Machine Learning on TV Drama View Count Prediction

Runming Huang, Zou Xinqi, Xinyuan Chanel Yang

Abstract

With the rise of the internet and video technology, the economic scale of the TV drama industry is growing continuously. The view counts of each TV series is often directly equivalent to the size of the economic benefits that can be generated by that TV production, and there are many features influencing the view counts, such as the casting, topic, the era of the launch, etc. The aim of this paper is to model the data of the TV drama library on the *Tencent Video* platform, through the creation and comparison of different models, and finally arrive at a modelling logic that can classify the view counts of TV products as low, medium, or high, and keep the accuracy within an acceptable range.

Keyword: view counts, TV drama, machine learning

1 Dataset

1.1 Data Acquisition

Due to the large number of features involved in the contribution to the final view counts performance, other existed databases, except for official video website platforms, tend to show incomplete data. Therefore, this paper will use crawler method to independently obtain information on all TV series products on a particular platform and the features that affect them. The features we take into consideration: episode number, view counts, produced year, TV drama description, TV drama topic and two primary stars of the TV drama. The data collection tool we use is the Selenium tool pairing with Crawler programming. Selenium is essentially browser driven, fully simulated browser operation, and the URL will play a role in guiding the interface in this part. Considering the needs to compare the advantages and disadvantages between different models and the multitude of features involved in training, we will concentrate on the specific platform *Tencent Video*, the video platform with the largest number of users in China.

number	title	episode number	view counts	year	description	topic	Star1	Star2
1	突围	45	13.3亿	2021	靳东闫妮揭5y巨款之谜	当代主旋律	靳东	闫妮
2	红旗渠	31	7339.1万	2021	十年奋战造千里长渠	当代主旋律		
3	功勋	48	4.0亿	2021	雷佳音周迅演绎时代楷模	当代主旋律	雷佳音	周迅
4	启航：当风起时	36	19.2亿	2021	吴磊侯明昊冲浪90年代	当代主旋律	吴磊	黄明昊
5	花开山乡	34	1.8亿	2021	讴歌新时代 致敬扶贫人	当代主旋律		
6	火红年华	32	4473.3万	2021	重温川南钢铁峥嵘岁月	当代主旋律		
7	青谷子	37	1.1亿	2021	张一山孙耀琦演绎知青故事	当代主旋律	张一山	孙耀琦
8	在希望的田野上	24	9211.1万	2021	曹骏安悦溪家乡搞事业	当代主旋律	曹骏	安悦溪
9	日头日头照着我	35	7348.3万	2021	年轻干部带领农民奔小康	当代主旋律		
10	那些日子	20	2216.3万	2021	扶贫英雄人物振兴乡村	当代主旋律		
11	山海情[原声版]	23	4.9亿	2021	群星携手扶贫攻坚	当代主旋律		
12	暴风眼	40	8.1亿	2021	杨幂张彬彬致敬国安战士	当代主旋律	杨幂	张彬彬
13	温州三家人	36	6332.4万	2021	温商后浪筑梦互联网，年轻人独立创业闯天地	当代主旋律		
14	鲜花盛开的山村	33	5478.2万	2021	小人物助力乡村满溢幸福	当代主旋律		
15	江山如此多娇	31	8491.6万	2021	罗晋袁姗姗扎根新农村	当代主旋律	罗晋	袁姗姗
16	我们的新时代	48	1.3亿	2021	谭松韵白敬亭演绎基层故事	当代主旋律	谭松韵	白敬亭

Fig. 1 Data Crawler Result

1.2 Data Preprocessing

The data preprocessing part starts with applying the classification values (0,1,2...) to replace the topic and year, aiming to numerate the related information so as to be applicable to model training. Based on the dataset we generated through web crawler, there are no null values for topic and year attributes.

At the same time, this paper finds that casting, is an important feature influencing view counts, as the fans of stars are always the large potential audience base for the TV series these stars have been performed in, thus casting should obviously be included in the choice of variables for the model. Hence, this paper uses *Sina Weibo* as the platform guideline, matching the names of the stars mentioned through the *Sina Weibo* fan database and replacing them with the number of fans as the numeric value of casting. Referring to the *media platform industry ratings card measure*, Only the top two stars in terms of number of fans will be retained as a model features.

For the information of view counts, which needs to be identified as the key attribute that we target to use for final analysis, this paper will implement the K-means clustering method to turn the view counts into different classes, the exact number of classes is based on the results of the K-means clustering. Based on the Fig.2, this paper will take 3 as the class number for classification model. In other words, we will categorize the view counts into high, medium and low classes by use of index 0,1,2; Values of $\frac{1}{3}^{th}$ largest, $\frac{2}{3}^{th}$ largest will be set as boundaries to divide our view counts into three classes equally.

We separate our data into train set, validation set and test set with ratio: 0.75: 0.15: 0.10. Training set is used to train every model, validation set is used to tune hyper-parameters and select features. Test set is used to test each model's accuracy at last.

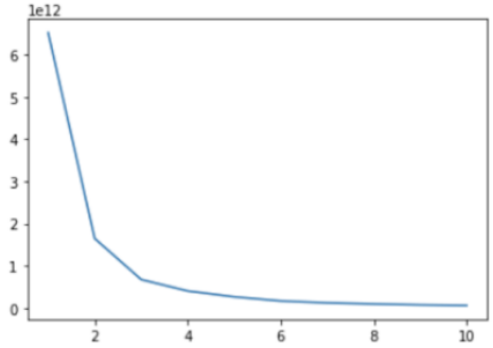

Fig. 2 Mean Error Vs Cluster numbers

2 Training Method and Hyper-parameter Interpretation

2.1 Tuning method

The Tuning method we use is GridSearch that we exhaustedly illustrate all combinations of hyper-parameters we try to tune. We will use hyper-parameters combinations with highest validation accuracy.

2.2 Ridge Regression and Logistic Regression

For the models, our loss function for Ridge Regression is as follows:

$$J_w = \min_{\omega} \left\{ (Y - X\omega)^T (Y - X\omega) + \lambda \omega^T \omega \right\}$$

Our loss function for Logistic Regression is as follows:

$$J(w) = -\sum_{i=1}^{N} \{ y_i \log p(x_i; \omega) + (1 - y_i) \log(1 - p(x_i;)) \}$$

We try to tune λ to optimize our model.

2.3 KNN

For weight we uses "distance" rather than "uniform", because we want closer neighbors have greater influence. For metrics, we use "Euclidean" over "Manhattan", "Hamming."

Euclidean space R^n is a finite-dimensional real vector space having an inner product defined on it, inducing a metric. A Euclidean distance matrix, an EDM in R(N×N,+) , is an exhaustive table of distance-square d_{ij} between points taken by pair from a list of N points $\{x_l ... N\}$ in R^n; the squared metric, the measure of distance-square:

$$d_{ij} = ||x_i - x_j||_2^2 = < x_i - x_j, x_i - x_j >$$

2.4 Decision Tree with Random Forest, Bagging, Boosting

For Decision tree, we will use Decision Tree with Random Forest, Bagging, Boosting and respectively tune on their hyper-parameters. For Boosting, we will try both Adaboosting and Gradient Boosting.

2.5 SVM

For SVM, we use Radial basis function Kernel over Polynomial Kernel, Sigmoid Kernel, Linear Kernel. Then we use GridSearch to tune L2 regularization λ and Γ in the kernel. The Radial basis function Kernel is as followers:

$$K(x, x') = exp(-\frac{||x - x'||^2}{2\Gamma^2})$$

2.6 Multi-Layer Neural Network

Considering having insufficient features in our Neural Network, we also use Natural Language Processing encoding with ONEHOT encoder to process the "description" column. For each column, there are 2952 rows of words, and each row gives a brief description of one drama. Since our database is Chinese dramas, the python package we use to divide words is *"jieba"*.

We download lots of basic Chinese word dictionaries from Github to make sure that it will not divide some meaningful word chunks apart. We also download 6 dictionaries of 18087 names of Chinese stars from *SouGou* and *Baidu* thesaurus to let *"jieba"* not detect names as useless word chunks. To remove stop words and special characters, we collect Chinese stop word dictionaries containing 3779 stop words from Github to clear them automatically. These dictionaries can be plugged into *"jieba"* as loading data for further selection and division.

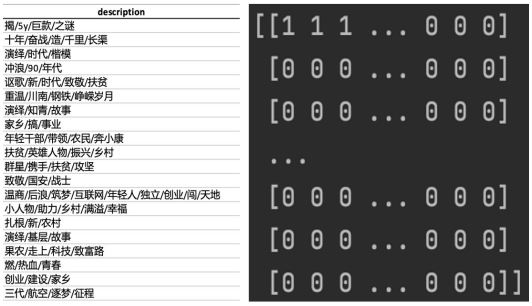

Fig. 3. NLP Result and ONEHOT Result

After getting the processed divided description data, we use ONEHOT encoder to transform the description column into a 0 and 1 MAP. By adding this feature into every drama and rerun our model, we can testify whether the model accuracy increases or decreases.

2.7 Feature Selection

For every model, after hyper-parameter tuning, it is conducive to find if the model will be better if reducing some features. Since there are only 5 features, this paper will consider to try out all different combinations of our features. We will select features with best validation accuracy.

3 Modelling and Training

3.1 Linear Regression and Logistic Regression Both with L2 regularization

For ridge regression, we tune the L2 regularization parameter λ on the validation set. The λ we choose is 0.011. After tuning λ, we process feature selections and select "episode number", "Star 1 fan number", "Star 2 fan number". The validation accuracy before selection is 0.4580, and 0.5351 after selection. Using chosen λ and selected features, the test accuracy is 0.5017. For logistic regression, we tune the L2 regularization parameter λ on the validation set. The best λ we get is 0.005. After tuning λ, we process feature selections and select "episode number", "Star 1 fan number". The validation accuracy before selection is 0.4678, and 0.5085 after selection. Using chosen λ and selected features, the test accuracy is 0.4610.

3.2 KNN modelling

For KNN, we improve the model by selecting distance function, points weighted methods, and the number of nearest neighbors. We choose them by adapting the GridSearch method. We choose Euclidean Distance over Manhattan Distance and Hamming Distance, "distance"(weight points by the inverse of their distance) weighted method over "uniform" and the number of nearest neighbors as 24 over [1,100]. After tuning, we process feature selections and select "episode number", "Star 1 fan number", "year". The validation accuracy before selection is 0.6349, and 0.7324 after selection. With selected hyper-parameters and features, the test accuracy is 0.6305.

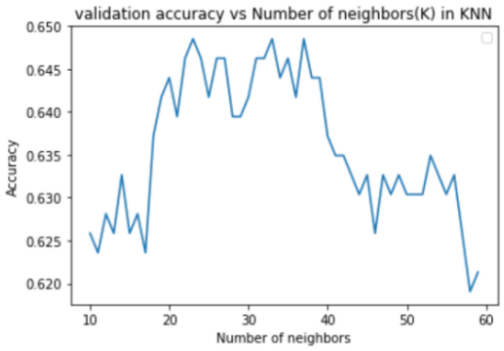

Fig. 4

3.3 Decision Tree with Random Forest, Bagging, Boosting, and Gradient Boosting

3.3.1 Decision Tree

For the general Decision Tree method, we only process feature selections and select "episode number", "Star 1 fan number", "year". The validation accuracy before selection is 0.6236, and 0.6666 after selection. With selected hyper-parameters and features, the test accuracy is 0.6237.

3.3.2 Random Forest

For Random Forest, we improve the model by selecting the loss function, the maximum depth of the tree, the number of features to consider when looking for the best split, the number of trees in the forest. After GridSearch, we choose Crossentropy Loss with the maximum depth of 22, the number of trees of 40, features each split of 2. After tuning, we process feature selections and find validation accuracy the same before and after selection. We drop "topic". Consequently, the test accuracy is 0.6576.

3.3.3 Bagging

Similarly, for bagging, we tune maximum features to consider each split and the number of trees in bagging. We choose the number of trees of 27, features each split of 2. After tuning, we process feature selections and get 0.7211 validation accuracy and 0.7075 before selection. We drop "Star 1 fan number". Consequently, the test accuracy is 0.6203.

3.3.4 Boosting

Adaboosting:
Similarly, for AdaBoosting, we only tune the number of trees in bagging. We choose the number of trees of 155. (see graph) After tuning, we process feature selections and get 0.6689 validation accuracy and 0.6531 before selection. We drop "Star 1 fan number". Consequently, the test accuracy is 0.6712.

Gradinet Boosting:
For Gradient boosting, we tune the maximum depth of the tree and the number of trees. We choose the number of trees of 22, maximum depth of 3 (two figures). After tuning, we process feature selections and get 0.7256 validation accuracy compared with 0.7075 before selection. We drop "Star 1 fan number". Consequently, the test accuracy is 0.6644. Fig. 4 is the graph fixing the maximum number of tree, representing the accuracy of choosing different maximum depth. Fig. 5 is the graph fixing the maximum depth, representing the accuracy of choosing different number of trees.

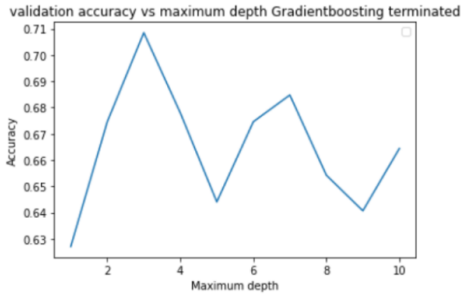

Fig. 5

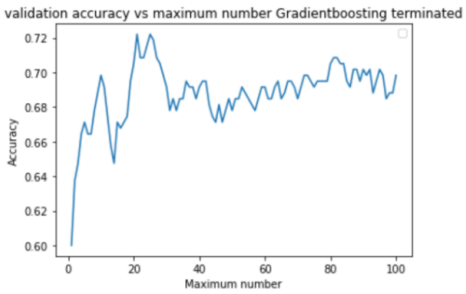

Fig. 6

3.4 SVM modelling

For SVM, we improve the model by selecting kernels first. We choose the Radial basis function kernel. Then we tune on L2 regularization λ and Γ in the kernel. We choose them by adapting the GridSearch method. We choose λ of 2.3 and Γ of 12. After tuning, we process feature selections and drop "topic". The validation accuracy before selection is 0.6531, and 0.6802 after selection. With selected hyper-parameters and features, the test accuracy is 0.6203. Fig. 6 fix the L2 regularization hyper-parameter with highest accuracy. Fig. 7 fix Γ with highest accuracy.

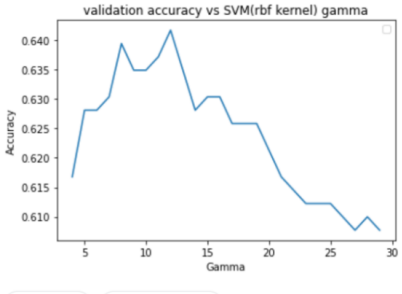

Fig. 7

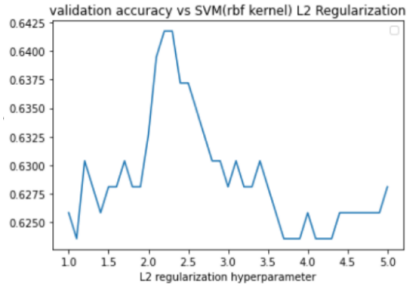

Fig. 8

3.5 Multi-layer neural network

For non-NLP, we build our Multi-layer Neural Network with 9 hidden layers and 128 nodes in each layer. We use SGD with Momentum(learning rate of 0.01, momentum rate of 0.9) and Adam(learning rate of 0.001) respectively. SGD with Momentum performs better on test set compared with Adam. The test accuracy of SGD with Momentum is 0.6881. For non-NLP, we build our Multi-layer Neural Network with 4 hidden layers and 128

nodes in each layer. We use SGD with Momentum(learning rate of 0.01, momentum rate of 0.9) and Adam(learning rate of 0.001) respectively. Adam performs better compared with SGD with Momentum. The test accuracy of Adam is 0.6102.

4 Results and Discussion

The results section details your metrics and experiments for the assessment of your solution. It allows you to compare your idea with other approaches you've tested.

4.1 Result

Model	Ridge Regression	Logistic Regression	KNN	Neural Network(With NLP)	NeuralNetwork(Without NLP)
Testing Accuracy	50.12%	46.10%	63.05%	68.82%	64.06%
Model	Decision Tree	Random Forest	Bagging	Boosting	SVM
Testing Accuracy	62.37%	65.76%	62.03%	67.12%	62.03%

Fig. 9 Result Table

Ridge Regression is relatively a simply model. It might not properly fit data when data is not linearly distributed. Logistic regression also suffers from assumed linearity. When data doesn't have close to linear boundary, Logistic regression might also have trouble to fit data properly. SVM has better performance. However, it suffers from noise from data. When data has relatively more noise, SVM might not fit properly. KNN has relatively better performance. However, since it has better performance on 3 features compared to 5 features, it might suffer from the curse of dimensionality. Decision Tree has good performance. Desicion Tree using Boosting performs better than Random Forest and Bagging. This might result from our relatively small data size. Neural Network has great performance. It performs better when NLP data is not added. Maybe the NLP data is not practical since our data set is not big enough. When NLP data is added, Neural Network will over-fit data, thus reducing Network layers help increase test accuracy. However, Neural Network is significantly smaller than other models.

4.2 Future Improvements

The bets of our models are much better than random guess. However, none of our models have accuracy higher than 70%. Further improvements can be made to increase our model's performance.

This paper argues that there are three main components worth future enhancement. The first is in regards to the feature selection. In the process of feature selection, we apply GridSearch. However, based on online research, according to paper from Universite de Montreal, random search may present more optimised results than GridSearch, and therefore more attempts can be made in this direction in the future. Another part worth future improvement is NLP analysis. The slogan and title of the TV product cannot be optimised for correctness with the current limited data and unoptimised structure, in the future, with more sufficient data. In this case, we can get more useful and high frequency words to treat them as our features.

Lastly, more data is needed for our model. Currently, we only collect data from *Tencent* platform. We can also collect data from other video platforms, such as *Youku*. We can also find more features for our data, including but not limited to posters of the TV drama.

References

[1] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization." *Journal of machine learning research*, vol. 13, no. 2, 2012.

[2] J. Dattorro, *Convex optimization & Euclidean distance geometry.* Lulu. com, 2010.