Data Mining Course Project Task 1

Lu Yan

June 2019

1 Introduction

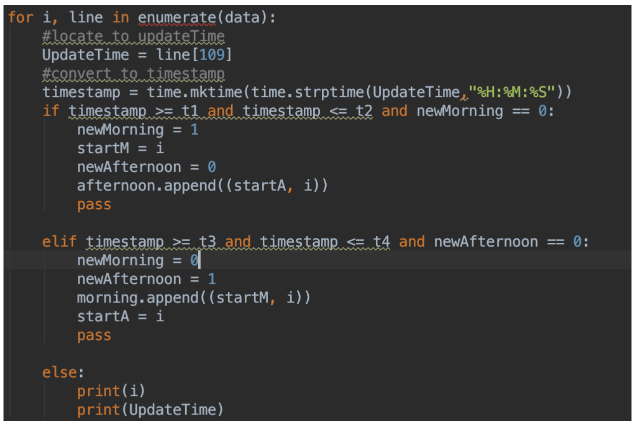
I finihsed the first task in the project. This task mainly contains three parts: preprocessing, model building and feature analysis. I conduct my experiments using python 3.6.3 within Anaconda virtual environment. This environment includes a long list of packages, to name the most important three, Sicikit Learn, Keras and Numpy. I tried both traditional Machine Learning methods and Deep Learning Methods to regress training data.

2 Preprocessing

2.1 Divide by Half Days

As we all know, stock prices are extremely sensitive to time. This is, prices change from day to day, even across half days. Thus, the first step I performed is *divide all the data into half days.* I obtained 44 afternoons and 45 mornings finally.

To explain in details, I use timestape to replace the original string-type time. Then I compare timestamp value of each data line with that of standard market opening and closing time (namely, 9:00am, 11:30am, 13:30pm, 3:00pm. I mask morning and afternoon using a coin-flipping way, as shown in Fig. 1.

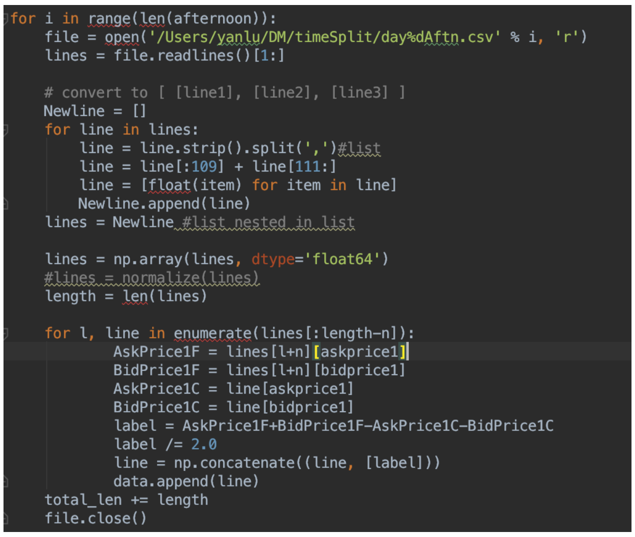


```python
for i, line in enumerate(data):
    #locate to updateTime
    UpdateTime = line[109]
    #convert to timestamp
    timestamp = time.mktime(time.strptime(UpdateTime,"%H:%M:%S"))
    if timestamp >= t1 and timestamp <= t2 and newMorning == 0:
        newMorning = 1
        startM = i
        newAfternoon = 0
        afternoon.append((startA, i))
        pass

    elif timestamp >= t3 and timestamp <= t4 and newAfternoon == 0:
        newMorning = 0
        newAfternoon = 1
        morning.append((startM, i))
        startA = i
        pass

    else:
        print(i)
        print(UpdateTime)
```

Figure 1: Code of Dividing by Half Days

2.2 Prepare Features and Labels

I noticed there are two columns of the original data is about time (Column 109 and 110). They are noise information and I remove them directly. Then I convert each features as numpy float32 type. After that, I compute labels as described in the PDF assignment using data from the future 10th line. Some lines in each half day data are omitted to guarantee validity. The details are shown in Fig. 2.



```
for i in range(len(afternoon)):
    file = open('/Users/yanlu/DM/timeSplit/day%dAftn.csv' % i, 'r')
    lines = file.readlines()[1:]

    # convert to [ [line1], [line2], [line3] ]
    Newline = []
    for line in lines:
        line = line.strip().split(',')#list
        line = line[:109] + line[111:]
        line = [float(item) for item in line]
        Newline.append(line)
    lines = Newline #list nested in list

    lines = np.array(lines, dtype='float64')
    #lines = normalize(lines)
    length = len(lines)

    for l, line in enumerate(lines[:length-n]):
            AskPrice1F = lines[l+n][askprice1]
            BidPrice1F = lines[l+n][bidprice1]
            AskPrice1C = line[askprice1]
            BidPrice1C = line[bidprice1]
            label = AskPrice1F+BidPrice1F-AskPrice1C-BidPrice1C
            label /= 2.0
            line = np.concatenate((line, [label]))
            data.append(line)
    total_len += length
    file.close()
```

Figure 2: Code of Assign Labels

3 Model Bulding

Before testing performance of different models, I normalize all features to range 0 to 1 and spilt training data and test data according to the ratio 0.7:0.3. Here are the results of different models: The detailed information of models are described below. To make the

Model	Test Loss
Random Forest	0.00021052356038247721
GBDT	0.0002146967208417868
AdaBoost	0.0002796147845284279
XgBoost	0.000212197086749754
DNN	0.0003950643704738468
CNN	0.00020575512558571062
LSTM	0.0003838463959882767
Stacked LSTM	0.00027102705006187876

Table 1: Regression Results of Models using MSE Loss

regression results more directly perceived through the senses, the Fig. 3 shows the mean predicted values of three models and values of each model.

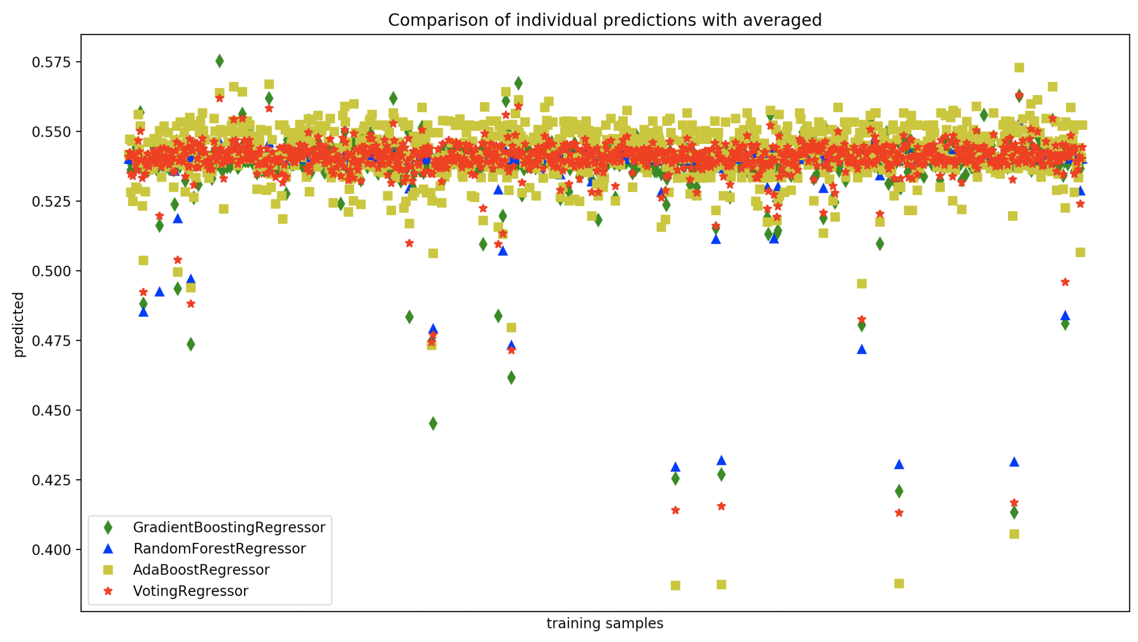


Figure 3: Regression Results Comparison

3.1 Random Forest

A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

In the experiment, I use a Random Forest regressor of max depth 2 and 100 estimators to learn the data. It achieves the loss 0.00021052356038247772.

Besides applying Random Forest to regress training data, I also explore the training effect variance with the size of training instances (as shown in Fig. 4). As the figure suggests, the training score (loss) increase with more training examples while the validation score (loss) decrease. This is possibly because model is able to generalize better with large training sets, although it may feel confused by the diverse instances.

3.2 GBDT

GB builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage a regression tree is fit on the negative gradient of the given loss function.

3.3 AdaBoost

An AdaBoost regressor is a meta-estimator that begins by fitting a regressor on the original dataset and then fits additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction. As such, subsequent regressors focus more on difficult cases.

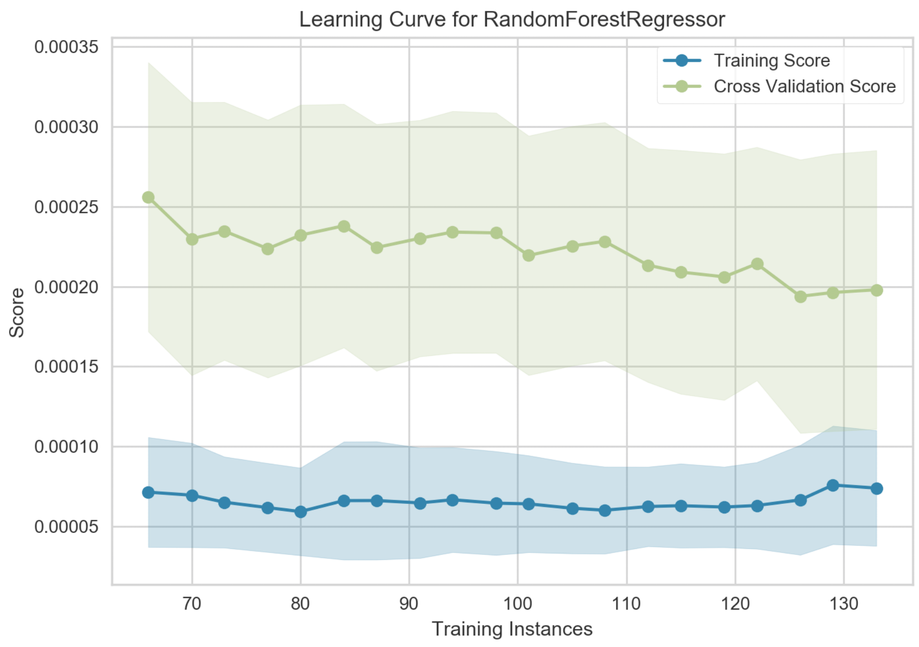


Figure 4: Train and Test performance Variance with Training Instances Size

3.4 XgBoost

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way. The same code runs on major distributed environment (Hadoop, SGE, MPI) and can solve problems beyond billions of examples.

3.5 DNN

DNN is one of the most widely used Deep Learning model. The only two kinds of layers it may contain are Dense Layer and Activation Layer. In the experiment I use ReLu for all the Activation Layers. The model visualization and training process is shown in Fig. 6(a) and Fig. 7(a) respectively.

3.6 CNN

CNN is mostly used in image processing and has extraordinary ability to extract local information. In the experiment, I treat each data line as one input to CNN. Of Course, I use a lot of padding. Its model visualization and training process is shown in Fig. 6(b) and Fig. 7(b), separately.

3.7 LSTM

LSTM is a variance of RNN, often used in time series problems. There are many variants of LSTM for univariate and multivariate problems. In the experiment I tried

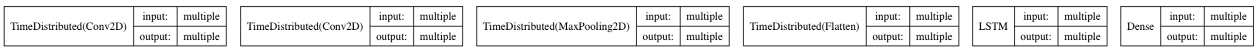
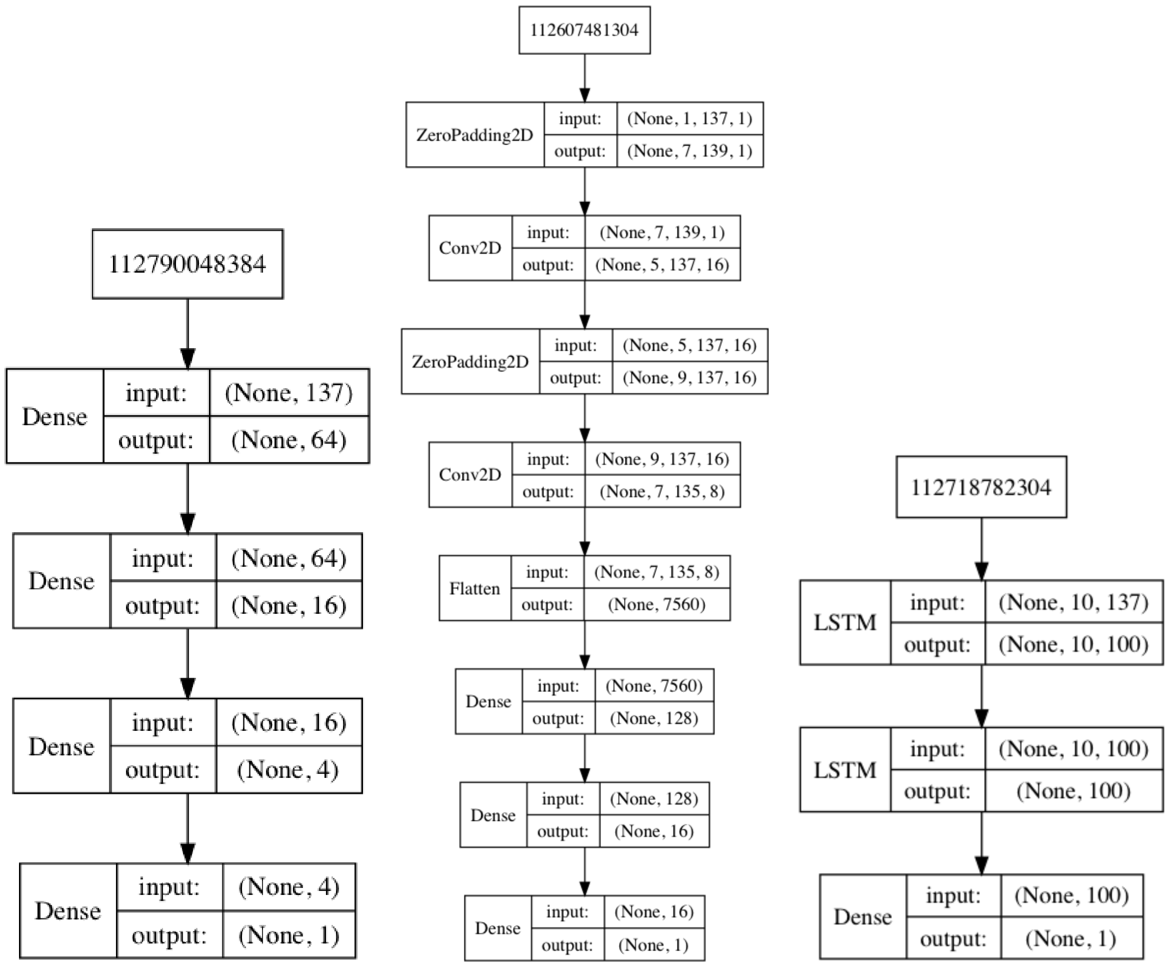


Figure 5: CNN LSTM Model

Vanilla LSTM, Stacked LSTM and CNN LSTM. Vanilla LSTM only achieves the loss of 0.0003838463959882767, larger than any other models. Thus, I tried more sophisticated model Stacked LSTM and it performs much better. I also tried CNN LSTM, but encounter some difficulties due to reference absence and time limitation. The test model is shon in Fig. 5.

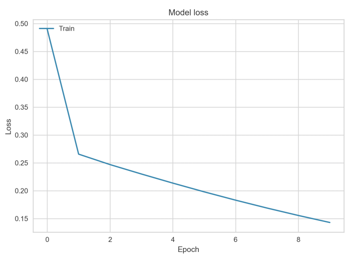


(a) a (b) b (c) c

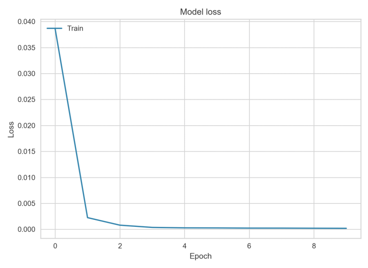
Figure 6: Model Visualization of DNN, CNN and Stacked LSTM

3.8 Result Analysis

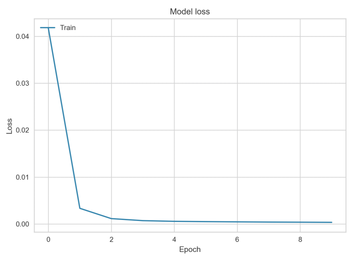
From Fig. 1, we can see the training method differences of three Deep Learning models. CNN has the best initial state and the best converging speed. DNN converges the slowest and the worst smoothness. In general, CNN achieves the best performance and ensemble methods are also excellent. This is probably due to the high linear relationship between features, as described in the last section.



(a) a



(b) b

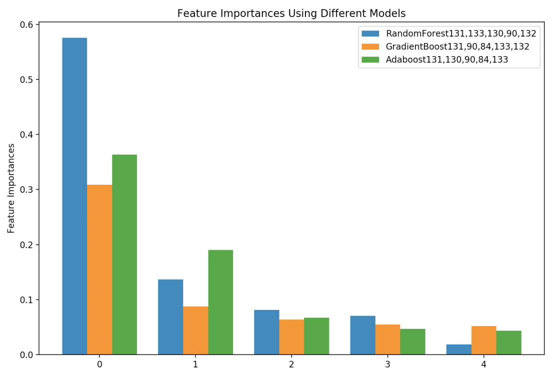


(c) c

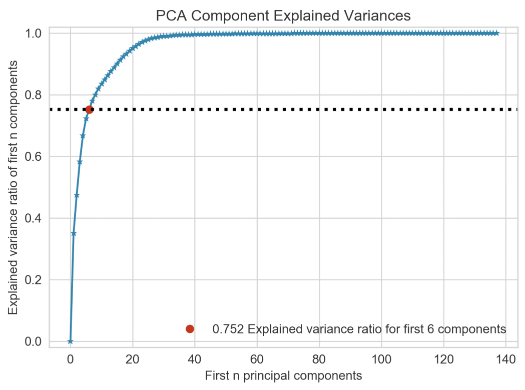
Figure 7: Training Process of DNN, CNN and Stacked LSTM, trained using batch size 50 and 10 epochs

4 Feature Analysis

In the project setting, there are 137 features for each sample. It is worthwhile to find out which features matter most. I tested the feature importances when putting trianing samples to each model and picked up the 5 most important ones, as suggested by Fig. 8 The figure



(a) Feature Importances of Each Model



(b) Explained Variance Ratios of the First N Features

Figure 8: Feature Analysis

shows the 131st feature, BidVolumn1 is of the most significance. The 90th (Indicator91), 133th (BidVolumn3) and the 84th (Indicator85) are other commonly thought important ones. Furthermore, it is also obvious that Random Forest produces most sparse features of the three ensemble models (with AdaBoost the following and GBDT the last).

From the figure, we can also conclude that the first five features play extremely important roles, which raise the question about what the relationship between them and other features? I apply PCA visualization to solve this problem.

The PCA results validates the conclusion above. 6 components can explain 75.2% of total information and about 40 features cover 100%.