【Python】 No.7 机器学习基础

NAU Analysts 2020-04-22 23:41

以下文章来源于THE QUESTION ，作者高镜骐

THE QUESTION

Welcome

前言

Halo大家好！上期的推送为大家讲解了如何通过Tushare来获取金融数据，那这一期的推送让我们来简单地看看机器学习基础。

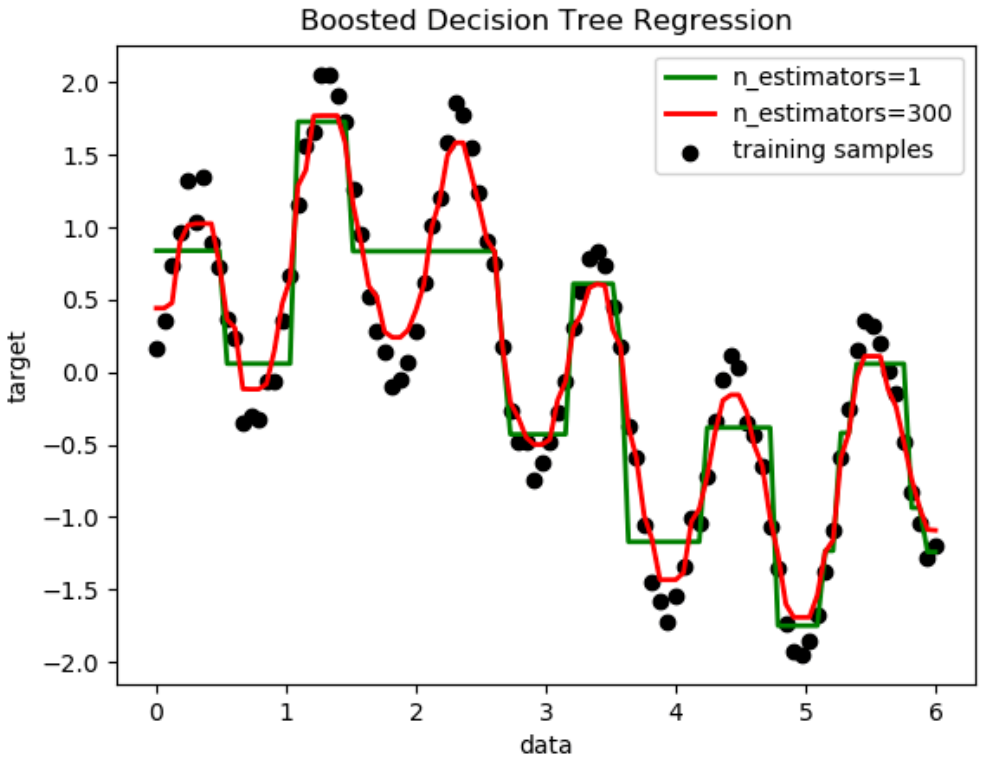
认识机器学习

机器学习就是让机器具备人一样学习的能力，专门研究计算机怎样模拟或实现人类的学习行为，以获取新的知识或技能，重新组织已有的知识结构使之不断改善自身的性能，它是人工智能的核心。

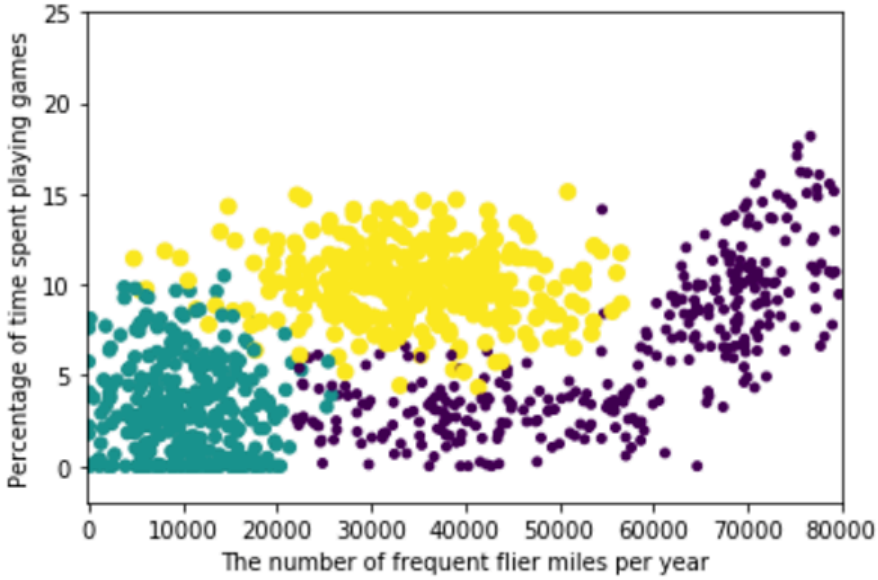
其实机器学习基础在我们的生活中已经有着广泛的应用，例如：数据挖掘、计算机视觉、自然语言处理、生物特征识别、搜索引擎、医学诊断、检测信用卡欺诈、证券市场分析、DNA序列测序、语音和手写识别、战略游戏和机器人运用等等。

机器学习三大任务

生活中，机器学习主要有三大任务。
当我们想根据已有数据对新数据进行结果预测时，比如预测股票的价格，这就是回归任务。回归任务的特点是我们想要预测的是一个连续的结果，我们的目标是构建一个好的机器学习模型，能够使得预测的结果和真实的结果尽可能的接近。



当我们根据已有数据对新数据进行分类时，比如预测股票的价格是涨还是跌，就这是一个分类任务。在日常生活中分类任务的场景往往比回归任务更多，因为人生和生活中更多的是选择题。比如在近期的疫情中我们可以利用人工智能机器学习算法，通过学习资深医生的临床诊断经验，训练出具有高准确率的新冠病毒感染识别模型，辅助医生快速进行临床诊断。由于已经具有了医生的诊断数据，并且有了诊断结果，因此可以利用机器学习中的监督学习模型进行训练。因为诊断结果是感染或者没有感染这两种情况，所以就是一个二分类模型。这就是分类任务；



需要特别指出的是，无论是回归任务还是分类任务，都属于我们已经获取了训练数据的目标数据，往往也将其称作标签。

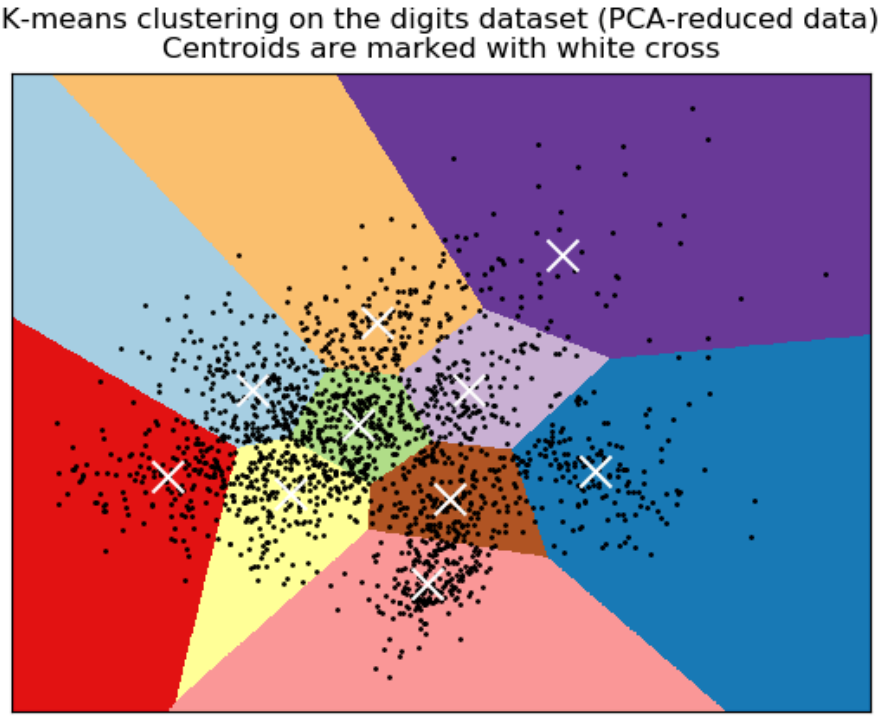
比如训练预测第二天股票涨跌的模型，我们能够获取到的历史数据已经告诉我们第二天的股价的真实情况，所以我们在训练模型时就可以拿模型的预测结果和真实结果进行对比。

在这种情况下，当我们训练出了一个模型后，如果发现模型的预测结果和真实结果相差太远，我们就可以让模型继续训练，以期达到一个满意的结果。

大家不难发现，在这种情况下训练模型，实际上我们是**有监督的**，当我们监督到分类任务的模型准确率较低就会让模型“回炉重造”。

与回归任务和分类任务这种有监督的学习不同。还有一种经典的任务，就是当我们不知道类别，只根据新数据特征来进行分类，比如通过观测一只股票的波动性等性质推断出它属于科技股，这就是聚类任务。

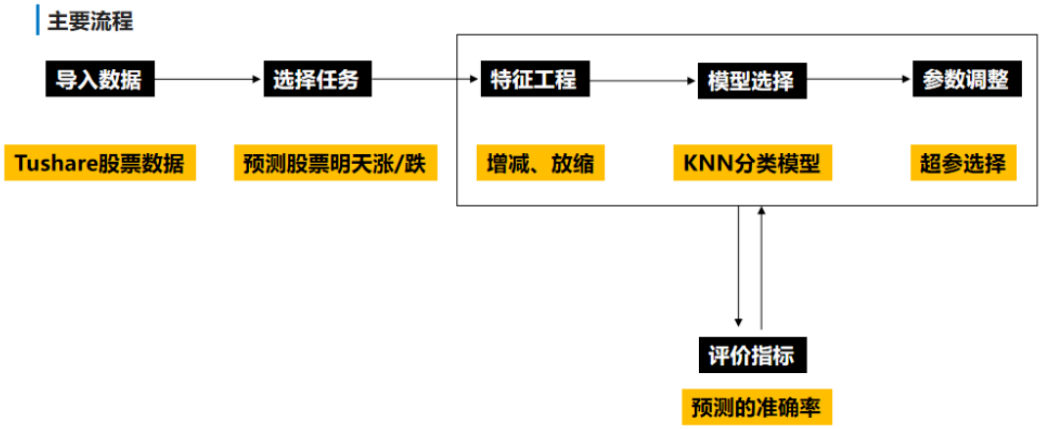
聚类任务之所以叫无监督的，是因为我们没有一个有效的数学指标去定义聚类的结果是好坏，很多时候是根据我们对数据的了解。



这三种任务我们之后都会以实际的金融案例进行学习。

机器学习主要流程

了解了机器学习，下面让我们来认识一下机器学习的主要流程。

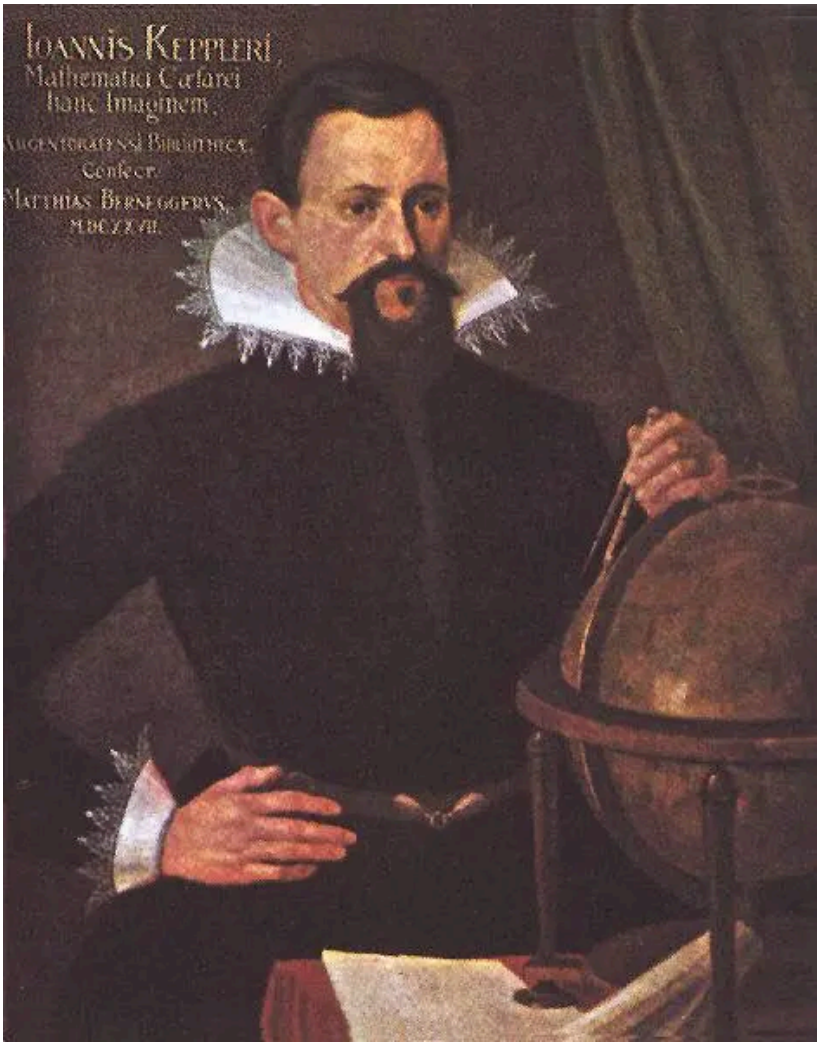


1. 数据统计：首先我们要通过各种途径获取数据，比如像之前推送介绍的从Tushare导入股票数据，也可以使用爬虫进行数据的收集。
2. 数据处理：获取数据后，这些数据通常是是原始缺乏调整的，我们要把数据进行调整，比如选择数据结构模型，通过增减、放缩，使得数据有序排列，方便我们从中寻找出特征规律。
3. 模型训练：在选择完模型后我们需要对参数进行调整，比如我们选择KNN模型，那么我们就需要对超参数进行调整，只有在最优的超参数下进行我们才能得到最优的结果。

4. 模型验证：对于模型的验证，我们不仅要通过已有的数据进行验证，更要通过模型预测的未来数据来验证模型的有效性，之所以我们构建模型进行机器学习，是因为我们要使用它预测未来，让我们可以更好地应对未来。

5. 模型使用：完成上述工作，我们就可以使用模型了。

其实理科生高中都接触过机器学习这一流程产生的模型结果，比如大家熟知的开普勒第三定律。首先，德国天文学家约翰尼斯·开普勒根据丹麦天文学家第谷·布拉赫等人的观测资料和星表**获取数据**，之后他想探寻行星运动周期的规律，这显然是一个回归任务，接着他**构建了初步模型**，通过对**数据的反复整理、计算**，并对模型的**参数不断调整**，最终发现某种参数的组合能够与我们的结果大致连成一条直线，即行星的运行周期T和R的二分之三次方成正比（R为轨道半径），因此他得出开普勒第三定律，这样的过程就是机器学习的主要流程。

1. 

对机器学习有了一定的认识，今后我们就可以使用机器学习进行更多的操作了。

认识SKlearn和KNN

机器学习中我们经常会使用到SKlearn，首先我们来认识一下SKlearn。

SKlearn是一个Python第三方提供的非常强力的机器学习库，它包含了从数据预处理到训练模型的各个方面。在实战使用SKlearn可以极大节省我们编写代码的时间以及减少我们的代码量，使我们有更多的精力去分析数据分布，调整模型和修改超参。

官网网址：

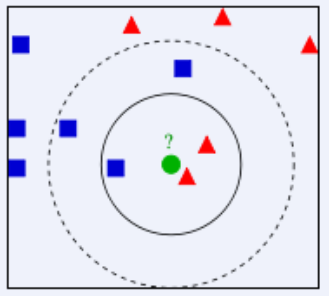
```
https://sklearn.apachecn.org/
https://scikit-learn.org/stable/modules/classes.html
https://sklearn.apachecn.org/
```

安装方法：

```
(bili) C:\Users\fcncassandra>pip install scikit-learn -i https://pypi.tuna.tsinghua.edu.cn/simple
```

初步认识SKlearn后，我们就来学习一下之前模型训练中提到的KNN。

K-Nearest Neighbour algorithm，称为邻近算法，是在给定一个有标签数据集的情况下，让我们预测一个新的数据属于哪一类别。也就是说在原有数据中，寻找与新数据最接近的k个数据，通过这k个数据的标签来判断新数据的类别。在这里，k就是超参数，预测的准确率就是对分类的检测评价。中国俗语说道：“物以类聚，人以群分”，从KNN中我们可以学到这看待世界的方法。



（在上图中，绿圆是我们需要预测的目标。k=3时，最接近绿圆的三个图标中红三角占2/3，因此绿圆被赋予红三角类；同理，k=5时，最接近绿圆的五个图标中蓝方占3/5，因此绿圆被赋予蓝方类，以此类推）

KNN手写数字集

下面我们通过KNN手写数字集进行展示。

在Excel或是PPT演示文稿中，给字体选择颜色时点其他颜色，选择RGB颜色模式时我们可以发现颜色分为红色、绿色、蓝色，通过输入不同的数值（范围0-255），我们可以得到各种各样的颜色。

（输入不同数值我们在右下角得到了不同的颜色）

知道了颜色可以数字化之后，我们就有了可以建模的基本条件了。在今天这个时代，之所以人工智能这么火，完全得益于大数据的支撑。人工智能的一些技术和理念其实几十年甚至上百年前就诞生了，但是一直不具备大量数字化的数据场景进行应用，而爆发式增长的互联网时代则在人们生活的各个维度上提供了可以数字化的原材料。

金融行业大多数岗位的工作性质每天都在和数据打交道、处理信息，所以光会自动化处理数据只能减轻我们日常的劳作，而深挖数据背后的逻辑规律，产生一个可以预测未来的“模型”才能让我们产生Insight。机器学习就提供了这样一个机会。

在接下的案例中，我们使用的是一个灰度图像的数据，即用一个数字来表示灰色的深浅，数字越大像素点越黑，越小像素点越白，下面就开始我们的手写数字集。

Sklearn中给予了非常多的实例可以供大家学习，在导入numpy、matplotlib后，这里我们导入Sklearn中的load_digits函数包。

```
import numpy as np
```

```
import matplotlib.pyplot as plt
from sklearn.datasets import load_digits
```

把load_digits赋值给digits，通过.keys形式可以观察到可供选取的数值类。

```
digits.keys()
```

```
dict_keys(['data', 'target', 'target_names', 'images', 'DESCR'])
```

获取digits中数据，获得的是二维数组。

```
digits['data']
```

```
array([[ 0.,  0.,  5., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ..., 10.,  0.,  0.],
       [ 0.,  0.,  0., ..., 16.,  9.,  0.],
       ...,
       [ 0.,  0.,  1., ...,  6.,  0.,  0.],
       [ 0.,  0.,  2., ..., 12.,  0.,  0.],
       [ 0.,  0., 10., ..., 12.,  1.,  0.]])
```

1797个样本，每个样本包括8*8像素的图像和一个[0, 9]整数的标签

用ravel函数将二维数组变为一维数组方便之后去重。

```
digits.data.ravel()
```

```
array([ 0.,  0.,  5., ..., 12.,  1.,  0.])
```

使用numpy取特异值的功能取出独立的数值（去重）。

```
np.unique(digits.data.ravel())
```

```
array([ 0.,  1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10., 11., 12.,
       13., 14., 15., 16.])
```

得到0-16这些数值后，拿第一个数值0为例，它的64个数据之前被变化成了一维维度，为了显示数字集效果，这里我们要把它重新变为8*8的维度。

```
digits.data[0].reshape(8,8)
```

```
array([[ 0.,  0.,  5., 13.,  9.,  1.,  0.,  0.],
       [ 0.,  0., 13., 15., 10., 15.,  5.,  0.],
       [ 0.,  3., 15.,  2.,  0., 11.,  8.,  0.],
       [ 0.,  4., 12.,  0.,  0.,  8.,  8.,  0.],
       [ 0.,  5.,  8.,  0.,  0.,  9.,  8.,  0.],
       [ 0.,  4., 11.,  0.,  1., 12.,  7.,  0.],
       [ 0.,  2., 14.,  5., 10., 12.,  0.,  0.],
       [ 0.,  0.,  6., 13., 10.,  0.,  0.,  0.]])
```

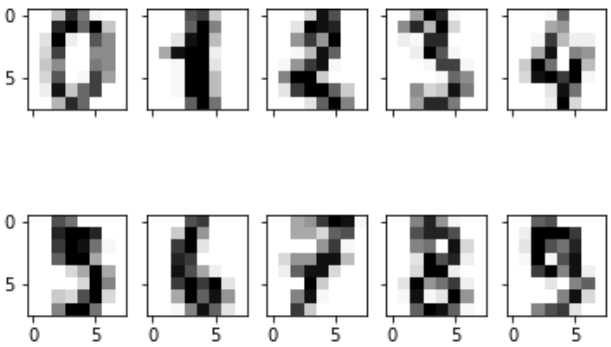
（在这颜色表现为灰度，其中数据为0的是白色，数据越大的对应灰色越深）

利用matplotlib中的imshow画图（画图方法在之前推送中有介绍）显示出数字集，可以发现数值形式效果可以表现出手写数字集。

```
fig, ax = plt.subplots(
        nrows=2,
        ncols=5,
        sharex=True,
        sharey=True, )

ax = ax.flatten()

for i in range(10):
    ax[i].imshow(digits.data[i].reshape(8,8), cmap='Greys',
    interpolation='nearest')
plt.show()
```



从SKlearn中调用KNN

接下来我们进一步探索，学习从Sklearn中调取KNN。

调取一个分类函数。（类）

```
from sklearn.neighbors import KNeighborsClassifier
```

数据初始化。（给“类”传入一个值形成一个KNN接受的对象，knn就代表KNN分类器初始化的结果）

```
knn=KNeighborsClassifier(10)
```

按照上文对KNN的介绍，在这里10的含义为k=10，即取距离预测目标最近的10个数据，这里的距离是指欧几米德距离，也就是我们熟悉的向量的距离。

通过fit函数返回模型参数。

```
knn.fit(Xtrain,Ytrain)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=10, p=2,
                     weights='uniform')
```

通过score函数给训练集打分。

```
knn.score(Xtrain, Ytrain)
```

```
0.9856801909307876
```

再通过score函数给测试集打分。

```
scores = knn.score(Xtest, Ytest)
scores
```

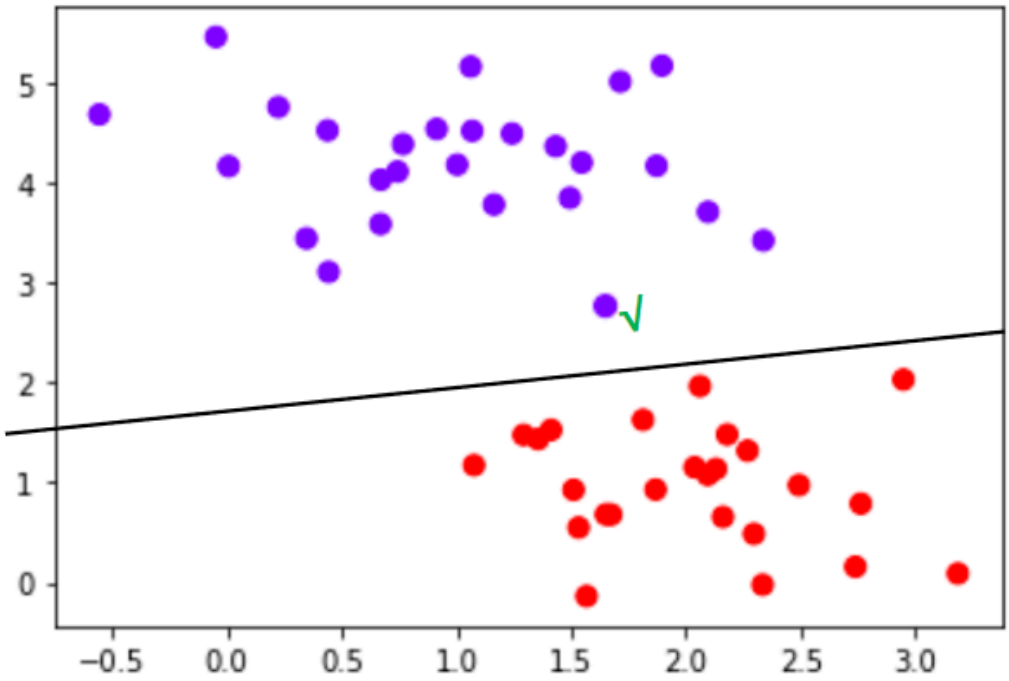
（score函数使用了predict接口）可以用测试集中模型预测正确的数量除以测试集数量来输出score。

```
Y_pre = knn.predict(Xtest)
(Y_pre == Ytest).sum() / Ytest.shape[0]
```

通过上述操作我们可以观察到训练集和测试集score的结果。
上述fit，predict，score函数形式是标准格式，我们在所有的算法中都能这样使用。

上述过程中，我们使用了训练集和测试集，这就涉及到了模型泛化的知识。对于为什么要划分训练集和测试集，我们在后面再进行解释。首先我们来认识一下模型泛化。

泛化是指：已经训练好的机器学习模型学习处理自己“已经见过的”数据和“没有见过”的数据的能力。



我们必须要明确的是，我们能给机器进行学习的数据，只是我们能搜集到一部分的历史数据，我们要的是基于历史数据训练出来的模型，能够提前预测出未来的情况。所以有时候根据历史数据可以推演出一套很好的逻辑，但是在预测未来时却漏洞百出，这种情况我们就成为模型泛化差。

模型泛化过程展示

从SKlearn中导入分割数据集。

```
from sklearn.model_selection import train_test_split
```

导入SKlearn中的乳腺癌数据集（二分类问题）。

```
from sklearn.datasets import load_breast_cancer
digits = load_breast_cancer()
```

设置X为数据，Y为标签，测试集大小0.3，随机种子801，返回四个值分别到Xtrain, Xtest, Ytrain, Ytest 。现在我们可以解释为什么要这样分割训练集和测试集了，因为为了评估模型泛化的能力，在这个例子中，我们先选70%的训练集让模型“看到”，而剩下模型“看不到”的30%的测试集就可以当成是未来的数据对模型的检验。

```
X, y = digits.data, digits.target
Xtrain, Xtest, Ytrain, Ytest = train_test_split(X, y,
test_size = 0.3, random_state = 801)
```

往往，我们习惯于多给一些数据来训练，从而当模型学到更多，这里的随机种子可以让我们在Xtrain、Ytrain中分得的结果相同，不同的随机种子可以获得不同的分割效果，从而让我们结果可以复现，这个数字按个人喜好设置。

创建test和train列表。

```
test_score = []
train_score = []
```

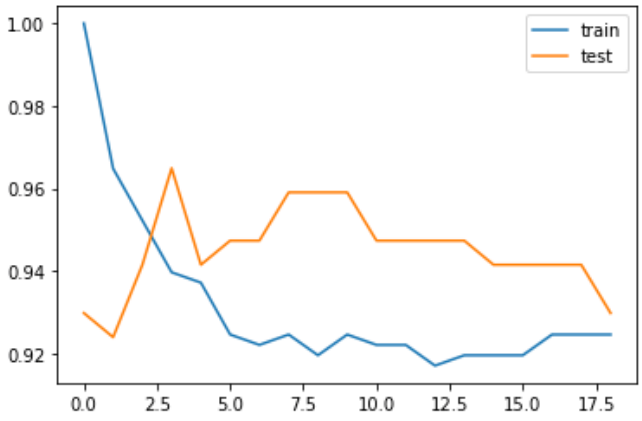
赋予i值1-20的循环（K为正整数），按照之前调取KNN的过程，实例化传入i值。fit后这里直接score打分。把Xtest、Ytest分数贴附到test列表，Xtrain、Ytrain分数贴附到train列表。

```
for i in range(1,20):
    knn=KNeighborsClassifier(i)
    knn.fit(Xtrain,Ytrain)
    test_score.append(knn.score(Xtest, Ytest))
    train_score.append(knn.score(Xtrain, Ytrain))
```

使用matplotlib绘图。

```
plt.plot(train_score, label='train')
plt.plot(test_score, label='test')
plt.legend()
```

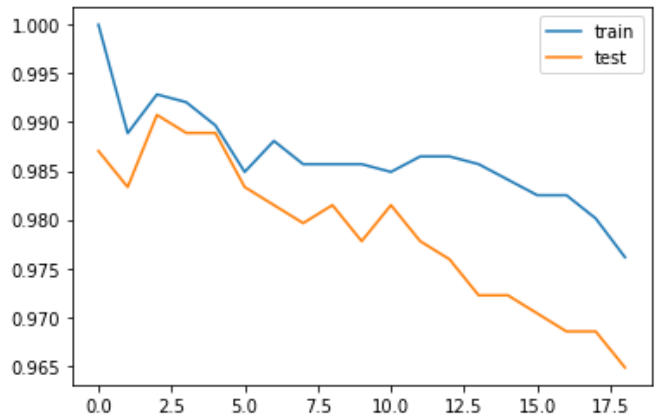
```
<matplotlib.legend.Legend at 0x2ee43780630>
```



我们发现随着K这个超参数提升，train是先降低，但是test升高了，这说明了如果K我们取特别低的时候，训练集的分数是很高，但是应对未来的测试集却得到了很差的结果。

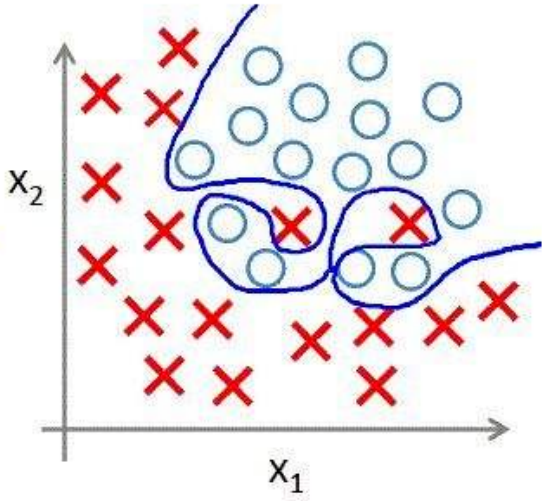
所以我们甘愿将K提升一点，让训练集的分数哪怕小点，但是应对未知的测试集也能得到比较好的结果，这也就是模型泛化的过程。由于K是正整数，所以K在2时我们可以获得最好的泛化效果。

当我们使用手写数字集数据代替乳腺癌数据时，就会得到下列图例。



我们发现这里test和train都在降低，就不存在泛化概念。

模型泛化还有一种表述就是过拟合。在训练模型中，当模型拟合程度太高，完全与训练集的数据切合，但是一换测试集就表现很差，我们就把这种情况叫做过拟合。



（如图中，过拟合情况下曲线完全区分了⭕和×）

在上述操作中我们对手写数字集数据和乳腺癌数据进行了操作，但在对股票数据进行预测时，由于股票受到多方面因素的影响，情况往往会变得更加复杂。
下面我们通过sklearn演示股票数据KNN预测。

通过sklearn演示股票数据KNN预测

我们可以通过KNN来进行构建投资策略。在股票方面，机器学习不仅需要数据的支撑，同时我们还需要思考它的经济意义.

下面我们通过股票“科大讯飞”进行演示，通过tushare返回的股票结果对股票下一天是否涨跌进行预测。

操作过程中，X为导入的数据，Y为下一天股票是否涨跌。
首先导入tushare,设置起始时间，输入“科大讯飞”股票代码（002230）。

```
import tushare as ts
s = '2018-10-01'
e = '2019-10-01'
kdxf = ts.get_hist_data('002230', start=s, end=e)
```

我们可以看到tushare返回的股票结果，包括开盘、最高、收盘、最低等结果。

kdxf

date	open	high	close	low	volume	price_change	p_change	ma5	ma10	ma20	v_ma5	v_ma10	v_ma20
2019-09-30	32.65	32.70	31.86	31.86	207633.30	-0.78	-2.39	33.026	33.297	33.984	297937.51	295729.26	364169.05
2019-09-27	32.55	33.08	32.64	32.46	173373.31	0.20	0.62	33.358	33.584	34.036	308640.69	300932.84	376871.19
2019-09-26	33.63	33.82	32.44	32.30	317300.81	-1.19	-3.54	33.600	33.778	34.076	326232.98	311098.88	384499.31
2019-09-25	34.26	34.46	33.63	33.42	300761.12	-0.93	-2.69	33.862	33.976	34.145	308507.14	316771.01	391020.42
2019-09-24	33.50	34.80	34.56	33.39	490619.00	1.04	3.10	33.806	34.132	34.159	291777.85	334736.88	414204.90
...	...	...	...	...	...	...	...	...	...	...	...	...	...
2018-10-12	23.68	23.68	23.11	21.83	568126.94	-1.10	-4.54	25.682	27.324	28.534	325952.63	302813.29	289198.89
2018-10-11	25.80	25.80	24.21	24.21	426746.41	-2.69	-10.00	26.774	28.007	28.960	254723.90	265983.76	280330.56
2018-10-10	27.17	27.35	26.90	26.16	216322.70	-0.25	-0.92	27.586	28.616	29.311	221500.06	256695.40	277750.95
2018-10-09	27.30	27.52	27.15	26.98	148314.19	0.11	0.41	27.998	28.902	29.558	228801.45	263359.91	286248.78
2018-10-08	27.88	28.06	27.04	27.00	270252.91	-1.53	-5.36	28.354	29.154	29.778	278063.96	273505.64	294995.18

243 rows × 13 columns

根据我们预测未来的目的，下面我们要构建x和y：

x - 所有的输入
y - 下一天是否涨跌

因为要处理数值，所以用values函数获得返回股票结果所包含的数值，再用shape函数显示它包含的行数和列数。

```
X = kdxf.values
X.shape
```

因为我们要进行的是分类问题，所以我们要为股票下一天是否涨跌构建标签，让涨（收盘价高于上一天收盘价）为1，跌（收盘价低于上一天收盘价）为0，我们要关注的是这只股票的收盘价（close），调取close。

kdxf['close']

接下来需要构建Y标签，可以将股票今日价格与上日价格相除结果取log的形式量化，这里使用了shift（1)让数据下移一位，以此构造今日价格除以上日价格的形式。由于上述下移操作，输出的y_第一位存在nan值（没有意义），我们可以使用y_[1:]将其去掉。

```
y_ = np.log(kdxf['close'] / kdxf['close'].shift(1)).values
y_.shape
```

下面用for、if、else语句对y_做出判断，为了解决次序的问题，可以利用enumerate给迭代器中添加索引形式。这里i为索引，j为真实值，利用y_[i]便可以取到对应的j值，做出判断。

```
for i,j in enumerate(y_):
    if j < 0:
        y_[i] = 0
    else:
        y_[i] = 1
```

让y_与X对应，因为之前shift（1）去掉第一位nan值的过程会造成y_计算时相比于X数量-1，所以这里为了让y_能一一正确对应上X，用x[:-1]可以不取X的最后一位。（X的第一位对应着y_第一位，类推）。

```
X_ = X[:-1]
```

做好上述了解，我们开始实际计算。
在之前科大讯飞股票数据中我们发现可以利用其中的price_change进行预测，大于0时股票涨（1），小于0时股票跌（0），用one_ or_ zero形式做出判断。

```
def one_or_zero(x):
    if x > 0:
        return 1
    else:
        return 0
```

用lambda函数把科大讯飞股票中price_change数值导入one_or_zero，可以输出y。

```
y = kdxf['price_change'].apply(lambda x : one_or_zero(x))
```

利用KNN分割数据集。

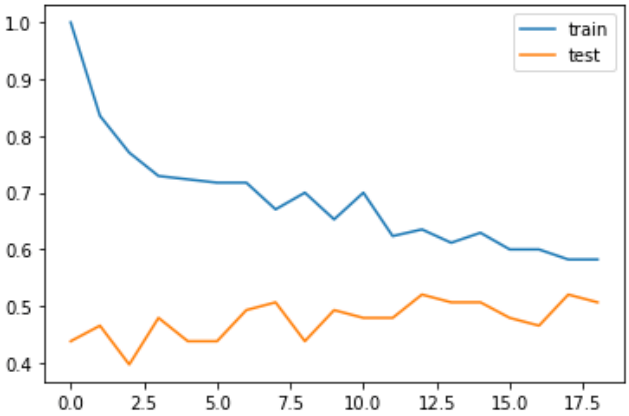
```
Xtrain, Xtest, Ytrain, Ytest = train_test_split(X, y, test_size = 0.3, rand
om_state = 801)
```

创建训练集、测试集、规定范围，从SKlearn中调用KNN 。（k为正整数）

```
test_score = []
train_score = []
for i in range(1,20):
    knn=KNeighborsClassifier(i)
    knn.fit(Xtrain,Ytrain)
    test_score.append(knn.score(Xtest, Ytest))
    train_score.append(knn.score(Xtrain, Ytrain))
```

最后使用matplotlib绘图。

```
plt.plot(train_score, label='train')
plt.plot(test_score, label='test')
plt.legend()
```



通过图例我们可以发现：
在K=1时，预测训练集时准确性可以达到90%，而预测测试集时的准确性只有40%左右。随着K的增大，训练集的分数在下降，测试集的分数有所上升，但是在波动。在K=12的时候，训练集和测试集都预测准确性高于50%，我们便可以选择这一点。虽然准确率没有那么高，但如果我们能够将准确率保持在51%，按照概率，我们在股票市场上反复投资也可以盈利。

总体来看，用来预测未来的测试集分数表现不高。这是因为和之前手写数字集相比，股票数据的影响因素很多，而且环境也在不停地改变。在手写数字集中，8*8的64个特征点大致是固定的，所以获得的值也会在固定的范围，预测的准确性会很高。而在股票中，特征因素太多，光有基础分析的指标是难以完全预测的，所以预测股票数据是十分困难的。

本期内容就到此结束了，希望能让大家对机器学习有所了解，让我们一起学习吧！Bye~

往期精彩内容

【Python】No.6 通过Tushare获取金融数据

【Python】No.5 Python在股票分析中的综合应用——金叉死叉交易策略与可视化

【Python】No.4 与三剑客无关的Python基础内容

【Python】No.3 Pandas的入门级教程

【Python】No.2 Numpy和Matplotlib初体验

【Python】No.1 Python安装部署和conda管理

本期撰写人：高镜骐
南京审计大学2018级CFA2班
邮箱：2215684404@qq.com