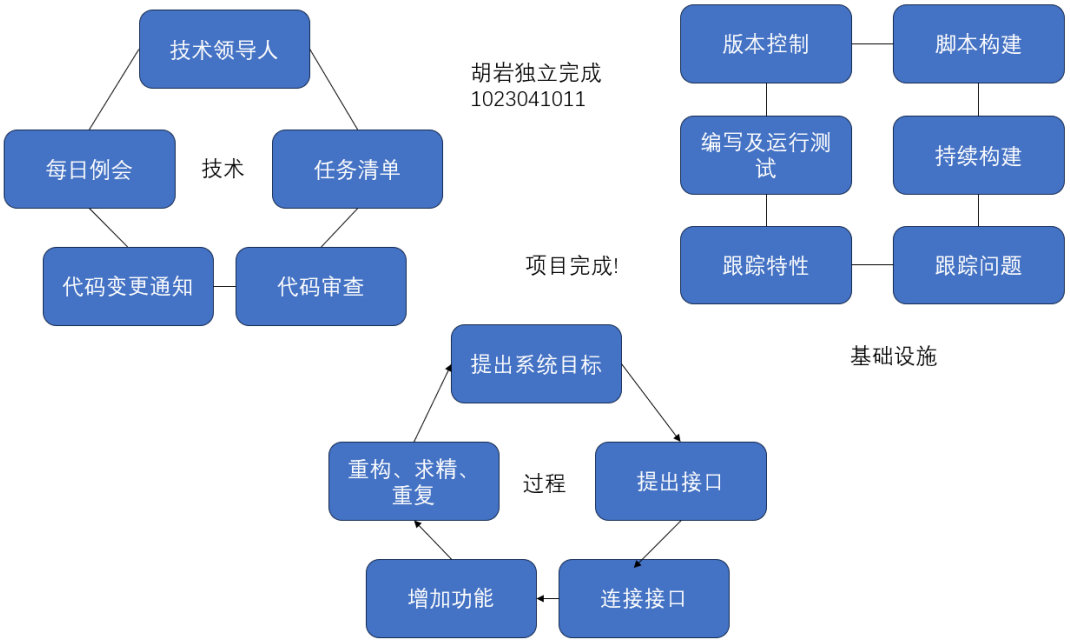
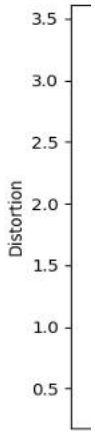


1.3 数据集

实验用数据集总共包含 3000 个样本

	A	B
1	V1	V2
2	2.072345	-3.24169
3	17.93671	15.78481
4	1.083576	7.319176
5	11.12067	14.40678
6	23.71155	2.557729
7	24.16993	32.02478
8	21.66578	4.892855
9	4.693684	12.34217
10	19.21191	-1.12137
11	4.230391	-4.44154
12	9.12713	23.60572
13	0.407503	15.29705
14	7.314846	3.309312
15	-3.4384	-12.0253
16	17.63935	-3.21235
17	4.415292	22.81555
18	11.94122	8.122487
19	0.725853	1.806819
20	8.185273	28.1326
21	-5.77359	1.0248
22	18.76943	24.16946

数据集



当时根据这个数据集画了个 kmeans 的聚类图

1.5 计算步骤

K-Means 聚类算法步骤实质是 EM 算法（最大期望算法（Expectation-Maximization algorithm,EM）)的模型优化过程，具体步骤如下：

（1） 随机选择 k 个样本作为初始簇类的均值向量；

K 值的选取对 K-means 影响很大，这也是 K-means 最大的缺点

```
# Number of clusters
k = 3
# X coordinates of random centroids
C_x = np.random.randint(0, np.max(X), size=k)
# Y coordinates of random centroids
C_y = np.random.randint(0, np.max(X), size=k)
C = np.array(list(zip(C_x, C_y)), dtype=np.float32)
```

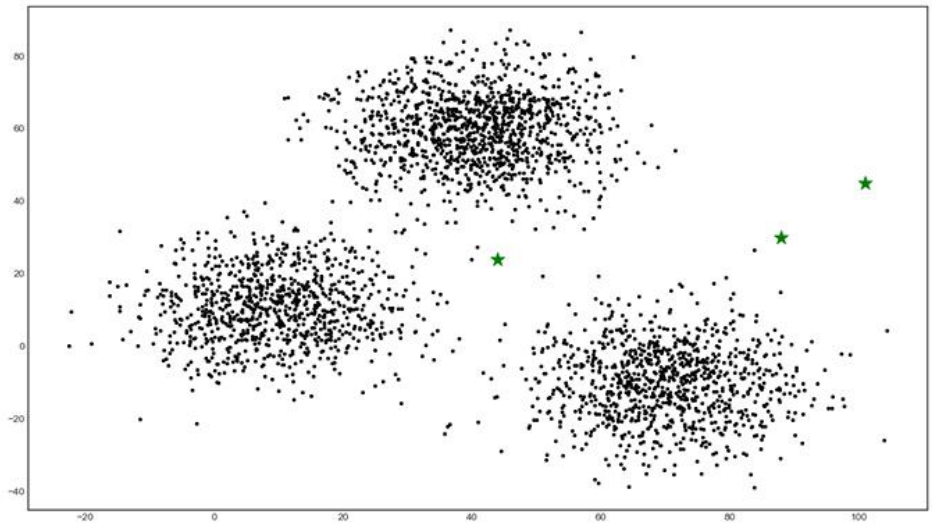
（2） 将每个样本数据集划分离它距离最近的簇；

（3） 根据每个样本所属的簇，更新簇类的均值向量；

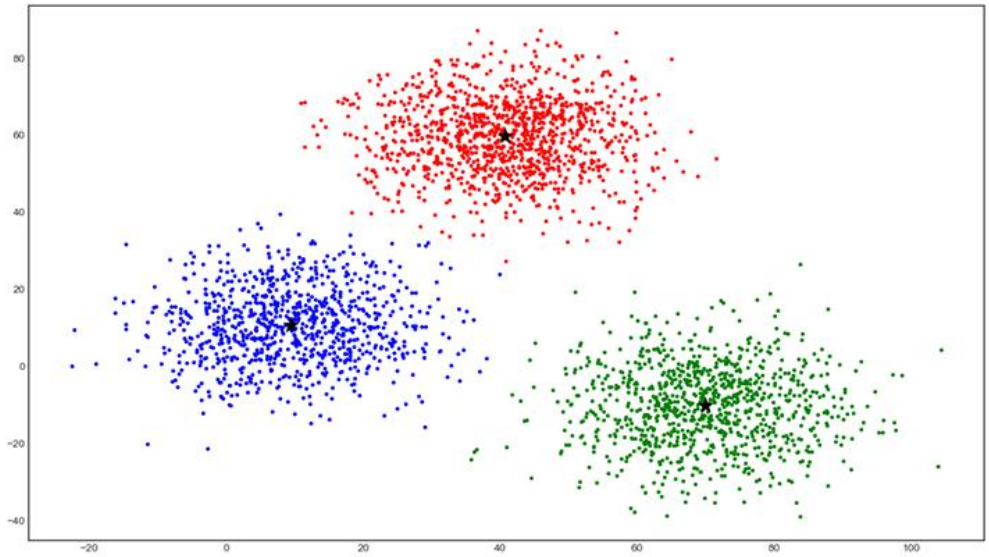
（4） 重复（2）（3）步，当达到设置的迭代次数或簇类的均值向量不再改变时，模型构建完成，输出聚类算法结果。

```
while error != 0:
    # Assigning each value to its closest cluster
    for i in range(len(X)):
        distances = dist(X[i], C)  # <class 'numpy.ndarray'>
        cluster = np.argmin(distances)
        clusters[i] = cluster
    # Storing the old centroid values
    C_old = deepcopy(C)
    # Finding the new centroids by taking the average value
    for i in range(k):
        points = [X[j] for j in range(len(X)) if clusters[j] == i]
        C[i] = np.mean(points, axis=0)
    error = dist(C, C_old, None)
```

K-Means 最核心的部分就是先固定中心点，调整每个样本所属的类别来减少损失



经过多次迭代，将各个样本划分到正确的簇中，并不断更新簇中心的位置。



如果选取不合适的 K 值，将会出现以下情况，逼迫算法将样本划分为 6 类。