Cleaning The DataSet:

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

In [2]:
```
import math
from collections import Counter
```

In [3]:
```
pd.options.mode.chained_assignment = None
```

In [4]:
```
#loading the student information dataset

data=pd.read_csv('./studentinfo.csv')
```

In [5]:
```
data.head()
```

Out[5]:

	new_id	batch	sex	dobday	dobmon	dobyear	fathermonin	sscins	sscboard	sscsu
0	1	20162	male	15.0	7.0	1999.0	60000/-	IDEAL SCHOOL AND COLLEGE	COMILLA	SCIENC
1	2	20022	m	NaN	NaN	NaN	NaN	NaN	NaN	Na
2	3	20120	male	1.0	8.0	1992.0	20000/-	TANGAIL TEC SCHOOL	BTEB	Na
3	4	20090	male	26.0	4.0	1990.0	30000/-	SENAPALLY HIGH SCHOOL	DHAKA	SCIENC
4	5	20111	male	1.0	2.0	1990.0	15000/-	JOYPURHAT R.B. GOVT. HIGH SCHOOL	RAJSHAHI	SCIENC

In [6]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2735 entries, 0 to 2734
Data columns (total 19 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   new_id       2735 non-null   int64  
 1   batch        2735 non-null   int64  
 2   sex          2711 non-null   object 
 3   dobday       2372 non-null   float64
 4   dobmon       2372 non-null   float64
 5   dobyear      2372 non-null   float64
 6   fathermonin  2294 non-null   object 
 7   sscins       2340 non-null   object 
 8   sscboard     2344 non-null   object 
 9   sscsub       2341 non-null   object 
 10  sscgrade     1983 non-null   object 
 11  ssctotal     2330 non-null   object 
 12  sscyear      2342 non-null   float64
 13  data         2340 non-null   object 
```

In [7]:
```
data.columns
```

Out[7]:
```
Index(['new_id', 'batch', 'sex', 'dobday', 'dobmon', 'dobyear', 'fathermonin',
       'sscins', 'sscboard', 'sscsub', 'sscgrade', 'ssctotal', 'sscyear',
       'data', 'hscboard', 'hscsub', 'hscgrade', 'hsctotal', 'hscyear'],
      dtype='object')
```

In [8]:
```
#Check size of data
data.shape
```

Out[8]:
```
(2735, 19)
```

In [9]:
```
#Describe numeric F]features
data.describe
```

Out[9]:
```
<bound method NDFrame.describe of       new_id  batch     sex  dobday  dobmon
dobyear fathermonin  \
0          1  20162    male    15.0     7.0   1999.0     60000/-
1          2  20022       m     NaN     NaN      NaN         NaN
2          3  20120    male     1.0     8.0   1992.0     20000/-
3          4  20090    male    26.0     4.0   1990.0     30000/-
4          5  20111    male     1.0     2.0   1990.0     15000/-
...      ...    ...     ...     ...     ...      ...         ...
2730    2731  20040     NaN     NaN     NaN      NaN         NaN
2731    2732  20120    male    10.0     1.0   1992.0       35000
2732    2733  20131    male    26.0     1.0   1989.0       60000
2733    2734  20070       m     NaN     NaN      NaN         NaN
2734    2735  20132  female    24.0    12.0   1995.0     30000/-

                                sscins  sscboard   sscsub sscgrade ssctotal  \
0            IDEAL SCHOOL AND COLLEGE   COMILLA  SCIENCE        A     4.56
1                                 NaN       NaN      NaN      NaN      NaN
2                TANGAIL TEC SCHOOL      BTEB      NaN        A     4.14
3              SENAPALLY HIGH SCHOOL     DHAKA  SCIENCE      NaN      4.5
4     JOYPURHAT R.B. GOVT. HIGH SCHOOL  RAJSHAHI  SCIENCE        A     4.25
...                                ...       ...      ...      ...      ...
2730                              NaN       NaN      NaN      NaN      NaN
2731          ANKUR SCHOOL & COLLEGE     DHAKA  SCIENCE       A+        5
2732         ST. GREGORYS HIGH SCHOOL     DHAKA  SCIENCE        A     4.38
2733                              NaN       NaN      NaN      NaN      NaN
2734        MIRPUR BANGLA HIGH SCHOOL     DHAKA  SCIENCE        A     4.38

      sscyear                                data hscboard   hscsub hscgrade  \
0      2014.0          IDEAL SCHOOL AND COLLEGE       NaN  SCIENCE      APP
1         NaN                                NaN       NaN      NaN      NaN
2      2009.0              TANGAIL TEC SCHOOL      BTEB      NaN        A
3      2006.0            SENAPALLY HIGH SCHOOL     DHAKA  SCIENCE      NaN
4      2006.0  JOYPURHAT R.B. GOVT. HIGH SCHOOL      BTEB  SCIENCE        B
...       ...                                ...       ...      ...      ...
2730      NaN                                NaN       NaN      NaN      NaN
2731   2008.0          ANKUR SCHOOL & COLLEGE     DHAKA  SCIENCE        A
2732   2007.0         ST. GREGORYS HIGH SCHOOL     DHAKA  SCIENCE        C
2733      NaN                                NaN       NaN      NaN      NaN
2734   2008.0        MIRPUR BANGLA HIGH SCHOOL     DHAKA  SCIENCE        A

     hsctotal  hscyear
0         NaN   2016.0
1         NaN      NaN
2        4.28   2011.0
3         3.8   2008.0
4        3.22   2010.0
...       ...      ...
2730      NaN      NaN
2731        4   2010.0
2732      2.9   2009.0
2733      NaN      NaN
2734      4.2   2010.0

[2735 rows x 19 columns]>
```

In [10]:
```
#Check if there is null values
data.isnull().sum()
```

Out[10]:
```
new_id           0
batch            0
sex             24
dobday         363
dobmon         363
dobyear        363
fathermonin    441
sscins         395
sscboard       391
sscsub         394
sscgrade       752
ssctotal       405
sscyear        393
data           395
hscboard       394
hscsub         399
hscgrade       784
hsctotal       410
hscyear        393
dtype: int64
```

In [11]:
```
#Loading the student grade dataset

grade_data=pd.read_csv('./csestudentsgrades.csv')
```

In [12]:
```
grade_data.head()
```

Out[12]:

	new_id	offered_id	semester_id	course_code	grade	gp	credits
0	267.0	NaN	1	CSE111	F	0.0	3
1	1273.0	NaN	1	CSE123	F	0.0	3
2	1273.0	NaN	1	CSE231	F	0.0	3
3	1273.0	NaN	1	CSE312	F	0.0	3
4	2032.0	NaN	1	MATH124	F	0.0	3

In [13]:
```
grade_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 65235 entries, 0 to 65234
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   new_id       64723 non-null  float64
 1   offered_id   53756 non-null  float64
 2   semester_id  65235 non-null  int64  
 3   course_code  65235 non-null  object 
 4   grade        65235 non-null  object 
 5   gp           65235 non-null  float64
 6   credits      65235 non-null  int64  
dtypes: float64(3), int64(2), object(2)
memory usage: 3.5+ MB
```

In [14]:
```
grade_data.columns
```

Out[14]:
```
Index(['new_id', 'offered_id', 'semester_id', 'course_code', 'grade', 'gp',
       'credits'],
      dtype='object')
```

In [15]:
```
#Check size of data
grade_data.shape
```

Out[15]:
```
(65235, 7)
```

In [16]:
```
#Describe numeric features
grade_data.describe
```

Out[16]:
```
<bound method NDFrame.describe of        new_id  offered_id  semester_id course
_code grade    gp  credits
0       267.0         NaN            1      CSE111     F  0.00        3
1      1273.0         NaN            1      CSE123     F  0.00        3
2      1273.0         NaN            1      CSE231     F  0.00        3
3      1273.0         NaN            1      CSE312     F  0.00        3
4      2032.0         NaN            1     MATH124     F  0.00        3
...       ...         ...          ...         ...   ...   ...      ...
65230   458.0     44247.0           45     CSE2026    B+  3.25        1
65231  2001.0     44247.0           45     CSE2026    B+  3.25        1
65232   536.0     44247.0           45     CSE2026    B+  3.25        1
65233  2560.0     44247.0           45     CSE2026    B+  3.25        1
65234  2531.0     44247.0           45     CSE2026     D  2.00        1

[65235 rows x 7 columns]>
```

In [17]:
```
#Check if there is null values
grade_data.isnull().sum()
```

Out[17]:
```
new_id           512
offered_id     11479
semester_id        0
course_code        0
grade              0
gp                 0
credits            0
dtype: int64
```

In [18]:
```
#Adding columns in our dataset

data['dif_ssc_hsc']=0
data['dif_hsc_uni']=0
data['drop_out']="regular"
```

In [19]:
```
for i in range(len(data)):

    print(i)

    total_conv=data['hsctotal'][i]
    try:
        total_conv=float(total_conv)
    except ValueError:
        total_conv=float(data['hscgrade'][i])
    data['hsctotal'][i]=total_conv

    total_conv=data['ssctotal'][i]
    try:
        total_conv=float(total_conv)
    except ValueError:
        total_conv=float(data['sscgrade'][i])
    data['ssctotal'][i]=total_conv


    tmp=(data['hscyear'][i]-data['sscyear'][i])-2
    if str(tmp)!= 'nan':

        data['dif_ssc_hsc'][i]=int(tmp)
        print(data['new_id'][i],tmp)

    tmp=(data['batch'][i]/10)-data['hscyear'][i]
    if str(tmp)!='nan':
        data['dif_hsc_uni'][i]=int(tmp)
```

```
0
1 0.0
1
2
3 0.0
3
4 0.0
4
5 2.0
5
6 0.0
6
7 0.0
7
8
9 1.0
9
10 0.0
10
```

In [20]:
```
#droping the columns

data=data.drop(['sscgrade','hscgrade','dobday','dobyear','dobmon','fathermonin','
data=data.dropna()
data=data.reset_index(drop=True)
```

In [21]:
```
#Creating function for the gpa calculation

def cgpa_calculation(grade_data1,all_subjects):

    #new grade points
    new_grade_point=sum(grade_data1['gp']*grade_data1['credits'])

    #new subject
    sub=set(grade_data1['course_code'])
    new_sub=list(sub.difference(all_subjects))

    mask=grade_data1['course_code'].isin(new_sub)
    #new credit earns
    new_credit_earn=sum(grade_data1[mask]['credits'])

    fail = len(grade_data1['course_code'][grade_data1['grade']=='F'])
    return new_sub,new_grade_point,new_credit_earn,fail
```

In [22]:
```
cgpa_calculation(grade_data,data)
```

Out[22]:
```
(['ENG1235',
  'ETE3032',
  'STA1223',
  'CSE322',
  'ENG1002',
  'CSE3025',
  'ENG1021',
  'ENG125',
  'CSE111',
  'CSE2013',
  'CSE4028',
  'MATH215',
  'MATH135',
  'CSE2032',
  'CHEM1031',
  'CSE433',
  'CSE123',
  'CSE1013',
  'CSE1215',
```

In [23]:
```
for i in range(8):
    tmp='semester'+str(i+1)+'_cgpa'
    data[tmp]=0.0
    tmp='semester'+str(i+1)+'_fail'
    data[tmp]=0
```

In [24]:
```
delete_row=[]
```

In [25]:
```
for i in range(len(data['new_id'])):
    print('-',i)
    t = time.time()
    ID=data['new_id'][i]
    search_id=grade_data['new_id']==ID
    if (search_id.any() == False):
        delete_row.append(i)
        continue

    id_info= grade_data[search_id]
    Batch_list= list(set(id_info['semester_id']))
    batch_len=len(Batch_list)

    if batch_len>8:
        delete_row.append(i)
        continue
    Batch_list.sort()
    #print (ID)

    id_subject=set()
    grade_point=0.0
    credit_earn=0.0
    total_fail=0
    mark=0

    for j in range(batch_len):
        batch=Batch_list[j]
        t = time.time()
        sub,gp,cre,fail= cgpa_calculation(id_info[id_info['semester_id']==batch],
        id_subject.update(sub)
        grade_point+=gp
        credit_earn+=cre
        total_fail+=fail

        tmp='semester'+str(j+1)+'_cgpa'
        if grade_point <2 :
            mark=0.0
        else :
            mark=grade_point/credit_earn

        data[tmp][i]=mark

        tmp1='semester'+str(j+1)+'_fail'
        data[tmp1][i]=fail

    if batch_len<8:
        for j in range(batch_len,8):
            tmp='semester'+str(j+1)+'_cgpa'
            data[tmp][i]=mark


    if (Batch_list[batch_len-1]<41 and  credit_earn<130) or (batch_len<3 and tota
        data['drop_out'][i]="dropOut"

print ("%.3f" % (time.time()-t))
```

- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17

In [26]:
```
data.drop(data.index[delete_row],inplace=True)
```

In [27]:
```
data.to_csv('cleanData.csv',sep=',',index=False)
```

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

Out[28]:

	new_id	batch	ssctotal	sscyear	hsctotal	hscyear	dif_ssc_hsc	dif_hsc_uni	drop_out	semest
3	6	20160	4.63	2013.0	3.83	2015.0	0	1	regular	
4	7	20162	5	2009.0	5	2011.0	0	5	regular	
6	10	20161	5	2012.0	4.1	2014.0	0	2	dropOut	
8	12	20170	4.75	2013.0	4	2015.0	0	2	regular	
9	13	20120	3.94	2009.0	3.9	2011.0	0	1	dropOut	

5 rows × 25 columns

The Data Has Been Cleaned Now...

In [29]:
```
import numpy as np
import matplotlib.pyplot as plt
from collections import Counter

from sklearn.metrics import classification_report
from sklearn.model_selection import learning_curve, GridSearchCV
from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier

from imblearn.over_sampling import SMOTE
from sklearn.model_selection import learning_curve
```

In [30]:
```
data =pd.read_csv('./cleanData.csv')
backUp=data.copy(deep=True)
```

In [31]:
```
data.head()
```

Out[31]:

	new_id	batch	ssctotal	sscyear	hsctotal	hscyear	dif_ssc_hsc	dif_hsc_uni	drop_out	semest
0	6	20160	4.63	2013.0	3.83	2015.0	0	1	regular	
1	7	20162	5.00	2009.0	5.00	2011.0	0	5	regular	
2	10	20161	5.00	2012.0	4.10	2014.0	0	2	dropOut	
3	12	20170	4.75	2013.0	4.00	2015.0	0	2	regular	
4	13	20120	3.94	2009.0	3.90	2011.0	0	1	dropOut	

5 rows × 25 columns

In [32]:
```
data.info
```

Out[32]:
```
<bound method DataFrame.info of       new_id  batch  ssctotal  sscyear  hsctotal  hscyear  dif_ssc_hsc  \
0          6  20160      4.63   2013.0      3.83   2015.0            0
1          7  20162      5.00   2009.0      5.00   2011.0            0
2         10  20161      5.00   2012.0      4.10   2014.0            0
3         12  20170      4.75   2013.0      4.00   2015.0            0
4         13  20120      3.94   2009.0      3.90   2011.0            0
...      ...    ...       ...      ...       ...      ...          ...
1397    2724  20160      4.25   2013.0      3.83   2015.0            0
1398    2726  20150      4.25   2012.0      3.80   2014.0            0
1399    2729  20170      3.69   2012.0      3.90   2015.0            1
1400    2732  20120      5.00   2008.0      4.00   2010.0            0
1401    2733  20131      4.38   2007.0      2.90   2009.0            0

      dif_hsc_uni drop_out  semester1_cgpa  ...  semester4_cgpa  \
0               1  regular        2.416667  ...        2.297872
1               5  regular        3.166667  ...        3.363636
2               2  dropOut        2.250000  ...        1.350000
3               2  regular        3.250000  ...        3.250000
4               1  dropOut        0.000000  ...        0.000000
...           ...      ...             ...  ...             ...
1397            1  regular        2.750000  ...        2.961957
1398            1  regular        2.583333  ...        2.588710
1399            2  regular        0.666667  ...        0.666667
1400            2  dropOut        1.666667  ...        1.666667
1401            4  dropOut        0.000000  ...        0.000000

      semester4_fail  semester5_cgpa  semester5_fail  semester6_cgpa  \
0                  2        2.297872               0        2.297872
1                  0        3.363636               0        3.363636
2                  0        1.350000               0        1.350000
3                  0        3.250000               0        3.250000
4                  0        0.000000               0        0.000000
...              ...             ...             ...             ...
1397               0        2.961957               0        2.961957
1398               0        2.588710               0        2.588710
1399               0        0.666667               0        0.666667
1400               0        1.666667               0        1.666667
1401               0        0.000000               0        0.000000

      semester6_fail  semester7_cgpa  semester7_fail  semester8_cgpa  \
0                  0        2.297872               0        2.297872
1                  0        3.363636               0        3.363636
2                  0        1.350000               0        1.350000
3                  0        3.250000               0        3.250000
4                  0        0.000000               0        0.000000
...              ...             ...             ...             ...
1397               0        2.961957               0        2.961957
1398               0        2.588710               0        2.588710
1399               0        0.666667               0        0.666667
1400               0        1.666667               0        1.666667
1401               0        0.000000               0        0.000000

      semester8_fail
0                  0
```

```
1                0
2                0
3                0
4                0
...            ...
1397             0
1398             0
1399             0
1400             0
1401             0

[1402 rows x 25 columns]>
```

In [33]:
```
data.columns
```

Out[33]:
```
Index(['new_id', 'batch', 'ssctotal', 'sscyear', 'hsctotal', 'hscyear',
       'dif_ssc_hsc', 'dif_hsc_uni', 'drop_out', 'semester1_cgpa',
       'semester1_fail', 'semester2_cgpa', 'semester2_fail', 'semester3_cgpa',
       'semester3_fail', 'semester4_cgpa', 'semester4_fail', 'semester5_cgpa',
       'semester5_fail', 'semester6_cgpa', 'semester6_fail', 'semester7_cgpa',
       'semester7_fail', 'semester8_cgpa', 'semester8_fail'],
      dtype='object')
```

In [34]:
```
#Check size of data
data.shape
```

Out[34]:
```
(1402, 25)
```

In [35]:
```
#Describe numeric F]features
data.describe
```

Out[35]:
```
<bound method NDFrame.describe of       new_id  batch  ssctotal  sscyear  hscto
tal  hscyear  dif_ssc_hsc  \
0          6  20160      4.63   2013.0      3.83   2015.0           0
1          7  20162      5.00   2009.0      5.00   2011.0           0
2         10  20161      5.00   2012.0      4.10   2014.0           0
3         12  20170      4.75   2013.0      4.00   2015.0           0
4         13  20120      3.94   2009.0      3.90   2011.0           0
...      ...    ...       ...      ...       ...      ...         ...
1397    2724  20160      4.25   2013.0      3.83   2015.0           0
1398    2726  20150      4.25   2012.0      3.80   2014.0           0
1399    2729  20170      3.69   2012.0      3.90   2015.0           1
1400    2732  20120      5.00   2008.0      4.00   2010.0           0
1401    2733  20131      4.38   2007.0      2.90   2009.0           0

      dif_hsc_uni drop_out  semester1_cgpa  ...  semester4_cgpa  \
0               1  regular        2.416667  ...        2.297872
1               5  regular        3.166667  ...        3.363636
2               2  dropOut        2.250000  ...        1.350000
3               2  regular        3.250000  ...        3.250000
4               1  dropOut        0.000000  ...        0.000000
...           ...      ...             ...  ...             ...
1397            1  regular        2.750000  ...        2.961957
1398            1  regular        2.583333  ...        2.588710
1399            2  regular        0.666667  ...        0.666667
1400            2  dropOut        1.666667  ...        1.666667
1401            4  dropOut        0.000000  ...        0.000000

      semester4_fail  semester5_cgpa  semester5_fail  semester6_cgpa  \
0                  2        2.297872               0        2.297872
1                  0        3.363636               0        3.363636
2                  0        1.350000               0        1.350000
3                  0        3.250000               0        3.250000
4                  0        0.000000               0        0.000000
...              ...             ...             ...             ...
1397               0        2.961957               0        2.961957
1398               0        2.588710               0        2.588710
1399               0        0.666667               0        0.666667
1400               0        1.666667               0        1.666667
1401               0        0.000000               0        0.000000

      semester6_fail  semester7_cgpa  semester7_fail  semester8_cgpa  \
0                  0        2.297872               0        2.297872
1                  0        3.363636               0        3.363636
2                  0        1.350000               0        1.350000
3                  0        3.250000               0        3.250000
4                  0        0.000000               0        0.000000
...              ...             ...             ...             ...
1397               0        2.961957               0        2.961957
1398               0        2.588710               0        2.588710
1399               0        0.666667               0        0.666667
1400               0        1.666667               0        1.666667
1401               0        0.000000               0        0.000000

      semester8_fail
```

```
0                0
1                0
2                0
3                0
4                0
...            ...
1397             0
1398             0
1399             0
1400             0
1401             0

[1402 rows x 25 columns]>
```

In [36]:
```
#Check if there is null values
data.isnull().sum()
```

Out[36]:
```
new_id            0
batch             0
ssctotal          0
sscyear           0
hsctotal          0
hscyear           0
dif_ssc_hsc       0
dif_hsc_uni       0
drop_out          0
semester1_cgpa    0
semester1_fail    0
semester2_cgpa    0
semester2_fail    0
semester3_cgpa    0
semester3_fail    0
semester4_cgpa    0
semester4_fail    0
semester5_cgpa    0
semester5_fail    0
semester6_cgpa    0
semester6_fail    0
semester7_cgpa    0
semester7_fail    0
semester8_cgpa    0
semester8_fail    0
dtype: int64
```

In [37]:
```
Class=data['drop_out'].copy(deep=True)
#data['sex'] = (data['sex']=='male')*1
```

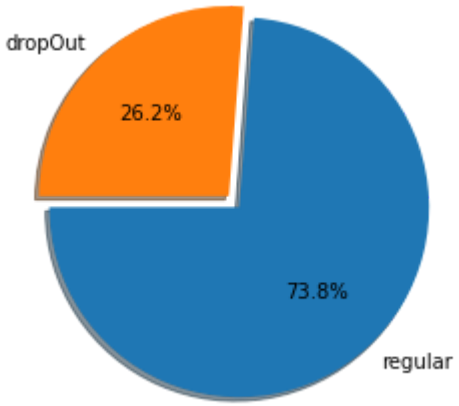
In [38]:
```
data=data.drop(['new_id', 'batch','sscyear','hscyear','drop_out'],axis=1)
```

In [39]:
```
data.shape
```

Out[39]:
```
(1402, 20)
```

In [40]:
```
x=Counter(Class)
print(x)
lab=x.keys()
val=x.values()
exp=[0,0.08]

fig1, ax1 = plt.subplots()
ax1.pie(val, explode=exp, labels=lab, autopct='%1.1f%%',shadow=True, startangle=1
ax1.axis('equal')
plt.show()
```

```
Counter({'regular': 1034, 'dropOut': 368})
```



From the pie chart it's clear that it's a imbalanced dataset with 4:1 ratio . So we want to do resampling the data set.As it's a small dataset so we want to do over-sampling the minority class by using SMOTE or the Synthetic Minority Over-sampling Technique.

In [41]:
```
sm = SMOTE (sampling_strategy='auto', random_state=1, k_neighbors=8)
data_sm,Class_sm = sm.fit_sample(data,Class)
print (Class_sm)
```

```
0       regular
1       regular
2       dropOut
3       regular
4       dropOut
         ...
2063    dropOut
2064    dropOut
2065    dropOut
2066    dropOut
2067    dropOut
Name: drop_out, Length: 2068, dtype: object
```

In [42]:
```
X_train, X_test, y_train, y_test = train_test_split(data_sm,Class_sm, test_size=0
```

In [43]:
```
def Learning_curve(model,name):

    train_sizes, train_scores, test_scores = learning_curve(model, X_train, y_tra
                        n_jobs=-1, cv=5, train_sizes=np.linspace(.1, 1.0, 5), ver
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)

    plt.figure()
    plt.title(name)

    plt.xlabel("Training examples")
    plt.ylabel("Score")


    plt.gca().invert_yaxis()
    plt.grid()

    plt.fill_between(train_sizes, train_scores_mean - train_scores_std, train_sc
    plt.fill_between(train_sizes, test_scores_mean - test_scores_std, test_scores

    plt.plot(train_sizes, train_scores_mean, 'o-', color="r", label="Training sc
    plt.plot(train_sizes, test_scores_mean, 'o-', color="g", label="Cross-validat
    plt.legend(loc='lower center')

    plt.ylim(-.1,1.1)
    plt.show()
```

Here our y-axis is 'score', not 'error', so the higher the score, the better the performance of the model.

High Bias

Training score (red line) decreases and plateau

- Indicates underfitting
- High bias

Cross-validation score (green line) stagnating throughout

- Unable to learn from data

Low scores (high errors)

High Variance

Training score (red line) is at its maximum regardless of training examples

- Indicates overfitting

Huge gap between cross-validation score and training score

- Indicates high variance scenario(overfitting)

In [44]:
```
data =pd.read_csv('./cleanData.csv')
```

In [45]:
```
data.head(5)
```

Out[45]:

	new_id	batch	ssctotal	sscyear	hsctotal	hscyear	dif_ssc_hsc	dif_hsc_uni	drop_out	semest
0	6	20160	4.63	2013.0	3.83	2015.0	0	1	regular	
1	7	20162	5.00	2009.0	5.00	2011.0	0	5	regular	
2	10	20161	5.00	2012.0	4.10	2014.0	0	2	dropOut	
3	12	20170	4.75	2013.0	4.00	2015.0	0	2	regular	
4	13	20120	3.94	2009.0	3.90	2011.0	0	1	dropOut	

5 rows × 25 columns

In [46]:
```
#Split Data Train/ Validate/Test
#Target = 'Dropout'
target = 'drop_out'
features = data.columns.drop('drop_out')

X = data[features]
y = data[target]

X.shape,y.shape
```

Out[46]:
```
((1402, 24), (1402,))
```

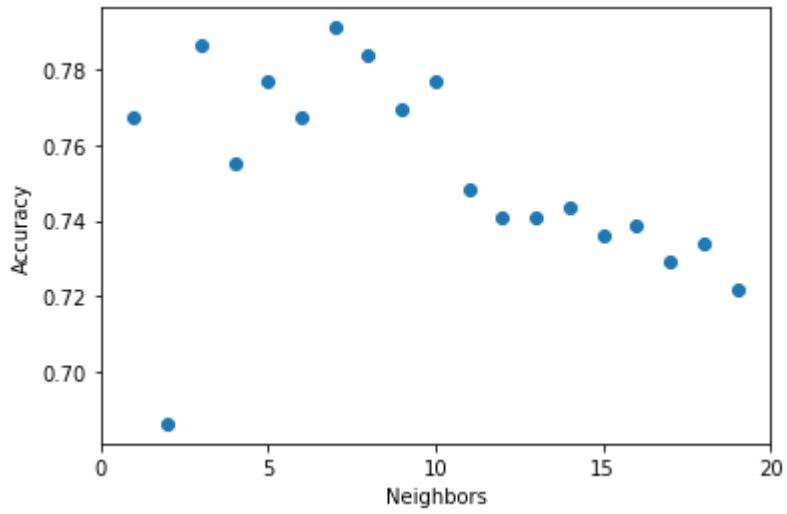
In [47]:
```
# Do train/validate/test 3-way split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.70, test_s
```

Models:

1: KN-Neighbors

In [48]:
```
k_range = range(1, 20)
scores = []
best_score=0
for k in k_range:
    knn_model = KNeighborsClassifier(n_neighbors=k,algorithm='auto',n_jobs=-1)
    tmp_model=knn_model.fit(X_train, y_train)
    tmp=knn_model.score(X_test, y_test)
    if best_score<tmp:
        best_score=tmp
        best_model=tmp_model
    scores.append(tmp)
plt.figure()
plt.xlabel('Neighbors')
plt.ylabel('Accuracy')
plt.scatter(k_range, scores)
plt.xticks([0,5,10,15,20])
```

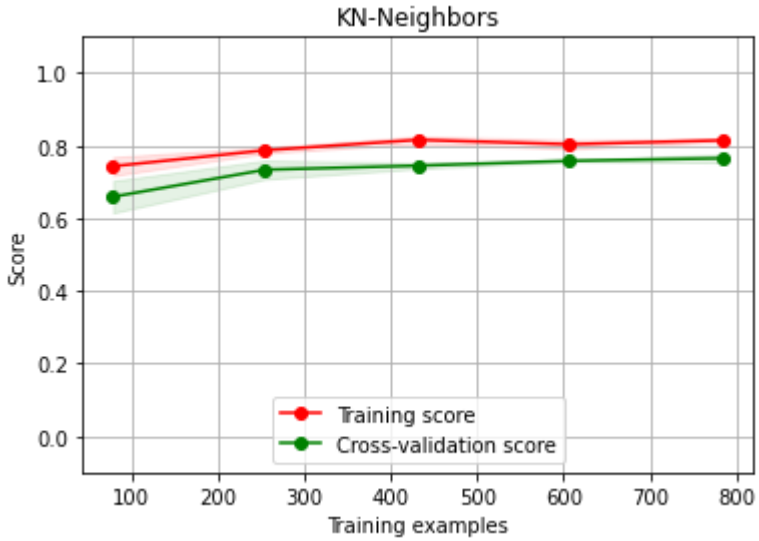
Out[48]:
```
([<matplotlib.axis.XTick at 0x7662e0308>,
  <matplotlib.axis.XTick at 0x7662e1548>,
  <matplotlib.axis.XTick at 0x7662e0b48>,
  <matplotlib.axis.XTick at 0x76630bb08>,
  <matplotlib.axis.XTick at 0x766313448>],
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```



In [49]:
```
print('Accuracy: %.2f'%((best_score*100)))
```

```
Accuracy: 79.10
```

In [50]:
```
Learning_curve(best_model,'KN-Neighbors')
```



As we can see the training score (red line) is at its maximum regardless of training examples which Indicates overfitting.

In [51]:
```
predictions =best_model.predict(X_test)
print(pd.crosstab(y_test, predictions,colnames=['Predicted'],rownames=['Actual'],
```

```
Predicted  dropOut  regular  All
Actual
dropOut         32       86  118
regular          2      301  303
All             34      387  421
```

In [52]:
```
print(classification_report(predictions, y_test))
```

```
              precision    recall  f1-score   support

     dropOut       0.27      0.94      0.42        34
     regular       0.99      0.78      0.87       387

    accuracy                           0.79       421
   macro avg       0.63      0.86      0.65       421
weighted avg       0.94      0.79      0.84       421
```

2: AdaBoost Classifier

In [53]:
```
from sklearn.ensemble import AdaBoostClassifier
from sklearn import datasets
from sklearn import metrics
```

In [54]:
```
# Create adaboost classifer object
abc = AdaBoostClassifier(n_estimators=50, learning_rate=1)
```

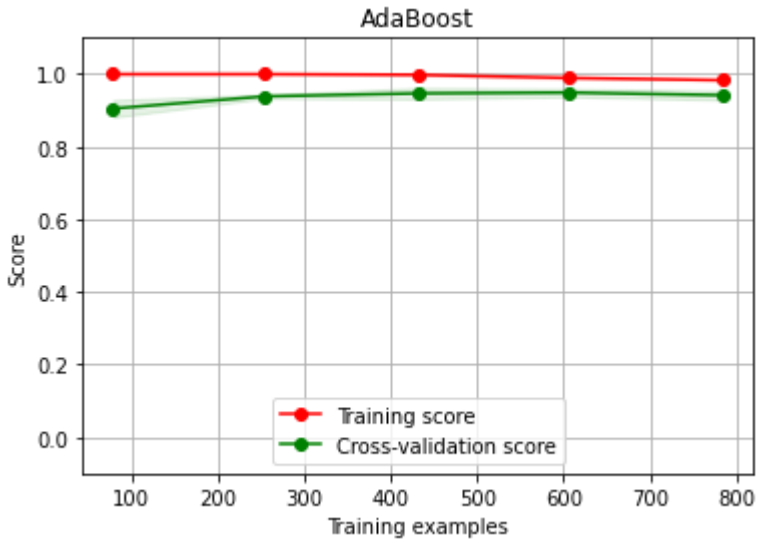
In [55]:
```
# Train Adaboost Classifer
model = abc.fit(X_train, y_train)
```

In [56]:
```
#Predict the response for test dataset
y_pred = model.predict(X_test)
```

In [57]:
```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred)*100)
```

```
Accuracy: 96.19952494061758
```

In [58]:
```
Learning_curve(model,'AdaBoost')
```



In [59]:
```
print(pd.crosstab(y_test, y_pred,colnames=['Predicted'],rownames=['Actual'], marg
```

```
Predicted  dropOut  regular  All
Actual
dropOut        111        7  118
regular          9      294  303
All            120      301  421
```

In [60]:
```
print(classification_report(y_pred, y_test))
```

```
              precision    recall  f1-score   support

     dropOut       0.94      0.93      0.93       120
     regular       0.97      0.98      0.97       301

    accuracy                           0.96       421
   macro avg       0.96      0.95      0.95       421
weighted avg       0.96      0.96      0.96       421
```

3: Naive Bayes Classifier

In [61]:
```
# training the model on training set
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)
```

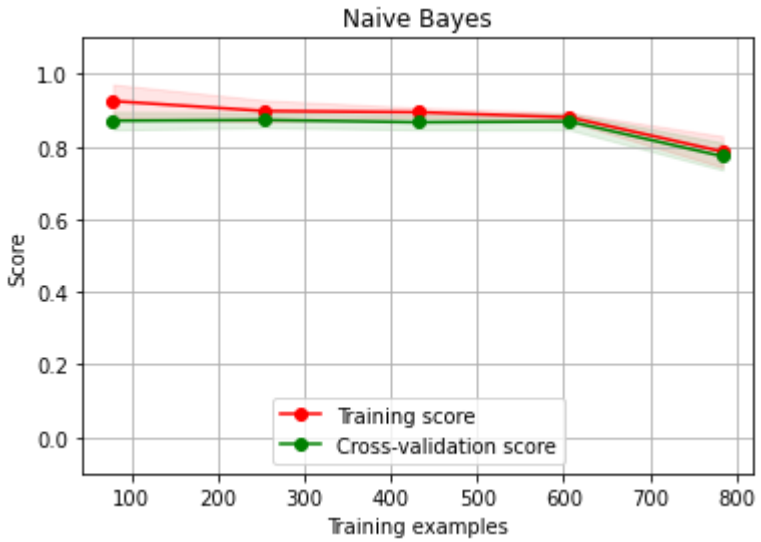
Out[61]:
```
GaussianNB()
```

In [62]:
```
# making predictions on the testing set
y_pred = gnb.predict(X_test)
```

In [63]:
```
# comparing actual response values (y_test) with predicted response values (y_pre
print("Naive Bayes model accuracy(in %):", metrics.accuracy_score(y_test, y_pred)
```

```
Naive Bayes model accuracy(in %): 74.58432304038006
```

In [64]:
```
Learning_curve(gnb,'Naive Bayes')
```



In [65]:
```
print(pd.crosstab(y_test, y_pred,colnames=['Predicted'],rownames=['Actual'], marg
```

```
Predicted  dropOut  regular  All
Actual
dropOut        112        6  118
regular        101      202  303
All            213      208  421
```

In [66]:
```
print(classification_report(y_pred, y_test))
```

```
              precision    recall  f1-score   support

     dropOut       0.95      0.53      0.68       213
     regular       0.67      0.97      0.79       208

    accuracy                           0.75       421
   macro avg       0.81      0.75      0.73       421
weighted avg       0.81      0.75      0.73       421
```

4: MLP (Multilayer Perceptron)

In [67]:
```
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
import seaborn as sns
```

In [68]:
```
clf = MLPClassifier(solver='adam', activation='relu', alpha=1e-4 ,
                    hidden_layer_sizes=(50,50,50), random_state=1, max_iter=11 ,
                    verbose= 10 ,learning_rate_init=0.1)
```

In [69]:
```
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

```
Iteration 1, loss = 12.58617054
Iteration 2, loss = 14.90997159
Iteration 3, loss = 3.93755438
Iteration 4, loss = 0.57240963
Iteration 5, loss = 0.57790091
Iteration 6, loss = 0.57080683
Iteration 7, loss = 0.57340219
Iteration 8, loss = 0.57443270
Iteration 9, loss = 0.57106865
Iteration 10, loss = 0.56792317
Iteration 11, loss = 0.56906899

C:\Users\Folio\Anaconda3\envs\tenseEnv\lib\site-packages\sklearn\neural_network
\_multilayer_perceptron.py:585: ConvergenceWarning: Stochastic Optimizer: Maxim
um iterations (11) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

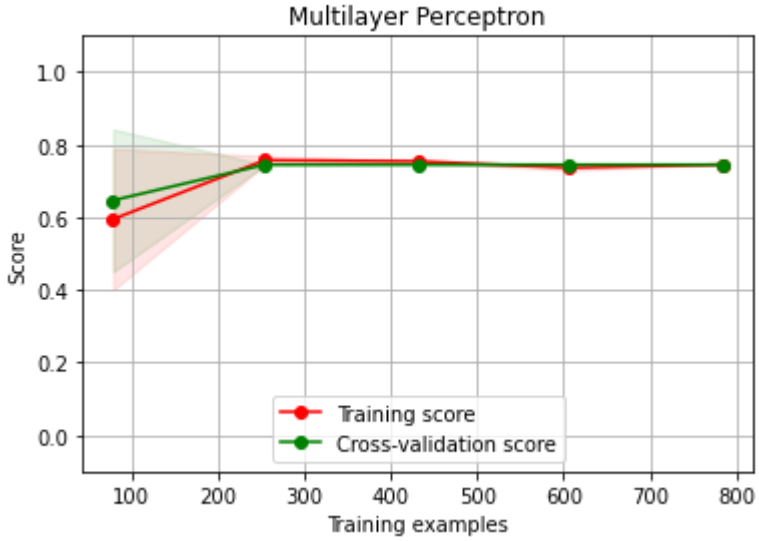
In [70]:
```
print (clf.n_layers_)
print (clf.n_iter_)
print (clf.loss_)
```

```
5
11
0.5690689913437746
```

In [71]:
```
print(accuracy_score(y_test, y_pred)*100)
```

```
71.97149643705463
```

In [72]:
```
Learning_curve(clf,'Multilayer Perceptron')
```



In [73]:
```
print(confusion_matrix(y_test, y_pred))
```

```
[[  0 118]
 [  0 303]]
```

In [74]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

     dropOut       0.00      0.00      0.00       118
     regular       0.72      1.00      0.84       303

    accuracy                           0.72       421
   macro avg       0.36      0.50      0.42       421
weighted avg       0.52      0.72      0.60       421


C:\Users\Folio\Anaconda3\envs\tenseEnv\lib\site-packages\sklearn\metrics\_classification.py:1221: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

5: Logistic Regression

In [75]:
```
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
```

In [76]:
```
model = LogisticRegression(solver='liblinear', random_state=0)
```

In [77]:
```
model.fit(X, y)
```

Out[77]:
```
LogisticRegression(random_state=0, solver='liblinear')
```

In [78]:
```
print(model.score(X, y)*100)
```

```
87.37517831669044
```

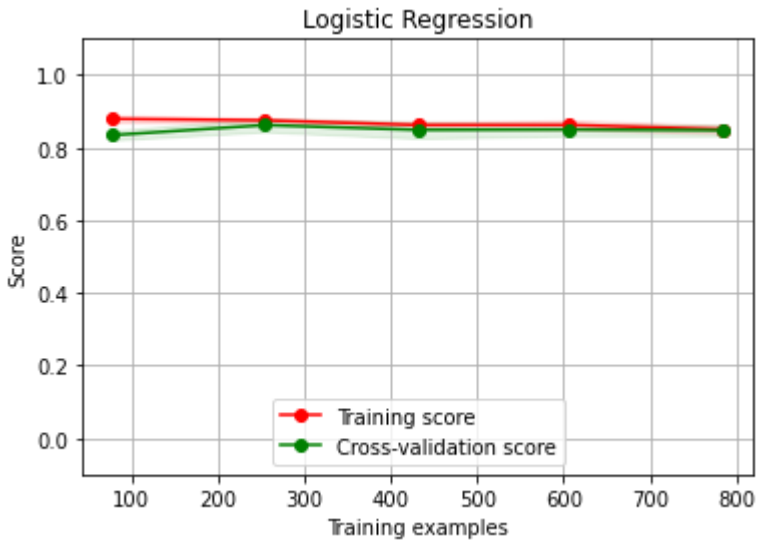
In [79]:
```
confusion_matrix(y, model.predict(X))
```

Out[79]:
```
array([[248, 120],
       [ 57, 977]], dtype=int64)
```

In [80]:
```
print(classification_report(y, model.predict(X)))
```

```
              precision    recall  f1-score   support

     dropOut       0.81      0.67      0.74       368
     regular       0.89      0.94      0.92      1034

    accuracy                           0.87      1402
   macro avg       0.85      0.81      0.83      1402
weighted avg       0.87      0.87      0.87      1402
```

In [81]:
```
Learning_curve(model,'Logistic Regression')
```



6: Random Forest

In [82]:
```
parameters={"max_depth": [10,11,12,13,14,15]
            ,"min_samples_split" :[2,3,4,5,6]
            ,"min_samples_leaf": [16,17,18]
            ,"n_estimators" : [25]
            ,"max_features": (4,5,6,"sqrt","auto")
            ,"criterion": ('gini','entropy')}
```

In [83]:
```
classifier = RandomForestClassifier( n_jobs=-1,random_state=5,oob_score =True)
```

In [84]:
```
rf_model= GridSearchCV(estimator=classifier,param_grid=parameters,cv=5)
```

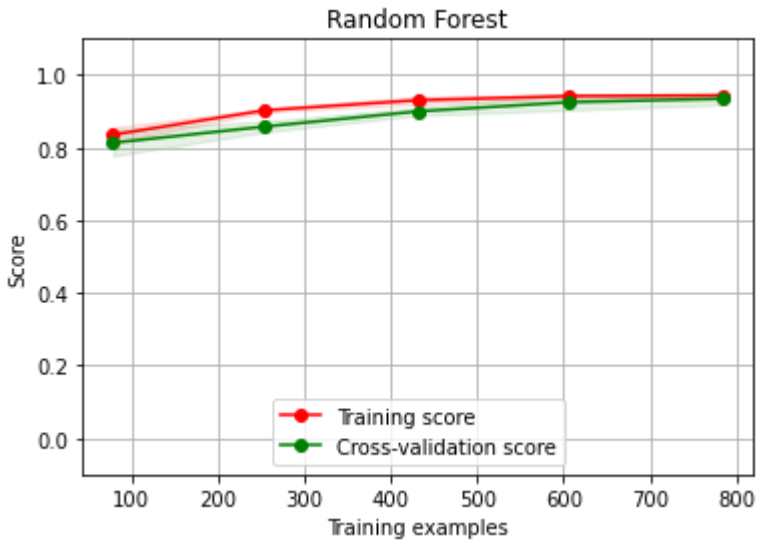
In [85]:
```
rf_model.fit(X_train, y_train)
```

Out[85]:
```
GridSearchCV(cv=5,
             estimator=RandomForestClassifier(n_jobs=-1, oob_score=True,
                                              random_state=5),
             param_grid={'criterion': ('gini', 'entropy'),
                         'max_depth': [10, 11, 12, 13, 14, 15],
                         'max_features': (4, 5, 6, 'sqrt', 'auto'),
                         'min_samples_leaf': [16, 17, 18],
                         'min_samples_split': [2, 3, 4, 5, 6],
                         'n_estimators': [25]})
```

In [86]:
```
print('Accuracy: %.2f'%((rf_model.score(X_test,y_test)*100)))
```

```
Accuracy: 91.69
```

In [87]:
```
Learning_curve(rf_model,'Random Forest')
```



In [88]:
```
rf_model.best_params_
```

Out[88]:
```
{'criterion': 'gini',
 'max_depth': 10,
 'max_features': 6,
 'min_samples_leaf': 16,
 'min_samples_split': 2,
 'n_estimators': 25}
```

In [89]:
```
predictions =rf_model.predict(X_test)
print(pd.crosstab(y_test, predictions, colnames=['Predicted'],rownames=['Actual']
```

```
Predicted  dropOut  regular  All
Actual
dropOut        104       14  118
regular         21      282  303
All            125      296  421
```

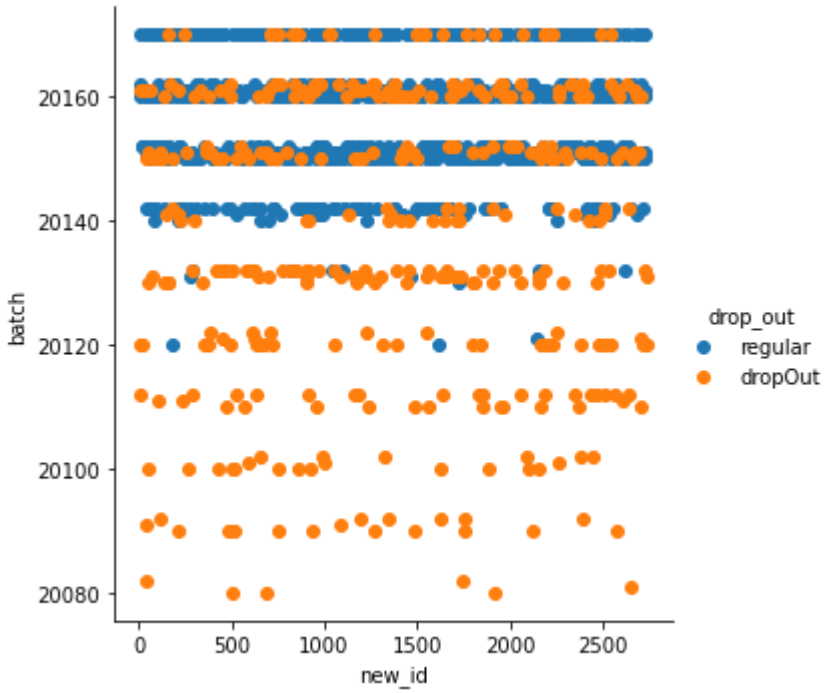
In [90]:
```
print(classification_report(predictions, y_test))
```

```
              precision    recall  f1-score   support

     dropOut       0.88      0.83      0.86       125
     regular       0.93      0.95      0.94       296

    accuracy                           0.92       421
   macro avg       0.91      0.89      0.90       421
weighted avg       0.92      0.92      0.92       421
```

In [91]:
```
import seaborn as sns
sns.FacetGrid(data, hue="drop_out", size=5) \
   .map(plt.scatter, 'new_id', 'batch') \
   .add_legend()
```

```
C:\Users\Folio\Anaconda3\envs\tenseEnv\lib\site-packages\seaborn\axisgrid.py:31
6: UserWarning: The `size` parameter has been renamed to `height`; please updat
e your code.
  warnings.warn(msg, UserWarning)
```

Out[91]:
```
<seaborn.axisgrid.FacetGrid at 0x766c0a488>
```



1. KNN = 79.10%
2. AdaBoost = 96.19%
3. Naive Bayes = 74.58%
4. MLP = 71.97%
5. Logistic Regression = 87.37%
6. Random Forest = 91.69%

In [ ]: