7506-2021 Coloquio 12-8-2021

Román Vázquez Lareu

TOTAL POINTS

70 / 100

QUESTION 1

1 Punto 1 20 / 50

✓ **- 0 pts Correct**

✓ **- 30 pts Punto c mal.**

1 Esto está radicalmente mal. Al devolver 1 o 0 luego no puede comparar contra el próximo registro porque ya no tiene la fecha. Pierde la estructura de los registros. Las operaciones de reduce tienen que devolver la misma estructura que usan como input.

QUESTION 2

2 Punto 2 50 / 50

✓ **- 0 pts Correcto.**

gradescope

```
tests = [("2021-01-15",4000, "l1",0),
         ("2021-01-15",4000, "l1",0),
         ("2021-01-15",4000, "l4",0),
         ("2021-02-15",4000, "l2",1),
         ("2021-02-15",3000, "l2",1),
         ("2021-03-13",5000, "l1",0),
         ("2021-03-15",3000, "l1",0),
         ("2021-03-15",4000, "l3",1),
         ("2021-03-15",5000, "l3",1),
         ("2021-03-15",4000, "l1",0),
         ("2021-03-11",1000, "l4",1),
         ("2021-03-11",5000, "l1",0),
         ("2021-10-12",5000, "l1",0),
         ("2021-02-12",5000, "l5",0)
         ]

localidades = [("l1","n1","BsAs"),
               ("l2","n2","p2"),
               ("l3","n3","p3"),
               ("l4","n4","BsAs"),
               ("l5","n5","p5")
               ]


testsRdd = sc.parallelize(tests)
localidadesRdd = sc.parallelize(localidades)
```

a)

```
tests_localidades_primer_trimestre = testsRdd.filter(lambda x: (x[0].split("-")[0]=="2021") & (x[0].split("-")[1] in ["01","02","03"])).cache()


id_localides_bsas = localidadesRdd.filter(lambda x: x[2]=="BsAs").map(lambda x: (x[0],x[2]) )
```

```
[('l1', 'BsAs'), ('l4', 'BsAs')]
```

```
tests_to_join = tests_localidades_primer_trimestre.map(lambda x: (x[2],x[1]) )
```

```
[('l1', 4000),
 ('l1', 4000),
 ('l4', 4000),
 ('l2', 4000),
 ('l2', 3000),
 ('l1', 5000),
```

```
 ('l1', 3000),
 ('l3', 4000),
 ('l3', 5000),
 ('l1', 4000),
 ('l4', 1000),
 ('l1', 5000),
 ('l5', 5000)]
```

```
tests_bsas = tests_to_join.join(id_localides_bsas).map(lambda x: (x[1][0],1) ).reduceByKey(lambda x,y: x+y).reduce(lambda x,y: x if x[1]>y[1] else y)[0]
tests_bsas
```

```
4000
```

b)

```
tests_localidades_primer_trimestre_positividad_nula = tests_localidades_primer_trimestre.map(lambda x: (x[2],(1,1)  if x[3] == 1 else (0,1) ) )\
                                                .reduceByKey(lambda x,y: (x[0]+y[0],x[1]+y[1]) ).filter(lambda x: (x[1][0]==0) )
tests_localidades_primer_trimestre_positividad_nula.map(lambda x: (1,1)).reduceByKey(lambda x,y: x+y).collect()[0][1]
```

```
2
```

c)

```
from datetime import datetime
def diferencia_entre_fechas(fecha1,fecha2):
    fecha1 = datetime.strptime(fecha1, "%Y-%m-%d")
    fecha2 = datetime.strptime(fecha2, "%Y-%m-%d")

    return abs((fecha2 - fecha1).days)
```

```
tests_localidades_primer_trimestre_por_localidad = tests_localidades_primer_trimestre.map(lambda x: (x[2],(x[0],x[3])))
```

```
[('l1', ('2021-01-15', 0)),
 ('l1', ('2021-01-15', 0)),
 ('l4', ('2021-01-15', 0)),
 ('l2', ('2021-02-15', 1)),
 ('l2', ('2021-02-15', 1)),
 ('l1', ('2021-03-13', 0)),
 ('l1', ('2021-03-15', 0)),
 ('l3', ('2021-03-15', 1)),
 ('l3', ('2021-03-15', 1)),
 ('l1', ('2021-03-15', 0)),
 ('l4', ('2021-03-11', 1)),
```

```
     ('l1', ('2021-03-11', 0)),
     ('l5', ('2021-02-12', 0))]
```

```
tests_localidades_primer_trimestre_por_localidad.reduceByKey(lambda x,y: (1 if (diferencia_entre_fechas(x[0],y[0])<=2 & (x[0]==1 | y[0]==1) )  else 0 ) )\
                                                                                                                        .filter(lambda x: x[1] == 1 ).collect()
```

1

1 Punto 1 20 / 50

✓ - 0 pts Correct

✓ - 30 pts Punto c mal.

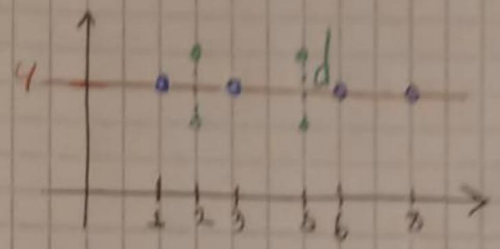
1 Esto está radicalmente mal. Al devolver 1 o 0 luego no puede comparar contra el próximo registro porque ya no tiene la fecha. Pierde la estructura de los registros. Las operaciones de reduce tienen que devolver la misma estructura que usan como input.

gradescope

2) a) $MSE = \frac{\sum_i (Y_i - \hat{Y}_i)^2}{m}$

$\hat{Y}_i$ = aproximación
Y_i = valor
m = # puntos

Entrenamos a la regresión con el de train.
La regresión más simple es la lineal



elija recta
$Y(x) = 4$

Error de train = 0 azul
Error de test = 0.5 verde

$$MSE_{train} = \sum_1^4 \frac{(Y_i - \hat{Y}_i)^2}{4} = \frac{(4-4)^2 + (4-4)^2 + (4-4)^2 + (4-4)^2}{4} = 0$$

$$MSE_{test} = \sum_1^4 \frac{(Y_i - \hat{Y}_i)}{4} = \frac{(d^2 + d^2 + d^2 + d^2)}{4} = \frac{4d^2}{4} = d^2$$

$MSE_{test} = d^2 = 0,5 \iff d = \sqrt{0,5} = 1/\sqrt{2}$

set train	set test
(1,4)	$(2, 4+1/\sqrt{2})$
(3,4)	$(2, 4-1/\sqrt{2})$
(6,4)	$(5, 4+1/\sqrt{2})$
(8,4)	$(5, 4-1/\sqrt{2})$

b) accuracy $train_{m_1} = 0,6$

accuracy $train_{m_2} = 0,5$

accuracy averaging $= 0,7$

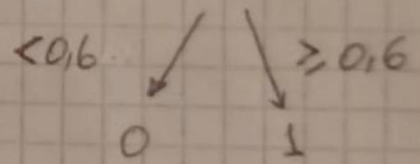
$\hat{Y}_1$	$\hat{Y}_2$	Y	$\frac{\hat{Y}_1 + \hat{Y}_2}{2}$	
✓ 0,1	✓ 0,1	0	0,1	✓
✓ 0,1	✓ 0,1	0	0,1	✓
✓ 0,1	✓ 0,1	0	0,1	✓
✗ 0,6	✓ 0,1	0	0,35	✓
✓ 0,1	✓ 0,1	0	0,1	✓
✓ 0,1	✗ 0,6	0	0,35	✓
✗ 0,9	✗ 0,9	0	0,9	✗
✗ 0,9	✗ 0,9	0	0,9	✗
✗ 0,9	✗ 0,9	0	0,9	✗
✓ 0,1	✗ 0,6	0	0,35	✓
6/10	5/10	↑ verdadero		7/10
⇓	⇓			⇓
0,6	0,5			0,7

c)

clase	label		mean enco
A	1		$\frac{1+1}{2} = 1$
A	1		$\frac{1+1}{2} = 1$
B	0	$\Rightarrow$	$\frac{0+1}{2} = 0,5$
B	1		$\frac{0+1}{2} = 0,5$
C	0		$\frac{0+1}{2} = 0,5$
C	1		$\frac{0+1}{2} = 0,5$

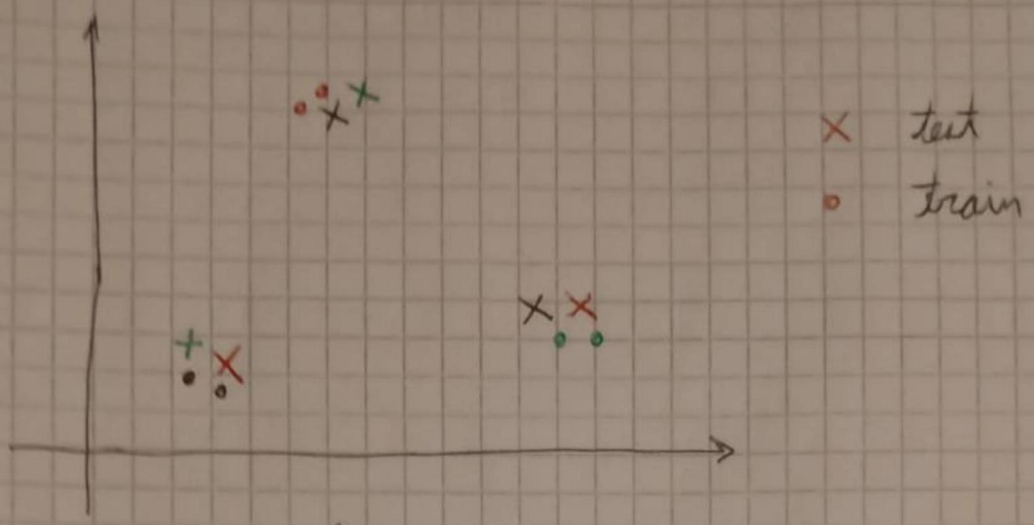
feature	label	$\hat{y}$	
1	1	1	✓
1	1	1	✓
0,5	0	0	✓
0,5	1	0	✗
0,5	0	0	✓
0,5	1	0	✗

árbol:



6/8 → 0,75

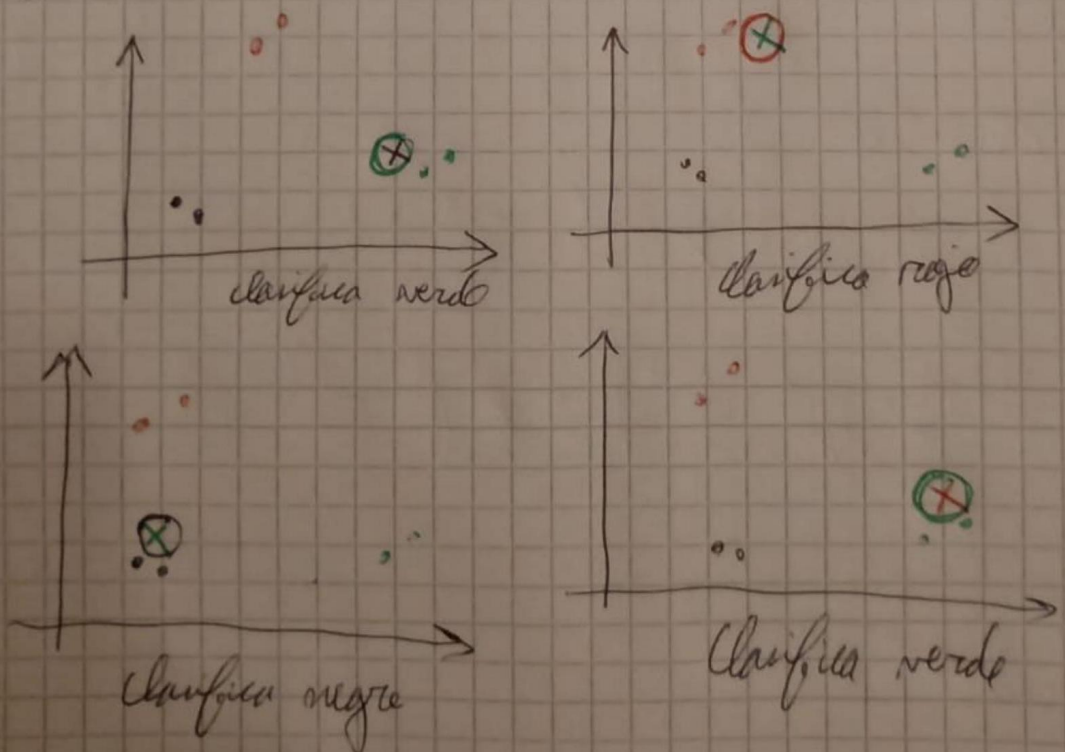
d)



Al momento de clasificar solo tengo en cuenta los de train.

La distancia que usa KNN es la euclídea, con el gráfico es evidente como clasificaría, no hace falta ver distancias

Ejemplo



2 Punto 2 **50 / 50**

✓ **- 0 pts** **Correcto.**

gradescope