Predictive Trading with Neural Trader App: A Machine Learning Approach to Trading Cryptos

User Guide for the Neural Trader Python App

Intro to the Neural Trader Python Application

Neural Trader App is a powerful tool that helps you trade Bitcoin on the Kucoin exchange using machine learning. The app connects to the Kucoin API using the ccxt library and retrieves real-time data on the current market price of Bitcoin. It then uses a neural network model to predict whether the market will trend upwards or downwards, and based on this prediction, the app will either buy or sell a small amount of Bitcoin.

The app is designed to be easy to use, with a straightforward interface and user-friendly controls. It begins by importing several libraries that it will use throughout its execution, including the ccxt library for interacting with the Kucoin API, the time library for adding delays, the numpy library for numerical computations, and the scikit-learn library for machine learning. The app also defines several variables such as the exchange name, API key and secret, symbol to trade, and the amount of Bitcoin to trade.

Once set up, the app enters an infinite loop to continuously trade Bitcoin. Within this loop, it retrieves the current ticker information for the symbol, gets the current bid and ask prices, and uses a neural network model to predict the market direction. If the prediction is bullish, the app will place a buy order at the ask price and add the premium to the price. If the prediction is bearish, the app will place a sell order at the bid price and subtract the premium from the price. The app also includes a try-except block to handle any errors that may occur during the trading process, such as a failed trade or an API error. If an error does occur, the app will log the error and continue with the next iteration of the loop.

The app uses a neural network model to predict whether the market will trend upwards or downwards. The neural network model is trained on historical data, using a supervised learning technique called regression analysis. This allows the model to learn the relationship between various features (such as the price of Bitcoin, the volume of trades, and the overall market sentiment) and the direction of the market (bullish or bearish).

Based on the prediction made by the neural network model, the app will either buy or sell a small amount of Bitcoin. If the prediction is bullish (i.e., the market is expected to trend upwards), the app will place a buy order at the ask price and add the premium to the price. If the prediction is bearish (i.e., the market is expected to trend downwards), the app will place a sell order at the bid price and subtract the premium from the price. If the market is bullish it will price a 0 if it is bearish it'll print a 1. The script uses these to decide to buy or sell if no open orders exist.

In addition, the app saves the trained neural network model and the best values for the hyperparameters to a file for later use. This allows the app to continue using the same model and hyperparameters on future runs without the need to retrain the model each time. This is useful for ensuring consistent performance and minimizing the time and resources needed for training the model.

Overall, Neural Trader App is a powerful and reliable tool for trading Bitcoin on the Kucoin exchange, using advanced machine learning techniques to make informed decisions and maximize returns.

This application can be used for 90 exchanges including Coinbase, Coinbase Pro, etc. You may need to changes some of the ccxt functions if the exchange uses something different. You can also use any cryptocurrency pair you'd like to use, refer to the ccxt documentation to learn how to print a response of exchanges and the cryptocurrencies you can trade.

How to Setup

If the bot is going to trade the BTC/USDT pair, you can use any pair, make sure the have the second part of the pair or the first part available. It needs to have USDT in the account in order to initiate a buy order of bitcoin and if you hold bitcoin it will sell it instead when you start the script. The script could be containerized into functions. I have purposely not done that to continue to experiment with the script, this is my personal opinion, you are welcome to containerized functions.

You will first need to run the learn.py file. This will create a file called model.pkl this is the neural network model that we will use to begin the main.py to use this model and use new streaming information download it in batches make a decision based on both the new incoming information it learned about and the model that it saved from its previous learning sessions.

Model is being trained on a 15 minute time frame downloaded in streaming batches using the ccxt libraries access to the exchanges OHLVC.

After running the train.py file for about an hour you have enough data to begin using the main.py file from now on forward unless you want to create a new model based on a new timeframe. I don't know if this is necessary but this is how you should go about it when training on new time frames.

These are the time frame you can use for KuCoin:
{'1m': '1min', '3m': '3min', '5m': '5min', '15m': '15min', '30m': '30min', '1h': '1hour', '2h': '2hour', '4h': '4hour', '6h': '6hour', '8h': '8hour', '12h': '12hour', '1d': '1day', '1w': ‘1week’}

User Guide for learn.py

This script is a trading bot that uses a machine learning model to predict the direction of the market (bullish or bearish) and makes trades accordingly.

The script begins by importing several modules:

- ccxt is a library for interacting with cryptocurrency exchanges
- time is a built-in Python library for working with time
- numpy is a library for working with arrays and numerical data
- sklearn is a library for machine learning tasks
- datetime is a built-in Python library for working with dates and times
- pickle is a built-in Python library for serializing and deserializing objects
- json is a built-in Python library for working with JSON data

Next, the script sets some variables:

- exchange_name is the name of the exchange that the script will use. In this case, it is set to 'kucoin'.
- exchange is an instance of the ccxt.Exchange class for the specified exchange.
- exchange.set_sandbox_mode is a method that sets the sandbox mode for the exchange. Sandbox mode is a simulated trading environment that allows you to test the script without actually making trades. In this case, sandbox mode is disabled.
- exchange.apiKey, exchange.secret, and exchange.password are the API keys that the script will use to authenticate with the exchange.
- symbol is the symbol that the script will trade on the exchange. In this case, it is set to 'BTC/USDT'.
- amount is the amount of BTC that the script will trade. In this case, it is set to 0.005 BTC.
- fee is the fee that the exchange charges for each trade. In this case, it is set to 0.001 BTC.
- premium is the premium that the script will add to the sell order. In this case, it is set to 0.002 BTC plus the fee.

The script then enters an infinite loop with a try/except block. The try block contains the main logic for the script, and the except block handles any errors that might occur.

Inside the try block, the script fetches the OHLCV (Open-High-Low-Close-Volume) data for the specified symbol and time frame. This data is used to train the machine learning model.
Next, the script defines a function called predict_market_direction that takes the OHLCV data as an input and returns a prediction for the market direction (0 for bullish, 1 for bearish).

This function does the following:
- Extracts the features and target variable from the data
- Specifies the values for the hyperparameters that the script wants to tune
- Creates a neural network model using the MLPClassifier class from sklearn
- Uses the GridSearchCV class to search through combinations of the hyperparameters and find the best combination
- Saves the best hyperparameters to a JSON file
- Updates the model with the best hyperparameters
- Trains the model using the features and target
- Saves the trained model to a file using pickle
- Makes a prediction using the trained model and the most recent OHLCV data

After the predict_market_direction function is defined, the script enters another infinite loop.

Inside this loop, the script does the following:

- Fetches the current ticker information for the symbol
- Checks the following:
- The current bid and ask prices for the symbol
- The current market direction prediction
- If the prediction is 0 (bullish), the script creates a sell order with a price that is premium higher than the current bid price.
- If the prediction is 1 (bearish), the script creates a buy order with a price that is premium lower than the current ask price.

After the trade is made, the script sleeps for 15 minutes before starting the loop again. This is to give the market time to move before making another trade.
The script continues to run indefinitely, making trades and retraining the model as necessary. If an error occurs during the execution of the script, the except block will handle the error and print an error message to the console.

User Guide for main.py

This script is a trading bot that uses machine learning to predict the direction of the market and makes trades accordingly on the KuCoin exchange. The bot is built using the ccxt library, which allows it to interact with the KuCoin API, and the sklearn library, which provides machine learning functionality.

The script starts by importing the necessary libraries and setting some initial variables, including the exchange name, symbol to trade, and amount of the trade. It then instantiates the Exchange class from the ccxt library and sets the sandbox mode to False, which means it will be making real trades on the KuCoin exchange. It also sets the API keys for the KuCoin account that will be used for trading.

The script then enters an infinite loop, in which it continuously fetches data from the exchange, predicts the direction of the market using a machine learning model, and places trades accordingly. The model is trained and updated using the data from the exchange, and the prediction is made based on the most recent data. If the prediction is bullish (indicating an upward trend in the market), the bot will place a limit buy order at the midpoint between the current bid and ask prices. If the prediction is bearish (indicating a downward trend in the market), the bot will place a limit sell order at a price that is slightly higher than the current bid price, with a premium added to account for the trading fee.

The script also includes some error handling and sleep statements to ensure that it continues to run smoothly and does not make too many requests to the exchange in a short period of time.

The first part of the script imports the necessary libraries:

- The ccxt library is used to interact with various cryptocurrency exchanges. In this case, it is used to communicate with the KuCoin exchange.
- The time library is used to pause the script for a certain amount of time using the sleep() function.
- The numpy library is used to work with arrays and perform mathematical operations on them.
- The sklearn library contains a variety of machine learning algorithms and tools, including the MLPClassifier, which is a neural network classifier that is used in this script to predict the direction of the market.
- The datetime library is used to get the current date and time.
- The pickle library is used to save and load the machine learning model to and from a file.

The next part of the script sets some initial variables

- The exchange_name variable is set to 'kucoin', which tells the script to use the KuCoin exchange.
- The exchange variable is set to an instance of the ccxt.kucoin class, which allows the script to communicate with the KuCoin API.
- The set_sandbox_mode() method is called on the exchange object and passed the value False, which means the bot will be making real trades on the KuCoin exchange.
- The apiKey, secret, and password properties of the exchange object are set to the API keys for the KuCoin account that will be used for trading.
- The symbol variable is set to 'BTC/USDT', which is the symbol for the BTC/USDT pair on the KuCoin exchange. This is the pair that the bot will be trading.
- The amount variable is set to 0.005, which is the amount of BTC that the bot will be trading each time it places an order.
- The fee variable is set to 0.001, which is the fee that KuCoin charges for each trade.
- The premium variable is set to .002, which is the profit that bot will take plus the fee.

The next part of the script enters an infinite loop, in which it continuously fetches data from the exchange and makes predictions about the market direction using a machine learning model:

- The script enters an infinite loop using the while True statement. This means that the script will run indefinitely until it is manually stopped.
- The try and except statements are used to handle any errors that may occur when the script is running. If an error occurs, the script will pause for 60 seconds and then continue running.
- The fetch_ohlcv() method of the exchange object is called to fetch data from the exchange. The 'BTC/USDT' argument specifies the symbol for the BTC/USDT pair on the KuCoin exchange, and the '15m' argument specifies the time frame for the data. The fetch_ohlcv() method returns a list of "candlestick" data points, which contain information about the open, high, low, close, and volume of the BTC/USDT pair over a 15-minute period.
- The model_file variable is set to 'model.pkl', which is the name of the file that will be used to save and load the machine learning model.
- The predict_market_direction() function is defined. This function takes two arguments: data, which is the list of candlestick data points that was fetched from the exchange, and model_file, which is the name of the file that will be used to save and load the model.
- The features variable is set to an array of the open, high, low, and close values from the data argument. The target variable

The next part of the script enters an infinite loop, in which it continuously fetches data from the exchange and makes predictions about the market direction using a machine learning model:

- The script enters an infinite loop using the while True statement. This means that the script will run indefinitely until it is manually stopped.
- The try and except statements are used to handle any errors that may occur when the script is running. If an error occurs, the script will pause for 60 seconds and then continue running.
- The fetch_ohlcv() method of the exchange object is called to fetch data from the exchange. The 'BTC/USDT' argument specifies the symbol for the BTC/USDT pair on the

KuCoin exchange, and the '15m' argument specifies the time frame for the data. The fetch_ohlcv() method returns a list of "candlestick" data points, which contain information about the open, high, low, close, and volume of the BTC/USDT pair over a 15-minute period.

- The model_file variable is set to 'model.pkl', which is the name of the file that will be used to save and load the machine learning model.
- The predict_market_direction() function is defined. This function takes two arguments: data, which is the list of candlestick data points that was fetched from the exchange, and model_file, which is the name of the file that will be used to save and load the model.
- The features variable is set to an array of the open, high, low, and close values from the data argument. The target variable

The predict_market_direction() function loads the machine learning model from the model_file, updates it with the new data using the partial_fit() method, and then saves the updated model back to the model_file. It then uses the updated model to make a prediction about the market direction based on the most recent data point in the data argument. If the prediction is 0, the market is considered to be bullish (indicating an upward trend), and if the prediction is 1, the market is considered to be bearish (indicating a downward trend).

The script then enters another infinite loop, in which it continuously checks the current market conditions and makes trades based on the prediction of the market direction.

- The script enters another infinite loop using the while True statement. This means that the bot will continuously check the market conditions and make trades until the script is manually stopped.
- The fetch_ticker() method of the exchange object is called to fetch the current ticker information for the BTC/USDT

The script then checks if there are any open orders (orders that have been placed but have not yet been filled or cancelled). If there are no open orders, the script will place a limit buy order if the market is predicted to be bullish, or a limit sell order if the market is predicted to be bearish. The limit buy order will be placed at the midpoint between the current bid and ask prices, and the limit sell order will be placed at a price that is slightly higher than the current bid price, with a premium added to account for the trading fee.

If there are open orders, the script will cancel them. This is to ensure that there are no conflicting orders that could cause problems when the bot is making trades.

Finally, the script includes some error handling and sleep statements to ensure that it continues to run smoothly and does not make too many requests to the exchange in a short period of time.