Documentazione

Definizione del problema e obiettivi (Problem statement and objectives)

L'obiettivo del progetto è la realizzazione di un classificatore binario in grado di prevedere l'andamento del prezzo di chiusura di una certa azienda, avendo in ingresso una serie di feature e fornendo in uscita segnali di BUY (1, prezzo in salita) o SELL (0, prezzo in discesa), secondo un certo criterio.

Scelta del Target

La natura del problema e l'ambiente finanziario in cui esso si svolge costituiscono una grande sfida che porta il nostro modello ad affrontare costantemente: **overfitting** e **rumore**.

Dato il target infatti, in qualunque modo esso si ponga, è difficile trovare feature così determinanti da indicare subito la strada per l'obiettivo al modello.

Per attenuare quanto possibile questa problematica si è giunti a una versione semplificata del metodo della Triple Barrier ovvero la **Double Barrier** :

- dato un giorno vengono esaminati i giorni successivi (di default 5). Si pongono due barriere, una superiore (take profit +2% risp. al Close) e una inferiore (stop loss -1% risp. al Close): se il prezzo tocca prima quella superiore allora l'etichetta del Target sarà 1, se il prezzo tocca per prima quella inferiore o rimane tra le due. l'etichetta sarà 0.

Obiettivi secondari

- Realizzare altri modelli alternativi di classi diverse a quella scelta per vedere come le teconologie differenti si comportano sul problema.
- Realizzare un banco di prova per i modelli sviluppati, in modo da vedere il loro comportamento in un contesto di trading simulato e confrontarli con strategie di trading basate su regole semplici.

Approccio all'analisi dei dati e risultati (Data analysis approach and findings)

La creazione e la scelta delle feature è stata guidata da uno studio del dominio cercando di capire cosa ognuna di esse potesse rappresentare nel mondo della finanza.

Molte però hanno natura o costruzione simile e rischiano di essere ridondanti. Al fine di aiutare il modello nel perseguire il target si è adotatto un metodo di selezione delle feature basato su tre valutazioni principali:

- **Correlazione lineare**: Attraverso la costruzione di una mappa di correlazione è possibile selezionare ed eventualmente rimuovere le feature che hanno una alta correlazione con altre feature.
- **Mutual Information Score (MI Score)**: questo tipo di valutazione invece mette a confronto la feature presa singolarmente e la correlazione (anche non lineare) che essa ha con il target. In parole povere, quanto quella feature aiuta a identificare il target da sola.
- **Importanza delle feature secondo il modello**: il modello scelto ha una funzionalità di *"feature importance"* che gli permette di dare un valore a una feature sulla base di quanto essa l'ha aiutato nel prevedere il target. Questa metrica va usata con cura poichè prende in considerazione tutte le feature, e quindi può cambiare di molto da un'iterazione all'altra.

Metodo di analisi e scoperte

Date queste metriche di valutazione le feature sono state scelte attraverso varie prove, tenendo in considerazione la valutazione che ognuna di esse ha ottenuto e i risultati del modello.

In linea generale ogni criterio di valutazione esposto non basta (da solo) a inquadrare l'utilità di una feature, ma va osservato nell'insieme, soprattutto rispetto a ciò che il modello indica come importante. Preso nota di questo ogni metrica avrebbe una indicazione propria su come gestire una feature:

- **Mappa di correlazione**: analizzando le feature a coppie, nelle coppie con correlazione ≥ 0.9 (in valore assoluto) si decide di scartarne una.
- **MI score**: le feature con un punteggio scarso o prossimo a 0 possono essere eliminate
- **Feature Importance del modello**: si analizza come l'importanza di ogni feature cambia alla rimozione di altre, rimosse in seguito alla valutazione e applicazione dei criteri esplicitati sopra

Ad esempio un metodo che è stato utilizzato per combinare i tre criteri sopra è il seguente: Prese le feature con meno importanza per il modello, a parità di MI SCORE uguale a 0 o prossimo a 0, è stata rimossa la feature che compariva più volte tra le coppie a correlazione lineare ≥ 0.9. Se alla successiva iterazione il modello migliorava i risultati allora si proseguiva altrimenti l'eliminazione veniva annullata.

Dopo varie iterazioni e uno studio più approfondito del dominio, siamo riusciti a identificare feature significative per il nostro target come le **Bande di Bollinger** (che rappresentano il trend) e feature meno significative come la **Open** che invece sono state rimosse.

Approccio al Machine learning e risultati (Machine learning methodology and results)

Scelta del modello (Model choice)

Il modello scelto per questo problema è XGBoost (eXtreme Gradient Boosting) ed è un algoritmo di machine learning supervisionato usato soprattutto per classificazione e regressione. Si basa sul *gradient boosting*, e lo sfrutta attraverso la costruzione di un insieme di alberi decisionali ognuno dei quali cerca di correggere gli errori commessi dai precedenti attraverso la minimizzazione di una funzione di perdita tramite il **gradiente**.

La scelta di questo modello è avvenuta dopo una serie di tentativi più o meno deludenti con altri modelli. Esso possiede un vasto parco di iperparametri da poter gestire utili a combattere l'overfitting (regolarizzazioni L1 e L2) e può eseguire una valutazione autonoma delle feature più importanti (feature importance).

Soglia delle previsioni

Il modello di XGBoost fornisce come risultato una probabilità che un dato campione appartenga alla classe positiva. Dovendo convertire tale probabilità in un segnale 1 o 0 è necessario introdurre una soglia che sancisca questa divisione. Dopo alcuni tentativi con soglia statica è stata introdotta una soglia dinamica che si basa sui valori più recenti restituiti dal modello. Tale soglia viene scelta grazie all'utilizzo del coefficiente di Matthew che fornisce una misura della qualità di una predizione.

La soglia è stata definita attraverso una metrica nota come *rolling-z-score*. La threshold viene calcolata come la media dei valori (μ) predetti negli ultimi n giorni più k volte la deviazione standard (σ):

$$threshold = \mu_{\text{su ultimi } n \text{ giorni}} + k \cdot \sigma_{\text{su ultimi } n \text{ giorni}}$$

k viene scelto sul validation set in modo da massimizzare il coefficente di Matthew.

Metriche di valutazione del modello

Per valutare i risultati delle predizioni del modello sono state utilizzate diverse metriche (in particolare la *overall accuracy* e la *precision* sulle singole classi) che accompagnate dalla *confusion matrix* hanno costituito la base per la valutazione del modello, almeno dal punto di vista statistico. Per una valutazione più vicina all'applicazione effettiva del modello è stata sviluppata una testbench che si occupa di implementare una semplice logica di investimento basata sulle predizioni del modello. Questa permette di visualizzare un possibile guadagno fornito dall'applicazione del modello.

Gestione della pipeline (Pipeline management)

Presi i dati a disposizione e noto il target, la pipeline per la creazione (e la raffinazione) del modello è stata realizzata attraverso i seguenti passi:

1. Creazione e aggiunta della colonna *Target* attraverso l'applicazione della **Double Barrier**
2. Divisione del dataset in 3 parti: Train (70%), Validation (15%) e Test (15%). I tre set vengono poi trasformati in matrici, rappresentazione più efficace per XGBoost.
3. Allenamento del modello sui dati di Train con valutazione della `log_loss` sui dati di Validation
4. Ottimizzazione automatizzata degli iperarametri tramite `optuna` il quale tenta di massimizzare la precisione (o altra funzione obiettivo specificata) del modello sul Validation set, attraverso la variazione degli iperparametri in un dominio specificato a priori.
5. Scelta della soglia per le previsioni (k).
6. Predizione sui dati di test mai visti dal modello
7. Valutazione delle performance del modello attraverso metriche e testbench

Al termine di una iterazione se ne possono eseguire altre modificando le feature seguendo i criteri di feature selection esposti sopra, tentando di ottenere un modello migliore.

Alcuni risultati

Come esempio vengono riportati qua sotto i risultati del modello che tenta le previsioni studiando le azioni di *Apple* dal gennaio 2018 al novembre 2024:

```
Classification Report (Test Set):
                precision    recall  f1-score   support
            1       0.52      0.23      0.32       120
            0       0.61      0.85      0.71       170

     accuracy                           0.59       290
    macro avg       0.56      0.54      0.52       290
 weighted avg       0.57      0.59      0.55       290
```

con la seguente matrice di confusione:

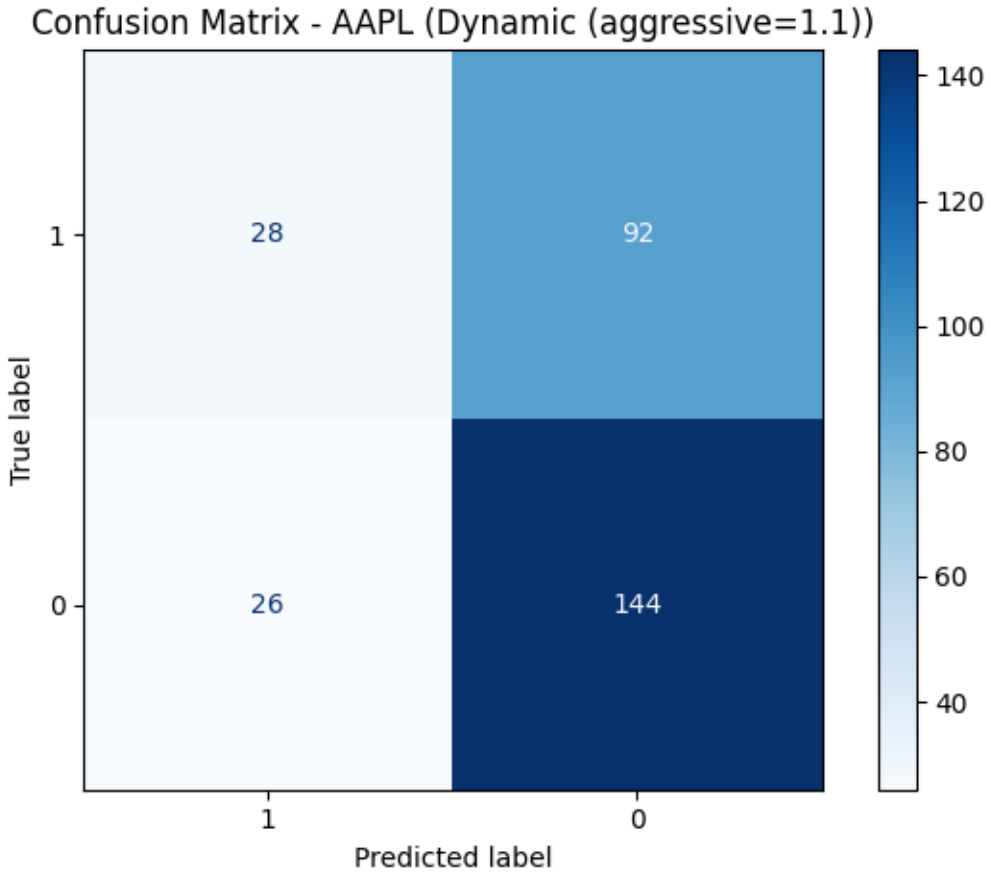

Figure 1: Matrice di confusione

- Poi abbiamo visto anche confrontato il modello con altre strategie, in particolare abbiamo idealizzato una strategia che sfrutta le azioni del modello, analizzata più in dettaglio sotto

Quello che abbiamo ottenuto sul periodo che va da 2024-09-26 a 2025-11-28, partendo da un buget di 1000$ con commissioni al 0.2% è:

Strategia	Capitale finale	Rendimento
Buy and Hold	1220.71	22.071%
Trend Chaser	1059.72	5.973%
Modello	1277.693	27.769%

Figure 2: Periodo testing

Dettagli tecnici implementativi (Technical Implementation Details)

Il progetto è stato sviluppato in **Python 3.12.3**, utilizzando **uv** come gestore dell'ambiente. Le principali specifiche tecniche sono le seguenti:

- L'intero sviluppo è stato effettuato in ambiente **Linux (Ubuntu)**, utilizzando **WSL** su sistema host **Windows**.
- I dati vengono acquisiti tramite la libreria `yfinance`; successivamente vengono calcolati gli indicatori tecnici mediante la libreria `ta` e salvati in file **CSV**, al fine di evitare calcoli ripetuti.
- Per l'elaborazione dei dati sono state utilizzate le librerie `pandas` e `numpy`, mentre la visualizzazione è stata realizzata tramite `matplotlib`.
- Il modello principale utilizzato è `XGBoost`, con ottimizzazione degli iperparametri effettuata tramite `Optuna`.
- La funzione che calcola il target della **Double Barrier** è stata implementata nel file `./target.py`
- La testbench che permette la valutazione del modello è stata implementata in `./TradingTestbench.ipynb` e la logica di trading è contentuta nel seguente segmento:

```
def model_strategy(row):
    price = float(row["Close"])
```

```
if row["model_action"] == 1:
    model_state["streak"] = 0
    if model_state["shares"] == 0:
        # Buy completo
        model_state["shares"] += (model_state["cash"] *
                                  (1 - commission_interest)) / price
        model_state["cash"] = 0.0
        model_state["streak"] = 0

if row["model_action"] == 0:
    model_state["streak"] += 1
    if model_state["shares"] != 0 and (model_state["streak"] > 5):
        # Vendita totale
        model_state["cash"] += model_state["shares"] * price *
                               (1 - commission_interest)
        model_state["shares"] = 0.0
        model_state["streak"] = 0

return model_state["cash"] + (model_state["shares"] * price)
```

In sintesi, quando il modello genera un segnale **BUY** e non sono presenti posizioni aperte, viene effettuato l'acquisto del massimo numero possibile di azioni. Le azioni vengono mantenute in portafoglio per **5 giorni**; qualora, durante questo periodo, il modello generi un ulteriore segnale **BUY**, il conteggio dei 5 giorni viene riavviato. Al termine di tale intervallo, tutte le azioni vengono vendute. Ogni operazione di mercato è soggetta a un costo di commissione, espresso dal parametro `commission_interest`.

- La generazione del modello avviene tramite il file `./ImprovedPipeline.ipynb`, che salva il modello addestrato insieme a diverse informazioni associate:
  - `model`: il modello addestrato
  - `features`: la lista delle feature utilizzate
  - `param`: gli iperparametri impiegati
  - `aggressive`: il valore di k utilizzato per la soglia dinamica
  - Alcuni modelli meno recenti includevano anche i campi `scaler` e `threshold`, ora obsoleti; tuttavia, in `./TradingTestbench.ipynb` è presente codice dedicato alla retrocompatibilità.
- Sono state inoltre utilizzate la libreria `statsmodels` per il modello **ARIMA** e `tensorflow` per il modello **LSTM**.
- Per lo sviluppo dell'applicativo è stato utilizzato **Flask**, insieme a **CSS personalizzato** e **Bootstrap** per la parte grafica.

- Ulteriori dettagli relativi alla struttura del progetto e allo scopo dei singoli file sono disponibili nel file `./README.md`.

Difficoltà incontrate e soluzioni (Challenges Encountered and Solutions)

- **Elevata rumorosità del target**: il problema è stato affrontato tramite la definizione del target mediante il **Double Barrier Method**.
- **Limitata conoscenza del dominio finanziario**: questa criticità ha richiesto un'estesa attività di ricerca per individuare le feature più rilevanti, portando anche alla scelta di modelli capaci di estrarre automaticamente le feature e all'uso di tecniche di selezione delle stesse.
- **Selezione degli iperparametri ottimali**: è stato osservato fin dalle prime analisi che il dataset non risultava perfettamente bilanciato; per questo motivo è stato utilizzato il parametro `scale_pos_weight` di XGBoost per compensare lo sbilanciamento delle classi. Inoltre, l'aumento del valore di `min_child_weight` ha mostrato un miglioramento delle prestazioni in termini di precisione. Al contrario, valori troppo elevati o troppo bassi di `max_depth` hanno evidenziato un incremento del rischio di overfitting.

Eventuali miglioramenti futuri (Possible Future Improvements)

In questo progetto è stato esplorato il problema del classificatore binario per la previsione del prezzo di chiusura di un titolo azionario, affrontando e risolvendo problemi come quelli visti sopra.

Tuttavia, sono rimasti molte altre strade da esplorare per migliorare ulteriormente il progetto, come:

- Introduzione di una metrica di valutazione del sudetto *investor sentiment*: perchè se correliamo, nel caso di Apple, il sentimento delle persone postumo hai famosi eventi che vengono tenuti annualmente, si nota come all'annuncio di AppleSilicon ci sia stato un picco e invece subito dopo l'annuncio del iPhone 15 si sia avuto un calo. Quindi l'idea sarebbe quella di introdurre una metrica che valuti il sentiment delle persone/investor sui social e vederne la correlazione con il prezzo.
- Lavorare sul'intervallo: noi abbiamo considerato 7 anni come dopo alcuni test iniziali e se si prendevano meno dati si aveva un calo di performance, e prendere più dati causava un aumento dell'overfitting. Però si potrebbe implementare un meccanismo di traning a finestra mobile, in modo da tenere il modello più aggiornato senza dover riaddestrarlo da zero.
- Esplorare modelli alternativi: Noi abbiamo ulilizzato `XGBoost` come modello per

motivi che abbiamo esposto sopra, ma come da secondo obbiettivo abbiamo analizzato anche altri modelli (come anche consigliato).

- Cercano online abbiamo trovato questo articolo che confronta vari modelli per la regresione del prezzo di chiusura di Apple, e si vede come **Prophet** è il peggiore tra tutti i modelli considerati nello studio, questo unito a opinioni online che lo sconsigliano per questo tipo di problema ci ha fatto scartare questa opzione questo task, allora lo abbiamo evitato.
- Abbiamo provato a vedere se **ARIMA** invece aveva del potenziale, abbiamo addestrato 5 modelli distiniti per la previsione il prezzo di close a distanza di 1,2,3,4 e 5 giorni, poi considerato il close del giorno corrente, ci abbiamo applicato la regola della double barrier per trasformarlo in un classificatore. Quello che abbiamo notato è che il risutato dipende molto dal peso della parte autoregressiva, differenziale e media mobile (P, D, Q), e passa da essere estremamente aggressivo (molti BUY) a estremamente conservativo (solo SELL), abbiamo trovato un punto di mezzo con la terna (1,1,0):

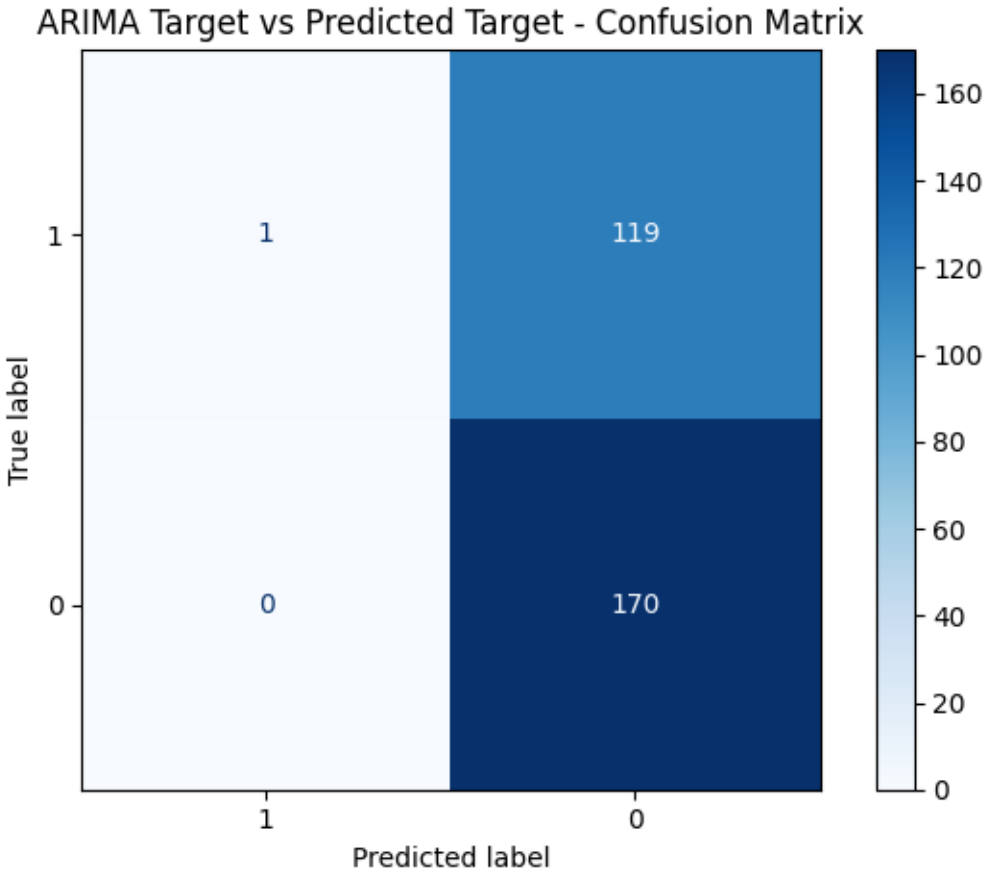

Figure 3: Matrice di confusione ARIMA

Abbiamo anche fatto un testbench simile a quello fatto per XGBoost con un capitale inizale di 1000 e commissioni del 0.2% e il risultato è stato:

Strategia	Capitale finale	Rendimento
Buy and Hold	1220.71	22.071%
ARIMA	1122.22	12.22%

Figure 4: Trading ARIMA

– Inoltra abbiamo esplorato anche LSTM, che possibile esamio di rete neurale, che si è rivelata più performante di ARIMA e XGBoost, applicandoci solo gli stessi miglioramenti pensati per XGBoost (feature selection e soglia dinamica). Nonostante questi si sono rilevalati sufficenti per ottenere dei risultati migliori, quindi un ulteriore studio su LSTM potrebbe portare a risultati ancora migliori.

Anche in questo caso abbiamo fatto un testbench simile a quello fatto per XGBoost con un capitale inizale di 1000 e commissioni del 0.2% e il risultato è stato:

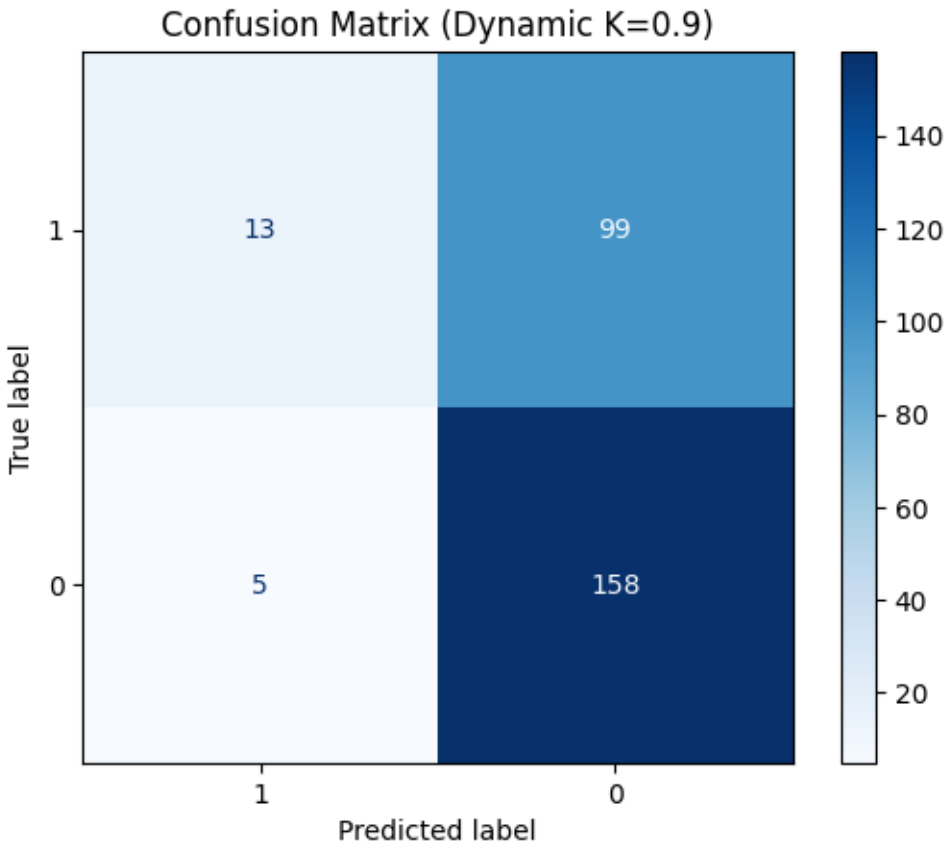

Figure 5: Matrice di confusione LSTM

Strategia	Capitale finale	Rendimento
Buy and Hold	\$1220.71	22.071%
LSTM	\$1390.29	39.029%

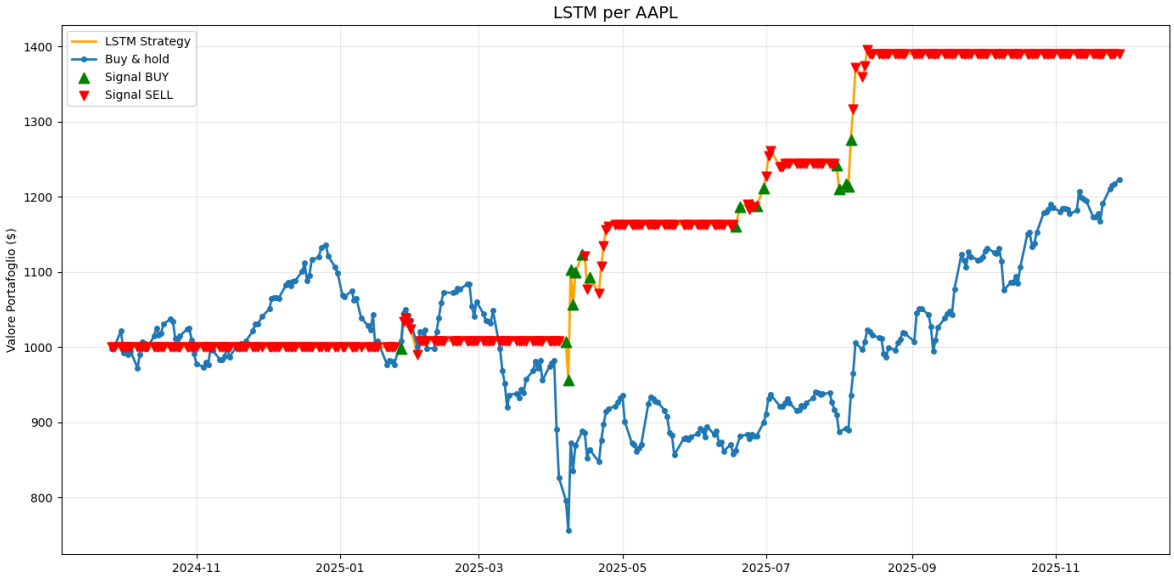

Figure 6: Trading LSTM