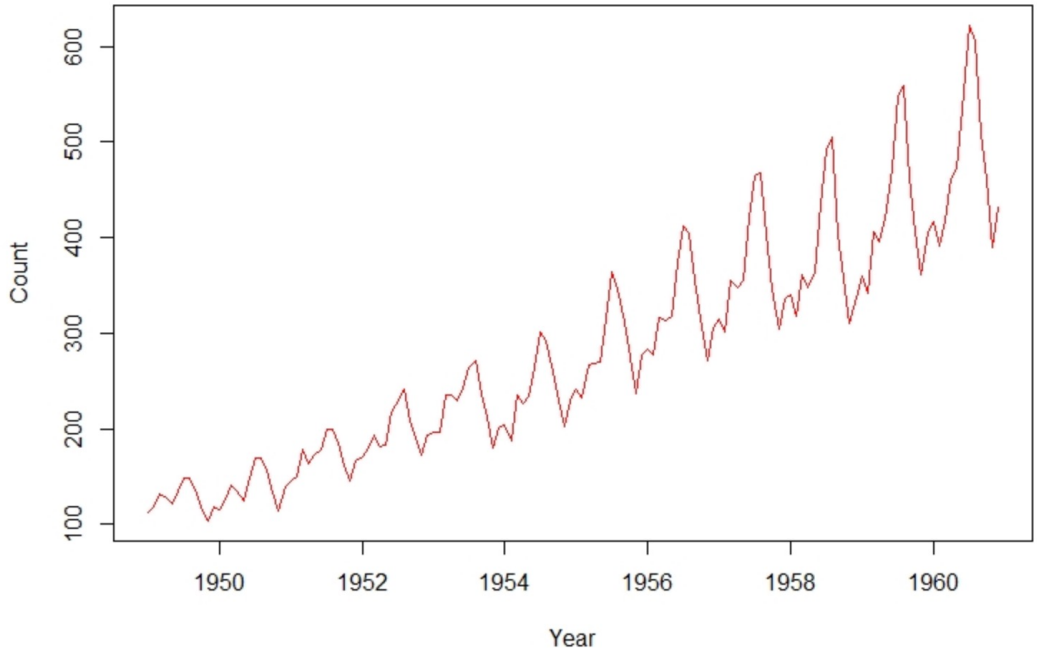
Time Series Data

Working with Time Series Data

◈Time series data representation

- Time series data
  - A sequence of data points collected over time intervals
  - The time intervals
    1) equally spaced as in the case of periodic metrics, or
    2) unequally spaced as in the case of events

[Important feature with time series]
- Successive observations are usually NOT independent
- future values can be predicted from past observations



- Dataset shape: ***N × L × F***
  - ***N**: Data size*
  - ***L**: Length*
  - ***F**: features*

- Examples of time series analysis
  - Weather prediction, earthquake prediction, and energy consumption prediction

Working with Time Series Data

◈Time series data representation

- Bike-sharing data (Washington, D.C., 2011 - 2012) - http://mng.bz/jgOx
- Each row is a separate hour of data
  - For every hour (row), the dataset reports the following features
  - Num of features: 17
    - ➢ Numerical data: 16
    - ➢ Nominal data: 1
      - » Whether situation
        - 1: clear
        - 2: mist
        - 3: light rain/snow
        - 4: heavy rain/snow

- Index of record: `instant`
- Day of month: `day`
- Season: `season` (`1`: spring, `2`: summer, `3`: fall, `4`: winter)
- Year: `yr` (`0`: 2011, `1`: 2012)
- Month: `mnth` (`1` to `12`)
- Hour: `hr` (`0` to `23`)
- Holiday status: `holiday`
- Day of the week: `weekday`
- Working day status: `workingday`
- Weather situation: `weathersit` (`1`: clear, `2`:mist, `3`: light rain/snow, `4`: heavy rain/snow)
- Temperature in °C: `temp`
- Perceived temperature in °C: `atemp`
- Humidity: `hum`
- Wind speed: `windspeed`
- Number of casual users: `casual`
- Number of registered users: `registered`
- Count of rental bikes: `cnt` will be used as target

Working with Time Series Data

◈Bikes sharing data representation (hour-fixed.csv)

```
instant, dteday, season, yr, mnth, hr, holiday, weekday, workingday, weathersit, temp, atemp, hum, windspeed, casual, registered, cnt
1, 2011-01-01, 1, 0, 1, 0, 0, 6, 0, 1, 0.24, 0.2879, 0.81, 0, 3, 13, 16
2, 2011-01-01, 1, 0, 1, 1, 0, 6, 0, 1, 0.22, 0.2727, 0.8,  0, 8, 32, 40
3, 2011-01-01, 1, 0, 1, 2, 0, 6, 0, 1, 0.22, 0.2727, 0.8, 0, 5, 27, 32
4, 2011-01-01, 1, 0, 1, 3, 0, 6, 0, 1, 0.24, 0.2879, 0.75, 0, 3, 10, 13
5, 2011-01-01, 1, 0, 1, 4, 0, 6, 0, 1, 0.24, 0.2879, 0.75, 0, 0, 1, 1
6, 2011-01-01, 1, 0, 1, 5, 0, 6, 0, 2, 0.24, 0.2576, 0.75, 0.0896, 0, 1, 1
7, 2011-01-01, 1, 0, 1, 6, 0, 6, 0, 1, 0.22, 0.2727, 0.8, 0, 2, 0, 2
8, 2011-01-01, 1, 0, 1, 7, 0, 6, 0, 1, 0.2,  0.2576, 0.86, 0, 1, 2, 3
...
...
...
17373, 2012-12-31, 1, 1, 12, 17, 0, 1, 1, 2, 0.26, 0.2879, 0.48, 0.0896, 14,150, 164
17374, 2012-12-31, 1, 1, 12, 18, 0, 1, 1, 2, 0.26, 0.2727, 0.48, 0.1343, 10,112, 122
17375, 2012-12-31, 1, 1, 12, 19, 0, 1, 1, 2, 0.26, 0.2576, 0.6, 0.1642, 11,108, 119
17376, 2012-12-31, 1, 1, 12, 20, 0, 1, 1, 2, 0.26, 0.2576, 0.6, 0.1642, 8, 81, 89
17377, 2012-12-31, 1, 1, 12, 21, 0, 1, 1, 1, 0.26, 0.2576, 0.6, 0.1642, 7, 83, 90
17378, 2012-12-31, 1, 1, 12, 22, 0, 1, 1, 1, 0.26, 0.2727, 0.56, 0.1343, 13, 48, 61
17379, 2012-12-31, 1, 1, 12, 23, 0, 1, 1, 1, 0.26, 0.2727, 0.65, 0.1343, 12, 37, 49
```

Working with Time Series Data

◈Time series data representation

- Bike-sharing data (Washington, D.C., 2011 - 2012) - http://mng.bz/jgOx
- Each row is a separate hour of data
  - Num of features: 17
    - ➢ Numerical data: 16 → 20
    - ➢ Nominal data: 1 → One-hot vector
      - » Whether situation
        - 1: clear → [1, 0, 0, 0]
        - 2: mist → [0, 1, 0, 0]
        - 3: light rain/snow → [0, 0, 1, 0]
        - 4: heavy rain/snow → [0, 0, 0, 1]

link_dl/_01_code/_03_real_world_data_to_tensors/i_hourly_bikes_sharing_data.py

Working with Time Series Data

◈Time series data representation

— Loading the bike-sharing data and change it into tensor

```
instant, dteday, season, yr, mnth, hr, holiday, weekday, workingday,
weathersit, temp, atemp, hum, windspeed, casual, registered, cnt
1, 2011-01-01, 1, 0, 1, 0, 0, 6, 0, 1, 0.24, 0.2879, 0.81, 0, 3, 13, 16
2, 2011-01-01, 1, 0, 1, 1, 0, 6, 0, 1, 0.22, 0.2727, 0.8,  0, 8, 32, 40
3, 2011-01-01, 1, 0, 1, 2, 0, 6, 0, 1, 0.22, 0.2727, 0.8,  0, 5, 27, 32
4, 2011-01-01, 1, 0, 1, 3, 0, 6, 0, 1, 0.24, 0.2879, 0.75, 0, 3, 10, 13
5, 2011-01-01, 1, 0, 1, 4, 0, 6, 0, 1, 0.24, 0.2879, 0.75, 0, 0, 1,  1
...
```

```
instant, dteday, season, yr, mnth, hr, holiday, weekday, workingday,
weathersit, temp, atemp, hum, windspeed, casual, registered
1, 2011-01-01, 1, 0, 1, 0, 0, 6, 0, 1, 0.24, 0.2879, 0.81, 0, 3, 13
2, 2011-01-01, 1, 0, 1, 1, 0, 6, 0, 1, 0.22, 0.2727, 0.8,  0, 8, 32
3, 2011-01-01, 1, 0, 1, 2, 0, 6, 0, 1, 0.22, 0.2727, 0.8,  0, 5, 27
4, 2011-01-01, 1, 0, 1, 3, 0, 6, 0, 1, 0.24, 0.2879, 0.75, 0, 3, 10
5, 2011-01-01, 1, 0, 1, 4, 0, 6, 0, 1, 0.24, 0.2879, 0.75, 0, 0, 1
...
```

```
BASE_PATH = str(Path(__file__).resolve().parent.parent.parent) 1
sys.path.append(BASE_PATH)

bikes_path = os.path.join(
  BASE_PATH, "_00_data", "e_time-series-bike-sharing-dataset", "hour-fixed.csv”
)

bikes_numpy = np.loadtxt(
    fname=bikes_path, dtype=np.float32, delimiter=",",
    skiprows=1, converters={
        1: lambda x: float(x[8:10])
    } # 1: Column Index, 2011-01-07 --> 07 --> 7.0 (from date to day)
)
bikes_data = torch.from_numpy(bikes_numpy).to(torch.float)   # >>> torch.Size([17520, 17])
print(bikes_data.shape)          # >>> torch.Size([17520, 17])
bikes_target = bikes_data[:, -1].unsqueeze(dim=-1)  # 'cnt'
bikes_data = bikes_data[:, :-1]  # >>> torch.Size([17520, 16])
```

link_dl/_01_code/_03_real_world_data_to_tensors/i_hourly_bikes_sharing_data.py

Working with Time Series Data

◈Time series data representation

– For all data, change nominal data by one-hot encoding

```
instant, dteday, season, yr, mnth, hr, holiday, weekday, workingday,
weathersit, temp, atemp, hum, windspeed, casual, registered
1, 2011-01-01, 1, 0, 1, 0, 0, 6, 0, 1, 0.24, 0.2879, 0.81, 0, 3, 13
2, 2011-01-01, 1, 0, 1, 1, 0, 6, 0, 1, 0.22, 0.2727, 0.8,  0, 8, 32
3, 2011-01-01, 1, 0, 1, 2, 0, 6, 0, 1, 0.22, 0.2727, 0.8,  0, 5, 27
4, 2011-01-01, 1, 0, 1, 3, 0, 6, 0, 1, 0.24, 0.2879, 0.75, 0, 3, 10
5, 2011-01-01, 1, 0, 1, 4, 0, 6, 0, 1, 0.24, 0.2879, 0.75, 0, 0, 1
...
```

```python
eye_matrix = torch.eye(4)

data_torch_list = []
for idx in range(bikes_data.shape[0]):  # range(17520)
  hour_data = bikes_data[idx]           # hour_data.shape: [17]
  weather_onehot = eye_matrix[hour_data[9].long() - 1]
  concat_data_torch = torch.cat(tensors=(hour_data, weather_onehot), dim=-1)
  # concat_data_torch.shape: [20]
  data_torch_list.append(concat_data_torch)

bikes_data = torch.stack(data_torch_list, dim=0)
bikes_data = torch.cat(
  [bikes_data[:, 1:9], bikes_data[:, 10:]], dim=-1
) # Drop 'instant' and 'whethersit' columns
print(bikes_data.shape)
# bikes_data.shape: [17520, 18]
```

```
dteday, season, yr, mnth, hr, holiday, weekday, workingday,
temp, atemp, hum, windspeed, casual, registered
2011-01-01, 1, 0, 1, 0, 0, 6, 0, 0.24, 0.2879, 0.81, 0, 3, 13, 1, 0, 0, 0
2011-01-01, 1, 0, 1, 1, 0, 6, 0, 0.22, 0.2727, 0.8,  0, 8, 32, 1, 0, 0, 0
2011-01-01, 1, 0, 1, 2, 0, 6, 0, 0.22, 0.2727, 0.8,  0, 5, 27, 1, 0, 0, 0
2011-01-01, 1, 0, 1, 3, 0, 6, 0, 0.24, 0.2879, 0.75, 0, 3, 10, 1, 0, 0, 0
2011-01-01, 1, 0, 1, 4, 0, 6, 0, 0.24, 0.2879, 0.75, 0, 0,  1, 1, 0, 0, 0
...
```

link_dl/_01_code/_03_real_world_data_to_tensors/i_hourly_bikes_sharing_data.py

Working with Time Series Data

◈Time series data representation

- Data size: train size, validation size, test size

```
sequence_size = 24
validation_size = 96
test_size = 24
y_normalizer = 100

data_size = len(bikes_data) - sequence_size
print("data_size: {0}".format(data_size))
# >>> data_size: 17496

train_size = data_size - (validation_size + test_size)
print("train_size: {0}, validation_size: {1}, test_size: {2}".format(
  train_size, validation_size, test_size
))
# >>> train_size: 17376, validation_size: 96, test_size: 24
```

link_dl/_01_code/_03_real_world_data_to_tensors/i_hourly_bikes_sharing_data.py

Working with Time Series Data

◈Train Data Preparation

```
row_cursor = 0

X_train_list = []
y_train_regression_list = []
for idx in range(0, train_size):
  sequence_data = bikes_data[idx: idx + sequence_size]
  sequence_target = bikes_target[idx + sequence_size - 1]
  X_train_list.append(sequence_data)
  y_train_regression_list.append(sequence_target)
  row_cursor += 1

X_train = torch.stack(X_train_list, dim=0).to(torch.float)
y_train_regression = torch.tensor(y_train_regression_list, dtype=torch.float32) / y_normalizer

m = X_train.mean(dim=0, keepdim=True)
s = X_train.std(dim=0, unbiased=False, keepdim=True)
X_train = (X_train - m) / s

print(X_train.shape, y_train_regression.shape)   # >>> torch.Size([17376, 24, 18]) torch.Size([17376])
```

idx: idx + sequence_size
(= 0: 4)

```
dteday, season, yr, mnth, hr, holiday, weekday, workingday,
temp, atemp, hum, windspeed, casual, registered
2011-01-01, 1, 0, 1, 0, 0, 6, 0, 0.24, 0.2879, 0.81, 0, 3, 13, 1, 0, 0, 0
2011-01-01, 1, 0, 1, 1, 0, 6, 0, 0.22, 0.2727, 0.8,  0, 8, 32, 1, 0, 0, 0
2011-01-01, 1, 0, 1, 2, 0, 6, 0, 0.22, 0.2727, 0.8,  0, 5, 27, 1, 0, 0, 0
2011-01-01, 1, 0, 1, 3, 0, 6, 0, 0.24, 0.2879, 0.75, 0, 3, 10, 1, 0, 0, 0
2011-01-01, 1, 0, 1, 4, 0, 6, 0, 0.24, 0.2879, 0.75, 0, 0,  1, 1, 0, 0, 0
...
```

idx + sequence_size - 1
(= 3)

```
cnt
16
40
32
13
1
...
```

Working with Time Series Data

◈Validation Data Preparation

```
X_validation_list = []
y_validation_regression_list = []

for idx in range(row_cursor, row_cursor + validation_size):
  sequence_data = bikes_data[idx: idx + sequence_size]
  sequence_target = bikes_target[idx + sequence_size - 1]
  X_validation_list.append(sequence_data)
  y_validation_regression_list.append(sequence_target)
  row_cursor += 1

X_validation = torch.stack(X_validation_list, dim=0).to(torch.float)
y_validation_regression = torch.tensor(y_validation_regression_list, dtype=torch.float32) / y_normalizer

X_validation = (X_validation - m) / s

print(X_validation.shape, y_validation_regression.shape)
# >>> torch.Size([96, 24, 18]) torch.Size([96])
```

link_dl/_01_code/_03_real_world_data_to_tensors/i_hourly_bikes_sharing_data.py

Working with Time Series Data

◈Test Data Preparation

```
X_test_list = []
y_test_regression_list = []

for idx in range(row_cursor, row_cursor + test_size):
  sequence_data = bikes_data[idx: idx + sequence_size]
  sequence_target = bikes_target[idx + sequence_size - 1]
  X_test_list.append(sequence_data)
  y_test_regression_list.append(sequence_target)
  row_cursor += 1

X_test = torch.stack(X_test_list, dim=0).to(torch.float)
y_test_regression = torch.tensor(y_test_regression_list, dtype=torch.float32) / y_normalizer

X_test -= (X_test - m) / s

print(X_test.shape, y_test_regression.shape)
# >>> torch.Size([24, 24, 18]) torch.Size([24])
```

Cryptocurrency Data

Working with Cryptocurrency Data

◈Cryptocurrency data representation (BTC_KRW.csv)

– https://finance.yahoo.com/quote/BTC-KRW/history/

Date,	Open,	High,	Low,	Close,	Volume
2014.9.17,	482611.8125,	483811.0313,	468121.0313,	473203.5,	21787470960
2014.9.18,	472713.0313,	476276.5313,	430991.4063,	442818.25,	35976322560
2014.9.19,	442466.6563,	447515.4063,	401278.375,	411989.3438,	39571102935
2014.9.20,	411861,	441730.5313,	406862.4063,	426711.75,	38469009780
2014.9.21,	425857.0938,	430387.1563,	410304.0313,	416189.6563,	27737663355
2014.9.22,	416480.8125,	423497.8125,	413571.1875,	418238.0938,	25092704000
2014.9.23,	418175.6875,	459175.125,	411846.7813,	453179.0313,	46898970050
2014.9.24,	453137.4688,	453774.5313,	437829.875,	440035.8438,	31845763629
…					
…					
…					
2023.10.26,	45587572,	47494456,	45515484,	46684732,	3.41709E+13
2023.10.27,	46686716,	47301572,	45854332,	46157296,	2.62528E+13
2023.10.28,	46157096,	46417712,	45348664,	45989148,	2.22665E+13
2023.10.29,	45986332,	46653140,	45941688,	46232964,	1.37796E+13
2023.10.30,	46232688,	47119508,	46040368,	46841244,	1.51357E+13
2023.10.31,	46832104,	46993024,	46033316,	46457872,	2.31396E+13

Working with Cryptocurrency Data

◈Pandas Dataframe from CSV

```
btc_krw_path = os.path.join(BASE_PATH, "_00_data", "k_cryptocurrency", "BTC_KRW.csv")
df = pd.read_csv(btc_krw_path)
print(df)
```

	Date	Open	...	Close	Volume
0	2014.9.17	4.826118e+05	...	4.732035e+05	2.178747e+10
1	2014.9.18	4.727130e+05	...	4.428182e+05	3.597632e+10
2	2014.9.19	4.424667e+05	...	4.119893e+05	3.957110e+10
3	2014.9.20	4.118610e+05	...	4.267118e+05	3.846901e+10
4	2014.9.21	4.258571e+05	...	4.161897e+05	2.773766e+10
...	...	...	...	...	...
3327	2023.10.27	4.668672e+07	...	4.615730e+07	2.625280e+13
3328	2023.10.28	4.615710e+07	...	4.598915e+07	2.226650e+13
3329	2023.10.29	4.598633e+07	...	4.623296e+07	1.377960e+13
3330	2023.10.30	4.623269e+07	...	4.684124e+07	1.513570e+13
3331	2023.10.31	4.683210e+07	...	4.645787e+07	2.313960e+13

```
row_size = len(df)
print("row_size:", row_size) # >>> row_size: 3332

columns = df.columns
print([column for column in columns]) # >>> ['Date', 'Open', 'High', 'Low', 'Close', 'Volume']

date_list = df['Date']
df = df.drop(columns=['Date'])
```

link_dl/_01_code/_03_real_world_data_to_tensors/h_cryptocurrency_data.py

Working with Cryptocurrency Data

◈Data Size

— train size, validation size, test size

```
sequence_size = 10
validation_size = 100
test_size = 50

data_size = row_size - sequence_size
print("data_size: {0}".format(data_size))                    # >>> data_size: 3322

train_size = data_size - (validation_size + test_size)

print("train_size: {0}, validation_size: {1}, test_size: {2}".format(
  train_size, validation_size, test_size
))
# >>> train_size: 3172, validation_size: 100, test_size: 50
```

link_dl/_01_code/_03_real_world_data_to_tensors/h_cryptocurrency_data.py

Working with Cryptocurrency Data

◈Train Data Preparation (1/2)

```
row_cursor = 0
y_normalizer = 1.0e7

X_train_list = []
y_train_regression_list = []
y_train_classification_list = []
y_train_date = []

for idx in range(train_size):
  sequence_data = df.iloc[idx: idx + sequence_size].values    # sequence_data.shape: (sequence_size, 5)
  X_train_list.append(torch.from_numpy(sequence_data))
  y_train_regression_list.append(df.iloc[idx + sequence_size]["Close"])
  y_train_classification_list.append(
    1 if df.iloc[idx + sequence_size]["Close"] >= df.iloc[idx + sequence_size - 1]["Close"] else 0
  )
  y_train_date.append(date_list[idx + sequence_size])
  row_cursor += 1
```

idx: idx + sequence_size

	Open	High	Low	Close	Volume
0	4.826118e+05	4.838110e+05	4.681210e+05	4.732035e+05	2.178747e+10
1	4.727130e+05	4.762765e+05	4.309914e+05	4.428182e+05	3.597632e+10
2	4.424667e+05	4.475154e+05	4.012784e+05	4.119893e+05	3.957110e+10
3	4.118610e+05	4.417305e+05	4.068624e+05	4.267118e+05	3.846901e+10
4	4.258571e+05	4.303872e+05	4.103040e+05	4.161897e+05	2.773766e+10
...	...	...	...	...	
3327	4.668672e+07	4.730157e+07	4.585433e+07	4.615730e+07	2.625280e+13
3328	4.615710e+07	4.641771e+07	4.534866e+07	4.598915e+07	2.226650e+13
3329	4.598633e+07	4.665314e+07	4.594169e+07	4.623296e+07	1.377960e+13
3330	4.623269e+07	4.711951e+07	4.604037e+07	4.684124e+07	1.513570e+13
3331	4.683210e+07	4.699302e+07	4.603332e+07	4.645787e+07	2.313960e+13

idx + sequence_size

Working with Cryptocurrency Data

◈Train Data Preparation (2/2)

```
...
# Features
X_train = torch.stack(X_train_list, dim=0).to(torch.float)
# Labels for regression task
y_train_regression = torch.tensor(y_train_regression_list, dtype=torch.float32) / y_normalizer
# Labels for classification task
y_train_classification = torch.tensor(y_train_classification_list, dtype=torch.int64)
print(y_train_classification)
# >>> tensor([0, 0, 1,  ..., 0, 0, 1])

m = X_train.mean(dim=0, keepdim=True)
s = X_train.std(dim=0, unbiased=False, keepdim=True)
X_train = (X_train - m) / s

print(X_train.shape, y_train_regression.shape, y_train_classification.shape)
# >>> torch.Size([3172, 10, 5]) torch.Size([3172]) torch.Size([3172])
print("Label - Start Date: {0} ~ End Date: {1}".format(y_train_date[0], y_train_date[-1]))
# >>> Label - Start Date: 2014.9.27 ~ End Date: 2023.6.3
```

Working with Cryptocurrency Data

◈Validation Data Preparation (1/2)

```
X_validation_list = []
y_validation_regression_list = []
y_validation_classification_list = []
y_validation_date = []

for idx in range(row_cursor, row_cursor + validation_size):
  sequence_data = df.iloc[idx: idx + sequence_size].values # sequence_data.shape: (sequence_size, 5)
  X_validation_list.append(torch.from_numpy(sequence_data))
  y_validation_regression_list.append(df.iloc[idx + sequence_size]["Close"])
  y_validation_classification_list.append(
    1 if df.iloc[idx + sequence_size]["Close"] >= df.iloc[idx + sequence_size - 1]["Close"] else 0
  )
  y_validation_date.append(date_list[idx + sequence_size])

  row_cursor += 1
```

link_dl/_01_code/_03_real_world_data_to_tensors/h_cryptocurrency_data.py

Working with Cryptocurrency Data

◈Validation Data Preparation (2/2)

```
...
# Features
X_validation = torch.stack(X_validation_list, dim=0).to(torch.float)
# Labels for regression task
y_validation_regression = torch.tensor(y_validation_regression_list, dtype=torch.float32) / y_normalizer
# Labels for classification task
y_validation_classification = torch.tensor(y_validation_classification_list, dtype=torch.int64)


X_validation = (X_validation - m) / s

print(X_validation.shape, y_validation_regression.shape, y_validation_classification.shape)
# >>> torch.Size([100, 10, 5]) torch.Size([100]) torch.Size([100])

print("Label - Start Date: {0} ~ End Date: {1}".format(y_validation_date[0], y_validation_date[-1]))
# >>> Label - Start Date: 2023.6.4 ~ End Date: 2023.9.11
```

link_dl/_01_code/_03_real_world_data_to_tensors/h_cryptocurrency_data.py

Working with Cryptocurrency Data

◈Test Data Preparation (1/2)

```
X_test_list = []
y_test_regression_list = []
y_test_classification_list = []
y_test_date = []

for idx in range(row_cursor, row_cursor + test_size):
  sequence_data = df.iloc[idx: idx + sequence_size].values  # sequence_data.shape: (sequence_size, 5)
  X_test_list.append(torch.from_numpy(sequence_data))
  y_test_regression_list.append(df.iloc[idx + sequence_size]["Close"])
  y_test_classification_list.append(
    1 if df.iloc[idx + sequence_size]["Close"] > df.iloc[idx + sequence_size - 1]["Close"] else 0
  )
  y_test_date.append(date_list[idx + sequence_size])
  row_cursor += 1
```

link_dl/_01_code/_03_real_world_data_to_tensors/h_cryptocurrency_data.py

Working with Cryptocurrency Data

◈Test Data Preparation (2/2)

```
...
# Features
X_test = torch.stack(X_test_list, dim=0).to(torch.float)
# Labels for regression task
y_test_regression = torch.tensor(y_test_regression_list, dtype=torch.float32) / y_normalizer
# Labels for classification task
y_test_classification = torch.tensor(y_test_classification_list, dtype=torch.int64)



X_test = (X_test - m) / s

print(X_test.shape, y_test_regression.shape, y_test_classification.shape)
# >>> torch.Size([50, 10, 5]) torch.Size([50]) torch.Size([50])

print("Label - Start Date: {0} ~ End Date: {1}".format(y_test_date[0], y_test_date[-1]))
# >>> Label - Start Date: 2023.9.12 ~ End Date: 2023.10.31
```

link_dl/_01_code/_03_real_world_data_to_tensors/h_cryptocurrency_data.py

Working with Cryptocurrency Data

◈ Data visulalization