Simple ML Model

In [ ]:
```
import pandas as pd
import numpy as np
```

```
data=pd.read_csv('crx.data',header=None)
```

In [26]:
```
#variable engineering or feature engineering
varname=['A'+str(s) for s in range(1,17)]
```

In [27]:
```
data.columns=varname
```

In [28]:
```
data.head(20)
```

Out[28]:

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16
0	b	30.83	0.000	u	g	w	v	1.250	t	t	1	f	g	00202	0	+
1	a	58.67	4.460	u	g	q	h	3.040	t	t	6	f	g	00043	560	+
2	a	24.50	0.500	u	g	q	h	1.500	t	f	0	f	g	00280	824	+
3	b	27.83	1.540	u	g	w	v	3.750	t	t	5	t	g	00100	3	+
4	b	20.17	5.625	u	g	w	v	1.710	t	f	0	f	s	00120	0	+
5	b	32.08	4.000	u	g	m	v	2.500	t	f	0	t	g	00360	0	+
6	b	33.17	1.040	u	g	r	h	6.500	t	f	0	t	g	00164	31285	+
7	a	22.92	11.585	u	g	cc	v	0.040	t	f	0	f	g	00080	1349	+
8	b	54.42	0.500	y	p	k	h	3.960	t	f	0	f	g	00180	314	+
9	b	42.50	4.915	y	p	w	v	3.165	t	f	0	t	g	00052	1442	+
10	b	22.08	0.830	u	g	c	h	2.165	f	f	0	t	g	00128	0	+
11	b	29.92	1.835	u	g	c	h	4.335	t	f	0	f	g	00260	200	+
12	a	38.25	6.000	u	g	k	v	1.000	t	f	0	t	g	00000	0	+
13	b	48.08	6.040	u	g	k	v	0.040	f	f	0	f	g	00000	2690	+
14	a	45.83	10.500	u	g	q	v	5.000	t	t	7	t	g	00000	0	+
15	b	36.67	4.415	y	p	k	v	0.250	t	t	10	t	g	00320	0	+
16	b	28.25	0.875	u	g	m	v	0.960	t	t	3	t	g	00396	0	+
17	a	23.25	5.875	u	g	q	v	3.170	t	t	10	f	g	00120	245	+
18	b	21.83	0.250	u	g	d	h	0.665	t	f	0	t	g	00000	0	+
19	a	19.17	8.585	u	g	cc	h	0.750	t	t	7	f	g	00096	0	+

In [29]:
```
data=data.replace('?',np.nan)
```

In [30]:
```
1 data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 690 entries, 0 to 689
Data columns (total 16 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   A1      678 non-null    object 
 1   A2      678 non-null    object 
 2   A3      690 non-null    float64
 3   A4      684 non-null    object 
 4   A5      684 non-null    object 
 5   A6      681 non-null    object 
 6   A7      681 non-null    object 
 7   A8      690 non-null    float64
 8   A9      690 non-null    object 
 9   A10     690 non-null    object 
 10  A11     690 non-null    int64  
 11  A12     690 non-null    object 
 12  A13     690 non-null    object 
 13  A14     677 non-null    object 
 14  A15     690 non-null    int64  
 15  A16     690 non-null    object 
dtypes: float64(2), int64(2), object(12)
memory usage: 86.4+ KB
```

In [31]:
```
1 #recast their variables to correct types
2 data['A2']=data['A2'].astype('float')
3 data['A14']=data['A14'].astype('float')
```

In [32]:
```
1 #encode A16 from + - to 1 and 0
2 data['A16']=data['A16'].map({'+':1,'-':0})
```

In [33]:
```
1 data['A16'].head(20)
```

Out[33]:
```
0     1
1     1
2     1
3     1
4     1
5     1
6     1
7     1
8     1
9     1
10    1
11    1
12    1
13    1
14    1
15    1
16    1
17    1
18    1
19    1
Name: A16, dtype: int64
```

In [34]:
```
1 #numerical variables
2 #discrete 3,4
3 #continuos 1.5,3.5
4 #categorical variables
5 #nominal (NAMES)-R,G,B,OOTY
6 #ORDINAL (ORDERING)-SUNDAY-SATURDAY
```

In [35]:
```
1 cat_columns=[c for c in data.columns if data[c].dtypes=='O']
```

In [36]:
```
1 data[cat_columns].head(20)
```

Out[36]:

	A1	A4	A5	A6	A7	A9	A10	A12	A13
0	b	u	g	w	v	t	t	f	g
1	a	u	g	q	h	t	t	f	g
2	a	u	g	q	h	t	f	f	g
3	b	u	g	w	v	t	t	t	g
4	b	u	g	w	v	t	f	f	s
5	b	u	g	m	v	t	f	t	g
6	b	u	g	r	h	t	f	t	g
7	a	u	g	cc	v	t	f	f	g
8	b	y	p	k	h	t	f	f	g
9	b	y	p	w	v	t	f	t	g
10	b	u	g	c	h	f	f	t	g
11	b	u	g	c	h	t	f	f	g
12	a	u	g	k	v	t	f	t	g
13	b	u	g	k	v	f	f	f	g
14	a	u	g	q	v	t	t	t	g
15	b	y	p	k	v	t	t	t	g
16	b	u	g	m	v	t	t	t	g
17	a	u	g	q	v	t	t	f	g
18	b	u	g	d	h	t	f	t	g
19	a	u	g	cc	h	t	t	f	g

In [37]:
```
1 num_columns=[c for c in data.columns if data[c].dtypes!='O']
```

In [38]:
```
1 data[num_columns].head(20)
```

Out[38]:

	A2	A3	A8	A11	A14	A15	A16
0	30.83	0.000	1.250	1	202.0	0	1
1	58.67	4.460	3.040	6	43.0	560	1
2	24.50	0.500	1.500	0	280.0	824	1
3	27.83	1.540	3.750	5	100.0	3	1
4	20.17	5.625	1.710	0	120.0	0	1
5	32.08	4.000	2.500	0	360.0	0	1
6	33.17	1.040	6.500	0	164.0	31285	1
7	22.92	11.585	0.040	0	80.0	1349	1
8	54.42	0.500	3.960	0	180.0	314	1
9	42.50	4.915	3.165	0	52.0	1442	1
10	22.08	0.830	2.165	0	128.0	0	1
11	29.92	1.835	4.335	0	260.0	200	1
12	38.25	6.000	1.000	0	0.0	0	1
13	48.08	6.040	0.040	0	0.0	2690	1
14	45.83	10.500	5.000	7	0.0	0	1
15	36.67	4.415	0.250	10	320.0	0	1
16	28.25	0.875	0.960	3	396.0	0	1
17	23.25	5.875	3.170	10	120.0	245	1
18	21.83	0.250	0.665	0	0.0	0	1
19	19.17	8.585	0.750	7	96.0	0	1

In [44]:
```
1 data1=pd.read_csv('loan.csv')
```

In [45]:
```
1 data1
```

Out[45]:

	customer_id	disbursed_amount	interest	market	employment	time_employed	householder
0	0	23201.5	15.4840	C	Teacher	<=5 years	RENT
1	1	7425.0	11.2032	B	Accountant	<=5 years	OWNER
2	2	11150.0	8.5100	A	Statistician	<=5 years	RENT
3	3	7600.0	5.8656	A	Other	<=5 years	RENT
4	4	31960.0	18.7392	E	Bus driver	>5 years	RENT
...	...	...	...	...	...	...	...
9995	9995	23750.0	11.0019	B	Civil Servant	<=5 years	MORTGAGE
9996	9996	11880.0	10.4923	B	Civil Servant	<=5 years	RENT
9997	9997	19950.0	8.4364	B	Accountant	>5 years	OWNER
9998	9998	4850.0	13.6409	C	Bus driver	>5 years	RENT
9999	9999	25584.0	17.1456	D	Accountant	>5 years	MORTGAGE

10000 rows × 14 columns

In [46]:
```
1 #continuous
2 data1['disbursed_amount'].unique()
```

Out[46]:
```
array([23201.5 ,  7425.  , 11150.  , ...,  6279.  , 12894.75, 25584.  ])
```

In [47]:
```
1 #discrete
2 data1['number_open_accounts'].unique()
```

Out[47]:
```
array([ 4., 13.,  8., 20., 14.,  5.,  9., 18., 16., 17., 12., 15.,  6.,
       10., 11.,  7., 21., 19., 26.,  2., 22., 27., 23., 25., 24., 28.,
        3., 30., 41., 32., 33., 31., 29., 37., 49., 34., 35., 38.,  1.,
       36., 42., 47., 40., 44., 43.])
```

In [49]:
```
1 data1['target'].unique()
```

Out[49]:
```
array([0, 1], dtype=int64)
```

In [50]:
```
1 data1['time_employed'].unique()
```

Out[50]:
```
array(['<=5 years', '>5 years', nan], dtype=object)
```

In [51]:
```
1 data1['householder'].value_counts()
```

Out[51]:
```
MORTGAGE    4957
RENT        4055
OWNER        988
Name: householder, dtype: int64
```

In [53]:
```
1 data1[['date_issued','date_last_payment']].dtypes
```

Out[53]:
```
date_issued          object
date_last_payment    object
dtype: object
```

In [54]:
```
1 data1['dateissueddt']=pd.to_datetime(data1['date_issued'])
```

In [55]:
```
1 data1.head()
```

Out[55]:

	customer_id	disbursed_amount	interest	market	employment	time_employed	householder	i
0	0	23201.5	15.4840	C	Teacher	<=5 years	RENT	8
1	1	7425.0	11.2032	B	Accountant	<=5 years	OWNER	10
2	2	11150.0	8.5100	A	Statistician	<=5 years	RENT	6
3	3	7600.0	5.8656	A	Other	<=5 years	RENT	10
4	4	31960.0	18.7392	E	Bus driver	>5 years	RENT	9

In [57]:
```
1 data1['month']=data1['dateissueddt'].dt.month
2 data1['month'].head(10)
```

Out[57]:
```
0     6
1     5
2    10
3     8
4     7
5     8
6     9
7     3
8     2
9    12
Name: month, dtype: int64
```

In [63]:
```
1 data1['day']=data1['dateissueddt'].dt.day
2 data1['year']=data1['dateissueddt'].dt.year
```

In [64]:
```
1
2 data1['year'].head(10)
```

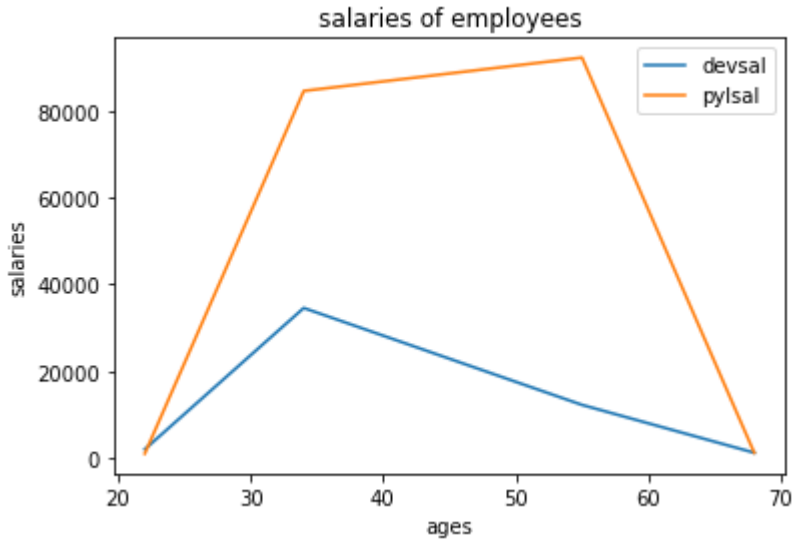
Out[64]:
```
0    2013
1    2014
2    2013
3    2015
4    2014
5    2013
6    2015
7    2015
8    2014
9    2013
Name: year, dtype: int64
```

In [65]:
```
1 data1['day'].head(10)
```

Out[65]:
```
0    11
1     8
2    26
3    20
4    22
5    21
6    27
7    20
8    14
9    25
Name: day, dtype: int64
```

In [71]:
```
#matplot
from matplotlib import pyplot as plt
plt.title('salaries of employees')
plt.xlabel('ages')
plt.ylabel('salaries')
dev_ages=[22,34,55,68]
dev_sal=[2000,34555,12234,1212]
py_sal=[1000,84555,92234,1212]
plt.plot(dev_ages,dev_sal,label='devsal')
plt.plot(dev_ages,py_sal,label='pylsal')
plt.legend()
```

Out[71]: <matplotlib.legend.Legend at 0x189b46fda60>



In [1]:
```
import sklearn.datasets
```

In [2]:
```
can=sklearn.datasets.load_breast_cancer()
```

In [3]:
```
X=can.data
y=can.target
```

In [4]:
```
print(X.shape,y.shape)
```

```
(569, 30) (569,)
```

In [5]:
```
import pandas as pd
data=pd.DataFrame(can.data,columns=can.feature_names)
```

In [6]:
```
data["class"]=can.target
```

In [7]:
```
1 data.head()
2 data.tail(5)
```

Out[7]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587

5 rows × 31 columns

In [8]:
```
1 data.describe()
```

Out[8]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	56
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	

8 rows × 31 columns

In [9]:
```
1 data["class"].value_counts()
```

Out[9]:
```
1    357
0    212
Name: class, dtype: int64
```

In [10]:
```
1 can.target_names
```

Out[10]:
```
array(['malignant', 'benign'], dtype='<U9')
```

In [11]:
```
1 data.groupby('class').mean()
```

Out[11]:

class	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	cor p
0	17.462830	21.604906	115.365377	978.376415	0.102898	0.145188	0.160775	0.08
1	12.146524	17.914762	78.075406	462.790196	0.092478	0.080085	0.046058	0.02

2 rows × 30 columns

In [12]:
```
1 data.groupby('mean radius').mean()
```

Out[12]:

mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	dim
6.981	13.43	43.79	143.5	0.11700	0.07568	0.00000	0.000000	0.1930	0
7.691	25.44	48.34	170.4	0.08668	0.11990	0.09252	0.013640	0.2037	0
7.729	25.49	47.98	178.8	0.08098	0.04878	0.00000	0.000000	0.1870	0
7.760	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.000000	0.1587	0
8.196	16.84	51.71	201.9	0.08600	0.05943	0.01588	0.005917	0.1769	0
...	...	...	...	...	...	...	...	...	
25.220	24.91	171.50	1878.0	0.10630	0.26650	0.33390	0.184500	0.1829	0
25.730	17.46	174.20	2010.0	0.11490	0.23630	0.33680	0.191300	0.1956	0
27.220	21.87	182.10	2250.0	0.10940	0.19140	0.28710	0.187800	0.1800	0
27.420	26.27	186.90	2501.0	0.10840	0.19880	0.36350	0.168900	0.2061	0
28.110	18.47	188.50	2499.0	0.11420	0.15160	0.32010	0.159500	0.1648	0

456 rows × 30 columns

In [13]:
```
1 from sklearn.model_selection import train_test_split
2 X1=data.drop('class',axis=1)
3 Y1=data['class']
4 x_train,x_test,y_train,y_test=train_test_split(X1,Y1,test_size=0.4,stratify=
5 print(Y1.shape,y_train.shape,y_test.shape)
```

```
(569,) (341,) (228,)
```

In [14]:
```
print(Y1.mean(),y_train.mean(),y_test.mean())
```

```
0.6274165202108963 0.6275659824046921 0.6271929824561403
```

In [15]:
```
x_train,x_test,y_train,y_test=train_test_split(X1,Y1,test_size=0.4)#without
print(Y1.shape,y_train.shape,y_test.shape)
```

```
(569,) (341,) (228,)
```

In [16]:
```
print(Y1.mean(),y_train.mean(),y_test.mean())
```

```
0.6274165202108963 0.6187683284457478 0.6403508771929824
```

In [17]:
```
x_train,x_test,y_train,y_test=train_test_split(X1,Y1,test_size=0.4,stratify=
print(Y1.shape,y_train.shape,y_test.shape)
```

```
(569,) (341,) (228,)
```

In [18]:
```
print(Y1.mean(),y_train.mean(),y_test.mean())
```

```
0.6274165202108963 0.6275659824046921 0.6271929824561403
```

In [19]:
```
1 print(X1.mean(),x_train.mean(),x_test.mean())
```

```
mean radius                14.127292
mean texture               19.289649
mean perimeter             91.969033
mean area                 654.889104
mean smoothness             0.096360
mean compactness            0.104341
mean concavity              0.088799
mean concave points         0.048919
mean symmetry               0.181162
mean fractal dimension      0.062798
radius error                0.405172
texture error               1.216853
perimeter error             2.866059
area error                 40.337079
smoothness error            0.007041
compactness error           0.025478
concavity error             0.031894
concave points error        0.011796
symmetry error              0.020542
fractal dimension error     0.003795
worst radius               16.269190
worst texture              25.677223
worst perimeter           107.261213
worst area                880.583128
worst smoothness            0.132369
worst compactness           0.254265
worst concavity             0.272188
worst concave points        0.114606
worst symmetry              0.290076
worst fractal dimension     0.083946
dtype: float64 mean radius                14.167018
mean texture               19.267273
mean perimeter             92.194839
mean area                 660.770968
mean smoothness             0.096472
mean compactness            0.103606
mean concavity              0.088231
mean concave points         0.049226
mean symmetry               0.180840
mean fractal dimension      0.062770
radius error                0.414993
texture error               1.227954
perimeter error             2.914932
area error                 42.020155
smoothness error            0.007114
compactness error           0.025655
concavity error             0.031797
concave points error        0.011905
symmetry error              0.020599
fractal dimension error     0.003838
worst radius               16.302003
worst texture              25.631584
worst perimeter           107.368065
worst area                888.655425
```

```
worst smoothness            0.131733
worst compactness           0.250276
worst concavity             0.266565
worst concave points        0.113628
worst symmetry              0.288151
worst fractal dimension     0.083792
dtype: float64 mean radius                14.067877
mean texture               19.323114
mean perimeter             91.631316
mean area                 646.092105
mean smoothness             0.096193
mean compactness            0.105440
mean concavity              0.089649
mean concave points         0.048461
mean symmetry               0.181643
mean fractal dimension      0.062839
radius error                0.390484
texture error               1.200252
perimeter error             2.792965
area error                 37.819846
smoothness error            0.006931
compactness error           0.025214
concavity error             0.032039
concave points error        0.011634
symmetry error              0.020458
fractal dimension error     0.003730
worst radius               16.220114
worst texture              25.745482
worst perimeter           107.101404
worst area                868.510088
worst smoothness            0.133320
worst compactness           0.260231
worst concavity             0.280599
worst concave points        0.116069
worst symmetry              0.292954
worst fractal dimension     0.084175
dtype: float64
```

In [44]:
```
x_binarised_train=x_train.apply(pd.cut,bins=2,labels=[1,0])
x_binarised_test=x_test.apply(pd.cut,bins=2,labels=[1,0])
```

In [45]:
```
type(x_binarised_train)
x_binarised_train=x_binarised_train.values
x_binarised_test=x_binarised_test.values
type(x_binarised_train)
```

Out[45]:
```
numpy.ndarray
```

In [42]:

```
import numpy as np
from random import randint

b=10
for b in range(x_binarised_train.shape[1]+1):
    accurate_rows=0
    for x,y in zip(x_binarised_train,y_train):
        y_pred=(np.sum(x)>=b)
        accurate_rows+=(y==y_pred)
    print('b=',b,accurate_rows, accurate_rows/x_binarised_train.shape[0])

'''
i=randint(0,x_binarised_train.shape[0])
b=10

if(np.sum(x_binarised_train[i:])>=b):
    print('model prediction is malignant')
else:
    print('model prediction is benign')

if(y_train[i]==1):
    print('Actual outcome is malignant')
else:
    print('Actual outcome is benign')'''
```

```
b= 0 214 0.6275659824046921
b= 1 214 0.6275659824046921
b= 2 214 0.6275659824046921
b= 3 214 0.6275659824046921
b= 4 214 0.6275659824046921
b= 5 214 0.6275659824046921
b= 6 214 0.6275659824046921
b= 7 214 0.6275659824046921
b= 8 214 0.6275659824046921
b= 9 214 0.6275659824046921
b= 10 214 0.6275659824046921
b= 11 214 0.6275659824046921
b= 12 214 0.6275659824046921
b= 13 216 0.6334310850439883
b= 14 218 0.6392961876832844
b= 15 219 0.6422287390029325
b= 16 223 0.6539589442815249
b= 17 226 0.6627565982404692
b= 18 227 0.6656891495601173
b= 19 229 0.6715542521994134
b= 20 238 0.6979472140762464
b= 21 243 0.7126099706744868
b= 22 252 0.7390029325513197
b= 23 264 0.7741935483870968
b= 24 270 0.7917888563049853
b= 25 278 0.8152492668621701
b= 26 289 0.8475073313782991
```

```
b= 27 288 0.844574780058651
b= 28 284 0.8328445747800587
b= 29 272 0.7976539589442815
b= 30 239 0.7008797653958945
```

Out[42]:
```
"\ni=randint(0,x_binarised_train.shape[0])\nb=10\n\nif(np.sum(x_binarised_train
[i:])>=b):\n    print('model prediction is malignant')\nelse:\n    print('model
prediction is benign')\n    \nif(y_train[i]==1):\n    print('Actual outcome is
malignant')\nelse:\n    print('Actual outcome is benign')"
```

In [47]:
```
b=27
accurate_rows=0
for x,y in zip(x_binarised_train,y_train):
    y_pred=(np.sum(x)>=b)
    accurate_rows+=(y==y_pred)
    print('b=',b,accurate_rows, accurate_rows/x_binarised_train.shape[0])
```

```
b= 27 269 0.7888563049853372
b= 27 270 0.7917888563049853
b= 27 271 0.7947214076246334
b= 27 271 0.7947214076246334
b= 27 272 0.7976539589442815
b= 27 272 0.7976539589442815
b= 27 273 0.8005865102639296
b= 27 273 0.8005865102639296
b= 27 274 0.8035190615835777
b= 27 275 0.8064516129032258

b= 27 276 0.8093841642228738
b= 27 277 0.8123167155425219
b= 27 278 0.8152492668621701
b= 27 279 0.8181818181818182
b= 27 280 0.8211143695014663
b= 27 281 0.8240469208211144
b= 27 282 0.8269794721407625
b= 27 283 0.8299120234604106
b= 27 284 0.8328445747800587
```

In [41]:
```
for b in range(x_binarised_test.shape[1]+1):
    accurate_rows=0
    for x,y in zip(x_binarised_test,y_test):
        y_pred=(np.sum(x)>=b)
        accurate_rows+=(y==y_pred)
    print('b=',b,accurate_rows, accurate_rows/x_binarised_train.shape[0])
```

```
b= 0 143 0.41935483870967744
b= 1 143 0.41935483870967744
b= 2 143 0.41935483870967744
b= 3 143 0.41935483870967744
b= 4 143 0.41935483870967744
b= 5 143 0.41935483870967744
b= 6 143 0.41935483870967744
b= 7 143 0.41935483870967744
b= 8 143 0.41935483870967744
b= 9 144 0.4222873900293255
b= 10 144 0.4222873900293255
b= 11 146 0.4281524926686217
b= 12 147 0.4310850439882698
b= 13 149 0.436950146627566
b= 14 149 0.436950146627566
b= 15 150 0.4398826979472141
b= 16 150 0.4398826979472141
b= 17 155 0.45454545454545453
b= 18 161 0.47214076246334313
b= 19 162 0.4750733137829912
b= 20 166 0.4868035190615836
b= 21 172 0.5043988269794721
b= 22 181 0.530791788856305
b= 23 187 0.5483870967741935
b= 24 192 0.5630498533724341
b= 25 197 0.5777126099706745
b= 26 202 0.592375366568915
b= 27 201 0.5894428152492669
b= 28 197 0.5777126099706745
b= 29 178 0.5219941348973607
b= 30 151 0.44281524926686217
```

In [56]:
```
b=28
accurate_rows=0
for x,y in zip(x_binarised_test,y_test):
    y_pred=(np.sum(x)>=b)
    accurate_rows+=(y==y_pred)
    print('b=',b,accurate_rows, accurate_rows/x_binarised_test.shape[0])
```

```
b= 28 175 0.7675438596491229
b= 28 176 0.7719298245614035
b= 28 176 0.7719298245614035
b= 28 177 0.7763157894736842
b= 28 178 0.7807017543859649
b= 28 179 0.7850877192982456
b= 28 180 0.7894736842105263
b= 28 181 0.793859649122807
b= 28 182 0.7982456140350878
b= 28 183 0.8026315789473685
b= 28 184 0.8070175438596491
b= 28 185 0.8114035087719298
b= 28 186 0.8157894736842105
b= 28 187 0.8201754385964912
b= 28 188 0.8245614035087719
b= 28 189 0.8289473684210527
b= 28 190 0.8333333333333334
b= 28 191 0.8377192982456141
b= 28 192 0.8421052631578947
```

In [57]:
```
b=27
accurate_rows=0
for x,y in zip(x_binarised_test,y_test):
    y_pred=(np.sum(x)>=b)
    accurate_rows+=(y==y_pred)
    print('b=',b,accurate_rows, accurate_rows/x_binarised_test.shape[0])
```

```
b= 27 178 0.7807017543859649
b= 27 179 0.7850877192982456
b= 27 180 0.7894736842105263
b= 27 181 0.793859649122807
b= 27 182 0.7982456140350878
b= 27 183 0.8026315789473685
b= 27 184 0.8070175438596491
b= 27 185 0.8114035087719298
b= 27 186 0.8157894736842105
b= 27 187 0.8201754385964912
b= 27 188 0.8245614035087719
b= 27 189 0.8289473684210527
b= 27 190 0.8333333333333334
b= 27 191 0.8377192982456141
b= 27 192 0.8421052631578947
b= 27 193 0.8464912280701754
b= 27 194 0.8508771929824561
b= 27 195 0.8552631578947368
b= 27 196 0.8596491228070176
```

In [ ]:
```
1
```