МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В. И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по практической работе №7

по дисциплине «Машинное обучение»

Студент гр. 1310 _____ Комаров Д. Е.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2025

Задание 1

Постановка задачи

Дан набор данных, представленный в таблице 1.

Таблица 1 – Данные для 1 задания

$\mathbf{x}_i$	a_1	a_2	a_3	Class
$\mathbf{x}_1$	T	T	5.0	Y
$\mathbf{x}_2$	T	T	7.0	Y
$\mathbf{x}_3$	T	F	8.0	N
$\mathbf{x}_4$	F	F	3.0	Y
$\mathbf{x}_5$	F	T	7.0	N
$\mathbf{x}_6$	F	T	4.0	N
$\mathbf{x}_7$	F	F	5.0	N
$\mathbf{x}_8$	T	F	6.0	Y
$\mathbf{x}_9$	F	T	1.0	N

Используя наивный байесовский классификатор, необходимо определить класс точки (T,F,1.0).

Код программы

```
import scipy.stats
import numpy as np
D = [
    ['T', 'T', 5.0, 'Y'],
    ['T', 'T', 7.0, 'Y'],
    ['T', 'F', 8.0, 'N'],
    ['F', 'F', 3.0, 'Y'],
    ['F', 'T', 7.0, 'N'],
    ['F', 'T', 4.0, 'N'],
    ['F', 'F', 5.0, 'N'],
    ['T', 'F', 6.0, 'Y'],
    ['F', 'T', 1.0, 'N']
]
point=['T','F',1.0]
DY=[d[:-1]  for d in D if d[3]=='Y']
DN=[d[:-1]  for d in D if d[3]=='N']
PY=len(DY)/len(D)
PN=len(DN)/len(D)
PYT0=len([dy for dy in DY if dy[0]=='T'])/len(DY)
PYF0=len([dy for dy in DY if dy[0]=='F'])/len(DY)
PYT1=len([dy for dy in DY if dy[1]=='T'])/len(DY)
PYF1=len([dy for dy in DY if dy[1]=='F'])/len(DY)
FY2=scipy.stats.norm(np.mean([dy[2]  for  dy  in  DY]),np.std([dy[2]  for  dy  in
DY],ddof=1))
PNT0=len([dn for dn in DN if dn[0]=='T'])/len(DN)
PNF0=len([dn for dn in DN if dn[0]=='F'])/len(DN)
PNT1=len([dn for dn in DN if dn[1]=='T'])/len(DN)
PNF1=len([dn for dn in DN if dn[1]=='F'])/len(DN)
FN2=scipy.stats.norm(np.mean([dn[2]  for  dn  in  DN]),np.std([dn[2]  for  dn  in
DN],ddof=1))
```

```
PpointInY=(PYT0   if   point[0]=='T'else   PYF0)*(PYT1   if   point[1]=='T'else
PYF1)*FY2.pdf(point[2])*PY
PpointInN=(PNT0   if   point[0]=='T'else   PNF0)*(PNT1   if   point[1]=='T'else
PNF1)*FN2.pdf(point[2])*PN
print(PpointInY,PpointInN)
if(PpointInY>PpointInN):
    print("Класс Y")
else:
    print("Класс N")
```

Результат выполнения

Точка (T,F,1.0) принадлежит к классу N.

Задание 2

Постановка задачи

Даны два класса c1 и c2 со следующими мат. ожиданиями и ковариациями

$$\mu_1 = (1,3),$$

$$\Sigma_1 = \begin{pmatrix} 5 & 3 \\ 3 & 2 \end{pmatrix},$$

$$\mu_2 = (5,5),$$

$$\Sigma_2 = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}.$$

Необходимо классифицировать точку (3,4), используя Байесовский вывод, предположив, что классы распределены по нормальному закону, и P(c1) = P(c2) = 0.5.

Код программы

```
import scipy.stats
m1=[1, 3]
m2=[5, 5]
cov1=[[5, 3],
      [3, 2]]
cov2=[[2, 0],
      [0, 1]]

point=[3,4]
Pc1=0.5
Pc2=0.5
c1=scipy.stats.multivariate_normal(m1,cov1)
c2=scipy.stats.multivariate_normal(m2,cov2)
PpointInc1=c1.pdf(point)*Pc1
PpointInc2=c2.pdf(point)*Pc2
print(PpointInc1,PpointInc2)
if(PpointInc1>PpointInc2):
    print("Класс c1")
else:
    print("Класс c2")
```

Результат выполнения

Точка (3,4) принадлежит к классу c1.

Задание 3

Постановка задачи

Дан набор данных, представленный в таблице 2.

Таблица 2 – Данные для 3 задания

Point	Age	Car	**Risk**
$\mathbf{x}_1$	25	Sports	*L*
$\mathbf{x}_2$	20	Vintage	*H*
$\mathbf{x}_3$	25	Sports	*L*
$\mathbf{x}_4$	45	SUV	*H*
$\mathbf{x}_5$	20	Sports	*H*
$\mathbf{x}_6$	25	SUV	*H*

Необходимо построить решающее дерево используя порог для чистоты (purity threshold) равным 100%. В качестве критерия для разделения необходимо использовать энтропию. Необходимо классифицировать наблюдение (Age=27,Car=Vintage).

Код программы

```
import math

D = [
    [25, "Sports", "L"],
    [20, "Vintage", "H"],
    [25, "Sports", "L"],
    [45, "SUV", "H"],
    [20, "Sports", "H"],
    [25, "SUV", "H"]
]

PL=len([d for d in D if d[2]=='L'])/len(D)
PH=len([d for d in D if d[2]=='H'])/len(D)

HD=-PL*math.log2(PL)-PH*math.log2(PH)
print("Entropy:",HD)
ages=sorted(set([d[0]for d in D]))
cars=list(set([d[1]for d in D]))
Dag=[[d for d in D if d[0]==ag]  for ag in ages]
PagesL=[(len([d for d in dag if d[2]=='L'])/len(dag)) if len(dag)>0 else 0 for
dag in Dag]
PagesH=[(len([d for d in dag if d[2]=='H'])/len(dag)) if len(dag)>0 else 0 for
dag in Dag]
Hages=[(-PagesH[i]*math.log2(PagesH[i])    if    PagesH[i]>0    else    0)+(-
PagesL[i]*math.log2(PagesL[i])   if   PagesL[i]>0   else   0)   for   i   in
range(len(ages))]
Hage=sum([Hages[i]*len(Dag[i])/len(D)for i in range(len(ages))])
```

```
Dca=[[d for d in D if d[1]==ca]  for ca in cars]
PcarsL=[(len([d for d in dca if d[2]=='L'])/len(dca)) if len(dca)>0 else 0 for
dca in Dca]
PcarsH=[(len([d for d in dca if d[2]=='H'])/len(dca)) if len(dca)>0 else 0 for
dca in Dca]
Hcars=[(-PcarsH[i]*math.log2(PcarsH[i])    if    PcarsH[i]>0    else    0)+(-
PcarsL[i]*math.log2(PcarsL[i])   if   PcarsL[i]>0   else   0)   for   i   in
range(len(cars))]
Hcar=sum([Hcars[i]*len(Dca[i])/len(D)for i in range(len(cars))])
print("Ages entropty",Hage)
print("Cars entropy",Hcar)

maxGain=0
for car in cars:
    DY=[d for d in D if d[1]==car]
    DN=[d for d in D if d[1]!=car]
    DYL=[d for d in DY if d[2]=='L']
    DYH=[d for d in DY if d[2]=='H']
    DNL=[d for d in DN if d[2]=='L']
    DNH=[d for d in DN if d[2]=='H']
    PYL=len(DYL)/len(DY)
    PYH=len(DYH)/len(DY)
    PNL=len(DNL)/len(DN)
    PNH=len(DNH)/len(DN)
    HDY=(-PYL*math.log2(PYL)if PYL>0 else 0)+(-PYH*math.log2(PYH) if PYH>0 else
0)
    HDN=(-PNL*math.log2(PNL)if PNL>0 else 0)+(-PNH*math.log2(PNH)if PNH>0 else
0)
    gain=HD-(len(DY)/len(D))*HDY-(len(DN)/len(D))*HDN
    if gain>maxGain:
        maxGain,bestDivCar=gain,car
        bestHDY,bestHDN=HDY,HDN
        bestDY,bestDN=DY,DN

cars.remove(bestDivCar)
print("First diviston:",[bestDivCar],cars)
print("Entripies after division:",bestHDY,bestHDN)
D=bestDY

maxGain=0
for div in [(ages[i]+ages[i+1])/2 for i in range(len(ages)-1)]:
    DY=[d for d in D if d[0]<=div]
    DN=[d for d in D if d[0]>div]
    DYL=[d for d in DY if d[2]=='L']
    DYH=[d for d in DY if d[2]=='H']
    DNL=[d for d in DN if d[2]=='L']
    DNH=[d for d in DN if d[2]=='H']
    PYL=len(DYL)/len(DY) if len(DY)>0 else 0
    PYH=len(DYH)/len(DY)if len(DY)>0 else 0
    PNL=len(DNL)/len(DN)if len(DN)>0 else 0
    PNH=len(DNH)/len(DN)if len(DN)>0 else 0
    HDY=(-PYL*math.log2(PYL)if PYL>0 else 0)+(-PYH*math.log2(PYH) if PYH>0 else
0)
    HDN=(-PNL*math.log2(PNL)if PNL>0 else 0)+(-PNH*math.log2(PNH)if PNH>0 else
0)
    gain=HD-(len(DY)/len(D))*HDY-(len(DN)/len(D))*HDN
    if gain>maxGain:
        maxGain,bestDivAge=gain,div
        bestHDY,bestHDN=HDY,HDN
        bestDY,bestDN=DY,DN

print("First diviston: <=",bestDivAge)
print("Entripies after division:",bestHDY,bestHDN)
```

```
point=[27,"Vintage"]

if point[1]==bestDivCar:
    if point[0]<=bestDivAge:
        cls='H'
    else:
        cls='L'
else:
    cls='H'
print ("Predicted class:",cls)
```

Результат выполнения

Полученное классифицирующее дерево представлено на рисунке 1.

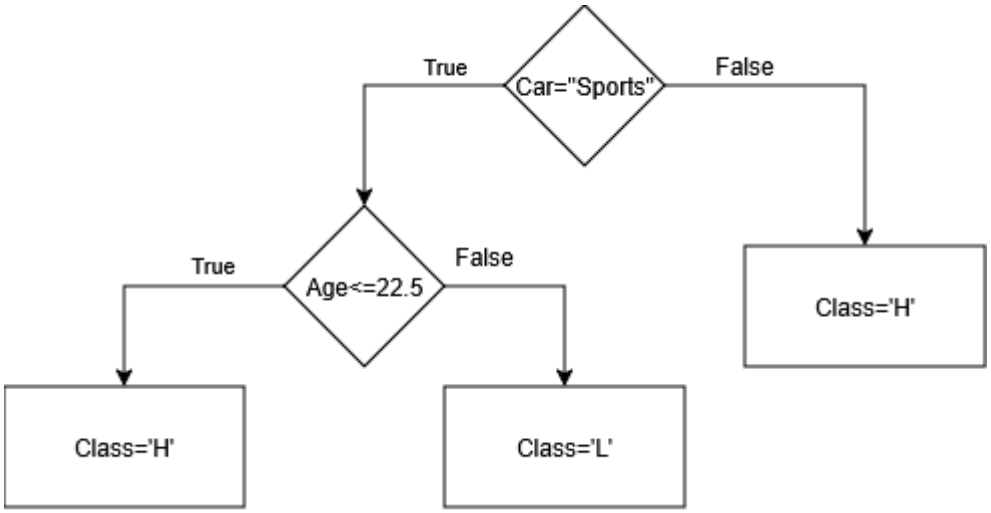

Рисунок 1 – Классифицирующее дерево

Наблюдение (Age=27,Car=Vintage) имеет класс H.