UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO

CORSO DI LAUREA IN INFORMATICA

HOUSE PREDICTION AND CLASSIFICATION

DOCUMENTAZIONE

Caso di studio di

Ingegneria della Conoscenza a.a. 2019-2020

Il progetto è stato realizzato da:

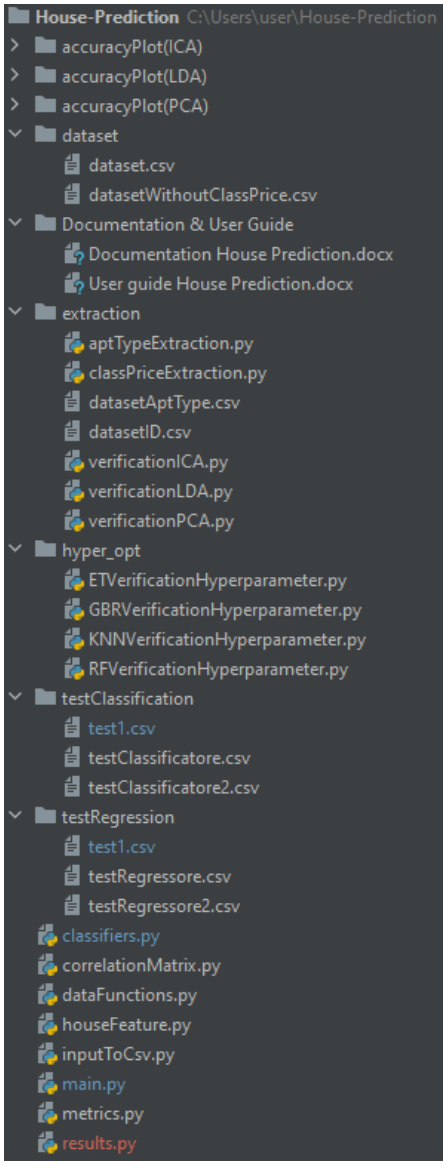
- Leonardo Bellizzi (684712)
  e-mail: l.bellizzi@studenti.uniba.it
- Giuseppe Conticchio (675622)
  e-mail: g.conticchio2@studenti.uniba.it
- Michelangelo Lopes (639742)
  e-mail: m.lopes@studenti.uniba.it

1. INTRODUZIONE E ORGANIZZAZIONE DEL PROGETTO:

Oggetto d'analisi del progetto è stato un dataset comprendente le informazioni relative a delle abitazioni della città di Taegù (Corea del Sud). Su questo dataset sono state svolte diverse operazioni tra cui:

1) Pre-processing dei dati contenuti all'interno del dataset;
2) Estrazione di features mediante CSP (ragionamento con vincoli);
3) Inserimento di algoritmi atti alla classificazione dei dati;
4) Inserimento di algoritmi atti alla regressione dei dati;
5) Primo test di classificatori e regressori.
6) Metriche di valutazione per l'accuratezza dei valori ottenuti dagli algoritmi di classificazione e regressione;
7) Ottimizzazione degli iperparametri per far sì che i classificatori ed i regressori possano dare il miglior risultato possibile;
8) Secondo test di classificatori e regressori.

Il progetto è suddiviso così come segue:



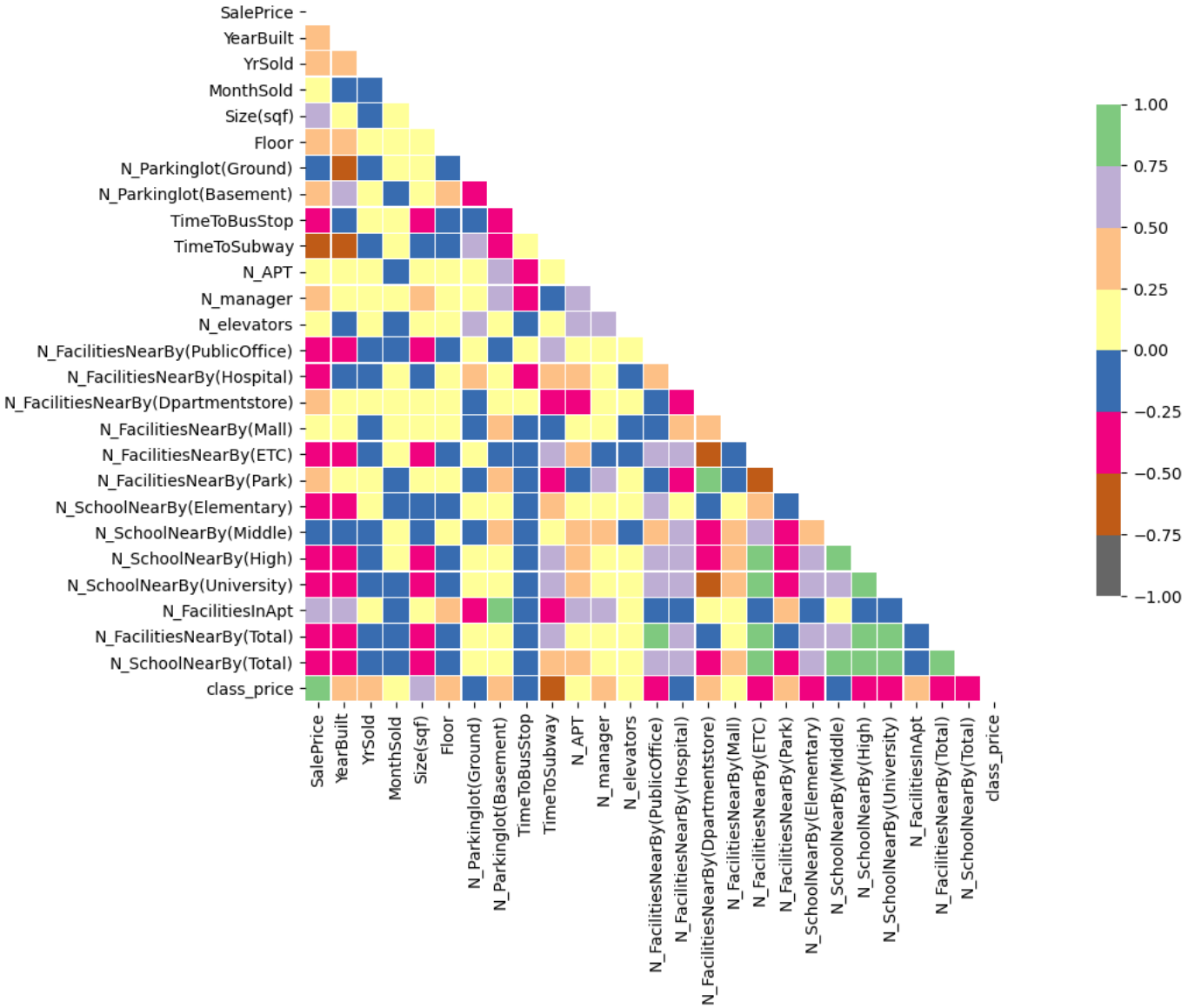
- Le **prima** e la **terza cartella** contengono gli screenshot degli esperimenti effettuati tramite le tecniche di feature extraction (PDA, ICA), mentre la **seconda** cartella contiene lo screenshot dei diversi tipi di solver utilizzati dalla tecnica LDA;
- Nella cartella **dataset** sono presenti il dataset "originale" sul quale si sono effettuate le estrazioni delle features ed il dataset senza la feature estratta;
- Nella cartella **Documentation & User Guide** è presente la documentazione del progetto all'interno della quale sono esplicati gli argomenti trattati e la Guida Utente che spiega come avviene l'esecuzione del programma da parte del sistema;
- Nella cartella **extraction** sono presenti i moduli riguardanti l'estrazione delle features *class_price* e *AptType* assieme ai tre moduli riguardanti le tre tecniche di features extraction utilizzate ed il dataset contenente la feature estratta *AptType* (non utilizzata ai fini della classificazione);
- Nella cartella **hyper_opt** sono presenti quattro moduli riguardanti l'utilizzo ed il settaggio degli iperparametri applicato per l'ExtraTree Classifier, il GradientBoosting Regressor, il K-NN Classifier ed il Random Forest;
- Nella cartella **testClassification** sono presenti i csv utilizzabili per testare i classificatori;
- Nella cartella **testRegression** sono presenti i csv utilizzabili per testare i regressori;
- Il modulo **classifiers.py** ha il compito di addestrare i classificatori sui dati di training;
- Il modulo **correlationMatrix.py** ha la funzione di plottare la matrice di correlazione delle features del dataset;
- Nel modulo **houseFeature.py** è sono presenti le funzioni che hanno il compito di caricare il csv e prendere le relative features;
- Il modulo **inputToCsv.py** ha il compito di scrivere su un csv i valori delle features che vengono inserite da tastiera dall'utente per classificare un immobile e predire il prezzo dello stesso (Opzione 3 e 4 del menù).
- Nel modulo **main.py** è presente il main del programma il quale dovrà essere eseguito dall'utente;
- Il modulo **metrics.py** ha il compito di restituire i valori relativi alle metriche dei classificatori e dei regressori e plottare le matrici di confusione dei classificatori e le rette di regressione dei regressori;
- Il modulo **results.py** si occupa di restituire i risultati ottenuti dalla fase di test.

2. DATASET E FEATURES:

I dati che sono stati presi in esame, per questo progetto, sono stati ottenuti da un dataset presente su Kaggle. Il dataset in oggetto contiene 5892 abitazioni (suddivise in 3948 per il training e 1944 per il test) le quali possiedono, sia delle caratteristiche relative all'immobile stesso e sia delle caratteristiche relative alle strutture presenti nelle vicinanze. Andando nel dettaglio le features sono:

- **SalePrice**: prezzo di vendita dell'immobile;
- **YearBuilt**: anno di costruzione;
- **YrSold**: anno di vendita;
- **MonthSold**: mese di vendita;
- **Size (sqf)**: dimensione dell'appartamento in piedi quadri;
- **Floor**: numero di piani;
- **N_Parkinglot(Ground)**: numero di stalli adibiti al parcheggio auto/moto a livello della sede stradale;
- **N_Parkinglot(Basement)**: numero di stalli adibiti al parcheggio auto/moto posti al di sotto della sede stradale (sotterranei);
- **TimeToBusStop**: tempo impiegato per andare dallo stabile alla più vicina fermata del Bus a piedi;
- **TimeToSubway**: tempo impiegato per andare dallo stabile alla più vicina fermata della Metropolitana a piedi;
- **N_APT**: numero dell'appartamento presente all'interno dello stabile;
- **N_Manager**: numero dei responsabili di vendita affiliati all'appartamento;
- **N_elevators**: numero degli ascensori;
- **N_FacilitiesNearBy(PublicOffice)**: numero di uffici pubblici presenti nelle vicinanze;
- **N_FacilitiesNearBy(Hospital)**: numero di ospedali presenti nelle vicinanze;
- **N_FacilitiesNearBy(Dpartmentstore)**: numero di grandi magazzini presenti nelle vicinanze;
- **N_FacilitiesNearBy(Mall)**: numero di centri commerciali presenti nelle vicinanze;
- **N_FacilitiesNearBy(ETC)**: numero di altri servizi presenti nelle vicinanze;
- **N_FacilitiesNearBy(Park)**: numero di parchi presenti nelle vicinanze;
- **N_SchoolNearBy(Elementary)**: numero di scuole elementari presenti nelle vicinanze;
- **N_SchoolNearBy(Middle)**: numero di scuole medie presenti nelle vicinanze;
- **N_SchoolNearBy(High)**: numero di scuole superiori presenti nelle vicinanze;
- **N_SchoolNearBy(University)**: numero di università presenti nelle vicinanze;
- **N_FacilitiesInApt**: numero di servizi presenti all'interno dell'appartamento
- **N_FacilitiesNearBy(Total)**: numero totale di servizi presenti nelle vicinanze;
- **N_SchoolNearBy(Total)**: numero totale di scuole presenti nelle vicinanze.

Inoltre per osservare il grado di correlazione presente tra le features si è utilizzata la **matrice di correlazione** la quale è così composta:



3. IN COSA CONSISTE…

Come detto nell'introduzione, sono state svolte diverse operazioni per arrivare al risultato finale.

La **prima operazione** ha riguardato la fase **di pre-processing** del dataset preso in esame, poiché esso è stato modificato per arrivare al dataset finale utilizzato dal sistema; questo perché nell'originale erano presenti dei dati in formato stringa, non utili per lo scopo del progetto e non interpretabili, ed in più erano presenti colonne con valori espressi in minuti (TimeToBusStop e TimeToSubway) le quali sono state convertite in numeri interi.

La **seconda operazione** si basa sul **ragionamento con vincoli (CSP)**, fondamentale per fare in modo che al dataset vengano aggiunte delle features mediante un ragionamento che coinvolga altre caratteristiche delle abitazioni. Sono state estratte la feature **class_price** e la feature **AptType**. **class_price** è la fascia di prezzo all'interno della quale si trova l'appartamento in oggetto. Le fasce di prezzo individuate sono 10 (da fascia A fino a fascia L, rispettivamente dalla fascia con appartamenti meno costosi fino alla fascia con appartamenti più costosi). **AptType** si riferisce alla tipologia di individuo che potrebbe vivere comodamente all'interno di uno degli appartamenti in base a delle caratteristiche dell'immobile e del suo vicinato. Sono stati considerati il numero di servizi presenti nei pressi dell'immobile come i parcheggi, i parchi, i centri commerciali e le scuole. In base a questi criteri l'appartamento può essere adatto ad una persona *Single*, ad una *Famiglia*, ad un *Anziano*, ad uno *Studente Universitario* ed infine ad una *Coppia sposata senza figli*.

La **terza operazione** è relativa alla **classificazione**, per la quale sono stati svolti numerosi studi nell'ambito del Machine Learning. Lo scopo principale è quello di classificare gli appartamenti, nella corretta fascia di prezzo ('Class Price'), in base alla miscellanea di caratteristiche degli appartamenti stessi e confrontare alcuni dei modelli di classificazione basati su apprendimento supervisionato.

La **quarta operazione** è quella relativa alla **predizione**, il cui scopo è quello di predire il prezzo vero e proprio di una casa passata in input tenendo conto di tutte le caratteristiche escluse 'SalePrice' e 'ClassPrice'. L'effettiva correttezza delle previsioni effettuate è verificata dai cosiddetti regressori i quali assegnano uno score (da 0 a 1) al prezzo ottenuto come output.

Per la **classificazione** sono stati utilizzati i seguenti modelli di classificazione:

- Case Based Reasoning: **K-Nearest Neighbors Classifier**
- Probabilistic Classifiers: **Naive Bayes Classifier**
- Ensamble Learning Models: **Random Forest Classifier & Extra Classifier**

Per la **predizione** sono stati utilizzati i seguenti modelli di predizione:

- **Simple Linear Regression**
- **Gradient Boosting Regression**
- **XGBoost Regression**

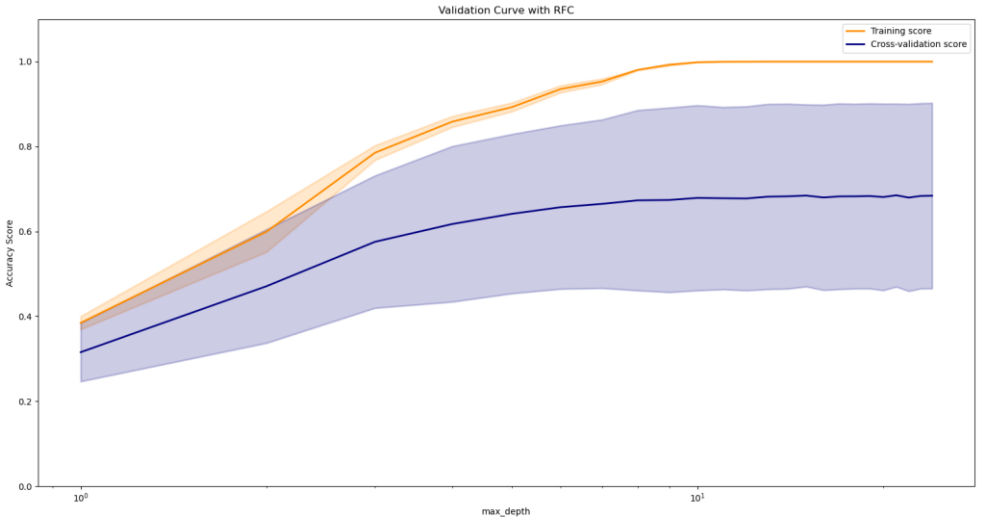
La **quinta operazione** si riferisce al test di classificatori e regressori appena costruiti per osservare gli score dei modelli utilizzati.

La **sesta operazione** è quella relativa alla **valutazione attraverso** determinate **metriche**, dei valori ottenuti dai classificatori e dai regressori, difatti, confrontando i diversi modelli utilizzati per la classificazione risulta che il **Random Forest Classifier** è quello **più preciso** poiché possiede i più alti valori di **Accuracy, Precision, Recall, F1-Measure;** invece confrontando i diversi modelli utilizzati per la regressione risulta che l'**XGBoost Regressor** è quello migliore poiché ha il punteggio, relativo all'$\mathbf{R^2}$, più elevato il che vuol dire che è il regressore che predice in maniera migliore il prezzo di una abitazione.

La **settima operazione** è relativa all'**utilizzo degli Iperparametri**. Al fine di rendere notevolmente alta l'accuratezza di ciascun classificatore e regressore utilizzato è stato seguito un procedimento di ottimizzazione degli iperparametri. Partiamo dal presupposto che ciascun modello di classificazione e regressione prevede la presenza di parametri opportunamente passati in fase di costruzione di un determinato modello, dunque ciascun valore associato a questi ultimi prende il nome di iperparametro. Se non esplicitati, ai parametri verranno associati valori di default, che molto spesso non permettono al modello di esaltare la sua massima accuratezza. Le tecniche di ottimizzazione utilizzate in questo progetto sono state:

- **Curva di validazione**

Tale metodo è utile al fine di verificare visivamente i valori potenzialmente ottimizzati di ciascun modello. Una curva di validazione può essere tracciata su un grafico, per mostrare come un modello si comporta con diversi valori di un singolo iperparametro.



[Nell'immagine curva di validazione per il Random Forest Classifier]

Attraverso questo grafico, si può notare come avviene tale procedimento e sulla base di quale metrica si stabilisce il giusto settaggio di un iperparametro. Nel grafico sopra riportato è rappresentata la curva di validazione per il parametro *"max_depth"* presente nel Random Forest Classifier;
sull'asse delle ordinate è presente il valore di accuratezza, metrica fondamentale ai fini dell'utilizzo di tale procedura, mentre sull'asse delle ascisse si ha il parametro che intendiamo settare sulla base di diversi valori che può avere.
Infine, le due curve (training score, cross-validation score) rappresentano il vero e proprio concetto principale di tale procedura. In base alle stesse si può controllare quale è il valore, dell'iperparametro, per cui l'accuratezza diventa massima.

- **Exhaustive grid search:**

Tale metodo, fornito da *GridSearchCV,* genera in maniera esaustiva i possibili candidati (iperparametri) attraverso una griglia di valori specificata opportunamente dal parametro *"param_grid",* caratterizzato da un range di valori per ogni singolo parametro specificato dall'utente. In maniera del tutto automatica, vengono valutate tutte le possibili combinazioni di assegnazioni degli iperparametri e viene mantenuta la combinazione migliore. Al termine di tale processo, verranno mostrati quelli che sono gli iperparametri migliori per un determinato modello di classificazione.

Delle tante procedure utili ai fini di tale topic, sono state scelte proprio queste due in quanto la prima si basa su un procedimento del tutto "manuale" e fortemente esplicativo, visto l'utilizzo di un grafico, il secondo invece è del tutto "automatico", tentando ogni combinazione, sulla base dell'accuratezza raggiunta in ogni singolo tentativo.

4. MODELLI DI CLASSIFICAZIONE UTILIZZATI

Al fine di ottenere una predizione sui nuovi esempi, sono stati applicati modelli di classificazione basati su apprendimento supervisionato, derivati dalla libreria sklearn. L'idea di utilizzare più modelli ha avuto lo scopo di valutare l'accuratezza di ogni singolo modello in fase di test.

K-Nearest Neighbors:

è un algoritmo utilizzato nel riconoscimento di pattern per la classificazione di oggetti basandosi sulle caratteristiche dei k oggetti più vicini a quello considerato. Un oggetto è classificato in base alla maggioranza dei voti dei suoi k vicini. k è un intero positivo tipicamente non molto grande. La scelta di k dipende dalle caratteristiche dei dati. Generalmente all'aumentare di k si riduce il rumore che compromette la classificazione. Al fine dell'apprendimento lo spazio multidimensionale viene partizionato in regioni in base alle posizioni e alle caratteristiche degli oggetti di apprendimento, rappresentati come vettori. Un oggetto è assegnato alla classe C se questa è la più frequente fra i k esempi più vicini all'oggetto sotto esame, la vicinanza si misura in base alla distanza fra punti. I vicini sono presi da un insieme di oggetti per cui è nota la classificazione corretta.

GaussianNB:

I classificatori basati sul modello Naïve Bayes, utilizzano il teorema di Bayes:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) P(x_1, \ldots, x_n \mid y)}{P(x_1, \ldots, x_n)}$$

Dove P(y|x1,…,xn) è la probabilità a posteriori, P(y) è la probabilità a priori, P(x1,…,xn|y) è la verosimiglianza e P(x1,…,xn) è la funzione di partizione. Nell' utilizzo del classificatore GaussianNB, si presume che la probabilità delle feature sia gaussiana:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

I parametri σy e μy sono stimati usando la massima probabilità.

Random Forest:

È un modello d'insieme ottenuto dall'aggregazione tramite bagging di alberi di decisione. Esso è un meta-stimatore che si adatta ad una serie di alberi decisionali addestrati su vari sotto-campioni del dataset e utilizza la media di ogni singolo output di ogni albero per migliorare l'accuratezza predittiva e il controllo del sovradattamento. Il Random Forest deve essere dotato di due matrici: una matrice X sparsa che contiene i campioni di addestramento e una matrice Y di dimensioni che contiene i valori target.

Extra Trees:

Tale modello è simile al precedente, la differenza risiede nella scelta degli alberi, la quale avviene in maniera puramente casuale.

5. MODELLI DI PREDIZIONE UTILIZZATI

Linear Regression:

In statistica la **regressione lineare** rappresenta un metodo di stima del valore atteso condizionato da una variabile dipendente o endogena, dati i valori di altre variabili indipendenti o esogene.

Gradient Boosting Regression:

Il Gradient boosting è una tecnica di machine learning di regressione e problemi di Classificazione statistica che producono un modello predittivo nella forma di un insieme di modelli predittivi deboli, tipicamente alberi di decisione. Costruisce un modello in maniera simile ai metodi di boosting, e li generalizza permettendo l'ottimizzazione di una funzione di perdita differenziabile arbitraria.

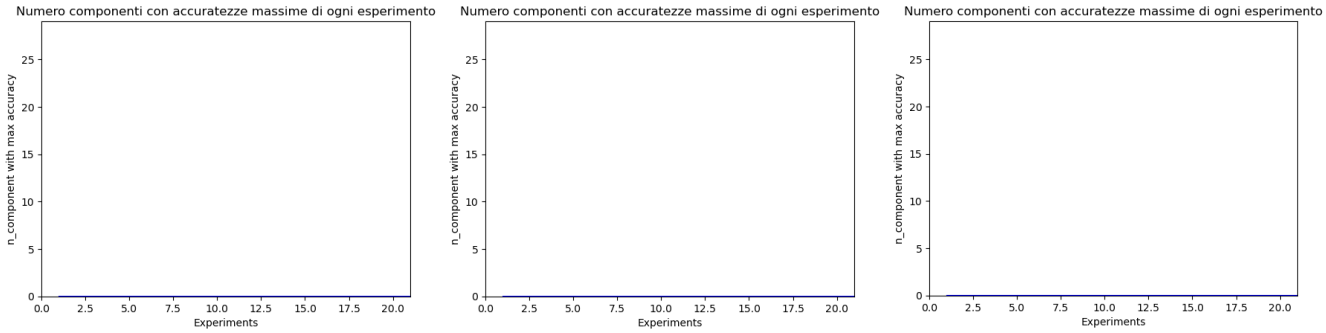
XGBoost Regression:

XGBoost è un'implementazione open source popolare ed efficiente dell'algoritmo degli alberi di gradient boosting. Il gradient boosting è un'algoritmo di apprendimento supervisionato che tenta di prevedere con precisione una variabile di destinazione combinando le stime di un insieme di modelli più semplici e deboli. Quando si utilizza potenziamento del gradiente per la regressione, gli studenti deboli sono alberi di regressione e ogni albero di regressione mappa un punto dati di input a una delle sue foglie che contiene un punteggio continuo. XGBoost riduce al minimo una funzione obiettivo regolarizzata (L1 e L2) che combina una funzione di perdita convessa (in base alla differenza tra gli output previsti e target) e un termine di penalità per la complessità del modello (in altre parole, le funzioni dell'albero di regressione). La preparazione procede in modo iterativo, aggiungendo nuovi alberi che prevedono i valori residuali o gli errori degli alberi antecedenti, che vengono quindi combinati con gli alberi precedenti per la previsione finale. Questa tecnica viene chiamata gradient boosting (potenziamento del gradiente) poiché utilizza un algoritmo di discesa del gradiente per ridurre al minimo le perdite durante l'aggiunta di nuovi modelli.

6. DISCUSSIONE DEI RISULTATI OTTENUTI

6.1.1 SPERIMENTAZIONI FEATURES EXTRACTION

La Feature Extraction mira a ridurre il numero di features in un set di dati creando nuove feature da quelle esistenti (quindi scartando le funzionalità originali). Questo nuovo set ridotto di funzionalità dovrebbe quindi essere in grado di riassumere la maggior parte delle informazioni contenute nel set originale di funzionalità. In questo modo, è possibile creare una versione riepilogativa delle feature originali da una combinazione del set originale.
Nella specifica applicazione di questo progetto, le tecniche di Feature Extraction sono risultate irrilevanti poiché tutte le feature presenti all'interno del dataset sono risultate subito rilevanti sia al fine della classificazione che della predizione. Ciò si può denotare dai risultati ottenuti dalle sperimentazioni effettuate su ogni tecnica di Feature Extraction: per ogni tecnica sono state svolte 30 sperimentazioni, da ognuna delle quali è stata recuperata l'accuratezza massima, calcolata per tutti i possibili valori del parametro *n_features*.
Sia per PCA, sia per ICA e sia per LDA i grafici ottenuti sono stati i seguenti:



Tutte e tre le tecniche, sono irrilevanti in questo caso.

6.1.2 ACCURATEZZE CLASSIFICATORI E REGRESSORI

Dopo aver eseguito il tuning dei diversi iperparametri di ciascun classificatore, essi sono stati utilizzati al fine di predire la giusta fascia di prezzo di una determinata abitazione. Di seguito vengono riportati tutti i valori delle metriche utilizzate al fine di valutare la "bontà" di ciascuno dei classificatori. Tutti i classificatori, sulla base di ciò che viene detto nella sezione precedente, sono stati addestrati su dati non sottoposti a tecniche di feature extraction, perché inefficienti al fine del task.

Classificatori	Accuracy	Precision	Recall	F1-Measure
K-NN	0.992	0.990	0.991	0.991
GaussianNB	0.898	0.894	0.896	0.892
RandomForest	0.998	0.998	0.997	0.998
ExtraTrees	0.990	0.990	0.989	0.990

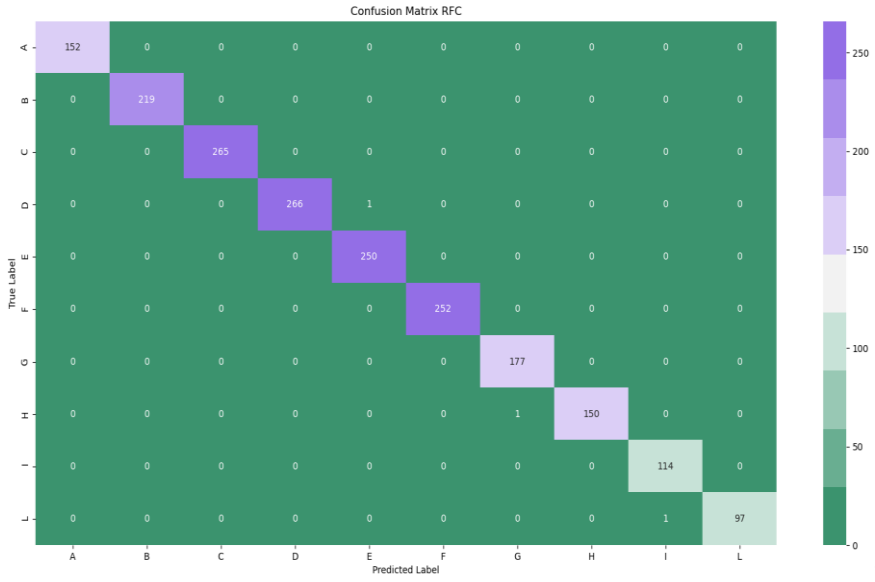
Regressori	R^2
Linear	0.864
Gradient Boost	0.968
XGBoost	0.975

5.1 CONCLUSIONE

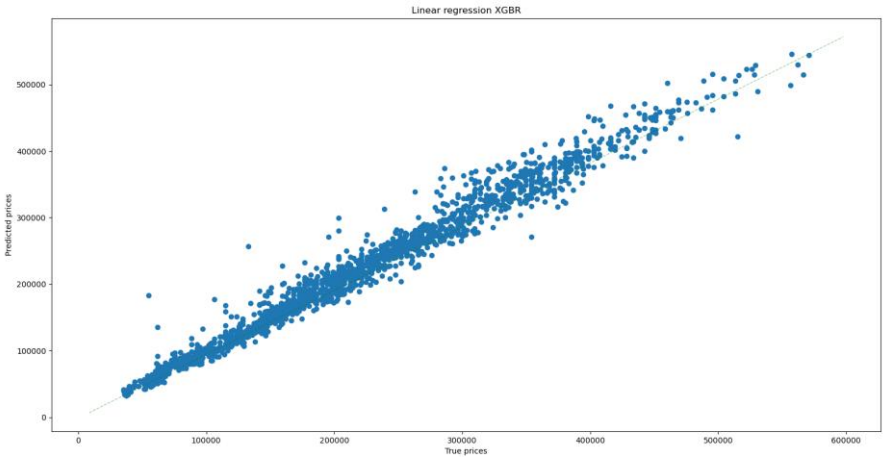
Dopo aver confrontato i diversi classificatori, si è arrivati alla conclusione di considerare il **Random Forest Classifier**. Tale scelta si è basata sulle metriche sopra riportate. Esso, visto il valore più alto di accuratezza, riesce a predire meglio quale sarà la fascia di prezzo d'appartenenza di una determinata abitazione.

Invece per quanto riguarda la predizione, dopo un'analisi dei valori si è arrivati alla conclusione che il miglior regressore è l'XGBoost poiché possiede la più alta $\mathbf{R^2}$, il che vuol dire che predice il prezzo vero e proprio di un immobile con una precisione piuttosto elevata.

Di seguito verranno mostrate la **matrice di confusione (1)** del Random Forest e la **retta di regressione (2)** dell'XGBoost.



(1)



(2)