

**Airbnb Pricing Model**

**Machine Learning and Content Analytics**

Final Project

**Students:** Ioannis Dekoulakos, Anna Makropoulou**,** Anastasios MoraitisAnastasios Safras

*Academic Year 2021 - 2022*

*Part-Time Program*

**Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα**

Contents

[Abstract 3](#_Toc113194206)

[1. Introduction 4](#_Toc113194207)

[Background 4](#_Toc113194208)

[Setting the Scene 4](#_Toc113194209)

[Our Mission 5](#_Toc113194210)

[2. Data Collection 5](#_Toc113194211)

[3. Models 8](#_Toc113194212)

[Model I: Price Prediction 8](#_Toc113194213)

[Descriptive Analysis and Data Exploration 10](#_Toc113194214)

[Methodology Set Up – Algorithm 18](#_Toc113194215)

[Model II: Using Tensorflow to Recommend Price Adjustment 30](#_Toc113194216)

[Dataset Overview 30](#_Toc113194217)

[Data Pre-processing 30](#_Toc113194218)

[Methodology Set Up – Algorithm 31](#_Toc113194219)

[LSTM Modeling 39](#_Toc113194220)

[4. Business Value and Further Discussion 41](#_Toc113194221)

[5. Members – Roles 42](#_Toc113194222)

[6. Time Plan – Project Roadmap 43](#_Toc113194223)

[References 43](#_Toc113194224)

# Abstract

Nowadays, many Airbnb hosts underestimate their properties and do not know how to maximize their profits. Airbnb is an American company that operates an [online marketplace](https://en.wikipedia.org/wiki/Online_marketplace) for lodging, primarily [homestays](https://en.wikipedia.org/wiki/Homestay) for vacation rentals, and tourism activities. Based in San Francisco, California, the platform is accessible via website and [mobile app](https://en.wikipedia.org/wiki/Mobile_app). Airbnb does not own any of the listed properties; instead, it profits by receiving [commission](https://en.wikipedia.org/wiki/Commission_(remuneration)) from each booking. The company was founded in 2008 by Brian Chesky, Nathan Blecharczyk, and Joe Gebbia. Airbnb is a shortened version of its original name, AirBedandBreakfast.com.

Εικόνα που περιέχει κείμενο, διαφορετικός, διάφορα, πολύχρωμος

Περιγραφή που δημιουργήθηκε αυτόματα

Airbnb is a platform where anyone can list a property for short-term rentals and at the same time individuals from across the globe, can book a property. It operates in more than 200 countries and has attracted so many fans and loyal recurrent customers that in many cases has overtaken the respective hotel industry, with red-hot growth. So, it is obvious that the competition has increased, and It has become a necessity for hosts to follow a more professional and smart management, especially in the pricing part. By observing this tendency in the market, the aim of this project is to build a service that will help Airbnb hosts get an estimate for the price of their property so that they would be able to maximize their profits or alternatively help novice hosts decide if they will eventually convert their property to an Airbnb listing, and at what price. The project focuses on the Greek market as the data collected are related solely to it.

# Introduction

## Background

Airbnb is a short-term rental service that allows everyone to list and book accommodations. Airbnb was founded in 2008 and the initial idea was that anyone that wanted extra income, could rent a room in his house to a traveler at a more competitive price than a hotel room. Later on, most users were able to choose from renting an entire place including flats, villas, beach houses etc. Within a decade, Airbnb managed to grow rapidly and now it is considered as the largest platform of accommodation around the world. In fact, Airbnb operates as of June 30 2021, in more than 220 countries around the world.

All in all, Airbnb is a web platform accessible via site and mobile application which brought huge changes in the way people can travel worldwide. It introduced a revolutionary peer-to-peer platform of accommodation, in which anyone can host guests or book a place to stay instantly. The opportunity that travelers have, to experience a stay like locals do, the direct communication with the owner and the reduced amount of money required for lodging larger spaces are only a few of the reasons why Airbnb has become a success story in the industry.

## Setting the Scene

Since most of the Airbnb hosts are users with no prior expertise in the travel and real estate industry, one of the important challenges that face, is how they will ultimately decide the listing price of their rental or if their initial assumptions as far as the pricing scheme decide, is the optimal. Even though Airbnb suggests a base-line price for the property, based on some basic criteria which among them are the number of bedrooms, bathrooms and location, the final price suggested is nothing more than a low average price of other properties with similar features nearby (location-wise). As a result, the final booking price of the rental may not be the most competitive one or in many cases the most realistic one, as only a few factors are taken into consideration, with them being mostly quantifiable features like the number of bedrooms or bathrooms as mentioned above, without focusing equally on rich attributes like the view, high ceilings or the existence of fireplace etc. and text attributes like the name and description of the listing. Therefore, the need for creating a pricing-scheme leveraging the power of the text and visual attributes offered along with the rest information accompanying each rental, could:

* Improve the final price suggestions,
* upgrade the user experience
* and in many cases boost SEO for home listings and travel websites.

## Our Mission

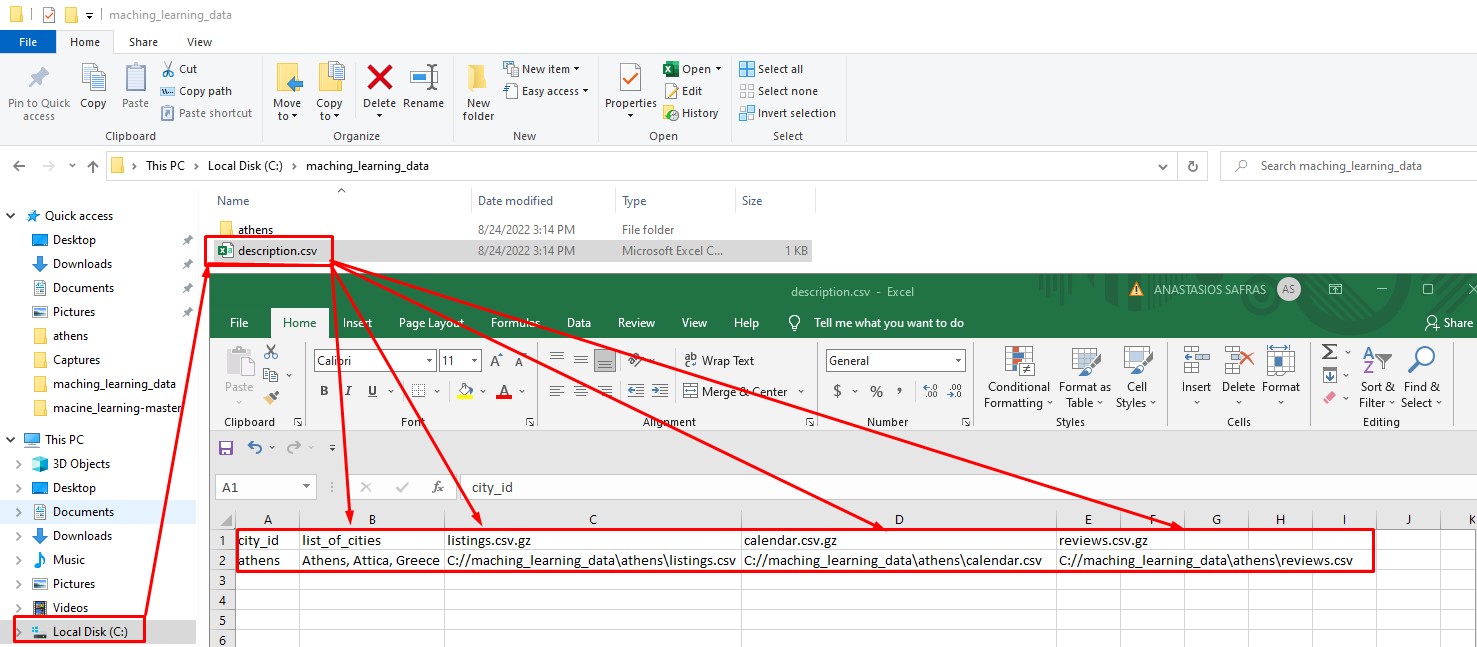
The project aims to build an easy-to-use platform leveraging our custom machine learning models built with data from the Greek Airbnb market. The purpose of this service is to offer the end-user a range of suggested pricing segments based on the criteria set as input. Such criteria are the name, description, location, the number of bedrooms, photos of the rental and other features. Consequently, any user expressing an interest to upload a listing on Airbnb can be assisted with a pricing model that will consider more criteria than the basic Airbnb’s pricing model or other models in the market in order to improve or verify their initial assumptions regarding the rental’s price. Moreover, the users will be benefited from this service in terms of having a clear pricing view and an estimator for the investment of their property, thus they can take into consideration more data that will help them decide whether they want to proceed with the Airbnb idea. In addition, the user would be able to choose among different pricing strategies for their listing, each one of them will result in a different forecast for a potential income. The pricing segments offered from our model *verify or suggest* an update on the price the owner sets as input. For example, for an initial price which leads to characterizing a property as expensive the model could suggest an update to the average pricing segment. Thus, the user should reconsider changing the initial assumed price to a more appropriate one. As a result, the pricing segments offered that match or need to be updated are the following:

* Expensive
* Average
* Cheap

# Data Collection

Initially the data collection was performed. The data were retrieved from the following url: http://insideairbnb.com/get-the-data.html. We downloaded data only for Athens, Greece and the columns kept refer to each property’s attributes. For example, some attributes that were considered are property id, name, description, bedrooms, latitude, longitude etc. Also, the null values were defined as “unknown”. The entire code defining for which attributes of each property id and for which city, data should be downloaded, can be found in the constants.py file.

After defining in the constants.py file from were and what data should be downloaded, the data acquisition took place, in order for them to be stored in the directory contained in the constants.py file. Furthermore, when the data acquisition process is completed a csv file is created, named descriptions.csv. The description.csv contains the columns city\_id, list\_of\_cities, listings.csv.gz, calendar.csv.gz, reviews.csv.gz as shown below:



Then in file “athens” the 3 separate csv files containing the listings, the calendar and the reviews can be found. The listings file contains information on each property, defined by a unique id , as presented below:

Εικόνα που περιέχει πίνακας

Περιγραφή που δημιουργήθηκε αυτόματα

The calendar file contains daily information related to the availability of the listing, as follows:

Εικόνα που περιέχει πίνακας

Περιγραφή που δημιουργήθηκε αυτόματα

Finally, the reviews file refers to the reviews of each listing made by past guests along with the free text of the entire review, which is presented below:

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

# Models

# Model I: Price Prediction

Importing the attributes of Airbnb listings created a data frame which has 56 attributes and 10911 observations (columns and rows as well).

Upon importing the dataset, actions regarding changes in data types of variables, exclusion of specific columns without valuable information, as well as merging some levels of categorical variables were considered. All performed transformations will be analyzed below.

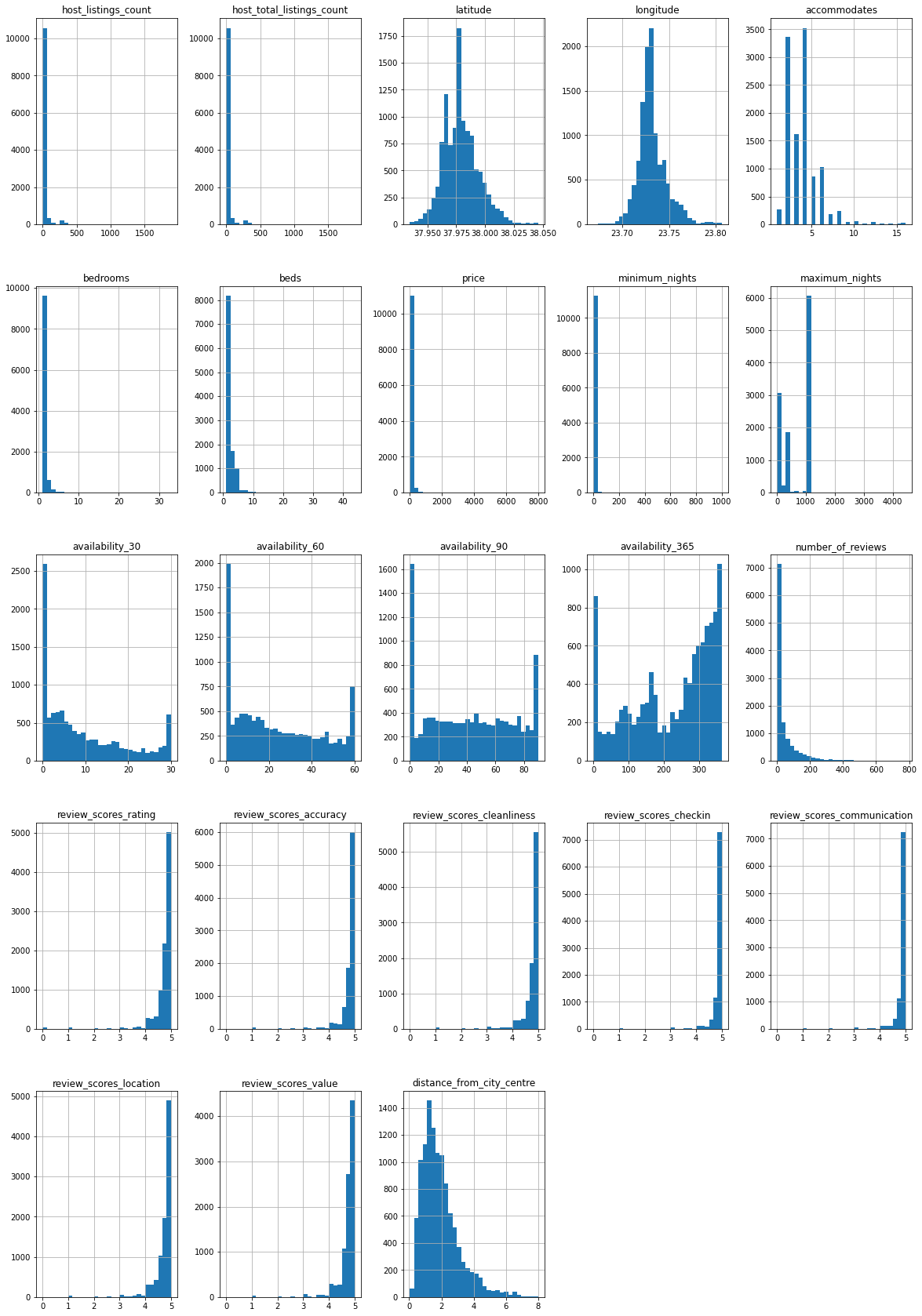
A general Overview of the Dataset, after removing the variables not taken into consideration.  
  


Figure 1: Histograms of all the numeric variables in the dataset

## Descriptive Analysis and Data Exploration

We will keep a logical range 20-500 euros, since the extreme prices seems to be not representative for the area:

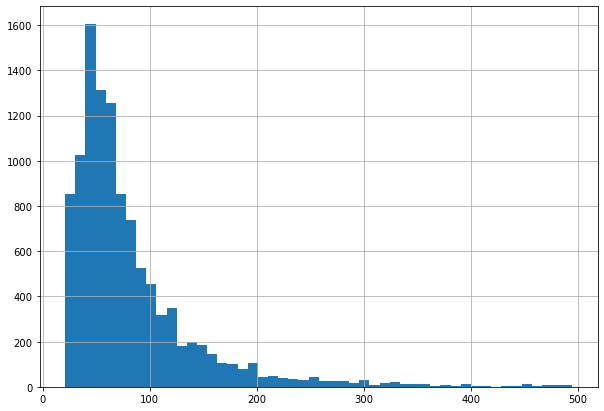


Figure 2: Histogram of Price

The Nightly advertised prices range from euros 21 to euros 495 euros. As expected, most property prices seem to be in the range 40-70 euros per day.

***How have prices changed over time?***

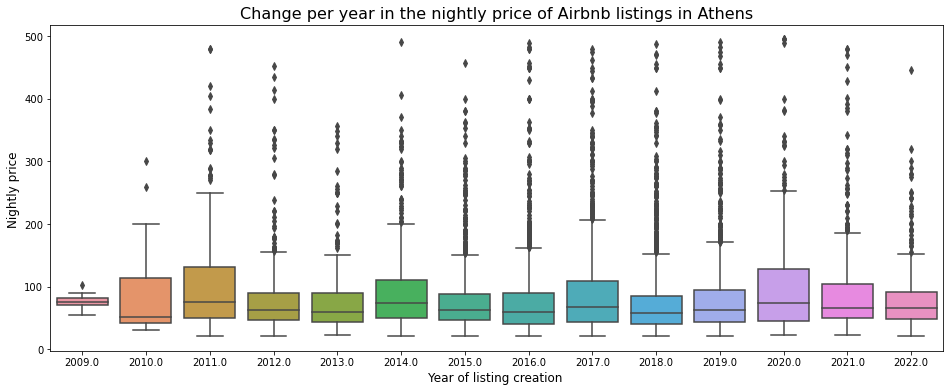


Figure 3: Box-plots for the nightly Price over the years

The average price per night for Airbnb listings in Athens seems to be higher in years 2011, 2014 & 2020. After 2020, it was expected to see a reduction in prices due to Covid-19, flights and tourism to our country decreased significantly.

In the following graphs the Average number of people accommodated are presented:

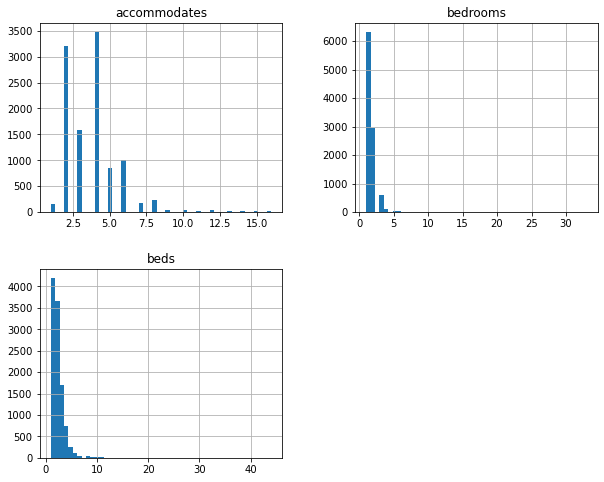
****

Figure 4: Bra-plots for the Average number of people accommodated

The most common property setup sleeps until 4 people and has 2 or 3 bedrooms.

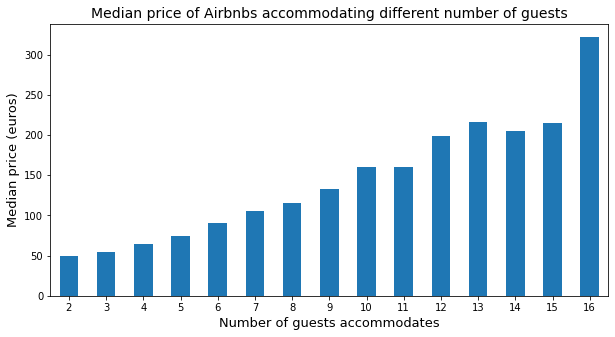
**

Figure 5: Median Price per number of Guests Accomodated

As expected, the more people a house accommodates, the more expensive its price is.

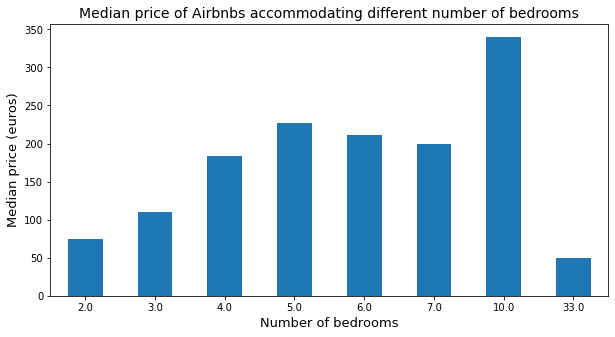


Figure 6: Median Price per Number of Bedrooms

The number of bedrooms has a significant effect on the price, it is noteworthy that a listing with 33 bedrooms has the lowest price. This makes sense, most likely this listing is a big old hotel or a hostel.

Hosts-Reviews-location

1. Hosts

***How long have hosts been listing properties on Airbnb in Athens?***

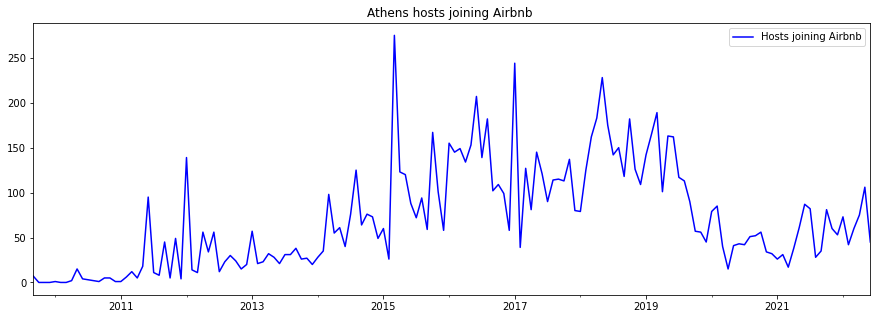
****

Figure 7:Yearly Evolution of Hosts Joining Airbnb

The oldest Greek listing that is currently live on Airbnb was first listed on the site around 2009. From 2011 onwards, the number of listings started increasing considerably.

We can see that there is a big peak in the number of hosts joining Airbnb in 2015. The Airbnb had become famous in Greece as well, encouraging more and more people to take advantage of their homes with a short-term rental.

1. Reviews

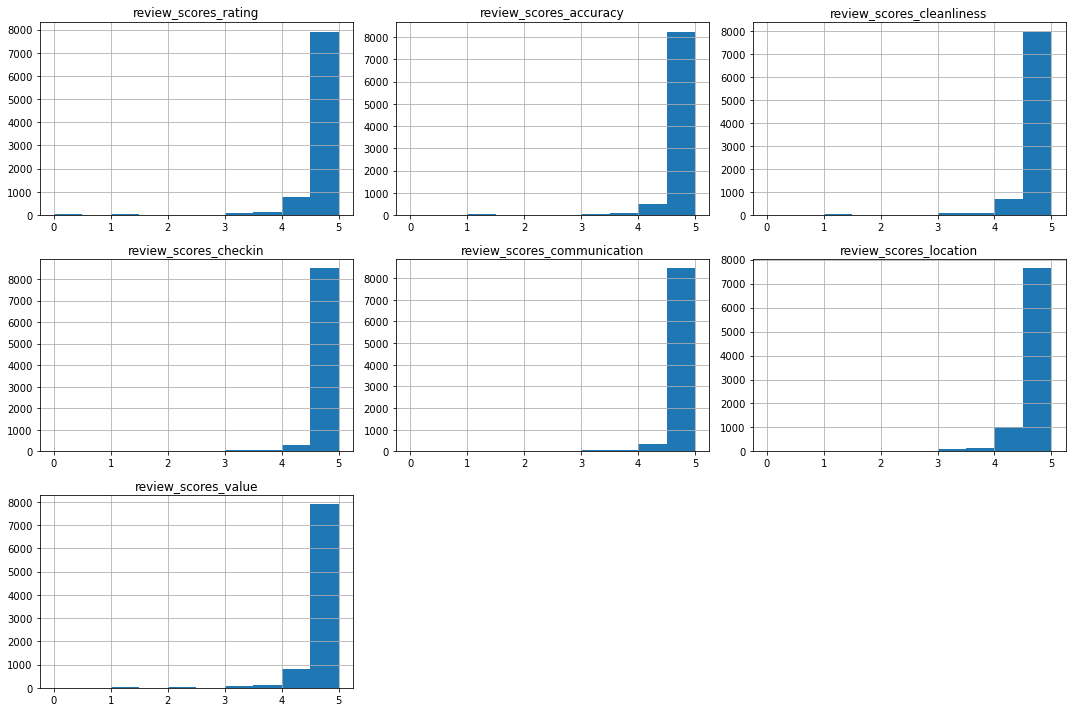


Figure 8: Reviews Score Rating

It seems that the highest percentage of scores is close to 5, something that conforms to the standards of Greek hospitality

1. Location

* the colour represents the price
* the size of the circle represents the number of reviews (so, the occupancy)

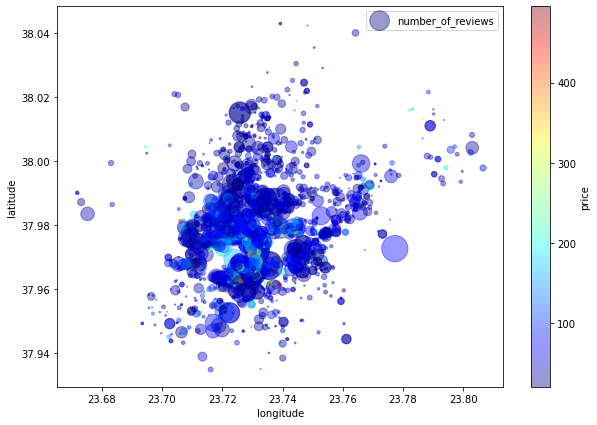


Figure 9: Volume of Reviews per Location

More central listings seem to have better occupancy at higher prices.

1. Amenities

We kept some important variables and turned them into dummy variables:

* Breakfast
* check\_in\_24h
* air-conditioning
* balcony
* nature\_and\_views
* bed linen
* TV
* coffee\_machine
* white\_goods
* elevator
* parking
* outdoor\_space
* hot\_tub\_sauna\_or\_pool
* internet
* pets\_allowed
* Essentials
* Kitchenette
* Pool
* Self-check-in
* Stove
* *heating*

We compared the proportions of these features that are true or and the median price of each category, exploring the relationship between the category and price.

***Hosts vs Super Hosts***

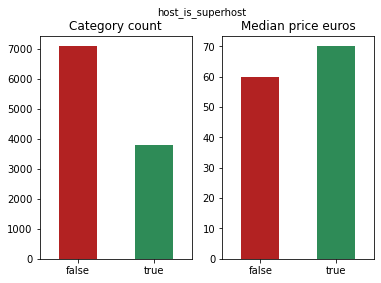


Figure 10: Hosts vs Super Hosts

About 35% of hosts are super hosts which improves the median price per night of their Airbnb listings.

***Pets allowed vs Not allowed***

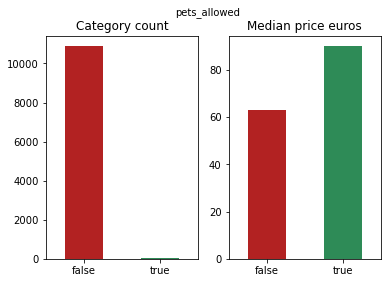


Figure 11: Pets Allowed

Only 2% of hosts allow pets, but this seems to significantly affect the price.

***General Amenities***

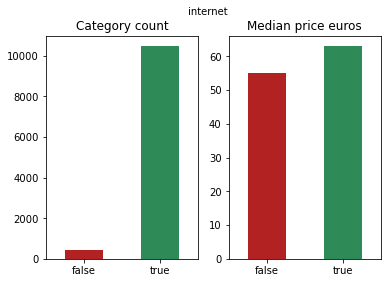


Figure 12: General Amenities

The offer of general amenities per listing does play a significant role in the variation of the median price as it does not differ notably form the median price of listings ,which do not offer general amenities.

***Nature and Views***

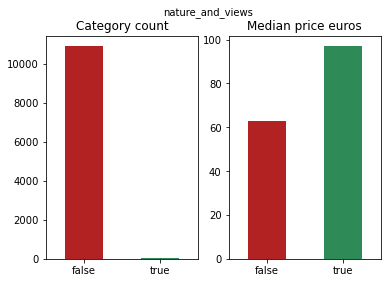


Figure 13: Nature and Views

Whether a rental offers a view or view, this does not seem to affect the median price.

***Internet***

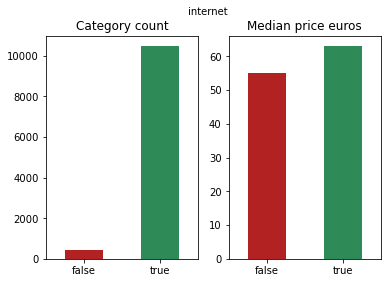


Figure 14: Internet

Only a small number of listings does include internet but similarly as the previous amenities offered this does not seem to affect the median price. The same stands for the price of rental including heating system.

***Heating***

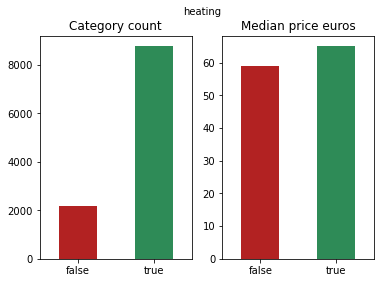


Figure 15: Heating

## Methodology Set Up – Algorithm

We built three different models in order to predict the price variable.

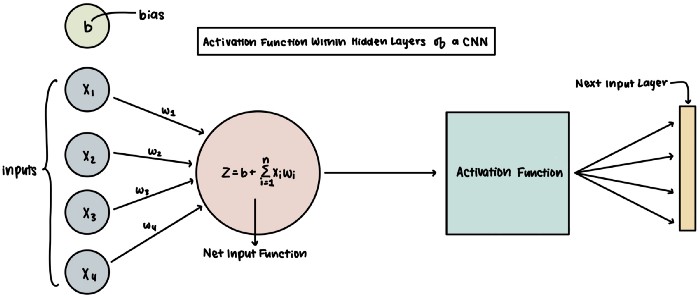
All 3 models have the same number of layers, but different hyperparameters. We distinguish those that use a different optimizer:

1. Optimizer = Adam, loss function = Mean Absolute Error, epochs = 250, batch\_size = 128, Dense layers(>1 neurons) activation function: selu
2. Optimizer = SGD, loss function = Mean Absolute Error, epochs = 250, batch\_size = 128
3. Optimizer = Ftrl, loss function = Mean Absolute Error, epochs = 450, batch\_size = 6

Before we move forward it is important that we understand the role of activation and loss functions.

***Activation Functions***

Activation functions map a neuron’s inputs to that of the next layer. In doing so, deep learning architectures are designed to tackle non-linear problems, which are much harder to solve than linear ones.

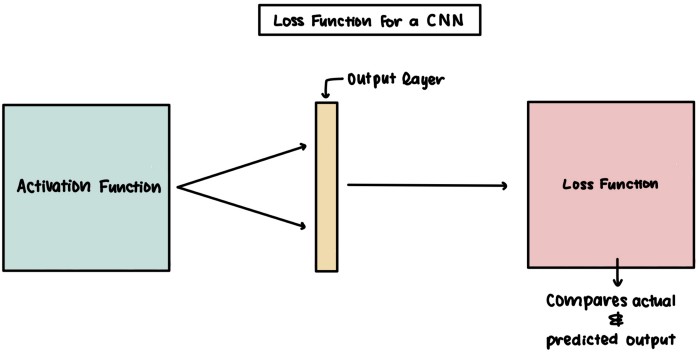


In this example, a neuron receives four inputs, each having a weight to determine its importance within the network. The net input is the summation of the input-weight products with the bias subsequently added.

Afterward comes the activation function, which will make the final call as to what gets passed onto the next input layer. While there are many different activation functions, here are some commonly used ones.

***Loss Functions***

Loss functions are synonymous with “cost functions” as they calculate the function’s loss to determine its viability.



Above we have an activation function outputting probabilities for the output layer. By comparing the predicted values and the actual answer, we can calculate a model’s accuracy and update the weights throughout our network. For example, let’s say that a Convolutional Neural Network predicts that an image has a 30% chance of being a cat, 10% chance of being a frog, and 60% chance of being a horse. If the actual label for the image were a cat, then there would be a very high loss.

Choosing the best loss function is crucial, considering that it is the best indicator of whether a model is over-or-under fit.

We used MAE instead of MSE, which places a more significant penalty in comparison to MAE when the difference between the predicted and actual value increases.

**1st Model Using Adam Optimizer**

Adam is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data. the algorithm calculates an exponential moving average of the gradient and the squared gradient, and the parameters beta1 and beta2 control the decay rates of these moving averages.

The initial value of the moving averages and beta1 and beta2 values close to 1.0 (recommended) result in a bias of moment estimates towards zero. This bias is overcome by first calculating the biased estimates before then calculating bias-corrected estimates.

The following output of the first model contains information on the parameters, the dense layers and the dropout rates:

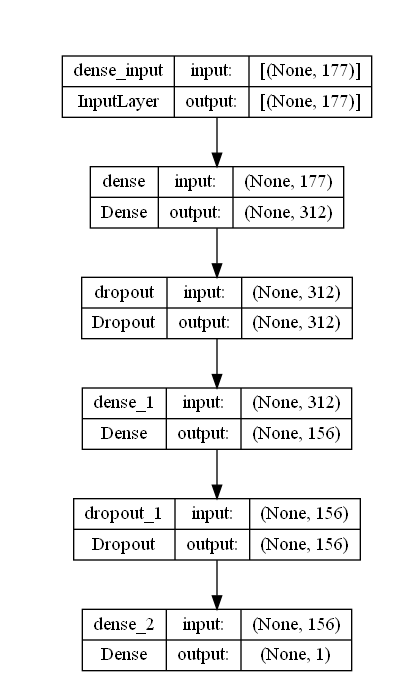
* **Parameters** (params): are the weights and biases that will be used for computation in all neurons of the CNN.
* **Dense Layer**: A dense layer is just a regular layer of neurons in a neural network. Each neuron receives input from all the neurons in the previous layer, thus densely connected. The layer has a weight matrix **W,**a bias vector **b,**and the activations of previous layer **a.**
* Dropout is a technique used to tackle overfitting. Dropout consists in randomly setting a fraction rate of input units to 0 at each update during training time, which helps prevent overfitting. Dropout is a regularization technique, which aims to reduce the complexity of the model with the goal to prevent overfitting.

The Mean Absolute Error for the validation and test data is significantly low as shown in the output below:

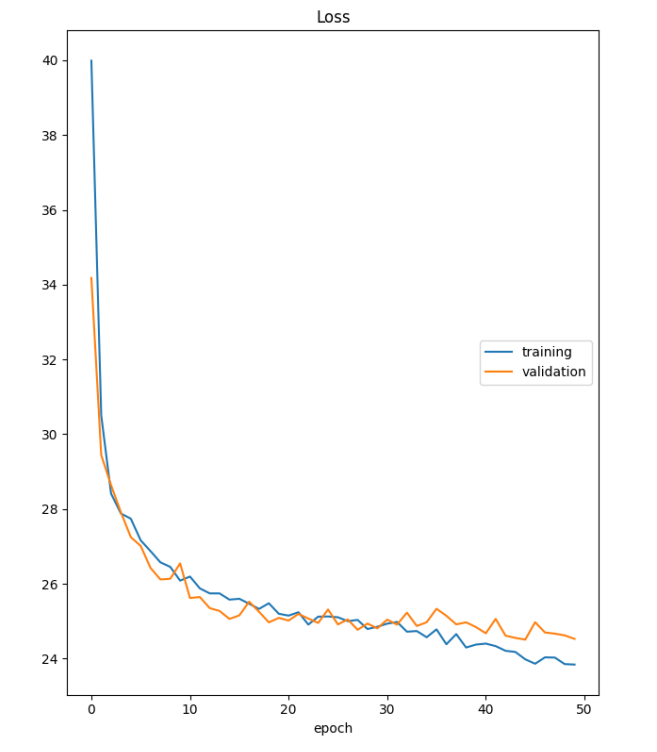
Εικόνα που περιέχει πίνακας

Περιγραφή που δημιουργήθηκε αυτόματα

The plot of the model built using the Adam optimizer, trained for 250 epochs is the following:



The loss function of the 1st model demonstrating how well it performs in the absence of overfitting problems:



**2nd Model Using SGD Optimizer**

SGD algorithm is an extension of the Gradient Descent and it overcomes some of the disadvantages of the GD algorithm. Gradient Descent has a disadvantage that it requires a lot of memory to load the entire dataset of n-points at a time to compute the derivative of the loss function. In the SGD algorithm derivative is computed taking one point at a time.

The advantage of SGD is that memory requirement is less compared to the GD algorithm as the derivative is computed taking only 1 point at once. Some disadvantages are:

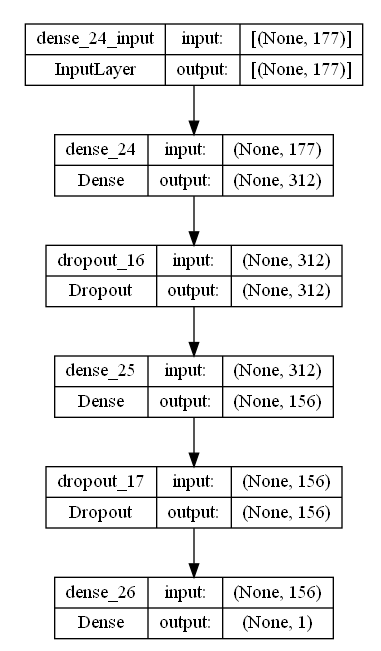
* The time required to complete 1 epoch is large compared to the GD algorithm.
* Takes a long time to converge.
* May stuck at local minima.

The model built using the SGD Optimizer performs equally well, with low MAE:

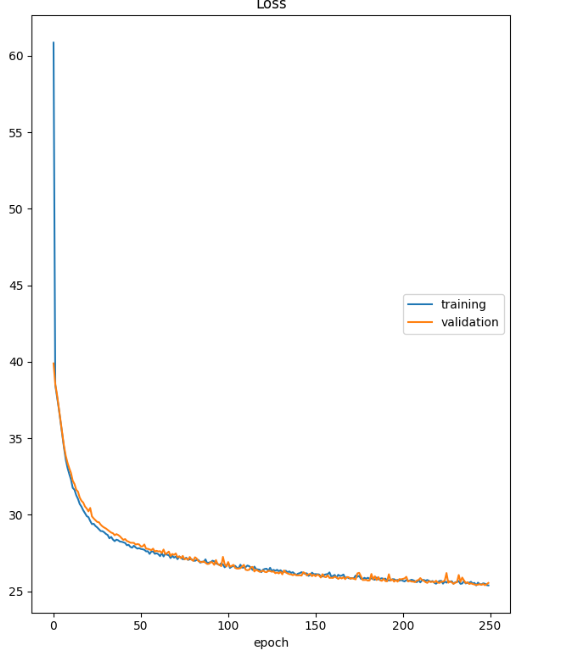
Εικόνα που περιέχει κείμενο, πίνακας

Περιγραφή που δημιουργήθηκε αυτόματα

The plot of the model built using the SGD optimizer, trained for 250 epochs is the following:



The loss function of the 2nd model returns very good results:



**3rd Model Using FTRL Optimizer**

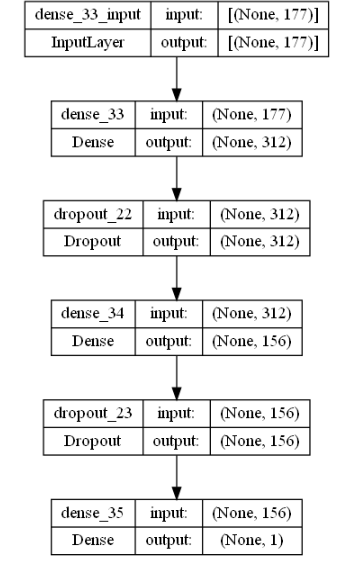
"Follow The Regularized Leader" (FTRL) is an optimization algorithm developed at Google for click-through rate prediction in the early 2010s. It is most suitable for shallow models with large and sparse feature spaces.

The model trained using the FTRL optimizer demonstrates the highest MAE, equal to 27.19 in the test dataset.

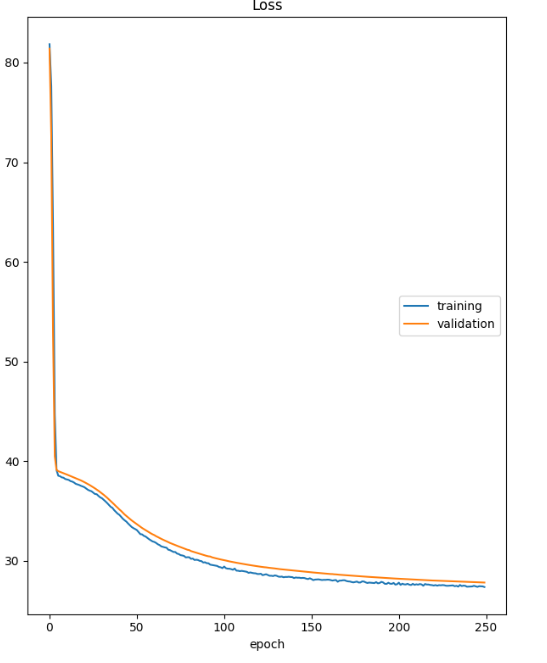
Εικόνα που περιέχει πίνακας

Περιγραφή που δημιουργήθηκε αυτόματα

The plot of the model with FTRL:



The loss function:



# Model II: Using Tensorflow to Recommend Price Adjustment

## Dataset Overview

Based on the:

* name,
* description,
* neighborhood overview and
* price

attributes we built a CNN model which initially assigns the listing to a potential category derived from the quantiles of the prices. A listing based on the price set as input can be characterized as:

* Expensive
* Average
* Cheap

If the initial assumed price is “fair” then the initial pricing segment the listing belongs to is verified, since it matches the predicted pricing segment. On the contrary, in the case where the property is initially characterized as “expensive” and it is predicted as “average”, the owner can consider adjusting the price based on the suggested pricing segment. Consequently, lowering the price of this particular listing could be the optimal way to proceed.

## Data Pre-processing

At this step cleaning the data took place. Three core tasks were performed as far as the data cleaning is concerned:

* Firstly, the null values were handled by setting them to “unknown”, as specified in the constants.py.
* Secondly, the price was fixed by replacing the “,” to “.” and finally normalized.
* Then NA values found in listings were removed.
* Dollar sign symbol “$” was added to define price variable.
* Also, fixed points for the range of prices were determined to determine the categories “cheap” – “average” – “expensive” and finally substitute the price column with the aforementioned categories while the predicted price retains the initial price value of the listing.
* Moreover, a new column containing the concatenation of the text attributes (name, description, neighborhood overview) was created to serve the purpose of the independent variable while price is the dependent variable.

In addition the dataset was split with a 10 to 1 ratio, to training and test.

Before proceeding with the model algorithms some further cleaning was conducted as far as the new columns with the concatenated text is concerned. More specifically, we removed:

* Emojis
* URLs
* Punctuation strings
* Translated words

After this further step of cleaning was implemented on the training and test dataset, we were able to proceed with the model’s algorithms.

To conclude we have:

**Initial dataset shape**: 11321 x 6

**Training dataset shape**: 10107 x 6

**Test dataset**: 1125 x 6

## Methodology Set Up – Algorithm

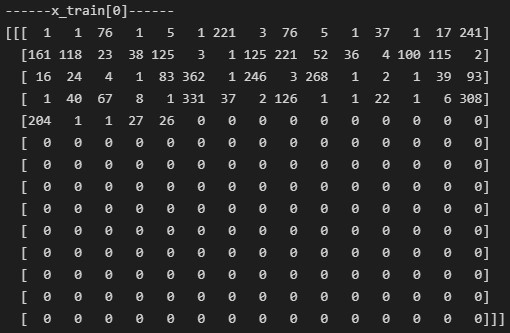
CNN is a kind of neural network; its [convolutional layer](https://analyticsindiamag.com/what-is-a-convolutional-layer/) differs from other neural networks. For example to perform image classification, CNN goes through every corner, vector and dimension of the pixel matrix.   
Convolutional layers consist of multiple features like detecting edges, corners, and multiple textures, making it a special tool for CNN to perform modeling. That layer slides across the image matrix and can detect all its features. This means each convolutional layer in the network can detect more complex features. As the feature expands, we need to expand the dimension of the convolutional layer.

We can consider text data as sequential data like data in time series. These kinds of data are represented by a one-dimensional matrix. Hence, we must work with a one-dimensional convolution layer. The idea of the model is almost the same, but the data type and dimension of convolution layers change. To work with TextCNN, we require a word embedding layer and a [one-dimensional convolutional network](https://analyticsindiamag.com/deep-learning-image-classification-with-cnn-an-overview/).

Word embedding represents the density of the word vector. It is a different way to preprocess the data. This embedding can map semantically similar words. It does not consider the text as a human language but maps the structure of sets of words used in the corpus. They aim to map words into a geometric space which is called an [embedding](https://analyticsindiamag.com/guide-to-word2vec-using-skip-gram-model/) space.

Most common words do not have a large index in our embedding space. Instead, the most uncommon words will get a higher index value which will be word count + 1 because they hold some information. Those whose occurrence is moderate will be given a moderate index value. Finally, 0 value is reserved and won’t be provided to any text.

We used a tokenizer to prepare our text input for the model, after we turned our free text into a sequence we encountered a problem, each sequence has a different length of words, and to specify the length of word sequence, we need to provide a maximum length parameter and to solve this, we need to use pad\_sequence(), which simply pads the sequence of words with zeros.



As you can see the maximum length is 210, we concluded that number is safe because the maximum number of words in our text was equal to 201.

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Next we transformed the text to a number and that number to a category, that way the “cheap” label was represented by the number 1 and the category was { [0],[1],[0] }, the “expensive” label with the number 2 and category { [0],[0],[1] } and last the “average” label with the number 0 and the category { [1],[0],[0] } as you can see below the first 10 rows.

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Keras is a high-level python deep learning API that's easier for ML beginners, as well as researchers. It is integrated as part of Tensorflow 2.0. To build the we used the Sequential model A Sequential model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor (source: <https://www.tensorflow.org/guide/keras/sequential_model>).

The model created has an embedding dimension of 64 and the maximum sequence length is 210. For the convolution layer we use 64 filters and we looked at 2 words at the time, the convolution layers also used the ReLu activation function. For the output layer we used 3 neurons with sigmoid as activation, because of the small number of available data we encountered heavy overfitting, which we tried to reduce by adding l2 regularizers, dropouts. The model’s summary is shown below:

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

In order to better understand the shape of the output we can have a look at the following representation:

For example the first text from the training text is the following: “katerinas feng shui home for those who love…”. Let’s take the first 6 words.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| INPUT TEXT | katerinas | feng | shui | home | for | those |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| INPUT TEXT | katerinas | feng | shui | home | for | those |
| TOKENIZER | 1 | 1 | 76 | 1 | 5 | 1 |

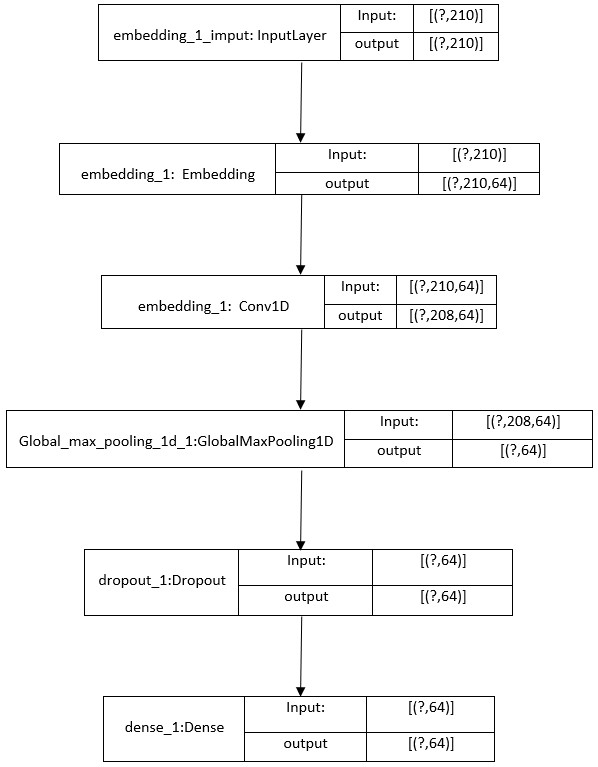
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Embedding Layer | | | | | | | | |
| INPUT  TEXT | TOKENIZER | 1 | 2 | 3 | 4 | 5 | … | 32 |
| katerinas | 1 | 0.06 | 0.01 | 0.08 | 0.03 | 0.06 | … | 0.13 |
| feng | 1 | 0.27 | 0.2 | 0.1 | 0.15 | 0.05 | … | 0.15 |
| shui | 76 | 0.24 | 0.16 | 0.06 | 0.36 | 0.24 | … | 0.02 |
| home | 1 | 0.16 | 0.22 | 0.27 | 0.04 | 0.16 | … | 0.08 |
| for | 5 | 0.15 | 0.06 | 0.24 | 0.16 | 0.24 | … | 0.19 |
| those | 1 | 0.36 | 0.09 | 0.24 | 0.07 | 0.25 | … | 0.11 |

Εικόνα που περιέχει πίνακας

Περιγραφή που δημιουργήθηκε αυτόματα

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1D Convolution | | | | | | Global Max Pooling |
| 1 | 0.3 | 0.24 | 0.35 | 0.21 | 0.18 | 0.35 |
| 2 | 0.16 | 0.42 | 0.27 | 0.36 | 0.19 | 0.42 |
| 3 | 0.22 | 0.28 | 0.24 | 0.16 | 0.45 | 0.45 |
| 4 | 0.27 | 0.22 | 0.02 | 0.22 | 0.37 | 0.37 |
| … | 0.24 | 0.55 | 0.08 | 0.21 | 0.38 | 0.55 |
| 64 | 0.15 | 0.36 | 0.19 | 0.57 | 0.29 | 0.57 |

The plot of the final model, where the batch size in our case was 256:



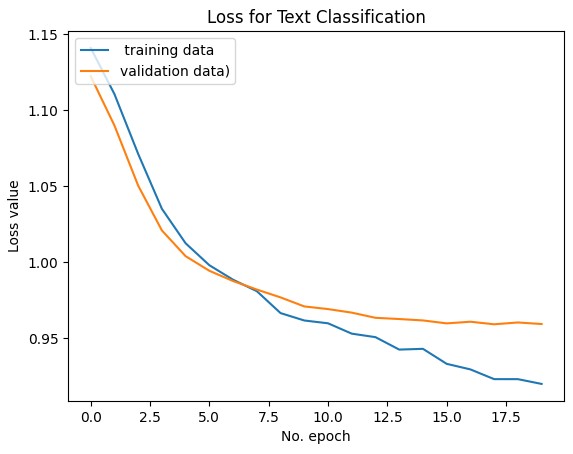
The model was trained for 20 epochs (We run it for 100 epochs with minimum change on the results) and the results arise are encouraging if we consider the small dataset used. At first, let’s explain the metrics to be evaluated:

* ***Precision***: The number of instances that are relevant, out of the total instances the model retrieved.
* ***Recall***: The number of instances which the model correctly identified as relevant out of the total relevant instances.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Categories | Precision | Recall | f1-score | Support |
|  |  |  |  |  |
| cheap | 0.56 | 0.66 | 0.61 | 372 |
| average | 0.49 | 0.36 | 0.41 | 396 |
| expensive | 0.62 | 0.69 | 0.65 | 357 |

For example, for the expensive category we have 62% precision, meaning that out of 100 listings in total 62 were predicted correctly. Also, recall is 62% indicating that out of 100 flagged as expensive, 62 were identified correctly.

We reduced the problem with the overfitting but it was still there, at a lower scale as we can see from the graph below.



After seeing that the model still has an issue with overfitting, we decided to try using an LSTM model.

Nothing changed in terms of the labels or the sequence of the text with the help of the tokenizer.

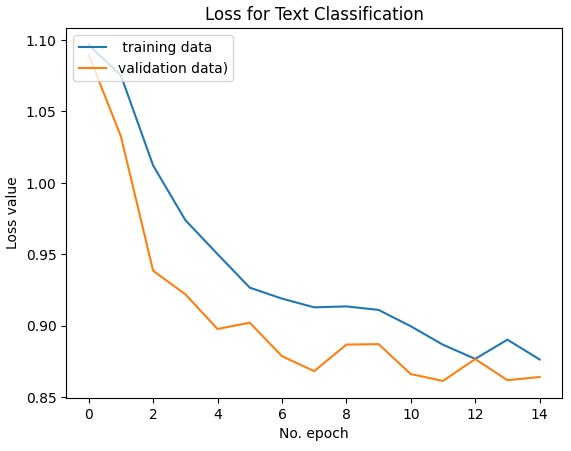
# LSTM Modeling

* Vectorize listings text, by turning each text into a sequence.
* Limit the data set to the top 500 words.
* Set the max number of words in each complaint at 210.
* First layer is the embedded layer that uses 64 length vectors.
* Variational dropout is set to 0.6
* Dropout and recurrent dropout was set to 0.6
* Next is the LSTM layer with 64 memory units
* Output is the layer that gives us 3 values same as the first model.
* As an activation function, we used softmax.
* We run 15 epochs with a batch size or 256.

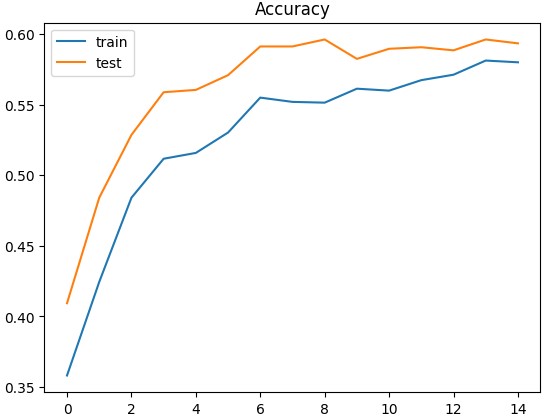
After the model trained for 15 epochs the results, we almost made overfitting go extinct in the expense of prediction. For us it was worth the tradeoff because we managed to solve our biggest problem which was the overfitting because of the small dataset we used. Let’s see the new models results:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Categories | Precision | Recall | f1-score | Support |
|  |  |  |  |  |
| cheap | 0.53 | 0.79 | 0.63 | 372 |
| average | 0.47 | 0.19 | 0.27 | 396 |
| expensive | 0.61 | 0.69 | 0.65 | 357 |

We can compare the loss values from the graph below and the previous graph and see the reduction of overfitting.



We could also check an accuracy plot for the LSTM model.



# Business Value and Further Discussion

AI/ML solutions model human reasoning by allowing the computer to make judgement calls (“inference”) based on past positive or negative outcomes. Just as for human reasoning, the result remains an educated guess. Building more sophisticated models can increase accuracy, but the business value of that accuracy needs to be weighed against the cost of building and training the model. As a result, although we are trying to leverage the power of machine learning models to gain insights, we should always keep in mind the business value these models bring.

It is well-known that explaining an ML model to an audience with no experience or expertise in this field can be a complicated and difficult procedure. Also, there is always the risk of overinflating expectations when using terms as Machine Learning and Artificial Intelligence. Consequently, our aim is to focus on the improvements such models bring in the industry we focused.

There is no doubt that our models need improvements and enhancements in order to maximize the value brought to our potential customers, but at the same time we try to keep in mind the business aspect of the project, which is to keep our clients happy with results. Thus, when building the models our goal is not only to achieve the highest accuracy, precision or recall KPI values but spot the turning point where we can offer our client a tool that can reduce the amount of time spent to decide a price for their listing and help him/her improve their initial assessments, while highlighting the value of right decisions vs the cost of wrong ones.

In Model I section, 3 models predicting the price of the listing were developed. The business value brought by such models is great. The client can submit the rental and by adding its features, he/she can hustle-free obtain a suggested price. This might sound simple but in reality it saves a lot of time and effort from a the time-consuming process of finding listings with similar attributes and characteristics in order to decide the final price. In addition, the advantage of getting a more competitive price should be highlighted, since this is something that in the long run will maximize the profits of the client.

As far as the Modell is concerned, without getting into details on the development of the algorithm, what we tried to achieve is quite straightforward and the possible value gained by the customer is notable. Also, it is easy to use and extra thought in understanding complicated terms or calculating formulas is requires. The customer has simply to consider adjusting the initial input price based on the suggestion made. Furthermore, it is considerably trouble-free to explain how it works. More specifically by leveraging text attributes of listings like the name, the description overview, the neighborhood overview and the price.

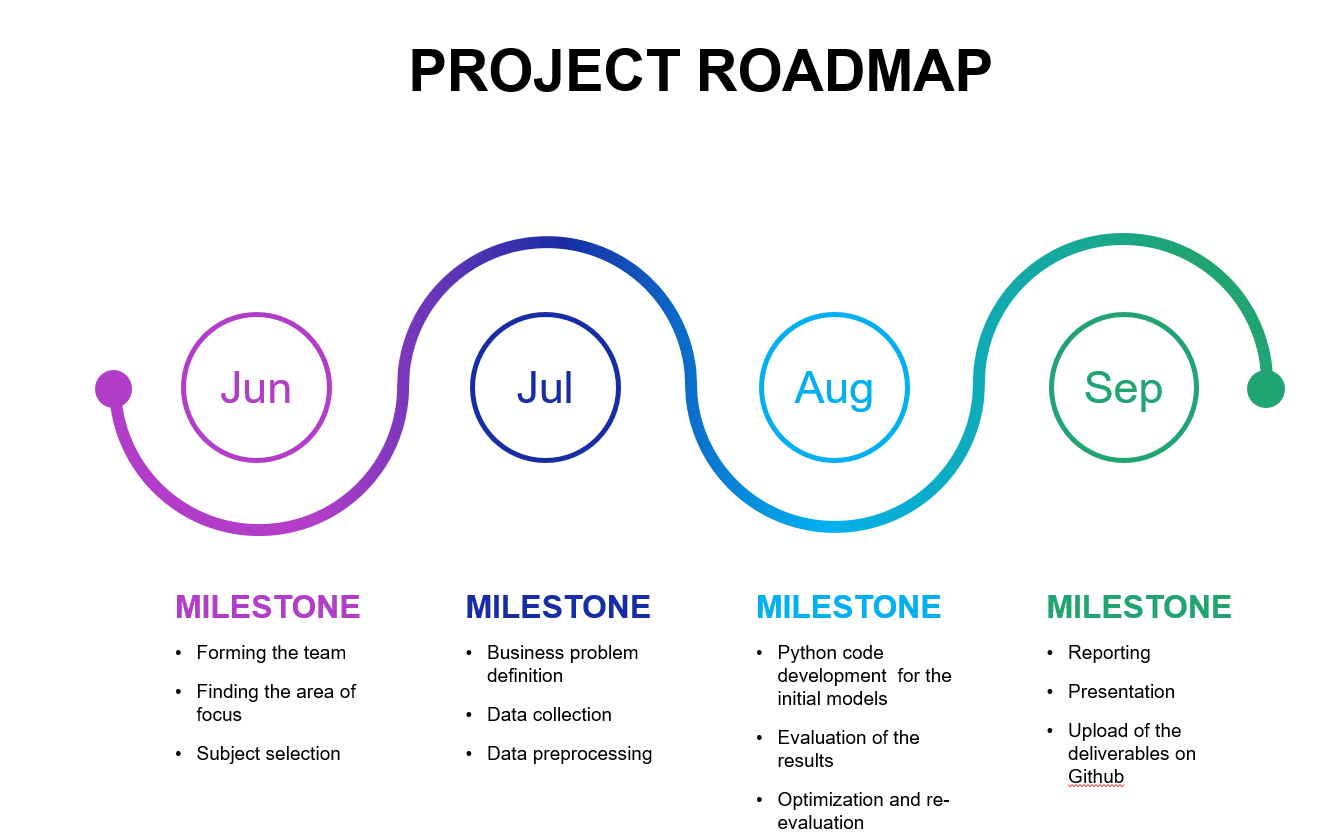
# Members – Roles

The team is consisted of 4 members with different backgrounds and consequently the contribution and perspective of each individual was valuable on this project. The entire process was a team effort but at the same time, each member was working independently in order to bring new ideas, recommendations and improvements on the project. The core tasks performed can be summarized as:

* Brainstorming on the industry of interest where the project would focus
* Identifying the problem that needed to be answered
* Data Collection
* Data Cleaning
* Code development
* Model selection and algorithms
* Reporting
* Presentation of the results

# Time Plan – Project Roadmap

Below is demonstrated the time-plan of the project with the core tasks performed:



# References

1. <https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Ftrl>
2. B. Jason, [How to Choose Loss Functions When Training Deep Learning Neural Networks](https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/) (2020), Machine Learning Mastery.
3. B. Pragati, [12 Types of Neural Network Activation Functions: How to Choose?](https://www.v7labs.com/blog/neural-networks-activation-functions) (2022), V7Labs.
4. S. Kumar, Overview of various Optimizers in Neural Networks (2020).