PROJECT

PROBLEM STATEMENT : Predictive study using the breast cancer diagnostic data set

Importing the libraries ¶

In [38]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

Reading the data

In [2]:

```
df=pd.read_csv(r"C:\Users\Lenovo\OneDrive\Desktop\Data Sets\BreastCancerPredict
df
```

Out[2]:

```
           id  diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  smoothness_

  0    842302          M        17.99         10.38          122.80     1001.0          0.
  1    842517          M        20.57         17.77          132.90     1326.0          0.(
  2  84300903          M        19.69         21.25          130.00     1203.0          0.‘
  3  84348301          M        11.42         20.38           77.58      386.1          0.‘
  4  84358402          M        20.29         14.34          135.10     1297.0          0.‘
...       ...        ...          ...           ...             ...        ...
564    926424          M        21.56         22.39          142.00     1479.0          0.
565    926682          M        20.13         28.25          131.20     1261.0          0.(
566    926954          M        16.60         28.08          108.30      858.1          0.(
567    927241          M        20.60         29.33          140.10     1265.0          0.
568     92751          B         7.76         24.54           47.92      181.0          0.(

569 rows × 33 columns
```

Data cleaning and preprocessing

In [4]:
```
df.columns
```

Out[4]:
```
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', 'Unnamed: 32'],
      dtype='object')
```

In [32]:
```
df.shape
```

Out[32]:
```
(569, 35)
```

In [33]:
```
df.head()
```

Out[33]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_me
0	842302	M	17.99	10.38	122.80	1001.0	0.118
1	842517	M	20.57	17.77	132.90	1326.0	0.084
2	84300903	M	19.69	21.25	130.00	1203.0	0.109
3	84348301	M	11.42	20.38	77.58	386.1	0.142
4	84358402	M	20.29	14.34	135.10	1297.0	0.100

5 rows × 35 columns

In [34]:
```
df.tail()
```

Out[34]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_me
564	926424	M	21.56	22.39	142.00	1479.0	0.111
565	926682	M	20.13	28.25	131.20	1261.0	0.097
566	926954	M	16.60	28.08	108.30	858.1	0.084
567	927241	M	20.60	29.33	140.10	1265.0	0.117
568	92751	B	7.76	24.54	47.92	181.0	0.052

5 rows × 35 columns

In [35]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 35 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
```

In [36]:
```
df.describe()
```

Out[36]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mea
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.00000
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.09636
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.01406
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.05263
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.08637
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.09587
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.10530
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.16340

8 rows × 34 columns

In [5]:

```
df.isnull().sum()
```

Out[5]:

```
id                           0
diagnosis                    0
radius_mean                  0
texture_mean                 0
perimeter_mean               0
area_mean                    0
smoothness_mean              0
compactness_mean             0
concavity_mean               0
concave points_mean          0
symmetry_mean                0
fractal_dimension_mean       0
radius_se                    0
texture_se                   0
perimeter_se                 0
area_se                      0
smoothness_se                0
compactness_se               0
concavity_se                 0
concave points_se            0
symmetry_se                  0
fractal_dimension_se         0
radius_worst                 0
texture_worst                0
perimeter_worst              0
area_worst                   0
smoothness_worst             0
compactness_worst            0
concavity_worst              0
concave points_worst         0
symmetry_worst               0
fractal_dimension_worst      0
Unnamed: 32                569
dtype: int64
```

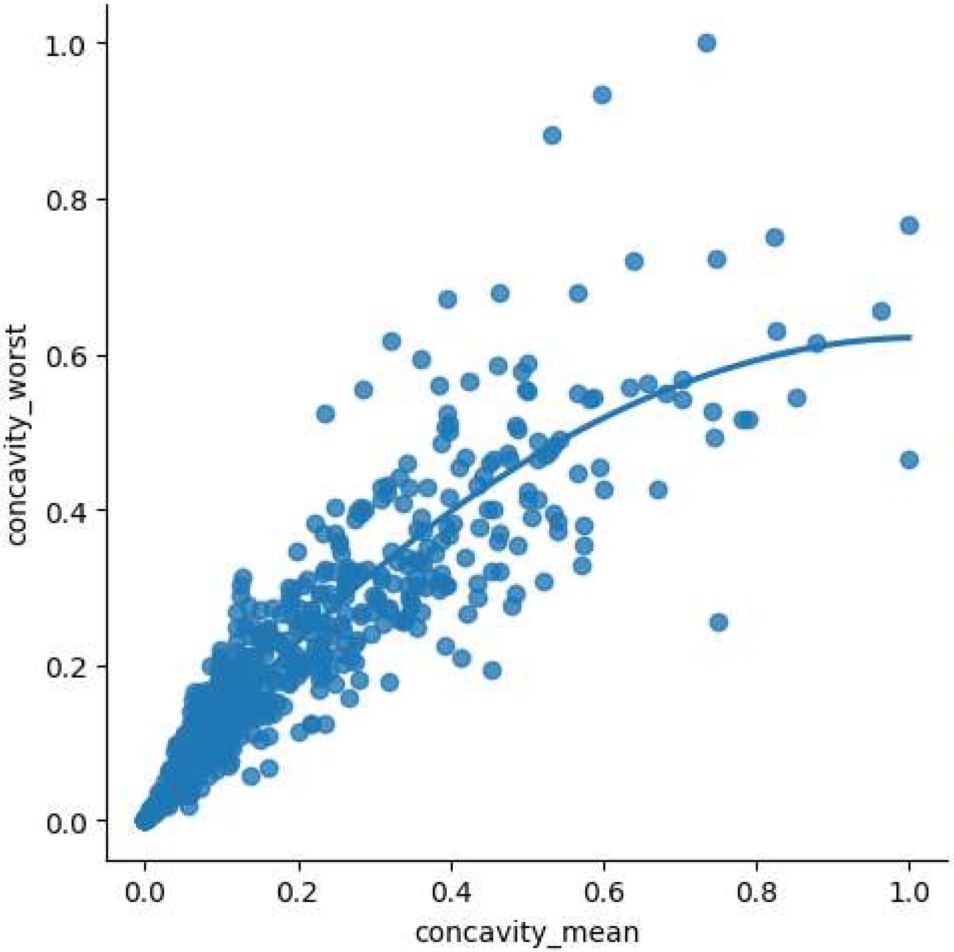
In [9]:

```
df.fillna(method="ffill",inplace=True)
```

Data Visualization

In [39]:
```
sns.lmplot(x="concavity_mean",y="concavity_worst",data=df,order=2,ci=None)
```

Out[39]:
```
<seaborn.axisgrid.FacetGrid at 0x2bab5cba850>
```



Applying Linear Regression

In [40]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

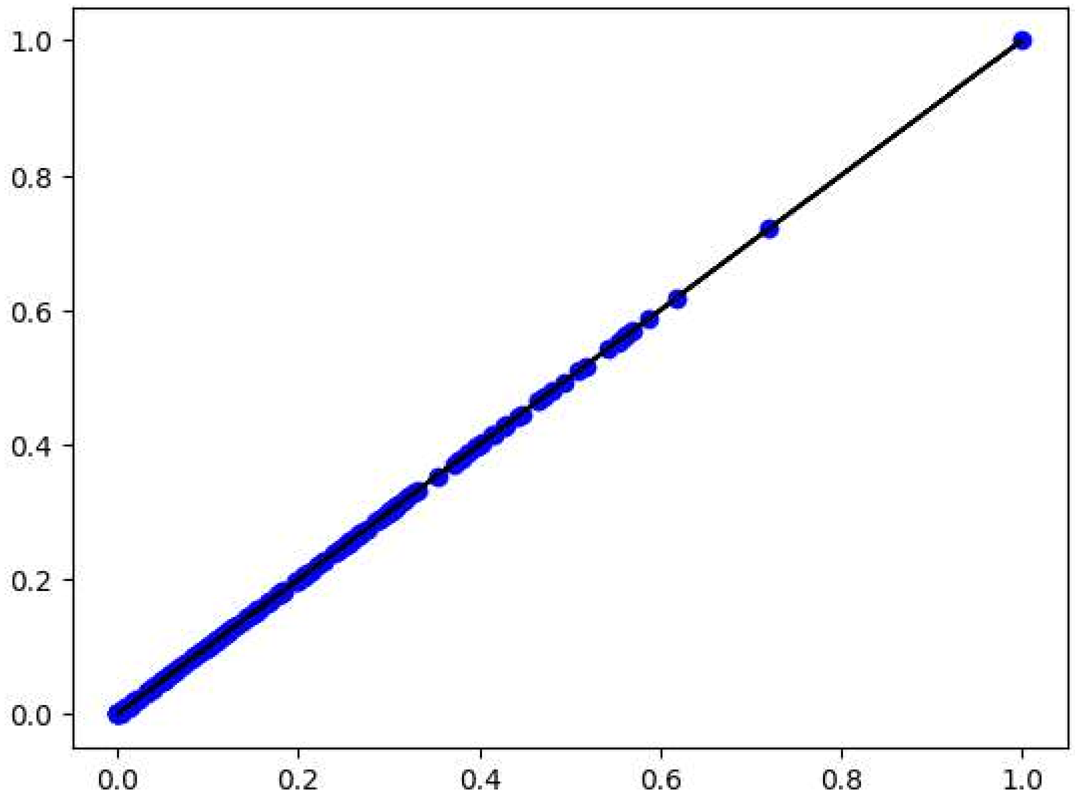
In [43]:
```
x=np.array(df['concavity_mean']).reshape(-1,1)
y=x=np.array(df['concavity_worst']).reshape(-1,1)
```

In [44]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30)
```

In [45]:
```
regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_train,y_train))
```

```
1.0
```

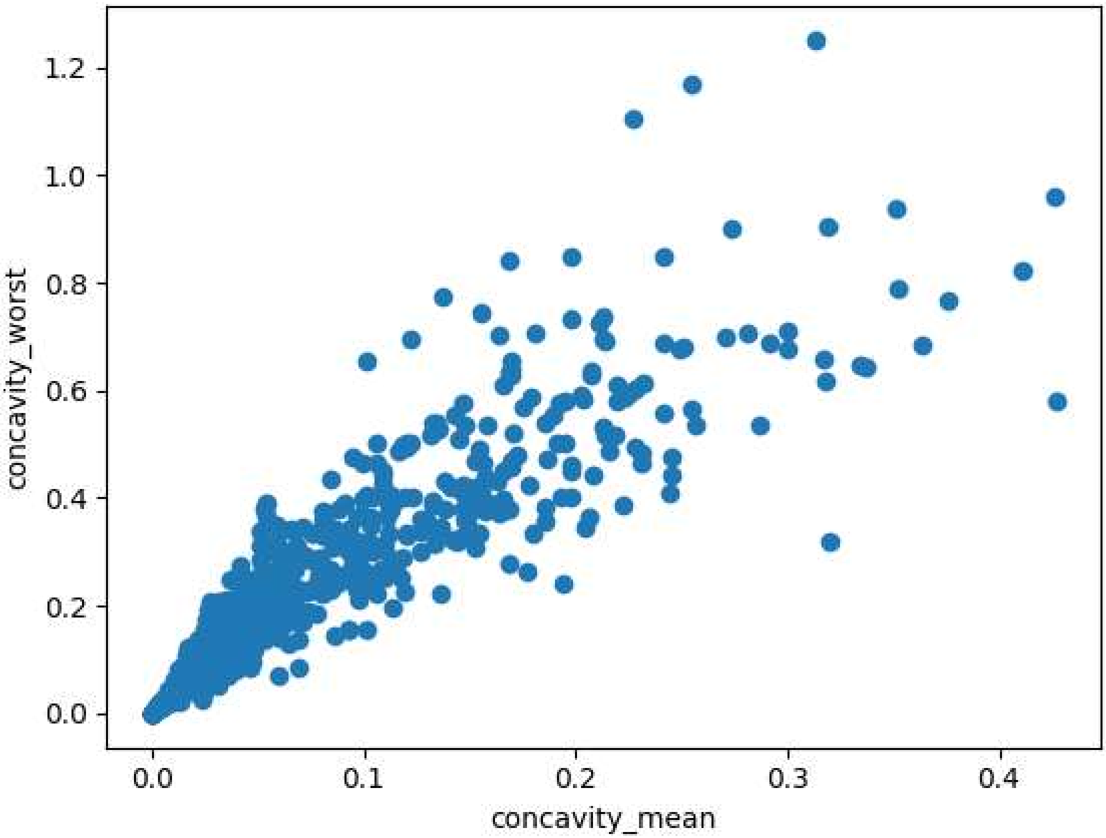
In [46]:
```
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='blue')
plt.plot(x_test,y_pred,color='black')
plt.show()
```



Here we got 100% accuracy so it is not a good model because no model is 100% accurate.Now we are going to implement KMeans.

In [10]:
```
plt.scatter(df["concavity_mean"],df["concavity_worst"])
plt.xlabel("concavity_mean")
plt.ylabel("concavity_worst")
```

Out[10]:
```
Text(0, 0.5, 'concavity_worst')
```



In [11]:
```
from sklearn.cluster import KMeans
```

In [12]:
```
km=KMeans()
km
```

Out[12]:
```
▾ KMeans

KMeans()
```

In [14]:

```
y_predicted=km.fit_predict(df[["concavity_mean","concavity_worst"]])
y_predicted
```

```
C:\Users\Lenovo\AppData\Local\Programs\Python\Python311\Lib\site-packages\skl
earn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` wil
l change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to su
ppress the warning
  warnings.warn(
```

Out[14]:

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

In [15]:

```
df["Cluster"]=y_predicted
df.head()
```

Out[15]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_me
0	842302	M	17.99	10.38	122.80	1001.0	0.118
1	842517	M	20.57	17.77	132.90	1326.0	0.084
2	84300903	M	19.69	21.25	130.00	1203.0	0.109
3	84348301	M	11.42	20.38	77.58	386.1	0.142
4	84358402	M	20.29	14.34	135.10	1297.0	0.100

5 rows × 34 columns

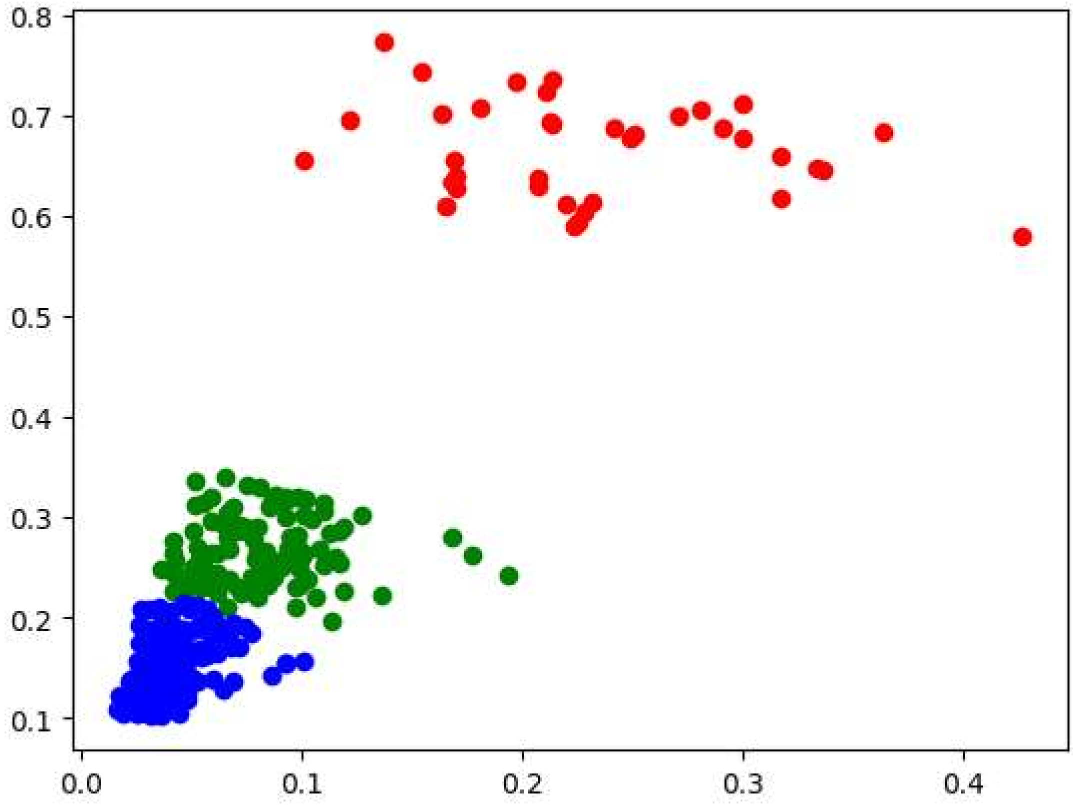
In [16]:

```
df1=df[df.Cluster==0]
df2=df[df.Cluster==1]
df3=df[df.Cluster==2]

plt.scatter(df1["concavity_mean"],df1["concavity_worst"],color="red")
plt.scatter(df2["concavity_mean"],df2["concavity_worst"],color="green")
plt.scatter(df3["concavity_mean"],df3["concavity_worst"],color="blue")
```

Out[16]:

```
<matplotlib.collections.PathCollection at 0x2bac32672d0>
```

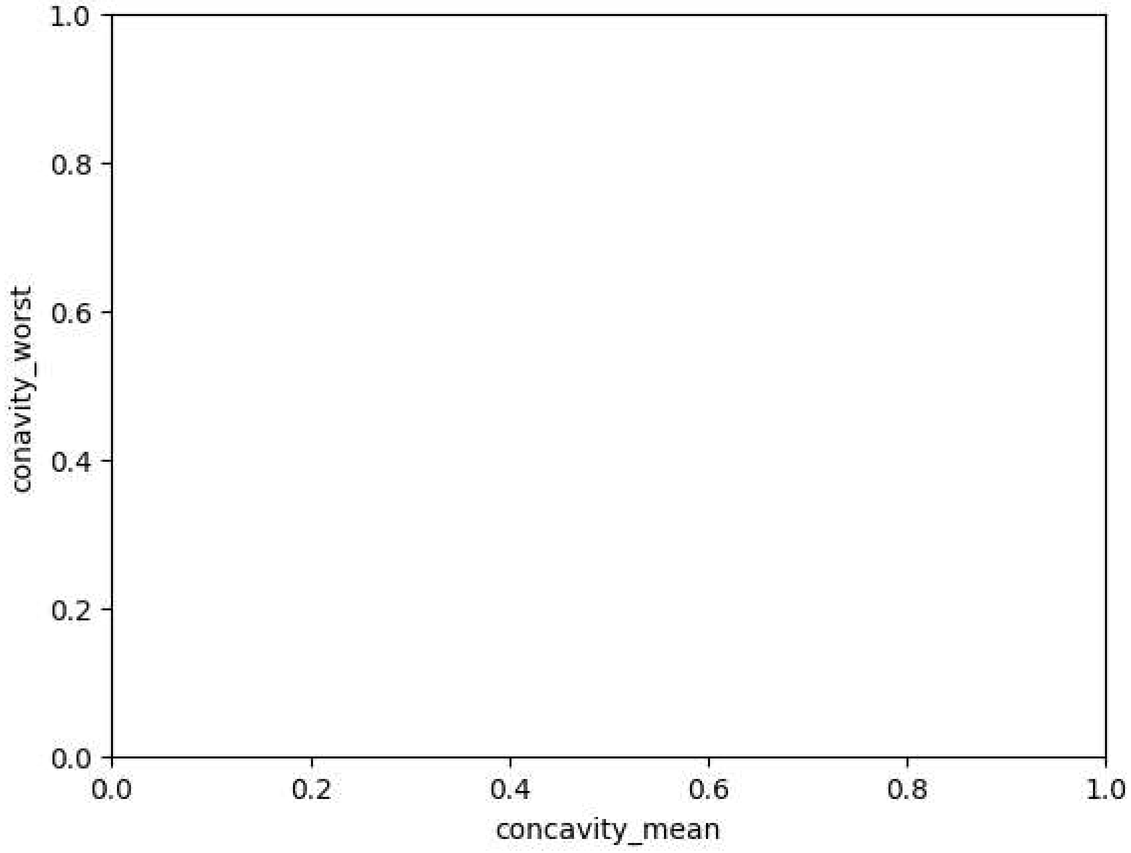


In [17]:

```
plt.xlabel("concavity_mean")
plt.ylabel("conavity_worst")
```

Out[17]:

```
Text(0, 0.5, 'conavity_worst')
```



In [18]:

```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
```

In [19]:

```
scaler.fit(df[["concavity_worst"]])
df["concavity_worst"]=scaler.transform(df[["concavity_worst"]])
df.head()
```

Out[19]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_me
0	842302	M	17.99	10.38	122.80	1001.0	0.118
1	842517	M	20.57	17.77	132.90	1326.0	0.084
2	84300903	M	19.69	21.25	130.00	1203.0	0.109
3	84348301	M	11.42	20.38	77.58	386.1	0.142
4	84358402	M	20.29	14.34	135.10	1297.0	0.100

5 rows × 34 columns

In [20]:

```
scaler.fit(df[["concavity_mean"]])
df["concavity_mean"]=scaler.transform(df[["concavity_mean"]])
df.head()
```

Out[20]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_me
0	842302	M	17.99	10.38	122.80	1001.0	0.118
1	842517	M	20.57	17.77	132.90	1326.0	0.084
2	84300903	M	19.69	21.25	130.00	1203.0	0.109
3	84348301	M	11.42	20.38	77.58	386.1	0.142
4	84358402	M	20.29	14.34	135.10	1297.0	0.100

5 rows × 34 columns

In [21]:

```
km=KMeans()
```

In [23]:

```
y_predicted=km.fit_predict(df[["concavity_mean","concavity_worst"]])
y_predicted
```

```
C:\Users\Lenovo\AppData\Local\Programs\Python\Python311\Lib\site-packages\skl
earn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` wil
l change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to su
ppress the warning
  warnings.warn(
```

Out[23]:

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

In [24]:
```
df["New Cluster"]=y_predicted
df.head()
```

Out[24]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_me
0	842302	M	17.99	10.38	122.80	1001.0	0.118
1	842517	M	20.57	17.77	132.90	1326.0	0.084
2	84300903	M	19.69	21.25	130.00	1203.0	0.109
3	84348301	M	11.42	20.38	77.58	386.1	0.142
4	84358402	M	20.29	14.34	135.10	1297.0	0.100

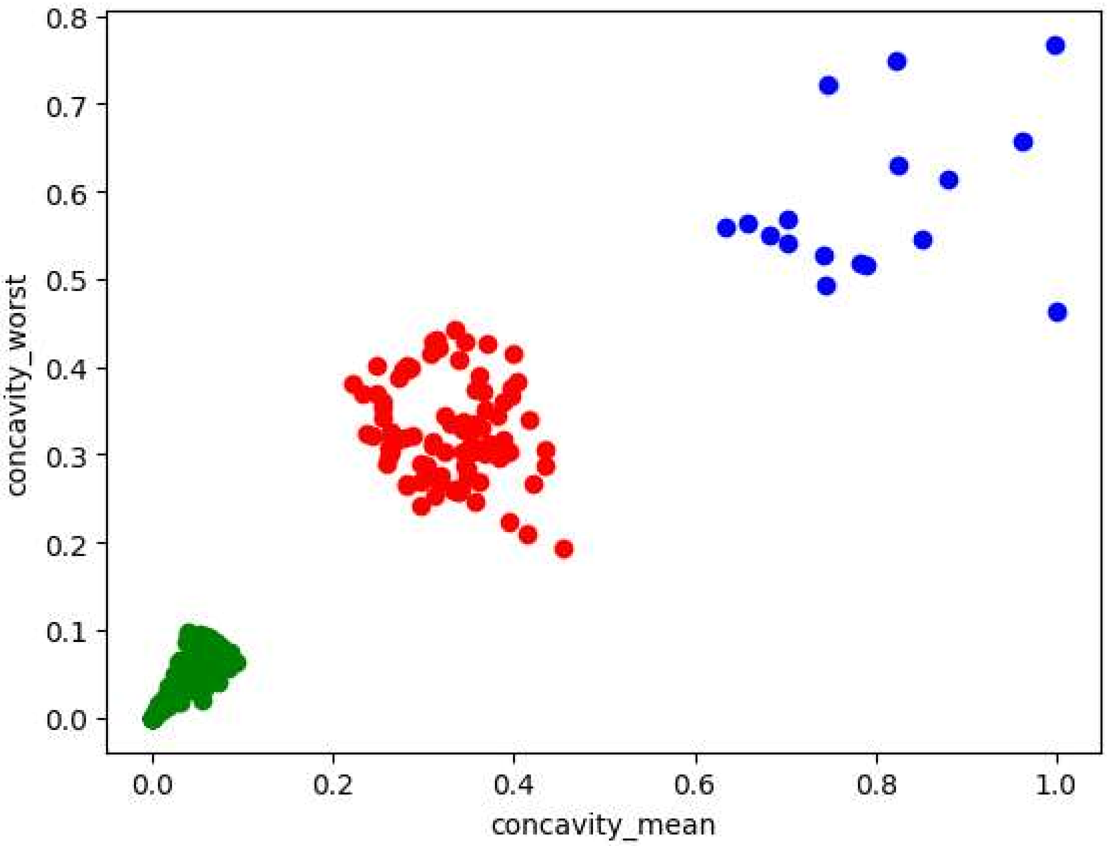
5 rows × 35 columns

In [25]:
```
df1=df[df["New Cluster"]==0]
df2=df[df["New Cluster"]==1]
df3=df[df["New Cluster"]==2]

plt.scatter(df1["concavity_mean"],df1["concavity_worst"],color="red")
plt.scatter(df2["concavity_mean"],df2["concavity_worst"],color="green")
plt.scatter(df3["concavity_mean"],df3["concavity_worst"],color="blue")

plt.xlabel("concavity_mean")
plt.ylabel("concavity_worst")
```

Out[25]:
```
Text(0, 0.5, 'concavity_worst')
```



In [26]:
```
km.cluster_centers_
```

Out[26]:
```
array([[0.32745965, 0.32597977],
       [0.03679972, 0.04489006],
       [0.7959645 , 0.58719696],
       [0.10542301, 0.13518171],
       [0.19993316, 0.23477417],
       [0.52024139, 0.41168862],
       [0.61401125, 0.84309904],
       [0.41086345, 0.54577025]])
```

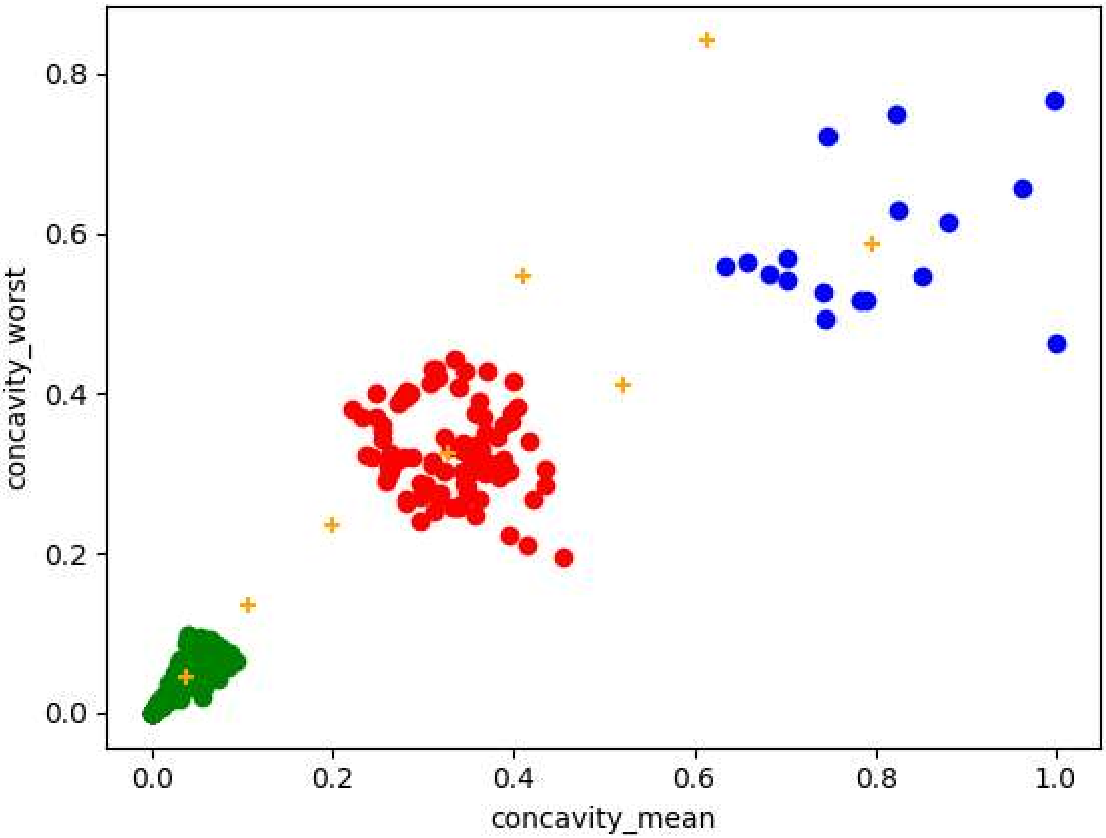
In [27]:
```
df1=df[df["New Cluster"]==0]
df2=df[df["New Cluster"]==1]
df3=df[df["New Cluster"]==2]

plt.scatter(df1["concavity_mean"],df1["concavity_worst"],color="red")
plt.scatter(df2["concavity_mean"],df2["concavity_worst"],color="green")
plt.scatter(df3["concavity_mean"],df3["concavity_worst"],color="blue")

plt.scatter(km.cluster_centers_[:,0],km.cluster_centers_[:,1],color="orange",ma

plt.xlabel("concavity_mean")
plt.ylabel("concavity_worst")
```

Out[27]:
```
Text(0, 0.5, 'concavity_worst')
```



In [28]:

```
k_rng=range(1,10)
sse=[]
```

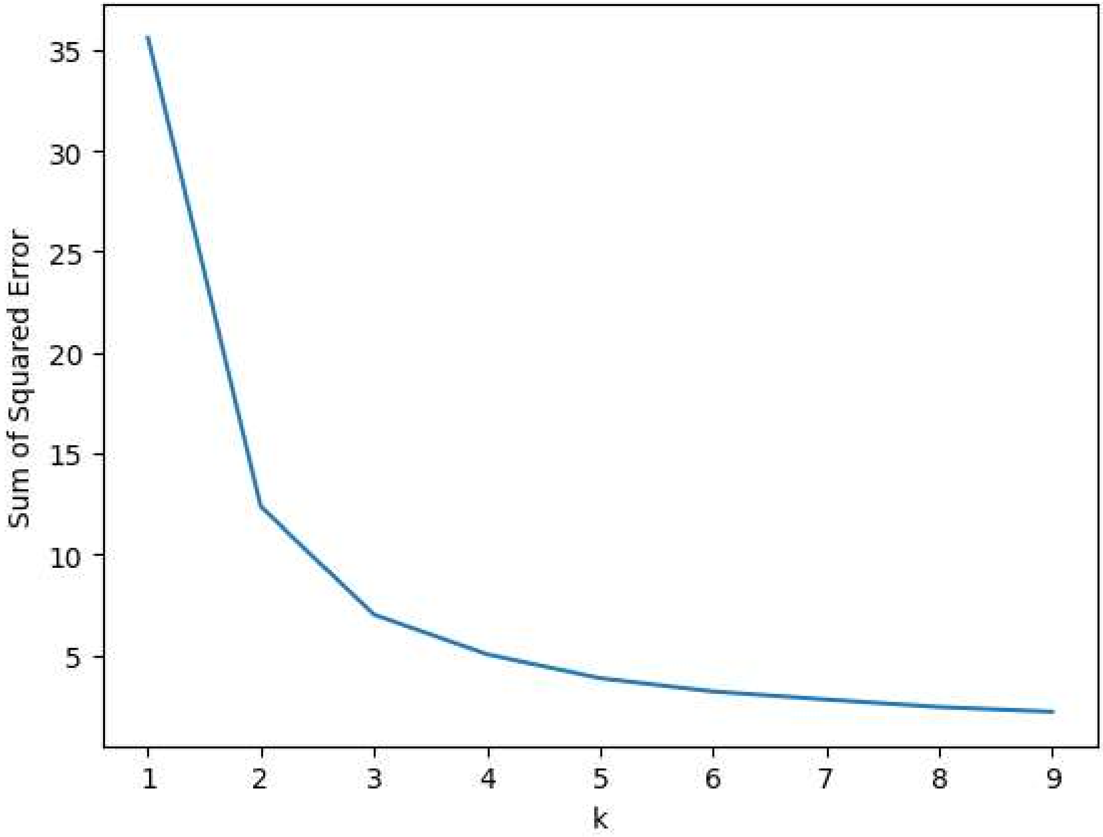
In [30]:

```
for k in k_rng:
    km=KMeans(n_clusters=k)
    km.fit(df[["concavity_mean","concavity_worst"]])
    sse.append(km.inertia_)
sse
```

```
C:\Users\Lenovo\AppData\Local\Programs\Python\Python311\Lib\site-packages\skl
earn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` wil
l change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to su
ppress the warning
  warnings.warn(
C:\Users\Lenovo\AppData\Local\Programs\Python\Python311\Lib\site-packages\skl
earn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` wil
l change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to su
ppress the warning
  warnings.warn(
C:\Users\Lenovo\AppData\Local\Programs\Python\Python311\Lib\site-packages\skl
earn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` wil
l change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to su
ppress the warning
  warnings.warn(
C:\Users\Lenovo\AppData\Local\Programs\Python\Python311\Lib\site-packages\skl
earn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` wil
l change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to su
ppress the warning
  warnings.warn(
C:\Users\Lenovo\AppData\Local\Programs\Python\Python311\Lib\site-packages\skl
earn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` wil
l change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to su
ppress the warning
  warnings.warn(
C:\Users\Lenovo\AppData\Local\Programs\Python\Python311\Lib\site-packages\skl
earn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` wil
l change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to su
ppress the warning
  warnings.warn(
C:\Users\Lenovo\AppData\Local\Programs\Python\Python311\Lib\site-packages\skl
earn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` wil
l change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to su
ppress the warning
  warnings.warn(
C:\Users\Lenovo\AppData\Local\Programs\Python\Python311\Lib\site-packages\skl
earn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` wil
l change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to su
ppress the warning
  warnings.warn(
C:\Users\Lenovo\AppData\Local\Programs\Python\Python311\Lib\site-packages\skl
earn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` wil
l change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to su
ppress the warning
  warnings.warn(
Out[30]: [35.588099422444216,
          12.362238940690691,
          7.005207400646277,
          5.040599245433761,
          3.847356007681436,
          3.1877599720365972,
          2.7885811415696944,
          2.4177251258785306,
          2.1790045494061587]
```

```
In [31]: plt.plot(k_rng,sse)
         plt.xlabel("k")
         plt.ylabel("Sum of Squared Error")
```

```
Out[31]: Text(0, 0.5, 'Sum of Squared Error')
```



Here we got the correct curve(Elbow curve) for the given dataset by using KMeans

CONCLUSION : We can conclude that clustering algorithm "KMeans" is the best model for the given dataset.