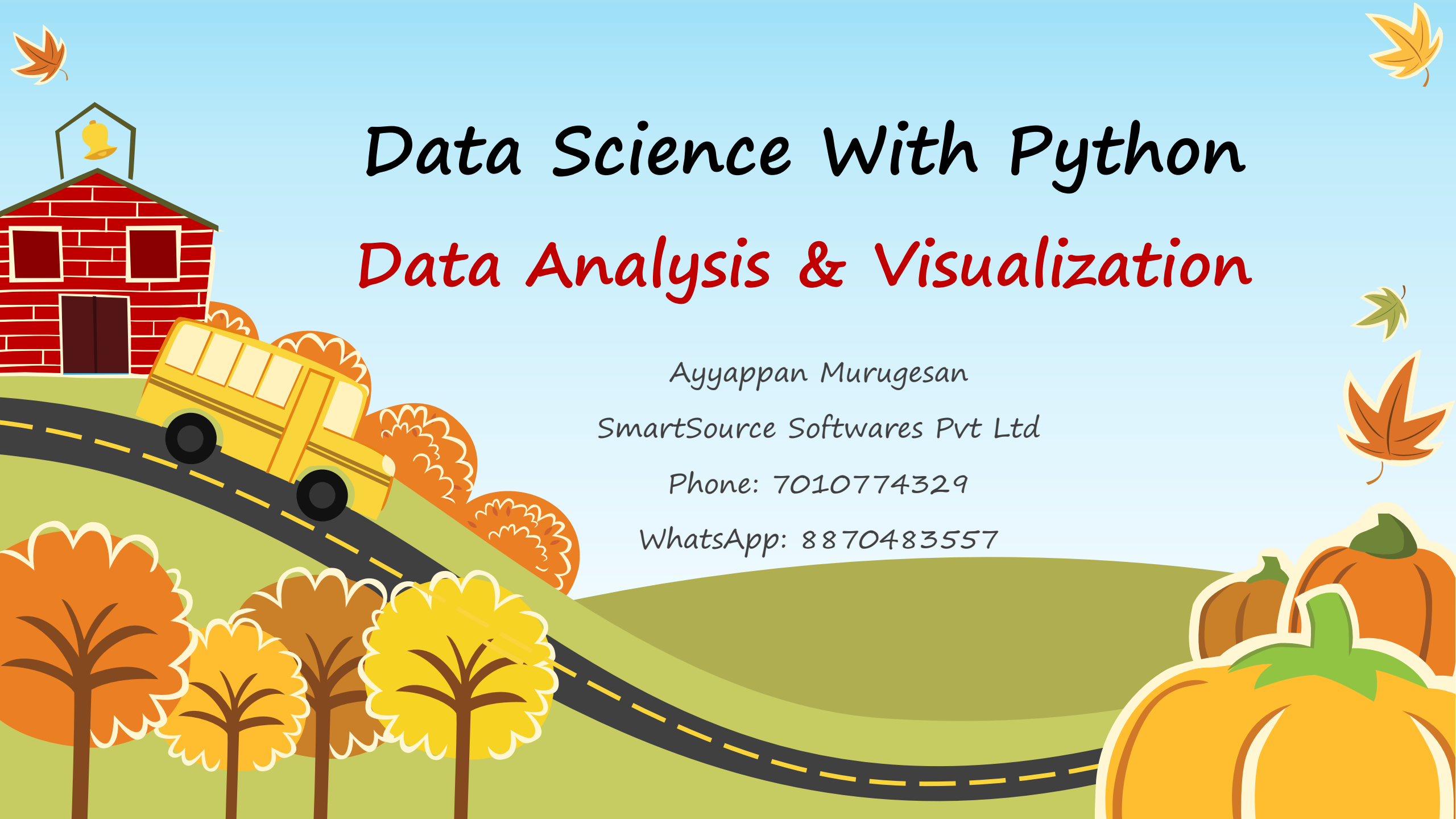


Data Science With Python

Data Analysis & Visualization

Ayyappan Murugesan

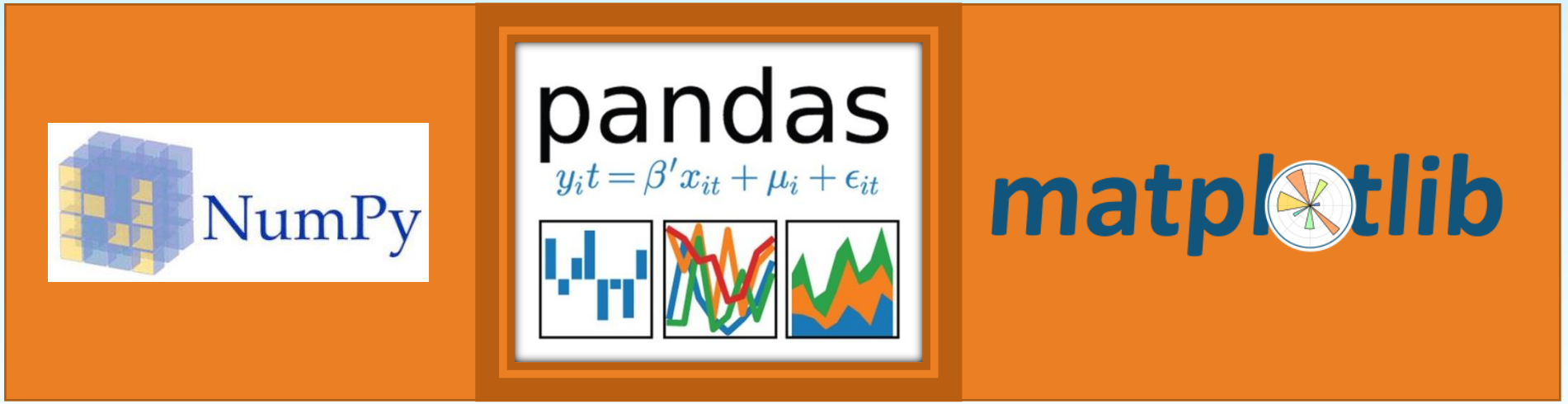
SmartSource Softwares Pvt Ltd

Phone: 7010774329

WhatsApp: 8870483557

Data Analysis

Data analysis is a process of inspecting, cleansing, transforming, and modeling data with the goal of discovering useful information, informing conclusions, and supporting decision-making.



NumPy – The fundamental package for scientific computing

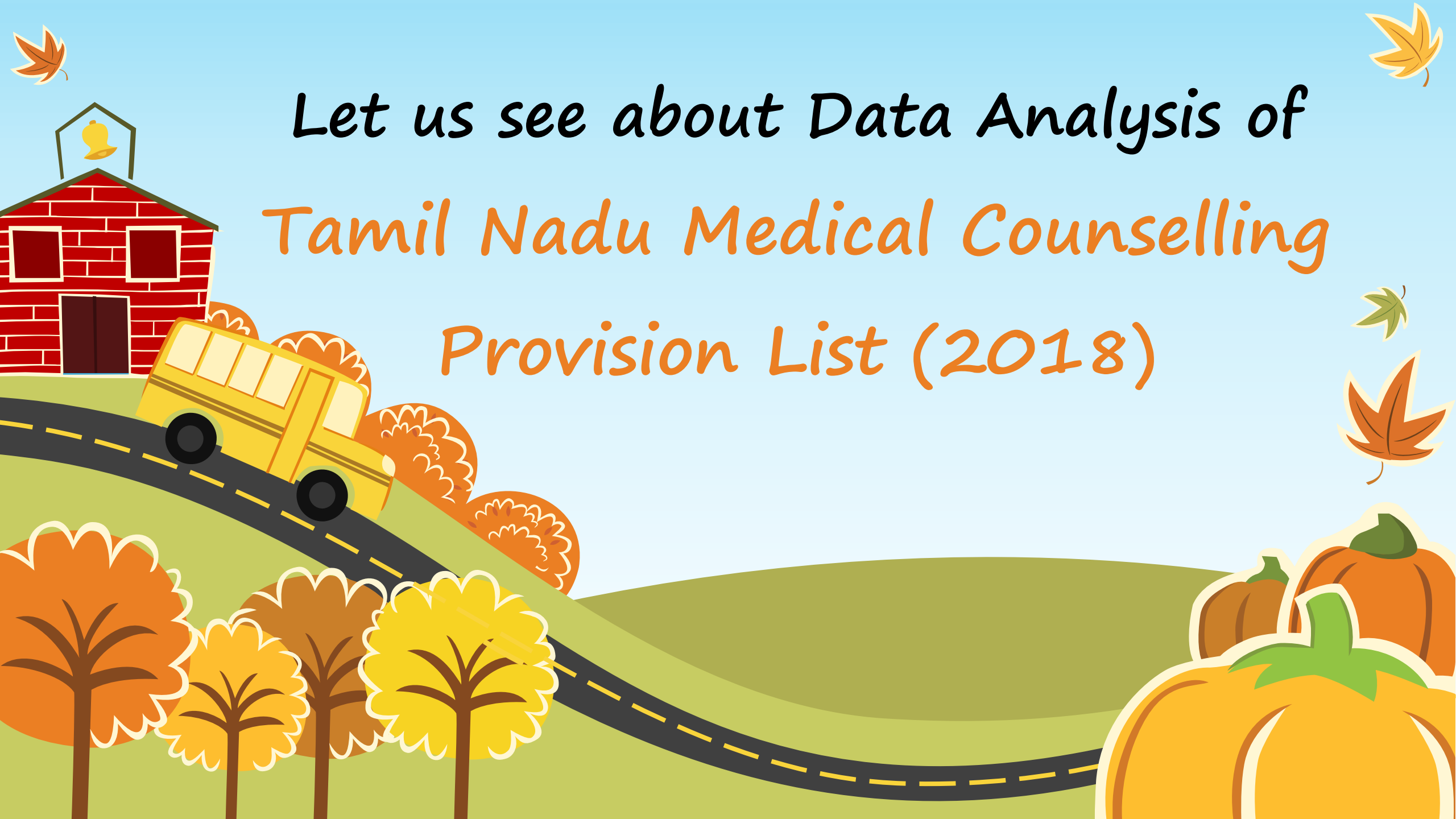
```
import numpy as np

a = np.array([2,3,4])

print(a.dtype)

b = np.array([(1.5,2,3), (4,5,6)])

print(b)

c = np.array( [ [1,2], [3,4] ], dtype=complex )

np.zeros( (3,4) )

np.ones( (2,3,4), dtype=np.int16 )

np.arange( 10, 30, 5 )

np.linspace( 0, 2, 9 )
```

```
b.min(axis=0)

b.sum(axis=0)            # axis=0 - Col

b.sum(axis=1)            # axis=1 - Row

b.cumsum(axis=1)         # a,a+b,a+b+c

a = np.arange(10)**3

print(a[2:5])

a[:6:2] = -1000

print(b[0:5, 1])     # each row in 2nd column of b

print(b[1:3, : ])    # each col in the 2&3 row of b

a.reshape(6,2)
```

Data Analysis - Pandas

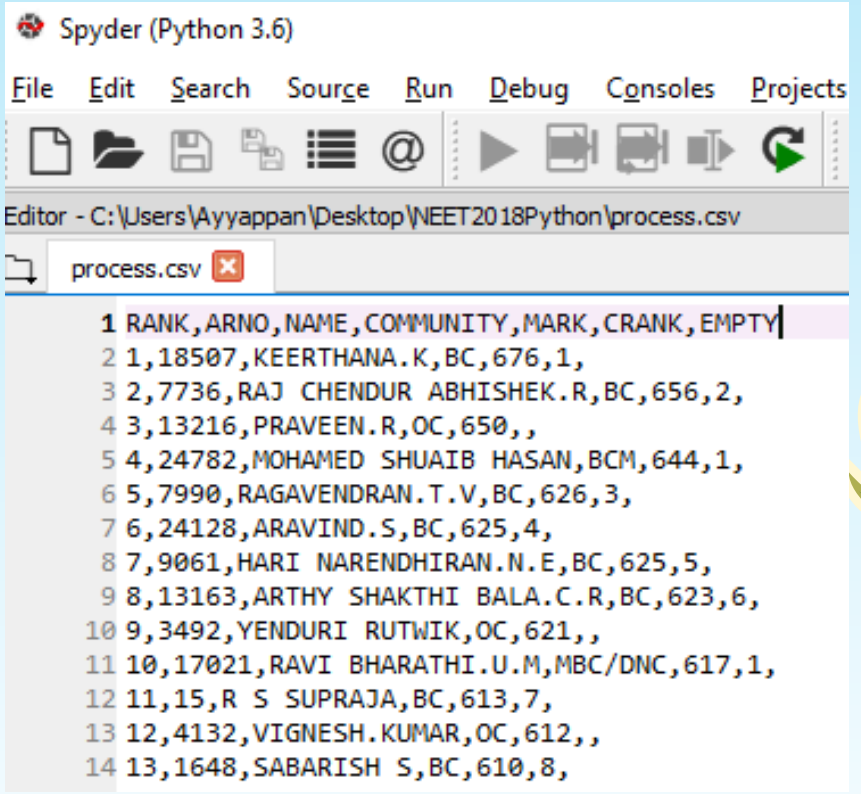
pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python

- Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet
- Ordered and unordered (not necessarily fixed-frequency) time series data.
- Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels


Let us see about Data Analysis of
Tamil Nadu Medical Counselling
Provision List (2018)

Consider ....

- We use data of Tamil Nadu Medical Counselling – Provisional List for NEET 2018 Candidates
- All the data available in the CSV File Format (process.csv)
- We use this data for **data analysis with Pandas Python Package**



Let Start Practice with Pandas

importing required python packages

```
import pandas as pd

import numpy as np
```

Read Data From CSV using Pandas

```
tneet=pd.read_csv("process.csv")
#tneet is Data Frame
```

```
print(tneet)
```

```
       RANK   ARNO                    NAME COMMUNITY MARK  CRANK EMPTY
0         1  18507            KEERTHANA.K        BC  676      1   NaN
1         2   7736  RAJ CHENDUR ABHISHEK.R        BC  656      2   NaN
2         3  13216             PRAVEEN.R        OC  650    NaN   NaN
3         4  24782   MOHAMED SHUAIB HASAN       BCM  644      1   NaN
4         5   7990        RAGAVENDRAN.T.V        BC  626      3   NaN
5         6  24128            ARAVIND.S         BC  625      4   NaN
6         7   9061   HARI NARENDHIRAN.N.E        BC  625      5   NaN
...     ...                        ...       ...  ...    ...   ...
25414 25412   5042         MADHUMITHA R         BC   96  12415   NaN
25415 25413   6914      METILDA SHARLIN C       BC   96  12416   NaN
25416 25414  16843          MANIDHARANI D       BC   96  12417   NaN
25417 25415  25984           GUNASEKAR S        SC   96   3864   NaN
25418 25416  21628      MOHAMMED RAYYAN A       BCM  96   1394   NaN
25419 25417   5806          ABDUL RAHIM P       BCM  96   1395   NaN

[25420 rows x 7 columns]
```

Get the number of records & column

```
print(tneet.index)
```

```
print(tneet.columns)
```

```
RangeIndex(start=0, stop=25420, step=1)
Index(['RANK', 'ARNO', 'NAME', 'COMMUNITY', 'MARK', 'CRANK', 'EMPTY'],
```

Usage of Head function

#Viewing the first 5 lines

print(tneet.head())

Index

	RANK	ARNO	NAME	COMMUNITY	MARK	CRANK	EMPTY
0	1	18507	KEERTHANA.K	BC	676	1	NaN
1	2	7736	RAJ CHENDUR ABHISHEK.R	BC	656	2	NaN
2	3	13216	PRAVEEN.R	OC	650	NaN	NaN
3	4	24782	MOHAMED SHUAIB HASAN	BCM	644	1	NaN
4	5	7990	RAGAVENDRAN.T.V	BC	626	3	NaN

#Viewing the first n lines

print(tneet.head(2))

	RANK	ARNO	NAME	COMMUNITY	MARK	CRANK	EMPTY
0	1	18507	KEERTHANA.K	BC	676	1	NaN
1	2	7736	RAJ CHENDUR ABHISHEK.R	BC	656	2	NaN

Set specific column as Index Column

```
tneet=tneet.set_index("RANK")
print(tneet.head())
```

Index

RANK	ARNO	NAME	COMMUNITY	MARK	CRANK	EMPTY
1	18507	KEERTHANA.K	BC	676	1	NaN
2	7736	RAJ CHENDUR ABHISHEK.R	BC	656	2	NaN
3	13216	PRAVEEN.R	OC	650	NaN	NaN
4	24782	MOHAMED SHUAIB HASAN	BCM	644	1	NaN
5	7990	RAGAVENDRAN.T.V	BC	626	3	NaN

Accessing specific Field's Data

```
name=tneet['NAME']
print(name)
```

```
RANK
1                    KEERTHANA.K
2            RAJ CHENDUR ABHISHEK.R
3                      PRAVEEN.R
4              MOHAMED SHUAIB HASAN
5                   RAGAVENDRAN.T.V
6                      ARAVIND.S
7              HARI NARENDHIRAN.N.E
                  ...
25411                ARUNACHAL.J.K
25412                 MADHUMITHA R
25413              METILDA SHARLIN C
25414                MANIDHARANI D
25415                  GUNASEKAR S
25416              MOHAMMED RAYYAN A
25417                  ABDUL RAHIM P
Name: NAME, Length: 25420, dtype: object
```

```
subset=tneet[['NAME','MARK']]
print(subset)
```

```
RANK
1                    KEERTHANA.K  676
2            RAJ CHENDUR ABHISHEK.R  656
3                      PRAVEEN.R  650
4              MOHAMED SHUAIB HASAN  644
5                   RAGAVENDRAN.T.V  626
6                      ARAVIND.S  625
7              HARI NARENDHIRAN.N.E  625
...                             ...  ...
25411                ARUNACHAL.J.K   96
25412                 MADHUMITHA R   96
25413              METILDA SHARLIN C   96
25414                MANIDHARANI D   96
25415                  GUNASEKAR S   96
25416              MOHAMMED RAYYAN A   96
25417                  ABDUL RAHIM P   96

[25420 rows x 2 columns]
```

label-based indexing

```
subset=tneet.loc[['1','10']]
print(subset)
```

RANK	ARNO	NAME	COMMUNITY	MARK	CRANK	EMPTY
1	18507	KEERTHANA.K	BC	676	1	NaN
10	17021	RAVI BHARATHI.U.M	MBC/DNC	617	1	NaN

```
subset=tneet.loc[:'5']
print(subset)
```

RANK	ARNO	NAME	COMMUNITY	MARK	CRANK	EMPTY
1	18507	KEERTHANA.K	BC	676	1	NaN
2	7736	RAJ CHENDUR ABHISHEK.R	BC	656	2	NaN
3	13216	PRAVEEN.R	OC	650	NaN	NaN
4	24782	MOHAMED SHUAIB HASAN	BCM	644	1	NaN
5	7990	RAGAVENDRAN.T.V	BC	626	3	NaN

integer-based indexing

```
subset=tneet.iloc[[0,9]]
print(subset)
```

RANK	ARNO	NAME	COMMUNITY	MARK	CRANK	EMPTY
1	18507	KEERTHANA.K	BC	676	1	NaN
10	17021	RAVI BHARATHI.U.M	MBC/DNC	617	1	NaN

```
subset=tneet.iloc[:5]
print(subset)
```

RANK	ARNO	NAME	COMMUNITY	MARK	CRANK	EMPTY
1	18507	KEERTHANA.K	BC	676	1	NaN
2	7736	RAJ CHENDUR ABHISHEK.R	BC	656	2	NaN
3	13216	PRAVEEN.R	OC	650	NaN	NaN
4	24782	MOHAMED SHUAIB HASAN	BCM	644	1	NaN
5	7990	RAGAVENDRAN.T.V	BC	626	3	NaN

label-based indexing

```
subset=tneet.loc[:'2',['NAME','MARK']]
print(subset)
```

RANK	NAME	MARK
1	KEERTHANA.K	676
2	RAJ CHENDUR ABHISHEK.R	656

integer-based indexing

```
subset=tneet.iloc[:2,[1,3]]
print(subset)
```

RANK	NAME	MARK
1	KEERTHANA.K	676
2	RAJ CHENDUR ABHISHEK.R	656

Del specific column

```
del tneet['EMPTY']
print(tneet)
```

RANK	ARNO	NAME	COMMUNITY	MARK	CRANK
1	18507	KEERTHANA.K	BC	676	1
2	7736	RAJ CHENDUR ABHISHEK.R	BC	656	2
3	13216	PRAVEEN.R	OC	650	NaN
4	24782	MOHAMED SHUAIB HASAN	BCM	644	1
5	7990	RAGAVENDRAN.T.V	BC	626	3
6	24128	ARAVIND.S	BC	625	4
7	9061	HARI NARENDHIRAN.N.E	BC	625	5
...	...	...	...	...	...
25411	5062	ARUNACHAL.J.K	MBC/DNC	96	5491
25412	5042	MADHUMITHA R	BC	96	12415
25413	6914	METILDA SHARLIN C	BC	96	12416
25414	16843	MANIDHARANI D	BC	96	12417
25415	25984	GUNASEKAR S	SC	96	3864
25416	21628	MOHAMMED RAYYAN A	BCM	96	1394
25417	5806	ABDUL RAHIM P	BCM	96	1395

[25420 rows x 5 columns]

Slicing Data using condition statement

```
oc=tneet[tneet.COMMUNITY=='OC']
bc=tneet[tneet.COMMUNITY=='BC']
bcm=tneet[tneet.COMMUNITY=='BCM']
mbc=tneet[tneet.COMMUNITY=='MBC/DNC']
sc=tneet[tneet.COMMUNITY=='SC']
sca=tneet[tneet.COMMUNITY=='SCA']
st=tneet[tneet.COMMUNITY=='ST']

print('OC:\t',len(oc))
print('BC:\t',len(bc))
print('BCM:\t',len(bcm))
print('MBC:\t',len(mbc))
print('SC:\t',len(sc))
print('SCA:\t',len(sca))
print('ST:\t',len(st))
```

```
OC:     1690
BC:     12417
BCM:    1395
MBC:    5491
SC:     3864
SCA:    447
ST:     113
```

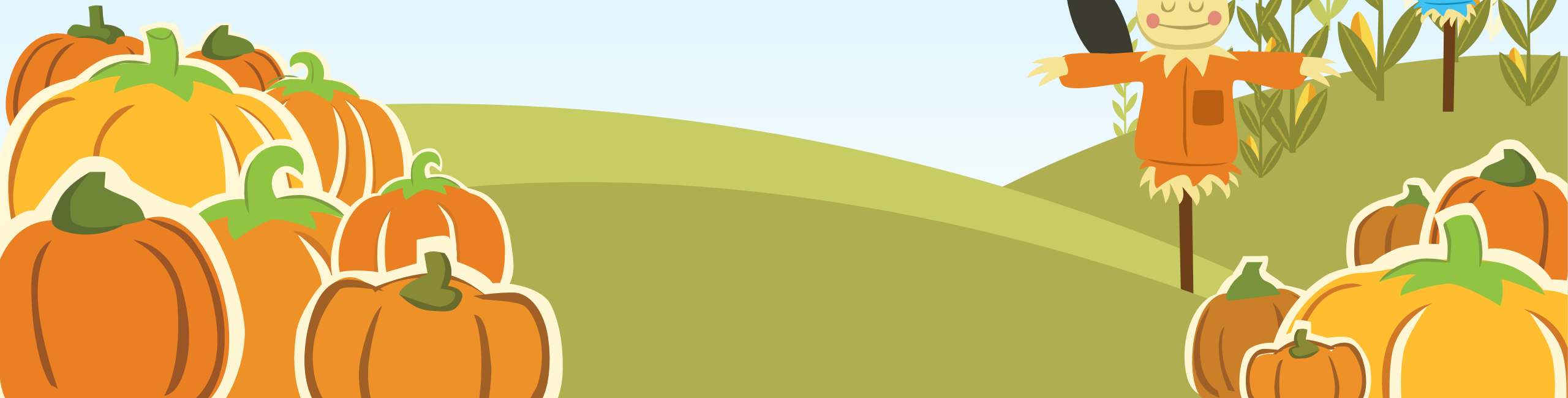
Wow,Now we can organizing big data set easily.

But How we can understand 25K Records?

Oh Bad, It is not easy.

Don't Worry

Data Visualization help us for it



matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms

- Line Plot
- Histograms
- Paths
- Three-dimensional plotting
- Streamplot
- Ellipses

- Bar charts
- Pie charts
- Tables
- Scatter plots
- Filled curves
- Log plots
- Polar plots

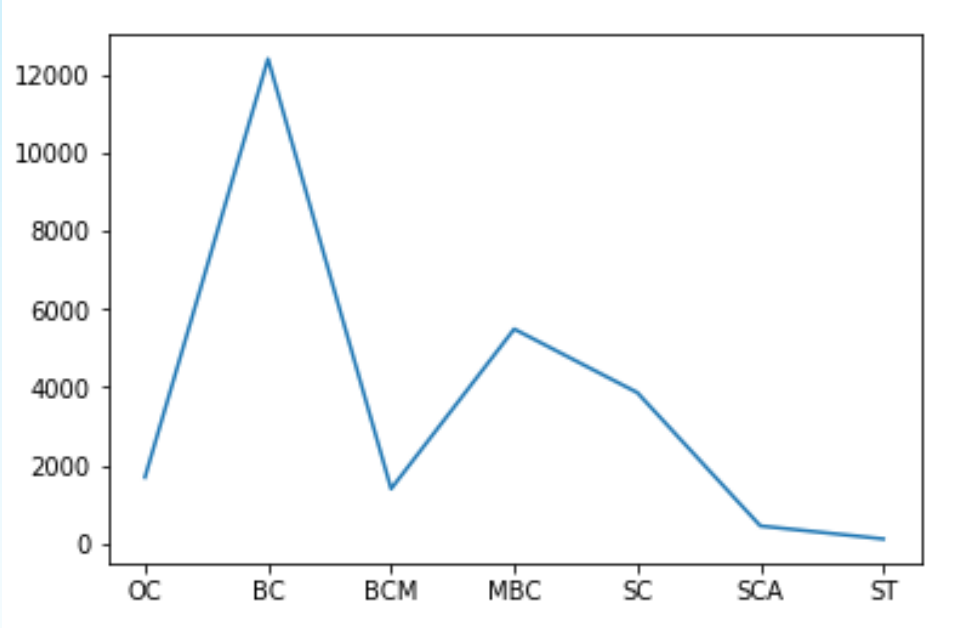
- Legends
- Pyplot
- ColorMaps
- Text annotation
- Wireframe plots
- Surface plots
- Contour plots

importing required python packages

```
import matplotlib.pyplot as plt
```

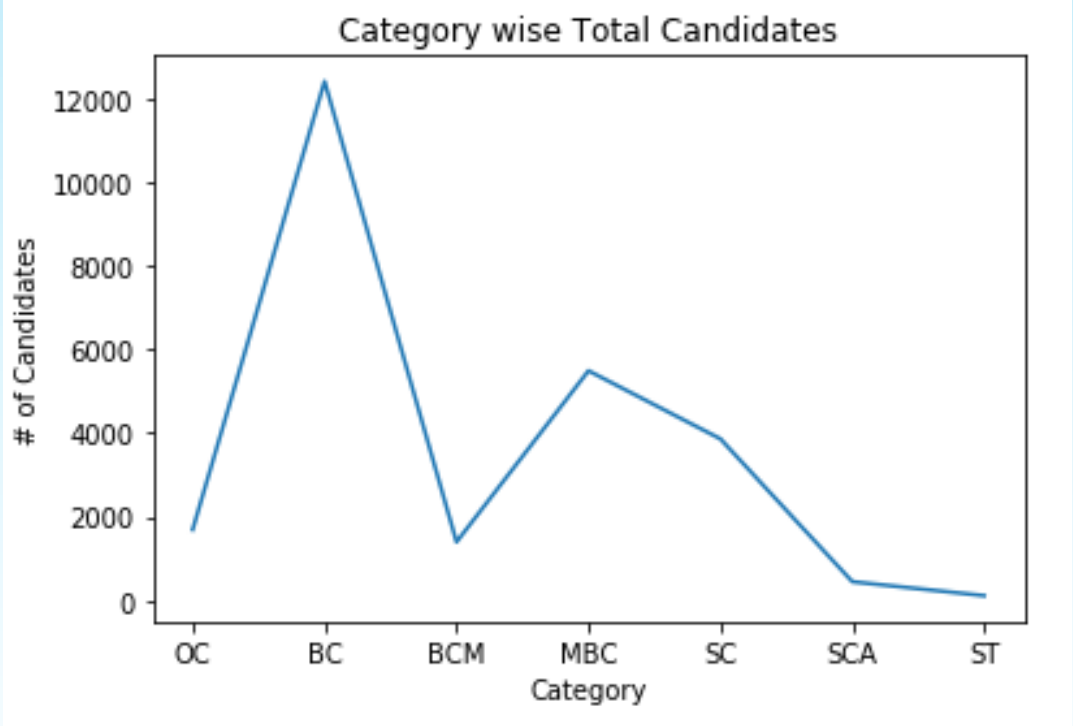
Plot chart with data

```
x=['OC','BC','BCM','MBC','SC','SCA','ST']

y=[len(oc),len(bc),len(bcm),len(mbc),len(sc),len(sca),len(st)]

plt.plot(x,y)

plt.show()
```



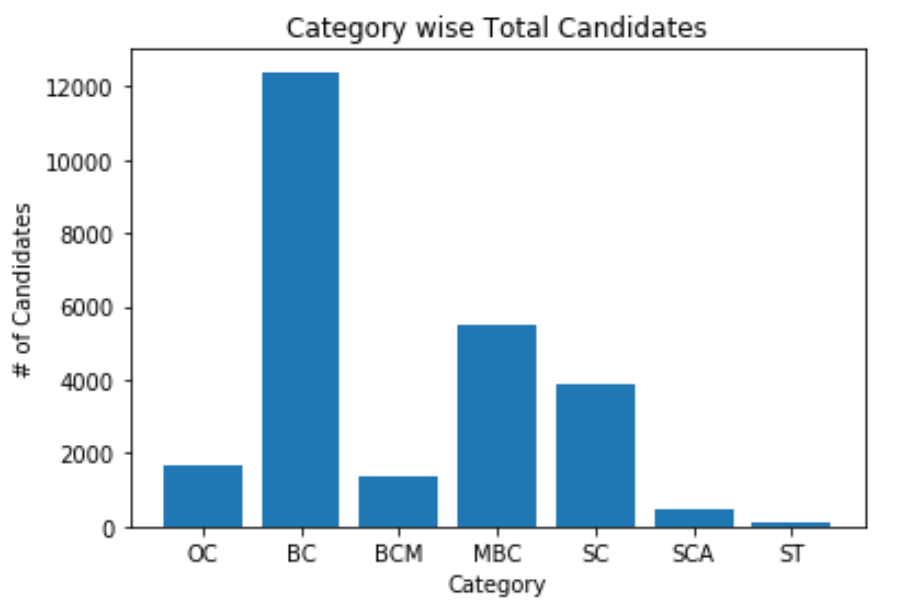
Specify Title, X Label & Y Label

```
plt.title('Category wise Total Candidates');

plt.xlabel('Category')

plt.ylabel('# of Candidates')

plt.show()
```



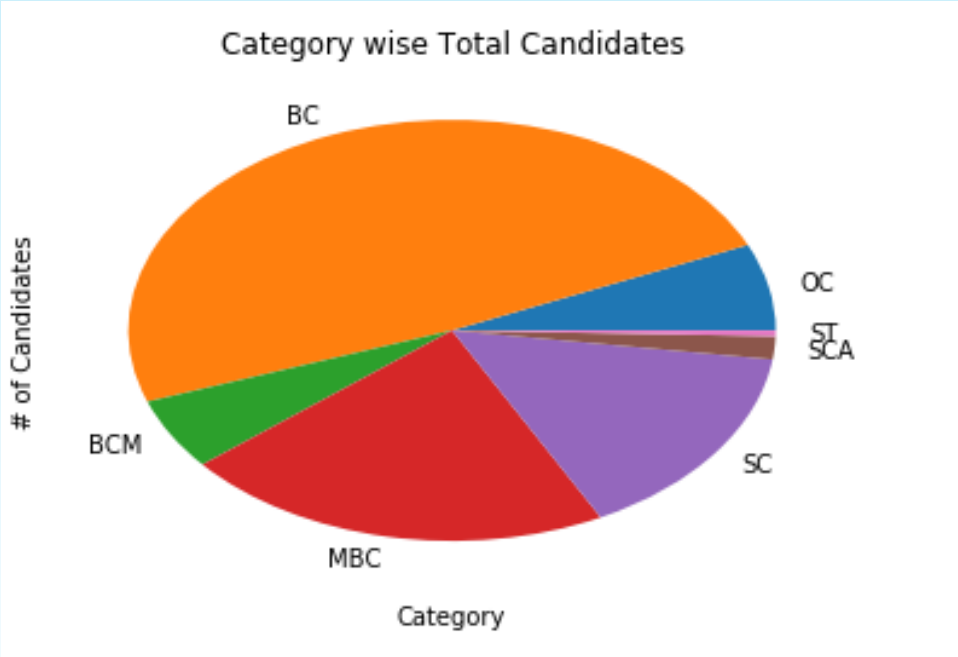
Draw Bar Graph

```
x=['OC','BC','BCM','MBC','SC','SCA','ST']
y=[len(oc),len(bc),len(bcm),len(mbc),len(sc),len(sca),len(st)]
plt.bar(x,y)
plt.title('Category wise Total Candidates');
plt.xlabel('Category')
plt.ylabel('# of Candidates')
plt.show()
```

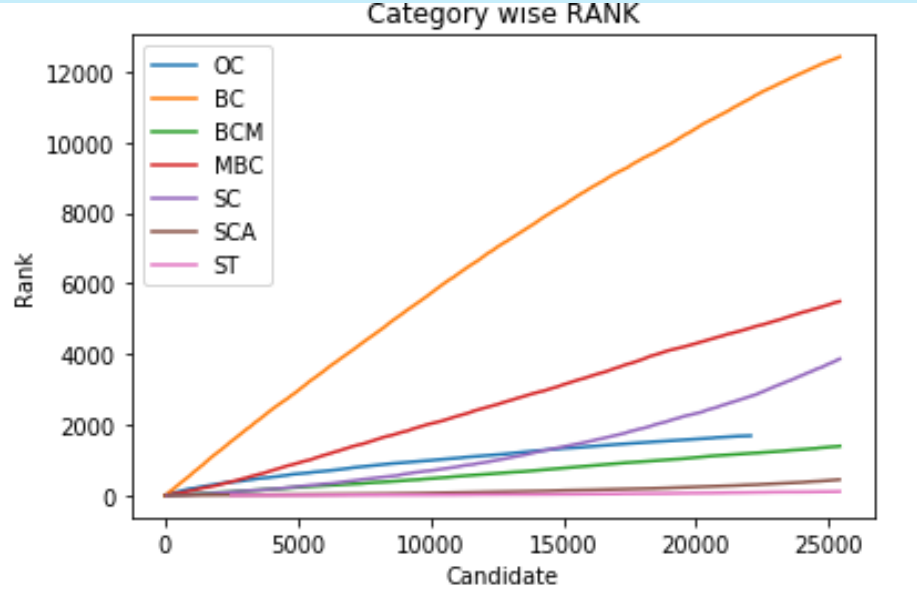


Draw Pip Chart

```
x=['OC','BC','BCM','MBC','SC','SCA','ST']
y=[len(oc),len(bc),len(bcm),len(mbc),len(sc),len(sca),len(st)]
plt.pie(y,labels=x)
plt.title('Category wise Total Candidates');
plt.xlabel('Category')
plt.ylabel('# of Candidates')
plt.show()
```



Example for Multi line chart with legends



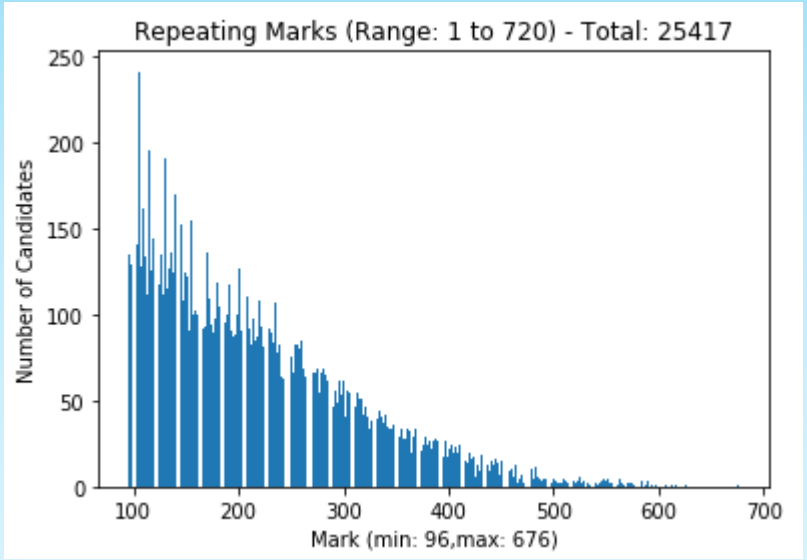

```
plt.plot(oc.index.astype('int'),np.arange(1,len(oc)+1))
plt.plot(bc.index.astype('int'),bc.CRANK.astype('int'))
plt.plot(bcm.index.astype('int'),bcm.CRANK.astype('int'))
plt.plot(mbc.index.astype('int'),mbc.CRANK.astype('int'))
plt.plot(sc.index.astype('int'),sc.CRANK.astype('int'))
plt.plot(sca.index.astype('int'),sca.CRANK.astype('int'))
plt.plot(st.index.astype('int'),st.CRANK.astype('int'))
plt.title('Category wise Rank Range');
plt.xlabel('Candidate')
plt.ylabel('Rank')
plt.legend(['OC','BC','BCM','MBC','SC','SCA','ST'])
plt.show()
```

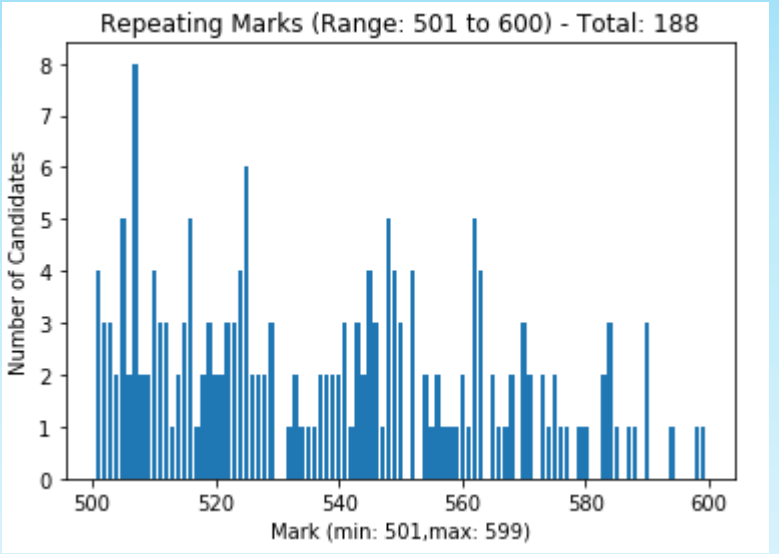

We learned both Pandas & Matplotlib
packages of Python.
Let us start , analysis given data set

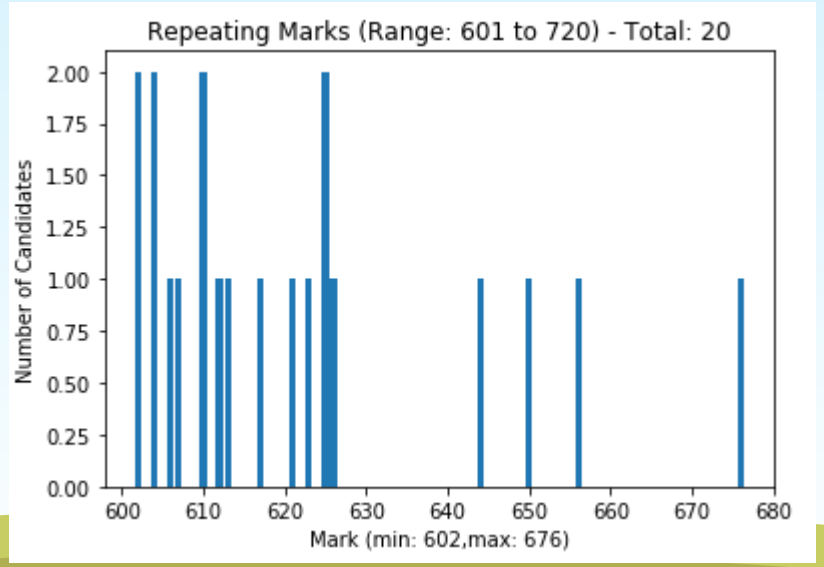
Count the marks, which is same mark to more than one students

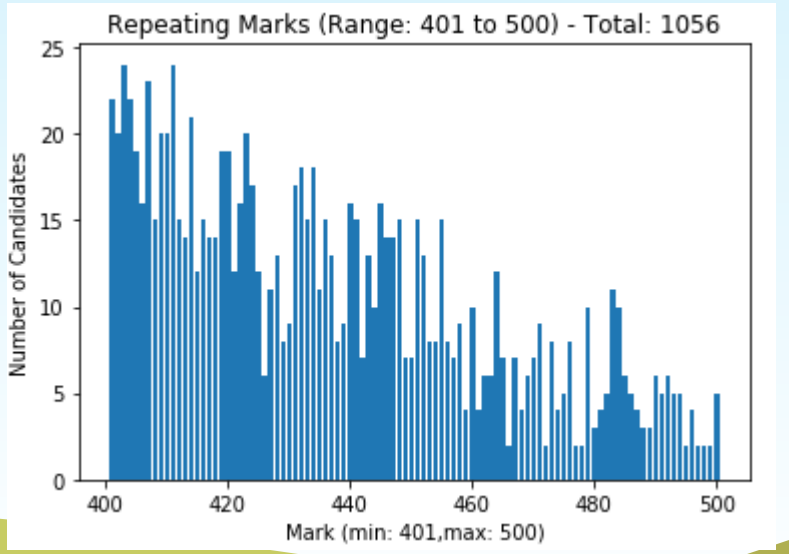
```
def repeatMarks(groups,start,stop):
    rgroups=groups.loc[start:stop]
    plt.bar(rgroups.index,rgroups.values)
    plt.title("Repeating Marks (Range: {0} to {1}) - Total: {2}".format(start,stop,np.sum(rgroups.values)))
    plt.xlabel('Mark (min: {0},max: {1})'.format(min(rgroups.index),max(rgroups.index)))
    plt.ylabel('Number of Candidates')
    plt.show()
```

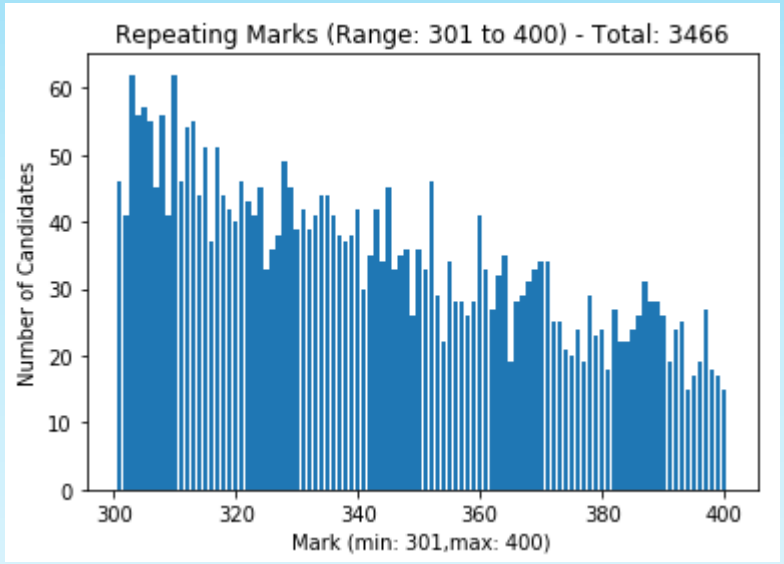
```
markgroup=tneet.groupby(['MARK'])

groups=markgroup.size()

repeatMarks(plt,groups,1,720)

repeatMarks(plt,groups,1,100)

repeatMarks(plt,groups,101,200)

repeatMarks(plt,groups,201,300)

repeatMarks(plt,groups,301,400)

repeatMarks(plt,groups,401,500)

repeatMarks(plt,groups,501,600)

repeatMarks(plt,groups,601,720)
```

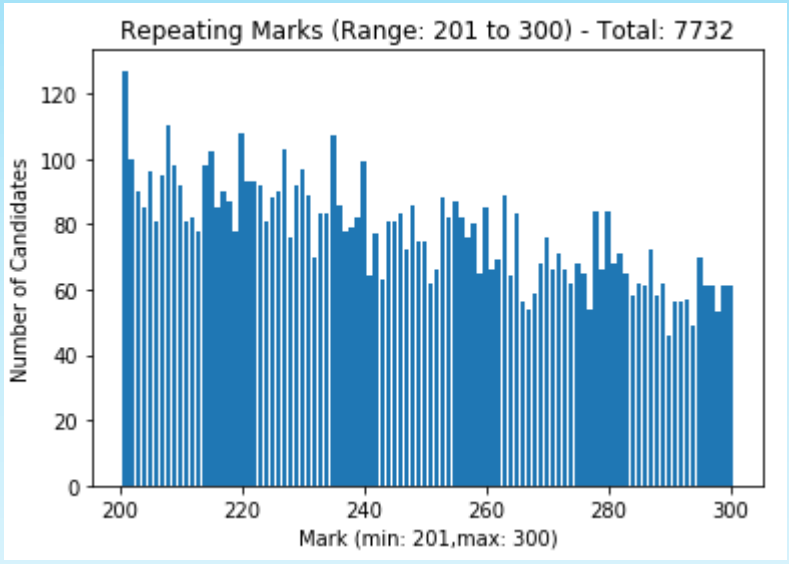

Repeating Marks (Range: 1 to 720) - Total: 25417
Number of Candidates
Mark (min: 96,max: 676)

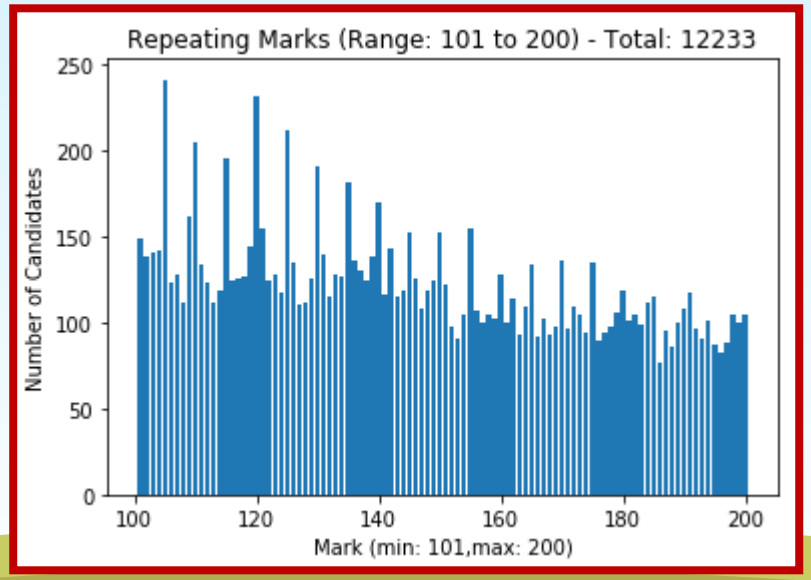

Repeating Marks (Range: 501 to 600) - Total: 188
Number of Candidates
Mark (min: 501,max: 599)

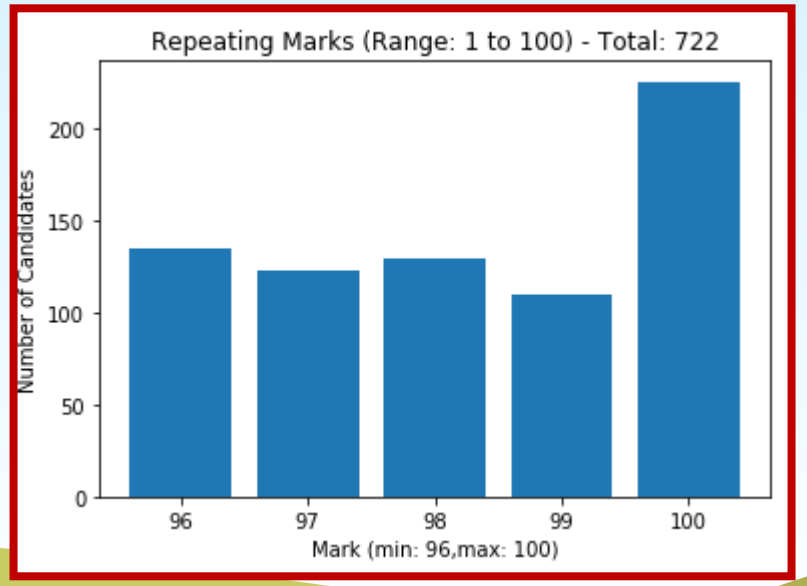

Repeating Marks (Range: 601 to 720) - Total: 20
Number of Candidates
Mark (min: 602,max: 676)


Repeating Marks (Range: 401 to 500) - Total: 1056
Number of Candidates
Mark (min: 401,max: 500)


Repeating Marks (Range: 301 to 400) - Total: 3466
Number of Candidates
Mark (min: 301,max: 400)

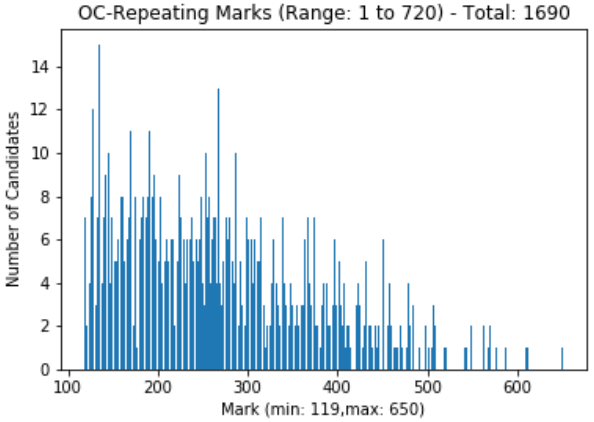

Repeating Marks (Range: 201 to 300) - Total: 7732
Number of Candidates
Mark (min: 201,max: 300)

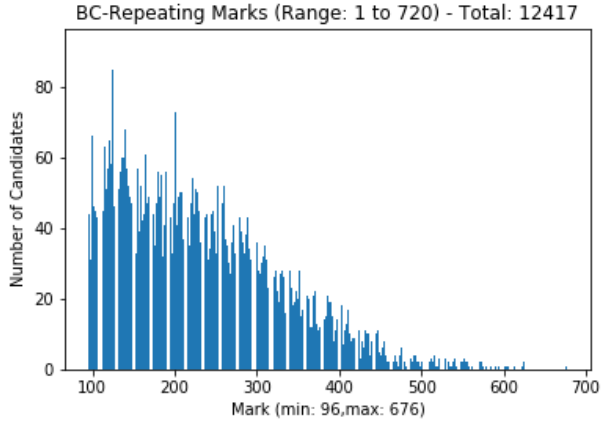

Repeating Marks (Range: 101 to 200) - Total: 12233
Number of Candidates
Mark (min: 101,max: 200)

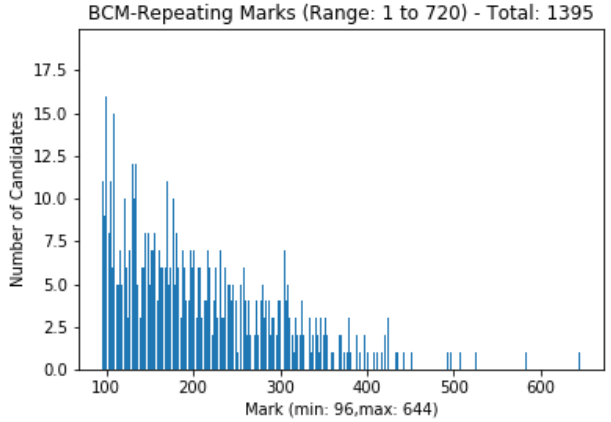

Repeating Marks (Range: 1 to 100) - Total: 722
Number of Candidates
Mark (min: 96,max: 100)

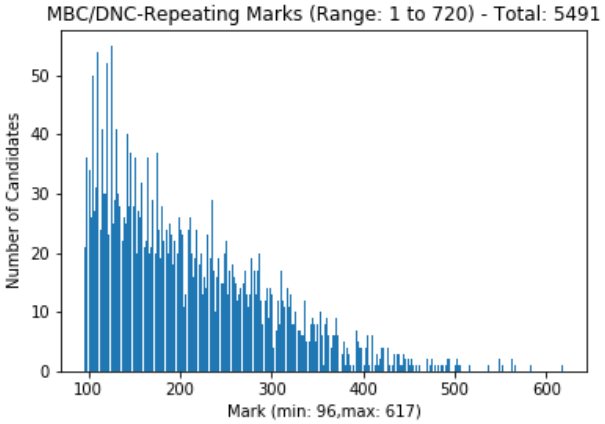
Category wise Repeating Mark Analysis

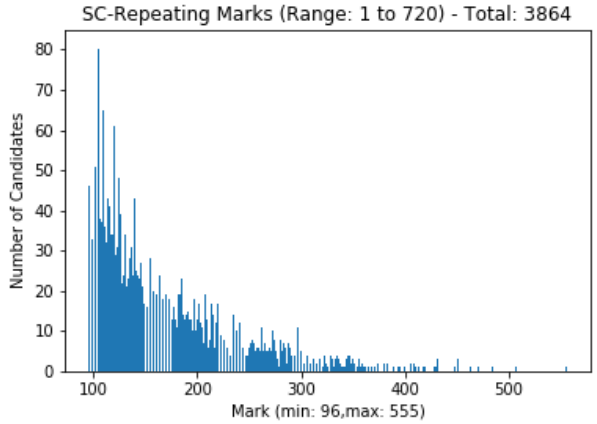
```
def repeatMarks(category,groups,start,stop):
    rgroups=groups.loc[start:stop]
    if len(rgroups)>0:
        plt.bar(rgroups.index,rgroups.values)
        plt.title("{3}-Repeating Marks (Range: {0} to {1}) - Total: {2}".format(start,stop,np.sum(rgroups.values),category))
        plt.xlabel('Mark (min: {0},max: {1})'.format(min(rgroups.index),max(rgroups.index)))
        plt.ylabel('Number of Candidates')
        plt.savefig('report/categoryRepeatMark/{0}-{1}-{2}.png'.format(category.replace("/","-"),start,stop))
        plt.show()
for category in ['OC','BC','BCM','MBC/DNC','SC','SCA','ST']:
    markgroup=tneet[tneet.COMMUNITY==category].groupby(['MARK'])
    groups=markgroup.size()
    repeatMarks(category,groups,1,720)
    repeatMarks(category,groups,601,720)
    repeatMarks(category,groups,501,600)
    repeatMarks(category,groups,401,500)
    repeatMarks(category,groups,301,400)
    repeatMarks(category,groups,201,300)
    repeatMarks(category,groups,101,200)
    repeatMarks(category,groups,1,100)
```

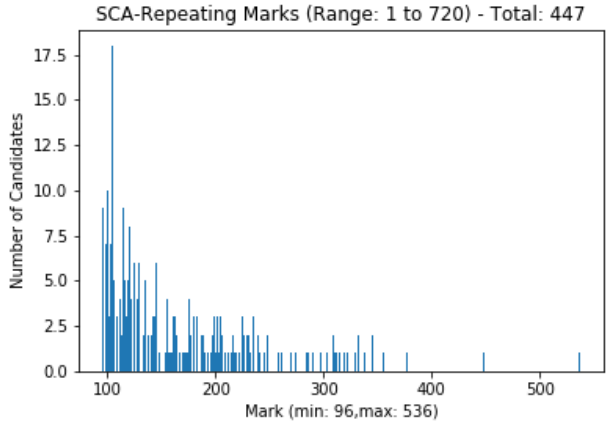

OC-Repeating Marks (Range: 1 to 720) - Total: 1690
Number of Candidates
Mark (min: 119,max: 650)

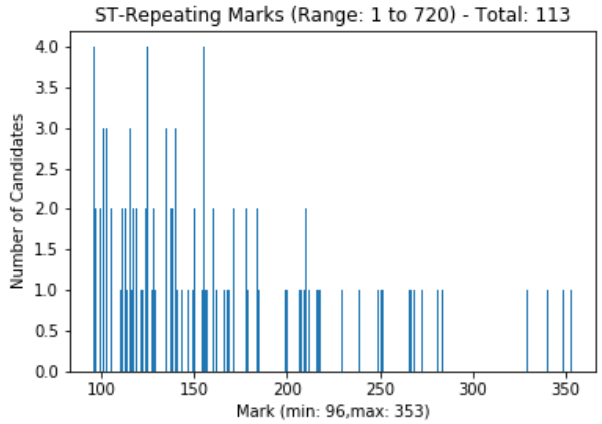

BC-Repeating Marks (Range: 1 to 720) - Total: 12417
Number of Candidates
Mark (min: 96,max: 676)

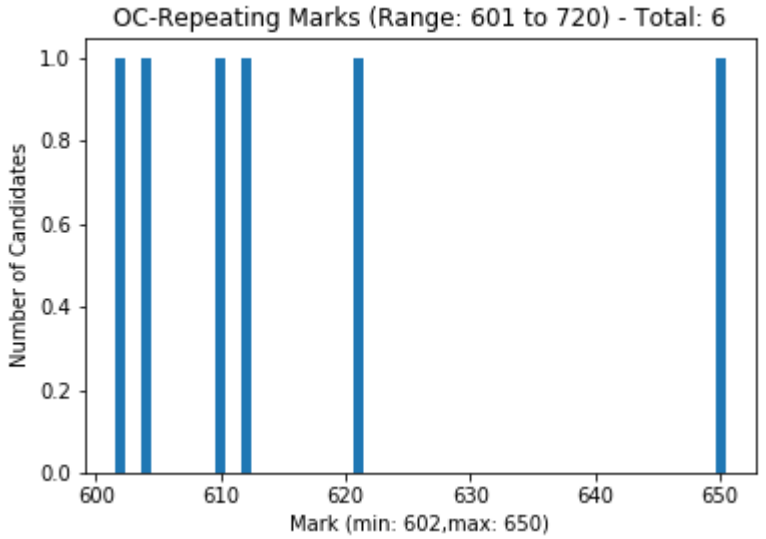

BCM-Repeating Marks (Range: 1 to 720) - Total: 1395
Number of Candidates
Mark (min: 96,max: 644)

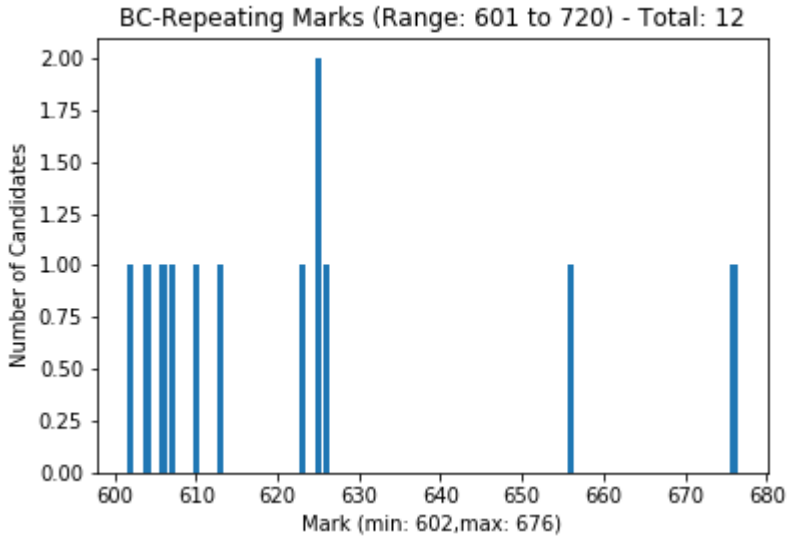

MBC/DNC-Repeating Marks (Range: 1 to 720) - Total: 5491
Number of Candidates
Mark (min: 96,max: 617)

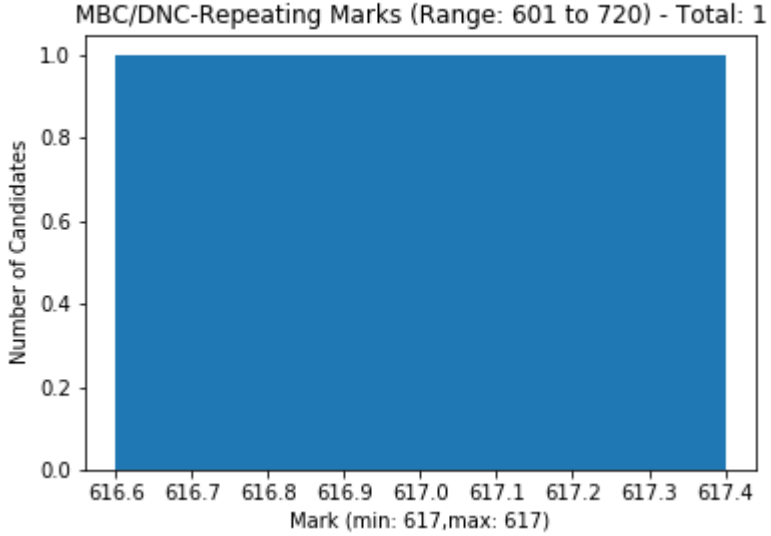

SC-Repeating Marks (Range: 1 to 720) - Total: 3864
Number of Candidates
Mark (min: 96,max: 555)

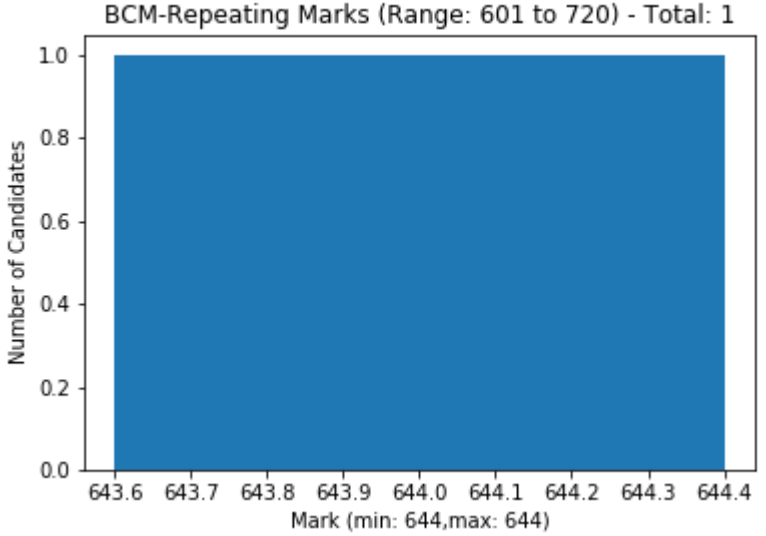

SCA-Repeating Marks (Range: 1 to 720) - Total: 447
Number of Candidates
Mark (min: 96,max: 536)

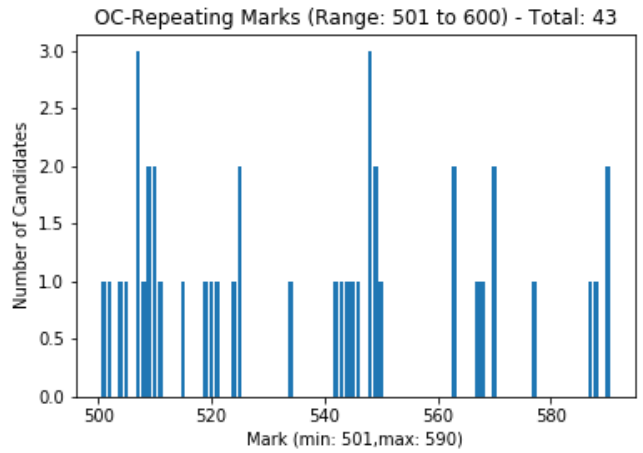

ST-Repeating Marks (Range: 1 to 720) - Total: 113
Number of Candidates
Mark (min: 96,max: 353)

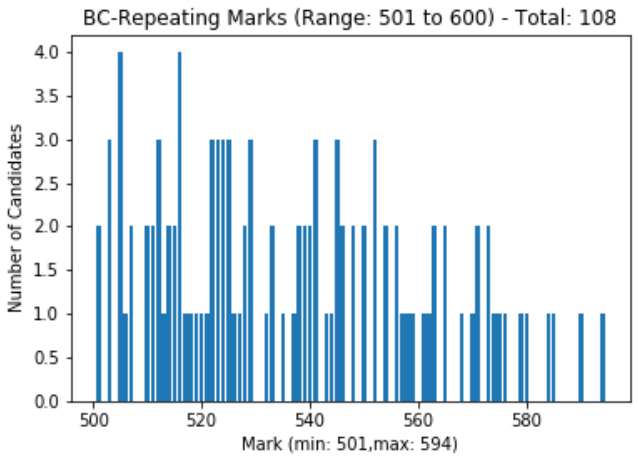

OC-Repeating Marks (Range: 601 to 720) - Total: 6
Number of Candidates
Mark (min: 602,max: 650)

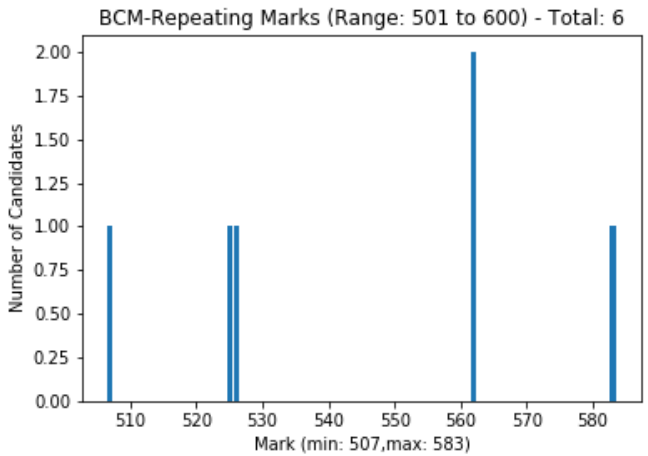

BC-Repeating Marks (Range: 601 to 720) - Total: 12
Number of Candidates
Mark (min: 602,max: 676)

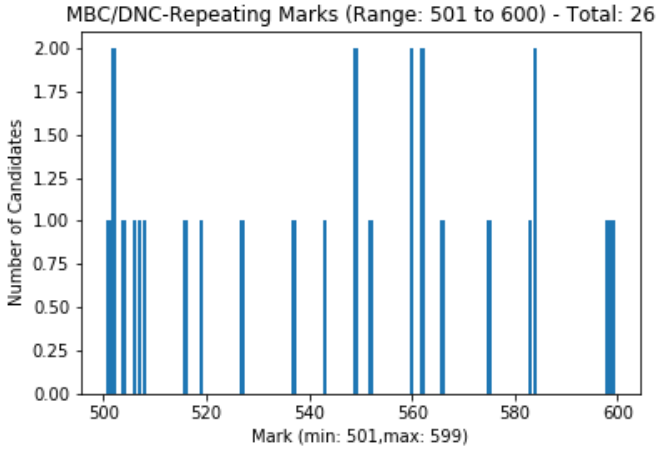

MBC/DNC-Repeating Marks (Range: 601 to 720) - Total: 1
Number of Candidates
Mark (min: 617,max: 617)

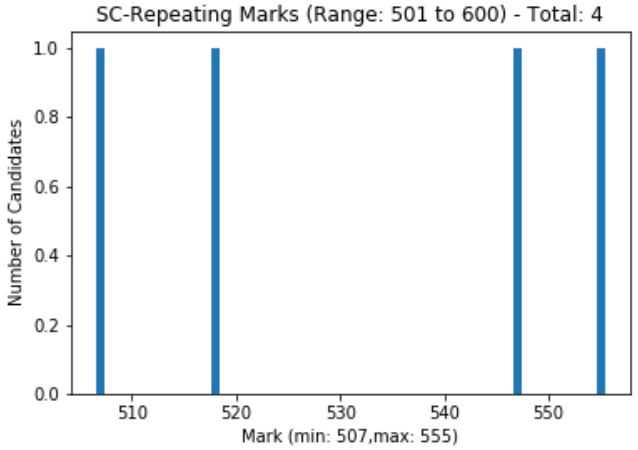

BCM-Repeating Marks (Range: 601 to 720) - Total: 1
Number of Candidates
Mark (min: 644,max: 644)

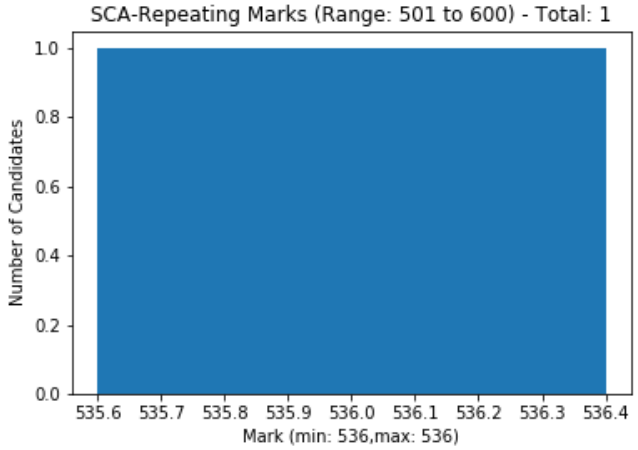

OC-Repeating Marks (Range: 501 to 600) - Total: 43
Number of Candidates
Mark (min: 501,max: 590)

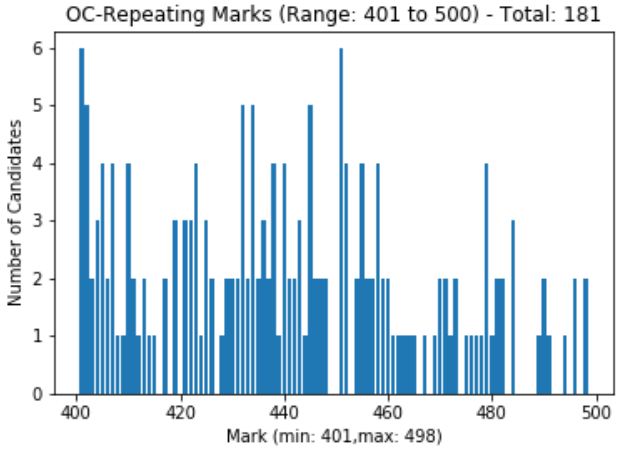

BC-Repeating Marks (Range: 501 to 600) - Total: 108
Number of Candidates
Mark (min: 501,max: 594)

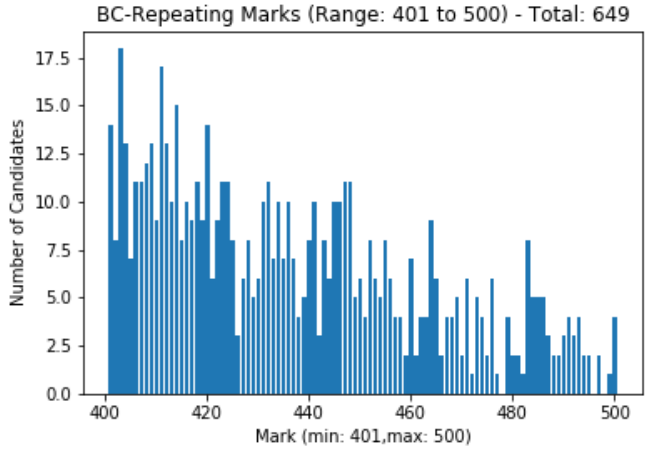

BCM-Repeating Marks (Range: 501 to 600) - Total: 6
Number of Candidates
Mark (min: 507,max: 583)

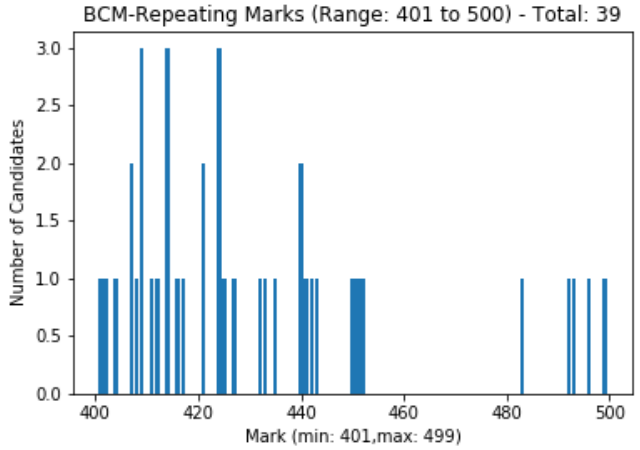

MBC/DNC-Repeating Marks (Range: 501 to 600) - Total: 26
Number of Candidates
Mark (min: 501,max: 599)

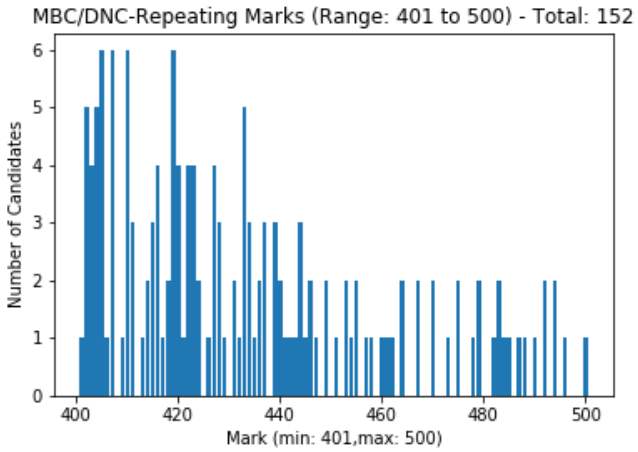

SC-Repeating Marks (Range: 501 to 600) - Total: 4
Number of Candidates
Mark (min: 507,max: 555)

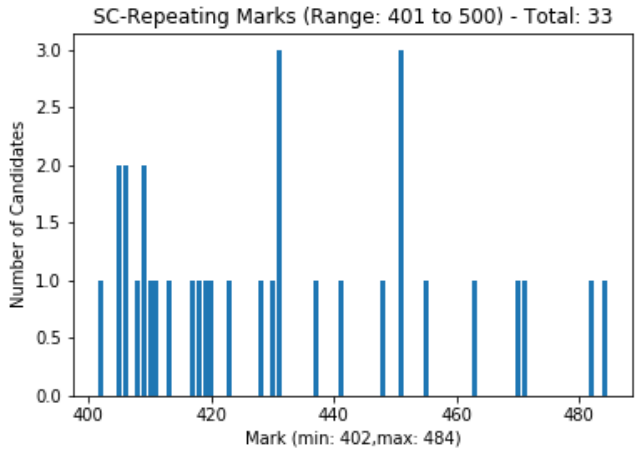

SCA-Repeating Marks (Range: 501 to 600) - Total: 1
Number of Candidates
Mark (min: 536,max: 536)

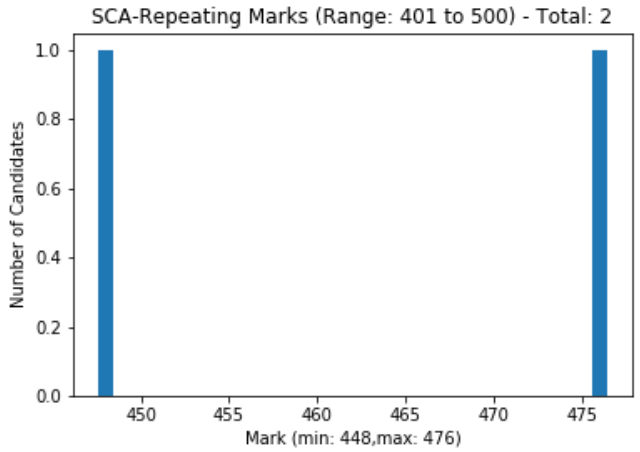

OC-Repeating Marks (Range: 401 to 500) - Total: 181
Number of Candidates
Mark (min: 401,max: 498)


BC-Repeating Marks (Range: 401 to 500) - Total: 649
Number of Candidates
Mark (min: 401,max: 500)


BCM-Repeating Marks (Range: 401 to 500) - Total: 39
Number of Candidates
Mark (min: 401,max: 499)


MBC/DNC-Repeating Marks (Range: 401 to 500) - Total: 152
Number of Candidates
Mark (min: 401,max: 500)


SC-Repeating Marks (Range: 401 to 500) - Total: 33
Number of Candidates
Mark (min: 402,max: 484)

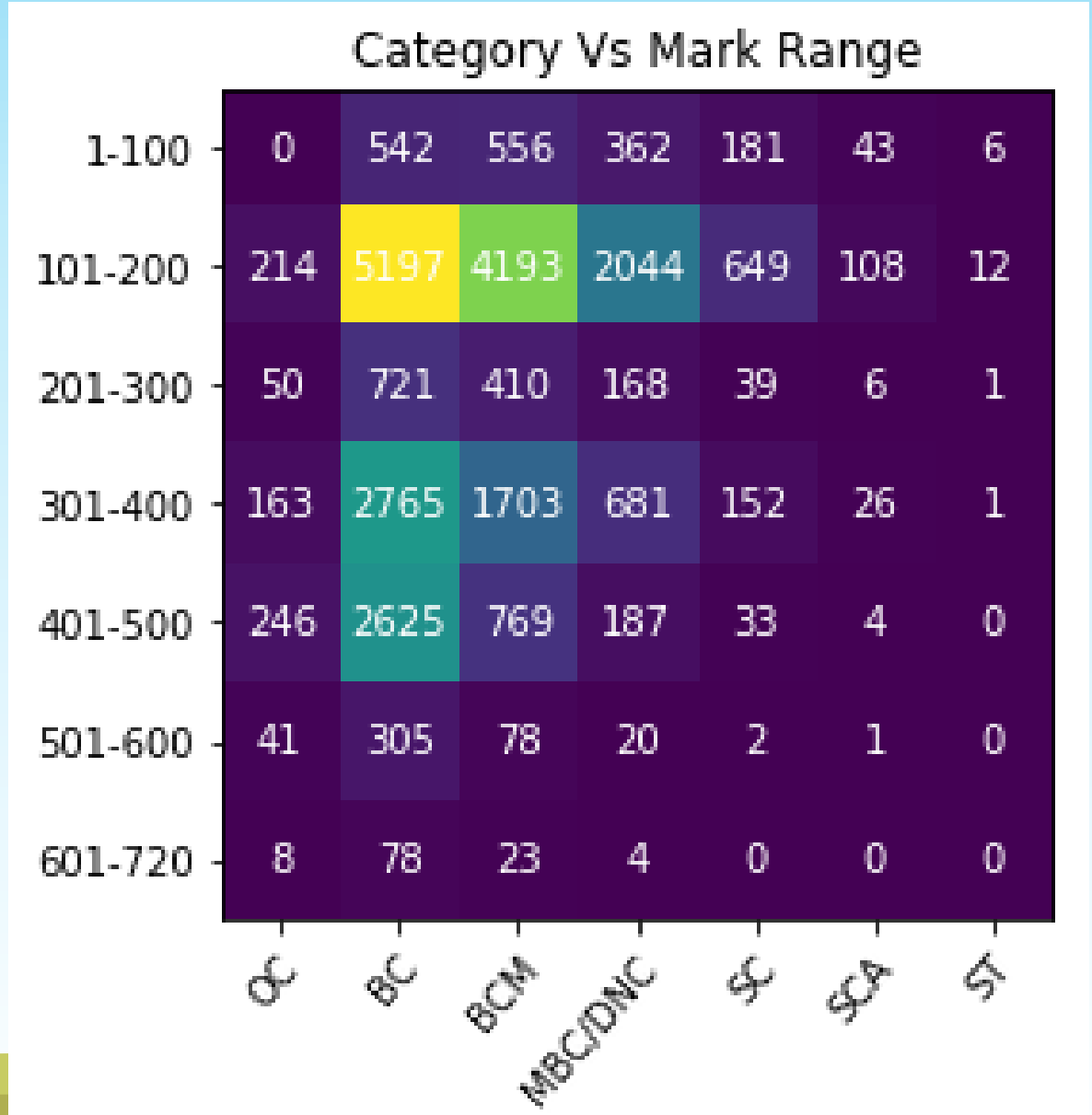

SCA-Repeating Marks (Range: 401 to 500) - Total: 2
Number of Candidates
Mark (min: 448,max: 476)

Category Vs Mark Range

```
categories=['OC','BC','BCM','MBC/DNC','SC','SCA','ST']
ranges=['1-100','101-200','201-300','301-400','401-500','501-600','601-720']
cateData=[]
for category in categories:
    markgroup=tneet[tneet.COMMUNITY==category].groupby(['MARK'])
    groups=markgroup.size()
    cdata=[]
    for rang in ranges:
        start,stop=rang.split("-")
        start=int(start)
        stop=int(stop)
        cdata.append(sum(groups.loc[start:stop].values))
    cateData.append(cdata)
cateData=np.array(cateData)
```

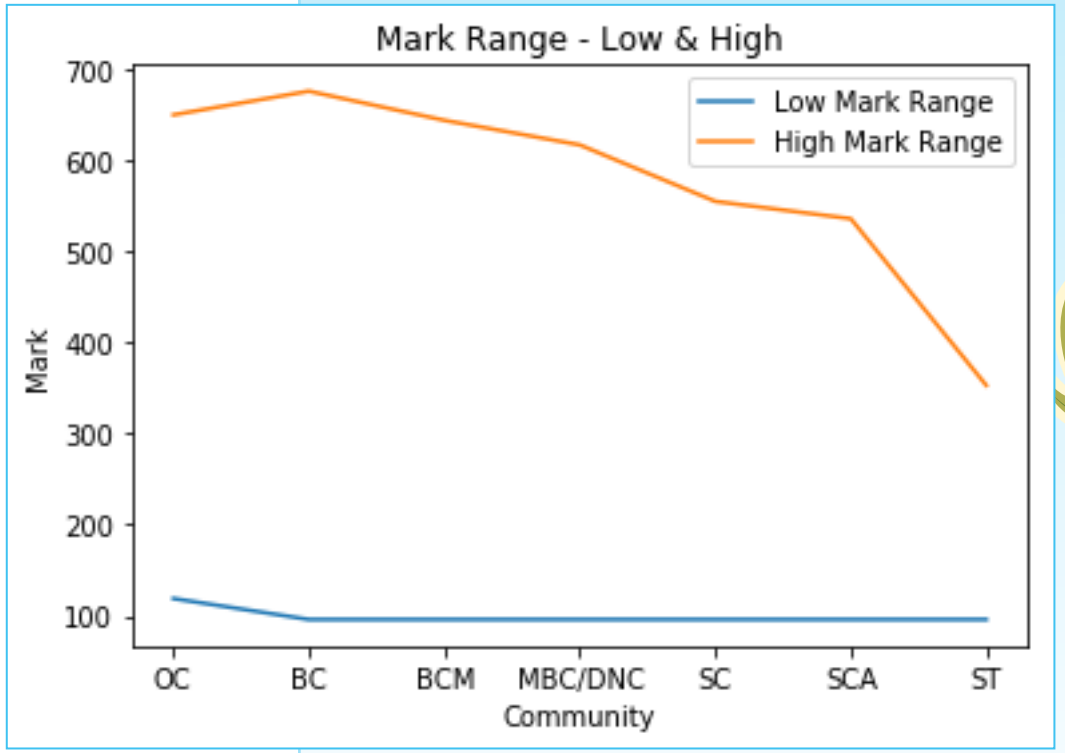
Plotting into heat maps

```
fig, ax = plt.subplots()
im = ax.imshow(cateData)
ax.set_xticks(np.arange(len(categories)))
ax.set_yticks(np.arange(len(ranges)))
ax.set_xticklabels(categories)
ax.set_yticklabels(ranges)
plt.setp(ax.get_xticklabels(), rotation=49, ha="right",
         rotation_mode="anchor")
for i in range(len(categories)):
    for j in range(len(ranges)):
        text = ax.text(j, i, cateData[i, j],ha="center", va="center", color="w")
ax.set_title("Category Vs Mark Range")
fig.tight_layout()
plt.show()
```


Category Vs Mark Range
1-100
101-200
201-300
301-400
401-500
501-600
601-720
0 542 556 362 181 43 6
214 5197 4193 2044 649 108 12
50 721 410 168 39 6 1
163 2765 1703 681 152 26 1
246 2625 769 187 33 4 0
41 305 78 20 2 1 0
8 78 23 4 0 0 0
OC
BC
BCM
MBC/DNC
SC
SCA
ST

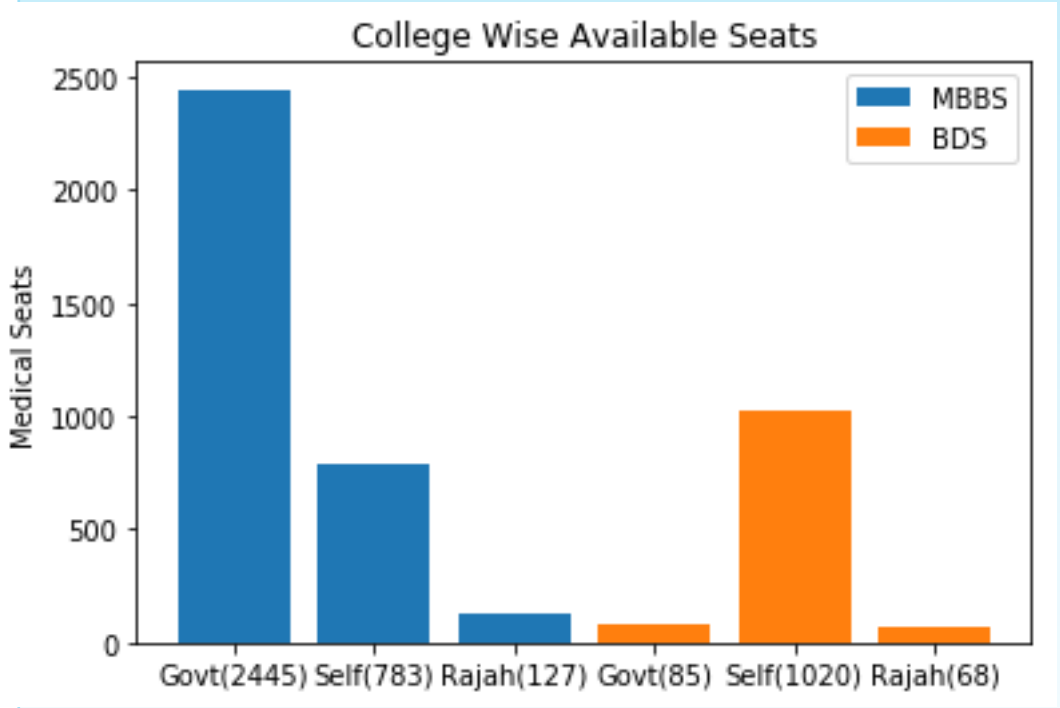
Mark Range – Low & High

```
categories=['OC','BC','BCM','MBC/DNC','SC','SCA','ST']
lowrange=[]
highrange=[]
for category in categories:
    markgroup=tneet[tneet.COMMUNITY==category]
    low=min(markgroup.MARK)
    high=max(markgroup.MARK)
    lowrange.append(low)
    highrange.append(high)

plt.plot(categories,lowrange)
plt.plot(categories,highrange)
plt.title('Mark Range – Low & High');
plt.xlabel('Community')
plt.ylabel('Mark')
plt.legend(['Low Mark Range','High Mark Range'])
plt.show()
```



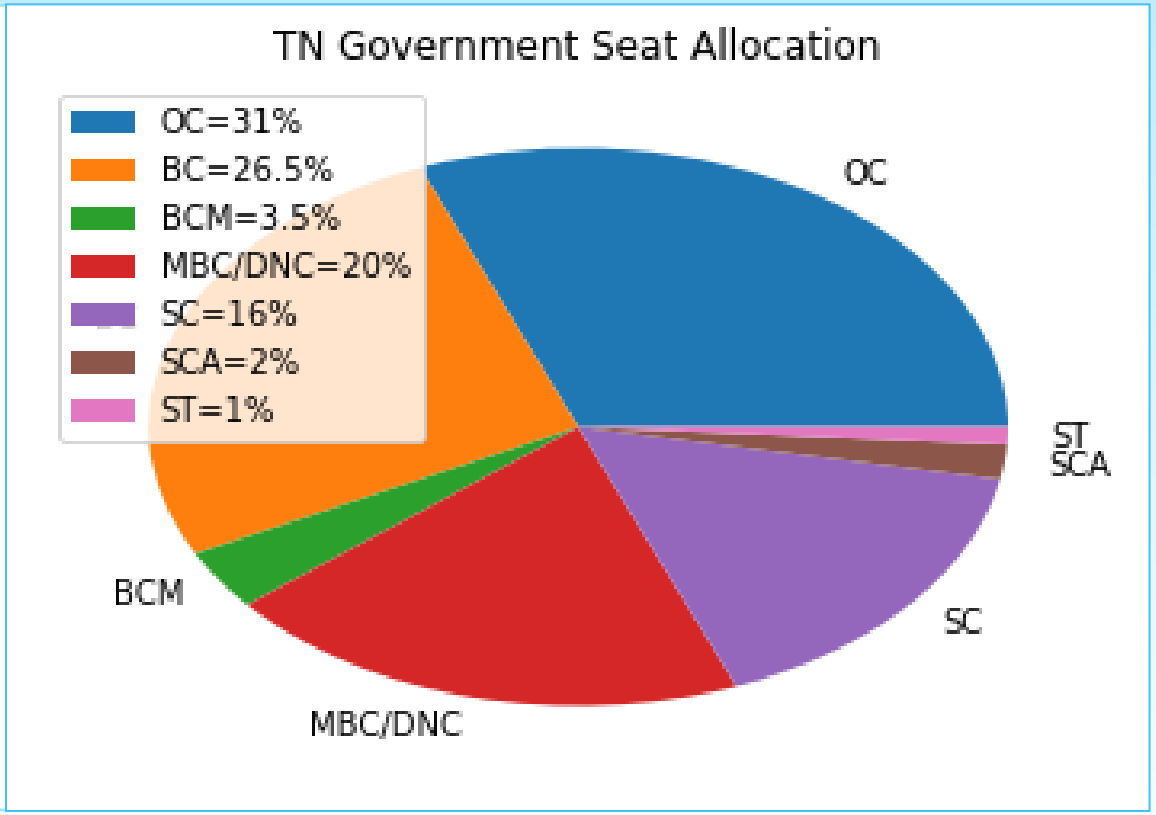
Available MBBS & BDS Seats

```
govtGQ=2445
selfGQ=783
rajGQ=127
bdsGQ=85
bdsSelfGQ=1020
rajbds=68
totalSeats=govtGQ+selfGQ+rajGQ+bdsGQ+bdsSelfGQ+rajbds
plt.bar(['Govt(2445)','Self(783)','Rajah(127)'],[govtGQ,selfGQ,rajGQ])
plt.bar(['Govt(85)','Self(1020)','Rajah(68)'],[bdsGQ,bdsSelfGQ,rajbds])
plt.title("College Wise Available Seats")
plt.ylabel('Medical Seats')
plt.legend(['MBBS','BDS'])
plt.show()
```

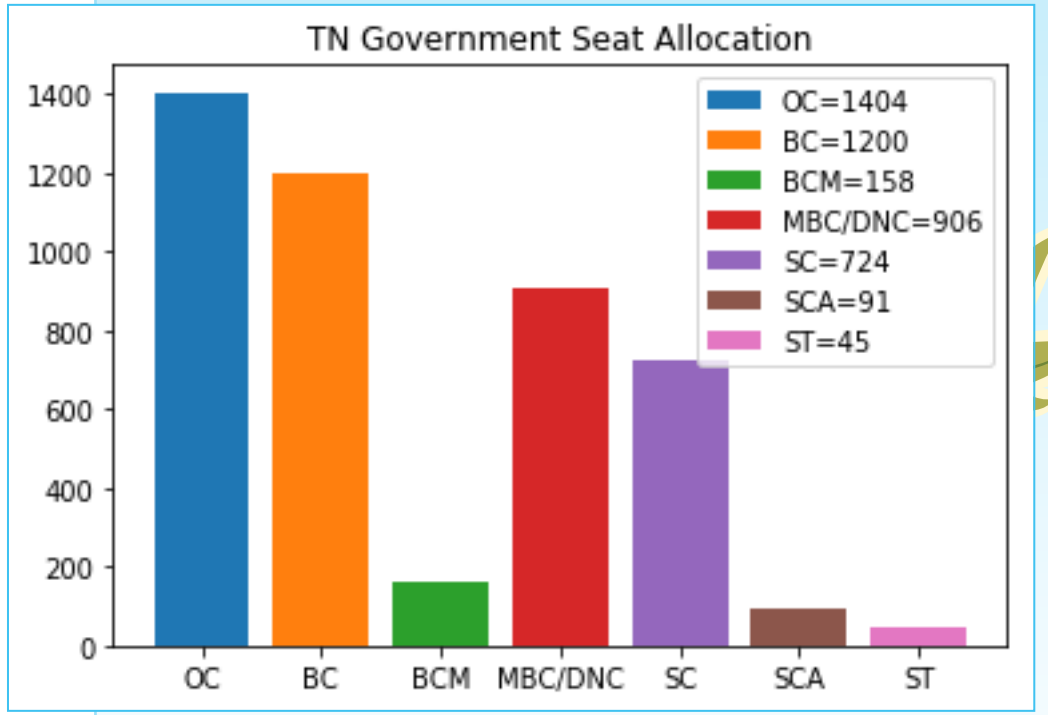


Seat Allocation By TN Government

```
allocation={'OC': 31,'BC': 26.5,'BCM': 3.5,
        'MBC/DNC': 20,'SC': 16,'SCA': 2,'ST': 1
}


plt.pie(allocation.values(),labels=allocation.keys())
plt.title("TN Government Seat Allocation")
plt.legend(["{!s}={!r}%".format(key,val) for (key,val)
in allocation.items()])
plt.show()
```



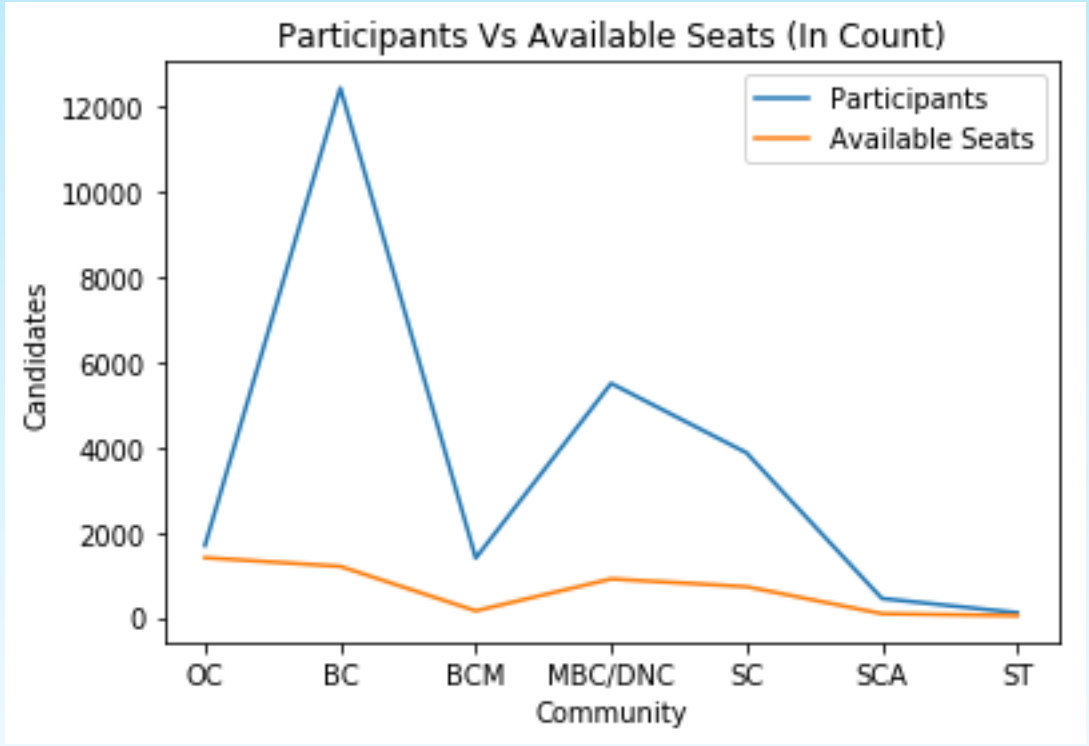
```
govtGQ=2445
selfGQ=783
rajGQ=127
bdsGQ=85
bdsSelfGQ=1020
rajbds=68
totalSeats=govtGQ+selfGQ+rajGQ+bdsGQ+bdsSelfGQ+rajbds
legends=[]
for (key,val) in allocation.items():
    seat=round((totalSeats*val)/100)
    legends.append("{!s}={!r}".format(key,seat))
    plt.bar([key],[seat])
plt.title("TN Government Seat Allocation")
plt.legend(legends)
plt.show()
```

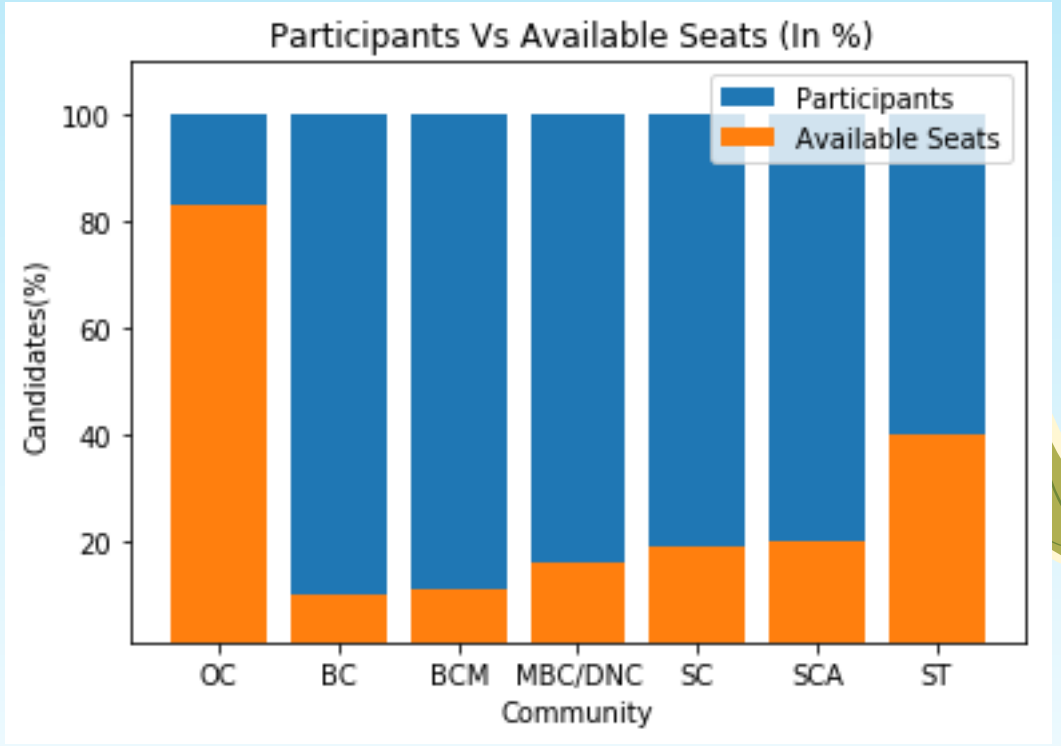


How many students get GQ Seat in each category

```
govtGQ=2445
selfGQ=783
rajGQ=127
bdsGQ=85
bdsSelfGQ=1020
rajbds=68
totalSeats=govtGQ+selfGQ+rajGQ+bdsGQ+bdsSelfGQ+rajbds
markgroup=tneet.groupby(['COMMUNITY'])
groups=markgroup.size()
for category in allocation.keys():
    seat=round((totalSeats*allocation[category])/100)
    totalSeat.append(100)
    cols.append(category)
    seatallot.append(round((seat/groups[category])*100))
    rseat.append(seat)
    parti.append(groups[category])
```

```
plt.plot(cols,parti)
plt.plot(cols,rseat)
plt.title("Participants Vs Available Seats (In Count)")
plt.legend(['Participants','Available Seats'])
plt.show()
plt.bar(cols,totalSeat)
plt.bar(cols,seatallot)
plt.ylim(1,110)
plt.title("Participants Vs Available Seats (In %) ")
plt.legend(['Participants','Available Seats'])
plt.xlabel('Community')
plt.ylabel('Candidates')
plt.show()
```

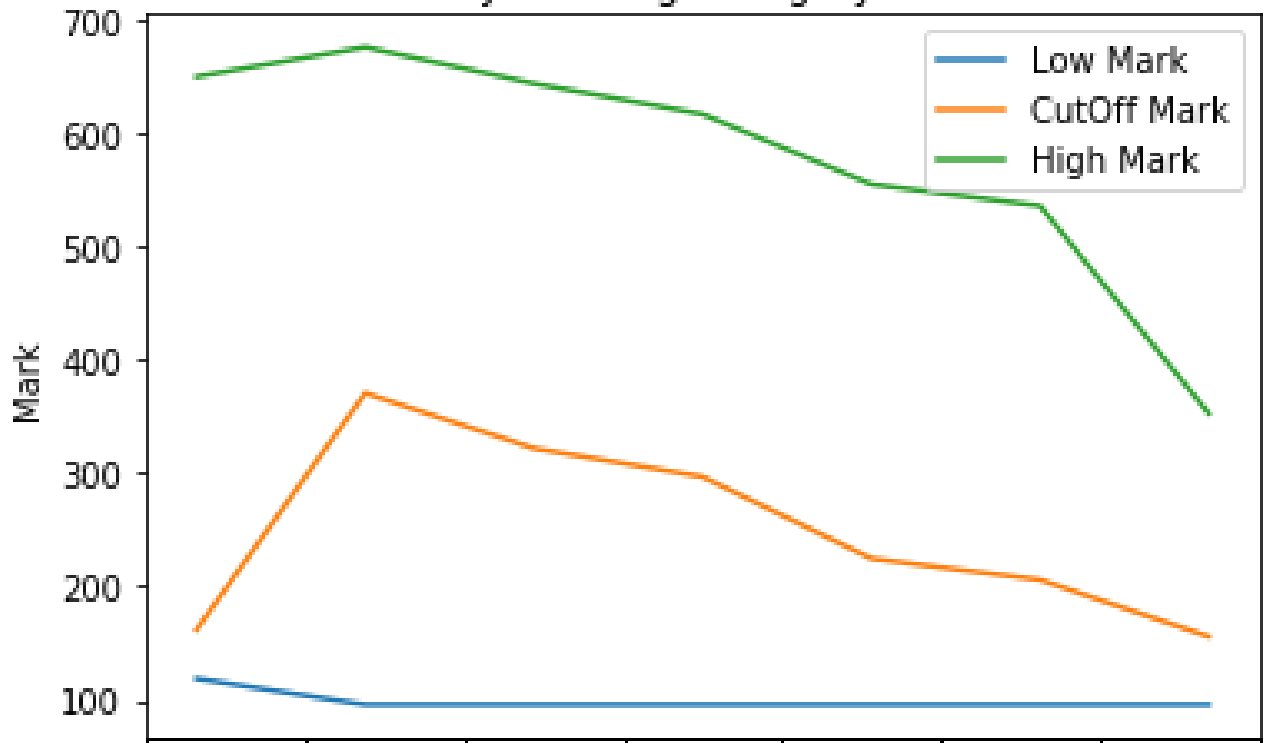

Participants Vs Available Seats (In Count)
Participants
Available Seats
12000
10000
8000
6000
4000
2000
0
Candidates
OC
BC
BCM
MBC/DNC
SC
SCA
ST
Community


Participants Vs Available Seats (In %)
Participants
Available Seats
100
80
60
40
20
Candidates(%)
OC
BC
BCM
MBC/DNC
SC
SCA
ST
Community

Cut Off Analysis

```
minmarks=[]
maxmarks=[]
lowmarks=[]
categories=allocation.keys()
for category in categories:
    seat=round((totalSeats*allocation[category])/100)
    allstudents=tneet[tneet.COMMUNITY==category]
    students=allstudents[:seat]
    minmarks.append(min(students.MARK))
    maxmarks.append(max(students.MARK))
    lowmarks.append(min(allstudents.MARK))
```

```
rows=['Low Mark','CutOff Mark','High Mark']
plt.plot(categories,lowmarks)
plt.plot(categories,minmarks)
plt.plot(categories,maxmarks)
plt.ylabel('Mark')
plt.title('Cut OFF Analysis Using Category Seat Allocation')
plt.legend(rows)
plt.table(cellText=[lowmarks,minmarks,maxmarks],
                    rowLabels=rows,
                    colLabels=tuple(categories),
                    loc='bottom')
plt.xticks([])
plt.show()
```



	OC	BC	BCM	MBC/DNC	SC	SCA	ST
Low Mark	119	96	96	96	96	96	96
CutOff Mark	162	371	322	297	225	206	156
High Mark	650	676	644	617	555	536	353

Why OC Cut OFF is Low?

Is there any wrong?

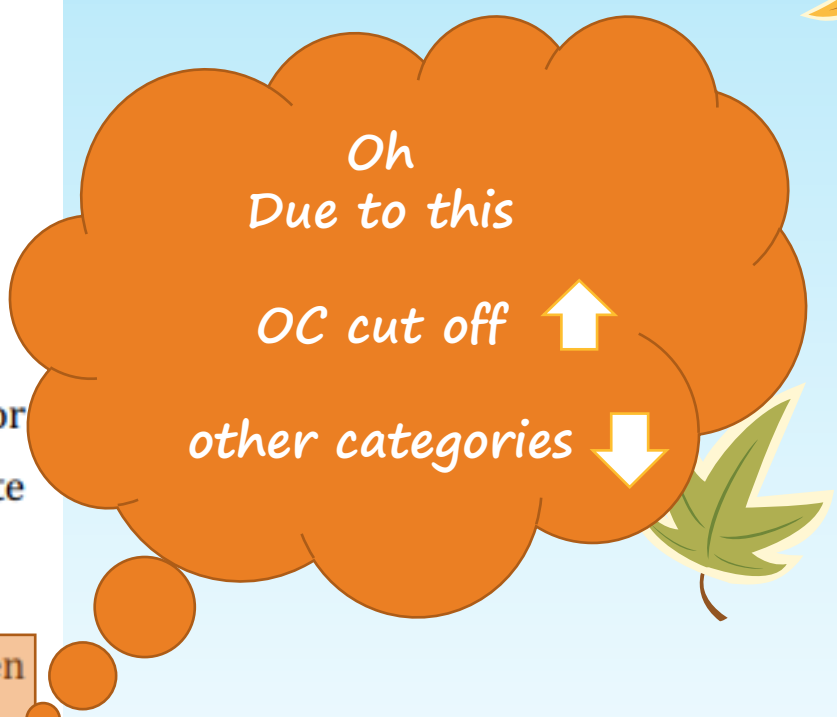
Government of Tamil Nadu as follows:-

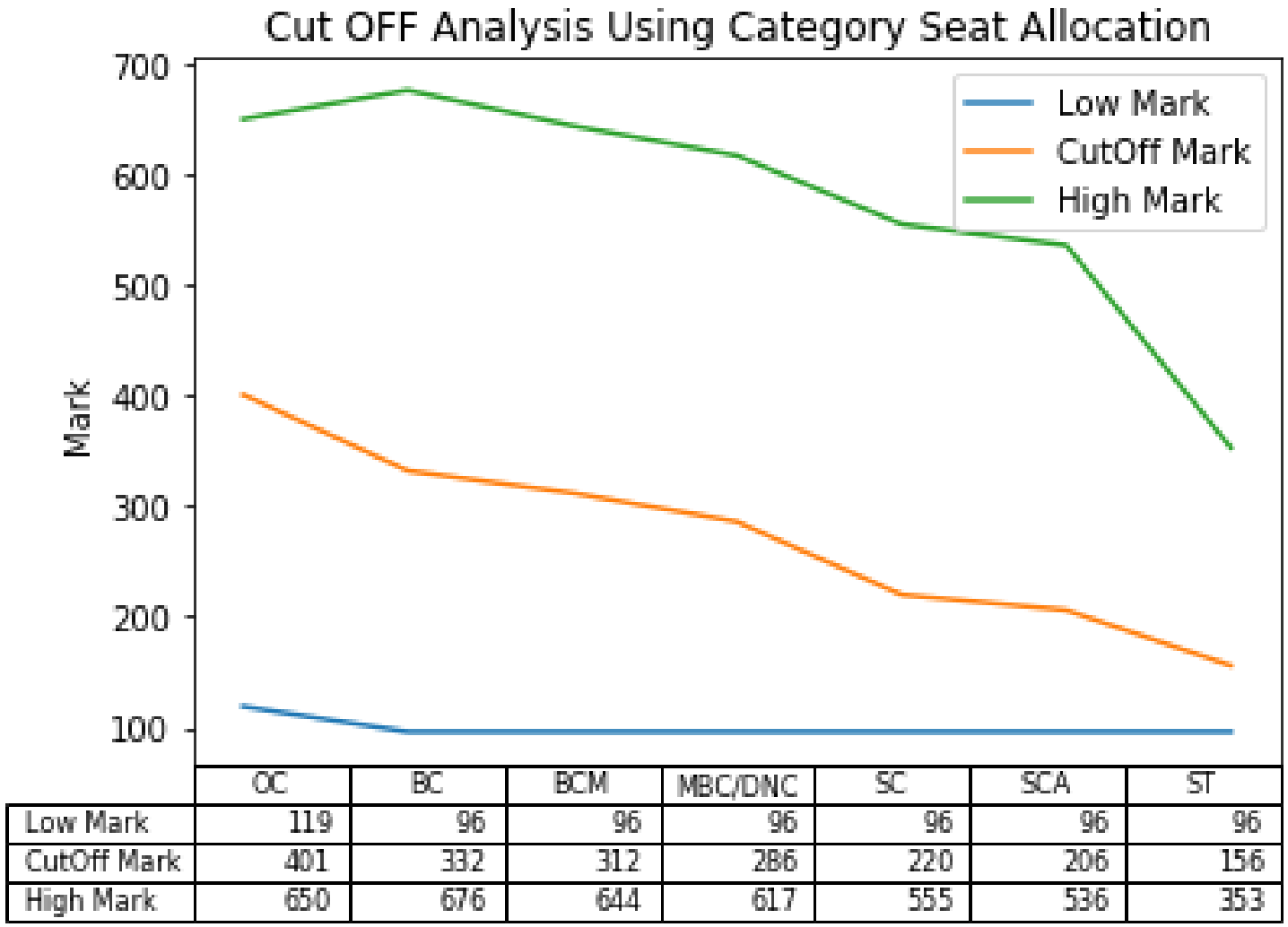
Open Competition	–	31 %
Backward Class	–	30 %
Most Back ward Class	–	20 %
Scheduled Caste	–	18 %
Scheduled Tribe	–	1 %

Within the 30% reservation for Backward Classes, 3.5% will be provided for Muslims and 16% of seats out of the 18% quota earmarked to Scheduled Caste shall be allocated to the Arunthathiyar Community.

BC, BCM, MBC / DNC, SC, SCA, ST candidates are eligible for Selection under Open Competition as per merit in addition to the reservation made for those categories.

Counselling Schedule will be available on the official websites only.



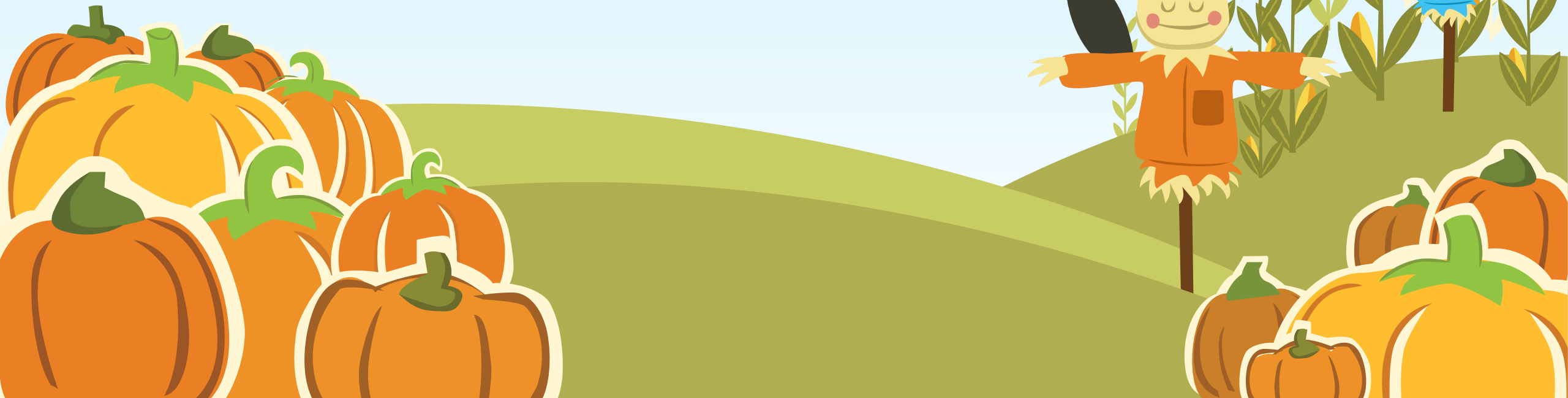


	OC	BC	BCM	MBC/DNC	SC	SCA	ST
Low Mark	119	96	96	96	96	96	96
CutOff Mark	401	332	312	286	220	206	156
High Mark	650	676	644	617	555	536	353

MBBS & BDS	
Category	Cut OFF
OC	401
BC	332
BCM	312
MBC/DNC	286
SC	220
SCA	206
ST	156

OC – Open Competition, Govt define some terms during counselling in this category

Any Query?



Cond .....

Ayyappan Murugesan

SmartSource Softwares Pvt Ltd

Phone: 7010774329

WhatsApp: 8870483557