Documentazione

In questo file ci concentriamo sugli aspetti più tecnici del progetto:

Definizione del problema e obbiettivi (Problem statement and objectives)

L'obbiettivo principale del progetto è la creazione di un classificatore binario in grado di prevedere l'andamento di un ticker azionario in un arco di tempo definito.

- Però come sappiamo, i mercati finanziari sono molto rumorosi quindi abbiamo definito un **target** in maniera da ridurre il rumore e trasformare un insieme di valori di close price in un'etichetta binaria, lo abbiamo nominato tra noi "*Double Barrier Method*".
  - Preso un giorno si va a vedere i giorni successivi (di default 5) e si controlla se in uno di questi giorni il prezzo supera una soglia superiore (take profit [2%]) o inferiore (stop loss [1%]) rispetto al prezzo del giorno di partenza. Se viene superata la soglia superiore l'etichetta sarà 1, se viene superata la soglia inferiore o rimane stagnante (non tocca nè un nell'altra) l'etichetta sarà 0.
- Quindi il nostro obbiettivo è quello di creare un modello che, dato un insieme di feature calcolate su una finestra temporale precedente al giorno di partenza, riesca a prevedere l'etichetta binaria definita dal target.

Ulteriori obbiettivi erano:

- 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.
- Realizzare altri modelli alternativi di classi diverse a quella scelta per vedere come le teconologie differenti si comportano sul problema.

Approccio al Data Analysis e riultati (Data analysis approach and findings)

Ricercando online abbiamo trovato una lista di possibili indici tecnici per l'analisi dei mercati finanziari, però data la nostra non eccelente conoscenza in ambito finanziario abbiamo deciso di usare un modello che in automatico fosse in grado di estrarre le feature più rilevanti.

- Quindi abbiamo categorizzato le fearure secondo tre metdologie:
  - **Correlazione lineare**: abbiamo visto quanto sono correlate linearmente

  - **Correlazione non lineare**: Stessa roba ma non lineare
  - **Importanza delle feature secondo il modello**: quelle che il modello ritiene più importanti per la predizione.

  Quindi abbiamo selezionato le feature che avevano un punteggio di 0 nella correlazione non lineare e tra queste abbiamo levato quelle che comparivano più volte nelle corelaizioni lineari in $|\cdot| > 0.9$, e poi abbiamo osservato l'importanza delle feature secondo il modello per vedere l'evoluzione in base alla rimozione delle feature.

- Altre feature che abbiamo rimosso sono state rimosse in base a un po' di conoscenze finanziarie di base, le fashe di Bollinge sono più importanti per il nostro taget rispetto a valori come l'open che non ha troppa infuenza sull'andamento ma fa solo da riferimento

Approccio di Machine Learning e risultati (Machine learning methodology and results)

Come già espresso sopra il modello che abbiamo utilizzato è l'XGBoost, un modello di boosting basato su alberi decisionali.

- Abbiamo scelto questo modello, un po' per la sua capacità di estrarre le feature più rilevanti e perchè è un modello che si prestava bene a questo tipo di problema, come anche il Random Forest (Abbandonato perchè dopo alcuni test preliminari non dava risultati soddisfacenti).
- Abbiamo diviso il dataset in 3 parti: training (70%), validation (15%) e test (15%).
- Andiamo ad allenare il modello sul training set e lo controlliamo per overfitting sul validation set utilizzando come funzione target la log-loss
- l'ottimizzazione degli iperparametri è fatta con un framework di ottimizzazione bayesiana (Optuna) che cerca di massimizzare la precisione sul validation set, variando i valori degli iperparmentri in domini specificati da noi, cosa che si è rivelata molto ultile, infatti spingendo per valori alti dell'attributo `min_child_weight` abbiamo notato un aumento delle perestazioni.
- Il modello di XGBoost fornisce come riusltato una probabilità che quel campione appartenga alla classe positiva, ma dopo del testing abbiamo notato che settare una soglia di conevrsione statica causava al modello di essere troppo conservativo o troppo aggressivo, quindi abbiamo introdotto una metrica per la soglia dinamica basata sui i recenti valori restituiti dal modello.
  - metrica nota come *rolling-z-score*, mettiamo come threshold la media dei valori predetti negli ultimi n giorni più k volte la deviazione standard

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

  - La k viene scelta sul validation set in modo da massimizzare il coefficente di Matthews (MCC) sul quel set

- Quindi per il nostro modello finale addestrato sui dati che vanno dal gennaio 2018 al novembre 2024, otteniamo:

```
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:

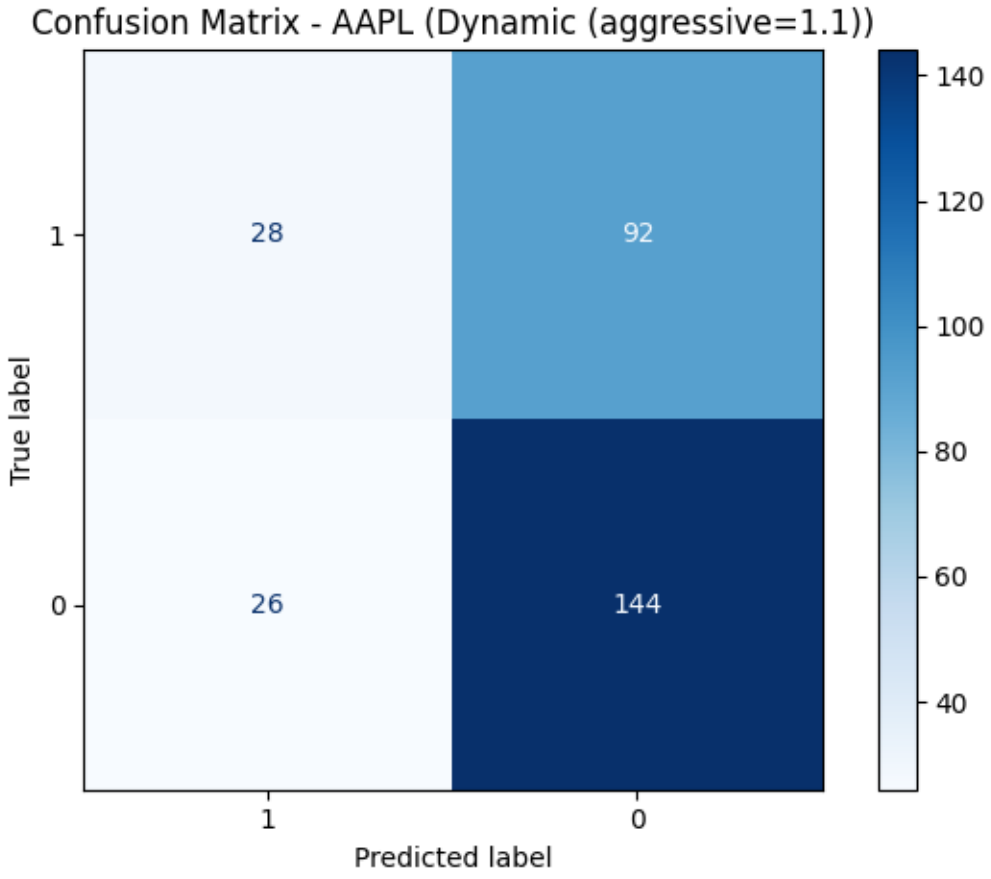

Figure 1: Matrice di confusione

- Poi abbiamo visto anche confrontato il modello con altre strategie, in particolare abbiamo idealizzato una strategia basata sulle azioni del modello
  - L'idea è di sfruttare la definizione del target per fare trading, quindi quando il modello predice un'etichetta positiva (1) e non si hanno azioni si compra il maggior numero di azioni disponibili con il capitale a disposizione, poi se per 5 giorni di fila il modello predice un'etichetta negativa (0) si vendono tutte le azioni possedute al 5 giorno, questo perchè si immaginamo che il modello sia corretti e quindi avendo 5 etichette negative consecuitve non ha più senso tenere le azioni, perchè non ci saranno variazioni positive nei prossimi giorni, però se nel intervallo di 5 giorni il modello predice un'etichetta positiva si continua a tenere le azioni e resettiamo il numero di giorni da aspettare.

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

Strategia	Capitale finale	Rendimento
Buy and Hold	1225.61	22.56%
Trend Chaser	1159.63	15.96%
Modello	1345.96	34.60%

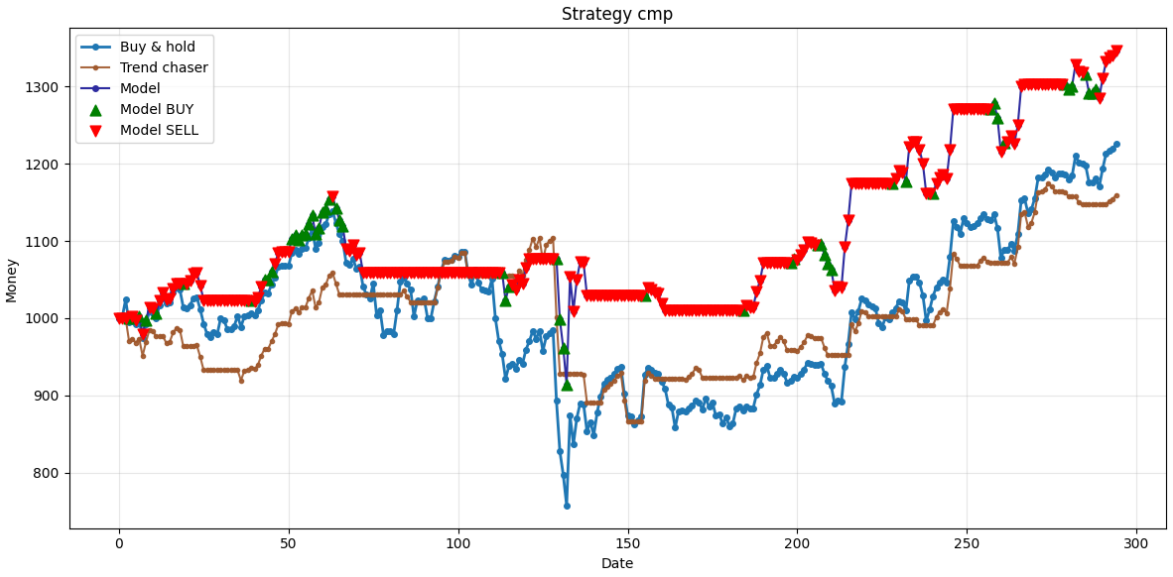

Figure 2: Periodo testing

Dettagli tecnici implementativi (Technical implementation details)

Il progetto è stato siluppato in Python 3.12.3 con uv per gestore dell'ambiente, utilizzando questi specfiche

- Abbiamo realizzato tutto su Linux (ubuntu), utlizzando WSL su sistema host Windows
- Preleviamo i dati con la libreria `yfinance`, e dopo avervi calcolato gli indici, utilizzando la libreria `ta`, sopra li salviamo in file CSV per evitare di doverli ricalcolare ogni volta
- Per lavorare con i dati abbiamo utilizzato `pandas`, `numpy` con `mathplotlib` per la visualizzazione
- Come già specificato il modello è `xgboost` con ottimizzazione degli iperparametri con `optuna`
- Quando si genera un modello con il file `./ImprovedPipeline.ipynb` si va a salvare il modello e altre informazioni relativo ad esso
  - `model`: l'attributo che contiene il modello addestrato
  - `features`: la lista delle feature utilizzate
  - `param`: gli iperparametri utilizzati
  - `aggressive`: il valore di k usato per la soglia dinamica
  - [alcuni modelli più vecchi contenevano i campi `scaler` e `threshold` che ora sono obsoleti, però esiste del codice per la retrocompatibilità nella testbench]
- Inoltre abbiamo ulilizzato la libreria `statsmodels` per ARIMA e `tensorflow` per il modello LSTM.
- Per l'applicativo abbiamo usato `flask` e un insieme di css custom e bootstrap per la parte grafica.
- Ulteriori dettagli riguardanti la struttura del progetto e il singolo scopo di ogni file rimandiamo al file `./README.md`

Difficoltà incontrate e soluzioni (Challenges encountered and solutions)

- La roumorosità del target, che è stata risolta con la definizione del target Double Barrier Method
- La relativa conosceza del settore finanziario, che ha portato a dover fare molte ricerche online per capire quali feature fossero più rilevanti oltre a preferire modelli in grado di estrarre feature in automatico e metodi di selezione delle feature.
- Selezione degli iperparametri ottimale, abbiamo notato da subito che il nostro dataset non era bilanciatissimo, quindi abbiamo utilizzato il parametro `scale_pos_weight` di XGBoost per bilanciare le classi, inoltre abbiamo notato che alzando il valore di `min_child_weight` si ottenevano migliori prestazioni in

termini di precisione, con che un valore troppo alto o basso di `min_depth` come maggiora fattore di overfitting.

Eventual Future improvements

Noi ci siamo limitati a utilizzare modelli che anche nel corso si sono più utilizzati, però esitono molte altre alternative come:

- PROPHET di META che però abbiamo letto online non essere particolamente adatto a obbiettivo (allegare ricerca)
- ARIMA, che siamo riusciti a tradurre in un classificatore binario, allenando 5 modelli per prevedere a 1,2,3,4,5 giorni di distanza e poi applicandoci sopra la Double Barrier, ma non ha dato risultati migliori di XGBoost (allegare i risultati ARIMA)
- Abbiamo anche esploarato LSTM, come prova e con solo le assunsioni che abbiamo fatto per l'xgboost siamo risuciti a ottenere risultati promettenti, quindi con un lavoro più approfondito si potrebbero ottenere risultati migliori (allegare i risultati LSTM)