LEIC-T 2023/2024
Aprendizagem - Machine Learning
Homework I
Deadline 29/9/2024 20:00
Submit on Fenix as pdf

I) Correlation (2pts)

Compute the correlation (Pearson correlation) and **Spearman's rank** for two variables **x**1 and **x**2. (Indicate all computational steps!)

(a) (1 pts)
x1 = (1, 3, 4, 6)
x2 = (-30, -10, 0, 20)
Why are the same?

pearsons correlation coefficient (rxy)
rxy = (Σ(x1 - média(x1))(x2 - média(x2))) / [√(Σ(x1 - média(x1))^2 * Σ(x2 - média(x2))^2)]
média(x1)= 3.5
média(x2)= -5

Σ(x1 - média(x1))(x2 - média(x2)) =
= (1-3.5)*(-30-(-5)) + (3-3.5)*(-10-(-5)) + (4-3.5)*(0-(-5)) + (6-3.5)*(20-(-5)) = 130
Σ(x1 - média(x1))^2 = (1-3.5)^2 + (3-3.5)^2 + (4-3.5)^2 + (6-3.5)^2 = 13
Σ(x2 - média(x2))^2 = (-30-(-5))^2 + (-10-(-5))^2 + (0-(-5))^2 + (20-(-5))^2 = 1300

rxy = 130/√(13*1300) = 1

Spearman's rank coefficient (rs)
rs = 1 - (6Σdi^2) / n(n^2 - 1) , n = 4 (#elementos do conjunto x1/x2)

x1(ordenado) = (1,2,3,4)
x2(ordenado) = (1,2,3,4)
di = x1i(ordenado) - x2i(ordenado)

Σdi^2 = 0

rs = 1 - (6*0) / 4(4^2 - 1) = 1

rs igual a rxy pois ambos os conjuntos crescem (quando um valor do conjunto x1 aumenta o valor na mesma posição de x2 também aumenta) pois rs é 1 e fazem-no de forma consistente como é possível perceber pelo valor 1 do resultado do coeficiente de correlação de Pearson

(b) (1pts)
x1 = (1, 3, 4, 6)
x2 = (-3, -0.5, 29, 30)
Why are they different?

pearsons correlation coefficient (rxy)
rxy = (Σ(x1 - média(x1))(x2 - média(x2))) / [√(Σ(x1 - média(x1))^2 * Σ(x2 - média(x2))^2)]
média(x1)= 3.5
média(x2)= 13.875

Σ(x1 - média(x1))(x2 - média(x2)) =
= (1-3.5)*(-3-13.875) + (3-3.5)*(-0.5-13.875) + (4-3.5)*(29-13.875) + (6-3.5)*(30-13.875) = 97.75
Σ(x1 - média(x1))^2 = (1-3.5)^2 + (3-3.5)^2 + (4-3.5)^2 + (6-3.5)^2 = 13
Σ(x2 - média(x2))^2 = (-3-13.875)^2 + (-0.5-13.875)^2 + (29-13.875)^2 + (30-13.875)^2 = 980.1875

rxy = 97.25/√(13*980.1875) = 0.8615

Spearman's rank coefficient (rs)

rs = 1 - (6Σdi^2) / n(n^2 - 1) , n = 4 (#elementos do conjunto x1/x2)

x1(ordenado) = (1,2,3,4)
x2(ordenado) = (1,2,3,4)
di = x1i(ordenado) - x2i(ordenado)

Σdi^2 = 0

rs = 1 - (6*0) / 4(4^2 - 1) = 1

rs diferente de rxy pois ambos os conjuntos crescem (quando um valor do conjunto x1 aumenta o valor na mesma posição de x2 também aumenta) pois rs é 1 mas não de forma consistente como é possível perceber pelo valor <1 do resultado do coeficiente de correlação de Pearson

II) Decision Trees (5pts)

F1	*F2*	*F3*	*F4*	*Output*
c	*a*	*b*	*x*	*n*
a	*a*	*c*	*a*	*t*
a	*b*	*b*	*a*	*t*
c	*b*	*c*	*x*	*m*
a	*b*	*b*	*c*	*f*

(a) (2 pts)
Determine the root of decision tree using the ID3 algorithm with the target "Output". Indicate the calculation. (Indicate all computational steps!)

p(n) = 1/5
p(t) = 2/5
p(m) = 1/5
p(f) = 1/5

I(Output) = - (1/5) * log2(1/5) - (2/5) * log2(2/5) - (1/5) * log2(1/5) - (1/5) * log2(1/5) = 1.922 bit

Gain(F1) = I(Output) - [(2/5) * I(S_c) + (3/5) * I(S_a)] = 0.9712 bit
I(S_c) = - (1/2) * log2(1/2) - (1/2) * log2(1/2) = 1 bit
I(S_a) = - (1/3) * log2(1/3) - (2/3) * log2(2/3) = 0.918 bit

Gain(F2) = I(Output) - [(2/5) * I(S_a) + (3/5) * I(S_b)] = 0.571 bit
I(S_a) = - (1/2) * log2(1/2) - (1/2) * log2(1/2) = 1 bit
I(S_b) = - (1/3) * log2(1/3) - (1/3) * log2(1/3) - (1/3) * log2(1/3) = 1.585 bit

Gain(F3) = I(Output) - [(3/5) * I(S_b) + (2/5) * I(S_c)] = 0.571 bit
I(S_c) = - (1/2) * log2(1/2) - (1/2) * log2(1/2) = 1 bit
I(S_b) = - (1/3) * log2(1/3) - (1/3) * log2(1/3) - (1/3) * log2(1/3) = 1.585 bit

Gain(F4) = I(Output) - [(2/5) * I(S_a) + (1/5) * I(S_c) + (2/5) * I(S_x)] = 1.522 bit
I (S_x) = - (1/2) * log2(1/2) - (1/2) * log2(1/2) = 1 bit
I (S_a) = - 1*log2(1) = 0 bit
I (S_c) = - 1*log2(1) = 0 bit

O atributo com maior ganho é o F4 sendo por isso a root da decision tree

(b) (2 pts)
Determine the decision tree using the ID3 algorithm with the target "Output". Indicate the calculation and draw your decision tree.

F4 é a root da decision tree

For a:

F1	F2	F3	Output
a	a	c	t
a	b	b	t

For c:

F1	F2	F3	Output
a	b	b	f

For x:

F1	F2	F3	Output
c	a	b	n
c	b	c	m

I(F1) = 1 * I(S_c) = 1 bit
I(S_c) = – (1/2) * log2(1/2) – (1/2) * log2(1/2) = 1 bit

Gain(F2) = (1/2) * I(S_a) + (1/2) * I(S_b) = 0 bit
I(S_a) = – 1*log2(1) = 0 bit
I(S_b) = – 1*log2(1) = 0 bit

I(F3) = (1/2) * I (S_b) + (1/2) * I(S_c) = 0 bit
I(S_c) = – 1*log2(1) = 0 bit
I(S_b) = – 1*log2(1) = 0 bit

Como a entropia de F2 e F3 são iguais e menores que a entropia de F1 vou escolher de forma aleatória um deles (F2)

For F2:

For a:

F1	F3	Output
c	b	n

For b:

F1	F3	Output
c	c	m

Decision Tree:

```
        F4
        /|\
       / | \
      /  |  \
     /   |   \
   a     c     x
  /      |      \
 t       f       F2
                 / \
                a   b
               /     \
              n       m
```

(c) (1 pts)
Draw the training confusion matrix for the learnt decision tree.

true = (n,t,t,m,f)
predict = (n,t,t,m,f)

p\t	n	t	m	f
n	1	0	0	0
t	0	2	0	0
m	0	0	1	0
f	0	0	0	1

LEIC-T 2023/2024
Aprendizagem - Machine Learning
Homework I
Deadline 29/9/2024 20:00
Submit on Fenix as pdf

III Software Experiments (3pts)

Download the jupyter notbook HM1_DT.ipynb.

We will use the build in wine data set:

Using chemical analysis to determine the origin of wines.
These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

Scikit-learn only supports **binary splits** and **numerical variables** for now

(a) (1pts)

Split the data using the command (in the notebook)
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=**value**,stratify=y,random_state=**your_group number**)
Partition data with train_test_split, with values 0.2, 0.5, 0.7 and indicate the depth and the accuracy of each the decision tree (if you have no group yet, put the last three digits of student nr).

code:

```
import matplotlib.pyplot as plt
from sklearn import metrics, datasets, tree
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

# 1. load
wine = datasets.load_wine()
X, y = wine.data, wine.target

# partition data with train_test_spli

#trian_size, sratify

value= #0.2,0.5,0.7
gn=36

X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=value,stratify=y,random_state=gn)

#radom_state: Controls the shuffling applied to the data before applying the split.
#Pass an int for reproducible output across multiple function calls
#Popular integer random seeds are 0 and 42
```

```
print("train size:",len(X_train),"\ntest size:",len(X_test))

# Instantiate the decision tree classifier with max_depth=3
clf = DecisionTreeClassifier(max_depth=10000)

# Fit the classifier to the training data
clf.fit(X_train, y_train)

# Predict the labels of the testing data
y_pred = clf.predict(X_test)

# Evaluate the performance of the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Fit the classifier to the training data
clf.fit(X_train, y_train)

# Get the depth of the tree
depth = clf.tree_.max_depth
print("Depth of the tree:", depth)
```

value = 0.2:

train size: 35
test size: 143
depth: 3
accuracy: 0.8951048951048951

value = 0.5:

train size: 89
test size: 89
depth: 5
accuracy: 0.9325842696629213

value = 0.7:

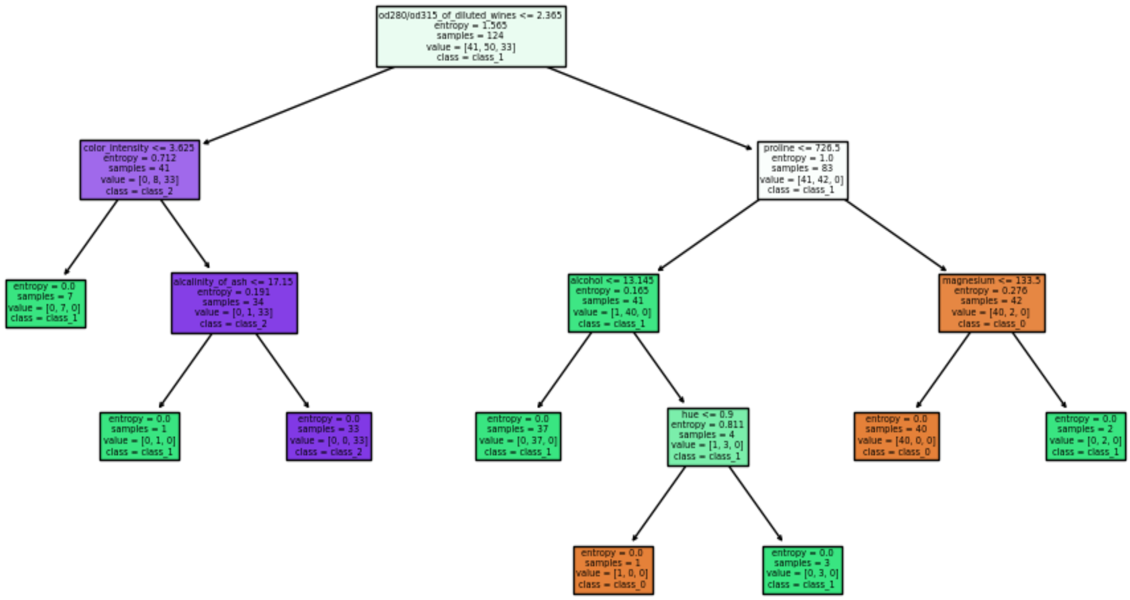
train size: 124
test size: 54
depth: 5
accuracy: 0.9444444444444444

(b) (1pts)

Draw the decision tree for the value 0.7 (copy the drawing into your document)

code:

```
import matplotlib.pyplot as plt
from sklearn import datasets, tree
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

SEED = 42

# 1. load
wine = datasets.load_wine()
X, y = wine.data, wine.target

# partition data with train_test_split

# train_size, stratify

value = 0.7
gn = 36

X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=value, stratify=y, random_state=gn)

# random_state: Controls the shuffling applied to the data before applying the split.
# Pass an int for reproducible output across multiple function calls
# Popular integer random seeds are 0 and 42

print("train size:", len(X_train), "\ntest size:", len(X_test))

# Instantiate the decision tree classifier with max_depth=3
clf = DecisionTreeClassifier(max_depth=3)

# Fit the classifier to the training data
clf.fit(X_train, y_train)

# Predict the labels of the testing data
y_pred = clf.predict(X_test)

# Evaluate the performance of the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Get the depth of the tree
depth = clf.tree_.max_depth
print("Depth of the tree:", depth)

# 2. learn classifier, random_state=SEED
predictor = DecisionTreeClassifier(criterion='entropy', random_state=SEED)
predictor.fit(X_train, y_train)

# 3. plot classifier
figure = plt.figure(figsize=(12, 6))
tree.plot_tree(predictor, feature_names=wine.feature_names, class_names=[str(i) for i in wine.target_names], filled=True)
plt.show()

figure.savefig("decision_tree.png")
```



(c) (1pts)

Now perform the same experiment without the command stratify=y,with the value 0.7

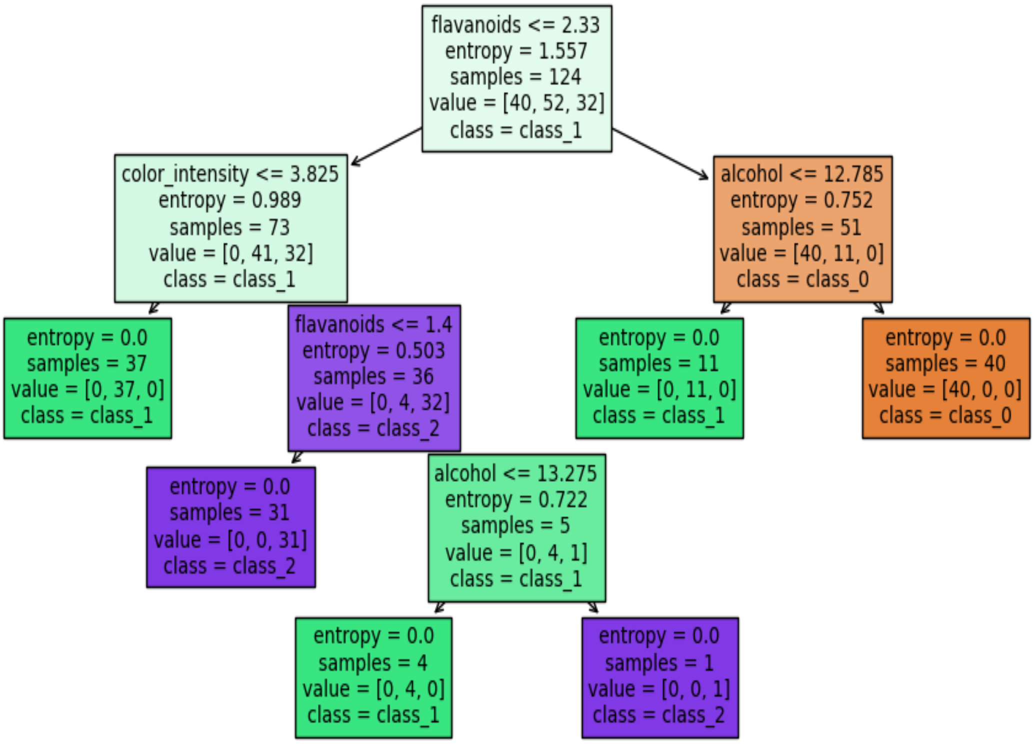
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=**0.7**, ,random_state=**your_group number**)

Draw the decision tree (for the value 0.7, copy the drawing into your document)

Are the result different, what is the meaning of the command stratify=y and stratified fashion? (See scikit-learn manual) and write in **one** sentence pls, do not copy the whole section (only the important part, if you copy…)

code:

mesmo que o anterior, mas sem o comando stratify=y



O parâmetro stratify na função train_test_split permite dividir um conjunto de dados em conjuntos de training e test, garantindo que as proporções de cada classe sejam as mesmas em ambos os conjuntos.
Sem o uso de stratify=y, a distribuição das classes nos conjuntos de training e test pode ser diferente da do conjunto de dados original, podendo levar a resultados biased, principalmente ao lidar com conjuntos de dados desequilibrados.