**Data analysis of Apple IOS mobile app user rating**

**Group 5 members:** Eleanor Zhang(zz2602), Mengran Ma (mm5354), Bingyu Sun (bs3142)

(Revision from previous report is highlighted in blue)

**Introduction**

Our previous data analysis examined the relationship between overall user rating of IOS mobile application(abbreviated as app) and 8 features including size in megabytes, price, average user rating for current version, content rating, genre of “Games”, number of supporting devices, number of screenshots shown for display, and number of supported languages. The raw dataset contains more than 7000 Apple iOS top trending mobile apps details in 2017. Model-based statistical analysis in previous report treated the outcome overall user rating as continuous variable. 10-fold cross validation with enlarged tuning parameter space (Fig 1.) leads to this MARS model which outperforms other models analyzed in the previous report:

3.35 - 0.187 h(user\_rating\_ver - 4.5) - 0.003 h(74 - size\_megabytes)\*h(4.5 - user\_rating\_ver) + 0.096 h(user\_rating\_ver - 3)\*prime\_genre - 0.035 h(4.5 - user\_rating\_ver)\*h(4 - lang\_num) + 0.46 h(user\_rating\_ver - 2.5) - 0.07 h(2.5 - user\_rating\_ver)\*h(sup\_devices\_num - 33)

including user rating of current app version, genre of “Games”, number of languages supported, size in megabytes and number of devices supported as well as their interactions in the form of hinge functions. High user rating of current version is strongly correlated with high overall user rating as shown in Fig 2. panel (A). Apps with very small size and low user rating of current version received even lower overall user rating at the left corner in panel (B). Games apps tend to have higher overall user rating than non-Games apps towards higher user rating of current app version as shown in panel (C).

However, given that user rating is highly discretized on the score scale from 1 to 5, regarding them as continuous variable may not fully describe its inherent nature. Furthermore, model-based statistical method has its own limitation and can be extended to model free as well as unsupervised approach. Therefore, our particular interest is to re-examine our research question using classification methods: what features can best predict high user rating versus medium/low rating of mobile app? This question can be addressed with a wide range of statistical methods to be discussed in this report.

**Data Preparation**

The distribution of overall user rating is in bell shape and slightly left skewed on the score scale from 1 to 5. Zero scores are removed from both overall user rating and current app version, because zero means no customer rating at the point of data collection. Only those mobile apps which have overall user rating score and nonzero current version rating will be considered. To simplify the classification problem, the outcome variable overall user rating is categorized into high rating (score ≥ 4) and medium/low rating (score﹤4). The filtered dataset with 5754 observations is partitioned into training set (75%) and testing set (25%). Training set is used to tune parameters as needed by cross validation (CV) and to compare CV errors across different methods and select final method accordingly. Testing data is used to confirm method selection.

**Exploratory Data Analysis**

The distribution of each quantitative feature in medium/low rating group and high rating group is roughly the same except number of screenshots shown for display and user rating of current app version (Fig 3.). For categorical variables, the genre of “Games” is observed way more in high rating apps within each level of content rating (Fig 4.). The distribution of content rating is similar in high and med/low rating group. Therefore, it is very likely the features user rating of current app version, genre of “Games” and screenshots number will be key variables in distinguishing high rating apps from medium/low rating apps.

**Materials and Methods**

Considering the computing efficiency, 10-fold cross validation with ROC curve summaries or accuracy for binary outcome predictions are used to tune parameters. Covariates are centered and scaled for classification methods based on distance (SVM and KNN).

***For linear decision boundary***

To begin with, linear methods with relatively strong assumptions are considered in terms of better interpretation comparing to flexible methods.

*Logistic Regression & Linear discriminant analysis (LDA)*

Logistic regression employs conditional probability (Y|X) to predict class labels, while LDA uses conditional probability (X|Y) and marginal probability of outcome to predict class labels. For normally distributed data, LDA is more efficient than logistic regression. Since our data from high rating group is roughly normally distributed, LDA may have improved model performance on predicting observations of high rating than logistic regression. The outcome of our dataset is binary, therefore, methods including LDA that are good at dealing with large number of classes may not be advantageous. Moreover, LDA outperforms logistic regression when data of two classes are well-separated. Our data, however, has intertwined observations from two classes. In general, LDA is not expected to improve our model much.

*SVM (Linear kernel: aka support vector classifier)*

A main difference between logistic regression and SVM is that only a subset of data (support vector) is used in SVM to decide decision boundary. Since high and med/low rating are not well-separated, support vector classifier allowing violation is chosen as the linear decision boundary for SVM. The final model selects cost to be 0.20, meaning support vector classifier in our model allows a moderate amount of data points inside margin or even misclassified. Only accuracy and kappa are appropriate to compare models containing SVM because SVM prediction is based on distance. In the report, ROCs are used to compare models. Therefore, only logistic regression is chosen for model comparison because logistic regression and SVM perform similarly on two non-separable classes.

***Nonlinear decision boundary***

When the underlying true classes cannot be separated by linear decision boundaries, nonlinear decision boundaries are utilized to improve model performance.

*Quadratic discriminant analysis (QDA)*

LDA imposes strong assumption on equal covariance across classes, our dataset, however, does not meet equal covariance assumption. Therefore, QDA may be a better model because it relaxes equal covariance assumption by fitting a nonlinear decision boundary.

*SVM (Radial kernel)*

Quadratic terms are added to the covariates to introduce non-linear decision boundaries in SVM. The optimal tuning parameters for the final model are 0.050 and 1.69 for gamma and cost respectively, indicating the final model has a relatively strong penalization on radial kernel. Since ROC is selected to compare models, SVM with radial kernel is not being compared.

*Naive Bayes*

Naive Bayes performs well when the number of covariates is large because it assumes that features are independent in each class. Our dataset has a small number of predictors—8 predictors in total, thus it is not expected to get improved model performance comparing to logistic regression, LDA, and QDA. The final model for Naive Bayes is relatively stringent with correction and kernel (fL=1, use kernel=true, bandwidth adjustment=1).

***Tree-based method for classification***

So far, our analysis is restricted by strong assumption imposed on the distribution of outcome or covariates, thereby model performance is susceptible to deviation from these assumptions. From this point on, these assumptions will be freed by tree-based methods, which also provide good balance of interpretability and prediction accuracy. The prediction accuracy will be further boosted by applying ensemble methods at the cost of reduced interpretability. However, some peeking into these “black box” by reporting variable of importance and partial dependence plot will reveal informative insights about how the algorithm predicts the class label of outcome.

*Single tree: CART classification tree and conditional inference tree*

To kick off, building a single tree with CV selected tuning parameter can serve appetite on the major contributor in separating high rating and med/low rating mobile app. The best CART classification subtree is pruned by complexity tuning parameter 0.0037 (Fig 5.). The splitting rules on the top of tree is user rating of current version. 92% of apps in the right most corner node are correctly classified as high rating by user rating current version more than 3.8. The majority of downstream nodes are classified by number of languages, size of app and number of devices supported. Conditional inference tree (CIT) is different from CART in stopping criterion by p value from permutation test. CIT ensures the right size of tree is grown without pruning and only select the covariates that most associated with outcome in each splitting step. From CIT tree result (Fig 6.), user rating of current version and genre of “Games” are the ones chosen. CIT tree and CART tree agree on using high current version rating as predictor for overall user rating. Since a single tree decision rules can be easily perturbed by small variation in training data, ensemble methods are implemented as followed in order to stabilize the prediction of overall high rating versus med/low rating of mobile apps.

*Bagging*

Bagging is a general-purpose procedure to reduce variance of a statistical learning method by averaging decisions made on bootstrap resampling data. The tuning parameter is minimal node size selected by CV. However, bagging will not reduce variance effectively if trees are highly correlated which caused by biased selection of strong contributor for each tree. This can be a concern in our dataset because as CART tree and CIT tree suggested, high user rating for current version can be a strong predictor for high overall rating score. Each tree built by bagging will very likely choose this predictor, which leads to highly correlated bagging trees.

*Random Forests*

Random forests method alleviates the problem arisen from bagging in our dataset by decorrelating the trees, using a random selection of features at each splitting step instead of considering all predictors. At each step, a random subspace with *m* features will be selected as candidates for splitting rule. Tuning parameters selected by CV include *m* and minimum node size.

*Boosting and Adaptive boosting*

Unlike bagging and random forests, boosting method grows trees in a sequential manner instead of relying on bootstrapped samples. It thrives on a weak learning manner, building small tree that aims to explain the remaining variation unexplained by the previous tree. Tuning parameters are number of trees to grow, learning rate, and depth of each tree. Because of this slow learning algorithm, boosting method can take long time to train because parallel computing is infeasible. Adaptive boosting is a further upgraded version of boosting. It strengthens the learning efficiency by assigning more weights to falsely predicted data in the previous tree.

***KNN***

K-Nearest-Neighbor classifiers is extremely flexible because it is model free and predicts class label based on classes of neighboring observations. Although KNN may provide good predictive model, it is unable to give information about contributing factors for our outcome—user rating. Our final KNN model selects tuning parameter—number of neighbors as 56 (k=56).

***Unsupervised learning: PCA***

Principal Component Analysis (PCA) are one of the major tools of unsupervised learning, revealing principle key components without considering outcome variable. PCA is applied to the feature space of entire dataset for the purpose of further exploration and confirmation of result findings from supervised method.

**Results and Discussion**

From the generalized linear model for binomial family, user rating for current app version, number of supporting devices, primary genre, number of supported languages, and number of screenshots shown for display are significant predictors. To predict the class labels, 0.5 is used as our cutoff for classifiers. The model has relatively good predictive ability because the accuracy is 0.86 on test data, indicating 86% of the observations are correctly classified. The agreement between prediction and observations is 51% (Kappa = 0.51), with 97% sensitivity and 46% specificity. In general, the model is efficient at predicting class of high score of user rating, but incapable of predicting data belonging to medium and low user rating scores. To improve our model, regularized logistic regression is used to introduce penalization. The final model selects ridge with extremely small penalization (alpha = 0, lambda = 0.0067), indicating regularized logistic regression and logistic regression perform similarly on our dataset. Based on CV ROC results, regularized logistic regression outperforms other linear methods (logistic regression and LDA), consistent with our data structure. Since logistic regression model fails to predict data from medium and low score class, a GAM model is fitted with nonlinear functions added to size megabytes, user rating for current app version, and number of supported languages. The covariates selected as non-linear component are based on exploratory analysis and F test. The GAM model has an increased accuracy of 87%, 61% Kappa, 94% sensitivity and 64% specificity. With GAM, data from med/low scores class is able to be correctly predicted. Since GAM from caret does not allow us to add nonlinear functions to specific covariates, GAM is not included in model selection even it exhibits better predictive potential than logistic regression.

Comparing across the CV results of area under curve (AUC) on Fig 7., boosting (2000 trees, interaction depth 7, learning rate 0.003) and adaptive boosting (4000 trees, interaction depth 8, learning rate 0.001) win over other methods and provide the maximal AUC. Since their CV results are very close, we chose boosting method to predict high rating score versus med/low rating and further investigate its prediction algorithm, although there is no straightforward way to present how it makes prediction based on covariates. The importance of variables can be evaluated by the total reduction of Gini index at each splitting node averaged across all trees (Fig 9.). User rating of current version is undoubtedly the most important feature in predicting high or med/low overall user rating. The partial dependence plot and local effect plot displays the global marginal effect and local effect of this predictor (Fig 10.). High rating of current version will generally admit high overall user rating. This is consistent with findings from exploratory data analysis and simple tree method. It is reasonable to expect a mobile app with high rating score for current version will also be rated highly overall for all version.

Each of statistical method applied in training set are used to predict outcome in testing set. The result displayed in Fig 8. indicates bagging gives higher AUC value than boosting, which does not agree with the model performance in training set. A plausible reason for this is that the dataset is predominantly mobile apps with high rating. Therefore, intuitively speaking, a biased selection of strong contributor in bagging will predict results well.

From PCA result, the correlation of each continuous variable with each principal component (PC) direction is shown in Fig 11. Size of app, price, number of devices supported, and number of screenshots are major loadings for the first PC. User rating of current version contribute the most in second PC. After adding a layer of the outcome variable, the PCA biplot (Fig 12.) implies apps with high overall user rating have more screenshots displayed number of languages, larger size, and higher price. User rating of current version does not seem to have an over-dominant contribution in the unsupervised analysis unlike its role in supervised method.

**Conclusion**

Overall, user rating of current version is a primary predictor evidenced in many techniques used, along with other secondary contributors such as app size, number of supported language and genre games. It is reasonable to expect app with high rating of latest version can be rated highly for all versions on average. The size of app can indicate large volume of visuals and customer friendly built-in functions, for example Games usually can take up considerable storage space in devices. More languages available in the app allow more diverse customer population. Thus, our findings can be helpful in predicting high rating apps based on these features.

**Reference**

1. Data website: <https://www.kaggle.com/ramamet4/app-store-apple-data-set-10k-apps>

2. Project Repository on Github: <https://github.com/zixuanzhang/DS_MTproject>

**Supplementary**

Variables:

* ***User rating for all version*** (outcome variable): user rating on the scale from 1 to 5; “0” means the app is not rated yet (removed for model fitting)
* ***Size megabytes***: To improve computation efficiency, the units of size bytes is changed into megabytes; 96% apps take less than 1000 mb.
* ***Price***: 99% apps cost less than $10; some apps requires over hundreds of dollars which might be annual subscription fee.
* ***User rating for current app version***
* ***Number of supporting device***
* ***Number of screenshots shown for display***
* ***Number of supported languages***: range from 0 to 75 with mean 5.
* ***Vpp Device Based Licensing Enabled***: binary (removed due to near zero variance)
* ***Content rating***: the age group suitable for use with 4 levels (categorical).
* ***Primary genre***: it includes 23 types of app; Games genre is oversampled, so this variable is coded as binary variable of being “Games” or not.

Figure 1. MARS model tuning parameter selection

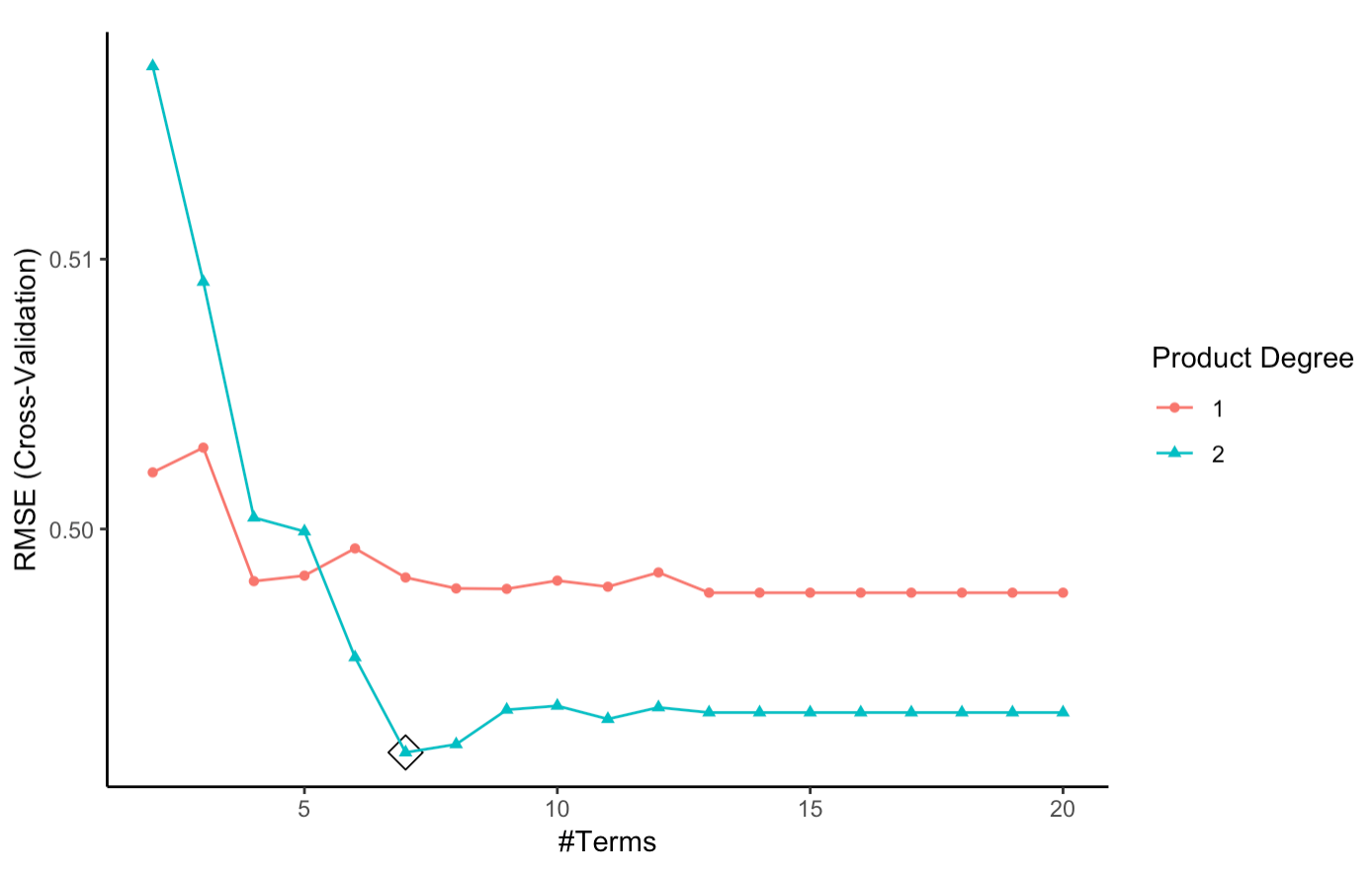
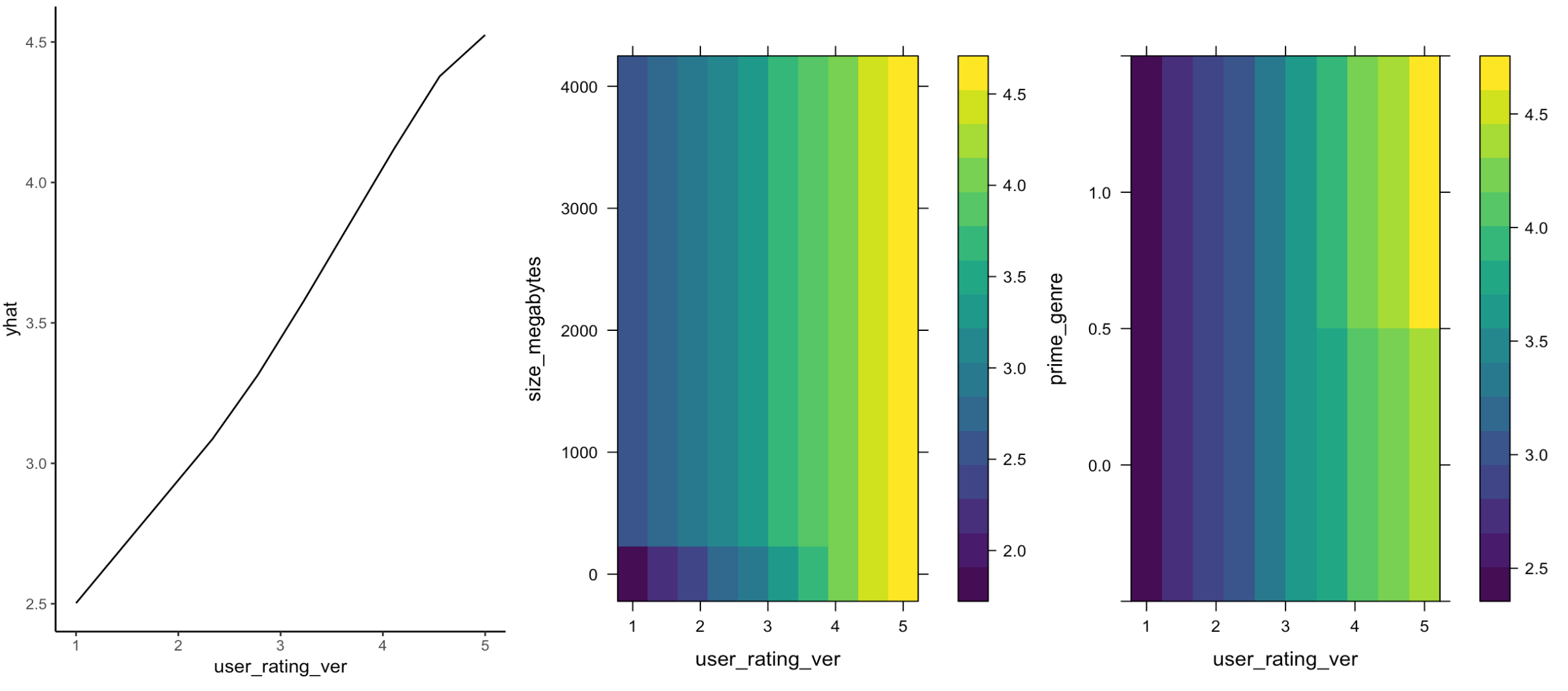


Figure 2. Partial dependence plot for marginal effect and interaction effect in MARS model



1. (B) (C)

Figure 3. Distribution of quantitative features for apps with high rating and medium/low rating group

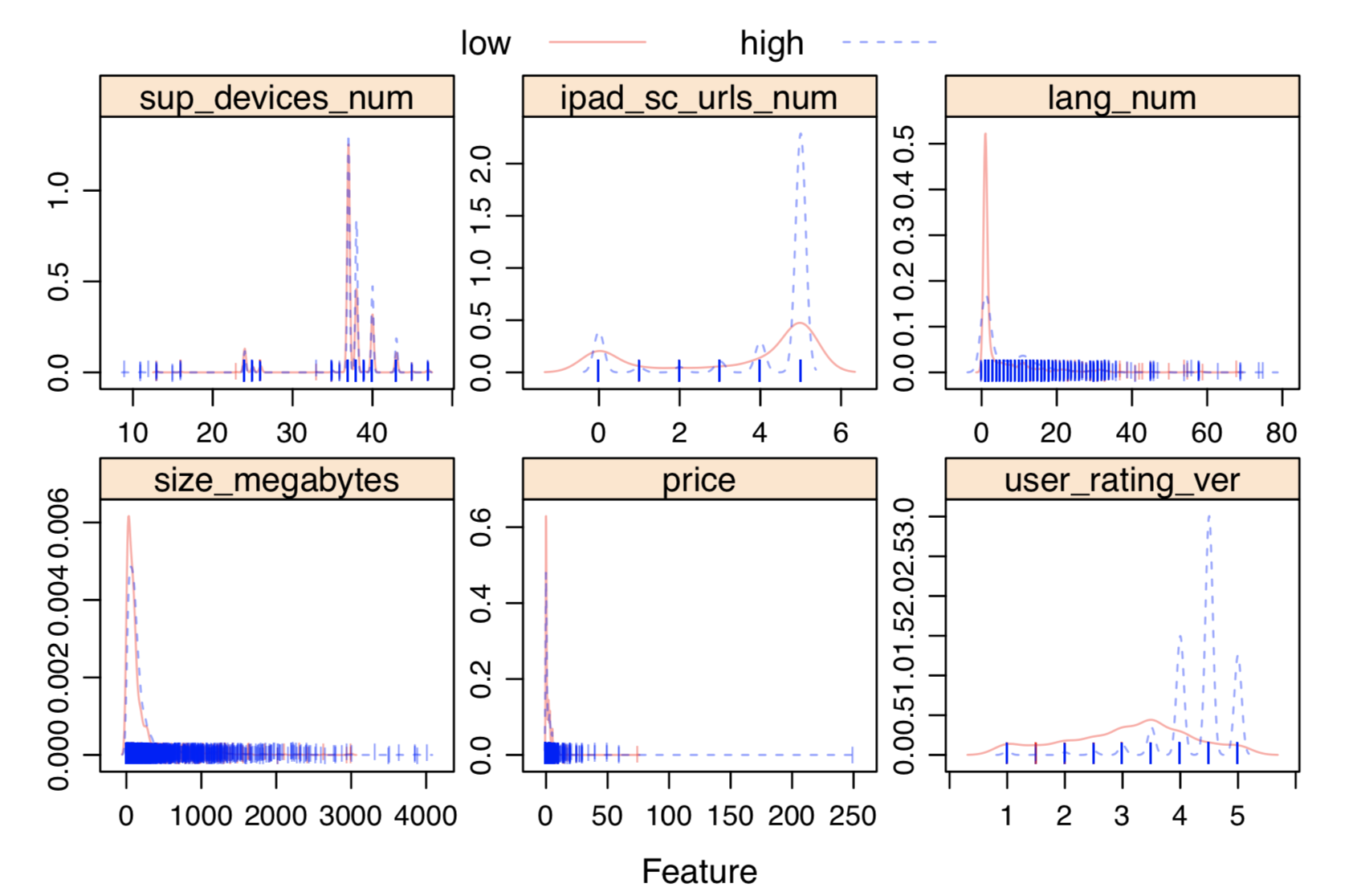


Figure 4. Frequency of “Games” apps in each content level within high rating and med/low rating group

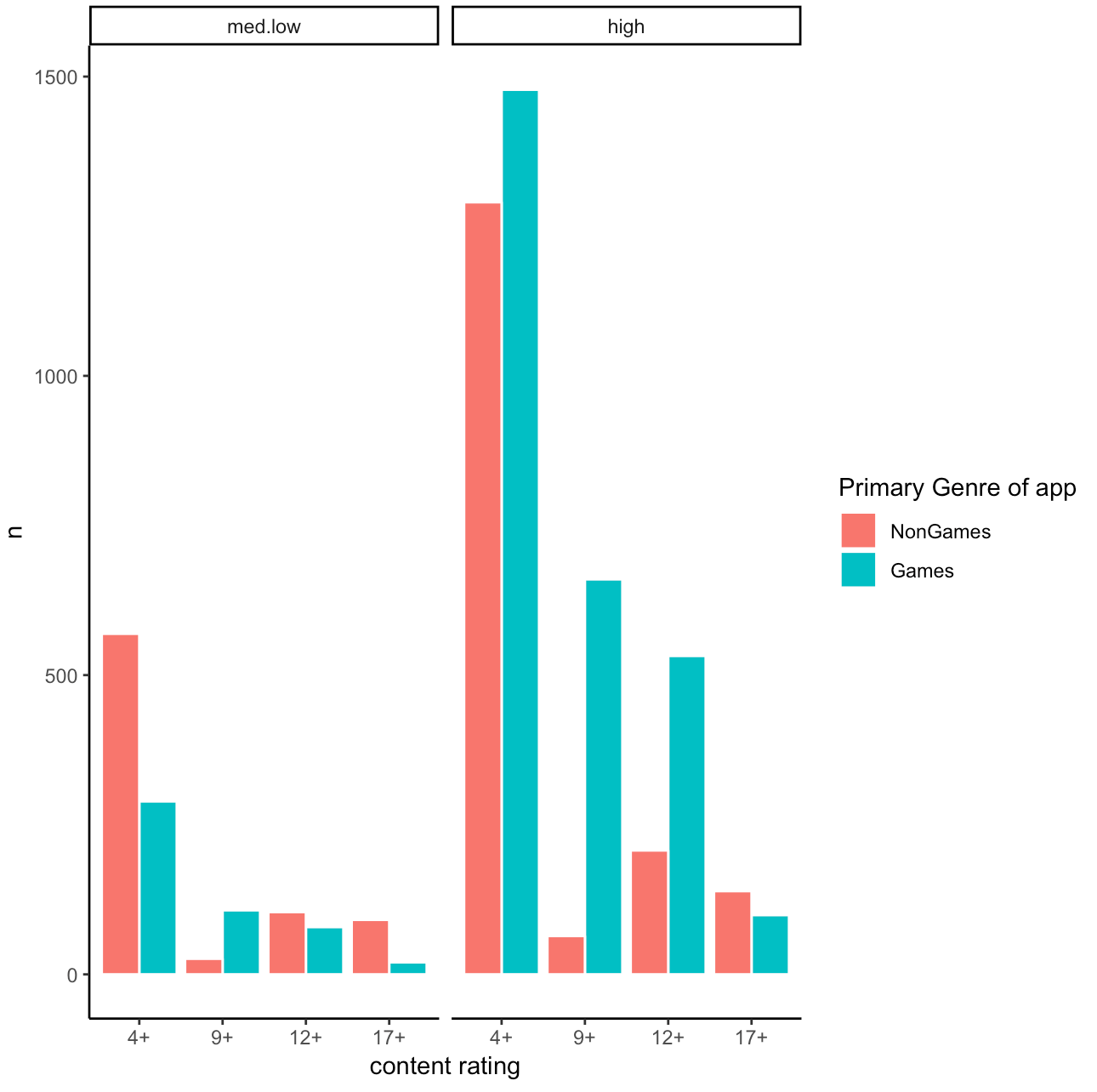


Figure 5. CART classification tree (complexity parameter = 0.0037)

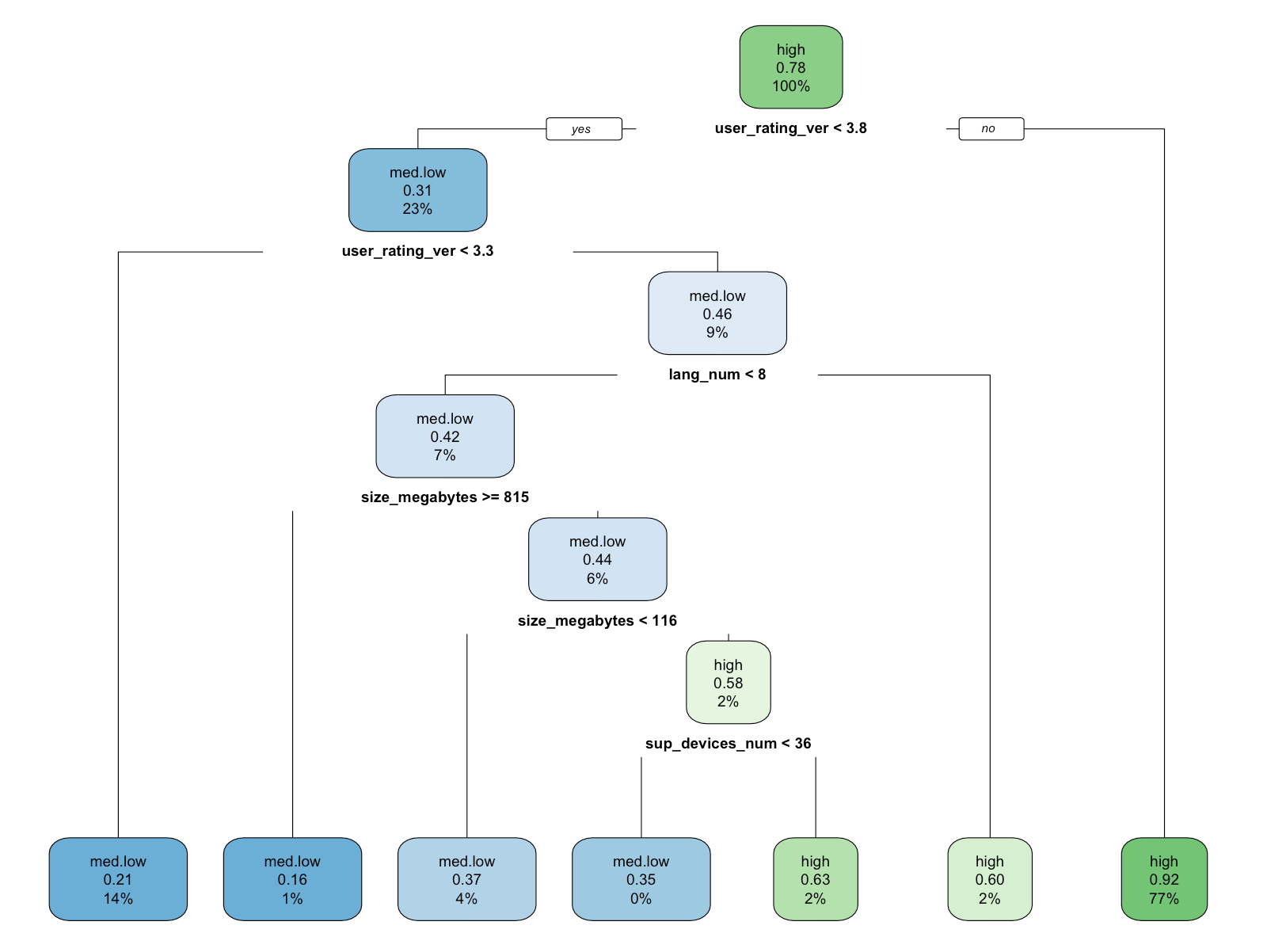


Figure 6. Conditional inference tree with criterion 1- p.value = 0.894

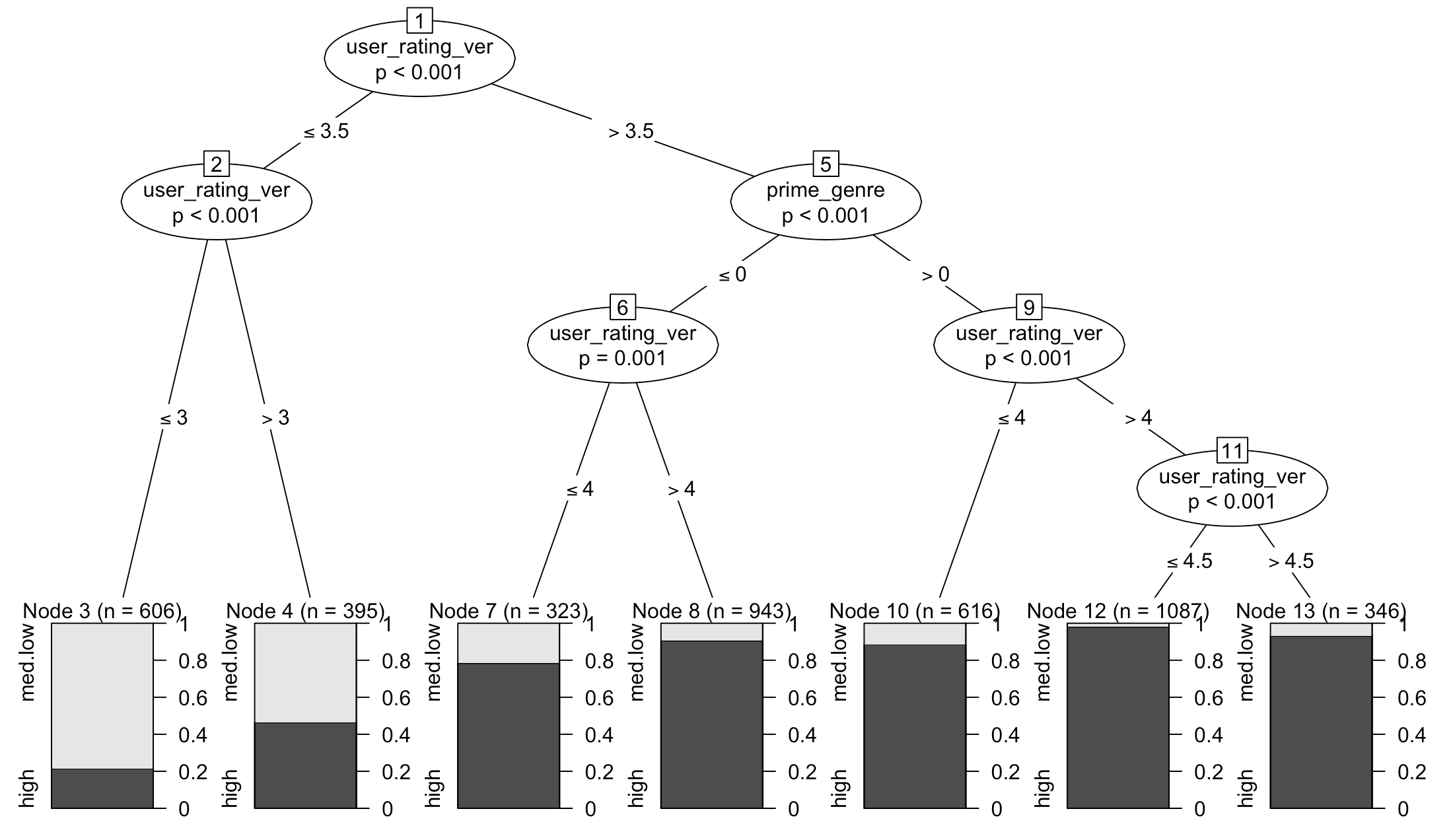


Figure 7. Cross validation ROC result between different statistical methods on training dataset

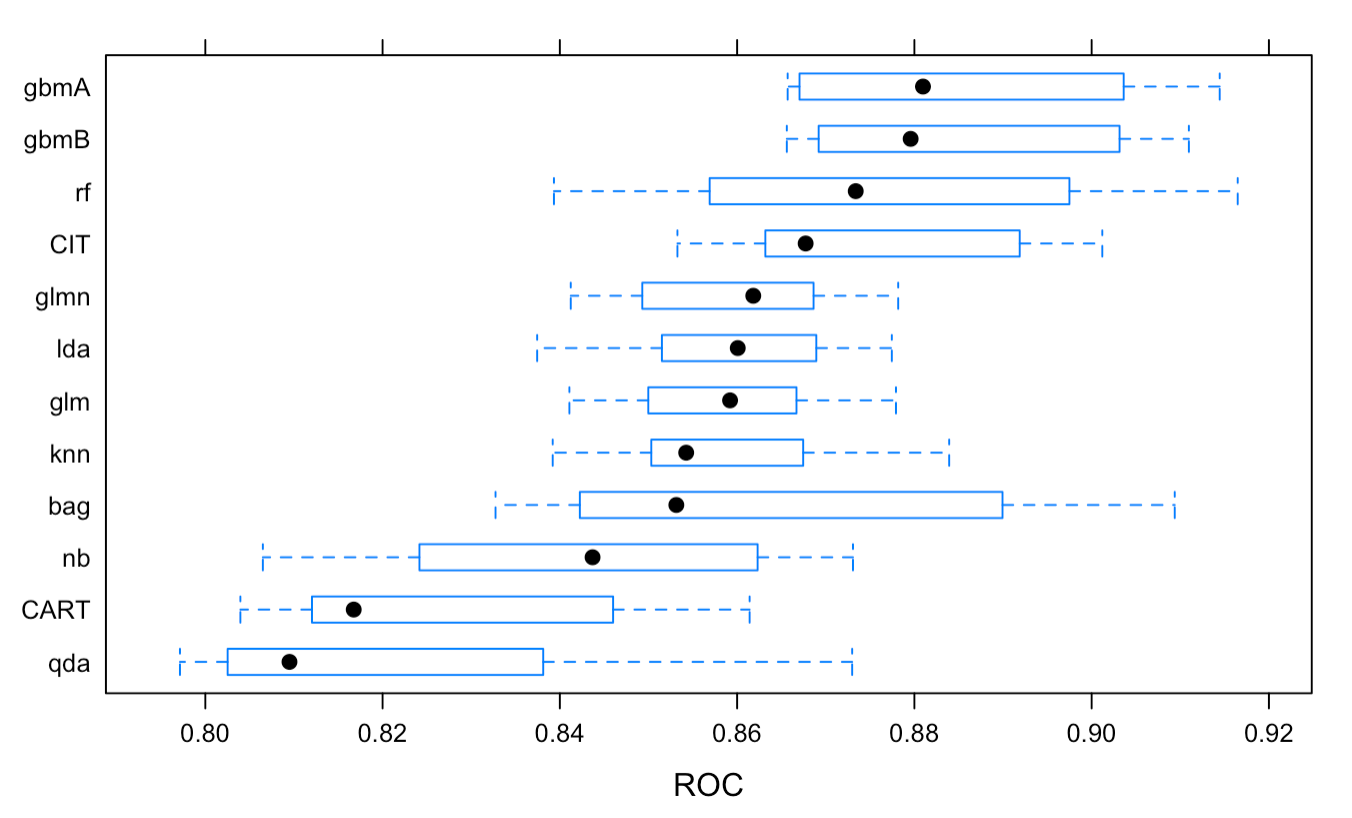


Figure 8. ROC result of different methods on testing dataset

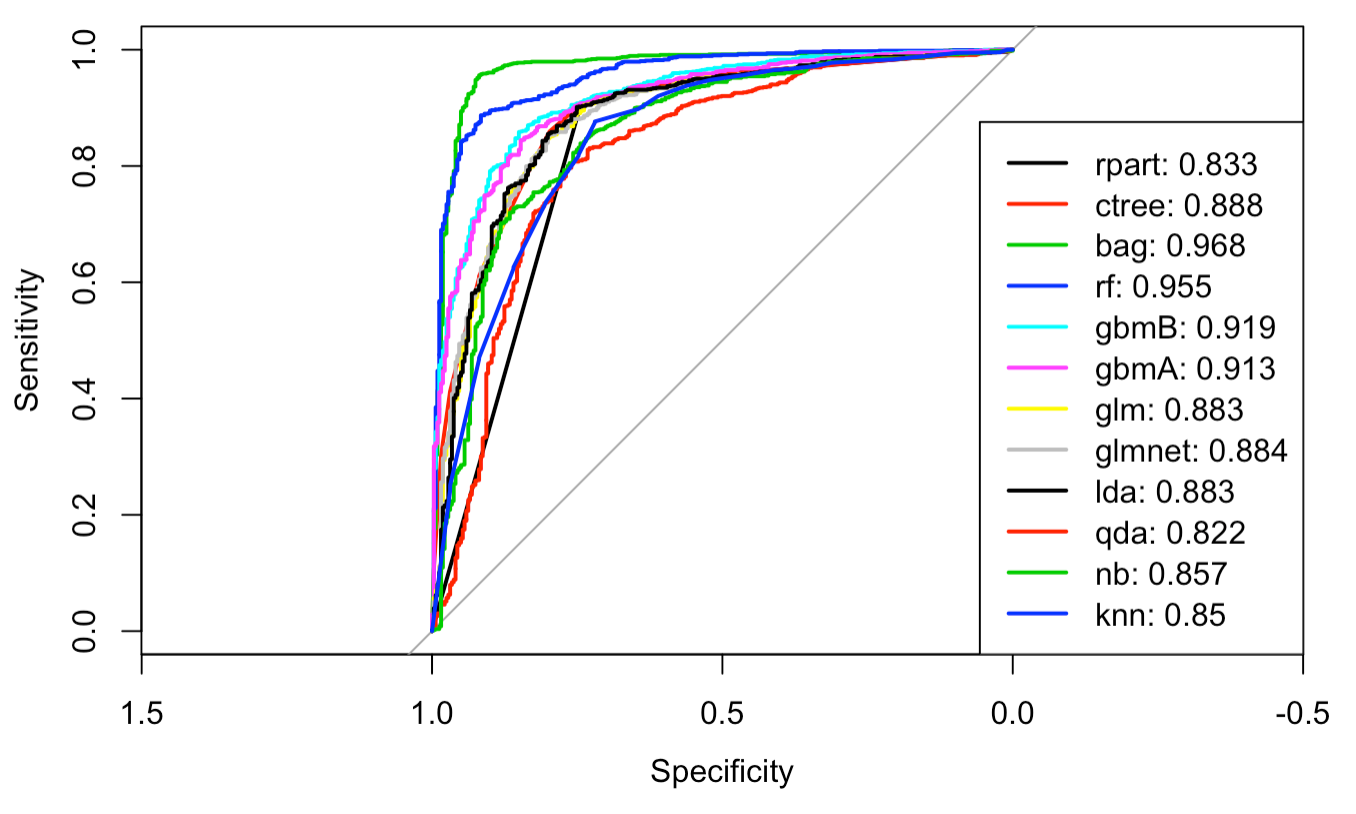


Figure 9. Variable of importance in boosting method

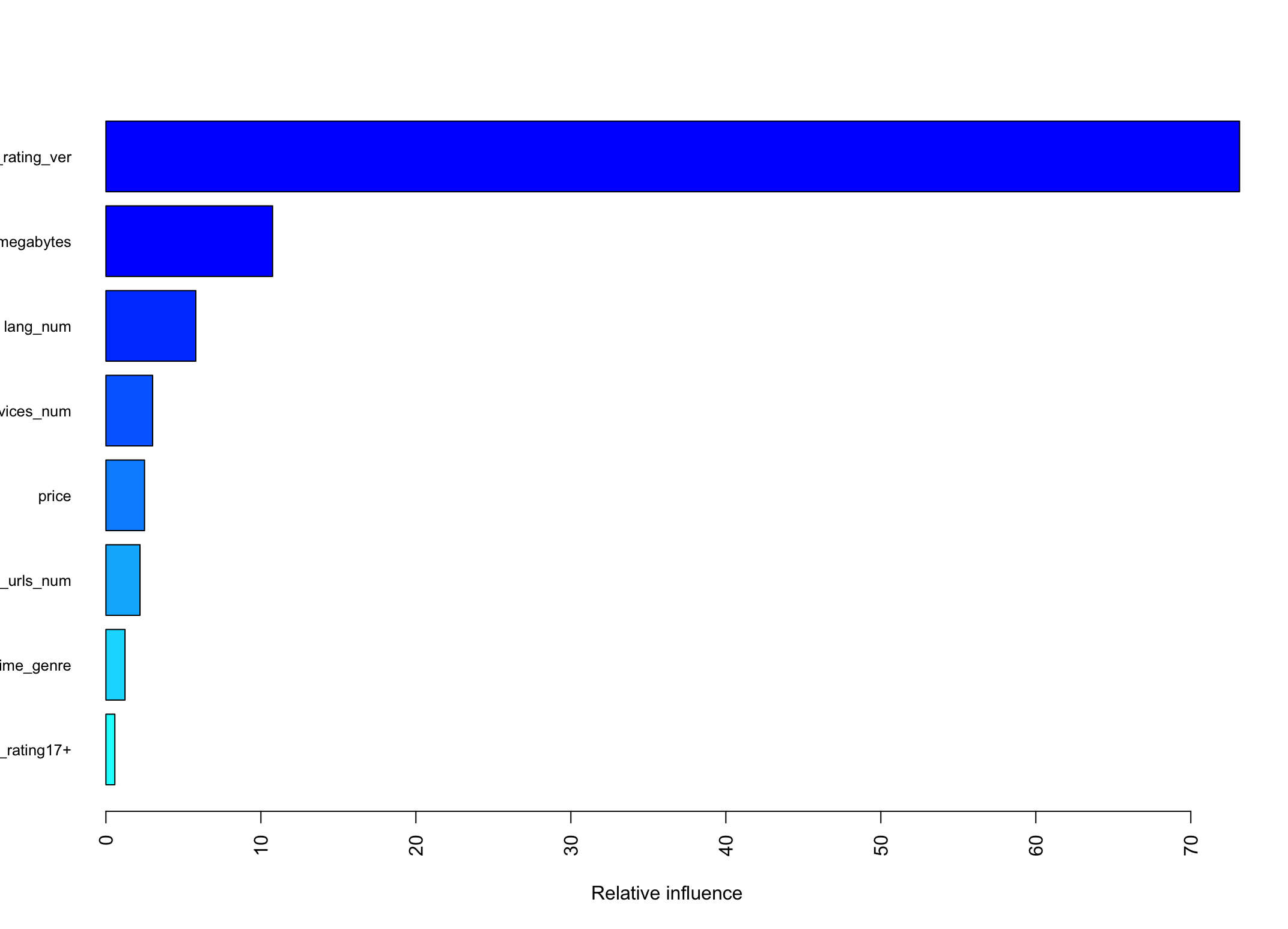


Figure 10. Partial dependence plot(left) and local effect (right) of user rating of current version in boosting method

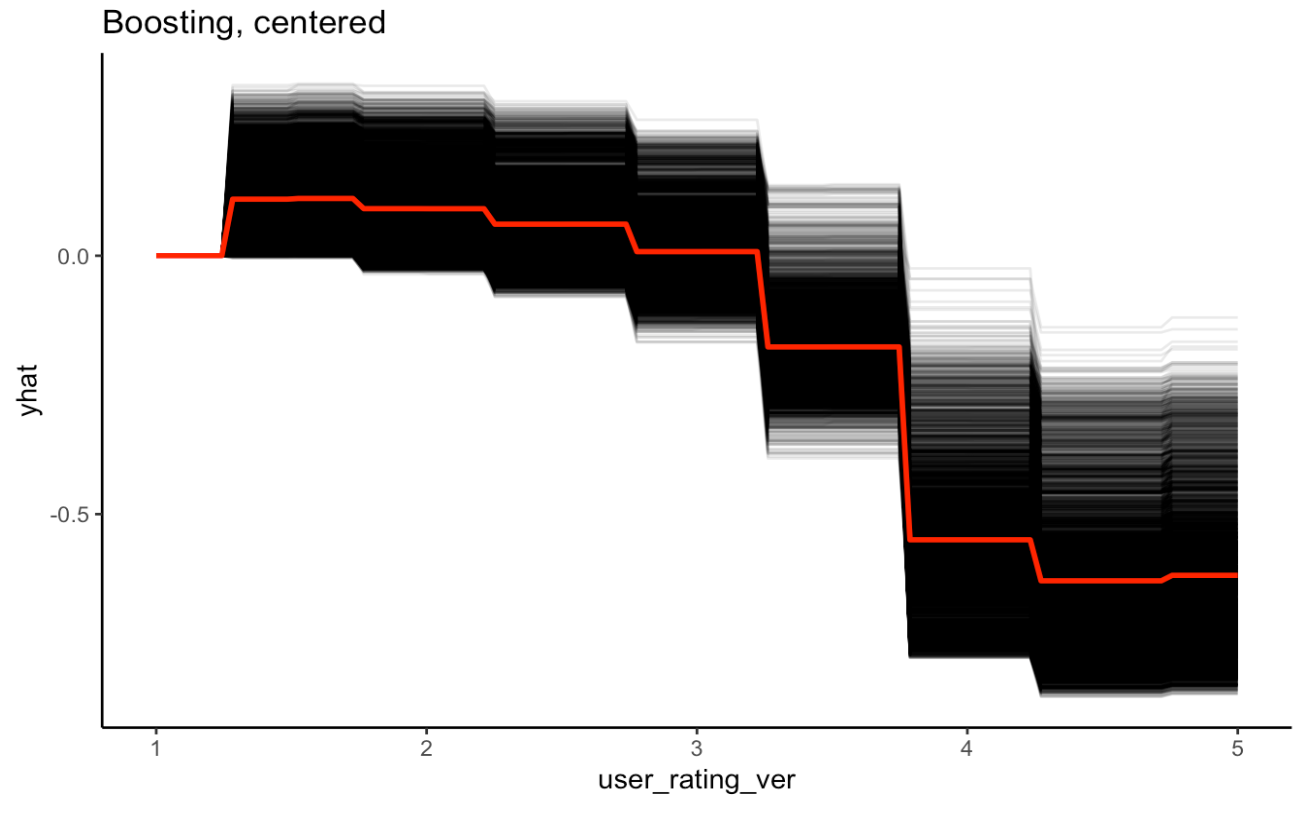
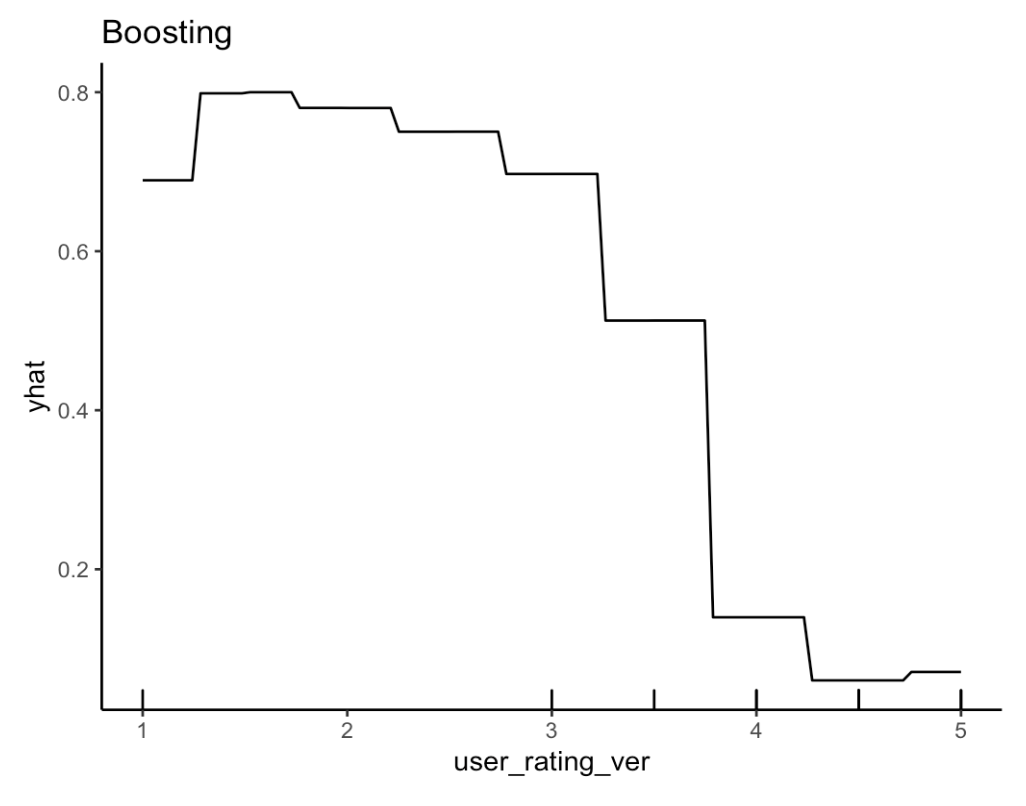


Figure 11. Correlation of each continuous predictor on PC directions

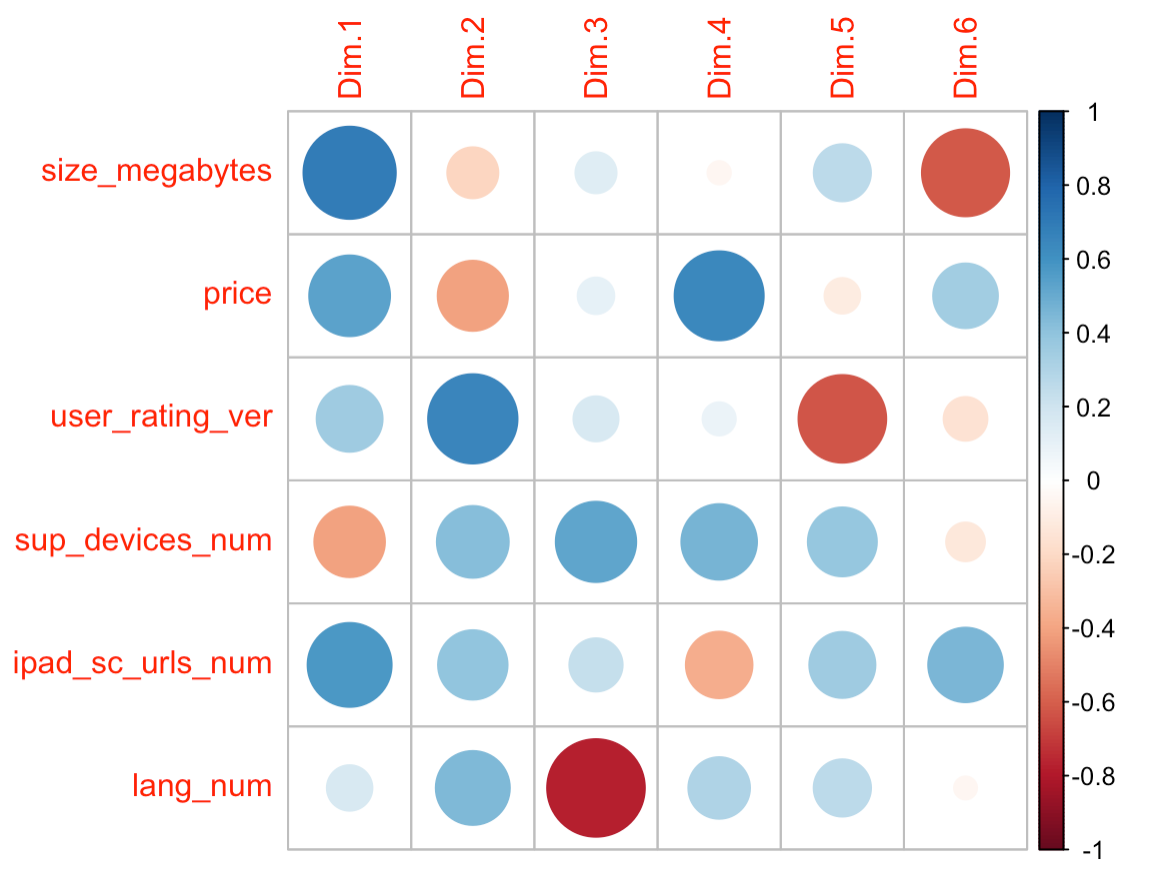


Figure 12. PCA biplot for outcome variable

