Exercício 5 - MO444 - Aprendizado de máquina e reconhecimento de padrões

Renato Lopes Moura - 163050

1 Código

```
import numpy as np
import pandas as pd
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.svm import SVR
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_absolute_error
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn import linear_model
from sklearn.neural_network import MLPRegressor

##############################################################################
#Carregando o conjunto de dados de treino do csv usando o pandas
data = pd.read_csv('train.csv', header=None)

#Separando os valores a serem estimados do resto dos dados
train_Y = data.pop(0)

#Separando os dados numericos dos categoricos
numericos = data.select_dtypes(include=['int64']).columns
categoricos = data.select_dtypes(include=['object']).columns

#Exibindo quais colunas contem cada tipo de dado
print "numericos: "+str(numericos.values)
print "categoricos: "+str(categoricos.values)

#Convertendo os dados categoricos para labels numericos
for column in categoricos:
        data[column] = pd.Categorical(data[column]).codes

##############################################################################
#Eliminando as colunas de dados numericos com variancia menor do que 1
numericos_new = []

for column in numericos:
        if data[column].var() < 1:
                data.pop(column)
        else:
                numericos_new.append(column)

numericos = pd.Index(numericos_new)
print "numericos restantes: "+str(numericos.values)

numericos_array = data[numericos].values

##############################################################################
#Juntando os dados de treino numericos e categoricos
train_X = np.concatenate((numericos_array, data[categoricos].values), axis=1)

#Redividindo o conjunto de treino para determinar o melhor regressor
```

```
X_train, X_test, y_train, y_test = train_test_split(train_X, train_Y, test_size=0.2)

################################################################################
#Aplicacao do SVM regressor
svm_parameters = {'C':[2**(-5), 2**(0), 2**(5), 2**(10)],
 'gamma':[2**(-15), 2**(-10), 2**(-5), 2**(0), 2**(5)]}

grid_svr = GridSearchCV(SVR(kernel='rbf'), svm_parameters, cv=3,
        scoring='neg_mean_absolute_error')
grid_svr.fit(X_train, y_train)

svr = SVR(C=grid_svr.best_params_['C'], gamma=grid_svr.best_params_['gamma'], kernel='rbf')
svr.fit(X_train, y_train)

y_pred = svr.predict(X_test)

print "O MAE do svr foi "+str(mean_absolute_error(y_test, y_pred))

################################################################################
#Aplicacao do Gradient Boosting Regression
gbr_parameters = {'n_estimators':[30,70,100],'learning_rate':[0.1,0.05],'max_depth':[5]}

grid_gbr = GridSearchCV(GradientBoostingRegressor(), gbr_parameters, cv=3,
        scoring='neg_mean_absolute_error')
grid_gbr.fit(X_train, y_train)

gbr = GradientBoostingRegressor(n_estimators=grid_gbr.best_params_['n_estimators'],
         learning_rate=grid_gbr.best_params_['learning_rate'],
          max_depth=grid_gbr.best_params_['max_depth'])
gbr.fit(X_train, y_train)

y_pred = gbr.predict(X_test)

print "O MAE do gbr foi "+str(mean_absolute_error(y_test, y_pred))

################################################################################
#Aplicacao do Bayesian Regression

bayes = linear_model.BayesianRidge()
bayes.fit(X_train, y_train)

y_pred = bayes.predict(X_test)

print "O MAE do bayes foi "+str(mean_absolute_error(y_test, y_pred))

################################################################################
#Aplicacao do Neural Net Regressor
nn_parameters = {'hidden_layer_sizes':[10,20,30,40]}

grid_nn = GridSearchCV(MLPRegressor(solver='lbfgs'), nn_parameters, cv=3)
grid_nn.fit(X_train, y_train)

nnet = MLPRegressor(hidden_layer_sizes=grid_nn.best_params_['hidden_layer_sizes'],
         solver='lbfgs')
nnet.fit(X_train, y_train)

y_pred = nnet.predict(X_test)

print "O MAE da nnet foi "+str(mean_absolute_error(y_test, y_pred))

################################################################################
#Carregando o conjunto de dados de teste do csv usando o pandas
```

```
data_test = pd.read_csv('test.csv', header=None)

#Convertendo os dados categoricos para labels numericos
for column in categoricos:
        data_test[column-1] = pd.Categorical(data_test[column-1]).codes

numericos_array_test = data_test[numericos-1].values

###############################################################################
#Juntando os dados de teste numericos e categoricos
test_X = np.concatenate((numericos_array_test, data_test[categoricos].values), axis=1)

###############################################################################
#Aplicacao do melhor regressor
svr = SVR(C=grid_svr.best_params_['C'], gamma=grid_svr.best_params_['gamma'], kernel='rbf')

#Ajustando sobre todos os dados de treino
svr.fit(train_X, train_Y)

y_pred = svr.predict(test_X)

np.savetxt("resultados.csv", y_pred)
```

2 Outputs

```
numericos: [ 1 2 3 10 13 14 18 19 21 23 24 25 26 27 31 32]
categoricos: [ 4 5 6 7 8 9 11 12 15 16 17 20 22 28 29 30]
numericos restantes: [ 1 2 3 10 13 18 19 21 24 26 27 31 32]

O MAE do svr foi 2.50887264402
O MAE do gbr foi 2.64994659664
O MAE do bayes foi 2.68360764544
O MAE da nnet foi 2.71065370945
```