Streaming Data Management and Time Series Analysis

Shady Abd El Kader 838487

Introduzione

Il seguente progetto ha l'obiettivo di analizzare una serie storica sui prezzi del mercato energetico e generare previsioni per l'anno successivo.
In particolare sono stati utilizzati 3 tipologie di modelli: ARIMA (Autoregressive integrated moving average), UCM (Unobserved Components Model) e RNN (recurrent neural network).
Per ogni tipologia di modello sono stati testati diversi algoritmi e sono stati confrontati tra loro in termini di test set.
Il dataset a disposizione contiene i prezzi giornalieri del mercato energetico dal 01/01/2010 al 31/12/2018 e l'obiettivo richiesto era di ottenere una previsione per il periodo 01/01/2019 - 30/11/2019.
Dato che la previsione doveva partire dal 01/01/19 si è deciso di generare un test set che iniziasse il primo gennaio per ottenere un test set simile alla previsione desiderata, per fare ciò si è deciso di definire il test set il periodo 01/01/2018 - 31/12/2018 e come training set il periodo 01/01/2010 - 31/12/2017.

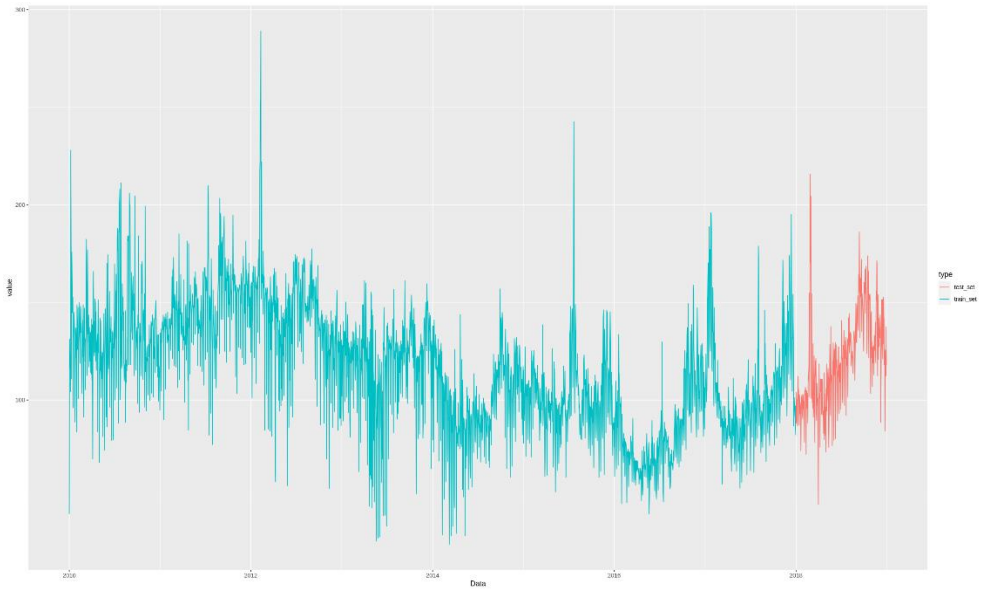

Figura 1 Dataset (training-set + test-set)

Gli algoritmi sono stati addestrati sul training set e a seconda dei risultati sul test set sono stati selezionati i migliori per ogni tipologia di modello.
Come misura per la qualità del modello è stato usato principalmente il MAPE (Mean absolute percentage error) per la sua semplicità di comunicazione, dato che il MAPE può portare a valori fuorvianti se il denominatore è una valore che tende a 0, il MAPE è stato affiancato da una misura di errore per i casi in cui i valori predetti potessero essere vicini allo 0, perciò è stato affiancato in alcuni casi dal MSE (Mean squared error).
I migliori algoritmi per ogni tipologia di modello sono stati poi utilizzati per generare le previsioni per il periodo 01/01/2019 – 30/11/2019.

ARIMA

Il primo modello utilizzato per la previsione dei prezzi elettrici è stato un modello ARIMA.
Per prima cosa si è osservato graficamente la serie storica (Figura 1) e si è notato un leggero trend decrescente e varianza non costante soprattutto in alcuni punti di picco della serie (questi potrebbero essere anomalie) ma non avendo abbastanza informazioni sul dataset si è preferito considerarli dati corretti della serie.
Si procede con analisi grafica delle trasformazioni sulla serie e con i test statistici per verificare la stazionarietà della serie (Figura 2):

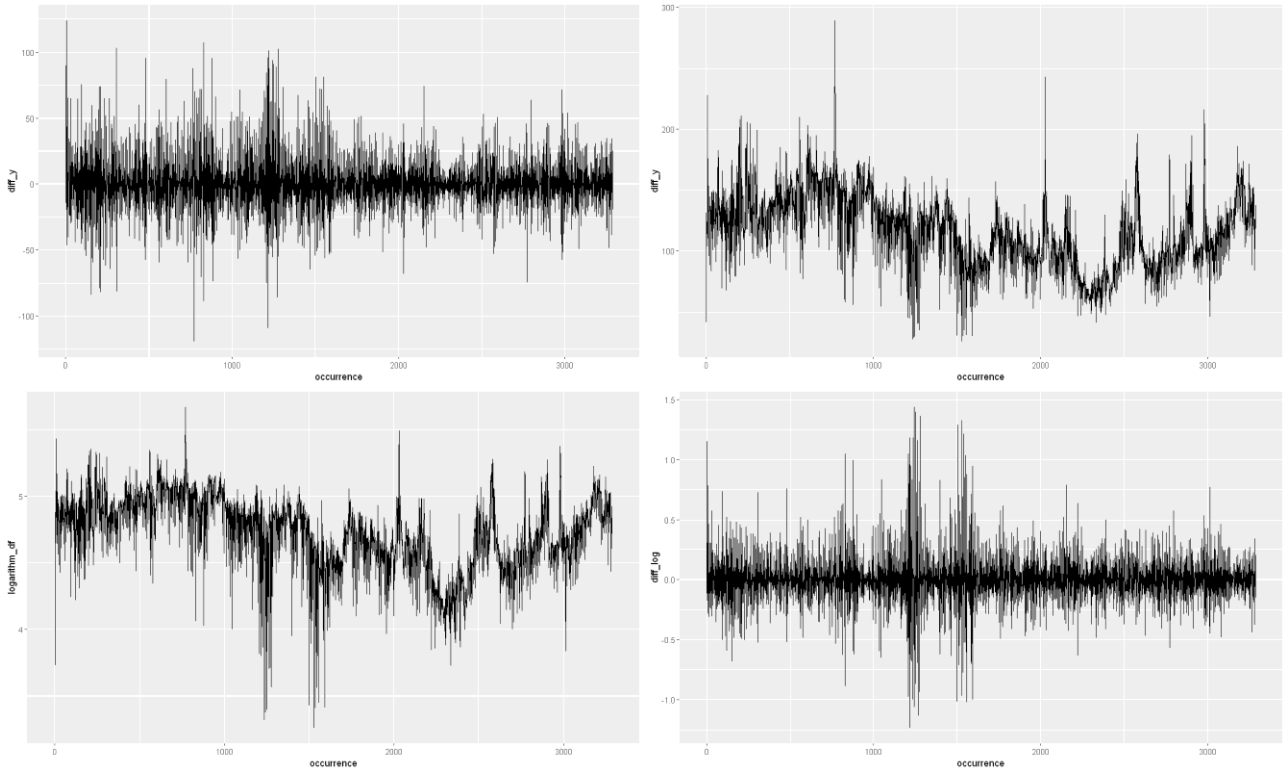

Figura 2 trasformazioni sulla serie storica (in ordine orario partendo dal grafico in alto a sinistra) 1) integrazione di primo grado, 2) nessuna trasformazione 3)trasformazione logaritmica 4) trasformazione logaritmica+ integrazione di primo grado

Ad una prima occhiata il grafico che più rispetta le condizioni di stazionarietà debole sembra essere il primo, si vuole quindi trovare conferma con un test statistico, per questo motivo viene usato l'Augmented Dickey-Fuller Test sulla serie originale. Il test rifiuta l'ipotesi nulla (almeno una radice caratteristica è uguale ad 1), si precede quindi a considerare la serie come stazionaria senza effettuare trasformazioni. Dato però che i grafici in (Figura 2) hanno restituito una serie storica che sembra essere più regolare, verrà valutata in seguito se considerare anche l'ipotesi in cui la serie sia non stazionaria valutando anche gli altri modelli.Per prima cosa si sono valutati i grafici di ACF (auto-correlation function) e PACF (partial auto-correlation function)

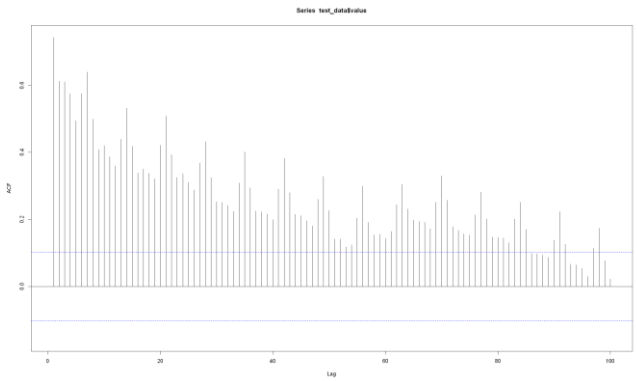

Figura 3 ACF serie storica

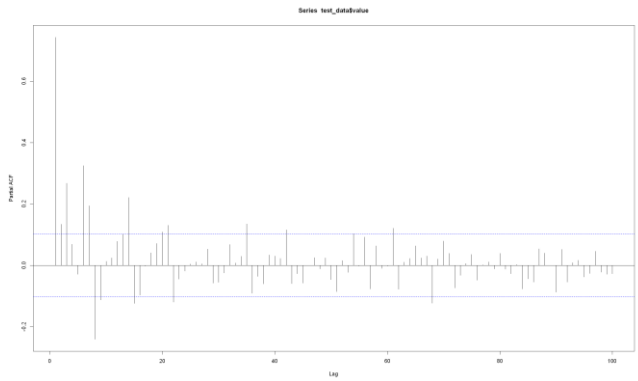

Figura 4 PACF serie storica

Si è notato subito una stagionalità a 7 giorni in quanto ogni 7 ritardi l'ACF cresce per poi decrescere a velocità geometrica.

Dopo aver inizialmente tentato con un modello SAR(1) SMA(1) con stagionalità=7 dai grafici aggiornati ACF e PACF sono stati notati alcuni ritardi periodici che ancora non rientravano nelle barre di confidenza, si è deciso di procedere quindi con modello SAR(1) SI(1) SMA(1)7.
A seguito del modello SAR(1) SI(1) SMA(1) S=7 alcuni ritardi risultavano ancora al di fuori delle barre di confidenza, quindi si è deciso di procedere con una grid search per valutare i migliori componenti di AR, e MA. A seguito però dei grafici in (Figura 2) si è deciso di valutare anche la serie storica integrata, logaritmica e integrata con l'applicazione del logaritmo, per queste ultime due si è deciso di utilizzare la funzione `auto.arima` su R per risparmiare sul costo computazionale.
Si sono confrontati i diversi modelli in termini di Training e test set, il modello migliore è risultato essere l'ARIMA (6,1,6) (1,1,1) 7 in quanto restituisce i migliori valori in termini di errore sia MAPE che MSE sia sul train che sul test set.

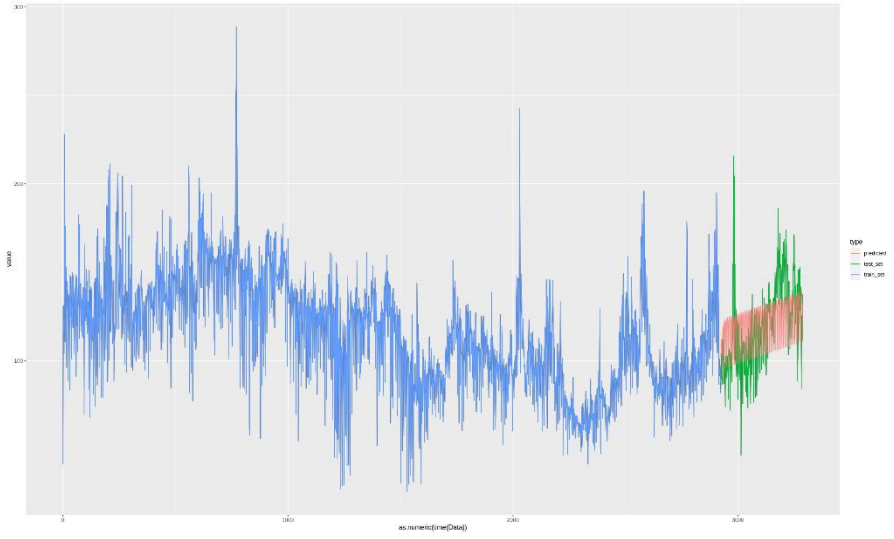

Figura 5 Train-set, test-set, previsione

Il modello ha restituito un MAPE sul training set del 9,65% e sul test set del 13,7%

UCM

Per i modelli UCM (Unobserved Components Model) si è deciso di iniziare dal LLT (local linear trend). Per fare ciò ci si è affidato al pacchetto "KFAS" di R e alla funzione SSModel.
Dato che nel modello ARIMA è stata evidenziata una stagionalità settimanale è stata aggiunta una

componente stagionale settimanale stocastica dummy, inoltre è stata aggiunta una componente stagionale annuale trigonometrica, per definire il miglior numero di armoniche possibile è stato usato un grid search per definire le armoniche comparando i risultati tramite l'indicatore MAPE, il numero di armoniche migliori è risultato essere 24, si è quindi scelto questo parametro per l'implementazione del LLT.

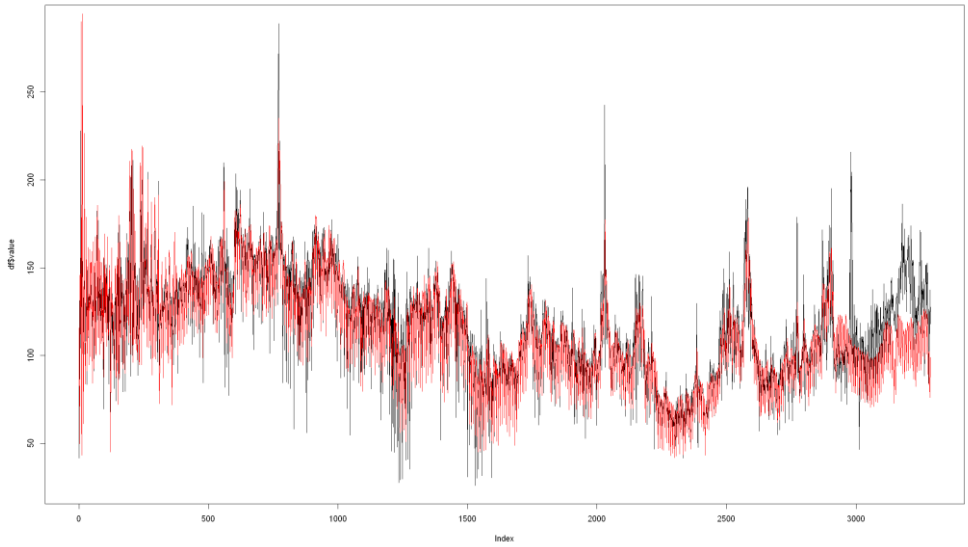

Figura 6 previsione con LLT sul dataset

Avendo notato graficamente dei valori più scostati rispetto al modello ARIMA è stato provato un modello RW (random walk), usando sempre una componente stagionale settimanale stocastica dummy e una componente annuale trigonometrica con 24 armoniche.

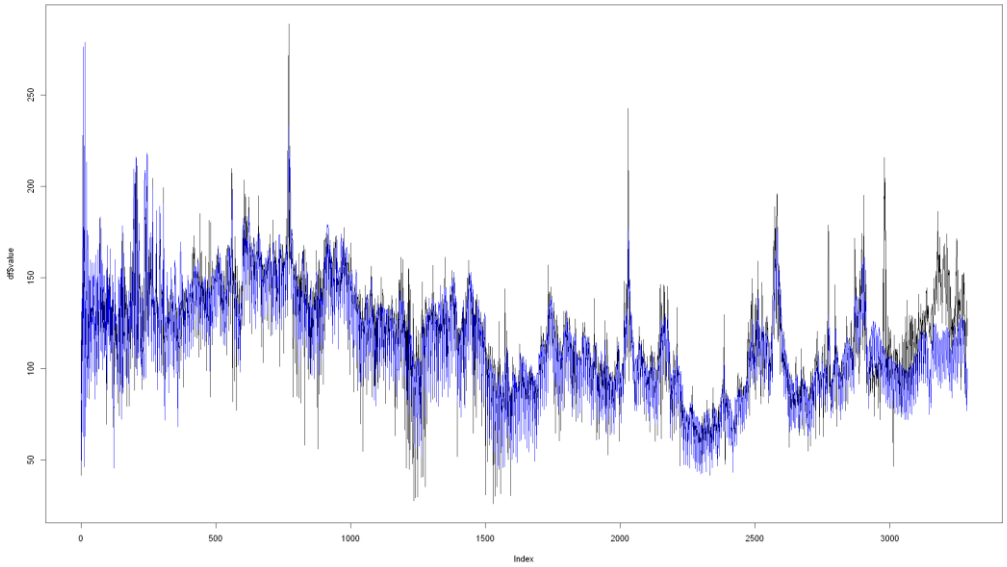

Figura 7 Random Walk sul dataset

Il risultato sul test-test risulta così migliorato a livello di MAPE, da 17,5% a 16,1%.
Il modello ha quindi restituito un MAPE sul training set del 7,6% e sul test set del 16,1% il che potrebbe significare in un over fitting sul training set. Si è quindi tentato di modellare un IRW tenendo conto dell'integrazione del modello ARIMA ma il risultato è peggiorato. SI è deciso quindi di mantenere il RW come miglior modello.

RNN

Per i modelli RNN (recurrent neural network) è stato usato la libreria Keras su python. Il pre-processing dei dati è stato effettuato normalizzando i valori attorno ad un range tra 0 e 1 tramite la funzione `MinMaxScaler` in modo da evitare eventuali instabilità dei pesi causati dal range di valori della serie storica non normalizzata. Avendo utilizzato un range tra 0 e 1, nel tuning del modello si è utilizzato il MSE invece che il MAPE, in quanto il MAPE avrebbe potuto portare ad errore dato che un denominatore uguale a 0 può far "esplodere" la frazione

Si è poi utilizzato un metodo sliding windows su 365 giorni in modo da utilizzare i dati dall'anno precedente per prevedere il nostro data point.

Per scegliere il modello ottimale si sono svolti diversi test con varie impostazioni del RNN e valutando i diversi approcci in termini di errore sulla previsione e osservando graficamente la rappresentazione (è capitato infatti che ad errori considerati "accettabili" corrispondesse una linea retta attorno alla media del test set)

Per scegliere le migliori impostazioni del RNN sono stati provati manualmente diversi set-up che variassero da 512 neuroni con metodo GRU (Gated recurrent unit) a 12 neuroni con metodo LSTM (Long short-term memory). Tendenzialmente il GRU è stato utilizzato per i set-up più onerosi dal punto di vista di neuroni in modo da alleggerire il carico computazionale della macchina.

Il metodo che ha restituito risultati migliori è stato un set-up con due layer LSTM ed uno denso con attivazione "sigmoid" per un totale di 25 neuroni su 100 epoche. Si è deciso inoltre di variare il numero di epoche a seconda del costo computazionale dell'algoritmo, benché questo possa non essere il metodo ottimale per valutare le diverse architetture, è risultato efficiente per comparare i diversi set up dal punto di vista performance/costo.

Il modello finale ha restituito un MAPE sul training set del 10% e sul test set del 14,6%

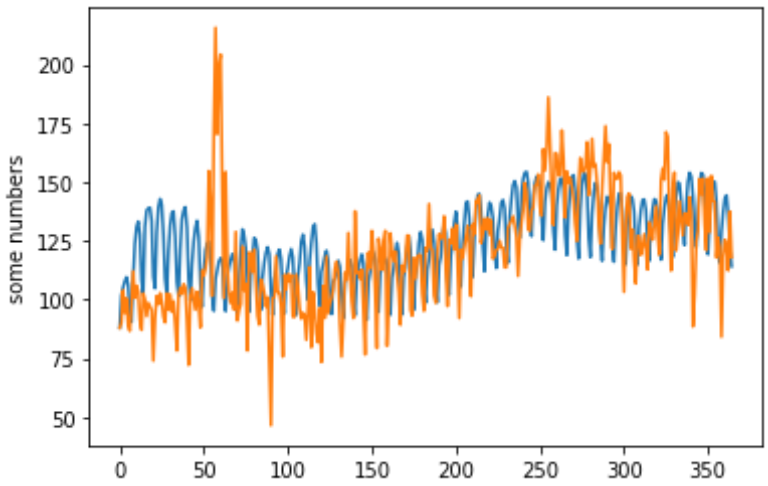

Figura 8 previsione LSTM su test set

Considerazioni finali

Comparando i risultati possiamo notare come l'ARIMA e il modello LSTM abbiano errori simili dal punto di vista di train e test set, mentre l'UCM come accennato in precedenza può aver subito un over fitting durante la modellizzazione.

Guardando il grafico delle previsioni ARIMA e LSTM si può però notare come il primo oscilli semplicemente attorno alla retta del trend che prevede, mentre il secondo oscilla attorno a quello che prevede essere l'andamento dei prezzi reali.

Tabella 1 Comparazione dei risultati in termini di MAPE

	ARIMA	UCM	LSTM
TRAIN SET	9,65%	7,6%	10%
TEST SET	13,7%	16,1%	14,6%

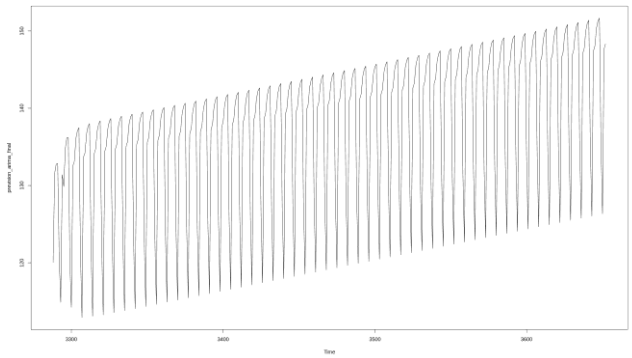

Figura 9 Previsione ARIMA

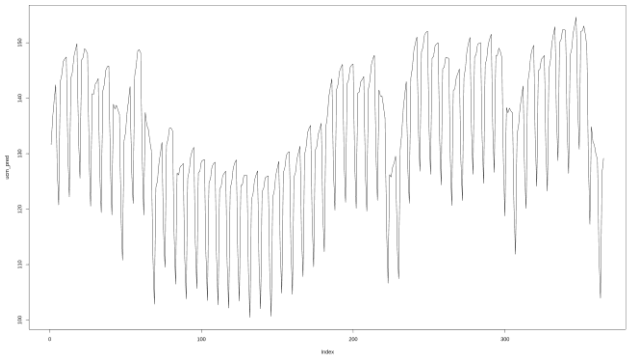

Figura 10 Previsione UCM

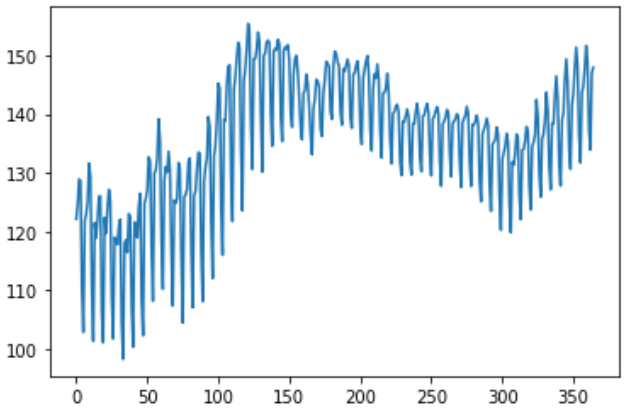

Figura 11 Previsione LSTM

Credo quindi che il modello ARIMA possa performare meglio su dati con un livello di aggregazione maggiore in cui lo scostamento dal trend è meno frequente e meno casuale. Per il modello UCM siccome i valori training e test set sono molto simili per il LLT e il RW (e addirittura peggiori per il IRW) potrebbe significare la necessità di riprovare il modello con parametri diversi al fine di trovare un set-up migliore che restituisca valori di errore non troppo distanti tra training set e test set.
Per quanto riguarda il modello di Machine Learning ci può essere difficolta nel trovare il numero di neuroni e layer ottimale ma la semplicità del pacchetto Keras permette di effettuare modifiche e generare un nuovo modello di machine learning in pochi passaggi, questo permette una grande flessibilità e semplicità nel generare modelli diversi e aggiustare il tiro per ottenere risultati migliori.