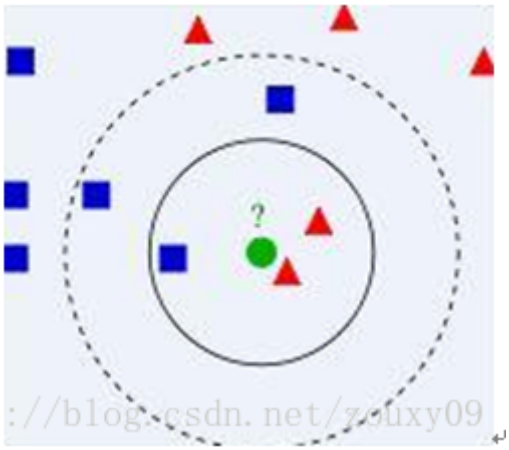
KNN

1. 算法原理

K 最近邻（k-Nearest Neighbor，KNN）分类算法可以说是最简单的机器学习算法了。它采用测量不同特征值之间的距离方法进行分类。它的思想很简单：如果一个样本在特征空间中的 k 个最相似（即特征空间中最邻近）的样本中的大多数属于某一个类别，则该样本也属于这个类别。



比如上面这个图，我们有两类数据，分别是蓝色方块和红色三角形，他们分布在一个上图的二维中间中。那么假如我们有一个绿色圆圈这个数据，需要判断这个数据是属于蓝色方块这一类，还是与红色三角形同类。怎么做呢？我们先把离这个绿色圆圈最近的几个点找到，因为我们觉得离绿色圆圈最近的才对它的类别有判断的帮助。那到底要用多少个来判断呢？这个个数就是 k 了。如果 k=3，就表示我们选择离绿色圆圈最近的 3 个点来判断，由于红色三角形所占比例为 2/3，所以我们认为绿色圆是和红色三角形同类。如果 k=5，由于蓝色四方形比例为 3/5，因此绿色圆被赋予蓝色四方形类。从这里可以看到，k 的值还是很重要的。

该算法在分类时有个主要的不足是，当样本不平衡时，如一个类的样本容量很大，而其他类样本容量很小时，有可能导致当输入一个新样本时，该样本的 K 个邻居中大容量类的样本占多数。因此可以采用权值的方法（和该样本距离小的邻居权值大）来改进。该方法的另一个不足之处是计算量较大，因为对每一个待分类的文本都要计算它到全体已知样本的距离，才能求得它的 K 个最近邻点。目前常用的解决方法是事先对已知样本点进行剪辑，事先去除对分类作用不大的样本。该算法比较适用于样本容量比较大的类域的自动分类，而那些样本容量较小的类域采用这种算法比较容易产生误分[参考机器学习十大算法]。

总的来说就是我们已经存在了一个带标签的数据库，然后输入没有标签的新数据后，将新数据的每个特征与样本集中数据对应的特征进行比较，然后算法提取样本集中特征最相似（最近邻）的分类标签。一般来说，只选择样本数据库中前 k 个最相似的数据。最后，选择 k 个最相似数据中出现次数最多的分类。其算法描述如下：

1）计算已知类别数据集中的点与当前点之间的距离；

2）按照距离递增次序排序；

3）选取与当前点距离最小的 k 个点；

4）确定前 k 个点所在类别的出现频率；

5）返回前 k 个点出现频率最高的类别作为当前点的预测分类。

KNN 做回归和分类的主要区别在于最后做预测时候的决策方式不同。KNN 做分类预测时，一般是选择多数表决法，即训练集里和预测的样本特征最近的 K 个样本，预测为里面有最多类别数的类别。而 KNN 做回归时，一般是选择平均法，即最近的 K 个样本的样本输出的平均值作为回归预测值。由于两者区别不大，虽然本文主要是讲解 KNN 的分类方法，但思想对 KNN 的回归方法也适用。由于 scikit-learn 里只使用了蛮力实现(brute-force)，KD 树实现(KDTree)和球树(BallTree)实现，本文只讨论这几种算法的实现原理。其余的实现方法比如 BBF 树，MVP 树等，在这里不做讨论。

2. 实验目的

通过实验了解 KNN 算法的思想

3. 实验环境

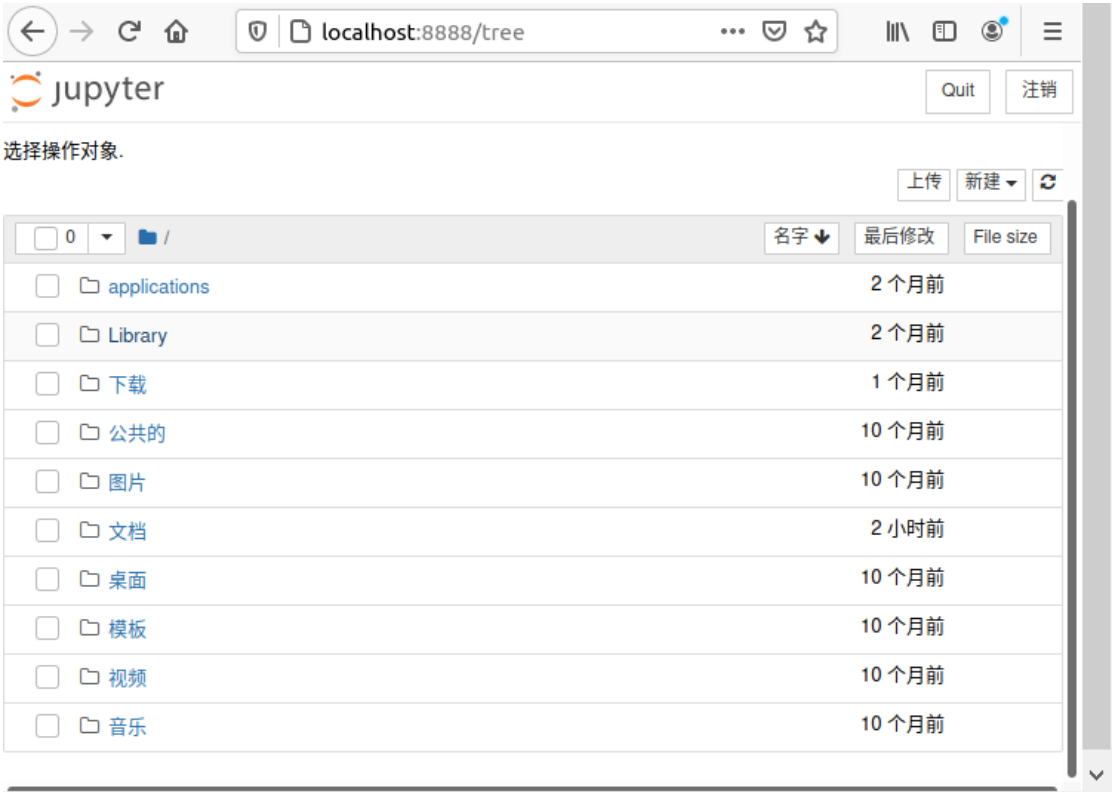
Ubuntu 20.04

Python 3.6

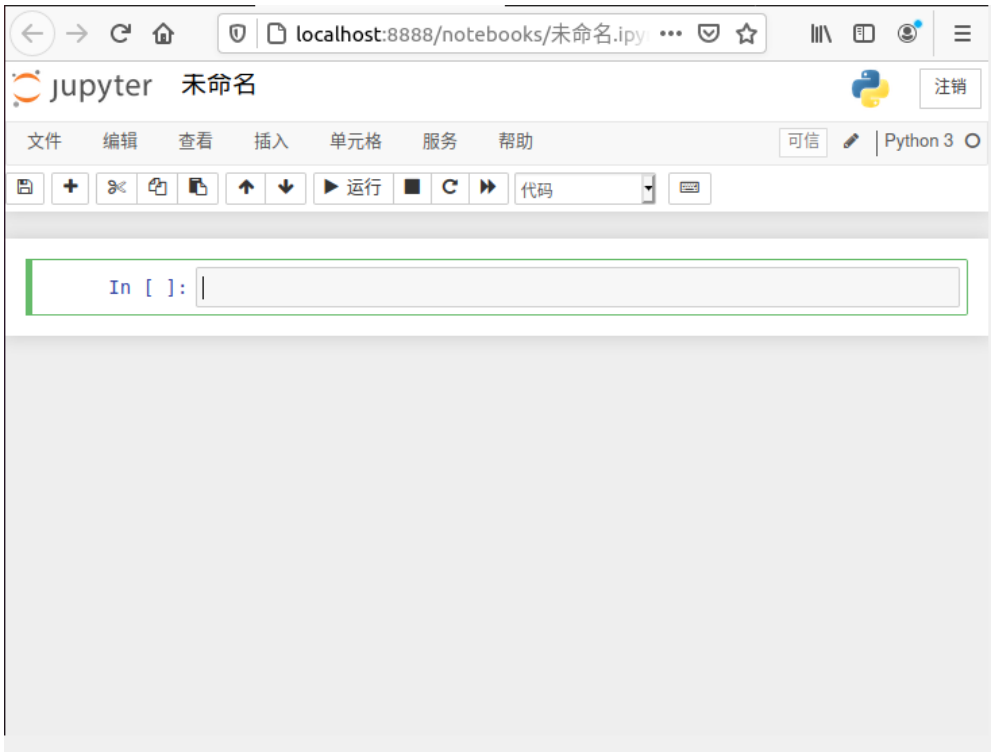
Jupyter notebook

4. 实验步骤

1)打开终端，然后输入 jupyter notebook，出现如下界面



2)选定特定文件夹，新建 ipynb 文件，在未命名出可重命名文件



5. 实操

6. Step 1：导入库

```
7. import numpy as np
8. import matplotlib.pyplot as plt
9. import pandas as pd
10.
```

11. Step 2：导入数据集

```
12.dataset = pd.read_csv('Social_Network_Ads.csv')
13.
```

14. Step 3：查看数据集

```
15.X = dataset.iloc[:, [2, 3]].values
16.y = dataset.iloc[:, 4].values
17.
```

18. Step 4：切分数据集

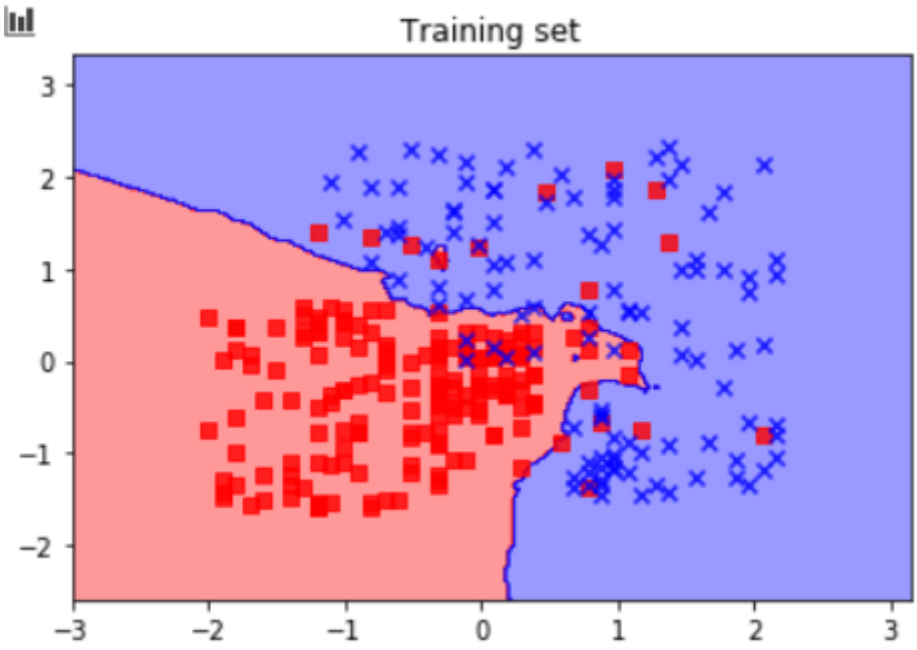
```
19.from sklearn.model_selection import train_test_split
20.X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
    = 0.25, random_state = 0)
21.
```

22. Step 5：特征缩放

```
23.from sklearn.preprocessing import StandardScaler
24.sc = StandardScaler()
25.X_train = sc.fit_transform(X_train)
26.X_test = sc.transform(X_test)
27.
```

28. Step 6：训练集上应用 KNN

```
29. from sklearn.neighbors import KNeighborsClassifier
30. classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
31. classifier.fit(X_train, y_train)
32.
33. Step 7: 预测测试集结果
34. y_pred = classifier.predict(X_test)
35.
36. Step 8: 可视化
37. plot_decision_regions(X_train, y_train, classifier=classifier)
38. plt.title("Training set")
```




```
39.
40.
41. plot_decision_regions(X_test, y_pred, classifier=classifier)
42. plt.title("Test set")
```

43.