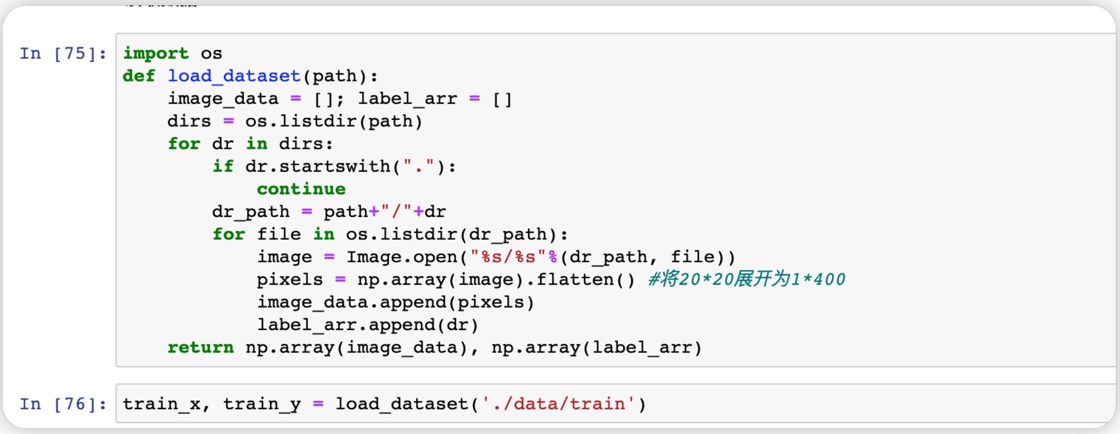
实验四

基于k近邻的车牌识别

2022年7月13日

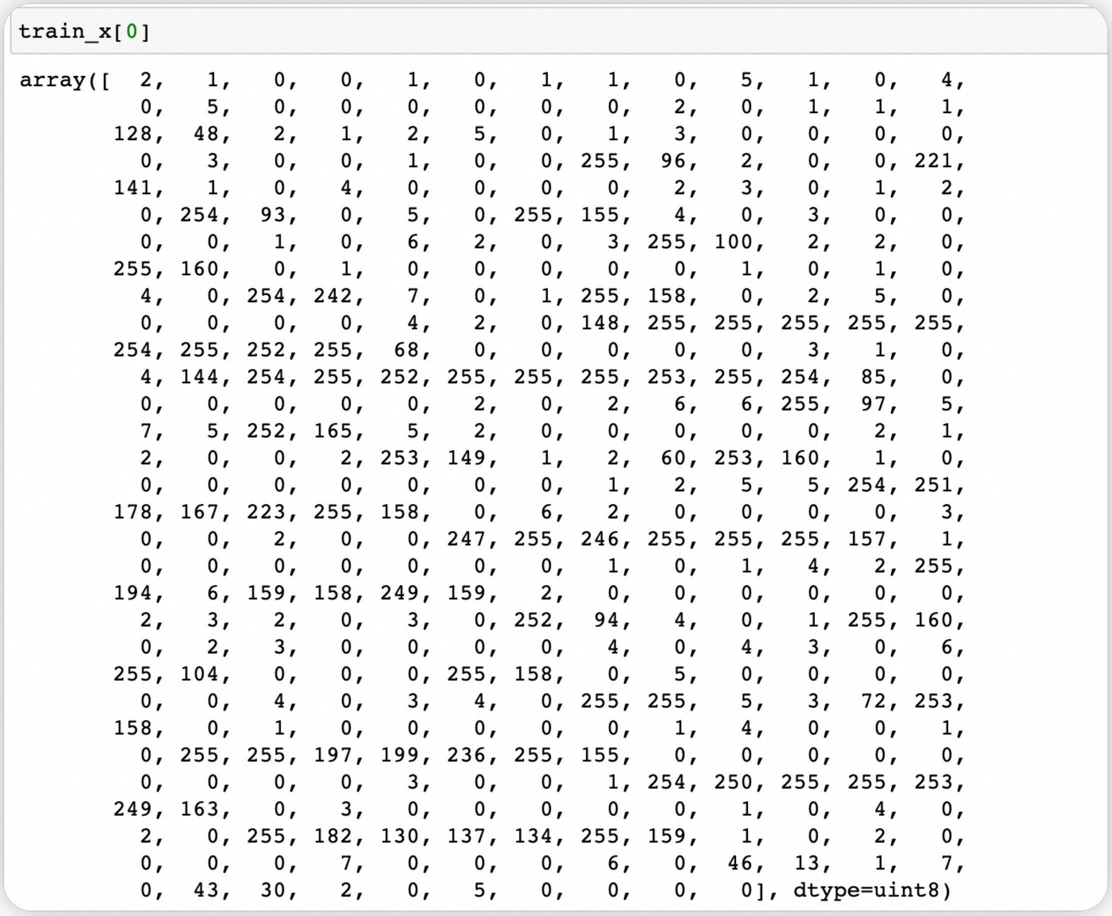
1. 读取数据



```
In [75]: import os
         def load_dataset(path):
             image_data = []; label_arr = []
             dirs = os.listdir(path)
             for dr in dirs:
                 if dr.startswith("."):
                     continue
                 dr_path = path+"/"+dr
                 for file in os.listdir(dr_path):
                     image = Image.open("%s/%s"%(dr_path, file))
                     pixels = np.array(image).flatten() #将20*20展开为1*400
                     image_data.append(pixels)
                     label_arr.append(dr)
             return np.array(image_data), np.array(label_arr)

In [76]: train_x, train_y = load_dataset('./data/train')
```

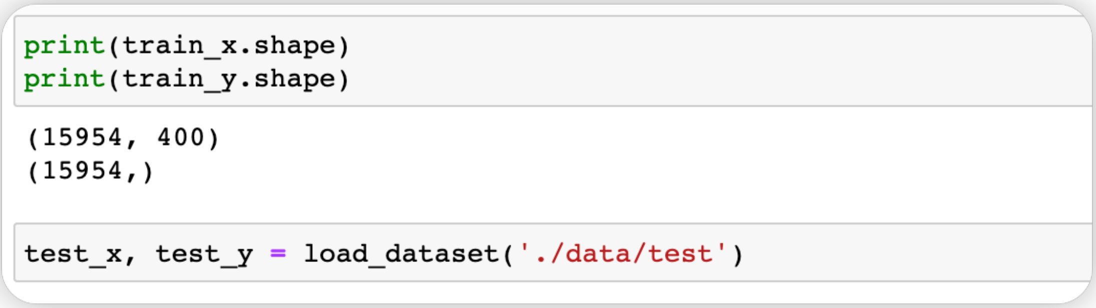
训练数据如下所示，每个取值为像素点。



```
train_x[0]
```

```
array([  2,   1,   0,   0,   1,   0,   1,   1,   0,   5,   1,   0,   4,
         0,   5,   0,   0,   0,   0,   0,   0,   2,   0,   1,   1,   1,
       128,  48,   2,   1,   2,   5,   0,   1,   3,   0,   0,   0,   0,
         0,   3,   0,   0,   1,   0,   0, 255,  96,   2,   0,   0, 221,
       141,   1,   0,   4,   0,   0,   0,   0,   2,   3,   0,   1,   2,
         0, 254,  93,   0,   5,   0, 255, 155,   4,   0,   3,   0,   0,
         0,   0,   1,   0,   6,   2,   0,   3, 255, 100,   2,   2,   0,
       255, 160,   0,   1,   0,   0,   0,   0,   0,   1,   0,   1,   0,
         4,   0, 254, 242,   7,   0,   1, 255, 158,   0,   2,   5,   0,
         0,   0,   0,   0,   4,   2,   0, 148, 255, 255, 255, 255, 255,
       254, 255, 252, 255,  68,   0,   0,   0,   0,   0,   3,   1,   0,
         4, 144, 254, 255, 252, 255, 255, 255, 253, 255, 254,  85,   0,
         0,   0,   0,   0,   0,   2,   0,   2,   6,   6, 255,  97,   5,
         7,   5, 252, 165,   5,   2,   0,   0,   0,   0,   0,   2,   1,
         2,   0,   0,   2, 253, 149,   1,   2,  60, 253, 160,   1,   0,
         0,   0,   0,   0,   0,   0,   0,   1,   2,   5,   5, 254, 251,
       178, 167, 223, 255, 158,   0,   6,   2,   0,   0,   0,   0,   3,
         0,   0,   2,   0,   0, 247, 255, 246, 255, 255, 255, 157,   1,
         0,   0,   0,   0,   0,   0,   0,   1,   0,   1,   4,   2, 255,
       194,   6, 159, 158, 249, 159,   2,   0,   0,   0,   0,   0,   0,
         2,   3,   2,   0,   3,   0, 252,  94,   4,   0,   1, 255, 160,
         0,   2,   3,   0,   0,   0,   0,   4,   0,   4,   3,   0,   6,
       255, 104,   0,   0,   0, 255, 158,   0,   5,   0,   0,   0,   0,
         0,   0,   4,   0,   3,   4,   0, 255, 255,   5,   3,  72, 253,
       158,   0,   1,   0,   0,   0,   0,   0,   1,   4,   0,   0,   1,
         0, 255, 255, 197, 199, 236, 255, 155,   0,   0,   0,   0,   0,
         0,   0,   0,   0,   3,   0,   0,   1, 254, 250, 255, 255, 253,
       249, 163,   0,   3,   0,   0,   0,   0,   0,   1,   0,   4,   0,
         2,   0, 255, 182, 130, 137, 134, 255, 159,   1,   0,   2,   0,
         0,   0,   0,   7,   0,   0,   0,   6,   0,  46,  13,   1,   7,
         0,  43,  30,   2,   0,   5,   0,   0,   0,   0], dtype=uint8)
```

同样读取测试数据。

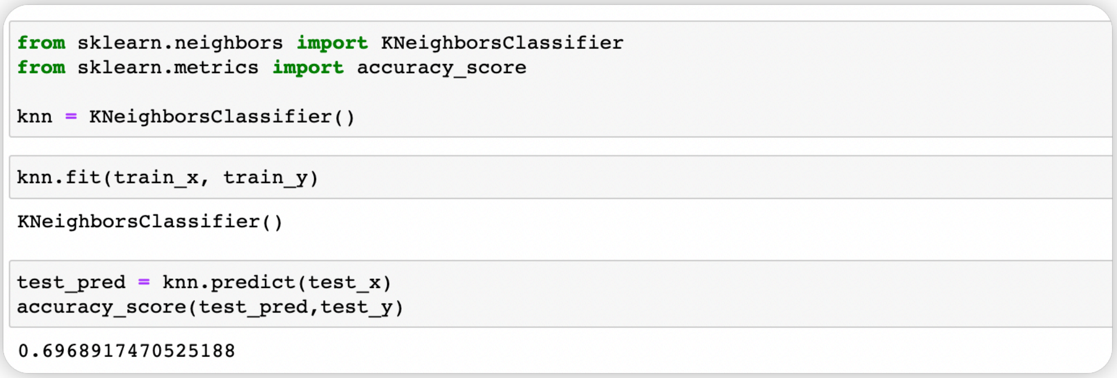


```
print(train_x.shape)
print(train_y.shape)
```

```
(15954, 400)
(15954,)
```

```
test_x, test_y = load_dataset('./data/test')
```

2. 构建KNN模型



```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

knn = KNeighborsClassifier()
```

```
knn.fit(train_x, train_y)
```

```
KNeighborsClassifier()
```

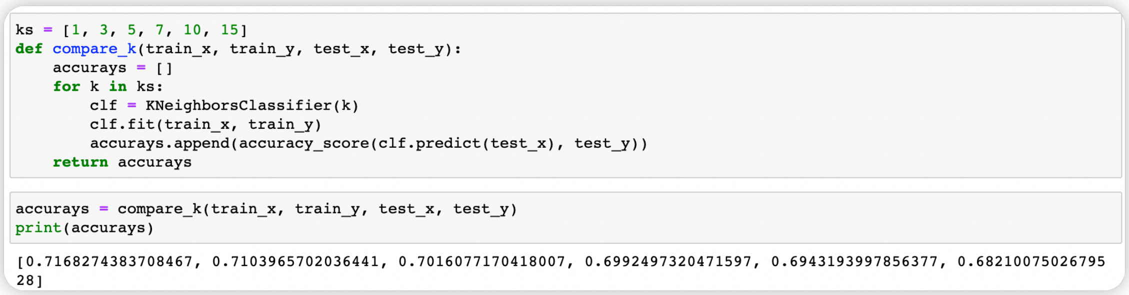
```
test_pred = knn.predict(test_x)
accuracy_score(test_pred,test_y)
```

```
0.6968917470525188
```

K默认值为5的情况下，预测的正确率为69.68%

3. 分析不同k取值准确率

K分别取1，3，5，7，10，15

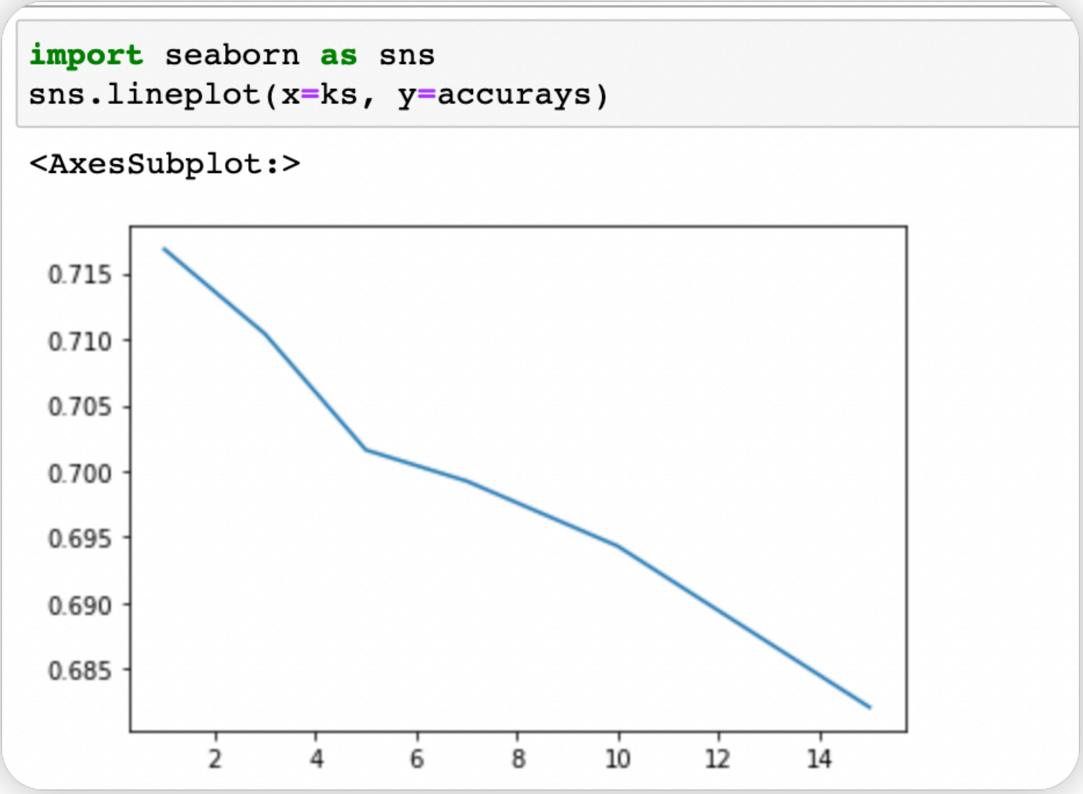


```
ks = [1, 3, 5, 7, 10, 15]
def compare_k(train_x, train_y, test_x, test_y):
    accurays = []
    for k in ks:
        clf = KNeighborsClassifier(k)
        clf.fit(train_x, train_y)
        accurays.append(accuracy_score(clf.predict(test_x), test_y))
    return accurays
```

```
accurays = compare_k(train_x, train_y, test_x, test_y)
print(accurays)
```

```
[0.7168274383708467, 0.7103965702036441, 0.7016077170418007, 0.6992497320471597, 0.6943193997856377, 0.68210075026795
28]
```

从预测结果上看，在该实验中k值越大，准确率越低，最高的正确率为71.68%。

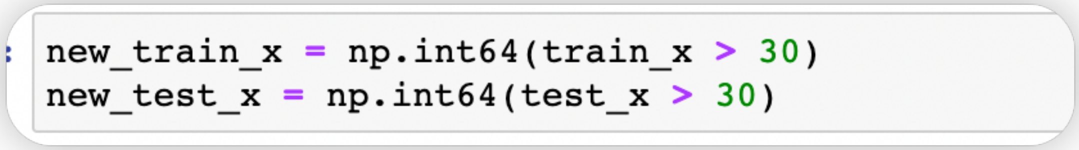


```
import seaborn as sns
sns.lineplot(x=ks, y=accurays)
```

```
<AxesSubplot:>
```

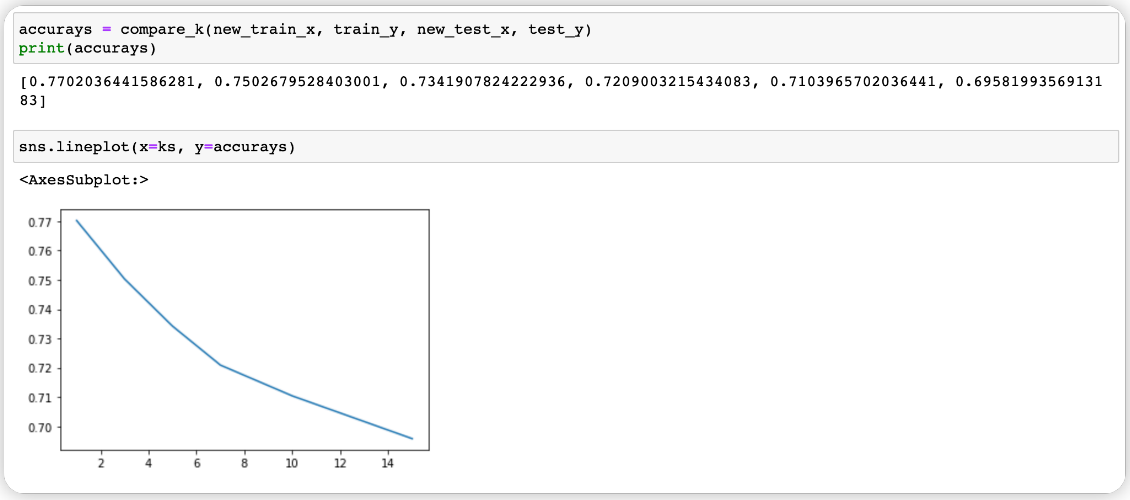
4. 数据二值化

KNN算法计算的距离与值的范围有关，需要归一化处理，我的做法是将大于30的取值为1，小于30的取值为0



```
new_train_x = np.int64(train_x > 30)
new_test_x = np.int64(test_x > 30)
```

用二值化的数据来预测，结果比原来的有较大提升，当k为1时，正确率为77.02%。



```
accurays = compare_k(new_train_x, train_y, new_test_x, test_y)
print(accurays)
```

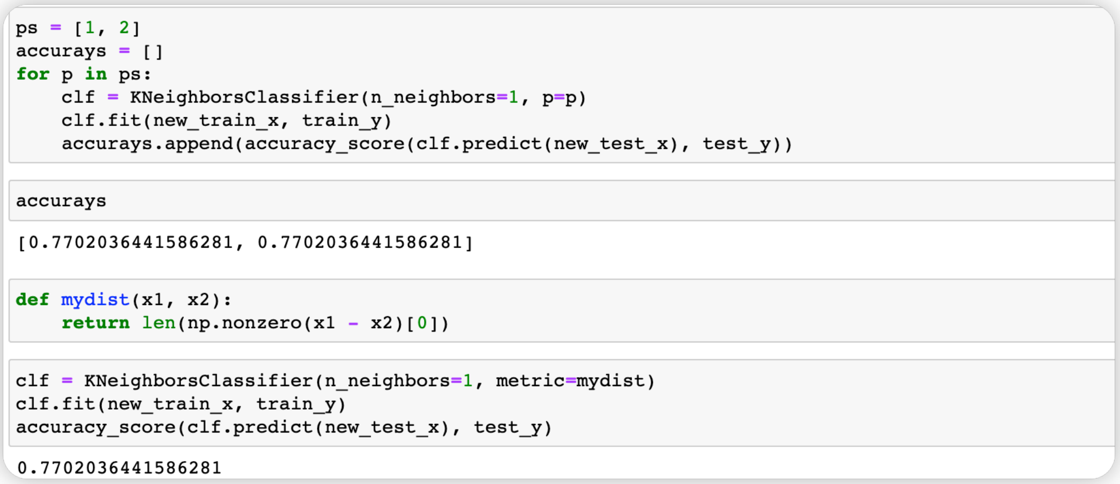
```
[0.7702036441586281, 0.7502679528403001, 0.7341907824222936, 0.7209003215434083, 0.7103965702036441, 0.69581993569131
83]
```

```
sns.lineplot(x=ks, y=accurays)
```

```
<AxesSubplot:>
```

5. 比较不同距离度量方式

对比了曼哈顿距离，欧式距离和汉明距离，二值化后这三个距离的计算应该是一样的。



```
ps = [1, 2]
accurays = []
for p in ps:
    clf = KNeighborsClassifier(n_neighbors=1, p=p)
    clf.fit(new_train_x, train_y)
    accurays.append(accuracy_score(clf.predict(new_test_x), test_y))
```

```
accurays
```

```
[0.7702036441586281, 0.7702036441586281]
```

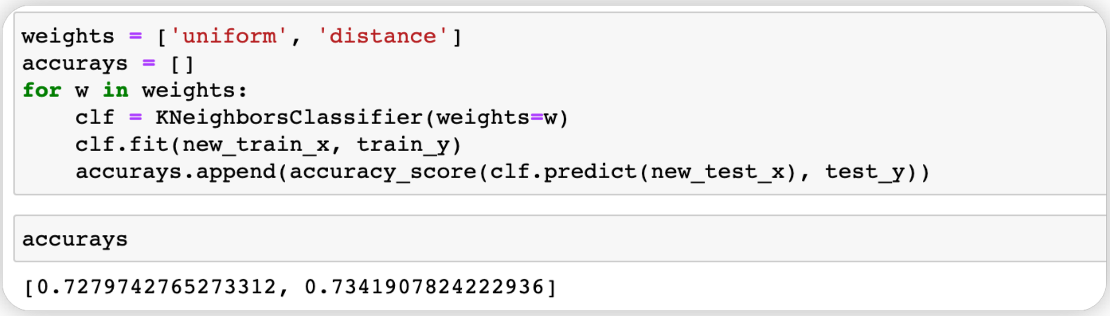
```
def mydist(x1, x2):
    return len(np.nonzero(x1 - x2)[0])
```

```
clf = KNeighborsClassifier(n_neighbors=1, metric=mydist)
clf.fit(new_train_x, train_y)
accuracy_score(clf.predict(new_test_x), test_y)
```

```
0.7702036441586281
```

6. 比较加权和平均影响

在默认k=5的情况下，用距离加权的方式预测的结果会比平均的要高。



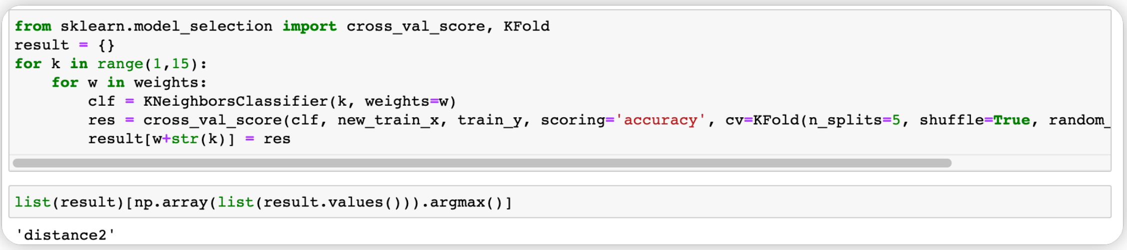
```
weights = ['uniform', 'distance']
accurays = []
for w in weights:
    clf = KNeighborsClassifier(weights=w)
    clf.fit(new_train_x, train_y)
    accurays.append(accuracy_score(clf.predict(new_test_x), test_y))
```

```
accurays
```

```
[0.7279742765273312, 0.7341907824222936]
```

7. 寻找最佳参数

用交叉验证的方式，遍历k从1到15，权重分别是距离加权和平均两种方式的预测结果。

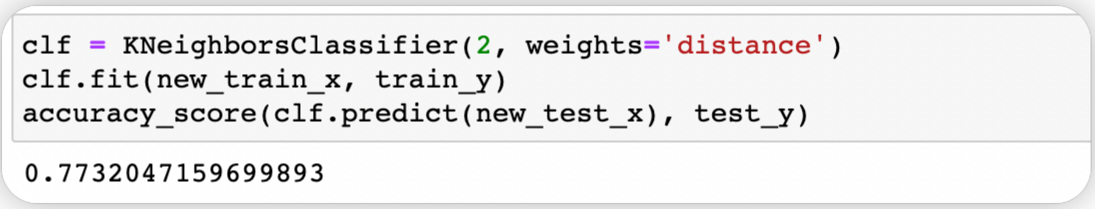


```
from sklearn.model_selection import cross_val_score, KFold
result = {}
for k in range(1,15):
    for w in weights:
        clf = KNeighborsClassifier(k, weights=w)
        res = cross_val_score(clf, new_train_x, train_y, scoring='accuracy', cv=KFold(n_splits=5, shuffle=True, random_
        result[w+str(k)] = res
```

```
list(result)[np.array(list(result.values())).argmax()]
```

```
'distance2'
```

结果表明，k为2，且使用距离加权的正确率最高。



```
clf = KNeighborsClassifier(2, weights='distance')
clf.fit(new_train_x, train_y)
accuracy_score(clf.predict(new_test_x), test_y)
```

```
0.7732047159699893
```

相较于k=1，提升了0.3%

8. 实验总结

本次实验用用 K-NN 的方法进行图像识别，比较了不同k值，准确率的变化，也比较了平均和加权对KNN的影响，最后找出模型的最佳参数。