Pre-Processing:

Cleaning

import pandas as pd

import numpy as np

import re

def fun1(maledf1,staging,femaledf1,part=1):

temp\_male = maledf1[(part-1)\*len(maledf1)/24:part\*len(maledf1)/24]

temp\_female = femaledf1[(part-1)\*len(femaledf1)/24:part\*len(femaledf1)/24]

temp = temp\_male.append(temp\_female)

file1 = staging + str(part) + '.csv'

temp.to\_csv(file1,sep='\t')

#Special Charter removal#

def preprocess(in\_file,staging):

df1=pd.DataFrame.from\_csv(in\_file, sep='\t')

df1['post'] = df1['post'].str.lower()

df1['post'] = df1['post'].str.replace('urllink', '')

df1['post'] = df1['post'].str.replace('http:\/\/.\*[\r\n]\*', '')

df1['post'] = df1['post'].str.replace('(', '')

df1['post'] = df1['post'].str.replace(')', '')

df1['post'] = df1['post'].str.replace('@', '')

df1['post'] = df1['post'].str.replace('#', '')

df1['post'] = df1['post'].str.replace('-', '')

df1['post'] = df1['post'].str.replace('\*', '')

df1['post'] = df1['post'].str.replace('^', '')

df1['post'] = df1['post'].str.replace("'and'nbsp;", '')

df1['post'] = df1['post'].str.replace('[', '')

df1['post'] = df1['post'].str.replace(']', '')

df1['post'] = df1['post'].str.replace('{', '')

df1['post'] = df1['post'].str.replace('}', '')

df1['post'] = df1['post'].str.replace('<', '')

df1['post'] = df1['post'].str.replace('>', '')

df1['post'] = df1['post'].str.replace('\s+', ' ')

df1['post'] = df1['post'].str.replace('~', '')

df1["post"] = [re.sub(r'\.\.+', '',str(i)) for i in df1["post"]]

df1["post"] = [re.sub(r'"', '',str(i)) for i in df1["post"]]

df1["post"] = [re.sub(r'|', '',str(i)) for i in df1["post"]]

df1["post"] = [re.sub(r'!', '',str(i)) for i in df1["post"]]

df1["post"] = [re.sub(r',', '',str(i)) for i in df1["post"]]

df1['post'] = df1['post'].str.strip()

df1=df1[df1.post<>'']

np.random.seed(1000)

maledf1=df1[df1.gender=='male']

maledf1 = maledf1.iloc[np.random.permutation(len(maledf1))]

femaledf1=df1[df1.gender=='female']

femaledf1 = femaledf1.iloc[np.random.permutation(len(femaledf1))]

for i in xrange(1,25):

fun1(maledf1,femaledf1,i)

print "create the Gender Sample " + str(i)

Document to vectorizations

import pandas as pd

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.feature\_extraction.text import TfidfVectorizer

from nltk.tokenize import RegexpTokenizer

import re

import time

start\_time=time.clock()

infile = pd.DataFrame.from\_csv('D:/Gender Identification Ext/GenderSample40K ADD.csv', sep=',')

dictGen = pd.DataFrame.from\_csv('D:/Gender Identification Ext/fdict.csv',encoding = 'iso8859\_16')

gender\_dict = dictGen.set\_index('id')['value'].to\_dict()

tokenizer=RegexpTokenizer(r'\w+')

infile['post'] = infile['post'].str.lower()

def funMono(arg1):

termm = tokenizer.tokenize(arg1)

return ' '.join(map(lambda x: gender\_dict.get(x,''),termm))

def funBi(arg1):

termb = re.findall(r"\b\w+\s\w+", arg1)

return ' '.join(map(lambda x: gender\_dict.get(x,''),termb))

def funTri(arg1):

termt = re.findall(r"\b\w+\s\w+\s\w+", arg1)

return ' '.join(map(lambda x: gender\_dict.get(x,''),termt))

def funTetra(arg1):

termte = re.findall(r"\b\w+\s\w+\s\w+\s\w+", arg1)

return ' '.join(map(lambda x: gender\_dict.get(x,''),termte))

infile['monogram'] = infile['post'].apply(funMono)

infile['bigram'] = infile['post'].apply(funBi)

infile['Trigram'] = infile['post'].apply(funTri)

infile['Tetragram'] = infile['post'].apply(funTetra)

infile['Doctagger'] = infile.monogram.str.cat(others=[infile.bigram, infile.Trigram, infile.Tetragram], sep=' ')

infile['Doctagger'] = infile['Doctagger'].str.replace('\s+', ' ')

infile['Doctagger'] = infile['Doctagger'].str.replace('^\s+', '')

######BIGRAM Vectorization

bigram\_vectorizer = TfidfVectorizer(ngram\_range=(1,1), token\_pattern=r'\b\w+\b', min\_df=1)

bi\_analyze = bigram\_vectorizer.build\_analyzer()

bi\_XArray = bigram\_vectorizer.fit\_transform(infile['Doctagger']).toarray()

df2 = pd.DataFrame(bi\_XArray,columns = bigram\_vectorizer.get\_feature\_names(), index = infile.index)

bi\_inputDTM = pd.concat([infile['gender'], infile['age'], infile['topic'], infile['star\_sign'], infile['date'], infile['post'], df2], axis=1)

bi\_inputDTM.to\_csv('D:/Gender Identification Ext/outputTermMatrix40.csv')

print(time.clock()-start\_time,":In Seconds")

Gender model data preparation:

import pandas as pd

import numpy as np

from sklearn.ensemble import RandomForestClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn import preprocessing

from sklearn.metrics import roc\_curve, auc

number = preprocessing.LabelEncoder()

train=pd.read\_csv('D:/Gender Identification Ext/outputTermMatrix40.csv')

#test1=pd.read\_csv()

df\_topic= pd.get\_dummies(train['topic'])

train = pd.concat([train, df\_topic], axis=1)

#Step-2: Convert categorical variable to numpy arrays and fill NaN values to zero.

def convert(data):

number = preprocessing.LabelEncoder()

data['gender'] = number.fit\_transform(data.gender)

data['age'] = number.fit\_transform(data.age)

data['topic'] = number.fit\_transform(data.topic)

data['star\_sign'] = number.fit\_transform(data.star\_sign)

data['date'] = number.fit\_transform(data.date)

data['post'] = number.fit\_transform(data.post)

data['abstractnoun'] = number.fit\_transform(data.abstractnoun)

data['adjall'] = number.fit\_transform(data.adjall)

data['adjpert'] = number.fit\_transform(data.adjpert)

data['adjppl'] = number.fit\_transform(data.adjppl)

data['advall'] = number.fit\_transform(data.advall)

data['article'] = number.fit\_transform(data.article)

data['basicprepositions'] = number.fit\_transform(data.basicprepositions)

data['causalwords'] = number.fit\_transform(data.causalwords)

data['causation'] = number.fit\_transform(data.causation)

data['certaintywords'] = number.fit\_transform(data.certaintywords)

data['change'] = number.fit\_transform(data.change)

data['cognitive'] = number.fit\_transform(data.cognitive)

data['communicationofideas'] = number.fit\_transform(data.communicationofideas)

data['conjunctions'] = number.fit\_transform(data.conjunctions)

data['constraining'] = number.fit\_transform(data.constraining)

data['determiners'] = number.fit\_transform(data.determiners)

data['dimensions'] = number.fit\_transform(data.dimensions)

data['exclusivewords'] = number.fit\_transform(data.exclusivewords)

data['existence'] = number.fit\_transform(data.existence)

data['fact'] = number.fit\_transform(data.fact)

data['familyreference'] = number.fit\_transform(data.familyreference)

data['form'] = number.fit\_transform(data.form)

data['formationofideas'] = number.fit\_transform(data.formationofideas)

data['functionalwords'] = number.fit\_transform(data.functionalwords)

data['hedgephrases'] = number.fit\_transform(data.hedgephrases)

data['individualvolition'] = number.fit\_transform(data.individualvolition)

data['inorganicmatter'] = number.fit\_transform(data.inorganicmatter)

data['intersocialvolitionsection'] = number.fit\_transform(data.intersocialvolitionsection)

data['iwords'] = number.fit\_transform(data.iwords)

data['litigious'] = number.fit\_transform(data.litigious)

data['moralaffections'] = number.fit\_transform(data.moralaffections)

data['motion'] = number.fit\_transform(data.motion)

data['negative'] = number.fit\_transform(data.negative)

data['negativemood'] = number.fit\_transform(data.negativemood)

data['negativetone'] = number.fit\_transform(data.negativetone)

data['neutraltone'] = number.fit\_transform(data.neutraltone)

data['nounact'] = number.fit\_transform(data.nounact)

data['nounanimal'] = number.fit\_transform(data.nounanimal)

data['nounartifact'] = number.fit\_transform(data.nounartifact)

data['nounattribute'] = number.fit\_transform(data.nounattribute)

data['nounbody'] = number.fit\_transform(data.nounbody)

data['nouncognition'] = number.fit\_transform(data.nouncognition)

data['nouncommunication'] = number.fit\_transform(data.nouncommunication)

data['nounevent'] = number.fit\_transform(data.nounevent)

data['nounfeeling'] = number.fit\_transform(data.nounfeeling)

data['nounfood'] = number.fit\_transform(data.nounfood)

data['noungroup'] = number.fit\_transform(data.noungroup)

data['nounlocation'] = number.fit\_transform(data.nounlocation)

data['nounmotive'] = number.fit\_transform(data.nounmotive)

data['nounobject'] = number.fit\_transform(data.nounobject)

data['nounperson'] = number.fit\_transform(data.nounperson)

data['nounphenomenon'] = number.fit\_transform(data.nounphenomenon)

data['nounplant'] = number.fit\_transform(data.nounplant)

data['nounpossession'] = number.fit\_transform(data.nounpossession)

data['nounprocess'] = number.fit\_transform(data.nounprocess)

data['nounquantity'] = number.fit\_transform(data.nounquantity)

data['nounrelation'] = number.fit\_transform(data.nounrelation)

data['nounshape'] = number.fit\_transform(data.nounshape)

data['nounstate'] = number.fit\_transform(data.nounstate)

data['nounsubstance'] = number.fit\_transform(data.nounsubstance)

data['nounsword'] = number.fit\_transform(data.nounsword)

data['nountime'] = number.fit\_transform(data.nountime)

data['nountops'] = number.fit\_transform(data.nountops)

data['number'] = number.fit\_transform(data.number)

data['order'] = number.fit\_transform(data.order)

data['organicmatter'] = number.fit\_transform(data.organicmatter)

data['other'] = number.fit\_transform(data.other)

data['personalaffections'] = number.fit\_transform(data.personalaffections)

data['positive'] = number.fit\_transform(data.positive)

data['positivemood'] = number.fit\_transform(data.positivemood)

data['positivetone'] = number.fit\_transform(data.positivetone)

data['preposition'] = number.fit\_transform(data.preposition)

data['prepositions'] = number.fit\_transform(data.prepositions)

data['pronouns'] = number.fit\_transform(data.pronouns)

data['quantity'] = number.fit\_transform(data.quantity)

data['relation'] = number.fit\_transform(data.relation)

data['religiousaffections'] = number.fit\_transform(data.religiousaffections)

data['social'] = number.fit\_transform(data.social)

data['soundadjectives'] = number.fit\_transform(data.soundadjectives)

data['spaceingeneral'] = number.fit\_transform(data.spaceingeneral)

data['stopwords'] = number.fit\_transform(data.stopwords)

data['superfluous'] = number.fit\_transform(data.superfluous)

data['sympatheticaffections'] = number.fit\_transform(data.sympatheticaffections)

data['time'] = number.fit\_transform(data.time)

data['uncertainty'] = number.fit\_transform(data.uncertainty)

data['verbbody'] = number.fit\_transform(data.verbbody)

data['verbchange'] = number.fit\_transform(data.verbchange)

data['verbcognition'] = number.fit\_transform(data.verbcognition)

data['verbcommunication'] = number.fit\_transform(data.verbcommunication)

data['verbcompetition'] = number.fit\_transform(data.verbcompetition)

data['verbconsumption'] = number.fit\_transform(data.verbconsumption)

data['verbcontact'] = number.fit\_transform(data.verbcontact)

data['verbcreation'] = number.fit\_transform(data.verbcreation)

data['verbemotion'] = number.fit\_transform(data.verbemotion)

data['verbmotion'] = number.fit\_transform(data.verbmotion)

data['verbperception'] = number.fit\_transform(data.verbperception)

data['verbpossession'] = number.fit\_transform(data.verbpossession)

data['verbsocial'] = number.fit\_transform(data.verbsocial)

data['verbstative'] = number.fit\_transform(data.verbstative)

data=data.fillna(0)

return data

train=convert(train)

#Step-3: Split the data set to train and validate

train['is\_train'] = np.random.uniform(0, 1, len(train)) <= .75

train, validate = train[train['is\_train']==True], train[train['is\_train']==False]

x\_train = train[['abstractnoun','adjall','adjpert','adjppl','advall','article','basicprepositions','causalwords','causation','certaintywords','change','cognitive','communicationofideas','conjunctions','constraining','determiners','dimensions','exclusivewords','existence','fact','familyreference','form','formationofideas','functionalwords','hedgephrases','individualvolition','inorganicmatter','intersocialvolitionsection','iwords','litigious','moralaffections','motion','negative','negativemood','negativetone','neutraltone','nounact','nounanimal','nounartifact','nounattribute','nounbody','nouncognition','nouncommunication','nounevent','nounfeeling','nounfood','noungroup','nounlocation','nounmotive','nounobject','nounperson','nounphenomenon','nounplant','nounpossession','nounprocess','nounquantity','nounrelation','nounshape','nounstate','nounsubstance','nounsword','nountime','nountops','number','order','organicmatter','other','personalaffections','positive','positivemood','positivetone','preposition','prepositions','pronouns','quantity','relation','religiousaffections','social','soundadjectives','spaceingeneral','stopwords','superfluous','sympatheticaffections','time','uncertainty','verbbody','verbchange','verbcognition','verbcommunication','verbcompetition','verbconsumption','verbcontact','verbcreation','verbemotion','verbmotion','verbperception','verbpossession','verbsocial','verbstative','Accounting', 'Advertising', 'Agriculture', 'Architecture', 'Arts', 'Automotive', 'Banking', 'Biotech', 'BusinessServices', 'Chemicals', 'Communications-Media', 'Construction', 'Consulting', 'Education', 'Engineering', 'Environment', 'Fashion', 'Government', 'HumanResources', 'Internet', 'InvestmentBanking', 'Law', 'LawEnforcement-Security', 'Manufacturing', 'Marketing', 'Military', 'Museums-Libraries', 'Non-Profit', 'Publishing', 'RealEstate', 'Religion', 'Science', 'Sports-Recreation', 'Student', 'Technology', 'Telecommunications', 'Tourism', 'Transportation', 'indUnk']]

y\_train = train['gender']

x\_validate = validate[['abstractnoun','adjall','adjpert','adjppl','advall','article','basicprepositions','causalwords','causation','certaintywords','change','cognitive','communicationofideas','conjunctions','constraining','determiners','dimensions','exclusivewords','existence','fact','familyreference','form','formationofideas','functionalwords','hedgephrases','individualvolition','inorganicmatter','intersocialvolitionsection','iwords','litigious','moralaffections','motion','negative','negativemood','negativetone','neutraltone','nounact','nounanimal','nounartifact','nounattribute','nounbody','nouncognition','nouncommunication','nounevent','nounfeeling','nounfood','noungroup','nounlocation','nounmotive','nounobject','nounperson','nounphenomenon','nounplant','nounpossession','nounprocess','nounquantity','nounrelation','nounshape','nounstate','nounsubstance','nounsword','nountime','nountops','number','order','organicmatter','other','personalaffections','positive','positivemood','positivetone','preposition','prepositions','pronouns','quantity','relation','religiousaffections','social','soundadjectives','spaceingeneral','stopwords','superfluous','sympatheticaffections','time','uncertainty','verbbody','verbchange','verbcognition','verbcommunication','verbcompetition','verbconsumption','verbcontact','verbcreation','verbemotion','verbmotion','verbperception','verbpossession','verbsocial','verbstative','Accounting', 'Advertising', 'Agriculture', 'Architecture', 'Arts', 'Automotive', 'Banking', 'Biotech', 'BusinessServices', 'Chemicals', 'Communications-Media', 'Construction', 'Consulting', 'Education', 'Engineering', 'Environment', 'Fashion', 'Government', 'HumanResources', 'Internet', 'InvestmentBanking', 'Law', 'LawEnforcement-Security', 'Manufacturing', 'Marketing', 'Military', 'Museums-Libraries', 'Non-Profit', 'Publishing', 'RealEstate', 'Religion', 'Science', 'Sports-Recreation', 'Student', 'Technology', 'Telecommunications', 'Tourism', 'Transportation', 'indUnk']]

y\_validate = validate['gender']

Random forest final hyper parameter we have decided is,

import pandas as pd

import numpy as np

from sklearn.ensemble import RandomForestClassifier

cfl2 = RandomForestClassifier(n\_estimators=350, criterion = 'entropy',max\_features=None,n\_jobs=-1,warm\_start=True)

Praveen Include the scoring part in this