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

```
Choose Files pima.csv
```

- **pima.csv**(text/csv) - 23279 bytes, last modified: 7/8/2025 - 100% done

```
Saving pima.csv to pima (1).csv
```

```
import pandas
import numpy
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix
from sklearn import model_selection
from sklearn.preprocessing import StandardScaler


from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score


from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV


from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC



from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier


url="pima.csv"
names=['preg','plas','pres','skin','test','mass','pedi','age','class']
data=pandas.read_csv(url,names=names)


print(data)
```

```
     preg  plas  pres  skin  test  mass   pedi  age  class
0       6   148    72    35     0  33.6  0.627   50      1
1       1    85    66    29     0  26.6  0.351   31      0
2       8   183    64     0     0  23.3  0.672   32      1
3       1    89    66    23    94  28.1  0.167   21      0
4       0   137    40    35   168  43.1  2.288   33      1
..    ...   ...   ...   ...   ...   ...    ...  ...    ...
763    10   101    76    48   180  32.9  0.171   63      0
764     2   122    70    27     0  36.8  0.340   27      0
765     5   121    72    23   112  26.2  0.245   30      0
766     1   126    60     0     0  30.1  0.349   47      1
767     1    93    70    31     0  30.4  0.315   23      0

[768 rows x 9 columns]
```

```
print(data.dtypes)
print(data.shape)
```

```
preg       int64
plas       int64
pres       int64
skin       int64
test       int64
mass     float64
pedi     float64
age        int64
class      int64
dtype: object
(768, 9)
```

```
#checking for null values
data.apply(lambda x:sum(x.isnull()),axis=0)
```

	0
preg	0
plas	0
pres	0
skin	0
test	0
mass	0
pedi	0
age	0
class	0

dtype: int64

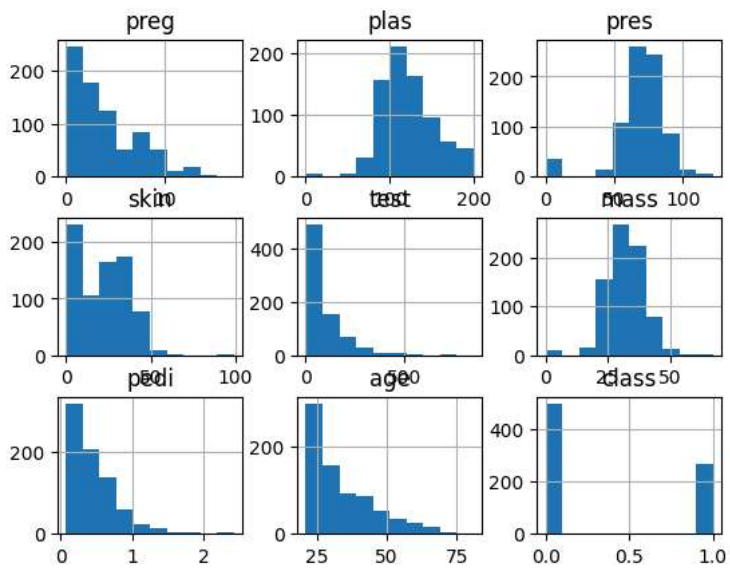
```
pandas.set_option('display.precision',2)
print(data.describe())
```

```
          preg    plas    pres    skin    test    mass    pedi     age   class
count  768.00  768.00  768.00  768.00  768.00  768.00  768.00  768.00  768.00
mean     3.85  120.89   69.11   20.54   79.80   31.99    0.47   33.24    0.35
std      3.37   31.97   19.36   15.95  115.24    7.88    0.33   11.76    0.48
min      0.00    0.00    0.00    0.00    0.00    0.00    0.08   21.00    0.00
25%      1.00   99.00   62.00    0.00    0.00   27.30    0.24   24.00    0.00
50%      3.00  117.00   72.00   23.00   30.50   32.00    0.37   29.00    0.00
75%      6.00  140.25   80.00   32.00  127.25   36.60    0.63   41.00    1.00
max     17.00  199.00  122.00   99.00  846.00   67.10    2.42   81.00    1.00
```

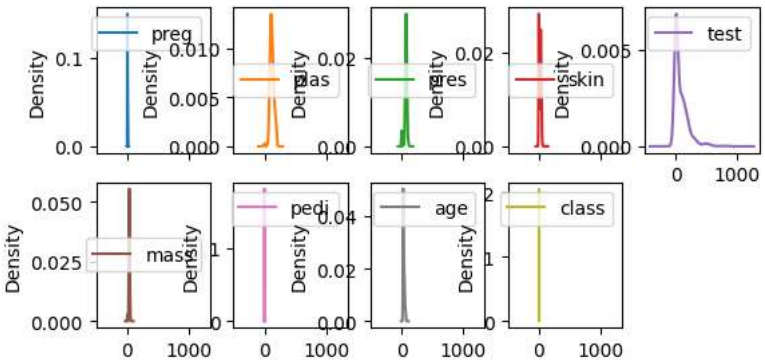
```
print("class size",data.groupby('class').size())
```

```
class size class
0    500
1    268
dtype: int64
```

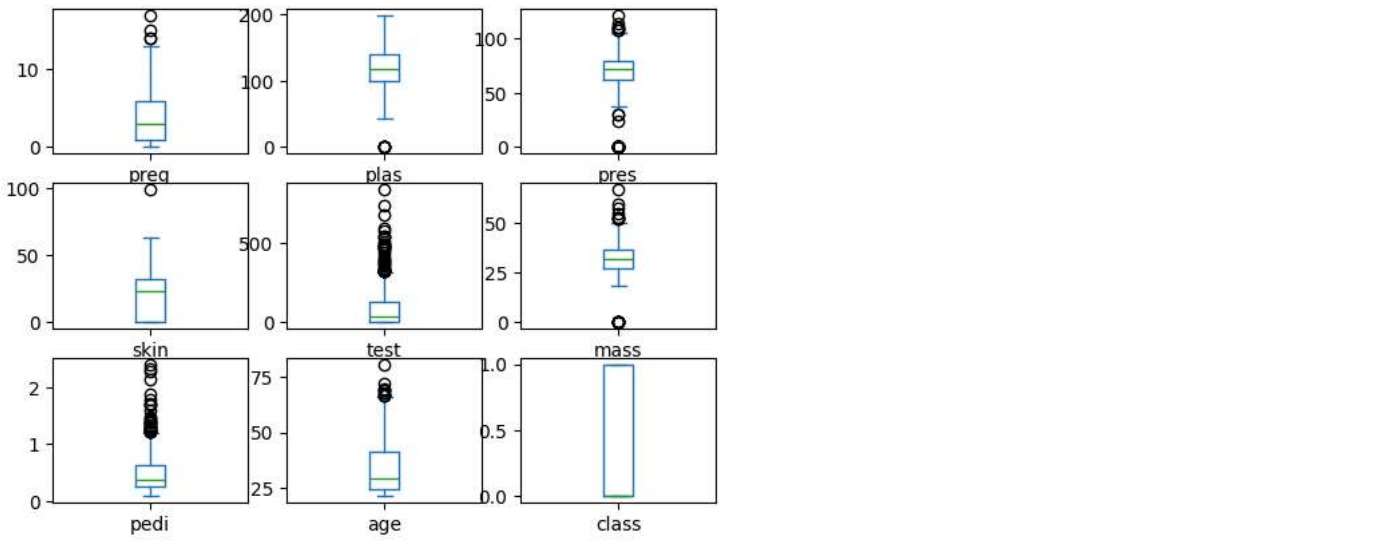
```
data.hist()
plt.show()
```



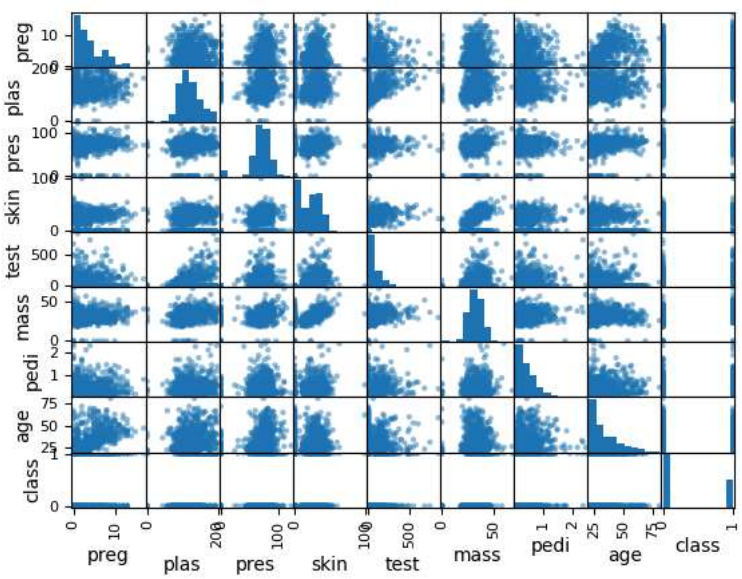

```
data.plot(kind='density',subplots=True,layout=(3,5))
plt.show()
```



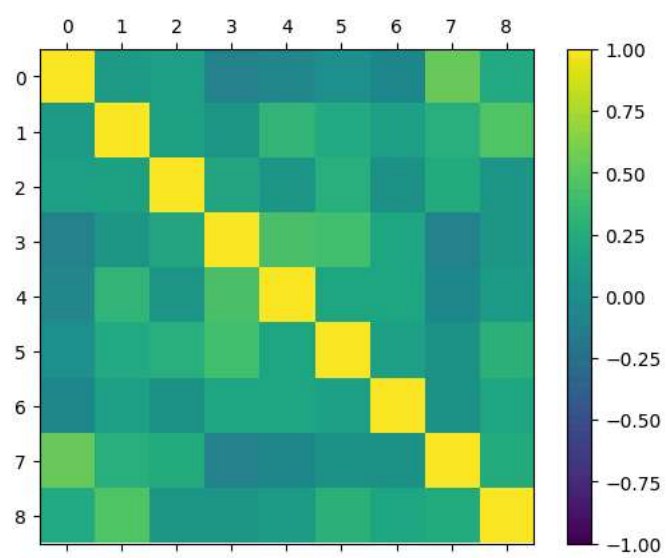

```
data.plot(kind='box',subplots=True,layout=(3,3))
plt.show()
```




```
scatter_matrix(data)
plt.show()
```




```
fig=plt.figure()
ax=fig.add_subplot(111)
cax=ax.matshow(data.corr(),vmin=-1,vmax=1)
fig.colorbar(cax)
plt.show()
```




```
array=data.values
X=array[:,0:8]
Y=array[:,8]


models=[]
models.append(('LR',LogisticRegression()))
models.append(('LDA',LinearDiscriminantAnalysis()))
models.append(('SVM',SVC()))
models.append(('CART',DecisionTreeClassifier()))
models.append(('KNN',KNeighborsClassifier()))
models.append(('NB',GaussianNB()))
results=[]
names=[]
scoring='accuracy'
test_size=0.33
seed=8
for name,model in models:
                        kfold=model_selection.ShuffleSplit(n_splits=10,test_size=test_size,random_state=seed)
                        cv_results=model_selection.cross_val_score(model,X, Y,cv=kfold,scoring=scoring)
                        results.append(cv_results)
                        names.append(name)
                        msg="%s:%f(%f)"%(name,cv_results.mean()*100,cv_results.std())
                        print(msg)
```

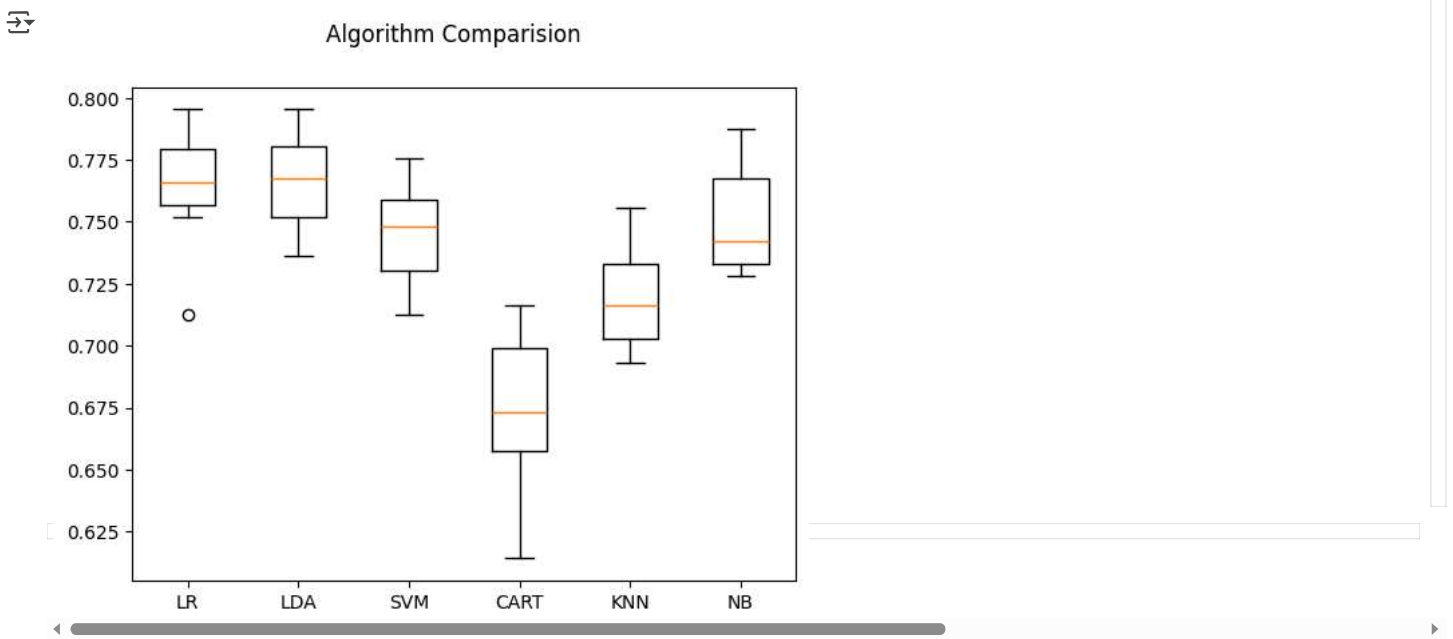
```
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
```

```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:465: ConvergenceWarning: lbfgs failed to converge (status=1)
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:465: ConvergenceWarning: lbfgs failed to converge (status=1)
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
LR:76.456693(0.021906)
LDA:76.614173(0.018144)
SVM:74.448819(0.018836)
CART:67.401575(0.029399)
KNN:71.850394(0.019307)
NB:75.039370(0.019935)
```

```
fig=plt.figure()
fig.suptitle('Algorithm Comparision')
ax=fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```

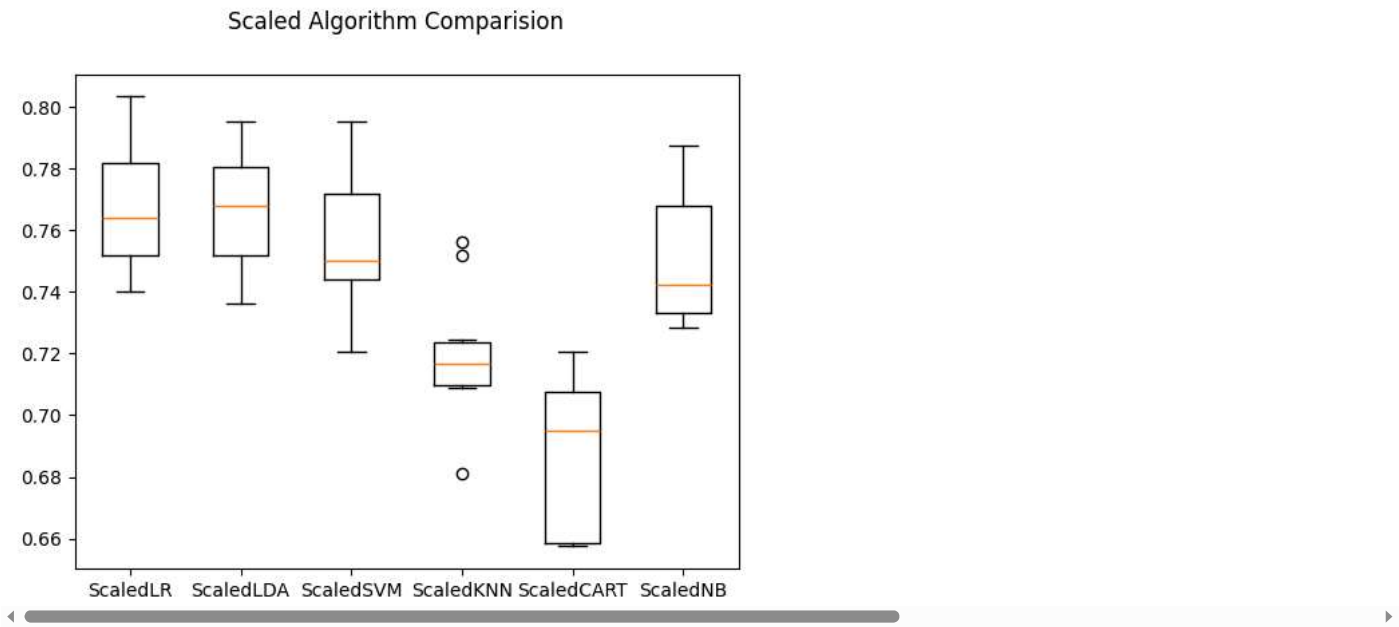



```
pipelines=[]
pipelines.append(('ScaledLR',Pipeline([('Scaler',StandardScaler()),('LR' ,LogisticRegression())])))
pipelines.append(('ScaledLDA',Pipeline([('Scaler',StandardScaler()),('LDA',LinearDiscriminantAnalysis())])))
pipelines.append(('ScaledSVM',Pipeline([('Scaler',StandardScaler()),('SVM',SVC())])))
pipelines.append(('ScaledKNN',Pipeline([('Scaler',StandardScaler()),('KNN',KNeighborsClassifier())])))
pipelines.append(('ScaledCART',Pipeline([('Scaler',StandardScaler()),('CART',DecisionTreeClassifier())])))
pipelines.append(('ScaledNB',Pipeline([('Scaler',StandardScaler()),('NB',GaussianNB())])))
results=[]
names=[]
scoring='accuracy'
test_size=0.33
seed=8
for name,model in pipelines:
                kfold=model_selection.ShuffleSplit(n_splits=10,test_size=test_size,random_state=seed)
                cv_results=model_selection.cross_val_score(model,X, Y,cv=kfold,scoring=scoring)
                results.append(cv_results)
                names.append(name)
```

```
    msg="%s:%f(%f)"%(name,cv_results.mean()*100,cv_results.std())
    print(msg)
```

```
ScaledLR:76.692913(0.019110)
ScaledLDA:76.614173(0.018144)
ScaledSVM:75.472441(0.022205)
ScaledKNN:71.968504(0.020518)
ScaledCART:68.740157(0.024850)
ScaledNB:75.039370(0.019935)
```

```
fig=plt.figure()
fig.suptitle('Scaled Algorithm Comparision')
ax=fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```




```
Scaler=StandardScaler().fit(X)
rescaledX=Scaler.transform(X)
neighbors=[1,3,5,7,9,11,13,15,17,19,21]
model=KNeighborsClassifier()
param_grid=dict(n_neighbors=neighbors)
kfold=model_selection.ShuffleSplit(n_splits=10,test_size=test_size,random_state=seed)
grid=GridSearchCV(estimator=model,param_grid=param_grid,scoring=scoring,cv=kfold)
grid_result=grid.fit(rescaledX,Y)
print("Best:%f using %s"%(grid_result.best_score_,grid_result.best_params_))
```

```
Best:0.743701 using {'n_neighbors': 17}
```

```
Scaler=StandardScaler().fit(X)
rescaledX=Scaler.transform(X)
c_values=[0.1,0.3,0.5,0.7,0.9,1.1,1.3,1.5,1.7,3.0]
kernel_values=['linear','poly','rbf','sigmoid']
param_grid=dict(C=c_values,kernel=kernel_values)
model=SVC()
kfold=model_selection.ShuffleSplit(n_splits=10,test_size=test_size,random_state=seed)
grid=GridSearchCV(estimator=model,param_grid=param_grid,scoring=scoring,cv=kfold)
grid_result=grid.fit(rescaledX,Y)
print("Best%f using %s:"%(grid_result.best_score_,grid_result.best_params_))
```

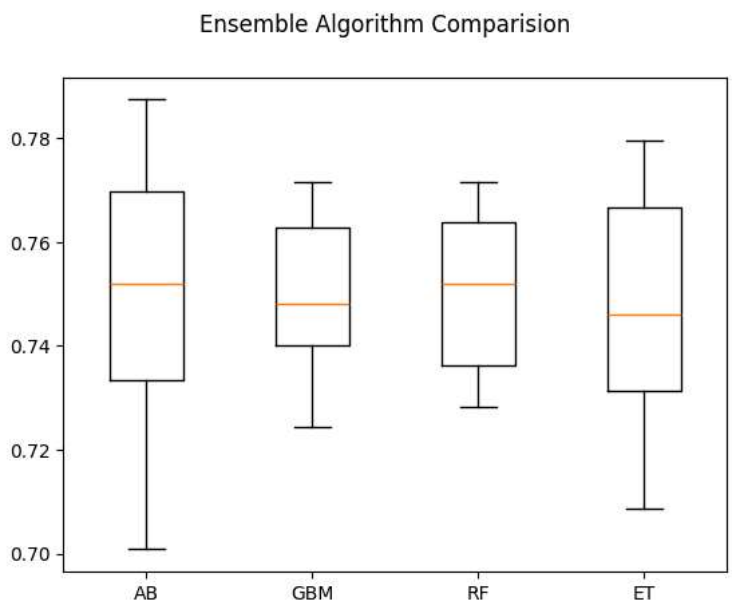
```
Best0.768110 using {'C': 0.3, 'kernel': 'linear'}:
```

```
ensembles=[]
ensembles.append(('AB',AdaBoostClassifier()))
ensembles.append(('GBM',GradientBoostingClassifier()))
ensembles.append(('RF',RandomForestClassifier()))
ensembles.append(('ET',ExtraTreesClassifier()))
results=[]
names=[]
scoring='accuracy'
```

```
test_size=0.33
seed=8
for name,model in ensembles:
                        kfold=model_selection.ShuffleSplit(n_splits=10,test_size=test_size,random_state=seed)
                        cv_results=model_selection.cross_val_score(model,X, Y,cv=kfold,scoring=scoring)
                        results.append(cv_results)
                        names.append(name)
                        msg="%s:%f(%f)"%(name,cv_results.mean()*100,cv_results.std())
                        print(msg)
```

```
AB:74.842520(0.025481)
GBM:75.000000(0.013892)
RF:75.039370(0.016156)
ET:74.803150(0.021420)
```

```
fig=plt.figure()
fig.suptitle("Ensemble Algorithm Comparision")
ax=fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```




```
Scaler=StandardScaler().fit(X)
rescaledX=Scaler.transform(X)
model=LogisticRegression()
model.fit(rescaledX,Y)
```

```
LogisticRegression
LogisticRegression()
```

```
rescaledValidationX=Scaler.transform(X)
predictions=model.predict(rescaledValidationX)
```

```
print(classification_report(Y,predictions))
```

```
              precision    recall  f1-score   support

         0.0       0.80      0.89      0.84       500
         1.0       0.74      0.58      0.65       268

    accuracy                           0.78       768
   macro avg       0.77      0.74      0.75       768
weighted avg       0.78      0.78      0.78       768
```

```
print(accuracy_score(Y,predictions))
```

```
0.7838541666666666
```

```
print(confusion_matrix(Y,predictions))
```

```
[[446  54]
 [112 156]]
```