

---

# **stockDL Documentation**

***Release v0.0.3***

**Ashish Papanai**

**Mar 02, 2021**



**CONTENTS:**

<b>1</b>	<b>stockDL</b>	<b>1</b>
1.1	stockDL package . . . . .	1
<b>2</b>	<b>Indices and tables</b>	<b>5</b>
	<b>Python Module Index</b>	<b>7</b>
	<b>Index</b>	<b>9</b>



## STOCKDL

## 1.1 stockDL package

### 1.1.1 Submodules

#### 1.1.2 stockDL.calculations module

This module comprises all functions to calculate the net yield and gross yield.

```
class stockDL.calculations.Calculations
    Bases: object

    gross_portfolio(df, w)
    gross_yield(df, v)
    net_yield(df, v)
    separate_ones(u)
```

#### 1.1.3 stockDL.data module

This module helps in data collection from Yahoo Finance API with the help of yfinance library. The stock data is loaded from its unique stock symbol, also called ticker. After the data is loaded, we drop the axis which isn't being used by us in the library. After dropping the unnecessary axis we calculate and store the first days of each trading month in a list.

```
class stockDL.data.Data_Loader(ticker)
    Bases: object

    date_calculations()

    end_date
        Stores the first day of each month from the starting date of stock to the day of executing the library

    load_data()
```

### 1.1.4 stockDL.main module

This Module handles the generation of results in the CLI, based on the ticker passed as the argument to the object of the Main class. The execution begins by retrieving the financial data from yfinance library which is handled in the data module of stockDL. After the data is collected the first trading day of each month is stored in a list by the date\_calculation() function in the data module. The data is then preprocessed by the preprocessor module which creates a window to run the predictions on. It also comprises min-max scaling which reduces sudden highs and lows in the data which would have resulted in anomalies. The training module trains the model created in the model module on the data retrieved by the preprocessing module. The plots module helps in plotting the necessary graphs for better visualisation for EDA. The result module uses the calculation and market modules to run the necessary calculations on the predictions, and generate net and the gross yield on the predictions obtained.

```
class stockDL.main.Main(ticker)
    Bases: object

    data_preprocessor
        This variable stores the monthly data for analysis by the model

    df_monthly
        This instance of train module will help in preventing the training to occur more than once thus reducing
        the probability of overfitting.

    plots
        Generates a comparison plot of the 4 methods used

    result
        Converts the pandas dataframe to JSON for better utilisation by web developers

    results
        Stores the final result in a pandas dataframe

    ticker
        this module helps in loading the data by calling an instance of data.py module and also helps in data
        preprocessing.

    train
        Trains the data on the defined models
```

### 1.1.5 stockDL.market module

This module stores the stock market variables related to the stock ticker, This module must be run after training the model by calling the train\_model() function in the train module.

```
class stockDL.market.Market(ticker)
    Bases: object

    v_lstm
        Stores the sign of the lstm predictions as +ve, -ve or zero which implies profit, loss and no profit no loss in
        the trade.

    v_mix
        Stores the sign of the mix model predictions as +ve, -ve or zero which implies profit, loss and no profit no
        loss in the trade.

    w_lstm
        This stores the predictions of the lstm model and reshapes it.

    w_mix
        This stores the predictions of the mix model (conv1D + LSTM) and reshapes it.
```

### 1.1.6 stockDL.models module

This module stores the brain of the library which is the Deep Learning model. The two Deep Learning strategies are defined in their respective methods. This module requires to run the preprocessing module so that the model gets the data to work on.

```
class stockDL.models.Models (ticker)
    Bases: object

    LSTM_Model (window, features)

    Mix_LSTM_Model (window, features)
```

### 1.1.7 stockDL.plots module

This module helps in plotting the various details related to the model and the predictions, This module requires the processed data from the pre-processing module, training data from the train module, Market information from the market module, and the calculations from the calculations module.

```
class stockDL.plots.Plots (ticker)
    Bases: object

    comparison_plots ()

    in_out ()

    plot_predictions ()

    plot_training_data (model_selected, metric='loss', val_metric='val_loss')
```

### 1.1.8 stockDL.preprocessing module

This module makes the data ready for predictions and calculations, This module requires the data module to get the data from the Yahoo finance API. This module has 3 main components: 1. Convert the daily stock data to monthly data for analysis based on the opening prices for the month. 2. Creating an analysis window for predictions. [6 years is the window size in stockDL] 3. Making data RNN ready.

```
class stockDL.preprocessing.data_preprocessing (ticker)
    Bases: object

    create_window (data, window_size)

    data_scaling (dfm)

    monthly_df (df)
```

### 1.1.9 stockDL.results module

This is the final module responsible for the result calculation and parsing the dataframe to JSON. This module requires data from preprocessing, calculations, market and all their dependencies.

```
class stockDL.results.Results (ticker)
    Bases: object

    result_calculations ()
```

### 1.1.10 stockDL.train module

This module handles the training of the two-deep learning strategies used in this library. It requires preprocessing and models modules and their dependencies.

```
class stockDL.train.Training(ticker)
    Bases: object

    learning_rate_reduction
        Uncomment and add tensorboard to callbacks to run TensorBoard for visualisation self.tensorboard =
        keras.callbacks.TensorBoard(
            log_dir='TensorBoard_Logs', histogram_freq=1, embeddings_freq=1
        )

    models
        Prevents false minima by reducing the learning rates on plateaus.

    train_model()
        Stores the history of the LSTM model.
```

### 1.1.11 Module contents

Version of module

## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

### S

- `stockDL`, 4
- `stockDL.calculations`, 1
- `stockDL.data`, 1
- `stockDL.main`, 2
- `stockDL.market`, 2
- `stockDL.models`, 3
- `stockDL.plots`, 3
- `stockDL.preprocessing`, 3
- `stockDL.results`, 3
- `stockDL.train`, 4



## C

Calculations (*class in stockDL.calculations*), 1  
 comparison\_plots() (*stockDL.plots.Plots method*), 3  
 create\_window() (*stockDL.preprocessing.data\_preprocessing method*), 3

## D

Data\_Loader (*class in stockDL.data*), 1  
 data\_preprocessing (*class in stockDL.preprocessing*), 3  
 data\_preprocessor (*stockDL.main.Main attribute*), 2  
 data\_scaling() (*stockDL.preprocessing.data\_preprocessing method*), 3  
 date\_calculations() (*stockDL.data.Data\_Loader method*), 1  
 df\_monthly (*stockDL.main.Main attribute*), 2

## E

end\_date (*stockDL.data.Data\_Loader attribute*), 1

## G

gross\_portfolio() (*stockDL.calculations.Calculations method*), 1  
 gross\_yield() (*stockDL.calculations.Calculations method*), 1

## I

in\_out() (*stockDL.plots.Plots method*), 3

## L

learning\_rate\_reduction (*stockDL.train.Training attribute*), 4  
 load\_data() (*stockDL.data.Data\_Loader method*), 1  
 LSTM\_Model() (*stockDL.models.Models method*), 3

## M

Main (*class in stockDL.main*), 2  
 Market (*class in stockDL.market*), 2  
 Mix\_LSTM\_Model() (*stockDL.models.Models method*), 3

Models (*class in stockDL.models*), 3  
 models (*stockDL.train.Training attribute*), 4  
 module  
     stockDL, 4  
     stockDL.calculations, 1  
     stockDL.data, 1  
     stockDL.main, 2  
     stockDL.market, 2  
     stockDL.models, 3  
     stockDL.plots, 3  
     stockDL.preprocessing, 3  
     stockDL.results, 3  
     stockDL.train, 4  
 monthly\_df() (*stockDL.preprocessing.data\_preprocessing method*), 3

## N

net\_yield() (*stockDL.calculations.Calculations method*), 1

## P

plot\_predictions() (*stockDL.plots.Plots method*), 3  
 plot\_training\_data() (*stockDL.plots.Plots method*), 3  
 Plots (*class in stockDL.plots*), 3  
 plots (*stockDL.main.Main attribute*), 2

## R

result (*stockDL.main.Main attribute*), 2  
 result\_calculations() (*stockDL.results.Results method*), 3  
 Results (*class in stockDL.results*), 3  
 results (*stockDL.main.Main attribute*), 2

## S

separate\_ones() (*stockDL.calculations.Calculations method*), 1  
 stockDL  
     module, 4  
 stockDL.calculations  
     module, 1

stockDL.data  
    module, 1  
stockDL.main  
    module, 2  
stockDL.market  
    module, 2  
stockDL.models  
    module, 3  
stockDL.plots  
    module, 3  
stockDL.preprocessing  
    module, 3  
stockDL.results  
    module, 3  
stockDL.train  
    module, 4

## T

ticker (*stockDL.main.Main attribute*), 2  
train (*stockDL.main.Main attribute*), 2  
train\_model() (*stockDL.train.Training method*), 4  
Training (*class in stockDL.train*), 4

## V

v\_lstm (*stockDL.market.Market attribute*), 2  
v\_mix (*stockDL.market.Market attribute*), 2

## W

w\_lstm (*stockDL.market.Market attribute*), 2  
w\_mix (*stockDL.market.Market attribute*), 2