**RELAZIONE PROGETTO MACHINE LEARNING & ARTIFICIAL INTELLIGENCE**

**R**ecurrent **N**eural **N**etwork

**Previsione dei Prezzi Futuri della Criptomoneta Ethereum**

**Introduzione**

Il codice implementa un sistema di previsione dei prezzi di chiusura di un dataset finanziario utilizzando reti neurali ricorrenti (LSTM e GRU). Oltre a fornire un dettaglio sulla struttura del codice, questa relazione include un resoconto dei test effettuati, criticità riscontrate, considerazioni su sviluppi futuri e possibili utilizzi della rete. Si discuteranno anche le scelte fatte.

**Struttura del Codice**

**1. File main\_Torch.py**

Il flusso di esecuzione è orchestrato dalla configurazione iniziale, passando attraverso addestramento, valutazione e visualizzazione.

**1.1 Configurazione**

**Caricamento Impostazioni:** Impostazioni come predizione futura, lunghezza dei dati storici, tasso di apprendimento, ecc., vengono caricate da un file di configurazione JSON.

**Preparazione Dati:** Il dataset viene letto, suddiviso in set di addestramento, validazione e test, e normalizzato.

**TensorBoard:** Utilizzo di TensorBoard per tracciare i parametri e le metriche del modello.

**1.2 Addestramento e Valutazione**

**Inizializzazione Modelli:** Sia LSTM che GRU vengono inizializzati, insieme agli ottimizzatori e la funzione di loss.

**Addestramento e Early Stopping:** I modelli vengono addestrati, e si implementa l'early stopping per prevenire l'overfitting.

**Salvataggio Modelli:** I modelli migliori vengono salvati su disco.

**Test e Confronto**: I modelli migliori vengono testati sui dati di test, e viene selezionato il modello con il RMSE più basso.

**1.3 Visualizzazione**

**Visualizzazione Predizioni e Residui:** Le previsioni vengono visualizzate graficamente insieme ai dati reali. Gli errori residui sono analizzati.

**2. File evaluation\_utils.py**

Contiene funzioni per valutare i modelli, tra cui il calcolo del RMSE e dei residui.

**3. File gru\_model.py e lstm\_model.py**

Contengono le definizioni delle reti neurali GRU e LSTM rispettivamente, con l'inizializzazione dei pesi e la definizione del forward pass.

**4. File train\_models.py**

Contiene la funzione train\_models che gestisce l'addestramento dei modelli LSTM e GRU, implementando anche l'early stopping.

**5. File visualization.py**

Contiene funzioni per visualizzare le previsioni e gli errori residui.

**Elementi Teorici**

**1. Reti Neurali Ricorrenti (RNN)**

Le RNN sono progettate per gestire dati sequenziali, mantenendo una memoria delle informazioni precedenti durante l'elaborazione di nuovi dati. Tuttavia, RNN tradizionali possono soffrire di problemi di vanishing o exploding gradient durante l'addestramento su sequenze lunghe. LSTM e GRU sono varianti progettate per superare questo problema.

**2. LSTM (Long Short-Term Memory)**

L'LSTM introduce una struttura complessa di celle e porte che consente di mantenere e dimenticare informazioni a lungo termine. L'uso di un cell state e di porte di input, output e forget rende gli LSTM particolarmente adatti a gestire sequenze lunghe.

**3. GRU (Gated Recurrent Unit)**

Il GRU è un'alternativa più leggera all'LSTM, utilizzando meno parametri ma mantenendo una capacità di gestione delle informazioni a lungo termine. La struttura di GRU include un "update gate" e un "reset gate", controllando il flusso delle informazioni attraverso la rete.

**4. Early Stopping**

L'early stopping è una tecnica di regolarizzazione che interrompe l'addestramento quando le performance sui dati di validazione iniziano a degradare, prevenendo così l'overfitting.

**Risultati dei Test e Criticità**

I test condotti hanno evidenziato le capacità predittive dei modelli LSTM e GRU. L'utilizzo di TensorBoard ha fornito una panoramica dettagliata delle prestazioni durante il processo di addestramento e validazione. Altri aspetti rilevanti includono:

**1. Analisi delle Metriche di TensorBoard**

Le curve di apprendimento e l'andamento delle metriche di addestramento e validazione sono state monitorate tramite TensorBoard. Ciò ha permesso di identificare il punto in cui l'overfitting inizia a manifestarsi e ha guidato l'implementazione dell'early stopping.

**2. Variazione dell'Early Stopping**

L'analisi dell'impatto della variazione del parametro di early stopping (patience) ha evidenziato la sua importanza nella gestione dell'overfitting. Un valore troppo basso potrebbe portare a un arresto prematuro, mentre un valore troppo alto potrebbe compromettere la generalizzazione del modello.

**3. Sensibilità agli Iperparametri**

Sono state effettuate prove di sensibilità rispetto agli iperparametri come il tasso di apprendimento (lr) e il numero di epoche (epochs). Le variazioni hanno mostrato come la scelta accurata di questi parametri influisca significativamente sulle prestazioni del modello.

**4. Inversione della Normalizzazione**

Sono state riscontrate delle problematiche in fase di visualizzazione del grafico per le previsioni dei due modelli, dovute al dover invertire la trasformazione applicata ai valori di chiusura del dataset per poter ottenere un risultato coerente all’input dato.

**Considerazioni su Sviluppi Futuri e Possibili Utilizzi**

Il processo di progettazione del modello e l'analisi dei risultati suggeriscono diverse direzioni per futuri sviluppi e possibili utilizzi:

**1. Ottimizzazione Iperparametri**

L'esplorazione più approfondita degli iperparametri potrebbe portare a ulteriori miglioramenti.

**2. Inclusione di Altri Dati**

L'integrazione di dati aggiuntivi, come notizie finanziarie o indicatori macroeconomici, potrebbe migliorare la capacità predittiva del modello.

**3. Esplorazione di Altri Modelli**

L'analisi comparativa con altri modelli di machine learning, come reti neurali convoluzionali (CNN) o modelli basati su trasformer, potrebbe fornire un quadro più completo delle opzioni disponibili.

**4. Implementazione di Tecniche Avanzate di Normalizzazione**

L'esplorazione di tecniche avanzate di normalizzazione, come la normalizzazione batch o layer, potrebbe migliorare la stabilità dell'addestramento.

**5. Utilizzo in Contesti Real-Time**

L'adattamento del modello per l'utilizzo in contesti real-time, come sistemi di trading automatico, potrebbe essere una direzione interessante per il futuro.

**Spiegazione delle Scelte Fatte**

Le scelte fatte durante il processo di progettazione sono state guidate dalla necessità di ottenere un modello accurato e generalizzabile. Alcune decisioni chiave includono:

**1. Scelta tra LSTM e GRU**

La selezione del modello migliore tra LSTM e GRU è stata basata su un’analisi delle prestazioni durante la fase di validazione. La scelta di utilizzare entrambi i modelli consente un confronto robusto.

**2. Utilizzo di TensorBoard**

L'integrazione di TensorBoard è stata fondamentale per monitorare l'addestramento e valutare le prestazioni del modello. La visualizzazione delle metriche ha facilitato la comprensione del comportamento del modello durante le diverse fasi.

**3. Validazione ad ogni Epoca**

Abbiamo optato per lo spostamento della validazione ad ogni epoca per poter meglio monitorare le prestazioni del modello, anche tramite apposito grafico Tensorboard sulla validation loss. Questo ci ha permesso di modificare gli iperparametri nella maniera più opportuna per ottenere un risultato migliore.

**4. Utilizzo di Early Stopping**

L’utilizzo di un metodo di Early Stopping ha permesso un miglioramento delle tempistiche relative all’addestramento, in quanto permette di interrompere l’esecuzione di tale fase nel momento in cui il modello presenta avanzamenti nell’apprendimento minimi.

**5. Utilizzo di Modello Pre Addestrato**

L’utilizzo di un modello pre addestrato ha permesso di tagliare totalmente la fase di training nel caso volessimo sfruttare un modello salvato già esistente.

**Comportamento delle Reti Neurali**

L'osservazione del comportamento delle reti neurali ha dimostrato la loro capacità di apprendere pattern complessi dai dati finanziari. L'early stopping è risultato essenziale per prevenire l'overfitting e garantire una buona generalizzazione.

**Conclusioni**

Il sistema di previsione implementato ha mostrato promettenti risultati nella previsione dei prezzi di Ethereum. L'analisi dettagliata dei test, delle criticità riscontrate e delle considerazioni sul futuro fornisce una visione completa delle prestazioni del modello e delle opportunità di sviluppo. La flessibilità del codice consente future iterazioni e sperimentazioni per migliorare ulteriormente le prestazioni e l'applicabilità del modello nel contesto finanziario.