House-prices-advanced-regression-techniques

Contents:

1. Abstract & Introduction…….……....2
2. Research ...……3 - 6
3. Methodology…..…….....7 – 10
4. Evaluation ……10 – 11
5. Conclusion and Future Work ..…….11 - 12
6. References ..…….12

Submitted By:

Shankar Pendse

R00195877

Abstract As part of this project, our main objective is to build a regression model that will predict the 'SalePrice' provided the features of the house or features related to the surroundings of the house. After preprocessing steps (handle categorical features and dealing with missing values and outliers in the given data) different base line models are built using KFold validations on train data and out those only the best performing models are selected for basic experimentation and Research part. The results are discussed briefly in the Evaluation section of this paper. In the end We can see that Simple Linear Regression and Bayesian Ridge Regression models are performing much better than the ensemble techniques such as Random Forest and Gradient Boost regressions

1 Introduction

Dataset is chosen from Kaggle from an ongoing competition at the moment for predicting the house price. This is categorized as advanced regression technique in Kaggle because of the nature of the data set.

Dataset consists of two parts, one for training and another for test. Training data consists of 1460 rows and 81 features (columns/attributes) including the target feature (labeled as "**SalePrice**"). It does not look good to provide the information about each attribute here in the report, I have included a separate file named data dictionary, which helps in understanding what each feature represents.

"**SalePrice**" which is our target feature also called as dependent feature, comes under the category of continuous valued feature, because of the values it has got. For continuous valued features we will be applying regression technique to predict its value.

It will be interesting to see how the provided features helps in determining the SalePrice of the house. All of them may be useful or only few of them, we will see in detail how the features provided are affecting in determining the SalePrice of the house.

2 Research

The dataset consists of total 81 features and around 43 of them are categorical, after handling them and converting them to some meaningful numerical features, we got a total of 199 features.

Since we have more features, feature selection is considered as a research area to see how many features are really important and contributing towards predicting the 'SalePrice'.

Feature selection is an important part while building any regression models, as the performance of our model mainly depends on the features that we select in the end.

Few things to consider with respect to features, while building Machine learning regression models :

1. There should not be multi collinearity (Independent variables should not be highly correlated). This nature has the capability to bring down the model performance significantly.
2. There could be few weak predictors of target variable (in our case target variable is 'SalePrice') which will not contribute to the value associated with the target variable, in this case we have to identify and remove them from our dataset
3. There could be few strong predictors of target variable, and we have to make sure that we are making use of them in our model.

We will consider what is known as **Backward elimination** technique for our research to identify the features which contributing significantly in predicting the 'SalePrice'

2.1 Backward Elimination

This is a technique where initially all features are considered to be contributing in determining/predicting the target variable value ('SalePrice') and iteratively, we will eliminate the features which are least significant, thus reducing the number of features to be used in our model which in turn will reduce the model complexity. (Higher the number of features, more complex the model will be)

Idea behind Backward elimination is to find out how significant the features are in determining the target variable using a statistical analysis known as **P-value.**

We can use OLS regressor from stats model library of python which gives us the summary of the models in the best possible way with P-values of each feature along with the R2 and adjusted R2 Scores of the model.

Steps involved in Backward Elimination:

1. First assume that all the features are important / significant in determining the value for target variable
2. Use all the features to fit the model "Ordinary Least Square' (OLS) which is equivalent of Linear Regression model in scikit learn library.
3. OLS model gives the summary with P-value associated with each feature the R2 and adjusted R2 Score
4. We set the threshold for P-value, and the features which have the P-value greater than the set threshold are removed from the dataset
5. Model is fit using the remaining features and final R2 score is calculated.

2.2 What is P-value

To understand what P-value is, we should know what Null Hypothesis and Alternate Hypothesis is. Let's look at them in the subsections below, but formally introduce what **P-value** is.

P-value is the probability of seeing the effect (E) when null hypothesis is true
P Value = P (E | H0)

2.2.1 Null Hypothesis and Alternate Hypothesis

In regression model, upon having the dataset, we think that all the provided so called independent variable are important/significant in determining the value of the target variable (dependent variable)
Null hypothesis is generally the opposite of what we want. So, in our case,
Null hypothesis is "**All features are not significant in determining the value of target variable**" (It is denoted as H0)

Alternate hypothesis is what we actually want. So, in our case, alternate hypothesis becomes: "**All features are significant in determining the value of target variable**" (It is denoted as Ha)

We basically want to reject the null hypothesis for that we have to find some evidence. (We can not reject anything just like that)

To reject the null hypothesis, **p-value** acts as evidence. As per formal definition of P-value, if we assume null hypothesis to be true, then p value should be some large number. Since it defines the probability, it should be larger than some fractional value between 0 and 1, and this fractional value is called as threshold/significance level denoted as alpha (α).

If P-value is lower than the threshold, we reject the null hypothesis. Let's see what happens if we reject or accept the null hypothesis with respect to the ones we defined earlier.

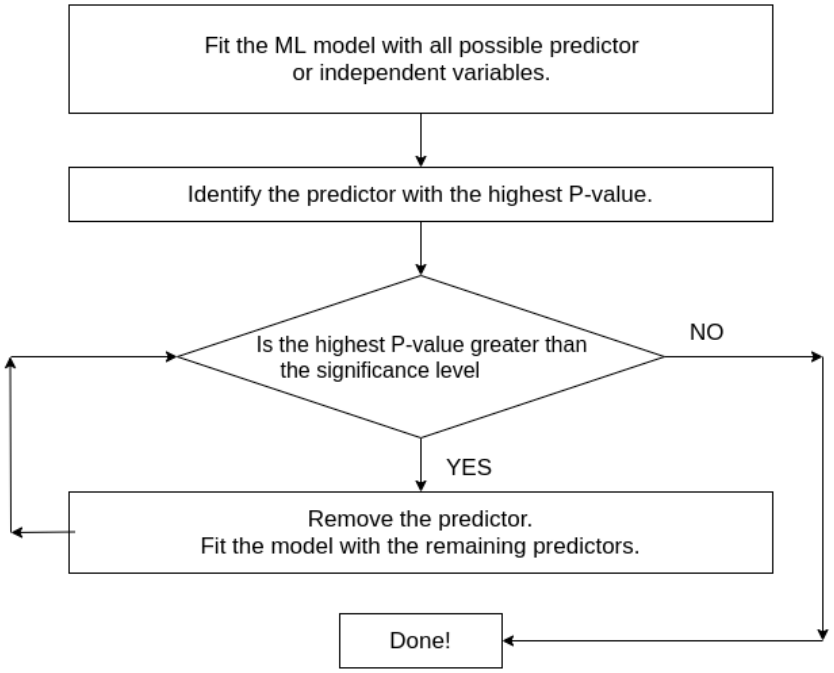
One thing to note here is that, we calculate P-value for each and every feature/attribute

1. If P-value for any feature is greater than the threshold, then our Null hypothesis is true which states that, **the feature in question is not significant/important in determining the value of target variable.** In this case we will remove this feature from our dataset

2. If P-value for any feature is lower than the threshold, then we reject our null hypothesis, and we keep this feature in our dataset

Usually the threshold level (α) or significance level is set to 5% (0.05)

Based on above points, let's have a look at the workflow that we employ in the backward elimination process:



Reference: https://towardsdatascience.com/backward-elimination-for-feature-selection-in-machine-learning-c6a3a8f8cef4

2.2.2 Why this step is important

In any regression problem, the predictors (attributes/features) plays very important role in determining the performance of the model. Few predictors will be negatively impacting, few others will be positively impacting, and few others will not have any impact on model performance.

Features which do not impact the model performance or whose contribution is so less that it is negligible, will simply increase the complexity of the model, and the negative impacting features will bring down the overall model performance.

It is very important to identify the strong, weak and negatively impacting features in our dataset so that we can take necessary action on them in order to have better preforming model

3 Methodology

Let's now detail the steps that we implemented for:

1. Establishing baseline model
2. Basic Experimentation
3. Research

3.1 Establishing baseline model

Below steps are followed to establish few base line models:

1. Preprocessing the data: For any machine learning algorithm, before passing the data for training, we have to prepare the data to be suitable for the algorithm. All algorithms will require the data to be in numerical form. There are different steps to complete the preprocessing the data, and some of them are listed below for the regression problem:

1. Handling Categorical Data
2. Handling Missing values
3. Scaling the data
4. Dealing with outliers

Now let's look at each of the above-mentioned steps one by one in detail with respect to our dataset:

1.1 Handling Categorical Data: Our data set has total of 81 features initially, out of which 43 are categorical (Ordinal and Nominal), depending on the description provided in the data dictionary, we have treated few attributes as Ordinal even though they are nominal in nature and used appropriate values to encode them into numbers. Few are purely nominal and they have been one-hot encoded using pandas get_dummies() method.

1.2. Handling Missing values: For missing values in training data, for few attributes we just used simple imputer with "mean" as strategy, and the same thing is applied on test data. For few attribute/s where the missing values or the attribute itself was not that important (determined from data dictionary provided), those attributes are dropped from the training data. As we have a separate test data set, there was need to check for null values in that dataset as well and they are imputed accordingly

1.3. Scaling the data: It is important to scale the features which vary largely, so that the machine learning model does not get biased towards such features. We

initially have made use of MinMax scaler which normalizes the data (each feature where this scaler is applied will have the values ranging between 0 and 1). This will preserve the shape of the data (no distortion from the original shape of the data)

1.4. Dealing with outliers: For any machine learning model, if there are outliers, it has negative impact on model performance. In other words, the model predictions will be shifted far from the original values (there will be greater variance).
We have made use of IsolationForest algorithm which is a machine learning algorithm to determine the outliers and we are just removing them from our train dataset. This has resulted in removal of 15 records from train data which were identified as outliers.

One thing to note here is that, we have carried out outlier detection after scaling our dataset, because Isolation Forest is a machine learning algorithm and machine learning algorithm performs better when data is scaled

2. Building Different Models: 6 different models (Linear Regression, KNN Regression, Support vector Regression, Bayesian Ridge Regression, Random Forest Regression and Gradient Boost Regression) have been built with default parameters using Kfold with 10 splits. Steps followed:

1. Split the train data into train and test set using train_test_split method of sklearn library.
2. Use train data with Kfold validation using 10 splits
3. Return all 10 models for each of 6 different models along with the scores of each of the 10 models individually

3. Hyper Parameter Tuning: Out of the above 6 different models, Linear Regression, Random Forest Regression and Gradient Boost Regression were the best performing models and they were selected for Hyper Parameter Tuning. Please note that, Linear Regression model does not have any hyper parameters to tune, so we did tune only Random Forest and Gradient Boost regression models. GridSearchCV is used to search for the right parameter selection

3.1 Random Forest Regression: This is one of the ensemble methods where the base models will be decision Trees and n_estimator is one of the parameters which decides how many decision trees will be constructed. The ensemble methods in general are said to be good in avoiding overfitting, so n_estimators can be high in number. Each decision tree will be built independently using different samples for training, with different subset of features. The final result will be the aggregation of all decision trees built.

Below parameters are used for tuning the best Random Forest regression model obtained from "**step 2 Building Different models**"

```
def param_tuning_RFM(BestRFM, train_data):

    RF_parameters = {"n_estimators"      : [1000,2000,3000],
                     "max_features"      : ["auto", "sqrt", "log2"],
                     "max_depth"         : [4,6,8,None],
                     "min_samples_split" : [2,3,4],
                     "warm_start"        : [True, False],
                     "random_state"      : [195877],
                     }
    RF_grid = GridSearchCV(BestRFM, RF_parameters, n_jobs=-1, cv=5, verbose = 10, return_train_score=True)
    RF_grid.fit(train_data.drop('SalePrice', axis = 1), train_data['SalePrice'])
    return RF_grid
```

3.2 Gradient Boost Regression: This is another Ensemble technique but instead of building the trees independently, each tree is built one at a time in such a way that previous weak learners are improved in next successive trees built and also results are combined right from the first tree all along the way unlike random forest regressor where the results are combined at the end. Below parameters are used for tuning the best model obtained from step 2

```
def param_tuning_GBM(BestGBM, train_data):
    GB_parameters = {'learning_rate': [0.01,0.02,0.03,0.04],
                     'subsample'    : [0.9, 0.5, 0.2, 0.1],
                     'n_estimators' : [1000,2000,3000],
                     'max_depth'    : [4,6,8,10,None],
                     'random_state' : [195877]
                    }
    GB_grid = GridSearchCV(BestGBM, GB_parameters, n_jobs=-1, cv=5, verbose = 10, return_train_score=True)
    GB_grid.fit(train_data.drop('SalePrice', axis = 1), train_data['SalePrice'])
    return GB_grid
```

3.2 Basic Experimentation:

We have used all features in building the baseline model and used Min max scaler for the features varying largely. In this basic experimentation, we will do the following steps:

3.2.1 Feature selection: We will select the features which are highly correlated with our target variable 'SalePrice'. Correlation identifies the linear relationship between the selected variables. For linear regression we should not be having the features which are internally highly correlated. So have first removed the least correlated features with respect to 'SalePrice' and next we have removed the features which are highly correlated

3.2.2 Using Standard Scaler instead of MinMax scaler: In our initial model building, we used MinMax scaler to scale the features which vary largely (In other

words we normalized the values for those features). As part of basic experimentation, we will make use of StandardScaler from scikit learn library to scale the values. Standard Scaler scales the values to have unit variance. We really do not know initially which scaling method is best suited for our model. So, we will have to try them both, unless we are sure about the data and its distribution

3.2.3 Rescaling the target variable (SalePrice) to have normal distribution: Machine learning model performance improves if we rescale the dependent/target variable, so it is always better to get the shape of the data in normal form before we train our Machine learning model. The same has been done using log transformation of NumPy module. Which transforms the data by applying log(1+x) to all the elements of the column (in this case our target variable SalePrice)

4 Evaluation

4.1 Initial Model Building : As articulated in section 3.1 all the steps are implemented and below are the results obtained : (**Below results can be referenced in the provided .ipynb under the heading Initial model building**)

Before Hyper Parameter Tuning		
SL/NO	MODELS	Kfold Best Model Score
1	Simple Linear Regression	0.911
2	KNN Regression	0.7555
3	Support Vector Regression	0.4472
4	Bayesian Ridge Regression	0.9266
5	Random Forest Regression	0.9165
6	Gradient Boost Regression	0.9321

Highlighted ones are our best models out of which Random Forest, Gradient Boost and Simple Linear regression models are considered for further tuning. Scores after parameter tuning are shown below.
Below results can be observed in the provided .ipynb at the cell number 33-34 and 40-41

After Parameter Tuning			
SL/NO	MODELS	TRAIN R2 SCORE	TEST R2 SCORE
1	Random Forest	0.9838	0.9105
2	Gradient Boost	0.9995	0.9431

As there are no hyper parameters to tune for Linear regression model, not showing the results again here. But it will be considered later during basic experimentation and for research part as well.

4.2 Basic Experimentation results : As detailed in the section 3.2, feature selection is done based on correlation and also different scaling approach has been implemented, the results are as shown below :

Below results can be observed in the provided .ipynb starting from cell number 143 through 155

SL/NO	MODELS	TRAIN R2 SCORE	TEST R2 SCORE
1	Random Forest	0.9827	0.882
2	Gradient Boost	0.999	0.904
3	Linear Regression	0.9225	0.9094
4	Bayesian Ridge Regression	0.9206	0.9111

We can see that Random Forest performance has gone down after our basic experimentation, but Linear Regression and Bayesian Ridge Regression are performing much better than Gradient Boost and Random Forest Regressor.

4.3 Research Results: As detailed in the section 2, we selected features using Backward elimination technique using P-value as our basis to decide which features to discard. Below are the results of the different models after using the features based on backward elimination approach:

Below results can be observed in the provided .ipynb starting from cell number 171 - 190

SL/NO	MODELS	TRAIN R2 SCORE	TEST R2 SCORE
1	Random Forest	0.9816	0.8801
2	Gradient Boost	0.9991	0.9177
3	Linear Regression	0.9236	0.9005
4	Bayesian Ridge Regression	0.923	0.9014

5 Conclusion

1. While establishing base line models, we are able to improve the performance of only Gradient Boost regression model using hyper parameter tuning techniques. Random forest did not show any improvement, there is slight reduction in the score. Though the scores are above 0.9, both the models are overfitting the training data, but better model is Gradient boost as the difference between train and test R2 score is less compared to Random Forest

2. After basic experimentation, performance of both Random Forest and Gradient boost models have reduced by considerable amount, but Simple Linear Regression and Bayesian Ridge regression models are performing much better when we compare their Train and Test score (there is no much difference)

3. After using backward elimination technique to select the features, we did not see any improvement in the performance of Random Forest and Gradient boost models, but Linear regression and Bayesian Ridge regression models have been performing much better than the ensemble techniques. This might be because we could be eliminating a few important features during the process

Future Work: All the below feature works are proposed towards improving the model performance

1. We can try using different value for threshold of P-value while eliminating the features using backward elimination technique
2. We can try out Forward selection technique to see if it helps in improving the model performance
3. We can try deriving new features from vast set of features available (feature engineering) One such example is to include a new feature "age" using year built and year of sale
4. We can investigate the categorical attributes and drop few after encoding using get_dummies() to do away with dummy trap

References:

1. https://www.kaggle.com/ashishsaxena2209/step-by-step-regression-backward-elimination

2. https://medium.com/@abhinav.mahapatra10/ml-basics-feature-selection-part-2-3b9b3e71c14a

3. https://www.datasciencecentral.com/profiles/blogs/decision-tree-vs-random-forest-vs-boosted-trees-explained

4. https://towardsdatascience.com/backward-elimination-for-feature-selection-in-machine-learning-c6a3a8f8cef4