# L0(1)

October 22, 2020

# 1

$$Recall = \frac{TP}{TP + FN} = \frac{C(F_s F_r)}{C(F_s F_r) + C(L_s F_r)}$$

$$Precision = \frac{TP}{TP + FP * q} = \frac{C(F_s F_r)}{C(F_s F_r) + C(F_s L_r) * q}$$

()

[466]:
```
import matplotlib.pyplot as plt
import random
import csv
```

,

[467]:
```
def data_mining(file_name):
    with open(file_name) as file:
        #                    ,
        csv = file.read().split('\n')
        csv.pop()
        #                              -
        csv.pop(12313)

        tmp = [j for j in csv.pop(0).split(';')]
        AMOUNT,p1,p2,p3,p4,p5,CLASS=4,25,26,27,28,29,30
        data =␣
↪[[tmp[AMOUNT],tmp[p1],tmp[p2],tmp[p3],tmp[p4],tmp[p5],tmp[CLASS]]]

        for i in csv:
            tmp = [j for j in i.split(';')]
            tmp =␣
↪[tmp[AMOUNT],tmp[p1],tmp[p2],tmp[p3],tmp[p4],tmp[p5],tmp[CLASS]]
            data.append(tmp)
        return data
```

[468]:
```
data = data_mining('     .csv')
```

[469]:
```
print('       =',len(data))
```

```
       = 29304
```

[470]:
```
def recall_precision(data,cut_off=0.5,q=10):
        answer= [['TPR(recall)','precision','FPR']]
        for i in range(1,6):
            TP = 0
            FN = 0
            TN = 0
            FP = 0
            for j in range(1,len(data)):
                if (float(data[j][i])>=cut_off) and (str(data[j][6])=='F'):
                    TP+=1
                elif (float(data[j][i])>=cut_off) and (str(data[j][6])=='G'):
                    FP+=1
                elif (float(data[j][i])<cut_off) and (str(data[j][6])=='G'):
                    TN+=1
                elif (float(data[j][i])<cut_off) and (str(data[j][6])=='F'):
                    FN+=1
            try:
                recall=TP/(TP+FN)
            except ZeroDivisionError:
                recall='0'
            try:
                precision=TP/(TP+FP*q)
            except ZeroDivisionError:
                precision='0'
            try:
                FPR=FP/(FP+TN)
            except ZeroDivisionError:
                FPR='0'
            answer.append([recall,precision,FPR])


        return answer
```

[477]:
```
answer_d={}
cut_off_mass = [i/100 for i in range(1,100)]
for cut_off in cut_off_mass:
    answer_d[cut_off]=recall_precision(data,cut_off,q=10)
```

[478]:
```
for cut_off in [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,0.95,0.99]:
    print('cut_off=',cut_off,'\n',answer_d[cut_off],'\n')
```

```
cut_off= 0.1
 [['TPR(recall)', 'precision', 'FPR'], [0.9791623470868754, 0.05749505402526252,
0.8096532970768185], [1.0, 0.0480196738286306, 1.0], [0.9781256479369687,
0.05705560426934357, 0.8154055326047168], [1.0, 0.0480196738286306, 1.0],
[0.9810284055567075, 0.056132587508823543, 0.832087015635622]]

cut_off= 0.2
 [['TPR(recall)', 'precision', 'FPR'], [0.8959154053493676, 0.05967737480319311,
0.7120744653035611], [0.9831018038565208, 0.05624455080870447,
0.832087015635622], [0.9569769852788721, 0.06560720961471489,
0.6874967316843591], [1.0, 0.04802445533118254, 0.9998954138994928],
[0.9634045200082936, 0.0621967298695562, 0.7327302201537416]]

cut_off= 0.3
 [['TPR(recall)', 'precision', 'FPR'], [0.7490151358075886, 0.05964419862137285,
0.5956701354390002], [0.9197594857972217, 0.0654074696627888,
0.6629189980651572], [0.9362430022807381, 0.07625537232650235,
0.5720859697746169], [0.9478540327596932, 0.06592733067499261,
0.6774041729854102], [0.9478540327596932, 0.06592733067499261,
0.6774041729854102]]

cut_off= 0.4
 [['TPR(recall)', 'precision', 'FPR'], [0.5844909807173958, 0.06205970412116943,
0.445589081211107], [0.8557951482479784, 0.07109943585547565,
0.5639805469853056], [0.9002695417789758, 0.09108071824131565,
0.4531715734978821], [0.8178519593613933, 0.0786548220819749,
0.4832400773937144], [0.9172714078374455, 0.07184267363874047,
0.597761857449145]]

cut_off= 0.5
 [['TPR(recall)', 'precision', 'FPR'], [0.2737922454903587, 0.03400677302635815,
0.39230246300266697], [0.7220609579100146, 0.07834205050334626,
0.4284892537781729], [0.8440804478540328, 0.11899681389026921,
0.31522250692882914], [0.5566037735849056, 0.07542955085067225,
0.34414056371908175], [0.8680281982168775, 0.07673696076544499,
0.5268001882549809]]

cut_off= 0.6
 [['TPR(recall)', 'precision', 'FPR'], [0.262699564586357, 0.03709885219020848,
0.34393139151806723], [0.6425461331121709, 0.07631313255682239,
0.39230246300266697], [0.7660170018660585, 1.0, 0.0], [0.04281567489114659,
0.05725772910023569, 0.03555927417246248], [0.8178519593613933,
0.0786626648984435, 0.48318778434346077]]
```

```
cut_off= 0.7
 [['TPR(recall)', 'precision', 'FPR'], [0.0011403690648973668,
0.18032786885245902, 0.0002614652512681065], [0.5578478125647937,
0.07562788998046409, 0.34393139151806723], [0.6660792038150529, 1.0, 0.0],
[0.04271200497615592, 0.05720633157456262, 0.03550698112220886],
[0.7133526850507983, 0.07766278032979312, 0.4273388066725932]]

cut_off= 0.8
 [['TPR(recall)', 'precision', 'FPR'], [0.0, '0', 0.0], [0.0025917478747667427,
0.3333333333333333, 0.0002614652512681065], [0.5757827078581795, 1.0, 0.0],
[0.04250466514617458, 0.05694444444444444, 0.03550698112220886],
[0.6376736471076093, 0.07577829520395216, 0.39230246300266697]]

cut_off= 0.9
 [['TPR(recall)', 'precision', 'FPR'], [0.0, '0', 0.0], [0.0, '0', 0.0],
[0.29421521874352063, 1.0, 0.0], [0.04250466514617458, 0.057103064066852366,
0.035402395021701616], [0.5562927638399336, 0.07543297345928925,
0.343931391518067 23]]

cut_off= 0.95
 [['TPR(recall)', 'precision', 'FPR'], [0.0, '0', 0.0], [0.0, '0', 0.0], [0.0,
'0', 0.0], [0.04105328633630521, 0.05852793378657996, 0.03331067301155676],
[0.4713871034625752, 0.0749007527962179, 0.2936777022433716]]

cut_off= 0.99
 [['TPR(recall)', 'precision', 'FPR'], [0.0, '0', 0.0], [0.0, '0', 0.0], [0.0,
'0', 0.0], [0.016794526228488493, 0.07706945765937202, 0.010144851749202531],
[0.295355587808418, 0.09169912131063118, 0.14757098781571928]]
```

# 2

$$Recall = \frac{TP}{TP + FN} = \frac{S(F_s F_r)}{S(F_s F_r) + S(L_s F_r)}$$

$$Precision = \frac{TP}{TP + FP * q} = \frac{S(F_s F_r)}{S(F_s F_r) + S(F_s L_r) * q}$$

S()-

[479]:
```
def recall_precision_money(data,cut_off=0.5,q=10):
        answer=[['TPR(recall_m)','precision_m','FPT_m']]

        for i in range(1,6):
            TP = 0
            FN = 0
            TN = 0
            FP = 0
```

```
            for j in range(1,len(data)):
                if data[j][0] != '':
                    sum = float(data[j][0])
                if (float(data[j][i])>=cut_off) and (str(data[j][6])=='F'):
                    TP+=sum
                elif (float(data[j][i])>=cut_off) and (str(data[j][6])=='G'):
                    FP+=sum
                elif (not(float(data[j][i])>=cut_off)) and␣
↪(str(data[j][6])=='G'):
                    TN+=sum
                elif (not(float(data[j][i])>=cut_off)) and␣
↪(str(data[j][6])=='F'):
                    FN+=sum
            try:
                recall=TP/(TP+FN)
            except ZeroDivisionError:
                recall='0'
            try:
                precision=TP/(TP+FP*q)
            except ZeroDivisionError:
                precision='0'
            try:
                FPR=FP/(FP+TN)
            except ZeroDivisionError:
                FPR='0'
            answer.append([recall,precision,FPR])

        return answer
```

[480]:
```
answer_m_d= {}
cut_off_mass_m = [i/100 for i in range(1,100)]
for cut_off in cut_off_mass_m:
    answer_m_d[cut_off] = recall_precision_money(data,cut_off,q=10)
```

[481]:
```
for cut_off in [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,0.95,0.99]:
    print('cut_off=',cut_off,'\n',answer_m_d[cut_off],'\n')
```

```
cut_off= 0.1
 [['TPR(recall_m)', 'precision_m', 'FPT_m'], [0.9611527195431848,
0.0517687399162424, 0.7568764547592146], [1.0, 0.041219717940729914, 1.0],
[0.9606739435808853, 0.05089141572953319, 0.7702528290603144], [1.0,
0.041219717940729914, 1.0], [0.9638613317244922, 0.048385072232333175,
0.8149863439192119]]

cut_off= 0.2
```

```
 [['TPR(recall_m)', 'precision_m', 'FPT_m'], [0.81678517285839,
0.06041458963137132, 0.5461201512219054], [0.9656096581162038,
0.04846858300111809, 0.8149863439192119], [0.9238082226019778,
0.06695580078097256, 0.5534542846049282], [1.0, 0.041220055811897124,
0.9999914508396527], [0.9302527728191226, 0.06213056802034052,
0.6037038589862197]]

cut_off= 0.3
 [['TPR(recall_m)', 'precision_m', 'FPT_m'], [0.6189048533047601,
0.05964233841460968, 0.4195156963721876], [0.8516494433240163,
0.0648191696490183, 0.5282492890893146], [0.8749234169356489,
0.08348051857226926, 0.4129643145696993], [0.8916700362364741,
0.07034261582601017, 0.5066342002374059], [0.8916700362364741,
0.07034261582601017, 0.5066342002374059]]

cut_off= 0.4
 [['TPR(recall_m)', 'precision_m', 'FPT_m'], [0.34471347110995054,
0.05187538762728368, 0.27086210111327846], [0.7598642724511975,
0.07666833017248294, 0.3934265589358551], [0.7919630260554712,
0.10285594827974293, 0.29697753463835047], [0.6903488899476272,
0.09138489486672956, 0.29509380238843735], [0.8407028104446791,
0.08110716642605337, 0.4094812839858981]]

cut_off= 0.5
 [['TPR(recall_m)', 'precision_m', 'FPT_m'], [0.1348273674362985,
0.02403058062238027, 0.23541597068415349], [0.5465374534411631,
0.08320895524603808, 0.2588845565861145], [0.717569574592336,
0.14478836285667895, 0.1822174153515674], [0.312259479151668,
0.06253526150189806, 0.20124797715517181], [0.745196034918961,
0.08780303385911312, 0.3328400344648049]]

cut_off= 0.6
 [['TPR(recall_m)', 'precision_m', 'FPT_m'], [0.11307674671142583,
0.023596050277863323, 0.20116361963201562], [0.434196753188775,
0.07346780833548236, 0.23541597068415349], [0.6084096745609308, 1.0, 0.0],
[0.03787087552239329, 0.035698721221001234, 0.0439796076495234],
[0.6903488899476272, 0.0913868193621236, 0.2950869630601596]]

cut_off= 0.7
 [['TPR(recall_m)', 'precision_m', 'FPT_m'], [0.0018586225324483995,
0.24380738731218699, 0.0002478350289716023], [0.31318994719993015,
0.06273456555005226, 0.20116361963201562], [0.4490645715358533, 1.0, 0.0],
[0.037858944194215455, 0.03569657442302823, 0.04396849374107192],
[0.5262536972468168, 0.08070921920869327, 0.25769787696192775]]

cut_off= 0.8
 [['TPR(recall_m)', 'precision_m', 'FPT_m'], [0.0, '0', 0.0],
[0.011966000378888007, 0.6748746307900155, 0.0002478350289716023],
```

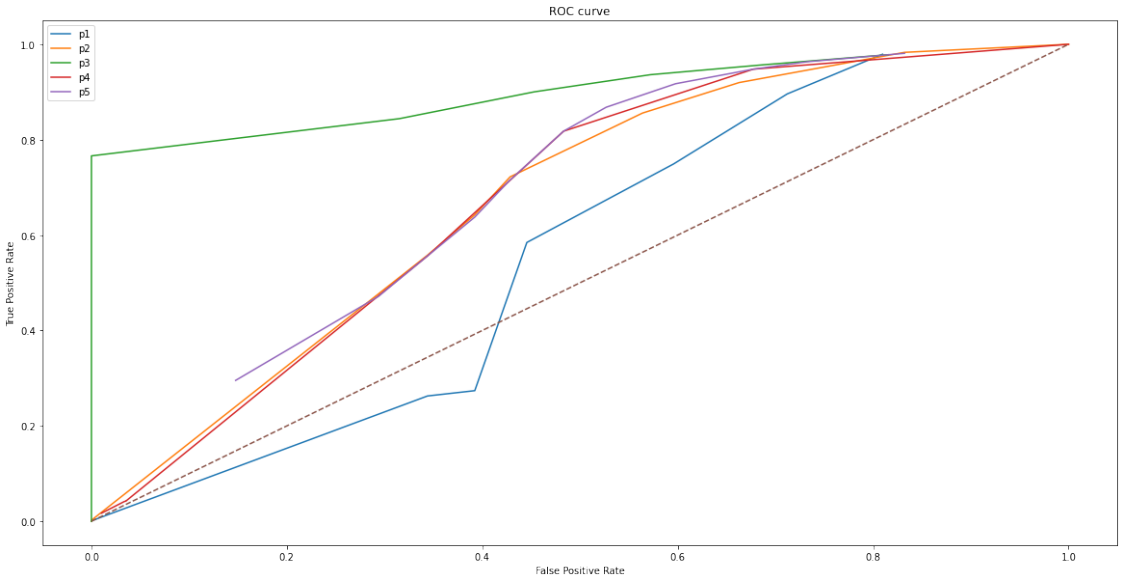
```
[0.34316834999218326, 1.0, 0.0], [0.03783766665896499, 0.03567722795078604,
0.04396849374107192], [0.4204670946216958, 0.07131035777768041,
0.23541597068415349]]

cut_off= 0.9
 [['TPR(recall_m)', 'precision_m', 'FPT_m'], [0.0, '0', 0.0], [0.0, '0', 0.0],
[0.13999321541806564, 1.0, 0.0], [0.03783766665896499, 0.03572926360440266,
0.04390208945464508], [0.31222627028823974, 0.06255360788775319,
0.20116361963201562]]

cut_off= 0.95
 [['TPR(recall_m)', 'precision_m', 'FPT_m'], [0.0, '0', 0.0], [0.0, '0', 0.0],
[0.0, '0', 0.0], [0.035015859779824444, 0.03528871987344086,
0.04115401093319049], [0.24647270069535948, 0.061620736996022635,
0.16136385444766513]]

cut_off= 0.99
 [['TPR(recall_m)', 'precision_m', 'FPT_m'], [0.0, '0', 0.0], [0.0, '0', 0.0],
[0.0, '0', 0.0], [0.010699098823644799, 0.03276641468416771,
0.013577992535037667], [0.14091886376653923, 0.06613375958093261,
0.08554931138239481]]
```

# 3

## 0.1 ROC 1

[482]:
```
FPR=[[]for i in range(5)]
TPR=[[]for i in range(5)]
for i in range(1,6):
    for cut_off in [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,0.95,0.99]:
        FPR[i-1].append(float(answer_d[cut_off][i][2]))
        TPR[i-1].append(float(answer_d[cut_off][i][0]))

plt.gcf().clear()
plt.figure(figsize=(20,10))
plt.title('ROC curve')
plt.plot(FPR[0], TPR[0], label='p1')
plt.plot(FPR[1], TPR[1], label='p2')
plt.plot(FPR[2], TPR[2], label='p3')
plt.plot(FPR[3], TPR[3], label='p4')
plt.plot(FPR[4], TPR[4], label='p5')
plt.plot([0, 1], [0, 1], '--')
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend()
plt.show()
```

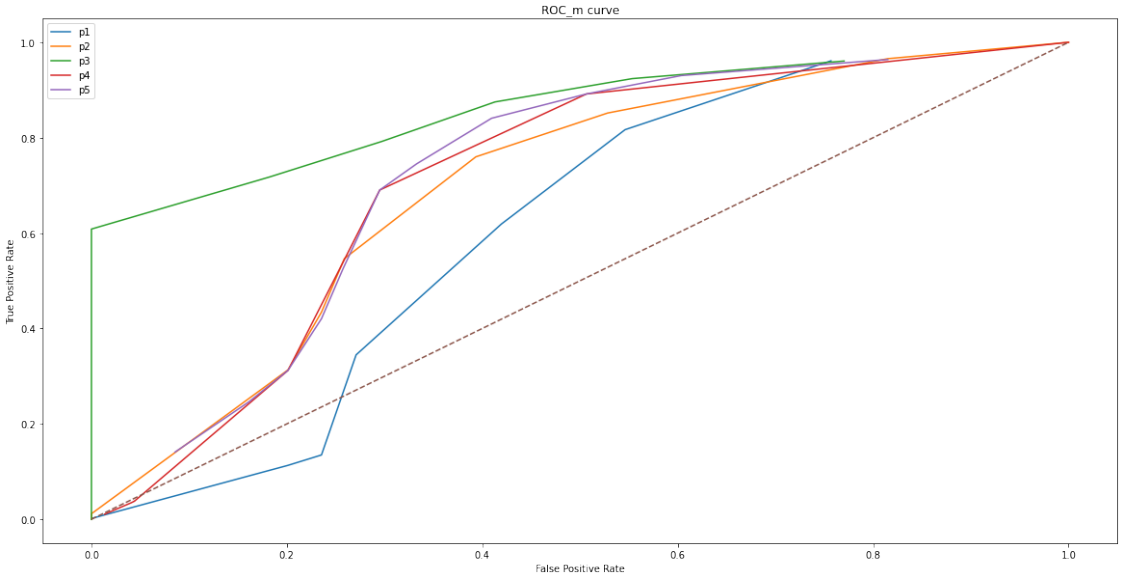
<Figure size 432x288 with 0 Axes>




## 0.2 ROC_m 2

[483]:
```
FPR_m=[[]for i in range(5)]
TPR_m=[[]for i in range(5)]
for i in range(1,6):
    for cut_off in [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,0.95,0.99]:
        FPR_m[i-1].append(float(answer_m_d[cut_off][i][2]))
        TPR_m[i-1].append(float(answer_m_d[cut_off][i][0]))

plt.gcf().clear()
plt.figure(figsize=(20,10))
plt.title('ROC_m curve')
plt.plot(FPR_m[0], TPR_m[0], label='p1')
plt.plot(FPR_m[1], TPR_m[1], label='p2')
plt.plot(FPR_m[2], TPR_m[2], label='p3')
plt.plot(FPR_m[3], TPR_m[3], label='p4')
plt.plot(FPR_m[4], TPR_m[4], label='p5')
plt.plot([0, 1], [0, 1], '--')
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend()
plt.show()
```

<Figure size 432x288 with 0 Axes>

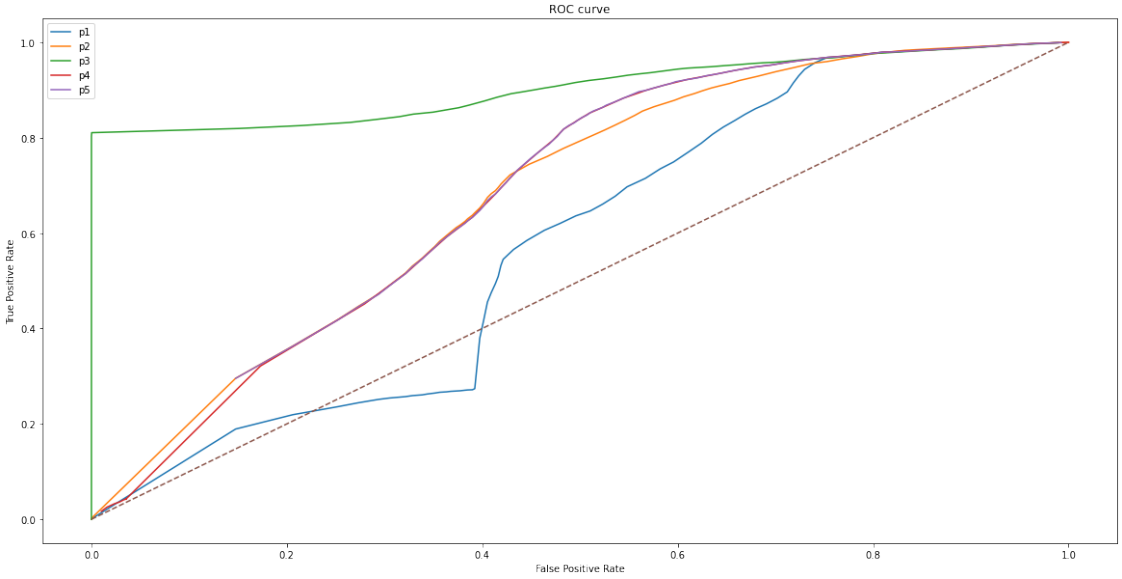



# 4

### 0.3 ROC 1

[484]:
```
FPR=[[]for i in range(5)]
TPR=[[]for i in range(5)]
cut_off_mass = [i/100 for i in range(1,100)]
for i in range(1,6):
    for cut_off in cut_off_mass:
        FPR[i-1].append(float(answer_d[cut_off][i][2]))
        TPR[i-1].append(float(answer_d[cut_off][i][0]))

plt.gcf().clear()
plt.figure(figsize=(20,10))
plt.title('ROC curve')
plt.plot(FPR[0], TPR[0], label='p1')
plt.plot(FPR[1], TPR[1], label='p2')
plt.plot(FPR[2], TPR[2], label='p3')
plt.plot(FPR[3], TPR[3], label='p4')
plt.plot(FPR[4], TPR[4], label='p5')
plt.plot([0, 1], [0, 1], '--')
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend()
plt.show()
```

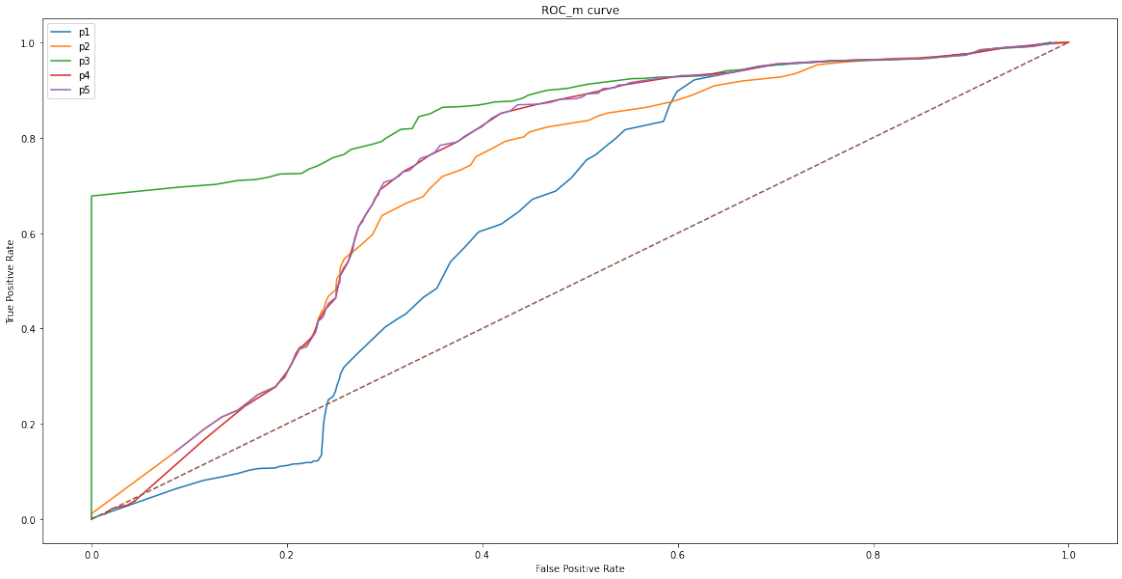
```
<Figure size 432x288 with 0 Axes>
```




### 0.4 ROC_m 2

[485]:
```
FPR_m=[[]for i in range(5)]
TPR_m=[[]for i in range(5)]
cut_off_mass_m = [i/100 for i in range(1,100)]
for i in range(1,6):
    for cut_off in cut_off_mass_m:
        FPR_m[i-1].append(float(answer_m_d[cut_off][i][2]))
        TPR_m[i-1].append(float(answer_m_d[cut_off][i][0]))

plt.gcf().clear()
plt.figure(figsize=(20,10))
plt.title('ROC_m curve')
plt.plot(FPR_m[0], TPR_m[0], label='p1')
plt.plot(FPR_m[1], TPR_m[1], label='p2')
plt.plot(FPR_m[2], TPR_m[2], label='p3')
plt.plot(FPR_m[3], TPR_m[3], label='p4')
plt.plot(FPR_m[4], TPR_m[4], label='p5')
plt.plot([0, 1], [0, 1], '--')
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend()
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```

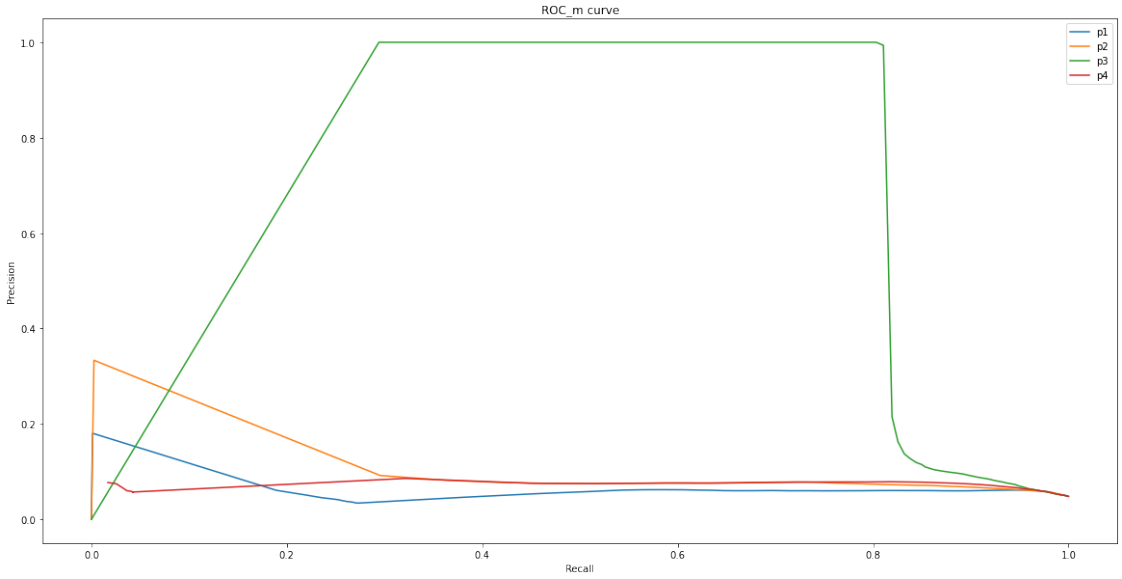



# 5

##Precision-recall

[486]:
```
recall=[[]for i in range(5)]
precs=[[]for i in range(5)]
cut_off_mass = [i/100 for i in range(1,100)]
for i in range(1,6):
    for cut_off in cut_off_mass_m:
        recall[i-1].append(float(answer_d[cut_off][i][0]))
        precs[i-1].append(float(answer_d[cut_off][i][1]))

plt.gcf().clear()
plt.figure(figsize=(20,10))
plt.title('ROC_m curve')
plt.plot(recall[0], precs[0], label='p1')
plt.plot(recall[1], precs[1], label='p2')
plt.plot(recall[2], precs[2], label='p3')
plt.plot(recall[3], precs[3], label='p4')
plt.ylabel('Precision')
plt.xlabel('Recall')
plt.legend()
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```




# 6

[487]:
```
5
```

```
  File "<ipython-input-487-d7631523e207>", line 1
    5
    ^
SyntaxError: invalid syntax
```

[488]:
```
with open('data.csv', 'w') as file:
    writer = csv.writer(file,delimiter=';',lineterminator='\n')
    writer.writerows(data)

cut_off_mass = [i/100 for i in range(1,100)]
with open('answer_d.csv', 'w') as file:
    writer = csv.writer(file,delimiter=';',lineterminator='\n')
    for cut_off in cut_off_mass:
        tmp = [[cut_off]]
        writer.writerows(tmp)
        writer.writerows(answer_d[cut_off])

with open('ROC.csv', 'w') as file:
    writer = csv.writer(file,delimiter=';',lineterminator='\n')
    writer.writerows([['X_FPR']])
    writer.writerows(FPR)
    writer.writerows([['Y_TPR']])
```

```
    writer.writerows(TPR)

with open('ROC_M.csv', 'w') as file:
    writer = csv.writer(file,delimiter=';',lineterminator='\n')
    writer.writerows([['X_FPR_M']])
    writer.writerows(FPR_m)
    writer.writerows([['Y_TPR_M']])
    writer.writerows(TPR_m)

with open('Precision-recall', 'w') as file:
    writer = csv.writer(file,delimiter=';',lineterminator='\n')
    writer.writerows([['X_recall']])
    writer.writerows(recall)
    writer.writerows([['Y_precision']])
    writer.writerows(precs)
```

# 7

$$p = \frac{A * p1 + B * p2 + C * p3}{A + B + C}$$

[489]:
```
data_ans = [['AMOUNT','p_fraud','CLASS']]
a = 0
b = 0.01
c = 0.99
for i in range(1,len(data)):
    p = (a*float(data[i][1])+b*float(data[i][2])+c*float(data[i][3]))/(a+b+c)
    data_ans.append([data[i][0],p,data[i][6]])
```

[490]:
```
def recall_precision_ans(data,cut_off=0.5,q=10):
        answer= [['TPR(recall)_ans','precision_ans','FPR_ans']]
        for i in [1]:
            TP = 0
            FN = 0
            TN = 0
            FP = 0
            for j in range(1,len(data)):
                if (float(data[j][i])>=cut_off) and (str(data[j][2])=='F'):
                    TP+=1
                elif (float(data[j][i])>=cut_off) and (str(data[j][2])=='G'):
                    FP+=1
                elif (float(data[j][i])<cut_off) and (str(data[j][2])=='G'):
                    TN+=1
                elif (float(data[j][i])<cut_off) and (str(data[j][2])=='F'):
                    FN+=1
            try:
                recall=TP/(TP+FN)
```

```
            except ZeroDivisionError:
                recall='0'
            try:
                precision=TP/(TP+FP*q)
            except ZeroDivisionError:
                precision='0'
            try:
                FPR=FP/(FP+TN)
            except ZeroDivisionError:
                FPR='0'
            answer.append([recall,precision,FPR])


        return answer
```

[491]:
```
answer_ans={}
cut_off_mass = [i/100 for i in range(1,100)]
for cut_off in cut_off_mass:
    answer_ans[cut_off]=recall_precision_ans(data_ans,cut_off,q=10)
```

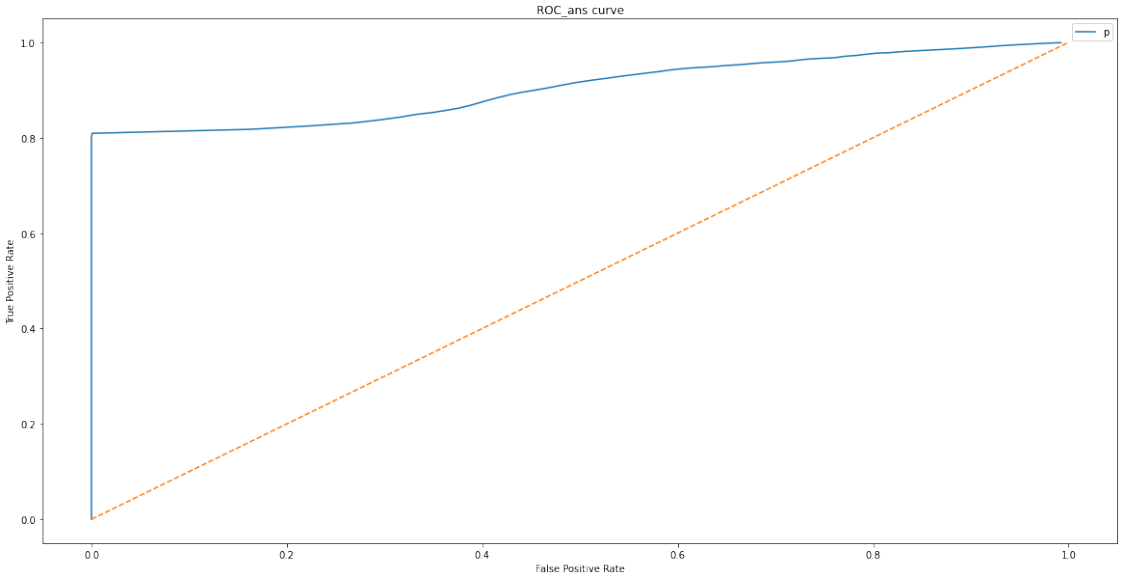
[492]:
```
for cut_off in [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,0.95,0.99]:
    print('cut_off=',cut_off,'\n',answer_ans[cut_off],'\n')
```

```
cut_off= 0.1
 [['TPR(recall)_ans', 'precision_ans', 'FPR_ans'], [0.9782293178519593,
0.05696485275829178, 0.8168697380118183]]

cut_off= 0.2
 [['TPR(recall)_ans', 'precision_ans', 'FPR_ans'], [0.9573916649388348,
0.06551271592239208, 0.6888563509909533]]

cut_off= 0.3
 [['TPR(recall)_ans', 'precision_ans', 'FPR_ans'], [0.9367613518556914,
0.07615291262135922, 0.5732364168801967]]

cut_off= 0.4
 [['TPR(recall)_ans', 'precision_ans', 'FPR_ans'], [0.899232842629069,
0.0910904813912459, 0.4525963499450923]]

cut_off= 0.5
 [['TPR(recall)_ans', 'precision_ans', 'FPR_ans'], [0.8431474186191167,
0.11865544256889836, 0.31590231658212625]]

cut_off= 0.6
 [['TPR(recall)_ans', 'precision_ans', 'FPR_ans'], [0.7629069044163383, 1.0,
0.0]]
```

```
cut_off= 0.7
 [['TPR(recall)_ans', 'precision_ans', 'FPR_ans'], [0.6635911258552768, 1.0,
0.0]]

cut_off= 0.8
 [['TPR(recall)_ans', 'precision_ans', 'FPR_ans'], [0.5745386688782915, 1.0,
0.0]]

cut_off= 0.9
 [['TPR(recall)_ans', 'precision_ans', 'FPR_ans'], [0.18401409910843874, 1.0,
0.0]]

cut_off= 0.95
 [['TPR(recall)_ans', 'precision_ans', 'FPR_ans'], [0.0, '0', 0.0]]

cut_off= 0.99
 [['TPR(recall)_ans', 'precision_ans', 'FPR_ans'], [0.0, '0', 0.0]]
```

[493]:
```
FPR_ans=[]
TPR_ans=[]
cut_off_mas = [i/100 for i in range(1,100)]
for i in [1]:
    for cut_off in cut_off_mass:
        FPR_ans.append(float(answer_ans[cut_off][i][2]))
        TPR_ans.append(float(answer_ans[cut_off][i][0]))

plt.gcf().clear()
plt.figure(figsize=(20,10))
plt.title('ROC_ans curve')
plt.plot(FPR_ans, TPR_ans, label='p')
plt.plot([0, 1], [0, 1], '--')
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend()
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```




$$G = 2AUC - 1$$

[494]:
```
G = 0
AUC = 0
for i in range(0,len(cut_off_mass)-1):
    AUC += abs(FPR_ans[i]-FPR_ans[i+1])*(TPR_ans[i]+ TPR_ans[i+1])/2
G = 2*AUC-1
print('AUC=',AUC)
print('G=',G)
```

```
AUC= 0.8956699625022073
G= 0.7913399250044146
```

, G= 0.7913399 (0,0.01,0.99). 3
, ,
( FPR )
G= 0.1331553 a,b,c=1,0,0 . 1 .
.

TPR FPR=0.1 .
3 , a,b,c=0,0,1 TPR FPR=0.1 .
FPR=0.1 cut_off = 0.54