Classification of Breast Cancer : Benign or Malignant

To classify if the cancer diagnosis is benign or malignant based on several observations/features
30 features are used, examples:

- 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)

Number of Instances: 569 Class Distribution: 212 Malignant, 357 Benign Target class:

- Malignant
- Benign

Importing Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

Reading the breast cancer dataframe

In [2]:
```
data_cancer = pd.read_csv("D:\\python\\final pro csv\\datasets_21088_27183_breast-cancer.csv")
```

Exploring Data

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

Out[3]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	s
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

In [4]:
```
data_cancer.columns
```

Out[4]:
```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mea
n',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mea
n',
       '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_s
e',
       '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', 'Unnamed: 32'],
      dtype='object')
```

In [5]:
```
data_cancer.shape
```

Out[5]:
```
(569, 33)
```

In [6]:
```
data_cancer.describe().T
```

Out[6]:

	count	mean	std	min	
id	569.0	3.037183e+07	1.250206e+08	8670.000000	869218.0
radius_mean	569.0	1.412729e+01	3.524049e+00	6.981000	11.7
texture_mean	569.0	1.928965e+01	4.301036e+00	9.710000	16.1
perimeter_mean	569.0	9.196903e+01	2.429898e+01	43.790000	75.1
area_mean	569.0	6.548891e+02	3.519141e+02	143.500000	420.3
smoothness_mean	569.0	9.636028e-02	1.406413e-02	0.052630	0.0
compactness_mean	569.0	1.043410e-01	5.281276e-02	0.019380	0.0
concavity_mean	569.0	8.879932e-02	7.971981e-02	0.000000	0.0
concave points_mean	569.0	4.891915e-02	3.880284e-02	0.000000	0.0
symmetry_mean	569.0	1.811619e-01	2.741428e-02	0.106000	0.1
fractal_dimension_mean	569.0	6.279761e-02	7.060363e-03	0.049960	0.0
radius_se	569.0	4.051721e-01	2.773127e-01	0.111500	0.2
texture_se	569.0	1.216853e+00	5.516484e-01	0.360200	0.8
perimeter_se	569.0	2.866059e+00	2.021855e+00	0.757000	1.6
area_se	569.0	4.033708e+01	4.549101e+01	6.802000	17.8
smoothness_se	569.0	7.040979e-03	3.002518e-03	0.001713	0.0
compactness_se	569.0	2.547814e-02	1.790818e-02	0.002252	0.0
concavity_se	569.0	3.189372e-02	3.018606e-02	0.000000	0.0
concave points_se	569.0	1.179614e-02	6.170285e-03	0.000000	0.0
symmetry_se	569.0	2.054230e-02	8.266372e-03	0.007882	0.0
fractal_dimension_se	569.0	3.794904e-03	2.646071e-03	0.000895	0.0
radius_worst	569.0	1.626919e+01	4.833242e+00	7.930000	13.0
texture_worst	569.0	2.567722e+01	6.146258e+00	12.020000	21.0
perimeter_worst	569.0	1.072612e+02	3.360254e+01	50.410000	84.1
area_worst	569.0	8.805831e+02	5.693570e+02	185.200000	515.3
smoothness_worst	569.0	1.323686e-01	2.283243e-02	0.071170	0.1
compactness_worst	569.0	2.542650e-01	1.573365e-01	0.027290	0.1
concavity_worst	569.0	2.721885e-01	2.086243e-01	0.000000	0.1
concave points_worst	569.0	1.146062e-01	6.573234e-02	0.000000	0.0
symmetry_worst	569.0	2.900756e-01	6.186747e-02	0.156500	0.2
fractal_dimension_worst	569.0	8.394582e-02	1.806127e-02	0.055040	0.0
Unnamed: 32	0.0	NaN	NaN	NaN	

Checking null values

In [7]:
```
data_cancer.isnull().sum()
```

Out[7]:
```
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
```

As we have seen there are no null entries except Unnamed:32 so we will delete it later before training

Mapping Diagnosis variable which is our Target variable to 0 and 1 : 1 for Malignant and 0 for Benign

In [8]:
```
data_cancer.loc[:,'diagnosis'] = data_cancer.diagnosis.map({'M':1, 'B':0
})
```

In [9]:
```
data_cancer.head()
```

Out[9]:

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

5 rows × 33 columns

In [10]:
```
data_cancer.tail()
```

Out[10]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	s
564	926424	1	21.56	22.39	142.00	1479.0	
565	926682	1	20.13	28.25	131.20	1261.0	
566	926954	1	16.60	28.08	108.30	858.1	
567	927241	1	20.60	29.33	140.10	1265.0	
568	92751	0	7.76	24.54	47.92	181.0	

5 rows × 33 columns

Checking the correlation among different features and target variable diagnosis

Create a list of potential Features to measure correlation

In [11]:
```
features = ['radius_mean', 'texture_mean', 'area_mean', 'perimeter_mean',
'smoothness_mean']
```

In [12]:
```
for f in features:
    related = data_cancer['diagnosis'].corr(data_cancer[f])
    print("%s: %f" % (f,related))
```

```
radius_mean: 0.730029
texture_mean: 0.415185
area_mean: 0.708984
perimeter_mean: 0.742636
smoothness_mean: 0.358560
```

Data Visualization

Visualization of important features in relation to target variable diagnosis to see on which features it is more related

plotting the correlation coefficients of each feature with "diagnosis".

In [13]:
```
cols= ['id', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mea
n',
       '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_s
e',
       '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']
```

In [14]:
```
for f in cols:
    correlations = data_cancer['diagnosis'].corr(data_cancer[f])
    print("%s: %f" % (f,correlations))
```

```
id: 0.039769
radius_mean: 0.730029
texture_mean: 0.415185
perimeter_mean: 0.742636
area_mean: 0.708984
smoothness_mean: 0.358560
compactness_mean: 0.596534
concavity_mean: 0.696360
concave points_mean: 0.776614
symmetry_mean: 0.330499
fractal_dimension_mean: -0.012838
radius_se: 0.567134
texture_se: -0.008303
perimeter_se: 0.556141
area_se: 0.548236
smoothness_se: -0.067016
compactness_se: 0.292999
concavity_se: 0.253730
concave points_se: 0.408042
symmetry_se: -0.006522
fractal_dimension_se: 0.077972
radius_worst: 0.776454
texture_worst: 0.456903
perimeter_worst: 0.782914
area_worst: 0.733825
smoothness_worst: 0.421465
compactness_worst: 0.590998
concavity_worst: 0.659610
concave points_worst: 0.793566
symmetry_worst: 0.416294
fractal_dimension_worst: 0.323872
```

In [15]:
```
correlations = [data_cancer['diagnosis'].corr(data_cancer[f]) for f in co
ls ]
```

In [16]:
```
len(correlations), len(cols)
```

Out[16]:
```
(31, 31)
```

The number of selected features and the correlations calculated are the same i.e. "31".

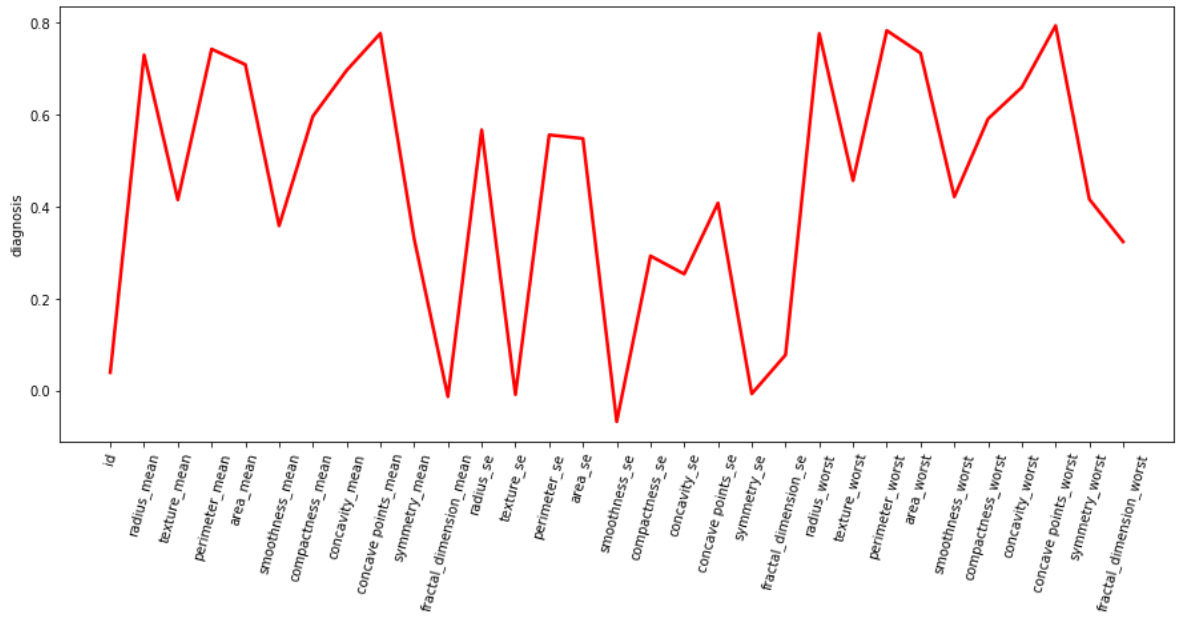
Looking at the Visualization of important features in relation to target variable diagnosis to see on which features it is more related

In [17]:
```
def plot_dataframe(data_cancer, y_label):
    color='red'
    fig = plt.gcf()
    fig.set_size_inches(15,6)
    plt.ylabel(y_label)

    graph = data_cancer.correlation.plot(linewidth=2.5, color=color)
    graph.set_xticks(data_cancer.index)
    graph.set_xticklabels(data_cancer.attributes, rotation=75);
    plt.show()
```

In [18]:
```
df2 = pd.DataFrame({'attributes': cols, 'correlation': correlations})
```

In [19]:
```
plot_dataframe(df2, 'diagnosis')
```



Data Cleaning: Handling Missing Data

As we have seen Unnamed: 32 is a column with full of NAN and some of the other features such as id is also non contributing feature in cancer prediction so we will be dropping those features before training the model.

In [20]:
```
unwantedcolumnlist=["diagnosis","Unnamed: 32","id"]
```

Convert data for 'Classification'

In [21]:
```
X = data_cancer.drop(unwantedcolumnlist,axis=1)
```

In [22]:
```
y = data_cancer[['diagnosis']]
```

In [23]:
```
Y=y.copy()
```

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

Out[24]:

	diagnosis
0	1
1	1
2	1
3	1
4	1

In [25]:
```
Y.columns
```

Out[25]:
```
Index(['diagnosis'], dtype='object')
```

In [26]:
```
clean_data= X.copy()
```

In [27]:
```
clean_data.head()
```

Out[27]:

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

5 rows × 30 columns

In [28]:
```
X.columns
```

Out[28]:
```
Index(['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_s
e',
       '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')
```

split training and testing dataset using sklearn to X_train, X_test,y_train,y_test

In [29]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.20, random_state=5)
```

In [30]:
```
#type(X_train)
#type(X_test)
#type(Y_train)
type(Y_test)
```

Out[30]:
```
pandas.core.frame.DataFrame
```

In [31]:
```
#Y_train.describe()
Y_test.describe()
```

Out[31]:

	diagnosis
count	114.000000
mean	0.421053
std	0.495908
min	0.000000
25%	0.000000
50%	0.000000
75%	1.000000
max	1.000000

In [32]:
```
#X_train.describe()
X_test.describe()
```

Out[32]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mea
count	114.000000	114.000000	114.000000	114.000000	114.00000
mean	14.381947	20.123947	93.961053	684.042105	0.09643
std	3.789884	4.418402	26.331994	393.879895	0.01413
min	8.219000	10.720000	53.270000	203.900000	0.06576
25%	11.760000	17.362500	75.817500	428.700000	0.08488
50%	13.530000	19.595000	86.765000	567.400000	0.09548
75%	16.152500	22.975000	107.375000	814.300000	0.10605
max	27.420000	30.720000	186.900000	2501.000000	0.14250

8 rows × 30 columns

In [33]:
```
diagnosis_class = DecisionTreeClassifier(max_leaf_nodes= 15 , random_state=0)
```

In [34]:
```
type(diagnosis_class)
```

Out[34]:
```
sklearn.tree._classes.DecisionTreeClassifier
```

In [35]:
```
print(diagnosis_class)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gin
i',
                       max_depth=None, max_features=None, max_leaf_nodes=
15,
                       min_impurity_decrease=0.0, min_impurity_split=Non
e,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecate
d',
                       random_state=0, splitter='best')
```

In [36]:
```
diagnosis_class.fit(X_train,Y_train)
```

Out[36]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gin
i',
                       max_depth=None, max_features=None, max_leaf_nodes=
15,
                       min_impurity_decrease=0.0, min_impurity_split=Non
e,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecate
d',
                       random_state=0, splitter='best')
```

predict on test

In [37]:
```
predict= diagnosis_class.predict(X_test)
```

In [38]:
```
predict[:15]
```

Out[38]:
```
array([1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0], dtype=int64)
```

In [39]:
```
Y_test['diagnosis'][:15]
```

Out[39]:
```
28     1
163    0
123    0
361    0
549    0
339    1
286    0
354    0
421    0
124    0
543    0
537    0
567    1
555    0
511    0
Name: diagnosis, dtype: int64
```

In [40]:
```
accuracy_score(y_true=Y_test,y_pred=predict)
```

Out[40]:
```
0.9473684210526315
```

That's Great we have achieved 94% accuracy to detect malignant or benign breast cancer.