Ethereum Price Prediction

Sommario

1. Obiettivi della ricerca

L’obiettivo della ricerca è quello di fare una previsione del prezzo di Ethereum in USD tramite tecniche di machine learning. Per effettuare una predizione su una serie temporale implementeremo una Recurrent Neural Network (RNN) con LSTM (Long Short-Term Memory).

1.1 Stato dell’arte

Ethereum è spesso indicato come la seconda criptovaluta più popolare, dopo Bitcoin. Ma a differenza del Bitcoin, Ethereum vuole essere molto più di un semplice mezzo di scambio o una riserva di valore.

Ethereum si definisce invece una rete informatica decentralizzata costruita sulla tecnologia blockchain. Esso può essere usato per comprare e vendere beni e servizi, come Bitcoin. Ha anche visto rapidi aumenti di prezzo negli ultimi anni, rendendolo di fatto un investimento speculativo.

Ma ciò che è unico di Ethereum è che gli utenti possono costruire applicazioni che "girano" sulla blockchain come il software "gira" su un computer. Queste applicazioni possono memorizzare e trasferire dati personali o gestire complesse transazioni finanziarie.

Una serie storica in statistica rappresenta un gruppo di variabili casuali sistemate in base al tempo. Lo studio delle serie temporali è un argomento molto diffuso e discusso anche perché in questo tipo di ricerche la quantità e mole di dati può essere esorbitante. Queste analisi sulle serie temporali si possono suddividere in diversi ambiti in base a che tipo di dati il nostro studio si riferisce; un’altro esempio oltre al nostro di tipo economico è quello delle previsioni meteo. Nel nostro caso specifico ci focalizzeremo sull’analisi dei mercati finanziari.

2. Strumenti utilizzati

Gli strumenti utilizzati per questo progetto sono cosi divisi:

Strumenti di codifica:
- Anaconda
- Jupyter Lab

Strumenti per i dati di trading:
- BitMEX (Testnet)

Strumenti di informazione:
- Documentazioni scientifiche sugli argomenti da noi trattati
  - “Predicting the Price of Bitcoin Using Machine Learning” - Sean McNally, Jason Roche, Simon Caton
  - “Real-Time Prediction of BITCOIN Price using Machine Learning Techniques and Public Sentiment Analysis” - S M Raju, Ali Mohammad Tarif

3. Recupero dei dati e costruzione del dataset iniziale

Per recuperare questi dati, dovevamo trovare un servizio online che ci desse tutti i dati storici della cryptovaluta. Dopo aver ricercato a fondo, ci siamo imbattuti su BitMEX. Quest'ultimo ha delle API molto accessibili e ha la possibilità di scaricare dati storici con vari timeframe, anche a 1 minuto.

“BitMEX è una piattaforma di trading che offre agli investitori l'accesso ai mercati finanziari globali usando solo Bitcoin. BitMEX è costruita da professionisti della finanza con oltre 40 anni di esperienza combinata e offre un'API completa e strumenti di supporto.”

Il dataset è stato costruito andando a scaricare, tramite le API di BitMEX, tutti i valori dell’indice ETHUSD scambiato sulla piattaforma con un timeframe di 1 ora.

Tramite Pandas riusciamo poi a leggere il CSV creato e quindi a preparare il dataset.

Preparazione:

- Per prima cosa, eliminiamo i campi da noi ritenuti superflui e i dati di tipo qualitativo e quindi quelle informazioni che tentano di descrivere un argomento più che misurarlo: si tratta di impressioni, opinioni e punti di vista.
- Controlliamo i vari tipi di dati disponibili nel dataset ed eventualmente li convertiamo (esempio: Data).
- Eliminazione dei valori nulli
- Studiamo la correlazione delle variabili tramite la libreria Seaborn per capire quale variabili mantenere. Quindi tramite la heatmap di correlazione vedremo la correlazione delle variabili indipendenti con la variabile di uscita Close. Selezioneremo poi nella prossima fase solo le caratteristiche che hanno una correlazione superiore a 0,5 (prendendo il valore assoluto) con la variabile di uscita. Questo coefficiente di correlazione ha valori compresi tra -1 e 1:
  - Un valore più vicino a **0** implica una correlazione più **debole** (lo 0 esatto implica nessuna correlazione)
  - Un valore più vicino a **1** implica una correlazione **positiva** più forte
  - Un valore più vicino a **-1** implica una correlazione **negativa** più forte

4. Preprocessing e data cleansing/cleaning

Il preprocessing dei dati comporta la trasformazione del set di dati grezzi in un formato comprensibile. Questa è una fase fondamentale nel data mining per migliorare l'efficienza dei dati. I metodi di pre-elaborazione dei dati influenzano direttamente i risultati di qualsiasi algoritmo analitico e in questo caso influenza direttamente la nostra rete neurale.

Il file da noi scaricato sarà composto da vari campi:
- Open
- High
- Low
- Close
- Trades
- Volume
- Vwap
- LastSize
- Turnover
- HomeNotional
- ForeignNotional

Dopo aver visto la correlazione andiamo a scegliere di mantenere solamente questi campi:
- Close (target)
- High
- Low
- Open
- Vwap: è un parametro di riferimento usato dai trader che fornisce il prezzo medio a cui un titolo è stato scambiato nel corso della giornata, basato sia sul volume che sul prezzo. È importante perché fornisce ai trader un'idea sia della tendenza che del valore di un titolo.

Per prepare i dati andremo a dividere il dataset creato in due: l'80% di esso sarà utilizzato per il training, il restante 20% verrà impiegato per il test e per vedere l'accuratezza del modello da noi creato.

Modellazione del Training-set e del Test-set:
- In questa fase andiamo a normalizzare i dati del training-set prendendo solo gli ultimi 24 elementi
- Li aggiungiamo al test-set
- Normalizziamo i dati del test-set e il target (output)
- Poiché andremo a utilizzare una rete LSTM dobbiamo creare una matrice tridimensionale con i valori trovati

Per la riduzione in scala dei dati multivariati e quindi per la parte di preprocessing andiamo ad utilizzare un algoritmo chiamato StandardScaler.

StandardScaler standardizza i dati, cioè sottrae la media da ogni valore del vettore e divide questa differenza per la deviazione standard. Questo passaggio del preprocessing viene fatto perché le performance del processo di learning sono di solito superiori quando il modello lavora su dati con applicata della standardizzazione.

5. Processi di addestramento classificatori/regressori e tecniche/algoritmi utilizzati

In questo progetto abbiamo implementato una rete neurale che analizza i dati storici per fare una previsione trovando dei pattern simili. Queste reti neurali sono un insieme di algoritmi che assomigliano molto al cervello umano e sono progettati per riconoscere i modelli.

Le reti neurali artificiali sono composte da un gran numero di elementi di elaborazione altamente interconnessi che lavorano insieme per risolvere un problema. Ogni livello successivo riceve l'output dal livello che lo precede, nello stesso modo in cui i neuroni più lontani dal nervo ottico ricevono segnali da quelli più vicini ad esso.

Per risolvere il problema del **gradient vanishing**, scegliamo di utilizzare LSTM.

5.1 LSTM e utilizzo di modelli/classi di Keras

LSTM
La Long Short-Term Memory (LSTM) è una versione avanzata dell'architettura delle reti neurali ricorrenti (RNN) che è stata progettata per modellare le sequenze cronologiche e le loro dipendenze a lungo raggio in modo più preciso delle RNN convenzionali, quindi adatto perfettamente come nel nostro caso a serie temporali.
Un modello ricorrente può imparare a utilizzare una lunga storia di input, se è rilevante per le previsioni che il modello sta facendo. Il modello accumulerà lo stato interno di 24 ore, prima di fare una singola previsione dell'ora successiva.

Funzione di attivazione
Per configurare la rete usiamo la funzione di attivazione standard lineare. Questa funzione di attivazione va a definire come la somma ponderata dell'input viene trasformata in un output da uno o più nodi in uno strato della rete, infatti la scelta della funzione di attivazione ha un grande impatto sulla capacità e le prestazioni della rete neurale.

Keras
È una libreria open source (con licenza MIT) scritta in Python. Lo scopo della libreria è permettere la configurazione rapida di reti neurali. Keras non funge da framework ma da interfaccia di facile

utilizzo (API) per l'accesso e la programmazione a diversi framework di apprendimento automatico.

Sequential
Dal pacchetto di Keras utilizziamo il modello Sequential per aggiungere n layer alla rete neurale.

Dense
La classe Dense (pacchetto Keras) invece rappresenta un normale layer composto da n neuroni, in pratica il classico schema della rete neurale artificiale in cui gli input vengono pesati e assieme al bias vengono trasferiti attraverso la funzione di attivazione all'output.

6. Comparazione ed esportazione dei risultati ottenuti

In questa fase andiamo a visualizzare le prestazioni del modello:
- Effettuiamo la prediction con i valori del test-set
- Invertiamo la prediction, perché i valori restituiti sono ancora normalizzati, per poi avere le misure originali in dollari
- Creiamo il dataset aggiungendo il target scelto (Close) e andiamo a configurare gl'indici come date per maggiore chiarezza

A questo punto abbiamo una visualizzazione complessiva del modello, in modo da capire se il modello è stato allenato nel modo migliore. Per poter decidere questo utilizziamo la tecnica dell' **RMSE.**

Essa è una misura di errore assoluta in cui le deviazioni vengono elevate al quadrato per evitare che valori positivi e negativi possano annullarsi l'uno con l'altro. RMSE è sempre non negativo ed e' un valore pari a 0 (quasi mai raggiunto nella pratica) indicherebbe una perfetta applicazione ai dati. In generale, un RMSE inferiore è migliore di uno più alto.

L'effetto di ogni errore su RMSE è proporzionale alla dimensione dell'errore al quadrato; quindi errori più grandi hanno un effetto sproporzionatamente grande sull'RMSE. Di conseguenza, RMSE è sensibile agli outlier. Non si può predire un arco di tempo molto grande perché l'errore tendere ad essere troppo elevato e quindi il modello risulterebbe inutile.

Per andare a prevedere le ore future, non avendo dati su cui andare a fare la prediction come nella fase di test, dobbiamo crearli. Per farlo andiamo a costruire in modo ricorsivo delle finestre sempre di 24 ore. Queste ci daranno in uscita la prediction dell'ora successiva. Quindi poi andremo ad aggiungerla in coda e faremo uscire dalla testa quella più vecchia, fino a quando non avremo predetto le n variabili che avremo inserito nel settaggio.

Infine esportiamo i risultati su grafici per poter visualizzare al meglio le prediction effettuate dal nostro modello.

Per avere un confronto valido effettuiamo la comparazione tra i valori predetti e valori realmente esistititi 24 dopo.

7. Conclusioni ed eventuali sviluppi futuri del progetto

Per andare a migliorare il modello si possono andare ad inserire un'ulteriori variabili che possono rendere sempre più affidabile il modello da noi creato. In questo nuovo millennio i social sono parte integrante della nostra vita. I giovani si riversano su varie di queste piattaforme, per sentirsi parte di gruppi, per trovare fortuna e per mille altri motivi.

Se si andasse a raccogliere i dati in tempo reale di Twitter e di Reddit, community in cui ci sono un sacco di investitori sia professionisti, sia amatoriali, si potrebbe avere un flusso da inserire nel

modello e andare a prevedere gli asset che sono più in voga al momento, per sapere il sentimento delle persone riguardo ad esso.

Un esempio, abbastanza eclatante è avvenuto poco tempo fa nella community di Reddit dove gli utenti si sono “messi d’accordo” sul fatto di comprare in massa azioni di GameStop.

8. Considerazioni personali

Secondo il nostro pensiero, questo modello sviluppato è sì buono, ma non è sempre affidabile perché ha in ingresso troppi pochi dati e quindi non è da tenere ancora in considerazione per un uso professionale.