Prediction of Steam Game Sales

Enmin Zhou
Github: https://github.com/Enmin/steam_game_sales

Introduction:

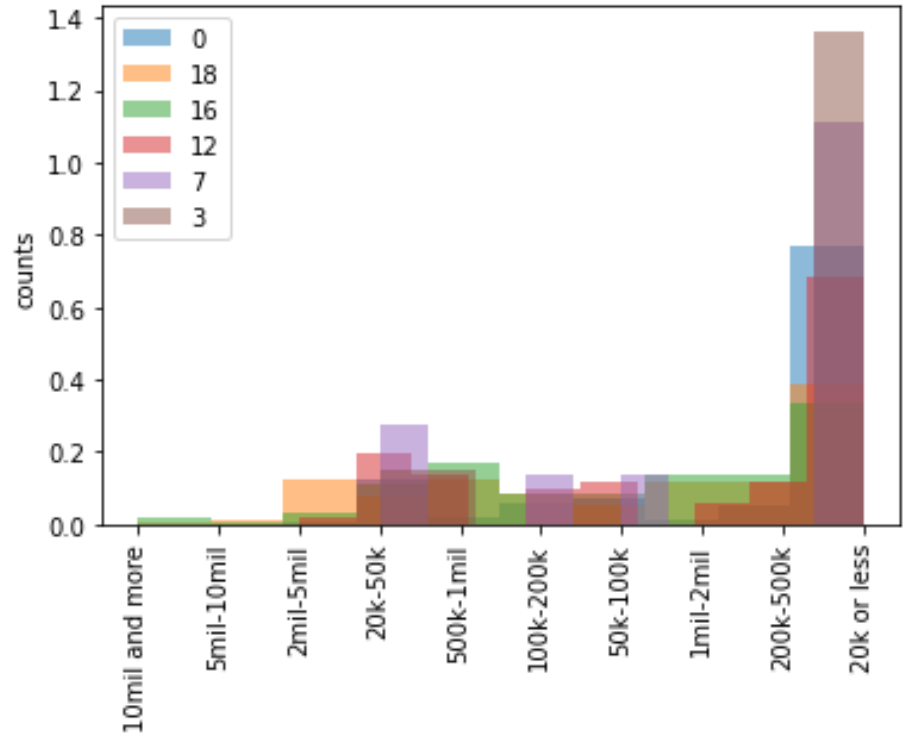
1. The target variable is the number of owners of a chosen game.
2. The problem is a classification
3. Game is an important field in entertainment industry. Knowing how to predict the popularity of a game will help us to design a game.
4. Number of data points: 27075. Number of features: 18
5. Feature Description
    appid: integer; it is the unique code of a game
    name: string; it is the name of a game
    release_date: string “yyyy-mm-dd”; the date of the game released
    english: categorical; 0 means the game does not support enligsh and 1 otherwise
    developer: categorical; string such as ‘valve’ represents the developing company of a game
    publisher: categorical; string such as ‘valve’ represents the publishing company of a game
    platforms: categorical; string such as ‘windows’ represents the system it requires to play a game
    required_age: categorical; string such as ‘18’ represents the minimum age requires to play a game
    categories: categorical; string such as ‘single-player’ or ‘online’ describes how to play a game
    genres: categorical; string such as ‘Action’ or ‘Strategy’ represents the game style
    steamspy_tags: categorical; string such as ‘World War II’ represents the background keywords of a game
    achievements: numerical; an integer that represents the number of in-game achievements if any
    postive_ratings: numerical; an integer that shows the like of the game
    negativve_ratings: numerical; an integer that represents the dislike of the game
    average_playtime: numerical; an integer that represents the average playtime of the game
    median_playtime: numerical; an integer that represents the median playtime of the game
    owners: categorical; a range that represents the number of owners of the game
    price: numerical; a float number that represents the sale price of the a game

6. The dataset if from Kaggle. There are in total 8 notebooks under this dataset, all of which performed some EDA, but none of them creates a model to solve a certain problem using this dataset.

Exploratory Data Analysis

Performing the EDA, I find out that some of the categories in categorical features only have one data point, such as ‘developer’ and ‘publisher’. Therefore, I merge those categories into one category. I also find out that the ‘release_date’ cannot be roughly categorized because two dates has strict relationship of time, so I transform ‘release_date’ to timestamp in order to show the time relationship.

See figures below:



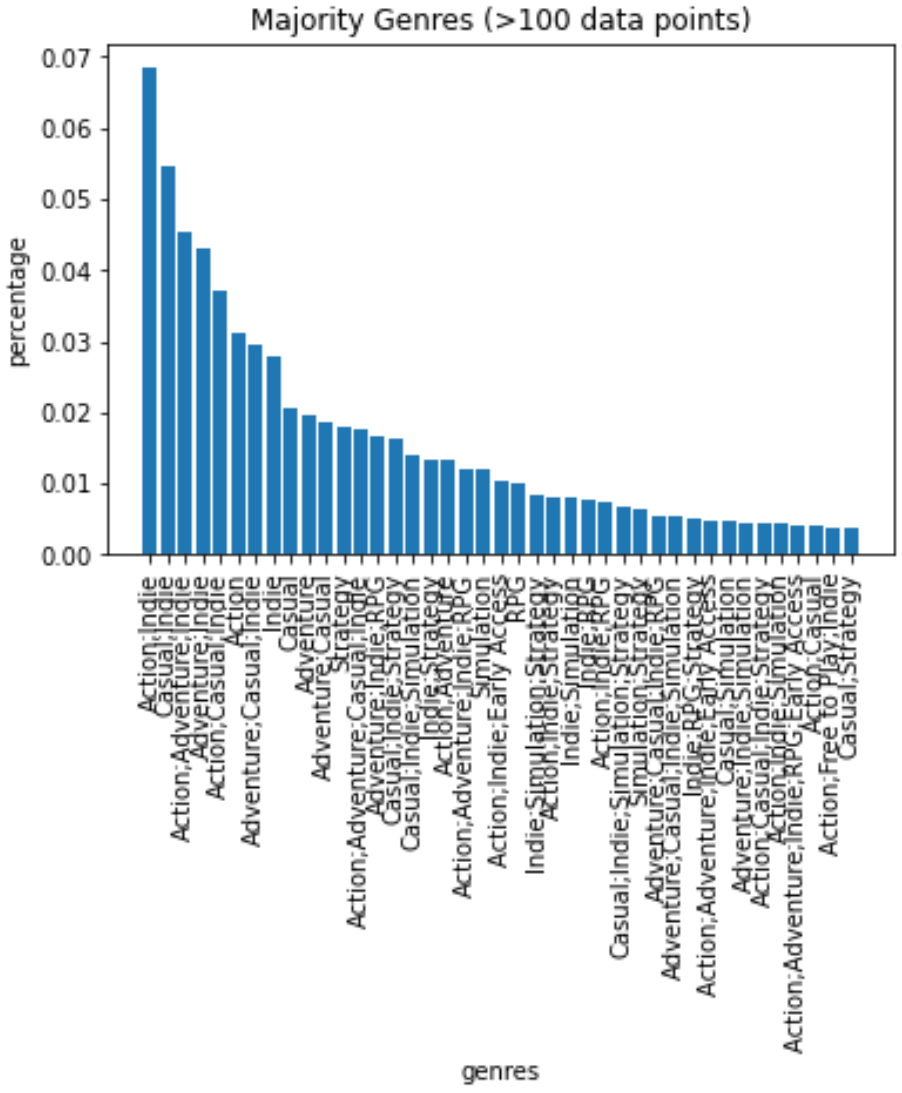


Fig.3 X is the target categorical variable "genres" and y is the percentage of each category. From the plot, we can see that "Action" and "Indie" style game is the majority

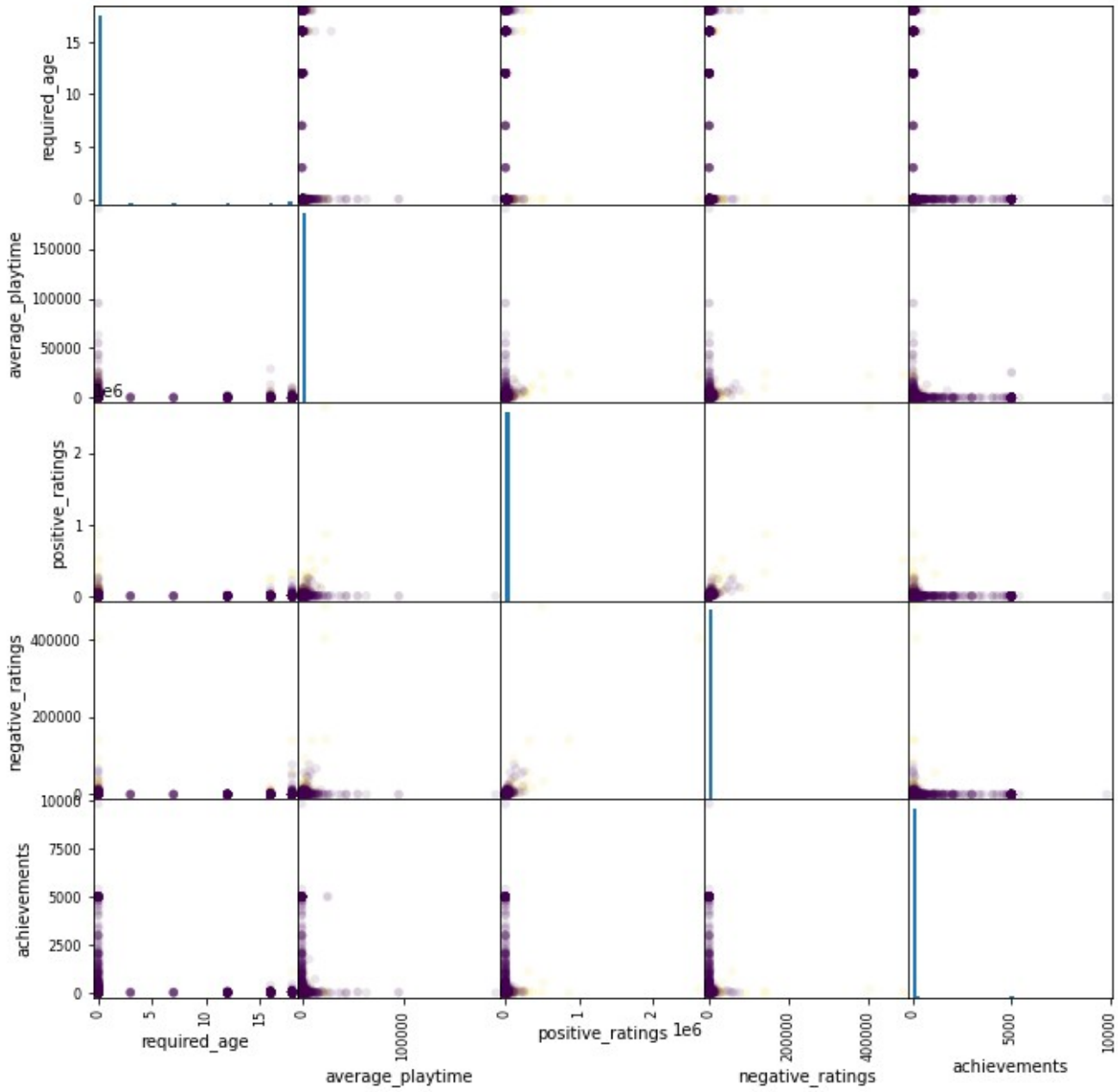


Fig.4 Scatter Matrix of 'required_age','average_playtime','positive_ratings','negative_ratings','achievements'

Data preprocessing

1. My dataset has 27075 data points so I decide to take 1000 data points as test data and the rest 26075 points as train and

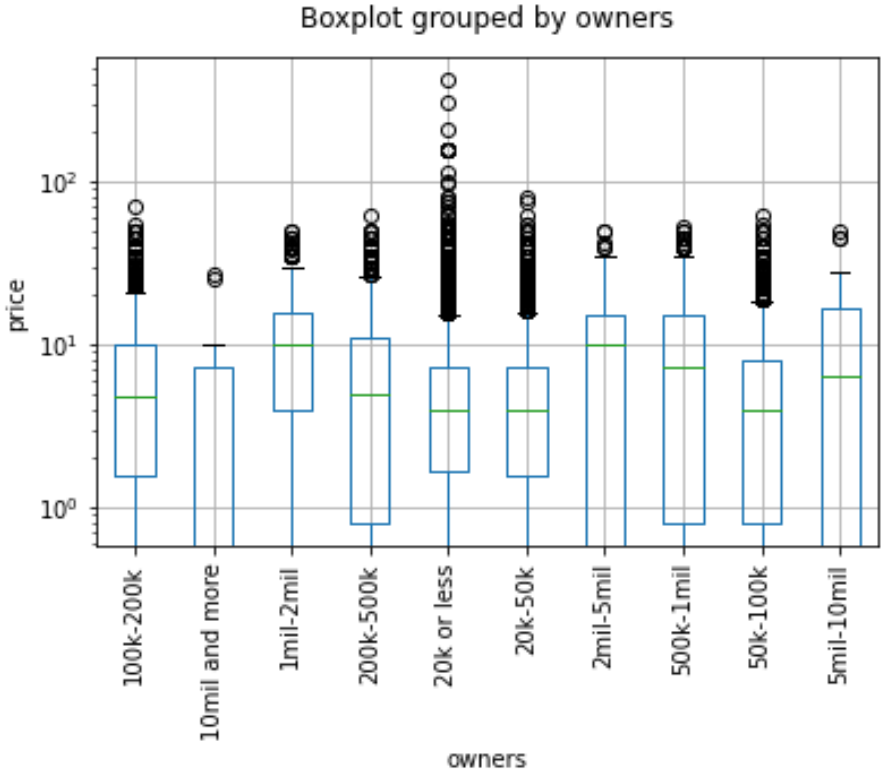


Fig.2 X is the target categorical variable "owners" and y is the price of the game scaled by log. From the plot, we can see that the price range is larger for game that has more owners

validation data.
2. My data set is IID because every data point is collected from a unique game.
3. It does not have group structure since no game is counted twice in this data set.
4. It is not a time series data since every game is only counted once after its release date.
5. My target variable is a categorical variable so that I need to use stratification when I split the dataset.
6. I use standard scaler for 'price','positive_ratings','negative_ratings','average_playtime','median_playtime' because they are continuous features and has no boundaries on them. Standardize these columns around zero will help the training process become easier.
I use onehot ecoder for 'developer', 'publisher', 'platforms', 'steamspy_tags','required_age','categories','genres','english' because they are categorical features and I preprocess these columns to make sure that the category in each feature has more than 1 data point. (If there is only one data point in a feature's category, I will gather those categories by setting them to 'other')
I use label encoder on the 'owners' feature, which is the target variable, because this feature was originally in the type of string.
7. I have 16 features in my preprocessed data. I drop the feature 'name' and 'appid' because they only identify a game and do not provide any feature information.

Methods

1. The splitting strategy that I have used is stratification. Since the data set is unbalanced in terms of the target variable, I used "train_test_split" with stratify set to target variable. The train and test are split in

the ratio of 9:1 since the data set is large and the test points will still be enough even with 10 percent of the whole dataset. Furthermore, I use “StraitifiedKfold” to further split train set into train and validation.
2. I tried 6 algorithms:

Algorithms	**Parameters Tuned**
Support Vector Machine Classifier	C (regularization parameter)
K Neighbors Classifier	n_neighbors
Random Forest Classifier	max_depth, max_features
Ridge Classifier	alpha
Stochastic Gradient Descent Classifier	alpha, l1_ratio, penalty
Logistic Regression	C, multi_class (with max_iter set to 10000 to converge)

3. I use f1_weighted to evaluate my results. The “f1_weighted” simply calculate the f1 score for every class of my target variable, and average using weights of each class. I choose this metric because my project is a multi-class classification task. The f1_score can show the harmonic mean of precision and recall score for my model to predict each class of target variable, while using “weights” to average the total f1_score can show the general performance of the whole task.
4. I set 10 random states so that each model with each combination of parameters will be measured and trained under 10 different random states. The standard deviations and mean value is calculated to choose the best model.
5. For feature importance, I also set 10 rounds to calculate the mean and standard deviations of permutation score before coming into a conclusion.
6. In general, I pass the ‘passthrough’ as the value for the parameter “remainder” in my “Column Transformer” because I manually create some one-hot features in the EDA section and the preprocess section. The reason I choose Support Vector Machine, Ridge and Logistic algorithm is that I want to see the performance of these binary classification algorithms on multi-label classifications. However, the result in the next section show that even though we can change multi-label into “one versus rest” type, these three algorithms performs not that well on the task.

Results

The general results:

Algorithms	Mean Score	Standard Deviations	# of Std above baseline
Random Forest	0.797	0.00290	84
K Neighbors	0.646	0.00572	16
Ridge	0.589	0.00241	15
Support Vector Machine	0.558	0.0010	4
Stochastic Gradient Descent	0.475	0.15	-0.5
baseline	0.554	N/A	0

1. From the “f1_weighted” score, the best model is “Random Forest Classifier” with max_depth 10 and max_features 1.0. The “Random Forest Classifier” has a 0.793 mean score in cross validation while remain a low standard deviation of 0.0029. The baseline “f1_weighted” score of the task is 0.554. The “Stochastic Gradient Descent Classifier” performs even worse than the baseline.

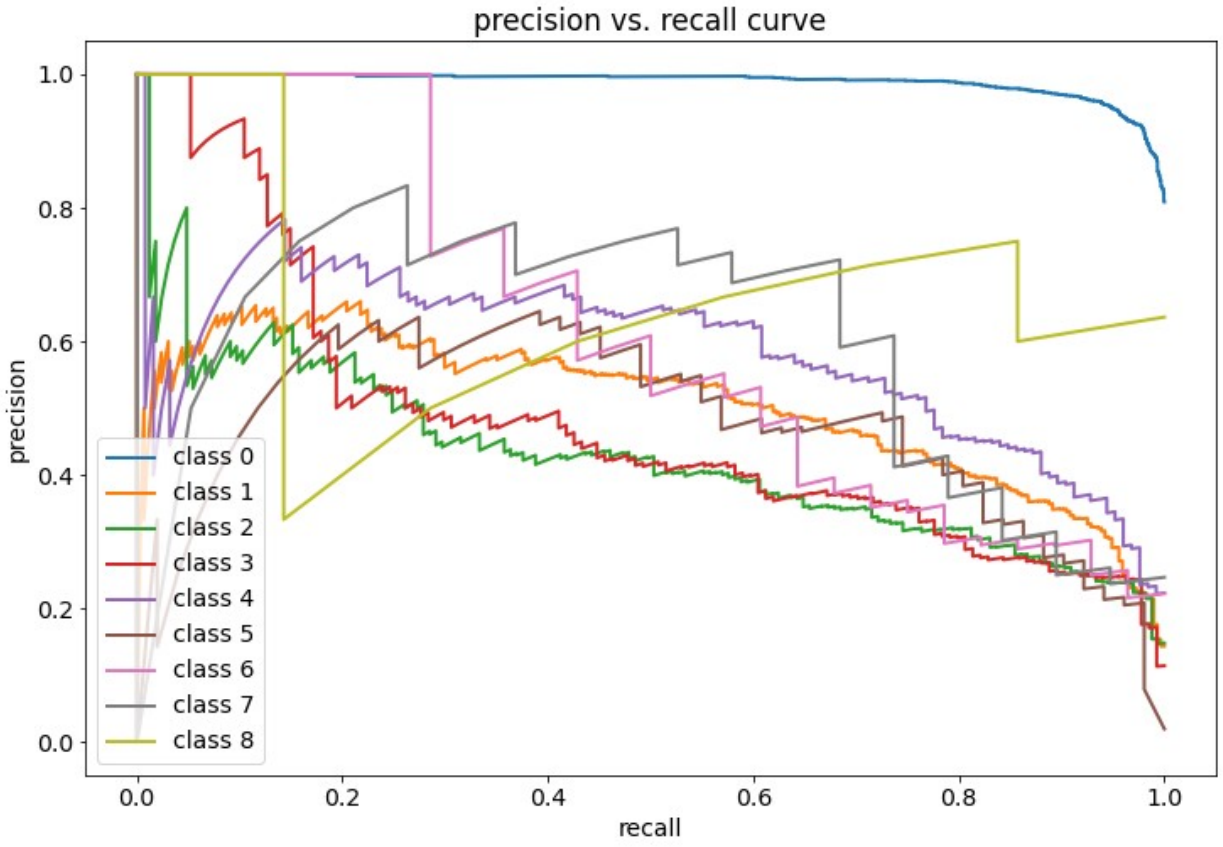


Fig.6 The result graph based on setting each class as a binary classification in the method of "one against all"

From the following confusion matrix, we can see that the model might make some mistakes between the classes that are neighbors. Meanwhile, the model performs strongly in the lowest 2 classes (0, 1) and highest 3 classes (6, 7, 8), but performs relatively weaker in the middle 4 classes (2, 3, 4, 5). The reason might be that the extreme cases have strong indicators while the middle 4 classes’ indicators are blurry.

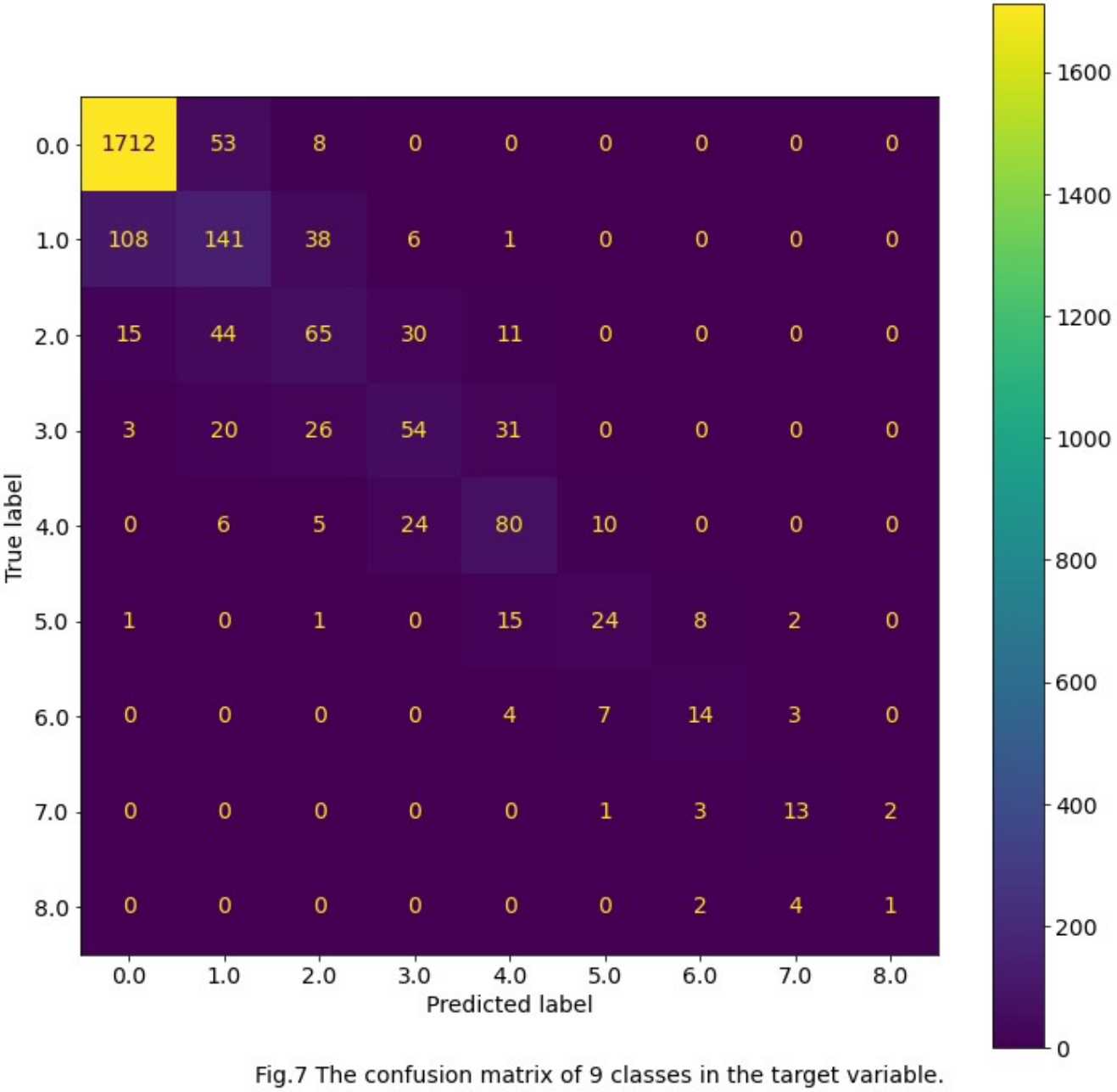
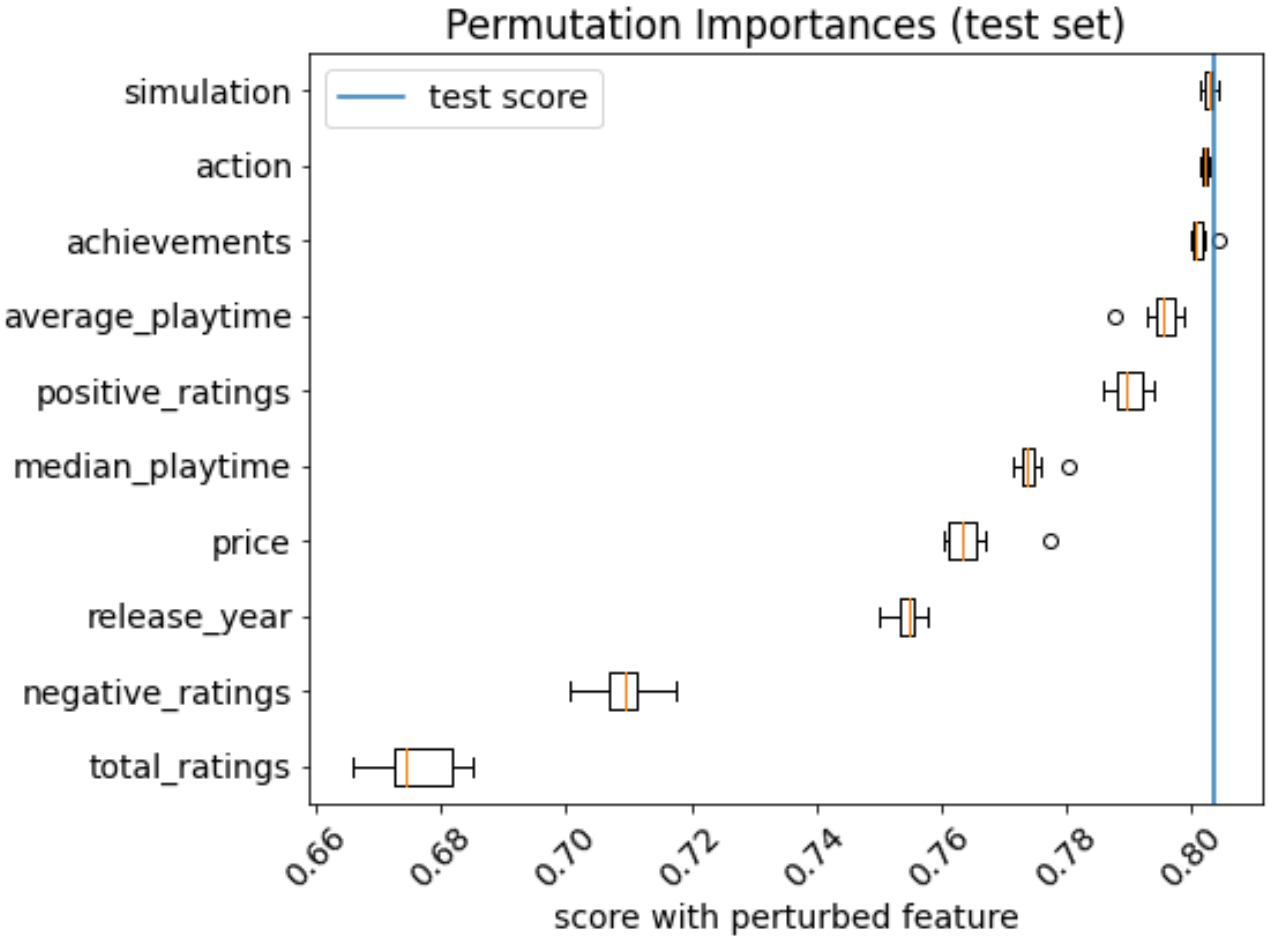
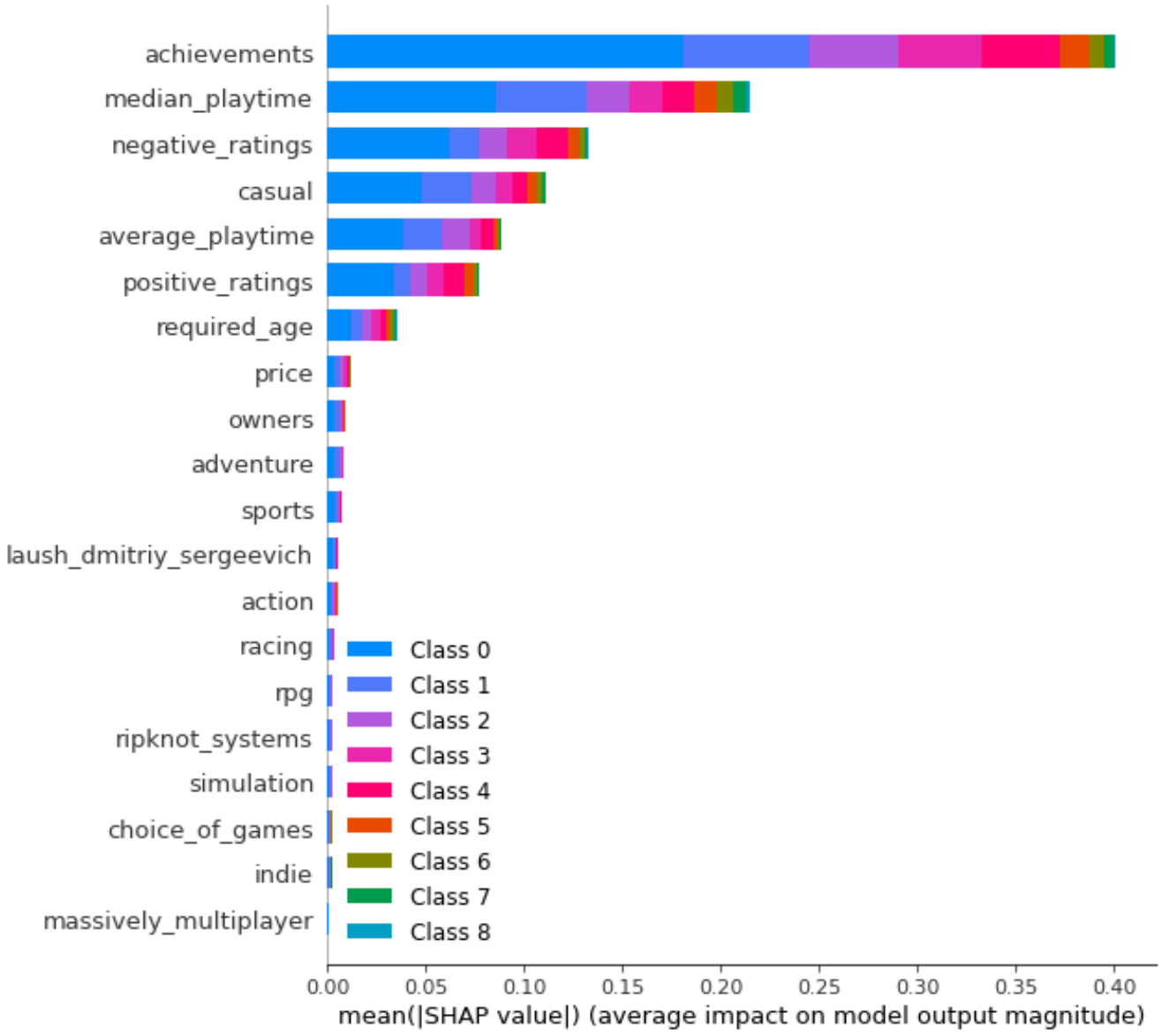


Fig.7 The confusion matrix of 9 classes in the target variable.

2. In the next step I analyze the feature importance. In the global sense, I use the permutation feature importance, and the results is in the following picture.



In the local sense, I calculate the shap value of the test set and the result graph is as following.



3. I have also tried to re-sample the training data-set to let the model fit a more balanced data-set. I have used two re-sample methods, the first one is ‘average’ and the second one is ‘upward’. ‘Average’ samples points from majority class to an average number while sample points from minority class to an average number with replacement. ‘Upward’ samples every class that is below a certain ‘ratio’ to the largest class to that ratio with replacement and keep others unchanged. However, both methods does not improve the test results. The ‘avg’ re-sampling gives a 0.74 test score while ‘upward’ gives a 0.75 test score. Both of which are worse than the original one.

Outlook

1. I haven’t tried “xgboost” but I expect its behavior to be the same as Random Forest Classifier, but this algorithm worth a try.
2. One more thing I can do is to use neural network and deep learning to learn the relationship between these features and the target variable, since the features might be too complex for the algorithms that I have tried.

3. The additional data source can be collected through games data outside the steam market. Since players who own games in the steam market also play games on other platforms, those data also reflect certain relationship between game sales certain features. By collecting those data, we can not only increment the size of our data points, but also expand our feature set so that more dimensions are included and some important new features would be explored.

Reference

dataset source on Kaggle: https://www.kaggle.com/nikdavis/steam-store-games?select=steam.csv

Github Repository: https://github.com/Enmin/steam_game_sales