SVM PROJECT

LOADING LIBRARIES

In [2]:

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

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_

from sklearn.svm import SVC
```

In [3]:

```
!pip show matplotlib
```

```
Name: matplotlib
Version: 3.7.3
Summary: Python plotting package
Home-page: https://matplotlib.org (https://matplotlib.org)
Author: John D. Hunter, Michael Droettboom
Author-email: matplotlib-users@python.org
License: PSF
Location: C:\Users\Shagun\AppData\Roaming\Python\Python311\site-packages
Requires: contourpy, cycler, fonttools, kiwisolver, numpy, packaging, pillow,
pyparsing, python-dateutil
Required-by: seaborn, wordcloud
```

CALLING THE DATASET

In [4]:

```
df=pd.read_csv("Breast_Cancer.csv")
```

In [5]:

```
df
```

Out[5]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_
0	842302	M	17.99	10.38	122.80	1001.0	0
1	842517	M	20.57	17.77	132.90	1326.0	0
2	84300903	M	19.69	21.25	130.00	1203.0	0
3	84348301	M	11.42	20.38	77.58	386.1	0
4	84358402	M	20.29	14.34	135.10	1297.0	0
...	...	...	...	...	...	...	
564	926424	M	21.56	22.39	142.00	1479.0	0
565	926682	M	20.13	28.25	131.20	1261.0	0
566	926954	M	16.60	28.08	108.30	858.1	0
567	927241	M	20.60	29.33	140.10	1265.0	0
568	92751	B	7.76	24.54	47.92	181.0	0

569 rows × 33 columns

PREPROCESSING OF DATA

In [6]:

```

df.isnull().sum() #We can see that there are no null values in the data
```

Out[6]:

```
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
```

MAPPING OF DIAGNOSIS

In [7]:

```
df["diagnosis"]=df["diagnosis"].map({"M":1,"B":0})
```

In [8]:

```
len(df)
```

Out[8]:

```
569
```

UNIQUESNESS OF DIAGNOSIS COLUMN

In [9]:
```
df.diagnosis.unique()
```

Out[9]:
```
array([1, 0], dtype=int64)
```

BASIN INFO OF THE DATA

In [10]:
```
df.describe()
```

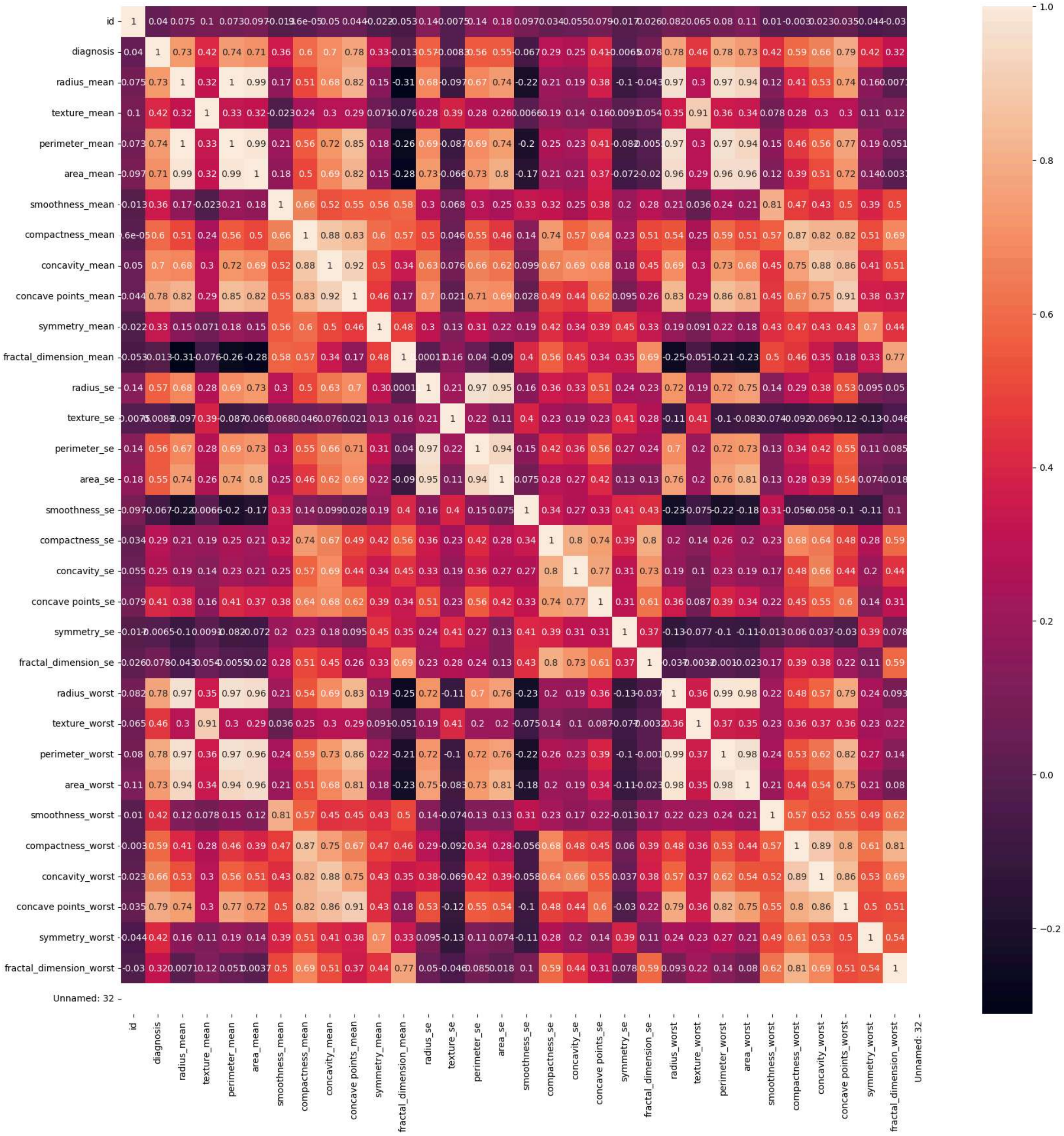
Out[10]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smc
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	
mean	3.037183e+07	0.372583	14.127292	19.289649	91.969033	654.889104	
std	1.250206e+08	0.483918	3.524049	4.301036	24.298981	351.914129	
min	8.670000e+03	0.000000	6.981000	9.710000	43.790000	143.500000	
25%	8.692180e+05	0.000000	11.700000	16.170000	75.170000	420.300000	
50%	9.060240e+05	0.000000	13.370000	18.840000	86.240000	551.100000	
75%	8.813129e+06	1.000000	15.780000	21.800000	104.100000	782.700000	
max	9.113205e+08	1.000000	28.110000	39.280000	188.500000	2501.000000	

8 rows × 33 columns

CORRELATION

In [11]:
```
correlation=df.corr()
plt.figure(figsize=(20, 20))
sns.heatmap(correlation,annot=True)
plt.show()
```



using the above heatmap, we find that the columns of the mean and worst of [radius,perimeter, area,compactness,concavity,concave points]

have high correlation with the diagnosis

CREATING OUR TRAINING DATASET USING THE HIGHLY CORRELATED COLUMNS

In [12]:
```
X = df.loc[:,["radius_mean","perimeter_mean","area_mean","compactness_mean","c
Y = df["diagnosis"]
```

In [13]:
```
X
```

Out[13]:

	radius_mean	perimeter_mean	area_mean	compactness_mean	concave points_mean	radius_worst
0	17.99	122.80	1001.0	0.27760	0.14710	25.380
1	20.57	132.90	1326.0	0.07864	0.07017	24.990
2	19.69	130.00	1203.0	0.15990	0.12790	23.570
3	11.42	77.58	386.1	0.28390	0.10520	14.910
4	20.29	135.10	1297.0	0.13280	0.10430	22.540
...	...	...	...	...	...	...
564	21.56	142.00	1479.0	0.11590	0.13890	25.450
565	20.13	131.20	1261.0	0.10340	0.09791	23.690
566	16.60	108.30	858.1	0.10230	0.05302	18.980
567	20.60	140.10	1265.0	0.27700	0.15200	25.740
568	7.76	47.92	181.0	0.04362	0.00000	9.456

569 rows × 10 columns

In [14]:
```
Y
```

Out[14]:
```
0      1
1      1
2      1
3      1
4      1
      ..
564    1
565    1
566    1
567    1
568    0
Name: diagnosis, Length: 569, dtype: int64
```

TRAIN TEST SPLIT

In [15]:

```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2,random_stat
```

```
# MODEL TRAINING AND PERFORMANCE


SVC IS THE SVM FOR THE CLASSIFICATION TASKS
IT CONTAINS PARAMETERS

C: DEFAULT=1.0
KERNEL: DEFAULT="RBF"
GAMMA:DEFAULT = 1/(n_features * X.var())
```

In [16]:

```
from sklearn.svm import SVC
from sklearn.metrics import classification_report, f1_score

# Train the model on the train set
model = SVC()
model.fit(X_train, Y_train)

# Predict on the test set
predictions = model.predict(X_test)

# Print prediction results
print("Classification Report:")
print(classification_report(Y_test, predictions))

# Print F1-score
f1 = f1_score(Y_test, predictions)
print("F1 Score:", f1)
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.92      1.00      0.96        71
           1       1.00      0.86      0.92        43

    accuracy                           0.95       114
   macro avg       0.96      0.93      0.94       114
weighted avg       0.95      0.95      0.95       114

F1 Score: 0.9249999999999999
```

hyperparameter tuning with grid search

In [17]:
```
#It is a technique used for hyperparameter tuning in machine learning specific

#grid search CV indentifies the combination of hyperparameters that yeilds the
```

grid search CV has the follwing parameters-

estimator- our model

param_grid= the dictonary we want to pass

refit= refit an estimator using the best found parameters on the whole dataset. default=True

cv= determins the cross validation splitting strategy. default=5

verbose= controls the number of messages. default=0

In [18]:

```
from sklearn.model_selection import GridSearchCV

param_grid = {"C" : [0.1,1,10],
              "gamma" : [1,0.1,0.01],
              "kernel" : ["rbf","linear"]}

##refit parameter will refit an estimator using the best founnd parameters on

grid = GridSearchCV(SVC(),param_grid, refit = True, verbose = 3)

grid.fit(X_train , Y_train)
```

```
Fitting 5 folds for each of 18 candidates, totalling 90 fits
[CV 1/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.637 total time=
0.0s
[CV 2/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.626 total time=
0.0s
[CV 3/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.626 total time=
0.0s
[CV 4/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.626 total time=
0.0s
[CV 5/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.626 total time=
0.0s
[CV 1/5] END .....C=0.1, gamma=1, kernel=linear;, score=0.956 total time=
0.1s
[CV 2/5] END .....C=0.1, gamma=1, kernel=linear;, score=0.901 total time=
0.0s
[CV 3/5] END .....C=0.1, gamma=1, kernel=linear;, score=0.956 total time=
0.1s
[CV 4/5] END .....C=0.1, gamma=1, kernel=linear;, score=0.912 total time=
0.2s
[CV 5/5] END .....C=0.1, gamma=1, kernel=linear;, score=0.901 total time=
0.0s
[CV 1/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.637 total time=
0.0s
[CV 2/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.626 total time=
0.0s
[CV 3/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.626 total time=
0.0s
[CV 4/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.626 total time=
0.0s
[CV 5/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.626 total time=
0.0s
[CV 1/5] END ...C=0.1, gamma=0.1, kernel=linear;, score=0.956 total time=
0.1s
[CV 2/5] END ...C=0.1, gamma=0.1, kernel=linear;, score=0.901 total time=
0.0s
[CV 3/5] END ...C=0.1, gamma=0.1, kernel=linear;, score=0.956 total time=
0.1s
[CV 4/5] END ...C=0.1, gamma=0.1, kernel=linear;, score=0.912 total time=
0.2s
[CV 5/5] END ...C=0.1, gamma=0.1, kernel=linear;, score=0.901 total time=
0.1s
[CV 1/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.637 total time=
0.0s
[CV 2/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.626 total time=
0.0s
[CV 3/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.626 total time=
0.0s
[CV 4/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.626 total time=
0.0s
[CV 5/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.626 total time=
0.0s
[CV 1/5] END ..C=0.1, gamma=0.01, kernel=linear;, score=0.956 total time=
0.1s
[CV 2/5] END ..C=0.1, gamma=0.01, kernel=linear;, score=0.901 total time=
0.0s
[CV 3/5] END ..C=0.1, gamma=0.01, kernel=linear;, score=0.956 total time=
0.1s
```

```
[CV 4/5] END ..C=0.1, gamma=0.01, kernel=linear;, score=0.912 total time=
0.2s
[CV 5/5] END ..C=0.1, gamma=0.01, kernel=linear;, score=0.901 total time=
0.1s
[CV 1/5] END .........C=1, gamma=1, kernel=rbf;, score=0.637 total time=
0.0s
[CV 2/5] END .........C=1, gamma=1, kernel=rbf;, score=0.626 total time=
0.0s
[CV 3/5] END .........C=1, gamma=1, kernel=rbf;, score=0.626 total time=
0.0s
[CV 4/5] END .........C=1, gamma=1, kernel=rbf;, score=0.626 total time=
0.0s
[CV 5/5] END .........C=1, gamma=1, kernel=rbf;, score=0.626 total time=
0.0s
[CV 1/5] END ......C=1, gamma=1, kernel=linear;, score=0.945 total time=
2.4s
[CV 2/5] END ......C=1, gamma=1, kernel=linear;, score=0.923 total time=
1.1s
[CV 3/5] END ......C=1, gamma=1, kernel=linear;, score=0.956 total time=
1.6s
[CV 4/5] END ......C=1, gamma=1, kernel=linear;, score=0.945 total time=
0.7s
[CV 5/5] END ......C=1, gamma=1, kernel=linear;, score=0.923 total time=
1.0s
[CV 1/5] END .......C=1, gamma=0.1, kernel=rbf;, score=0.637 total time=
0.0s
[CV 2/5] END .......C=1, gamma=0.1, kernel=rbf;, score=0.626 total time=
0.0s
[CV 3/5] END .......C=1, gamma=0.1, kernel=rbf;, score=0.626 total time=
0.0s
[CV 4/5] END .......C=1, gamma=0.1, kernel=rbf;, score=0.626 total time=
0.0s
[CV 5/5] END .......C=1, gamma=0.1, kernel=rbf;, score=0.626 total time=
0.0s
[CV 1/5] END ....C=1, gamma=0.1, kernel=linear;, score=0.945 total time=
2.4s
[CV 2/5] END ....C=1, gamma=0.1, kernel=linear;, score=0.923 total time=
0.9s
[CV 3/5] END ....C=1, gamma=0.1, kernel=linear;, score=0.956 total time=
1.5s
[CV 4/5] END ....C=1, gamma=0.1, kernel=linear;, score=0.945 total time=
0.8s
[CV 5/5] END ....C=1, gamma=0.1, kernel=linear;, score=0.923 total time=
0.9s
[CV 1/5] END ......C=1, gamma=0.01, kernel=rbf;, score=0.857 total time=
0.0s
[CV 2/5] END ......C=1, gamma=0.01, kernel=rbf;, score=0.681 total time=
0.0s
[CV 3/5] END ......C=1, gamma=0.01, kernel=rbf;, score=0.670 total time=
0.0s
[CV 4/5] END ......C=1, gamma=0.01, kernel=rbf;, score=0.879 total time=
0.0s
[CV 5/5] END ......C=1, gamma=0.01, kernel=rbf;, score=0.681 total time=
0.0s
[CV 1/5] END ...C=1, gamma=0.01, kernel=linear;, score=0.945 total time=
2.4s
[CV 2/5] END ...C=1, gamma=0.01, kernel=linear;, score=0.923 total time=
1.0s
[CV 3/5] END ....C=1, gamma=0.01, kernel=linear;, score=0.956 total time=
1.6s
[CV 4/5] END ....C=1, gamma=0.01, kernel=linear;, score=0.945 total time=
0.9s
[CV 5/5] END ....C=1, gamma=0.01, kernel=linear;, score=0.923 total time=
1.2s
[CV 1/5] END .........C=10, gamma=1, kernel=rbf;, score=0.637 total time=
0.0s
[CV 2/5] END .........C=10, gamma=1, kernel=rbf;, score=0.626 total time=
0.0s
[CV 3/5] END .........C=10, gamma=1, kernel=rbf;, score=0.626 total time=
0.0s
[CV 4/5] END .........C=10, gamma=1, kernel=rbf;, score=0.626 total time=
0.0s
[CV 5/5] END .........C=10, gamma=1, kernel=rbf;, score=0.626 total time=
0.0s
[CV 1/5] END ......C=10, gamma=1, kernel=linear;, score=0.934 total time=
5.8s
[CV 2/5] END ......C=10, gamma=1, kernel=linear;, score=0.912 total time=
2.2s
[CV 3/5] END ......C=10, gamma=1, kernel=linear;, score=0.978 total time=
4.0s
[CV 4/5] END ......C=10, gamma=1, kernel=linear;, score=0.934 total time=
3.7s
[CV 5/5] END ......C=10, gamma=1, kernel=linear;, score=0.923 total time=
7.0s
[CV 1/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.637 total time=
0.0s
[CV 2/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.626 total time=
0.0s
[CV 3/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.626 total time=
0.0s
[CV 4/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.626 total time=
0.0s
[CV 5/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.626 total time=
0.0s
[CV 1/5] END ....C=10, gamma=0.1, kernel=linear;, score=0.934 total time=
5.4s
[CV 2/5] END ....C=10, gamma=0.1, kernel=linear;, score=0.912 total time=
2.2s
[CV 3/5] END ....C=10, gamma=0.1, kernel=linear;, score=0.978 total time=
3.5s
[CV 4/5] END ....C=10, gamma=0.1, kernel=linear;, score=0.934 total time=
3.5s
[CV 5/5] END ....C=10, gamma=0.1, kernel=linear;, score=0.923 total time=
6.3s
[CV 1/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.857 total time=
0.0s
[CV 2/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.890 total time=
0.0s
[CV 3/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.934 total time=
0.0s
[CV 4/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.879 total time=
0.0s
[CV 5/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.681 total time=
0.0s
```

```
[CV 1/5] END ...C=10, gamma=0.01, kernel=linear;, score=0.934 total time=
5.3s
[CV 2/5] END ...C=10, gamma=0.01, kernel=linear;, score=0.912 total time=
2.1s
[CV 3/5] END ...C=10, gamma=0.01, kernel=linear;, score=0.978 total time=
3.6s
[CV 4/5] END ...C=10, gamma=0.01, kernel=linear;, score=0.934 total time=
3.6s
[CV 5/5] END ...C=10, gamma=0.01, kernel=linear;, score=0.923 total time=
6.5s
```

Out[18]:

```
▸ GridSearchCV
▸ estimator: SVC
    ▸ SVC
```

What fit does is a bit more involved than usual. first, it runs the same loop with cross validation, to find the best parameter combination, once it has the best combination, it runs fit again on all data passed to fit (without cross-validation),to build a single new model using the best parameter setting

In [19]:

```python
#to find the best parameter after tuning

print(grid.best_params_)

#to find how our model looks after hyper parameter tuning
print(grid.best_estimator_)
```

```
{'C': 1, 'gamma': 1, 'kernel': 'linear'}
SVC(C=1, gamma=1, kernel='linear')
```

In [20]:

```python
grid_predictions = grid.predict(X_test)
f1_grid = f1_score(Y_test, grid_predictions)
print("F1 Score (Grid Search):", f1_grid)
```

```
F1 Score (Grid Search): 0.9761904761904763
```

In [ ]: