ctr-prediction-demo

May 23, 2019

项目地址：https://github.com/georgethrax/ctr-prediction-demo

1 环境配置

1.1 安装 Anaconda3：python3 发行版

安装 Anaconda3 后，本文所用到的 python 库就已经包含在内了。从 https://www.anaconda.com/distribution/ 下载安装包并安装即可。

1.2 运行代码

从 GitHub 下载本项目 https://github.com/georgethrax/ctr-prediction-demo 后，有以下几种方式来运行代码：

1.2.1 通过 jupyter notebook 运行代码（推荐的方式）

打开控制台（Windows CMD，Linux/MacOS Terminal），跳转到本项目文件所在的目录

```
cd ctr-prediction-demo
```

启动 jupyter notebook

```
jupyter notebook
```

此时会自动浏览器 #### 打开本项目中的 `ctr-prediction-demo.ipynb` 文件，按顺序执行代码即可

1.2.2 通过 spyder 运行代码

spyder 是随 Anaconda 安装好的一个轻量级 python IDE。用 spyder 打开 `ctr_prediction-demo.py` 并运行即可。

1.2.3 直接在控制台运行代码

打开控制台（Windows CMD，Linux/MacOS Terminal），跳转到本项目文件所在的目录

```
cd ctr-prediction-demo
```

执行代码

```
python ctr-prediction-demo.py
```

2 问题描述

- 问题背景：2015 在线广告点击率 (CTR) 预估大赛 https://www.kaggle.com/c/avazu-ctr-prediction
- 任务目标：根据广告的特征数据，预测一个广告是否被用户点击 (点击/未点击的二分类问题)
- 数据文件：`ctr_data.csv`。原始数据过大，这里截取 10000 条数据。
- 数据字段：
  - id
  - click 是否点击，0/1
  - hour
  - C1 一个个类别型特征 (categorical feature)，具体业务含义被隐去
  - banner_pos
  - site_id
  - site_domain
  - site_category
  - app_id
  - app_domain
  - app_category
  - device_id
  - device_ip
  - device_model
  - device_type
  - device_conn_type
  - C14-C21 一些类别型特征

其中，id 不使用，click 被作为标签，其他字段可以被用作特征

3 收集数据

这里假设数据已经收集并整理为磁盘文件 ctr_data.csv

In [20]:
```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.cross_validation import train_test_split
from sklearn.metrics import accuracy_score, roc_auc_score, log_loss
import warnings
warnings.simplefilter("ignore")
```

In [21]:
```
# 读取数据集
df = pd.read_csv("./ctr_data.csv", index_col=None)
df.head()
```

```
Out[21]:              id  click      hour    C1  banner_pos   site_id site_domain  \
         0  1.000009e+18      0  14102100  1005           0  1fbe01fe    f3845767
         1  1.000017e+19      0  14102100  1005           0  1fbe01fe    f3845767
         2  1.000037e+19      0  14102100  1005           0  1fbe01fe    f3845767
         3  1.000064e+19      0  14102100  1005           0  1fbe01fe    f3845767
         4  1.000068e+19      0  14102100  1005           1  fe8cc448    9166c161

           site_category    app_id app_domain  ... device_type device_conn_type    C14  \
         0      28905ebd  ecad2386   7801e8d9  ...           1                2  15706
         1      28905ebd  ecad2386   7801e8d9  ...           1                0  15704
         2      28905ebd  ecad2386   7801e8d9  ...           1                0  15704
         3      28905ebd  ecad2386   7801e8d9  ...           1                0  15706
         4      0569f928  ecad2386   7801e8d9  ...           1                0  18993

            C15  C16   C17  C18  C19     C20  C21
         0  320   50  1722    0   35      -1   79
         1  320   50  1722    0   35  100084   79
         2  320   50  1722    0   35  100084   79
         3  320   50  1722    0   35  100084   79
         4  320   50  2161    0   35      -1  157

         [5 rows x 24 columns]
```

4 特征工程

为简单起见，这里仅考虑特征选择和类别型特征编码。

实际场景中，可能面临缺失值处理、离群点处理、日期型特征编码、数据降维等等。

4.1 特征选择

设置用到的字段/特征/列

```
In [22]: cols_data = ['C1','banner_pos', 'site_domain',  'site_id', 'site_category','app_id',\
                      'app_category',  'device_type',  'device_conn_type', 'C14', 'C15','C16']
         cols_label = ['click']
```

由设置好的特征字段，构造数据集 X 和标签 y

```
In [23]: X = df[cols_data]
         y = df[cols_label]
```

4.2 特征编码

特征编码：将原始数据的字符串等特征转换为模型能够处理的数值型特征。LR,SVM 类模型可以使用 OneHotEncoder。决策树类模型可以使用 LabelEncoder。

为简单起见，本文仅讨论决策树类模型，故仅使用 LabelEncoder 特征编码

```
In [24]: cols_categorical = ['site_domain', 'site_id', 'site_category', 'app_id', 'app_category
         lbl = LabelEncoder()

         for col in cols_categorical:
             print(col)
             X[col] = lbl.fit_transform(X[col])
```

```
site_domain
site_id
site_category
app_id
app_category
```

```
In [25]: X.head()
```

Out[25]:

	C1	banner_pos	site_domain	site_id	site_category	app_id	\
0	1005	0	301	43	2	293	
1	1005	0	301	43	2	293	
2	1005	0	301	43	2	293	
3	1005	0	301	43	2	293	
4	1005	1	169	374	0	293	

	app_category	device_type	device_conn_type	C14	C15	C16
0	0	1	2	15706	320	50
1	0	1	0	15704	320	50
2	0	1	0	15704	320	50
3	0	1	0	15706	320	50
4	0	1	0	18993	320	50

4.3 划分训练集、测试集

这里采用训练集占 80%，测试集占 20%

```
In [40]: X_train,X_test, y_train, y_test =  \
             train_test_split(X, y, test_size=0.2, random_state=0)
```

5 建立一个模型，并训练、测试

这里调用一个 sklearn 算法库中现成的决策树分类器 DecisionTreeClassifier，记为 clf1

5.1 创建模型

创建一个分类模型，命名为 clf1，使用默认模型参数

```
In [27]: clf1 = DecisionTreeClassifier()
```

5.2 训练

在训练集上训练分类器 clf1

```
In [28]: clf1.fit(X_train, y_train)
```

```
Out[28]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                     max_features=None, max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, presort=False, random_state=None,
                     splitter='best')
```

5.3 预测

使用训练好的分类器 clf1，在测试集上预测分类结果预测结果有两种形式 - y_score: 为每个测试样本 x 预测一个 0.0~1.0 的实数，表示 x 被分类为类别 1 的概率 - y_pred: 为每个测试样本 x 预测一个 0/1 类别标签。当 y_score(x) > 0.5 时，y_pred(x) = 1。当 y_score(x) < 0.5 时，y_pred(x) = 0。

```
In [29]: # 分类器预测的类别为 1 的概率值/分数值
         y_score = clf1.predict_proba(X_test)[:, clf1.classes_ == 1]

         # 按阈值 (默认 0.5) 将 y_score 二值化为 0/1 预测标签
         y_pred = clf1.predict(X_test)
```

5.4 评估

评估预测结果，使用 ACC, AUC, logloss 等评价指标。ACC, AUC 越接近于 1，logloss 越小，分类效果越好。

```
In [30]: acc = accuracy_score(y_test, y_pred)
         auc = roc_auc_score(y_test, y_score)
         logloss = log_loss(y_test, y_score)
         print(acc, auc, logloss)
```

```
0.818 0.6621736672845748 2.0666912551211327
```

6 修改模型参数，重新训练、测试

为模型 clf1 换一组参数，记为 clf1_p1

出于演示目的，不妨令 clf1_p1 中的一个模型参数修改为 max_leaf_nodes=10。(clf1 原参数为 max_leaf_nodes=None)

6.1 创建模型

```
In [31]: clf1_p1 = DecisionTreeClassifier( max_leaf_nodes=10)
```

6.2 训练

```
In [32]: clf1_p1.fit(X_train, y_train)
```

```
Out[32]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                        max_features=None, max_leaf_nodes=10,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=2,
                        min_weight_fraction_leaf=0.0, presort=False, random_state=None,
                        splitter='best')
```

6.3 预测

```
In [33]: y_score = clf1_p1.predict_proba(X_test)[:, clf1_p1.classes_ == 1]
         y_pred = clf1_p1.predict(X_test)
```

6.4 评估

```
In [34]: acc = accuracy_score(y_test, y_pred)
         auc = roc_auc_score(y_test, y_score)
         logloss = log_loss(y_test, y_score)
         print(acc, auc, logloss)
```

```
0.828 0.6583099862375015 0.43256841278416175
```

从评估指标来看，模型 clf1_p1 比 clf1 差。

7 更换模型，重新训练、测试

这里换一个 sklearn 库中现成的 GradientBoostingClassifier，记为 clf2

```
In [35]: from sklearn.ensemble import GradientBoostingClassifier
         clf2 = GradientBoostingClassifier()
```

7.1 训练

```
In [36]: clf2.fit(X_train, y_train)
```

```
Out[36]: GradientBoostingClassifier(criterion='friedman_mse', init=None,
                           learning_rate=0.1, loss='deviance', max_depth=3,
                           max_features=None, max_leaf_nodes=None,
                           min_impurity_decrease=0.0, min_impurity_split=None,
                           min_samples_leaf=1, min_samples_split=2,
                           min_weight_fraction_leaf=0.0, n_estimators=100,
                           presort='auto', random_state=None, subsample=1.0, verbose=0,
                           warm_start=False)
```

7.2 预测

```
In [37]: y_score = clf2.predict_proba(X_test)[:, clf2.classes_ == 1]
         y_pred = clf2.predict(X_test)
```

7.3 评估

```
In [38]: acc = accuracy_score(y_test, y_pred)
         auc = roc_auc_score(y_test, y_score)
         logloss = log_loss(y_test, y_score)
         print(acc, auc, logloss)
```

```
0.8225 0.6870040936411639 0.4252623099250592
```

从测试集上的评估指标来看，模型 clf2 比 clf1,clf1_p1 好

8 模型迭代

将收集数据、特征工程、模型选择、模型参数选择、训练测试等步骤反复迭代，直到评价指标令人满意为止。