Master BI&BDA

BETS INSIGHTS

Technical Documentation

A.A. 2020/2021

Group 4
Davide Airaghi – *Engineer*
Luca Gaddoni – *Business Analyst*
Daniele Raimondi – *Data Scientist*
Federico Reggiani – *Mathematician*
Chiara Teruzzi – *Engineer*

Contents

Data Collecting

La raccolta dati dai siti Betfair.it e Diretta.it è avvenuta tramite scraper Python,utilizzando il framework Selenium, eseguito su notebook Google Colaboratory.

Questa decisione è stata presa per motivi sia di organizzazione e gestione sia architetturali. Infatti non potendo far girare lo scraper quando volevamo ma dovendo sottostare all'organizzazione delle partite era fondamentale per noi poter utilizzare più computer per raccogliere i dati e quindi era necessario poter contare su prestazioni e configurazioni identiche tra le diverse macchine, inoltre la condivisione è stata notevolmente facilitata. Considerato poi che la maggior parte della nostra architettura è stata sviluppata interamente in cloud, abbiamo deciso quindi di utilizzare questa tecnologia.

Abbiamo utilizzato questo approccio per raccogliere i dati in tempo reale principalmente per due motivi. Innanzitutto era fondamentale avere il timestamp ogni volta che estraevamo un dato per poterlo relazionare poi con altre informazioni. In aggiunta, non avremmo avuto accesso a questi dati finite le partite o, come nel caso delle statistiche, avremmo potuto recuperare solamente il dato aggregato che non ci sarebbe servito a nulla per poter analizzare le fluttuazioni delle quote interne alla partita.

Infine abbiamo deciso di essere molto prudenti nella gestione dei possibili errori negli scraper poiché non potevamo permetterci di perdere tutti i dati della partita per un'informazione male estratta, soprattutto perché avremmo dovuto aspettare giorni per una nuova partita.

Scraper Betfair

Lo scraper Betfair si occupa di raccogliere le quote e informazioni più volte al minuto dal sito Betfair Exchange.

Alla fine dell'esecuzione viene generato il dataset e caricato automaticamente su Azure dopo aver generato un codice univoco a partire dai dati della partita (stesso di Diretta).

La struttura del dato è conosciuta a priori quindi abbiamo optato per un output impostato in colonne.

Per poter seguire i mercati più interessanti abbiamo dovuto configurare 6 webdriver diversi che si occupavano di estrarre le quote da ogni singolo mercato.

Questa soluzione nelle prime iterazioni rallenta l'esecuzione dello script, però abbiamo aggiunto delle condizioni che chiudono webdriver mano a mano che viene segnato un gol: questo ci permette di avere uno scraper più veloce nel momenti più delicati della partita. Abbiamo deciso di non implementare soluzioni più complicate e dispendiose per poter accelerare l'esecuzione dello script poichè allo stato attuale ci bastano poche quote al minuto e abbiamo preferito invece testare lo scraper e ottenere dati reali per poter costruire l'intera architettura.

L'impostazione dello scraper è così strutturata:

Installazione dei pacchetti necessari (tra cui Selenium, chromium-chromedriver, Azure…), import delle librerie (time, pandas, numpy, matplotlib…), definizione di funzioni (come create_id_match e

upload_file_to_Azure che ci permettono rispettivamente di generare il codice univoco per la partita e caricare il dataset su Azure in automatico) ed infine configurazione dei sei web-driver necessari.

Il funzionamento è semplice: dopo aver inizializzato tutte le variabili, inizia il ciclo while che prosegue fino a che la partita non finisce o che non raggiunge un numero massimo di righe. Questa condizione, che è stata implementata con due controlli separati, è stata aggiunta per evitare che il ciclo iteri all'infinito nella possibilità che la scritta "Finito" (scritta che compare sul sito una volta che il match è terminato) venga mancata. Capita a volte che quando la partita termina, la pagina venga chiusa quasi all'istante e se lo scraper non si aggiorna abbastanza velocemente rischia di perdere questa informazione; così facendo preveniamo questo caso. La variabile "rows" utilizzata a tal fine viene valorizzata in due maniere differenti: nel primo caso vengono aggiunte 50 o 100 iterazioni (in base al numero di gol e quindi al numero di webdriver ancora aperti) alle iterazioni già avvenute quando si entra nel recupero del secondo tempo. Nel caso che per tutto il recupero non vengano mai visualizzati i minuti (ad esempio che la partita termini senza recupero, oppure per problemi di estrazione del dato), allora pre-valorizziamo "rows" con un valore massimo prefissato che fa si che il ciclo finisca comunque.

All'inizio del ciclo While, dopo aver controllato quali mercati sono ancora aperti, viene selezionata, tramite una query "css-selector", la porzione di pagina contenente le quote che ci interessano e successivamente vengono estratti tali dati. In caso di fallimento di recupero del dato la variabile viene settata di default a "None".

Infine trasformiamo alcuni dati, generiamo il dataset e carichiamo tutto su Azure.

Scraper Diretta

Lo scraper Diretta si occupa di raccogliere le statistiche, commenti partita e informazioni più volte al minuto dal sito di Diretta.

Alla fine dell'esecuzione verranno generati due dataset (uno per le statistiche della partita e uno per i commenti) e caricati automaticamente su Azure dopo aver generato un codice univoco a partire dai dati della partita (stesso di Betfair).

La struttura del dato non è conosciuta a priori poiché vengono, a volte, aggiunte statistiche diverse durante lo svolgimento della partita. Abbiamo quindi optato per un output impostato in righe.

Per poter raccogliere le statistiche della partita utilizziamo tre webdriver diversi per poter estrarre contemporaneamente i dati dell'intera partita, del primo tempo e del secondo tempo (scelta per avere una migliore precisione per il dato "possesso palla" da poter utilizzare poi nella fase di analisi).

Finita la partita recuperiamo tutti i commenti (insieme alle azioni salienti marcate da un'icona e riassunte con una parola chiave) in una volta sola perché è risultato essere di più facile implementazione e, come prima, abbiamo deciso di avere accesso ad un dato "pronto all'uso" il prima possibile. Inoltre al momento analizziamo le partite una volta finite, quindi non ci interessa recuperare i dati in tempo reale ed inoltre sono riportati i minuti dei commenti che ci permettono di mettere in relazione i commenti con statistiche e quote.

L'impostazione dello scraper è strutturata generalmente uguale a quella dello scraper precedente. In aggiunta viene definita una nuova funzione che ci permette di identificare le icone e selezionare quindi il commento saliente e la tipologia a cui fa riferimento (ad esempio cartellino rosso, rigore, gol…).

Una volta configurato tutto il necessario si passa alla parte che si occupa dell'estrazione delle statistiche. In questo caso, dopo aver inizializzato le variabili, troviamo un doppio ciclo while. Il primo ciclo while termina una volta che la partita finisce e in questo caso non abbiamo problemi di gestione della chiusura della pagina poiché la pagina web rimane presente ad oltranza (per maggiore sicurezza abbiamo aggiunto un controllo sul numero massimo di righe). Il secondo ciclo While invece è dinamico: il numero di statistiche può variare da iterazione a iterazione. All'interno del primo While selezioniamo, come prima, la porzione di pagina interessata e nel secondo ci occupiamo di estrarre il dato. Anche in questo caso, se il dato viene male estratto la variabile viene impostata di default su "None". Inoltre recuperando circa 4 set di statistiche complete al minuto, abbiamo inserito un ulteriore try/except anche prima del secondo ciclo, visto che il dato è piu che ridondato all'interno dello stesso minuto (arco di tempo col quale verranno visualizzati i dati in fase di analisi) e sarebbe molto più problematico il fatto di saltare l'intera partita per gli stessi motivi esposti sopra.

Finita la partita e salvato il dataset, si lascia aggiornare la pagina e si passa infine al blocco dei commenti.

In questo caso l'unico ciclo While presente termina con il commento "fischio d'inizio" (sempre presente) poiché estraiamo i commenti in senso contrario. Questa scelta è stata fatta per pura comodità di implementazione. Inoltre conosciamo a priori il numero dei commenti (visto che la partita è terminata).

Terminata questa sezione, viene creato il dataset commenti e caricato, insieme alle statistiche, su Azure.

Gestione minuti partita

Per tutti e tre i dataset generati (quote, statistiche e commenti) un dato particolarmente importante riguarda il minuto della partita poiché ci permette di mettere in relazione tutti i dati tra loro. Premettiamo che a noi servono soltanto i minuti e non anche i secondi di partita, quindi nel dataset il primo minuto utile sarà il 1' fino ad arrivare oltre il 90' (dipende dal recupero del secondo tempo).

E' sorto dunque il problema di come gestire il recupero del primo tempo poiché i minuti finali dello stesso e quelli iniziali del secondo tempo coincidono, ad esempio un primo tempo con due minuti di recupero terminerà circa al 47' ma il secondo tempo inizierà sempre dal 45' (il problema non si pone nel caso del finale del secondo tempo poiché dopo il novantesimo minuto non c'è rischio che il dato si duplichi in alcun modo).

Visto che generalmente il recupero del primo tempo è molto corto e generalmente poco importante abbiamo quindi deciso di comprimere tutto il recupero del primo tempo nel minuto 45 e iniziare il secondo tempo con il minuto 46 (questo è in linea con il fatto di iniziare la partita con il minuto 1 al posto del minuto 0). Per fare questo abbiamo implementato diversi controlli che gestiscono i minuti del primo tempo (creando una variabile ad hoc che ci segnalasse se fossimo nel primo o secondo tempo) in un modo diverso da quelli del secondo.

Funzione create_id_match

Tra le funzioni da noi create, la funzione "create_id_match" ricopre un ruolo centrale per la nostra architettura. Come già accennato la funzione si occupa di generare il codice univoco per la partita che verrà utilizzato per accedere, oltre al campo minuto, ai tre dataset generati. Nello specifico la

funzione si occupa di interrogare il Database su Azure ottenendo l'id dei due team della partita e generando un nuovo record con chiave univoca nella tabella DIM_Match.

La chiave é così composta:

- id del team che gioca in casa (preceduto a sinistra da sei '0')
- id del team ospite (preceduto a sinistra da sei '0')
- data match in formato yyyymmdd

Per esempio

id_team_1 = 5

id_team_2 = 9

data match = 07/09/2020

→ **0000050000092020907**

La funzione viene richiamata in tutti gli scraper, il primo che parte andrà a censire il nuovo id_match e gli altri lo utilizzeranno di conseguenza.

Soluzione Cloud con Microsoft Azure

Soluzione

La maggior parte della nostra architettura è stata sviluppata interamente in cloud utilizzando il servizio Microsoft Azure.

In particolare i servizi utilizzati in microsoft Azure sono:

- Blob storage container (per organizzazione File)
- Data Factory (per ETL)
- SQL Server (su cui è stato creato il DB SQL)
- Database SQL, 2 GB (per Storage)
- Logic App (per invio mail in caso di errore)

Abbiamo scelto la soluzione in cloud soprattutto per la sua caratteristica di alta disponibilità che ci ha consentito di eseguire le procedure di caricamento, di analisi e, in generale, di fruizione del dato, in qualsiasi momento senza doverci appoggiare a macchine fisiche.

Inoltre il servizio Azure ci ha consentito di portare il processo ad un alto livello di automatizzazione tanto che l'utente deve semplicemente caricare il file sul container per poter fruire qualche istante dopo del dato già trasformato e in qualità sul DWH.

Data Processing

Il processo di scraping genera tre differenti file con diversi campi e diverse granularità ma la modalità di caricamento e processamento del dato è condivisa da tutti e tre i dataset.

E' possibile scorporare il processo in due macrofasi:

1. Caricamento file .csv sul blob storage dedicato su Azure
2. ETL pipeline

Caricamento file .csv su Blob Storage dedicato

Nella fase di scraping abbiamo predisposto due diversi notebook Colab per estrarre da due differenti siti dedicati alle partite:

- Betfair.it: per exchange delle quote delle partite
- Diretta.it: per statistiche live e commenti delle partite

Una volta finito lo scraping e una volta generato il relativo csv, la funzione **upload_file_to_Azure** che riceve come parametri il path del file e il container dedicato avvia il caricamento del file csv nell'apposito container.

Sono presenti tre diversi container per accogliere i diversi csv

- bets-datasets-betfair
- bets-datasets-diretta-comment
- bets-datasets-diretta-stats

Più un container di backup per storare i file che sono stati correttamente caricati

- dump-loaded-files

ETL pipeline

Per effettuare il processamento del dato ricevuto sono state create tre differenti pipeline su una Data Factory dedicata su Microsoft Azure.

Le pipeline sono tra loro architetturalmente molto simili e presentano le stesse attività:

Figura 1 ETL pipeline BetFair

- **TRIGGER**
  Ogni pipeline ha associato un apposito trigger che scatena l'esecuzione della stessa. Il trigger resta in ascolto sul container relativo alla pipeline e farà partire l'esecuzione della

pipeline qualora si verificasse l'evento: "nuovo file blob" ovvero se un nuovo file viene depositato sul container.
Ignora i file vuoti.

- **TRUNCATING**
  All'inizio dell'esecuzione vengono troncate la tabella di staging (_stg) e quella dove avvengono le trasformazioni (_etl) questo per eliminare qualsiasi residuo di precedenti esecuzioni.

- **EXTRACTING – STAGING AREA**
  Una volta letto il .csv questo viene caricato all'interno della tabella di staging precedentemente troncata. Non è ovviamente presente nessun tipo di constraints e tutti i campi hanno il datatype nvarchar in modo da consentire nella maggior parte dei casi il caricamento del dato.
  Questo ci consente di poter effettuare delle query su una tabella nel caso ci fossero dei problemi successivi durante il processo di ETL e risulta sicuramente più agevole che cercare la causa di un caricamento fallito all'interno dello stesso csv.

- **TRANSFORMING**
  In questa fase avvengono le trasformazioni e la pulizia vera e propria del dato appena caricato in staging area. Tutte queste operazioni vengono di fatto eseguite da una stored procedure che applica le logiche di:

  - Aggregazione: la granularità del dato é attestabile sulla decina di secondi ma per le nostre analisi la granularità necessaria è il minuto pertanto il dato viene aggregato secondo questo attributo.
  - Gestione valori null
  - Filtering: filtriamo i momenti di prematch e intervallo
  - Nel caso del dataset delle statistiche pivotiamo il dato in modo da avere tutte le statistiche per minuto su un'unica tupla

  Il dato viene caricato nella tabella con suffisso etl che presenta lo stesso tracciato della tabella di warehouse ma senza vincoli di integrità referenziali – not nullable.

- **LOADING – WAREHOUSE AREA**
  La tabella con suffisso etl viene ribaltata all'interno della tabella dedicata nella warehouse area.

- **ERROR HANDLING**
  L'error handling avviene tramite:

  - La stored procedure sp_bets_log che logga sia errori ma anche altre informazioni di log.
  - L'app per la logica si occupa di inviare una mail a seguito di una richiesta "http" che scatena la pipeline stessa ai componenti del gruppo riportando l'errore avvenuto con questo formato:

Data Factory: ETL-Bets

Pipeline Name: ETL-bets-Diretta_Comments

Error message:
ErrorCode=UserErrorInvalidColumnMappingColumnNotFound,'
Type=Microsoft.DataTransfer.Common.Shared.HybridDeliveryE
xception,Message=Column 'id' specified in column mapping
cannot be found in source
data.,Source=Microsoft.DataTransfer.ClientLibrary,'

- **ALTRE OPERAZIONI**:
  Una volta terminato il caricamento della warehouse area avvengono le seguenti operazioni:

  - Backup: il file appena caricato viene migrato dal suo container al container di backup dump-loaded-files. Questo per avere sempre la sorgente del dato nel suo stato più grezzo.

  - Storage cleanup: tutti i file vengono cancellati all'interno del container

  - Troncamento tabella di staging ed etl

Oggetti del DWH

Stored procedure:

- sp_bets_log: procedura utilizzata per le operazioni di log
- sp_etl_comm: procedura utilizzata per la trasformazione del dato in input per quanto riguarda il dataset dei commenti del match
- sp_etl_shares: procedura utilizzata per la trasformazione del dato in input per quanto riguarda il dataset delle quote del match
- sp_etl_stats: procedura utilizzata per la trasformazione del dato in input per quanto riguarda il dataset delle statistiche del match
- sp_truncate_table: procedura che opera il troncamento di una tabella passata come parametro
- sp_populate_DIM_Time: procedura utilizzata per il popolamento della tabella dimensione tempo.

Tabelle:

- bets_log: tabella di log
- dec_bets_log_type: tabella per la decodifica del codice del log
- DIM_Match: tabella dimensione che raccoglie tutti i match al momento raccolti
- DIM_Time: tabella dimensione per il tempo delle partite
- DIM_Teams: tabella che raccoglie tutti i team presenti nelle partite raccolte
- stg_STATS: tabella di staging per le statistiche
- stg_COMM: tabella di staging per i commenti
- stg_SHARES: tabella di staging per le quote
- etl_STATS: tabella dove viene scritto il dato delle statistiche trasformato dalla staging area
- etl_SHARES: tabella dove viene scritto il dato delle quote trasformato dalla staging area
- etl_COMM: tabella dove viene scritto il dato dei commenti trasformato dalla staging area
- wrh_STATS: tabella di warehouse per le statistiche

- wrh_COMM: tabella di warehouse per i commenti
- wrh_SHARES: tabella di warehouse per le quote

Viste:

- VW_BETS_COMMENTS: vista contenente tutti i commenti caricati
- VW_BETS_STATS: vista contenente tutte le statistiche caricate
- VW_BETS_SHARES: vista contenente tutte le quote caricate
- VW_BETS_RECONC: é la vista di riconciliazione che mette a fattor comune statiche,quote e commenti. E' la vista che viene utilizzata nelle nostre analisi e nella dashboard su PowerBI. Tutto il dato raccolto è fruibile da questa vista

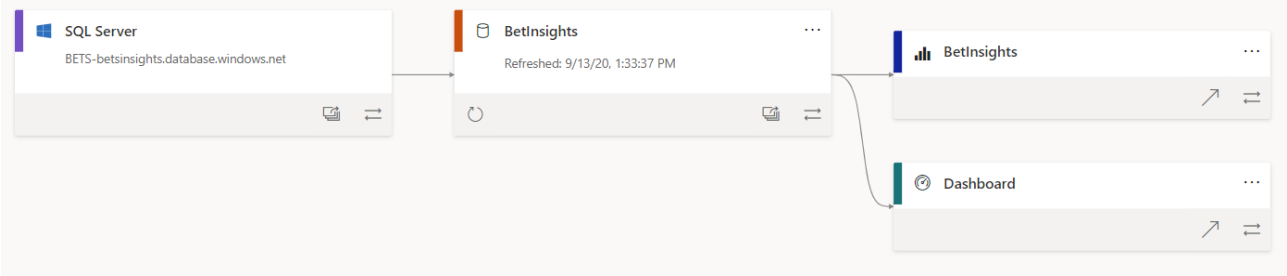
Data Visualization

La presentazione finale dei risultati è avvenuta tramite tool di Data Visualization Microsoft Power BI.

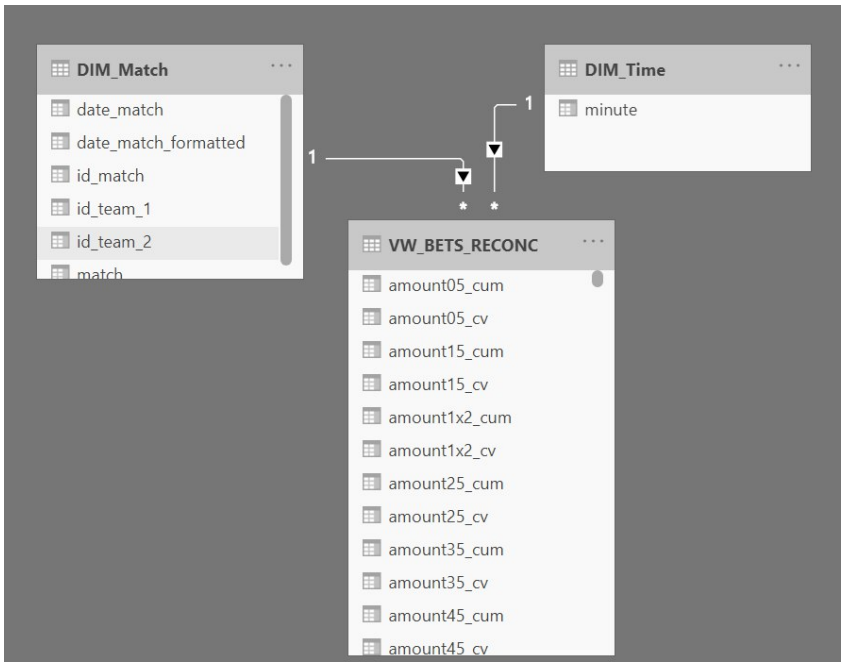
Lavorando su Power BI desktop, in locale, e generando poi un'App online (scaricabile anche su Mobile) su Power BI Service.

Connessione

Power BI è connesso in modalità Import al database "BETS" sul Server su Azure.



Modello



Le varie tabelle dei fatti su SQL sono state riconciliate in'unica vista, contente quindi dati relativi alle statistiche della partita e commenti (derivanti da Diretta.it) e l'andamento delle quote (del sito di Betfair Exchange).

Le tabelle dei fatti sono 3:

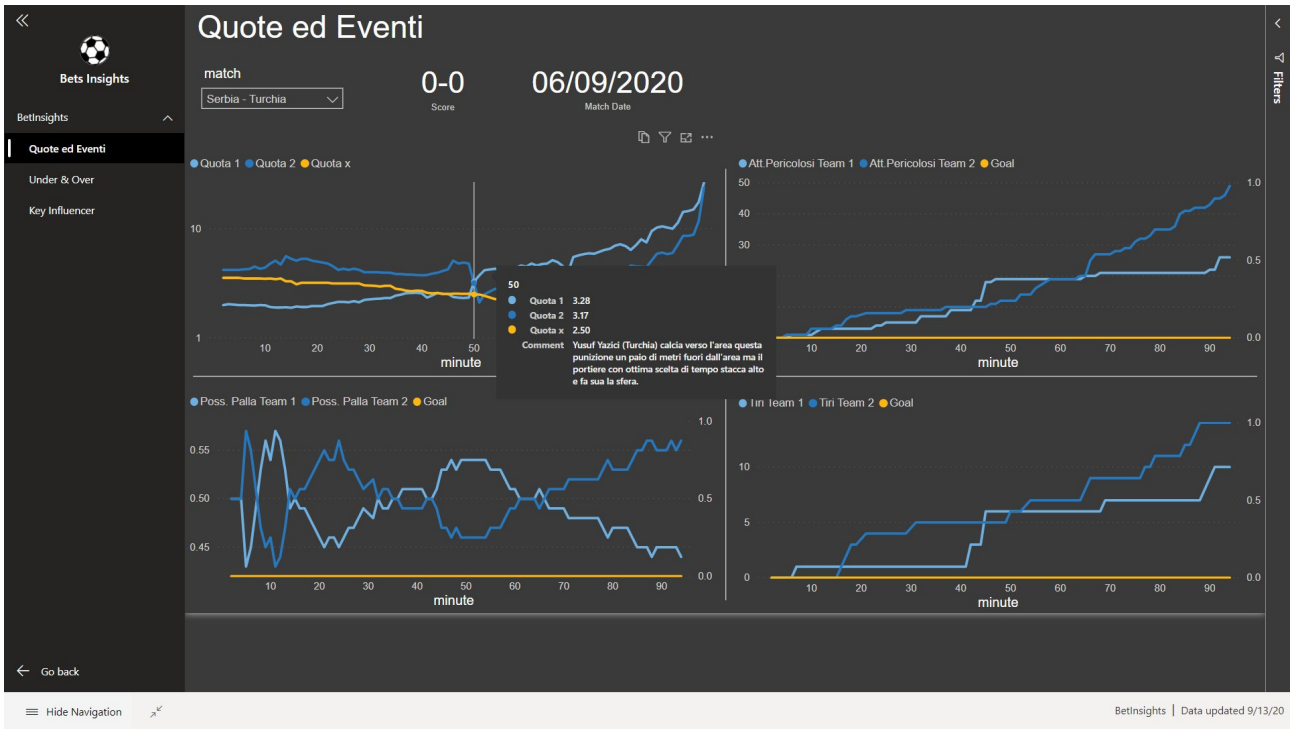
- DIM_Match ➔ conetenente data e team relativi ad uno specifico match
- DIM_Teams ➔ contenente tutte le squadre censite
- DIM_Time ➔ conetente tutti i minuti delle partite

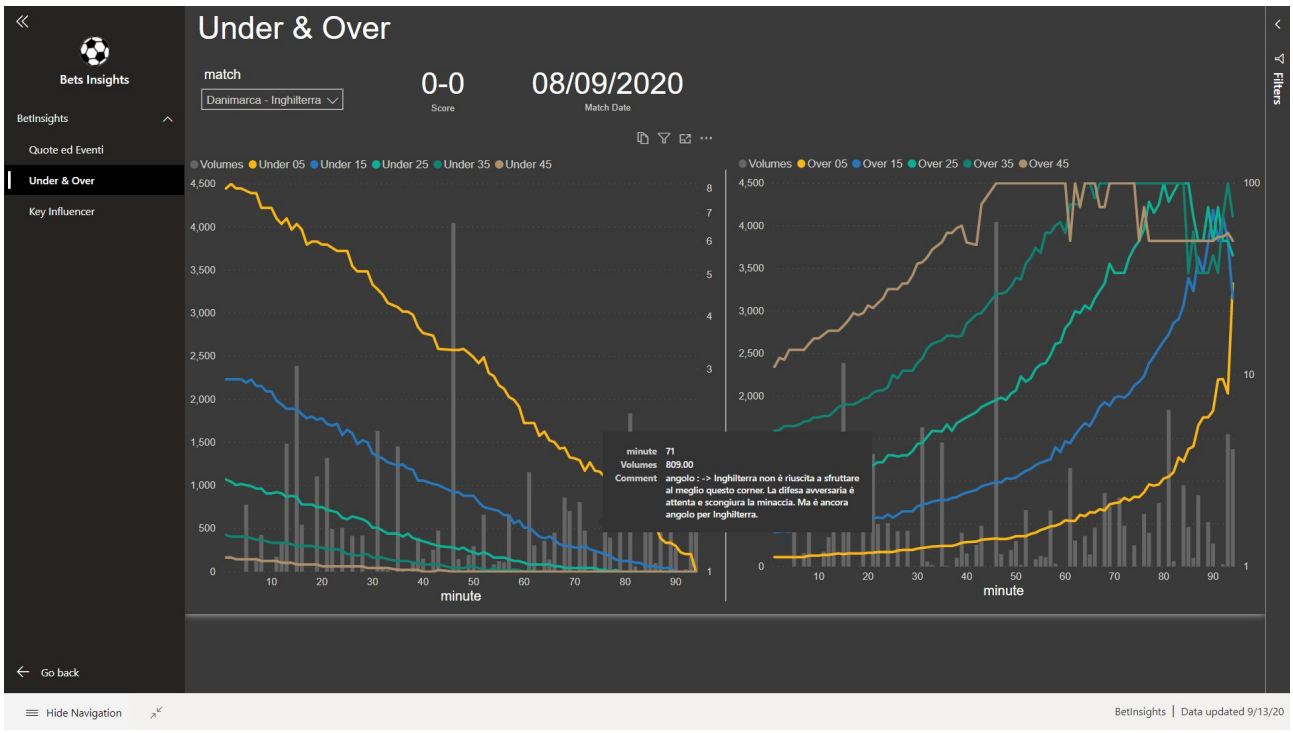
Power BI App

L'app finale di Power BI Services è costituita da un filter pane (per filtrare match o data match) e da 4 sezioni pricipali (a seconda dell'analisi):

1. **Quote ed Eventi** per analizzare l'andamento delle 3 quote (di Betfair) in base ad Attacchi pericolosi e Goal (rilevati da Diretta), possesso palla e tiri (informazioni scaricate da Statistiche di Diretta) per valuatre sempre l'influenza delle azioni della partita sull'andamento delle quote
2. **Under & Over** per analizzare gli Over e gli Under di Betfair
3. **Key Influencer** per vedere cosa influenza di più la fluttuazione delle quote di Betfair

Tutti i grafici sono analizzati per il minuto della tabella DIM_Times.


Quote ed Eventi
Bets Insights
match
Serbia - Turchia
0-0
Score
06/09/2020
Match Date
BetInsights
Quote ed Eventi
Under & Over
Key Influencer
Quota 1 Quota 2 Quota x
Att.Pericolosi Team 1 Att.Pericolosi Team 2 Goal
Poss. Palla Team 1 Poss. Palla Team 2 Goal
Tiri Team 1 Tiri Team 2 Goal
minute
50
Quota 1 3.28
Quota 2 3.17
Quota x 2.50
Comment Yusuf Yazici (Turchia) calcia verso l'area questa punizione un palo di metri fuori dall'area ma il portiere con ottima scelta di tempo stacca alto e fa sua la sfera.
Filters
Go back
Hide Navigation
BetInsights | Data updated 9/13/20


Under & Over
Bets Insights
match
Danimarca - Inghilterra
0-0
Score
08/09/2020
Match Date
BetInsights
Quote ed Eventi
Under & Over
Key Influencer
Volumes Under 05 Under 15 Under 25 Under 35 Under 45
Volumes Over 05 Over 15 Over 25 Over 35 Over 45
minute
minute 71
Volumes 809.00
Comment angolo : -> Inghilterra non è riuscita a sfruttare al meglio questo corner. La difesa avversaria è attenta e scongiura la minaccia. Ma è ancora angolo per Inghilterra.
Filters
Go back
Hide Navigation
BetInsights | Data updated 9/13/20

Le analisi sono svolte per singolo Match di cui si riportano data e punteggio finale.

Insights

L'obiettivo di questa sezione riguarda la ricerca di dinamiche di dipendenza tra l'andamento delle quote di eventi sportivi (calcio) e statistiche di gioco.

Per fare ciò, analizziamo le fonti betfair.it e diretta.it.

Le determinanti delle quote di partenza delle squadre di calcio sono molteplici, che ne evidenziano le probabilità di successo; squadre più forti sulla carta, presenteranno più probabilità di successo e quindi minor quota.

Nel corso della partita, però, le quote sono libere di muoversi ed il mercato ne determina le fluttuazioni. Le aspettative degli scommettitori dipendono da ciò che si vede durante la partita e, quindi, dalle statistiche di gioco, che ne racchiudono le dinamiche (ma anche dai commenti delle azioni si potrebbero trarre informazioni importanti).

Il nostro lavoro si fonda su questa ricerca.

Abbiamo deciso di valutare le quote sull'esito finale "1X2" e 5 mercati diversi di under/over ("0.5","1.5","2.5","3.5","4.5"). In aggiunta abbiamo considerato anche i volumi monetari abbinati in valuta EUR.

Le statistiche di gioco presenti su diretta.it sono molteplici: abbiamo deciso di considerare come principali il possesso palla, il numero di attacchi pericolosi e i tiri totali delle due squadre.

L'assunto iniziale riguarda il fatto che, quanto più una squadra domina sull'altra, presentando statistiche più marcate, tanto più le aspettative su essa cresceranno, costringendo i giocatori ad effettuare puntate sulla stessa, influenzandone i prezzi, che caleranno.

Eventi come i calci d'angolo e le punizioni dal limite, essendo molto pericolosi, portano ad oscillazioni di breve periodo di qualche punto percentuale, mentre eventi più importanti come un cartellino rosso o un infortunio di giocatori chiave, creano degli scostamenti evidenti, anche dell'ordine del 100%.

Come aggiunta, abbiamo valutato anche i volumi abbinati sui vari mercati.

Per ultimo, ci siamo focalizzati sui mercati degli under che, per loro caratteristica, convergono per definizione verso quota 1 (perchè con il passare del tempo e mantenendosi vincente la selezione, l'esito under risulterà sempre più probabile, fino a che, a partita finita, l'esito diverrà certo, chiudendo a 1).

La domanda che ci siamo posti è stata: a seguito di un gol, dove si collocherà il nuovo prezzo di un under? La risposta non è banale: si collocherà, in media, nel prezzo del mercato di gerarchia inferiore, nel momento appena precedente il goal.

Abbiamo osservato che, a seguito di un goal, si genera grande volatilità, ed il prezzo di atterraggio da noi stimato sarà la risultante di un equilibrio di prezzo tra domanda e offerta: ci saranno pertanto diverse opportunità di profitto, avendo a disposizione un trading system automatizzato.

Esso sarà altresì utile non solo nella previsione del prezzo di atterraggio a seguito di un goal, ma anche durante tutta la durata della partita. Avendo dimostrato che il decremento delle quote è costante nel tempo, le inefficienze sono molte e le opportunità di profitto altrettante.