# **Breast Cancer Classification**

## **Devanshi Nigam**

Aim: The goal of the project is to predict the type of cancer cell (benign or malignant).

**ML Model:** To achieve this, I utilized Python and the Random Forest Classification algorithm.

**Dataset:** <a href="https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data">https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data</a>

The datasets contain medical dataset of breast cancer patients with different parameters such as radius, texture, perimeter, area, smoothness, compactness

#### **Importing Dataset**

In [4]:	<pre>df = pd.read_csv('data-cancer.csv')</pre>
In [5]:	df.head()
Out[5]:	

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_m
0	842302	М	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30
1	842517	М	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0
2	84300903	М	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1!
3	84348301	М	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2
4	84358402	М	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1!

5 rows × 33 columns

#### Clearing data

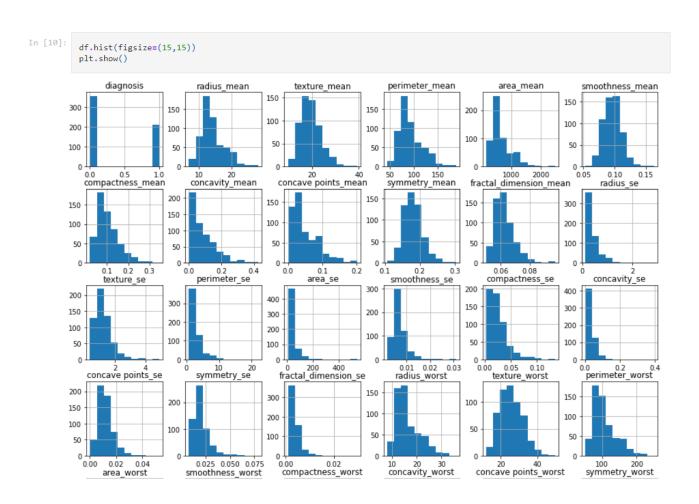
```
In [7]: columns_to_drop = [0,32]
    df.drop(df.columns[columns_to_drop], axis=1, inplace=True)

In [8]: # Diagnosis (M = malignant = 1, B = benign = 1)
    df['diagnosis'] = df['diagnosis'].map({'M': 1, 'B': 0})
    df['diagnosis'].value_counts()
Out[8]: 0 357
    1 212
    Name: diagnosis, dtype: int64
```

ar.ae	scribe()							
	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000
mean	0.372583	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799
std	0.483918	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720
min	0.000000	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000
25%	0.000000	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560
50%	0.000000	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540
75%	1.000000	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700
max	1.000000	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800

### **Data Visualization:**

## 1) Histogram:



# 2) Correlation:

In [11]:

plt.figure(figsize=[20,20])
sns.heatmap(df.corr(),annot=True)

Out[11]: <AxesSubplot:

]: <axessubplo< th=""><th>t:&gt;</th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th></axessubplo<>	t:>																																
diagnosis	- 1	0.73	0.42	0.74	0.71	0.36	0.6	0.7	0.78	0.334	0.013	0.57-0	0.008	30.56	0.55	0.067	0.29	0.25	0.410	2900.0	0.078	0.78	0.46	0.78 0	.73 0.4	2 0.5	9 0.66	0.79	0.42	0.32		-1.0	
radius_mean	-0.73	1		1	0.99				0.82	0.15	0.31	0.65	0.097		0.74	-0.22		0.19	0.38	-0.1 -	0.043	0.97	0.3	0.97 0	.94 0.1	.2 0.4	1 0.53	0.74		0.007			
texture_mean	0.42	0.32	1	0.33	0.32	0.023		0.3	0.29	0.0714	0.076	0.28	0.39	0.28	0.260	0.0066		0.14	0.160	.0 <b>0</b> 91	0.054	0.35	0.91	0.36 0	34 0.0	78 0.2		0.3		0.12			
perimeter_mean	0.74	1		1	0.99			0.72	0.85	0.18	0.26	0.69	0.087		0.74	-0.2		0.23	0.41	0.0820	0.005	0.97	0.3	0.97 0	.94 0.1	.5 0.4	6 0.56	0.77		0.051			
area_mean	0.71	0.99		0.99	1	0.18			0.82	0.15	0.28	0.73	0.066	0.73	0.8	-0.17		0.21	0.374	0.072-	-0.02	0.96	0.29	0.96 0	.96 0.1	2 0.3	9 0.51	0.72	0.140	0.0037		-0.8	
smoothness_mean	0.36	0.17	-0.023	0.21	0.18	1	0.66	0.52	0.55		0.58		0.068			0.33	0.32	0.25	0.38			0.21	1.036	0.24 0	.21 0.8	0.4	7 0.43	0.5		0.5			
compactness_mean	0.6	0.51		0.56	0.5	0.66	1	0.88	0.83	0.6	0.57	0.5	0.046	0.55		0.14	0.74		0.64	0.23	0.51	0.54	0.25	0.59 0	51 0.5	7 0.8	7 0.82	0.82		0.69			
concavity_mean		_															-																
concave points_mean											-												-				_		_				
symmetry_mean										_										-												-0.6	
fractal_dimension_mean								_			_									_							6 0.35	_					
radius_se		_										_											-					_					
texture_se							_												_									_					
perimeter_se					_																											-0.4	
area_se		_																			-							_					
smoothness_se compactness_se		_		_			_				_									- 1	_	_	-	_	_		_	_					
compactness_se concavity_se																	-																
concavity_se									_									-										_					
symmetry_se		_																	_								_	_				-0.2	
fractal dimension se										-						-		_	-	_													
radius werst								_	_	-	-			_		_					_							_					
texture_worst											-		_									_	-										
perimeter worst								_	_		-											-	_				_	_				-0.0	
area_worst	0.73	0.94		0.94	0.96	0.21			0.81	0.18	0.23	0.75	D.D83	0.73	0.81	-0.18		0.19	0.34	-0.114	0.023	0.98	0.35	0.98	1 0.2	1 0.4	4 0.54	0.75		0.08		- 4.0	
smoothness worst	0.42	0.12	0.078	0.15	0.12	0.81		0.45	0.45	0.43	0.5	0.14	0.074	0.13	0.13	0.31		0.17	0.224	0.013	0.17	0.22	0.23	0.24 0	.21 1	0.5				0.62			
compactness_worst	0.59	0.41			0.39	0.47	0.87	0.75	0.67		0.46	0.29	0.092			0.056	0.68		0.45	0.05			0.36	0.53 0	44 0.5	7 1	0.89	0.8		0.81			
concavity_worst	0.66	0.53		0.56		0.43	0.82	0.88	0.75			0.38 4	0.069			0.058	0.64		0.55 (	0.037	0.38	0.57	0.37			2 0.8	9 1	0.86		0.69			
concave points_worst	- 0.79	0.74		0.77	0.72	0.5	0.82	0.86	0.91		0.18	0.53	-0.12			-0.1	0.48	0.44	0.6		0.22	0.79	0.36	0.82 0	.75 0.9	5 0.8	0.96	1		0.51		0.	2
symmetry_worst	0.42	0.16		0.19	0.14		0.51	0.41	0.38	0.7	0.33	0.095	0.13	0.11	0.074	-0.11		0.2	0.14		0.11	0.24	0.23	0.27 0	.21 0.4	9 0.6	1 0.53	0.5	1	0.54			
fractal_dimension_worst	0.32	0.0071		0.051	0.0037					0.44	0.77	0.05	0.046	0.085		0.1	0.59		0.31	0.078	0.59			0.14 0	.08 0.0	2 0.8	0.69	0.51	0.54	1			
	- sysoubego	radius mean -	texture_mean_	perimeter mean -	area_mean -	smoothness mean -	compactness_mean _	concavity_mean -	concave points mean -	symmetry_mean -	ectal_dimension_mean -	radius_se -	escantes.	perimeter se -	- School	- as ssautpooms	ompactness se -	concavity_se -	concave points_se -	symmetry se	fractal_dimension_se	- radius_worst	besture_worst_	perimeter_worst -	area worst	compactness worst	concevity worst -	concave points_worst -	symmetry_worst -	hactal_dimension_worst -			
									-		ž																			£			

## **Data Analysis:**

For data analysis, I used machine learning model Random Forest Classification algorithm.

### ML Modeling

```
In [12]:
          from sklearn.model_selection import train_test_split
          x\_train, \ x\_test, \ y\_train, \ y\_test = train\_test\_split(df.iloc[:,1:], \ df['diagnosis'], \ test\_size = 0.2, \ random\_state = 42)
In [13]: from sklearn preprocessing import MinMaxScaler
          normal = MinMaxScaler()
In [14]:
          #Fitting Data
          normal fit = normal.fit(x train)
          new xtrain = normal fit.transform(x train)
          new_xtest = normal_fit.transform(x_test)
          #print(new_xtrain)
          #print(new_xtest)
          Using Random Forest Classifier
          from sklearn.ensemble import RandomForestClassifier
          from sklearn.metrics import classification_report
          rand = RandomForestClassifier()
In [16]:
          #Fitting Data
          fit_rand = rand.fit(new_xtrain, y_train)
           #predicting score
           rand_score = rand.score(new_xtest, y_test)
          print('Score of model is : ', rand_score*100,'%')
        Score of model is : 96.49122807017544 %
          Error Detection
 In [17]: #Display Error
           #Calculating Mean Squared Error
           from sklearn.metrics import mean_squared_error
           Yhat = rand.predict(new_xtest)
           {\tt rand\_MSE = mean\_squared\_error(y\_test, Yhat)}
           rand_RMSE = np.sqrt(rand_MSE)
print('Mean Square Error is: ', rand_MSE)
           print('Root Mean Square Error is: ', rand_RMSE)
         Mean Square Error is: 0.03508771929824561
         Root Mean Square Error is: 0.1873171623163388
          Prediction
 In [18]: x_predict = list(rand.predict(x_test))
           predicted_df = {'predicted_values': x_predict,'original_values': y_test}
           print(classification_report(x_predict, y_test))
                       precision recall f1-score support
                           0.00
                                   0.00 0.00
                    0
                           1.00
                                     0.38 0.55
                                                           114
                       0.38
0.50 0.19 0.27
1.00 0.38 0.55
             accuracy
                                                         114
114
114
            macro avg
         weighted avg
```