Supervised Machine Learning

Final Assignment

A Comparative Analysis of Random Forest and Regularized Regression Approaches on Predicting Success of Online Music Courses

November 22, 2022

630510

1 Introduction

Udemy is one of the most popular platforms for online learning in recent years. Due to the COVID-19, it has gained even more significance as it offers many free and beginner level courses on almost any subject, including music. It is likely that these courses have been successful in helping students learn and improve their skills in music, as many of them have received positive reviews from students. However, it is difficult to provide specific information on the success of Udemy's musical instrument courses, as this information is not publicly available. The analysis of the determinants of the subscriber number can help similar companies or content creators to optimize the success of their courses. Therefore, we attempt to answer the research question:

To what extent random forest and regularized regression techniques predict the success of a Udemy course on musical instruments by considering courses' price, number of reviews and lectures, level, rating and the content duration?

2 Data

The original data-set contains 3.682 records of courses from 4 subjects (Business Finance, Graphic Design, Musical Instruments and Web Design) taken from Udemy. (Willden, 2019) For this paper, we opted to use the courses on Musical Instruments to narrow down our focus. The data-set is also publicly available on Kaggle. The data set contains 680 observations with 7 different features on number of subscribers, price, number of reviews, number of lectures, target level (whether it is a beginner, intermediate, advanced or all-levels course), rating and content duration in hours. Our initial checks suggest that we do not have any omitted variable problem for this data set. The dependent variable that will be used in this analysis is number of subscribers. All of the remaining features will be used as independent variables. One thing that we find particularly interesting about this data set is that it enables us to isolate the effect of the number of reviews so that we will be able to see the correlation between rating and number of subscribers more clearly. A short summary of the data set can be seen in the following table:

variable name	mean	variance	description
num subscribers	1245	0.32221306.0	number of users subscribing the course
price	49.6	1696.41774	price of the course
num reviews	47	114065.39	number of reviews that the course has
num lectures	38.3	1826.6819	number of lectures offered in the course
level	x	x	dummy variable indicating the level of course
Rating	0.309	0.092294	Rating of the course on a scale of 0 to 1
content duration	2.85	12.044115	the total number of hours offered by the course

3 Methods

To answer our research question, we will utilize regularized regression, namely Ridge and Lasso regression as a first step. In the next step, we will implement bagging and random forest to provide a comparative analysis with regularized regression techniques. First step should be more efficient if the data generation process (DGP) is close to being linear or higher degree polynomial whereas the second step comes in handy in case we have DGP which is far away from being linear. This is an important question because we would like to have a better idea of DGP if we would like to do some forecast or prediction with the existing sample. However, one should note that none of these methods are sufficient in making causal interpretations on their own as they face severe endogeneity issues. Furthermore, it is infeasible to even interpret random forest techniques due to its high-dimensional and non-parametric nature.

Risk of over-fitting is one of main problems when researchers develop a model for prediction. Ridge and Lasso regressions (Hoerl, 1962; Hoerl and Kennard, 1968; Tibshirani, 1996) are the regression models where over-fitting is taken into account by imposing a penalty on the loss function. In the case of Ridge regression, all of the regressor coefficients are penalized equally. They work very well in cases where we might have multicollinearity or in cases where we have many predictors. Assuming that we have n number of observations with p number of dependent variables, following equation characterizes loss function of a Ridge regression that we want to minimize:

$$L^{\text{ridge}}(\beta) = (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) + \lambda\beta^T\beta, \tag{1}$$

where $\mathbf{y}$ is output variable vector of size n, $\mathbf{X}$ is a matrix of input data of size $n \times p$, β is a vector of coefficients of size p, and λ is the positive hyperparameter that influences the Ridge penalty. In case we have λ¿0 as a result of this minimization, it means that we have shrunk β towards 0. The analytical solution of the Ridge regression is the following:

$$\hat{\beta}^{\text{ridge}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}. \tag{2}$$

One should note that standardization is an important part of Ridge regressions as the variance of $\mathbf{X}$ affects the results. Therefore, we also standardize the columns to z-score.

As stated above, Lasso regression also puts a penalty on the loss function to tackle the over-fitting problem. The difference with the Ridge regression is that it penalizes the sum of absolute values of coefficients rather than the square of coefficients. Due to its nature, it does a better job than Ridge regression if there are few non-zero coefficients and a lot of coefficients which are close to zero. With the similar logic, we can argue that it performs worse than Ridge if we have similar values for coefficients. The loss function for Lasso regression is given as the following:

$$L^{\text{Lasso}}(\beta) = (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) + \lambda||\beta||_1, \tag{3}$$

where we have the same notation as equation 1. Note that Lasso regression does have a closed form solution as the loss function is not differentiable.

One important note regarding the regularized regression is the choice of hyperparameter. In Lasso and Ridge, one technique is to use k-fold cross validation to get the optimal λ. Briefly, the algorithm divides training data into K non-overlapping folds and uses each fold as a test set and remaining as training set at a time. We calculate the Mean Squared Error (MSE) for K times and we find the λ that minimizes the root of average of these MSEs. It is important to choose the same λ for the comparability of our results in our study.

Both bagging and random forest (Ho, 1995) are the techniques which uses decision trees. It works by creating a tree-like model of decisions, with each internal node representing a decision based on the value of a specific feature, and each leaf node representing the outcome of the decision. A number of decision trees are trained on different random subsets of the data that we obtained via bootstrapping. The predictions of the individual trees are then combined to make the final prediction, using a soft or hard voting or just averaging the bootstrap estimates. This way, we can reduce over-fitting since each single decision tree are less likely to over-fit the data when they are trained on only a subset of it.

Bootstrapping is a sampling technique where we sample from the data we have with replacement. Each bootstrap sample should be the same size as the original dataset. In bagging and random forest, we utilize each bootstrap sample to train our decision trees.

For bagging, we follow the following equation to make a prediction in a regression case:

$$\hat{f}_{avg}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x), \tag{4}$$

where B is the number of bootstrapped samples and $\hat{f}^{*b}(x)$ is the estimates obtained from each sample b $\in$ B. We know that when we do bootstrapping, each bootstrap sample b contains only about two thirds of all observations. Therefore, each bootstrapped sample comes with its own unique test sample. These are called out-of-bag observations and we utilize them to test the error.

The difference between bagging and random forest is that in random forest we can focus on only a number of independent variables (features) whereas in bagging, number of features that we use to train our decision trees should be equal to the number of available features. Random forests work better when each individual tree is correlated with each other. The following equation is helpful explaining the reasoning:

$$Var(\sum_{b=1}^{B} Z_j) = Var\sum_{b=1}^{B}(Z_j) + 2\sum_{1<j<k<B} Cov(Z_j, Z_k), \tag{5}$$

where each Z is the decision tree trained in each bootstrapped sample b. By focusing on only a subset of features when each split is made, we can decrease the correlation among trees which corresponds to the second term of the right hand side of the equation above. Because of this, random

forest is doing a better job than bagging when there are only few strong features.

4 Results

Before we present the results of our analysis, there are some points worth mentioning. As we have explained in the previous section, k-fold cross validation is utilized to select the best lambda for the analysis. We decided to use the implementation that is already embedded in the *glmnet* package rather than writing our own algorithm. For both Lasso and Ridge regressions, we report the optimal lambda values. Furthermore, we also tried to use the same values of lambda for comparisons of coefficients. However, this does not change any result so we only report the coefficients that we obtain by using the optimal lambdas of each regression.

Second point is about the implementation of ridge regression. Since we have only few independent variables compared to the number of observations, we decided to replace each predictor by its polynomial basis and modeled the interaction effects. This way, we have a better prediction. However, the trade-off is that interpretability of the coefficient is now harder. That's why, we present the results for both with and without replacement of polynomials.

After standardizing our data, we first run a ridge regression where the dependent variable is the number of subscribers. The lambda value which minimizes the loss function for ridge is 790.6. Due to the reason explained above, we also run a ridge regression where we replace predictors with their polynomial basis. We tuned the model to make interaction matrix close to a square matrix as much as possible. The lambda value which minimizes the loss function for this is 1389. In figure 1, RSMEs for different values of lambda for standard ridge and polynomial ridge can be seen. The lowest RMSE we can get for ridge regression is 4068 whereas for polynomial version it is 5445. This suggests that ridge regression is doing a better job than polynomial one even though we have few explanatory variables.

In figure 2, we can see the RMSE for different log values of lambda for Lasso regression. The minimum best lambda we get is 175.8 which is much lower than lambdas for ridge regression. If lasso regression does not penalize much, this potentially means that there are not many high influence features. The lowest RMSE we get is 4068, which is again lower than the ridge regression.

When we compare the coefficient vector β of predictors, we see that the most influencing predictor is the number of reviews in all of the regularized regressions. The regression results can be found in the appendix, we opted to not include them here as they are quite similar. If we have only one significant predictor, lasso is expected to be better. This explains the lower RMSE. In the appendix, we provide the graph of coefficients with different penalty parameters. In line with our results, it is clearly seen that we have only one significant predictor.

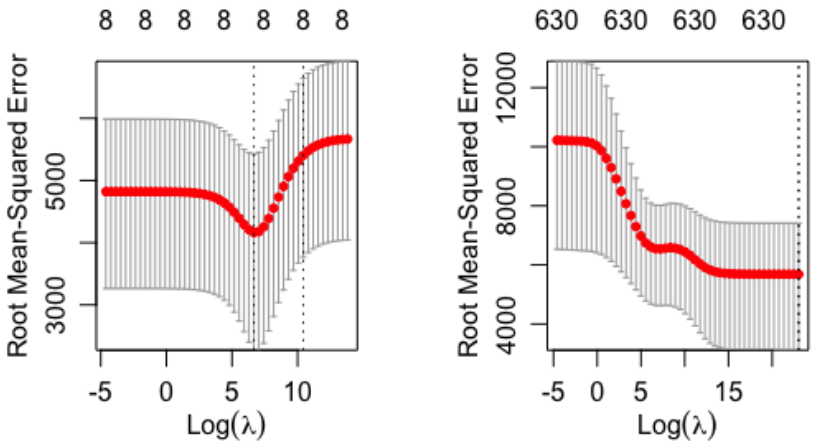

Figure 1: Left-side:RSME for different values of lambda for Ridge Regression
Right-side:RSME for different values of lambda for polynomial Ridge Regression

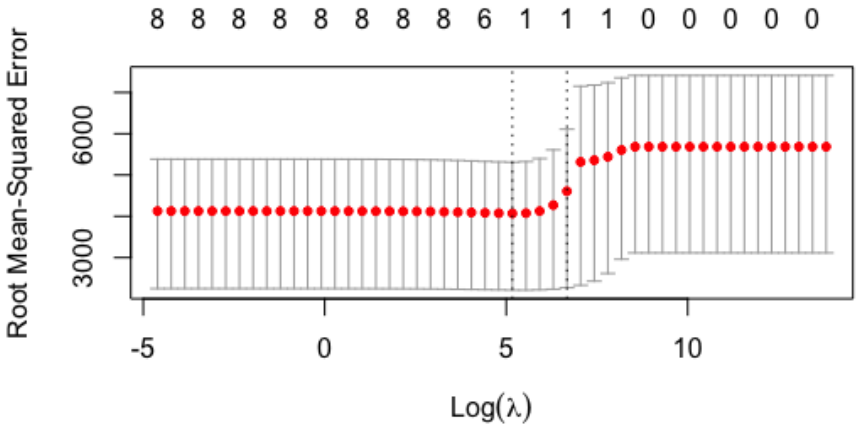

Figure 2: RSME for different values of lambda for Lasso Regression

Next, we implement bagging and random forest. For random forest, we decided to use only five predictors for training each tree as this specification had the lowest RMSE. The errors with respect to number of trained trees can be seen in the figure 3. From the figures, we see that both methods require approximately 80 trees to minimize out-of-bag errors. The lowest RMSE we get from bagging is 4159 whereas for random forest it is 3998. This suggest that random forest is doing a slightly better job in predicting compared to the rest of the techniques. The summary table for our results can be seen in table 1.

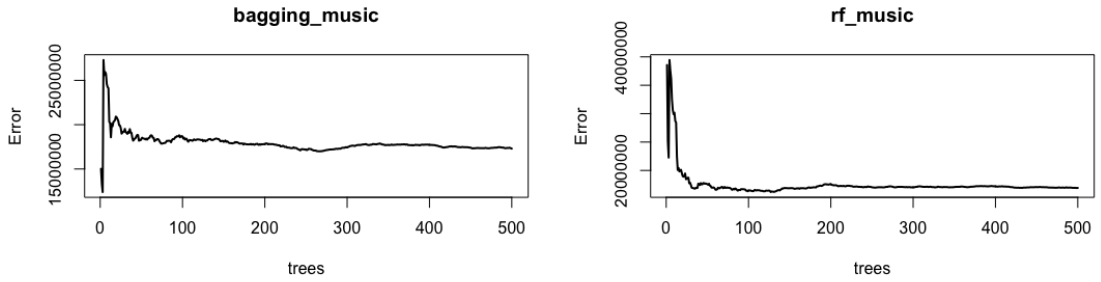

Figure 3: Left-side: OBB errors for number of generated trees for Bagging
Right-side: OBB errors for number of generated trees for Random forest with 5 features

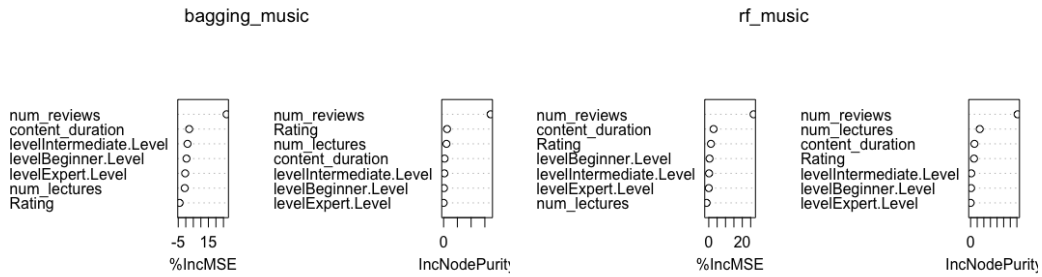

Figure 4: Left-side: Variable importance of Bagging
Right-side: Variable importance of Random forest with 4 features

	Lasso	Ridge	PolyRidge	RF4	Bagging
RMSE	4068	4164	5445	3998	4159

Table 1: RMSE of Lasso, Ridge, Polynomial Ridge, Random Forest with 4 features and Bagging

Similar to what we have found with regularized regressions, we see that number of reviews is the most dominant feature in predicting the number of subscribers. The figures regarding the variable of importance for both bagging and random forest can be seen in figure 4. After that, we see that the content duration plays a bigger role.

All in all, we can argue that random forest with five features is doing the best job in predicting the number of subscribers given our predictors. However, the difference with the other techniques is quite small. This is mainly due to the fact that we have only one dominant predictor: number of reviews.

5 Conclusion

In this paper we analyzed which technique is doing a better job in predicting the number of subscribers of the Udemy's musical instrument courses given the data on price, content duration, level

of the course, rating, number of reviews and number of lectures. Firstly, we have shown evidence that the number of reviews is much more dominant than the other explanatory variables via our ridge and lasso results. Consequently, we have found that lasso is much better at predicting. When it comes to bagging and random forest, we showed that random forest with five features is doing the best job among the specifications we have considered. Even though interpretability of decision tree methods are questionable, we showed that the number of reviews is again playing the biggest role in lowering the RMSE. One limitation of our analysis is that we do not know whether this is a causal relationship or not. Since a popular course already has more subscribers, it is natural that it would have a higher number of reviews therefore the result is intuition-wise trivial. However, we think that this analysis is still important as it can give the preliminary information on how to design a popular course.

6 Appendix

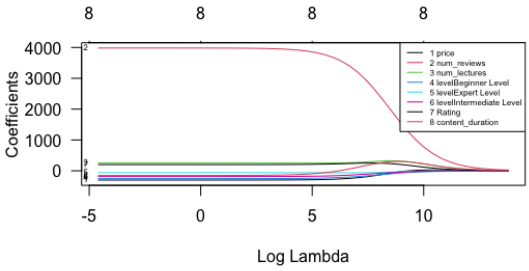

Figure 5: Coefficients with different penalty parameters in Ridge regression

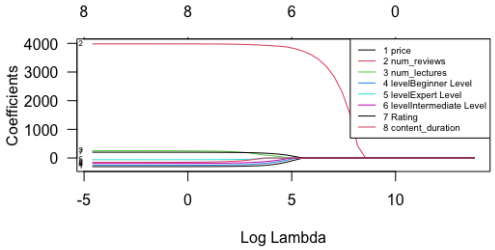

Figure 6: Coefficients with different penalty parameters in Lasso regression

```
> print(ridge_lassolambda$beta)
8 x 1 sparse Matrix of class "dgCMatrix"
                              s0
price                   -288.40
num_reviews             3834.09
num_lectures             252.29
levelBeginner Level     -239.80
levelExpert Level        -62.31
levelIntermediate Level -174.57
Rating                   211.93
content_duration         -89.06
> print(finalresult_ridgepoly$beta)
8 x 1 sparse Matrix of class "dgCMatrix"
                               s0
price                    -210.0
num_reviews             3072.1
num_lectures             281.6
levelBeginner Level     -176.3
levelExpert Level        -64.1
levelIntermediate Level -118.4
Rating                   247.5
content_duration         144.4
> print(finalresult_ridge$beta)
8 x 1 sparse Matrix of class "dgCMatrix"
                              s0
price                   -246.67
num_reviews             3401.33
num_lectures             263.56
levelBeginner Level     -202.87
levelExpert Level        -65.59
levelIntermediate Level -141.25
Rating                   236.44
content_duration          59.54
>
```

Figure 7: Coefficients of regularized regressions

References

Ho, T. K. (1995). Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, Volume 1, pp. 278–282. IEEE.

Hoerl, A. E. (1962). Applications of ridge analysis toregression problems. *Chemical Engineering Progress 58*, 54–59.

Hoerl, A. E. and R. W. Kennard (1968). On regression analysis and biased estimation. *Technometrics 10*, 422–423.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological) 58*(1), 267–288.

Willden, C. (2019). Udemy courses dataset. *Concept Center*.

7 Code

7.1 Code

Listing 1: SML code for Final Assignment

```
### file path etc
#rm(list=ls())
options(scipen=6, digits=4)
##packages anf libraries
if (!require("pacman")) install.packages("pacman")
pacman::p_load(ggplot2, tidyverse, tinytex, rmarkdown, glmnet, matlib, MASS, pdist, dplyr,
    plotrix, kernlab, ranger, randomForest, )
library(matlib)
library(glmnet, quietly = TRUE)
library(caTools)

##Data
music<-read.csv("/Users/ekuleru/Desktop/TI Second year/Supervised Machine Learning/Final
    Assignment/udemy course dataset/Udemymusic.csv")
df<- subset(music, select = -c(url, course_id, course_title, published_timestamp, subject)
    )
music<-df[,c(2,1,3,4,5,6,7)]
#Check whether we have omitted variables
sum(complete.cases(music))
summary(music)
#Organising and scaling
y<- as.vector(music$num_subscriber)
X <- model.matrix(num_subscribers ~ . , data = music)
X[, 2:9] <- scale(X[, 2:9])
X <- X[, -1]
set.seed(8913)

##Ridge Regression
result_ridge <- glmnet(X, y, alpha = 0, lambda = 10^seq(-2, 6, length.out = 50),
                  standardize = FALSE)
plot(result_ridge, xvar = "lambda", label = TRUE, las = 1)
legend("topright", lwd = 1, col = 1:6, bg = "white", legend = pasteCols(t(cbind(1:ncol(X),
    " ",colnames(X)))), cex = .5)
result_ridge.cv<- cv.glmnet(X, y, alpha = 0, lambda = 10^seq(-2, 6, length.out = 50),
                       standardize = FALSE)
print(result_ridge.cv$lambda.min)# Best cross validated lambda
##Min lambda is 790.6

## To plot Root Mean Squared Error (RMSE) to be on the same scale as y:
result_ridge.cv$cvm <- result_ridge.cv$cvm^0.5
result_ridge.cv$cvup <- result_ridge.cv$cvup^0.5
result_ridge.cv$cvlo <- result_ridge.cv$cvlo^0.5
plot(result_ridge.cv, ylab = "Root Mean-Squared Error")
#lowest RMSE= 4164
#Results with the best lambda
finalresult_ridge <- glmnet(X, y, alpha = 0, lambda = result_ridge.cv$lambda.min,
                   intercept = TRUE)
print(finalresult_ridge$beta)

#however, the umber of parameters is very small compared to the number of observations.
#that's why, i also tried t replace each predictor by its polynomial basis and model
    interaction

deg<-7
X.poly.ridge <- model.matrix(~ 0 + poly(X[, 1], degree = deg)
                        + poly(X[, 2], degree = deg)
                        + poly(X[, 3], degree = deg)
                        + poly(X[, 7], degree = deg)
                        + poly(X[, 8], degree = deg), data = as.data.frame(X[, 2:8]))
```

```
X.inter.action <- model.matrix( ~ .^2, data = as.data.frame(X.poly.ridge))
dim(X.inter.action)
#I dont know why i could not use 4th,5th and 6th column. I guess that since they are dummy variables there
# don't have enough unique points and it creates problem when we try to create higher degree polynomial

result_ridgepoly.cv <- cv.glmnet(X.inter.action, y, alpha = 0,
                        lambda = 10^seq(-2, 10, length.out = 50), nfolds = 10)
print(result_ridgepoly.cv$lambda.min) # Best cross validated lambda
##Min lambda is 1389

print(result_ridgepoly.cv$lambda.1se) # Conservative est. of best lambda (1 stdev)

#RSME for ridge polynomial
plot(result_ridgepoly.cv$lambda, result_ridgepoly.cv$cvm^.5, log = "x", col = "red", type = "p", pch = 20,
        xlab = expression(lambda), ylab = "RMSE", las = 1)
#another RSME graph
result_ridgepoly.cv$cvm <- result_ridgepoly.cv$cvm^0.5
result_ridgepoly.cv$cvup <- result_ridgepoly.cv$cvup^0.5
result_ridgepoly.cv$cvlo <- result_ridgepoly.cv$cvlo^0.5
plot(result_ridgepoly.cv, ylab = "Root Mean-Squared Error")
#lowest RMSE we get is = 5445
#Results with the best lambda
finalresult_ridgepoly <- glmnet(X, y, alpha = 0, lambda = result_ridgepoly.cv$lambda.min,
                            intercept = TRUE)
print(finalresult_ridgepoly$beta)

## Lasso Regression

result_lasso <- glmnet(X, y, alpha = 1, lambda = 10^seq(-2, 6, length.out = 50),
                       standardize = FALSE)
plot(result_lasso, xvar = "lambda", label = TRUE, las = 1)
legend("topright", lwd = 1, col = 1:6, bg = "white", legend = pasteCols(t(cbind(1:ncol(X), " ",colnames(X)))), cex = .6)
result_lasso.cv<- cv.glmnet(X, y, alpha = 1, lambda = 10^seq(-2, 6, length.out = 50),
                            standardize = FALSE)
#RSME for Lasso = 4068

result_lasso.cv$cvm <- result_lasso.cv$cvm^0.5
result_lasso.cv$cvup <- result_lasso.cv$cvup^0.5
result_lasso.cv$cvlo <- result_lasso.cv$cvlo^0.5
plot(result_lasso.cv, ylab = "Root Mean-Squared Error")

##Lasso does not want to penalize much meaning that there are not many high influence features

print(result_lasso.cv$lambda.min)# Best cross validated lambda
##Min lambda is 175.8
#Results with the best lambda
finalresult.lasso <- glmnet(X, y, alpha = 1,
                            lambda = result_lasso.cv$lambda.min)
print(finalresult.lasso$beta)

#use same lambda for comparison
finalresult.lasso_lambda <- glmnet(X, y, alpha = 1,
                            lambda = result_ridgepoly.cv$lambda.min)
print(finalresult.lasso_lambda$beta)

finalresult.lasso_lambda2 <- glmnet(X, y, alpha = 1,
                                    lambda = result_ridge.cv$lambda.min)
print(finalresult.lasso_lambda2$beta)

ridge_lassolambda<-glmnet(X, y, alpha = 0, lambda = result_lasso.cv$lambda.min,
          standardize = FALSE)
```

```r
print(ridge_lassolambda$beta)
#We dont see a difference when we switch lambdas


## Random Forest
#Organizing the data
bag_data<-as.data.frame(X[, 2:8])
sort(unique(colnames(bag_data)))
names(bag_data) <- make.names(names(bag_data))

#Bagging implementation
bagging_music <- randomForest(y ~ ., data = bag_data, mtry = 7, ntree = 500, importance =
    TRUE, do.trace = TRUE)
bagging_music
sqrt(bagging_music$mse[length(bagging_music$mse)])
#RMSE=4159

#Random forest implementation
plot(bagging_music, lwd = 2)
varImpPlot(bagging_music)
# it seems that number of reviews is the most important indicator. we do not know the
    causality though
rf_music <- randomForest(y ~ ., data = bag_data, mtry = 4, ntree = 500, importance = TRUE,
     do.trace = TRUE) #look at 4 features
rf_music
plot(rf_music, lwd = 2)
varImpPlot(rf_music)
randomForest::importance(rf_music)
sqrt(rf_music$mse[length(rf_music$mse)])
#RMSE=4114

rf_music2 <- randomForest(y ~ ., data = bag_data, mtry = 2, ntree = 500, importance = TRUE
    , do.trace = TRUE)#look at 2 features
rf_music2
plot(rf_music2 , lwd = 2)
varImpPlot(rf_music2 )
sqrt(rf_music2 $mse[length(rf_music2 $mse)])
#RMSE=4150

#Also apply permutation just for the curiosity
music_bag_ranger <- ranger(y ~ ., data = bag_data, mtry = 4,
                           num.trees = 500, importance = "permutation")
music_bag_ranger

#higher r^2 in bagging-->intuition: no overfitting as we increase number of features bc
    only few are significant

#Prediction (something went wrong here)

#Create training and test set before standardization
split = sample.split(music$num_subscribers, SplitRatio = 0.75)
training_set = subset(music, split == TRUE)
test_set = subset(music, split == FALSE)
#standardize for comparison with previous sections
y<- as.vector(training_set$num_subscriber)
X_train <- model.matrix(num_subscribers ~ . , data = training_set)
X_train[, 2:9] <- scale(X_train[, 2:9])
X_train <- X_train[, -1]

X_test <- model.matrix(num_subscribers ~ . , data = test_set)
X_test[, 2:9] <- scale(X_test[, 2:9])
X_test <- X_test[, -1]

bag_testdata<-as.data.frame(X_test[, 2:8])
sort(unique(colnames(bag_testdata)))
names(bag_testdata) <- make.names(names(bag_testdata))
```

```
177
178 y_pred = predict(bagging_music, newdata = bag_testdata)
179 testing<- as.numeric(scale(test_set[, 1]))
180
181 sqrt(mean((testing - y_pred)^2)) #Something went wrong
182
183 cm = table(test_set[, 1], y_pred)
184 cm
185 #I guess it is not meaningful to construct this table for non-dummy outcome variables
```