**Spolaor Riccardo 864877**

***Web Intelligence Predict 2020***

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

Introduzione

Lo scopo del progetto è quello di predirre l’outcome del torneo di tennis Australian Open del 2020 costruendo un modello di predizione a partire dai dati delle singole partite forniti dal sito <http://www.tennis-data.co.uk/> e simulando le partite match per match.

Ho deciso di impostare il lavoro in parti distinte dividendo i singoli processi di analisi in vari Notebook diversi perché fosse più chiara la loro visualizzazione. Sono stati creati dei file Python separati all’interno del folder .*/python\_files* per gestire funzioni che vengono richiamate più volte o utilizzate in Notebook differenti oppure 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 che sono contenuto a loro volta all’interno di file .zip. Per ogni file i tornei sono ordinati per data. Ho deciso di ignorare il file riguardante l’anno 2000 in quanto non contiene, al contrario degli altri, informazioni riguardo agli odd dei bookmaker per le partite e di utilizzare tutti gli altri.

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 in un unico oggetto DataFrame, mantenendo l’ordine temporale dei match.

Ho fatto in seguito un lavoro preliminare di Data Pre-Processing. Certi passaggi di questa fase sono stati inseriti quando ero già giunto a passi successivi dell’analisi, in quanto per alcune feature mi sono accorto solo in un momento successivo della presenza di dati sporchi o non corretti.

In primo luogo ho analizzato quali fossero i dati che contenevano valori reali, ma erano stati assegnati erroneamente ad object ed ho corretto i valori errati trasformandoli in reali (o NaN).

Ho poi corretto i typo dei nomi dei giocatori nelle colonne ***Winner*** e ***Loser*** eliminando spazi vuoti prima e dopo dei nomi, trasformando tutte le stringhe in maiuscolo e procedendo ad altre correzioni manuali.

Ho infine analizzato le feature categoriali per capire quante occorrenze di ogni label fossero presenti e per capire come potessi gestirle nel momento in cui le avrei trasformate in numeriche.

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 da vincitore e perdente.

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 riempito i valori anomali con NaN in modo da poterli riempire con valori più accettabili nei passi successivi.

Dato 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 da vincitore e perdente, è stata aggiunta una feature contenente valori binari per indicare se il valore è stato aggiunto da me o meno.

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 valori 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 valori 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

**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 categoriali ed ho quindi deciso di ordinare i giocatori in base alla differenza tra vittorie e sconfitte e prendere i migliori 25. Ho poi creato per ognuno di essi due feature binaria 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 deciso di dare 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:**

Per quanto riguarda la feature Tournament, il numero di tornei presenti era troppo elevato ed è stato quindi fatto un Grading Parziale ordinando i 25 tornei più importanti in base al loro prestigio da 1 a 25 e destinando valore 0 a tutti gli altri tornei. Le fonti da cui ho preso informazioni sull’importanza dei tornei 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 modificare il dataset in modo da renderlo adatto alla predizione. Dato che ogni entry del dataset era costruita in modo da esplicitare il vincitore ed il perdente di ogni match era necessario chiamare Winner PlayerA e Loser PlayerB ed le feature collegate al vincitore ed al perdente di conseguenza ed aggiungere 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. Questo nuovo label sarebbe stato usato come previsore per l’output della partita.

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 il PlayerA e il PlayerB e le feature a loro collegate di alcuni match per avere anche alcuni valori che indicassero la vittoria del giocatore B (Winner = 1).

Per ottenere un dataset bilanciato era necessario avere un numero equo di righe con valori 0 e 1 per la nuova feature aggiunta 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 delle entry del dataset era molto elevata già in principio, ho deciso invece di invertire solo le righe pari del dataset, in modo da avere comunque un modello bilanciato, ma mantenendo la cardinalità originale.

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

* 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 ho 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 2001 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 iperparametri.

**Decision Tree Classifier**

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

* max\_depth;
* max\_features.

Best Max Depth: 5 - Accuracy: 0.6928549334982987

Best Max Features: 40 - Accuracy: 0.6932261057841015

Dall’analisi della decomposizione Varianza e Distorsione Quadratica delle previsioni del modello basata sulla profondità dell’albero è visibile come all’aumentare della profondità l’albero fa Overfitting, diminuendo la distorsione, ma aumentando la varianza.

**Random Forest Classifier**

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

* n\_estimators;
* max\_depth.

Best Estimators Number 300 - Accuracy: 0.6864831425920198

Best Depth 10 - Accuracy: 0.6941540364986081

Dall’analisi della decomposizione Varianza e Distorsione Quadratica delle previsioni del modello basata sul numero di stimatori della foresta è visibile come all’aumentare del numero di stimatori la varianza diminuisca e la distorsione rimanga sempre stabile. La random forest predice meglio di un albero fully grown, ma peggio 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.