PROBLEM 1

Instance	A	B	C	Class
1	F	T	120	B
2	T	F	1090	B
3	T	T	245	M
4	F	F	589	M
5	T	T	877	M

(a) How much information about the class is gained by knowing whether or not the value of feature C is less than 475?
(b) How much information about the class is gained by knowing whether or not the value of features A and B are different?

PART A		
C>475	0.60	
C<475	0.40	
H_C(Y)	0.97	
H_C(Y|C<475)	1.00	1 B 1 M
H_C(Y|C>475)	0.92	1 B 2 M
InfoGain	**0.02**	

PART B		
A==B	0.60	
A!=B	0.40	
H_AB(Y)	0.97	
H_AB(Y|A==B)	0.00	0 B 3 M
H_AB(Y|A!=B)	0.00	2 B 0 M
InfoGain	**0.97**	

PROBLEM 2

Suppose we want to learn a k-nearest neighbor model with the following data set and we are using Leave One Out Cross Validation (LOOCV) to select k. What would LOOCV pick - k = 1 or k = 2 or k = 3. Use Manhattan distance for calculations.

Instance	Feature 1	Feature 2	Class
1	2	3	1
2	4	4	1
3	4	5	0
4	6	3	1
5	8	3	0
6	8	4	0

Results of script (script attached at end of pdf):

k	**accuracy**
1	**0.5**
2	**0.42**
3	**0.39**

So the algorithm would choose k=1, but this isn't usually a good idea

PROBLEM 3

X	Y	Z	Count
1	1	1	36
1	1	0	4
1	0	1	2
1	0	0	8
0	1	1	9
0	1	0	1
0	0	1	8
0	0	0	32

(a) In the first round of the sparse candidate algorithm, compute the mutual information between:

i. I(X, Z)

ii. I(Y, Z)

(b) Which feature should be selected as candidate parent for Z? Why?

(c) Estimate the parameters of the Bayes net: X -> Y -> Z

Calculation tables are attached at the end of the pdf

(a)		
I(X,Z) =	sum(P(x, z)*log(P(x,z)/(P(x)P(z)))) =	**0.1328449618**
I(Y,Z) =	sum(P(y, z)*log(P(y,z)/(P(y)P(z)))) =	**0.3973126097**

(b)
Y is a better parent for Z because more info is gained between the two

(c)

P(X)	
F	T
0.5	0.5

P(Y|X)		
X	F	T
T	0.2	0.8
F	0.8	0.2

P(Z|Y)		
Z	F	T
T	0.18	0.82

F	0.89	0.11

PROBLEM 4

	x	y
1	12	4
2	3	18
3	6	11
4	5	5

(a)	polynomial of degree 2	k(x,z) =	(x*z)^d
(b)	polynomial of degree up to 2	k(x,z) =	(x*z+1)^d
(c)	RBF with gamma=1	k(x,z) =	exp(-gamma*abs(x-z)^2)

Kernel Matrix:	k(x1,y1)	k(x1,y2)	k(x1,y3)	k(x1,y4)
	k(x2,y1)	k(x2,y2)	k(x2,y3)	k(x2,y4)
	k(x3,y1)	k(x3,y2)	k(x3,y3)	k(x3,y4)
	k(x4,y1)	k(x4,y2)	k(x4,y3)	k(x4,y4)

(a)			
2304	46656	17424	3600
144	2916	1089	225
576	11664	4356	900
400	8100	3025	625

(b)			
2401	47089	17689	3721
169	3025	1156	256
625	11881	4489	961
441	8281	3136	676

(c)			
6.23515E+27	4.31123E+15	2.718281828	1.90735E+21
2.718281828	5.20306E+97	6.23515E+27	54.59815003
54.59815003	3.45466E+62	72004899337	2.718281828
2.718281828	2.48752E+73	4.31123E+15	1

PROBLEM 5

Concept class 'C', pair of circles centered at the origin (0,0) with radii r and r+a
Training instances: real-valued features x_1 and x_2 w/ a binary class label $y \in \{0,1\}$
y = 1 predicts for instances inside donut, and y = 0 otherwise
Show that C is PAC learnable.

Infinite Hypothesis Space due to real valued features

$$m \geq \frac{1}{\epsilon}\left(4\log_2\left(\frac{2}{\delta}\right) + 8VC(H)\log_2\left(\frac{13}{\epsilon}\right)\right)$$

(upper bound)

$$m \leq \max\left[\frac{1}{\epsilon}\log\left(\frac{1}{\delta}\right), \frac{VC(C)-1}{32\epsilon}\right]$$

(lower bound)

$VC(H) = 4$
$\delta = 0.01$
$\epsilon = 0.05$
} from example

$$m \geq 5146$$

Script for solving problem 2:

```
from __future__ import division
from __future__ import print_function
import math

def distance(x1, x2, y1, y2):
    d = math.fabs(x1-x2) + math.fabs(y1-y2)
    return d

data = {1: [2, 3, 1],
        2: [4, 4, 1],
        3: [4, 5, 0],
        4: [6, 3, 1],
        5: [8, 3, 0],
        6: [8, 4, 0]}
ks = [1, 2, 3]
accuracy = 0.0
pref_k = 0
for k in ks:
    correct = 0
    for inst in data.keys():
        # testing instance
        test = data[inst]
        temp = dict(data)
        del temp[inst]
        # get dist from testing instance to all training instances
        dist = {}
        for key, value in temp.iteritems():
            x1 = test[0]
            y1 = test[1]
            x2 = value[0]
            y2 = value[1]
            d = distance(x1, x2, y1, y2)
            dist[key] = d
        # get k nearest neighbor instances
        nn = []
        for i in range(1, k+1):
            near_val = min(dist.values())
            near_inst = [ins for ins, d in dist.iteritems() if d == near_val]
            for num in near_inst:
                # tie breaking is just first-in wins (i.e. lower instance #)
                if len(nn) == k:
```

```
                break
            else:
                pn = temp[num]
                nn.append(pn[2])
                dist.pop(num)
    # counter for if nearest neighbor(s) produces correct prediction
    for n in nn:
        if test[2] == n:
            correct = correct + 1
frac = correct / (k*6)
print(k, correct, frac)
```

Tables for problem 3

P(X=0)	*P(X=1)*	*P(Y=0)*	*P(Y=1)*	*P(Z=0)*	*P(Z=1)*
0	**36**	**0**	**36**	**0**	**36**
0	**4**	**0**	**4**	**4**	**0**
0	**2**	**2**	**0**	**0**	**2**
0	**8**	**8**	**0**	**8**	**0**
9	**0**	**0**	**9**	**0**	**9**
1	**0**	**0**	**1**	**1**	**0**
8	**0**	**8**	**0**	**0**	**8**
32	**0**	**32**	**0**	**32**	**0**
0.5	***0.5***	***0.5***	***0.5***	***0.45***	***0.55***

P(X=0,Z=0)	*P(X=0,Z=1)*	*P(X=1,Z=0)*	*P(X=1,Z=1)*
0	**0**	**0**	**36**
0	**0**	**4**	**0**
0	**0**	**0**	**2**
0	**0**	**8**	**0**
0	**9**	**0**	**0**
1	**0**	**0**	**0**
0	**8**	**0**	**0**
32	**0**	**0**	**0**
0.33	***0.17***	***0.12***	***0.38***

P(Y=0,Z=0)	*P(Y=0,Z=1)*	*P(Y=1,Z=0)*	*P(Y=1,Z=1)*
0	**0**	**0**	**36**
0	**0**	**4**	**0**
0	**2**	**0**	**0**
8	**0**	**0**	**0**
0	**0**	**0**	**9**
0	**0**	**1**	**0**
0	**8**	**0**	**0**
32	**0**	**0**	**0**

0.4	*0.1*	*0.05*	*0.45*

P(X=0,Y=0)	*P(X=0,Y=1)*	*P(X=1,Y=0)*	*P(X=1,Y=1)*
0	**0**	**0**	**36**
0	**0**	**0**	**4**
0	**0**	**2**	**0**
0	**0**	**8**	**0**
0	**9**	**0**	**0**
0	**1**	**0**	**0**
8	**0**	**0**	**0**
32	**0**	**0**	**0**
0.4	*0.1*	*0.1*	*0.4*