```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score, mean_squared_error, r2_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
import seaborn as sns
```

```
from google.colab import files
data=files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Dataset of Diabetes 1.csv to Dataset of Diabetes 1 (1).csv
```

```
df=pd.read_csv("/content/Dataset of Diabetes 1 (1).csv")
```

```
df
```

	ID	Gender	AGE	HbA1c	HDL	LDL	BMI	CLASS
0	502	F	50	4.9	2.4	1.4	24.0	N
1	735	M	26	4.9	1.1	2.1	23.0	N
2	420	F	50	4.9	2.4	1.4	24.0	N
3	680	F	50	4.9	2.4	1.4	24.0	N
4	504	M	33	4.9	0.8	2.0	21.0	N
...	...	...	...	...	...	...	...	...
494	397	M	55	11.7	1.3	2.3	30.0	Y
495	681	F	58	9.0	3.2	1.4	35.0	Y
496	749	M	55	10.0	1.2	3.4	33.0	Y
497	321	F	54	10.7	1.1	4.2	39.0	Y
498	381	M	58	8.0	0.9	2.0	29.0	Y

499 rows × 8 columns

```
arr=np.array(df)
arr
```

```
array([[502, 'F', 50, ..., 1.4, 24.0, 'N'],
       [735, 'M', 26, ..., 2.1, 23.0, 'N'],
       [420, 'F', 50, ..., 1.4, 24.0, 'N'],
       ...,
       [749, 'M', 55, ..., 3.4, 33.0, 'Y'],
       [321, 'F', 54, ..., 4.2, 39.0, 'Y'],
       [381, 'M', 58, ..., 2.0, 29.0, 'Y']], dtype=object)
```

```
df.isnull().sum()
```

```
ID        0
Gender    0
AGE       0
HbA1c     0
HDL       0
LDL       0
BMI       0
CLASS     0
dtype: int64
```

```
df.isnull().sum().sum()
```

```
0
```

```
df.shape
```

```
(499, 8)
```

```
df.replace({'Gender':{'M':0,'F':1}},inplace=True)
```

```
x=df.drop(columns='ID',axis=1)
df=x
```

```
x=df.drop(columns='CLASS',axis=1)
y=df['CLASS']
train_x,test_x,train_y,test_y=train_test_split(x,y,test_size=(0.3))
train_x.shape
```

```
(349, 6)
```

```
train_x
```

	Gender	AGE	HbA1c	HDL	LDL	BMI
313	0	60	9.0	0.80	3.70	30.0
35	1	39	4.0	1.10	2.60	22.0
432	0	52	7.9	1.10	2.50	28.0
116	0	50	6.2	0.60	1.00	24.0
362	1	60	6.2	1.00	4.40	27.0
...	...	...	...	...	...	...
268	1	55	5.9	1.00	2.00	30.0
117	1	49	6.0	0.75	1.35	23.0
90	0	30	5.6	0.90	3.30	24.5
49	1	47	5.0	1.00	2.40	22.0
59	1	44	4.1	1.00	3.50	21.0

349 rows × 6 columns

```
test_x.shape
```

```
(150, 6)
```

```
test_x
```

	Gender	AGE	HbA1c	HDL	LDL	BMI
98	0	60	0.9	1.1	3.6	24.0
275	0	56	7.6	0.9	3.3	26.0
66	1	35	5.0	1.3	2.4	20.0
179	0	49	5.2	3.9	0.8	24.0
339	1	61	11.5	1.1	2.5	26.0
...	...	...	...	...	...	...
128	0	50	5.9	1.0	3.7	25.0
195	0	50	4.0	1.1	3.2	23.0
401	0	60	6.8	0.7	4.1	33.0
264	1	55	5.9	1.0	2.0	30.0
14	1	50	4.0	1.2	2.2	24.0

150 rows × 6 columns

```
pd.DataFrame(train_x).describe()
```

```
          Gender      AGE     HbA1c       HDL       LDL       BMI
```

```
train_y.shape
```

```
(349,)
```

```
  std   0.498739  9.138748  2.099589  0.587388  1.057292  5.331973
```

```
from sklearn.preprocessing import StandardScaler


sc=StandardScaler()
```

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
train_x_sc=sc.fit_transform(train_x)
pd.DataFrame(train_x_sc).describe()
```

	0	1	2	3	4	5
count	3.490000e+02	3.490000e+02	3.490000e+02	3.490000e+02	3.490000e+02	3.490000e+02
mean	1.030694e-16	2.048664e-16	1.864157e-16	-3.200242e-16	-1.072049e-16	4.549051e-16
std	1.001436e+00	1.001436e+00	1.001436e+00	1.001436e+00	1.001436e+00	1.001436e+00
min	-9.147907e-01	-2.517858e+00	-2.915789e+00	-1.163042e+00	-2.217844e+00	-1.744950e+00
25%	-9.147907e-01	-5.453945e-01	-7.694347e-01	-4.810838e-01	-7.023716e-01	-8.058646e-01
50%	-9.147907e-01	3.312557e-01	-1.016802e-01	-1.401044e-01	-1.340694e-01	-5.459612e-02
75%	1.093146e+00	5.504183e-01	6.137711e-01	2.008749e-01	7.183840e-01	8.657077e-01
max	1.093146e+00	2.851625e+00	3.189396e+00	9.236828e+00	6.874991e+00	2.809615e+00

```
from sklearn.linear_model import Perceptron
p=Perceptron()
p.fit(train_x_sc,train_y)
train_pred=p.predict(train_x_sc)
test_pred=p.predict(test_x)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:444: UserWarning: X has feature names, but Perceptron was fitted without fea
  f"X has feature names, but {self.__class__.__name__} was fitted without"
```

```
from sklearn.metrics import accuracy_score
print('Training Accuracy =',accuracy_score(train_pred,train_y))
```

```
Training Accuracy = 0.7449856733524355
```

```
print('Test Accuracy =',accuracy_score(test_y,test_pred))
```

```
Test Accuracy = 0.6866666666666666
```

```
from sklearn.svm import SVC
svc=SVC()
```

```
svc.fit(train_x_sc,train_y)
```

```
SVC()
```

```
train_pred_svc=svc.predict(train_x_sc)
test_pred_svc=svc.predict(test_x)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:444: UserWarning: X has feature names, but SVC was fitted without feature na
  f"X has feature names, but {self.__class__.__name__} was fitted without"
```

```
print("Training",accuracy_score(train_pred_svc,train_y))
```

```
Training 0.9054441260744985
```

```
print("Test",accuracy_score(test_pred_svc,test_y))
```

```
Test 0.6866666666666666
```

```
svc1=SVC(kernel="rbf",C=0.5)
svc1.fit(train_x_sc,train_y)
```

```
train_pred_svc1=svc1.predict(train_x_sc)
test_pred_svc1=svc1.predict(test_x)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:444: UserWarning: X has feature names, but SVC was fitted without feature na
  f"X has feature names, but {self.__class__.__name__} was fitted without"
```

```
print("Train",accuracy_score(train_pred_svc1,train_y))
```

```
Train 0.8681948424068768
```

```
print("test",accuracy_score(test_pred_svc1,test_y))
```

```
test 0.6866666666666666
```

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=3)
knn.fit(train_x_sc,train_y)
train_pred_knn=knn.predict(train_x_sc)
test_pred_knn=knn.predict(test_x)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:444: UserWarning: X has feature names, but KNeighborsClassifier was fitted w
  f"X has feature names, but {self.__class__.__name__} was fitted without"
```

```
print("Train accuracy",accuracy_score(train_pred_knn,train_y))
print("test accuracy",accuracy_score(test_pred_knn,test_y))
```

```
Train accuracy 0.9226361031518625
test accuracy 0.6866666666666666
```

```
from sklearn.tree import DecisionTreeClassifier
dt=DecisionTreeClassifier()
dt.fit(train_x_sc,train_y)
train_pred_dt=dt.predict(train_x_sc)
test_pred_dt=dt.predict(test_x)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:444: UserWarning: X has feature names, but DecisionTreeClassifier was fitted
  f"X has feature names, but {self.__class__.__name__} was fitted without"
```

```
print("Training",accuracy_score(train_pred_dt,train_y))
print("testing",accuracy_score(test_pred_dt,test_y))
```

```
Training 1.0
testing 0.6866666666666666
```

```
dt1=DecisionTreeClassifier(criterion='entropy')
dt1.fit(train_x_sc,train_y)
train_pred_dt1=dt1.predict(train_x_sc)
test_pred_dt1=dt1.predict(test_x)
```

```
print("Training value",accuracy_score(train_pred_dt1,train_y))
print("testing value",accuracy_score(test_pred_dt1,test_y))
```

```
Training value 1.0
testing value 0.6866666666666666
```

```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier()
rf.fit(train_x_sc,train_y)
train_pred_rf=rf.predict(train_x_sc)
test_pred_rf=rf.predict(test_x)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:444: UserWarning: X has feature names, but RandomForestClassifier was fitted
  f"X has feature names, but {self.__class__.__name__} was fitted without"
```

```
print("Training",accuracy_score(train_pred_rf,train_y))
print("Testing",accuracy_score(test_pred_rf,test_y))
```

```
Training 1.0
Testing 0.6866666666666666
```

```
from sklearn.ensemble import VotingClassifier
```

```
p=Perceptron()
svm=SVC()
knn=KNeighborsClassifier()
```

```
vc=VotingClassifier(estimators=[('perceptron',p),('svm',svm),('knn',knn)],voting='hard', weights=[3,1,1])
vc.fit(train_x_sc,train_y)
train_pred_vc=vc.predict(train_x_sc)
test_pred_vc=vc.predict(test_x)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:444: UserWarning: X has feature names, but Perceptron was fitted without fea
  f"X has feature names, but {self.__class__.__name__} was fitted without"
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:444: UserWarning: X has feature names, but SVC was fitted without feature na
  f"X has feature names, but {self.__class__.__name__} was fitted without"
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:444: UserWarning: X has feature names, but KNeighborsClassifier was fitted w
  f"X has feature names, but {self.__class__.__name__} was fitted without"
```

```
print("Training",accuracy_score(train_pred_vc,train_y))
print("Testing",accuracy_score(test_pred_vc,test_y))
```

```
Training 0.7449856733524355
Testing 0.6866666666666666
```

```
from sklearn.ensemble import BaggingClassifier
bag=BaggingClassifier(base_estimator=knn,n_estimators=20)
bag.fit(train_x_sc,train_y)
train_pred_bag=bag.predict(train_x_sc)
test_pred_bag=bag.predict(test_x)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:444: UserWarning: X has feature names, but BaggingClassifier was fitted with
  f"X has feature names, but {self.__class__.__name__} was fitted without"
```

```
print("Training",accuracy_score(train_pred_bag,train_y))
print("Testing",accuracy_score(test_pred_bag,test_y))
```

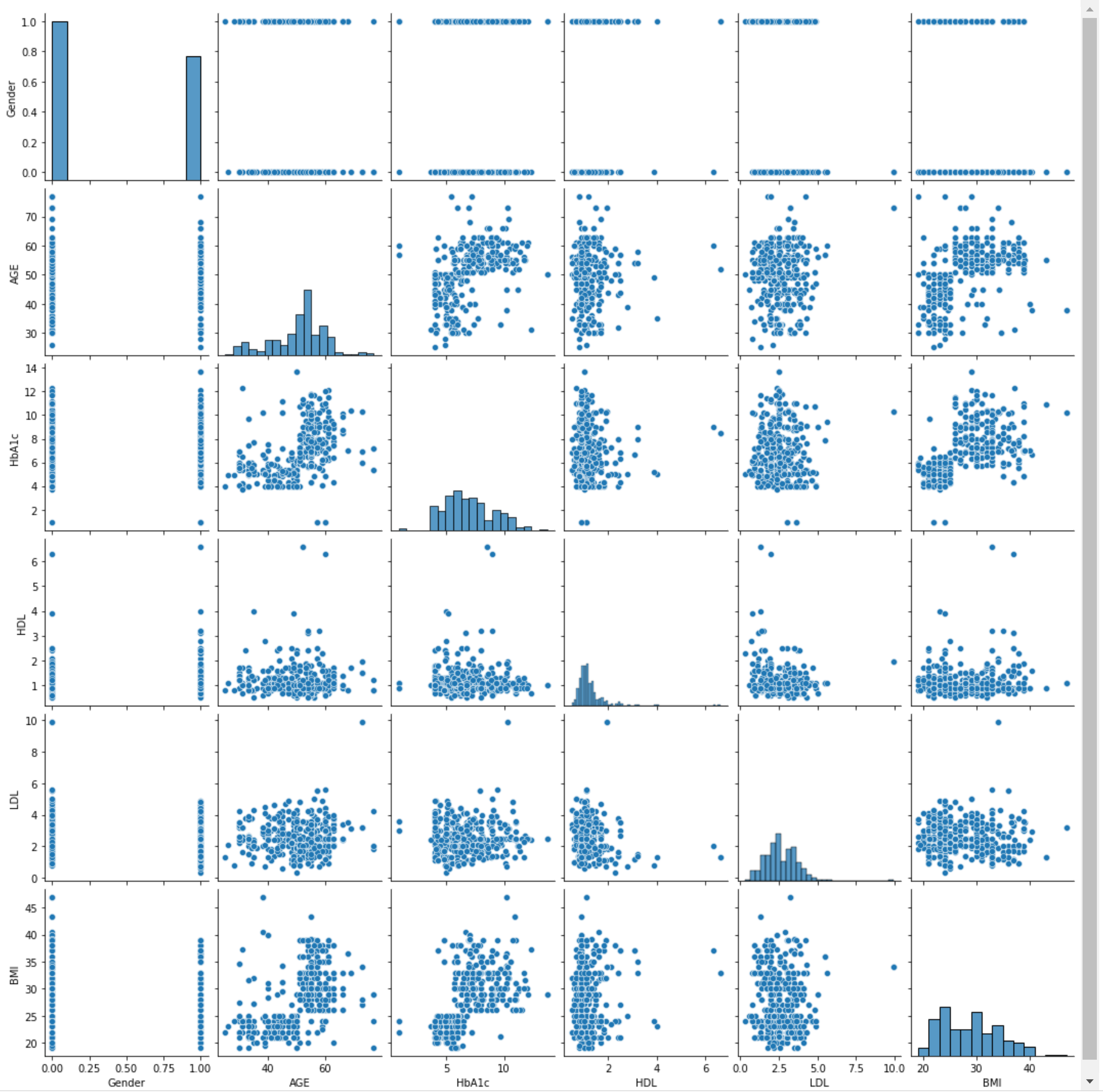
```
Training 0.9140401146131805
Testing 0.6866666666666666
```

```
cols=df.columns
```

```
cols
```

```
Index(['Gender', 'AGE', 'HbA1c', 'HDL', 'LDL', 'BMI', 'CLASS'], dtype='object')
```

```
sns.pairplot(df)
plt.show()
```


Gender
1.0
0.8
0.6
0.4
0.2
0.0
AGE
70
60
50
40
30
HbA1c
14
12
10
8
6
4
2
HDL
6
5
4
3
2
1
LDL
10
8
6
4
2
0
BMI
45
40
35
30
25
20
0.00
0.25
0.50
0.75
1.00
Gender
40
60
AGE
5
10
HbA1c
2
4
6
HDL
0.0
2.5
5.0
7.5
10.0
LDL
20
30
40
BMI