Problem statement:-

Predictive study using the breast cancer diagnostic data set

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
import scipy.cluster.hierarchy as sch
from sklearn.cluster import AgglomerativeClustering
import warnings
warnings.simplefilter(action='ignore')
```

In [2]:
```
df = pd.read_csv(r"C:\Users\HP\OneDrive\Documents\BreastCancerPrediction.csv")
df
```

Out[2]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800
...	...	...	...	...	...	...	...	...	...
564	926424	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400
566	926954	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140
568	92751	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000

569 rows × 33 columns

In [3]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   id                       569 non-null    int64  
 1   diagnosis                569 non-null    object 
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
 32  Unnamed: 32              0 non-null      float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```

In [4]:
```
1 df.drop('Unnamed: 32',axis=1,inplace=True)
```

In [5]:
```
1 df.info()
```

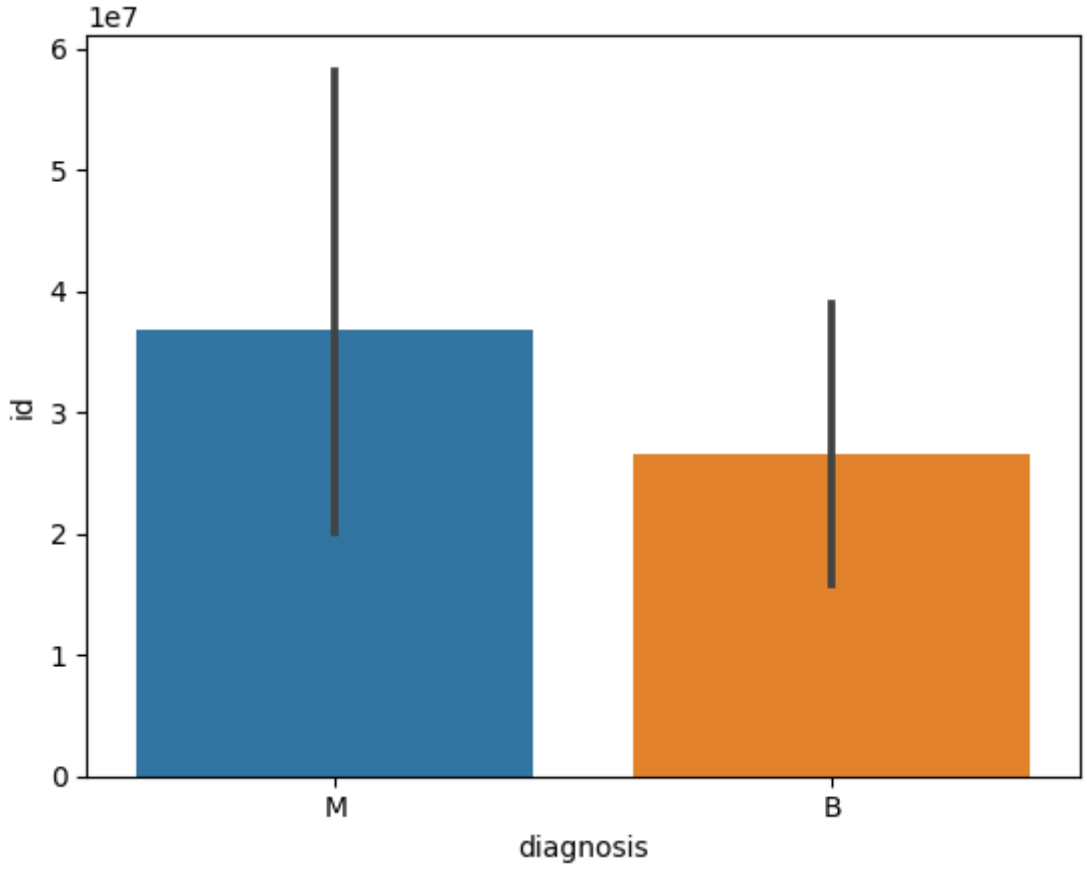
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   id                       569 non-null    int64  
 1   diagnosis                569 non-null    object 
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
dtypes: float64(30), int64(1), object(1)
memory usage: 142.4+ KB
```

In [6]:
```
1 df['diagnosis'].value_counts()
```

Out[6]:
```
diagnosis
B    357
M    212
Name: count, dtype: int64
```

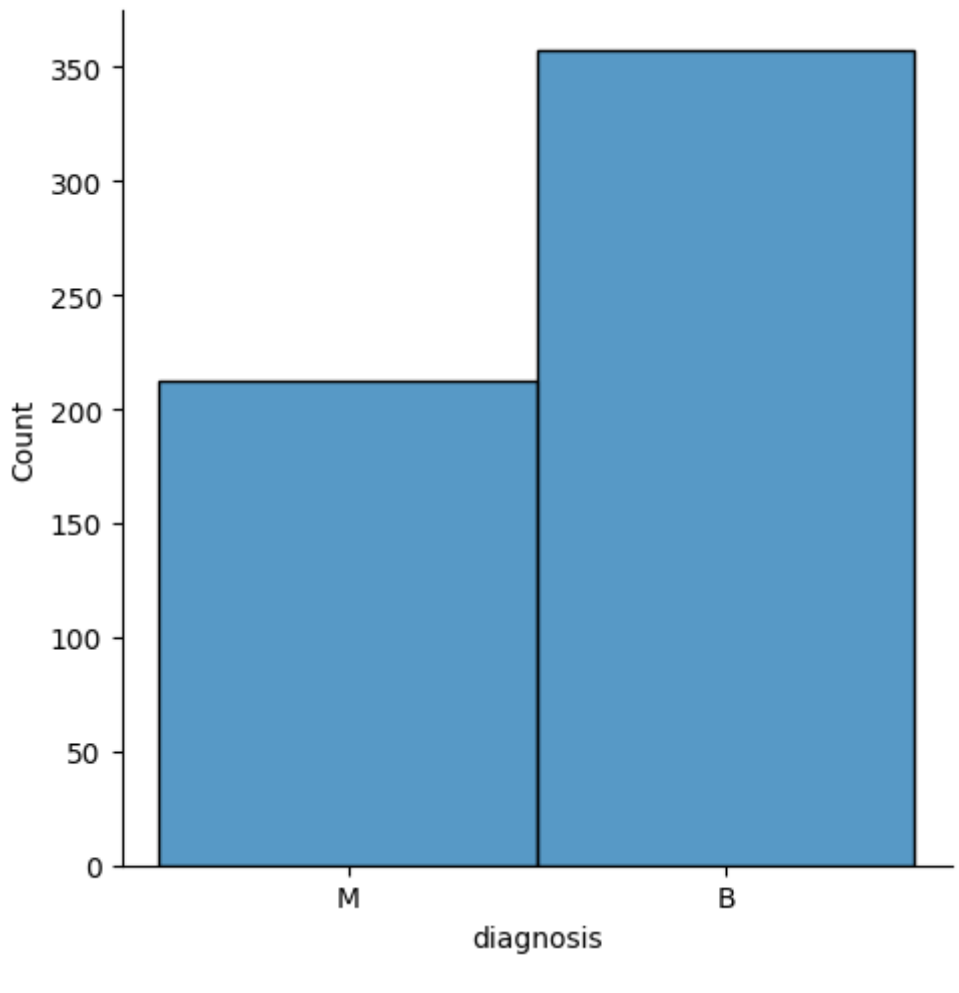
In [7]:
```
1 sns.barplot(x='diagnosis',y='id',data=df)
```

Out[7]:
```
<Axes: xlabel='diagnosis', ylabel='id'>
```



In [8]:
```
1 sns.displot(df['diagnosis'])
```

Out[8]: <seaborn.axisgrid.FacetGrid at 0x197d05756c0>



In [9]:
```
1 s={'diagnosis':{'B':1,'M':2}}
2 df = df.replace(s)
3 df
```

Out[9]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean
0	842302	2	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010
1	842517	2	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690
2	84300903	2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740
3	84348301	2	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140
4	84358402	2	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800
...	...	...	...	...	...	...	...	...	...
564	926424	2	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390
565	926682	2	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400
566	926954	2	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251
567	927241	2	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140
568	92751	1	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000

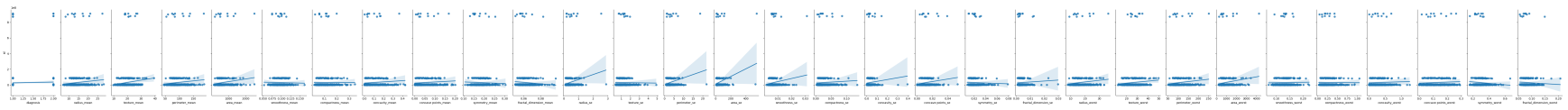
569 rows × 32 columns

In [10]:
```
1 df.columns
```

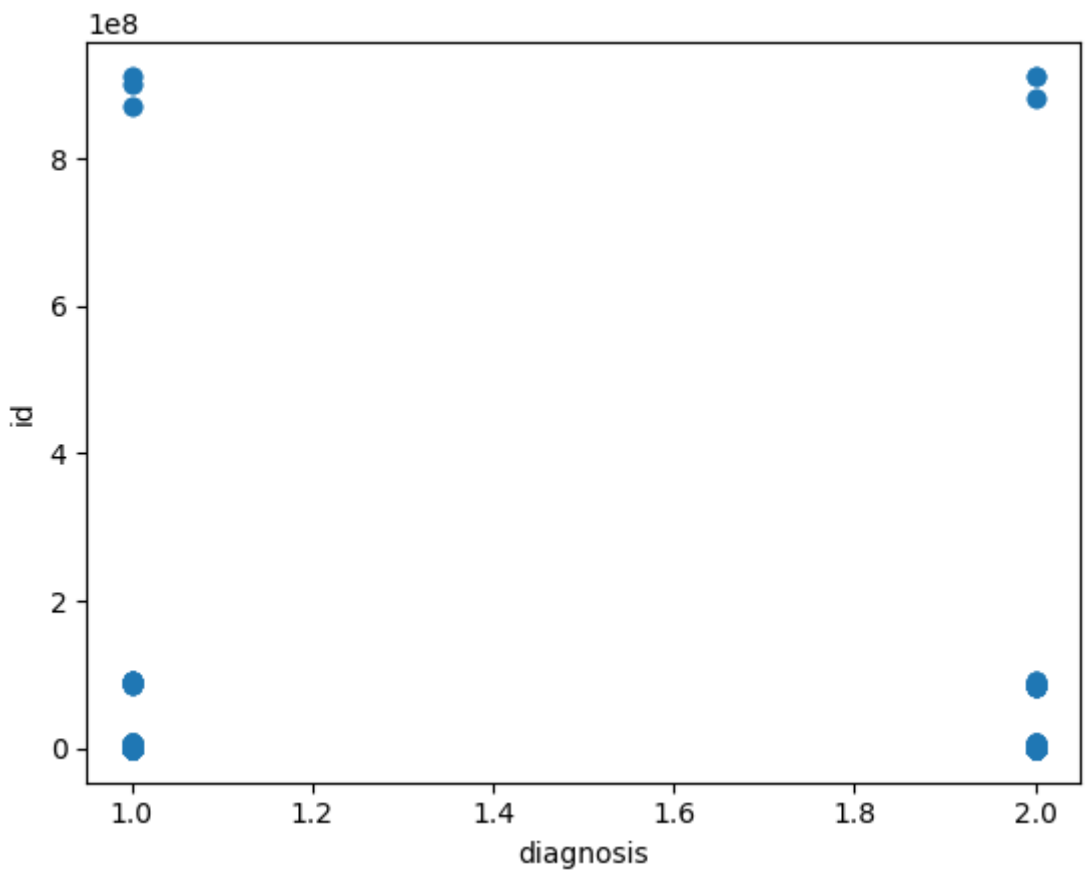
Out[10]:
```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst'],
      dtype='object')
```

In [11]:
```
import matplotlib.pyplot as plt
import seaborn as sns
sns.pairplot(df,x_vars=['diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst'],y_vars=['id'],height=5,aspect=0.5,kind='reg')
```

Out[11]: <seaborn.axisgrid.PairGrid at 0x197dd619360>



In [12]:
```
plt.scatter(df['diagnosis'],df['id'])
plt.xlabel("diagnosis")
plt.ylabel("id")
plt.show()
```



In [13]:
```
from sklearn.cluster import KMeans
km = KMeans()
km
```

Out[13]:
KMeans
KMeans()

In [14]:
```
y_predicted = km.fit_predict(df[['diagnosis','id']])
y_predicted
```

Out[14]:
```
array([0, 0, 6, 6, 6, 0, 0, 6, 0, 6, 0, 6, 0, 0, 6, 6, 0, 6, 0, 3, 3, 3,
       3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       6, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0,
       0, 0, 0, 0, 0, 0, 0, 3, 3, 3, 3, 3, 3, 0, 3, 3, 3, 3, 3, 6, 6,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 6, 0, 4, 4, 3, 0,
       3, 3, 3, 3, 0, 0, 3, 3, 0, 3, 3, 3, 0, 3, 3, 3, 6, 0, 0, 0, 0, 0,
       0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 3, 7, 3, 3, 7, 3, 3, 3, 3, 3, 2,
       3, 3, 3, 3, 3, 2, 2, 2, 2, 0, 0, 2, 2, 2, 0, 2, 2, 0, 0, 2, 2, 0,
       0, 2, 0, 0, 0, 0, 2, 0, 0, 2, 0, 3, 0, 0, 2, 0, 0, 2, 0, 0, 0, 0,
       0, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 0, 3, 3, 3,
       3, 0, 3, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0,
       0, 0, 0, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 0, 0, 3, 0,
       0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0,
       0, 3, 0, 3, 3, 0, 3, 1, 1, 0, 3, 3, 3, 0, 3, 3, 3, 3, 3, 3, 3, 0,
       3, 0, 0, 3, 3, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0,
       2, 2, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 2,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 2, 0, 3, 3, 3, 3, 0, 5, 3, 3, 3, 0, 0, 3, 3, 3, 3, 3, 5, 5,
       3, 5, 5, 3, 3, 3, 3, 0, 3, 3, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       2, 2, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 2, 2,
       0, 0, 0, 2, 2, 2, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

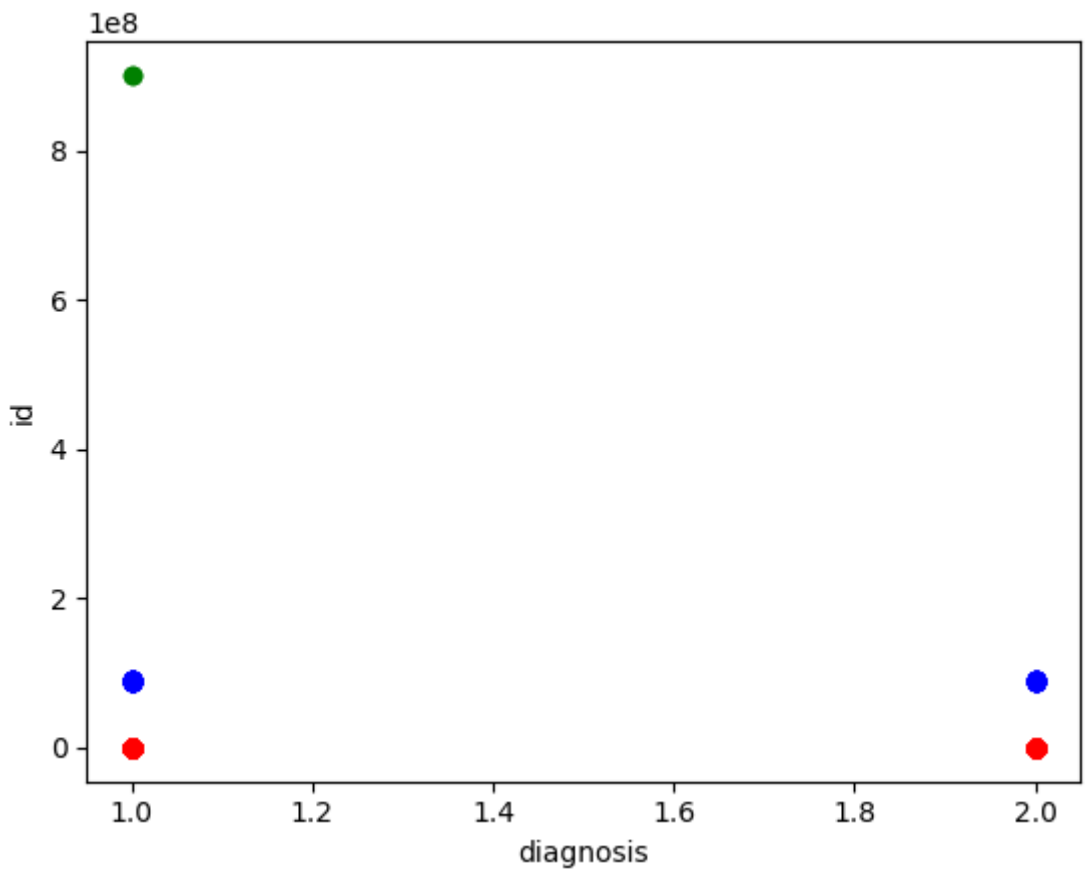
In [15]:
```
df['Cluster']=y_predicted
df.head()
```

Out[15]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	p
0	842302	2	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	
1	842517	2	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	
2	84300903	2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	
3	84348301	2	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	
4	84358402	2	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	

5 rows × 33 columns

In [16]:
```
df1 = df[df.Cluster == 0]
df2 = df[df.Cluster == 1]
df3 = df[df.Cluster == 2]
plt.scatter(df1['diagnosis'],df1['id'],color="red")
plt.scatter(df2['diagnosis'],df2['id'],color="green")
plt.scatter(df3['diagnosis'],df3['id'],color="blue")
plt.xlabel('diagnosis')
plt.ylabel('id')
plt.show()
```



In [17]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [18]:
```
scaler=MinMaxScaler()
```

In [19]:
```
scaler.fit(df[["id"]])
df["id"]=scaler.transform(df[["id"]])
df.head()
```

Out[19]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	p
0	0.000915	2	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	
1	0.000915	2	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	
2	0.092495	2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	
3	0.092547	2	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	
4	0.092559	2	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	

5 rows × 33 columns

In [20]:
```
scaler.fit(df[["diagnosis"]])
df["diagnosis"]=scaler.transform(df[["diagnosis"]])
df.head()
```

Out[20]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	p
0	0.000915	1.0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	
1	0.000915	1.0	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	
2	0.092495	1.0	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	
3	0.092547	1.0	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	
4	0.092559	1.0	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	

5 rows × 33 columns

In [21]:
```
km=KMeans()
```

In [22]:
```
y_predicted = km.fit_predict(df[["id","diagnosis"]])
y_predicted
```

Out[22]:
```
array([0, 0, 5, 5, 5, 0, 0, 5, 0, 5, 0, 5, 0, 0, 5, 5, 0, 5, 0, 6, 6, 6,
       7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       5, 0, 4, 0, 1, 1, 1, 1, 1, 0, 0, 4, 0, 0, 1, 1, 1, 1, 0, 1, 5, 0,
       1, 1, 1, 1, 0, 1, 0, 0, 6, 7, 6, 7, 7, 6, 1, 6, 7, 7, 6, 7, 5, 5,
       1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
       5, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 4, 4, 1, 2, 2, 6, 1,
       6, 6, 7, 6, 1, 1, 6, 7, 0, 6, 7, 6, 1, 7, 7, 6, 4, 0, 0, 1, 1, 1,
       1, 5, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 5, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 6, 6, 3, 6, 7, 3, 7, 7, 6, 6, 7, 5,
       6, 6, 6, 7, 6, 4, 4, 4, 4, 0, 0, 4, 4, 5, 1, 4, 5, 0, 1, 5, 4, 1,
       1, 4, 0, 1, 1, 1, 4, 1, 0, 4, 0, 7, 0, 0, 5, 0, 0, 5, 0, 0, 0, 0,
       0, 5, 6, 6, 6, 6, 6, 6, 7, 6, 7, 6, 6, 7, 6, 6, 7, 6, 0, 7, 6, 6,
       6, 1, 6, 4, 4, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 5, 1, 1, 1, 1, 1,
       1, 1, 1, 4, 4, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 4, 4, 1, 1, 7, 0,
       0, 1, 1, 1, 1, 5, 1, 0, 1, 0, 1, 1, 1, 0, 4, 1, 1, 1, 1, 1, 1, 0,
       0, 7, 1, 6, 6, 1, 6, 2, 2, 1, 6, 6, 6, 0, 7, 6, 7, 7, 7, 6, 7, 0,
       6, 1, 1, 6, 6, 7, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 4, 1, 0, 0, 1, 1,
       4, 4, 1, 1, 5, 1, 1, 1, 1, 1, 1, 1, 5, 1, 1, 1, 1, 1, 0, 1, 1, 5,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 4, 4, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,
       1, 0, 4, 1, 7, 6, 7, 6, 1, 3, 6, 7, 6, 1, 1, 6, 6, 6, 6, 6, 3, 3,
       6, 2, 2, 6, 6, 6, 7, 1, 6, 6, 1, 6, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 0, 4, 4, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
       4, 4, 1, 0, 1, 1, 0, 1, 5, 1, 0, 0, 1, 1, 1, 5, 1, 1, 1, 1, 4, 4,
       1, 1, 1, 4, 4, 5, 1, 0, 5, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1])
```

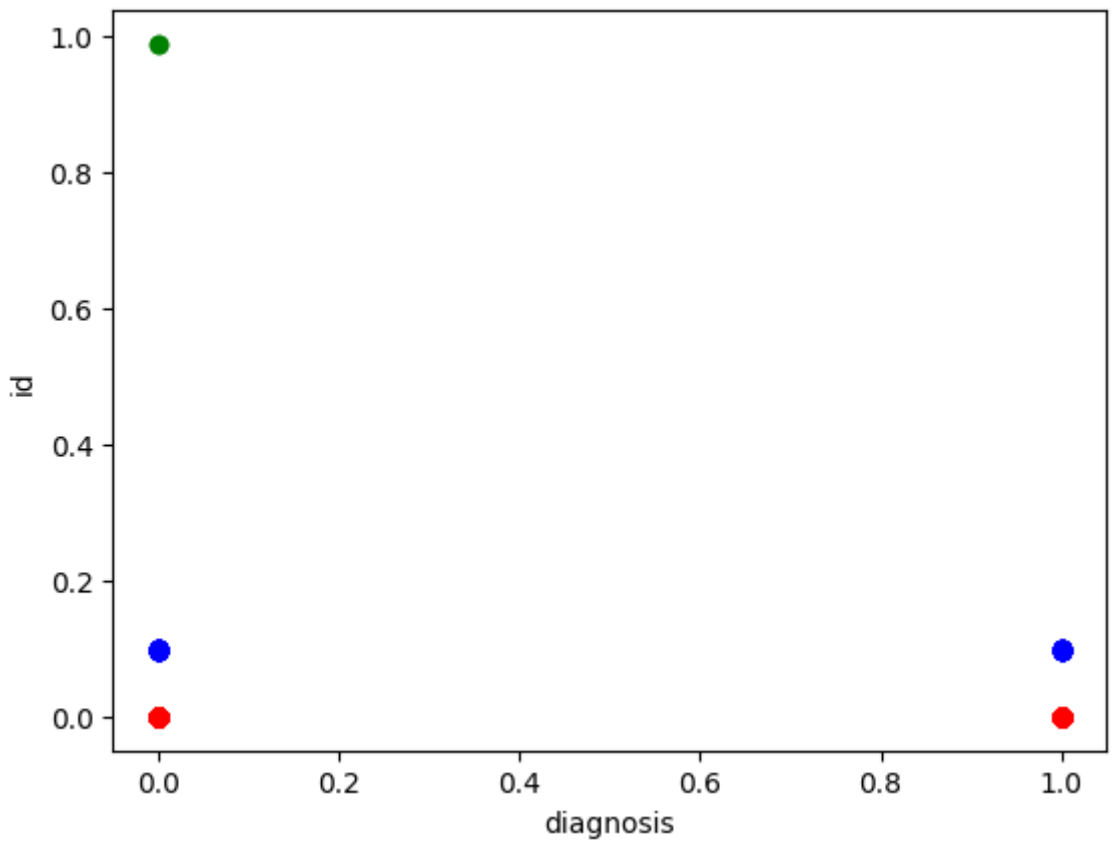
In [23]:
```
df['New Cluster'] =y_predicted
df.head()
```

Out[23]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	p
0	0.000915	1.0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	
1	0.000915	1.0	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	
2	0.092495	1.0	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	
3	0.092547	1.0	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	
4	0.092559	1.0	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	

5 rows × 34 columns

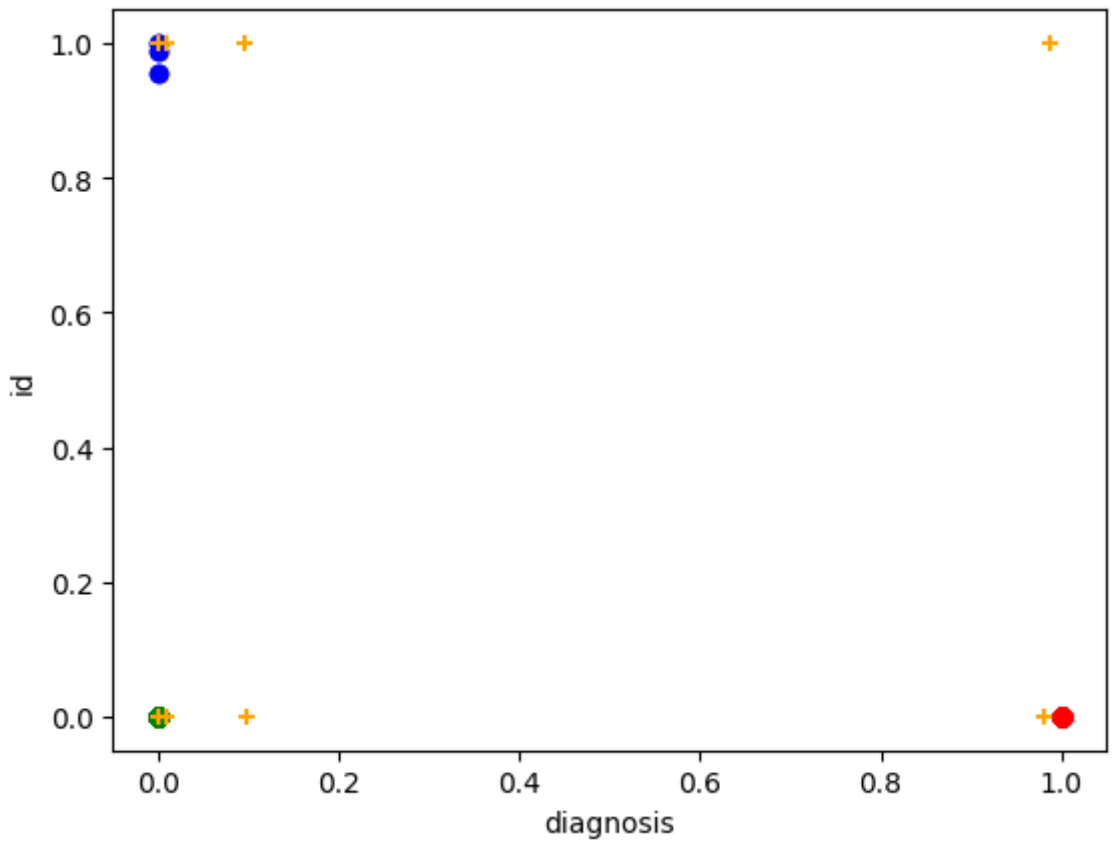
In [24]:
```
df1 = df[df.Cluster == 0]
df2 = df[df.Cluster == 1]
df3 = df[df.Cluster == 2]
plt.scatter(df1['diagnosis'],df1['id'],color="red")
plt.scatter(df2['diagnosis'],df2['id'],color="green")
plt.scatter(df3['diagnosis'],df3['id'],color="blue")
plt.xlabel('diagnosis')
plt.ylabel('id')
plt.show()
```



In [25]:
```
km.cluster_centers_
```

Out[25]:
```
array([[ 8.66630181e-04,  1.00000000e+00],
       [ 8.73559903e-04, -3.88578059e-16],
       [ 9.81489980e-01,  0.00000000e+00],
       [ 9.86676020e-01,  1.00000000e+00],
       [ 9.80482633e-02,  2.22044605e-16],
       [ 9.60508811e-02,  1.00000000e+00],
       [ 9.75498263e-03,  3.33066907e-16],
       [ 9.70020395e-03,  1.00000000e+00]])
```

In [26]:
```
df1 = df[df["New Cluster"]==0]
df2 = df[df["New Cluster"]==1]
df3 = df[df["New Cluster"]==2]
plt.scatter(df1["diagnosis"],df1["id"],color="red")
plt.scatter(df2["diagnosis"],df2["id"],color="green")
plt.scatter(df3["diagnosis"],df3["id"],color="blue")
plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color="orange",marker = "+")
plt.xlabel("diagnosis")
plt.ylabel("id")
plt.show()
```

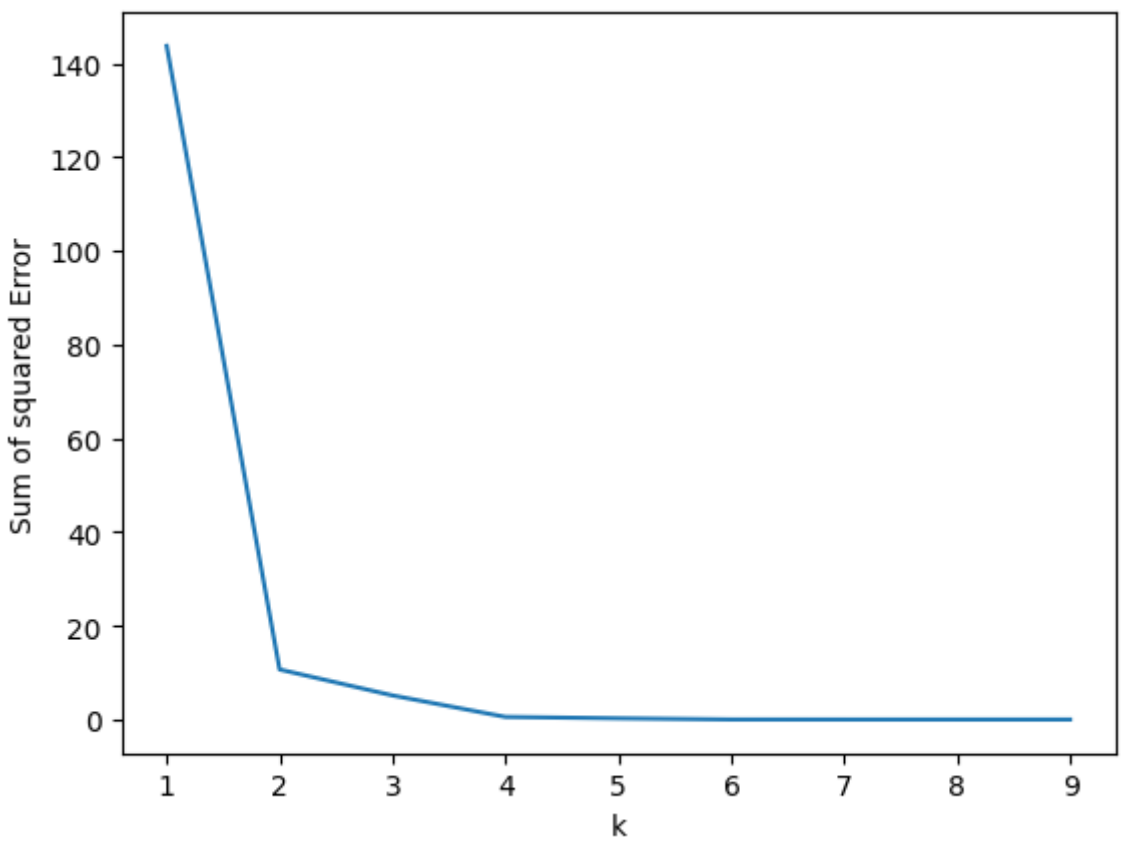


In [27]:
```
k_rng = range(1,10)
sse = []
for k in k_rng:
    km = KMeans(n_clusters = k)
    km.fit(df[["diagnosis","id"]])
    sse.append(km.inertia_)
sse
```

Out[27]:
```
[143.70229416919855,
 10.673085295041425,
 5.137978657842326,
 0.5525613909924392,
 0.25225372036782046,
 0.010231495775075754,
 0.006029269300985468,
 0.0038475118077155816,
 0.0018609816860278147]
```

In [28]:
```
plt.plot(k_rng,sse)
plt.xlabel("k")
plt.ylabel("Sum of squared Error")
```

Out[28]: Text(0, 0.5, 'Sum of squared Error')



Conclusion:-

M:malignant- cansor cells will spred over other body parts
B:benign- cansor but not dangerous
B is more but not Harmful but M is less but Harmful

In [ ]:
```

```