Predizione dell'andamento del prezzo dell'oro

Ragazzi Luca | Programmazione di Applicazioni Data Intensive | 09/07/2018

INDICE

INTRODUZIONE

In questo progetto lavoro su una serie temporale (il prezzo dell'oro), dove si ha una variabile Y (t) che varia in base al tempo t.

SCOPO DEL PROGETTO

Scopo del progetto è addestrare un modello di predizione in grado di prevedere l'andamento del prezzo dell'oro in una giornata Y (t), che varia in base giornaliera, dati i prezzi dei giorni precedenti Y (t – 1), Y (t – 2), etc.

ORGANIZZAZIONE DEL TESTO

Nel capitolo 2 introduco il progetto e gli approcci adottati per arrivare alla soluzione. Il capitolo è suddiviso da una parte iniziale teorica che spiega la strategia utilizzata, seguita da una parte una pratica dove commento le porzioni di codice più importanti.

Nel capitolo 3 mostro i vari test effettuati per raggiungere l'obiettivo e confronto i risultati ottenuti con i vari metodi utilizzati.

Nel capitolo 4 traggo le conclusioni finali e parlo degli eventuali sviluppi futuri del progetto.

Nel capitolo 5 cito i diversi siti web che ho usufruito per giungere alla soluzione o per chiarimenti durante il percorso progettuale.

CASO DI STUDIO

CONSIDERAZIONI INIZIALI

Lo storico dei dati sul prezzo dell'oro li ho reperiti sulla seguente pagina web: "https://www.gold.org/data/gold-price", dove è possibile scaricare i dati in forma di file Excel. In questo file sono presenti diverse tabelle che memorizzano i valori del prezzo in diversi formati: giorno, settimana, mese, anno.

Per questo specifico caso di studio ho deciso di utilizzare il dataset dei prezzi settimanali, perché ho pensato sia più semplice analizzare l'andamento del prezzo su questa scala, rispetto per esempio a quella giornaliera (dove possono verificarsi troppe variazioni non importanti) e quella mensile/annuale (pochi dati su cui effettuare l'addestramento e la valutazione).

APPROCCIO ADOTTATO – PARTE CONCETTUALE

Ho deciso di trasformare il dataset dei valori del prezzo dell'oro in un dataset dove ogni valore è sostituito dalla differenza col valore della settimana precedente.

Ho adottato questa strategia perché ho pensato che è più importante prevedere se il prezzo dell'oro aumenterà o diminuirà in futuro, anziché conoscere il valore effettivo.

Per raggiungere l'obiettivo prefissato ho addestrato il mio modello con diverse tecniche di regressione del Machine Learning, tra cui regressione lineare, regressione polinomiale e regressione con metodi kernel. Per ognuno ho valutato l'accuratezza della previsione e confrontato i risultati ottenuti, al fine di capire quale modello riesca a raggiungere un'accuratezza maggiore.

LA REGRESSIONE NEL MACHINE LEARNING

Obiettivo dell'analisi di regressione è l'addestramento di un modello che, dati i valori di una o più variabili indipendenti X_1, X_2, ..., X_n, riesca a prevedere una variabile dipendente Y non nota a priori.

Nella regressione lineare il valore di Y è stimato essere una combinazione lineare dei valori delle variabili indipendenti X. Con l'analisi di regressione vanno ricercati i valori dei coefficienti che minimizzano l'errore.

Nella regressione polinomiale vengono aggiunti, a quella lineare, termini di grado superiore, per far si che il modello approssimi meglio i dati (se questi non sono approssimabili con una funzione lineare, ovvero se la variabile da predire non ha un andamento monotono rispetto alle X).

Come detto in precedenza, la regressione polinomiale comporta l'introduzione di nuove variabili per rappresentare i dati (aggiunta di termini di grado superiore), aumentando però la complessità computazionale, per cui se si vuole utilizzare un

polinomio con un grado elevato bisogna tener presente che i tempi di calcolo aumentano sensibilmente.

Una soluzione al problema sopra illustrato è l'utilizzo dei metodi kernel, che in pratica ottengono un modello non lineare, su cui lavorare, senza però aumentare la dimensione del problema (permettono di calcolare i prodotti scalari tra vettori nello spazio trasformato direttamente sui dati originali). Questo risolve indubbiamente il problema dell'elevato tempo di calcolo della regressione polinomiale.

APPROCCIO ADOTTATO: IL CODICE

Nell'applicazione sviluppata ho scritto diversi metodi utili alla comprensione del problema:

Per come è strutturato il file Excel che ho scaricato, il metodo che ho valutato migliore per caricare i dati è il seguente:

- ```
  def get_data(filename, sheetname):
      dataset = pd.read_excel(
          filename,
          sheet_name=sheetname,
          index_col=3,
          usecols=5,
          skiprows=8,
      ).dropna(axis=1)
      return dataset
  ```

Con il seguente codice carico i dati nella variabile "gold" ed estraggo solo la colonna relativa agli "Euro", importandola come una Series di pandas:

- ```
  gold = get_data("gold_price.xls", "Weekly_EndofPeriod")
  gold = gold["Euro"]
  ```

Per mostrare alcuni grafici sull'andamento del prezzo ho scritto due metodi; il primo mostra semplicemente il grafico del nostro dataset, mentre il secondo mostra il grafico in un lasso di tempo specifico:

- ```
  def show_plot(dataset):
      dataset.plot(title="Grafico dell'andamento del prezzo dell'oro dal 1979 al
      giorno d'oggi", figsize=(12, 4))
      plt.ylabel("Euro")
      plt.show()
  ```

- ```
  def show_period_plot(i_date, f_date, dataset):
      dataset[i_date: f_date].plot(style="o-", figsize=(12, 4))
      plt.ylabel("Euro")
      plt.show()
  ```

La strategia incomincia alla creazione di una nuova variabile "gold_diff", a cui assegno un nuovo DataFrame composto dalla differenza di prezzo da una settimana all'altra:

- ```
  gold_diff = gold.diff()
  ```
  ```

Ho utilizato 2 metodi “extract_features” e “accuracy”, visti a lezione, per creare rispettivamente 2 DataFrame su cui addestrare il modello e una funzione per il calcolo dell’accuratezza. Il metodo “accuracy” verifica se il valore predetto e quello originale hanno lo stesso segno, ovvero se riusciamo a predire in maniera corretta se si verificherà un aumento o un calo del prezzo in futuro.

- ```
  def extract_features(dataset, weeks):
      y = gold_diff
      X = pd.DataFrame({"col{}".format(i): dataset.shift(i)
          for i in range(1, weeks+1)}).dropna()
      y = dataset.reindex_like(X)
      return X, y
  ```

- ```
  def accuracy(model, X, y):
      return (np.sign(y) == np.sign(model.predict(X))).mean()
  ```

Successivamente ho addestrato il mio modello con diverse tecniche di regressione che riporto:

- Regressione lineare
- Regressione polinomiale
- Regressione lineare con regolarizzazione
- Regressione polinomiale con regolarizzazione
- Regressione con metodi kernel

Ho estratto dal mio DataFrame “gold_diff”, tramite il metodo “extract_features” due variabili per il training set e due per il validation set (utilizzando circa l’80% dei dati per l’addestramento e il restante 20% per la validazione), e per tutti i metodi di regressione ho addestrato il modello e valutato l’accuratezza della previsione col metodo “accuracy”.

TEST SPERIMENTALI E RISULTATI

OBIETTIVI DEI TEST

Ogni modello di regressione è composto da parametri e da iperparametri. I parametri sono quei coefficienti che moltiplicano ciascuna variabile, più l'eventuale intercetta, e sono ottimizzati tramite l'addestramento del modello sul training set. Gli iperparametri sono quei valori che imposta manualmente l'utente, per cui non sono trovati o ottimizzati durante il training set. Tra questi iperparametri posso ricordare il grado del polinomio, i parametri dei metodi kernel o il peso della regolarizzazione. Per ottenere l'accuratezza maggiore della predizione ho utilizzato un metodo, visto a lezione, per scoprire quale è la miglior combinazione degli iperparametri che porta al miglior risultato.

Per questo scopo ho utilizzato una classe fornita da "scikit-learn": GridSearchCV. Essa esegua la cross-validation (diverse combinazioni per testare un modello) su tutte le possibili combinazioni di un insieme di parametri dati.

Tramite la GridSearchCV ho testato quale fosse la miglior combinazione dei parametri per le varie tipologie di regressione, tramite "gs.best_params_" e "gs.best_score_".

Per esempio, per la regressione con metodo kernel di tipo RBF la combinazione migliore è alpha=0.1 e gamma=0.01.

L'accuratezza maggiore è risultata del 52.16% con la regressione polinomiale con regolarizzazione di grado 4. Ora elenco tutte le accuratezze sul modello con i vari metodi:

- Accuratezza regressione lineare: 48.15%
- Accuratezza regressione polinomiale: 49.85%
- Accuratezza regressione lineare con regolarizzazione: 48.15%
- Accuratezza regressione polinomiale di grado 4 con regolarizzazione: 52.16%
- Accuratezza regressione con metodo kernel polinomiale di grado 8: 48.53%
- Accuratezza regressione con metodo kernel RBF: 49.37%

CONCLUSIONI E SVILUPPI FUTURI

Concludendo questa esperienza posso dire che non sono riuscito a trovare un modello di previsione con una buona accuratezza (infatti il migliore risulta essere del 52.16%), d'altro canto probabilmente non ho testato il modello con tutti i metodi possibili o con tutte le combinazioni degli iperparametri.
La mia applicazione web mostra all'utente solo le tabelle relative al dataset in diverse configurazioni e il grafico dell'andamento del prezzo.
Un importante sviluppo per il futuro del progetto è indubbiamente un'applicazione web dove l'utente può interagire dinamicamente col dataset e magari poter fare anche previsioni in live cambiando eventualmente anche i parametri dei modelli utilizzati o utilizzarne nuovi.

BIBLIOGRAFIA

Per lo svolgimento del progetto ho consultato le slide date a lezione e diversi siti sulla documentazione ufficiale di pandas e matplotlib che riporto qui sotto:

- https://matplotlib.org/api/_as_gen/matplotlib.pyplot.ylabel.html
- https://matplotlib.org/api/_as_gen/matplotlib.pyplot.plot.html
- https://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.read_excel.html
- https://www.gold.org/data/gold-price