Jorge Gerardo Iglesias Ortiz - A01653261

Alejandro Hernández De la Torre - A01651516

Santiago Orozco Quintero - A01658308

Carlos Andres Barredeaz Rios - A01653183

Jorge Yepez Frutos - A01652661

Actividad Evaluable Patrones con K-means

1. Carga tus datos.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sb
from sklearn.cluster import KMeans
from sklearn.metrics import pairwise_distances_argmin_min
from datetime import datetime
from datetime import date

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

df1 = pd.read_csv('analisis.csv')
df2 = pd.read_csv('avocado.csv')

print(df2)
```

```
       Unnamed: 0        Date  AveragePrice  Total Volume     4046       4225  \
0               0  2015-12-27          1.33      64236.62  1036.74   54454.85
1               1  2015-12-20          1.35      54876.98   674.28   44638.81
2               2  2015-12-13          0.93     118220.22   794.70  109149.67
3               3  2015-12-06          1.08      78992.15  1132.00   71976.41
4               4  2015-11-29          1.28      51039.60   941.48   43838.39
...           ...         ...           ...           ...      ...        ...
18244           7  2018-02-04          1.63      17074.83  2046.96    1529.20
18245           8  2018-01-28          1.71      13888.04  1191.70    3431.50
18246           9  2018-01-21          1.87      13766.76  1191.92    2452.79
18247          10  2018-01-14          1.93      16205.22  1527.63    2981.04
18248          11  2018-01-07          1.62      17489.58  2894.77    2356.13

          4770  Total Bags  Small Bags  Large Bags  XLarge Bags          type  \
0       48.16     8696.87     8603.62       93.25          0.0  conventional
1       58.33     9505.56     9408.07       97.49          0.0  conventional
2      130.50     8145.35     8042.21      103.14          0.0  conventional
3       72.58     5811.16     5677.40      133.76          0.0  conventional
4       75.78     6183.95     5986.26      197.69          0.0  conventional
...       ...         ...         ...         ...          ...           ...
18244    0.00    13498.67    13066.82      431.85          0.0       organic
18245    0.00     9264.84     8940.04      324.80          0.0       organic
18246  727.94     9394.11     9351.80       42.31          0.0       organic
```

```
18247   727.01    10969.54    10919.54       50.00          0.0       organic
18248   224.53    12014.15    11988.14       26.01          0.0       organic

       year            region
0      2015            Albany
1      2015            Albany
2      2015            Albany
3      2015            Albany
4      2015            Albany
...     ...               ...
18244  2018  WestTexNewMexico
18245  2018  WestTexNewMexico
18246  2018  WestTexNewMexico
18247  2018  WestTexNewMexico
18248  2018  WestTexNewMexico

[18249 rows x 14 columns]
```

Se cargaron los datos de análisis y avocado

2. Si determinas que alguna variable no sirve basándose en la actividad pasada, elimínala y justifica por qué quitaste o no variables.

In [2]:
```
df2[["year","Date"]]
```

Out[2]:

	year	Date
0	2015	2015-12-27
1	2015	2015-12-20
2	2015	2015-12-13
3	2015	2015-12-06
4	2015	2015-11-29
...	...	...
18244	2018	2018-02-04
18245	2018	2018-01-28
18246	2018	2018-01-21
18247	2018	2018-01-14
18248	2018	2018-01-07

18249 rows × 2 columns

In [3]:
```
def year_fraction(datef):
    start = date(datef.year, 1, 1).toordinal()
    year_length = date(datef.year+1, 1, 1).toordinal() - start
    return datef.year + float(datef.toordinal() - start) / year_length

def strtoyearf(v):
    for i in range(0, len(v)):
        v[i] = year_fraction(datetime. strptime(v[i], '%Y-%m-%d').date())
    return v

df2["Date"] = strtoyearf(list(df2["Date"]))
```

```
print(df2["Date"])
```

```
0        2015.986301
1        2015.967123
2        2015.947945
3        2015.928767
4        2015.909589
            ...
18244    2018.093151
18245    2018.073973
18246    2018.054795
18247    2018.035616
18248    2018.016438
Name: Date, Length: 18249, dtype: float64
```

In [4]:
```
df2[["year","Date"]]
```

Out[4]:

	year	Date
0	2015	2015.986301
1	2015	2015.967123
2	2015	2015.947945
3	2015	2015.928767
4	2015	2015.909589
...	...	...
18244	2018	2018.093151
18245	2018	2018.073973
18246	2018	2018.054795
18247	2018	2018.035616
18248	2018	2018.016438

18249 rows × 2 columns

Se podría decir que year era una de las variables que considerábamos quitar y que primero se utilizó para un análisis rápido para su k-means. Una vez que se convirtió la fecha en un valor de año con un decimal que representa los meses y días, la variable year se volvió irrelevante al ser menos preciso para lo que queríamos estudiar. Por lo tanto, las demás variables son relevantes debido a que ofrecen mucha información incluso si está en string. De estas variables decidimos usar como "y" el average price para saber cómo es que también las otras variables seleccionadas que serían los volúmenes, bolsas y fechas afectaban el precio.

3 y 4. Determina un valor de k. Utilizando scikitlearn calcula los centros del algoritmo k-means.

Análisis.csv

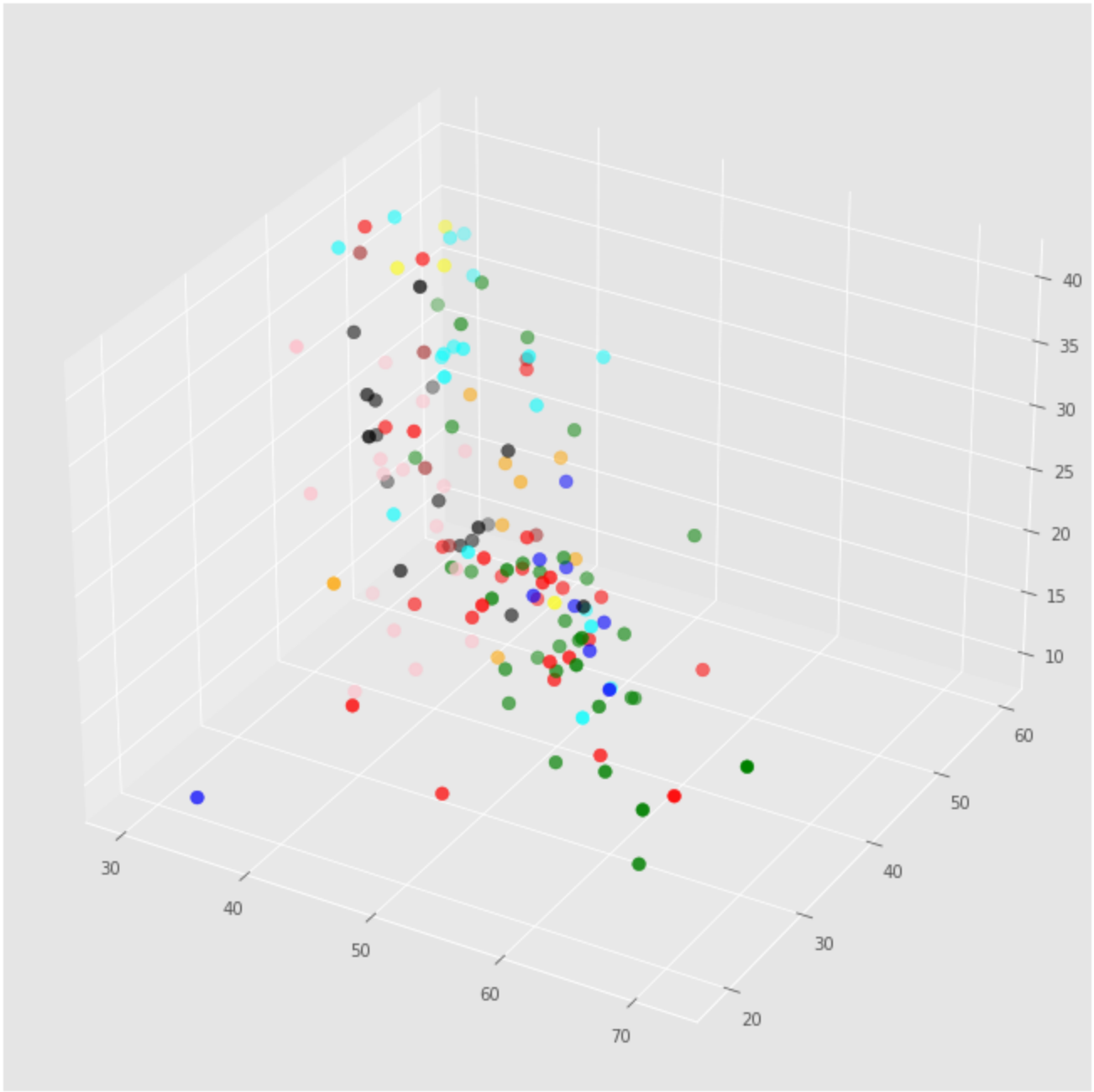
In [5]:
```
X = np.array(df1[["op","ex","ag"]])
y = np.array(df1['categoria'])
X.shape
```

Out[5]:
```
(140, 3)
```

In [6]:
```
fig = plt.figure()
ax = Axes3D(fig)
colores=['blue','red','green','blue','cyan','yellow','orange','black','pink','brown','pu
asignar=[]
for row in y:
    asignar.append(colores[int(row)])
ax.scatter(X[:, 0], X[:, 1], X[:, 2], c=asignar,s=60)
```

```
C:\Users\JGIO\AppData\Local\Temp\ipykernel_9880\724161956.py:2: MatplotlibDeprecationWar
ning: Axes3D(fig) adding itself to the figure is deprecated since 3.4. Pass the keyword
argument auto_add_to_figure=False and use fig.add_axes(ax) to suppress this warning. The
default value of auto_add_to_figure will change to False in mpl3.5 and True values will
no longer work in 3.6.  This is consistent with other Axes classes.
  ax = Axes3D(fig)
```

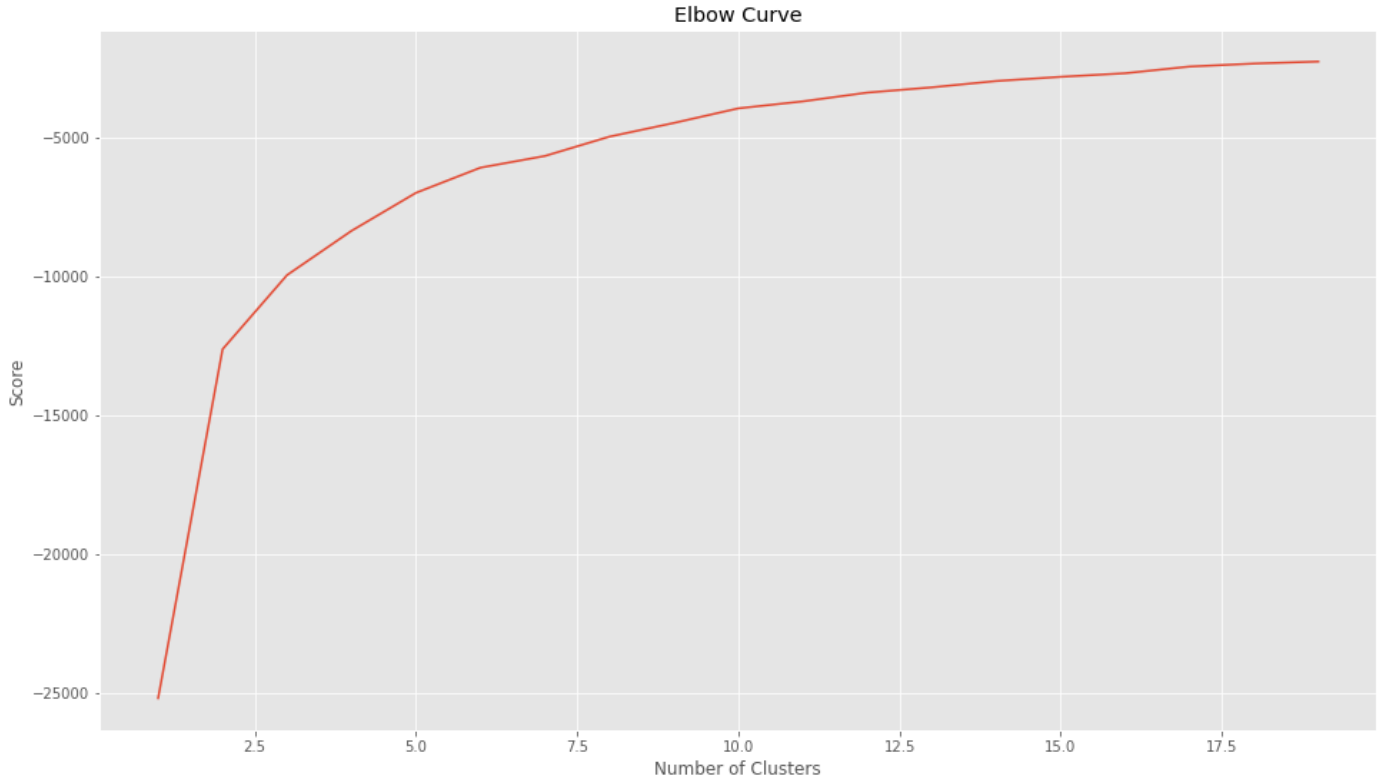
Out[6]:
```
<mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x22dbc46ebe0>
```



In [7]:
```
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()
```

```
C:\Users\JGIO\anaconda3\envs\TC1002S\lib\site-packages\sklearn\cluster\_kmeans.py:1036:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are l
ess chunks than available threads. You can avoid it by setting the environment variable
OMP_NUM_THREADS=1.
  warnings.warn(
```



In [8]:
```
kmeans = KMeans(n_clusters=5).fit(X)
centroids = kmeans.cluster_centers_
print(centroids)
```

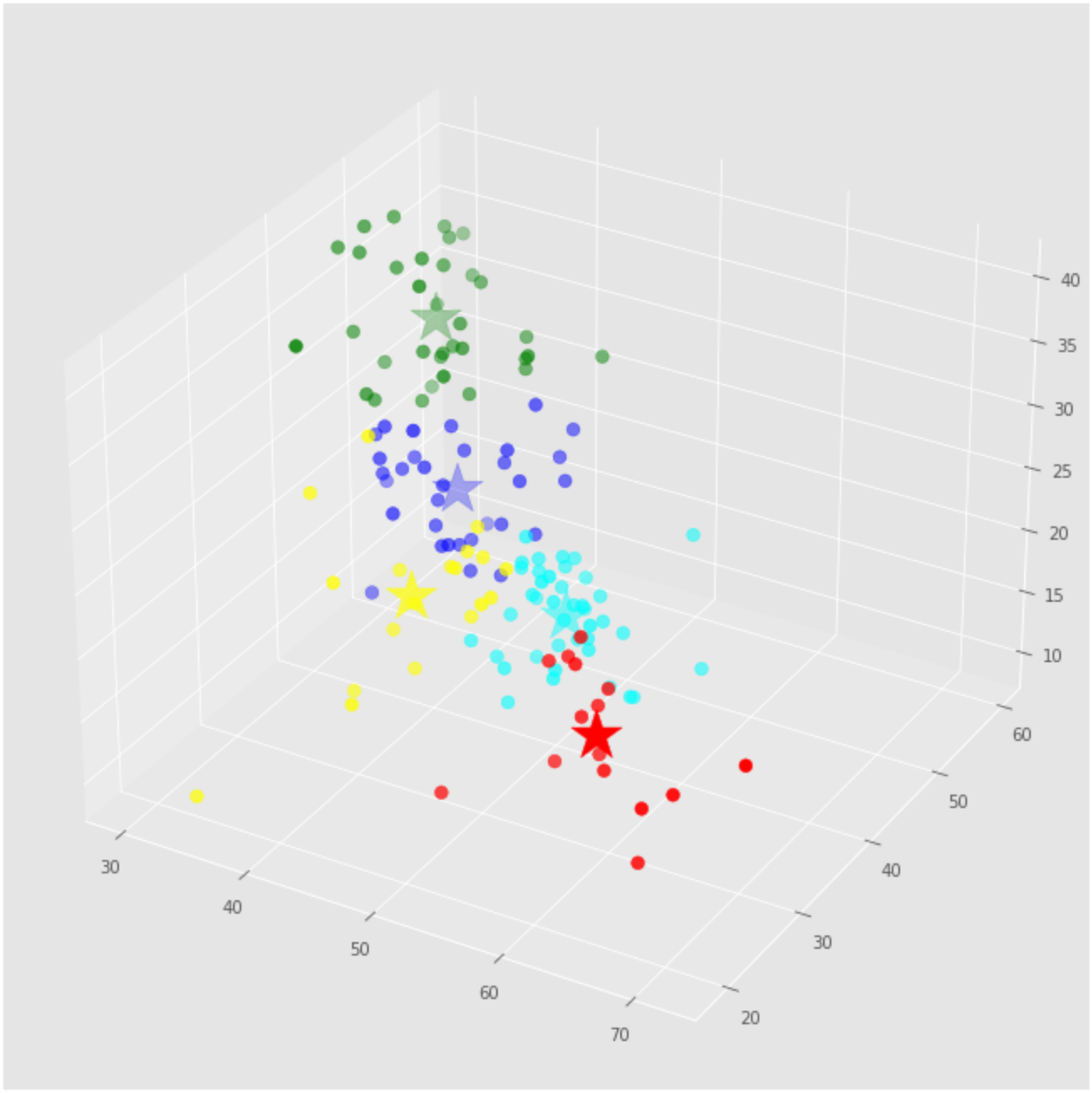
```
[[58.70462307 30.53566167 15.72207033]
 [35.90241306 47.56828232 33.58748762]
 [40.10497084 43.34066891 23.14697356]
 [50.42588653 40.66723528 17.30892463]
 [42.968253   32.53013537 20.93305995]]
```

In [9]:
```
labels = kmeans.predict(X)
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)
```

```
C:\Users\JGIO\AppData\Local\Temp\ipykernel_9880\3420450788.py:9: MatplotlibDeprecationWa
rning: Axes3D(fig) adding itself to the figure is deprecated since 3.4. Pass the keyword
argument auto_add_to_figure=False and use fig.add_axes(ax) to suppress this warning. The
default value of auto_add_to_figure will change to False in mpl3.5 and True values will
no longer work in 3.6.  This is consistent with other Axes classes.
  ax = Axes3D(fig)
```

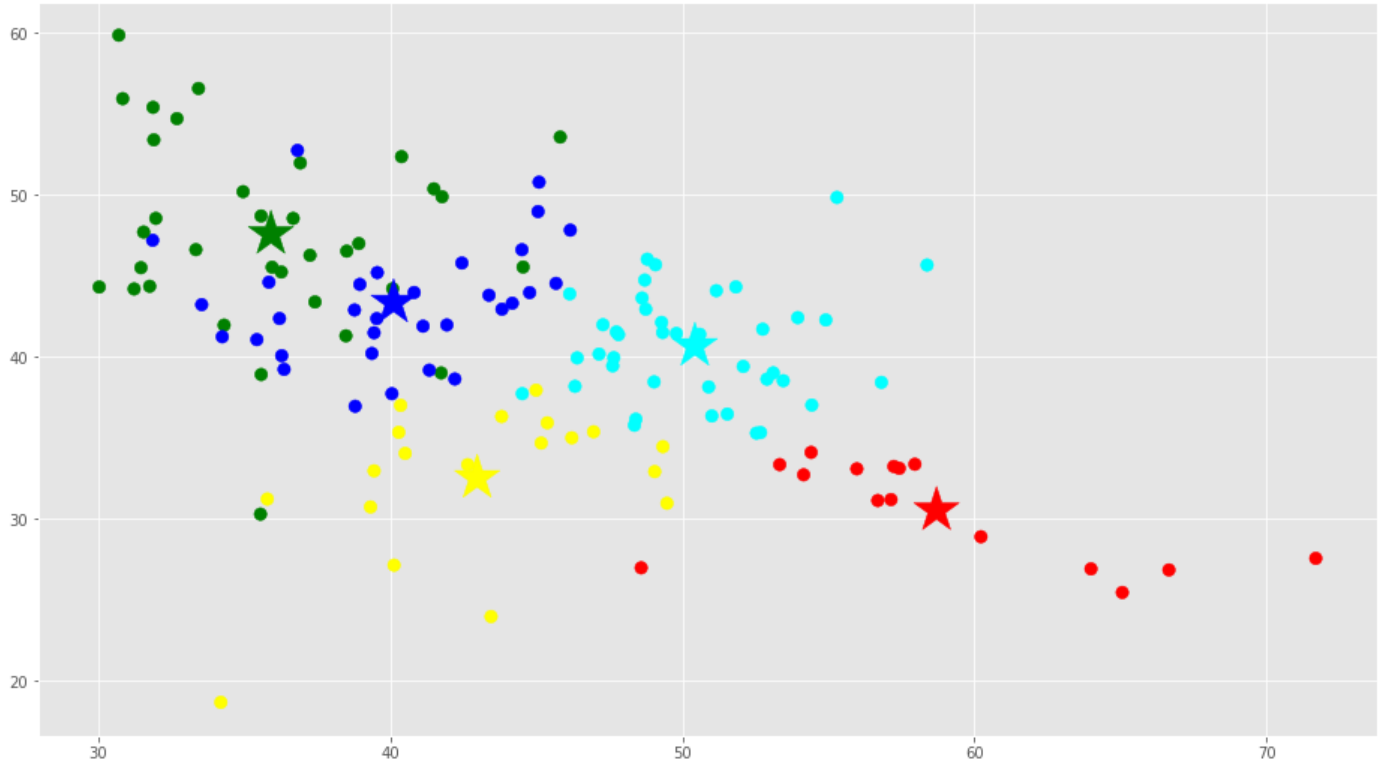
Out[9]:
```
<mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x22dbcee22e0>
```



In [10]:

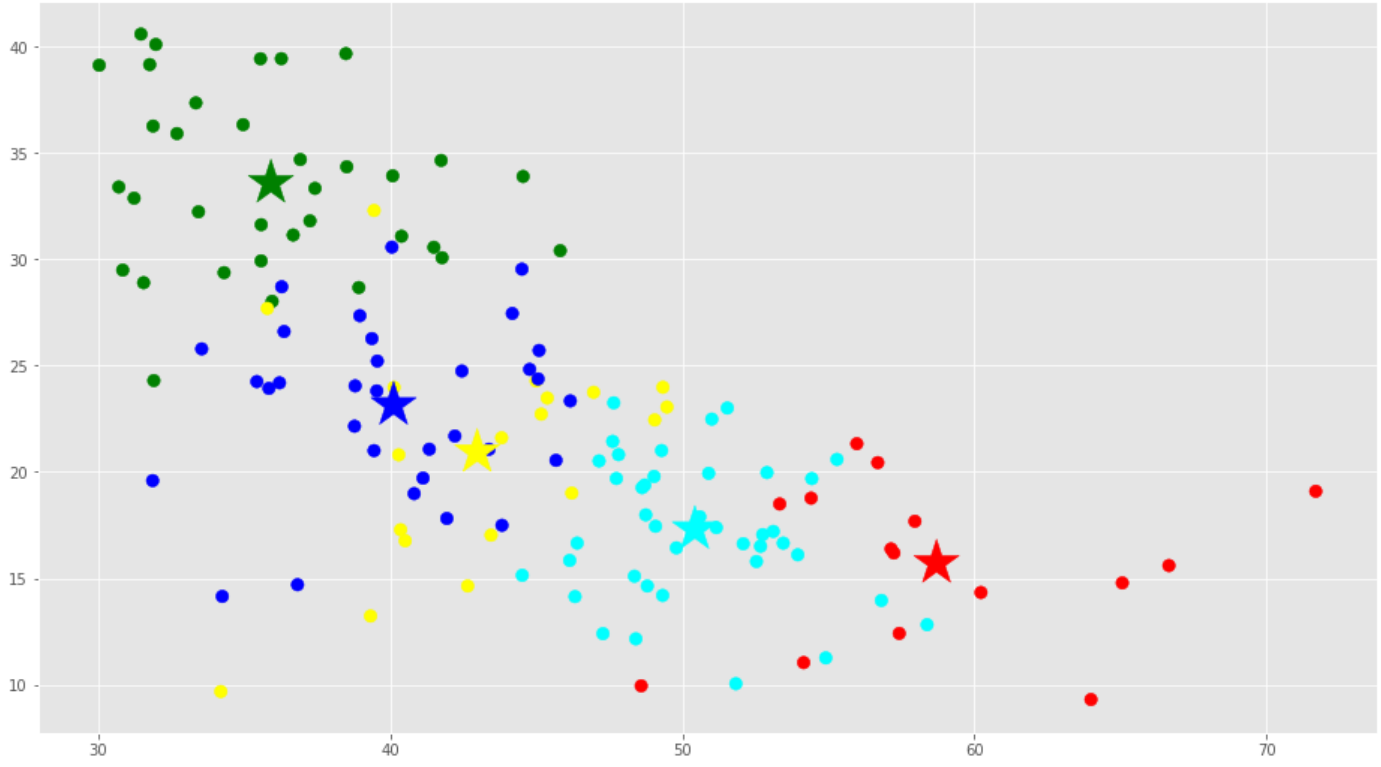
```
f1 = df1['op'].values
f2 = df1['ex'].values

plt.scatter(f1, f2, c=asignar, s=70)
plt.scatter(C[:, 0], C[:, 1], marker='*', c=colores, s=1000)
plt.show()
```



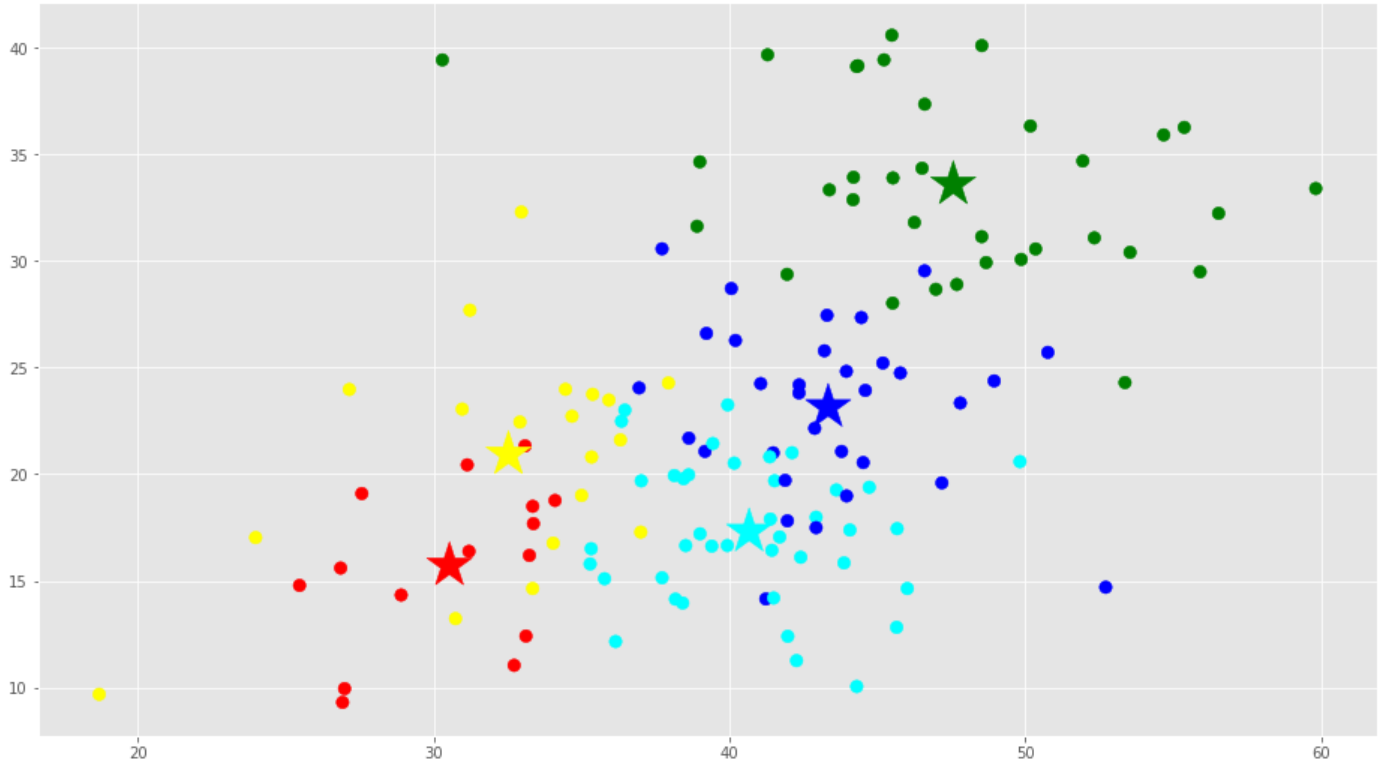
In [11]:
```
f1 = df1['op'].values
f2 = df1['ag'].values

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



In [12]:
```
f1 = df1['ex'].values
f2 = df1['ag'].values

plt.scatter(f1, f2, c=asignar, s=70)
plt.scatter(C[:, 1], C[:, 2], marker='*', c=colores, s=1000)
plt.show()
```



In [13]:
```
copy =  pd.DataFrame()
copy['usuario']=df1['usuario'].values
copy['categoria']=df1['categoria'].values
copy['label'] = labels;
cantidadGrupo =  pd.DataFrame()
cantidadGrupo['color']=colores
cantidadGrupo['cantidad']=copy.groupby('label').size()
cantidadGrupo
```

Out[13]:

	color	cantidad
0	red	15
1	green	34
2	blue	32
3	cyan	40
4	yellow	19

In [14]:
```
group_referrer_index = copy['label'] ==0
group_referrals = copy[group_referrer_index]

diversidadGrupo =  pd.DataFrame()
diversidadGrupo['categoria']=[0,1,2,3,4,5,6,7,8,9]
diversidadGrupo['cantidad']=group_referrals.groupby('categoria').size()
diversidadGrupo
```

Out[14]:

	categoria	cantidad
0	0	NaN
1	1	5.0
2	2	8.0
3	3	1.0
4	4	1.0

5	5	NaN
6	6	NaN
7	7	NaN
8	8	NaN
9	9	NaN

In [15]:
```
closest, _ = pairwise_distances_argmin_min(kmeans.cluster_centers_, X)
closest
```

Out[15]:
```
array([ 82,  98,  64,  21, 120], dtype=int64)
```

In [16]:
```
users=df1['usuario'].values
for row in closest:
    print(users[row])
```

```
JudgeJudy
maria_patino
ierrejon
carmenelectra
SarahPalinUSA
```

In [17]:
```
X_new = np.array([[45.92,57.74,15.66]])

new_labels = kmeans.predict(X_new)
print(new_labels)
```

```
[2]
```

Avocado.csv

In [18]:
```
X = np.array(df2[["Total Volume","Total Bags","Date"]])
y = np.array(df2['AveragePrice'])
X.shape
```

Out[18]:
```
(18249, 3)
```

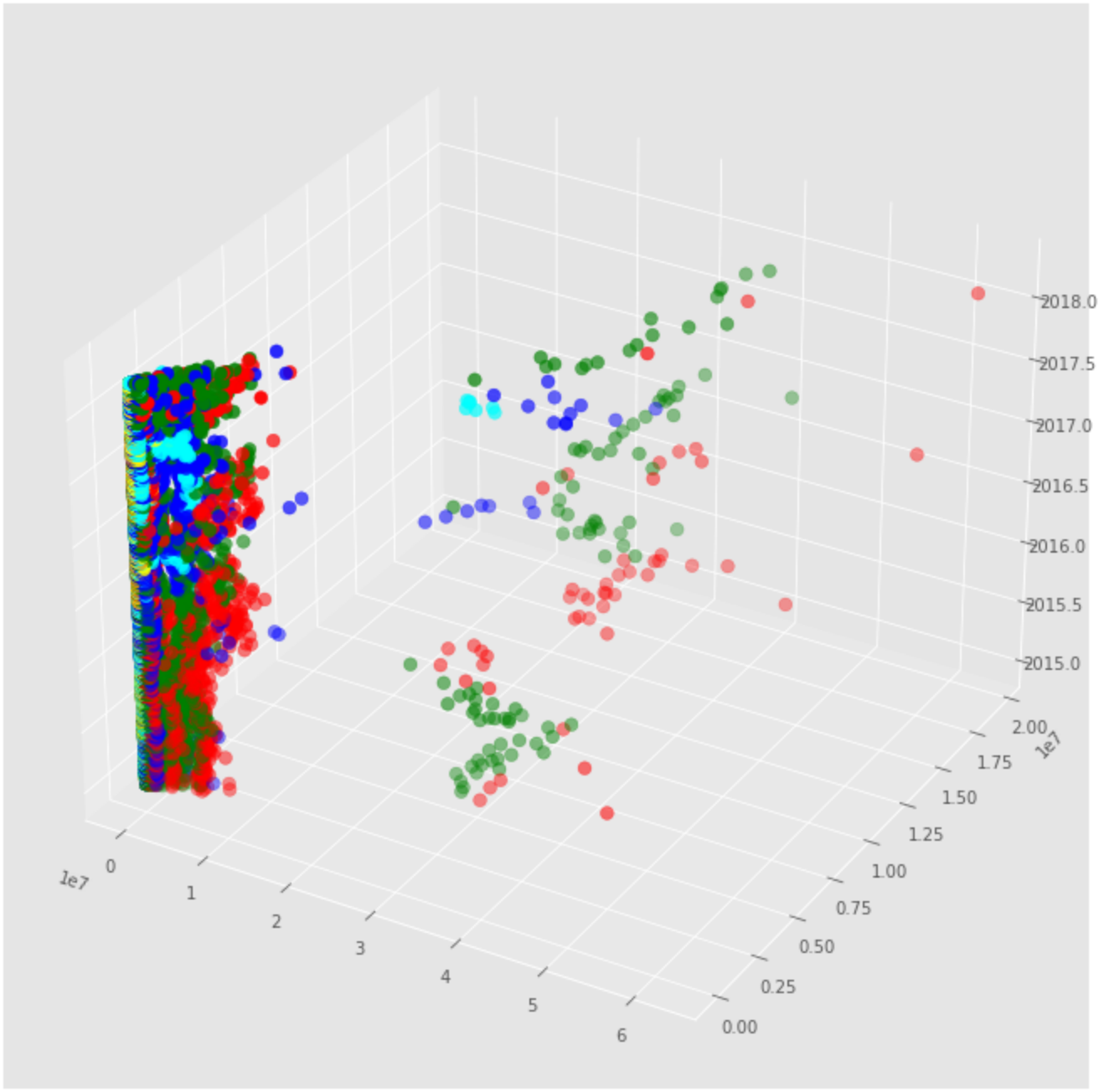
In [19]:
```
min(list(np.array(df2['AveragePrice'])))
#print(max(list(np.array(df2['AveragePrice']))))
```

Out[19]:
```
0.44
```

In [20]:
```
fig = plt.figure()
ax = Axes3D(fig)
colores=['blue','red','green','blue','cyan','yellow','orange','black','pink','brown','pu
asignar=[]
miny = min(list(np.array(df2['AveragePrice'])))
maxy = max(list(np.array(df2['AveragePrice'])))
for row in y:
    asignar.append(colores[int((float(row)-miny)*(12/maxy))])
ax.scatter(X[:, 0], X[:, 1], X[:, 2], c=asignar,s=60)
```

```
C:\Users\JGIO\AppData\Local\Temp\ipykernel_9880\3357876766.py:2: MatplotlibDeprecationWa
rning: Axes3D(fig) adding itself to the figure is deprecated since 3.4. Pass the keyword
argument auto_add_to_figure=False and use fig.add_axes(ax) to suppress this warning. The
default value of auto_add_to_figure will change to False in mpl3.5 and True values will
no longer work in 3.6.  This is consistent with other Axes classes.
  ax = Axes3D(fig)
```

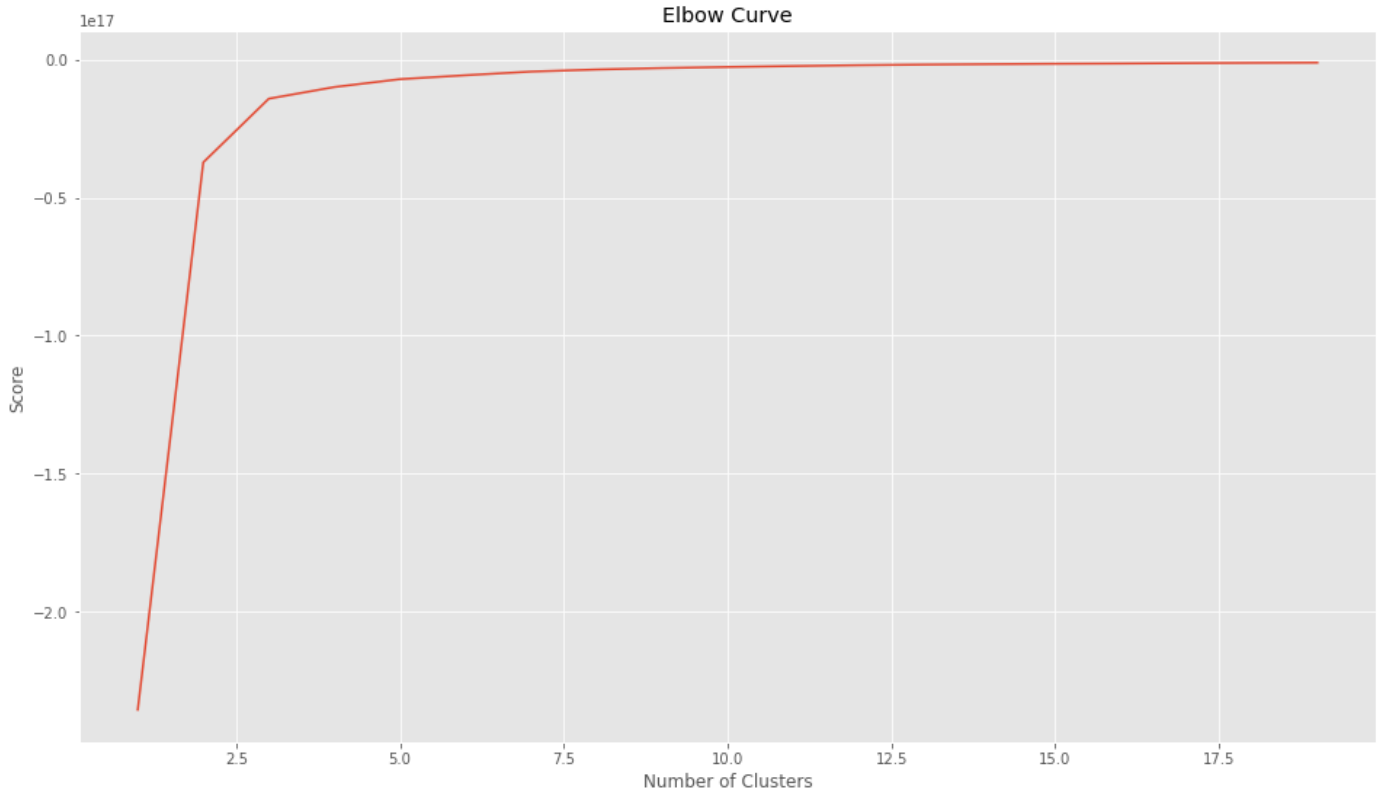
Out[20]:
```
<mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x22dbd692700>
```



In [21]:

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

```
[KMeans(n_clusters=1), KMeans(n_clusters=2), KMeans(n_clusters=3), KMeans(n_clusters=4),
KMeans(n_clusters=5), KMeans(n_clusters=6), KMeans(n_clusters=7), KMeans(), KMeans(n_clu
sters=9), KMeans(n_clusters=10), KMeans(n_clusters=11), KMeans(n_clusters=12), KMeans(n_
clusters=13), KMeans(n_clusters=14), KMeans(n_clusters=15), KMeans(n_clusters=16), KMean
s(n_clusters=17), KMeans(n_clusters=18), KMeans(n_clusters=19)]
[-2.3539280575142707e+17, -3.7264276072727416e+16, -1.4208643635568442e+16, -99958988491
45880.0, -7145581020396259.0, -5726514178884037.0, -4404887115644078.0, -365160916251136
8.5, -3120554495756974.5, -2734327262735823.5, -2395480135684406.0, -2091602403809442.2,
-1872054818897416.0, -1710142116734658.2, -1567896434767288.8, -1475739055240345.0, -134
2936787691762.8, -1235542820656088.8, -1155978376468676.0]
```



In [22]:
```
kmeans = KMeans(n_clusters=3).fit(X)
centroids = kmeans.cluster_centers_
print(centroids)
```

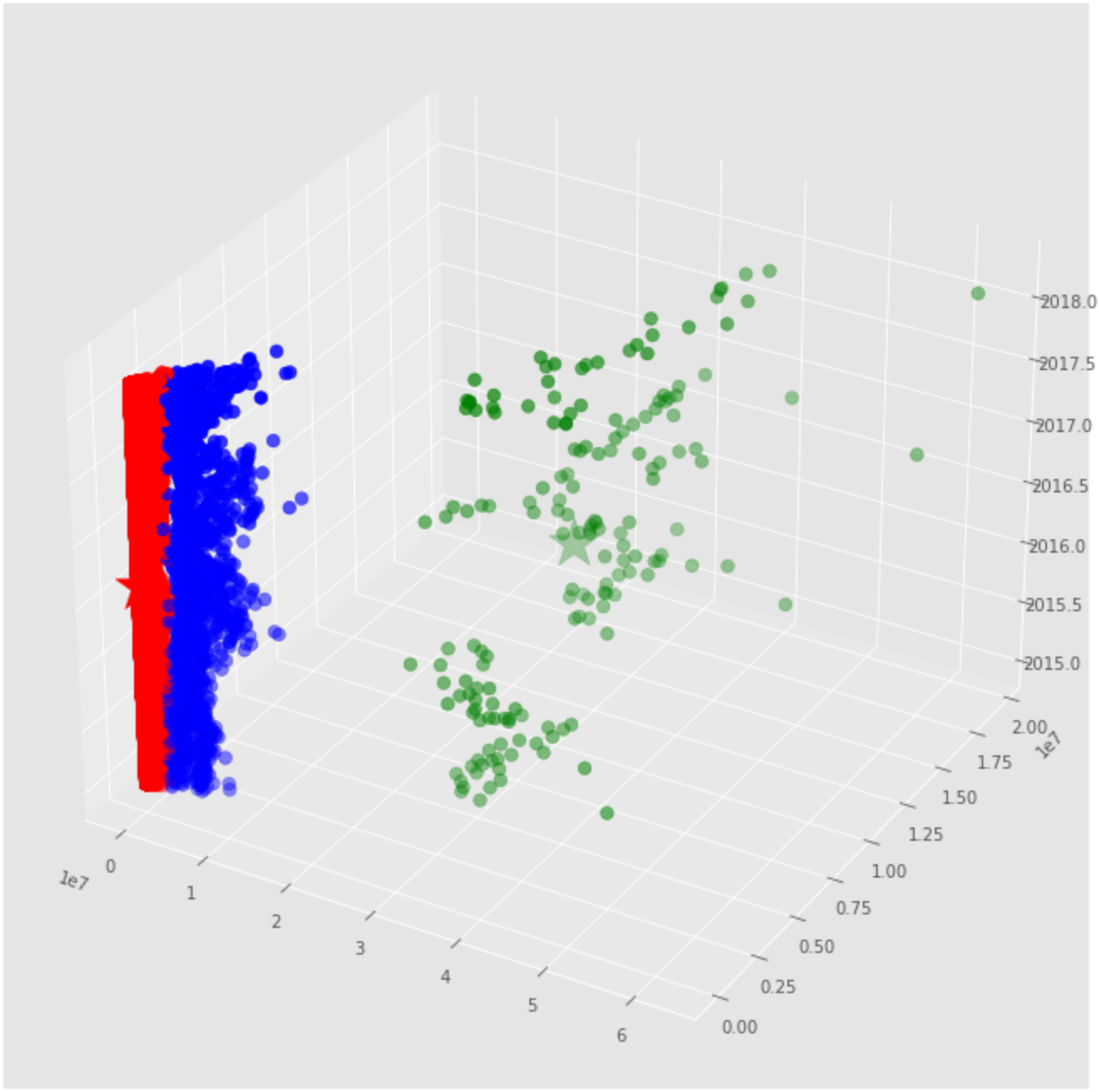
```
[[2.40087076e+05 7.20529553e+04 2.01661710e+03]
 [3.37350390e+07 9.19049275e+06 2.01661771e+03]
 [4.45007684e+06 1.23738713e+06 2.01662483e+03]]
```

In [23]:
```
labels = kmeans.predict(X)
C = kmeans.cluster_centers_
colores=['red','green','blue']
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)
```

```
C:\Users\JGIO\AppData\Local\Temp\ipykernel_9880\1405421501.py:9: MatplotlibDeprecationWarning: Axes3D(fig) adding itself to the figure is deprecated since 3.4. Pass the keyword argument auto_add_to_figure=False and use fig.add_axes(ax) to suppress this warning. The default value of auto_add_to_figure will change to False in mpl3.5 and True values will no longer work in 3.6.  This is consistent with other Axes classes.
  ax = Axes3D(fig)
```

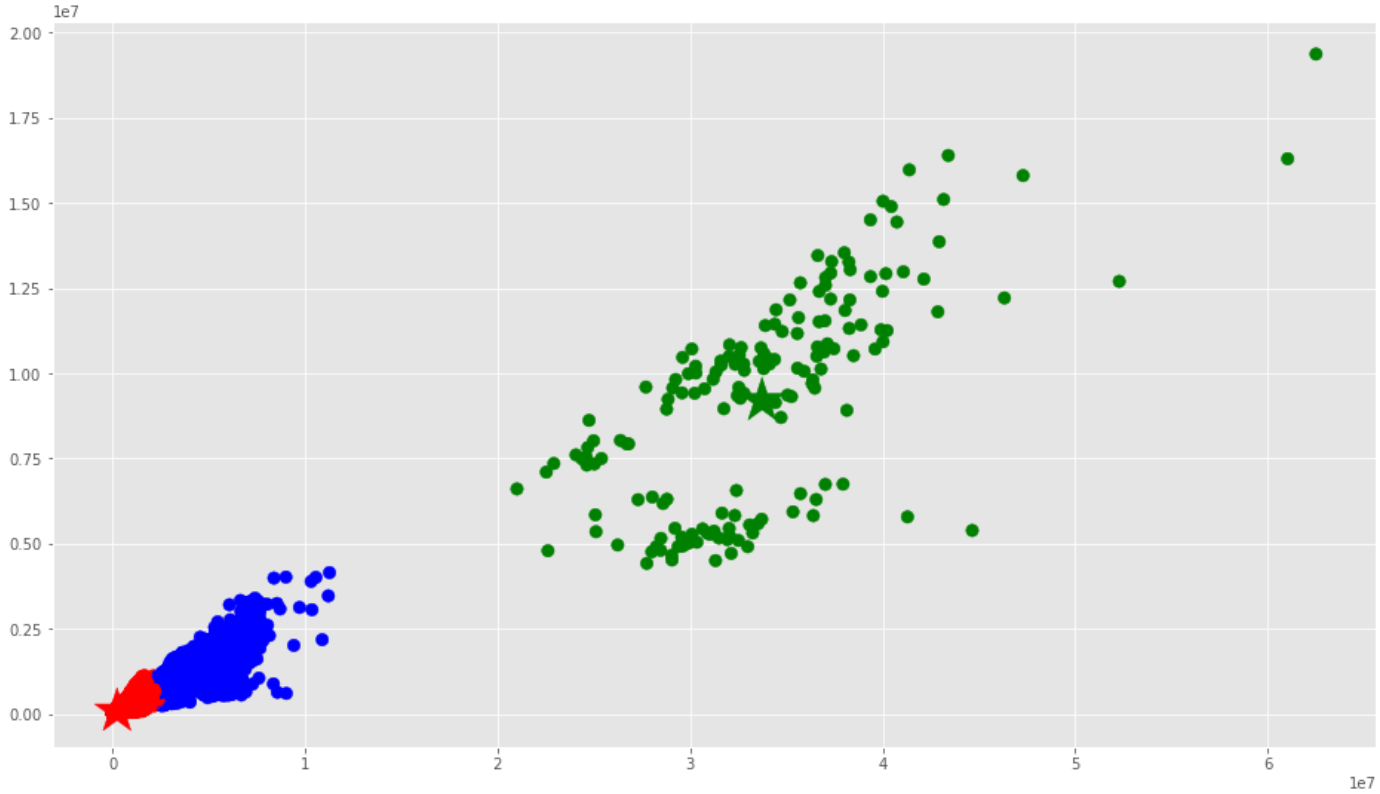
Out[23]:
```
<mpl_toolkits.mplot3d.art3d.Path3DCollection at 0x22dbd6c42e0>
```



In [24]:

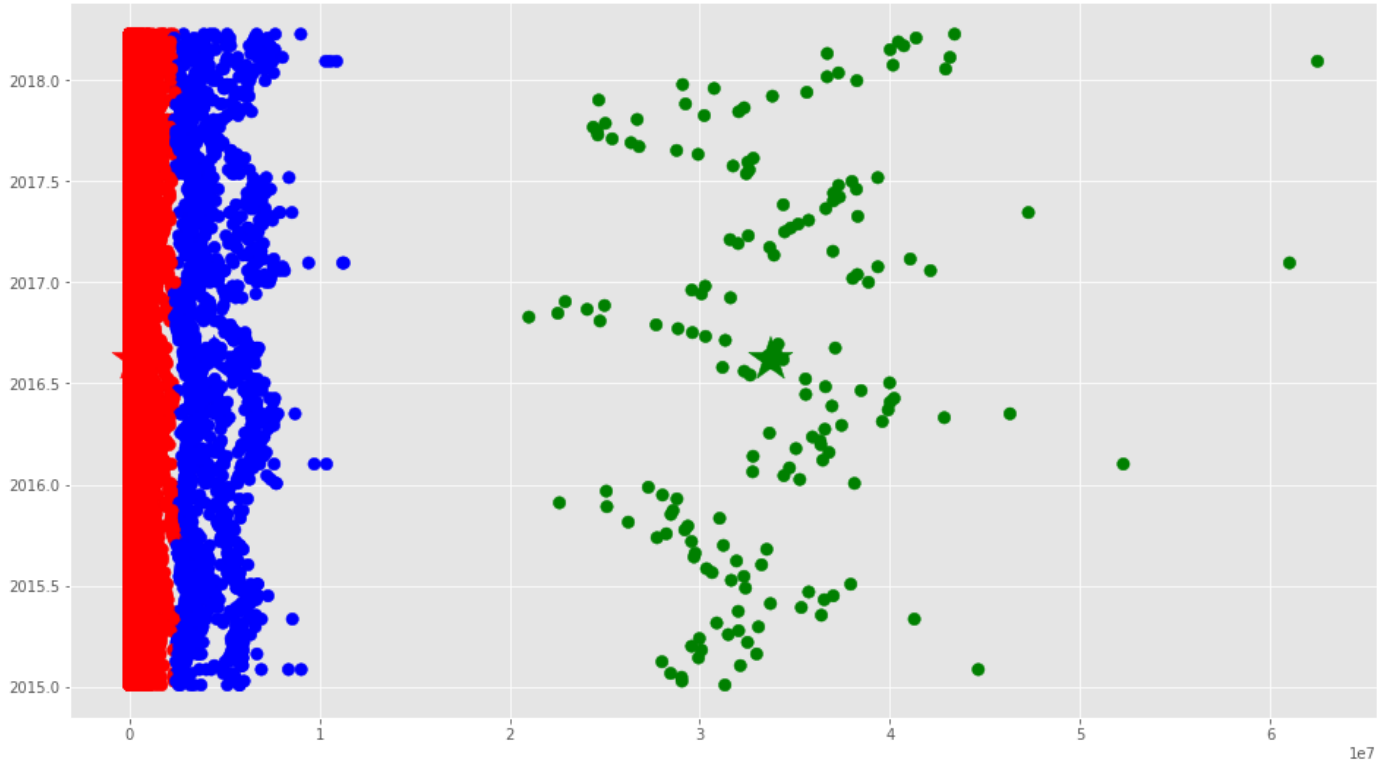
```
f1 = df2['Total Volume'].values
f2 = df2['Total Bags'].values

plt.scatter(f1, f2, c=asignar, s=70)
plt.scatter(C[:, 0], C[:, 1], marker='*', c=colores, s=1000)
plt.show()
```



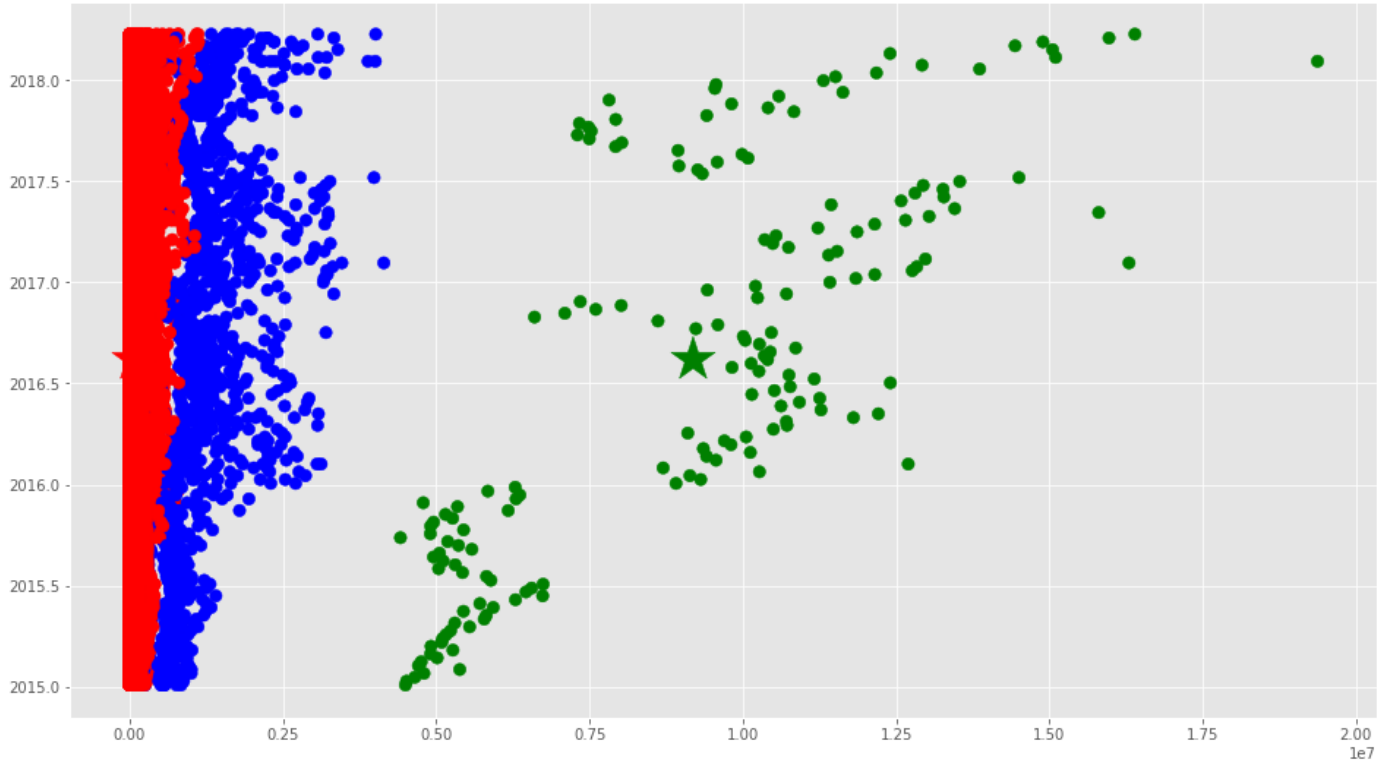
In [25]:
```
f1 = df2['Total Volume'].values
f2 = df2['Date'].values

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



In [26]:
```
f1 = df2['Total Bags'].values
f2 = df2['Date'].values

plt.scatter(f1, f2, c=asignar, s=70)
plt.scatter(C[:, 1], C[:, 2], marker='*', c=colores, s=1000)
plt.show()
```



In [27]:
```
copy =  pd.DataFrame()
copy['region']=df2['region'].values
copy['AveragePrice']=df2['AveragePrice'].values
copy['label'] = labels;
cantidadGrupo =  pd.DataFrame()
cantidadGrupo['color']=colores
cantidadGrupo['cantidad']=copy.groupby('label').size()
cantidadGrupo
```

Out[27]:

	color	cantidad
0	red	16778
1	green	169
2	blue	1302

In [28]:
```
group_referrer_index = copy['label'] ==0
group_referrals = copy[group_referrer_index]

diversidadGrupo =  pd.DataFrame()
diversidadGrupo['AveragePrice']=[0,1,2,3,4,5,6,7,8,9]
diversidadGrupo['cantidad']=group_referrals.groupby('AveragePrice').size()
diversidadGrupo
```

Out[28]:

	AveragePrice	cantidad
0	0	NaN
1	1	143.0
2	2	59.0
3	3	2.0
4	4	NaN
5	5	NaN
6	6	NaN

7	7	NaN
8	8	NaN
9	9	NaN

In [29]:
```
closest, _ = pairwise_distances_argmin_min(kmeans.cluster_centers_, X)
closest
```

Out[29]:
```
array([4190, 5498, 1532], dtype=int64)
```

In [30]:
```
users=df2['region'].values
for row in closest:
    print(users[row])
```

```
Nashville
TotalUS
Northeast
```

Basado en los centros responde las siguientes preguntas:

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

Si, porque de manera similar a como se hace con el diagrama de cajas nos muestra una tendencia en los datos para poder hacer inferencias sobre ellos basados en la proximidad con otro conjunto de valores.

- ¿Cómo obtuviste el valor de k a usar?

Como se vio anteriormente, utilizamos los comandos de Kmeans dentro de la librería de SciKitLearn con la cual devuelve los valores de la k a usar a partir de los parámetros dados. Para un entendimiento mejor se podrían observar los resultados obtenidos en los puntos 3 y 4.

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

No debido a que los grupos de valores que se encontraron no eran tan numerosos de modo que si implementamos una k más grande los grupos serían tan numerosos que la información proporcionada por observar la distribución de estos mismos no sería significativa.

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

In [31]:
```
def distancia3d(p1, p2):
    l = pow((p2[0]-p1[0])**2+(p2[1]-p1[1])**2+(p2[2]-p1[2])**2,0.5)
    return l

centroidsl = centroids.tolist()
for i in range(2):
    for j in range(i+1,3):
        print(f'Punto {i+1} y Punto {j+1}')
        print(distancia3d(centroidsl[i],centroidsl[j]))
```

```
Punto 1 y Punto 2
34713941.678482994
Punto 1 y Punto 3
4368296.8742654165
Punto 2 y Punto 3
30345689.906663775
```

De lo que se puede observar, es que los centroides siendo punto 2 y 3 son los más cercanos entre sí,

mientras que los centroides siendo los puntos 1 y 3 son los mas lejanos. Por lo tanto, de nuestros análisis se observa que Northeast y TotalUS son los mas cercanos y que tiene centido ya que pertenecen a la misma región.

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

Estarían muy despegados unos de otros abarcando un rango de valores muy amplios por lo que no podríamos tener información importante sobre el comportamiento de los datos por la amplitud del intervalo de valores.

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

Los centros nos indican la tendencia de relación que tienen unos valores con otros por lo que al verlos podemos saber qué tan parecidos entre sí son los valores que se encuentran próximos a dicho centro, en el caso particular de los datos que graficamos podemos denotar cual es la relación que hubo entre el precio del producto, la cantidad vendida y el año en que se produjo lo cual nos permite apreciar tendencias en el precio según la cosecha y el volumen del producto.

In [ ]: