Industrial Metals Forecaster Documentation

Release 1.0.0

Oliver Boom

Aug 09, 2019

CONTENTS:

CHAPTER

ONE

PREPROCESSING MODULE

Src.preprocessing.**clean_data**(*df*, *n_std=20*)
Removes any outliers that are further than a chosen number of standard deviations from the mean

Parameters

- **df** (*dataframe*) – the finacial time series
- **n_std** (*int*) – the number of standard deviations from the mean

Returns the cleaned financial time series

Return type dataframe

Src.preprocessing.**clean_dict_gen**(*universe_dict*)
Returns a dictionary of cleaned dataframes

Parameters **universe_dict** (*dict*) – a dictionary of financial time series

Returns the cleaned financial time series

Return type dict

Src.preprocessing.**column_rename**(*universe_dict*)
Appends the name of the instrument name to the columns

Parameters **universe_dict** (*dict*) – a dictionary of financial time series

Returns the financial time series

Return type dict

Src.preprocessing.**dimension_reduce**(*data_X*, *n_dim*)
Performing PCA to reduce the amount of

Parameters

- **data_X** (*np.array*) – array to perform reduction on
- **n_dim** (*int*) – number of dimensions to reduce to

Returns reduced dataset

Return type np.array

Src.preprocessing.**dimension_selector**(*data_X*, *thresh=0.98*)
Returns the number of dimensions that reaches the threshold level of desired variance

Parameters

- **data_X** (*np.array*) – dataset to perform reduction on
- **thresh** (*float*) – the amount of variance that must be contained in reduced dataset

Returns the amount of dimensions needed to conatin the threshold variance

Return type int

Src.preprocessing.**feature_spawn**(*df*)
Spawns features for each instrument Returns df with the following columns for each instrument Log Returns EWMA 1 day EWMA 1 week EWMA 1 month EWMA 1 quarter EWMA 6 months EWMA 1 year Rolling vol 1 week Rolling vol 1 month Rolling vol 1 quarter

Src.preprocessing.**generate_dataset**(*universe_dict*, *lag=5*, *lg_returns_only=True*, *price_only=False*)
Generates the full dataset

Parameters

- **universe_dict** (*dict*) – a dictionary of financial time series
- **lag** (*int*) – the amount of days the returns are calculated between
- **lg_returns_only** (*bool*) – whether to return a dataset of log returns only
- **price_only** (*bool*) – whether to return a dataset of raw prices only

Returns the financial time series

Return type dataframe

Src.preprocessing.**generate_lg_return**(*df_full*, *lag=1*)
Returns a dictionary containing dataframes with the additional log returns column

Parameters

- **df_full** (*dataframe*) – the financial time series
- **lag** (*int*) – the amount of days the returns are calculated between

Returns the financial time series with log returns

Return type dataframe

Src.preprocessing.**generate_target**(*df_full*, *target_col='price_cu_lme'*, *lag=5*)
Generate the target variable

Src.preprocessing.**log_returns**(*series*, *lag=1*)
Calculate log returns between adjacent close prices

Parameters

- **series** (*numpy array*) – prices to calculate the log returns on
- **lag** (*int*) – the amount of days the returns are calculated between

Returns the series of log returns

Return type numpy array

Src.preprocessing.**price_rename**(*universe_dict*)
Renaming the column of the dataframe values to price

Parameters **universe_dict** (*dict*) – a dictionary of financial time series

Returns financial time series

Return type dict

Src.preprocessing.**slice_series**(*data_X*, *data_y*, *series_length*, *dataset_pct=1.0*)
TODO

Src.preprocessing.**truncate_window_length**(*universe_dict*)
Chopping the length of all of the dataframes to ensure that they are all between the same dates

Parameters **universe_dict** (*dict*) – the financial time series

Returns the truncated financial time series

Return type dict

Src.preprocessing.**universe_select**(*path*, *commodity_name*)
Selects the instruments believed to be of interest for the commodity selected

Parameters

- **path** (*type string*) – path to the csv folder
- **commodity_name** (*type string*) – the name of the metal being inspected

Returns financial time series relevant to the commodity

Return type dict

CHAPTER

TWO

DEEPLEARNING MODULE

class Src.deeplearning.**DeepLearning**(*model*, *data_X*, *data_y*, *n_epochs*, *optimiser*, *batch_size*, *loss_function=MSELoss()*, *device='cpu'*, *seed=42*, *debug=True*, *disp_freq=20*, *fig_disp_freq=50*, *early_stop=True*, *early_verbose=False*, *patience=50*, *rel_tol=0*, *scaler_data_X=None*, *scaler_data_y=None*)

Class to perform training and validation for a given model

Parameters

- **model** (*LSTM*) – the neural network model
- **data_X** (*np.array*) – the training dataset
- **data_y** (*np.array*) – the target dataset
- **n_epochs** (*int*) – the number of epochs of training
- **optimiser** (*torch.optim*) – the type of optimiser used
- **batch_size** (*int*) – the batch size
- **loss_function** (*torch.nn.modules.loss*) – the loss function used
- **device** (*string*) – running on cpu or CUDA
- **seed** (*int*) – the random seed set
- **debug** (*bool*) – whether to print some parameters for checking
- **disp_freq** (*int*) – the frequency that training/validation metrics will be printed
- **fig_disp_freq** (*int*) – the frequency that training/validation prediction figures will be made
- **early_stop** (*bool*) – whether early stopping is utilized
- **early_verbose** (*bool*) – whether to print out the early stopping counter
- **patience** (*int*) – the amount of epochs without improvement before stopping
- **rel_tol** (*float*) – the relative improvement percentage that must be achieved
- **scaler_data_X** – the data X scaler object for inverse scaling
- **scaler_data_y** – the dataX y scaler object for inverse scaling

Rtype scaler_data_X sklearn.preprocessing.data.MinMaxScaler

Rtype scaler_data_y sklearn.preprocessing.data.MinMaxScaler

create_data_loaders()

Forms iterators to pipeline in the data

evaluate(*model*, *test_loader*)
Evaluates the performance of the network on given data for a given model

A lot of overlap of code with validation. Only kept separate due to inspection of attibutes made easier when running simulations if kept separate

Parameters **test_loader** (*torch.utils.data.dataloader.DataLoader*) – the iterator that feeds in the data of choice

Returns the error metric for that dataset

Return type float

live_pred_plot()
Plots the training predictions, validation predictions and the live training/validation losses

size_check()
Checks the size of the datasets

train(*train_loader*)
Performs a single training cycle and returns the loss metric for the training dataset

Parameters **train_loader** (*torch.utils.data.dataloader.DataLoader*) – the iterator that feeds in the training data

Returns the error metric for that epoch

Return type float

train_val_test()
Splits the dataframes in to a training, validation and test set and creates torch tensors from the underlying numpy arrays

training_wrapper()
The wrapper that performs the training and validation

validate(*val_loader*)
Evaluates the performance of the network on unseen validation data

Parameters **val_loader** (*torch.utils.data.dataloader.DataLoader*) – the iterator that feeds in the validation data

Returns the error metric for that epoch

Return type float

class Src.deeplearning.**early_stopping**(*patience*, *rel_tol*, *verbose=False*)
Counter to implement early stopping

If validation accuracy has not relative improved below a relative tolerance set by the user than it breaks the training

If rel_tol is set to 0 it becomes a common counter

Parameters

- **patience** (*int*) – the amount of epochs without improvement before stopping
- **rel_tol** (*float*) – the relative improvement percentage that must be achieved
- **verbose** (*bool*) – whether to print the count number
- **best_score** (*float*) – the best score achieved so far
- **counter** (*int*) – the amount of epochs without improvement so far
- **stop** (*bool*) – whether stopping criteria is achieved

Src.deeplearning.**full_save**(*model*, *model_name*, *optimiser*, *num_epoch*, *learning_rate*, *momentum*, *weight_decay*, *use_lg_returns*, *PCA_used*, *data_X*, *train_loss*, *val_loss*, *test_loss*, *train_time*, *hidden_dim*, *mse*, *mae*, *mde*, *path='Models/CSVs/'*)
Saves the models weights and hyperparameters to a pth file and csv file

Src.deeplearning.**model_load**(*model_name*, *path='Models/'*)
Loading function for models from google drive

Src.deeplearning.**model_save**(*model*, *name*, *path='Models/'*)
Saving function to keep track of models

Src.deeplearning.**param_strip**(*param*)
Strips the key text info out of certain parameters

Src.deeplearning.**set_seed**(*seed*, *device='cpu'*)
Sets the random seeds to ensure detemrinistic behaviour

Parameters

- **seed** (*int*) – the random seed number that is set
- **device** (*string*) – whether running on cpu or CUDA

Returns confirmation that seeds have been set

Return type bool

CHAPTER

THREE

UTILS MODULE

Src.utils.**check_day_frequency**(*df*, *day_col_name='ds'*)
Returns a barchart showing the frequency of the days of the week within a dataframe

Parameters **df** (*pd.DataFrame*) – the time series to visualise

Src.utils.**check_length**(*universe_dict*)
Checks the name of all the dataframes in the dictionary of instruments

Parameters **universe_dict** (*dict*) – a dictionary of financial time series

Src.utils.**df_std**(*df*, *col_name*)
Returns the standard deviation of a dataframes column

Parameters

- **df** (*pd.DataFrame*) – a dataframe of time series
- **col_name** – the column of interest

Returns the standard deviation of the series on interest

Return type float

Src.utils.**evaluate**(*y_true*, *y_pred*, *log_ret=False*)
Calculated the error metric for a dataframe of predictions and observed values

Parameters

- **y_true** (*np.array*) – The observed values
- **y_pred** (*np.array*) – The predicted values

Return mse, mae, mde Returns the mean squared error, mean absolute accuracy and mean directional accuracy metrics

Return type float

Src.utils.**inverse_log_returns**(*original_prices*, *log_returns*, *lag=5*, *shift=0*)
Takes a dataframes of predicted log returns and original prices and returns an array of predicted absolute prices

Parameters

- **original_prices** (*pd.DataFrame*) – a dataframe of absolute prices
- **log_returns** (*pd.DataFrame*) – a dataframe of log returns
- **lag** (*int*) – the lag duration of the log returns
- **shift** (*int*) – whether to offset the series forwards of backwards

Returns the raw prices indicated by the log returns

Return type pd.Series

Src.utils.**mean_absolute_percentage_error**(*y_true*, *y_pred*)
Calculated the mean absolute percentage error metric between two arrays

Parameters

- **y_true** (*np.array*) – The observed values
- **y_pred** (*np.array*) – The predicted values

Returns The mean absolute percentage error of the series

Return type float

Src.utils.**mean_directional_accuracy**(*y_true*, *y_pred*)
Calculated the mean directional accuracy error metric between two series

Parameters

- **y_true** (*pd.Series*) – The observed values
- **y_pred** (*pd.Series*) – The predicted values

Returns The mean direcional accuracy of the series

Return type float

Src.utils.**mean_directional_accuracy_log_ret**(*y_true*, *y_pred*)
Calculated the mean directional accuracy error metric between two log return series

Parameters

- **y_true** (*pd.Series*) – The observed values
- **y_pred** (*pd.Series*) – The predicted values

Returns The mean direcional accuracy of the series

Return type float

Src.utils.**visualise_df**(*df*)
Visualises the features for an instrument :param df: the time series to visualise :type df: pd.DataFrame

Src.utils.**visualise_universe**(*universe_dict*)
Plots the price and log return for every instrument in the univese dictionary

Parameters **universe_dict** (*dict*) – a dictionary of financial time series to visualise

3.1 Indices and tables

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

INDEX