HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

SCHOOL OF INFORMATION COMMUNICATION TECHNOLOGY

Capstone AI Project

HOUSE PRICE PREDICTION

Group 4

Nguyen Van Thanh Tung	20190090
Nguyen Huy Hoang	20194433
Nguyen Mau Tra	20200624
Nguyen Thi Linh	20200349

Under the assistance of:

Assoc. Prof. Than Quang Khoat

Semester 20212

TABLE OF CONTENTS

FINAL REPORT
HOUSE PRICE PREDICTION

0. TEAM MEMBERS:

- Nguyễn Văn Thanh Tùng – 20190090
- Nguyễn Huy Hoàng – 20194433
- Nguyễn Mậu Trà – 20200624
- Nguyễn Thị Linh – 20200349

1. PROBLEM DESCRIPTION

House price prediction is an important concept in the real estate industry and has been a popular problem in research for years since the traditional house price prediction depend on cost and sale price comparison does not satisfy the accepted standards and certification process. In addition to getting accurate prediction it is important to know the factors that have a significant impact on the house price. In this project on House Price Prediction, our task is to **predict house prices in Boston using different approaches**.

2. DATASET DESCRIPTION: BOSTON HOUSING DATASET

Source: https://www.kaggle.com/code/prasadperera/the-boston-housing-dataset/data

The Boston Housing Dataset is a derived from information collected by the U.S. Census Service concerning housing in the area of Boston MA. It consists of 506 records with 13 attributes each. The attributes are:

- 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)

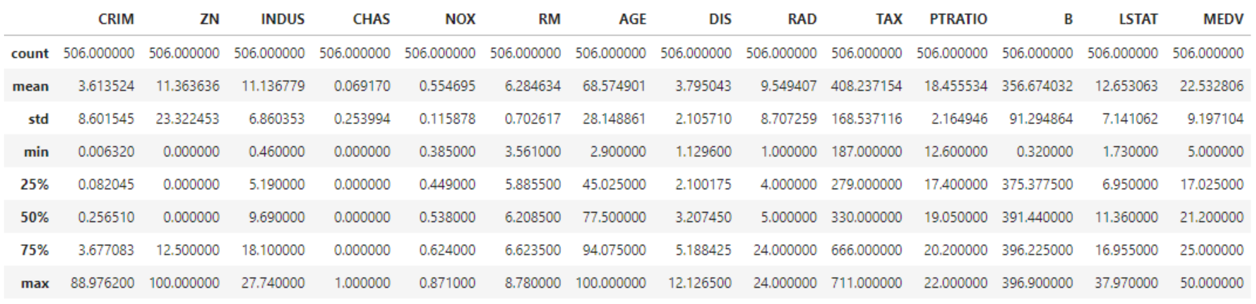
- NOX: nitric oxides concentration (parts per 10 million)
- RM: average number of rooms per dwelling
- AGE: proportion of owner-occupied units built prior 1940
- DIS: weighted distances to five Boston employment centers
- RAD: index of accessibility to radial highways
- TAX: full-value property-tax rate per $10,000
- PTRATIO: pupil-teacher ratio by town
- B: $(1000(B_k - 0.63)^2$ where B_k is the proportion of blacks by town
- LSTAT: % lower status of the population
- MEDV: Median value of owner-occupied homes in $1000’s (unit). **This is the response value that we have to predict.**

3. INPUT AND OUTPUT DESCRIPTION

- **Input:** Representation of the features of a house (a vector of considerable features).
- **Output:** A number (MEDV) represents the mean value of owner-occupied home in that neighborhood.

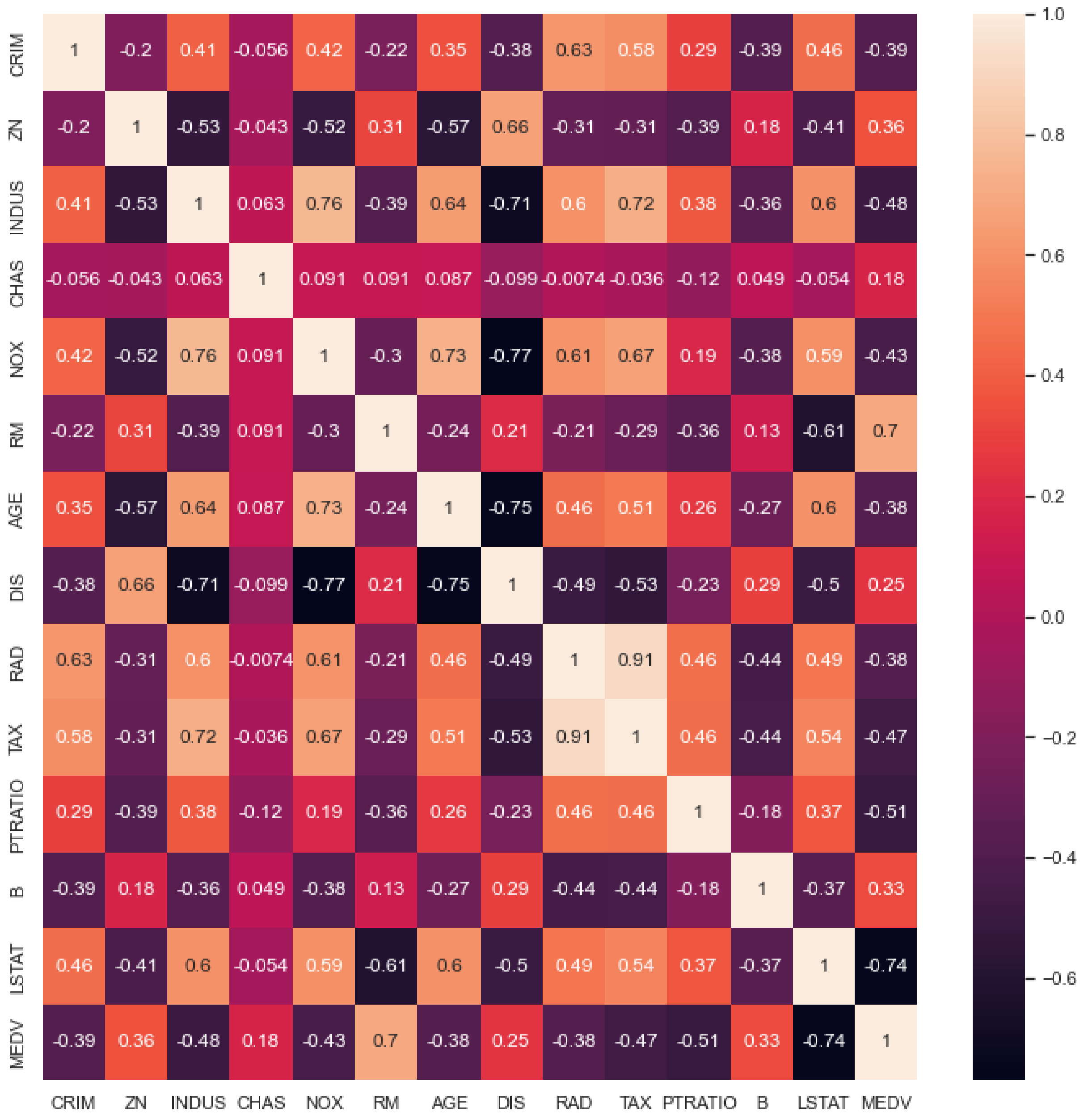
4. EXPLORATORY DATA ANALYSIS

Before training, we load the data into the Jupyter notebook to analyze its exploratory variables. First, missing values are checked and there is 0 missing value. After that, some basics statistics are displayed.



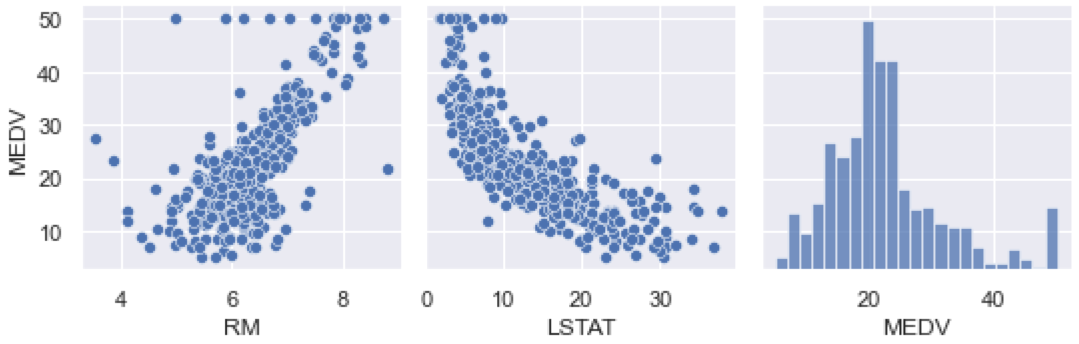
	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.455534	356.674032	12.653063	22.532806
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.164946	91.294864	7.141062	9.197104
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	1.730000	5.000000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000	375.377500	6.950000	17.025000
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000	391.440000	11.360000	21.200000
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000	396.225000	16.955000	25.000000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000	37.970000	50.000000

Then, we plot a heat map to survey the correlation between each feature.

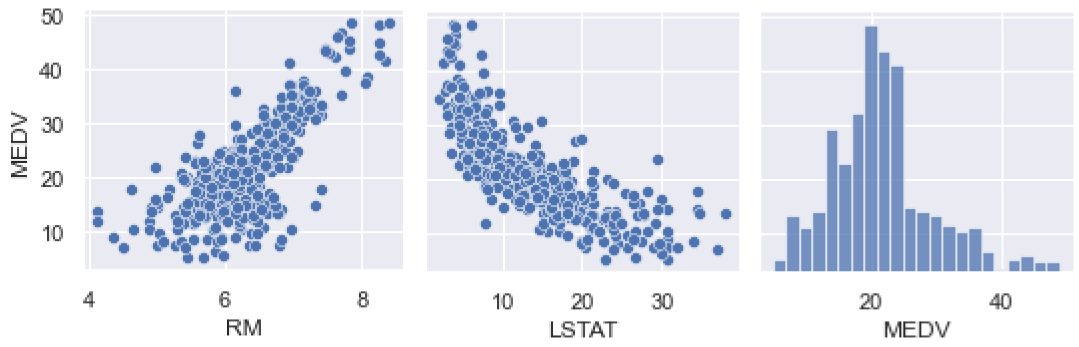


	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
CRIM	1	-0.2	0.41	-0.056	0.42	-0.22	0.35	-0.38	0.63	0.58	0.29	-0.39	0.46	-0.39
ZN	-0.2	1	-0.53	-0.043	-0.52	0.31	-0.57	0.66	-0.31	-0.31	-0.39	0.18	-0.41	0.36
INDUS	0.41	-0.53	1	0.063	0.76	-0.39	0.64	-0.71	0.6	0.72	0.38	-0.36	0.6	-0.48
CHAS	-0.056	-0.043	0.063	1	0.091	0.091	0.087	-0.099	-0.0074	-0.036	-0.12	0.049	-0.054	0.18
NOX	0.42	-0.52	0.76	0.091	1	-0.3	0.73	-0.77	0.61	0.67	0.19	-0.38	0.59	-0.43
RM	-0.22	0.31	-0.39	0.091	-0.3	1	-0.24	0.21	-0.21	-0.29	-0.36	0.13	-0.61	0.7
AGE	0.35	-0.57	0.64	0.087	0.73	-0.24	1	-0.75	0.46	0.51	0.26	-0.27	0.6	-0.38
DIS	-0.38	0.66	-0.71	-0.099	-0.77	0.21	-0.75	1	-0.49	-0.53	-0.23	0.29	-0.5	0.25
RAD	0.63	-0.31	0.6	-0.0074	0.61	-0.21	0.46	-0.49	1	0.91	0.46	-0.44	0.49	-0.38
TAX	0.58	-0.31	0.72	-0.036	0.67	-0.29	0.51	-0.53	0.91	1	0.46	-0.44	0.54	-0.47
PTRATIO	0.29	-0.39	0.38	-0.12	0.19	-0.36	0.26	-0.23	0.46	0.46	1	-0.18	0.37	-0.51
B	-0.39	0.18	-0.36	0.049	-0.38	0.13	-0.27	0.29	-0.44	-0.44	-0.18	1	-0.37	0.33
LSTAT	0.46	-0.41	0.6	-0.054	0.59	-0.61	0.6	-0.5	0.49	0.54	0.37	-0.37	1	-0.74
MEDV	-0.39	0.36	-0.48	0.18	-0.43	0.7	-0.38	0.25	-0.38	-0.47	-0.51	0.33	-0.74	1

From the graph, we can see that RM and LSTAT are highly associated with MEDV (our target variable) with correlation coefficient 0.7 and -0.74 respectively. To justify our hypothesis, we use scatter plot of each feature to MEDV and a histogram of the respond variable. As a result, RM and LSTAT are both linearly correlated with MEDV. However, we notice there are some unusual data point related to the values MEDV = 50, RM < 4 and RM > 8.5 at the graph between those two. In addition, similar pattern can also be seen at the plot of MEDV and LSTAT when MEDV = 50, and the histogram represent the normal distribution except for the data when MEDV =50. That is why, we strongly believe that those data points are outlier and remove them immediately.

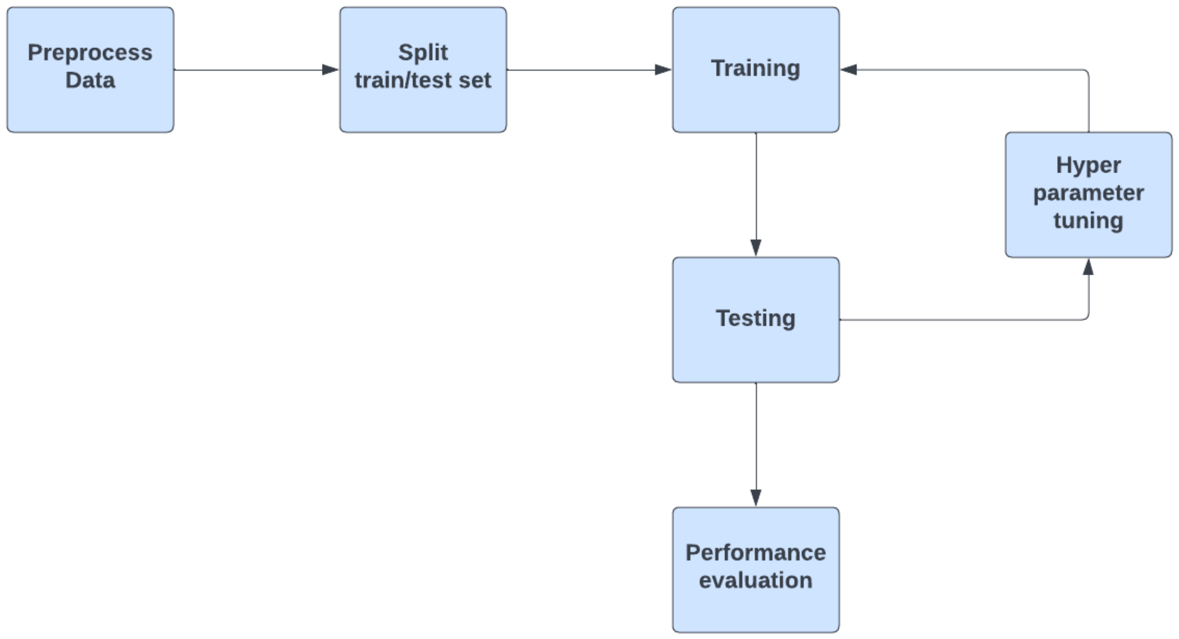


After removing outliners, the heatmap and scatter plots are sketched again. In this time, the correlation coefficient between RM, LSTAT and MEDV slightly increase to 0.73 and -0.76 respectively. Moreover, uncommon data points are disappeared and the histogram is normally distributed.



Finally, the processed dataset is exported.

5. DATA I/O PIPELINE



First and for most, preprocessed data is read into python notebook and split into training set and testing set with ratio 80% training and 20% testing. Second, we pass the train set into the machine learning model. After the training phase, the testing set is passed into the learned model to generate predictions which are used to evaluate model's performance. The next step is hyper parameters tuning, which help discover the best parameters for our machine learning modes and improve their performance. After that, the training and testing phase are executed again in order to produce the best results.

In the training phase, we use three different models corresponding to two approaches (linear regression and tree-based). To be more specific, other than linear regression, tree-based approach includes Decision Tree, Random Forest and Gradient Boosting Regressor. All the basic knowledge about these three model are explained in appendix section.

6. EVALUATION METRICS

6.1. R-SQUARE METRIC

We make use of the R^2 metrics, i.e. The coefficient of determination:

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

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

$\underline{y} = \frac{1}{n}\sum_1^n$

$\underline{y}$: 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.

6.2. MEAN SQUARE ERROR

The mean square error (MSE) tell you how close a regression line is to a set of points. It does this by taking the distances from the points to the regression line (these distances are the "error") and squaring them. The squaring is necessary to remove any negative signs. It also gives more weight to larger differences. It is called mean square error as you are finding the average of a set of errors. The lower the MSE, the better the forecast.

MSE formula:

$$MSE = \frac{1}{n}\sum_i (y_i - f_i)^2$$

where,

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

6.3. MEAN ABSOLUTE PERCENTAGE ERROR

The mean absolute percentage error (MAPE) – also called the mean absolute percentage deviation (MAPD) – measures accuracy of a forecast system. It is measures this accuracy as a percentage, and can be calculates as the average absolute percent error for each time period minus actual values divided by actual values.

The MAPE is the most common measure used to forecast error, probably because the variable's units are scaled to percentage units, which makes it easier to understand. It works best if there are no extremes to the data (and no zero) It is often used as a loss function in regression analysis and model evaluation.

Formula for MAPE:

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|$$

where,

A_t: the actual value

F_t: the forecast value

7. EXPERIMENTAL RESULTS

	R2		MSE		MAPE	
	Train	Test	Train	Test	Train	Test
Linear Regression	0.825	0.699	10.654	20.041	0.135	0.172
Kmeans + Linear Regression	0.916	0.817	5.129	12.237	0.093	0.132
Decision Tree	**1.0**	0.737	**0.0**	17.539	**0.0**	0.156
DT + hypertuning	0.910	0.853	5.459	9.827	0.098	0.122
Random Forest	0.982	0.860	1.057	9.312	0.040	0.112
RF + hypertuning	0.984	0.874	0.922	8.427	0.038	0.107
Gradient Boosting	0.976	0.893	1.449	7.127	0.051	0.100
GD + hypertuning	**0.992**	**0.899**	**0.490**	**6.749**	**0.028**	**0.099**

The table represents the results of training and testing process of each model based on three evaluation metrics R2, MSE and MAPE. As we can see, GD + hypertuning rank first in every category with 0.899 R2, 6.749 MSE and 0.099 MAPE for testing dataset. This is because GB builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage a regression tree is fit on the negative gradient of the given loss function. In addition, the table also shows that every tree-based model performs better after hyper parameter tuning. Another noticeable is that the Decision Tree model have got the absolute score in the training phase due to the fact that the model has memorized every single instance in the training dataset.

8. APPENDIX

8.1. REGRESSION PROBLEM

- Definition:

Regression problem belongs to supervised learning.

The goal of the regression problem is to predict a vector of continuous (i.e., real) values

$$f: X \rightarrow Y$$

where Y is a vector of real values

- Performance evaluation:

8.2. LINEAR REGRESSION

- Definition:

Given an input example, to predict a real-valued output

A simple-but-effective machine learning method appropriate when the target function (to be learned) is a linear function:

$$f(x) = w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n = w_0 + \sum_{i=1}^{n} w_i x_i \qquad (w_i, x_i \in R)$$

To learn (approximately a target function f:

$$f: \boldsymbol{X} \rightarrow Y$$

- X: input space (n-dimensional vector space - R^n)
- Y: output space (real-value space – R)
- f: the function to be learned (a linear mapping function)

In fact, to learn a vector of weights: $w = (w_0, w_1, w_2, \ldots, w_n)$

- Training/ Test examples:

For each training example $x = (x_1, x_2, \ldots, x_n)$, where $x_i \in R$

- Expected output value: c_x ($\in R$)
- Real output value (produced by the system): $y_x = w_0 + \sum_{i=1}^{n} w_i x_i$
  The real output value y_x is expected to (approximately) be c_x

For each test sample $z = (z_1, z_2, \ldots, z_n)$

- To predict its output value by applying the learned target function f

- Error function:

The linear regression learning algorithm needs to define an error (loss) function.

→ To evaluate the degree of error made by the system in the training phase

Definition of the error function E:

- The system's error for each training example x:

$$E(x) = \frac{1}{2}(c_x - y_x)^2 = \frac{1}{2}(c_x - w_0 - \sum_{i=1}^{n} w_i x_i)^2$$

- The system's error for the entire training set D:

$$E = \sum_{x \in D} E(x) = \frac{1}{2}\sum_{x \in D}(c_x - y_x)^2 = \frac{1}{2}\sum_{x \in D}(c_x - w_0 - \sum_{i=1}^{n} w_i x_i)^2$$

- Algorithm:

The learning of the target function f is equivalent to the learning of the weights vector w to minimize the training error E.

The training phase:

- Initialize the weights vector $\boldsymbol{w}$
- Compute the training error $\boldsymbol{E}$
- Update the weights vector $\boldsymbol{w}$ according to **the delta rule**
- Repeat, until converging to a (locally) small error value $\boldsymbol{E}$

The prediction phase:

For a new example z, its output value is computed by:

$$f(z) = w_0^* + \sum_{i=1}^{n} w_i^* x_i^*$$

where $w^* = (w_0^*, w_1^*, \dots, w_n^*)$ is the learned weights vector.

- Delta rule:

To update the weights vector w towards the direction of decreasing the training error E

- η is the learning rate (a positive constant)
  → Define the degree of changing the weight values at a learning step.
- Instance-to-instance/incremental update:
  $$w_i \leftarrow w_i + \eta(c_x - y_x)x_i$$
- Batch update:
  $$w_i \leftarrow w_i + \eta \sum_{x \in D} (c_x - y_x)x_i$$

8.3. DECISION TREES

- Definition:

Decision Trees (DT) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

It is capable of learning disjunctive expressions and robust to noisy data. The approach is one of the most widely used methods for inductive inference and has been successfully applied to a range of real-world applications.

- Decision Trees Representation:

▪ Each internal node represents an attribute to be tested by instances
▪ Each branch from a node corresponds to a possible value of the attribute associated with that node
▪ Each leaf node represents a classification (a class label)
▪ A learned DT classifies an instance by sorting it down the tree, from the root to some leaf node
  → The classification associated with the leaf node is used for the instance
▪ A DT represents a disjunction of conjunctions of constrains on the attribute values of instances
▪ Each path from the root to a leaf corresponds to a conjunction of attribute tests
▪ The tree itself is a disjunction of these conjunctions

- Decision Trees Learning: ID3 algorithm

▪ Perform a greedy search through the space of possible DT
▪ Construct a DT in a top-down fashion, starting from its root node
▪ At each node, the test attribute is the one (of the candidate attributes) that best classifies the training instances associated with the node
▪ A descendant (sub-tree) of the node is created for each possible value of the test attribute, and the training instances are sorted to the appropriate descendant node
▪ Every attribute can appear at most once along any path of the tree
▪ The tree growing process continues
  - Until the (learned) DT perfectly classifies the training instances, or
  - Until all the attributes have been used

- Selection of the test attribute

A very important task in DT learning: at each node, how to choose the test attribute?

To select the attribute that is most useful for classifying the training instances associated with the node.

How to measure an attribute's capability of separating the training instances according to their target classification?

→ Use a statistical measure – *Information Gain*

- Entropy
  A common used measure in the Information Theory field.
  To measure the impurity (inhomogeneity) of a set of instances.
  The entropy of a set S relative to a c-class classification:

$$Entropy(S) = -\sum_{i=1}^{c} p_i \log_2 p_i$$

  where p_i is the proportion of instances in S belonging to class i, and $0. \log_2 0 = 0$

- Information Gain
  Information gain of an attribute relative to a set of instances is the expected reduction in entropy and caused by partitioning the instances according to the attribute.
  Information gain of attribute A relative to set S:

$$Gain(S, A) = Entropy(A) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

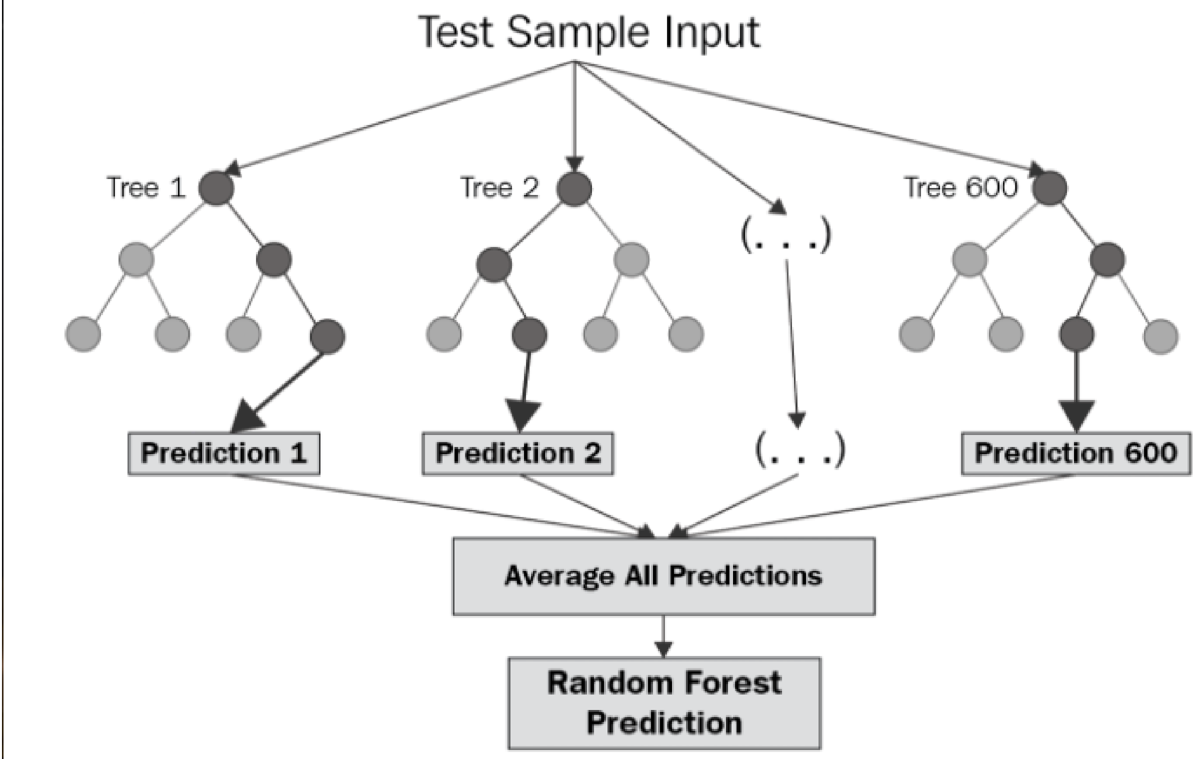
  Where $Values(A)$ is the set of possible values of attribute A, and

$$S_v = \{x | x \ \in S, x_A = v\}$$

  In the above formula, the second term is the expected value of the entropy after S is partitioned by the values of attribute A.

8.4. RANDOM FOREST

Random forest regression is a supervised learning algorithm that uses ensemble learning method for regression. Ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model.



The diagram above shows the structure of a Random Forest. You can notice that the trees run in parallel with no interaction among them. A random forest operates by constructing several decision trees during training time and outputting the mean of the classes as the prediction of all the trees. To get a better understanding of the Random Forest algorithm, let's walk through the steps:

1. Pick at random k data points from the training set.
2. Build a decision tree associated to these k data points.
3. Choose the number N of trees you want to build and repeat steps 1 and 2.
4. For a new data point, make each one of your N-tree trees predict the value of y for the data point in question and assign the new data to the average across all of the predicted y values.

A Random Forest Regression model is powerful and accurate. It usually performs great on many problems, including features with non-linear relationships. Disadvantage, however, include the following: there is no interpretability, overfitting may easily occur, we must choose the number of tress to include in the model.

8.5. GRADIENT BOOSTING REGRESSION

“Boosting”’ in machine learning is a way of combining multiple simple models into a single composite model. This is also why boosting is known as an additive model, since simple models are added one at a time, while keeping existing trees in the model unchanged. As we combine more and more simple models, the complete final model becomes a stronger predictor. The term “gradient” in “gradient boosting” comes from the fact that the algorithm uses gradient descent to minimize the loss.

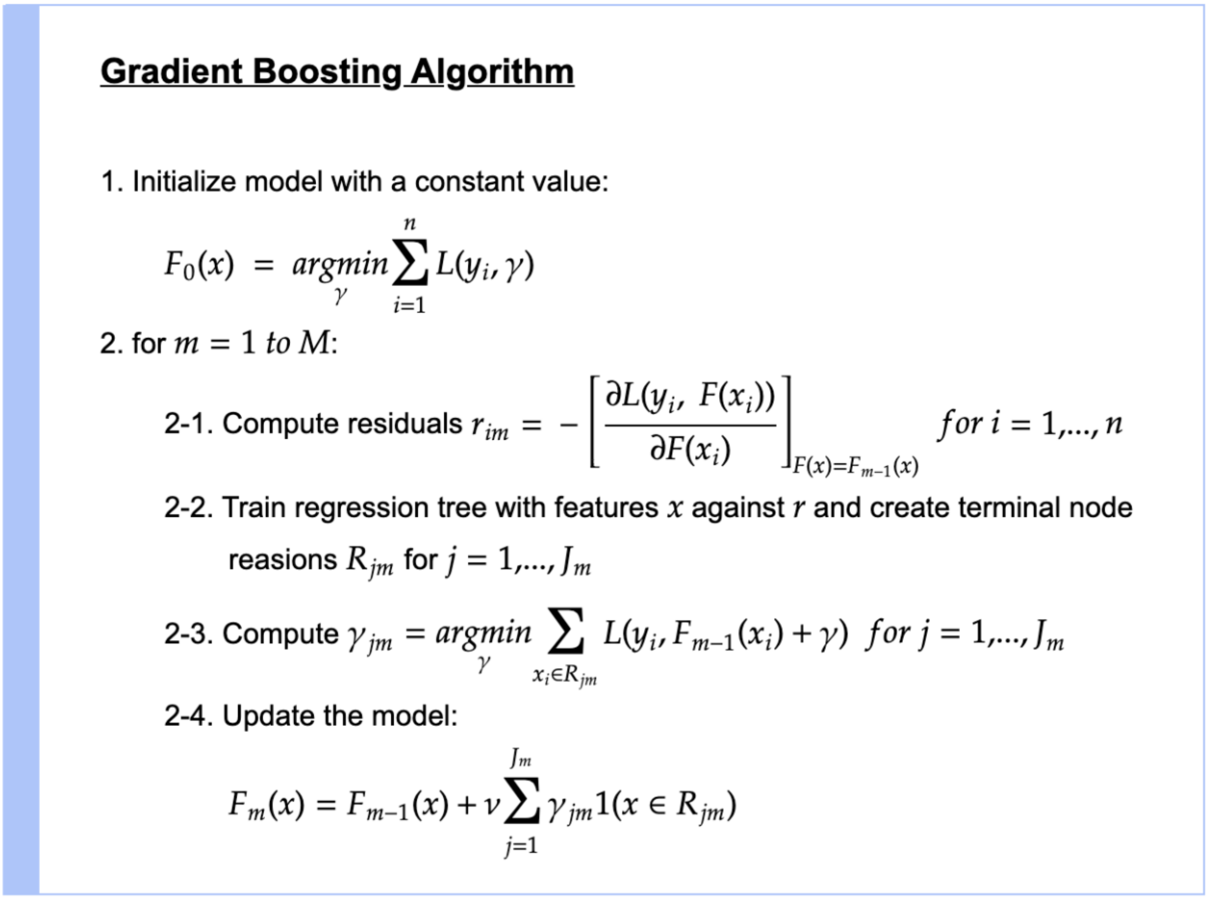
Decision trees are used as the weak learners in gradient boosting. Decision trees solves the problem of machine learning by transforming the data into tree representation. Each internal node of the tree representation denotes an attribute and each leaf node denotes a class label. The loss function is generally the squared error. The loss function needs to be differentiable.

Also like linear regression we have concepts of residuals in Gradient Boosting Regression as well. Gradient boosting Regression calculates the difference between the current prediction and the known correct target value. This difference is called residual. After that Gradient Boosting Regression trains a weak model that maps features to that residual. This residual predicted by a weak model is added to the existing model input and thus this process nudges the model towards the correct target. Repeating this step again and again improves the overall model prediction.

The high level steps that we follow to implement Gradient Boosting Regression is as below:

1. Select a weak learner
2. Use an additive model
3. Define a loss function
4. Minimize the loss function

Here is the whole algorithm in math formulas:



Gradient Boosting Algorithm

1. Initialize model with a constant value:

$$F_0(x) = \underset{\gamma}{argmin} \sum_{i=1}^{n} L(y_i, \gamma)$$

2. for $m = 1 \; to \; M$:

2-1. Compute residuals $r_{im} = -\left[\frac{\partial L(y_i,\; F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad for \; i = 1, \ldots, n$

2-2. Train regression tree with features x against r and create terminal node reasions R_{jm} for $j = 1, \ldots, J_m$

2-3. Compute $\gamma_{jm} = \underset{\gamma}{argmin} \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma) \quad for \; j = 1, \ldots, J_m$

2-4. Update the model:

$$F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} 1(x \in R_{jm})$$

9. ASSIGNED TASKS

Member	Task
Nguyễn Văn Thanh Tùng	- Random Forest - Gradient Boosting Regression - Exploratory Data Analysis (50%) - Slides preparation (50%)
Nguyễn Huy Hoàng	- Kmeans + Linear Regression - Exploratory Data Analysis (50%) - Writing Final Report

Nguyễn Mậu Trà	- Choose topic - Decision Tree - Slides preparation (50%)
Nguyễn Thị Linh	- Collect data - Linear Regression - Check and edit report

REFERENCES

1. The Boston Housing Dataset
2. Machine Learning course, Prof. Tran Quang Khoat, School of Information and Communication Technology, Hanoi University of Science and Technology
3. Mean Square Error
4. Mean Absolute Percentage Error
5. Random Forest Regression
6. All You Need to Know about Gradient Boosting Algorithm – Part 1. Regression
7. Implementing Gradient Boosting in Python