In [35]:
```
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error
```

In [3]:
```
data = pd.read_csv("cancer_classification.csv")
```

In [4]:
```
data
```

Out[4]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809
...	...	...	...	...	...	...	...	...	...
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587

569 rows × 31 columns

In [5]:
```
data.isna().sum()
```

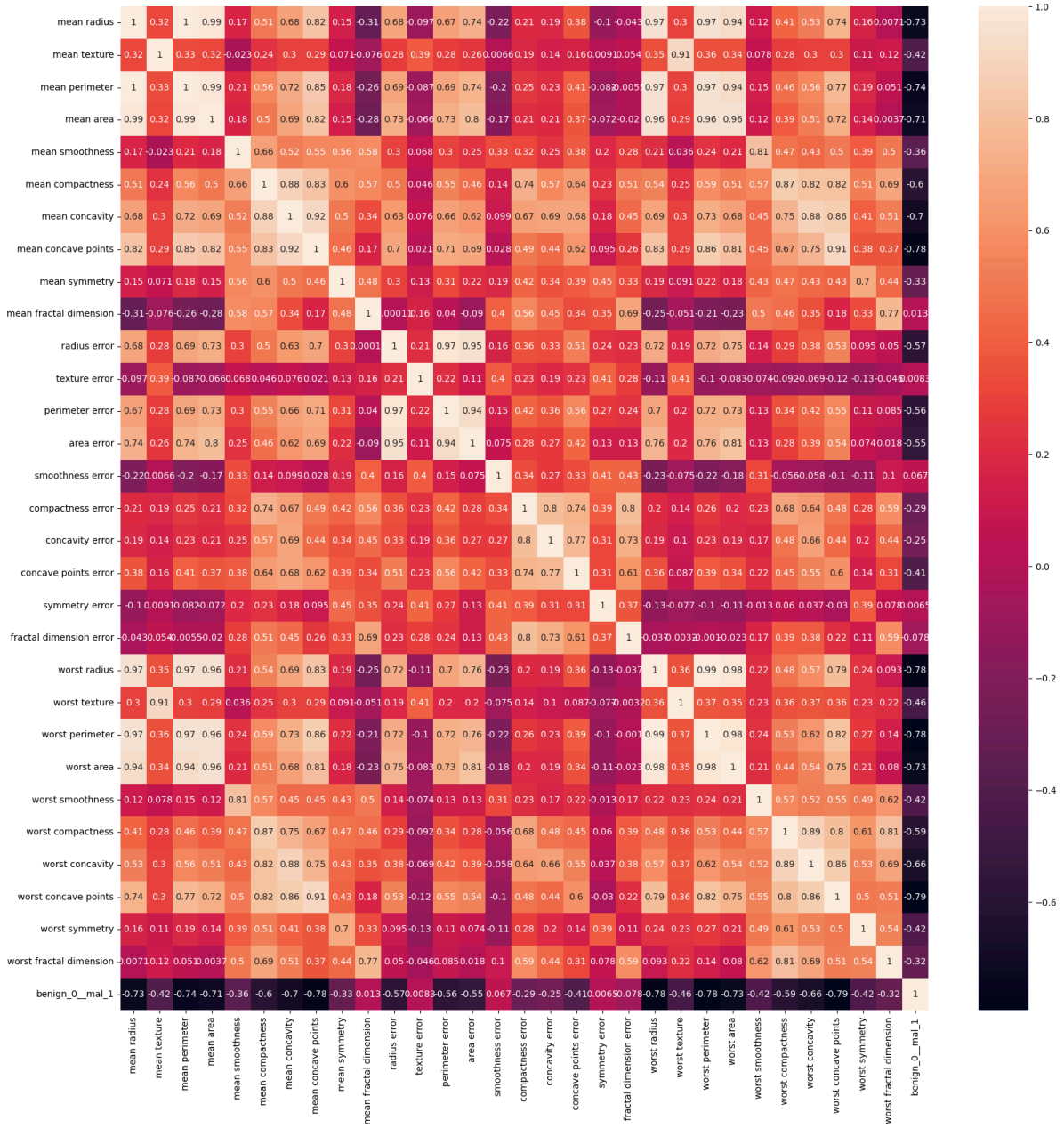
Out[5]:
```
mean radius                0
mean texture               0
mean perimeter             0
mean area                  0
mean smoothness            0
mean compactness           0
mean concavity             0
mean concave points        0
mean symmetry              0
mean fractal dimension     0
radius error               0
texture error              0
perimeter error            0
area error                 0
smoothness error           0
compactness error          0
concavity error            0
concave points error       0
symmetry error             0
fractal dimension error    0
worst radius               0
worst texture              0
worst perimeter            0
worst area                 0
worst smoothness           0
worst compactness          0
worst concavity            0
worst concave points       0
worst symmetry             0
worst fractal dimension    0
benign_0__mal_1            0
dtype: int64
```

In [6]:
```
data.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 [8]:

```
plt.figure(figsize = (20,20))
sns.heatmap(data.corr() , annot = True)
plt.show()
```



In [9]:
```
data.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 [10]:
```
x = data.drop("benign_0__mal_1" , axis = 1)
```

In [116…
```
y = data[["benign_0__mal_1"]]
```

In [117…
```
x
```

Out[117]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809
...	...	...	...	...	...	...	...	...	...
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587

569 rows × 30 columns

In [118…

```
y
```

Out[118]:

	benign_0__mal_1
0	0
1	0
2	0
3	0
4	0
...	...
564	0
565	0
566	0
567	0
568	1

569 rows × 1 columns

In [119…

```
scalar = StandardScaler()
```

In [120…

```
scalar
```

Out[120]:

```
StandardScaler
StandardScaler()
```

In [121…
```
x_train , x_test , y_train , y_test = train_test_split(x,y ,test_size = 0.2 , randc
```

In [122…
```
scalar_x_train = scalar.fit_transform(x_train)
```

In [123…
```
scalar_x_test = scalar.transform(x_test)
```

In [124…
```
scalar_x_train
```

Out[124]:
```
array([[-0.32314973, -0.00339413, -0.3025939 , ..., -0.23263754,
         1.25588559,  0.45401723],
       [ 0.1465522 , -0.82494149,  0.03913478, ..., -0.85877786,
        -0.61512811, -1.53579736],
       [-1.58615805, -0.171744  , -1.5769571 , ..., -1.75859336,
         0.45903735, -0.14791458],
       ...,
       [-0.48451972, -0.97084471, -0.55009592, ..., -1.13136949,
         0.03604082, -0.91208135],
       [ 0.92170446, -0.51293307,  0.8901062 , ...,  1.11592475,
         0.45201667, -0.14963439],
       [-1.10694682, -1.60384021, -1.08865362, ..., -0.72209507,
         0.09922703, -0.37378233]])
```

In [125…
```
scalar_x_test
```

Out[125]:
```
array([[-0.10702921,  0.72836663,  0.07766302, ...,  1.65925047,
         1.25588559,  3.40290983],
       [-0.51909901, -0.07746807, -0.58066724, ..., -1.14205025,
        -0.57300397, -0.66901555],
       [-0.25687277,  1.32095817, -0.32018288, ..., -0.82905748,
        -0.76607293, -0.90921501],
       ...,
       [-1.10694682, -0.62516631, -1.08446577, ..., -0.46869786,
        -0.33254536,  0.63345023],
       [-0.23093831,  0.49492148, -0.27244137, ..., -0.8895818 ,
        -0.42732467, -0.9837399 ],
       [-0.76115399, -0.27050925, -0.76032607, ..., -0.90583513,
        -1.1802936 , -0.42193688]])
```

In [126…
```
logistic = LogisticRegression()
```

In [127…
```
logistic.fit(scalar_x_train , y_train)
```

```
C:\Users\godde\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1184: DataC
onversionWarning: A column-vector y was passed when a 1d array was expected. Pleas
e change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[127]:
▾ LogisticRegression

LogisticRegression()

In [128…
```
prediction = logistic.predict(scalar_x_test)
```

In [129…
```
prediction[10:20]
```

Out[129]:
```
array([1, 0, 0, 0, 0, 1, 1, 0, 1, 1], dtype=int64)
```

In [130…
```
y_test[10:20]
```

Out[130]:

	benign_0__mal_1
307	1
168	0
27	0
250	0
3	0
169	1
381	1
56	0
103	1
550	1

In [131…
```
error = mean_squared_error(prediction , y_test)
```

In [132…
```
error
```

Out[132]:
```
0.02631578947368421
```

In [133…
```
knn = KNeighborsClassifier(n_neighbors=400)
```

In [134…
```
knn
```

Out[134]:
```
KNeighborsClassifier(n_neighbors=400)
```

In [135…
```
knn.fit(scalar_x_train , y_train)
```

```
C:\Users\godde\anaconda3\Lib\site-packages\sklearn\neighbors\_classification.py:228: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  return self._fit(X, y)
```

Out[135]:
```
KNeighborsClassifier(n_neighbors=400)
```

In [136…
```
prediction1 = knn.predict(scalar_x_test)
```

In [137…
```
prediction[:10]
```

Out[137]:
```
array([0, 1, 1, 1, 1, 1, 0, 1, 1, 0], dtype=int64)
```

In [138…
```
y_test[:10]
```

Out[138]:

	benign_0__mal_1
14	0
334	1
457	1
101	1
346	1
500	1
385	0
275	1
496	1
492	0

In [139…
```
error2 = mean_squared_error(prediction , y_test)
```

In [140…
```
error
```

Out[140]:
```
0.02631578947368421
```

In [ ]:
```

```

In [141…
```
navi = GaussianNB()
```

In [142…
```
navi
```

Out[142]:
```
GaussianNB()
```

In [143…
```
navi.fit(scalar_x_train , y_train)
```

```
C:\Users\godde\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1184: DataC
onversionWarning: A column-vector y was passed when a 1d array was expected. Pleas
e change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[143]:
```
GaussianNB()
```

In [144…
```
prediction3 = navi.predict(scalar_x_test)
```

In [145…
```
prediction3[:10]
```

Out[145]:
```
array([0, 1, 1, 1, 1, 1, 1, 1, 1, 0], dtype=int64)
```

In [146…
```
y_test[:10]
```

Out[146]:

	benign_0__mal_1
14	0
334	1
457	1
101	1
346	1
500	1
385	0
275	1
496	1
492	0

In [147…
```
error3 = mean_squared_error(prediction3 , y_test)
```

In [148…
```
error3
```

Out[148]:
```
0.07894736842105263
```

In [149…
```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
```

In [150…
```
future = SelectKBest(score_func = chi2 , k = 0).fit(x_train ,y_train)
```

In [ ]:
```

```

In [ ]:
```

```

In [151…
```
k = future.scores_
```

In [152…
```
k
```

Out[152]:
```
array([2.09953151e+02, 7.78418265e+01, 1.57936947e+03, 4.28794942e+04,
       1.25935885e-01, 4.00893635e+00, 1.51349324e+01, 8.40465411e+00,
       1.87157595e-01, 3.76906121e-04, 2.61333324e+01, 2.20809221e-01,
       1.82486837e+02, 6.72694156e+03, 3.20561697e-03, 3.73586448e-01,
       6.39992841e-01, 2.09482110e-01, 9.58744011e-03, 1.73408512e-03,
       3.93597328e+02, 1.40169930e+02, 2.91101735e+03, 9.14774476e+04,
       3.33072778e-01, 1.47434088e+01, 3.04159053e+01, 1.03565620e+01,
       8.62573832e-01, 1.70316089e-01])
```

In [153…
```
from sklearn.manifold import TSNE
```

In [159…
```
s = TSNE(n_components=2 , random_state = 0)
```

In [155…
```
k = s.fit(scalar_x_train , y_train)
```

In [156…
```
plt.scatter(x = k[:,:0] , y = k[:,:1] , c=y, cmap='viridis', s=100)
plt.show()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[156], line 1
----> 1 plt.scatter(x = k[:,:0] , y = k[:,:1] , c=y, cmap='viridis', s=100)
      2 plt.show()

TypeError: 'TSNE' object is not subscriptable
```

In [157…
```
print(type(k))
```

```
<class 'sklearn.manifold._t_sne.TSNE'>
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: