Linear Regression In House Prices Prediction

Group : Run As One

May 6, 2019

1 Group member

Zhou Fang 6286914
Shuai Zhang 6286951
Ruyu Li 6286963
Binglin Liu 6287013
Zhiwen Liu 6287402

2 Introduction

2.1 Aim

The goal of this task is to design and compare the performance of three regression predictors using a data set provided by the teacher. We use regression predictors studied in the class. We study the data Carefully by reading features (variables), especially the range of reasonable values, meanings, measurement methods, and so on.

2.2 General Data Description

These data are going to be used to predict the house prices. As for a set of data, here are 79 features, and 1 target "SalePrice". We are going to use these feature to train a model and then use the this model to predict the testing data results.

2.2.1 Some statistics about the dataset

This describes the mean, std, min of each numeric features the dataset. It can be seen that the feature "LotArea","YearBuild","YearRemodAdd" has a large sample mean and variance (compared to other feature), which means that in the process of data modeling, those features may need to be normalized to reduce the impact of large variance on the process of model building.

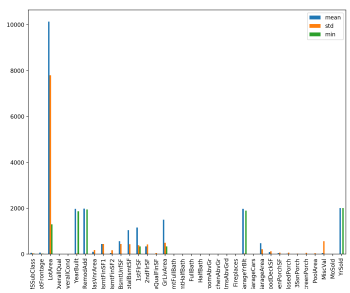

Figure 1: statistics

2.2.2 Missing value rate

As can be seen from this chart, in the data set, whether it is a numeric attribute or a character attribute, there are a large number of missing values(NA), so we need to fill in the missing values in the data preprocessing stage.

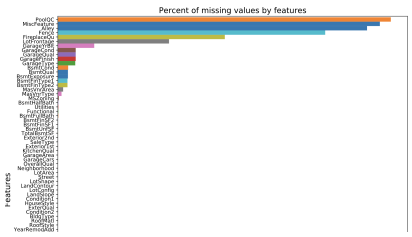

Figure 2: Missing Rate

2.2.3 Data distribution

Most of the data in the dataset is not normally distributed, Such as the "SalePrice" so we need to find some features that deviate significantly from the normal distribution, and perform log data correction to obtain more accurate results in data modeling.

3 Data preparation

Data Preparation is a very important process which would have significant impact on the accuracy, Therefore we need to deal with carefully. In our experiment, we undertook various methods to prepare the data which can be described as following steps below:

- Drop the outliers
- Replace the missing value
- Data Transform (Data type conversion)
- Data Transform (log transform)
- Normalization

3.1 Drop the outliers

3.1.1 What is outliers

The outliers refers to individual values in a sample whose value deviates significantly from the remaining observations of the sample to which it belongs.
For example : This is the the scatter figure with the "1stFirSf" and "SalePrice", the point which being marked is a outlier.
Those outliers will produce some deviation to our model if not being dropped. Therefore in this step we need to find out all the outliers and drop it from the training data set.

3.1.2 How to drop outliers

Since there are very few outliers in this experiment, we chose to manually delete the outliers.

- Step 1 : Draw a scatter plot between all attributes and goals.

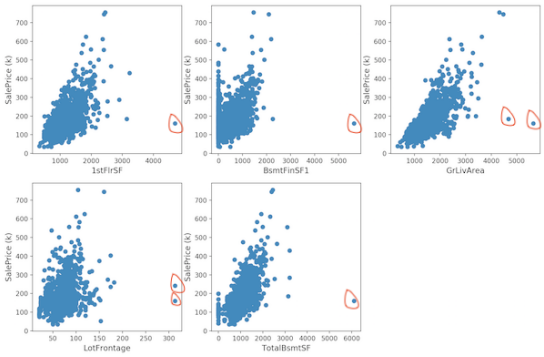

Figure 3: Outliers

- Step 2 : Manually select the figures with an outliers point (observed, you can find that these pictures with outliers points are mostly caused by one point.):
- Step 3 : use the python delete those outliers:

3.2 Fix the missing value

3.2.1 What is the missing value and why should we fix it

Before processing missing data, it is very necessary to understand the mechanism and form of missing data, and the processing of missing values should be analyzed on a case-by-case basis.Because the missing attribute sometimes does not mean the missing data, and the missing itself contains information, it needs to be filled reasonably according to the information that the missing value may contain in different application scenarios. It can be divided into three types.b
Missing completely at random(MCAR): The missing value is random and does not depend on any incomplete or complete variables. The analysis performed on the data is unbiased.
missing at random(MAR): the missing data is related to the observed variables, but independent of the characteristics of the unobserved data.
non-ignorable missing(NIM) or not missing at random(NMAR):The missing value depends on the incomplete variable itself.
Missing values result in incomplete observation data. If it is not completely at random, the generation of incomplete data will affect the results of training and testing.So,we should fix it.

3.2.2 How to fix the missing value

There are two data types in this experiment : numbers and characters

- Type of number means that the value of this features is numeric.
  For example the feature : "1stFlrSF" describes First Floor square feet which is numeric.
- Type of characters represents which types of values the attribute belongs to .
  For example the feature : "ExterQual" describes Evaluates the quality of the material on the exterior.

There should be different treatments for different types of exact values.

- For the missing value with number type :
  For the missing value processing of the numeric type is relatively simple, directly use the mean value to fill the missing value of this feature.
  all_data[numeric_feats] = all_data[numeric_feats].fillna(all_data.mean())

- For the missing value with characters type :
  Comparing numeric types, missing values of character types are a bit difficult to handle, because this type can be divided into the following cases:
  - When there are many missing values for this Features, we have reason to treat the missing value directly as a new classification, that is, replace "NA" with "None", at which point "None" becomes a new classification in this attribute.
    train_data[object_feat].fillna('None', inplace=True)
  - When the missing value of this attribute is relatively small, we don't have to deal with it as a new classification. We can just directly fill this missing value into the mode classification of this attribute.
    train_data[object_feat].fillna(train_data[object_feat].mode(), inplace=True)

3.3 Data Transform (Data type conversion)

3.3.1 Why should we transform the characters to numbers in our regression model?

In the establishment of regression models, we must be cautious in dealing with non-numeric attributes. Non-numeric attributes cannot be directly applied to regression models, so we must convert non-numeric attribute values into numeric attribute values. There are also a variety of conversion methods, we will talk about their pros and cons in the next section.

3.4 How to do Data type conversion

In the data type conversion method, our experiment tried the following two ways.

- In a feature column, replace different character categories with different numbers. For example:
  'GarageCond' : { 'Ex': 1, 'Gd': 2,'TA': 3,'Fa': 4,'Po': 5,}
  With this mapping mechanism, we can convert character categories into numeric categories. But there is a bad point about this method: different categories should have no different in numerical values. In detail, 'Po' can be mapped to 5, 'Po' can be mapped to 1, However there is no reason to explain why 'Po' is bigger then the 'Ex' after mapping. This different in numerical values may result different influences to our model build. Therefore we try another method.

- In a feature column, we take a new column to correspond to one of the original attributes, and then delete the original one. This method is what we use in our experiment.
  For example: After transform: We removed the original column of GarageCond and added 5 columns

GarageCond
Ex

  based on all its value categories. If the original data is Ex, then the new column now corresponds to the GarageCond_Ex column fills 1 and the others fill 0. This approach is obviously more scientific, because in this method have no more different numerical, which is more fair to all the categories

3.5 Data Transform (log transform)

3.5.1 Why should we do the log transform to our data ?

The logarithm of data conversion has the following advantages:
When the magnitude of the independent variables studied is inconsistent, taking the logarithm can eliminate

GarageCond_Ex	GarageCond_Gd	GarageCond_Ta	GarageCond_Fa	GarageCond_Po
1	0	0	0	0

the case where the order of magnitude differs greatly.
Taking the logarithm can eliminate the heteroscedasticity.
Taking the logarithm can transform the nonlinear variable relationship into a linear relationship, which is more convenient for parameter estimation.

3.5.2 How to do the log transform?

In this experiment, the way to do log conversion is very simple, that is, directly put the value in this feature to replace by the Log(x+1) :
Stew the SalePrice
train_data['SalePrice'] = np.log1p(train_data['SalePrice'])
We take the "salePrice" as a example, this is the histogram before we do the log transform:
Similarly, we apply log transformation to other numeric properties.

3.6 Normalize

3.6.1 Why should we do the Normalization ?

Data normalization is the scaling of data into small, specific intervals. In some comparison and evaluation index processing, it is often used to remove the unit limit of data and convert it into a dimensionless pure value, so as to facilitate the comparison and weighting of indicators of different units or orders of magnitude.

3.6.2 How to do the normalization?

In our experiments, the way to do Normalization is also very simple. First, we find the features with large scale of value, such as : "LotFrontage", "YearBuilt", "YearRemodAdd" and do the normalization to all the value with those features.

```
# Norm
all_data['LotFrontage'] = all_data['LotFrontage'].apply(
    lambda x: (x - all_data['LotFrontage'].min()) / (all_data['LotFrontage'].max() - all_data['LotFrontage'].min()))
all_data['YearBuilt'] = all_data['YearBuilt'].apply(
    lambda x: (x - all_data['YearBuilt'].min()) / (all_data['YearBuilt'].max() - all_data['YearBuilt'].min()))
all_data['YearRemodAdd'] = all_data['YearRemodAdd'].apply(
    lambda x: (x - all_data['YearRemodAdd'].min()) / (all_data['YearRemodAdd'].max() - all_data['YearRemodAdd'].min()))
```

Figure 4: norm code

3.7 Train data and test data

In this experiment, we did not manually divide the training data and validation data. Because in the aspect of data verification, we used cross-validation (as explained in the next section), cross-validation can automatically be used for our The data set is divided into train and validation and then verified, and is completely random, and the verification process can specify the round (this experiment is 5 rounds), the accuracy of the model can be described by the mean of the final score of the cross-validation.
For test data, we do the same preprocessing work as train data. After modeling and verification, we can use the model to predict test data, and then submit the results to kaggle to get the ranking.

4 Predictors

4.1 Normal linear regression

4.1.1 What is normal linear regression

In linear regression, we use a linear model to fit our input values.
$\hat{y}(w, x) = w_0 + w_1 x_1 + \cdots + w_p x_p$
the y is the target value of the input vector x, and the $\hat{y}$ is the predict value which compute by our linear model. Therefore our goal is to find the best vector $w = (w_1, \cdots, w_p)$ to give the smallest error between the target value y with the predict value $\hat{y}$.The error function we usually to use in normal linear regression is called square error:$\min_w ||Xw - y||_2{}^2$
Therefore, we can use this error function to find the best W to fit our input to build a linear regression model, once we find the best W, we have built a model and to predict the test data.

4.2 How to normal linear regression in sklearn

sing normal linear regression in Sklearn is very simple, just do as the following steps but additional steps are required for the best parameters (discussed in the next section). reg = linear_model.LinearRegression(alpha = bestalpha)
reg.fit (train_data, target)
reg.predict(test_data)

4.3 Ridge regression

4.3.1 What is rigde regression

Ridge Regression is improved least squares estimation which add the penalty term which is different from the normal linear regress.
Ridge Regression solves some of the problems of ordinary least squares by penalizing the size of the coefficients. The ridge factor is minimized by the sum of squares of the residuals with penalty $\min_w ||Xw - y||_2{}^2 + \alpha||w||_2{}^2$
Among them, $\alpha \geq 0$ is the complexity parameter of the control coefficient shrinkage: the larger the value ofα, the larger the shrinkage, so the coefficient is more robust to collinearity.
Therefore, in this model, We can reduce the impact of small correlations on modeling results. But the impact cannot be reduced to 0, cause this algorithm cannot drop the feature which has small correlation.

4.3.2 How to use rigde regression in sklearn

Using the ridge method on Sklearn is very simple, you only need to implement the modeling and prediction according to the following steps, but additional steps are required for the best parameters (discussed in the next section)
reg = linear_model.Ridge (alpha = bestalpha)
reg.fit (train_data, target)
reg.predict(test_data)

4.4 Lasso regression

4.4.1 What is Lasso regression

Lasso Regression is a linear model for estimating sparse parameters, especially for the reduction of the number of parameters.
When we want to set up a machine learning model, we always want to find as much as feature to estimation, but not all the features are fit for the model. Now we need to reduce the number of feature to get more specific model. Lasso Regression model help us do what we want to do.
Just like the ridge regression, lasso regression also add a penalty to the absolute value of the regression coefficient.
$\min_w ||Xw - y||_2^2 + \alpha(||w||)_1$
The lasso regression is a little different from the ridge regression, which uses absolute values instead of square values in the penalty part. This results in a penalty make some parameter estimates are equal to zero. The larger the penalty value is, the closer the estimate will be to zero. This will help reduce some features, because some of the coefficient will become zero. Also this will cause us to select variables outside of the given n variables.

4.4.2 How to use lasso regression in sklearn

Using the lasso method on Sklearn is very simple, you only need to implement the modeling and prediction according to the following steps, but additional steps are required for the best parameters (discussed in the next section) lassocv = LassoCV(alphas=best_alpha, cv=5, random_state=0)
model = lassocv.fit(X, y)
predicted = model.predict(X)
In our experiments, we need to pass in the data which after preparation, and fit them with their Labels. and just predict our test data.

5 Evaluation

5.1 Evaluation method : Cross Validation

In the process of finding the best parameters, we used the method of cross-validation, and then calculated the cross-validation score of the parameter once in a parameter set. Finally, selecting the best performing parameter is the best parameter.
Cross validation probably means that for the original data we have to divide part of it into train_data and part of it into test_data. Train_data is used for training and test_data is used for testing accuracy. The result of testing on test_data is called cross-validation score. in the process of cross validation we randomly divide train_data and test_data several times and calculate their respective cross-validation score on them.
Therefore by using the cross validation, we do not need to divide the dataset into train_data and the test_data manually, because this algorithm will choose train_data and test_data randomly for several times to evaluate the accuracy of the model.
Here are the steps which cross validation takes

5.2 Evaluation in normal linear regression

In a normal linear regression model, we do not have the penalties to reduce the impact of high-correlation features on model building, therefore, In our experiment we will drop the high-correlation manually, meanwhile we will also implement the model with all the features to see the differences

- Keep all the feature
  we can implement this model by using sklearn, and do the cross validation
  norm = linear_model.LinearRegression()
  norm.fit(feats, target)
  norm.fit(feats, target)
  score = cross_val_score(norm, feats, target, cv=5)
  then we get the final score:

Figure 5: normliner

- Drop some high-correlation features
  we compute the correlation between SalePrice with all the features, and draw the figure to select the high-correlation features to drop We took the features which correlation lower than 0.7 which means that we drop the two features : OverallQual, GrlivArea, There are the result after drop the two features below.
  We got a little improvement in the results, but still no more than 90%.

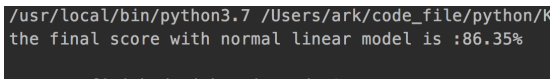

Figure 6: normliner-drop

5.3 Evaluation in ridge regression

There is a penalty in the ridge regression, so unlike the ordinary regression, the design of the best parameter alpha is also designed to represent the effect of the penalty on the modeling result. In the experiment, we first set a flag for all the alphas. Range, then use the loop to calculate the error of each alpha, then select the one with the smallest error as the best parameter. Here is what we do in python when using the cross validation.

```
alphas = np.arange(0.1, 50.0, 0.1)
errors_rigde =[]
for alpha in alphas:
    Ridge = linear_model.Ridge(alpha=alpha)
    Ridge.fit(feats, target)
    errors_rigde.append(np.sqrt(-cross_val_score(Ridge, feats, target, scoring="neg_mean_squared_error", cv=5)).mean
```

We can draw a figure to describe the trend between the alpha with the score.
In this way, we find the best alpha in the ridge regression is 7.8 model and the final score is : 91.88%

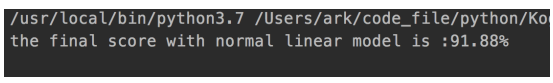

Figure 7: ridge

5.4 Evaluation in lasso regression

There is a penalty in the lasso regression, so unlike the ordinary regression, the design of the best parameter alpha is also designed to represent the effect of the penalty on the modeling result.

```
alphas_to_test = np.linspace(0.00001, 0.01)
errors_lasso = []
for a in alphas_to_test:
    model = linear_model.Lasso(alpha=a)
    model.fit(X, y)
    errors_lasso.append(np.sqrt(-cross_val_score(model, X, y, scoring='neg_mean_squared_error', cv=5)).mean())
```

Figure 8: lasso-code

In experiments, we first set up a list to save the range of alpha, and then use lassoCV function to get the best one to train the model.
We can draw a figure to describe the trend between the alpha with the score
In this way, we find the best alpha in the ridge regression is 0.00042 model and the final score is : 92.32%

```
the final score with Lasso model is :92.32%
```

Figure 9: lasso-resualt

5.5 tate the comparative evaluation and justify the differences

It can be seen from the results that the accuracy of ridge and lasso is higher than that of ordinary linear regression. The fundamental reason is that ordinary linear regression has no penalty, which leads to the result of modeling that is easily affected by unimportant attributes. There are many attributes in the second prediction, so the accuracy of using ordinary linear regression is not very good.
The prediction results of lasso and ridge are also slightly different. Among them, lasso has the highest prediction accuracy. In our understanding, there are too many attributes of data, and lasso can automatically delete attributes that are lower than target-related lines, and ridge cannot delete. It can only reduce the impact, so in the case of too many attributes, lasso will perform better.

Linear Regression is 86.35%
Ridge Regression is 91.88%
Lasso Regression is 92.32%

6 Conclusions

6.1 best parameter

We draw a line graph of the different parameters and final results in the ridge and lasso models to find the best parameters.You can also clearly see the effect on the results between different parameters.

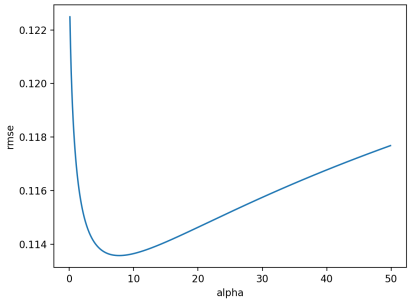

Figure 10: rigde-error

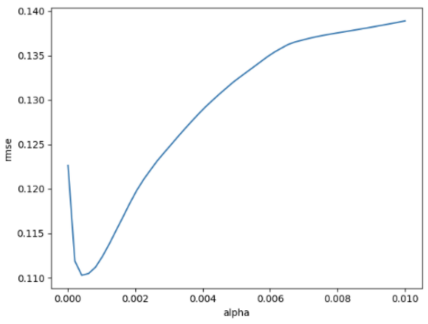

Figure 11: lasso-error

6.2 best subgroup

In normal linear regression, because the model can't choose features by itself, we try to choose the best subgroup to build the prediction model, and other features are discarded. The basis of our choice is that, according to the relevance of each attribute to the target, the attribute whose relevance reaches a threshold can be discarded.We used the selected subgroup to make common predictions, and the accuracy of the results obtained was improved compared to all attributes. Below is a diagram of the correlation values between the various attributes.

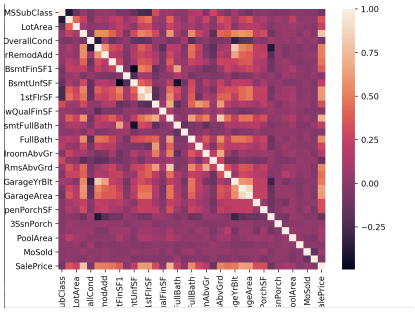

Figure 12: correlation

6.3 comprehensive summary of the three models

- Linear regression are easy to understand and the calculation is not complicated however it is poorly fitted to nonlinear data, very sensitive to outliers

- ridge regression
  Advantages: Ridge regression can solve the problem that the number of features is larger than the sample size. Disadvantages: No feature selection function Ridge regression is used to deal with the following two types of problems: 1.The data point is less than the number of variables 2.There is collinearity between variables

- Lasso Regression
  Advantages:

  1.Lasso regression adds the penalty to the coefficient, it reduces the influences from the coefficient(such as, two coefficient may cause two features cancel each other out)

  2.It gives the model the capacity that can choose the features by itself, at the meantime helping reduce the number of features.

  3.It can calculate much faster than ridge regression.
  Disadvantages:

  1.The alpha in the lasso regression is difficult to choose. If alpha is set too large, the result will close to zero. If alpha is set to small, the result will overfitting.
  Lasso regression is suitable for high dimensional statistics. The sample size is relatively small, but the index is very large, that is, the small N big P problem.

The biggest difference between ridge regression and Lasso regression is that the ridge regression introduces the L2 norm penalty term, and the Lasso regression introduces the L1 norm penalty term. The Lasso regression can make many θ in the loss function become 0, which is better. Yuling regression, because the ridge regression is to have all θ exist, so the calculated amount of Lasso regression will be much smaller than the ridge regression.