



**Politecnico
di Torino**

Politecnico di Torino

Corso di Laurea in Ingegneria Gestionale L8

a.a. 2021/2022

Sessione di Laurea Ottobre 2022

Valutazione del mercato immobiliare di Melbourne

Relatore:

Prof. Fulvio Corno

Candidato:

Sferlazzo Daniele (S272348)

Sommario

Capitolo 1: Proposta di Progetto.....	4
1.1 Studente proponente.....	4
1.2 Titolo della proposta.....	4
1.3 Descrizione del problema proposto.....	4
1.4 Descrizione della rilevanza gestionale del problema.....	4
1.5 Descrizione dei data-set per la valutazione.....	4
1.6 Descrizione preliminare degli algoritmi coinvolti.....	5
1.7 Descrizione preliminare delle funzionalità previste per l'applicazione software.....	6
Capitolo 2: Descrizione dettagliata del problema affrontato.....	7
Capitolo 3: Descrizione dei data-set utilizzati.....	8
3.1 Tabella "Properties"	8
3.2 Tabella "Tourism"	9
Capitolo 4: Descrizione delle strutture dati e degli algoritmi utilizzati.....	11
4.1 Strutture dati.....	11
4.1.1 it.polito.tdp.tesi.....	11
4.1.2 it.polito.tdp.tesi.model.....	12
4.1.3 it.polito.tdp.tesi.db.....	12
4.2 Algoritmi utilizzati.....	12
4.2.1 EstateDao.....	12
4.2.2 Model.....	16
4.2.3 RicorsioneMensilRent.....	19
4.2.4 Simulatore.....	20
Capitolo 5: Diagramma delle classi principali.....	26
5.1 Descrizione delle classi.....	26
5.2 Diagramma delle classi.....	28
Capitolo 6: Interfaccia utente e video dimostrativo.....	29
6.1 Fase iniziale: creazione del portafoglio di investimenti.....	30
6.2 Seconda fase: calcolo del tempo di ritorno del capitale investito.....	31
Capitolo 7: Risultati sperimentali.....	33

7.1 Lista di dimensioni massime non filtrata.....	33
7.2 Lista filtrata.....	34
7.3 Ricorsione con lista di piccole dimensioni.....	35
7.4 Ricorsione con lista di grandi dimensioni.....	36
7.5 Algoritmo di simulazione.....	36
Capitolo 8: Valutazione dei risultati ottenuti.....	39
Capitolo 9: Conclusioni.....	40
Capitolo 10: Licenza.....	41

Capitolo 1: Proposta di Progetto

1.1 Studente proponente

S272348 Sferlazzo Daniele

1.2 Titolo della proposta

Valutazione del mercato immobiliare di Melbourne e realizzazione di un apposito piano di acquisti

1.3 Descrizione del problema proposto

Il mercato immobiliare è a volte complicato da valutare a causa del grande numero di offerte ed abitazioni disponibili all'acquisto. Questo programma consente di analizzare le varie offerte immobiliari, andando a fornire all'utente vari filtri per le abitazioni da selezionare e genera un valido piano di investimenti per l'agenzia immobiliare o per l'investitore del settore, oltre a calcolare un periodo di ritorno del capitale così investito.

1.4 Descrizione della rilevanza gestionale del problema

Andando a supportare le decisioni degli investitori del settore, il programma fornisce una valida lista di abitazioni da acquistare ed un elenco delle zone più proficue, migliorando la scelta degli investitori, diminuendo il rischio di perdita di capitale investito e stimando un eventuale periodo di ritorno del capitale investito tramite il guadagno generato dall'affitto degli immobili così acquistati (considerando anche la possibilità che la casa non venga affittata in determinati periodi).

1.5 Descrizione dei data-set per la valutazione

Per il programma verranno utilizzati due data-set:

<https://www.kaggle.com/datasets/dansbecker/melbourne-housing-snapshot>

Il primo è un database in cui ogni riga rappresenta un'abitazione, con il suo prezzo, la zona e altre caratteristiche.

In particolare verranno utilizzate le seguenti informazioni:

- Address: chiave univoca dei dati
- Suburb, PropertyCount e RegionName: il quartiere in cui è situata l'abitazione, il numero di abitazioni nello stesso ed il nome della Regione in considerazione
- Price: il prezzo di vendita dell'abitazione
- Altre informazioni utilizzate per filtrare i vari edifici quali la dimensione, il numero di stanze, la regione, etc...

<https://www.kaggle.com/datasets/luisblanche/quarterly-tourism-in-australia>

Il secondo dataset contiene le informazioni sul turismo stagionale in Australia, tramite il quale si potrà calcolare la probabilità che un'abitazione venga affittata in un determinato mese dell'anno.

1.6 Descrizione preliminare degli algoritmi coinvolti

In una fase preliminare, per supportare le varie decisioni dell'utente, vengono calcolate le migliori regioni in cui poter investire, tramite l'analisi del prezzo medio di vendita degli immobili e del numero di abitazioni nella regione.

Nella prima fase del programma, inserito il budget disponibile da investire, un obiettivo dell'analisi ed eventuali filtri, tramite l'utilizzo di un algoritmo ricorsivo viene calcolata una lista di immobili soddisfacenti le condizioni, tale che sia di dimensione massima (ovvero comprenda l'acquisto del maggior numero di immobili) oppure abbia il massimo valore di affitto complessivo.

Nella seconda fase del programma, selezionata un'abitazione tra quelle della lista, partendo dalla probabilità che la casa non venga affittata mensilmente, attraverso l'utilizzo di un algoritmo di simulazione e tramite l'analisi di diversi tipi di affitto, sia a lungo termine sia turistico, viene calcolato il periodo di ritorno del capitale.

1.7 Descrizione preliminare delle funzionalità previste per l'applicazione software

Una volta avviata l'applicazione, viene stampata una lista delle regioni consigliate per effettuare l'acquisto e una lista delle tipologie di appartamenti più costosi. Ciò aiuta l'utente nella selezione di filtri che verranno utilizzati nella parte successiva.

Una volta effettuata la scelta dei filtri, inserito il budget disponibile per l'investimento e selezionato un obiettivo dell'investimento tra:

- maggior numero di appartamenti acquistati
- minor numero di appartamenti acquistati
- maggior capitale investito

il programma inizialmente filtra tutte le abitazioni presenti nel data-set, grazie alle informazioni fornite dall'utente e in seguito, tramite l'utilizzo di un algoritmo ricorsivo, genera una lista di appartamenti che soddisfino le necessità dell'utente.

Generata la lista degli immobili, l'utente ha la possibilità di selezionare uno tra quelli presenti e il programma, considerando la probabilità che l'immobile non venga affittato, effettua un algoritmo di simulazione per ottenere il periodo di ritorno del capitale investito, tramite l'utilizzo di apposite stime.

Capitolo 2: Descrizione dettagliata del problema affrontato

Sempre più investitori vengono scoraggiati all'idea di investire nel settore immobiliare a causa dell'enorme mole di informazioni da valutare prima di avere un'idea chiara su cosa investire, dell'ampia concorrenza nel settore e del lungo periodo di ritorno del capitale investito.

Risulta quindi fondamentale lo sviluppo di un software per il supporto alle decisioni, capace di analizzare il mercato e in grado di fornire all'utente un'adeguata rappresentazione dei possibili investimenti da poter intraprendere.

Il software risulta utile non solo al singolo investitore, che in questo modo ha a disposizione un quadro generale del mercato, ma anche alle varie aziende del settore immobiliare, le quali potranno quindi presentare al cliente una panoramica dell'investimento.

Inizialmente viene mostrata all'utente una panoramica del mercato, attraverso quattro liste:

- quartieri con il maggior numero di abitazioni
- quartieri con costo medio ad abitazione maggiore
- quartieri con costo medio ad abitazione minore
- venditori con il maggior numero di offerte

Partendo da queste informazioni preliminari, l'utente seleziona i filtri ed inserisce il budget massimo, che consentono al software di elaborare un apposito portafoglio di investimenti per l'utente, contenente le abitazioni selezionate in base alle informazioni inserite.

Oltre alla selezione di appositi investimenti, il software si occupa anche di calcolarne il periodo di ritorno del capitale investito, in modo da diminuire il rischio di perdita dell'investimento ed aumentando la sicurezza degli investitori. Per fare ciò, viene utilizzato un algoritmo di simulazione che stima gli affitti mensili della proprietà (tramite l'analisi di dati del turismo di Melbourne) e calcola il numero di mesi affinché il guadagno complessivo degli affitti sia maggiore o uguale al capitale investito inizialmente per l'acquisto della proprietà.

Capitolo 3: Descrizione dei data-set utilizzati

Nel software viene utilizzato un database con due tabelle, ognuna delle quali è stata ottenuta filtrando e rielaborando due diversi data-set.

Il primo data-set contiene tutte le vendite di immobili nella città di Melbourne nel 2016/2017, e l'unica modifica effettuata nel database è l'aggiunta di un campo "#", che funge da identificativo univoco alle varie abitazioni, in quanto il campo "address" non dispone della proprietà di unicità richiesta. Il data-set è disponibile all'indirizzo:

<https://www.kaggle.com/datasets/anthonypino/melbourne-housing-market>

Il secondo data-set racchiude tutte le informazioni del turismo stagionale in Australia. Dopo essere stato filtrato per contenere soltanto le informazioni richieste dal software, ovvero quelle riguardanti la città di Melbourne, è stato poi rielaborato, in modo da risultare più fruibile durante lo sviluppo del programma. Il data-set è disponibile all'indirizzo:

<https://www.kaggle.com/datasets/luisblanche/quarterly-tourism-in-australia>

Qui sotto una descrizione delle versioni finali dei due data-set, ottenute dopo le modifiche effettuate sui due data-set iniziali ed utilizzate nel software.

3.1 Tabella "Properties"

Per lo sviluppo del Software è stato utilizzato in primo luogo un data-set contenente le informazioni riguardanti la vendita di proprietà immobiliari nella città di Melbourne nel 2016/2017.

Questo data-set è di notevole utilità allo sviluppo, in quanto permette di poter catalogare lo svariato numero di offerte in base a numerosi parametri, permettendone così un'analisi migliore.

Ogni riga(entry) della tabella rappresenta un immobile, rappresentato in maniera univoca da un intero incrementale per ogni nuova aggiunta nella tabella (primary key). Oltre a questo, vi sono altri 21 campi nella tabella, ed un totale di 13.580 proprietà. Tra i vari campi, qui sotto quelli maggiormente utilizzati per lo sviluppo del software:

- 1) #: identificativo univoco dell'abitazione;
- 2) Suburb: quartiere;
- 3) Address: indirizzo;
- 4) SellerG: agenzia di vendita;
- 5) PropertyCount: numero di proprietà presenti nel quartiere dell'abitazione;
- 6) Rooms: numero di stanze;
- 7) Car: numero di posti auto;
- 8) Price: prezzo in dollari


#	Nome	Tipo di dati			
 1	#	INT	11	Postcode	INT
2	Suburb	VARCHAR	12	Bedroom	INT
3	Address	VARCHAR	13	Bathroom	INT
4	Rooms	INT	14	Car	INT
5	Type	VARCHAR	15	Landsize	INT
6	Price	INT	16	BuildingArea	INT
7	Method	VARCHAR	17	YearBuilt	INT
8	SellerG	VARCHAR	18	CouncilArea	VARCHAR
9	Date	DATE	19	Latitude	DOUBLE
10	Distance	DOUBLE	20	Longitude	DOUBLE
11	Postcode	INT	21	RegionName	VARCHAR
			22	PropertyCount	INT

Figura 3.1: Rappresentazione delle colonne della tabella “Properties”, con i seguenti tipi di dati.

3.2 Tabella “Tourism”

Il secondo data-set utilizzato rappresenta uno spaccato del turismo nella città di Melbourne, fornendone una rappresentazione stagionale. Questo data-set risulta importante nella seconda

parte dello sviluppo del software, dove per prevedere tramite un apposito algoritmo la probabilità che la casa venga affittata, bisogna effettuare un apposito studio sui dati.

Ogni entry della tabella rappresenta la distribuzione del turismo in un apposito trimestre, tramite 4 diversi tipi di turismo:

- 1) Business (Lavoro);
- 2) Holiday (Vacanza);
- 3) Visiting (Visita);
- 4) Other (Altro);

Nel database sono presenti soltanto le entry dal 2007 al 2016, ma nel data-set iniziale sono presenti dati fino al 1998, che per questione di troppa obsolescenza dei dati non sono stati alla fine considerati nello sviluppo del software.


#	Nome	Tipo di dati
 1	Quarter	DATE
2	Business	DOUBLE
3	Holiday	DOUBLE
4	Visiting	DOUBLE
5	Other	DOUBLE

Figura 3.2: Rappresentazione delle colonne della tabella “Tourism” con i seguenti tipi di dati.

Capitolo 4: Descrizione delle strutture dati e degli algoritmi utilizzati

4.1 Strutture dati

Il software è stato sviluppato in linguaggio Java, ed utilizza un'interfaccia grafica JavaFX. Per organizzare le varie classi sono stati adoperati due paradigmi:

- 1) MVC (Model View Controller), costituito dal Model che fornisce i metodi per accedere ai dati, dalla View che si occupa di visualizzare i dati contenuti nel Model e di gestire le interazioni con l'utente e dal Controller che riceve i comandi dall'utente e modifica lo stato degli altri due componenti;
- 2) DAO (Data Access Object), consistente in una classe che si occupa della rappresentazione in java delle tabelle del database, in modo da fornire i dati richiesti dal database al model

4.1.1 it.polito.tdp.Tesi

Package contenente la classe Main, ovvero la classe eseguibile del programma: La classe EntryPoint si occupa invece di caricare l'interfaccia utente, e di collegarla alla classe Model. L'FXMLController stabilisce le interazioni tra programma ed interfaccia utente, permettendo quindi all'utente di poter inserire valori ed ottenerne un apposito risultato.

4.1.2 it.polito.tdp.Tesi.model

In questo package sono presenti tutte quelle classi che si occupano dell'elaborazione dei dati. Due di queste classi, Property e TourismData, rappresentano le due tabelle del database (rispettivamente properties e tourism), mentre la classe Model si occupa di effettuare tutte le operazioni per l'elaborazione dei dati estrapolate dalla classe Dao. La classe StringAndInt è una semplice classe Wrapper di un intero ed una stringa, utilizzata per riportare dei risultati della classe Dao (utilizzata prevalentemente nella parte di popolazione dei filtri). La classe RicorsioneMensilRent

implementa un algoritmo di ricorsione (contenente una funzione che richiama sé stessa fino al raggiungimento di un caso terminale) per ottenere la lista di abitazioni con il maggiore affitto mensile complessivo. Infine, la classe `Simulatore` contiene quei metodi necessari per effettuare un algoritmo di simulazione (dove dei determinati parametri vengono soggetti ad eventi successivi ed influenzati da un certo grado di casualità, per ottenere lo stato finale del sistema) utilizzato per calcolare il tempo di ritorno del capitale investito, mentre la classe `Event` rappresenta l'evento di affitto mensile della casa, con un campo `data` ed uno `EventType` che rappresenta il tipo di affitto (è presente anche al suo interno un intero che serve per gestire gli affitti stagionali di tipo BUSINESS).

4.1.3 it.polito.tdp.Tesi.db

Questo package contiene tutte le classi che si occupano dell'interazione tra il software ed il database. La classe `DBConnect` si occupa della connessione al database, tramite l'utilizzo della libreria `HikariCP`, mentre `EstateDao` fornisce al `Model` i metodi per estrapolare informazioni dalle due tabelle del database

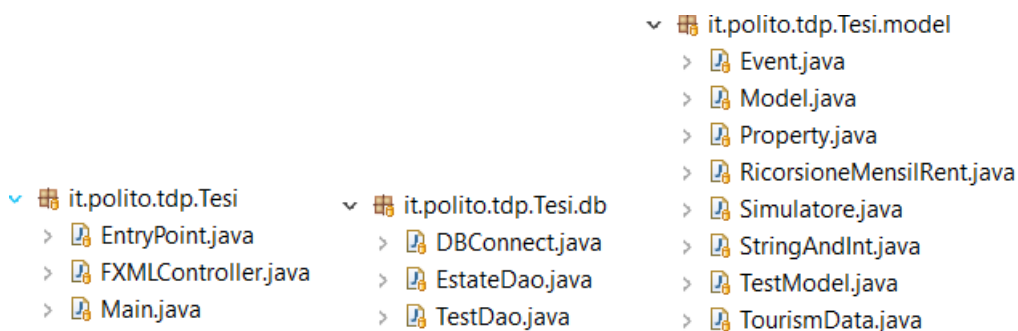


Figura 4.1: Classi utilizzate nel software, suddivisi nei 3 package.

4.2 Algoritmi utilizzati

4.2.1 EstateDao

getAllPropertiesAsc/Desc e getPropertiesFilteredAsc/Desc: questi metodi interrogano il database per restituire una lista di tutte le abitazioni disponibili, in ordine crescente o decrescente di costo

(questo ordinamento risulterà utile per le successive valutazioni). L'unica differenza tra i due metodi sta nel filtraggio delle abitazioni, tramite i filtri inseriti dall'utente, effettuato dai metodi `getPropertiesFilteredAsc/Dsc`. Per similarità degli algoritmi, qui sotto verrà soltanto riportato `getPropertiesFilteredAsc`.

```
public List<Property> getPropertiesFilteredAsc(boolean car, int priceMax, int rooms, String suburb, String sellerG){
    String sql = "SELECT * FROM properties WHERE ";
    if(car) {
        sql += "car>0 AND ";
    }
    sql += "price<=? AND rooms=? AND suburb = ? ";
    if(sellerG != null) {
        sql += "AND sellerG = ? ";
    }
    sql += "ORDER BY price asc";
    try {
        //STABILISCO LA CONNESSIONE
        Connection conn = DBConnect.getConnection();
        PreparedStatement st = conn.prepareStatement(sql);

        List<Property> result = new ArrayList<Property>();

        st.setInt(1, priceMax);
        st.setInt(2, rooms);
        st.setString(3, suburb);

        if(sellerG != null) {
            st.setString(4, sellerG);
        }

        //ESEGUO LA QUERY
        ResultSet res = st.executeQuery();
        //SCANSIONO IL RISULTATO, E LO SALVO NELLA LISTA RESULT
        while(res.next()) {
            Property p = new Property(res.getInt("#"), res.getString("Suburb"), res.getString("Address"), res.getInt("Rooms"),
                res.getString("Type"), res.getInt("Price"), res.getString("Method"), res.getString("SellerG"), res.getDate("Date").toLocalDate(),
                res.getDouble("Distance"), res.getInt("Postcode"), res.getInt("Bedroom"), res.getInt("Bathroom"), res.getInt("Car"),
                res.getInt("Landsize"), res.getInt("BuildingArea"), res.getInt("YearBuilt"), res.getString("CouncilArea"),
                res.getDouble("Latitude"), res.getDouble("Longitude"), res.getString("RegionName"), res.getInt("PropertyCount"));
            result.add(p);
        }
        conn.close();
        return result;
    } catch (SQLException e) {
        e.printStackTrace();
        return null;
    }
}
```

Figura 4.2: Codice del metodo `getPropertiesFilteredAsc`.

`getAllTourismData` e `getLast20TourismData`: metodi semplici che interrogano il database per ottenere tutti i `tourismData` al suo interno, e nel caso di `getAllTourismData` essi verranno restituiti senza successive operazioni, mentre con `getLast20TourismData` vengono prima ordinati in ordine decrescente di data e poi ne vengono salvati soltanto 20 all'interno della lista. Per similarità degli algoritmi, qui sotto verrà riportato soltanto il metodo `getLast20TourismData`.

```

public List<TourismData> getLast20TourismData(){
    final String sql = "SELECT * FROM tourism t ORDER BY t.Quarter desc";
    try {
        //STABILISCO LA CONNESSIONE
        Connection conn = DBConnect.getConnection();
        PreparedStatement st = conn.prepareStatement(sql);

        List<TourismData> result = new ArrayList<TourismData>();

        //ESEGUO LA QUERY
        ResultSet res = st.executeQuery();

        //SCANSIONO IL RISULTATO, E LO SALVO NELLA LISTA RESULT
        for(int i = 0; i < 20; i++) {
            res.next();
            TourismData t = new TourismData(res.getDate("Quarter").toLocalDate(), res.getDouble("Business"),
                res.getDouble("Holiday"), res.getDouble("Visiting"), res.getDouble("Other"));
            result.add(t);
        }
        conn.close();
        return result;
    } catch (SQLException e) {
        e.printStackTrace();
        return null;
    }
}

```

Figura 4.3: Codice del metodo getLast20TourismData.

getSuburbsWMostHouses, getMostExpensiveSuburb/cheapestSuburb, getSellersWMostSells: questi metodi vengono utilizzati nella fase preliminare del programma, dove, dopo l'accensione, vengono mostrate varie informazioni per aiutare l'investitore: quartieri con più case, quartieri più e meno economici, e venditori con il maggior numero di vendite. Per similarità degli algoritmi, qui sotto verrà riportato soltanto il metodo getSuburbsWMostHouses:

```

//FUNZIONI PER OTTENERE LE VALUTAZIONE DI PARTENZA DEL PROGRAMMA
public List<StringAndInt> getSuburbsWMostHouses(){
    final String sql = "SELECT p.Suburb, COUNT(*) AS n "
        + "FROM properties p "
        + "GROUP BY p.Suburb "
        + "ORDER BY COUNT(*) desc";
    try {
        //STABILISCO LA CONNESSIONE
        Connection conn = DBConnect.getConnection();
        PreparedStatement st = conn.prepareStatement(sql);

        List<StringAndInt> result = new ArrayList<StringAndInt>();

        //ESEGUO LA QUERY
        ResultSet res = st.executeQuery();

        //SCANSIONO IL RISULTATO, E LO SALVO NELLA LISTA RESULT
        for(int i = 0; i < 10; i++) {
            res.next();
            StringAndInt si = new StringAndInt(res.getString("Suburb"), res.getInt("n"));
            result.add(si);
        }
        conn.close();
        return result;
    } catch (SQLException e) {
        e.printStackTrace();
        return null;
    }
}

```

Figura 4.3: Codice del metodo getSuburbsWMostHouses.

getAllSuburbs, getAllSellerG: questi metodi interrogano il database per restituire rispettivamente una lista di tutti i quartieri e di tutti i venditori. Vengono utilizzati nella fase preliminare del programma, dove ho bisogno di popolare i menù a tendina (ComboBox) nell'interfaccia utente, in modo da poter essere poi selezionati dall'utente come filtri. Per similarità degli algoritmi, qui sotto verrà riportato soltanto il metodo getAllSuburbs.

```

public List<String> getAllSuburbs(){
    final String sql = "SELECT DISTINCT Suburb FROM properties ORDER BY suburb asc";
    try {
        //STABILISCO LA CONNESSIONE
        Connection conn = DBConnect.getConnection();
        PreparedStatement st = conn.prepareStatement(sql);

        List<String> result = new ArrayList<String>();

        //ESEGUO LA QUERY
        ResultSet res = st.executeQuery();

        //SCANSIONO IL RISULTATO, E LO SALVO NELLA LISTA RESULT
        while(res.next()) {
            String s = res.getString("Suburb");
            result.add(s);
        }
        conn.close();
        return result;
    }catch (SQLException e) {
        e.printStackTrace();
        return null;
    }
}

```

Figura 4.4: Codice del metodo getAllSuburbs.

4.2.2 Model

initialAnalysis: metodo che riceve le informazioni dai metodi: getSuburbsWMostHouses; getMostExpensiveSuburb/cheapestSuburb; getSellersWMostSells. Tutte queste informazioni vengono rielaborate in una singola stringa, che poi verrà stampata su schermo nell'FXMLController, in modo da risultare facilmente fruibili all'utente.


```

public String initialAnalysis() {
    this.dao = new EstateDao();
    String result = "Quartieri più costosi:\n";
    List<StringAndInt> lSi = this.dao.getMostExpensiveSuburb();
    for(StringAndInt si : lSi)
        result += si+"\n";
    result += "\n";

    lSi = this.dao.getCheapestSuburb();
    result += "Quartieri meno costosi:\n";
    for(StringAndInt si : lSi)
        result += si + "\n";
    result += "\n";

    lSi = this.dao.getSuburbsWMostHouses();
    result += "Quartieri con più abitazioni:\n";
    for(StringAndInt si : lSi)
        result += si + "\n";
    result += "\n";

    lSi = this.dao.getSellerWMostSells();
    result += "Venditori con più offerte:\n";
    for(StringAndInt si : lSi)
        result += si + "\n";

    return result;
}

```

Figura 4.5: Codice del metodo initialAnalysis.

getRecommendedList: riceve vari filtri ed un'obiettivo dell'analisi dall'FXMLController, selezionati dall'utente e chiama in base all'obiettivo getPropertiesFilteredAsc o Desc. Nel caso in cui l'obiettivo sia "Maggior numero di abitazioni" o "Minor numero di abitazioni", il metodo elabora il risultato e lo restituisce al controller per venire visualizzato. Se invece l'obiettivo sia "Maggior guadagno mensile", dopo aver ottenuto la lista delle abitazioni si deve chiamare il metodo di ricorsione della classe RicorsioneMensilRent per ottenere la lista da inviare all'FXMLController.

```

public List<Property> getRecommendedList(String obiettivo, int budget, boolean car, int priceMax, String sellerG, String suburb, int rooms){
    this.dao = new EstateDao();

    if(obiettivo.equals("Maggior numero di abitazioni")) { //OBIETTIVO 1
        int somma = 0;
        List<Property> properties = this.dao.getPropertiesFilteredAsc(car, priceMax, rooms, suburb, sellerG);

        if(properties.size()==0) {
            //LISTA VUOTA, NON E' STATA TROVATA NESSUNA ABITAZIONE
            return properties;
        }

        List<Property> resultList = new ArrayList<Property>();
        while(somma<=budget && properties.size()!=0) {
            resultList.add(properties.get(0));
            somma += properties.get(0).getPrice();
            properties.remove(0);
        }
        if(somma>budget) {
            somma -= resultList.get(resultList.size()-1).getPrice();
            resultList.remove(resultList.size()-1);
        }
        return resultList;
    }

    if(obiettivo.equals("Minor numero di abitazioni")) { //OBIETTIVO 2

        if(obiettivo.equals("Minor numero di abitazioni")) { //OBIETTIVO 2
            int somma = 0;
            List<Property> properties = this.dao.getPropertiesFilteredDesc(car, priceMax, rooms, suburb, sellerG);
            if(properties.size()==0) {
                //LISTA VUOTA, NON E STATA TROVATA NESSUNA ABITAZIONE
                return properties;
            }
            List<Property> resultList = new ArrayList<Property>();
            while(somma<=budget && properties.size()!=0) {
                resultList.add(properties.get(0));
                somma += properties.get(0).getPrice();
                properties.remove(0);
            }
            if(somma>budget) {
                somma -= resultList.get(resultList.size()-1).getPrice();
                resultList.remove(resultList.size()-1);
            }

            if(resultList.size()==0) { //GESTISCO CASO LISTA VUOTA
                int fl = 0;
                for(int i = 0; i<properties.size() && fl==0; i++) {
                    if(properties.get(i).getPrice()<budget) {
                        fl=1;
                        resultList.add(properties.get(i));
                    }
                }
            }
            return resultList;
        }

        if(obiettivo.equals("Maggior guadagno mensile")) { //OBIETTIVO 3

        if(obiettivo.equals("Maggior guadagno mensile")) { //OBIETTIVO 3
            //ALGORITMO DI RICORSIONE

            List<Property> properties = this.dao.getPropertiesFilteredDesc(car, priceMax, rooms, suburb, sellerG);

            RicorsioneMensilRent ricorsione = new RicorsioneMensilRent();
            List<Property> resultList = ricorsione.effettuaRicorsione(properties, budget);
            return resultList;
        }

        return null;
    }
}

```

Figura 4.6: codice del metodo getRecommendedList.

4.2.3 RicorsioneMensilRent

effettuaRicorsione: metodo che ha il compito di definire le variabili utilizzate nella ricorsione (grazie anche ai dati ricevuti dal model), e di chiamare la ricorsione stessa. Questo metodo viene richiamato dal model ogni volta che deve ottenere una lista di abitazioni con obiettivo “Maggior guadagno mensile”, poiché il metodo ricorsione effettivo è privato (ovvero è richiamabile soltanto da un metodo all’interno della classe RicorsioneMensilRent, come appunto `effettuaRicorsione`).

```
public List<Property> effettuaRicorsione(List<Property> propertiesDesc, int budget) {
    this.propertiesDesc = propertiesDesc;
    this.budget = budget;

    List<Property> parziale = new ArrayList<Property>();
    int ctParziale = 0;
    int rentParziale = 0;
    this.maxRent = 0;
    this.ctResult = 0;

    this.ricorsione(parziale, ctParziale, rentParziale);

    return resultList;
}
```

Figura 4.7: codice del metodo `effettuaRicorsione`

ricorsione: l’effettivo metodo ricorsivo (che richiama sé stesso fino al raggiungimento di una condizione terminale). Prova prima ad aggiungere una ad una le abitazioni alla lista di soluzione parziale (con le condizioni che l’abitazione non sia già presente nella lista, e che il prezzo complessivo della nuova lista così ottenuta non superi il budget massimo inserito dall’utente) e poi, nel caso in cui non ho potuto aggiungere abitazioni alla lista, raggiunge un caso terminale, dove dopo aver controllato se l’affitto complessivo della lista è maggiore dell’affitto mensile complessivo della lista precedentemente salvata, salvo la soluzione parziale come soluzione (creo una nuova lista a partire dalla soluzione parziale, e la salvo come soluzione).

```

private void ricorsione(List<Property> parziale, int ctParziale, int rentParziale) {

    //CASO NORMALE
    int flAggiunta = 0;
    for(Property p : propertiesDesc) {
        if(!parziale.contains(p) && ctParziale + p.getPrice()<=budget) {    //CHECK SU CONTENIMENTO E SU BUDGET RIMANENTE
            //EFFETTUA RICORSIONE
            flAggiunta = 1;

            parziale.add(p);
            ctParziale += p.getPrice();
            rentParziale += p.getMensilRent();

            //RICORSIONE
            this.ricorsione(parziale, ctParziale, rentParziale);

            //BACKTRACK
            parziale.remove(p);
            ctParziale -= p.getPrice();
            rentParziale -= p.getMensilRent();
        }
    }
    if(flAggiunta == 0 && rentParziale > this.maxRent) {    //NON HO AGGIUNTO NULLA NELLA LISTA, => CASO TERMINALE
        //SOLUZIONE VALIDA, => SALVO SOLUZIONE
        resultList = new ArrayList<Property>(parziale);
        maxRent = rentParziale;
        this.ctResult = ctParziale;
    }
}
}

```

Figura 4.8: codice del metodo ricorsione

4.2.4 Simulatore

init: metodo che inizializza i vari parametri della simulazione e lo stato del mondo. Inoltre chiama il metodo `generateTurismoAttuale` per ottenere i dati attuali del turismo, partendo dai dati precedenti, e genera anche l'evento iniziale (essendo questa una ricorsione particolare, poiché non ho una coda degli eventi, ma soltanto un singolo evento che viene generato ad ogni iterazione del metodo `run`, fino alla condizione di conclusione della simulazione)

```

public void init(Property abitazione, List<TourismData> turismoPrecedente) {
    //INIZIALIZZARE IL PRIMO EVENTO ED I PARAMETRI

    this.abitazione = abitazione;
    this.turismoPrecedente = turismoPrecedente;

    this.guadagno = 0;
    this.affitti = new ArrayList<Event>();
    this.mesiTot = 0;
    this.guadagno = 0;
    this.flAffittoSemestrale = 0;

    //DEVO INIZIALIZZARE IL PRIMO EVENTO
    this.dataAttuale = this.turismoPrecedente.get(0).getQuarter().plusMonths(3);
    this.generateTurismoAttuale();
    this.generateEvent();
}

```

Figura 4.9: codice del metodo init.

run: una volta chiamato questo metodo e fino a quando il guadagno complessivo degli affitti ricevuti non sia maggiore del prezzo di acquisto dell’abitazione, il metodo avanzerà ogni volta la data di un mese, andando a generare in ogni nuova stagione i nuovi dati del turismo attuale tramite il metodo generateTurismoAttuale, e gestirà l’evento di quel mese (ovvero il tipo di affitto). Per fare ciò verrà utilizzato un metodo di previsione detto “moving average”, che utilizza i precedenti 20 periodi per calcolare lo stato del periodo attuale.

```

public void run() {
    int i = 0;
    while(guadagno<abitazione.getPrice()) { //LIMITE DELLA SIMULAZIONE
        dataAttuale = dataAttuale.plusMonths(1); //AGGIORNO LA DATA, POICHE' PASSA UN MESE
        i++;
        if(i==3) {
            i=0;
            this.generateTurismoAttuale();
            //POICHE HO CAMBIATO STAGIONE
        }
        this.handleEvent(eventoAttuale);
    }
}

```

Figura 4.10: codice del metodo run.

handleEvent: metodo che si occupa di gestire il tipo di affitto mensile (suddiviso in Business, Visita, Vacanza, Altro e nessun affitto) aggiungendo eventualmente il guadagno mensile al guadagno totale, e generando un nuovo evento (poiché non ho una lista di eventi, ma ogni volta che gestisco un evento ne devo generare uno successivo). La generazione del nuovo evento è anche influenzata dal tipo di affitto precedente:

- 1) Business: 50% che lo stesso affittuario riaffitti per il prossimo mese ed il 20% che riaffitti per tutta la stagione;
- 2) Holiday: 20% che lo stesso affittuario riaffitti per il prossimo mese;
- 3) Visiting: 10% che lo stesso affittuario riaffitti per il prossimo mese;
- 4) Other: 5% che lo stesso affittuario riaffitti per il prossimo mese.

Nel caso in cui la casa non venga riaffittata, oppure se non è stata affittata precedentemente, allora ci sarà il 3.4% che la casa rimanga sfitta (dato ottenuta dal sito: <https://www.domain.com.au/research/vacancy-rates-june-2022-1149519/>) e nel caso in cui venga affittata per il prossimo mese il tipo di affitto verrà calcolato tramite le informazioni ottenute dal metodo generateTurismoAttuale.

```
private void handleEvent(Event e) {  
    this.affitti.add(e);  
    this.mesiTot++;  
  
    if(e.getType()==EventType.BUSINESS) {  
        guadagno += this.abitazione.getMensilRent();  
  
        if(e.getDurata()>0) {  
            eventoAttuale = new Event(dataAttuale, EventType.BUSINESS, e.getDurata()-1);  
            this.flAffittoSemestrale = 1; //OVVERO IN QUESTO PERIODO HO AVUTO UN AFFITTO SEMESTRALE UNICO  
            return;  
        }  
  
        if(this.flAffittoSemestrale == 1) {  
            this.flAffittoSemestrale = 0;  
            this.generateEvent();  
            return; //IN QUESTO MODO NON POSSO AVERE ASSICURATO UN BUSINESS DOPO UN AFFITTO SEMESTRALE DI BUSINESS  
        }  
  
        double random = Math.random();  
        if(random<0.5) {  
            //RIAFFITTO PER UN SINGOLO MESE  
            eventoAttuale = new Event(dataAttuale, EventType.BUSINESS,0);  
            return;  
        }  
        if(random<0.7) {  
            //RIAFFITTO PER L'INTERA STAGIONE  
  
            eventoAttuale = new Event(dataAttuale, EventType.BUSINESS, 2);  
  
            return;  
        }  
        //NON HO RIAFFITTO, QUINDI DEVO GENERARE UN NUOVO EVENTO CASUALE  
        this.generateEvent();  
        return;  
    }  
  
    if(e.getType()==EventType.HOLIDAY) {
```

```

if(e.getType()==EventType.HOLIDAY) {
    guadagno += this.abitazione.getMensilRent();
    double random = Math.random();
    if(random<0.2) {
        //RIAFFITTO PER UN SINGOLO MESE
        eventoAttuale = new Event(dataAttuale, EventType.HOLIDAY,0);
    }
    //NON HO RIAFFITTO, QUINDI DEVO GENERARE UN NUOVO EVENTO CASUALE
    this.generateEvent();
    return;
}

if(e.getType()==EventType.VISITING) {
    guadagno += this.abitazione.getMensilRent();
    double random = Math.random();
    if(random<0.1) {
        //RIAFFITTO PER UN SINGOLO MESE
        eventoAttuale = new Event(dataAttuale, EventType.VISITING,0);
    }
    //NON HO RIAFFITTO, QUINDI DEVO GENERARE UN NUOVO EVENTO CASUALE
    this.generateEvent();
    return;
}

if(e.getType()==EventType.OTHER) {
    guadagno += this.abitazione.getMensilRent();
    double random = Math.random();
    if(random<0.05) {
        //RIAFFITTO PER UN SINGOLO MESE
        eventoAttuale = new Event(dataAttuale, EventType.OTHER,0);
    }
    //NON HO RIAFFITTO, QUINDI DEVO GENERARE UN NUOVO EVENTO CASUALE
    this.generateEvent();
    return;
}

if(e.getType()==EventType.NONE) {
    this.generateEvent();
    return;
}
}

```

Figura 4.11: codice del metodo handleEvent

generateTurismoAttuale: metodo che, tramite la previsione “moving average”, riesce ad ottenere i dati del turismo attuale tramite l’analisi delle precedenti 20 stagioni (lista ottenuta inizialmente dal model, richiamando il metodo del dao getLast20TourismData). Oltre a questo, in modo da preparare la lista dei 20 dati per la prossima previsione, il metodo elimina l’elemento più vecchio dalla lista e lo rimpiazza con l’oggetto TourismData appena creato.

```

private void generateTurismoAttuale() {
    double business = 0;
    double holiday = 0;
    double visiting = 0;
    double other = 0;
    for(TourismData t : turismoPrecedente) {
        business += t.getBusiness();
        holiday += t.getHoliday();
        visiting += t.getVisiting();
        other += t.getOther();
    }

    LocalDate nuovoQuarter = turismoPrecedente.get(0).getQuarter().plusMonths(3);
    business = business/turismoPrecedente.size();
    holiday = holiday/turismoPrecedente.size();
    visiting = visiting/turismoPrecedente.size();
    other = other/turismoPrecedente.size();

    turismoAttuale = new TourismData(nuovoQuarter, business, holiday, visiting, other);
    turismoPrecedente.remove(turismoPrecedente.size()-1);
    turismoPrecedente.add(0,turismoAttuale);
    //RIMUOVO L'ELEMENTO PIU' VECCHIO DALLA LISTA, ED AGGIUNGO IL NUOVO ELEMENTO IN PRIMA POSIZIONE
}

```

Figura 4.12: codice del metodo generateTurismoAttuale.

generateEvent: metodo che viene richiamato ogni volta che bisogna generare un nuovo tipo di evento che non comprenda il riaffitto (a quello ci pensa direttamente il metodo handleEvent). Dopo aver creato un double casuale compreso tra 0 ed 1, se quel numero è minore del vacancy rate (la probabilità che la casa rimanga sfitta quel mese), allora verrà generato un evento di tipo NONE (ovvero casa non affittata). Altrimenti, ottenute le informazioni all'interno dell'oggetto turismoAttuale, vengono calcolate in percentuale le probabilità che un nuovo affitto sia di uno dei quattro tipi. Per determinare di quale tipo sia il nuovo evento verrà utilizzato lo stesso metodo con il double random utilizzato precedentemente.


```

private void generateEvent() {
    double random = Math.random();
    if(random<0.034) {
        //CASA SFITTA
        eventoAttuale = new Event(dataAttuale, EventType.NONE,0);
        return;
    }
    double somma = turismoAttuale.getBusiness() + turismoAttuale.getHoliday() + turismoAttuale.getVisiting() + turismoAttuale.getOther();
    double pBusiness = turismoAttuale.getBusiness()/somma;
    double pHoliday = turismoAttuale.getHoliday()/somma;
    double pVisiting = turismoAttuale.getVisiting()/somma;

    random = Math.random();

    if(random<pBusiness) {
        //BUSINESS
        eventoAttuale = new Event(dataAttuale, EventType.BUSINESS,1);
        return;
    }
    if(random<pBusiness + pHoliday) {
        //HOLIDAY
        eventoAttuale = new Event(dataAttuale, EventType.HOLIDAY,0);
        return;
    }
    if(random<pBusiness + pHoliday + pVisiting) {
        //VISITING
        eventoAttuale = new Event(dataAttuale, EventType.VISITING,0);
        return;
    }

    //OTHER
    eventoAttuale = new Event(dataAttuale, EventType.OTHER,0);
    return;
}

```

Figura 4.13: codice del metodo generateEvent.

Capitolo 5: Diagramma delle classi principali

5.1 Descrizione delle classi

- Main: si occupa di eseguire il software
- FXMLController: permette di gestire l'interfaccia grafica e comunica gli input utente al Model
- EntryPoint: collega il Model all'FXMLController
- DBConnect: si occupa della connessione al database
- EstateDao: contiene i metodi per richiedere informazioni dal database e salvarle in strutture dati apposite
- TestDao: classe di test per il package Dao
- Event: rappresenta l'evento di affitto mensile utilizzato nell'algoritmo di ricorsione
- Model: classe che comprende quei metodi per elaborare i dati e per collegare l'FXMLController al Dao o alle due classi contenenti algoritmi di ricorsione e simulazione
- Property: classe che rappresenta una riga nella tabella "properties" (il prezzo di affitto dell'abitazione non è presente all'interno del database, ma è stato calcolato secondo le istruzioni presenti al sito: <https://smartasset.com/mortgage/how-much-you-should-charge-for-rent>)
 - Prezzo dell'abitazione inferiore a 1.000.000\$ allora affitto = $1.1\% \times \text{prezzo}$
 - Prezzo dell'abitazione compreso tra 1.000.000\$ e 2.000.000\$ allora affitto = $1.05\% \times \text{prezzo}$
 - Prezzo dell'abitazione compreso tra 2.000.000\$ e 3.000.000\$ allora affitto = $1.00\% \times \text{prezzo}$
 - Prezzo dell'abitazione compreso tra 3.000.000\$ e 4.000.000\$ allora affitto = $0.95\% \times \text{prezzo}$
 - Prezzo dell'abitazione compreso tra 4.000.000\$ e 5.000.000\$ allora affitto = $0.90\% \times \text{prezzo}$

- Prezzo dell'abitazione compreso tra 5.000.000\$ e 7.000.000\$ allora affitto = $0.85\% \times \text{prezzo}$
- Prezzo dell'abitazione compreso tra 7.000.000\$ e 9.000.000\$ allora affitto = $0.80\% \times \text{prezzo}$

- RicorsioneMensilRent: classe contenente i metodi utilizzati per l'algoritmo di ricorsione necessario al calcolo della lista con massimo guadagno mensile
- Simulatore: classe che comprende l'algoritmo di simulazione per il calcolo del periodo di ritorno del capitale investito
- StringAndInt: classe wrapper di una stringa ed un intero, utilizzata per salvare dei risultati di alcuni metodi presenti nell'EstateDao
- TestModel: classe di test per il package Model
- TourismData: classe che rappresenta una riga della tabella "tourism" del database

Capitolo 6: Interfaccia utente e video dimostrativo

L'interfaccia utente risulta suddivisa in due parti: una prima dove l'utente può consultare informazioni ed inserire filtri per ottenere una lista raccomandata di investimenti; una seconda dove, una volta selezionata un'abitazione, verrà calcolato il tempo di ritorno dell'investimento. Per maggiori informazioni sull'utilizzo del software, è disponibile un video dimostrativo al link:

<https://youtu.be/nXlga59pYew>

The screenshot shows a web application titled "Valutazione del mercato immobiliare di Melbourne". The interface is divided into several sections:

- Header:** "Valutazione del mercato immobiliare di Melbourne"
- Instructions:** "Inserire un budget ed un obiettivo dell'analisi per ottenere una lista di abitazioni adatta alle sue esigenze nella tabella sottostante"
- Input Fields:**
 - Budget:
 - Obiettivo:
 - Quartiere:
 - Venditore:
 - Numero Stanze:
 - Prezzo massimo:
 - Posto Auto: ☐
- Buttons:** "Calcola Insieme", "Mostra analisi preliminare", "Analizza proprietà"
- Analisi preliminare delle proprietà:** A list of the most expensive neighborhoods with their postcodes:
 - Quartieri più costosi:
 - Kooyong(2185000)
 - Canterbury(2180240)
 - Middle Park(2082529)
 - Albert Park(1941355)
 - Brighton(1930158)
 - Balwyn(1869878)
 - Eaglemont(1831695)
 - Balwyn North(1793405)
 - Malvern(1764992)
- Table:** A table with columns: Suburb, Address, Rooms, Price, SellerG, PostCode, Car, RegionName. The table is currently empty, displaying "Nessun contenuto nella tabella".
- Footer:** "Dopo aver ottenuto la lista di abitazioni consigliate, selezionarne una dal menù a tendina sottostante per calcolarne il periodo di ritorno del capitale investito"

Figura 6.1: Interfaccia utente completa.

6.1 Fase iniziale: creazione del portafoglio di investimenti

In questa fase iniziale l'utente può controllare le informazioni contenute nell'area di testo a destra, comprendenti una lista di quartieri più e meno costosi, di quartieri con più abitazioni e di venditori con più offerte.

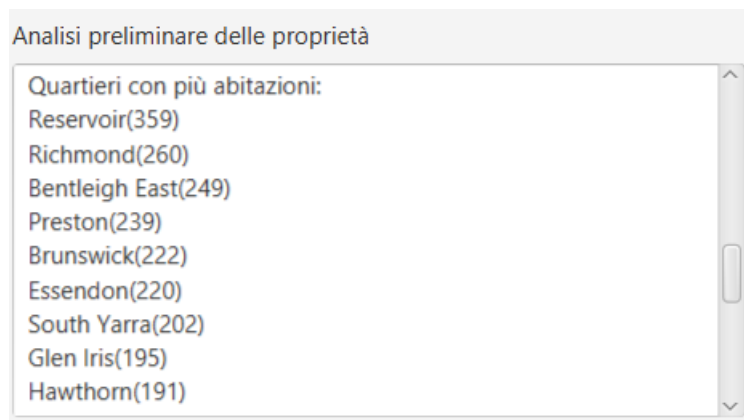


Figura 6.2: Schermata di testo contenente le informazioni preliminare fornite all'utente.

Oltre a ciò, l'utente può provare una propria lista personale di appartamenti consigliati, tramite l'inserimento di un budget massimo disponibile, di un obiettivo dell'analisi (l'utente vuole acquistare il maggior numero di case, il minimo oppure avere il massimo affitto mensile complessivo disponibile con quel budget) e di un quartiere per filtrare le case. Questi 3 parametri sono gli unici obbligatori, mentre tutti gli altri filtri presenti nell'interfaccia sono opzionali. Cliccando il pulsante "Calcola Insieme", il programma stamperà nella schermata di testo a destra informazioni generali sulla lista ottenuta (numero di appartamenti, costo complessivo ed eventualmente affitto mensile complessivo) e nella tabella sottostante la lista delle abitazioni consigliate. Nel caso in cui almeno uno dei 3 campi obbligatori specificati precedentemente non venga inserito, il programma stamperà un messaggio di errore nell'area a destra, invitando l'utente a ricontrollare i valori.

Analisi preliminare delle proprietà

Inserire il budget disponibile

Figura 6.3: Messaggio di errore visualizzato quando si prova ad iniziare l’analisi senza aver prima inserito il campo “Budget”.

Suburb	Address	Rooms	Price	SellerG	PostCode	Car	RegionName
Abbotsford	5/29 Church St	2	480000	hockingstuart	3067	1	Northern Metropolitan
Abbotsford	11/205 Gipps St	1	470000	Nelson	3067	1	Northern Metropolitan
Abbotsford	116/56 Nicholson St	1	457000	Jellis	3067	1	Northern Metropolitan
Abbotsford	7/20 Abbotsford St	1	441000	Greg	3067	1	Northern Metropolitan
Abbotsford	5/20 Abbotsford St	1	426000	Greg	3067	1	Northern Metropolitan
Abbotsford	6/241 Nicholson St	1	300000	Biggin	3067	1	Northern Metropolitan

Figura 6.4: Risultato dell’analisi.

6.2 Seconda fase: calcolo del tempo di ritorno del capitale investito

Questa seconda fase sarà disponibile soltanto dopo aver ricevuto la lista di appartamenti consigliati. Nella parte inferiore della schermata utente è presente un menù a tendina dove è possibile selezionare una delle abitazioni consigliate dal software ed un pulsante che, una volta selezionata l’abitazione, avvierà l’algoritmo di simulazione per ottenere il periodo di ritorno del capitale, che verrà mostrato nell’area di testo in basso a destra. Se il menù a tendina risulta vuoto è perché l’utente non ha ancora effettuato l’analisi nella parte precedente e nel caso in cui si provi ad iniziare il calcolo verrà visualizzato un messaggio di errore nell’area a destra.

Dopo aver ottenuto la lista di abitazioni consigliate, selezionarne una dal menù a tendina sottostante per calcolarne il periodo di ritorno del capitale investito

Selezionare un'Abitazione dall'apposita tendina

Figura 6.5: Messaggio di errore ricevuto nel caso si provi ad analizzare la proprietà senza aver prima selezionato una proprietà nell'apposito menù a tendina.

Dopo aver ottenuto la lista di abitazioni consigliate, selezionarne una dal menù a tendina sottostante per calcolarne il periodo di ritorno del capitale investito

6/241 Nicholson ...

Tempo di ritorno del capitale investito: 93 mesi
 2017-01-01 (HOLIDAY)
 2017-02-01 (HOLIDAY)
 2017-03-01 (HOLIDAY)
 2017-04-01 (NONE)
 2017-05-01 (VISITING)
 2017-06-01 (VISITING)
 2017-07-01 (BUSINESS)
 2017-08-01 (BUSINESS)
 2017-09-01 (OTHER)

Figura 6.6: Risultato del calcolo del periodo di ritorno del capitale, contenente prima il numero di mesi totali e poi, mese per mese, il motivo dell'affitto.

Capitolo 7: Risultati sperimentali

7.1 Lista di dimensioni massime non filtrata

L'obiettivo di questa analisi è di valutare le prestazioni del programma nel caso venga cercata la lista di dimensione massima possibile. Per fare ciò, sono stati inseriti come parametri il quartiere con più abitazioni (Reservoir, facilmente individuabile dai risultati dell'analisi preliminare), la funzione obiettivo "Maggior numero di abitazioni" ed un budget "infinito" (ovvero sufficiente a poter acquistare tutte le abitazioni nel quartiere). Non vengono inseriti altri parametri opzionali.

Valutazione del mercato immobiliare di Melbourne

Valutazione del mercato immobiliare di Melbourne

Inserire un budget ed un obiettivo dell'analisi per ottenere una lista di abitazioni adatta alle sue esigenze nella tabella sottostante

Budget: 500000000 Calcola Insieme

Obiettivo: Maggior numero di abitazioni

Reservoir Venditore

Numero Stanze: Prezzo massimo:

☐ Posto Auto Mostra analisi preliminare

Analisi preliminare delle proprietà

Lista composta da 359 abitazioni, per un totale di: 247713200\$

Suburb	Address	Rooms	Price	SellerG	PostCode	Car	RegionName
Reservoir	1/164 Leamington St	1	216000	Brad	3073	1	Northern Metropolitan
Reservoir	2/164 Leamington St	1	220000	Brad	3073	1	Northern Metropolitan
Reservoir	4/1067 High St	2	240000	Stockdale	3073	1	Northern Metropolitan
Reservoir	1/40 Godley St	1	260000	hockingstuart	3073	1	Northern Metropolitan
Reservoir	14/696 Plenty Rd	2	270000	hockingstuart	3073	1	Northern Metropolitan
Reservoir	1/31 Kenilworth St	1	270000	Stockdale	3073	1	Northern Metropolitan
Reservoir	8/99 Barton St	2	300000	Love	3073	1	Northern Metropolitan

Figura 7.1: risultato dell'analisi per trovare la lista più grande di abitazioni.

Come aspettato, il risultato comprende tutte le 359 abitazioni del quartiere Reservoir, per un totale di 247.713.200\$. L'analisi impiega complessivamente un tempo di 40ms per ottenere il risultato.

7.2 Lista filtrata

In questo caso vogliamo partire dal risultato precedente per osservare il comportamento del programma nel caso in cui vengano inseriti dei filtri da una lista di grandi dimensioni. Per fare ciò partiamo dai parametri inseriti nel caso “7.1 Lista massima non filtrata”, ed aggiungiamo che si cercano soltanto abitazioni con un numero di stanze maggiore o uguale a 2, con posto auto disponibile, prezzo massimo di 1200000 (abbastanza da escludere all’incirca 10 abitazioni) e che sia venduta da “Bary” (con il maggior numero di offerte disponibili nel quartiere).

The screenshot shows a web application titled "Valutazione del mercato immobiliare di Melbourne". It features a search form with the following fields and values:

- Budget: 500000000
- Obiettivo: Maggior numero di abitazioni
- Reservoir: (selected)
- Barry: (selected)
- Rooms: 2
- Price: 1200000
- Posto Auto: ☒

Buttons include "Calcola Insieme" and "Mostra analisi preliminare". A summary box on the right states: "Lista composta da 86 abitazioni, per un totale di: 60105250\$". Below the form is a table with 8 columns: Suburb, Address, Rooms, Price, SellerG, PostCode, Car, and RegionName. The table lists 7 properties, all from "Barry" in the "Northern Metropolitan" region.

Suburb	Address	Rooms	Price	SellerG	PostCode	Car	RegionName
Reservoir	6/833 High St	3	301000	Barry	3073	1	Northern Metropolitan
Reservoir	4/62 St Vigeons Rd	2	350000	Barry	3073	1	Northern Metropolitan
Reservoir	2/20 Crookston Rd	2	352000	Barry	3073	1	Northern Metropolitan
Reservoir	2/7 Mattea Ct	2	360000	Barry	3073	1	Northern Metropolitan
Reservoir	197 Albert St	2	365000	Barry	3073	1	Northern Metropolitan
Reservoir	3/7 Mattea Ct	2	380000	Barry	3073	1	Northern Metropolitan
Reservoir	2/33 Ashley St	2	380000	Barry	3073	1	Northern Metropolitan

Figura 7.2: risultato dell’analisi per trovare una lista di quartiere tali da soddisfare tutti i parametri opzionali e che sia di massima dimensione possibile.

Per rispettare i nuovi vincoli inseriti, il numero di abitazioni trovate risulta inferiore rispetto a quello del caso precedente, con soltanto 86 abitazioni selezionate. Il tempo impiegato dall’analisi è di 24ms, inferiore perciò al tempo impiegato nel caso precedente dove non venivano utilizzati i parametri opzionali.

7.3 Ricorsione con lista di piccole dimensioni

L'obiettivo di questa analisi e della successiva è invece quello di testare l'algoritmo di ricorsione, richiamabile avviando un'analisi con la funzione obiettivo "Maggior Guadagno mensile". In questo primo caso prendiamo come parametro della ricorsione una lista di abitazioni piccola, selezionando come quartiere Werribee, con 50 abitazioni, e come budget 2000000.

Valutazione del mercato immobiliare di Melbourne

Inserire un budget ed un obiettivo dell'analisi per ottenere una lista di abitazioni adatta alle sue esigenze nella tabella sottostante

Budget:

Obiettivo:

Suburb: Venditore:

Numero Stanze: Prezzo massimo:

☐ Posto Auto

Analisi preliminare delle proprietà

Lista composta da 3 abitazioni, per un totale di: 2000000\$
Guadagno complessivo mensile: 22000\$

Suburb	Address	Rooms	Price	SellerG	PostCode	Car	RegionName
Werribee	43 Stawell St	3	830000	YPA	3030	2	Western Metropolitan
Werribee	17 Timbarra Dr	4	665000	hockingstuart	3030	2	Western Metropolitan
Werribee	6 Cassowary Av	3	505000	Greg	3030	4	Western Metropolitan

Figura 7.3: risultato dell'analisi per trovare la lista con maggior guadagno mensile nel quartiere di Werribee con un budget massimo.

L'analisi restituisce una lista di 3 abitazioni, con prezzo complessivo di 2000000\$ e a causa dell'utilizzo dell'algoritmo ricorsivo il software impiega un tempo maggiore rispetto ai casi precedenti per ottenere il risultato, ovvero 1s.

7.4 Ricorsione con lista di grandi dimensioni

Per questa analisi è stato scelto un quartiere con un numero maggiore di abitazioni, ovvero Coburg con 190 proprietà. L'obiettivo dell'analisi rimane sempre "Maggior guadagno mensile", in modo da richiamare l'algoritmo ricorsivo, con un budget di ancora 2000000.

Valutazione del mercato immobiliare di Melbourne

Inserire un budget ed un obiettivo dell'analisi per ottenere una lista di abitazioni adatta alle sue esigenze nella tabella sottostante

Budget:

Obiettivo:

☐ Posto Auto

Analisi preliminare delle proprietà

Lista composta da 2 abitazioni, per un totale di: 2000000\$
Guadagno complessivo mensile: 22000\$

Suburb	Address	Rooms	Price	SellerG	PostCode	Car	RegionName
Coburg	13 Shackell St	4	1000000	Brad	3058	1	Northern Metropolitan
Coburg	4 Miller St	3	1000000	Nelson	3058	1	Northern Metropolitan

Figura 7.4: risultato dell'analisi con algoritmo ricorsivo, per trovare la lista di abitazioni con maggior guadagno mensile nel quartiere di Coburg e con un determinato budget massimo.

7.5 Algoritmo di simulazione

Per testare il comportamento dell'algoritmo di simulazione sono state utilizzate varie abitazioni del data-set, per avere a disposizione diversi risultati.

Dopo aver ottenuto la lista di abitazioni consigliate, selezionarne una dal menù a tendina sottostante per calcolarne il periodo di ritorno del capitale investito

6/241 Nicholson ... Analizza proprietà

Tempo di ritorno del capitale investito: 92 mesi

- 2017-01-01 (VISITING)
- 2017-02-01 (BUSINESS)
- 2017-03-01 (BUSINESS)
- 2017-04-01 (HOLIDAY)
- 2017-05-01 (VISITING)
- 2017-06-01 (VISITING)
- 2017-07-01 (VISITING)
- 2017-08-01 (VISITING)
- 2017-09-01 (HOLIDAY)

Figura 7.5: soluzione dell’algoritmo di simulazione per l’abitazione in via “6/241 Nicholson St.”

Dopo aver ottenuto la lista di abitazioni consigliate, selezionarne una dal menù a tendina sottostante per calcolarne il periodo di ritorno del capitale investito

5/20 Abbotsford ... Analizza proprietà

Tempo di ritorno del capitale investito: 93 mesi

- 2017-01-01 (BUSINESS)
- 2017-02-01 (BUSINESS)
- 2017-03-01 (BUSINESS)
- 2017-04-01 (BUSINESS)
- 2017-05-01 (HOLIDAY)
- 2017-06-01 (VISITING)
- 2017-07-01 (HOLIDAY)
- 2017-08-01 (BUSINESS)
- 2017-09-01 (BUSINESS)

Figura 7.6: soluzione dell’algoritmo di simulazione per l’abitazione in via “5/20 Abbotsford St.”

Dopo aver ottenuto la lista di abitazioni consigliate, selezionarne una dal menù a tendina sottostante per calcolarne il periodo di ritorno del capitale investito

11/205 Gipps St (...) Analizza proprietà

Tempo di ritorno del capitale investito: 93 mesi

- 2017-01-01 (BUSINESS)
- 2017-02-01 (BUSINESS)
- 2017-03-01 (OTHER)
- 2017-04-01 (HOLIDAY)
- 2017-05-01 (BUSINESS)
- 2017-06-01 (BUSINESS)
- 2017-07-01 (VISITING)
- 2017-08-01 (BUSINESS)
- 2017-09-01 (BUSINESS)

Figura 7.7: soluzione dell’algoritmo di simulazione per l’abitazione in via “11/205 Gipps St.”

Dopo aver ottenuto la lista di abitazioni consigliate, selezionarne una dal menù a tendina sottostante per calcolarne il periodo di ritorno del capitale investito

21/659 Victoria St... Analizza proprietà

Tempo di ritorno del capitale investito: 95 mesi

- 2017-01-01 (VISITING)
- 2017-02-01 (BUSINESS)
- 2017-03-01 (BUSINESS)
- 2017-04-01 (HOLIDAY)
- 2017-05-01 (HOLIDAY)
- 2017-06-01 (VISITING)
- 2017-07-01 (VISITING)
- 2017-08-01 (BUSINESS)
- 2017-09-01 (BUSINESS)

Figura 7.8: soluzione dell'algoritmo di simulazione per l'abitazione in via "21/659 Victoria St."

Le 4 analisi impiegano rispettivamente 7ms, 7ms, 9ms ed 8ms, con tempi nettamente inferiori rispetto alle analisi che richiedono l'algoritmo ricorsivo, ma anche con tempi inferiori alle due analisi svolte inizialmente.

Capitolo 8: Valutazione dei risultati ottenuti

Osservando i risultati precedenti, un dato che può sembrare inizialmente strana è il tempo impiegato dalle prime due analisi, con il tempo dell'analisi con filtri inferiore al tempo di quella senza. Per spiegare ciò, deve essere considerato che la principale componente del tempo complessivo dell'analisi è il tempo di apertura della connessione con il database e di esecuzione della query, entrambi presenti ed uguali in entrambi i casi. Il minore impiego di tempo della seconda operazione è spiegato da una presenza di una lista più corta ricevuta dal Model, con conseguenti minori operazioni di elaborazione.

Spostandoci ai dati ottenuti tramite l'algoritmo ricorsivo, si può notare che il tempo di esecuzione aumenta con l'aumento della dimensione della lista da considerare. Inoltre, una particolarità che accomuna entrambe le soluzioni è il costo complessivo delle due liste, che in entrambi i casi corrispondono al budget inizialmente inserito. Perciò un metodo valido per poter risolvere liste particolarmente grandi in tempi ragionevoli sarebbe quello di approssimare il prezzo di affitto mensile all' 1% del costo dell'abitazione per ogni abitazione, ed andare a calcolare semplicemente la lista di costo maggiore possibile con i parametri impostati, per ottenere la lista risultato.

Valutando infine i risultati dell'algoritmo di simulazione, è osservabile che ogni valutazione impiega all'incirca lo stesso tempo di esecuzione di 8ms e che il tempo di ritorno del capitale investito è circa di 93 mesi. Ciò è spiegabile poiché l'algoritmo di simulazione dipende soltanto dal rapporto prezzo di affitto mensile sul costo totale dell'abitazione e da dati esterni (quelli sul turismo) che dipendono soltanto da valori fissi nel database e dal caso. Perciò, approssimando come fatto precedentemente il prezzo di affitto mensile all' 1% del costo dell'abitazione, il tempo di ritorno del capitale risulterebbe essere di circa 100 mesi.

Capitolo 9: Conclusioni

Il software è in grado di prestarsi come un valido supporto alle decisioni per aziende del settore e privati che vogliono intraprendere questo tipo di investimento. Ciò avviene tramite la visualizzazione di un quadro del mercato di Melbourne, la creazione di un portafoglio immobiliare in base alle esigenze dell'utente ed il calcolo del tempo di ritorno del capitale investito, attraverso l'analisi del turismo precedente e la previsione del turismo successivo.

La criticità del software rilevata riguarda l'elevato tempo di esecuzione della richiesta della lista quando si chiede il maggior ritorno mensile nel caso si abbia un budget considerevole. Un modo per ridurre questo tempo sarebbe, come spiegato in precedenza, approssimare il rapporto affitto mensile su costo dell'abitazione all' 1%, in modo da poter così trasformare l'algoritmo utilizzato in un semplice calcolo della lista con prezzo massimo rispetto al budget inserito. Questo consentirebbe di semplificare anche l'algoritmo di simulazione successivo, rendendo il tempo di ritorno del capitale dipendente soltanto dal caso e non più anche dal prezzo dell'abitazione.

L'obiettivo del software rimane comunque compiuto, ponendosi come strumento per ridurre il rischio agli investitori del settore e facilitando l'ingresso a nuovi investitori, attraverso la facile accessibilità ai dati riassuntivi del mercato.

Capitolo 10: Licenza



Quest'opera è distribuita con licenza Creative Commons Attribuzione - Non commerciale -
Condividi allo stesso modo 4.0 Internazionale.

Copia della licenza consultabile al sito web:

<https://creativecommons.org/licenses/by-nc-sa/4.0/>