D6

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [4]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\8_BreastCancerPrediction.csv")
df
```

Out[4]:

	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

In [5]:
```
df.head(10)
```

Out[5]:

	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	843786	M	12.45	15.70	82.57	477.1	0.12
6	844359	M	18.25	19.98	119.60	1040.0	0.09
7	84458202	M	13.71	20.83	90.20	577.9	0.11
8	844981	M	13.00	21.82	87.50	519.8	0.12
9	84501001	M	12.46	24.04	83.97	475.9	0.11

10 rows × 33 columns

In [6]:
```
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 [5]:
```
dff=df.drop("Unnamed: 32",axis=1)
```

In [8]:
```
dff.describe()
```

Out[8]:

	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 × 31 columns

In [6]:
```
dff.columns
```

Out[6]:
```
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')
```

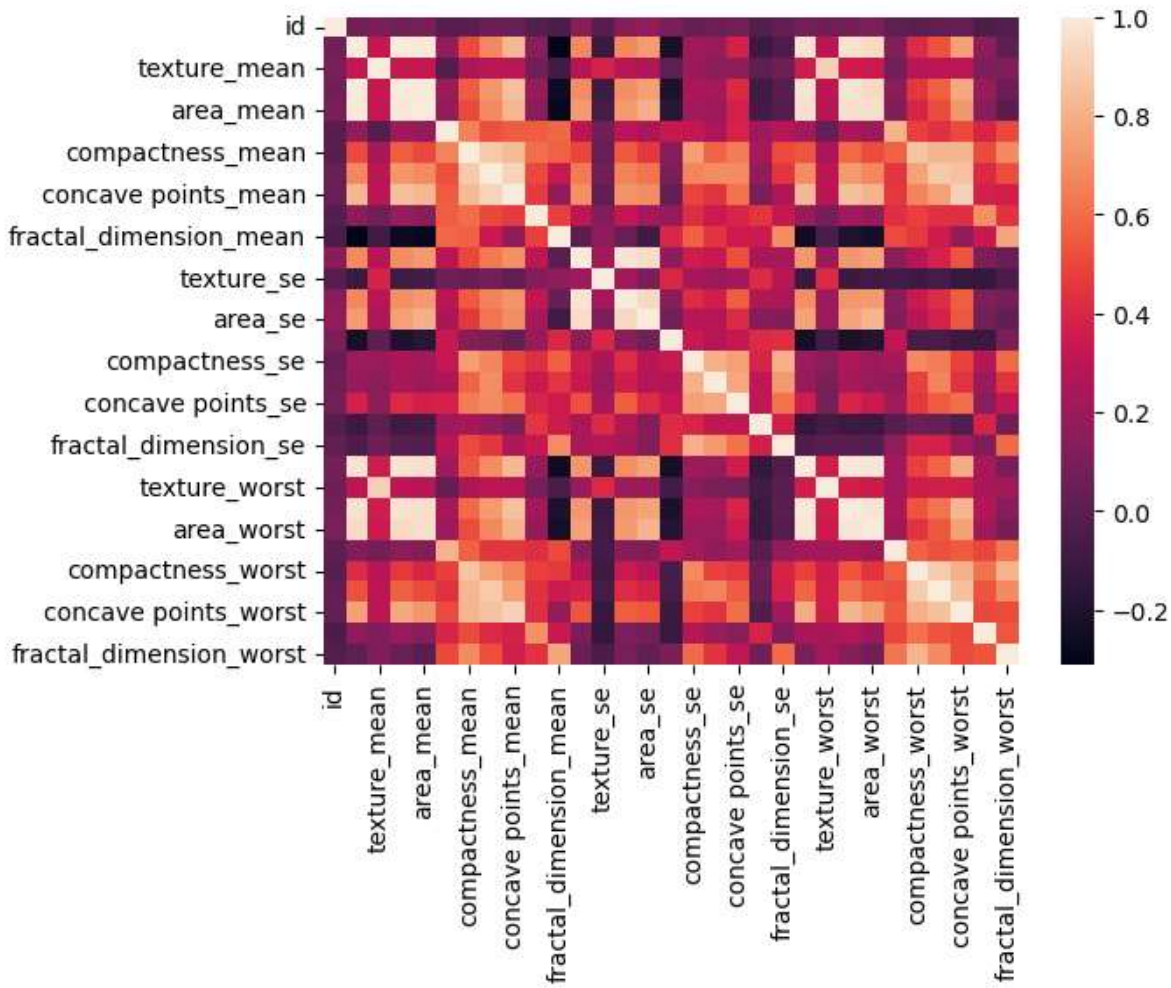
sns.pairplot(dff)

sns.distplot(dff["texture_worst"])

In [13]:
```
df1=df[['id', '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']]
```

In [14]:
```
sns.heatmap(df1.corr())
```

Out[14]: <Axes: >



In [16]:
```
x=df1[['id', '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=df1["texture_worst"]
```

In [17]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [18]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[18]:
```
▾ LinearRegression
LinearRegression()
```

In [19]:
```
print(lr.intercept_)
```

```
6.573534250264856e-08
```

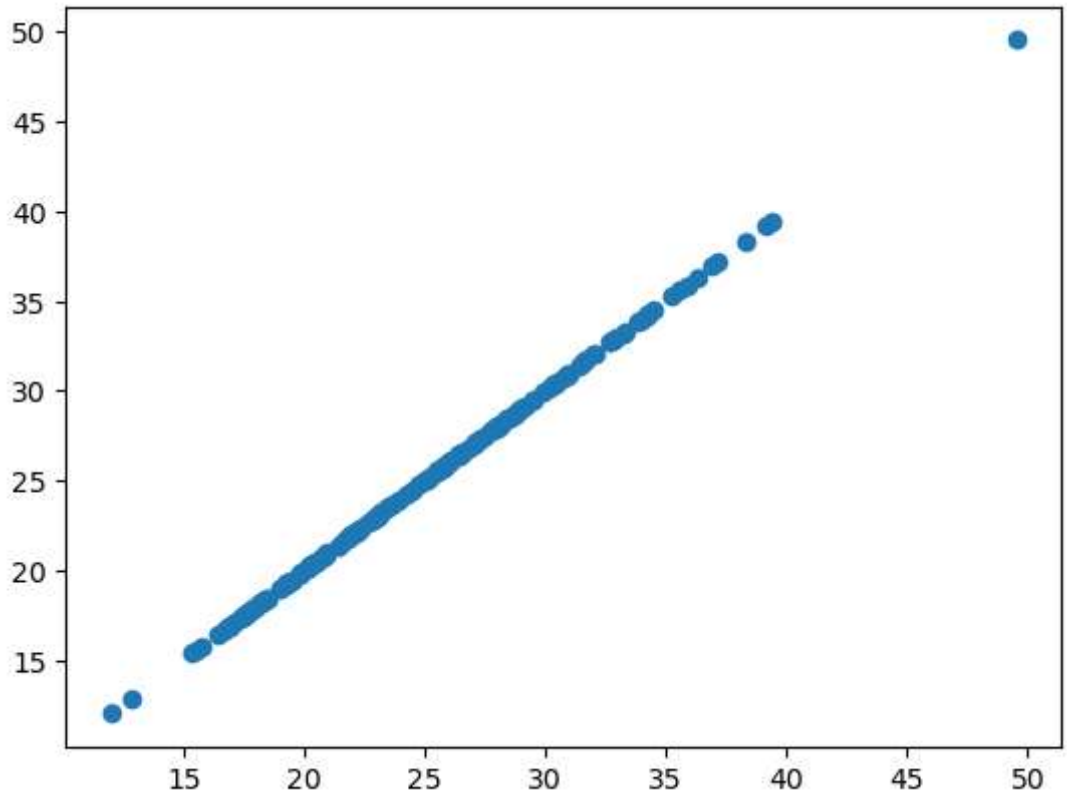
In [20]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[20]:

	Co-efficient
id	2.201906e-17
radius_mean	-8.756362e-08
texture_mean	1.212056e-09
perimeter_mean	1.352952e-08
area_mean	-1.492237e-10
smoothness_mean	-1.107278e-07
compactness_mean	-2.412592e-07
concavity_mean	-1.365733e-08
concave points_mean	2.251890e-07
symmetry_mean	-9.095843e-07
fractal_dimension_mean	2.264502e-07
radius_se	1.569921e-07
texture_se	-9.268980e-10
perimeter_se	-3.156758e-08
area_se	-3.680535e-11
smoothness_se	4.367222e-08
compactness_se	5.591750e-07
concavity_se	1.028080e-06
concave points_se	-2.970168e-09
symmetry_se	1.604074e-08
fractal_dimension_se	-5.090723e-08
radius_worst	-2.104681e-08
texture_worst	1.000000e+00
perimeter_worst	3.495093e-09
area_worst	5.743664e-11
smoothness_worst	-3.323748e-07
compactness_worst	-3.978459e-07
concavity_worst	-9.464521e-08
concave points_worst	5.077955e-07
symmetry_worst	7.018295e-07
fractal_dimension_worst	-7.592206e-08

In [21]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[21]:
```
<matplotlib.collections.PathCollection at 0x202aaa1f950>
```



In [22]:
```
print(lr.score(x_test,y_test))
```

```
0.9999999999999999
```

In [23]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [24]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

```
C:\Users\user\anaconda3\Lib\site-packages\sklearn\linear_model\_ridge.py:216:
LinAlgWarning: Ill-conditioned matrix (rcond=1.39846e-18): result may not be
accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
```

Out[24]:
```
▾ Ridge
Ridge(alpha=10)
```

In [25]:
```
rr.score(x_test,y_test)
```

Out[25]:
```
0.9999906754449169
```

In [26]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[26]:
```
Lasso
Lasso(alpha=10)
```

In [27]:
```
la.score(x_test,y_test)
```

Out[27]:
```
0.9168216401120295
```

In [ ]: