Exercício de apoio - Semana 8: Usando k-NN para identificar classes de preços de telefones móveis

A empresa XCelulares quer entrar no mercado de telefones móveis, mas não sabe como precificar os modelos de aparelhos que pretende criar. A empresa teve acesso a uma base de dados contendo as principais características de vários modelos de telefones, incluindo o seu preço, dividido em quatro faixas de valores.

Use o algoritmo k-vizinhos mais próximo (k-NN) para criar um modelo de classificação para que a XCelulares consiga classificar os modelos que irá desenvolver nas faixas de preço corretas.

O conjunto de dados para o exercício está na seguinte URL:

https://github.com/higoramario/univesp-com360-mineracao-dados/raw/main/com360-telefones-moveis.csv

Para este exercício, realize as seguintes etapas:

1. Verifique se há dados nulos que precisam ser tratados.
2. Transforme o atributo alvo **price_range** em valores inteiros para poder usar o k-NN.
3. Observe a distribuição das medidas de tendência central e de dispersão dos dados nas colunas.
4. Gere os histogramas dos atributos.
5. Verifique se há a presença de anomalias nos atributos.
6. Separe os dados em 10% para teste e o restante para treinamento.
7. Use o **KNeighborsClassifier**, da biblioteca scikit learn, para classificar os dados.
8. Use a validação cruzada (**cross_validate**) para treinar o k-NN.
9. Varie os parâmetros de **KNeighborsClassifier** e **cross_validate** e descubra a combinação de valores que obtém a melhor acurácia.
10. Treine o seu modelo com os melhores valores de parâmetros obtidos.
11. Confira qual resultado de acurácia obtido para os dados de teste.

Dicas e observações:

1. A classe **KNeighborsClassifier** possui três parâmetros que aceitam valores diferentes. Altere os valores desses três parâmetros e descubra aquele que obtém a melhor acurácia.

1.1. Importe a biblioteca: **from sklearn.neighbors import KNeighborsClassifier**

1.2. Crie o modelo do classificador: **knn = KNeighborsClassifier(n_neighbors= 5, weights ='uniform', algorithm='auto')**

- **n_neighbors**: número de vizinhos (k, que recebe valores inteiros).
- **weights**: medida de similaridade, que recebe os valores **'uniform'** ou **'distance'**.
- **algorithm**: algoritmo usado pelo k-NN, que recebe os valores **'auto'**, **'ball_tree'**, **'brute'** ou **'kd_tree'**.

2. A validação cruzada pode ser feita usando a função **cross_validate**. Além dos parâmetros do classificador, varie também o número de combinações e descubra aquela que obtém a melhor acurácia.

   2.1. Importe a biblioteca: **from sklearn.model_selection import cross_validate**

   2.2. Exemplo de chamada da função: **validacao_cruzada = cross_validate(knn, atributos_treino, classes_treino, cv=10)**

   2.3. **cv** é o número de combinações

3. Para ver a acurácia média obtida pela validação cruzada, importe a biblioteca **numpy** e execute o comando **np.mean(validacao_cruzada['test_score'])**.

4. Use os melhores valores de **n_neighbors, weights, algorithm e cv** para treinar e testar o modelo:

   4.1. Treinamento: **knn.fit(atributos_treino,classes_treino)**.

   4.2. Teste: **predicao = knn.predict(atributos_teste)**.

5. Veja qual resultado de acurácia você obteve usando **accuracy_score**.

6. Se for preciso, consulte o código do tutorial da Semana 4 para ver um exemplo de uso de **accuracy_score** e da separação dos dados em treinamento e teste (**train_test_split**).

7. Você pode também consultar a documentação do scikit learn:

- k-NN: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
- Teste e treinamento: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
- Validação cruzada: https://scikit-learn.org/0.21/modules/generated/sklearn.model_selection.cross_validate.html
- Acurácia: https://scikit-learn.org/0.21/modules/generated/sklearn.metrics.accuracy_score.html#sklearn.metrics.accuracy_score
- RandomizedSearchCV: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html

8. Uma forma **alternativa** de testar diferentes valores de parâmetros automaticamente é o código a seguir, usando a classe **RandomizedSearchCV**. Veja um exemplo de código abaixo:

```
from sklearn.model_selection import RandomizedSearchCV


# lista com valores para os parâmetros n_neighbors, weights e parameters

n_neighbors = [4,5,10,15,20,30,40]

weights = ['uniform' , 'distance']

algorithm = ['auto' , 'ball_tree' , 'kd_tree' ,'brute']

parameters = dict(n_neighbors = n_neighbors, weights = weights, algorithm =
algorithm)


# experimente valores diferentes para o número de combinações (cv)

clf = RandomizedSearchCV(classificadorKNN, parameters, cv = 5, n_iter = 10)

clf.fit(atributos_treino,classes_treino)

print('acurácia: ' , clf.best_score_)

print('melhores parâmetros: ' , clf.best_params_)

print('melhor classificador: ' , clf.best_estimator_)
```

Descrição dos atributos da base de dados:

- **battery_power**: quantidade de energia armazenada em miliamperes hora (mAh)
- **blue**: bluetooth (sim ou não)
- **clock_speed**: velocidade do microprocessador em gigahertz (GHz)
- **dual_sim**: suporta dois chips (sim ou não)
- **fc**: câmera frontal em megapixels
- **four_g**: 4G (sim ou não)
- **int_memory**: memória interna em gigabytes
- **m_dep**: profundidade do dispositivo em cm
- **mobile_wt**: peso
- **n_cores**: quantidade de núcleos do processador
- **pc**: câmera principal em megapixels
- **px_height**: resolução de altura da tela em pixels
- **px_width**: resolução de largura da tela em pixels
- **ram**: memória RAM em megabytes
- **sc_h**: altura da tela em cm
- **sc_w**: largura da tela em cm
- **talk_time**: tempo de bateria com o telefone em uso realizando chamadas
- **three_g**: 3G (sim ou não)
- **touch_screen**: toque em tela (sim ou não)
- **wifi**: suporte a wi-fi (sim ou não)
- **price_range**: faixa de preços, que pode ser **too-cheap** (muito barato), **cheap** (barato), **fair** (preço justo) ou **expensive** (caro). Este é o atributo de classe.

Versões das bibliotecas:

Esse exercício foi feito usando as seguintes versões de biblioteca:

```
matplotlib==3.2.2
numpy==1.21.6
Python 3.7.13
pandas==1.3.5
scikit-learn==1.0.2
```