Un Ensemble di Modelli Boosting Per La Previsione degli Immobili nella Contea di King

Federico Ravenda[1]

[1]**Università degli Studi di Milano-Bicocca, CdLM Clamses - STAT, matricola nr. 829449**

ABSTRACT

Nella presente trattazione sono stati utilizzati multipli approcci per lo studio del dataset contenente informazioni relative al prezzo degli immobili nella contea di King e dintorni, nello stato di Washington, USA. In particolare sono state utilizzate diverse tecniche di ingegnerizzazione delle variabili e di Preprocessing ed è stato valutato l'impatto di queste sulle performance di differenti modelli .
In un contesto di *competizione*, l'obiettivo principale è quello di ottenere le migliori performance su un set di dati di cui non si conoscono i veri valori delle variabili risposta.
L'obiettivo dell'analisi è stato, dunque, fin da subito, quello di andare a minimizzare una certa misura di errore - il *Mean Absolute Error* (**MAE**) - sfruttando alcuni dei modelli più recenti e performanti basati sui metodi ensemble.
Le analisi sono state svolte utilizzando il linguaggio R. In particolare, sono state utilizzate le librerie `h2o` [1] nella fase di *Preprocessing* e *Feature Engineering*, `mlr3` [2] per la fase di *Modelling* e `Keras` [3] per implementare la sezione relativa all'*Entity Embedding*.
Al seguente link https://github.com/Fede-stack/Applied-Predictive-Modelling è presente il repository contenente tutti i files utilizzati per le analisi.

Keywords: R, Forecasting, House Price Prediction

CONTENTS

1 INTRODUZIONE AL DATASET – A QUICK OVERWIEW

I dati di riferimento si compongono in:

- Un dataset di Training, costituito da 17'293 osservazioni
- Un dataset di Testing, costituito da 4'320 osservazioni

Entrambi i dataset contano *18* variabili esplicative relative alle caratteristiche degli immobili (posizione geografica, metratura, zipcode di riferimento, numero di stanze, ...) presenti, principalmente, nella contea di King, nello stato di Washington.

La Figura 1 mostra la dislocazione degli immobili e la loro fascia di prezzo (i punti arancioni e rossi scuri rappresentano le fasce più alte, quelli verdi e blu quelle più basse). Si osserva come vi sono delle regioni in cui il livello dei prezzi è più alto: un caso è Mercer Island; qui, i prezzi delle case, sono mediamente più alti che nelle zone circostanti.

La variabile *target*, oggetto di previsione, è la variabile '*Price*', che fa riferimento al prezzo di ciascun immobile

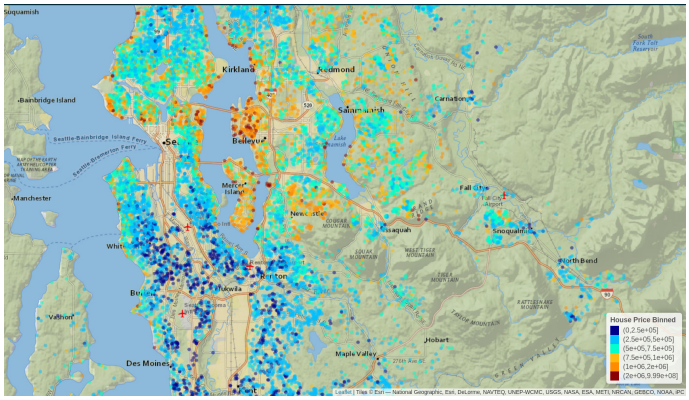

Figure 1. Mappa degli immobili con relativa fascia di prezzo

(per ridurre l'elevata dispersione si è opportunamente deciso di effettuare una trasformazione logaritmica della variabile). Non sono presenti valori mancanti in nessuno dei due set di dati.
L'obiettivo delle analisi, come detto in precedenza, è quello di andare a minimizzare una misura di errore, il MAE, sui dati di testing, di cui non si conosce la vera etichetta.

Al fine di valutare le performance dei modelli utilizzati si è scelto di creare un set di validazione tramite holdout, con uno splitting $80\% - 20\%$ rispetto al dataset di training. Questa porzione di dataset è stata utile nella fase finale delle analisi per scegliere quali modelli utilizzare e quali escludere per il computo finale delle previsioni.

2 EDA

Le variabili del dataset sono così riassunte:

- **date_sold**: Data relativa alla vendita dell'immobile. Il periodo di riferimento considerato all'interno del dataset va dal *2014-05-02* al *2015-05-27*.
- **price**: Prezzo dell'immobile (in scala log10)
- **view**: Indicatore della qualità dell'immobile (indice tra 0 e 4)
- **condition**: Indicatore che fa riferimento alla condizione dell' appartamento (indice tra 1 e 5)
- **floors**: Numero di piani
- **bedrooms**: Numero di stanze da letto
- **bathrooms**: Numero di stanze da bagno
- **waterfront**: variabile binaria (assume modalità '1' se l'immobile si affaccia ad un corso d'acqua, '0' alternativamente)
- **zip_code**: codice postale della proprietà
- **sqft_living**: Dimensione della zona vivibile espressa in m^2
- **sqft_lot**: Dimensione del lotto in m^2
- **sqft_above**: Dimensione della zona vivibile al di sopra del suolo in m^2
- **sqft_basement**: Dimensione dello spazio vivibile al di sotto del suolo in m^2
- **yr_built**: Anno in cui è stato costruito l'immobile
- **year_renovated**: Anno in cui l'immobile è stato ristrutturato, se non è stato ristrutturato viene indicato l'anno di costruzione
- **nn_sqft_living**: Dimensione media dello spazio vivibile dei 15 immobili più vicini a quello considerato (espressi in m^2)
- **nn_sqft_lot**: Dimensione media dei lotti di terra per i 15 immobili più vicini (espressi in m^2)
- **latitude**: Latitudine
- **longitude**: Longitudine

La maggior parte delle covariate presenti nel dataset ha natura numerica. Alcune di queste, riportate in rosso, hanno natura *continua*, mentre quelle citate in blu, hanno natura *discreta*.

La variabile *zip_code* è invece di tipo *categoriale* con un numero di modalità pari a 70. Utilizzando un approccio classico, *dummy variable*, l'idea è quella di creare $(70 - 1)$ variabili dicotomiche (0-1). La dimensionalità del dataset aumenta considerabilmente (il numero di variabili dummy è di gran lunga superiore rispetto al numero di variabili originali) e questa situazione porta a dover affrontare due tipi di problemi:

- Risorse computazionali onerose
- Curse of Dimensionality

Per questo motivo, durante la trattazione, verrano considerati anche degli approcci alternativi, più recenti, rispetto all'approccio dummy (target encoding, entity embedding).

Dalla Figura 2 si osserva come alcune variabili sono fortemente correlate con la variabile risposta: ci si aspetta, dunque, che queste contribuiscano sensibilmente alla spiegazione della variabile target.
La Figura 3, in basso a destra, mostra uno scatterplot tra la variabile esplicativa *sqft_living* e la variabile risposta *Price*. Si aggiunge una curva blu in sovraimpressione che

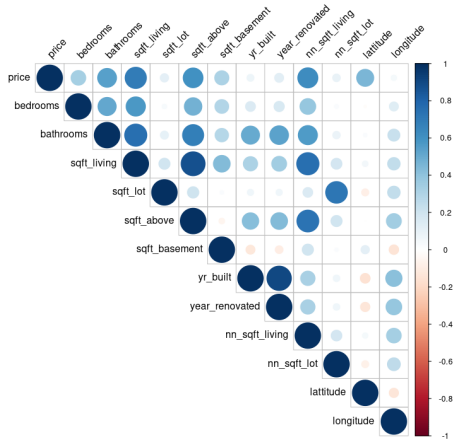

Figure 2. Grafico delle correlazioni tra variabili

rappresenta la regressione **LOESS**. Si osserva come la covariata in questione, come lecito aspettarsi dalla forte correlazione con la risposta, sia estremamente utile a spiegare la variabile risposta.
I restanti grafici fanno riferimento a boxplot relativi alla variabile prezzo condizionati ad alcune delle variabili categoriali selezionate: si osserva come la presenza di un corso d'acqua nei pressi dell'immobile sia un fattore discriminante per la variabile prezzo. Anche la variabile *Condition*, come lecito aspettarsi, discrimina correttamente la variabile risposta: più la condizione è buona, più il prezzo dell'immobile risulterà, mediamente, più alto.
La variabile *month* sembra non essere così discriminante come le precedenti: tuttavia si nota come, in alcuni mesi, probabilmente, si è più inclini a fare degli investimenti per immobili con prezzi mediamente più alti (soprattutto nei mesi estivi).

3 PREPROCESSING & FEATURE ENGINEERING

Gli algoritmi di apprendimento utilizzano i dati di input per produrre previsioni. Molto spesso, tuttavia, i dati forniti potrebbero non essere sufficienti per creare un buon modello. In questi casi può essere utile estrarre dalle variabili informazioni aggiuntive da dare in input al modello. Un'efficace fase di *Feature Engineering* può rivelarsi cruciale per migliorare le performance degli algoritmi.

L'ingegnerizzazione delle variabili ha principalmente due obiettivi:

- Preparazione del set di dati di input di modo che questi siano compatibili con i requisiti dell'algoritmo di apprendimento.
- Migliorare le prestazioni dei modelli apprendimento.

Di seguito, verranno approfondite le diverse tecniche utilizzate per creare nuova informazione a partire dalle variabili a disposizione.

3.1 Preprocessing

Durante la fase di esplorazione dei dati sono state rilevate delle incongruenze tra gli zip code forniti dal dataset e quelli ricavati sfruttando le funzioni di *reverse geocoding* presenti nella libreria `tidygeocoder` di R. Nonostante l'incoerenza, si è deciso di tenere le informazioni provenienti dal dataset originario; a queste sono state aggiunte ulteriori variabili, estrapolate utilizzando l'interprete Python, con riferimento al nome della città e alla contea in cui sono dislocati gli immobili (a partire dalle informazioni di latitudine e longitudine).

Al seguente link è stato possibile estrarre e integrare informazioni relative agli indici dei prezzi delle case nello stato di Washington a livello trimestrale durante l'arco temporale osservato.

3.2 Feature Engineering

Come si osserva dalla Figura 2 alcune covariate sono fortemente correlate fra di loro, i.e. collineari (e.g. "sqft_living"-"sqft_above", "year_built"-"year_renovated", ...), il che può rappresentare una ridondanza di informazione.
Si è proceduto, dunque, a creare delle nuove variabili a partire da quelle originali. Le variabili create sono le seguenti:

- *rooms*: Come somma delle variabili *bathroom* e *bedrooms*, fortemente correlate fra di loro
- *is.renovated*: Si tratta di una variabile *dummy*, se le variabili *year_renovated* e *year_built* coincidono assume valore 1.
- *years_back*: Calcola gli anni che sono passati dalla data di costruzione dell'edificio.
- *is.below*: Variabile *dummy*, se la casa possiede uno spazio al di sotto del suolo la variabile assume valore 1.
- *percentage_below*: Percentuale di metratura della casa che appartiene all'area relativa al sottosuolo.
- *is.pierce.county*: Variabile *dummy*, se la variabile *county* assume la modalità *"Pierce County"* allora la variabile dummy assumerà valore 1. Si è deciso di creare questa variabile dicotomica perchè, a differenza delle altre contee (King, Snohomish e Kitsnap) i prezzi delle case nella contea di Pierce sono, mediamente, molto più bassi.
- *is.mercer.island*: Variabile *dummy*, si è osservato (anche visivamente dalla mappa in Figura 1) come l'area in cui siano concentrati i prezzi delle case più alti sono localizzati a Mercer Island.

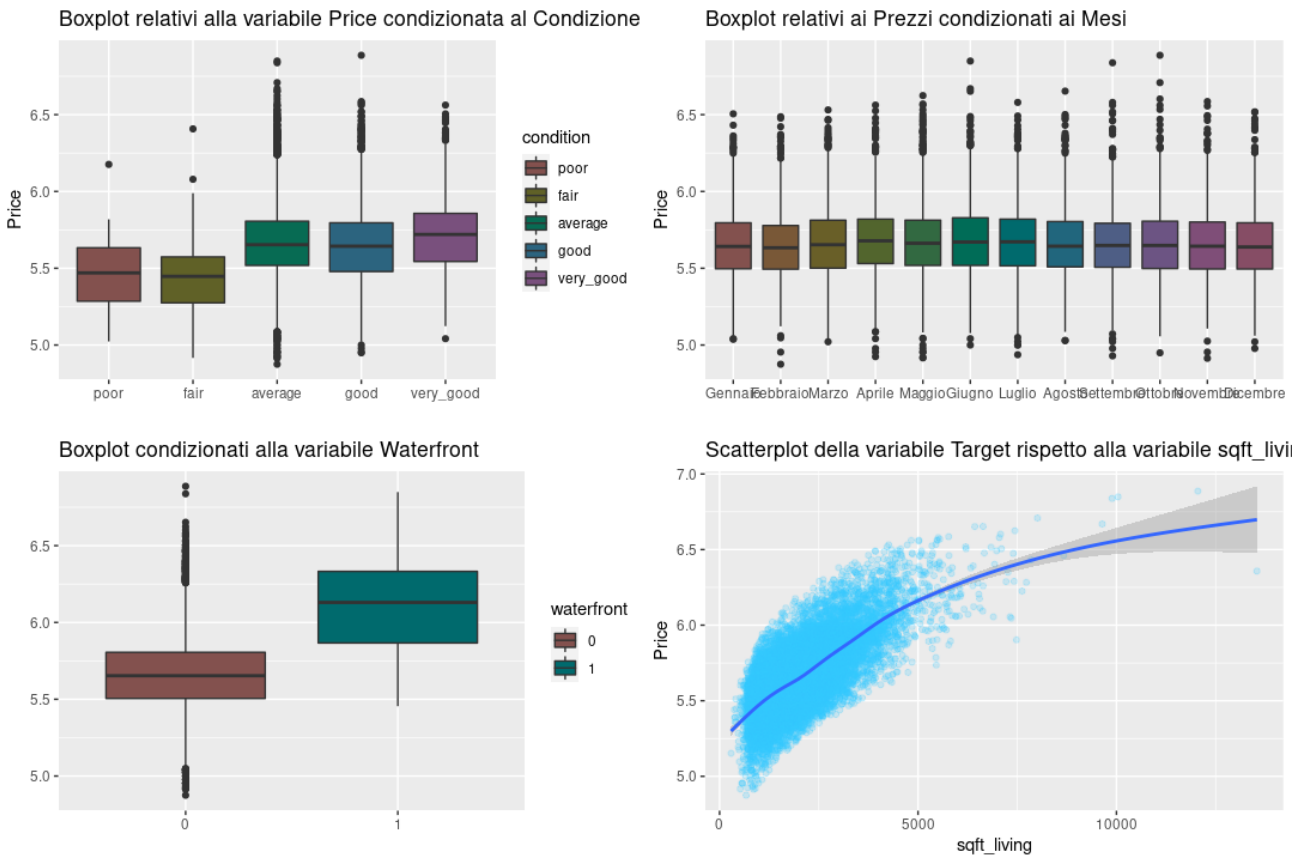

Figure 3. Da sinistra verso destra: In alto i boxplot condizionati alla variabile price delle variabili, rispettivamente, "condition" e "month". In basso il boxplot condizionato della variabile "waterfront" e lo scatterplot con la variabile "sqft_living" sull'asse delle ascisse e "price" sull'asse delle ordinate

Una volta inserite le nuove variabili manualmente create (sopra elencate) si è proceduto ad eliminare le seguenti variabili: *date_sold, yr_built, year_renovated, sqft_above, sqft_basement, sqft_lot, bedrooms, bathrooms*.

3.2.1 Isolation Forest

Isolation Forest [4] è un algoritmo simile, come idea, a Random Forest ed è costruito sulla base di alberi decisionali. Isolation Forest, tuttavia, ha come obiettivo quello di identificare valori anomali o *outliers*, isolando ciascun punto del dataset sulla base della sua maggiore o minore distanza dagli altri punti.
L'idea principale è che è meno probabile che le osservazioni che compiono un lungo percorso per giungere alle foglie dell'albero siano anomalie poiché hanno richiesto più splitting per essere isolate dall'algoritmo. Allo stesso modo, i campioni che finiscono in rami più corti indicano osservazioni anomale poiché è più facile per l'albero separarli da altri campioni.
Nel caso in esame è stata utilizzato una generalizzazione di Isolation Forest, *Extended Isolation Forest* [5]. Si è deciso di utilizzare come iperparametro relativo al numero di alberi **700** e gli scores di anomalie restituiti dall'algoritmo sono stati utilizzati per creare una nuova variabile da aggiungere al dataset.

3.2.2 Target Encoding

Nel dataset in oggetto sono presenti alcune features di natura categoriale, una in particolare, *zip_code*, come già anticipato, possiede un numero di modalità piuttosto alto.
Il target encoding è un approccio che proviene dal mondo Bayesiano: questo, infatti, usa l'informazione della variabile target per codificare le variabili categoriali. L'idea è quella di calcolare la media della variabile target per ciascuna modalità di ciascuna variabile categoriale e sostituire la categoria con il valore medio ottenuto.
Questo approccio permette di creare un *link* tra la variabile target e la variabile categoriale (che, diversamente, con il One Hot Encoding, non si riuscirebbe a creare).
Una volta effettuato il target encoding sulla variabile "*zip_code*" si è aggiunta al dataset la variabile associata a questi nuovi valori codificati e la si è chiamata "*zip_code_te*".

3.2.3 Feature Binning

Il Binning (o discretizzazione) è una tecnica utilizzata per trasformare una variabile continua in una variabile categorica. Questo metodo introduce non-linearità e tende a

migliorare le performance del modello. Si è effettuato un binning per tre delle variabili continue presenti nel dataset (*"sqft_living", "nn_sqft_living e "nn_sqft_lot"*), scegliendo come numero di bins **20** dopo aver valutato differenti configurazioni manualmente. Sono state, dunque, aggiunte 3 nuove features al dataset contenenti le variabili "binnate".

3.2.4 Entity Embedding

Gli Embeddings, ormai ampiamente noti nel campo dell' *Information Retrieval*, sono delle rappresentazioni nello spazio ridotto in cui si possono tradurre dei vettori *High Dimensional*.
Gli Embeddings sono un'utilissima soluzione nella gestione delle variabili categoriali, poichè evitano molte delle insidie relative ad un One Hot Encoding.
Questo metodo offre 2 principali vantaggi:

- Si limita il numero di colonne del dataset.
- Gli Embeddings, per natura, raggruppano variabili intrinsecamente simili.

L'idea è quella di creare degli Embeddings per la variabile categoriale *zip_code*, concatenarli con il dataset di partenza e allenare un modello per apprendere dagli Embeddings costruiti [6].
La dimensione dell'embedding definisce lo spazio in cui mappiamo le variabili categoriali. Jeremy Howard [7] fornisce una *rule of thumb* sul numero di dimensioni da utilizzare: size of embedding = *min(50, numero di categorie/2)*. Nel caso in esame poniamo questo parametro uguale a **35**.

4 MODELLING & RESULTS

La fase di Modelling è stata svolta utilizzando il pacchetto `mlr3`, un'ecosistema sufficientemente maturo che ospita la maggior parte dei modelli di *machine learning* implementati in `R`.
Dato che l'obiettivo dell'analisi è quello di ottenere le migliori performance (si vuole minimizzare il MAE come misura di bontà del modello) su un test set, si è ricorso all'utilizzo di alcuni dei modelli che, generalmente, tendono ad essere quelli più utilizzati anche nelle competizioni *kaggle* nel contesto di dati strutturati [1] e sono basati su *Boosting*, per ottenere i risultati migliori. Nella fattispecie, sono stati utilizzati i 4 seguenti modelli di regressione: [2]

1. *XGBoost*
2. *CatBoost*
3. *LightGBM*
4. *BART*

Per ciascuno di questi modelli è stata effettuata *ottimizzazione degli iperparametri*. Dato che il tempo di allenamento di questi algoritmi (fatta eccezione per lightGBM) è piuttosto alto (in alcuni casi > 5 minuti) si è deciso di utilizzare un approccio di ***ottimizzazione bayesiana*** con *cross-validazione* e budget ridotto.
Si è utilizzata, come baseline per valutare le performance dei modelli, il valore di MAE ottenuto a partire da un modello di regressione lineare con tutte le esplicative (**MAE**: *0.0599*).
In Tabella 1 vengono riportati i risultati ottenuti dai modelli elencati in termini di MAE. Questi risultati sono ottenuti sul set di validazione (contenente il 20% delle osservazioni originarie). I modelli sono stati allenati sul dataset in cui non sono presenti gli embeddings, fatta eccezione per CatBoost che è stato allenato su entrambi i datasets (nel dataset in cui sono presenti gli embeddings è stata eliminata la variabile *zip_code*). Si osserva come Catboost, senza Entity Embedding, sia il modello più performante.

Table 1. Score, in termini di Mean Absolute Error, dei modelli utilizzati sul set di validazione.

	MAE
CatBoost	0.0489
LightGBM	0.0520
BART	0.0515
XGBoost	0.0501
CatBoost con Entity Embedding	0.0492

Si è deciso, infine, di utilizzare un Ensemble (una semplice media delle previsioni di questi modelli) che ha ulteriormente migliorato lo score sul set di validazione (**MAE: 0.0402**).

[1] https://medium.com/analytics-vidhya/xgboost-lightgbm-and-other-kaggle-competition-favorites

[2] *XGBoost* [8] è un'implementazione alternativa al classico *gradient boosting*, scalabile e altamente performante. In XGBOOST gli alberi vengono costruiti in parallelo, anzichè in maniera sequenziale come nei *gradient boosting decision trees*. L'algoritmo scansiona i valori del gradiente e utilizza queste somme parziali per valutare la qualità delle divisioni ad ogni possibile split nel set di training. A differenza degli altri due modelli citati, XGBOOST, nella maggior parte delle sue implementazioni, non è in grado di gestire variabili categoriali.

CatBoost [9] è un algoritmo basato su alberi decisionali e gradient boosting, come XGBOOST. La particolarità di questo algoritmo risiede nel fatto che performa particolarmente bene in contesti di dataset contenenti un numero di variabili categoriali elevato.

LighGBM [10] è, anch'esso, un algoritmo basato su gradient boosting, con la caratteristica di essere particolarmente rapido da un punto di vista computazionale. La particolarità di questo modello, a differenza degli altri algoritmi basati su alberi, risiede nel fatto di costruire l'albero verticalmente, anzichè orizzontalmente. In sostanza gli *splits* avvengono nella foglia di ciascun albero e non nei nodi intermedi. L'algoritmo cerca la foglia con la funzione di perdità più alta da lasciare crescere.

Bart [11] è l'acronimo di Bayesian Additive Regression Trees. Gli alberi di regressione additiva bayesiana (BART) sono simili ai metodi basati su boosting in quanto ogni albero contribuisce alla previsione finale come weak learner. Per evitare l'overfitting, BART utilizza una a priori che forza ciascun albero a spiegare un subset limitato di relazioni tra covariate e variabile risposta.

5 CONCLUSION

Durante le analisi ci si è voluti concentrare, in particolar modo, sulle fasi di Preprocessing e Feature Engineering che hanno notevolmente migliorato le performance di ciascuno dei modelli selezionati (in particolar modo Isolation Forest e Feature Binning).
Il modello che è stato utilizzato per la previsione sul test set non labellato è l'Ensemble dei modelli riportati in Tabella 1: questi sono stati allenati sull'intero dataset e, per ciascuna osservazione del test set, è stata effettuata una media degli algoritmi selezionati, in modo da ottenere uno stimatore più robusto.

REFERENCES

[1] Aiello, S., Eckstrand, E., Fu, A., Landry, M. & Aboyoun, P. Machine learning with r and h2o. *H2O booklet* **550** (2016).

[2] Becker, M. *et al.* mlr3 book. *URl: https://mlr3book. mlr-org. com* (2020).

[3] Arnold, T. B. kerasr: R interface to the keras deep learning library. *J. Open Source Softw.* **2**, 296 (2017).

[4] Liu, F. T., Ting, K. M. & Zhou, Z.-H. Isolation forest. In *2008 eighth ieee international conference on data mining*, 413–422 (IEEE, 2008).

[5] Hariri, S., Kind, M. C. & Brunner, R. J. Extended isolation forest. *IEEE Transactions on Knowl. Data Eng.* **33**, 1479–1489 (2019).

[6] Guo, C. & Berkhahn, F. Entity embeddings of categorical variables. *arXiv preprint arXiv:1604.06737* (2016).

[7] Howard, J. & Gugger, S. *Deep Learning for Coders with fastai and PyTorch* (O'Reilly Media, 2020).

[8] Chen, T. *et al.* Xgboost: extreme gradient boosting. *R package version 0.4-2* **1**, 1–4 (2015).

[9] Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V. & Gulin, A. Catboost: unbiased boosting with categorical features. *Adv. neural information processing systems* **31** (2018).

[10] Ke, G. *et al.* Lightgbm: A highly efficient gradient boosting decision tree. *Adv. neural information processing systems* **30** (2017).

[11] Chipman, H. A., George, E. I. & McCulloch, R. E. Bart: Bayesian additive regression trees. *The Annals Appl. Stat.* **4**, 266–298 (2010).