模型 预测模型80 常见的机器学习算法代码及其Python代码

模型视角 模型视角 2022-10-27 07:00 发表于上海

收录于合集
#模型 216 #预测模型 70 #机器学习 38 #Python 12 #数学建模 451

机器学习算法分类

一般来说,机器学习算法有以下三类。

1. 监督式学习算法
   这类算法由一个目标变量或结果变量(或因变量)组成。这些变量由已知的一系列预示变量(自变量) 预测而来。利用这一系列变量, 可以生成一个将输入值映射到期望输出值的函数。这个训练过程会一直持续, 直到模型在训练数据上获得期望的精确度。监督式学习的例子有：线性回归、决策树、随机森林算法、 K 最近邻算法、逻辑回归等。

2. 非监督式学习算法
   这类算法没有任何目标变量或结果变量要预测或估计。它用在不同的组内聚类分析。这种分析方式被广泛地用来细分客户, 根据干预的方式分为不同的用户组。非监督式学习的例子在：关联算法、 K 均值算法等。

3. 强化学习算法 这类算法训练机器进行决策。原理是这样的：机器被放在一个能让它通过反复试错来训练自己的环境中。机器从过去的经验中进行学习, 并且尝试利用了解最透彻的知识作出精确的商业判断。强化学习算法的例子有马尔可夫决策过程。

常见的机器学习算法及其 **Python** 代码

常见的机器学习算法有：(1) 线性回归；(2) 逻辑回归；(3) 决策树；(4)支持向量机 (SVM) 分类; (5) 朴素贝叶斯分类; (6)K 最近邻算法; (7)K 均值算法; (8) 随机森林算法; (9)降维算法: (10)Gradient Boost 和 Adaboost 算法。
下面我们对上面的机器学习算法逐一介绍, 并给出其主要的 Python 代码。

线性回归

线性回归通常用于根据连续变量估计实际数值(如房价、呼叫次数、总销售额等)。我们通过拟合最佳直线来建立自变量和因变量的关系。这条最佳直线叫作回归线, 并且用 $y = ax + b$ 这一

线性等式来表示。

假设在不问对方体重的情况下, 让一个五年级的孩子按体重从轻到重的顺序对班上的同学排序, 你觉得这个孩子会怎么做? 他很可能会目测人们的身高和体形,综合这些可见的参数来排列他们，这是现实生活中使用线性回归的例子.实际上,这个孩子发现了身高和体形、体重有一定的关系,这个关系看起来很像上面的等式。在这个等式中:y : 因变量;

x :自变量;

a : 斜率;

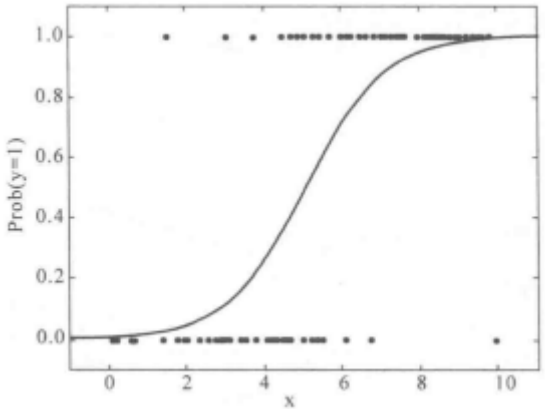
b : 截距。

系数 a 和 b 可以通过最小二乘法获得。

```
from sklearn.datasets import load_iris
X,y1 = load_iris().data[:,:2],load_iris().target
X,y2 = load_iris().data[:,:2],load_iris().data[:,2]

from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X,y2)
model.score(X,y2)
```

逻辑回归

这是一个分类算法, 而不是一个回归算法。该算法可根据已知的一系列因变量估计离散数值(如二进制数值 0 或 1 , 是或否, 真或假)。简单来说, 它通过将数据拟合进一个逻辑函数来预估一个事件出现的概率。因此, 它也被叫作逻辑回归。因为它预估的是概率, 所以它的输出值大小在 0 和 1 之间 (正如所预计的一样)。如下图所示



我们再通过一个简单的例子来理解这个算法。假设你朋友让你解开一个谜题。只会有两个结果：你解开了或是没有解开。想象你要解答许多道题来找出你所擅长的主题。这个研究的结果就会像是这样：假设题目是一道十年级的三角函数题, 你有 70% 的可能会解开这道题。然而, 若题目是道五年级的历史题, 你只有 30% 的可能性回答正确。这就是逻辑回归能提供给你的信

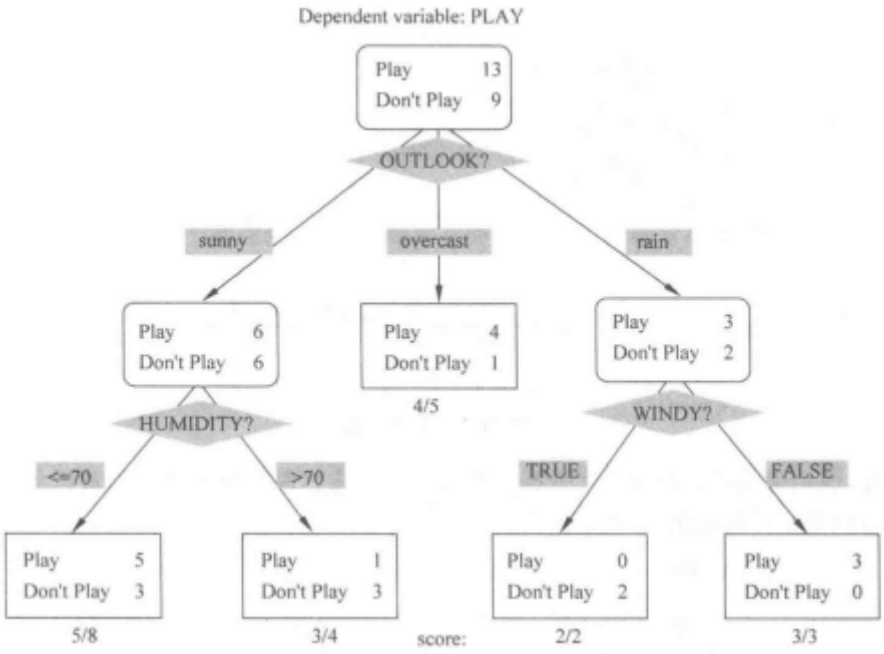
息。

从数学上看, 在结果中, 概率的对数使用的是预测变量的线性组合模型.

```
from sklearn.datasets import load_iris
X,y1 = load_iris().data[:,:2],load_iris().target
X,y2 = load_iris().data[:,:2],load_iris().data[:,2]

from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X,y1)
model.score(X,y1)
```

决策树

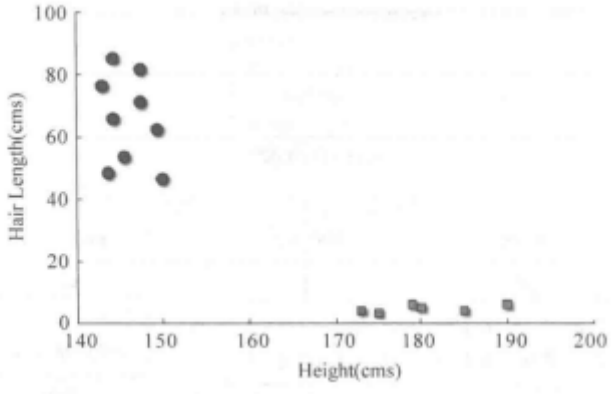
决策树这个监督式学习算法通常被用于分类问题。令人惊奇的是, 它同时适用于分类变量和连续因变量。在这个算法中, 我们将总体分成两个或更多的同类群。这是根据最重要的属性或者自变量来分成尽可能不同的组别。如图 所示。



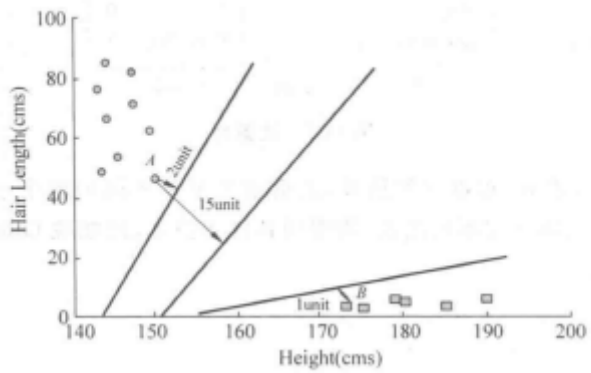

```
from sklearn.datasets import load_iris
X,y1 = load_iris().data[:,:2],load_iris().target
X,y2 = load_iris().data[:,:2],load_iris().data[:,2]

from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
model.fit(X,y1)
model.score(X,y1)
```

支持向量机(SVM) 分类

这是一种分类方法。在这个算法中, 我们将每个数据在 N 维空间中用点标出 (N 是所 有的特征总数), 每个特征的值是一个坐标的值。
例如, 如果我们只有身高和头发长度两个特征, 我们会在二维空间中标出这两个变量, 每个点有两个坐标 (这些坐标叫作支持向量), 如图所示。



现在,我们会找到将两组不同数据分开的一条直线。两个分组中距离最近的两个点到这条线的距离同时最优化,如图所示。



图中的黑线将数据分类优化成两个小组,两组中距离最近的点 (图中 A 点和 B 点)到达罟线的距离满足最优条件。这条直线就是我们的分割线。接下来,测试数据落到直线的哪一边, 我们就将它分到哪一类去。

```
from sklearn.datasets import load_iris
X,y1 = load_iris().data[:,:2],load_iris().target
X,y2 = load_iris().data[:,:2],load_iris().data[:,2]

from sklearn.svm import SVC
model = SVC()
model.fit(X,y1)
model.score(X,y1)
```

朴素贝叶斯分类

在预示变量间相互独立的前提下, 根据贝叶斯定理可以得到朴素贝叶斯这个分类方法。用更简单的话来说,一个朴素贝叶斯分类器假设一个分类的特性与该分类的其他特性不相关。举例来说,如果一个水果又圆又红并且直径大约是 8 cm, 那么这个水果可能会是苹果。即便这些特性互相依赖或者依赖于别的特性存在, 朴素贝叶斯分类器还是会假设这些特性 分别独立地暗示这个水果是个苹果。

朴素贝叶斯模型易于建造, 且对于大型数据集非常有用。虽然简单, 但是朴素贝叶斯的 表现却超越了非常复杂的分类方法。贝叶斯定理提供了一种从 $P(c)$ 、 $P(x)$ 和 $P(x \mid c)$ 计算后验概率 $P(c \mid x)$ 的方法。请看以下等式：

这里, $P(c \mid x)$ 是已知预示变量 (属性) 的前提下, 类 (目标) 的后验概率, $P(c)$ 是类的先验概率, $P(x \mid c)$ 是可能性, 即已知类的前提下, 预示变量的概率, $P(x)$ 是预示变量的先验概率。

> 例题 设有一个天气的训练集和对应的目标变量“Play”。我们需要根据天气情况, 将 “玩”和“不玩”的参与情况进行分类。执行步骤如下。

步骤 1：把数据集转换成频率表。

步骤 2: 利用类似“当 Overcast 可能性为 0.29 时, 玩(Play) 的可能性为 0.64 ”这样的概率, 创造 Likelihood 表格。

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

Likelihood table				
Weather	No	Yes		
Overcast		4	=4/14	0.29
Rainy	3	2	=5/14	0.36
Sunny	2	3	=5/14	0.36
All	5	9		
	=5/14	=9/14		
	0.36	0.64		

步骤 3：使用朴素贝叶斯公式来计算每一类的后验概率。后验概率最大的类就是预测结果。

问题：如果天气晴朗, 参与者就能玩。这个陈述正确吗?

我们可以使用讨论过的方法解决这个问题。于是 P (会玩|晴朗) $=P$ (晴朗|会玩) X P(会玩)/P (晴朗)

我们有 P (晴朗 | 会玩) $=3/9=0.33, P($ 晴朗 $)=5/14=0.36, P($ 会玩 $)=9/14=0.64$ 。算法通常被用于文本分类, 以及涉及多个类的问题。

```
from sklearn.datasets import load_iris
X,y1 = load_iris().data[:,:2],load_iris().target
X,y2 = load_iris().data[:,:2],load_iris().data[:,2]

from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(X,y1)
model.score(X,y1)
```

K 最近邻 (KNN) 算法

K 最近邻算法 (K-Nearest Neighbor), 又称 KNN 算法。该算法可用于分类问题和回归问题。然而, 在业界内, K 最近邻算法更常用于分类问题。K 最近邻算法是一个简单的算 法。它储存所有的案例, 通过周围 K 个案例中的大多数情况划分新的案例。根据一个距离函数, 新案例会被分配到它的 K 个近邻中最普遍的类别中.这距离函数可以是欧式距变、曼哈顿距离、明氏距离或者是汉明距离。前三个距离函数用于分类变量。如果 $K=1$, 新案例就直接被分到离其最近的案例所属的类别中。有时侈,使用 KNN 建模时, 选择 K 的取值是一个挑战。在现实生活中广泛地应用了 K 最邻近算法。例如, 想要了解一个完全陌生的人, 你也许想要去找他的好朋友们或者他的囡子来获得他的信息。

```
from sklearn.datasets import load_iris
X,y1 = load_iris().data[:,:2],load_iris().target
X,y2 = load_iris().data[:,:2],load_iris().data[:,2]

from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier()
model.fit(X,y1)
model.score(X,y1)
```

K 均值算法

K 均值算法是一种非监督式学习算法, 它能解决聚类问题。使用 K 均值算法来将一个数据归入一定数量的集群(假设有 K 个集群)的过程是简单的。一个集群内的数据点是均匀齐次的,并且异于别的集群。
K 均值算法形成集群的原理如下:

1. K 均值算法给每个集样选择 K 个点。这些点称作为质心。
2. 每一个数据点与距离最近的质心形成一个集群, 也就是 K 个集群。
3. 根据现有的类别成员, 找到每个类别的质心。现在我们有了新质心。

4. 当我们有新质心后,重复步骤 2 和步骤 3 。找到距离每个数据点最近的质心, 并与新 的 k 集群联系起来。重复这个过程, 直到数据都收敛了, 也就是质心不再改变。

如何决定 K 值？K 均值算法涉及集群, 每个集群有自己的质心。一个集群内的质心和各数据点之间距 离的平方和形成了这个集群的平方值之和。同时, 当所有集群的平方值之和加起来的时候, 就组成了集群方案的平方值之和。
我们知道, 当集群的数量增加时, K 值会持续下降。但是, 如果你将结果用图表来表 示,你会看到距离的平方总和快速减少。到某个值 K 之后, 减少的速度就大大下降了。在 此. 我们可以找到集群数量的最优值。

```
from sklearn.datasets import load_iris
X,y1 = load_iris().data[:,:2],load_iris().target
X,y2 = load_iris().data[:,:2],load_iris().data[:,2]

from sklearn.cluster import KMeans
model = KMeans(n_clusters=2)
model.fit(X)
model.predict(X)
```

随机森林算法

随机森林是表示决策树总体的一个专有名词在随机森林算法中, 我们有一系列的决策树 (因此又名“森林”)。为了根据一个新对象的属性将其分类, 每一个决策树有一个分类, 称之为这个决策树“投票”给该分类。这个森林选择获得森林里(在所有树中) 获得票数最多的分类。

```
from sklearn.datasets import load_iris
X,y1 = load_iris().data[:,:2],load_iris().target
X,y2 = load_iris().data[:,:2],load_iris().data[:,2]

from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(X,y1)
model.score(X,y1)
```

降维算法

近年来, 信息呈指数增长。电子商务公司更详细地捕捉关于顾客的资料：个人信息、网络浏览记录、他们的喜恶、购买记录、反馈以及别的许多信息, 比你身边的杂货店售货员更加关注你。

作为一个数据科学家,我们提供的数据包含许多特点。这听起来给建立一个经得起考 验的模型提供了很好的材料,但有一个挑战：如何从 1000 或者 2000 个变量中分辨出出最重 要的变量呢? 在这种情况下,降维算法和别的一些算法(比如决策树、随机森林、PCA、因子 分析)帮助我们根据相关矩阵、缺失的值的比例和别的要素来找出这些重要变量。

```
from sklearn.datasets import load_iris
X,y1 = load_iris().data[:,:2],load_iris().target
X,y2 = load_iris().data[:,:2],load_iris().data[:,2]

from sklearn.decomposition import PCA
model = PCA(n_components=1)
model.fit(X)
model.transform(X)
```


参考资料：

- 朱顺全 经济金融数据分析及其Python应用

收录于合集 #模型 216

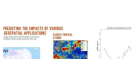
上一篇
模型 预测模型76 复杂模型事后解析方法之部分依赖图

下一篇
模型 评价模型40 体育课程学习评价的实施过程

喜欢此内容的人还喜欢

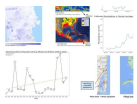
促进"深度学习"的五个阶段
数之音

Google Earth Engine（GEE）——通过地理空间应用分析、可视化和预



测各种自然灾害的影响

我也想吃a



一文入门XGBoost建模

用Python学机器学习