In [0]:
```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
from tensorflow.python.client import device_lib
import tensorflow as tf
print(tf.test.gpu_device_name())
print(device_lib.list_local_devices())
```

```
/device:GPU:0
[name: "/device:CPU:0"
device_type: "CPU"
memory_limit: 268435456
locality {
}
incarnation: 934462565136851997
, name: "/device:XLA_CPU:0"
device_type: "XLA_CPU"
memory_limit: 17179869184
locality {
}
incarnation: 2256987005148651679
physical_device_desc: "device: XLA_CPU device"
, name: "/device:XLA_GPU:0"
device_type: "XLA_GPU"
memory_limit: 17179869184
locality {
}
incarnation: 11892243292689056584
physical_device_desc: "device: XLA_GPU device"
, name: "/device:GPU:0"
device_type: "GPU"
memory_limit: 14800692839
locality {
  bus_id: 1
  links {
  }
}
incarnation: 12858181563179433425
physical_device_desc: "device: 0, name: Tesla T4, pci bus id: 0000:00:04.0,
compute capability: 7.5"
]
```

In [0]:
```
from tensorflow.keras import applications
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import optimizers
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dropout, Flatten, Dense
```

In [0]:
```
import tensorflow.keras.backend as K
K.clear_session()
```

In [0]:
```
!apt-get install -y -qq software-properties-common python-software-propertie
!add-apt-repository -y ppa:alessandro-strada/ppa 2>&1 > /dev/null
!apt-get update -qq 2>&1 > /dev/null
!apt-get -y install -qq google-drive-ocamlfuse fuse
from google.colab import auth
auth.authenticate_user()
from oauth2client.client import GoogleCredentials
creds = GoogleCredentials.get_application_default()
import getpass
!google-drive-ocamlfuse -headless -id={creds.client_id} -secret={creds.clier
vcode = getpass.getpass()
!echo {vcode} | google-drive-ocamlfuse -headless -id={creds.client_id} -secr
```

```
E: Package 'python-software-properties' has no installation candidate
Selecting previously unselected package google-drive-ocamlfuse.
(Reading database ... 130824 files and directories currently installed.)
Preparing to unpack .../google-drive-ocamlfuse_0.7.3-0ubuntu3~ubuntu18.04.1
_amd64.deb ...
Unpacking google-drive-ocamlfuse (0.7.3-0ubuntu3~ubuntu18.04.1) ...
Setting up google-drive-ocamlfuse (0.7.3-0ubuntu3~ubuntu18.04.1) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Please, open the following URL in a web browser: https://accounts.google.co
m/o/oauth2/auth?client_id=32555940559.apps.googleusercontent.com&redirect_u
ri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=https%3A%2F%2Fwww.googleapis.c
om%2Fauth%2Fdrive&response_type=code&access_type=offline&approval_prompt=fo
rce (https://accounts.google.com/o/oauth2/auth?client_id=32555940559.apps.g
oogleusercontent.com&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope
=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive&response_type=code&access_
type=offline&approval_prompt=force)
..........
Please, open the following URL in a web browser: https://accounts.google.co
m/o/oauth2/auth?client_id=32555940559.apps.googleusercontent.com&redirect_u
ri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=https%3A%2F%2Fwww.googleapis.c
om%2Fauth%2Fdrive&response_type=code&access_type=offline&approval_prompt=fo
rce (https://accounts.google.com/o/oauth2/auth?client_id=32555940559.apps.g
oogleusercontent.com&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope
=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive&response_type=code&access_
type=offline&approval_prompt=force)
Please enter the verification code: Access token retrieved correctly.
```

In [0]:
```
!mkdir -p drive
!google-drive-ocamlfuse drive
```

In [0]:
```
import os
os.chdir("drive/Colab Notebooks")
```

Pre-process the data in excel, and get the 4 academic years Quanzhou Normal University libarary book borrowing record of undergraduate. We use undergraduates' year of class, gender, shcool to predict what kind of books they are likely to borrow. we use Chinese books classification principal to divide all the books into 22 groups.

In [5]:

```
names =[
'ID', # undergraduate class
'Gender', 'School', 'Type',
]
y2014=[]
y2015=[]
y2016=[]
y2017=[]
df2014 = pd.read_csv('/content/drive/MLproject/2014.csv',names=names)
df2015 = pd.read_csv('/content/drive/MLproject/2015.csv',names=names)
df2016 = pd.read_csv('/content/drive/MLproject/2016.csv',names=names)
df2017 = pd.read_csv('/content/drive/MLproject/2017.csv',names=names)

df2014 = df2014.dropna()
df2015 = df2015.dropna()
df2016 = df2016.dropna()
df2017 = df2017.dropna()

T1 = np.array(df2014['Type'])
T2 = np.array(df2015['Type'])
T3 = np.array(df2016['Type'])
T4 = np.array(df2017['Type'])
a1 =T1.shape[0]
a2 =T2.shape[0]
a3 =T3.shape[0]
a4 =T4.shape[0]
# y2014=np.zeros((a1)).tostring()
# y2015=np.zeros((a2)).tostring()
# y2016=np.zeros((a3)).tostring()
# y2017=np.zeros((a4)).tostring()

for i in range(a1):
    y = str(T1[i])
    y2014.append(y[0])
for i in range(a2):
    y = str(T2[i])
    y2015.append(y[0])
for i in range(a3):
    y = str(T3[i])
    y2016.append(y[0])
for i in range(a4):
    y = str(T4[i])
    y2017.append(y[0])
print(y2017[:10])   #get the first character of call number string of each b
X2014 = np.array(df2014[['ID','Gender','School']])
X2015 = np.array(df2015[['ID','Gender','School']])
X2016 = np.array(df2016[['ID','Gender','School']])
X2017 = np.array(df2017[['ID','Gender','School']])

for i in range(a1):
    if X2014[i][0]<12e7:
        X2014[i][0]=3
        continue
    if 12e7<X2014[i][0]<13e7:
        X2014[i][0]=2
        continue
```

```
    if 13e7<X2014[i][0]<14e7:
        X2014[i][0]=1
        continue
    if 14e7<X2014[i][0]:
        X2014[i][0]=0

for i in range(a2):
    if X2015[i][0]<13e7:
        X2015[i][0]=3
    if 13e7<X2015[i][0]<14e7:
        X2015[i][0]=2
    if 14e7<X2015[i][0]<15e7:
        X2015[i][0]=1
    if 15e7<X2015[i][0]:
        X2015[i][0]=0

for i in range(a3):
    if X2016[i][0]<14e7:
        X2016[i][0]=3
    if 14e7<X2016[i][0]<15e7:
        X2016[i][0]=2
    if 15e7<X2016[i][0]<16e7:
        X2016[i][0]=1
    if 16e7<X2016[i][0]:
        X2016[i][0]=0

for i in range(a4):
    if X2017[i][0]<15e7:
        X2017[i][0]=3
    if 15e7<X2017[i][0]<16e7:
        X2017[i][0]=2
    if 16e7<X2017[i][0]<17e7:
        X2017[i][0]=1
    if 17e7<X2017[i][0]:
        X2017[i][0]=0
print(X2014.shape)
print(X2014[10000:10030])
Xtr=np.vstack((X2014,X2016,X2017))
b1=np.array(y2014).reshape((a1,1))
b2=np.array(y2016).reshape((a3,1))
b3=np.array(y2017).reshape((a4,1))
ytr=np.vstack((b1,b2,b3))
Xts=X2015
yts=np.array(y2015).reshape((a2,1))
print(Xtr.shape)
print(ytr.shape)

print(Xts.shape)
print(yts.shape)
```

```
['I', 'B', 'K', 'F', 'I', 'I', 'I', 'T', 'T', 'I']
(62871, 3)
[[2 'F' 'School of Literature and Communication']
 [2 'F' 'School of Literature and Communication']
 [2 'F' 'School of Literature and Communication']
```

```
[2 'F' 'School of Literature and Communication']
[1 'F' 'School of Educational Science']
[2 'F' 'TS School of Business and Information Technology']
[2 'F' 'TS School of Business and Information Technology']
[0 'M' 'School of Foreign Languages']
[0 'M' 'School of Foreign Languages']
[0 'M' 'School of Foreign Languages']
[0 'M' 'School of Literature and Communication']
[2 'M' 'School of Foreign Languages']
[1 'F' 'School of Foreign Languages']
[0 'F' 'School of Foreign Languages']
[0 'F' 'School of Foreign Languages']
[2 'F' 'School of Foreign Languages']
[1 'F' 'School of Resources and Environmental Science']
```

In order to use on-hot coding, we assign different number to different class in second and third features for Xtr and Xts

In [6]:

```
def one_hot(X):
    X_row=X.shape[0]
    for i in range(X_row):
        if X[i][1]=='M':
            X[i][1]=0
        if X[i][1]=='F':
            X[i][1]=1
    for i in range(X_row):
        if X[i][2]=='School of Resources and Environmental Science'or X[i][2
            X[i][2]=0
        if X[i][2]=='School of Applied Technology':
            X[i][2]=1
        if X[i][2]=='School of Chemical Engineering and Material'or X[i][2]=
            X[i][2]=2
        if X[i][2]=='School of Educational Science':
            X[i][2]=3
        if X[i][2]=='School of Fine Arts and Design':
            X[i][2]=4
        if X[i][2]=='School of Foreign Languages':
            X[i][2]=5
        if X[i][2]=='School of Literature and Communication':
            X[i][2]=6
        if X[i][2]=='School of Maths and Computer Science':
            X[i][2]=7
        if X[i][2]=='School of Music and Dance':
            X[i][2]=8
        if X[i][2]=='School of Physical Education':
            X[i][2]=9
        if X[i][2]=='School of Physics and Information Engineering':
            X[i][2]=10
        if X[i][2]=='School of Politics and Social Development':
            X[i][2]=11
        if X[i][2]=='School of Sailing':
            X[i][2]=12
        if X[i][2]=='TS School of Business and Information Technology':
            X[i][2]=13
    return(X)

one_hot(X=Xtr)
one_hot(X=Xts)
print(Xtr[:10,:])
print(Xts[:10,:])
```

```
[[1 1 13]
 [1 1 13]
 [1 1 13]
 [1 1 13]
 [3 1 6]
 [3 1 6]
 [3 1 6]
 [2 1 3]
 [2 1 3]
 [2 1 3]]
[[1 1 1]
```

```
[2 1 13]
[2 1 13]
[2 1 13]
[2 1 13]
[3 1 1]
[3 1 1]
[3 1 1]
[2 1 13]
[3 0 2]]
```

Use one-hot coding, 4 codes for class year, 2 codes for gender, 14 codes for shcool. So, for each student, need 20 codes.

Type *Markdown* and LaTeX: α^2

In [7]:

```
print(Xtr[:5,:])

from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder(categories='auto')
enc.fit(Xtr)
# print("enc_values_ is:",enc.n_values_)
Xtr1=enc.transform(Xtr).toarray()
Xts1=enc.transform(Xts).toarray()
print(Xtr1.shape)
print(Xtr1[:5,:])
```

```
[[1 1 13]
 [1 1 13]
 [1 1 13]
 [1 1 13]
 [3 1 6]]
(137391, 20)
[[0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 1. 0. 1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]]
```

1.LogisticsRegression:

In [8]:
```
ytr1=ytr.ravel()
from sklearn import linear_model
logreg = linear_model.LogisticRegression(verbose=10, solver='sag',\
                                         multi_class='multinomial',max_iter=
logreg.fit(Xtr1,ytr1)
yhat=logreg.predict_proba(Xts1)
print(yhat.shape)
a=yhat[:100]
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent wor
kers.

convergence after 36 epochs took 14 seconds
(57245, 22)

[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:   14.2s remaining:
0.0s
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:   14.2s finished
```

Print the first three class with largest probability for testing data. We will recommend students these three kind of books.

In [9]:
```
class_name1=['A','B','C','D','E','F','G','H','I','J','K','N','O','P','Q','R'
b=np.zeros((10,3))
for i in range(10):
    b[i]=np.argsort(-a[i])[:3]    ## b is the first three largest class's ir


ntop = 3
res_dict = {}
for i in range(ntop):
    class_name = []
    class_prob = []
    for j in range(10):
        b1=int(b[j][i])
        class_name.append(class_name1[b1])
        class_prob.append(a[j][b1])

    name_col = str('class %d' % i)
    prob_col = str('prob %d' % i)
    res_dict[name_col] = class_name
    res_dict[prob_col] = class_prob

df = pd.DataFrame(data=res_dict)
df
```

Out[9]:

	class 0	prob 0	class 1	prob 1	class 2	prob 2
0	I	0.544209	H	0.122286	K	0.070817
1	I	0.397560	T	0.180065	F	0.137745
2	I	0.397560	T	0.180065	F	0.137745
3	I	0.397560	T	0.180065	F	0.137745
4	I	0.397560	T	0.180065	F	0.137745
5	I	0.291480	U	0.190304	F	0.141148
6	I	0.291480	U	0.190304	F	0.141148
7	I	0.291480	U	0.190304	F	0.141148
8	I	0.397560	T	0.180065	F	0.137745
9	I	0.188973	T	0.174859	O	0.120709

In [10]:
```
Xts_row=Xts.shape[0]
b=np.zeros((Xts_row,3))
counter=0
for i in range(Xts_row):
    b[i]=np.argsort(-yhat[i])[:3]    ## b is the first three largest class's

for i in range(Xts_row):
    max1=int(b[i][0])
    max2=int(b[i][1])
    max3=int(b[i][2])
    if yts[i] == class_name1[max1] or yts[i]== class_name1[max2] or yts[i]==
        counter=counter+1
log_accuracy=counter/Xts_row
print(log_accuracy)
```

```
0.7502663988121233
```

2.SVM:

In [11]:
```
svmnum=20000   # train and test sample number
from sklearn import svm
svcrbf = svm.SVC(probability=True,  kernel="rbf", C=0.1, gamma=.001,verbose=
svcrbf.fit(Xtr1[:svmnum,:],ytr1[:svmnum])
yhat=svcrbf.predict_proba(Xts1[:svmnum])
a=yhat[:100]
```

```
[LibSVM]
```

In [12]:
```
print(a[:5])
```

```
[[0.00479515 0.05591309 0.02362029 0.01050184 0.00240698 0.05065337
  0.01279957 0.13281572 0.51133113 0.03939331 0.07131815 0.00183547
  0.01073194 0.00192827 0.00258753 0.00590441 0.00300688 0.04778115
  0.00745797 0.00064557 0.00137536 0.00119685]
 [0.00458502 0.04958149 0.0218675  0.01754072 0.00267347 0.06793454
  0.0163636  0.09933344 0.44829186 0.05278125 0.07064934 0.00226998
  0.0095298  0.00229179 0.00365817 0.00740017 0.00202904 0.11494328
  0.00148877 0.00072926 0.00245471 0.00160281]
 [0.00458502 0.04958149 0.0218675  0.01754072 0.00267347 0.06793454
  0.0163636  0.09933344 0.44829186 0.05278125 0.07064934 0.00226998
  0.0095298  0.00229179 0.00365817 0.00740017 0.00202904 0.11494328
  0.00148877 0.00072926 0.00245471 0.00160281]
 [0.00458502 0.04958149 0.0218675  0.01754072 0.00267347 0.06793454
  0.0163636  0.09933344 0.44829186 0.05278125 0.07064934 0.00226998
  0.0095298  0.00229179 0.00365817 0.00740017 0.00202904 0.11494328
  0.00148877 0.00072926 0.00245471 0.00160281]
 [0.00458502 0.04958149 0.0218675  0.01754072 0.00267347 0.06793454
  0.0163636  0.09933344 0.44829186 0.05278125 0.07064934 0.00226998
  0.0095298  0.00229179 0.00365817 0.00740017 0.00202904 0.11494328
  0.00148877 0.00072926 0.00245471 0.00160281]]
```

In [13]:
```
for i in range(ntop):
    class_name = []
    class_prob = []
    for j in range(10):
        b1=int(b[j][i])
        class_name.append(class_name1[b1])
        class_prob.append(a[j][b1])

    name_col = str('class %d' % i)
    prob_col = str('prob %d' % i)
    res_dict[name_col] = class_name
    res_dict[prob_col] = class_prob

df = pd.DataFrame(data=res_dict)
df
```

Out[13]:

	class 0	prob 0	class 1	prob 1	class 2	prob 2
0	I	0.511331	H	0.132816	K	0.071318
1	I	0.448292	T	0.114943	F	0.067935
2	I	0.448292	T	0.114943	F	0.067935
3	I	0.448292	T	0.114943	F	0.067935
4	I	0.448292	T	0.114943	F	0.067935
5	I	0.286387	U	0.019422	F	0.156400
6	I	0.286387	U	0.019422	F	0.156400
7	I	0.286387	U	0.019422	F	0.156400
8	I	0.448292	T	0.114943	F	0.067935
9	I	0.316972	T	0.121753	O	0.124186

In [14]:
```
b=np.zeros((svmnum,3))
counter=0
for i in range(svmnum):
    b[i]=np.argsort(-yhat[i])[:3]    ## b is the first three largest class's

for i in range(svmnum):
    max1=int(b[i][0])
    max2=int(b[i][1])
    max3=int(b[i][2])
    if yts[i] == class_name1[max1] or yts[i]== class_name1[max2] or yts[i]==
        counter=counter+1
log_accuracy=counter/svmnum
print(log_accuracy)
```

```
0.73795
```

In [0]:
```
```