



Treball de Fi de Grau

GRAU D'ENGINYERIA INFORMÀTICA

**Facultat de Matemàtiques
Universitat de Barcelona**

DESENVOLUPAMENT D'UN MODEL PREDICTIU DEL PREU DELS ALLOTJAMENTS TURÍSTICS DE BARCELONA

Pere Antoni Manresa Rigo

Director: Jordi Vitrià

Realitzat a: Departament de Matemàtica
Aplicada i Anàlisi.
UB

Barcelona, 28 de juny de 2016

Taula de Continguts

Abstract	4
1. Introducció i Motivacions	5
2. Metodologia	8
2.1. Obtenció de les dades	8
2.2. Neteja i exploració de les dades	9
2.3. Desenvolupament del model predictiu	9
2.3.1. Introducció als models predictius	9
2.3.2. Sistemes de regressió	12
2.3.2.1. Regressió Lineal	12
2.3.2.2. Ridge Regression	13
2.3.2.3. Lasso Regression	14
2.3.2.4. Elastic Net	15
2.3.2.5. Bayesian Ridge Regression	16
2.3.3. Ensemble Learning	18
2.3.3.1. Adaptive Boosting	18
2.3.3.1.1. Gradient Boosting Regressor	19
2.4. Microneighbourhoods	21
2.5. Extracció característiques de les imatges	22
2.6. Selecció de variables	23
2.7. Aplicació Docker	24
3. Desenvolupament	26
3.1. Obtenció de les dades	26
3.1.1. Web Scraping	26
3.1.1.1. Realitzar peticions web	27
3.1.1.2. Obtenir pàgines web dels allotjaments	27
3.1.1.3. Iteració, obtenir i processar dades	28

3.1.1.4. Actualitzar base de dades	30
3.1.1.5. Inferències del procés	30
3.1.2. Obtenció de les imatges dels allotjaments	31
3.2. Preparació de les dades	32
3.3. Exploració de les dades	35
3.3.1. Què explorar?	35
3.3.2. El model d'ocupació de Airbnb	36
3.3.2.1. Disponibilitat	36
3.3.2.2. Activitat	37
3.3.2.3. Ingrés	38
3.4. Desenvolupament del model predictiu	40
3.4.1. Preparació de les dades	40
3.4.2. Aplicació dels models lineals	41
3.4.3. Optimització dels resultats	44
3.4.3.1. Mètodes ensemble	44
3.4.3.2. Cerca dels millors paràmetres	44
3.4.3.2.1. Apache Spark	45
3.4.3.3. Selecció de variables	45
3.5. Microneighbourhoods	46
3.5.1. Extracció dels polígons	46
3.5.2. Exemple d'execució	49
3.6. Extracció característiques de les imatges	51
3.6.1. Obtenció del vector de característiques	52
3.6.2. Afegir les noves característiques al model predictiu	53
3.7. Aplicació Docker	54
4. Resultats	56
4.1. Exploració de les dades	56
4.1.1. Dades interessants de Barcelona	56
4.1.2. Distribució d'allotjaments per barris	56

4.1.3. El model ocupacional de Airbnb.	58
4.2. Microneighbourhoods.	66
4.3. Model predictiu.	71
4.3.1. Resum dels resultats.	76
5. Discussió	77
5.1. Treball futur.	79
Agraïments	80
Referències	81

Abstract

This project carries out a development of a statistical modelling research which intends to represent the most important features of Airbnb users' usage of its services. This research comprises of a deployment of different statistical learning techniques, from building lineal regression models, and implementing ensemble algorithms such as Gradient Boosting Regressor, to apply state-of-the-art optimization methods. The goal of this project is not to create a deep and flexible statistical model by deploying complex algorithms but leverage a wide range of simpler techniques in order to build a stronger and more comprehensive model. The innovative market of software products within the touristic sector has been one of the main targets of many actual companies. The Airbnb methodology's real battle against many old-fashioned accommodation services has raised the interest of many housing companies which are investing a lot of money in understanding their secrets. The illustration of this project's results will help readers to achieve a better understanding of how Airbnb is actually used by its users, and how they could obtain a compensating revenue out from its services.

CAPÍTOL 1:

INTRODUCCIÓ I MOTIVACIÓ

La ràpida evolució i escalabilitat del turisme en Barcelona, juntament amb l'augment de l'ús del Internet amb el seus serveis, ha estat generant una quantitat enorme de dades brutes encara per tractar. Experts en l'entorn de ciència de les dades han focalitzat molts dels seus projectes en inferir conclusions interessants sobre aquestes dades per després prendre decisions que poden marcar una considerable diferència dins un negoci del sector turístic. La ciència de les dades és un entorn dins de la tecnologia de la computació i anàlisi matemàtic que requereix de tecnologies i mètodes que permeten tractar dades no processades en grans quantitats, entendre patrons i comportaments que segueixen mitjançant la implementació de models estadístics, visualització d'aquestes dades d'una forma intel·ligent i proporcionar una eina en gran part informativa que ajuda a entendre un problema a partir de l'anàltica de dades.

Una part dels projectes de l'entorn de la ciència de les dades desenvolupats sobre Barcelona enfoca el sector turístic: hotels, immobiliàries, lloguer. On estudis dels registres dels clients i del seu comportament en aquest sector poden donar pas a entendre i inferir conclusions que després es poden convertir en decisions claus per l'èxit d'un negoci. Malgrat això, fa uns anys va sorgir un nou adversari dins d'aquest sector, fent competència real a les grans empreses hoteleres i immobiliàries dedicades al profit turístic, aquest és l'origen de les empreses on els seus clients o usuaris ofereixen les seves propietats privades com allotjaments d'estància curta per un preu mòdicament assequible. Aquest servei també es coneix com servei *bed and breakfast*. Airbnb [1] és una de les empreses més importants en aquest sector, on ofereixen aquest servei a través de la seva pàgina web d'una manera senzilla i propera al usuari real, convertint-se en tot un gegant dins del sector turístic.

Varis experts en el sector d'analítica de dades han començat a estudiar comportaments dels usuaris de Airbnb, realitzant estudis estadístics sobre les dades històriques de la seva pàgina web. InsideAirbnb [2] és un equip sense ànims de lucre que ha anat extraient les dades de Airbnb per realitzar estudis exhaustius per descobrir coses interessants, com per exemple estimar els ingressos dels amfitrions per calcular la rendibilitat dels usuaris de Airbnb, prendre Airbnb realment com un negoci on es pot extreure un benefici real, etc. InAtlas [3] és una empresa espanyola dins el sector del Big Data i analítica de localitzacions que realitzen estudis estadístics sobre diferents entorns per ajudar a prendre decisions dins negocis. Aquest equip també s'ha centrat en estudiar l'activitat present en Airbnb en Barcelona, proporcionant informació sobre percentatges d'ocupació dels allotjaments, percentatge dels usuaris oferint més d'un allotjament, etc. Airbnb mateix té un equip expert en la ciència de les dades que de tan en quant també van extraient informació interessant sobre l'ús dels usuaris sobre la seva pàgina web, com per exemple com modelar un algoritme que ajudi a escollir un preu per un habitatge segons les seves característiques, el alts i baixos del mercat del sector turístic en la ciutat corresponent, la seva localització, etc [4, 5]

Aquest projecte es centra en portar un estudi estadístic de l'activitat dels usuaris de Airbnb en Barcelona mitjançant el desenvolupament d'un model predictiu del valor dels allotjaments turístics. En aquest projecte s'ha intentat portar tots els estudis anomenats en el paràgraf anterior un pas més enllà mitjançant la combinació de les diferents idees introduïdes més la implementació de tècniques de la ciència de la computació sobre les dades per obtenir una sèrie de resultats que intentaran reforçar i millorar el màxim possible el funcionament del model predictiu.

Barcelona resideix entre les 10 ciutats europees més visitades i les que inverteixen més diners en millorar els serveis del sector turístic [6]. Per aquesta i mil raons òbvies més, és interessant també desenvolupar un estudi descriptiu de les dades obtingudes de Airbnb on il·lustrarà l'activitat dels usuaris seguit d'una sèrie d'estimacions dels ingressos obtinguts emprant aquest servei. Una pregunta molt general que ens fem aquí és: quan de rentable és emprar Airbnb donades les meves propietats amb les seves concretes característiques? Una part d'aquest projecte es centrarà en contestar aquesta pregunta juntament amb algunes més mitjançant el desenvolupament d'un model ocupacional inspirat per el projecte InsideAirbnb.

En el camp del modelatge estadístic existeix l'anàlisi de regressió, que es defineix com un procés estadístic que estima les relacions entre variables, concretament es focalitza en l'estudi de la relació entre una variable dependent (resposta) i una o més variables independents (predictors). Aquest àmbit dins de l'estadística ha estat molt explotat i profunditzat des dels seus començaments, donant pas a una infinitat de sistemes de predicció. En aquest projecte s'empraran una sèrie de models estadístics per il·lustrar la diferència i contrast que existeix en l'aplicabilitat de cada un, ja que un model no està considerat com "bo" o "dolent", sinó que cadascú té les seves avantatges i inconvenients de cara a un problema en concret. El problema aquí està en saber escollir entre les infinites possibilitats, els millors models que es poden ajustar millor a les dades del problema. Es farà ús de sistemes de regressió simples com són la regressió lineal i models de regularització (Ridge, Lasso, Bayesian Ridge), sistemes més complexos que es basen en la combinació de diferents models débils per crear un model més fort (mètodes ensemble, com el Gradient Boosting Regressor), mètodes d'optimització de resultats dels models (com la cerca dels millors paràmetres en un clúster de Apache Spark [7]), i en l'extracció dels resultats d'un model de predicció mitjançant arbres per crear visualitzacions interessants dependent del valor de cada allotjament (els anomenats *microneighbourhoods* [4]).

Finalment, el projecte acabarà amb la implementació d'una aplicació del model estadístic sobre una imatge de la plataforma Docker [8]. El desenvolupament d'aplicacions sobre aquesta plataforma està dins del *state-of-the-art* de l'entorn de la ciència de les dades degut principalment a la seva àmplia configurabilitat, escalabilitat, robustesa i facilitat d'ús que proporciona. Tot científic de les dades dessitja que el seu projecte tingui una aplicabilitat dins el món real, això és, que la gent emprí el seu model per obtenir un profit. La principal desvantatge d'això és en l'àmpli rang de dependències que es necessita per executar els programes que implementen aquests models, sumant-li que cada dependència és única en cada sistema operatiu (Mac OS, Windows, Linux...). Docker proporciona una arquitectura que permet "instal·lar i contenir" totes les dependències hardware i software sobre una única plataforma, de manera que cada cop que algú desitgi emprar l'aplicació de Docker, simplement haurà d'executar el programa mitjançant una simple comanda sense preocupar-se de les dependències.

L'estructura del projecte es divideix en 5 capítols principals: el capítol 2 consisteix en explicar la metodologia que seguirà el projecte per implementar cada un dels objectius que s'han exposat; el capítol 3 profunditzarà en detall tècnic l'aplicació dels mètodes que s'han il·lustrat en la secció de metodologia; el capítol 4 exposarà els resultats obtinguts de tot el projecte juntament amb l'assoliment dels objectius; el capítol 5 consisteix en discutir i entendre els resultats obtinguts, seguit d'una projecció dels resultats cap a un treball futur; i finalment el capítol 6 es redactaran les conclusions extretes durant el transcurs del desenvolupament del projecte, què s'ha demostrat i què no, què s'hagués pogut millorar o canviar, etc.

CAPÍTOL 2: METODOLOGIA

Aquesta secció es focalitza en anomenar i explicar les diferents metodologies i tècniques que ha seguit el projecte per realitzar cada un dels objectius exposats d'una forma abstracta i encarada al àmbit informatiu i conceptual. No s'entraran en detalls tècnics i d'implementació en aquesta secció.

El projecte segueix l'estructura d'un projecte de ciència de les dades i com tal implica una sèrie d'etapes, aquestes són: tècniques d'obtenció del conjunt de dades d'interès; neteja i preparació de les mateixes per convertir-les en unes dades de qualitat aptes per aplicar-hi una recerca estadística; exploració de les dades, incloent un anàlisi exhaustiu de les dades acompanyat amb visualitzacions i dades interessants; desenvolupar un sistema predictiu emprant diferents tècniques i models estadístics; i per últim, però opcional, implementar una aplicació on es poden emprar directe o indirectament els resultats obtinguts en les parts anteriors.

2.1 Obtenció de les dades

Com ja s'ha comentat, com allotjament turístic es prendran els allotjaments de renta presents a la plataforma de Airbnb. Cal anotar que la plataforma Airbnb no ofereix un dipòsit públic de les dades dels seus allotjaments. Llavors l'obtenció de les dades s'ha realitzat seguint una metodologia d'extracció de dades de pàgines web anomenada *web scraping* [9]. Tot procés d'extracció s'ha portat a terme seguint els protocols ètics i de privacitat cap als autors de les dades (dels allotjaments), això és, no conservar informació confidencial com noms propis, enllaços a pàgines web personals, etc.

La tècnica *web scraping* és un mètode d'extracció de dades directament del codi font d'una pàgina web. Aquestes dades són indirectament accessibles, per indirectament ens referim a que no es segueix un mètode trivial d'obtenció d'aquestes i per accessibles es refereix a que estan disponibles a algun lloc dels recursos disponibles de les pàgines web, com per exemple les rutes *Xpath*. Tot el procés del desenvolupament del *scraper* es comentarà en més profunditat en la secció Desenvolupament.

En termes d'emmagatzemat de les dades s'ha preferit d'un sistema de base de dades distribuït facilitat per l'empresa PostgreSQL [10] degut principalment a la seva arquitectura intuïtiva i facilitat d'ús. La fàcil implementació dels mètodes i transaccions de base de dades aportats per l'API descrita en la documentació ha sigut

un terme clau al escollir aquest sistema com a forma d'emmagatzemar les dades del projecte.

2.2 Neteja i exploració de les dades

Les parts de neteja i preparació de les dades obtingudes mitjançant el procés de web scraping en unes dades de qualitat es desenvoluparà en el mateix entorn on s'implementarà la part d'exploració de les mateixes dades. Això és degut principalment a que un procés de neteja de les dades depèn en gran part de l'ús que es farà de les mateixes en un futur. La preparació d'unes dades de qualitat és un procés crucial dins un projecte d'estudi dins l'àmbit de la ciència de les dades ja que les són el recurs més important i determinant de la qualitat dels resultats.

Les eines principals que s'empraran dins del procés d'exploració i anàlisi de les dades seran l'entorn de programació en el qual es processaran les dades, juntament amb enriquir les assumpcions i conclusions obtingudes amb recursos externs com articles i estudis estadístics semblants al tòpic introduït, visualització de dades emprant llibreries de programació, etc. El desenvolupament de l'estudi es descriurà a la secció de Desenvolupament i els resultats a la secció de Resultats.

2.3 Desenvolupament del model predictiu

Aquesta secció es centrarà primer de tot en introduir què és un model de predicció estadístic i el seu funcionament general amb l'objectiu de facilitar l'enteniment conceptual, entrant molt poc en detalls. Tenir una base introductòria sòlida de quines mètriques que busca un model predictiu és important per llavors entendre les tres diferents tècniques que s'han portat a terme per construir els models. A continuació s'exposaran els diferents mètodes de predicció, que són: desenvolupament d'un model predictiu a partir d'implementar diferents sistemes de regressió lineal juntament amb algunes tècniques d'optimització; extracció de les zones imaginàries que delimita la distribució del preu dels habitatges, anomenats *microneighbourhoods*, de Barcelona a partir de les regions de classificació obtingudes mitjançant un mètode de regressió per arbres; i per últim, extreure característiques de les imatges dels allotjaments mitjançant l'ús de xarxes neuronals convolucionals (CNN) per després aplicar-les al model.

2.3.1 Introducció als models predictius

El desenvolupament d'un sistema predictiu o un model estadístic de predicció consisteix en la construcció d'un model que entén les dades que es passen com entrada o *input*, per predir o estimar una sortida o *output* a partir d'una sèrie de variables o predictors proporcionats per les dades d'entrada. Aquest model el podem

entendre com un model de busca patrons i similituds dins les dades a partir de definicions i fòrmules matemàtiques, concretament aplicades dins de l'entorn de l'estadística, per després estimar un valor o categoria d'una observació a testejar. Com s'ha dit, aquest projecte es centra amb la pregunta de: podem predir el preu dels allotjaments turístics a partir de les característiques dels mateixos? On segueix la següent pregunta, una mica més concreta, que formula: com podem entendre les imatges dels habitatges de manera que sabem que una és millor que l'altre i que llavors tindrà un pes major en el rol d'estimar un preu? Totes aquestes preguntes les intentarem respondre a partir dels resultats del nostre model de predicció.

En conclusió la resposta o *output* a estimar és el preu dels allotjaments, llavors s'ha d'estudiar quins mètodes o, comunament anomenats dins l'aprenentatge automàtic, *learners*, es poden emprar per treure el màxim profit de les dades d'entrada. Depenent de l'estructura de les dades un model funcionarà millor que un altre, quan diem funcionar ens referim a la capacitat que es té per estimar un valor de la variable de sortida amb un error mínim, això és, intuïtivament, minimitzar la discrepància entre el valor estimat i el valor real de l'observació. Aquesta metodologia varia segons el tipus de resposta, ja que depèn si és una variable de resposta numèrica o categòrica el model d'avaluació de l'estimació serà diferent. Com que al nostre projecte el preu és una variable numèrica, llavors té sentit que només anomenem la mètrica d'avaluació d'una resposta numèrica. N'hi ha moltes, però la més comuna és la mètrica dels *least squares* (LS). Llavors, tenint una base de dades de la forma $\{y_i, x_{i1}, \dots, x_{ip}\}_{i=1}^n$ on n és el nombre total d'observacions del conjunt de dades, y_i és el valor real de la resposta a la i -èssima observació i x_{ip} és el valor i -èssim de la variable explicativa o predictora X del p -vector dimensional. Aquest p -vector representa els p predictors de les dades. Llavors *least squares* intenta minimitzar la *Residual Sum of Squares* (RSS), que té la forma següent:

$$RSS = \sum_{i=1}^n (y_i - f(x_i))^2 \quad (2.1)$$

On $f(x_i)$ és la funció que ajusta el model de predicció per obtenir el valor predit de l'observació i -èssima, es a dir, estimació o predicció de y_i . En altres paraules, el RSS desitja explicar com de bé s'ajusta el model a les dades a partir de la diferència absoluta del valor predit amb el valor real de les observacions tractades. Un RSS amb valor baix significarà que l'error és petit i llavors el model s'ajusta bé. Aquest mètode s'anomena un mètode de selecció de models o *model selection*.

Com s'ha dit anteriorment, hi ha diferents mètodes de selecció de models depenent de l'aplicació que es faci referència. Quan estem parlant d'una variable de resposta de valors numèrics el primer model que podem pensar d'aplicar és un regressor lineal. Un regressor lineal es defineix com un model estadístic que és especialista predir o estimar una variable de resposta numèrica y a partir d'una variable explicativa o predictora X. Aquesta variable explicativa pot ser de dos tipus, simple

(incloent un sol predictor, p sent igual a 1) o múltiple (incloent més d'un predictors, p major que 1). Segons la dimensió de X estarem parlant de *simple linear regression* o *multivariate linear regression*. Per agilitzar, el que s'ha de saber d'aquest sistema és la forma que té la funció per entrenar el model, que en el cas d'aplicar-la al RSS, equivaldria a ser la funció $f(x_i)$. Segueix la forma següent:

$$f(x_i) = \beta_0 + \sum_{j=1}^p \beta_j x_{ij} + \epsilon_i \quad (2.2)$$

On β és un vector de $p+1$ dimensions on cada element β_i s'anomena coeficient de regressió. L'estimació del valor es basa a partir d'aquests coeficients que són calculats a partir de minimitzar el *least squares* de cada variable x_{ij} a la que s'emparella [11], veure Figura 1. Per a més detalls de com es calculen aquests coeficients vegeu [12].

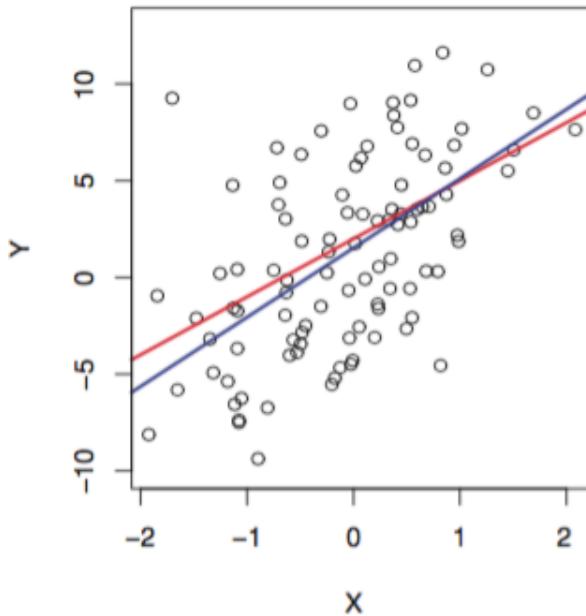


Figura 1. Gràfica realitzada amb unes dades simulades. La línia vermella representa la relació real que hi ha entre x i y , que es conevida com la línia de regressió popular. Mentre que la línia blava és la línia dibuixada a partir dels resultats d'aplicar el least squares a $f(x)$. [12]

Una vegada introduït el concepte d'un model de predicción, concretament un model de regressió lineal i el seu mètode d'avaluació de resultats, podem donar pas a les seccions que expliquen què s'ha fet en aquest projecte.

2.3.2 Sistemes de regressió

Aquesta secció té l'objectiu d'introduir de manera conceptual els sistemes de regressió lineals que s'han emprat en aquesta part del projecte juntament amb les tècniques d'optimització necessàries per millorar els resultats obtinguts. Els detalls de la implementació es troben redactats a la secció Desenvolupament.

Com ja s'ha mencionat, degut principalment a que la variable de resposta a predir és una variable numèrica, s'empraran sistemes de regressió ja que després de la seva invenció es sap que són els mètodes que funcionen millor de cara a variables escalars. Si la variable fos de tipus categòric (valors en etiquetes), llavors empraríem mètodes de classificació que segueixen una estructura semblant als regressors. Els regressors lineals que s'han aplicat en aquest projecte són: *Lineal Regression*, *Ridge Regression*, *Lasso Regression*, *ElasticNets*, i *Bayesian Ridge*.

A continuació s'explicarà el funcionament, els detalls i perquè s'han escollit cada un d'ells, a excepció de la regressió lineal ja que ja s'ha explicat en la secció anterior, simplement es descriurà perquè s'ha escollit.

2.3.2.1 Regressió Lineal

El Regressor Lineal és un dels primers sistemes d'anàlisi de regressió que varen ser exhaustivament estudiats, i més endavant emprats en una enorme quantitat d'aplicacions. Això és degut a la seva fàcil aplicabilitat, interpretabilitat i a la gran quantitat d'informació que proporciona els seus resultats.

És un dels primers models que s'han de pensar en aplicar sobre un problema de regressió. Entre les seves avantatges hi ha la trivial il·lustració de la relació o sinergia que tenen les variables predictores amb la resposta, això crea dues avantatges més: primer ajuda a tenir una idea de quines variables tindran més importància en el model sobre unes altres, i segon, dependent de les dimensions de les dades (nombre de predictors p), aquesta relació es pot veure representada gràficament sobre dues dimensions i llavors arribar a millor conclusió conceptual.

A pesar de la seva senzilla estructura matemàtica, la regressió lineal funciona sorprendentment bé en algun tipus de dades, sobretot en aquelles on hi ha un nombre de predictors petit i dades voluminoses, concretament quan $n \gg p$. Però quan això no passa el model sol acabar estant sobre-entrenat (*overfitting*) o poc entrenat (*underfitting*). El combat contra *overfitting* i *underfitting* és un dels problemes més importants a resoldre dins del món del aprenentatge automàtic i mineria de dades. Es tracten d'uns dels principals problemes per a que un model no s'ajusti bé a unes dades reals de test, es a dir, unes observacions completament noves fora del conjunt d'entrenament del model.

2.3.2.2 Ridge Regression

La Regressió Ridge o també anomenada *Tikhonov regularization* [13] és una variant de la regressió lineal encarada a problemes de regularització. En termes d'estadística, regularització o *shrinkage* fa referència a un procediment que té l'efecte que els coeficients d'estimació tendeixen a reduir-se cap a zero en relació al mètode de selecció del model (per exemple, minimització de (2.1)). Alguns mètodes de regularització fan que els coeficients acabin sent exactament zero, però altres només per a un cert paràmetre *shrinkage* molt gran. Aquest procediment redueix considerablement la variància del model i en conseqüència lluita millor contra el problema del *overfitting*. Com s'ha dit, depenent del mètode de regularització aplicat, alguns dels coeficients poden acabar sent reduïts fins a exactament zero. Llavors estarem parlant d'un mètode que també aplica selecció de variables, es a dir, un mètode que selecciona un subconjunt de predictors m on $m < p$.

Si recordem la formulació de l'error RSS (2.1) on en aquest cas la funció $f(x_i)$ és la mateixa que la de regressió lineal descrita en (2.2), llavors *Ridge Regression* és molt similar al *least squares*, però la funció de minimització està formulada una mica diferent. Concretament, els coeficients de la regressió Ridge són valors que minimitzen

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p \beta_j^2 \quad (2.3)$$

on $\lambda \geq 0$ és un paràmetre d'ajustament o *shrinkage parameter*, assignat amb anterioritat. Aquesta equació es coneix com la penalització L2. L'equació (2.3) compensa dos criteris molt importants: els coeficients estimats busquen minimitzar l'error RSS, com la regressió lineal; i mitjançant el segon terme, $\lambda \sum_{j=1}^p \beta_j^2$, anomenat penalització de regularització o *shrinkage penalty*, dóna un valor petit quan els coeficients β_1, \dots, β_p són propers a zero, llavors té l'efecte explícit anteriorment de reducció del valor dels coeficients. El paràmetre λ serveix per controlar el grau de penalització que s'aplica sobre els valors dels coeficients. Quan $\lambda = 0$, vegeu que no hi ha penalització i llavors el segon terme no tendrà efecte sobre (2.3), i llavors estarem parlant d'un mètode que produirà una minimització del *least squares* (com la regressió lineal). Però, quan $\lambda \rightarrow \infty$, l'impacte de la penalització és major i llavors els coeficients seran molt propers a zero. El millor valor de λ es troba realitzant *cross-validation*.

La regressió Ridge funciona millor quan hi ha un nombre alt de predictors altament correlacionats entre ells. En estadística, això pot ser un problema de cara a la construcció d'un bon model que s'ajusti bé a les dades, aquest problema s'anomena *multicollinearity* [14]. Quan això passa, *least squares* mostra un valor de *bias* petit però un valor de variància molt gran, llavors el model tendirà a estimar valors lluny

del valor real, com està visualitzat en la Figura 2. Per aquesta raó s'ha de visualitzar les diferents combinacions de predictors entre ells per veure si hi pot haver un problema de *multicollinearity*. En general la *Ridge Regression* funciona bé quan es té aquest problema.

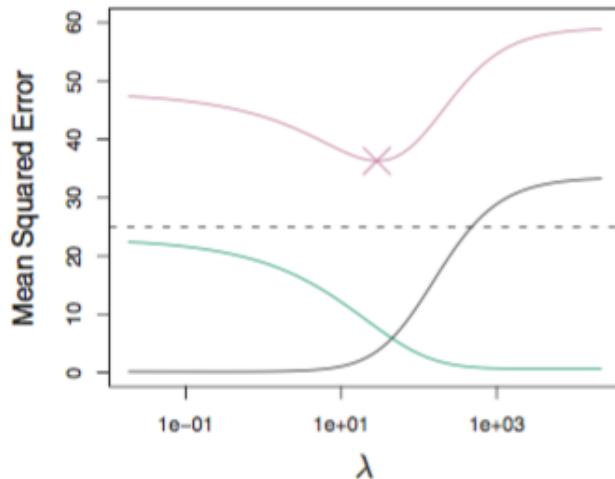


Fig 2. Representació del impacte del paràmetre λ . En vermell es mostra l'error MSE, en verd es representa la variància i en negre el bias. Es pot apreciar la baixada de variància i pujada del bias a mesura que augmenta el valor de λ . [12]

2.3.2.3 Lasso Regression

La regressió Lasso és un mètode de regressió lineal que realitza un procés de regularització i selecció de variable per millorar la precisió de predictibilitat i interpretabilitat del model estadístic que produeix. També es coneix com regressió L1. Com la *Ridge Regression*, Lasso també és un mètode de regularització i llavors fa tendir els coeficients cap a 0 afegint una penalització a la funció que minimitza l'error. Però, a diferència de Ridge, Lasso és un mètode de selecció de variables perquè fa que alguns coeficients, per certs valors del *shrinkage parameter*, siguin exactament zero. Això com ara veurem, té alguns avantatges i inconvenients.

Lasso calcula els seus coeficients de manera que minimitzin la funció següent:

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p |\beta_j| \quad (2.4)$$

On tots els paràmetres emprats són exactament els mateixos que la funció de regressió Ridge, incloent el paràmetre de regularització λ . La diferència aquí està en el valor absolut del segon terme $\lambda \sum_{j=1}^p |\beta_j|$, el qual implica que per certs valors de λ

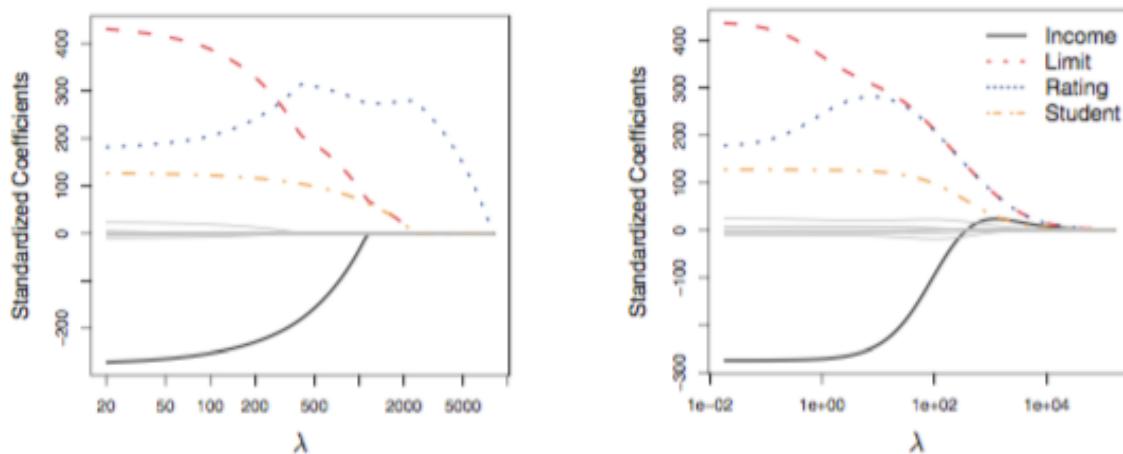


Fig 3. Representació del comportament dels coeficients a mesura de que λ augmenta o disminueix. A l'esquerra s'aprecia el comportament aplicant Lasso, on efectivament els valors acaben sent exactament zero; i a la dreta Ridge, on els valors s'aproximen però no arriben a ser zero fins a uns valors molt alts de λ . [12]

els coeficients seran exactament zero i llavors, com s'ha dit, Lasso és un mètode de regularització i selecció de variables al mateix temps.

Vegeu la Figura 3 per apreciar el comportament dels valors dels coeficients dependent del valor de λ , tant per *Ridge Regression* i *Lasso Regression*. El millor valor de λ però s'escull via *cross-validation*.

La principal diferència entre Ridge i Lasso és el seu entorn recomanat d'aplicabilitat. Ridge com s'ha dit funciona millor quan hi ha *multicollinearity*, Lasso en canvi funciona millor quan els predictors no estan correlacionats entre ells. La principal millora del Lasso és la interpretabilitat que guanya el model, fent baixar considerablement la variància i llavors l'error de predicció del model, amb el cost d'un petit augment del bias.

2.3.2.4 Elastic Net

Elastic Net és també un model de regularització que combina linealment les penalitzacions L1 i L2 per crear un únic i millor classificador.

Lasso Regression presenta una sèrie de limitacions aplicant el terme *shrinkage* $\lambda \sum_{j=1}^p |\beta_j|$ (veure fórmula (2.4)). Per exemple, en el cas de que p sigui gran i n petit (dades amb moltes dimensions però poques observacions), Lasso selecciona com a màxim n variables abans de que convergeixi. A més si hi ha un nombre de variables

molt correlacionades entre elles, Lasso tendeix a seleccionar una variable d'un grup i ignorar les altres, disminuint la capacitat de predir acuradament perquè la variància del model augmenta. Llavors per eliminar aquestes limitacions, introdúim el mètode *naïve Elastic Net*, el qual afegeix la part quadràtica de la penalització $|\beta_j|^2$, és a dir, la que s'empra en la regressió Ridge. Els coeficients del mètode Elastic Net estan definits de la manera següent:

$$\hat{\beta} = \operatorname{argmin}_{\beta} (\|y - X\beta\|^2 + \lambda_2 \|\beta\|^2 + \lambda_1 \|\beta\|_1) \quad (2.5)$$

Aquest procés es pot veure com el mètode *least squares* amb penalització. Sent $\alpha = \lambda_2 / (\lambda_1 + \lambda_2)$; llavors resoldre $\hat{\beta}$ en l'equació (2.5) és equivalent al problema d'optimització

$$\hat{\beta} = \operatorname{argmin}_{\beta} (\|y - X\beta\|^2) \quad \text{subject to } (1 - \alpha) \|\beta\|_1 + \alpha \|\beta\|^2 \leq t \text{ for some } t \quad (2.6)$$

El terme $(1 - \alpha) \|\beta\|_1 + \alpha \|\beta\|^2$ s'anomena la penalització Elastic Net [15], la qual és una combinació convexa de la regressió Ridge i Lasso. Per convexa ens referim a que només té un únic mínim. Quan $\alpha = 1$, la *naïve elastic net* es converteix en una simple regressió Ridge. Un cop dit això, donam pas a la funció millorada de Elastic Net, la qual minimitza (2.6) per obtenir els seus coeficients [16].

$$\hat{\beta} = \min_{\beta} \left(\frac{1}{2n} \|y - X\beta\|^2 + \alpha \|\beta\|_1 + \frac{(1 - \alpha)}{2} \|\beta\|^2 \right) \quad (2.7)$$

Una de les avantatges de (2.7) és, a part de que millora la desavantatge de la regressió Lasso comentada abans, té la capacitat d'elegir un grup de predictors correlacionats entre ells. De manera que mentre que Lasso és probable que agafi aleatoriament una variable d'aquest grup, Elastic Net les agafarà totes fent que es millori el rendiment del classificador per un cas en el que hi ha moltes variables correlacionades entre elles. Aquesta capacitat és coneguda com el *grouping effect* [33]. Segons molts estudis, es sap que Elastic Net proporciona un millor classificador que Lasso en el cas de que les dades presentin moltes variables correlacionades entre elles.

2.3.2.5 Bayesian Ridge Regression

Bayesian Ridge és una de les tècniques de regressió Bayesiana, les quals s'empren per introduir paràmetres de regularització a l'estimació. Al igual que Ridge, Lasso i Elastic Net, la Bayesian Regression és també un mètode de regularització, però la

diferència és que els paràmetres *shrinkage* no s'assignen amb anterioritat, sinó que es van afinant a mesura que s'aplica el mètode segons les dades del problema.

Això es pot fer mitjançant la introducció de *uninformative priors* (això és, valors que expressen informació general o imprecisa sobre una variable) als paràmetres del model. La regularització L2 que s'empra en Ridge és equivalent a trobar una solució *posteriori* màxima sota les *priors* Gaussianes sobre els paràmetres λ amb precisió $w - 1$ [15]. En lloc d'assignar λ manualment, és possible de tractar-la com una variable aleatòria per ser estimada a través d'entrenar el model amb les dades. La principal avantatge de les Regressions Bayesianes és que adaptant-se a les dades proporcionades, es pot arribar a calcular paràmetres de regularització per després ser emprats en l'estimació dels coeficients del model. La principal desavantatge però és la seva poca interpretabilitat.

La *Bayesian Ridge Regression* és una variant de la regressió Bayesiana. No s'entrarà molt en detall sobre la seva formulació però s'ha de saber que els paràmetres són tractats com a variables aleatòries de manera que són estimats a mesura que es va entrenant el model. Aquests paràmetres són estimats de manera que maximitzin la funció *marginal log likelihood*. Si vegeu la Figura 4, es pot veure representat el comportament de la Bayesian Ridge en el càlcul dels coeficients comparat amb el mètode tradicional *Ordinary Least Squares (OLS)*. Es pot apreciar que els coeficients calculats mitjançant Bayesian Ridge són regularitzats cap a zero, de manera que els estabilitza. Aquest efecte fa la Bayesian Ridge Regression un classificador més fort sobre *ill-posed problems*. Breument, un *ill-posed problem* és aquell que no es pot aplicar un clàssic mètode estadístic d'estimació (com el least squares) ja que proporcionen una estimació molt pobre i imprecisa del terme *bias* del model. Llavors aquests problemes fa que un petit canvi a les dades tendeixi a canviar completament el funcionament del model i llavors caure més fàcilment en *overfitting* [16].

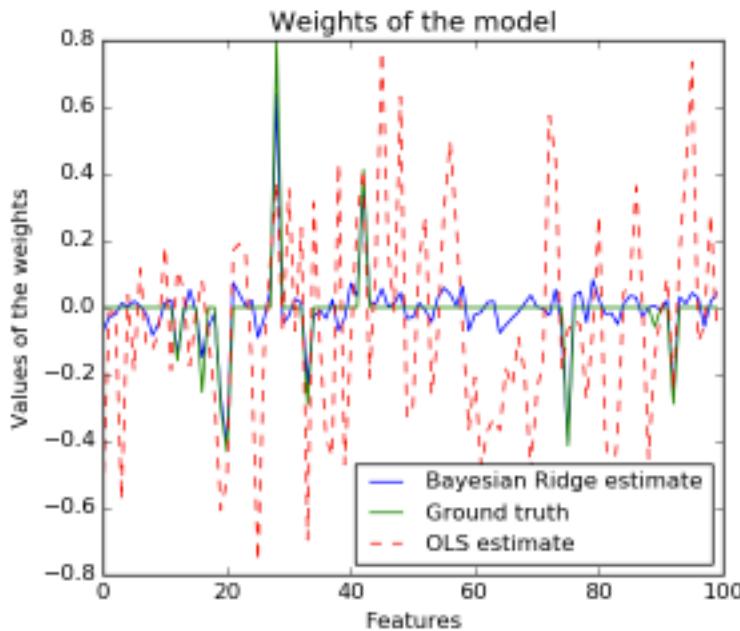


Fig 4. Representació del comportament dels coeficients mitjançant Bayesian Ridge i Ordinary Least Squares. [15]

2.3.3 Ensemble Learning

En l'àmbit d'aprenentatge automàtic i mineria de dades, un classificador ensemble és un mètode que combina una sèrie de classificadors dèbils o simples (regressió lineal, Ridge, etc) per obtenir un classificador més fort amb menys probabilitats d'error. La idea principal d'un mètode ensemble és ajuntar models dèbils amb alta variància sobre la resposta (es a dir, classificadors que simulen un comportament semblant en classificació aleatòria del 50%) però poca relació entre ells per obtenir un model més sólid amb menys variància, menys bias i llavors més generalització (menys error en el conjunt de test). D'entre els mètodes ensemble més coneguts hi ha els *Random Forests* i *Adaptive Boosting*. Existeix una millora del Adaptive Boosting anomenada *Gradient Boosting Regressor*, la qual és la que s'ha emprat en aquest projecte.

2.3.3.1 Adaptive Boosting

Per entendre el mètode ensemble Gradient Boosting Regressor (GBR) primer s'ha de saber en què consisteix el mètode de Adaptive Boosting o AdaBoost.

El mètode AdaBoost, creat per Yoav Freund i Robert Schapire en el seu treball [17], consisteix en que la sortida d'una sèrie d'algoritmes d'aprenentatge dèbils es combinada en una suma amb pesos que representa la sortida final del classificador Boosting. Aquesta suma de pesos és perquè el AdaBoost és adaptatiu, es tradueix com que en els classificadors dèbils se lis assigna un petit canvi a favor d'aquelles instàncies mal classificades (o amb molt error en el cas de regressió) per els classificadors anteriors. AdaBoost es sap que és un mètode ensemble que aprèn lentament, millorant a poc a poc el resultat dels classificadors obtenint un classificador final fort sobre les debilitats dels dèbils. L'estructura d'aquest mètode és de generar un nombre B d'arbres petits (normalment arbres de profunditat 1 o *stumps*, arbres de profunditat 2 o profunditat 4), on cada sortida és tractada amb una penalització λ que controla la seva variància.

2.3.3.1.1 Gradient Boosting Regressor

El mètode GBR funciona d'una manera similar que AdaBoost però se li afegeix una funció *loss* arbitrària que permet optimitzar la sortida de cada sub-arbre (classificador dèbil) de forma que generalitza millor la sortida final (reduïx més el bias i la variància). Aquesta idea va ser principalment originada per l'observació de Leo Breiman [18] que diu que Boosting pot ser interpretat com un algoritme d'optimització sobre una funció *loss* adaptable. Després Jerome H. Friedman [19, 20] va desenvolupar el mètode de Gradient Boosting Regression.

Com altres mètodes de Boosting, GBR combina classificadors dèbils en un classificador més fort, en una forma iterativa. L'objectiu és crear un model F que prediu valors $\hat{y} = F(x)$, on $F(x)$ minimitza el MSE $(\hat{y} - y)^2$ on y són els valors reals de les observacions testejades. En cada iteració b de les B iteracions del gradient boosting, es creu que hi ha un model imperfecte F_b de manera que adaptativament es va canviant la sortida de cada model creat en cada iteració de manera que

$$F_{b+1}(x) = F_b(x) + h(x)$$

On h és una estimació que intenta crear un millor model a cada iteració. Com es pot trobar h ? La qüestió és que h idealment hauria de satisfer

$$h(x) = y - F_m(x)$$

Llavors cada iteració entrena el model F_{m+1} a partir dels residuals $y - F_m(x)$, de manera que cada nou model creat és millor que l'anterior. La idea general del GBR és que la funció *loss* $y - F_m(x)$ és el gradient negatiu de la funció *loss* quadràtica de l'error $\frac{1}{2}(y - F_m(x))^2$ [21].

Més detalladament, GBR considera models additius de la forma

$$F(x) = \sum_{b=1}^B \gamma_b h_b(x)$$

on $h_b(x)$ és una específica funció *loss*. Additius perquè en cada iteració es re-entrena la funció $F(x)$ de la manera següent:

$$F_b(x) = F_{b-1}(x) + \gamma_b h_b(x)$$

A cada iteració la funció *loss* és minimitzada de manera que la funció $F_b(x)$ ens queda de la forma:

$$F_b(x) = F_{b-1}(x) + \operatorname{argmin}_h \sum_{i=1}^n L(y_i, F_{b-1}(x_i) - h(x)) \quad (2.8)$$

En regressió, el model inicial $F_0(x)$ s'assigna mitjançant la mitja dels valors de test. GBR intenta resoldre el problema de (2.8) mitjançant d'un descens "brusc". La direcció d'aquest descens és el negatiu del gradient de la funció *loss* avaluada al model actual $F_{b-1}(x)$, on aquest gradient pot ser calculat mitjançant qualsevol funció *loss* diferenciable:

$$F_b(x) = F_{b-1}(x) + \gamma_b \sum_{i=1}^n \nabla_F L(y_i, F_{b-1}(x_i)) \quad (2.9)$$

On el valor del pas "brusc" γ_b es escollit satisfent la següent equació:

$$\gamma_b = \operatorname{argmin}_\gamma \sum_{i=1}^n L(y_i, F_{b-1}(x_i)) - \gamma \frac{\partial L(y_i, F_{b-1}(x_i))}{\partial F_{b-1}(x_i)} \quad (2.10)$$

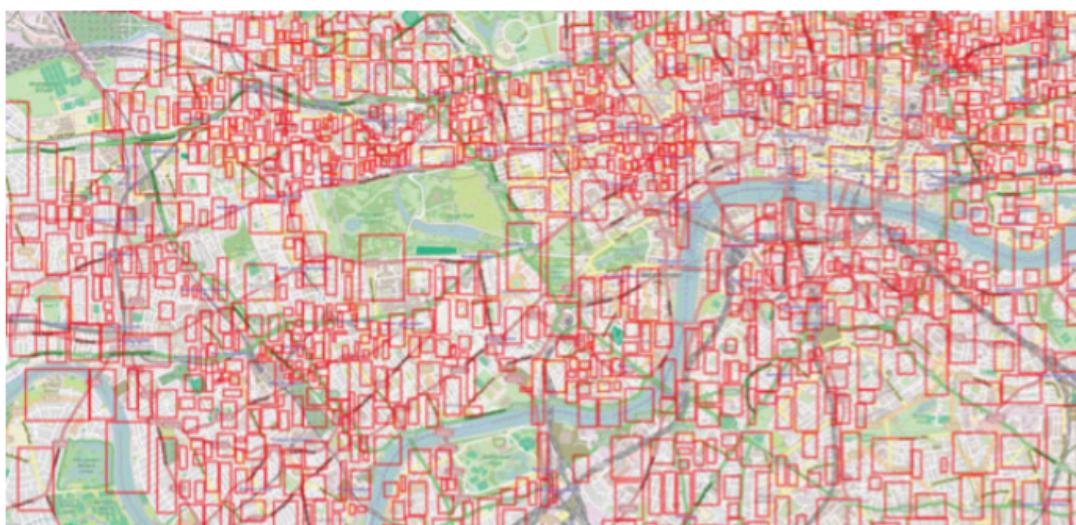
En conclusió, la peça clau d'un bon GBR és escollir la funció *loss* correcte. Per regressió, *least-squares* és una selecció natural degut a que sempre funciona bé dins els marges del cost computacional. També existeix la funció *least absolute deviation* que també funciona molt bé per regressió, on el model inicial $F_0(x)$ és escollit per la mediana dels valors de test.

2.4 Microneighbourhoods

Segons l'article publicat per l'equip de Data Science de Airbnb [4], és possible crear un classificador a partir d'algunes característiques clau dels habitatges, com per exemple el preu i la localització en coordenades, per a que dibuixi línies imaginaries que determinen *microneighbourhoods* virtuals sobre el mapa de Barcelona. De tal manera que cada *microneighbourhood* té un preu assignat i llavors els allotjaments residint dins del mateix tindrien aquest preu assignat com a estàndard. En aquesta secció es descriu quins mètodes s'han emprat per extreure i visualitzar els *microneighbourhoods* de Barcelona.

En la Figura 5 es pot veure el resultat d'aplicar un classificador per dibuixar els *microneighbourhoods* de San Francisco a partir de la localització i el preu dels allotjaments turístics de Airbnb. Observant aquest resultat un pot concloure que es tracta d'un classificador on traça regions rectangulars, sense tenir en compte el post-processament que s'ha aplicat per obtenir aquestes regions separades unes de les altres.

Un classificador senzill que pot realitzar el mateix treball és un *Decision Tree* (DT). Un DT és un model estadístic (regressor i classificador) que infereix una sèrie de regions de classificació en les quals totes les observacions testejades que resideixen dins d'una certa regió, se li assignarà el valor de la mitja dels valors les observacions d'entrenament que s'han emprat per entrenar el DT. El terme “Decision” ve donat perquè el model realitza una sèrie seqüencial de *splits* on en cada un d'ells es decideix anar per un fill o l'altre segons un cert valor major o menor que un altre. I “Tree” perquè aquesta sèrie seqüencial de *splits* es pot representar en forma d'arbre.



(/img/09OLAirBnBlondonMap-1439840175883.jpg)

Illustration: Airbnb

Fig 5. Microneighbourhoods inferits de la regió de San Francisco. [4]

Es pot apreciar que com més profunditat té l'arbre del DT, més nodes terminals hi haurà i llavors més regions classificatòries. Un altre paràmetre important és el nombre mínim d'observacions que resideixen dins cada node terminal. És molt important escollir el nombre correcte de manera que no es construeixin regions amb un nombre insignificant d'observacions, qual cosa no tindria gaire sentit en termes de crear un *microneighbourhood*.

En la secció de Desenvolupament i Resultats es veurà com s'ha aplicat un Decision Tree per obtenir les capses o regions de les quals s'han inferit els microneighbourhoods. S'ha emprat el recurs online de CartoDB per visualitzar els resultats finals.

2.5 Extracció de característiques de les imatges

Dels algoritmes coneguts dins del món de l'aprenentatge automàtic, les xarxes neuronals (*Neuronal Networks*), concretament les xarxes neuronals convolucionals (CNN), són els sistemes que s'ha demostrat que funcionen millor en els camps de reconeixement d'objectes sobre imatges. En una CNN la idea estructural segueix l'estructura del sistema neuronal humà. En una xarxa neuronal humana, cada neurona tracta una porció de la informació, per després ser enviada a altres neurones i finalment acabar ajuntant tota la informació inferida a les neurones més importants. Combinant diferents informacions obtingudes de porcions petites de l'objecte a tractar resulta en un model que pot entendre millor el problema però també trigarà més a ser entrenat.

Llavors en una CNN la idea seria, introduint una imatge com *input*, assignar a cada neurona principal (primera capa) cada píxel de la imatge juntament amb un pes individualment escollit. Cada neurona processa una funció simple, després a mesura que s'avança més profundament dins la xarxa, les neurones més profundes agrupen la informació obtinguda dels píxels individuals en grups, de manera que són tractats d'una forma més complexa. Detalls de l'estructura d'una CNN no entra dins l'àmbit d'aquest projecte i llavors no s'explicarà en profunditat. El que si que cal anotar és la demostrada altíssima capacitat de predictibilitat de les CNN, concretament la CNN Hybrid-PlacesCNN entrenada amb el conjunt de dades Places (més de 2 milions d'imatges amb més de 200 categories) per l'institut de tecnologia de MIT [22].

Malauradament, les CNN tenen la particularitat de que necessiten un volum molt gran d'imatges (milions d'unitats) per esperar una capacitat de predicción decent. Degut a aquesta limitació i al nombre d'imatges descarregades dels habitatges

(aproximadament 300.000), no s'ha pogut desenvolupar una CNN pròpia. Per això, s'ha fet ús de la Hybrid-Places-CNN per obtenir les característiques de les nostres imatges.

Les característiques obtingudes es combinaran al model predictiu lineal construït mitjançant els procediments explicats en la secció 3.3 per observar si es millora o no la precisió del classificador.

2.6 Selecció de variables

Un model amb molts de predictors tendeix a ser un model molt complex on la capacitat de generalització disminueix degut al *overfitting*. També un model amb tantes variables és més costós d'entrenar ja que s'han de computar molts més coeficients (un per cada predictor). Llavors existeixen solucions a aquests problemes que s'anomenen mètodes de selecció de variables, on prometen les millors:

- Combatre i disminuir l'*overfitting* del model.
- Construeix models més senzills i interpretables.
- Millora de la capacitat de generalització i llavors millora de l'error de predicció.

Per aquestes raons seria òptim aplicar un mètode de selecció de variables per veure si el model predictiu construït respon bé a les millors que suposaria aplicar-ho.

Hi ha un mètode bastant senzill però que assegura escollir el millor sub-conjunt de variables per al model: el *best subset selection*. El seu funcionament consisteix en provar cada possible combinació k de p predictors,avaluant el model resultant sobre la resposta mitjançant *cross-validation*. La complexitat de l'algoritme equival a realitzar $\binom{p}{k}$ per a cada valor de k tal que $1 \leq k \leq p$. El problema que té aquest mètode és obvi, el cost computacional que requereix calcular la millor combinació de variables creix exponencialment a mesura que n augmenta.

Una alternativa és la regressió L1 o Lasso Regression, un mètode molt conegut de selecció de variables on, segons el valor del *shrinkage parameter*, els coeficients acaben tenint un valor igual a zero conservant només aquells que ofereixen una major relació directe amb la resposta. Es sap que Lasso funciona bé quan tenim variables poc correlacionades entre elles. Ja que en cas contrari, l'algoritme tendeix a escollir una variable aleatòria d'un grup de variables molt correlacionades entre elles. Això farà que el model no esculli les millors variables per ajustar-se a les dades.

En aquest projecte es farà ús del mètode de *best subset selection*.

2.7 Aplicació Docker

Tot projecte del l'entorn de Ciència de les Dades necessita d'un enfocament d'aplicabilitat. En altres paraules, es necessita d'una manera que els usuaris usin el model predictiu que s'ha desenvolupat per treure-hi un profit directe o indirecte. Docker proporciona una arquitectura que satisfà tots els requisits per a que una aplicació tingui futur sobre qualsevol plataforma. La idea principal en desenvolupar una aplicació mitjançant Docker és aprofitar la portabilitat, integració i la facilitat d'ús que proporcionen les aplicacions creades amb aquest *framework*. Entre d'altres, les avantatges més importants de Docker són [8]

- Les dependències de l'aplicació ja no seran un problema. La majoria de les aplicacions dins de la Ciència de les Dades manegen una gran quantitat de llibreries pesades (numpy, scipy, pandas, scikit-learn, matplotlib, etc). Desenvolupant una aplicació Docker es té en compte aquestes dependències i són instal·lades en *background* quan s'executa el programa.
- Solució del problema del canvi constant de les infraestructures de les empreses actuals, fent que necessitis escalar l'aplicació cada cop que vulguis executar-la sobre un servidor diferent. Docker fa possible conservar tot el codi de la infraestructura i de l'aplicació, fent el sistema més manejable entre sistemes operatius.

Principalment per aquestes raons i perquè Docker està dins dels frameworks més emprats en el *state-of-the-art* en la ciència de la computació, en aquest projecte s'ha volgut desenvolupar i llançar una aplicació Docker per a un futur poder ser desenvolupada completament i treure el màxim profit.

CAPÍTOL 3: DESENVOLUPAMENT

En la secció de Metodologia s'han introduït els mètodes, funcions, i models que s'empraran en el projecte i la motivació del perquè s'han escollit. En aquesta secció s'explicaran en detall els procediments implementats emprant els mètodes exposats. Primerament s'explicarà el mètode de *web scraping* en detall per obtenir les dades; el desenvolupament de la secció descriptiva o exploratòria de les dades juntament amb la seva preparació de les dades; l'aplicació dels models predictius explicats a la secció de Metodologia sobre les dades i obtenció dels *microneighbourhoods* de Barcelona; utilització de la xarxa neuronal convolucional Places-CNN de Caffe per obtenir les característiques de les imatges per després ajuntar-les al model predictiu; i finalment com s'ha implementat l'aplicació Docker integrant el model dels microneighbourhoods.

3.1 Obtenció de les dades

3.1.1 *Web Scraping*

L'obtenció de les dades dels allotjaments turístics de Barcelona s'ha portat a terme mitjançant la tècnica de *web scraping* sobre la pàgina web oficial de Airbnb. L'estructura que es seguirà és la que es mostra en la Figura 6. Tot el procés està implementat sobre un fitxer de codi en llenguatge Python, versió 3.4.

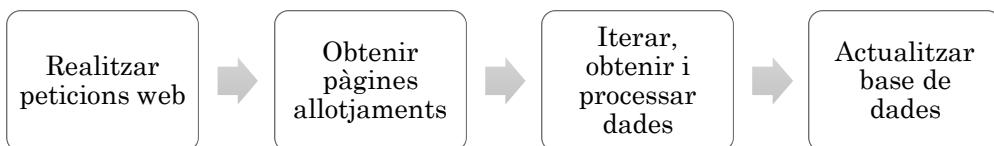


Fig 6. Esquema del procediment per realitzar el scraping a la pàgina web de Airbnb.

A continuació s'explica en detall cada apartat de l'esquema de la Figura 6.

3.1.1.1 Realitzar peticions web

Tot procés de *web scraping* necessita d'una eina que realitza les peticions a la web per obtenir informació de la mateixa. En el cas d'aquest projecte, s'ha fet us de la llibreria *requests* per realitzar totes les peticions degut a la seva fàcil interfície i gran flexibilitat d'aplicació de paràmetres opcionals. Un dels paràmetres més importants per al procés és la utilització de servidors *proxy*.

Degut principalment a que les peticions a la pàgina web es faran des d'un script i que el flux de peticions serà molt ràpid i voluminos, s'ha de tenir en compte en la implementació perquè normalment les pàgines web amb molt de tràfic, com és el cas de Airbnb, solen tenir un sistema de seguretat que bloqueja de seguida qualsevol IP que realitza peticions a la web un nombre de cops molt elevat en poc temps. Aquesta mesura de seguretat assegura que l'ample de banda servidor web no es col·lapsi, evitant implicar operacions de manteniment molt llargues i costoses.

Per solucionar aquest problema, entre peticions s'ha implementat un *sleep* que "adorm" el programa un nombre aleatori de segons, d'aquesta manera les peticions es faran més lentes i seguint un patró aleatori. A més, s'ha codificat un algorisme que permet canviar les direccions IP d'origen de les peticions mitjançant l'assignació de diferents direccions *proxy*. Aquests dos canvis implicaran que les peticions es realitzin de manera aleatòria amb un temps variable, on cada IP que realitza la petició serà diferent a l'anterior. Aquesta metodologia farà impossible a la pàgina web de detectar que un programa s'està executant sobre ella. Les direccions *proxy* es materialitzen com a direccions IP i ports on es passen com a input al servidor *proxy* web, de tal manera que aquest servidor empra la direcció *proxy* com a identificador per realitzar la petició a la pàgina web. Aquestes direccions han estat inicialitzades al arxiu de configuració que llegeix el programa a l'inici de la seva execució.

3.1.1.2 Obtenir pàgines web dels allotjaments

Un cop s'ha aconseguit un sistema prou robust i segur per accedir a la pàgina web, es hora d'obtenir les dades de les habitacions. El primer pas que s'ha de realitzar és iterar sobre totes les pàgines que mostren varies habitacions, si es mira la pàgina de Airbnb per la ciutat de Barcelona s'apreciarà que es mostren un nombre de 10 o 15 habitacions per pàgina. Llavors el que s'ha de fer és accedir de forma seqüencial a totes aquestes pàgines per obtenir tots els identificadors *room_id* de les habitacions. La iteració de tota la seqüència de pàgines es realitzarà per a tots els barris de Barcelona. Aquest identificador s'emmagatzemarà dins de la base de dades PostgreSQL i servirà després per construir cada enllaç de cada habitació que permetrà realitzar la petició.

El procediment d'obtenció de cada *room_id* s'ha realitzat cercant dins del codi font HTML de cada pàgina aquella peça de codi que indica quines habitacions en concret es mostren. Un cop localitzat, Google Chrome proporciona una eina molt bona per obtenir el Xpath d'aquesta porció de codi. S'emprarà llavors aquest Xpath amb els mètodes de la llibreria *lxml* per obtenir cada un dels identificadors. S'ha d'anotar que totes les pàgines mencionades anteriorment segueixen una estructura semblant, llavors el mateix Xpath servirà per totes elles.

3.1.1.3 Iterar, obtenir i processar dades

Un cop obtinguts tots els identificadors de les habitacions, el següent pas és iterar sobre cada un d'ells accedint a la base de dades per obtenir cada identificador, construir la pàgina web de l'habitació en qüestió per poder realitzar la petició i llavors tenir via lliure per obtenir el codi font i obtenir les dades.

Aquest pas és el més important i el que tarda més en executar-se, ja que s'obté totes les característiques de cada habitació que després s'empraran per realitzar l'estudi més endavant. Les característiques a obtenir de cada habitació són:

<i>host_id</i>	És l'identificador de l'amfitrió de l'habitació
<i>room_type</i>	És el tipus d'habitació que s'ofereix, pot esser la casa completa (Entire room/apt), una habitació privada (Private room) o bé una habitació compartida (Shared room).
<i>neighbourhood</i>	És el barri d'on es troba l'habitació.
<i>overall_satisfaction</i>	És la mitja de les puntuacions que han donat els comentaris a la seva estància a l'habitació.
<i>accommodates</i>	Són el nombre d'habitacions que ofereix l'amfitrió dins l'habitacle.
<i>bedrooms / bathrooms</i>	El nombre de llits i banys disponibles.
<i>price</i>	El preu de l'habitacle (en euros).
<i>minstay</i>	L'estància mínima a reservar (en dies) de qualsevol convidat dins de l'habitacle.
<i>images</i>	S'obtenen els enllaços dels recursos que mostren les imatges de l'habitacle. Es guarda dins un diccionari les imatges enumerades juntament amb el seu enllaç i també amb un atribut que diu si la imatge en qüestió ha estat feta per un professional de Airbnb.
<i>property type</i>	És el tipus d'habitacle, pot esser un apartament, hotel, casa al camp, vila, etc.

<i>reviews count</i>	És el nombre total de comentaris que té l'habitatge.
<i>neighbourhood overview</i>	És una descripció del barri en el que es troba l'habitatge proporcionada per l'amfitrió. Hi ha amfitrions que no proporcionen aquest tipus d'informació llavors hi haurà mostres dins la base de dades amb aquest camp en NULL
<i>transit</i>	És una descripció de les opcions de transport públic que es poden apreciar als voltants de l'habitatge: si hi ha línies de metro o bus properes, etc. Com abans, hi ha amfitrions que no proporcionen aquest tipus d'informació llavors hi haurà mostres dins la base de dades amb aquest camp en NULL.
<i>host is superhost</i>	Mostra si l'amfitrió és un “super usuari”. Quan es té aquest tipus de rol dins Airbnb vol dir que has pagat una quota mensual i que gaudeixes d'uns serveis que els usuaris “normals” no tenen.
<i>amenities</i>	Són el nombre d'habitacions que ofereix l'amfitrió a part de l'habitació en qüestió.
<i>cancellation policy</i>	És la política de cancel·lació de reserva que ofereix l'habitació. Pot ser de tres tipus: moderada, flexible o estricta.
<i>latitude i longitude</i>	Coordenades de la localització de l'habitació.
<i>cleaning fee</i>	És la taxa que s'ha de pagar per la neteja del pis (si escau).
<i>security deposit</i>	És el depòsit o fiança que s'ha d'ingressar juntament amb el preu de l'estància a l'habitació.
<i>weekly discount / monthly discount</i>	Són els descomptes que s'inclouen al preu final si l'estància és de més d'una setmana o d'un mes.

Les característiques presentades s'han escollit racionalment tinguent en compte significació que podrien tenir dins un estudi estadístic i en la construcció del model predictiu. És possible que algunes de les característiques obtingudes presentin informació redundant, però degut al cost d'implementar i executar el procediment de s'ha d'aprofitar la màxima extracció de dades amb una sola tirada. És més fàcil eliminar les característiques que no són necessàries que tornar a realitzar el *scraping*.

S'ha d'anotar que el procés d'extracció ha donat lloc a múltiples errors desconeguts en un primer moment, com per exemple un canvi dins l'estructura del codi font de la pàgina pot provocar que el Xpath d'algunes de les característiques no funcioni i

llavors és necessari tornar a cercar la porció de codi on es troba la informació desitjada i obtenir el seu Xpath de nou. Altres problemes han pogut venir d'errors de la xarxa (problemes de connexió a Internet o errors 403 els quals indiquen que la petició realitzada ha retornat un valor NULL). Aquests últims s'han ocasionat degut principalment a les proxies escollides, ja que algunes no funcionaven correctament.

En la Figura 7 es proporciona una porció del codi del *scraper* on s'empra la llibreria *lxml* (línia 780) per obtenir la informació específica relacionada amb les imatges de l'allotjament mitjançant la introducció de la seva corresponent direcció Xpath. Després es processa aquesta informació per convertir-la en un diccionari en format *json*, per després obtenir el contingut dels atributs “listing” i “photos”. Seguidament s'obté cada enllaç de cada imatge i es guarda en un fitxer extern. Per poder obtenir després la referència de cada enllaç amb la seva corresponent imatge i habitació, es guarda en la base de dades una enumeració d'aquests enllaços (variable *IMAGES_CURRENT_INDEX*).

```

778 ▼    def __get_photos(self, tree):
779 ▼        try:
780            temp = tree.xpath("//meta[@id='_bootstrap-listing']/@content")
781            if len(temp) > 0:
782
783                s = str(temp[0])
784                photos = json.loads(s)["listing"]["photos"]
785
786                dict_photos={}
787                i = 0
788                delete=?aki_policy=large"
789                global IMAGES_CURRENT_INDEX
790                #IMAGES_CURRENT_INDEX = int(FILE_IMG_INDEX.read())
791                print(str(IMAGES_CURRENT_INDEX))
792                for photo in photos:
793
794                    photo_src = photo['large'][:len(photo['large'])-len(delete)]
795                    FILE_IMG_SRC.write(str(IMAGES_CURRENT_INDEX)+:+photo_src+\n')
796
797                    dict_photos[i] = (IMAGES_CURRENT_INDEX,photo['is_professional'])
798
799                    IMAGES_CURRENT_INDEX += 1
800                    i += 1
801
802                    self.photos = json.dumps(dict_photos)
803
804                else:
805                    print("XPATH FOR PHOTOS DOES NOT WORK. length(temp)=0")
806
807            except:
808                print("Error from getphotos")
809                self.photos = None

```

Fig 7. Funció del *scraper* que obté els enllaços dels recursos de les imatges de cada allotjament mitjançant l'ús del Xpath i llibreria *lxml*.

Com que qualsevol tipus d'error no esperat dins del programa podria aturar l'execució del script i llavors perdre el recompte d'habitacions accedides, s'ha creat una segona base de dades on s'emmagatzemen totes les dades de l'execució del programa: quines habitacions s'han visitat, de quines s'han obtingut les dades correctament i quines no, quins barris s'han visitat i quins queden per visitar, etc. De manera que cada cop que s'aturi i s'engegui el programa, l'execució tornarà al lloc on s'havia parat l'últim cop. El codi representatiu d'aquesta idea ha estat

implementada principalment per Tom Slee [23] però modificat durant el transcurs del desenvolupament d'aquest projecte.

3.1.1.4 Actualitzar Base de Dades

Després de tot el procediment d'extracció de dades de cada habitació, és necessari afegir totes aquestes dades obtingudes dins de la base de dades PostgreSQL i després actualitzar-la. Les comandes SQL d'inserció, extracció, actualització i comunicació amb la base de dades s'han realitzat mitjançant la llibreria *psycopg2*. La funció que insereix la informació obtinguda és la mostrada en la Figura 8.

```

406    def __insert(self):
407        """ Insert a room into the database. Raise an error if it fails """
408        try:
409            conn = connect()
410            cur = conn.cursor()
411            sql = """
412                insert into room (
413                    room_id, host_id, room_type, neighborhood, price, deleted, latitude, longitude, survey_id, images, minstay, bedrooms,
414                    bathrooms, accommodates, overall_satisfaction, reviews_count, neighborhood_overview, transit, host_is_superhost,
415                    property_type, amenities, cancellation_policy, cleaning_fee, sec_deposit, w_discount, m_discount
416                )
417                """
418            sql += """
419                values (%s, %s, %s)"""
420            insert_args = (
421                self.room_id, self.host_id, self.room_type, self.neighborhood, self.price, self.deleted, self.latitude, self.longitude,
422                self.survey_id, self.photos, self.minstay, self.bedrooms, self.bathrooms, self.accommodates, self.overall_satisfaction,
423                self.reviews_count, self.neighborhood_overview, self.transit, self.host_is_superhost, self.property_type, self.amenities,
424                self.cancellation_policy, self.cleaning_fee, self.sec_deposit, self.w_discount, self.m_discount)
425            # ADD PHOTOS HERE
426            cur.execute(sql, insert_args)
427            cur.close()
428            conn.commit()
429            logger.info("Room " + str(self.room_id) + ": inserted")
430        except psycopg2.IntegrityError:
431            # logger.info("Room " + str(self.room_id) + ": insert failed")
432            conn.rollback()
433            cur.close()
434            raise
435        except:
436            conn.rollback()
437            raise

```

Fig 8. Funció que insereix les dades “scrapejades” dins de la base de dades PostgreSQL.

3.1.1.5 Inferències del procés

La implementació específica i detallada que requereix el procediment de *web scraping*, l'alt cost computacional de l'execució de totes les iteracions d'extracció de dades en forma seqüencial, i a més del model conceptual tan complexa que proporciona entendre els nivells de implementació del *scraping* (d'entendre les peticions web fins entendre l'estructuració dels Xpath) i la continua exposició i dependència d'errors desconeguts i factors externs (funcionament dels proxies) ha fet que aquesta part del projecte sigui la més llarga i costosa a portar a terme.

Si es pogués requerir d'un historial de canvis de l'estructura del codi font de la pàgina web de Airbnb i d'un recurs de direccions *proxy* estables (VPN, servidors proxies de pagament) segurament la implementació del codi seria més robusta i flexible de cara a aquests canvis interns dins de l'estructura de la web. També una implementació del bucle d'iteracions de les habitacions en forma paral·lela, es a dir, implementant

threads i aprofitant al màxim les capacitats de la CPU del meu ordinador hagués millorat l'eficiència i rapidesa de l'extracció de les dades.

3.1.2 Obtenció de les imatges dels allotjaments

Tenint en compte les limitacions a nivell computacional de les operacions a disc (llegir i escriure fitxers sobre directoris) i l'alt nombre d'imatges a descarregar (casi 250.000), s'ha decidit construir una estructura jeràrquica de directoris per emmagatzemar les imatges dependent del seu identificador d'habitació (*room_id*). Aquesta estructura es pot veure resumida en la Figura 9.

Llavors s'ha implementat un programa sobre Python 2.7 que té la funció de crear aquesta estructura de directoris. Com es pot veure en la descripció de la Figura 7 el nombre total de directoris és de 20 milions, això no és un problema des de que cada cop que es guarda una imatge al ser descarregada en un directori s'obté la ruta absoluta del directori en la que s'ha emmagatzemat i s'escriu en un fitxer a part. Llavors cada execució d'accendir a un directori mitjançant la seva ruta absoluta té una complexitat trivial ($O(1)$). L'únic problema que pot ocasionar és traslladar la base de dades a un altre disc, ja que comprimir els 20 milions de directoris més les casi 250.000 imatges és una operació molt costosa que pot trigar més de 6 hores en un disc dur normal HD (tarda aproximadament 5 hores en comprimir-se tot en un disc dur sòlid SSD). El fitxer comprimit pesa uns 4,98GB a disc.

El procés de descàrrega de els imatges ha sigut, juntament amb el *web scraping*, el procés més lent del projecte. El temps de descàrrega ha estat un total de tres setmanes amb una mitja de 8 hores de descàrrega per dia.

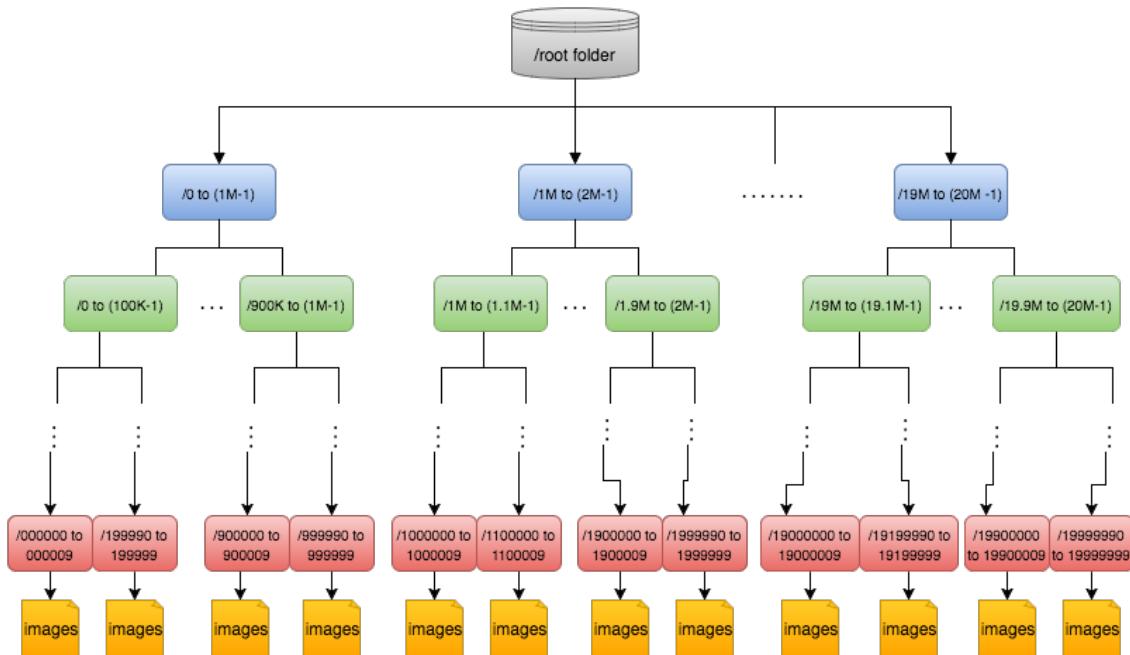


Fig 9. Estructura jeràrquica de fitxers per emmagatzemar les imatges. El nombre del directori correspon al room_id del que prové la imatge descarregada. Els directoris de color blau corresponen als identificadors en magnitud de milions, anant del 0, 1, 2, ... 20 milions. A mesura que es baixa per l'arbre la magnitud disminueix una desena, de manera que cada directori conté 10 sub-directoris. Llavors si hi ha 20 directoris principals (color blau) i després 10 sub-directoris dins cada directori per cada nivell, i 6 nivells de profunditat, el nombre total de directoris creats és $20 \cdot 10^6 = 20.000.000$.

3.2 Neteja i preparació de les dades

Al finalitzar el procés de *web scraping* i descàrrega de les imatges es té les següents bases de dades:

- Una base de dades PostgreSQL amb 13,517 observacions i 28 columnes.
- 229,541 imatges descarregades.

La base de dades PostgreSQL s'ha exportat en un format *csv* (*comma separated value*) i s'ha processat en un fitxer Notebook IPython dins de l'entorn de programació Jupyter [24]. Per obtenir les dades en una estructura de dades manejable en Python, s'ha emprat primer la llibreria *json* [25] per obtenir les dades en forma de diccionari i poder tractar-les més còmodament, seguit de les llibreries *numpy* [26] i *pandas* [27].

El primer pas que s'ha realitzat és obtenir les dades mitjançant la llibreria *csv* [28], la qual permet llegir un fitxer en format *csv* per convertir-lo en un diccionari Python.

Després s'ha iterat per cada observació de la base de dades, es a dir, iterar per cada habitació i les seves característiques, per conservar només les columnes que ens interessen per a l'estudi. Les columnes que s'han eliminat són:

neighbourhood	S'ha eliminat perquè degut a un error en el <i>web scraping</i> s'ha obtingut una informació que no correspon als barris reals dels allotjaments.
deleted / last_modified	Degut a que és una columna temporal que ajudava a l'execució del programa de <i>scraping</i> .
survey_id	Eliminat perquè és una clau a una taula secundària de la base de dades i llavors no té sentit conservar-la.
images	No interessa conservar aquesta informació per als models.
reviews_count	S'ha eliminat perquè presentava un nombre molt gran de valors <i>Nan</i> (cel·la sense valor).
neighbourhood overview / transit	eliminat també ja que no es pot emprar cap informació útil d'elles de cara al model estadístic. Com a molt es podria haver desenvolupat un model de processament de text per extreure conclusions del seu contingut, com per exemple si l'allotjament està prop del transport públic, quins punts turístics hi ha al voltant, etc. Però com que és una informació que proporciona l'amfitrió del habitatge, doncs cap la possibilitat de que hagi mentit i proporcioni una informació falsa.
host_listings_count	Eliminat degut al seu alt contingut de valors <i>Nan</i> .
cleaning_fee / sec_deposit / w_discount / m_discount	Eliminades degut al seu alt percentatge de valors <i>Nan</i> i que també és una informació poc fiable i significant segons els usuaris de Airbnb.

Algunes dades són interessants per a l'estudi, però necessiten ser obtingudes d'una altre manera. Aquestes són el cas de les *amenities* (o facilitats que ofereix un

habitatge), *neighbourhood* i *host_listings_count* (el nombre d'allotjaments totals que ofereix l'amfitrió).

Per el cas de les dades *amenities* tenim la informació en forma d'un string en format *json* en la columna *amenities* de la base de dades. El que s'ha fet és obtenir la informació parsejant aquest string mitjançant els mètodes de la llibreria *json* i obtenir les *amenities* o facilitats en forma de llista, de manera que cada observació ofereix les facilitats que apareixen en aquesta llista. El problema aquí és que hi ha un total de 38 possibles valors, un nombre massa gran per afegir cada un dels valors com a columna de la base de dades. El que s'ha fet és realitzar una selecció de les millors facilitats que pot oferir un habitatge segons l'experiència dels usuaris de Airbnb. Lògicament, *shampoo*, *indoor fireplace* i *Iron* entre altres no són facilitats que ajuden a decidir entre llogar o no un habitatge. Les *amenities* més importants i les que s'han afegir a la base de dades com a *dummy variables* (columnes de valor binari 0 o 1, representant si l'allotjament té la facilitat (1) o no (0)) són:

<i>Kitchen</i>	<i>Internet</i>	<i>Essentials</i>
<i>Heating</i>	<i>Air Conditioning</i>	<i>Breakfast</i>
<i>Pets Allowed</i>	<i>Smoking Allowed</i>	<i>Elevator in Building</i>
<i>24-Hour Check-in</i>	<i>Washer</i>	<i>TV</i>
<i>Lock on Bedroom Door</i>		

En el cas d'assignar el barri on pertany cada allotjament, s'ha fet ús d'una base de dades *geojson* [29] on representa cada un dels barris de Barcelona en forma de polígon o estructura tancada de coordenades (es pot descarregar i visualitzar en [30]). El procediment ha estat iterar per cada coordenada d'un allotjament (en latitud i longitud) i assignar el nom del polígon (o nom del barri) on la coordenada resideix. El resultat és una columna extra en la base de dades amb el nom de *neighbourhood* indicant el nom del barri on es situa l'allotjament. S'ha obtingut un total de 69 barris.

Per saber el nombre d'habitacions que té cada amfitrió, s'ha realitzat una consulta SQL sobre la base de dades PostgreSQL agrupant tots els identificadors dels amfitrions (columna *host_id*) més el compte total de cops que apareix cada identificador a la base de dades. Per exemple, si un identificador d'amfitrió X apareix tres cops (es a dir, en tres files) se li assigna un valor de *host_listings_count* de 3. El resultat final s'obté iterant sobre tots els habitatges i assignar el corresponent comptador total d'habitacions segons l'identificador d'amfitrió de l'habitació tractat.

Després de fer la selecció de les columnes, s'ha realitzat una neteja de les observacions que presentaven valors estranys (preu 0€, 0 llits, estància mínima de 0

nits, etc) o NaN. En conclusió, la base de dades sense netejar obtinguda directament del procés de *web scraping* tenia 13.157 observacions i 28 columnes, la base de dades final després de la neteja consta d'unes **11482 observacions i 28 columnes**. Per evitar ambigüïtats, a partir d'ara anomenarem a aquesta base de dades **AirbnbDB**.

3.3 Exploració de les dades

Aquesta secció es centra en explorar exhaustivament AirbnbDB per trobar dades curioses, descriptives i informatives sobre els allotjaments de Airbnb amb l'ajuda de recursos externs (articles informatius de la situació de Airbnb dins Barcelona, dades estadístiques generals, etc). La majoria de la informació s'extraurà directament de la base de dades emprant típiques tècniques estadístiques, juntament amb visualitzacions gràfiques de les dades (a partir de la llibreria *matplotlib* [31]) i visualitzacions geogràfiques generades amb el recurs online CartoDB [32]

L'estudi estadístic principal ha estat inspirat per els autors de l'estudi *InsideAirbnb* [2], un projecte sense ànims de lucre que ofereix un conjunt d'eines i dades amb l'objectiu d'inspirar aquelles persones interessades en explorar com Airbnb està realment usat en les diferents ciutats del món.

3.3.1 Què explorar?

L'estructura principal del desenvolupament del model descriptiu és semblant al que es fa en *The Occupancy Model* de *InsideAirbnb*. Les preguntes que ens fem en aquesta secció són:

- Com Airbnb està sent realment emprat per els seus usuaris i com està afectant als barris de Barcelona?
- Quina informació podem obtenir dels diferents tipus d'allotjament que ofereixen els amfitrions (casa sencera, habitació privada o habitació compartida)? Quins habitatges tenen més probabilitats a ser fraudulents?
- Quins són els allotjaments amb més disponibilitat? Quins factors juguen en decidir aquest valor?
- Creant una estimació de l'activitat de cada habitatge tenint en compte el percentatge d'ocupació que té, la balança entre les nits disponibles i les nits que ha tingut clients, i el preu per nit, ens podem preguntar el següent:
 - Com es comparen els ingressos obtinguts emprant Airbnb amb els ingressos obtinguts dels propietaris quan ofereixen habitatges de lloguer de llarg termini?

- Pot Airbnb competir contra les agències tributàries de lloguer com hotels, apartaments, i *beds and breakfasts*? Qui sortiria guanyant?
- Fa el nombre de nits reservades per any impossible per un amfitrió emprar els seus habitatges oferts com a casa residencial (com un pis de lloguer de llarg termini)?
- Quina conseqüència té per a la economia de Barcelona i dels seus barris emprar els habitatges solament com a allotjaments turístics de lloguer? Influeix en els dades estadístiques? Quines avantatges i desavantatges presenta?
- Introduint els termes *recent* i *freqüent* a un allotjament, quines conclusions podem extreure?

Mitjançant el processament de la base de dades AirbnbDB i les eines que proporciona Python, juntament amb visualització de gràfics amb *matplotlib* i creació de mapes amb CartoDB, s'intentarà respondre a totes les preguntes que s'han formulat en el paràgraf anterior. Per a més detalls vegeu la secció de Resultats. Per ajudar a contestar les preguntes formulades s'ha desenvolupat un model d'ocupació estadístic, també inspirat per *InsideAirbnb*.

3.3.2 El model d'ocupació de Airbnb

Aquest model combina les dades AirbnbDB més una formulació de preguntes de caràcter estadístic per obtenir una sèrie d'estimacions amb l'objectiu d'enriquir l'estudi dels allotjaments turístics de Barcelona amb estadístics interessants i útils.

Aquest model consisteix en calcular estimacions que pugui respondre als temes: disponibilitat dels allotjaments; la seva activitat en Airbnb; i els ingressos que tenen els seus usuaris.

3.3.2.1 Disponibilitat

Disponibilitat cada mes i any de tots els habitatges a partir de d'un calendari proporcionat per l'equip de *InsideAirbnb* [2].

Aquest calendari ve en forma d'arxiu *csv*, el qual es llegeix mitjançant la llibreria *pandas* i la seva funció *.from_csv()*. Proporciona les dates de les nits disponibles d'uns aproximadament 14,000 allotjaments de Airbnb. Aquesta base de dades s'ha obtingut al gener del 2016, s'ha de tenir en compte llavors que el procés de *web scraping* per obtenir les dades útils d'aquest projecte s'ha realitzat el mes de març del 2016. Durant aquests dos mesos pogueren haver hagut canvis en les dades de Airbnb i llavors el nombre d'habitacions igual que les seves característiques poden ser diferents. Per això s'ha de realitzar una fusió del calendari amb els habitatges de la

base de dades AirbnbDB de manera que només es conservin aquells habitatges que tenen informació present al calendari. Els habitatges restants els eliminarem de l'estudi.

Aquesta operació de fusió, a part de filtrar els allotjaments que només estan presents dins de la base de dades del calendari, genera 12 columnes, una per cada mes de l'any, on s'emmagatzema els dies disponibles del mes durant el 2016. A partir d'això després es crea una columna anomenada *Availability* on calcula el percentatge de disponibilitat que ofereix un determinat habitatge a partir de la suma dels dies disponibles més els dies totals de l'any, com es mostra en la fórmula (3.1). Un cop la fusió s'ha realitzat, AirbnbDB ha quedat modificada de manera que ara té **8240** observacions, un 28,23% menys que abans.

$$\text{Availability} (\%) = \frac{\text{total dies disponibles any}}{365} \times 100 \quad (3.1)$$

3.3.2.2 Activitat

InsideAirbnb també proporciona una altre base de dades anomenada *reviews* on emmagatzema les dates de tots els comentaris publicats per cada allotjament des del 2009. Si ajuntem la informació que es pot extreure d'aquestes dades amb la disponibilitat calculada en la secció anterior podem obtenir una estimació de l'activitat i el percentatge d'ocupació d'un habitatge. Anoteu que aquí també hi haurà d'haver una fusió dels habitatges que apareixen en *reviews* amb els de AirbnbDB, llavors és possible que el nombre d'observacions variï després del procés.

Per evitar costos computacionals extres, només s'ha conservat per cada mes la mitja del sumatori dels comentaris publicats des del 2009 fins el 2016 per aquell mes en concret. La suma total d'aquestes mitjanes és el nombre final de comentaris que ha rebut l'habitacle. El nombre d'observacions que li queda a AirbnbDB després d'haver-la ajuntat amb la *reviews* és de 6493.

A partir d'aquest nombre de comentaris, es pot estimar el nombre de reserves que obtindran els habitatges per any. Si tenim en compte el raonament que ha proporcionat *InsideAirbnb* alhora de calcular l'estimació de reserves [2], veurem que un 50% del total de comentaris és una bona mesura d'estimació on es té en compte crear una mínima credibilitat sobre el càlcul tenint en compte les dades que es tenen. Hi ha estudis que empren un 72%, però és una estimació massa amable i llavors perd credibilitat. Altres afirmen que un 30,5% és una sòlida estimació on podria tenir menys error i llavors més impacte. Però *InsideAirbnb* es va donar compte que el 30,5% no tenia en compte el nombre d'habitacles eliminats abans de fer la recollida de les dades. Un percentatge del 50% compensa la balança entre les dues hipòtesis i llavors serà el valor emprat. La calculació ve donada en la fórmula (3.2).

$$Est. Reserves = \left(\sum_{m=1}^{12} reviews_m \right) * 0.5 \quad (3.2)$$

Si tenim una estimació del nombre de reserves que obtindria un allotjament, llavors podem obtenir una estimació del període de estància mínima (en nombre de nits) dels clients que reservin en un cert habitatge. Aquí però l'únic problema és que no hi ha recursos que indiquen la mitja de nits en les que els usuaris es queden en els allotjaments de Airbnb en Barcelona. Així com a San Francisco sí que es té aquesta estimació [5] (5,5 nombre de nits de mitja), en el cas de Barcelona es prendrà un valor escollit baix raonament de sentit comú de 3 nits de mitja. Per al cas dels habitatges on l'estància mínima és de més de 3 nits, es substituirà per el seu valor d'estància mínima.

Després es pot calcular una estimació del total de nits en les que els habitatges tindrien reserva multiplicant l'estimació del nombre de reserves que obtindrien per any per l'estància mínima en nits (3.3), anomenarem aquest valor *nits ocupades*. Si ara dividim aquest nombre per el nombre total de dies disponibles que es té per any, obtindriem el percentatge d'ocupació de cada habitatge (3.4). Això es tradueix com el percentatge d'estimació dels dies disponibles on l'habitatge tindrà una reserva. Anomenem aquesta estimació *occupancy rate*.

$$nits\ ocupades = Est. reserves * estància\ min \quad (3.3)$$

$$occupancy\ rate\ (%) = \frac{nits\ ocupades}{\# total\ dies\ disponibles\ any} * 100 \quad (3.4)$$

S'ha comprovat que existeixen casos on el percentatge és major a 100, això és degut al pobre càlcul de l'estimació de reserves per any, on hi ha cops en que un habitatge tindrà més nits amb reserva de les que realment té disponibles. En aquests casos se'ls hi ha assignat un valor de *occupancy rate* del 100%.

3.3.2.2 Ingrés

Amb les dades produïdes podem calcular una estimació dels ingressos que obté cada habitatge. Si tenim el nombre estimat de reserves que obtindrà un allotjament per any, el nombre mínim de nits que cada client haurà d'estar al reservar i el preu de cada nit, podem estimar els ingressos (en euros) que s'obtindria per any, i llavors també els ingressos que s'obtindrien cada mes. Cal tenir en compte però un cert marge d'error per augmentar credibilitat en les estimacions, això és degut a que

l'estimació del nombre de reserves té un error. Aquest error s'ha atenuat afegint un filtre del 50% al nombre estimat de reserves que obtindria un habitatge. D'aquesta manera ens assegurem un major marge i més credibilitat en el càlcul. La fórmula segueix la senzilla estructura mostrada en la fórmula (3.5).

Abans però de realitzar els càlculs, de la mateixa manera que s'ha afegit un filtre per incrementar la credibilitat de la hipòtesi de l'estimació de les reserves per any (3.2) també s'ha d'aplicar un filtre per els preus dels habitatges. Això és degut a la gran quantitat de *outliers* que té la columna *price* (vegeu la gràfica esquerra de la Figura 10). Els habitatges que tenen aquest preu tan alt són casos excepcionals que no són significatius dins el càlcul de l'estimació de l'ingrés. Aquests *outliers* llavors no expressen cap mena d'informació útil en aquest cas concret. Eliminant-los s'obtindrà una assumpció molt més sòlida de l'ingrés de cada habitatge. L'eliminació es pot fer fàcilment estandarditzant el valor dels preus i conservar a la base de dades només aquelles observacions que estiguin dins del rang [-3, +3] desviacions estàndard σ . Realitzant aquesta neteja ens queden unes dades molt més simètriques i representatives, tal com es mostra en la gràfica de la dreta de la Figura 10.

$$\frac{\text{ingrés}}{\text{any}} (\text{€}) = (\text{est. reserves} * 0.5) * \text{preu} * \text{estancia min} \quad (3.5)$$

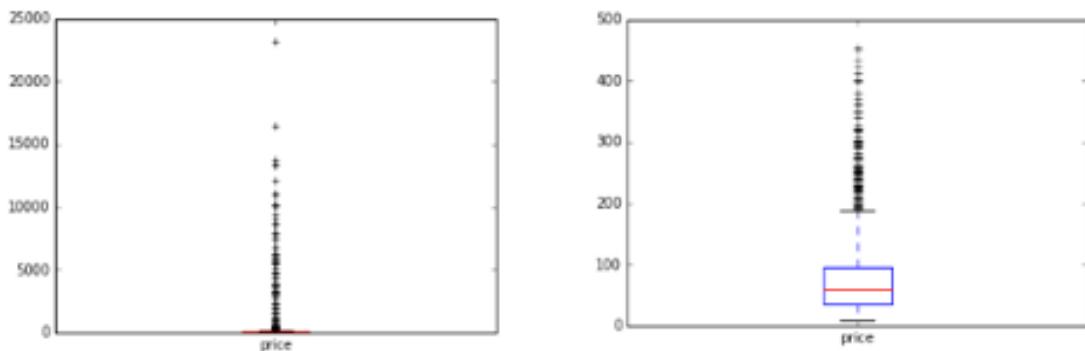


Fig 10. Representació del boxplot de la columna *price*. A l'esquerra, es mostren els valors dels preus abans de realitzar la neteja dels outliers. A la dreta, després de la neteja.

En la secció de Resultats s'exposaran els valors obtinguts aplicant el model d'ocupació presentat.

3.4 Desenvolupament del model predictiu

En aquest apartat es descriu com s'han aplicat els models predictius explicats en la secció de Metodologia (concretament secció 2.3) sobre la base de dades AirbnbDB mitjançant l'ús de la llibreria *scikit-learn* [34]. Tot el procés s'ha portat a terme sobre un arxiu Notebook de IPython sobre l'entorn de programació Jupyter.

3.4.1 Preparació de les dades

Primer de tot, cal saber que realitzar un estudi estadístic descriptiu té una sèrie de diferències a aplicar un model predictiu lineal en termes d'estructura de dades. En els models predictius s'ha d'eliminar qualsevol tipus d'informació que tingui finalitats visuals de la base de dades. Com per exemple en el cas de AirbnbDB: l'identificador de l'amfitrió de l'habitacle (*host_id*); el nombre de comentaris publicats durant cada mes (només conservarem el nombre total d'aquests comentaris); també s'eliminaran tots els valors estimats en el Model d'Ocupació de Airbnb ja que són estimacions i llavors la falta de credibilitat absoluta en l'obtenció de les dades pot desestabilitzar el model i empitjorar el seu funcionament. En aquest cas també eliminarem les coordenades latitud i longitud ja que no presenten cap mena d'informació significativa en la creació d'un model predictiu lineal. Com única referència geogràfica tindrem el barri de cada habitatge. Llavors les columnes que ens queden en la base de dades són:

<i>24h_checkin</i>	<i>Air_conditioning</i>	<i>Bedroom_lock</i>	<i>Breakfast</i>
<i>Essentials</i>	<i>Heating</i>	<i>Internet</i>	<i>Kitchen</i>
<i>Elevator</i>	<i>Pets_allowed</i>	<i>Smoking_allowed</i>	<i>TV</i>
<i>Washer</i>	<i>Acommodates</i>	<i>Bathrooms</i>	<i>Bedrooms</i>
<i>Price</i>	<i>Property_type</i>	<i>Rating</i>	<i>Room_type</i>
<i>Superhost</i>	<i>Neighbourhood</i>	<i>Total_rcounts</i>	<i>Cancel_pol</i>
<i>L_count</i>	<i>minstay</i>		

Encara hi ha un problema, que les variables *property_type*, *room_type* i *neighbourhood* són variables categòriques. Es sap que els models predictius lineals tendeixen a funcionar millor si les variables categòriques de les dades es presenten en format numèric binari, es a dir, amb dos possibles valors 0 o 1, en lloc de tenir variables categòriques amb més de 2 possibles classes. Existeix el mètode que s'ha mencionat abans de convertir una columna categòrica en una variable *dummy*. Una

columna categòrica es defineix com una columna amb un valor (tant sigui caràcter com numèric) que discrimina l'observació dins una classe c_i d'un conjunt $C=\{c_1, c_2, \dots, c_n\}$, sent n un nombre enter finit. Llavors la conversió seria crear n columnes extres per cada variable categòrica, on cada columna correspondria a una categoria c del conjunt C . Les columnes que s'han convertit en variables *dummy* han estat les mostrades en la taula següent, on n representa el nombre de variables o columnes dummy que s'han hagut d'aplicar per cada predictor categòric:

<i>Room Type</i>	<i>Property Type</i>	<i>Cancelation Policy</i>	<i>Neighbourhood</i>
$n=3$	$n=9$	$n=3$	$n=69$

En conclusió, després de la neteja i preparació de les dades tenim una base de dades AirbnbDB amb 6369 observacions i 106 predictors.

3.4.2 Aplicació dels models lineals

Com ja s'ha dit en la secció de Metodologia, es farà ús dels següents regressors lineals (secció 2.3.2):

- Regressió Lineal
- Regressió Ridge
- Regressió Lasso
- Elastic Nets
- Bayesian Ridge

S'empren regressors lineals perquè són ideals per tractar una variable de resposta escalar o numèrica. En el cas d'aquest projecte la variable de resposta on realitzarem l'estudi de predictibilitat és el preu dels habitatges. Com ja s'ha dit, el procediment s'efectuarà sobre codi Python en un Notebook de Jupyter. La llibreria que empraren és la *scikit-learn* degut a que ofereix un àmplia aplicabilitat de models predictius (casi tots els coneguts estan disponibles) més un gran abast de sistemes d'avaluació com *cross-validation*, *bootstrap*, etc.

Abans d'aplicar els models s'han de plantejar una sèrie de coses. En el primer cas, i més important, una mètrica d'avaluació del nostre model. El mètode més típic per avaluar una regressió és mitjançant el *residual sum of squares* (RSS), on es pot traduir com la mitja de les distàncies de les observacions estimades a les reals al quadrat (fórmula (2.1) de la secció 2.3.1). El resultat d'aplicar RSS serà l'error o la desviació en € que el model s'allunya al valor real. El problema aquí és la gran quantitat de *outliers* que resideixen en les dades. Malgrat la neteja dels *outliers* que

s'ha fet en la secció 3.3.2.3 (vegeu Figura 10), la mitja del preu ens surt 69 € però encara allotjaments amb preu de més de 3000€. Això pot ser un problema ja que el model pot tenir “massa” en compte els *outliers* que realment tindrien poca significació al patró que segueix el preu dels allotjaments turístics. Podem pensar per exemple que hi hagi aproximadament 50 allotjaments luxosos a cada barri on el seu preu és molt diferent als de la mitja del barri en qüestió. Llavors aquí introduïm una altre mètrica més interessant on no es té en compte els *outliers* de les dades: la *median squared error* (MeSE). Un exemple del seu funcionament sobre unes dades externes simulades es pot veure representat en la Figura 11.

Després el següent pas és comprovar si existeix correlació entre les variables predictores escalars, on pot ocasionar perdre capacitat de predicció al nostre model. Aquest efecte és conegut com *multicollinearity* [14]. S'ha representat en gràfics els valors de les variables predictores numèriques més la resposta en la Figura 12. Com es pot apreciar, no existeix cap problema de *multicollinearity* en les nostres dades al no haver-hi cap combinació de predictors amb alt índex de correlació.

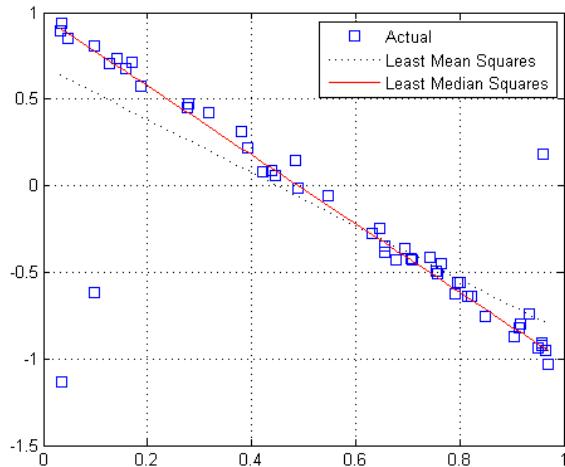


Fig 11. Funcionament de la minimització del least mean squares contra el least median squares en el cas de dades simulades amb outliers.

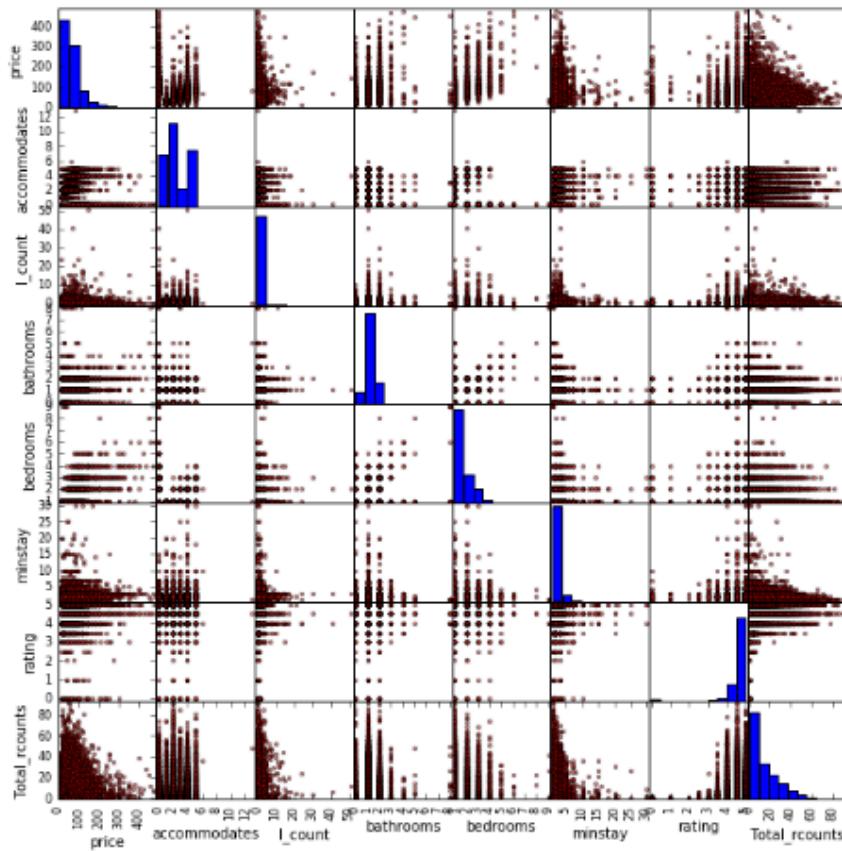


Fig 12. Representació gràfica de totes les possibles combinacions de variables predictores numèriques juntament amb la resposta preu. A la diagonal de la gràfica es representa l'histograma dels valors de cada variable.

Ara és quan es pot començar a aplicar els models. Les funcions emprades de la llibreria *scikit-learn* han estat [15]

- `linear_model.LinearRegression()`
- `linear_model.Ridge()`
- `linear_model.Lasso()`
- `linear_model.ElasticNet()`
- `linear_model.BayesianRidge()`

Les dades han estat dividides en dos conjunts, un anomenat conjunt d'entrenament o *training set* i l'altre anomenat conjunt de test o *test set*. El propòsit d'aquesta partició és de crear un conjunt pel qual el model s'entrenarà i generarà els coeficients que s'ajustaran a les dades (*training set*) per després provar el model construït sobre unes dades de validació (*test set*). Per dividir-les s'ha emprat el mètode disponible en la llibreria *scikit-learn* `train_test_split` on retorna 4 conjunts de dades:

- `X_train` conté el conjunt d'entrenament sense la variable de resposta preu de la partició d'entrenament.

- `y_train` conté el conjunt de test, es a dir, només la columna preu amb les mateixes observacions que `X_train`.
- `X_test` conté el conjunt d'entrenament sense la variable de resposta de la partició de test.
- `y_test` conté el conjunt de test amb les mateixes observacions que el conjunt `X_test`.

Després s'entrenen els 5 regressors lineals mitjançant els sub-conjunts `X_train` i `y_train` per després ser testejats amb el conjunt `y_test`. Els resultats s'exposaran en la secció de Resultats.

3.4.3 Optimització dels resultats

Degut a que aplicar directament els regressors lineals sense realitzar una optimització de les dades abans és molt poc comú en el món real ja que no es poden esperar resultats decents realment aprofitables. Llavors es portaran a terme una sèrie de mètodes que asseguren trobar una millor solució al problema.

3.4.3.1 Mètodes Ensemble

Aplicar un regressor *Ensemble*, concretament el Gradient Boosting Regressor (GBR), ja que combinen múltiples regressors simples o dèbils entrenats amb una estratègia intel·ligentment diferent sobre la resposta per obtenir uns resultats amb menor variància. En la secció 2.3.3 de Metodologia s'expliquen en detall els mètodes Ensemble juntament amb el GBR.

Per construir el model es farà ús del mètode `GradientBoostingRegressor()` de la llibreria scikit-learn [35]. Cal anotar que aquest mètode deixa assignar opcionalment una sèrie de paràmetres de configuració de la construcció del model. Escollir bé aquests paràmetres pot esser la diferència entre obtenir resultats bons i resultats molt bons.

3.4.3.2 Cerca dels millors paràmetres

Realitzar una cerca dels millors paràmetres d'afinació o *tuning parameters* a assignar al mètode Ensemble per assegurar que obtindrem els millors resultats. Aquesta cerca s'anomena *Grid Search* i obté els resultats dels models entrenats amb cada possible combinació dels paràmetres *tuning* assignats al mètode de cerca, retornant la informació del model amb la millor combinació d'aquests paràmetres.

L'avantatge d'aquest mètode és que, gràcies a la llibreria *scikit-learn*, ofereix la possibilitat de visualitzar l'evolució de l'error a mesura que s'anaven cercant diferents combinacions de paràmetres. La desavantatge però és que per cercar els millors paràmetres només es pot realitzar testejant exhaustivament cada possible valor, ja que no hi ha una mètrica general de quins paràmetres usar als problemes. Llavors la complexitat dels càculs augmenta exponencialment a mesura que s'augmenten les possibles combinacions dels paràmetres. Per testejar cada combinació es fa mitjançant *cross-validation*. El mètode facilitat per la llibreria *scikit-learn* `GridSearchCV()` realitza exactament el mètode que apuntem [36].

3.4.3.2.1 Apache Spark

Degut principalment al gran cost computacional que requereix realitzar el *grid search* dels millors paràmetres, s'ha hagut d'estudiar la possibilitat d'emprar algun mètode d'optimització per reduir el temps d'execució. Entre molts, existeix l'entorn d'optimització Apache Spark, el qual ens pot proporcionar un clúster virtual on les computacions dels programes executades en ell es realitzen 100 cops més ràpid en memòria que emprant l'altre famosa tècnica d'optimització computacional MapReduce de Hadoop [7]. I 10 cops més ràpid en disc. Apache Spark s'executa sobre una arquitectura que empra grafs acíclics dirigits, la qual suporta fluxos de dades cícliques i computació *in-memory*. Apache Spark es coneix com un dels recursos d'optimització computacional que garanteix millors resultats en termes de cost computacional i rapidesa d'execució sobre disc i dins memòria (CPU).

El problema però és que per aprofitar la velocitat de computació que ofereix Spark s'ha d'utilitzar de manera distribuïda conjuntament amb més d'una computadora a través d'un clúster virtual. D'aquesta manera s'aprofiten tots els processadors dels diferents ordinadors combinats del clúster per disminuir notablement el cost computacional de les operacions. El problema és que aquest projecte no s'ha dissenyat per emprar Spark en aquest escenari, sinó que simplement s'ha instanciat un clúster on emprarà tots els processadors de l'ordinador que s'ha generat el codi font amb l'objectiu d'il·lustrar l'ús de l'entorn Apache Spark.

Per realitzar el *grid search* mitjançant Apache Spark s'ha d'inicialitzar l'entorn Jupyter amb la variable `pyspark` amb disponibilitat de tots els processadors hàbils de l'ordinador. Al iniciar Jupyter, es veu que s'ha instanciat automàticament un clúster virtual de Spark (context de Spark) configurable mitjançant una variable global `sc`. En la documentació de [7] s'explica com crear i realitzar operacions dins un clúster de Spark. Després s'importa la llibreria `spark_skearn`, on proporciona la funció `GridSearchCV` [37] que funciona de la mateixa manera que la funció `GridSearchCV` de `sklearn`. Executar llavors la funció de *grid search* per un rang

prou gran de paràmetres sobre un clúster virtual és ara possible gràcies a aquest recurs, on abans era computacionalment inviable sobre *raw CPU*.

3.4.3.3 Selecció de variables

Realitzar una selecció de les variables més significatives també és un pas important dins la millora de la predictibilitat d'un model. Quan en un model se li afegeixen variables de més que no tenen cap relació directe amb la resposta, aquestes augmenten la complexitat del model i llavors el *bias* augmenta, resultant en uns models més difícils d'entendre, menys precisos i més probables a crear *overfitting*. A la secció 2.6 de Metodologia s'explica el mètode de selecció del subconjunt de k variables que presenten menys variància sobre la resposta.

Es farà ús del mètode `SelectKBest()` de la llibreria `sklearn` [38] per obtenir una transformació de les dades passades per paràmetre. On la base de dades resultant contindrà els k millors predictors.

3.5 Microneighbourhoods

Per a la cerca dels *microneighbourhoods* de Barcelona s'ha construït un arbre de decisió de regressió o *Decision Tree Regressor* (DT). De manera que la idea és emprar les regions de classificació que construeix el model com a limitadors de cada *microneighbourhood*. Per a obtenir-los es presenten els següents reptes:

- Per a la creació del model només faran falta les coordenades de cada habitatge, llavors només es conserven les columnes latitud i longitud. És correcte llavors conservar el *mean squared error* (MSE) com a mètrica de *splitting* del DT?
- Un cop construït el model, com s'obtenen les coordenades que defineixen cada polígon del *microneighbourhood*?

Per al primer apartat, en una primera instància pot semblar que les variables latitud i longitud poden ser valors molt poc significants alhora de predir un preu en un regressor lineal. On la típica mètrica MSE per avaluar un *split* pot no donar un resultat òptim. Llavors una solució més intuïtiva pot esser calcular les regions de classificació a partir de la “distància” (en aquest cas la distància és la diferència de preus entre dos punts geogràfics). De manera que cada regió tingui aquells habitatges on hi ha un preu semblant. Però realment el que passa és que el MSE ja té en compte aquesta “distància” entre punts geogràfics alhora de realitzar els *splits*. Si recordem que el MSE és la diferència entre el valor estimat i la mitja dels valors

en els que resideixen en la regió classificatòria. Llavors té sentit conservar aquesta mètrica.

3.5.1 Extracció dels polígons

Per l'extracció dels *microneighbourhoods* intuitivament s'ha d'implementar un algorisme on per cada fulla de l'arbre resultant o regió classificatòria, s'ha d'extreure les 4 coordenades necessàries per dibuixar el polígon que definirà els límits del *microneighbourhood*. El codi que obté les coordenades de cada regió classificatòria es pot veure en la Figura 13.

```

1 def getTreeBoxes(tree, feature_names):
2
3     leaves = tree.tree_.children_left == -1
4     leaves = np.arange(0,dt.tree_.node_count)[leaves]      # Get the leaves nodes
5     values = tree.tree_.value                                # Get all nodes' values
6     decision_paths = getDecisionPaths(tree)                 # Get decision paths of all leaves
7     tree_boxes = []                                         # Final boxes (polygons) yielded by decision tree
8
9
10    min_lat = min(dat.latitude)                            # A polygon follows the following structure:
11    min_lon = min(dat.longitude)                          # poly = Polygon( ((0, 0), (0, 1), (1, 1), (0, 0)) )
12    max_lat = max(dat.latitude)
13    max_lon = max(dat.longitude)
14
15    for (ind,i) in enumerate(leaves):
16        lat_up, lat_down, lon_left, lon_right = max_lat, min_lat, min_lon, max_lon
17
18        path = decision_paths[ind][1]                      # decision path of the i-th leaf
19        value = round(float(values[i]),2)                  # price predicted in the i-th leaf
20
21        for idx in range(len(path)-1, -1, -1):           # going backwards on the list means going from firsts
22            thresholds to final ones
23
24            if feature_names[path[idx][0]] == 'longitude':   # went left
25                if path[idx][1] == -1:
26                    lon_right = path[idx][2]
27                else:
28                    lon_left = path[idx][2]
29
30            else:                                           # went right
31                if path[idx][1] == -1:
32                    lat_up = path[idx][2]                      # went left (predicted region downwards)
33                else:
34                    lat_down = path[idx][2]                     # went right (upwards)
35
36
37    # Polygon's construction
38    b = ((lon_left,lat_up),(lon_left,lat_down),(lon_right,lat_down),(lon_right,lat_up),(lon_left,lat_up))
39    tree_boxes.append([b,value])
40
41 return tree_boxes

```

Fig 13. Codi que extreu els microneighbourhoods d'un Decision Tree passat per paràmetre.

La funció `getTreeBoxes(tree, feature_names)` extreu les coordenades necessàries per construir els polígons que representaran cada *microneighbourhood*. Els paràmetre `tree` és l'objecte `DecisionTreeRegressor` de la llibreria `scikit-learn` ja entrenat, i `feature_names` és una llista on conté el noms els predictors emprats al DT com a Strings, és a dir, “longitud” i “latitud”.

Les línies 3 i 4 obtenen els índexs dels nodes terminals de l'arbre emprant l'objecte *tree* de *scikit-learn* (no hi ha documentació sobre aquest objecte en el repositori de scikit-learn) i els emmagatzemen en una llista *numpy*. Després es recullen els valors de tots els nodes de l'arbre (valor -1 per els nodes no terminals) i després s'obtenen tots els camins de decisió de l'arbre de cada node terminal mitjançant la funció *getDecisionPaths* representada en la Figura 14. Aquesta funció obté els camins de cada node terminal amb el format següent:

$$\text{decision_paths}[indexLeaves] = (\text{leaf}, \text{path})$$

```

25 | def getDecisionPaths(tree):
26 |     # get the nodes which are leaves
27 |     leaves = tree.tree_.children_left == -1
28 |     leaves = np.arange(0,dt.tree_.node_count)[leaves]
29 |
30 |     # build a simpler tree as a nested list: [split feature, split threshold, left node, right node]
31 |     thistree = [dt.tree_.feature.tolist()]
32 |     thistree.append(dt.tree_.threshold.tolist())
33 |     thistree.append(dt.tree_.children_left.tolist())
34 |     thistree.append(dt.tree_.children_right.tolist())
35 |
36 |     # The returning list will contain tuples in which will be represented each leaf node with its decision_path
37 |     # example: print(decision_paths[0]) # (4,
38 |     #                                (1, -1, 41.40018081665039),
39 |     #                                (0, 1, 2.109476089477539),
40 |     #                                (0, -1, 2.2099030017852783))
41 |
42 |     decision_paths = []
43 |     for (ind,nod) in enumerate(leaves):
44 |         # get the decision rules in numeric list form
45 |         rules = []
46 |         RevTraverseTree(thistree, nod, rules)
47 |         #print("----> Rules of leaf node", nod, "are:", rules)
48 |         decision_paths.append((nod,rules))
49 |
50 |     return decision_paths

```

Fig 14. Codi de la funció *getDecisionPaths*.

On *leaf* és l'índex del node terminal tractat i *path* és el camí que s'ha seguit per arribar fins a ell. El valor *path* és una llista de tuples on els valors de la llista estan ordenats de manera que el primer element és una tupla que indica la informació de l'últim *split* que ha realitzat el DT per arribar al valor del node terminal; mentre que l'últim element és el primer *split* que s'ha realitzat en direcció cap al node terminal (normalment sol esser el node pare). Cada tupla tindrà els valors “(*índex predictor emprat* (0 indica longitud i 1 latitud), *direcció seguida esquerra* (-1) o *dreta* (1), *valor del threshold del split*)”. Aquesta estructura té el nom de *rules* dins el codi i s'obté mitjançant la funció recursiva *RevTraverseTree* mostrada en la Figura 15.

```

1 def RevTraverseTree(tree, node, rules):
2     """
3         Traverse an skl decision tree from a node (presumably a leaf node)
4         up to the top, building the decision rules. The rules should be
5         input as an empty list, which will be modified in place. The result
6         is a nested list of tuples: (feature, direction (left=-1), threshold)
7         The "tree" is a nested list of simplified tree attributes:
8         [split feature, split threshold, left node, right node]
9         """
10        # now find the node as either a left or right child of something
11        # first try to find it as a left node
12        try:
13            prevnode = tree[2].index(node) #tree[2] means child_left
14            leftright = -1
15        except ValueError:
16            # failed, so find it as a right node
17            prevnode = tree[3].index(node) #tree[3] means child_right
18            leftright = 1
19        # now let's get the rule that caused prevnode to -> node
20        rules.append((tree[0][prevnode], leftright, tree[1][prevnode]))
21        # if we've not yet reached the top, go up the tree one more step
22        if prevnode != 0:
23            RevTraverseTree(tree, prevnode, rules)
24

```

Fig 15. Codi de la funció recursiva *RevTraverseTree*.

La funció *RevTraverseTree* pren l'arbre de decisió, el node actual i la llista resultant *rules* com a paràmetres. De manera que cada crida es tracta d'un node en concret i s'actualitza la llista *rules*. La propietat recursiva de la funció ofereix recórrer l'arbre de baix cap a dalt, baixant en profunditat. El seu funcionament és: primer es comprova si el node en qüestió és un fill esquer o dret d'algún node pare, on s'assigna el valor de *leftright* dependent de quin tipus de fill sigui; després s'ha d'obtenir la regla o *rule* que ha causat la transició del node pare al node fill que s'ha tractat, això és indicar quin predictor s'ha emprat, indicar si és fill esquer o dret, i indicar el valor *threshold* que ha ocasionat la transició o *split* (línia 20 de la Figura 15); i finalment es comprova si s'ha arribat a l'arrel de l'arbre, on en cas contrari s'executa de nou la funció per el node pare del node tractat. El resultat d'executar la funció és la llista *rules* actualitzada amb tots els camins de cada node terminal codificats en el format explícit abans.

Un cop obtinguts tots els camins dels nodes terminals, tornem a la funció principal *getTreeBoxes* (Figura 13). Ara per obtenir els valors de les coordenades del *microneighbourhood* s'han d'assignar cinc punts geogràfics (tuples de longitud i latitud) on el primer punt ha de ser el mateix que l'últim de manera que el polígon es pugui tancar i llavors el codi funcioni. Per això, s'instancien els valors màxims i mínims de longitud i latitud. S'han pres aquests valors ja que si s'empressin podríem representar tota Barcelona sobre un mateix quadre, llavors cada *microneighbourhood* seria un sub-quadre més petit. La idea és llavors iterar per cada node terminal i anar actualitzant els valors de quatre coordenades estratègiques per pintar el polígon *lat_up*, *lat_down*, *lon_left*, *lon_right*. En la Figura 16 es representen els valors de cada combinació de les coordenades descrites fins ara (màxims i mínims, més les coordenades estratègiques).

3.5.2 Exemple d'execució

L'estratègia a seguir és, seguint el camí del DT des del node arrel fins al node terminal (és a dir, anar en sentit contrari al de la llista `decision_paths`), anar actualitzant els valors de coordenades estratègiques comprovant cada *split* de l'arbre. Per exemple, si tenim el camí del node terminal t estructurat de la següent manera

Profunditat	Variable	Direcció	<i>Threshold</i>
2	latitud	esquerra	41.401
1	longitud	dreta	2.109
0 (arrel)	longitud	esquerra	2.209

sabem que t té una profunditat 2+1, i que si disminuïm una unitat la profunditat de t obtenim el seu pare que ha tingut la següent decisió: *if latitud <= 41.401*. Després sabem que aquest node ha vingut d'un pare on ha pres la decisió: *if longitud > 2.109*; i finalment que el pare d'aquest últim node és el node arrel (profunditat 0) el qual s'ha preguntat: *if longitud <= 2.209*. Vegeu que l'operant “ \leq ” indica la decisió que segueix un fill esquer i “ $>$ ” un fill dret.

El flux que seguirà el bucle per actualitzar i obtenir els valors de `lat_up`, `lat_down`, `lon_left`, `lon_right` en el cas de l'exemple que hem introduït és:

Valors inicials pel node t .

- $\text{Lat_up} = \text{max_lat}$
- $\text{Lat_down} = \text{min_lat}$
- $\text{Lon_left} = \text{min_lon}$
- $\text{Lon_right} = \text{max_lon}$
- Valor del microneigh = 39,4€

Camí de decisions del node t , de major a menor profunditat:

Variable	Direcció	<i>Threshold</i>
latitud	esquerra	41.401
longitud	dreta	2.109
longitud	esquerra	2.209

Per cada decisió del camí del node t :

Iteració 1	<i>Decisió</i>	<i>Assignem</i>
	If latitud ≤ 41.401	$\text{lat_up} = 41.401$
Iteració 2	<i>Decisió</i>	<i>Assignem</i>
	If longitud > 2.109	$\text{lon_left} = 2.109$
Iteració 3	<i>Decisió</i>	<i>Assignem</i>
	If longitud ≤ 2.209	$\text{lon_right} = 2.209$

Els resultats finals són:

- lat_up = 41.401
- lat_down = min_lat
- lon_left = 2.109
- lon_right = 2.209

On el *microneighbourhood* resultant vendria donat per el següent polígon:

- (lat_up, lon_left) → (41.401, 2.109)
- (lat_up, lon_right) → (41.401, 2.209)
- (lat_down, lon_left) → (min_lat, 2.109)
- (lat_down, lon_right) → (min_lat, 2.209)
- (lat_up, lon_left) → (41.401, 2.109)
- Valor → 39,4€

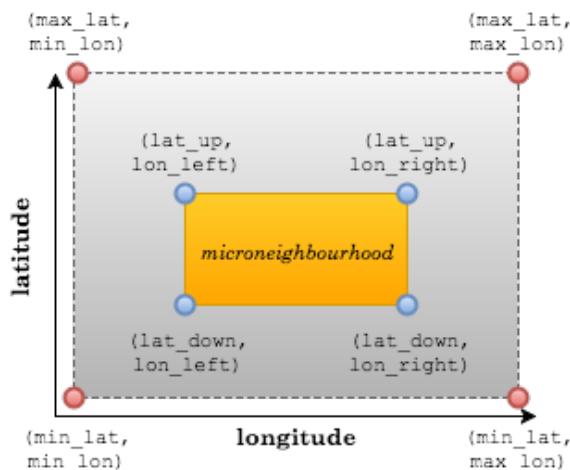


Fig 16. Gràfica que mostra les regions marcades per els punts geogràfics emprats en l'algorisme d'obtenció dels microneighbourhoods. Els punts vermells corresponen als punts màxims i mínims de les coordenades, marcant el quadrant màxim de Barcelona (en gris). Mentre que els punts blaus corresponen als punts amb els valors actualitzats, definint la regió d'un microneighbourhood (en taronja).

Fitxeu-vos que el valor del *microneighbourhood* no és més que el valor que el DT ha assignat a la regió classificatòria tractada. El nombre de polígons final varia segons les característiques del DT. Les característiques més importants són el *min samples leaf* o mínim nombre d'observacions per node terminal, i *max depth* o màxima profunditat de l'arbre. La idea és jugar amb aquests dos valors fins obtenir un conjunt de *microneighbourhoods* decent i que tingui sentit.

Finalment tots els *microneighbourhoods* obtinguts s'han codificat en un arxiu *geojson* en part gràcies a la llibreria *shapely.geometry* [39] que ofereix una interfície que codifica els polígons en format *geojson* de manera fàcil.

3.6 Extracció de les característiques de les imatges

Per a la extracció de les característiques de cada imatge s'ha emprat la xarxa neuronal convolucional Hybrid Places-CNN (explicada en la secció de Metodologia), on cada iteració o *forward* de la xarxa proporciona unes 1183 variables en forma de distribució de probabilitats (es a dir, per cada imatge s'obtenen 1183 característiques amb menor o major probabilitat). Entre les opcions disponibles de llibreries d'aplicació de xarxes neuronals, com TensorFlow de Google [40], Caffe és la que proporciona més velocitat de computació degut a la seva excel·lent arquitectura i també una interfície fàcil d'emprar amb una àmplia documentació [41].

3.6.1 Obtenció del vector de característiques

Per obtenir els vectors de característiques emprant la CNN, el que s'ha fet és reutilitzar el codi *wrapper* proporcionat per Caffe on ja carrega la Hybrid Places-CNN tant en CPU com en GPU. Abans de tot s'han obtingut els directoris absoluts de les imatges descarregades dels habitatges a partir del fitxer *json* que conté tota la informació de les imatges dels allotjaments (concretament el fitxer *metadata-json.txt* contingut en el codi font del projecte). Les 1183 característiques estan dividides en dos subconjunts, 205 categories de escenes referent a la base de dades Places i les 978 categories restants pertanyen al conjunt d'entrenament de ILSVRC2012 ImageNet, una base de dades amb més de 3,6 milions d'imatges [42].

Primer s'importa la llibreria Caffe de Python 2.7 juntament amb la xarxa neuronal (localitzada dins dels repositoris interns de llibreries de Caffe). S'assigna el mode GPU al objecte Caffe degut a que el codi s'executarà sobre un ordinador amb una GPU TESLA NVIDIA K80. Si recordem que tenim aproximadament unes 240,000 imatges i es sap que una iteració de la xarxa neuronal (es a dir, tractar una imatge) triga aproximadament 2 segons sobre el mode CPU, llavors el temps per tractar totes les imatges és d'aproximadament

$$\text{Temps iteració CNN sobre CPU} = 2\text{s} * 240,000\text{iter} = 480,000\text{s} = 5,5 \text{ dies.}$$

Mentre que es sap que sobre GPU la xarxa realitza ~13 iteracions per segon, això significa en una optimització del temps de

Temps iteració CNN sobre GPU = $240,000 / 13 = 4000\text{s} = 1,2$ hores.

Degut a aquesta considerable millora de temps d'execució i ja que disposem del hardware necessari el codi s'ha executat sobre GPU. Després es carrega la xarxa neuronal del repositori de Caffe seguit d'una sèrie de preparacions, entre ells instanciar un transformador de dades que processarà cada imatge d'entrada abans de ser enviada a la xarxa, de manera que així ens assegurem la màxima aplicabilitat i robustesa del model. Les imatges estaran escalades mitjançant el transformador en una forma de 227x227 píxels en color RGB.

Un cop preparat tot això, el següent pas és classificar les imatges i obtenir el vector de probabilitats. S'ha implementat un bucle on a cada iteració una imatge és pre-processada mitjançant el transformador, després es passa com entrada a la xarxa i es realitza un forward. El resultat és el vector de 1183 posicions on cada posició i correspon a la probabilitat de que la imatge pertanyi a la classe C_i . Per obtenir el màxim valor de probabilitat i llavors la classe de la imatge es fa ús de la funció `argmax()` proporcionada per `numpy`.

Un exemple visual d'un vector de característiques seria el que es mostra en la Figura 17. On el valor de les x correspon a la classe C_i i les y corresponen a les probabilitats.

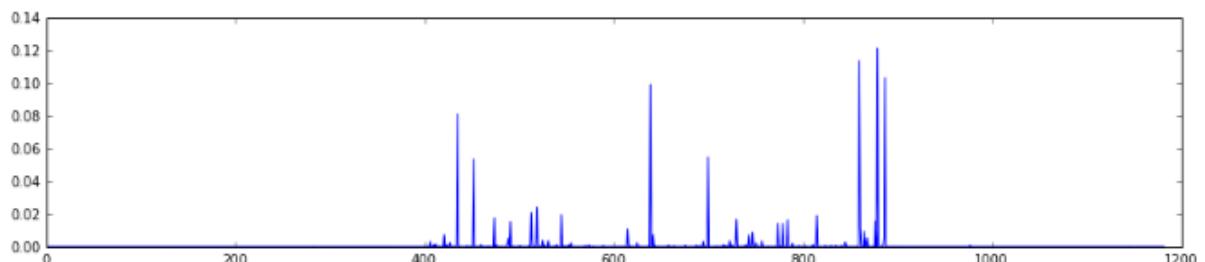


Fig 17. Vector de distribució de probabilitats de la classificació de la xarxa Hybrid Places-CNN sobre una imatge.

El resultat final de l'extracció és una estructura matricial `numpy` de 228,307 línies amb 1183 columnes, la qual es guarda en un fitxer en format `npy`.

3.6.2 Afegir les noves característiques al model predictiu

La matriu de característiques obtinguda en l'apartat anterior s'obté mitjançant la funció `load` de `numpy` per poder ser tractada sobre codi Python. Cada columna de

la matriu representa una de les 1183 possibles categories que processa la xarxa Hybrid Places-CNN, però aquestes estan representades numèricament del 1 al 1183. Llavors el que s'ha fet és cercar per la web un fitxer de referència que especifica el nom de les classes per cada columna, processar i assignar el nom de les corresponents classes.

Seguidament s'ha obtingut un vector de 228,307 posicions on s'especifica la classe assignada segons la distribució de probabilitats. Si recordem que cada allotjament li corresponen una sèrie d'imatges, llavors la idea és combinar totes les característiques obtingudes de les imatges de l'habitatge en qüestió per després ajuntar-les al model. Però el problema és que pot ser que l'allotjament x tingui unes característiques diferents a les d'un allotjament y . Llavors s'han ajuntat totes les característiques de les imatges de tots els habitatges en un únic vector, el qual després de ser processat es veu que té més de 1000 posicions. Aquest nombre de classes és molt gran per afegir-les directament al model predictiu, llavors s'hauria de fer algun tipus de pre-processament abans.

Pot ser hi hagin algunes classes que es puguin ignorar depenent de la seva ocurrència, segons el nombre de vegades que apareix una classe podem dir que aquella classe en concret serà o no important al model. Llavors s'ha trobat correcte filtrar només aquelles característiques que apareixen més de 100 cops i afegir-les al model. El resultat de la selecció és una matriu de 13,109 files (cada allotjament) i 137 columnes (les categories extretes de les imatges). Per afegir aquesta matriu al model s'haurà de fusionar la matriu amb la base de dades que s'ha emprat per a la predicción (6369 allotjaments amb 106 variables) de manera que es conservin només els 6369 allotjaments (fusió de tipus *inner join*).

3.7 Aplicació Docker

Abans d'aventurar-se en la interfície Docker, s'ha de codificar el programa que s'executarà dins l'aplicació. S'ha implementat un programa sobre codi Python 2.7 que empra el model del Decision Tree obtingut en la secció dels microneighbourhoods, de manera cada cop que s'executa el programa necessita dues variables: latitud i longitud, i retorna un preu. Després s'ha implementat una JSON API per comunicar les prediccions mitjançant l'ús d'un port assignat. S'ha emprat la llibreria *flask-restful* per crear una API simple, de manera que cada cop que obtengui un GET retorni una predicción del preu per una localització passada per paràmetre.

Per construir una imatge Docker és necessari també escriure un Dockerfile. Un Dockerfile és un arxiu de text on s'instancien una sèrie de comandes que satisfà la

instal·lació de les dependències, constructor del port on la API escoltarà, i quins programes Python s'han d'executar.

Un cop tot generat, és hora de crear la imatge Docker on anirà implementada la nostra petita aplicació. Per crear-la s'ha executat la següent comanda sobre la terminal Docker:

```
docker build -force-rm=true -t airbnb_app .
```

On el nom que tendrà la imatge és `airbnb_app`, ara per executar l'aplicació simplement s'ha de posar la següent comanda en la terminal per tenir operativa l'aplicació JSON API escoltant en el port `localhost:5000`.

```
docker run -net host -d -name myapp airbnb_app
```

CAPÍTOL 4: RESULTATS

En aquesta secció s'exposaran els resultats obtinguts aplicant les diferents tècniques i metodologies que hem anat introduint i explicant en les seccions de Metodologia i Desenvolupament sobre les dades. L'estructura del contingut d'aquesta secció serà: primer de tot, exposició de resultats de la part exploratòria de les dades juntament amb algunes visualitzacions; representació dels resultats de l'extracció dels *microneighbourhoods* de Barcelona sobre mapes; i finalment els resultats del model predictiu.

4.1 Exploració de les dades

4.1.1 Dades interessants de Barcelona

Segons varis estudis demogràfics, Barcelona és una de les ciutats amb més demanda turística d'Europa amb més de 8 milions de visitants cada any (i en augment) i més de 18 milions de pernoctacions segons dades dels hotels de Barcelona [6]. Concretament, Barcelona resideix en la quarta ciutat més visitada d'Europa.

Amb la rapidíssima evolució de l'activitat web durant els últims 10 anys, amb més i més gent de totes les edats emprant ja el World Wide Web (WWW), les empreses com Airbnb es centren enaprofitar aquesta evolució per proporcionar serveis molt més còmodes i amb molta més aplicabilitat. Airbnb, entre altres, són empreses de caràcter turístic on els seus usuaris ofereixen les seves propietats privades per a que altres usuaris puguin passar una o més nits. Aquests tipus de serveis cada cop creixen més, Airbnb sent el líder per excel·lència ja té a la seva disposició més de 60 milions d'amfitrions emprant la seva web en més de 34,000 ciutats arreu del món [1]. La lluita constant de les grans empreses tributàries (hotels i apartaments de lloguer) contra aquest nou tipus de servei turístic és un entorn molt interessant per aventurar-s'hi i descobrir realment quin és el factor que fa de Airbnb un servei més còmode que els hotels.

4.1.2 Distribució d'allotjaments per barris

Després d'obtenir i netejar les dades del *web scraping*, emprant un fitxer *geojson* que delimitava tots els barris de Barcelona en polígons de coordenades s'ha assignat el barri de cada allotjament. És llavors interessant veure la distribució dels habitatges que hi ha per cada barri per tenir una idea principal de quines zones poden ser més

rentables en termes de població d'allotjaments. En la Figura 18 es veu una representació de la quantitat d'habitacions que hi ha per barri, on Dreta de l'Eixample (casi 450 habitatges), Sant Pere, Santa Caterina i Ribera i el Raval (sobre 400 habitatges els dos) són els barris on hi ha més allotjaments de Airbnb.

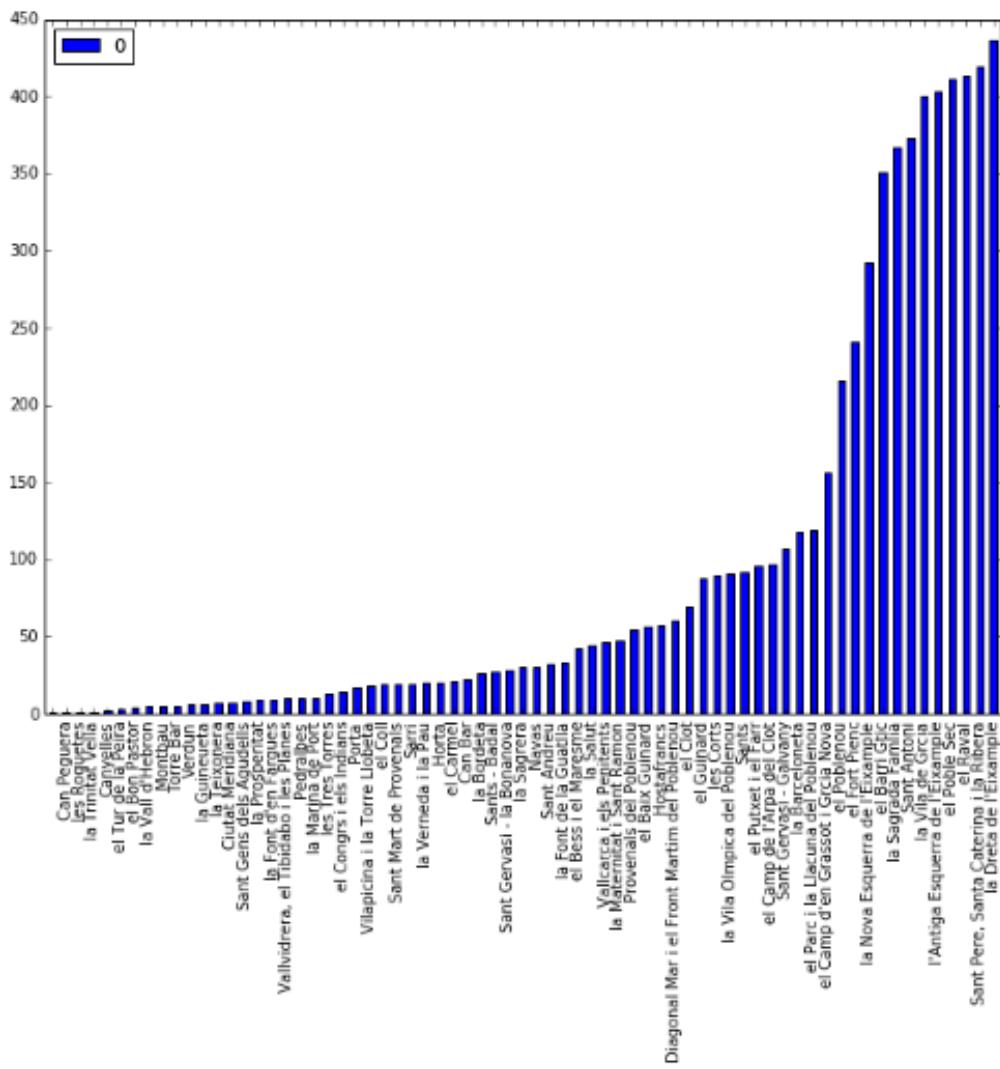


Fig 18. Distribució dels habitatges sobre els barris de Barcelona.

Segons varis estudis geogràfics i turístics de Barcelona [6] els barris més visitats són Eixample, Ciutat Vella i Sants-Montjuïc. En la Figura 19 es veu la densitat dels habitatges en un mapa centrat en els barris que reben més turistes.

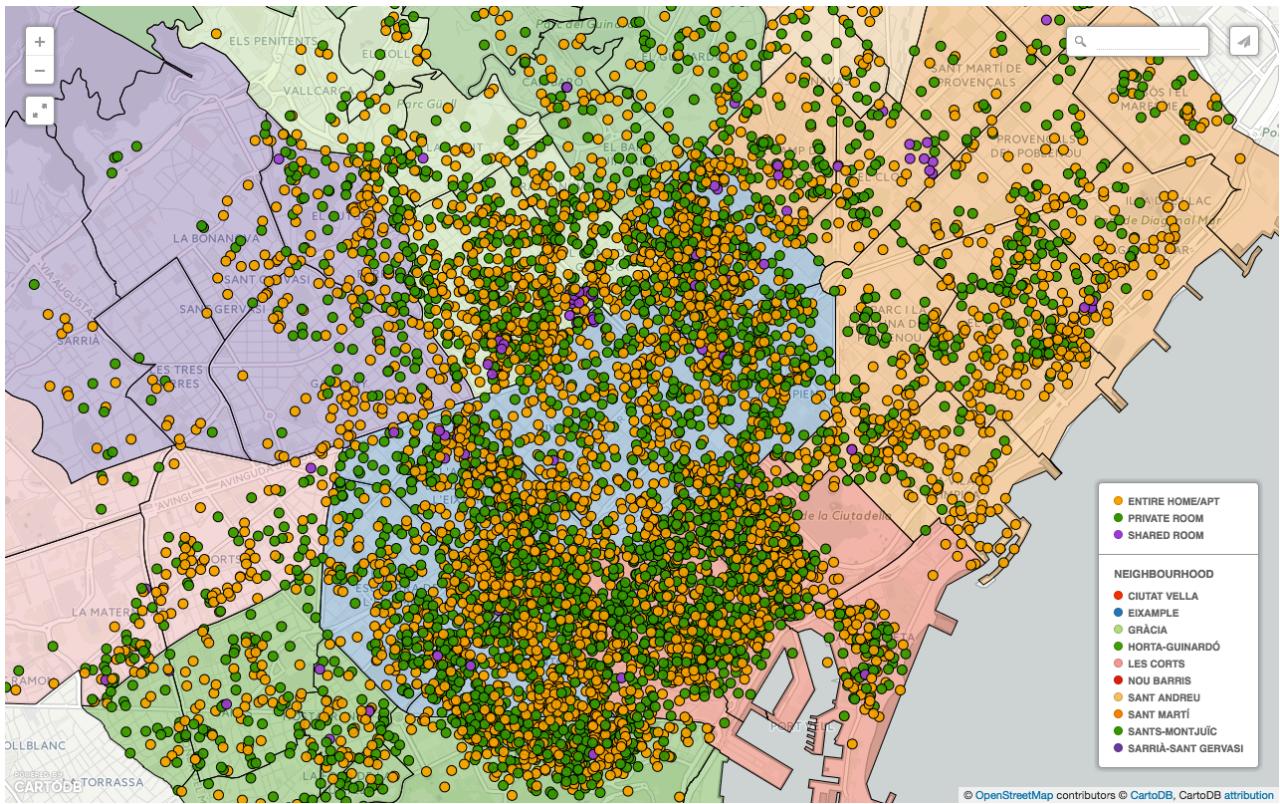


Fig 19. Representació de la densitat d'allotjaments turístics sobre mapa de Barcelona. Els barris on hi ha més densitat d'habitatges Ciutat Vella en color vermell al sud-est de Barcelona; Gràcia en color verd fluix al centre sobre l'Eixample; i Eixample en color blau al centre.

Aparentment vegem que els barris on hi ha més demanda turística són els que hi ha una major densitat d'allotjaments. Llavors podem prendre la hipòtesi que en aquestes zones distribució de característiques dels habitatges presentarà algunes diferències respecte a les dels altres barris.

4.1.3 El model ocupacional de Airbnb

En la secció 3.3.1 de Desenvolupament s'han formulat algunes preguntes interessants de recerca estadística sobre els allotjament turístics. Seguidament, s'ha introduït un model ocupacional de Airbnb inspirat en el projecte de recerca Airbnb de l'equip InsideAirbnb, el qual intenta calcular algunes dades estadístiques interessants a partir de la base de dades per poder contestar a les preguntes formulades. Aquestes són les següents:

- Com Airbnb està sent realment emprat per els seus usuaris i com està afectant als barris de Barcelona?

- Quina informació podem obtenir dels diferents tipus d'allotjament que ofereixen els amfitrions (casa sencera, habitació privada o habitació compartida)? Quins habitatges tenen més probabilitats a ser fraudulents?
- Alguns amfitrions ofereixen més d'un habitatge en Airbnb. Tenen aquests amfitrions més probabilitats a tenir un negoci emprant Airbnb? És probable que aquests amfitrions no visquin a les propietats que ofereixen? O pitjor encara, és probable que no siguin els propietaris?
- Quins són els allotjaments amb més disponibilitat? Quins factors juguen en decidir aquest valor?
- Creant una estimació de l'activitat de cada habitatge tenint en compte el percentatge d'ocupació que té, la balança entre les nits disponibles i les nits que ha tingut clients, i el preu per nit, ens podem preguntar el següent:
 - Com es comparen els ingressos obtinguda emprant Airbnb amb els ingressos d'ofrir un habitatge de lloguer de llarg termini?
 - Pot Airbnb competir contra les agències tributàries de lloguer com hotels, apartaments, i *beds and breakfasts*? Qui sortiria guanyant?
 - Fa el nombre de nits reservades per any impossible per un amfitrió emprar els seus habitatges oferts com a casa residencial (com un pis de lloguer de llarg termini)?
 - Quina conseqüència té per a la economia de Barcelona i dels seus barris emprar els habitatges solament com a allotjaments turístics de lloguer? Influeix en els dades estadístiques? Quines avantatges i desavantatges presenta?
 - Introduint els termes *recent* i *freqüent* a un allotjament, quines conclusions podem extreure?

En un total de 6369 allotjaments de Airbnb útils (es a dir, aptes per aplicar el model ocupacional), un 50,4% són allotjament que ofereixen la casa sencera (3210 habitatges), un 48,5% són habitacions privades (3087) i un casi negligible 1,1% pertany a les habitacions compartides (veure Figura 20).

A partir d'aquests resultats podem concloure que un 98,9% dels habitatges de Airbnb estan oferint un servei semblant al dels hotels (habitacions privades) i als dels apartaments de lloguer (habitacions sencers).

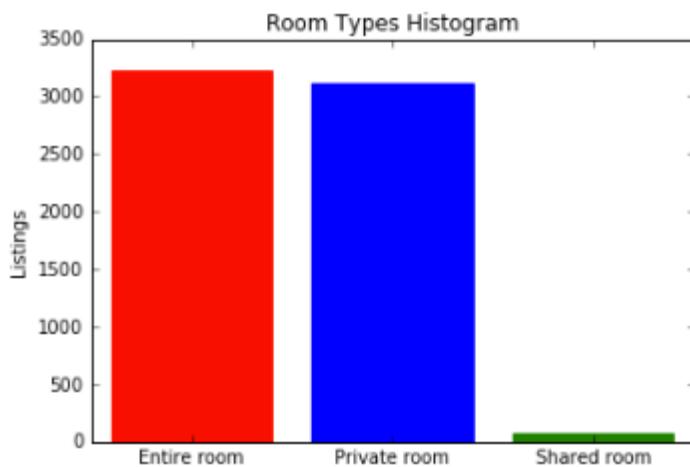


Fig 20. Histograma dels habitatges segons el seu tipus de propietat.

A partir de l'estimació de les reserves que efectuaran els clients de Airbnb a Barcelona i l'estància mínima dels habitatges s'ha estimat que l'estància mitja en nombre de nits dels habitatges per any és 16,5. Es a dir, un habitatge tindrà una ocupació de mitja 16,5 nits per any.

Si realitzem la mitja dels comentaris totals entre els 12 mesos de l'any, obtenim una estimació de 1,3 comentaris per mes per cada habitatge. Això es tradueix com un habitatge s'estima que rebrà 1,3 reserves per mes. Si vegem la distribució dels comentaris publicats per mes, i multipliquem això per la mitja de l'estància mínima podem obtenir una estimació de les nits que es passen en allotjaments de Airbnb per mes. Vegeu la Taula 1 per els resultats.

Nits/mes	Gener	Febrer	Març	Abril	Maig	Juny
Airbnb / Hotels*	1,1 / 5,4	1,0 / 5,5	1,9 / 7,9	2,4 / 8,9	3,2 / 9,6	3,4 / 8,9
	Juliol	Agost	Setembre	Octubre	Novembre	Desembre
Airbnb / Hotels*	3,7 / 10,6	4,0 / 11,2	4,1 / 9,3	4,1 / 9,7	2,4 / 6,8	1,7 / 6,2

*Segons les dades hoteleres recopilades al 2014 [6]

Taula 1. Distribució mensual de la mitja de l'estància en nits

Si tenim en compte el nombre d'habitacions que ofereixen cada una de les dues plataformes, veurem que segons [6] els sector hoteler ofereix aproximadament 35,000 habitacions, mentre que en l'estudi de Airbnb només es tenen en compte 6369 allotjaments. Aquesta diferència de valors és una de les raons principals de cara a la considerable diferència de nits de la Taula 1. Si interpretem les estimacions de

Airbnb com valors amb molt error, llavors aquesta diferència disminueix també considerablement. Imagineu que si tinguéssim un error de 4 nits per mes, hi hauria alguns mesos on l'estància en allotjaments de Airbnb podria competir contra les grans empreses hoteleres i llavors podria resultaria en un negoci assequible.

El preu mitjà dels allotjaments turístics de Airbnb en Barcelona és de 69€ per nit. Aplicant el model ocupacional per calcular l'estimació del percentatge d'ocupació dels allotjaments ens surt que el 11,9% dels allotjaments disponibles tendiran a estar sempre ocupats. Combinant aquestes dades amb l'estimació de l'estància mínima obtenim una ingrés mensual de 94,2€ de mitja per habitatge. Això és una estimació de l'ingrés mensual que un amfitrió tindria emprant Airbnb com a servei.

Si calculem la disponibilitat que ofereixen els habitatges obtenim que el 68,8% dels habitatges ofereixen alta disponibilitat (un total de 4382 allotjaments). Un habitatge té alta disponibilitat si la seva disponibilitat estimada (nombre de dies disponibles a l'any entre el nombre total de dies de l'any, llavors una disponibilitat del 100% significa que està disponible els 365 dies de l'any) és major a la disponibilitat mitjana de tots els habitatges, es a dir, un 78,3%.

Hi ha alguns casos on els amfitrions ofereixen més d'un allotjament, aquests casos són els anomenats *multi-listing hosts*. Segons les dades, hi ha un 16,8% dels allotjaments (exactament 1067) que són d'un amfitrió amb més d'un habitatge. Aquests casos són uns dels més probables en tenir un negoci mitjançant l'ús del servei Airbnb.

Les estadístics que s'han introduït fins ara són dades extretes directament de la base de dades aplicant el model ocupacional. El que és interessant però és saber realment l'ingrés que reben cada un dels amfitrions mitjançant Airbnb. Podria ser l'ús d'aquesta pàgina realment un negoci rentable?

A partir de la fórmula (3.5) del model ocupacional es pot estimar l'ingrés anual i mensual dels allotjaments. En la Taula 2 està representada la distribució dels ingressos mensuals més l'anual, i la Figura 21 es pot veure una gràfica de les dades de la Taula 2.

Mes	Ingressos mensuals
Gener	38,5€
Febrer	34,9€
Març	66,6€
Abril	85,7€
Maig	111,2€
Juny	115,8€
Juliol	123,2€
Agost	137,4€
Setembre	136,2€
Octubre	142,2€
Novembre	80,5€
Desembre	57,5€
Total (ingrés anual)	1129,7€

Taula 2. Ingrés mensual de cada mes de l'any

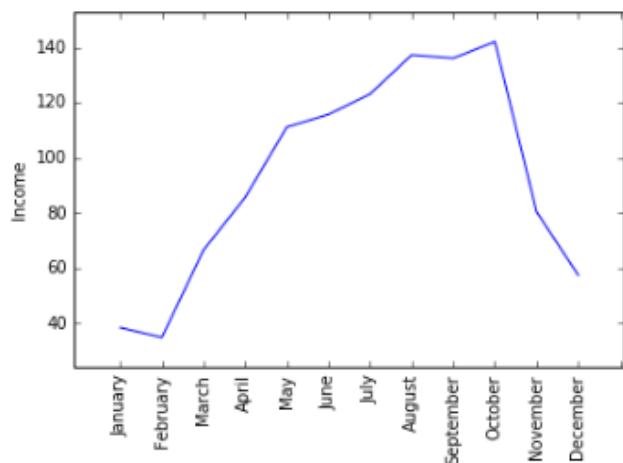


Fig 21. Representació de les dades de la Taula 2.

Vegem que els mesos on es presenta més activitat turística (estiu) és on els ingressos augmenten. Llavors podem veure una relació directe entre el nombre de turistes que visiten Barcelona amb el profit que treuen els allotjaments.

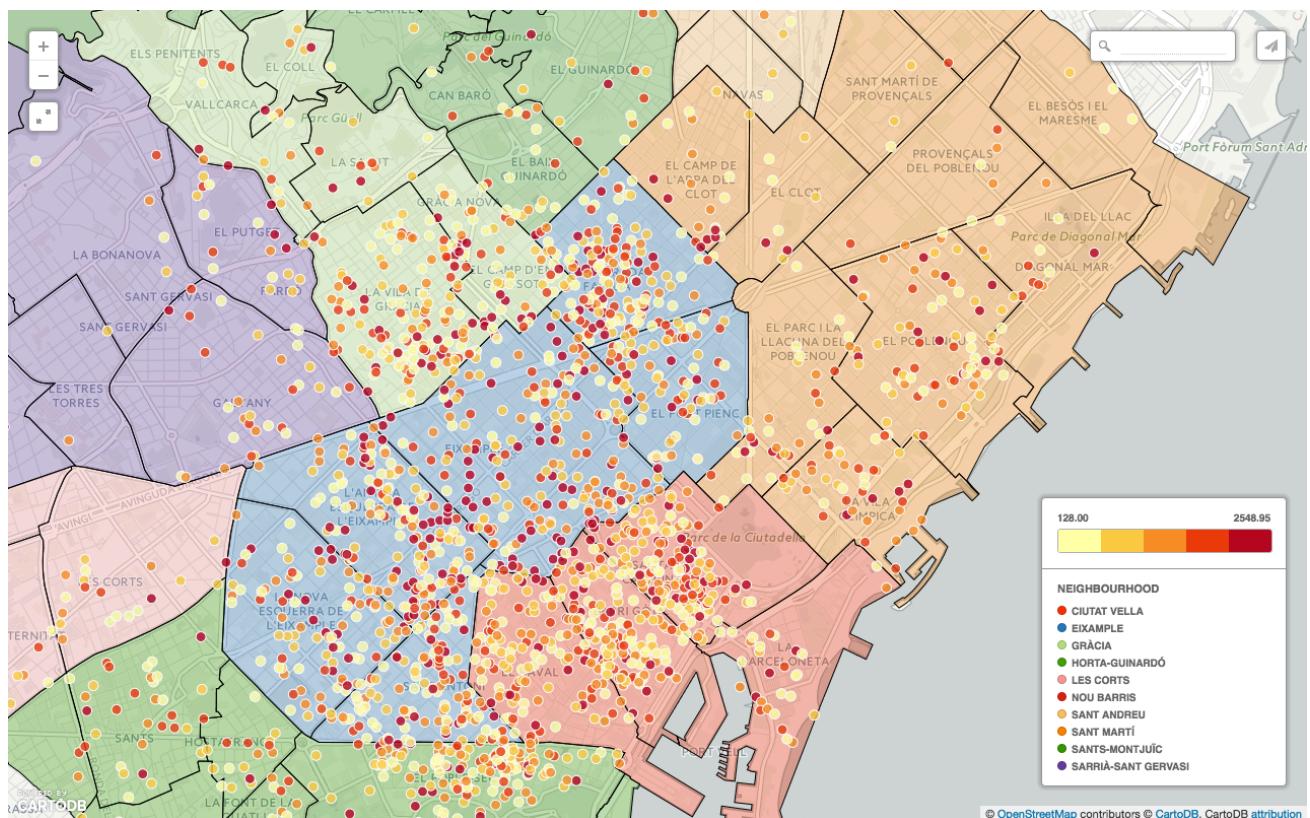


Fig 22. Representació dels allotjaments més rentables segons els seus ingressos mensuals.

Podem també calcular els ingressos que obtindrien alguns habitatges estratègics. Per exemple, si calculem l'ingrés anual per cada habitatge i obtenim aquells en els que el seu ingrés mensual supera a la mitja (94,15€), els podem representar sobre un mapa per veure quins barris presenten un nombre d'allotjaments més rentable (veure Figura 22). Veient el mapa podem veure que la part nord de Ciutat Vella, a més del centre de l'Eixample i Dreta de l'Eixample són els barris on hi ha una densitat de pisos rentables més alta. Aquesta dada concorda amb la hipòtesi descrita abans, la qual afirmava que les zones on la densitat d'allotjaments més alta és més probable que hi hagi una densitat d'allotjaments rentables més alta.

En la Taula 3 es representa la informació dels 5 amfitrions amb més ingressos mensuals de Barcelona. La disponibilitat representa el percentatge mitjà de dies de l'any que ofereix els seus allotjaments (100% indica que estan disponibles els 365 dies de l'any); ocupació representa una estimació del percentatge de dies que els allotjaments tindran reserva. Preu indica el preu mitjà d'entre tots els allotjaments de l'amfitrió. La columna Total hab. indica el nombre d'allotjaments totals que ofereix l'amfitrió en Barcelona. Barri indica el barri on l'amfitrió té més allotjaments. Ingrés/m i Ingrés/y és l'estimació dels ingressos mensuals i anuals que haurien de tenir els amfitrions.

Top	Disponibilitat	Ocupació	Preu	Total hab.	Barri*	Ingrés/m	Ingrés/y
1	95,6%	20,1%	299,4€	14	AEE	5329,5€	63953,6€
2	85,1%	9,9%	108,2€	20	AEE	4701,3€	56415,5€
3	82%	9,7%	97,1€	8	NEE	4172,5€	50070,3€
4	88,7%	15,8%	114,2€	6	PS	3690,3€	44283,3€
5	87,5%	9,9%	120,5€	21	SF	3384,2€	40609,8€

*AEE=Antiga Esquerra de l'Eixample. NEE=Nova Esquerra de l'Eixample. PS=Poble Sec.

SF=Sagrada Família

Taula 3. Representació dels top5 amfitrions amb més ingressos.

De la Taula 3 es pot concloure que la disponibilitat i ocupació no són factors sòlids alhora de preveure una major rentabilitat. Ni tampoc ho és el nombre d'allotjaments que s'ofereixen. Per una il·lustració d'aquest comportament fixeu-vos en els casos top3 i top5, on el tercer tindrà ocupat un 9,7% del 82% dels dies de l'any de mitja en els seus 8 allotjaments, mentre que el quint té menys ingrés tenint 21 allotjaments però semblant percentatge d'ocupació i, a més, un preu més alt. Això ve degut a la diferència de l'estimació de les reserves que tindran, el top3 té una mitja d'estimació de reserves de 21,3 mentre que el top5 té un 5,6. Si recordem el model ocupacional, l'ingrés (fórmula (3.5)) es calcula a partir de l'estimació del nombre de reserves més el preu i l'estància mínima. Podem veure també que Eixample és el barri on sembla haver-hi més rendibilitat.

OpenDataBCN [43] ofereix una àmplia quantitat de bases de dades de Barcelona, entre elles hi ha els *hotspots* turístics o punts turístics populars de Barcelona. En la Figura 23 es representen els allotjaments més rentables sobre el mapa de Barcelona juntament amb els *hotspots* turístics. Com podem observar, definitivament hi ha una relació entre la rendibilitat dels habitatges si estan localitzats a prop d'algún punt turístic important, ja que la densitat dels pisos rentables és aparentment major als voltants dels punts turístics.

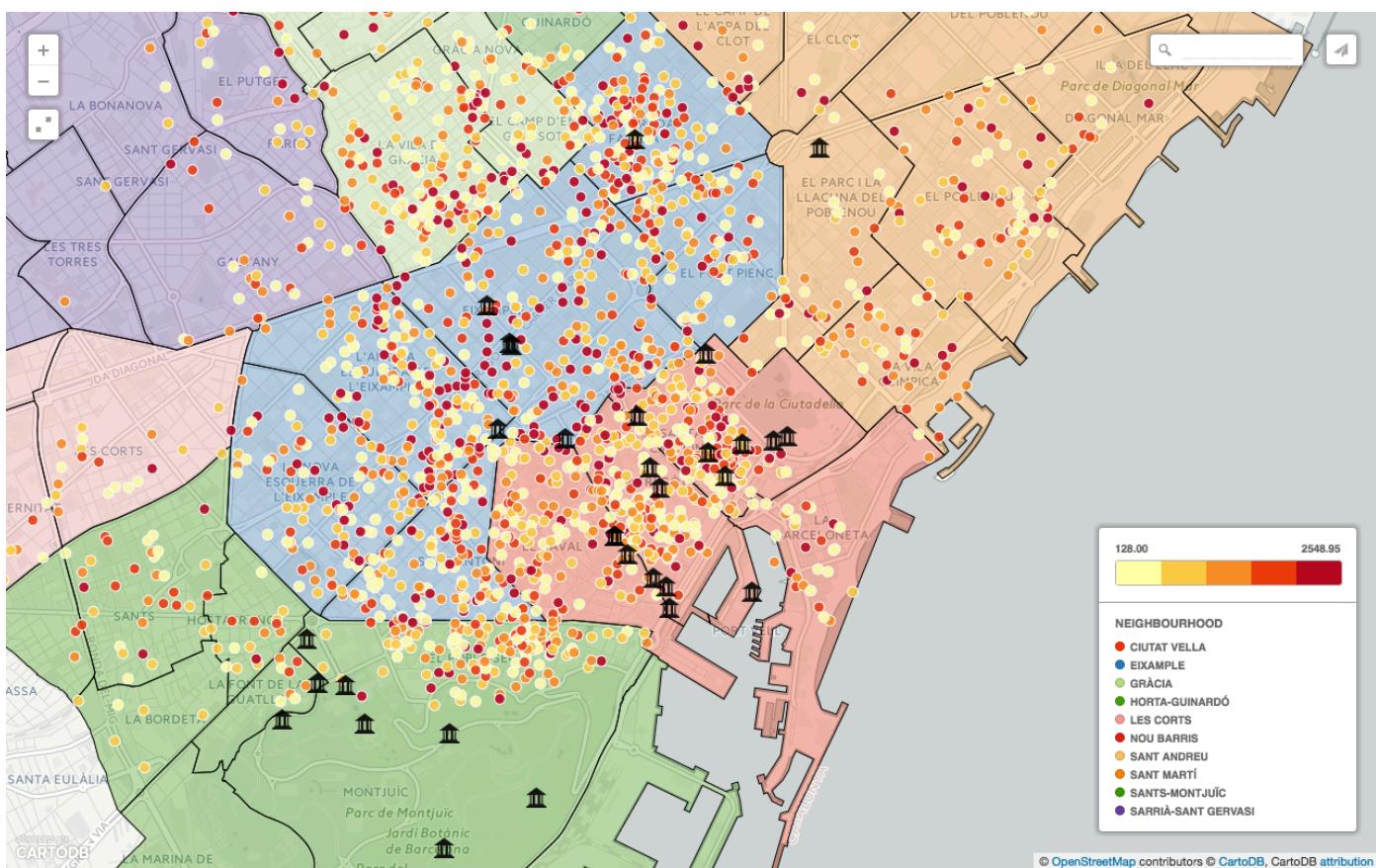


Figura 23. Representació cartogràfica de la localització dels pisos més rentables amb els punts turístics més visitats de Barcelona.

Entre altres dades d'interès, és captivant veure la diferència entre l'ingrés mensual que reben de mitja aquells amfitrions que tenen alta disponibilitat en el seu(s) allotjament(s) contra els que no. Els habitatges amb una disponibilitat més alta a la mitja tenen un ingrés de 92,63€ mensuals, mentre que aquells que no tenen tanta disponibilitat tenen un ingrés de 97,51€ per mes. Això pot ser degut a la hipòtesi de que alguns amfitrions només tenen disponible el seu habitatge aquelles etapes de l'any on es sap que la demanda és major, jugar amb el preu dependent del mercat i de l'oferta i llavors treure més profit que aquells que tenen l'habitatge disponible més del 80% de dies de l'any.

Vegem també la diferència entre els ingressos que tenen aquells amfitrions que tenen més d'un habitatge en Barcelona. Aquells que només tenen un allotjament tenen un ingrés mensual de 89,4€, mentre que aquells que en tenen més d'un obtenen de mitja 304,71€ per mes.

4.2 Microneighbourhoods

Per desenvolupar els *microneighbourhoods* de Barcelona s'han portat a terme les tècniques i metodologies explicades en la secció 2.4 de Metodologia i 3.5 de Desenvolupament. A continuació s'exposaran els resultats obtinguts sobre mapes generats amb CartoDB.

Com a primera visualització dels resultats de l'extracció dels polígons dels microneighbourhoods mitjançant l'aplicació del Decision Tree, s'han assignat uns paràmetres de *min leaf samples* amb valor 100 i *max depth* de 5. L'arbre resultant és mostra en la Figura 24, mentre que la representació dels polígons extrets sobre un mapa es veu en la Figura 25.

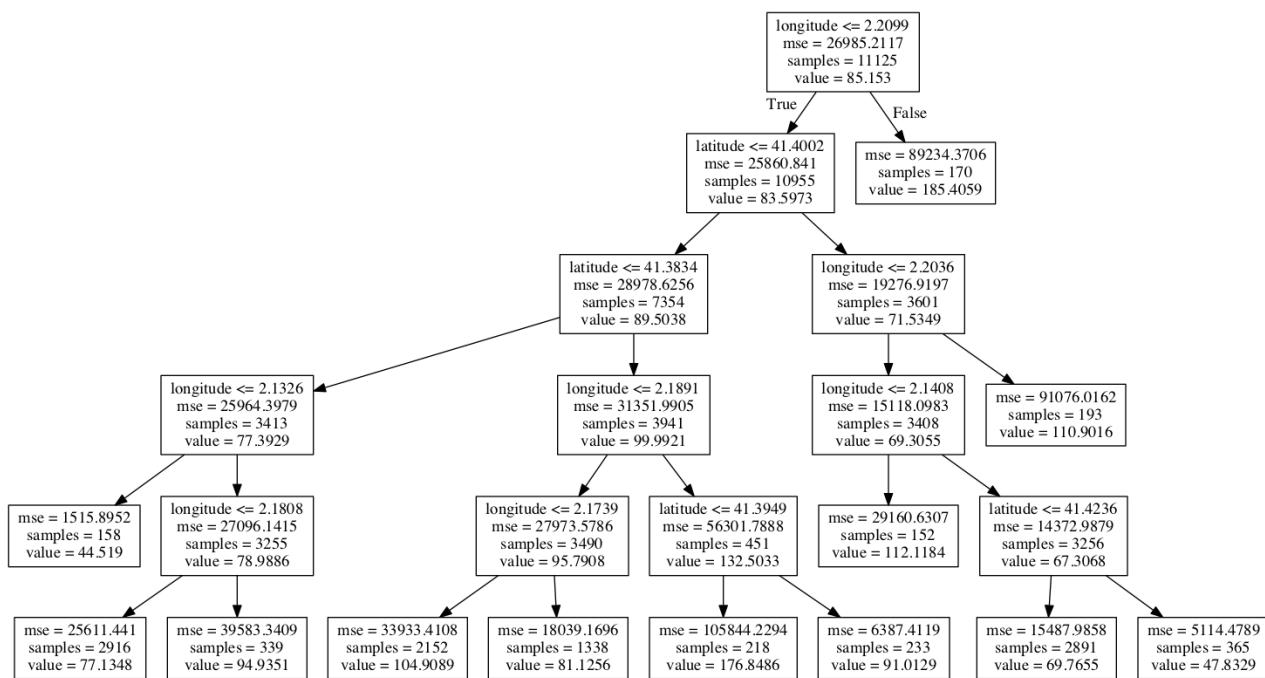


Fig 24. Representació de l'arbre de decisió dels microneighbourhoods amb *min_samples_leaf*=100 i *max_depth*=5.

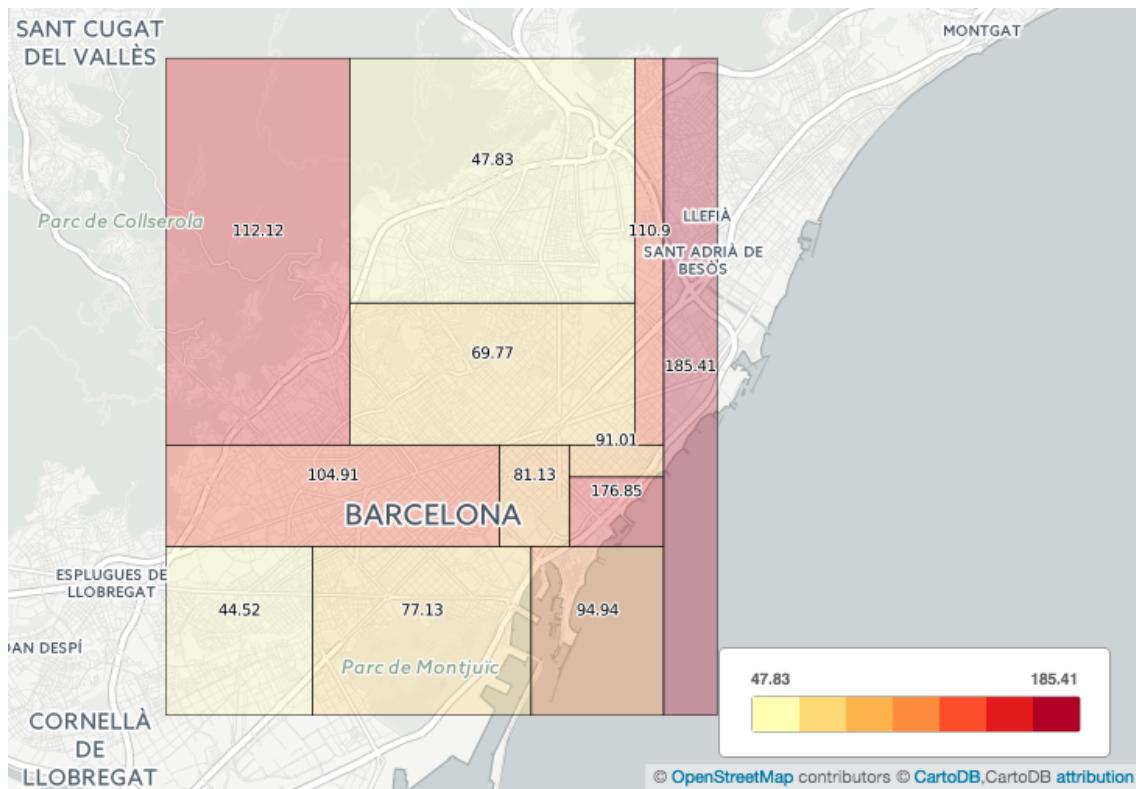


Fig 25. Microneighbourhoods generats amb un DT amb $\text{min_leaf_samples}=100$ i $\text{max_depth}=5$.

Com es pot apreciar en la Figura 25, degut a la generalitat que representen els valors dels paràmetres del DT construït, surten uns microneighbourhoods molt pobres en termes de detallats, sent uns polígons molt grans.

Seguidament s'han realitzat tres experiments més variant els valors dels paràmetres. En les següents il·lustracions es poden veure els microneighbourhoods generats per valors de $\text{min_samples_leaf}=150$ i $\text{max_depth}=10$ (Figura 26); per $\text{min_samples_leaf}=150$ i $\text{max_depth}=20$ (Figura 27); i per $\text{min_samples_leaf}=150$ i $\text{max_depth}=30$ (Figura 28).

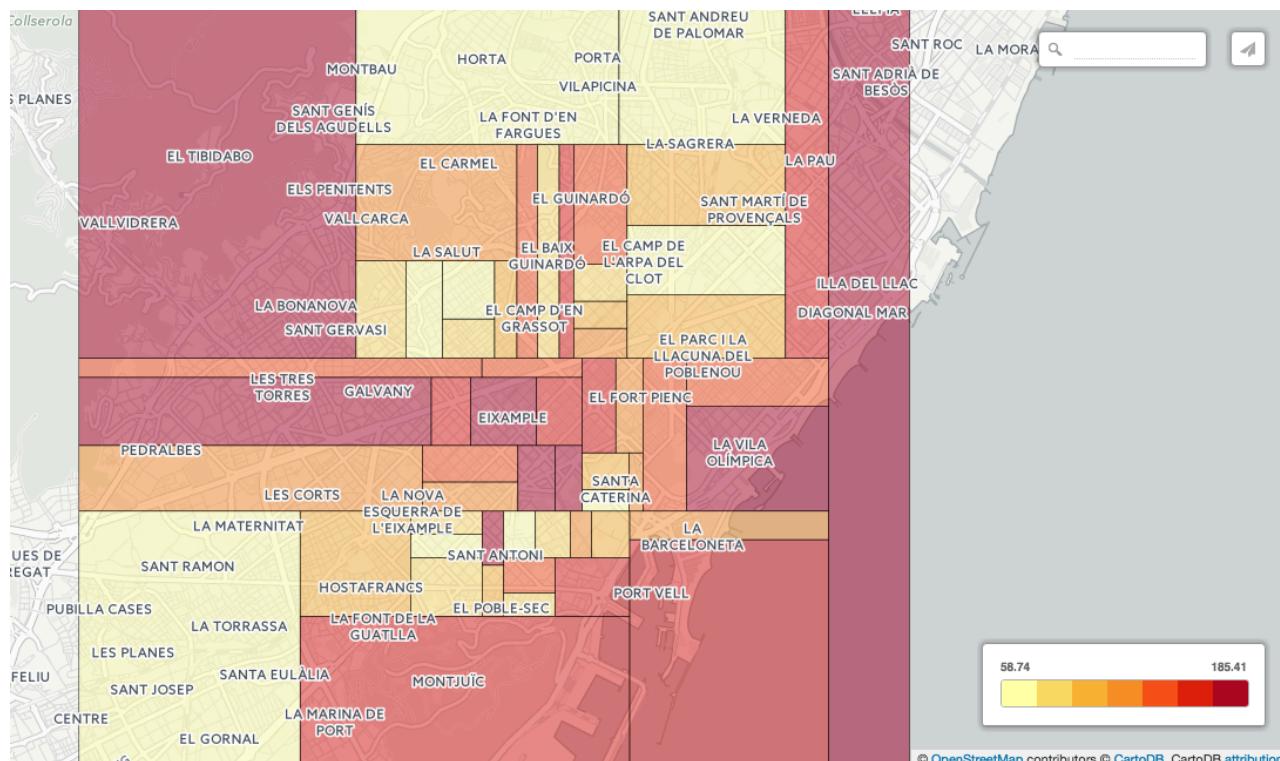


Fig 26. 49 microneighbourhoods generats amb $\text{min_samples_leaf}=150$ i $\text{max_depth}=10$.

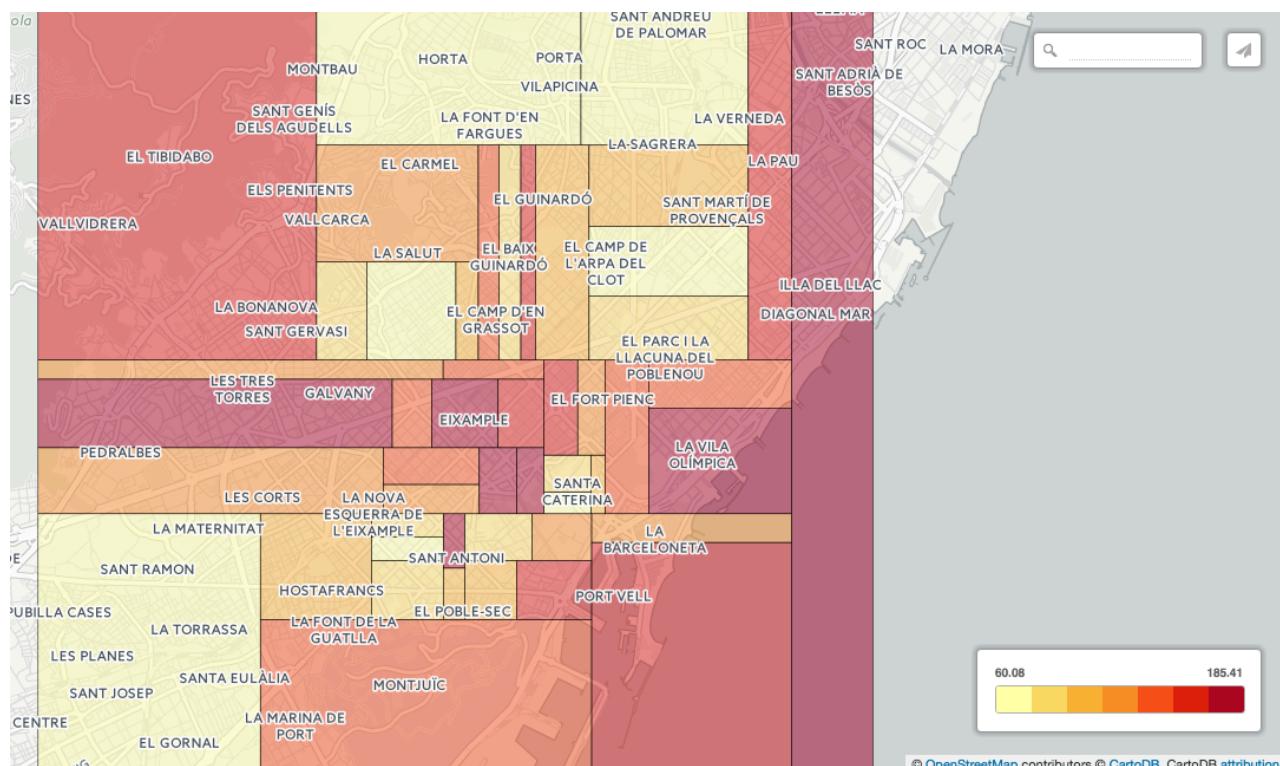


Fig 27. 57 microneighbourhoods generats amb $\text{min_samples_leaf}=150$ i $\text{max_depth}=30$.

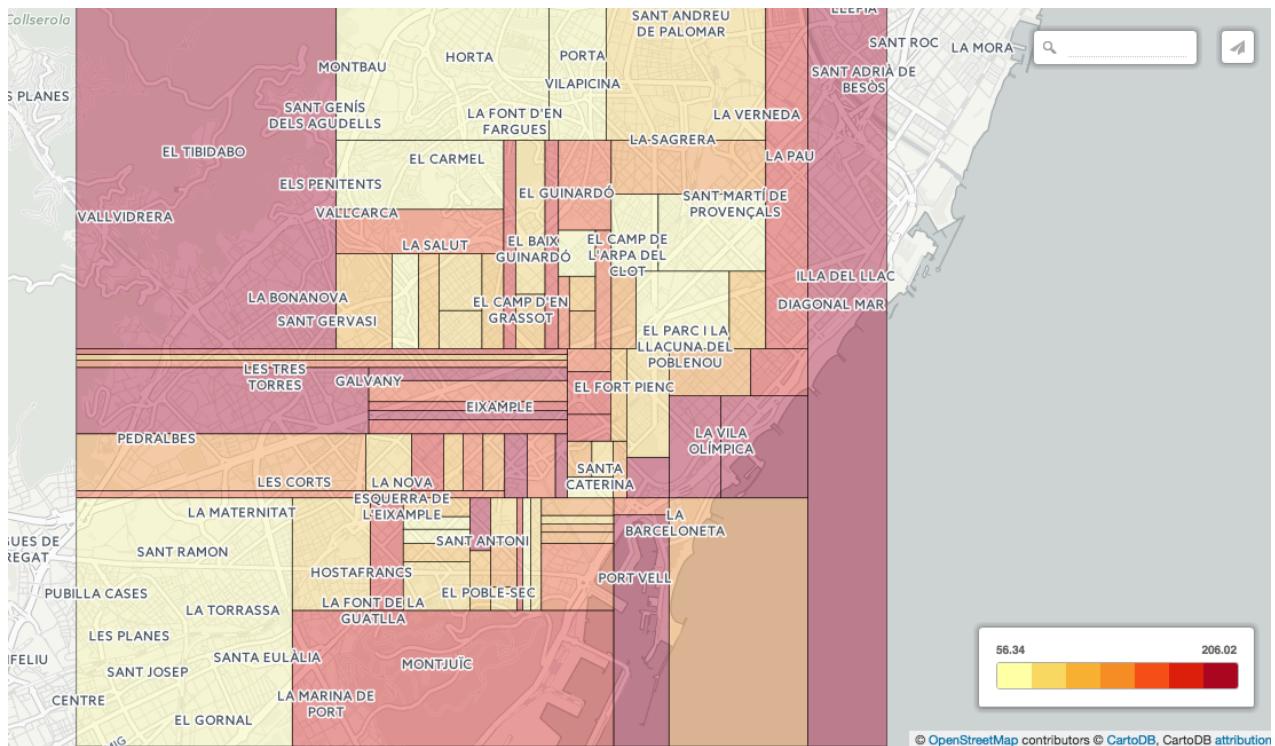


Fig 28. 88 microneighbourhoods generats amb $\text{min_samples_leaf}=100$ i $\text{max_depth}=20$.

Els arbres de decisió generats amb min_samples_leaf 150 i max_depth de 10 i 30 semblen construir microneighbourhoods més generals on delimiten zones més reals de Barcelona que el generat amb min_samples_leaf 100 i max_depth 20. Vegeu que la mida del polígon depèn bàsicament en el paràmetre min_samples_leaf i el nombre de polígons en el paràmetre max_depth . Llavors les zones exteriors de Barcelona presenten uns microneighbourhoods més grans degut per la baixa densitat d'allotjaments que hi ha comparat amb el centre de Barcelona. Traduint aquests resultats podem veure una fàcil i acolorida visualització de les zones on els allotjaments turístics són més cars.

Si ens fixem però en els microneighbourhoods obtinguts en San Francisco per l'equip de Airbnb (Figura 5 de la secció 2.4 en Metodologia) vegem que hi ha hagut algun tipus de post-processament que ha causat que les zones estiguin separades entre elles i que algunes zones on no hi ha allotjaments no es processin i llavors no es representi cap polígon allà. Es podria fer el mateix amb els resultats d'aquest projecte?

Estudiant el cas s'ha pensat en una forma de post-processar els microneighbourhoods obtinguts: afegir per a totes aquelles combinacions de coordenades on no representen cap allotjament un valor "trampa" del preu que representarien. De manera que el DT generarà moltes més regions degut a que la

densitat de població d'allotjaments serà màxima, però en contrast hi haurà moltes regions on el seu valor regressor serà semblant al valor “trampa”. Llavors simplement podem eliminar totes aquelles regions on el valor estigui al voltant del valor trampa i conservar les regions amb un valor diferent. A més, aquest processament generarà una distribució de les dades més semblant a una gaussiana i llavors els models construïts sobre aquestes dades seran models més robusts amb menys error.

Per assignar el valor “trampa” més òptim, s’ha provat primer de tot assignar la mitja del preu, però degut als pobres resultats obtinguts s’ha pensat assignar el valor 0. La Figura 29 il·lustra el resultat d’aquest post-processament, una distribució de 115 microneighbourhoods més neta i creible. El DT s’ha generat mitjançant els paràmetres `min_leaf_sample=100` i `max_depth=40`.

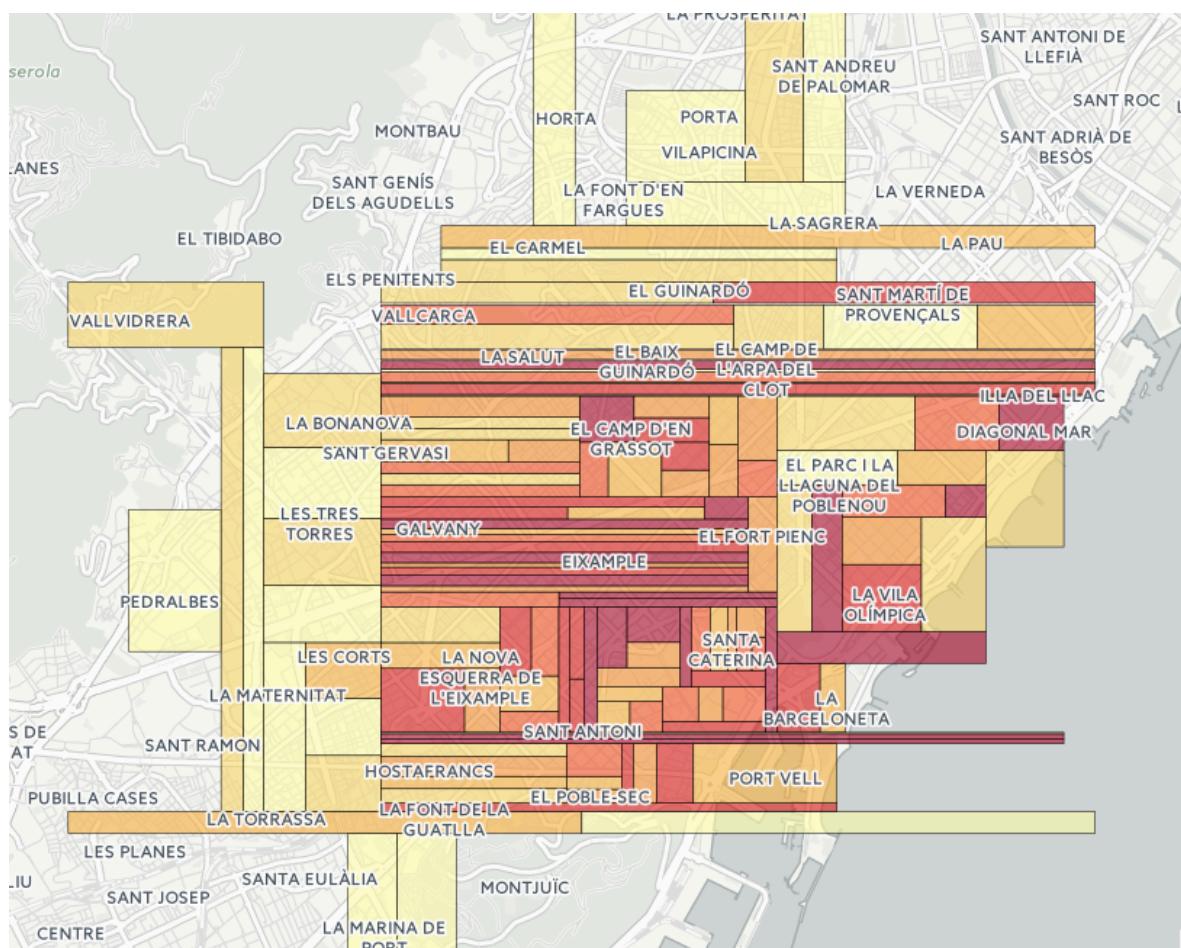


Fig 29. Microneighbourhoods generats amb un DT amb paràmetres `min_samples_leaf=100` i `max_depth=40` després de realitzar el post-processament de les dades.

4.3 Model predictiu

El desenvolupament del model predictiu del preu dels allotjaments es pot veure si es segueix la secció 3.4 de Desenvolupament. Com es diu, primer es realitza una neteja de les dades per aplicar models lineals simples (regressió lineal, Ridge, Lasso, Elastic Net, BayesianRidge) i models ensemble com el Gradient Boosting Regressor (GBR) amb una sèrie de mètodes d'optimització (cerca Grid dels millors paràmetres, selecció de variables, etc). Els resultats s'exposaran a continuació.

Aplicant els models lineals simples sobre una base de dades de 6369 observacions i 104 variables, on la variable de resposta és el preu (valor escalar), obtenim un error absolut de la mediana com el que es mostra en la gràfica de la Figura 30. El millor resultat és el que s'aconsegueix mitjançant el model BayesianRidge amb un error aproximadament de **16€**.

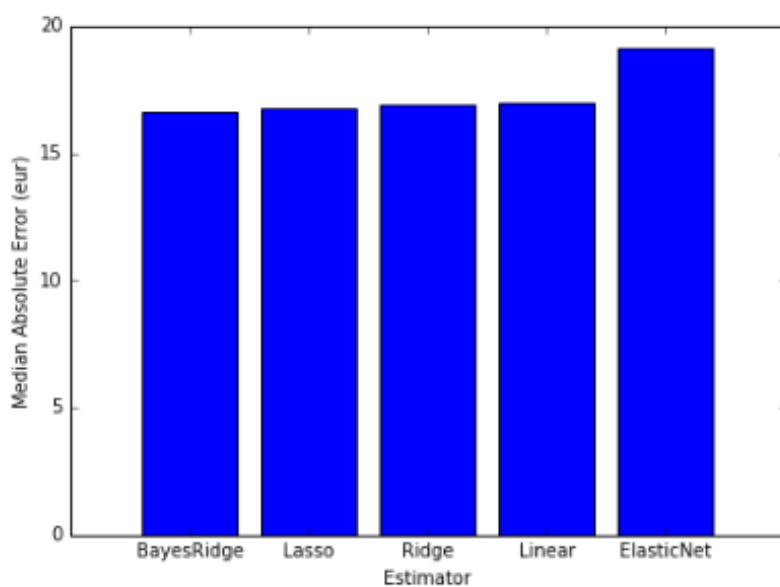


Fig 30. Resultats dels models lineals.

Aquest resultat es pot millorar aplicant mètodes ensemble tal com s'explica en la secció de Desenvolupament. Implementant el model GBR amb els paràmetres de la Taula 4, s'obté un error absolut de la mediana de **12,35€**, un resultat considerablement millor que l'obtingut amb els models simples.

Nombre estimadors	300
Màxima profunditat	4
Learning rate	0.01
Min samples split	1
Loss function	“ls” / “lad”

Taula 4. Paràmetres emprats primerament en el Gradient Boosting Regressor.

Si ara analitzem com l'error és afectat a cada ronda del boosting (veure Figura 31), vegem que a mesura que el nombre d'estimadors emprats augmenta, la desviació absoluta en el model disminueix però la pendent de la corba va disminuint convertint-se en una línia recta en algun punt llunyà. Llavors emprant un nombre major d'estimadors podríem obtenir encara millor resultat però s'ha de tenir en compte de no assignar un nombre d'estimadors molt alt ja que l'error del model no millorarà i possiblement s'acabi creant *overfitting*.

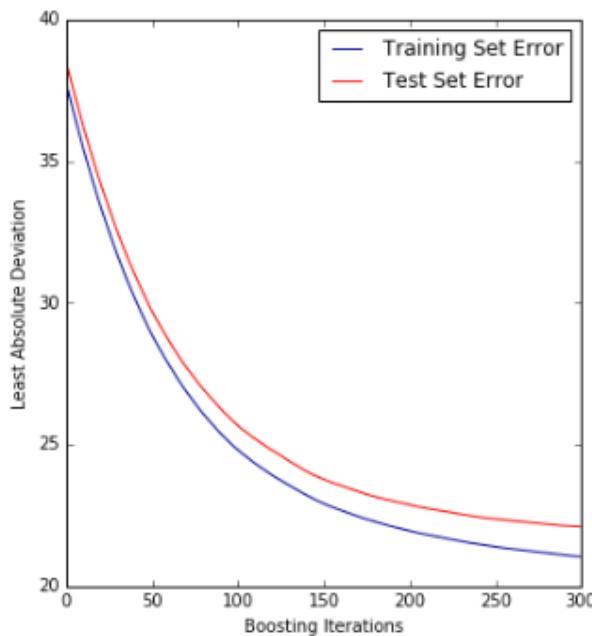


Fig 31. Comportament de la desviació absoluta en el GBR a mesura que el nombre d'estimadors augmenta.

Com hem vist en la Taula 4, només s'ha empatat un sol valor per a la majoria dels paràmetres. Ara podem realitzar una grid search a partir de l'error *cross-validation* per trobar els millors paràmetres a partir dels rangs de valors mostrats en la Taula 5 en un clúster de Apache Spark.

Obtenim que els paràmetres que proporcionen menys error són: Nombre d'estimadors 700; Learning rate 0.01; màxima profunditat 4; min samples split 1 i la

Least Absolute Deviation com a funció Loss. Construint un GBR amb aquesta configuració de paràmetres proporciona un error de **11,52€**.

Nombre estimadors	300 / 500 / 700
Màxima profunditat	1 / 2 / 4
Learning rate	0.01 / 0.001
Min samples split	1
Loss function	“ls” / “lad”

Taula 5. Paràmetres del model emprats en el grid search.

En la Figura 32 es mostra la importància que ha tingut cada variable alhora de construir el model GBR. S'aprecia que la millor variable indiscutiblement ha estat si un allotjament ofereix la casa sencera o no (“Entire home/apt”), seguit del nombre de comentaris publicats a l'allotjament (“Total_rcounts”). Vegem que les 5 variables amb més importància (excepte “Entire home/apt”) són variables escalars, on per exemple el nombre d'habitacions ofertes tenen més significació que la puntuació que té l'habitacle. Observem que la Vila Olímpica del Poblenou i la Dreta del Eixample són els barris on el preu té més variància damunt els altres barris. El barri Gòtic és el tercer barri més representatiu en la construcció del model.

Ara constraint de nou tots els models afegint però totes les característiques extrems de les imatges mitjançant la Hybrid Places-CNN, obtenim uns resultats per als models lineals com es mostra en la Figura 33. Observem que ara l'error disminueix, sent Lasso el millor regressor amb un error d'aproximadament **15€** ($\sim 1\text{€}$ menys que el BayesianRidge sense les característiques de les imatges).

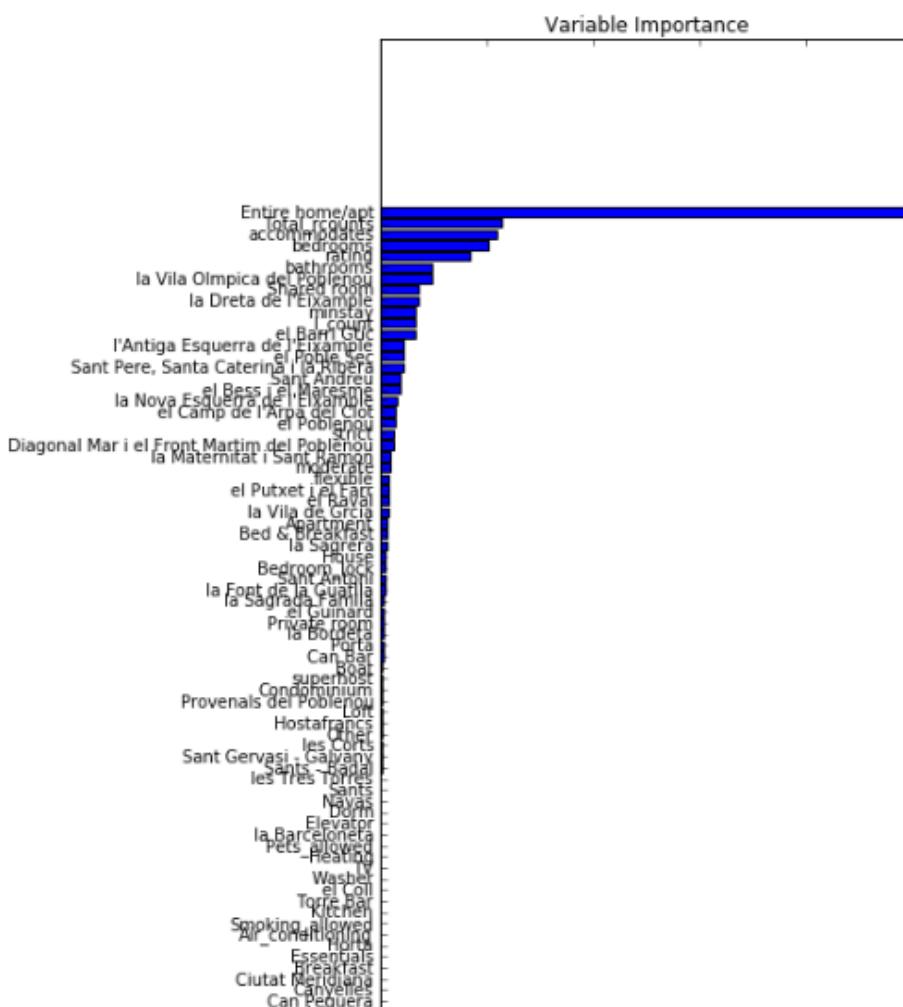


Fig 32. Importància de les variables en la construcció del GBR. S'ha retallat la gràfica degut a la seva grandària i que les variables que apareixien més avall no tenien cap tipus d'importància i eren, llavors, irrelevants a il·lustrar-les.

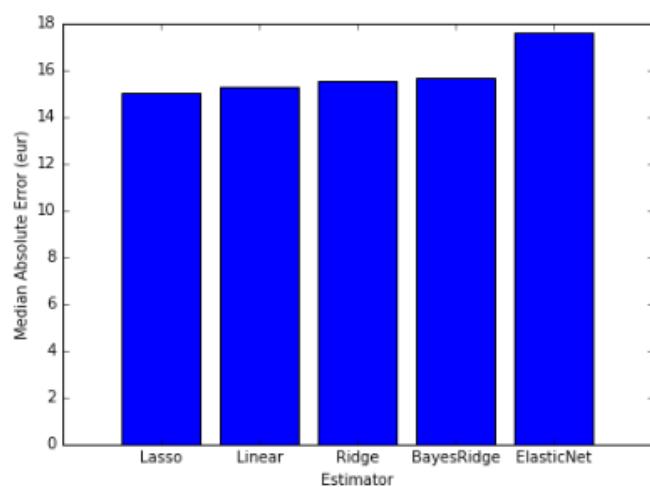


Fig 33. Resultats dels models lineals afegint les característiques de les imatges com a nous predictors al model.

Si ara construïm un Gradient Boosting Regressor amb els paràmetres trobats mitjançant la cerca grid, ens surt un error de **11,07€**. Vegem que afegir les característiques de les imatges definitivament millora el resultat del model.

Aplicant el mètode de selecció de variables mitjançant la selecció de subconjunt de k millors variables segons el seu valor de variància sobre la resposta sobre els dades amb les característiques de les imatges afegides, per als rangs de valors de k de [2,250] ens surt una gràfica de l'error dels models lineals com la de la Figura 34. Vegem que per a $k=1$ el BayesRidge i la regressió lineal proporcionen un error d'aproximadament **14€**. Aquest error té sentit degut a que els models lineals simples tendeixen a funcionar molt millor amb menys predictors que presenten més relació directe sobre la resposta, i que no estiguin correlacionats entre ells. A mesura que k augmenta, l'error es va semblant més a l'obtingut mitjançant els models construïts sense aplicar selecció de variables. Anoteu que el resultat ElasticNet no s'ha il·lustrat en la gràfica degut a que donava un error superior a 17€.

Constraint un GBR amb la millor selecció de paràmetres mitjançant grid search i amb el subconjunt de 55 millors variables ($k=55$) obtenim un error de **11,616€**. Com podem observar, aquesta configuració del GBR proporciona més error que l'obtingut mitjançant els anteriors GBR construïts.

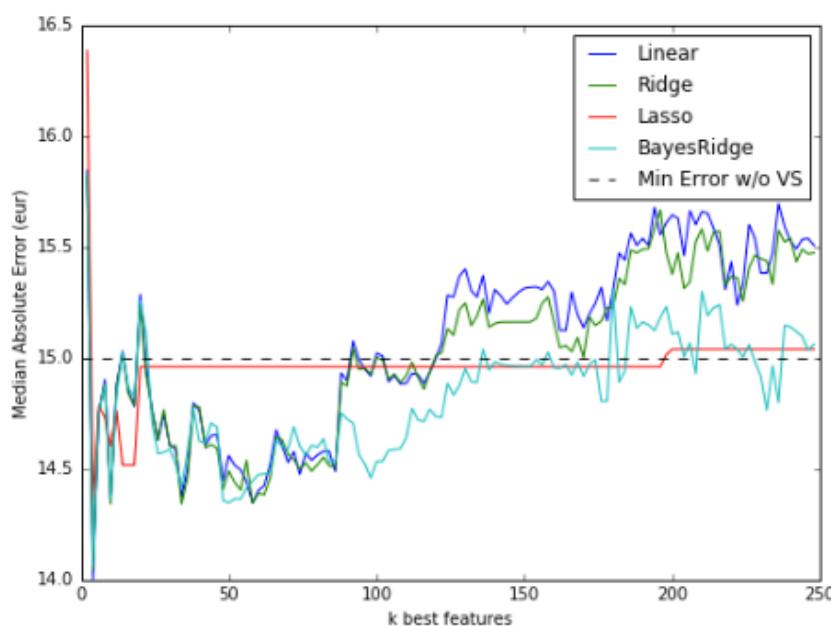


Fig 34. Distribució de l'error obtingut pels models lineals simples per al rang de k millors subconjunts de variables. La línia negra discontinua representa l'error mínim que s'ha obtingut mitjançant el model BayesianRidge sense realitzar selecció de variables.

4.3.1 Resum dels resultats

Per a una millor assimilació dels resultats obtinguts del model predictiu, en la Taula 6 estan il·lustrats tots els errors obtinguts de tots els models aplicats, per a cada una de les diferents configuracions implementades.

	Configuració 1	Configuració 2	Configuració 3
Reg. Lineal	~17€	~15,3€	~14€ *
Ridge	~16,9€	~15,7€	~14,2€ *
Lasso	~16,5€	~15€	~14,3€ *
BayesRidge	~16,1€	~15,8€	~14,2€ *
Elastic Net	~19€	~17,9€	>17€ *
GBR	11,52€	11,07€*	11,61€

Taula 6. Resum dels errors absoluts de la mediana dels models predictius sobre la resposta preu.

Els resultats en asterisc representen els millors resultats de cada fila, es a dir, el millor resultat proporcionat pel model en les tres configuracions aplicades. I el nombre en negreta és el millor resultat de tots els models en totes les configuracions.

La configuració 1 correspon als models construïts amb les dades obtingudes directament de la neteja principal, es a dir, les característiques extrems de les imatges i la selecció de les millors k variables no s'ha aplicat a aquesta configuració. El resultat del GBR és l'obtingut assignant els millors paràmetres trobats mitjançant el grid search.

La configuració 2 il·lustra els resultats dels models un cop s'han afegit les característiques de les imatges. El GBR aquí també s'ha construït mitjançant la millor configuració de paràmetres possible.

La configuració 3 correspon als models construïts amb la base de dades reduïda amb les k millors variables. Per als models lineals simples, s'ha escollit un valor de k=1, mentre que per al GBR s'ha escollit un valor de k=55.

CAPÍTOL 5: DISCUSSIÓ

Durant el transcurs del projecte s'han anat formulant preguntes interessants sobre les conclusions que podríem extreure de les dades. El capítol 4 de Resultats intenta contestar cada una d'aquestes preguntes. Algunes d'elles, com per exemple

- Com Airbnb està sent emprat per els seus usuaris a Barcelona? Quins habitatges tenen més probabilitats a ser part d'un negoci? Quins són els pisos més rentables?

S'ha contestat mitjançant les estimacions extretes de l'aplicació del model ocupacional. Malgrat que el model sigui una sèrie de formulacions per obtenir estimacions properes a la realitat, al cap i a la fi són estimacions i aquests valors obtinguts tenen un error sobre la realitat. Una forma òptima de disminuir aquest error és aplicar el *scraping* durant un temps més prolongat (1 – 2 anys) en intervals de 2-3 setmanes. Aquest projecte si es recorda només ha realitzat un sol procés de scraping i llavors les dades tenen una única data d'obtenció, no es presenta cap evolució en elles. Amb una evolució històrica dels allotjaments s'hagués pogut formular un model ocupacional molt més fort, estimacions més creïbles i també un model predictiu millor. També hagués estat pràctic afegir dades del sector turístic recents (2016) per contrastar la balança de ingressos/pèrdues de Airbnb i dels hotels, però degut a la poca quantitat de recursos actuals de Barcelona no s'ha pogut portar a terme. També si es disposés de dades de les despeses i impostos que han de pagar els amfitrions dels habitatges publicats a Airbnb, juntament amb l'estimació dels ingressos que obtenen es podria obtenir una estimació del guany net que obtindrien.

Els sistemes de regressió lineals aplicats han demostrat funcionar prou bé comparat amb els resultats obtinguts amb el mètode ensemble GBR. Normalment la capacitat de predicción dels sistemes ensemble és molt major que els models simples. Però com s'ha argumentat en la secció de Introducció i Metodologia, cada sistema de modelatge estadístic té les seves avantatges i inconvenients sobre cada tipus de problema. De manera que un model pot funcionar molt bé en un escenari, però en un altre completament diferent el mateix model pot proporcionar el pitjor resultat. La regressió lineal és un dels mètodes més simples però així i tot ha demostrat la seva bona capacitat de predicción en la secció de selecció de variables, superant l'error de tots els altres mètodes lineals. Això és degut a que la construcció dels coeficients en la regressió lineal funciona millor quan hi ha molt poques variables estant relacionades directament amb la variable dependent.

En la secció de construcció dels models de predicció s'han implementat tres configuracions: en la primera s'ha tingut en compte només els predictors obtinguts després de la neteja de variables obtingudes del procés de *scraping*; en la segona configuració s'han afegit les característiques de les imatges; i en l'última s'ha aplicat un mètode de selecció de variables en les dades de la segona configuració.

En aquest projecte s'han implementat 4 mètodes de regularització: ElasticNet, Lasso, Ridge i BayesianRidge. Elastic Net és el model que ha proporcionat els pitjors resultats en totes les configuracions, això pot ser degut a que el seu algoritme fa que el model funcioni millor quan hi hagin moltes variables correlacionades entre elles, però en el nostre model s'ha demostrat que els predictors presenten un valor baix de correlació entre ells. Ridge Regression també ha sofrit una pujada de l'error degut a aquest escenari. Per això Lasso i BayesianRidge han estat els que ha proporcionat el millor resultat dels 4 mètodes de regularització des de que demostren un millor funcionament quan les variables tenen poca relació entre elles. En termes de *overfitting* es pot deduir la hipòtesi de que no n'hi ha ja que l'error que s'obté amb els models no és del tot baix, sinó que deixa un marge de predicció prou gran (15€ de mitja).

En el cas del Gradient Boosting Regressor, es sap que són models que demostren gran capabilitat de predicció però s'ha d'anar en compte en la seva construcció, ja que són models difícils d'afinar degut a la gran possible quantitat de paràmetres que es pot testejar i també tendeixin a crear *overfitting* molt fàcilment. Entre les actuacions del GBR que s'ha implementat en aquest projecte, el GBR creat amb els paràmetres obtinguts després de realitzar la grid search en un clúster de Apache Spark ha proporcionat el millor resultat. Cal anotar que l'última configuració (la del model construït amb les dades amb les k millors variables), el GBR ha incrementat el seu error comparat amb les altres dues configuracions. Això és degut a que els mètodes ensemble combinen molts d'estimadors dèbils analitzant cada variable. De manera que si hi ha variables amb poca relació amb la resposta, aquestes generaran un estimador amb molta variància, i llavors en uns resultats pobres. L'algoritme del gradient descendent del boosting combina els errors dels models de manera que cada estimador creat en cada iteració aprèn dels errors dels anteriors, llavors si cada estimador dèbil presenta una variància gran, al combinar-los el resultat és un classificador molt més fort que el que es crearia si les variables tinguessin més relació amb la resposta.

Mitjançant l'extracció dels microneighbourhoods s'ha apreciat una amable visualització de les zones més cares de Barcelona. A través dels diferents mapes creats amb les dades s'ha arribat a la forta conclusió de que la part nord de Ciutat Vella, juntament amb la part centre i dreta de l'Eixample resideixen els allotjaments amb més rendibilitat però també menys econòmics. Amb l'ús d'aquest estudi es desitja proporcionar al tercer de informació útil de quines pautes hauria de seguir per obtenir el màxim benefici d'emprar Airbnb en Barcelona.

5.1 Treball futur

En aquesta secció focalitzem la projecció que podrien tenir els resultats del projecte en un futur. Malgrat que la implementació de la aplicació amb Docker del projecte és una aplicació trivial i senzilla, s'ha vist que aquesta plataforma ofereix un ampli rang de possibilitats de cara a la projecció d'un producte. Si es tinguessin dades del mercat immobiliari de Barcelona, una distribució dels preus de les diferents etapes de l'any semblant a la que empren les empreses que assignen els preus dels vols, i informació demogràfica de Barcelona actualitzada, es podria desenvolupar un model de recomanació del preu dels allotjaments que es desitgin publicar en la pàgina web de Airbnb. Aerosolve [44] és una eina de recomanació de preus creada per Airbnb la qual implementa un sistema molt semblant.

Les imatges dels allotjaments solament s'han emprat per l'extracció de les característiques aplicant la Hybrid Places-CNN ja entrenada. Degut al poc nombre d'imatges descarregades no ha estat possible construir una CNN pròpia. Si s'haguessin tingut els recursos necessaris, s'hauria estudiat la possibilitat d'implementar una CNN de manera que hagués estat entrenada inicialment amb un conjunt molt gran d'imatges per després afegir el seu preu com a característica. Ajuntant els resultats idealment hauríem d'obtenir una CNN la qual per cada imatge d'entrada, se li assigna un preu després de realitzar un forward.

Seria interessant també ajuntar l'aplicació pensada amb Docker juntament amb les dades del mercat, distribució de preus segons les dates de l'any, etc, amb la CNN entrenada amb les imatges dels allotjaments per desenvolupar un recomanador de preus definitiu. D'aquesta manera l'aplicació suportaria qualsevol tipus de característica en l'habitatge (imatges, facilitats, descripcions del barri, localització, mercat, etc) per després ser processada en el procés de recomanació.

Agraïments

M'agradaria donar les gràcies a aquelles persones que m'han donat suport i ajudat mitjançant la seva experiència i saviesa en escriure aquest projecte.

Primer de tot, gràcies al director d'aquest projecte, Jordi Vitrià, per tutejar-me i guiar-me de la millor manera possible durant tot el desenvolupament del projecte. Gràcies també a la Universitat de Barcelona (UB) per facilitar tots els recursos i bibliografia emprats en l'execució del projecte.

Seguidament, gràcies a Tom Slee per compartir el seu codi que implementa el *web scraping* amb mi i adaptar-se a les necessitats del projecte.

Finalment, gràcies a la meva família i amics, per sempre esser allà quan més els he necessitat.

Referències

- [1] *Lloguers per a les vacances, cases, apartaments i habitacions – Airbnb* (27 juny de 2016). Pàgina web oficial, obtingut de www.airbnb.com
- [2] *Inside Airbnb. Adding data to the debate.* (27 juny de 2016). Pàgina web oficial, obtingut de www.insideairbnb.com
- [3] *InAtlas – BigData and Location Analytics* (28 juny de 2016). Pàgina web oficial, obtingut de www.inatlas.com
- [4] Hill, Dan (20 agost 2015). *The Secret of Airbnb's Pricing Algorithm* [Blog post]. Obtingut de <http://spectrum.ieee.org/computing/software/the-secret-of-airbnbs-pricing-algorithm>
- [5] Airbnb Inc (2014). *Airbnb Economic Impact* [Blog post]. Obtingut de <http://blog.airbnb.com/economic-impact-airbnb/#barcelona>
- [6] Turisme de Barcelona, Diputació de Barcelona, Ajuntament de Barcelona (Juny 2015). *Estadístiques de turisme a Barcelona i comarques* [Document PDF]. Obtingut de <http://professional.barcelonaturisme.com/imgfiles/estad/Est2014.pdf>
- [7] *Apache Spark – Lightning-fast cluster computing* (28 juny 2016). Pàgina web oficial, obtinguda de <https://spark.apache.org/>
- [8] Docker – Build, Ship, and Run Any App, Anywhere (28 juny 2016). Pàgina web oficial, obtinguda de www.docker.com
- [9] Wikipèdia, the free encyclopedia (21 juny 2016). *Web Scraping* [Article wikipèdia]. Obtingut de https://en.wikipedia.org/wiki/Web_scraping
- [10] *PostgreSQL: The worlds most advanced open source database* (28 juny 2016). Pàgina web oficial, obtinguda de www.postgresql.org
- [11] Wikipèdia, the free encyclopedia (20 juny 2016). *Ordinary Least Squares* [Article wikipèdia]. Obtingut de https://en.wikipedia.org/wiki/Ordinary_least_squares
- [12] James. Gareth. Witten. Daniela, Hastie. Trevor, and Tibshirani. Robert (2014). *An Introduction to Statistical Learning: with Applications in R*.
- [13] Tikhonov, A. N. (1963). "О решении некорректно поставленных задач и методе регуляризации". Doklady Akademii Nauk SSSR 151: 501–504.. Translated in "Solution of incorrectly formulated problems and the regularization method". Soviet Mathematics 4: 1035–1038.
- [14] Kumar, T. Krishna (1975). "Multicollinearity in Regression Analysis". Review of Economics and Statistics 57 (3): 365–366. JSTOR 1923925.
- [15] Scikit-learn ORG (2010-2014). *Generalized Linear Models* [Documentation]. Obtingut de http://scikit-learn.org/stable/modules/linear_model.html

- [16] O'Sullivan, Finbarr. "A Statistical Perspective on Ill-Posed Inverse Problems." *Statist. Sci.* 1 (1986), no. 4, 502--518. doi:10.1214/ss/1177013525.
<http://projecteuclid.org/euclid.ss/1177013525>.
- [17] Freund, Yoav and Schapire, Robert E. (1999) "A Short Introduction to Boosting"
- [18] Breiman, Leo (Juny 1997). "Arcing the Edge"
- [19] Friedman, J.H. (Febrer 1999). "Greedy Function Approximation: A Gradient Boosting Machine"
- [20] Friedman, J.H. (Març 1999). "Stochastic Gradient Boosting"
- [21] Wikipèdia, the free encyclopedia (20 juny 2016). *Gradient Boosting* [Article wikipèdia]. Obtingut de https://en.wikipedia.org/wiki/Gradient_boosting
- [22] Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. "Learning Deep Features for Scene Recognition using Places Database." *Advances in Neural Information Processing Systems 27* (NIPS), 2014. PDF Supplementary Materials
- [23] Tom Slee (juny 2016). Pàgina web oficial, obtingut de www.tomslee.net
- [24] Project Jupyter 2016 (Juny 2016). Pàgina web oficial, obtingut de www.jupyter.org
- [25] The Python Software Foundation (1999-2016). *18.2 json – JSON encoder and decoder* [Documentation]. Obtingut de <https://docs.python.org/2/library/json.html>
- [26] The Scipy community (2008-2009). *NumPy 1.11* (Maig 29, 2016). Obtingut de <http://docs.scipy.org/doc/numpy/reference/>
- [27] Lambda Foundry, Inc. And PyData Development Team (2011-2012). *Pandas 0.18.1* (Maig 2016). Obtingut a <http://pandas.pydata.org/>
- [28] The Python Software Foundation (1999-2016). *13.1 csv – CSV File Reading and Writing* [Documentation]. Obtingut de <https://docs.python.org/2/library/csv.html>
- [29] Farwell, Corey (2015). *Python-geojson library*. Obtingut de <https://github.com/frewsxcv/python-geojson>
- [30] Lopez, Guille (Gener 2015). *Barcelona-neighbourhoods-geojson* [Github]. Obtingut de <https://gist.github.com/guillelopez/26f57e5966b5f8825e9b>
- [31] Hunter, J.D (2007). *Matplotlib: A 2D graphics environment*. IEEE COMPUTER SOC. *Journal of Computing In Science & Engineering*, Volum 9, Número 3, pp 90—95.
- [32] CartoDB, Inc. *Map your World's Data – CartoDB*. Pàgina web oficial, obtingut a www.cartodb.com
- [33] Zou, Hui; Hastie, Trevor (2005). "Regularization and Variable Selection via the Elastic Net". *Journal of the Royal Statistical Society, Series B*: 301–320.

- [34] David Cournapeau (2007). *Scikit-learn: Machine Learning in Python*. The scikit-learn organization. Pàgina web oficial, obtingut a www.scikit-learn.org
- [35] Scikit-learn org. *Gradient Tree Boosting* [Documentació]. Obtingut a <http://scikit-learn.org/stable/modules/ensemble.html#gradient-tree-boosting>
- [36] Scikit-learn org. *GridSearchCV* [Documentació]. Obtingut a http://scikit-learn.org/stable/modules/generated/sklearn.grid_search.GridSearchCV
- [37] Spark sklearn team. *Spark_sklearn Documentation* [Documentació]. Obtingut a <http://pythonhosted.org/spark-sklearn/>
- [38] Scikit-learn org. *SelectKBest* [Documentació]. Obtingut a http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest
- [39] Gillies, Sean. (Desembre 31, 2013). *The Sapeley User Manual* [Blog post]. Obtingut a <http://toblerity.org/shapely/manual.html>
- [40] Google Brain Team, Google Inc. *TensorFlow – an Open Source Software Library for Machine Intelligence*. Pàgina web oficial, obtingut a www.tensorflow.org
- [41] Jia, Yangqing and Shelhamer, Evan and Donahue, Jeff and Karayev, Sergey and Long, Jonathan and Girshick, Ross and Guadarrama, Sergio and Darrel, Trevor. (2014). *Caffe: Convolutional Architecture for Fast Feature Embedding*. Journal of arXiv preprint arXiv:1408.5093.
- [42] Olga Russakovsky*, Jia Deng*, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. (* = equal contribution). *ImageNet Large Scale Visual Recognition Challenge*. IJCV, 2015.
- [43] Ajuntament de Barcelna. *Open Data de l'Ajuntament de Barcelona*. Obtingut de www.opendata.bcn.cat/opendata/en
- [44] Yee, Hector and Ifrach, Bar. (04 juny 2015). *Aerosolve: Machine learning for humans – Airbnb Engineering* [Blog post]. Obtingut a <http://nerds.airbnb.com/aerosolve/>