

A Stock Price Prediction Model Based on MLP

Yuxiong Sun^{1,a,*}

¹University of Alberta, 116 St and 85 Ave, Edmonton, Alberta, T6G 2R, Canada

a. yuxiong_sun@163.com

*corresponding author

Abstract: This paper proposes a stock price prediction model based on a multilayer perceptron (MLP) network, using NVDA (NVIDIA Corporation) stock as the research subject. First, the fundamental principles and structure of the MLP network are introduced, along with its advantages in regression tasks, particularly its ability to handle complex nonlinear relationships. Then, historical data of NVDA stock is preprocessed using Python and relevant libraries (such as Pandas, NumPy, Matplotlib, and PyTorch), including data normalization and the division of training and testing sets. This paper provides a descriptive summary of the construction of the MLP model and its parameter setup and optimization. In training, the loss function employed is mean squared error (MSE), and the parameter update rule is the Adam optimizer. Experiments are conducted to evaluate the model's efficiency by analyzing its prediction, and the actual vs. predicted prices are represented visually. The MLP model demonstrates effectiveness in NVDA stock price forecasting tests where it delivers precise price fluctuation pattern recognition. This paper investigates the model prediction accuracy and enhancement potential together with financial market prospects for the MLP model's stock price prediction capabilities.

Keywords: stock price prediction, multilayer perceptron, regression task, data preprocessing, visual analysis

1. Introduction

Stock price prediction stands as a fundamental research subject for investors and researchers because of the complex financial market characteristics and high market variability. Investors who succeed with accurate stock price predictions make better investments that help minimize risks while earning higher profits. The development of efficient stock price prediction methods together with their practical implementation stands crucial for our current era.

With advances in data science and computer technology, machine learning and deep learning methods have become increasingly popular in stock price prediction. Statistical techniques and stochastic process theories were prevalent in early stock price prediction models. While these techniques are capable of modeling stock price movements to some extent, they are usually coupled with complex mathematical computations and are plagued by the inability to handle high-dimensional nonlinear data.

As a classical neural network model, the multilayer perceptron (MLP) is a good representation of complex patterns and relationships in data through multilayer neuron connections and nonlinear

activation functions. Compared with traditional statistical methods, the MLP model is more flexible and adaptive, particularly better at handling high-dimensional nonlinear problems.

This paper proposes an MLP-based stock price forecasting model, using NVDA (NVIDIA Corporation) stock as the research subject. Being a prominent international semiconductor firm, NVIDIA's stock price varies very often and is thus a typical example for time series analysis. This paper initially presents the basic concept and architecture of MLP, and then discusses the preprocessing of data in detail, including data normalization, feature extraction, and split of training and test sets. Then, a forecasting model for NVDA stock prices based on MLP is designed and implemented in Python with the required libraries (i.e., Pandas, NumPy, Matplotlib, and PyTorch).

During model training, the mean squared error (MSE) serves as the loss function, while the Adam optimizer is used for optimizing model parameters. In order to validate the effectiveness of the model, the data set is divided into a training and a test set, and experiments are executed in parallel to test the predictive capability with results inspected visually.

Experimental results show that the MLP model is highly accurate in predicting NVDA stock prices, successfully capturing price fluctuation trends. This paper also presents the model's predictive performance and areas for improvement, suggesting future research directions, including incorporating other deep learning techniques and optimization algorithms to improve prediction accuracy and robustness.

Briefly, the empirical evidence of the applicative value of the MLP model in stock price prediction under this research has provided investors with a real tool of great reference value. As anticipated, it is hoped that this research outcome will be a good reference for researchers and investors in the industry and will drive the development and application of stock price prediction technologies.

1.1. Research Background

MLP stands as a feedforward neural network that applies multiple neuron layers by combining an entry layer and intermediate hidden layers until it reaches the output layer. Each neuron in a layer is fully connected to every neuron in the next layer, enabling complex non-linear data mapping. MLP demonstrates strong performance in regression and classification problems by discovering intricate patterns in the data. Multilayer perceptron operates as a pattern recognition system for past stock price records to generate future price trend forecasts.[1]

2. Introduction to the MLP Model

2.1. Overview of the MLP Network

A Multi-Layer Perceptron (MLP) represents a feedforward neural network that features interconnected neurons organized in three parts consisting of an input layer and one or more hidden layers and an output layer. Neurons in a given layer maintain complete connections with the following layer neurons to create advanced non-linear data transformations. The Multi-Layer Perceptron shows strong performance in regression and classification functions because it accurately detects complex hidden connections among data variables. Using a stock price prediction task MLP extracts historical price features for making future trend forecasts.[2]

2.2. Basic Structure of MLP

The fundamental structure of an MLP consists of the following components:

Input Layer

The neurons in the input layer are equal to the number of input features. Input features in stock price prediction are typically closing prices of previous days. For instance, if closing prices of the previous five days are the input features, five neurons will be in the input layer.

Hidden Layer

The hidden layer comprises multiple neurons, each performing nonlinear transformations through activation functions. The number of hidden layers and the number of neurons per layer are hyperparameters that must be fine-tuned for specific tasks. Common activation functions include Rectified Linear Unit (ReLU), Sigmoid, and Tanh. In this study, the ReLU activation function is selected due to its ability to mitigate the vanishing gradient problem and accelerate network training.

Output Layer

The number of neurons in the output layer depends on the requirements of the prediction task. For stock price forecasting, the output layer typically contains a single neuron that directly outputs the predicted price value. The activation function of the output layer is usually a linear function to accommodate regression tasks.[3]

2.3. Working Mechanism of MLP Units

The working mechanism of an MLP unit can be described in the following steps:

Forward Propagation

The neural network processes input data through successive layers starting from the input to the output layer during forward propagation. Each neuron performs weighted summation of its incoming signals followed by non-linear transformation through activation functions.

Backward Propagation

The model calculates prediction vs actual value errors during backward propagation and uses the chain rule to propagate error data from the back end of the network to the front end. The model learns about the data features through regular weight and bias updates which decrease errors. The Mean Squared Error Loss (MSE) operates as the loss metric while the adaptive Adam optimizer optimizes performance.

Parameter Updating

Each training round causes the model to modify the parameters of all its neurons through backward propagation analysis. By automatically adjusting the learning rate of individual parameters through the Adam optimizer both training performance and model stability show improvement.

These steps let the MLP network learn increasingly complex data patterns and features from the input information to achieve effective stock price prediction. The following sections will present explanations regarding both data preprocessing protocols and the implementation framework of the MLP model alongside its training procedure.

2.4. Application of MLP in Stock Price Prediction

Stock price prediction involves the Multi-Layer Perceptron (MLP) model which extracts valuable patterns from historical data to make price movement predictions. A preliminary stage entails gathering NVDA (NVIDIA Corporation) historical stock price data for normalization purposes. The prediction uses closing price data from the previous five days as its input features together with the closing value of the sixth day as its output for forecasting. The model consists of three layers including the input layer and hidden layers and the output layer where ReLU functions for activation. MSE functions as the loss criterion during training while Adam performs the parameter updates. The experimental tests demonstrate the MLP model successfully detects stock market trend patterns.

3. Quantitative Analysis of MLP

Stock price prediction requires quantitative analysis methods to move forward effectively. Structured analysis enhances predictive capabilities of the model so that it performs better. This research undertook four sequential data processing procedures: data collection followed by data cleaning and then data normalization and feature engineering.

Historical stock price data from NVDA (NVIDIA Corporation) was obtained with key information points including date, opening price, highest price, lowest price, closing price as well as trading volume. The processing success depends heavily on maintaining complete and accurate data records.

During data cleaning operations, researchers need to remove both data gaps and exceptional values throughout the dataset. The procedure ensures data quality and consistency through data integrity checks that lead to either imputation or removal of missing values and handling of abnormal data points.[4]

3.1. Data Reading

The first step involves reading stock data from its CSV format. The Pandas library implements `read_csv` to perform this import operation. The `file_path` variable contains the path address of the dataset:

```
file_path = './NVDA.csv'
data = pd.read_csv(file_path)
```

3.2. Creating Features and Labels

The `create_features_and_labels` function accepts past closing prices from `n` days to form features while assigning the day `n+1` closing price as the label value. The variable `n_days` determines the window dimensions used for the features. Finally the function produces a feature matrix `X` along with a label vector `y`. This function is then used to generate the feature matrix `X` and label vector `y`.

```
def create_features_and_labels(data, n_days=5):
    for i in range(len(data) - n_days):
        features.append(data[i:i+n_days].values)
        labels.append(data[i+n_days])
    return np.array(features), np.array(labels)
n_days = 5
X, y = create_features_and_labels(data['Close'], n_days)
```

3.3. Data Splitting

The dataset is split into a training set and a test set in an 8:2 ratio. The input data is reshaped to meet the input requirements of Keras, which expects a three-dimensional tensor: [number of samples, time steps, number of features].

```
# Splitting training and test sets
train_size = int(len(X) * 0.8)
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]
# Reshaping input data
X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
X_test = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))
```

4. Stock Price Prediction Model Based on MLP

The study builds a stock price prediction system through the implementation of a multilayer perceptron (MLP). Historical stock data from NVDA (NVIDIA Corporation) receives training and testing in the model to estimate upcoming stock price directions. The following sections break down important code information through detailed explanations.

4.1. Definition of the MLP Model

The stock price prediction model uses an MLP structure. The feedforward neural network MLP consists of three components: an input layer followed by one or multiple hidden layers which lead to an output layer. The model organization follows this structure:

```
# 2. Constructing the MLP model
model.add(Flatten(input_shape=(n_days, 1)))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(1))
# Compiling the model
model.compile(optimizer='adam', loss='mean_squared_error')
```

The Sequential model in Keras is a type of model that stacks layers in sequence, making it suitable for processing data step by step from input to output. The Flatten layer transforms the input data from a two-dimensional tensor (n_days, 1) into a one-dimensional tensor (n_days,), allowing it to connect to fully connected (Dense) layers. Here, input_shape=(n_days, 1) specifies that the input features are the closing prices of the past n days. The Dense layer is a fully connected layer. The first hidden layer consists of 64 neurons, each using the ReLU (Rectified Linear Unit) activation function. ReLU helps mitigate the vanishing gradient problem and improves training speed. The second hidden layer contains 32 neurons with the same activation function. Increasing the number of hidden layers and neurons enables the model to learn more complex data patterns. The output layer consists of a single neuron without an activation function since this is a regression task aiming to predict a continuous value (the stock price for the next day)[5].

4.2. Data Loading and Preprocessing

First, the historical stock data of NVDA is loaded and preprocessed. The preprocessing steps include converting the date column to datetime format, setting the date as the index, selecting the closing price as the feature, and creating feature-label pairs.

```
# 1. Data loading and preprocessing
file_path = './NVDA.csv'
data = pd.read_csv(file_path)
# Check basic data information
print(data.head())
print(data.info())
# Select closing price as the feature
data['Date'] = pd.to_datetime(data['Date'])
data.set_index('Date', inplace=True)
data = data[['Close']]
```

4.3. Model Training

The MLP model is trained using the training dataset. The training process involves forward propagation to compute outputs and backpropagation to adjust model parameters based on the loss function.

```
epochs=50: The model is trained for 50 epochs.  
validation_split=0.2: 20% of the training data is used as the validation set.  
batch_size=32: The model updates weights using batches of 32 samples.  
history = model.fit(X_train, y_train, epochs=50, validation_split=0.2, batch_size=32)
```

4.4. Data Transformation and Model Training

The performance of the model is evaluated on the test data, with the loss value computed and printed.

```
# Model Evaluation  
loss = model.evaluate(X_test, y_test)  
print(f'Test Loss: {loss}')
```

4.5. Model prediction

The trained model is used to predict the test data, and the predicted results are compared with the actual values. A visualization is created for analysis.

```
# 5. Prediction  
# Predict using the test dataset  
predictions = model.predict(X_test)  
# Visualizing the Prediction Results  
plt.figure(figsize=(10, 6))  
plt.plot(y_test, label='True Prices')  
plt.plot(predictions, label='Predicted Prices')  
plt.legend()  
plt.show()
```

4.6. Visualization of the Training Process

A graph illustrating the changes in loss values during training and validation is plotted to intuitively display the model's training performance.

```
# Visualizing the Training Process  
plt.figure(figsize=(10, 6))  
plt.plot(y_test, label='True Prices')  
plt.plot(predictions, label='Predicted Prices')  
plt.title('MLP-Based Prediction of NVIDIA Stock - True Prices vs. Predicted Prices',  
fontproperties=font_prop)  
plt.xlabel('Time', fontproperties=font_prop)  
plt.ylabel('Price', fontproperties=font_prop)  
plt.legend(prop=font_prop)  
plt.show(), # See Figure 1
```

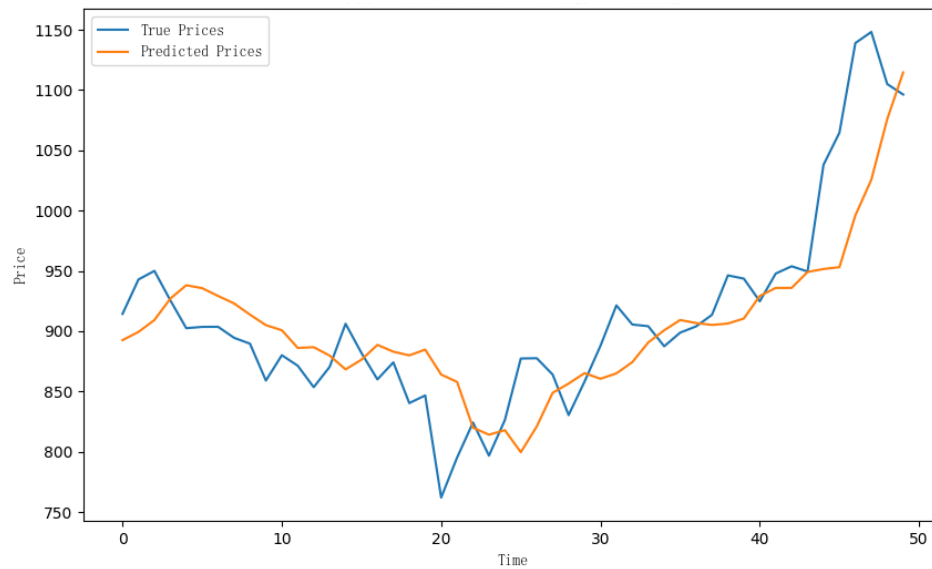


Figure 1: MLP-Based Prediction of NVIDIA Stock - True Prices vs. Predicted Prices

5. Conclusion

Model Construction: A multilayer perceptron (MLP) model with two hidden layers was built using Keras.

Data Reading and Preprocessing: Data was read from a CSV file, with dates converted to datetime format, the closing price selected as the feature, and the dataset split into training and test sets.

Model Training: The model was trained using the training dataset, utilizing the Adam optimizer and The Mean Squared Error (MSE) loss function.

Model Evaluation: The model's performance was evaluated on the test dataset, with the loss value printed.

Model Prediction and Result Visualization: Predictions were generated for the test dataset, and a comparison between actual and predicted values was visualized using plots.

Visualization of the Training Process: A graph depicting the changes in loss values during training and validation was plotted to illustrate the training progress.

References

- [1] Ma, J. (2024). Research on regression and classification in stock price trend prediction. *Computer Knowledge and Technology*, 20(12), 12–14, 23.
- [2] Yan, W. (2024). Application of machine learning in stock prediction. *Information Systems Engineering*, 2024(04), 40–43.
- [3] Kong, Y., Huang, Z., Deng, H., et al. (2024). Design of a quantitative trading algorithm based on deep reinforcement learning. *Journal of Nanchang University (Natural Science Edition)*, 48(01), 24–29, 35.
- [4] Sun, N., Zhou, S., & Pan, Z. (2023). Multi-feature stock price prediction based on the XGBoost-LSTM model. *Mathematical Modeling and Its Applications*, 12(04), 32–39.
- [5] Fan, H., Zhu, Y., & Li, J. (2024). Stock prediction method based on attention mechanism and feature fusion. *Journal of Shandong Institute of Business and Technology*, 38(01), 57–68, 76.