PARCIAL SEGUNDO CORTE

PRESENTADO POR:

JESUS DAVID SUAREZ PEÑA

PRESENTADO AL DOCENTE:

ALVARO AGUSTIN OÑATE BOWEN

UNIVERSIDAD POPULAR DEL CESAR

FACULTAD DE INGENIERÍA Y TECNOLOGÍAS

INGENIERIA DE SISTEMAS

VALLEDUPAR - CESAR

2020

1. **Tipo de Variable**

```
25 print(datos.dtypes)
```

```
In [3]: runfile('C:/Users/JESUS/.spyder-py3/parcial_regresion.py', wdir='C:/Users/
JESUS/.spyder-py3')
sepal.length    float64
sepal.width     float64
petal.length    float64
petal.width     float64
variety          object
```

El Data set cuenta con 5 variables las cueles 4 son de tipo float y una de tipo string que es la especie

2. **Resumen estadístico del Data Set Iris**

```
23 datos=pd.read_csv('iris.csv')
24 print(datos.describe())
```

```
In [38]: runfile('C:/Users/JESUS/.spyder-py3/parcial_regresion.py', wdir='C:/Users/JESUS/.spyder-py3')
       sepal.length  sepal.width  petal.length  petal.width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.057333      3.758000     1.199333
std        0.828066     0.435866      1.765298     0.762238
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
```

3. El Coeficiente de Correlación, te permite ver con un índice cuánto está de asociadas dos mediciones (dos variables cuantitativas principalmente). El coeficiente va entre -1 y 1.

- **Calcular el Coeficiente de Correlación, Covarianzas de las variables.**

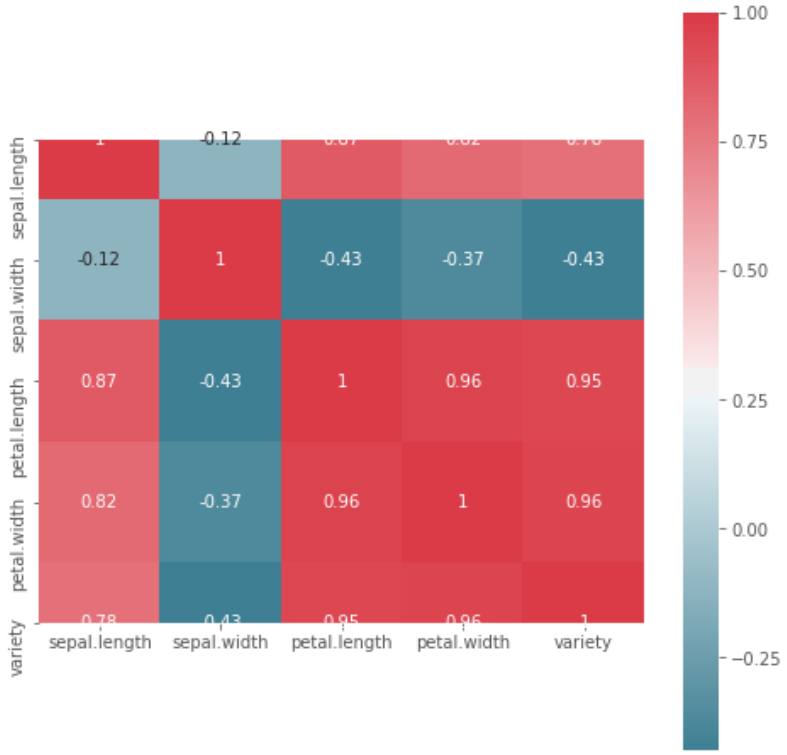
Calculo de la covarianza del Data set

```
25 print(datos.cov())
```

```
In [39]: runfile('C:/Users/JESUS/.spyder-py3/parcial_regresion.py', wdir='C:/Users/JESUS/.spyder-py3')
              sepal.length  sepal.width  petal.length  petal.width
sepal.length      0.685694    -0.042434      1.274315     0.516271
sepal.width      -0.042434     0.189979     -0.329656    -0.121639
petal.length      1.274315    -0.329656      3.116278     1.295609
petal.width       0.516271    -0.121639      1.295609     0.581006
```

Mapa de correlación de Pearson

```
f, ax = plt.subplots(figsize=(8, 8))
corr = datos.corr()
sb.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool),
           cmap=sb.diverging_palette(220, 10, as_cmap=True),
           square=True, annot=True ,ax=ax)
```



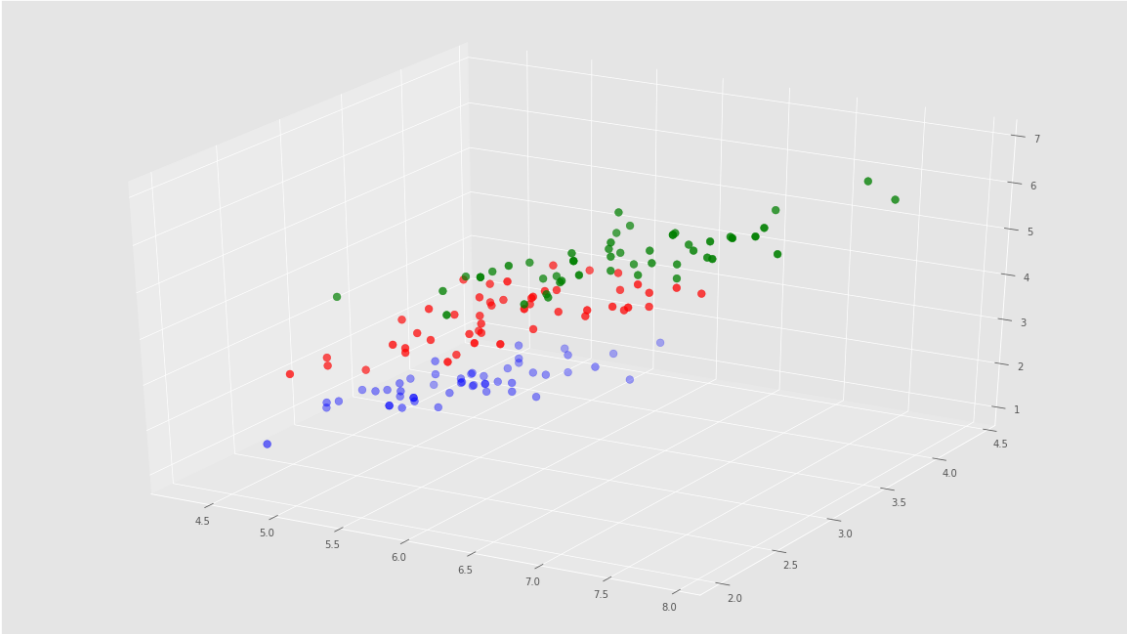
Como se observa en el mapa las variables que tienen la relación más fuerte son las que se acercan a uno.

- Grafica los Diagramas de Dispersión (Coordenadas Cartesianas, en Tres Dimensiones y Coordenadas polares.) de todos los pares de las 4 variables de Iris usando un color y un carácter distinto para cada especie.

Se procede a realizar el diagrama de dispersión en 3D

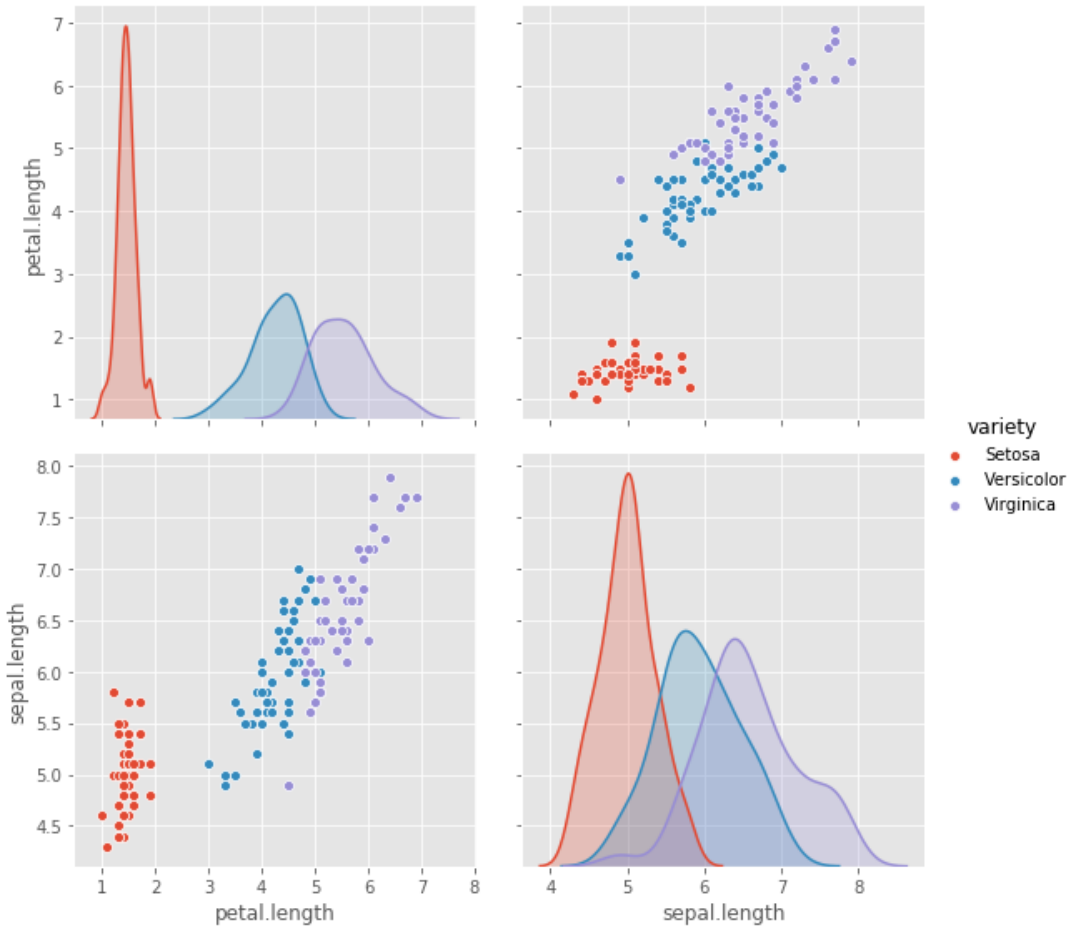
Donde cada color diferente se refiere a la variedad de la flor que corresponde su nombre

```
X = np.array(datos[["sepal.length","sepal.width","petal.length",'petal.width']])
Y = np.array(datos['variety'])
X.shape


colores=['blue','red','green']
asignar=[]
for row in Y:
    asignar.append(colores[row])

fig = plt.figure()
ax = Axes3D(fig)
ax.scatter(X[:, 0], X[:, 1], X[:, 2],X[:,3], c=asignar,s=60)
```



- Graficar relación entre la longitud del pétalo y la longitud del sépalo con las Especies o Grupos

```
sb.pairplot(datos.dropna(), hue='variety',size=4,vars=['petal.length','sepal.length'],kind='scatter')
```



- Ya una vez visto la relación y el comportamiento de las variables del Data set Iris, procedo a realizar el modelo de regresión lineal que permita predecir la Longitud del Pétalo de la Flor basándose en la longitud del Sépalo.

Se procede tomando las variables con la cual se va trabajar el modelo de regresión lineal.

```
59 dataX =datos[["sepal.length"]]
60 X_train = np.array(dataX)
61 y_train = datos['petal.length'].values
```

Se crea el objeto de regresión lineal.

```
64 regr = linear_model.LinearRegression()
```

El siguiente paso es el entrenamiento del modelo.

```
67 regr.fit(X_train, y_train)
```

Luego se prosigue hacer las predicciones del modelo.

```
70 y_pred = regr.predict(X_train)
```

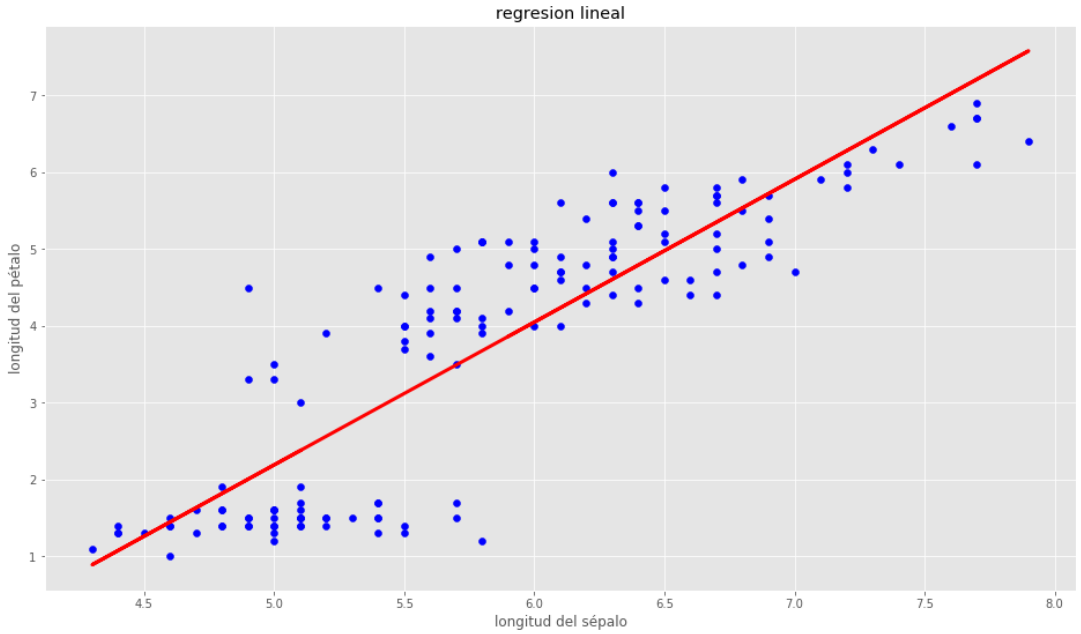
Y para finalizar se realiza la predicción

```
123 y_Dl = regr.predict([[4.7]])
124 print('Prediccion: %.2f '% y_Dl)
```

Predicción: 1.63

De lo siguiente se observa que cada vez que la variable predictora aumenta la variable explicativa aumenta

- Estimación de la línea de regresión



En la siguiente grafica se observa la relación lineal que existe entre la Longitud del Pétalo de la Flor basándose en la longitud del Sépalo, y trazando con una línea roja la línea de predicción del modelo de regresión.

- Resultado del Modelo de Regresión (Coeficientes o Parámetros del Modelo).

```
72 print('Coeficiente  \n', regr.coef_)
73
74 print('término independiente <b>: \n', regr.intercept_)
75
76 print("Error Cuadrado Medior: %.2f" % mean_squared_error(y_train, y_pred))
77
78 print('Puntaje de Varianza: %.2f' % r2_score(y_train, y_pred))
```

```
In [53]: runfile('C:/Users/JESUS/.spyder-py3/parcial_regresion.py', wdir='C:/Users/
JESUS/.spyder-py3')
Coeficiente
 [1.85843298]
término independiente <b>:
 -7.10144336960245
Error Cuadrado Medior: 0.74
Puntaje de Varianza: 0.76
```

En el resultado del modelo se puede observar el coeficiente, el termino independiente b, error cuadrático medio y la puntación de la varianza.

De la ecuación de la recta y = mX + b nuestra pendiente «m» es el coeficiente 1,858 y el término independiente «b» es -7,101

El error cuadrático que arroja el modelo es bajo ya que existe una buena correlación entre las variables escogida

La varianza del modelo de regresión esta en puntaje bueno ya que el mejor puntaje es 1.

- Construir la ecuación lineal de la longitud del pétalo usando los estimados de los coeficientes ββ del intercepto y la longitud del sépalo.

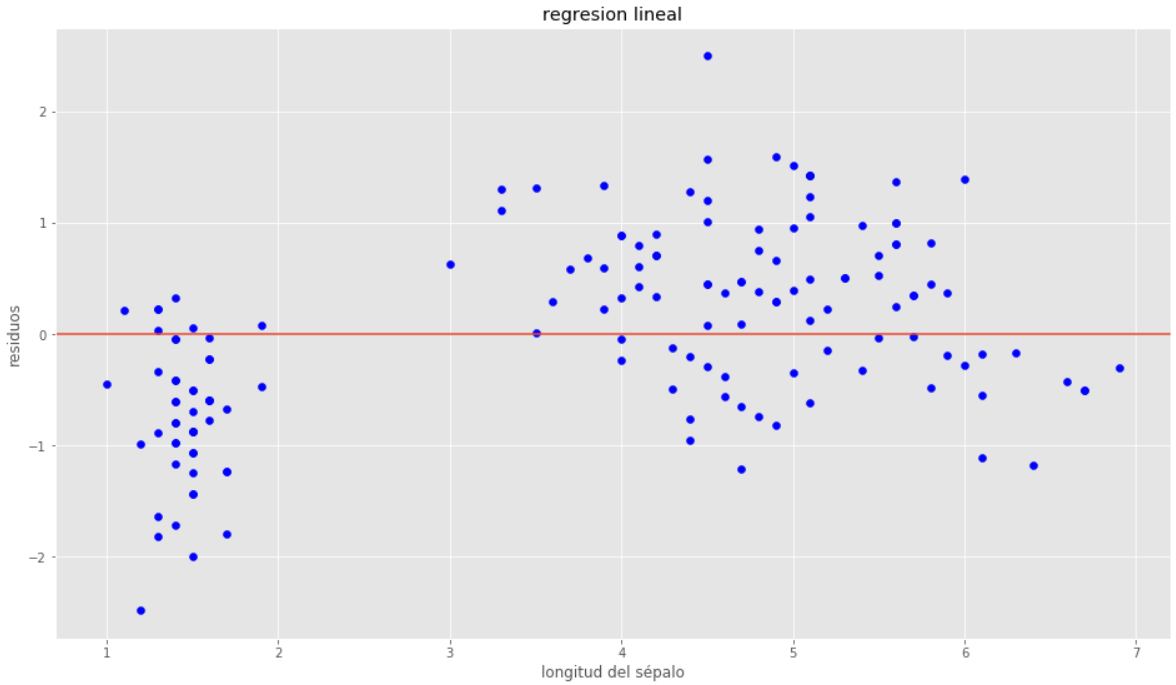
```
82 print("La ecuacion del modelo es igual a ->  ",'y =',regr.coef_, 'x',regr.intercept_ )
```

```
La ecuacion del modelo es igual a ->   y = [1.85843298] x -7.10144336960245
```

4. Se pueden utilizar los residuos para ver si el modelo de regresión lineal es adecuado. La validez del modelo de regresión lineal depende del cumplimiento de 3 condiciones, Asociación lineal entre las variables, Normalidad de los residuales y Variabilidad constante.

- **Asociación lineal entre las Variables**

```
108 x=datos['petal.length']
109 plt.scatter(x,residuos,color='blue')
110 plt.axhline(y=0, xmin=0, xmax=8)
111 plt.title("regresion lineal")
112 plt.xlabel("longitud del sépalo")
113 plt.ylabel("residuos")
114 plt.show()
```

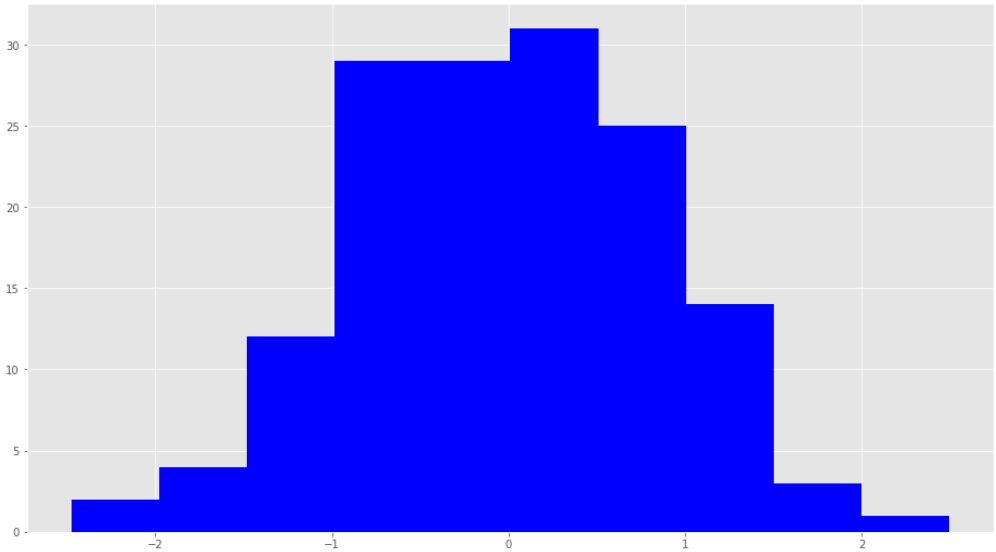


En la gráfica de relación lineal se observa la distribución simetría de los residuos alrededor de la línea roja, donde la relación de la distribución es simétrica negativa ya que hay mas valores del lado negativo de la grafica

- **Normalidad de los Residuales**

Grafica de histograma de los residuos del modelo

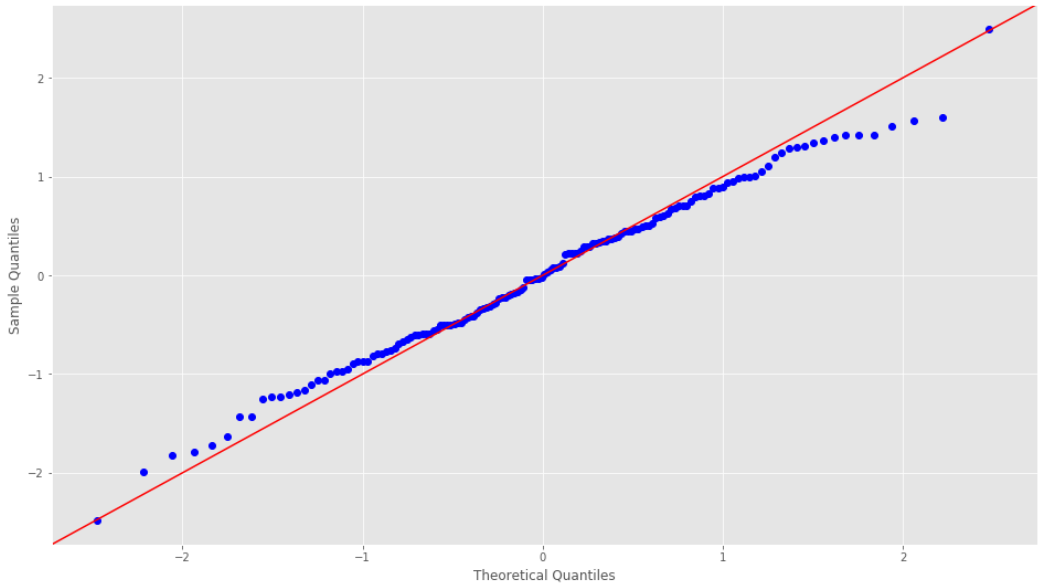
```
100 plt.hist( residuos,color='blue')
101 plt.show()
```



El la grafica de histograma se representa una distribución normal de los residuos del modelo

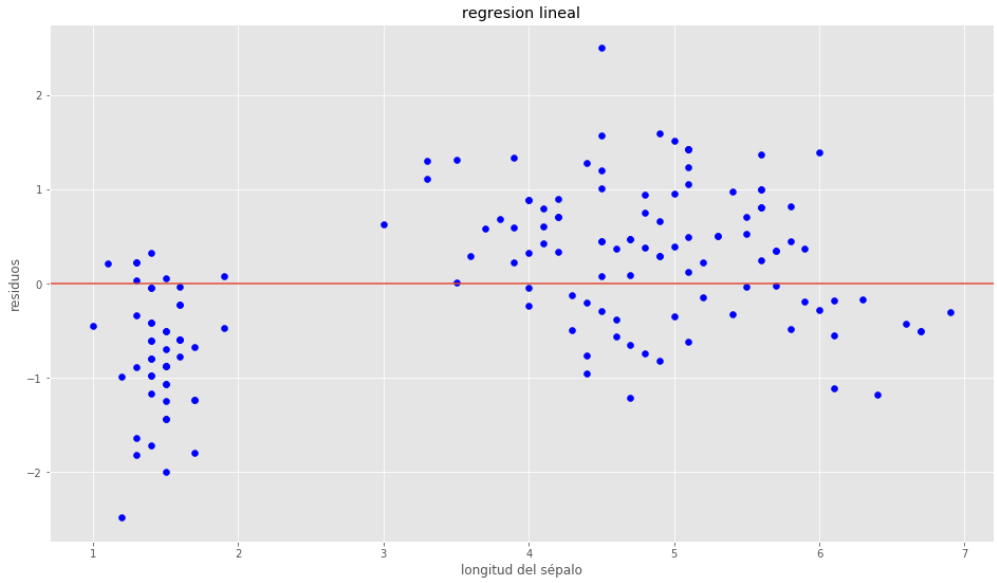
Grafica Q-Q de los residuos del modelo.

```
104 sm.qqplot(residuos, line='45')
105 pylab.show()
```



De la siguiente grafica se puede decir que la distribución de los residuos es normal

- **Variabilidad constante**



CLUSTER

1. **Cargar el conjunto de datos**

```
20 datos=pd.read_csv('iris.csv')
```

Cargamos en la variable datos el data set iris

2. **Convertir las variables nominales a binarias**

```
24 _,idlx = np.unique(datos['variety'],return_inverse=True)
25 datos['variety'] = idlx
```

En este caso se convirtió la variable nominal a numérica para poder trabajar con el algoritmo

Index	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	0
1	4.9	3	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5	3.6	1.4	0.2	0
5	5.4	3.9	1.7	0.4	0
6	4.6	3.4	1.4	0.3	0
7	5	3.4	1.5	0.2	0
8	4.4	2.9	1.4	0.2	0
9	4.9	3.1	1.5	0.1	0
10	5.4	3.7	1.5	0.2	0
11	4.8	3.4	1.6	0.2	0
12	4.8	3	1.4	0.1	0
13	4.3	3	1.1	0.1	0

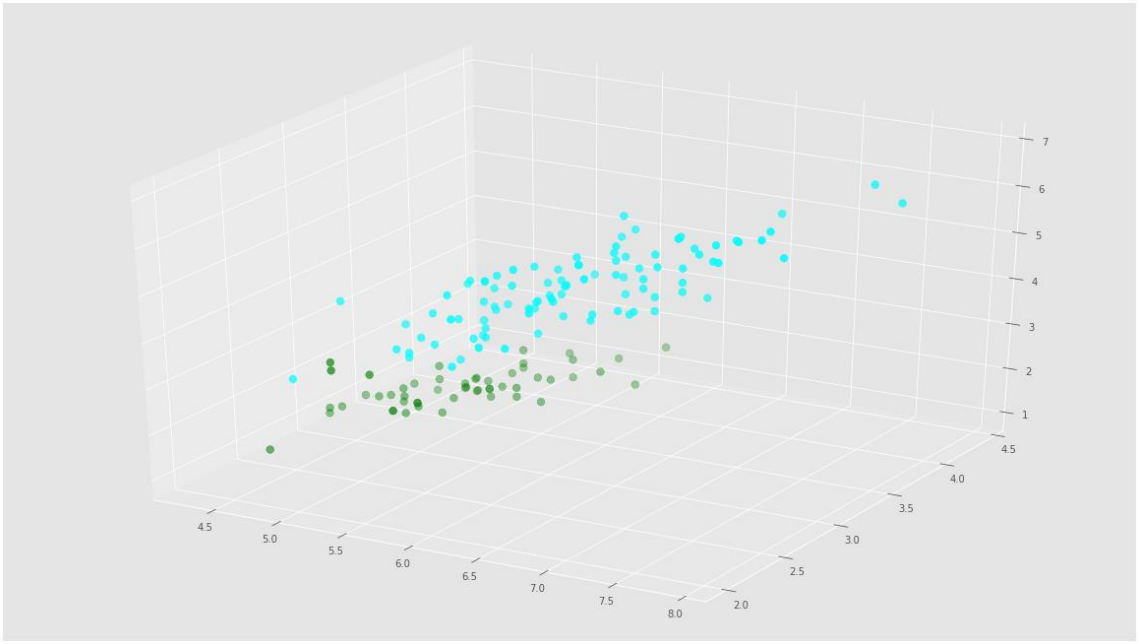
3. **Agrupe el conjunto de datos usando diferentes valores de k=1,2,3,4, 5 …. 14.**

A continuación, se realizarán graficas en 3D para valores diferentes de K

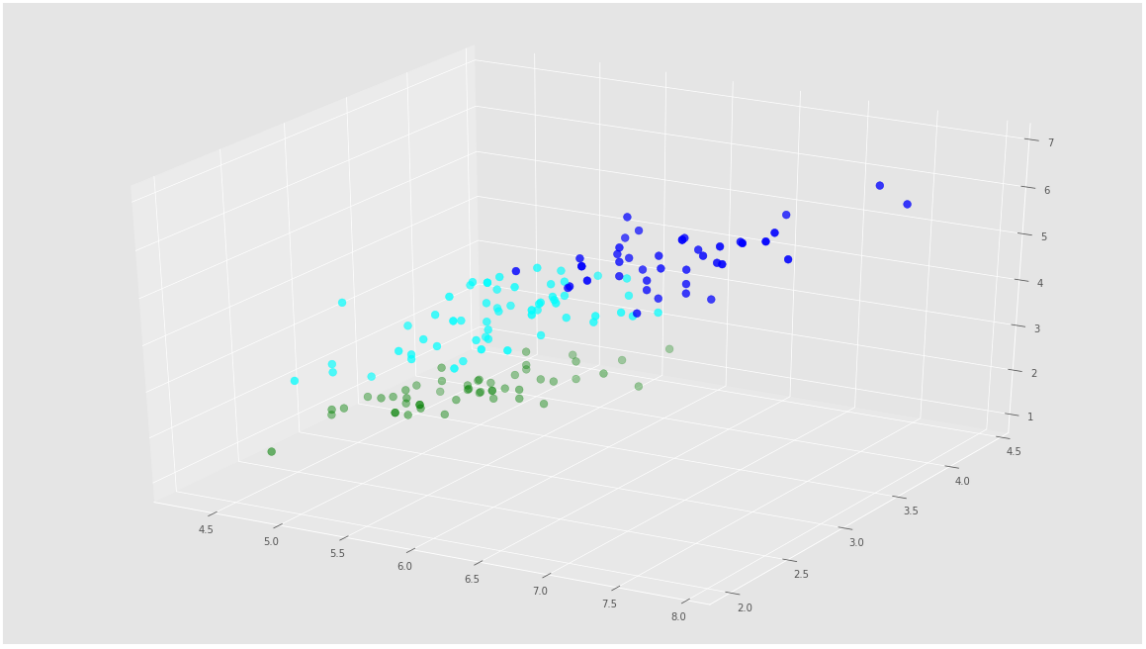
```
44 kmeans = KMeans(n_clusters=3).fit(X)
```

```
50 labels = kmeans.predict(X)
51 C = kmeans.cluster_centers_
52 colores=['green','red','black']
53 asignar=[]
54 for row in labels:
55     asignar.append(colores[row])
56
57 fig = plt.figure()
58 ax = Axes3D(fig)
59 ax.scatter(X[:, 0], X[:, 1], X[:, 2],X[:, 3], c=asignar,s=60)
```

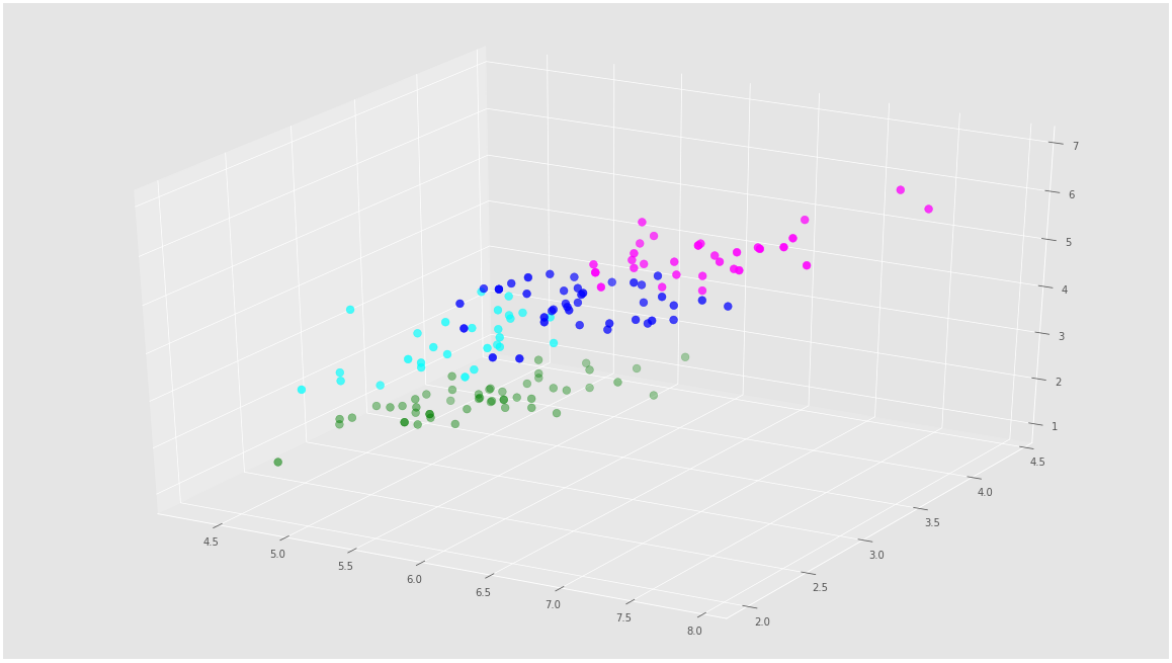
Para un k =2



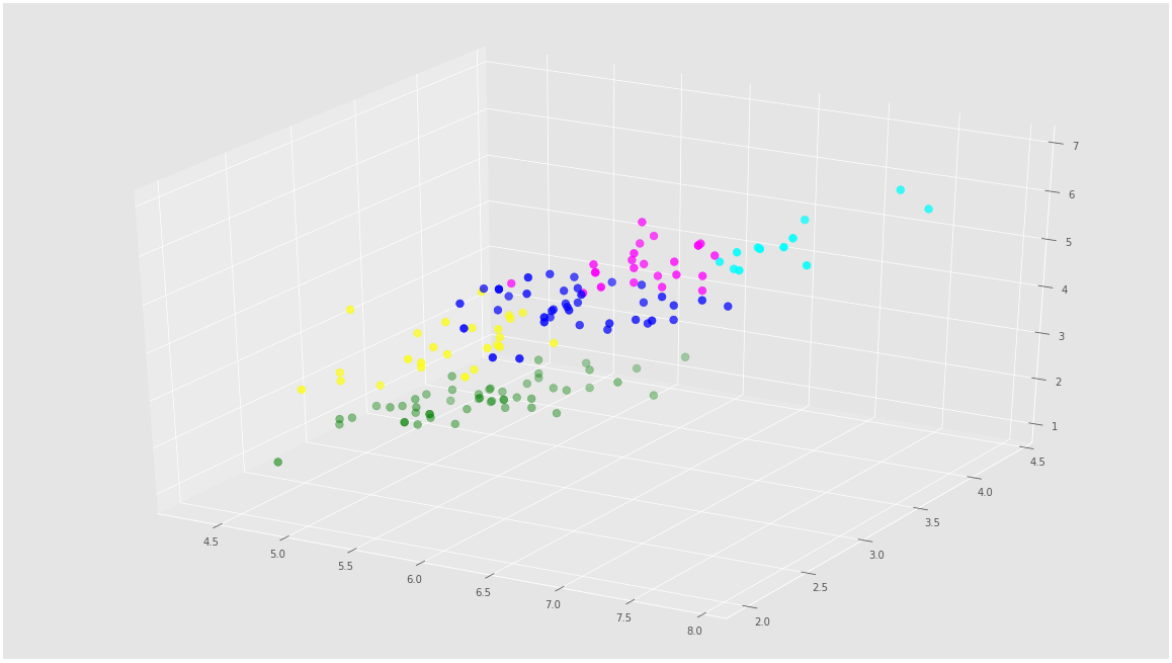
Para un K =3



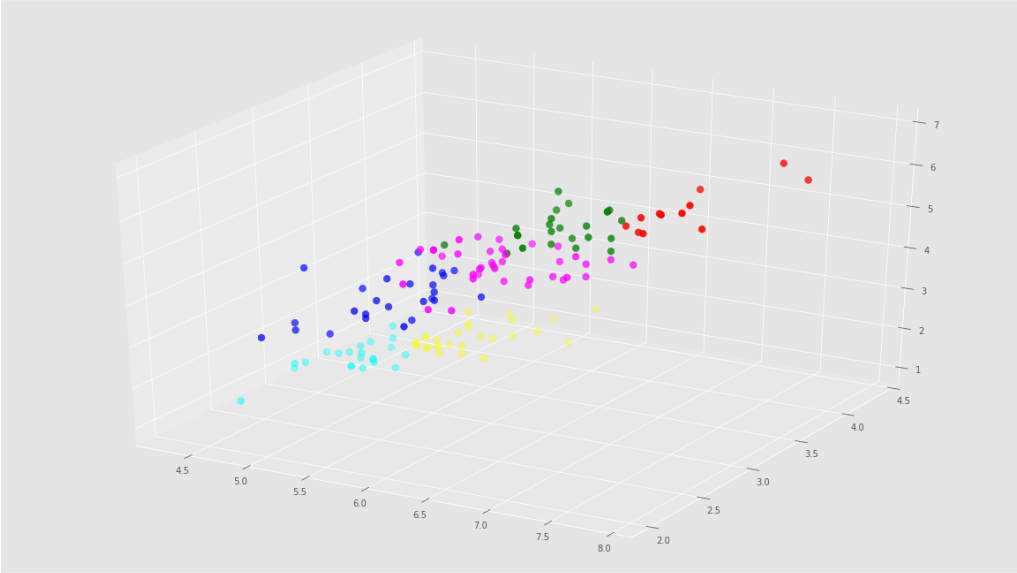
Para un K =4



Para un k =5

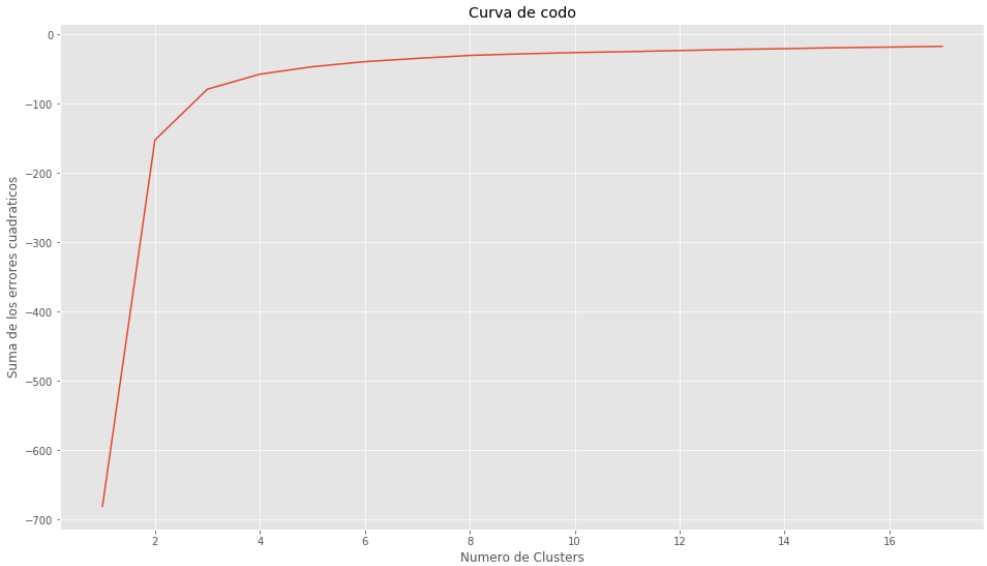


Para un K =6



4. **Grafica de SSE para cada k.**

```
Nc = range(1, 18)
kmeans = [KMeans(n_clusters=i) for i in Nc]
error = [kmeans[i].fit(X).score(X) for i in range(len(kmeans))]


plt.plot(Nc,error)
plt.xlabel('Numero de Clusters')
plt.ylabel('Suma de los errores cuadraticos')
plt.title('Curva de codo')
plt.show()
```



5. **¿Cuál es el mejor valor de k?**

En la representación del la grafica de K se observa el codo de la línea el K optimo para realizar el clúster donde el codo se encuentra en un K=3

```
45 kmeans = KMeans(n_clusters=3).fit(X)
46 centroids = kmeans.cluster_centers_
47 print(centroids)
```

```
[150 rows x 5 columns]
[[6.85       3.07368421 5.74210526 2.07105263]
 [5.006      3.428      1.462      0.246     ]
 [5.9016129  2.7483871  4.39354839 1.43387097]]
```

Esta matriz correspondes a los centroides para un K =3

```
50 labels = kmeans.predict(X)
51
52 C = kmeans.cluster_centers_
53 colores=['green','red','black']
54 asignar=[]
55 for row in labels:
56     asignar.append(colores[row])
57
58 fig = plt.figure()
59 ax = Axes3D(fig)
60 ax.scatter(X[:, 0], X[:, 1], X[:, 2],X[:, 3], c=asignar,s=60)
61 ax.scatter(C[:, 0], C[:, 1], C[:, 2], marker='8', c=colores, s=1000)
```

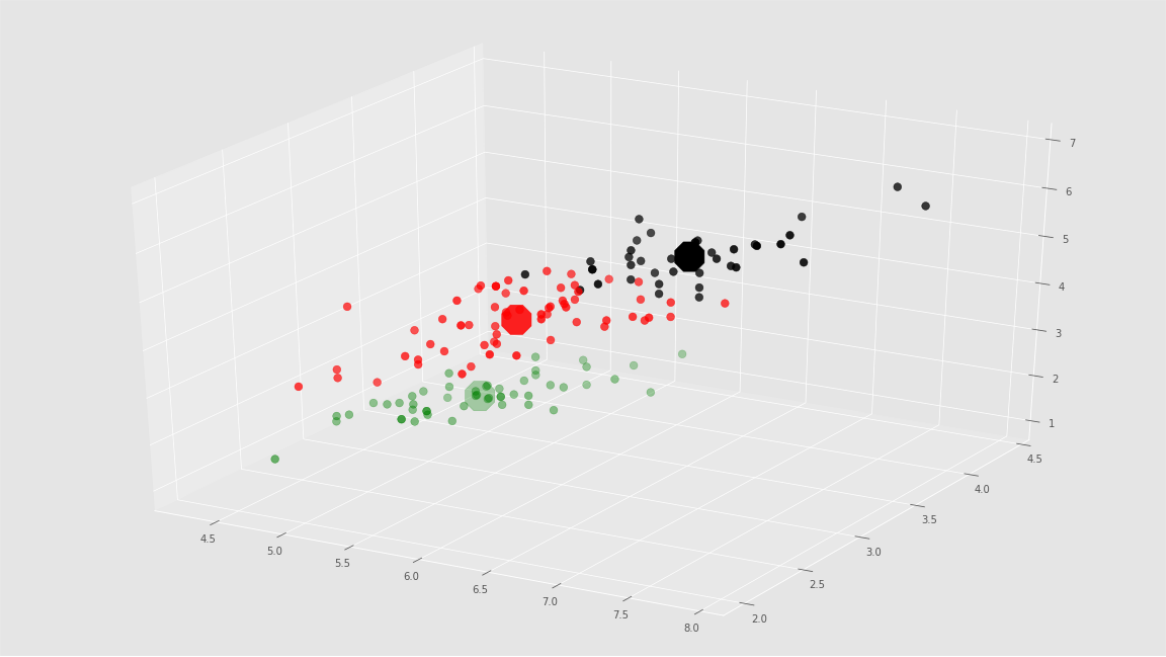


Grafico de clúster para un K =3 con su representación de los centroides para cada grupo representado.

Se observa que la distribución de los datos se encuentra bien agrupados y no están tan separados de sus centroides.