ASSIGNMENT

- Name : Achal Rajesh Mate
- Roll No : 2203541
- Enroll No : MITU20BTCSD001
- Branch : CSE
- Class : TY CSE Is - 3
- Guided By : Prof Nagesh Jadhav Sir

Title:- Using inbuilt dataset of Breast cancer from scikit learn Implement PCA algorithm

Objectives:-
1.To learn about dimensionality reduction techniques
2.To implement principle component analysis

Theory:

Dimensionality reduction :

Its is reducing the dimensionality of a dataset.dimensionality is the number of dimensions, features or input variables associated in a dataset

Main approaches to dimensionality reduction

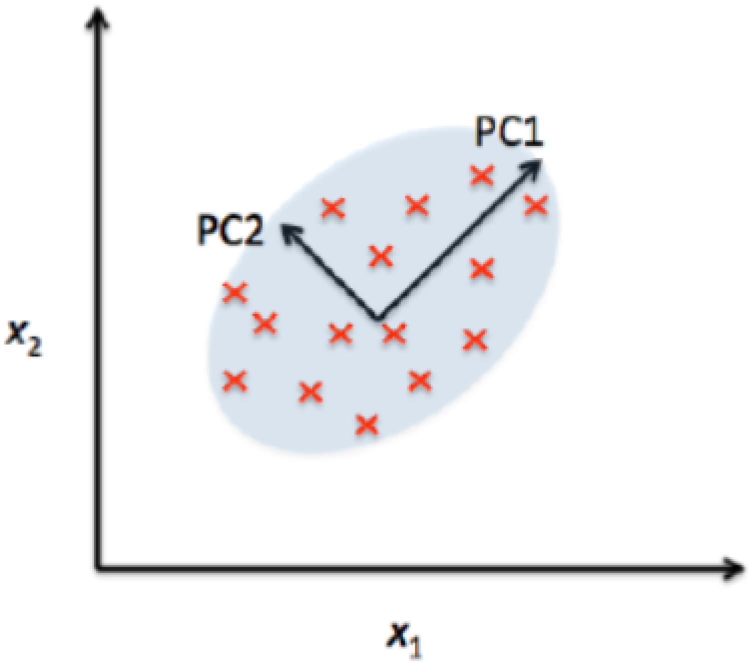
There are two main approaches to dimensionality reduction:

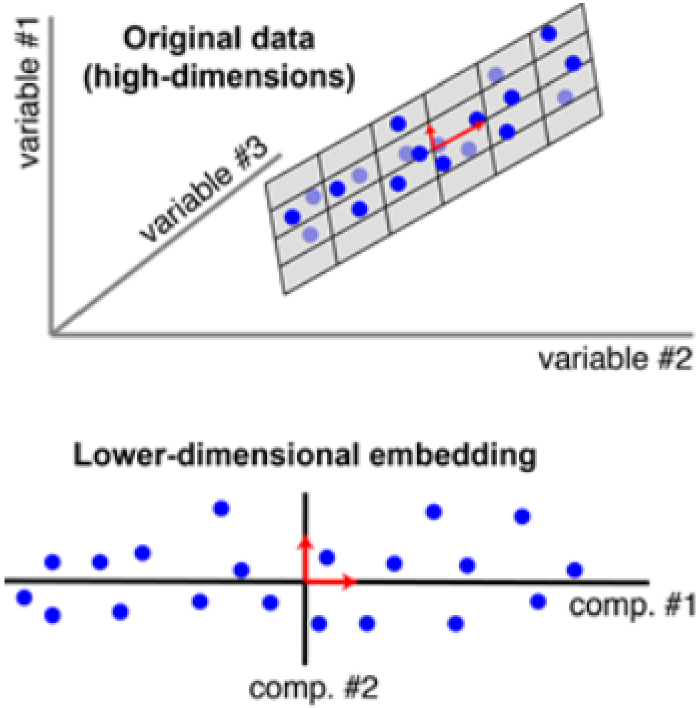
- Linear methods
- Non-linear methods (Manifold learning)

PCA

- Principal Component Analysis (PCA) is an unsupervised, non-parametric statistical technique primarily used for dimensionality reduction in machine learning.
- High dimensionality means that the dataset has a large number of features. The primary problem associated with high-dimensionality in the machine learning field is model overfitting, which reduces the ability to generalize beyond the examples in the training set.
- The ability to generalize correctly becomes exponentially harder as the dimensionality of the training dataset grows, as the training set covers a dwindling fraction of the input space.
- Models also become more efficient as the reduced feature set boosts learning rates and diminishes computation costs by removing redundant features.
- PCA can also be used to filter noisy datasets, such as image compression. The first principal component expresses the most amount of variance. Each additional component expresses less variance and more noise, so representing the data with a smaller subset of principal components preserves the signal and discards the noise. PCA is an unsupervised learning algorithm as the directions of these components is calculated purely from the explanatory feature set without any reference to response variables.

The number of feature combinations is equal to the number of dimensions of the dataset and in general set the maximum number of PCAs which can be constructed.

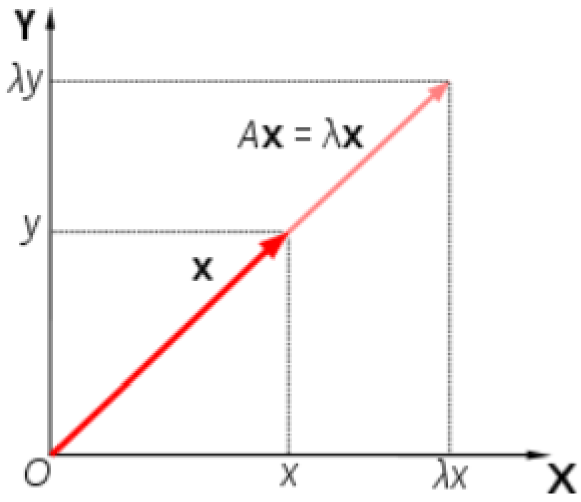




Each blue point corresponds to an observation, and each principal component reduces the three dimensions to two. The algorithm finds a pair of orthogonal vectors (red arrows) that define a lower-dimensional space (grey plane) to capture as much variance as possible from the original dataset.

Measurement

Eigenvectors and eigenvalues are measures used to quantify the direction and the magnitude of the variation captured by each axis. Eigenvector describes the angle or direction of the axis through the data space, and the eigenvalue quantifies the magnitude of the variance of the data on the axis.

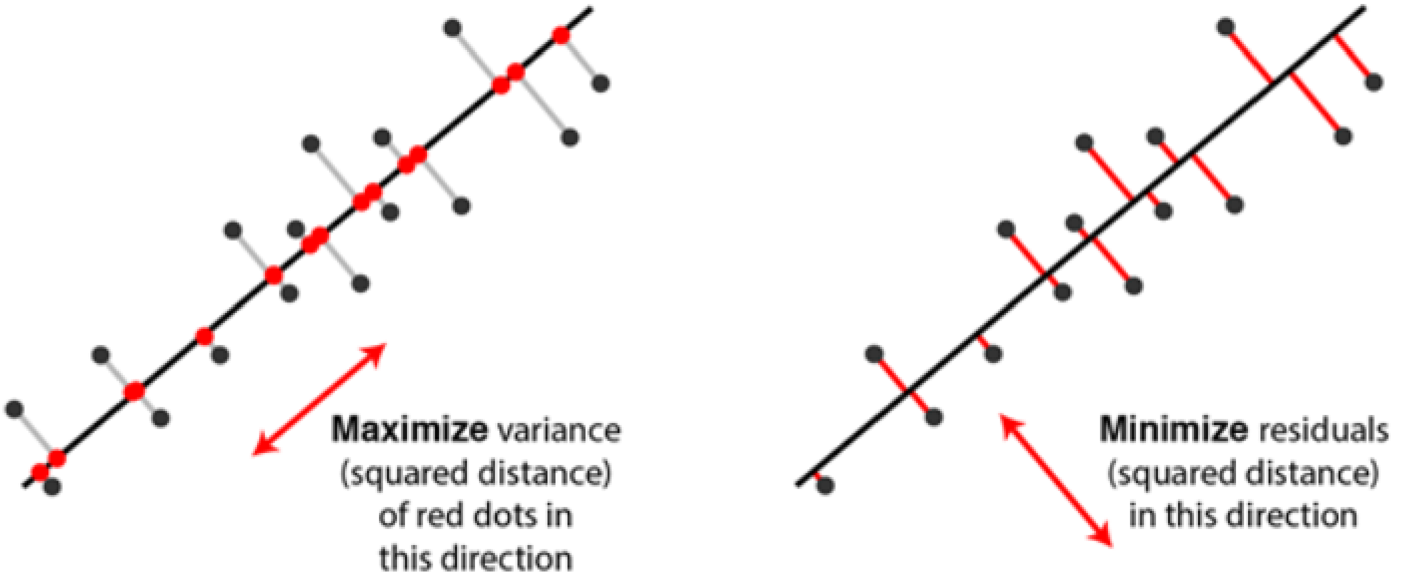


A is an x n matrix, ƛ is the eigenvalue, and the X is the eigenvector.

The number of feature combinations is equal to the number of dimensions of the dataset. For example, a dataset with ten features will have ten eigenvalues/eigenvector combinations.

The correlation between each principal component should be zero as subsequent components capture the remaining variance. Correlation between any pair of eigenvalue/eigenvector is zero so that the axes are orthogonal, i.e., perpendicular to each other in the data space.

The line which maximizes the variance of the data once it is projected into the data space is equivalent to finding the path which minimizes the least-squares distance of the projection.



Dataset used and its attributes

DataSet Link : https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(diagnostic)
Dataset Information : Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.

Features Explanations:

- Number of Instances: 569
- Number of Attributes: 30 numeric, predictive attributes and the class #### Attribute Information:
- radius (mean of distances from center to points on the perimeter)
- texture (standard deviation of gray-scale values)
- perimeter
- area
- smoothness (local variation in radius lengths)

- compactness (perimeter^2 / area - 1.0)
- concavity (severity of concave portions of the contour)
- concave points (number of concave portions of the contour)
- symmetry
- fractal dimension ("coastline approximation" - 1)

- The mean, standard error, and "worst" or largest (mean of the three largest values) of these features were computed for each data, resulting in 30 features. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.

- Missing Values: 569
- Class Distribution:
  - 212 - Malignant,
  - 357 - Benign Depending on the types of cells in a tumor, it can be:
- Benign - The tumor doesn’t contain cancerous cells.
- Malignant - The tumor contains cancerous cells.

Import All Necessary Library

In [1]:

```
import pandas as pd
import numpy as np

import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import warnings
warnings.filterwarnings("ignore")
```

In [47]:

```
from sklearn import preprocessing
from scipy.stats import norm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.decomposition import PCA
```

Load the DataSet

In [3]:
```
df = pd.read_csv("Breast Cancer Data.csv")
```

In [4]:
```
df.head()
```

Out[4]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	...
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	...
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	...
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	...
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	...

5 rows × 33 columns

View First 5 Rows

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

Out[5]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	...
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	...
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	...
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	...
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	...

5 rows × 33 columns

Set Option to View all Rows and Columns

In [6]:

```
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
```

Dimensions of the Dataset

In [7]:

```
df.shape
```

Out[7]: (569, 33)

Dataset contains 569 instances with 33 rows

Concise Summary

In [8]:

```
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
```

From above result diagnosis is the feature in object datatype and Unnamed: 32 feature contains all now value lets check by using pandas function

Check the Missing Data

In [9]:

```
df.isnull().sum()
```

Out[9]:

```
id                         0
diagnosis                  0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              0
compactness_se             0
concavity_se               0
```

```
concave points_se          0
symmetry_se                0
fractal_dimension_se       0
radius_worst               0
texture_worst              0
perimeter_worst            0
area_worst                 0
smoothness_worst           0
compactness_worst          0
concavity_worst            0
concave points_worst       0
symmetry_worst             0
fractal_dimension_worst    0
Unnamed: 32              569
dtype: int64
```

From the above observation Unnamed: 32 column contains all the Null values

So it would be better to drop the column

Drop the Column Containing Missing Value

In [10]:

```
df.drop('Unnamed: 32',axis = 1,inplace = True)
```

Recheck the Missing Value is present or not

In [11]:

```
df.isnull().sum().sum()
```

Out[11]:

```
0
```

Statistical Summary of data

In [12]:

```
df.describe()
```

Out[12]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	sym
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	sym
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	

Columns of the dataset

In [13]:

```
df.columns
```

Out[13]:

```
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')
```

Check The types of values of Diagnosis Present In dataset

In [14]:

```
df.diagnosis.unique()
```

Out[14]:

```
array(['M', 'B'], dtype=object)
```

Count number of Malignant (M) or Benign (B) Cells

In [15]:

```
df['diagnosis'].value_counts()
```

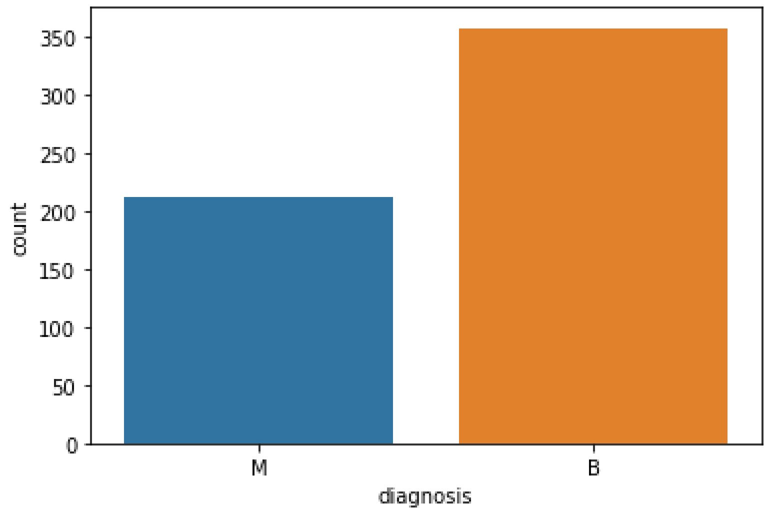
Out[15]:
```
B    357
M    212
Name: diagnosis, dtype: int64
```

From above result we can say that M=Maligant is of 212 instance which is postive prediction

B= Benign is of 357 instance which is negative prediction

In [16]:
```
sns.countplot('diagnosis',data=df,label ="Diagnosis")
```

Out[16]:
```
<AxesSubplot:xlabel='diagnosis', ylabel='count'>
```

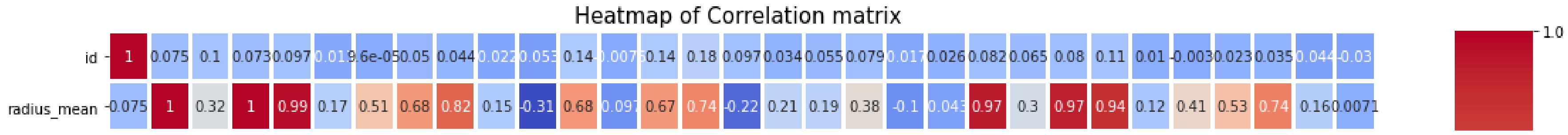


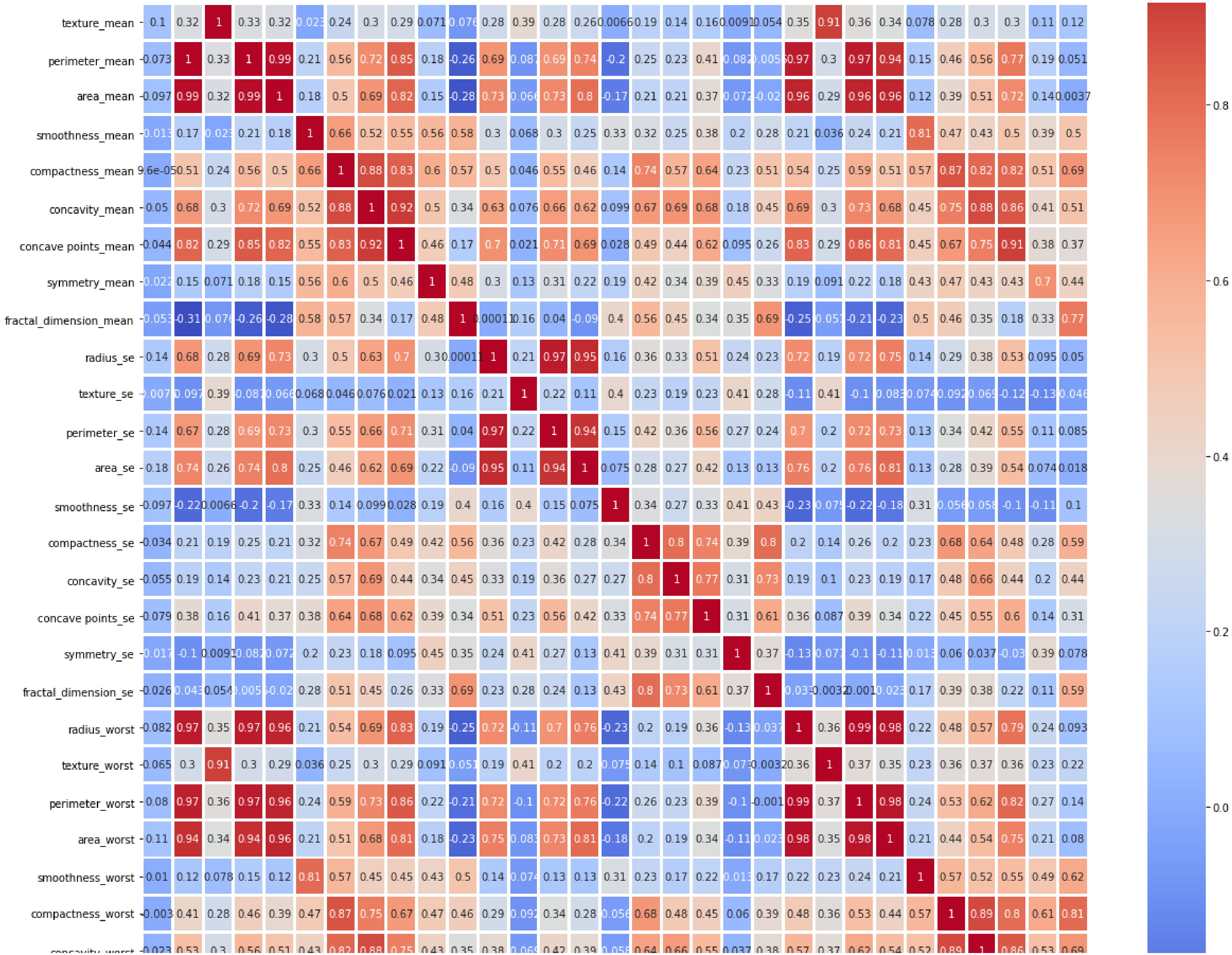
Dataset Contain Maximum Number of Negative Predictions

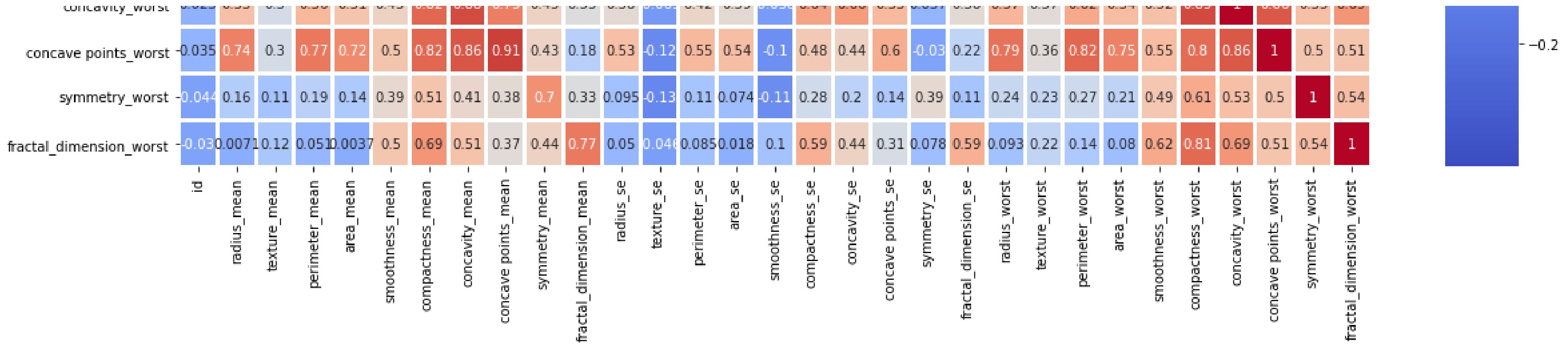
Correlation Between Data

In [18]:
```
plt.figure(figsize=(20,20))
plt.title("Heatmap of Correlation matrix ",fontsize=15)
sns.heatmap(df.corr(), annot = True, cmap ='coolwarm', linewidths=2)
```

Out[18]:
```
<AxesSubplot:title={'center':'Heatmap of Correlation matrix '}>
```




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
0.8
0.6
0.4
0.2
0.0



Data Preprocessing -

Feature elemination

In [22]:

```
# feature is not for our use as it consist of id of the patient
df.drop('id',axis=1,inplace= True)
```

In [23]:

```
df.isnull().sum().sum()
```

Out[23]:

```
0
```

Feature encoding

Converting Categorical features into numberic features

In [19]:

```
le = preprocessing.LabelEncoder()
df['diagnosis']=le.fit_transform(df['diagnosis'])
```

In [20]:

```
df['diagnosis'].unique()
```

Out[20]:

```
array([1, 0])
```

In [24]:
```
df.head()
```

Out[24]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_me
0	1	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2
1	1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1
2	1	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2
3	1	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2
4	1	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1

Now our data is ready to pass the model and perform ML building.

In [25]:
```
cancer_pos_rate = np.sum(df.diagnosis) / len(df.diagnosis) *100
print('Breast Cancer +ve rate - ',cancer_pos_rate)
```

```
Breast Cancer +ve rate -  37.258347978910365
```

The given dataset contains only 3.7 % of cancer +ve data which is balance data for building accurate model.

Spliting Dataset for Feature Selection of Model Building

The process dataset contain 569 instance ,31 columns including target variable and zero null values

In [39]:
```
X =df.drop(['diagnosis'],axis = 1)
y = df['diagnosis']
```

In [40]:
```
scalar = preprocessing.StandardScaler()
```

In [41]:
```
col = X.columns
```

In [42]:
```
X =  scalar.fit_transform(X)
```

In [43]:

```
X = pd.DataFrame(X,columns=col)
X.head()
```

Out[43]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	fractal_
0	1.097064	-2.073335	1.269934	0.984375	1.568466	3.283515	2.652874	2.532475	2.217515	
1	1.829821	-0.353632	1.685955	1.908708	-0.826962	-0.487072	-0.023846	0.548144	0.001392	
2	1.579888	0.456187	1.566503	1.558884	0.942210	1.052926	1.363478	2.037231	0.939685	
3	-0.768909	0.253732	-0.592687	-0.764464	3.283553	3.402909	1.915897	1.451707	2.867383	
4	1.750297	-1.151816	1.776573	1.826229	0.280372	0.539340	1.371011	1.428493	-0.009560	

In [45]:

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2)
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

Out[45]:

```
((455, 30), (114, 30), (455,), (114,))
```

In [50]:

```
lr = LogisticRegression().fit(X_train,y_train)
print(lr.score(X_train,y_train))
print(lr.score(X_test,y_test))
```

```
0.9956043956043956
0.956140350877193
```

In [57]:

```
components = None
pca = PCA(n_components = 0.85)
# perform PCA on the scaled data
pca.fit(X)
```

Out[57]:

```
PCA(n_components=0.85)
```

In [58]:

```
print("Eigenvalues:")
```

```python
print(pca.explained_variance_)
print("Variances (Percentage):")
print(pca.explained_variance_ratio_ * 100)
print()
print("EigenVectors")
print(pca.components_)
print("Cumulative Variances (Percentage):")
print(np.cumsum(pca.explained_variance_ratio_ * 100))
components = len(pca.explained_variance_ratio_)
print(f'Number of components: {components}')
# Make the scree plot
plt.plot(range(1, components + 1), np.cumsum(pca.explained_variance_ratio_ * 100))
plt.xlabel("Number of components")
plt.ylabel("Explained variance (%)")
```

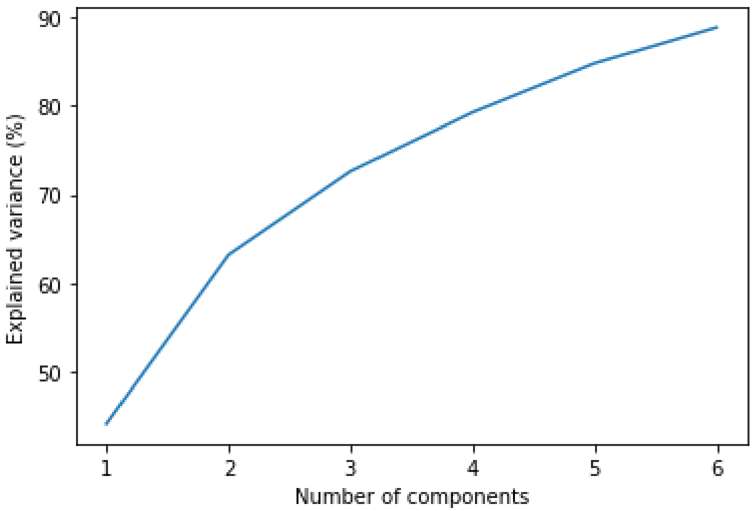
```
Eigenvalues:
[13.30499079  5.7013746   2.82291016  1.98412752  1.65163324  1.20948224]
Variances (Percentage):
[44.27202561 18.97118204  9.39316326  6.60213492  5.49576849  4.02452204]

EigenVectors
[[ 2.18902444e-01  1.03724578e-01  2.27537293e-01  2.20994985e-01
   1.42589694e-01  2.39285354e-01  2.58400481e-01  2.60853758e-01
   1.38166959e-01  6.43633464e-02  2.05978776e-01  1.74280281e-02
   2.11325916e-01  2.02869635e-01  1.45314521e-02  1.70393451e-01
   1.53589790e-01  1.83417397e-01  4.24984216e-02  1.02568322e-01
   2.27996634e-01  1.04469325e-01  2.36639681e-01  2.24870533e-01
   1.27952561e-01  2.10095880e-01  2.28767533e-01  2.50885971e-01
   1.22904556e-01  1.31783943e-01]
 [-2.33857132e-01 -5.97060883e-02 -2.15181361e-01 -2.31076711e-01
   1.86113023e-01  1.51891610e-01  6.01653628e-02 -3.47675005e-02
   1.90348770e-01  3.66575471e-01 -1.05552152e-01  8.99796818e-02
  -8.94572342e-02 -1.52292628e-01  2.04430453e-01  2.32715896e-01
   1.97207283e-01  1.30321560e-01  1.83848000e-01  2.80092027e-01
  -2.19866379e-01 -4.54672983e-02 -1.99878428e-01 -2.19351858e-01
   1.72304352e-01  1.43593173e-01  9.79641143e-02 -8.25723507e-03
   1.41883349e-01  2.75339469e-01]
 [-8.53124284e-03  6.45499033e-02 -9.31421972e-03  2.86995259e-02
  -1.04291904e-01 -7.40915709e-02  2.73383798e-03 -2.55635406e-02
  -4.02399363e-02 -2.25740897e-02  2.68481387e-01  3.74633665e-01
   2.66645367e-01  2.16006528e-01  3.08838979e-01  1.54779718e-01
   1.76463743e-01  2.24657567e-01  2.88584292e-01  2.11503764e-01
  -4.75069900e-02 -4.22978228e-02 -4.85465083e-02 -1.19023182e-02
  -2.59797613e-01 -2.36075625e-01 -1.73057335e-01 -1.70344076e-01
  -2.71312642e-01 -2.32791313e-01]
 [ 4.14089623e-02 -6.03050001e-01  4.19830991e-02  5.34337955e-02
```

```
  1.59382765e-01  3.17945811e-02  1.91227535e-02  6.53359443e-02
  6.71249840e-02  4.85867649e-02  9.79412418e-02 -3.59855528e-01
  8.89924146e-02  1.08205039e-01  4.46641797e-02 -2.74693632e-02
  1.31687997e-03  7.40673350e-02  4.40733510e-02  1.53047496e-02
  1.54172396e-02 -6.32807885e-01  1.38027944e-02  2.58947492e-02
  1.76522161e-02 -9.13284153e-02 -7.39511797e-02  6.00699571e-03
 -3.62506947e-02 -7.70534703e-02]
[ 3.77863538e-02 -4.94688505e-02  3.73746632e-02  1.03312514e-02
 -3.65088528e-01  1.17039713e-02  8.63754118e-02 -4.38610252e-02
 -3.05941428e-01 -4.44243602e-02 -1.54456496e-01 -1.91650506e-01
 -1.20990220e-01 -1.27574432e-01 -2.32065676e-01  2.79968156e-01
  3.53982091e-01  1.95548089e-01 -2.52868765e-01  2.63297438e-01
 -4.40659209e-03 -9.28834001e-02  7.45415100e-03 -2.73909030e-02
 -3.24435445e-01  1.21804107e-01  1.88518727e-01  4.33320687e-02
 -2.44558663e-01  9.44233510e-02]
[ 1.87407904e-02 -3.21788366e-02  1.73084449e-02 -1.88774796e-03
 -2.86374497e-01 -1.41309489e-02 -9.34418089e-03 -5.20499505e-02
  3.56458461e-01 -1.19430668e-01 -2.56032561e-02 -2.87473145e-02
  1.81071500e-03 -4.28639079e-02 -3.42917393e-01  6.91975186e-02
  5.63432386e-02 -3.12244482e-02  4.90245643e-01 -5.31952674e-02
 -2.90684919e-04 -5.00080613e-02  8.50098715e-03 -2.51643821e-02
 -3.69255370e-01  4.77057929e-02  2.83792555e-02 -3.08734498e-02
  4.98926784e-01 -8.02235245e-02]]
Cumulative Variances (Percentage):
[44.27202561 63.24320765 72.63637091 79.23850582 84.73427432 88.75879636]
Number of components: 6
```

Out[58]:

```
Text(0, 0.5, 'Explained variance (%)')
```



```
In [59]: pca_components = abs(pca.components_)
         print(pca_components)
```

```
[[2.18902444e-01 1.03724578e-01 2.27537293e-01 2.20994985e-01
  1.42589694e-01 2.39285354e-01 2.58400481e-01 2.60853758e-01
  1.38166959e-01 6.43633464e-02 2.05978776e-01 1.74280281e-02
  2.11325916e-01 2.02869635e-01 1.45314521e-02 1.70393451e-01
  1.53589790e-01 1.83417397e-01 4.24984216e-02 1.02568322e-01
  2.27996634e-01 1.04469325e-01 2.36639681e-01 2.24870533e-01
  1.27952561e-01 2.10095880e-01 2.28767533e-01 2.50885971e-01
  1.22904556e-01 1.31783943e-01]
 [2.33857132e-01 5.97060883e-02 2.15181361e-01 2.31076711e-01
  1.86113023e-01 1.51891610e-01 6.01653628e-02 3.47675005e-02
  1.90348770e-01 3.66575471e-01 1.05552152e-01 8.99796818e-02
  8.94572342e-02 1.52292628e-01 2.04430453e-01 2.32715896e-01
  1.97207283e-01 1.30321560e-01 1.83848000e-01 2.80092027e-01
  2.19866379e-01 4.54672983e-02 1.99878428e-01 2.19351858e-01
  1.72304352e-01 1.43593173e-01 9.79641143e-02 8.25723507e-03
  1.41883349e-01 2.75339469e-01]
 [8.53124284e-03 6.45499033e-02 9.31421972e-03 2.86995259e-02
  1.04291904e-01 7.40915709e-02 2.73383798e-03 2.55635406e-02
  4.02399363e-02 2.25740897e-02 2.68481387e-01 3.74633665e-01
  2.66645367e-01 2.16006528e-01 3.08838979e-01 1.54779718e-01
  1.76463743e-01 2.24657567e-01 2.88584292e-01 2.11503764e-01
  4.75069900e-02 4.22978228e-02 4.85465083e-02 1.19023182e-02
  2.59797613e-01 2.36075625e-01 1.73057335e-01 1.70344076e-01
  2.71312642e-01 2.32791313e-01]
 [4.14089623e-02 6.03050001e-01 4.19830991e-02 5.34337955e-02
  1.59382765e-01 3.17945811e-02 1.91227535e-02 6.53359443e-02
  6.71249840e-02 4.85867649e-02 9.79412418e-02 3.59855528e-01
  8.89924146e-02 1.08205039e-01 4.46641797e-02 2.74693632e-02
  1.31687997e-03 7.40673350e-02 4.40733510e-02 1.53047496e-02
  1.54172396e-02 6.32807885e-01 1.38027944e-02 2.58947492e-02
  1.76522161e-02 9.13284153e-02 7.39511797e-02 6.00699571e-03
  3.62506947e-02 7.70534703e-02]
 [3.77863538e-02 4.94688505e-02 3.73746632e-02 1.03312514e-02
  3.65088528e-01 1.17039713e-02 8.63754118e-02 4.38610252e-02
  3.05941428e-01 4.44243602e-02 1.54456496e-01 1.91650506e-01
  1.20990220e-01 1.27574432e-01 2.32065676e-01 2.79968156e-01
  3.53982091e-01 1.95548089e-01 2.52868765e-01 2.63297438e-01
  4.40659209e-03 9.28834001e-02 7.45415100e-03 2.73909030e-02
  3.24435445e-01 1.21804107e-01 1.88518727e-01 4.33320687e-02
  2.44558663e-01 9.44233510e-02]
 [1.87407904e-02 3.21788366e-02 1.73084449e-02 1.88774796e-03
  2.86374497e-01 1.41309489e-02 9.34418089e-03 5.20499505e-02
  3.56458461e-01 1.19430668e-01 2.56032561e-02 2.87473145e-02
  1.81071500e-03 4.28639079e-02 3.42917393e-01 6.91975186e-02
```

```
 5.63432386e-02 3.12244482e-02 4.90245643e-01 5.31952674e-02
 2.90684919e-04 5.00080613e-02 8.50098715e-03 2.51643821e-02
 3.69255370e-01 4.77057929e-02 2.83792555e-02 3.08734498e-02
 4.98926784e-01 8.02235245e-02]]
```

In [60]:

```
print('Top 4 most important features in each component')
print('===============================================')
for row in range(pca_components.shape[0]):
    # get the indices of the top 4 values in each row
    temp = np.argpartition(-(pca_components[row]), 4)

    # sort the indices in descending order
    indices = temp[np.argsort((-pca_components[row])[temp])][:4]

    # print the top 4 feature names
    print(f'Component {row}: {df.columns[indices].to_list()}')
```

```
Top 4 most important features in each component
===============================================
Component 0: ['concavity_mean', 'compactness_mean', 'concavity_worst', 'smoothness_mean']
Component 1: ['symmetry_mean', 'symmetry_se', 'symmetry_worst', 'diagnosis']
Component 2: ['radius_se', 'area_se', 'concave points_se', 'concave points_worst']
Component 3: ['radius_worst', 'radius_mean', 'radius_se', 'area_mean']
Component 4: ['area_mean', 'compactness_se', 'area_worst', 'concave points_mean']
Component 5: ['concave points_worst', 'concave points_se', 'area_worst', 'concave points_mean']
```

In [62]:

```
transformed_df = pd.DataFrame(pca.transform(X),
                              columns=['PC1', 'PC2', 'PC3',
                                       'PC4', 'PC5','PC6'])
```

In [63]:

```
transformed_df.head()
```

Out[63]:

	PC1	PC2	PC3	PC4	PC5	PC6
0	9.192837	1.948583	-1.123166	3.633731	-1.195110	1.411424
1	2.387802	-3.768172	-0.529293	1.118264	0.621775	0.028656
2	5.733896	-1.075174	-0.551748	0.912083	-0.177086	0.541452
3	7.122953	10.275589	-3.232790	0.152547	-2.960878	3.053422

	PC1	PC2	PC3	PC4	PC5	PC6
4	3.935302	-1.948072	1.389767	2.940639	0.546747	-1.226495

In [64]:
```
X_train,X_test,y_train,y_test = train_test_split(transformed_df,y,test_size=0.2)
```

In [65]:
```
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

Out[65]:
```
((455, 6), (114, 6), (455,), (114,))
```

In [66]:
```
lr1 = LogisticRegression().fit(X_train,y_train)
print(lr1.score(X_train,y_train))
print(lr1.score(X_test,y_test))
```

```
0.9736263736263736
0.9736842105263158
```

Conclusion

Thus we have successfully studied and Implemented PCA on Breast Cancer Dataset

In [ ]:
```

```