# 《数据挖掘工程师实战》—取之精华，去伪存真

## 一、课前准备

### 1.1 讨论对于数据的理解

### 1. 2 熟练掌握Python

## 二、课堂主题

有这么一句话在业界广泛流传：数据和特征决定了机器学习的上限，而模型和算法只是逼近这个上限而已。那特征工程到底是什么呢？顾名思义，其本质是一项工程活动，目的是最大限度地从原始数据中提取特征以供算法和模型使用。

## 三、课堂目标

### 3.1 能够理解特征工程的用途和作用

### 3.2 能够掌握数值型、类别型和时间型的处理方式

## 四、知识要点

### 4.1 特征工程—数值型

- 特征工程

  特征 —>数据中抽取出来的对结果预测有用的信息

  特征工程师使用专业背景知识和技巧处理数据，使得特征能在机器学习算法中发挥更好的作用
- 互联网公司的数据挖掘工程师们工作内容是什么?

  研究各种高深的前沿算法与模型?

  深度学习N层复杂神经网络结构的搭建?

  ...
- 大部分数据科学比赛的时间都消耗在哪儿?

  构建复杂模型?

  调参与优化模型?

开课吧 kaikeba.com

...

- 实际上

  跑数据，各种MR、hiveSQL，数据仓库搬砖

  数据清洗，数据清洗，数据清洗...

  分析业务，分析case，找特征，找特征...

  简单可解释性好的模型为主，甚至一招LR打天下

- 工业界的特征工程有多大作用呢?
  - 某搜索引擎厂，广告部门点击率预估问题?

    2周内可以完成一次特征迭代，有效的情况下AUC提升约1-2% 一个月左右完成模型的小优化，很多时候AUC提升只有5%左右
  - 阿里天池与Kaggle比赛

    大于50%的时间花在分析业务场景，挖掘有效特征上

    特征有效的情况下，借助于基本模型，轻松进top10%

## 4.1.1 Standard Scaler $\frac{x_i-\mu}{\sigma}$

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
import numpy as np
import pandas as pd
np.set_printoptions(suppress=True)#在控制台输出过程中，默认小数会以科学计数法的形式输出

import warnings
warnings.filterwarnings("ignore") # 忽视
```

```
views = pd.DataFrame([1295., 25., 19000., 5., 1., 300.], columns=['views'])
print(views)
```

```
ss = StandardScaler()#初始化标准差
views['zscore'] = ss.fit_transform(views[['views']])
print(views)
```

```
vw =  np.array(views["views"])
print((vw[2] - np.mean(vw)) / np.std(vw))
```

## 4.1.2 Min-Max Scaler $\frac{x_i-min(x)}{max(x)-min(x)}$

```
mms = MinMaxScaler()
views['minmax'] = mms.fit_transform(views[['views']])
print(views)
```

开课吧 kaikeba.com

# 4.1.3 多项式特征

- 离散变量是指其数值只能用自然数或整数单位计算的则为离散变量.例如,企业个数,职工人数,设备台数等,只能按计量单位数计数,这种变量的数值一般用计数方法取得.
- 在一定区间内可以任意取值的变量叫连续变量,其数值是连续不断的,相邻两个数值可作无限分割,即可取无限个数值.例如,生产零件的规格尺寸,人体测量的身高,体重,胸围等为连续变量,其数值只能用测量或计量的方法取得.
- 如果变量可以在某个区间内取任一实数，即变量的取值可以是连续的，这随机变量就称为连续型随机变量， 比如，公共汽车每15分钟一班，某人在站台等车时间x是个随机变量， x的取值范围是[0,15)，它是一个区间，从理论上说在这个区间内可取任一实数3.5、√20等，因而称这随机变量是连续型随机变量。
- 模型是使用离散特征还是连续特征，其实是一个“海量离散特征+简单模型” 同 “少量连续特征+复杂模型”的权衡。既可以离散化用线性模型，也可以用连续特征加深度学习。就看是喜欢折腾特征还是折腾模型了。通常来说，前者容易，而且可以n个人一起并行做，有成功经验；后者目前看很赞，能走多远还须拭目以待

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib as mpl
import scipy.stats as spstats #统计函数

%matplotlib inline
mpl.style.reload_library()
mpl.style.use('classic') #经典的
mpl.rcParams['figure.facecolor'] = (1, 1, 1, 0) # 画布背景颜色
mpl.rcParams['figure.figsize'] = [6.0, 4.0] #宽是6.0英寸 长是4.0英寸
mpl.rcParams['figure.dpi'] = 100 #图片像素点
```

```
poke_df = pd.read_csv('datasets/Pokemon.csv', encoding='utf-8')
print(poke_df.head(5))
```

```
atk_def = poke_df[['Attack', 'Defense']]
print(atk_def.head())
```

```
print(poke_df[['Attack', 'Defense']].describe())
```

- 多项式特征跟SVM的核函数一样，低维变换到高维，多项式为2

开课吧 kaikeba.com

```
from sklearn.preprocessing import PolynomialFeatures

pf = PolynomialFeatures(degree=2, interaction_only=False,
include_bias=False)
res = pf.fit_transform(atk_def)
print(res)
#                           49^2     49*49     49^2
```

```
intr_features = pd.DataFrame(res, columns=['Attack', 'Defense', 'Attack^2',
'Attack x Defense', 'Defense^2'])
print(intr_features.head(5))
```

## 4.1.4 数据分箱技术binning特征

```
fcc_survey_df = pd.read_csv('datasets/fcc_coder_survey_subset.csv',
encoding='utf-8')
print(fcc_survey_df[['ID.x', 'EmploymentField', 'Age', 'Income']].head())
```

```
fig, ax = plt.subplots()
fcc_survey_df['Age'].hist(color='#A9C5D3')# 蓝色D3的话就是去不去饱和、满意度
ax.set_title('Developer Age Histogram', fontsize=12)
ax.set_xlabel('Age', fontsize=12)
ax.set_ylabel('Frequency', fontsize=12) #Frequency频率
plt.show()
```

- Binning based on rounding

```
Age Range: Bin
---------------
 0 -  9   : 0.0
10 - 19   : 1.0
20 - 29   : 2.0
30 - 39   : 3.0
40 - 49   : 4.0
50 - 59   : 5.0
60 - 69   : 6.0
  ... and so on
```

```
fcc_survey_df['Age_bin_round'] =
np.array(np.floor(np.array(fcc_survey_df['Age']) / 10.))
print(fcc_survey_df[['ID.x', 'Age', 'Age_bin_round']].iloc[1071:1076])
```

## 4.1.4 分位数切分

开课吧 kaikeba.com

```
print(fcc_survey_df[['ID.x', 'Age', 'Income']].iloc[4:9])
```

```
fig, ax = plt.subplots()
fcc_survey_df['Income'].hist(bins=20, color='#A9C5D3')#
ax.set_title('Developer Income Histogram', fontsize=12)
ax.set_xlabel('Developer Income', fontsize=12)
ax.set_ylabel('Frequency', fontsize=12)
plt.show()
```

```
quantile_list = [0, .25, .5, .75, 1.]
quantiles = fcc_survey_df['Income'].quantile(quantile_list)
print(quantiles)
```

```
fig, ax = plt.subplots()
fcc_survey_df['Income'].hist(bins=20, color='#A9C5D3')

for quantile in quantiles:
    qvl = plt.axvline(quantile, color='r') #axvline：用于画竖线，0分位、1/4分位、1/2分位、3/4分位和1分位
ax.legend([qvl], ['Quantiles'], fontsize=10)

ax.set_title('Developer Income Histogram with Quantiles', fontsize=12)
ax.set_xlabel('Developer Income', fontsize=12)
ax.set_ylabel('Frequency', fontsize=12)
plt.show()
```

```
quantile_labels = ['0-25Q', '25-50Q', '50-75Q', '75-100Q']
fcc_survey_df['Income_quantile_range'] = pd.qcut(fcc_survey_df['Income'],
                                                 q=quantile_list)
fcc_survey_df['Income_quantile_label'] = pd.qcut(fcc_survey_df['Income'],
                                                 q=quantile_list,
labels=quantile_labels)
print(fcc_survey_df[['ID.x', 'Age', 'Income',
                     'Income_quantile_range', 'Income_quantile_label']].iloc[4:9])
```

## 4.1.6 对数变换 COX-BOX

```
fcc_survey_df['Income_log'] = np.log((1+ fcc_survey_df['Income']))
print(fcc_survey_df[['ID.x', 'Age', 'Income', 'Income_log']].iloc[4:9])
```

## 4.1.7 小结

- 下列表述不正确的是:

A. 特征工程是使用专业知识和技巧处理数据，获取有效的数据表达

开课吧 kaikeba.com

B. 数据清洗部分会对脏数据进行处理

C. 特征降维是特征选择

D. 统计分析是我们定位脏数据的方法之一

### 4.1.8 检查点

- 下面哪些是我们会用到的对数值型列做幅度缩放的方法:

  A. 最大最小值幅度缩放

  B. 标准化

  C. 归一化

  D. 去均值

## 4.2 特征工程—类别型

### 4.2.1 LabelEncoder

- Label标准化

```
vg_df = pd.read_csv('datasets/vgsales.csv', encoding = "ISO-8859-1") #单字节编码，最多能表示的字符范围是0-255，应用于英文系列。比如，字母a的编码为0x61=97
print(vg_df[['Name', 'Platform', 'Year', 'Genre', 'Publisher']].iloc[1:7])
```

```
genres = np.unique(vg_df['Genre'])
print(genres)
```

```
from sklearn.preprocessing import LabelEncoder

gle = LabelEncoder()
genre_labels = gle.fit_transform(vg_df['Genre'])
genre_mappings = {index: label for index, label in enumerate(gle.classes_)}
print(genre_mappings)
```

```
vg_df['GenreLabel'] = genre_labels
print(vg_df[['Name', 'Platform', 'Year', 'Genre', 'GenreLabel']].iloc[1:7])
#变成离散型数据
```

### 4.2.2 Map

开课吧 kaikeba.com

```
poke_df = pd.read_csv('datasets/Pokemon.csv', encoding='utf-8')
"""
frac    抽取行的比例,例如frac=0.8，就是抽取其中80%。
random_state=None,取得数据不重复
random_state=1,可以取得重复数据
reset_index(drop=True):有时候，我们可能需要打混后数据集的index（索引）还是按照正常的排序
"""

poke_df = poke_df.sample(random_state=1, frac=1).reset_index(drop=True)

print(np.unique(poke_df['Generation']))
```

```
gen_ord_map = {'Gen 1': 1, 'Gen 2': 2, 'Gen 3': 3,
               'Gen 4': 4, 'Gen 5': 5, 'Gen 6': 6}

poke_df['GenerationLabel'] = poke_df['Generation'].map(gen_ord_map)
print(poke_df[['Name', 'Generation', 'GenerationLabel']].iloc[9:19])
```

## 4.2.3 One-hot Encoding

```
print(poke_df[['Name', 'Generation', 'Legendary']].iloc[4:10])
```

```
from sklearn.preprocessing import OneHotEncoder, LabelEncoder

#标签编码，Gen 2——>1 ,Gen 6——>5
gen_le = LabelEncoder()
gen_labels = gen_le.fit_transform(poke_df['Generation'])
poke_df['Gen_Label'] = gen_labels

#False——>0 True——>1
leg_le = LabelEncoder()
leg_labels = leg_le.fit_transform(poke_df['Legendary'])
poke_df['Lgnd_Label'] = leg_labels

poke_df_sub = poke_df[['Name', 'Generation', 'Gen_Label', 'Legendary',
'Lgnd_Label']]
print(poke_df_sub.iloc[4:10])
```

开课吧 kaikeba.com

```
gen_ohe = OneHotEncoder()
gen_feature_arr = gen_ohe.fit_transform(poke_df[['Gen_Label']]).toarray()

gen_feature_labels = list(gen_le.classes_)
print (gen_feature_labels)

gen_features = pd.DataFrame(gen_feature_arr, columns=gen_feature_labels)
print("-----------------------------------------------------------")
print(gen_features)
```

## 4.2.4 Get Dummy

```
#数据和特征决定了机器学习的上限，而模型和算法只是逼近这个上限而已,dummies:仿制品，poke:一桶、一堆
gen_dummy_features = pd.get_dummies(poke_df['Generation'],
drop_first=True)#drop_first表示去除one-hot编码后的第一列数据，反之就有第一列
print(pd.concat([poke_df[['Name', 'Generation']], gen_dummy_features],
axis=1).iloc[4:10])#axis=1按照行计算
```

```
##drop_first表示去除one-hot编码后的第一列数据，反之就有第一列
gen_onehot_features = pd.get_dummies(poke_df['Generation'])
print(pd.concat([poke_df[['Name', 'Generation']], gen_onehot_features],
axis=1).iloc[4:10])
```

```
popsong_df = pd.read_csv('datasets/song_views.csv', encoding='utf-8')
print(popsong_df.head(10))
```

## 4.2.5 二值特征

```
watched = np.array(popsong_df['listen_count'])
watched[watched >= 1] = 1 #大于等于1的表示已经被听过了，小于1的没有听过了
popsong_df['watched'] = watched
print(popsong_df.head(10))
```

```
from sklearn.preprocessing import Binarizer
#threshold：临界值
bn = Binarizer(threshold=0.9)#其中参数threshold表示大于0.9的数据表示为1，小于等于0.9的数表示为0
pd_watched = bn.transform([popsong_df['listen_count']])
print(pd_watched) #转换后为二维数组
pd_watched = pd_watched[0]#就一个元素，一维数组
popsong_df['pd_watched'] = pd_watched #变量名列
```

开课吧 kaikeba.com

### 4.2.6 检查点

- 对类别型变量做one-hot编码，可以保证每个类别取到的不同值(颜色:红、黄、蓝)地位一致?

  正确答案:对

- 为什么有时候要对特征做归一化呢?

  归一化后加快了梯度下降求最优解的速度

  归一化有可能提高精度

## 4.3 特征中重要的Python插件

### 4.3.1 Permutation Importance

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier #集成算法对解释模型效果是很好的

import warnings
warnings.filterwarnings("ignore")
```

```
data = pd.read_csv('./datasets/FIFA Statistics.csv')
print(data.head(5) )
```

```
#Man of the Match 最佳球员作为y值
y = (data['Man of the Match'] == "Yes")  # 转换标签
print(y[:5])
```

```
feature_names = [i for i in data.columns if data[i].dtype in [np.int64]]#列表推导式，只取数值型数据，非数值型就不要了
X = data[feature_names]
print(X.head())
```

```
train_X, val_X, train_y, val_y = train_test_split(X, y, random_state=1)
my_model = RandomForestClassifier(random_state=0).fit(train_X, train_y)
```

- eli5文档：https://eli5.readthedocs.io/en/latest/tutorials/xgboost-titanic.html#explaining-weights

开课吧 kaikeba.com

```
#pip install eli5，科学上网，国外的慢
#pip install -i https://pypi.tuna.tsinghua.edu.cn/simple eli5
import eli5
from eli5.sklearn import PermutationImportance
# 由大到小，进行降序
perm = PermutationImportance(my_model, random_state=1).fit(val_X, val_y)
eli5.show_weights(perm, feature_names = val_X.columns.tolist())#把perm加载进来

#±后面的数字表示多次随机重排之间的差异值
```

## 4.3.2 Partial Dependence Plots

前提同样是应用在模型建立完成后进行使用，概述如下：

- 首先选中一个样本数据，此时想观察Ball Possession这一列对结果的影响。
- 保证其他特征列不变，改变当前观察列的值，例如选择40%,50%,60%分别进行预测，得到各自的结果。
- 对比结果就能知道当前列(Ball Possession)对结果的影响情况。

```
from matplotlib import pyplot as plt
from pdpbox import pdp, get_dataset, info_plots #pip install -i
https://pypi.tuna.tsinghua.edu.cn/simple pdpbox

pdp_goals = pdp.pdp_isolate(model=my_model, dataset=val_X,
model_features=feature_names, feature='Goal Scored')#进球数量

pdp.pdp_plot(pdp_goals, 'Goal Scored')
plt.show()
```

```
feature_to_plot = 'Distance Covered (Kms)'

rf_model = RandomForestClassifier(random_state=0).fit(train_X, train_y)

pdp_dist = pdp.pdp_isolate(model=rf_model, dataset=val_X,
model_features=feature_names, feature=feature_to_plot)

pdp.pdp_plot(pdp_dist, feature_to_plot)
plt.show()
```

- 双特征观察
- 需要先改一下源码：将Anaconda3\Lib\site-packages\pdpbox 下pdp_plot_utils.py文件中contour_label_fontsize=fontsiz改成fontsize=fontsize

开课吧 kaikeba.com

```
features_to_plot = ['Goal Scored', 'Distance Covered (Kms)']
inter1  =  pdp.pdp_interact(model=rf_model, dataset=val_X,
model_features=feature_names, features=features_to_plot)

pdp.pdp_interact_plot(pdp_interact_out=inter1,
feature_names=features_to_plot, plot_type='contour')
plt.show()
```

## 4.3.3 SHAP Values

- 可以直观的展示每一个特征对结果走势的影响

```
row_to_show = 5
data_for_prediction = val_X.iloc[row_to_show]
data_for_prediction_array = data_for_prediction.values.reshape(1, -1)


print(my_model.predict_proba(data_for_prediction_array))
```

- 导入shap工具包可能出现问题，在Anaconda3/lib/site-packages/numpy/lib/arraypad.py中添加下面两个函数保存，重新加载即可，保存成utf-8

```
import shap
explainer = shap.TreeExplainer(my_model)

#计算第5个的数据,一共是2行的数据,
shap_values = explainer.shap_values(data_for_prediction)
```

- 返回的SHAP values中包括了negative和positive两种情况，通常选择一种（positive）即可。

```
shap.initjs()#显示格式转换
shap.force_plot(explainer.expected_value[1], shap_values[1],
data_for_prediction)
```

- Summary Plots

```
explainer = shap.TreeExplainer(my_model)

shap_values = explainer.shap_values(val_X)

shap.summary_plot(shap_values[1], val_X)
```

开课吧 kaikeba.com

- 其中y轴表示每一个特征，x轴表示对结果是促进还是抑制的，不同的颜色表示作用效果，Color Bar颜色越红，值越大
- “GoalScored”蓝色的点代表进球数越少，该特征越抑制结果 ，进球数越多，红色点，对结果是促进的作用

# 五、拓展点、未来计划、行业趋势

## 5.1 特征获取需要解决两个问题:一是确定选择算法,二是确定评价标准

## 5.2 简单的可解释性高的模型，能线性模型就不非线性，特征提取+特征工程+统计学习可以的，就不上黑匣子deep Learning

# 六、总结

## 6.1 标准化和归一化异同?

这俩都是线性变换，将原有数据进行了放缩，归一化一般放缩到[0,1]，标准化则转为服从标准正态分布，数据的大小顺序没有发生改变 影响归一化的主要是两个极值，而标准化里每个数据都会影响，因为计算均值和标准差。

## 6.2 类别型数据你是如何处理的？比如游戏品类，地域，设备

序号编码、one-hot编码、二进制编码

## 6.3 序号编码、one-hot编码、二进制编码都是什么？适合怎样的类别型数据?

序号编码通常用于处理类别建具有大小关系的数据 one-hot编码用于处理类别建不具有大小关系的特征。但是当类别取值较多的时候需要注意 (1) 用稀疏向量来节省空间 (2) 配合特征选择来降低维度。二进制编码，先用序号编码给每个类别赋予一个类别id，然后将类别id转为二进制编码，本质上是利用二进制对id进行了hash映射，得到0-1特征向量，且维度少于one-hot编码。

# 七、作业

## 7.1 注意老师发的笔记和代码，务必都跑通

开课吧 kaikeba.com