Analysis and Prediction of House Price

Author:
Haoyu Shen

September 23, 2021

1 Introduction

House price is becoming more and more expensive in modern society. As purchasing a house might be the biggest challenge that individuals face, knowing better about price contribute to make a better decision. The aim of this project is to predict sale prices of houses using different aspects of houses. This may help the buyers to find the best option with limited budget when they are looking for a new home, meanwhile, this can also help the sellers to find an appropriate price range.

2 Data Cleaning

We have a dataset that contains total of 1460 observations with 81 variables. We split the dataset into two parts: 1) training set, which contains all the variables, and 2) test set that has one less variable which is our predicted variable, SalePrice. Since these two sets of data came from the same original dataset, the consistency is ensured.

We have two different kinds of variables from our data: numerical variables and categorical variables. As we can't apply them to our models without processing, we need to find/understand the structure/ meanings of both variables.

2.1 Numerical variables

We have got a total of 4 types of numerical variables from the description of our data.

- Square footage, such as the size of a bathroom or total square feet of the living area.
- Amounts of rooms, such as how many bathrooms, they are full or half.
- Time, such as when did the house was built, or when was it last sold time.
- Quality and condition, with a rated scale from 1-10.

2.2 Categorical variables

Categorical variables contain label values rather than numerical values, some of them may have a natural relationship to each other like ordering. Unfortunately, most of the data in our dataset are not related to each other and we have to apply one-hot-encoding.

As a first step, each unique category value is assigned with an integer value. This is called the label encoding or the integer encoding and is easily reversible. After that, an one-hot encoding can be applied to the integer representation. This is where the integer encoded variable is removed and a new binary variable is added for each unique integer value.

For instance, 'RoofStyle' has value of 'Gable' and 'Other', so there are two categories and therefore two binary variables are needed. A '1' is placed for the selected 'RoofStyle' and '0' for the other.

Gable,	Other
1,	0
0,	1

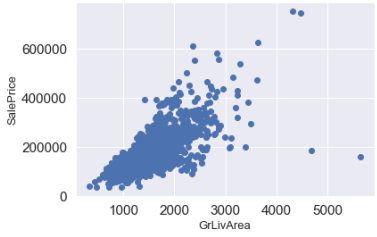
The reason why we apply one hot encoding is we want to include all essential features, no matter numerical or categorical.

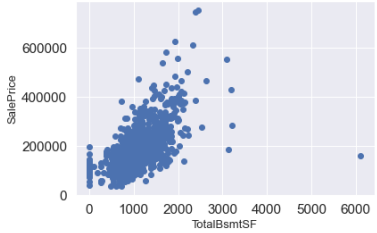
2.3 Outliers

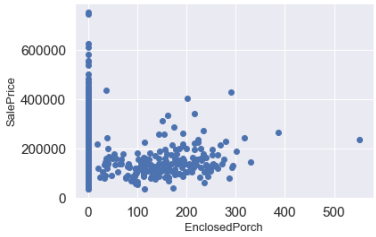
After we know how our data is constructed, we want to remove outliers that may affect our prediction. An outlier is an observation that lies an abnormal distance from other values in a random sample from a population. On the other hand, outliers should be investigated carefully as often they contain valuable information about the process under investigation or the data gathering and recording process.

The following three plots are categories that we figure out with no strong correlation with our regular final SalePrice but with outliers.

- TotalBsmtSF, the total square feet of basement area.
- EnclosedPorch, enclosed porch area in square feet.
- GrLivArea, above grade (ground) living area in square feet.





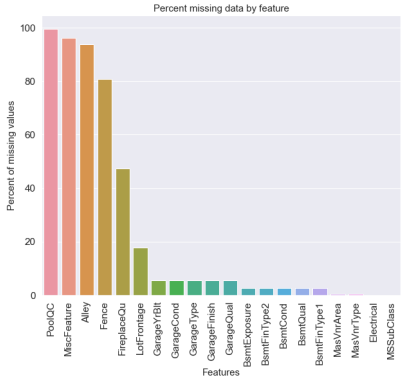


In our case, as we want to predict the final sale price, these outliers don't contain valuable information. Therefore, remove them will help our model to be more accurate.

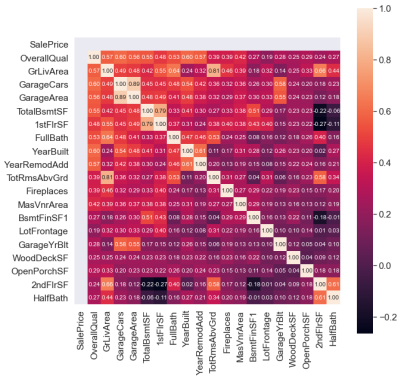
2.4 Missing values

Processing missing values is another essential part when we are doing data preparation as missing values may cause our models to have error or be inaccurate.

	Total No of missing val	% of Missing val	No of unique val
PoolQC	1452	99.656829	3
MiscFeature	1403	96.293754	4
Alley	1366	93.754290	2
Fence	1177	80.782430	4
FireplaceQu	690	47.357584	5
LotFrontage	259	17.776253	110
GarageYrBlt	81	5.559369	97
GarageCond	81	5.559369	5
GarageType	81	5.559369	6
GarageFinish	81	5.559369	3
GarageQual	81	5.559369	5
BsmtExposure	38	2.608099	4
BsmtFinType2	38	2.608099	6
BsmtCond	37	2.539465	4
BsmtQual	37	2.539465	4
BsmtFinType1	37	2.539465	6
MasVnrArea	8	0.549073	325
MasVnrType	8	0.549073	4
Electrical	1	0.068634	5
MSSubClass	0	0.000000	15



As PoolQC, MiscFeature, and Alley have high percentages of missing value that greater than 90%, we will drop all of them. Also, we see there still many features with Nah or value of 0, we replace them with 'None' or median of other house as a placeholder to make all features meaningful.



From the heatmap, the lighter the shaded area is, the stronger the correlation two different features have. Also, as we want to further explore our data, we want to figure out categories with high correlation with both SalePrice and other features therefore we can drop some of the redundant data.

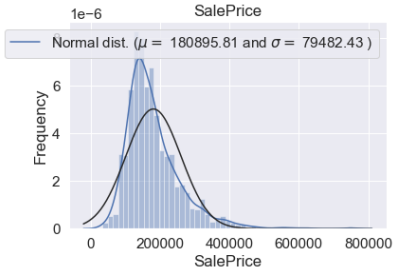
GarageCars and GarageArea are highly correlated. As GarageCars has a higher correlation with SalePrice, we will drop GarageArea.

GrLiveArea is highly correlated with TotRmsAbvGrd. We will remove TotRmsAbvGrd as GrLivArea has a higher correaltion with SalePrice.

TotalBsmtSF is highly correlated with 1stFlrSF. We will remove 1stFlrSF as TotalBsmtSF has a higher correaltion with SalePrice.

3 Target

After data cleaning, we take a look at our target variable, which is SalePrice.



From the plot, we see that our target variable is right skewed which means the average will be higher than the median and as we want it to be a symmetrical distribution, which make the average become accurately lies on the middle, we do log transformation.

3.1 Log Transformation

This time our data seems perfect and ready to be fitted.

4 Models

```
X = features[:labels.shape[0]]
y = labels

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.05, random_state = 0)

print("number of training samples: ", len(X_train))
print("number of test samples:", len(y_test))
```

```
number of training samples:  1384
number of test samples: 73
```

We first split our data set into training set and test test by a proportion 5%. Therefore, we have 1384 training samples and 73 test samples.

```
mlr = LinearRegression()
mlr.fit(X_train,y_train)
mlr.score(X_test,  y_test)
mlr_score = mlr.score(X_test,y_test)
pred_mlr = mlr.predict(X_test)
expl_mlr = explained_variance_score(pred_mlr,y_test)
```

```
tr_regressor = DecisionTreeRegressor(random_state=0)
tr_regressor.fit(X_train,y_train)
tr_regressor.score(X_test,y_test)
pred_tr = tr_regressor.predict(X_test)
decision_score=tr_regressor.score(X_test,y_test)
expl_tr = explained_variance_score(pred_tr,y_test)
```

```
rf_regressor = RandomForestRegressor(n_estimators=100, random_state = 0)
rf_regressor.fit(X_train,y_train)
rf_regressor.score(X_test,y_test)
rf_pred =rf_regressor.predict(X_test)
rf_score=rf_regressor.score(X_test,y_test)
expl_rf = explained_variance_score(rf_pred,y_test)
```

We fitted three models, multiple linear regression, decision tree, and random forest.

Multiple linear regression, also known simply as multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable.

A decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label.

A random forest algorithm consists of many decision trees, and it utilizes ensemble learning.

	Model	Score	Explained Variance Score
2	Random forest Regression	0.912851	0.914046
0	Multiple Linear Regression	0.882565	0.897880
1	Decision Tree	0.761799	0.795456

We selected the best algorithm which is random forest to predict our testing data set and got a score of 0.14808. After this, we want to improve our model by introducing cross validation and combine our models.

5 Improve models

We may have overfitting issue in our model, which is a model that would just repeat the labels of the samples that it has just seen would have a perfect score but would fail to predict anything useful on yet-unseen data.

Therefore, cross validation is required. The k-fold cross-validation procedure is a standard method for estimating the performance of a machine learning algorithm or configuration on a dataset. Repeated k-fold cross-validation provides a way to improve the estimated performance of a machine learning model.

5.1 Cross validation

```
n_folds = 5

def rmsle_cv(model):
    rmse= np.sqrt(-cross_val_score(model, X_train.values, y_train, scoring="neg_mean_squared_error", cv=n_folds))
    return(rmse)
```

We apply cross-val-score function which is the simplest way to use cross-validation where we fitting a model and computing the score 5 consecutive times.

5.2 New Models

```
Svr = make_pipeline(RobustScaler(), SVR(gamma= 0.0003, kernel='rbf', C=15, epsilon=0.008))
score = rmsle_cv(Svr)
print("Support Vector Regression score: {:.4f} ({:.4f})\n".format(score.mean(), score.std()))
```

```
Support Vector Regression score: 0.1851 (0.0142)
```

```
lasso = make_pipeline(RobustScaler(), Lasso(alpha =0.0004, random_state=1))
score = rmsle_cv(lasso)
print("Lasso Regression score: {:.4f} ({:.4f})\n".format(score.mean(), score.std()))
```

```
Lasso Regression score: 0.1125 (0.0112)
```

```
ENet = make_pipeline(RobustScaler(), ElasticNet(alpha=0.004, l1_ratio=.08, random_state=3))
score = rmsle_cv(ENet)
print("Elastic Regression score: {:.4f} ({:.4f})\n".format(score.mean(), score.std()))
```

```
Elastic Regression score: 0.1125 (0.0121)
```

We introduced three new models, which are Support Vector Regression, Lasso Regression, and Elastic Regression. We also can see their score that helps us build our ensemble model.

SVR are supervised learning models with associated learning algorithms that analyze data for classification and regression analysis.

Lasso is a regression analysis method that performs both variable selection and regularization in order to enhance the accuracy of prediction and interpretability of the resulting statistical model.

Finally, the elastic net is a regularized regression method that linearly combines the penalties of the Lasso and Ridge method.

5.3 Ensemble model

```
averaged_models = AveragingModels(models = (rf_regressor, lasso, ENet), weights = (0.25, 0.375, 0.375))

score = rmsle_cv(averaged_models)
print(" Averaged base models score: {:.4f} ({:.4f})\n".format(score.mean(), score.std()))
```

```
Averaged base models score: 0.1116 (0.0125)
```

```
averaged_models.fit(X_train, y_train)
```

```
AveragingModels(models=(RandomForestRegressor(random_state=0),
                        Pipeline(steps=[('robustscaler', RobustScaler()),
                                        ('lasso',
                                         Lasso(alpha=0.0004, random_state=1))]),
                        Pipeline(steps=[('robustscaler', RobustScaler()),
                                        ('elasticnet',
                                         ElasticNet(alpha=0.004, l1_ratio=0.08,
                                                    random_state=3))])),
                weights=(0.25, 0.375, 0.375))
```

```
averaged_models.score(X_test, y_test)
```

```
0.9392208957126431
```

This time we have fitted an ensemble model that calculates the weighted average of each model based on their accuracy. We observed the best accuracy with ensemble the three top-performed models: Elastic Regression, Lasso Regression and Random forest. This gives a public score of 0.12985 on Kaggle Competition.

6 Discussion and recommendation

With the ensemble model, we have a training accuracy of 0.94 and a public score of 0.13. It out performs 80% of other models. However, we can still improve it if we introduce more models and modify the weights of each model in our ensemble model. Stacked regressor model and ridge regression are two good options if we do this kind of analysis again.

Also, we can improve our data cleaning process. We only remove three categories with outliers from our raw data while other categories can also be further cleaned.

From the heatmap of our data, we can see there are more categories that with high correlation with SalePrice can be excluded from the model training process.

To avoid issue of overfitting, we can remove data that is over detailed, or at least combine some of them. For instance, combine the number of half bathroom and full bathroom into one single category as the total amount.