Analisi del ritardo dei treni nel mese di giugno 2022

Data Management

Andreea Maria Dobre (882514), Luca Sammarini (884591), Eugenio Tarolli Bramè (889038).
CDLM in Data Science, Università degli Studi di Milano-Bicocca
Anno Accademico 2021/2022

Indice

1 Introduzione

Nella quotidianità, un numero elevato di persone utilizza come mezzo di trasporto pubblico il treno per i propri spostamenti, che sia per lavoro o per motivi di studio.

Il fenomeno del pendolarismo è infatti un fenomeno molto sviluppato in Italia, soprattutto al nord e in prossimità di grandi città [1].

Uno studio recente del 31 dicembre 2019 calcola che siano 30.214.401 le persone che effettuano spostamenti quotidiani per recarsi al luogo di studio o di lavoro; ciò ha evidenziato un aumento rispetto ad uno studio precedente del 2011 dove le persone erano 28.871.447; si tratta quindi di un incremento del 2.1% e ciò sottolinea l'aumento di importanza assunto dalle ferrovie del nostro Stato [2].

In questo progetto, abbiamo deciso di focalizzare la nostra attenzione sui ritardi che i treni accumulano, creando grossi disagi ai passeggeri.

Utilizzando le API del sito *ViaggiaTreno*, siamo stati in grado di raccogliere dati nel mese di Giugno per circa tre settimane, più precisamente dal 9 al 30 giugno. Nei dati raccolti, si trovano l'orario di partenza e arrivo dei treni dalle stazioni ferroviarie; in questo modo si è in grado di poter ricavare ritardo o anticipo di ogni treno.

2 Data Ingestion e Caricamento

Come primo step, ci si è focalizzati sulla raccolta dei dati per creare la base da cui partire per la creazione del dataset finale. Le informazioni principali da ricavare sono relative alle diverse stazioni italiane e a tutti i treni che passano giornalmente per quelle stazioni.

Di seguito saranno trattati i passaggi compiuti per ottenere il dataset.

2.1 Creazione del file contenente l'elenco e il codice identificativo delle stazioni ferroviarie italiane

L'obiettivo iniziale è stato quello di recuperare dal portale *ViaggiaTreno* di *Trenitalia* il nome e il rispettivo codice di tutte le stazioni presenti in Italia. L'elenco scaricato verrà successivamente impiegato per ottenere

i dati di tutti i treni circolanti. Per l'acquisizione si è utilizzata la tecnica API che ci ha permesso di interrogare il portale al seguente url:

`http://www.viaggiatreno.it/infomobilita/resteasy/viaggiatreno/autocompletaStazione/` + lettera dell'alfabeto.

È stato però necessario implementare il codice per ogni lettera dell'alfabeto tramite l'utilizzo di una struttura iterativa. Non esiste infatti un metodo per ottenere tutte le stazioni con un solo comando.

Una volta eseguito ciò, il risultato da noi ottenuto si presenta nella forma di un elenco di stazioni e i relativi codici identificativi.

Le informazioni raccolte sono immagazzinate all'interno di un file *.csv* con il nome di *stazioni.csv*, che si presenta come in Figura 1.

	nome_stazione	id
0	ABANO TERME	S05700
1	ABBADIA LARIANA	S01416
2	ABBASANTA	S12873
3	ABBIATEGRASSO	S01062
4	ACATE	S12409
...	...	...
2939	ZAPPULLA	S12029
2940	ZERBINATE	S05163
2941	ZINASCO NUOVO	S00401
2942	ZOAGLI	S04721
2943	ZURICH ALTSTETTEN	S03001

Figura 1: *stazioni.csv*

Sono state ottenute **2944** stazioni in ordine alfabetico.

2.2 Creazione del file contenente il numero di binari per ogni stazione

Con l'obiettivo di creare un dataset composto da *nome stazione*, *codice identificativo* e *numero binari* è stato necessario interrogare ulteriori fonti web. Abbiamo quindi interrogato, con la tecnica di Web Scraping, il sito ufficiale di *RFI* e le pagine *Wikipedia* delle stazioni ferroviarie italiane.

Questa operazione è stata eseguita su due siti differenti poiché un solo sito non è stato sufficiente per ottenere il numero di binari di tutte le stazioni scaricate.

Si descrivono ora gli script implementati per il raggiungimento del nostro obiettivo. Nel caso del sito di RFI si è utilizzata la libreria *Selenium* per l'interazione con la pagina web (`https://www.rfi.it/it/stazioni.html`) e successivamente la libreria *Beautiful Soup* per lo scraping . Si sono così scaricate tutte le stazioni presenti sul sito con il rispettivo numero di binari. Nel caso di Wikipedia sono state utilizzate le API di Wikidata (`https://wikidata.reconci.link/it/api?queries=`) per ottenere, tramite reconciliation, gli URL delle pagine Wikipedia delle stazioni mancanti; successivamente si è proceduto allo scraping tramite *Beautiful Soup*.

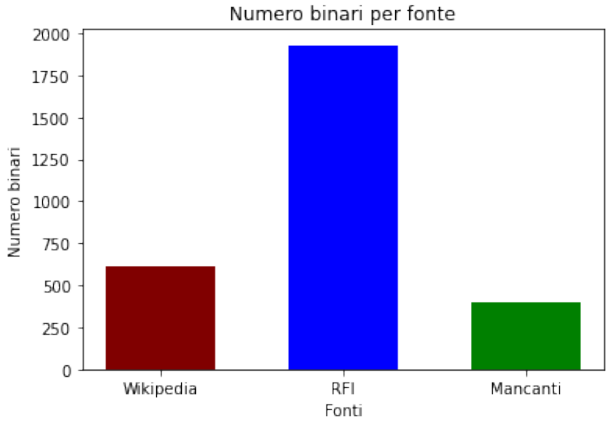

Figura 2: *1929 binari sono stati scaricati dal sito di RFI, 614 dalle pagine di Wikipedia e delle restanti 399 stazioni non è stato possibile reperire il numero di binari.*

2.3 Orario di partenza e arrivo dei treni

Ulteriore step di questo progetto è quello di poter trovare i singoli treni che circolano ogni giorno nel Paese. Ciò è stato possibile mediante la creazione di due liste di dizionari, ottenuti tramite le API di ViaggiaTreno. La prima tipologia di dizionari consiste in coppie chiave-valore costituite rispettivamente dal codice del treno e dal codice identificativo della stazione da cui parte il treno; la seconda consiste in coppie chiave-valore costituite rispettivamente dal codice del treno e dall'orario di partenza del treno in formato UNIX. Grazie a questi sono state interrogate le API di *ViaggiaTreno* allo scopo di ottenere i dati sui treni circolanti (`http://www.viaggiatreno.it/infomobilita/resteasy/viaggiatreno/andamentoTreno/`codice stazione/codice treno/orario di partenza in formato UNIX).

I dati ottenuti per ogni treno sono in formato json e contengono 86 campi.

Il download è stato agevolato da una automatizzazione implementata la quale consiste di un timer che, ad orario prestabilito (ore 22:50), richiede al portale di *ViaggiaTreno* le informazioni relative ai treni dell'intera giornata appena trascorsa. L'orario è stato stabilito secondo il criterio che il tempo computazionale del programma implementato potesse terminare prima delle ore 24:00, orario in cui *Trenitalia* cancella i dati della giornata conclusasi. Ciò ha portato alla perdita di qualche valore.

In Figura 3 è possibile osservare l'implementazione del timer sopra citato.

```
# TIMER
x=datetime.datetime.today()
y=x.replace(day=x.day, hour=22, minute=50, second=0, microsecond=0)
delta_t=y-x
secs=delta_t.seconds+1
```

Figura 3: *Regolazione del "timer" alle ore 22.50*

Lo step successivo è consistito nel salvare i dati appena scaricati, in formato json, sia in locale sia in MongoDB; quest'ultimo è un DBMS non relazionale, orientato ai documenti, come verrà descritto successivamente.

Come già affermato, è stato necessario svolgere questa operazione per un tempo prolungato con l'intento di poter ricoprire la durata di circa tre settimane. Così facendo si sono considerati anche treni con frequenza minore (non giornalieri ma settimanali). La presa dati è stata svolta dal 9 giugno 2022 al 30 giugno 2022. È necessario segnalare che in data 17 giugno il programma ha impiegato più tempo per il download dei dati, non riuscendo ad ultimare lo scaricamento prima di mezzanotte, comportando la perdita di 881 treni su 8280.

Sono stati ottenuti in totale i dati di **166.524** treni.

In seguito alle procedure appena presentate ed elencate, il materiale recuperato è il seguente:

- nome stazione;
- codice identificativo stazione;
- numero dei binari per ogni stazione;
- stato orario di ciascun treno.

3 Data Preparation

Prima di procedere con il resto del lavoro, è stato importante unire tutti i dataset con i dati delle stazioni allo scopo di realizzare un unico dataset.

Innanzitutto si è proceduto alla unificazione dei primi due dataset, quello delle stazioni di ViaggiaTreno con quello delle stazioni di RFI. Nel tentativo di unificazione ci si è accorti che i nomi delle stazioni non erano totalmente coincidenti e si è quindi proceduto all'unificazione utilizzando la libreria **RecordLinkage**. Questa è utile per collegare i record all'interno o tra le origini dati basandosi sulla somiglianza sintattica tra le stringhe. Si sono quindi aggregati i due dataset utilizzando il campo del nome della stazione per tutte le stazioni che avessero una somiglianza superiore al 90% secondo l'alogritmo di Jaro-Winkler. Si è scelto di utilizzare questo valore in quanto ad una analisi di un sottoinsieme di dati, si è visto che la correttezza dell'associazione fosse superiore al 90%.

Facendo ciò si è potuto osservare come non fossero stati ottenuti i binari di tutte le stazioni. Come detto in precedenza, sono state selezionate le stazioni mancanti e tramite scraping si è proceduto all'interrogazione delle rispettive pagine Wikipedia.

A questo punto è stato necessario unire il dataset ottenuto in precedenza con quest'ultimo sul campo *nome stazione*. Si è osservato come mancassero ancora 399 valori ma non essendo stato possibile reperire ulteriori fonti si è deciso di terminare la fase di acquisizione dati considerando quei campi come nulli. In questo modo si è creato il dataset finale chiamato *stazioni_finale.csv*, mostrato in Figura 4.

	id	nome_stazione	binari_finale
0	S05700	ABANO TERME	2.0
1	S01416	ABBADIA LARIANA	2.0
2	S12873	ABBASANTA	3.0
3	S01062	ABBIATEGRASSO	2.0
4	S12409	ACATE	1.0
...	...	...	...
2935	S12029	ZAPPULLA	2.0
2936	S05163	ZERBINATE	1.0
2937	S00401	ZINASCO NUOVO	1.0
2938	S04721	ZOAGLI	2.0
2939	S03001	ZURICH ALTSTETTEN	NaN

Figura 4: *stazioni_finale.csv*

4 Data Storage

Come luogo di raccolta dei dati abbiamo pensato a MongoDB il quale è un DBMS document-based dove i dati raccolti sono in formato json ed è sono modellabili con grande semplicità. Un'altra caratteristica molto importante è che si tratta di un database con grande scalabilità; ciò significa che la struttura stabilita rimane invariata anche con un quantitativo differente di file json. Infatti, se la nostra presa dati fosse durata per un anno intero, il caricamento e il trattamento dei documenti su MongoDB sarebbe stata la medesima di quella implementata in questo progetto. Questo permette, qualora si volessero aggiungere dati a quelli già presenti, di farlo con agilità. Un altro vantaggio di tale DBMS è che permette lo sharding. Ciò consente, qualora i dati dovessero diventare un numero particolarmente elevato, di suddividere il carico di lavoro e il salvataggio su molti calcolatori.

Ogni file json scaricato e successivamente caricato su MongoDB, con la modalità descritta nei paragrafi precedenti, descrive un solo treno; perciò avendo preso in considerazione 166.524 treni, si hanno altrettanti documenti raccolti. I documenti sono stati caricati in un database contenente due collezioni: *TreniRegionali* e *AltriTreni*. Si è deciso di procedere con questa suddivisione in quanto i treni regionali rappresentano il 91.5% dei treni totali. Non aveva quindi senso suddividere i treni in ulteriori collezioni.

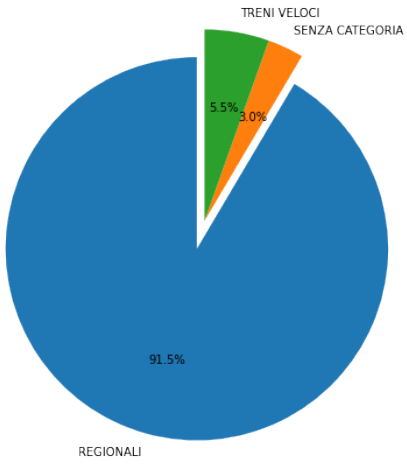

Figura 5: *Categoria treni: 91.5% sono treni regionali, 5.5% sono treni ad alta velocità e il restante 3% è costituito da treni il cui valore di categoria non è stato catalogato.*

4.1 Data Modeling


```
_id: ObjectId('631f5020be5a391810538c33')
numero_treno: "12113"
categoria: "REG"
stato: "regolare"
origine: "ACQUI TERME"
id_origine: "S00867"
destinazione: "GENOVA BRIGNOLE"
id_destinazione: "S04702"
data: "09-06-2022"
ritardo_finale: 4
fermate: Array
  0: Object
  1: Object
  2: Object
    tipo_fermata: "F"
    nome_stazione: "PRASCO CREMOLINO"
    id_stazione: "S04101"
    binari_tot_stazione: "2.0"
    arrivo_teorico: "05:25"
    arrivo_reale: "05:24"
    ritardo_arrivo: -1
    partenza_teorica: "05:26"
    partenza_reale: "05:27"
    ritardo_partenza: 1
```


Figura 6: *File MongoDB con tutte le categorie. In figura, treno numero 12113 con stazione di partenza ad ACQUI TERME e arrivo a GENOVA BRIGNOLE.*

Tra tutti i dati scaricati dal sito *ViaggiaTreno*, si è deciso di mantenere nel database finale quelli mostrati in figura 6. Infatti, di tutti gli 86 campi scaricati, una parte consistente risultava nulla e altri risultavano ridondanti. I campi scelti, 21, sono quelli ritenuti più indicativi e utili allo scopo.

Di seguito una descrizione dei campi utilizzati.

- **_id**: codice identificativo univoco assegnato al documento da MongoDB durante il caricamento;
- **numero_treno**: codice identificativo del treno;
- **categoria**: tipologia di treno; valori possibili: "REG", "MET", "IC", "EC", "NCL", "EN";
- **stato**: "regolare", "parzialmente soppresso", "cancellato";
- **origine**: nome della stazione di origine;
- **id_origine**: codice identificativo univoco della stazione di origine;
- **destinazione**: nome della stazione di destinazione;
- **id_destinazione**: codice identificativo univoco della stazione di destinazione;
- **data**: data del viaggio;
- **ritardo_finale**: ritardo del treno all'arrivo all'ultima stazione in minuti (il valore assunto da questo campo può risultare essere negativo nell'eventualità che il treno arrivi in anticipo);
- **fermate**: array delle fermate eseguite dal treno, se il treno ha come stato "cancellato", il campo ha valore nullo. Ogni elemento dell'array fermate possiede i seguenti campi:
  - **tipo_fermata**: indicazione del tipo di fermata tra partenza, "P", fermata intermedia, "F", e arrivo "A";
  - **nome_stazione**: nome della stazione in cui si effettua la fermata;
  - **id_stazione**: codice identificativo univoco della stazione in cui si effettua la fermata;
  - **binari_tot_stazione**: numero totale di binari della stazione in cui si effettua la fermata;
  - **arrivo_teorico**: orario teorico di arrivo alla stazione (per le fermate di tipo "F" ed "A");
  - **arrivo_reale**: orario effettivo di arrivo alla stazione (per le fermate di tipo "F" ed "A");
  - **ritardo_arrivo**: ritardo di arrivo in minuti (per le fermate di tipo "F" ed "A");
  - **partenza_teorica**: orario teorico di partenza dalla stazione (per le fermate di tipo "P" ed "F");
  - **partenza_reale**: orario effettivo di partenza dalla stazione (per le fermate di tipo "P" ed "F");
  - **ritardo_partenza**: ritardo di partenza in minuti (per le fermate di tipo "P" ed "F").

4.2 Query con MongoDB

Abbiamo creato alcune query con le quali è stato possibile interrogare il dataset creato in MongoDB. Ciò ci ha permesso di conoscere più nel dettaglio i valori dei file treni utilizzati e di rendere più rapida la loro lettura.

4.3 Prima query

Si sono creati due campi nel dataset di MongoDB tramite le query che si possono osservare in Figura 7 e

Figura 8. Ciò ha permesso quindi di ampliare la rapidità di lettura del dataset creato fornendo con maggior rapidità le informazioni relative ai due campi creati, ovvero: *numero_fermate* e *ritardo_medio_treno* (Figura 9).

```
pipeline = [{"$project":
                {"numero_fermate": { "$cond": { "if": {"$isArray": "$fermate"},
                                                "then": {"$size": "$fermate"}, "else": None}}
                }}]

lista_regionali = list(treni_regionali.aggregate(pipeline))
lista_altri_treni = list(altri_treni.aggregate(pipeline))

for elemento in lista_regionali :
    indice = elemento["_id"]
    numero_fermate = elemento["numero_fermate"]
    treni_regionali.find_one_and_update({"_id": indice}, {"$set": {"numero_fermate": numero_fermate}})

for elemento in lista_altri_treni:
    indice = elemento["_id"]
    numero_fermate = elemento["numero_fermate"]
    altri_treni.find_one_and_update({"_id": indice}, {"$set": {"numero_fermate": numero_fermate}})
```

Figura 7: *Query 1.a: creazione campo numero_fermate.*

```
pipeline = [{"$unwind":"$fermate"},
            {"$group": {"_id": {"id": "$_id", "numero_treno": "$numero_treno",
                                "id_origine": "$id_origine", "id_destinazione": "$id_destinazione"},
                        "ritardo_medio_treno": {"$avg": {"$cond":
                                                         [{"$eq":["$fermate.tipo_fermata", "P"]},
                                                         "$fermate.ritardo_partenza",
                                                         "$fermate.ritardo_arrivo"]
                                                         }}}}]

lista_regionali = list(treni_regionali.aggregate(pipeline))
lista_altri_treni = list(altri_treni.aggregate(pipeline))

for elemento in lista_regionali :
    indice = elemento["_id"]["id"]
    ritardo_medio_treno = elemento["ritardo_medio_treno"]
    treni_regionali.find_one_and_update({"_id": indice}, {"$set": {"ritardo_medio_treno": ritardo_medio_treno}})

for elemento in lista_altri_treni:
    indice = elemento["_id"]["id"]
    ritardo_medio_treno = elemento["ritardo_medio_treno"]
    altri_treni.find_one_and_update({"_id": indice}, {"$set": {"ritardo_medio_treno": ritardo_medio_treno}})
```

Figura 8: *Query 1.b: creazione campo ritardo_medio_treno.*

```
_id: ObjectId('631f5020be5a391810538c33')
numero_treno: "12113"
categoria: "REG"
stato: "regolare"
origine: "ACQUI TERME"
id_origine: "S00867"
destinazione: "GENOVA BRIGNOLE"
id_destinazione: "S04702"
data: "09-06-2022"
ritardo_finale: 4
› fermate: Array
numero_fermate: 14
ritardo_medio_treno: 3.4545454545454546
```

Figura 9: *Query 1: File MongoDB in cui è possibile osservare la creazione di due campi: numero_fermate e ritardo_medio_treno.*

Dalla Figura 9 si può osservare come il treno 12113 che percorre la tratta Acqui Terme Genova Brignole compia 14 fermate e che abbia un tirardo medio di 3.45 minuti.

4.4 Seconda query

Nella query presente in Figura 10 si è voluto osservare il ritardo medio per ogni treno regionale. I risultati del ritardo sono espressi in minuti e riportati in ordine decrescente di ritardo; è possibile osservarne i risultati in Figura 10.

```
pipeline = [{"$group": {"_id": {"numero_treno":"$numero_treno",
                                "id_origine":"$idOrigine",
                                "id_destinazione":"$idDestinazione"},
                        "ritardo_medio": {"$avg": "$ritardo_finale"}}},
            {"$sort": SON ([("ritardo_medio", -1)])}]

lista_ritardi_regionali = pd.DataFrame(list(treni_regionali.aggregate(pipeline)))
```

Figura 10: *Query 2: ritardo medio per ogni treno regionale.*

	_id	ritardo_medio
0	{'numero_treno': '25772'}	79.000000
1	{'numero_treno': '23748'}	55.000000
2	{'numero_treno': '17477'}	45.500000
3	{'numero_treno': '5373'}	43.000000
4	{'numero_treno': '21068'}	40.666667
5	{'numero_treno': '1708'}	38.500000
6	{'numero_treno': '25762'}	38.000000
7	{'numero_treno': '18131'}	36.500000
8	{'numero_treno': '10023'}	36.333333
9	{'numero_treno': '22410'}	36.312500

Figura 11: *Query 2: output dei primi 10 treni regionali con maggior ritardo in minuti.*

4.5 Terza query

Nella terza query tramite il comando "*unwind*" si è selezionato il vettore fermate e per ogni stazione si sono ottenuti sia il numero di binari sia i ritardi medi per ogni stazione. Si possono osservare il codice utilizzato e i valori di output rispettivamente alla Figura 12 e Figura 13.

```
pipeline = [{"$unwind":"$fermate"},
            {"$group": {"_id": "$fermate.nome_stazione",
                        "binari": {"$first": "$fermate.binari_tot_stazione"},
                        "ritardo_medio_stazione": {"$avg": {"$cond":
                                                            [{"$eq":["$fermate.tipo_fermata", "P"]},
                                                            "$fermate.ritardo_partenza",
                                                            "$fermate.ritardo_arrivo"]
                                                            }}}}]

lista_ritardi_regionali_stazione = pd.DataFrame(list(treni_regionali.aggregate(pipeline)))
lista__ritardi_altri_treni_stazione = pd.DataFrame(list(altri_treni.aggregate(pipeline)))
```

Figura 12: *Query 3: ritardo medio per fermata*

	_id	binari	ritardo_medio_stazione
0	BELLA MURO	2.0	10.909091
1	VAGLIA	2.0	NaN
2	STRESA	4.0	6.516432
3	MONTEROSSO	3.0	5.950166
4	PERUGIA	4.0	4.065934
...	...	...	...
390	NOLA INTERPORTO	NaN	13.000000
391	SPOLETO	3.0	4.534884
392	SARZANA	4.0	4.107784
393	NOVI LIGURE	3.0	2.307692
394	MESSINA MARITTIMA	6.0	19.846154

Figura 13: *Query 3: output di 10 stazioni con i relativi binari e ritardi medi.*

5 Data Quality

Per questo tipo di analisi sono stati scelti i parametri di Completezza e di Currency in quanto sono stati ritenuti essere le due dimensioni più adatte al nostro lavoro.

5.1 Completezza

La completezza corrisponde alla copertura con la quale il fenomeno osservato è rappresentato nell'insieme di dati; ha quindi il compito di misurare la disponibilità di tutti quei dati che ci si aspetta di ottenere da un progetto di studio.

Inizialmente è interessante calcolare la completezza dei valori dei binari delle stazioni. Infatti, nonostante siano stati presi dati da fonti diverse, non si è riusciti a trovare tutti i dati dei binari necessari. Analizzando il dataset *stazioni_finale.csv*, è possibile osservare che mancano ancora 399 valori di binari. Abbiamo quindi una completezza **binari** di circa **l'86%**.

Riguardo alla completezza degli altri campi tramite una query sviluppata in Python è stato possibile osservare le completezze riportate in tabella.

Campo	Val. nulli	%
binari	399	86%
fermata_nome_stazione	0	100%
categoria	5078	97%
stato	0	100%
origine	0	100%
destinazione	0	100%
id_origine	0	100%
id_destinazione	0	100%
data	0	100%
ritardo_finale	0	100%
numero_fermate	0	100%
ritardo_medio_treno	4055	98%
tipo_fermata	0	100%
fermata.nome_stazione	0	100%
fermata.id_stazione	0	100%
fermata.binari_tot_stazione	68284	96%
fermata.arrivo_teorico	0	100%
fermata.arrivo_reale	372445	81%
fermata.partenza_teorica	0	100%
fermata.partenza_reale	167688	91%

5.2 Currency

La currency misura con quale rapidità i dati sono aggiornati.

Nel nostro caso specifico abbiamo proceduto al calcolo di questa dimensione prendendo in considerazione il valore del campo *fermata.arrivo_teorico* e l'orario in cui ogni giorno si azionava il timer; come detto in precedenza quest'ultimo attivava la tecnica di API sul portale di *ViaggiaTreno* e immediatamente caricava su MongoDB il file json del treno appena scaricato. Considerati quindi questi due orari, arrivo del treno e azionamento del timer, si è proceduto a creare una colonna denominata *differenza* che consiste semplicemente nel sottrarre questi due campi ottenendo così il valore di currency relativo ad ogni treno. Come ultimo step abbiamo calcolato la currency media come la media di tutti i valori appena calcolati; questo valore è pari a 07h 07m 08s per il collection *AltriTreni* e pari a 08h 20m 14s per il collection *TreniRegionali*. È possibile osservare i singoli valori in Figura 14 e Figura 15.

	_id	ora	differenza
0	{'id': 631f50ccbe5a39181053b8e2, 'n_t': '20781'}	12:17	0 days 10:33:00
1	{'id': 631f5f28be5a391810560c42, 'n_t': '4636'}	15:37	0 days 07:13:00
2	{'id': 631f57b3be5a39181054d027, 'n_t': '21128'}	21:54	0 days 00:56:00
3	{'id': 631f5020be5a391810539d73, 'n_t': '21292'}	19:45	0 days 03:05:00
4	{'id': 631f52adbe5a39181054062b, 'n_t': '17977'}	23:02	0 days 00:00:00

Figura 14: *Valori currency TreniRegionali*

	_id	ora	differenza
0	{'id': 631f5438be5a391810545ac2, 'n_t': '608'}	15:00	0 days 07:50:00
1	{'id': 631f52aebe5a391810541c3e, 'n_t': '20979'}	18:08	0 days 04:42:00
2	{'id': 631f5178be5a39181053e92a, 'n_t': '1964'}	16:20	0 days 06:30:00
3	{'id': 631f566abe5a39181054b558, 'n_t': '9405'}	12:48	0 days 10:02:00
4	{'id': 631f5178be5a39181053e87e, 'n_t': '9811'}	21:48	0 days 01:02:00

Figura 15: *Valori currency AltriTreni*

6 Osservazioni finali e sviluppi futuri

Il lavoro svolto raggiunge l'obiettivo che prefissato, ovvero studiare i ritardi dei singoli treni regionali ed alta velocità che ogni giorno percorrono le ferrovie del nostro Paese. Inoltre è stato possibile creare un dataset che contenesse il numero di binari di circa l'86% delle stazioni italiane. Il lavoro svolto risulta soddisfacente in quanto non risulta essere facilmente accessibile un dataset che associ le stazioni al numero di binari presente in ognuna di esse.

Come sviluppo futuro si ritiene sia necessario ottenere i valori dei binari delle restanti stazioni rendendo più completo il lavoro svolto dai membri di questo gruppo. Si reputa inoltre essere di grande aiuto, per l'esplorazione e il pieno raggiungimento dell'obiettivo, il poter estendere la ricerca non solo a 3 settimane del mese di giugno ma anche ad un intero anno solare. In questo modo è possibile ottenere un numero maggiore di treni aumentando la statistica dei ritardi e rendendone più preciso il valore. Tutto ciò, avendo utilizzato MongoDB si può svolgere senza cambiare la struttura del dataset che si è riusciti a creare in questo progetto.

Bibliografia

[1] *Mobilità giornaliera per studio o lavoro*. URL: https://www.urbanindex.it/indicatori/mobilita-giornaliera-per-studio-o-lavoro/.

[2] *Gli spostamenti per motivi di studio o lavoro nel 2019 secondo il censimento permanente della popolazione*. URL: https://www.istat.it/it/archivio/257621.