Relatório Reconhecimento de Padrões: Atividade 05 – Pré-processamento de Dados

Carlos Emmanuel Pereira Alves
Curso de Bacharelado em Ciência da Computação
Universidade Federal do Agreste de Pernambuco (UFAPE)
Garanhuns, Brasil
carlos.emmanuel.236@gmail.com

1) Nesta questão você deve utilizar a base Student Performance, archive.ics.uci.edu/ml/datasets/Student+Performance (ver arquivo student-mat.csv no student.zip).

a) Explique qual a forma mais adequada para converter todos os atributos da base para numéricos.

Primeiro vamos separar os dados que já são numéricos e não precisam ser convertidos, que são: age, Medu, Fedu, traveltime, studytime, failures, famrel, freetime, goout, Dalc, Walc, health, absences, G1, G2, G3.

Agora com os atributos restantes vamos fazer a conversão para os que são binários e não-binários.
Binários (nominais e ordinais):
school, sex, address, famsize, Pstatus, schoolsup, famsup, paid, activities, nursery, higher, internet, romantic.

Não binário (nominais):
Mjob, Fjob, reason, guardian.

b) Converta todos os atributos da base para numéricos (exceto a classe).

Primeiramente eu separei as colunas binárias, depois eu carreguei a base com o pandas, depois usei o LabelEncoder do sklearn para transformar os atributos binários, depois disso utilizei a make_column_transformer junto com o OneHotEncoder para transformar os atributos não binários e depois ajustar as colunas da base incluindo os novos atributos e apagando os antigos.

```
#Letra B
DataTable.max_columns = 50

cols = [
    'school', 'sex', 'age', 'address', 'famsize', 'Pstatus', 'Medu', 'Fedu', 'Mjob', 'Fjob', 'reason', 'guardian', 'traveltime', 'studytime', 'failures', 'schoolsup',
    'famsup', 'paid', 'activities', 'nursery', 'higher', 'internet', 'romantic', 'famrel', 'freetime', 'goout', 'Dalc', 'Walc', 'health', 'absences', 'G1', 'G2', 'G3'
    ]

binary = ['school', 'sex', 'address', 'famsize', 'Pstatus', 'schoolsup', 'famsup', 'paid', 'activities', 'nursery', 'higher', 'internet', 'romantic']

data = pd.read_csv('student-mat.csv', header=0, names=cols, sep= ';')

for i in range(len(binary)):
  le = LabelEncoder()
  data[binary[i]] = le.fit_transform(data[binary[i]])

transformer = make_column_transformer((OneHotEncoder(), ['Mjob', 'Fjob', 'reason', 'guardian']), remainder='passthrough')
transformed = transformer.fit_transform(data)
transformed_data = pd.DataFrame(transformed, columns=transformer.get_feature_names_out())

transformed_data.head()
```

Resultado:

	onehotencoder__Mjob_at_home	onehotencoder__Mjob_health	onehotencoder__Mjob_other	onehotencoder__Mjob_services	onehotencoder__Mjob_teacher	onehotencoder__Fjob_at_home	onehotencoder_
0	1.0	0.0	0.0	0.0	0.0	0.0	
1	1.0	0.0	0.0	0.0	0.0	0.0	
2	1.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	1.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	1.0	0.0	0.0	0.0	

5 rows x 46 columns

c) Assuma a última coluna (G3, que representa a nota final de cada estudante) como classe. Converta esta coluna (atributo numérico) para uma variável categórica binária. Após esta conversão é possível realizar a tarefa a seguir.

Converti utilizando o critério de que 14 é a nota mínima exigida para ser aprovado então se G3 < 14; G3 = Failed; e se G3 > 13; G3 = Approved. Utilizei a função do pandas, cut, ele transforma variáveis numéricas em categóricas, de acordo com o intervalo dado em bins, e as strings a serem colocadas no lugar, são colocadas no parâmetro labels. Ex: os atributos que estiverem do intervalo -1 a 13, sem incluir o -1 e incluindo o 13, vão receber Failed no lugar.

```
#Letra C
transformed_data['remainder__G3'] = pd.cut(x=transformed_data['remainder__G3'],
                       bins=[-1, 13, 20],
                       labels=['Failed ', 'Approved'])

transformed_data.head()
```

Resultado:

remainder__G1	remainder__G2	remainder__G3
5.0	6.0	Failed
5.0	5.0	Failed
7.0	8.0	Failed
15.0	14.0	Approved
6.0	10.0	Failed

d) Calcule o intervalo de confiança da acurácia para o 100 repetições de holdout 50/50 utilizando o classificador 1-NN com distância Euclidiana. Aqui eu tive que ajeitar a variável das colunas por causa do OneHotEncoder e do make_column_transformer que mudaram o nome das colunas, depois separei a classe dos atributos, e fiz o treinamento com o Knn, guardando os scores na variável data_scores. Para o cálculo do intervalo de confiança primeiro eu obtive o desvio padrão, utilizando a função numpy.std, e a média com o numpy.average. Depois eu calculei o intervalo de confiança utilizando a fórmula dada em aula, e arredondei em 5 casas decimais.

```
#Letra D
cols_new = [
    'onehotencoder__Mjob_at_home',  'onehotencoder__Mjob_health', 'onehotencoder__Mjob_other',  'onehotencoder__M
]
data_x = transformed_data[cols_new[:-1]]
data_y = transformed_data['remainder__G3']
data_scores = []

for i in range(100):
  data_train_x, data_test_x, data_train_y, data_test_y = train_test_split(data_x, data_y, test_size=0.5)

  knn = KNeighborsClassifier(n_neighbors = 1, weights="distance", metric="euclidean")
  knn.fit(data_train_x, data_train_y)

  predict = knn.predict(data_test_x)
  data_score = knn.score(data_test_x, data_test_y)
  data_scores = np.append(data_scores, data_score)

med_1 = np.average(data_scores)
dev_1 = data_scores.std()

confidence_interval_1_n = round(med_1 - (1.96 * dev_1), 5)
confidence_interval_1_p = round(med_1 + (1.96 * dev_1), 5)

print(f'Intervalo de confiança da taxa de acerto: {confidence_interval_1_n}   {confidence_interval_1_p}')
```

Resultado:

```
Intervalo de confiança da taxa de acerto: 0.86033   0.92897
```

2) Utilizando a base Forest Fires. archive.ics.uci.edu/ml/datasets/Forest+Fires

a) Indique a forma mais adequada de converter para numéricos cada um dos atributos da base.
Os únicos atributos que precisam ser convertidos são: month e day.
A forma mais adequada vai ser a utilização de dados cíclicos, por se tratar de dias da semana e do mês.

b) Realize a conversão da base conforme a resposta indicada.
Primeiro eu separei os dias como de 1 a 7 e os meses de 1 a 12, seguindo o conceito dado em sala.

```
DAYS = {
  'sun': 1,
  'mon': 2,
  'tue': 3,
  'wed': 4,
  'thu': 5,
  'fri': 6,
  'sat': 7,
}

MONTHS = {
  'jan': 1,
  'feb': 2,
  'mar': 3,
  'apr': 4,
  'may': 5,
  'jun': 6,
  'jul': 7,
  'aug': 8,
  'sep': 9,
  'oct': 10,
  'nov': 11,
  'dec': 12,
}
```

Depois eu carreguei a base usando o pandas, e fiz variáveis para guardar seno e cosseno de todos, depois eu fiz um for passando por todos os atributos, e troquei o que tinha, pelo número definido a o dia/mês correspondente. Depois fiz o cálculo com a fórmula dada em sala, e guardei nas variáveis de seno e cosseno já definidas. No final eu adicionei novas colunas de seno e cosseno para dia e mês, e apaguei as colunas com os dados não numéricos.

```
cols = ['X', 'Y', 'month', 'day', 'FFMC', 'DMC', 'DC', 'ISI', 'temp', 'RH', 'wind', 'rain', 'area']

data = pd.read_csv('forestfires.csv', header=0, names=cols, sep= ',')

month_cos_all = []
month_sin_all = []
day_cos_all = []
day_sin_all = []

for i, row in data.iterrows():

  month = MONTHS[row['month']]

  month_cos = round(math.cos(2 * math.pi * month / 12), 2)
  month_cos_all = np.append(month_cos_all, month_cos)

  month_sin = round(math.sin(2 * math.pi * month / 12), 2)
  month_sin_all = np.append(month_sin_all, month_sin)

  day = DAYS[row['day']]

  day_cos = round(math.cos(2 * math.pi * day / 7), 2)
  day_cos_all = np.append(day_cos_all, day_cos)

  day_sin = round(math.sin(2 * math.pi * day / 7), 2)
  day_sin_all =  np.append(day_sin_all, day_sin)

data['month_cos'] = month_cos_all
data['month_sin'] = month_sin_all
data['day_cos'] = day_cos_all
data['day_sin'] = day_sin_all

data.drop(['month', 'day'], inplace=True, axis=1)

data.head()
```

Resultado:

	X	Y	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area	month_cos	month_sin	day_cos	day_sin
0	7	5	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.0	0.0	1.00	0.62	-0.78
1	7	4	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.0	0.5	-0.87	-0.90	0.43
2	7	4	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.0	0.5	-0.87	1.00	-0.00
3	8	6	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.0	0.0	1.00	0.62	-0.78
4	8	6	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.0	0.0	1.00	0.62	0.78

3) Utilizando a base Car Evaluation.
archive.ics.uci.edu/ml/datasets/Car+Evaluation

a) Indique a forma mais adequada de converter para numéricos cada um dos atributos da base.
Para a conversão a melhor forma será utilizar o ordinal não binário porque os valores indicam uma ordem que vai do mais baixo ao mais alto. Teremos de converter toda a base pois alguns atributos podem assumir valores numéricos e categóricos.

b) Realize a conversão da base conforme a resposta indicada.
Eu primeiro atribui a variáveis com o nome dos atributos, os possíveis valores desses atributos. Depois carreguei a base usando o pandas. Então utilizei o OrdinalEncoder para converter todos os atributos para numéricos, passando como parâmetro as variáveis definidas antes, ele então utiliza o índice 0 como valor = 0 e assim sucessivamente. Depois eu apenas ajeitei os dados para as colunas ficarem organizadas com os respectivos nomes.

```
cols = ['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class']

buying = ['low', 'med', 'high', 'vhigh']
maint = ['low', 'med', 'high', 'vhigh']
doors = ['2', '3', '4', '5more']
persons = ['2', '4', 'more']
lug_boot = ['small', 'med', 'big']
safety = ['low', 'med', 'high']
classification = ['unacc', 'acc', 'good', 'vgood']

data = pd.read_csv('car.data', header=None, names=cols, sep=',')

enc = OrdinalEncoder(categories=[buying, maint, doors, persons, lug_boot, safety, classification])
data = pd.DataFrame(enc.fit_transform(data), columns=cols)

data.head()
```

Resultado:

	buying	maint	doors	persons	lug_boot	safety	class
0	3.0	3.0	0.0	0.0	0.0	0.0	0.0
1	3.0	3.0	0.0	0.0	0.0	1.0	0.0
2	3.0	3.0	0.0	0.0	0.0	2.0	0.0
3	3.0	3.0	0.0	0.0	1.0	0.0	0.0
4	3.0	3.0	0.0	0.0	1.0	1.0	0.0

4) A base Heart Disease (hungarian) possui alguns valores de atributos omissos. Realize o experimento descrito abaixo utilizando o classificador 1-NN. Divida a base em treino (90%) e teste (10%) de forma estratificada. Calcule o intervalo de confiança para a taxa de acerto do classificador utilizando 100 repetições deste experimento.
Primeiro eu carreguei a base com o pandas, e fiz um replace, onde estava ? coloquei como null, para facilitar o preenchimento das colunas com o SimpleImputer, utilizei a estratégia de média dele para substituir os valores.

```
cols = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal', 'num']

data = pd.read_csv('processed.hungarian.data', header=None, names=cols, sep=',')

data = data.replace('?', np.nan)

imputer = SimpleImputer(strategy='mean', missing_values=np.nan)
imputer = imputer.fit(data)
data.iloc[:,:] = imputer.transform(data)
```

Eu separei então a classe dos atributos, depois utilizei o StratifiedShuffleSplit para separar as repetições, depois treinei o knn, guardando os resultados em data_scores. Para o cálculo do intervalo de confiança primeiro eu obtive o desvio padrão, utilizando a função numpy.std, e a média com o numpy.average. Depois eu calculei o intervalo de confiança utilizando a fórmula dada em aula, e arredondei em 5 casas decimais.

```
data_x = data[cols[:-1]].to_numpy()
data_y = data['num'].to_numpy()

stratified = StratifiedShuffleSplit(n_splits=100, test_size= 0.1)

data_scores = np.array([])

for train_index, test_index in stratified.split(data_x, data_y):
  data_x_train, data_x_test = data_x[train_index], data_x[test_index]
  data_y_train, data_y_test = data_y[train_index], data_y[test_index]

  knn = KNeighborsClassifier(n_neighbors = 1, weights="distance", metric="euclidean")
  knn.fit(data_x_train, data_y_train)

  predict = knn.predict(data_x_test)

  precision, recall, fscore, support = precision_recall_fscore_support(data_y_test, predict)

  data_scores = np.append(data_scores, fscore)

med_1 = np.average(data_scores)
dev_1 = data_scores.std()

confidence_interval_1_n = round(med_1 - (1.96 * dev_1), 5)
confidence_interval_1_p = round(med_1 + (1.96 * dev_1), 5)

print(f'Intervalo de confiança da taxa de acerto: {confidence_interval_1_n}   {confidence_interval_1_p}')
```

Resultado:

```
Intervalo de confiança da taxa de acerto: 0.32766   0.87293
```

a) Preencha os valores omissos no conjunto de treino.
b) Preencha os valores omissos no conjunto de teste utilizando o método e os valores definidos para o conjunto de treino.

5) Utilizando a base de dados Wine https://archive.ics.uci.edu/ml/datasets/wine, para cada um dos casos abaixo, realize 100 repetições de Holdout 50/50 e calcule o intervalo de confiança da acurácia utilizando o classificador 1-NN com distância Euclidiana. Realize testes de hipótese por sobreposição dos intervalos de confiança comparando os pré-processamentos de cada um dos casos abaixo com a base de dados original:
Primeiro eu carreguei a base usando o pandas, depois separei a classe dos atributos. Então fiz as 100 repetições de holdout 50/50, primeiro para a base original, então treinei o Knn, guardando os resultados em data_scores. Para o cálculo do intervalo de confiança primeiro eu obtive o desvio padrão, utilizando a função numpy.std, e a média com o numpy.average. Depois eu calculei o intervalo de confiança utilizando a fórmula dada em aula, e arredondei em 5 casas decimais.

```
cols = ['class', 'alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium', 'total_phenols', 'flavanoi

data = pd.read_csv('wine.data', header=None, names=cols)

data_x = data[cols[1:]]
data_y = data['class']

#original
data_scores = []

for i in range(100):
  data_train_x, data_test_x, data_train_y, data_test_y = train_test_split(data_x, data_y, test_size=0.5)

  knn = KNeighborsClassifier(n_neighbors = 1, weights="distance", metric="euclidean")
  knn.fit(data_train_x, data_train_y)

  predict = knn.predict(data_test_x)
  data_score = knn.score(data_test_x, data_test_y)
  data_scores = np.append(data_scores, data_score)

med = np.average(data_scores)
dev = data_scores.std()

confidence_interval_n = round(med - (1.96 * dev), 5)
confidence_interval_p = round(med + (1.96 * dev), 5)

print(f'Intervalo de confiança da taxa de acerto: {confidence_interval_n}   {confidence_interval_p}')
```

Resultado:

```
Intervalo de confiança da taxa de acerto: 0.62667   0.79288
```

a) Com todas as características ajustadas para o intervalo [0,1].

Aqui usei o MinMaxScaler para ajustar o intervalo entre 0 e 1, e fiz o resto como expliquei acima.

```
#ajustado para [0, 1]

scaler = MinMaxScaler(feature_range=(0,1))
scaled_data = scaler.fit_transform(data_x)

data_scores = []

for i in range(100):
  data_train_x, data_test_x, data_train_y, data_test_y = train_test_split(scaled_data, data_y, test_size=0.5)

  knn = KNeighborsClassifier(n_neighbors = 1, weights="distance", metric="euclidean")
  knn.fit(data_train_x, data_train_y)

  predict = knn.predict(data_test_x)
  data_score = knn.score(data_test_x, data_test_y)
  data_scores = np.append(data_scores, data_score)

med = np.average(data_scores)
dev = data_scores.std()

confidence_interval_n = round(med - (1.96 * dev), 5)
confidence_interval_p = round(med + (1.96 * dev), 5)

print(f'Intervalo de confiança da taxa de acerto ajustado para [0, 1]: {confidence_interval_n}   {confidence_interval_p}')
```

Resultado:

```
Intervalo de confiança da taxa de acerto ajustado para [0, 1]: 0.90403   0.98901
```

b) Com todas as características ajustadas para ter média zero e desvio padrão igual a um.
Aqui usei o StandardScaler para padronizar os dados, e fiz o resto como expliquei acima.

```
#padronizado

scaler = StandardScaler()
scaled_data = scaler.fit_transform(data_x)

data_scores = []

for i in range(100):
  data_train_x, data_test_x, data_train_y, data_test_y = train_test_split(scaled_data, data_y, test_size=0.5)

  knn = KNeighborsClassifier(n_neighbors = 1, weights="distance", metric="euclidean")
  knn.fit(data_train_x, data_train_y)

  predict = knn.predict(data_test_x)
  data_score = knn.score(data_test_x, data_test_y)
  data_scores = np.append(data_scores, data_score)

med = np.average(data_scores)
dev = data_scores.std()

confidence_interval_n = round(med - (1.96 * dev), 5)
confidence_interval_p = round(med + (1.96 * dev), 5)

print(f'Intervalo de confiança da taxa de acerto padronizado: {confidence_interval_n}   {confidence_interval_p}')
```

Resultado:

```
Intervalo de confiança da taxa de acerto padronizado: 0.91439   0.98921
```

No original e no ajustado para [0, 1] não existe sobreposição de intervalos, então temos uma diferença significativa entre eles.
No original e no padronizado também não temos sobreposição de intervalos, e também temos uma diferença significativa entre eles.
No ajustado para [0, 1] e no padronizado temos sobreposição de intervalos, portanto não existe uma diferença significativa entre eles.