House price prediction

The data file contains records of 774 houses sales. Choose randomly 695 of them for training and the rest for testing

1. Use the columns x_1 containing the area of each house and x_2 containing the number of bedrooms in order to create a regression model of the form

$$\hat{y} = \beta_1 x_1 + \beta_2 x_2 + c$$

2. Create a scatter plot of actual and predicted sale price for all 774 houses.
3. Use the following 5 houses to test the efficiency of the above method

House	X_1	X_2	Actual price
1	846	1	115000
2	1324	2	234500
3	1150	3	198000
4	3037	4	528000
5	3984	5	572500

4. Find the Root-Mean-Square Error (RMSE) for the training and test data. If they are close enough we have increased confidence that our model has reasonable generalization ability

5. We develop a more complicated regression model for the house sales data.. For this more complex model we will use four base attributes or original features:

• x_1 is the area of the house (in 1000 square feet),
• x_2 is the number of bedrooms,
• x_3 is equal to one if the property is a condominium, and zero otherwise,
• x_4 is the five-digit ZIP code.

Here we examine a more complicated model, with 8 basis

$$\hat{y} = \sum_{i=1}^{n} \theta_i \, f_i(x)$$

These basis functions are described below

The first basis function is the constant $f_1(x) = 1$. The next two are functions of x_1, the area of the house,

$f_2(x) = x_1$, $f_3(x) = \max\{x_1 - 1.5, 0\}$.

In words, $f_2(x)$ is the area of the house, and $f_3(x)$ is the amount by which the area exceeds 1.5 (i.e., 1500 square feet). The first three basis functions contribute to the price prediction model a piecewise-linear function of the house area

$$\theta_1 f_1(x) + \theta_2 f_2(x) + \theta_3 f_3(x) = \begin{cases} \theta_1 + \theta_2 x_1 & x_1 \leq 1.5 \\ \theta_1 + (\theta_2 + \theta_3) x_1 & x_1 > 1.5, \end{cases}$$

with one knot at 1.5.

The basis function f4(x) is equal to the number of bedrooms x2. The basis function f5(x) is equal to x3, i.e., one if the property is a condominium, and zero otherwise. In these cases we simply using the original feature value, with no transformation or modification.

The last three basis functions are again Boolean, and indicate or encode the location of the house. We partition the 62 different ZIP codes present in the data set into four groups, corresponding to different areas as shown in the following table. The basis functions f6, f7, and f8 give a one-hot encoding of the four groups of ZIP codes.

x_4	$f_6(x)$	$f_7(x)$	$f_8(x)$
95811, 95814, 95816, 95817, 95818, 95819	0	0	0
95608, 95610, 95621, 95626, 95628, 95655, 95660, 95662, 95670, 95673, 95683, 95691, 95742, 95815, 95821, 95825, 95827, 95833, 95834, 95835, 95838, 95841, 95842, 95843, 95864	1	0	0
95624, 95632, 95690, 95693, 95757, 95758, 95820, 95822, 95823, 95824, 95826, 95828, 95829, 95831, 95832	0	1	0
95603, 95614, 95630, 95635, 95648, 95650, 95661, 95663, 95677, 95678, 95682, 95722, 95746, 95747, 95762, 95765	0	0	1

6. Find the new model resulting by the new base. Do question 2.

Cross validation

Cross-validation is an extension of out-of-sample validation that can be used to get even more confidence in the generalization ability of a model, or more accurately, a choice of basis functions used to construct a model. We divide the original data set into 10 sets, called folds. We then fit the model using folds 1, 2, . . . , 9 as training data, and fold 10 as test data. (So far, this is the same as out-of-sample validation.) We then fit the model using folds 1, 2, . . . , 8, 10 as training data and fold 9 as the test data. We continue, fitting a model for each choice of one of the folds as the test set. We end up with 10 (presumably different) models, and 10 assessments of these models using the fold that was not used to fit the model. (We have described 10-fold cross-validation here; 5-fold cross-validation is also commonly used.) If the test fit performance

of these 10 models is similar, we can expect the same, or at least similar, performance on new unseen data. In cross-validation we can also check for stability of the model coefficients. This means that the model coefficients found in the different folds are similar to each other. Stability of the model coefficients further enhances our confidence in the model.

To obtain a single number that is our guess of the prediction RMS error we can expect on new, unseen data, it is common practice to compute the RMS test set error across all 10 folds. For example, if ε1 , . . . , ε10 are the RMS prediction errors ,obtained by our models on the test folds, we take

$$\sqrt{(\epsilon_1^2 + \cdots + \epsilon_{10}^2)/10}$$

as our guess of the RMS error our models might make on new data. In a plot like that in figure 13.11, the RMS test error over all folds is plotted, instead of the RMS test error on the single data or validation set, as in that plot. The single number is called the **RMS cross-validation** error, or simply the RMS test error (when cross-validation is used).

7. Use the cross validation technique for the above 2 models to testify the confidence in the generalization of the model.

Remark

In order to reproduce the results arriving from the use of np.random() function you can reset the seed of this global RNG at the beginning of a script using the np.random.seed function.