NOVA IMS Information Management School

MGI

Mestrado em Gestão de Informação
Master Program in Information Management

TYPE EXTRACTION FROM REAL ESTATE LISTINGS

Alexandra Martins Serras

Project Work presented as partial requirement for obtaining the Master's degree in Information Management

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School

Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa

TYPE EXTRACTION FROM REAL ESTATE LISTINGS

by

Alexandra Martins Serras

Project Work presented as partial requirement for obtaining the Master's degree in Information Management, with a specialization in Knowledge Management and Business Intelligence

Advisor: Rui Alexandre Henriques Gonçalves

February 2020

ACKNOWLEDGEMENTS

To my family and friends that have supported me every step of the way.

ABSTRACT

As the real estate market grows, products that aggregate listings from several websites start to appear. With the number of real estate websites, and consequently the number of property listings that exist, it is not feasible to tag listings by hand .

This tagging is fundamental in order to create products from these extracted listings. Products like Automated Valuation Models, Outlier Detection and even search filters depend on a reliable extraction of the property type.

This project had the aim of proving that to create such model we don't need to resort to black box algorithms that give us little interpretability and require a higher level of expertise to debug and maintain. These type of algorithms also tend to require more data to train, which means the data has to be manually labelled which can prove to be a time consuming task.

By using a list of keywords to extract from the text and an XGBoost model created a package that extracts the type of listing and gives us some logging information.

In this project we managed to get a 95% accuracy across all categories, however the model struggled when encountering listings that can be identified as new development. This approach proved that we don't always need a state of the art model, which can be complicated to understand, to obtain good results.

KEYWORDS

Text Classification; Real Estate; Xgboost;Supervised Learning

INDEX

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 PROBLEM IDENTIFICATION

With the rise of real estate websites, the need to aggregate listings from several websites into one has arisen. In order to be able to have a centralized website where someone could go an see all of the listings in the market, websites like Idealista, Imovirtual and Casafari have been created.

When a listing is posted by the realtor itself, one can be confident that it will have the necessary information, whether it is visible or not, to group listings by type. This filter is a basic need for a real estate website since the user does not want to have to thread through lots of listings just to find their preferred type.

In websites like Idealista, which aggregate information from several different sources, there is the need to create a way to automatically assign a type to a property. Another difference from the traditional real estate website is that Idealista parses listings from several countries, in several languages.

When aggregating information from several realtors by scraping their websites there is no failproof way of ensuring we are scraping the type of listing. In order to ensure that every listing scraped has a type associated, we need to create an algorithm that analyses its characteristics and associates its category.

The model we will develop has the aim of automatically assign a category to a listing. These categories are plot, house, apartment, new development, and investment.

The main variables when choosing the pipeline and structure of the project were the following:

- Execution time - Since the algorithm will be running in real-time, one listing cannot take a long time to go through the pipeline.
- Development time - The priority is to have a system running that can reliably identify each category. Having a system totally independent of language and extracting keywords without aid could be possible, however, the development time would be considerably higher.

- Hardware limitation - Due to the low amount of GPU's available, using algorithms that relies heavily on a GPU, like to run was out of the question.

1.2 NEW DEVELOPMENTS

New developments are a special kind of property. These listings are of properties that are not built yet and that give an investor the opportunity to buy the final product in advance. The main characteristics of this type of property are:

- They can either be a house, an investment or an apartment.
- They can only be for sale.
- Their construction state has to be 'under construction'.

These types of listings have the following difficulties:

- The description often lacks keywords that separate them from normal listings.
- Often, unless the pictures are checked, it is not possible to distinguish a new development from a normal apartment due to the lack of indicators in the text.

The main way of identifying a new development is by looking at the pictures. Given that these listings are of properties finished in the future, they tend to have computer-generated pictures, easily distinguishable from real pictures by humans and computers.

1.3 OTHER CATEGORIES

The other categories we will attempt to classify in this model will be apartment, house, investment, other and plot. Usually listings for these categories are straight forward, the category will appear in the tittle as well as in the description as we can see in Figure 1.

Apartamento T2, remodelado, com vistas desafogadas na Amora, Seixal

Description

Fantástico apartamento T2 (com 3 divisões) no coração da **Amora**, com **vistas desafogadas**, uma **ampla exposição solar** e próximo de **transportes públicos**.

Este imóvel possui uma **sala acolhedora com 13m2** com **pavimento cerâmico** e **orientada a Poente** (o que lhe garante uma excelente exposição solar e elevado conforto térmico ao longo de todo o ano) que, através de uma **dupla portada** de corpo inteiro dá acesso a uma **marquise com 5,2m2** que também permite aceder à cozinha.

A **cozinha possui 5m2 de área útil**, foi totalmente remodelada há menos de 5 anos (incluindo a substituição da canalização) possui uma bancada de granito Monção a toda a dimensão da cozinha, bem como se encontra revestida a cerâmico negro para garantir uma maior higiene e contraste com a restante cozinha. Evidencia **linhas modernas e funcionais** e encontra-se **parcialmente equipada** para maior conforto dos compradores, sendo que a **lista de equipamento** inclui: forno, exaustor, esquentador e fogão a gás.

Através de um **corredor** com uma área de cerca de **4,8m2**, acede-se à área dos quartos (**ambos orientados a Nascente**), sendo que o que se encontra junto à casa de banho possui uma área de **13,6m2** e o **segundo quarto** uma área de **13,7m2**.

Figure 1 – Example of title and description of a listing. Source: idealista.com

2 LITERATURE REVIEW

2.1 REAL ESTATE

According to [1], the estimated size of the professionally managed real estate market was of 8.5 trillion dollars in 2017. With the increase of independent real estate agencies, the market has lost most of its transparency. To combat this, companies like Idealista, Zillow or CASAFARI have been created. These companies aggregate listings from several real estate agencies websites and build a database with them. This enables companies to build better appraisal or market analysis algorithms.

Zillow is a marketplace that gathers data from several different websites into their own and processes it in order to gain insights. Their main focus[2] is on home valuation models such as Zestimate. Zestimate[3] is a tool that estimates the market value of a house. It takes into consideration many of the home's characteristics (one of them being the type) and outputs an estimate on how much the house is worth.

The main topic studied in real estate is appraisals. An appraisal is the act of assigning a value to a property, often by computer systems called Automated Valuation Model (AVM). This value can vary according to several variables like the location or type of property. Countless algorithms have been designed with the aim of calculating what we should pay for a real estate listing, each with different approaches and input variables. In [4] a genetic algorithm is attempted on houses in the city of Naples, taking into account the location, the area, the price, the maintenance status and the number of floors. In [1] the authors developed an algorithm to evaluate listings published in Salamanca in Idealista. The highest performing one was an ensemble of regression trees that took as an input the zone, the postal code, the street name and number, the floor number, the type of asset (apartment, house etc), the constructed area, the floor area, the construction year, the number of rooms and baths, if the property is a duplex or a penthouse, if the property has a lift, a box room, a swimming pool, a garden, parking, the parking price, and the community costs.

A common theme in these approaches is that the type of listing is either limited to only one[4], or there is a variable stating which type the listing is. Every algorithm requires clean data to perform well but when using information that was automatically extracted

from different sources this cannot be assured, so there is the need to create algorithms that ensure that this data has the right values, either by extracting type and location or by removing outliers from our dataset.

Another topic studied in real estate is market analytics. Market Analytics is used to give the client, or realtor a visual representation of the competing properties in the market. It can also be used in order to see how costly a specific area is by gathering properties on sale, dividing them into groups and averaging their sale value. Websites like Idealista[5], that gather information from several different sources have started offering their clients this product as it can give them a vast overview of the market.

2.2 TEXT MINING

2.2.1 Applications

With the increasing amount of data collected every day, text processing has become one of the most important techniques in machine learning. According to [6] text mining "refers generally to the process of extracting interesting and non-trivial information and knowledge from unstructured text.". Websites like Twitter, Reddit, Facebook collect most of their data in the form of text. As most of the data collected is in an unstructured form, text mining has become necessary to extract knowledge and business insights.

With the widespread use of text data, there are many industries that make use of it. Companies that want to keep up with the opinion of the customers about their products can employ keyword extraction or sentiment analysis on their data. It can be employed to structure medical documents or consultation reports, allowing models to be built that can change the way medicine is done today.

The main uses of text mining today is keyword or keyphrase extraction, text categorization, and sentiment analysis[7].

Sentiment analysis is used to extract and identify subjective information from the text, mainly if the sentiment is positive, negative or neutral. This is mainly used by companies to gauge the happiness of their customers by analysis tweets, reviews or

even comments.This technique is mostly used to assess the performance of a product or monitoring the reputation of a brand on social media[8].

Keyphrase or keyword extraction is the process of extracting the most important and relevant words or sentences from a document. This technique is used in several different areas, such as recommendation systems, and translation.

2.2.2 Keyword Extraction

Keyword extraction consists of extracting a set of significant words from a document. This can be used to create labels for news articles, extract keywords to input into an unsupervised model to be able to cluster together documents by the type of keyword.

There are two main approaches to keyword extraction: the bag of words and tfidf.

2.2.2.1 Bag of words

The bag of words[9] is a model where the frequency of each term in a document is counted to serve as an input to a classifier. The model counts the frequency of each term across a document without regarding the semantic relationship.

It requires some pre-processing to ensure that the same word is not considered different. For example, ‘Apartment’ and ‘apartment’ would not be considered the same term by the model because of the different cases.

This model has some drawback, for example, if we do not remove stop words from the text, the bag of words will consider them a keyword due to their frequency.

2.2.2.2 TFIDF

The Term Frequency-Inverse Document Frequency[10] is a statistic that is used to understand which words are most important in a document. Unlike bag of words, TF-IDF does not simply count the number of times a word has appeared in a document, but rather takes the number of occurrences of the word and counters with the frequency of

the word in the whole corpus. This is done to ensure that words that are more common than others do not have an unnecessary higher weight. Take, for example, the word ‘and’, this word is extremely common and in a bag of words performed without any stop word removal would show up as one of the most important keywords. Since TF-IDF counters the number of occurrences in one document with the number of occurrences in the whole corpus, this word will have a much lower ranking in TF-IDF.

This statistic is the result of the product between the term frequency and the inverse document frequency. The term frequency is the number of times a word appears in the document using the following formula:

$$0{,}5 + \frac{0{,}5 * frequency}{maximum\ occurences\ of\ words}$$

The inverse document frequency is the weight used to measure the importance of a given term. This is used to reduce the importance of terms that occur frequently in a corpus and increase the importance of words that occur rarely.

$$\log \frac{Total\ number\ of\ documents}{number\ of\ documents\ the\ term\ appears\ in}$$

However, this statistic does not perform well in small documents due to the fact that in short texts a word tends to show up only once, which leads to having too many terms as candidates with the same score.

2.2.2.3 Rule-Based Approach

The most basic type of text categorization is the rule-based approach. This approach only needs a set of rules implemented that are then run against the document. An example of the use of this approach is [11] one where the author managed to create a part-of-speech tagger with comparable performance to probabilistic methods. The advantage of this approach is that the rules can be understood by a person, however, they require much more maintenance than other approaches[12].

2.2.2.4 Dictionary Based Approach

A dictionary-based approach is used when we have a previously created list of keywords and we just filter the text for them. This is useful when the number of keywords known and in a relatively small quantity or with a well defined scope[13].

This approach is also useful when we have texts written in several languages. Given that each language has its own set of stop words and lemmatization, some approaches include filtering the language beforehand and building a model for each one. Other approaches translate the languages to one common language and run the model on the translation. This has significant drawbacks as we are relying on the correct translation to be done.

If we have a small set of words that we want to detect and feed our classifier, having a dictionary will significantly increase the speed. However, we must ensure that whenever a new keyword is introduced into the system we add it to the dictionary.

2.2.3 Categorization

According to [6] text categorizations consists of "identifying the main themes of a document by placing the document into a pre-defined set of topics." This is done by using a supervised learning algorithm which inputs labeled data to train and then classifies new, unseen data.

A text categorization model can be divided into three steps[14]:

1. Feature extraction
2. Training
3. Predicting on unseen data

In the first step, we extract features from the text and convert them into a vector that the model can read. This is done by first pre-processing the data by removing stop words, lemmatizing, converting every word to its lowercase alternative and then implementing some keyword extraction technique like the bag of words and tfidf. After these techniques, we can filter out the number of features by their importance. In the second

step we use the features previously created and train a classification model with them and then evaluate the performance of the classifier with our test set.

It can be used in many different domains, from classifying customer feedback to aid customer success employees in improving their performance by knowing what the topic of the comment is about.

2.3 SUPERVISED LEARNING

When approaching a problem that requires the use of machine learning there are two main methods that can be followed: unsupervised and supervised learning. The main difference between the two methods is the prior knowledge of the truth[15].

An unsupervised learning algorithm tries to make sense of the unlabeled data provided in order to extract features and knowledge. The most popular application of this methodology is in clustering. A supervised algorithm, on the other hand, uses a pre-labeled dataset and tries to fit the data in order to predict new observations.

Supervised learning algorithms are typically used in classification and regression problems. In classification problems we want to predict a discrete value, for example, extracting the topic of a text document. In regression problems we are trying to predict a continuous value, like the price of a house in Lisbon according to its characteristics.

For regression problems[16], the algorithms typically used are linear regression, regression trees or neural networks. For classification problems we typically use simple decision trees, an ensemble of decision trees or neural networks. Since the problem we will be discussing fits in the categorical category we will only discuss decision trees, random forests and gradient boosted trees.

2.3.1 Decision Trees

Decision trees[17] are a non-parametric model that can be used either in classification or regression problems. As the name suggests, it builds a model in the shape of a tree. It divides the original dataset into smaller and smaller datasets until it can associate only

one label to the data. The main advantages of using decision trees is how easy they are to interpret and how quick they are to train, however, a small change in the data can cause the structure of the tree to change and thus some predictions to also change.

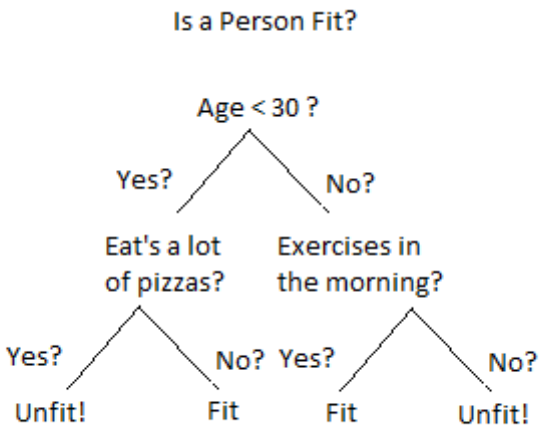

Figure 2 - Example of a decision tree[18]

To build a decision tree we must go through different steps:

1. Splitting – deciding which variable to use in order to split the dataset. This decision is usually made by using the Information gain as a metric to indicate how good a feature is.

2. Pruning – limiting the depth of the tree to avoid cases where we overfit. This can be achieved by removing duplicate rules from the tree.

This model is the base of the ensemble algorithms we will use and discuss in this project.

2.3.2 Random Forest

The random forest[19] algorithm is an ensemble of decision trees. An ensemble is a technique the combines a “set of models to produce an estimator that has better predictive performance than the individual components” .

The random forest algorithm works by randomly sampling the dataset with replacement, known as bagging, and selecting a random subset of features. By using both of these methods we ensure that each decision tree is trained differently and learns the dataset in a different way. This creates a more robust model since each tree is fitted to different sets of data so the probability of it being affected by outliers is lower.

The final prediction, in regards to classification problems, is made by taking the class which is most predicted by the individual models, this is known by voting.

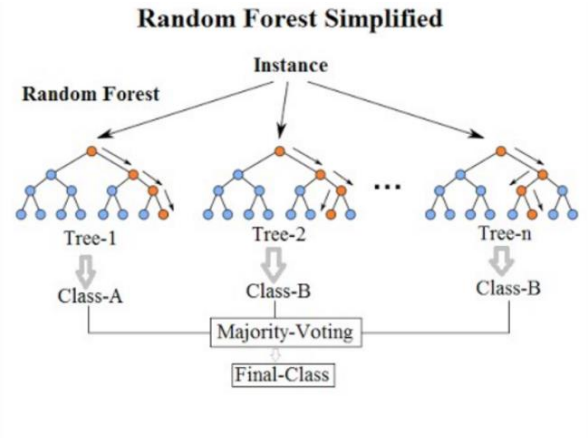

Figure 3 - Example of a random forest[20]

2.3.3 Gradient Boosted Trees

Gradient boosted trees[21] are another type of ensemble model that, like random forests, use the decision tree model as its base model. The main difference between this model and the random forest is that instead of training the decision trees on different subsets of data, we train the decision trees on the residuals of the previously trained decision tree.

To use this we must first set a loss function to optimize. This function varies according to our objectives, it can be mean squared errors or simply the difference between the predicted and the actual value.

This model works by training a simple model, calculating the error residuals, fitting a new model on the residuals and adding both models. It then fits another model on the residuals of the previous models and so on until the loss function reaches its minimum. When it is time to predict the model with output a weighted sum of the predictions of each individual tree.

2.4 FEATURE SELECTION

Feature selection is an essential step when developing a model. Not only does it allow us to increase the model performance, but also, increase its interpretability (a model with five variables is much more understandable than a model with fifteen).

The feature selection method we will be using in this project is the recursive feature elimination[22]. This approach is superior to other methods like Principal components analysis (PCA) since it retains the original variables whereas the PCA creates new, hard to understand, variables from the input, however it requires the algorithm we are using to have a way of measuring feature importance.

It starts off by training the model with all of the features and ranking them by importance. Then it removes the feature deemed least important, retrains the model, assigns new importance values and computes the performance metrics we chose (depending on the problem it can either be accuracy, precision or even the multiclass classification error rate), it starts the training process again, ranks the features according to importance and removes the least important. This process goes on until the performance metrics can no longer be improved by removing features.

3 METHODOLOGY

In the following chapter describes the methodology followed for this model. We will explore the dataset, explain the decisions made when transforming the variables, the reason why the model was chosen and discuss techniques to prevent overfitting.

We will also present the software used and discuss the pipeline of the overall project, one for training the model and one to use in production to predict on demand.

3.1 PIPELINE

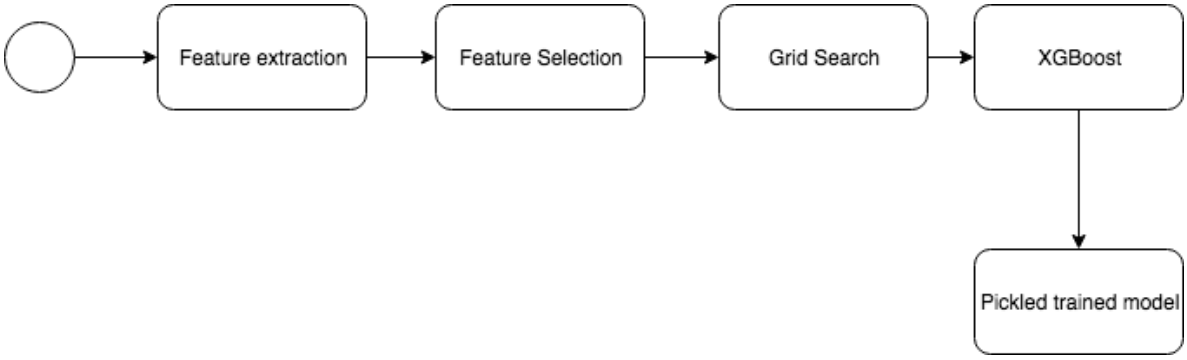

Figure 4 - Training Pipeline

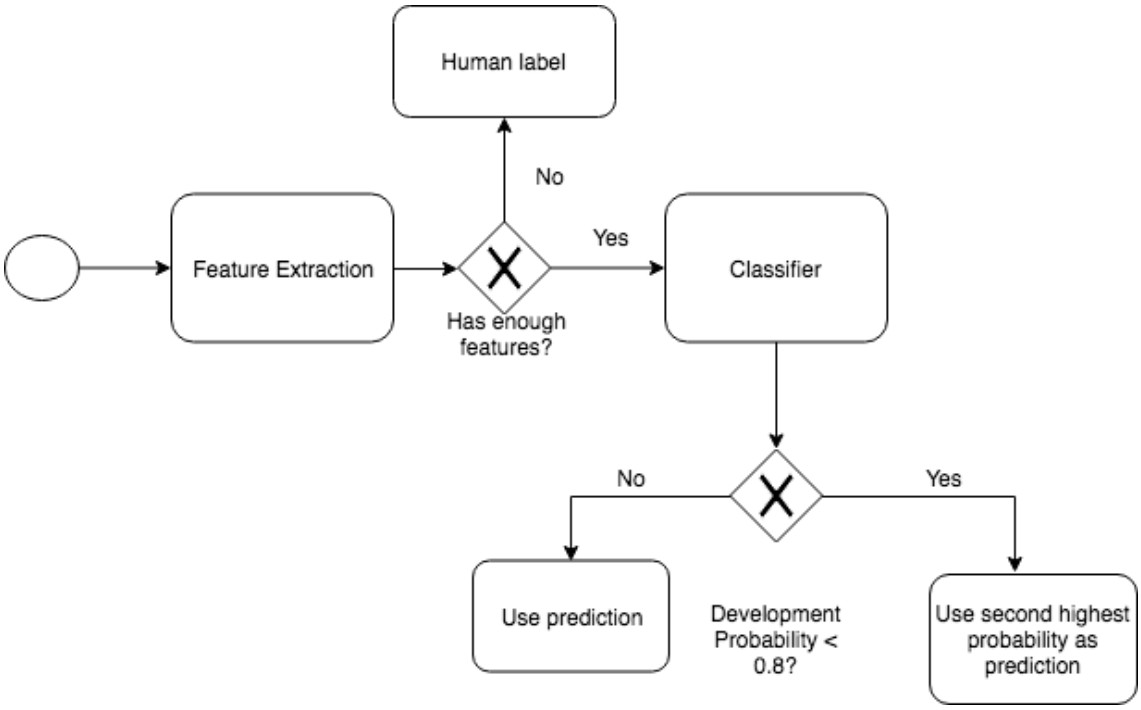

Figure 5 - Production Pipeline

Given that we do not want to retrain the model every time we wish to predict one listing, the project will be divided into two separate pipelines. One for the training phase (Figure 4) of the model, which will be run whenever more training data needs to be added in order to reflect the database, and the production pipeline to be run whenever we have new listings that need a type to be associated with them.
Figure 5 shows the proposed pipeline for predicting new listing types. There are two decision points:

1. It begins with data processing, followed by data transformation and finally feature extraction. If by the end of this part of the pipeline there are not enough features (no keywords in the title, description or subtype fields), the listing will be sent to a human annotator that will be in charge of manually classifying the property.
2. In order to reduce the chance of missing a new development type of listing, when the algorithm classifies a listing as new development, this listing must have a probability above 80%, if this is not the case, the second-highest probability will be used to decide the type.

3.2 SOFTWARE USED

To implement the pipeline presented above the programming language Python was chosen. This is an open-source programming language that can be used in many different ways most notoriously for machine learning. It has a large community supporting it and creating packages that can be installed and used easily.
For this project, the packages xgboost, sklearn, pandas, pymysql, pickle, and unidecode were used.

3.3 DATA PREPROCESSING

3.3.1 Data Extraction

The data used to develop this algorithm is obtained from crawling different real estate websites for listings. For the training of the model, we used a dataset with 7000 listings which have been balanced to ensure that each class has the same number of listings.

For testing we used a dataset composed of listings extracted in two days, 16000 listings. The testing dataset is not balanced given that we want to simulate two days of predictions so we will naturally have more samples for apartment and house than for plot.

3.3.2 Initial Variables

In our dataset, each real estate listing will have the following attributes:

Table 1 - Initial variables

Variable	Description	Example
Price	The price of the listing	0 - no price set 230 000, 1 000 000
Description	Description of the listing	
Title	Title of the listing	
Subtype	Type of the listing	‘Apartment’,’ House’, ‘Terreno’
Bedrooms	Number of bedrooms in the property	0,1,2,3,4
Bathrooms	Number of bathrooms in the property	0,1,2,3,4
Location	The location of the	‘Lisbon’, ‘Madrid’

	property	
URL	The URL of the website it was taken from	
Condition	Condition of the property	‘New’, ‘Used’, ‘Para Recuperar’, ‘In construction’
Total Area	The total area of the listing	122, 195
Living Area	The living area of the listing	122,195
Plot Area	The plot area of the listing	122,195
Construction Year	The Construction year of the listing	2014, 1985, 2019
Rent Status	If the property is available for rent	‘Active’, ‘hold’, ‘none’
Group	Target variable. The type of property	‘Apartment’, ‘House’, ‘Investment’, ‘Plot’, ‘Other’,’New Development’

3.3.3 Data Transformation

3.3.3.1 Continuous variables

- Year after(0/1) - 0 if the construction year variable is before the current year, 1 if the construction year is after the current year
- Max area - the maximum between living_area and total_area
- Rent (0/1) - if the rent status is ‘active’ then 1, if ‘hold’ or ‘none’ then 0
- Has typology - if the description or title contain T1, T2, T3.

3.3.3.2 Text variables

This section describes the operations applied to the description, subtype, title, condition and URL variables.

For all of the variables, the following initial steps were followed:

1. *Nan* values were filled with a space.
2. The column was converted to text. Cases where the field was composed of only numbers would cause an exception.
3. Every word was converted to lowercase.
4. Every word with an accent was converted to a version without an accent. E.g: não gets converted to nao.

For the following sections, a dictionary-based approach was chosen where a list with the accepted keywords is used when running the bag of words method. This approach was chosen due to the small length of the text, which makes it difficult to extract keywords, either by using TF-IDF or bag of words.

A Bag of Words is a package available through sklearn[23] that counts how many times a word occurs in each document. This is used due to the fact that the models we use take as input numerical values, so we must find a way of translating our text into numbers. By limiting the bag of words with a specific vocabulary, we reduce the number of candidates to words we know are associated with a category. In order to only output the categories in English, after we run the bag of words we switch each occurrence to its equivalent in English.

The variables subtype and title are transformed with the process described above, whereas the description, URL and condition need some additional steps.

3.3.3.2.1 Description

When analysing a real estate listing we can tell that generally the group of the listing is mentioned in the beginning of the description, whereas additional groups the property might have (for example garage is in the 'other' category, however it might be mentioned in a listing for an apartment as an important feature) are generally situated in the end of the description. In order to give different weight to the words found in the beginning and words found in the end we divided the description into three equal parts and allow the model to assign these weights by itself.

For each part of the description we run the bag of words package to obtain the number of keywords associated with each group. In the end this process generates eighteen new features (six for the first part of the description, six for the second part and six for the third part).

A new feature is also generated from the description. This feature indicates whether the typology (T1, T2, etc) was stated in the description. This feature was created for cases when the type of property is not specifically stated like we often see in Idealista listings.

3.3.3.2.2 Url

When processing the Url what we are looking for are situations where the type of the listing is written on it.

An example is 'www.tecnocasa.es/venta/piso/madrid/madrid/323410.html', in this case, we want to extract the word 'Piso' as it means apartment in Spanish and it gives us a clue when neither the description, the title or the subtype give us enough information.

To achieve this, besides the operations described before, the URL is also separated by '/' and joined again as a normal string. A bag of words is then applied to it and the words that match are counted. This transformation generates six new features.

3.3.3.2.3 Condition

One characteristic of new developments is that usually their condition is set as 'under construction' or 'in project'. The condition variable will only be used for these cases.

The condition field is similar to the title or subtype variables, however, when generating the vocabulary for the bag of words to use we are only going to use words that have a group set as 'new development'.

3.3.4 Correlation Matrix

The next step is to analyze the correlation between our variables. This analysis allows us to understand which variables might have a relationship with each other. There are three types of correlation: positive, negative and neutral. When we have a positive correlation between variables it means that when one increases the other also increases, for this model we considered values above 0.75 as positively correlated. On the other hand, when the correlation is negative, when one variable increases, the other decreases, we considered values below -0.75 to be negatively correlated. The final type is neutral where no relationship can be observed.

In our dataset the variables for the house title and the house subtype have a correlation of 0.84, the variables plot title and plot subtype have a correlation of 0.91 and other url and other title have a correlation of 0.77.

We have chosen to keep all of the variables and let xgboost decide which one to use in the splitting of branches. We will also optimize the col_samplebytree further on.

3.3.5 Missing Values

In this section, the missing values will be analyzed. For this section, only the price, bedrooms, bathrooms, total area, living area and plot area variables are going to be considered. When extracting the information from the websites the parser replaces every non existent numerical value with zero. Due to this fact, there are no Nan values in these fields, however, we will consider the value zero as missing values.

For the plot area, bedrooms and bathrooms, some conditions will be imposed before the analysis. Given the nature of these variables, we will only consider them as missing in certain conditions:

- The number of bedrooms and bathrooms and living area will only be considered a missing value when the listing belongs to the apartment or house category.
- The plot area will only be considered as missing whenever the type of listing is ‘plot’.

Table 2 - Percentage of missing values

	Bedrooms	Bathrooms	Living Area	Plot Area	Price
Apartments	13.60%	33.92%	49.52%	-	18.24%
Houses	19.52%	34.08%	60.96%	-	4.16%
Plots	-		-	50.48%	3.12%

Table 2 contains the percentage of listings that have a missing value in each variable. We can see that 50% of the listings that are classified as a plot do not have their area set. When analyzing missing values in the house category, we can see that it generally does not have the living area defined. It is also possible to observe that the number of bathrooms tends to be missing more times than the number of bedrooms from the listing, this could be because we are often more interested in knowing the typology of the property rather than the number of bathrooms.

Even though we have identified these missing values, we will not impute any of them. For the imputation to be reliable the type of listing (what we are trying to predict) would be necessary. For example, if we try to impute the price of an apartment by using the median, it will be skewed because of listings that are either in an expensive zone (Madrid, Lisbon) or by other types of listings (a house will typically have a higher price than an apartment). Similarly, how would we be able to impute bedrooms without knowing which type of listing it is? A plot will not have either bedroom, bathrooms or living area, if we impute these values for houses, we will have to impute them for plots since we have no way of filtering listings by type.

3.3.6 Ranges

Besides the analysis of the missing values, we also analyzed the range of our transformed variables. This was done in order to better understand our dataset and help us assess the results from the feature selection. In this section, we will be analyzing the ranges of the subtype, title, first and second part of the description. The third part will be ignored as it usually does not give us much information.

As we can see from Figure 9 there is no point in using how many new development keywords exist in the subtype as it is always zero. The same thing happens with the keywords for ‘Other’.

As expected, the number of keywords in any category is higher in the first part of the description than in the second part.

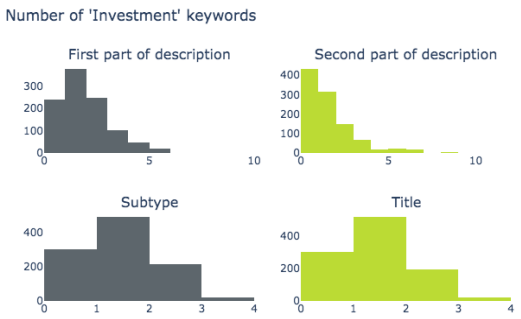

Figure 6 - 'Investment' keyword range.

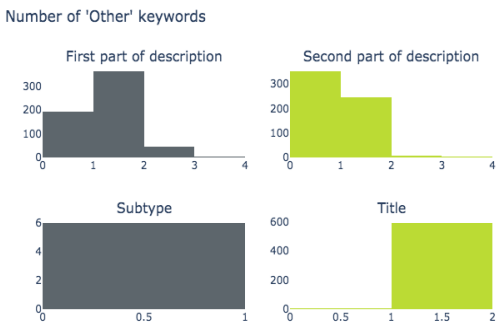

Figure 7 - 'Other' keyword range.

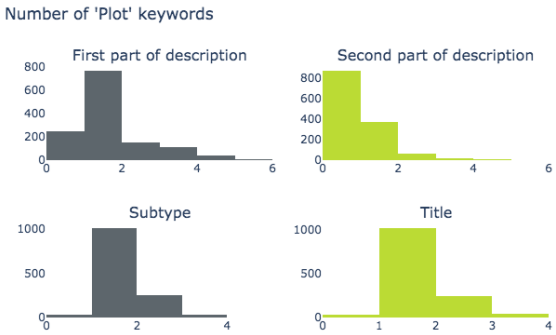

Figure 8 - 'Plot' keyword range.

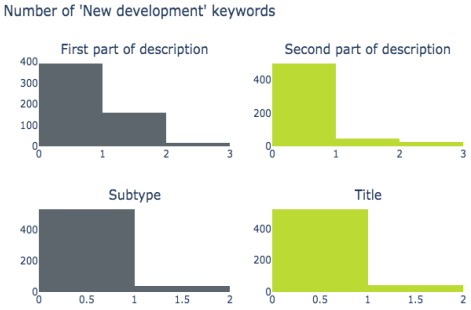

Figure 9 - 'New Development' keyword range.

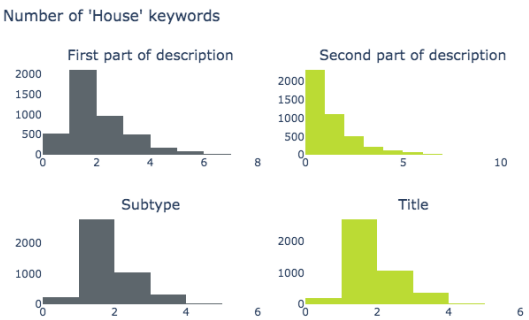

Figure 10 - 'House' keyword range.

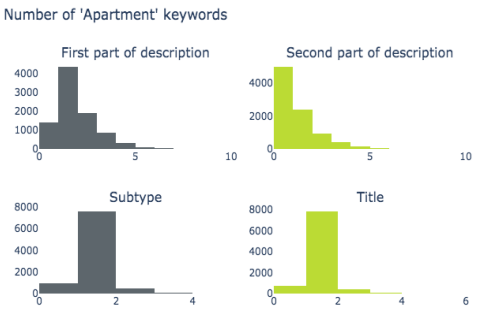

Figure 11 - 'Apartment' keyword range.

3.4 DATA PREPARATION

3.4.1 Feature Selection

After extracting the features, we get a sparse matrix with forty-four columns.

In order to eliminate the unnecessary features, we ran a recursive feature elimination and cross-validation package. This package comes bundled with sklearn in Python.

The way this package works is it starts with all of the features in the model, uses these features to train and cross-validate it. Afterward, it removes one feature, retrains the model and scores the validation dataset. If the score improves, the feature is permanently removed. This process is repeated until the score is no longer improved by removing any feature.

After running the package, as shown in Figure 12, the number of features that causes the cross-validation score to stagnate is thirty-two. The final features are the following:

- price,
- maximum area,
- if the listing is for rent,
- if the condition is for new development,
- if the listing has a plot area, if the construction year is in the future,
- the number of keywords associated with ‘house’ in the first and second part of the description,
- the number of keywords associated with ‘Apartment’ in the first and second part of the description,
- the number of keywords associated with ‘Plot’ in the first part of the description,
- the number of keywords associated with ‘Other’ in the first and second part of the description,
- the number of keywords associated with ‘Investment’ in the first and second part of the description,
- the number of keywords associated with ‘New Development’ in the first part of the description,
- the number of ‘Apartment’ keywords in the subtype, the number of ‘House’ keywords in the subtype,
- the number of ‘Plot’ keywords in the subtype,

- the number of ‘Investment’ keywords in the subtype,
- the number of ‘Other’ keywords in the subtype,
- the number of ‘New Development’ keywords in the subtype,
- the number of ‘Apartment’ keywords in the title,
- the number of ‘House’ keywords in the title,
- the number of ‘Investment’ keywords in the title,
- the number of ‘New Development’ keywords in the title,
- the number of ‘Plot’ keywords in the title,
- the number of ‘Other’ keywords in the title,
- the number of ‘Apartment’, ‘House’, ‘Investment’ and ‘Other’ keywords in the URL.

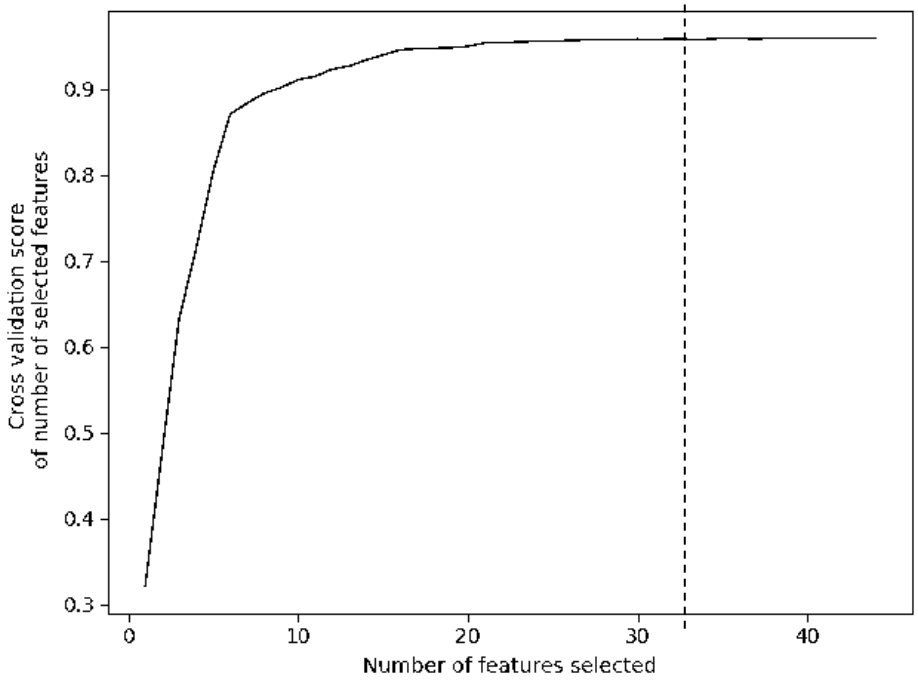

Figure 12 - Number of selected features vs cross-validation score.

3.5 MODEL

The model used to predict the category of the listing was a gradient boosting algorithm, XGBoost. The XGBoost implementation of gradient boosted trees was chosen due to the fact that the library allows parallelization across several CPUs and the algorithm itself was programmed with the aim of reducing computing time. This couple with the fact that other algorithms, like random forest and decision trees, present a slightly lower accuracy this algorithm was chosen.

The XGBoost algorithm is most known for being the most efficient gradient boosted trees algorithm. It has used in many competitions like the Netflix Prize [24], where Netflix released a dataset of 100 million movie ratings and challenged everyone to build a model that could beat the accuracy of their own.

XGBoost was first introduced in [25], with the aim of optimizing the performance and speed of normal gradient boosting algorithms. It is capable of performing Gradient Boosting, Stochastic Gradient Boosting and Regularized Gradient Boosting. It works by boosting several decision trees (either classification or regression (CART)). Boosting is used to convert weak learners to strong learners, it does this by training the weak learners sequentially trying to correct the previous incorrect predictions. Since XGBoost uses gradient boosting it tries to fit the new predictor to the residual errors of the previous predictor.

According to the authors XGBoost can be used as either a standalone, in a competition, or integrated into a real-world production pipeline. An example of this real-world production pipeline is how facebook uses it to predict ad click-through rates[26]. In that paper XGBoost is used in conjunction with probability calibration.

LightGBM, another gradient boosting algorithm was also tested. This algorithm is considered superior to XGBoost mainly in terms of speed. After running the same dataset through the algorithm, we found that it produced slightly inferior results to XGBoost, with similar training and testing speed.

We also considered joining both algorithms for our predictions, however, given that it tended to make the same mistakes as XGBoost it would only be confirming its predictions and bring no further advantage to our model.

3.5.1 Overfitting Strategy

Overfitting occurs when a model learns the training dataset ‘too well’. It happens when the model adapts itself to the original training data and ends up learning the noise and randomness.

An overfitted model is easy to spot. The model will have high accuracy on its training set, however, when using new, unseen data it will struggle to make the correct predictions, making the test dataset accuracy much lower than the training.

Tree-based algorithms are susceptible to overfitting, so we must ensure that it does not happen to our model. There are two ways of preventing overfitting:

- Splitting the dataset into folds and performing cross-validation on them.
- Tuning the parameters designed to prevent it.

3.5.2 Parameter Tuning

Before creating the final model, we must first run a grid search for parameter tuning. Given the propensity of xgboost to overfit, we need to tune some parameters to ensure this does not happen.

There are two types of parameters that can be adjusted to avoid overfitting. To control the model complexity, we will tune the maximum depth for the base learners. To add randomness and make training robust to noise we will tune colsample_bytree, which sets the subsample of columns to use when constructing a tree and eta, which is the learning rate. Besides these parameters, we will also tune the n_estimators parameter, which sets the number of trees to fit.

After running a grid search on models built with stratified k fold and cross-validation, the optimal parameters and the ones that are going to be used in the model were 0.5 for colsample_bytree, a learning rate of 0.1, a max_depth of 11 and 100 estimators.

3.5.3 Stratification and cross-validation

The second strategy to avoid overfitting is to split the data using a stratified k fold strategy, which splits the data into five different subsets. In order to ensure that each of these splits is representative of each class in the dataset, we used the stratification technique, this way ensuring that each fold will have a set number of each class.

Besides using the stratification technique, we also ran cross-validation on it. This approach uses the datasets mentioned above, chooses one subset to run validation and uses the rest to train the model. In order to minimize the variability of the results, the cross-validation is run five times with different datasets for training and validation. By running cross-validation we reduce variability and ensure that the model we chose is really the best possible with the features and parameters we chose.

After running this, we choose the best performing model, pickle it and use it in production.

3.5.4 New Development Rule

Given the nature of new developments, as seen in the confusion matrix presented in the results section, the classifier tends to assign this class to too many listings, which leads us to have a low precision for this class. In order to reduce these occurrences, we applied a rule where a listing to be classified as 'new development' must have a new development probability above 0.8.

This threshold was chosen by analyzing the probability assigned to the class when it was rightly classified and when it was wrongly classified. By analyzing these values, we can see that when the classifier is only 60% sure of its prediction, it gets it wrong. We then created a plot that correlates the number of times that a listing is wrongly classified as a new development with the probability the classifier outputs for that prediction. By analyzing Figure 13 we can see that we have two peaks: when the probability is 0.6 and when the probability is 0.8. To decide which peak to choose we must first decide what is more important: identifying as many new developments as possible or having few false positives. For this model we opted for identifying as many new developments as possible.

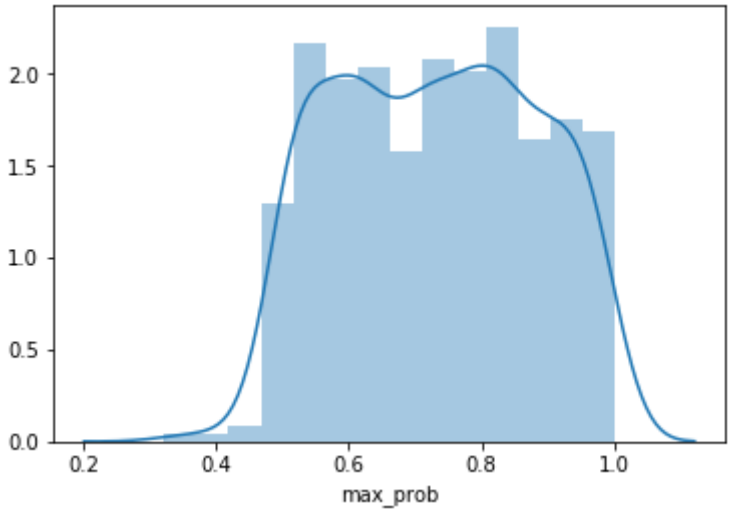

Figure 13 - Distribution of probabilities.

4 RESULTS

4.1 METRICS

In order to evaluate the performance of our model, we chose to analyze it by calculating the accuracy, precision, and recall. Besides these metrics, we also created a confusion matrix.

For the following formulas, we denote *tp* as true positives which are the number of positive instances correctly classified, *tn* as true negatives, which are the number of negative instances correctly classified.

On the other hand, we use *fp*, false positives, to denote the number of positive instances wrongly classified and *fn*, false negatives, to denote the number of negative instances wrongly classified.

The accuracy measures the percentage of correct predictions in the total amount of predictions. The formula is as follows[27]:

$$\frac{tp + tn}{tp + fn + tn + fp}$$

The recall measures how many of the positive cases we correctly classified in the class. This means that the recall of 'Apartment' is how many listings of that class are correctly classified in the dataset divided by how many listings we have of that class. The formula for this metric is[27]:

$$\frac{tp}{tp + tn}$$

It is important to note that having this metric as 1 does not mean our model is good. If our model classifies everything as an apartment, the recall for this class will be 1 whereas in the other classes it will be 0.

The precision complements the recall metric. This metric measures how precise our model is. In the above example our precision would be low given that this metric takes into account false positives, so if we classifying everything as an apartment, the number of false positives will be high. The formula is[27]:

$$\frac{tp}{tp + fp}$$

Finally, we will be using a confusion matrix to understand if there is a pattern to the errors of our model. This can be a useful tool if the model often mistakes one class for another. It can give us a hint of which features to check. The confusion matrix has the following structure:

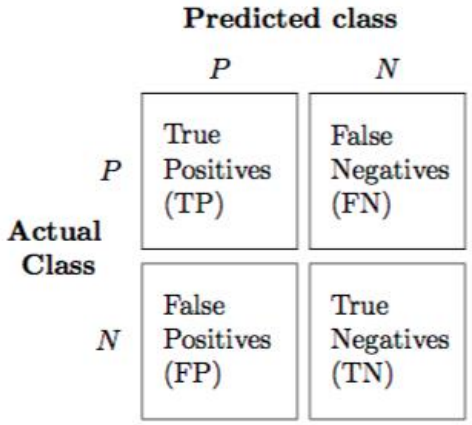

Figure 14 - Confusion Matrix Structure

4.2 EVALUATION

After training our model, we must predict new occurrences. Our test dataset had 17834 listings, 9044 apartments, 4408 houses, 1424 new developments, 1331 plots, 1018 investments and 609 other.

If we predict without the 0.8 rule for new developments, we get an accuracy of 93%, if, on the other hand, we apply the rule, our accuracy increases to 95.7%.

By analyzing the confusion matrix in Figure 15 we can see that even though we implemented the threshold, we still get 345 false positives. This can be further reduced by increasing the threshold, however, it depends on what we prioritse for the model, correctly classifying as many as possible or having few false positives. If we opt to have a lower amount of false positives we will decrease our recall but increase our precision.

Since the listings for the other classes do not have the same issues as new developments (as discussed in the introduction), we are able to achieve higher precision and recall scores.

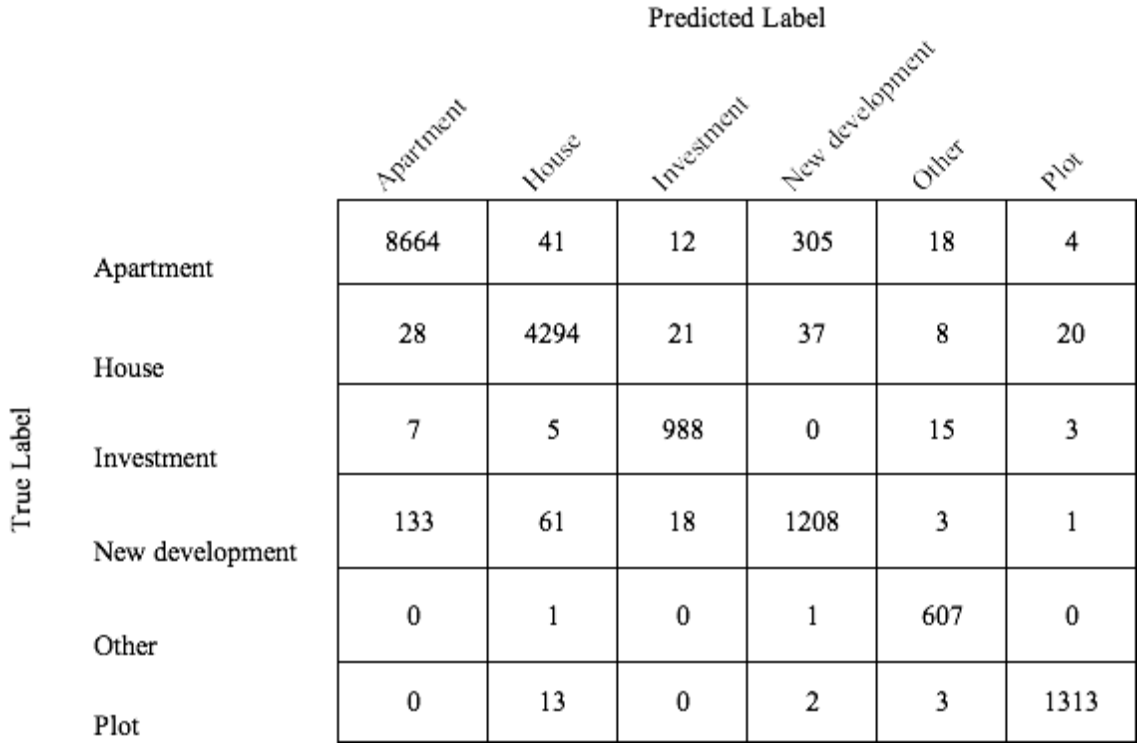

Figure 15 - Confusion Matrix

Table 3 - Table of Metrics

	Precision	Recall	Number of listings
Apartment	0.98	0.95	9044
House	0.97	0.97	4408
Plot	0.98	0.98	1331
Other	0.92	1	609
Investment	0.96	0.98	1018
New Development	0.76	0.88	1424

4.3 FEATURE IMPACT

In order to better understand the results, we got, we used the package 'Shap' to produce graphs that show the importance of each feature in classifying each class.

The graphs obtained for 'Apartment'(Figure 16), 'Investment' (Figure 20), 'Plot' (Figure 17) and 'House'(Figure 21) classifications are the expected output, however we can see that when predicting ' Other' (Figure 18), the features apartment title and apartment subtype are high in importance, which can help explain the missed apartment as 'Other'.

We can also see in Figure 19 that for new developments the features which carry the most weight in the decision are the number of keywords in the title and in the first part of the description, the price and whether the listing is for rent or not (Figure 19). The fact that the price and the rent status have high importance can help us justify the number of listings that are classified wrongly as new development.

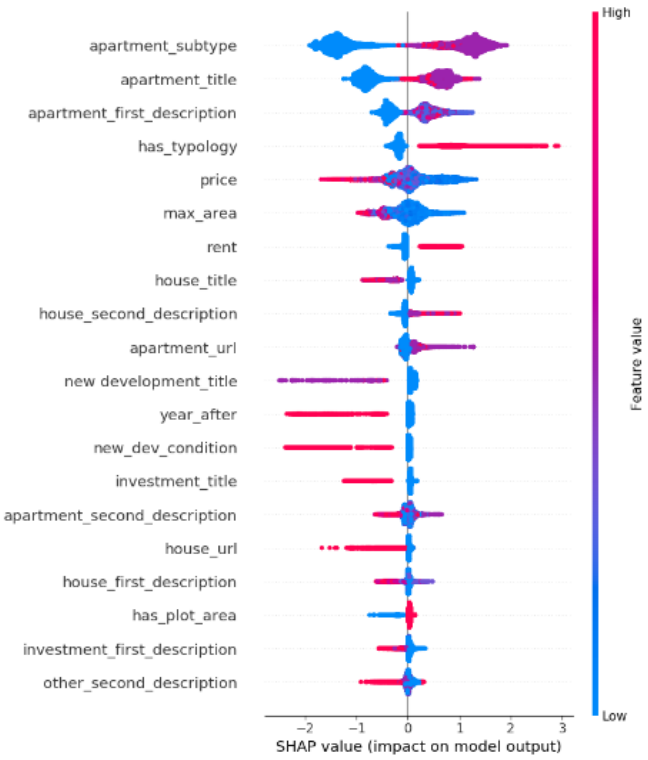

Figure 16 - Apartment Feature Importance.

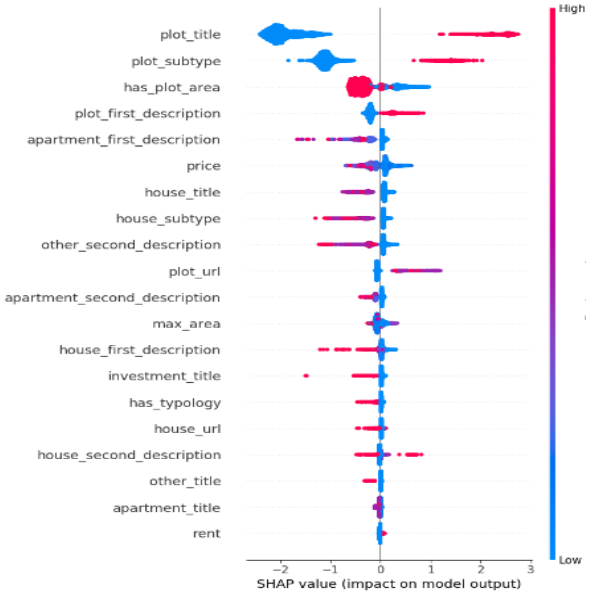

Figure 17 - Plot Feature Importance.

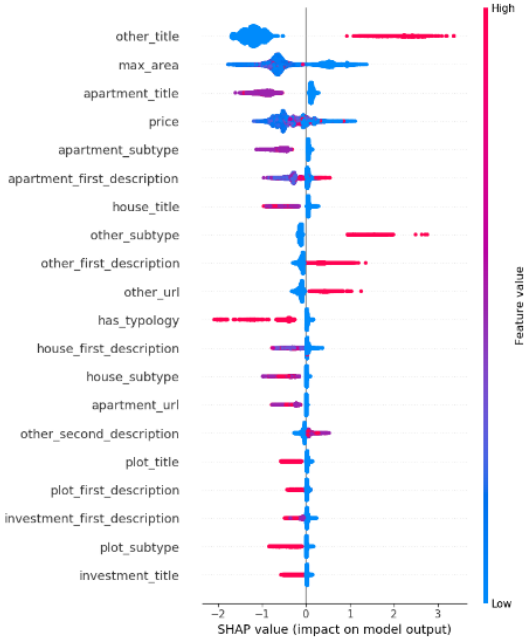

Figure 18 - Other Feature Importance.

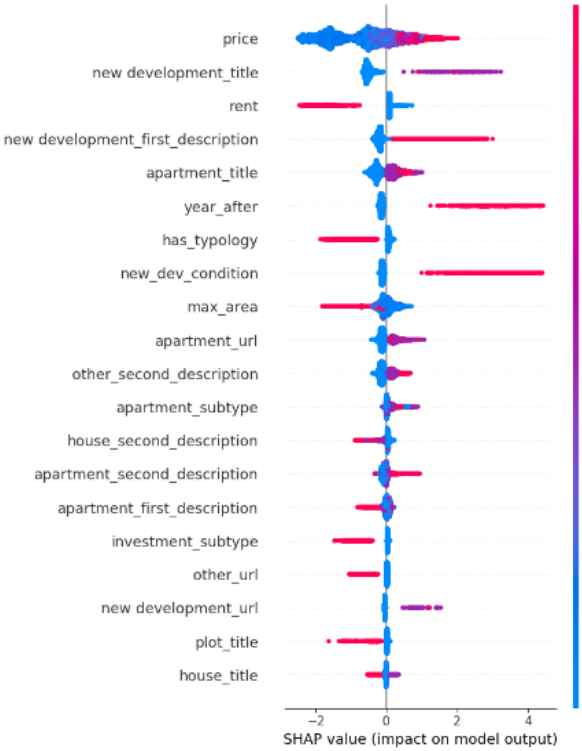

Figure 19 - New Development Feature Importance.

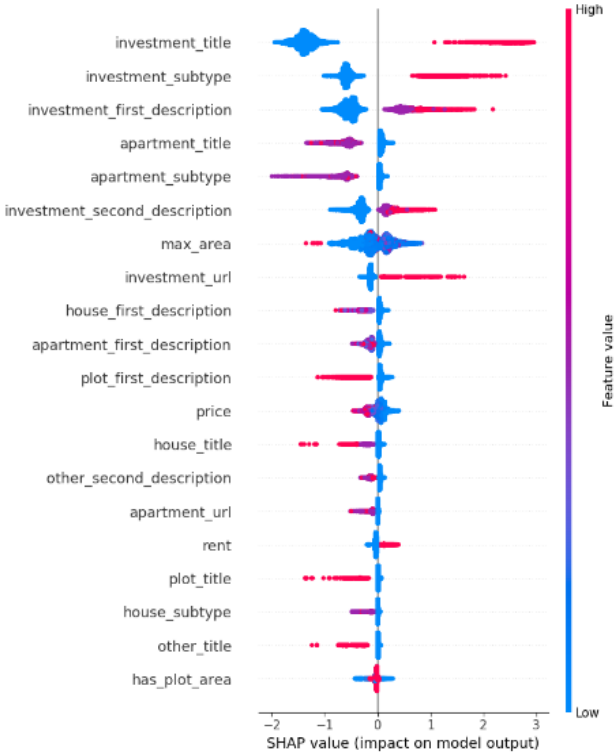

Figure 20 - Investment Feature Importance.

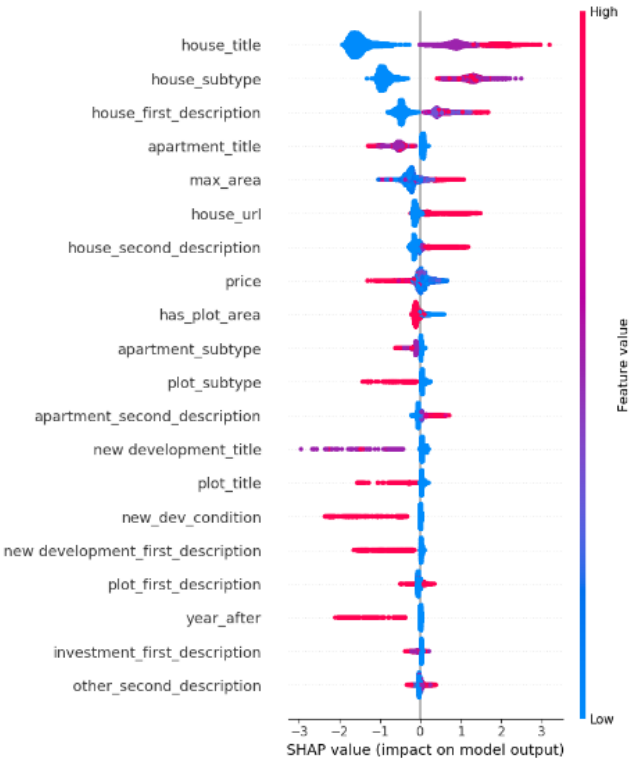

Figure 21 - House Feature Importance.

Overall we can see that for apartments, houses, investment and plots the features that are deemed the most important (apartment keywords in the subtype, title and description for apartment, house keywords in the title subtype and description for houses, plot keywords in the subtype and title for plots and investment keywords in the title and subtype for investments) make sense in an user perspective. However, the mismatch between our expectations and the outcome of the feature importance for the other and new development classes can indicate an issue with our training set (samples that sould be set as apartment are set as other) or erroneous parameters.

4.4 Model Monitoring

Having the model assign a prediction is not enough in the long run. We must be able to keep track of the certainty of the model as well as what it takes into consideration when making the prediction.

In order to keep track of this we will use the probabilities outputted by the XGBOOST model as well as the output of the shap class used above. This way we can log how certain the model was in the prediction and which features it considered the most important.

To effectively monitor the performance of the model, some samples must be randomly selected and sent for human evaluation. In this case we can use samples where the probability for each of the classes is low and after being labelled, we add this sample to a dataset. If we start finding several cases where the model fails to predict the correct label, we can retrain it.

Besides allowing us to monitor the performance of the model, we can also detect edge cases not caught in the initial analysis, detect new cases that might arise and finally, aid in debugging why the model made some predictions.

5 CONCLUSION

Automatic tagging of real estate listings is a necessity when extracting listings from several different sources. When developing a product that offers appraisals, a model that reliably assigns its type is fundamental.

In this project we showed that there is no need to have an extensive data preprocessing pipeline, or a complicated black box algorithm to have adequate results. This type of approach reduces overhead (lower computing power necessary, pipeline easier to understand to newcomers) and is sufficiently fast for real-time tagging.

By using a list of keywords to search in, we eliminate the need for a complex pipeline where we would need to tag the part of speech and remove stopwords. Since the list of words can contain multiple languages, we also eliminate the need of creating a model for each language or a step in the preprocessing phase where we would identify the language and use the appropriate stopwords and part of speech tagging.

By implementing a threshold for the probability of the new developments class we decrease the amount of false positives for that class given that we are more conservative when predicting. This implementation improved our model from 93% to 95%.

Overall we managed to get good results across the board, we got an accuracy of 95% on all classes and a precision of 98% in apartments, a precision of 97% in houses, a precision of 96% in investments, a precision of 98% in plots and a precision of 92% in other. The category where we obtained poorer results was in new developments, where we got a precision of 76%. In this category is also interesting to look at the recall of 88%. This means we are able to identify 88% of new developments present in our database, which means that if we have 2000000 new developments in our database we will only identify 1760000, so the customers will still be able to find 240000 wrongly classified listings.

Another issue we avoid with this approach is the size of some descriptions being insufficient. If the author of the listing did not use the property category enough in the description we would find occasions where another non-important word was selected by the model.

We considered creating one model to predict each category and then choosing the category which had the highest percentage of certainty. This proved to be an incorrect approach since the accuracy gain was marginal and the complexity of the package increased substantially given that instead of having to provide support for just one model we would have to provide support for six.

In order to provide some easier interpretability of the model, we decided that besides providing the prediction we will also provide the probability of each class, and the calculated features. This way when we need to debug a prediction it will be much easier and faster to diagnose where it went wrong.

Despite the accuracy presented by the algorithm, some limitations are present. The dictionary-based approach, while easy and fast to implement, requires the insertion of new keywords whenever new languages or types are parsed.

Trying to identify new developments is also a limitation, given that sometimes it lacks keywords for that type and can make the classifier wrongly categorize normal listings. To improve this limitation, we recommend the introduction of an image classifier algorithm, however, it will make the whole process last longer and consume more resources.

A new type of listing has been emerging in the last few years in order to appeal to students: rooms for rent. Using this approach, it is not possible to identify ‘rooms’ as a type since the word ‘rooms’ usually is used to inform the buyer how many rooms a house has and not necessarily that only a room is for rent.

This project has shown that we don’t always need a complex model to solve text categorization problem. This approach reduces besides being more understandable for business users, has lower costs given that it requires less computing power to train and predict. If we were to choose a more complex model with tensorflow or any other neural network we would be increasing train and prediction times and thus computing power.

Given that the type of websites that this model benefits are relatively new in the market, this project presented a proof of concept for what can be achieved with a simple model, proving that sometimes we don’t need a state of the art program to obtain good results.

6 BIBLIOGRAPHY

[1] A. Baldominos, I. Blanco, A. J. Moreno, R. Iturrarte, Ó. Bernárdez, and C. Afonso, "Identifying real estate opportunities using machine learning," *Appl. Sci.*, vol. 8, no. 11, 2018.

[2] "Data Science at Zillow - Zillow Tech Hub." [Online]. Available: https://www.zillow.com/tech/data-science-overview-2017/. [Accessed: 07-Dec-2019].

[3] "What is a Zestimate? Zillow's Zestimate Accuracy | Zillow." [Online]. Available: https://www.zillow.com/zestimate/. [Accessed: 09-Dec-2019].

[4] V. Del Giudice, P. De Paola, and F. Forte, "Using genetic algorithms for real estate appraisals," *Buildings*, vol. 7, no. 2, pp. 1–12, 2017.

[5] "idealista/data — idealista." [Online]. Available: https://www.idealista.pt/data/. [Accessed: 29-Aug-2019].

[6] V. Gupta and G. S. Lehal, "A Survey of Text Mining Techniques and Applications - Volume 1, No. 1, August 2009 - JETWI," *J. Emerg. Technol. Web Intell.*, vol. 1, no. 1, pp. 60–76, 2014.

[7] D. Sarkar (2006), *Text Analytics with Python :A Practical and Real-World Approach and to and Gaining Actionable and Insights from and Your Data.* .

[8] R. Feldman, "Techniques and applications for sentiment analysis," *Commun. ACM*, vol. 56, no. 4, p. 82, 2013.

[9] Y. Zhang, R. Jin, and Z. H. Zhou, "Understanding bag-of-words model: A statistical framework," *Int. J. Mach. Learn. Cybern.*, vol. 1, no. 1–4, pp. 43–52, 2010.

[10] S. Menaka and N. Radha, "Text classification using keyword extraction technique," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 3, no. 12, pp. 2277–128, 2013.

[11] E. Brill, "A Simple Rule-Based Part of Speech Tagger," *Proc. of the 3rd Conf. on*

Applied Natural Language Processing, pp 152–155, 1992

[12] "5 Text Analytics Approaches: A Comprehensive Review." [Online]. Available: https://getthematic.com/insights/5-text-analytics-approaches/. [Accessed: 17-Feb-2020].

[13] H. V. Cook and L. J. Jensen, "A guide to dictionary-based text mining," *Methods Mol. Biol.*, vol. 1939, pp. 73–89, 2019.

[14] "Text Classification: a comprehensive guide to classifying text with machine learning | MonkeyLearn." [Online]. Available: https://monkeylearn.com/text-classification/. [Accessed: 29-Aug-2019].

[15] "Difference Between Supervised, Unsupervised, & Reinforcement Learning | NVIDIA Blog." [Online]. Available: https://blogs.nvidia.com/blog/2018/08/02/supervised-unsupervised-learning/. [Accessed: 30-Jan-2020].

[16] "Regression vs. Classification Algorithms | Oracle Data Science." [Online]. Available: https://blogs.oracle.com/datascience/regression-vs-classification-algorithms. [Accessed: 30-Jan-2020].

[17] T. Mitchell, *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1997.

[18] "Decision Trees Explained Easily." [Online]. Available: https://medium.com/@chiragsehra42/decision-trees-explained-easily-28f23241248. [Accessed: 26-Jan-2020].

[19] L. Breiman, "Random forests," *Machine Learning* 45, pp 5–32 (2001).

[20] "Random Forests®, Explained." [Online]. Available: https://www.kdnuggets.com/2017/10/random-forests-explained.html. [Accessed: 25-Jan-2020].

[21] J. Friedman, "Greedy Function Approximation: A Gradient Boosting Machine," *Notes Queries*, vol. s5-VII, no. 169, p. 236, 1877.

[22] H. A. Le Thi, V. V. Nguyen, and S. Ouchani, "Gene selection for cancer

classification using DCA," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5139 LNAI, pp. 62–72, 2008.

[23] "sklearn.feature_extraction.text.CountVectorizer — scikit-learn 0.21.3 documentation." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html. [Accessed: 29-Jul-2019].

[24] J. Bennett, S. L.-P. of K. cup and Workshop, and U. 2007, "The Netflix Prize," *Cs.Uic.Edu*2007, .

[25] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, vol. 13-17-Augu, pp. 785–794, 2016.

[26] X. He *et al.*, "Practical lessons from predicting clicks on ads at Facebook," *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2014.

[27] M. Hossin and N. Sulaiman, "A Review on Evaluation Metrics For Data Classification Evaluations," *Int. J. Data Min. Knowl. Manag. Process*, vol. 5, no. 2, pp. 1–11, 2015.