20BDS0146
VENNELA G

DIGITAL ASSESSMENT-03

BreastcancerdatasetfromUCI

In[1]:
```
importmatplotlib.pyplotasplt
importseabornassns
importpandasaspdim
port numpy as
npimportmatplotlib
%matplotlibinline
```

ReadingofData

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

Top5rowsofdata

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

Out[3]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
0	842302	M	17.99	10.38	122.80	1001.0	0.11840
1	842517	M	20.57	17.77	132.90	1326.0	0.08474
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960
3	84348301	M	11.42	20.38	77.58	386.1	0.14250
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030

5 rows × 32 columns

In[4]:
```
df.info()
```

```
<class'pandas.core.frame.DataFrame'>Range
Index:569entries,0to568
Datacolumns(total32columns):
 #   Column                  Non-NullCountDtype
---  ------                  --------------  -----
 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         569non-null     float64
```

```
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave_points_mean      569 non-null    float64
10symmetry_mean               569 non-null    float64
11fractal_dimension_mean      569 non-null    float64
12radius_se                   569 non-null    float64
13texture_se                  569 non-null    float64
14perimeter_se                569 non-null    float64
15area_se                     569 non-null    float64
16smoothness_se               569 non-null    float64
17compactness_se              569 non-null    float64
18concavity_se                569 non-null    float64
19concave_points_se           569 non-null    float64
20symmetry_se                 569 non-null    float64
21fractal_dimension_se        569 non-null    float64
22radius_worst                569 non-null    float64
23texture_worst               569 non-null    float64
24perimeter_worst             569 non-null    float64
25area_worst                  569 non-null    float64
26smoothness_worst            569 non-null    float64
27compactness_worst           569 non-null    float64
28concavity_worst             569 non-null    float64
29concave_points_worst        569 non-null    float64
30symmetry_worst              569 non-null    float64
31fractal_dimension_worst     569 non-null    float64
dtypes:float64(30),int64(1),object(1)memo
ryusage:142.4+KB
```

Droppingofidindata

In[5]:

```
df.drop(['id'],axis=1,inplace=True)df.hea
d()
```

Out[5]:

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

5 rows × 31 columns

Attributesofdata

In[6]:

```
attributes=list(df.columns)[1:]attribute
s
```

Out[6]:

```
['radius_mean','texture_m
 ean',
 'perimeter_mean','area_m
 ean',
 'smoothness_mean',
```

```
'concavity_mean',
'concave_points_mean','sy
mmetry_mean',
'fractal_dimension_mean',
'radius_se','texture_se'
,
'perimeter_se','area_se'
,
'smoothness_se','compact
ness_se','concavity_se',
'concave_points_se','symm
etry_se',
'fractal_dimension_se','r
adius_worst',
'texture_worst',
'perimeter_worst','area_w
orst',
'smoothness_worst','comp
actness_worst','concavit
y_worst',
'concave_points_worst','symmetry_worst'
,
'fractal_dimension_worst']
```

In[7]:

```
inputs=df[attributes].copy()inputs
```

Out[7]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean
0	17.99	10.38	122.80	1001.0	0.11840	0.27760
1	20.57	17.77	132.90	1326.0	0.08474	0.07864
2	19.69	21.25	130.00	1203.0	0.10960	0.15990
3	11.42	20.38	77.58	386.1	0.14250	0.28390
4	20.29	14.34	135.10	1297.0	0.10030	0.13280
...	...	...	...	...	...	..
564	21.56	22.39	142.00	1479.0	0.11100	0.11590
565	20.13	28.25	131.20	1261.0	0.09780	0.10340
566	16.60	28.08	108.30	858.1	0.08455	0.10230
567	20.60	29.33	140.10	1265.0	0.11780	0.27700
568	7.76	24.54	47.92	181.0	0.05263	0.04362

569 rows × 30 columns

ReplacingDiagnosisMandBby1and0

In[8]:

```
df['diagnosis']=df['diagnosis'].replace(['M','B'],[1,0])output
=df['diagnosis'].copy()
output
```

Out[8]:

```
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
```

In[9]:

```
inputs.describe()
```

Out[9]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_m
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345

8 rows × 30 columns

Scalingthevalues

In[10]:

```
fromsklearn.preprocessingimportMinMaxScalerscaler=MinMaxScaler()
scaler.fit(df[attributes])
```

Out[10]:

```
MinMaxScaler()
```

In[11]:

```
inputs[attributes]=scaler.transform(inputs[attributes])inputs.descri
be()
```

Out[11]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_me
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.0000
mean	0.338222	0.323965	0.332935	0.216920	0.394785	0.2606
std	0.166787	0.145453	0.167915	0.149274	0.126967	0.1619
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000
25%	0.223342	0.218465	0.216847	0.117413	0.304595	0.1396
50%	0.302381	0.308759	0.293345	0.172895	0.390358	0.2246
			.416765	0.271135	0.475490	0.3405
radius_meantexture_meanperimeter_meanarea_meansmoothness_meancompactness_memax						1.000000
	1.000000	1.000000	1.000000	1.000000	1.0000	

8 rows × 30 columns

TrainingandTestingdatasplitting

In[12]:

```
fromsklearn.model_selectionimporttrain_test_split
train_x,test_x,train_y,test_y=train_test_split(inputs,output,test_size=0.20,ran
```

In[13]:

```
train_x.head()
```

Out[13]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean
338	0.145251	0.264457	0.142492	0.070965	0.433962	0.165266
427	0.180747	0.414948	0.172759	0.091792	0.319401	0.116711
406	0.433480	0.174163	0.418147	0.278473	0.382053	0.201307
96	0.246060	0.274941	0.234953	0.130477	0.468268	0.157015
490	0.249373	0.430504	0.237648	0.137010	0.264422	0.100055

5 rows × 30 columns

In[14]:

```
train_y.head()
```

Out[14]:

In[15]:

75%	0.416442	0.408860	0

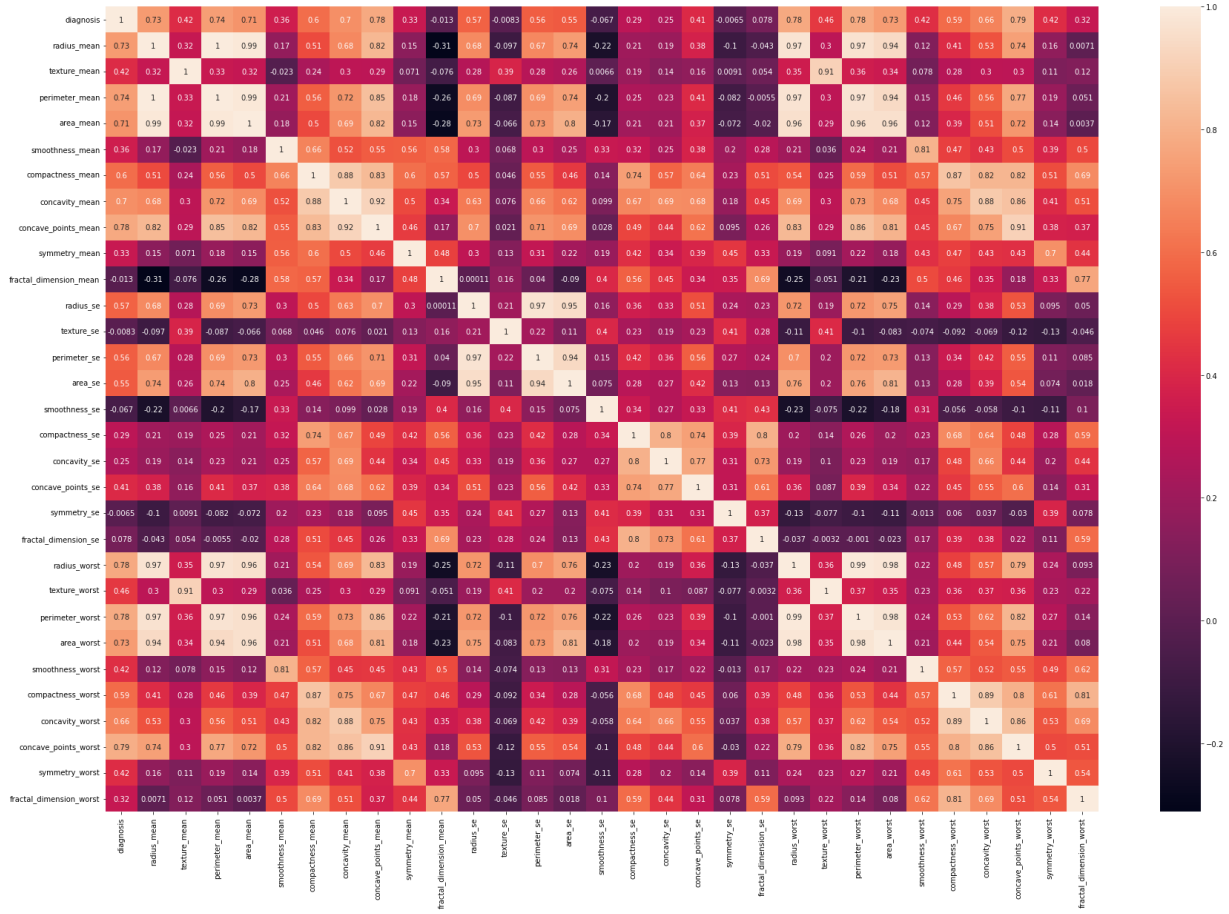
```
338    0
427    0
406    0
96     0
490    0
Name: diagnosis,dtype:int64
```

CorrelationMatrix

```
plt.figure(figsize=(30,20))
sns.heatmap(df.corr(),annot=True)
```

Out[15]:

```
<AxesSubplot:>
```



In[17]:

```
cor=df.corr()
cor1=cor[cor['diagnosis']>=0.3]
```

Featurecolumnsseperation

In[18]:

```
feature=list(cor1.index)
feature=feature[1:]#Removingtheoutcomecolumn
feature
```

Out[18]:

```
['radius_mean','texture_m
ean',
'perimeter_mean','area_m
ean',
'smoothness_mean','compa
ctness_mean','concavity_
mean',
'concave_points_mean','sy
mmetry_mean',
'radius_se',
'perimeter_se','area_se'
,
'concave_points_se','radi
us_worst',
'texture_worst',
'perimeter_worst','area_w
orst',
'smoothness_worst',
'compactness_worst','conc
avity_worst',
'concave_points_worst',
```

```
'symmetry_worst',
'fractal_dimension_worst']
```

DecisionTreeClassifierwithallfeatures

In[19]:
```
fromsklearn.treeimportDecisionTreeClassifiermode
l=DecisionTreeClassifier(random_state=0)
```

In[20]:
```
%%time
model.fit(train_x,train_y)
```

Out[20]:
```
Walltime:9.02ms
DecisionTreeClassifier(random_state=0)
```

In[21]:
```
fromsklearn.metricsimportaccuracy_score,confusion_matrixtest_preds=m
odel.predict(test_x)
acc1=accuracy_score(test_y,test_preds)
```

In[22]:
```
fromsklearn.metricsimportprecision_score
fromsklearn.metricsimportrecall_score
fromsklearn.metricsimportf1_score
prec1=precision_score(test_y,test_preds)re
c1=recall_score(test_y,test_preds)
f11=f1_score(test_y,test_preds)
```

Accuracy

In[23]:
```
acc1
```

Out[23]:
```
0.9122807017543859
```

PrecisionScore

In[88]:
```
prec1
```

Out[88]:
```
0.8627450980392157
```

Recall score

In[89]:
```
rec1
```

Out[89]:
```
0.9361702127659575
```

F1score

In[90]:
```
f11
```

Out[90]:
```
0.8979591836734694
```

DecisionTreeClassifierwithselectedfeatures

In[24]:
```
model1=DecisionTreeClassifier(random_state=0)
```

In[25]:
```
%%time
model1.fit(train_x[feature],train_y)
```

Out[25]:
```
Walltime:8.36ms
DecisionTreeClassifier(random_state=0)
```

In[26]:
```
test_preds1=model1.predict(test_x[feature])
```

In[27]:
```
acc12=accuracy_score(test_y,test_preds1)
prec12=precision_score(test_y,test_preds1)rec12=
recall_score(test_y,test_preds1)
f112=f1_score(test_y,test_preds1)
```

PrecisionScore

In[28]:
```
prec12
```

Out[28]:
```
0.9166666666666666
```

AccuracyScore

In[29]:
```
acc12
```

Out[29]:
```
0.9385964912280702
```

Recall Score

In[96]:
```
rec12
```

Out[96]:
```
0.9361702127659575
```

F1Score

In[97]:
```
f112
```

Out[97]:
```
0.9263157894736843
```

RandomForestClassifierwithallfeatures

In[30]:
```
fromsklearn.ensembleimportRandomForestClassifier
model2=RandomForestClassifier(n_jobs=-1,random_state=0)
```


In[31]:

```
%%time
model2.fit(train_x,train_y)
```

```
Walltime:156ms
RandomForestClassifier(n_jobs=-1,random_state=0)
```

Out[31]:

In[32]:

```
test_preds2=model2.predict(test_x)
```

In[33]:

```
acc2=accuracy_score(test_y,test_preds2)
prec2=precision_score(test_y,test_preds2)rec2=re
call_score(test_y,test_preds2)
f12=f1_score(test_y,test_preds2)
```

AccuracyScore

In[34]:

```
acc2
```

```
0.9649122807017544
```

Out[34]:

PrecisionScore

In[35]:

```
prec2
```

```
0.9387755102040817
```

Out[35]:

Recall Score

In[36]:

```
rec2
```

```
0.9787234042553191
```

Out[36]:

F1Score

In[37]:

```
f12
```

```
0.9583333333333333
```

Out[37]:

RandomForestClassifierwithselectedfeatures

In[38]:

```
model3=RandomForestClassifier(n_jobs=-1,random_state=0)
```

In[39]:

```
%%time
model3.fit(train_x[feature],train_y)
```

```
Walltime:95.9ms
                                                dom_state=0)
```

In[40]:
```
test_preds3=model3.predict(test_x[feature])
```

In[41]:
```
acc22=accuracy_score(test_y,test_preds3)
prec22=precision_score(test_y,test_preds3)rec22=
recall_score(test_y,test_preds3)
f122=f1_score(test_y,test_preds3)
```

Accuracyscore

In[42]:
```
acc22
```

Out[42]:
```
0.9649122807017544
```

Recall score

In[43]:
```
rec22
```

Out[43]:
```
0.9574468085106383
```

PrecisionScore

In[44]:
```
prec22
```

Out[44]:
```
0.9574468085106383
```

F1Score

In[45]:
```
f122
```

Out[45]:
```
0.9574468085106385
```

SVMClassifierwithallfeatures

In[46]:
```
fromsklearnimportsvm
```

In[47]:
```
%%time
clf=svm.SVC(kernel='linear')
```

```
Walltime:0ns
```

In[48]:
```
clf.fit(train_x,train_y)
test_preds4=clf.predict(test_x)
```

acc3= accuracy_score(test_y, test_preds4) prec3 = precision_score(test_y, test_preds4) rec3 =recall_score(test_y, test_preds4) f13 = f1_score(test_y, test_preds4)

AccuracyScore

In[50]:
```
acc3
```

Out[50]:
```
0.9649122807017544
```

PrecisionScore

In[51]:
```
prec3
```

Out[51]:
```
0.9777777777777777
```

Recall Score

In[52]:
```
rec3
```

Out[52]:
```
0.9361702127659575
```

F1Score

In[53]:
```
f13
```

Out[53]:
```
0.9565217391304347
```

SVMClassifierwithselectedfeatures

In[54]:
```
%%time
clf1=svm.SVC(kernel='linear')
```

```
Walltime:0ns
```

In[55]:
```
clf1.fit(train_x[feature],train_y)
```

Out[55]:
```
SVC(kernel='linear')
```

In[56]:
```
test_preds5=clf1.predict(test_x[feature])acc32=
accuracy_score(test_y,test_preds5)
prec32=precision_score(test_y,test_preds5)rec32=
recall_score(test_y,test_preds5)
f132=f1_score(test_y,test_preds5)
```

AccuracyScore

In[57]:
```
acc32
```

Out[57]:
```
0.9649122807017544
```

PrecisionScore

In[58]:
```
prec32
```

Out[58]: 0.9777777777777777

RecallScore

In[59]:
```
rec32
```

Out[59]: 0.9361702127659575

F1Score

In[60]:
```
f132
```

Out[60]: 0.9565217391304347

KNNClassifierwithallfeatures

In[61]:
```
fromsklearn.neighborsimportKNeighborsClassifierneigh=KN
eighborsClassifier(n_neighbors=3)
```

In[62]:
```
%%time
neigh.fit(train_x,train_y)
```

Out[62]:
```
Walltime:2.68ms
KNeighborsClassifier(n_neighbors=3)
```

In[63]:
```
test_preds6=neigh.predict(test_x)
```

In[64]:
```
acc4=accuracy_score(test_y,test_preds6)
prec4=precision_score(test_y,test_preds6)rec4=re
call_score(test_y,test_preds6)
f14=f1_score(test_y,test_preds6)
```

AccuracyScore

In[65]:
```
acc4
```

Out[65]: 0.9649122807017544

PrecisionScore

In[66]:
```
prec4
```

Out[66]: 1.0

Recall Score

In[67]:

Out[67]:
```
0.91489361702127614
```

F1Score

In[68]:
```
f14
```

Out[68]:
```
0.9555555555555556
```

KNNClassifierwithselectedfeatures

In[69]:
```
neigh1=KNeighborsClassifier(n_neighbors=3)
```

In[70]:
```
%%time
neigh1.fit(train_x[feature],train_y)
```

Out[70]:
```
Walltime:2ms
KNeighborsClassifier(n_neighbors=3)
```

In[71]:
```
test_preds7=neigh1.predict(test_x[feature])
```

In[72]:
```
acc42=accuracy_score(test_y,test_preds7)
prec42=precision_score(test_y,test_preds7)rec42=
recall_score(test_y,test_preds7)
f142=f1_score(test_y,test_preds7)
```

AccuracyScore

In[73]:
```
acc42
```

Out[73]:
```
0.9649122807017544
```

PrecisionScore

In[74]:
```
prec42
```

Out[74]:
```
0.9777777777777777
```

Recall Score

In[75]:
```
rec42
```

Out[75]:
```
0.9361702127659575
```

F1Score

In[76]:
```
f142
```

Out[76]: 0.9565217391304347

GuassianNaiveBayesClassifierwithallfeatures

In[77]:
```
fromsklearn.naive_bayesimportGaussianNBgnb=G
aussianNB()
```

In[78]:
```
%%time
gnb.fit(train_x,train_y)
```

Out[78]:
```
Walltime:3msGa
ussianNB()
```

In[79]:
```
test_preds8=gnb.predict(test_x)
acc5=accuracy_score(test_y,test_preds8)
prec5=precision_score(test_y,test_preds8)rec5=re
call_score(test_y,test_preds8)
f15=f1_score(test_y,test_preds8)
```

AccuracyScore

In[80]:
```
acc5
```

Out[80]: 0.9035087719298246

PrecisionScore

In[81]:
```
prec5
```

Out[81]: 0.875

Recall Score

In[82]:
```
rec5
```

Out[82]: 0.8936170212765957

F1Score

In[83]:
```
f15
```

Out[83]: 0.8842105263157894

GuassianNaiveBayesClassifierwithselected features

In[84]:
```
gnb1=GaussianNB()
```

In[85]:
```
%%time
gnb1.fit(train_x[feature],train_y)
```

Out[85]:
```
Walltime:5msGa
ussianNB()
```

In[86]:
```
test_preds9=gnb1.predict(test_x[feature])
```

In[87]:
```
acc52=accuracy_score(test_y,test_preds9)
prec52=precision_score(test_y,test_preds9)rec52=
recall_score(test_y,test_preds9)
f152=f1_score(test_y,test_preds9)
```

AccuracyScore

In[88]:
```
acc52
```

Out[88]:
```
0.9122807017543859
```

PrecisionScore

In[89]:
```
prec52
```

Out[89]:
```
0.8775510204081632
```

Recall Score

In[90]:
```
rec52
```

Out[90]:
```
0.9148936170212766
```

F1Score

In[91]:
```
f152
```

Out[91]:
```
0.8958333333333333
```

COMPARISIONTABLEOFALLALGORITHMSWITH ALLFEATURES

In[92]:
```
d={'Algorithm':['Decisontree','Randomforest','SVM','KNN','Bayesianclassifer'],'Acresul
twithallFeatures=pd.DataFrame(d)
resultwithallFeatures
```

Out[92]:

	Algorithm	Accuracy score	Precision Score	Recall score	F1score
0	Decison tree	0.912281	0.862745	0.936170	0.897959

	Algorithm	Accuracyscore	PrecisionScore	Recallscore	F1score
1	Random forest	0.964912	0.938776	0.978723	0.958333
2	SVM	0.964912	0.977778	0.936170	0.956522
3	KNN	0.964912	1.000000	0.914894	0.955556
4	Bayesian classifer	0.903509	0.875000	0.893617	0.884211

COMPARISIONTABLEOFALLALGORITHMSWITH SELECTEDFEATURES

In[93]:

```
d1={'Algorithm':['Decisontree','Randomforest','SVM','KNN','Bayesianclassifer'],'Aresul
twithselectedFeatures=pd.DataFrame(d1)
resultwithselectedFeatures
```

Out[93]:

	Algorithm	Accuracyscore	PrecisionScore	Recallscore	F1score
0	Decison tree	0.938596	0.916667	0.936170	0.926316
1	Random forest	0.964912	0.957447	0.957447	0.957447
2	SVM	0.964912	0.977778	0.936170	0.956522
3	KNN	0.964912	0.977778	0.936170	0.956522
4	Bayesian classifer	0.912281	0.877551	0.914894	0.895833

In[]: