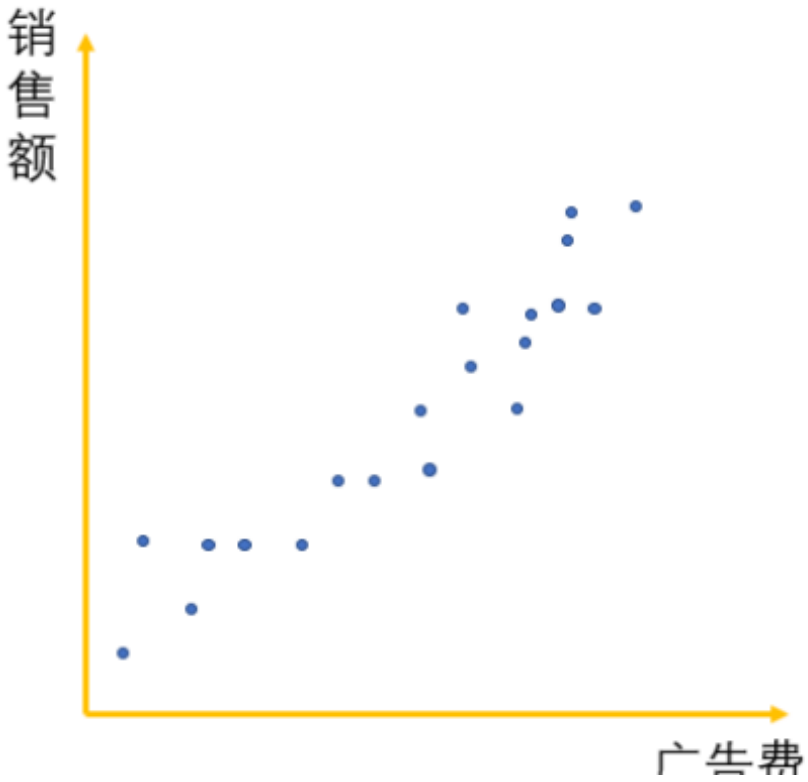
经过了这么久的学习，我们终于结束了分类和聚类算法的相关内容，这一课时我将为你讲解关于回归算法的内容。

从标题可以看出来，我们这次课程会涉及线性回归和逻辑回归，这两个回归有什么样的含义呢？虽然都叫作回归，它们的处理方式有什么不同呢？带着这些问题，我们就开始本课时的学习吧。

一个例子

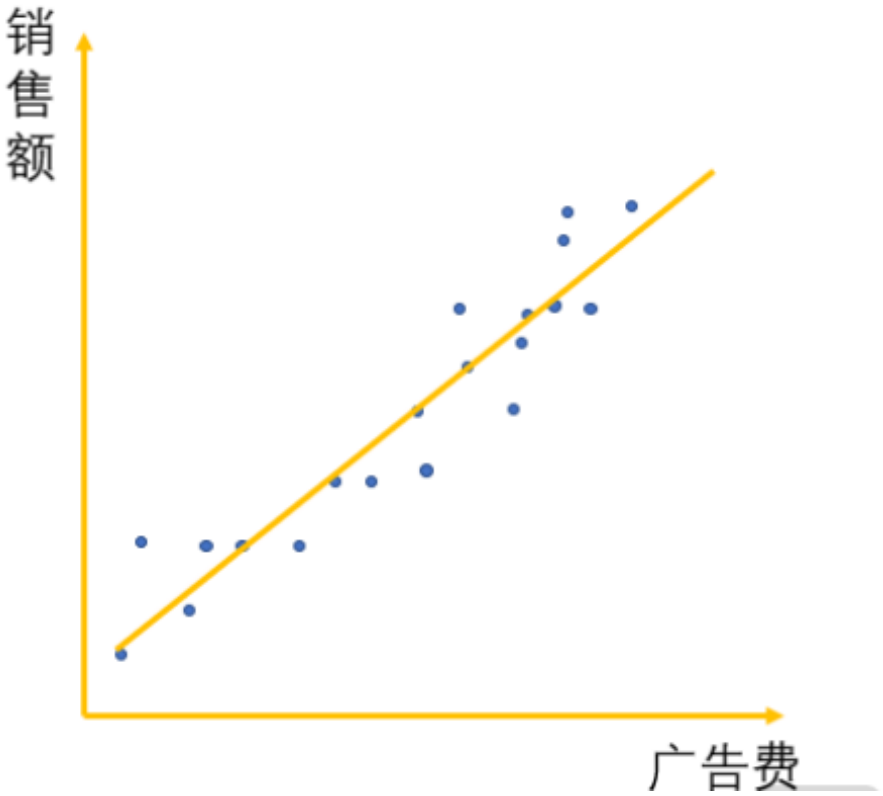
我们还是先从一个例子出发。想象你已经是一家公司的 CEO，你的公司旗下有着优质的明星产品——一种新型的保健品“星耀脑黄金”。为了让你的产品卖得更好，你要到处去投放广告，让大家都知道这个产品，激发大家购买的欲望。我们都知道投放广告是要花钱的，投放得越多，钱花得越多；知道的人越多，产品卖得越多。

根据历史累计的广告投放经费和销售额，我们可以画出一些点。



从这个图上可以看出，有些点位的收益相对较高，有些点位的收益相对较低，但是总体是一个正相关的关系。很自然地，你内心感觉需要画出一条线，就如下面图中的一样，来拟合投放广告经费和销售额的关系。

虽然它对于每一个单点来说都不是那么精确，但是它却十分简洁，有了这条线，你只需要设定一个广告费的数额，就一定可以算出一个销售额。这样当你在开股东大会的时候，就可以跟大家说：“我花这么多广告费是有价值的，只要我们的投放广告费达到多少多少亿，销售额也就能达到多少多少亿。”



经过了上面的步骤，我们其实就已经完成了线性回归的模型构建，即根据已有的数据去寻找一条直线，尽量接近这些数据，以用于对以后的数据进行预测，但是具体该怎么去找这条线呢？接下来我们看一下线性回归的原理。

线性回归的算法原理

首先我们要明确一下，什么是线性，什么是非线性。

线性： 结果与特征之间是一次函数关系，比如表现在上面的例子中就是一条直线。

非线性： 结果与特征之间不是一次函数关系，比如二次函数、三次函数，表现在图中是一条曲线，就是非线性的。

明白了这两个词，我们再来看回归模型是如何构建的。

比如在我们的例子中只有一个变量就是广告费，一个结果值是销售额。我们假设它们符合线性关系，那么我们先预设一个方程：

$$y=ax+b$$

所以我们只要根据数据求出 a 和 b 就完成了。

但是这里有一个困难，那就是我们的数据不是完全分布在这条线上的，那就需要一个方法来评估每次生成的线的效果，然后去进行相应的调整，以达到最好的效果。提到优化，这里又需要引入两个概念，让我慢慢介绍。

损失函数： 不要被这个高大上的名称吓到，用一句话来解释，就是计算每一个样本点的结果值和当前的函数值的差值。当然具体到这里面，所使用的是残差平方和（Sum of Squares for Error），这是一种最常用的损失函数。如果你对具体的公式感兴趣，可以在网上查到它的具体信息。

最小二乘法： 知道了损失函数，只要有一条线，我们就可以通过损失函数来计算假设结果为这条线的情况下，损失值的大小。而这里的最小二乘法就是要找到一组 a、b 的值，使得损失值达到最小。这里的二乘就是平方的意思。

到这里是不是对算法原理的理解就清晰很多了呢？那么我们再来看下它有哪些优缺点。

线性回归的优缺点

优点

- **运算速度快。**由于算法很简单，而且符合非常简洁的数学原理，不管是建模速度，还是预测速度都是非常快的。
- **可解释性强。**由于最终我们可以得到一个函数公式，根据计算出的公式系数就可以很明确地知道每个变量的影响大小。
- **对线性关系拟合效果好。**当然，相比之下，如果数据是非线性关系，那么就不合适了。

缺点

- **预测的精确度较低。**由于获得的模型只是要求最小的损失，而不是对数据良好的拟合，所以精确度略低。
- **不相关的特征会影响结果。**对噪声数据也比较难处理，所以在数据处理阶段需要剔除不相关的特征以及噪声数据。
- **容易出现过拟合。**尤其在数据量较少的情况下，可能出现这种问题。

逻辑回归（Logistic Regression）

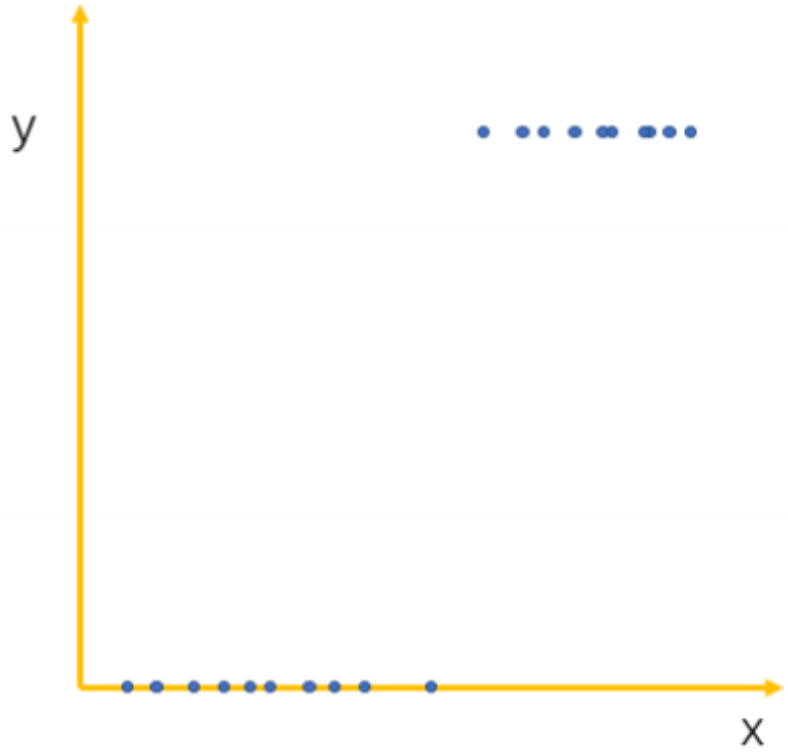
到这里，你可能比较疑惑，为什么我标题里写的线性回归与逻辑回归，而上面一直在讲线性回归？别担心，我们马上就讲到逻辑回归的相关内容。

如果你已经了解了上面的线性回归，其实你就已经掌握了回归的方法，对于不同的回归算法，只不过是把对应的函数进行替换，把线性方程替换成非线性方程，或者其他各种各样的方程，以便更好地拟合数据。

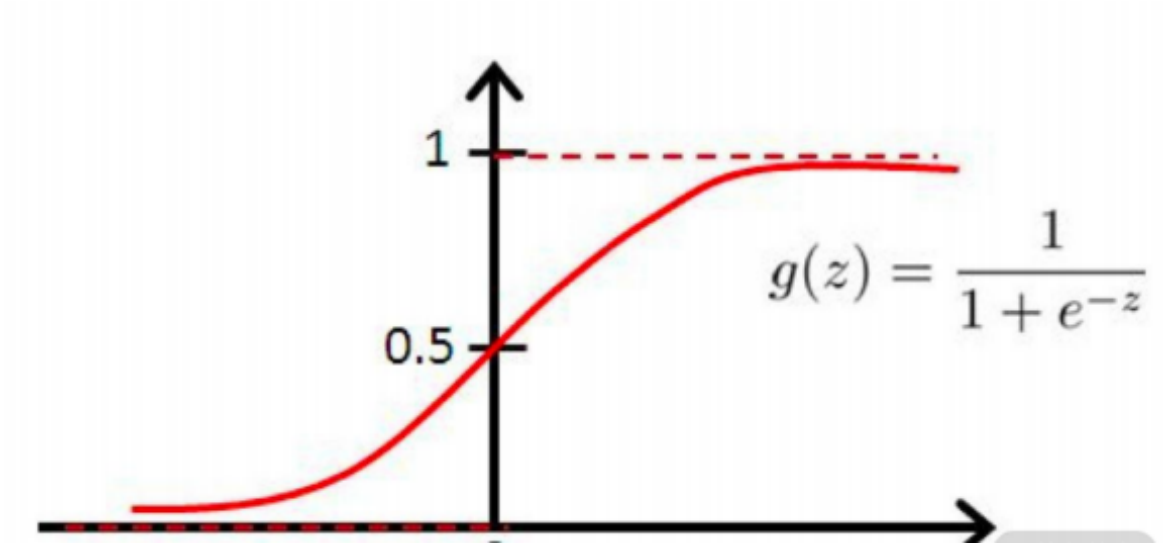
而这里为什么要把逻辑回归拎出来呢？因为我们常说的回归算法是对比分类算法来说的，回归与分类有很多相似性，是有监督学习的两大分支。我们前面也介绍过，它们的区别是分类算法输出的是离散的分类结果，而回归算法输出的是连续数值结果，这两种结果可以通过一定的方法进行转换。

而逻辑回归就是这样一个典型的例子。它虽然叫作回归，但实际上却是用来解决分类问题，只不过在中间过程中，它使用的是回归的方法。

关于分类问题，假设一个二分类问题，只有一个变量 x 会引起结果标签的变化，那么把它俩表示在平面上就是下图这样的效果，前半部分的结果都是 0，后半部分的结果都是 1。



在逻辑回归算法中，使用了 sigmoid 函数来拟合数据。可以看出 sigmoid 函数有点像是被掰弯的线性函数直线，这样函数的取值范围被限定在了 0 和 1 之间，很明显，使用这个函数去拟合上面的二分类结果要比线性直线好得多。



（图片来源于百度）

当然，选择 sigmoid 曲线并不是偶然的，而是通过了精密的计算之后得出的方案。这里涉及了概率对数函数（Logit）的数学推导，这也是逻辑回归名字的来源。同时，在输出结果的时候，借助了极大似然估计的方法来对预测出的结果进行损失的估计。

当然，逻辑回归里所涉及的内容还有很多，其中使用了大量的数学推导，这里就不再做过多的介绍，如果你对推导过程感兴趣，可以进行更深入的学习。下面我们进入到动手环节，尝试在代码中使用线性回归来解决问题。

尝试动手

在代码环节，今天我们仍然尝试自己来生成数据。

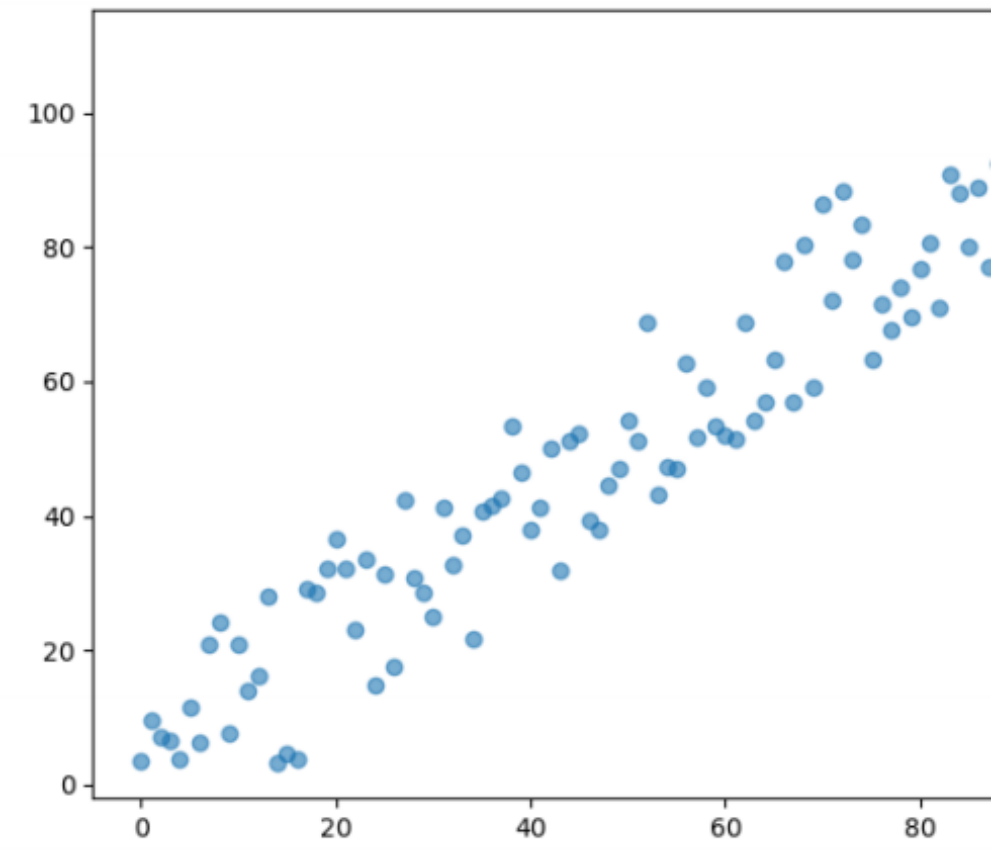
首先，仍然是引入我们需要用到的包。

```
import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression

from sklearn.model_selection import train_test_split

import numpy as np
```

然后，我们在这里生成数据。我假设了我们的数据偏移量为 2.128，并且生成了 100 个点，作为我们的样本数据。

```
def generateData():

    X = []

    y = []

for i in range(0, 100):

        tem_x = []

        tem_x.append(i)

        X.append(tem_x)

        tem_y = []

        tem_y.append(i + 2.128 + np.random.uniform(-15,15))

        y.append(tem_y)

    plt.scatter(X, y, alpha=0.6)

return X,y
```

生成完的数据可以在下面的图中看到。

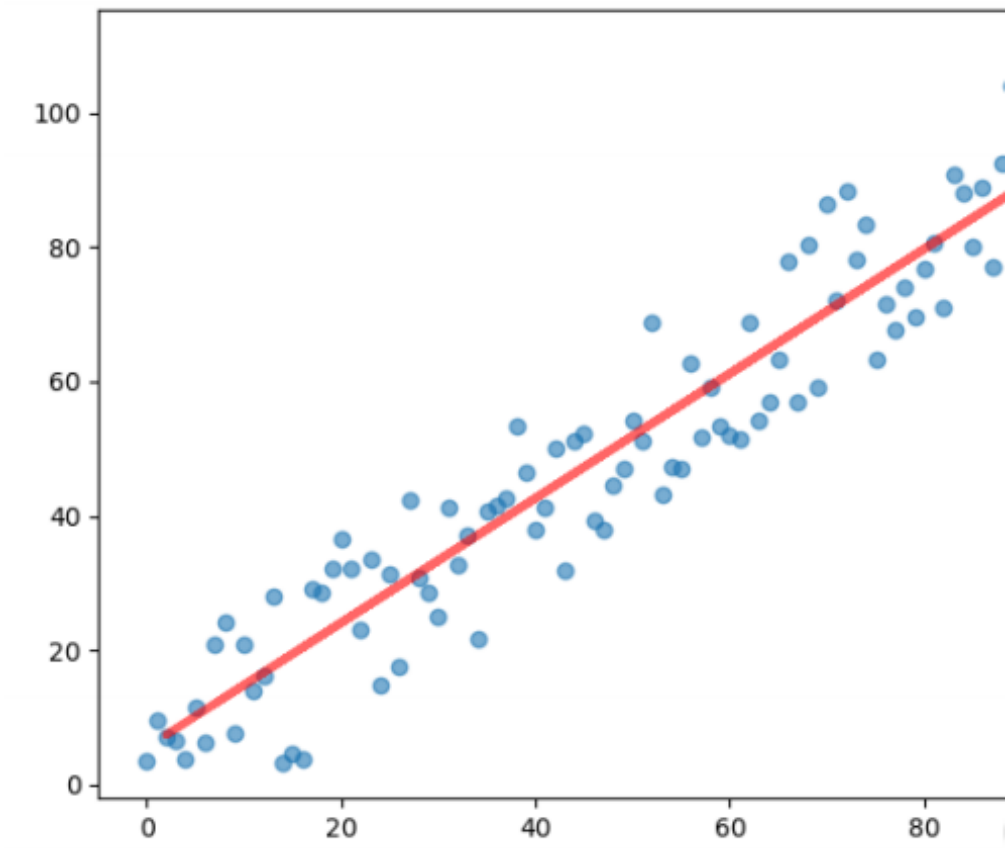


在我们的主方法中，首先使用生成样本的方法生成了我们的数据，然后这次使用了 sklearn 中自带的数据切割方法对数据进行了切分，80% 作为训练集、20% 作为测试集。然后调用了线性回归算法，并使用预测方法对测试数据进行预测。

```
if __name__ == '__main__':

    np.random.seed(0)

    X,y = generateData()

    print(len(X))

    X_train,X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=0)

    regressor = LinearRegression()

    regressor.fit(X_train, y_train)

    y_result = regressor.predict(X_test)

    plt.plot(X_test, y_result, color='red',alpha=0.6, linewidth=3,
label='Predicted Line')
```

```
    plt.show()
```

在最后，根据预测的结果绘制出了算法学习到的直线，如下图显示：



通过上面的代码，我们成功实现了使用线性回归算法来学习数据的规律，并拟合出一条直线。当新数据来了之后，直接把样本数据套入这个直线的方程中，就可以计算出结果。

总结

完成了动手环节，让我们再来回顾一下本课时的重点内容。在这节课中，我们介绍了回归方法，其中主要讲解了线性回归，同时简单介绍了逻辑回归。它俩虽然都有 “回归” 这个字眼，却存在着一些区别，当然，也有着一些相似。然后我们借助工具包实现了线性回归的代码调用，并绘制了相应的图像来展示回归的效果。

回归方法是非常常用的数据分析和数据挖掘方法，它的原理简单、运行快速，在很多数值型的预测需求中都发挥着巨大的价值。当然，除了这一小节中讲的线性回归和逻辑回归，还有很多不同的回归方程可以使用，以解决不同的问题。