Relazione di fine tirocinio

Tags	Report
Select	Sent

Parte 1 → Studio:

Il percorso di tirocinio ha avuto inizio con un periodo iniziale di studio e approfondimento, utilizzando materiali forniti dal Professor Podda e altri reperiti online, relativi agli argomenti di interesse. Inizialmente, si è approfondita l'architettura e il funzionamento del servizio cloud Azure attraverso la visione di video mirati, al fine di comprendere le parole chiave fondamentali per l'utilizzo del servizio. Successivamente, si è proceduto con una ricerca più dettagliata su tre macroargomenti specifici:

- Forecasting
- Tecniche basate sugli LLM per la previsione delle serie temporali
- Approfondimento di Superset

Per quanto riguarda il primo e il terzo argomento, è stato possibile reperire una vasta mole di materiale. Tuttavia, per il secondo argomento, è stato più difficile trovare risorse su cui basarsi per sperimentare tecniche di questo tipo.

Parte 2 → Superset:

Nella seconda parte del percorso, oltre allo studio continuo e alla ricerca per approfondire gli argomenti principali, si è dedicato tempo allo studio del software open

source di visualizzazione dati, Superset. Inizialmente si è sospettato che ci fossero problemi riguardanti l'esattezza dei dati visualizzati. Le dashboard e i grafici creati con questo software non corrispondevano perfettamente ai valori e ai grafici standard di visualizzazione dei costi su Azure. Sono state eseguite delle prove utilizzando il dataset dei costi cloud di VisioScentiae, che sembravano non essere corretti nonostante la capacità di ricreare perfettamente le dashboard di Azure su Superset.

Per verificare eventuali errori nell'esportazione dei dati dal provider cloud, si è provato ad utilizzare un altro account Azure di prova con un saldo demo. In questo caso, sono stati ottenuti costi equivalenti in tutte le chart.

Dopo vari tentativi, è emerso che, come precedentemente ipotizzato, l'errore era legato all'esportazione dei dati da Azure. Il problema si presentava costantemente nell'ultimo giorno dei dati; escludendo quest'ultimo, i calcoli tornavano corretti. Pertanto, si è deciso di posticipare l'esportazione dei costi a un momento successivo durante la giornata, considerando che alcuni dati potrebbero non essere stati ancora salvati e inclusi insieme a quelli dei giorni precedenti.

Parte 3 → Forecasting:

Dopo aver risolto il problema con Superset, ho avviato la successiva fase del mio lavoro: iniziare a fare previsioni sui costi cloud a mia disposizione. Inizialmente ho utilizzato i dati del cloud di VisioScentiae, ma questi mostravano una varianza minima. Anche dopo aver testato diversi modelli di machine e deep learning, non si sono ottenute previsioni sufficienti in caso di variazioni improvvisi dei costi.

Il principale problema di questa fase è stato, fino alla fine, la scarsità di dati. Ho provato tre diversi dataset, ma nessuno di essi copriva un periodo di costi superiore ai 100 giorni. Dopo aver effettuato le prime previsioni con il dataset di Visio, ho iniziato a utilizzare altri dati disponibili pubblicamente su Internet. Si tratta di un dataset che comprende circa 90 giorni di costi di un'Azure subscription anonima. Questi dati sembravano più promettenti sin dall'inizio, principalmente per la presenza di un maggior numero di categorie da cui provenivano i costi. Queste categorie si sarebbero poi rivelate utili nelle strategie di previsione.

In particolare, il dataset in questione è reperibile su Kaggle con il nome di 'Azure Subscription Costs'. Da subito ha mostrato una maggiore varianza tra i dati, rendendoli più facilmente prevedibili attraverso modelli di forecasting.

Modelli e tecniche testate:

Nel corso dell'intero periodo dedicato a questo compito, ho testato un vasto numero di modelli e tecniche al fine di comprendere e studiare quale fosse la soluzione ottimale. Ho iniziato con modelli già noti e ampiamente testati, come XGBoost, per poi esplorare nuove tecniche che integrano previsioni su singole categorie in modo intelligente.

> Le due architetture principali testate sono state XGBoost, utilizzata come baseline iniziale, e le reti neurali LSTM, particolarmente adatte per lavorare su serie temporali.

In ogni prova, i dati forniti ai modelli consistevano nei costi dei giorni precedenti, insieme ad altre informazioni temporali utili per aiutare il modello a predire, come il giorno della settimana, il mese o il giorno dell'anno. Nel caso in cui avessi avuto a disposizione un dataset più esteso, coprendo anni di costi, sarebbe stato vantaggioso includere ulteriori informazioni come l'anno o i trimestri. Ho sperimentato diverse configurazioni sul numero di giorni precedenti considerati dal modello, al fine di individuare la configurazione ottimale per le diverse architetture testate.

Le principali metriche valutative considerate sono state due:

- RMSE (Root Mean Square Error): una metrica standard per valutare le prestazioni di una regressione.
- Direzionalità dell'accuratezza (Directional accuracy): una metrica che misura se il modello prevede correttamente un aumento dei costi, così come si manifesta nei valori reali. Questa metrica è utile per penalizzare i modelli che semplicemente traslano indietro di un giorno l'andamento dei valori reali.

XGBoost Base:

Inizialmente, i parametri utilizzati sono stati quelli standard, provenienti da un tutorial per la creazione di notebook volti all'addestramento di modelli per il time series forecasting. Successivamente, al fine di migliorare le prestazioni secondo le metriche considerate, si è proceduto con il tuning degli iperparametri. Per il tuning, si è impiegata la libreria HyperOpt, la quale mira a minimizzare una metrica specifica (in questo caso, l'RMSE), esplorando numerose combinazioni di parametri per individuare quelli che minimizzano tale metrica. Utilizzando tali parametri ottimizzati, è stato quindi addestrato il modello.

```
#PARAMETRI PROTAGONISTI DEL TUNING

parameters = {'colsample_bytree': ,
 'gamma': ,
 'learning_rate': ,
 'max_depth': ,
 'min_child_weight': ,
 'n_estimators': ,
 'subsample': }
```

Possiamo dire senza dubbio che tra le due prove quello che funziona quasi sempre meglio è la seconda, ovvero il modello con i parametri tunati sia dal punto di vista dell’RMSE che, ed è quello in cui si differenziano maggiormente, nella Directional Accuracy.

LSTM:

Durante i test con le reti LSTM, sono state eseguite diverse prove iniziali variando le architetture, tuttavia mantenendo un numero ridotto di layer. Tra le diverse sperimentazioni sono state incluse l'utilizzo di livelli bidirezionali LSTM e l'impiego di livelli convolutivi precedenti ai livelli LSTM per la trasmissione dei dati. Tuttavia, nelle prove più recenti con i risultati più soddisfacenti, si è scelto di incrementare le dimensioni delle reti LSTM, riducendo al contempo il numero di livelli dello stesso tipo con un numero decrescente di unità (256, 128, 64, 64).

```
model = Sequential()
model.add(InputLayer((9, 1)))
model.add(LSTM(256, activation='relu', return_sequences=True))
model.add(LSTM(128, activation='relu', return_sequences=True))
model.add(LSTM(64, activation='relu', return_sequences=True))
model.add(LSTM(64, activation='relu'))
model.add(Dense(1))
```

Per tutte le diverse prove condotte, l'architettura principale è stata sostanzialmente simile: un maggior numero di livelli consente al modello di riconoscere più efficacemente una vasta gamma di pattern nei dati durante l'addestramento. Se ci fossero stati meno livelli, l'addestramento sarebbe stato più rapido ma molte informazioni sarebbero potute andare perdute.

Ogni approccio è stato testato considerando diverse quantità di giorni precedenti su cui basare le previsioni (2-4-6). Inoltre, sono stati condotti 10 addestramenti e tra questi è stato selezionato il modello sulla base della directional accuracy sul set di validazione. L'approccio di addestramento è stato di tipo walk-forward, con 5 passaggi: ogni passaggio comprendeva rispettivamente 50, 10 e 5 dati per i set di training, validation e test. Le valutazioni delle metriche sono state ottenute dalla media delle metriche in ciascuno dei 5 passaggi nell'addestramento.

Oltre ai giorni precedenti, i modelli hanno ricevuto informazioni temporali, come il mese e il giorno della settimana da predire. Durante l'addestramento di ciascun approccio, è stata utilizzata una metrica e una loss personalizzate che considerano sia l'RMSE che la directional accuracy, permettendo un possibile aumento del peso di una delle due metriche per conferirle maggiore importanza. È importante notare che durante gli addestramenti, i pesi sono rimasti equamente distribuiti tra le due.

Per garantire un approccio il più robusto possibile, sono state effettuate 4 prove con le architetture LSTM:

1. Modello univariato basato sui costi totali giornalieri (somma di tutte le categorie).
2. Modello multivariato basato sui costi delle singole categorie, effettuando previsioni separate per ciascuna categoria e sommandole per ottenere il costo totale previsto.
3. Modello multivariato simile al precedente, ma invece di sommare i costi, sono stati utilizzati i risultati di un metamodello che unisce intelligentemente le previsioni di tutte le categorie.
4. Modello multivariato che considera i singoli valori di tutte le categorie insieme, fornendo il costo totale senza una fase di somma separata, poiché un singolo modello analizza tutti i valori delle diverse categorie.

Per quanto riguarda la divisione in categorie, il dataset conteneva costi relativi a più di 50 categorie, ma solo 8 di queste erano presenti nei dati per più del 90% dei giorni. Di conseguenza, ho scelto di effettuare previsioni separate solo per le categorie presenti quasi tutti i giorni e aggregare i costi delle categorie meno frequenti in una singola previsione.

Tra i 4 approcci, il terzo è stato quello su cui ho concentrato maggiormente l'attenzione, in quanto il problema principale risiedeva nella ricerca di un metamodello che massimizzasse contemporaneamente le due metriche in questione (RMSE e DA). I modelli testati sono stati:

- Stessa architettura LSTM utilizzata per l'addestramento e le singole previsioni.
- Rete neurale con più livelli dense sovrapposti.
- Modello XGBoost di base.
- Modello XGBoost con tuning degli iperparametri.

Tra i 4 approcci, quello che ha ottenuto i risultati migliori è stato l'ultimo, in quanto il tuning dei parametri è stato eseguito sui dati utilizzati per l'addestramento e mirato a minimizzare sia l'RMSE che la DA.

Parte 4 → Risultati e conclusioni:

In conclusione ho creato delle tabelle contenenti tutti i valori delle due metriche dati da tutte le prove che sono state fatte con i diversi modelli:

Models and approaches	RMSE	Directional Accuracy
XGB base	17.450	0.26
XGB tuned	14.292	0.63

È importante notare che nella tabella, i valori forniti dal modello XGBoost con il tuning degli iperparametri rappresentano i risultati ottenuti dall'addestramento migliore o dai migliori parametri. Durante le varie prove, sono stati ottenuti anche risultati minori, ma si è scelto di selezionare il miglior risultato tra tutte le prove. Questo principio si applica anche a tutti gli approcci LSTM, in cui sono stati eseguiti diversi addestramenti. Nella tabella, sono presentati i valori migliori sul test set, selezionati in base all'addestramento con il risultato migliore ottenuto sul validation set.

Giorni indietro visibili ai modelli	2 Days	4 Days	6 Days
LSTM Approach 1	RMSE = 11 DA = 0.64	RMSE = 14,1 DA = 0.48	RMSE = 17,3 DA = 0.48
LSTM Approach 2	RMSE = 15,44 DA = 0.68	RMSE = 17,2 DA = 0.56	RMSE = 14,63 DA = 0.64
LSTM Approach 3 (XGB TUNED, 2 days) (NN with 2 dense layer, 4 days) (NN with 2 dense layer, 6 days	RMSE = 14,1 DA = 0.75	RMSE = 12,85 DA = 0.68	RMSE = 15,30 DA = 0.68

LSTM Approach 4	RMSE = 12,9 DA = 0.52	RMSE = 22 DA = 0.48	RMSE = 18,28 DA = 0.52

Inoltre è importante notare che, in ciascuno degli approcci, il risultato presentato è quello ottenuto dall'addestramento migliore, che ha mostrato valori più elevati soprattutto per quanto riguarda la directional accuracy. Riguardo al metamodello, in particolare, i risultati, soprattutto nelle prove con 4 e 6 giorni, sono stati migliori, ma in media non risultano altrettanto soddisfacenti, attestandosi intorno al 0.6 di accuratezza.

L'approccio che porta alle prestazioni migliori in termini di directional accuracy è quello che utilizza il metamodello. Tuttavia, è importante notare che il metamodello XGBoost con Hyperparameter Tuning sembra comportarsi meglio in più casi. Quando il tuning è effettuato basandosi anche sul test set, i risultati sono migliori. Tuttavia, mostrando solo i dati di addestramento e di validazione (durante la scelta dei parametri), i risultati peggiorano all'aumentare del numero di step. Inoltre, l'approccio con i costi sommati sembra funzionare in modo particolarmente efficace, ma non sembra prevedere altrettanto bene un repentino aumento dei prezzi, aspetto che invece viene spesso predetto in modo soddisfacente dal metamodello.