uc3m | Universidad Carlos III de Madrid

University Degree in Economics

2022-2023

Bachelor Thesis

Econometrics meets AI: Predicting Airbnb prices using Machine Learning

José Jaén Delgado

NIU: 100385929

1st Andrés Modesto Alonso Fernández

2nd Ricardo Aler Mur

Getafe, 2023

Abstract

Predictive models for Airbnb rental prices are built using Data Mining techniques and Machine Learning algorithms. The goal is to provide an accurate and scalable statistical toolkit for economic agents to make optimal decisions. It is found that Artificial Intelligence models outperform traditional econometric methods at the expense of interpretability, for which a solution is proposed.

Keywords: Algorithms, ML Modeling, Bayesian Statistics, XAI

Mediante técnicas de minería de datos y algoritmos de Machine Learning se estiman diferentes modelos predictivos para precios de Airbnb. El objetivo es proveer a los agentes económicos de una herramienta estadística precisa y escalable para tomar decisiones óptimas. Se encuentra que los modelos de Inteligencia Artificial superan en rendimiento predictivo a los métodos econométricos clásicos al precio de una menor interpretabilidad, para lo cual se propone una solución.

Keywords: Algoritmos, Modelización ML, Estadística Bayesiana, XAI

Contents

List of Figures

List of Tables

1 Introduction

"*Econometrics is the original Data Science*" Joshua Angrist (2020)

Economic agents face various utility optimization problems on a daily basis. Firms seek to maximize revenue subject to a cost function and input constraints. Households, on the other hand, choose optimal consumption bundles given a specific budget and individual preferences. Likewise, central banks conduct monetary policy by setting interest rates to attain price stability, maximizing social welfare.

Nowadays, society is immersed in the Big Data era, where numerous data sources are available for agents to make informed decisions. Pernagallo and Torrisi (2019) argue that real time communication networks and continuous technological breakthroughs result in information overload, ultimately overwhelming economic agents. What used to be scarce data has become massively accessible to the public.

Hardly any market is resilient to this issue and even experienced users may struggle to effectively process data to make profitable transactions. Agents might benefit from applying Machine Learning (ML) to large datasets.

Consequently, Artiticial Intelligence (AI) could be used to overcome agents' limited computational capacity in the form of ML and Deep Learning (DL) models. However, predictive models that stem from sophisticated algorithms suffer from the so-called 'black box' problem. Said issue is a longstanding concern related with the lack of interpretability of AI models. Rudin (2019) reviewed the tradeoffs of favoring such models over interpretable ones and concluded that the former are not always necessary to successfully fulfill predictive tasks.

In this project predictive accuracy and model interpretability commutation is studied for the real estate market. An AI framework for solving economic agents' utility optimization problem is proposed using online Airbnb listings data. Several ML and DL models will be built to accurately predict rental pricing in Los Angeles for June 2022, providing eXplainable Artificial Intelligence (XAI) methods to understand contributing factors to final predictions.

The rest of the document is structured as follows: Section II briefly introduces concepts and definitions related to AI. Section III reviews AI applications in Econometrics. Section IV presents the utility maximization problem that is solved for real estate economic agents, describes the relevant dataset and presents Statistical Inference results. Section V explains ML notation and methodological aspects. Section VI provides insights into modeling and XAI. Finally, Section VII concludes and identifies some limitations of the AI framework.

2 Basic Concepts of AI

Grewal (2014) defines AI as "*the mechanical simulation system*" of human intelligence that collects knowledge and information for "*processing intelligence of universe and disseminating it to the eligible in the form of actionable intelligence*". Indeed, what has made AI stand out is the ability to build intelligent systems that reproduce human cognition.

Machine Learning, as expounded by Mithell (1997), is a subfield of AI "*concerned with the question of how to construct computer programs that automatically improve with experience*". When the family of learning algorithms used for predictive tasks are Neural Networks, ML is called Deep Learning. DL specifically focuses on developing various model architectures that mimic biological neural networks.

Despite being used almost interchangeably, AI models and algorithms are not identical concepts. An algorithm is "*a procedure that is run on data to create a model*" (Brownlee, 2020). Algorithms *learn* by being fit to datasets. Contrastingly, a model is the output of AI algorithms, as well as the set of assumptions and restrictions on the joint distribution of the predictors and target variable. Thus, an AI model encompasses several algorithms and defines theoretical and empirical rules followed by the latter.

Even though ML and DL are usually linked with Data Science rather than Statistics, a considerable number of AI algorithms are built upon the latter (if not the vast majority). Not coincidentally, as it will be rapidly noticed by Data Analytics professionals such as econometricians and statisticians when reading Section V, AI algorithms extend or slightly change 'traditional' quantitative methods.

3 Econometrics and AI

In fact, econometricians have been using ML and DL algorithms for some time. For example, Angrist and Frandsen (2019) resorted to ML for automating the selection of control variables, reducing bias in the reduced form equation of the Two-Stage Least Squares (TSLS) estimator. With respect to policy evaluation, Dube et al. (2021) estimated the impact of minimum wages on employment and income with Classification And Regression Tree (CART) algorithms, greedy function approximations and regularized logistic regression. Hansen et al. (2018) studied how external communication of internal deliberation impacted monetary policy decisions of the Federal Open Market Committee (FOMC) using Natural Language Processing. These are just few examples of how econometricians can benefit from adopting ML.

With respect to the specific study of rental prices, econometricians have largely focused on hedonic models, a class of linear regression models that estimates the monetary value contribution of different variables such as location, air pollution and number of rooms to prices. Apart from deriving causal effects, sheer prediction was also in the interest of certain research.

Recently, ML and DL algorithms have been been widely adopted by econometricians, showing a considerable improvement in predictive performance over OLS estimation method. Limsombunchai et al. (2004) compared Artificial Neural Networks with classical hedonic price models and concluded that DL algorithms outperformed the latter in predictive terms. Embaye et al. (2021) resorted to algorithms such as Random Forest and Gradient Boosting Machine, procuring better results than OLS estimation. It was noted, however, that the explanatory capacity of ML algorithms was scarce.

Even though AI algorithms clearly ousted OLS, this is not to say that it is possible to adopt them for every purpose. Econometrics is the "*the application of mathematics and statistical methods to the analysis of economic data*" (Kayode and Tang, 2013), mainly aiming to establish causality between economic variables. ML and DL are mostly about finding functions that best map covariates with the target variable. Optimal data fitting is not enough to explain relationships between variables, let alone derive causal effects.

Moreover, lack of interpretability is hindering human capacity to oversight and understand the predictions of AI models. While econometricians have developed numerous statistical tests and draw much of their attention to correctly interpreting model outputs, the AI community's focalpoint has been refining performance of algorithms. Nevertheless, there is growing interest in building mathematical tools for model interpretability and XAI methods will be applied to the champion model (best performing algorithm) in section VI.

4 Utility Maximization Problem & Airbnb Data

4.1 Utility function

The statistical toolkit to be built with AI algorithms serves the purpose of assisting economic agents in making optimal decisions. Naturally, a utility function has to be defined to understand how predictive models can contribute to such goal.

Let the i^{th} agent's problem be expressed as:

$$\begin{aligned} \arg\max_{\hat{p}_i} u_i(\hat{p}_i, s_{ij}) = & \sum_{j=1}^{n} \beta_j u_i(s_{ij}) - |p_i - \hat{p}_i| \\ \text{s.t. } \hat{p}_i = & f(p_i|\mathbf{x}_i; \mathbf{w}_i) \end{aligned} \tag{1}$$

where $\beta_j \in (0,1)$ is the importance given by the agent to $u_i(s_{i,j})$, a function dependent on unobservable subjective factors $s_{i,j}$, p_i is the actual listing price and $\hat{p}_i$ is the predicted price. $f(p_i|\mathbf{x}_i; \mathbf{w}_i)$ is a target function mapping data $\mathbf{x}_i$ and parameters $\mathbf{w}_i$ to p_i. Note that if either data or prior beliefs were available for $u_i(s_{ij})$, it could be incorporated into AI algorithms and thus estimated.

Since the first term of the utility maximization problem cannot be tackled by AI, predictive models optimize agents' utility by providing accurate predictions in the sense that $|p_i - \hat{p}_i|$ is minimized.

In economic terms, an Airbnb guests benefit from identifying fair priced listings, while real estate firms and Airbnb hosts obtain price margins for their marketing strategy.

4.2 Listings Data

The datasets used in this project were created by Inside Airbnb, a mission driven project that seeks to debate the role of renting residential homes to tourists with data. Inside Aibnb collects quarterly information about Airbnb listings across different countries via web scrapping and makes it available for the public.

Concretely, the city of Los Angeles was chosen to be analyzed during June 2022. A total of 42,041 houses are included with 74 different characteristics. After some data cleaning and feature engineering steps to be expounded in the next section 41,853 listings and 75 predictors were used for modeling. Although the reader may check the appendix for further details, the variables of the listings dataset can be generally classified into four groups: geographical data, personal information about hosts (host name, host profile picture, total active time in Airbnb, etc), residential houses characteristics (number of bathrooms, amenities, availability, etc) and rating metrics of listings (cleanliness, communication, etc).

Additionally, a specific dataset of Airbnb guests' reviews was incorporated into the analysis, amounting to 1,511,891 opinions about 32,413 distinct listings. Only comments written in the English language were considered, so the final dataset ended up with 1,460,047 records. The dimensions of the listings and reviews datasets considerably vary, making it imperative to perform Data Mining before combining both into a joint database. For such purpose, different data preprocessing techniques had to be applied to the reviews dataset in order to turn originally unstructured data into an acceptable format for predictive modeling.

4.3 Feature Engineering

Feature Engineering is the "*process of using domain knowledge to select and transform the most relevant variables from raw data*" (Ng, 2019). For instance, the target variable price was transformed by taking logarithms to address skewed data. Also, since some information originally available in the datasets was of no predictive use, several predictors were ruled out. For an explanation of the final data matrix please refer to the appendix.

4.3.1 Data Imputation Algorithms

One of the drawbacks of web scrapping is that it may not be possible to retrieve all the data about specific individuals. Additionally, sometimes hosts simply do not post complete information about listings.

Although generally perceived as an analytical obstacle, missing data can be indicative of certain patterns. Indeed, creating features by one-hot encoding the presence of null values has proven to be highly effective in Kaggle competitions (Pandey, 2020).

Once missing data is accounted for with one-hot encoding, data imputation comes into play. Imputing missing values is much preferable over eliminating observations that lack some variables since a considerable information loss would be caused. Nonetheless, it is sometimes justified to directly delete some data rows if their number is negligible. Actually, in this project inactive users were directly discarded.

A custom k-nearest neighbor algorithm (k-NN) and MICE algorithm (Multiple Imputation by Chained Equations) were applied to impute data.

For the custom k-NN algorithm, three statistical distance matrices were computed on non-missing data: Mahalanobis distance for quantitative predictors, Jaccard distance for binary variables and Hamming distance for the rest of categorical features. Mahalanobis distance was preferred over commonly used Euclidean distance since the former accounts for correlation between covariates and remains unchanged after scale changes. As for Jaccard and Hamming distance matrices, they were suitable metrics for qualitative data. Based on a combination of these distances (called Gower distance matrix), the six nearest neighbors were identified for each listing in the dataset. Leveraging such information, missing data was imputed by taking the mean value of the neighbors (and then adjusted for binary and categorical variables).

As for MICE algorithm, it was implemented as described by van Buuren and Groothuis-Oudshoorn (2011). MICE combines predicted values for missing data from m copies of the original incomplete dataset into a pooled estimate. Prediction is flexible since different estimators are allowed (Bayesian Ridge Regression and Random Forest were chosen). Algorithm X in the appendix summarizes how MICE works.

4.3.2 Natural Language Processing

Natural Language Processing (NLP) techniques were used to extract information from the unstructured reviews dataset. Emojis, symbols, punctuation and internet links were removed to clean the data. Then text was forced to lowercase and non-English comments were eliminated as to maximize homogeneity among users' comments. Most importantly, the opinion expressed by Airbnb guests was extracted by perfoming Sentiment Analysis.

Unfortunately, no score rating metric was available so training a NLP model was unfeasible since there were no labels. However, by defining regular expressions (regex) and tweaking VADER (Valence Aware Dictionary and sEntiment Reasoner), a lexicon and rule-based sentiment analysis tool attuned to social media created by developers Hutto and Gilbert (2014), sentiment was estimated in an unsupervised manner.

Gender was guessed based on a database containing 40,000 first names collected by Michael (2006), since hosts' names were available.

4.3.3 Computer Vision

The original dataset provided personal information about hosts, among which an internet link to their profile picture was available. For those hosts that did upload a photograph, a Computer Vision (CV) model was passed to predict their gender.

Resorting to transfer learning, a pre-trained model built by Serengil (2020) was used to guess hosts' gender. Firms and couples were assigned a third category within the gender feature. Sample images displaying the CV model performance are shown:

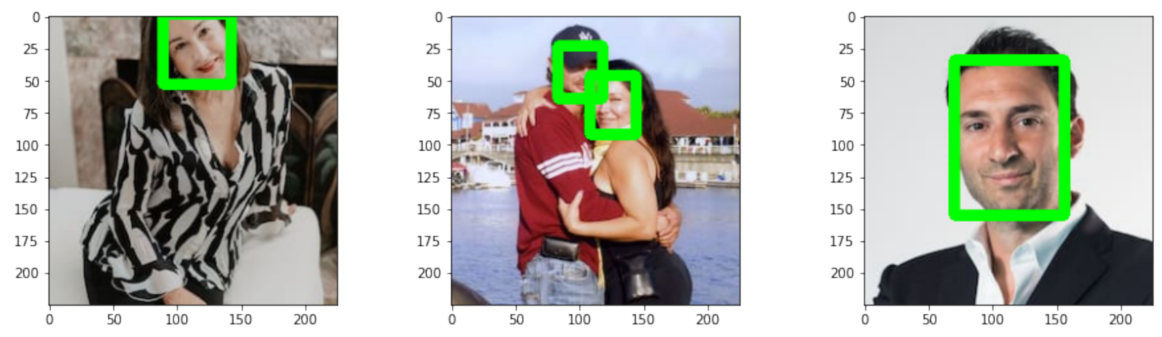


Figure 1: Face Recognition with CV

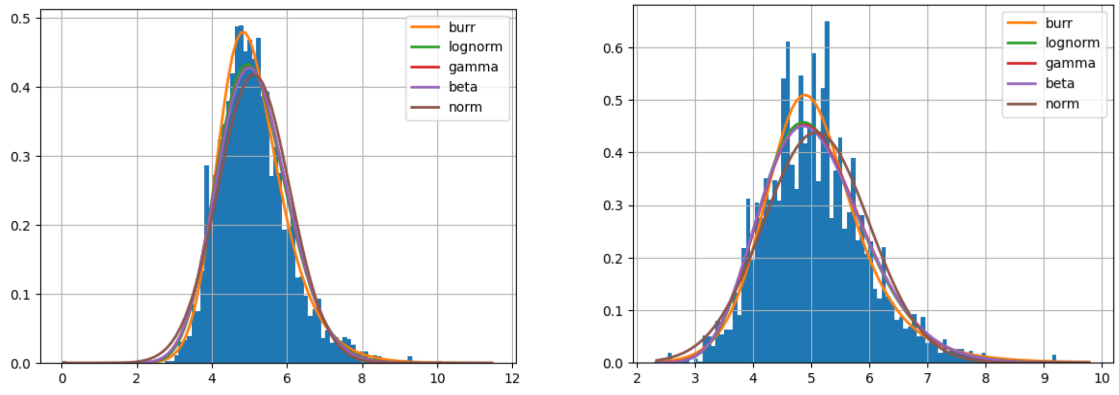
4.4 Tackling online myths: Statistical Inference

According to Davidson and Gleim (2022) the wage gap persists even in the sharing economy. Statistical Inference can shed light on the truth of such statement by verifying whether the median price set by women is truly lower than the median price of men.

Let y_m denote the transformed median price, formally it is sought to test:

$$H_0 : y_m^{♀} = y_m^{♂}$$
$$H_1 : y_m^{♀} < y_m^{♂}$$

A Bayesian hypothesis testing approach is used as suggested by Gelman et al. (1995). Probability distributions for describing the likelihood function and prior beliefs of y^i for $i \in \{♀, ♂\}$ are needed. Five continuous density functions were fit to prices.



(a) Female prices (b) Male prices

Figure 2: Probability distributions

According to goodness of fit metrics such as sum of squared error, Akaike's Information Criteria (AIC) and Bayesian Information Criteria (BIC), the Burr Type III distribution was the preferred density function (details provided in the appendix).

Now a prior belief has to be defined for a parameter of the Burr distribution. Let $y^i|\theta \sim \text{Burr}(\omega, \theta)$, then a conjugate prior is $\theta \sim \text{Gamma}(\alpha, \beta)$.

It is shown in the appendix that the marginal distribution of y^i after obtaining the posterior distribution $p(\theta|y_1^i, \ldots, y_n^i)$ takes the form $y^i \sim \text{Pareto}\left(\alpha + n, \beta + \sum_{i=1}^{n} \ln(1 + y_i^{-\omega})\right)$.

Thus, we can obtain the probability that the median female price is less than the median male price by sampling from the predictive distribution for each gender.

After 300 Monte Carlo simulations, it was noted that results were sensitive to a tolerance threshold $|y_m^{♀} - y_m^{♂}| \geq \varepsilon$. For $\varepsilon = 0$, the probability of the female median price being smaller than the male median price was 1. Nevertheless, differences were not significantly large and so if ε was set to the median difference value between gender prices across all Monte Carlo simulations it was obtained that $\Pr(H_1|y) = 0.5$. Furthermore, the remaining 0.5 probability is evenly distributed among $\Pr(H_0|y)$ and $\Pr(y_m^{♀} > y_m^{♂})$, so there is no evidence of a 'wage' gap in Airbnb. A frequentist sign test revealed the same outcome.

5 Methodology

5.1 Machine Learning notation

Despite mutually sharing various concepts, ML has adopted new terminology for well-established labels in Econometrics.

- Regression in ML is only applied in cases where the '*target*' (dependent variable or regressand) is continuous. For discrete targets '*classification*' is performed (even though logistic regression might be used for such tasks).
- The sample data to estimate parameters is renamed to '*training set*'.
- Regressors, predictors or covariates become '*features*' and parameters are often referred to as '*weights*'.

For the sake of consistency, ML nomenclature will be used, although when necessary the econometric analogues will be mentioned.

Thus, the listings dataset $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^{n}$ is composed of feature vectors $\mathbf{x}_i \in \Re^p$ and a target $\mathbf{y}_i \in \Re$. Each $\mathbf{x}_i$ contains data for the i^{th} listing, concretely p variables or features. If stacked together, feature vectors lead to feature matrix $\mathbf{X} \in \Re^{n \times p}$.

Since there exists a target vector $\mathbf{y} = \{y_i\}_{i=1}^n$ with $y_i \in \Re \; \forall i$, predicting Airbnb prices is a *supervised learning* task (specifically a *regression* problem). Such problem will be solved by estimating the target function $f(\mathbf{y}|\mathbf{X};\mathbf{w})$, which maps feature vectors $\mathbf{x}_i$ to target vector $\mathbf{y}$ through weights $\mathbf{w} \in \Re^p$ (non-parametric models are not considered).

Dataset $\mathcal{D}$ will be randomly partitioned into three disjoint sets: training set $\mathcal{T}$, validation set $\mathcal{V}$ and test set $\mathcal{F}$ so that $\mathcal{T} \cup \mathcal{V} \cup \mathcal{F} = \mathcal{D}$. Specifically, these sets amount to 77%, 14%, 9% of total data (training, validation and test, respectively).

Algorithms learn from the training set $\mathcal{T}$ optimizing a cost function (sum of individual loss functions) $J(\mathbf{w}, \boldsymbol{\lambda}) = \frac{1}{n}\sum_i L_i(\mathbf{w}, \boldsymbol{\lambda})$ and $\hat{w_i}$ are obtained by performing cross-validation. This entails that hyperparameters $\boldsymbol{\lambda}$ (user-specified values to control the learning process) are tuned on the validation set and then assessed in $\mathcal{F}$ so as to obtain an unbiased estimate of model performance.

Root Mean Squared Error (RMSE) is the chosen model performance metric to select the 'best or champion' model. Chai (2014) lists some desirable properties such as sharing in the same measurement units as the target and satisfying the triangle inequality requirement for a distance metric.

Figure X in the appendix provides a graphical illustration of the AI framework.

5.2 Hyperparameter selection procedure

After having learned from the training set and been refined with the validation set, the AI algorithm can be assessed in the test set and study its performance.

During the refining process, not only weights are optimized to attain accurate results but hyperparameters $\boldsymbol{\lambda}$ are also included in the cost function. These hyperparameters serve the purpose of regularizing AI algorithms, acting as "*penalty terms in the loss function*" (Bishop, 1995). Shrinking $\mathbf{w}$ during the learning process through $\boldsymbol{\lambda}$ leads to better generalization on test data. Otherwise AI algorithms tend to *overfit* the training data, procuring noticeable worse results when exposed to non-training instances.

Finding optimal hyperparameters is no trivial task since $\boldsymbol{\lambda}$ is manually selected by the AI practitioner. Some popular methods are Grid Search and Random Search, where a vector of user-defined hyperparameter values are tried during training. Although for some tasks such approaches are sufficient, they are highly inefficient. Bergstra et al. (2011) developed a series of greedy algorithms that leverage past evaluations of hyperparameter values to select the most promising $\boldsymbol{\lambda}$ to evaluate in $J(\mathbf{w}, \boldsymbol{\lambda})$. For that purpose, Bayesian Inference is performed on a probabilistic model (surrogate model).

Two probability distributions are defined based on the following rule: values of $\boldsymbol{\lambda}$ that increase model accuracy are part of $l(\boldsymbol{\lambda})$, whereas poor perfoming values of hyperparameters belong to $g(\boldsymbol{\lambda})$. Performance is measured in terms of a threshold y^*. Tree Parzen Estimator (TPE) is a greedy algorithm that maximizes the expected improvement in model performance based on hyperparameter values drawn from these probability distributions. Clearly, the ratio $\frac{g(\boldsymbol{\lambda})}{l(\boldsymbol{\lambda})}$ is desired to be minimized, since optimal hyperparameter values $\boldsymbol{\lambda}^*$ should be drawn from probability distribution $l(\boldsymbol{\lambda})$ rather than $g(\boldsymbol{\lambda})$. Prior beliefs $p(\boldsymbol{\lambda})$ are updated at each evaluation, yielding posterior distribution $p(\boldsymbol{\lambda}^*|\mathbf{y})$. The following pseudo-code gives additional details on how TPE works to find the best hyperparameters.

Algorithm 1: Tree Parzen Estimator

```
Input: Prior probability distribution $p(\boldsymbol{\lambda})$
Output: Optimal hyperparameters $\boldsymbol{\lambda}^*$
1 Set probabilistic domain of hyperparameters: $\boldsymbol{\lambda} \sim p(\boldsymbol{\lambda})$
2 Define $J(\mathbf{w}, \boldsymbol{\lambda}) = \sum_{i=1}^{n} L_i(\mathbf{w}_i, \boldsymbol{\lambda}_i)$
3 Construct probability distributions $p(\boldsymbol{\lambda}|\mathbf{y}) = \begin{cases} l(\boldsymbol{\lambda}) & \text{if } y < y^* \\ g(\boldsymbol{\lambda}) & \text{if } y \geq y^* \end{cases}$
4 Build surrogate model $p(\mathbf{y}|\boldsymbol{\lambda}) = \frac{p(\boldsymbol{\lambda}|\mathbf{y})p(\mathbf{y})}{p(\boldsymbol{\lambda})}$
5 for $t = 1, \ldots, n$ do
      /* Update $\boldsymbol{\lambda}^{(t)}$ */
      $\boldsymbol{\lambda}^{(t+1)} \leftarrow \arg\max_{y^*} \int_{-\infty}^{y^*} (y^* - y) \frac{p(\boldsymbol{\lambda}^{(t)}|\mathbf{y})p(\mathbf{y})}{p(\boldsymbol{\lambda}^{(t)})} \partial y$
6 Obtain optimal weights $\hat{\mathbf{w}} \leftarrow \arg\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} L_i(\mathbf{w}, \boldsymbol{\lambda}^*)$
```

A graphical representation is available in the appendix in Figure X.

6 Machine Learning Modeling

6.1 Bayesian Ridge Regression

An empirical Bayesian learning framework is proposed as the baseline model. Given prior knowledge on weights $\mathbf{w}$, Bayes' Theorem will be used to update such distribution of beliefs over the model space. Bayesian modeling offers some advantages (Bishop, 2006):

- **Measurability of prediction uncertainty:** Bayesian Inference procures a probability distribution for weights, allowing to quantify uncertainty around model outputs by estimating conditional distribution $p(\mathbf{y}|\mathbf{x})$.
- **Good generalization performance:** Facilitated by imposing a complexity penalty term in the form of a conditional prior probability distribution $p(\mathbf{w}|\boldsymbol{\lambda})$.
- **Sparsity of inferred predictors:** Many of the weights exhibit posterior distributions $p(\mathbf{w}, \boldsymbol{\lambda}, \alpha|\mathbf{y})$ that are sharply peaked around zero, so the learned algorithm makes predictions based on the most relevant features. Thus, feature selection is carried out in a similar fashion to Least Absolute Shrinkage and Selection Operator (LASSO) Regression.

The Bayesian Ridge Regression Model to be estimated is expressed as:

$$\mathbf{y} = f(\mathbf{y}|\mathbf{X};\mathbf{w}) + \boldsymbol{\varepsilon} \tag{2}$$

For which a set of restrictions on the joint distribution of $\{\mathbf{x}_i, y_i\}$ is imposed:

- Stochastic process $\{\mathbf{x}_i, y_i\}$ originates from random sampling.
- The parameterized conditional function is linear $f(\mathbf{y}|\mathbf{X};\mathbf{w}) = \mathbf{X}\mathbf{w}$
- The prior distribution of target vector $\mathbf{y}$ is $p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \boldsymbol{\alpha}) = \mathcal{N}(\mathbf{X}\mathbf{w}, \sigma^2\mathbf{I}_p)$
- The prior distribution of coefficient vector $\mathbf{w}$ is $p(\mathbf{w}|\boldsymbol{\lambda}) = \mathcal{N}(\mathbf{0}, \boldsymbol{\lambda}^{-1}\mathbf{I}_p)$
- ε_i are samples drawn from a noise process such that $\boldsymbol{\varepsilon} \overset{\text{i.i.d}}{\sim} \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}_p)$

Even if an economic agent has no real estate market expertise knowledge, Bayesian Inference allows the parameters of a prior distribution to be estimated from data by building hierarchical models. In order to do that, marginal probability functions have to be defined for hyperpriors $\boldsymbol{\lambda}$ and σ^2, introducing an additional level of beliefs. Specifically:

$$p(\boldsymbol{\lambda}) = \prod_{i=0}^{n} \frac{\beta^\alpha}{\Gamma(\alpha)} \lambda_i^{\alpha-1} e^{-\lambda_i/\beta}, \quad p(\sigma^2) = \frac{\theta^\delta}{\Gamma(\delta)} \sigma^{2(\delta-1)} e^{-\sigma^2 \theta} \tag{3}$$

Note that hyperpriors follow a Gamma distribution, a selection justified by the fact that when operating with the previously assumed distributions $p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \boldsymbol{\alpha})$ and $p(\mathbf{w}|\boldsymbol{\lambda})$, an analytical result can be derived thanks to conjugate priors. Consequently, there is no need to resort to Markov Chain Monte Carlo (MCMC) algorithms for sampling posterior distributions. As an example, the first three weight posterior distribution plots for the intercept, `description` and `host_since` are presented:

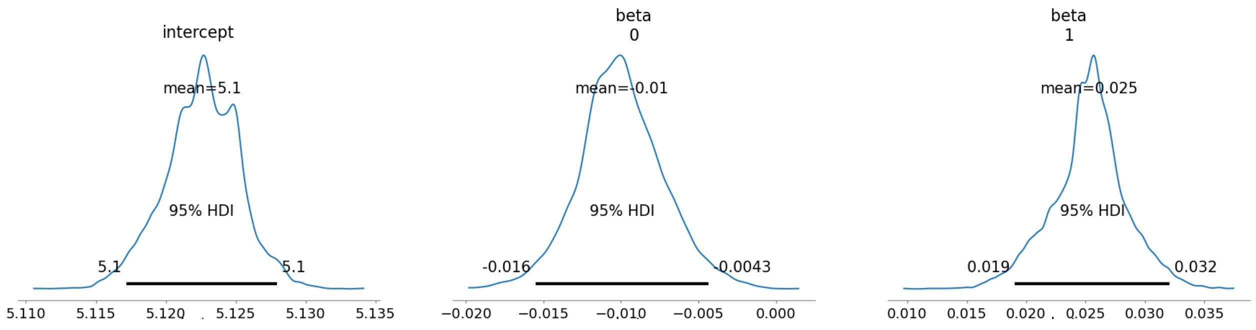


Figure 3: Posterior distribution of $\mathbf{w}$. 95% credible intervals are represented with a solid line

Note how a credible interval is given for each w_i, allowing to analyze predictive uncertainty rather than obtaining unequivocal point estimates. Furthermore, the interpretation of credible intervals is far clearer than confidence intervals. The zero value is not contained within any of the credible intervals above, so the null hypothesis of no individual statistical significance is rejected with probability 95% for these weights.

It is possible to test whether all weights really contribute to enhance predictive performance. A second specification was estimated using only the most relevant features according to the champion model (to be uncovered in the following subsections). Consequently, model comparison is carried out between a complete model (using all 75 features) and a restricted one (top 20 relevant features). For such purpose, Deviance Information Criterion (DIC) is used. Let DIC be expressed as $D(\theta) = -2\ln(p(y|\theta)) + C$, where y is the data, θ represents unkwown parameters, $C \in \mathfrak{R}$ and $p(y|\theta)$ is the likelihood function. Just like AIC and BIC, a lower DIC is associated with higher predictive performance. After estimating the expected log pointwise predictive density using Pareto-smoothed importance sampling leave-one-out cross-validation, it was found that the full model is the preferred specification (50,195.4 vs 54,910.2 DIC). Evidence was found in favor of the statistical significance of features different from the most relevant ones.

6.2 Elastic Net Regression

For comparison purposes, a Linear Regression Model is estimated in a frequentist fashion. The weight vector $\mathbf{w}$ is obtained by solving the following optimization problem:

$$\hat{\mathbf{w}} = \arg\min_{\tilde{\mathbf{w}}} \; (\mathbf{y} - \mathbf{X}\tilde{\mathbf{w}})'(\mathbf{y} - \mathbf{X}\tilde{\mathbf{w}}) + \lambda_1 \|\tilde{\mathbf{w}}\|_1 + \lambda_2 \|\tilde{\mathbf{w}}\|_2^2 \quad , \tag{4}$$

where $\tilde{\mathbf{w}}$ is a running parameter, $\|\mathbf{w}\|_1$ is the ℓ_1-norm of the weight vector and λ_1 is the associated regularization term. Likewise, $\|\mathbf{w}\|_2^2$ is the ℓ_2-norm of $\mathbf{w}$ and λ_2 is its regularization term.

Zou and Hastie (2005) combined ℓ_1 and ℓ_2-norm regularization to overcome some drawbacks of LASSO, obtaining a new regularization and feature selection method. The result is that few non-zero weights are left while maintaining properties of Ridge Regression. LASSO tends to vanish many $\mathbf{w}_i$, sometimes resulting in poorer performance due to information loss.

Note that some restrictions on the joint distribution of $\{\mathbf{x}_i, y_i\}$ can be relaxed for this model: heteroskedasticity is allowed, there is no need to define prior distributions and random sampling can be reduced to ergodic stationarity.

Despite being a notably less restrictive model in terms of assumptions, Elastic Net Regression performs worse than Bayesian Ridge Regression (0.5232 vs 0.5104 RMSE on test data, respectively). Thus, as of now the best perfoming model is Bayesian Regression.

6.3 Random Forest

Random Forest algorithm is an example of an "*Ensemble Method*", which combines multiple predictors. Ho (1998) argued that by aggregating weak learners predictive performance could significantly improve. It is imperative that such learners are not identical. Since Random Forest only consists of Decision Trees, diversity stems from training said algorithm on random subsets of data selected by "*bootstrap aggregating*" or "*bagging*" with a multinomial probability distribution. Since Decision Trees are extremely sensitive to the data they are exposed to, training them with dissimilar datasets leads to different predictions $\hat{\mathbf{y}}_i$. Finally, the output of Random Forest is the mean of prediction vector $\hat{\mathbf{y}}_i$.

Decision Trees are trained with the CART algorithm. Firstly, data is split into two subsets using the j^{th} feature and establishing a threshold t_j (i.e, # amenities ≤ 3). The pair (j, t_j) is selected by minimizing a cost function $J(j, t_j)$. CART recursively performs this operation until $J(j, t_j)$ cannot be reduced anymore or the maximum depth of Decision Trees has been reached (a hyperparameter). Random Forest further increases stochasticity by considering only a random subset of features, trading a higher bias for a lower variance.

Taddy et al. (2015) derived ensembles of Decision Trees through a non-parametric Bayesian model, allowing Random Forest to be interpreted as "*a sample from a posterior* [distribution] *over trees*". Leveraging that the Dirichlet distribution is a conjugate prior of the multinomial distribution, the bootstrap posterior distribution is also Dirichlet. In practice, an exponential distribution is used to define the prior since the `ensemble` module of `scikit-learn` normalizes the draws from the posterior distribution. This Bayesian adaption of Random Forest has been implemented along with the Frequentist version.

Unlike classical methods, Random Forest and the rest of ML algorithms require little to no restrictions on $\{\mathbf{x}_i, y_i\}$. In fact, there is no need to impose linearity or any probability distribution on the data (only the prior distribution for Bayesian Random Forest), predictors can far exceed the number of observations and redudancy is not a problem as feature selection is carried out by CART algorithm. Random Forest is also insensitive to data scaling due to the threshold t_j and random sampling is not necessary to hold as long as bootstrap procures sufficiently representative samples of data (Géron, 2019).

It is possible to make Random Forest even more stochastic by selecting random thresholds t_j rather than the ones that minimize $J(j, t_j)$. Geurts etl al. (2006) coined this algorithm as Extremely Randomized Trees and found that time complexity is lowered as well as variance.

The following table displays RMSE on test data of the different RF algorithms.

Table 1: Random Forest algorithm results

Frequentist RF	Bayesian RF	Frequentist ERF	Bayesian ERF
0.4551	0.4205	0.4507	0.4572

Notice how all Random Forest algorithms outperform Linear Regression. Bayesian Random Forest is the preferred model as it achieves the smallest RMSE.

6.4 Neural Networks

Aritifical Neural Networks (ANN) are Deep Learning algorithms that mimic biological neural circuits by transmitting information through neurons from different layers. Below, a graphical representation of the ANN architecture used in this project is shown.

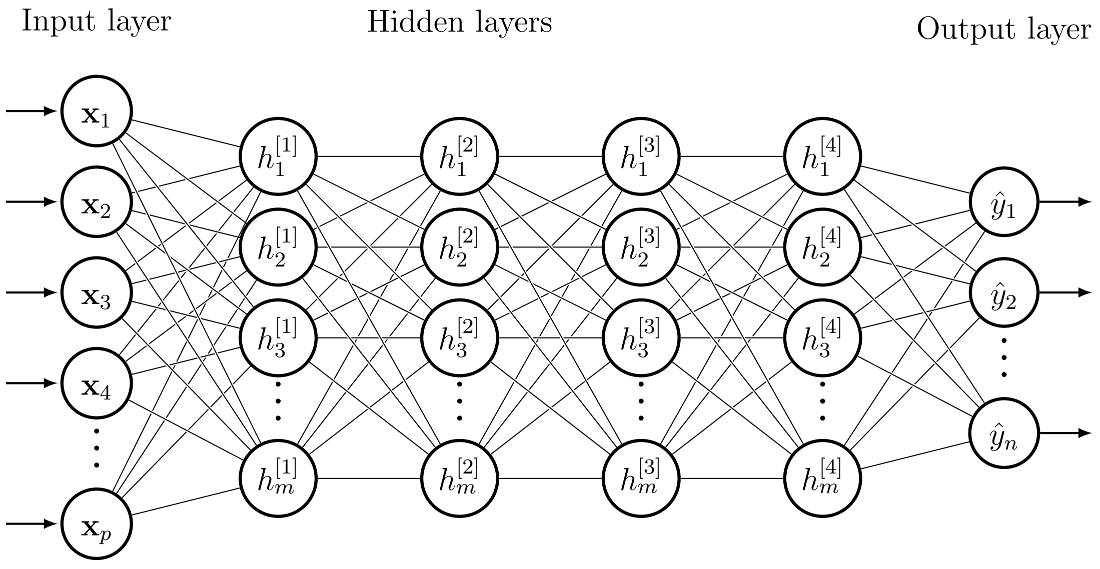


Figure 4: Artificial Neural Network Architecture

Note that the ANN architecture is composed of three layers: input layer, hidden layers and an output layer (predictions). ANN learn from data stemming from the feature vectors $\mathbf{x}_i$ in the input layer, process such information in the j^{th} neuron located in the i^{th} hidden layer, and finally procure predictions $\hat{\mathbf{y}}$ in the output layer.

Neurons transmit information by forwarding data from the input layer to the output layer (*feed forward*). Then, $J(\mathbf{w}, \boldsymbol{\lambda})$ is computed and optimized by propagating it to the earlier layers (*back propagation*). This process is repeated until convergence is attained by either not reducing prediction error anymore or defining a maximum number of iterations.

Typically, *batches* (a fraction of the training data) are fed to the ANN for some cycles called *epochs* (number of times training data has been fully updated).

Unlike Decision Trees and Random Forests, ANN need to find minima of $J(\mathbf{w}, \boldsymbol{\lambda})$ using Gradient Descent algorithm. For the sake of avoiding vanishing gradients problem, data is standardized using the mean and standard deviation of the training set. Additionally, neurons are randomly initialized following He (2015) and a nonsaturating link function or activation function is applied to each hidden layer's ouput. Exponential Linear Unit (ELU) has been chosen as the preferred activation function as it forces negative outputs to lie close to zero, while not altering positive ones. This overcomes issues with Rectified Linear Unit function (ReLU), which was not differentiable for negative values. The intuition behind ELU is that it makes backpropagation easier and faster by having an average output closer to 0, which alleviates vanishing gradients problem (Clevert et al., 2015).

The optimization algorithm used to train the ANN was Nesterov-accelerated Adaptive Moment Estimation (Nadam), developed by Dozat (2015). Nadam includes an adaptive learning rate to classical Adam algorithm called Root Mean Squared Propagation (RMSProp) which increases performance and convergence speed. Additionally, Elastic Net Regularization was applied as to penalize overconfident predictions. Algorithm X in the appendix provides further technical details on the training process of the ANN.

The architectural complexity of ANNs makes them ideal to solve intricate predictive problems. However, when trained in medium-sized (less than 100,000 observations) tabular datasets, Grinsztajn et al. (2022) showed that tree-based models like Random Forests and XGBoost tend to outperform ANNs. Even when ANNs procure more accurate predictions, training takes longer and the number of parameters is astronomical compared to the aforementioned algorithms. Indeed, the ANN trained in this model is composed of 1,591,532 parameters, a total overkill considering it does not even beat Bayesian RF.

Drawbacks like training time and algorithm complexity might deter AI practitioners from adapting ANNs to Bayesian Statistics, since the latter is associated with demanding compute resources. Nevertheless, the possibility of quantifying uncertainty, better predictive performance and above all, Variational Inference (VI), have made it possible to create Bayesian Neural Networks (BNNs).

A BNN is a stochastic ANN trained using Bayesian Inference. BNNs can be built following a similar logic to the Frequentist one by defining prior distributions on the weights $p(\mathbf{w})$, updating them according to evidence $p(\mathbf{y}|\mathbf{w}, \mathbf{X})$ so as to quantify uncertainty around predictions $p(\hat{\mathbf{y}}|\mathbf{X}, \mathbf{w})$ (posterior predictive distribution). In VI, a variational distribution parametrized by learnable weights $q_\phi(H)$ is approximated to the true posterior distribution $p(H|\mathbf{y}, \mathbf{X})$ with a modified Stochastic Gradient Descent algorithm (Blei et al., 2016).

As BNNs introduce penalization with prior distributions, only two hidden layers were needed with no Elastic Net Regularization.

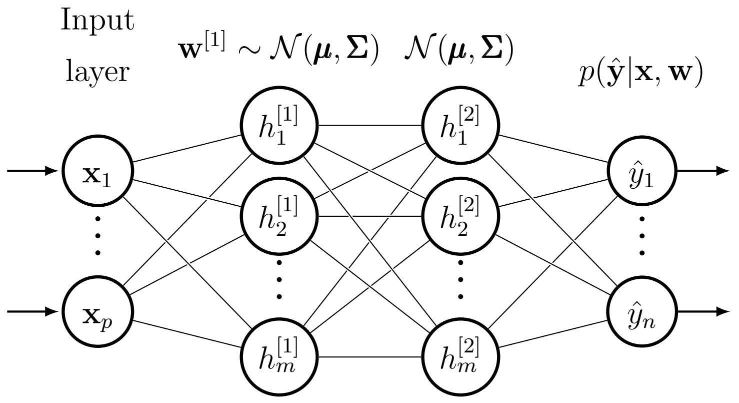

Input layer
$\mathbf{w}^{[1]} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
$p(\hat{\mathbf{y}}|\mathbf{x}, \mathbf{w})$
$\mathbf{x}_1$
$\mathbf{x}_p$
$h_1^{[1]}$
$h_2^{[1]}$
$h_m^{[1]}$
$h_1^{[2]}$
$h_2^{[2]}$
$h_m^{[2]}$
$\hat{y}_1$
$\hat{y}_n$

Figure 5: Bayesian Neural Network Architecture

The Frequentist ANN performed worse than the BNN on test data (0.4305 vs 0.4024 RMSE). As of now, the preferred model is the BNN in terms of predictive accuracy.

6.5 XGBoost

7 Conclusion

Angrist, J. (2020). Rajk College for Advanced Studies Interview. Retrieved 13/11/202.

Pernagallo, G. and Torrisi, B. (2019). “A theory of information overload applied to perfectly efficient financial markets”. *Review of Behavioral Finance.*

Rudin, C. (2019). “Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead”. *Nature Machine Intelligence*, Vol 1, May.

Grewal, P. (2014). “A Critical Conceptual Analysis of Definitions of Artificial Intelligence as Applicable to Computer Engineering”. *Journal of Computer Engineering*, 16, 9-13.

Mitchell, T. (1997). “*Machine Learning*”. McGraw Hill.

Brownlee, J. (2020). “Difference Between Algorithm and Model in Machine Learning”. Retrieved from *Machine Learning Machinery* (04/11/2022).

Angrist, J. and Frandsen B. (2019). “Machine Labor”. *National Bureau of Economic Research.*

Dube, A. Cengiz, D., Lindner, A., Zentler-Munro, D. (2019). “Seeing Beyond the Trees: Using Machine Learning to estimate the impact of Minimum Wages on Labor Market Outcomes”. *National Bureau of Economic Research.*

Hansen, S., McMahon, M., Prat, A. (2019). “Transparency and Deliberation Within the FOMC: A Computational Linguistics Approach”. *The Quarterly Journal of Economics*, Volume 133, Issue 2, May.

Limsombunchai, V., McMahon, M., Prat, A. (2019). “House Price Prediction: Hedonic Price Model vs. Artificial Neural Network”. *American Journal of Applied Sciences.*

Embaye, W., Zereyesus, Y., Chen, B. (2021). “Predicting the rental value of houses in household surveys in Tanzania, Uganda and Malawi: Evaluations of hedonic pricing and machine learning approaches”. *Public Library of Science.*

Kayode, E. and Tang, B. (2021). “The Role of Econometrics Data Analysis Method in the Social Sciences (Education) Research”. *Journal of Education and Practice.*

Inside Airbnb. Data retrieved from Inside Airbnb website.

Ng, A. (2019). “Machine Learning and AI via Brain Simulations”. *Stanford University.*

Pandey, P. (2020). "*A Guide to Handling Missing values in Python*". Kaggle Notebook.

van Buuren, S. and Oudshoorn, K.G (2011). "mice: Multivariate Imputation by Chained Equations in R". *Journal of Statistical Software.*

Hutto, C. and Gilbert E. (2014). "VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text". *Eighth International AAAI Conference on Weblogs and Social Media.*

Michael, J. (2006). "Gender-verification by forename". *Authokey forum.*

Serengil, S. (2020). "LightFace: A Hybrid Deep Face Recognition Framework". *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)* serengil

Davidson, A. and Gleim, M. (2020). "Female Airbnb hosts earn thousands less per year than male hosts ". *The Conversation.* Retrieved 10/10/2022 serengil

Gelman, A., Carlin, J., Stern, H., Dunson, D., Vehtari, A., Rubin, D. (1995). *"Bayesian Data Analysis"*. Chapman and Hall/CRC.

Chai, T. (2014). "Root mean square error (RMSE) or mean absolute error (MAE)?– Arguments against avoiding RMSE in the literature". *Geoscientific Model Development.*

Bishop, C. (1995). *"Neural Networks for Pattern Recognition"*. Oxford University Press.

Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B. (2011). "Algorithms for Hyper-Parameter Optimization". *Advances in Neural Information Processing Systems.*

Bishop, C. (2006). *"Pattern Recognition and Machine Learning"*. Springer.

Zou, H. and Hastie, T. (2005). "Regularization and Variable Selection via the Elastic Net". *Journal of the Royal Statistical Society.*

Ho, T. (1998). "The Random Subspace Method for Constructing Decision Forests". *IEEE Transactions on Pattern Analysis and Machine Intelligence.*

Taddy, M., Chen, C., Yu, J., Wyle, M. (2015). "Bayesian and Empirical Bayesian Forests". *International Conference on Machine Learning.*

Géron, A. (2019). *"Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow"*. O'Reilly.

Geurts, P., Ernst, D., Wehenkel, L. (2006). "Extremely Randomized Trees". *Machine Learning*, Springer Verlag.

He, K., Zhang, X., Ren, S., Sun, J. (2015). "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". *Microsoft Research.*

Clevert, D., Unterthiner, T., Hochreiter, S. (2015). "Fast and Accurate Deep Networks Learning by Exponential Linear Units". *International Conference on Learning Representations.*

Grinsztajn, L., Oyallon, E., Varoquaux, G. (2020). "Why do tree-based models still outperform deep learning on tabular data?". *NeurIPS 2022 Track Datasets and Benchmarks Program Chairs.*

Blei, D., Kucukelbir, A., McAuliffe, J. (2020). "Variational Inference: A Review for Statisticians". *Journal of the American Statistical Association, Vol. 112.*

Appendix

Algorithm 2: Iterative Imputer Algorithm

```
  Input: Incomplete dataset $\mathcal{D}$
  Output: Imputed dataset $\mathcal{D}^*$
1 Create $m$ copies of $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_m\}$
2 Iteratively sample $p(\mathbf{y}_i|\mathbf{y}_{-i}, \boldsymbol{\theta}_i)$ from each $\mathcal{D}_j$
3 Draw $(\boldsymbol{\theta}^{(t)}, \mathbf{y}^{(t)})$ with Gibbs sampler:
4 for $t = 1, \ldots, n$ do
      $$\begin{aligned} \boldsymbol{\theta}_1^{(t)} &\sim p(\boldsymbol{\theta}_1|\mathbf{y}_1^{\text{obs}}, \mathbf{y}_2^{(t-1)}, \ldots, \mathbf{y}_p^{(t-1)}) \\ \mathbf{y}_1^{(t)} &\sim p(\mathbf{y}_1|\mathbf{y}_1^{\text{obs}}, \mathbf{y}_2^{(t-1)}, \ldots, \mathbf{y}_p^{(t-1)}, \boldsymbol{\theta}_1^{(t)}) \\ &\vdots \\ \boldsymbol{\theta}_p^{(t)} &\sim p(\boldsymbol{\theta}_p|\mathbf{y}_p^{\text{obs}}, \mathbf{y}_1^{(t)}, \ldots, \mathbf{y}_{p-1}^{(t)}) \\ \mathbf{y}_p^{(t)} &\sim p(\mathbf{y}_p|\mathbf{y}_p^{\text{obs}}, \mathbf{y}_1^{(t)}, \ldots, \mathbf{y}_p^{(t)}, \boldsymbol{\theta}_p^{(t)}) \end{aligned}$$
5 Estimate $Q^{(1)}, \ldots, Q^{(m)}$ and pool results: $\bar{Q} = \text{pool}(\hat{Q}^{(1)}, \ldots, \hat{Q}^{(m)})$
```

Algorithm 3: Random Forest Algorithm

```
1 for $i = 1, \ldots, k$ do
      /* Randomly partition data into $\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(k)}$ */
      $$\mathcal{D}^{(i)} \sim \frac{\Gamma\left(\sum_{i=1}^{n} x_i + 1\right)}{\prod_{i=1}^{n} \Gamma(x_i + 1)} \prod_{i=1}^{n} \theta_i^{x_i}$$
2 Select $(j, t_j)$ in each $\mathcal{D}^{(i)}$ obtain final estimate $\hat{\mathbf{y}}$:
3 for $i = 1, \ldots, k$ do
      $j^{(i)} \sim \mathcal{U}[1, p]$
      $j^{(i)}, t_j^{(i)} \leftarrow \arg \min_{j^{(i)}, t_j^{(i)}} \frac{n_l}{n} \sum_l \left(\mathbf{y}^{(i)} - \hat{\mathbf{y}}_l^{(i)}\right)^2 + \frac{n_r}{n} \sum_r \left(\mathbf{y}^{(i)} - \hat{\mathbf{y}}_r^{(i)}\right)^2$
      $\hat{\mathbf{y}} = \frac{1}{n} \sum_i \hat{\mathbf{y}}^{(i)}$
  Note: Subscripts $l, r$ denote the left-hand and right-hand split, respectively.
```

Algorithm 4: Bayesian Random Forest Algorithm

```
1 for $i = 1, \dots, k$ do
      /* Randomly partition data into $\mathcal{D}^{(1)}, \dots, \mathcal{D}^{(k)}$ */
      $$\mathcal{D}^{(i)} \sim \frac{\Gamma\left(\sum_{i=1}^{n} x_i + \alpha_i\right)}{\prod_{i=1}^{n} \Gamma(x_i + \alpha_i)} \prod_{i=1}^{n} \theta_i^{x_i + \alpha_i - 1}$$
2 Select $(j, t_j)$ in each $\mathcal{D}^{(i)}$ and obtain final estimate $\hat{\mathbf{y}}$:
3 for $i = 1, \dots, k$ do
      $j^{(i)} \sim \mathcal{U}[1, p]$
      $j^{(i)}, t_j^{(i)} \leftarrow \arg \min_{j^{(i)}, t_j^{(i)}} \frac{n_l}{n} \sum_l \left(\mathbf{y}^{(i)} - \hat{\mathbf{y}}_l^{(i)}\right)^2 + \frac{n_r}{n} \sum_r \left(\mathbf{y}^{(i)} - \hat{\mathbf{y}}_r^{(i)}\right)^2$
      $\hat{\mathbf{y}} = \frac{1}{n} \sum_i \hat{\mathbf{y}}^{(i)}$
```

Algorithm 5: Extremely Randomized Forest Algorithm

```
1 for $i = 1, \dots, k$ do
      /* Randomly partition data into $\mathcal{D}^{(1)}, \dots, \mathcal{D}^{(k)}$ */
      $$\mathcal{D}^{(i)} \sim \frac{\Gamma\left(\sum_{i=1}^{n} x_i + 1\right)}{\prod_{i=1}^{n} \Gamma(x_i + 1)} \prod_{i=1}^{n} \theta_i^{x_i}$$
2 Select $(j, t_j)$ in each $\mathcal{D}^{(i)}$ and obtain final estimate $\hat{\mathbf{y}}$:
3 for $i = 1, \dots, k$ do
      $j^{(i)} \sim \mathcal{U}[1, p], \quad t_j \sim \mathcal{U}[\min\{\mathbf{X}_j\}, \max\{\mathbf{X}_j\}]$
      $j^{(i)}, t_j^{(i)} \leftarrow \arg \min_{j^{(i)}, t_j^{(i)}} \frac{n_l}{n} \sum_l \left(\mathbf{y}^{(i)} - \hat{\mathbf{y}}_l^{(i)}\right)^2 + \frac{n_r}{n} \sum_r \left(\mathbf{y}^{(i)} - \hat{\mathbf{y}}_r^{(i)}\right)^2$
      $\hat{\mathbf{y}} = \frac{1}{n} \sum_i \hat{\mathbf{y}}^{(i)}$
```

Algorithm 6: Extremely Randomized Bayesian Forest Algorithm

```
1 for $i = 1, \ldots, k$ do
      /* Randomly partition data into $\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(k)}$ */
      $\mathcal{D}^{(i)} \sim \frac{\Gamma\left(\sum_{i=1}^{n} x_i + \alpha_i\right)}{\prod_{i=1}^{n} \Gamma(x_i + \alpha_i)} \prod_{i=1}^{n} \theta_i^{x_i + \alpha_i - 1}$
2 Select $(j, t_j)$ in each $\mathcal{D}^{(i)}$ and obtain final estimate $\hat{\mathbf{y}}$:
3 for $i = 1, \ldots, k$ do
      $j^{(i)} \sim \mathcal{U}[1, p], \quad t_j \sim \mathcal{U}[\min\{\mathbf{X}_j\}, \max\{\mathbf{X}_j\}]$
      $j^{(i)}, t_j^{(i)} \leftarrow \arg\min_{j^{(i)}, t_j^{(i)}} \frac{n_l}{n} \sum_l \left(\mathbf{y}^{(i)} - \hat{\mathbf{y}}_l^{(i)}\right)^2 + \frac{n_r}{n} \sum_r \left(\mathbf{y}^{(i)} - \hat{\mathbf{y}}_r^{(i)}\right)^2$
      $\hat{\mathbf{y}} = \frac{1}{n} \sum_i \hat{\mathbf{y}}^{(i)}$
```

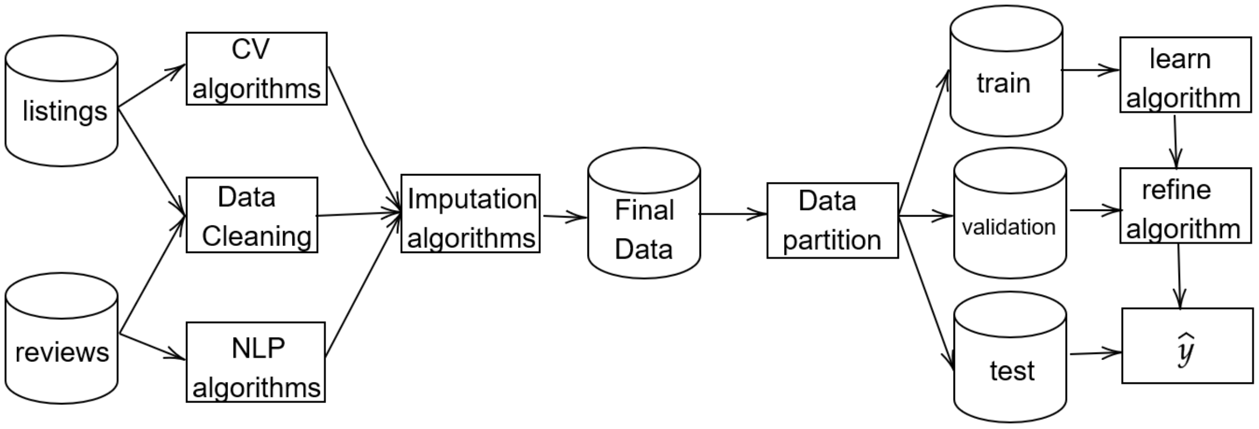


Figure 6: AI Framework

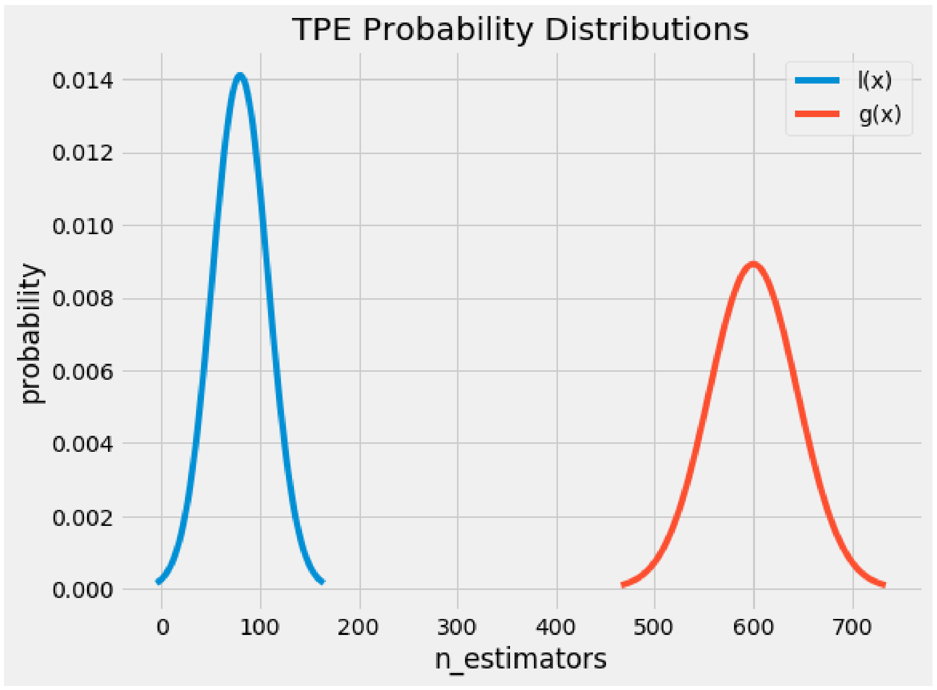


Figure 7: TPE Algorithm

Marginal probability distribution of prices $f(y)$

$$f(y) = \int_0^{+\infty} f(y|\theta) f(\theta)\ \partial\theta$$

$$= \int_0^{+\infty} \frac{\omega\theta y^{-(\omega+1)}}{(1+y^{-\omega})^{\theta+1}} \frac{\beta^\alpha}{\Gamma(\alpha)} \theta^{\alpha-1} e^{-\theta\beta}\ \partial\theta$$

$$= \frac{\omega y^{-(\omega+1)}\beta^\alpha}{\Gamma(\alpha)} \int_0^{+\infty} \theta^\alpha e^{-\theta\beta} \frac{1}{(1+y^{-\omega})^{\theta+1}}\ \partial\theta$$

$$= \frac{\omega y^{-(\omega+1)}\beta^\alpha}{\Gamma(\alpha)} \int_0^{+\infty} \theta^\alpha e^{-\theta\beta} \exp\left\{\ln\left[\frac{1}{(1+y^{-\omega})^{\theta+1}}\right]\right\} \partial\theta$$

$$= \frac{\omega y^{-(\omega+1)}\beta^\alpha}{\Gamma(\alpha)} \int_0^{+\infty} \theta^\alpha e^{-\theta\beta} \exp\left\{-(\theta+1)\ln(1+y^{-\omega})\right\} \partial\theta$$

$$= \frac{\omega y^{-(\omega+1)}}{1+y^{-\omega}} \frac{\beta^\alpha}{\Gamma(\alpha)} \int_0^{+\infty} \theta^\alpha \exp\left\{-\theta[\beta + \ln(1+y^{-\omega})]\right\} \partial\theta$$

$$= \frac{\omega y^{-(\omega+1)}}{1+y^{-\omega}} \frac{\beta^\alpha}{\Gamma(\alpha)} \frac{\Gamma(\alpha^*)}{\beta^{*\alpha^*}} \int \frac{\beta^{*\alpha^*}}{\Gamma(\alpha^*)} \theta^{\alpha^*-1} e^{-\theta\beta^*}\ \partial\theta$$

$$= \frac{\omega y^{-(\omega+1)}}{1+y^{-\omega}} \frac{\beta^\alpha}{\Gamma(\alpha)} \frac{\alpha\Gamma(\alpha)}{[\beta + \ln(1+y^{\omega})]^{\alpha+1}}$$

$$= \frac{\alpha\beta^\alpha}{[\beta + \ln(1+y^{-\omega})]^{\alpha+1}} \left(-\frac{\omega y^{-(\omega+1)}}{1+y^{-\omega}}\right) \qquad \text{Let } z \coloneqq \ln(1+y^{-\omega})$$

$$= \frac{\alpha\beta^\alpha}{(\beta+z)^{\alpha+1}} \left(-\frac{\partial z(y)}{\partial y}\right) \quad \implies y \sim \text{Pareto}(\alpha,\beta)$$

Posterior probability distribution $f(\theta|\{y_i\}_{i=1}^n)$

$$f(\theta|\{y_i\}_{i=1}^n) = \frac{l(y|\theta)f(\theta)}{\displaystyle\int_0^{+\infty} f(y|\theta)f(\theta)\ \partial\theta}$$

$$\propto \prod_{i=1}^n \frac{\omega\theta y_i^{-(\omega+1)}}{(1+y_i^{-c})^{\theta+1}} \frac{\beta^\alpha}{\Gamma(\alpha)}\theta^{\alpha-1}e^{-\theta\beta}$$

$$\propto \theta^{\alpha+n-1}e^{-\theta\beta}\prod_{i=1}^n \exp\left\{\ln\left[\frac{1}{(1+y_i^{-\omega})^{\theta+1}}\right]\right\}$$

$$\propto \theta^{\alpha+n-1}e^{-\theta\beta}\prod_{i=1}^n \exp\left\{-(\theta+1)\ln(1+y_i^{-\omega})\right\}$$

$$\propto \theta^{\alpha+n-1}e^{-\theta\beta}\exp\left\{-\sum_{i=1}^n(\theta+1)\ln(1+y_i^{-\omega})\right\}$$

$$\propto \theta^{\alpha+n-1}e^{-\theta\beta}\exp\left\{-\sum_{i=1}^n\theta\ln(1+y_i^{-\omega})\right\}$$

$$\propto \theta^{\alpha+n-1}\exp\left\{-\theta\left(\beta+\sum_{i=1}^n\ln(1+y_i^{-\omega})\right)\right\}$$

$$= \frac{\left(\beta+\displaystyle\sum_{i=1}^n\ln(1+y_i^{-\omega})\right)^{\alpha+n}}{\Gamma(\alpha+n)}\theta^{\alpha+n-1}\exp\left\{-\theta\left(\beta+\sum_{i=1}^n\ln(1+y_i^{-\omega})\right)\right\}$$

$$= \frac{\beta^{*\alpha^*}}{\Gamma(\alpha^*)}\theta^{\alpha^*-1}e^{-\theta\beta^*} \quad\Longrightarrow\quad f(\theta|\{y_i\}_{i=1}^n) = \text{Gamma}\left(\alpha+n, \beta+\sum_{i=1}^n\ln(1+y_i^{-\omega})\right)$$

Bayesian Random Forest: Posterior probability distribution $f(\theta|\{x_i\}_{i=1}^{p})$

Let $x|\theta \sim \mathrm{Multi}(\theta)$ and $\theta \sim \mathrm{Dir}(\alpha)$, then

$$f(\theta|x_i\}_{i=1}^{p}) = \frac{l(x|\theta)f(\theta)}{\displaystyle\int_0^{+\infty} f(x|\theta)f(\theta)\ \partial\theta}$$

$$\propto \prod_{i=1}^{p} \theta^{x_i} \frac{\Gamma\left(\displaystyle\sum_{i=1}^{p}\alpha_i\right)}{\displaystyle\prod_{i=1}^{p}\Gamma(\alpha_i)} \prod_{i=1}^{p}\theta_i^{\alpha_i-1}$$

$$\propto \prod_{i=1}^{p} \theta_i^{\alpha_i+x_i-1} \frac{\Gamma\left(\displaystyle\sum_{i=1}^{p}\alpha_i\right)}{\displaystyle\prod_{i=1}^{p}\Gamma(\alpha_i)}$$

$$= \frac{\Gamma\left(\displaystyle\sum_{i=1}^{p}\alpha_i + x_i\right)}{\displaystyle\prod_{i=1}^{p}\Gamma(\alpha_i+x_i)} \prod_{i=1}^{p}\theta_i^{\alpha_i+x_i-1}$$

Consequently, $f(\theta|\mathbf{x}) = \mathrm{Dir}(\alpha + \mathbf{x})$