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

In [57]:
```
df=pd.read_csv(r"C:\Users\Admin\Downloads\8_BreastCancerPrediction - 8_BreastCancerPredictio
df
```

Out[57]:

ırst	texture_worst	perimeter_worst	area_worst	smoothness_worst	compactness_worst	concavity_worst	conca points_wo
380	17.33	184.60	2019.0	0.16220	0.66560	0.7119	0.26
090	23.41	158.80	1956.0	0.12380	0.18660	0.2416	0.18
570	25.53	152.50	1709.0	0.14440	0.42450	0.4504	0.24
910	26.50	98.87	567.7	0.20980	0.86630	0.6869	0.25
540	16.67	152.20	1575.0	0.13740	0.20500	0.4000	0.16
...	...	...	...	...	...	...	
450	26.40	166.10	2027.0	0.14100	0.21130	0.4107	0.22
590	38.25	155.00	1731.0	0.11660	0.19220	0.3215	0.16
980	34.12	126.70	1124.0	0.11390	0.30940	0.3403	0.14
740	39.42	184.60	1821.0	0.16500	0.86810	0.9387	0.26
456	30.37	59.16	268.6	0.08996	0.06444	0.0000	0.00

In [58]:
```
df.head()
```

Out[58]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0

5 rows × 32 columns

In [59]:
```
df.describe()
```

Out[59]:

st	texture_worst	perimeter_worst	area_worst	smoothness_worst	compactness_worst	concavity_worst	conca points_wor
)0	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.0000
)0	25.677223	107.261213	880.583128	0.132369	0.254265	0.272188	0.1146
42	6.146258	33.602542	569.356993	0.022832	0.157336	0.208624	0.0657
)0	12.020000	50.410000	185.200000	0.071170	0.027290	0.000000	0.0000
)0	21.080000	84.110000	515.300000	0.116600	0.147200	0.114500	0.0649
)0	25.410000	97.660000	686.500000	0.131300	0.211900	0.226700	0.0999
)0	29.720000	125.400000	1084.000000	0.146000	0.339100	0.382900	0.1614
)0	49.540000	251.200000	4254.000000	0.222600	1.058000	1.252000	0.2910

In [60]:
```
df.info()
```

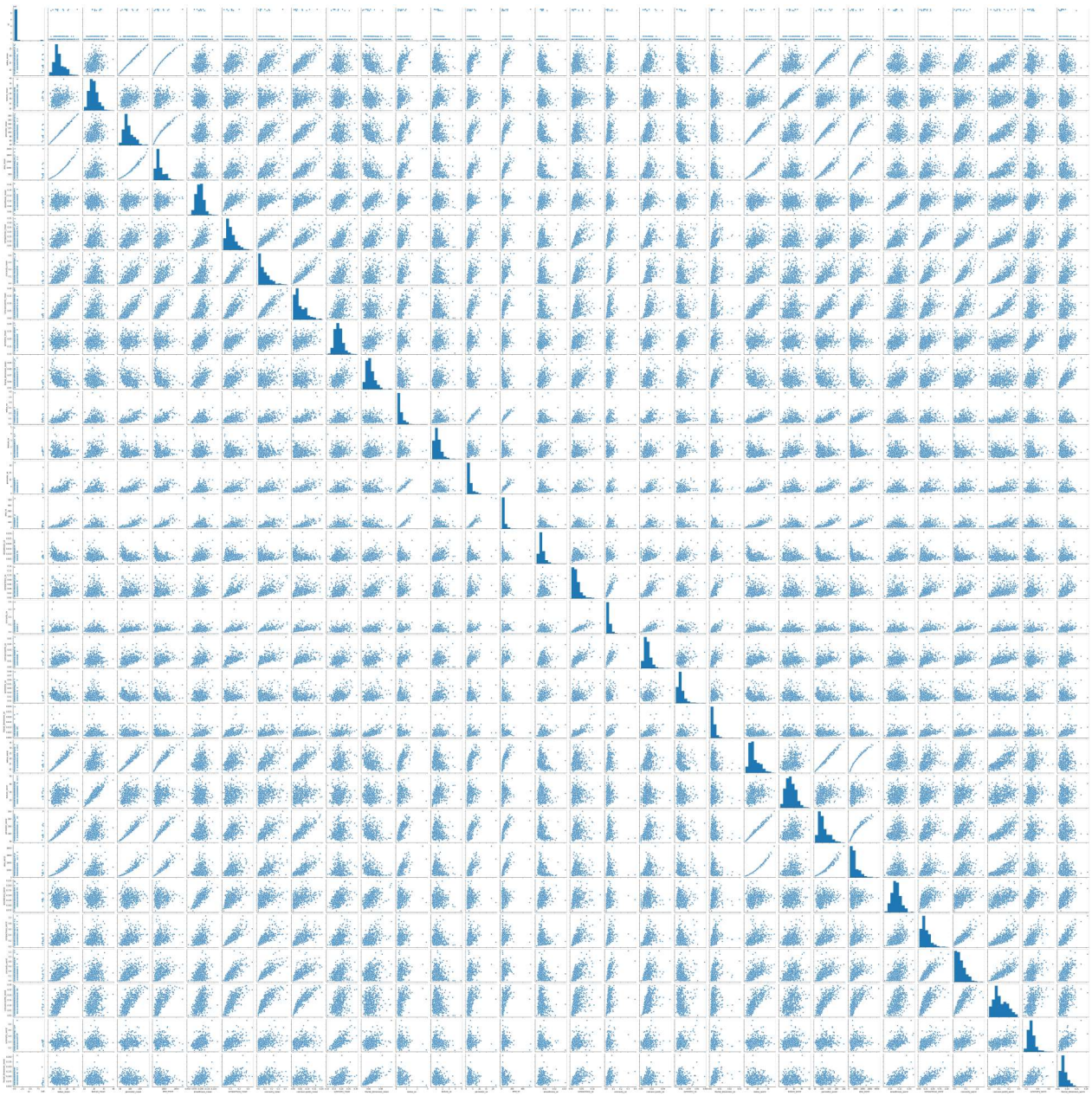
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   id                       569 non-null    int64  
 1   diagnosis                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
dtypes: float64(30), int64(1), object(1)
memory usage: 142.4+ KB
```

In [61]:
```
df.columns
```

Out[61]:
```
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')
```

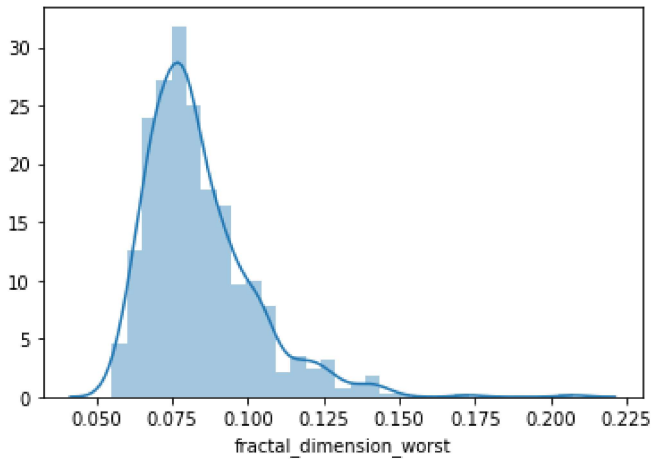
In [62]:
```
sns.pairplot(df)
```

Out[62]:
```
<seaborn.axisgrid.PairGrid at 0x1f68f9116a0>
```



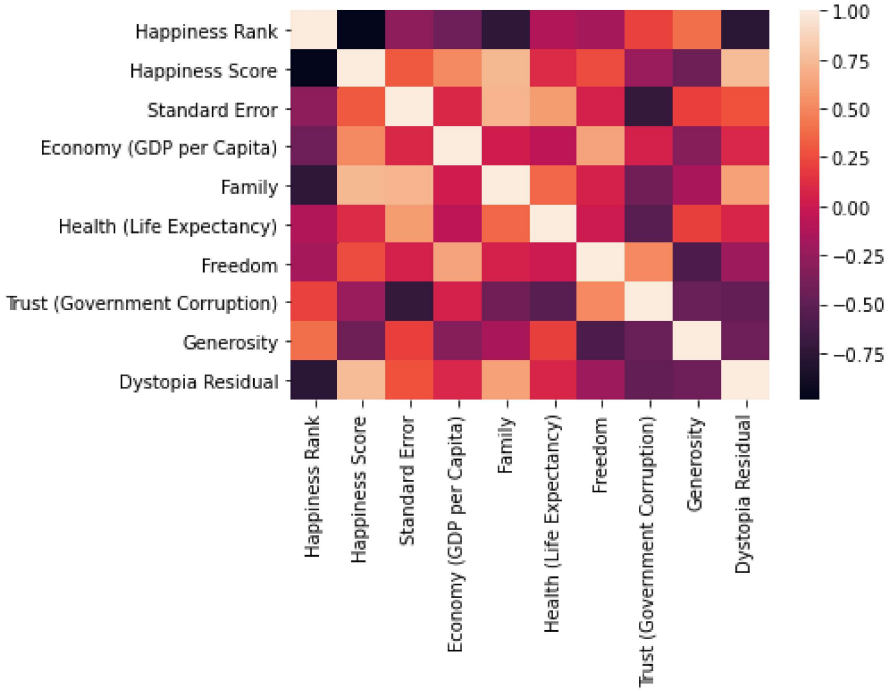
In [65]:
```
sns.distplot(df['fractal_dimension_worst'])
```

Out[65]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1f6af908f40>
```



In [67]:
```
sns.heatmap(df1.corr())
```

Out[67]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1f6b2988fa0>
```



In [68]:
```
x=df2[['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']]
y=df2['fractal_dimension_worst']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-68-74cd4312818b> in <module>
----> 1 x=df2[['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
      2        'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
      3        'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
      4        'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
      5        'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',

~\anaconda3\lib\site-packages\pandas\core\frame.py in __getitem__(self, key)
   2804             if is_iterator(key):
   2805                 key = list(key)
-> 2806             indexer = self.loc._get_listlike_indexer(key, axis=1, raise_missing=Tru
e)[1]
   2807
   2808         # take() does not accept boolean indexers

~\anaconda3\lib\site-packages\pandas\core\indexing.py in _get_listlike_indexer(self, key, a
xis, raise_missing)
   1550             keyarr, indexer, new_indexer = ax._reindex_non_unique(keyarr)
   1551
-> 1552         self._validate_read_indexer(
   1553             keyarr, indexer, o._get_axis_number(axis), raise_missing=raise_missing
   1554         )

~\anaconda3\lib\site-packages\pandas\core\indexing.py in _validate_read_indexer(self, key,
indexer, axis, raise_missing)
   1638             if missing == len(indexer):
   1639                 axis_name = self.obj._get_axis_name(axis)
-> 1640                 raise KeyError(f"None of [{key}] are in the [{axis_name}]")
   1641
   1642             # We (temporarily) allow for some missing keys with .loc, except in

KeyError: "None of [Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mea
n',\n       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',\n
'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',\n       'radius_se', 'tex
ture_se', 'perimeter_se', 'area_se', 'smoothness_se',\n       'compactness_se', 'concavity_
se', 'concave points_se', 'symmetry_se',\n       'fractal_dimension_se', 'radius_worst', 't
exture_worst',\n       'perimeter_worst', 'area_worst', 'smoothness_worst',\n       'compac
tness_worst', 'concavity_worst', 'concave points_worst',\n       'symmetry_worst'],\n
dtype='object')] are in the [columns]"
```

In [69]:
```
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 [70]:
```
from sklearn.linear_model import LinearRegression
lr= LinearRegression()
lr.fit(x_train,y_train)
```

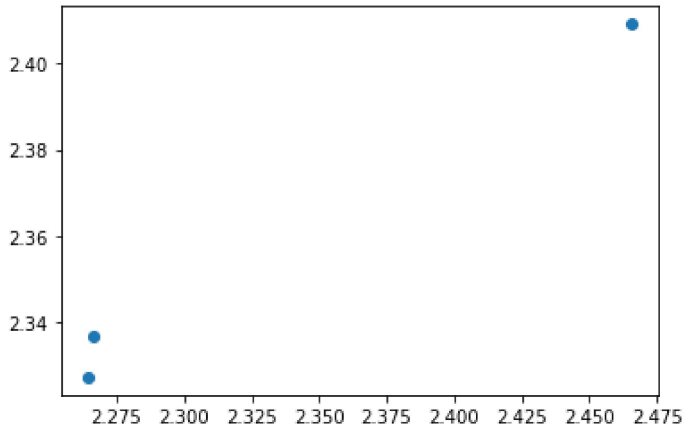
Out[70]:
```
LinearRegression()
```

In [71]:
```
print(lr.intercept_)
```

```
5.304895380269695
```

In [72]:
```
prediction= lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[72]: <matplotlib.collections.PathCollection at 0x1f6b41e7a00>



In [73]:
```
print(lr.score(x_test,y_test))
```

```
0.5481411348778482
```

In [ ]: