**COMP 4211 Project Report (Spring 2019)**

CHEN, Yifei (20328874)，LI, Siqi (20328056)

**Part 0: Project Introduction**

Our project applies the dataset about Google app store, which include googleplaystore.csv and googleplaystore\_user\_reviews.csv, to implement this project. Our objectives of the projects are:

1. Predicting sentiments for reviews;
2. Predicting ratings for apps.

The project is under the environment of python3. We will present the details of each part in the following sections

**Part 1: Predicting sentiments for reviews**

Inspired by what we learned about classification model in class, we decide to use the decision tree model to solve this problem. Furthermore, we also apply some advanced decision tree model for this question to make some comparison.

1. Data Preprocessing

Firstly, we load the data from googleplaystore\_user\_reviews.csv via panda:

图片包含 屏幕截图

描述已自动生成

Then, what we need are valid rows (with effective Translated\_Review and sentiment). And we change the value of sentiment to integers for convenience of future training:

图片包含 屏幕截图

描述已自动生成

To get a better input feature set, we also need to regularize the translated review, which contains:

1. Delete ‘!’, ‘.’, ‘''’, ‘...’, ‘:’, ‘#’ and so on;
2. Delete the words which are unnecessary, which are absolutely without sentiment (eg. ‘a’, ‘for’, ‘are’, ‘the’, and so on);
3. Break the sentence in to words (features);
4. Lower cases;
5. Lemmatization (eg. Liked -> like, plays -> play).

图片包含 屏幕截图

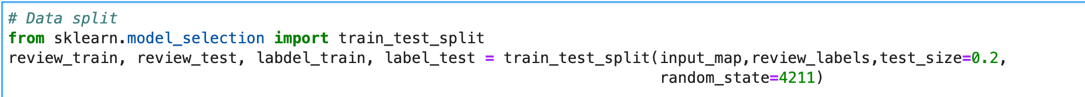
描述已自动生成

There are too many features in total (total number of unique words) actually. We don’t want a model which are time costing and likely overfit, so we only select 2000 features in input. Also, each review will be represented as a vector of these feature, the value is the number of the occurrence of corresponding word.

图片包含 屏幕截图

描述已自动生成

Lastly, with train-test split:



1. Method 1: Single decision tree

For this method, we import the decision tree classifier from scikit-learn. And we construct a class for my model, which includes initialization, train, test, and confusion matrix functions:

图片包含 屏幕截图

描述已自动生成

图片包含 屏幕截图

描述已自动生成

Then what we do is to set different max-depth of tree, see the results (the criterion can also be changed, here we only use “entropy” as the way to calculate information gain):

图片包含 屏幕截图

描述已自动生成

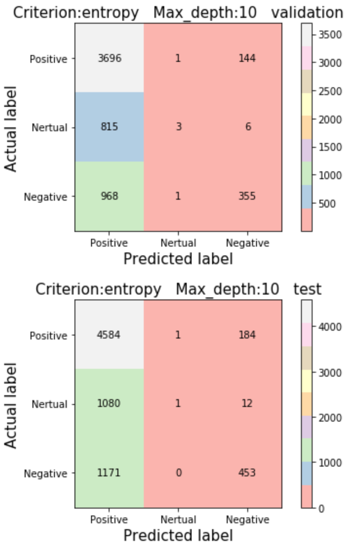
Below are the results:

图片包含 文字, 报纸

描述已自动生成

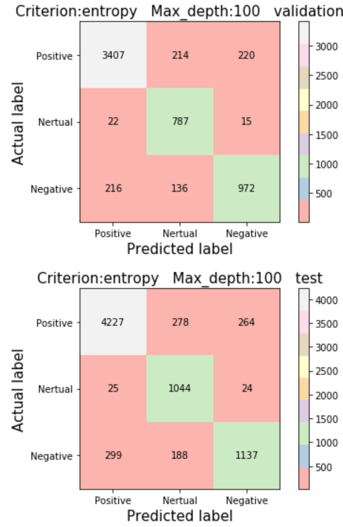
图片包含 屏幕截图, 地图

描述已自动生成

图片包含 屏幕截图

描述已自动生成图片包含 屏幕截图

描述已自动生成



We find that the accuracy increases with the increasement of the depth of the tree, while it also takes more time. Finally, when we set the depth to be 100, the accuracy is close to 86%

Beyond that, we also find how to visualize the tree, in case that the tree with the depth of 100 is to large. Here we only visualize a small part.

图片包含 屏幕截图

描述已自动生成

1. Method 2: Random Forest

Random forest is an advanced ensemble learning method which use multiple trees and subsets of dataset. Here we import the model from sckit-learn and use GridSearchCV to tune the parameter set.

图片包含 屏幕截图, 文字

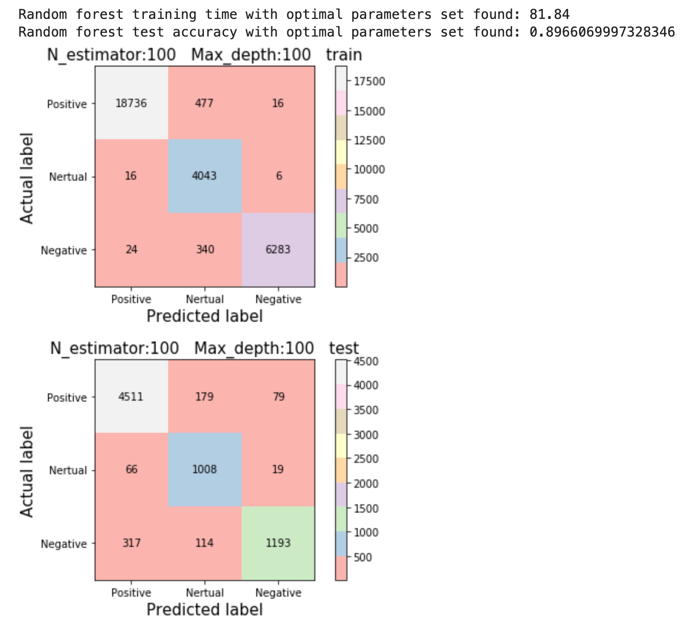
描述已自动生成

Here is the report:

图片包含 文字, 屏幕截图

描述已自动生成

Then we choose the best parameter set to train the model as well as testing it:



Here we see that it takes more time than single decision tree with the same max\_depth, but it shows higher accuracy.

One interesting we find is that in the cases of small max\_depth, the model does not perform better than single decision tree. We guess this is because the model suffers from bagging of attributes especially when constructing shallow trees (less judgements and low accuracy). Even though there is an effect of ‘ensemble’, the model cannot perform pretty well.

1. Method 3: Gradient Boosting Decision Tree

Besides using the above two method, we also try another advanced decision tree method to implement the task, which focuses on reducing the error to optimize the model.

Unfortunately, GDBT takes lots of time when we set the depth to be large. For this method, we just set depth to be 5 or 10.

图片包含 屏幕截图

描述已自动生成

And the result from gridSearchCV

图片包含 屏幕截图

描述已自动生成

Then we apply this parameter set:

图片包含 屏幕截图

描述已自动生成

It is obvious that GDBT takes significantly more time (687.85 as shown in figure) than the former two methods. However, what surprises us is that it performs much better than the former two methods in the same max-depth (67% vs 65% vs 73%).

**Part 2: Predicting ratings of the APP**

Inspired by some news that some APP companies hire people to rate them higher, we hope to use the other features to predict the rating of the APP and find a better model to help marketing research. In this part, we use two models, namely, linear regression and KNN model.

Here are all the packages that we used:

图片包含 文字, 屏幕截图

描述已自动生成

1. Data Preprocessing

Here are the variables in the file of GoogleAppStore.csv:

图片包含 屏幕截图

描述已自动生成

In this part, we did two steps. First is to deal with the missing data problem and the second is to transfer the type of the variables to numeric ones for KNN model training.

* 1. After loading the data, we find the summary of the missing values. Most of the missing values are Rating, which is our dependent variable. Thus, we tried to two methods to deal with the problem. One is to delete all the missing value (around 1400) and the other is to use median imputation trick to fill the blank. And we build df2 and df data framework respectively.

图片包含 屏幕截图

描述已自动生成图片包含 屏幕截图

描述已自动生成

After processing the code, we double checked the missing value and we can see there is no missing value left for both df and df2 data framework.

图片包含 屏幕截图

描述已自动生成

* 1. Transfer the type of the variables in to numerical ones

For the following steps, in order to process the data in the (KNN) machine learning algorithms, we need to first convert them from strings to numbers. Here are the brief steps that we use:

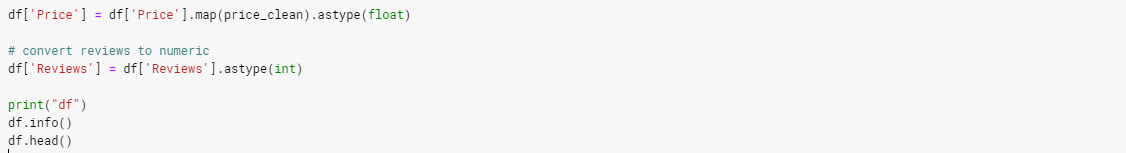
1. Delete the characteristics such as M, k, +, ... and so on
2. Keep the unit the same. We convert the size all into the units of M.
3. Transfer the category values into a set of dummies.
4. Transfer all the strings into int/float

图片包含 屏幕截图

描述已自动生成

图片包含 屏幕截图

描述已自动生成



Same operations are conducted to df2.

1. Training the model to predict the rating
   1. Split the data into training data and test data

In our project, we use Ranking as Y and other parameters as X. Here is the name and the order of the variables () and the rest of them are all category dummies.

X:

图片包含 文字, 报纸

描述已自动生成



We split the data into training set and testing set randomly with the ratio of 4:1 for both df and df2. Here is the code to do it:

图片包含 屏幕截图

描述已自动生成

* 1. Linear Regression Model with Machine Learning.

In this part, we use package sklearn to simulate the linear regression model of both datasets.

图片包含 屏幕截图

描述已自动生成

And here is the summary of the prediction scores for df and df2. As df has more data and there is a possibility of overestimation of the model(as we use median to imuputate), df-score is reasonable to be a bit higher than df2.

|  |  |  |
| --- | --- | --- |
|  | Df | df2 |
| Scores | 0.8909545779324055 | 0.8895438549421302 |

And the coefficients of the models are

Array1([ 1.50095393e-08, 4.04776442e-04, -1.00368831e-10, 5.64456313e-02,

-8.52792123e-04, 1.73335607e-04, 1.77019458e-01, 3.92565141e+00,

3.61221886e+00, 3.56379021e+00, 3.39633660e+00, 3.18165777e+00,

2.91597366e+00, 2.76091902e+00, 2.45396653e+00, 2.59238334e+00,

2.22738048e+00, 2.34638672e+00, 6.53794864e-01, 1.83008306e+00,

1.64743068e+00, 8.73540109e-01, 1.55862330e+00, 1.35762081e+00,

1.28975414e+00, 9.53192024e-01, -2.04679535e+00, 5.10878464e-01,

-1.65735618e+00, -1.01738305e+00, -6.24581691e-01, -7.50583416e-02,

-9.11517948e-01, 2.17817112e-01, 3.74920930e-01, -1.95027140e-01,

-7.15134488e-01, -4.85635995e-01, -1.65703228e+00, -1.26279713e+00])

Array2 ([ 1.32047150e-08, 9.57315902e-04, 1.62036632e-11, 6.49847289e-02,

-8.22135046e-04, 2.15862994e-03, 1.75212427e-01, 3.93711052e+00,

3.65375638e+00, 3.62417650e+00, 3.34258626e+00, 3.08019270e+00,

2.87869358e+00, 2.73823847e+00, 2.33950757e+00, 2.59837083e+00,

2.26402765e+00, 2.37100539e+00, 6.63118082e-01, 1.84486859e+00,

1.54567372e+00, 8.78664008e-01, 1.56672156e+00, 1.32626721e+00,

1.24208863e+00, 9.07304812e-01, -2.04944264e+00, 4.59255172e-01,

-1.64986969e+00, -9.78979564e-01, -5.94404928e-01, -3.74283075e-02,

-9.21340102e-01, 2.54165027e-01, 4.12564411e-01, -2.06124415e-01,

-7.30563216e-01, -4.89426779e-01, -1.64892974e+00, -1.23270231e+00])

We can see from the coefficients that the categories play an important role in the rating of the APP. It could because people using specific APP has specific personality (strict or casual); or it could because certain kinds of APP are easier to improve.

* 1. K-Nearest Neighbors (KNN)

We first look at the 17 closest neighbors and compare the accuracy with two model:

图片包含 屏幕截图

描述已自动生成

|  |  |  |
| --- | --- | --- |
|  | Df | df2 |
| Scores | 0.8959 | 0.8922 |

The similar argument could be conducted as the above.

Then, we conducted several other numbers of neighbors to re-run the code. Here is the graph of their scores:

df dataset:

图片包含 文字, 地图

描述已自动生成

df2 dataset:

图片包含 文字, 地图

描述已自动生成

* 1. Compare different models

Linear regression is an example of a parametric approach because it assumes a linear functional form for f(X). On the other hand, K-Nearest Neighbors (KNN), which is a non-parametric method.

Linear regression methods

1. Advantages
   1. Easy to fit. One needs to estimate a small number of coefficients.
   2. Easy to interpret the relationship.
2. Disadvantages
   1. They make strong assumptions about the form of f(X).
   2. Suppose the true relationship is far from linear, then the resulting model will provide a poor fit to the data, and any conclusions drawn from it will be suspect.

KNN models

1. Advantages
   1. They do not assume an explicit form for f(X), providing a more flexible approach.
2. Disadvantages
   1. They can be often more complex to understand and interpret
   2. If there is a small number of observations per predictor, then parametric methods then to work better.

Also, from the result of our code, we could find that the results are kind of similar. We would recommend to use dataset df and linear regression model to interpret which factors are more important to raise the ranking; we would recommend KNN model for model prediction.

**Workload Distribution**

Part 1: CHEN, Yifei.

Part 2: LI, Siqi.

Video: CHEN, Yifei; LI, Siqi.

Report: CHEN, Yifei; LI, Siqi.