In [13]:
```
import pandas as pd
import seaborn as sns
```

Data Set

[https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data (https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data)](https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data)

In [14]:
```
df = pd.read_csv("Data.csv")
```

In [15]:
```
df.head()
```

Out[15]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_m
0	842302	M	17.99	10.38	122.80	1001.0	0.11
1	842517	M	20.57	17.77	132.90	1326.0	0.08
2	84300903	M	19.69	21.25	130.00	1203.0	0.10
3	84348301	M	11.42	20.38	77.58	386.1	0.14
4	84358402	M	20.29	14.34	135.10	1297.0	0.10

5 rows × 33 columns

In [16]:

```
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 [17]:

```
df.describe()
```

Out[17]:

	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 × 32 columns

In [18]:

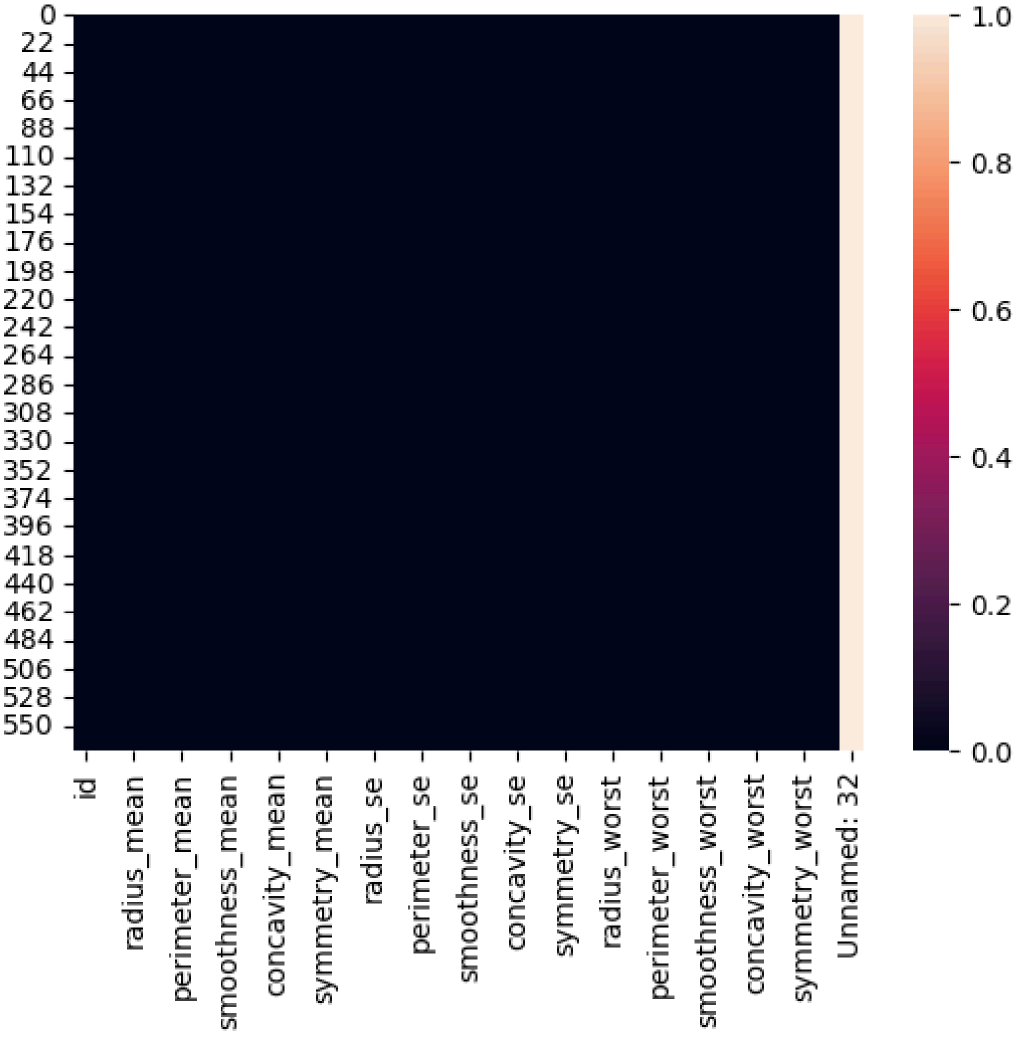
```
df.isnull().sum()
```

Out[18]:

```
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 [19]:
```
sns.heatmap(df.isnull()) #Find Missing Values
```

Out[19]:
```
<Axes: >
```



In [20]:
```
df.drop(['id','Unnamed: 32'],axis=1,inplace=True) #Drop Column
```

In [21]:
```
df
```

Out[21]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	con
0	M	17.99	10.38	122.80	1001.0	0.11840	
1	M	20.57	17.77	132.90	1326.0	0.08474	
2	M	19.69	21.25	130.00	1203.0	0.10960	
3	M	11.42	20.38	77.58	386.1	0.14250	
4	M	20.29	14.34	135.10	1297.0	0.10030	
...	...	...	...	...	...	...	
564	M	21.56	22.39	142.00	1479.0	0.11100	
565	M	20.13	28.25	131.20	1261.0	0.09780	
566	M	16.60	28.08	108.30	858.1	0.08455	
567	M	20.60	29.33	140.10	1265.0	0.11780	
568	B	7.76	24.54	47.92	181.0	0.05263	

569 rows × 31 columns

In [26]:
```
df.diagnosis = [1 if value =='M' else 0 for value in df.diagnosis]
```

In [27]:
```
df.head(10)
```

Out[27]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	comp
0	1	17.99	10.38	122.80	1001.0	0.11840	
1	1	20.57	17.77	132.90	1326.0	0.08474	
2	1	19.69	21.25	130.00	1203.0	0.10960	
3	1	11.42	20.38	77.58	386.1	0.14250	
4	1	20.29	14.34	135.10	1297.0	0.10030	
5	1	12.45	15.70	82.57	477.1	0.12780	
6	1	18.25	19.98	119.60	1040.0	0.09463	
7	1	13.71	20.83	90.20	577.9	0.11890	
8	1	13.00	21.82	87.50	519.8	0.12730	
9	1	12.46	24.04	83.97	475.9	0.11860	

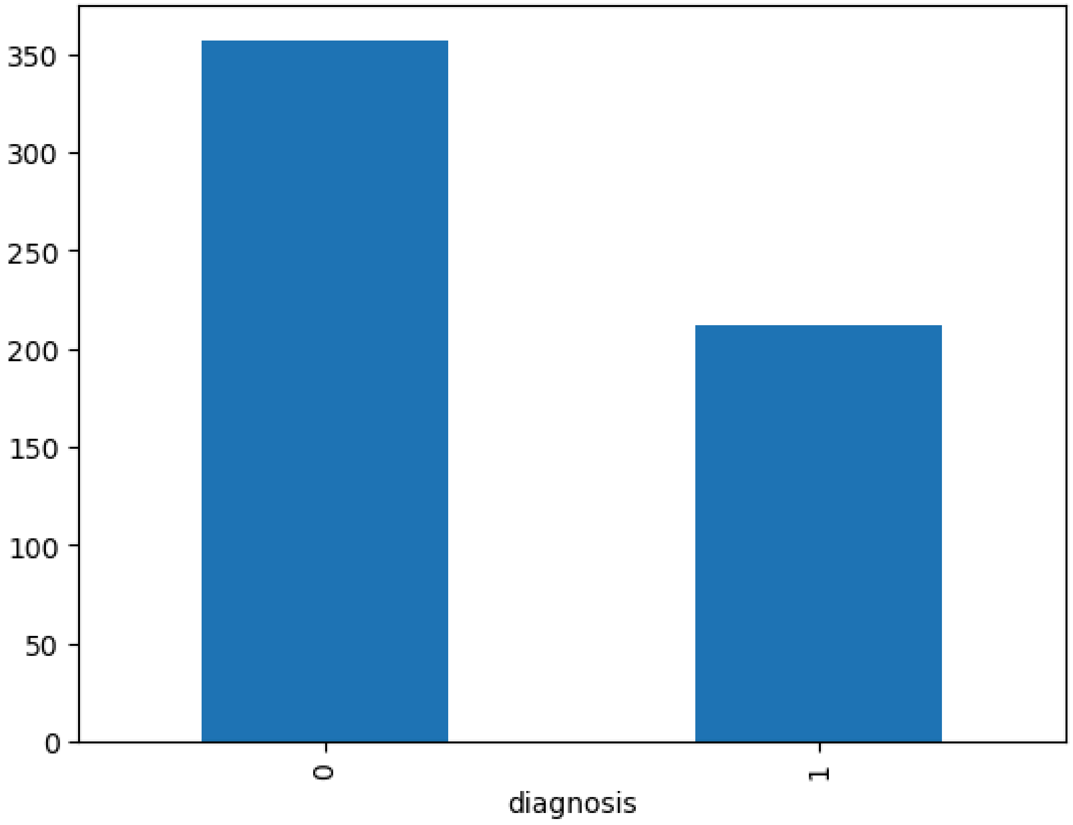
10 rows × 31 columns

In [36]:
```
df.diagnosis.unique()
```

Out[36]:
```
array([1, 0], dtype=int64)
```

In [46]:
```
df.diagnosis.value_counts().plot(kind='bar')
```

Out[46]:
```
<Axes: xlabel='diagnosis'>
```



In [49]:
```
#Divide Variables
y=df.diagnosis
X = df.drop(['diagnosis'],axis=1)
```

In [50]:
```
X
```

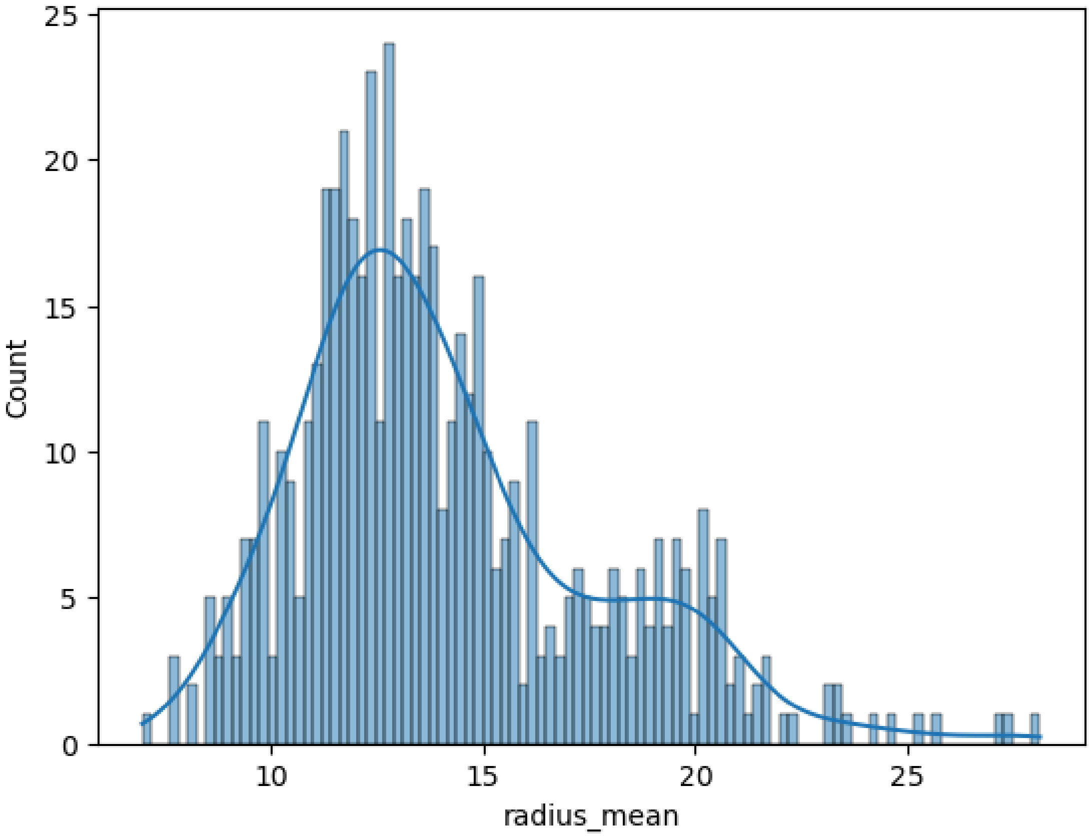
Out[50]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_r
0	17.99	10.38	122.80	1001.0	0.11840	0.2
1	20.57	17.77	132.90	1326.0	0.08474	0.0
2	19.69	21.25	130.00	1203.0	0.10960	0.1
3	11.42	20.38	77.58	386.1	0.14250	0.2
4	20.29	14.34	135.10	1297.0	0.10030	0.1
...	...	...	...	...	...	
564	21.56	22.39	142.00	1479.0	0.11100	0.1
565	20.13	28.25	131.20	1261.0	0.09780	0.1
566	16.60	28.08	108.30	858.1	0.08455	0.1
567	20.60	29.33	140.10	1265.0	0.11780	0.2
568	7.76	24.54	47.92	181.0	0.05263	0.0

569 rows × 30 columns

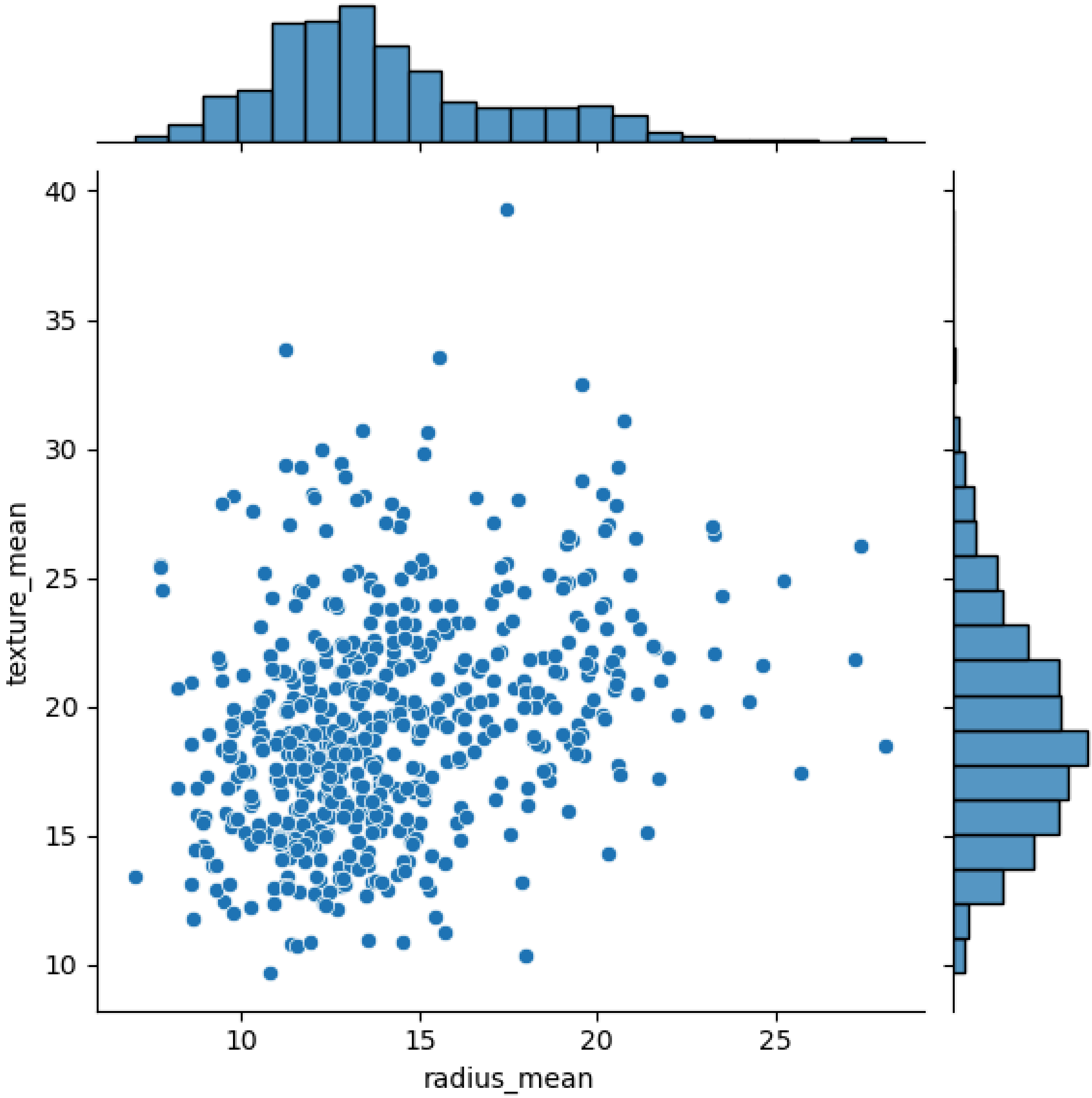
In [56]:
```
sns.histplot(df.radius_mean,bins=100,kde=True)
```

Out[56]:
```
<Axes: xlabel='radius_mean', ylabel='Count'>
```



In [57]:
```
sns.jointplot(x='radius_mean',y='texture_mean',data=df)
```

Out[57]:
```
<seaborn.axisgrid.JointGrid at 0x1ad1c2f8b10>
```



Normalize Data

In [59]:
```
from sklearn.preprocessing import StandardScaler
```

In [60]:
```
scaler = StandardScaler()
```

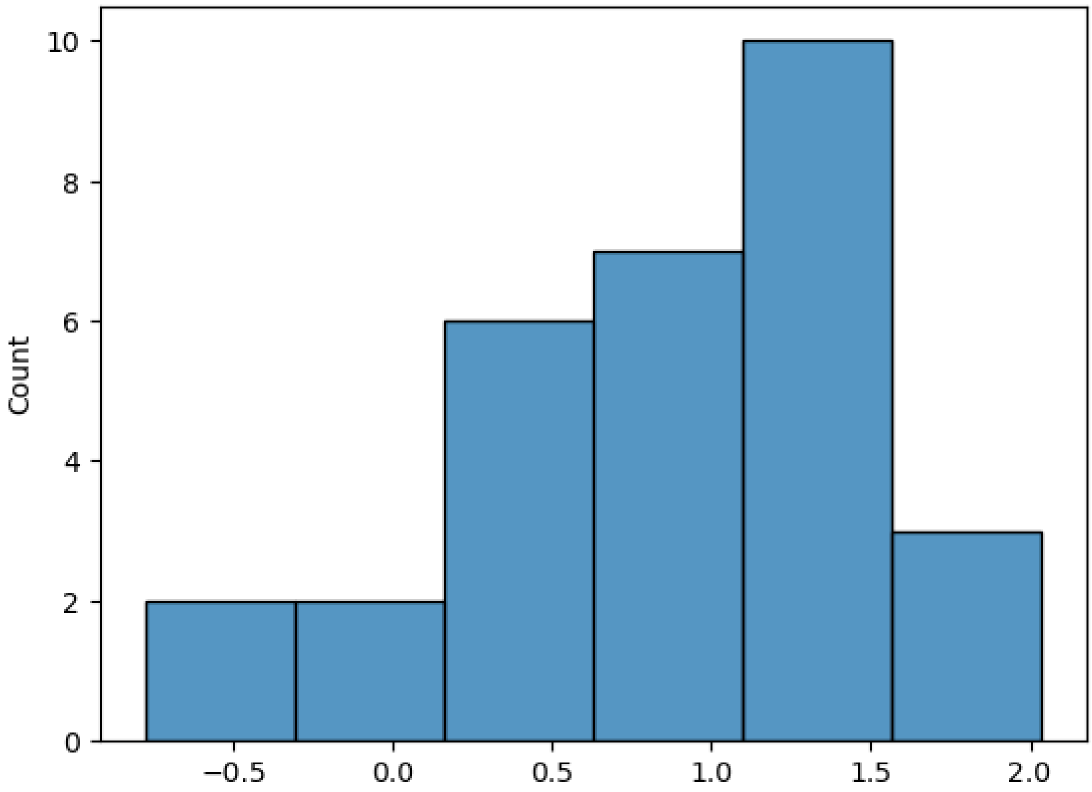
In [61]:
```
nor_X = scaler.fit_transform(X)
```

```
In [62]: nor_X
```

```
Out[62]: array([[ 1.09706398, -2.07333501,  1.26993369, ...,  2.29607613,
                  2.75062224,  1.93701461],
                [ 1.82982061, -0.35363241,  1.68595471, ...,  1.0870843 ,
                 -0.24388967,  0.28118999],
                [ 1.57988811,  0.45618695,  1.56650313, ...,  1.95500035,
                  1.152255  ,  0.20139121],
                ...,
                [ 0.70228425,  2.0455738 ,  0.67267578, ...,  0.41406869,
                 -1.10454895, -0.31840916],
                [ 1.83834103,  2.33645719,  1.98252415, ...,  2.28998549,
                  1.91908301,  2.21963528],
                [-1.80840125,  1.22179204, -1.81438851, ..., -1.74506282,
                 -0.04813821, -0.75120669]])
```

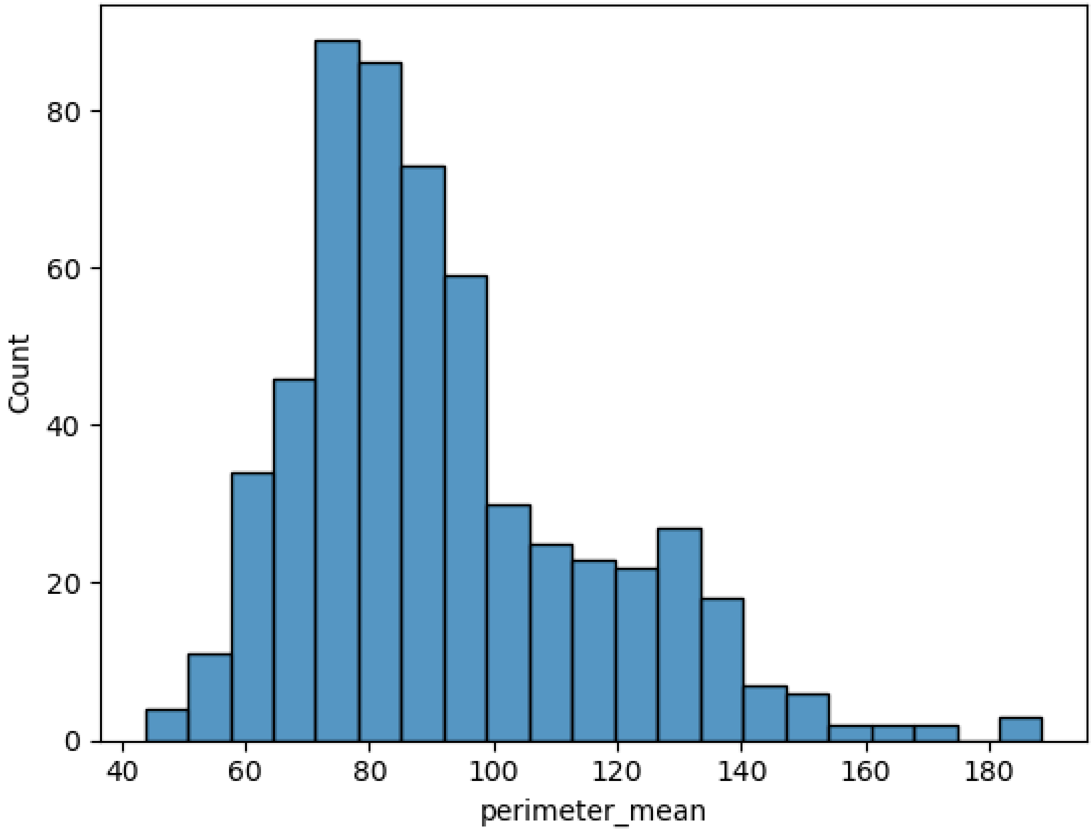
```
In [78]: sns.histplot(nor_X[2])
```

```
Out[78]: <Axes: ylabel='Count'>
```



```
In [79]: sns.histplot(X.perimeter_mean)

Out[79]: <Axes: xlabel='perimeter_mean', ylabel='Count'>
```



```
In [81]: from sklearn.model_selection import train_test_split

In [85]: X_train, X_test, y_train, y_test=train_test_split(nor_X,y,test_size=0.3,random

In [86]: X_train

Out[86]: array([[-0.10999635, -0.32105347, -0.15854246, ..., -0.82857392,
                 -0.89100191, -0.76506065],
                [-0.2150816 , -0.67476767, -0.24174666, ..., -0.37801891,
                 -1.37957166, -0.42480753],
                [ 0.15981713, -1.23559085,  0.25747857, ..., -0.05795583,
                 -0.11932056,  0.45076239],
                ...,
                [ 0.04621146, -0.57470379, -0.06874782, ..., -1.23756033,
                 -0.71628161, -1.26047806],
                [-0.04183295,  0.07687501, -0.03497186, ...,  1.03683652,
                  0.45013821,  1.19444266],
                [-0.5530585 ,  0.28631105, -0.60751564, ..., -0.61357437,
                 -0.33448538, -0.84042616]])
```

Regression

```
In [87]: from sklearn.linear_model import LogisticRegression
In [88]: lr = LogisticRegression()
```

```
In [89]: lr.fit(X_train,y_train)
```

```
Out[89]: ▾ LogisticRegression
         LogisticRegression()
```

Predict The Values

```
In [92]: y_pred = lr.predict(X_test)
```

```
In [93]: y_pred
```

```
Out[93]: array([0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0,
                1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0,
                0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0,
                1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1,
                0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0,
                1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1,
                0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0,
                0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0], dtype=int64)
```

```
In [94]: y_test
```

```
Out[94]: 204    0
         70     1
         131    1
         431    0
         540    0
               ..
         69     0
         542    0
         176    0
         501    1
         247    0
         Name: diagnosis, Length: 171, dtype: int64
```

Evaluation the Model

```
In [95]: from sklearn.metrics import accuracy_score
```

```
In [96]: accuracy = accuracy_score(y_pred,y_test)
```

```
In [102]: accuracy
```

```
Out[102]: 0.9824561403508771
```

```
In [104]: print(f"Accuracy : {accuracy: .2f}")

          Accuracy :  0.98
```

Thank you

By Lahiru Sadakelum