Rachid LAMJOUN
PG AI Program

Building user-based recommendation model for Amazon

PG AI – Machine Learning Project:

Project2: Building user-based recommendation model for Amazon

Writeup

We have the subject of building an automatic learning model that provides the ratings for each of the users to the movies offered by the Amazon platform.

The model will be based on a dataset which brings a certain number of rating feedback from users who present their appreciation about movies. Obviously not all movies are rated by customers and that is the role of the model to be created to propose relevant ratings.

Before beginning the construction of the model and to become familiar with the dataset we will try to deal with the following points: the DataFrame of data is represented by **df**

1. Which movies have maximum views ?

```
#-----------------
#--max views
#-----------------
# Max views
Sr=(df.describe().T['count']== df.describe().T['count'].max())
list(Sr[Sr==True].index)
```

```
['Movie127']
```

2. Which movies have maximum ratings ?

```
# sum column values
obj_t=df[df.columns[1:]].sum().sort_values(ascending=False) # except the first column user_id
obj_t
```

```
Movie127    2313.0
Movie140     578.0
Movie16      320.0
Movie103     272.0
Movie29      243.0
             ...
Movie54        1.0
Movie116       1.0
Movie115       1.0
Movie55        1.0
Movie1         1.0
Name: count, Length: 206, dtype: float64
```

```
obj_t_fr=pd.DataFrame(obj_t,columns=['rating_cumul'])
list(obj_t_fr[obj_t_fr['rating_cumul']==obj_t.max()].index)
```

```
['Movie127']
```

3. Define the top 5 movies with the maximum ratings.

```
#--------------------
#--- Five first with max rating
#--------------------
list((df[df.columns[1:]].sum().sort_values(ascending=False)[:5]).index)
```

```
['Movie127', 'Movie140', 'Movie16', 'Movie103', 'Movie29']
```

4. What is the average rating for each movie ?

```
obj_t=df[df.columns[1:]].mean().sort_values(ascending=True).round(2)
obj_t[5:18]
```

```
Movie154    1.00
Movie69     1.00
Movie90     1.83
Movie59     2.00
Movie53     2.00
Movie3      2.00
Movie73     2.00
Movie171    2.00
Movie159    3.00
Movie20     3.00
Movie26     3.00
Movie83     3.00
Movie64     3.00
dtype: float64
```

5. Define the top 5 movies with the least audience.

```
#----------------
#--least views (audience)
#----------------
series_t=df.describe().T['count'].sort_values(ascending=True)[:5]
list(series_t.index)
```

```
['Movie1', 'Movie71', 'Movie145', 'Movie69', 'Movie68']
```

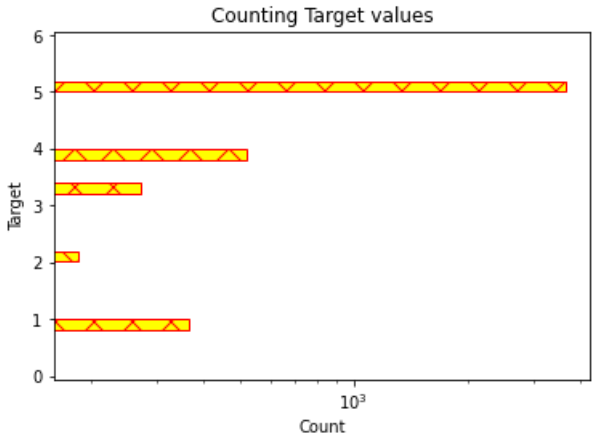
To build our ratings proposal model, we need to **prepare a working dataset** that will be built in the preprocessing phase and will respond to the following points:

- The Target column is added.
- Each value > 0 is copied to the Target column and replaced with 1.
- All nan values will be replaced by 0
- The ‘user_id’ column will be numeric coded which embodies the link between the ratings of a given user.

The call to the **get_df_ready_for_fit()** function will perform these operations and generate the working dataframe for the perform of our models.

```
df_2_vc=df_2['Target'].value_counts()
df_2_vc
```

```
5.0    3659
4.0     521
1.0     363
3.0     272
2.0     185
Name: Target, dtype: int64
```




```
# percentage
df_2_vc/df_2_vc.sum()*100
```

```
5.0    73.18
4.0    10.42
1.0     7.26
3.0     5.44
2.0     3.70
Name: Target, dtype: float64
```

The Target 5 is over-represented with more than **73%** which poses a certain problem for us for the construction of our models. This imbalance of data will be concretely observed by making predictions by the **sklearn.svm.SVC** model of the training data.

```
from sklearn.svm import SVC
model_svc=SVC(random_state=0)
model_svc.fit(X_train,y_train)
ypred=model_svc.predict(X_train)
list(set(ypred))
```

```
[5.0]
```

These predictions are reduced to a single value of 5.

To balance the data, we will use two techniques, and depending on the case, we will opt for which models are the best. The two techniques are two techniques are **RandomOverSampler**() and **SMOTE**().

In the following we will try to analyze these two main models **XGBClassifier (XGBoost)** and **RandomForestClassifier.**

The Best model obtained is a RandomForestClassifier, and the oversampling was provided by RandomOverSampler() :

```
------------:::::::::::::::-------------
============
RandomForestClassifier
============
0.584
[[  8   2   2   9  47]
 [  5   1   3   4  28]
 [  8   2   1   7  39]
 [  6   5   3   9  78]
 [ 34  23  44  67 565]]
              precision    recall  f1-score   support

         1.0       0.13      0.12      0.12        68
         2.0       0.03      0.02      0.03        41
         3.0       0.02      0.02      0.02        57
         4.0       0.09      0.09      0.09       101
         5.0       0.75      0.77      0.76       733

    accuracy                           0.58      1000
   macro avg       0.20      0.20      0.20      1000
weighted avg       0.57      0.58      0.58      1000

-----------:::::::::::::::-------------
```

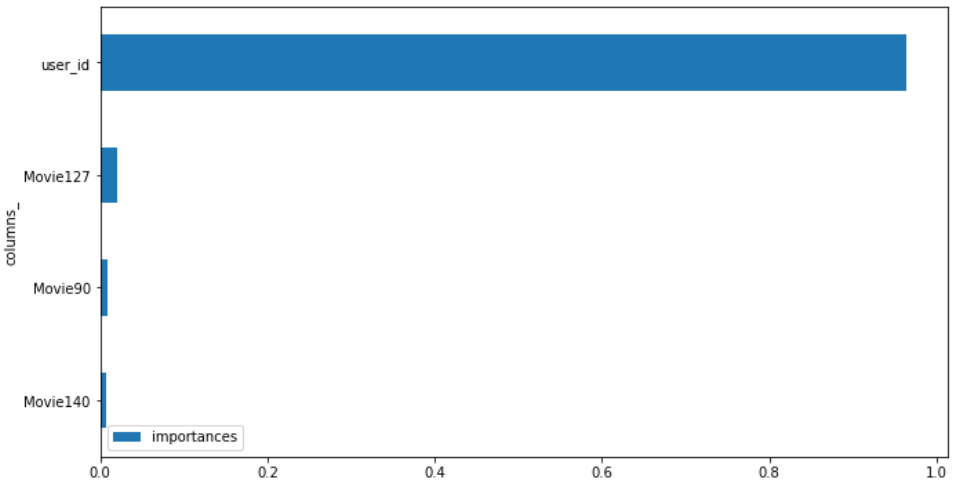
By analyzing the feature_importances_ and by restriction, only to these 4 features [user_id, Movie127, Movie140, Movie90] we obtain a slightly better result :

```
============
RandomForestClassifier
============
0.599
[[  9   3   2   9  45]
 [  4   1   4   3  29]
 [  7   1   2   5  42]
 [  5   5   5  15  71]
 [ 39  24  33  65 572]]
              precision    recall  f1-score   support

         1.0       0.14      0.13      0.14        68
         2.0       0.03      0.02      0.03        41
         3.0       0.04      0.04      0.04        57
         4.0       0.15      0.15      0.15       101
         5.0       0.75      0.78      0.77       733

    accuracy                           0.60      1000
   macro avg       0.22      0.22      0.22      1000
weighted avg       0.58      0.60      0.59      1000

----------:::::::::::::::-------------
```

Despite several attempts to search for better parameters by **RandomizedSearchCV** () no better performing model has been found.

with inconsistent data it can be interesting to approach the recommendation of movies by a binary approach that we think can build more meaningful models. In the last paragraph of this talk we will explore this approach.

But before we go any further, let's look at this graph of "feature_importance_", where we used the threshold 0.005:

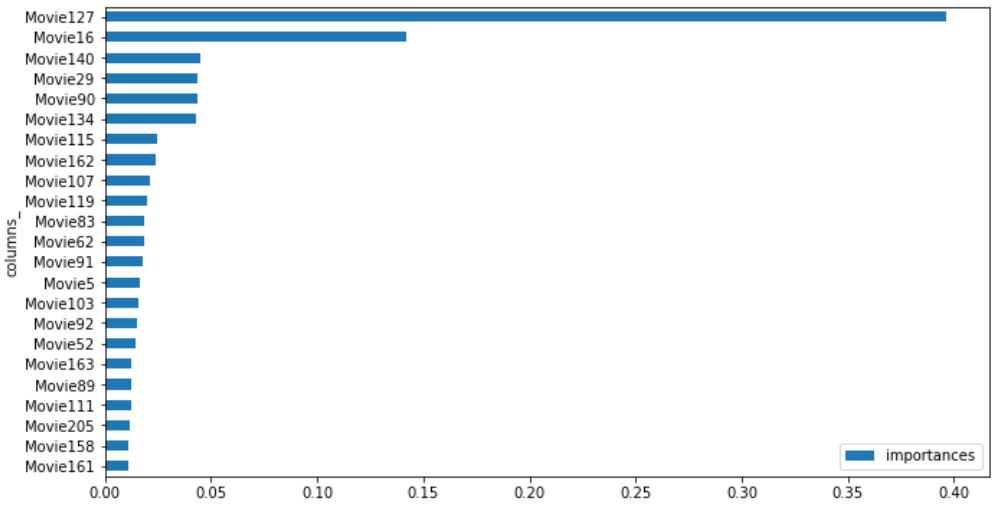


We see that the user_id feature plays a main role for our model. It would be interesting to see if we can build a model that will only use information from movie ratings and that will have

at least the performance of the previous model. But unfortunately, we were only able to build the following model:

```
============
RandomForestClassifier
============
0.454
[[ 51   0   0   0  17]
 [ 32   1   0   3   5]
 [ 32   0   0   5  20]
 [ 46   1   1  20  33]
 [299   5   1  46 382]]
              precision    recall  f1-score   support

         1.0       0.11      0.75      0.19        68
         2.0       0.14      0.02      0.04        41
         3.0       0.00      0.00      0.00        57
         4.0       0.27      0.20      0.23       101
         5.0       0.84      0.52      0.64       733

    accuracy                           0.45      1000
   macro avg       0.27      0.30      0.22      1000
weighted avg       0.65      0.45      0.51      1000

-----------::::::::::::::::::::::::::::::
```

After an oversampling by the SMOTE() method and restricting to the 23 features of the following graph features_importances_ with the threshold 0.01.



We note that the movie Movie127 is essential in the approach of the model.

As we have already mentioned, let's look at another approach to remedy the inconsistency of data that makes sense for our case study. It would be interesting for a user who has been able to give ratings to movies to find other movies that are likely to interest him. So, let's try to build a more efficient model than those already built, and which answers this question.

Recommendation perspective with binary Target:

We think it would be wise from a customer movie recommendation perspective, to create a model that will suffer less from the inconsistency of the data we have, considering that the predicted 4 and 5 ratings may mean that the movie may be recommendable.
After adapting our dataset with a binary Target, we had the following situation:

```
code = {1.0:0,2.0:0,3.0:0,4.0:1,5.0:1}
df_3['Target'] =df_3['Target'].replace(code)  #.astype(np.float64)
df_3['Target'].value_counts()
```

```
1.0    4180
0.0     820
Name: Target, dtype: int64
```

```
df3_t=df3_t_c/df3_t_c.sum()*100
df3_t
```

```
1.0    83.6
0.0    16.4
Name: Target, dtype: float64
```

The Target representing the positive recommendation represents more than 83% which represents a true data imbalance. Moreover, we see it with the XGBClassifier model gives interesting results:

```
============
XGBoost
============
0.836
[[  2 164]
 [  0 834]]
              precision    recall  f1-score   support

         0.0       1.00      0.01      0.02       166
         1.0       0.84      1.00      0.91       834

    accuracy                           0.84      1000
   macro avg       0.92      0.51      0.47      1000
weighted avg       0.86      0.84      0.76      1000

----------::::::::::::::-------------
```

But unfortunately, the metrics are very unbalanced and especially for class 0.

The oversampling by RandomOverSampler() operation has better results than by the SMOTE() method:

```
============
RandomForestClassifier
============
0.735
[[ 33 133]
 [132 702]]
              precision    recall  f1-score   support

         0.0       0.20      0.20      0.20       166
         1.0       0.84      0.84      0.84       834

    accuracy                           0.73      1000
   macro avg       0.52      0.52      0.52      1000
weighted avg       0.73      0.73      0.73      1000

----------:::::::::::::::-------------
```

As we hoped we were able to build a more efficient model with more balanced metrics and an interesting prediction rate for the class representing the recommendation.

Let's see if it is possible to optimize this model.
After looking at the 'feature_importance_' of this model from RandomOverSampler() and using the 0.0008 threshold we were able to have a model of the **same order of performance** but restricting only to **53 features instead of 207**. We can even restrict ourselves to these seven features [user_id, Movie127, Movie140, Movie90, Movie29, Movie52, Movie103] with the following performance (slightly less) :

```
============
RandomForestClassifier
============
0.722
[[ 33 133]
 [145 689]]
              precision    recall  f1-score   support

         0.0       0.19      0.20      0.19       166
         1.0       0.84      0.83      0.83       834

    accuracy                           0.72      1000
   macro avg       0.51      0.51      0.51      1000
weighted avg       0.73      0.72      0.73      1000

----------:::::::::::::::-------------
```

Despite several attempts to search for better parameters by **GridSearchCV**() no better performing model has been found.

Looking at this figure of "**feature_importancs_",** we realize that "**user_id**" plays a very important role in the predictions of our model. We then asked ourselves the questions is that

the link between the ratings of the movies relating to a given client is so important? and isn't there a good model that is generated only by the ratings of each of the movies ?

This time we used oversampling by the SMOTE() module to have a more efficient model:

```
============
XGBoost
============
0.601
[[118  48]
 [351 483]]
              precision    recall  f1-score   support

         0.0       0.25      0.71      0.37       166
         1.0       0.91      0.58      0.71       834

    accuracy                           0.60      1000
   macro avg       0.58      0.64      0.54      1000
weighted avg       0.80      0.60      0.65      1000

-----------::::::::::::::::-------------
============
RandomForestClassifier
============
0.594
[[119  47]
 [359 475]]
              precision    recall  f1-score   support

         0.0       0.25      0.72      0.37       166
         1.0       0.91      0.57      0.70       834

    accuracy                           0.59      1000
   macro avg       0.58      0.64      0.54      1000
weighted avg       0.80      0.59      0.65      1000

-----------::::::::::::::::-------------
```

By analyzing the “feature_importancs_” of the RandomForestClassifier model, restricting ourselves to the following three Movies [Movie127, Movie140, Movie90] gives a little better models:

```
============
XGBoost
============
0.604
[[115  51]
 [345 489]]
              precision    recall  f1-score   support

         0.0       0.25      0.69      0.37       166
         1.0       0.91      0.59      0.71       834

    accuracy                           0.60      1000
   macro avg       0.58      0.64      0.54      1000
weighted avg       0.80      0.60      0.65      1000

-----------:::::::::::::::::-------------
============
RandomForestClassifier
============
0.604
[[115  51]
 [345 489]]
              precision    recall  f1-score   support

         0.0       0.25      0.69      0.37       166
         1.0       0.91      0.59      0.71       834

    accuracy                           0.60      1000
   macro avg       0.58      0.64      0.54      1000
weighted avg       0.80      0.60      0.65      1000

-----------:::::::::::::::::-------------
```