Actividad Evaluable: Patrones con K-means

Eduardo Rodríguez Gil - A01274913, Jose Manuel Neri Villeda - A01706450, Héctor Javier Calderón González - A01067542

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sb
import sklearn
from sklearn.cluster import KMeans
from sklearn.metrics import pairwise_distances_argmin_min
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.preprocessing import scale
import sklearn.metrics as sm
from sklearn import datasets


%matplotlib inline
from mpl_toolkits.mplot3d import Axes3D
plt.rcParams['figure.figsize'] = (16, 9)
plt.style.use('ggplot')
```

Cargamos los datos de entrada del archivo csv

In [5]:
```
dataframe = pd.read_csv(r"Bitcoin.csv") # Base de datos
dataframe.head()
```

Out[5]:

	Date	Price	Open	High	Low
0	Apr 25, 2021	49561.9	50088.2	50438.8	49226.5
1	Apr 24, 2021	50088.9	51140.8	51183.0	48775.2
2	Apr 23, 2021	51143.6	51707.1	52099.9	47659.4
3	Apr 22, 2021	51729.5	53821.3	55408.4	50590.9
4	Apr 21, 2021	53820.2	56479.5	56764.4	53657.6

Para este punto en nuestra base de datos quitamos dos variables la de Volume y la de Change, ya que no hacías un gran cambio en nuestros datos, ya que al momento de graficar no los tomábamos en cuenta al no ser unos valores numéricos.

In [6]:
```
dataframe.describe()
```

Out[6]:

	Price	Open	High	Low
count	421.000000	421.000000	421.000000	421.000000
mean	21471.073872	21372.344181	22028.754869	20687.659857
std	17492.702670	17448.718099	18024.928136	16785.882734
min	4826.000000	4815.200000	5369.300000	3869.500000

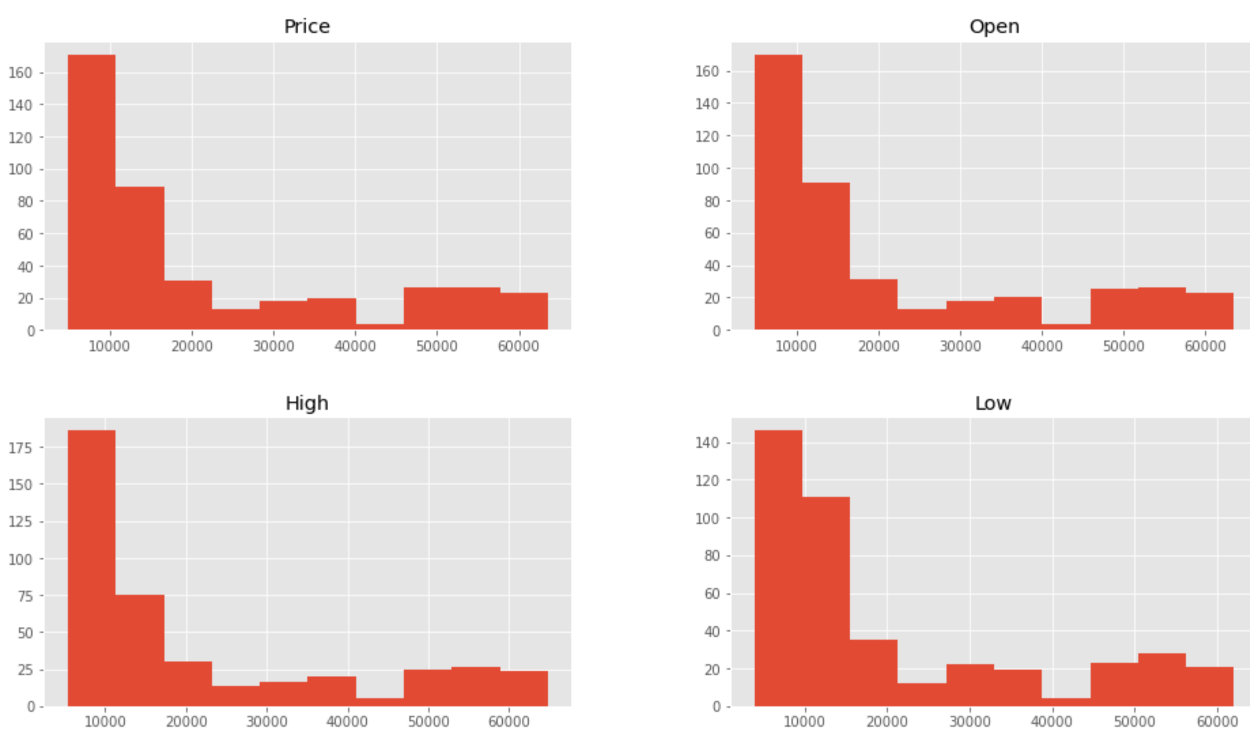
	Price	Open	High	Low
25%	9314.000000	9300.800000	9458.300000	9184.200000
50%	11557.200000	11533.500000	11766.900000	11315.900000
75%	32958.900000	32499.600000	34348.300000	30850.000000
max	63540.900000	63544.200000	64778.000000	62067.500000

In [7]:
```
# Vemos en cuanto esta el Precio de la Bitcoin
print(dataframe.groupby('Date').size())
```

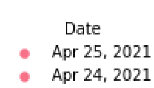
```
Price
4826.0     1
5030.0     1
5182.7     1
5261.1     1
5361.4     1
          ..
61195.3    1
61379.7    1
62980.4    1
63216.0    1
63540.9    1
Length: 421, dtype: int64
```

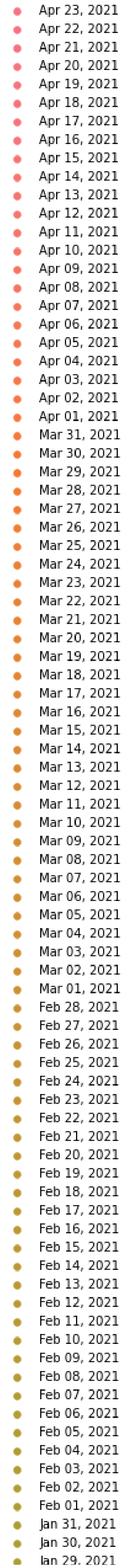
Visualizamos los datos

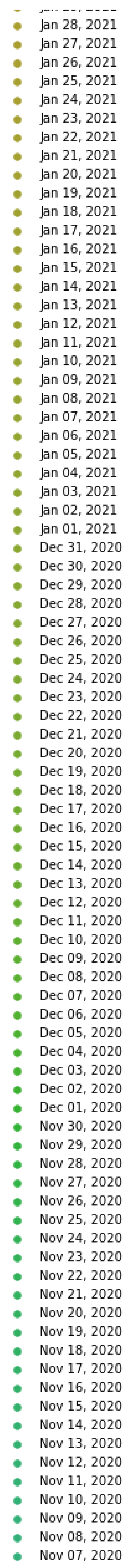
In [16]:
```
dataframe.drop(['Date'], 1).hist()
plt.show()
```

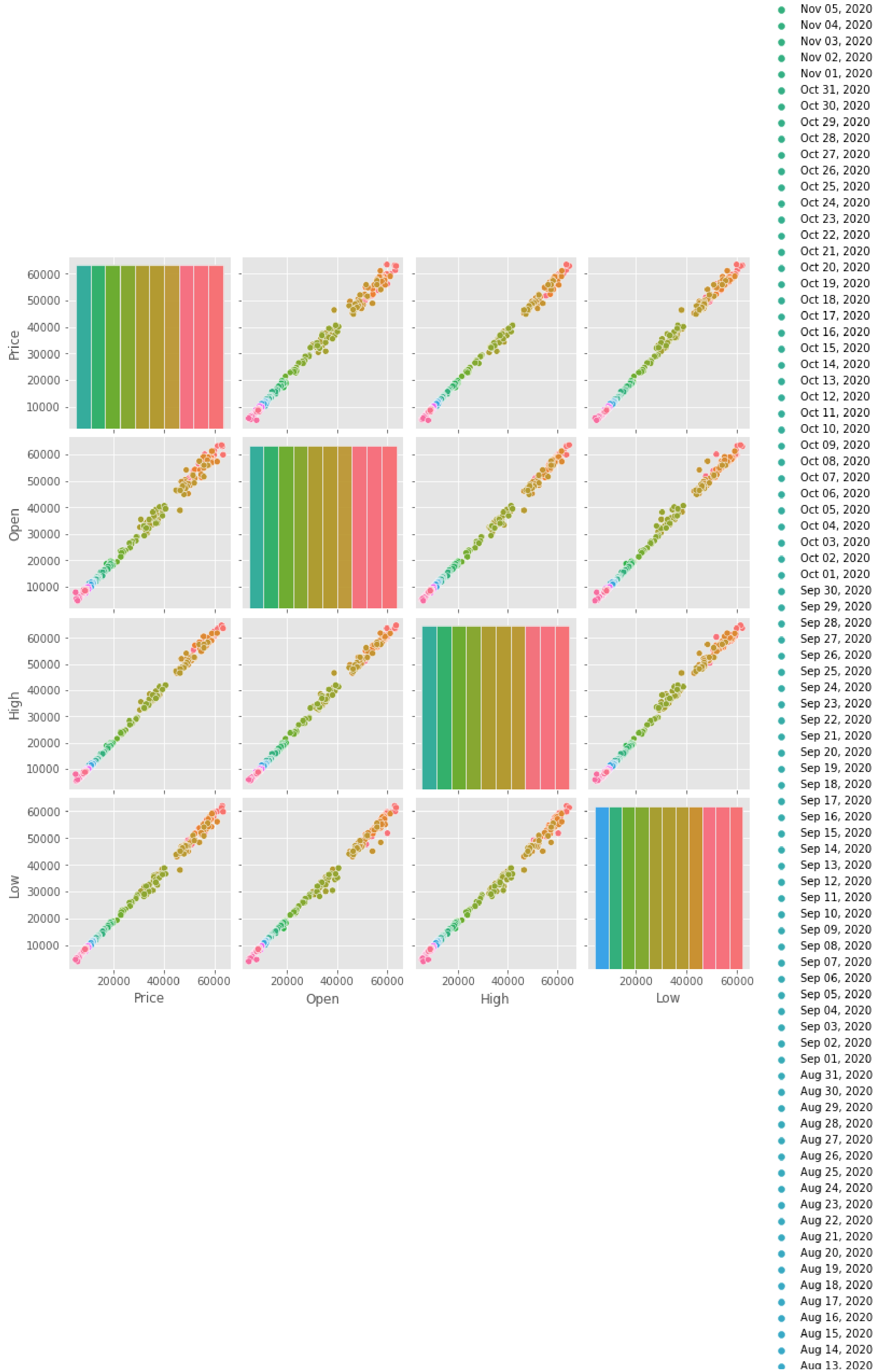


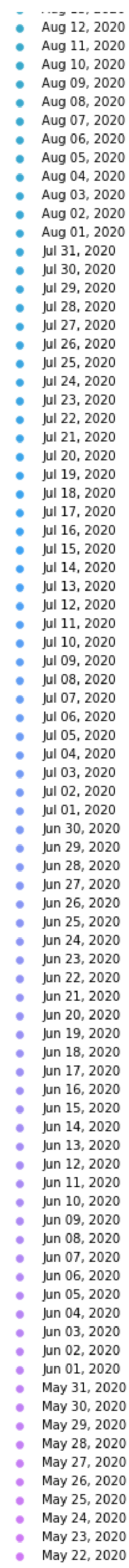
In [22]:
```
sb.pairplot(dataframe, hue = "Date", diag_kind = "hist");
```




Apr 23, 2021
Apr 22, 2021
Apr 21, 2021
Apr 20, 2021
Apr 19, 2021
Apr 18, 2021
Apr 17, 2021
Apr 16, 2021
Apr 15, 2021
Apr 14, 2021
Apr 13, 2021
Apr 12, 2021
Apr 11, 2021
Apr 10, 2021
Apr 09, 2021
Apr 08, 2021
Apr 07, 2021
Apr 06, 2021
Apr 05, 2021
Apr 04, 2021
Apr 03, 2021
Apr 02, 2021
Apr 01, 2021
Mar 31, 2021
Mar 30, 2021
Mar 29, 2021
Mar 28, 2021
Mar 27, 2021
Mar 26, 2021
Mar 25, 2021
Mar 24, 2021
Mar 23, 2021
Mar 22, 2021
Mar 21, 2021
Mar 20, 2021
Mar 19, 2021
Mar 18, 2021
Mar 17, 2021
Mar 16, 2021
Mar 15, 2021
Mar 14, 2021
Mar 13, 2021
Mar 12, 2021
Mar 11, 2021
Mar 10, 2021
Mar 09, 2021
Mar 08, 2021
Mar 07, 2021
Mar 06, 2021
Mar 05, 2021
Mar 04, 2021
Mar 03, 2021
Mar 02, 2021
Mar 01, 2021
Feb 28, 2021
Feb 27, 2021
Feb 26, 2021
Feb 25, 2021
Feb 24, 2021
Feb 23, 2021
Feb 22, 2021
Feb 21, 2021
Feb 20, 2021
Feb 19, 2021
Feb 18, 2021
Feb 17, 2021
Feb 16, 2021
Feb 15, 2021
Feb 14, 2021
Feb 13, 2021
Feb 12, 2021
Feb 11, 2021
Feb 10, 2021
Feb 09, 2021
Feb 08, 2021
Feb 07, 2021
Feb 06, 2021
Feb 05, 2021
Feb 04, 2021
Feb 03, 2021
Feb 02, 2021
Feb 01, 2021
Jan 31, 2021
Jan 30, 2021
Jan 29, 2021


Jan 28, 2021
Jan 27, 2021
Jan 26, 2021
Jan 25, 2021
Jan 24, 2021
Jan 23, 2021
Jan 22, 2021
Jan 21, 2021
Jan 20, 2021
Jan 19, 2021
Jan 18, 2021
Jan 17, 2021
Jan 16, 2021
Jan 15, 2021
Jan 14, 2021
Jan 13, 2021
Jan 12, 2021
Jan 11, 2021
Jan 10, 2021
Jan 09, 2021
Jan 08, 2021
Jan 07, 2021
Jan 06, 2021
Jan 05, 2021
Jan 04, 2021
Jan 03, 2021
Jan 02, 2021
Jan 01, 2021
Dec 31, 2020
Dec 30, 2020
Dec 29, 2020
Dec 28, 2020
Dec 27, 2020
Dec 26, 2020
Dec 25, 2020
Dec 24, 2020
Dec 23, 2020
Dec 22, 2020
Dec 21, 2020
Dec 20, 2020
Dec 19, 2020
Dec 18, 2020
Dec 17, 2020
Dec 16, 2020
Dec 15, 2020
Dec 14, 2020
Dec 13, 2020
Dec 12, 2020
Dec 11, 2020
Dec 10, 2020
Dec 09, 2020
Dec 08, 2020
Dec 07, 2020
Dec 06, 2020
Dec 05, 2020
Dec 04, 2020
Dec 03, 2020
Dec 02, 2020
Dec 01, 2020
Nov 30, 2020
Nov 29, 2020
Nov 28, 2020
Nov 27, 2020
Nov 26, 2020
Nov 25, 2020
Nov 24, 2020
Nov 23, 2020
Nov 22, 2020
Nov 21, 2020
Nov 20, 2020
Nov 19, 2020
Nov 18, 2020
Nov 17, 2020
Nov 16, 2020
Nov 15, 2020
Nov 14, 2020
Nov 13, 2020
Nov 12, 2020
Nov 11, 2020
Nov 10, 2020
Nov 09, 2020
Nov 08, 2020
Nov 07, 2020
Nov 06, 2020


Nov 05, 2020
Nov 04, 2020
Nov 03, 2020
Nov 02, 2020
Nov 01, 2020
Oct 31, 2020
Oct 30, 2020
Oct 29, 2020
Oct 28, 2020
Oct 27, 2020
Oct 26, 2020
Oct 25, 2020
Oct 24, 2020
Oct 23, 2020
Oct 22, 2020
Oct 21, 2020
Oct 20, 2020
Oct 19, 2020
Oct 18, 2020
Oct 17, 2020
Oct 16, 2020
Oct 15, 2020
Oct 14, 2020
Oct 13, 2020
Oct 12, 2020
Oct 11, 2020
Oct 10, 2020
Oct 09, 2020
Oct 08, 2020
Oct 07, 2020
Oct 06, 2020
Oct 05, 2020
Oct 04, 2020
Oct 03, 2020
Oct 02, 2020
Oct 01, 2020
Sep 30, 2020
Sep 29, 2020
Sep 28, 2020
Sep 27, 2020
Sep 26, 2020
Sep 25, 2020
Sep 24, 2020
Sep 23, 2020
Sep 22, 2020
Sep 21, 2020
Sep 20, 2020
Sep 19, 2020
Sep 18, 2020
Sep 17, 2020
Sep 16, 2020
Sep 15, 2020
Sep 14, 2020
Sep 13, 2020
Sep 12, 2020
Sep 11, 2020
Sep 10, 2020
Sep 09, 2020
Sep 08, 2020
Sep 07, 2020
Sep 06, 2020
Sep 05, 2020
Sep 04, 2020
Sep 03, 2020
Sep 02, 2020
Sep 01, 2020
Aug 31, 2020
Aug 30, 2020
Aug 29, 2020
Aug 28, 2020
Aug 27, 2020
Aug 26, 2020
Aug 25, 2020
Aug 24, 2020
Aug 23, 2020
Aug 22, 2020
Aug 21, 2020
Aug 20, 2020
Aug 19, 2020
Aug 18, 2020
Aug 17, 2020
Aug 16, 2020
Aug 15, 2020
Aug 14, 2020
Aug 13, 2020
Price
Open
High
Low
10000
20000
30000
40000
50000
60000


Aug 12, 2020
Aug 11, 2020
Aug 10, 2020
Aug 09, 2020
Aug 08, 2020
Aug 07, 2020
Aug 06, 2020
Aug 05, 2020
Aug 04, 2020
Aug 03, 2020
Aug 02, 2020
Aug 01, 2020
Jul 31, 2020
Jul 30, 2020
Jul 29, 2020
Jul 28, 2020
Jul 27, 2020
Jul 26, 2020
Jul 25, 2020
Jul 24, 2020
Jul 23, 2020
Jul 22, 2020
Jul 21, 2020
Jul 20, 2020
Jul 19, 2020
Jul 18, 2020
Jul 17, 2020
Jul 16, 2020
Jul 15, 2020
Jul 14, 2020
Jul 13, 2020
Jul 12, 2020
Jul 11, 2020
Jul 10, 2020
Jul 09, 2020
Jul 08, 2020
Jul 07, 2020
Jul 06, 2020
Jul 05, 2020
Jul 04, 2020
Jul 03, 2020
Jul 02, 2020
Jul 01, 2020
Jun 30, 2020
Jun 29, 2020
Jun 28, 2020
Jun 27, 2020
Jun 26, 2020
Jun 25, 2020
Jun 24, 2020
Jun 23, 2020
Jun 22, 2020
Jun 21, 2020
Jun 20, 2020
Jun 19, 2020
Jun 18, 2020
Jun 17, 2020
Jun 16, 2020
Jun 15, 2020
Jun 14, 2020
Jun 13, 2020
Jun 12, 2020
Jun 11, 2020
Jun 10, 2020
Jun 09, 2020
Jun 08, 2020
Jun 07, 2020
Jun 06, 2020
Jun 05, 2020
Jun 04, 2020
Jun 03, 2020
Jun 02, 2020
Jun 01, 2020
May 31, 2020
May 30, 2020
May 29, 2020
May 28, 2020
May 27, 2020
May 26, 2020
May 25, 2020
May 24, 2020
May 23, 2020
May 22, 2020
May 21, 2020

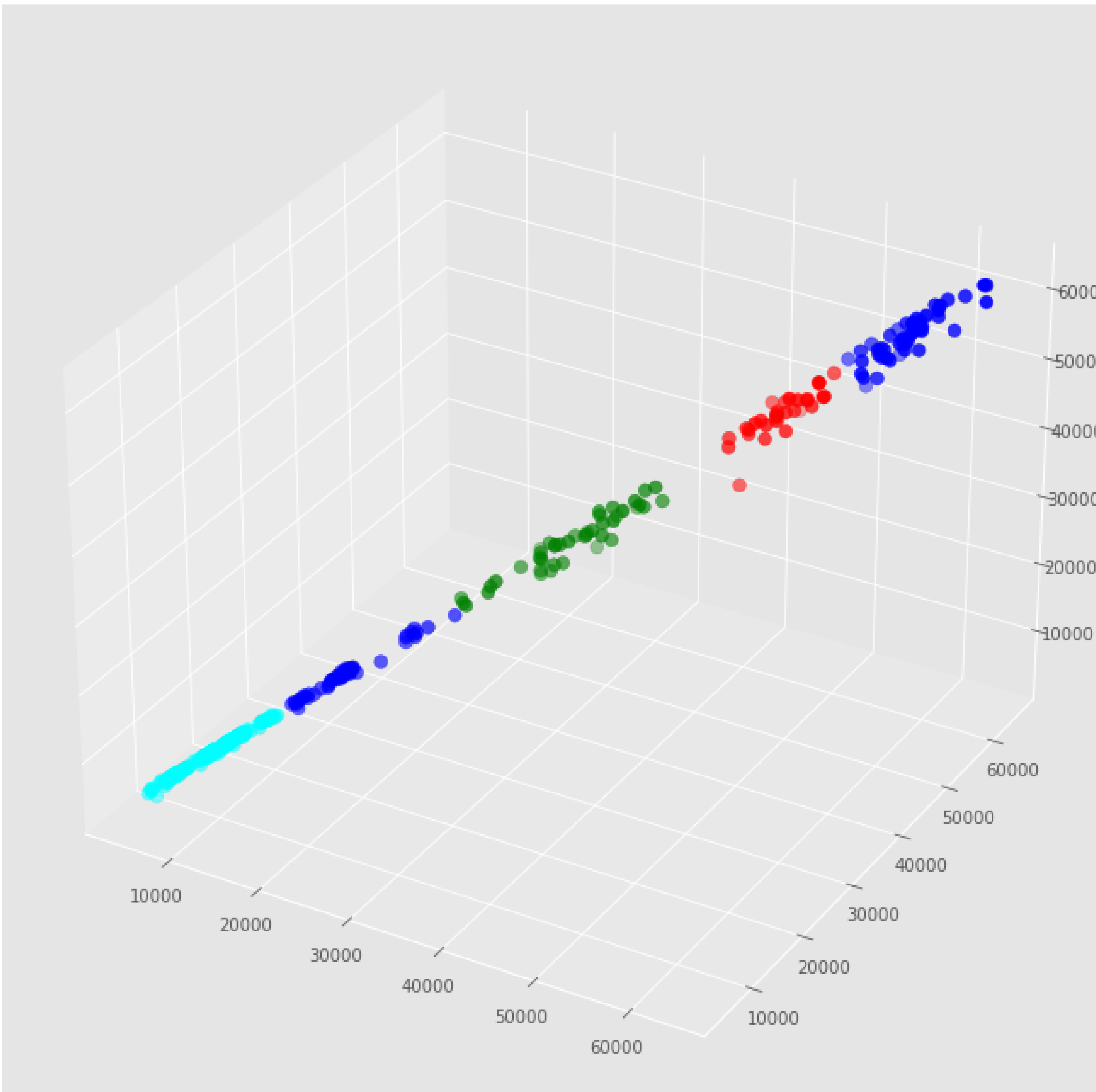
May 20, 2020
May 19, 2020
May 18, 2020
May 17, 2020
May 16, 2020
May 15, 2020
May 14, 2020
May 13, 2020
May 12, 2020
May 11, 2020
May 10, 2020
May 09, 2020
May 08, 2020
May 07, 2020
May 06, 2020
May 05, 2020
May 04, 2020
May 03, 2020
May 02, 2020
May 01, 2020
Apr 30, 2020
Apr 29, 2020
Apr 28, 2020
Apr 27, 2020
Apr 26, 2020
Apr 25, 2020
Apr 24, 2020
Apr 23, 2020
Apr 22, 2020
Apr 21, 2020
Apr 20, 2020
Apr 19, 2020
Apr 18, 2020
Apr 17, 2020
Apr 16, 2020
Apr 15, 2020
Apr 14, 2020
Apr 13, 2020
Apr 12, 2020
Apr 11, 2020
Apr 10, 2020
Apr 09, 2020
Apr 08, 2020
Apr 07, 2020
Apr 06, 2020
Apr 05, 2020
Apr 04, 2020
Apr 03, 2020
Apr 02, 2020
Apr 01, 2020
Mar 31, 2020
Mar 30, 2020
Mar 29, 2020
Mar 28, 2020
Mar 27, 2020
Mar 26, 2020
Mar 25, 2020
Mar 24, 2020
Mar 23, 2020
Mar 22, 2020
Mar 21, 2020
Mar 20, 2020
Mar 19, 2020
Mar 18, 2020
Mar 17, 2020
Mar 16, 2020
Mar 15, 2020
Mar 14, 2020
Mar 13, 2020
Mar 12, 2020
Mar 11, 2020
Mar 10, 2020
Mar 09, 2020
Mar 08, 2020
Mar 07, 2020
Mar 06, 2020
Mar 05, 2020
Mar 04, 2020
Mar 03, 2020
Mar 02, 2020
Mar 01, 2020

Creamos el modelo

In [23]:
```
# Para el ejercicio, sólo seleccionamos 3 dimensiones, para poder graficarlo
X = np.array(dataframe[["Price", "High", "Low"]])
y = np.array(dataframe['Date'])
X.shape
```

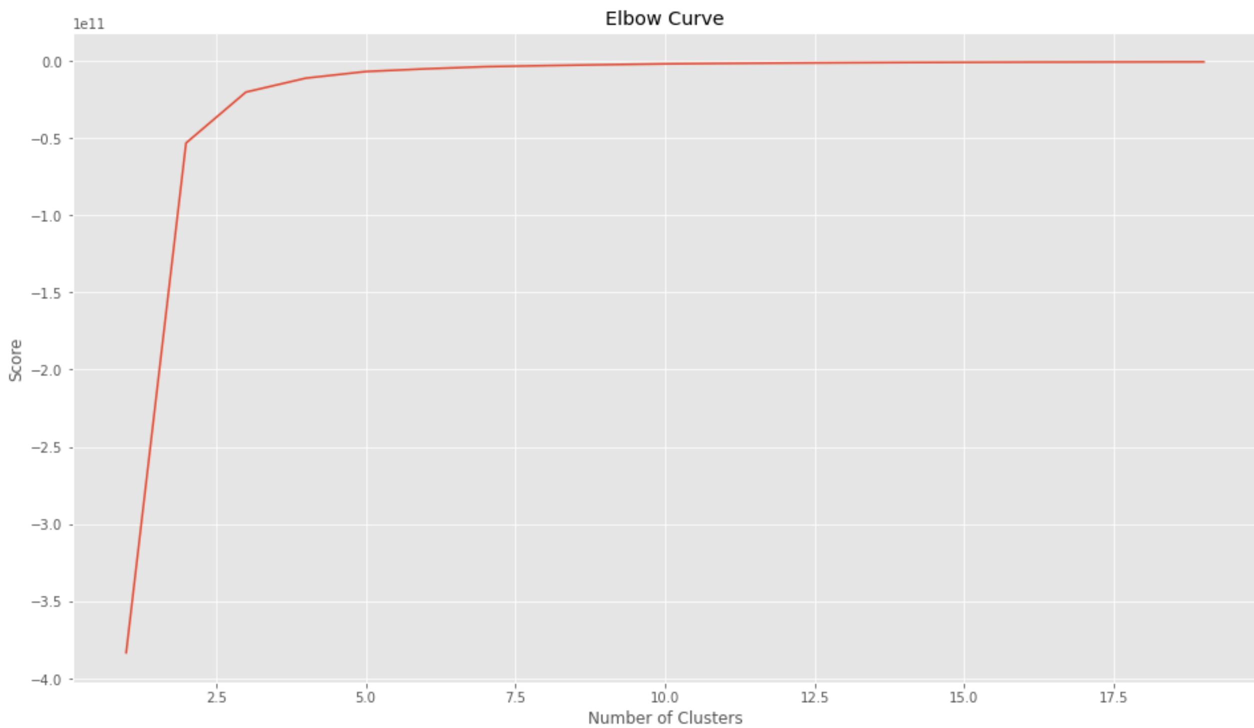
Out[23]: (421, 3)

In [38]:
```
fig = plt.figure()
ax = Axes3D(fig)
colores = ['blue', 'red', 'green','blue', 'cyan', 'yellow', 'orange', 'black', 'pink',
asignar = []
for row in labels:
    asignar.append(colores[row])
ax.scatter(X[:, 0], X[:, 1], X[:, 2], c = asignar, s = 60);
```



Buscamos el valor K

In [25]:
```
Nc = range(1, 20)
kmeans = [KMeans(n_clusters = i) for i in Nc]
kmeans
score = [kmeans[i].fit(X).score(X) for i in range(len(kmeans))]
score
plt.plot(Nc, score)
plt.xlabel('Number of Clusters')
plt.ylabel('Score')
plt.title('Elbow Curve')
plt.show()
```

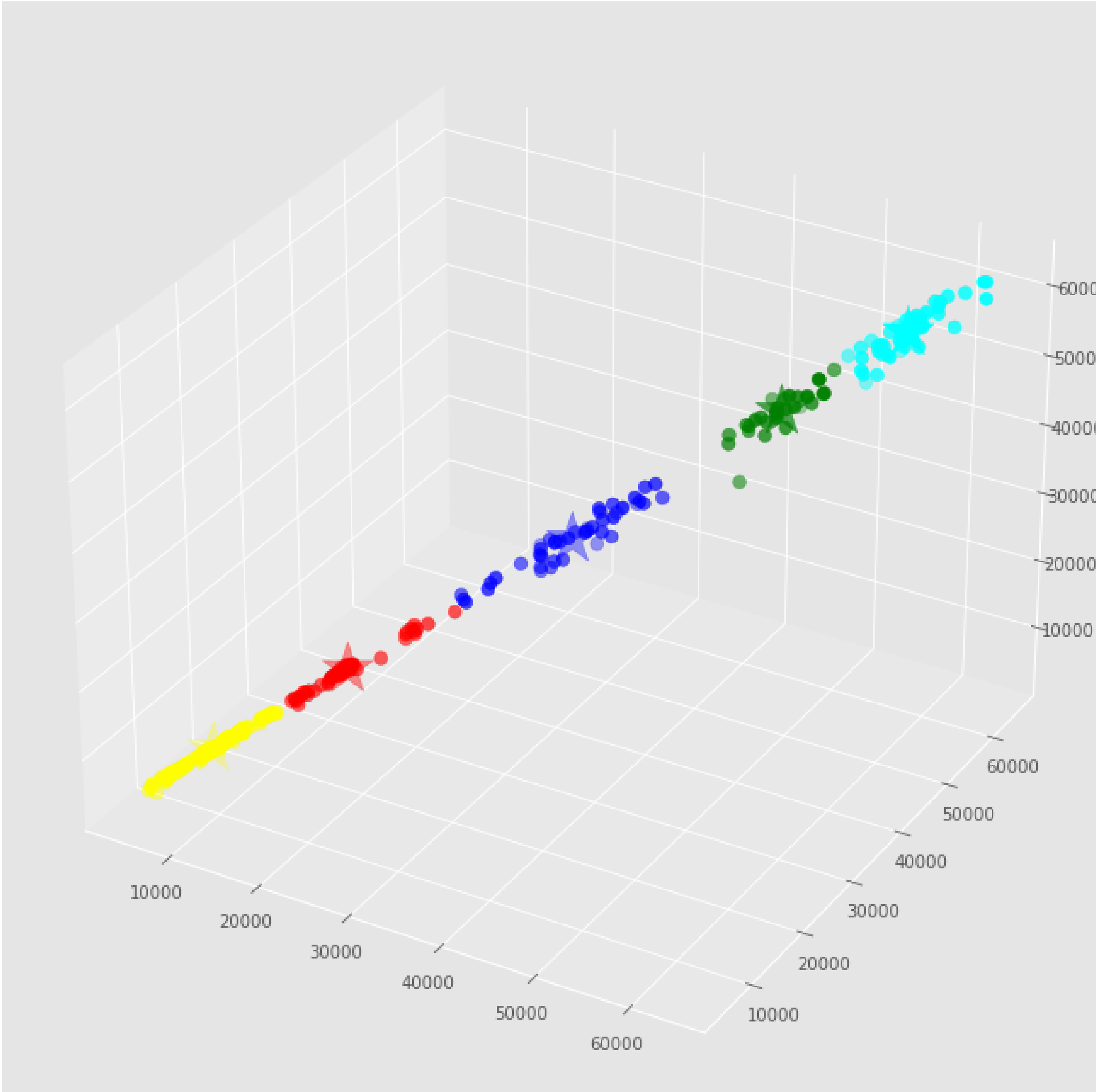


In [26]:
```
# Para el ejercicio, elijo 5 como un buen valor de K, pero podría ser otro.
kmeans = KMeans(n_clusters = 5).fit(X)
centroids = kmeans.cluster_centers_
print(centroids)
```

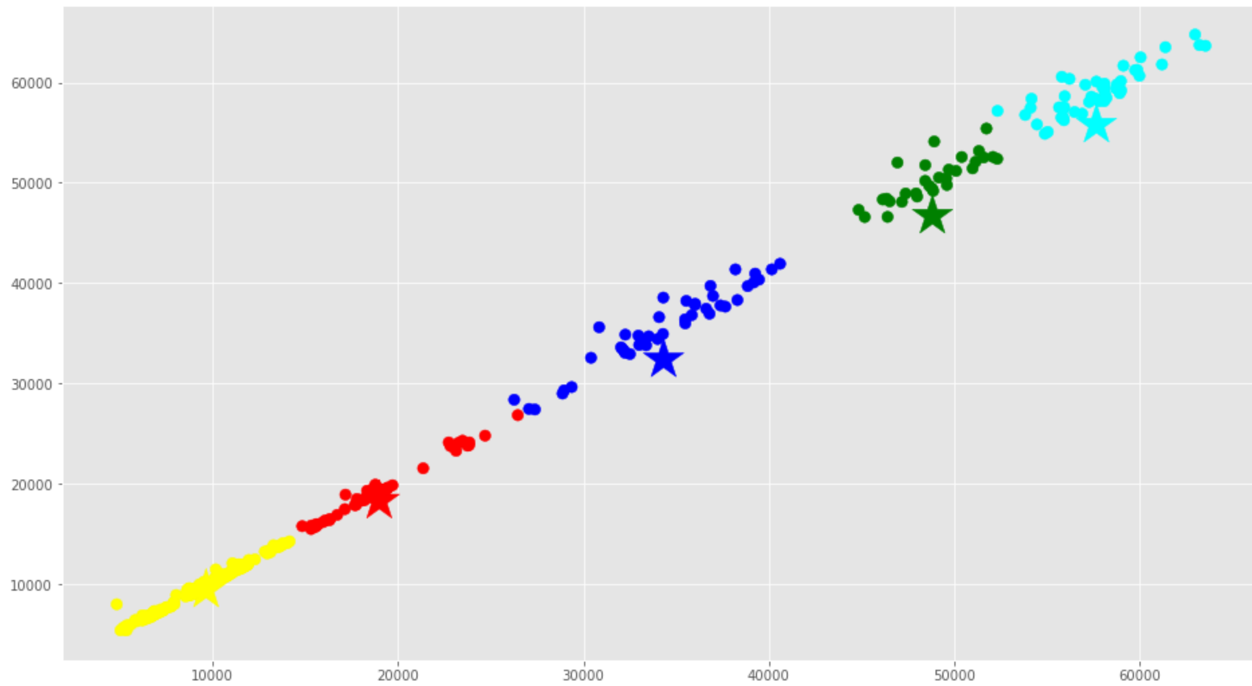
```
[[18985.90384615 19351.15192308 18280.81538462]
 [48816.53666667 50391.33       46695.24      ]
 [34317.38837209 35616.37906977 32374.81860465]
 [57660.5106383  59107.12978723 55776.62553191]
 [ 9646.05341365  9825.5686747   9415.36787149]]
```

In [27]:
```
# Obtener las etiquetas de cada punto de nuestros datos
labels = kmeans.predict(X)
# Obtenemos los centroids
C = kmeans.cluster_centers_
colores = ['red', 'green', 'blue', 'cyan', 'yellow']
asignar = []
for row in labels:
    asignar.append(colores[row])

fig = plt.figure()
ax = Axes3D(fig)
```

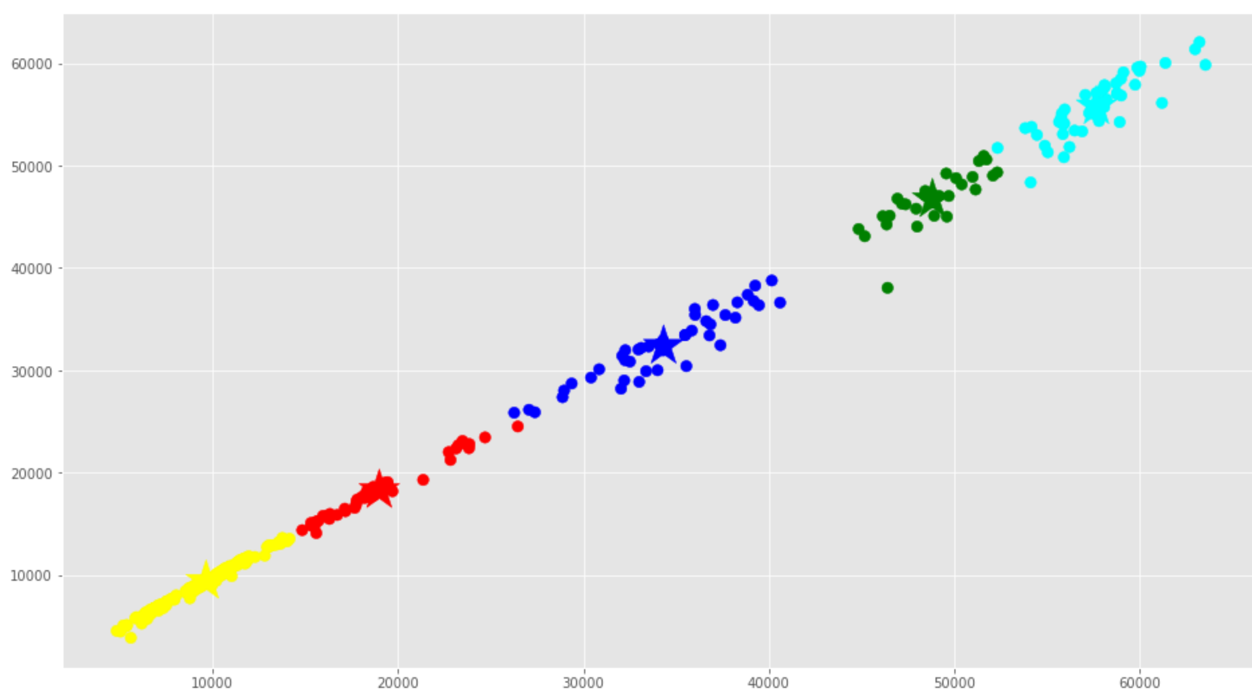
```
ax.scatter(X[:, 0], X[:, 1], X[:, 2], c = asignar, s = 60)
ax.scatter(C[:, 0], C[:, 1], C[:, 2], marker = '*', c = colores, s = 1000);
```



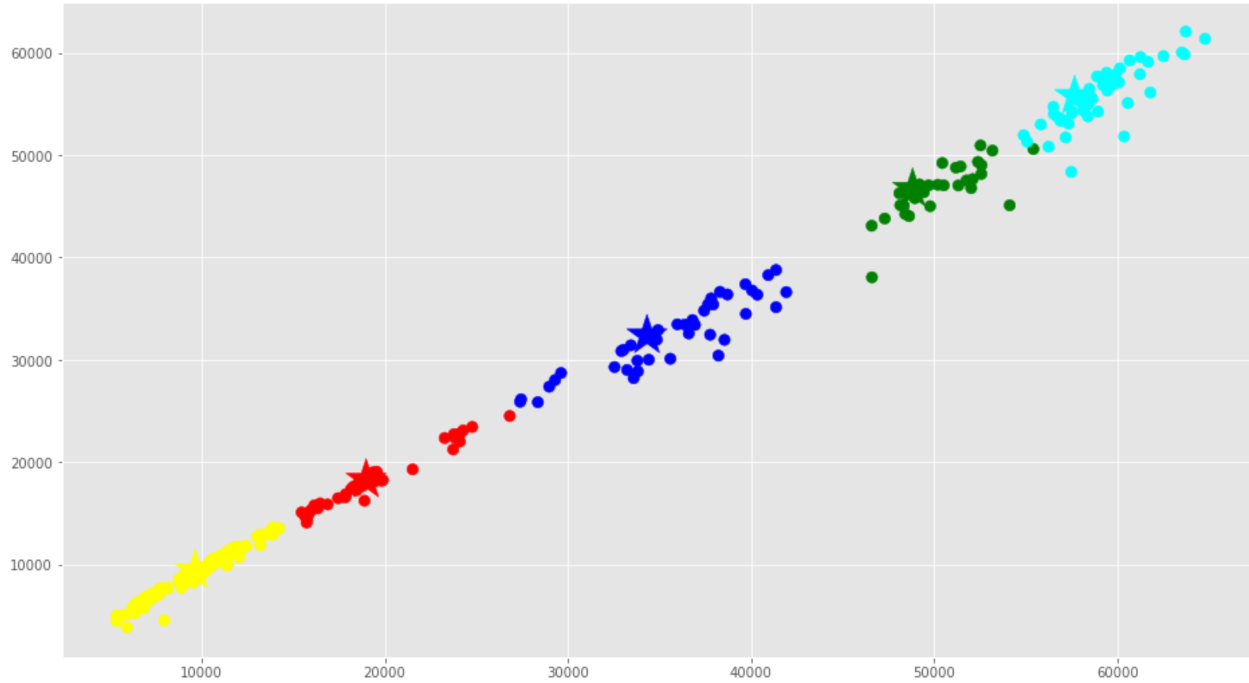
In [28]:
```
# Hacemos una proyección a 2D con los diversos ejes
f1 = dataframe['Price'].values
f2 = dataframe['High'].values

plt.scatter(f1, f2, c = asignar, s = 70)
plt.scatter(C[:, 0], C[:, 2], marker = '*', c = colores, s = 1000);
plt.show()
```



In [29]:
```
f1 = dataframe['Price'].values
f2 = dataframe['Low'].values

plt.scatter(f1, f2, c = asignar, s = 70)
plt.scatter(C[:, 0], C[:, 2], marker = '*', c = colores, s = 1000);
plt.show()
```



In [30]:
```
f1 = dataframe['High'].values
f2 = dataframe['Low'].values

plt.scatter(f1, f2, c = asignar, s = 70)
plt.scatter(C[:, 0], C[:, 2], marker = '*', c = colores, s = 1000);
plt.show()
```



Evaluando los resultados

In [39]:
```
print (classification_report(labels, labels));
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        52
           1       1.00      1.00      1.00        30
           2       1.00      1.00      1.00        43
           3       1.00      1.00      1.00        47
           4       1.00      1.00      1.00       249

    accuracy                           1.00       421
   macro avg       1.00      1.00      1.00       421
weighted avg       1.00      1.00      1.00       421
```

Preguntas

¿Crees que estos centros puedan ser representativos de los datos? ¿Por qué?

Si, ya que nos ayuda a tener un mejor análisis de nuestros datos y a ver los de otra manera.

¿Cómo obtuviste el valor de k a usar?

En la actividad decidimos elegir 5 como un buen valor de K para que no nos muestre tantos datos, pero en si pudiéramos a ver elegido cualquier otro valor.

¿Los centros serían más representativos si usaras un valor más alto? ¿Más bajo?

En nuestro punto de vista pensamos que un valor más alto, ya que los datos serian un poco más representativos.

¿Qué distancia tienen los centros entre sí? ¿Hay alguno que este muy cercano a otros?

Tienen casi la misma distancia todos los centros entre sí, solo hay uno centro que tiene menos distancia entre otro y esto hace que este más cercano.

¿Qué pasaría con los centros si tuviéramos muchos outliers en el análisis de cajas y bigotes?

Tendríamos a tener más centros ya que tendríamos más variantes.

¿Qué puedes decir de los datos basándose en los centros?

Que casi todos los datos llegan a ser similares entre si y solo uno puede llegar a cambiar un poco en cuanto a los centros.