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: https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data

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

Importing Dataset

In [4]:
```
df = pd.read_csv('data-cancer.csv')
```

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

Out[5]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_m
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2
4	84358402	M	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
```

In [9]:
```
df.describe()
```

Out[9]:

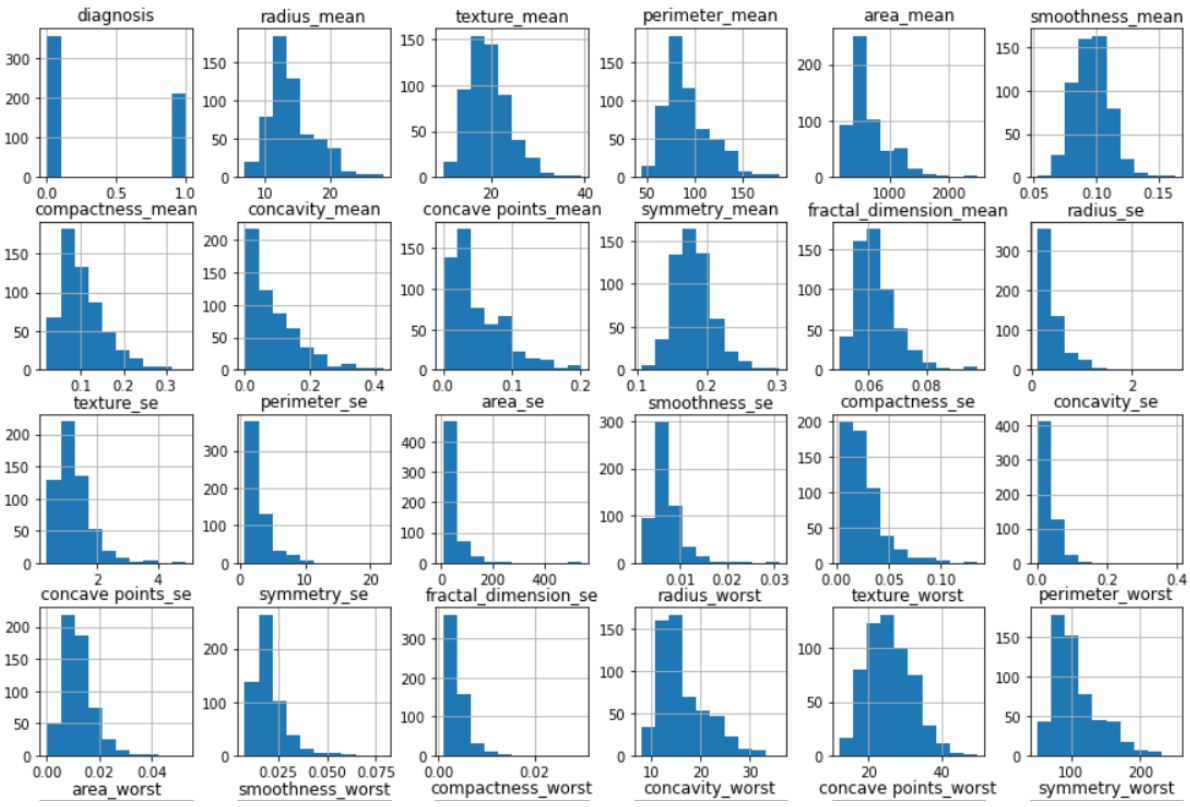
	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000
mean	0.372583	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799
std	0.483918	3.524049	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

8 rows × 31 columns

Data Visualization:

1) Histogram:

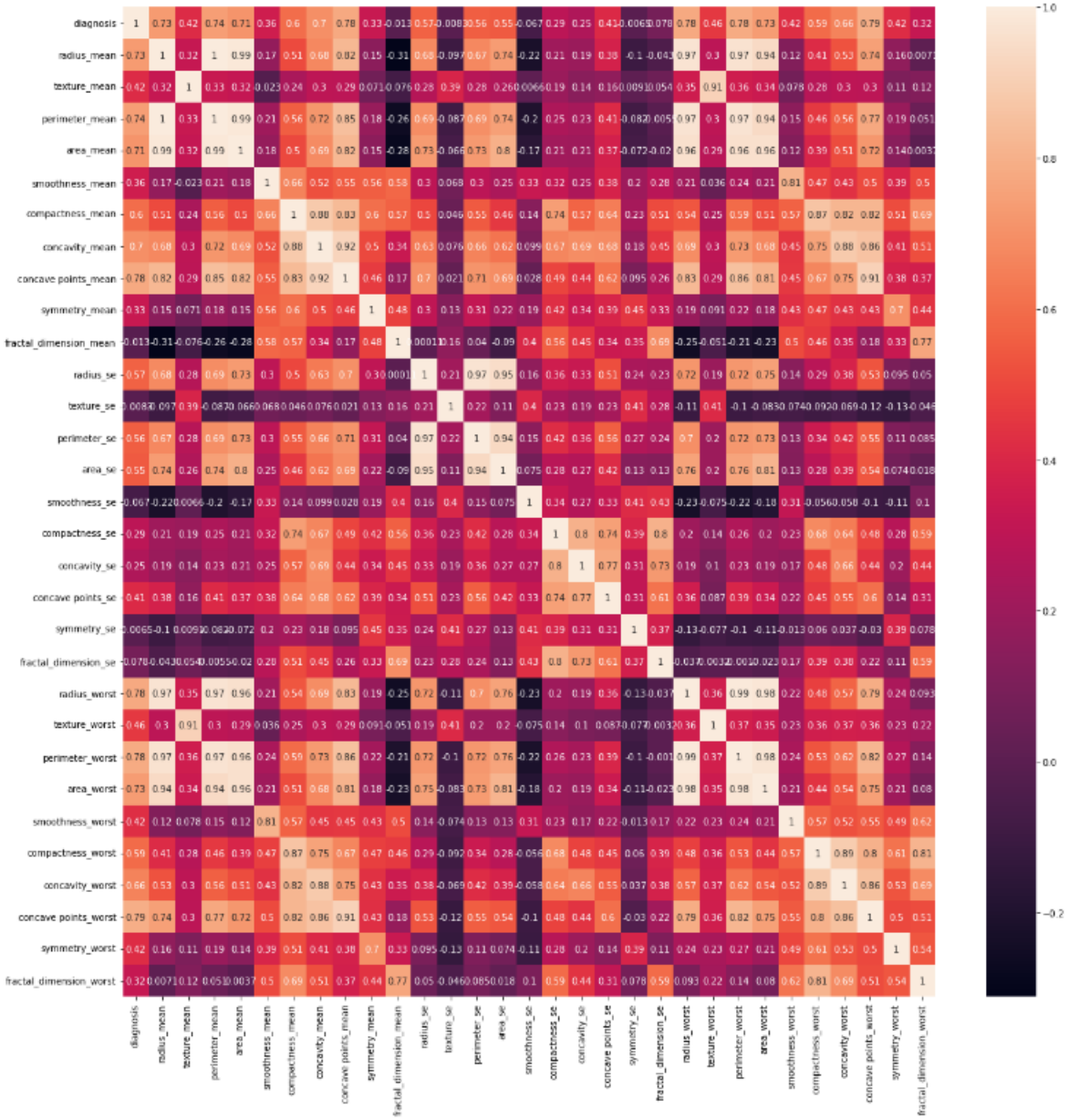
In [10]:
```
df.hist(figsize=(15,15))
plt.show()
```



2) Correlation:

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

Out[11]: <AxesSubplot:>



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

In [15]:
```
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)
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       0.00      0.00      0.00         0
           1       1.00      0.38      0.55       114

    accuracy                           0.38       114
   macro avg       0.50      0.19      0.27       114
weighted avg       1.00      0.38      0.55       114
```