In [14]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as stats
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
from sklearn.cluster import KMeans
df = pd.read_csv("C:/Users/User/Documents/ingres.csv")
df
```

Out[14]:

	a	b	c	d	e	f	g	h	i
0	1.51735	13.02	3.54	1.69	72.73	0.54	8.44	0.00	0.07
1	1.53125	10.73	0.00	2.10	69.81	0.58	13.30	3.15	0.28
2	1.52300	13.31	3.58	0.82	71.99	0.12	10.17	0.00	0.03
3	1.51768	12.56	3.52	1.43	73.15	0.57	8.54	0.00	0.00
4	1.51813	13.43	3.98	1.18	72.49	0.58	8.15	0.00	0.00
...	...	...	...	...	...	...	...	...	...
209	1.52152	13.12	3.58	0.90	72.20	0.23	9.82	0.00	0.16
210	1.51848	13.64	3.87	1.27	71.96	0.54	8.32	0.00	0.32
211	1.51784	12.68	3.67	1.16	73.11	0.61	8.70	0.00	0.00
212	1.51841	12.93	3.74	1.11	72.28	0.64	8.96	0.00	0.22
213	1.51321	13.00	0.00	3.02	70.70	6.21	6.93	0.00	0.00

214 rows × 9 columns

In [15]:

```
df.describe()
```

Out[15]:

	a	b	c	d	e	f	g	
count	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.00
mean	1.518365	13.407850	2.684533	1.444907	72.650935	0.497056	8.956963	0.17
std	0.003037	0.816604	1.442408	0.499270	0.774546	0.652192	1.423153	0.49
min	1.511150	10.730000	0.000000	0.290000	69.810000	0.000000	5.430000	0.00
25%	1.516522	12.907500	2.115000	1.190000	72.280000	0.122500	8.240000	0.00
50%	1.517680	13.300000	3.480000	1.360000	72.790000	0.555000	8.600000	0.00
75%	1.519157	13.825000	3.600000	1.630000	73.087500	0.610000	9.172500	0.00
max	1.533930	17.380000	4.490000	3.500000	75.410000	6.210000	16.190000	3.15

In [16]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 214 entries, 0 to 213
Data columns (total 9 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   a       214 non-null    float64
 1   b       214 non-null    float64
 2   c       214 non-null    float64
 3   d       214 non-null    float64
 4   e       214 non-null    float64
 5   f       214 non-null    float64
 6   g       214 non-null    float64
 7   h       214 non-null    float64
 8   i       214 non-null    float64
dtypes: float64(9)
memory usage: 15.2 KB
```

In [17]:
```
from sklearn.datasets import make_blobs

dataset, classes = make_blobs(n_samples=214, n_features=9, centers=4, cluster_st
# make as panda dataframe for easy understanding
df = pd.DataFrame(dataset, columns=['var1', 'var2', 'var3', 'var4', 'var5', 'var
df.head(5)
```

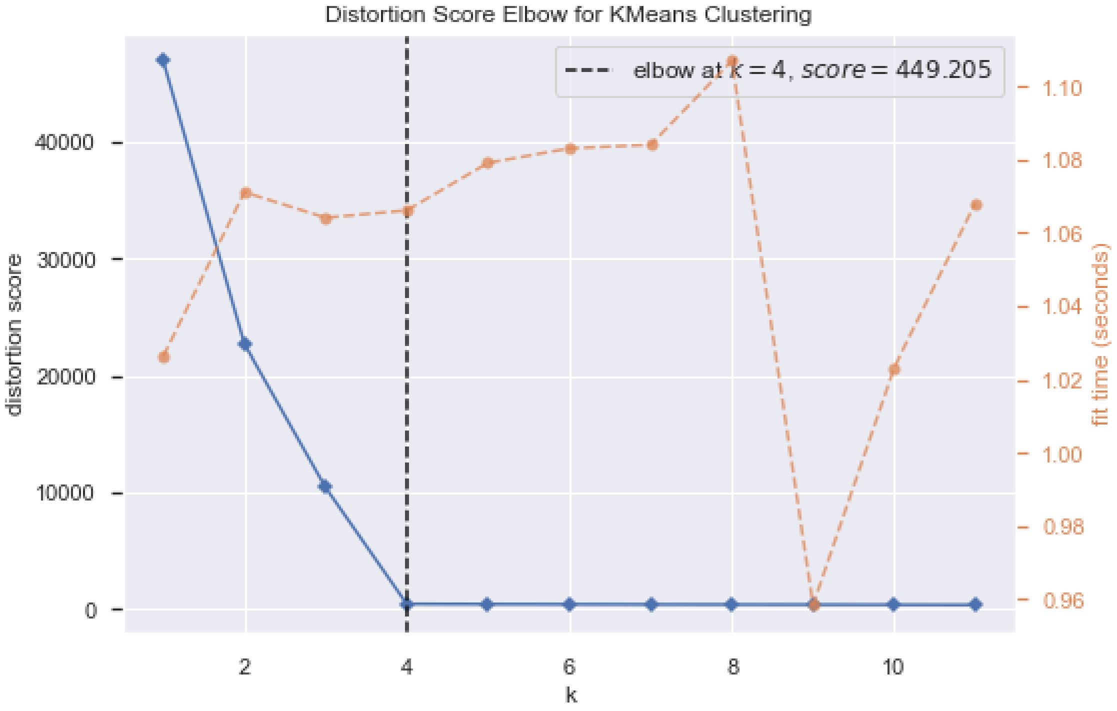
Out[17]:

	var1	var2	var3	var4	var5	var6	var7	var8	var9
0	9.131912	0.258991	-0.436645	-4.245609	5.763715	-1.435468	1.351031	-9.503602	2.916602
1	4.839846	6.608900	10.052645	6.096092	-1.045162	5.061298	-6.474112	2.856966	-6.865834
2	9.282017	0.291446	-1.429697	-5.043951	5.454493	-1.232514	1.728513	-9.748414	1.987273
3	-2.516522	5.361693	0.111528	0.729357	8.738177	-8.530331	-8.481497	-9.920301	6.640685
4	1.165846	5.433442	2.034139	0.419691	-1.699895	2.686084	-1.007515	7.065062	9.304886

In [18]:
```
#to find the optimal no of clusters in dataset
from yellowbrick.cluster import KElbowVisualizer
model = KMeans()
visualizer = KElbowVisualizer(model, k=(1,12)).fit(df)
visualizer.show()
```

```
C:\Users\User\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1332: User
Warning: KMeans is known to have a memory leak on Windows with MKL, when there
are less chunks than available threads. You can avoid it by setting the environ
ment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\User\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1332: User
Warning: KMeans is known to have a memory leak on Windows with MKL, when there
are less chunks than available threads. You can avoid it by setting the environ
ment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\User\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1332: User
Warning: KMeans is known to have a memory leak on Windows with MKL, when there
are less chunks than available threads. You can avoid it by setting the environ
ment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\User\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1332: User
Warning: KMeans is known to have a memory leak on Windows with MKL, when there
are less chunks than available threads. You can avoid it by setting the environ
ment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\User\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1332: User
Warning: KMeans is known to have a memory leak on Windows with MKL, when there
are less chunks than available threads. You can avoid it by setting the environ
ment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\User\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1332: User
Warning: KMeans is known to have a memory leak on Windows with MKL, when there
are less chunks than available threads. You can avoid it by setting the environ
ment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\User\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1332: User
Warning: KMeans is known to have a memory leak on Windows with MKL, when there
are less chunks than available threads. You can avoid it by setting the environ
ment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\User\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1332: User
Warning: KMeans is known to have a memory leak on Windows with MKL, when there
are less chunks than available threads. You can avoid it by setting the environ
ment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\User\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1332: User
Warning: KMeans is known to have a memory leak on Windows with MKL, when there
are less chunks than available threads. You can avoid it by setting the environ
ment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\User\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1332: User
Warning: KMeans is known to have a memory leak on Windows with MKL, when there
are less chunks than available threads. You can avoid it by setting the environ
ment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\User\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1332: User
```

```
Warning: KMeans is known to have a memory leak on Windows with MKL, when there
are less chunks than available threads. You can avoid it by setting the environ
ment variable OMP_NUM_THREADS=1.
  warnings.warn(
```



```
Out[18]: <AxesSubplot:title={'center':'Distortion Score Elbow for KMeans Clustering'}, x
label='k', ylabel='distortion score'>
```

```
In [19]: from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=4, init='k-means++', random_state=0).fit(df)
```

```
C:\Users\User\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1332: User
Warning: KMeans is known to have a memory leak on Windows with MKL, when there
are less chunks than available threads. You can avoid it by setting the environ
ment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

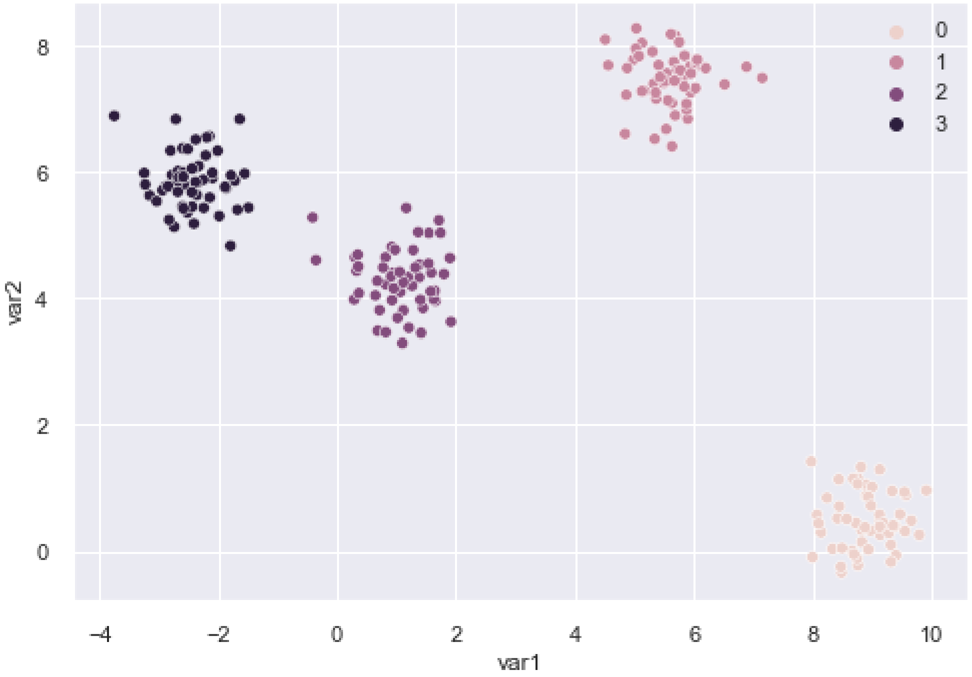
In [20]:
```
kmeans.labels_  # same as kmeans.predict(df)
from array import array
arr = (np.array([2, 0, 3, 1, 2, 3, 0, 3, 3, 3, 3, 2, 0, 0, 2, 3, 1, 1, 1, 2, 1, 0
       2, 0, 2, 2, 1, 2, 2, 3, 1, 3, 0, 2, 0, 3, 0, 3, 3, 1, 1, 1, 1, 3,
       2, 0, 3, 1, 1, 3, 1, 0, 0, 1, 3, 1, 0, 2, 3, 2, 1, 3, 2, 3, 1, 3,
       2, 1, 0, 0, 2, 2, 3, 3, 0, 1, 0, 0, 2, 2, 1, 3, 2, 0, 0, 3, 3, 2,
       0, 0, 1, 1, 1, 3, 3, 2, 0, 1, 3, 3, 1, 2, 2, 1, 1, 0, 3, 2, 2, 3,
       1, 0, 0, 2, 2, 3, 0, 0, 1, 3, 1, 0, 3, 2, 3, 0, 3, 0, 2, 3, 0, 2,
       0, 1, 1, 0, 1, 1, 2, 1, 2, 0, 2, 2, 0, 2, 3, 2, 0, 1, 1, 1, 3, 0,
       2, 3, 1, 0, 1, 2, 1, 2, 2, 0, 0, 1, 3, 2, 2, 0, 2, 3, 0, 1, 1, 1,
       3, 3, 0, 3, 0, 2, 3, 2, 3, 0, 0, 1, 3, 1, 2, 2, 3, 1, 0, 0, 0, 3,
       1, 2]))
np.array([(str('u'), [1,2,3])], dtype=[('x', 'a1'), ('y', list)])
kmeans.inertia_
kmeans.n_iter_
kmeans.cluster_centers_
```

Out[20]:
```
array([[ 8.89761649,  0.49427344, -1.5390784 , -4.74528203,  5.43197908,
        -0.92746453,  1.26159202, -9.59065898,  2.32184716],
       [ 5.60042075,  7.48954086,  9.68799643,  6.00968239, -0.7100781 ,
         5.45778514, -7.59678344,  2.73302385, -7.15310901],
       [ 1.06974437,  4.29443339,  2.02946323,  0.83467167, -1.64428184,
         2.91882068, -1.22987957,  7.68738619,  9.27603469],
       [-2.42308412,  5.86846463,  0.5863834 ,  1.21606483,  8.50757834,
        -8.55838342, -8.20826313, -9.67515823,  6.59429503]])
```

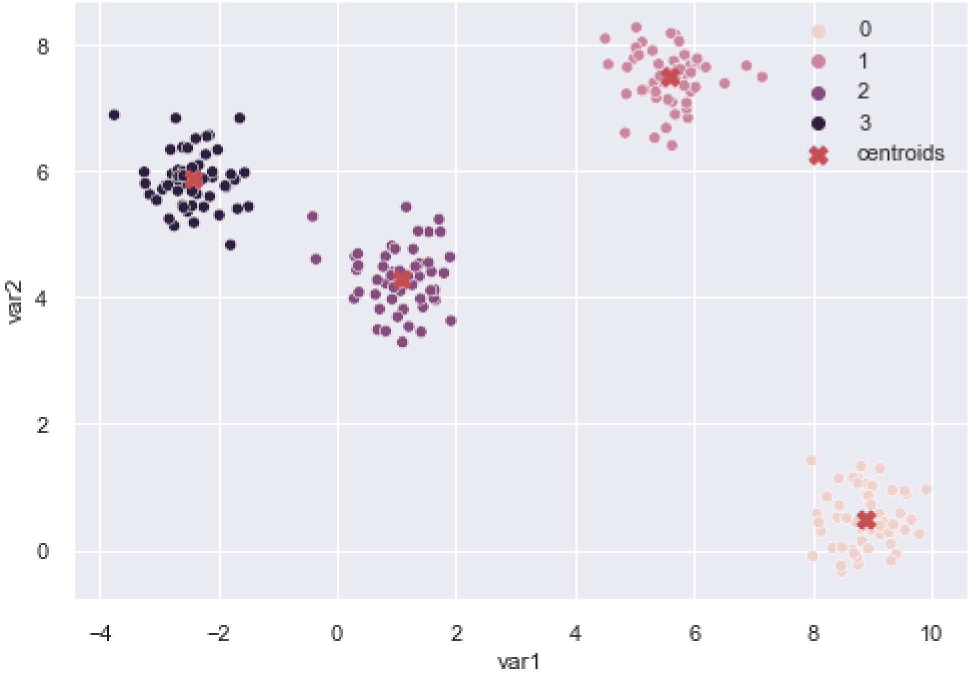
In [21]:
```
from collections import Counter
Counter(kmeans.labels_)
```

Out[21]:
```
Counter({0: 53, 1: 53, 3: 54, 2: 54})
```

In [22]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.scatterplot(data=df, x="var1", y="var2", hue=kmeans.labels_)
plt.show()
```



In [23]:
```
sns.scatterplot(data=df, x="var1", y="var2", hue=kmeans.labels_)
plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1],
            marker="X", c="r", s=80, label="centroids")
plt.legend()
plt.show()
```



In [ ]:
```
#conclusion: there are 4 distinctive no of formulations
```

In [ ]: