Course Project Report

I. Students

Full name	Student ID
Pham Quang Hieu	20194432
Nguyen Vu Thien Trang	20194459
Nguyen Van Thanh Tung	20190090

II. Problem description

2.1 Description

This is a well known problem named “Boston housing prices”

Given a data set where each record contains 12 explanatory attributes (specified later) which can affect the housing price of a neighborhood in Boston and a response attribute which tells us about the median of the housing price (MEDV) in that neighbourhood. Our task is to determine the medv of a new neighborhood given all the 12 related attributes of that neighborhood.

2.2 Input and output description

- Input: Given a numeric vector of 12 explanatory variables.
- Output: A number (MEDV) represents the mean value of owner-occupied homes in that neighborhood.

2.3 Data set description

The dataset consists of 507 records.

There are **13** attributes in each record of the dataset. They are:

1. CRIM - per capita crime rate by town: A **crime rate** is **calculated** by dividing the number of reported **crimes** by the total population;
2. ZN - proportion of residential land zoned for lots over 25,000 sq.ft: Since such kind of zoning restricts construction of small lot houses, we expect ZN to be

positively related to housing values. Because zoning proxies the exclusivity, social class, and outdoor amenities of a community.

3. INDUS - proportion of non-retail business acres per town. INDUS serves as a proxy for the externalities associated with industry-noise, heavy traffic and unpleasant visual effects.
4. CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise). CHAS represents if the house has a river view or not.
5. NOX - nitric oxides concentration (parts per 10 million)
6. RM - average number of rooms per dwelling
7. AGE - proportion of owner-occupied units built prior to 1940 (this dataset is collected around 1970-1978)
8. DIS - weighted distances to five Boston employment centres (entered under natural log form)
9. RAD - index of accessibility to radial highways: rated from 1 to 24, the greater the value the better the ability to access highways from that neighborhood.
10. TAX - full-value property-tax rate ($/$10,000). Measures the cost of public services in each community.
11. PTRATIO - pupil-teacher ratio by town
12. LSTAT - % lower status of the population - proportion of adults without high school education.
13. MEDV - Median value of owner-occupied homes in $1000's (unit). **This is the response value that we have to predict.**

III. First approach: Hypothesis - Kmeans – Ridge – KNN

3.1 Theory base.

- 1st step: We apply the K-mean clustering algorithm to figure out some kind of similarity or relation between all the neighbourhoods in the training set.

- 2nd step: Then apply Ridge Regression for each of the clusters to find a corresponding optimal regression function.

$$L(y_i, x_i, w) = \sum_{i=1}^{N} (y_i - w \cdot x_i)^2 + \lambda \sum_{j=1}^{m} w_j{}^2$$

Loss function of Ridge Regression.

where w is a m-dimensional weight vector, λ is a regularization term, which will penalize the model for learning large weight values.

- 3rd step: In the testing phase, given a new data sample, we will classify it into one of the clusters defined (by using K- nearest neighbours) and then apply the corresponding regression function learned.

3.2 Dataset processing

As can be seen from the dataset description, almost all attributes are numeric and ordinal, only two of them are categorical, namely CHAS (nominal) and RAD (ordinal).

First let's take a look at how the explanatory variables relate to the response variable MEDV.

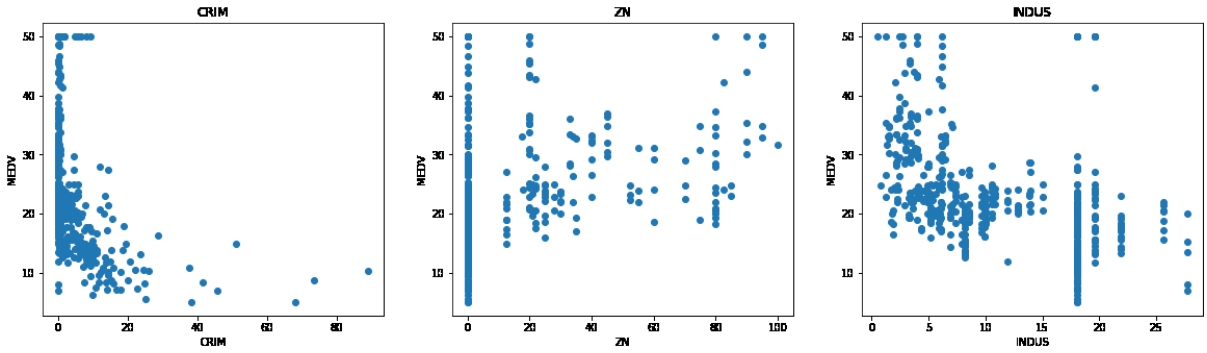

figure 1. relation between CRIM, ZN, INDUS and MEDV

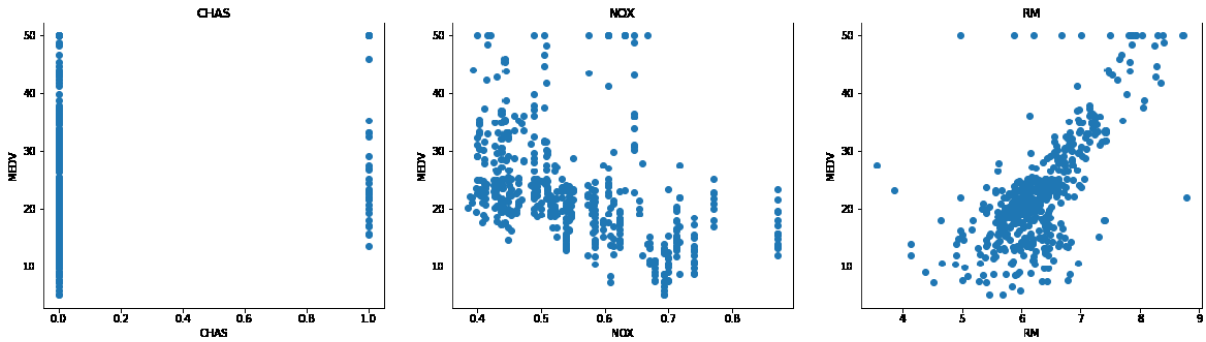

figure 2. relation between CHAS, NOX, RM and MEDV

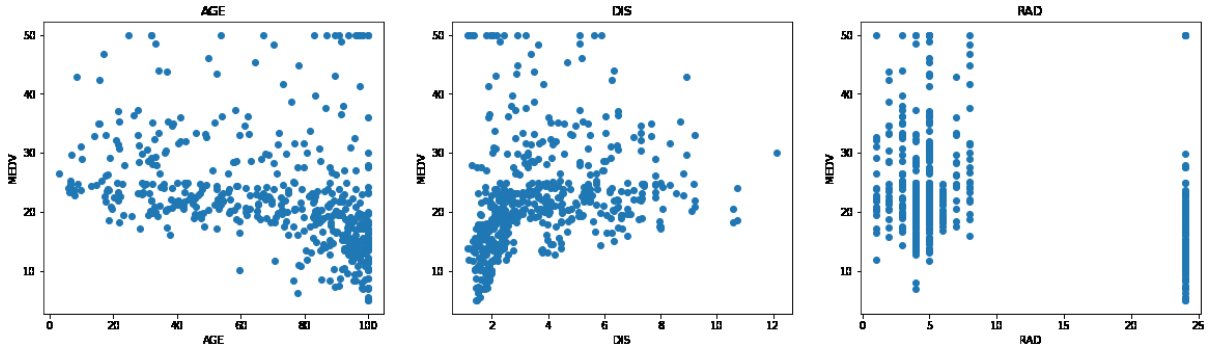

figure 3. relation between AGE, DIS, RAD and MEDV

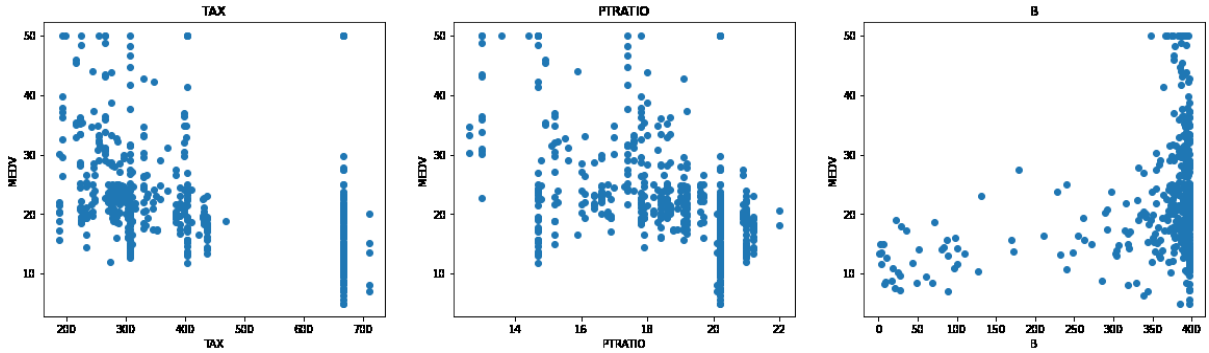

figure 4. relation between TAX, PTRATIO, B and MEDV

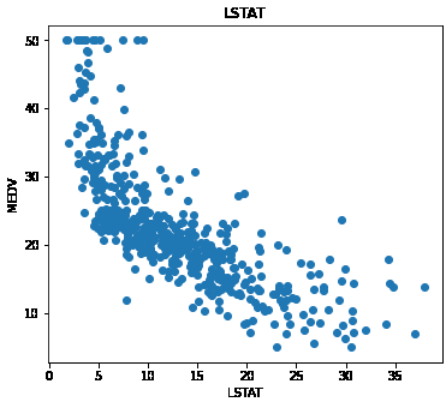

figure 5. relation between LSTAT and MEDV

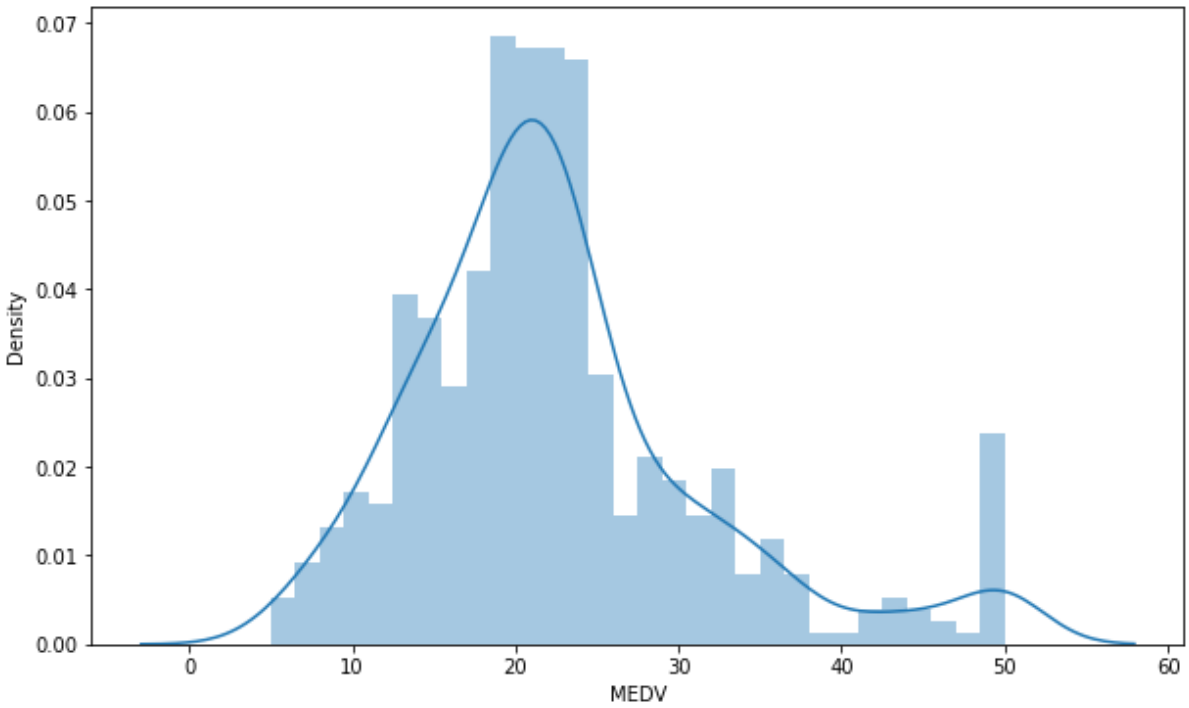

figure 6. Distribution plot of MEDV.

As can be seen from the above relation graph and the distribution plot, those records having a value of MEDV equals to 50 seem to be outliers. Hence, in our second approach, which will make use of Ridge Regression (an algorithm belonging to the family of Linear regression). We decided to remove those records and also those which have RM < 4 or RM > 8, which may affect our "almost linear relation" between RM and MEDV. After the removals, our dataset's size in the first approach reduced from 506 records to 487 records. After that, we perform a "Sequential feature selection" strategy with normal Ridge Regression algorithm, a greedy method, starting with all features, at each step, the algorithm will choose to remove a feature that leads to the smallest decrement in total r2_score (see more detail in the file KMeans-Ridge-KNN-process-data.ipynb).

```
In [13]: df_2.corr()
```

Out[13]:

	CRIM	ZN	NOX	RM	AGE	DIS	TAX	PTRATIO	B	LSTAT	MEDV
CRIM	1.000000	-0.198263	0.421144	-0.220544	0.352202	-0.381220	0.584905	0.286023	-0.381640	0.463252	-0.451583
ZN	-0.198263	1.000000	-0.511344	0.322129	-0.562421	0.672835	-0.300701	-0.380447	0.174848	-0.425043	0.406117
NOX	0.421144	-0.511344	1.000000	-0.334280	0.727418	-0.766889	0.664259	0.183895	-0.383751	0.621803	-0.529655
RM	-0.220544	0.322129	-0.334280	1.000000	-0.271715	0.248489	-0.285220	-0.301827	0.106777	-0.639802	0.729039
AGE	0.352202	-0.562421	0.727418	-0.271715	1.000000	-0.741960	0.496940	0.265696	-0.275598	0.642624	-0.495784
DIS	-0.381220	0.672835	-0.766889	0.248489	-0.741960	1.000000	-0.527890	-0.243011	0.296633	-0.542955	0.372350
TAX	0.584905	-0.300701	0.664259	-0.285220	0.496940	-0.527890	1.000000	0.448858	-0.446407	0.577576	-0.580075
PTRATIO	0.286023	-0.380447	0.183895	-0.301827	0.265696	-0.243011	0.448858	1.000000	-0.170989	0.362429	-0.521559
B	-0.381640	0.174848	-0.383751	0.106777	-0.275598	0.296633	-0.446407	-0.170989	1.000000	-0.366786	0.368325
LSTAT	0.463252	-0.425043	0.621803	-0.639802	0.642624	-0.542955	0.577576	0.362429	-0.366786	1.000000	-0.760424
MEDV	-0.451583	0.406117	-0.529655	0.729039	-0.495784	0.372350	-0.580075	-0.521559	0.368325	-0.760424	1.000000

figure 7. Correlation between the selected attributes after performing feature selection.

We observed that RM and LSTAT are highly correlated together and also both of them are highly correlated to MEDV. Hence we add into our dataset another column (named 'RM*LSTAT') which is calculated by component wise product between RM and LSTAT to manage the implicit change between RM and LSTAT while fitting the model.

3.3 Evaluation metrics & notation

Notations:

- X: the explanatory variable matrix
- X[<attribute_name>]: a attribute's data column
- Y: the response variable column matrix (MEDV)
- x_i: the i-th record explanatory variable
- y_i: the i-th record response variable (MEDV).

Evaluation metrics:

We make use of the R-squared metrics, i.e. The coefficient of determination:

$$R^2 = 1 - \frac{SS_{res}}{S_{tot}}$$

where:

$$SS_{tot} = \sum_i (\underline{y} - y_i)^2 : sum\ of\ square\ total$$

$$SS_{res} = \sum_i (y_i - f_i)^2 : sum\ of\ square\ residual$$

$$\underline{y} = \frac{1}{n}\sum_1^n y_i : mean\ value\ of\ the\ response\ variable\ in\ the\ dataset.$$

$$y_i : true\ response\ variable's\ value\ of\ the\ i-th\ record$$

$$f_i : predicted\ value\ of\ the\ response\ variable\ for\ the\ i-th\ record$$

In the best case when the predicted value always perfectly equals the true value ($f_i = y_i, \forall i$), the metric value will equal to 1. And if the learned model performance is worse than a model which always returns the mean value of the response variable, the metric value will be negative.

Thus, we are aiming at making the R^2 value as close to 1 as possible. Since the dataset's size is small, we evaluate our model of the first approach by the cross-validation method with 5 folds.

3.4 Implementation.

As stated before, there are two categorical variables: "RAD" and "CHAS".

We will remove them out of the training phase of the whole dataset and make an interesting hypothesis

CHAS indicates if a house has a river view or not (1 for yes, 0 for no). RAD indicates the accessibility to the highway of a house (the greater the RAD is, the better the accessibility). Our hypothesis here is that, without those two attributes, the housing price will be lower, and with the contribution of those attributes, the housing price will be boosted by some proportion. Let's say:

$$y_i = y_{i_{original}} * (100\% + \frac{p_1 * x_i['CHAS']}{100} + \frac{p_2 * x_i['RAD']}{100})$$

where: $y_{i_{original}}$ is the MEDV value without considering the two attributes "CHAS" and "RAD".

With this hypothesis, In the training phase we will try many different combinations of p1 and p2 (brute force search) to select the one that gives the best value for the R-squared metric.

Firstly, with the K-means algorithm, we choose the k-means++ strategy for initializing the initial cluster, since it selects the initial cluster based on the probability distribution of the datasets and it has been proven to help the algorithm converge faster in practice.

Because later on, each cluster will be assigned to its own linear regressor and our data is very small in size, we choose only small values for the number of clusters "k1" (eg: 2, 3, 4, 5, 6).

With the value of k1 from 2 to 4 the algorithm works just fine, but the greater the value of k the easier the model will be overfitted. With k1 = 5 or k1 = 6, when the system comes to the test phase, there may be some clusters that have no members in the test set according to the K-NN classifier. Thus, that means we waste our resources at the training phase to train Ridge regressor for those clusters. By practical experiment we choose k1 = 2 for the k-means algorithm. Since we only have two clusters in this algorithm, the number of nearest neighbors in K-NN does not really affect the outcome of the algorithm, thus we choose the weight by distance method and the number of neighbors to take into consideration equals to 10. (by default of the sklearn library).

Before fitting data into the Ridge Regression model, we have scaled the data to the range of [0, 1] and also added in a bias factor for every record. Our practical experiment shows that the best value for the regularization term “lambda” (which is called “alpha” in sklearn) is 0.01 in this case (a greater value will make the whole algorithm perform badly on both train and test data).

The final result (R-squared) is: around 0.87 for the train data and around 0.86 for the test data, the amount of overfit observed when we run with different hypothesis value of p1 and p2 is fluctuating around 0.015 and 0.016.

IV. Another approach: Tree-based algorithms.

In this approach, we similarly build three different regression tree-based models, which are: the decision tree regressor, the random forest regressor, and the XGBoost regressor. And then perform hyperparameter tuning on them by using grid search.

Firstly, for the decision tree regressor, the mechanism of splitting is similar to that of the classification model, except for one core thing is that the algorithm is aiming at maximizing the reduction of the variance of the model result after splitting at each node instead of purity score.

Secondly, for the random forest regressor, this method builds multiple regression trees based on a strategy called bootstrap sampling, which has been included in the theory lecture.

Of the two above methods, according to the theory class, and by experiment, we have proven that the random forest outperforms the original regression tree model in this problem, since out data is very small in size, and the random forest have a better mechanism of exploiting the data. The result show that, at the end, the random forest show a final R-squared score on test data ≈ 0.90 while that of regression tree is only ≈ 0.85. However, because of our very limited data size, both of the two mentioned algorithms are very easy to overfit, thus although the score on test data are higher than that of the regression tree, the amount of overfitting for random forest is much greater than that of the regression tree.

That result haven't satisfied our expectation, thus we are looking for a more interesting method, which is called "XGBoost", from some of the article we have read, this method has won in many competition, we have studied in details about the mathematical behind, but within the scope of this report we just explain how the algorithm works and cite the resource that we have study for the math.

The main idea behind XGBoost is that, it will start with an initial value, which is the mean of the response variable, or by default in the library, it is (0.5), and then with that initial value, it will:

1. Compute the residual value (the difference between predicted and true value)
2. Build an XGBoost unique tree, it is in fact a regression tree to predict the residual value, but the splitting method is based on the two metrics called "similarity score" and "gain". The splitting which results in a higher value of gain will be chosen.

$$Similarity = \frac{(sum\ of\ the\ residual\ on\ a\ leaf)^2}{total\ number\ of\ residuals\ on\ that\ leaf + \lambda}$$

$$Gain = Similarity\ of\ left\ child + Similarity\ of\ right\ child - Similarity\ of\ root$$

Intuitively, this formula can be interpreted as, if the similarity of a node is different in terms of sign, it will cancel out each other in the numerator, thus it will result in a low

similarity score. Vice versa, if the residual in a leaf is the same in terms of sign, this means, the similarity will be high. Lambda here plays the role as a regularization parameter. Because, later on, we will prune the tree based on the value of similarity score on external nodes, a higher lambda results in a lower similarity score, thus leading to more pruning.

3. After predicting all the new residual values for all records, we will come back to step 1 and repeat the whole process with all of the new predicted residual values until the algorithm converges.

4. The final predicted value of the XGBoost algorithm will be evaluated by:

$$Initial\ value\ +\ learning\ rate\ *\ (residual\ predicted\ by\ 1^{st}\ tree + residual\ predicted\ by\ 2^{nd} tree + \dots)$$

The similarity score formula above is a gradient-based formula, it aims at reducing the residual gap between the predicted and the true value of the response variable.

And that is also the reason why the regression tree built in this algorithm called, XGBoost "unique" tree, it removes the "random" factor out of the algorithm.

This is an awesome algorithm, but once again, our dataset is very limited, it can not prevent overfitting. The final result on test data is ≈ 0,91 which is better than that of random forest and regression trees and the amount of overfit is also reduced.

V. References & resource

1. XGBoost Part 1 (of 4): Regression; StatQuest
2. XGBoost Part 3 (of 4): Mathematical Details; StatQuest
3. Machine Learning Project: Predicting Boson House Prices With Regression
4. Boston Home Prices Prediction And Evaluation
5. Predicting House Prices Using Scikit Learn's Random Forest Model

Table of Contents