Project

Chaoyi Tsai

5/18/2022

Introduction

The data set contains information on clickstream from an online store offering clothing for pregnant women. Data are from five months of 2008 and include, among others, product category, location of the photo on the page, country of origin of the IP address, and product price in US dollars. There are 132380 observations and 14 variables with no missing values in the original data set.

My goal is to compare the prediction performances between the Negative Binomial GLM model and common machine learning techniques, including lasso regression, ridge regression, general additive model, partial least square regression, XGBoost, KNN for Regression, stochastic gradient boosting, bagging, elasticnet, ensemble, boosting, and neural network. All codes are written in R using the packages `lme4` and `Caret`. The evaluation criterion for choosing the best machine learning model is RMSE (root-mean-square error).

Table 1: Data Information

Column	Description
year	Year
month	From April (4) to August (8)
day	Day number of the month
order	Sequence of clicks during one session
country	Variable indicating the country of origin of the IP address
session.ID	Variable indicating session id (short record)
page.1..main.category.	Concerns the main product category (1=trousers, 2=skirts, 3=blouses, 4=sale)
page.2..clothing.model.	Information about the code for each product
colour	Colour of product
location	Photo location on the page, the screen has been divided into six parts (1=top left, 2=top in the middle, 3=top right, 4=bottom left, 5=bottom in the middle,6=bottom right)
model.photography	Variable with two categories (en face, profile)
price	Price in US dollars
price.2	Variable informing whether the price of a particular product is higher than the average price for the entire product category
page	Page number within the e-store website (from 1 to 5)

Exploratory Data Analysis

From figure1, we can see that, though most of the categories in each feature have primarily even percentages, most orders came from a single country (Poland). Interestingly, most items that were ordered were placed on the first page, perhaps because customers are reluctant to browse the following pages. Figure2 shows that while some countries have more continuous orders over these months, such as Poland, Lithuania, and the Czech Republic, others only make orders in a relatively short period.

Since my goal is to compare the prediction performance between the parametric and nonparametric models,

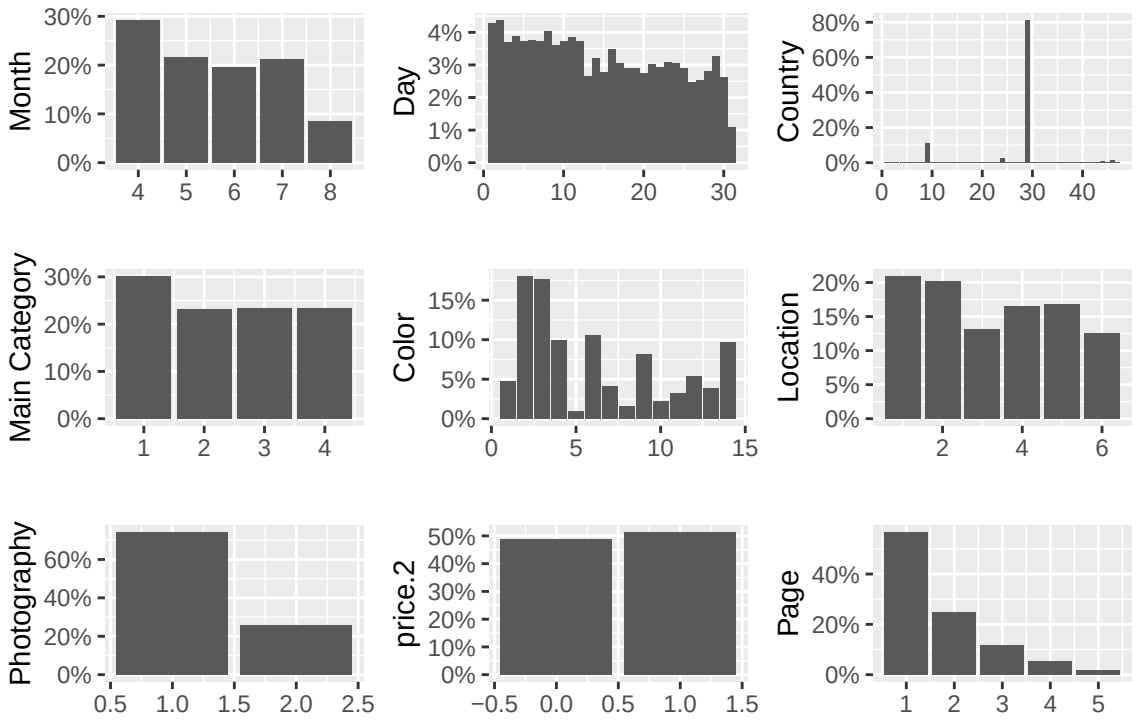

Figure 1: Barplots

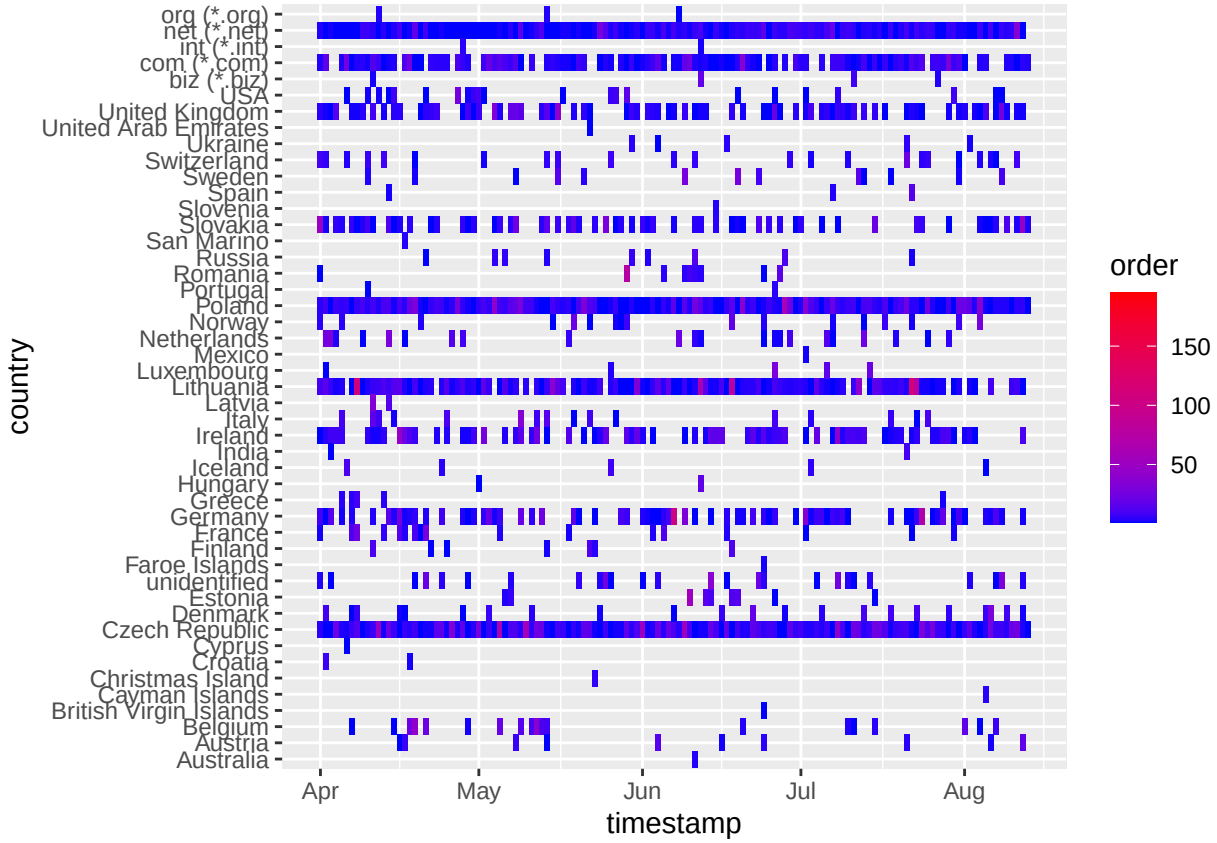

Figure 2: Response Variable across Countries

I would utilize the function `createDataPartition` to create a stratified random sample of the data into training and test sets. Here, 80% of the samples are used for training, and the rest are used for testing.

Generalized Linear Mixed-Effects Model

While the outcome variable order is count data, meaning that it should be Poisson distribution, due to the existence of a thick right tail, I use the function `fitdist` to confirm that the outcome variable is most similar to the negative binomial distribution.

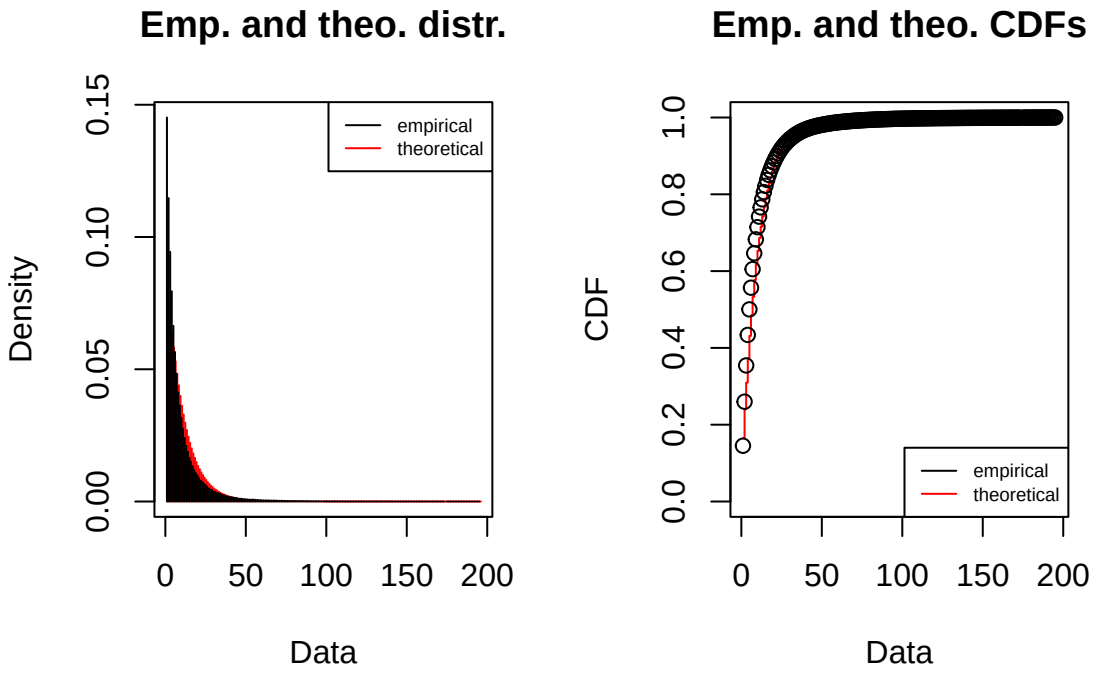

Figure 3: Distribution of the Response Variable

To fit a generalized linear mixed-effects model (GLMM) for the negative binomial family, I use the function `glmer.nb` in the package `lme4`. The covariates include all the variables listed in the table1 except the variable year and the session ID because the former variable is constant, and the latter is not informative. The only random effect is country, and the rest are fixed effects as I expect there to be inter-nation differences I'm not interested in but want to account for. Below is the model I fit:

$$g(\mu_i) = \beta_0 + \sum_{j=1}^{11} \beta_j X_{ij} \; where \; g(\mu_i) = log(\frac{\mu_i}{k(1+\frac{\mu_i}{k})})$$

$$\mu_i = E(Y_i|X_{1i}, ..., X_{11i}) \; and \; Y_{ij}|\mu_{ij} \sim NB(k, p_i)$$

The default optimization method (corresponding to nAGQ=1L) puts all of the coefficients into the general nonlinear optimization call. But owing to the large number of fixed-effects, with nAGQ=0L, these coefficients are optimized within the much faster penalized iteratively reweighted least squares (PIRLS) algorithm. The default will generally provide a better estimate because the deviance at the estimate is lower, but the difference is usually tiny, and the time difference is enormous. The second option tells the function to fit using the `nloptwrap` optimizer to speed up the running time. Since the objective is to compare the forecast performance between each model, I would not make inference or check if the model assumptions are valid, such as whether the residuals are i.i.d. and there is systematic variation in them.

Data Preprocessing

Categorical Encoding

Label Encoding and One-Hot Encoding are two standard techniques for handling categorical variables. I only transformed the variable `page.2..clothing.model.` into a dummy variable as this feature is not ordinal and there are too many categorical classes. Using the label encoding instead would be computationally laboring due to the high dimensions. On the other hand, I use the label encoding for the rest of the categorical attributes owing to their ordinal characteristic.

Zero- and Near Zero-Variance Predictors

In some situations, the data generating mechanism can create predictors that only have a single unique value (i.e., a "zero-variance predictor"). Many models (excluding tree-based models) may cause the model to crash or the fit to be unstable. Similarly, predictors might have only a handful of unique values with very low frequencies. The concern here is that these predictors may become zero-variance predictors when the data are split into cross-validation/bootstrap sub-samples or that a few samples may have an undue influence on the model. These "near-zero-variance" predictors may need to be identified and eliminated before modeling.

Identifying Correlated Predictors

While some models thrive on correlated predictors (such as PLS), other models may benefit from reducing the level of correlation between the predictors. By discarding highly-correlated attributes, the range of the correlation between each variable is between -0.75 and 0.5.

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.743681 -0.079564 -0.006051 -0.023041  0.028483  0.496576
```

Linear Dependencies

The function `LinearCombos` utilizes the QR decomposition of a matrix to enumerate sets of linear combinations. Each linear combination will incrementally remove columns from the matrix and test to see if the dependencies have been resolved. These dependencies can arise when large numbers of binary chemical fingerprints are used to describe the structure of a molecule.

Cross-Validation

To do 10 folds cross-validation, we specify `method="cv"` and `number = 10` in the function `trainControl`. Moreover, by specifying `allowParallel = TRUE`, we can speed up the running time using the technique of parallel processing.

Nonparametric Models

Ridge regression

This method is in the package `elasticnet`. The only tuning hyperparameter is the weight decay, which is controlled by `lambda` in the function. We can see from figure4 that when λ equals 0.005, the RMSE can reach its minimum.

Lasso regression

This method is in the package `fraction`. The only tuning hyperparameter is the fraction of the full solution, which is controlled by `fraction` in the function. Figure5 below suggests that when the fraction equals 0.996, the RMSE is the smallest, though the difference in RMSE between each tuning hyperparameter specified is minimal.

Generalized Additive Model using Splines

This method is in the package `mgcv`. The tuning hyperparameters include whether to make feature selection and the smoothing parameter estimation method, which is controlled by `select` and `method`, respectively. Specifically, "GCV.Cp" uses GCV for unknown scale parameters and Mallows' Cp for known scale. "GACV.Cp" is equivalent, but using GACV in place of GCV. "REML" for REML estimation, including unknown scale, "P-REML" for REML estimation, but using a Pearson estimate of the scale. "ML" and "P-ML" are similar but use maximum likelihood in place of REML. The best combination of tuning hyperparameters would be the "GACV.Cp" estimation and feature selection judging from figure6. Moreover, whether using the "GACV.Cp" estimation or not, choosing the feature selection method generates the best result.

eXtreme Gradient Boosting

This method is in the package `xgboost`. The tuning hyperparameters include:

- `nrounds` (Number of Boosting Iterations)
- `max_depth` (Maximum Tree Depth)
- `eta` (Shrinkage)
- `colsample_bytree` (Subsample Ratio of Columns)
- `gamma` (Minimum Loss Reduction)
- `min_child_weight` (Minimum Sum of Instance Weight)
- `subsample` (Subsample Percentage)

Considering the time cost of picking the best candidates, I did not tune the last three hyperparameters. My finding indicates that when having 151 boosting iterations, 8 layers of trees, η=0.072 and subsample ratio of columns being 0.999 can generate the most promising outcome.

Partial Least Squares Regression

This method is in the package `pls`. The only tuning hyperparameter is the number of components, which is controlled by `ncomp`. The optimal number of principal components included in the PLS model is 13. This captures 99% of the variation in the predictors and 6% of the variation in the outcome variable.

Stochastic Gradient Boosting

This method is in the package `gbm`. The tuning hyperparameters include:

- `n.trees` (Number of Boosting Iterations)
- `interaction.depth` (Max Tree Depth)
- `shrinkage` (Shrinkage)
- `n.minobsinnode` (Minimum Terminal Node Size)

Setting 180 boosting iterations, 14 layers of trees, shrinkage=0.15, and minimum terminal node size to 12 can offer the slightest error.

KNN for Regression

This method does not require any additional package. The only tuning hyperparameter is the number of neighbors, which is controlled by `k`. The outcome indicates that the number of neighbors should be 129 to be the most appropriate.

Bagging

Initially, I used this method in the package `caret`. However, when running the model, there was an error showing missing values in resampled performance measures resulting from the fact that the tree did not find a good split and used the average of the outcome as the predictor. That's fine, but we cannot calculate R^2 since the variance of the predictions is zero. Accordingly, I use the model in the package `ipred` and `plyr` instead. Though this method has no tuning hyperparameters, a model-specific variable importance metric is available. The result suggests that the most critical factor determining the number of orders is the page on which the item is located. Other vital attributes include whether information about the product is provided and whether it is trousers or on sale, which seems reasonable as consumers might be too lazy to browse items in the later pages.

Boosted Linear Model

This method is in the package `bst`. The tuning hyperparameters include the number of boosting iterations and the level of shrinkage, which is controlled by `mstop` and `nu`, respectively. To achieve the smallest RMSE, we need 233 boosting iterations and the level of shrinkage set to 0.95.

Neural Network

This method is in the package `nnet`. The number of hidden units and the weighted decay is the tuning hyperparameters controlled by `size` and `decay`, respectively. The final values used for the model were size equal to 0.00001 and decay equal to 0.004.

ElasticNet

This method is in the package `elasticnet`.The tuning hyperparameters include the fraction of the full solution and the weight decay, which is controlled by `fraction` and `lambda`, respectively. Our analysis shows that picking λ=0.00007 and fraction=0.98 give the smallest prediction error RMSE.

Ensembles

`CaretEnsemble` uses a `glm` to create a simple linear blend of models, and `caretStack` uses a caret model to combine the outputs from several component caret models. Here, I'm utilizing the 11 above-mentioned nonparametric methods to create the Ensemble.

Comparison

From the following density plots (figure16 ~ figure18), we can see that most models perform evenly concerning the prediction performance. Yet, if we compare the best model in each method, figure19 shows that XGBoost is the best and partial least squares regression has the worst result.

Measure Model Performance on Testing Data

From table2, I found that XGBoost is the best choice. The potential reason for this might be that outliers have minimal impact on it, and it handles large-sized data sets well. This is true since most of the number of orders is small. I also found that people on Kaggle win most of the competitions with XGBoost. On the other hand, stochastic gradient boosting is also a competitive candidate. These are the two methods that surpass other candidates. Table3 presents the result of rounding the prediction to integers. We can see that there is no evident difference in the two tables regarding the prediction performance.

Conclusion

From the time cost standpoint, running the Generalized Additive Model is most time-consuming as it took over 6 hours to tune only 12 combinations of hyperparameters. While XGBoost defeated other methods,

Ensemble might be a good alternative considering the time taken since it only needs less than half the time XGBoost used to run. If producing a slight error is allowed, with only taking one minute, Elasticnet undoubtedly conquered other defendants. The same conclusion can also be drawn if the prediction is rounded to an integer. Generally, most of the models perform well. Still, depending on how many hyperparameters we plan to tune, the time to produce results may vary. While it took about 40 seconds to run the linear mixed model, it still could not compete with other contenders. Once having a more in-depth understanding of the data, the more appropriate cluster structure can derive a better result.

On the other hand, even with the support of parallel computing, I gave up fitting the random forest model since it took forever to generate result, not to mention to tune hyperparameters. The reason may be that there are more than 13 thousands observations in the data set. Additionally, I quit fitting the support vector machine eventually, as the space complexity for kernel matrix is $O(n^2)$. When n is in order of 10^5, getting SVM to work is almost impossible since it would require much memory that my laptop can not tackle.

Given that the outcome variable is an integer, I have read some forums in which people treat it as a categorical variable instead. This might provide us with more powerful insight into selecting the optimal model. To sum up, through taking this course, I feel that people always strive to reduce all sorts of losses by dividing data with different thresholds or using gradient descent in the nonparametric world. By combining various techniques, a new and powerful method might be born.

Tables and Figures

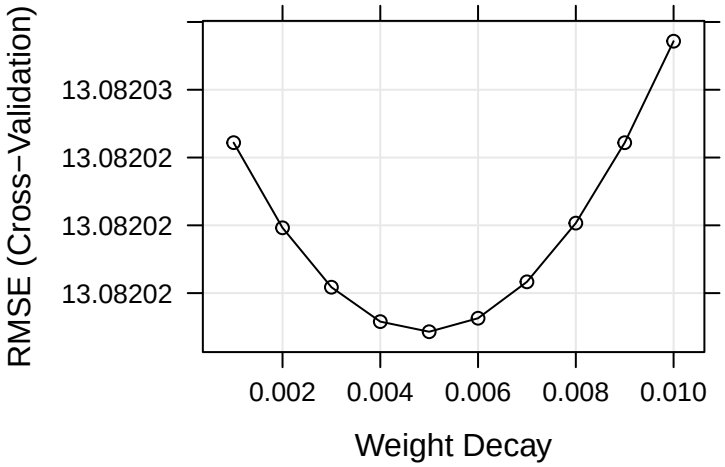

Figure 4: Ridge

Table 2: Comparision between Different Models

	RMSE	Rsquared	MAE	Minutes Taken
GLM	15.29700	-3.07337	7.92572	0.68595
Ridge	12.98263	0.05791	7.72645	0.66444
Lasso	12.98249	0.05792	7.72638	0.10336
GAM	12.97032	0.05967	7.72104	324.78914
XGB	11.98094	0.19833	7.36460	34.45855
PLS	12.98250	0.05792	7.72642	0.08495
SGB	12.14169	0.17638	7.48079	28.78452
KNN	13.00464	0.05514	7.66005	45.42456
Bagging	13.15741	0.03224	7.87362	0.93278
Boosting	12.98374	0.05769	7.73454	8.97356
Neural Network	16.00897	NA	8.79812	1.27750

	RMSE	Rsquared	MAE	Minutes Taken
Elasticnet	12.98256	0.05790	7.72644	0.92050
Ensemble	12.51500	0.12442	7.60821	13.53153

Table 3: Comparison between Different Models

	RMSE	Rsquared	MAE
GLM	15.30961	-2.99901	7.93355
Ridge	12.98883	0.05708	7.72545
Lasso	12.98877	0.05708	7.72617
GAM	12.97386	0.05919	7.71735
XGB	11.98568	0.19762	7.36263
PLS	12.98845	0.05712	7.72572
SGB	12.14197	0.17633	7.47571
KNN	13.00640	0.05493	7.65791
Bagging	13.16244	0.03224	8.00719
Boosting	12.98890	0.05698	7.73122
Neural Network	16.00897	NA	8.79812
Elasticnet	12.98831	0.05713	7.72539
Ensemble	12.51897	0.12388	7.60238

References

1. The Caret Package
2. A Brief Introduction to caretEnsemble

Data Source

This data set comes from UC Irvine Machine Learning Repository.

£apczyński M., Bia³ow¹s S. (2013) Discovering Patterns of Users' Behaviour in an E-shop - Comparison of Consumer Buying Behaviours in Poland and Other European Countries, "Studia Ekonomiczne", nr 151, "La société de l'information : perspective européenne et globale : les usages et les risques d'Internet pour les citoyens et les consommateurs", p. 144-153.

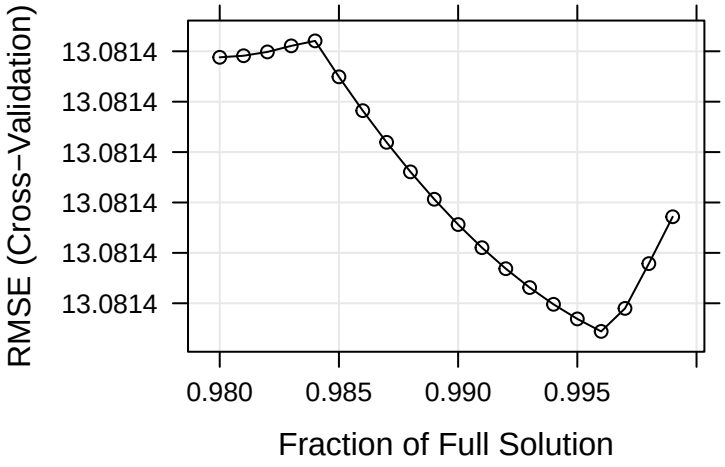

Figure 5: Lasso

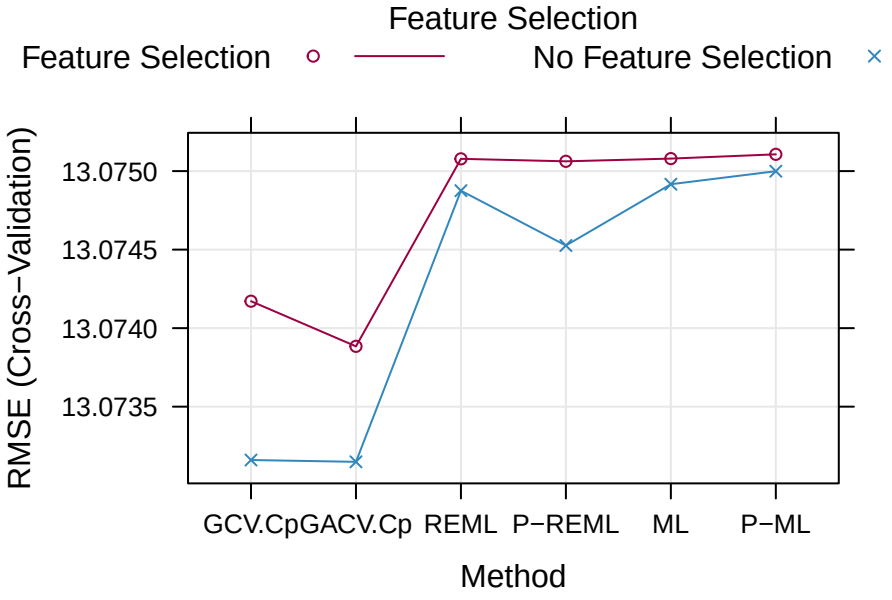

Figure 6: General Additive Model

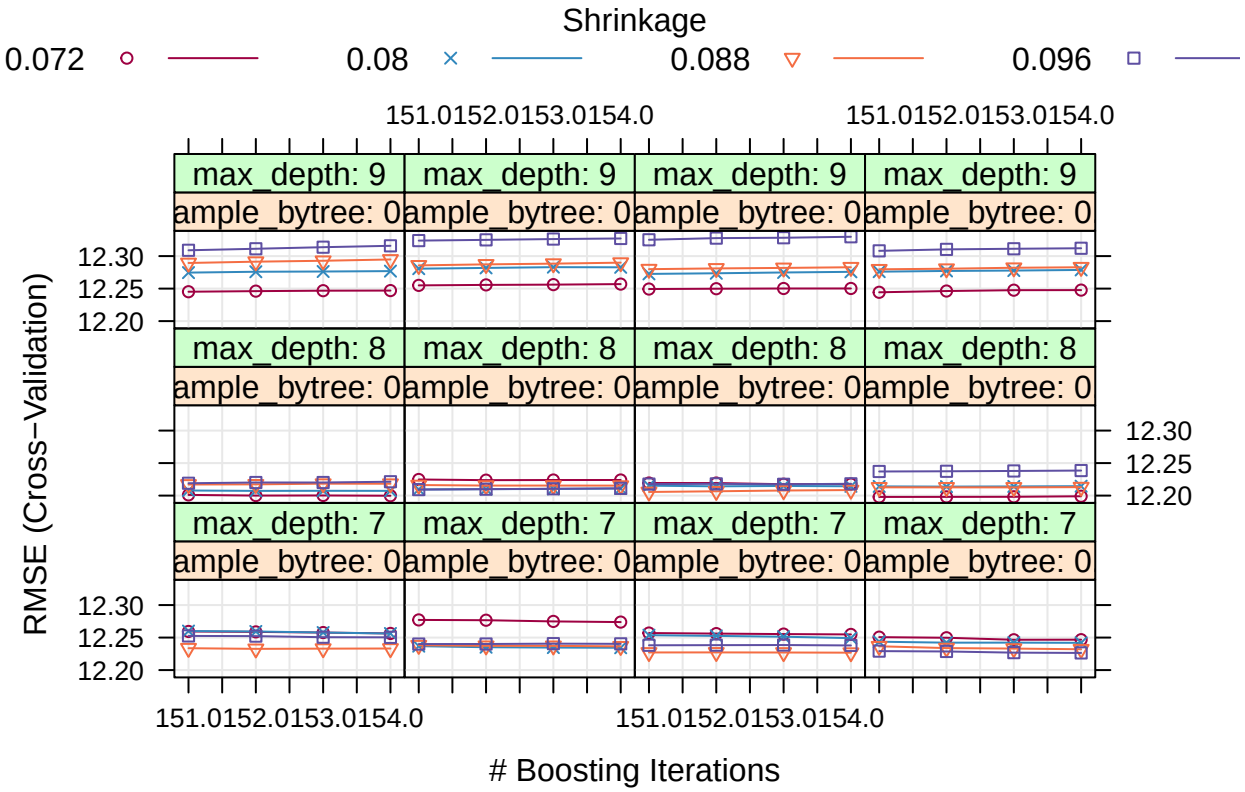

Figure 7: Extreme Gradient Boosting

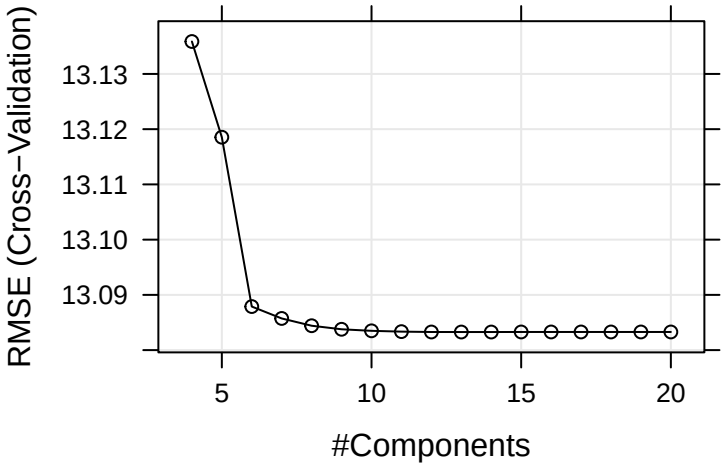

Figure 8: Partial Least Square Regression

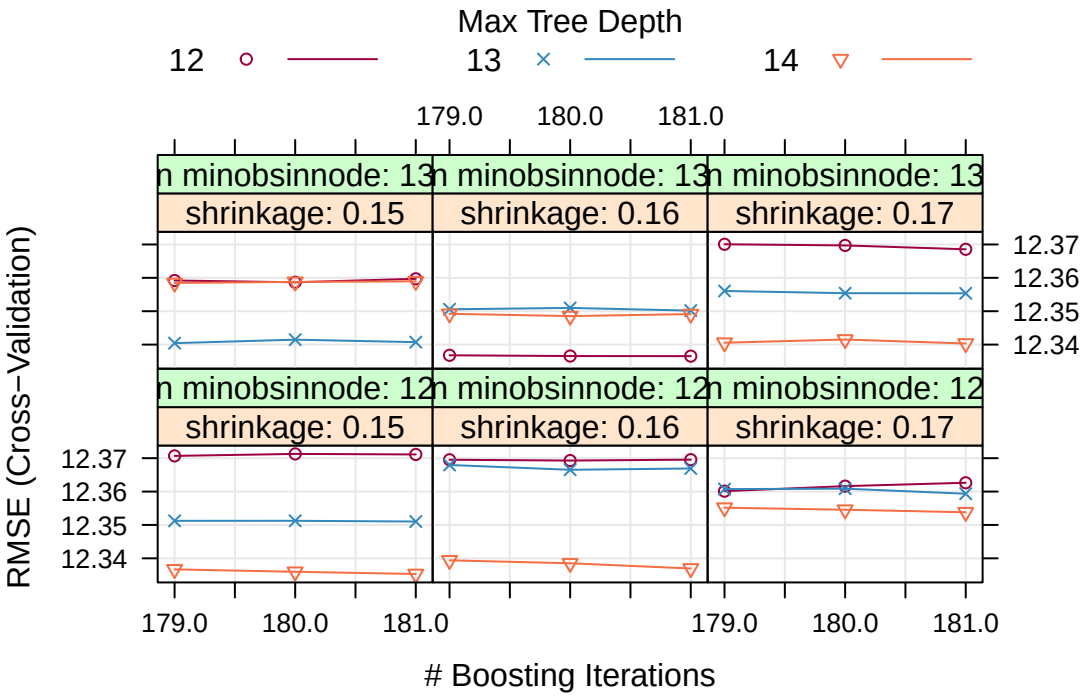

Figure 9: Stochastic Gradient Boosting

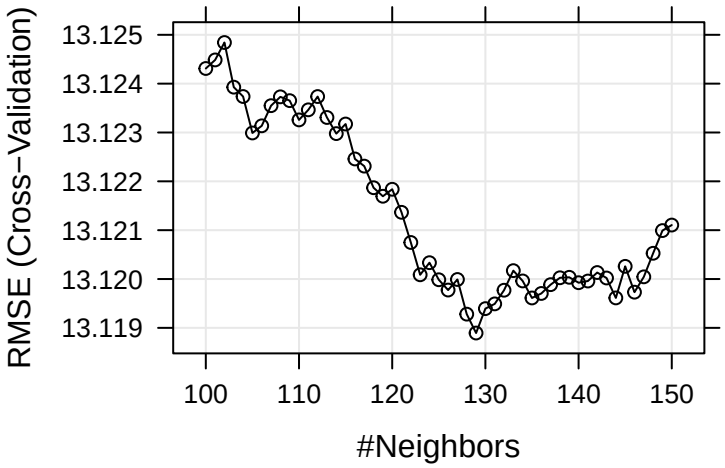

Figure 10: KNN for Regression

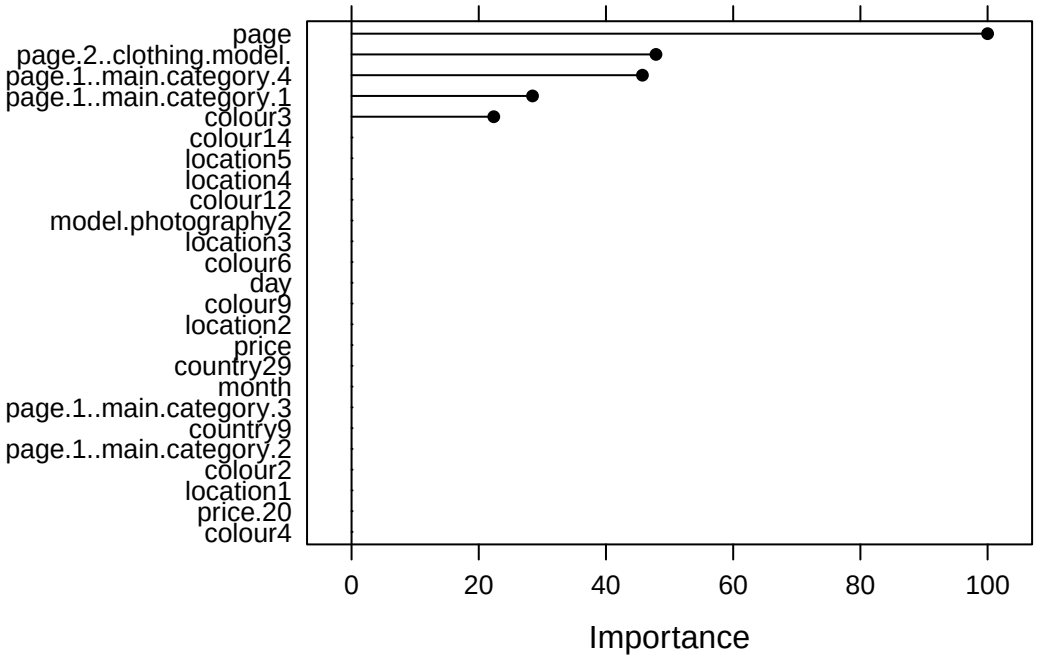

Figure 11: Bagging

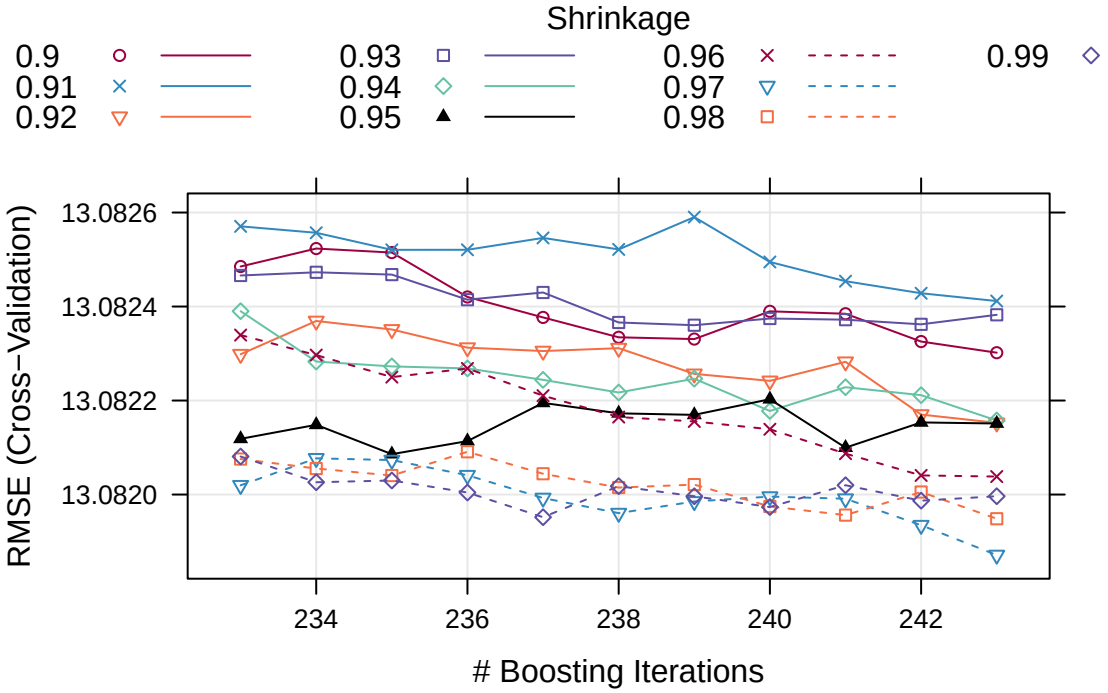

Figure 12: Boosting

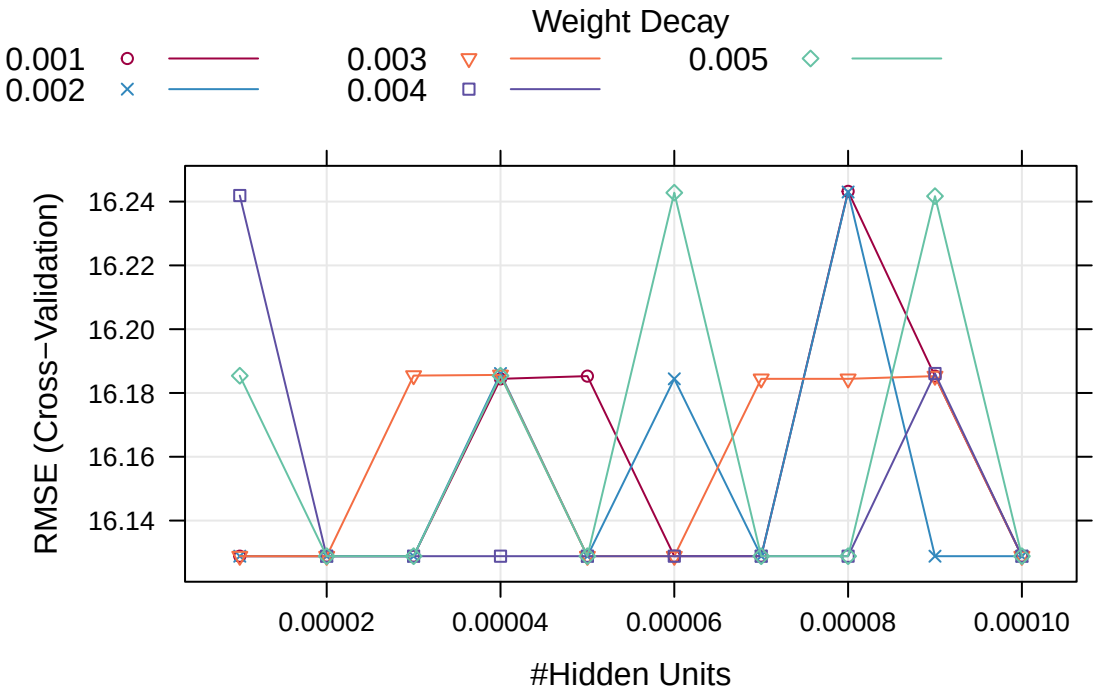

Figure 13: Neural Network

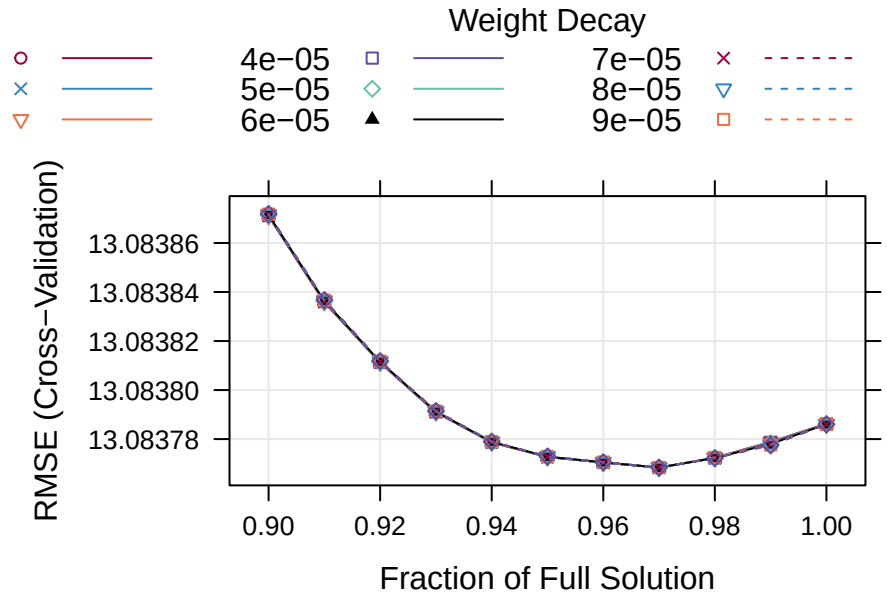

Figure 14: ElasticNet

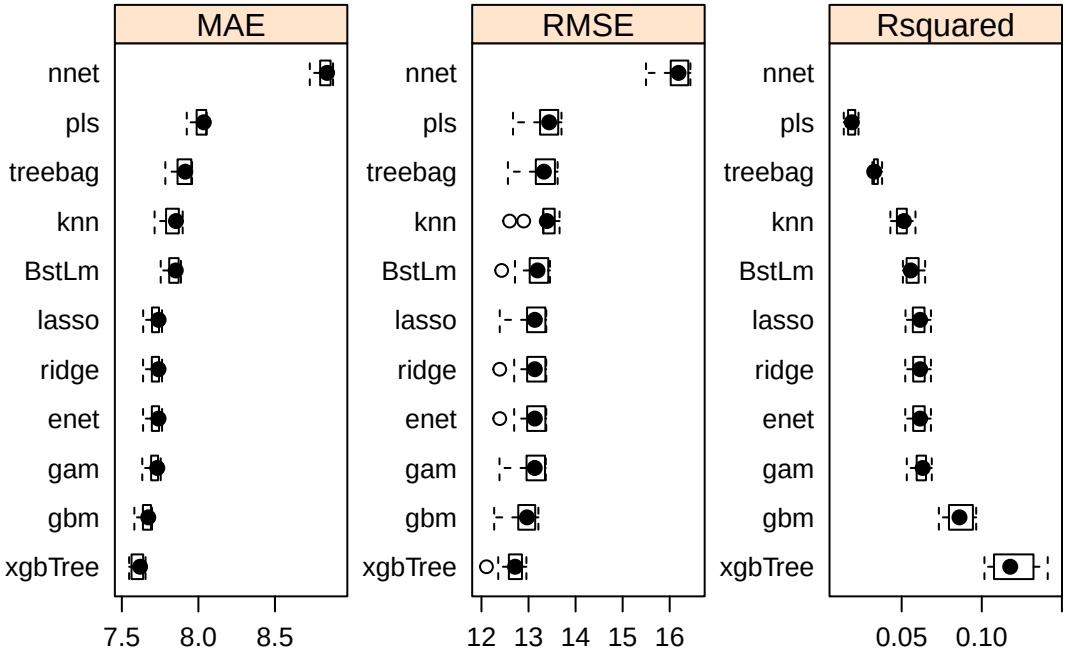

Figure 15: Ensembles

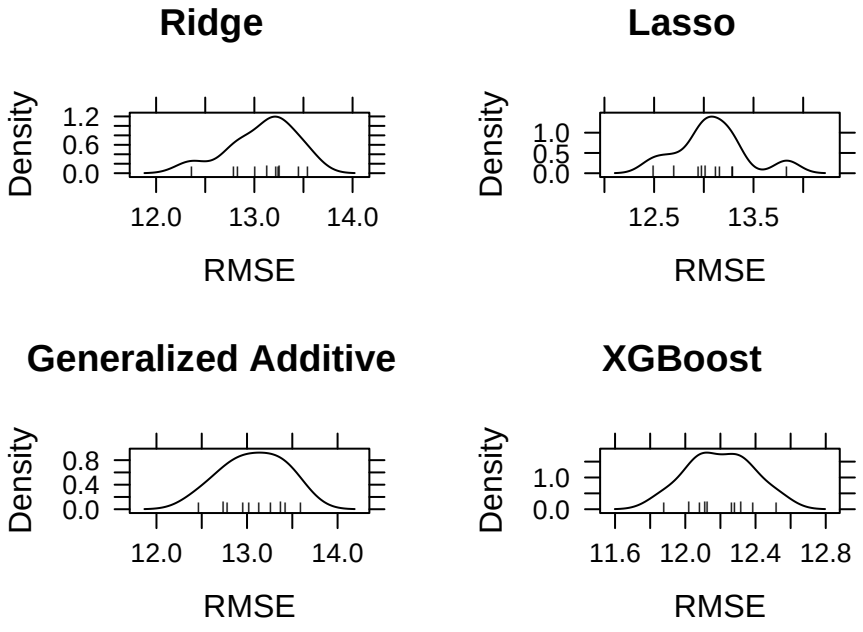

Figure 16: Density Plot for each model

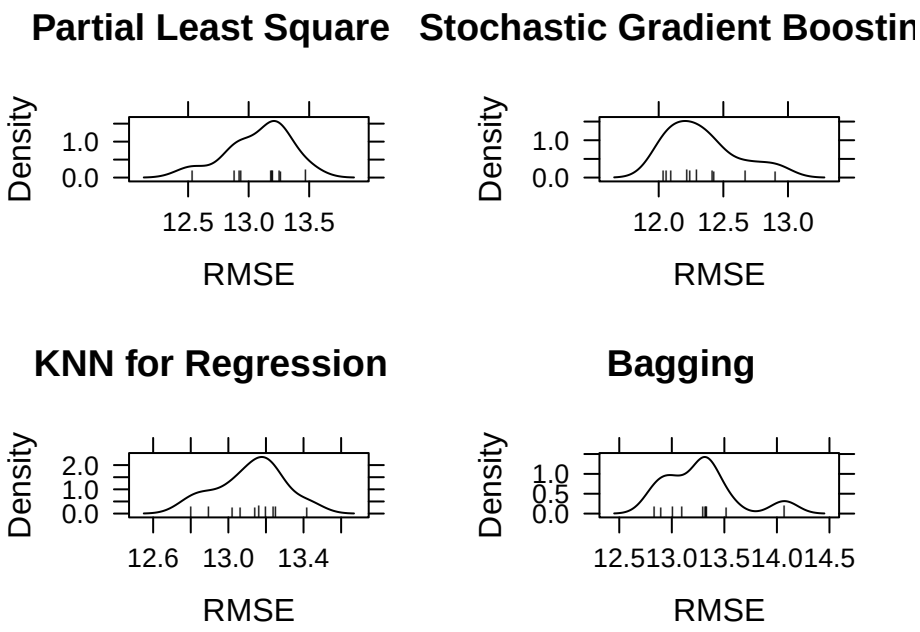

Figure 17: Density Plot for each model

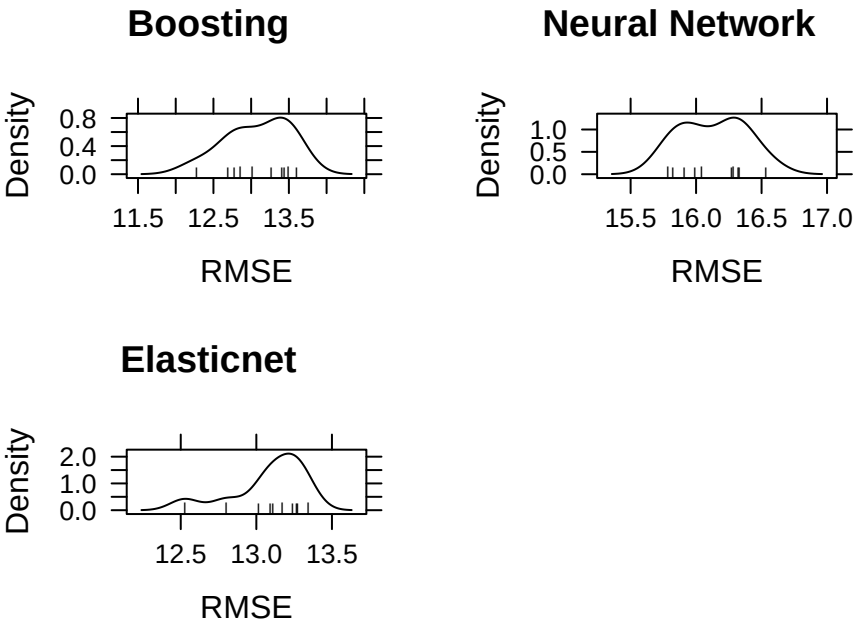

Figure 18: Density Plot for each model

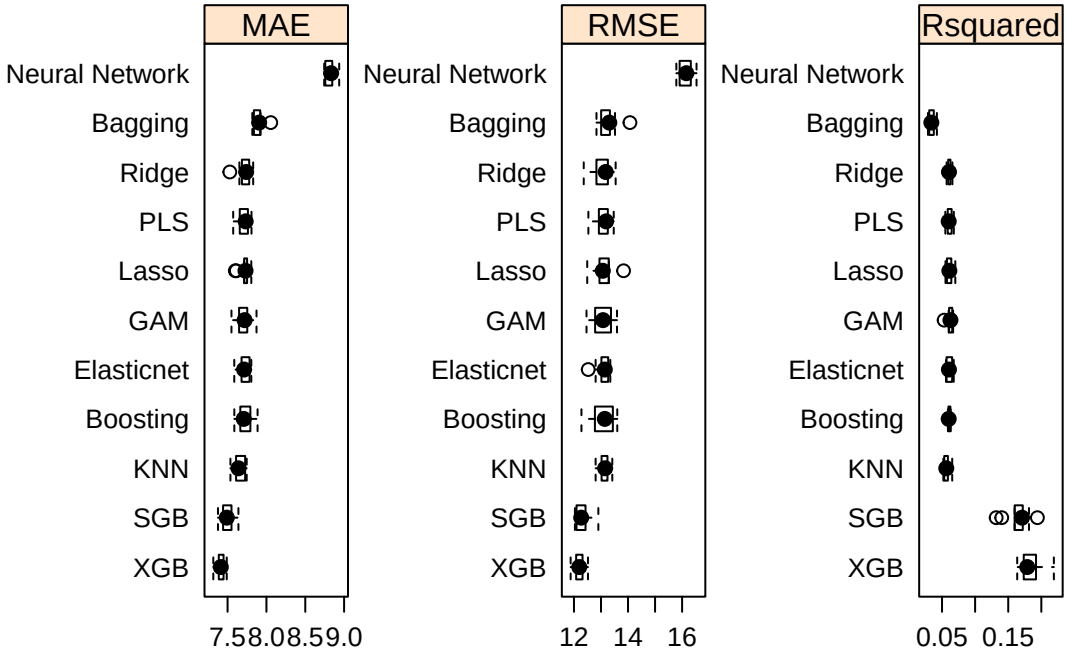

Figure 19: Comparison between Different Models