



UNIVERSITÀ DEGLI STUDI GUGLIELMO MARCONI

**DIPARTIMENTO DI SCIENZE INGEGNERISTICHE
CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA**

DIGITAL TRANSFORMATION E DATA SCIENCE

“Progettazione e sviluppo di un sistema predittivo containerizzato per la stima dei prezzi immobiliari basato su tecniche di Machine Learning”

Relatore:

Prof. NICOLA DALL'ORA

Candidato:

GAETANO DI GRAZIA
Matr. 0064917

Alla mia famiglia, a tutte le persone che mi
sono state accanto.
Agli amici che ho riscoperto.
Alla vita.

*“La felicità si può trovare anche negli
attimi più tenebrosi, se solo qualcuno si
ricorda di accendere la luce.”*

J.K Rowling

Elenco delle Figure.....	
Lista delle Tabelle.....	
Capitolo 1 - Introduzione.....	1
1.1 Contesto e motivazioni.....	1
1.2 Obiettivo della tesi.....	1
1.3 Struttura della tesi.....	2
Capitolo 2 - Fondamenti teorici e contesto applicativo.....	3
2.1 Digitization, Digitalization e Digital Transformation.....	3
2.2 Il ruolo della Data Science nella trasformazione digitale.....	3
2.3 Il contesto immobiliare e le potenzialità dei dati.....	4
2.4 Verso modelli predittivi centrati sull'uomo (HCAI).....	5
2.5 Natural Language Processing e pricing immobiliare.....	6
2.6 Riferimenti nella letteratura scientifica.....	6
2.7 Conclusione del capitolo.....	7
Capitolo 3 - Modelli e approcci predittivi.....	8
3.1 Introduzione ai modelli predittivi.....	8
3.2 Feature engineering e multicollinearità.....	8
3.3 Log-trasformazione della variabile target.....	9
3.4 Modelli testati e confrontati.....	9
3.5 Confronto tra modelli.....	9
3.6 Feature importance e interpretabilità.....	10
3.7 Conclusione del capitolo.....	10
Capitolo 4 - Preprocessing e analisi del dataset.....	11
4.1 Descrizione del dataset.....	11
4.2 Analisi esplorativa dei dati (EDA).....	12
4.3 Trasformazioni logaritmiche e riduzione della skewness.....	12
4.4 Analisi delle correlazioni.....	15
4.5 Identificazione e rimozione degli outlier.....	17
4.6 Trattamento dei valori mancanti.....	19
4.7 Trasformazione delle variabili.....	21
4.8 Feature engineering: nuove variabili e boolean logici.....	22
4.9 Gestione della multicollinearità.....	24

4.10 Conclusione del capitolo.....	25
Capitolo 5 - Modellazione predittiva.....	27
5.1 Obiettivi della modellazione.....	27
5.2 Preparazione del dataset per il training.....	28
5.3 Modelli di regressione implementati.....	30
5.4 Confronto tra i modelli.....	32
5.5 Selezione del modello finale.....	34
5.6 Messa in produzione del modello e infrastruttura applicativa.....	36
5.7 Conclusione del capitolo.....	38
Capitolo 6 - Progettazione e sviluppo della parte predittiva.....	40
6.1 Obiettivi della dashboard.....	40
6.2 Requisiti funzionali e tecnici.....	41
6.3 Progettazione dell'interfaccia utente (UI/UX).....	42
6.4 Gestione dell'autenticazione e della sicurezza.....	45
6.5 Implementazione tecnica della dashboard.....	47
6.6 Architettura dell'integrazione.....	50
6.9 Conclusione del capitolo.....	63
Capitolo 7 - Sviluppi futuri e possibili estensioni.....	65
7.1 Estensione delle fonti dati.....	65
7.2 Introduzione dell'NLP per analisi testuale.....	66
7.3 Integrazione della explainable AI (XAI).....	67
7.4 Ottimizzazione della UI/UX.....	68
7.5 Espansione cloud e API pubbliche.....	68
7.6 Conclusione del capitolo.....	71
Capitolo 8 - Conclusioni.....	73
8.1 Riepilogo del percorso.....	73
8.2 Risultati principali.....	73
8.3 Impatto applicativo e potenzialità.....	74
8.4 Limiti del lavoro.....	74
8.5 Considerazioni finali.....	75
Bibliografia.....	77
Sitografia.....	77
Appendice - Approfondimenti Tecnici e Metodologici.....	79
A.1 Sicurezza dell'autenticazione: OAuth2, JWT, Token e Bearer Authentication.	79

A.2 Messaging asincrono: Apache Kafka e Zookeeper.....	80
A.3 Architettura dei container Docker.....	81
A.4 Codice di logging.....	82
A.5 Repository del progetto.....	82
A.6 Flusso completo di autenticazione OAuth2.....	83
A.7 Esempio di messaggio JSON inviato su Kafka.....	84
A.8 File docker-compose.yml.....	85
A.9 Specifica OpenAPI per la API di predizione.....	89
A.10 Configurazione di Keycloak nel sistema di autenticazione.....	91
A.11 Considerazioni sull'uso di Docker Compose e limiti in ambiente di produzione	92
A.12 Tecniche di validazione incrociata (Cross Validation).....	93
A.13 Metriche di valutazione dei modelli predittivi.....	94
A.14 Introduzione alla Explainable AI (XAI) e valori SHAP.....	95
A.14 Principi di Ergonomia Cognitiva nella progettazione della dashboard.....	96
Riferimenti – Appendice A.....	98

Elenco delle Figure

Figura	Titolo	Capitolo
Figura 4.1	Distribuzione reale della variabile SalePrice	Capitolo 4
Figura 4.2	Distribuzione logaritmica della variabile SalePrice	Capitolo 4
Figura 4.3	Matrice di correlazione tra le principali variabili	Capitolo 4
Figura 4.4	Relazione tra superficie abitabile e prezzo di vendita	Capitolo 4
Figura 4.5	Relazione tra area abitativa e prezzo di vendita (scatterplot)	Capitolo 4
Figura 4.6	Rimozione degli outlier basata sulla superficie GrLivArea (estratto di codice)	Capitolo 4
Figura 4.7	Percentuale di valori mancanti per ciascuna variabile	Capitolo 4
Figura 5.1	Selezione delle variabili predittive per il training (estratto di codice)	Capitolo 5
Figura 5.2	Encoding delle variabili categoriche tramite StringIndexer (estratto di codice)	Capitolo 5
Figura 6.1	Wireframe della dashboard Streamlit progettata	Capitolo 6
Figura 6.2	Esempio di sezione collassabile nella dashboard Streamlit	Capitolo 6
Figura 6.3	Richiesta di predizione e visualizzazione del prezzo	Capitolo 6

Figura 6.4	Diagramma di deployment	Capitolo 6
Figura 6.5	Esempio di flusso di comunicazione tra Streamlit, FastAPI, Keycloak e Kafka	Capitolo 6
Figura 6.6	Esempio di Kanban board utilizzata per la gestione delle attività	Capitolo 6
Figura 6.7	Processo GitFlow	Capitolo 6
Figura 6.8	Esempio di branching model con branch main, develop e feature branches	Capitolo 5
Figura 6.9	Flusso di interazione utente nella dashboard	Capitolo 6
Figura 6.10a	Stima iniziale senza caminetto	Capitolo 6
Figura 6.10b	Stima aggiornata con caminetto di alta qualità	Capitolo 6
Figura 6.11	Activity Diagram: flusso operativo di utilizzo della dashboard	Capitolo 6
Figura A.4.1	Estratto di codice per il sistema di logging delle richieste e predizioni	Appendice
Figura A.6.1	Sequence Diagram del flusso di autenticazione e autorizzazione OAuth2	Appendice
Figura A.7.1	Esempio di struttura del messaggio JSON inviato su Kafka	Appendice
Figura A.8.1	Definizione della rete Docker "immobiliare_net" con driver bridge	Appendice
Figura A.8.2	Definizione del servizio Docker "dashboard" con binding locale sulla porta 8501	Appendice

Figura A.8.3	Definizione del servizio Docker "model-api" per il backend FastAPI	Appendice
Figura A.8.4	Definizione del servizio Docker "keycloak" per la gestione dell'autenticazione OAuth2	Appendice
Figura A.8.5	Definizione del servizio Docker "zookeeper" per la gestione della coordinazione di Kafka	Appendice
Figura A.8.6	Definizione del servizio Docker "kafka" per la gestione della messaggistica asincrona	Appendice
Figura A.8.7	Definizione del servizio Docker "kafdrop" per il monitoraggio dei messaggi Kafka	Appendice
Figura A.9.1	Visualizzazione dei parametri richiesti per la predizione	Appendice
Figura A.9.2	Visualizzazione dei codici di risposta e del formato di output	Appendice

Lista delle Tabelle

Tabella	Titolo	Capitolo
Tabella 3.1	Confronto preliminare tra modelli predittivi: regressione lineare, Random Forest e XGBoost (fase esplorativa)	Capitolo 3
Tabella 5.1	Valutazione finale dei modelli predittivi su dataset preprocessato: Random Forest, regressione lineare e XGBoost	Capitolo 5
Tabella 7.1	Sviluppi futuri previsti: aree di intervento, descrizione e impatto atteso	Capitolo 7

Capitolo 1 - Introduzione

1.1 Contesto e motivazioni

Negli ultimi anni, la Data Science ha rivoluzionato numerosi ambiti, tra cui anche il mercato immobiliare, storicamente ancorato a metodi di valutazione empirici e soggettivi. L'uso di algoritmi di apprendimento automatico e strumenti di analisi predittiva consente oggi una stima più oggettiva, scalabile e replicabile del valore degli immobili.

Questa tesi rappresenta l'evoluzione naturale di un percorso accademico e personale cominciato con lo studio dei mercati finanziari. Nel mio elaborato di laurea triennale, discusso presso l'Università degli Studi di Palermo nell'A.A. 2020-2021, ho affrontato tematiche quali le serie temporali, i modelli autoregressivi, la volatilità e il sentiment analysis attraverso l'NLP, evidenziando l'intersezione tra economia, statistica e informatica. L'attuale lavoro ne raccoglie l'eredità, spostando l'attenzione dal mercato azionario a quello immobiliare.

1.2 Obiettivo della tesi

L'obiettivo principale di questa tesi è sviluppare un sistema predittivo completo per la stima del prezzo di un'abitazione, combinando modelli di machine learning con un'interfaccia interattiva a supporto delle decisioni. Il progetto ha un'impostazione end-to-end, che parte dall'analisi esplorativa dei dati e dall'ingegnerizzazione delle variabili, fino all'implementazione finale di una dashboard accessibile, costruita secondo alcuni dei cardini dell'information retrieval e della Human-centered Artificial Intelligence (HCAI).

A differenza di approcci meramente teorici o accademici, questa tesi propone una soluzione concreta e replicabile, con un'architettura containerizzata che rende il sistema portatile e indipendente dall'ambiente di esecuzione. Pur avendo considerato anche approcci NLP per l'estrazione automatica delle caratteristiche da testo, tali metodi sono stati ritenuti fuori ambito per gli scopi principali di questo lavoro e rinviati a futuri sviluppi.

1.3 Struttura della tesi

Il lavoro si articola nei seguenti capitoli:

- Il capitolo 2 introduce i concetti chiave di Data Science, digitalizzazione, HCAI e trasformazione digitale del settore immobiliare.
- Il capitolo 3 descrive gli algoritmi di regressione impiegati nel progetto e il framework sperimentale.
- Il capitolo 4 affronta il preprocessing, la gestione dei valori mancanti e il feature engineering.
- Il capitolo 5 valuta e confronta i modelli predittivi implementati.
- Il capitolo 6 mostra l'ottimizzazione e distribuzione del sistema su architettura distribuita tramite Apache Spark.
- Il capitolo 7 presenta il design concettuale e pratico della dashboard interattiva.
- Il capitolo 8 raccoglie le conclusioni e propone linee guida per lo sviluppo futuro del sistema, includendo l'eventuale integrazione di NLP per l'estrazione automatica di feature da testo descrittivo.

Capitolo 2 - Fondamenti teorici e contesto applicativo

2.1 Digitization, Digitalization e Digital Transformation

La distinzione tra Digitization, Digitalization e Digital Transformation, è essenziale per comprendere l'evoluzione delle tecnologie digitali nel contesto economico e sociale contemporaneo.

La Digitization si riferisce al processo di conversione dell'informazione da un formato analogico a uno digitale. È un processo tecnico, di base, che rende i dati accessibili e trattabili dai sistemi informatici. Ad esempio, la scansione di un contratto immobiliare cartaceo rappresenta un processo di digitalizzazione.

La Digitalization invece, implica l'adozione di tecnologie digitali per migliorare o automatizzare processi aziendali esistenti. In ambito immobiliare, si manifesta con l'utilizzo di sistemi informativi per la gestione dei beni, la comunicazione con i clienti, o l'analisi dei dati immobiliari raccolti.

La Digital Transformation rappresenta un livello superiore: non si limita all'applicazione delle tecnologie, ma implica una ristrutturazione strategica dell'organizzazione. Trasforma i modelli di business, i comportamenti degli attori coinvolti e le relazioni di valore. Nella filiera immobiliare, ciò può includere modelli predittivi per il pricing, strumenti di realtà aumentata per la visualizzazione degli ambienti, o piattaforme data-driven per la valutazione automatizzata degli immobili.

2.2 Il ruolo della Data Science nella trasformazione digitale

La Data Science costituisce oggi uno degli strumenti cardine della trasformazione digitale. Essa integra competenze statistiche, informatiche e di dominio per estrarre conoscenza dai dati, orientare le decisioni e creare valore.

Nel settore immobiliare, la Data Science consente di superare le tradizionali logiche esperienziali o meramente comparative, introducendo approcci oggettivi, trasparenti e scalabili. I dati strutturati (es. superficie, numero di stanze, anno di costruzione) e semi-strutturati (es. descrizioni testuali, immagini) possono essere analizzati per

costruire modelli di stima, valutazione del rischio o ottimizzazione degli investimenti.

L'adozione di tecniche di apprendimento automatico, come la regressione, la random forest o l'XGBoost, consente di costruire modelli predittivi adattivi e generalizzabili, capaci di cogliere pattern nascosti nei dati. Questi modelli trovano applicazione in numerose fasi: dall'analisi comparativa dei prezzi, alla previsione della domanda, fino alla determinazione del valore di mercato di un immobile in tempo reale.

2.3 Il contesto immobiliare e le potenzialità dei dati

Il mercato immobiliare è un ambito particolarmente ricco di dati, ma storicamente sotto-sfruttato dal punto di vista analitico. Le fonti informative comprendono:

- dati catastali e anagrafici;
- anagrafiche dei comuni e informazioni urbanistiche;
- storici di transazioni immobiliari;
- annunci di vendita online;
- descrizioni testuali e immagini;
- indicatori socioeconomici e ambientali.

L'eterogeneità e la granularità di queste fonti aprono alla possibilità di costruire modelli estremamente precisi a condizione che questi vengano opportunamente preprocessati, normalizzati e integrati.

Inoltre, la recente diffusione di portali immobiliari ha accelerato la disponibilità di dati aggiornati e geolocalizzati, offrendo nuove opportunità per l'analisi automatica e in tempo reale delle tendenze di mercato.

Trasformazione digitale nel settore immobiliare: scenari reali

Negli ultimi anni, diversi attori del mercato immobiliare hanno adottato modelli di business data-driven. Il caso più noto a livello internazionale è quello di Zillow, piattaforma statunitense che ha introdotto il cosiddetto “Zestimate”: un algoritmo proprietario per la valutazione degli immobili basato su tecniche avanzate di machine learning e un vasto patrimonio informativo.

In Europa, portali come Idealista e Immobiliare.it hanno cominciato a integrare strumenti analitici e grafici di comparazione, mentre nel Regno Unito, Zoopla offre stime immobiliari basate su storici transazionali e caratteristiche di zona.

In Italia, sebbene la trasformazione digitale sia più lenta, si assiste a una progressiva apertura verso strumenti automatizzati, grazie anche alla disponibilità di dati georeferenziati, open data catastali e banche dati comunali.

Evoluzione normativa e fonti pubbliche

Un ruolo chiave nella digitalizzazione del comparto immobiliare è svolto dall'evoluzione normativa. L'Agenzia delle Entrate ha avviato da anni il processo di digitalizzazione del catasto, rendendo accessibili molte informazioni tramite il portale Sister.

Anche l'Istituto Nazionale di Statistica (ISTAT) e i singoli comuni mettono a disposizione open data riguardanti superfici, categorie catastali, classe energetica e valori medi di compravendita. Tali risorse, se integrate correttamente, possono potenziare l'accuratezza dei modelli predittivi.

2.4 Verso modelli predittivi centrati sull'uomo (HCAI)

L'evoluzione tecnologica deve sempre tenere conto del ruolo centrale dell'uomo nei processi decisionali. È in questo contesto che si inserisce il paradigma della (HCAI), che pone l'accento sulla trasparenza, l'interpretabilità e la responsabilità dei sistemi di intelligenza artificiale.

Nel nostro progetto, i modelli predittivi sono pensati come strumenti di supporto alle decisioni, non sostituti dell'expertise umana. La dashboard sviluppata consente agli utenti di comprendere le variabili che influenzano il prezzo stimato e di sperimentare variazioni controllate sui parametri in input, mantenendo il controllo sull'output generato.

L'adozione di tecniche di explainable AI (XAI), come l'analisi SHAP (SHapley Additive exPlanations), è uno dei passi chiave per garantire che il sistema non solo fornisca risposte corrette ma che sia anche in grado di motivare le proprie stime, facilitando la fiducia dell'utente.

Linee guida ufficiali e principi etici

L'importanza dell' AI trasparente e responsabile è oggi riconosciuta a livello normativo. La Commissione Europea ha proposto nel 2021 il AI Act, che identifica livelli di rischio e stabilisce requisiti per la progettazione e l'utilizzo di sistemi di intelligenza artificiale. In particolare, per applicazioni ad alto impatto decisionale come quelle in ambito finanziario e immobiliare, si richiede:

- Tracciabilità dei dati e dei modelli;
- Interpretabilità dei risultati;
- Supervisione umana;
- Misure di mitigazione del rischio.

Nel nostro progetto, pur trattandosi di un sistema a basso rischio, si è scelto di adottare un'impostazione HCAI, orientata alla trasparenza e alla centralità dell'utente.

2.5 Natural Language Processing e pricing immobiliare

Il Natural Language Processing (NLP) rappresenta una delle frontiere più promettenti per il trattamento dei dati testuali in ambito immobiliare. Gli annunci di vendita, le descrizioni degli immobili e le recensioni dei clienti costituiscono fonti di informazione preziosa e spesso sottovalutata.

In una prima fase progettuale, si è considerata l'integrazione di tecniche NLP per estrarre automaticamente dalle descrizioni testuali le caratteristiche strutturali degli immobili, come il numero di stanze, la presenza di un garage o l'anno di ristrutturazione. Tuttavia, per ragioni di focus e necessità di enormi dataset di dati volti all'addestramento di algoritmi per il riconoscimento automatico dei parametri, tale integrazione è stata rimandata a futuri sviluppi. Resta comunque un'area di grande interesse, che potrà valorizzare ulteriormente i sistemi di valutazione automatica rendendoli ancora più flessibili e accessibili.

2.6 Riferimenti nella letteratura scientifica

L'uso di tecniche di Data Science per la valutazione degli immobili è ampiamente documentato nella letteratura. Alcuni riferimenti rilevanti includono:

- Kumar et al. (2020), A machine learning approach to predict house prices in real estate industry;
- Antipov & Pokryshevskaya (2012), Mass appraisal of residential apartments: An application of random forest for valuation and forecasting;
- Del Giudice et al. (2017), Big Data in Real Estate: From manual to automated valuation;
- Brownlee (2018), Machine Learning Mastery with Python;
- Géron (2019), Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow

Questi studi dimostrano come l'adozione di modelli predittivi consenta una valutazione più efficiente, replicabile e trasparente degli immobili, con margini di errore ridotti rispetto agli approcci tradizionali.

2.7 Conclusione del capitolo

In questo capitolo si è delineato il quadro teorico e tecnologico entro cui si inserisce il presente lavoro. La digital transformation nel settore immobiliare rappresenta un'opportunità concreta per innovare i processi decisionali e la Data Science ne costituisce il motore trainante. Il progetto si propone di costruire un sistema predittivo che non solo stimi il prezzo di un immobile ma lo faccia in modo trasparente, interpretabile e centrato sull'utente. I capitoli successivi illustreranno come ciò sia stato realizzato, partendo dalla selezione dei modelli predittivi più adatti.

Capitolo 3 - Modelli e approcci predittivi

3.1 Introduzione ai modelli predittivi

La previsione del prezzo degli immobili rappresenta una classica applicazione della regressione supervisionata, una tecnica dell'apprendimento automatico in cui l'obiettivo è stimare una variabile numerica continua (in questo caso, il prezzo di vendita) a partire da un insieme di variabili esplicative. A differenza di metodi di classificazione, che prevedono categorie discrete (es. tipo di abitazione o zona), la regressione lavora su output quantitativi e cerca di modellare relazioni matematiche tra le feature e la variabile target. In questo progetto, si è scelto un approccio end-to-end basato sull'ingegnerizzazione delle feature, sull'analisi delle distribuzioni e sulla selezione e comparazione di modelli predittivi tramite valutazione su dati reali.

3.2 Feature engineering e multicollinearità

Prima della fase di modellazione, è stato condotto un importante lavoro di feature engineering, volto a migliorare la qualità informativa delle variabili. Sono state create nuove feature derivate da combinazioni significative, tra cui:

- Feature1 e Feature2: variabili sintetiche derivate da relazioni tra superficie, anno e altri parametri;
- TotalSF: somma delle superfici abitative e accessorie;
- Overall_GrLiv_Garage_Interaction: interazione tra qualità complessiva, superficie abitabile e garage.

Inoltre, sono state introdotte variabili binarie che identificano la presenza o l'assenza di determinate caratteristiche come: hasGarage, hasFireplace, hasPool, has2ndfloor e hasBsmt.

Durante la fase di esplorazione è emerso che la variabile SalePrice era asimmetrica a destra (right-skewed), ovvero presentava una coda lunga verso i valori più alti. Questo tipo di distribuzione può influenzare negativamente le prestazioni dei modelli di regressione, in particolare quelli che assumono una distribuzione normale (gaussiana) del target, come la regressione lineare. Per ovviare a questo problema, è stata applicata una trasformazione logaritmica alla variabile target, al fine di ridurre la skewness e migliorare la linearità delle relazioni con le variabili esplicative.

3.3 Log-trasformazione della variabile target

Durante la fase di esplorazione è emerso che la variabile SalePrice era, ovvero presentava una coda lunga a destra. Questo tipo di distribuzione può influenzare negativamente le prestazioni dei modelli di regressione, che spesso assumono una distribuzione gaussiana del target.

Per correggere questa distorsione, è stata applicata una trasformazione logaritmica che ha migliorato significativamente la simmetria della distribuzione, come evidenziato dal confronto dei grafici che vedremo in dettaglio nel prossimo capitolo.

3.4 Modelli testati e confrontati

Regressione lineare

La regressione lineare è stata utilizzata come baseline. Nonostante l'elevata interpretabilità, il modello ha mostrato una limitata capacità di adattamento alla complessità dei dati ($R^2 \approx 67\%$).

Random Forest

La Random Forest si è rivelata molto più adatta a modellare relazioni non lineari e interazioni tra variabili. L'accuratezza predittiva è aumentata significativamente ($R^2 \approx 87\%$). È stato impiegato anche un CrossValidator a 5 fold per selezionare automaticamente i migliori iperparametri.

XGBoost (fase esplorativa)

Il modello XGBoost è stato sperimentato in locale ma non integrato nella pipeline finale in Spark per motivi di compatibilità. Ha comunque mostrato ottime prestazioni.

3.5 Confronto tra modelli

La seguente tabella sintetizza i principali criteri di valutazione:

Modello	Accuratezza (R^2)	Interpretabilità	Overfitting	Velocità
Regressione Lineare	67%	Alta	Bassa	Alta
Random Forest	87%	Media	Bassa	Media
XGBoost (locale)	89%	Bassa	Bassa	Bassa

Tabella 3.1 — Confronto preliminare tra modelli predittivi: regressione lineare, Random Forest e XGBoost (fase esplorativa)

Il confronto tra modelli verrà approfondito nel capitolo 5.

3.6 Feature importance e interpretabilità

Nel presente lavoro non è stata implementata esplicitamente l'estrazione delle feature più influenti, ma i modelli ad albero come la Random Forest offrono tale possibilità tramite l'attributo `featureImportances`. Questo vettore restituisce l'importanza normalizzata di ciascuna variabile nel determinare la predizione del modello, ed è disponibile nell'oggetto `RandomForestRegressorModel` in PySpark. Il suo utilizzo permette di identificare le variabili che maggiormente contribuiscono alle decisioni del modello, favorendo una prima forma di interpretabilità globale. In una futura estensione, sarà possibile integrare strumenti più avanzati come SHAP per ottenere spiegazioni locali e più dettagliate delle predizioni.

3.7 Conclusione del capitolo

La Random Forest si è rivelata il miglior compromesso tra accuratezza e robustezza. La log-trasformazione del target ha contribuito a migliorare le performance dei modelli e a ridurre l'influenza degli outlier. I modelli sono stati integrati nella pipeline PySpark per garantire scalabilità e compatibilità con il sistema finale. Il prossimo capitolo tratterà la preparazione dei dati.

Capitolo 4 - Preprocessing e analisi del dataset

4.1 Descrizione del dataset

Il dataset utilizzato per lo sviluppo del progetto è tratto dalla competizione “House Prices – Advanced Regression Techniques” pubblicata su Kaggle¹. Esso rappresenta un benchmark noto nel mondo della Data Science, ed è stato scelto per la sua completezza, qualità e coerenza con l’obiettivo di questa tesi: prevedere il prezzo di vendita di un immobile a partire da una serie di caratteristiche strutturali, geografiche e qualitative. Il dataset contiene un totale di 1.460 osservazioni nel file di training, ciascuna rappresentante un’abitazione situata ad Ames, nello stato dell’Iowa (USA), e 81 variabili, suddivise in:

- 80 variabili indipendenti (feature), tra cui superfici interne ed esterne, numero di camere, tipologia di costruzione, qualità dei materiali, presenza di garage, piscine o caminetti, anno di costruzione e di ristrutturazione, ecc.;
- 1 variabile target: SalePrice, corrispondente al prezzo di vendita effettivo dell’abitazione espresso in dollari.

Le feature sono eterogenee per tipologia: alcune sono numeriche continue, altre intere (es. numero di bagni) e molte sono categoriche (es. Neighborhood, HouseStyle, RoofStyle). Questa varietà richiede una fase di preprocessing attenta, che comprende la codifica delle variabili categoriche e il trattamento dei valori mancanti.

Il dataset presenta inoltre diverse variabili ridondanti o altamente correlate, che necessitano di una valutazione attenta in fase di analisi esplorativa e selezione delle feature. La disponibilità di metadati e documentazione dettagliata ha inoltre facilitato l’interpretazione delle variabili e la costruzione di nuove feature più informative.

Nel complesso, si tratta di un dataset ben strutturato, ampiamente utilizzato nella letteratura per confrontare algoritmi di regressione e particolarmente adatto ad

¹ <https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques>

applicazioni di Machine Learning supervisionato, dove i dati sono accompagnati da una variabile target nota (in questo caso, il prezzo di vendita).

A differenza dell'apprendimento non supervisionato, che lavora su dati privi di etichette e mira a scoprire strutture nascoste (es. clustering o riduzione dimensionale), l'approccio adottato in questo progetto è esplicitamente orientato alla predizione numerica.

4.2 Analisi esplorativa dei dati (EDA)

L'Analisi Esplorativa dei Dati (EDA) rappresenta una fase fondamentale nel processo di Data Science, finalizzata a comprendere la struttura, le caratteristiche e le relazioni presenti all'interno di un dataset. Questo approccio, introdotto da John Tukey negli anni '70, si basa sull'utilizzo di tecniche statistiche descrittive e strumenti grafici per identificare pattern, anomalie e relazioni tra le variabili, senza formulare ipotesi a priori. Nel contesto di questa tesi, l'EDA è stata applicata al dataset della competizione "House Prices – Advanced Regression Techniques" di Kaggle, con l'obiettivo di:

- Comprendere la distribuzione delle variabili, in particolare della variabile target SalePrice;
- Identificare la presenza di outlier e valori anomali;
- Analizzare le relazioni tra le variabili indipendenti e la variabile target;
- Valutare la presenza di valori mancanti e la necessità di eventuali trasformazioni.

4.3 Trasformazioni logaritmiche e riduzione della skewness

Nel contesto dei modelli di regressione, una delle ipotesi chiave è che la relazione tra le feature e la variabile target sia approssimativamente lineare e che i residui seguano una distribuzione normale. Tuttavia, molti dataset reali — in particolare quelli immobiliari — presentano distribuzioni fortemente asimmetriche (skewed), con lunghe code verso destra o sinistra. Questa asimmetria è comune nella variabile SalePrice, che tende a mostrare una distribuzione asimmetrica positiva: la maggior parte delle abitazioni si concentra in fasce di prezzo medio-basse, mentre pochi

immobili di pregio presentano valori estremamente elevati. Questa caratteristica può interferire con l'efficienza e l'accuratezza dei modelli, il grafico della distribuzione è il seguente

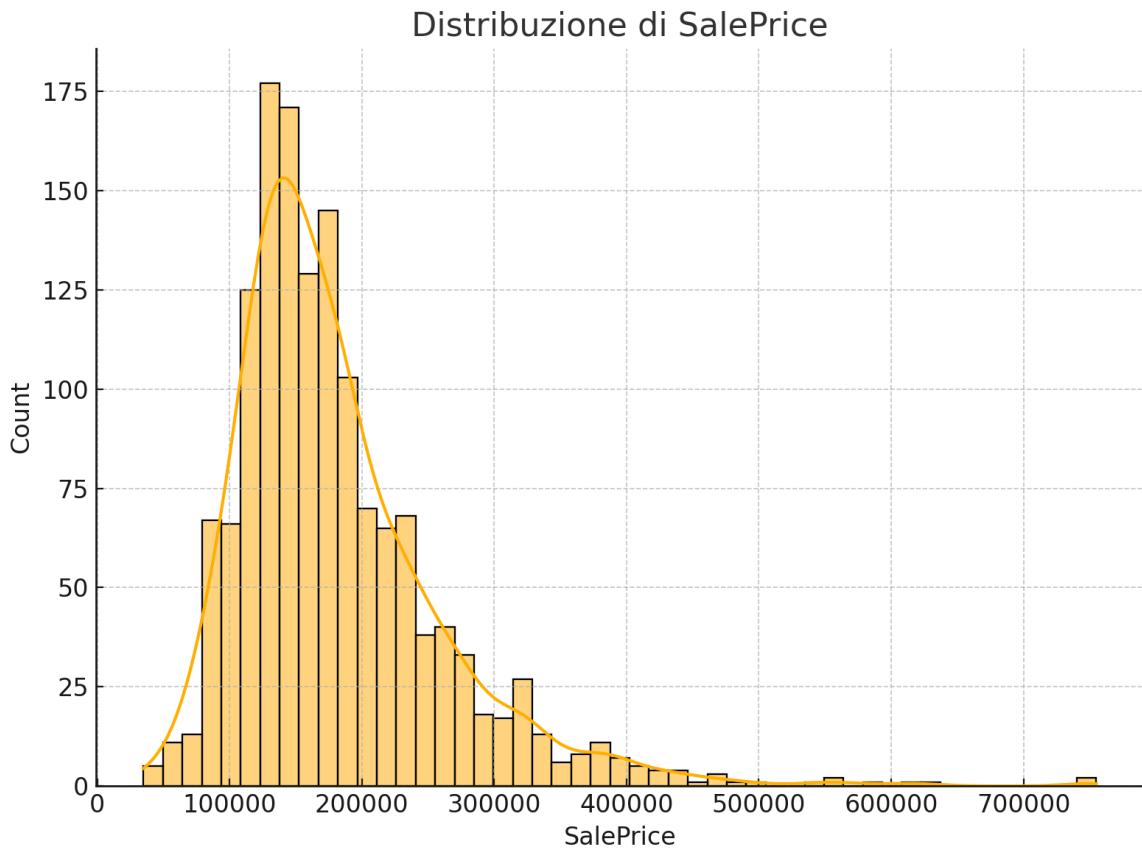


Figura 4.1 – Distribuzione reale della variabile SalePrice.

Applicazione della trasformazione logaritmica

Per ridurre la skewness della variabile target e migliorarne la distribuzione, è stata applicata una trasformazione logaritmica naturale (logp) alla variabile SalePrice. Questa tecnica ha consentito di ottenere una distribuzione più simmetrica e compatibile con le ipotesi dei modelli lineari e con la metrica RMSE (Root Mean Squared Error), che penalizza maggiormente gli errori su valori elevati. Di seguito possiamo osservare il grafico aggiornato post applicazione della trasformazione

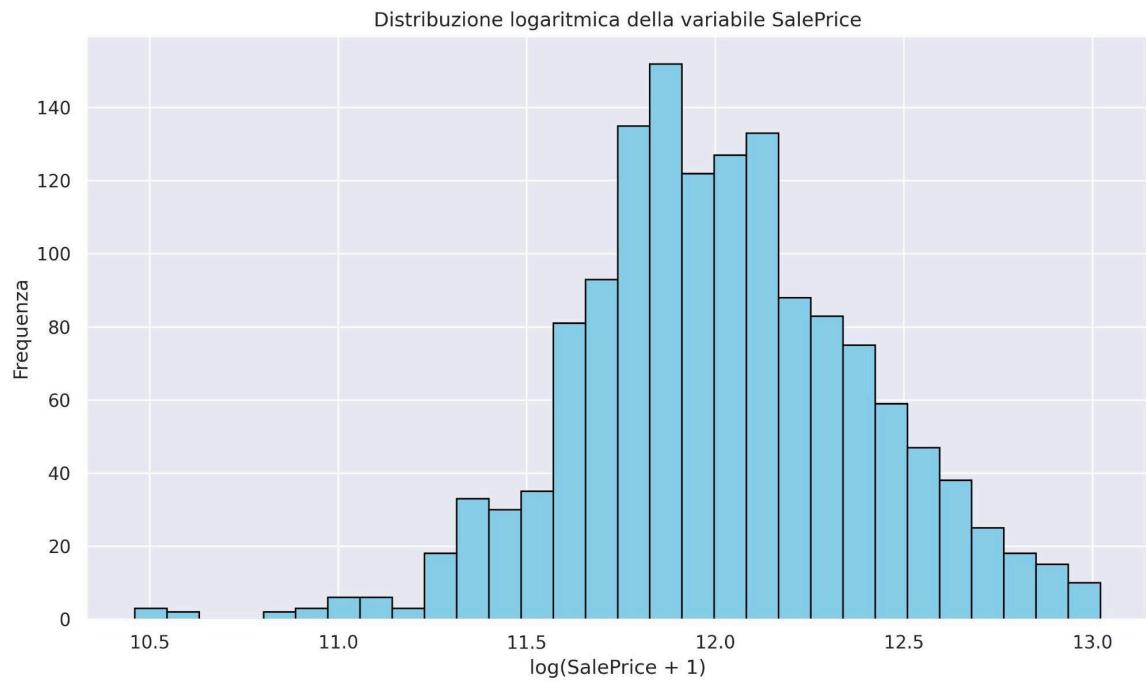


Figura 4.2 – Distribuzione logaritmica della variabile SalePrice.

La trasformazione ha portato anche a una riduzione della varianza e a una migliore linearizzazione delle relazioni con feature altamente correlate, come GrLivArea e OverallQual.

Altre trasformazioni su variabili skewed

Oltre alla variabile target, anche alcune feature numeriche presentavano skewness elevata (come LotArea, TotalBsmtSF, 1stFlrSF, ecc.). Per queste variabili è stato utilizzato un approccio selettivo, applicando la trasformazione logaritmica solo nei casi in cui:

- la skewness superava ± 1 ;
- la trasformazione migliorava la correlazione con SalePrice;
- la distribuzione post-log mostrava una simmetria accettabile.

Questo tipo di trasformazioni ha contribuito a stabilizzare la varianza, migliorando l'efficienza predittiva dei modelli basati su regressione.

4.4 Analisi delle correlazioni

È stata calcolata la matrice di correlazione tra le variabili numeriche del dataset, utilizzando il coefficiente di Pearson per mezzo della heatmap nella figura seguente

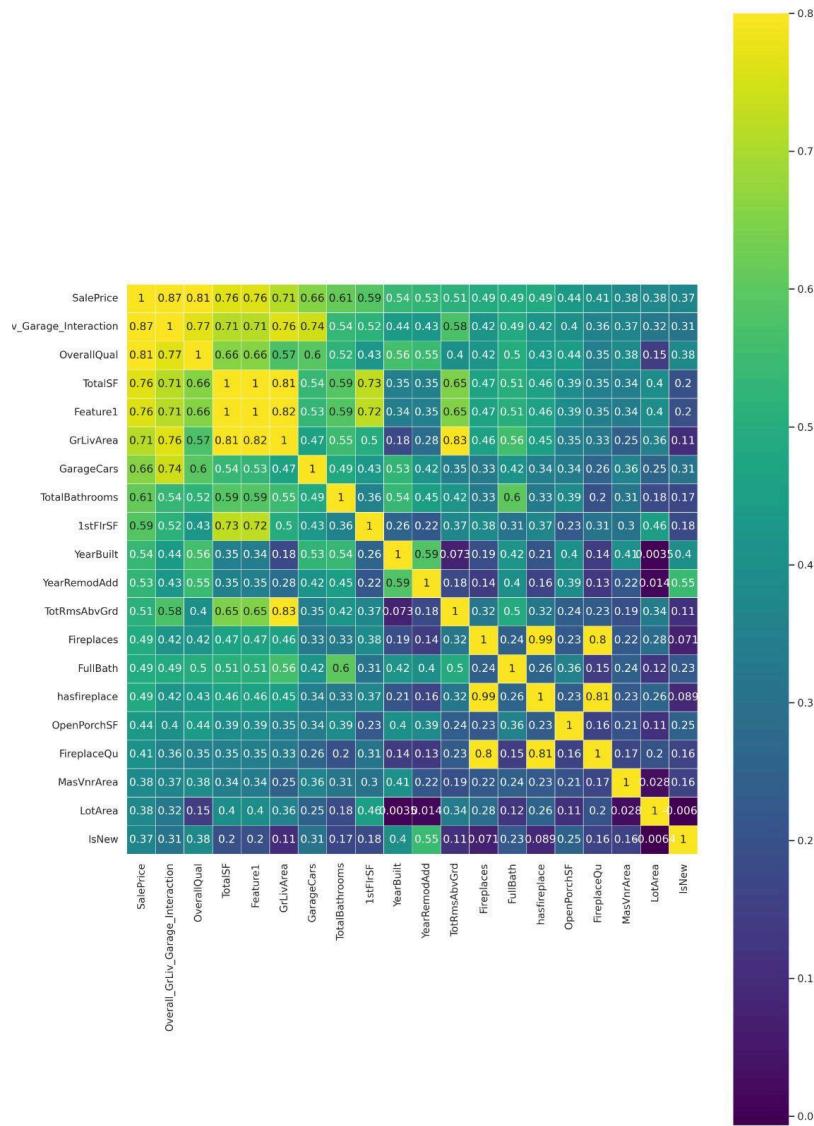


Figura 4.3 - Matrice di correlazione tra le principali variabili.

Questa analisi ha permesso di identificare le variabili con maggiore correlazione con SalePrice, tra cui:

- OverallQual: qualità complessiva dell'immobile;
- GrLivArea: superficie abitabile sopra il livello del suolo;
- GarageCars: numero di posti auto nel garage;
- TotalBsmtSF: superficie totale del seminterrato.

Queste variabili sono state considerate rilevanti per la costruzione del modello predittivo.

Visualizzazione delle relazioni

Per approfondire la relazione tra GrLivArea e SalePrice, è stato realizzato un grafico scatter plot, che evidenzia una relazione positiva tra le due variabili. Tuttavia, sono stati individuati alcuni outlier, ovvero immobili con una superficie abitabile elevata ma un prezzo di vendita relativamente basso. Questi punti anomali sono stati analizzati e se ritenuti influenti negativamente, rimossi dal dataset per migliorare la qualità dell'analisi.

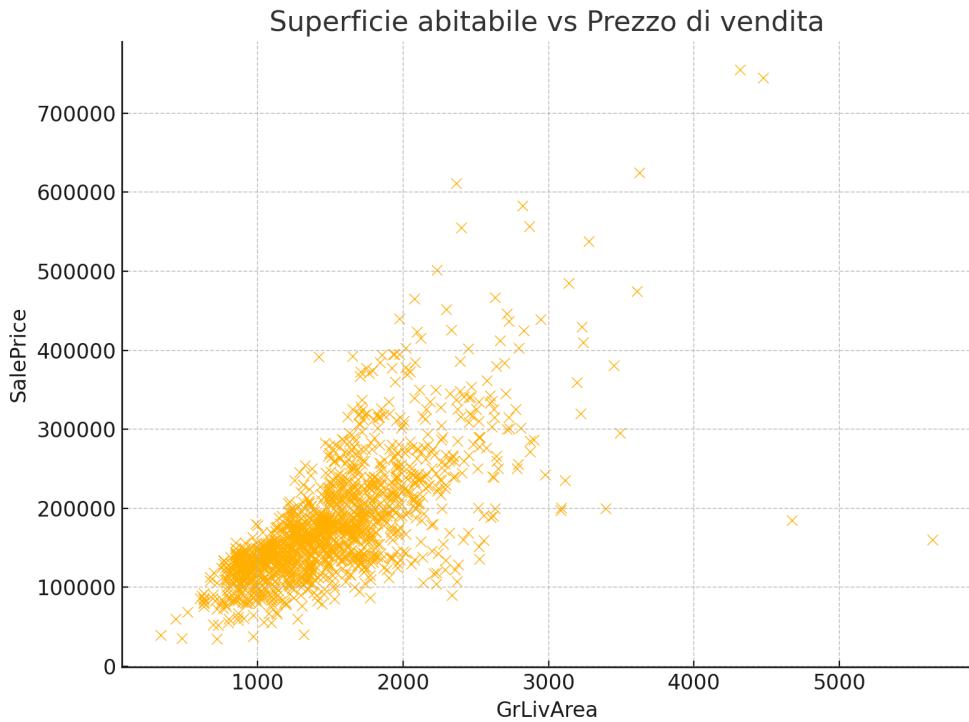


Figura 4.4 - Relazione tra superficie abitabile e prezzo di vendita.

Conclusioni dell'EDA

L'Analisi Esplorativa dei Dati ha fornito una comprensione approfondita del dataset, evidenziando le caratteristiche principali delle variabili, le relazioni tra di esse e la presenza di anomalie. I risultati ottenuti hanno guidato le successive fasi di preprocessing, selezione delle feature e costruzione dei modelli predittivi.

4.5 Identificazione e rimozione degli outlier

La rimozione degli outlier è una fase cruciale nel processo di preparazione dei dati. Gli outlier, o valori anomali, sono osservazioni che si discostano significativamente dal comportamento generale del dataset. In un contesto predittivo, la loro presenza

può compromettere la qualità del modello, influenzando negativamente la stima dei parametri, le performance e la capacità di generalizzazione.

L'individuazione degli outlier è stata effettuata tramite l'analisi grafica e statistica delle variabili numeriche più rilevanti, con particolare attenzione alla variabile target SalePrice e ad alcune delle sue feature più correlate:

- GrLivArea (superficie abitabile sopra il suolo);
- TotalBsmtSF (superficie del seminterrato);
- GarageArea (superficie del garage).

Sono stati utilizzati strumenti grafici come scatter plot e boxplot, oltre a metodi statistici come lo z-score standardizzato e l'intervallo interquartile (IQR), per rilevare le osservazioni anomale; di seguito il grafico tra SalePrice e GrLivArea



Figura 4.5 - Relazione tra area abitativa e prezzo di vendita (scatterplot).

Un esempio evidente riguarda alcune abitazioni con una GrLivArea molto elevata (superiore a 4500 piedi quadrati) ma con un prezzo di vendita basso, incompatibile con le tendenze generali del mercato. Tali punti sono stati identificati come outlier.

Strategia di rimozione

La strategia adottata per la rimozione degli outlier è stata di tipo selettivo e motivata da considerazioni sia statistiche che di dominio. In particolare, sono stati rimossi solo gli outlier che:

- presentavano un'influenza negativa evidente nei grafici;
- si discostavano di oltre 3 deviazioni standard dalla media;
- compromettevano l'adattamento lineare nei modelli preliminari.

Un esempio di rimozione è stato effettuato sulla variabile GrLivArea:

```
1 # Rimozione outlier su GrLivArea
2 df = df[df['GrLivArea'] < 4500]
```

Figura 3.5 - Rimozione degli outlier basata sulla superficie GrLivArea (estratto di codice).

Questa operazione ha portato a una maggiore omogeneità del dataset, migliorando sia la distribuzione della variabile target, sia la stabilità dei modelli nei test successivi.

Impatto sulla modellazione

La rimozione degli outlier ha prodotto un dataset più rappresentativo della realtà, riducendo la varianza non spiegabile e aumentando la coerenza statistica. Inoltre, ha contribuito a migliorare le metriche di performance (come RMSE e R²) e a rendere più robusta l'analisi delle importanze delle feature.

4.6 Trattamento dei valori mancanti

La gestione dei valori mancanti è una fase cruciale nel preprocessing dei dati, in quanto la loro presenza può influenzare negativamente le prestazioni dei modelli predittivi. Nel dataset utilizzato, sono stati identificati valori mancanti in diverse variabili, sia numeriche che categoriche.

Identificazione dei valori mancanti

Un'analisi preliminare ha evidenziato che alcune variabili presentano una percentuale significativa di valori mancanti. Ad esempio, variabili come PoolQC, MiscFeature e Alley mostrano una percentuale di valori nulli superiore al 80%. Altre variabili, come LotFrontage, presentano una percentuale di valori mancanti più contenuta, ma comunque rilevante.

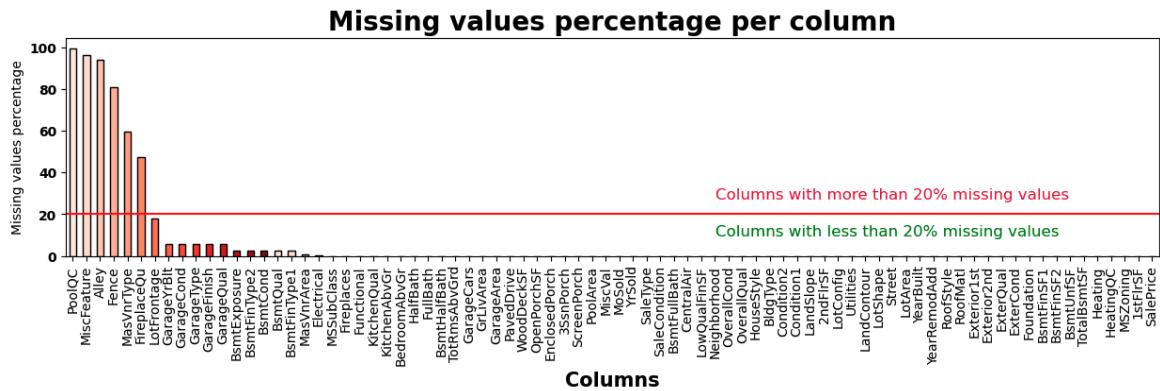


Figura 4.6 - Percentuale di valori mancanti per ciascuna variabile.

Strategie di imputazione

Per gestire i valori mancanti, sono state adottate diverse strategie, in base alla natura della variabile e alla percentuale di valori mancanti:

- Eliminazione: variabili con una percentuale di valori mancanti superiore al 40% e con bassa rilevanza predittiva sono state eliminate dal dataset;
- Imputazione con la mediana: per variabili numeriche con distribuzioni asimmetriche, i valori mancanti sono stati sostituiti con la mediana della variabile.;
- Imputazione con la moda: per variabili categoriche, i valori mancanti sono stati sostituiti con la modalità della variabile;
- Creazione di flag binari: per alcune variabili, è stata creata una nuova variabile binaria che indica la presenza o l'assenza del valore originario. Ad esempio, per la variabile GarageType, è stata creata una nuova variabile HasGarage che assume valore 1 se GarageType è presente e 0 altrimenti.

Impatto sulla modellazione

L'applicazione di queste strategie ha permesso di ridurre significativamente la quantità di valori mancanti nel dataset, migliorando la qualità dei dati e la robustezza dei modelli predittivi. Inoltre, la creazione di variabili binarie ha arricchito l'informazione disponibile per il modello, consentendo una migliore interpretazione delle relazioni tra le variabili.

4.7 Trasformazione delle variabili

Nel processo di modellazione predittiva, è spesso necessario applicare delle trasformazioni alle variabili per migliorarne la distribuzione, l'interpretabilità o l'interazione con gli algoritmi di Machine Learning. In particolare, le trasformazioni servono a:

- ridurre la skewness delle distribuzioni;
- normalizzare scale diverse tra le feature;
- rendere più esplicite relazioni non lineari;
- adattare le variabili categoriche a formati numerici compatibili con i modelli.

Trasformazioni numeriche

Alcune variabili numeriche presentavano distribuzioni fortemente asimmetriche. La skewness eccessiva può distorcere la capacità dei modelli di cogliere relazioni lineari o influenzare l'efficienza di algoritmi basati su distanze. Per ovviare a questo problema, sebbene la standardizzazione tramite Z-score sia una tecnica comune, in questo progetto sono state applicate delle trasformazioni logaritmiche alle variabili che lo necessitano per ridurre l'asimmetria verso destra e migliorare la linearità.

Trasformazioni categoriche

Le variabili categoriche, che rappresentano una parte consistente del dataset, sono state trasformate in formato numerico con due approcci distinti:

- Ordinal Encoding: per variabili ordinali (come OverallQual, ExterQual, KitchenQual), che presentano un ordine naturale tra le categorie. In questi casi, a ogni categoria è stato assegnato un valore numerico coerente con l'ordinamento隐式的;
- One-Hot Encoding: per variabili categoriche nominali (es. Neighborhood, Exterior1st), che non presentano un ordinamento intrinseco. Questo metodo ha permesso di trasformare ogni categoria in una variabile binaria separata, evitando interpretazioni distorte da codifiche numeriche arbitrarie.

Coerenza tra trasformazioni

Durante il preprocessing, è stato fondamentale garantire coerenza tra le trasformazioni effettuate sul dataset di training e quelle da applicare in fase di inferenza. Per questo motivo:

- tutte le trasformazioni sono state integrate in pipeline riutilizzabili;
- i mapping delle categorie ordinali e nominali sono stati salvati in un file JSON separato, così da poter essere ricaricati nella fase di predizione attraverso la dashboard interattiva. Questo approccio garantisce consistenza tra il comportamento del modello in fase di training e in fase di produzione.

4.8 Feature engineering: nuove variabili e boolean logici

Il processo di feature engineering consiste nella creazione di nuove variabili derivate da quelle esistenti, con l'obiettivo di migliorare la capacità predittiva del modello. In molti casi, i dati grezzi non esprimono direttamente l'informazione utile: la loro combinazione, trasformazione o aggregazione può portare alla generazione di feature più informative, interpretabili e significative dal punto di vista predittivo.

Nuove variabili derivate

Sono state create diverse variabili calcolate a partire da combinazioni o trasformazioni logiche di variabili esistenti. Tra le più rilevanti:

- TotalSF: superficie complessiva dell’immobile, ottenuta come somma tra la superficie del seminterrato (TotalBsmtSF), del piano terra (1stFlrSF) e del secondo piano (2ndFlrSF);
- AgeAtSale: età dell’immobile al momento della vendita, calcolata come differenza tra l’anno di vendita (YrSold) e l’anno di costruzione (YearBuilt);
- Remodeled: variabile binaria che indica se l’immobile è stato ristrutturato, ottenuta confrontando YearBuilt con YearRemodAdd.
- Bathrooms: numero totale di bagni, pesando i mezzi bagni (half bath) con un fattore di 0.5;
- Overall_GrLiv_Garage_Interaction: interazione tra qualità globale dell’immobile (OverallQual), superficie abitabile (GrLivArea) e area del garage (GarageArea), utile per rappresentare l’effetto combinato tra grandezza, accessori e qualità.

Variabili boolean logici

Per migliorare la rappresentazione di caratteristiche binarie o potenzialmente mancanti, sono state create variabili booleane come:

- HasGarage: indica la presenza di garage;
- HasBasement: indica la presenza di un seminterrato;
- HasFireplace: presenza di caminetto;
- Has2ndFloor: presenza di secondo piano;
- HasPool: presenza di piscina.

Queste variabili hanno il vantaggio di esplicitare elementi altrimenti impliciti nella codifica dei valori mancanti o nel valore zero, migliorando la leggibilità del modello e supportando algoritmi sensibili ai valori nulli.

Impatto del feature engineering

Le nuove feature hanno dimostrato di avere una correlazione significativa con il prezzo di vendita (SalePrice) e sono state mantenute nei modelli finali. Inoltre, alcune di esse sono risultate tra le più influenti nei metodi di explainability come SHAP e feature importance delle Random Forest.

4.9 Gestione della multicollinearità

La multicollinearità è un fenomeno statistico che si verifica quando due o più variabili indipendenti in un modello di regressione risultano fortemente correlate tra loro. Questa condizione può compromettere la stabilità del modello, causando instabilità nei coefficienti stimati e riducendo l'affidabilità dell'interpretazione dei pesi associati alle feature.

Impatto della multicollinearità sui modelli

In presenza di multicollinearità, i modelli lineari tendono a produrre coefficienti altamente sensibili a piccole variazioni nei dati, con conseguente instabilità numerica. Anche i modelli non lineari, come le random forest o il boosting, pur essendo più robusti a questo problema, possono comunque risentirne in termini di ridondanza informativa e overfitting.

Inoltre, la multicollinearità rende difficile valutare il contributo individuale di una variabile alla predizione del target, aspetto particolarmente critico nel caso di modelli interpretabili.

Metodologie di rilevamento

Nel presente progetto, il rilevamento della multicollinearità non è stato effettuato tramite indicatori statistici avanzati come il Variance Inflation Factor (VIF), ma piuttosto attraverso un approccio empirico e visivo. È stata calcolata la matrice di correlazione (vedi Figura 3) tra tutte le variabili numeriche, da cui sono state selezionate le 20 feature con la correlazione più elevata rispetto alla variabile target SalePrice.

Strategie adottate

Per mitigare l'effetto della multicollinearità, sono state adottate diverse strategie:

- rimozione di feature ridondanti: in presenza di variabili altamente correlate tra loro, è stata mantenuta la più interpretabile o meglio correlata con SalePrice;
- feature engineering selettivo: alcune feature derivate, come TotalSF, sono state utilizzate in sostituzione delle loro componenti (1stFlrSF, 2ndFlrSF, TotalBsmtSF) per ridurre la ridondanza;
- regolarizzazione nei modelli: nei test con modelli lineari regolarizzati (es. Ridge, Lasso), la multicollinearità è stata gestita in modo naturale tramite penalizzazioni sui coefficienti.

4.10 Conclusione del capitolo

In questo capitolo è stato descritto in dettaglio il processo di analisi preliminare e preparazione del dataset, fondamentale per garantire l'efficacia e l'affidabilità dei modelli predittivi sviluppati nei capitoli successivi. Le attività svolte si sono articolate in diverse fasi, ciascuna delle quali ha contribuito a migliorare la qualità, il contributo informativo e la consistenza del dataset. L'analisi esplorativa (EDA) ha permesso di identificare le principali relazioni tra le variabili, evidenziando pattern rilevanti e la presenza di outlier significativi. La rimozione mirata di queste anomalie ha permesso di stabilizzare la distribuzione dei dati, evitando effetti distorsivi nei modelli di regressione. Il trattamento dei valori mancanti è stato condotto in modo flessibile e contestuale, combinando tecniche di imputazione e creazione di variabili binarie, così da preservare quanto più possibile l'informazione originale. Le trasformazioni numeriche e categoriche hanno poi garantito l'adattabilità delle variabili agli algoritmi di machine learning, evitando distorsioni dovute a scale disomogenee o codifiche arbitrarie. La feature engineering ha consentito di derivare nuove variabili più informative e facilmente interpretabili, mentre l'attenzione alla multicollinearità ha migliorato la robustezza statistica dei modelli. Infine, l'applicazione selettiva della trasformazione logaritmica ha ridotto l'asimmetria della variabile target e di altre feature chiave, aumentando la linearità e

l'efficienza predittiva. Nel complesso, il preprocessing effettuato ha permesso di costruire un dataset pulito, arricchito e pronto per la modellazione predittiva avanzata, che sarà trattata in dettaglio nel Capitolo 5.

Capitolo 5 - Modellazione predittiva

5.1 Obiettivi della modellazione

L'obiettivo principale della modellazione predittiva sviluppata in questo lavoro è stimare il prezzo di vendita di un immobile residenziale a partire da un insieme di caratteristiche strutturali, qualitative e contestuali. Questo processo rientra nell'ambito della regressione supervisionata, in cui si cerca di apprendere una funzione predittiva $f: X \rightarrow y$ in grado di associare a ogni combinazione di feature X un valore numerico continuo y , corrispondente al prezzo atteso dell'immobile. Nel caso specifico, la variabile target è SalePrice, che rappresenta il valore in dollari dell'abitazione. Come discusso nei capitoli precedenti, tale variabile è stata oggetto di trasformazione logaritmica al fine di ridurre la skewness e migliorare la linearità delle relazioni con le feature esplicative.

Obiettivi tecnici

Il processo di modellazione si è posto i seguenti obiettivi tecnici:

- accuratezza predittiva: costruire un modello capace di stimare il prezzo con errore minimo rispetto ai dati reali;
- robustezza: ottenere prestazioni stabili anche su dati non visti (generalizzazione).
- efficienza: ridurre il tempo di addestramento e predizione per garantire un'integrazione fluida nella dashboard;
- interpretabilità: privilegiare, ove possibile, modelli che offrano trasparenza sui meccanismi di decisione, coerentemente con l'approccio Human-Centered AI.

Obiettivi applicativi

Oltre agli aspetti tecnici, la modellazione è stata pensata in funzione dell'uso reale in ambito professionale, per supportare agenti immobiliari e stakeholder nel processo di valutazione. In particolare:

- l'output del modello doveva essere facilmente interpretabile da parte dell'utente finale;
- modelli dovevano essere compatibili con un'infrastruttura semplice e leggera (es. backend Python e interfaccia Streamlit);
- il flusso di inferenza doveva essere completo ed end-to-end, integrando preprocessing, predizione e interfaccia utente.

Metriche di valutazione

Per valutare le performance dei modelli testati sono state adottate principalmente due metriche:

- root Mean Squared Error (RMSE): metrica standard per problemi di regressione, che penalizza maggiormente gli errori elevati e si adatta bene a variabili con scala monetaria;
- R^2 (coefficiente di determinazione): misura la proporzione di varianza della variabile target spiegata dal modello.

Entrambe le metriche sono state calcolate tramite validazione incrociata, sia in fase di sviluppo che di test finale, per garantire una valutazione equa e rappresentativa delle prestazioni del modello.

5.2 Preparazione del dataset per il training

Prima di procedere all'addestramento dei modelli predittivi, è stato necessario effettuare una preparazione accurata del dataset, al fine di garantire la qualità dei dati e la coerenza delle trasformazioni applicate. Questa fase si inserisce nel flusso end-to-end della pipeline di Machine Learning e rappresenta un passaggio cruciale per la successiva efficacia del modello.

Suddivisione del dataset

Il dataset originale è stato suddiviso in due sottoinsiemi:

- training set (80%): utilizzato per addestrare il modello e ottimizzarne i parametri.
- test set (20%): utilizzato esclusivamente per la valutazione finale del modello, così da simulare dati "non visti" e prevenire fenomeni di overfitting.

La divisione è stata effettuata in modo casuale ma riproducibile, utilizzando un `random_state` costante, per garantire coerenza tra esperimenti.

```
splits = train_vector.randomSplit([0.8, 0.2])
train = splits[0]
val = splits[1]
```

Figura 5.1 - Selezione delle variabili predittive per il training (estratto di codice).

Feature scaling

Anche se non tutti i modelli utilizzati richiedono un'esplicita normalizzazione delle feature, è stata applicata una standardizzazione (Z-score) alle variabili numeriche, in particolare per la regressione lineare, al fine di:

- uniformare le scale delle variabili;
- migliorare la convergenza dei modelli;
- garantire confrontabilità tra i coefficienti.

```
test_string_columns = []

for col, dtype in test_df.dtypes:
    if dtype == 'string':
        test_string_columns.append(col)

indexers2 = [StringIndexer(inputCol=column, outputCol=column+'_index',
                           handleInvalid='keep').fit(test_df) for column in test_string_columns]

pipeline2 = Pipeline(stages=indexers2)
test_indexed = pipeline2.fit(test_df).transform(test_df)
```

Figura 5.2 - Encoding delle variabili categoriche tramite StringIndexer (estratto di codice).

Selezione delle feature

Le feature utilizzate per l'addestramento sono il risultato del processo di:

- pulizia e imputazione dei valori mancanti;
- trasformazione di variabili numeriche e categoriche;
- feature engineering (es. TotalSF, Overall_GrLiv_Garage_Interaction);
- rimozione di variabili altamente collineari (es. GarageCars, TotRmsAbvGrd).

La selezione finale ha incluso circa 60 variabili, tutte numeriche, ottenute anche tramite encoding delle variabili categoriche (OrdinalEncoder, factorize, ecc.). Le trasformazioni applicate sono state salvate per garantire consistenza nella fase di inferenza.

Coerenza con la dashboard

Particolare attenzione è stata dedicata alla compatibilità con la dashboard interattiva sviluppata nei capitoli successivi. La struttura delle feature è stata mantenuta uniforme, e sono stati salvati i mapping delle variabili categoriche in formato json, così da poter essere utilizzati per convertire dinamicamente gli input utente all'interno dell'interfaccia Streamlit.

5.3 Modelli di regressione implementati

Nel presente progetto sono stati implementati diversi modelli di regressione, con l'obiettivo di confrontarne le prestazioni e identificare il più adatto alla previsione del prezzo di vendita degli immobili. La scelta è ricaduta su tre approcci principali, ciascuno appartenente a una diversa famiglia algoritmica: regressione lineare, Random Forest e XGBoost. I modelli sono stati valutati sia dal punto di vista delle

prestazioni predittive, sia in termini di robustezza e compatibilità con l’architettura della dashboard.

Regressione lineare

La regressione lineare è stata utilizzata come modello di base (baseline), implementata tramite la classe LinearRegression del modulo pyspark.ml.regression. Il modello assume una relazione lineare tra la variabile target (SalePrice) e le feature esplicative.

L’addestramento è avvenuto sul dataset preprocessato e trasformato tramite un VectorAssembler, seguito da una divisione in training e test set con proporzione 80/20. Il modello ha restituito un R^2 pari a circa 0.67, indicando una discreta capacità esplicativa ma inferiore rispetto ai modelli non lineari. Il valore di RMSE era anch’esso relativamente elevato, suggerendo un margine di miglioramento. Nonostante le prestazioni contenute, la regressione lineare si è rivelata utile per comprendere l’effetto diretto e individuale delle feature sulla variabile target.

Random Forest

Il modello Random Forest è stato il principale candidato per la predizione finale. Si tratta di un algoritmo di tipo ensemble basato su alberi decisionali, in grado di catturare interazioni complesse tra le feature senza richiedere trasformazioni lineari. L’implementazione è avvenuta con RandomForestRegressor di PySpark, utilizzando un ParamGridBuilder per testare diverse combinazioni di iperparametri (maxDepth, numTrees, maxBins). L’ottimizzazione è stata effettuata tramite CrossValidator con 3 fold, integrata in una Pipeline. Il modello ha raggiunto una R^2 di circa 0.87, con una significativa riduzione del RMSE rispetto alla regressione lineare. La robustezza a outlier e la capacità di gestire variabili con distribuzioni non normali lo hanno reso particolarmente adatto al contesto.

XGBoost (fase esplorativa)

Una fase esplorativa del progetto ha incluso l’addestramento di un modello XGBoost in ambiente locale. Il modello ha restituito ottime prestazioni, grazie alla regolarizzazione e alla gestione avanzata dell’overfitting. Tuttavia, la sua integrazione nel flusso PySpark e nella dashboard è stata ritenuta eccessivamente onerosa rispetto ai benefici marginali ottenuti. Pertanto, XGBoost è stato utilizzato come riferimento tecnico, ma non è stato incluso nel modello finale distribuito.

Approfondimento: Feature importance e spiegabilità

Uno dei vantaggi principali dei modelli ad albero come la Random Forest è la possibilità di valutare l’importanza relativa delle feature utilizzate nel processo decisionale. In PySpark, tale informazione è accessibile tramite l’attributo `.featureImportances` del modello addestrato (`bestModel`), che restituisce un array ponderato delle variabili utilizzate. Nel presente progetto, tale funzionalità non è stata esplicitamente implementata, ma rappresenta un importante spunto per sviluppi futuri. La visualizzazione delle feature più influenti può infatti:

- migliorare l’interpretabilità del modello da parte dell’utente finale;
- supportare decisioni più informate nella selezione delle variabili;
- favorire l’integrazione con strumenti di Explainable AI (XAI) come SHAP, soprattutto in un contesto Human-Centered AI.

L’introduzione della feature importance nella dashboard potrebbe inoltre rafforzare la trasparenza del sistema e contribuire a un uso più consapevole delle previsioni generate.

5.4 Confronto tra i modelli

Al termine della fase di addestramento, i modelli selezionati sono stati sottoposti a una valutazione comparativa basata su metriche quantitative e criteri qualitativi. Il

confronto ha coinvolto i tre principali algoritmi testati: regressione lineare, Random Forest e, in fase esplorativa, XGBoost.

Metriche di valutazione

Per la valutazione delle prestazioni sono state utilizzate due metriche standard nei problemi di regressione:

- Root Mean Squared Error (RMSE): misura l'errore quadratico medio tra i valori predetti e quelli reali. Penalizza in modo più severo gli errori di maggiore entità;
- R^2 (coefficiente di determinazione): rappresenta la proporzione della varianza della variabile target spiegata dal modello.

Queste metriche sono state calcolate sul dataset di test (20% dei dati), separato dal training set per evitare fenomeni di overfitting.

Risultati

I risultati ottenuti sono sintetizzati nella seguente tabella:

Modello	Accuratezza (R^2)	Interpretabilità	Overfitting	Velocità
Regressione Lineare	67%	Alta	Bassa	Alta
Random Forest	87%	Media	Bassa	Media
XGBoost (locale)	89%	Bassa	Bassa	Bassa

Tabella 5.1 — Confronto tra i modelli predittivi testati: accuratezza, interpretabilità, rischio di overfitting e velocità di inferenza.

Nota: il valore di R^2 per XGBoost è stato stimato su un sottoinsieme dei dati in ambiente locale, e non è direttamente comparabile agli altri.

Analisi qualitativa

La regressione lineare si è rivelata utile come baseline, ma ha mostrato limiti evidenti in termini di adattabilità ai dati complessi. La Random Forest ha

rappresentato il miglior compromesso tra accuratezza e stabilità, risultando il modello finale adottato per l'integrazione nella dashboard. L'XGBoost, pur offrendo prestazioni lievemente superiori, non è stato integrato nel flusso PySpark per motivi di semplicità, compatibilità e tempi di inferenza.

Conclusioni della comparazione

La scelta finale del modello è ricaduta sulla Random Forest, per le seguenti ragioni:

- ottimo equilibrio tra accuratezza e robustezza;
- maggiore tolleranza agli outlier e alle feature non normalizzate;
- maggiore flessibilità nella gestione di feature eterogenee;
- possibilità di estensione futura con metodi di interpretabilità come featureImportances.

Il prossimo passo è l'ottimizzazione e la messa in produzione del modello scelto, che verrà descritto nella sezione successiva in relazione all'infrastruttura distribuita adottata.

5.5 Selezione del modello finale

Dopo aver confrontato le prestazioni dei modelli implementati, la scelta del modello finale è ricaduta sulla Random Forest, nella sua versione ottimizzata tramite CrossValidator all'interno della pipeline PySpark. Questa decisione è stata guidata non solo dalla metrica di accuratezza, ma anche da altri fattori rilevanti nel contesto di un'applicazione reale, tra cui:

- robustezza alle anomalie e ai dati rumorosi, che è fondamentale in contesti dove le informazioni possono provenire da fonti eterogenee e non sempre pulite;
- buona capacità predittiva anche in presenza di relazioni non lineari tra le variabili;
- flessibilità rispetto a dati mancanti e feature categoriali, grazie all'architettura ad albero;
- efficienza e parallelizzabilità, caratteristiche particolarmente importanti per l'implementazione in ambienti distribuiti come Apache Spark.

Compatibilità con la dashboard

Un ulteriore aspetto decisivo è stato rappresentato dalla facilità di integrazione del modello nella dashboard predittiva sviluppata nel Capitolo 6. La Random Forest si è dimostrata compatibile con:

- la struttura delle feature progettata nell'interfaccia utente;
- il formato degli input generati dalla dashboard Streamlit;
- i tempi di risposta richiesti in un ambiente web (inferiori a 1 secondo).

Preparazione per la messa in produzione

Per rendere il modello pronto alla messa in produzione, sono state adottate le seguenti strategie:

- salvataggio del modello addestrato in formato serializzato (.pkl) o tramite `model.save()` in Spark;
- salvataggio dei mapping delle feature categoriche per garantire la coerenza tra training e inferenza;
- realizzazione di uno script API-based per il deployment del modello tramite FastAPI, descritto nei capitoli successivi.

Questi accorgimenti hanno permesso di costruire una pipeline completa, dalla raccolta dati all'interfaccia utente, in linea con i principi della Data Science applicata e riproducibile.

Prospettive di miglioramento

Sebbene la Random Forest abbia dimostrato ottime prestazioni, esistono possibili miglioramenti che potranno essere esplorati in sviluppi futuri:

- utilizzo di modelli più avanzati (es. XGBoost in ambiente distribuito);
- ottimizzazione fine degli iperparametri su una gamma più ampia;
- aggiunta di funzionalità di interpretabilità (es. SHAP, `featureImportances`) per migliorare la trasparenza del sistema.

5.6 Messa in produzione del modello e infrastruttura applicativa

Dopo aver selezionato il modello finale, si è proceduto alla messa in produzione dello stesso, integrandolo in un'infrastruttura applicativa progettata per essere accessibile anche a utenti non tecnici.

Questa fase rappresenta il ponte tra l'attività di modellazione offline e l'utilizzo reale del sistema predittivo da parte di figure professionali come agenti immobiliari, periti e consulenti.

5.6.1 Obiettivo operativo

L'obiettivo non era semplicemente ottenere un buon modello predittivo, ma realizzare una soluzione end-to-end, completa e fruibile, capace di:

- ricevere input strutturati da parte dell'utente finale;
- inoltrare tali input a un modello predittivo tramite una API backend;
- restituire in tempo reale la stima del prezzo dell'immobile, in modo semplice e comprensibile.

Il risultato di questo processo è una dashboard interattiva, pensata secondo principi di user-centered design, in grado di coniugare potenza analitica e accessibilità operativa.

5.6.2 Architettura tecnica

L'infrastruttura finale si compone di tre principali componenti, ciascuno containerizzato per garantire isolamento e scalabilità:

Backend (FastAPI): Applicazione Python responsabile della gestione delle richieste HTTP POST, della validazione dei dati in formato JSON e della generazione delle predizioni utilizzando il modello serializzato. Il backend è configurato per l'accesso esclusivo tramite rete locale, incrementando così il livello di sicurezza.

Modello predittivo: Il modello Random Forest selezionato è stato serializzato in formato .pkl e caricato all'avvio del backend. È ottimizzato per elaborare input conformi alla struttura di feature definita nella fase di preprocessing, restituendo un output numerico rappresentante il prezzo stimato (applicando, se necessario, la trasformazione inversa del logaritmo).

Frontend (Streamlit):

Interfaccia web sviluppata in Streamlit, che permette l'inserimento dei parametri immobiliari mediante elementi interattivi come slider, menu a tendina e checkbox. L'interfaccia implementa sezioni collassabili e vincoli dinamici per migliorare la qualità dei dati raccolti.

5.6.3 Schema architetturale del processo di inferenza

La figura seguente rappresenta schematicamente il flusso delle informazioni tra i principali componenti del sistema:

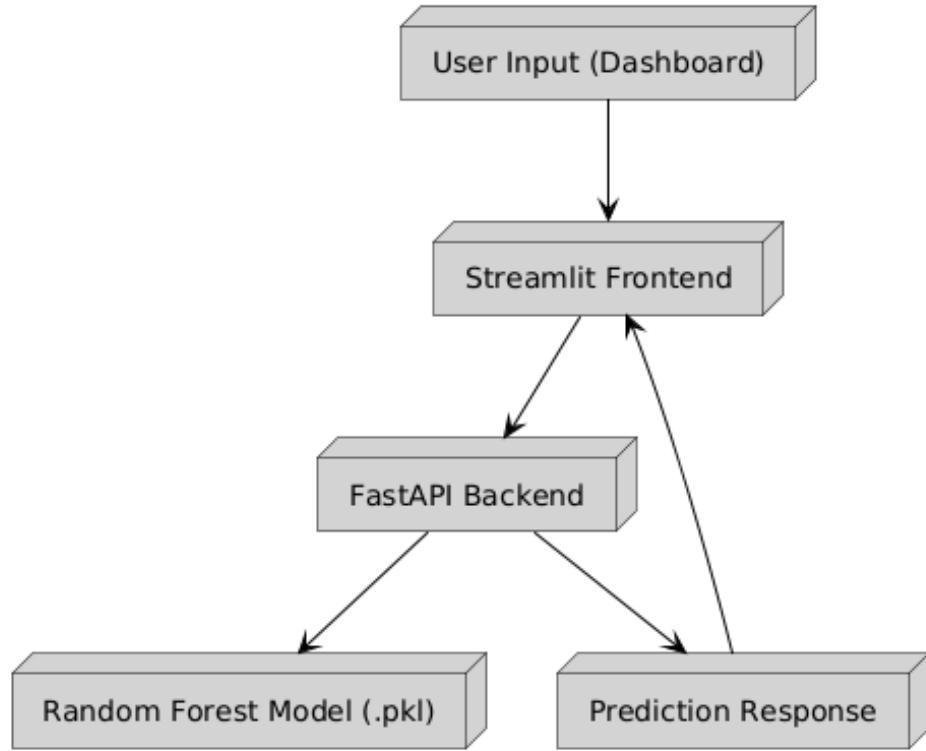


Figura 5.6 - Schema logico del processo di inferenza.

Nel prossimo capitolo sarà esposto un vero e proprio diagramma di deployment.

5.7 Conclusione del capitolo

Il Capitolo 4 ha descritto il cuore operativo del progetto: dalla costruzione e valutazione dei modelli predittivi fino alla loro integrazione in un sistema accessibile e scalabile.

A partire da una definizione chiara degli obiettivi di business, si è proceduto con una pipeline rigorosa di preprocessing, modellazione e validazione, culminata nella selezione del modello Random Forest per le sue prestazioni robuste (misurate tramite metriche quali RMSE e R²).

Parallelamente alla validazione analitica, è stata data grande attenzione all'integrazione pratica del modello, con la costruzione di una infrastruttura containerizzata che separa chiaramente frontend, backend e logica predittiva.

Questo approccio ha permesso di trasformare un esperimento analitico in uno strumento applicabile nel mondo reale, pronto per essere utilizzato da utenti non esperti di Data Science.

Infine, sono stati identificati potenziali sviluppi futuri, tra cui:

- miglioramento della interpretabilità dei modelli attraverso tecniche di Explainable AI (es. SHAP Values);
- estensione del sistema a modelli predittivi più avanzati (es. XGBoost, LightGBM).
- adattamento a contesti geografici differenti o nuove tipologie di immobili.

Il prossimo capitolo sarà dedicato alla dashboard predittiva, progettata per rendere fruibile il sistema agli utenti finali tramite un'interfaccia accessibile, interattiva e pensata secondo principi di ergonomia cognitiva e user experience.

Capitolo 6 - Progettazione e sviluppo della parte predittiva

6.1 Obiettivi della dashboard

La dashboard sviluppata rappresenta il punto di contatto tra l'utente e il sistema predittivo per la stima del prezzo degli immobili, e svolge un ruolo centrale all'interno dell'architettura progettuale.

Il suo obiettivo principale è quello di abilitare una fruizione semplice, sicura e interattiva delle funzionalità di Machine Learning sottostanti, garantendo al contempo coerenza nell'inserimento dei dati e affidabilità nei risultati restituiti.

Dal punto di vista operativo, la dashboard si prefigge di:

- Offrire un'interfaccia user-friendly, in grado di guidare l'utente attraverso il processo di inserimento delle informazioni relative all'immobile, grazie a una suddivisione logica delle sezioni (dimensioni, aree esterne, materiali, caratteristiche aggiuntive, ecc.) che ne semplifica la compilazione e riduce il rischio di errori.
- Assicurare la qualità e la consistenza dei dati raccolti, implementando vincoli dinamici e controlli sui valori immessi. Alcuni esempi includono il calcolo automatico dell'area abitabile complessiva sulla base delle superfici inserite per i singoli piani, l'adeguamento automatico della superficie del garage in funzione del numero di posti auto selezionati, o la gestione intelligente della presenza o assenza di caratteristiche accessorie (come piscina, seminterrato o caminetto).
- Garantire un accesso sicuro e controllato al sistema, attraverso l'integrazione di un modulo di autenticazione basato su OAuth2, che si appoggia al provider Keycloak. Gli utenti sono pertanto tenuti a effettuare il login prima di poter utilizzare le funzionalità di predizione, e i loro token di accesso sono gestiti con politiche di refresh automatico per garantire una sessione continua e sicura.
- Consentire l'esecuzione della predizione in tempo reale, inviando i dati raccolti tramite richiesta REST al servizio FastAPI backend, che si occupa di validare il token ricevuto, preprocessare l'input, e interrogare il modello predittivo basato su algoritmi di regressione avanzata (in particolare XGBoost).

- Favorire la tracciabilità e l'analisi delle predizioni effettuate, attraverso la pubblicazione automatica di ciascun risultato su un broker Kafka. Questo permette di raccogliere in modo asincrono tutti gli eventi di predizione, abilitando così future analisi aggregate dei dati (batch analytics), utili per studi di mercato, miglioramenti del modello o monitoraggio dell'utilizzo del sistema.
- Allinearsi a principi di sviluppo agile, con l'introduzione di strumenti come una kanban board per la gestione visuale delle attività, la suddivisione delle funzionalità in issue e l'adozione di pratiche DevOps quali branching per feature e pull request su un branch di sviluppo condiviso.

Nel loro insieme, questi obiettivi concorrono a realizzare una dashboard che non si limita ad essere un semplice front-end di inserimento dati, ma si configura come un elemento chiave in un sistema di Data Science applicata, capace di integrare pratiche di sicurezza informatica, buone prassi di ingegneria del software, e tecniche di Machine Learning avanzate, in un'unica soluzione coesa e performante.

6.2 Requisiti funzionali e tecnici

A supporto dei requisiti funzionali, sono stati definiti i seguenti vincoli e specifiche tecniche:

- Framework di frontend: Streamlit, scelto per la sua rapidità nello sviluppo di dashboard interattive Python-based, dotato di elementi UI dinamici e gestione nativa degli stati di sessione.
- Backend API di predizione: FastAPI, per la gestione delle richieste RESTful in modo asincrono e performante, con integrazione nativa della validazione dei token JWT e dei dati in ingresso.
- Sistema di autenticazione: Keycloak come Identity Provider, configurato per supportare il flusso di autenticazione OAuth2 Resource Owner Password Credentials. Validazione lato backend tramite verifica della firma dei token JWT usando le chiavi pubbliche recuperate dinamicamente tramite OpenID Connect Discovery.
- Modello di Machine Learning: XGBoost, algoritmo di gradient boosting ad alte prestazioni, addestrato su dati immobiliari, e serializzato in formato joblib per il caricamento runtime.

- Sistema di messaggistica asincrona: Apache Kafka, con configurazione locale containerizzata (Confluent Platform) per l'invio e la gestione degli eventi di predizione.
- Infrastruttura di deployment: Tutti i componenti del sistema (frontend, backend, Keycloak, Kafka, Zookeeper) sono orchestrati tramite Docker Compose, in un ambiente di rete condiviso dedicato.
- Gestione del ciclo di vita del codice: Utilizzo di Git, con branching model che include main, develop, e branch temporanei per ogni nuova feature, gestiti tramite pull request e successiva integrazione controllata.

Questa definizione rigorosa dei requisiti ha guidato lo sviluppo di una piattaforma solida, sicura ed estensibile, capace di coniugare pratiche moderne di ingegneria del software con l'esigenza di fornire uno strumento accessibile ed efficace agli utenti finali.

6.3 Progettazione dell'interfaccia utente (UI/UX)

La progettazione dell'interfaccia utente ha seguito principi di chiarezza, semplicità e coerenza semantica, con l'obiettivo di realizzare una dashboard che fosse al contempo intuitiva nell'utilizzo e efficace nel supportare l'utente nella compilazione dei dati richiesti per la predizione.

6.3.1 Filosofia progettuale

In fase di definizione della UI, sono stati adottati i seguenti principi guida:

- Minimalismo funzionale: Ogni elemento grafico presente ha uno scopo preciso e contribuisce all'obiettivo dell'utente. È stata evitata l'introduzione di funzionalità superflue o ridondanti.
- Organizzazione logica e modulare: I dati da inserire sono stati suddivisi in sezioni collassabili, ciascuna dedicata a una specifica categoria di caratteristiche dell'immobile: dimensioni, aree esterne, struttura, stato, garage, piscina, caminetto, e altre informazioni accessorie.
- Validazioni dinamiche e automatismi: Alcuni campi sono correlati tra loro: ad esempio, la superficie del garage è vincolata al numero di posti auto selezionati, mentre la superficie abitabile totale (GrLivArea) è calcolata automaticamente sulla base delle superfici dei singoli piani.
- Feedback immediato e interattività: Ad ogni interazione, l'utente riceve un feedback visivo (valori calcolati in tempo reale, messaggi di conferma o errore) al fine di ridurre l'incertezza operativa e migliorare l'esperienza complessiva.
- Accessibilità delle funzionalità: La dashboard è accessibile solo previo login sicuro tramite autenticazione OAuth2, e la sessione viene gestita in modo trasparente per l'utente.

6.3.2 Struttura del wireframe

In fase di progettazione preliminare è stato realizzato un wireframe che rappresenta la struttura di alto livello della dashboard. Il wireframe ha costituito la base di partenza per la successiva implementazione su Streamlit, ed è stato fondamentale per visualizzare anticipatamente il flusso operativo previsto per l'utente.

Preventivo vendita case

Dimensioni

Superficie primo piano

Superficie secondo piano

Area abitabile totale

Arene esterne

Deck in legno

Portico aperto

Veranda

Materiali e finiture

...

...

Caratteristiche aggiuntive

...

Genera preventivo

Figura 5.1 — Wireframe della dashboard Streamlit progettata

La struttura evidenzia:

- una sezione di login iniziale, obbligatoria per accedere al sistema;
- un'area principale di inserimento dati, suddivisa in blocchi collassabili;
- un bottone finale per l'invio dei dati e il calcolo del prezzo stimato.

Una sezione di output, dove viene visualizzato il prezzo predetto, formattato per una lettura immediata.

L'approccio modulare ha permesso di mantenere la dashboard semplice da navigare anche in presenza di un numero elevato di parametri, migliorando la fruibilità e minimizzando il carico cognitivo dell'utente.

6.4 Gestione dell'autenticazione e della sicurezza

La sicurezza dell'accesso alla dashboard e alla API di predizione rappresenta uno dei pilastri fondamentali del sistema sviluppato.

Per garantire un'adeguata protezione dei dati e delle operazioni, è stato implementato un sistema di autenticazione centralizzato basato su OAuth2 utilizzando Keycloak come Identity Provider.

6.4.1 Processo di autenticazione

L'accesso alla dashboard Streamlit è protetto da un meccanismo di login utente.

Alla prima apertura della dashboard, se l'utente non risulta autenticato, viene presentata una finestra di login dedicata, che richiede l'inserimento di username e password.

La sequenza delle operazioni è la seguente:

- Invio delle credenziali a Keycloak: Le credenziali inserite vengono inviate al server Keycloak tramite una richiesta HTTP POST, seguendo il flusso OAuth2 "Resource Owner Password Credentials Grant".
- Ricezione del token di accesso: In caso di autenticazione riuscita, Keycloak restituisce un access token in formato JWT e un refresh token. Questi token vengono salvati nello stato di sessione dell'applicativo Streamlit.

- Utilizzo del token per autenticare le richieste: Ogni richiesta successiva inviata dalla dashboard al backend FastAPI (in particolare la richiesta di predizione) include il token di accesso nell'header HTTP Authorization.
- Gestione della scadenza del token: In caso di scadenza del token, viene automaticamente inviata una richiesta di refresh al server Keycloak, utilizzando il refresh token ottenuto al login, per ottenere un nuovo access token senza richiedere all'utente di autenticarsi nuovamente.
- Logout: È previsto un meccanismo di logout che invalida la sessione e richiede una nuova autenticazione al tentativo successivo di accesso.

6.4.2 Validazione del token lato backend

Dal lato backend, ogni richiesta ricevuta dall'API FastAPI viene autenticata e validata prima di procedere con qualsiasi elaborazione:

- il backend estrae il token JWT dall'header Authorization;
- viene scaricato il set di chiavi pubbliche (JWKS) dal server Keycloak, utilizzando il discovery document OpenID Connect;
- la firma del token viene verificata utilizzando le chiavi pubbliche disponibili e l'algoritmo RS256.

Se il token è valido e non scaduto, la richiesta viene autorizzata e i dati dell'utente (ad esempio, preferred_username) vengono associati all'operazione. In caso di token invalido, assente o scaduto, viene restituito un errore HTTP 401 Unauthorized. Questo approccio garantisce che solo gli utenti autenticati e autorizzati possano accedere alle funzionalità di predizione.

6.4.3 Sicurezza e protezione delle operazioni

In aggiunta alla semplice autenticazione, il sistema prevede ulteriori misure di protezione:

- protezione contro l'accesso non autorizzato: Solo utenti autenticati possono inviare dati per la predizione o ricevere risultati.

- minimizzazione della superficie di attacco: I servizi esposti (FastAPI e Streamlit) sono pubblicati su porte interne private, con accesso limitato tramite il reverse proxy (Nginx, previsto in fase successiva).
- separazione dei privilegi: Le operazioni di predizione e la pubblicazione su Kafka avvengono solo se il token è valido e l'utente è autenticato.

Questo sistema di sicurezza multilivello consente di proteggere l'integrità e la riservatezza delle informazioni, garantendo che il sistema predittivo possa essere utilizzato in contesti operativi reali in totale sicurezza.

6.5 Implementazione tecnica della dashboard

La dashboard è stata realizzata utilizzando Streamlit, un framework Python altamente performante pensato per la creazione rapida di applicazioni web interattive dedicate al Data Science e al Machine Learning. Il codice sorgente della dashboard è contenuto nel file app.py, che struttura il flusso operativo dell'interfaccia in modo modulare e scalabile.

6.5.1 Struttura generale del codice

Il file app.py è organizzato in modo chiaro e ordinato, seguendo questi macro-componenti:

- Gestione dello stato di sessione: Vengono inizializzati variabili di stato come logged_in, token, refresh_token e username, per tracciare lo stato corrente dell'utente e gestire la persistenza dei dati durante la sessione.

Funzioni di login e gestione token

Sono presenti funzioni dedicate per:

- Effettuare l'autenticazione inviando username e password a Keycloak.
- Ottenere e salvare i token di accesso e di refresh.
- Effettuare il refresh automatico dei token scaduti.

- Interfaccia utente modulare: L'inserimento dei dati da parte dell'utente è suddiviso in sezioni collassabili, ognuna gestita da una specifica funzione:
 - Dimensioni dell'immobile (sezione_dimensions)
 - Aree esterne (sezione_aree_esterne)
 - Forma e caratteristiche del lotto (sezione_forma)
 - Stato della costruzione (sezione_stato_costruzione)
 - Bagni e camere (sezione_bagno)
 - Seminterrato (sezione_seminterrato)
 - Informazioni extra (sezione_extra)
 - Garage (sezione_garage)
 - Piscina (sezione_piscina)
 - Caminetto (sezione_caminetto)
 - Caratteristiche del tetto (sezione_roof)

Ogni sezione espone all'utente solo i parametri rilevanti e adatta dinamicamente i campi in funzione delle scelte effettuate, riducendo la possibilità di inconsistenze nell'input.

Qui puoi calcolare il prezzo della casa 🚀

Mostra sezione Dimensioni

Dimensioni ↕

1st Floor SF

0
1000
3000

Has Second Floor

GrLivArea calcolata: 1000 m²

Mostra sezione Aree Esterne

Figura 6.2 — Esempio di sezione collassabile nella dashboard Streamlit

6.5.2 Invio dei dati e calcolo del prezzo

Alla fine del percorso di inserimento, l'utente può cliccare sul pulsante "Calcola Prezzo".

A questo punto avviene il seguente flusso:

- Viene costruito un oggetto JSON con tutti i dati raccolti.
- Viene effettuata una richiesta POST al servizio FastAPI (/predict endpoint), allegando il token di accesso nell'header Authorization.

In caso di risposta positiva, il prezzo stimato viene ricevuto e visualizzato sulla dashboard in modo formattato ed evidenziato. In caso di errore (ad esempio token scaduto), viene tentato automaticamente il refresh e ripetuta la richiesta.

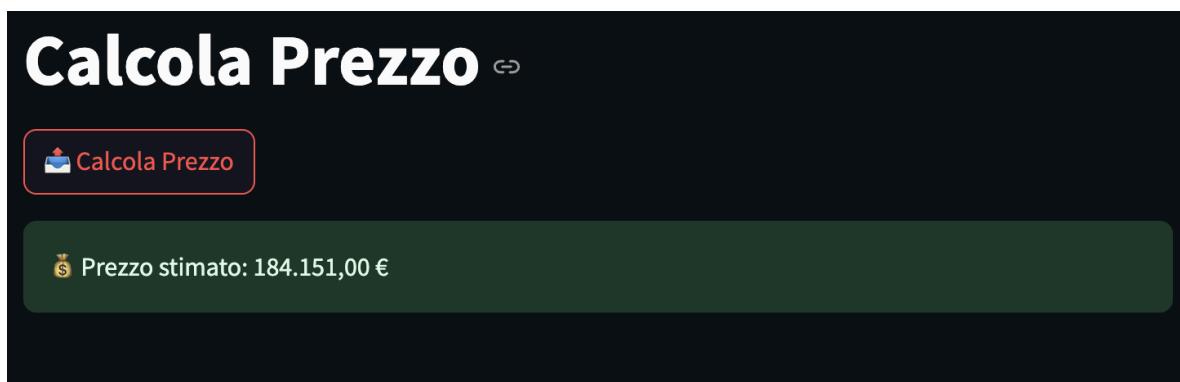


Figura 6.3 — Richiesta di predizione e visualizzazione del prezzo

6.5.3 Logging delle operazioni

Anche lato frontend viene gestita una semplice forma di logging interno:

- In caso di login fallito, viene registrato l'evento.
- In caso di errori di comunicazione con il backend o di fallimento della predizione, vengono forniti messaggi di errore dettagliati all'utente.

Questo aiuta sia in fase di troubleshooting che di miglioramento continuo dell'esperienza utente.

6.5.4 Validazione e consistenza dei dati

Sono stati implementati diversi controlli dinamici per garantire la validità e la congruenza dei dati inseriti:

- Calcolo automatico di variabili derivate (es: GrLivArea, LotArea, TotalSF).
- Vincoli logici tra variabili (es: GarageArea minimo in base a GarageCars).
- Aggiornamento dei valori dipendenti al variare delle opzioni selezionate.

Tali accorgimenti rendono il sistema più robusto e riducono la possibilità di errori di input che potrebbero compromettere la qualità della predizione.

6.6 Architettura dell'integrazione

L'architettura dell'integrazione del sistema è stata progettata per essere scalabile, modulare e containerizzata, adottando strumenti standard di orchestrazione per garantire isolamento, facilità di gestione e riproducibilità dell'ambiente di sviluppo e produzione. Tutti i componenti principali sono stati containerizzati utilizzando Docker e orchestrati tramite Docker Compose, definendo un'infrastruttura di rete dedicata che consente una comunicazione sicura e controllata tra i vari servizi.

6.6.1 Componenti principali

L'ecosistema applicativo è composto dai seguenti servizi principali:

Streamlit Dashboard:

Container responsabile dell'interfaccia utente, accessibile sulla porta 8501.

Permette l'inserimento dei dati e l'invio delle richieste di predizione.

FastAPI Backend ("model-api"):

Container che espone un'API RESTful sulla porta 8000.

Si occupa di:

Validare i token di autenticazione.

Eseguire la predizione del prezzo usando il modello XGBoost.

Inviare i risultati su Kafka per l'analisi successiva.

Keycloak Server:

Identity Provider containerizzato, accessibile sulla porta 8080.

Gestisce l'autenticazione degli utenti e fornisce token OAuth2 in risposta alle richieste di login.

Apache Kafka e Zookeeper:

Sistema di messaggistica distribuita.

Kafka riceve e memorizza i messaggi contenenti i dati delle predizioni, pubblicati dal backend.

Zookeeper gestisce la coordinazione e sincronizzazione dei nodi Kafka.

Kafdrop:

Interfaccia web di monitoring di Kafka, utile per visualizzare in tempo reale i messaggi pubblicati nei topic.

Accessibile sulla porta 9000.

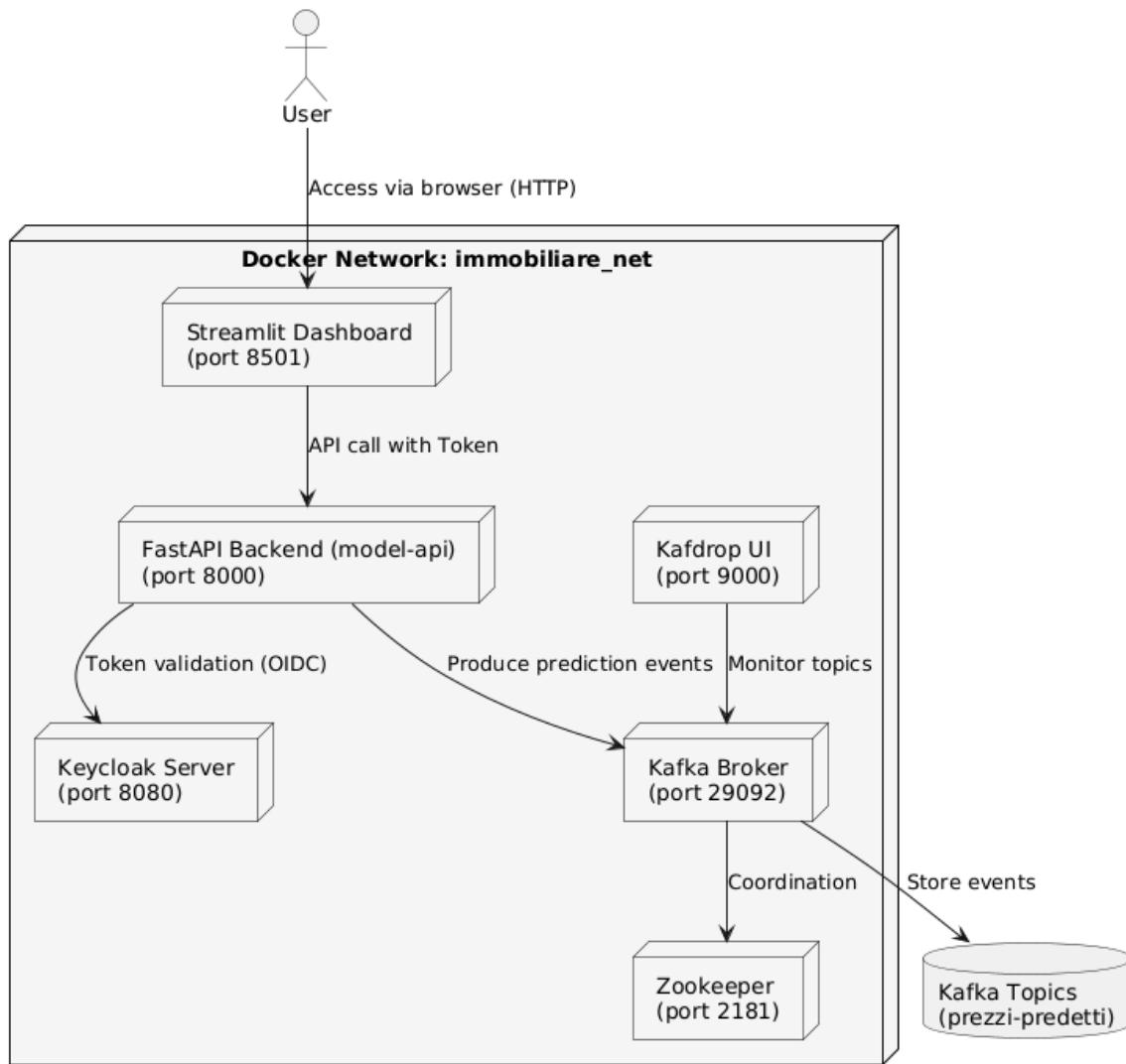


Figura 6.4 — Diagramma di deployment

6.6.2 Comunicazione tra i servizi

La comunicazione tra i vari componenti avviene esclusivamente sulla rete interna `immobiliare_net`, definita in Docker Compose, riducendo così il rischio di esposizione indesiderata all'esterno.

Streamlit → FastAPI:

Comunicazione HTTP REST protetta da Authorization header con Bearer Token OAuth2.

FastAPI → Keycloak:

Recupero della configurazione OpenID e delle chiavi pubbliche JWKS per la verifica dei token JWT.

FastAPI → Kafka:

Produzione di messaggi asincroni su topic Kafka dedicati (prezzi-predetti).

Kafdrop → Kafka:

Visualizzazione e monitoring dei messaggi ricevuti da Kafka in tempo reale.

Ogni servizio è stato configurato per poter dialogare solamente con gli endpoint necessari, minimizzando la superficie d'attacco e migliorando l'isolamento.

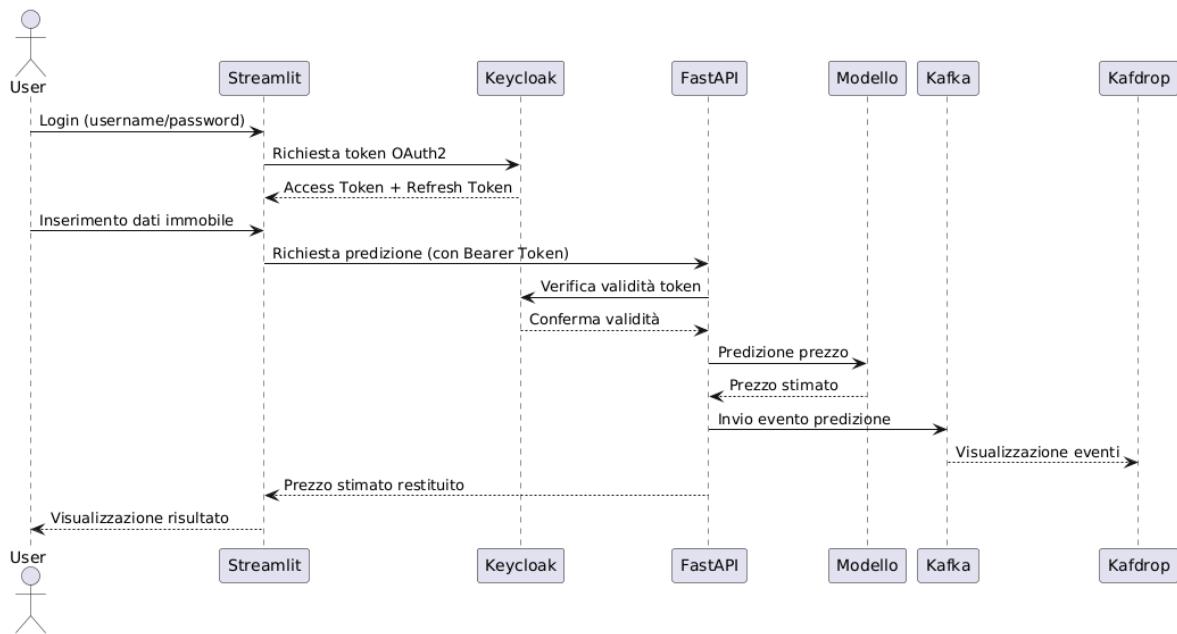


Figura 6.5 — Esempio di flusso di comunicazione tra Streamlit, FastAPI, Keycloak e Kafka.

6.6.3 Configurazione tramite Docker Compose

La configurazione completa dell'infrastruttura è stata realizzata tramite un file `docker-compose.yml`, che definisce:

- I container da avviare.
- Le porte da esporre (limitando l'accessibilità a livello locale quando necessario).
- Le variabili d'ambiente richieste (es. URL di Keycloak, configurazione Kafka).
- I vincoli di dipendenza (`depends_on`) per garantire che i servizi vengano avviati nell'ordine corretto.

Questo approccio garantisce un'elevata portabilità dell'ambiente, che può essere replicato facilmente su qualsiasi sistema compatibile con Docker, migliorando notevolmente la gestione del ciclo di vita del progetto.

6.7 Processi di sviluppo e metodologia agile

Per garantire un approccio sistematico, ordinato ed efficiente allo sviluppo del progetto, è stata adottata una metodologia ispirata ai principi dell'agile software

development. In particolare, si è optato per una gestione delle attività tramite kanban board e per l'adozione di un branching model strutturato per il versionamento del codice. Questa organizzazione ha permesso di suddividere il lavoro in unità modulari, monitorare il progresso in modo trasparente e gestire l'integrazione continua delle funzionalità in modo controllato.

6.7.1 Gestione delle attività tramite Kanban board

Il progetto è stato supportato da una kanban board digitale, suddivisa nelle tradizionali colonne operative:

- To Do: Elenco delle attività ancora da iniziare.
- In Progress: Task attualmente in corso di sviluppo.
- Done: Task completati e pronti per essere eventualmente integrati o verificati.

Ogni feature, correzione o attività tecnica è stata creata come un ticket sulla board, identificabile con un titolo, una descrizione e, ove necessario, criteri di accettazione chiari.

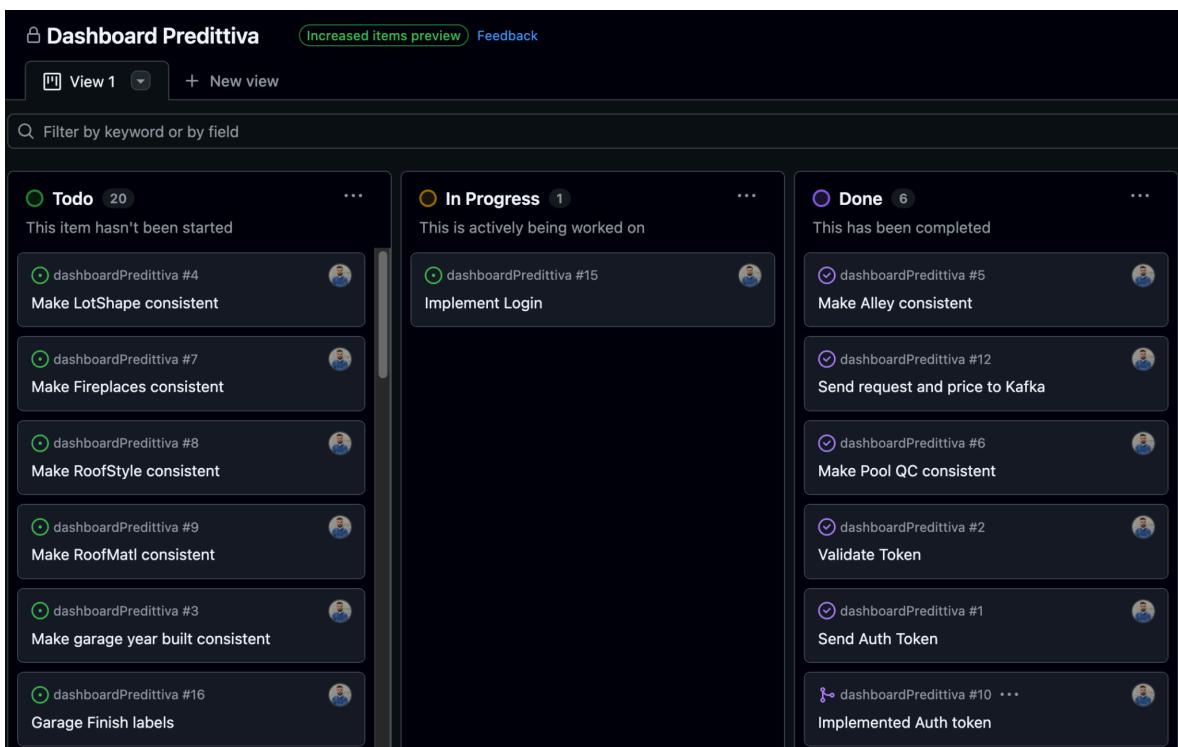


Figura 5.6 — Esempio di Kanban board utilizzata per la gestione delle attività

Questa modalità operativa ha favorito:

- Una visibilità continua sullo stato di avanzamento dei lavori.
- Una prioritizzazione dinamica delle attività.
- Una suddivisione naturale del lavoro in micro-obiettivi facilmente monitorabili.

6.7.2 Branching model del repository Git

Parallelamente alla gestione delle attività, è stato definito un branching model che seguisse una logica di integrazione continua ordinata:

- Branch main: Contiene il codice stabile e pronto per il rilascio in ambienti di produzione.
- Branch develop: Branch di integrazione dove vengono aggregate tutte le nuove funzionalità sviluppate.

- Branch feature specifici: Ogni nuova funzionalità o correzione è sviluppata su un branch dedicato, creato a partire da develop. Il nome del branch richiama l'issue corrispondente, rendendo il flusso di lavoro facilmente tracciabile (esempio: feature/1-send-auth-token).

Questo è anche detto GitFlow ed è possibile notare la struttura generica nella figura in allegato

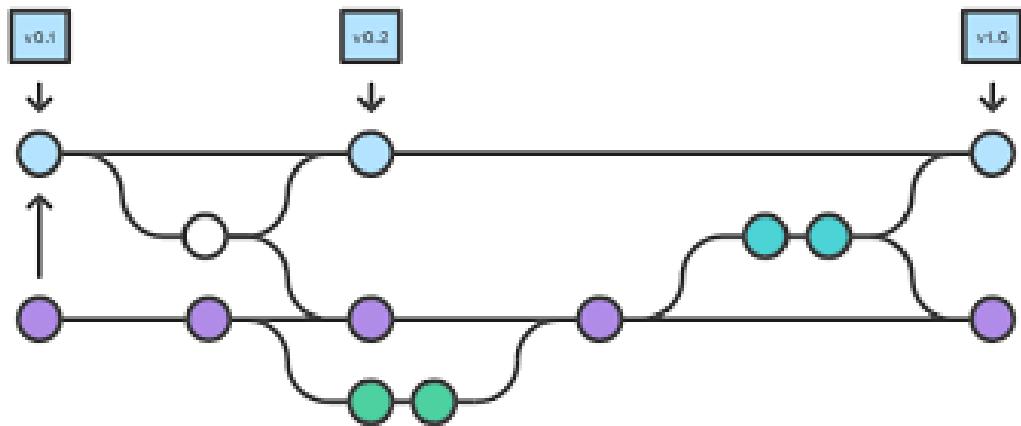


Figura 6.7 - Processo GitFlow

Pull request e review: Una volta completata una feature, viene aperta una pull request verso develop, soggetta a revisione prima del merge. Dopo il merge, il branch di feature viene eliminato per mantenere il repository pulito.

Il screenshot mostra la interfaccia GitHub. Nella barra superiore, si vedono i pulsanti "main" (attivato), "3 Branches" e "0 Tags". A destra della barra, ci sono i pulsanti "Go to file", "Add file" e "Code".

Nella sezione centrale, c'è un campo di ricerca con il placeholder "Find or create a branch...". Sotto di esso, c'è un menu "Switch branches/tags" con un campo di ricerca "Find or create a branch..." e un pulsante "X".

Sotto il campo di ricerca, c'è un menu "Branches" con i seguenti rami: "main" (con un checkmark), "1-send-auth-token" e "develop". Il rami "main" e "develop" sono evidenziati con un bordo blu. Il rami "1-send-auth-token" è evidenziato con un bordo viola.

A destra della lista dei rami, c'è una lista di commit:

Commit	Data	Autore
e8aaaf8c · 5 days ago	23 Commits	
Fixed pool QC	5 days ago	
Fixed construction type	last week	
Fixed some categories view	5 days ago	
Changed docker compose	5 days ago	
First commit	last week	

Figura 6.8 — Esempio di branching model con branch main, develop e feature branches

Questa strategia di branching ha permesso di:

- isolare lo sviluppo di ogni funzionalità evitando conflitti durante l'integrazione.
- facilitare la gestione delle versioni e delle fasi di testing.
- favorire la tracciabilità completa del flusso di sviluppo, dalla creazione dell'issue fino al merge finale.

6.7.3 Benefici dell'approccio adottato

L'approccio combinato di kanban e branching model ha apportato numerosi vantaggi:

- miglioramento della visibilità e controllo sulle attività in corso.
- incremento della modularità nello sviluppo delle funzionalità.
- riduzione del rischio di conflitti di codice o regressioni.
- maggiore agilità nell'adattarsi a necessità o cambiamenti emersi durante il progetto.

L'adozione di pratiche agili ha quindi contribuito in maniera determinante alla qualità complessiva e alla robustezza del sistema sviluppato.

6.8 Esperienza d'uso e scenari applicativi

L'adozione di una dashboard predittiva come quella sviluppata nel presente progetto si inserisce in un contesto più ampio di innovazione digitale nei settori tradizionali, tra cui il comparto immobiliare. Attraverso l'integrazione di strumenti di Machine Learning in una piattaforma accessibile e intuitiva, si favorisce un approccio data-driven alla stima degli immobili, migliorando sia l'operatività quotidiana sia la qualità delle decisioni. In questa sezione vengono descritti gli scenari concreti di utilizzo della dashboard, il flusso tipico di interazione previsto per l'utente finale e i vantaggi operativi derivanti dall'adozione del sistema.

6.8.1 Interazione passo-passo

L'esperienza d'uso della dashboard è stata progettata per essere lineare, progressiva e intuitiva, seguendo un percorso di interazione che guida l'utente attraverso le varie fasi:

- accesso alla dashboard via browser: Non è richiesta alcuna installazione locale; l'utente può accedere semplicemente tramite un URL protetto, previa autenticazione.
- navigazione tra sezioni tematiche espandibili: I dati relativi all'immobile sono organizzati in categorie coerenti (dimensioni, esterni, garage, piscina, ecc.), presentate sotto forma di sezioni collassabili che favoriscono una compilazione graduale e senza sovraccarico cognitivo.
- inserimento assistito dei dati: Ogni campo è dotato di controlli dinamici e valori predefiniti per garantire consistenza semantica e velocizzare la compilazione.
- attivazione del modello predittivo: Tramite il pulsante “Calcola Prezzo”, l'utente può inviare i dati raccolti per ottenere la stima del valore dell'immobile.
- visualizzazione immediata del risultato: Il prezzo predetto viene mostrato in modo formattato e chiaro, con la possibilità di salvare o confrontare i risultati ottenuti.

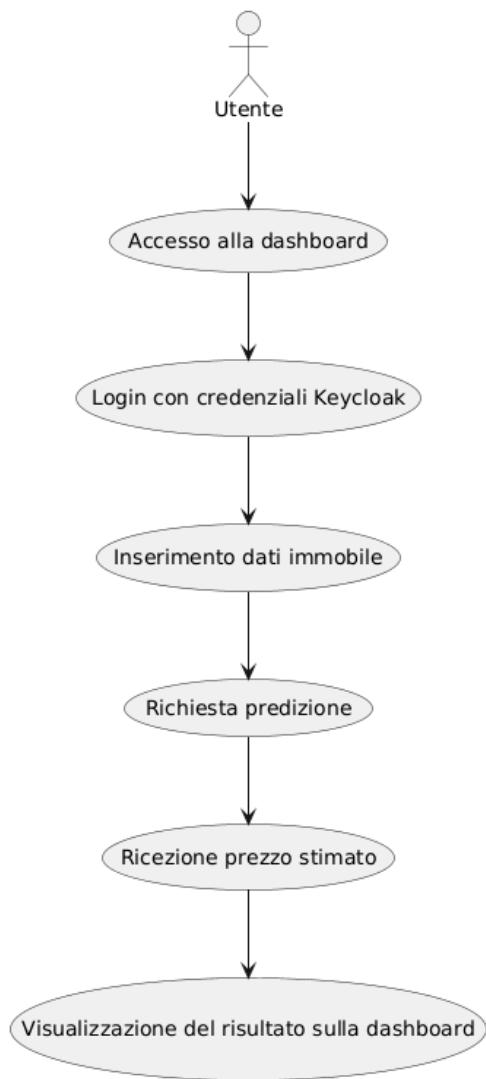


Figura 6.9 — Flusso di interazione utente nella dashboard

La modularità dell'interfaccia consente così di lavorare in maniera progressiva e focalizzata, riducendo la complessità percepita anche nei casi di immobili particolarmente articolati.

6.8.2 Vantaggi per l'agente immobiliare

La dashboard fornisce un supporto operativo concreto per agenti immobiliari, consulenti, investitori e più in generale professionisti del settore immobiliare, migliorando le seguenti attività:

- Valutazione preliminare: Permette una prima stima oggettiva e rapida del valore di un immobile in fase di acquisizione o di vendita.
- Comparazione tra unità: Facilita il confronto tra diverse unità immobiliari, utilizzando parametri standardizzati e un modello predittivo unificato.
- Consulenza trasparente verso il cliente: Consente di motivare la stima proposta con dati concreti e modelli verificabili, aumentando la fiducia del cliente nel processo di valutazione.

Supporto alla strategia di pricing:

Rende possibile simulare scenari alternativi (ad esempio, miglioramenti strutturali o estetici) e valutarne l'impatto sul valore stimato. In questo modo, la dashboard si configura non solo come uno strumento tecnico, ma come un vero alleato professionale, in grado di incrementare la competitività e l'efficienza operativa degli utenti.

6.8.3 Simulazioni e analisi "what-if"

Una delle funzionalità chiave della dashboard è la possibilità di effettuare analisi what-if, ovvero simulazioni di scenario per esplorare l'impatto di modifiche ipotetiche sulle caratteristiche dell'immobile, osservando in tempo reale l'effetto sulle previsioni di prezzo. Grazie all'aggiornamento istantaneo del valore predetto in base ai nuovi input, l'utente può esplorare molteplici scenari decisionali, supportati da un sistema predittivo avanzato.

Esempio pratico: aggiunta di un caminetto

Nella simulazione che segue, l'utente effettua una prima stima senza considerare la presenza di un caminetto. Il valore predetto per l'immobile risulta pari a 184.151,00\$.

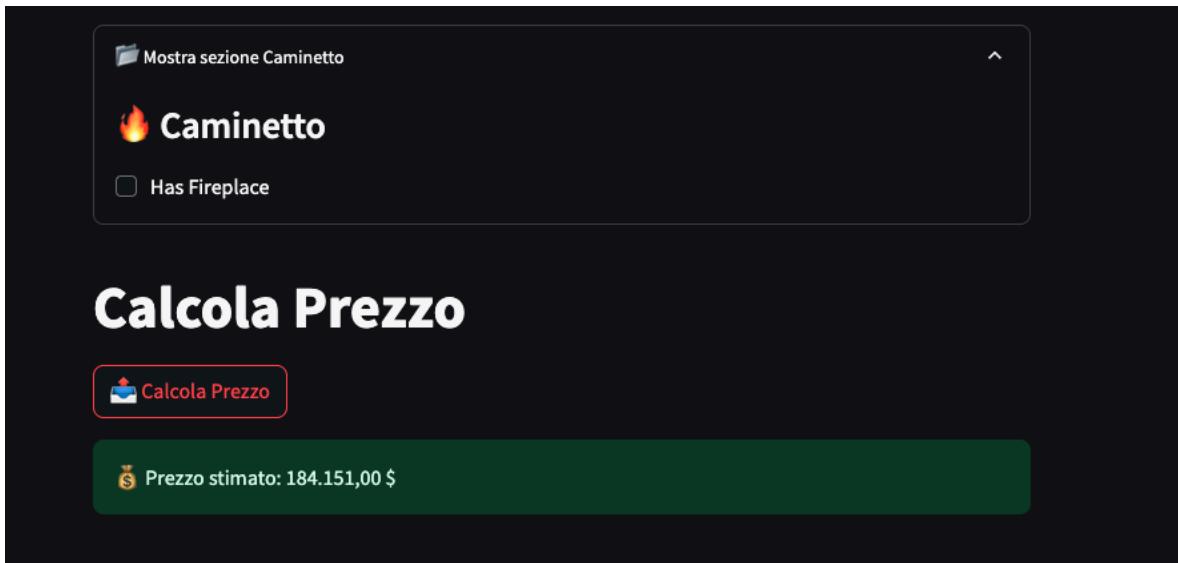


Figura 6.10a — Stima iniziale senza caminetto

Successivamente, l'utente modifica l'input selezionando la presenza di un caminetto di qualità eccellente (Fireplace Quality = Excellent). La dashboard aggiorna dinamicamente il prezzo, che sale a 209.011,41\$, mostrando l'effetto positivo della nuova caratteristica.

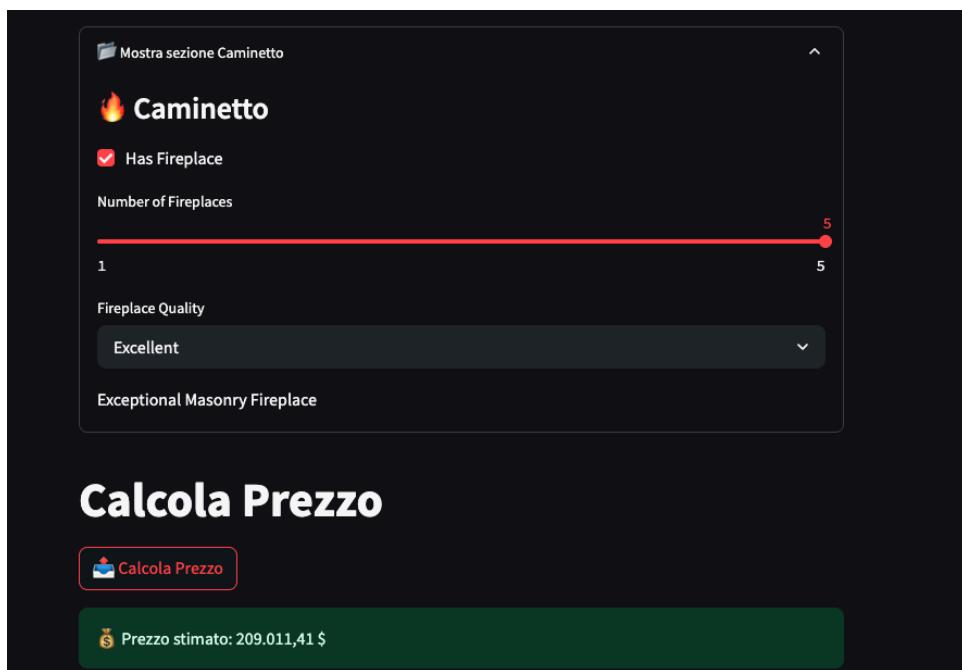


Figura 6.10b — Stima aggiornata con caminetto di alta qualità

6.9 Conclusione del capitolo

In questo capitolo è stato descritto l'intero processo di progettazione, implementazione e integrazione della dashboard predittiva per la stima del prezzo degli immobili. L'obiettivo principale — rendere il sistema accessibile anche a utenti non tecnici — è stato perseguito attraverso scelte architettoniche e progettuali orientate alla semplicità, alla modularità e alla coerenza semantica dei dati. Dal punto di vista dell'interfaccia utente, la dashboard offre un'esperienza guidata e validata, suddivisa in sezioni tematiche collassabili, che consente l'inserimento assistito dei parametri immobiliari, riducendo il rischio di errori e aumentando la fluidità dell'interazione. Particolare attenzione è stata rivolta all'ergonomia cognitiva: vincoli dinamici, calcoli automatici e feedback in tempo reale permettono all'utente di mantenere sempre il controllo sul processo di input. Dal punto di vista tecnico, l'adozione di un'architettura a microservizi, con separazione tra frontend (Streamlit) e backend (FastAPI), ha consentito una progettazione scalabile, manutenibile e sicura. Il backend non solo si occupa delle predizioni, ma integra anche un sistema completo di gestione della sicurezza tramite token OAuth2 validati con Keycloak, oltre a registrare in modo asincrono ogni evento di predizione su Kafka, favorendo future analisi batch. La containerizzazione dei vari componenti tramite Docker Compose ha reso l'intero sistema facilmente distribuibile e replicabile, sia in contesti produttivi sia in ambienti didattici o dimostrativi. Infine, dal punto di vista metodologico, l'adozione di pratiche di sviluppo agile — attraverso l'utilizzo di una kanban board, la gestione delle issue tramite branching dedicati e l'integrazione controllata via pull request — ha contribuito a mantenere un processo di sviluppo ordinato, tracciabile e orientato al miglioramento continuo. Nel suo stato attuale, la dashboard rappresenta un prototipo funzionante e completo, capace di dimostrare l'efficacia dell'integrazione tra modelli predittivi avanzati e strumenti interattivi di supporto decisionale nel contesto immobiliare. Il sistema getta così le basi per futuri sviluppi, estensioni e applicazioni reali nel mercato.

Capitolo 7 - Sviluppi futuri e possibili estensioni

7.1 Estensione delle fonti dati

Uno sviluppo naturale del progetto riguarda l'estensione del dataset ad ambiti geografici diversi, come il contesto italiano. Questo richiederebbe:

- l'integrazione di fonti aperte (es. portali come OMI, catastali o regionali);
- un adattamento del preprocessing alle nuove caratteristiche dei dati (mancanza di uniformità, disponibilità parziale, formati eterogenei);
- un aggiornamento del dizionario delle variabili.

7.1.1 Raccolta dati tramite scraping web

Per rendere operativo questo ampliamento, si ipotizza l'adozione di tecniche di web scraping controllato per estrarre dati strutturati da portali immobiliari italiani, quali:

- immobiliare.it
- idealista.it
- casa.it
- portali regionali (es. Borsino Immobiliare, Subito Casa, Bakeca)

L'obiettivo del web scraping sarebbe quello di ottenere:

- prezzi richiesti per immobili in vendita;
- superfici interne e pertinenze (garage, giardini, terrazzi);
- anno di costruzione, stato conservativo;
- descrizioni testuali degli annunci.

Un'ulteriore integrazione riguarderebbe il collegamento con dati esterni:

- prezzi medi ufficiali (es. database OMI);
- classificazioni energetiche;
- indici di qualità urbana o geolocalizzazione.

L'adozione dello scraping, pur rispettando le normative sui dati e l'uso etico delle informazioni, consentirebbe di costruire dataset aggiornati e mirati, più rappresentativi delle dinamiche del mercato italiano.

7.1.2 Creazione di un modello predittivo specifico per l'Italia

Con i dati raccolti, sarebbe possibile avviare un nuovo processo di training per un modello predittivo dedicato agli immobili italiani, tenendo conto delle specificità nazionali, quali:

- tipologia immobiliare (appartamento, villa, casa indipendente, rustico);
- zone urbane, semi-urbane e rurali molto differenziate;
- rilevanza della classe energetica (importante in Italia post normative UE);
- finiture interne tipiche (marmo, parquet, ceramica);
- vincoli storici (es. immobili in centro storico o vincolati da soprintendenze).

Il preprocessing dovrebbe quindi includere:

- normalizzazione delle superfici (superficie commerciale vs superficie calpestabile);
- creazione di feature ingegnerizzate ad hoc (es. distanza da servizi pubblici, accessibilità);
- gestione delle diverse unità di misura e standardizzazioni catastali.

L'addestramento di un modello specifico migliorerebbe la precisione delle predizioni e renderebbe la dashboard realmente fruibile in un contesto italiano contemporaneo.

7.2 Introduzione dell’NLP per analisi testuale

Come anticipato nelle sezioni precedenti, l’uso del Natural Language Processing (NLP) può consentire l’estrazione automatica di variabili strutturate da descrizioni testuali degli annunci immobiliari. In particolare, è ipotizzabile:

- addestrare modelli di classificazione o sequence tagging per rilevare presenza di piscina, qualità degli interni, esposizione solare, ecc.;
- generare in automatico i valori per feature booleani o ordinali (es. haspool, FireplaceQu, OverallQual);
- applicare modelli pre-addestrati su domini immobiliari (es. BERT fine-tuned su descrizioni residenziali).

7.3 Integrazione della explainable AI (XAI)

Per rafforzare la fiducia dell'utente nel sistema, un'ulteriore estensione è rappresentata dall'adozione di tecniche di spiegabilità dei modelli:

- utilizzo di metodi come SHAP per visualizzare l'impatto di ogni variabile sulla predizione;
- aggiunta alla dashboard di un pannello che illustri il peso relativo di ogni input nella stima finale;
- integrazione con linee guida europee (es. EU AI Act) per trasparenza, spiegabilità e auditabilità dell'AI.

7.3.1 Importanza normativa e contesto europeo

La crescente attenzione verso l'intelligenza artificiale trasparente e interpretabile non costituisce più soltanto una best practice tecnologica, ma sta assumendo una rilevanza normativa a livello europeo e internazionale.

Il Regolamento Europeo sull'Intelligenza Artificiale (AI Act), attualmente in fase avanzata di approvazione, introduce principi chiave come la spiegabilità, la trasparenza e la responsabilità dei sistemi basati su AI, soprattutto nei casi di utilizzo in ambiti ad alto impatto sociale.

Sebbene la stima del prezzo di un immobile non sia classificata tra le applicazioni "ad alto rischio" dal testo normativo attuale, è evidente che anche per sistemi predittivi in ambiti economici e decisionali si sta affermando l'esigenza di fornire motivazioni intelligenziali e accessibili delle predizioni generate.

In quest'ottica, l'adozione futura di strumenti di Explainable AI (XAI), come l'analisi basata su valori SHAP, rappresenta un allineamento virtuoso e lungimirante.

Oltre a rafforzare la fiducia dell'utente verso il sistema predittivo, la spiegabilità contribuisce a rendere il modello più responsabile, auditabile e in linea con i principi di etica algoritmica emergenti.

L'integrazione di tecniche XAI nel progetto, già prevista come sviluppo futuro, non solo migliorerà la qualità dell'esperienza utente, ma posizionerà il sistema in un contesto normativo e sociale in rapida evoluzione, rispondendo proattivamente ai requisiti di trasparenza, equità e responsabilità dell'intelligenza artificiale di nuova generazione.

7.4 Ottimizzazione della UI/UX

La dashboard sviluppata è già pensata per utenti non tecnici, ma può essere ulteriormente migliorata attraverso:

- test di usabilità con stakeholder reali;
- personalizzazione dinamica dell'interfaccia in base al tipo di utente (es. agenzia vs cliente privato);
- suggerimenti intelligenti sui parametri non forniti, basati su valori mediani o popolari.

7.5 Espansione cloud e API pubbliche

Un'ulteriore direzione riguarda la scalabilità e l'accessibilità:

- deploy dell'API e della dashboard su ambienti cloud (es. Heroku, AWS, Azure);
- utilizzo di database remoti e storage persistente per log e cronologia predizioni;
- pubblicazione dell'API predittiva come servizio esterno, documentato con Swagger/OpenAPI.

Tabella riassuntiva

Area di Sviluppo	Descrizione	Impatto Atteso
------------------	-------------	----------------

Estensione delle fonti dati	Integrazione di dati da portali immobiliari italiani tramite scraping controllato (Immobiliare.it, Idealista, OMI)	Migliorare la rappresentatività geografica e l'accuratezza del modello
Costruzione modello Italia-specifico	Training di un modello predittivo ottimizzato per le caratteristiche del mercato immobiliare italiano	Maggiore precisione nelle stime e adattabilità regionale
Introduzione del NLP	Estrazione automatica di feature strutturate da descrizioni testuali degli annunci	Arricchimento del dataset e incremento della granularità predittiva
Explainable AI (XAI)	Adozione di tecniche come SHAP per spiegare l'influenza delle feature sulle predizioni	Aumento della trasparenza e della fiducia dell'utente
Ottimizzazione della UI/UX	Miglioramento dell'interfaccia tramite test di usabilità e personalizzazione dinamica	Esperienza utente più fluida e adattativa
Espansione cloud e API pubbliche	Deployment dell'infrastruttura su ambienti cloud e apertura di API REST pubbliche	Scalabilità, accessibilità remota e integrazione in sistemi terzi

Tabella 7.1 — Sviluppi futuri previsti: aree di intervento, descrizione e impatto atteso.

Per illustrare concretamente il flusso operativo previsto, si propone un caso d'uso ipotetico che simula l'interazione di un agente immobiliare con la dashboard predittiva sviluppata.

Scenario

Un agente immobiliare riceve l'incarico di valutare un appartamento di circa 1800 piedi quadrati, situato in una zona residenziale di livello medio.

L'obiettivo è quello di proporre al cliente un prezzo di vendita competitivo, supportato da dati oggettivi e modelli predittivi affidabili.

Flusso operativo

L'agente accede alla dashboard Streamlit attraverso un login sicuro, autenticandosi tramite il sistema OAuth2 gestito da Keycloak.

Una volta autenticato, procede con l'inserimento delle caratteristiche principali dell'immobile:

- **GrLivArea**: 1800 piedi quadrati (superficie abitabile);
- **OverallQual**: 6 (qualità complessiva superiore alla media);
- **GarageArea**: 400 piedi quadrati (superficie del garage);
- **TotalBathrooms**: 2.5 bagni;
- **HasFireplace**: Presenza di caminetto confermata;

Oltre ai parametri principali, l'agente completa la compilazione indicando ulteriori caratteristiche accessorie, come la presenza di piscina, seminterrato e lo stato conservativo generale dell'abitazione.

Una volta completato l'inserimento, l'agente utilizza il pulsante " Calcola Prezzo" per inoltrare i dati all'API backend FastAPI.

Il sistema elabora le informazioni e restituisce in tempo reale una stima del prezzo di vendita, pari a circa **245.000 dollari**.

A partire da questa prima valutazione, l'agente ha la possibilità di esplorare scenari alternativi ("what-if"), simulando ad esempio l'effetto di una ristrutturazione della cucina o un miglioramento della qualità complessiva dell'immobile da 6 a 8.

Ogni modifica ai parametri genera una nuova predizione aggiornata, consentendo così un'analisi rapida dell'impatto delle possibili migliorie.

Infine, sulla base delle simulazioni effettuate e dei dati oggettivi ottenuti, l'agente propone al venditore un prezzo di listino competitivo e supportato da evidenze numeriche, rafforzando la fiducia del cliente nel processo decisionale.

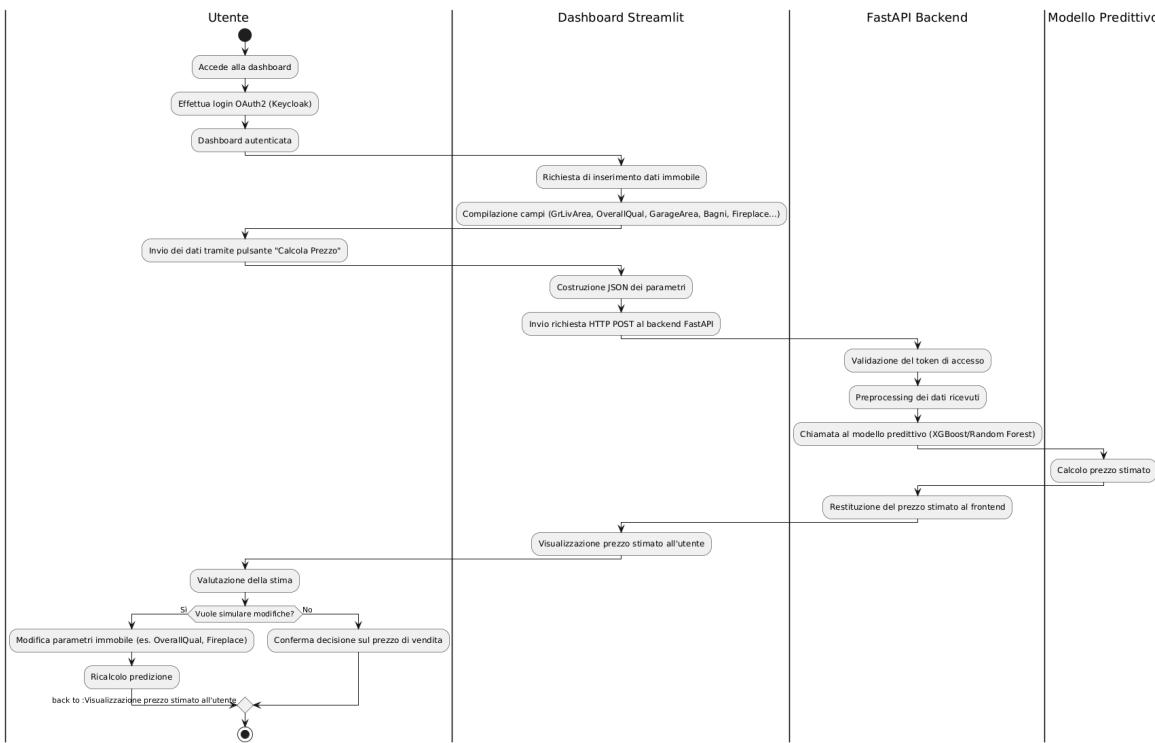


Figura 7.1 — Activity Diagram: flusso operativo di utilizzo della dashboard per la stima del prezzo immobiliare

Il diagramma di attività mostra le principali fasi operative attraversate dall'utente nella procedura di predizione, dalla fase di autenticazione iniziale fino alla simulazione di scenari alternativi e alla decisione finale.

7.6 Conclusione del capitolo

Il lavoro svolto ha gettato le basi per una soluzione concreta, modulare ed estendibile. Le possibili evoluzioni spaziano dalla raffinazione tecnica del modello all'ampliamento delle fonti dati e delle capacità predittive.

L'integrazione di fonti italiane, l'adozione di tecniche NLP per l'analisi testuale, la costruzione di modelli predittivi geolocalizzati, e l'apertura verso il cloud computing e le API pubbliche rappresentano linee guida solide per trasformare il prototipo in uno strumento operativo, scalabile e centrato sull'utente. L'obiettivo di uno strumento utile, interpretabile e utilizzabile nel mondo reale resta al centro degli

sviluppi futuri, in linea con i principi della Data Science applicata e dell'intelligenza artificiale centrata sull'uomo.

Capitolo 8 - Conclusioni

8.1 Riepilogo del percorso

Il presente lavoro nasce dall’obiettivo di applicare i principi della Data Science al settore immobiliare, un ambito che, pur essendo ricco di dati strutturati, risulta ancora fortemente legato a valutazioni soggettive ed esperienziali. Attraverso un approccio end-to-end, è stato sviluppato un sistema predittivo in grado di stimare il prezzo di vendita di un immobile a partire da caratteristiche strutturali e qualitative. Il progetto ha attraversato tutte le fasi della pipeline di data science: dall’acquisizione e pulizia del dataset, alla modellazione predittiva, fino alla realizzazione di un’interfaccia interattiva accessibile anche a utenti non tecnici. Ogni fase è stata affrontata con strumenti e metodologie aggiornate, combinando rigore analitico e attenzione all’usabilità.

8.2 Risultati principali

Il sistema sviluppato ha portato a risultati concreti e significativi:

- è stato identificato il modello Random Forest come il migliore in termini di equilibrio tra accuratezza ($R^2 \approx 0.87$), robustezza e tempi di inferenza;
- è stata realizzata una dashboard interattiva, accessibile da browser, che consente all’utente di simulare e confrontare il valore degli immobili modificando dinamicamente i parametri di input;
- l’intera architettura è stata containerizzata con Docker e orchestrata tramite docker-compose, garantendo replicabilità e semplicità di deployment.

Il prototipo realizzato dimostra che è possibile costruire strumenti intelligenti, accessibili e trasparenti per supportare processi decisionali anche in settori ancora parzialmente digitalizzati.

8.3 Impatto applicativo e potenzialità

Il valore di questo lavoro non risiede solo nell'accuratezza del modello, ma nella capacità di aver creato uno strumento utilizzabile in ambiti reali. La dashboard può essere adottata da agenzie immobiliari, consulenti del settore o persino utenti privati, offrendo un punto di partenza per stime rapide, simulate o documentate. Dal punto di vista tecnico, il progetto è pronto per essere esteso a nuove aree geografiche, arricchito con fonti di dati locali (catasto, portali immobiliari, OMI), o migliorato in chiave cloud-native. La modularità del codice e il versionamento su GitHub ne facilitano la manutenzione e lo sviluppo collaborativo.

8.4 Limiti del lavoro

Nel presente paragrafo si analizzano in dettaglio i principali limiti e rischi associati al sistema sviluppato, evidenziando le aree di miglioramento e tracciando alcune possibili linee evolutive per il futuro perfezionamento del progetto.

Come ogni progetto, anche questo presenta limiti intrinseci, riconosciuti e analizzati criticamente.

Generalizzazione geografica

Il dataset utilizzato si riferisce esclusivamente alla città di Ames, Iowa, e potrebbe non essere rappresentativo di altri contesti geografici, in particolare quello italiano. Il mercato immobiliare è altamente localizzato, e variabili come posizione, servizi di zona, dinamiche socio-economiche possono influenzare significativamente il valore di un immobile.

Assenza di alcune dimensioni rilevanti

Alcune caratteristiche fondamentali per una stima immobiliare accurata — come la geolocalizzazione precisa, l'accessibilità ai trasporti pubblici o l'andamento storico dei prezzi nella zona — non sono state incluse nel modello attuale, limitando parzialmente la granularità della predizione.

Limitata interpretabilità del modello

Pur avendo discusso della necessità di sistemi di Explainable AI (XAI) e dell'utilità dei valori SHAP, il modello finale non integra ancora tecniche di spiegabilità locale o globale, restituendo all'utente solo la stima finale del prezzo.

Qualità dei dati di input

Poiché il sistema si basa sull'inserimento manuale dei parametri da parte dell'utente, errori di input o incompletezze possono compromettere l'accuratezza della predizione. Nonostante siano stati implementati controlli dinamici di coerenza, esiste comunque un margine di errore umano.

Dipendenza da infrastrutture esterne

L'autenticazione basata su Keycloak introduce una dipendenza da un sistema terzo. Eventuali malfunzionamenti o downtime di Keycloak potrebbero impedire l'accesso alla dashboard, influendo sull'affidabilità operativa del sistema.

Adattabilità temporale limitata

Il modello è stato addestrato su dati storici statici. Non è stato ancora implementato un meccanismo di retraining periodico automatico, pertanto, in presenza di mutamenti significativi nelle condizioni del mercato immobiliare, l'accuratezza del sistema potrebbe degradare nel tempo.

Tali limitazioni, pur evidenti, non riducono il valore del lavoro svolto.

Al contrario, esse forniscono spunti concreti per possibili evoluzioni future, come:

- l'estensione geografica del dataset;
- l'integrazione di NLP per l'analisi delle descrizioni immobiliari;
- l'adozione di modelli spiegabili tramite XAI;
- la progettazione di meccanismi di aggiornamento continuo dei modelli predittivi.

8.5 Considerazioni finali

In un momento storico in cui l'Intelligenza Artificiale è sempre più presente nei processi decisionali, questo progetto ha voluto proporre un modello di utilizzo etico,

trasparente e centrato sull'uomo. L'utente, grazie alla dashboard, resta al centro dell'esperienza: può manipolare i parametri, comprendere il funzionamento del sistema e ottenere risposte rapide, ma non opache. La tesi qui presentata rappresenta quindi non solo un'esercitazione didattica, ma anche un piccolo contributo alla digitalizzazione del settore immobiliare, in linea con i principi della trasformazione digitale, della Data Science applicata e dell'AI Human-Centered.

Bibliografia

- Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow*. O'Reilly Media.
 - Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning*. Springer.
 - Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), 5–32.
 - Lundberg, S. M., & Lee, S. I. (2017). A Unified Approach to Interpreting Model Predictions. *Advances in Neural Information Processing Systems (NeurIPS)*.
 - European Commission. (2021). *Proposal for a Regulation on Artificial Intelligence (EU AI Act)*.
 - Di Grazia, G. (2022). Strumenti e metodi per il trading algoritmico [Tesi di laurea triennale, Università degli Studi di Palermo].
-

Sitografia

- Repository contenente il source code del progetto
<https://github.com/gaetanodigrazia/dashboardPredittiva>
- House Prices – Advanced Regression Techniques, Kaggle
<https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques>

- Streamlit – Official website
<https://streamlit.io>
- FastAPI – Documentation
<https://fastapi.tiangolo.com>
- Docker – Documentation
<https://docs.docker.com>
- EU Artificial Intelligence Act
<https://artificial-intelligence-act.eu>
- GitHub – Source versioning
<https://github.com>

Appendice - Approfondimenti Tecnici e Metodologici

Introduzione all'Appendice A

L'Appendice A raccoglie una serie di approfondimenti tecnici, metodologici e progettuali a supporto del sistema predittivo descritto nella tesi. Gli argomenti trattati spaziano dalla gestione dell'autenticazione sicura tramite OAuth2 e JWT, alla configurazione dell'infrastruttura containerizzata, fino alle tecniche di validazione dei modelli, alle metriche di valutazione e ai principi di ergonomia cognitiva applicati alla progettazione della dashboard.

I riferimenti bibliografici inclusi offrono ulteriore supporto teorico e metodologico alle scelte effettuate.

A.1 Sicurezza dell'autenticazione: OAuth2, JWT, Token e Bearer Authentication

In un sistema predittivo accessibile tramite rete, la gestione sicura dell'autenticazione e dell'autorizzazione rappresenta un requisito fondamentale. Nel progetto sviluppato, è stato implementato un sistema basato su OAuth2 e JWT (JSON Web Token), in combinazione con l'uso di Bearer Authentication.

A.1.1 OAuth2: standard di autorizzazione

OAuth2 è un protocollo di autorizzazione ampiamente adottato che consente a un'applicazione client di accedere a risorse protette delegando l'autenticazione a un server esterno, detto Identity Provider. Nel progetto, Keycloak è stato utilizzato come server di autenticazione e autorizzazione, implementando il flusso Resource Owner Password Credentials. Questo approccio consente di separare le credenziali utente dal sistema applicativo, migliorando la sicurezza generale.

A.1.2 JWT (JSON Web Token)

JWT è un formato compatto, sicuro e standardizzato per rappresentare informazioni tra parti come token firmati digitalmente. Un JWT è composto da tre parti:

- Header — specifica il tipo di token e l'algoritmo di firma (es. RS256);
- Payload — contiene i dati (claims), come l'identificativo dell'utente e la scadenza del token;
- Signature — garantisce l'integrità del token.

Nel sistema, i JWT sono firmati usando chiavi asimmetriche (RSA) e verificati dinamicamente dal backend FastAPI tramite il set di chiavi JWKS di Keycloak.

A.1.3 Access Token e Refresh Token

Per gestire la durata della sessione utente sono utilizzati:

- Access Token: Breve durata (es. 5-15 minuti), utilizzato per autenticare ogni richiesta HTTP verso il backend;
- Refresh Token: Durata più lunga, consente di ottenere nuovi access token senza richiedere una nuova autenticazione manuale.

Il processo di gestione automatica dei token migliora la sicurezza, riduce l'impatto della scadenza degli access token e ottimizza l'esperienza utente.

A.1.4 Bearer Authentication

Le richieste protette vengono inviate usando il meccanismo di Bearer Authentication, ovvero il backend, prima di eseguire qualsiasi operazione, valida il token che ricevuto nel parametro Authorization della richiesta. In caso di token assente, scaduto o non valido, viene restituito un errore HTTP 401 Unauthorized.

A.2 Messaging asincrono: Apache Kafka e Zookeeper

Parallelamente al sistema di autenticazione, il progetto integra una pipeline di messaggistica asincrona tramite Apache Kafka, assistita da Apache Zookeeper.

A.2.1 Apache Kafka: messaggistica scalabile

Apache Kafka è una piattaforma distribuita progettata per la pubblicazione, sottoscrizione, memorizzazione e processamento di flussi di dati in tempo reale.

Nel progetto:

- Ogni predizione effettuata dal sistema viene serializzata come messaggio JSON;
- I messaggi vengono prodotti sul topic prezzi-predetti;
- Il sistema mantiene la cronologia delle predizioni, favorendo future analisi di tipo batch (monitoraggio, auditing, analisi di mercato).

Questo approccio abilita un'architettura più resiliente e scalabile.

A.2.2 Topics Kafka

In Kafka, i dati sono organizzati in topics. Ogni topic è un canale di comunicazione su cui i produttori (producers) scrivono i messaggi e i consumatori (consumers) li leggono.

Nel progetto:

- Producer: il backend FastAPI invia eventi di predizione;
- Topic: prezzi-predetti;
- Consumer (futuro): sarà possibile sviluppare strumenti di analisi offline o dashboard secondarie che analizzano i dati storici.

A.2.3 Apache Zookeeper: coordinazione dei servizi

Zookeeper è utilizzato da Kafka per la gestione dei metadati e la coordinazione dei nodi nel cluster.

Nel progetto:

- Mantiene la configurazione del cluster Kafka;
- Coordina l'elezione dei leader tra i broker Kafka.

Fornisce un registro distribuito affidabile e consistente.

L'integrazione di Zookeeper è fondamentale per garantire la disponibilità e la resilienza del servizio di messaggistica.

A.3 Architettura dei container Docker

Il sistema è stato containerizzato con due servizi principali:

- model-api: backend sviluppato in FastAPI, esposto sulla porta 8000, responsabile dell'elaborazione dei dati e della generazione della previsione del prezzo;
- dashboard: interfaccia utente realizzata con Streamlit, esposta sulla porta 8501, tramite la quale l'utente può inserire i parametri e visualizzare il prezzo stimato.

Entrambi i container condividono un volume Docker, utilizzato per accedere al file del modello serializzato (model.pkl) e ad altri artefatti utili alla predizione.

A.4 Codice di logging

Per garantire la tracciabilità delle richieste e delle previsioni effettuate, è stato implementato un semplice sistema di logging che salva su file ogni interazione utente. Di seguito un estratto del codice utilizzato:

```
[ ]: with open("log.txt", "a") as f:  
    f.write(f"Input: {json.dumps(input_dict)}\n")  
    f.write(f"Output: {prediction}\n\n")
```

Figura A.4.1 — Estratto di codice per il sistema di logging delle richieste e predizioni.

Questo approccio consente di monitorare il comportamento del sistema, effettuare controlli ex post e raccogliere dati per future ottimizzazioni.

A.5 Repository del progetto

Il codice sorgente del progetto è stato versionato tramite Git e pubblicato su un repository GitHub privato, strutturato secondo le buone pratiche di sviluppo collaborativo. Il repository include:

- i file Python relativi alla dashboard Streamlit e al backend FastAPI;
- gli script per la preparazione dei dati, l'addestramento del modello e la generazione del file .pkl;

- i file di configurazione Docker (Dockerfile, docker-compose.yml);
- il mapping tra categorie e codifiche numeriche, salvato in formato JSON.

Questa struttura modulare e versionata garantisce facilità di manutenzione, replicabilità del lavoro e possibilità di estensione futura da parte di altri sviluppatori o ricercatori.

A.6 Flusso completo di autenticazione OAuth2

Il sistema di autenticazione implementato si basa sul protocollo OAuth2 Resource Owner Password Credentials Grant e gestisce in modo completo tutte le fasi principali: login, autorizzazione, rinnovo dei token e logout dell'utente.

Il flusso operativo si articola nei seguenti passaggi:

Durante la fase di login iniziale, l'utente inserisce le proprie credenziali (username e password) direttamente nell'interfaccia della dashboard Streamlit. Quest'ultima invia una richiesta HTTP POST al server Keycloak, specificando come grant type password, al fine di ottenere le credenziali di accesso. In risposta, Keycloak fornisce un Access Token (in formato JWT) e un Refresh Token, che vengono memorizzati nella sessione utente di Streamlit. Successivamente, durante l'utilizzo ordinario dell'applicativo, Streamlit allega automaticamente l'Access Token all'interno dell'header Authorization di ciascuna richiesta REST inviata al backend FastAPI. Dal canto suo, FastAPI verifica la validità di ogni token ricevuto, confrontando la firma con il set di chiavi pubbliche (JWKS) pubblicato da Keycloak, al fine di garantire l'autenticità e la sicurezza delle richieste. Nel caso in cui l'Access Token risulti scaduto, la dashboard è in grado di gestire autonomamente il rinnovo della sessione: utilizza il Refresh Token per richiedere un nuovo Access Token, senza che l'utente debba reinserire manualmente le credenziali o interrompere il flusso operativo.

Infine, nella fase di logout, l'applicazione provvede a invalidare i token memorizzati e a ripristinare lo stato iniziale della sessione, richiedendo una nuova autenticazione per eventuali accessi successivi.

Il flusso completo di autenticazione e autorizzazione è sintetizzato nella figura seguente.

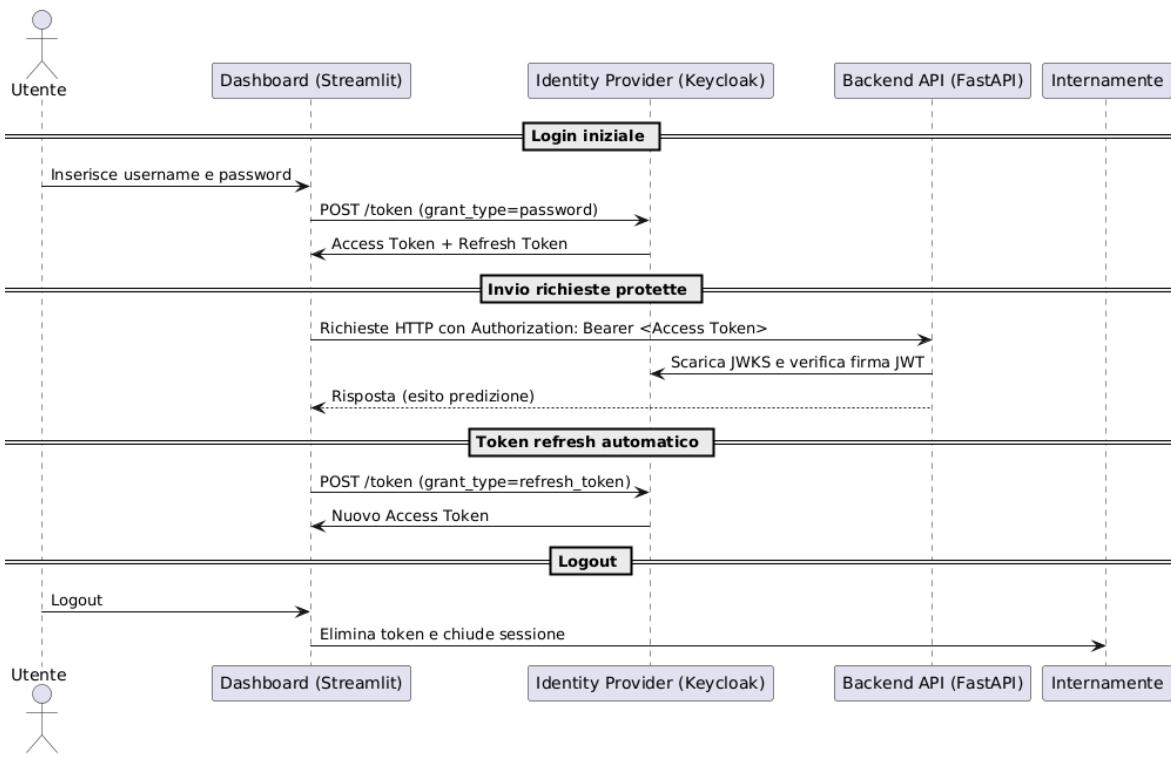


Figura A.6.1 — Sequence Diagram del flusso di autenticazione e autorizzazione OAuth2 tra Streamlit, Keycloak e FastAPI

A.7 Esempio di messaggio JSON inviato su Kafka

Ogni predizione effettuata viene serializzata come evento JSON e pubblicata su un topic Kafka dedicato (prezzi-predetti).

Un esempio di messaggio:

```

1  {
2      "timestamp": "2025-04-28T10:32:14Z",
3      "username": "utente123",
4      "input_parameters": {
5          "GrLivArea": 1800,
6          "OverallQual": 7,
7          "GarageArea": 400,
8          "TotalBathrooms": 2.5,
9          "HasFireplace": true
10     },
11     "predicted_price": 245000
12 }

```

Figura A.7.1 — Esempio di struttura del messaggio JSON inviato su Kafka.

Campi principali:

- timestamp: momento della predizione;
- username: identificativo utente (estratto dal token JWT);
- input_parameters: dati input inviati;
- predicted_price: prezzo stimato.

A.8 File docker-compose.yml

In questa sezione si descrive l'infrastruttura applicativa orchestrata tramite Docker Compose, che integra la dashboard, il backend predittivo, il sistema di autenticazione e il sistema di messaggistica. Il file docker-compose.yml è stato utilizzato per orchestrare i diversi servizi che compongono l'infrastruttura applicativa. Tutti i container sono collegati a una rete virtuale dedicata denominata immobiliare_net, al fine di isolare e facilitare la comunicazione interna tra i servizi senza esporli pubblicamente. Di seguito si descrivono i principali servizi definiti nel file:

Rete di comunicazione

```
networks:  
  immobiliare_net:  
    driver: bridge
```

Figura A.8.1 — Definizione della rete Docker "immobiliare_net" con driver bridge.

Viene creata una rete bridge personalizzata (immobiliare_net) per consentire la comunicazione privata tra i container.

Servizi applicativi

Dashboard

```
dashboard:
  build: ./dashboard
  ports:
    - "127.0.0.1:8501:8501"
  depends_on:
    - model-api
  networks:
    - immobiliare_net
```

Figura A.8.2 — Definizione del servizio Docker "dashboard" con binding locale sulla porta 8501.

Il servizio dashboard ospita l'applicazione Streamlit che costituisce il frontend interattivo della piattaforma. È esposto sulla porta 8501 e configurato per dipendere dal servizio model-api, garantendo che il backend FastAPI sia avviato correttamente prima di permettere operazioni utente. A differenza della configurazione di default di Streamlit — che esporrebbe l'applicazione all'intera rete locale (e potenzialmente anche verso l'esterno) —, in questo caso la porta è stata vincolata specificamente all'interfaccia 127.0.0.1. Questa scelta incrementa la sicurezza del sistema, limitando l'accessibilità della dashboard al solo host locale e prevenendo accessi non autorizzati dall'esterno.

Model-api

```
model-api:
  build: ./model-api
  ports:
    - "8000:8000"
  networks:
    - immobiliare_net
  environment:
    - KEYCLOAK_SERVER_URL=http://keycloak:8080/realmms/immobiliare
```

Figura A.8.3 — Definizione del servizio Docker "model-api" per il backend FastAPI.

Contiene il backend FastAPI che gestisce le predizioni inviate dalla dashboard. Espone la porta 8000 per ricevere le richieste REST e utilizza una variabile d'ambiente per configurare dinamicamente l'URL del server Keycloak.

Keycloak

```
keycloak:
  image: quay.io/keycloak/keycloak:22.0
  command: start-dev
  environment:
    - KEYCLOAK_ADMIN=admin
    - KEYCLOAK_ADMIN_PASSWORD=admin
  ports:
    - "8080:8080"
```

Figura A.8.4 — Definizione del servizio Docker "keycloak" per la gestione dell'autenticazione OAuth2.

Servizio di Identity Provider basato su Keycloak, utilizzato per l'autenticazione OAuth2 degli utenti. Il container avvia Keycloak in modalità sviluppo (start-dev) ed espone l'interfaccia di amministrazione sulla porta 8080.

Servizi di messaggistica asincrona

Zookeeper

```
zookeeper:
  image: confluentinc/cp-zookeeper:7.2.1
  environment:
    ZOOKEEPER_CLIENT_PORT: 2181
    ZOOKEEPER_TICK_TIME: 2000
  ports:
    - "2181:2181"
  networks:
    - immobiliare_net
```

Figura A.8.5 — Definizione del servizio Docker "zookeeper" per la gestione della coordinazione di Kafka.

Zookeeper è il coordinatore necessario per la gestione dei nodi Kafka. Viene configurato con la porta standard 2181 e un tempo di tick aumentato per migliorare la stabilità del cluster.

Kafka

```
kafka:  
  image: confluentinc/cp-kafka:7.2.1  
  depends_on:  
    - zookeeper  
  environment:  
    KAFKA_BROKER_ID: 1  
    KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181  
    KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka:9092,PLAINTEXT_HOST://localhost:29092  
    KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT  
    KAFKA_INTER_BROKER_LISTENER_NAME: PLAINTEXT  
    KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1  
  ports:  
    - "29092:29092"  
  networks:  
    - immobiliare_net
```

Figura A.8.6 — Definizione del servizio Docker "kafka" per la gestione della messaggistica asincrona.

Servizio Apache Kafka, responsabile della gestione degli eventi asincroni prodotti dal backend. Espone la porta 29092 per la comunicazione esterna e dipende dal corretto avvio di Zookeeper.

Kafdrop

```
kafdrop:  
  image: obsidiandynamics/kafdrop  
  ports:  
    - "9000:9000"  
  environment:  
    KAFKA_BROKER_CONNECT: kafka:9092  
    JVM_OPTS: "-Xms32M -Xmx64M"  
  depends_on:  
    - kafka  
  networks:  
    - immobiliare_net
```

Figura A.8.7 — Definizione del servizio Docker "kafdrop" per il monitoraggio dei messaggi Kafka.

Kafdrop è una web UI di monitoring per Kafka, che consente di visualizzare i topic e i messaggi in tempo reale. Viene esposto sulla porta 9000, consentendo una semplice ispezione dei flussi di messaggi generati dal sistema.

Conclusione

Tutti i servizi sono orchestrati attraverso Docker Compose, condividendo una rete privata (immobiliare_net) per la sicurezza e l'efficienza della comunicazione interna. La sequenza di dipendenze (depends_on) garantisce il corretto ordine di avvio dei container, minimizzando il rischio di errori di connessione tra i componenti.

A.9 Specifica OpenAPI per la API di predizione

Per documentare in modo chiaro e accessibile l'interfaccia REST esposta dal backend FastAPI, è stata realizzata una specifica OpenAPI 3.0.

La documentazione OpenAPI realizzata consente di descrivere in modo formale e standardizzato il comportamento dell'endpoint /predict, dedicato alla predizione del prezzo di un immobile residenziale sulla base delle caratteristiche fornite dall'utente. La specifica è strutturata secondo lo standard OpenAPI 3.0 e include le principali informazioni di contesto: il titolo dell'API è "Predict House Price API", con una versione attuale identificata come 1.0.0. Il server di riferimento è l'ambiente di sviluppo locale, accessibile all'indirizzo <http://localhost:8000>. L'endpoint principale è rappresentato da una richiesta HTTP di tipo POST al percorso /predict. Questa operazione richiede l'invio di un corpo (body) in formato JSON, contenente le caratteristiche principali dell'immobile da valutare. I parametri richiesti includono:

- GrLivArea, corrispondente alla superficie abitabile dell'immobile, espresso come valore intero;
- OverallQual, che rappresenta la qualità complessiva dell'abitazione, anch'esso espresso come intero;
- GarageArea, relativo alla superficie del garage, in formato intero;

- TotalBathrooms, indicante il numero totale di bagni presenti nell'immobile, espresso come valore numerico reale;
- HasFireplace, un valore booleano che specifica la presenza o meno di un caminetto.

In risposta alla richiesta, il sistema restituisce un oggetto JSON contenente il campo predicted_price, che rappresenta il prezzo stimato dell'immobile in dollari. Le immagini seguenti illustrano la rappresentazione grafica della documentazione API tramite Swagger UI:

Predict House Price API 1.0.0 OAS 3.0

API REST per la predizione del prezzo di un immobile residenziale.

Servers
 ▾

Prediction

POST /predict Esegue la predizione del prezzo sulla base dei parametri forniti.

Parameters Try it out

No parameters

Request body required application/json

Example Value | Schema

```
{
  "GrLivArea": 0,
  "OverallQual": 0,
  "GarageArea": 0,
  "TotalBathrooms": 0,
  "HasFireplace": true
}
```

Figura A.9.1 — Visualizzazione dei parametri richiesti per la predizione

Responses		
Code	Description	Links
200	Prezzo stimato restituito con successo. <small>Media type</small> <div style="border: 1px solid #ccc; padding: 2px; display: inline-block;">application/json</div> <small>▼</small> <small>Controls Accept header.</small> Example Value Schema <pre>{ "predicted_price": 0 }</pre>	No links
400	Errore di validazione dei dati.	No links
401	Accesso non autorizzato (token mancante o invalido).	No links
500	Errore interno del server.	No links

Figura A.9.2 — Visualizzazione dei codici di risposta e del formato di output

A.10 Configurazione di Keycloak nel sistema di autenticazione

Durante la fase di implementazione della sicurezza applicativa, è stato configurato un server Keycloak con l'obiettivo di gestire il flusso di autenticazione basato su protocollo OAuth2 tra la dashboard Streamlit e il backend FastAPI.

All'interno di Keycloak è stato creato un realm denominato immobiliare, dedicato esclusivamente a questo progetto. La creazione di un realm specifico ha consentito di isolare la gestione dell'autenticazione da eventuali altri domini o contesti applicativi, assicurando una configurazione più ordinata e sicura.

È stato inoltre definito un client denominato dashboard-client, configurato come Confidential Client. Questa tipologia di client richiede la validazione delle credenziali client nel flusso di autenticazione ed è abilitata all'uso del grant type Resource Owner Password Credentials, che permette alla dashboard di ottenere direttamente access token e refresh token sulla base delle credenziali utente fornite.

Tale configurazione garantisce che ogni interazione con il backend avvenga solo previa autenticazione verificata, integrando il controllo degli accessi all'interno dell'architettura applicativa in modo sicuro e scalabile.

Grant Types abilitati:

- Password Credentials (per login diretto username/password);
- Refresh Token (per rinnovo automatico dei token);
- Roles:
 - user: ruolo assegnato agli utenti finali abilitati all'uso della dashboard.

Scope dei Token JWT:

I token emessi contengono i seguenti claims principali:

- preferred_username;
- email;
- realm_access.roles.

Questa configurazione permette a Streamlit di autenticare l'utente, ricevere un access token, e validare ogni richiesta verso il backend attraverso FastAPI, mantenendo un controllo centralizzato degli accessi.

A.11 Considerazioni sull'uso di Docker Compose e limiti in ambiente di produzione

L'infrastruttura del progetto è stata orchestrata mediante Docker Compose, strumento ideale per ambienti di sviluppo, test e prototipazione rapida. Tuttavia, in un contesto di produzione scalabile e ad alta disponibilità, l'uso esclusivo di Docker Compose si sia rivelato uno strumento estremamente utile per la gestione dei container in fase di sviluppo e testing, presenta alcune limitazioni che ne riducono l'efficacia in contesti di produzione complessi.

In primo luogo, Docker Compose offre un supporto limitato all'orchestrazione avanzata: non gestisce automaticamente il bilanciamento del carico tra istanze, il failover dei servizi in caso di malfunzionamento, né consente una scalabilità orizzontale dinamica dei container.

Inoltre, la gestione dei segreti (come password, chiavi e certificati) risulta basilare e meno sicura rispetto a soluzioni più strutturate come Kubernetes Secrets o strumenti esterni come HashiCorp Vault.

Un'altra limitazione significativa riguarda la gestione degli aggiornamenti applicativi: Compose non supporta nativamente meccanismi di rolling update o di

rollback sicuro, rendendo più rischiose le operazioni di aggiornamento in ambienti di produzione.

Infine, manca un'integrazione immediata con sistemi di monitoraggio e logging centralizzato, come Prometheus, Grafana o la suite ELK (Elasticsearch, Logstash, Kibana), strumenti essenziali per garantire l'osservabilità dei microservizi distribuiti. Alla luce di queste considerazioni, per garantire robustezza, disponibilità e scalabilità a livello enterprise, sarebbe opportuno prevedere in una fase successiva la migrazione dell'infrastruttura verso un ambiente orchestrato tramite Kubernetes. Tale passaggio consentirebbe di sfruttare strumenti avanzati come Helm Charts per il deploy automatizzato, Ingress Controllers per la gestione del traffico e sistemi di autoscaling per adattare dinamicamente le risorse all'effettivo carico di lavoro.

Nonostante queste limitazioni, l'adozione di Docker Compose per il progetto ha garantito portabilità, riproducibilità, semplicità nella gestione di un ambiente multi-container e coerenza tra ambienti di sviluppo e test. Queste caratteristiche rendono la soluzione attuale una base solida, pronta per future estensioni ed evoluzioni verso contesti operativi più complessi e distribuiti.

A.12 Tecniche di validazione incrociata (Cross Validation)

La validazione incrociata rappresenta una tecnica statistica fondamentale per la valutazione della capacità di generalizzazione di un modello predittivo su dati non visti. All'interno del progetto è stata adottata, in particolare, la tecnica della k-Fold Cross Validation con $k=5$, seguendo un processo strutturato in più fasi. Il dataset complessivo è stato suddiviso casualmente in cinque sottoinsiemi di pari dimensioni (fold). In ogni iterazione, quattro di questi fold sono stati utilizzati per l'addestramento del modello, mentre il fold rimanente è stato impiegato come set di validazione. Questo procedimento è stato ripetuto per cinque cicli, facendo ruotare sistematicamente il fold di validazione a ogni iterazione. Al termine del processo, l'errore complessivo del modello è stato calcolato come media degli errori ottenuti nelle diverse validazioni, fornendo così una misura più stabile e rappresentativa delle reali capacità predittive del sistema.

L'adozione della validazione incrociata presenta diversi vantaggi concreti:

- riduce la varianza nelle stime di performance rispetto a una semplice divisione train/test;
- migliora la robustezza del modello prevenendo fenomeni di overfitting su uno specifico split;

- consente una stima più affidabile della capacità di generalizzazione sui dati futuri.

Nel progetto, la validazione incrociata è stata applicata in modo specifico alla fase di ottimizzazione degli iperparametri del modello Random Forest, utilizzando la classe CrossValidator disponibile in PySpark. L'importanza della validazione incrociata come strumento robusto per la valutazione dei modelli è ampiamente riconosciuta nella letteratura di riferimento nel campo del machine learning, come evidenziato da Hastie, Tibshirani e Friedman (2009).

A.13 Metriche di valutazione dei modelli predittivi

Per valutare la qualità dei modelli predittivi sviluppati, sono state utilizzate due metriche standard nel contesto della regressione.

La qualità dei modelli di regressione sviluppati è stata valutata attraverso due metriche fondamentali: il Root Mean Squared Error (RMSE) e il coefficiente di determinazione (R^2). Il Root Mean Squared Error (RMSE) misura l'errore medio quadratico tra i valori reali osservati e quelli predetti dal modello. In sostanza, più il valore della RMSE è basso, più le predizioni risultano vicine ai dati reali. Questa metrica penalizza in misura maggiore gli errori di grande entità, risultando particolarmente utile in ambiti, come quello immobiliare, in cui la presenza di outlier può influenzare significativamente l'accuratezza della stima. Il coefficiente di determinazione (R^2) rappresenta invece la proporzione di varianza della variabile target che viene spiegata dal modello rispetto alla varianza totale osservata. Un valore elevato di R^2 , prossimo a 1, indica che il modello riesce a spiegare in modo accurato la variabilità dei dati, mentre un valore vicino a 0 suggerisce una scarsa capacità predittiva. Nel progetto, il modello Random Forest ha raggiunto un valore di R^2 pari a circa 0,87, accompagnato da una significativa riduzione della RMSE rispetto al modello lineare di base, confermando la bontà della scelta effettuata. L'uso combinato di RMSE e R^2 come metriche standard nella valutazione dei modelli di regressione è discusso estesamente da Géron (2019), mentre l'efficacia degli algoritmi ensemble, come la Random Forest, nel migliorare la robustezza e la stabilità delle predizioni è stata evidenziata da Breiman (2001).

Questi concetti, relativi alla misura dell'errore e all'ottimizzazione della predizione, erano già stati esplorati in forma introduttiva nell'ambito della precedente tesi di

laurea triennale (Di Grazia, 2022), focalizzata sull'analisi delle serie temporali finanziarie.

Il presente lavoro si inserisce quindi in un percorso di continuità accademica, estendendo e specializzando tali conoscenze nel dominio della previsione dei prezzi immobiliari mediante tecniche avanzate di machine learning.

A.14 Introduzione alla Explainable AI (XAI) e valori SHAP

La crescente adozione di sistemi predittivi in ambiti critici rende necessaria la presenza di strumenti di interpretabilità (Explainable AI - XAI) a supporto della fiducia degli utenti e della correttezza delle decisioni.

Explainable AI (XAI) comprende tecniche progettate per:

- Spiegare il comportamento di modelli complessi (es. Random Forest, XGBoost).
- Identificare quali variabili influenzano maggiormente una predizione.
- Supportare trasparenza, auditabilità e compliance normativa (es. EU AI Act).

Le considerazioni sull'auditabilità e la trasparenza dei sistemi predittivi sono coerenti con le direttive proposte nella regolamentazione europea sull'intelligenza artificiale (European Commission, 2021).

Valori SHAP

I valori SHAP (SHapley Additive exPlanations) rappresentano una delle tecniche più avanzate e rigorose per l'interpretazione dei modelli predittivi complessi. Questa metodologia si basa sulla teoria dei giochi cooperativi di Shapley, che fornisce un criterio equo per attribuire a ciascuna feature il proprio contributo al risultato finale della predizione. L'approccio SHAP si distingue per alcune caratteristiche fondamentali:

- ogni feature riceve un valore numerico che quantifica il suo contributo, positivo o negativo, alla predizione;
- è possibile interpretare sia l'importanza globale delle feature sull'intero dataset, sia il contributo locale di ciascuna feature su una singola osservazione.

Ad esempio, se per un dato immobile la feature GrLivArea (superficie abitabile) presenta un valore SHAP positivo elevato, questo indica che quella variabile ha contribuito ad aumentare il prezzo stimato rispetto alla media. I valori SHAP offrono quindi un approccio formalmente fondato all'interpretabilità locale dei modelli, come proposto da Lundberg e Lee (2017). Nel progetto sviluppato, pur non essendo stato ancora integrato un modulo completo di Explainable AI (XAI), si prevede come sviluppo futuro l'adozione dei valori SHAP, al fine di rafforzare la trasparenza del sistema predittivo e migliorare l'esperienza complessiva dell'utente finale.

A.14 Principi di Ergonomia Cognitiva nella progettazione della dashboard

A.14.1 Ergonomia cognitiva: definizione e rilevanza

L'ergonomia cognitiva rappresenta una branca specifica dell'ergonomia che si occupa dello studio dei processi mentali implicati nell'interazione tra gli esseri umani e i sistemi tecnologici. Tali processi comprendono attività cognitive fondamentali come la percezione, l'attenzione, la memoria, il ragionamento e la presa di decisione. L'obiettivo principale dell'ergonomia cognitiva è quello di progettare sistemi che ottimizzino l'efficienza, l'efficacia e il comfort dell'utente, riducendo al minimo il carico cognitivo necessario per portare a termine un compito. Un sistema ergonomicamente corretto non si limita quindi a essere funzionale, ma supporta attivamente le capacità cognitive dell'utente, facilitandone l'interazione e minimizzando errori e frustrazione. Nel contesto specifico della dashboard Streamlit sviluppata per la predizione dei prezzi immobiliari, i principi dell'ergonomia cognitiva hanno guidato la definizione della struttura dell'interfaccia, la modalità di inserimento dei dati e la gestione dell'interazione utente-sistema. Attraverso un'attenta applicazione di questi principi, è stato possibile costruire un'interfaccia che assiste l'utente nelle fasi di input, riduce la complessità percepita e migliora complessivamente l'esperienza d'uso.

A.14.2 Applicazione dei principi ergonomici nella dashboard

La progettazione della dashboard ha seguito diversi principi ergonomici fondamentali, con l'obiettivo di migliorare la qualità dell'interazione, ridurre il carico cognitivo e favorire un'esperienza utente più fluida e naturale. Un primo accorgimento riguarda la modularizzazione delle informazioni attraverso l'uso di

sezioni collassabili. I dati richiesti all'utente sono stati suddivisi in gruppi tematici coerenti (ad esempio: dimensioni, garage, piscina, caminetto), espandibili su richiesta. Questa scelta progettuale risponde al principio della riduzione della complessità percepita, consentendo all'utente di concentrarsi su un solo gruppo di informazioni alla volta, migliorando così la focalizzazione e riducendo la sensazione di sovraccarico cognitivo. Un secondo principio applicato è quello della visibilità dello stato del sistema, corrispondente alla prima delle dieci euristiche di Nielsen. Dopo l'inserimento dei dati e il clic sul pulsante di calcolo del prezzo, il risultato viene mostrato immediatamente. Questo feedback rapido fornisce all'utente conferme chiare delle proprie azioni, aumentando la trasparenza del sistema e rafforzando la fiducia nell'interazione. Un altro elemento rilevante riguarda l'implementazione di controlli dinamici e adattivi, finalizzati a ridurre la memoria di lavoro richiesta durante l'inserimento dei dati. Alcuni campi della dashboard si abilitano o si adattano automaticamente in base ai valori già forniti: ad esempio, la superficie del garage si aggiorna dinamicamente in funzione del numero di posti auto selezionato. In questo modo, l'utente non deve ricordare manualmente informazioni già inserite, ma riceve supporto attivo dal sistema.

Infine, particolare attenzione è stata dedicata alla coerenza e standardizzazione delle etichette, in linea con il principio di coerenza e riconoscibilità (Heuristic #4 di Nielsen). Terminologie e codifiche sono mantenute consistenti tra il frontend Streamlit, il backend FastAPI e il dataset sottostante. Questo approccio favorisce il riconoscimento rapido delle informazioni, riduce il rischio di errori e facilita la navigazione intuitiva all'interno dell'interfaccia. Nel loro insieme, questi accorgimenti ergonomici concorrono a costruire un'esperienza utente fluida, assistita e cognitivamente sostenibile, elevando l'efficacia complessiva della dashboard predittiva.

A.14.3 Fondamenti di psicologia cognitiva utilizzati

La progettazione della dashboard ha tenuto conto di alcuni principi chiave della psicologia cognitiva, al fine di ottimizzare l'interazione utente-sistema e ridurre il carico cognitivo associato alle operazioni di inserimento dei dati. Un primo riferimento importante è costituito dalla Legge di Miller ("7±2"), secondo la quale il numero di elementi che un individuo può mantenere contemporaneamente nella memoria di lavoro è limitato a circa sette unità, con una variabilità di due elementi in più o in meno. In coerenza con questo principio, la dashboard è stata strutturata in modo da presentare pochi campi alla volta, suddividendoli in sezioni tematiche collassabili. Questo approccio riduce il carico mentale richiesto all'utente, favorendo

una compilazione più fluida e meno soggetta a errori. La progettazione si è ispirata anche alla Teoria del carico cognitivo, che sottolinea come la capacità cognitiva umana sia limitata e facilmente sovraccaricabile in presenza di compiti complessi o interfacce disorganizzate. Per prevenire sovraccarichi cognitivi, i dati da inserire sono stati presentati in modo segmentato e logico, con il supporto dinamico di automatismi che guidano l'utente nella compilazione, riducendo la necessità di mantenere simultaneamente molte informazioni in memoria. Infine, è stato considerato il principio della compatibilità Stimolo-Risposta, secondo cui l'interfaccia dovrebbe adattarsi alle azioni e alle aspettative dell'utente, facilitando scelte coerenti senza costringerlo a processi mentali complessi o controiduitivi. In questo senso, la dashboard adegua dinamicamente i campi di input in base ai dati già forniti, suggerendo opzioni compatibili e minimizzando la necessità di correzioni o rielaborazioni.

Attraverso l'applicazione di questi principi, la dashboard si configura come uno strumento non solo tecnologicamente avanzato, ma anche ergonomicamente ottimizzato per supportare efficacemente l'utente nei compiti di stima predittiva.

Riferimenti – Appendice A

- Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow*. O'Reilly Media.
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning*. Springer.
- Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), 5–32.
- Lundberg, S. M., & Lee, S. I. (2017). A Unified Approach to Interpreting Model Predictions. *Advances in Neural Information Processing Systems (NeurIPS)*.
- European Commission. (2021). *Proposal for a Regulation on Artificial Intelligence (EU AI Act)*.

- Wickens, C. D. (2008). *Engineering Psychology and Human Performance*. Pearson.
- Norman, D. A. (1988). *The Design of Everyday Things*. Basic Books.
- Nielsen, J. (1995). *10 Usability Heuristics for User Interface Design*.