Forecasting gold price with the XGBoost algorithm and SHAP interaction values

Niccolò Mazzi - 7076194

Corso di laurea Magistrale in Ingegneria Informatica
Università degli Studi di Firenze
Febbraio 2022

Contents

1 Abstract

Nel progetto presentato di seguito si fa riferimento all'articolo [7], del quale si tenta di replicare i risultati.
Nell'articolo vengono presentate alcune tecniche di Machine learning al fine di effettuare un forecasting del prezzo dell'oro, comparandone le performance.
Infine si presenta una analisi effettuata utilizzando il pacchetto SHAP, ovvero un pacchetto che offre una serie di tecniche finalizzate a facilitare l'interpretazione dei risultati ottenuti tramite le suddette tecniche.

2 Introduzione

L'analisi del prezzo dell'oro e delle sue fluttuazioni è da sempre stato un argomento centrale per tutti gli investitori, in quanto si ritiene che esso sia un buon indicatore dell'andamento dell'economia mondiale.
Per effettuare tale analisi esistono tre categorie principali di tecniche di machine learning:

1. **Metodi classici**: Descrivono principalmente la relazione lineare tra variabili attraverso una specifica formulazione analitica ex ante

2. **Metodi di intelligenza artificale**: Sono uno dei tipi più importanti di modelli di apprendimento automatico, che sono stati introdotti ed esaminati per la previsione dei prezzi delle materie prime. Tali modelli sono caratterizzati dalla loro struttura non lineare, dalla flessibilità e dal processo di apprendimento basato sui dati

3. **Approci ibridi**: sono una combinazione di reti neurali artificiali e diversi modelli per prevedere le fluttuazioni di una variabile come i prezzi delle materie prime, i rendimenti di mercato, ecc.

Divesi studi hanno mostrato che vi sono diversi fattori che influenzano il prezzo dell'oro come il valore delle valute ([1]), il prezzo dei metalli ([2][4]), il prezzo del petrolio ([5] [6]) e i tassi di cambio ([3]).
Il metodo SHAP sarà, quindi, utilizzato per assegnare ad ognuno di tali fattori una rilevanza nel determinare il prezzo dell'oro.

3 Descrizione del Dataset

I dati che andranno a comporre il nostro dataset sono una serie di misurazioni mensili effettuate nel periodo che va da Gennaio 1986 e Dicembre 2019, per un complessivo di 408 misurazioni per ogni categoria.
Le misurazioni prese in esame nell'implementazione che andremo ad illustrare

sono quelle inerenti i prezzi dell'argento e del petrolio, il tasso di cambio USD-Euro e USD-CNY[1] e lo Standard & Poor 500 [2].
Il dataset è stato, poi, suddiviso in training set e test set con una percentuale $80\% - 20\%$ rispettivamente utilizzando la porzione di codice sottostante, al fine di poterlo utilizzare per addestrare i modelli che andremo ad usare.

```
#Splitting the data
X = data
Y = Gold['Price']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2,
↪  random_state=2)
```

4 Previsioni dei modelli

Si illustrano, quindi, le implementazioni dei modelli utilizzati in tale analisi, affiancate dalle rispettive performance

4.1 Regressione Lineare

La regressione lineare è un'analisi statistica che analizza l'effetto di variabili indipendenti selezionate su una variabile dipendente esplicativa. La regressione lineare utilizza il metodo dei minimi quadrati ordinari per stimare la relazione lineare tra le variabili. La funzione dei modelli previsionali è espressa come segue:

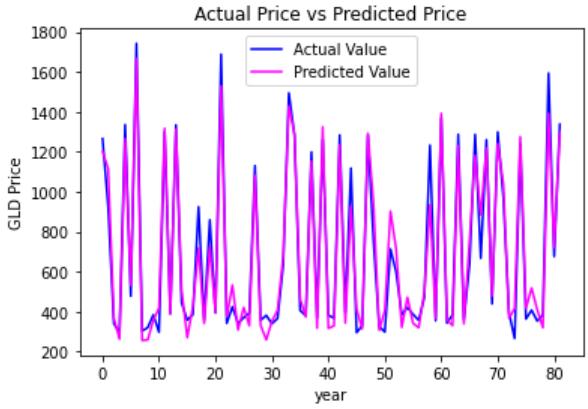
$$Y_t = \beta_0 + \beta_1 X_1 + \ldots + \beta_n X_n + \epsilon_t$$

dove Y_t è il valore previsto al tempo t, X_t è un vettore di k variabili predittive al tempo t, β_j sono i coefficienti stimati e ϵ_t è un termine di errore casulae al tempo t.
Si presenta di seguito il codice per l'implementazione di tale modello:

```
reg = LinearRegression().fit(X_train, Y_train)
pred = reg.predict(X_test)
Y_t = list(Y_test)
LRpred = pd.DataFrame(pred)
Plot(Y_t,pred)
model = "Linear Regression"
Performance(Y_test, LRpred, model)
```

La cui esecuzione produrrà il seguente risultato

[1] **renminbi** = valuta legale cinese
[2] **SP_500** = indice azionario che segue l'andamento di un paniere azionario formato dalle 500 aziende statunitensi a maggiore capitalizzazione.




```
Performance obtained byLinear Regression
****************************************
****************************************
** RMSE: 90.750263
** MSE: 8235.610267
** MAE: 69.495978
** R2 Score: 0.955955
****************************************
****************************************
```

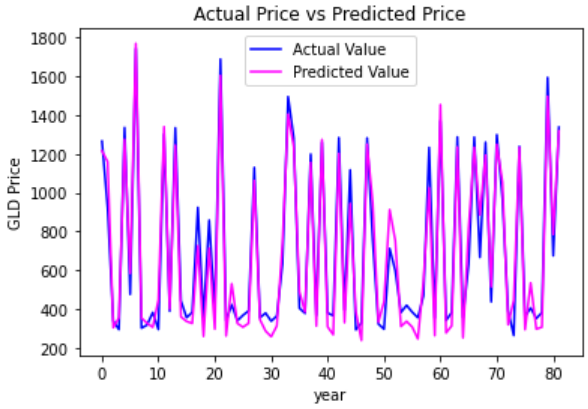
4.2 Rete Neurale

Le reti neurali (NN) sono un insieme di neuroni formali associati a livelli e che operano in parallelo.
In una rete, ogni sottogruppo elabora indipendentemente dagli altri e trasmette il risultato della sua analisi al sottogruppo successivo.
Il primo livello è chiamato livello di input. Riceverà i dati di origine che vogliamo utilizzare per l'analisi. La sua dimensione è quindi determinata direttamente dal numero di variabili di input.
Il secondo strato è nascosto, nel senso che ha solo un'utilità intrinseca per la rete e non ha contatto diretto con l'esterno.
Il terzo livello è chiamato livello di output. Fornisce il risultato ottenuto dopo la compilazione da parte della rete dei dati inseriti nel primo livello. Ogni neurone raccoglie le informazioni fornite dai neuroni dello strato precedente e quindi calcola il suo potenziale di attivazione. Questo potenziale viene quindi trasformato da una funzione per determinare l'impulso inviato ai neuroni del livello successivo.
Il maggiore svantaggio comportato dall'utilizzo delle reti neurali è che i risultati ottenuti non sono interpretabili, in quanto ciò che accade all'interno dei livelli nascosti non è esplicitato.
Nell'implementazione presentata si utilizza una rete neurale di tipo sequenziale con 4 livelli nascosti. I primi 3 livelli avranno una funzione di attivazione di tipo *"relu"*, ovvero la *rectified linear unit activation function* definita come

$f(x) = x^+ = max(0, x)$ dove x è l'input, mentre l'ultimo avrà una funzione di arrivazione lineare, la quale serve soltanto per eseguire un plot più accurato, dal momento che non cambia in alcun modo la somma pesata degli input ma restituisce direttamente i valori. A seguire il codice della rete neurale sopra descritta.

```
NN = keras.Sequential()
NN.add(Dense(408, activation='relu'))
NN.add(Dense(64, activation='relu'))
NN.add(Dense(32, activation='relu'))
NN.add(Dense(1, activation='linear'))
NN.compile(loss='mse',optimizer='adam')
X_tr = np.asarray(X_train).astype(np.float32)
Y_tr = np.asarray(Y_train).astype(np.float32)
NN.fit(X_tr, Y_tr, epochs=150)
NNpred = NN.predict(X_test)
Y_t = np.array(Y_test, dtype=float)
Plot(Y_t, NNpred)
model = "Neural Network"
Performance(Y_test, NNpred, model)
```

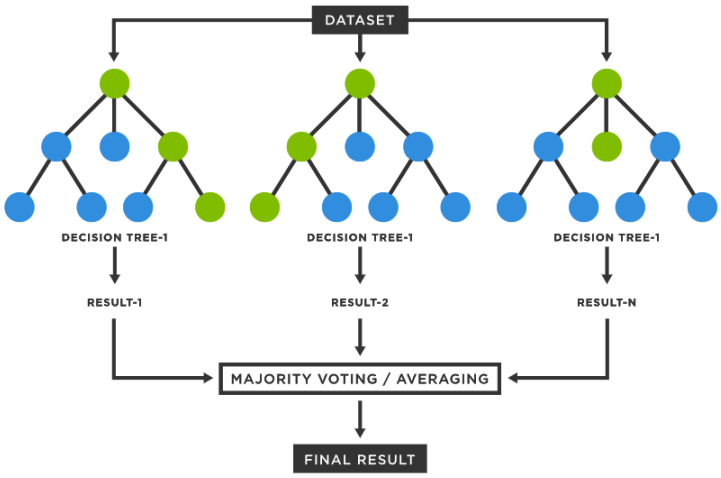
La cui predizione sarà la seguente




```
Performance obtained byNeural Network
****************************************
****************************************
** RMSE: 99.051951
** MSE: 9811.289076
** MAE: 84.056347
** R2 Score: 0.947528
****************************************
****************************************
```

4.3 Random Forest

Random Forest è una tecnica potente ma relativamente semplice. Consente l'identificazione rapida e automatica di informazioni rilevanti da set di dati estremamente grandi. Il più grande punto di forza dell'algoritmo è che si basa sulla raccolta di molte previsioni (alberi) piuttosto che sulla fiducia in una sola.



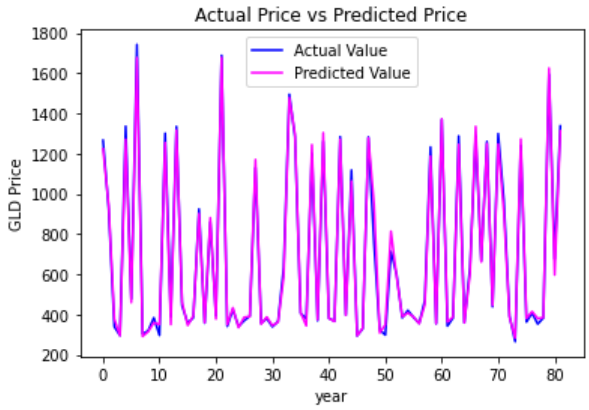
Dopo la fase di train, i valori predetti saranno espressi come

$$Y_t = \frac{1}{T}\sum_{h=1}^{T} l_k(x)$$

dove $l_k(x)$ è il set dei *random trees* e X è il vettore delle T variabili in input. L'implementazione dell'algoritmo Random Forest usata nel progetto è la seguente

```
RF = RandomForestRegressor()
RF = RF.fit(X_train, Y_train);
RFpred = RF.predict(X_test)
Y_t = list(Y_test)
RFpred = pd.DataFrame(RFpred)
Plot(Y_t, RFpred)
model = "Random Forest"
Performance(Y_test, RFpred, model)
```

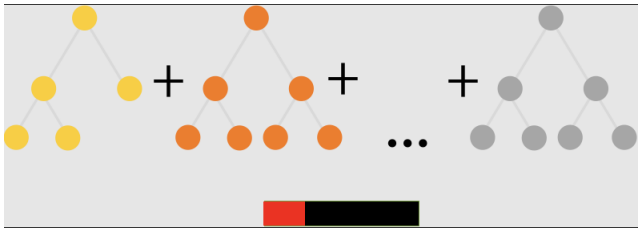
E produrrà i seguenti risultati




```
Performance obtained byRandom Forest
****************************************
****************************************
** RMSE: 36.878527
** MSE: 1360.025727
** MAE: 23.764667
** R2 Score: 0.992726
****************************************
****************************************
```

4.4 LightGBM

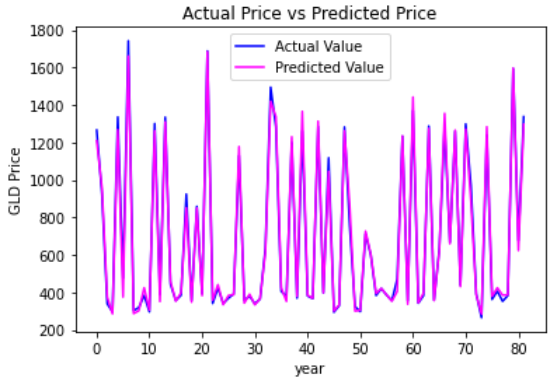
LightGBM è un framework di *gradient boosting* basato su alberi decisionali per aumentare l'efficienza del modello e ridurre l'utilizzo della memoria.



Esso utilizza due nuove tecniche: un campionamento laterale basato su gradiente e un bundle di funzionalità esclusive (EFB) che soddisfa i limiti dell'algoritmo basato su istogramma utilizzato principalmente in tutti i framework GBDT (Gradient Boosting Decision Tree). Le due tecniche di GOSS ed EFB descritte di seguito formano le caratteristiche dell'algoritmo LightGBM.
Diverse istanze di dati hanno ruoli diversi nel calcolo del guadagno di informazioni. Le istanze con gradienti maggiori (cioè istanze poco addestrate) contribuiranno maggiormente al guadagno di informazioni. GOSS mantiene quelle istanze con gradienti elevati e rilascia solo casualmente quelle istanze con gra-

dienti piccoli per mantenere l'accuratezza della stima del guadagno di informazioni.

```
LGBM = lgb.LGBMRegressor()
Y = Y_train.astype('int32')
LGBM = LGBM.fit(X_train,Y)
LGBMpred = LGBM.predict(X_test)
Y_t = list(Y_test)
LGBMpred = pd.DataFrame(LGBMpred)
Plot(Y_t, LGBMpred)
model = "LGBM"
Performance(Y_test, LGBMpred, model)
```




```
Performance obtained byLGBM
****************************************
****************************************
** RMSE: 37.509178
** MSE: 1406.938461
** MAE: 26.098297
** R2 Score: 0.992475
****************************************
****************************************
```

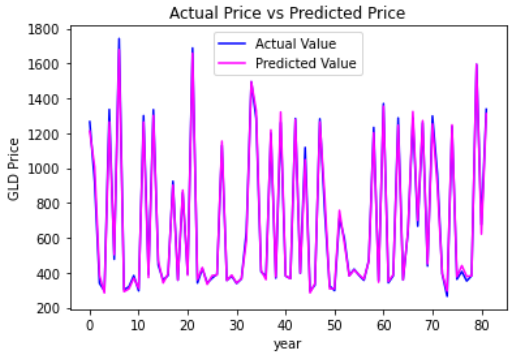
4.5 CatBoost

CatBoost è un algoritmo di apprendimento automatico open source relativamente nuovo, sviluppato nel 2017 da una società chiamata Yandex. Yandex è una controparte russa di Google, che lavora all'interno dei servizi di ricerca e informazione.
Uno dei vantaggi principali di CatBoost è la sua capacità di integrare una varietà di diversi tipi di dati, come immagini, audio o funzionalità di testo in un unico framework.
CatBoost offre anche un modo singolare di gestire i dati categoriali, richiedendo un minimo di trasformazione delle caratteristiche categoriali, al contrario della

maggior parte degli altri algoritmi di apprendimento automatico che non possono gestire valori non numerici.
Dal punto di vista dell'ingegneria delle funzionalità, la trasformazione da uno stato non numerico a valori numerici può essere un compito non banale e noioso e CatBoost rende questo passaggio obsoleto.
CatBoost si basa sulla teoria degli alberi decisionali e del gradient boosting. L'idea principale di tale boosting è di combinare in sequenza molti modelli deboli e quindi, attraverso una ricerca di tipo greedy, creare un modello predittivo forte. Poiché il gradient boosting si adatta agli alberi decisionali in sequenza, gli alberi adattati impareranno dagli errori degli alberi precedenti e quindi ridurranno gli errori. Questo processo di aggiunta di una nuova funzione a quelle esistenti viene continuato fino a quando la funzione di perdita selezionata non viene più ridotta.
Nella procedura di crescita degli alberi decisionali, CatBoost non segue modelli di gradient boosting simili. Invece, CatBoost fa crescere alberi ignari, il che significa che gli alberi vengono cresciuti imponendo la regola che tutti i nodi allo stesso livello testano lo stesso predittore con la stessa condizione, e quindi un indice di una foglia può essere calcolato con operazioni bit per bit.
La procedura ad albero ignaro consente uno schema di adattamento semplice ed efficienza sulle CPU, mentre la struttura ad albero funziona come una regolarizzazione per trovare una soluzione ottimale ed evitare l'overfitting.

```
CatB = CatBoostRegressor()
Y = Y_train.astype('int32')
CatB = CatB.fit(X_train,Y)
CatBpred = CatB.predict(X_test)
Y_t = list(Y_test)
CatBpred = pd.DataFrame(CatBpred)
Plot(Y_t, CatBpred)
model = "CatBoost"
Performance(Y_test, CatBpred, model)
```

Dall'esecuzione del suscritto algoritmo si ottiene:




```
Performance obtained byCatBoost
****************************************
****************************************
** RMSE: 29.610148
** MSE: 876.760854
** MAE: 21.310638
** R2 Score: 0.995311
****************************************
****************************************
```

4.6 XGBoost

XGBoost è un algoritmo di Machine Learning presentato per la prima volta in [3], esso è basato su un albero decisionale che utilizza un framework di potenziamento del gradiente. Nei problemi di previsione che coinvolgono dati non strutturati (immagini, testo, ecc.) le reti neurali artificiali tendono a superare tutti gli altri algoritmi o framework. Tuttavia, quando si tratta di dati strutturati/tabulari di piccole e medie dimensioni, gli algoritmi basati sull'albero decisionale sono considerati i migliori al momento.

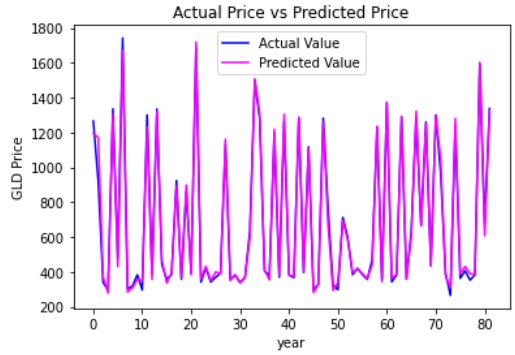
XGBoost e Gradient Boosting Machines (GBM) sono entrambi metodi ad albero che applicano il principio del potenziamento dei learner deboli utilizzando un'architettura di gradient descend. Tuttavia, XGBoost migliora il framework GBM di base attraverso l'ottimizzazione dei sistemi e miglioramenti algoritmici.

Figure 1: Come XGBoost ottimizza GBM

A seguire l'implementazione Python dell' algoritmo XGBoost

```
XGB = xgb.XGBRegressor()
XGB = XGB.fit(X_train,Y_train)
XGBpred = XGB.predict(X_test)
Y_t = list(Y_test)
XGBpred = pd.DataFrame(XGBpred)
Plot(Y_t, XGBpred)
model = "XGBoost"
Performance(Y_test, XGBpred, model)
```

E i risultati ottenuti eseguendolo sul dataset in esame




```
Performance obtained byXGBoost
****************************************
****************************************
** RMSE: 40.364454
** MSE: 1629.289130
** MAE: 24.617851
** R2 Score: 0.991286
****************************************
****************************************
```

Figure 2: Come XGBoost ottimizza GBM

5 Comparazione delle performance dei modelli

Al fine di valutare le performance dei modelli sono stati utilizzati 4 indivatori, ovvero:

- **RMSE** (root mean square error) : è la misura di quanto bene una regressione si adatta ai punti dati. RMSE può anche essere interpretato come deviazione standard nei residui.
- **MSE** (mean square error) - Per calcolare l'MSE, si prende la differenza tra le previsioni del modello e la verità , se ne calcola il quadrato e si ricava la media sull'intero set di dati. Il MSE non sarà mai negativo, visto che

quadramo sempre gli errori. Il MSE è formalmente definito dalla seguente equazione:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

- **MAE** (mean absolute error) : Dato un set di dati di test, il MAE del modello si riferisce alla media dei valori assoluti di ciascun errore di previsione su tutte le istanze del set di dati di test. L'errore di previsione è la differenza tra il valore effettivo e il valore previsto per quell'istanza. Tale errore può essere formalmente descritto come

$$MAE = \frac{\sum_{i=1}^{n}|y_i - \lambda(x_i)|}{n}$$

- $\mathbf{R}^2$ (coefficiente di determinazione) : R^2 è una misura statistica che rappresenta la bontà di adattamento di un modello di regressione. Il valore ideale per r-quadrato è 1. Più il valore di r-quadrato è vicino a 1, migliore è il modello montato.

  R^2 è un confronto della somma residua dei quadrati con la somma totale dei quadrati. La somma totale dei quadrati viene calcolata sommando i quadrati di distanza perpendicolare tra i punti dati e la linea media.

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

Il modello che ha fornito la migliore approssimazione sarà quello che avrà RMSE, MSE e MAE minori, e R^2 maggiore.
Nel caso preso in esame, le migliori performance sotto questo punto di vista sono state fornite dall'algoritmo CatBoost, seguito in ordine da Random Forest, da LGBM, da XGBM e, infine, dalle Reti Neurali e dalla Regressione Lineare.

Tali risultati sono tutti in linea con le aspettative, eccezion fatta per le reti neurali, le cui scarse performance possono esssere motivate dalla ridotta dimensione del dataset, in quanto per produrre risultati soddisfacenti esse necessitano di una quantità maggiore di dati.

6 SHAP values analysis

Per capire a fondo l'importanza che i valori in input hanno nella determinazione dell'output ottenuto si effettua una analisi utilizzando i valori SHAP.

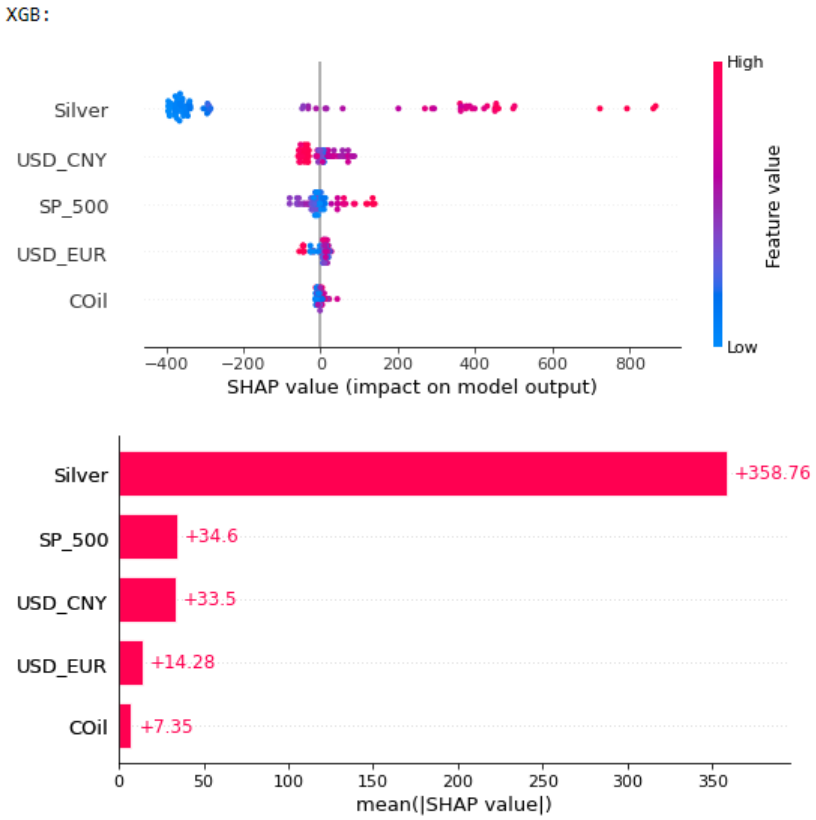
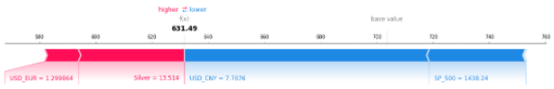


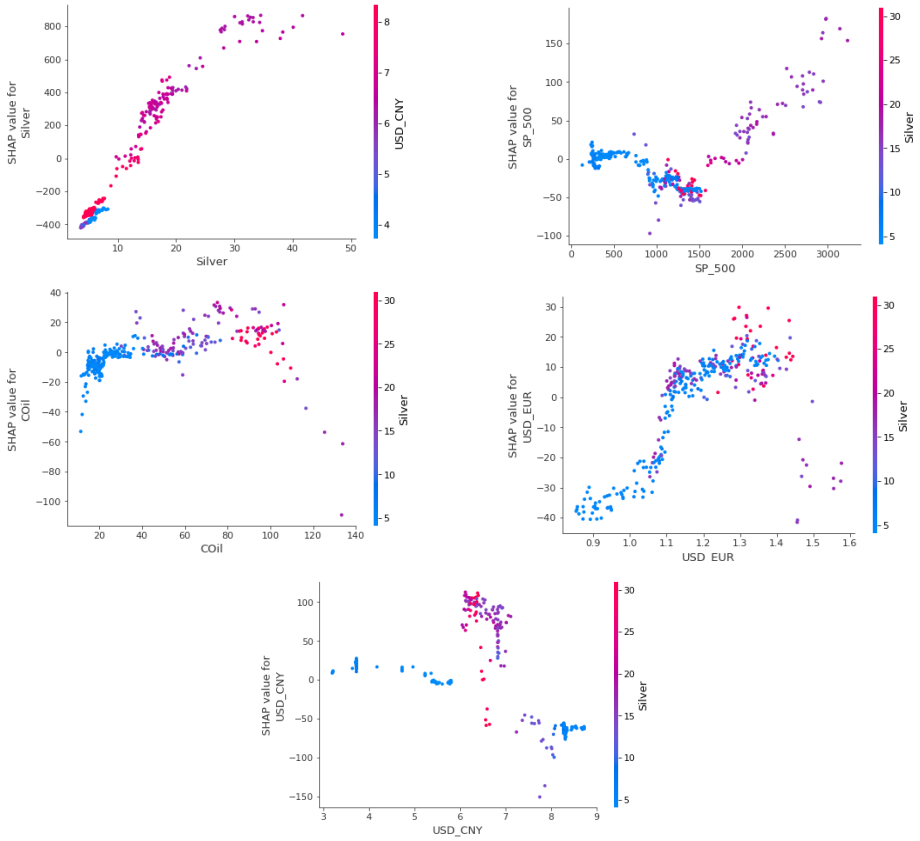
Figure 3: SHAP analysis relativa al modello XGB

Il primo grafico di cui sopra è il grafico riassuntivo SHAP del modello XGBoost. Da esso, per ogni dataset, possiamo determinare come esso influenzerà il risultato finale: il prezzo dell'argento, per esempio, sarà il più influente e un'incremento del suo prezzo comporta un aumento del prezzo dell'oro. Al contrario, il prezzo del petrolio, essendo tutte le sue feature prossime allo 0, sarà il fattore meno influente al fine della determinazione dell'output.
Il secondo grafico mostra l'importanza delle feature, senza esplicitare se l'influenza avviene in senso negativo o positivo.In tale grafico le caratteristiche sono ordinate in base a quanto hanno influenzato la previsione del modello e l'asse x rappresenta la media del valore SHAP assoluto di ciascuna caratteristica.
Il pacchetto SHAP ci consente, inoltre, di effettuare un'ulteriore analisi volta a determinare l'importanza marginale dei dati nella determinazione dell'output:



In tale grafico, che rappresenta la 105^A previsione di XGboost, i valori in rosso (USD-EUR e Silver) hanno un impatto positivo sulla predizione, ovvero la spingono verso valori più alti, mentre quelli segnati in blu (USD-CNY e SP_500) hanno un impatto negativo.
Un'ulteriore analisi che il pacchetto SHAP ci permette di effettuare consiste nei cosiddetti "*Dependence plots*". Tali grafici mostrno come il modello dipenda dalla caratteristica data e sono come un'estensione più ricca dei classici grafici di dipendenza paritale. La dispersione verticale dei punti dati rappresenta gli effetti di interazione.
I dati all'interno di tali grafici sono così disposti:

- Ogni punto è una singola previsione (riga) dal set di dati.
- L'asse x è il valore della funzione (dalla matrice X).
- L'asse y è il valore SHAP per quella caratteristica, che rappresenta quanto sapere il valore di quella caratteristica cambia l'output del modello per la previsione di quel campione.
- Il colore corrisponde a una seconda caratteristica che potrebbe avere un effetto di interazione con la caratteristica che stiamo tracciando (per impostazione predefinita questa seconda caratteristica viene scelta automaticamente). Se è presente un effetto di interazione tra quest'altra caratteristica e la caratteristica che stiamo tracciando, verrà visualizzato come un modello di colorazione verticale distinto.



Per esempio, nel caso del grafico che unisce le misurazioni di SP_500 in relazione al prezzo dell'argento, possiamo notare che bassi valori di SP_500, in correlazione a bassi valori del prezzo dell' argento, influenzano negativamente il prezzo dell'oro.

7 Conclusioni

In definitiva possiamo dire che, tramite l'impiego di tecniche di machine learning, è possibile predirre in maniera soddisfacente l'andamento del prezzo dell'oro dimostrando, inoltre, che i fattori presi in considerazione influenzano significativamente l'andamento del prezzo dell'oro.
Inoltre, questo studio presenta il metodo SHAP per unificare il campo dell'apprendimento automatico interpretabile. In effetti, la tecnica proposta fornisce una visualizzazione ricca di attribuzioni di caratteristiche individualizzate che migliora l'interpretabilità delle fluttuazioni del prezzo dell'oro.
Inoltre, Tree SHAP migliora la nostra comprensione dei modelli ad albero. Offre un mezzo approfondito per interpretare i risultati di un framework complesso come XGBoost e anche estrarre relazioni non lineari di funzionalità sull'output di un modello. Mostriamo come le spiegazioni additive di Shapley possono essere utilizzate per interpretare i risultati di XGBoost progettati per prevedere il prezzo dell'oro.
A livello pratico, gli investitori tramite l'utilizzo di modelli similari a quelli presentati potrebbero essere guidati in maniera migliore nelle loro scelte in modo da investire in maniera più sicura i loro soldi.
Lo studio presentato ha, però, una limitazione comune a tutti gli studi simili. Sebbene il modello proposto possa ottenere previsioni molto accurate, va anche riconosciuto che i mercati si basano su una serie di variabili, come le decisioni geopolitiche che possono risultare in movimenti imprevedibili.
Una eventuale ricerca futura potrebbe ampliare il nostro lavoro considerando variabili aggiuntive, come fattori politici o commerciali, nonché fasi di instabilità economica, che sono generalmente fattori determinanti del prezzo dell'oro, così da rendere le previsioni effettuate ancora più attendibili.

References

[1] R Beckmann J. Czudaj. "Gold as an infation hedge in a time-varying coefcient framework". In: *North American Journal of Economics and Finance* (2013). DOI: https://doi.org/10.1016/j.najef.2012.10.007.

[2] Kristjanpoller W. Minutolo M. C. "Forecasting volatility of oil price using an artifcial neural network-GARCH model". In: *Expert Systems with Applications* (2016). DOI: https://doi.org/10.1016/j.eswa.2016.08.045.

[3] Akbar M. Iqbal F. Noor F. " Bayesian analysis of dynamic linkages among gold price, stock prices, exchange rate and interest rate in Pakistan". In: *Resources Policy* (2019). DOI: https://doi.org/10.1016/j.resourpol.2019.03.003.

[4] Bhatia V. Das D. Tiwari A. K. Shahbaz M. Hasim H. M. "Do precious metal spot prices infuence each other? Evidence from a nonparametric causality-in-quantiles approach". In: *Resources Policy* (2018). DOI: https://doi.org/10.1016/j.resourpol.2017.12.008.

[5] Elie B. Naji J. Dutta A. Uddin G. S. "Gold and crude oil as safe-haven assets for clean energy stock indices: Blended copulas approach. " In: *Energy* (2019). DOI: https://doi.org/10.1016/j.energy.2019.04.155.

[6] Kanjilal K. Ghosh S. "Dynamics of crude oil and gold price post 2008 global fnancial crisis—New evidence from threshold vector error-correction model". In: *Resources Policy* (2017). DOI: https://doi.org/10.1016/j.resourpol.2017.04.001.

[7] Sami Ben Jabeur Salma Mefteh-Wali Jean-Laurent Viviani. "Forecasting gold price with the XGBoost algorithm and SHAP interaction values". In: *Annals of Operations Research* (2021). DOI: https://doi.org/10.1007/s10479-021-04187-w.