PROBLEM STATEMENT: To predict and study using the breast cancer diagnostic dataset.

In [20]:

```
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline
```

In [21]:

```
df=pd.read_csv(r"C:\Users\Teju\Downloads\BreastCancerPrediction.csv")
df
```

Out[21]:

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

569 rows × 33 columns

In [22]:

```
df.head()
```

Out[22]:

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

5 rows × 33 columns

In [23]:

```
df.tail()
```

Out[23]:

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

5 rows × 33 columns

In [24]:

```
df.info()
```

```
 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 [25]:

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

Out[25]:

```
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 [26]:

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

Out[26]:

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

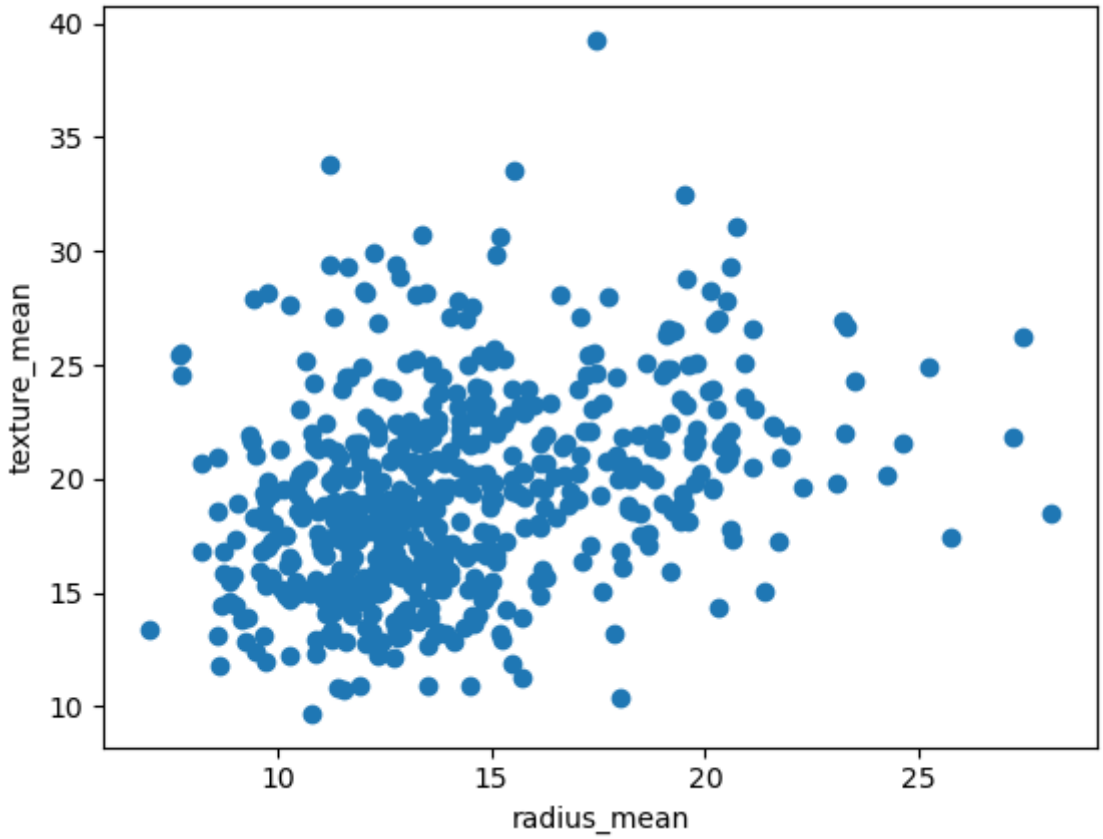
569 rows × 32 columns

In [27]:

```
plt.scatter(df["radius_mean"],df["texture_mean"])
plt.xlabel("radius_mean")
plt.ylabel("texture_mean")
```

Out[27]:

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



In [28]:

```
from sklearn.cluster import KMeans
km=KMeans()
km
```

Out[28]:

```
KMeans()
```

In [29]:

```
y_predicted=km.fit_predict(df[["radius_mean","texture_mean"]])
y_predicted
```

```
C:\Users\Teju\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto'
in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\Teju\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, wh
en there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=3.
  warnings.warn(
```

Out[29]:

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

In [30]:

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

Out[30]:

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

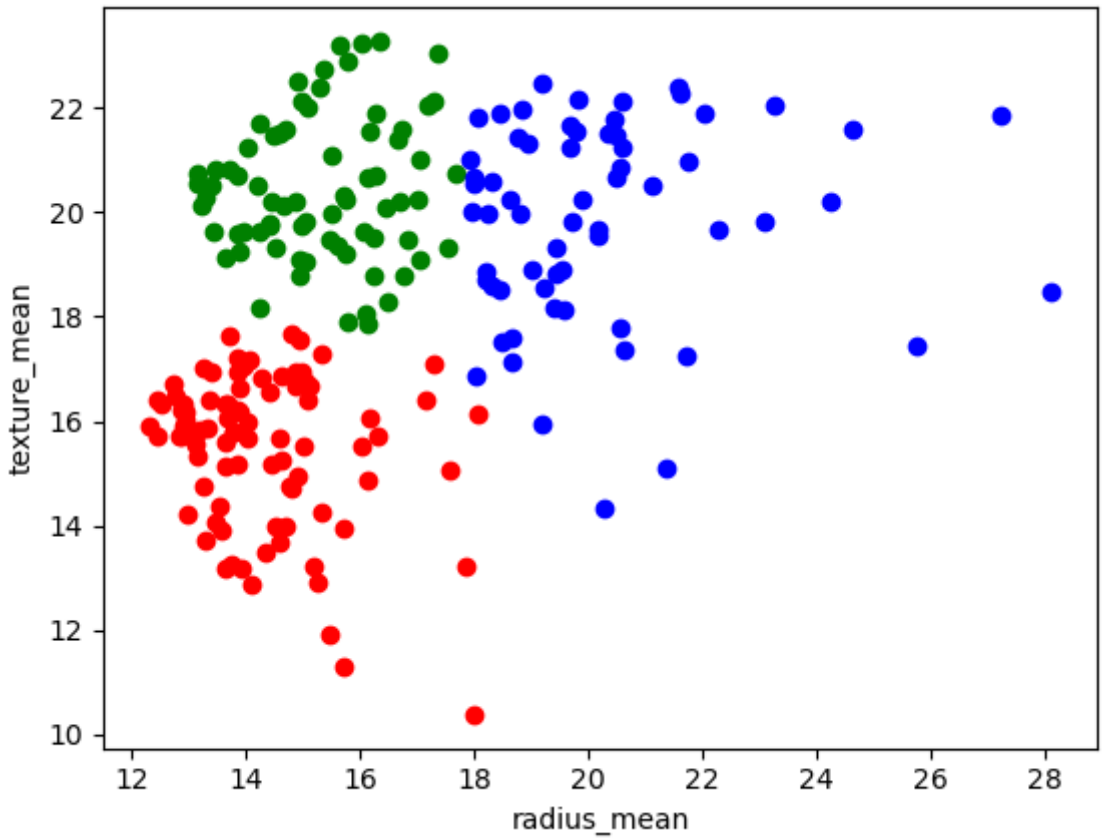
5 rows × 34 columns

In [31]:

```
df1=df[df.cluster==0]
df2=df[df.cluster==1]
df3=df[df.cluster==2]
plt.scatter(df1["radius_mean"],df1["texture_mean"],color="red")
plt.scatter(df2["radius_mean"],df2["texture_mean"],color="green")
plt.scatter(df3["radius_mean"],df3["texture_mean"],color="blue")
plt.xlabel("radius_mean")
plt.ylabel("texture_mean")
```

Out[31]:

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



In [32]:

```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
scaler.fit(df[["texture_mean"]])
df["texture_mean"]=scaler.transform(df[["texture_mean"]])
df.head()
```

Out[32]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothnes
0	842302	M	17.99	0.022658	122.80	1001.0	
1	842517	M	20.57	0.272574	132.90	1326.0	
2	84300903	M	19.69	0.390260	130.00	1203.0	
3	84348301	M	11.42	0.360839	77.58	386.1	
4	84358402	M	20.29	0.156578	135.10	1297.0	

5 rows × 34 columns

In [33]:

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

Out[33]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothnes
0	842302	M	0.521037	0.022658	122.80	1001.0	
1	842517	M	0.643144	0.272574	132.90	1326.0	
2	84300903	M	0.601496	0.390260	130.00	1203.0	
3	84348301	M	0.210090	0.360839	77.58	386.1	
4	84358402	M	0.629893	0.156578	135.10	1297.0	

5 rows × 34 columns

In [34]:

```
y_predicted=km.fit_predict(df[["radius_mean","texture_mean"]])
y_predicted
```

```
C:\Users\Teju\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto'
in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\Teju\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, wh
en there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=3.
  warnings.warn(
```

Out[34]:

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

In [35]:

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

Out[35]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothnes
0	842302	M	0.521037	0.022658	122.80	1001.0	
1	842517	M	0.643144	0.272574	132.90	1326.0	
2	84300903	M	0.601496	0.390260	130.00	1203.0	
3	84348301	M	0.210090	0.360839	77.58	386.1	
4	84358402	M	0.629893	0.156578	135.10	1297.0	

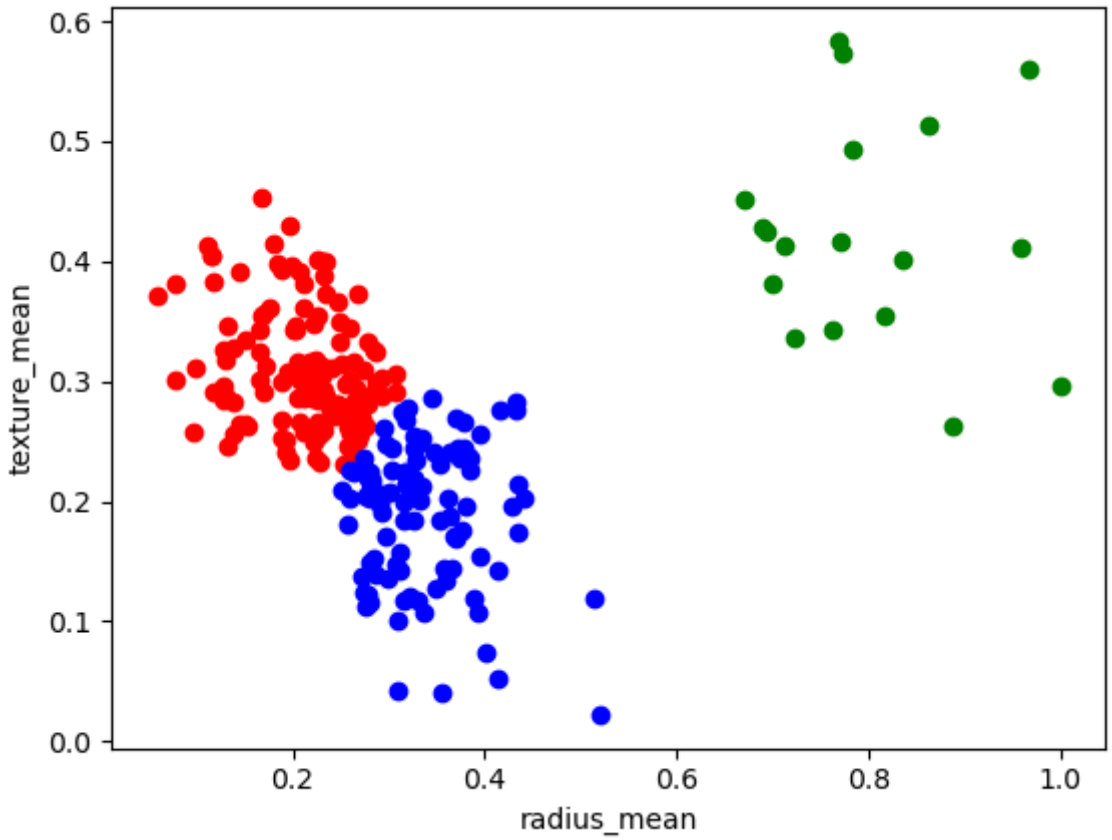
5 rows × 35 columns

In [36]:

```
df1=df[df["New Cluster"]==0]
df2=df[df["New Cluster"]==1]
df3=df[df["New Cluster"]==2]
plt.scatter(df1["radius_mean"],df1["texture_mean"],color="red")
plt.scatter(df2["radius_mean"],df2["texture_mean"],color="green")
plt.scatter(df3["radius_mean"],df3["texture_mean"],color="blue")
plt.xlabel("radius_mean")
plt.ylabel("texture_mean")
```

Out[36]:

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



In [37]:

```
km.cluster_centers_
```

Out[37]:

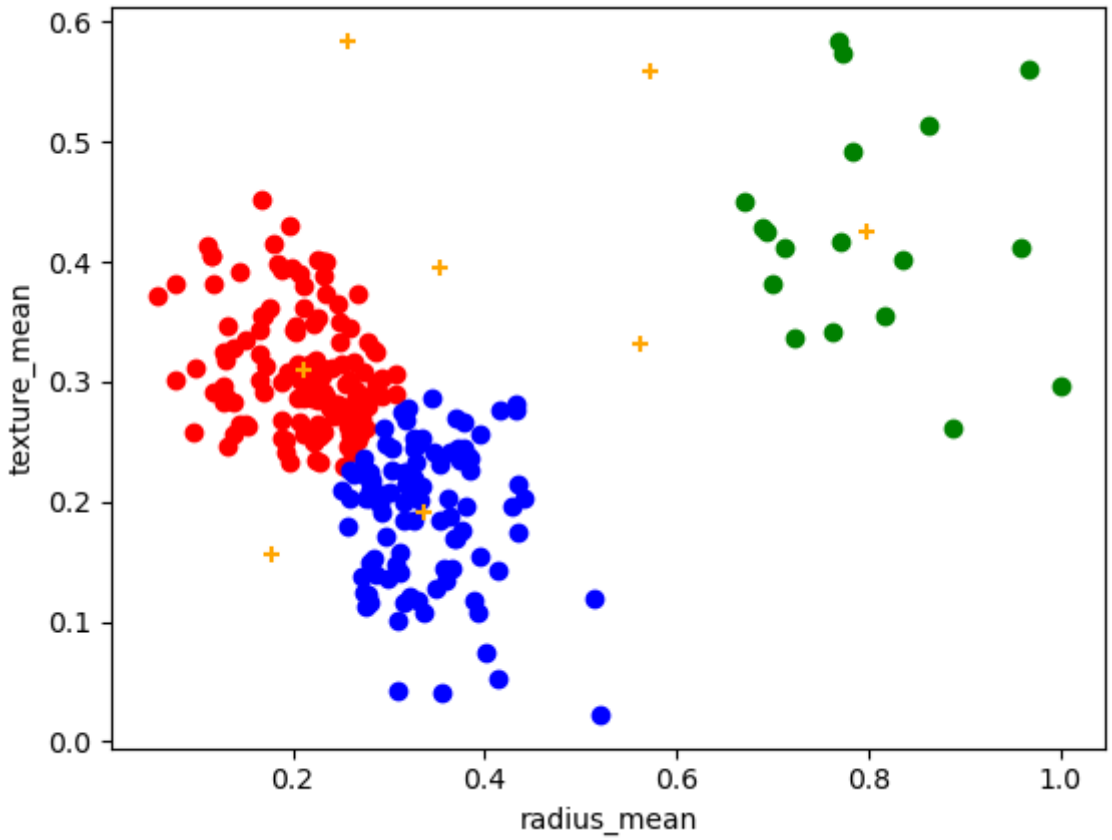
```
array([[0.2104771 , 0.31042356],
       [0.79840767, 0.42469846],
       [0.33570532, 0.19063107],
       [0.25627183, 0.58431314],
       [0.57132058, 0.55893025],
       [0.35339953, 0.39439771],
       [0.17694105, 0.15527139],
       [0.56287997, 0.33184226]])
```

In [38]:

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

Out[38]:

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



In [39]:

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

In [40]:

```
for k in k_rng:
    km=KMeans(n_clusters=k)
    km.fit(df[["radius_mean","texture_mean"]])
    sse.append(km.inertia_)
#km.inertia_ will give you the value of sum of square error
print(sse)
plt.plot(k_rng,sse)
plt.xlabel("K")
plt.ylabel("Sum of Squared Error")
```

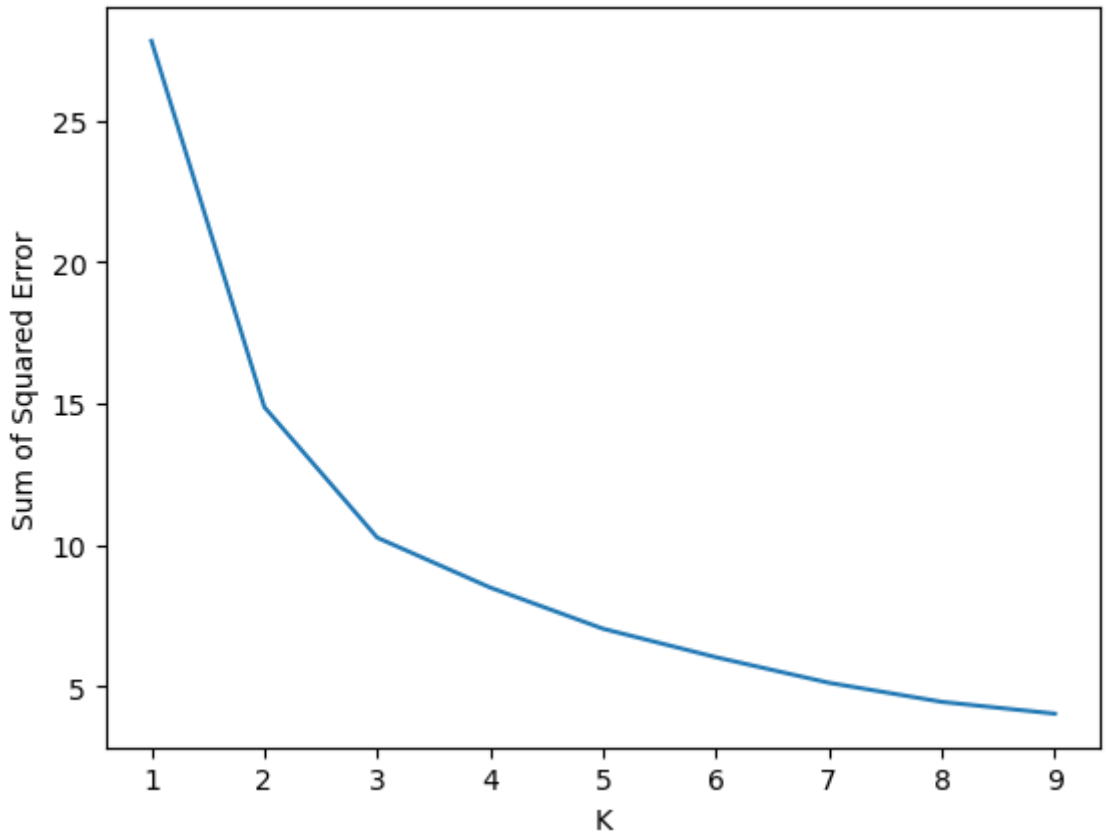
```
C:\Users\Teju\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto'
in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\Teju\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, wh
en there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=3.
  warnings.warn(
C:\Users\Teju\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto'
in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\Teju\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, wh
en there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=3.
  warnings.warn(
C:\Users\Teju\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto'
in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\Teju\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, wh
en there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=3.
  warnings.warn(
C:\Users\Teju\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto'
in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\Teju\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, wh
en there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=3.
  warnings.warn(
C:\Users\Teju\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto'
in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\Teju\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, wh
en there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=3.
  warnings.warn(
C:\Users\Teju\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto'
in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\Teju\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, wh
en there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=3.
  warnings.warn(
C:\Users\Teju\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto'
in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\Teju\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, wh
en there are less chunks than available threads. You can avoid it by setti
```

```
ng the environment variable OMP_NUM_THREADS=3.
  warnings.warn(
C:\Users\Teju\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto'
in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\Teju\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, wh
en there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=3.
  warnings.warn(
C:\Users\Teju\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto'
in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\Teju\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1382:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, wh
en there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=3.
  warnings.warn(
```

```
[27.81750759504308, 14.87203295827117, 10.2527514961052, 8.48846417415106
4, 7.030668267339052, 6.024690703935413, 5.12063124658119, 4.4458585717882
82, 4.028880497150342]
```

Out[40]:

```
Text(0, 0.5, 'Sum of Squared Error')
```



Conclusion:

Based on accuracy of all models that were implemented we can conclude that "K-Means Clustering" is the best model for the given dataset

In [ ]: