CANCER PREDICTION USING PRINCIPAL COMPONENT ANALYSIS

PCA captures maximum variance by rotating on the axis and first PC has highest variance. Each and every component is created in such a way that they are orthogonal to each other. It reduces the number of numeric features only.

1. Dimension reduction (Curse of Dimesionality proposed by Richard in 1961) approach for numeric independent features only.

2. Data must be normally distributed and scaled properly.

3. No outliers should be there.

4. There is no correlation between any principal components.

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


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

In [2]:
```
df = pd.read_csv('data.csv')
```

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

Out[3]:

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

5 rows × 33 columns

Data Preprocessing

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
id                         569 non-null int64
diagnosis                  569 non-null object
radius_mean                569 non-null float64
texture_mean               569 non-null float64
perimeter_mean             569 non-null float64
area_mean                  569 non-null float64
smoothness_mean            569 non-null float64
compactness_mean           569 non-null float64
concavity_mean             569 non-null float64
concave points_mean        569 non-null float64
symmetry_mean              569 non-null float64
fractal_dimension_mean     569 non-null float64
radius_se                  569 non-null float64
texture_se                 569 non-null float64
perimeter_se               569 non-null float64
area_se                    569 non-null float64
smoothness_se              569 non-null float64
compactness_se             569 non-null float64
concavity_se               569 non-null float64
concave points_se          569 non-null float64
symmetry_se                569 non-null float64
fractal_dimension_se       569 non-null float64
radius_worst               569 non-null float64
texture_worst              569 non-null float64
perimeter_worst            569 non-null float64
area_worst                 569 non-null float64
smoothness_worst           569 non-null float64
compactness_worst          569 non-null float64
concavity_worst            569 non-null float64
concave points_worst       569 non-null float64
symmetry_worst             569 non-null float64
fractal_dimension_worst    569 non-null float64
Unnamed: 32                0 non-null float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
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
```

In [6]:
```
X = df.drop(columns=['Unnamed: 32','id','diagnosis'])
y = df['diagnosis']
```

In [7]:
```
X.head()
```

Out[7]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_
0	17.99	10.38	122.80	1001.0	0.11840	0
1	20.57	17.77	132.90	1326.0	0.08474	0
2	19.69	21.25	130.00	1203.0	0.10960	0
3	11.42	20.38	77.58	386.1	0.14250	0
4	20.29	14.34	135.10	1297.0	0.10030	0

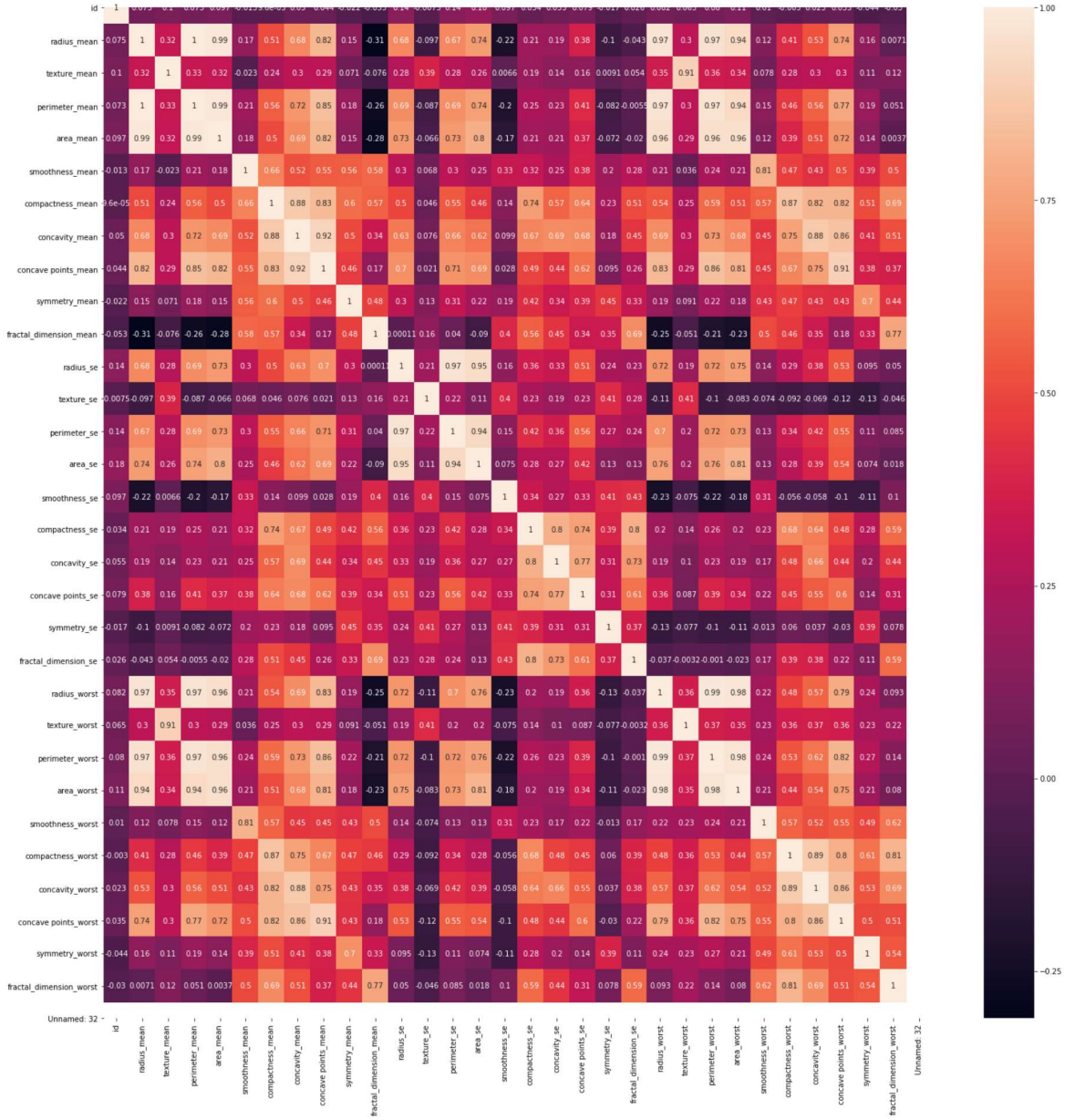
5 rows × 30 columns

In [8]:
```
y.value_counts()
```

Out[8]:
```
B    357
M    212
Name: diagnosis, dtype: int64
```

In [9]:

```
plt.figure(figsize=(25,25))
sns.heatmap(df.corr(), annot=True);
```

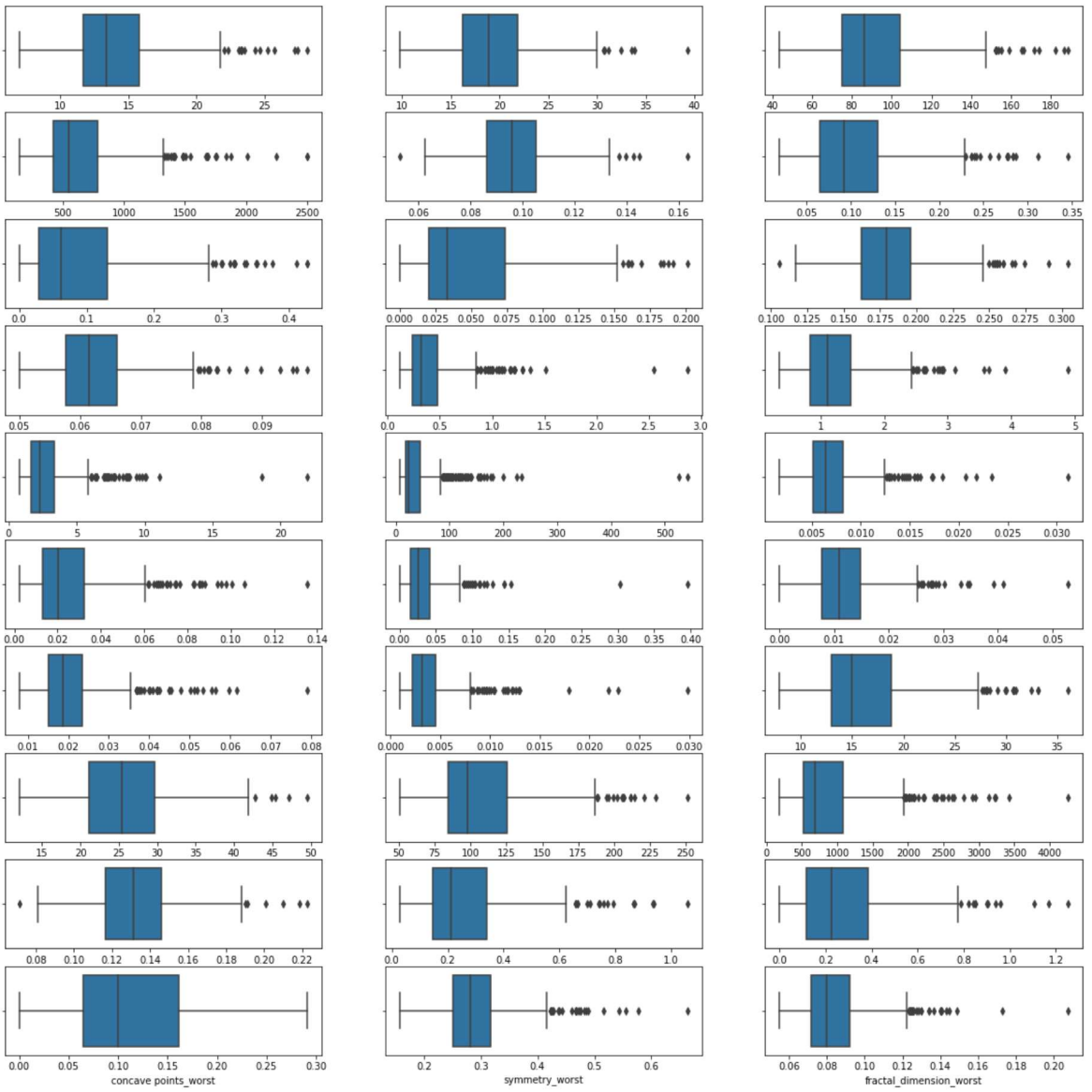


In [10]:
```
cols = X.columns
len(cols)
```

Out[10]: 30

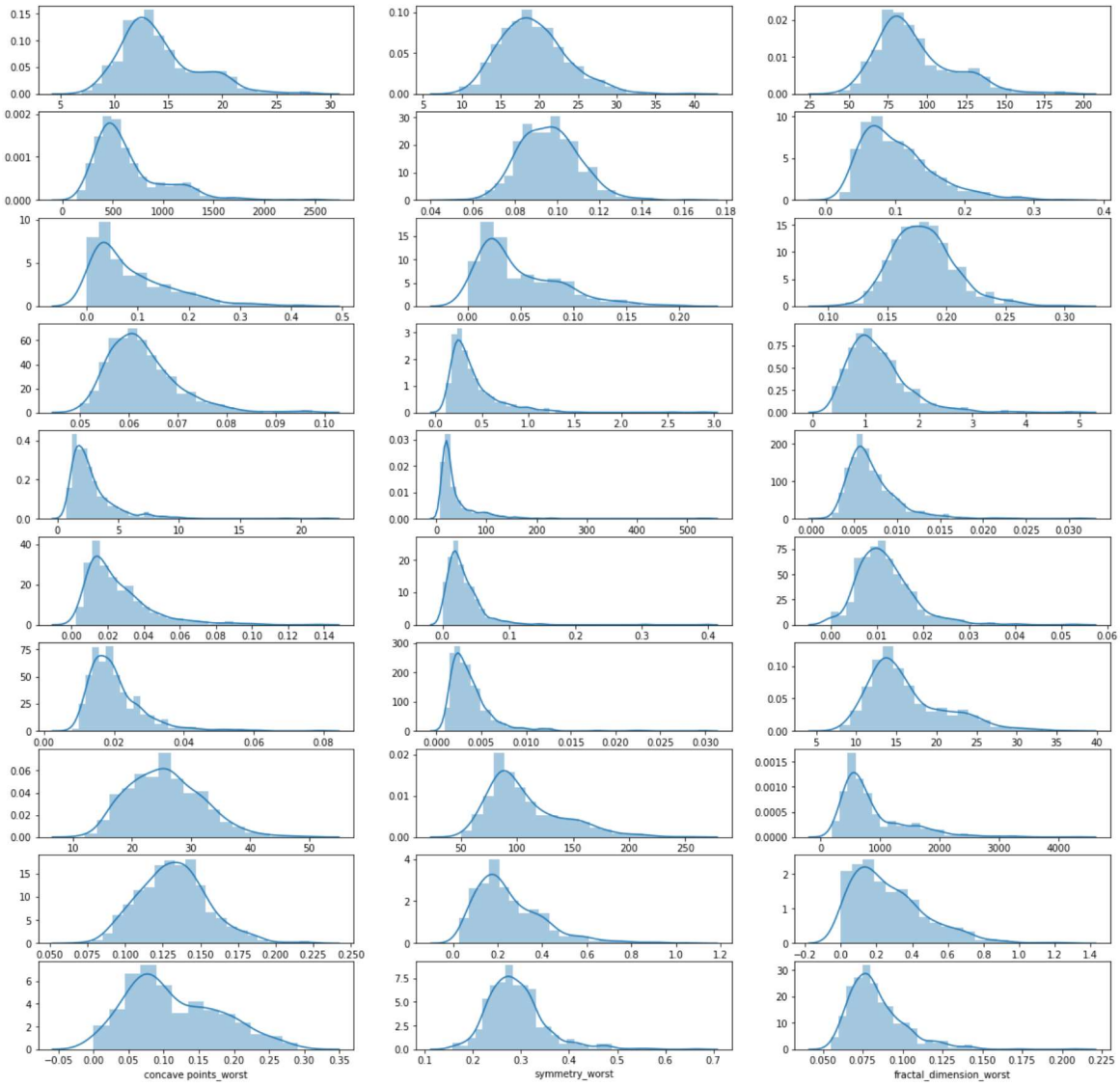
In [11]:

```
plt.figure(figsize=(20,20))

for i in range(len(cols)):
  plt.subplot(10,3,i+1)
  sns.boxplot(x = cols[i],data=X)
```



In [12]:

```
plt.figure(figsize=(20,20))

for i in range(len(cols)):
  plt.subplot(10,3,i+1)
  sns.distplot(X[cols[i]])
```



In [13]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, r
```

In [14]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

log = LogisticRegression()

log.fit(X_train,y_train)
y_pred = log.predict(X_test)

accuracy_score(y_pred, y_test)
```

Out[14]: 0.9385964912280702

In [15]:
```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [16]:
```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier()

rf.fit(X_train,y_train)
y_pred = rf.predict(X_test)
accuracy_score(y_test, y_pred)
```

Out[16]: 0.9473684210526315

In [17]:
```
y_train_pred = rf.predict(X_train)
accuracy_score(y_train_pred, y_train)
```

Out[17]: 0.9978021978021978

In [18]:
```
from sklearn.linear_model import LogisticRegression

log = LogisticRegression()

log.fit(X_train,y_train)
y_pred = log.predict(X_test)

accuracy_score(y_pred, y_test)
```

Out[18]: 0.9736842105263158

In [19]:
```
from sklearn.decomposition import PCA

pca = PCA(n_components = 10, random_state=42)
```

In [20]:
```
X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.transform(X_test)
```

In [21]:
```
print(X_train_pca.shape)
print(X_train.shape)
```

```
(455, 10)
(455, 30)
```

In [22]:
```
log_pca = LogisticRegression()

log_pca.fit(X_train_pca,y_train)
y_pred = log_pca.predict(X_test_pca)

accuracy_score(y_pred, y_test)
```

Out[22]: 0.9736842105263158

In [23]:
```
rf_pca = RandomForestClassifier()

rf_pca.fit(X_train_pca,y_train)
y_pred = rf_pca.predict(X_test_pca)
accuracy_score(y_test, y_pred)
```

Out[23]: 0.9736842105263158

In [24]:
```
X_train = pd.DataFrame(X_train, columns=X.columns)
X_test = pd.DataFrame(X_test, columns=X.columns)

X_train_pca = pd.DataFrame(X_train_pca)
X_test_pca = pd.DataFrame(X_test_pca)
```

In [25]:
```
X_train.head()
```

Out[25]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_
0	0.518559	0.891826	0.424632	0.383925	-0.974744	-0.6
1	-0.516364	-1.639710	-0.541349	-0.542961	0.476219	-0.6
2	-0.368118	0.455515	-0.388250	-0.402970	-1.432979	-0.3
3	0.205285	0.726168	0.400330	0.070612	0.243253	2.2
4	1.243005	0.194195	1.210377	1.206652	-0.111442	0.0

5 rows × 30 columns

In [26]:
```
X_train_pca.head()
```

Out[26]:

	0	1	2	3	4	5	6	7	
0	-0.875937	-2.571309	-0.548511	-1.635793	0.262836	0.109819	-0.506414	-0.214460	-0.1
1	-2.505128	0.192453	-0.404965	2.447875	-0.600618	0.007704	1.094288	0.073162	0.5
2	-1.354117	0.417851	2.545009	-1.218872	1.766641	0.708774	-0.766716	-0.053032	-0.4
3	4.854091	3.017576	-1.626587	-0.702197	2.814424	-0.375296	0.473987	0.060816	0.1
4	2.926225	-1.866546	2.486468	0.104045	-0.181489	-0.974601	0.773459	-0.177466	-0.2

In [ ]:
```

```