In [0]:

```
## Reading Statistics Papers
```

In [2]:

```
## Read the statistics papers from 2016 to 2018 and store it in a file
import urllib
url = 'http://export.arxiv.org/oai2?verb=ListRecords&set=stat&from=2016-01-01&until=2018-11-31&metadataPrefi
x=arXiv'
data = urllib.request.urlopen(url).read()

stat = open('stat1', 'wb')
stat.write(data)
```

Out[2]:

```
2032744
```

In [0]:

```
## Extract the title and abstract from papers - Read from finance1 to finance2

!xml_grep 'title|abstract' stat1  > stat2.txt
```

In [0]:

```
## Remove Junk lines , here we remove first 3 lines and last 3 lines which are not necessary
!cat stat2.txt | tail -n +4 | head -n -3 > stat3.txt
```

In [5]:

```
## Reading packages for Text classification
from sklearn import model_selection, preprocessing, linear_model, naive_bayes, metrics, svm
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn import decomposition, ensemble

import pandas, numpy, string
from keras.preprocessing import text, sequence
from keras import layers, models, optimizers
from nltk import word_tokenize
from nltk.corpus import stopwords
import sklearn
#import sklearn_crfsuite
#from sklearn_crfsuite import scorers
#from sklearn_crfsuite import metrics
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
from sklearn.decomposition import TruncatedSVD
from sklearn.metrics import accuracy_score
from sklearn import metrics
```

```
Using TensorFlow backend.
```

In [6]:

```
## Stopwords import and removal
import nltk
from nltk.corpus import stopwords

nltk.download('stopwords')
stopwords = set(stopwords.words('english'))
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [7]:

```python
# load the dataset # dataset contains combined labels and text from all training papers
data = open('labeled_sentences (1).txt').read()[:-2]
labels, texts = [], []
for i, line in enumerate(data.split("\n")):
    content = line.split()
    #print(content)
    labels.append(content[0])
    filtered_sentence = [w.lower() for w in content[1:] if not w in stopwords]
    texts.append(filtered_sentence)

# create a dataframe using texts and lables
trainDF = pandas.DataFrame()
trainDF['text'] = texts
trainDF['label'] = labels
print(trainDF['label'].unique())
trainDF.head(2)
```

```
['MISC' 'AIMX' 'OWNX' 'CONT' 'BASE']
```

Out[7]:

	text	label
0	[minimum, description, length, principle, onli...	MISC
1	[underlying, model, class, discrete,, total, e...	MISC

In [0]:

```python
## Used the obtained dataset for training
train_x, valid1_x, train_y, valid1_y = model_selection.train_test_split(trainDF['text'], trainDF['label'],te
st_size=0)
```

In [9]:

```python
## Convert from list to string
tempp = []

for item in train_x:
    tempp.append(" ".join(item))
#print(len(train_x))

#tempp1 =[]
#for item1 in valid_x:
    #tempp1.append(" ".join(item1))

#print(len(tempp1))

temp =[]
temp_len=0
for item2 in texts:
    temp.append(" ".join(item2))
    temp_len = temp_len+len(texts)
print(len(temp))
print(temp_len)
print(type(temp))
```

```
18627
346965129
<class 'list'>
```

In [0]:

```python
# create a count vectorizer object
count_vect = CountVectorizer(analyzer='word', token_pattern=r'\w{1,}')
count_vect.fit(temp)

# transform the training and validation data using count vectorizer object
xtrain_count =  count_vect.transform(tempp)
```

In [0]:

```
## Create a classifier
import csv
trainDF2 = pandas.DataFrame()

def train_model(classifier, feature_vector_train, label, feature_vector_valid, is_neural_net=False):
    # fit the training dataset on the classifier
    #std_clf = make_pipeline(StandardScaler(with_mean=False), TruncatedSVD(100), MultinominalNB())
    #std_clf.fit(feature_vector_train, label)
    classifier.fit(feature_vector_train, label)

    # predict the labels on validation dataset
    #predictions = classifier.predict(feature_vector_valid)
    predictions = classifier.predict(feature_vector_valid)
    return predictions
    #tt = classifier.predict(feature_vector_valid)
    #labels3 = classifier.predict(feature_vector_valid)

    #trainDF2['labels'] = labels3
    #trainDF2['text']= valid_x
    #print(trainDF2)
```

In [12]:

```
## Read title and abstracts and loop through them
import re
global_list = []
title_list =[]

test = open("stat3.txt",'r').read().split("</abstract>")
#print(test[1])
for idx,i in enumerate(test):
  title = re.findall(r"(?<=<title>).*(?=</title>)",i.replace("\n",""))
  #print(title)
  abstract = re.findall(r"(?<=<abstract>).*",i.replace("\n",""))
  #print(abstract[0].replace("\n",""))
  nlist = re.split(r"(?:(?<=[^i]\.)|\.(?=[^e]))",abstract[0].replace('"',"").replace('\n',''))
  #temp_abs = re.sub(r"((?<=[^i]\.)|\.(?=[^e]))","\n",abstract[0])
  #print(abstract)
  #temp_str = temp_abs.split("\n")
  #print(temp_str[0])
  #print(nlist[1])
  global_list.append(nlist)
  title_list.append(title)
  #print(global_list)

  if idx >50:
    #print(global_list)
    break
  #print(abstract[0])
  #nlist = re.split(r"(?:(?<=[^i]\.)|\.(?=[^e]))",str(abstract))

  #print(nlist[1])

  #tempp1 =[]
  '''
  for idx, item1 in enumerate(nlist):

    if idx > 1 :
      break;
      print(item1)
      tempp1.append(" ".join(item1))
    #print(tempp1)

    xvalid_count =  count_vect.transform(tempp1)
    for item in nlist:
      print(item)
      valid_x = item
      #accuracy = train_model(naive_bayes.MultinomialNB(), xtrain_count, train_y, xvalid_count)

  '''
  #print(global_list[0])
  #print(global_list[1])
  #print(global_list[2])
  #for idx, item1 in enumerate(global_list) :
  #  if idx > 1:
  #    break
  #  print(item1)
    #tempp1.append(" ".join(item1))
    #xvalid_count =  count_vect.transform(tempp1)
    #accuracy = train_model(naive_bayes.MultinomialNB(), xtrain_count, train_y, xvalid_count)
```

```
/usr/lib/python3.6/re.py:212: FutureWarning: split() requires a non-empty pattern match.
  return _compile(pattern, flags).split(string, maxsplit)
```

In [13]:

```
## Print triples from data

#print(global_list[1])
for idx, (item, title) in enumerate(zip(global_list, title_list)):



  #print(item)
  valid_x = item
  xvalid_count =  count_vect.transform(valid_x)
  accuracy = train_model(linear_model.LogisticRegression(), xtrain_count, train_y, xvalid_count)
  #print("\n\n")
  if idx>1:
    break

  title_id = hash(str(title))
  abstract_id = hash(str(item))
  line1 = "<https://w3id.org/skg/articles/" + str(title_id) + "> <http://xmlns.com/foaf/0.1/name>" + '"' + "
 ".join(title) + '"' +"."
  line2 = "<https://w3id.org/skg/articles/" + str(title_id) + "> <http://purl.org/dc/terms/abstract> <http:/
/purl.org/dc/terms/abstract/" + str(abstract_id)+ ">"
  line3 = "<https://w3id.org/skg/articles/" + str(abstract_id) +"><http://purl.org/dc/terms/abstract/text>"
+ '"' + " ".join(item) + '"'
  print(line1,line2,line3,sep ="\n")
  for acc,element in zip(accuracy,item):
    print('<http://purl.org/dc/terms/abstract/{} > "{}"'.format(acc, element))
    #line4 = ("<http://purl.org/dc/terms/abstract/" + str(acc) + ">" + '"' + str(element) + '"' )
```

<https://w3id.org/skg/articles/-7734700797018972234> <http://xmlns.com/foaf/0.1/name>"Reduced b
ias nonparametric lifetime density and hazard estimation".
<https://w3id.org/skg/articles/-7734700797018972234> <http://purl.org/dc/terms/abstract> <http:
//purl.org/dc/terms/abstract/8690684252333690204>
<https://w3id.org/skg/articles/8690684252333690204><http://purl.org/dc/terms/abstract/text>" K
ernel-based nonparametric hazard rate estimation is considered with aspecial class of infinite-
order kernels that achieves favorable bias and meansquare error properties A fully automatic a
nd adaptive implementation of adensity and hazard rate estimator is proposed for randomly right
 censored data Careful selection of the bandwidth in the proposed estimators yields estimatesth
at are more efficient in terms of overall mean squared error performance, andin some cases achi
eves a nearly parametric convergence rate Additionally,rapidly converging bandwidth estimates
are presented for use in second-orderkernels to supplement such kernel-based methods in hazard
rate estimation Simulations illustrate the improved accuracy of the proposed estimator againsto
ther nonparametric estimators of the density and hazard function A real dataapplication is als
o presented on survival data from 13,166 breast carcinomapatients."
<http://purl.org/dc/terms/abstract/MISC > " Kernel-based nonparametric hazard rate estimation
is considered with aspecial class of infinite-order kernels that achieves favorable bias and me
ansquare error properties"
<http://purl.org/dc/terms/abstract/MISC > " A fully automatic and adaptive implementation of ad
ensity and hazard rate estimator is proposed for randomly right censored data"
<http://purl.org/dc/terms/abstract/OWNX > "Careful selection of the bandwidth in the proposed e
stimators yields estimatesthat are more efficient in terms of overall mean squared error perfor
mance, andin some cases achieves a nearly parametric convergence rate"
<http://purl.org/dc/terms/abstract/OWNX > " Additionally,rapidly converging bandwidth estimates
 are presented for use in second-orderkernels to supplement such kernel-based methods in hazard
 rate estimation"
<http://purl.org/dc/terms/abstract/OWNX > "Simulations illustrate the improved accuracy of the
proposed estimator againstother nonparametric estimators of the density and hazard function"
<http://purl.org/dc/terms/abstract/OWNX > " A real dataapplication is also presented on surviva
l data from 13,166 breast carcinomapatients."
<https://w3id.org/skg/articles/1711393881944931099> <http://xmlns.com/foaf/0.1/name>"Quantile a
nd Probability Curves Without Crossing".
<https://w3id.org/skg/articles/1711393881944931099> <http://purl.org/dc/terms/abstract> <http:/
/purl.org/dc/terms/abstract/-2721053281257085775>
<https://w3id.org/skg/articles/-2721053281257085775><http://purl.org/dc/terms/abstract/text>"
This paper proposes a method to address the longstanding problem of lack ofmonotonicity in esti
mation of conditional and structural quantile functions,also known as the quantile crossing pro
blem The method consists in sorting ormonotone rearranging the original estimated non-monotone
 curve into a monotonerearranged curve We show that the rearranged curve is closer to the true
quantile curve in finite samples than the original curve, establish afunctional delta method fo
r rearrangement-related operators, and derivefunctional limit theory for the entire rearranged
curve and its functionals Wealso establish validity of the bootstrap for estimating the limit
law of thethe entire rearranged curve and its functionals Our limit results are genericin that
 they apply to every estimator of a monotone econometric function,provided that the estimator s
atisfies a functional central limit theorem andthe function satisfies some smoothness condition
s Consequently, our resultsapply to estimation of other econometric functions with monotonicit
yrestrictions, such as demand, production, distribution, and structuraldistribution functions
We illustrate the results with an application toestimation of structural quantile functions usi
ng data on Vietnam veteranstatus and earnings."
<http://purl.org/dc/terms/abstract/OWNX > " This paper proposes a method to address the longst
anding problem of lack ofmonotonicity in estimation of conditional and structural quantile func
tions,also known as the quantile crossing problem"
<http://purl.org/dc/terms/abstract/OWNX > " The method consists in sorting ormonotone rearrangi
ng the original estimated non-monotone curve into a monotonerearranged curve"
<http://purl.org/dc/terms/abstract/OWNX > " We show that the rearranged curve is closer to the
truequantile curve in finite samples than the original curve, establish afunctional delta metho
d for rearrangement-related operators, and derivefunctional limit theory for the entire rearran
ged curve and its functionals"
<http://purl.org/dc/terms/abstract/OWNX > " Wealso establish validity of the bootstrap for esti
mating the limit law of thethe entire rearranged curve and its functionals"
<http://purl.org/dc/terms/abstract/OWNX > " Our limit results are genericin that they apply to
every estimator of a monotone econometric function,provided that the estimator satisfies a func
tional central limit theorem andthe function satisfies some smoothness conditions"
<http://purl.org/dc/terms/abstract/MISC > " Consequently, our resultsapply to estimation of oth
er econometric functions with monotonicityrestrictions, such as demand, production, distributio
n, and structuraldistribution functions"
<http://purl.org/dc/terms/abstract/OWNX > " We illustrate the results with an application toest
imation of structural quantile functions using data on Vietnam veteranstatus and earnings."