Stock Market Prediction with Machine Learning

MATLAB FINAL PROJECT

Danyel Koca
ID: 1023281915

Introduction

Stock markets have captured the interest of generations of people including mathematicians, engineers, and scientists. Due to abundance of financial data and spike in computational power, it is increasingly believed that stock market returns can be predicted with due information and computing power. In this project, we test some of the indicators that financial analysts use to produce buy and sell signals.

Background

The dominant belief among finance and economics scholars is that the stock market follows a *random walk*, which implies that the returns of tomorrow can not be predicted with any information in hand. This notion creates the foundation of *Efficient Markets Hypothesis* (EMH) which asserts that all the available information is already incorporated in the stock prices, and the market price is the best estimate of *value*. A corollary of the EMH is that any irrational divergence from true value of a stock is going to be corrected quickly by profit-seeking market participants. Therefore, some economists argue that market prediction is a fool's errand. This notion is illustrated by a popular story in economics:

Two economists – one young and one old – walking down the street together. The young economist looks down and sees a $20 bill on the street and says, "Hey, look a twenty-dollar bill!" Without even looking, his older and wiser colleague replies, "Nonsense. If there had been a twenty-dollar lying on the street, someone would have already picked it up by now."

Economists like to deal with theoretical, ideal concepts which may not be the case in real-life. One of the underlying assumptions of the EMH is that market participants are *rational* who behave in exact accordance with their self-interest. In markets, it's not always prefrontal cortex who commands market participants, every once in a while, amygdala hijackes the brain when the individual is facing a fight or flight situation or an easy (often times highly risky) profit. When all market participants behave similarly, this causes an irrational exuberance which pushes the value of securities to uncharted waters.
We always see large divergences in stock prices that do not get corrected for a long time. One example is the dot-com bubble around the turn of the millennium, where stocks were sold at ridiculous prices, even though the underlying value of stocks were minimal. On the other hand, when pessimism prevails in the markets, as in the financial crisis of 2008, everybody tries to get out of the market, no matter what the underlying value of the stocks are.

> *The rational man-like the Loch Ness monster-is sighted often, but photographed rarely.*
> *David Dream*

Stock market history is full of booms and busts, and this irrationality gives stock market analysts hope for the case of predictability of stock prices. It also discourages some, as stock prediction, in effect, is predicting the psychology of market participants.

Two techniques analysts use for buy-sell decisions are the *fundamental analysis* and *technical analysis*. Fundamental analysis is a method that contrasts market-value (stock price) of a firm to its underlying value (also called *book-value* which is the assets of a firm subtracted by its liabilities). For example, if market price of a firm is many multiples of its book-value, that stock might be overbought - over-priced. (e.g. Netflix: Price-to-book ratio = 44 as of July) On the other hand, if a firm is unpopular among investors, it might sell at a discount to its actual value. (e.g. Deutsche Bank: Price-to-book ratio = 0.28 as of July). This might seem ridiculous to the layperson, because even if the Deutsche Bank were to liquidate now, you would get 500% of your initial investment. This is akin to buying a box which has $1 inside for 28 cents.
Book value of firms may be irrelevant to many investors. Because you wouldn't pay the book value if were you to buy stocks. You wouldn't get the book value if were you sell stocks. However, fundamental analysis can be used to determine bargain stocks and avoid the expensive ones.
While fundamental analysis helps investor determine *what* to buy/sell, technical analysis helps investors determine *when* to buy/sell. Technical analysis uses the market data such as closing price, volume of trade, intraday high/low to determine if a stock is at the end of its boom cycle, or if a momentum is building up in the stock. Researchers have shown that, even if the stock market resembles a random walk, stock prices are not totally independent of the previous data. There are various anomalies taking place in the stock market, for example, stocks of small firms tend to do better than large firms. In January, stock market yields superior returns compared to other months and so on. All this anomalies hint at a structure of market that is not inherently unpredictable. Due to these anomalies and exuberance that takes place in the stock market, we believe that, if we can determine the momentum, volatility, and cyclic behavior of stocks then we can generate buy/sell signals that can provide superior returns to a simple buy-hold strategy.

Method

Data

In this project, we analyze the S&P 500 index which is a market-capitalization weighted index of the largest 500 companies headquartered in the U.S. The data frequency is daily and ranges from January 1950 to July 2018. We get the historical data of daily price movements from Yahoo Finance. We work this data in Python to make it compatible for Matlab. The data retrieved from Yahoo Finance includes: date, open, high, low, close, adjusted close, and volume. Open is the opening price of the S&P 500 index on New York Stock Exchange (NYSE). High and low are the highest and lowest prices achieved intraday, respectively. Close is the closing price of the stock on NYSE. Adjusted close takes mergers, acquisitions and dividends into account and adjusts closing price accordingly. For this analysis, we use adjusted close, which is a more realistic representation of the stock market at the end of trading day. Finally, volume is the number of times that the stocks of components of S&P 500 has changed hands on that day at NYSE. Moreover, we get VIX (Implied Volatility Index by Chicago Board Options Exchange) data from Yahoo Finance. For this project, we determined 13 technical indicators that

are used by analysts trying to time the market. These are:

1. 50-day Simple Moving Average
2. 26-day Exponential Moving Average
3. Relative Strength Index
4. Implied Volatility Index
5. 10-day Realized Volatility Index
6. B% Oscillator
7. Alexander's Filter
8. Smoothed Stochastic Oscillator
9. 14-day Money Flow Index
10. Accumulation - Distribution Line
11. Chaikin Oscillator
12. Moving Average Convergence - Divergence Oscillator
13. Williams R% Oscillator

All the indicators except VIX Index can be computed from the historical price data of S&P 500 Index. Implied volatility is calculated by using Black-Scholes Option pricing model, which requires price data of options. We import this data from Yahoo Finance. We skip the explanation of these indicators, but it must be noted that all these indicators are used to gauge the momentum and volatility present in the market, in order to determine whether the market is bullish (likely to go up) or bearish (likely to go down).

Analysis

`data_analysis.m` takes the data files in the directory and computes indicators. `data_plot.m` is a visualizer. This program lets the user plot S&P 500 index over a desired period of time with a desired indicator. Let's work the program and plot the bull market that has been taking place after the financial crisis in 2008:

```
>>>data_plot

Indicator Visualizer

Please enter the number
that corresponds to the number of indicator below to see the time series:
1 - 50 Day Simple Moving Average
2 - 26 Day Exponential Moving Average
3 - Relative Strength Index
4 - Implied Volatility Index
```

```
5 - 10 Day Realized Volatility
6 - B% Oscillator
7 - Alexander's Filter
8 - D% Osciallator
9 - Money Flow Index
10- Accumulation - Distribution Line
11- Chaikin Oscillator
12- Moving Average Convergence - Divergence -- Signal
13- Williams R%
and the time range: {indicator_no,'dd.MM.yyyy','dd.MM.yyyy'}
e.g.: {12,'01.04.1987','05.04.1989'}
Or leave the range section empty,
if you want to see the time series for the whole dataset. e.g.: 5
>>>{12,'01.04.2008','05.04.2018'}
```

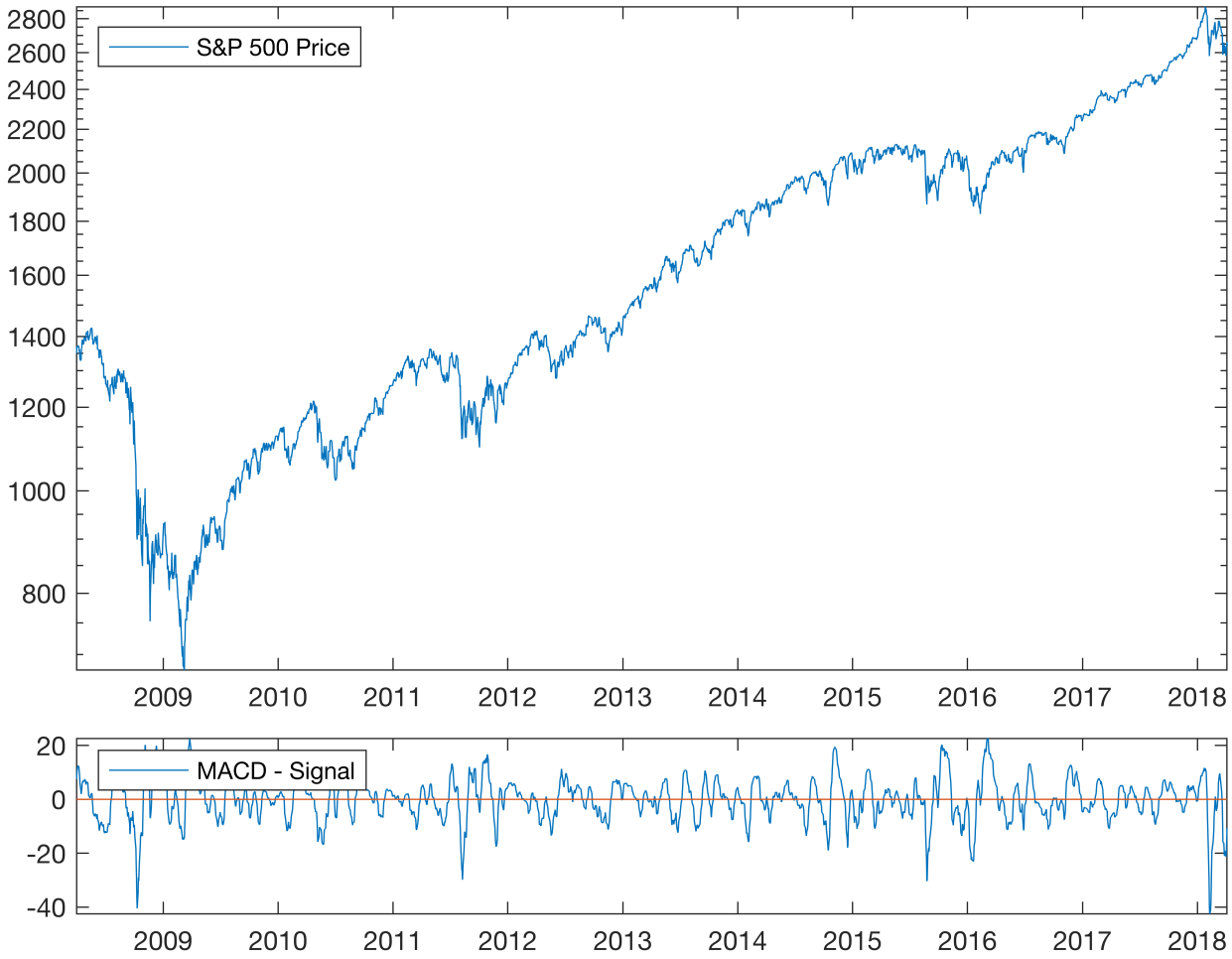

Figure 1: Bull market since 2008: one of the longest bull markets of the U.S. history.

Let's plot the October 1987 crash, also called as the *Black Monday*. This is shown in Figure 2. From here on we start designing the machine learning algorithm. We create a neural network with 1 hidden layer which can have arbitrary number of neurons. In order to check if the model is working we test it with toy data.

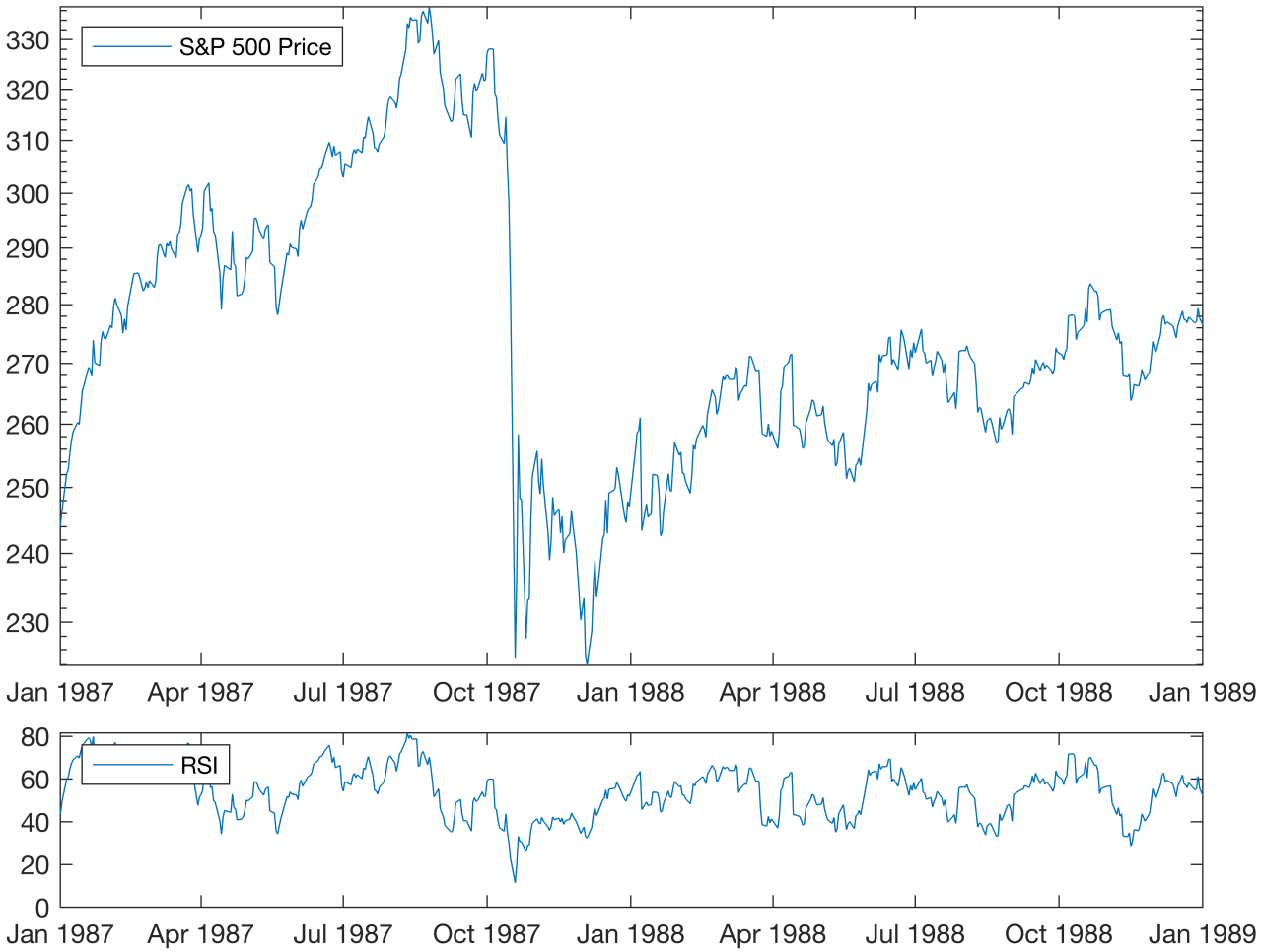

Figure 2: October 19, 1987: Black Monday

```
>>>project_toy1
```

Figure 3 (Figures are at the end of the paper from here on) shows the distribution of the whole dataset. Figure 4 shows the loss function. It can be seen that loss goes to 0 at the second epoch. However, it must be noted that the loss function can get stuck at a local minimum which is above 0. Or, if the rate of gradient descent (learning rate) is small, convergence may take long time and there is a higher probability that the loss function will get stuck at a local minimum. Figure 5 shows the accuracy of the model tested with test data, which is 20% of the original data. Blue color shows the labels that were correctly identified by the model. Figure 6 shows the spatial distribution of the predicted labels. Now that we know that the model can act as an XOR gate, we can train it with some other nonlinear data.

```
>>>project_toy2
```

This dataset and learning model can be observed in Figures 7, 8, 9, and 10. As seen in Figure 7, the loss function got stuck in the local minimum, but thanks to the magnitude of the rate of gradient descent, it was able to jump out and move to the global minimum. The last dataset we test our model has a peculiar shape:

```
>>>project_toy3
```

The loss function of this distribution has many local minima. Therefore the model always gets stuck on a local minimum. If we increase the rate of gradient descent, the loss function jumps to an even worse minimum. This low performance is due to the fact that there is only 1 hidden layer in the model. If we were to increase the sophistication of the model, the model would surely learn this complex distribution. However this testing was helpful to show us the limitations of our model. Results can be seen in Figures 11, 12, 13, and 14.

Now that we made sure that the model is working, we can feed the financial data.

```
>>>project_stock
```

The inputs are 13 indicators for each day. Label for each day is a 1x3 matrix. If the stock goes up next day, the 1^{st} element is 1, if goes down, the 2^{nd} element is 1. If the closing price does not change the next day, the 3^{rd} element is 1. The rest of the elements are 0.
When we train the model, the algorithm can not find any meaningful relationship between inputs and labels, and accuracy is always below 0.6.
After this, we change the labels into a continuous scale. We first take the logarithm (base 10) of the closing prices due to the fact that stock prices rise exponentially. Then we normalize the data by using below formula:

$$\text{Normal Price} = \frac{\text{Price} - \text{Minimum Price}}{\text{Maximum Price} - \text{Minimum Price}}$$

Minimum and maximum prices are obtained from the whole dataset. We also log-normalize the simple and exponential moving averages since they show the same exponential behavior. We also normalize all the features except Alexander's Filter, which is binary. When we train the model with 4 epochs, we get satisfying results. Please see the cost function in Figure 12. It can be observed that the loss is concentrated around 0 at the last epoch. The upward deviances we see in the loss function was expected due to the unpredictability of stock market. Although R^2 is not meaningful for non-linear relationships, we compute it in order to have a glimpse on accuracy. Please see Figure 13 for the plot of predicted vs. actual values and the R^2. It looks the model has done a good job. However, there may be some errors underway which are creating superfluous relationships and giving us a wrong picture of accuracy.

Conclusion

Although what Burton Malkiel has said in his famous finance primer *Random Walk Down Wall Street*: *"A blindfolded monkey throwing darts at a newspaper's financial pages could select a portfolio that would do just as well as one carefully selected by experts."* , there seems to be a relationship between the indicators and the overall stock market. Future work will be focused on creating a fast, modular algorithm with arbitrary number of hidden layers as well as a recurrent neural network, finding quality data, and training models on these datasets.

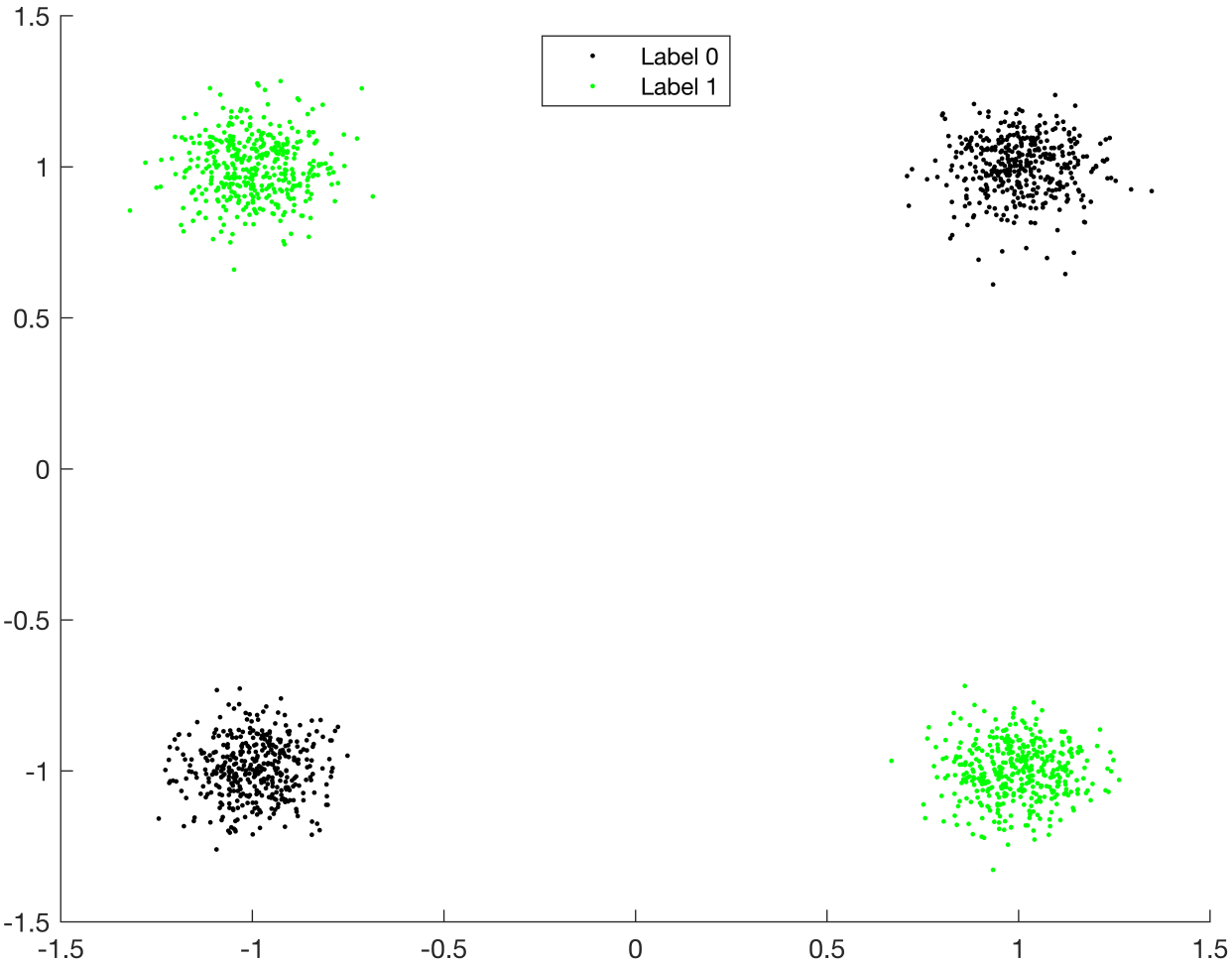

Figure 3: Dataset - 1

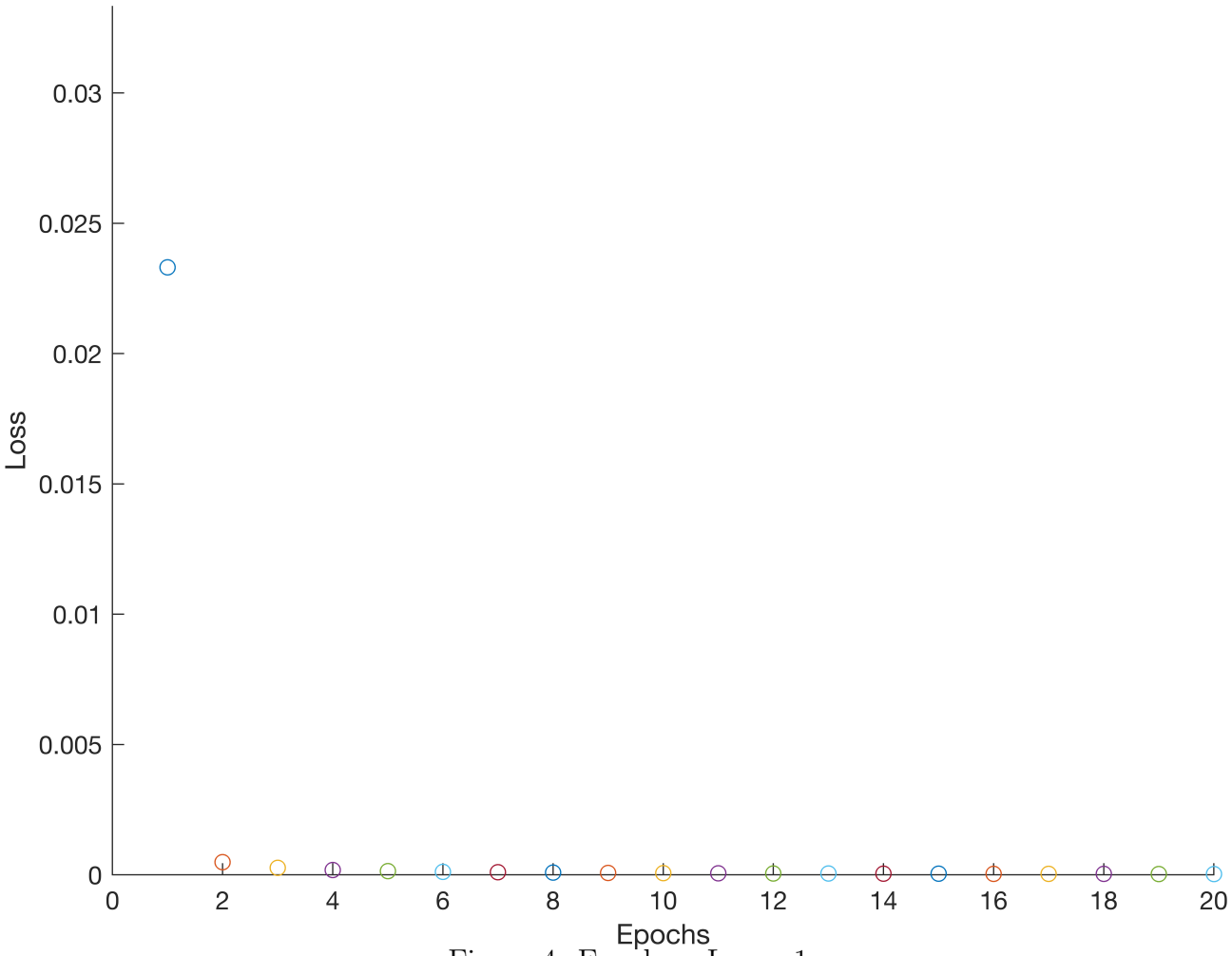

Figure 4: Epoch vs Loss - 1

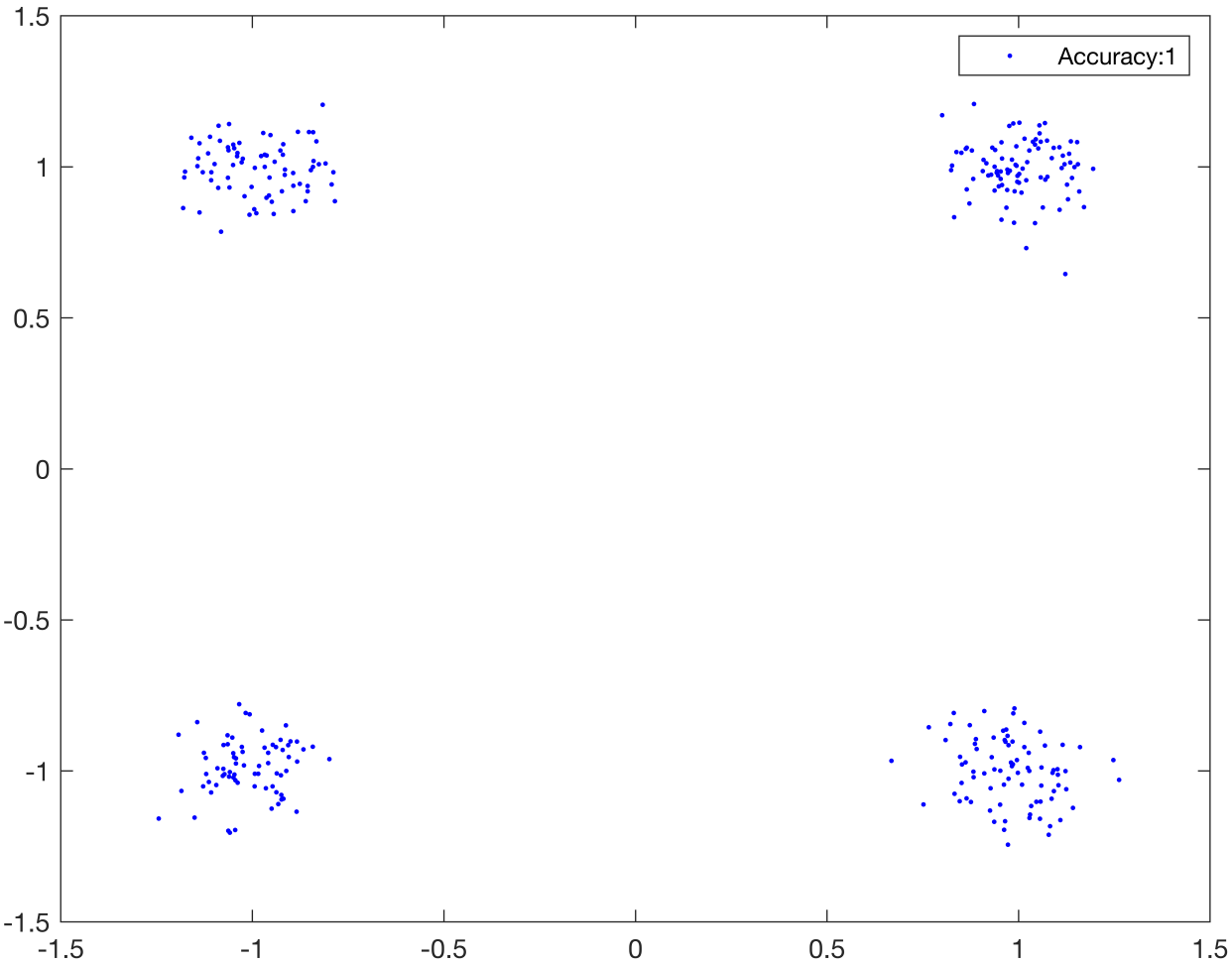

Figure 5: Accuracy on test data - 1

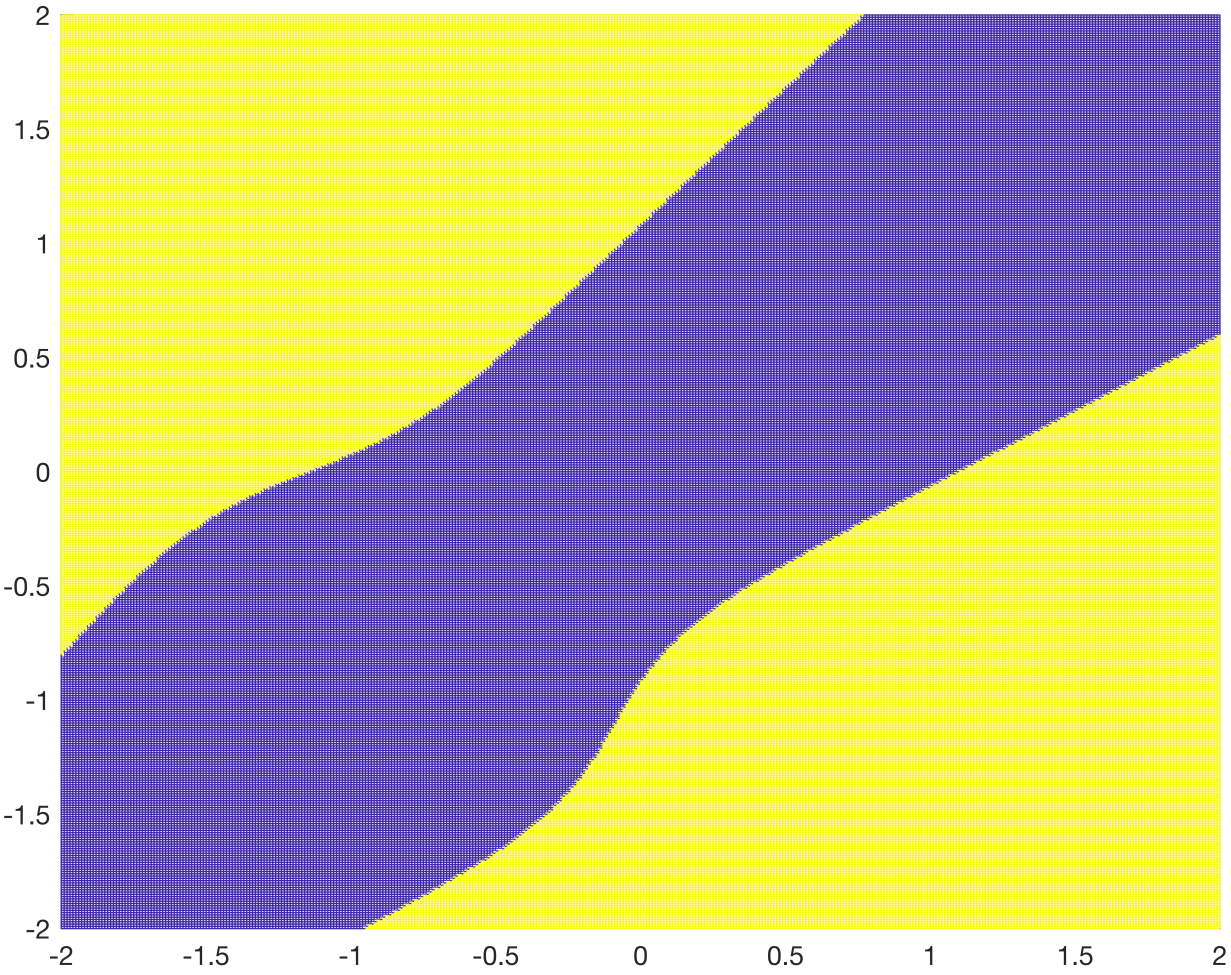

Figure 6: Spatial distribution of the model predictions in the first toy data. Yellow: 1, Violet: 0.

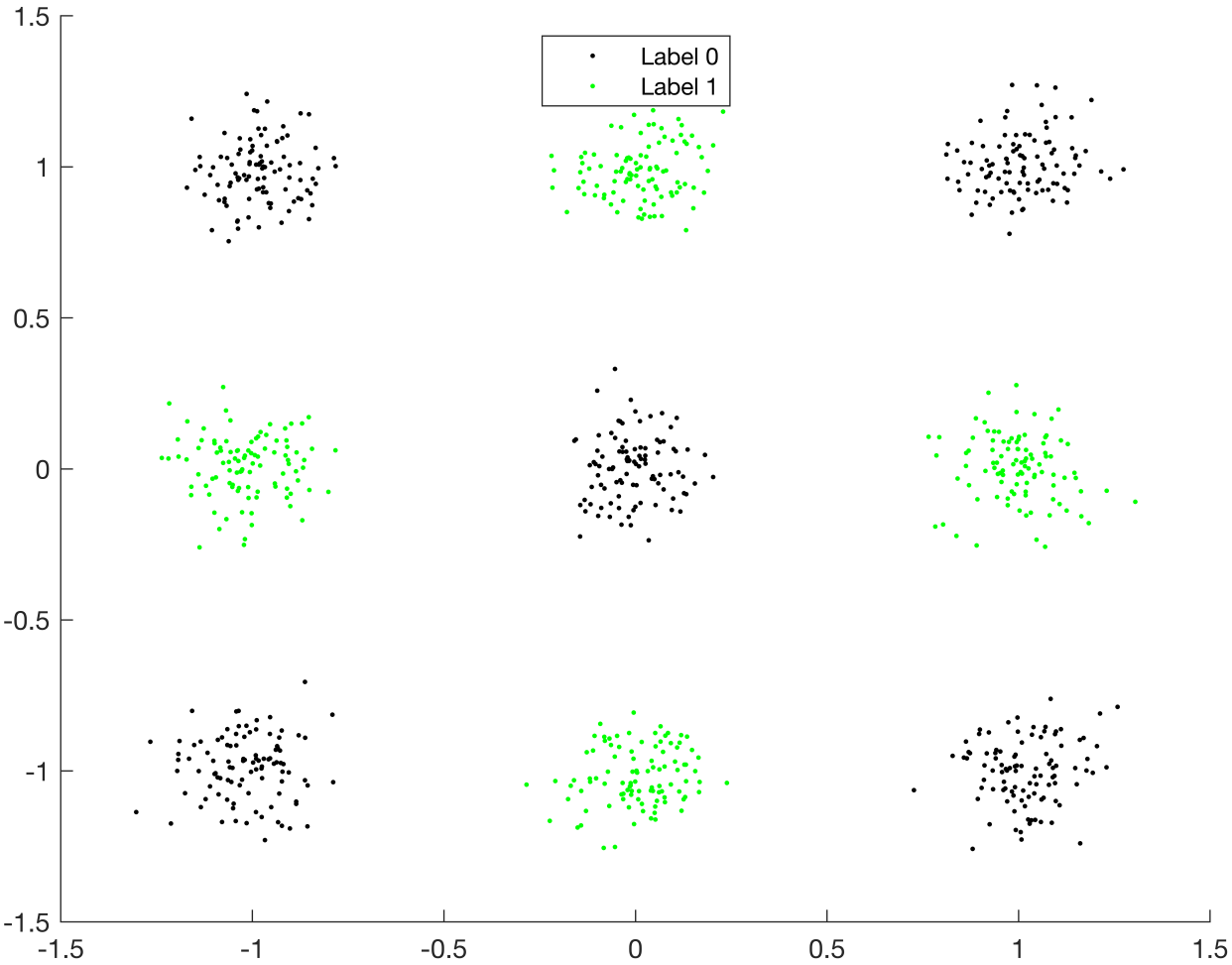

Figure 7: Dataset - 2

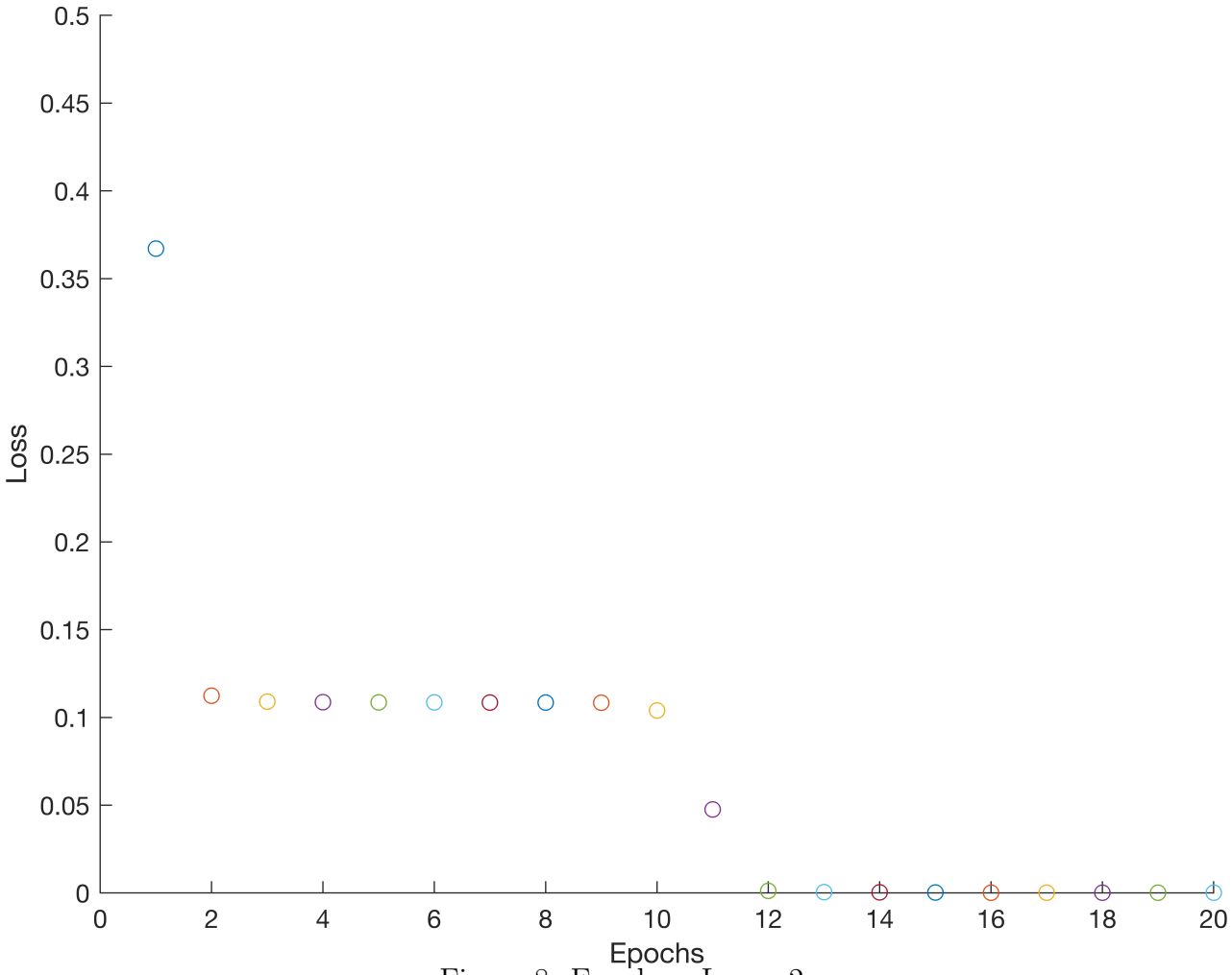

Figure 8: Epoch vs Loss - 2

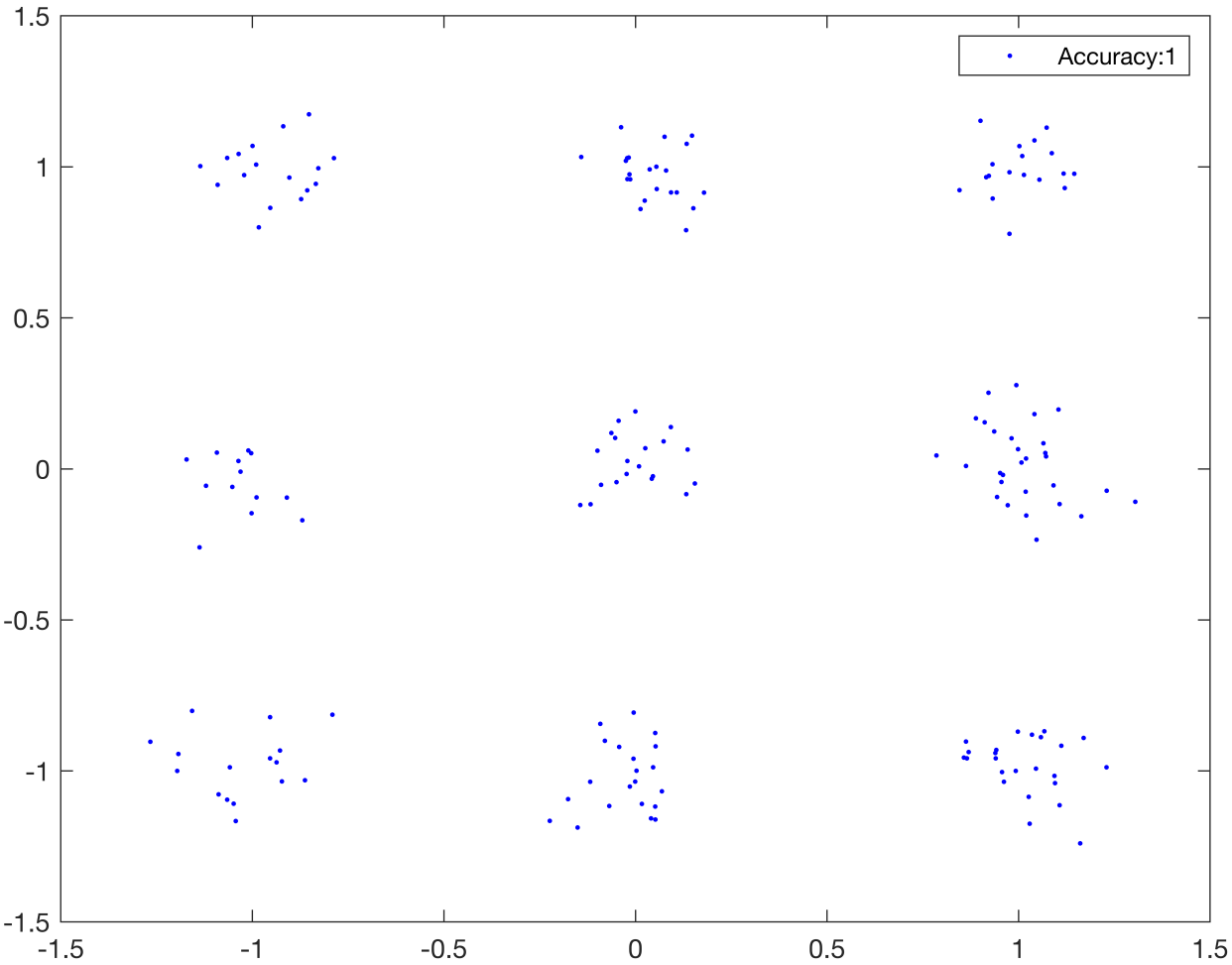

Figure 9: Accuracy on test data - 2

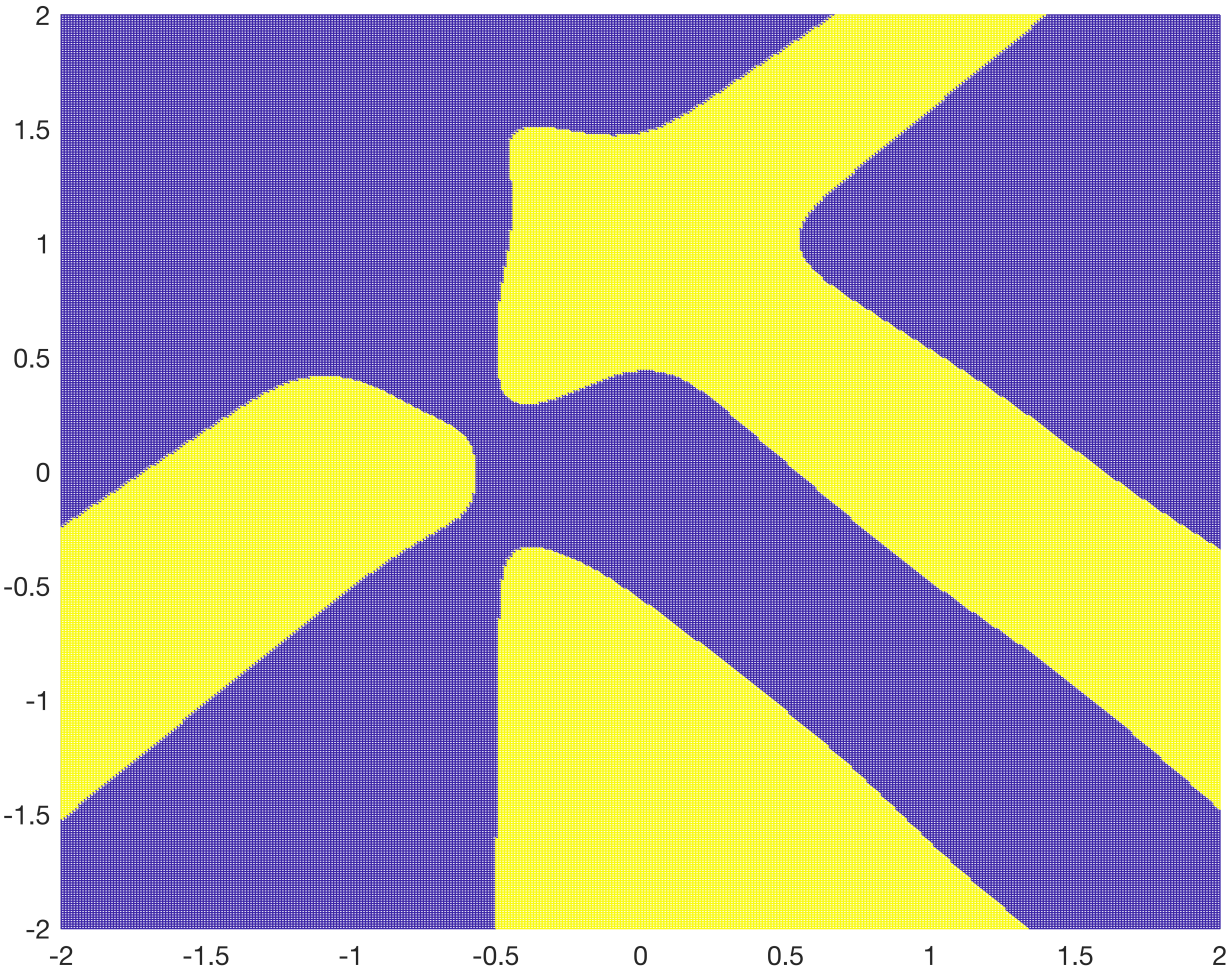

Figure 10: Spatial distribution of the model predictions in the second toy data. Yellow: 1, Violet: 0.

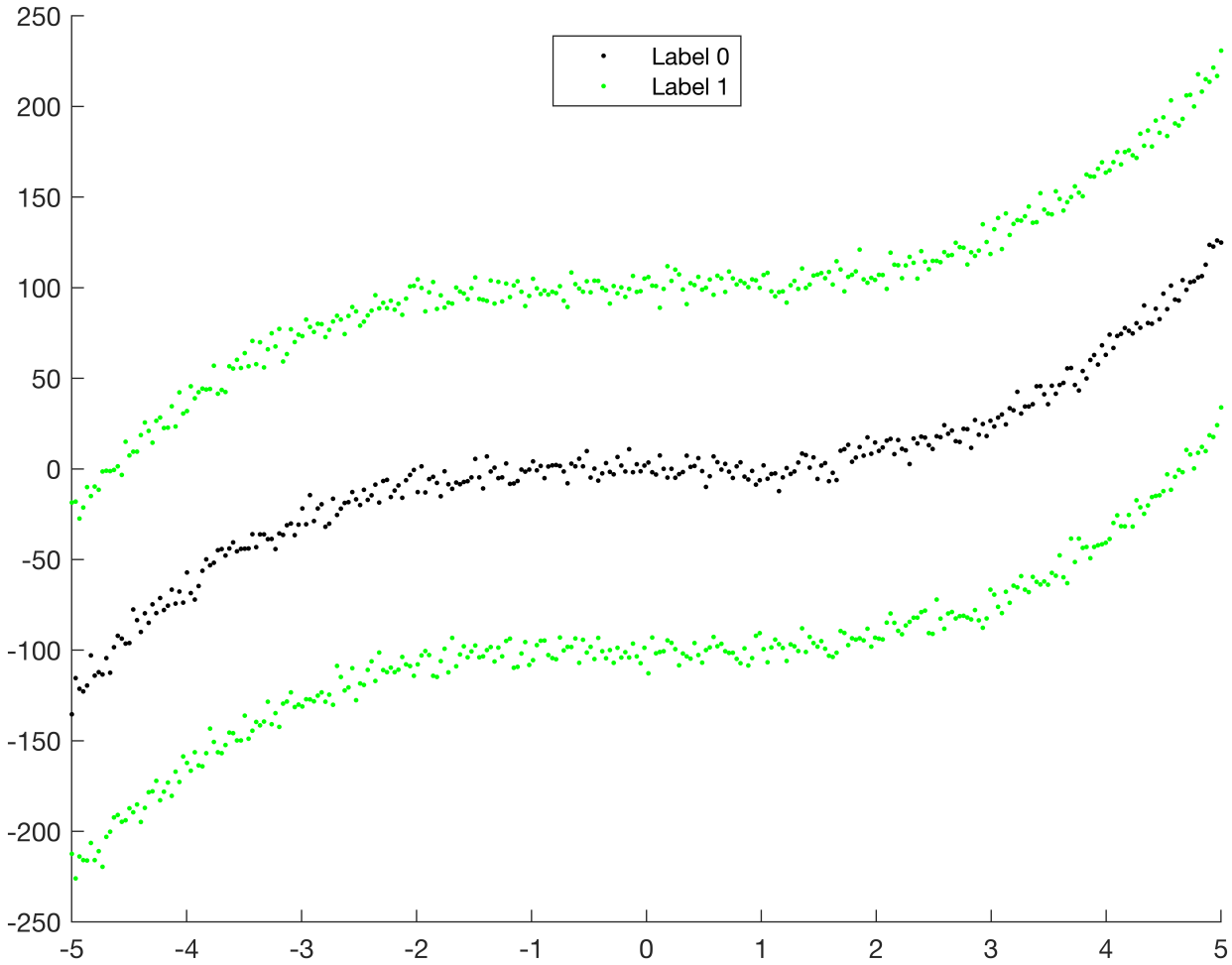

Figure 11: Dataset - 3

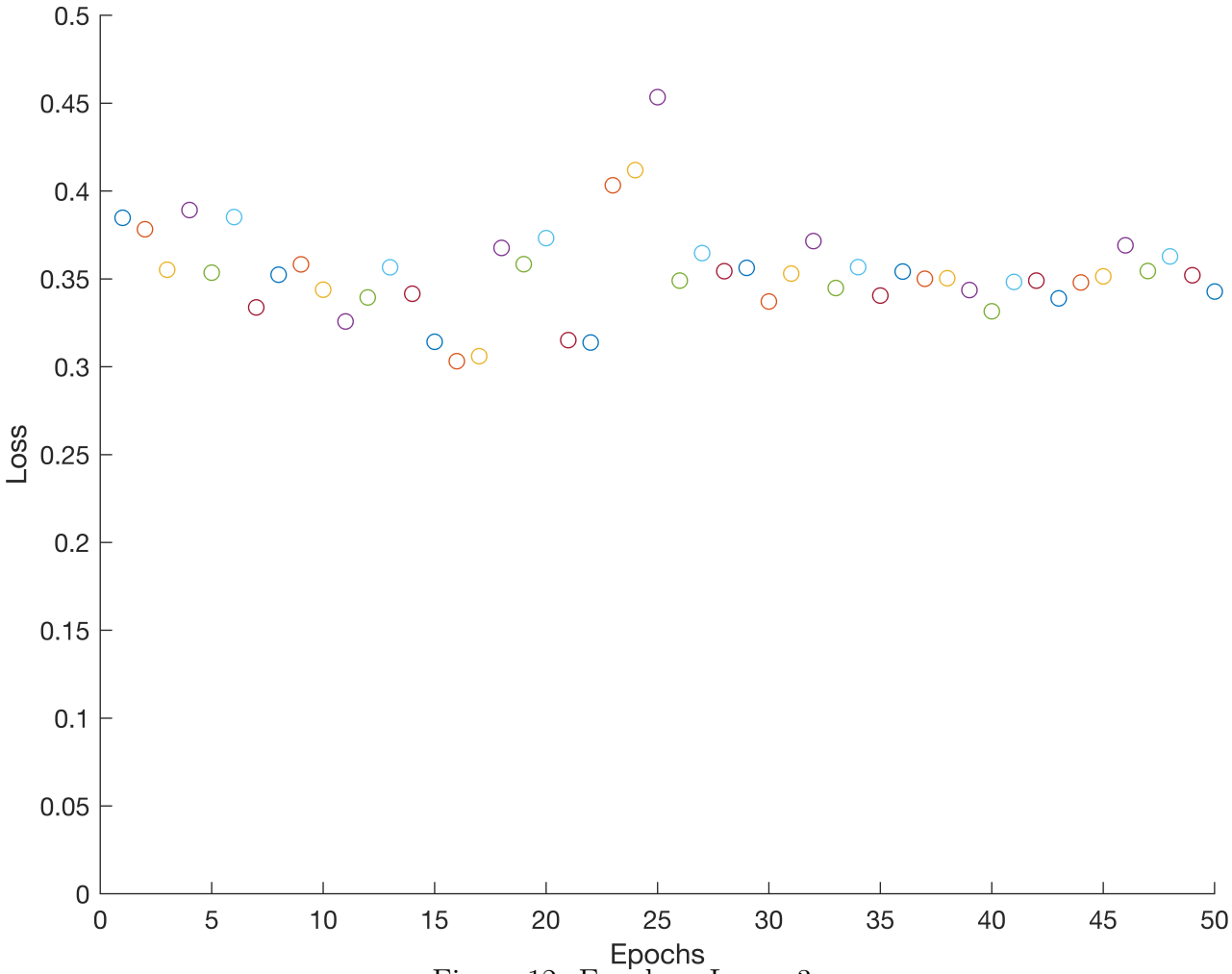

Figure 12: Epoch vs Loss - 3

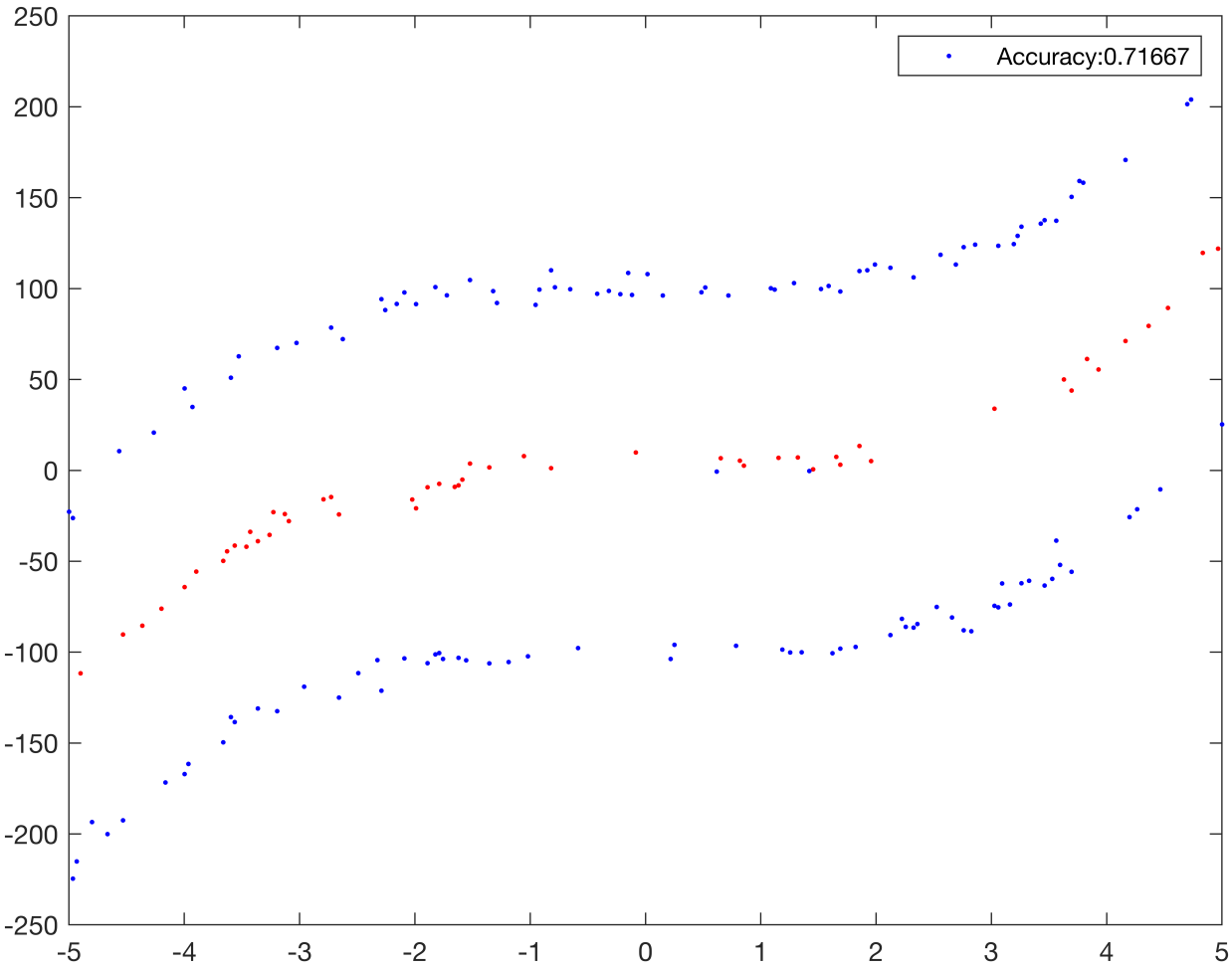

Figure 13: Accuracy on test data - 3

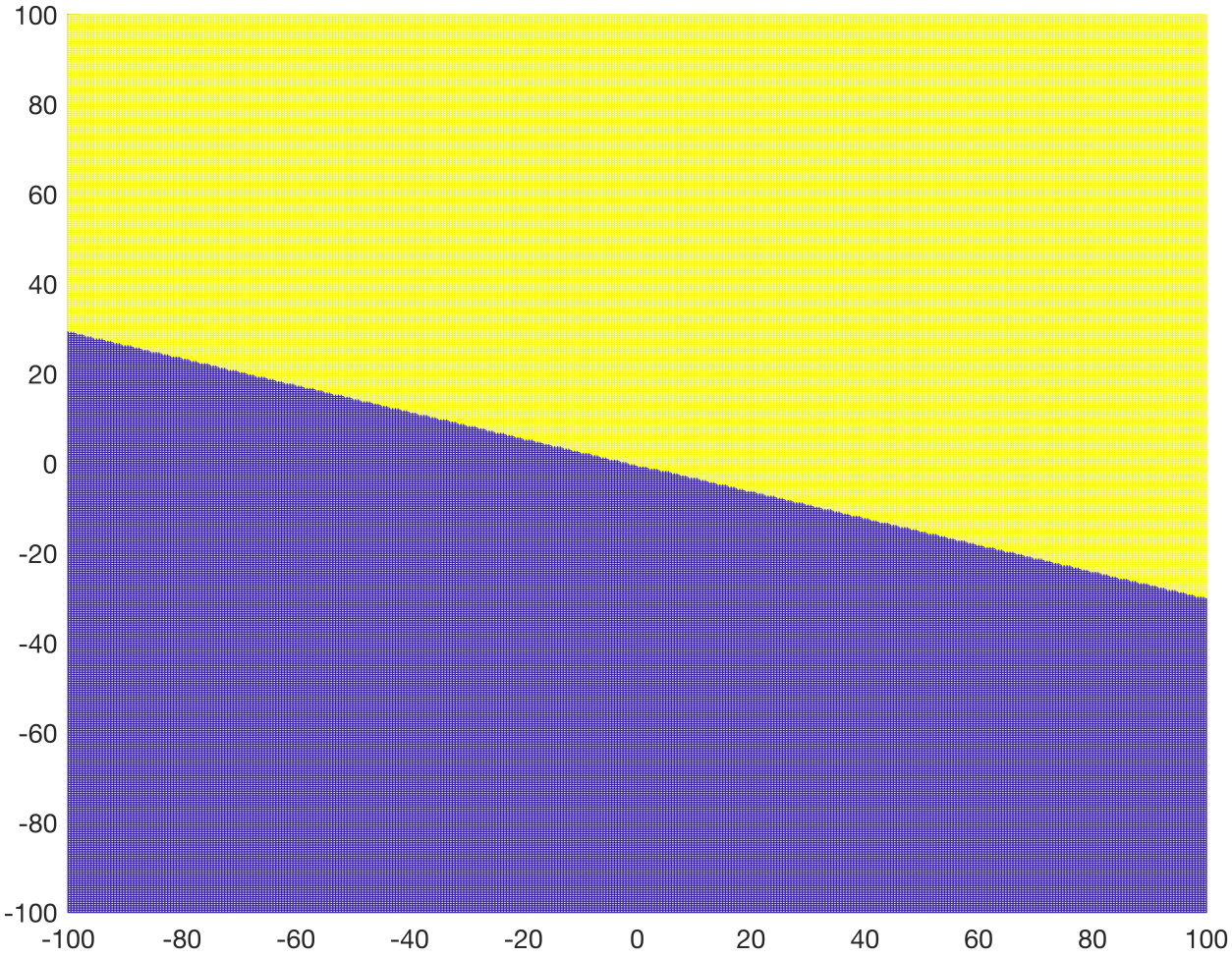

Figure 14: Spatial distribution of the model predictions in the third toy data. Yellow: 1, Violet: 0.

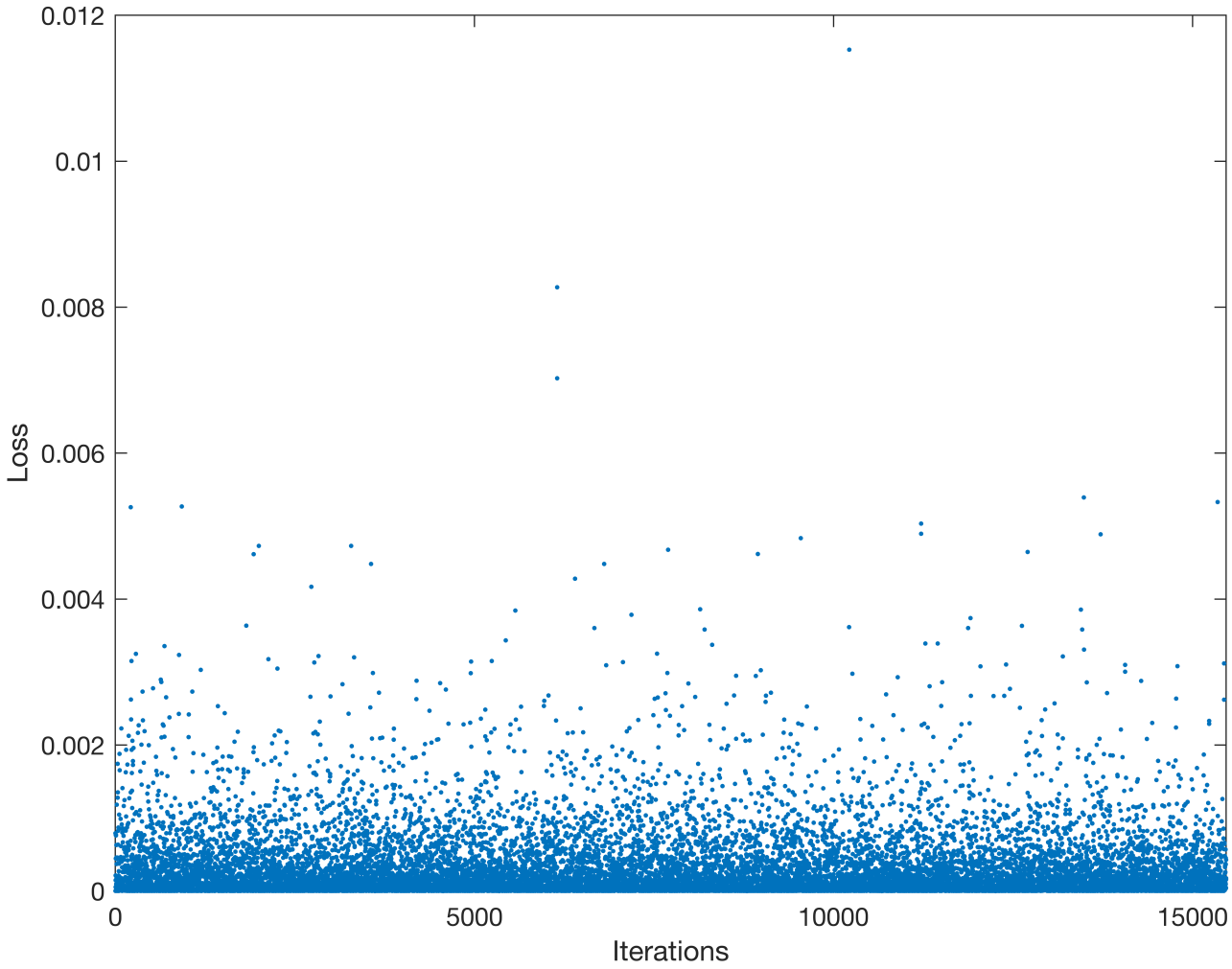

Figure 15: Stock market data - Loss at the last epoch

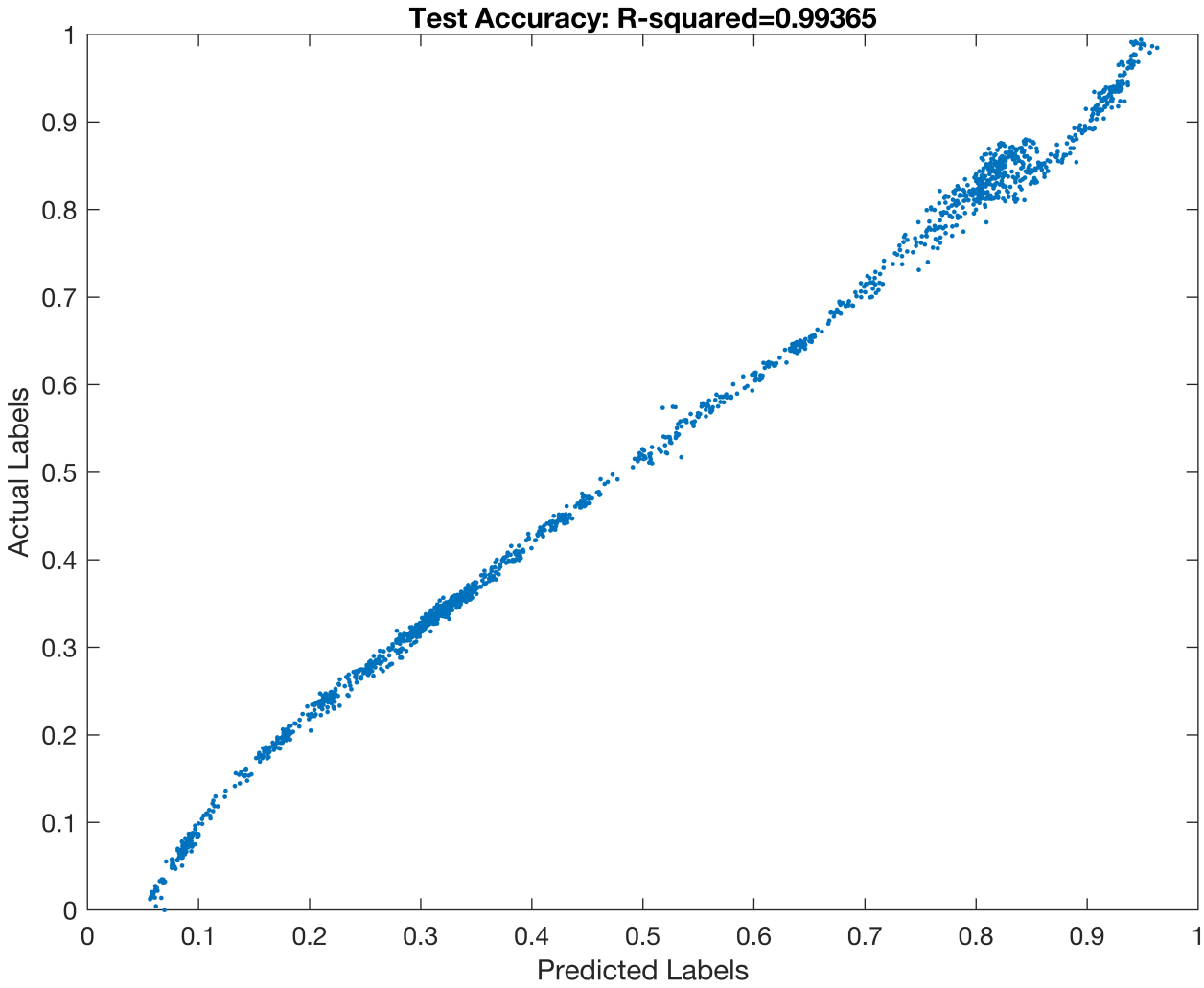

Figure 16: Accuracy of the model on test data