In [1]:
```python
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sb

%matplotlib inline
```

In [2]:
```python
import warnings
warnings.simplefilter("ignore")
```

In [4]:
```python
df = pd.read_csv('banknote.csv')
```

In [5]:
```python
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

In [6]:
```python
df.head()
```

Out[6]:

	V1	V2	V3	V4	Class
0	3.62160	8.6661	-2.8073	-0.44699	1
1	4.54590	8.1674	-2.4586	-1.46210	1
2	3.86600	-2.6383	1.9242	0.10645	1
3	3.45660	9.5228	-4.0112	-3.59440	1
4	0.32924	-4.4552	4.5718	-0.98880	1

In [7]:
```python
df.shape
```

Out[7]: (1372, 5)

In [9]:
```python
df.describe()
```

Out[9]:

	V1	V2	V3	V4	Class
count	1372.000000	1372.000000	1372.000000	1372.000000	1372.000000
mean	0.433735	1.922353	1.397627	-1.191657	1.444606
std	2.842763	5.869047	4.310030	2.101013	0.497103
min	-7.042100	-13.773100	-5.286100	-8.548200	1.000000
25%	-1.773000	-1.708200	-1.574975	-2.413450	1.000000
50%	0.496180	2.319650	0.616630	-0.586650	1.000000
75%	2.821475	6.814625	3.179250	0.394810	2.000000
max	6.824800	12.951600	17.927400	2.449500	2.000000

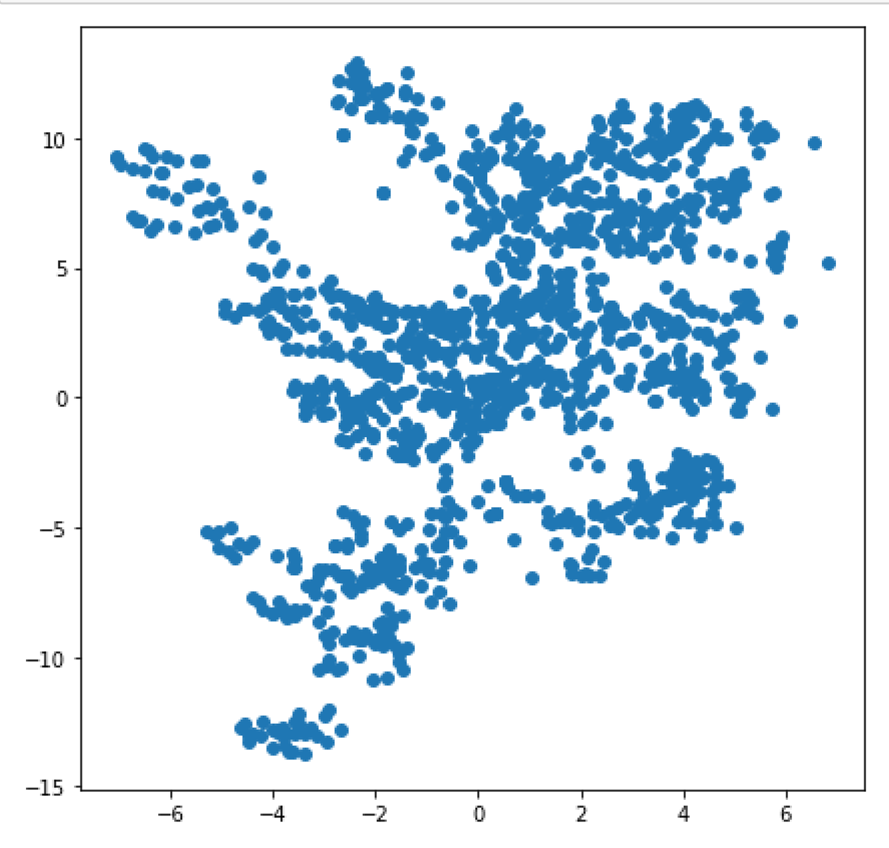
In [10]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1372 entries, 0 to 1371
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   V1      1372 non-null   float64
 1   V2      1372 non-null   float64
 2   V3      1372 non-null   float64
 3   V4      1372 non-null   float64
 4   Class   1372 non-null   int64
dtypes: float64(4), int64(1)
memory usage: 53.7 KB
```

In [11]:
```python
df[df.duplicated()].shape[0]
```

Out[11]: 24

In [12]:
```python
plt.figure(figsize = [7, 7])
plt.scatter(df.V1, df.V2);
```

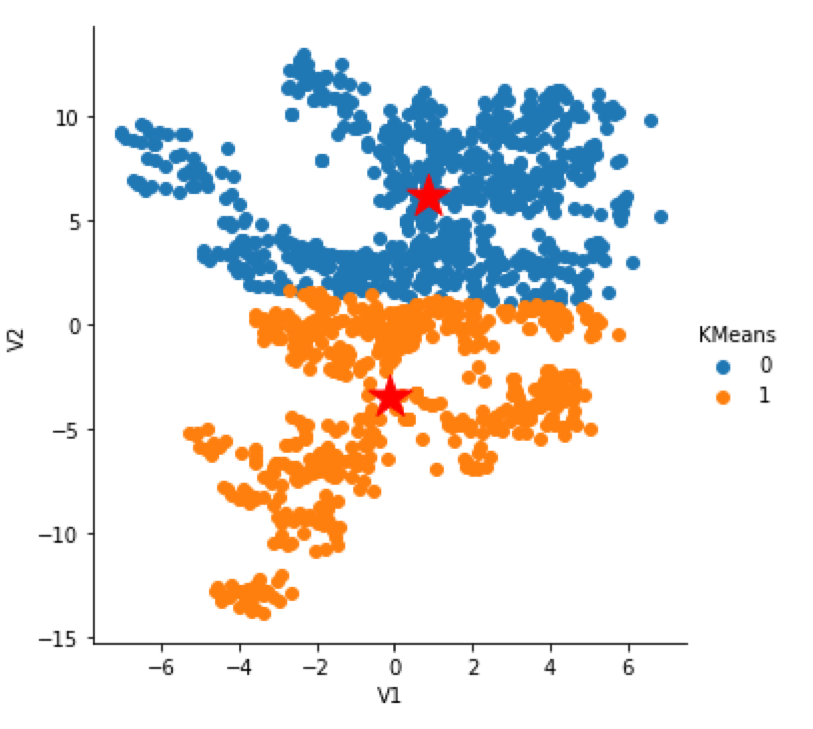


The data distribution in the graph is not too widespread, neither too centered at one place, therefore it is worth trying to computing K-Means on this dataset. There is no obvious cluster in spherical shapes so we should expect the K-Means model won't work perfectly here.

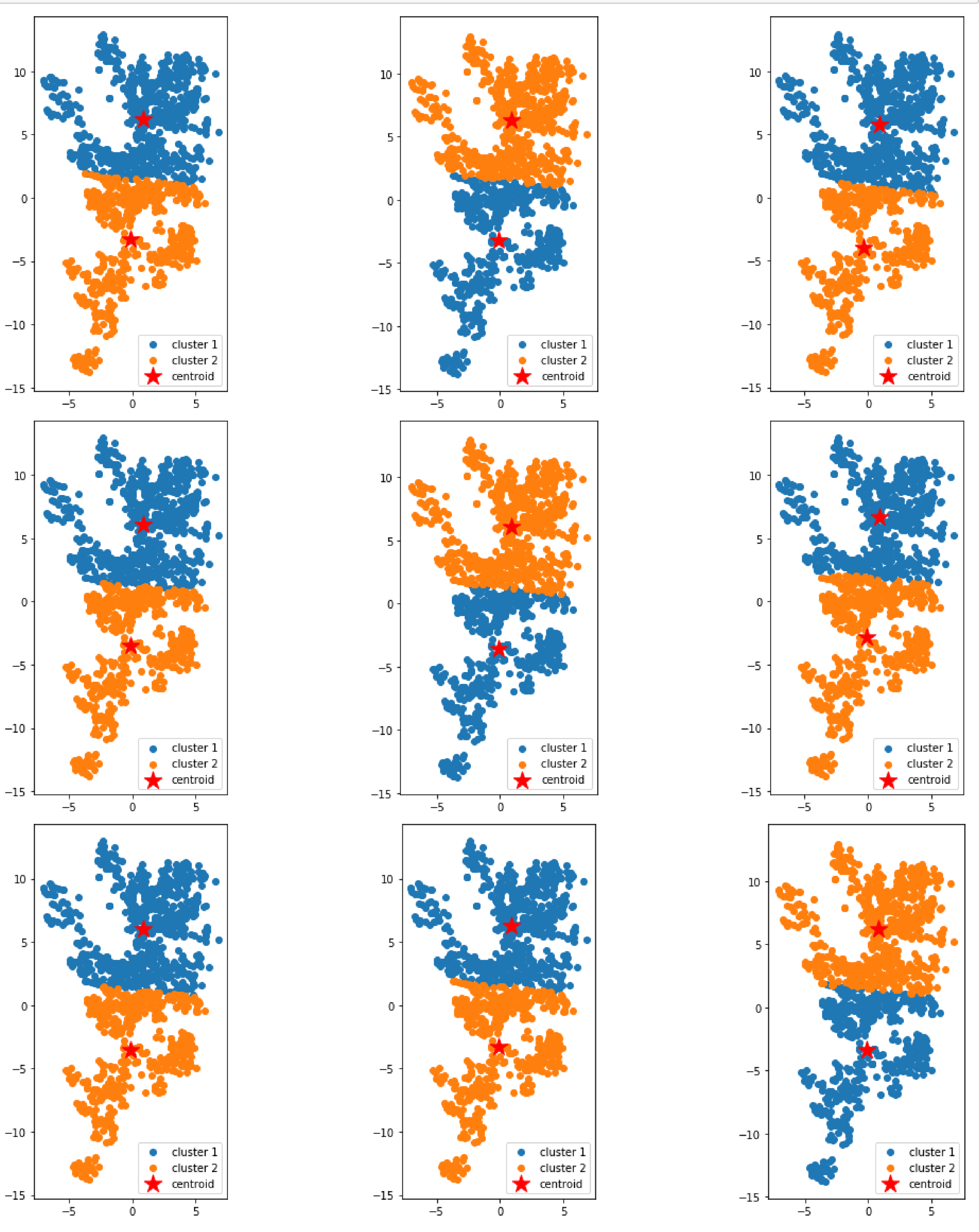
K-means clusttering

In [15]:
```python
from sklearn.cluster import KMeans
data = np.column_stack(( df.V1, df.V2))  # we use only V1 and V2

# compute K-Means
km_res = KMeans(n_clusters = 2).fit(data)
clusters = km_res.cluster_centers_

# put the assigned labels to the original dataset
df['KMeans'] = km_res.labels_

#plot out the result
g = sb.FacetGrid(data = df, hue = 'KMeans', size = 5)
g.map(plt.scatter, 'V1', 'V2')
g.add_legend();
plt.scatter(clusters[:,0], clusters[:,1], s=500, marker='*', c='r')
```

Out[15]: <matplotlib.collections.PathCollection at 0x13009928088>



In [16]:
```python
from sklearn.datasets.samples_generator import (make_blobs,
                                                make_circles,
                                                make_moons)
from sklearn.cluster import KMeans, SpectralClustering

n_iter = 9
fig, ax = plt.subplots(3, 3, figsize=(16, 16))
ax = np.ravel(ax)
centers = []
for i in range(n_iter):
    # Run local implementation of kmeans
    km = KMeans(n_clusters=2,
                max_iter=3)
    km.fit(data)
    centroids = km.cluster_centers_
    centers.append(centroids)
    ax[i].scatter(data[km.labels_ == 0, 0], data[km.labels_ == 0, 1],
                  label='cluster 1')
    ax[i].scatter(data[km.labels_ == 1, 0], data[km.labels_ == 1, 1],
                  label='cluster 2')
    ax[i].scatter(centroids[:, 0], centroids[:, 1],
                 c='r', marker='*', s=300, label='centroid')
    ax[i].legend(loc='lower right')
    ax[i].set_aspect('equal')
plt.tight_layout();
```



After running K-Means for 9 times, the results we got are very similar, which means the K-Means here is stable.

In [17]:
```python
km_res.cluster_centers_
```

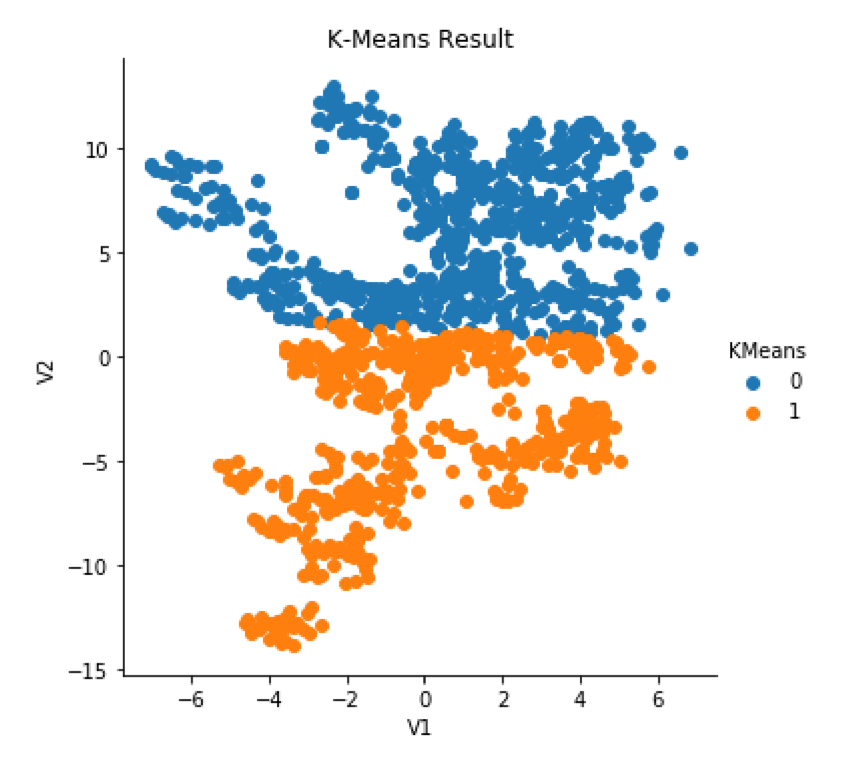
Out[17]:
```
array([[ 0.86960048,  6.12717909],
       [-0.12376677, -3.45591265]])
```

In [18]:
```python
df['KMeans'] = km_res.labels_
df.groupby('KMeans').describe()
```

Out[18]:

	V1								V2				
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%
KMeans													
0	773.0	0.867409	2.908508	-7.0421	-0.95403	1.11660	3.1896	6.8248	773.0	6.108644	3.096698	1.0367	3.2570
1	599.0	-0.125914	2.655152	-5.2943	-2.18825	-0.65767	2.0159	5.7403	599.0	-3.479989	3.834604	-13.7731	-6.3061

In [20]:
```python
df_labels=pd.read_csv('banknote.csv')
g = sb.FacetGrid(data = df_labels, hue = 'Class', size = 5)
g.map(plt.scatter, 'V1', 'V2')
g.add_legend()
plt.title("Data With Correct Lables")


# plot the data computed by K-Means
g = sb.FacetGrid(data = df, hue = 'KMeans', size = 5)
g.map(plt.scatter, 'V1', 'V2')
g.add_legend()
plt.title("K-Means Result");
```



In [21]:
```python
df["KMeans"] = df["KMeans"].map({0: 1, 1: 2})
```

In [22]:
```python
df.head()
```

Out[22]:

	V1	V2	V3	V4	Class	KMeans
0	3.62160	8.6661	-2.8073	-0.44699	1	1
1	4.54590	8.1674	-2.4586	-1.46210	1	1
2	3.86600	-2.6383	1.9242	0.10645	1	2
3	3.45660	9.5228	-4.0112	-3.59440	1	1
4	0.32924	-4.4552	4.5718	-0.98880	1	2

In [23]:
```python
correct = 0

for i in range(0,1372):
    if df.Class[i] == df["KMeans"][i]:
        correct+=1
print(correct/1371)
```

0.6528081692195478

Accuracy = 65.3 %

In [ ]: