Spolaor Riccardo 864877
Web Intelligence Predict 2020

 # Scrivi che inizialmente hai tolto righe nel filling missing values, poi hai deciso di tenerle

Introduzione

Il progetto si prefigge di predirre l'outcome del torneo di tennis Australian Open 2020 definendo un modello di predizione a partire dai dati delle singole partite forniti dal sito http://www.tennis-data.co.uk/ e simulando la contesa match per match.
Ho deciso di impostare il lavoro in parti distinte, dividendo i singoli processi di analisi in vari Notebook affinché fosse chiara la visualizzazione. Sono stati creati dei file Python separati all'interno del folder `./python files` per gestire funzioni che vengono richiamate più volte o che vengono utilizzate in Notebook differenti e anche per funzioni contenenti molte righe di codice.

00 Dataset Download and Examination

Il sito http://www.tennis-data.co.uk/ contiene i dati dei match suddivisi per anno all'interno di file nel formato .csv, in ogni file i tornei sono ulteriormente ordinati per data. Ho deciso di ignorare il file riguardante l'anno 2000 in quanto non conteneva, al contrario degli altri, informazioni riguardo agli odd dei bookmaker, dato ritenuto fondamentale per la predizione.
Ho importato i singoli file in ordine di data in oggetti DataFrame della libreria `pandas` scaricandoli direttamente da internet e aggiungendo una feature `csvID` progressiva per distinguerli. Ho poi unito ogni DataFrame, mantenendo l'ordine temporale dei match.

In seguito ho svolto un lavoro preliminare di Data Pre-Processing, riscontrando che alcune feature contenenti valori numerici erano state assegnate erroneamente al tipo object e sono intervenuto correggendo le anormalità.
Dando seguito al processo di analisi ho rilevato la presenza di dati sporchi o non corretti per le feature Winner e Loser, che ho corretto eliminando gli spazi vuoti riscontrati prima e dopo i nomi, trasformando tutte le stringhe in maiuscolo e procedendo ad altre correzioni manuali.

Ho infine realizzato i grafici delle occorrenze dei valori per ogni feature categoriale per capire come potessero essere successivamente gestite.

01 Data Cleaning – Missing Values Filling

Il passo successivo è stato quello di riempire i valori mancanti del dataset. Le feature contenenti valori mancanti erano quelle del Rank del vincitore e del perdente, dei punti del vincitore e del perdente, degli odd dei bookmaker e dei game e i set vinti dei contendenti.
Prima di procedere mi sono accorto tramite un'analisi delle statistiche di queste feature che `MaxL` conteneva un valore massimo anomalo e la sua deviazione standard era molto alta. Ho quindi modificato i valori anomali con *NaN* in modo da poterli riempire con dati più accettabili nei passi successivi.

Riscontrato che molte odd dei singoli bookmaker erano mancanti ho deciso di tenere solo le feature che specificavano la scommessa massima e media dei bookmaker. Per ogni feature a cui ho inferito i missing values, eccetto che per i game ed i set vinti dai giocatori, è stata aggiunta una nuova feature contenente valori binari per indicare la precedente mancanza del dato.
Inizialmente avevo deciso di riempire i valori del Rank e dei Punti del vincitore e del perdente utilizzando la media del valore di quella feature per quel giocatore e di eliminare ogni riga rimanente che contenesse ancora valori *NaN*. Ho poi invece deciso di riempire i valori rimanenti dopo il primo filling con il valore del Rank massimo + 1 e il valore dei Punti minimi – 1, ipotizzando che nel caso in cui il giocatore non avesse mai Rank o Punti assegnati doveva essere una new entry.
Per quanto riguarda i dati mancanti degli odd massimi e medi dei bookmaker, avevo deciso di riempirli sfruttando i valori delle scommesse dei singoli bookmaker per quel giocatore per quella specifica partita, qualora fossero presenti, e poi eliminare le righe che contenevano ancora dati *NaN*. Ho sostituito in seguito l'eliminazione delle righe ad un ulteriore filling sfruttando la media dei valori di quella feature per tutti i match.
Ho infine riempito i valori mancanti dei game e dei set vinti dal giocatore con il numero 0 per indicare che quel giocatore non aveva vinto nessun game o set. L'unico valore dei set vinti dal vincitore di una partita completata (secondo la feature `Comment`) era di un match al meglio di 5 ed è stato riempito con il valore 3.
Il motivo per cui ho deciso, in un secondo momento, di non eliminare nessuna riga, ma di procedere al filling di tutti i valori mancanti è dovuto al fatto che l'accuracy delle previsioni testata nei passi successivi mantenendo il dataset con tutti i match risultava maggiore rispetto all'accuracy delle previsioni eliminandone alcuni.

02 Data Cleaning – Categorical Data Transformation

In questo passo del processo di analisi ho gestito la trasformazione delle variabili categoriali perché fossero successivamente utilizzabili dai modelli di previsione, senza riscontrare particolari criticità.

Dummy Features

Ho trasformato in binaria l'unica feature contenente due soli valori:

- `Court`.

One-Hot-Encoding

Ho utilizzato la seguente tecnica per trasformare feature categoriali n-arie prive di relazioni gerarchiche:

- `Surface; Comment`.

One-Hot-Encoding Parziale

Trasformare le colonne `Winner` e `Loser` con lo One-Hot-Encoding avrebbe allargato troppo il dataset, dato il numero elevato di valori ed ho quindi deciso di ordinare i giocatori in base alla differenza tra vittorie e sconfitte considerando i migliori 25. Ho poi creato per ognuno di essi due feature binarie per indicare se il giocatore A (`Winner`) è quel giocatore e se il giocatore B (`Loser`) è quel giocatore. Ho poi aggiunta una colonna `OtherA` e `OtherB` binaria se il Vincitore o il Perdente non sono nessuno tra i 25 migliori giocatori.

Grading

Grazie anche in parte all'analisi delle occorrenze dei valori delle variabili categoriali fatti al passo *00 Dataset Download and Examination*, ho assegnato un valore crescente in base all'ordine alle seguenti variabili:

- `Round`;
- `Series` (in base all'importanza del tipo di Torneo);
- `Location` (in base al numero di match giocati in quel luogo).

Grading Parziale:

La feature `Tournament` presentava un numero di valori troppo elevato per utilizzare la tecnica del one-hot-encoding o un grading totale, ho quindi ordinato i 25 tornei più importanti in base al loro prestigio da 1 a 25 e destinando valore 0 a tutti gli altri. Le fonti utilizzate sono le seguenti:

- https://www.tennis365.com/tennis-top-10/top-10-atp-tour-500-series-events-where-does-queens-club-rank/
- https://www.tennis365.com/tennis-top-10/top-10-biggest-non-grand-slam-tournaments-on-the-tennis-tour/
- https://www.forbes.com/sites/monteburke/2012/05/30/what-is-the-most-prestigious-grand-slam-tennis-tournament/#2370e5097696

03 Building The Prediction Model

Il passo successivo è stato quello di adeguare il dataset per costruire il modello di predizione. Ogni entry del dataset era costruita in modo da esplicitare il vincitore ed il perdente di ogni match, ho rinominato `Winner` in `PlayerA` e `Loser` in `PlayerB` e le feature a loro collegate di conseguenza. Ho aggiunto una nuova feature Winner binaria che assume valore 0 se la partita è stata vinta da `PlayerA` e 1 se la partita è stata vinta da `PlayerB` e che sarebbe stata usata come previsore per l'output. Inizialmente, per come era costruito il dataset, ogni valore della nuova feature Winner è stato inizializzato a 0. Per poter allenare un modello era necessario invertire le informazioni dei giocatori di alcuni match per avere anche alcuni valori che indicassero la vittoria del `PlayerB` (`Winner` = 1).

Per ottenere un dataset bilanciato era necessario avere un numero equo di righe con valori 0 e 1 per `Winner`. Un'opzione considerata è stata quella di duplicare il dataset, invertire le righe della copia e riunire i due dataset mantenendo l'ordine temporale dei match. Per una questione computazionale, dato che la cardinalità dell'insieme delle entry del dataset era molto elevata già in principio, ho deciso di invertire solo le righe pari del dataset, in modo da avere comunque un modello bilanciato, ma non esageratamente grande, tenendo anche conto che non c'erano variazioni di attendibilità rispetto alle prove effettuate anche con la prima tecnica descritta.

Ho poi eliminato le feature che contenevano dati non utilizzabili per la previsione, in quanto indicavano le statistiche del match o non erano significative:

- `Date; PlayerA; PlayerB; 1A; 1B; 2A; 2B; 3A; 3B; 4A; 4B; 5A; 5B; setsA; setsB; Awarded; Completed; Disqualified; Retired; Sched; Walkover`.

04 Validating The Prediction Model

Ho poi provato a testare l'accuracy delle previsioni di due modelli. La divisione del dataset in train e validation è stata gestita in modo da mantenere l'ordine temporale dei dati. Il training dataset è stato costruito con i primi ⅔ dei dati del dataset ed il test dataset con i rimanenti.
Non è stato necessario applicare tecniche di bilanciamento del dataset quali Smote Oversampling in quanto le classi erano bilanciate per come le avevo costruite al passo precedente.
Prima di procedere alla vera e propria costruzione e validazione del modello ho ricercato da quale anno in poi (csvID) l'accuracy fosse migliore per un Decision Tree Classifier a cui è stato applicato un tuning sulla profondità. È risultato che l'accuracy massima fosse migliore per i dati dal 2002 in poi:

```
Best Max csvID: 1 - Accuracy: 0.6928549334982987
```

Per gestire il tuning degli iperparametri dei modelli ho deciso di non applicare una esaustiva grid search per motivi computazionali, ma di gestire il tuning di ognuno di essi singolarmente.
Il processo di scelta degli iperparametri è stato eseguito a tentativi ed ho notato che, per ognuno dei modelli, dopo il tuning di un paio di essi l'accuracy non migliorava più in modo significativo se aggiungevo ulteriori aggiustamenti degli iperparametri.

Decision Tree Classifier

Gli iperparametri considerati per il tuning dell'Albero Classificatore sono stati:

- max_depth; max_features.

Dall'analisi della Decomposizione della Varianza e Distorsione Quadratica delle previsioni del modello basata sulla profondità dell'albero è visibile come, all'aumentare di questa, l'albero faccia Overfitting, diminuendo la distorsione, ma aumentando la varianza.

```
Best Max Depth: 5 - Accuracy: 0.6928549334982987
Best Max Features: 40 - Accuracy: 0.6932261057841015
```

Random Forest Classifier

Gli iperparametri considerati per il tuning della Random Forest di Classificazione sono stati:

- n_estimators; max_depth.

Dall'analisi della decomposizione della Varianza e Distorsione Quadratica delle previsioni del modello basate solo sul numero di stimatori della foresta è visibile come, all'aumentare del numero di essi, la varianza diminuisca e la distorsione rimanga stabile. Una random forest con 300 stimatori presenta un errore più basso rispetto ad un albero fully grown, ma più alto di quello di un albero tunato, in quanto riesce ad abbassare la distorsione, ma non riesce a gestire in modo altrettanto efficiente la varianza aumentando l'errore totale.
Considerando anche la profondità della foresta si evidenzia che, all'aumentare di questa, la varianza cresce e la distorsione diminuisce. Una random forest con 300 stimatori e profondità 10 predice i risultati con variazioni minimali positive rispetto ad un albero tunato.

```
Best Estimators Number 300 - Accuracy: 0.6864831425920198
Best Depth 10 - Accuracy: 0.6941540364986081
```

05 Adding New Features

Ho ripreso il dataset precedente al passo *03 Building The Prediction Model* per aggiungere nuove feature ritenute importanti per aumentare la qualità della previsione.

Feature basate su quelle fornite dal Dataset

Sono state aggiunte le seguenti feature:

- `OpponentsPlayed`: Il numero di partite disputate tra i due contendenti prima del match;
- `OpponentsWRatio(A/B)`: Il rapporto tra partite vinte dal giocatore A o B e giocate tra i due contendenti prima del match;
- `FadigueTournGames(A/B)`: Il numero di game giocati da A o B nell'attuale torneo prima del match, come misura di stanchezza.
- `FadigueTournSets(A/B)`: Il numero di set giocati da A o B nell'attuale torneo prima del match, come misura di stanchezza.
- `WalkoverLast(A/B)`: feature binaria per indicare se il giocatore A o B aveva subito un infortunio l'ultimo match (`Walkover` = 1)
- `RetiredLast(A/B)`: feature binaria per indicare se il giocatore A o B si era ritirato l'ultimo match (`Retired` = 1)
- `Played(A/B)`: Il numero di partite giocate dal giocatore A o B prima di quel match
- `WonRatio(A/B)`: Il rapporto tra partite vinte dal giocatore A o B e giocate prima del match

- `PlayedCourt(A/B)`: Il numero di partite giocate dal giocatore A o B nello stesso tipo di campo del match prima della partita;
- `WonRatioCourt(A/B)`: Il rapporto tra partite vinte dal giocatore A o B e giocate prima della partita nello stesso tipo di campo del match;
- `5_setsMean(A/B)`: La media del numero di game vinti dal giocatore A o B nei 5 match precedenti alla partita;
- `5_gamesMean(A/B)`: La media del numero di set vinti dal giocatore A o B nei 5 match precedenti alla partita;

Data Integration
Ho deciso di integrare i dati aggiungendo una feature che indicasse la mano con cui gioca il giocatore A (`Winner`) `HandA`, e la mano con cui gioca il giocatore B (`Loser`) `HandB`.
Le informazioni sono state recuperate dal dataset presente al seguente link:
- https://www.kaggle.com/sijovm/atpdata

Essendoci informazioni mancanti per alcuni giocatori, le ho recuperate presso i seguenti link:
- https://www.ultimatetennisstatistics.com/
- https://www.tennisexplorer.com/

È stato necessario un lavoro di data pre-processing per rendere i nomi dei giocatori nello stesso formato.
Oltre alla mano con cui giocano i contendenti sono state aggiunte le seguenti feature:
- `PlayedVsSameHanded(A/B)`: Il numero di partite disputate dal giocatore A o B contro giocatori che usano la stessa mano dell'avversario prima del match;
- `WonRatioVsSameHanded(A/B)`: Il rapporto tra partite vinte dal giocatore A o B e giocate contro giocatori che usano la stessa mano dell'avversario prima del match;

Feature di Confronto
Sono state infine aggiunte delle feature per esprimere un confronto di maggioranza tra le caratteristiche del giocatore A e quelle del giocatore B.
Ho ritenuto opportuno aggiungere le nuove feature riga per riga, sfruttando il fatto che le partite sono ordinate in base alla data. Utilizzando il metodo *`itertuples()`* applicato al dataframe, che itera attraverso le righe come `namedtuples`, sono così riuscito ad ottenere tempi di computazione migliori rispetto a quelli risultanti utilizzando il metodo *`iterrows()`*.

---

06 Rebuilding The Prediction Model

Eseguo nuovamente le procedure in *03 Building The Prediction Model* tenendo conto delle nuove variabili aggiunte.

---

07 Revalidating The Prediction Model

Ho ritestato l'accuracy delle previsioni utilizzando quattro modelli differenti. Le considerazioni sulla divisione del dataset in train e validation e sul metodo di tuning degli iperparametri è stato lo stesso del punto *04 Validating The Prediction Model* con l'aggiunta di una breve Feature Relevance Analysis prima della vera e propria Validation. È inoltre risultata più precisa l'accuracy considerati i dati di tutti gli anni:
`Best Max csvID: 0 - Accuracy: 0.6944347217360868`

Feature Relevance Analysis
Ho verificato l'accuratezza delle predizioni di un albero con applicato il tuning alla profondità per il dataset con tutte le feature e per il dataset eliminando le feature con le caratteristiche specifiche del giocatore A e il giocatore B, lasciando solo quelle di confronto tra le caratteristiche dei due giocatori (*caratteristicaA>caratteristicaB*). Ho ritenuto opportuno applicare questo tipo di analisi in quanto, a mio avviso, i modelli che sfruttano Alberi di Decisione raggiungono risultati di maggior precisione considerando solo confronti binari.
Ho applicato il dataset con l'eliminazione delle feature in quanto risulta avere un'accuracy migliore.
- Dataset con tutte le feature: `Best Depth: 5 - Accuracy: 0.6922762804806907`
- Dataset considerando solo le feature di confronto: `Best Depth: 3 - Accuracy: 0.6944347217360868`

Decision Tree Classifier
Gli iperparametri considerati per il tuning dell'Albero Classificatore sono stati:
- max_depth; max_features.

Dall'analisi della decomposizione della Varianza e Distorsione Quadratica delle previsioni del modello si evince come l'albero abbia un comportamento pari a quello osservato al punto *04 Validating The Prediction Model.*
L'aggiunta delle nuove feature non ha migliorato di molto l'accuratezza rispetto ai modelli costruiti al passo *04 Validating The Prediction Model.* Questo significa che le feature aggiunte sono poco rilevanti per la predizione.
`Best Max Depth: 3 - Accuracy: 0.6944347217360868`
`Best Max Features: 44 - Accuracy: 0.6944930579862326`

AdaBoost Classifier

Ho deciso di non procedere al Bagging del modello in quanto dall'analisi della Decomposizione della Varianza e Distorsione Quadratica delle previsioni dell'albero, la distorsione è la maggiore fonte di errore. Per provare a migliorarla ho utilizzato la tecnica di AdaBoost.
L'iperparametro considerato per il tuning dell'AdaBoost Classifier è stato n_estimators e ho deciso di dare in pasto al modello l'albero tunato al punto precedente, in quanto AdaBoost preferisce small tree con poca varianza per migliorare la distorsione.
L'analisi della Decomposizione della Varianza e Distorsione Quadratica delle previsioni del modello mostra come all'aumentare del numero di stimatori, l'errore si abbassi di poco, ma la varianza cresca aumentando l'errore totale.
L'accuratezza del modello tunato risulta essere praticamente al pari rispetto a quella dell'albero tunato.

```
Best Number of Estimators: 3 - Accuracy: 0.6942013767355034
```

Random Forest Classifier

Gli iperparametri considerati per il tuning della Random Forest di Classificazione sono stati:

- n_estimators; max_depth.

L'analisi della decomposizione della Varianza e Distorsione Quadratica delle previsioni del modello evidenzia che la Random Forest ha un comportamento pari a quello della Random Forest al punto *04 Validating The Prediction Model.*
Una random forest con 175 stimatori e profondità 15 predice i risultati con variazioni estremamente minimali negative rispetto ad un albero tunato ed è possibile considerare le sue previsioni alla pari dell'albero.
Ribadisco che l'aggiunta delle nuove feature non ha migliorato di molto l'accuratezza rispetto ai modelli costruiti al passo *04 Validating The Prediction Model* e che le feature aggiunte sono poco rilevanti ai fini della predizione.

```
Best Estimators Number 175 - Accuracy: 0.6651499241628748
Best Depth 15 - Accuracy: 0.6942597129856493
```

.

Recursive Feature Elimination

Applicando la tecnica del Recursive Feature Elimination alla Random Forest con tuning dei parametri descritta al paragrafo precedente ho ottenuto la metà delle feature più significative per la predizione.
Le predizioni di una Random Forest tunata nuovamente su n_estimators e max_depth ha portato ad una leggera riduzione dell'accuratezza del modello, ma non troppo significativa.
Ho deciso di applicare questo modello per prevedere i risultati del torneo in quanto utilizza un numero di parametri inferiore, garantendo una complessità computazionale più bassa ed un'accuratezza non troppo ridotta rispetto ai modelli che utilizzavano tutte le feature.

```
Best Estimators Number 150 - Accuracy: 0.661766421654416
Best Depth 9 - Accuracy: 0.6940263679850659
```

Inspection of Prediction Results

I risultati dell'ispezione fatta sulle predizioni del modello ottenuto tramite la Recursive Feature Elimination evidenziano come l'accuratezza aumenti prevedendo partite dove entrambi i giocatori sono tra i migliori 25 e riesce ad essere ancora più efficace quando la previsione dei match in cui almeno uno dei due giocatori è uno tra i migliori 25.

- `Accuracy on matches played by the most important players: 0.7333333333333333`
- `Accuracy on matches played by the least important players: 0.6599084368868542`
- `Accuracy on matches played by at least one important player: 0.7684317718940937`

---

08 Prediction Simulation

A questo punto ho selezionato i giocatori ammessi al torneo Australian Open 2020 ed ho simulato i singoli match di ogni round considerando coppie casuali di tennisti. Dopo il risultato simulato di ogni match ho eliminato dalla lista di giocatori il perdente, prevedendo così la vittoria di --- ed il secondo posto di ---.

Le feature di ogni match sono state inserite utilizzando lo storico delle partite precedenti.
Il modello di previsione andrebbe utilizzato durante lo svolgimento del torneo inserendo di volta in volta i dati relativi alla stanchezza in base al numero di game e set giocati e la media dei game e set vinti delle ultime cinque partite. Non ho utilizzato le feature relative al confronto della stanchezza dei due giocatori non avendo a disposizione dati reali.
Ho inserito comunque i dati relativi al confronto della media dei set e game vinti considerando gli ultimi cinque match precedenti l'inizio del torneo.
Ho inserito la comparazione tra gli odd medi e quella tra gli odd massimi dei bookmaker applicando la media dei precedenti per ogni giocatore, non potendo accedere ad i dati reali.
Il modello prevede di esplicitare il confronto tra i rank e quello tra i punti che ho inserito utilizzando il valore dell'ultimo Rank e quello dell'ultimo punteggio dei due contendenti.

Ritengo che il modello costruito raggiunga una buona accuratezza pur nel limite dei dati utilizzati e nella semplicità della sua costruzione. Rimangono comunque numerose variabili imponderabili che possono vanificare qualsiasi tipo di previsione per quanto accurata.