```
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV,KFold
from keras.wrappers.scikit_learn import KerasClassifier
from keras.layers import Dropout

data=pd.read_csv('Downloads/pima-indians-diabetes.data.csv')
data
```

	6	148	72	35	0	33.6	0.627	50	1
0	1	85	66	29	0	26.6	0.351	31	0
1	8	183	64	0	0	23.3	0.672	32	1
2	1	89	66	23	94	28.1	0.167	21	0
3	0	137	40	35	168	43.1	2.288	33	1
4	5	116	74	0	0	25.6	0.201	30	0
...	...	...	...	...	...	...	...	...	...
762	10	101	76	48	180	32.9	0.171	63	0
763	2	122	70	27	0	36.8	0.340	27	0
764	5	121	72	23	112	26.2	0.245	30	0
765	1	126	60	0	0	30.1	0.349	47	1
766	1	93	70	31	0	30.4	0.315	23	0

```
x=data.iloc[:,0:8]
y=data.iloc[:,8]

def model():
    model=Sequential()
    model.add(Dense(12,kernel_initializer='uniform',activation='relu'))
    model.add(Dense(8,kernel_initializer='uniform',activation='relu'))
    model.add(Dense(1,kernel_initializer='uniform',activation='sigmoid'))
    model.compile(loss='binary_crossentropy',optimizer='adam',metrics='accuracy')
    return model

model=KerasClassifier(build_in=model,verbose=0)
batch_size=30
epochs=10
param_grid=dict(batch_size=batch_size,epochs=epochs)
gsv=GridSearchCV(estimator=model,param_grid=param_grid,cv=KFold(),verbose=10)
gsvresult=gsv.fit_transform(x,y)

mean=gsvresult.csv_result_['mean_test_score']
```

```
std=gsvresult.csv_result_['std_test_score']
params=gsvresult.csv_result_['params_test_score']
for mean,std,params in zip(mean,std,params):
    print('{},{}with:{}'.format(mean,std,params))

def model():
    model=Sequential()
    model.add(Dense(12,kernel_initializer='uniform',activation='relu'))
    model.add(Dense(8,kernel_initializer='uniform',activation='relu'))
    model.add(Dense(1,kernel_initializer='uniform',activation='sigmoid'))
    model.compile(loss='binary_crossentropy',optimizer='adam',metrics='accuracy')
    return model

model=KerasClassifier(build_in=model,verbose=0,batch_size=30,epochs=10)
learning_rate=[0.001,0.01,0.1]
dropout_rate=[0.1,0.2,0.0]
param_grid=dict(learning_rate=learning_rate,dropout_rate=dropout_rate)
gsv=GridSearchCV(estimator=model,param_grid=param_grid,cv=KFold(),verbose=10)
gsvresult=gsv.fit_transform(x,y)

mean=gsvresult.csv_result_['mean_test_score']
std=gsvresult.csv_result_['std_test_score']
params=gsvresult.csv_result_['params_test_score']
for mean,std,params in zip(mean,std,params):
    print('{},{}with:{}'.format(mean,std,params))

def model():
    model=Sequential()
    model.add(Dense(12,kernel_initializer='uniform',activation='relu'))
    model.add(Dense(8,kernel_initializer='uniform',activation='relu'))
    model.add(Dense(1,kernel_initializer='uniform',activation='sigmoid'))
    model.compile(loss='binary_crossentropy',optimizer='adam',metrics='accuracy')
    return model

model=KerasClassifier(build_in=model,verbose=0,batch_size=30,epochs=10)
activation_function=['softmax','tanh','relu','linear']
init=['uniform','normal','zero']
param_grid=dict(activation_function=activation_function,init=init)
gsv=GridSearchCV(estimator=model,param_grid=param_grid,cv=KFold(),verbose=10)
gsvresult=gsv.fit_transform(x,y)

mean=gsvresult.csv_result_['mean_test_score']
std=gsvresult.csv_result_['std_test_score']
params=gsvresult.csv_result_['params_test_score']
```

```
for mean,std,params in zip(mean,std,params):
    print('{},{}with:{}'.format(mean,std,params))

def model():
    model=Sequential()
    model.add(Dense(12,kernel_initializer='uniform',activation='relu'))
    model.add(Dense(8,kernel_initializer='uniform',activation='relu'))
    model.add(Dense(1,kernel_initializer='uniform',activation='sigmoid'))
    model.compile(loss='binary_crossentropy',optimizer='adam',metrics='accuracy')
    return model

model=KerasClassifier(build_in=model,verbose=0,batch_size=30,epochs=10)
neuron1=[10,20,30]
neuron2=[2,4,8]
param_grid=dict(neuron1=neuron1,neuron2=neuron2)
gsv=GridSearchCV(estimator=model,param_grid=param_grid,cv=KFold(),verbose=10)
gsvresult=gsv.fit_transform(x,y)

mean=gsvresult.csv_result_['mean_test_score']
std=gsvresult.csv_result_['std_test_score']
params=gsvresult.csv_result_['params_test_score']
for mean,std,params in zip(mean,std,params):
    print('{},{}with:{}'.format(mean,std,params))

def model():
    model=Sequential()
    model.add(Dense(12,kernel_initializer='uniform',activation='relu'))
    model.add(Dense(8,kernel_initializer='uniform',activation='relu'))
    model.add(Dense(1,kernel_initializer='uniform',activation='sigmoid'))
    model.compile(loss='binary_crossentropy',optimizer='adam',metrics='accuracy')
    return model

model=KerasClassifier(build_in=model,verbose=0,batch_size=30,epochs=10)
model.fit(x,y)
predict=model.predict(x)
print(accuracy_score(y,predict))

def model():
    model=Sequential()
    model.add(Dense(12,kernel_initializer='uniform',activation='relu'))
    model.add(Dense(8,kernel_initializer='uniform',activation='relu'))
    model.add(Dense(1,kernel_initializer='uniform',activation='sigmoid'))
    model.compile(loss='binary_crossentropy',optimizer='adam',metrics='accuracy')
    return model
```

```
model=KerasClassifier(build_in=model,verbose=0,batch_size=30,epochs=10)
learning_rate=[0.001,0.01,0.1]
dropout_rate=[0,0.1,0.2]
activation_function=['softmax','tanh','relu','linear']
init=['normal','uniform','zero']
neuron1=[10,20,30]
neuron2=[2,4,8]
param_grid=dict(learning_rate=learning_rate,dropout_rate=dropout_rate,activation_function=act
ivation_function,init=init,neuron1=neuron1,neuron2=neuron2)
gsv=GridSearchCV(estimator=model,param_grid=param_grid,cv=KFold(),verbose=10)
gsvresult=gsv.fit_transform(x,y)

mean=gsvresult.csv_result_['mean_test_score']
std=gsvresult.csv_result_['std_test_score']
params=gsvresult.csv_result_['params_test_score']
for mean,std,params in zip(mean,std,params):
    print('{},{}with:{}'.format(mean,std,params))
```