Communications for Statistical Applications and Methods
2021, Vol. 28, No. 3, 217–232
https://doi.org/10.29220/CSAM.2021.28.3.217
Print ISSN 2287-7843 / Online ISSN 2383-4757

Predicting movie audience with stacked generalization by combining machine learning algorithms

Junghoon Park[a], Changwon Lim[1,a]

[a]Department of Applied Statistics, Chung-Ang University, Korea

Abstract

The Korea film industry has matured and the number of movie-watching per capita has reached the highest level in the world. Since then, movie industry growth rate is decreasing and even the total sales of movies per year slightly decreased in 2018. The number of moviegoers is the first factor of sales in movie industry and also an important factor influencing additional sales. Thus it is important to predict the number of movie audiences. In this study, we predict the cumulative number of audiences of films using stacking, an ensemble method. Stacking is a kind of ensemble method that combines all the algorithms used in the prediction. We use box office data from Korea Film Council and web comment data from Daum Movie (www.movie.daum.net). This paper describes the process of collecting and preprocessing of explanatory variables and explains regression models used in stacking. Final stacking model outperforms in the prediction of test set in terms of RMSE.

Keywords: stacking, stacked generalization, ensemble, machine learning, data mining, movie audience prediction

1. Introduction

According to the 2018 Korean Film Council (KOFIC) closing report, the Korean domestic film industry grew steadily before 2013. In addition, the number of audiences has remained at about the same level for five years after exceeding 210 million reaching the world's highest level in the number of movie-watching per capita (Korean Film Council, 2019). However, as the film industry matures, the growth rate is decreasing and the total number of movie audiences per year even slightly decreased in 2018. The profit structure of the Korean film industry is also showing signs of change. The number of movies with total product cost more than 8 billion Korean Won has increased, while the estimated returns are reduced. In contrast, medium and low budget films of 3 billion to 5 billion Korean Won are showing an increase in net profits (Korean Film Council, 2019).

Movie's box office is the primary factor of movie sales and has a huge influence on additional sales. Therefore it is very important to predict the box office of films at the time of film industrial change. However, judging whether a movie is a hit or not is very complicated because the aspect of attraction of audiences changes due to a number of factors even after release. For this reason, many studies have been conducted to predict the success of movies. Yu and Lee (2018) hypothesized the factors influencing the breakthrough of 10 million audiences and conducted a study on the relationship between the hypothesized factors and the breakthrough of 10 million viewers. After dividing into 4 groups according to the hypothesis set by the researchers, they used logistic regression and neural

[1] Corresponding author: Department of Applied Statistics, 84, Heukseok-ro, Dongjak-Gu, Seoul 06974, Korea.
E-mail: clim@cau.ac.kr

Published 31 May 2021 / journal homepage: http://csam.or.kr

network algorithms for each hypothesized group to classify that movies would or not have more than 10 million audiences. Lee *et al.* (2018) conducted a study to predict the number of moviegoers by selecting variables based on the information gain rate. The multivariate regression model was used to compare before and after variable selection in terms of R^2, MAPE, etc, and the results of multivariate linear regression were confirmed to be improved by selecting variables based on information gain rates. Lee (2018) predicted the number of audiences in the first week of the released movies and the success of the box office by three stages through machine learning techniques. The machine learning techniques used for the first-week audience prediction were support vector machine (SVM) and neural network, having an accuracy of 76.61% and 70.16%, respectively. After selecting variables using the decision tree model called rpart, the first-week audience prediction was improved to 79.84% for SVM and to 79.84% for neural network.

In this paper, we consider a stacking model for predicting movie audiences. Stacking is an ensemble method that consists of two steps, Level 0 and Level 1. The first step is called Level 0, and any algorithms for regression or classification can make up Level 0 models. Fine-tuned Level 0 models are used to make predictions for the target variable. The second step, called Level 1, ensembles algorithms of Level 0 by taking predictions of Level 0 models as independent variables to predict the target variable. This results in reducing the biases of predicted values from Level 0. The model which ensembles the Level 0 predictions best is selected as the Level 1 model. We train 9 models for Level 0 and experiment 5 models for Level 1 after collecting and preprocessing independent variables which are known to be important for predicting movie audiences. The final stacking model outperforms in predicting the cumulative number of audiences of the test dataset.

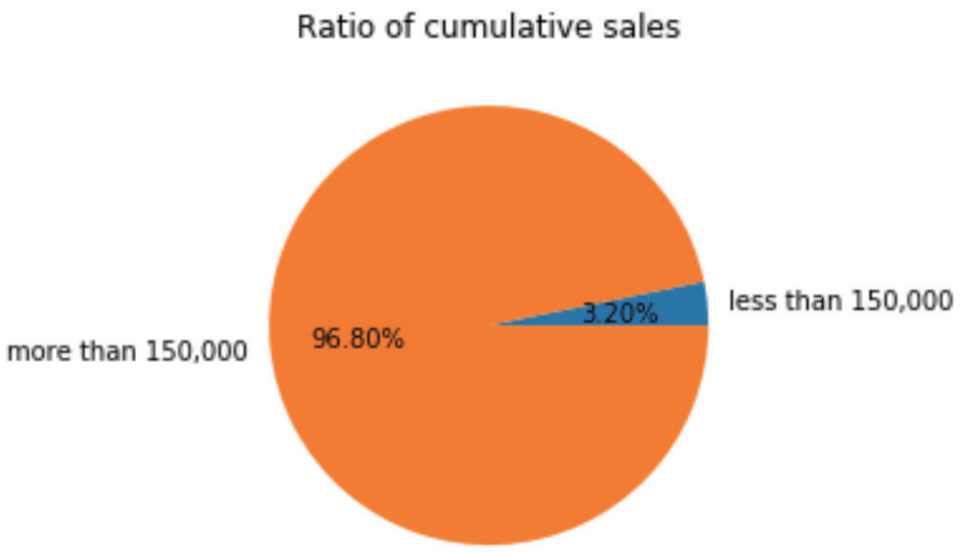
This paper is organized into 5 sections. Section 2 describes the detailed process of collecting and preprocessing explanatory variables with some graphs. Section 3 explains the process of stacking and fitting with the hyperparameter tuning of regression models at Level 0 and its results. Section 4 explains Level 1 models with hyperparameter tuning, variable selections, and provides the predictions of the test dataset by the final stacking model. Section 5 describes the conclusion and some future challenges.

2. Data description

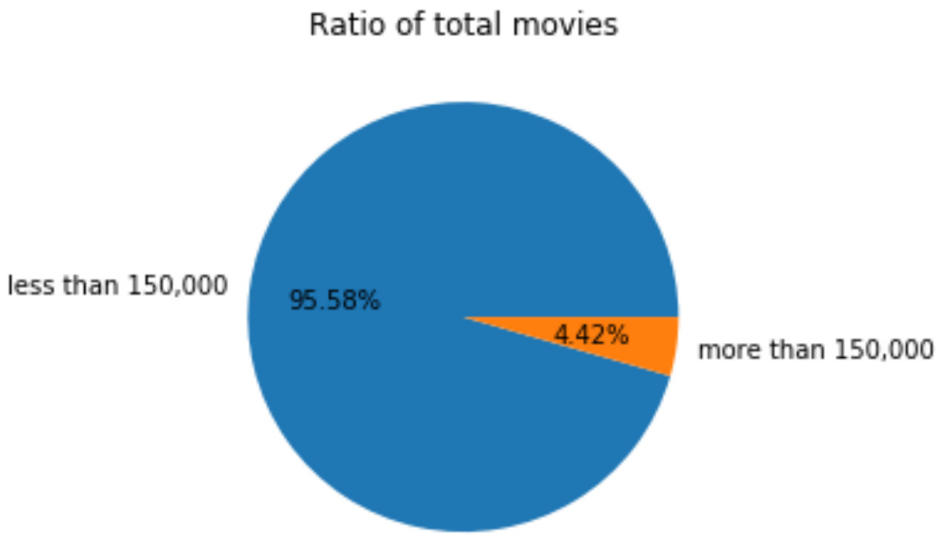
2.1. Collecting data

The data considered in this paper daily box office data of the Korea Film Council and movie comment data of Daum Movie (www.movie.daum.net). The films considered were released between July 2017 and June 2019. Box office data includes the film's general information, including nationality, director, actors, genre, release date, cumulative audiences, and distributors. The movie comment data are collected one week from the release date of each movie. The title of the movie, the date of the comments, the number of comments per day, and ratings of comments are collected by crawling using Python API Selenium and Beautiful Soup (Lawson, 2015). The comment data, like for the daily box office data, are collected about movies released between July 2017 and June 2019.

In addition, we aim at commercial films having cumulative audiences of 150,000 or more. Movies with cumulative audiences of 150,000 or less account for only 3.2% of the total cumulative sales of movies released in two years between September 2017 and June 2019 as shown in Figure 1. Besides the movies with less than 150,000 audiences were 95.58% between July 2017 and June 2019, those movies's sales were only 3.2% in same period. For those reasons, we predicted movie audiences with over 150,000, which takes the most charges in movie industry. Therefore, 222 films are finally used in this study. All data are collected and preprocessed by R (R Core Team, 2019) and Python (Van



(a) Ratio for cumulative sales



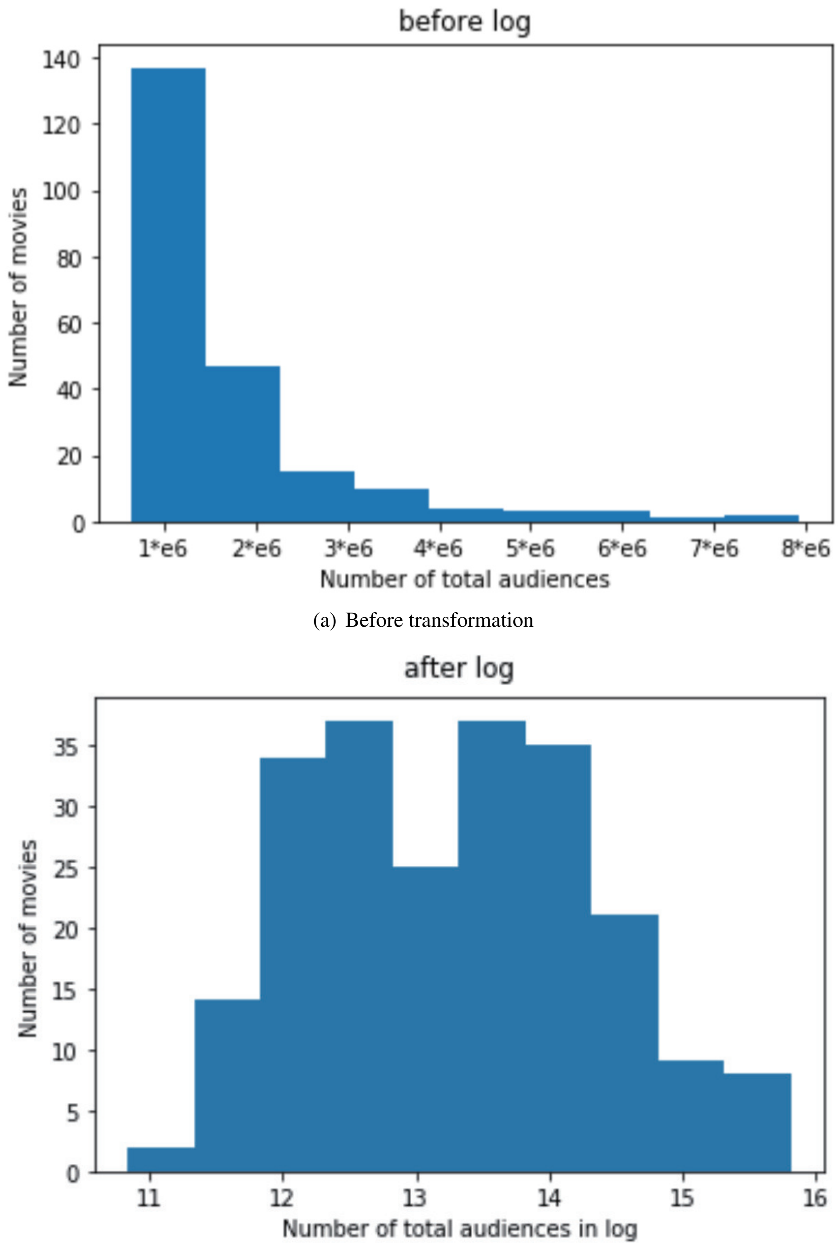
(b) Ratio for frequency of movies

Figure 1: *Pie chart comparing movies over 150,000 audiences and not.*

Rossum and Drake, 2009).

2.2. Independent variables

The prediction of the number of movie audiences should be based on independent variables that play an important role in the success of films. The collected independent variables are nationality,



(a) Before transformation

(b) After transformations

Figure 2: *Before and after log transformation of movie audiences.*

grade, the release date, the release day of the week, the average number of audiences by distributors, a cluster of distributors, the average number of audiences by directors, a cluster of directors, the average number of audiences by genres, a cluster of genres, weekly rating by web commenters, the

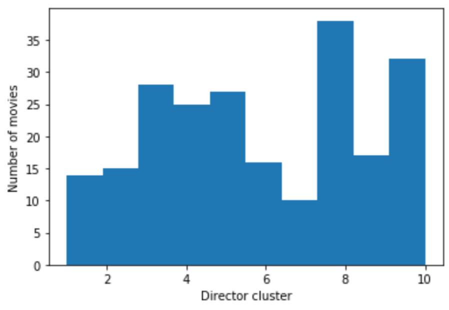


Figure 3: *Histogram of director clusters by k-means clustering.*

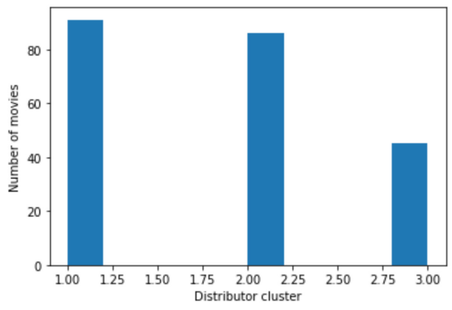


Figure 4: *Histogram of distributor clusters by k-means clustering.*

weekly number of comments, and a cumulative number of audiences of the first week after release. Among them, the average number of audiences for distributors, the average number of audiences of last 5 movies for directors, the average number of audiences for genres, the number of audiences during the first week, and the final number of total audiences are taken log transformation in order to stabilize their distributions. Histograms of a cumulative number of audiences before and after log transformation are shown in Figure 2 as examples. The detailed description of the independent variables is as follows.

2.2.1. Distributor, director and genre

When there are two or more values in the genre, distributor and director variables for each film in the box office data of Korea Film Council, the first value is set as the representative value. Then distributor, director and genre variables are reduced to clusters, respectively, through k-means clustering based on elbow-plot. As a result, the distributor, director and genre variables have three, ten and six groups, respectively, and the results of the k-means clustering for directors and distributors are shown in Figures 3 and 4. In addition, the average number of audiences by distributors, the average number of audiences by directors, and the average number of audiences by genres are added as independent variables to prevent information loss.

2.2.2. Nationality

In the case of multiple values in the nationality variable, as in the case of distributor and director, the first value is set as the representative value based on the film registration information in the Korea Film Council. Korea, which has the highest frequency, is assigned to group 1, the United States to group 2, and the others to group 3.

2.2.3. Release day of the week and month

It is known that there is a big difference in the success of films according to the day of the week and month of the release. Also, most films are known to be released mainly on Tuesday, Wednesday and Thursday in order to maximize the audience attraction of the first weekend (Kim *et al.*, 2010). Therefore, the movies that are released on Monday, Tuesday, Wednesday, Thursday and Friday are assigned to 1, 2, 3, 4, 5 respectively as dummy variables.

The variable about the month of release is set by the fact that the average number of audiences by month varies due to the characteristics of the movie market. Assuming that weather would affect the sales of movies, variable of released month was added based on the average of total audiences

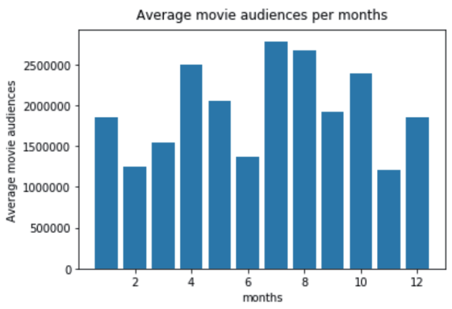


Figure 5: *Bar chart of average number of audiences by month.*

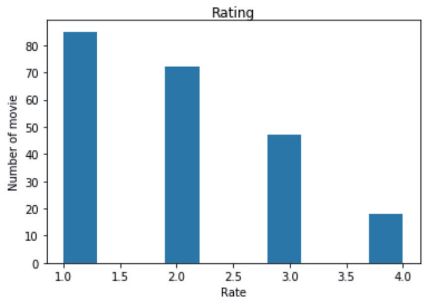


Figure 6: *Histogram of movie rating.*

per month. In Figure 5, average of audiences per movies differs from each moths. Therefore, movies released in July and August are assigned to be into group 1 representing the peak season, January, April, May, October and December into group 2 representing the middle sales season, and the remaining February, March, June, September and November are grouped into group 3 for the lower sale season.

2.2.4. Movie rating

In terms of the movie rating, the 15+ rated movies are assigned to be in group 1, 12+ rated movies in group 2, the all rated movies in group 3, and the 19+ rated movies in group 4 by their frequencies as in Figure 6.

2.2.5. Cumulative number of audiences, number of comments, average score of comments during first week

It is known that the success of a movie is highly correlated to the success of the movie during the first week and as well as by the word of mouth (Park, 2012). Therefore, in order to improve the accuracy of prediction of the cumulative number of audiences of a film, the cumulative number of audiences during the first seven days after the release is collected as an independent variable. And for measuring the effect by the word of mouth, the crawling method is used for the variable representing how many people mention the movie on the Internet during the first 7 days as comments containing rating of the movie in Daum Movie (www.movie.daum.net) using the Python API, BeautifulSoup and Selenium (Lawson, 2015). In conclusion, after the movie is released, the number and average rating score of comments for 7 days of each movie are collected.

3. Models

3.1. Stacking

Stacking (also called stacked generalization, will be used interchangeably) was first proposed in 1992 by David H. Wolpert on the journal *Neural Network*. By going beyond selecting the best one among multiple algorithms that are trained to solve the same problem, stacking ensembles all the trained algorithms. The stacked generalization consists of two steps: Level 0 and Level 1. Any algorithm to solve the same problem can be a candidate that makes up Level 0. Level 0 algorithms are trained to predict the target variable. Level 1 model takes predictions from Level 0 models as input features and is trained to predict the target variable. This task means that the Level 1 model ensembles Level 0 models by reducing biases of predictions from Level 0 models.

It is known that the fine hyperparameter tuning of the Level 1 model results in higher performance than just selecting the single best model. However, the best ensemble of the predictions from the Level 0 models is somewhat unclear because of the fact that the stacking model structure is so complicated. Therefore, it is important to carefully and accurately adjust the hyperparameter tuning of each level.

Additionally, one of the most important things to note about stacking is to prevent overfitting (Wolpert, 1992). If Level 0 models are fitted, then the Level 1 model should not be trained using the training dataset used for Level 0 models. Otherwise, the Level 1 models likely result in overfitting. In order to prevent overfitting in Level 1 step, the training dataset must be split into Level 0 and Level 1 datasets and should not be shuffled. Predicted values from Level 0 models are ensembled by Level 1 models and the final stacking model takes the best model. In this paper, 30% of the total 222 movies (= 67) are the test dataset, 20% among the remaining movies (= 31) are Level 1 training dataset, and the others (= 124) are Level 0 training dataset.

Assume that f_{ij} is j^{th} algorithm for i^{th} level of stacking modeling for $i = 1$ (Level 0), 2 (Level 1), $j = 1, \ldots, J$. Then the prediction of the dependent variable from the j^{th} level 0 model, $\hat{y}_{0j}$, can be expressed as

$$\hat{y}_{0j} = f_{1j}(X_{\text{Lv0}}), \quad j = 1, 2, \ldots, J, \tag{3.1}$$

where X_{Lv0} is the independent variable of Level 0 train set. In the course of learning the level 0 model, the hyperparameters are tuned using randomized search with 5-fold cross validation. Randomized search, unlike Grid search, fits the model by randomly selecting a combination of hyperparameters with random sampling without replacement among all combinations within the hyperparameter range which are determined by the researcher. Then, the best performing combination of hyperparameters is selected based on the Root Mean Square Error (RMSE) through 5-fold cross validation as described by

$$\text{RMSE}_i = \sqrt{\frac{1}{n_i} \sum_{l=1}^{n_i} \left(\hat{y}_{0j,i} - y_{\text{Lv0},i}\right)^2}, \quad i = 1, \ldots, 5, \tag{3.2}$$

where n_i is the number of samples in the i^{th} fold and $\hat{y}_{0j,i}$ is the predicted value of $y_{\text{Lv0},i}$ using the j^{th} Level 0 model. The final RMSE is the average of all RMSE_i of 5 folds.

While keeping in mind that model index in Level 0 is 1 to J, we make X_{md} called meta data which is the data space for the Level 1 model whose total shape is the size of Level 1 train set times J. The trained Level 0 models are then used to generate predictions $\hat{y}_{1j}$ to make up the X_{md} of the target values of the Level 1 train set as follows:

$$\hat{y}_{1j} = f_{1j}(X_{\text{Lv1}}), \quad j = 1, \ldots, J, \tag{3.3}$$

where X_{Lv1} is independent variables of Level 1 train set. These predictions $\hat{y}_{1j}$ make up the X_{md}, that is, X_{md} consists of predicted values of Level 1 train dataset by J models from Level 0 and then Level 1 model ensembles the predicted values by fitting on X_{md} set.

When training Level 1 models, we need to find an algorithm that can ensemble this X_{md} best. However, since the data set X_{md} for meta learner consists of predictions produced by the Level 0 models to solve the same problem, the correlations between predictions are naturally high. Therefore, variable selection is added in Level 1 step. After selecting some algorithms as candaidates for Level 1 model, the process of selecting the best performing hyperparameter of Level 1 model through randomized search with 5-fold cross validation as done for Level 0 models. In addition, we need to find the best combination of J variables. There are $2^J - 1$ subsets of variables in X_{md}. Let $X_{\mathrm{md},s}$ denote the s^{th} subset in data space for Level 1 models. Then, the prediction obtained by j^{th} Level 1 model based on $X_{\mathrm{md},s}$ can be expressed by

$$\hat{y}_{2js} = f_{2j}(X_{\mathrm{md},s}), \quad s = 1, 2, \ldots, 2^J - 1, \tag{3.4}$$

and the RMSE in first fold is calculated by

$$\mathrm{RMSE}_1 = \sqrt{\frac{1}{m_1}\sum_{k=1}^{m_1}\left(\hat{y}_{2js,k} - y_{\mathrm{Lv1},k}\right)^2}, \tag{3.5}$$

where m_1 is the number of samples in first fold and the final RMSE is the average of all RMSE_i of 5 folds.

We train several regression models for Level 0 and Level 1, and select one as the final meta learner among Level 1 models which has the best subset of variables $X_{\mathrm{md},s}$ and the hyperparamaters with the smallest RMSE. In summary, we perform exhaustive search considering all $(2^J - 1)$ variable combinations while tuning the optimal hyperparameters for each variable combination by randomized search with 5-fold cross validation. As a result, the model combination of Level 0 models and its meta learner with optimal hyperparameters are selected based on the criterion of RMSE with 5-fold cross validation.

Now, we will briefly describe the nine models for Level 0 and the five models for Level 1, and show the process of hyperparameter tuning and variable selection. The nine models used in the Level 0 are from the field of four most common themes used in the regression problem: Regression models with penalty term (Ridge, Lasso, and Elastic Net), distance based algorithms (KNN regression and Support Vector Regression (SVR)), tree based ensemble models (Extra Tree and Random Forest), and boosting methods (AdaBoost.R2 and Gradient Boosting). In Level 1, Ridge, Random Forest, AdaBoost.R2, Gradient Boosting and Artificial Neural Network (ANN) are used. Then the final stacking model is obtained after all the hyperparameter tuning and variable selection by the criteria of 5-fold cross validation RMSE.

3.2. Regression models with penalty term: ridge, lasso, and elastic net

A ridge regression estimator is biased unlike the traditional Ordinary Least Squares (OLS) method. The vector of regression coefficients β is estimated by minimizing the error sum of squares for OLS, but the ridge regression uses L2 norm to add regulation on coefficients as follows:

$$\hat{\beta}_{\mathrm{Ridge}} = \arg\min_{\beta}\left(\left\|Y - X^T\beta\right\|^2 + \lambda\|\beta\|_2^2\right), \tag{3.6}$$

where λ is a non-negative penalty parameter and if it is zero, the ridge regression estimator becomes the OLS estimator. Unlike OLS, it allows some bias, but shows much better model performance than OLS by greatly reducing variance in MSE bias-variance trade off. Ridge regression is also known as a regression analysis method that can solve multicollinearity. In this study, randomized search with 5-fold cross validation is performed by dividing into 1,000 points from 0.001 to 1 to obtain an optimal value of the hyperparameter λ.

Lasso regression is similar but somewhat different. Lasso regression is an abbreviation for least absolute shrinkage and selection operator, and the coefficient of the variable is obtained by adding the

L1 norm penalty, unlike the ridge regression, as follows:

$$\hat{\beta}_{\text{Lasso}} = \arg\min_{\beta}\left(\left\|Y - X^T\beta\right\|^2 + \lambda\|\beta\|_1\right). \tag{3.7}$$

There is also the Elastic Net which combines the ridge and the lasso regression methods.

$$\hat{\beta}_{\text{ElasticNet}} = \arg\min_{\beta}\left(\left\|Y - X^T\beta\right\|^2 + \lambda_1\|\beta\|_1 + \lambda_2\|\beta\|_2^2\right), \tag{3.8}$$

where λ_1 and λ_2 are the penalty parameters. The coefficient of Elastic Net obtained through the equation (3.8) includes the constraints of Ridge and Lasso (Zou *et al.*, 2005). As with Ridge and Lasso regression, hyperparameter tuning is performed with randomized search with 5-fold cross validation. Python libraries sklearn.linear_model.Ridge, sklearn.linear _model.Lasso and sklearn.linear _model.ElasticNet are used for Ridge, Lasso, and Elastic Net, respectively.

3.3. Distance based algorithms: KNN regression and support vector regression

KNN is one of a kind of supervised learning and is very intuitive and an easy algorithm for regression and classification. KNN calculates the predicted value according to the similarity of k most similar data points rather than building a model and so is classified as lazy learning (Song *et al.*, 2017). The measure of similarity is calculated from the distance between the data points and this fact makes the KNN regression to be the simplest form in machine learning field. Euclidean and manhattan are popularly used as distance criteria, and the nearest k data points are found through these distance measures. In case of regression, the average of the k nearest data points's target values is used and in the case of classification, predicted value is just the majority vote of the k nearest data points of the input. For the data used in this study, to determine the optimal k, 20 KNN regressions are performed from 1 to 20, and the optimal k is selected with the smallest 5-fold cross validation RMSE.

SVM, like KNN, is a type of supervised learning that can be used for regression and classification (Pontil *et al.*, 1998). The SVM starts by finding a hyper plane that can sort the data well. The optimal hyper plane is the line that maximizes the margin between each entity. Hyper plane in SVM has

$$f(x, a) = \sum_{sv}\left(y_i a_i K(x, x_i) + b\right), \tag{3.9}$$

where $K(\,)$ is a kernal, a and b are the parameters to be estimated, and sv are support vectors. The support vector represents the data points closest to this hyperplane and the process goes through to find the hyperplane that provides the largest distance between this data point and the hyperplane. In addition, the kernel function mapping input values into feature space is used to derive linear separation and that makes more easier to find a hyper plane of the data.

Applying the SVM with this feature to a regression problem, we consider non-negative slack variables ζ_i^+ and ζ_i^-, and perform the regression as the optimization problem of the following equation:

$$\text{minimize}_{w,b}\left(\frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{n}\left(\zeta_i^+ + \zeta_i^-\right)\right), \tag{3.10}$$

subject to $y_i - \omega x_i - b \le \epsilon + \zeta_i^+$, $\omega x_i + b - y_i \le \epsilon + \zeta_i^-$, where slack variables ζ_i^+ and ζ_i^- represent margins between each data point and hyper planes, ϵ represents how large the epsilon-tube would be, affecting training loss function by controlling the distance between hyper plane and actual value, and C is the penalty parameter. The dual form of this optimizer of the SVR is calculated using the Lagrange multiplier (Paniagua-Tineo *et al.*, 2011). In this study, we set $\epsilon = 0.1$, and the hyperparameter combination consists of the kernel $K(\ )$, C and γ. Note that γ is the coefficient of $K(\ )$ and controls how smooth the splines would be. For hyperparameter tuning, randomized search with 5-fold cross validation is performed. Python libraries sklearn.neighbors.KNeighborsRegressor and sklearn.svm.SVR are used for KNN regression and SVR, respectively.

3.4. Tree based ensemble models: random forest and extra tree

Both the Extra tree and the random forest are ensemble models derived from the tree method. Random Forest was first proposed by Breiman (1999) and is known to outperform decision trees. Random Forest is a way to ensemble K independent trees so needs K bootstrap resamples (Segal, 2004). For details, each tree is an unpruned tree and the best split is performed with only m variables out of a total of p variables randomly selected for each node of the trees. In the case of the regression problem, the average is calculated in K trees for the final estimation, and the majority vote is calculated in the K trees in the classification problem (Liaw and Wiener, 2002). Error also can be estimated through out-of-bag errors.

The Extra Tree is an extremely randomized tree that ensemble unpruned regression or decision trees. It is similar to Random Forest that it extracts K features randomly to split at each node in Regression Tree, except that each tree to be ensembled is trained by a complete learning sample rather than bootstrap resmaple. More importantly, the cut-point of variables are set randomly rather than best split based on local sample (Geurts and Louppe, 2011). Therefore, the model changes only by the K value of the Extra Tree. It is known that setting K to $\sqrt{p}$ is usually good, and it is known that time consumption is smaller than that of Random Forest.

In this study, hyperparameter tuning is performed through randomized search with 5-fold cross validation using a combination of number of estimators, which is the number of trees to ensemble, and max of depth, the maximum size of tree. Python libraries sklearn.ensemble.ExtraTreesRegressor and sklearn.ensemble.RandomForestRegressor were used for Extra Tree and Random Forest, respectively.

3.5. Boosting algorithms for regression : AdaBoost.R2 and Gradient Boosting

AdaBoost.R2 (Solomatine and Shrestha, 2004) is a way to ensemble weak learners sequentially. It is an algorithm used for a regression problem and an ad hoc modified version of AdaBoost.R, and proceeds by updating the weight through an iteration to make the defined average loss function smaller. It is designed to basically ensemble the regression trees and has a learning process that increases the weight of poorly predicted regression trees and lowers the weight of well predicted regression trees. The average loss function for t^{th} iteration is

$$\overline{L}_t = \sum_{i=1}^{m} L_t(i)D_t(i), \tag{3.11}$$

where L_t is

$$\frac{l_t(i)}{\mathrm{Denom}_t}, \quad \left[\frac{l_t(i)}{\mathrm{Denom}_t}\right]^2, \quad 1 - \exp\left[\frac{l_t(i)}{\mathrm{Denom}_t}\right], \tag{3.12}$$

at linear, square law, exponential, respectively, $l_t = |f_t(x_i) - y_i|$, and $\mathrm{Denom}_t = \max(l_t(i))$, $i = 1, \ldots, m$.

Through $\overline{L}_t, \beta_t = \overline{L}_t/(1 - \overline{L}_t)$ is calculated, then the weight $D_t(i)$ is updated to

$$D_{t+1}(i) = \frac{D_t(i)\beta^{1-L_t(i)}}{Z_t}, \tag{3.13}$$

where Z_t is the normalized factor of D_t (Solomatine and Shrestha, 2004). The hyperparameters in AdaBoost.R2 are the learning rate and the total number of regression trees (Shrestha and Solomatine, 2006).

Gradient Boosting regression is a forward stage-wise fashion model, similar to AdaBoost.R2, and has the learning process to reduce the expected value of a given loss function. When the loss function is at the m^{th} iteration, the negative gradient is calculated as (Zhang and Haghani, 2015)

$$r_{im} = -\left[\frac{\partial L(y_i, F(X_i))}{\partial F(X_i)}\right]_{F(X)=F_{m-1}}, \tag{3.14}$$

where $L(\ )$ is the given loss function. The equation is updated with a base learner $h_m(x)$ that fits a pseud-residual, and model $F_m(x)$ is updated as $F_m(x) = F_{m-1}(x) + \zeta h_m(x)$ that reflects the appropriate learning rate ζ (Friedman, 2002).

The hyperparameters to tune at this time are the number of iterations, the size and learning rate of the regression tree, which is the base learner. The hyperparameter tuning is important for the prediction of unseen data because these three choices must be properly selected to avoid over-fitting the train set. In this study, Python is used, and the default weak learner is set to CART, which is the default for both the AdaBoost.R2 and Gradient Boosting methods. Python libraries sklearn.ensemble.AdaBoostRegressor and sklearn.ensemble.GradientBoostingRegressor are used for AdaBoost.R2 and gradient boosting, respectively.

3.6. Artificial neural network

ANN is a model conceived by human neurons and consists of input layer, hidden layer and output layer. Each layer is connected by weight, and receives the information of the previous layer and passes the output which went through the activation function to the next layer. In the learning process, the weights connecting the nodes of the layers are updated through iteration using backpropagation in a way to reduce the predefined loss function so that the output value approaches the target value. Artificail Neural Networks are widely used in various fields because it has a high explanatory power for nonlinear relationships between independent and dependent variables.

In this study, the number of hidden layers is fixed to 1, and the hyperparameter tuning of activation function, number of nodes, and optimizer for all $2^9 - 1$ variable combinations is performed to achieve the best subset of variables and its the best combinations of hyperparameters. To make the ANN model simple, we fixed only one hidden layer, remaining more complex architecture of ANN for further study. Library named Keras.Sequential was used for building ANN in this study. Likewise, other 4 Level 1 models were trained in process of variable selection among all $2^9 - 1$ variable subsets each with its hyperparameter tuning simultaneously.

4. Results

4.1. Result of final Level 0 models and product meta learner data space

The purpose of this study is to predict the cumulative audience of films at September first in 2019 which was released between July 2017 and June 2019. When predicting the audience of each movie,

Table 1: Results of Level 0 models

Level 0 model	Hyperparameter region	Best hyperparameter	Level 0 train RMSE
Ridge regression	alpha : (0.001~1), 1,000	alpha = 0.105	0.15465
Lasso regression	alpha : (0.001~1), 1,000	alpha = 0.02	0.16194

Elastic net	alpha : (0.001~1), 1,000
	l1_ratio : (0.001~1), 1,000
		alpha = 0.18
		l1_ratio = 0.728
			0.22114

KNN regression	k : (1~20), 20	k = 4	0.27092

SVR	kernel : (rbf, sigmoid, poly), 3
	C : (100~2,000), 200
	gamma : (1e-10~1e-1), 10
		kernel = sigmoid
		C = 1,100
		gamma = 0.01
			0.09567

Extra tree	n_estimators : (10~1,000), 991
	max_depth : (1~5), 5
		n_estimators = 153
		max_depth = 5
			0.09955

Random forest	n_estimators : (10~1,000), 991
	max_depth : (1 5), 5
	min_samples_split : (1~5), 5
		n_estimators = 72
		max_depth = 5
		min_samples_split = 2
			0.10078

AdaBoost.R2	learning_rate : (0.01~1), 100
	n_estimators : (10~1,000), 991
		learning_rate = 0.33
		n_estimators = 19
			0.14508

Gradient boosting	learning_rate : (0.01~1), 100
	n_estimators : (10~1,000), 991
	max_depth : (1~10), 10
		learning_rate = 0.12
		n_estimator = 537
		max_depth = 5
			0.0003122

Table 2: Results of level 1 models

Level 0 model	Hyperparameter region	Best hyperparameter	Feature subset	RMSE
Ridge regression	alpha : (0.001~1), 1000	alpha = 0.352	Gradient Boosting	0.009

Random forest	n_estimators : (10~1000), 991
	max_depth : (1~5), 5
	min_samples_split : (1~5), 5
		n_estimators = 136
		max_depth = 5
		min_samples_split = 2
			Elastic net,
			KNN regression,
			Random forest,
			Ridge, SVR
				0.01125

AdaBoost.R2	learning_rate : (0.01~1),100
	n_estimators : (10~1000), 991
		learning_rate = 0.7
		n_estimators = 96
			Lasso,
			Gradient Boosting
				0.04018

Gradient boosting	learning_rate : (0.01~1), 100
	n_estimators : (10~1000), 991
	max_depth : (1~10), 10
		learning_rate = 0.98
		n_estimators = 30
		max_depth = 5
			Elastic net, Lasso	0.00022

ANN	n_nodes : (1~20), 20
	Optimizers : (sgd,rmsprop,adam), 3
	Activation : (tanh,sigmoid,relu), 3
		n_nodes = 20
		Optimizer = adam
		Activation = relu
			Gradient boosting,
			Extra tree,
			SVR
				0.05278

cumulative audiences of movies were fixed at August first in 2019 because even if the movies were finished screening in big cities like Seoul, small theater in countryside still screens the movies and that audiences are also counted in KOFIC data. The result of Level 0 models is summarized in Table 1. All the hyperparameter notations of each model are following the expression in Python libraries and other following tables would do, too. And in hyperparameter region column, there are regions of hyperparameters and the number of counts of each hyperparameter and the selected hyperparameters are in best hyperparameter column.

For comparing all nine Level 0 models, the Gradient Boosting method performed the best. Also, SVR and Random Forest, which are known for good results, showed proper performance at this data. The lowest performing algorithm was Lasso regression, showing 0.16194 train RMSE. Comparing each theme, Ridge regression and Lasso regression resulted in train RMSE of about 0.15465 and 0.16194, respectively, and Elastic Net which combines Ridge and Lasso seems to perform poorer than Ridge and Lasso regression. Comparing KNN regression and SVR, which are algorithms based on the distance between data points, we can see that KNN which is a kind of lazy learner, yields a bigger train RMSE than SVR. And also, Extra Tree, a ensemble methods using complete learning

Table 3: Final stacking model

Level 0 model	Hyperparameter region	Level 1 model	Best hyperparameter	Feature selection	Test RMSE
Ridge regression	alpha = 0.105	Gradient boosting	learning_rate = 0.98 n_estimators = 30 max_depth = 5	Elastic net, Lasso	0.00074
Lasso regression	alpha = 0.02				
Elastic net	alpha = 0.018 l1_ratio = 0.728				
KNN regression	$k = 4$				
SVR	kernel = sigmoid C = 1100 gamma = 0.01				
Extra tree	n_estimators = 153 max_epth = 5				
Random forest	n_estimators = 72 min_samples_split = 2 max_depth = 5				
AdaBoost.R2	n_estimators = 19 learning_rate = 0.33				
Gradient boosting	learning_rate = 0.12 n_estimators = 537 max_depth = 5				

Table 4: Model comparison

Model step	Model name	Test set RMSE
Level 0	Ridge regression	0.39864
Level 0	Lasso regression	0.39367
Level 0	Elastic net	0.36080
Level 0	KNN regression	0.34334
Level 0	SVR	0.51540
Level 0	Extra tree	0.31316
Level 0	Random forest	0.35079
Level 0	AdaBoost.R2	0.38480
Level 0	Gradient boosting	0.18156
Level 1	**Stacking model**	**0.00074**

sample and random splitting with CART, showed better performance than Random Forest which is using bootstrap resampling and perfect splitting. Also, for estimating variance importance, the rate variable of each movie was the most important variable recording 91% of total importance. Gradient Boosting showed superior performance among boosting-using algorithm for regression comparing to AdaBoost.R2 in this data set.

4.2. Results of level 1 models

In Table 2, the results of 5 Level 1 models are shown. It can be seen that the performance of 5 Level 1 models are very powerful by the fact that combining the 9 model's results with variable selection and hyperparameter tuning. Ridge regression for highly correlated data shows remarkable performance in the sense of RMSE contrary to other regressors in Level 1 but the fact that just one input variable, which is Gradient Boosting output, is considerable in the sense of model architecture. And simillar results can be seen in Random Forest and AdaBoost.R2. They both showed impresive results with RMSE as 0.01125 and 0.04018 each. Also, in case of ANN, while the number of input nodes is selected to 3, which is a prediction of the Gradient Boosting method, Extra tree, and SVR, the number of hidden nodes is determined to be 20. In the case of Random Forest, the selected variables were the most with 5, and the Level 1 train RMSE also showed good performance with 0.01125. The best

Table 5: Final prediction of movie audiences in test set

Movie title	Predicted value	Target value	Movie title	Predicted value	Target value
Alog with the Gods: The Two worlds	14410221.9	14410721	Aladdin	11159042.2	11157279
1987:When the Day Comes	7201120.1	7201370	The Battleship Island	6591941.2	6592170
Captain Marvel	5800868.7	5801070	Jurrassic World: Fallen Kingdom	5661431.9	5661231
AQUAMAN	5051016.9	5038143	The Spy Gone North	4951170.6	4951690
Kingsman: The Golden Circle	4941728.2	4945486	The Mummy	3681479.5	3689290
Money	3325048.8	3325311	MEMOIR OF A MURDERER	2223232.3	2223094
Innocent Witness	2533246.1	2533334	Detective K: Secret of the Living Dead	2434349.2	2437146
Fantastic Beasts: The Crimes of Grindelwald	2413978.2	2414062	Toy Story 4	3328215.3	3328229
FENGSHUI	2079253.8	2079326	HitandRun Squad	1828211.5	1826256
Justice League	1788108.7	1786386	Ralph Breakds the Internet	1760587.2	1758891
Door Lock	1557897.5	1559945	Little Forest	1488607.7	1485408
Foregotten	1387017.2	1387011	Golden Slumber	1381317.2	1382358
Happy Death Day	1380505.5	1382650	Ocean's 8	1331864.0	1331858
The Vanished	1315740.9	1315735	The Mimic	1306443.8	1306438
Birthday	1184056.6	1183530	What a Man Wants	1194134.8	1194229
Hello Carbot the Movie: The Cretaceous Period	864337.8	864757	Dark Phoenix	851338.6	850960
Men in Blakc: International	814188.7	813794	LOVE+SLING	766110.0	766104
47 Meters Down	575115.0	575115	Insidious: The Last Key	551426.8	551953
The Grinch	546562.0	546553	Tomb Raider	534041.7	534211
Mary and the Witch's Flower	536270.5	535448	Seven Years Night	525303.4	526078
The Shape of Water	493973.7	494097	Car 3	466728.8	466047
Detector Conan: Crimson Love Leter	448508.1	448915	THE SOUL-MATE	446831.8	446717
Fall in Love at First Kiss	369075.4	369230	Theater Version Dionsaur Mecard: The Island of Tinysaurs	381776.1	381973
Loving Vincent	401736.5	401550	Peter Rabbit	379874.8	379748
MAN OF WILL	357620.4	357874	Godzilla: King of the Monsters	353017.8	353012
Crayon Shin-chan: Burst Serving! Kung Fu Boys	347852.5	347823	Dumbo	340152.3	340411
Paddington 2	339094.6	339014	Truth or Dare	310954.8	310962
Cinderella and Secret Prince	284648.0	284763	Paul, Apostle of Christ	160308.0	160277
Wonder	179840.9	179806	Criminal Conspiracy	260566.3	260512
12 Strong	223765.5	223740	Fifty Shades Freed	217744.1	217785
The Whispering	217534.2	217455	The House with a Clock in its Walls	214558.8	214452
The Hurricane Heist	214966.3	214949	American Assassin	205430.2	205544
The Curse of La Llorona	202683.6	202756	Namiya	181920.3	181885
Upgrade	190429.0	190387			

result was the Gradient Boosting method, which shows Level 1 train RMSE as 0.00022. Moreover, Lasso and Elastic Nets were selected as Gradient Boosting's input variables, and just the 30 small iterations showed very good results.

4.3. Final stacking model and result

As a result, the final selected stacking model and test RMSE which is calculated by the test set that are never used in training of Level 0 and Level 1, are shown in Table 3. Test RMSE was calculated to be 0.00074. When the stacking model is applied to test set, stacking model still has great power to predict the cumulative movie audience. Table 4 shows the RMSE in test set evaluated by all models from Level 0 models to final stacking model. Each model represents the final state after selecting hyperparameter using randomized search cross validation. The final stacking model which ensembles the algorithms in Level 0 has the smallest RMSE of test set comparing to others. So we can conclude that stacking which is a kind of ensembling method outperforms than other single algorithms.

Table 5 shows all the movie names in test set and the inverse log transformation of the target values, and the predicted values. It can be seen that the test set yielded very close predictions by the stacking model with 9 Level 0 models and Level 1 meta learner.

5. Discussion

In this paper, collecting and preprocessing of 13 independent variables are described to predict the final cumulative audience of the film. To prevent overfitting, the model was constructed with process of estimating generalized error based on the RMSE calculated through 5-fold cross validation, starting with separating the train data set for learning Level 0 and Level 1. By the stacking models trained with two steps of Level 0 and Level 1, a wide range of hyperparameter tuning leads to derive the improved performance than just a single model. As a conclusion, the proposed stacking model has shown that predictions of movie audiences are very close to the actual final cumulative audience.

It is meaningful for measuring and predicting the cumulative audience to help understand Korean domestic movie market. Also this study is meaningful that it uses stacking which is not frequently used in prediction of movie audiences before. However, just 9 Level 0 models and 5 Level 1 models with movies within 2 years were used in this paper. If more samples were collected and more models were trained at each level, it could lead to improve the performance of stacking. With the result of this study, we are expecting that more studies would be followed to improve the performance and bring further growth of understanding movie industry.

References

Breiman L (1999). Random forests, *UC Berkeley TR567*.

Friedman JH (2002). Stochastic gradient boosting, *Computational Statistics & Data Analysis*, **38**, 367–378.

Geurts P and Louppe G (2011). Learning to rank with extremely randomized trees, *Proceedings of Machine Learning Research*, **14**, 49–61.

Kim SY, Lim SH, and Jung YS (2010). A Comparative study of predictors of film performance by film types: focused on art and commercial films, *The Journal of the Korea Contents Association*, **10**, 381–389.

Korean Film Council (2019). 2018 Nyeon hangug yeonghwa san-eob gyeolsan bogoseo, Busan, Korea.

Lawson R (2015). *Web Scraping with Python* (1st ed), Packt Publishing Ltd.

Lee JM (2018). Juyo byeonsu seontaeggwa decision tree-leul hwalyounghan gaebong cheot ju yeonghwa heunghaeng yecheug-e daehan mosin leoning gibub yeongu (Doctoral dissertation), Hanyang University.

Lee K, Park J, Kim I, and Choi Y (2018). Predicting movie success with machine learning techniques: ways to improve accuracy, *Information Systems Frontiers*, **20**, 577–588.

Liaw A and Wiener M (2002). Classification and regression by randomforest, *R News*, **2**, 18–22.

Paniagua Tineo A, Salcedo Sanz S, Casanova Mateo C, Ortiz García EG, Cony MA, and Hernández Martín E (2011). Prediction of daily maximum temperature using a support vector regression algorithm, *Renewable Energy*, **36**, 3054–3060.

Park SY (2012). SNSleul tonghae gujeon hyogwaga yeonghwaheunghaeng–e daehae michineun yeonghyang : ssuny–ui salyeleul jungsim–eulo, *The Journal of the Korea Contents Association*, **12**, 40–53.

Pontil M, Rifkin R and Evgeniou T (1998). From regression to classification in support vector machines.

R Core Team (2019). R: A language and environment for statistical computing, R Foundation for Statistical Computing, URL https://www.R-project.org/.

Segal MR (2004). Machine learning benchmarks and random forest regression, *UCSF: Center for Bioinformatics and Molecular Biostatistics*, Retrieved June 2, 2021, from: https://escholarship.org /uc/item/35x3v9t4.

Solomatine DP and Shrestha DL (2004). AdaBoost. RT: a boosting algorithm for regression problems. In *Proceedings of the 2004 IEEE International Joint Conference on Neural Networks* (IEEE Cat. No. 04CH37541), 1163–1168.

Solomatine DP and Shrestha DL (2006). Experiments with AdaBoost. RT, an improved boosting scheme for regression, *Neural Computation*, **18**, 1678–1710.

Song Y, Liang J, Lu J, and Zhao X (2017). An efficient instance selection algorithm for k nearest neighbor regression, *Neurocomputing*, **251**, 26–34.

Van Rossum G and Drake FL (2009). *Python 3 Reference Manual*, SohoBooks, United States.

Wolpert DH (1992). Stacked Generalization, *Neural Network*, **5**, 241–259.

Yu JP and Lee EH (2018). A model of predictive movie 10 million spectators through big data analysis, *The Korean Journal of Bigdata*, **3**, 63–71.

Zhang Y and Haghani A (2015). A gradient boosting method to improve travel time prediction, *Transportation Research Part C: Emerging Technologies*, **58**, 308–324.

Zou H and Hastie T (2005). Regularization and variable selection via the elastic net, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **67**, 301–320.

Received October 20, 2020; Revised January 08, 2021; Accepted February 13, 2021