Play with 600519 data: TimeLine Assignment

Yu-Chieh Jack Ho

1/15 2021

Outlines

- Problem Formulation
- Feature Generation
- Model Development & Experiments
- Summary
- Future Work

Problem Formulation

- A regression problem
  - y (Target): nextMidpt
  - Available **X**: book and trade data
  - Objective: minimizing RMSE

- Approach the goal by
  - Selecting and generating useful features from raw book/trade data
    - Python + pandas + numpy
  - Developing models which extract useful info. from those features
    - Tensorflow + keras

Feature Generation

Please refer to **data_proc.ipynb** for details

Feature Generation

- Data exploring Preprocessing:
  - Data Exploring & cleaning (e.g., check and remove NaN)
  - TimeStamp ➔ DateTime for data splitting
  - Design pipelines which allows frequent feature change
- Normalization:
  - Scaler: **MiniMax** / Srandard scaler
  - Fit Scaler with only training set

Feature Generation

- Price features:
  - Raw features
    - ['Bid1', 'Bid2', 'Bid3', 'Bid4', 'Bid5’]
    - ['Ask1', 'Ask2', 'Ask3', 'Ask4', 'Ask5’]
    - Midpt
  - Statistical features
    - ['MicroPrice', 'Bid_Mean', 'Ask_Mean’]
  - Distance features
    - ['Spread1', 'Spread2', 'Spread3', 'Spread4', 'Spread5', 'SpreadMean’]
  - More targets (as possible regularizers)
    - next_Bid1, next_Ask1

Feature Generation

- Size features
  - Raw features
    - ['Bid1Size', 'Bid2Size', 'Bid3Size', 'Bid4Size', 'Bid5Size', 'Bid_Total_Size’]
    - ['Ask1Size', 'Ask2Size', 'Ask3Size', 'Ask4Size', 'Ask5Size', 'Ask_Total_Size’]
  - Distribution features: proportion of bid/ask size in each sample
    - ['Bid1SizeProp', 'Bid2SizeProp', 'Bid3SizeProp', 'Bid4SizeProp', 'Bid5SizeProp’]
    - ['Ask1SizeProp', 'Ask2SizeProp', 'Ask3SizeProp', 'Ask4SizeProp', 'Ask5SizeProp’]
  - Ratio features
    - Bid/Ask Ratio
      - ['BidAskRatio1', 'BidAskRatio2', 'BidAskRatio3', 'BidAskRatio4', 'BidAskRatio5', 'BidAskRatioTotal']
    - Queue Imbalance features
      - ['Q_ImB1', 'Q_ImB2', 'Q_ImB3', 'Q_ImB4', 'Q_ImB5']

Feature Generation

- More feature engineering
  - Moving average of selected features
  - Percentage Change of selected features

Model Development

- Baseline model
  - Linear Regression

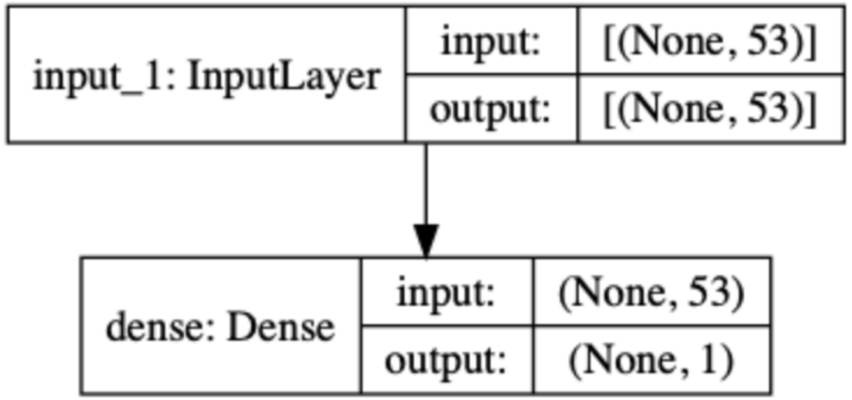
- DNN model
  - A MLP model which mimic the kernel trick

- 2-winged DNN model
  - A DNN model which extract the info. from size and price features independently

Experimental Setting

- Callback functions leveraged:
  - ReduceLROnPlateau: Monitored val_loss and reduce learning rate when the metric has stopped improving
  - EarlyStopping: Monitored val_loss and stop training when the metric has stopped improving.
  - ModelCheckpoint: Select and save the best model according to the val_loss during learning
- {'batch_size': 128, 'epochs': 300, 'optimizer': 'adam’, ‘loss’: mean_squared_error}
- The manual selection of features affect the RMSE slightly, but did not change the related accuracy of models. In the following slides, I will report their performance when training with all features

Baseline Model

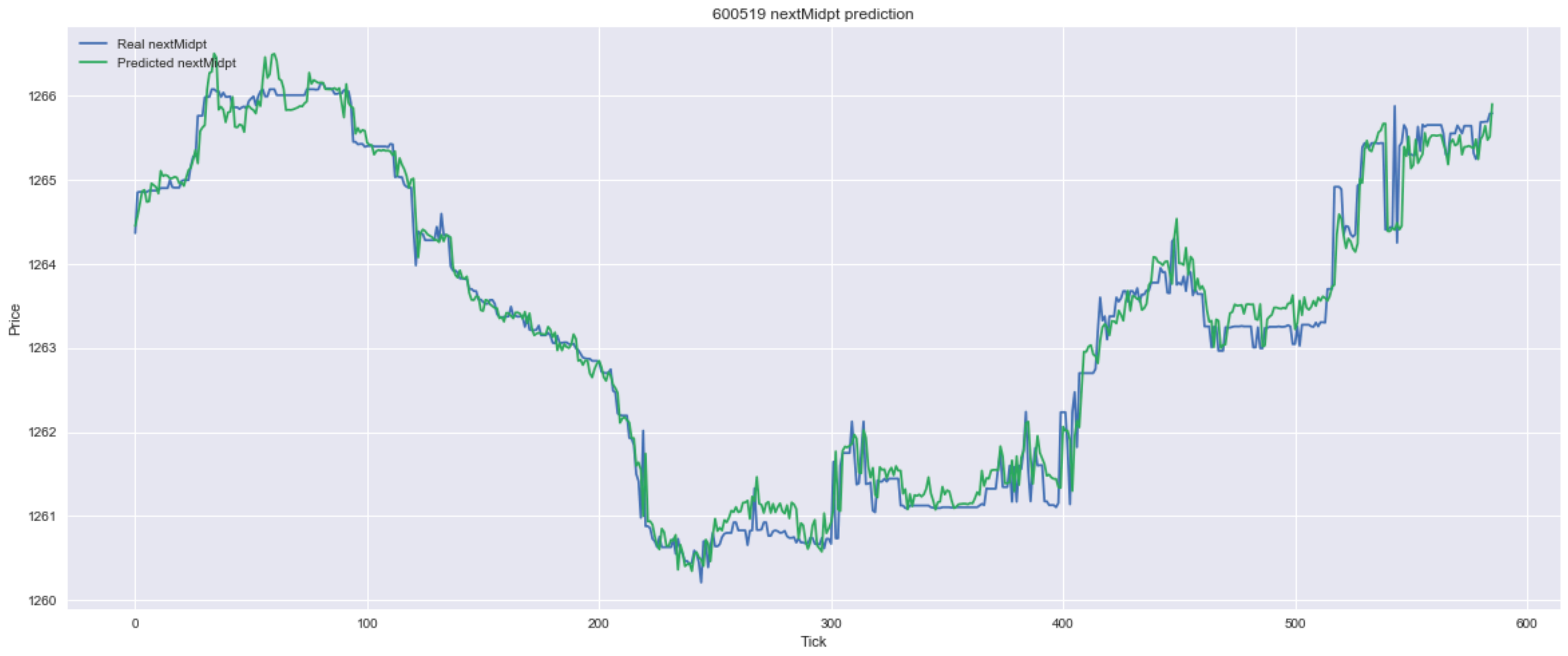


Please refer to **baseline_model.ipynb** for details

Baseline model + features without normalization

Evaluation metrics
Training Data - MSE: 0.0820, RMSE: 0.2863
Validation Data - MSE: 0.0801, RMSE: 0.2830

A baseline which is **better than expected**



Baseline model + features with normalization

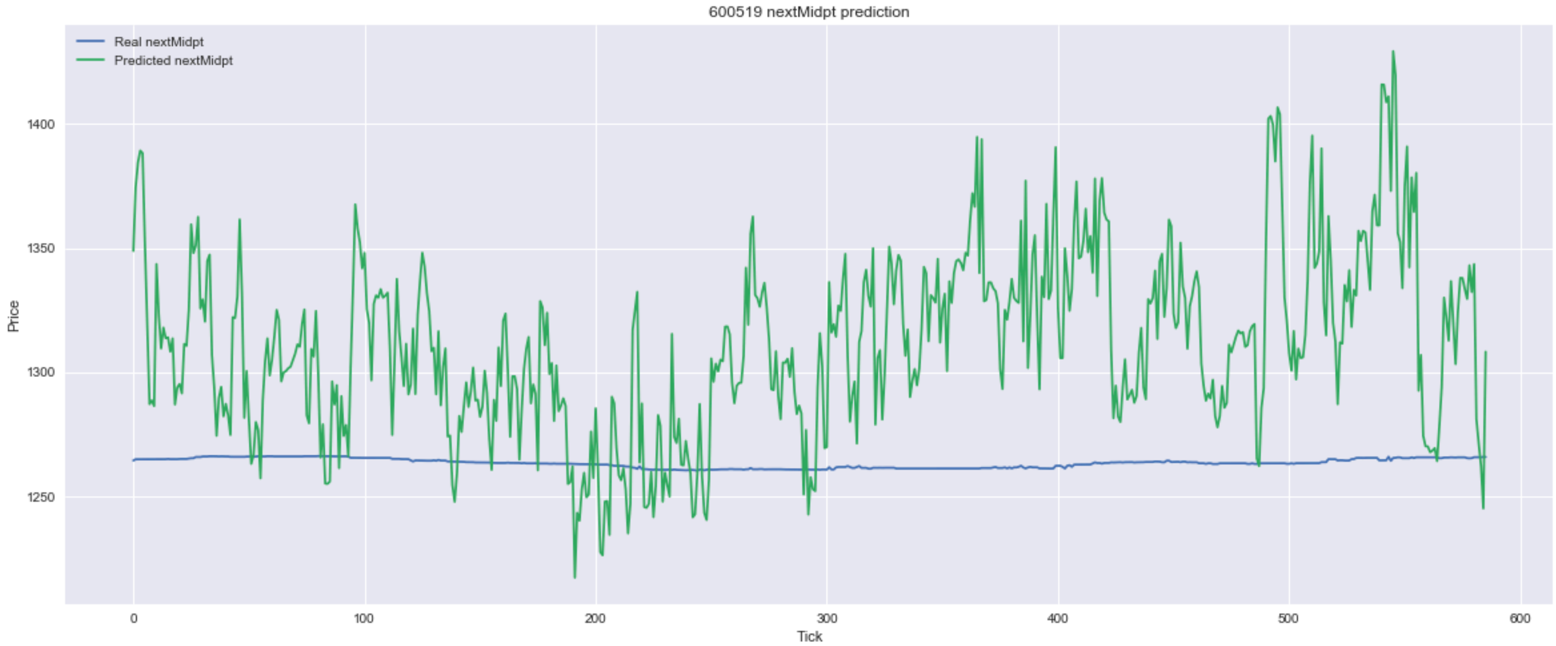
- However…

  Evaluation metrics

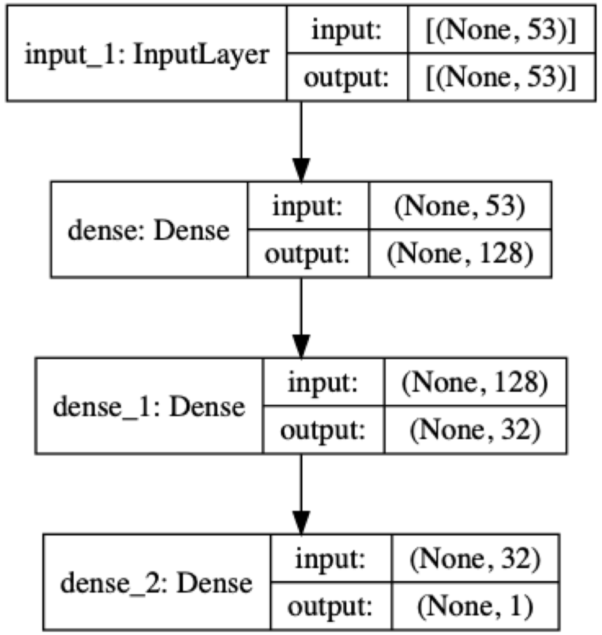
  Training Data –

  MSE: 274388.6250, RMSE: 523.8212

  Validation Data - MSE: 22248.3984, RMSE: 149.1590



Linear activation
Relu activation (nonlinear)



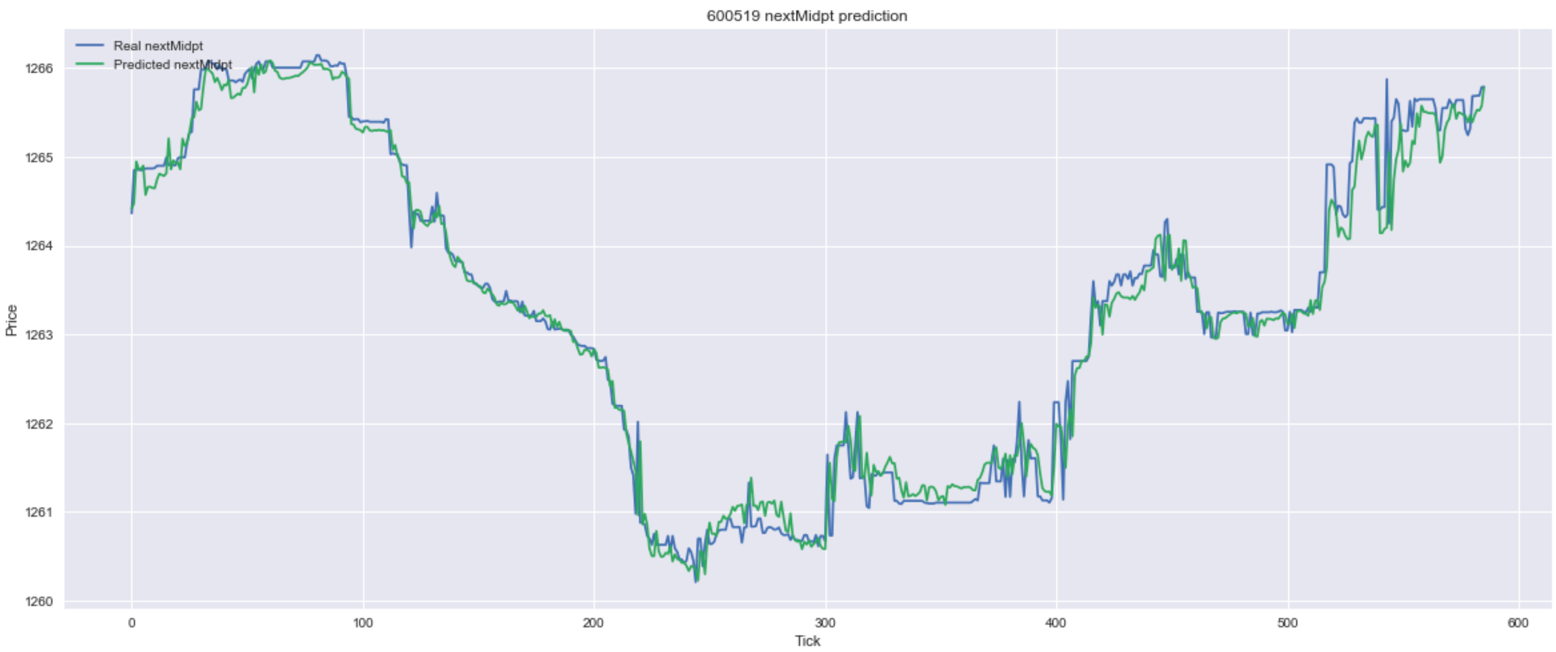
DNN Model

The model with has **best performance** so far

Please refer to **dnn_model.ipynb** for details

Linear DNN model + features without normalization

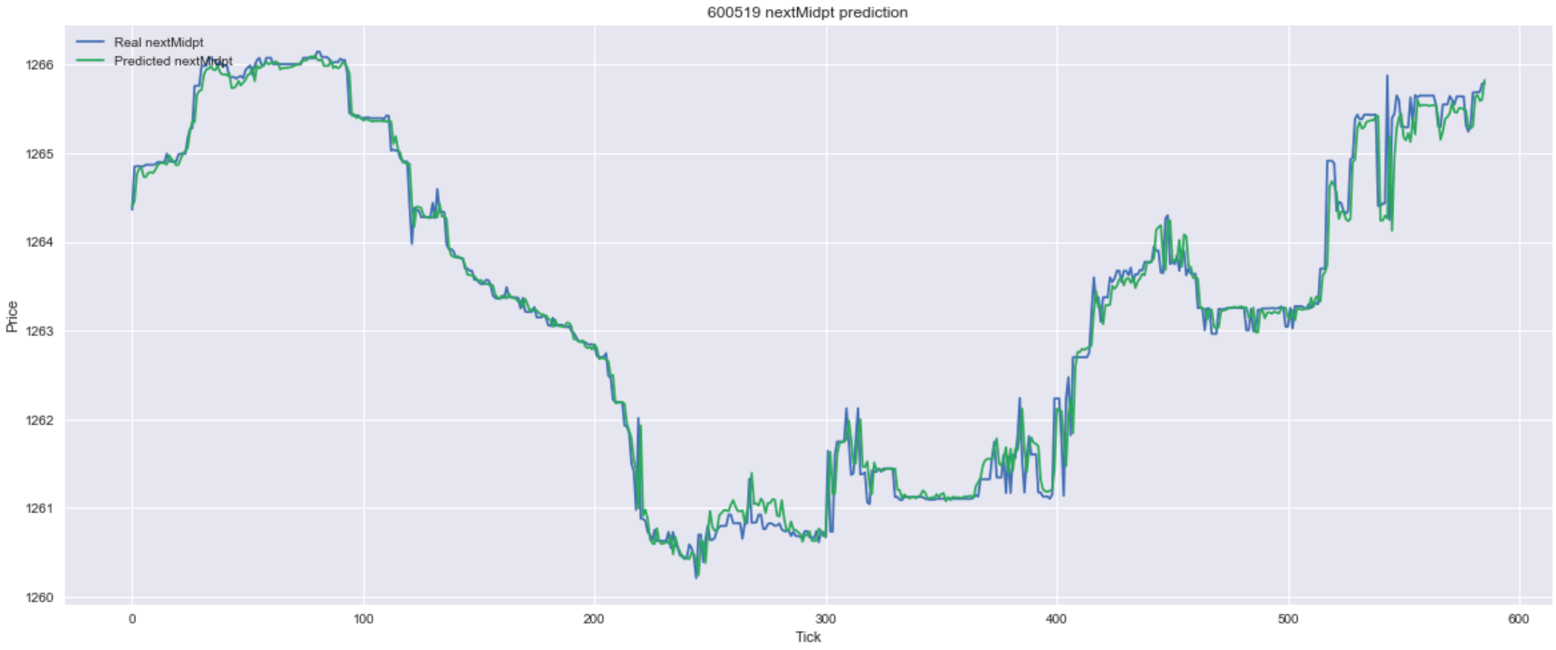
Evaluation metrics
Training Data - MSE: 0.0824, RMSE: 0.2871
Validation Data - MSE: 0.0768, RMSE: 0.2772



Linear DNN model + features with normalization

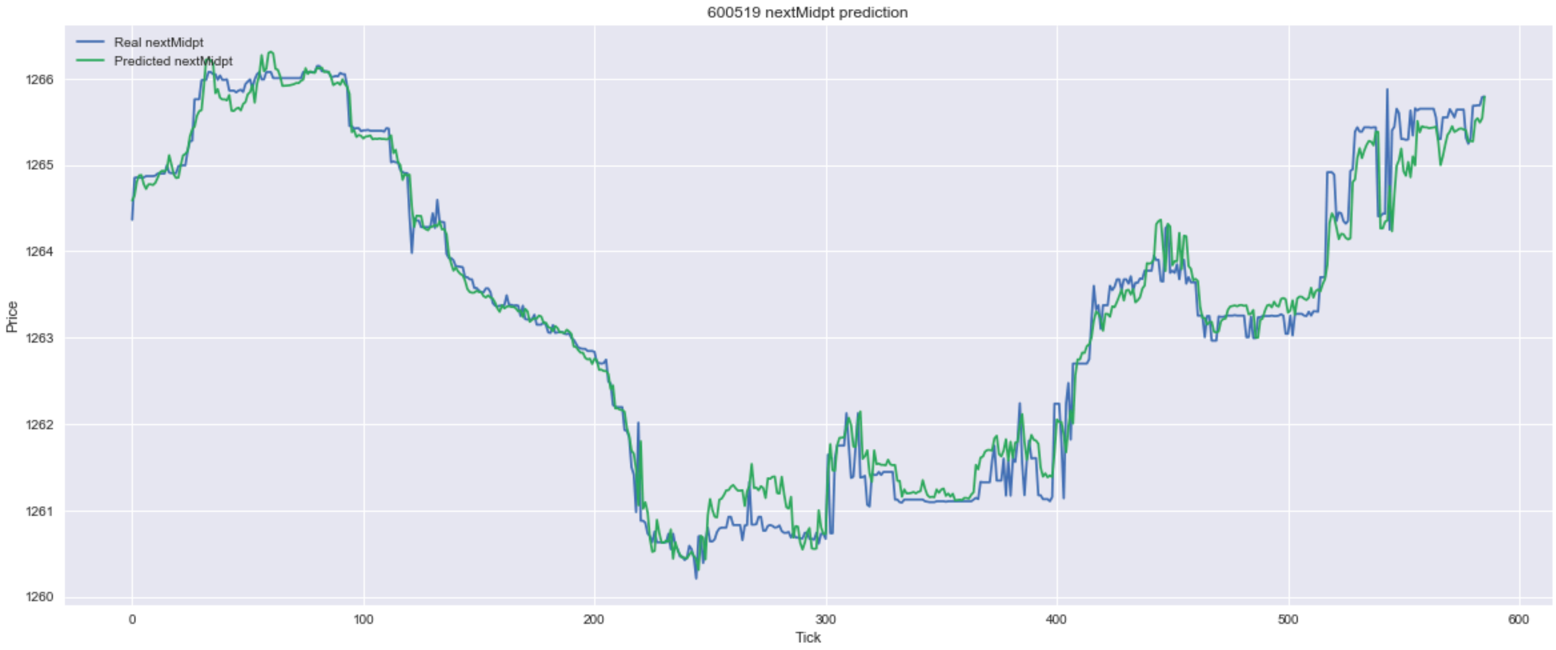
Evaluation metrics
Training Data - MSE: 0.0644, RMSE: 0.2538
Validation Data - MSE: 0.0628, RMSE: 0.2507

Best Performance so far



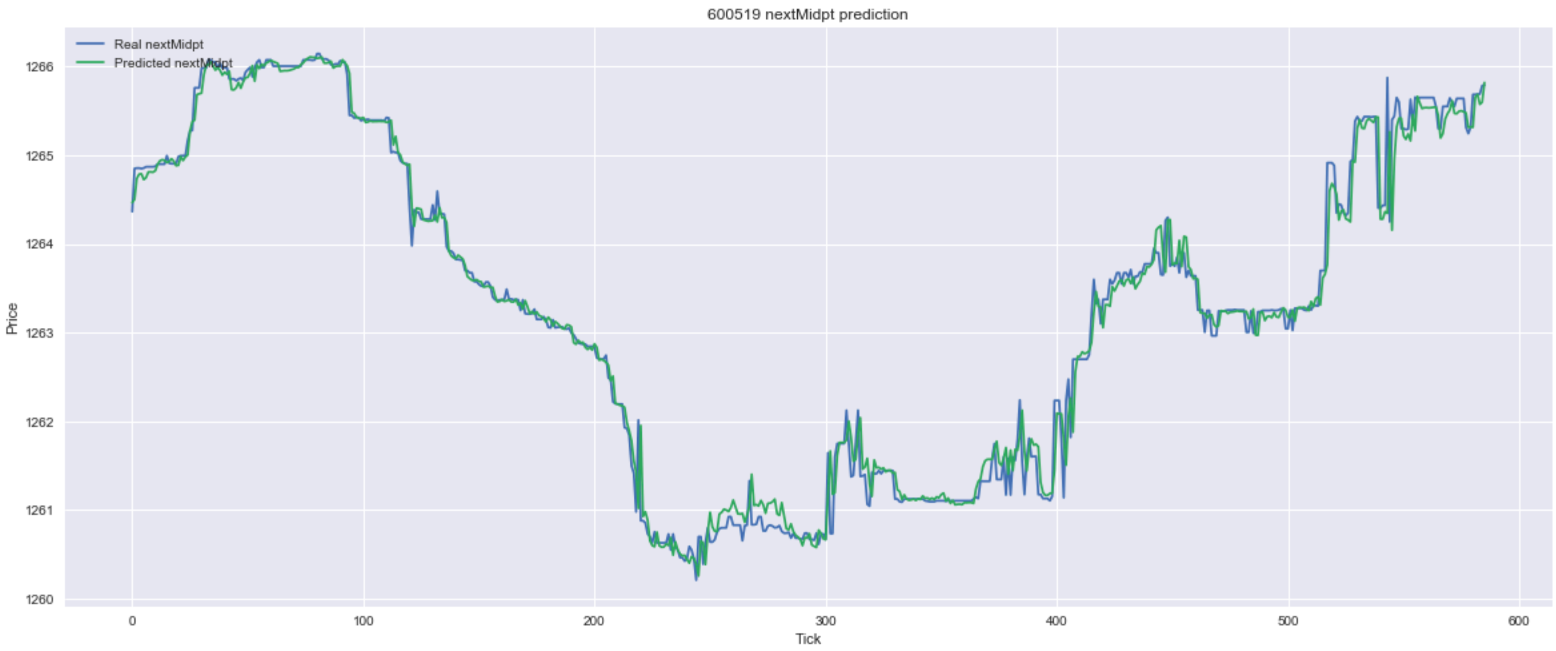
Nonlinear DNN model + features without normalization

Evaluation metrics
Training Data - MSE: 0.0792, RMSE: 0.2814
Validation Data - MSE: 0.0787, RMSE: 0.2805

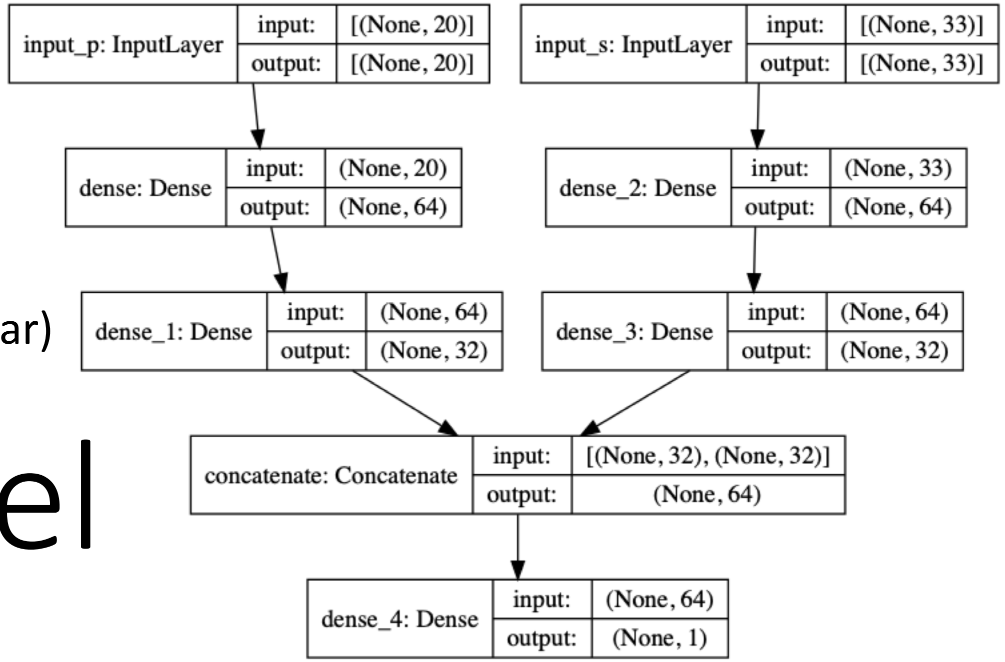


Nonlinear DNN model + features with normalization

Evaluation metrics
Training Data - MSE: 0.0653, RMSE: 0.2556
Validation Data - MSE: 0.0635, RMSE: 0.2520



Linear activation
Relu activation (nonlinear)



2-Winged DNN Model

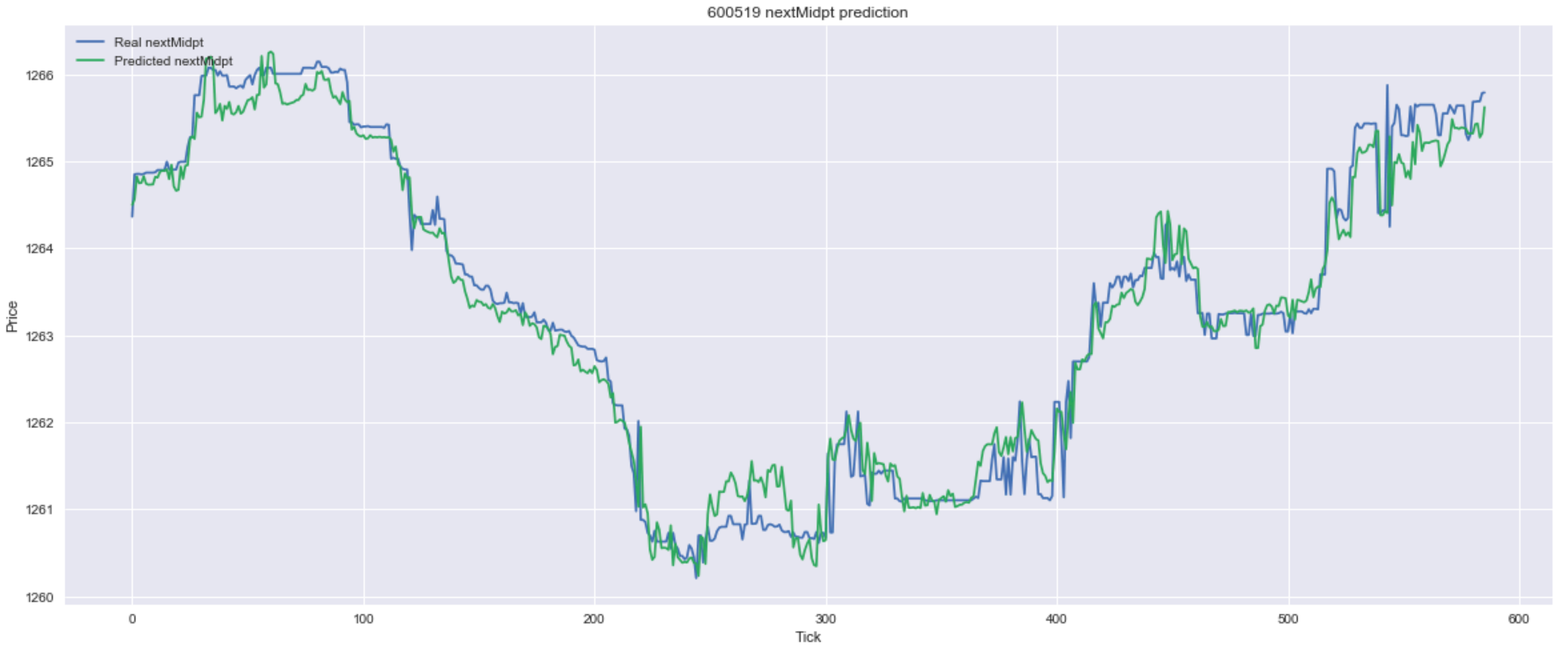
Separate the price and size features, extract information

Independently with 2 sub-networks, and concatenate the results for for final regression.

Please refer to **2w_dnn_model.ipynb** for details

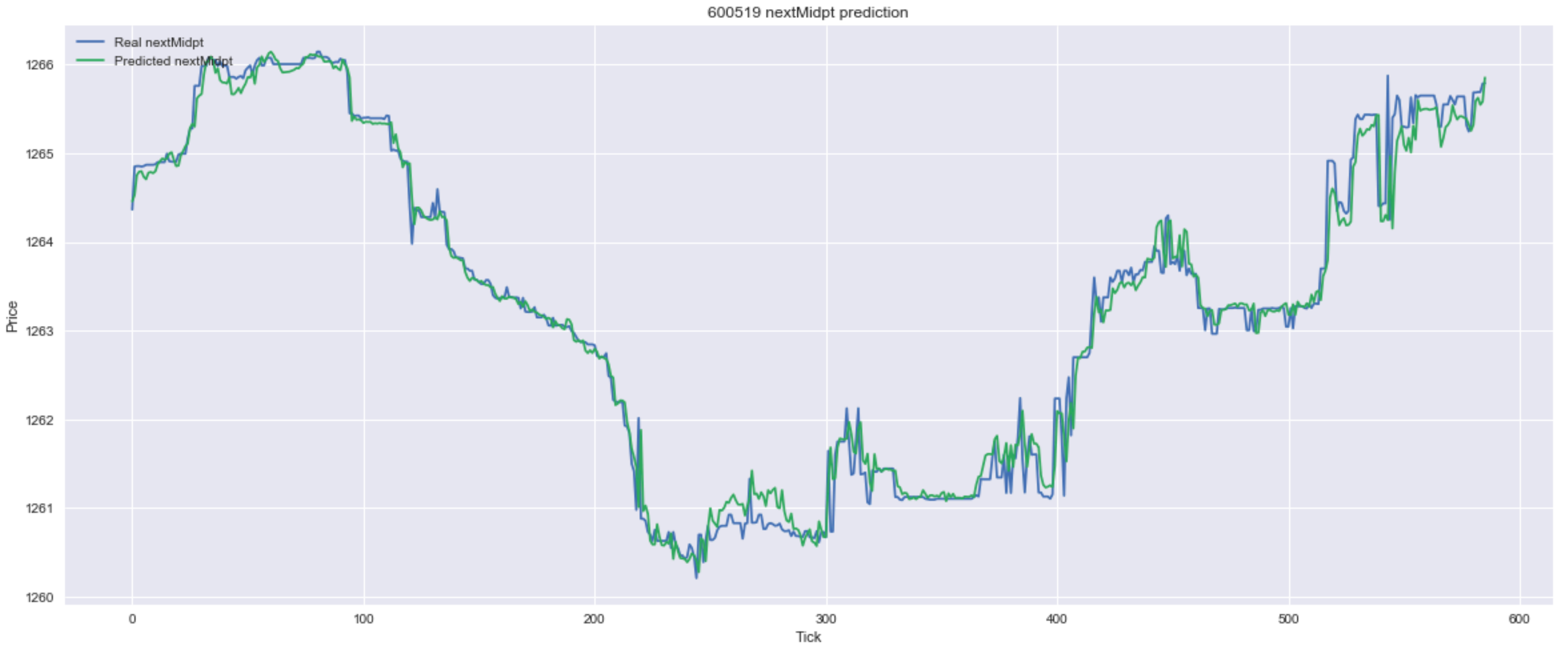
Linear 2W-DNN model + features without normalization

Evaluation metrics
Training Data - MSE: 0.1054, RMSE: 0.3247
Validation Data - MSE: 0.0999, RMSE: 0.3161



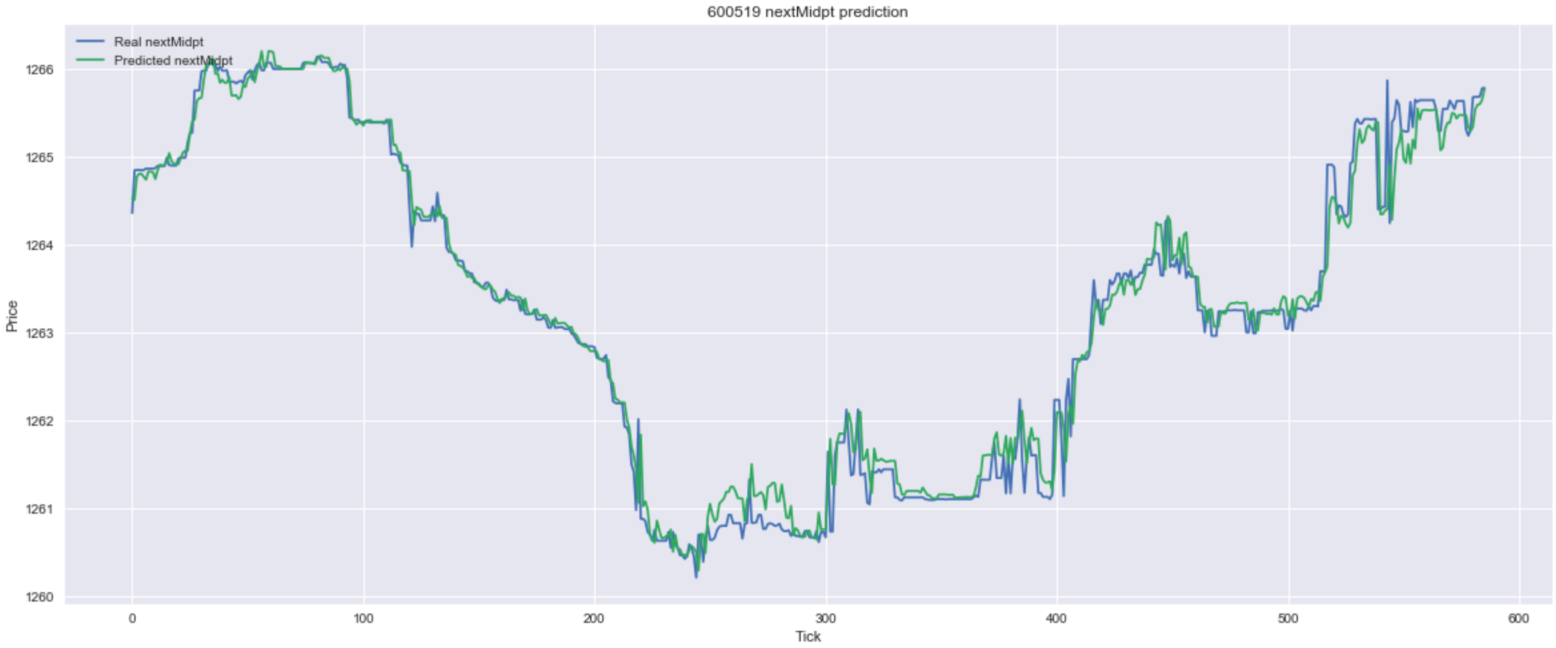
Linear 2W-DNN model + features with normalization

Evaluation metrics
Training Data - MSE: 0.0675, RMSE: 0.2598
Validation Data - MSE: 0.0664, RMSE: 0.2578



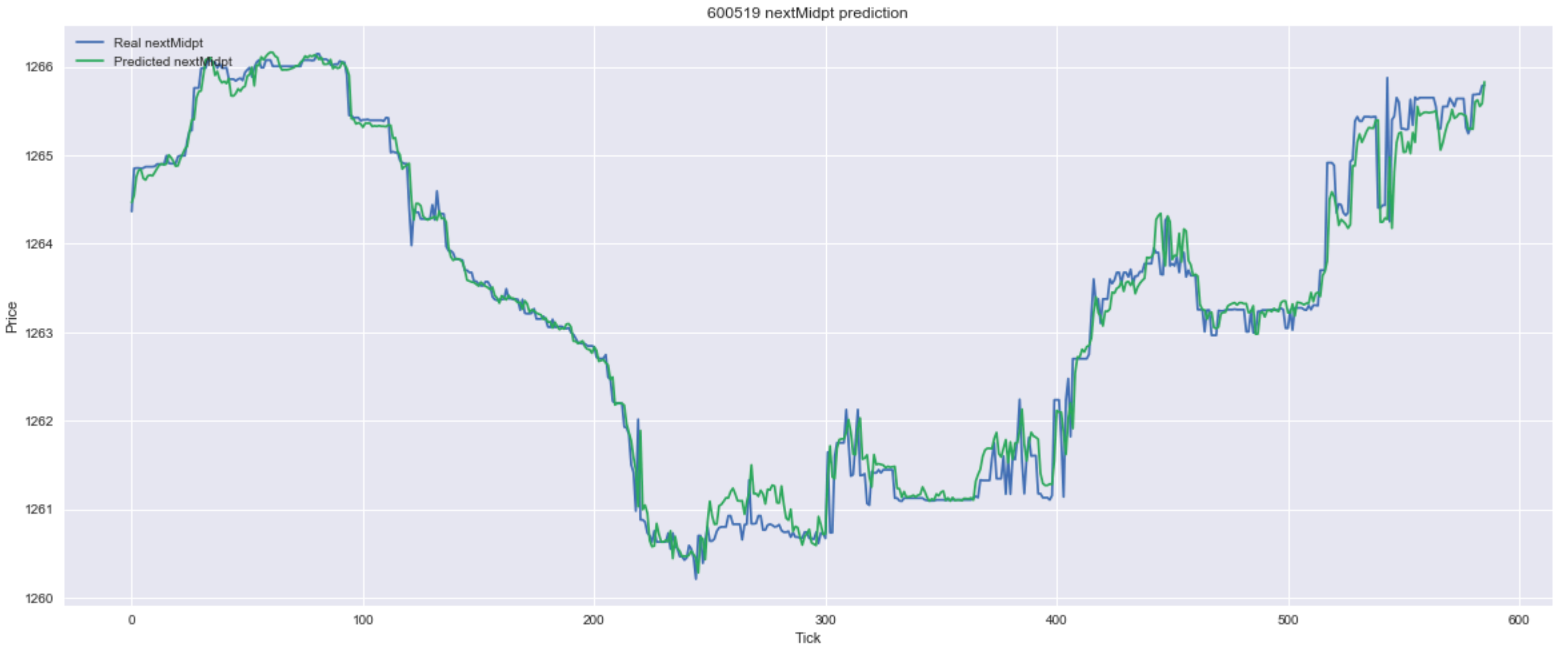
Nonlinear 2W-DNN model + features without normalization

Evaluation metrics
Training Data - MSE: 0.0711, RMSE: 0.2667
Validation Data - MSE: 0.0696, RMSE: 0.2638



Nonlinear 2W-DNN model + features with normalization

Evaluation metrics
Training Data - MSE: 0.0727, RMSE: 0.2696
Validation Data - MSE: 0.0683, RMSE: 0.2613



Summary

- The linear DNN model gave the best performance so far
  - **MSE: 0.0628, RMSE: 0.2507** on validation set.
- The performance of 2W-DNN gave similar performance (but did not surpass) with more training epochs (e.g., 1000)
- The performance maybe slightly improved by carefully hyperparameters tuning and feature selection.

Future Work

- It would be interesting to explore more **features, targets** and **model** structures
  - Features
    - Features from trade data
    - Indicators
  - Targets
    - Signals
  - Models
    - Attention model for time-series analysis

- Looking forward to sharing and learning with you!☺