Projet_classification_final (1)

March 9, 2023

[1]:
```
import pandas as pd
from sklearn.feature_selection import SelectFromModel
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn import tree
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.model_selection import cross_val_score
from sklearn.feature_selection import SelectKBest, mutual_info_classif, chi2,␣
↪f_classif
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score , confusion_matrix, precision_score,␣
↪recall_score, classification_report
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import GridSearchCV
from sklearn.tree import export_graphviz
import warnings
warnings.filterwarnings("ignore")
from sklearn.tree import export_graphviz
import graphviz
from sklearn.model_selection import LeaveOneOut
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import RFECV
import matplotlib.pyplot as plt
```

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

[3]:
```
data.head(3)
```

[3]:
```
         id diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
0    842302         M        17.99         10.38           122.8     1001.0
```

```
1    842517         M        20.57         17.77           132.9     1326.0
2  84300903         M        19.69         21.25           130.0     1203.0

   smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \
0          0.11840           0.27760          0.3001              0.14710
1          0.08474           0.07864          0.0869              0.07017
2          0.10960           0.15990          0.1974              0.12790

   …  texture_worst  perimeter_worst  area_worst  smoothness_worst  \
0  …          17.33            184.6      2019.0            0.1622
1  …          23.41            158.8      1956.0            0.1238
2  …          25.53            152.5      1709.0            0.1444

   compactness_worst  concavity_worst  concave points_worst  symmetry_worst  \
0             0.6656           0.7119                0.2654          0.4601
1             0.1866           0.2416                0.1860          0.2750
2             0.4245           0.4504                0.2430          0.3613

   fractal_dimension_worst  Unnamed: 32
0                  0.11890          NaN
1                  0.08902          NaN
2                  0.08758          NaN

[3 rows x 33 columns]
```

[4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   id                       569 non-null    int64
 1   diagnosis                569 non-null    object
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
```

```
 16  smoothness_se           569 non-null    float64
 17  compactness_se          569 non-null    float64
 18  concavity_se            569 non-null    float64
 19  concave points_se       569 non-null    float64
 20  symmetry_se             569 non-null    float64
 21  fractal_dimension_se    569 non-null    float64
 22  radius_worst            569 non-null    float64
 23  texture_worst           569 non-null    float64
 24  perimeter_worst         569 non-null    float64
 25  area_worst              569 non-null    float64
 26  smoothness_worst        569 non-null    float64
 27  compactness_worst       569 non-null    float64
 28  concavity_worst         569 non-null    float64
 29  concave points_worst    569 non-null    float64
 30  symmetry_worst          569 non-null    float64
 31  fractal_dimension_worst 569 non-null    float64
 32  Unnamed: 32             0 non-null      float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```

[5]:

```
data.describe()
```

[5]:

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

       smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \
count       569.000000        569.000000      569.000000           569.000000
mean          0.096360          0.104341        0.088799             0.048919
std           0.014064          0.052813        0.079720             0.038803
min           0.052630          0.019380        0.000000             0.000000
25%           0.086370          0.064920        0.029560             0.020310
50%           0.095870          0.092630        0.061540             0.033500
75%           0.105300          0.130400        0.130700             0.074000
max           0.163400          0.345400        0.426800             0.201200

       symmetry_mean  …  texture_worst  perimeter_worst   area_worst  \
count     569.000000  …     569.000000       569.000000   569.000000
mean        0.181162  …      25.677223       107.261213   880.583128
std         0.027414  …       6.146258        33.602542   569.356993
min         0.106000  …      12.020000        50.410000   185.200000
```

```
25%          0.161900  ...       21.080000        84.110000   515.300000
50%          0.179200  ...       25.410000        97.660000   686.500000
75%          0.195700  ...       29.720000       125.400000  1084.000000
max          0.304000  ...       49.540000       251.200000  4254.000000

       smoothness_worst  compactness_worst  concavity_worst  \
count        569.000000         569.000000       569.000000
mean           0.132369           0.254265         0.272188
std            0.022832           0.157336         0.208624
min            0.071170           0.027290         0.000000
25%            0.116600           0.147200         0.114500
50%            0.131300           0.211900         0.226700
75%            0.146000           0.339100         0.382900
max            0.222600           1.058000         1.252000

       concave points_worst  symmetry_worst  fractal_dimension_worst  \
count            569.000000      569.000000               569.000000
mean               0.114606        0.290076                 0.083946
std                0.065732        0.061867                 0.018061
min                0.000000        0.156500                 0.055040
25%                0.064930        0.250400                 0.071460
50%                0.099930        0.282200                 0.080040
75%                0.161400        0.317900                 0.092080
max                0.291000        0.663800                 0.207500

       Unnamed: 32
count          0.0
mean           NaN
std            NaN
min            NaN
25%            NaN
50%            NaN
75%            NaN
max            NaN

[8 rows x 32 columns]
```

[6]:
```
data.columns
```

[6]:
```
Index(['id', 'diagnosis', '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_se',
       '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')
```

1 Data Preprocessing

1.0.1 Missing values

[7]:
```
data.isna().sum()
```

[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
```

[8]:
```
# suppression de la colonne Unnamed : 32
data.drop('Unnamed: 32',axis = 1,inplace = True)
```

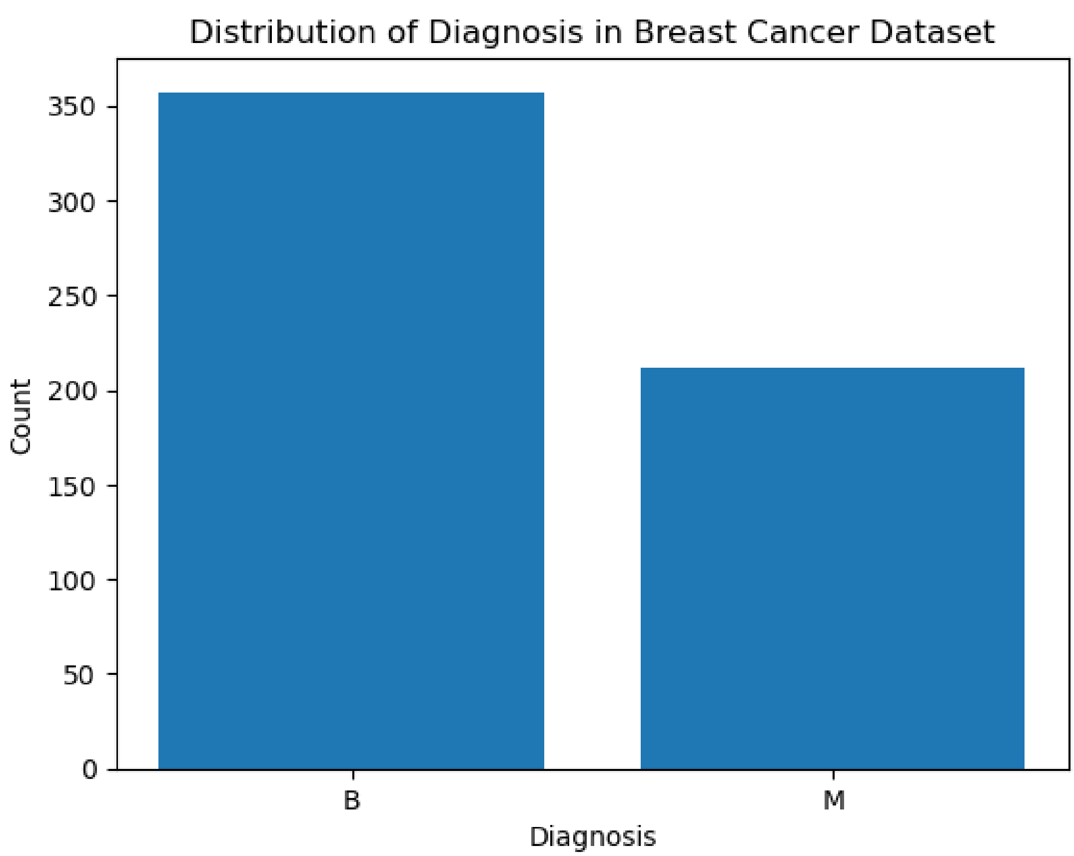
[9]:
```
data.drop('id',axis = 1,inplace = True)
```

[10]:
```
data.columns
```

[10]:
```
Index(['diagnosis', '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_se',
       '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')
```

1.0.2 Unbalanced data

[11]:
```
counts = data['diagnosis'].value_counts()
plt.bar(counts.index, counts.values)
plt.xlabel('Diagnosis')
plt.ylabel('Count')
plt.title('Distribution of Diagnosis in Breast Cancer Dataset')
plt.show()
```

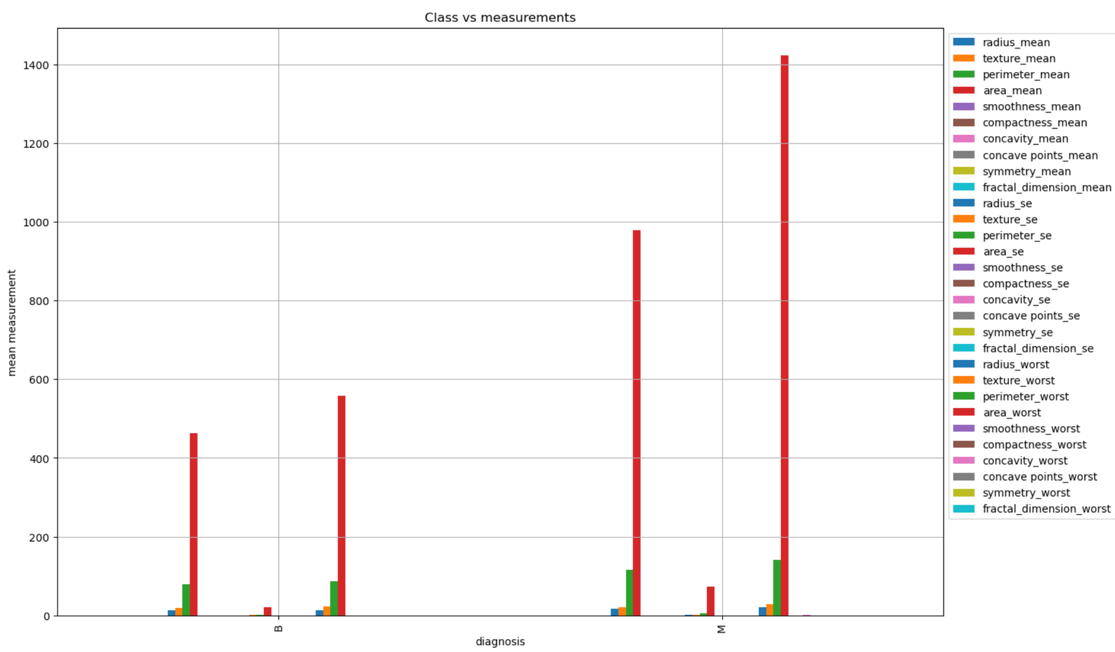


1.0.3 Normalization

[12]:
```
X = data.drop('diagnosis',axis=1)
y = data['diagnosis']
```

[13]:
```
data.groupby(by = "diagnosis").mean()
data.groupby(by="diagnosis").mean().plot(kind="bar", figsize=(15,10))
plt.title('Class vs measurements')
plt.ylabel('mean measurement')
plt.grid(True)
plt.legend(loc="upper left", bbox_to_anchor=(1,1))
```

[13]: <matplotlib.legend.Legend at 0x220339e2730>



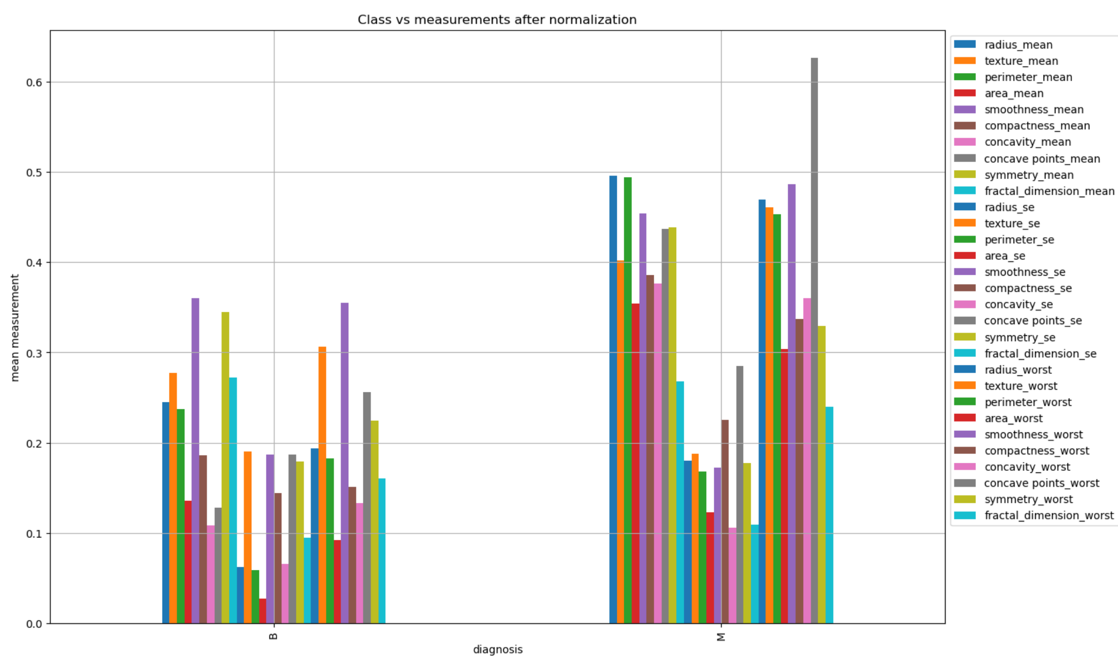
[14]:
```
scaler = MinMaxScaler(feature_range=(0, 1))
X = scaler.fit_transform(X)
np.set_printoptions(precision=5)
```

[15]:
```
#Save new dataset
feature_columns = ['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_se',
        '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']

df = pd.DataFrame(X, columns = feature_columns)
X = pd.DataFrame(X, columns = feature_columns)
df['diagnosis'] = y
```

[16]:
```
df.groupby(by = "diagnosis").mean()
df.groupby(by="diagnosis").mean().plot(kind="bar", figsize=(15,10))
plt.title('Class vs measurements after normalization')
plt.ylabel('mean measurement')
plt.grid(True)
plt.legend(loc="upper left", bbox_to_anchor=(1,1))
```

[16]: <matplotlib.legend.Legend at 0x2202d2f3fd0>



[17]:
```
df['diagnosis'] = df['diagnosis'].map({'B':0,'M':1})
```

[18]:
```
# split of the data
X = df.drop('diagnosis',axis = True)
y = df['diagnosis']
```

1.0.4 Data balancing

[19]:
```
# Under sampler
```

[20]:
```
pip install imblearn
```

```
Requirement already satisfied: imblearn in c:\users\21261\anaconda3\lib\site-
packages (0.0)
Requirement already satisfied: imbalanced-learn in
c:\users\21261\anaconda3\lib\site-packages (from imblearn) (0.10.1)
Requirement already satisfied: scipy>=1.3.2 in
c:\users\21261\anaconda3\lib\site-packages (from imbalanced-learn->imblearn)
(1.9.1)
Requirement already satisfied: joblib>=1.1.1 in
c:\users\21261\anaconda3\lib\site-packages (from imbalanced-learn->imblearn)
(1.2.0)
Requirement already satisfied: numpy>=1.17.3 in
c:\users\21261\anaconda3\lib\site-packages (from imbalanced-learn->imblearn)
(1.21.5)
Requirement already satisfied: scikit-learn>=1.0.2 in
c:\users\21261\anaconda3\lib\site-packages (from imbalanced-learn->imblearn)
(1.0.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
c:\users\21261\anaconda3\lib\site-packages (from imbalanced-learn->imblearn)
(2.2.0)
Note: you may need to restart the kernel to use updated packages.
```

[84]:
```
from imblearn.under_sampling import RandomUnderSampler
rus = RandomUnderSampler(random_state=0)
X_resampled, y_resampled = rus.fit_resample(X, y)
```

[85]:
```
X_resampled
```

[85]:
```
     radius_mean  texture_mean  perimeter_mean  area_mean  smoothness_mean  \
0       0.308060      0.425769        0.297975   0.177094         0.314977
1       0.264991      0.293879        0.249050   0.146554         0.282567
2       0.373373      0.355090        0.361620   0.227953         0.390358
3       0.082967      0.241123        0.079331   0.038515         0.462851
4       0.223816      0.252959        0.213461   0.117413         0.407240
..           …             …               …          …                …
419     0.659709      0.520122        0.685578   0.510498         0.517017
420     0.690000      0.428813        0.678668   0.566490         0.526948
421     0.622320      0.626987        0.604036   0.474019         0.407782
422     0.455251      0.621238        0.445788   0.303118         0.288165
423     0.644564      0.663510        0.665538   0.475716         0.588336

     compactness_mean  concavity_mean  concave points_mean  symmetry_mean  \
0            0.176676        0.111317             0.168191       0.378283
```

```
1            0.069873        0.004358             0.014533       0.321717
2            0.196522        0.159888             0.246074       0.215657
3            0.168395        0.000000             0.000000       0.467172
4            0.128918        0.089246             0.160984       0.230303
..                …              …                    …             …
419          0.626403        0.743674             0.732604       0.550000
420          0.296055        0.571462             0.690358       0.336364
421          0.257714        0.337395             0.486630       0.349495
422          0.254340        0.216753             0.263519       0.267677
423          0.790197        0.823336             0.755467       0.675253

     fractal_dimension_mean  …  radius_worst  texture_worst  \
0                  0.152064  …      0.256848       0.527719
1                  0.180918  …      0.198150       0.294776
2                  0.158382  …      0.287442       0.438699
3                  0.442713  …      0.079687       0.287313
4                  0.231466  …      0.180719       0.249733
..                      …  …             …              …
419                0.396588  …      0.581999       0.463486
420                0.132056  …      0.623266       0.383262
421                0.113100  …      0.560655       0.699094
422                0.137321  …      0.393099       0.589019
423                0.425442  …      0.633582       0.730277

     perimeter_worst  area_worst  smoothness_worst  compactness_worst  \
0           0.241994    0.126229          0.297365           0.139525
1           0.175059    0.093123          0.215479           0.037789
2           0.266398    0.147070          0.333025           0.108188
3           0.067732    0.032393          0.494156           0.100620
4           0.169381    0.082653          0.403685           0.074424
..               …           …                …                …
419         0.640918    0.401543          0.459156           0.379651
420         0.576174    0.452664          0.461137           0.178527
421         0.520892    0.379915          0.300007           0.159997
422         0.379949    0.230731          0.282177           0.273705
423         0.668310    0.402035          0.619626           0.815758

     concavity_worst  concave points_worst  symmetry_worst  \
0           0.182268              0.440550        0.257441
1           0.004456              0.030144        0.185295
2           0.135783              0.349485        0.158486
3           0.000000              0.000000        0.173467
4           0.121486              0.377663        0.198502
..               …                   …               …
419         0.527077              0.873540        0.268874
420         0.328035              0.761512        0.097575
421         0.256789              0.559450        0.198502
```

```
422          0.271805               0.487285        0.128721
423          0.749760               0.910653        0.497142

     fractal_dimension_worst
0                   0.092680
1                   0.060803
2                   0.071822
3                   0.220451
4                   0.104486
..                       …
419                 0.286567
420                 0.105667
421                 0.074315
422                 0.151909
423                 0.452315

[424 rows x 30 columns]
```

[86]:
```
y_resampled
```

[86]:
```
0      0
1      0
2      0
3      0
4      0
      ..
419    1
420    1
421    1
422    1
423    1
Name: diagnosis, Length: 424, dtype: int64
```

1.1 Feature selection

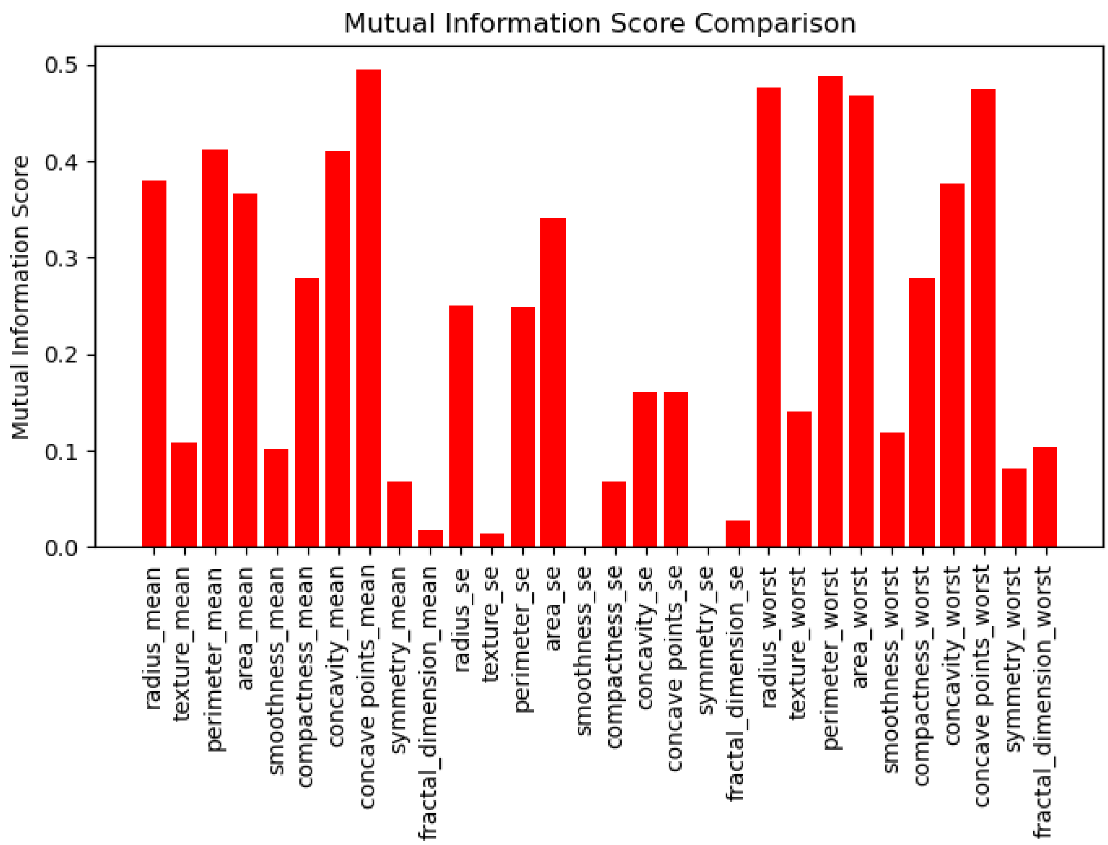
1.1.1 Information gain

[24]:
```
MI_score = mutual_info_classif(X_resampled,  y_resampled , random_state=0)
for feature in zip(feature_columns, MI_score):
    if feature[1]>0.30:
        print(feature)
```

```
('radius_mean', 0.3790613211089362)
('perimeter_mean', 0.4120872655607555)
('area_mean', 0.3662171010510915)
('concavity_mean', 0.410597450084802)
('concave points_mean', 0.49467688911271535)
('area_se', 0.3414216405120378)
```

```
('radius_worst', 0.4764124126823539)
('perimeter_worst', 0.4878617892664423)
('area_worst', 0.4667577077808542 4)
('concavity_worst', 0.3762803586441521)
('concave points_worst', 0.47395110395 07369)
```

[25]:
```
plt.figure(figsize=(8,4))
plt.bar(x=feature_columns, height=MI_score, color='red')
plt.xticks(rotation='vertical')
plt.ylabel('Mutual Information Score')
plt.title('Mutual Information Score Comparison')
plt.show()
```



[26]:
```
selected_features = [feature_columns[i] for i in range(len(feature_columns)) if␣
↪MI_score[i] > 0.3 ]
X_selected = X_resampled[selected_features]
```

[27]:
```
selected_features
```

[27]:
```
['radius_mean',
 'perimeter_mean',
 'area_mean',
 'concavity_mean',
 'concave points_mean',
 'area_se',
 'radius_worst',
 'perimeter_worst',
 'area_worst',
 'concavity_worst',
 'concave points_worst']
```

[28]:
```
X_selected
```

[28]:
```
     radius_mean  perimeter_mean  area_mean  concavity_mean  \
0       0.308060        0.297975   0.177094        0.111317
1       0.264991        0.249050   0.146554        0.004358
2       0.373373        0.361620   0.227953        0.159888
3       0.082967        0.079331   0.038515        0.000000
4       0.223816        0.213461   0.117413        0.089246
..           …              …          …              …
419     0.659709        0.685578   0.510498        0.743674
420     0.690000        0.678668   0.566490        0.571462
421     0.622320        0.604036   0.474019        0.337395
422     0.455251        0.445788   0.303118        0.216753
423     0.644564        0.665538   0.475716        0.823336

     concave points_mean   area_se  radius_worst  perimeter_worst  area_worst  \
0               0.168191  0.025024      0.256848         0.241994    0.126229
1               0.014533  0.029227      0.198150         0.175059    0.093123
2               0.246074  0.028088      0.287442         0.266398    0.147070
3               0.000000  0.041181      0.079687         0.067732    0.032393
4               0.160984  0.020654      0.180719         0.169381    0.082653
..                   …        …            …                …          …
419             0.732604  0.209186      0.581999         0.640918    0.401543
420             0.690358  0.283710      0.623266         0.576174    0.452664
421             0.486630  0.172279      0.560655         0.520892    0.379915
422             0.263519  0.077976      0.393099         0.379949    0.230731
423             0.755467  0.148335      0.633582         0.668310    0.402035

     concavity_worst  concave points_worst
0           0.182268              0.440550
1           0.004456              0.030144
2           0.135783              0.349485
3           0.000000              0.000000
4           0.121486              0.377663
..               …                    …
```

```
419          0.527077                0.873540
420          0.328035                0.761512
421          0.256789                0.559450
422          0.271805                0.487285
423          0.749760                0.910653

[424 rows x 11 columns]
```

1.1.2 KNN using information gain

[29]:
```
X_train, X_test, y_train, y_test = train_test_split(X_selected, y_resampled,␣
↪test_size=0.2, random_state=0)
```

[114]:
```
def knn(X_train,X_test,y_train,y_test):
    knn = KNeighborsClassifier(n_neighbors=5)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    accuracy = metrics.accuracy_score(y_test, y_pred)
    precision_score = metrics.precision_score(y_test, y_pred)
    recall_score = metrics.recall_score(y_test, y_pred)
    f1_score = metrics.f1_score(y_test,y_pred)
    print("Accuracy du KNN : " , accuracy)
    print("precision score du KNN : ", precision_score )
    print("recall score du KNN : ", recall_score )
    print("f1 score du KNN : ",f1_score)
```

[115]:
```
knn(X_train, X_test, y_train, y_test)
```

```
Accuracy du KNN :  0.972027972027972
precision score du KNN :  0.9841269841269841
recall score du KNN :  0.9538461538461539
f1 score du KNN :  0.96875
```

1.1.3 KNN avec grid search

[32]:
```
#creat a new KNN model
def knn_grid_search(X_selected, y_resampled):
    Knn2 = KNeighborsClassifier()
    grid_param={'n_neighbors': range(1,31),
 'weights' : ['uniform', 'distance'],
 'metric' : ['euclidean', 'manhattan', 'minkowski']}
    grid = GridSearchCV(Knn2, grid_param, cv = 10, scoring = 'accuracy')
    grid.fit(X_selected,y_resampled)

    grid1 = GridSearchCV(Knn2, grid_param, cv = 10, scoring = 'precision')
    grid1.fit(X_selected,y_resampled)

    grid2 = GridSearchCV(Knn2, grid_param, cv = 10, scoring = 'recall')
```

```
    grid2.fit(X_selected,y_resampled)

    grid3 = GridSearchCV(Knn2, grid_param, cv = 10, scoring = 'f1')
    grid3.fit(X_selected,y_resampled)

    print('grid best score accuracy',grid.best_score_)
    print('grid best score precision',grid1.best_score_)
    print('grid best score recall',grid2.best_score_)
    print('grid best score f1 score',grid3.best_score_)

    print(grid.best_params_)
    print(grid.best_estimator_)
```

[33]:
```
knn_grid_search(X_selected, y_resampled)
```

```
grid best score accuracy 0.9529346622369876
grid best score precision 0.9802130325814536
grid best score recall 0.9482683982683981
grid best score f1 score 0.9525095824954022
{'metric': 'manhattan', 'n_neighbors': 21, 'weights': 'distance'}
KNeighborsClassifier(metric='manhattan', n_neighbors=21, weights='distance')
```

1.1.4 SVM based on information gain

[34]:
```
def svm(X_train, X_test, y_train, y_test):
    svm = SVC()
    svm.fit(X_train, y_train)
    y_pred = svm.predict(X_test)
    print(classification_report(y_test, y_pred))
```

[35]:
```
svm(X_train, X_test, y_train, y_test)
```

```
              precision    recall  f1-score   support

           0       0.96      0.94      0.95        48
           1       0.92      0.95      0.93        37

    accuracy                           0.94        85
   macro avg       0.94      0.94      0.94        85
weighted avg       0.94      0.94      0.94        85
```

1.1.5 SVM with grid search

[36]:
```
def svm_grid_search(X_train,X_test,y_train ,y_test):
    param_grid = {'C': [0.1, 1, 10], 'kernel': ['linear', 'rbf', 'poly'],␣
 ↪'gamma': ['scale', 'auto']}
    svm = GridSearchCV(SVC(), param_grid=param_grid, cv=5, n_jobs=-1)
```

```
    svm.fit(X_train, y_train)
    best_params = svm.best_params_
    best_accuracy = svm.best_score_
    y_pred = svm.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print(classification_report(y_test, y_pred))
    print("Accuracy : ",accuracy)
```

[37]:
```
svm_grid_search(X_train,X_test,y_train ,y_test)
```

```
              precision    recall  f1-score   support

           0       0.94      1.00      0.97        48
           1       1.00      0.92      0.96        37

    accuracy                           0.96        85
   macro avg       0.97      0.96      0.96        85
weighted avg       0.97      0.96      0.96        85

Accuracy :  0.9647058823529412
```

1.1.6 Decision tree

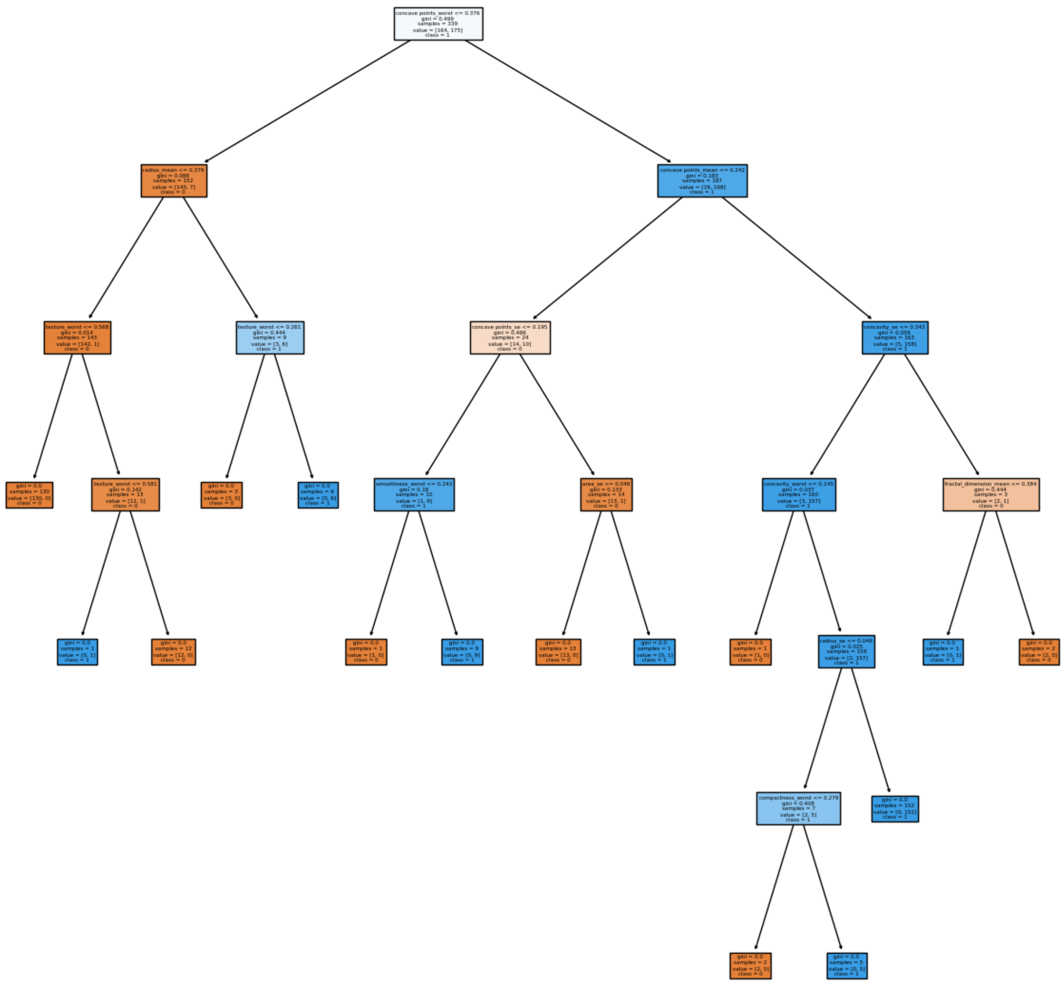
[125]:
```
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled,
 ↪test_size=0.2, random_state=0)
```

[126]:
```
clf = tree.DecisionTreeClassifier()
clf.fit(X_train, y_train)
y_pred= clf.predict(X_test)
acc2 = metrics.accuracy_score(y_test, y_pred)
precision =  metrics.precision_score(y_test, y_pred)
recall = metrics.recall_score(y_test, y_pred)
f1_score = metrics.f1_score(y_test , y_pred)
print("Accuracy:", acc2)
print("Precision: ",precision)
print("Recall: ",recall)
print("F1 score : ",f1_score)
```

```
Accuracy: 0.9529411764705882
Precision:  0.9459459459459459
Recall:  0.9459459459459459
F1 score :  0.9459459459459459
```

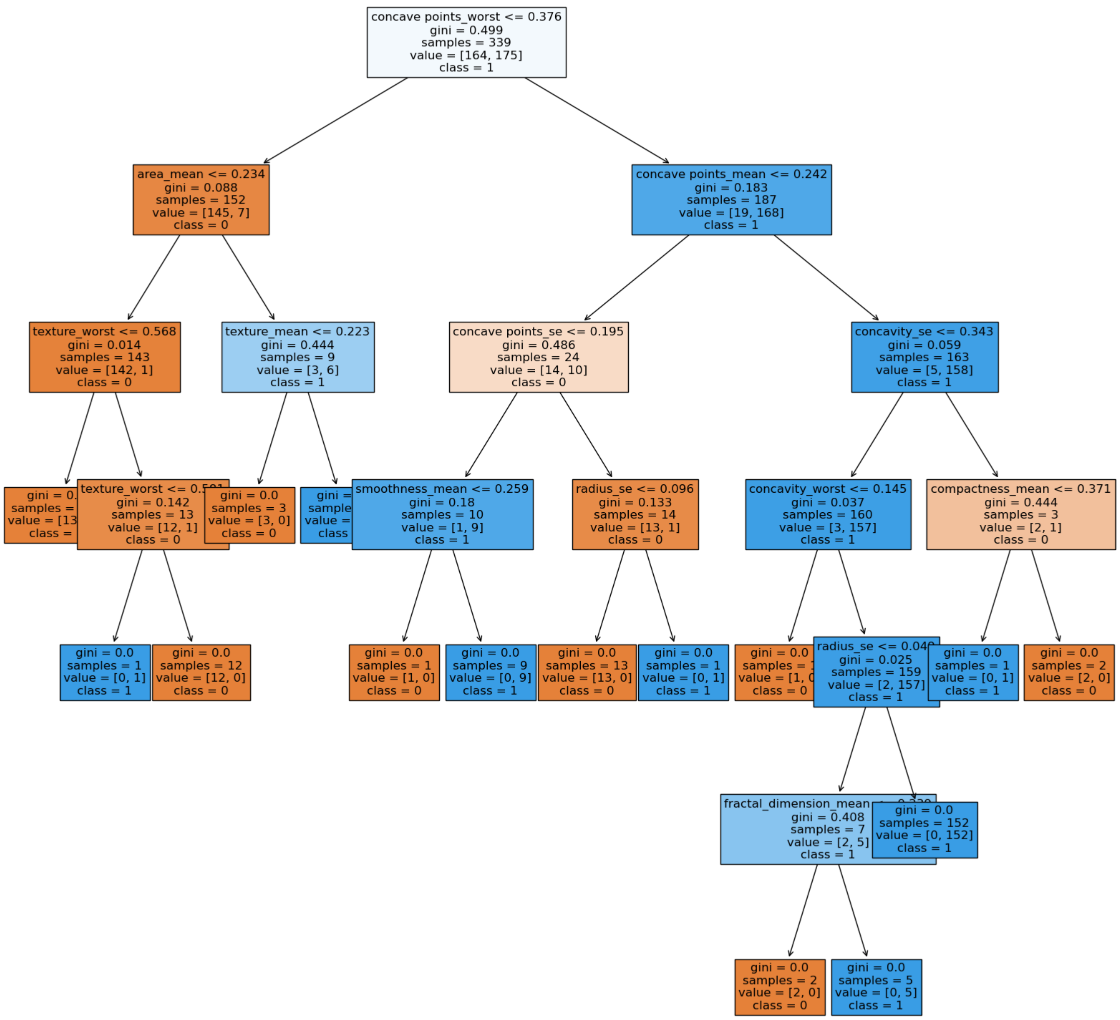
[127]:
```
from sklearn import tree
clf = tree.DecisionTreeClassifier()
clf.fit(X_train, y_train)
fig = plt.figure(figsize=(15,15))
_ = tree.plot_tree(clf,
```

```
feature_names=feature_columns
                            ,
class_names=["0","1"],
filled=True)
```



```
[128]: from sklearn import tree
import matplotlib.pyplot as plt

clf = tree.DecisionTreeClassifier()
clf.fit(X_train, y_train)

fig, ax = plt.subplots(figsize=(20,20))
_ = tree.plot_tree(clf,
```

```
                        feature_names=feature_columns,
                        class_names=["0","1"],
                        filled=True,
                        fontsize=12)

plt.show()
```



[42]:

```
#Grid search
DT = tree.DecisionTreeClassifier()
params = {
 'criterion': ['gini', 'entropy'],
 'max_depth': [2, 4, 6, 8, 10],
 'min_samples_split': [2, 4, 6, 8, 10],
```

```
 'min_samples_leaf': [1, 2, 3, 4, 5]
    }
grid = GridSearchCV(DT, params, cv = 10, scoring = 'accuracy')
grid.fit(X_resampled,y_resampled)

grid1 = GridSearchCV(DT, params, cv = 10, scoring = 'precision')
grid1.fit(X_resampled,y_resampled)

grid2 = GridSearchCV(DT, params, cv = 10, scoring = 'recall')
grid2.fit(X_resampled,y_resampled)

grid3 = GridSearchCV(DT, params, cv = 10, scoring = 'f1')
grid3.fit(X_resampled,y_resampled)

print("Accuracy",grid.best_score_)
print("Precision",grid1.best_score_)
print("Recall",grid2.best_score_)
print("f1 score",grid3.best_score_)

print(grid.best_params_)
print(grid.best_estimator_)
```

```
Accuracy 0.9553156146179402
Precision 0.9654761904761905
Recall 0.958008658008658
f1 score 0.9526802258055802
{'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 3,
'min_samples_split': 2}
DecisionTreeClassifier(criterion='entropy', max_depth=4, min_samples_leaf=3)
```

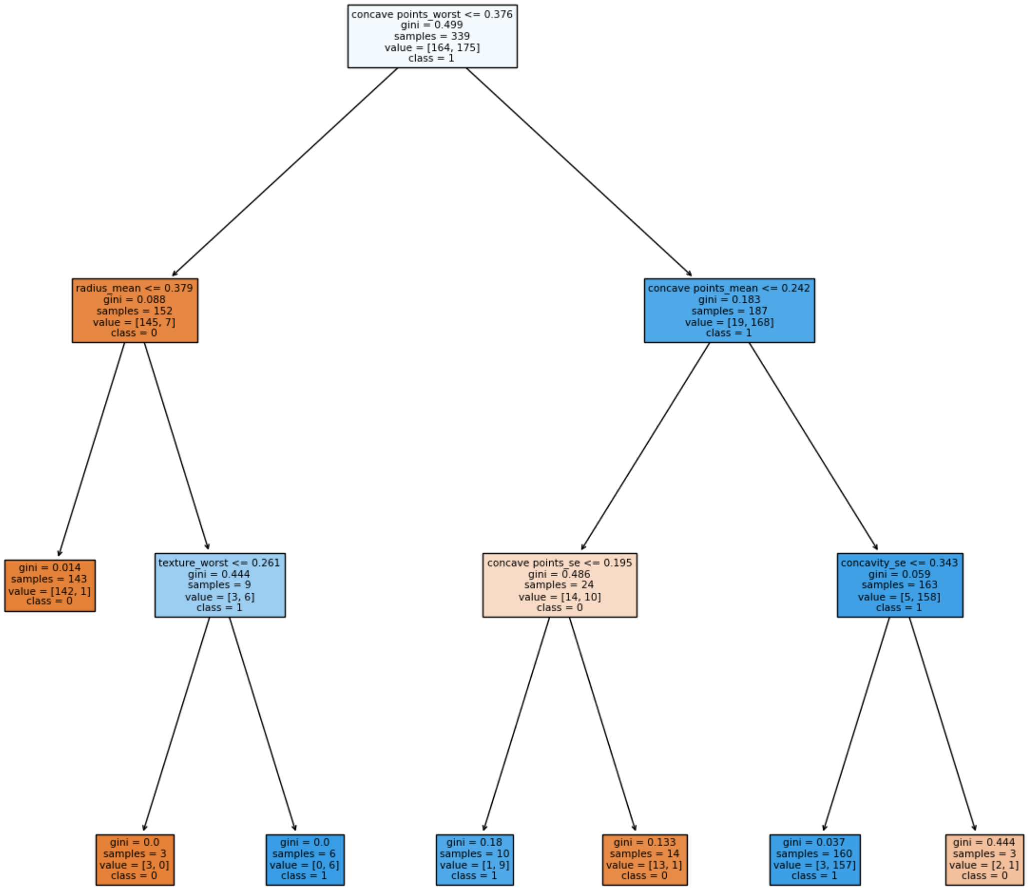
[129]:
```
# Use a pruning algorithm to prune the decision tree
clf = tree.DecisionTreeClassifier()
path = clf.cost_complexity_pruning_path(X_train, y_train)
ccp_alphas = path.ccp_alphas[:-1]
clfs = []
for ccp_alpha in ccp_alphas:
    clf = tree.DecisionTreeClassifier(ccp_alpha=ccp_alpha)
    clf.fit(X_train, y_train)
    clfs.append(clf)
# Evaluate the pruned decision tree using the testing data
acc_scores = []
for clf in clfs:
    y_pred = clf.predict(X_test)
    acc_score = accuracy_score(y_test, y_pred)
    prec_score = precision_score(y_test , y_pred)
    acc_scores.append(acc_score)
# Find the best pruning parameter based on accuracy score
```

```python
best_clf = clfs[acc_scores.index(max(acc_scores))]
# Evaluate the best pruned decision tree using the testing data
y_pred = best_clf.predict(X_test)
acc_score = accuracy_score(y_test, y_pred)
precision = precision_score(y_test , y_pred)
recall = recall_score(y_test , y_pred)
f1_score = metrics.f1_score(y_test , y_pred)

print("Accuracy score: {:.2f}".format(acc_score))
print("precision score: {:.2f}".format(precision))
print("recall score: {:.2f}".format(recall))
print("f1 score: {:.2f}".format(f1_score))

#print("ccp_alpha: {:.3f}".format(ccp_alphas[acc_scores.
 ↪index(max(acc_scores))]))
```

```
Accuracy score: 0.96
precision score: 0.95
recall score: 0.97
f1 score: 0.96
```

[130]:

```python
from sklearn import tree
fig = plt.figure(figsize=(15,15))
_ = tree.plot_tree(best_clf,
 feature_names=feature_columns,
 class_names=["0","1"],filled=True)
```



1.2 Correlation

[45]:
```
X_resampled
```

[45]:
```
     radius_mean  texture_mean  perimeter_mean  area_mean  smoothness_mean  \
0       0.308060      0.425769        0.297975   0.177094         0.314977
1       0.264991      0.293879        0.249050   0.146554         0.282567
2       0.373373      0.355090        0.361620   0.227953         0.390358
3       0.082967      0.241123        0.079331   0.038515         0.462851
4       0.223816      0.252959        0.213461   0.117413         0.407240
..           …             …              …          …               …
419     0.659709      0.520122        0.685578   0.510498         0.517017
420     0.690000      0.428813        0.678668   0.566490         0.526948
```

```
421     0.622320      0.626987         0.604036   0.474019         0.407782
422     0.455251      0.621238         0.445788   0.303118         0.288165
423     0.644564      0.663510         0.665538   0.475716         0.588336

     compactness_mean  concavity_mean  concave points_mean  symmetry_mean  \
0            0.176676        0.111317             0.168191       0.378283
1            0.069873        0.004358             0.014533       0.321717
2            0.196522        0.159888             0.246074       0.215657
3            0.168395        0.000000             0.000000       0.467172
4            0.128918        0.089246             0.160984       0.230303
..                …               …                    …              …
419          0.626403        0.743674             0.732604       0.550000
420          0.296055        0.571462             0.690358       0.336364
421          0.257714        0.337395             0.486630       0.349495
422          0.254340        0.216753             0.263519       0.267677
423          0.790197        0.823336             0.755467       0.675253

     fractal_dimension_mean  …  radius_worst  texture_worst  \
0                  0.152064  …      0.256848       0.527719
1                  0.180918  …      0.198150       0.294776
2                  0.158382  …      0.287442       0.438699
3                  0.442713  …      0.079687       0.287313
4                  0.231466  …      0.180719       0.249733
..                      …  …           …              …
419                0.396588  …      0.581999       0.463486
420                0.132056  …      0.623266       0.383262
421                0.113100  …      0.560655       0.699094
422                0.137321  …      0.393099       0.589019
423                0.425442  …      0.633582       0.730277

     perimeter_worst  area_worst  smoothness_worst  compactness_worst  \
0           0.241994    0.126229          0.297365           0.139525
1           0.175059    0.093123          0.215479           0.037789
2           0.266398    0.147070          0.333025           0.108188
3           0.067732    0.032393          0.494156           0.100620
4           0.169381    0.082653          0.403685           0.074424
..               …           …                 …                  …
419         0.640918    0.401543          0.459156           0.379651
420         0.576174    0.452664          0.461137           0.178527
421         0.520892    0.379915          0.300007           0.159997
422         0.379949    0.230731          0.282177           0.273705
423         0.668310    0.402035          0.619626           0.815758

     concavity_worst  concave points_worst  symmetry_worst  \
0           0.182268              0.440550        0.257441
1           0.004456              0.030144        0.185295
2           0.135783              0.349485        0.158486
```

```
3           0.000000             0.000000        0.173467
4           0.121486             0.377663        0.198502
..               …                 …              …
419         0.527077             0.873540        0.268874
420         0.328035             0.761512        0.097575
421         0.256789             0.559450        0.198502
422         0.271805             0.487285        0.128721
423         0.749760             0.910653        0.497142

     fractal_dimension_worst
0                   0.092680
1                   0.060803
2                   0.071822
3                   0.220451
4                   0.104486
..                       …
419                 0.286567
420                 0.105667
421                 0.074315
422                 0.151909
423                 0.452315

[424 rows x 30 columns]
```

[46]:
```
y_resampled
```

[46]:
```
0      0
1      0
2      0
3      0
4      0
      ..
419    1
420    1
421    1
422    1
423    1
Name: diagnosis, Length: 424, dtype: int64
```

[47]:
```
df_corr = pd.DataFrame(X_resampled , columns = X_resampled.columns)
```

[48]:
```
df_corr['diagnosis'] = y_resampled
```

[49]:
```
df_corr
```

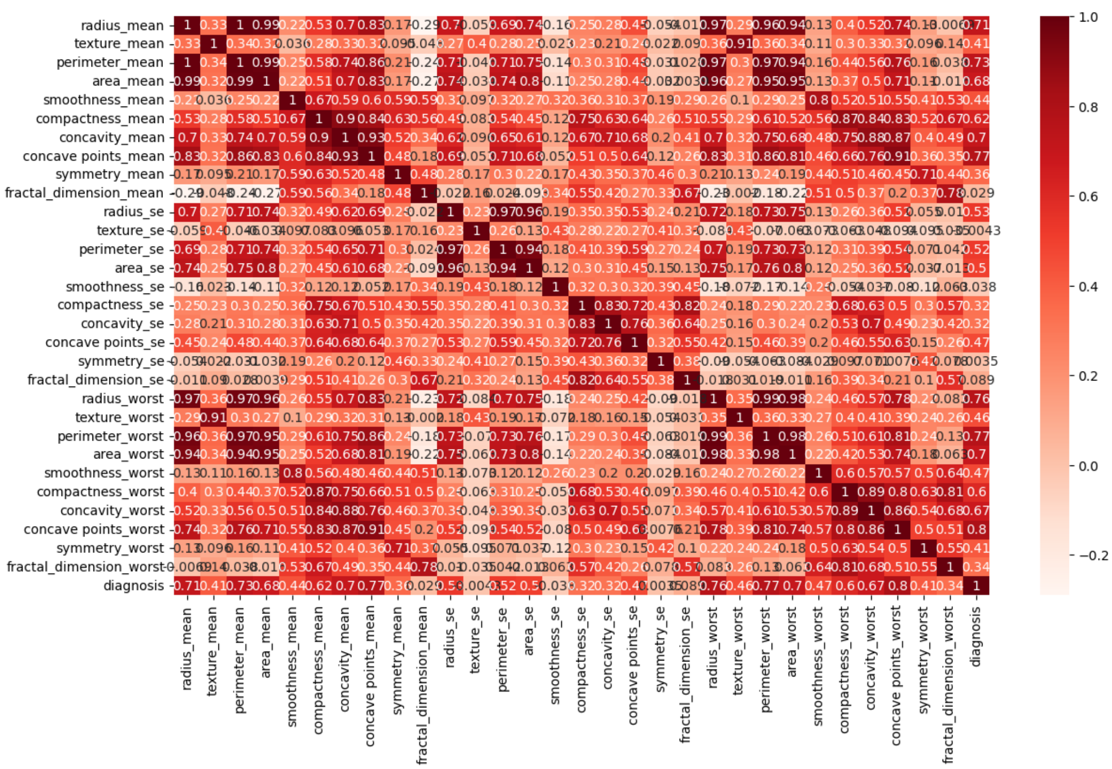
[49]:
```
     radius_mean  texture_mean  perimeter_mean  area_mean  smoothness_mean  \
0       0.308060      0.425769        0.297975   0.177094         0.314977
```

```
1       0.264991      0.293879        0.249050   0.146554          0.282567
2       0.373373      0.355090        0.361620   0.227953          0.390358
3       0.082967      0.241123        0.079331   0.038515          0.462851
4       0.223816      0.252959        0.213461   0.117413          0.407240
..           …             …             …          …               …
419     0.659709      0.520122        0.685578   0.510498          0.517017
420     0.690000      0.428813        0.678668   0.566490          0.526948
421     0.622320      0.626987        0.604036   0.474019          0.407782
422     0.455251      0.621238        0.445788   0.303118          0.288165
423     0.644564      0.663510        0.665538   0.475716          0.588336

     compactness_mean  concavity_mean  concave points_mean  symmetry_mean  \
0            0.176676        0.111317             0.168191       0.378283
1            0.069873        0.004358             0.014533       0.321717
2            0.196522        0.159888             0.246074       0.215657
3            0.168395        0.000000             0.000000       0.467172
4            0.128918        0.089246             0.160984       0.230303
..                …               …                    …              …
419          0.626403        0.743674             0.732604       0.550000
420          0.296055        0.571462             0.690358       0.336364
421          0.257714        0.337395             0.486630       0.349495
422          0.254340        0.216753             0.263519       0.267677
423          0.790197        0.823336             0.755467       0.675253

     fractal_dimension_mean  …  texture_worst  perimeter_worst  area_worst  \
0                  0.152064  …       0.527719         0.241994    0.126229
1                  0.180918  …       0.294776         0.175059    0.093123
2                  0.158382  …       0.438699         0.266398    0.147070
3                  0.442713  …       0.287313         0.067732    0.032393
4                  0.231466  …       0.249733         0.169381    0.082653
..                      …  …            …                …           …
419                0.396588  …       0.463486         0.640918    0.401543
420                0.132056  …       0.383262         0.576174    0.452664
421                0.113100  …       0.699094         0.520892    0.379915
422                0.137321  …       0.589019         0.379949    0.230731
423                0.425442  …       0.730277         0.668310    0.402035

     smoothness_worst  compactness_worst  concavity_worst  \
0            0.297365           0.139525         0.182268
1            0.215479           0.037789         0.004456
2            0.333025           0.108188         0.135783
3            0.494156           0.100620         0.000000
4            0.403685           0.074424         0.121486
..                …                  …                …
419          0.459156           0.379651         0.527077
420          0.461137           0.178527         0.328035
421          0.300007           0.159997         0.256789
```

```
422          0.282177          0.273705          0.271805
423          0.619626          0.815758          0.749760

     concave points_worst  symmetry_worst  fractal_dimension_worst  diagnosis
0                0.440550        0.257441                 0.092680          0
1                0.030144        0.185295                 0.060803          0
2                0.349485        0.158486                 0.071822          0
3                0.000000        0.173467                 0.220451          0
4                0.377663        0.198502                 0.104486          0
..                    …               …                        …          …
419              0.873540        0.268874                 0.286567          1
420              0.761512        0.097575                 0.105667          1
421              0.559450        0.198502                 0.074315          1
422              0.487285        0.128721                 0.151909          1
423              0.910653        0.497142                 0.452315          1

[424 rows x 31 columns]
```

[50]:

```python
# Matrice de correlation
#Using Pearson Correlation
plt.figure(figsize=(14,8))
cor = df_corr.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.Reds)
plt.show()
```



[51]:
```
#Correlation with output variable
cor_target = abs(cor["diagnosis"])
#Selecting highly correlated features
relevant_features = list(cor_target[abs(cor_target) > 0.5].index)
relevant_features.remove('diagnosis')
relevant_features
# we choose the variables that have a correlation higher to 0.5 with our target␣
↪variable
```

[51]:
```
['radius_mean',
 'perimeter_mean',
 'area_mean',
 'compactness_mean',
 'concavity_mean',
 'concave points_mean',
 'radius_se',
 'perimeter_se',
 'area_se',
 'radius_worst',
 'perimeter_worst',
 'area_worst',
 'compactness_worst',
 'concavity_worst',
 'concave points_worst']
```

[52]:
```
X_selected_corr = X_resampled[relevant_features]
```

[53]:
```
X_selected_corr
```

[53]:
```
     radius_mean  perimeter_mean  area_mean  compactness_mean  concavity_mean  \
0       0.308060        0.297975   0.177094          0.176676        0.111317
1       0.264991        0.249050   0.146554          0.069873        0.004358
2       0.373373        0.361620   0.227953          0.196522        0.159888
3       0.082967        0.079331   0.038515          0.168395        0.000000
4       0.223816        0.213461   0.117413          0.128918        0.089246
..           …             …          …                …               …
419     0.659709        0.685578   0.510498          0.626403        0.743674
420     0.690000        0.678668   0.566490          0.296055        0.571462
421     0.622320        0.604036   0.474019          0.257714        0.337395
422     0.455251        0.445788   0.303118          0.254340        0.216753
423     0.644564        0.665538   0.475716          0.790197        0.823336

     concave points_mean  radius_se  perimeter_se   area_se  radius_worst  \
0               0.168191   0.044288      0.046082  0.025024      0.256848
1               0.014533   0.058084      0.045422  0.029227      0.198150
```

```
2              0.246074  0.043744      0.039533  0.028088      0.287442
3              0.000000  0.146804      0.113556  0.041181      0.079687
4              0.160984  0.048380      0.046412  0.020654      0.180719
..                  …         …             …         …            …
419            0.732604  0.308057      0.376997  0.209186      0.581999
420            0.690358  0.385479      0.325873  0.283710      0.623266
421            0.486630  0.236828      0.209490  0.172279      0.560655
422            0.263519  0.124896      0.125713  0.077976      0.393099
423            0.755467  0.222524      0.236300  0.148335      0.633582

     perimeter_worst  area_worst  compactness_worst  concavity_worst  \
0           0.241994    0.126229           0.139525         0.182268
1           0.175059    0.093123           0.037789         0.004456
2           0.266398    0.147070           0.108188         0.135783
3           0.067732    0.032393           0.100620         0.000000
4           0.169381    0.082653           0.074424         0.121486
..               …           …                 …                …
419         0.640918    0.401543           0.379651         0.527077
420         0.576174    0.452664           0.178527         0.328035
421         0.520892    0.379915           0.159997         0.256789
422         0.379949    0.230731           0.273705         0.271805
423         0.668310    0.402035           0.815758         0.749760

     concave points_worst
0                0.440550
1                0.030144
2                0.349485
3                0.000000
4                0.377663
..                    …
419              0.873540
420              0.761512
421              0.559450
422              0.487285
423              0.910653

[424 rows x 15 columns]
```

[54]:
```python
# splitting the data
X_train, X_test, y_train, y_test = train_test_split(X_selected_corr,
 ↪y_resampled, test_size=0.2, random_state=0)
```

1.2.1 KNN based on correlation

[55]:
```python
knn(X_train, X_test, y_train, y_test)
```

```
Accuracy du KNN :  0.9529411764705882
precision score du KNN :  0.9459459459459459
```

```
recall score du KNN :  0.9459459459459459
f1 score du KNN :  0.9459459459459459
```

1.2.2 KNN with grid search

[56]:
```
knn_grid_search(X_selected_corr, y_resampled)
```

```
grid best score accuracy 0.9552602436323365
grid best score precision 0.9765424430641823
grid best score recall 0.938961038961039
grid best score f1 score 0.9543019563155696
{'metric': 'manhattan', 'n_neighbors': 16, 'weights': 'distance'}
KNeighborsClassifier(metric='manhattan', n_neighbors=16, weights='distance')
```

1.2.3 SVM based on correlation

[57]:
```
svm(X_train, X_test, y_train, y_test)
```

```
              precision    recall  f1-score   support

           0       0.96      0.96      0.96        48
           1       0.95      0.95      0.95        37

    accuracy                           0.95        85
   macro avg       0.95      0.95      0.95        85
weighted avg       0.95      0.95      0.95        85
```

1.2.4 SVM with grid search

[58]:
```
svm_grid_search(X_train,X_test,y_train ,y_test)
```

```
              precision    recall  f1-score   support

           0       0.98      0.98      0.98        48
           1       0.97      0.97      0.97        37

    accuracy                           0.98        85
   macro avg       0.98      0.98      0.98        85
weighted avg       0.98      0.98      0.98        85

Accuracy :  0.9764705882352941
```

2 Select Kbest

[59]:
```
selector = SelectKBest(chi2, k=10)
X_new = selector.fit_transform(X_resampled, y_resampled)
mask = selector.get_support()
```

```
selected_features = X_resampled.columns[mask]
selected = selected_features.values
selected
```

[59]:
```
array(['perimeter_mean', 'area_mean', 'concavity_mean',
       'concave points_mean', 'radius_worst', 'perimeter_worst',
       'area_worst', 'concavity_worst', 'concave points_worst',
       'diagnosis'], dtype=object)
```

[60]:
```
X_selected_kbest = X_resampled[selected]
```

[61]:
```
X_selected_kbest.drop('diagnosis',axis = 1, inplace = True)
```

[62]:
```
X_selected_kbest
```

[62]:
```
     perimeter_mean  area_mean  concavity_mean  concave points_mean  \
0          0.297975   0.177094        0.111317             0.168191
1          0.249050   0.146554        0.004358             0.014533
2          0.361620   0.227953        0.159888             0.246074
3          0.079331   0.038515        0.000000             0.000000
4          0.213461   0.117413        0.089246             0.160984
..              …          …             …                  …
419        0.685578   0.510498        0.743674             0.732604
420        0.678668   0.566490        0.571462             0.690358
421        0.604036   0.474019        0.337395             0.486630
422        0.445788   0.303118        0.216753             0.263519
423        0.665538   0.475716        0.823336             0.755467

     radius_worst  perimeter_worst  area_worst  concavity_worst  \
0        0.256848         0.241994    0.126229         0.182268
1        0.198150         0.175059    0.093123         0.004456
2        0.287442         0.266398    0.147070         0.135783
3        0.079687         0.067732    0.032393         0.000000
4        0.180719         0.169381    0.082653         0.121486
..            …              …           …              …
419      0.581999         0.640918    0.401543         0.527077
420      0.623266         0.576174    0.452664         0.328035
421      0.560655         0.520892    0.379915         0.256789
422      0.393099         0.379949    0.230731         0.271805
423      0.633582         0.668310    0.402035         0.749760

     concave points_worst
0                0.440550
1                0.030144
2                0.349485
3                0.000000
4                0.377663
```

```
..                   …
419             0.873540
420             0.761512
421             0.559450
422             0.487285
423             0.910653

[424 rows x 9 columns]
```

[63]:
```
# splitting the data
X_train, X_test, y_train, y_test = train_test_split(X_selected_kbest,␣
 ↪y_resampled, test_size=0.2, random_state=0)
```

2.0.1 KNN based on selectKbest

[64]:
```
knn(X_train, X_test, y_train, y_test)
```

```
Accuracy du KNN :  0.9411764705882353
precision score du KNN :  0.9210526315789473
recall score du KNN :  0.9459459459459459
f1 score du KNN :  0.9333333333333332
```

2.0.2 KNN with grid search

[65]:
```
knn_grid_search(X_selected_kbest, y_resampled)
```

```
grid best score accuracy 0.95531561461794
grid best score precision 0.9802130325814536
grid best score recall 0.9528138528138529
grid best score f1 score 0.9546179401993357
{'metric': 'manhattan', 'n_neighbors': 29, 'weights': 'distance'}
KNeighborsClassifier(metric='manhattan', n_neighbors=29, weights='distance')
```

2.0.3 SVM

[66]:
```
svm(X_train, X_test, y_train, y_test)
```

```
              precision    recall  f1-score   support

           0       0.96      0.94      0.95        48
           1       0.92      0.95      0.93        37

    accuracy                           0.94        85
   macro avg       0.94      0.94      0.94        85
weighted avg       0.94      0.94      0.94        85
```

2.0.4 SVM with grid search

[67]:
```
svm_grid_search(X_train,X_test,y_train ,y_test)
```

```
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        48
           1       1.00      0.95      0.97        37

    accuracy                           0.98        85
   macro avg       0.98      0.97      0.98        85
weighted avg       0.98      0.98      0.98        85

Accuracy :  0.9764705882352941
```

2.1 Wrappers

[68]:
```
import joblib
import sys
sys.modules['sklearn.externals.joblib'] = joblib
from mlxtend.feature_selection import SequentialFeatureSelector as SFS
from mlxtend.feature_selection import ExhaustiveFeatureSelector as EFS
```

[69]:
```
X_train, X_test, y_train, y_test = train_test_split( X_resampled, y_resampled,␣
↪test_size=0.33, random_state=42)
# create the classifier with n_estimators = 100

clf = RandomForestClassifier(n_estimators=100, random_state=0)



# fit the model to the training set

clf.fit(X_train, y_train)

# view the feature scores

feature_scores = pd.Series(clf.feature_importances_, index=X_train.columns).
↪sort_values(ascending=False)

selected = feature_scores[feature_scores.values >0.01]
selected_features = selected.index.values
```

[70]:
```
selected_features
```

[70]:
```
array(['diagnosis', 'concave points_mean', 'radius_worst',
       'concave points_worst', 'area_worst', 'perimeter_worst',
       'concavity_mean', 'radius_mean', 'perimeter_mean',
```

```
       'concavity_worst', 'area_se', 'compactness_mean'], dtype=object)
```

[71]:
```
X_selected_rf = X_resampled[selected_features]
```

[72]:
```
X_selected_rf.drop('diagnosis',axis =1,inplace = True)
```

[73]:
```
X_selected_rf
```

[73]:
```
     concave points_mean  radius_worst  concave points_worst  area_worst  \
0               0.168191      0.256848              0.440550    0.126229
1               0.014533      0.198150              0.030144    0.093123
2               0.246074      0.287442              0.349485    0.147070
3               0.000000      0.079687              0.000000    0.032393
4               0.160984      0.180719              0.377663    0.082653
..                   …             …                     …           …
419             0.732604      0.581999              0.873540    0.401543
420             0.690358      0.623266              0.761512    0.452664
421             0.486630      0.560655              0.559450    0.379915
422             0.263519      0.393099              0.487285    0.230731
423             0.755467      0.633582              0.910653    0.402035

     perimeter_worst  concavity_mean  radius_mean  perimeter_mean  \
0           0.241994        0.111317     0.308060        0.297975
1           0.175059        0.004358     0.264991        0.249050
2           0.266398        0.159888     0.373373        0.361620
3           0.067732        0.000000     0.082967        0.079331
4           0.169381        0.089246     0.223816        0.213461
..               …               …            …               …
419         0.640918        0.743674     0.659709        0.685578
420         0.576174        0.571462     0.690000        0.678668
421         0.520892        0.337395     0.622320        0.604036
422         0.379949        0.216753     0.455251        0.445788
423         0.668310        0.823336     0.644564        0.665538

     concavity_worst   area_se  compactness_mean
0           0.182268  0.025024          0.176676
1           0.004456  0.029227          0.069873
2           0.135783  0.028088          0.196522
3           0.000000  0.041181          0.168395
4           0.121486  0.020654          0.128918
..               …         …                 …
419         0.527077  0.209186          0.626403
420         0.328035  0.283710          0.296055
421         0.256789  0.172279          0.257714
422         0.271805  0.077976          0.254340
423         0.749760  0.148335          0.790197
```

```
[424 rows x 11 columns]
```

[74]:
```
# splitting the data
X_train, X_test, y_train, y_test = train_test_split(X_selected_rf, y_resampled,
 test_size=0.2, random_state=0)
```

2.1.1 KNN based on Random forest wrapper

[75]:
```
knn(X_train, X_test, y_train, y_test)
```

```
Accuracy du KNN :  0.9529411764705882
precision score du KNN :  0.9459459459459459
recall score du KNN :  0.9459459459459459
f1 score du KNN :  0.9459459459459459
```

[76]:
```
knn_grid_search(X_selected_rf, y_resampled)
```

```
grid best score accuracy 0.9529346622369876
grid best score precision 0.9758840282524492
grid best score recall 0.9482683982683981
grid best score f1 score 0.9526342347386023
{'metric': 'manhattan', 'n_neighbors': 8, 'weights': 'distance'}
KNeighborsClassifier(metric='manhattan', n_neighbors=8, weights='distance')
```

2.2 SVM

[77]:
```
svm(X_train, X_test, y_train, y_test)
```

```
              precision    recall  f1-score   support

           0       0.96      0.94      0.95        48
           1       0.92      0.95      0.93        37

    accuracy                           0.94        85
   macro avg       0.94      0.94      0.94        85
weighted avg       0.94      0.94      0.94        85
```

[78]:
```
svm_grid_search(X_train,X_test,y_train ,y_test)
```

```
              precision    recall  f1-score   support

           0       0.98      0.98      0.98        48
           1       0.97      0.97      0.97        37

    accuracy                           0.98        85
   macro avg       0.98      0.98      0.98        85
weighted avg       0.98      0.98      0.98        85
```

```
Accuracy :  0.9764705882352941
```

2.3 Wrapper the Recursive Feature Elimination :

[79]:
```
from sklearn.feature_selection import RFE
from sklearn.svm import SVC

# Initialize an SVM classifier and an RFE feature selector
svm = SVC(kernel='linear')
rfe = RFE(estimator=svm, n_features_to_select=10, step=1)

# Fit the RFE selector to the data and get the selected feature indices
rfe.fit(X_resampled, y_resampled)
selected_indices = rfe.get_support(indices=True)

selected_names = [X_resampled.columns[i] for i, selected in enumerate(rfe.
 ↪support_) if selected]

# Print the selected feature names
print('Selected features:', selected_names)
```

```
Selected features: ['radius_mean', 'perimeter_mean', 'radius_se',
'concavity_se', 'concave points_se', 'radius_worst', 'texture_worst',
'area_worst', 'concavity_worst', 'diagnosis']
```

[80]:
```
X_selected_rfe = X_resampled[selected_names]
```

[81]:
```
X_selected_rfe.drop('diagnosis',axis=1,inplace=True)
```

[82]:
```
X_selected_rfe
```

[82]:

	radius_mean	perimeter_mean	radius_se	concavity_se	concave points_se	\
0	0.308060	0.297975	0.044288	0.052904	0.224285	
1	0.264991	0.249050	0.058084	0.004697	0.055389	
2	0.373373	0.361620	0.043744	0.054369	0.224095	
3	0.082967	0.079331	0.146804	0.000000	0.000000	
4	0.223816	0.213461	0.048380	0.049949	0.224474	
..	…	…	…	…	…	
419	0.659709	0.685578	0.308057	0.198106	0.497064	
420	0.690000	0.678668	0.385479	0.131263	0.464861	
421	0.622320	0.604036	0.236828	0.099747	0.317863	
422	0.455251	0.445788	0.124896	0.119444	0.294942	
423	0.644564	0.665538	0.222524	0.179722	0.315211	

	radius_worst	texture_worst	area_worst	concavity_worst
0	0.256848	0.527719	0.126229	0.182268
1	0.198150	0.294776	0.093123	0.004456

```
2        0.287442       0.438699    0.147070          0.135783
3        0.079687       0.287313    0.032393          0.000000
4        0.180719       0.249733    0.082653          0.121486
..          …              …           …               …
419      0.581999       0.463486    0.401543          0.527077
420      0.623266       0.383262    0.452664          0.328035
421      0.560655       0.699094    0.379915          0.256789
422      0.393099       0.589019    0.230731          0.271805
423      0.633582       0.730277    0.402035          0.749760

[424 rows x 9 columns]
```

[83]:
```python
# splitting the data
X_train, X_test, y_train, y_test = train_test_split(X_selected_rfe,␣
 ↪y_resampled, test_size=0.2, random_state=0)
```

2.4 KNN

[84]:
```python
knn(X_train, X_test, y_train, y_test)
```

```
Accuracy du KNN :  0.9294117647058824
precision score du KNN :  0.918918918918919
recall score du KNN :  0.918918918918919
f1 score du KNN :  0.918918918918919
```

[85]:
```python
knn_grid_search(X_selected_rfe, y_resampled)
```

```
grid best score accuracy 0.9599667774086378
grid best score precision 0.9684523809523811
grid best score recall 0.9623376623376624
grid best score f1 score 0.9598331126238102
{'metric': 'manhattan', 'n_neighbors': 14, 'weights': 'distance'}
KNeighborsClassifier(metric='manhattan', n_neighbors=14, weights='distance')
```

2.5 SVM

[86]:
```python
def svm(X_train, X_test, y_train, y_test):
    svm = SVC()
    svm.fit(X_train, y_train)
    y_pred = svm.predict(X_test)
    print(classification_report(y_test, y_pred))
```

[87]:
```python
svm(X_train, X_test, y_train, y_test)
```

```
              precision    recall  f1-score   support

           0       0.94      0.96      0.95        48
           1       0.94      0.92      0.93        37
```

```
    accuracy                           0.94        85
   macro avg       0.94      0.94      0.94        85
weighted avg       0.94      0.94      0.94        85
```

[88]:
```
svm_grid_search(X_train,X_test,y_train ,y_test)
```

```
              precision    recall  f1-score   support

           0       0.94      0.96      0.95        48
           1       0.94      0.92      0.93        37

    accuracy                           0.94        85
   macro avg       0.94      0.94      0.94        85
weighted avg       0.94      0.94      0.94        85

Accuracy :  0.9411764705882353
```

3 Unbalanced data methods Smote

[89]:
```
df['diagnosis'].value_counts()
```

[89]:
```
0    357
1    212
Name: diagnosis, dtype: int64
```

[90]:
```
df
```

[90]:
```
     radius_mean  texture_mean  perimeter_mean  area_mean  smoothness_mean  \
0       0.521037      0.022658        0.545989   0.363733         0.593753
1       0.643144      0.272574        0.615783   0.501591         0.289880
2       0.601496      0.390260        0.595743   0.449417         0.514309
3       0.210090      0.360839        0.233501   0.102906         0.811321
4       0.629893      0.156578        0.630986   0.489290         0.430351
..           …             …              …          …                …
564     0.690000      0.428813        0.678668   0.566490         0.526948
565     0.622320      0.626987        0.604036   0.474019         0.407782
566     0.455251      0.621238        0.445788   0.303118         0.288165
567     0.644564      0.663510        0.665538   0.475716         0.588336
568     0.036869      0.501522        0.028540   0.015907         0.000000

     compactness_mean  concavity_mean  concave points_mean  symmetry_mean  \
0            0.792037        0.703140             0.731113       0.686364
1            0.181768        0.203608             0.348757       0.379798
2            0.431017        0.462512             0.635686       0.509596
3            0.811361        0.565604             0.522863       0.776263
4            0.347893        0.463918             0.518390       0.378283
..                …               …                    …              …
```

```
564          0.296055        0.571462            0.690358       0.336364
565          0.257714        0.337395            0.486630       0.349495
566          0.254340        0.216753            0.263519       0.267677
567          0.790197        0.823336            0.755467       0.675253
568          0.074351        0.000000            0.000000       0.266162

     fractal_dimension_mean  …  texture_worst  perimeter_worst  area_worst  \
0                  0.605518  …       0.141525         0.668310    0.450698
1                  0.141323  …       0.303571         0.539818    0.435214
2                  0.211247  …       0.360075         0.508442    0.374508
3                  1.000000  …       0.385928         0.241347    0.094008
4                  0.186816  …       0.123934         0.506948    0.341575
..                      …   …            …               …          …
564                0.132056  …       0.383262         0.576174    0.452664
565                0.113100  …       0.699094         0.520892    0.379915
566                0.137321  …       0.589019         0.379949    0.230731
567                0.425442  …       0.730277         0.668310    0.402035
568                0.187026  …       0.489072         0.043578    0.020497

     smoothness_worst  compactness_worst  concavity_worst  \
0            0.601136           0.619292         0.568610
1            0.347553           0.154563         0.192971
2            0.483590           0.385375         0.359744
3            0.915472           0.814012         0.548642
4            0.437364           0.172415         0.319489
..                …                  …                …
564          0.461137           0.178527         0.328035
565          0.300007           0.159997         0.256789
566          0.282177           0.273705         0.271805
567          0.619626           0.815758         0.749760
568          0.124084           0.036043         0.000000

     concave points_worst  symmetry_worst  fractal_dimension_worst  diagnosis
0                0.912027        0.598462                 0.418864          1
1                0.639175        0.233590                 0.222878          1
2                0.835052        0.403706                 0.213433          1
3                0.884880        1.000000                 0.773711          1
4                0.558419        0.157500                 0.142595          1
..                    …               …                       …          …
564              0.761512        0.097575                 0.105667          1
565              0.559450        0.198502                 0.074315          1
566              0.487285        0.128721                 0.151909          1
567              0.910653        0.497142                 0.452315          1
568              0.000000        0.257441                 0.100682          0

[569 rows x 31 columns]
```

[93]:
```
X
```

[93]:

```
     radius_mean  texture_mean  perimeter_mean  area_mean  smoothness_mean  \
0       0.521037      0.022658        0.545989   0.363733         0.593753
1       0.643144      0.272574        0.615783   0.501591         0.289880
2       0.601496      0.390260        0.595743   0.449417         0.514309
3       0.210090      0.360839        0.233501   0.102906         0.811321
4       0.629893      0.156578        0.630986   0.489290         0.430351
..           …             …               …          …                …
564     0.690000      0.428813        0.678668   0.566490         0.526948
565     0.622320      0.626987        0.604036   0.474019         0.407782
566     0.455251      0.621238        0.445788   0.303118         0.288165
567     0.644564      0.663510        0.665538   0.475716         0.588336
568     0.036869      0.501522        0.028540   0.015907         0.000000

     compactness_mean  concavity_mean  concave points_mean  symmetry_mean  \
0            0.792037        0.703140             0.731113       0.686364
1            0.181768        0.203608             0.348757       0.379798
2            0.431017        0.462512             0.635686       0.509596
3            0.811361        0.565604             0.522863       0.776263
4            0.347893        0.463918             0.518390       0.378283
..                …               …                    …              …
564          0.296055        0.571462             0.690358       0.336364
565          0.257714        0.337395             0.486630       0.349495
566          0.254340        0.216753             0.263519       0.267677
567          0.790197        0.823336             0.755467       0.675253
568          0.074351        0.000000             0.000000       0.266162

     fractal_dimension_mean  …  radius_worst  texture_worst  \
0                  0.605518  …      0.620776       0.141525
1                  0.141323  …      0.606901       0.303571
2                  0.211247  …      0.556386       0.360075
3                  1.000000  …      0.248310       0.385928
4                  0.186816  …      0.519744       0.123934
..                      …  …           …              …
564                0.132056  …      0.623266       0.383262
565                0.113100  …      0.560655       0.699094
566                0.137321  …      0.393099       0.589019
567                0.425442  …      0.633582       0.730277
568                0.187026  …      0.054287       0.489072

     perimeter_worst  area_worst  smoothness_worst  compactness_worst  \
0           0.668310    0.450698          0.601136           0.619292
1           0.539818    0.435214          0.347553           0.154563
2           0.508442    0.374508          0.483590           0.385375
3           0.241347    0.094008          0.915472           0.814012
4           0.506948    0.341575          0.437364           0.172415
```

```
..        …         …            …           …
564  0.576174  0.452664     0.461137    0.178527
565  0.520892  0.379915     0.300007    0.159997
566  0.379949  0.230731     0.282177    0.273705
567  0.668310  0.402035     0.619626    0.815758
568  0.043578  0.020497     0.124084    0.036043

     concavity_worst  concave points_worst  symmetry_worst  \
0           0.568610              0.912027        0.598462
1           0.192971              0.639175        0.233590
2           0.359744              0.835052        0.403706
3           0.548642              0.884880        1.000000
4           0.319489              0.558419        0.157500
..               …                     …              …
564         0.328035              0.761512        0.097575
565         0.256789              0.559450        0.198502
566         0.271805              0.487285        0.128721
567         0.749760              0.910653        0.497142
568         0.000000              0.000000        0.257441

     fractal_dimension_worst
0                   0.418864
1                   0.222878
2                   0.213433
3                   0.773711
4                   0.142595
..                       …
564                 0.105667
565                 0.074315
566                 0.151909
567                 0.452315
568                 0.100682

[569 rows x 30 columns]
```

[114]:
```
y
```

[114]:
```
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
```

[115]:
```
from imblearn.over_sampling import SMOTE
oversample = SMOTE(k_neighbors=3)
X_smote, y_smote = oversample.fit_resample(X, y)
```

[116]:
```
X_smote
```

[116]:
```
     radius_mean  texture_mean  perimeter_mean  area_mean  smoothness_mean  \
0       0.521037      0.022658        0.545989   0.363733         0.593753
1       0.643144      0.272574        0.615783   0.501591         0.289880
2       0.601496      0.390260        0.595743   0.449417         0.514309
3       0.210090      0.360839        0.233501   0.102906         0.811321
4       0.629893      0.156578        0.630986   0.489290         0.430351
..           …             …              …          …               …
709     0.367634      0.581892        0.376596   0.227153         0.538995
710     0.556667      0.317027        0.537899   0.409033         0.273154
711     0.427079      0.383549        0.421857   0.274840         0.440676
712     0.435488      0.330092        0.453632   0.281896         0.477590
713     0.349033      0.365141        0.348066   0.210141         0.407677

     compactness_mean  concavity_mean  concave points_mean  symmetry_mean  \
0            0.792037        0.703140             0.731113       0.686364
1            0.181768        0.203608             0.348757       0.379798
2            0.431017        0.462512             0.635686       0.509596
3            0.811361        0.565604             0.522863       0.776263
4            0.347893        0.463918             0.518390       0.378283
..                …              …                   …             …
709          0.438382        0.386852             0.384937       0.575483
710          0.183928        0.210193             0.278150       0.452697
711          0.325105        0.253603             0.294954       0.512528
712          0.540719        0.545367             0.478627       0.566299
713          0.285548        0.253641             0.285680       0.414782

     fractal_dimension_mean  …  radius_worst  texture_worst  \
0                  0.605518  …      0.620776       0.141525
1                  0.141323  …      0.606901       0.303571
2                  0.211247  …      0.556386       0.360075
3                  1.000000  …      0.248310       0.385928
4                  0.186816  …      0.519744       0.123934
..                      …  …           …            …
709                0.407227  …      0.366481       0.666169
710                0.014982  …      0.500331       0.363063
711                0.221357  …      0.411742       0.509960
712                0.362368  …      0.336391       0.298011
713                0.272496  …      0.292908       0.497094
```

```
     perimeter_worst  area_worst  smoothness_worst  compactness_worst  \
0           0.668310    0.450698          0.601136           0.619292
1           0.539818    0.435214          0.347553           0.154563
2           0.508442    0.374508          0.483590           0.385375
3           0.241347    0.094008          0.915472           0.814012
4           0.506948    0.341575          0.437364           0.172415
..               …           …               …                …
709         0.401470    0.208288          0.631406           0.599183
710         0.475312    0.322063          0.211934           0.183422
711         0.387724    0.235170          0.490489           0.322270
712         0.355222    0.184625          0.454784           0.414086
713         0.306903    0.155099          0.478750           0.311564

     concavity_worst  concave points_worst  symmetry_worst  \
0           0.568610              0.912027        0.598462
1           0.192971              0.639175        0.233590
2           0.359744              0.835052        0.403706
3           0.548642              0.884880        1.000000
4           0.319489              0.558419        0.157500
..               …                   …              …
709         0.546024              0.617763        0.512624
710         0.242989              0.413138        0.262584
711         0.268737              0.472589        0.415778
712         0.481531              0.626685        0.370250
713         0.315265              0.522470        0.292343

     fractal_dimension_worst
0                   0.418864
1                   0.222878
2                   0.213433
3                   0.773711
4                   0.142595
..                       …
709                 0.454898
710                 0.067407
711                 0.213753
712                 0.250317
713                 0.271976

[714 rows x 30 columns]
```

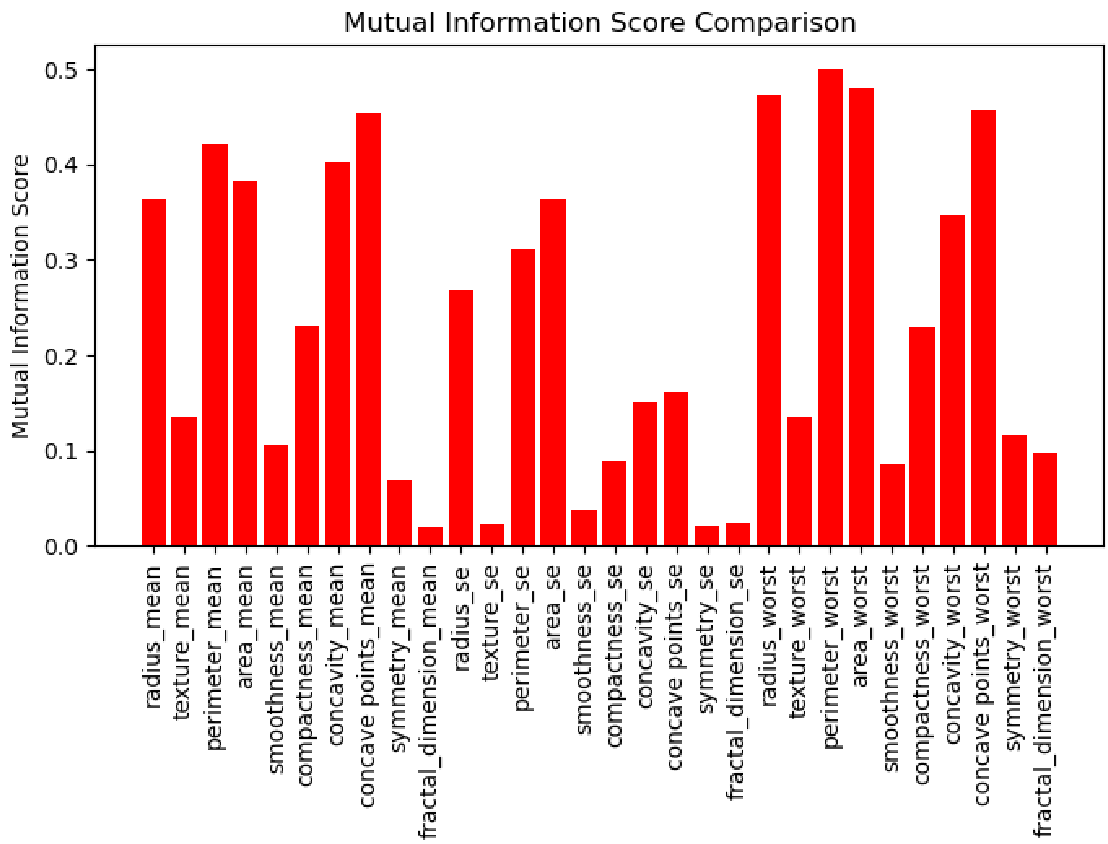
[117]:
```
y_smote.value_counts()
```

[117]:
```
1    357
0    357
Name: diagnosis, dtype: int64
```

[118]:
```
# information gain
MI_score = mutual_info_classif(X_smote,  y_smote , random_state=0)
for feature in zip(feature_columns, MI_score):
    if feature[1]>0.30:
        print(feature)
```

```
('radius_mean', 0.36267197586780675)
('perimeter_mean', 0.42227347410878435)
('area_mean', 0.3830290669088263)
('concavity_mean', 0.40219732276547515)
('concave points_mean', 0.4534264220811439)
('perimeter_se', 0.3102076682911583)
('area_se', 0.3633570139714395)
('radius_worst', 0.4733901657815698)
('perimeter_worst', 0.5001892273843289)
('area_worst', 0.47923049764730474)
('concavity_worst', 0.34669874006624646)
('concave points_worst', 0.45661096537561363)
```

[119]:
```
plt.figure(figsize=(8,4))
plt.bar(x=feature_columns, height=MI_score, color='red')
plt.xticks(rotation='vertical')
plt.ylabel('Mutual Information Score')
plt.title('Mutual Information Score Comparison')
plt.show()
```



[120]:
```
selected_features = [feature_columns[i] for i in range(len(feature_columns)) if␣
↪MI_score[i] > 0.3 ]
X_selected = X_smote[selected_features]
```

[121]:
```
X_selected
```

[121]:
```
     radius_mean  perimeter_mean  area_mean  concavity_mean  \
0       0.521037        0.545989   0.363733        0.703140
1       0.643144        0.615783   0.501591        0.203608
2       0.601496        0.595743   0.449417        0.462512
3       0.210090        0.233501   0.102906        0.565604
4       0.629893        0.630986   0.489290        0.463918
..           …             …          …               …
709     0.367634        0.376596   0.227153        0.386852
710     0.556667        0.537899   0.409033        0.210193
711     0.427079        0.421857   0.274840        0.253603
712     0.435488        0.453632   0.281896        0.545367
713     0.349033        0.348066   0.210141        0.253641

     concave points_mean  perimeter_se   area_se  radius_worst  \
```

```
0                0.731113      0.369034  0.273811      0.620776
1                0.348757      0.124440  0.125660      0.606901
2                0.635686      0.180370  0.162922      0.556386
3                0.522863      0.126655  0.038155      0.248310
4                0.518390      0.220563  0.163688      0.519744
..                    …            …        …            …
709              0.384937      0.107996  0.053709      0.366481
710              0.278150      0.203343  0.135932      0.500331
711              0.294954      0.087632  0.060754      0.411742
712              0.478627      0.111577  0.053725      0.336391
713              0.285680      0.081291  0.036075      0.292908

     perimeter_worst  area_worst  concavity_worst  concave points_worst
0           0.668310    0.450698         0.568610              0.912027
1           0.539818    0.435214         0.192971              0.639175
2           0.508442    0.374508         0.359744              0.835052
3           0.241347    0.094008         0.548642              0.884880
4           0.506948    0.341575         0.319489              0.558419
..               …           …               …                    …
709         0.401470    0.208288         0.546024              0.617763
710         0.475312    0.322063         0.242989              0.413138
711         0.387724    0.235170         0.268737              0.472589
712         0.355222    0.184625         0.481531              0.626685
713         0.306903    0.155099         0.315265              0.522470

[714 rows x 12 columns]
```

[122]:
```
X_train, X_test, y_train, y_test = train_test_split(X_selected, y_smote,␣
 ↪test_size=0.2, random_state=0)
```

[123]:
```
knn(X_train, X_test, y_train, y_test)
```

```
Accuracy du KNN :  0.951048951048951
precision score du KNN :  0.9393939393939394
recall score du KNN :  0.9538461538461539
f1 score du KNN :  0.9465648854961831
```

[124]:
```
knn_grid_search(X_selected, y_smote)
```

```
grid best score accuracy 0.9692488262910798
grid best score precision 0.971979836979837
grid best score recall 0.9804761904761904
grid best score f1 score 0.9696325594357622
{'metric': 'euclidean', 'n_neighbors': 1, 'weights': 'uniform'}
KNeighborsClassifier(metric='euclidean', n_neighbors=1)
```

[125]:
```
svm(X_train, X_test, y_train, y_test)
```

```
              precision    recall  f1-score   support
```

```
           0       0.97      0.96      0.97        78
           1       0.95      0.97      0.96        65

    accuracy                           0.97       143
   macro avg       0.96      0.97      0.96       143
weighted avg       0.97      0.97      0.97       143
```

[126]:
```
svm_grid_search(X_train,X_test,y_train ,y_test)
```

```
              precision    recall  f1-score   support

           0       0.97      0.99      0.98        78
           1       0.98      0.97      0.98        65

    accuracy                           0.98       143
   macro avg       0.98      0.98      0.98       143
weighted avg       0.98      0.98      0.98       143

Accuracy :  0.9790209790209791
```

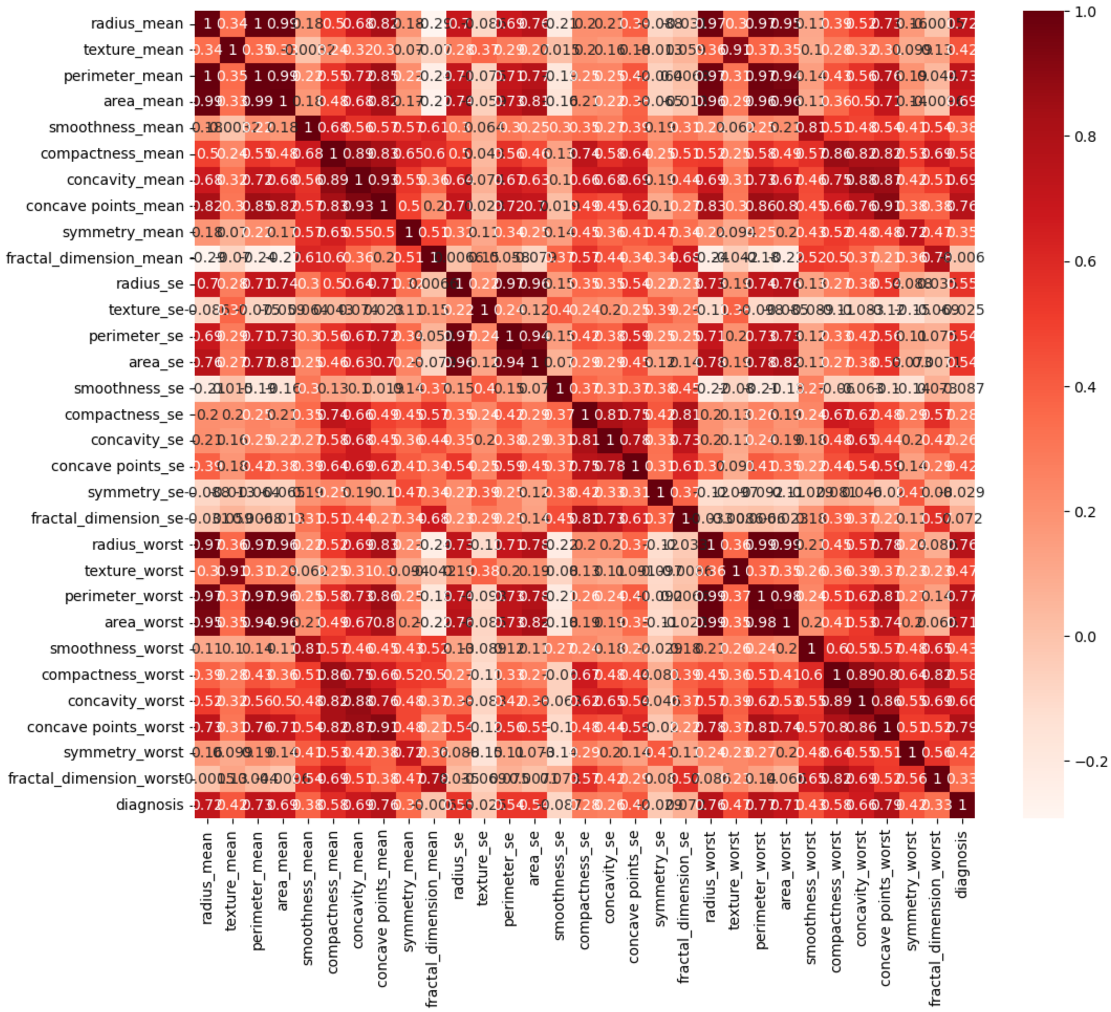
[127]:
```
data_smote = pd.DataFrame(X_smote, columns = X_smote.columns)
```

[128]:
```
data_smote["diagnosis"] = y_smote
```

[129]:
```
data_smote['diagnosis'].value_counts()
```

[129]:
```
1    357
0    357
Name: diagnosis, dtype: int64
```

[130]:
```
#Using Pearson Correlation
plt.figure(figsize=(12,10))
cor = data_smote.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.Reds)
plt.show()
```



```
[131]: #Correlation with output variable
cor_target = abs(cor["diagnosis"])
#Selecting highly correlated features
relevant_features = list(cor_target[abs(cor_target) > 0.5].index)
relevant_features.remove('diagnosis')
relevant_features
# we choose the variables that have a correlation higher to 0.5 with our target␣
↪variable
```

```
[131]: ['radius_mean',
 'perimeter_mean',
 'area_mean',
 'compactness_mean',
 'concavity_mean',
 'concave points_mean',
```

```
 'radius_se',
 'perimeter_se',
 'area_se',
 'radius_worst',
 'perimeter_worst',
 'area_worst',
 'compactness_worst',
 'concavity_worst',
 'concave points_worst']
```

[132]:
```
X_selected_corr = X_smote[relevant_features]
```

[133]:
```
X_selected_corr
```

[133]:
```
     radius_mean  perimeter_mean  area_mean  compactness_mean  concavity_mean  \
0       0.521037        0.545989   0.363733          0.792037        0.703140
1       0.643144        0.615783   0.501591          0.181768        0.203608
2       0.601496        0.595743   0.449417          0.431017        0.462512
3       0.210090        0.233501   0.102906          0.811361        0.565604
4       0.629893        0.630986   0.489290          0.347893        0.463918
..           …               …          …                …              …
709     0.367634        0.376596   0.227153          0.438382        0.386852
710     0.556667        0.537899   0.409033          0.183928        0.210193
711     0.427079        0.421857   0.274840          0.325105        0.253603
712     0.435488        0.453632   0.281896          0.540719        0.545367
713     0.349033        0.348066   0.210141          0.285548        0.253641

     concave points_mean  radius_se  perimeter_se   area_se  radius_worst  \
0               0.731113   0.356147      0.369034  0.273811      0.620776
1               0.348757   0.156437      0.124440  0.125660      0.606901
2               0.635686   0.229622      0.180370  0.162922      0.556386
3               0.522863   0.139091      0.126655  0.038155      0.248310
4               0.518390   0.233822      0.220563  0.163688      0.519744
..                   …          …            …        …            …
709             0.384937   0.100471      0.107996  0.053709      0.366481
710             0.278150   0.204485      0.203343  0.135932      0.500331
711             0.294954   0.104716      0.087632  0.060754      0.411742
712             0.478627   0.083650      0.111577  0.053725      0.336391
713             0.285680   0.063979      0.081291  0.036075      0.292908

     perimeter_worst  area_worst  compactness_worst  concavity_worst  \
0           0.668310    0.450698           0.619292         0.568610
1           0.539818    0.435214           0.154563         0.192971
2           0.508442    0.374508           0.385375         0.359744
3           0.241347    0.094008           0.814012         0.548642
4           0.506948    0.341575           0.172415         0.319489
..                …           …                  …                …
```

```
709        0.401470   0.208288           0.599183          0.546024
710        0.475312   0.322063           0.183422          0.242989
711        0.387724   0.235170           0.322270          0.268737
712        0.355222   0.184625           0.414086          0.481531
713        0.306903   0.155099           0.311564          0.315265

     concave points_worst
0                0.912027
1                0.639175
2                0.835052
3                0.884880
4                0.558419
..                    …
709              0.617763
710              0.413138
711              0.472589
712              0.626685
713              0.522470

[714 rows x 15 columns]
```

[134]:
```
# splitting the data
X_train, X_test, y_train, y_test = train_test_split(X_selected_corr, y_smote,
 ↪test_size=0.2, random_state=0)
```

[135]:
```
knn(X_train, X_test, y_train, y_test)
```

```
Accuracy du KNN :  0.958041958041958
precision score du KNN :  0.9402985074626866
recall score du KNN :  0.9692307692307692
f1 score du KNN :  0.9545454545454547
```

[136]:
```
knn_grid_search(X_selected_corr, y_smote)
```

```
grid best score accuracy 0.9621870109546166
grid best score precision 0.9686363636363635
grid best score recall 0.9748412698412698
grid best score f1 score 0.9628758759094641
{'metric': 'manhattan', 'n_neighbors': 1, 'weights': 'uniform'}
KNeighborsClassifier(metric='manhattan', n_neighbors=1)
```

[137]:
```
svm(X_train, X_test, y_train, y_test)
```

```
              precision    recall  f1-score   support

           0       0.97      0.96      0.97        78
           1       0.95      0.97      0.96        65

    accuracy                           0.97       143
```

```
   macro avg       0.96      0.97      0.96       143
weighted avg       0.97      0.97      0.97       143
```

[138]:
```
svm_grid_search(X_train,X_test,y_train ,y_test)
```

```
              precision    recall  f1-score   support

           0       0.97      0.97      0.97        78
           1       0.97      0.97      0.97        65

    accuracy                           0.97       143
   macro avg       0.97      0.97      0.97       143
weighted avg       0.97      0.97      0.97       143

Accuracy :  0.972027972027972
```

[139]:
```
# Select Kbest
selector = SelectKBest(chi2, k=10)
X_new = selector.fit_transform(X_smote, y_smote)
mask = selector.get_support()
selected_features = X_smote.columns[mask]
selected = selected_features.values
selected
```

[139]:
```
array(['perimeter_mean', 'area_mean', 'concavity_mean',
       'concave points_mean', 'radius_worst', 'perimeter_worst',
       'area_worst', 'concavity_worst', 'concave points_worst',
       'diagnosis'], dtype=object)
```

[140]:
```
X_selected_kbest_smote = X_smote[selected]
```

[141]:
```
X_selected_kbest_smote
```

[141]:
```
     perimeter_mean  area_mean  concavity_mean  concave points_mean  \
0          0.545989   0.363733        0.703140             0.731113
1          0.615783   0.501591        0.203608             0.348757
2          0.595743   0.449417        0.462512             0.635686
3          0.233501   0.102906        0.565604             0.522863
4          0.630986   0.489290        0.463918             0.518390
..              …          …             …                   …
709        0.376596   0.227153        0.386852             0.384937
710        0.537899   0.409033        0.210193             0.278150
711        0.421857   0.274840        0.253603             0.294954
712        0.453632   0.281896        0.545367             0.478627
713        0.348066   0.210141        0.253641             0.285680

     radius_worst  perimeter_worst  area_worst  concavity_worst  \
```

```
0        0.620776         0.668310   0.450698         0.568610
1        0.606901         0.539818   0.435214         0.192971
2        0.556386         0.508442   0.374508         0.359744
3        0.248310         0.241347   0.094008         0.548642
4        0.519744         0.506948   0.341575         0.319489
..            …              …          …                …
709      0.366481         0.401470   0.208288         0.546024
710      0.500331         0.475312   0.322063         0.242989
711      0.411742         0.387724   0.235170         0.268737
712      0.336391         0.355222   0.184625         0.481531
713      0.292908         0.306903   0.155099         0.315265

     concave points_worst  diagnosis
0                0.912027          1
1                0.639175          1
2                0.835052          1
3                0.884880          1
4                0.558419          1
..                    …         …
709              0.617763          1
710              0.413138          1
711              0.472589          1
712              0.626685          1
713              0.522470          1

[714 rows x 10 columns]
```

[142]:

```
# splitting the data
X_train, X_test, y_train, y_test = train_test_split(X_selected_kbest_smote,
 ↪y_smote, test_size=0.2, random_state=0)
```

[143]:

```
knn(X_train, X_test, y_train, y_test)
```

```
Accuracy du KNN :  1.0
precision score du KNN :  1.0
recall score du KNN :  1.0
f1 score du KNN :  1.0
```

[144]:

```
knn_grid_search(X_selected_kbest_smote, y_smote)
```

```
grid best score accuracy 1.0
grid best score precision 1.0
grid best score recall 1.0
grid best score f1 score 1.0
{'metric': 'euclidean', 'n_neighbors': 1, 'weights': 'uniform'}
KNeighborsClassifier(metric='euclidean', n_neighbors=1)
```

[145]:

```
svm(X_train, X_test, y_train, y_test)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        78
           1       1.00      1.00      1.00        65

    accuracy                           1.00       143
   macro avg       1.00      1.00      1.00       143
weighted avg       1.00      1.00      1.00       143
```

[146]:
```
svm_grid_search(X_train, X_test, y_train, y_test)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        78
           1       1.00      1.00      1.00        65

    accuracy                           1.00       143
   macro avg       1.00      1.00      1.00       143
weighted avg       1.00      1.00      1.00       143

Accuracy :  1.0
```

3.0.1 wrapper

[147]:
```
X_smote
```

[147]:
```
     radius_mean  texture_mean  perimeter_mean  area_mean  smoothness_mean  \
0       0.521037      0.022658        0.545989   0.363733         0.593753
1       0.643144      0.272574        0.615783   0.501591         0.289880
2       0.601496      0.390260        0.595743   0.449417         0.514309
3       0.210090      0.360839        0.233501   0.102906         0.811321
4       0.629893      0.156578        0.630986   0.489290         0.430351
..           …             …               …          …               …
709     0.367634      0.581892        0.376596   0.227153         0.538995
710     0.556667      0.317027        0.537899   0.409033         0.273154
711     0.427079      0.383549        0.421857   0.274840         0.440676
712     0.435488      0.330092        0.453632   0.281896         0.477590
713     0.349033      0.365141        0.348066   0.210141         0.407677

     compactness_mean  concavity_mean  concave points_mean  symmetry_mean  \
0            0.792037        0.703140             0.731113       0.686364
1            0.181768        0.203608             0.348757       0.379798
2            0.431017        0.462512             0.635686       0.509596
3            0.811361        0.565604             0.522863       0.776263
4            0.347893        0.463918             0.518390       0.378283
..                …               …                    …              …
709          0.438382        0.386852             0.384937       0.575483
```

```
710         0.183928      0.210193           0.278150      0.452697
711         0.325105      0.253603           0.294954      0.512528
712         0.540719      0.545367           0.478627      0.566299
713         0.285548      0.253641           0.285680      0.414782

     fractal_dimension_mean  …  texture_worst  perimeter_worst  area_worst  \
0                  0.605518  …       0.141525         0.668310    0.450698
1                  0.141323  …       0.303571         0.539818    0.435214
2                  0.211247  …       0.360075         0.508442    0.374508
3                  1.000000  …       0.385928         0.241347    0.094008
4                  0.186816  …       0.123934         0.506948    0.341575
..                      …   …            …              …           …
709                0.407227  …       0.666169         0.401470    0.208288
710                0.014982  …       0.363063         0.475312    0.322063
711                0.221357  …       0.509960         0.387724    0.235170
712                0.362368  …       0.298011         0.355222    0.184625
713                0.272496  …       0.497094         0.306903    0.155099

     smoothness_worst  compactness_worst  concavity_worst  \
0            0.601136           0.619292         0.568610
1            0.347553           0.154563         0.192971
2            0.483590           0.385375         0.359744
3            0.915472           0.814012         0.548642
4            0.437364           0.172415         0.319489
..                …                 …                …
709          0.631406           0.599183         0.546024
710          0.211934           0.183422         0.242989
711          0.490489           0.322270         0.268737
712          0.454784           0.414086         0.481531
713          0.478750           0.311564         0.315265

     concave points_worst  symmetry_worst  fractal_dimension_worst  diagnosis
0                0.912027        0.598462                 0.418864          1
1                0.639175        0.233590                 0.222878          1
2                0.835052        0.403706                 0.213433          1
3                0.884880        1.000000                 0.773711          1
4                0.558419        0.157500                 0.142595          1
..                    …              …                      …        …
709              0.617763        0.512624                 0.454898          1
710              0.413138        0.262584                 0.067407          1
711              0.472589        0.415778                 0.213753          1
712              0.626685        0.370250                 0.250317          1
713              0.522470        0.292343                 0.271976          1

[714 rows x 31 columns]
```

[148]:
```
y_smote
```

[148]:
```
0      1
1      1
2      1
3      1
4      1
      ..
709    1
710    1
711    1
712    1
713    1
Name: diagnosis, Length: 714, dtype: int64
```

[149]:
```
X_train, X_test, y_train, y_test = train_test_split( X_smote, y_smote,␣
 ↪test_size=0.33, random_state=42)
# create the classifier with n_estimators = 100

clf = RandomForestClassifier(n_estimators=100, random_state=0)



# fit the model to the training set

clf.fit(X_train, y_train)

# view the feature scores

feature_scores = pd.Series(clf.feature_importances_, index=X_train.columns).
 ↪sort_values(ascending=False)

selected = feature_scores[feature_scores.values >0.01]
selected_features = selected.index.values
```

[150]:
```
selected_features
```

[150]:
```
array(['diagnosis', 'radius_worst', 'concave points_worst', 'area_worst',
       'concave points_mean', 'perimeter_worst', 'concavity_mean',
       'perimeter_mean', 'radius_mean', 'concavity_worst', 'area_se',
       'texture_mean'], dtype=object)
```

[151]:
```
X_selected_rf_smote = X_smote[selected_features]
```

[152]:
```
# splitting the data
X_train, X_test, y_train, y_test = train_test_split(X_selected_rf_smote,␣
 ↪y_smote, test_size=0.2, random_state=0)
```

[153]:
```
knn(X_train, X_test, y_train, y_test)
```

```
Accuracy du KNN :  1.0
precision score du KNN :  1.0
recall score du KNN :  1.0
f1 score du KNN :  1.0
```

[154]:
```
knn_grid_search(X_selected_rf_smote , y_smote)
```

```
grid best score accuracy 1.0
grid best score precision 1.0
grid best score recall 1.0
grid best score f1 score 1.0
{'metric': 'euclidean', 'n_neighbors': 1, 'weights': 'uniform'}
KNeighborsClassifier(metric='euclidean', n_neighbors=1)
```

[155]:
```
svm(X_train, X_test, y_train, y_test)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        78
           1       1.00      1.00      1.00        65

    accuracy                           1.00       143
   macro avg       1.00      1.00      1.00       143
weighted avg       1.00      1.00      1.00       143
```

[156]:
```
svm_grid_search(X_train, X_test, y_train, y_test)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        78
           1       1.00      1.00      1.00        65

    accuracy                           1.00       143
   macro avg       1.00      1.00      1.00       143
weighted avg       1.00      1.00      1.00       143

Accuracy :  1.0
```

3.0.2 Wrapper recursive features elemination

[109]:
```python
from sklearn.feature_selection import RFE
from sklearn.svm import SVC

# Initialize an SVM classifier and an RFE feature selector
svm = SVC(kernel='linear')
rfe = RFE(estimator=svm, n_features_to_select=10, step=1)

# Fit the RFE selector to the data and get the selected feature indices
```

```
rfe.fit(X_smote, y_smote)
selected_indices = rfe.get_support(indices=True)

selected_names = [X_resampled.columns[i] for i, selected in enumerate(rfe.
 ↪support_) if selected]

# Print the selected feature names
print('Selected features:', selected_names)
```

```
Selected features: ['radius_mean', 'concave points_mean', 'radius_se',
'radius_worst', 'texture_worst', 'perimeter_worst', 'area_worst',
'smoothness_worst', 'concave points_worst', 'symmetry_worst']
```

[110]:
```
X_selected_rfe_smote = X_smote[selected_names]
```

[111]:
```
X_selected_rfe_smote
```

[111]:
```
     radius_mean  concave points_mean  radius_se  radius_worst  texture_worst  \
0       0.521037             0.731113   0.356147      0.620776       0.141525
1       0.643144             0.348757   0.156437      0.606901       0.303571
2       0.601496             0.635686   0.229622      0.556386       0.360075
3       0.210090             0.522863   0.139091      0.248310       0.385928
4       0.629893             0.518390   0.233822      0.519744       0.123934
..           …                    …          …             …              …
709     0.388456             0.256380   0.108656      0.306150       0.437301
710     0.306098             0.470705   0.103634      0.315222       0.302132
711     0.365918             0.223111   0.077374      0.332806       0.236218
712     0.638719             0.306977   0.205080      0.579789       0.377599
713     0.687596             0.715509   0.286213      0.660887       0.364345

     perimeter_worst  area_worst  smoothness_worst  concave points_worst  \
0           0.668310    0.450698          0.601136              0.912027
1           0.539818    0.435214          0.347553              0.639175
2           0.508442    0.374508          0.483590              0.835052
3           0.241347    0.094008          0.915472              0.884880
4           0.506948    0.341575          0.437364              0.558419
..               …           …                 …                     …
709         0.298084    0.161883          0.332679              0.402120
710         0.298679    0.170446          0.761657              0.685184
711         0.318329    0.183558          0.519499              0.444969
712         0.554043    0.385231          0.346674              0.506193
713         0.660659    0.466581          0.378645              0.802264

     symmetry_worst
0          0.598462
1          0.233590
2          0.403706
3          1.000000
```

```
4          0.157500
..              …
709        0.265297
710        0.328455
711        0.340645
712        0.139067
713        0.254533

[714 rows x 10 columns]
```

[116]:
```
# splitting the data
X_train, X_test, y_train, y_test = train_test_split(X_selected_rfe_smote,␣
↪y_smote, test_size=0.2, random_state=0)
```

[117]:
```
knn(X_train, X_test, y_train, y_test)
```

```
Accuracy du KNN :  0.972027972027972
precision score du KNN :  0.9841269841269841
recall score du KNN :  0.9538461538461539
f1 score du KNN :  0.96875
```

[162]:
```
knn_grid_search(X_selected_rfe_smote,y_smote)
```

```
grid best score accuracy 1.0
grid best score precision 1.0
grid best score recall 1.0
grid best score f1 score 1.0
{'metric': 'euclidean', 'n_neighbors': 1, 'weights': 'uniform'}
KNeighborsClassifier(metric='euclidean', n_neighbors=1)
```

[163]:
```
def svm(X_train, X_test, y_train, y_test):
    svm = SVC()
    svm.fit(X_train, y_train)
    y_pred = svm.predict(X_test)
    print(classification_report(y_test, y_pred))
```

[164]:
```
svm(X_train, X_test, y_train, y_test)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        78
           1       1.00      1.00      1.00        65

    accuracy                           1.00       143
   macro avg       1.00      1.00      1.00       143
weighted avg       1.00      1.00      1.00       143
```

[165]:
```
svm_grid_search(X_train, X_test, y_train, y_test)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        78
           1       1.00      1.00      1.00        65

    accuracy                           1.00       143
   macro avg       1.00      1.00      1.00       143
weighted avg       1.00      1.00      1.00       143

Accuracy :  1.0
```

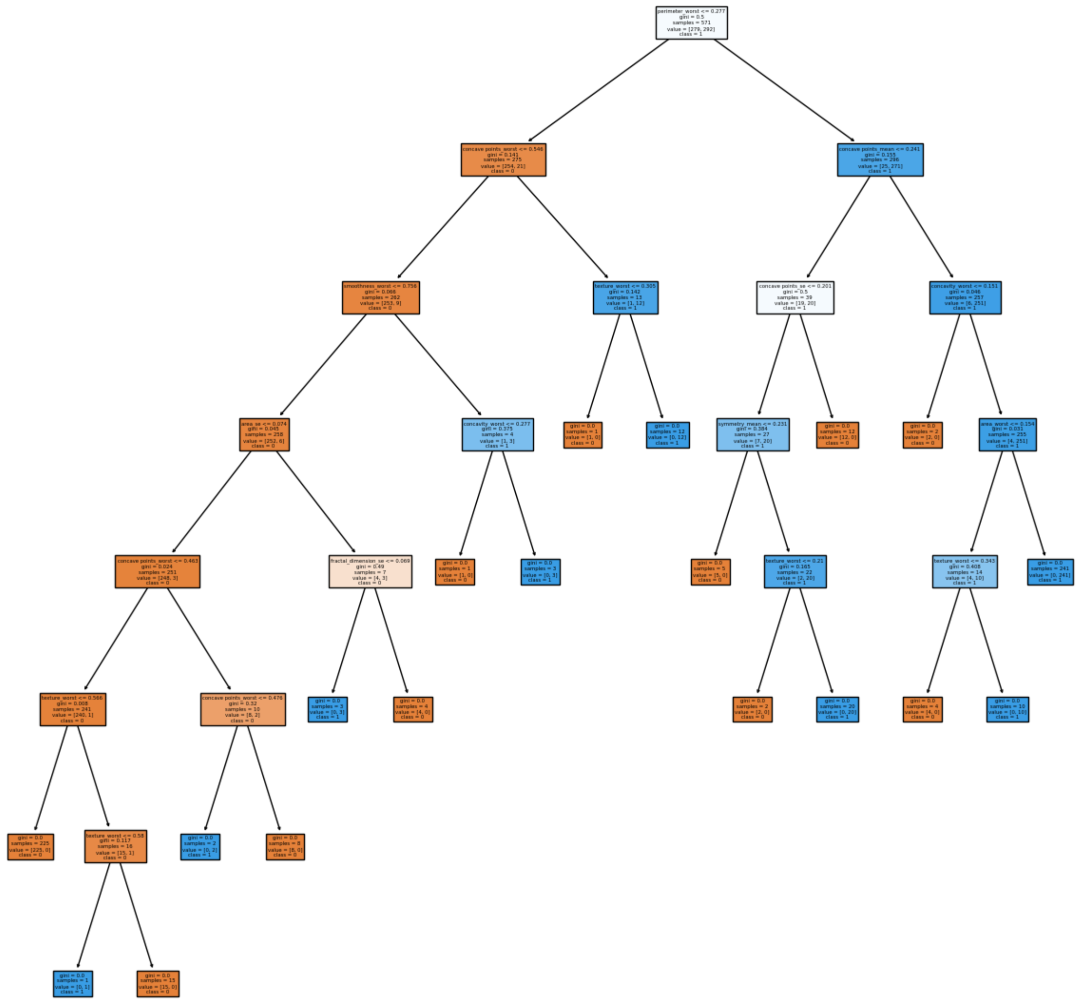
[166]:
```
## Decision tree
```

[122]:
```
# splitting the data
X_train, X_test, y_train, y_test = train_test_split(X_smote, y_smote,␣
 ↪test_size=0.2, random_state=0)
```

[123]:
```
clf = tree.DecisionTreeClassifier()
clf.fit(X_train, y_train)
y_pred= clf.predict(X_test)
acc2 = metrics.accuracy_score(y_test, y_pred)
precision =  metrics.precision_score(y_test, y_pred)
recall = metrics.recall_score(y_test, y_pred)
f1_score = metrics.f1_score(y_test , y_pred)
print("Accuracy:", acc2)
print("Precision: ",precision)
print("Recall: ",recall)
print("F1 score : ",f1_score)
```

```
Accuracy: 0.951048951048951
Precision:  0.9264705882352942
Recall:  0.9692307692307692
F1 score :  0.9473684210526316
```

[120]:
```
from sklearn import tree
clf = tree.DecisionTreeClassifier()
clf.fit(X_train, y_train)
fig = plt.figure(figsize=(15,15))
_ = tree.plot_tree(clf,
 feature_names=feature_columns
                   ,
 class_names=["0","1"],
 filled=True)
```



[131]:
```
#Grid search
DT = tree.DecisionTreeClassifier()
params = {
 'criterion': ['gini', 'entropy'],
 'max_depth': [2, 4, 6, 8, 10],
 'min_samples_split': [2, 4, 6, 8, 10],
 'min_samples_leaf': [1, 2, 3, 4, 5]
    }
grid = GridSearchCV(DT, params, cv = 10, scoring = 'accuracy')
grid.fit(X_smote,y_smote)

grid1 = GridSearchCV(DT, params, cv = 10, scoring = 'precision')
grid1.fit(X_smote,y_smote)
```

```
grid2 = GridSearchCV(DT, params, cv = 10, scoring = 'recall')
grid2.fit(X_smote,y_smote)

grid3 = GridSearchCV(DT, params, cv = 10, scoring = 'f1')
grid3.fit(X_smote,y_smote)

print("Accuracy",grid.best_score_)
print("Precision",grid1.best_score_)
print("Recall",grid2.best_score_)
print("f1 score",grid3.best_score_)

print(grid.best_params_)
print(grid.best_estimator_)
```

```
Accuracy 0.9594483568075116
Precision 0.966233582704171
Recall 0.9804761904761905
f1 score 0.9619993310401753
{'criterion': 'entropy', 'max_depth': 8, 'min_samples_leaf': 3,
'min_samples_split': 8}
DecisionTreeClassifier(criterion='entropy', max_depth=8, min_samples_leaf=3,
                       min_samples_split=8)
```

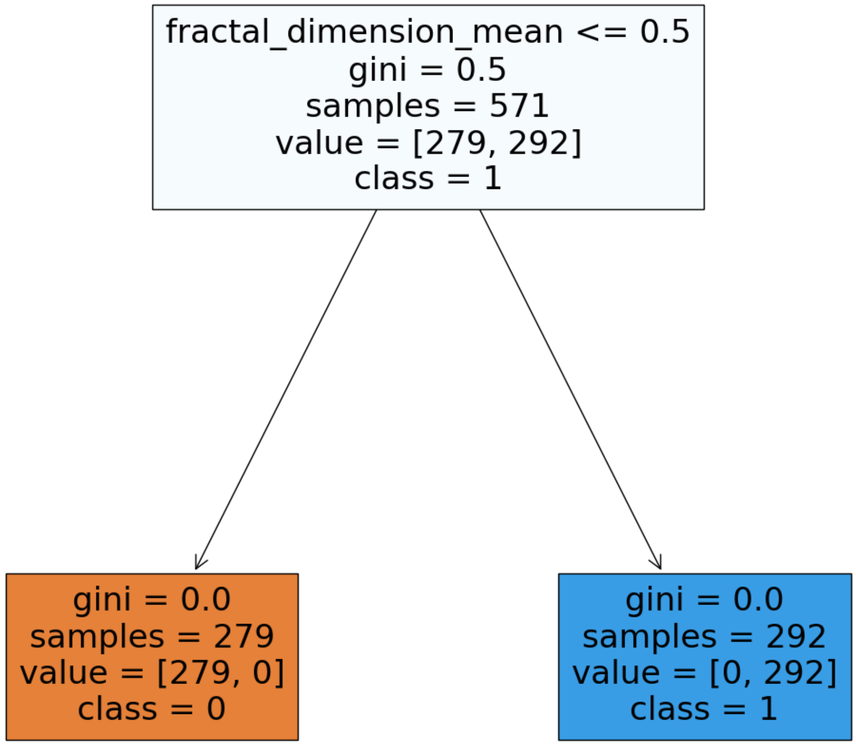
[132]:

```
# Use a pruning algorithm to prune the decision tree
clf = tree.DecisionTreeClassifier()
path = clf.cost_complexity_pruning_path(X_train, y_train)
ccp_alphas = path.ccp_alphas[:-1]
clfs = []
for ccp_alpha in ccp_alphas:
    clf = tree.DecisionTreeClassifier(ccp_alpha=ccp_alpha)
    clf.fit(X_train, y_train)
    clfs.append(clf)
# Evaluate the pruned decision tree using the testing data
acc_scores = []
for clf in clfs:
    y_pred = clf.predict(X_test)
    acc_score = accuracy_score(y_test, y_pred)
    prec_score = precision_score(y_test , y_pred)
    acc_scores.append(acc_score)
# Find the best pruning parameter based on accuracy score
best_clf = clfs[acc_scores.index(max(acc_scores))]
# Evaluate the best pruned decision tree using the testing data
y_pred = best_clf.predict(X_test)
acc_score = accuracy_score(y_test, y_pred)
precision = precision_score(y_test , y_pred)
recall = recall_score(y_test , y_pred)
```

```
f1_score = metrics.f1_score(y_test , y_pred)

print("Accuracy score: {:.2f}".format(acc_score))
print("precision score: {:.2f}".format(precision))
print("recall score: {:.2f}".format(recall))
print("f1 score: {:.2f}".format(f1_score))

#print("ccp_alpha: {:.3f}".format(ccp_alphas[acc_scores.
↪index(max(acc_scores))]))
```

```
Accuracy score: 0.96
precision score: 0.95
recall score: 0.97
f1 score: 0.96
```

[171]:
```
from sklearn import tree
fig = plt.figure(figsize=(15,15))
_ = tree.plot_tree(best_clf,
 feature_names=feature_columns,
 class_names=["0","1"],filled=True)
```



3.1 Unbalanced data Over sampler methods

[172]:
```
X
```

[172]:
```
     radius_mean  texture_mean  perimeter_mean  area_mean  smoothness_mean  \
0       0.521037      0.022658        0.545989   0.363733         0.593753
1       0.643144      0.272574        0.615783   0.501591         0.289880
2       0.601496      0.390260        0.595743   0.449417         0.514309
3       0.210090      0.360839        0.233501   0.102906         0.811321
4       0.629893      0.156578        0.630986   0.489290         0.430351
..           …             …              …         …              …
564     0.690000      0.428813        0.678668   0.566490         0.526948
565     0.622320      0.626987        0.604036   0.474019         0.407782
```

```
566     0.455251      0.621238        0.445788   0.303118          0.288165
567     0.644564      0.663510        0.665538   0.475716          0.588336
568     0.036869      0.501522        0.028540   0.015907          0.000000

     compactness_mean  concavity_mean  concave points_mean  symmetry_mean  \
0            0.792037        0.703140             0.731113       0.686364
1            0.181768        0.203608             0.348757       0.379798
2            0.431017        0.462512             0.635686       0.509596
3            0.811361        0.565604             0.522863       0.776263
4            0.347893        0.463918             0.518390       0.378283
..                …               …                    …             …
564          0.296055        0.571462             0.690358       0.336364
565          0.257714        0.337395             0.486630       0.349495
566          0.254340        0.216753             0.263519       0.267677
567          0.790197        0.823336             0.755467       0.675253
568          0.074351        0.000000             0.000000       0.266162

     fractal_dimension_mean  …  radius_worst  texture_worst  \
0                  0.605518  …      0.620776       0.141525
1                  0.141323  …      0.606901       0.303571
2                  0.211247  …      0.556386       0.360075
3                  1.000000  …      0.248310       0.385928
4                  0.186816  …      0.519744       0.123934
..                      …   …            …              …
564                0.132056  …      0.623266       0.383262
565                0.113100  …      0.560655       0.699094
566                0.137321  …      0.393099       0.589019
567                0.425442  …      0.633582       0.730277
568                0.187026  …      0.054287       0.489072

     perimeter_worst  area_worst  smoothness_worst  compactness_worst  \
0           0.668310    0.450698          0.601136           0.619292
1           0.539818    0.435214          0.347553           0.154563
2           0.508442    0.374508          0.483590           0.385375
3           0.241347    0.094008          0.915472           0.814012
4           0.506948    0.341575          0.437364           0.172415
..               …           …                …                 …
564         0.576174    0.452664          0.461137           0.178527
565         0.520892    0.379915          0.300007           0.159997
566         0.379949    0.230731          0.282177           0.273705
567         0.668310    0.402035          0.619626           0.815758
568         0.043578    0.020497          0.124084           0.036043

     concavity_worst  concave points_worst  symmetry_worst  \
0           0.568610              0.912027        0.598462
1           0.192971              0.639175        0.233590
2           0.359744              0.835052        0.403706
```

```
3           0.548642               0.884880        1.000000
4           0.319489               0.558419        0.157500
..               ...                    ...            ...
564         0.328035               0.761512        0.097575
565         0.256789               0.559450        0.198502
566         0.271805               0.487285        0.128721
567         0.749760               0.910653        0.497142
568         0.000000               0.000000        0.257441

     fractal_dimension_worst
0                   0.418864
1                   0.222878
2                   0.213433
3                   0.773711
4                   0.142595
..                       ...
564                 0.105667
565                 0.074315
566                 0.151909
567                 0.452315
568                 0.100682

[569 rows x 30 columns]
```

[173]:

```
y
```

[173]:

```
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
```

3.2 Over sampling

[174]:

```
from imblearn.over_sampling import RandomOverSampler
rus = RandomOverSampler()
X_oversampling, y_oversampling = rus.fit_resample(X, y)
```

[175]:

```
X_oversampling
```

[175]:
```
     radius_mean  texture_mean  perimeter_mean  area_mean  smoothness_mean  \
0       0.521037      0.022658        0.545989   0.363733         0.593753
1       0.643144      0.272574        0.615783   0.501591         0.289880
2       0.601496      0.390260        0.595743   0.449417         0.514309
3       0.210090      0.360839        0.233501   0.102906         0.811321
4       0.629893      0.156578        0.630986   0.489290         0.430351
..           …             …              …          …              …
709     0.692366      0.425093        0.695253   0.535949         0.578406
710     0.394671      0.255665        0.410545   0.241697         0.730071
711     0.395617      0.153872        0.405708   0.237922         0.493545
712     0.313739      0.516402        0.305853   0.186299         0.381421
713     0.434427      0.400068        0.431276   0.282630         0.434865

     compactness_mean  concavity_mean  concave points_mean  symmetry_mean  \
0            0.792037        0.703140             0.731113       0.686364
1            0.181768        0.203608             0.348757       0.379798
2            0.431017        0.462512             0.635686       0.509596
3            0.811361        0.565604             0.522863       0.776263
4            0.347893        0.463918             0.518390       0.378283
..                …              …                    …              …
709          0.580701        0.658388             0.776342       0.556566
710          0.641126        0.573571             0.617296       0.675758
711          0.595424        0.486645             0.484891       0.737879
712          0.201613        0.202085             0.223111       0.277273
713          0.334397        0.244377             0.278976       0.555556

     fractal_dimension_mean  …  radius_worst  texture_worst  \
0                  0.605518  …      0.620776       0.141525
1                  0.141323  …      0.606901       0.303571
2                  0.211247  …      0.556386       0.360075
3                  1.000000  …      0.248310       0.385928
4                  0.186816  …      0.519744       0.123934
..                      …  …           …             …
709                0.339090  …      0.651014       0.445629
710                0.547599  …      0.348630       0.283582
711                0.428812  …      0.360726       0.188166
712                0.184288  …      0.322305       0.619670
713                0.188500  …      0.410530       0.523987

     perimeter_worst  area_worst  smoothness_worst  compactness_worst  \
0           0.668310    0.450698          0.601136           0.619292
1           0.539818    0.435214          0.347553           0.154563
2           0.508442    0.374508          0.483590           0.385375
3           0.241347    0.094008          0.915472           0.814012
4           0.506948    0.341575          0.437364           0.172415
..               …           …                 …                  …
709         0.605558    0.465936          0.521891           0.528189
```

```
710        0.345585    0.182757          0.695569           0.410406
711        0.371981    0.195561          0.447930           0.551183
712        0.289805    0.177276          0.365383           0.162034
713        0.394890    0.243266          0.451232           0.269921

     concavity_worst  concave points_worst  symmetry_worst  \
0           0.568610              0.912027        0.598462
1           0.192971              0.639175        0.233590
2           0.359744              0.835052        0.403706
3           0.548642              0.884880        1.000000
4           0.319489              0.558419        0.157500
..               …                     …               …
709         0.563339              0.832302        0.446087
710         0.353754              0.765979        0.333728
711         0.503594              0.822337        0.611473
712         0.253115              0.406873        0.214075
713         0.238978              0.450859        0.377489

     fractal_dimension_worst
0                   0.418864
1                   0.222878
2                   0.213433
3                   0.773711
4                   0.142595
..                       …
709                 0.299488
710                 0.420176
711                 0.291355
712                 0.124164
713                 0.138725

[714 rows x 30 columns]
```

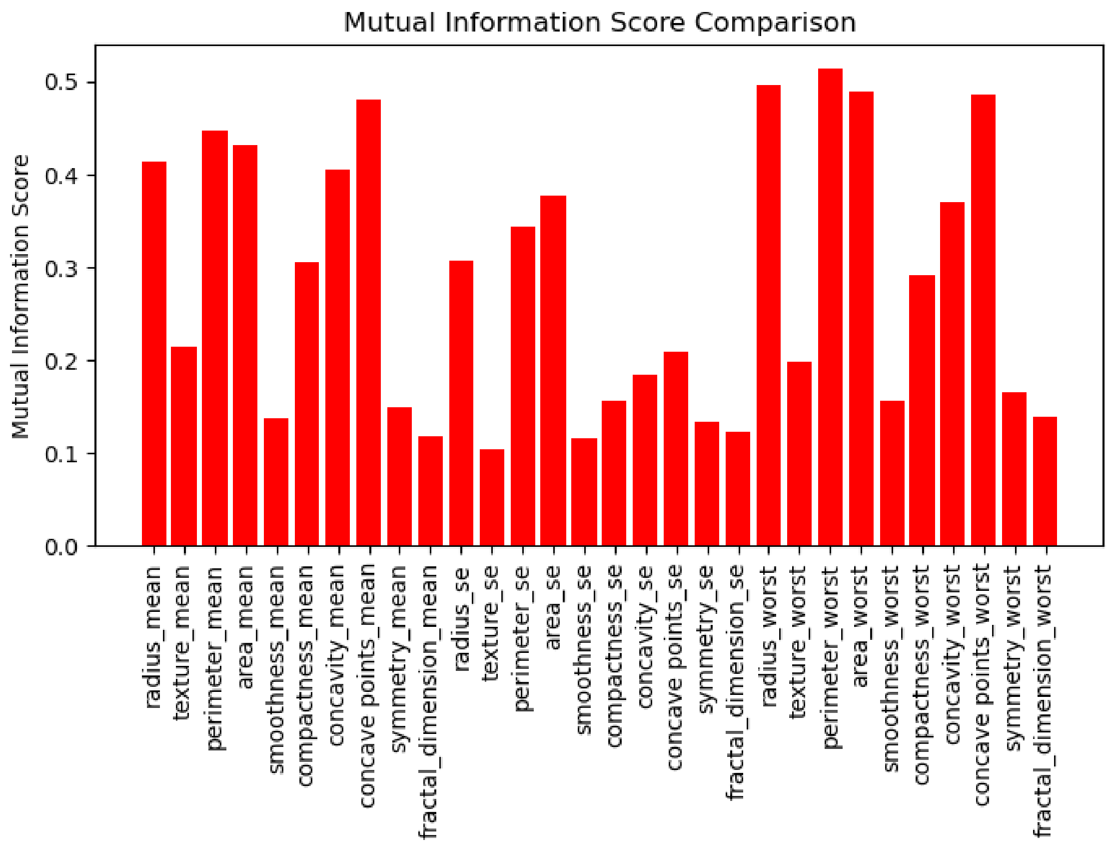
[176]:
```
y_oversampling
```

[176]:
```
0      1
1      1
2      1
3      1
4      1
      ..
709    1
710    1
711    1
712    1
713    1
Name: diagnosis, Length: 714, dtype: int64
```

[177]:
```
# information gain
MI_score = mutual_info_classif(X_oversampling,  y_oversampling , random_state=0)
for feature in zip(feature_columns, MI_score):
    if feature[1]>0.30:
        print(feature)
```

```
('radius_mean', 0.4133085495567619)
('perimeter_mean', 0.4473629655717368)
('area_mean', 0.43171828371890686)
('compactness_mean', 0.305059643908004)
('concavity_mean', 0.40517789650428226)
('concave points_mean', 0.480230407000652)
('radius_se', 0.30735060999231867)
('perimeter_se', 0.3438191837108908)
('area_se', 0.37763026802866495)
('radius_worst', 0.49623541498187906)
('perimeter_worst', 0.5139462061728981)
('area_worst', 0.48930203610532796)
('concavity_worst', 0.36912822036724213)
('concave points_worst', 0.48558750895509517)
```

[178]:
```
plt.figure(figsize=(8,4))
plt.bar(x=feature_columns, height=MI_score, color='red')
plt.xticks(rotation='vertical')
plt.ylabel('Mutual Information Score')
plt.title('Mutual Information Score Comparison')
plt.show()
```



[179]:
```
selected_features = [feature_columns[i] for i in range(len(feature_columns)) if␣
↪MI_score[i] > 0.3 ]
X_selected_over = X_oversampling[selected_features]
```

[180]:
```
X_selected_over
```

[180]:
```
     radius_mean  perimeter_mean  area_mean  compactness_mean  concavity_mean  \
0       0.521037        0.545989   0.363733          0.792037        0.703140
1       0.643144        0.615783   0.501591          0.181768        0.203608
2       0.601496        0.595743   0.449417          0.431017        0.462512
3       0.210090        0.233501   0.102906          0.811361        0.565604
4       0.629893        0.630986   0.489290          0.347893        0.463918
..           …             …          …               …               …
709     0.692366        0.695253   0.535949          0.580701        0.658388
710     0.394671        0.410545   0.241697          0.641126        0.573571
711     0.395617        0.405708   0.237922          0.595424        0.486645
712     0.313739        0.305853   0.186299          0.201613        0.202085
713     0.434427        0.431276   0.282630          0.334397        0.244377

     concave points_mean  radius_se  perimeter_se   area_se  radius_worst  \
```

```
0              0.731113  0.356147     0.369034  0.273811     0.620776
1              0.348757  0.156437     0.124440  0.125660     0.606901
2              0.635686  0.229622     0.180370  0.162922     0.556386
3              0.522863  0.139091     0.126655  0.038155     0.248310
4              0.518390  0.233822     0.220563  0.163688     0.519744
..                  …         …            …         …            …
709            0.776342  0.185660     0.160251  0.138566     0.651014
710            0.617296  0.198334     0.155680  0.098353     0.348630
711            0.484891  0.118523     0.123781  0.071177     0.360726
712            0.223111  0.124932     0.099138  0.067871     0.322305
713            0.278976  0.116495     0.098337  0.068880     0.410530

     perimeter_worst  area_worst  concavity_worst  concave points_worst
0           0.668310    0.450698         0.568610              0.912027
1           0.539818    0.435214         0.192971              0.639175
2           0.508442    0.374508         0.359744              0.835052
3           0.241347    0.094008         0.548642              0.884880
4           0.506948    0.341575         0.319489              0.558419
..               …           …                …                     …
709         0.605558    0.465936         0.563339              0.832302
710         0.345585    0.182757         0.353754              0.765979
711         0.371981    0.195561         0.503594              0.822337
712         0.289805    0.177276         0.253115              0.406873
713         0.394890    0.243266         0.238978              0.450859

[714 rows x 14 columns]
```

[181]:
```
y_oversampling
```

[181]:
```
0      1
1      1
2      1
3      1
4      1
      ..
709    1
710    1
711    1
712    1
713    1
Name: diagnosis, Length: 714, dtype: int64
```

[182]:
```
X_train, X_test, y_train, y_test = train_test_split(X_selected_over,␣
↪y_oversampling, test_size=0.2, random_state=0)
```

[183]:
```
knn(X_train, X_test, y_train, y_test)
```

```
Accuracy du KNN :  0.9370629370629371
```

```
precision score du KNN :  0.9375
recall score du KNN :  0.9230769230769231
f1 score du KNN :  0.9302325581395349
```

[184]:
```
knn_grid_search(X_selected_over,y_oversampling)
```

```
grid best score accuracy 0.9706964006259782
grid best score precision 0.9677298249819597
grid best score recall 0.9888888888888889
grid best score f1 score 0.9714095547561736
{'metric': 'manhattan', 'n_neighbors': 7, 'weights': 'distance'}
KNeighborsClassifier(metric='manhattan', n_neighbors=7, weights='distance')
```

3.2.1 SVM

[185]:
```
svm(X_train, X_test, y_train, y_test)
```

```
              precision    recall  f1-score   support

           0       0.96      0.96      0.96        78
           1       0.95      0.95      0.95        65

    accuracy                           0.96       143
   macro avg       0.96      0.96      0.96       143
weighted avg       0.96      0.96      0.96       143
```

[186]:
```
svm_grid_search(X_train, X_test, y_train, y_test)
```

```
              precision    recall  f1-score   support

           0       0.97      0.97      0.97        78
           1       0.97      0.97      0.97        65

    accuracy                           0.97       143
   macro avg       0.97      0.97      0.97       143
weighted avg       0.97      0.97      0.97       143

Accuracy :  0.972027972027972
```

4 Correlation

[187]:
```
X_oversampling
```

[187]:
```
   radius_mean  texture_mean  perimeter_mean  area_mean  smoothness_mean  \
0     0.521037      0.022658        0.545989   0.363733         0.593753
1     0.643144      0.272574        0.615783   0.501591         0.289880
2     0.601496      0.390260        0.595743   0.449417         0.514309
3     0.210090      0.360839        0.233501   0.102906         0.811321
```

```
4       0.629893      0.156578        0.630986   0.489290         0.430351
..            …            …              …        …                …
709     0.692366      0.425093        0.695253   0.535949         0.578406
710     0.394671      0.255665        0.410545   0.241697         0.730071
711     0.395617      0.153872        0.405708   0.237922         0.493545
712     0.313739      0.516402        0.305853   0.186299         0.381421
713     0.434427      0.400068        0.431276   0.282630         0.434865

     compactness_mean  concavity_mean  concave points_mean  symmetry_mean  \
0            0.792037        0.703140             0.731113       0.686364
1            0.181768        0.203608             0.348757       0.379798
2            0.431017        0.462512             0.635686       0.509596
3            0.811361        0.565604             0.522863       0.776263
4            0.347893        0.463918             0.518390       0.378283
..                …              …                    …             …
709          0.580701        0.658388             0.776342       0.556566
710          0.641126        0.573571             0.617296       0.675758
711          0.595424        0.486645             0.484891       0.737879
712          0.201613        0.202085             0.223111       0.277273
713          0.334397        0.244377             0.278976       0.555556

     fractal_dimension_mean  …  radius_worst  texture_worst  \
0                  0.605518  …      0.620776       0.141525
1                  0.141323  …      0.606901       0.303571
2                  0.211247  …      0.556386       0.360075
3                  1.000000  …      0.248310       0.385928
4                  0.186816  …      0.519744       0.123934
..                      …  …           …              …
709                0.339090  …      0.651014       0.445629
710                0.547599  …      0.348630       0.283582
711                0.428812  …      0.360726       0.188166
712                0.184288  …      0.322305       0.619670
713                0.188500  …      0.410530       0.523987

     perimeter_worst  area_worst  smoothness_worst  compactness_worst  \
0           0.668310    0.450698          0.601136           0.619292
1           0.539818    0.435214          0.347553           0.154563
2           0.508442    0.374508          0.483590           0.385375
3           0.241347    0.094008          0.915472           0.814012
4           0.506948    0.341575          0.437364           0.172415
..               …           …               …                  …
709         0.605558    0.465936          0.521891           0.528189
710         0.345585    0.182757          0.695569           0.410406
711         0.371981    0.195561          0.447930           0.551183
712         0.289805    0.177276          0.365383           0.162034
713         0.394890    0.243266          0.451232           0.269921
```

```
     concavity_worst  concave points_worst  symmetry_worst  \
0           0.568610              0.912027        0.598462
1           0.192971              0.639175        0.233590
2           0.359744              0.835052        0.403706
3           0.548642              0.884880        1.000000
4           0.319489              0.558419        0.157500
..               …                     …             …
709         0.563339              0.832302        0.446087
710         0.353754              0.765979        0.333728
711         0.503594              0.822337        0.611473
712         0.253115              0.406873        0.214075
713         0.238978              0.450859        0.377489

     fractal_dimension_worst
0                   0.418864
1                   0.222878
2                   0.213433
3                   0.773711
4                   0.142595
..                       …
709                 0.299488
710                 0.420176
711                 0.291355
712                 0.124164
713                 0.138725

[714 rows x 30 columns]
```

[188]:
```
y_oversampling
```

[188]:
```
0      1
1      1
2      1
3      1
4      1
      ..
709    1
710    1
711    1
712    1
713    1
Name: diagnosis, Length: 714, dtype: int64
```

[190]:
```
data_over = pd.DataFrame(X_oversampling , columns = X_oversampling.columns)
```

[191]:
```
data_over
```

[191]:

```
     radius_mean  texture_mean  perimeter_mean  area_mean  smoothness_mean  \
0       0.521037      0.022658        0.545989   0.363733         0.593753
1       0.643144      0.272574        0.615783   0.501591         0.289880
2       0.601496      0.390260        0.595743   0.449417         0.514309
3       0.210090      0.360839        0.233501   0.102906         0.811321
4       0.629893      0.156578        0.630986   0.489290         0.430351
..           …             …               …          …               …
709     0.692366      0.425093        0.695253   0.535949         0.578406
710     0.394671      0.255665        0.410545   0.241697         0.730071
711     0.395617      0.153872        0.405708   0.237922         0.493545
712     0.313739      0.516402        0.305853   0.186299         0.381421
713     0.434427      0.400068        0.431276   0.282630         0.434865

     compactness_mean  concavity_mean  concave points_mean  symmetry_mean  \
0            0.792037        0.703140             0.731113       0.686364
1            0.181768        0.203608             0.348757       0.379798
2            0.431017        0.462512             0.635686       0.509596
3            0.811361        0.565604             0.522863       0.776263
4            0.347893        0.463918             0.518390       0.378283
..                …              …                    …              …
709          0.580701        0.658388             0.776342       0.556566
710          0.641126        0.573571             0.617296       0.675758
711          0.595424        0.486645             0.484891       0.737879
712          0.201613        0.202085             0.223111       0.277273
713          0.334397        0.244377             0.278976       0.555556

     fractal_dimension_mean  …  radius_worst  texture_worst  \
0                  0.605518  …      0.620776       0.141525
1                  0.141323  …      0.606901       0.303571
2                  0.211247  …      0.556386       0.360075
3                  1.000000  …      0.248310       0.385928
4                  0.186816  …      0.519744       0.123934
..                      …   …           …             …
709                0.339090  …      0.651014       0.445629
710                0.547599  …      0.348630       0.283582
711                0.428812  …      0.360726       0.188166
712                0.184288  …      0.322305       0.619670
713                0.188500  …      0.410530       0.523987

     perimeter_worst  area_worst  smoothness_worst  compactness_worst  \
0           0.668310    0.450698          0.601136           0.619292
1           0.539818    0.435214          0.347553           0.154563
2           0.508442    0.374508          0.483590           0.385375
3           0.241347    0.094008          0.915472           0.814012
4           0.506948    0.341575          0.437364           0.172415
..               …           …                …                  …
709         0.605558    0.465936          0.521891           0.528189
```

```
710         0.345585     0.182757           0.695569              0.410406
711         0.371981     0.195561           0.447930              0.551183
712         0.289805     0.177276           0.365383              0.162034
713         0.394890     0.243266           0.451232              0.269921

     concavity_worst  concave points_worst  symmetry_worst  \
0           0.568610              0.912027        0.598462
1           0.192971              0.639175        0.233590
2           0.359744              0.835052        0.403706
3           0.548642              0.884880        1.000000
4           0.319489              0.558419        0.157500
..               …                     …             …
709         0.563339              0.832302        0.446087
710         0.353754              0.765979        0.333728
711         0.503594              0.822337        0.611473
712         0.253115              0.406873        0.214075
713         0.238978              0.450859        0.377489

     fractal_dimension_worst
0                   0.418864
1                   0.222878
2                   0.213433
3                   0.773711
4                   0.142595
..                       …
709                 0.299488
710                 0.420176
711                 0.291355
712                 0.124164
713                 0.138725

[714 rows x 30 columns]
```

[192]:
```
data_over['diagnosis'] = y_oversampling
```

[193]:
```
data_over
```

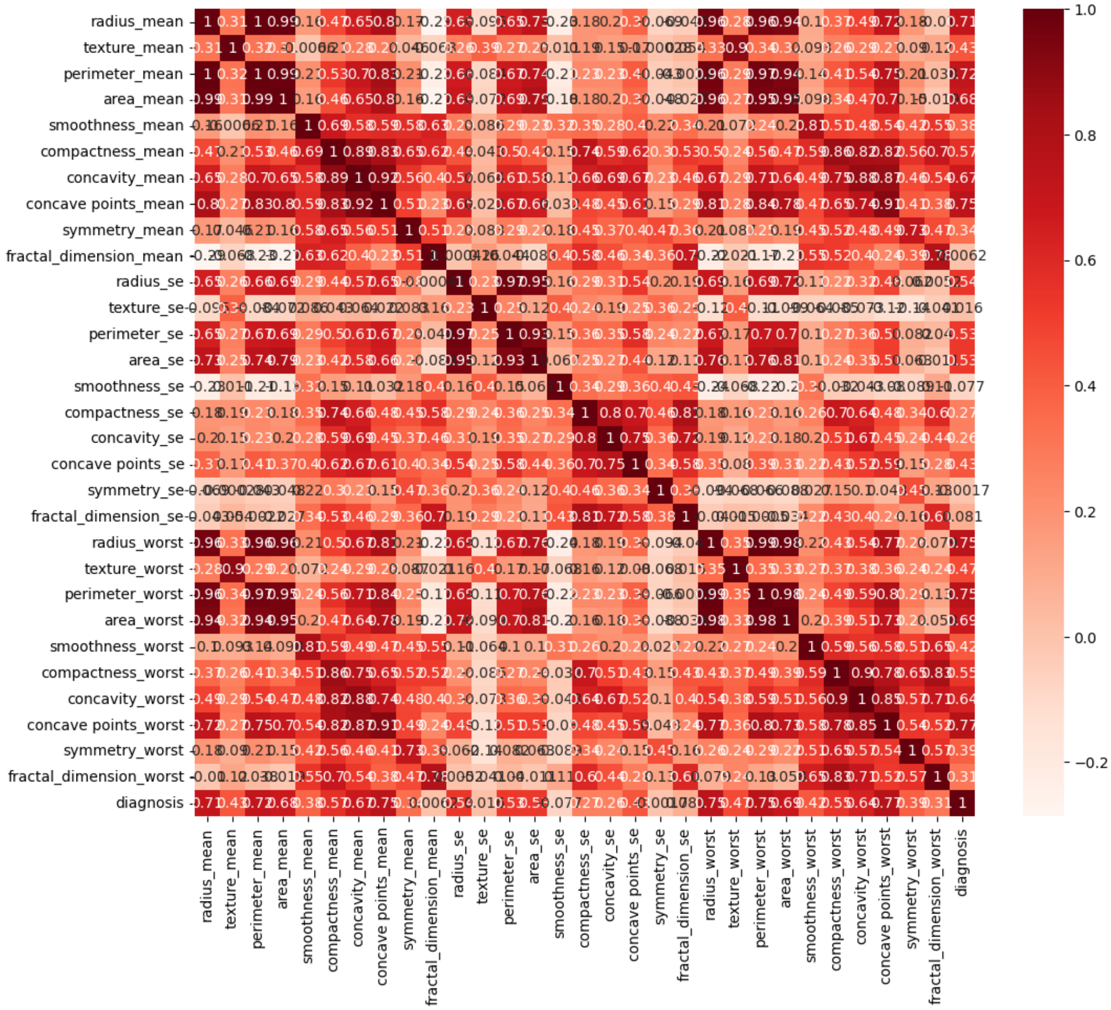
[193]:
```
     radius_mean  texture_mean  perimeter_mean  area_mean  smoothness_mean  \
0       0.521037      0.022658        0.545989   0.363733         0.593753
1       0.643144      0.272574        0.615783   0.501591         0.289880
2       0.601496      0.390260        0.595743   0.449417         0.514309
3       0.210090      0.360839        0.233501   0.102906         0.811321
4       0.629893      0.156578        0.630986   0.489290         0.430351
..           …             …              …          …                …
709     0.692366      0.425093        0.695253   0.535949         0.578406
710     0.394671      0.255665        0.410545   0.241697         0.730071
711     0.395617      0.153872        0.405708   0.237922         0.493545
```

```
712     0.313739      0.516402        0.305853   0.186299         0.381421
713     0.434427      0.400068        0.431276   0.282630         0.434865

     compactness_mean  concavity_mean  concave points_mean  symmetry_mean  \
0            0.792037        0.703140             0.731113       0.686364
1            0.181768        0.203608             0.348757       0.379798
2            0.431017        0.462512             0.635686       0.509596
3            0.811361        0.565604             0.522863       0.776263
4            0.347893        0.463918             0.518390       0.378283
..                …               …                    …              …
709          0.580701        0.658388             0.776342       0.556566
710          0.641126        0.573571             0.617296       0.675758
711          0.595424        0.486645             0.484891       0.737879
712          0.201613        0.202085             0.223111       0.277273
713          0.334397        0.244377             0.278976       0.555556

     fractal_dimension_mean  …  texture_worst  perimeter_worst  area_worst  \
0                  0.605518  …       0.141525         0.668310    0.450698
1                  0.141323  …       0.303571         0.539818    0.435214
2                  0.211247  …       0.360075         0.508442    0.374508
3                  1.000000  …       0.385928         0.241347    0.094008
4                  0.186816  …       0.123934         0.506948    0.341575
..                      …   …            …               …           …
709                0.339090  …       0.445629         0.605558    0.465936
710                0.547599  …       0.283582         0.345585    0.182757
711                0.428812  …       0.188166         0.371981    0.195561
712                0.184288  …       0.619670         0.289805    0.177276
713                0.188500  …       0.523987         0.394890    0.243266

     smoothness_worst  compactness_worst  concavity_worst  \
0            0.601136           0.619292         0.568610
1            0.347553           0.154563         0.192971
2            0.483590           0.385375         0.359744
3            0.915472           0.814012         0.548642
4            0.437364           0.172415         0.319489
..                …                  …                …
709          0.521891           0.528189         0.563339
710          0.695569           0.410406         0.353754
711          0.447930           0.551183         0.503594
712          0.365383           0.162034         0.253115
713          0.451232           0.269921         0.238978

     concave points_worst  symmetry_worst  fractal_dimension_worst  diagnosis
0                0.912027        0.598462                 0.418864          1
1                0.639175        0.233590                 0.222878          1
2                0.835052        0.403706                 0.213433          1
3                0.884880        1.000000                 0.773711          1
```

```
4          0.558419      0.157500            0.142595         1
..              ...           ...                 ...       ...
709        0.832302      0.446087            0.299488         1
710        0.765979      0.333728            0.420176         1
711        0.822337      0.611473            0.291355         1
712        0.406873      0.214075            0.124164         1
713        0.450859      0.377489            0.138725         1

[714 rows x 31 columns]
```

[194]:

```python
#Using Pearson Correlation
plt.figure(figsize=(12,10))
cor = data_over.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.Reds)
plt.show()
```



[195]:
```
#Correlation with output variable
cor_target = abs(cor["diagnosis"])
#Selecting highly correlated features
relevant_features = list(cor_target[abs(cor_target) > 0.5].index)
relevant_features.remove('diagnosis')
relevant_features
# we choose the variables that have a correlation higher to 0.5 with our target␣
↪variable
```

[195]:
```
['radius_mean',
 'perimeter_mean',
 'area_mean',
 'compactness_mean',
 'concavity_mean',
 'concave points_mean',
 'radius_se',
 'perimeter_se',
 'area_se',
 'radius_worst',
 'perimeter_worst',
 'area_worst',
 'compactness_worst',
 'concavity_worst',
 'concave points_worst']
```

[196]:
```
data_corr_over = data_over[relevant_features]
data_corr_over
```

[196]:
```
     radius_mean  perimeter_mean  area_mean  compactness_mean  concavity_mean  \
0       0.521037        0.545989   0.363733          0.792037        0.703140
1       0.643144        0.615783   0.501591          0.181768        0.203608
2       0.601496        0.595743   0.449417          0.431017        0.462512
3       0.210090        0.233501   0.102906          0.811361        0.565604
4       0.629893        0.630986   0.489290          0.347893        0.463918
..           …               …          …                 …              …
709     0.692366        0.695253   0.535949          0.580701        0.658388
710     0.394671        0.410545   0.241697          0.641126        0.573571
711     0.395617        0.405708   0.237922          0.595424        0.486645
712     0.313739        0.305853   0.186299          0.201613        0.202085
713     0.434427        0.431276   0.282630          0.334397        0.244377

     concave points_mean  radius_se  perimeter_se   area_se  radius_worst  \
0               0.731113   0.356147      0.369034  0.273811      0.620776
1               0.348757   0.156437      0.124440  0.125660      0.606901
2               0.635686   0.229622      0.180370  0.162922      0.556386
3               0.522863   0.139091      0.126655  0.038155      0.248310
4               0.518390   0.233822      0.220563  0.163688      0.519744
```

```
..                 …        …           …        …           …
709         0.776342  0.185660     0.160251  0.138566    0.651014
710         0.617296  0.198334     0.155680  0.098353    0.348630
711         0.484891  0.118523     0.123781  0.071177    0.360726
712         0.223111  0.124932     0.099138  0.067871    0.322305
713         0.278976  0.116495     0.098337  0.068880    0.410530

     perimeter_worst  area_worst  compactness_worst  concavity_worst  \
0           0.668310    0.450698           0.619292         0.568610
1           0.539818    0.435214           0.154563         0.192971
2           0.508442    0.374508           0.385375         0.359744
3           0.241347    0.094008           0.814012         0.548642
4           0.506948    0.341575           0.172415         0.319489
..               …          …                 …               …
709         0.605558    0.465936           0.528189         0.563339
710         0.345585    0.182757           0.410406         0.353754
711         0.371981    0.195561           0.551183         0.503594
712         0.289805    0.177276           0.162034         0.253115
713         0.394890    0.243266           0.269921         0.238978

     concave points_worst
0                0.912027
1                0.639175
2                0.835052
3                0.884880
4                0.558419
..                    …
709              0.832302
710              0.765979
711              0.822337
712              0.406873
713              0.450859

[714 rows x 15 columns]
```

[197]:
```python
X_over_corr = X_oversampling[relevant_features]
X_over_corr
```

[197]:
```
     radius_mean  perimeter_mean  area_mean  compactness_mean  concavity_mean  \
0       0.521037        0.545989   0.363733          0.792037        0.703140
1       0.643144        0.615783   0.501591          0.181768        0.203608
2       0.601496        0.595743   0.449417          0.431017        0.462512
3       0.210090        0.233501   0.102906          0.811361        0.565604
4       0.629893        0.630986   0.489290          0.347893        0.463918
..           …              …          …                …               …
709     0.692366        0.695253   0.535949          0.580701        0.658388
710     0.394671        0.410545   0.241697          0.641126        0.573571
```

711	0.395617	0.405708	0.237922	0.595424	0.486645
712	0.313739	0.305853	0.186299	0.201613	0.202085
713	0.434427	0.431276	0.282630	0.334397	0.244377

	concave points_mean	radius_se	perimeter_se	area_se	radius_worst	\
0	0.731113	0.356147	0.369034	0.273811	0.620776	
1	0.348757	0.156437	0.124440	0.125660	0.606901	
2	0.635686	0.229622	0.180370	0.162922	0.556386	
3	0.522863	0.139091	0.126655	0.038155	0.248310	
4	0.518390	0.233822	0.220563	0.163688	0.519744	
..	…	…	…	…	…	
709	0.776342	0.185660	0.160251	0.138566	0.651014	
710	0.617296	0.198334	0.155680	0.098353	0.348630	
711	0.484891	0.118523	0.123781	0.071177	0.360726	
712	0.223111	0.124932	0.099138	0.067871	0.322305	
713	0.278976	0.116495	0.098337	0.068880	0.410530	

	perimeter_worst	area_worst	compactness_worst	concavity_worst	\
0	0.668310	0.450698	0.619292	0.568610	
1	0.539818	0.435214	0.154563	0.192971	
2	0.508442	0.374508	0.385375	0.359744	
3	0.241347	0.094008	0.814012	0.548642	
4	0.506948	0.341575	0.172415	0.319489	
..	…	…	…	…	
709	0.605558	0.465936	0.528189	0.563339	
710	0.345585	0.182757	0.410406	0.353754	
711	0.371981	0.195561	0.551183	0.503594	
712	0.289805	0.177276	0.162034	0.253115	
713	0.394890	0.243266	0.269921	0.238978	

	concave points_worst
0	0.912027
1	0.639175
2	0.835052
3	0.884880
4	0.558419
..	…
709	0.832302
710	0.765979
711	0.822337
712	0.406873
713	0.450859

[714 rows x 15 columns]

4.1 KNN based on correlation

[198]:
```
# splitting the data
X_train, X_test, y_train, y_test = train_test_split(X_over_corr,␣
↪y_oversampling, test_size=0.2, random_state=0)
```

[199]:
```
knn(X_train, X_test, y_train, y_test)
```

```
Accuracy du KNN :  0.951048951048951
precision score du KNN :  0.9393939393939394
recall score du KNN :  0.9538461538461539
f1 score du KNN :  0.9465648854961831
```

[200]:
```
knn_grid_search(X_over_corr, y_oversampling)
```

```
grid best score accuracy 0.9692292644757433
grid best score precision 0.9686692312059959
grid best score recall 0.9888888888888889
grid best score f1 score 0.9697774346105428
{'metric': 'manhattan', 'n_neighbors': 1, 'weights': 'uniform'}
KNeighborsClassifier(metric='manhattan', n_neighbors=1)
```

4.2 SVM BASED ON CORRELATION

[201]:
```
svm(X_train,X_test,y_train ,y_test)
```

```
              precision    recall  f1-score   support

           0       0.96      0.96      0.96        78
           1       0.95      0.95      0.95        65

    accuracy                           0.96       143
   macro avg       0.96      0.96      0.96       143
weighted avg       0.96      0.96      0.96       143
```

[202]:
```
svm_grid_search(X_train,X_test,y_train ,y_test)
```

```
              precision    recall  f1-score   support

           0       0.97      0.97      0.97        78
           1       0.97      0.97      0.97        65

    accuracy                           0.97       143
   macro avg       0.97      0.97      0.97       143
weighted avg       0.97      0.97      0.97       143

Accuracy :  0.972027972027972
```

[208]:
```
X_oversampling
```

[208]:

```
     radius_mean  texture_mean  perimeter_mean  area_mean  smoothness_mean  \
0       0.521037      0.022658        0.545989   0.363733         0.593753
1       0.643144      0.272574        0.615783   0.501591         0.289880
2       0.601496      0.390260        0.595743   0.449417         0.514309
3       0.210090      0.360839        0.233501   0.102906         0.811321
4       0.629893      0.156578        0.630986   0.489290         0.430351
..           …             …               …          …              …
709     0.692366      0.425093        0.695253   0.535949         0.578406
710     0.394671      0.255665        0.410545   0.241697         0.730071
711     0.395617      0.153872        0.405708   0.237922         0.493545
712     0.313739      0.516402        0.305853   0.186299         0.381421
713     0.434427      0.400068        0.431276   0.282630         0.434865

     compactness_mean  concavity_mean  concave points_mean  symmetry_mean  \
0            0.792037        0.703140             0.731113       0.686364
1            0.181768        0.203608             0.348757       0.379798
2            0.431017        0.462512             0.635686       0.509596
3            0.811361        0.565604             0.522863       0.776263
4            0.347893        0.463918             0.518390       0.378283
..                …              …                    …              …
709          0.580701        0.658388             0.776342       0.556566
710          0.641126        0.573571             0.617296       0.675758
711          0.595424        0.486645             0.484891       0.737879
712          0.201613        0.202085             0.223111       0.277273
713          0.334397        0.244377             0.278976       0.555556

     fractal_dimension_mean  …  texture_worst  perimeter_worst  area_worst  \
0                  0.605518  …       0.141525         0.668310    0.450698
1                  0.141323  …       0.303571         0.539818    0.435214
2                  0.211247  …       0.360075         0.508442    0.374508
3                  1.000000  …       0.385928         0.241347    0.094008
4                  0.186816  …       0.123934         0.506948    0.341575
..                      …  …            …                …           …
709                0.339090  …       0.445629         0.605558    0.465936
710                0.547599  …       0.283582         0.345585    0.182757
711                0.428812  …       0.188166         0.371981    0.195561
712                0.184288  …       0.619670         0.289805    0.177276
713                0.188500  …       0.523987         0.394890    0.243266

     smoothness_worst  compactness_worst  concavity_worst  \
0            0.601136           0.619292         0.568610
1            0.347553           0.154563         0.192971
2            0.483590           0.385375         0.359744
3            0.915472           0.814012         0.548642
4            0.437364           0.172415         0.319489
..                …                  …                …
709          0.521891           0.528189         0.563339
```

```
710          0.695569          0.410406          0.353754
711          0.447930          0.551183          0.503594
712          0.365383          0.162034          0.253115
713          0.451232          0.269921          0.238978

     concave points_worst  symmetry_worst  fractal_dimension_worst  diagnosis
0                0.912027        0.598462                 0.418864          1
1                0.639175        0.233590                 0.222878          1
2                0.835052        0.403706                 0.213433          1
3                0.884880        1.000000                 0.773711          1
4                0.558419        0.157500                 0.142595          1
..                    …             …                       …        …
709              0.832302        0.446087                 0.299488          1
710              0.765979        0.333728                 0.420176          1
711              0.822337        0.611473                 0.291355          1
712              0.406873        0.214075                 0.124164          1
713              0.450859        0.377489                 0.138725          1

[714 rows x 31 columns]
```

[209]:
```
X_oversampling.drop('diagnosis',axis = 1,inplace = True)
```

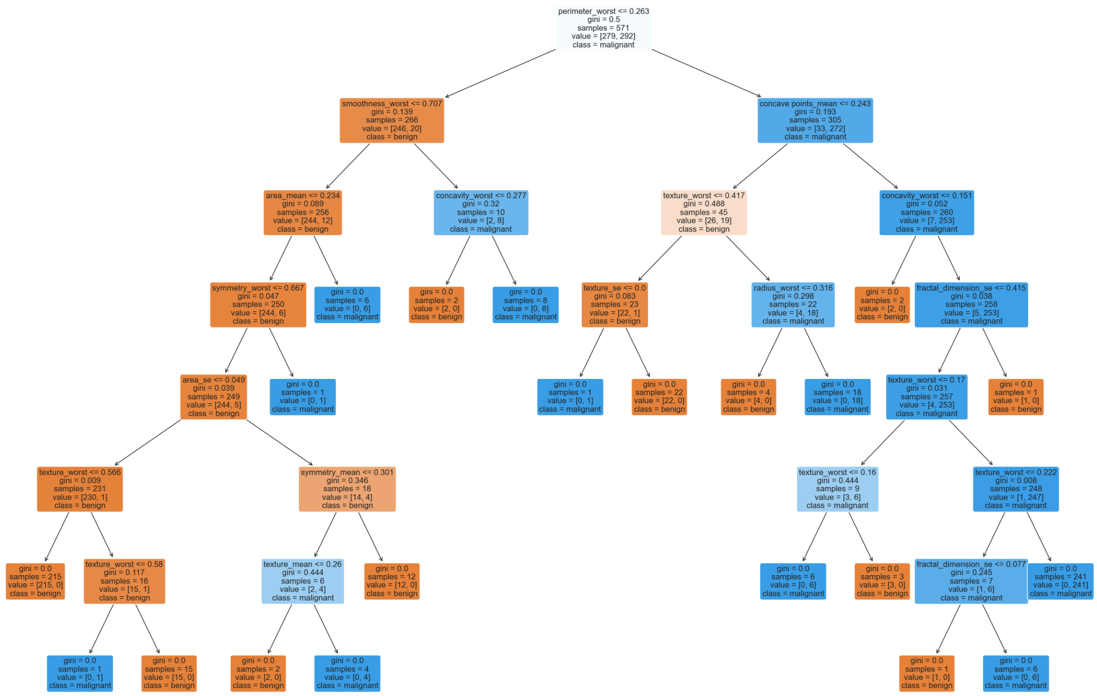
5 DT

[124]:
```
X_train, X_test, y_train, y_test = train_test_split(X_oversampling,␣
 ↪y_oversampling, test_size=0.2, random_state=0)
clf = tree.DecisionTreeClassifier()
clf.fit(X_train, y_train)
y_pred= clf.predict(X_test)
acc2 = metrics.accuracy_score(y_test, y_pred)
precision =  metrics.precision_score(y_test, y_pred)
recall = metrics.recall_score(y_test, y_pred)
f1_score = metrics.f1_score(y_test , y_pred)
print("Accuracy:", acc2)
print("Precision: ",precision)
print("Recall: ",recall)
print("F1 score : ",f1_score)
```

```
Accuracy: 0.951048951048951
Precision:  0.9393939393939394
Recall:  0.9538461538461539
F1 score :  0.9465648854961831
```

[211]:
```
clf = tree.DecisionTreeClassifier()
clf.fit(X_train, y_train)
sns.set(font_scale=100, style="whitegrid", rc={"axes.grid": False})
fig, ax = plt.subplots(figsize=(30, 20))
```

```
tree.plot_tree(clf, feature_names=X_oversampling.columns,␣
↪class_names=['benign', 'malignant'], filled=True, rounded=True, fontsize=12,␣
↪ax=ax)
plt.show()
```



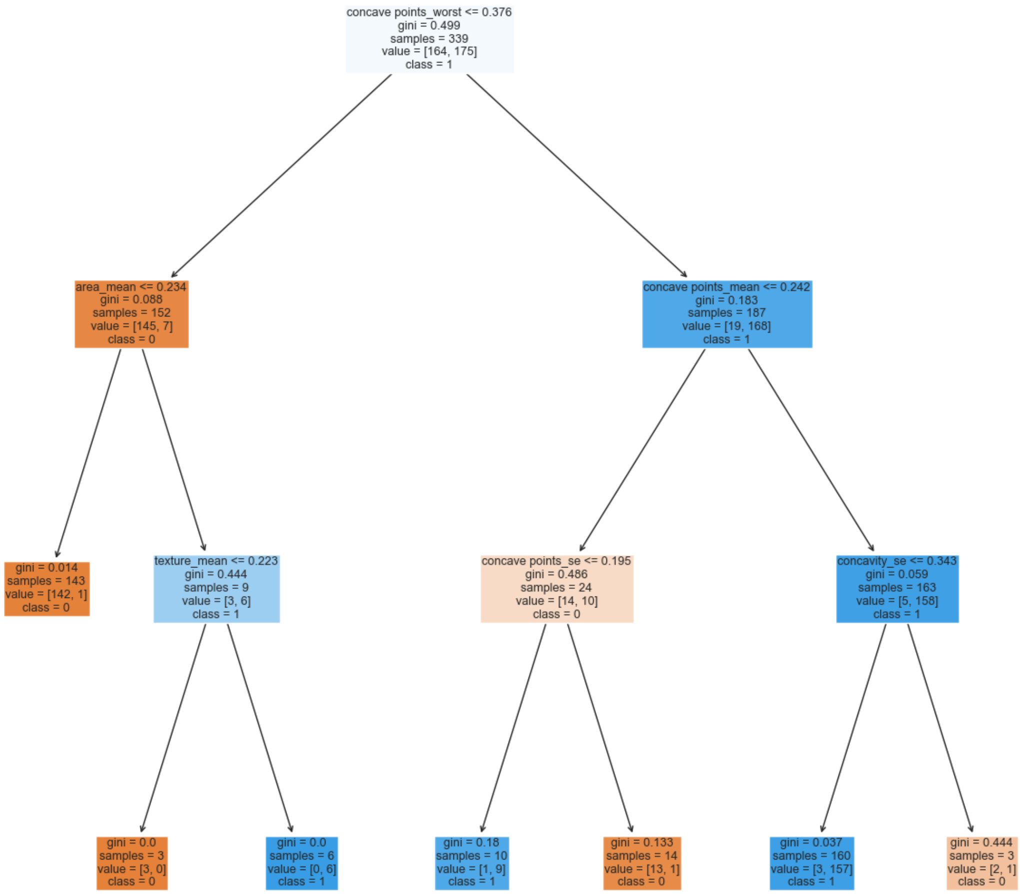
[133]:

```
#Grid search
DT = tree.DecisionTreeClassifier()
params = {
 'criterion': ['gini', 'entropy'],
 'max_depth': [2, 4, 6, 8, 10],
 'min_samples_split': [2, 4, 6, 8, 10],
 'min_samples_leaf': [1, 2, 3, 4, 5]
    }
grid = GridSearchCV(DT, params, cv = 10, scoring = 'accuracy')
grid.fit(X_oversampling,y_oversampling)

grid1 = GridSearchCV(DT, params, cv = 10, scoring = 'precision')
grid1.fit(X_oversampling,y_oversampling)

grid2 = GridSearchCV(DT, params, cv = 10, scoring = 'recall')
grid2.fit(X_oversampling,y_oversampling)

grid3 = GridSearchCV(DT, params, cv = 10, scoring = 'f1')
```

```
grid3.fit(X_oversampling,y_oversampling)

print("Accuracy",grid.best_score_)
print("Precision",grid1.best_score_)
print("Recall",grid2.best_score_)
print("f1 score",grid3.best_score_)

print(grid.best_params_)
print(grid.best_estimator_)
```

```
Accuracy 0.9650430359937403
Precision 0.9637876952582834
Recall 0.986111111111111
f1 score 0.966697748712026
{'criterion': 'entropy', 'max_depth': 6, 'min_samples_leaf': 5,
'min_samples_split': 6}
DecisionTreeClassifier(criterion='entropy', max_depth=6, min_samples_leaf=5,
                       min_samples_split=6)
```

[134]:
```
from sklearn import tree
sns.set(font_scale=2, style="whitegrid", rc={"axes.grid": False})
fig, ax = plt.subplots(figsize=(15, 15))
tree.plot_tree(best_clf, feature_names=feature_columns, class_names=["0", "1"],␣
 ↪filled=True, ax=ax)
plt.show()
```



6 Select Kbest

[216]:
```
selector = SelectKBest(chi2, k=10)
X_new = selector.fit_transform(X_oversampling, y_oversampling)
mask = selector.get_support()
selected_features =X_oversampling.columns[mask]
selected = selected_features.values
selected
```

[216]:
```
array(['radius_mean', 'perimeter_mean', 'area_mean', 'concavity_mean',
       'concave points_mean', 'radius_worst', 'perimeter_worst',
       'area_worst', 'concavity_worst', 'concave points_worst'],
```

```
       dtype=object)
```

[217]:
```
X_selected_kbest_oversampling = X_oversampling[selected]
X_selected_kbest_oversampling
```

[217]:
```
     radius_mean  perimeter_mean  area_mean  concavity_mean  \
0       0.521037        0.545989   0.363733        0.703140
1       0.643144        0.615783   0.501591        0.203608
2       0.601496        0.595743   0.449417        0.462512
3       0.210090        0.233501   0.102906        0.565604
4       0.629893        0.630986   0.489290        0.463918
..           …             …          …              …
709     0.692366        0.695253   0.535949        0.658388
710     0.394671        0.410545   0.241697        0.573571
711     0.395617        0.405708   0.237922        0.486645
712     0.313739        0.305853   0.186299        0.202085
713     0.434427        0.431276   0.282630        0.244377

     concave points_mean  radius_worst  perimeter_worst  area_worst  \
0               0.731113      0.620776         0.668310    0.450698
1               0.348757      0.606901         0.539818    0.435214
2               0.635686      0.556386         0.508442    0.374508
3               0.522863      0.248310         0.241347    0.094008
4               0.518390      0.519744         0.506948    0.341575
..                   …             …              …           …
709             0.776342      0.651014         0.605558    0.465936
710             0.617296      0.348630         0.345585    0.182757
711             0.484891      0.360726         0.371981    0.195561
712             0.223111      0.322305         0.289805    0.177276
713             0.278976      0.410530         0.394890    0.243266

     concavity_worst  concave points_worst
0           0.568610              0.912027
1           0.192971              0.639175
2           0.359744              0.835052
3           0.548642              0.884880
4           0.319489              0.558419
..               …                    …
709         0.563339              0.832302
710         0.353754              0.765979
711         0.503594              0.822337
712         0.253115              0.406873
713         0.238978              0.450859

[714 rows x 10 columns]
```

[218]:
```python
# splitting the data
X_train, X_test, y_train, y_test =␣
↪train_test_split(X_selected_kbest_oversampling , y_oversampling, test_size=0.
↪2, random_state=0)
```

[219]:
```python
knn(X_train, X_test, y_train, y_test)
```

```
Accuracy du KNN :  0.9440559440559441
precision score du KNN :  0.9384615384615385
recall score du KNN :  0.9384615384615385
f1 score du KNN :  0.9384615384615385
```

[220]:
```python
knn_grid_search(X_selected_kbest_oversampling, y_oversampling)
```

```
grid best score accuracy 0.9706377151799688
grid best score precision 0.9598742822272234
grid best score recall 0.9916666666666668
grid best score f1 score 0.9714854980390385
{'metric': 'euclidean', 'n_neighbors': 1, 'weights': 'uniform'}
KNeighborsClassifier(metric='euclidean', n_neighbors=1)
```

6.1 SVM

[221]:
```python
svm(X_train, X_test, y_train, y_test)
```

```
              precision    recall  f1-score   support

           0       0.96      0.95      0.95        78
           1       0.94      0.95      0.95        65

    accuracy                           0.95       143
   macro avg       0.95      0.95      0.95       143
weighted avg       0.95      0.95      0.95       143
```

[222]:
```python
svm_grid_search(X_train,X_test,y_train ,y_test)
```

```
              precision    recall  f1-score   support

           0       0.96      0.96      0.96        78
           1       0.95      0.95      0.95        65

    accuracy                           0.96       143
   macro avg       0.96      0.96      0.96       143
weighted avg       0.96      0.96      0.96       143

Accuracy :  0.958041958041958
```

6.2 Wrappers

[223]:
```
X_train, X_test, y_train, y_test = train_test_split(X_oversampling,␣
    ↪y_oversampling, test_size=0.33, random_state=42)
clf = RandomForestClassifier(n_estimators=100, random_state=0)
clf.fit(X_train, y_train)
# Calculate feature importances and select features with scores greater than 0.
    ↪01
feature_scores = pd.Series(clf.feature_importances_, index=X_train.columns).
    ↪sort_values(ascending=False)
selected = feature_scores[feature_scores.values > 0.01]
selected_features = selected.index.values
selected_features
```

[223]:
```
array(['perimeter_worst', 'radius_worst', 'concave points_worst',
       'perimeter_mean', 'area_worst', 'concave points_mean', 'area_se',
       'concavity_mean', 'area_mean', 'concavity_worst', 'radius_se',
       'radius_mean', 'perimeter_se', 'texture_worst',
       'compactness_worst', 'fractal_dimension_worst', 'smoothness_worst',
       'texture_mean'], dtype=object)
```

[224]:
```
X_selected_randomforest = X_oversampling[selected_features]
X_selected_randomforest
```

[224]:
```
     perimeter_worst  radius_worst  concave points_worst  perimeter_mean  \
0           0.668310      0.620776              0.912027        0.545989
1           0.539818      0.606901              0.639175        0.615783
2           0.508442      0.556386              0.835052        0.595743
3           0.241347      0.248310              0.884880        0.233501
4           0.506948      0.519744              0.558419        0.630986
..               …             …                     …               …
709         0.605558      0.651014              0.832302        0.695253
710         0.345585      0.348630              0.765979        0.410545
711         0.371981      0.360726              0.822337        0.405708
712         0.289805      0.322305              0.406873        0.305853
713         0.394890      0.410530              0.450859        0.431276

     area_worst  concave points_mean   area_se  concavity_mean  area_mean  \
0      0.450698             0.731113  0.273811        0.703140   0.363733
1      0.435214             0.348757  0.125660        0.203608   0.501591
2      0.374508             0.635686  0.162922        0.462512   0.449417
3      0.094008             0.522863  0.038155        0.565604   0.102906
4      0.341575             0.518390  0.163688        0.463918   0.489290
..          …                   …         …             …          …
709    0.465936             0.776342  0.138566        0.658388   0.535949
710    0.182757             0.617296  0.098353        0.573571   0.241697
711    0.195561             0.484891  0.071177        0.486645   0.237922
```

```
712    0.177276             0.223111  0.067871         0.202085   0.186299
713    0.243266             0.278976  0.068880         0.244377   0.282630

     concavity_worst  radius_se  radius_mean  perimeter_se  texture_worst  \
0           0.568610   0.356147     0.521037      0.369034       0.141525
1           0.192971   0.156437     0.643144      0.124440       0.303571
2           0.359744   0.229622     0.601496      0.180370       0.360075
3           0.548642   0.139091     0.210090      0.126655       0.385928
4           0.319489   0.233822     0.629893      0.220563       0.123934
..               …          …            …             …              …
709         0.563339   0.185660     0.692366      0.160251       0.445629
710         0.353754   0.198334     0.394671      0.155680       0.283582
711         0.503594   0.118523     0.395617      0.123781       0.188166
712         0.253115   0.124932     0.313739      0.099138       0.619670
713         0.238978   0.116495     0.434427      0.098337       0.523987

     compactness_worst  fractal_dimension_worst  smoothness_worst  \
0             0.619292                 0.418864          0.601136
1             0.154563                 0.222878          0.347553
2             0.385375                 0.213433          0.483590
3             0.814012                 0.773711          0.915472
4             0.172415                 0.142595          0.437364
..                 …                        …                 …
709           0.528189                 0.299488          0.521891
710           0.410406                 0.420176          0.695569
711           0.551183                 0.291355          0.447930
712           0.162034                 0.124164          0.365383
713           0.269921                 0.138725          0.451232

     texture_mean
0        0.022658
1        0.272574
2        0.390260
3        0.360839
4        0.156578
..            …
709      0.425093
710      0.255665
711      0.153872
712      0.516402
713      0.400068

[714 rows x 18 columns]
```

[225]:

```
# splitting the data
X_train, X_test, y_train, y_test = train_test_split(X_selected_randomforest,
 ↪y_oversampling, test_size=0.2, random_state=0)
```

6.3 KNN BASED ON RANDOM FOREST

[226]:
```
knn(X_train, X_test, y_train, y_test)
```

```
Accuracy du KNN :  0.951048951048951
precision score du KNN :  0.953125
recall score du KNN :  0.9384615384615385
f1 score du KNN :  0.9457364341085271
```

[227]:
```
knn_grid_search(X_selected_randomforest, y_oversampling)
```

```
grid best score accuracy 0.9846439749608763
grid best score precision 0.9833247533247531
grid best score recall 0.9888888888888889
grid best score f1 score 0.9846619289557742
{'metric': 'manhattan', 'n_neighbors': 10, 'weights': 'distance'}
KNeighborsClassifier(metric='manhattan', n_neighbors=10, weights='distance')
```

6.4 SVM

[228]:
```
svm(X_train, X_test, y_train, y_test)
```

```
              precision    recall  f1-score   support

           0       0.94      0.99      0.96        78
           1       0.98      0.92      0.95        65

    accuracy                           0.96       143
   macro avg       0.96      0.96      0.96       143
weighted avg       0.96      0.96      0.96       143
```

[229]:
```
svm_grid_search(X_train,X_test,y_train ,y_test)
```

```
              precision    recall  f1-score   support

           0       0.96      0.99      0.97        78
           1       0.98      0.95      0.97        65

    accuracy                           0.97       143
   macro avg       0.97      0.97      0.97       143
weighted avg       0.97      0.97      0.97       143

Accuracy :  0.972027972027972
```

6.5 Wrapper the Recursive Feature Elimination

[230]:
```python
from sklearn.feature_selection import RFE
from sklearn.svm import SVC

# Initialize an SVM classifier and an RFE feature selector
svm = SVC(kernel='linear')
rfe = RFE(estimator=svm, n_features_to_select=10, step=1)

# Fit the RFE selector to the data and get the selected feature indices
rfe.fit(X_oversampling, y_oversampling)
selected_indices = rfe.get_support(indices=True)

selected_names = [X_oversampling.columns[i] for i, selected in enumerate(rfe.
 ↪support_) if selected]

# Print the selected feature names
print('Selected features:', selected_names)
```

```
Selected features: ['radius_mean', 'concave points_mean', 'radius_se',
'perimeter_se', 'radius_worst', 'texture_worst', 'perimeter_worst',
'area_worst', 'smoothness_worst', 'concave points_worst']
```

[231]:
```python
X_selected_RFE = X_oversampling[selected_names]
X_selected_RFE
```

[231]:
```
     radius_mean  concave points_mean  radius_se  perimeter_se  radius_worst  \
0       0.521037             0.731113   0.356147      0.369034      0.620776
1       0.643144             0.348757   0.156437      0.124440      0.606901
2       0.601496             0.635686   0.229622      0.180370      0.556386
3       0.210090             0.522863   0.139091      0.126655      0.248310
4       0.629893             0.518390   0.233822      0.220563      0.519744
..           …                    …          …             …             …
709     0.692366             0.776342   0.185660      0.160251      0.651014
710     0.394671             0.617296   0.198334      0.155680      0.348630
711     0.395617             0.484891   0.118523      0.123781      0.360726
712     0.313739             0.223111   0.124932      0.099138      0.322305
713     0.434427             0.278976   0.116495      0.098337      0.410530

     texture_worst  perimeter_worst  area_worst  smoothness_worst  \
0         0.141525         0.668310    0.450698          0.601136
1         0.303571         0.539818    0.435214          0.347553
2         0.360075         0.508442    0.374508          0.483590
3         0.385928         0.241347    0.094008          0.915472
4         0.123934         0.506948    0.341575          0.437364
..             …                …           …                 …
709       0.445629         0.605558    0.465936          0.521891
710       0.283582         0.345585    0.182757          0.695569
```

```
711       0.188166          0.371981    0.195561            0.447930
712       0.619670          0.289805    0.177276            0.365383
713       0.523987          0.394890    0.243266            0.451232

     concave points_worst
0                0.912027
1                0.639175
2                0.835052
3                0.884880
4                0.558419
..                    …
709              0.832302
710              0.765979
711              0.822337
712              0.406873
713              0.450859

[714 rows x 10 columns]
```

[232]:

```
# splitting the data
X_train, X_test, y_train, y_test =␣
 ↪train_test_split(X_selected_RFE,y_oversampling, test_size=0.2,␣
 ↪random_state=0)
```

[233]:

```
knn(X_train, X_test, y_train, y_test)
```

```
Accuracy du KNN :  0.958041958041958
precision score du KNN :  0.9682539682539683
recall score du KNN :  0.9384615384615385
f1 score du KNN :  0.953125
```

[236]:

```
knn_grid_search(X_selected_RFE, y_oversampling)
```

```
grid best score accuracy 0.9846048513302035
grid best score precision 0.9857936507936508
grid best score recall 0.9916666666666668
grid best score f1 score 0.9847723735973783
{'metric': 'euclidean', 'n_neighbors': 17, 'weights': 'distance'}
KNeighborsClassifier(metric='euclidean', n_neighbors=17, weights='distance')
```

[237]:

```
def svm(X_train, X_test, y_train, y_test):
    svm = SVC()
    svm.fit(X_train, y_train)
    y_pred = svm.predict(X_test)
    print(classification_report(y_test, y_pred))
```

[238]:

```
svm(X_train, X_test, y_train, y_test)
```

```
              precision    recall  f1-score   support
```

```
           0       0.96      0.99      0.97        78
           1       0.98      0.95      0.97        65

    accuracy                           0.97       143
   macro avg       0.97      0.97      0.97       143
weighted avg       0.97      0.97      0.97       143
```

[239]:
```python
svm_grid_search(X_train,X_test,y_train ,y_test)
```

```
              precision    recall  f1-score   support

           0       0.96      0.99      0.97        78
           1       0.98      0.95      0.97        65

    accuracy                           0.97       143
   macro avg       0.97      0.97      0.97       143
weighted avg       0.97      0.97      0.97       143

Accuracy :  0.972027972027972
```

6.6 Comparaison

6.6.1 using data set before balancing

[73]:
```python
X
```

[73]:
```
     radius_mean  texture_mean  perimeter_mean  area_mean  smoothness_mean  \
0       0.521037      0.022658        0.545989   0.363733         0.593753
1       0.643144      0.272574        0.615783   0.501591         0.289880
2       0.601496      0.390260        0.595743   0.449417         0.514309
3       0.210090      0.360839        0.233501   0.102906         0.811321
4       0.629893      0.156578        0.630986   0.489290         0.430351
..           …             …               …          …                …
564     0.690000      0.428813        0.678668   0.566490         0.526948
565     0.622320      0.626987        0.604036   0.474019         0.407782
566     0.455251      0.621238        0.445788   0.303118         0.288165
567     0.644564      0.663510        0.665538   0.475716         0.588336
568     0.036869      0.501522        0.028540   0.015907         0.000000

     compactness_mean  concavity_mean  concave points_mean  symmetry_mean  \
0            0.792037        0.703140             0.731113       0.686364
1            0.181768        0.203608             0.348757       0.379798
2            0.431017        0.462512             0.635686       0.509596
3            0.811361        0.565604             0.522863       0.776263
4            0.347893        0.463918             0.518390       0.378283
..                …               …                    …              …
```

```
564         0.296055        0.571462            0.690358       0.336364
565         0.257714        0.337395            0.486630       0.349495
566         0.254340        0.216753            0.263519       0.267677
567         0.790197        0.823336            0.755467       0.675253
568         0.074351        0.000000            0.000000       0.266162

     fractal_dimension_mean  …  radius_worst  texture_worst  \
0                  0.605518  …      0.620776       0.141525
1                  0.141323  …      0.606901       0.303571
2                  0.211247  …      0.556386       0.360075
3                  1.000000  …      0.248310       0.385928
4                  0.186816  …      0.519744       0.123934
..                      …   …           …            …
564                0.132056  …      0.623266       0.383262
565                0.113100  …      0.560655       0.699094
566                0.137321  …      0.393099       0.589019
567                0.425442  …      0.633582       0.730277
568                0.187026  …      0.054287       0.489072

     perimeter_worst  area_worst  smoothness_worst  compactness_worst  \
0           0.668310    0.450698          0.601136           0.619292
1           0.539818    0.435214          0.347553           0.154563
2           0.508442    0.374508          0.483590           0.385375
3           0.241347    0.094008          0.915472           0.814012
4           0.506948    0.341575          0.437364           0.172415
..               …           …               …                 …
564         0.576174    0.452664          0.461137           0.178527
565         0.520892    0.379915          0.300007           0.159997
566         0.379949    0.230731          0.282177           0.273705
567         0.668310    0.402035          0.619626           0.815758
568         0.043578    0.020497          0.124084           0.036043

     concavity_worst  concave points_worst  symmetry_worst  \
0           0.568610              0.912027        0.598462
1           0.192971              0.639175        0.233590
2           0.359744              0.835052        0.403706
3           0.548642              0.884880        1.000000
4           0.319489              0.558419        0.157500
..               …                    …             …
564         0.328035              0.761512        0.097575
565         0.256789              0.559450        0.198502
566         0.271805              0.487285        0.128721
567         0.749760              0.910653        0.497142
568         0.000000              0.000000        0.257441

     fractal_dimension_worst
0                   0.418864
```

```
1                     0.222878
2                     0.213433
3                     0.773711
4                     0.142595
..                         …
564                   0.105667
565                   0.074315
566                   0.151909
567                   0.452315
568                   0.100682

[569 rows x 30 columns]
```

[76]:

```python
y
```

[76]:

```
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
```

[77]:

```python
# splitting the data
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.25,␣
 ↪random_state=0)
```

[78]:

```python
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
accuracy_knn = metrics.accuracy_score(y_test, y_pred)
precision_score_knn = metrics.precision_score(y_test, y_pred)
recall_score_knn = metrics.recall_score(y_test, y_pred)
f1_score_knn = metrics.f1_score(y_test,y_pred)
print("Accuracy du KNN : " , accuracy_knn)
print("precision score du KNN : ", precision_score_knn )
print("recall score du KNN : ", recall_score_knn )
print("f1 score du KNN : ",f1_score_knn)
```

```
Accuracy du KNN :  0.972027972027972
precision score du KNN :  1.0
recall score du KNN :  0.9245283018867925
f1 score du KNN :  0.9607843137254902
```

[79]:
```
svm = SVC()
svm.fit(X_train, y_train)
y_pred = svm.predict(X_test)
accuracy_svm = metrics.accuracy_score(y_test, y_pred)
precision_score_svm = metrics.precision_score(y_test, y_pred)
recall_score_svm = metrics.recall_score(y_test, y_pred)
f1_score_svm = metrics.f1_score(y_test,y_pred)
print("Accuracy  : " , accuracy_svm)
print("precision score  : ", precision_score_svm )
print("recall score  ", recall_score_svm )
print("f1 score  ",f1_score_svm)
```

```
Accuracy du KNN :  0.972027972027972
precision score du KNN :  0.9803921568627451
recall score du KNN :  0.9433962264150944
f1 score du KNN :  0.9615384615384616
```

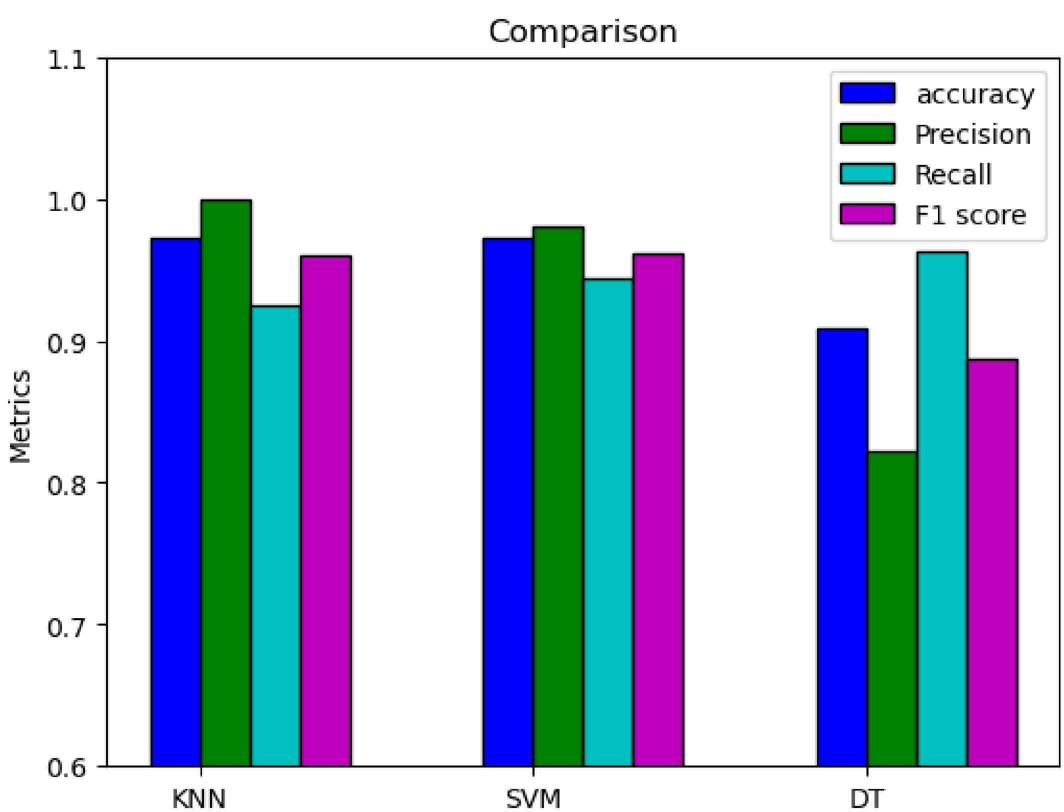
[80]:
```
clf = tree.DecisionTreeClassifier()
clf.fit(X_train, y_train)
y_pred= clf.predict(X_test)
accuracy_dt = metrics.accuracy_score(y_test, y_pred)
precision_score_dt =  metrics.precision_score(y_test, y_pred)
recall_score_dt = metrics.recall_score(y_test, y_pred)
f1_score_dt = metrics.f1_score(y_test , y_pred)
print("Accuracy:", accuracy_dt)
print("Precision: ",precision_score_dt)
print("Recall: ",recall_score_dt)
print("F1 score : ",f1_score_dt)
```

```
Accuracy: 0.9090909090909091
Precision:  0.8225806451612904
Recall:  0.9622641509433962
F1 score :  0.8869565217391304
```

[81]:
```
acc = [accuracy_knn , accuracy_svm , accuracy_dt]
pre = [precision_score_knn,precision_score_svm,precision_score_dt]
rec = [recall_score_knn , recall_score_svm, recall_score_dt]
f1 = [f1_score_knn , f1_score_svm, f1_score_dt]
```

[82]:
```
n=3
r = np.arange(n)
width = 0.15

plt.bar(r, acc, color = 'b',
        width = width, edgecolor = 'black',
        label='accuracy')
plt.bar(r + width, pre, color = 'g',
        width = width, edgecolor = 'black',
```

```
        label='Precision')

plt.bar(r + 2*width, rec, color = 'c',
        width = width, edgecolor = 'black',
        label='Recall')

plt.bar(r + 3*width, f1, color = 'm',
        width = width, edgecolor = 'black',
        label='F1 score')

plt.xlabel("")
plt.ylabel("Metrics")
plt.title("Comparison")

# plt.grid(linestyle='--')
plt.xticks(r + width/2,['KNN','SVM','DT'])
plt.legend()
plt.ylim(0.6,1.1)

plt.show()
```



6.7 balanced data : Under sampling

[87]:
```
X_resampled
```

[87]:
```
     radius_mean  texture_mean  perimeter_mean  area_mean  smoothness_mean  \
0       0.308060      0.425769        0.297975   0.177094         0.314977
1       0.264991      0.293879        0.249050   0.146554         0.282567
2       0.373373      0.355090        0.361620   0.227953         0.390358
3       0.082967      0.241123        0.079331   0.038515         0.462851
4       0.223816      0.252959        0.213461   0.117413         0.407240
..           …             …              …          …               …
419     0.659709      0.520122        0.685578   0.510498         0.517017
420     0.690000      0.428813        0.678668   0.566490         0.526948
421     0.622320      0.626987        0.604036   0.474019         0.407782
422     0.455251      0.621238        0.445788   0.303118         0.288165
423     0.644564      0.663510        0.665538   0.475716         0.588336

     compactness_mean  concavity_mean  concave points_mean  symmetry_mean  \
0            0.176676        0.111317             0.168191       0.378283
1            0.069873        0.004358             0.014533       0.321717
2            0.196522        0.159888             0.246074       0.215657
3            0.168395        0.000000             0.000000       0.467172
4            0.128918        0.089246             0.160984       0.230303
..                …               …                    …              …
419          0.626403        0.743674             0.732604       0.550000
420          0.296055        0.571462             0.690358       0.336364
421          0.257714        0.337395             0.486630       0.349495
422          0.254340        0.216753             0.263519       0.267677
423          0.790197        0.823336             0.755467       0.675253

     fractal_dimension_mean  …  radius_worst  texture_worst  \
0                  0.152064  …      0.256848       0.527719
1                  0.180918  …      0.198150       0.294776
2                  0.158382  …      0.287442       0.438699
3                  0.442713  …      0.079687       0.287313
4                  0.231466  …      0.180719       0.249733
..                      …   …           …             …
419                0.396588  …      0.581999       0.463486
420                0.132056  …      0.623266       0.383262
421                0.113100  …      0.560655       0.699094
422                0.137321  …      0.393099       0.589019
423                0.425442  …      0.633582       0.730277

     perimeter_worst  area_worst  smoothness_worst  compactness_worst  \
0           0.241994    0.126229          0.297365           0.139525
1           0.175059    0.093123          0.215479           0.037789
2           0.266398    0.147070          0.333025           0.108188
```

```
3            0.067732    0.032393          0.494156            0.100620
4            0.169381    0.082653          0.403685            0.074424
..                ...         ...               ...                 ...
419          0.640918    0.401543          0.459156            0.379651
420          0.576174    0.452664          0.461137            0.178527
421          0.520892    0.379915          0.300007            0.159997
422          0.379949    0.230731          0.282177            0.273705
423          0.668310    0.402035          0.619626            0.815758

     concavity_worst  concave points_worst  symmetry_worst  \
0            0.182268              0.440550        0.257441
1            0.004456              0.030144        0.185295
2            0.135783              0.349485        0.158486
3            0.000000              0.000000        0.173467
4            0.121486              0.377663        0.198502
..                ...                   ...             ...
419          0.527077              0.873540        0.268874
420          0.328035              0.761512        0.097575
421          0.256789              0.559450        0.198502
422          0.271805              0.487285        0.128721
423          0.749760              0.910653        0.497142

     fractal_dimension_worst
0                   0.092680
1                   0.060803
2                   0.071822
3                   0.220451
4                   0.104486
..                       ...
419                 0.286567
420                 0.105667
421                 0.074315
422                 0.151909
423                 0.452315

[424 rows x 30 columns]
```

[88]:
```
y_resampled
```

[88]:
```
0      0
1      0
2      0
3      0
4      0
      ..
419    1
420    1
```

```
421    1
422    1
423    1
Name: diagnosis, Length: 424, dtype: int64
```

[89]:
```
# splitting the data
X_train, X_test, y_train, y_test = train_test_split(X_resampled,y_resampled,␣
 ↪test_size=0.25, random_state=0)
```

[90]:
```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
accuracy_knn = metrics.accuracy_score(y_test, y_pred)
precision_score_knn = metrics.precision_score(y_test, y_pred)
recall_score_knn = metrics.recall_score(y_test, y_pred)
f1_score_knn = metrics.f1_score(y_test,y_pred)
print("Accuracy du KNN : " , accuracy_knn)
print("precision score du KNN : ", precision_score_knn )
print("recall score du KNN : ", recall_score_knn )
print("f1 score du KNN : ",f1_score_knn)
```

```
Accuracy du KNN :  0.9716981132075472
precision score du KNN :  0.9767441860465116
recall score du KNN :  0.9545454545454546
f1 score du KNN :  0.9655172413793104
```

[91]:
```
svm = SVC()
svm.fit(X_train, y_train)
y_pred = svm.predict(X_test)
accuracy_svm = metrics.accuracy_score(y_test, y_pred)
precision_score_svm = metrics.precision_score(y_test, y_pred)
recall_score_svm = metrics.recall_score(y_test, y_pred)
f1_score_svm = metrics.f1_score(y_test,y_pred)
print("Accuracy  : " , accuracy_svm)
print("precision score : ", precision_score_svm )
print("recall score  : ", recall_score_svm )
print("f1 score  : ",f1_score_svm)
```

```
Accuracy du KNN :  0.9716981132075472
precision score du KNN :  0.9767441860465116
recall score du KNN :  0.9545454545454546
f1 score du KNN :  0.9655172413793104
```

[ ]:
```
clf = tree.DecisionTreeClassifier()
clf.fit(X_train, y_train)clf = tree.DecisionTreeClassifier()
clf.fit(X_train, y_train)
y_pred= clf.predict(X_test)
accuracy_dt = metrics.accuracy_score(y_test, y_pred)
```

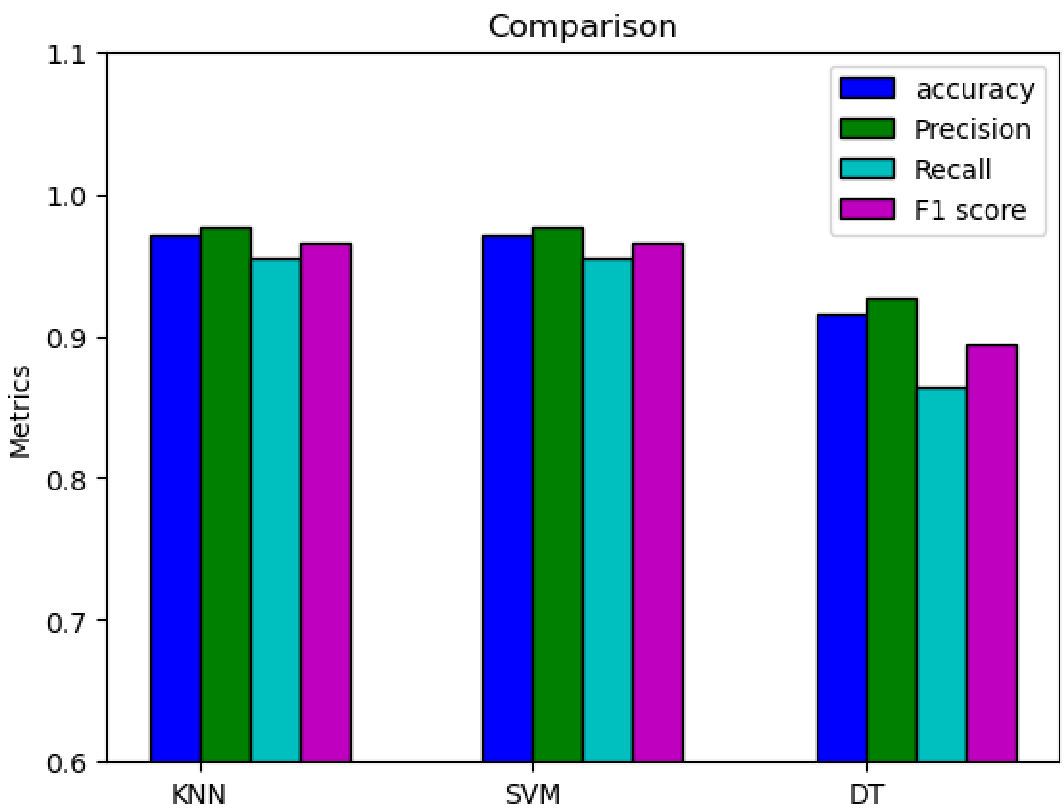
```
precision_score_dt =  metrics.precision_score(y_test, y_pred)
recall_score_dt = metrics.recall_score(y_test, y_pred)
f1_score_dt = metrics.f1_score(y_test , y_pred)
print("Accuracy:", accuracy_dt)
print("Precision: ",precision_score_dt)
print("Recall: ",recall_score_dt)
print("F1 score : ",f1_score_dt)
y_pred= clf.predict(X_test)
accuracy_dt = metrics.accuracy_score(y_test, y_pred)
precision_score_dt =  metrics.precision_score(y_test, y_pred)
recall_score_dt = metrics.recall_score(y_test, y_pred)
f1_score_dt = metrics.f1_score(y_test , y_pred)
print("Accuracy:", accuracy_dt)
print("Precision: ",precision_score_dt)
print("Recall: ",recall_score_dt)
print("F1 score : ",f1_score_dt)clf = tree.DecisionTreeClassifier()
clf.fit(X_train, y_train)
y_pred= clf.predict(X_test)
accuracy_dt = metrics.accuracy_score(y_test, y_pred)
precision_score_dt =  metrics.precision_score(y_test, y_pred)
recall_score_dt = metrics.recall_score(y_test, y_pred)
f1_score_dt = metrics.f1_score(y_test , y_pred)
print("Accuracy:", accuracy_dt)
print("Precision: ",precision_score_dt)
print("Recall: ",recall_score_dt)
print("F1 score : ",f1_score_dt)
```

[92]:
```
clf = tree.DecisionTreeClassifier()
clf.fit(X_train, y_train)
y_pred= clf.predict(X_test)
accuracy_dt = metrics.accuracy_score(y_test, y_pred)
precision_score_dt =  metrics.precision_score(y_test, y_pred)
recall_score_dt = metrics.recall_score(y_test, y_pred)
f1_score_dt = metrics.f1_score(y_test , y_pred)
print("Accuracy:", accuracy_dt)
print("Precision: ",precision_score_dt)
print("Recall: ",recall_score_dt)
print("F1 score : ",f1_score_dt)
```

```
Accuracy: 0.9150943396226415
Precision:  0.926829268292683
Recall:  0.8636363636363636
F1 score :  0.8941176470588236
```

[93]:
```
acc = [accuracy_knn , accuracy_svm , accuracy_dt]
pre = [precision_score_knn,precision_score_svm,precision_score_dt]
rec = [recall_score_knn , recall_score_svm, recall_score_dt]
f1 = [f1_score_knn , f1_score_svm, f1_score_dt]
```

[94]:
```
n=3
r = np.arange(n)
width = 0.15

plt.bar(r, acc, color = 'b',
        width = width, edgecolor = 'black',
        label='accuracy')
plt.bar(r + width, pre, color = 'g',
        width = width, edgecolor = 'black',
        label='Precision')

plt.bar(r + 2*width, rec, color = 'c',
        width = width, edgecolor = 'black',
        label='Recall')

plt.bar(r + 3*width, f1, color = 'm',
        width = width, edgecolor = 'black',
        label='F1 score')

plt.xlabel("")
plt.ylabel("Metrics")
plt.title("Comparison")

# plt.grid(linestyle='--')
plt.xticks(r + width/2,['KNN','SVM','DT'])
plt.legend()
plt.ylim(0.6,1.1)

plt.show()
```



6.8 Balanced data random over sampler

[95]:
```
from imblearn.over_sampling import RandomOverSampler
rus = RandomOverSampler()
X_oversampling, y_oversampling = rus.fit_resample(X, y)
```

[96]:
```
# splitting the data
X_train, X_test, y_train, y_test =␣
 ↪train_test_split(X_oversampling,y_oversampling, test_size=0.25,␣
 ↪random_state=0)
```

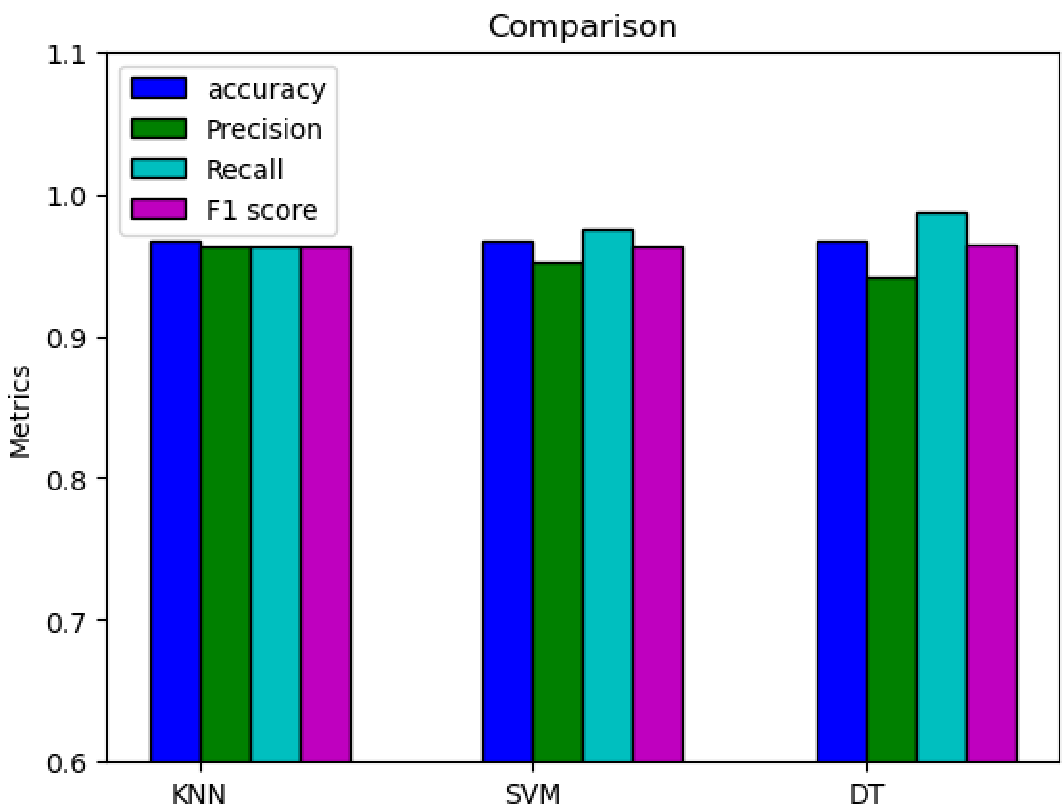
[103]:
```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
accuracy_knn = metrics.accuracy_score(y_test, y_pred)
precision_score_knn = metrics.precision_score(y_test, y_pred)
recall_score_knn = metrics.recall_score(y_test, y_pred)
f1_score_knn = metrics.f1_score(y_test,y_pred)
print("Accuracy du KNN : " , accuracy_knn)
print("precision score du KNN : ", precision_score_knn )
print("recall score du KNN : ", recall_score_knn )
```

```python
print("f1 score du KNN : ",f1_score_knn)

svm = SVC()
svm.fit(X_train, y_train)
y_pred = svm.predict(X_test)
accuracy_svm = metrics.accuracy_score(y_test, y_pred)
precision_score_svm = metrics.precision_score(y_test, y_pred)
recall_score_svm = metrics.recall_score(y_test, y_pred)
f1_score_svm = metrics.f1_score(y_test,y_pred)
print("Accuracy SVM : " , accuracy_svm)
print("precision score SVM  : ", precision_score_svm )
print("recall score SVM  ", recall_score_svm )
print("f1 score du SVM: ",f1_score_svm)

clf = tree.DecisionTreeClassifier()
clf.fit(X_train, y_train)
y_pred= clf.predict(X_test)
accuracy_dt = metrics.accuracy_score(y_test, y_pred)
precision_score_dt =  metrics.precision_score(y_test, y_pred)
recall_score_dt = metrics.recall_score(y_test, y_pred)
f1_score_dt = metrics.f1_score(y_test , y_pred)
print("Accuracy DT:", accuracy_dt)
print("Precision DT: ",precision_score_dt)
print("Recall DT",recall_score_dt)
print("F1 score DT: ",f1_score_dt)
```

```
Accuracy du KNN :  0.9664804469273743
precision score du KNN :  0.9629629629629629
recall score du KNN :  0.9629629629629629
f1 score du KNN :  0.9629629629629629
Accuracy SVM :  0.9664804469273743
precision score SVM  :  0.9518072289156626
recall score SVM   0.9753086419753086
f1 score du SVM:  0.9634146341463414
Accuracy DT: 0.9664804469273743
Precision DT:  0.9411764705882353
Recall DT 0.9876543209876543
F1 score DT:  0.963855421686747
```

[104]:
```python
acc = [accuracy_knn , accuracy_svm , accuracy_dt]
pre = [precision_score_knn,precision_score_svm,precision_score_dt]
rec = [recall_score_knn , recall_score_svm, recall_score_dt]
f1 = [f1_score_knn , f1_score_svm, f1_score_dt]

n=3
r = np.arange(n)
width = 0.15
```

```python
plt.bar(r, acc, color = 'b',
        width = width, edgecolor = 'black',
        label='accuracy')
plt.bar(r + width, pre, color = 'g',
        width = width, edgecolor = 'black',
        label='Precision')

plt.bar(r + 2*width, rec, color = 'c',
        width = width, edgecolor = 'black',
        label='Recall')

plt.bar(r + 3*width, f1, color = 'm',
        width = width, edgecolor = 'black',
        label='F1 score')

plt.xlabel("")
plt.ylabel("Metrics")
plt.title("Comparison")

# plt.grid(linestyle='--')
plt.xticks(r + width/2,['KNN','SVM','DT'])
plt.legend()
plt.ylim(0.6,1.1)

plt.show()
```



6.9 Balanced data SMOTE

[105]:
```
from imblearn.over_sampling import SMOTE
oversample = SMOTE(k_neighbors=3)
X_smote, y_smote = oversample.fit_resample(X, y)
```

[106]:
```
# splitting the data
X_train, X_test, y_train, y_test = train_test_split(X_smote,y_smote,␣
↪test_size=0.25, random_state=0)
```

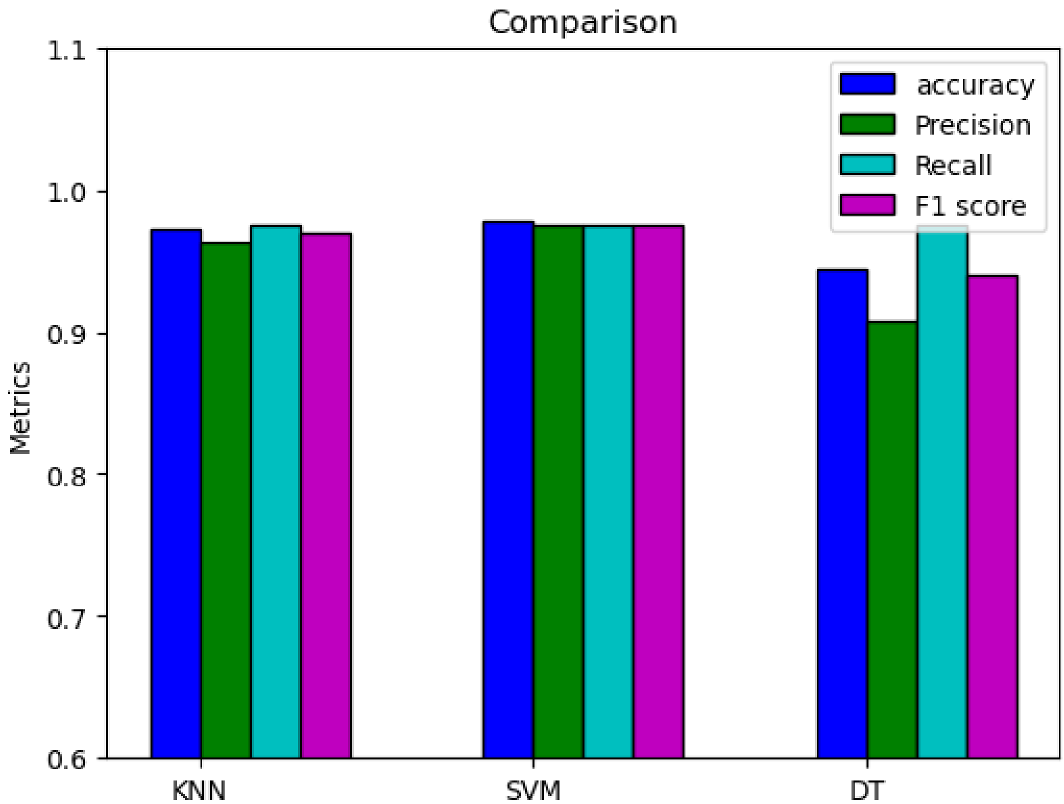
[107]:
```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
accuracy_knn = metrics.accuracy_score(y_test, y_pred)
precision_score_knn = metrics.precision_score(y_test, y_pred)
recall_score_knn = metrics.recall_score(y_test, y_pred)
f1_score_knn = metrics.f1_score(y_test,y_pred)
print("Accuracy du KNN : " , accuracy_knn)
print("precision score du KNN : ", precision_score_knn )
print("recall score du KNN : ", recall_score_knn )
print("f1 score du KNN : ",f1_score_knn)
```

```python
svm = SVC()
svm.fit(X_train, y_train)
y_pred = svm.predict(X_test)
accuracy_svm = metrics.accuracy_score(y_test, y_pred)
precision_score_svm = metrics.precision_score(y_test, y_pred)
recall_score_svm = metrics.recall_score(y_test, y_pred)
f1_score_svm = metrics.f1_score(y_test,y_pred)
print("Accuracy SVM : " , accuracy_svm)
print("precision score SVM  : ", precision_score_svm )
print("recall score SVM  ", recall_score_svm )
print("f1 score du SVM: ",f1_score_svm)

clf = tree.DecisionTreeClassifier()
clf.fit(X_train, y_train)
y_pred= clf.predict(X_test)
accuracy_dt = metrics.accuracy_score(y_test, y_pred)
precision_score_dt =  metrics.precision_score(y_test, y_pred)
recall_score_dt = metrics.recall_score(y_test, y_pred)
f1_score_dt = metrics.f1_score(y_test , y_pred)
print("Accuracy DT:", accuracy_dt)
print("Precision DT: ",precision_score_dt)
print("Recall DT",recall_score_dt)
print("F1 score DT: ",f1_score_dt)
```

```
Accuracy du KNN :  0.9720670391061452
precision score du KNN :  0.9634146341463414
recall score du KNN :  0.9753086419753086
f1 score du KNN :  0.9693251533742332
Accuracy SVM :  0.9776536312849162
precision score SVM  :  0.9753086419753086
recall score SVM   0.9753086419753086
f1 score du SVM:  0.9753086419753086
Accuracy DT: 0.9441340782122905
Precision DT:  0.9080459770114943
Recall DT 0.9753086419753086
F1 score DT:  0.9404761904761905
```

[108]:
```python
acc = [accuracy_knn , accuracy_svm , accuracy_dt]
pre = [precision_score_knn,precision_score_svm,precision_score_dt]
rec = [recall_score_knn , recall_score_svm, recall_score_dt]
f1 = [f1_score_knn , f1_score_svm, f1_score_dt]

n=3
r = np.arange(n)
width = 0.15
```

```python
plt.bar(r, acc, color = 'b',
        width = width, edgecolor = 'black',
        label='accuracy')
plt.bar(r + width, pre, color = 'g',
        width = width, edgecolor = 'black',
        label='Precision')

plt.bar(r + 2*width, rec, color = 'c',
        width = width, edgecolor = 'black',
        label='Recall')

plt.bar(r + 3*width, f1, color = 'm',
        width = width, edgecolor = 'black',
        label='F1 score')

plt.xlabel("")
plt.ylabel("Metrics")
plt.title("Comparison")

# plt.grid(linestyle='--')
plt.xticks(r + width/2,['KNN','SVM','DT'])
plt.legend()
plt.ylim(0.6,1.1)

plt.show()
```


Comparison
1.1
1.0
0.9
0.8
0.7
0.6
Metrics
accuracy
Precision
Recall
F1 score
KNN
SVM
DT

```
[ ]:
```

```
[ ]:
```