Team: Jin He

Class: CS5350

Instructor: Shandian Zhe

Date: May 4, 2019

Final Report

what problem did you choose

As I mentioned in my last report, I continue with my last topic about house-price prediction after the mid-term report, I just analyze the housing price in some cities of China due to the data is more lucency and convenient to translated it the training data.

Why is it important or interesting?

This topic is very popular in the whole of the world, almost all people are interested in the tendency of the house pricing in the future, as we know, house price is always more mystery if we compare with other countries. If I can use machine learning knowledge to get some specific data for the future house price in China, I think the data would help many people in China to decide when the time is the best to buy a house. Meanwhile, the government maybe could get some idea how to control the house price recently to maximize the benefits for the Chinese government and Chinese people.

Why did you use machine learning techniques to solve it?

In this project, I used DecisionTreeRegression to implement the necessary code part, my training set include the population in every area, it includes the median set and the median income in this area. I research lots of things and ask some agents at the housing companies, the purpose I did it is I have to get the business purpose of the

housing price prediction, and what kind of model does the company want. It always decided the design processing, and what kind of algorithm I choose, and how much time I should spend to adjust it. The data set that I used have its own tags. And In this project, I used small data for the input, so the price always can not get a big change. From my data sets, each data row has 10 properties, and it contains 20640 data totally. We have total_bedrooms property data 20433 rows, and some of these data include empty value. And almost all property has a Integer value in my data set. The only ocean_proximity is an object, and it actually contains some repeat value. So I used it as a special character.

My Solution:

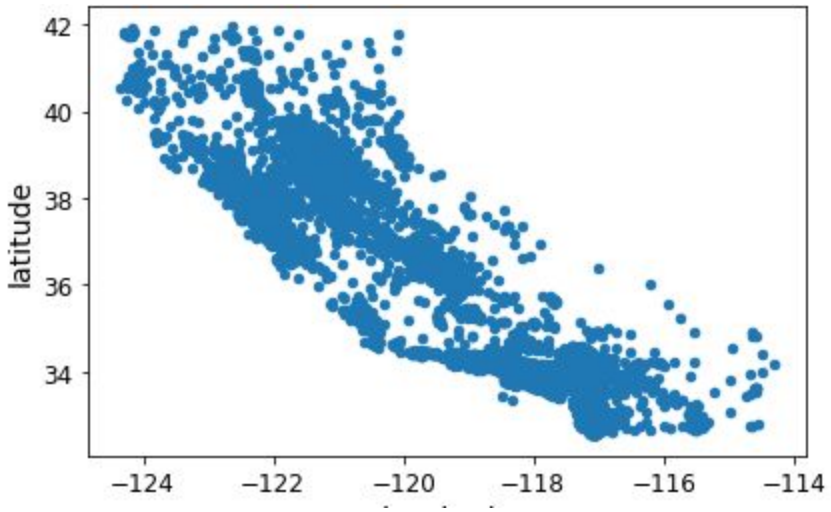
I used some new data set to do the model training. So I think it’s necessary to divide the data to traning data and test data. While I .apply the trained model to the test data, the error should be generalization error.

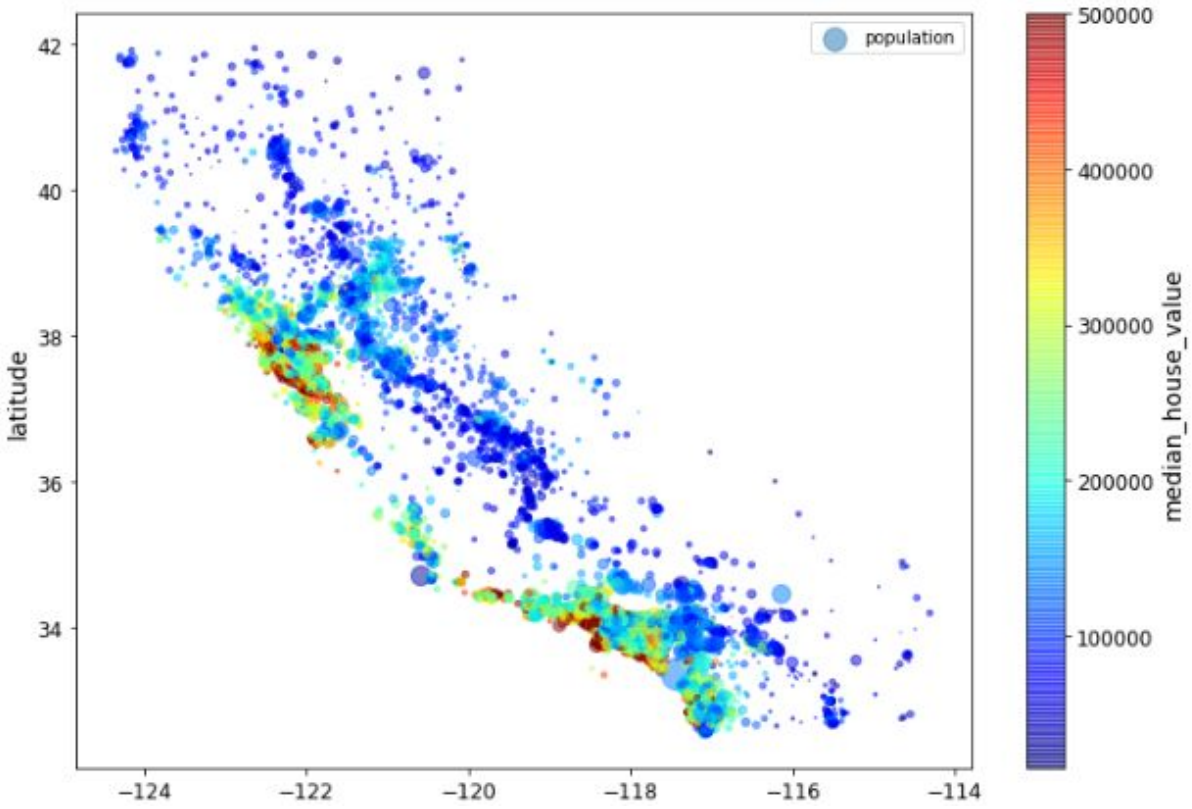
The parameter random_stateis used to set up a random seed, it will generate a random number and then put the number into different data set, these data sets will use a same random number to divide to do the index search. Test_size is a size of test set.

Suppose we learn from interviews with experts that median income is an important attribute for house price forecasting, so we want to conduct stratified sampling based on median income. Because median income is a continuous numeric attribute, we first need to create a category-type attribute with a median income. Let's first take a look at the distribution of the median income attribute.

Most media's median income is concentrated at 2-5 (units are tens of thousands of dollars), but some are much larger than 6. The important point of layered sampling is to ensure that each layer has enough data, so we should not have too many layers; the method chosen in this book is to divide the median income by 1.5 and then round up through the ceil function. Make it a discrete integer value, and finally categorize all values greater than 5 into one class (that is, all of them are 5.0); you can view this newly created property by value_counts()

My alpha setting here is 0.2, the original book is 0.1, you can try other values yourself, the closer to zero, the higher the transparency; after setting alpha, the distribution law becomes very obvious, the density of dark places Bigger.





Now let's add more attributes. Set the radius of each dot to indicate the population of a certain area. The dot color indicates the house price. Use a defined colormap (jet) to indicate the price of the house

To further explore the correlation between attributes, you can use the corr() method to simply calculate the Pearson correlation coefficient and then see how each attribute relates to the price:

I can see that median income (median_income) is highly correlated with house prices. Another way to look at correlations is to use the scatter_matrix function of the Pandas library, which draws a scatter plot between each numeric property. Because there are

too many attributes, we only select a few more highly correlated attributes to further explore

If the scatter plot on the diagonal should be a straight line, it doesn't make sense, so Pandas draws a histogram of the property instead. From the scatter plot above, it is clear that median income (median_income) has the highest correlation with house prices, so next we focus on this property to explore

It can be seen that the correlation between income and house price is very high. The points in the scatter chart are relatively concentrated and show a positive trend; and the house price has a very obvious upper limit of around $500000. In addition, we can also find it at $450000 and The position of $350000 also has a horizontal line. Later, these two horizontal lines are likely to be the data noise we want to eliminate. This is for everyone to explore.

Our final step before data preprocessing is to try to combine the new features. In this example, we might want to combine the total number of rooms (total_rooms) and the number of houses (households). (rooms_per_household), there is a similar number of bedrooms per bedroom (bedrooms_per_room), number of people per house (population_per_household), etc.

Experimental Evalution:

We will be surprised to find that the prediction error is zero and the visible model fits the training data very well. But in this case, we usually think that he is over-fitting. At this time, we need to separate a part from the training set data to verify the model.

We can use the train_test_split function to split the training set into a training set and a validation set; but a better approach is to use the k-fold cross-validation provided by sklearn. In this example, we implemented a 10-fold cross-validation that divides the data set into ten small data sets. It trains and validates the model ten times on the training set, each time selecting a different set as the verification of the remaining nine. Used for training. Input parameters include trained models, pre-processed training data, dataset labels (ie, house prices), indicators for measuring errors, and k-values for k-fold cross-validation

Future Plan:

We tried a few models that we thought were better through a series of attempts, then the next step we have to do is to adjust their parameters. One approach is to manually try hyperparameters of various combinations, but this is cumbersome. Sklearn provides a grid search GridSearchCV, you only need to input the various hyper-parameters you want to try, it will automatically perform various combinations and verification for you.