Data and Web Mining a.a. 2020-21 [[CT0509]](https://moodle.unive.it/course/view.php?id=7914)

Quintavalle Sebastiano, matricola 878500

# Il progetto

Il progetto consiste in un task di predizione presente nella piattaforma [kaggle](https://www.kaggle.com/). Il task [**Zillow Prize: Zillow’s Home Value Prediction (Zestimate)**](https://www.kaggle.com/c/zillow-prize-1/data)mette a disposizione alcuni dataset che danno informazione riguardo alcune case di tre contee americane (Los Angeles, Orange and Ventura, California) e sul *logerror* della loro vendita tra il 2016 e il 2017.

Il ***logerror***, ossia la variabile risposta oggetto della predizione, è una **trasformazione anonimizzata del prezzo di vendita**. In particolare, è la differenza tra il logaritmo del valore predetto dall’algoritmo *Zestimate* e il vero prezzo di vendita.

# Piano di lavoro

Il modellamento dei dati e la costruzione dei modelli sono processati con l’ausilio di alcuni *JupyterNotebook*, strumento di sviluppo interattivo basato su un kernel *python3*.

Il progetto è stato diviso in 6 notebook:

1. Preparazione dei dati
2. Divisione fra contee
3. Selezione delle feature
4. Primo modello
5. Secondo modello
6. Model stacking

Il file di consegna contiene sia i notebook con estensione .*ipynb* già eseguiti nella cartella principale, che una sottocartella *pdf* dove è contenuta la loro rispettiva conversione in *.pdf*.

La motivazione divisione del lavoro in più notebook deriva da due esigenze principali. La prima è la **divisione logica e modulare** del progetto in sottotask. La seconda deriva invece da fini pratici: gli algoritmi di Machine learning hanno un tempo computazionale non banale e la macchina su cui sono stati eseguiti può impiegare anche qualche ora per eseguire un notebook; per questo motivo la divisione in sottotask è utile a **creare dei checkpoints**: al termine della esecuzione di un notebook le modifiche apportate ai dataset sono salvate in una specifica cartella sotto forma di *file.csv*; in particolare le cartelle sono:

* **dataset**,contiene i dati grezzi provenienti da kaggle.
* **preparazione**,contiene i dati preprocessati.
* **regioni**,contiene i dati preprocessati divisi per contea.
* **selezione**,contiene i dati preprocessati e divisi per contea su cui è stata fatta una selezione delle feature.

Durante alcuni algoritmi vengono inoltre prodotti dei **grafici**, che sono stati salvati nella cartella *images*. I grafici contengono:

* la visualizzazione bidimensionale delle case divise per contea (*regioni.jpg*).
* il clustering delle contee con algoritmo k-means (*k\_means.jpg*).
* il risultato di un task di classificazione sull’identificatore della contea (*classification.jpg*)
* un grafico a barre dell’importanza della feature prima della selezione (*all\_feature\_importance.jpg*)
* l’andamento di *bias*, *variance* e *mse* di un DecisionTreeRegressor al crescere del massimo numero di foglie su Train e Validation (*DecisionTreeRegressor\_train/validation.jpg*) e la visualizzazione dell’albero selezionato (*DecisionTreeRegressor.jpg*).
* l’andamento di *bias*, *variance* e *mse* di un AdaBoostRegressor al crescere del numero di modelli su Train e Validation (*AdaBoost\_train/validation.jpg*)
* l’andamento di *bias*, *variance* e *mse* di un RandomForestRegressor al crescere del numero di alberi su train e validation (*RandomForestRegressor\_train/validation.jpg*), e un grafico a barre dell’importanza delle feature del modello selezionato. (*RandomForestRegressor\_FeatureImportance.jpg*).

Segue una descrizione generale delle operazioni eseguite nei notebook, delle scelte effettuate e delle analisi dei risultati. Descrizioni più dettagliate accompagnano le singole operazioni nei notebook.

# Preparazione dei dati

Il primo task si occupa di **trasformare i dati grezzi** dei dataset forniti da kaggle. Sono qui riassunte la serie di operazioni apportate al dataset.

Il dataset fornisce per ciascun anno 2016 e 2017 due dataset: uno con una **descrizione delle case** (con circa una settantina di feature) e uno che porta informazione sulla **vendita della casa**, di cui sono a disposizione l’identificatore della casa venduta, la data di vendita e il suo *logerror*. È importante precisare che le case vendute almeno una volta sono una piccola parte della totalità del dataset.

I dataset 2016 e 2017 sono uniti e sono mantenute solo le case di cui si dispone del *logerror*; la data di vendita diventa una nuova feature (dataset X) e il *logerror* è usato come variabile risposta (dataset y).

I dati sono divisi in **Train, Validation** e **Test** in proporzione 6:2:2. Nelle successive operazioni tutte le modifiche apportate saranno giustificate alla luce dei dati del Train e riportate in maniera opaca su Validation e Test (dati che devono risultare inediti).

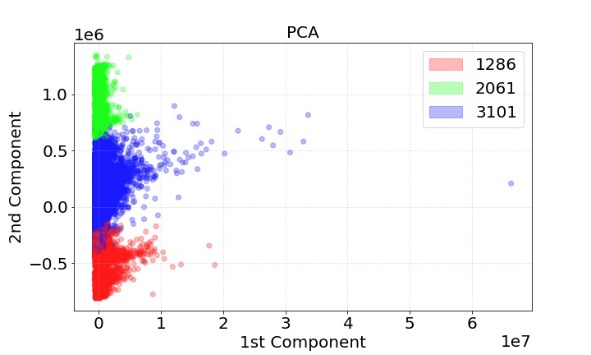
Operazioni effettuate:

* **Rappresentazione non corretta dei Nan**: per alcune feature i valori mancanti sembrano poter rappresentare un altro tipo di informazione. (Ad esempio *fireplaceflag* ha due tipi di valori: Nan e la stringa “True”, che potrebbe essere facilmente convertita in una colonna binaria).
* **Rimozione degli outlier**: rimuovo dal dataset di Train istanze con un valore di risposta estremo.
* **Rimozione colonne con alta percentuale di Nan**: rimozione di feature poche informative causa un’altissima percentuale di valori mancanti.
* **Rimozione di feature ridondanti**: analisi di variabili che portano la stessa informazione ed eliminazione di una delle due. (ad esempio, *fips* e *centurtrackblock* hanno la stessa informazione numerica ma su scala decimale differente).
* **Rimozione delle righe con molti Nan**: rimozione dal dataset di Train istanze poco significative.
* **Conversione di valori non numerici**: presenza di tre feature rappresentate come stringa, di difficile gestione da parte di algoritmi di Machine-Learning. Due di queste sono categoriali e rappresentano delle sigle: di queste sono mantenute solo le 5 più frequenti e le altre sono classificate come *rare*; con questa approssimazione si ottengono comunque dati significativi della maggior parte della preparazione senza che il numero di colonne esploda nel One-Hot Encoding. L’altra colonna è la data di vendita, convertita in un intero che rappresenta il numero di giorni trascorsi dal primo dell’anno.
* **Aggiunta di nuove feature**: inserimento di 5 nuove feature, 3 di queste basate sul rapporto tra due feature già esistenti. Le altre due sono il *logerror* medio per il mese di vendita e il *logerror* medio per le case entro una certa area.
* **Individuazione di valori discreti**: uso di un’euristica: sono classificate come discrete le variabili con meno di 30 valori distinti. Tra questi avviene una distinzione tra categoriali e ordinali, che saranno trattate in maniera distinta.
* **Missing-Flag**: aggiunta di una colonna per segnalare l’assenza di un valore per le colonne con una percentuale di valori mancanti oltre una certa soglia.
* **Riempimento dei valori mancanti**: sostituisco i valori mancanti di una colonna per le variabili numeriche e ordinali con la mediana della colonna.
* **One-Hot Encoding**: applico il One-Hot Encoding allenato sul dataset di Train per le variabili categoriali.

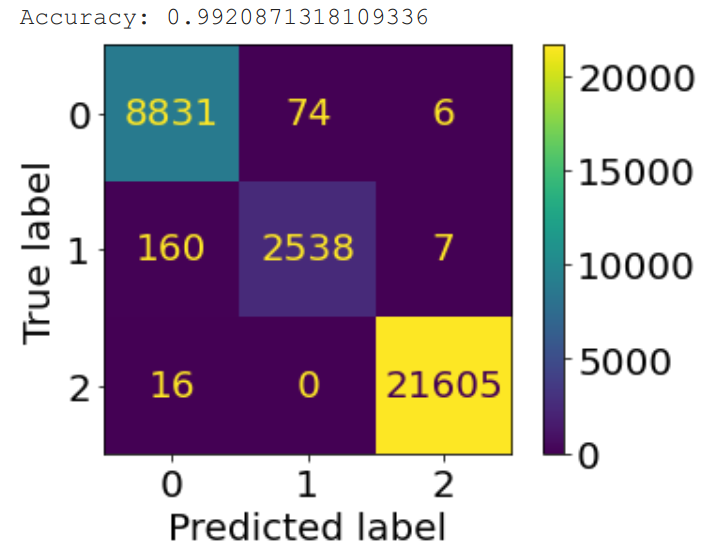
# Divisione tra regioni

Il notebook analizza se possa essere sensato **dividere il dataset in sottoinsiemi** a seconda della contea.

Per farlo viene prima fornita una rappresentazione grafica bidimensionale grazie alla PCA, che sembra favorire la tesi iniziale.



Seppur un algoritmo di clustering non riesca a individuare in maniera precisa le contee, un **task di classificazione** invece riesce nell’intento con un’ottima accuracy.



Alla luce dei risultati si deduce sia sensato dividere il dataset originale in funzione della contea.

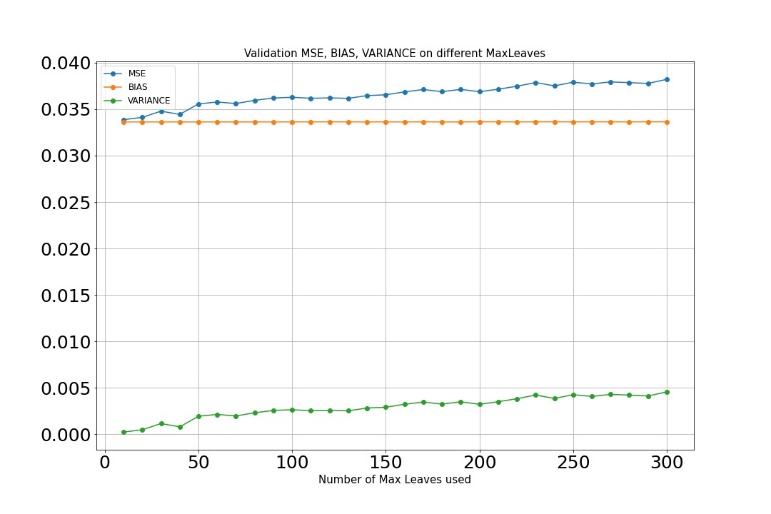
# Selezione delle feature

Complessivamente le istanze di Train, Validation e Test contano all’incirca 170mila osservazioni, ciascuna con circa 70 feature. È condotta una indagine per verificare se sia **possibile eliminare feature** poco importanti o fortemente correlate ad altre. In prima istanza è impiegata una **foresta** da cui ricavare un **ranking dell’importanza delle featur**e; il ranking è poi impiegato per analizzare il **comportamento dell’errore** aggiungendo le feature una per volta. In seconda istanza è adoperata una **eliminazione ricorsiva delle feature 1-step** per ottenere le migliori feature per quel modello.

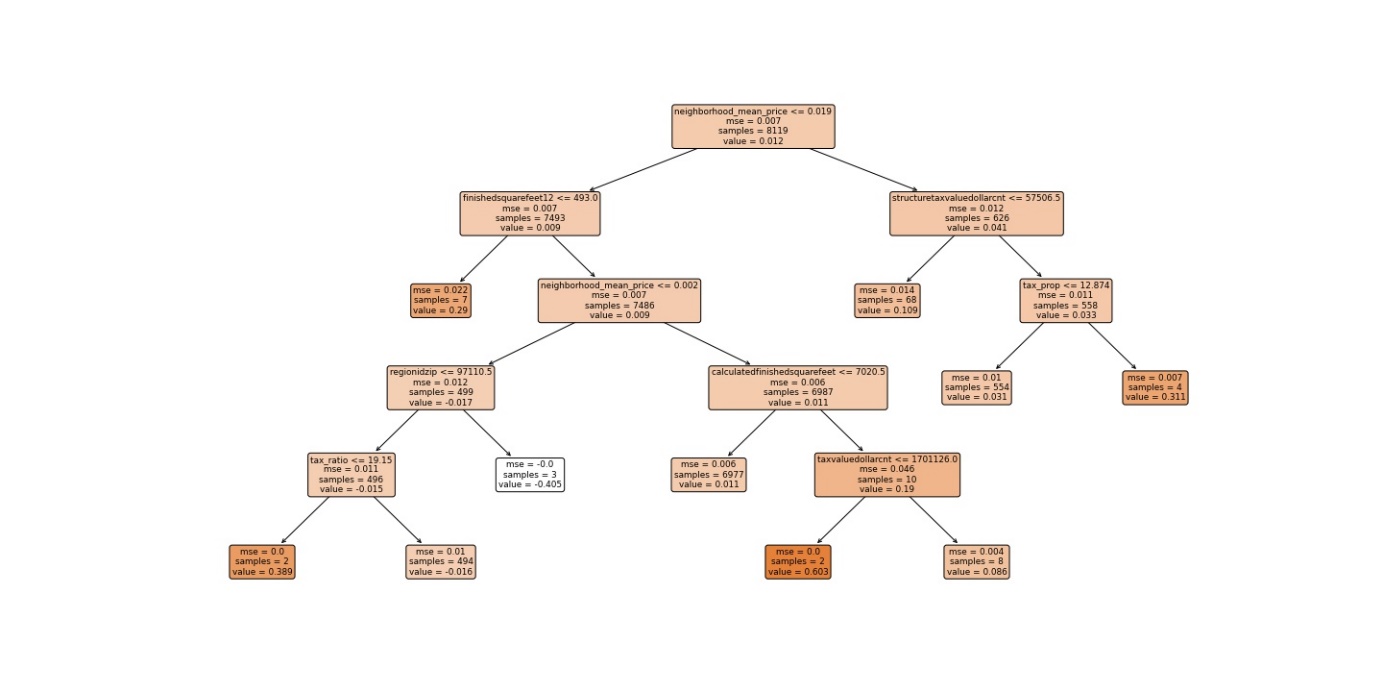
I risultati delle due analisi sono comparabili, entrambe selezionano una cinquantina di feature per la prima contea e una trentina per la seconda e la terza. Le feature con maggiore importanza sono quelle legate alle tasse, probabilmente perché capaci di catturare più aspetti della casa in un unico dato. Le feature aggiunte in fase di preprocessing sono tutte selezionate e hanno un’importanza rilevante. Mantengo i risultati della selezione ricorsiva perché più sensibile alla correlazione tra feature.

# Primo modello

Il primo modello costruito è un **albero di decisione**. Si fa uso di una funzione per fare **tuning dell’iperparametro** *max\_leafes\_node*, ossia il **massimo numero di foglie** dell’albero: gli alberi sono allenati sul dataset di Train e l’errore è calcolato sul dataset di Validation. Il modello selezionato mantiene come massimo numero di foglie il valore che testato su Validation ha prodotto l’errore minore. L’andamento su Validation è simile per tutte e tre le contee:



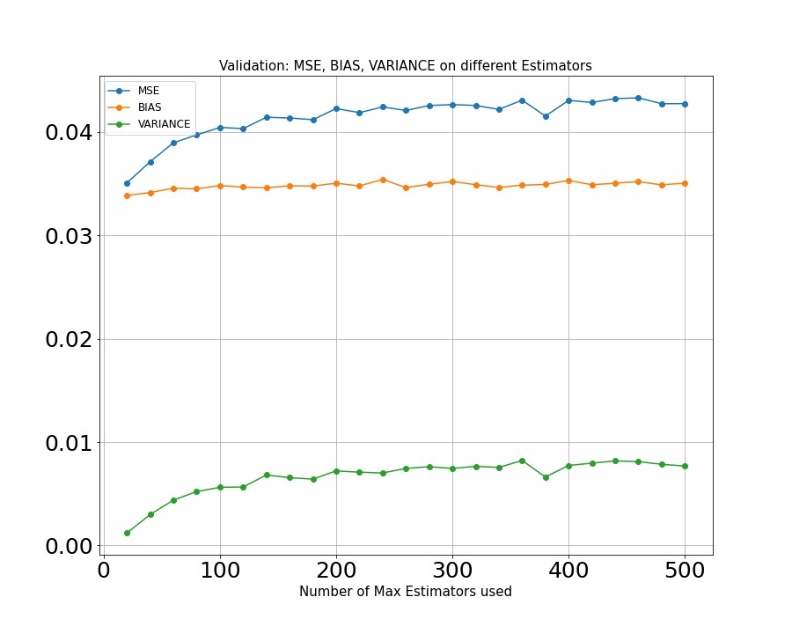
Il ***bias*** **rimane** pressoché **costante** mentre la varianza aumenta: il modello selezionato è quello con il minimo numero di foglie testato. Analizzando l’albero:



Si nota che l’albero non è in grado di specializzarsi in maniera bilanciata: una foglia contiene la maggior parte delle osservazioni la cui predizione è molto simile al *base-rank*.

Per provare a ridurre il *bias* uso un algoritmo di **boosting**, capace inoltre di migliorare l’accuracy partendo anche da *weak-learners­*, come gli alberi selezionati.

Usando un AdaBoostRegressor il cui iperparametro n\_*estimators* è tunato in maniera simile al punto precedente:

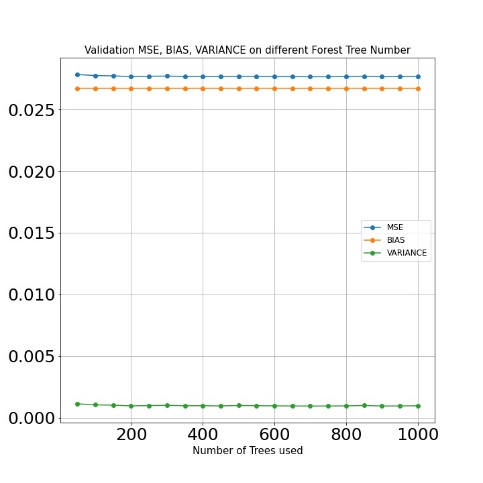


Il risultato non è quello sperato: il ***bias* rimane stabile**, mentre si registra un **aumento della varianza**. Il *bias* non riesce a scendere una certa soglia perché l’albero di decisione non riesce ad imparare dai dati. Il *mean-squared error* delle predizioni sul test è molto grande e simile al *base-rank*.

Questo potrebbe anche essere dovuto al fatto che la variabile da predirre *logerror* non sia influenzata unicamente dal prezzo di vendita della casa, ma anche dalla predizione fatta da *Zestimate*, di cui però si ha un’informazione molto ridotta.

# Secondo Modello

È costruita una **foresta**, il cui **iperparametro** riguardante il **numero di alberi** è tunato usando un dataset di Validation. L’andamento dell’errore è il seguente:



Il ***bias*** rimane **costante** mentre la varianza sembra **diminuire leggermente al crescere del numero di stimatori** ma in maniera non significativa.

Calcolando l’errore quadratico medio sul dataset di test questo non è soddisfacente poiché non abbatte in maniera netta il *base-rank* del modello banale. Le motivazioni sono probabilmente analoghe al punto precedente.

La feature-importance degli alberi premia le variabili legate alle tasse, ma anche le colonne aggiunte in fase di preparazione hanno un’importanza rilevante.

# Model Stacking

Costruisco un **modello di regressione lineare** su un dataset le cui colonne sono le predizioni dei tre modelli costruiti in precedenza per l’insieme di Train. Combino così le **predizioni di modelli eterogenei** assegnando un peso a ciascuno.

Il modello ottenuto ha un errore sul Test nettamente migliore dei tre modelli precedenti: se i tre modelli non riuscivano ad abbattere la soglia del base-rank, ora **l’errore è circa dieci volte più piccolo**.

### Predizioni Migliori

Le case su cui l’errore è molto piccolo sono quasi sempre case di media dimensione, spesso di città differenti, i cui valori sono vicini alla media del dataset e che dunque si trovano in una regione molto popolata. Queste case si trovano in un’area del dominio densa per cui la predizione è meno incerta.

### Predizioni Peggiori

Si evidenziano due tipi di case il cui errore sulla predizione è molto ampio:

* Case molto grandi, con una metratura significativa e un alto numero di stanze. Probabilmente queste case ricadono in uno spazio di dominio decisamente poco popolato in cui l’incertezza è alta.
* Case con dati anomali, come ad esempio zero bagni e zero camere da letto. Probabilmente causa di dati mal compilati e dunque non significativi.