Preliminares

Numero de imagens: 339
Matriz que armazena as imagens: features

Abaixo o “shape” da matriz:
features.shape
(339, 360880)

São 339 imagens, sendo:

Numero de imagens	classe
113	0
113	1
113	2

Cada imagem tem 36.0880 features (***evito, aqui, a palavra atributo p/ não conflitar com os atributos dos objetos do pacote scikit***)

Matriz que indica as classes de cada imagem: y

Abaixo o shape da matriz de classes:
y.shape
(339,)

Isto é, para cada imagem na matriz features, existe um elemento na matriz y (no indice correspondente) indicando a classe a qual aquela imagem pertence. Esta matriz tem 113 zeros, 113 numeros 1 e 113 numeros 2, sequencialmente.

Codigo:
```
1: anova_sk = SelectKBest(f_classif, k=16818)
2: anova_sk.fit(features, y)
3: reduced_features = anova_sk.transform(features)
4: clf = svm.SVC(C=1, kernel='linear')
5: kfold = cross_validation.KFold(len(reduced_features), n_folds=10)
6: scores = cross_validation.cross_val_score(clf, reduced_features, y, cv=10, n_jobs=-1)
7: print (scores.mean())
```

1) Criação de um objeto anova:

anova_sk = SelectKBest(f_classif, k=16818)

Apresentação do objeto:
print (anova_sk)
SelectKBest(k=16818, score_func=<function f_classif at 0x7f8e96426848>)

Neste ponto anova_sk é apenas um objeto preparado para selecionar as Kbest 16,818 features de uma distribuição qualquer (ainda não foi feita esta atribuição)

2) Atribuição da distribuição e as respectivas classes ao objeto anova:

anova_sk.fit(features, y)
Neste ponto o obejto passa a ter 2 **atributos** que não tinha até aqui (calculados pelo método fit):

Atributo 1 : pvalues_ = Array , também com o mesmo shape das imagens, cada elemento deste array representa o "valor-p" para a feature correspondente em cada imagem
Shape de pvalues_ : (360880,)

Atributo 2: scores_ = Array que tem o mesmo shape das imagens da distribuição atribuida, este array contem a pontuação para cada feature das imagens.
Shape de scores_ : (360880,)

Importante notar que, embora o objeto anova tenha sido construido para selecionar as 16,818 Kbest features, tanto o **Atributo 1** (que representa os pvalues) quanto o **Atributo 2** (que representa os scores) contem informações a respeito de todas as features das imagens, ou seja 360880 elementos, cada um representando, respectivamente os pvalues e a "pontuação" de cada feature.

Estes **atributos** scores_ e pvalues_, para a classificação, poderiam ser ignorados pelo usuário (o desenvolvedor), a não ser para extrair informaçoes adicionais, como é o caso aqui.

3) A tranformação dos dados

Até o momento, os dados originais não sofreram nenhuma modificação, até aqui temos um objeto do tipo SelectKBest que conhece estes dados, e tem 2 arrays auxiliares em que guarda o calculo dos p-values e dos scores para cada feature (colunas das imagens).

Agora é o momento de se fazer a tranformação dos dados, isto é, vamos extrarir as K melhores features dos dados originais, como, ao criar o objeto SelectKBest passamos como parametro K=16,818, então exatamente este número de features será selecionado:

reduced_features = anova_sk.transform(features)

Shape de reduced_features = (339, 16818)

Agora a nova distribuição, contem 339 imagens, mas cada imagem contem 16,818 features (as melhores entre as 360880 dos dados originais), O método transform apenas utilizou as informações contidas nos **atributos** scores_ e/ou p-values para fazer esta seleção.

4) Criando o classificador

clf = svm.SVC(C=1, kernel='linear')
Como na criação do objeto anova, aqui criamos apenas um objeto, sem nehuma atribuição de dados, o parametro C representa o tipo do SVM para a função de minimização de erro, se é que entendi direito, mais neste link: https://www.statsoft.com/Textbook/Support-Vector-Machines#Classification SVM

5) Criando o objeto para fazer a cross-validation

kfold = cross_validation.KFold(len(reduced_features), n_folds=10)
Novamente, esta linha apenas cria o objeto, aqui passo como parametro a quantidade de elementos da distribuição, lembrando que o shape de reduced_features = (339, 16818), então len(reduced_features) será igual a 339, poderia ser usado também reduced_features.shape[0].

6) Calculando os scores (accuracia) por cross-validation

accuracias = cross_validation.cross_val_score(clf, reduced_features, y, cv=kfold, n_jobs=-1)
O método cross_val_score retorna um vetor com os scores (acuracia) de cada iteração da cross-validation

7) Apresentando os resultados

```
print (scores)
array([ 1.        ,  1.        ,  1.        ,  1.        ,  1.        ,
1.        ,  1.        ,  1.        ,  0.97058824,  1.        ])

print (scores.mean())
0.997058823529
```

Porque K= 16,818 no objeto SelectKbest?

A única coisa que continua obscura para mim em relação ao SelectKbest é a determinação do parametro K, vou descrever aqui, porque escolhi o número 16.318:

Conversando com o Yan ele me falou que nos experimentos dele, as features eram selecionadas em função dos pvalues<0.05, então fiz o seguinte:

Considerando o **Atributo 1** do passo **2)** que é o array com todos os p-values, selecionei apenas os elementos menores que 0.05 e então verifiquei o shape to array resultante, que deu exatamente o número 16.318