# Exercício 3

Mateus Augusto Bellomo Agrello Ruivo RA:147338

19 de outubro de 2016

## 1 Questão 1

```
def myZscore(a):
    mu = np.mean(a)
    sigma = np.std(a)
    return (a-mu)/sigma

def imputationWithMean(X):
    col_mean = stats.nanmean(X, axis=0)
    idx = np.where(np.isnan(X))
    X[idx] = np.take(col_mean, idx[1])
    return X

def getIndexesColsSTDEVZero(X):
    colsToDelete = []
    idx = 0
    for col in X.T:
        sigma = np.std(col)
        if sigma == 0:
            colsToDelete.append(idx)
        idx = idx+1
    return colsToDelete

X = genfromtxt(’secom.data’, delimiter=’ ’)
y = genfromtxt(’secom_labels.data’, delimiter=’ ’, usecols=(0))


# do imputation in columns with mean
X = imputationWithMean(X)
# remove columns with zero standard deviation
X = np.delete(X, getIndexesColsSTDEVZero(X), 1)
# apply Z-score in columns
X = np.apply_along_axis(myZscore, 0, X)
```

## 2 Questão 2

```
pca = PCA(n_components=0.79)
pca.fit(X)
XPCA = pca.transform(X)

# K neighbors method
parameters = {’n_neighbors’:[ 1, 5, 11, 15, 21, 25 ]}
knn = KNeighborsClassifier()
scores = cross_val_score(GridSearchCV(knn, parameters, cv=3, scoring=’accuracy’), XPCA, y, cv=5)
print ’+-----------------------------------------------+’
print "K neighbors"
print scores
print "accuracy: %.2f%%" % (100 * np.mean(scores))
print ’+-----------------------------------------------+’
```

## 3 Questão 3

```
# SVM with RBF kernel method
parameters = {’C’: [2**-5, 2**-2, 2**0, 2**2, 2**5],
              ’gamma’: [2**-15, 2**-10, 2**-5, 2**0, 2**5]}
svc = SVC(kernel=’rbf’)
scores = cross_val_score(GridSearchCV(svc, parameters, cv=3, scoring=’accuracy’), X, y, cv=5)
print ’+-----------------------------------------------+’
print "SVM with RBF kernel"
print scores
print "accuracy: %.2f%%" % (100 * np.mean(scores))
print ’+-----------------------------------------------+’
```

## 4 Questão 4

```
# Neural Network method
parameters = {’hidden_layer_sizes’: [10, 20, 30, 40]}
mlp = MLPClassifier(max_iter=400)
scores = cross_val_score(GridSearchCV(mlp, parameters, cv=3, scoring=’accuracy’), X, y, cv=5)
print ’+-----------------------------------------------+’
print "Neural Network"
print scores
print "accuracy: %.2f%%" % (100 * np.mean(scores))
print ’+-----------------------------------------------+’
```

## 5 Questão 5

```
# Random Forest method
parameters = {’n_estimators’: [100, 200, 300, 400],
              ’max_features’: [10, 15, 20, 25]}
rfc = RandomForestClassifier()
scores = cross_val_score(GridSearchCV(rfc, parameters, cv=3, scoring=’accuracy’), X, y, cv=5)
print ’+-----------------------------------------------+’
print "Random Forest"
print scores
print "accuracy: %.2f%%" % (100 * np.mean(scores))
print ’+-----------------------------------------------+’
```

## 6 Questão 6

```
# Gradient Boosting method
parameters = {’n_estimators’: [30, 70, 100],
              ’learning_rate’: [0.1, 0.05]}
gbc = GradientBoostingClassifier(max_depth=5)
scores = cross_val_score(GridSearchCV(gbc, parameters, cv=3, scoring=’accuracy’), X, y, cv=5)
print ’+-----------------------------------------------+’
print "Gradient Boosting"
print scores
print "accuracy: %.2f%%" % (100 * np.mean(scores))
print ’+-----------------------------------------------+’
```

## 7 Questão 8

```
+-----------------------------------------------+
K neighbors
```

```
[ 0.92356688 0.93312102 0.93312102 0.93290735 0.93589744]
accuracy: 93.17%
+-----------------------------------------------+
+-----------------------------------------------+
SVM with RBF kernel
[ 0.93312102 0.93312102 0.93312102 0.93290735 0.93589744]
accuracy: 93.36%
+-----------------------------------------------+
+-----------------------------------------------+
Neural Network
[ 0.48726115 0.83121019 0.91719745 0.77316294 0.91025641]
accuracy: 78.38%
+-----------------------------------------------+
+-----------------------------------------------+
Random Forest
[ 0.92993631 0.93312102 0.93312102 0.93290735 0.93589744]
accuracy: 93.30%
+-----------------------------------------------+
+-----------------------------------------------+
Gradient Boosting
[ 0.53184713 0.91401274 0.92993631 0.9201278 0.92948718]
accuracy: 84.51%
+-----------------------------------------------+
```

## 8 Referências

[1] http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
[2] http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
[3] http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html
[4] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
[5] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html
[6] http://scikit-learn.org/stable/modules/cross_validation.html