In [2]:
```
 # AdaBoost Classification
from pandas import read_csv
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import AdaBoostClassifier

filename = 'D:\\Dataset\pima-indians-diabetes.csv'
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class
dataframe = read_csv(filename, names=names)
array = dataframe.values
X = array[:,0:8]
Y = array[:,8]

num_trees = 30
kfold = KFold(n_splits=10)
model = AdaBoostClassifier(n_estimators=num_trees)
results = cross_val_score(model, X, Y, cv=kfold)
print(results.mean()*100)
```

```
75.39473684210527
```

In [4]:
```
 # Stochastic Gradient Boosting Classification
from pandas import read_csv
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import GradientBoostingClassifier

filename = 'D:\\Dataset\pima-indians-diabetes.csv'
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class
dataframe = read_csv(filename, names=names)
array = dataframe.values
X = array[:,0:8]
Y = array[:,8]

num_trees = 80
kfold = KFold(n_splits=10)
model = GradientBoostingClassifier(n_estimators=num_trees)
results = cross_val_score(model, X, Y, cv=kfold)
print(results.mean()*100)
```

```
77.46411483253588
```

In [19]:
```
# Voting Ensemble for Classification
from pandas import read_csv
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import VotingClassifier

import warnings
warnings.filterwarnings("ignore")

filename = 'D:\\Dataset\pima-indians-diabetes.csv'
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class
dataframe = read_csv(filename, names=names)
array = dataframe.values
X = array[:,0:8]
Y = array[:,8]

kfold = KFold(n_splits=2)

# create the sub models
estimators = []
model1 = LogisticRegression()
estimators.append(('LR', model1))
model2 = DecisionTreeClassifier()
estimators.append(('CART', model2))
model3 = SVC()
estimators.append(('svm', model3))

# create the ensemble model
#ensemble = VotingClassifier(estimators)
#results = cross_val_score(ensemble, X, Y, cv=kfold)
#print(results)
#print(results.mean()*100)


test_size = 0.33
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=test_size)
#model = LogisticRegression

model1.fit(X_train, Y_train)
result_model1 = model1.score(X_test, Y_test)
print(result_model1*100.0)

model2.fit(X_train, Y_train)
result_model2 = model2.score(X_test, Y_test)
print(result_model2*100.0)

model3.fit(X_train, Y_train)
result_model3 = model3.score(X_test, Y_test)
print(result_model1*100.0)

ensemble = VotingClassifier(estimators,voting='hard')
```

```
ensemble.fit(X_train, Y_train)
result = ensemble.score(X_test, Y_test)
print(result*100.0)
```

```
77.16535433070865
72.83464566929135
77.16535433070865
76.37795275590551
```

In [20]:

```
# KNN Classification with default parameters
from pandas import read_csv
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.neighbors import KNeighborsClassifier

filename = "D:\\Dataset\pima-indians-diabetes.csv"
names = ["preg", "plas", "pres", "skin", "test", "mass", "pedi", "age", "class
dataframe = read_csv(filename, names=names)
array = dataframe.values
X = array[:,0:8]
Y = array[:,8]

num_folds = 10
kfold = KFold(n_splits=10)
model = KNeighborsClassifier()
results = cross_val_score(model, X, Y, cv=kfold)
print(results.mean()*100)
```

```
72.6555023923445
```

In [22]:
```
# KNN Classification with n_neighbors parameter tunning
import numpy
from pandas import read_csv
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV

filename = "D:\\Dataset\pima-indians-diabetes.csv"
names = ["preg", "plas", "pres", "skin", "test", "mass", "pedi", "age", "class
dataframe = read_csv(filename, names=names)
array = dataframe.values
X = array[:,0:8]
Y = array[:,8]

num_folds = 10


neighbors = [1,3,5,7,9,11,13,15,17,19,21,27,33]
param_grid = dict(n_neighbors=neighbors)
model = KNeighborsClassifier()
kfold = KFold(n_splits=num_folds)
grid = GridSearchCV(estimator=model, param_grid=param_grid, scoring='accuracy'
grid_result = grid.fit(X, Y)

print("Best:", grid_result.best_score_, grid_result.best_params_)
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print(mean, param)
```

```
Best: 0.7592105263157894 {'n_neighbors': 21}
0.6770505809979495 {'n_neighbors': 1}
0.7056049213943951 {'n_neighbors': 3}
0.7265550239234451 {'n_neighbors': 5}
0.7448051948051948 {'n_neighbors': 7}
0.7396103896103897 {'n_neighbors': 9}
0.7408407382091592 {'n_neighbors': 11}
0.7460526315789473 {'n_neighbors': 13}
0.7473684210526315 {'n_neighbors': 15}
0.7565276828434724 {'n_neighbors': 17}
0.7513328776486671 {'n_neighbors': 19}
0.7592105263157894 {'n_neighbors': 21}
0.7435235816814765 {'n_neighbors': 27}
0.7448393711551606 {'n_neighbors': 33}
```

In [23]:
```
# Ridge Regression (L2 Regularization)
from pandas import read_csv
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import Ridge


filename = 'D:\\Dataset\housing.csv'
names = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX
dataframe = read_csv(filename, delim_whitespace=True, names=names)
array = dataframe.values
X = array[:,0:13]
Y = array[:,13]

num_folds = 10
kfold = KFold(n_splits=10)
model = Ridge(alpha=1) #L2 regularization term add here in ridge regression
scoring = 'neg_mean_squared_error'
results = cross_val_score(model, X, Y, cv=kfold, scoring=scoring)
print(results.mean())
```

```
-34.07824620925927
```

In [24]:
```
#Grid Search for Ridge regression (L2 Regularization) Algorithm Tuning
import numpy
from pandas import read_csv
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV

filename = 'D:\\Dataset\housing.csv'
names = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX
dataframe = read_csv(filename, delim_whitespace=True, names=names)
array = dataframe.values
X = array[:,0:13]
Y = array[:,13]

alphas = numpy.array([1,0.1,0.01,0.001,0.0001,0])
param_grid = dict(alpha=alphas) #alpha is tuning here
model = Ridge()
grid = GridSearchCV(estimator=model, param_grid=param_grid)
grid.fit(X, Y)

for i in ['mean_test_score', 'std_test_score','rank_test_score']:
    print(i," : ",grid.cv_results_[i]) #the scores for all the scorers are ava
print(grid.best_score_) #Score of best_estimator on the left out data.
print(grid.best_estimator_.alpha) #To observe the alpha value from best_estima
```

```
mean_test_score  :  [0.38921758 0.36095293 0.35414275 0.35336374 0.35328472
0.35327592]
std_test_score  :  [0.36347007 0.37465033 0.37637099 0.37654812 0.37656587
0.37656784]
rank_test_score  :  [1 2 3 4 5 6]
0.3892175824102412
1.0
```

In [ ]: