HW6_NN_stock_predictions

March 2, 2020

1 Adv Big Data HW 6 Summary of article using NN in forcasting stock prices

1.0.1 Article: "Stock Buy/Sell Prediction Using Convolutional Neural Network" by: Asutosh Nayak

1.0.2 source: https://towardsdatascience.com/stock-market-action-prediction-with-convnet-8689238feae3

1.0.3 github for article: https://github.com/DarkKnight1991/stock_cnn_blog_pub/

1.0.4 Article is based on a paper called "Algorithmic Financial Trading with Deep Convolutional Neural Networks: Time Series to Image Conversion Approach"

1.0.5 source: https://www.researchgate.net/publication/324802031_Algorithmic_Financial_Trac

1.1 I found the article and research paper very interesting. The article is great at summarizing the paper and providing examples but there are certain aspects that I don't fully understand.

1.1.1 From the article:

1.2 1) What does the paper say? –> Researchers took a time series problem and turned it into an image classification problem which is really awesome.

- first they calculated 15 different technical indicators over 15 different periods for each day in the training data
- next, convert the 225 (15 x 15) features into 15 x 15 images
- then, label the data as "buy", "sell" or "hold" based on an algorithm the researchers provided in the paper
- finally, they trained a CNN like any other image classification problem

1.3 ** Feature Engineering ** was mostly comprised of calculating a range of technical indicators such as Simple Moving Averages

ex: Below is an image from the article that show the calculation of a 6 day simple moving average, from there the researchers would calculate another 14 SMA's so they would end up with 6, 7, 8, …, 20 day SMA's

47.56	
48.31	
48.94	
47.94	
47.69	
47.75	48.03167
51.5	48.68833
54.63	49.74167
55.75	50.87667
55.13	52.075
56.63	53.565
55.38	54.83667
54	55.25333
55.5	55.39833
55.44	55.34667
54.5	55.24167
58.75	55.595
59	56.19833
56.5	56.615
61.19	57.56333

This approach of a sliding window is used to create as many technical indicators as they need, but its also how they would train and test the CNN

After this feature engineering we should have 225 new features and if you reshape them into a 15 x 15 array we get an image.

It is important to note that we need to keep related technical indicators spatially close –> Related pixels should be close to each other. For me the worst part or most disappointing part of the paper was how they labeled the dataset as Buy, Sell, Hold basically, they **created an 11 day window using closing price and if the middle number is maximum = label the last day "sell", if its the window minimum = "buy", else = "hold" –> The basic idea is to identify troughs to buy at and crests to sell at in any 11 day window** As much as I liked this paper and article, I would love to look into applying the analysis using a different strategy for labeling Buy, Sell, or Hold. Even if I dont find a better set of rules to label I hope I am able to find a better explanation for why this was chosen

1.3.1 Training

The authors used a rolling window similar to the sliding window used above to calculate the technical indicators. The paper uses data from 2000 to 2019 and would train on 5 years of data from 2000-2004, then test on 1 year 2005, from there use the model and retrain on years 2001-2005 and test on 2006.

The article mentioned the paper had some points about the model architecture that were missing and caused issues in reproducing the results from the paper. The paper didn't mention the strides and padding used and the article author couldn't get the sliding window to work. The model was just to large so for the article he used the full training data with cross validation.

The Keras model was also trainded using Early Stopping and REduceLROnPlateau. These are parameters I'm not too familiar with so I will need to do more research on what they do.

1.3.2 Evaluation

The paper uses 2 types of model evaluation 1. Computational -> Confusion matrix, F1 Score, and class wise precision 2. Financial -> Back testing calls and measuring Profits and Losses vs the market as a whole 3. From the article, the author added the use of teh Kappa statistic which I believe compares an Observed Accuracy with an Expected Accuracy (random chance) but I'm not familiar with this metric.

1.4 2) Implementation

The article mentions that the paper didn't produce expected results and left out some of the methodology so the author made some minor changes.

Data for the project was gathered from Alph Vantage: https://www.alphavantage.co/

[28]:
```
import pandas as pd

# use alph vantage to get stock data for Microsoft, Google and Amazon
# url/api calls for each stock
msft_url = "https://www.alphavantage.co/query?
↪function=TIME_SERIES_DAILY_ADJUSTED&outputsize=full&symbol=MSFT&apikey=58C7PZSCC43Z8L1S&dat
googl_url = "https://www.alphavantage.co/query?
↪function=TIME_SERIES_DAILY_ADJUSTED&outputsize=full&symbol=GOOGL&apikey=58C7PZSCC43Z8L1S&da
amzn_url = "https://www.alphavantage.co/query?
↪function=TIME_SERIES_DAILY_ADJUSTED&outputsize=full&symbol=AMZN&apikey=58C7PZSCC43Z8L1S&dat
```

```python
# read the csv's into pandas DataFrames
adj_daily_msft = pd.read_csv(msft_url)
adj_daily_googl = pd.read_csv(googl_url)
adj_daily_amzn = pd.read_csv(amzn_url)
```

[25]:
```python
# print out the shapes of the the three df's
print(adj_daily_msft.shape, adj_daily_googl.shape, adj_daily_amzn.shape)
```

```
(5032, 9) (3910, 9) (5032, 9)
```

[26]:
```python
# look at the first 5 rows of one of the df's
adj_daily_amzn.head()
```

[26]:
```
    timestamp     open     high      low    close  adjusted_close   volume  \
0  2020-03-02  1906.49  1954.51  1870.00  1953.95         1953.95  6712446
1  2020-02-28  1814.63  1889.76  1811.13  1883.75         1883.75  9493797
2  2020-02-27  1934.38  1975.00  1882.76  1884.30         1884.30  8143993
3  2020-02-26  1970.28  2014.67  1960.45  1979.59         1979.59  5240402
4  2020-02-25  2026.42  2034.60  1958.42  1972.74         1972.74  6219094

   dividend_amount  split_coefficient
0              0.0                1.0
1              0.0                1.0
2              0.0                1.0
3              0.0                1.0
4              0.0                1.0
```

[27]:
```python
# look at the last five rows of one of the df's
adj_daily_googl.tail()
```

[27]:
```
       timestamp    open    high     low    close  adjusted_close    volume  \
3905  2004-08-25  104.76  108.00  103.88  106.000         53.1641   9188600
3906  2004-08-24  111.24  111.60  103.57  104.870         52.5974  15247300
3907  2004-08-23  110.76  113.48  109.05  109.400         54.8694  18256100
3908  2004-08-20  101.01  109.08  100.50  108.310         54.3227  22834300
3909  2004-08-19  100.01  104.06   95.96  100.335         50.3228  44659000

      dividend_amount  split_coefficient
3905              0.0                1.0
3906              0.0                1.0
3907              0.0                1.0
3908              0.0                1.0
3909              0.0                1.0
```

1.5 Feature Engineering

Here is where the author deviated a bit from the paper, to avoid making calculation errors the author looked for library or implementations for all of the indicators used in the paper but couldn't find some of them. Also some of the indicators like WMA, HMA, etc were very slow so saving the data after running it would be needed which means the analysis can't be used live. The paper also used an adjust ration to adjust the prices but there didnt seem to be a reference on how the adjustment was calculated.

below is a screen shot from the article showing the technical indicators that were used

```python
# this function adds the new features to the dataframe passed as parameter
def calculate_technical_indicators(self, df, col_name, intervals):
        # get_RSI(df, col_name, intervals)  # faster but non-smoothed RSI
        get_RSI_smooth(df, col_name, intervals)  # momentum
        get_williamR(df, col_name, intervals)  # momentum
        get_mfi(df, intervals)  # momentum
        # get_MACD(df, col_name, intervals)  # momentum, ready to use +3
        # get_PPO(df, col_name, intervals)  # momentum, ready to use +1
        get_ROC(df, col_name, intervals)  # momentum
        get_CMF(df, col_name, intervals)  # momentum, volume EMA
        get_CMO(df, col_name, intervals)  # momentum
        get_SMA(df, col_name, intervals)
        get_SMA(df, 'open', intervals)
        get_EMA(df, col_name, intervals)
        get_WMA(df, col_name, intervals)
        get_HMA(df, col_name, intervals)
        get_TRIX(df, col_name, intervals)  # trend
        get_CCI(df, col_name, intervals)  # trend
        get_DPO(df, col_name, intervals)  # Trend oscillator
        get_kst(df, col_name, intervals)  # Trend
        get_DMI(df, col_name, intervals)  # trend
        get_BB_MAV(df, col_name, intervals)  # volatility
        # get_PSI(df, col_name, intervals)  # can't find formula
        get_force_index(df, intervals)  # volume
        get_kdjk_rsv(df, intervals)  # ready to use, +2*len(intervals), 2 rows
        get_EOM(df, col_name, intervals)  # volume momentum
        get_volume_delta(df)  # volume +1
        get_IBR(df)  # ready to use +1
```

create_features.py hosted with ❤ by GitHub view raw

1.6 Labeling data

Again the algorithm used in the article is the same as the one used in the paper

below is how it was implemented in the paper

Algorithm 1 Labelling Method

```
procedure LABELLING()
    windowSize = 11 days
    while(counterRow < numberOfDaysInFile)
        counterRow++
        If (counterRow > windowSize)
            windowBeginIndex = counterRow − windowSize
            windowEndIndex = windowBeginIndex + windowSize − 1
            windowMiddleIndex = (windowBeginIndex + windowEndIndex)/2
            for (i = windowBeginIndex; i <= windowEndIndex; i++)
                number = closePriceList.get(i)
                if(number < min)
                  min = number
                  minIndex = closePriceList.indexOf(min)
                if(number > max)
                  max = number
                  maxIndex = closePriceList.indexOf(max)
            if(maxIndex == windowMiddleIndex)
                result = "SELL"
            elif(minIndex == windowMiddleIndex)
                result = "BUY"
            else
                result = "HOLD"
```

1.7 After the feature engineering

Our data frame will look like this

	timestamp	open	high	low	close	adjusted_close	volume	rsi_6	rsi_7	rsi_8	...	eom_19	eom_20	eom_21	eom_22	eom_23
0	2000-03-01	102.00	105.50	100.06	100.25	67.2395	10807800	10.769231	8.571429	12.876254	...	-23.656988	-23.656988	-23.656988	-23.656988	-23.656988
1	2000-03-02	100.50	105.44	99.50	103.12	69.1645	11192900	29.354583	26.695174	21.606529	...	-16.451501	-16.451501	-16.451501	-16.451501	-16.451501
2	2000-03-03	107.25	110.00	106.06	108.00	72.4376	10162800	50.545209	45.207481	42.141929	...	215.554768	215.554768	215.554768	215.554768	215.554768
3	2000-03-06	109.94	111.00	101.00	103.06	69.1243	10747400	32.388833	37.235432	34.363373	...	-188.882893	-188.882893	-188.882893	-188.882893	-188.882893
4	2000-03-07	106.00	107.00	101.69	103.00	69.0840	10035100	37.782761	33.681008	38.370550	...	-87.573118	-87.573118	-87.573118	-87.573118	-87.573118

5 rows × 450 columns

1.8 Normalization

The article uses MinMaxScaler from Sklearn to normalize the data between [0, 1] the paper used a range between [-1, 1].

I'm not sure of the difference and need to look into this more.

1.9 Feature Selection

From above we can see the type of indicators that were used, from there they were grouped together (momentum, oscillator, etc.)

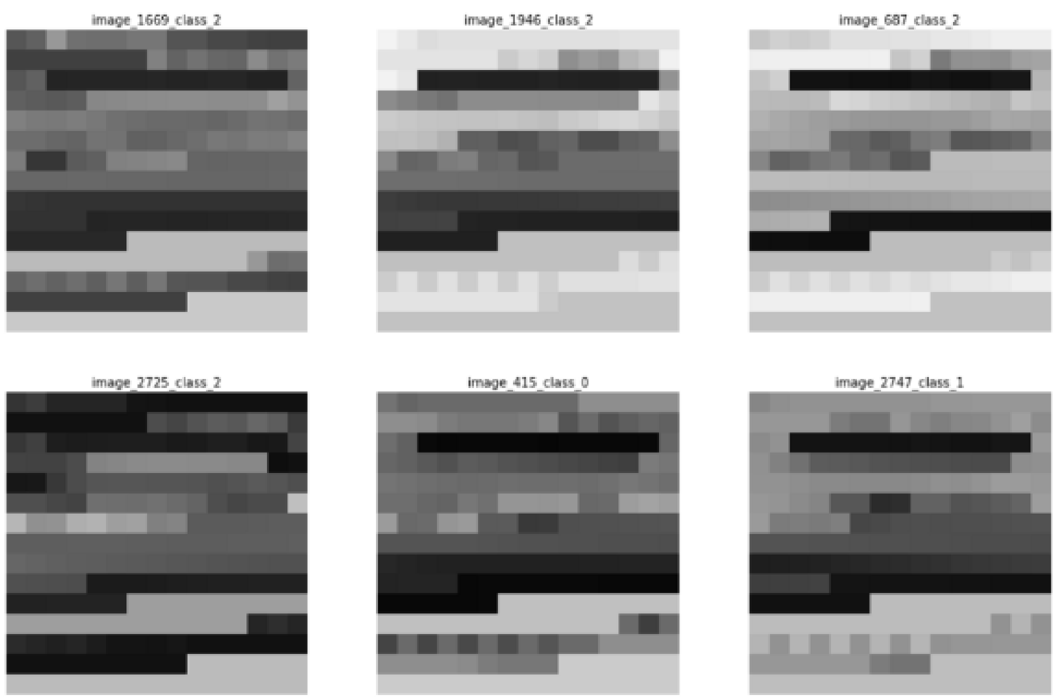
The author in the article trained many different CNNs and decided the features were not good enough, so he added many more then used some feature selection methods like f_classif and mutual_info_classif from Sklearn chosing high quality features that were common to both.

1.10 Reshape data into images

After using feature selection we have tabular data with 225 features and can convert it to images. Below is a snipet of code from the article and how the images look.

```
dim = int(np.sqrt(num_features))
x_train = reshape_as_image(x_train, dim, dim)
x_cv = reshape_as_image(x_cv, dim, dim)
x_test = reshape_as_image(x_test, dim, dim)
# adding a 1-dim for channels (3)
x_train = np.stack((x_train,) * 3, axis=-1)
x_test = np.stack((x_test,) * 3, axis=-1)
x_cv = np.stack((x_cv,) * 3, axis=-1)
print("final shape of x, y train/test {} {} {} {}".format(x_train.shape, y_train.shape, x
```

reshape_as_image.py hosted with ❤ by GitHub view raw



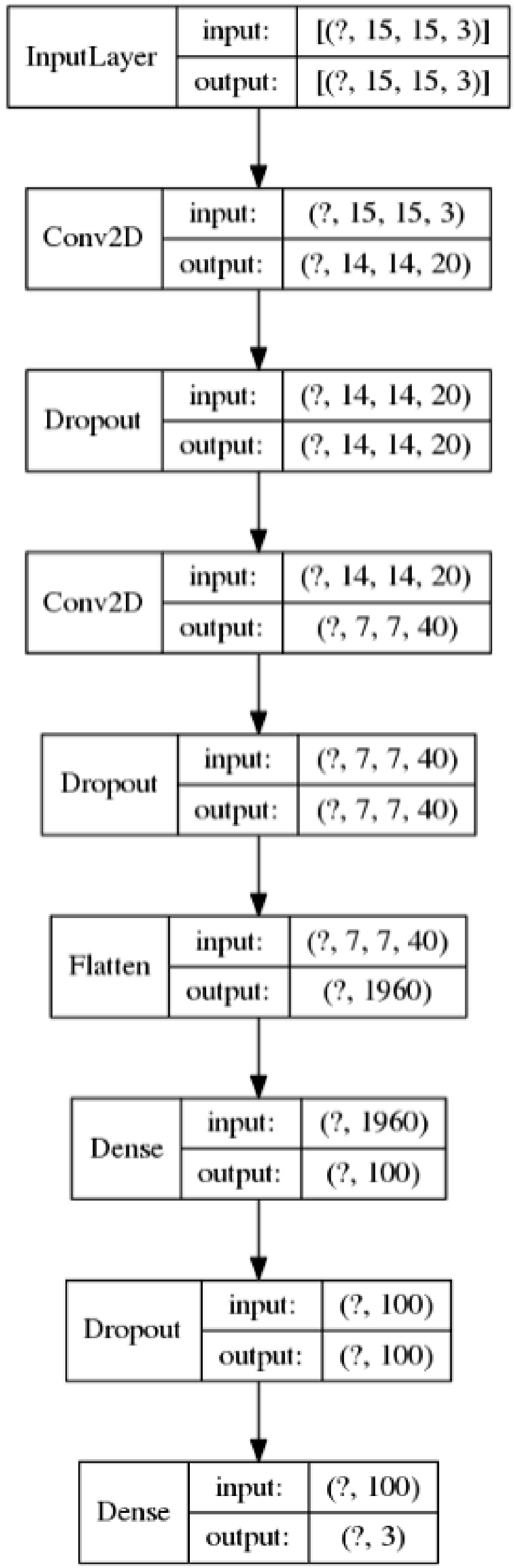
1.11 Biggest Issue

The biggest issue now is the class imbalance that we have. Hold is by far the magority class with few Buy and Sell labels. The paper mentions "resampling" but dosent provide details on how so the article author tried oversampling and SMOTE before settling on "sample weights"

1.12 Training

Methodology in paper had missing parts and the data proved to large for the article author so he didnt use the sliding window method for training and testing.

below is the best CNN configuration he found



1.13 Results

From the article here are the results for Walmart

```
baseline acc: 87.5
[[ 57   0   7]
 [  0  54   7]
 [ 53  55 767]]
F1 score (weighted) 0.8912214152727558
F1 score (macro) 0.7389316779518738
F1 score (micro) 0.878
cohen's Kappa 0.5972746718778883
precision of class 0 = 0.89
precision of class 1 = 0.89
precision of class 2 = 0.88
precision avg 0.8866666666666667
```

The model seems to be able to identify buy/sell instances but as the paper notes, lots of false entry and exit points are generated. This could be due to the imbalanced classes, in order to catch the Buy and Sell classes there is a tradeoff where flase buy/sells are generated.

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: