LSE Department of Mathematics

Summative Project

House Prices: Feature Engineering and Regression Techniques

Candidate ID: 33815 38816

Course: MA429 Algorithmic Techniques for Data Mining

Department of Mathematics

April 2020

Contents

1 Executive summary

Different factors involves in assessing house values and it is meaningful and interesting to find out the most important ones from **79 features** of a house and produce as accurate prediction as possible with different features using data mining techniques. This is an ongoing Kaggle competition and we mainly focused in **feature engineering** and **regression modeling** to tackle the problem. Data preprocessing and exploratory data analysis were first conducted to explore the target variable and most correlated predictors with intuitive visualizations on a tidy data set. Quick-built models like random forest and linear regression then helped to prove our initial thoughts in creative feature construction that benefits the models performance. After setting up different well-tuned single models, **Elastic-Net regression, Random Forest, SVM and XGBoost**, and then comparing their results we found that a linear regression with regularization term has **RMSLE of 0.1127** and beats all the other single models and has higher interpretability and training efficiency.So a simple linear regression model is fair enough in prediction among all the single models while **ensemble method stacking** combining 3 single models can even improve the results further,bringing down the **RMSLE to 0.1095**. By extracting features importance from different models, we drew an conclusion that general features such as overall quality and total size are more significant than features of any specific facilities, for example, having a pool or not. While specifically, basement, garage and kitchen are the most decisive facilities in terms of house pricing. For location factor, the obvious differences can only be seen in some neighbourhoods (e.g. fancy neighbourhoods). The results given by our final ensemble model ranked us at top 12% among over 4700 teams in this competition.

2 Introduction

What are the most important factors influencing the house prices? With these factors, how accurate can we predict the house prices? This is an ongoing kaggle competition and we took part in this competition as our MA429 final project to gain deeper understanding and practical experience in data mining. As kaggle describes:

"Ask a home buyer to describe their dream house, and they probably won't begin with the height of the basement ceiling or the proximity to an east-west railroad. But this playground competition's dataset proves that much more influences price negotiations than the number of bedrooms or a white-picket fence.

With 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa, this competition challenges you to predict the final price of each home."

Our goal is to apply what we've learnt in MA429 and by self-study to get higher position in the competition (i.e. lower Root Mean Squared Logarithmic Error (RMSLE) since RMSLE is the score of this competition). And this project mainly focuses on **feature engineering** and **regression techniques** implementation to achieve that.

3 Preliminary analysis and Pre-Processing

Before beginning the task of predicting the house price, it is essential to have a good understanding of the data set. Particularly, getting a view of different patterns for house price could be useful.

3.1 Data Imputation and Transformation

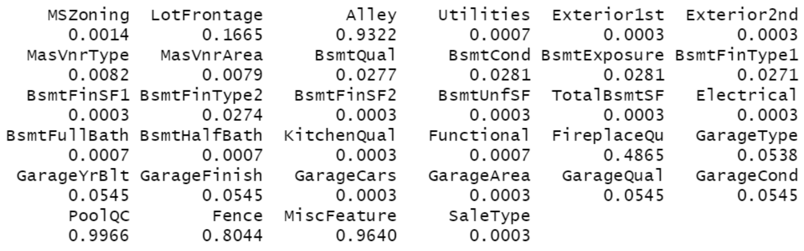
3.1.1 Loading and Merging Data

House pricing data set consists of 1460 and 1459 observations in training and test data respectively, and has 80 features to predict the final sale price. Among these features, 43 are character and 37 are numerical. While quite a few of the character variables are actually categorical variables with order and can therefore be transformed to numeric to reserve the ordinal information and the others can be transformed into normal factor variables for one-hot encoding or other encoding methods depending on specific model implemented, we provided more details of the preprocessing steps below. Roughly, the features can be classified into 4 groups (quality and condition of different facilities, location, space and time-series).

Kaggle has divided the data into training and testing set for us so we do not need to manually do that, while in order to conduct the preliminary analysis and preprocessing, we should merge them together and redivide it into the original training and testing set before modelling in order to submit testing set prediction on Kaggle. It is also worth noting that the testing set does not contain the values of target variable **SalePrice** so that we have no access to the actual testing error before submission and we need to apply k-fold cross-validation to estimate it.

3.1.2 Handling Missing Values

Firstly, we removed the ID column in the data frame and stored it separately for submission, since this is just a number and contributes nothing to the price. Then we started to deal with the missing values. In the original data set, we have 15424 missing values represented 6.6% of the total data. If we delete all the missing values, the number of observation will decrease to 0 ! That is to say every observation has at least one feature missing! So it is significant and very necessary to impute the missing values. To analyse missing values in each column, we can find that there are 34 features have missing values:



```
    MSZoning  LotFrontage        Alley    Utilities  Exterior1st  Exterior2nd
      0.0014       0.1665       0.9322       0.0007       0.0003       0.0003
  MasVnrType   MasVnrArea     BsmtQual     BsmtCond BsmtExposure BsmtFinType1
      0.0082       0.0079       0.0277       0.0281       0.0281       0.0271
  BsmtFinSF1 BsmtFinType2   BsmtFinSF2    BsmtUnfSF  TotalBsmtSF   Electrical
      0.0003       0.0274       0.0003       0.0003       0.0003       0.0003
BsmtFullBath BsmtHalfBath  KitchenQual   Functional  FireplaceQu   GarageType
      0.0007       0.0007       0.0003       0.0007       0.4865       0.0538
 GarageYrBlt GarageFinish   GarageCars   GarageArea   GarageQual   GarageCond
      0.0545       0.0545       0.0003       0.0003       0.0545       0.0545
      PoolQC        Fence  MiscFeature     SaleType
      0.9966       0.8044       0.9640       0.0003
```

Figure 1: Features with missing values(%)

Fortunately the data description file shows that most of the NAs in the data set are not "truly missing" but mean do not have that attributes. For example, NA of **BsmtQual** means "No Basement" and NA of **MiscFeature** simply means no extra miscellaneous features. So we just filled these NAs with "None" to show that this house do not have that specific attributes and R code will see it as a separate category, similar to NA encoding. Also, it is reasonable to fill the NAs in area or number or quality of these facilities related features with "0", since they do not have that.
On top of that, for most "truly missing" values of categorical variables, we imputed them with the most common values in this attributes. Actually, for some attributes, the data description file tells us "assume typical type unless ..." and thus it is reasonable to do this imputation. In particular, the variable **LotFrontage**(Linear feet of street connected to the house) has 486 missing and it is numerical. Since the area of each street connected to the house property most likely have a similar area to other houses in its neighborhood , we filled in missing values by the mean **LotFrontage** of the neighborhood. Furthermore, it's worth noting that we need to compute the mean values of training data and test data separately to avoid data leakage. For full details and whole process of imputation of missing values, please refer to the Rmd file.

3.1.3 Categorical or Numerical

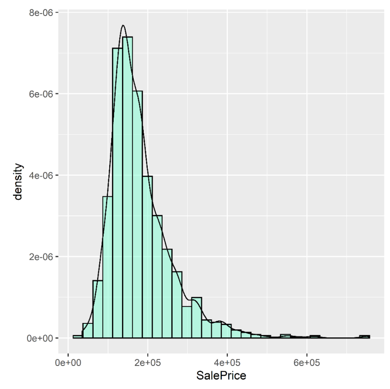
In this part, we transformed the categorical variables that contain order information into numeric ones. By checking the description file of the data, we knew that some features are ordinal. For example, **ExterQual** (Evaluates the quality of the material on the exterior) has 5 classes : Excellent, Good, Average/Typical, Fair, Poor. It is obvious that excellent is much better than poor so it is reasonable to assign order numbers (5 for excellent and 1 for poor in this case) to represent these classes reserving the ordinal information.
Similarly, features: **Utilities**, **LandSlope**, **BsmtQual**, **HeatingQC**, **KitchenQual**, **Functional** and others similar can also be transformed into numerical features.
However, not all the number of numerical variables have meaningful order information. For instance, **MSSubClass** is numeric in the original data set, it represents the building class and contains 16 different types and it uses different number to represent them, like 85 represent "SPLIT FOYER". While these numbers do not contain ordinal information so we just transformed it back to normally categorical feature for later one-hot encoding. The data set also contains some time-series columns and we will handle these variables later in another way after exploring their trends in the next section.

3.2 Exploratory Data Analysis

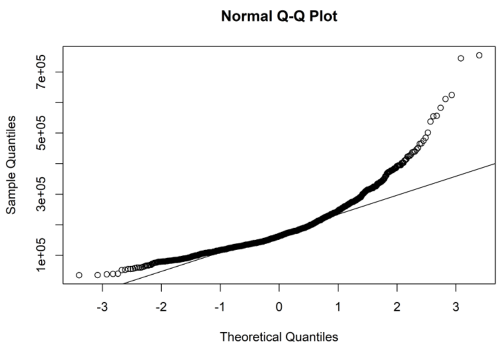
Exploratory Data Analysis is one of the most important part in data mining, it can not only give us some straightforward insights from the data but also can provide some guidelines on how to do the further feature engineering and modelling. Here we tried to explore the statistics characteristics of different variables and how they affect each other sing some intuitive visualizations. We mainly focus on those that have higher correlation or importance of the target variable **SalePrice**.

3.2.1 Explore Target Variables

To begin with, let's see the distribution of the response variable **SalePrice**.



(a) Density



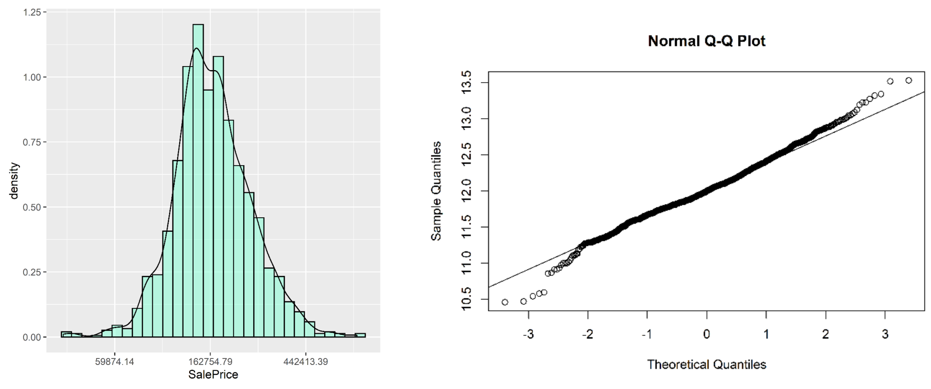
(b) Normal Q-Q plot

Figure 2: SalePrice

As we can see, the sale prices are right skewed and far away from a normal distribution. This can be explained by only few people can afford very expensive houses. Logarithmic transformation is sensible to apply here, it can not only make the target variable look more 'normal' (normality of response variable is an important assumption of some model such as linear regression, it helps the model more robust to predict unseen data) but also manually let the later models to minimize the RMLSE which is the metrics on Kaggle (RMSLE is usually used when we don't want to penalize huge differences in the predicted and the actual values when both predicted and true values are huge numbers). Formula of RMLSE is shown below:

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(log(\hat{y}_i + 1) - log(y_i + 1))}$$

Moreover, if a house has more rooms or in a better area that will probably multiply the house's value, not add to it. It is therefore sensible to conduct logarithmic transformation on the response variables. Plotting after the transformation is shown below:



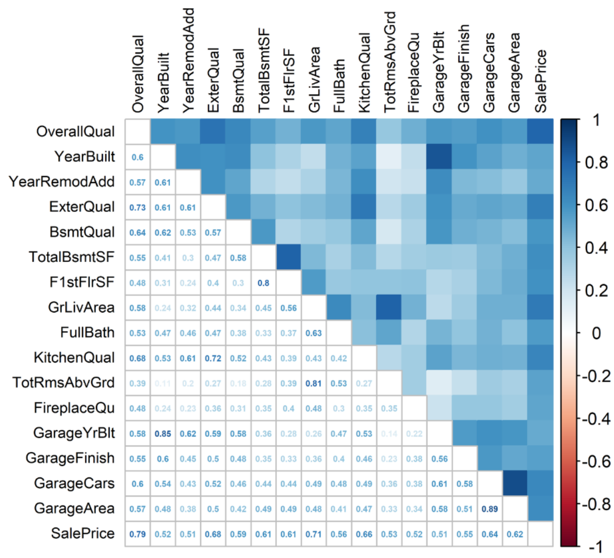
(a) Density (b) Normal Q-Q plot

Figure 3: log(SalePrice)

And we can see, after taking logarithm, the **SalePrice** follows almost a normal distribution.

3.2.2 Explore Highly correlated Numerical Variables

Then we explored the predictors that are highly correlated with the reponse variable **SalePrice** since they are the most powerful candidates on predicting **SalePrice** and a correlation matrix is created to show how the numeric features correlate with our response variable.

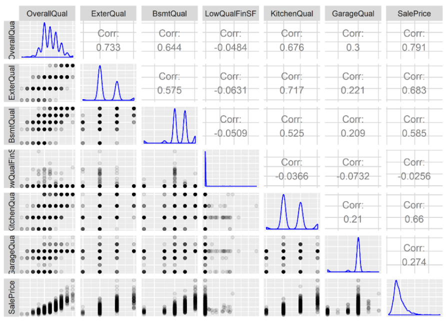


	OverallQual	YearBuilt	YearRemodAdd	ExterQual	BsmtQual	TotalBsmtSF	F1stFlrSF	GrLivArea	FullBath	KitchenQual	TotRmsAbvGrd	FireplaceQu	GarageYrBlt	GarageFinish	GarageCars	GarageArea	SalePrice
OverallQual																	
YearBuilt	0.6																
YearRemodAdd	0.57	0.61															
ExterQual	0.73	0.61	0.61														
BsmtQual	0.64	0.62	0.53	0.57													
TotalBsmtSF	0.55	0.41	0.3	0.47	0.58												
F1stFlrSF	0.48	0.31	0.24	0.4	0.3	0.8											
GrLivArea	0.58	0.24	0.32	0.44	0.34	0.45	0.56										
FullBath	0.53	0.47	0.46	0.47	0.38	0.33	0.37	0.63									
KitchenQual	0.68	0.53	0.61	0.72	0.52	0.43	0.39	0.43	0.42								
TotRmsAbvGrd	0.39	0.11	0.2	0.27	0.18	0.28	0.39	0.81	0.53	0.27							
FireplaceQu	0.48	0.24	0.23	0.36	0.31	0.35	0.4	0.48	0.3	0.35	0.35						
GarageYrBlt	0.58	0.85	0.62	0.59	0.58	0.36	0.28	0.26	0.47	0.53	0.14	0.22					
GarageFinish	0.55	0.6	0.45	0.5	0.48	0.35	0.33	0.36	0.4	0.46	0.23	0.38	0.56				
GarageCars	0.6	0.54	0.43	0.52	0.46	0.44	0.44	0.49	0.48	0.49	0.36	0.38	0.61	0.58			
GarageArea	0.57	0.48	0.38	0.5	0.42	0.49	0.49	0.48	0.41	0.47	0.33	0.34	0.58	0.51	0.89		
SalePrice	0.79	0.52	0.51	0.68	0.59	0.61	0.61	0.71	0.56	0.66	0.53	0.52	0.51	0.55	0.64	0.62	

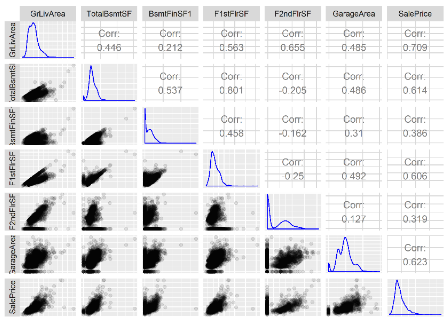
Figure 4: Correlation of Numerical Variables

In the correlation matrix, the number on the bottom left is the magnitude of the correlation, and the colors on the upper right indicate the levels of correlation, the deeper the more correlated. By observing the figure, the top two relevant features of **SalePrice** are **OverallQual** and **GrLivArea**, which is overall quality of houses and ground floor living area square feet respectively. Not surprisingly, quality and size of a house should be two of the most important factors that determine the house prices. But it is also worth noting that, many predictors are also highly correlated, i.e., multicollinearity would be a problem in some models such as linear regression. Moreover, we found that there are quite a lot variables describing the quality and size of different facilities in the house and it is very likely that they are highly correlated because a bigger and nicer house are more likely to have bigger and nicer kitchen and so on. Thus, we created two pairs plot to visualize their correlation.

Quality & Area related variables against **SalePrice**



(a) Quality related variables against SalePrice



(b) Area related variables against SalePrice

Figure 5: Pair plot

As we can see, most of them are positive correlated. We kept this fact in mind and would apply methods to deal with multicollinearity issue in the modelling part (e.g. pca and non-parametric methods can be tried). Overall quality has the highest correlation with SalePrice among numeric variables, it rates the overall material and finish of the house on a scale from 1 (very poor) to 10 (very excellent).

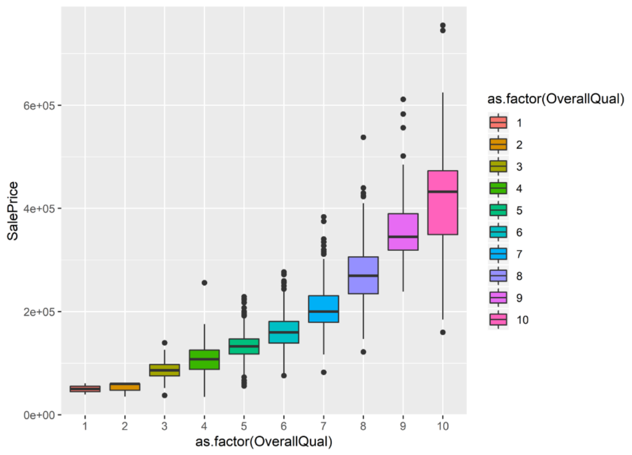
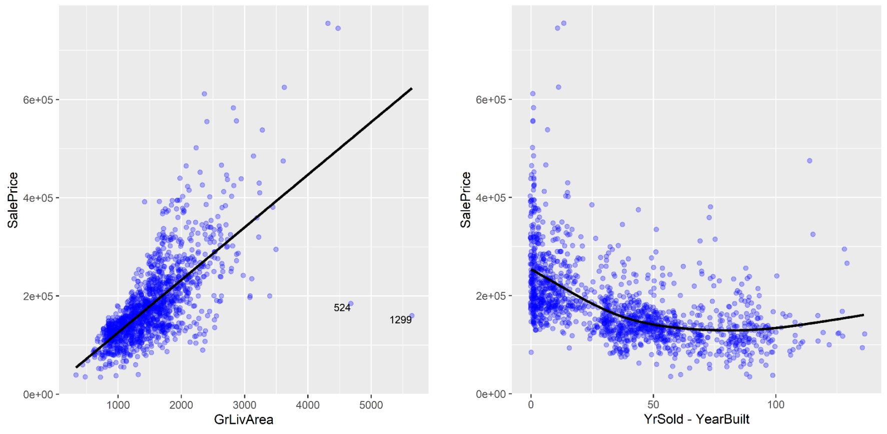


Figure 6: OverallQual against Saleprice

Except some outliers, it is evident that a house with higher overall quality tend to be sold at a higher price.

Following Figure 7(a) is the scatter plot of **GrLivArea**(Above Ground Living Area) against **SalePrice**, which is the second highest predictors correlated with **SalePrice**.

It makes a lot of sense that bigger house worth more money. Besides, it is very clear that there are 2 outliers in the plot. After checking their other attributes, we found that they are actually good house but sold at low price and we thus removed them from the training set.



(a) GroundLivingArea against Saleprice (b) Age of the house against Saleprice

Figure 7: Scatter plot

YrBuilt is also another important factor as the correlation matrix shows, and as we've mentioned in the previous section, year related variables are all time-series data that may not fit in the normal regression model. Here we differ the year of sold and the year built to represent the age of a house, and see how **SalePrice** differs among distinct ages. The picture 7(b) shows that new houses tend to worth more and this roughly coincide the saying "loving the new and loathing the old". While it is notable that some old houses also sold at high price and this may because many mansions are aged. We also apply this difference to other time-series variables and we found the absolute value of correlation between these variables and response variable maintain unchanged or are even higher.

3.2.3 Explore Important Categorical Variables

Though correlation matrix can reflect the importance of predictors in terms of their correlation with target variables, while there are two main defects doing so.

1. Pearson Correlation can only reflect linear correlation, while many relations among variables are always nonlinear in practice.

2. Pearson Correlation can only be calculated among the numerical variables, and thus we need to apply other methods to figure out the importance of categorical variables.

Tree-based methods can be implemented to complement the correlation analysis, e.g. random forest can use permutation method to see how the loss function varies before and after each variable being permutated (takes the idea of permutation test). Also, random forest can find more complex relations among variables because it grows deeper tree compared with boosting tree methods[1] Here we just want to have a glance at the variables importance but not to get the best prediction, so we do not need to conduct an elaborate tuning of random forest in this step.

After training a random forest model, we can get the importance of categorical features. Following is the table of top 5 features and their importance value (after scaling to 0-100).

Feature	Neighbourhood	MSSubClass	GarageType	HouseStyle	MSZoning
Importance	46.42811857	6.56368104	6.39740859	2.81830255	2.33251120

Table 1: Categorical features importance

From the splitting algorithm's point of view, all the dummy variables are independent. One-Hot encoding hurts the performance of tree-based models so it is important to keep all the original categorical variables in the model instead of performing one-hot encoding. Actually, random forest does not require all numeric input and we did try that the RMSE of random forest model would increase after conducting one-hot encoding in this data set[2].

According to the importance of categorical variables above, the top 3 important categorical variables are **Neighborhood**, **MSSubClass** and **GarageType**.

We first have a look at the relation between **Neighborhood** and **SalePrice**.

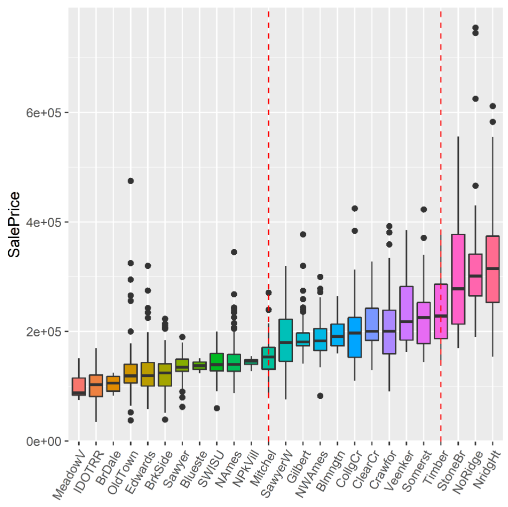
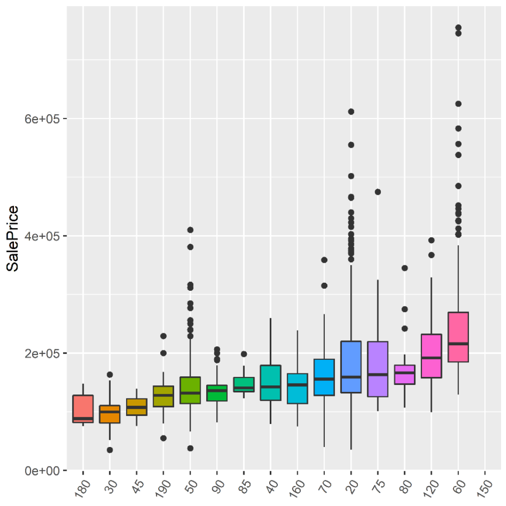
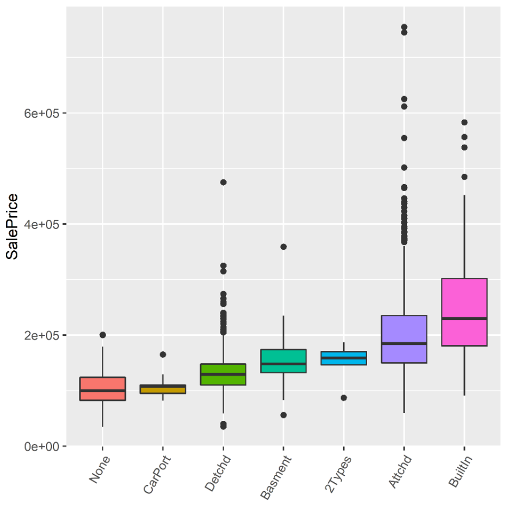


Figure 8: Neighborhood against SalePrice

We can see that the **SalePrice** distribution of different neighbourhoods are quite different and this is reasonable since location is always one of the most important factor of houses prices. Specifically, different neighbourhoods can be roughly clustered into 3 groups considering their houses **SalePrice** distribution and this can be a potential new feature substituting for neighbourhood and we'll try this in the next section.



(a) **MSSubClass** vs **SalePrice**



(b) **GarageType** vs **SalePrice**

It is also reasonable to see that a house with a built in garage is generally much more expensive than a house without a garage.

To summarize, the EDA part tells us that quality, size, location and building type are the crucial factors deciding house prices and this verifies our general knowledge.

4 Feature Engineering

Wikipedia defines feature engineering by "Feature engineering is the process of using domain knowledge of the data to create features that make machine learning algorithms work. " In this part, we mainly focused on creating some new features that have more predictive power in potential by combining the existing predictors. We used a simple lasso regression and random forest to test whether the 5 fold cross-validation error decrease after adding the new features, if yes, we add the new feature and delete the original combining variables, otherwise we maintain the data set unchanged and test other new features. We chose lasso regression and random forest to test our features selection because they are two completely different models, some features useful for linear regression may not work for tree-based model and thus this can give us more confidence on choosing better features. The feature engineering methods mainly come from HJ van Veen slides.[3]

1. **Transform All the Year-related Variables to Age**

   Time series data is hard to analyse (autocorrelation, time trend etc.) and may need to conduct analysis separtely. If we do not want to contain time series data in the normal machine learning model, we can transform time series data into meaningfully numerical data, for example, we can use the difference between **YearSold** and other year values to represent the building age. In this case, it is reasonable to do that as we have seen the negative correlation between age and **SalePrice** in the last section (EDA Figure 7(b)).Here, we substituted the **YearBuilt**, **YearRemodAdd** and **GarageYrBlt** by the "age" of them, say, their difference between themselves and **SalePrice**.

2. **Create a new feature to represent basement related variables** Since Basement has 7 numerical features and they are not equally important, it's a good choice to combine

these features and make it more powerful in response variable prediction. So we created a new feature called **TotalBsmtQual** representing the total rating of basement taking their different qualities and sizes into account. The formula is given below:

$$\begin{aligned} TotalBsmtQual = 0.735 * ToBsmtSF * BsmtQual + 0.452 * BFinType1 * BFinSF1 \\ + 0.016 * BFinType2 * BFinSF2 + 0.222 * BsmtUnfSF \end{aligned} \tag{1}$$

We used their correlation coefficients with **SalePrice** as the combination coefficients

3. **Split Neighborhood into 3 ordinal classes** Through Figure 8, we can clearly see the neighborhood can be split into 3 classes of high, median and low sale price separated by red dash lines. This become an ordinal categorical feature and by foregoing approach, we assigned order number to them. However, that didn't give us a better result after running random forest model and lasso regression so we decided to keep the **Neighbourhood** variable as its origin.

4. **Aggregate total bathrooms** There are 4 bathroom features. Individually, they are not very important. However, after aggregating them into one feature to represent the total number of bathrooms, it became a more powerful variable. Since half-bath does not possess all the facilities, we multiple 0.5 when aggregating.

$$TotalBathrooms = FullBath + 0.5 * HalfBath + BsmtFullBath + 0.5 * BsmtHalfBath \tag{2}$$

5. **Sum the area of 1st floor and 2nd floor** Some house has second floor and some may not. It is reasonable that total square feet above the ground floor is more related to the house prices compared with the single floor. We didn't add ground floor here since it has been the most powerful predictors according to the correlation matrix.

6. **If a house has a pool** Since most houses do not have a pool thus two features **PoolQC** and **PoolArea** may be overkilled. So we just simplified them to a boolean variable representing whether a house has a pool.

After doing all the above feature wrapper, the RMSE of the simple lasso regression decreased from 0.129 to 0.118 and the random forest decreased from 0.136 to 0.130, both without hyper-parameter tuning in this step. Besides, we have also tried PCA for dimensionality reduction and elimination of collinearity for linear regression. Unexpectedly, it did not give us a better result even reserving the principal components of 95% variance both in lasso regression and random forest and therefore we decided not to apply PCA in the models. Tree-based models do not rely on the collinearity assumption and typically PCA will not give them better results according to ESL[4]. For lasso regression, the L_1 regularization can also tackle collinearity problem by shrinking some of the correlated predictors to 0 during the optimization process and this may be one of the reason why PCA does not work here.

5 Data Mining Techniques

In this part, different regression methods were implemented with the features creating in the previous section and their cross validation errors (RMSE) were compared. All the models were built with the R **caret**(Classification And REgression Training) library to make them more comparable. Caret is a set of functions that attempt to streamline the process for creating predictive models[5] and it has became the most common choice for performing supervised learning in R just like scikit-learn in python. Hyperparameters of each model were tuned by adding the *tuneGrid* arguments when calling *train* function in caret. In order to maintain comparability and convenience of implementing later models ensemble, all models used the same 10 folds cross validation set during tuning considering the sample size is not large compared with the size of feature space.

5.1 Elastic Net Regression

In the EDA section, we have seen that the data has multicollinearity problem and the normal linear regression model is sensitive to it. While DAVID GOLD[6] shows that tolerant methods can be used to reduce the sensitivity of regression parameters to multicollinearity and this is accomplished through penalized regression. Hui Zou[7] proved that elastic net dominates the lasso under collinearity in his paper. So what is Elastic Net Regression? Elastic Net Regression can basically be seen as the combination of lasso regression and ridge regression to deal with sparsity and consistency of the model at the same time by adding a new hyperparameter alpha to balance the fraction of L_1 regularization and L_2 regularization. Therefore, Elastic Net Regression is a good choice to fit the data considering the linear correlation between predictors and response variable and the multicollinearity among the predictors. It is worth noting that this model can only accept the numeric input and the formula in train function automatically changes all factor variables into dummy variables using one-hot encoding.

Hyperparameter tuning result of Elastic Net Regression is shown below:

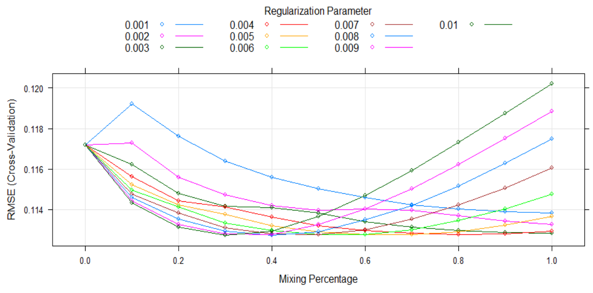


Figure 9: RMSE of Elastic Net Regression

Finally, the Elastic Net Regression with optimal hyperparameter (alpha = 0.4, lambda = 0.008) returns the best RMSE, which is 0.1127.

5.2 Random Forest

Tree-based models are nonparametric models and they do not have too many assumptions of the data. In the lecture, we have knew that a single tree can not give us good enough prediction while random forest uses bagging idea to reduce the out of bag error by introducing majority voting mechanism and subset of features. The most important hyperparamter of random forest is the number of randomly selected predictors for each tree, this typically is set to the square root of the total number of predictors while practically this may not always give us the best result dealing with different data sets. In addition, instead of using the normal 'Randomforest' method in R we used 'ranger' method to train the random forest model. This can give us almost the same result with much faster training process. Tuning result of random forest is shown below:

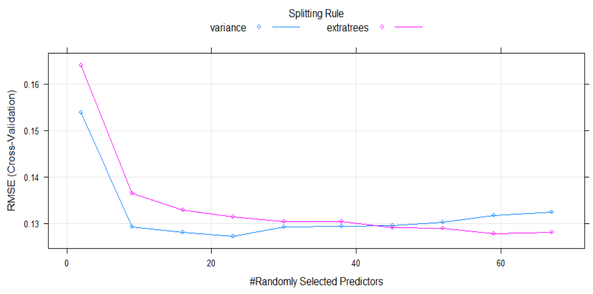


Figure 10: RMSE of random forest

It shows that random forest using the variance splitting rules with 24 randomly selected predictors gives us the lowest RMSE, which is 0.1272.

5.3 SVM with Linear Kernel

In the lecture, we have learnt that how SVM can tackle the classification problem by constructing a hyperplane which maximizes the margin. While with minor change of the optimization formula, SVM can also be used to tackle the regression problem[4], also known as SVR (Support Vector Regression). Basically, SVR tries to use regression residuals to construct a zone making the majority of the data fall within the zone and minimizing the distances. A major benefit of using SVR is that it is a non-parametric technique, the output model from SVR does not depend on distributions of the underlying dependent and independent variables. That is to say, data always occupies only a low-dimensional subspace of the feature space, and regularization results in the learner dealing only with that subspace[4] and that's why curse of dimensionality is not always a problem for SVM, especially when the data space is linearly separable. It is also important to scale and center all the data to make all the features comparable in the high-dimensional space for SVM model. And SVM need to well tune its regularization term (C) to make it more robust for overfitting and the tuning result is shown below:

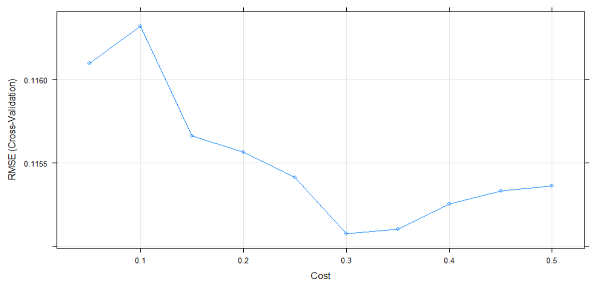


Figure 11: RMSE of SVM regression

It shows that SVR with cost parameter 0.3 gives us the best result: 0.1150 of RMSE. Actually, we have also tried SVM with raidal kernel and it did even worse than the linear one, one reason can be the sample size is not large enough compared with the feature space size (in this case linear kernel can often work well enough according to Andrew Ng machine learning course notes) and data may already been linearly separable considering the linear regression has already gives us a fair enough results.

5.4 XGBoost

In MA429 lecture, we've learnt that boosting can improve the performance of weak classifier by building models sequentially to minimize the errors from previous models while taking influence of high-performing models into account. The most famous boosting algorithm should be AdaBoost. However, there are 3 main defects of this algorithm:

1. Can only be applied into classification problem.
2. Hard to trained and not easy to converge especially in high dimension.
3. Vulnerable to overfitting.

Leo Breiman observed that boosting can be interpreted as an optimization algorithm on a suitable cost function[8]. While Gradient Boosting Machines, which apply gradient descent algorithm to minimize the errors were subsequently developed by the co-author of ESL Jerome Friedman[4].

Tianqi Chen and Carlos Guestrin publised a paper named *XGBoost: A Scalable Tree Boosting System* on SIGKDD 2016 conference and the XGBoost algorithm became one the most popular and powerful model in the industry and data mining competitions. Compared with traditional gradient boosting algorithm, it has the following main advantages:

1. 10 times faster to train using parallel processing.
2. Tree pruning using depth-first approach.
3. L2 norm regularization for avoiding overfitting.
4. Flexible cost function and reserving more information of loss function
5. By retrieving second order Tailor expansion of it.

XGBoost has quite a lot hyperparameters that need to be tuned, the most important maximum depth of tree in each iteration (max_depth), subsample rate (subsample) and learning rate (eta)[9]. The optimal max_depth, subsample rate and eta is 4,0.05,0.5 respectively, giving us the best RMSE 0.1141.

5.5 Stacking

Comparing the results of previous models, we can see that Elastic Net Regression has lowest RMSE value with 0.1127. But we are not very satisfied with this outcome, so a further ensemble method, stacking, can be used to decrease the variance and error and also to improve the prediction power.

Stacking is an ensemble method apart from bagging and boosting. It often considers heterogeneous strong learners whereas bagging and boosting consider mainly homogeneous weak learners. Besides,stacking learns to combine the base models using a meta-model whereas bagging and boosting combine weak learners following deterministic algorithms. The idea of stacking is to learn several different strong learners and combine them by training a meta-model (usually linear regression or logistic regression) and then output predictions based on the multiple predictions returned by these models[10].

In each model, one fold of prediction is taken to combine a new feature, and this new feature become the input of the final meta-model.The meta-model learns from these predictions and give its final outcome.

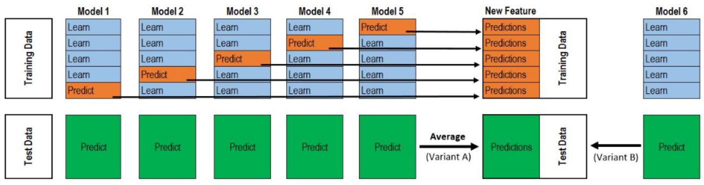
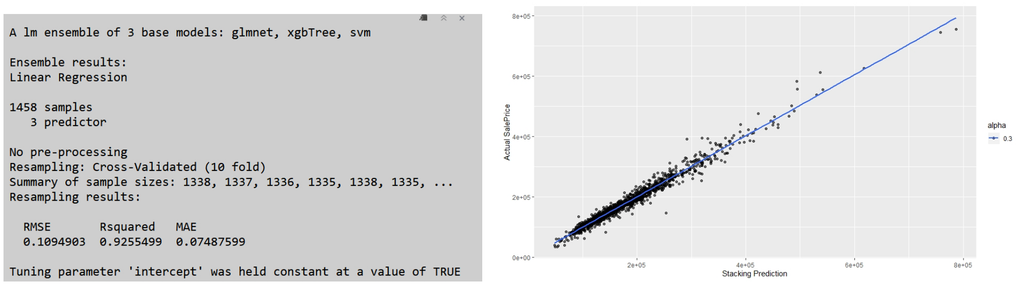


Figure 12: Principle of Stacking

In this project, we used tuned Elastic-Net regression, SVM and XGBoost model as the input model for stacking (we do not input Random Forest since its performance is not as good as the other three), then used linear regression model as the meta-model (also called blender).We can see the RMSE of stacking model(0.1095) is lower than any single base model. Also, figure 13(b) shows the relation between the stacking predicted **SalePrice** and actual **SalePrice**.

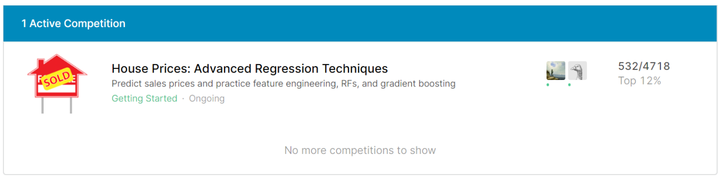


```
A lm ensemble of 3 base models: glmnet, xgbTree, svm

Ensemble results:
Linear Regression

1458 samples
   3 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1338, 1337, 1336, 1335, 1338, 1335, ...
Resampling results:

  RMSE       Rsquared   MAE
  0.1094903  0.9255499  0.07487599

Tuning parameter 'intercept' was held constant at a value of TRUE
```

(a) RMSE of stacking model (b) Stacking Prediction VS Actual SalePrice

Figure 13: Outcome of stacking model

Finally, we uploaded the prediction result of testing set on Kaggle and the testing error is 0.1173 which is 7.1% higher than the validation error. And this result ranked us at top 12% among over 4700 teams!



1 Active Competition

House Prices: Advanced Regression Techniques
Predict sales prices and practice feature engineering, RFs, and gradient boosting
Getting Started · Ongoing

532/4718
Top 12%

No more competitions to show

Figure 14: Final Kaggle ranking

5.6 Different Models Comparison

Since stacking model combines the different models and predict the final outcome with a meta-model taking the predictions of single models as inputs, we can consider stacking model as a further ensemble method to reduce prediction variance. Though stacking gave us the best result, we can not get the importance of features of stacking model due to different model input and thus we only compare the results of single model in this part. The distributions of error of the other 4 models during cross-validation are shown in Figure 15. As we can see, elastic net regression is the most successful one since most of its cross-validation error concentrated in a smaller scope while random forest is the worse and its cross-validation errors have high variance and fall in a larger scope. This once again proves that when the most useful features have a smooth, nearly linear dependence on the covariates or the dependence is multivariate linear and smooth, we should not expect that random forest can perform much better than linear regression with regularization terms.

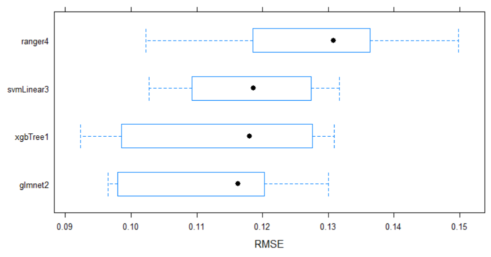
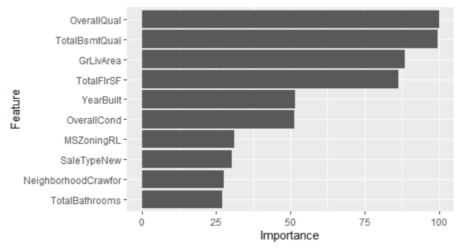


Figure 15: comparison of four models

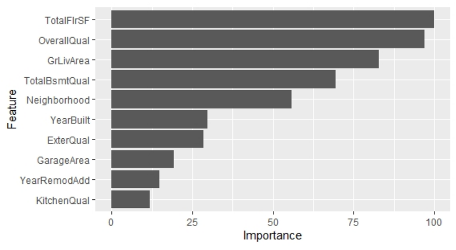
Model	Elastic-Net Regression	Random Forest	XGBoost	SVM
Cross-Validation Time	19.47	60.08	4066.87	360.82
Final Model Running Time	0.11	0.36	6.25	1.88

Table 2: Different Models Running Time (s)

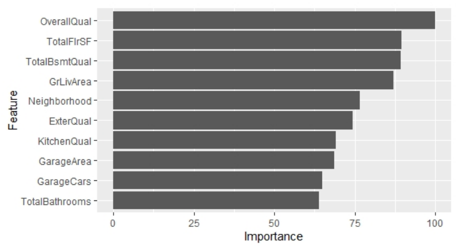
So what are the most important features determining the price of a house? Variables importance given by the four different models are show in Figure13. If we carefully check them we will find that they agree on some most important features while there are quite a few inconsistencies among them. All the models agree that **overall quality, total basement quality, ground floor size and total upper floor size** are the most important 4 features. But some of them attach more significance to neighbourhood while others consider age of houses a more determined factor. This is because different models use different encoding methods and feature importance calculation. For example, random forest do not require one-hot encoding and therefore can store the information of the originally categorical variables while elastic-net can only accept numeric input. Also, according to caret documentation, elastic-net regression uses the absolute value of the coefficients to calculate importance while XGBoost gets the importance through the boosting steps. We can also find that the new features that we created in the feature engineering section ranked high by different models meaning that they are 'better' features compared with the original data set.



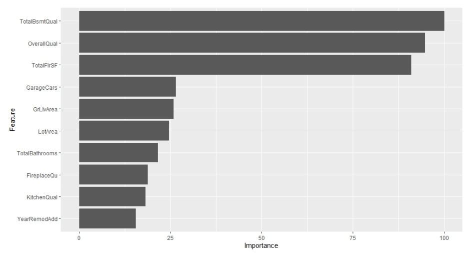
(a) Variables Importance of Elastic-Net regression



(b) Variables Importance of Random Forest



(c) Variables Importance of SVM



(d) Variables Importance of XGBoost

Figure 16: Variables Importance of different models

6 Summary of Results

In this project, we fully analysed the house prices data set and mainly focused on feature engineering and setting up sensible regression models to minimize out-of-sample error. The main findings of the whole data mining process are summarized as below:

1. As the development of machine learning, more and more advanced models have been developed to tackle the complex data mining problems. While in some cases, a simple linear model is good enough to give us a fair result. In this project, we even found that a linear regression with regularization term beats all the other single models and has higher interpretability and training efficiency. Though the real world is complex, linear correlation is still important and it illustrates that sometimes 'less is more'.

2. In terms of private held real estate pricing, general features are more significant than any specific features, i.e., overall quality and total size are more decisive than whether the house has heating or fireplaces. That's why suitable features combination does improve the performance of the models. While specifically, basement, garage and kitchen are the most important facilities determining house prices. Location should always be important, while this pattern can only be seen in some specific neighbourhoods (e.g. fancy neighborhoods), for the majority of the common neighbourhoods, there's no obvious differences.

As Francois Chollet said "Developing good models requires iterating many times on your initial ideas." And we had a lot fun in making what have learnt into practice and utilizing different data mining techniques to prove our initial thoughts. While this is not a thorough project, there are still many works worth trying such as more sensible features engineering with industry knowledge, more elaborate data preprocessing, considering more economics factors from external database etc. It is very rewarding and worthwhile to explore on the domain of data mining.

References

[1] Gilles Louppe, Louis Wehenkel, Antonio Sutera, and Pierre Geurts. Understanding variable importances in forests of randomized trees. In *Advances in neural information processing systems*, pages 431–439, 2013.

[2] Rakesh Ravi. One-hot encoding is making your tree-based ensembles worse, here's why? Website, Jan 11,2019.

[3] HJvanVeen. Feature engineering. Website, Feb 20, 2017. `https://www.slideshare.net/HJvanVeen/feature-engineering-72376750`.

[4] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction.* Springer Science & Business Media, 2009.

[5] Max Kuhn. Building predictive models in r using the caret package. Website, March 27, 2013. `https://topepo.github.io/caret/index.html`.

[6] David Gold. Dealing with multicollinearity: A brief overview and introduction to tolerant methods. Website, Feb 22, 2017. `https://waterprogramming.wordpress.com/2017/02/22/dealing-with-multicollinearity-a-brief-overview-and-introduction-to-tolerant-methods/`.

[7] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2):301–320, 2005.

[8] Leo Breiman. Arcing the edge. Technical report, Technical Report 486, Statistics Department, University of California at . . . , 1997.

[9] Tianqi Chen. Xgboost documentation. Website, Feb 22, 2020. `https://xgboost.readthedocs.io/en/latest/index.html`.

[10] Joseph Rocca. Ensemble methods:bagging, boosting and stacking. Website, April 23, 2019. `https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205/`.