In [1]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```python
df = pd.read_csv('cancer_classification.csv')
df
```

Out[2]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	worst area	worst smoothness	wor compactnes
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	0.07871	...	17.33	184.60	2019.0	0.16220	0.665
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	0.05667	...	23.41	158.80	1956.0	0.12380	0.186
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	0.05999	...	25.53	152.50	1709.0	0.14440	0.424
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	0.09744	...	26.50	98.87	567.7	0.20980	0.866
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	0.05883	...	16.67	152.20	1575.0	0.13740	0.205
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623	...	26.40	166.10	2027.0	0.14100	0.211
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533	...	38.25	155.00	1731.0	0.11660	0.192

In [3]:

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   mean radius              569 non-null    float64
 1   mean texture             569 non-null    float64
 2   mean perimeter           569 non-null    float64
 3   mean area                569 non-null    float64
 4   mean smoothness          569 non-null    float64
 5   mean compactness         569 non-null    float64
 6   mean concavity           569 non-null    float64
 7   mean concave points      569 non-null    float64
 8   mean symmetry            569 non-null    float64
 9   mean fractal dimension   569 non-null    float64
 10  radius error             569 non-null    float64
 11  texture error            569 non-null    float64
 12  perimeter error          569 non-null    float64
 13  area error               569 non-null    float64
 14  smoothness error         569 non-null    float64
 15  compactness error        569 non-null    float64
 16  concavity error          569 non-null    float64
 17  concave points error     569 non-null    float64
 18  symmetry error           569 non-null    float64
 19  fractal dimension error  569 non-null    float64
 20  worst radius             569 non-null    float64
 21  worst texture            569 non-null    float64
 22  worst perimeter          569 non-null    float64
 23  worst area               569 non-null    float64
 24  worst smoothness         569 non-null    float64
 25  worst compactness        569 non-null    float64
 26  worst concavity          569 non-null    float64
 27  worst concave points     569 non-null    float64
 28  worst symmetry           569 non-null    float64
 29  worst fractal dimension  569 non-null    float64
 30  benign_0__mal_1          569 non-null    int64  
dtypes: float64(30), int64(1)
memory usage: 137.9 KB
```

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   mean radius              569 non-null    float64
 1   mean texture             569 non-null    float64
 2   mean perimeter           569 non-null    float64
 3   mean area                569 non-null    float64
 4   mean smoothness          569 non-null    float64
 5   mean compactness         569 non-null    float64
 6   mean concavity           569 non-null    float64
 7   mean concave points      569 non-null    float64
 8   mean symmetry            569 non-null    float64
 9   mean fractal dimension   569 non-null    float64
 10  radius error             569 non-null    float64
 11  texture error            569 non-null    float64
 12  perimeter error          569 non-null    float64
 13  area error               569 non-null    float64
 14  smoothness error         569 non-null    float64
 15  compactness error        569 non-null    float64
 16  concavity error          569 non-null    float64
 17  concave points error     569 non-null    float64
 18  symmetry error           569 non-null    float64
 19  fractal dimension error  569 non-null    float64
 20  worst radius             569 non-null    float64
 21  worst texture            569 non-null    float64
 22  worst perimeter          569 non-null    float64
 23  worst area               569 non-null    float64
 24  worst smoothness         569 non-null    float64
 25  worst compactness        569 non-null    float64
 26  worst concavity          569 non-null    float64
 27  worst concave points     569 non-null    float64
 28  worst symmetry           569 non-null    float64
 29  worst fractal dimension  569 non-null    float64
 30  benign_0__mal_1          569 non-null    int64
dtypes: float64(30), int64(1)
memory usage: 137.9 KB
```

In [5]:

```
df.describe()
```

Out[5]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	p
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	...	569.000000	56
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0.181162	0.062798	...	25.677223	10
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414	0.007060	...	6.146258	3
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	0.049960	...	12.020000	5
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900	0.057700	...	21.080000	8
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200	0.061540	...	25.410000	9
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700	0.066120	...	29.720000	12
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	0.097440	...	49.540000	25

8 rows × 31 columns

In [6]:

```
df.describe().transpose()
```

Out[6]:

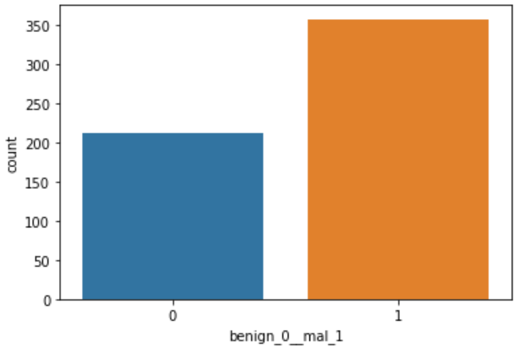
	count	mean	std	min	25%	50%	75%	max
mean radius	569.0	14.127292	3.524049	6.981000	11.700000	13.370000	15.780000	28.11000
mean texture	569.0	19.289649	4.301036	9.710000	16.170000	18.840000	21.800000	39.28000
mean perimeter	569.0	91.969033	24.298981	43.790000	75.170000	86.240000	104.100000	188.50000
mean area	569.0	654.889104	351.914129	143.500000	420.300000	551.100000	782.700000	2501.00000
mean smoothness	569.0	0.096360	0.014064	0.052630	0.086370	0.095870	0.105300	0.16340
mean compactness	569.0	0.104341	0.052813	0.019380	0.064920	0.092630	0.130400	0.34540
mean concavity	569.0	0.088799	0.079720	0.000000	0.029560	0.061540	0.130700	0.42680
mean concave points	569.0	0.048919	0.038803	0.000000	0.020310	0.033500	0.074000	0.20120
mean symmetry	569.0	0.181162	0.027414	0.106000	0.161900	0.179200	0.195700	0.30400
mean fractal dimension	569.0	0.062798	0.007060	0.049960	0.057700	0.061540	0.066120	0.09744
radius error	569.0	0.405172	0.277313	0.111500	0.232400	0.324200	0.478900	2.87300
texture error	569.0	1.216853	0.551648	0.360200	0.833900	1.108000	1.474000	4.88500
perimeter error	569.0	2.866059	2.021855	0.757000	1.606000	2.287000	3.357000	21.98000
area error	569.0	40.337079	45.491006	6.802000	17.850000	24.530000	45.190000	542.20000
smoothness error	569.0	0.007041	0.003003	0.001713	0.005169	0.006380	0.008146	0.03113
compactness error	569.0	0.025478	0.017908	0.002252	0.013080	0.020450	0.032450	0.13540
concavity error	569.0	0.031894	0.030186	0.000000	0.015090	0.025890	0.042050	0.39600
concave points error	569.0	0.011796	0.006170	0.000000	0.007638	0.010930	0.014710	0.05279
symmetry error	569.0	0.020542	0.008266	0.007882	0.015160	0.018730	0.023480	0.07895
fractal dimension error	569.0	0.003795	0.002646	0.000895	0.002248	0.003187	0.004558	0.02984
worst radius	569.0	16.269190	4.833242	7.930000	13.010000	14.970000	18.790000	36.04000
worst texture	569.0	25.677223	6.146258	12.020000	21.080000	25.410000	29.720000	49.54000
worst perimeter	569.0	107.261213	33.602542	50.410000	84.110000	97.660000	125.400000	251.20000
worst area	569.0	880.583128	569.356993	185.200000	515.300000	686.500000	1084.000000	4254.00000
worst smoothness	569.0	0.132369	0.022832	0.071170	0.116600	0.131300	0.146000	0.22260
worst compactness	569.0	0.254265	0.157336	0.027290	0.147200	0.211900	0.339100	1.05800
worst concavity	569.0	0.272188	0.208624	0.000000	0.114500	0.226700	0.382900	1.25200
worst concave points	569.0	0.114606	0.065732	0.000000	0.064930	0.099930	0.161400	0.29100
worst symmetry	569.0	0.290076	0.061867	0.156500	0.250400	0.282200	0.317900	0.66380
worst fractal dimension	569.0	0.083946	0.018061	0.055040	0.071460	0.080040	0.092080	0.20750
benign_0__mal_1	569.0	0.627417	0.483918	0.000000	0.000000	1.000000	1.000000	1.00000

In [7]:

```
sns.countplot(x='benign_0__mal_1',data=df)
```

Out[7]:

<AxesSubplot:xlabel='benign_0__mal_1', ylabel='count'>



In [8]:

```
df.corr()
```

Out[8]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	w perim
mean radius	1.000000	0.323782	0.997855	0.987357	0.170581	0.506124	0.676764	0.822529	0.147741	-0.311631	...	0.297008	0.96
mean texture	0.323782	1.000000	0.329533	0.321086	-0.023389	0.236702	0.302418	0.293464	0.071401	-0.076437	...	0.912045	0.35
mean perimeter	0.997855	0.329533	1.000000	0.986507	0.207278	0.556936	0.716136	0.850977	0.183027	-0.261477	...	0.303038	0.97
mean area	0.987357	0.321086	0.986507	1.000000	0.177028	0.498502	0.685983	0.823269	0.151293	-0.283110	...	0.287489	0.95
mean smoothness	0.170581	-0.023389	0.207278	0.177028	1.000000	0.659123	0.521984	0.553695	0.557775	0.584792	...	0.036072	0.23
mean compactness	0.506124	0.236702	0.556936	0.498502	0.659123	1.000000	0.883121	0.831135	0.602641	0.565369	...	0.248133	0.59
mean concavity	0.676764	0.302418	0.716136	0.685983	0.521984	0.883121	1.000000	0.921391	0.500667	0.336783	...	0.299879	0.72
mean concave points	0.822529	0.293464	0.850977	0.823269	0.553695	0.831135	0.921391	1.000000	0.462497	0.166917	...	0.292752	0.85
mean symmetry	0.147741	0.071401	0.183027	0.151293	0.557775	0.602641	0.500667	0.462497	1.000000	0.479921	...	0.090651	0.21
mean fractal dimension	-0.311631	-0.076437	-0.261477	-0.283110	0.584792	0.565369	0.336783	0.166917	0.479921	1.000000	...	-0.051269	-0.20
radius error	0.679090	0.275869	0.691765	0.732562	0.301467	0.497473	0.631925	0.698050	0.303379	0.000111	...	0.194799	0.71
texture error	-0.097317	0.386358	-0.086761	-0.066280	0.068406	0.046205	0.076218	0.021480	0.128053	0.164174	...	0.409003	-0.10
perimeter error	0.674172	0.281673	0.693135	0.726628	0.296092	0.548905	0.660391	0.710650	0.313893	0.039830	...	0.200371	0.72
area error	0.735864	0.259845	0.744983	0.800086	0.246552	0.455653	0.617427	0.690299	0.223970	-0.090170	...	0.196497	0.76
smoothness error	-0.222600	0.006614	-0.202694	-0.166777	0.332375	0.135299	0.098564	0.027653	0.187321	0.401964	...	-0.074743	-0.21
compactness error	0.206000	0.191975	0.250744	0.212583	0.318943	0.738722	0.670279	0.490424	0.421659	0.559837	...	0.143003	0.26
concavity error	0.194204	0.143293	0.228082	0.207660	0.248396	0.570517	0.691270	0.439167	0.342627	0.446630	...	0.100241	0.22
concave points error	0.376169	0.163851	0.407217	0.372320	0.380676	0.642262	0.683260	0.615634	0.393298	0.341198	...	0.086741	0.39
symmetry error	-0.104321	0.009127	-0.081629	-0.072497	0.200774	0.229977	0.178009	0.095351	0.449137	0.345007	...	-0.077473	-0.10
fractal dimension error	-0.042641	0.054458	-0.005523	-0.019887	0.283607	0.507318	0.449301	0.257584	0.331786	0.688132	...	-0.003195	-0.00
worst radius	0.969539	0.352573	0.969476	0.962746	0.213120	0.535315	0.688236	0.830318	0.185728	-0.253691	...	0.359921	0.99
worst texture	0.297008	0.912045	0.303038	0.287489	0.036072	0.248133	0.299879	0.292752	0.090651	-0.051269	...	1.000000	0.36
worst perimeter	0.965137	0.358040	0.970387	0.959120	0.238853	0.590210	0.729565	0.855923	0.219169	-0.205151	...	0.365098	1.00
worst area	0.941082	0.343546	0.941550	0.959213	0.206718	0.509604	0.675987	0.809630	0.177193	-0.231854	...	0.345842	0.97
worst smoothness	0.119616	0.077503	0.150549	0.123523	0.805324	0.565541	0.448822	0.452753	0.426675	0.504942	...	0.225429	0.23
worst compactness	0.413463	0.277830	0.455774	0.390410	0.472468	0.865809	0.754968	0.667454	0.473200	0.458798	...	0.360832	0.52
worst concavity	0.526911	0.301025	0.563879	0.512606	0.434926	0.816275	0.884103	0.752399	0.433721	0.346234	...	0.368366	0.61
worst concave points	0.744214	0.295316	0.771241	0.722017	0.503053	0.815573	0.861323	0.910155	0.430297	0.175325	...	0.359755	0.81
worst symmetry	0.163953	0.105008	0.189115	0.143570	0.394309	0.510223	0.409464	0.375744	0.699826	0.334019	...	0.233027	0.26
worst fractal dimension	0.007066	0.119205	0.051019	0.003738	0.499316	0.687382	0.514930	0.368661	0.438413	0.767297	...	0.219122	0.13
benign_0__mal_1	-0.730029	-0.415185	-0.742636	-0.708984	-0.358560	-0.596534	-0.696360	-0.776614	-0.330499	0.012838	...	-0.456903	-0.78

31 rows × 31 columns

In [9]:

```
df.corr()['benign_0__mal_1']
```

Out[9]:

```
mean radius               -0.730029
mean texture              -0.415185
mean perimeter            -0.742636
mean area                 -0.708984
mean smoothness           -0.358560
mean compactness          -0.596534
mean concavity            -0.696360
mean concave points       -0.776614
mean symmetry             -0.330499
mean fractal dimension     0.012838
radius error              -0.567134
texture error              0.008303
perimeter error           -0.556141
area error                -0.548236
smoothness error           0.067016
compactness error         -0.292999
concavity error           -0.253730
concave points error      -0.408042
symmetry error             0.006522
fractal dimension error   -0.077972
worst radius              -0.776454
worst texture             -0.456903
worst perimeter           -0.782914
worst area                -0.733825
worst smoothness          -0.421465
worst compactness         -0.590998
worst concavity           -0.659610
worst concave points      -0.793566
worst symmetry            -0.416294
worst fractal dimension   -0.323872
benign_0__mal_1            1.000000
Name: benign_0__mal_1, dtype: float64
```

In [10]:

```
df.corr()['benign_0__mal_1'].sort_values()
```

Out[10]:

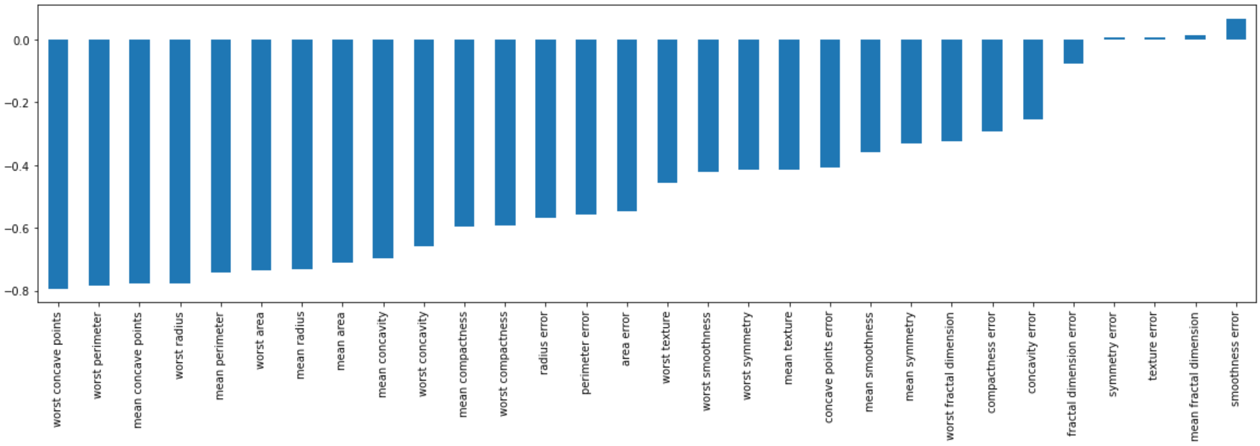
```
worst concave points      -0.793566
worst perimeter           -0.782914
mean concave points       -0.776614
worst radius              -0.776454
mean perimeter            -0.742636
worst area                -0.733825
mean radius               -0.730029
mean area                 -0.708984
mean concavity            -0.696360
worst concavity           -0.659610
mean compactness          -0.596534
worst compactness         -0.590998
radius error              -0.567134
perimeter error           -0.556141
area error                -0.548236
worst texture             -0.456903
worst smoothness          -0.421465
worst symmetry            -0.416294
mean texture              -0.415185
concave points error      -0.408042
mean smoothness           -0.358560
mean symmetry             -0.330499
worst fractal dimension   -0.323872
compactness error         -0.292999
concavity error           -0.253730
fractal dimension error   -0.077972
symmetry error             0.006522
texture error              0.008303
mean fractal dimension     0.012838
smoothness error           0.067016
benign_0__mal_1            1.000000
Name: benign_0__mal_1, dtype: float64
```

In [11]:

```
plt.figure(figsize=(20,5))
df.corr()['benign_0__mal_1'][:-1].sort_values().plot(kind='bar')
```

Out[11]:

<AxesSubplot:>

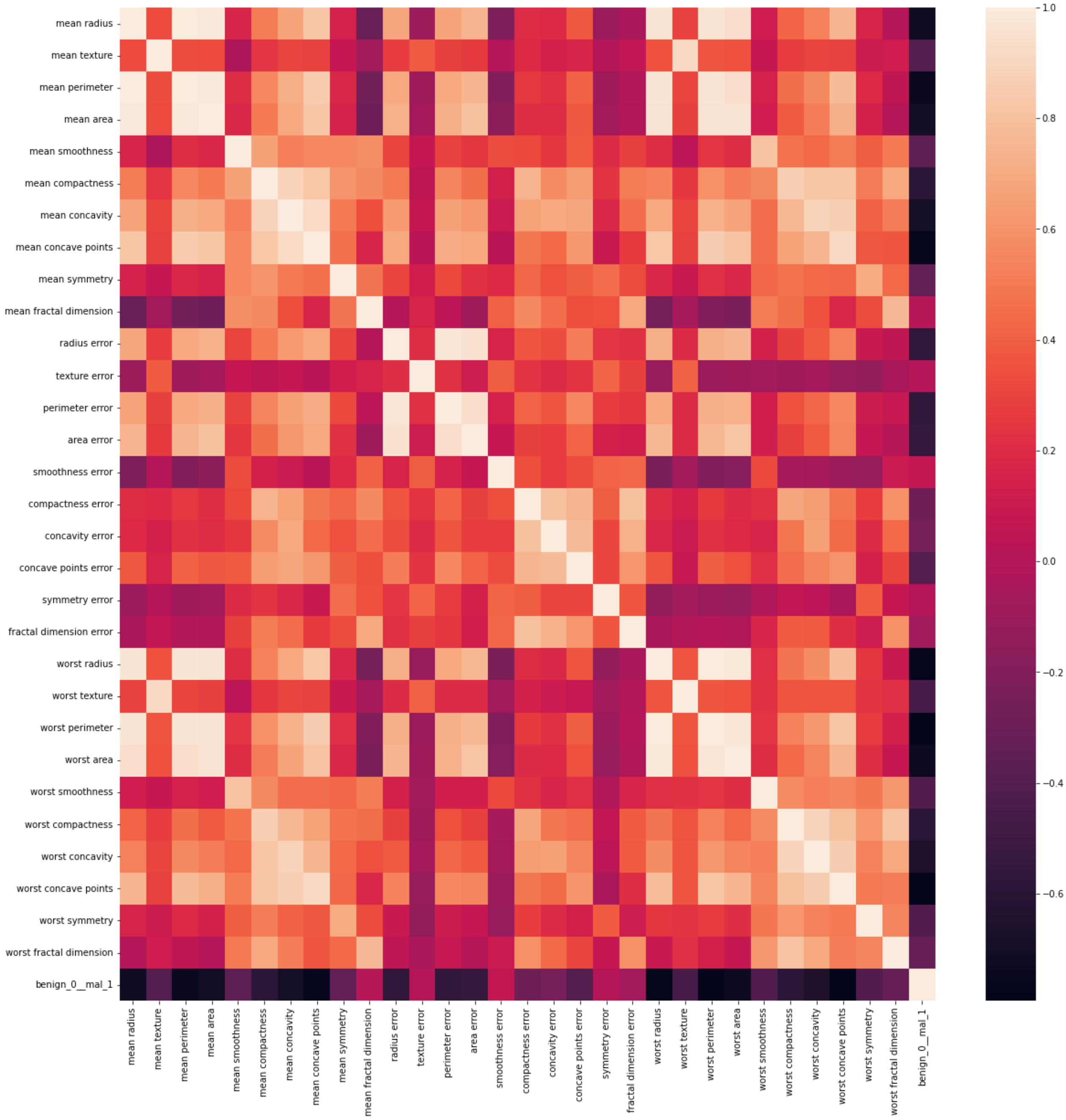


In [12]:

```
plt.figure(figsize=(20,20))
sns.heatmap(df.corr())
```

Out[12]:

<AxesSubplot:>



In [13]:

```
X = df.drop('benign_0__mal_1',axis=1)
X
```

Out[13]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius	worst texture	worst perimeter	worst area	sm
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	0.07871	...	25.380	17.33	184.60	2019.0	
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	0.05667	...	24.990	23.41	158.80	1956.0	
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	0.05999	...	23.570	25.53	152.50	1709.0	
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	0.09744	...	14.910	26.50	98.87	567.7	
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	0.05883	...	22.540	16.67	152.20	1575.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623	...	25.450	26.40	166.10	2027.0	
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533	...	23.690	38.25	155.00	1731.0	
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648	...	18.980	34.12	126.70	1124.0	
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016	...	25.740	39.42	184.60	1821.0	
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05884	...	9.456	30.37	59.16	268.6	

569 rows × 30 columns

In [14]:

```
y = df['benign_0__mal_1']
y
```

Out[14]:

```
0      0
1      0
2      0
3      0
4      0
      ..
564    0
565    0
566    0
567    0
568    1
Name: benign_0__mal_1, Length: 569, dtype: int64
```

In [15]:

```
from sklearn.model_selection import train_test_split
```

In [16]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=101)
```

In [17]:

```
from sklearn.preprocessing import MinMaxScaler
```

In [18]:

```
scaler = MinMaxScaler()
```

In [19]:

```
X_train = scaler.fit_transform(X_train )
```

In [20]:

```
X_test = scaler.transform(X_test)
```

In [21]:

```
X_train.shape
```

Out[21]:

```
(426, 30)
```

In [22]:

```
from tensorflow.keras.models import Sequential
```

In [23]:

```
from tensorflow.keras.layers import Dense,Dropout
```

In [24]:

```
model = Sequential()

model.add(Dense(30,activation='relu'))
model.add(Dense(15,activation='relu'))

#BINARY CLASSIFICATION
model.add(Dense(1,activation='sigmoid'))

model.compile(loss='binary_crossentropy', optimizer='adam')
```

In [25]:

```
model.fit(x=X_train,y=y_train,epochs=600,validation_data=(X_test,y_test))
```

```
Epoch 362/600
14/14 [==============================] - 0s 7ms/step - loss: 0.0267 - val_loss: 0.1339
Epoch 363/600
14/14 [==============================] - 0s 10ms/step - loss: 0.0260 - val_loss: 0.1428
Epoch 364/600
14/14 [==============================] - 0s 8ms/step - loss: 0.0237 - val_loss: 0.1360
Epoch 365/600
14/14 [==============================] - 0s 7ms/step - loss: 0.0243 - val_loss: 0.1391
Epoch 366/600
14/14 [==============================] - 0s 7ms/step - loss: 0.0250 - val_loss: 0.1382
Epoch 367/600
14/14 [==============================] - 0s 6ms/step - loss: 0.0268 - val_loss: 0.1411
Epoch 368/600
14/14 [==============================] - 0s 7ms/step - loss: 0.0249 - val_loss: 0.1411
Epoch 369/600
14/14 [==============================] - 0s 7ms/step - loss: 0.0272 - val_loss: 0.1295
Epoch 370/600
14/14 [==============================] - 0s 7ms/step - loss: 0.0289 - val_loss: 0.1454
Epoch 371/600
14/14 [==============================] - 0s 7ms/step - loss: 0.0249 - val_loss: 0.1428
```

In [26]:

```
model.history.history
```

```
  0.05232915282249451,
  0.05275832861661911,
  0.05383510887622833,
  0.051781561225652695,
  0.053157903254032135,
  0.05184987187385559,
  0.050306033343076706,
  0.05028820410370827,
  0.05361328646540642,
  0.04998074471950531,
  0.05622515454888344,
  0.05313117429614067,
  0.05243847146630287,
  0.049958933144807816,
  0.05005553364753723,
  0.04976627603173256,
  0.049546558409929276,
  0.048712220042943954,
  0.049473732709884644,
  0.054022736847400665,
```

In [27]:

```
pd.DataFrame(model.history.history)
```

Out[27]:

	loss	val_loss
0	0.702173	0.690970
1	0.673321	0.664181
2	0.650638	0.641579
3	0.626690	0.613711
4	0.597449	0.578630
...	...	...
595	0.010054	0.141733
596	0.009934	0.157385
597	0.009776	0.142729
598	0.009835	0.174191
599	0.011670	0.146801

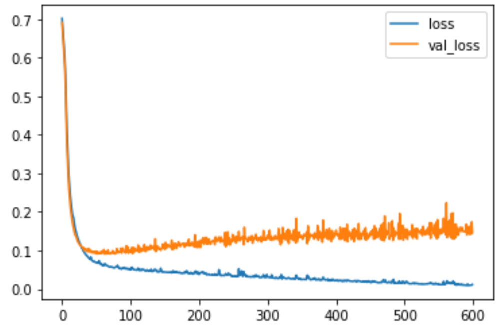
600 rows × 2 columns

In [28]:

```
pd.DataFrame(model.history.history).plot()
```

Out[28]:

```
<AxesSubplot:>
```



Dealing with Overfitting

In [29]:

```
model = Sequential()

model.add(Dense(30,activation='relu'))
model.add(Dense(15,activation='relu'))

#BINARY CLASSIFICATION
model.add(Dense(1,activation='sigmoid'))

model.compile(loss='binary_crossentropy', optimizer='adam')
```

In [30]:

```
from tensorflow.keras.callbacks import EarlyStopping
```

In [31]:

```
help(EarlyStopping)
```

```
 |           keys are prefixed with `val_`. For training epoch, the values of
 |           the `Model`'s metrics are returned. Example:
 |           `{'loss': 0.2, 'accuracy': 0.7}`.
 |
 |  on_train_begin(self, logs=None)
 |      Called at the beginning of training.
 |
 |      Subclasses should override for any actions to run.
 |
 |      Args:
 |          logs: Dict. Currently no data is passed to this argument for this
 |            method but that may change in the future.
 |
 |  on_train_end(self, logs=None)
 |      Called at the end of training.
 |
 |      Subclasses should override for any actions to run.
 |
 |      Args:
```

In [32]:

```
early_stop = EarlyStopping(monitor='val_loss', mode='min',verbose=2,patience=25)
```

In [33]:

```
model.fit(x=X_train,y=y_train,epochs=600,validation_data=(X_test,y_test),
         callbacks=[early_stop])
```

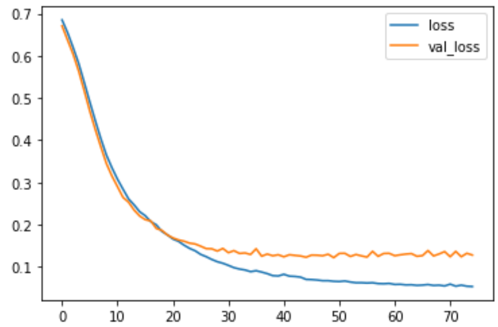
```
Epoch 1/600
14/14 [==============================] - 1s 16ms/step - loss: 0.6852 - val_loss: 0.6711
Epoch 2/600
14/14 [==============================] - 0s 6ms/step - loss: 0.6550 - val_loss: 0.6374
Epoch 3/600
14/14 [==============================] - 0s 5ms/step - loss: 0.6197 - val_loss: 0.6040
Epoch 4/600
14/14 [==============================] - 0s 5ms/step - loss: 0.5821 - val_loss: 0.5638
Epoch 5/600
14/14 [==============================] - 0s 5ms/step - loss: 0.5372 - val_loss: 0.5175
Epoch 6/600
14/14 [==============================] - 0s 5ms/step - loss: 0.4910 - val_loss: 0.4693
Epoch 7/600
14/14 [==============================] - 0s 5ms/step - loss: 0.4463 - val_loss: 0.4253
Epoch 8/600
14/14 [==============================] - 0s 7ms/step - loss: 0.4036 - val_loss: 0.3843
Epoch 9/600
14/14 [==============================] - 0s 4ms/step - loss: 0.3649 - val_loss: 0.3447
Epoch 10/600
```

In [34]:

```
pd.DataFrame(model.history.history).plot()
```

Out[34]:

```
<AxesSubplot:>
```



adding dropout layers

In [35]:

```
model = Sequential()

model.add(Dense(30,activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(15,activation='relu'))
model.add(Dropout(0.5))

#BINARY CLASSIFICATION
model.add(Dense(1,activation='sigmoid'))

model.compile(loss='binary_crossentropy', optimizer='adam')
```

In [36]:

```
model.fit(x=X_train,y=y_train,epochs=600,validation_data=(X_test,y_test),
         callbacks=[early_stop])
```

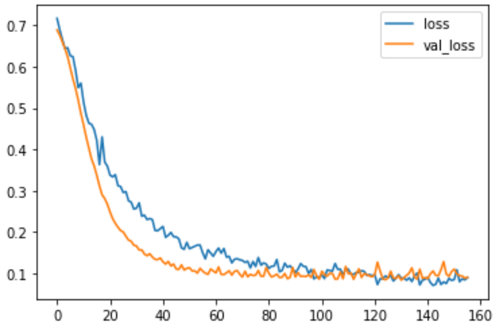
```
Epoch 1/600
14/14 [==============================] - 1s 17ms/step - loss: 0.7168 - val_loss: 0.6889
Epoch 2/600
14/14 [==============================] - 0s 5ms/step - loss: 0.6897 - val_loss: 0.6766
Epoch 3/600
14/14 [==============================] - 0s 5ms/step - loss: 0.6657 - val_loss: 0.6602
Epoch 4/600
14/14 [==============================] - 0s 6ms/step - loss: 0.6436 - val_loss: 0.6436
Epoch 5/600
14/14 [==============================] - 0s 5ms/step - loss: 0.6458 - val_loss: 0.6238
Epoch 6/600
14/14 [==============================] - 0s 5ms/step - loss: 0.6265 - val_loss: 0.5977
Epoch 7/600
14/14 [==============================] - 0s 5ms/step - loss: 0.6242 - val_loss: 0.5703
Epoch 8/600
14/14 [==============================] - 0s 6ms/step - loss: 0.5912 - val_loss: 0.5471
Epoch 9/600
14/14 [==============================] - 0s 6ms/step - loss: 0.5493 - val_loss: 0.5187
Epoch 10/600
```

In [37]:

```
pd.DataFrame(model.history.history).plot()
```

Out[37]:

```
<AxesSubplot:>
```



In [45]:

```
predictions = (model.predict(X_test) > 0.5).astype("int32")
```

```
5/5 [==============================] - 0s 2ms/step
```

In [46]:

```
predictions.shape
```

Out[46]:

```
(143, 1)
```

In [48]:

```
pd.DataFrame(predictions, columns=['class'])
```

Out[48]:

	class
0	1
1	1
2	1
3	0
4	1
...	...
138	0
139	1
140	0
141	1
142	0

143 rows × 1 columns

In [49]:

```
from sklearn.metrics import classification_report, confusion_matrix
```

In [51]:

```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.98      0.98      0.98        55
           1       0.99      0.99      0.99        88

    accuracy                           0.99       143
   macro avg       0.99      0.99      0.99       143
weighted avg       0.99      0.99      0.99       143
```

In [52]:

```
print(confusion_matrix(y_test,predictions))
```

```
[[54  1]
 [ 1 87]]
```

Done!!