D6

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

In [2]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\8_BreastCancerPrediction.csv")
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

In [3]:
```
df.head(10)
```

Out[3]:

	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 [4]:
```
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 [6]:
```
dff.describe()
```

Out[6]:

	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 [7]:
```
dff.columns
```

Out[7]:
```
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')
```

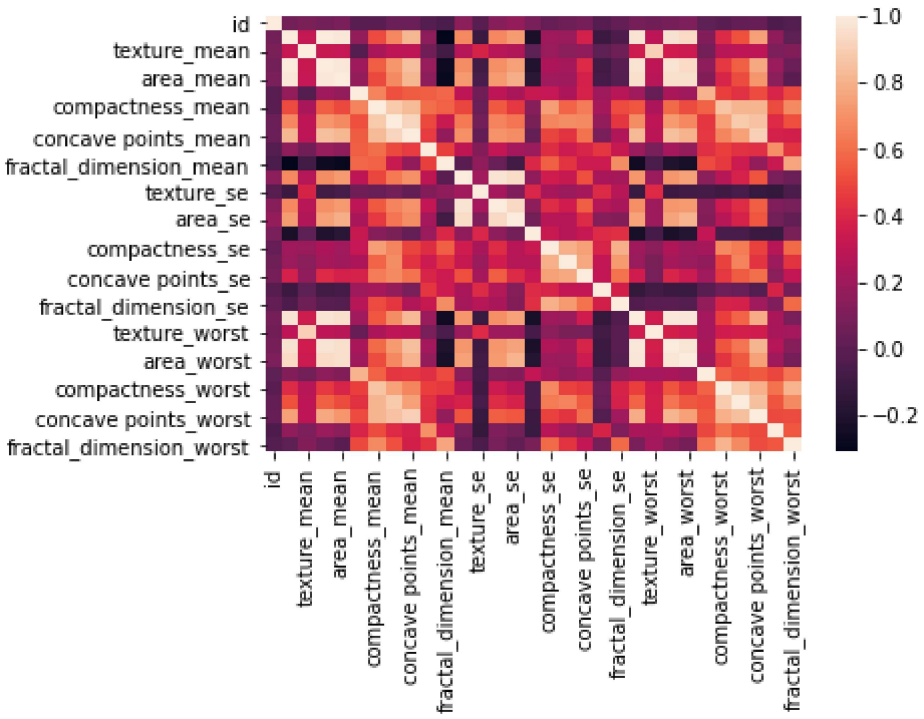
sns.pairplot(dff)

sns.distplot(dff["texture_worst"])

In [8]:
```
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 [9]:
```
sns.heatmap(df1.corr())
```

Out[9]: <AxesSubplot:>



In [10]:
```
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 [11]:
```
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 [12]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[12]: LinearRegression()

In [13]:
```
print(lr.intercept_)
```

```
8.89242457446926e-08
```

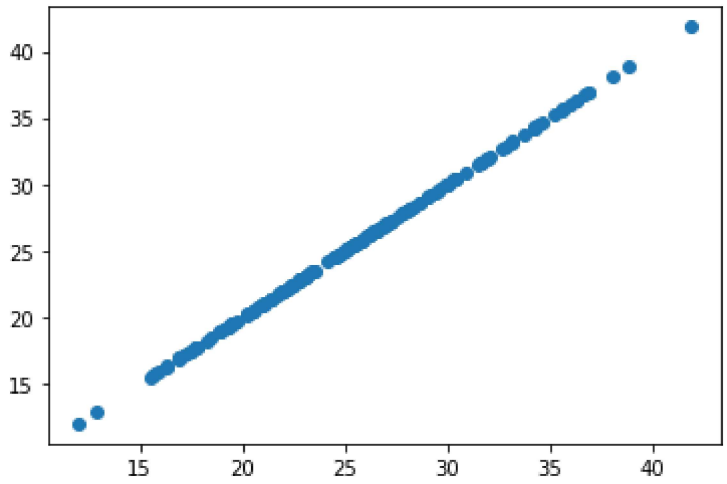
In [14]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[14]:

	Co-efficient
id	7.663738e-18
radius_mean	3.426648e-08
texture_mean	8.518972e-10
perimeter_mean	-5.866259e-09
area_mean	2.549629e-11
smoothness_mean	-3.795397e-08
compactness_mean	3.142413e-07
concavity_mean	3.195433e-08
concave points_mean	1.103615e-07
symmetry_mean	-1.967653e-07
fractal_dimension_mean	1.276786e-07
radius_se	-1.271730e-07
texture_se	1.399937e-08
perimeter_se	1.827812e-08
area_se	4.450911e-11
smoothness_se	-2.192070e-07
compactness_se	-2.963036e-07
concavity_se	-4.623420e-07
concave points_se	-2.875040e-07
symmetry_se	2.202751e-08
fractal_dimension_se	-3.701710e-08
radius_worst	1.815147e-08
texture_worst	1.000000e+00
perimeter_worst	-2.589576e-09
area_worst	-2.400441e-12
smoothness_worst	-2.886599e-07
compactness_worst	1.750133e-07
concavity_worst	2.712796e-09
concave points_worst	-7.695251e-08
symmetry_worst	-2.016974e-07
fractal_dimension_worst	2.204833e-07

In [15]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[15]:
```
<matplotlib.collections.PathCollection at 0x1b4de4f6310>
```



In [16]:
```
print(lr.score(x_test,y_test))
```

```
1.0
```

In [17]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[17]:
```
ElasticNet()
```

In [18]:
```
print(en.coef_)
```

```
[ 5.83780327e-12  0.00000000e+00  3.71107900e-02  0.00000000e+00
 -3.47609022e-04 -0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00  0.00000000e+00 -0.00000000e+00
  0.00000000e+00 -0.00000000e+00 -8.15778127e-04 -0.00000000e+00
  0.00000000e+00  0.00000000e+00 -0.00000000e+00 -0.00000000e+00
 -0.00000000e+00  0.00000000e+00  9.46392629e-01  0.00000000e+00
  3.73266115e-04  0.00000000e+00  0.00000000e+00  0.00000000e+00
  0.00000000e+00  0.00000000e+00  0.00000000e+00]
```

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

```
0.5975145383094116
```

In [20]:
```
print(en.predict(x_train))
```

```
[33.0570086  30.57523772 26.13542445 25.58872593 29.47586415 33.06428386
 23.17117916 24.3683706  25.44371043 22.95282544 20.71765082 19.31687688
 26.45901023 27.90497831 31.84807233 16.65025278 21.11599906 30.98747683
 23.22469265 33.64264288 20.07916812 17.11037099 25.16847888 42.30385313
 25.25010849 17.15560201 24.18134283 26.43399852 33.76219524 28.63045857
 32.5856129  34.01216921 28.12027494 25.6466281  20.01136147 31.00953919
 27.21477628 31.51817001 28.8956953  21.88646819 15.9713809  30.55980005
 15.88835126 30.29186749 34.44451282 26.57832697 31.28193662 25.20555403
 24.02268158 20.13897622 23.60383578 18.46375721 28.98268857 33.43671379
 38.91836186 21.07657466 23.60195558 30.10839197 22.07134844 19.3431609
 32.40951777 19.93367835 31.49696144 31.76154552 31.37620211 20.89553308
 19.38941435 28.05362555 25.9299481  25.22919256 17.34914652 21.19599771
 16.16083342 19.5343959  19.94771023 26.05840212 17.91947354 24.43109687
 20.57665532 16.05627965 25.36144244 24.38470112 28.5920795  26.4892072
 25.92315824 24.32518574 22.56502517 23.04901655 20.89274747 26.29266932
 35.34479144 27.92620421 28.80001689 34.65020779 31.99144061 25.31326238
 31.49087515 33.70104795 31.7755163  26.77245119 30.46762957 20.39576421
 22.27110867 29.12130482 30.0280499  19.70577313 24.04737013 27.76149086
 27.08323694 21.88729939 15.09229935 27.2568122  29.51161686 26.95632496
 33.02766127 18.58680115 33.66746977 30.56535114 19.77221908 36.85610524
 21.80817334 21.65448558 22.98556418 23.62845284 23.94442211 23.03165295
 30.1305111  31.58865459 33.87568386 22.39417275 28.32757382 31.58476573
 28.08892055 29.0575929  17.57754254 23.42355607 31.60886844 27.33522477
 37.94560433 33.20393086 27.27824564 17.17377294 34.88436772 19.3387123
 24.59473985 27.78911395 25.29990924 24.84852755 31.68773806 27.16295459
 17.48273924 44.68595598 18.57083219 22.17139673 22.96316689 47.00967945
 19.40739334 21.56453733 21.64091968 28.78500521 17.06462445 24.92803908
 24.99196863 30.89623166 33.54531944 20.35812356 17.08748945 28.84528339
 20.78903103 25.0824555  26.98121004 24.23056364 21.51278086 23.11139544
 25.29758129 20.45030958 14.40357814 30.30691489 25.54327006 17.93792398
 36.84955671 15.64259519 24.79463739 20.11344095 31.77676062 31.33032601
 35.66700269 37.97241182 31.54226072 30.2718779  24.37847765 30.6001011
 28.6390718  20.79163242 28.32014708 24.66770451 26.06217744 18.50326742
 27.27675615 27.85045222 19.90160597 25.86775098 20.60534428 21.59860879
 30.81250813 30.9244086  35.20874805 32.28792963 18.17140904 18.18596807
 29.22928849 26.05728108 20.87758572 28.24521358 28.35375297 25.48925937
 27.94709144 26.59733726 35.00443642 25.80516942 16.22287632 31.77484619
 19.83028367 29.35052272 23.74577268 19.2386678  25.00376684 31.16439285
 18.39821821 21.5802639  16.10365871 30.93404722 23.02710702 23.38845014
 22.95243953 26.52622862 29.73368198 20.98259456 31.29612758 31.33599167
 32.15358943 17.97860675 21.82980431 25.5550711  27.71018657 18.50680992
 21.23279254 19.83841984 24.36338899 16.78548729 22.76777836 16.966544
 22.18690623 16.61111468 31.92200784 29.00373154 22.54588824 23.61856172
 30.21689248 27.22771276 17.81731714 27.76338556 17.92789522 30.00368491
 25.76220554 25.63469197 17.28406257 41.42173321 27.81952024 29.01601539
 24.2125601  31.04057256 22.08722631 27.68879602 26.22918249 19.78110181
 19.77178646 24.22126913 21.23404997 14.50273785 41.16309685 25.76115587
 30.27633916 19.60904623 16.02580295 21.89419929 17.81086743 29.08050298
 17.61797878 36.67643003 27.77741462 23.08037956 25.24682317 33.28687077
 19.51896342 25.41991645 37.5249139  19.5996549  34.16786462 39.16243264
 24.82381934 37.18938225 19.49597677 18.15666724 20.70159801 33.52636274
 21.82348523 26.36676602 23.80646439 19.34343621 21.74699047 23.88851348
 21.49139567 19.19887572 26.71330536 16.72535671 31.99256936 23.94496523
 40.06998802 15.62728597 30.82073755 19.11420386 32.81144336 20.46353404
 19.9937165  20.14864188 26.51802668 29.42872126 23.01467109 17.33903665
 31.44928142 21.66677932 45.1885059  28.65491212 34.67067779 26.67080153
 22.69615124 24.07623142 26.37479974 22.05948556 18.5255291  21.88705894
```

```
16.38481121 26.01928842 30.63363319 32.73566886 23.18384833 15.88925469
35.75132172 18.33562887 25.35901977 27.97267222 23.17606163 23.18524636
25.11503025 32.60312074 22.52303211 31.73674    28.19221284 18.3845345
33.08309894 28.04917018 23.45066633 21.00640371 18.01328575 36.68909391
22.46930269 17.78204661 38.86035751 24.96792084 16.41927714 16.58630294
33.1702551  30.44113873 25.52281767 35.5246863  27.63991878 23.22476729
40.0416737  24.78822491 19.33108379 38.27107203 26.94880715 26.29213319
27.77625124 18.47547229 34.52821385 24.3896652  25.51063593 27.59838291
28.26863943 17.26880751 25.13798062 28.33877876 12.82899529 24.45440735
21.43933997 48.8363787 ]
```

In [21]:
```
print(en.score(x_train,y_train))
```

```
0.9991945568883723
```

In [22]:
```
from sklearn import metrics
```

In [23]:
```
print("Mean Absolytre Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolytre Error: 1.3551108035562188e-08
```

In [24]:
```
print("Mean Square Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Square Error: 2.901890509692446e-16
```

In [25]:
```
print("Root Mean Square Error:",np.sqrt(metrics.mean_absolute_error(y_test,pre
```

```
Root Mean Square Error: 0.00011640922659120362
```

In [ ]:
```

```