물리화학적 특징의 귀납적 방식을 통한 마약 물질 분류

Narcotic drugs(마약)

마약류(drugs)란 일반적으로 느낌, 생각 또는 행태에 변화를 줄 목적으로 섭취하는 **정신에 영향을 주는 물질(psychoactive substance)**를 말한다.

분류

1. 효능에 의한 분류
   - 억제제 : 중추 신경계의 기능을 저하
   - 각성제 : 중추 신경계의 활동을 강화
   - 환각제 : 감각의 왜곡

2. 법률에 의한 분류

But 화학적 특성에 대한 마약성 물질의 원천적 구별은 불가능

논문 분석

Chem. Res. Toxicol. **2005,** *18,* 536−555

Structural Alerts—A New Classification Model to Discriminate Excess Toxicity from Narcotic Effect Levels of Organic Compounds in the Acute Daphnid Assay

Peter C. von der Ohe, Ralph Kühne, Ralf-Uwe Ebert, Rolf Altenburger, Matthias Liess, and Gerrit Schüürmann*

Department of Chemical Ecotoxicology, UFZ Centre for Environmental Research, Permoserstrasse 15, D-04318 Leipzig, Germany

Received July 27, 2004

Abstract

Quantitative and qualitative structure-activity relationships (QSARs) have a great potential to support the risk assessment of chemicals, provided there are tools available that allow evaluation of the suitability of QSARs for the compounds of interest. In this context, a pragmatic approach is to discriminate excess toxicity from narcotic effect levels, because the latter can be estimated from QSARs and thus have a low priority for experimental testing. To develop a respective scheme for the acute daphnid toxicity as one of the primary ecotoxicological endpoints, 1067 acute toxicity data entries for 380 chemicals involving the daphnid species Daphnia magna were taken from the on-line literature, and quality checks such as water solubility were employed to eliminate apparently odd data entries. For 36 known narcotics with LC50 values referring to D. magna, a reference baseline QSAR is derived. Compounds with LC50 values above a certain threshold defined relative to their predicted baseline toxicity are classified as exerting excess toxicity. Three simple discrimination schemes are presented that enable the identification of excess toxicity from structural alerts based on the presence or absence of certain heteroatoms and their chemical functionality. Moreover, a two-step classification approach is introduced that enables a prioritization of organic compounds with respect to their need for experimental testing. The discussion includes reaction mechanisms that may explain the association of structural alerts with excess toxicity, a comparison with predictions derived from mode of action-based classification schemes, and a statistical analysis of the discrimination performance in terms of detailed contingency table statistics.

What does LC_{50} mean?

LC stands for "Lethal Concentration". LC values usually refer to the concentration of a chemical in air but in environmental studies it can also mean the concentration of a chemical in water.

According to the (Organisation for Economic Cooperation and Development) (OECD) Guidelines for the Testing of Chemicals, a traditional experiment involves groups of animals exposed to a concentration (or series of concentrations) for a set period of time (usually 4 hours). The animals are clinically observed for up to 14 days.

The concentrations of the chemical in air that kills 50% of the test animals during the observation period is the LC_{50} value. Other durations of exposure (versus the traditional 4 hours) may apply depending on specific laws.

Table 1. Compounds with 48 h Daphnia Toxicity in Terms of Log LC_{50}, Log K_{ow}, Log T_e, and the Prediction Results of Six CMs[a]

no.	CAS	name	log LC_{50} (mol/L)	log K_{ow}	log T_e	CM1	CM2	CM3	4-MOA CM	7-MOA CM	2-MOA CM
				training set							
1	50293	DDT	−7.89	6.79	0.79	0	0	0	4	7	0
2	51285	2,4-dinitrophenol	−4.62	1.73	1.86	1	0	0	3	4	0
3	52686	trichlorofon	−6.31	−0.28	5.27	1	0	3.2	NA	5/6	0
4	55389	fenthion	−6.79	4.08	2.01	1	3.1	3.1	4	6	0
5	55630	nitroglycerine	−3.85	1.51	1.28	1	0	0	NA	1	0
6	56382	parathion	−8.17	3.73	3.70	1	3.1	3.1	4	6	0
7	58140	pyrimethamine	−4.63	2.41	1.29	1	0	0	NA	1	0
8	58899	lindane	−5.39	4.26	0.46	0	0	0	1	7	0
9	58902	2,3,4,6-tetrachlorophenol	−6.12	4.09	1.34	0	0	0	NA	4	0
10	59063	ethopabate	−3.07	1.90	0.17	1	0	0	3	3	0
11	59507	4-chloro-3-methylphenol	−4.85	2.70	1.26	0	0	0	2	2	0
12	60515	dimethoate	−4.94	0.28	3.42	1	3.1	3.1	4	6	0
13	60571	dieldrin	−6.28	5.45	0.33	0	0	0	3	5/7	1
14	62533	aniline	−5.33	1.08	3.13	1	8.1	8.1	2	2	0
15	62555	thioacetamide	−3.64	−0.83	3.07	1	0	0	NA	1	0
16	62566	thiourea	−3.84	−1.31	3.68	1	7.1	7.1	NA	1	0
17	62737	dichlorvos	−9.10	0.60	7.30	1	2.1	2.1	NA	6	1
18	63252	carbaryl	−7.33	2.35	4.04	1	6.1	6.1	NA	1	0
19	68122	*N*,*N*-dimethylformamide	−0.70	−0.93	0.22	1	0	0	NA	5	0
20	72208	endrin	−6.38	5.45	0.43	0	0	0	3	5/7	1
21	74839	methyl bromide	−4.63	1.18	2.34	0	0	0	1	1	0
22	75058	acetonitrile	−1.06	−0.15	−0.10	1	0	0	NA	1	0
23	75070	acetaldehyde	−0.55	−0.17	−0.59	0	0	0	3	5	1
24	75081	ethyl mercaptan	−5.56	1.27	3.19	1	4.1	4.1	NA	1	1
25	75150	carbon disulfide	−4.56	1.94	1.62	1	0	0	NA	1	0
26	75218	ethylene oxide	−2.32	−0.05	1.08	0	0	0	3	5	1
27	75252	bromoform	−3.74	1.79	0.92	0	0	0	1	1	0
28	75354	1,1-dichloroethene	−3.28	2.12	0.18	0	0	0	1	1	0
29	77474	hexachlorocyclopentadiene	−6.72	4.63	1.47	0	0	0	1/3	7	0
30	78591	isophorone	−3.06	2.62	−0.47	1	1.1	1.1	3	5	1
31	78999	1,1-dichloropropane	−3.57	2.25	0.36	0	0	0	1	1	0
32	79061	acrylamide	−2.65	−0.81	2.06	1	1.1	1.1	NA	5	1
33	79094	propionic acid	−3.17	0.58	1.39	0	0	0	NA	1	0
34	83410	1,2-dimethyl-3-nitrobenzene	−4.56	2.91	0.78	1	0	0	2	1	0
35	83421	2-chloro-6-nitrotoluene	−4.61	3.00	0.76	1	0	0	2	1	0
36	84662	diethyl phthalate	−3.61	2.65	0.06	0	0	0	3	3	0
37	84742	dibutyl phthalate	−4.88	4.61	−0.36	0	0	0	3	3	0
38	85018	phenanthrene	−5.36	4.35	0.35	0	0	0	1	1	0
39	85687	butyl benzyl phthalate	−5.19	4.84	−0.24	0	0	0	3	3	0
40	86306	*N*-nitrosodiphenylamine	−4.40	3.16	0.42	1	0	0	3	5	0

마약류 데이터 가져오기


한국마약퇴치운동본부
마약류폐해 알리미
마약류
마약류(임시 마약류 포함)로
지정 된 물질 분류
마약류
마약구분
마약류
마약
약리작용
전체
한글명
영문명
검색
총 129건
NO
한글명/영문명
마약류관리에관한법률
마약구분
129
테트라히드로푸라닐펜타닐
Tetrahydrofuranylfentanyl, THF-F
제2조2호마목에 해당하는 마약
마약류 > 마약
128
4-플루오로이소부티르펜타닐
4-Fluoroisobutyrfentanyl, 4-FIBF
제2조2호마목에 해당하는 마약
마약류 > 마약

비 마약류 데이터 가져오기

DRUGBANK

Drugs

DB00328	Indometacin	Oral indometacin is indicated for symptomatic management of moderate to severe rheumatoid arthritis including acute flares of chronic disease, moderate to severe ankylosing spondylitis, moderate to severe osteoarthritis, acute painful shoulder (bursitis and/or tendinitis) and acute gouty arthritis.[A177871,Label] Intravenous indometacin is indicated to induce closure of a hemodynamically significant patent ductus arteriosus in premature infants weighing between 500 and 1750 g when after 48 hours usual medical management (e.g., fluid restriction, diuretics, digitalis, respiratory support, etc.) is ineffective.[F4600]
DB00337	Pimecrolimus	For treatment of mild to moderate atopic dermatitis.
DB00411	Carbamoylcholine	Primarily used in the treatment of glaucoma, but is also used during ophthalmic surgery.
DB00461	Nabumetone	For acute and chronic treatment of signs and symptoms of osteoarthritis and rheumatoid arthritis.

Showing 1 to 10 of 158 entries

‹ 1 2 3 4 5 … 16 ›

Pubchempy library를 통해 화합물 들의 cid 정보 가져오기

```
#!/share/anaconda22/bin/python
import pubchempy
from pubchempy import get_cids

a =  raw_input("input filename : ")

file = open("%s.txt" % a, "a")
num_lines = sum(1 for line in open('%s.txt' % a))
if num_lines == 0:
        firstline = "%-15s%-15s%-15s%s"%("Number", "Pubchem CID", "Narcotic", "Name")
        file.write(firstline + '\n')
print("################################################")
comp = 1
while comp != "":
        print("........................................")
        comp = raw_input("input compound(Press Enter to quit) : ")
        if comp == "":
                print("check [%s.txt] on the current location" % a)
                print("################################################")
                break
        try:
                cid = str(get_cids(comp, "name")[0])

        except IndexError:
                print("Cannot find cid")
                continue


        inpt = "%-15d%-15s%-15s%s" % (num_lines ,cid, "1", comp)
        print("%-20s%s" %(comp, cid) )
        num_lines += 1
        file.write(inpt + '\n')

file.close()
```

Narcotics: 117

2	72287	0	methotrimeprazine
3	2206	0	antipyrine
4	5468	0	tiaprofenic acid
5	123619	0	etoricoxib
6	54445	0	castanospermine
7	445154	0	resveratrol
8	4641	0	oxyphenbutazone
9	4754	0	phenacetin
10	4488	0	niflumic acid
11	4495	0	nimesulide
12	39941	0	benoxaprofen
13	5733	0	zomepirac
14	40632	0	pirfenidone
15	445154	0	SRT501
16	67986221	0	PTC299
17	92337	0	tarenflurbil
18	11561674	0	apremilast
19	5318517	0	andrographolide
20	27400	0	pizotifen
21	6918173	0	icatibant
22	5472495	0	exisulind
23	16135415	0	ziconotide
24	948	0	nitrous oxide

nonNarcotics: 154

Number	Pubchem CID	Narcotic	Name
0	62156	1	Carfentanil
2	60575	1	Ocfentanil
3	13544015	1	U-47700
4	13653606	1	Furanylfentanyl
5	61996	1	3-Methylfentanyl
6	15129	1	Noracymethadol
7	5463854	1	Norlevorphanol
8	10090	1	Normethadone
9	5462508	1	Normorphine
10	9925873	1	Norcodeine
11	22391	1	Norpipanone
12	5464304	1	Nicodicodine
13	5362460	1	Nicomorphine
14	5463872	1	Nicocodine
15	5362456	1	Desomorphine
16	92943	1	Dextromoramide
17	5463863	1	Drotebanol
18	17036	1	Dimenoxadol
19	10668	1	Dimethylthiambutene
20	28397	1	Dimepheptanol
21	62370	1	Diampromide
22	6833	1	Diethylthiambutene
23	48194	1	Dioxaphetyl butyrate

총 271개 화합물

1. Narcotics의 파일을 가지고 SDF파일을 다운 받음

$ /share/bin/**downSDF Narcotics**
—> 결과 2D_Narcotics_Compounds.sdf 파일이 만들어짐

2. Cal_PaDel 프로그램을 이용하여 디스크립터 계산 및 csv파일로 정리

$ /share/bin/**Cal_PaDEL Narcotics**
—> 결과 Narcotics_2D_Descriptor.csv 파일이 만들어짐

In [1]:
```
import pandas as pd
```

In [17]:
```
df = pd.read_csv("./Narcotics_2D_Descriptor.csv")
```

In [27]:
```
df.head(5)
```

Out[27]:

	Name	nAcid	ALogP	ALogp2	AMR	apol	naAromAtom	nAromBond	nAtom	nHeavyAtom	...	WTPT-1	WTPT-2	WTPT-3	WTPT-4
0	62156	0	1.0696	1.144044	119.5096	66.849790	0	0	59	29	...	58.692103	2.023866	14.722511	7.740678
1	60575	0	0.8630	0.744769	110.1738	61.084411	0	0	54	27	...	54.924985	2.034259	14.790773	5.256255
2	13544015	0	0.3777	0.142657	85.0399	50.191446	0	0	43	21	...	41.649057	1.983288	13.879900	2.548358
3	13653606	0	1.0205	1.041420	118.9939	63.380618	0	0	54	28	...	57.839295	2.065689	12.612451	5.598455
4	61996	0	1.7143	2.938824	114.2398	63.485790	0	0	56	26	...	52.912561	2.035098	9.457515	2.505748

5 rows × 1446 columns

In [29]:
```
df['Narcotics'].head(5)
```

Out[29]:
```
0    1
1    1
2    1
3    1
4    1
Name: Narcotics, dtype: int64
```

1. 한국마약퇴치운동본부에서 마약류>마약에 해당하는 data 117개 수집
2. Drugbank에서 nonNarcotic 화합물 154개 수집
3. 마약은 endpoint 1, 비마약은 endpoint 0.

No.	Pubchem CID	Narcotic	Name
0	62156	1	Carfentanil
2	60575	1	Ocfentanil
3	13544015	1	U-47700
4	13653606	1	Furanylfentanyl
5	61996	1	3-Methylfentanyl
6	15129	1	Noracymethadol
7	5463854	1	Norlevorphanol
8	10090	1	Normethadone
9	5462508	1	Normorphine
10	9925873	1	Norcodeine
11	22391	1	Norpipanone
12	5464304	1	Nicodicodine
13	5362460	1	Nicomorphine
14	5463872	1	Nicocodine
15	5362456	1	Desomorphine
16	92943	1	Dextromoramide
17	5463863	1	Drotebanol

Number	Pubchem CID	Narcotic	Name
0	5161	0	salsalate
1	54682045	0	choline magnesium tris
2	72287	0	methotrimeprazine
3	2206	0	antipyrine
4	5468	0	tiaprofenic acid
5	123619	0	etoricoxib
6	54445	0	castanospermine
7	445154	0	resveratrol
8	4641	0	oxyphenbutazone
9	4754	0	phenacetin
10	4488	0	niflumic acid
11	4495	0	nimesulide
12	39941	0	benoxaprofen
13	5733	0	zomepirac
14	40632	0	pirfenidone
15	445154	0	SRT501
16	67986221	0	PTC299
17	92337	0	tarenflurbil
18	11561674	0	apremilast
19	5318517	0	andrographolide
20	27400	0	pizotifen

Pyqsar를 이용한 QSAR model 구축 순서

pyqsar package : https://github.com/crong-k/pyqsar_tutorial

1.Data load 및 불가용 descriptor 제거
2.Data Scaling
    - Min, Max 값을 맞춰준다.
3. Feature(descriptor)를 hierarchical clustering방법으로 묶어준다.
4. clustering 된 정보를 바탕으로 pyqsar의 feature selection을 진행한다.
    - Genetic Algorithm이용
5. Selected feature를 가지고 Multiple Linear Regression으로 모델은 만들어 결과 확인

Data load 및 불가용 descriptor 제거

Data Load

```
1  import pandas as pd
2  csv_file_name = "Narcotics_2D_Descriptor.csv"
3  sample_data = pd.read_csv(csv_file_name,sep=",")
4
5  X_data = sample_data.iloc[:,1:-1]
6  y_data = sample_data.iloc[:,-1:]
```

```
1  print(X_data.shape, y_data.shape)
```

```
((271, 1444), (271, 1))
```

총 271개의 화합물, descriptor는 1444개

Hello pyqsar!

```
import pyqsar
from pyqsar import data_tools as dt
```

Remove empty feature

```
X_data.shape
```

```
(271, 1444)
```

```
X_data = dt.rm_empty_feature(X_data)
```

Remove NaN & Infinity

```
X_data = dt.rmNaN(X_data)
```

```
for des in X_data:
    my_list = X_data[des].values
    for val in my_list:
        if val == "Infinity" or val == "-Infinity":
            #print des
            del X_data[des]
            break
```

```
X_data.shape
```

```
(271, 636)
```

Empty, NAN, infinity 값을 갖는 descriptor 제거
1444 --> 636

Normalize

sklearn의 MinMaxScaler를 이용해 값의 편향을 막기 위해 적절한 값으로 Normalization하는 것

Data scaling

```
1 from sklearn.preprocessing import MinMaxScaler
2 header = list(X_data.columns.values)
3 scaler = MinMaxScaler()
4 X_data_scaled = scaler.fit_transform(X_data)
5 X_data = pd.DataFrame(X_data_scaled, columns=header)
```

```
1 X_data.head()
```

	nAcid	apol	nAtom	nHeavyAtom	nH	nC	nN	nO	nS	nF	...	MW	AMW
0	0.0	0.169461	0.161850	0.149425	0.174419	0.235294	0.055556	0.09375	0.0	0.000000	...	0.135060	0.040260
1	0.0	0.154159	0.147399	0.137931	0.156977	0.215686	0.055556	0.06250	0.0	0.333333	...	0.125797	0.044860
2	0.0	0.125247	0.115607	0.103448	0.127907	0.156863	0.055556	0.03125	0.0	0.000000	...	0.109564	0.065358
3	0.0	0.160253	0.147399	0.143678	0.151163	0.235294	0.055556	0.06250	0.0	0.000000	...	0.127337	0.046816
4	0.0	0.160532	0.153179	0.132184	0.174419	0.225490	0.055556	0.03125	0.0	0.000000	...	0.118096	0.028948

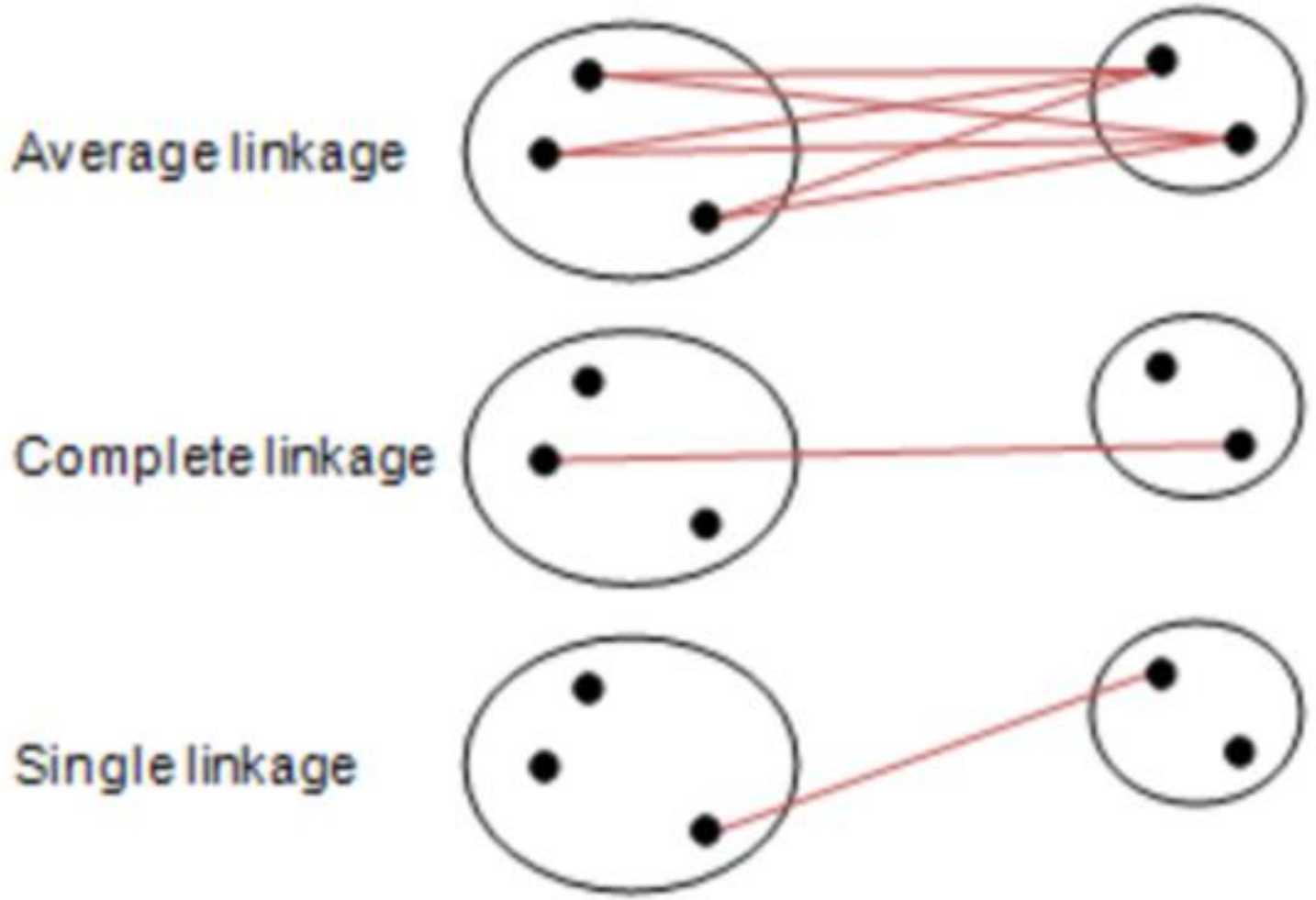
Feature clustering

Hierachical clustering의 기본 원리는 두 클러스터 사이의 거리를 측정해서 거리가 가까운 클러스터끼리 묶는 방식이다. 두 클러스터의 거리를 측정할 때 어디를 기준점으로 할것인가를 결정해야 한다.

Pyqsar에서는 hierachycal clustering 방법을 이용하며 아래의 결과와 같이 Average linkage방식이 원본 데이터와 가장 비슷하다는 것을 알 수 있다.

```
1 from pyqsar import clustering as cl
2 # calculate cophenetic correlation coefficient
3 cl.cophenetic(X_data)
```

```
average linkage cophenet: 0.8774754803064834
complete linkage cophenet: 0.839440512126338
single linkage cophenet: 0.3452863858533916 6
```

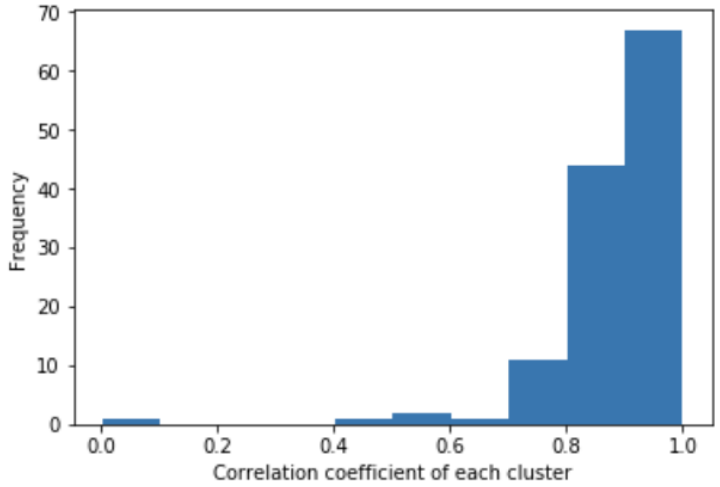


complete linkage : 두 클러스터상에서 가장 먼 거리를 이용해서 측정하는 방식
single linkage : 두 클러스터에서 가장 가까운 거리가 기준점
average linkage : 각 클러스터내의 각 점에서 다른 클러스터내의 모든 점 사이의 거리에 대한 평균을 사용하는 방식

```python
# clustering
clust = cl.FeatureCluster(X_data, 'average', 2)
clust_info = clust.set_cluster()
```

```
Cluster 225 ['MATS3c']
Cluster 226 ['AATSC3c']
Cluster 227 ['nF8Ring', 'nT8Ring']
Cluster 228 ['SCH-7', 'VCH-7']
Cluster 229 ['nT10HeteroRing', 'nF10HeteroRing']
Cluster 230 ['nF9HeteroRing', 'nT9Ring', 'nTRing', 'nT10Ring', 'nF9Ring', 'nFRing', 'nFG1
 'nF11HeteroRing', 'nT12HeteroRing', 'nT11Ring', 'nT8HeteroRing', 'nTG12Ring', 'nF12Ring'
eroRing', 'nTG12HeteroRing', 'nF8HeteroRing', 'nF11Ring', 'nT12Ring', 'nT9HeteroRing', 'n
Cluster 231 ['MDEC-23']
Cluster 232 ['MDEC-33']
Cluster 233 ['nT6Ring', 'n6Ring']
Cluster 234 ['SCH-6', 'VCH-6']
Cluster 235 ['nHeteroRing']
Cluster 236 ['ATSC2v', 'ATSC2p', 'ATSC2i']
Cluster 237 ['ATSC2e']
Cluster 238 ['CIC0', 'MLogP']
Cluster 239 ['CIC1']
Cluster 240 ['MDEC-12']
Cluster 241 ['MDEC-13']
Cluster 242 ['ATSC5i']
```

```python
clust.cluster_dist()
```



위의 결과를 바탕으로
average linkage방식으로 clustering 진행

Depth는 2로 진행

Feature selection

Pyqsar의 feature_selection_single/multi를 위의 cluster 정보를 이용하여 4개의 descriptor 만 추려냄.

```
from pyqsar import feature_selection_single as fss
single_set = fss.selection(X_data, y_data,
                           clust_info,
                           model='regression',
                           learning=5000,
                           bank=500,
                           component=4)
```

```
Start time :  20:19:02
Regression
1000 => 20:33:27 [0.7058177898469794, ['AATS3p', 'BCUTp-1h', 'GGI4', 'nF8HeteroRing']]
2000 => 20:44:55 [0.7058177898469794, ['AATS3p', 'BCUTp-1h', 'GGI4', 'nF8HeteroRing']]
3000 => 20:56:21 [0.7058177898469794, ['AATS3p', 'BCUTp-1h', 'GGI4', 'nF8HeteroRing']]
4000 => 21:07:47 [0.7058177898469794, ['AATS3p', 'BCUTp-1h', 'GGI4', 'nF8HeteroRing']]
```

Feature selection using multi core

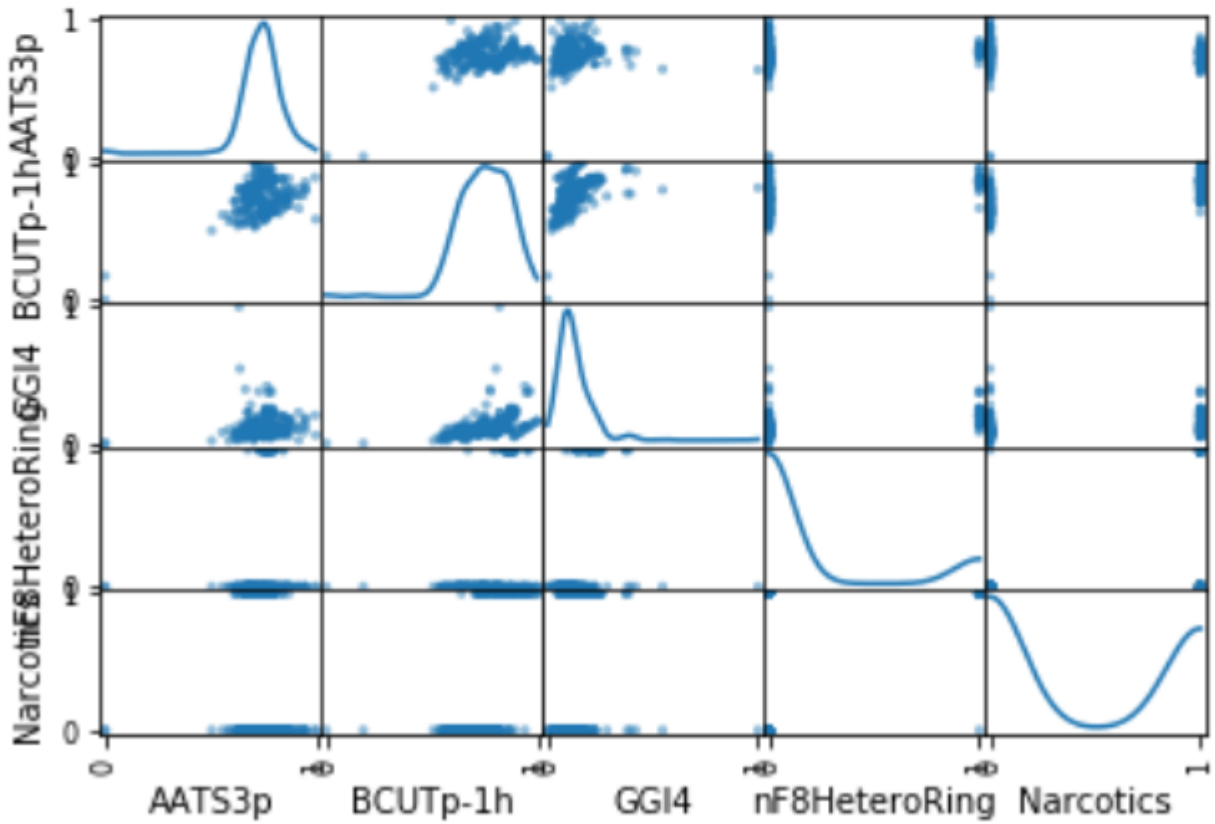
```
from pyqsar import feature_selection_multi as fsm
select_m = fsm.MultiSelection(X_data, y_data,
                              clust_info,
                              model='regression',
                              learning=10000,
                              bank=200,
                              component=4)

multi_set = select_m.run(n_core=4, run_job=3) #run_job : Number of times to perform the selection function
```

```
Regression
[0.7058177898469794, ['AATS3p', 'BCUTp-1h', 'GGI4', 'nF8HeteroRing']]
[0.7058177898469794, ['AATS3p', 'BCUTp-1h', 'GGI4', 'nF8HeteroRing']]
[0.7058177898469794, ['AATS3p', 'BCUTp-1h', 'GGI4', 'nF8HeteroRing']]
```

두 결과 모두
R^2 = 0.705817
Features =
['AATS3p', 'BCUTp-1h', 'GGI4', 'nF8HeteroRing']
로 같았다.

Out[23]:

	AATS3p	BCUTp-1h	GGI4	nF8HeteroRing	Narcotics
AATS3p	1.000000	0.441689	0.129748	0.116450	-0.113335
BCUTp-1h	0.441689	1.000000	0.461321	0.398557	0.630488
GGI4	0.129748	0.461321	1.000000	0.389790	0.146288
nF8HeteroRing	0.116450	0.398557	0.389790	1.000000	0.477845
Narcotics	-0.113335	0.630488	0.146288	0.477845	1.000000



Multilinear Regression modeling & Cross validation

위에서 추려낸 4개의 descriptor로 식을 만들기 위하여
pyqsar의 export_model을 이용하여 multilinear regression 모델을 만들었다.

```
from pyqsar import export_model as em

#mymodel_s = em.ModelExport(X_data,y_data, single_set)
mymodel_m = em.ModelExport(X_data,y_data, multi_set)
```

```
#mymodel_s.mlr()
mymodel_m.mlr()
```

```
Model features:  ['AATS3p', 'BCUTp-1h', 'GGI4', 'nF8HeteroRing']
Coefficients:  [[-2.44553576  3.49525715 -1.71977663  0.42114135]]
Intercept:  [-0.24103206]
RMSE: 0.268653
R^2: 0.705818
```

식으로 만들면

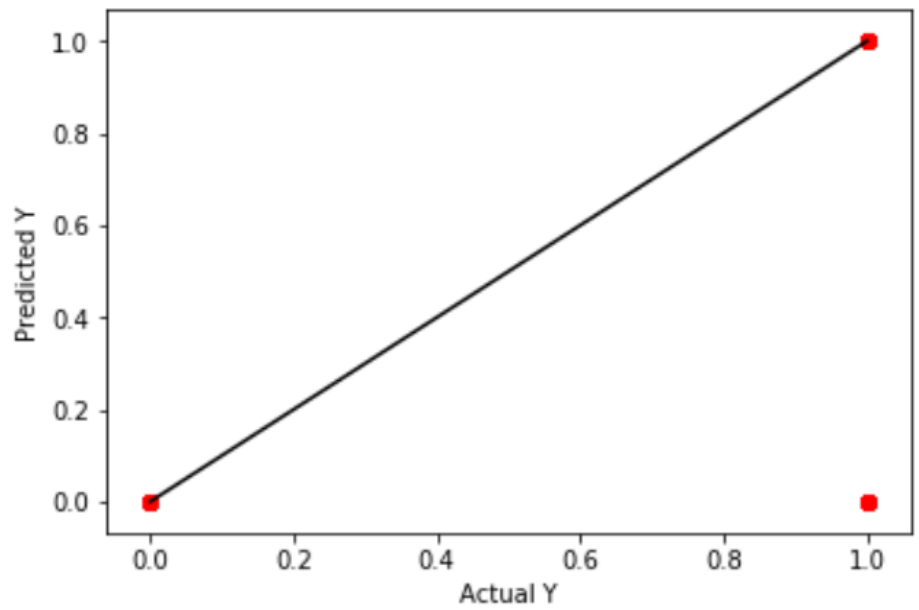
$$y = -2.44553576x1 + 3.49525715x2 - 1.71977663x3 + 0.42114135x4 - 0.24103206$$

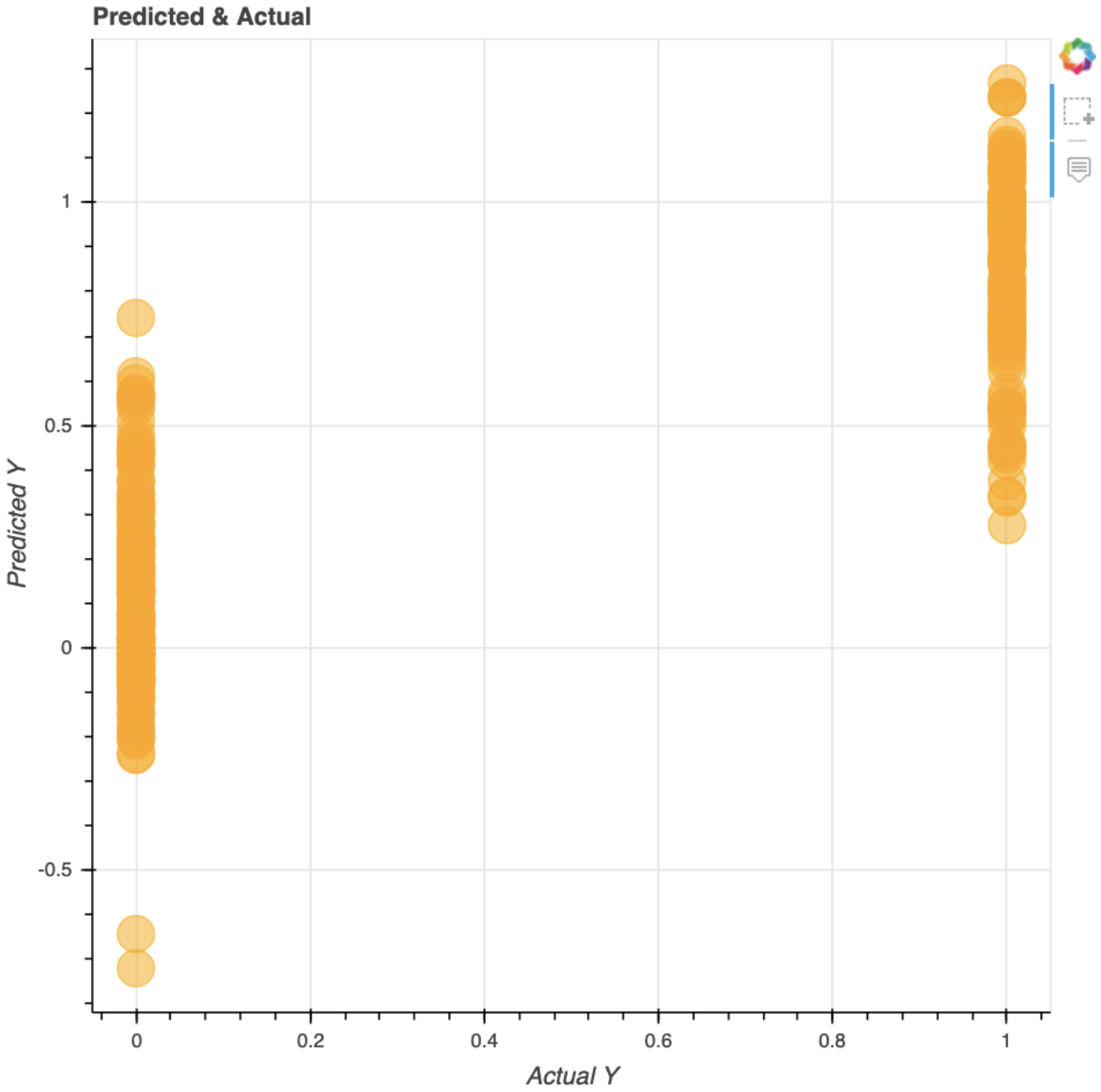
Cross validation

In [27]:
```
from pyqsar import cross_validation as cv
#cv.k_fold(X_data, y_data, single_set, k=5, run=100)
cv.k_fold(X_data, y_data, multi_set, k=5, run=100)
```

```
/share/anaconda2/lib/python2.7/site-packages/pyqsar/cross_validation.py:35: FutureWarning: Method .as_matrix w
ill be removed in a future version. Use .values instead.
  x = X_data.loc[:,feature_set].as_matrix()
/share/anaconda2/lib/python2.7/site-packages/pyqsar/cross_validation.py:36: FutureWarning: Method .as_matrix w
ill be removed in a future version. Use .values instead.
  y =y_data.as_matrix()
```

```
R^2CV mean: 0.706575
Q^2CV mean: 0.688087
RMSE CV : 0.569845
Features set = ['AATS3p', 'BCUTp-1h', 'GGI4', 'nF8HeteroRing']
Model coeff =  [[-2.35609051  3.37099251 -1.68142851  0.44762477]]
Model intercept =  [-0.22475924]
```





모델로 예측한 값과 실제값을 비교하는 plot

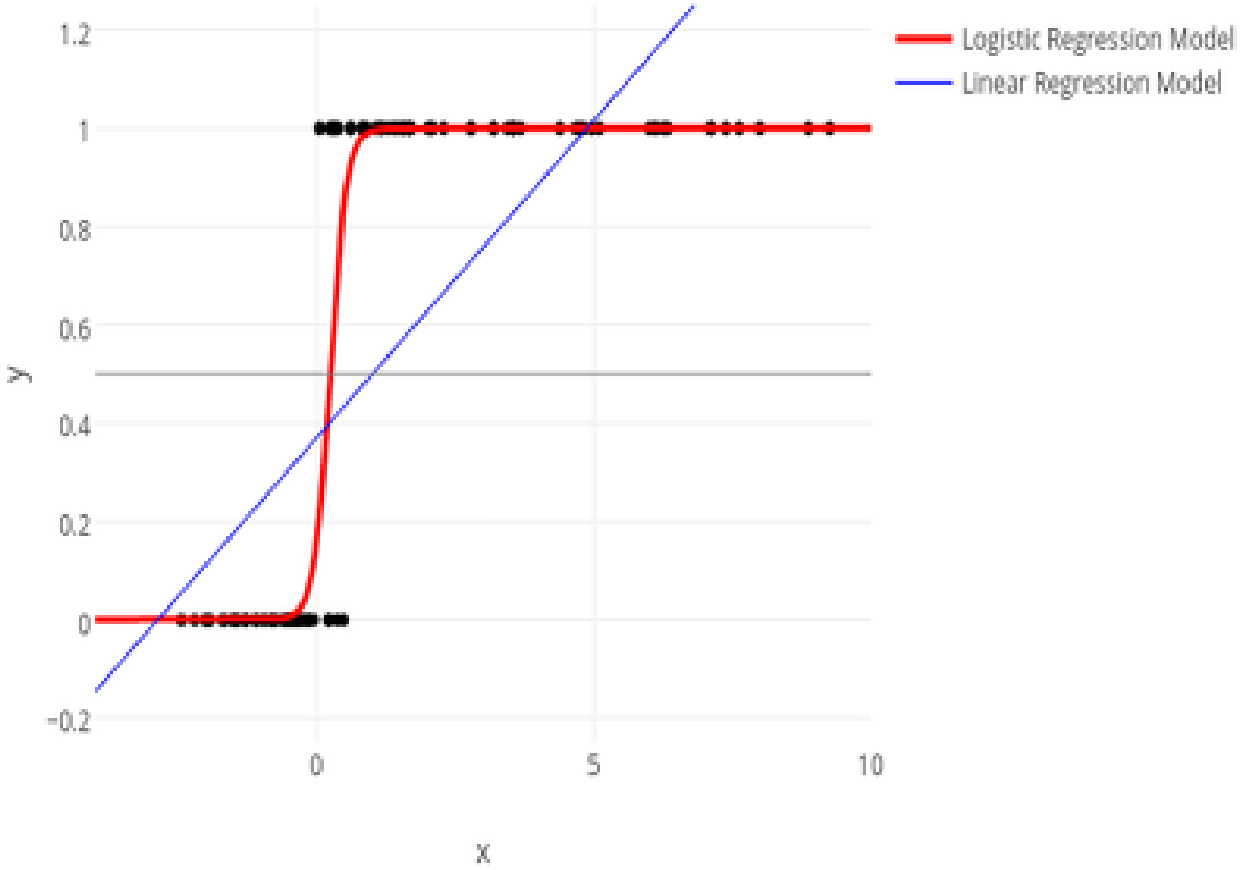
실제 Narcotic 화합물은 1 근처에, Nonnarcotic 화합물은 0근처에 있는 것이 보이지만 Spectrum이 너무 넓고 1을 넘어가는 값과, 0 아래로 내려가는 값도 보임

Linear regression으로 classification을 하는 것에 한계를 느낌

→ Logistic regression으로 시도

Logistic regression

logistic regression 은 분류(classification) 문제를 위한 회귀 방법



$$H(x) = Wx + b$$

$$H(x) = z$$

$$0 < g(z) < 1$$

$$g(z) = \frac{1}{(1 + e^{-z})}$$

$$g(z) = \frac{1}{(1 + e^{-(Wx+b)})}$$

선형식을 $g(z) = \frac{1}{(1 + e^{-z})}$ 에 대입해 0과 1 의 값을 갖도록

Data split

Logistic regression 검증을 위해 train data와 test data로 나누었다.
Sklearn 안의 train_test_split 함수를 이용.
총 271개의 데이터를 Train set : test set = 80 : 20 비율로 나눔

Data split

```
[ ]
1 from sklearn.model_selection import train_test_split
2 X_train, X_test, y_train, y_test = train_test_split(X_data, y_data, test_size=0.2)
3 print(X_train.shape, X_test.shape)
```

```
((216, 636), (55, 636))
```

feature selection

Test data는 model을 만들 때 영향을 주어서는 안되기 때문에
Train data만을 가지고 다시 feature selection을 진행하였다.

Feature selection using multi core

```
from pyqsar import feature_selection_multi as fsm
select_m = fsm.MultiSelection(X_train, y_train,
                              clust_info,
                              model='regression',
                              learning=10000,
                              bank=200,
                              component=4)

multi_set = select_m.run(n_core=4, run_job=3) #run_job : Number of times to perform the
```

```
Regression
[0.7402321376569486, ['BCUTp-1h', 'R_TpiPCTPC', 'nBase', 'nHBDon_Lipinski']]
[0.7402321376569486, ['BCUTp-1h', 'R_TpiPCTPC', 'nBase', 'nHBDon_Lipinski']]
[0.7402321376569486, ['BCUTp-1h', 'R_TpiPCTPC', 'nBase', 'nHBDon_Lipinski']]
```

Logistic regression 모델 만들기

위의 feature를 가지고 logistic regression 모델 만들기

Logistic regression

```
1  from sklearn.linear_model import LogisticRegression
2  from sklearn.metrics import mean_squared_error , r2_score
3  import numpy as np
4  from matplotlib import pyplot as plt
5  X_train_fs = X_train.loc[:, multi_set]
6  X_test_fs = X_test.loc[:, multi_set]
7
8
```

```
1  LR_model = LogisticRegression()
2  LR_model.fit(X_train_fs, y_train)
3  logistic_coef = pd.DataFrame(LR_model.coef_,columns = multi_set)
4  logistic_coef
```

```
/share/anaconda2/lib/python2.7/site-packages/sklearn/linear_model/logistic.py:433:
be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
/share/anaconda2/lib/python2.7/site-packages/sklearn/utils/validation.py:761: Data
y was passed when a 1d array was expected. Please change the shape of y to (n_samp
  y = column_or_1d(y, warn=True)
```

	BCUTp-1h	R_TpiPCTPC	nBase	nHBDon_Lipinski
0	3.732551	-4.318077	1.265596	-1.173896

결과

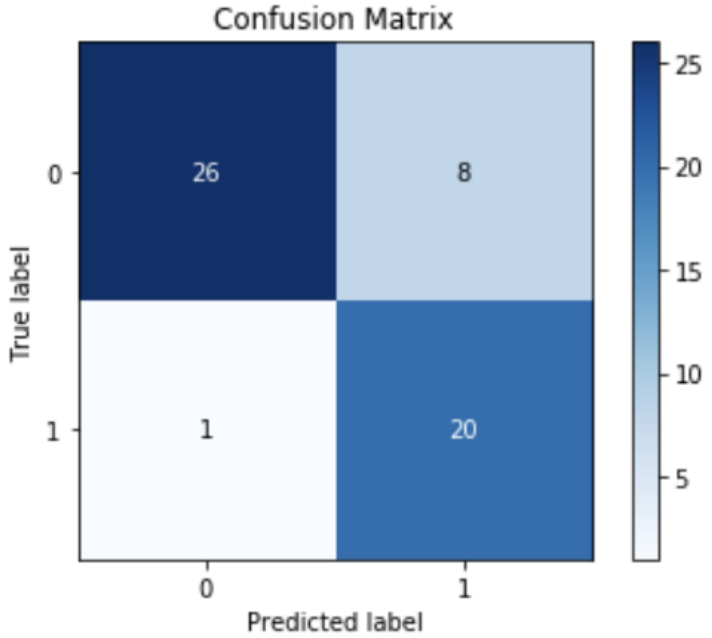
```
predict_LR = LR_model.predict(X_test_fs)
```

```
from sklearn import metrics
print('Accuracy: %.2f' % metrics.accuracy_score(y_test,
                                                predict_LR))
```

```
Accuracy: 0.84
```

```
import scikitplot as skplt
skplt.metrics.plot_confusion_matrix(y_test, predict_LR)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4adcf67290>
```



해당 모델은 test data에서 0.84의 정확도를 갖는 것으로 나타남

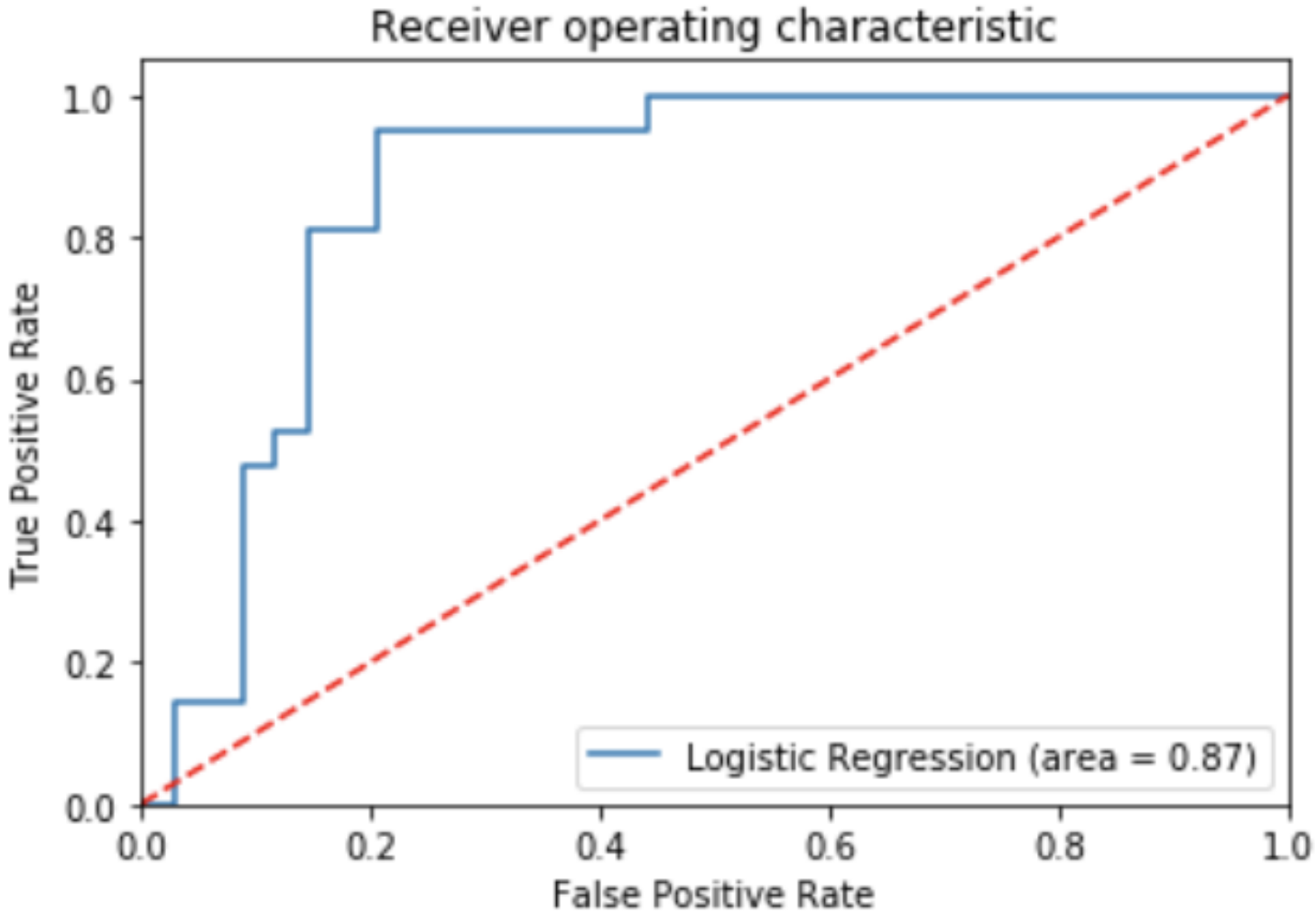
총 55개의 test data 중
실제 마약 34개 중 26개를 마약일 것이라고 예측하였고

실제 마약이 아닌 21개 중 20개를 마약이 아니라고 예측하였다.

ROC curve

```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
import matplotlib.pyplot as plt

#logit_roc_auc = roc_auc_score(y_test, predict_LR)
pred_p = LR_model.predict_proba(X_test_fs)[:,1]
fpr, tpr, thresholds = roc_curve(y_test, pred_p)
plt.figure()
logic_roc_auc = metrics.auc(fpr , tpr)
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logic_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```

ROC curve



ROC curve의 곡선이 빨간선과 멀수록(면적이 1에 가까울수록) 좋은 결과

24	34328	1	Difenoxin	opioid	
25	13505	1	Diphenoxylate	opioid	
26	13331	1	Dipipanone	opioid	
27	5359421	1	Dihydromorphine	opioid	
28	107765	1	Dihydroetorphine	opioid	
29	5284543	1	Dihydrocodeine	opioid	
30	5360696	1	Racemethorphan	NMDA	
31	9648	1	Racemoramide	opioid	
32	60815	1	Remifentanil	opioid	
33	5359272	1	Levorphanol	opioid	
34	5362449	1	Levomethorphan	opioid	
35	10453145	1	Levomoramide	opioid	
36	5362482	1	Levophenacylmorphan		opioid
37	4095	1	Methadone	opioid	
38	31331	1	MethadoneIntermediate		
39	5362518	1	Methyldesorphine	opioid	
40	5464303	1	Methyldihydromorphine		opioid
41	62518	1	Metazocine	opioid	
42	5359353	1	Metopon	opioid	
43	567581	1	MoramideIntermediate		opioid

우리가 가지고 있는 마약(Narcotic)데이터가 거의 대부분 opioid계열 마약인 것으로 보여짐
Opioid가 아닌 다른 receptor 계열의 마약이 들어오면 모델 적용이 어려울 수 있음.