how-to-use-the-lstm-model-for-multi-step-forecasting