House prices prediction in kaggle

- the main goal of this competition is to predict the house prices of feature ,according to current collect house feature

- Training data and testing data including various feature of House ex. MSSubClass: The building class
  MSZoning: The general zoning classification. Utilities: Type of utilities available

- We want to prediction y: sale prices ,it's a regression problem

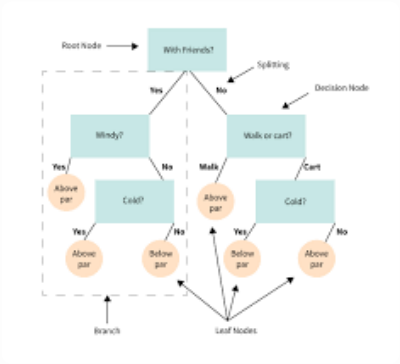
- The competition evaluation using Root-Mean-Squared-Error (RMSE):

> **Image Not Showing**
> Possible Reasons
> - The image was uploaded to a note you don't have access to
> - The note containing the image has been deleted
>
> Learn More →

Module build -decision tree from scratch

Decision tree theory

- Main Structure



- What parameters do I need to define during the training process?

  - Splitting criteria for each node: As shown in the figure, training data needs to define how to split the data at each node using specific criteria to divide the data into two categories.

- Final node (leaf): How to define the class label of its data.

- Defining Module Parameters Principles:

  - The definition of splitting criteria should ensure that the class labels are divided as cleanly as possible.
    → It is necessary to first define how to quantify a node and what standard constitutes cleanliness, which involves using the calculation method of entropy:

$$E = -\sum p(X) \cdot \log_2(p(X)))$$

$$p_x \text{ is the ratio of each class to the total}, p(X) \cdot \log_2(p(X))$$

$$p(X) = \frac{n_x}{n}$$

    → The optimal splitting standard is the one that can maximize the reduction of entropy. The evaluation criterion will use information gain. If the data is split from one node (parent) into two nodes (children), the calculation method for information gain is:

$$IG = E(\text{parent}) - [\text{weighted average}] \cdot E(\text{children})$$

    → Select data features and standard values that can maximize information gain.

- Common Python Modules: Random Forest, XGBoost, AdaBoost

How to implement in python

1. **data preprocessing**:

- load data and store in pandas dataframe

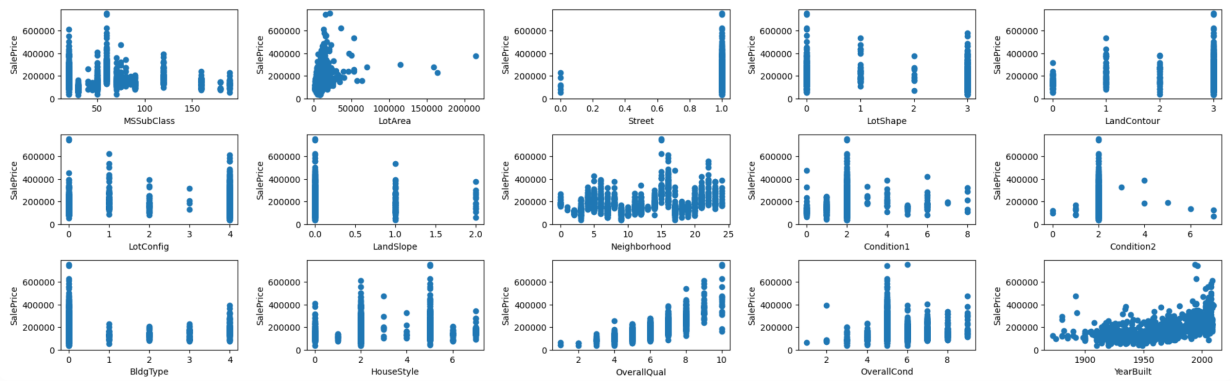
- processing nan data drap

- label encoding : let object type data transform to digit

- split train to trainx, trainy

- split a part of train data to validation

2. **manifest data each feature and target distribution using matplotlib scatter**



3. **Build a decision tree class from scratch **:

- Define the node of decision tree:

→ each node has feature, critira ,and label (optional)

- inititalize feature of training:

```
        self.min_samples_split = min_samples_split ## represent training min
sample of nodes
        self.max_depth = max_depth ## training maximum depth
        self.n_feats = n_feats ## num of features comparing during training
```

- initialize tree nodes: roots, mse dictionary, feature importance ditionary

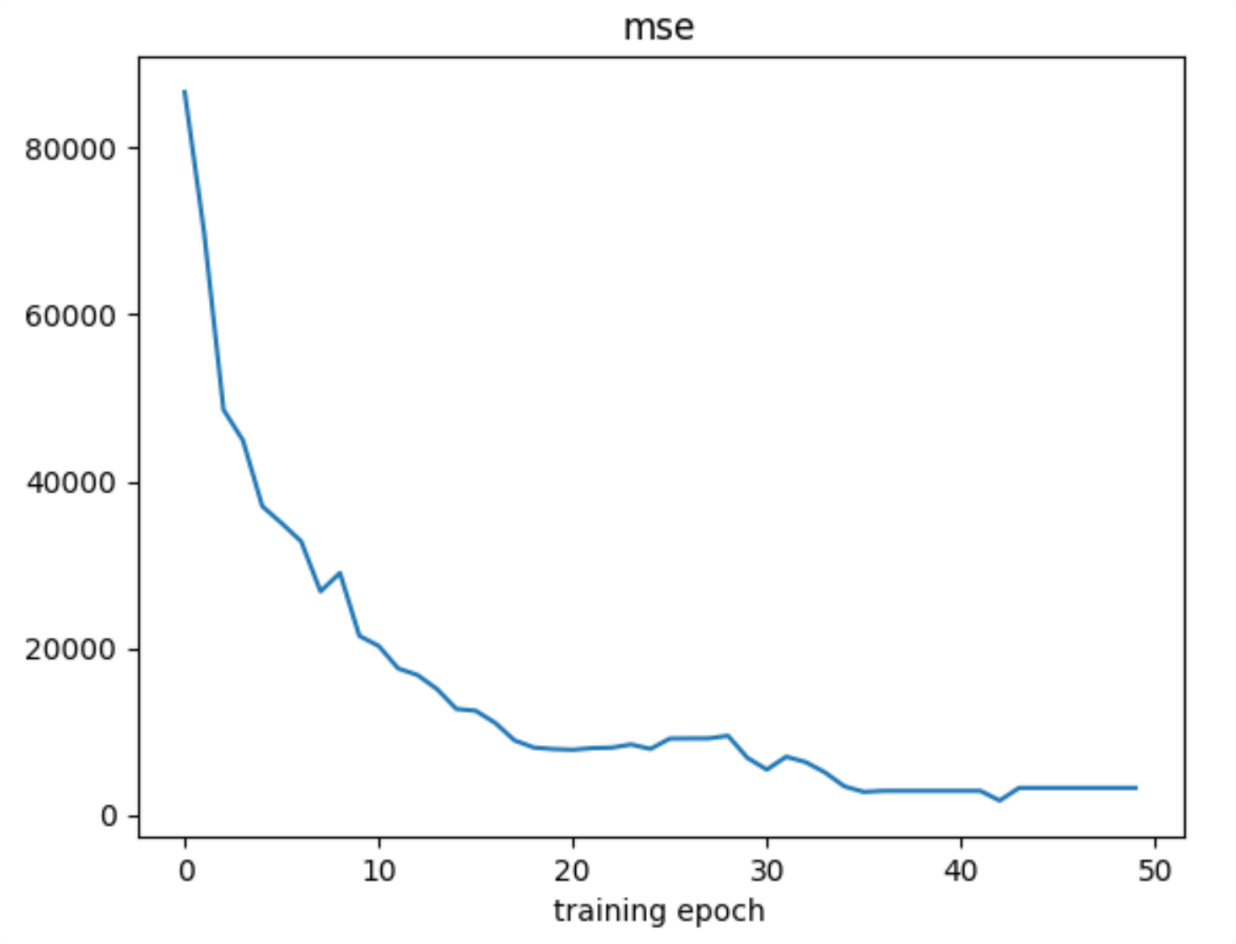
- Build a fit fucntion: using for training
    - use grow tree fucntion to find all node spliting feature, critiria
    - using best critira function to calculate information gain to find best spliting feature and critiria
    - using recursive method to find left and right node

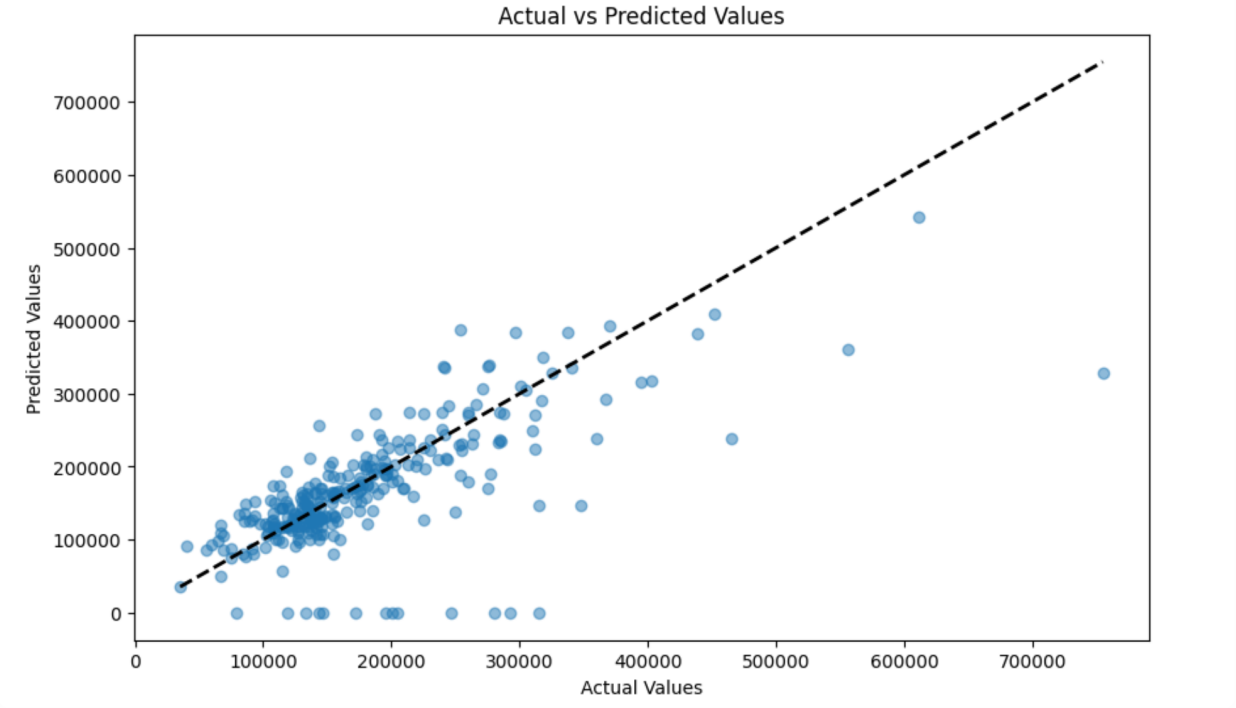
- Build a prediction function:
    - using traverse tree function to predict each data

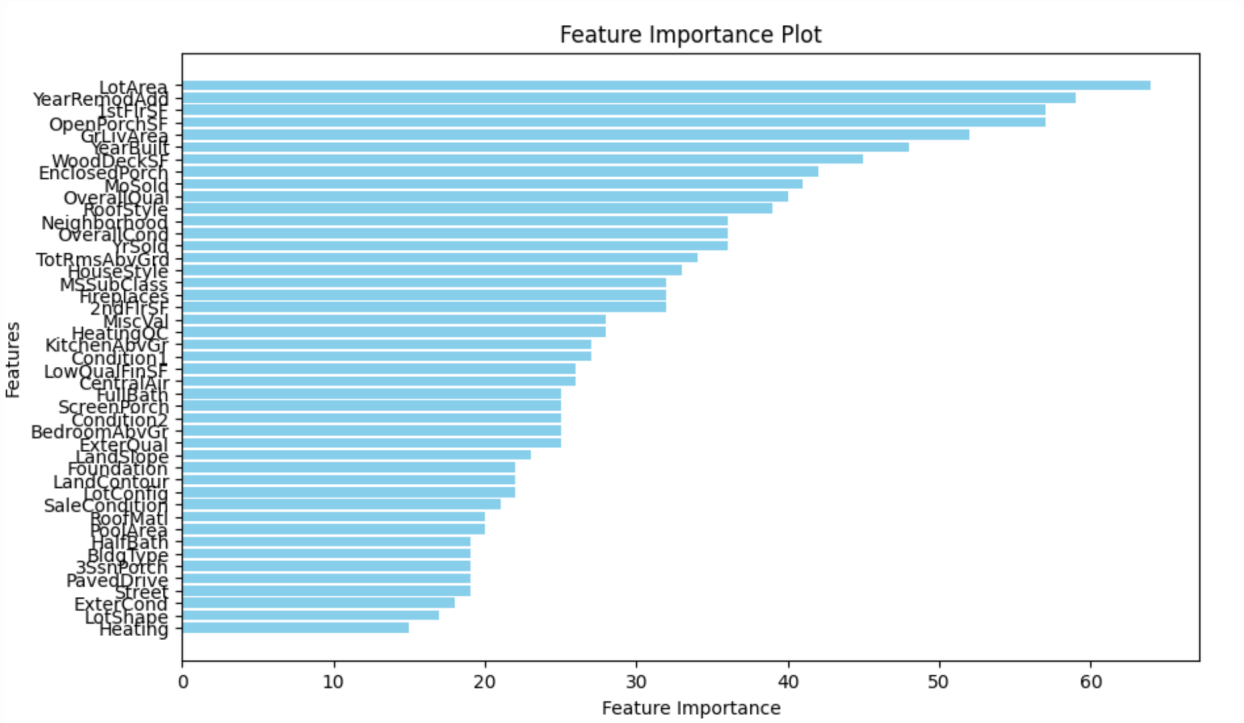
4. **find feature importance**:

- using the feature split times in decision tree to define the feature importance

Prediction result:






Feature Importance Plot
LotArea
YearRemodAdd
1stFlrSF
OpenPorchSF
GrLivArea
YearBuilt
WoodDeckSF
EnclosedPorch
MoSold
OverallQual
RoofStyle
Neighborhood
OverallCond
YrSold
TotRmsAbvGrd
HouseStyle
MSSubClass
Fireplaces
2ndFlrSF
MiscVal
HeatingQC
KitchenAbvGr
Condition1
LowQualFinSF
CentralAir
FullBath
ScreenPorch
Condition2
BedroomAbvGr
ExterQual
LandSlope
Foundation
LandContour
LotConfig
SaleCondition
RoofMatl
PoolArea
HalfBath
BldgType
3SsnPorch
PavedDrive
Street
ExterCond
LotShape
Heating
Features
0
10
20
30
40
50
60
Feature Importance