ST443 - GROUP PROJECT

Group 9*

85770

81024

81090

83310

84316

*All the student contributed equally to the project (20%).

Table of Contents

PART 1: REAL WORLD DATA

Section 1. Introduction

1.1. Databases – train.csv and test.csv

The data is available on Kaggle as part of the competition "House Prices: Advanced Regression Techniques". The data is in two files train.csv and test.csv. The training dataset consist 1460 observation with 79 explanatory variables with the response "SalePrice". The testing data contains 1459 observations with corresponding 79 variables and does not contain the dependent variable "SalePrice". Out of the 79 variables 46 are categorical and 33 are continuous.

1.2. Kaggle competition objective

The objective of the competition is to predict the "SalePrice" in the testing data as accurately as possible. To measure the accuracy of the model one needs submit the predictions to Kaggle's website and then the submissions are evaluated on Root-Mean-Squared-Error (RMSE) between the logarithm of the predicted value and the logarithm of the observed sales price.

1.3. Links

Kaggle's competition website: https://www.kaggle.com/c/...

Our GitHub repository: https://github.com/heymic/ST443-Project-group9

Section 2. Data Cleaning and Transformations

2.1. Missing data and imputation

The datasets contain some missing values (NAs). After a closer look at the structure of the data, one finds that some of the missing values could represent either a separate level of the categorical variables or missing values. For instance, NAs in the "*PoolQuality*" variable could either mean that is no actual pool in the house or that the quality of the pool was not recorded.

To solve this step, we cautiously looked at both the data description document attached to the files and other corresponding variable (f.e for *PoolQuality -> PoolArea).* After deciding whether the data point is indeed missing, or it represents a level of the variable, we have decided to group the variables together in to order to assign the most accurate value using the mean, mode or median.

For instance, we have grouped the mean of the *PoolArea* together with *PoolQuality*. Through data inspection we notice that some observations have a positive *PoolArea* while having NAs in the *PoolQuality*, which clearly indicates that these NAs in *PoolQuality* are missing values. In this case we have assign the corresponding value to *PoolQuality* calculated by grouping, by the mean of *PoolArea*.

This approach has been applied to several other variables within the data set, see the attached code for the data cleaning we have done to the original dataset.

2.2. Transformation of the response variable *SalePrice*

Linear models are known to perform better when the response variable is normally distributed. Plotting the distribution of *SalePrice* we noticed that it is skewed to the right. Correcting for the skewness, we applied a logarithmic transformation on the response variable to obtain a "bell-shaped" distribution.

Section 3. Regression

3.1. Model Selection

We have trained many models on different techniques of transformations and imputations of the data and compared them by the 10-fold Cross-Validation errors. We have achieved the best outcome by following the procedures described in Section 2 above.[1]

3.2. Model Performance Comparison:

The two tables below summarise the RMSE of our models on predicting the sales price of the testing data set. As mentioned in Section 1.1, the test set does not have the *SalesPrice* variable, the below RMSE is given to us after submitting our predictions to Kaggle online.

No	Model	RMSE, Kaggle Score
1	Gradient Boosting Model (gbm) tuned	0.12765
2	Lasso with 10fold CV λmin	0.13110
3	Xgboost tuned	0.13380
4	Random Forest with m = p/3 = 26	0.14685
5	Random Forest with m = $\sqrt{p} = 9$	0.14723
6	Bagging Tree	0.15255
7	Linear Regression with all parameters	0.16705
8	Lasso with 10fold CV λ1se	0.16991
9	Regression Tree	0.22079

Combined Models, average of predictions	RMSE, Kaggle Score
No.1 + No.2	0.12438
No.1 + No.2 + No.3	0.12492

3.3. Best Regression Model

The best model obtained (highlighted in green) is the average of the predictions of the gradient of the Gradient Boosting Model (No.1) with the Lasso λmin (No.2) model.

Based on separate models the Gradient Boosting Model and tuned on R package "gbm" performs the best. Interestingly, the Lasso model with lambda chosen as minimum in the Cross-Validation procedure outperforms the random forest methods. These could lead to the hypothesis that house prices are linearly dependent to factors.

3.4. Other approaches

The dataset consists many categorical factors that are ordinal, meaning that one can order them from highest to lowest value. We have tried to train the model by grouping together these ordinal factors based on the Sales Price corresponding mean and treat them as numeric. However, this approach resulted in higher Cross-Validation error rate and higher test error.

[1] Graphs from the regression and classification models are found in the R codes, attached in the appendix at the end of this report.

Section 4. Classification

4.1. Dependent variable

The aim of the classification approaches applied on the dataset is to predict whether or not the house will be sold above the median price.

A binary variable was created based on the log transformed *SalePrice* variable in the dataset. Mean was chosen as the statistics to divide the sale price, as the skewness of *SalePrice* has already been corrected by the logarithmic transformation. Hence, any observation with log(*SalePrice*) greater than the mean would be classified as "High" and "Low" otherwise.

4.2. Data splitting

Testing for error rate would be different from section 3 above, as Kaggle does not accept classification predictions for the testing dataset and the testing data set does not have *SalePrice* for us to manually create our own confusion matrix to identify the error rate.

Hence, the original training data will be randomly divided, with 50% used as testing data and the remaining 50% to be used as training data.

4.3. Model Selection and Results

Using similar methodologies as Section 3, we have trained many models on different techniques and the table below summarises our findings.

No	Model	Error Rate Misclassification
1	Random Forest with m = p/3 = 26	0.0808219
2	Gradient Boosting Model (gbm)	0.0821918
3	Ridge with 10-fold CV λ1se	0.0890411
4	Support Vector Classifier	0.0890411
5	Ridge with 10-fold CV λmin	0.0917808
6	Lasso with 10-fold CV λmin	0.0931507
7	Bagging Tree	0.0986301
8	Lasso with 10-fold CV λ1se	0.1000000
9	Classification Tree (Pruned)	0.1164384
10	Classification Tree	0.1219178

4.4. Best Classification Model

The best classification model here is the random forest model, with the lowest error rate of 8.08%. It is followed closely by the gradient boosting model. The models used for classification generally have lower classification error rate when compared to the regression models in Section 3.

Section 5. Conclusion

Despite not having a basis of comparison for our classification models as Kaggle does not accept submissions for classification results. However, with our best regression model, we have scored in the top 30% of the competition. To improve this result, one could try to group, impute and transform the data in different ways seeking the minimal testing score. Looking at the website scoreboard the best model seems to be averages out of many different models. Improving the model in this way is trial and error, because the some of the models will lead to lower CV error rate and some to larger.

Part 2: Estimation of graphical model using lasso related approaches

Section 1: Mathematical Introduction

1.1. Graphical Models

A graphical model is a probabilistic model which illustrates the conditional dependence structure among p random variables, $X = (X_1, \dots, X_P)^T$. For a network with p nodes, one node is for each variable, along with edges connecting a subset of the nodes. These edges represent the structure of conditional dependence of the p variables.

Specifically, for $1 \leq j, l \leq p$, $c_{jl} = \text{Cov}(X_j, X_l | X_k, 1 \leq k \leq p, k \neq j, l)$ represents the covariance of X_j and X_l conditional on the remaining variables. Then nodes j and l are connected by an edge if and only if $c_{jl} \neq 0$. We will mainly focus on two Gaussian graphical models through the lasso regularization in this part, node-wide lasso approach and graphical lasso approach.

Let G = (V, E) denotes an undirected graph with vertex set V = {1, … , p} and edge set

$$E = \{(j,l) : c_{jl} \neq 0, 1 \leq j, l \leq p, j \neq l\}$$

1.2. Node-wise LASSO approach

Node-wise lasso approach is one of the methods used to estimate the conditional dependence structure of Gaussian graphical models, more specifically, for each node $j \in V$, do regression X_j on the remaining variables X_l, $l \in V, l \neq j$, in the form of

$$X_j = \sum_{1 \leq l \leq p, l \neq j} \beta_{jl} X_l + \varepsilon_{jl}$$

To obtain a sparse solution for β_{jl}'s, the lasso approach can be implemented to select which component in $\{\beta_{jl}, l \in V, l \neq j\}$ is non-zero. For a certain tuning parameter λ, let the lasso estimator for β_{jl} to be $\hat{\beta}_{jl}$: If $\hat{\beta}_{jl} \neq 0$, then nodes j and l are estimated to be connected. Following rules, which are called node-wise lasso 1 and node-wise lasso 2, respectively, can be used to estimate E in (1):

$$\hat{E}_1 = \{(j,l) : \beta_{jl} \neq 0 \text{ and } \beta_{lj} \neq 0, 1 \leq j, l \leq p, j \neq l\},$$

$$\hat{E}_2 = \{(j,l) : \beta_{jl} \neq 0 \text{ or } \beta_{lj} \neq 0, 1 \leq j, l \leq p, j \neq l\}.$$

1.3. Graphical lasso approach

Graphical lasso approach is used for depicting the conditional dependence structure of Gaussian graphical models through the use of L1 (lasso) regularization.

Consider observations $(X_1, \dots, X_n)$ from multivariate Gaussian distribution with covariance matrix Σ, one can show that $c_{jl} = 0$ if and only if $\theta_{jl} = 0$, where θ_{jl} is the (j, l)-th entry of the inverse covariance matrix, $\theta = \Sigma^{-1}$, also called precision matrix.

The basic assumptions for the model is that the observations are from a multivariate Gaussian distribution with covariance matrix Σ. If the $\theta_{jl} = 0$, then variables i and j are conditionally independent, given all the other variables. Thus, imposing L1 penalty for the estimation of Σ^{-1} will increase its sparsity.

The graphical lasso approach is able to estimate the inverse covariance matrix θ under a multivariate normal model by maximizing the $\ell 1$-penalized log-likelihood ($\log \det\theta - \mathrm{trace}(S\theta) + \lambda \sum_{j\neq l} \theta_{jl}$).

The edge set can be represented by E={(j,l) : $\theta_{jl} \neq 0$, $1 \leq j, \ l \leq p, j \neq l$ }. With a certain choice of tuning parameter, the estimated edge set can be shown as $\hat{E}_3$={(j,l) : $\hat{\theta}_{jl} \neq 0$, $1 \leq j, \ l \leq p, j \neq l$ }.

Section 2: Results and conclusions

2.1. Simulation settings

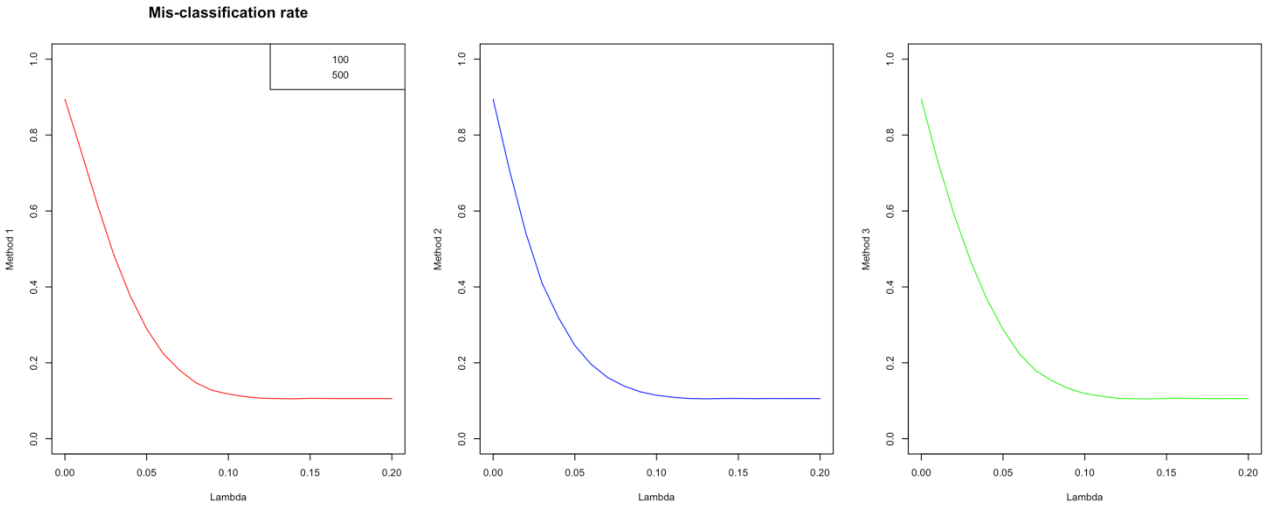
To compare the three different estimation methods, we have estimated the edges with different complexity and we have plotted the miss-classification rate and the ROC curves. The following combinations of p and n have been used:

1. p=50 and n=500;
2. p=100 and n=200;
3. p=100 and n=500;
4. p=100 and n=1000;
5. p=150 and n=500.

These combinations allowed us to compare different results of the same methods when p is fixed (100 edges) and n varies (200,500,1000) and when p varies (50, 100, 150) and n is fixed (500). We have considered a penalization factor lambda which varies from 0.0 to 0.2.

2.2. Miss-classification error

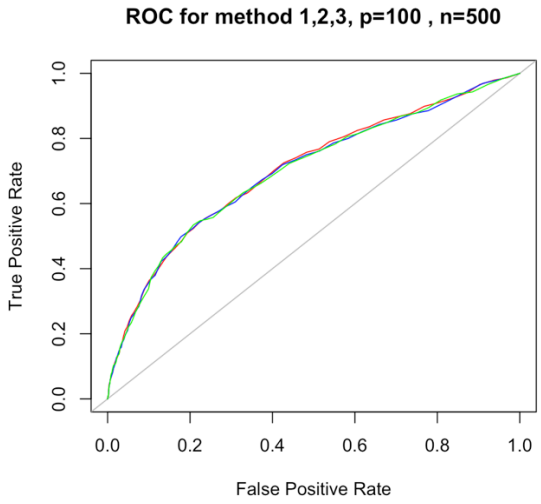
The first observation that we made is about the shape of the graph of the miss-classification error vs lambda. In the following graph lambda varies from 0 (no penalization) to 0.2 (maximum penalization considered), p is equal to 100 and n is equal to 500.



We can see that the miss-classification error start from 0.9 when lambda is equal to 0 and tends towards 0.1 when lambda is big. This can be explained easily recalling the setting of the problem. To generate the set of the vertices, we have use a Bernoulli random variable: a couple of edges is connected with probability 0.1 and it is not connected with probability 0.9. When lambda is equal to zero the solution of the regression it is not sparse and each parameter is different from zero. Thus, all vertices are estimated to be existing and we make an error for all the vertices which are not existing in the real set (roughly 90% of the total possible combinations of edges). Conversely, if lambda is big, the penalization term is too strong and all vertices are estimated to not exist. In this case we make an error each time that the vertices exist (roughly 10% of the times). It is important to underline that the miss-classification rate is not a good index too choose the model. In fact, in our example, it suggests picking a model which predicts that all vertices are not existing.

2.3. ROC curves: comparison between different models

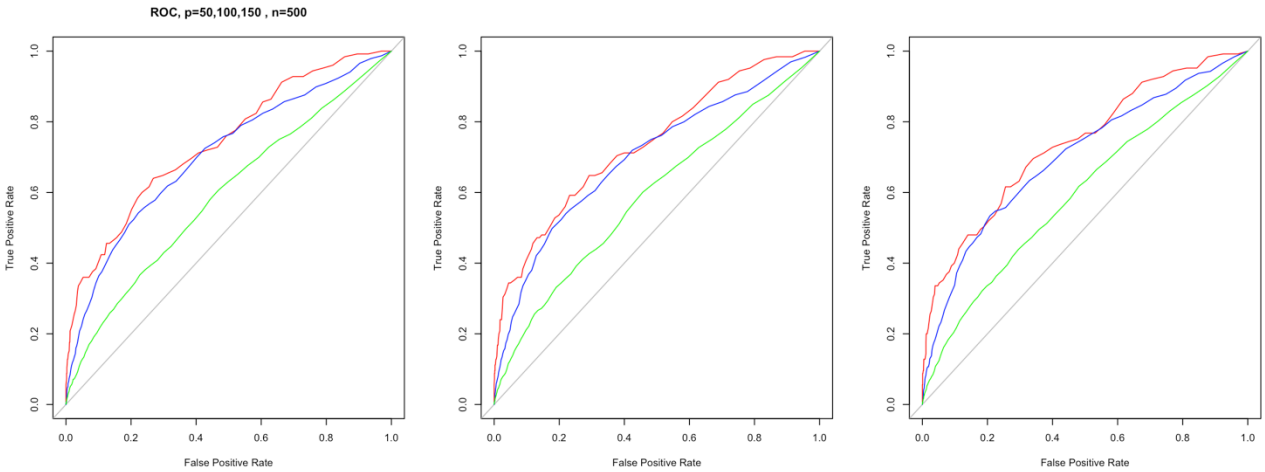
To compare the different estimation methods, we plotted the ROC curves for each method for different level of p and n. The following graph shows the ROC curve for each method for p=100 and n=500.



We can see that the outcome of the different calibration methods is very similar. The area under the ROC curve (AUROC) for method 1, 2 and 3 is respectively 0.704, 0.701 and 0.700. Similar results are obtained with different level of p and n. Hence, the difference between the three estimation methods is negligible.

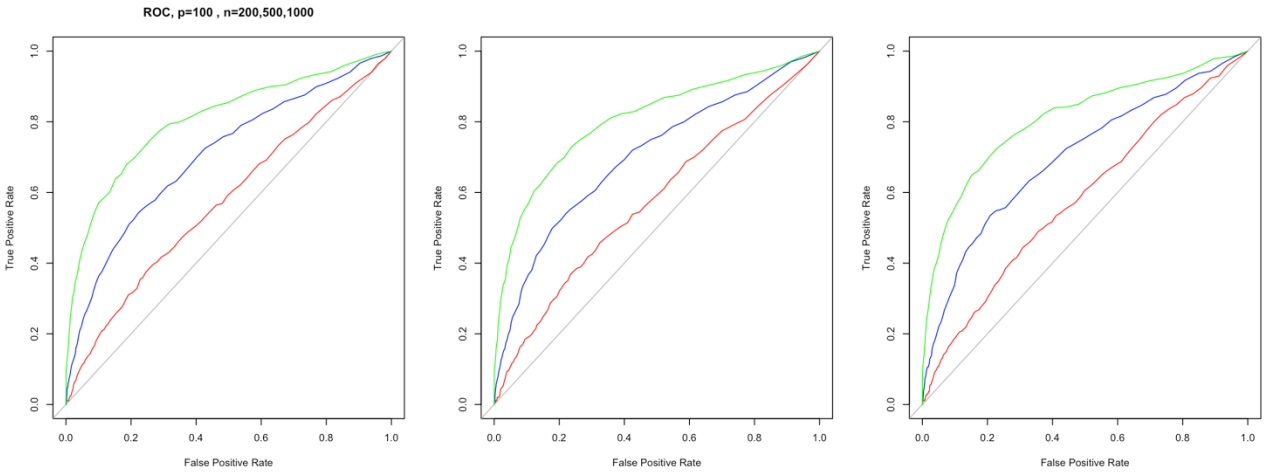
2.4. ROC curves: comparison between different p and n

In this paragraph, we analyze the how the quality of the estimation changes at different levels of p and n. Firstly we will analyze what happen when p changes. The following graphs show the ROC curves for methods 1,2,3, n=500 and p=50 (red line), p=100 (blue line) and p=150 (green line).



We can see that when p increases the AUROC decreases. This result is intuitive. Indeed, when the information available (n) is fixed, if the complexity of the problem (p) increases the quality of the estimation decreases.

When p is fixed, and n varies we expect that the AUROC will increase as well. Ceteris paribus, if the information available increase, the quality of the estimation should increase. The following graphs show the ROC curves for methods 1,2,3, p=100 and n=200 (red line), p=500 (blue line) and p=1000 (green line).



As expected, the ROC curves with n=1000 dominates the ROC curves for n=500 and n=200.

2.5. ROC curves: numerical results

The value of the AUROC in the five cases considered are reported in the following table.

			AUROC		
p	n	n/p	$\hat{E}_1$	$\hat{E}_2$	$\hat{E}_3$
50	500	10	0.7400509	0.7372255	0.7371309
100	200	2	0.5745979	0.5754654	0.5823614
100	500	5	0.7040512	0.7007001	0.6999867
100	1000	10	0.8066268	0.8045442	0.8079276
150	500	3.33	0.5905767	0.5911834	0.5953109

From Table 1, increasing p for same sample size n, AUROC decreases for all 3 methods when p increases from 50 to 100 and from 100 to 150. For same value of p with increasing sample size n, it is shown that AUROC is likely to increase based on our results for n = 200, 500 and 1000. Among the 6 combinations, the setting with $\mathrm{p} = 100, \mathrm{n} = 1000$ shows the highest value of AUROC (i.e. approx. 0.80), while the lowest is from with $\mathrm{p} = 100, \mathrm{n} = 200$.

Finally, we run the code 50 times with p=100 and n=500 to evaluate the mean and the variance of the AUROC estimation. The results are reported in table 2.

AUROC	Mean	Standard deviation	Variance
Method 1	0.6512545	0.02899642	0.0008407924
Method 2	0.6500842	0.02859026	0.0008174031
Method 3	0.6533424	0.02875512	0.0008268572

2.6 Conclusions

In this exercise, we compared the three estimations methods and they gave approximately the same results in term of AUROC and miss-classification rate. There is no evidence that one method is significantly better than the others. However, running the algorithm 50 times the graphic lasso regression perform slightly better than the node-wise lasso regression.

We evaluated the performance of the three methods with various p and n. We concluded that if p increases the AUROC decreases, while if n increases the AUROC increases as well.

Bibliography

Chen, T. He, T. Benesty, M. Khotilovich, V. and Tang, Y. (2017). *xgboost: Extreme Gradient Boosting. R package version 0.6-4*. https://CRAN.R-project.org/package=xgboost

Friedman J., Hastie T., Tibshirani R. (2010). *Regularization Paths for Generalized Linear Models via Coordinate Descent*. Journal of Statistical Software, 33(1), 1-22. URL http://www.jstatsoft.org/v33/i01/.

Friedman, J. Hastie, T. Tibshirani, R. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer-Verlang, ch. 17.

Friedman J, Hastie T. and Tibshirani R. (2014). *glasso: Graphical lasso- estimation of Gaussian graphical models*. R package version 1.8. https://CRAN.R-project.org/package=glasso

Liaw A. and Wiener M. (2002). *Classification and Regression by randomForest.* R News 2(3), 18--22.

Kuhn M. Contributions from Jed Wing, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, Brenton Kenkel, the R Core Team, Michael Benesty, Reynald Lescarbeau, Andrew Ziem, Luca Scrucca, Yuan Tang, Can Candan and Tyler Hunt. (2017). *caret: Classification and Regression Training*. R package version 6.0-77. https://CRAN.R-project.org/package=caret

Ridgeway G. with contributions from others (2017). *gbm: Generalized Boosted Regression Models. R package version 2.1.3*. https://CRAN.R-project.org/package=gbm

Ripley, B. (2016). *tree: Classification and Regression Trees*. R package version 1.0-37.

Venables, W. N. & Ripley, B. D. (2002) *Modern Applied Statistics with S. Fourth Edition*. Springer, New York. ISBN 0-387-95457-0

Wickham, H. Francois, F. Henry, L. Müller, K. (2017). *dplyr: A Grammar of Data Manipulation*. R package version 0.7.4.

Zeileis, A. and Grothendieck G. (2005). *zoo: S3 Infrastructure for Regular and Irregular Time Series. Journal of Statistical Software*, 14(6), 1-27. doi:10.18637/jss.v014.i06

Greg Ridgeway with contributions from others (2017). gbm: Generalized Boosted Regression Models. R package version 2.1.3. https://CRAN.R-project.org/package=gbm