网络新技术实验3报告

实验源码：`lab3代码.ipynb`

实验环境：vscode + colab

数据解释：

`movies.dat`的数据如下

```
1::Toy Story (1995)::Animation|Children's|Comedy
2::Jumanji (1995)::Adventure|Children's|Fantasy
3::Grumpier Old Men (1995)::Comedy|Romance
4::Waiting to Exhale (1995)::Comedy|Drama
5::Father of the Bride Part II (1995)::Comedy
6::Heat (1995)::Action|Crime|Thriller
7::Sabrina (1995)::Comedy|Romance
8::Tom and Huck (1995)::Adventure|Children's
9::Sudden Death (1995)::Action
10::GoldenEye (1995)::Action|Adventure|Thriller
```

`ratings.dat`的数据如下：

```
UserID::MovieID::Rating::Timestamp

- UserIDs range between 1 and 6040
- MovieIDs range between 1 and 3952
- Ratings are made on a 5-star scale (whole-star ratings only)
- Timestamp is represented in seconds since the epoch as returned by
time(2)
- Each user has at least 20 ratings



1::1193::5::978300760
```

```
1::661::3::978302109
1::914::3::978301968
1::3408::4::978300275
1::2355::5::978824291
1::1197::3::978302268
1::1287::5::978302039
1::2804::5::978300719
1::594::4::978302268
1::919::4::978301368
```

`users.dat`的数据如下:

```
1::F::1::10::48067
2::M::56::16::70072
3::M::25::15::55117
4::M::45::7::02460
5::M::25::20::55455
6::F::50::9::55117
7::M::35::1::06810
8::M::25::12::11413
9::M::25::17::61614

User information is in the file "users.dat" and is in the following
format:

UserID::Gender::Age::Occupation::Zip-code

All demographic information is provided voluntarily by the users and
is
not checked for accuracy.  Only users who have provided some
demographic
information are included in this data set.

- Gender is denoted by a "M" for male and "F" for female
- Age is chosen from the following ranges:
```

思路：

1. 导入数据集
2. 多份数据合并到一个数据集内
3. 划分训练集和预测集
4. 进行评分预测，计算precision

实验报告中仅包含核心代码，完整代码见

import需要使用的库

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g.
pd.read_csv)
import seaborn as sns

import matplotlib
import matplotlib.pyplot as plt

import os
for dirname, _, filenames in os.walk('data'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

导入数据集

```
df_m = pd.read_csv("/content/data/movielens/movies.dat",
engine='python', sep='::', names=["MovieID", "Title",
"Genres"],encoding='ISO-8859-1')

df_m.head()

df_r = pd.read_csv("/content/data/movielens/ratings.dat",
engine='python', sep='::', names=["UserID", "MovieID", "Rating",
"Timestamp"],encoding='ISO-8859-1')
df_r.head()

df_u = pd.read_csv("/content/data/movielens/users.dat",
engine='python', sep='::', names=["UserID", "Gender", "Age",
"Occupation", "Zip-code"],encoding = 'ISO-8859-1')
df_u.head()
```

```
data/movielens/ratings.dat
data/movielens/movies.dat
data/movielens/users.dat
```

合并数据集

将`df_m`,`df_r`,`df_u`几项数据合并

```
df_merged1 = df_m.merge(df_r, how='outer')
df_merged1.head()

df_merged2 = df_u.merge(df_r, how='inner')
df_merged2.head()

df_merged3 = df_merged1.merge(df_merged2, how='inner')
df_merged3.head()

df_merged3.UserID = df_merged3.UserID.astype(int)
df_merged3.Rating = df_merged3.Rating.astype(int)
df_merged3.head()
```

合并完的`master_data`如图所示:

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Age	Occupation	Zip-code
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268.0	F	1	10	48067
1	1	Toy Story (1995)	Animation|Children's|Comedy	6	4	978237008.0	F	50	9	55117
2	1	Toy Story (1995)	Animation|Children's|Comedy	8	4	978233496.0	M	25	12	11413
3	1	Toy Story (1995)	Animation|Children's|Comedy	9	5	978225952.0	M	25	17	61614
4	1	Toy Story (1995)	Animation|Children's|Comedy	10	5	978226474.0	F	35	1	95370

检查数据，按照User.id进行排序

```
df_merged3.shape
df_merged3.sort_values(by=['UserID'], ascending=True)
```

```
df_merged3.shape
df_merged3.sort_values(by=['UserID'], ascending=True)
```

Python

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Age	Occupation	Zip-code
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268.0	F	1	10	48067
73691	2028	Saving Private Ryan (1998)	Action|Drama|War	1	5	978301619.0	F	1	10	48067
112908	3186	Girl, Interrupted (1999)	Drama	1	4	978300019.0	F	1	10	48067
39160	1193	One Flew Over the Cuckoo's Nest (1975)	Drama	1	5	978300760.0	F	1	10	48067
36477	1097	E.T. the Extra-Terrestrial (1982)	Children's|Drama|Fantasy|Sci-Fi	1	4	978301953.0	F	1	10	48067
...	...	...	...	...	...	...	...	...	...	...
81022	2248	Say Anything... (1989)	Comedy|Drama|Romance	858	3	975344295.0	M	35	14	10013
88876	2453	Boy Who Could Fly, The (1986)	Drama|Fantasy	858	5	975343822.0	M	35	14	10013
47356	1265	Groundhog Day (1993)	Comedy|Romance	858	4	975344235.0	M	35	14	10013
46742	1259	Stand by Me (1986)	Adventure|Comedy|Drama	858	4	975344219.0	M	35	14	10013
81775	2279	Urban Legend (1998)	Horror|Thriller	858	3	975344820.0	M	35	14	10013

建立一个叫做master_data的数据集，并且将性别一项从['F','M']改为[0,1]

```
master_data = df_merged3[['UserID', 'MovieID', 'Title', 'Rating',
'Genres', 'Zip-code', 'Gender', 'Age', 'Occupation', 'Timestamp']]
master_data.head()
master_data['Gender'].replace(['F','M'],[0,1],inplace=True)
```

```
master_data.sort_values(by=['MovieID'], ascending=True)
```

Python

	UserID	MovieID	Title	Rating	Genres	Zip-code	Gender	Age	Occupation	Timestamp
0	1	1	Toy Story (1995)	5	Animation|Children's|Comedy	48067	0	1	10	978824268.0
199	613	1	Toy Story (1995)	5	Animation|Children's|Comedy	10562	1	35	20	975812101.0
198	611	1	Toy Story (1995)	4	Animation|Children's|Comedy	20715	1	35	0	977155870.0
197	610	1	Toy Story (1995)	5	Animation|Children's|Comedy	77025	1	25	4	975861277.0
196	606	1	Toy Story (1995)	5	Animation|Children's|Comedy	49507	0	1	10	975869266.0
...	...	...	...	...	...	...	...	...	...	...
132931	486	3952	Contender, The (2000)	3	Drama|Thriller	91367	1	56	0	976215698.0
132932	524	3952	Contender, The (2000)	4	Drama|Thriller	91320	1	18	0	976173386.0
132933	531	3952	Contender, The (2000)	4	Drama|Thriller	22206	0	18	14	983563881.0
132919	319	3952	Contender, The (2000)	3	Drama|Thriller	33436	0	50	6	989110784.0
132968	856	3952	Contender, The (2000)	4	Drama|Thriller	02453	0	45	6	975345466.0

132969 rows × 10 columns

用one-hot对类型相进行编码

效果如下：

```
   Action  Adventure  Animation  Children's  Comedy  Crime  Documentary  \
0       0          0          1           1       1      0            0
1       0          0          1           1       1      0            0
2       0          0          1           1       1      0            0
3       0          0          1           1       1      0            0
4       0          0          1           1       1      0            0

   Drama  Fantasy  Film-Noir  Horror  Musical  Mystery  Romance  Sci-Fi  \
0      0        0          0       0        0        0        0       0
1      0        0          0       0        0        0        0       0
2      0        0          0       0        0        0        0       0
3      0        0          0       0        0        0        0       0
4      0        0          0       0        0        0        0       0

   Thriller  War  Western
0         0    0        0
1         0    0        0
2         0    0        0
3         0    0        0
4         0    0        0
```

将种类项加入到master_data项，

```
master_features = pd.merge(md_small, one_hot_genres, left_index=True,
right_index=True)
master_features.head()

X_feature = md_small.drop(['Zip-code'], axis=1)
```

经过处理以后，feature的数据如图所示：

	MovieID	Rating	Zip-code	Gender	Age	Occupation	Action	Adventure	Animation	Children's	...	Fantasy	Film-Noir	Horror	Musical	Mystery	Romance	Sci-Fi	Thriller	War	Western
0	1	5	48067	0	1	10	0	0	1	1	...	0	0	0	0	0	0	0	0	0	0
1	1	4	55117	0	50	9	0	0	1	1	...	0	0	0	0	0	0	0	0	0	0
2	1	4	11413	1	25	12	0	0	1	1	...	0	0	0	0	0	0	0	0	0	0
3	1	5	61614	1	25	17	0	0	1	1	...	0	0	0	0	0	0	0	0	0	0
4	1	5	95370	0	35	1	0	0	1	1	...	0	0	0	0	0	0	0	0	0	0

移除了`zip-code`,因为对训练没什么帮助。在训练的特征中选择了职业，年龄，性别作为最重要的特征。

训练

在训练时，将数据的75%作为训练集，25%作为预测集

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test =
train_test_split(X_feature_small_trimmed,Y_target,random_state=1)

from sklearn.linear_model import LogisticRegression
```

Logistic regression最适合用于预测分类数据，需要对训练数据进行Logistic regression，数据集在达到最大迭代次数时不断抛出非收敛错误。 可以通过如下方式增加代码最大迭代。

```
#logreg = LogisticRegression(solver='lbfgs',class_weight='balanced',
max_iter=100000)
logreg = LogisticRegression(max_iter=100000)

logreg.fit(x_train,y_train)

y_pred = logreg.predict(x_test)
```

使用precision验证训练的结果

```
from sklearn import metrics
print('precision is:')
metrics.precision_score(y_test,y_pred, average="micro")
```

得到数据:

```
precision is:

0.3545108005082592
```

> *Referrence : https://www.kaggle.com/code/srinag/movielens-rating-prediction-modeling*