# AI Challenge

Boston Housing Prices

Francisco Simões

January 2021

## 1 The Problem

The size of global real estate market was 9.6 trillion USD in 2019[2], one of the biggest industries in the world. In pricing houses a lot of variables must be considered, as Polly Adler titled her famous book, *A House is Not a Home*, and so the buyer searches for not only the properties features but also features of the to make said house his home.

In this report we set to model the house prices in the Boston's suburbs. One of the ways to solve this problem would be to evaluate the houses case by case, which banks and individuals do, resorting to experts to appraise the house. We will take an *algorithmic* approach to the problem, and so we will train and test various machine learning models in order to find the one that fits the best our data and will be able to estimate house prices given a set of variables.

## 2 The Data

The data set used in this work consists on 333 data points with the following 16 features[1], all quantitative:

- **crim** per capita crime rate by town;
- **zn** proportion of residential land zoned for lots over 25,000 sq.ft;
- **indus** proportion of non-retail business acres per town;
- **chas** Charles River dummy variable (= 1 if tract bounds river; 0 otherwise);
- **nox** nitrogen oxides concentration (parts per 10 million);
- **rm** average number of rooms per dwelling;
- **age** proportion of owner-occupied units built prior to 1940;
- **dis** weighted mean of distances to five Boston employment centres;

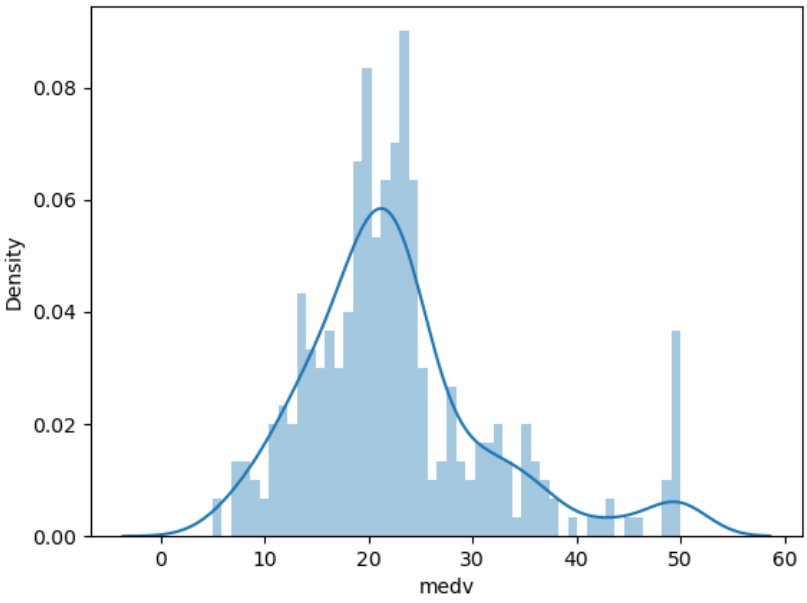


Figure 1: Distribution of the target variable *medv*.

- **rad** index of accessibility to radial highways;
- **tax** full-value property-tax rate per USD 10,000;
- **ptratio** pupil-teacher ratio by town;
- **black** 1000(Bk - 0.63)$^2$ where Bk is the proportion of blacks by town;
- **lstat** lower status of the population (percent);
- **medv** median value of owner-occupied homes in USD 1000s.

The data is clean, i.e. no missing values and the only presence of outliers seem to be of natural outliers.

Our target variable is the latter, *medv*, since our objective is to build a model to estimate house prices based on the data set. In fig.1 we can see how the prices of the houses are distributed in the Boston area, given our data sample. The mean is $\sim$ USD 22,77 with standard deviation $\sim$ USD 9,17. The minimum value is USD 5,00 and the maximum is USD 50,00.

For an initial analysis on the features, we compute the correlation of all of the variables with respect to our target variable. We can see in fig.2 that there are two variables that significantly correlate with our target variable, *lstat* and

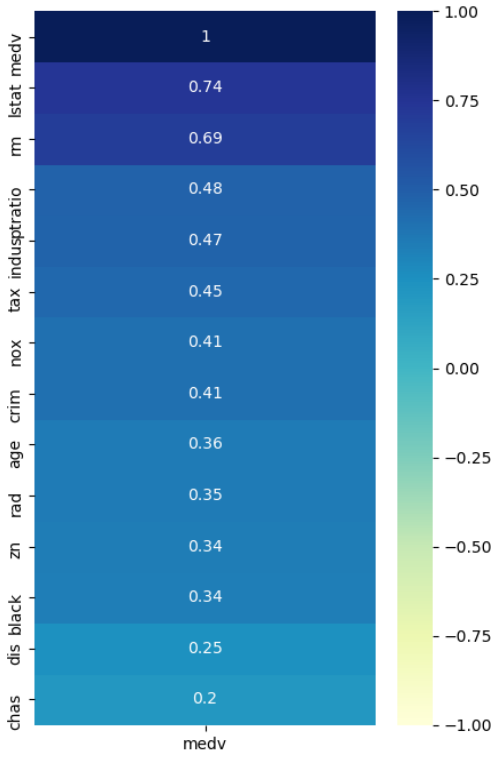


Figure 2: Peterson correlation of all variables with respect to *medv*, absolute value.

*rm*, the percentage of lower status of the population and the average room count per dwelling, respectively. From fig.3 we get that *medv* and *lstat* are negatively correlated, which means that the property's price goes down as the lower status population in the neighbourhood/surrounding area increases. Meanwhile, *medv* and *rm* are positively correlated which is to be expected, the more rooms per house, usually, the more expensive it is. Notice there are some exceptions/outliers in both of these features, we assume these to be natural outliers since one can imagine that a very nice house will be costly even if it has fewer rooms, and vice-versa, compensating with additional features.

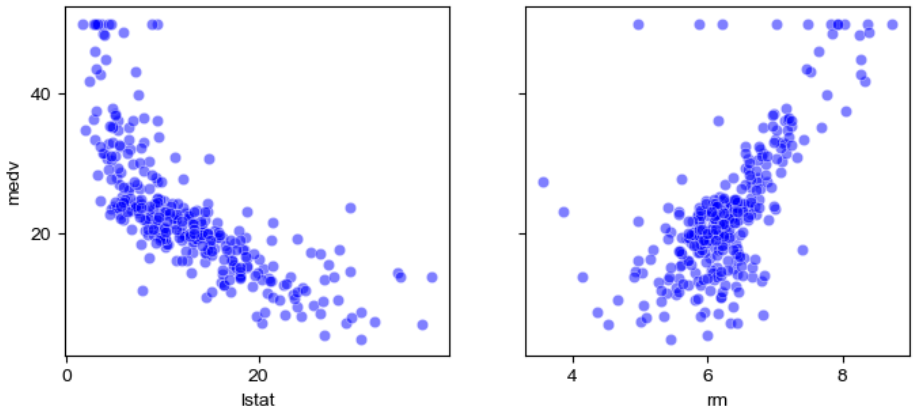


Figure 3: *medv* scatter plots as a function of *lstat* (left) and *rm* (right).

# 3 The Methods

To model our problem we used machine learning algorithms. In training the algorithms we used a randomized sample of 40% of the data points of the data set and the remaining 60% were dedicated to test the models produced. Consider $\mathbf{y}$ to be the target variable of our data and $\mathbf{X}$ to be the features.

## Multiple Linear Regression

In this method we use the data to generate a linear predictor function that minimizes the mean squared error of the training set fitting:

$$\mathbf{y} = \mathbf{X}\beta + \varepsilon, \tag{1}$$

where $\varepsilon$ is a noise or error term and $\beta$ is the coefficient vector that will define our model. Fitting the model consists solving the following least squares problem:

$$\beta = \underset{\beta}{\operatorname{argmin}} \sum_i \left(\beta \mathbf{X}_i - \mathbf{y}_i\right)^2. \tag{2}$$

## Lasso Regression

The Lasso method is very similar to Linear Regression but uses regularization to avoid over-fitting the data during the training phase. Regularization consists on adding a "penalty" term $\lambda$ in order to increase bias to achieve lesser variance than the Linear Regression model during the training phase in hopes that this leads to a better fitting model on test data. We can see the similarities with

Linear Regression in the Lasso formulation below:

$$\beta = \underset{\beta}{\operatorname{argmin}} \sum_i \left(\beta \mathbf{X}_i - \mathbf{y}_i\right)^2 + \sum_j |\beta_j| \,. \tag{3}$$

### Decision Tree Regression

Decision trees are predictive models that use a set of binary rules to calculate a target value. Each individual tree is a fairly simple model that has branches, nodes and leaves. We can predict our target variable $\mathbf{y}$ by answering true or false statements regarding $\mathbf{X}$ embedded in the tree's nodes. The way to train (or construct) such trees is a bit hard to grasp because humans usually don't think this way, but it is as simple as it sounds, the model tries to find the rule that minimizes the mean square error at each binary step.

### Deep Neural Networks

Artificial neural networks are actually inspired by the human nervous system. In the case of *deep* neural networks, there are several hidden layers that are at it's simplest a function that takes on a fixed size input and produces a fixed size output. In the training phase, the algorithm assesses how significant one of the inputs is for the output. In the intermediate layers the output of one layer is the input of the following. At the last layer the output should be the target variable.

## 4 The Results

As described before we train the algorithms on a random sample of 40% of the data set. We repeated this process on 100 unique randomly selected training sets and tested the resulting models on the corresponding test sets. When testing, we computed the root mean square error[1] of each of the models obtained with the different algorithms and the results are summarized in table 1.

| | Mean | Median | 25-percentile | 75-percentile |
|---|---|---|---|---|
| Decision Trees | 5.00 | 5.03 | 4.37 | 5.51 |
| Linear Regression | 5.26 | 5.26 | 4.93 | 5.55 |
| Lasso | 5.64 | 5.64 | 5.34 | 5.96 |
| Neural Networks | 12.42 | 9.99 | 8.91 | 12.47 |

Table 1: Root mean square error (RMSE) results of 100 models of each algorithm.

[1] $RMSE = \sqrt{\frac{\sum_{i=1}^{N} (\hat{\mathbf{y}} - \mathbf{y})^2}{N}}$

### Example

For demonstration purposes we selected a random training set and a data point from the test set. After obtaining models from each of our algorithms we predicted the point's target feature using those models. The results are in table 2.

| **Observed Value** | **20.10** |
|---|---|
| Decision Trees | 19.30 |
| Linear Regression | 20.08 |
| Lasso | 19.70 |
| Neural Networks | 20.59 |

Table 2: Predicting *medv*, our target variable, using the different methods.

## 5 The Conclusions

In this work we were successful in modeling our data set with several algorithms and predicting the target variable, *medv*, using those models, giving us an insight of house price prediction, in particular in the Boston area.

We would recommend using either Linear Regression or Decision Trees to model this problem since they yielded the best results expressed in table 1 as the methods with the least RMSE. These methods also performed well in our example test, with Linear Regression being 0.01% off of the observed variable for the example point.

The Lasso regression is more appropriate when the training set has very limited variability, our data set is well distributed so we don't see the need of employing this method. We this being said, the results using this method were reasonable and we would leave up to the user to choose using this method if he sees fit.

The Deep Neural Network method is highly tunable and, despite doing our best, we never seemed to get satisfactory results from it due to it functioning as a *black box*, we would not recommend this method. Despite of it's result when testing in our example, this method yields poor estimates when the target variable is far from the mean of the distribution.

## References


[1] Kaggle. Boston housing. `https://www.kaggle.com/c/boston-housing/`, 2017.

[2] MSCI. Real estate market size 2019/20. `https://www.msci.com/real-estate/market-size-report`, 2020.