Stock movement:

Predict a 5% rise in 10 days with GAN

--758B Group Project,

Zheming Chen, Chengqi Wu, Yan Yan

Abstract

Stock price prediction is an important issue in the financial world, as it contributes to the development of effective strategies for stock exchange transactions. In this report, we propose a Generative adversarial network (GAN) framework employing fully connected neural network as generator and convolutional neural network (CNN) as discriminator, use the last layer of CNN as features to forecast price movements of stocks in IT sectors. We built a pipeline on Google Drive to download 72 IT sector stocks from Alpha Vantage API, held out recent 255 trade days (average trade day per year) as test data, take the rest as training data for each IT sector stock, trained the GAN network with 50k iterations. In each batch training, we used normalized 20 historical days of stock’s data as input, predicted its movement direction in 10 days with another CNN classifier. In the testing set, the baseline XGBoost model’s accuracy is 61.70% with a TPR of 53.0%, and the CNN model trained with the balanced data outperforms the other models with 85.45% overall accuracy and an 81.30% TPR. Future work will concentrate on how to make this model to be more economic useful.

**Report Outline**

**1.Introduction**

**1.1 Motivation**

**1.2 Problem definition**

**1.3 Challenge**

**2. Literature review**

**2.1 Existing methods on stock movement predictions**

**2.2. Generative Adversarial Network**

**2.3. Convolutional Neural Network**

**2.4 Extreme Gradient Boosting**

**3. Models and Methodology**

**3.1 Data source**

**3.2 GAN Framework**

**3.3 Training GAN**

**3.4 Prediction model**

**3.5 Project difficulties and solutions**

**4. Results and conclusion**

**4.1 Evaluation metrics**

**4.2 Model results**

**5. Limitation and future work**

**6. Reference**

**1.Introduction**

* 1. **Motivation**

In the financial world, predicting stock prices is an promising topic, as a reasonably accurate prediction can yield high financial benefits and hedge against market risks. Financial analysts, stock traders and researchers suggest that stock market movements can be predicted based on statistical methods and historical data. Determining the historical pattern of financial time series is basically the major work for Wall Street.

For price prediction problem, there are two major ways: the one is the prediction error between the actual price and the predicted value (mainly RMSE (root mean square error) or RMSRE (root mean square relative error)); the other is direction prediction accuracy, which means the percentage of the correct prediction of the price sequence direction, because moving up and down is really important for decision making. As long as prediction can give investor some hints before stock disaster, it will prevent huge loss.

From investor’s perspective, they care more about profit rather than actual price of a certain stock. So we choose to predict the direction, to be specifically, whether certain stock will increase more than 5% in 10 interval trade day. Besides, we include 72 IT sector stocks, the price of stock different from each other, and direction is a more appropriate goal for our problem.

**1.2 Problem definition**

In this report, we propose a deep learning model to forecast the movements of stocks based on its historical data.

We will try to predict the price movements of 72 stocks in IT sectors, such as Apple (AAPL), PayPal (PYPL), etc. Under the assumption that stock movement is not completely a random walk, we believe there is hidden patterns for stocks of the same sector, and we want our deep neural network model to capture that to gain profits for investors.

In our network, we tried to predict whether a IT sector stock with rise 5% or not in 10 days. It is another binary classification problem, as there is numeric factor involved, we believe extreme gradient boosting model can perform well on it.

* 1. **Challenge**

Stock price predicting is never an easy job, it involves industry performance, investor sentiment, company news and performance, the complexity and nonstationary stochastic variables and many other economic factors--as many people regard it as a random walk. Also, it’s a hard task to determine how well our model performance is in existing data. And even it is gaining high return so far, it is hard to tell what it will be in the future.

Overfitting is another issue. Having a lot of features and neural networks we need to make sure we prevent overfitting and be mindful of the total loss. In order to achieve promising performance, most of these ways require careful selection of input variables, establishing predictive model with professional financial knowledge, and adopting various statistical methods for arbitrage analysis. We want to capture the feature without the noise.

On the other hand, the price of a stock is entirely determined by the confidence of the investor in future price, whether it will increase or decreases plus the expected dividends and prices received of potential acquisitions. Since stock prices are a function of investor behavior, stocks cannot be said to be “mispriced”. Its pricing mechanism is very bad and inefficient, and it runs a crazy and illogical process. Many price changes in stocks are based on quirks of human behavior, not on good financial awareness, which add more difficulty on our prediction model, since we can only include limited factors into our model.

**2. Literature review**

This section introduces the related work from the stock market prediction method.

**2.1 Existing methods on stock movement predictions**

As for existing methods, there are three common prediction methods: fundamental analysis, technical analysis, and machine learning.

The fundamental analysis attempts to measure the value by examining relevant economic and financial factors, which can be qualitative or quantitative. The fundamental analysis studies any factors that may have influence on the value of the security, including macroeconomic factors (such as economic and industry conditions) and microeconomic factors (such as financial conditions and company management). The ultimate goal of fundamental analysis is to generate a quantitative value that an investor can compare with the current price to indicate whether the stock is undervalued or overvalued.

The technical analysis in stock investment theory is an analytical method for predicting price direction through research on statistical trends gathered from trading activity, such as price movement and volume. There are many technical analysis methods, including Auto Regression, Moving Average, ARIMA, GARCH model, Neural Network model. As we are making binary classification predictions, SVM, Logistic Regression, ensemble methods such as Random Forest, Bagging and Boosting could be applied for this problem.

Unlike fundamental analysts, which attempt to evaluate a security's intrinsic value, technical analysts focus on patterns of price movements, trading signals and various other analytical charting tools to evaluate a security's strength or weakness. Technical analysis can be used on any security with historical trading data. This includes stocks, futures, commodities, fixed-income, currencies, and other securities. In fact, technical analysis is far more prevalent in commodities and forex markets where traders focus on short-term price movements.

In the past 20 years, due to the development of storage and tracking systems, a large amount of historical data can be used for analysis, so machine learning technology has become the main trend of new works. Since stock price is highly non-linear, and sometimes even completely random. Traditional time series methods (such as ARIMA and GARCH models) are only valid when the series is stationary. However, the main problem arises in implementing these models in real-time trading systems because of the lack of stability when adding new data. On contrast, neural networks do not require any stationarity. Furthermore, neural networks by nature are effective in finding the relationships between data and using it to predict (or classify) new data.

**2.2. Generative Adversarial Network**

Generative adversarial network (GAN) is a framework for estimating generative models via an adversarial process, in which we simultaneously train two models: a generative model G that captures the data distribution and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game. In the space of arbitrary functions G and D, a unique solution exists, with G recovering the training data distribution and D equal to 0.5 everywhere. While G and D are defined by multilayer perceptions in, most researches recently constructed G and D on the basis of Long Short-Term Memory (LSTM) or convolutional neural network (CNN) for a large variety of application.

**2.3. Convolutional Neural Network**

Convolutional Neural Network (CNN) is a class of deep, feed-forward artificial neural networks that has successfully been applied to analyzing visual imagery. A CNN consists of an input layer and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of convolutional layers, pooling layers, fully connected layers, and normalization layers. CNN also has many applications such as image and video recognition, recommender systems, and natural language processing.

**2.4 Extreme Gradient Boosting**

Extreme Gradient Boosting (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](https://en.wikipedia.org/wiki/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. Models and Methodology**

In this section, we will illustrate how we gather data, the detail of our GAN framework, CNN, XGBoost classifiers, and how we conduct our predictions.

**3.1 Data source**

The dataset we used consists of 72 IT sector stocks historical daily data, has 6 columns: timestamp; open; high; low; close and volume. Each represents the day of the stock price; stock’s open price; stock’s highest price of the day; stock’s lowest price of the day; the close price of the day, and the trade volume of the stock within that day.

Below is the sample data:

A screenshot of a cell phone

Description automatically generated

table 3-1 sample data

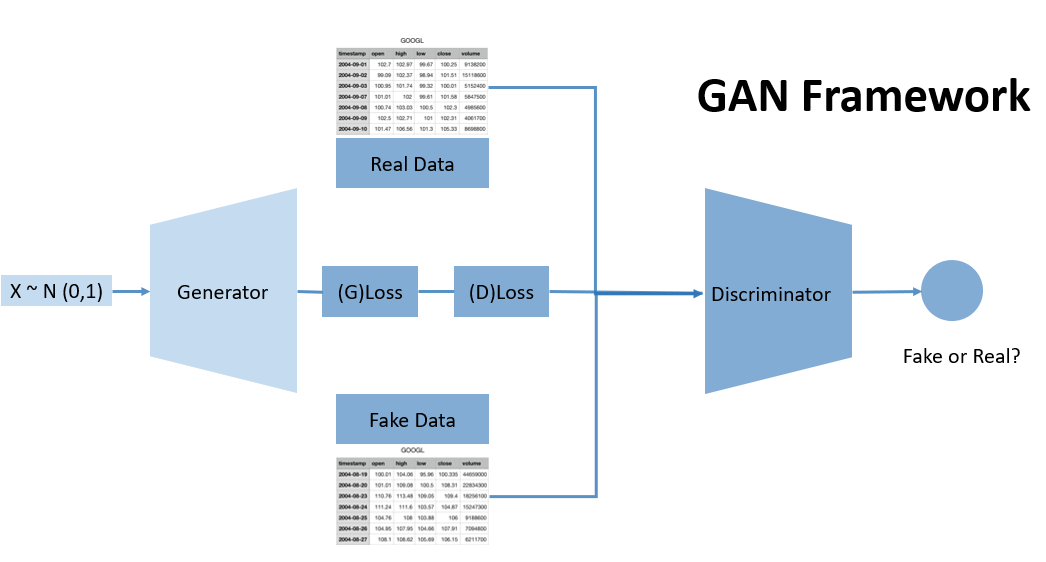
To gather 72 stocks life-long daily data and processing them in scale, we built up a pipeline on Google Colab to facilitate this procedure. The raw data comes from Alpha Vantage API, and stored in Google Drive.

Google Colab is a free cloud service and supports free GPU, where people can improve Python programming language coding skills. develop deep learning applications using popular libraries such as Keras, TensorFlow, PyTorch, and OpenCV.

Alpha Vantage is a leading provider of free APIs for real time and historical data on stocks, forex (FX), and digital/crypto currencies, which is driven by rigorous research, cutting edge technology, and a disciplined focus on democratizing access to data.

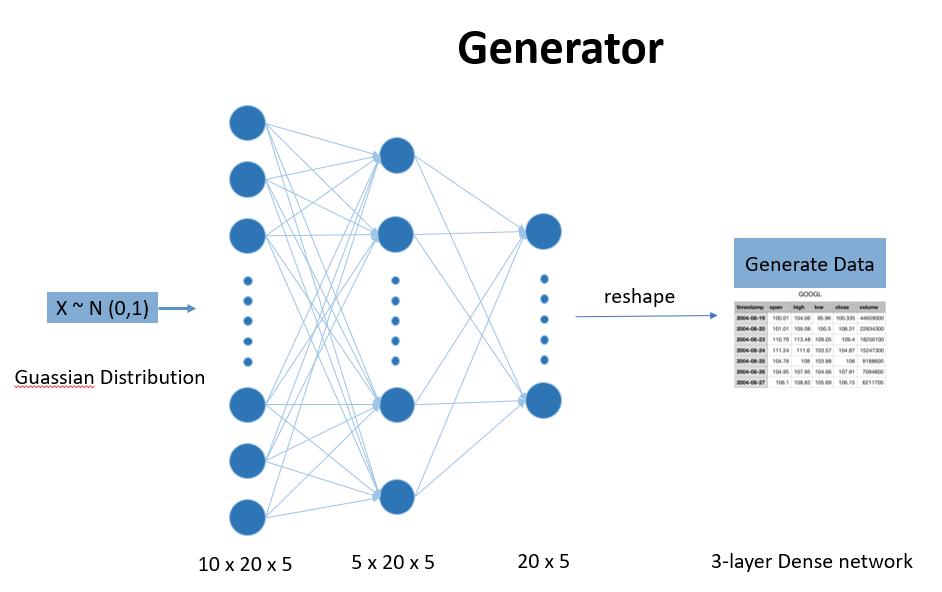
**3.2 GAN Framework**

Our GAN framework consists of a generator and a discriminator.

picture 3-1 GAN framework

**3.2.1 Generator**

We used a 3-layer Dense network as generator. The generator is trained to generate data that similar to the historical data of the target stocks we gathered from a gaussian distribution. We are using 20 days of stock data in each batch training, and there are 5 features for each day of a stock (except *timestamp*), the output dimension of the generator is [20\*5] each time for each stock. Number of neurons in each layer is 1000, 500 and 100.



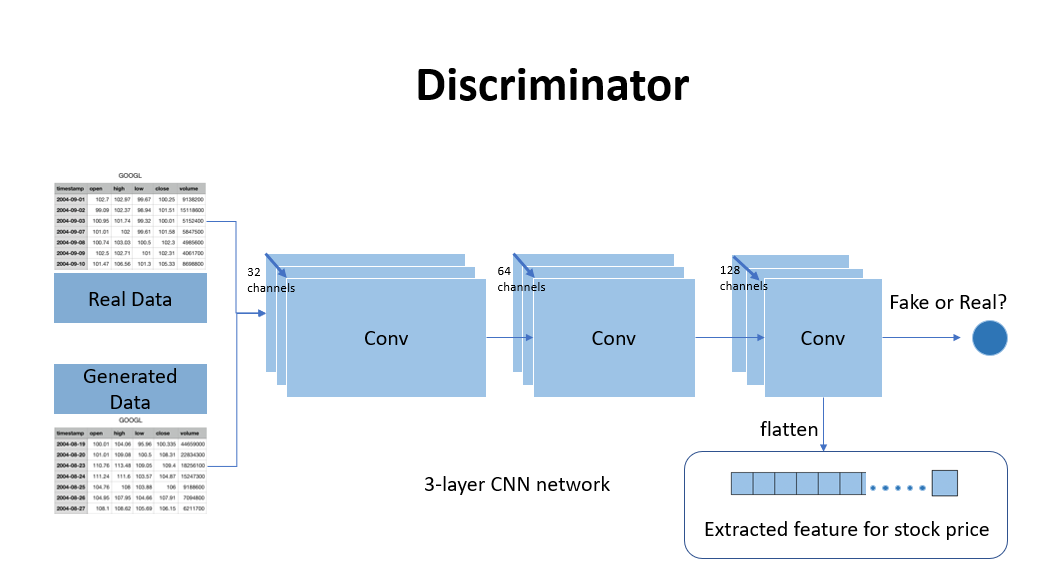
picture 3-2 Generator network

**3.2.2 Discriminator**

We used a 3-layer Convolutional Neural Network as our discriminator. The discriminator is trained to tell the difference between the fake data from the generator and the real data we gathered from Alpha Vantage. The error from the discriminator is used to train the generator to defeat the discriminator. The whole idea comes from Game Theory, it forces the generator to fake data as real as possible, in the end the discriminator can hardly tell the difference between the two.

The CNN network has 3 convolution layers, each has 32, 64 and 128 filters to represent the hidden pattern of stock price movement.

Inspired by word embedding, we also created a flatten layer consists of the last relu layers in the CNN network. The parameters in flatten layer is used as features for further usage.



picture 3-3 Discriminator network

**3.2.3 Loss Function**

The cross entropy between two distributions, which we’ll call p and q, is defined as:

where p and q denote a “true” and an “empirical/estimated” distribution, respectively. Both are discrete distributions; hence we can sum over their individual components, denoted with i.

To apply this loss function to the current binary classification task, we define the true distribution as P[yi=0]=1 if yi=0, or P[yi=1]=1 if yi=1. Putting in 2-D vector form, it’s either [1,0] or [0,1]. Intuitively, we know for sure which class this belongs to, so it makes sense for a probability distribution to be a “one-hot” vector.

Thus, for one data point x1 and its label, we get the following loss function:

In the case of GANs, we can say a little more about what these terms mean. In particular, our xi only come from two sources: either xi∼p data, the true data distribution, or xi=G(z) where z∼p generator, the generator’s distribution, based on some input code z. It might be z∼Unif[0,1] but we will leave it unspecified. This is the loss function for the discriminator, J(J).

**3.3 Training GAN**

**3.3.1 Data Normalization**

We want to train a generalized model that works well for every IT sector stock. If we train our model based on a certain stock, there is potential overfitting issue, which means the model may perform well on that training stock, but would perform badly on other stocks. On the other hand, data from one stock is limited, with more training data, we expect our model to be better in capture features that can hardly noticed by eyes.

However, different stock has different price ranges. To deal with this problem, we first split data every 20 days, which is the number of historical days we used to make prediction, and normalized price and volume within each period.

The normalized price are derived as same with volume.

,

After that, we delete the NA records generated from normalization procedure.

**3.3.2 Split training and testing dataset**

To avoid overfitting issue, for each IT sector stock historical data, we held out latest 255 records as testing data (average 255 trade days in a year). For historical day reason, to avoid the connection between training and testing dataset, we use stock data before 300 days as training dataset originally.

Due to the unbalanced data issue, we resampled the testing set to have similar number of observations with both labels. We will illustrate this in detail in section 3.5.

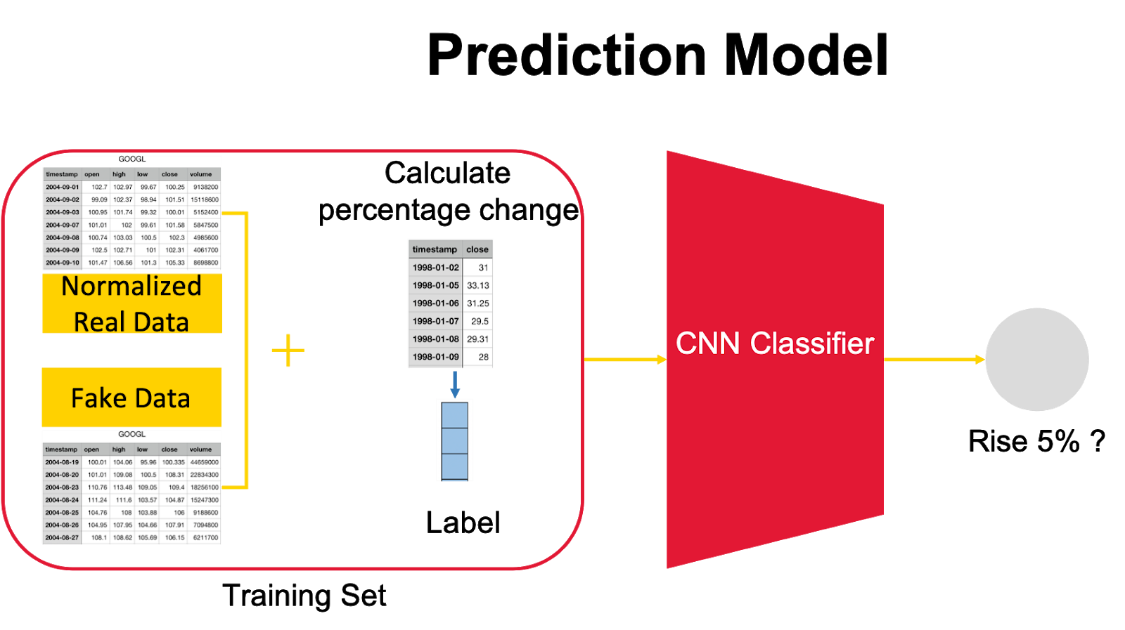
**3.4 Prediction model**

We used a 3-layer Convolutional Neural Network as our classifier in prediction. The dataset consists of two parts:

**Predictors (X):** The parameters in the flattened last hidden layer of Discriminator;

**Labels(y):** Calculated from original dataset’s percentage change. 1 if rise 5% in 10 days otherwise 0.

Below is the structure of our prediction model:



picture 3-4 Prediction model structure

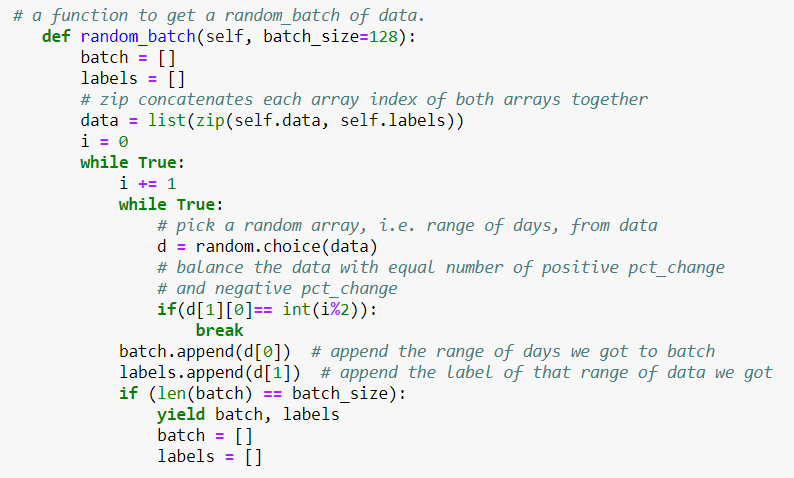
**3.5 Project difficulties and solutions**

**3.5.1 Limited and unbalanced data**

For some stocks, such as Bilibili (BILI), listed on Nasdaq in March last year, doesn’t have enough data for training. Without enough data most model can not work well, and the result isn’t stable in statistic sense.

On the other hand, we want to predict whether a stock could rise 5% in 10 days. It’s inevitable that most of the time a stock cannot rise that much, thus most of our label is ‘No’. As a result, without resampling out dataset is unbalanced.

To tackle these issues, we used GAN to generate fake stock price data. During training, we used equal number of positive and negative labels in each batch; For testing, we resampled and constructed a balanced testing set as well.

****

**3.5.2 Price and volume scales issue**

In the beginning, we tried to fake real stock daily data but got unsatisfying results. The volumes, which supposed to be significantly larger than daily price, is hard to fake with an input of Gaussian distribution for generator. Then we realized where the problem is.

Unlike image pixels which all values ranging from 0 to 255, our 5 selected features, including stock price and trade volume don’t have a fixed range. What’s more, different stocks have different ranges, such as Apple’s daily trade volume is significantly larger than average.

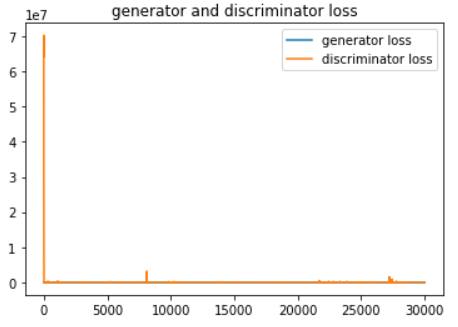
To deal with this scale problem, we normalized stock data within 20 days by columns. If we normalize all data together, stock price would be relatively small due to the large trade volume.

1. GAN Results after 50k epoch without normalization:

Generated Data Real Data

Generator loss =11042.477135150582, Discriminator loss =3.5966453122496604 at 50k iteration. From loss Discriminator results we also noticed the underperformance:



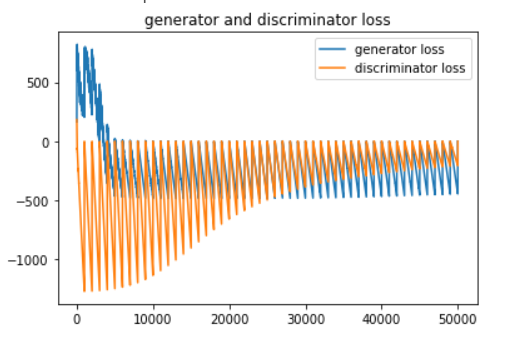
picture 3-5.1 GAN loss in 50k iteration without normalization

1. GAN Results after 50k epoch with standard normalization:

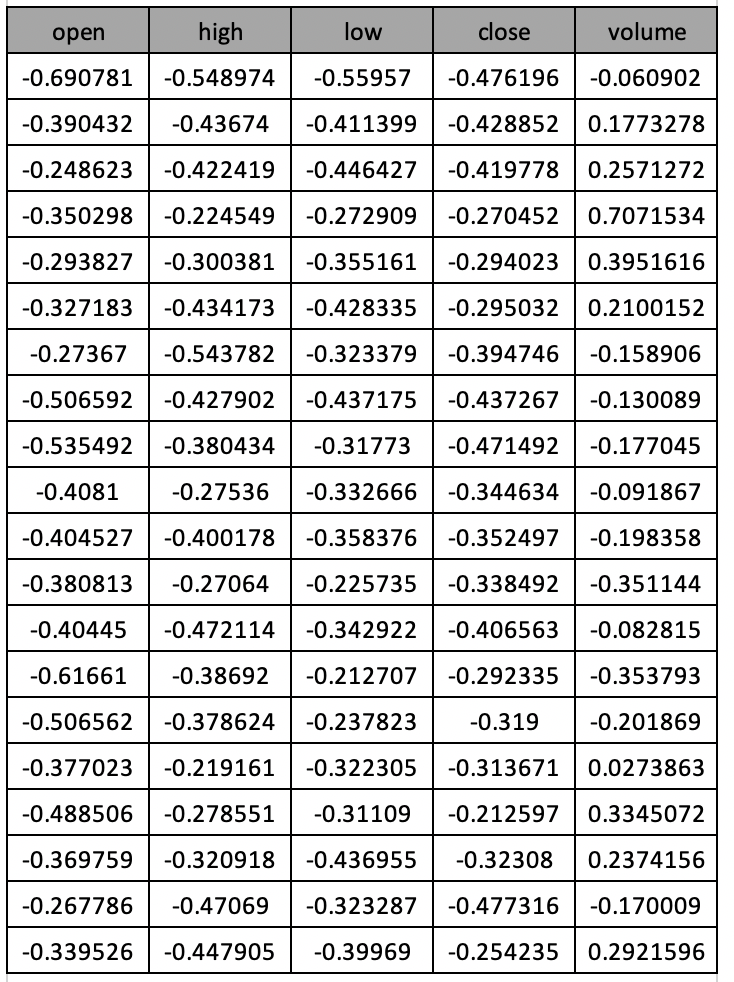
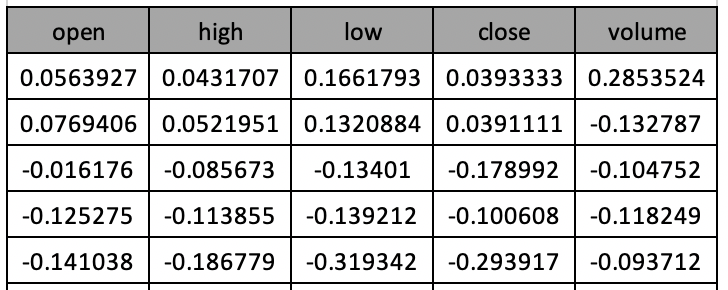
Generated Data Real Data

With normalization, generator and discriminator work well with respect to its loss. However, human beings are able to tell the difference between the two, as generated data doesn’t have absolute 0 or 1 values (original minimal or maximum values within 20 window size).

Generator loss =1.5416529471576212, Discriminator loss =1.0951287390738726 at 50k iteration.

picture 3-5.2 GAN loss in 50k iteration with standard normalization

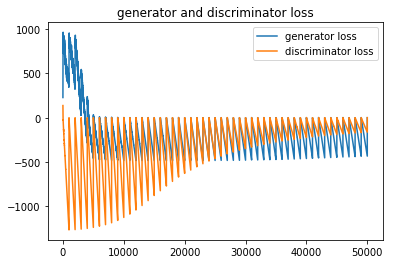
1. GAN Results after 50k epoch with mean normalization:

Generated Data Real Data

With normalization, after 50000 times iterations, the losses of generator and discriminator are both low and converged. Human beings cannot tell the difference between fake data and real data.

Generator loss =1.4119627691209315, Discriminator loss =1.1382573942840097 at 50k iteration.

picture 3-5.2 GAN loss in 50k iteration with mean normalization

**4. Results and conclusion**

**4.1 Evaluation metrics**

We have 2 methods to evaluate our model: accuracy and True Positive Rate (TPR). We choose accuracy because it’s one of the most important way to evaluate a model. We also choose TPR because the cost of this misclassification is very high, since investors will lose money when they misclassified a falling stock as a raising stock.

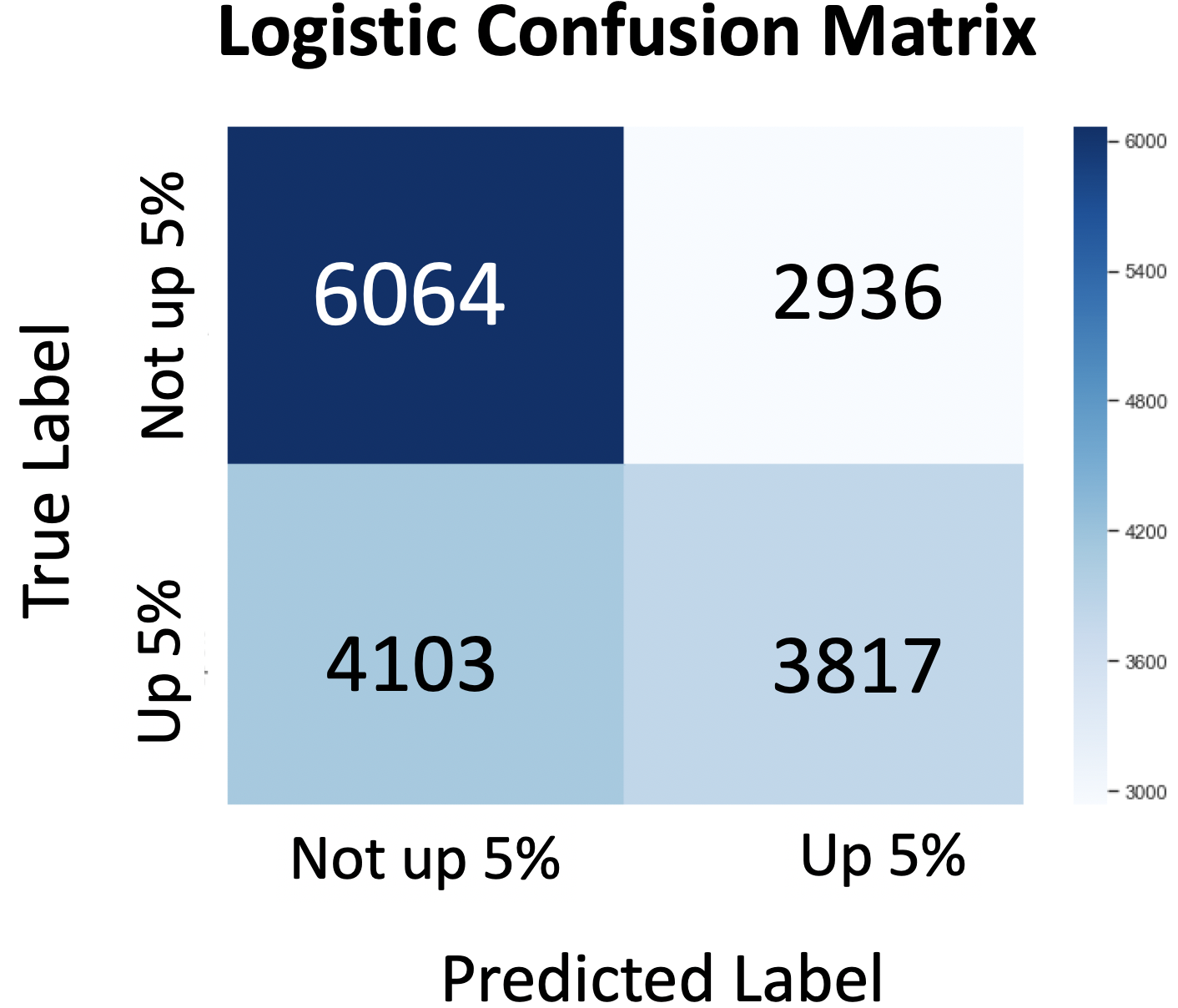
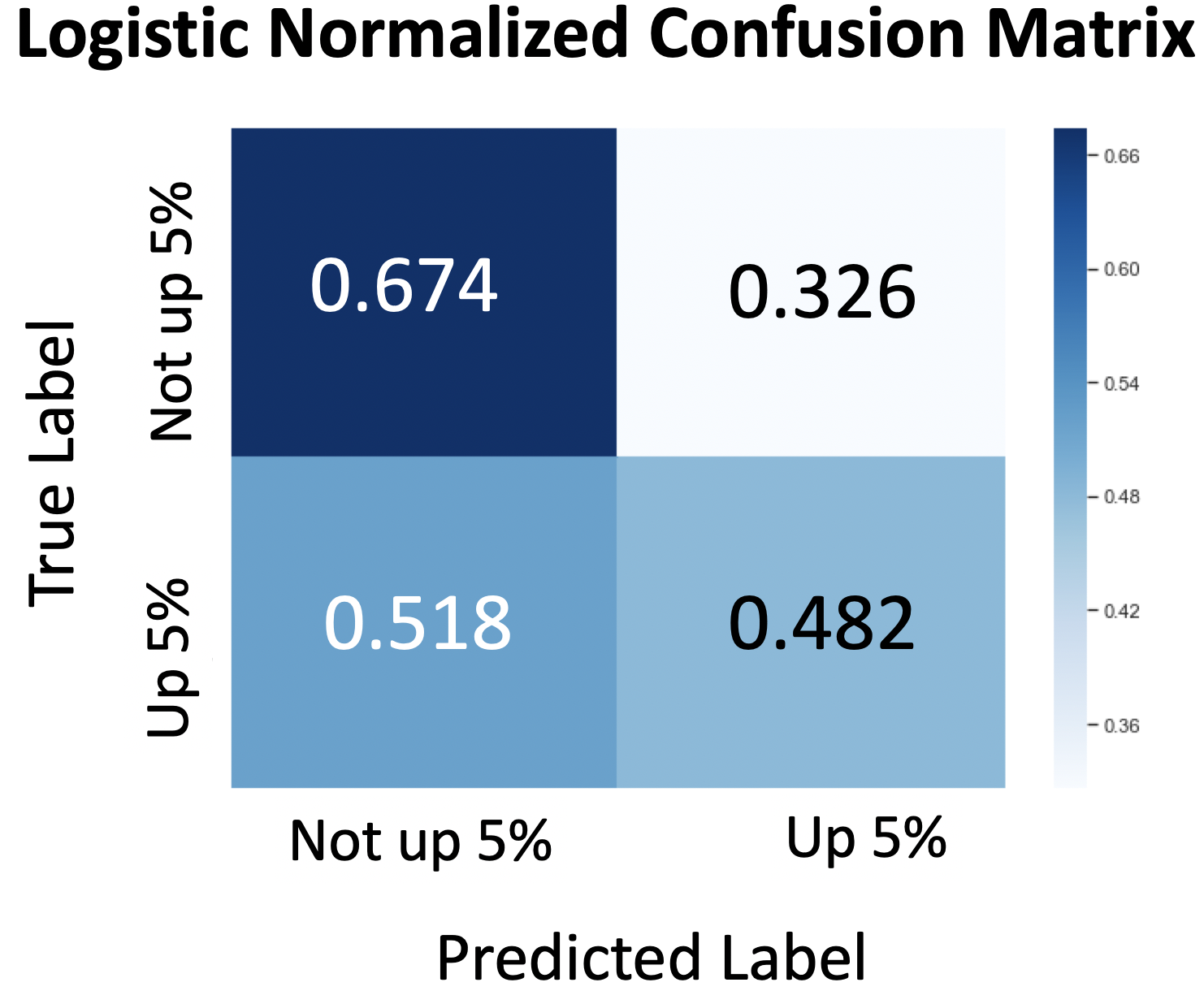
**4.2 Model results**

**4.2.1 CNN model trained with unbalanced data**

In this project, we first trained a CNN model in 72 stock data to predict if a stock’s price will go up by 5%. We also trained a logistic model as our baseline since it’s a commonly used binary classification model. During this process, we encountered a problem that some new stocks don’t have enough training data. We can only put them aside.

The testing set we used was a resampled dataset, in which there is about 50% samples’ price will go up by 5% and 50% will not.

In the testing set, the baseline Logistic model’s accuracy is 58.40% with a True Positive Rate (TPR) of 48.2%, and the CNN model’s accuracy is 71.97% with a TPR of 57.83%.

** **

picture 4-2.1 Logistic Classifier results

For overall accuracy, CNN model performed better than the baseline logistic model.However, the TPR is only 57.83%. There is still room for improvement. We noticed another problem is that our training data is unbalanced (most stocks’ price will not go up by 5%).

**A screenshot of a cell phone

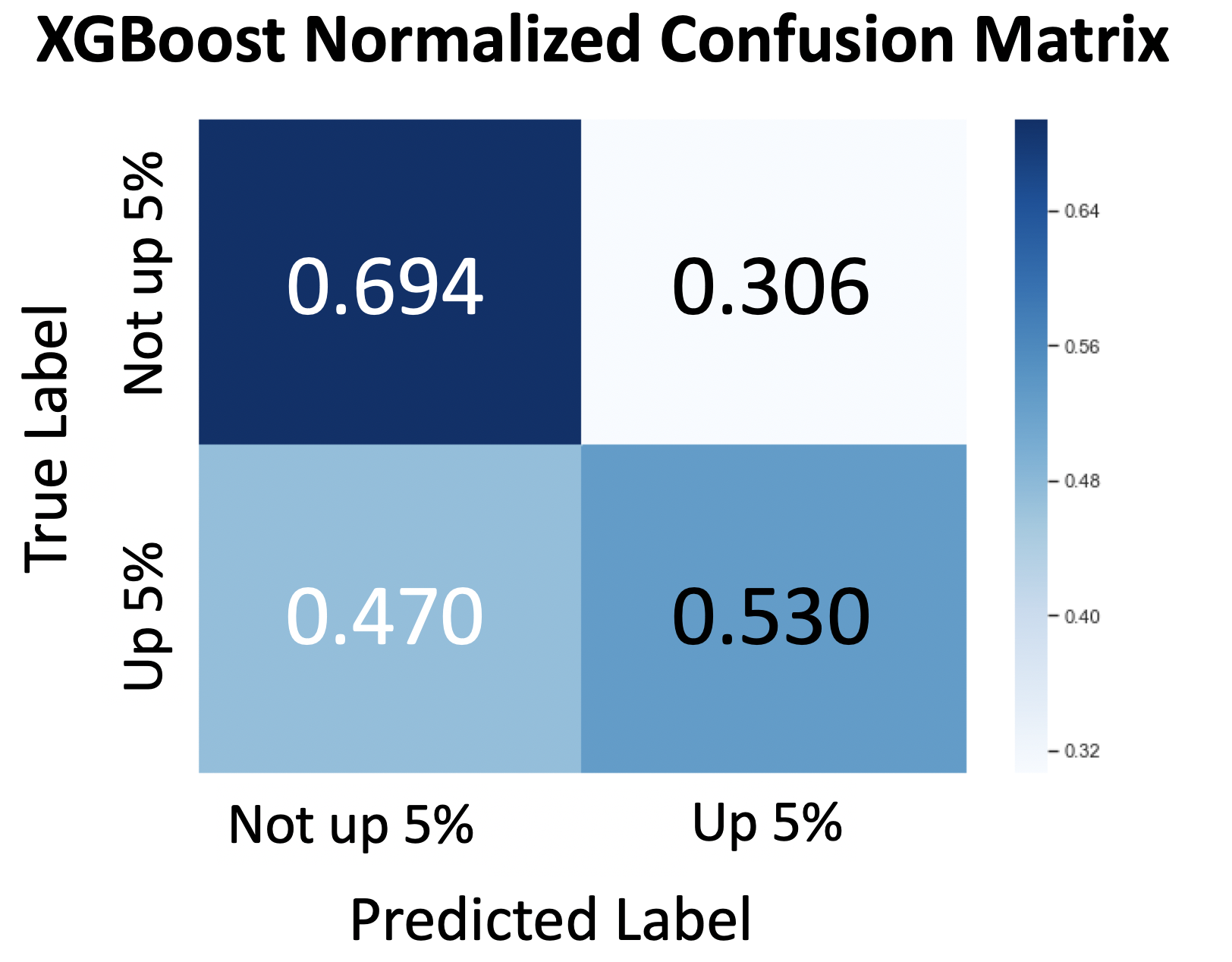
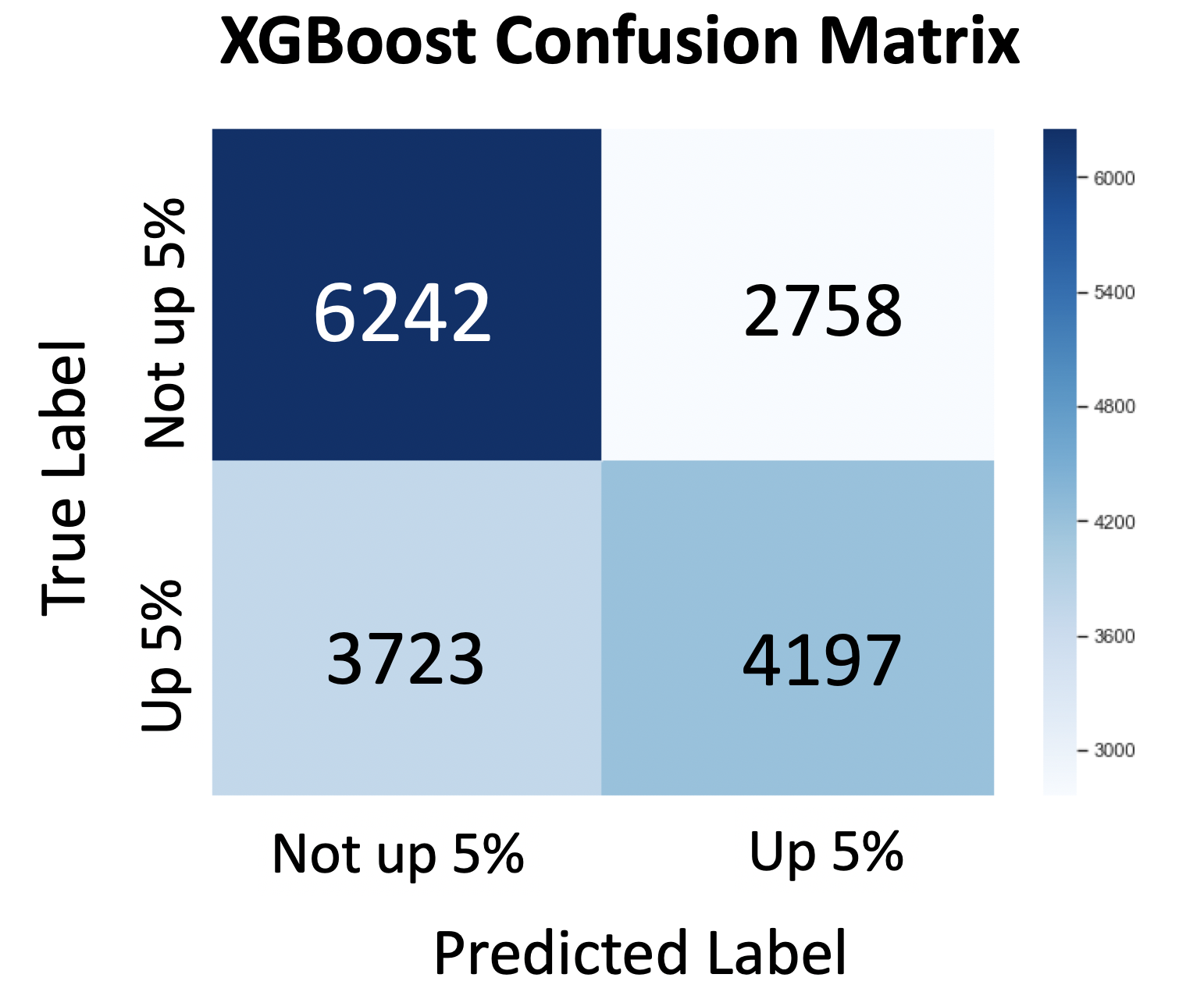
Description automatically generated A screenshot of a cell phone

Description automatically generated**

picture 4-2.2 CNN Classifier (trained with unbalanced data) results

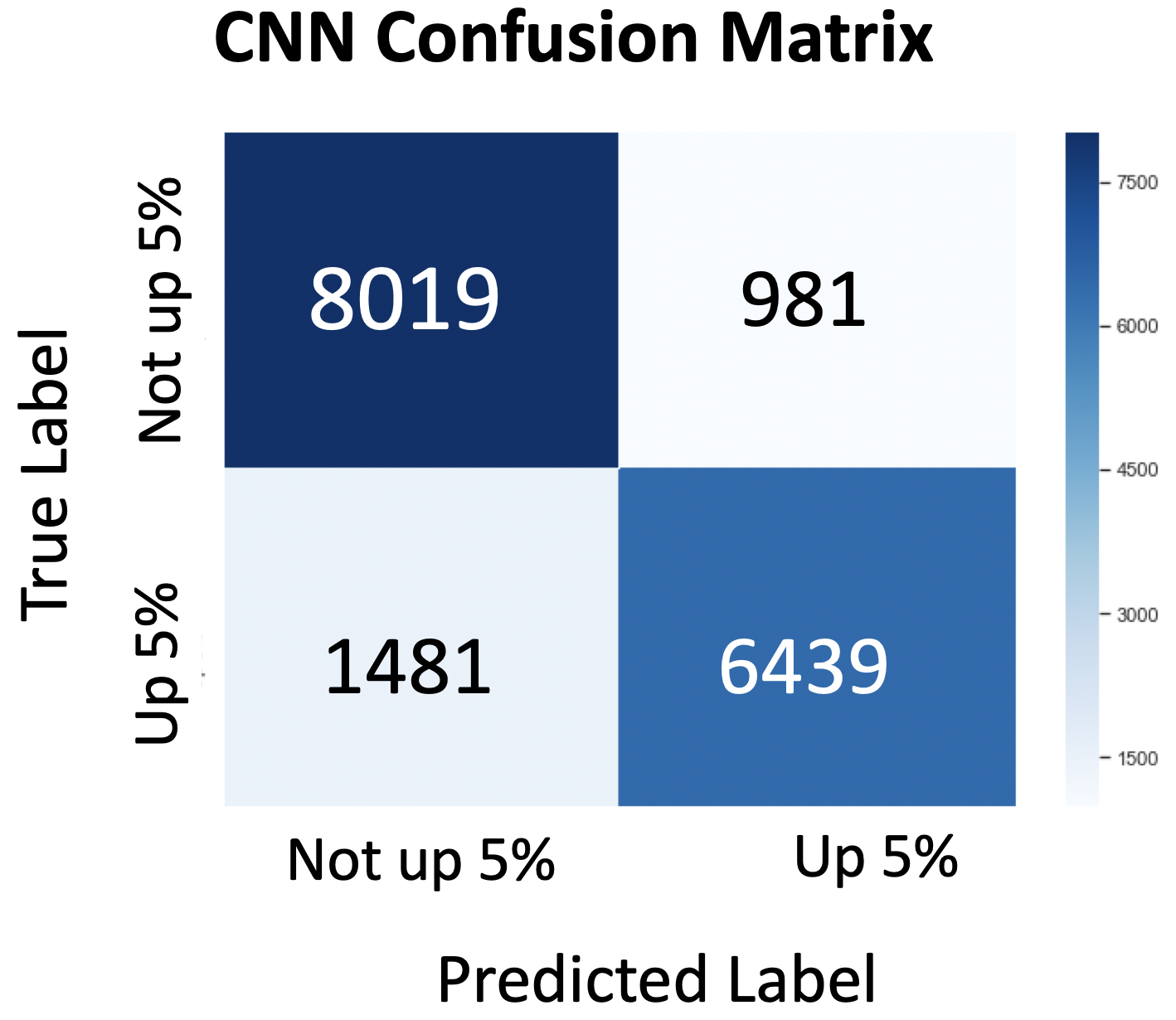
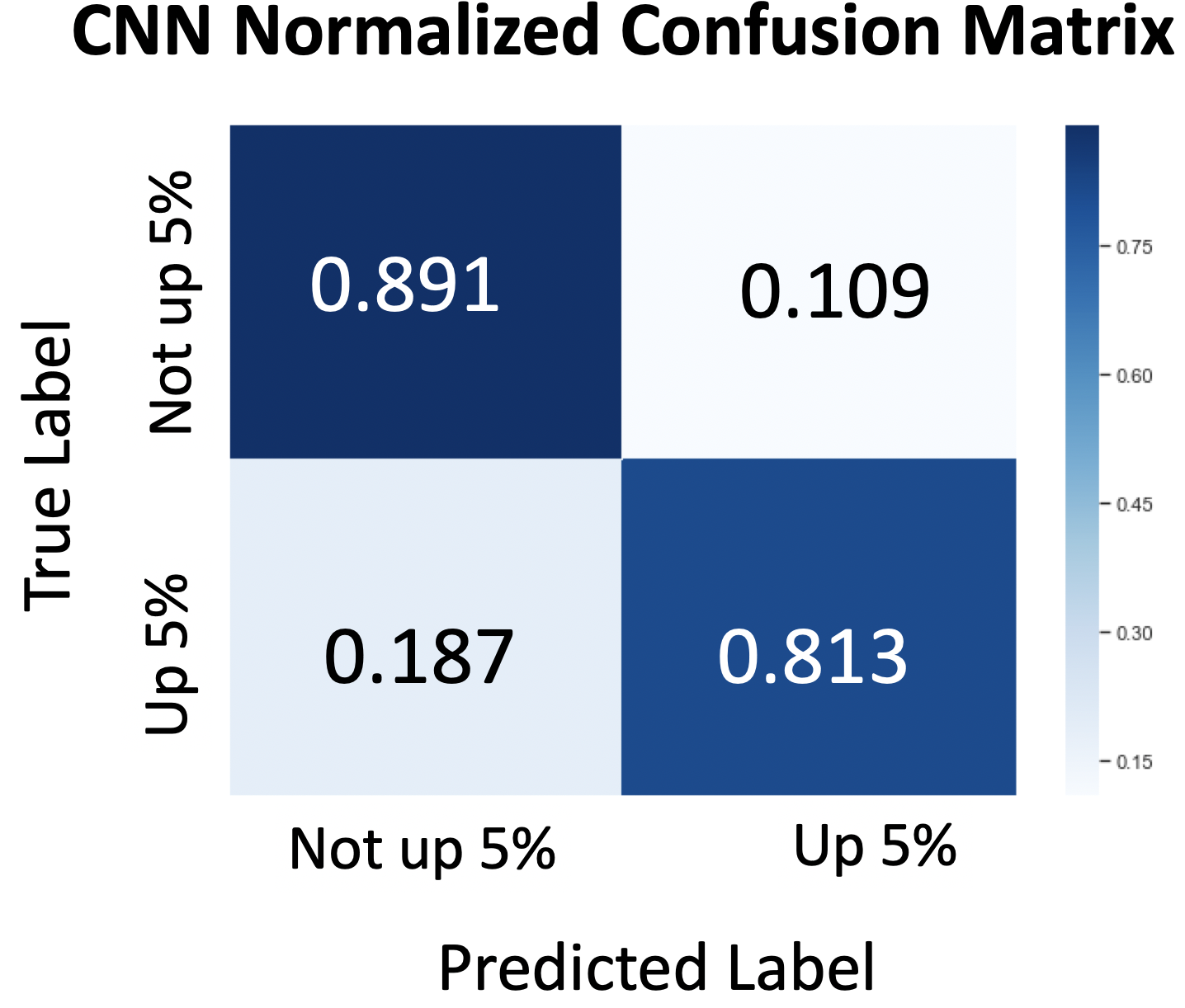
**4.2.2 CNN model trained with balanced data**

To solve the above problems, we introduced Generative adversarial network (GAN). For those stocks whose information is not enough, we use GAN to generate their fake data. Then we used GAN to generate more than 3,000 fake data whose price will go up by 5%. We used the generated fake data and part of our real data to train a CNN model and an XGBoost model as our baseline.



picture 4-2.3 XGBoost Classifier (trained with balanced data) results

In the testing set, the baseline XGBoost model’s accuracy is 61.70% with a TPR of 53.0%, and the CNN model trained with the balanced data outperforms the other three models with 85.45% accuracy and an 81.30% TPR.

** **

picture 4-2.4 CNN Classifier (trained with balanced data) results

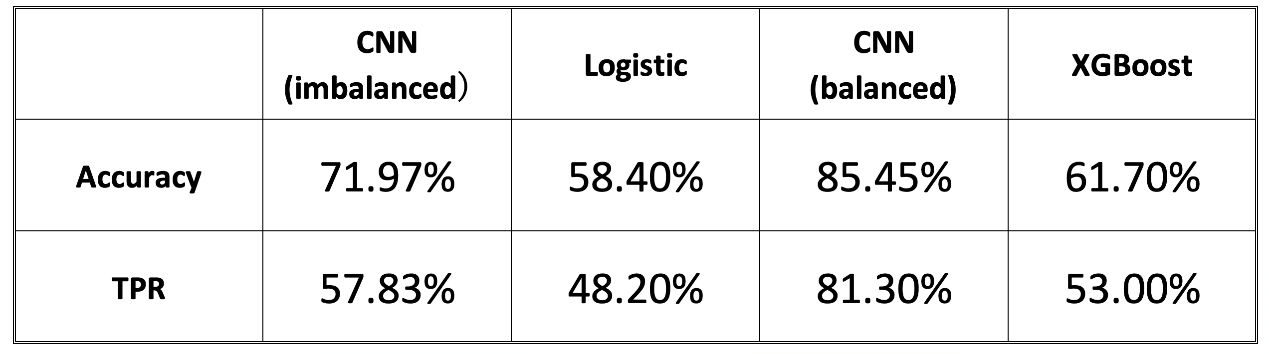


table4-2.4 CNN classifier performance compared with baselines

**5. Limitation and future work**

We have proposed and demonstrated the efficacy of our model for predicting the changing direction of stock price in 10 days with accuracy of 92.6% TPR. Our model still has its limitation. And we will out more effort on these subjects in the future.

First of all, neural networks with other loss functions and different architecture could be explored. From an architecture perspective, we would like to refine our architecture to test deeper models with more layers, different number of neurons and customize our loss function to better deal with unbalanced data. Also, we have only explored using CNN as our generator, using other types of networks (e.g. LSTM) might helps us to achieve higher accuracy.

Second, although our model has achieve good performance on predicting price change in 10 days. But it only got 55% accuracy on predicting the outcome in second day, which will provide extra value to stock price’s change. We will explore more solution to improve it, which may also provide particularly useful information for stock holder.

Furthermore, historical price data is just small part of features according to stock price prediction problem. There are many others we could incorporate into our model, for example financial news, political influences, competitor trends, and economic indicators. By adding more input features to capture the full scale of complexities of a stock market, the robustness of our model should be improved.

**6. Referrence**

[1] Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A.C., Bengio, Y., 2014. Generative adversar-ial nets, in: Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada, pp. 2672–2680.

[2] Ding, X., Zhang, Y., Liu, T., Duan, J., 2015. Deep learning for event-driven stock prediction, in: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015, pp. 2327–2333.

[3] A. Booth, E. Gerding, and F. McGroarty, “Automated trading with performance weighted random forests and seasonality,” Expert Systems with Applications, vol. 41, no. 8, pp. 3651–3661, 2014.

[4] A. Bagheri,H.Mohammadi Peyhani, andM. Akbari, “Financial forecasting using ANFIS networkswithQuantum-behaved Particle Swarm Optimization,” Expert Systems with Applications, vol. 41, no. 14, pp. 6235–6250, 2014.

[5] Y. Son, D.-J. Noh, and J. Lee, “Forecasting trends of highfrequency KOSPI200 index data using learning classifiers,”Expert Systems with Applications, vol. 39, no. 14, pp. 11607–11615, 2012.

[6] I. Aldridge and S. Krawciw, “Real-Time Risk: What Investors Should Know About FinTech,” in High-Frequency Trading, and Flash Crashes, JohnWiley&Sons, Inc., Hoboken, NJ, USA, 2017.

[7] F. A. De Oliveira, C. N. Nobre, and L. E. Z´arate, “Applying Artificial Neural Networks to prediction of stock price and improvement of the directional prediction index - Case study

of PETR4, Petrobras, Brazil,” Expert Systems with Applications,

vol. 40, no. 18, pp. 7596–7606, 2013.