关于大作业批改及评分：

- 批改工作量确实很大
- 难免有疏漏和判断标准不一的问题
- 文无第一，武无第二

Python 程序设计与数据科学导论大作业

单人作业一：Movielens1M 分析

徐俐欣 2100015855

光华管理学院

2023.05.04

目 录

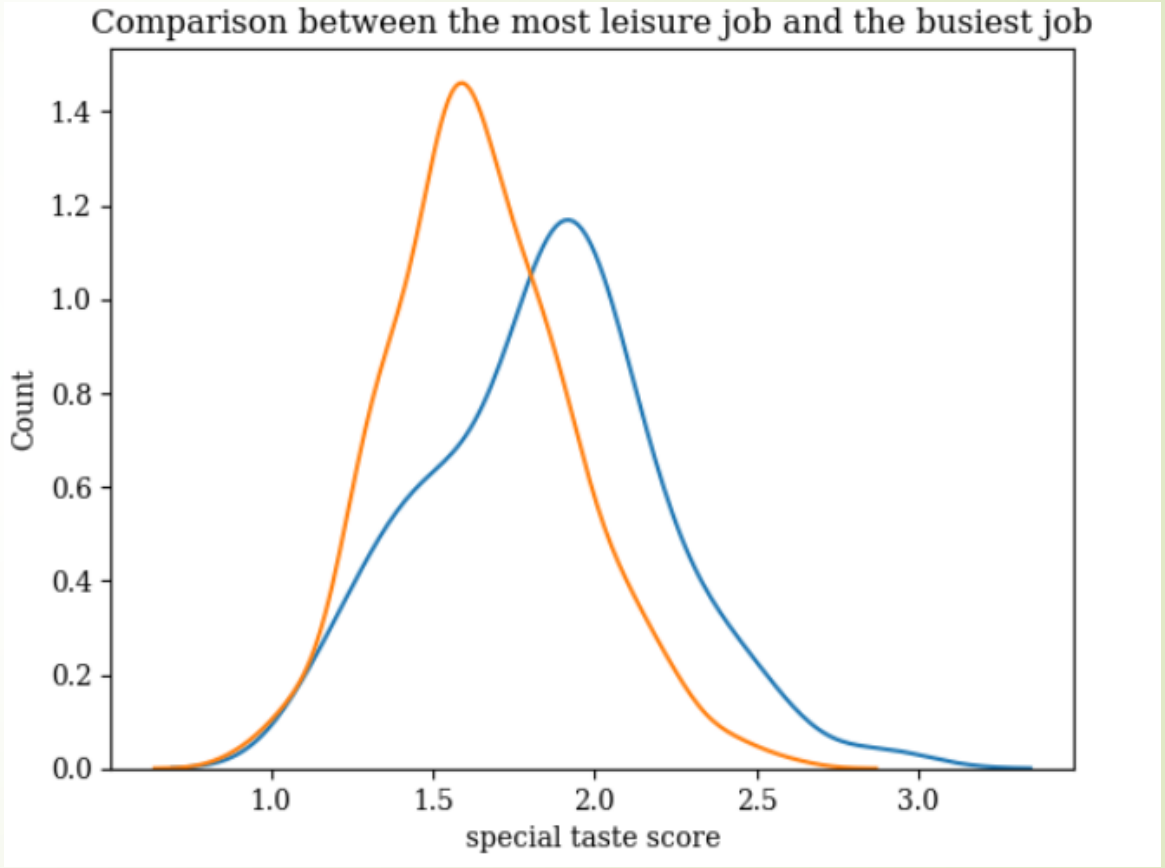
2000013064 张泽楷

```
# 研究有独特品味的都是哪些人
data = data_orig
movie_popularity = data.groupby('movie_id')['user_id'].count()
movie_num = len(movies)

# 处理数据，定义"独特品味"的值
data['movie_specialty'] = data['movie_id'].apply(lambda x: np.log(movie_num/movie_popularity.loc[x]))
user_special_taste = data.groupby('user_id')['movie_specialty'].mean().reset_index(name="special_taste_scor
user_special_taste = pd.merge(user_special_taste, users, on='user_id')

age_list = users['age_desc'].unique()
gender_list = ['F','M']
occ_list = users['occ_desc'].unique()
```

```
14
15  # 人工评测：最有独特品味职业和最空闲职业的命中率
16  top6hit = (4+4+4+4+3)/5
17  print(f"{top6hit}/6")
```

	occ_desc	special_taste_score
0	retired	1.862514
1	writer	1.853165
2	unemployed	1.846372
3	K-12 student	1.833271
4	artist	1.813899
5	customer service	1.805580
6	clerical/admin	1.802295
7	other or not specified	1.799106
8	self-employed	1.797269
9	tradesman/craftsman	1.791920
10	academic/educator	1.754723
11	doctor/health care	1.736265
12	lawyer	1.732230
13	farmer	1.730894
14	homemaker	1.724964
15	executive/managerial	1.701489
16	college/grad student	1.694023
17	sales/marketing	1.687978
18	scientist	1.674935
19	technician/engineer	1.650981
20	programmer	1.635625

3.8/6



用神经网络编码模型实现年龄预测

```
# 1900012725-陈奕奇
# 首先，我们构建一下数据集
user_movie_matrix_dataframe = ratings.pivot_table('rating', index='user_id', colu
# print(user_movie_matrix_dataframe)
user_movie_matrix_dataframe = pd.merge(left=user_movie_matrix_dataframe, right=us
valid_mask = user_movie_matrix_dataframe.apply(lambda x: (x != 0).sum () > 100,
# print(valid_mask)
user_movie_matrix_dataframe_valid = user_movie_matrix_dataframe.loc[valid_mask]
# print(user_movie_matrix_dataframe_valid)
# print(user_movie_matrix_dataframe_valid.shape)
X_test = user_movie_matrix_dataframe_valid.sample(frac=0.2, random_state=42)
X_train = user_movie_matrix_dataframe_valid.drop(X_test.index)
X_train_np = np.array(X_train)  # 为之后做神经网络准备数据
X_test_np = np.array(X_test)

age_dict = {'Under 18':0, '18-24':1, '25-34':2, '35-44':3, '45-49':4, '50-55':5,
age_dict_np = {'Under 18':[0], '18-24':[1], '25-34':[2], '35-44':[3], '45-49':[4]
gender_dict = {'F':0, 'M':1}
gender_dict_np = {'F':[0], 'M':[1]}  # 这里是为了转化为one-hot向量，以供后面深度
y_age_test = X_test['age_desc'].map(age_dict)
y_age_train = X_train['age_desc'].map(age_dict)
from sklearn.preprocessing import MultiLabelBinarizer
mlb = MultiLabelBinarizer()
y_age_test_np = np.array(mlb.fit_transform(X_test['age_desc'].map(age_dict_np)))
y_age_train_np = np.array(mlb.fit_transform(X_train['age_desc'].map(age_dict_np))
```

Ordinal Regression with Multiple Output CNN for Age Estimation (Zhenxing Niu1 2016 cvpr)

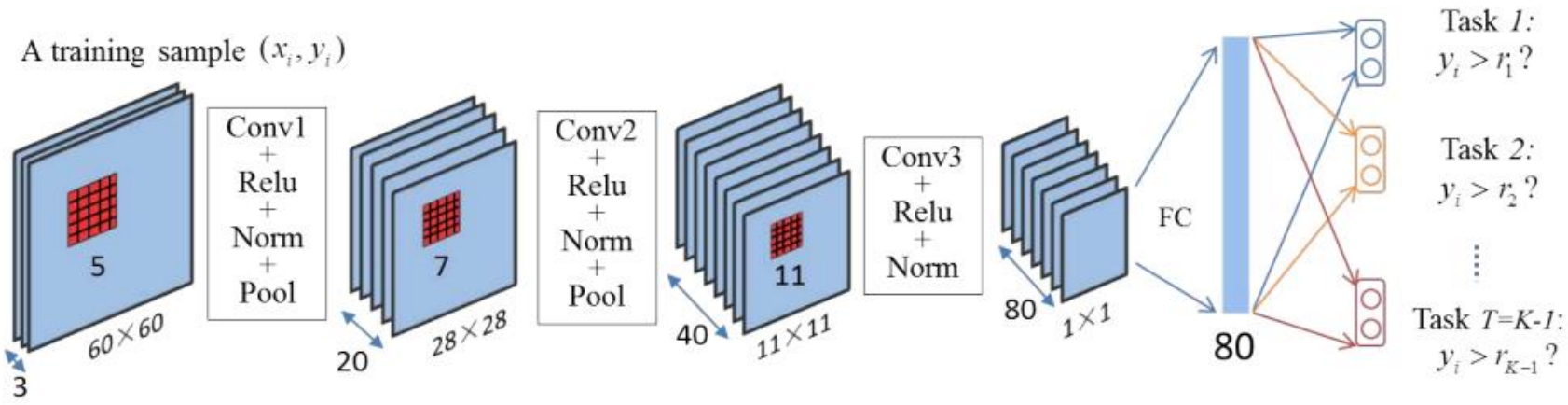
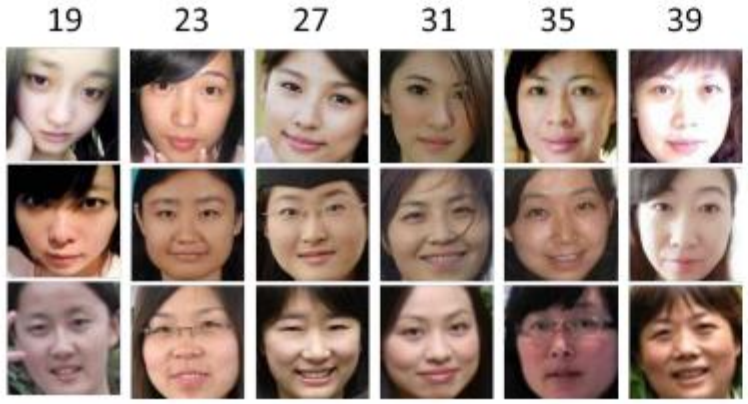
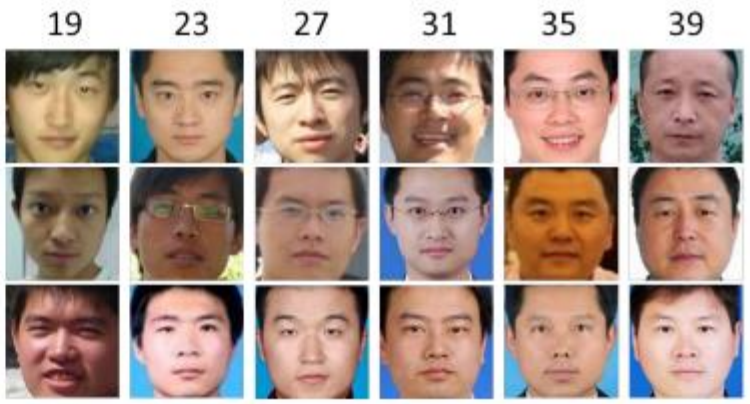


Figure 2. The architecture of the proposed Multiple Output CNN



(a) female



(b) male

时间序列分析基础-C16

胡俊峰 北京大学

2023/04/08

内容摘要

- 时间序列问题简介
- 线性回归与预测模型
- 序列成分分解基础
- 平稳序列与序列平稳化
- 序列谱分析与滤波

时间序列：随时间变化的（一组）数据观测

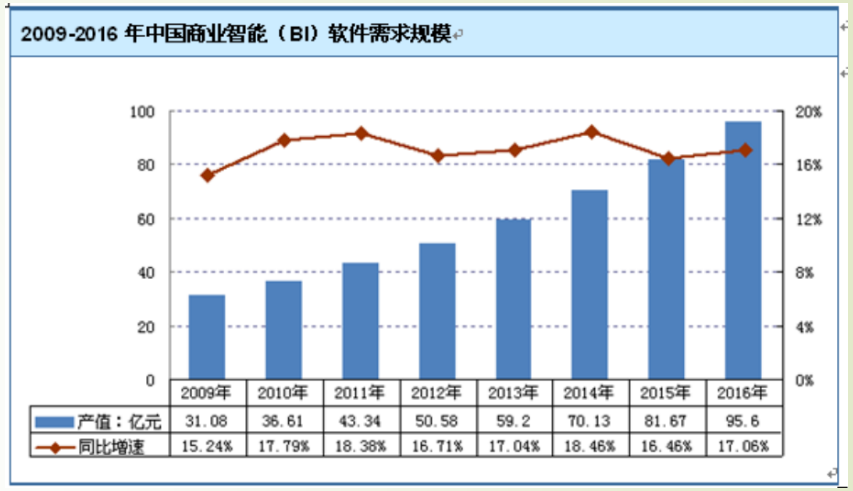
- 一般情况下，时间间隔是一致的，数据观测在时间序列上是连续的。特殊情况下也可以有间隔
- 时间序列可以来源于一组离散的由 时间索引的数据，也可能来源于连续信号的离散时间点采样

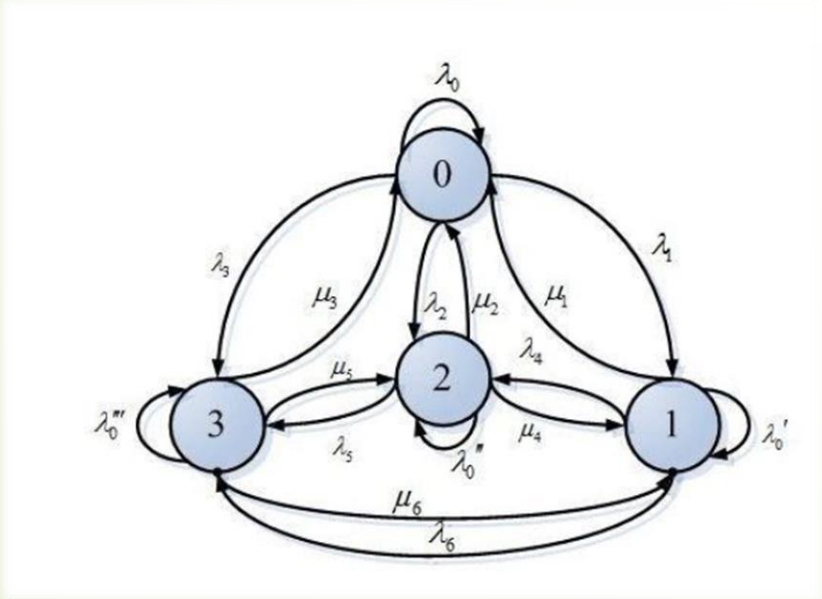
时间序列分析（TSA）的主要研究领域

- 趋势预测（回归）、成分分析，滤波
- 实时交互与控制
- 序列信号模式识别

趋势预测（回归）分析

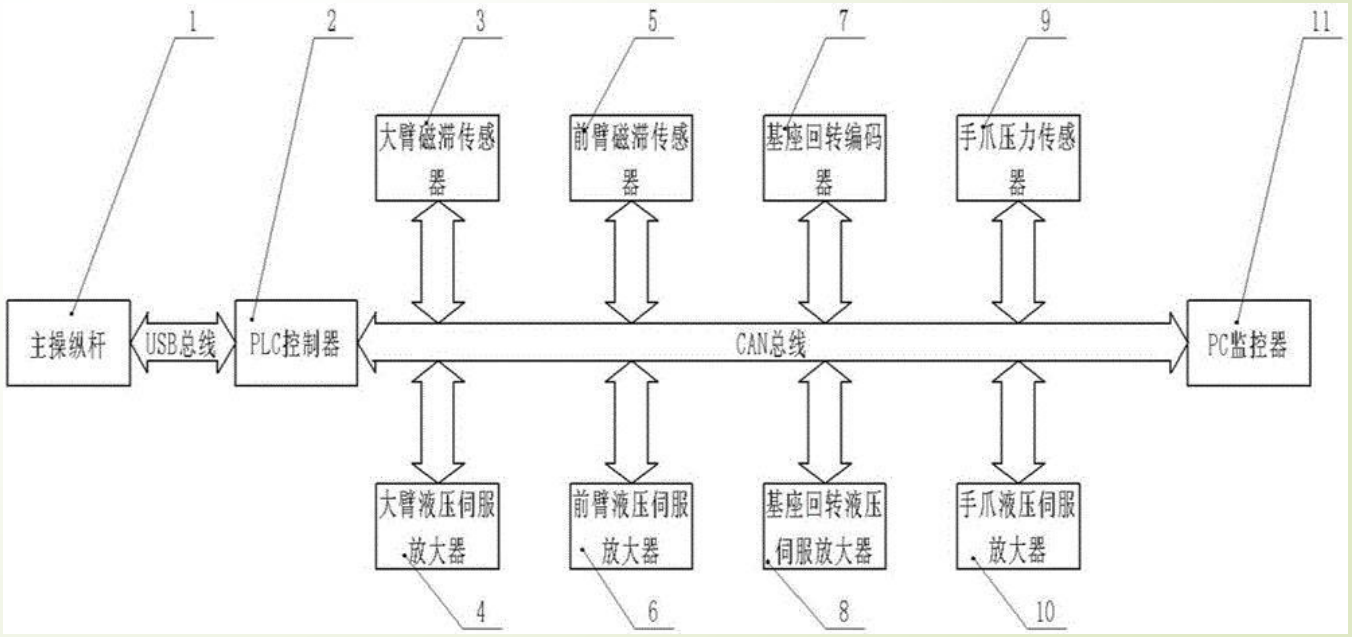
- 需求（流量）预测
- 金融量化分析
- 网络（生态）演化分析





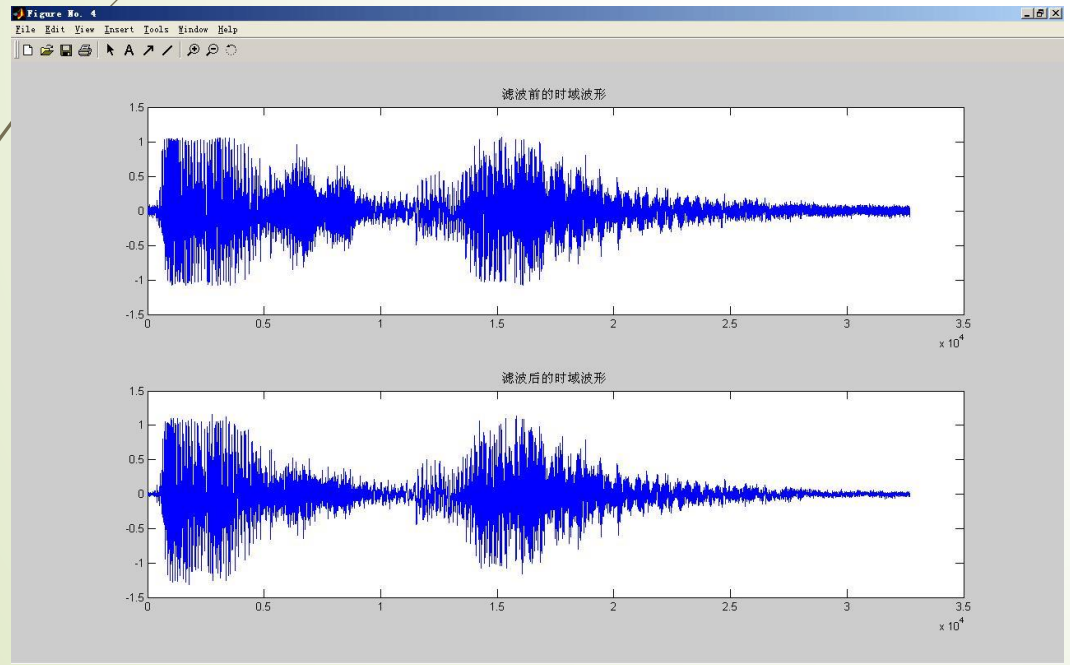
实时交互与控制

- 飞行器控制与预警系统
- 自动机械控制与自动驾驶系统

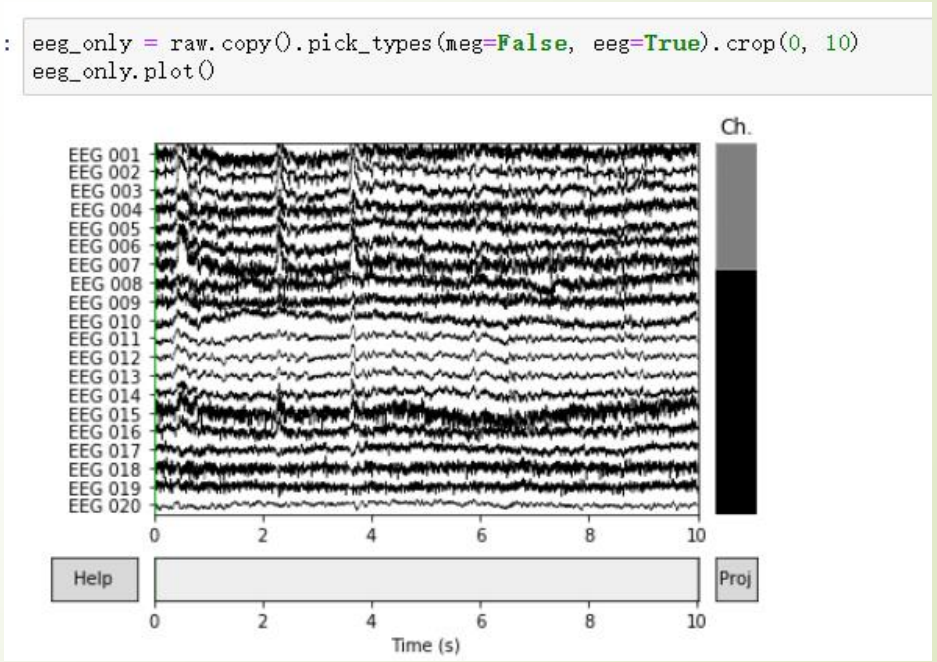


序列信号模式识别（语言模型理解）

- 语音信号识别
- 视频（动作）内容分析
- 生物信号（序列）处理




```
: eeg_only = raw.copy().pick_types(meg=False, eeg=True).crop(0, 10)
  eeg_only.plot()
```



TSA研究涉及的主要知识领域

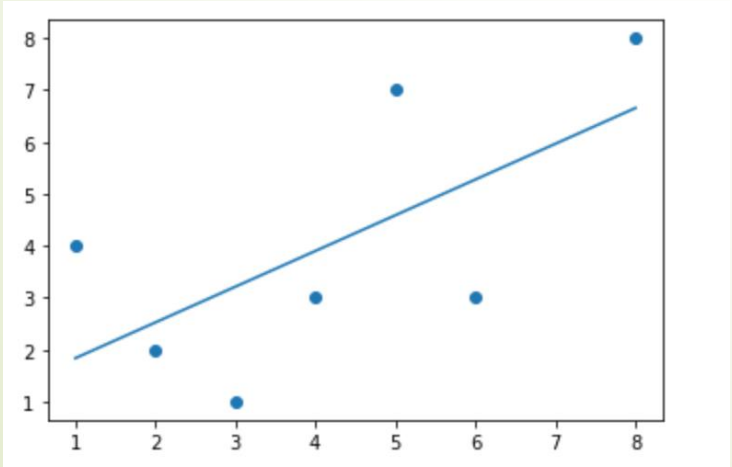
- 数字信号处理技术（谱分析）
- 统计与多元回归分析
- 模式识别与特征工程
- 深度学习：RNN+CNN

时间序列与传统的线性回归（预测）模型

- 模型回归
- 时间序列的成分分析与滤波
- 平稳序列模型

最小二乘loss线性回归

```
#多个样本点，最小二乘拟合
x = np.array([1, 2, 3, 4, 5, 6,8])
y = np.array([4, 2, 1, 3, 7,3,8])
X = x[:, np.newaxis]
model = LinearRegression().fit(X, y) # 简单线性回归
yfit = model.predict(X)
plt.scatter(x, y)
plt.plot(x, yfit)
```



多元函数线性回归

```
X1 = np.array([[1, 1], [1, 2], [2, 2], [2, 3]])
y1 = np.multi(X1, np.array([1, 2])) + 3                    # y = 1 * x_0 + 2 * x_1 + 3n
reg = LinearRegression().fit(X1, y1)                       # 二元函数回归

print(reg.score(X1, y1), reg.coef_, reg.intercept_)
reg.predict(np.array([[3, 5]]))
```

```
1.0 [1. 2.] 3.0000000000000018

array([16.])
```

```
y2 = X1 @ np.array([[0, 1],[1,0]])+ [1,3]

reg = LinearRegression().fit(X1, y2)                                    # 矩阵变换回归

print(reg.score(X1, y2), reg.coef_, reg.intercept_)
reg.predict(np.array([[3, 5]]))
```

```
1.0 [[-1.66533454e-16  1.00000000e+00]
 [ 1.00000000e+00 -1.11022302e-16]] [1. 3.]

array([[6., 6.]])
```

Ordinal Regression（有序回归）

- 回归目标为一组有序的分类

```
url = "https://stats.idre.ucla.edu/stat/data/ologit.dta"
data_student = pd.read_stata(url)  # DTA是Stata使用的一种专有的非文本文件格式
```

```
data_student.head(5)
```

	apply	pared	public	gpa
0	very likely	0	0	3.26
1	somewhat likely	1	0	3.21
2	unlikely	1	1	3.94
3	somewhat likely	0	0	2.81
4	somewhat likely	0	0	2.53

```
data_student['apply'].dtype
```

```
CategoricalDtype(categories=['unlikely', 'somewhat likely', 'very likely'], ordered=True)
```

Logit ordinal regression:

模型学习

```
import scipy.stats as stats
from statsmodels.miscmodels.ordinal_model import OrderedModel # ver > 0.13

mod_log = OrderedModel(data_student['apply'],
                       data_student[['pared', 'public', 'gpa']],
                       distr='logit')

res_log = mod_log.fit(method='bfgs', disp=False)
res_log.summary()
```

	coef	std err	z	P>|z|	[0.025	0.975]
pared	1.0476	0.266	3.942	0.000	0.527	1.569
public	-0.0586	0.298	-0.197	0.844	-0.642	0.525
gpa	0.6158	0.261	2.363	0.018	0.105	1.127
unlikely/somewhat likely	2.2035	0.780	2.827	0.005	0.676	3.731
somewhat likely/very likely	0.7398	0.080	9.236	0.000	0.583	0.897

Ordinal Logistic Regression | R Data Analysis Examples (ucla.edu)

模型预测

```
predicted = res_log.model.predict(res_log.params, exog=data_student[['pared', 'public', 'gpa']])
predicted
```

	apply	pared	public	gpa
0	very likely	0	0	3.26
1	somewhat likely	1	0	3.21
2	unlikely	1	1	3.94

```
array([[0.54884071, 0.35932276, 0.09183653],
       [0.30558191, 0.47594216, 0.21847593],
       [0.22938356, 0.47819057, 0.29242587],
       ...,
```

statsmodels.miscmodels.ordinal_model.OrderedModel.fit — statsmodels