In [1]:
```
from sklearn.pipeline import Pipeline
from sklearn.neural_network import MLPRegressor, MLPClassifier
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [2]:
```
import pandas as pd, numpy as np, json, re, pickle

from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, roc_auc_sco
re, auc, precision_recall_fscore_support
from sklearn.metrics import classification_report
from sklearn.neural_network import MLPClassifier
```

In [3]:
```
def documents( corpus):
    return list( corpus.reviews())

def continuous( corpus):
    return list( corpus.scores())

def make_categorical( corpus):
    """
    terrible : 0.0 < y < = 3.0
    okay : 3.0 < y < = 5.0
    great : 5.0 < y < = 7.0
    amazing : 7.0 < y < = 10.1
    """
    return np.digitize( continuous( corpus), [0.0, 3.0, 5.0, 7.0, 10.1])
```

In [4]:

```
#from sklearn.externals import joblib
from sklearn.model_selection import cross_val_score

def train_model( data, model, continuous = True, saveto = None, cv = 12):
    """
    Trains model from corpus at specified path; constructing cross-valida
tion
    scores using the cv parameter, then fitting the model on the full dat
a. Returns the scores.
    """
    # Load the corpus data and labels for classification

    X = documents( corpus)


    if continuous:
        y = continuous( corpus)
        scoring = 'r2_score'
    else:
        y = make_categorical( corpus)
        scoring = 'f1_score'

    # Compute cross-validation scores
    scores = cross_val_score( model, X, y, cv = cv, scoring = scoring)

    # Write to disk if specified

    if saveto: joblib.dump( model, saveto)

    # Fit the model on entire dataset

    model.fit( X, y)

    # Return scores

    return scores
```

In [5]:
```
# convert to lower case
def to_lower(string: str) -> str:
    return string.lower()

# Load libraries
import unicodedata
import sys

def remove_puncs(text):

    # Create a dictionary of punctuation characters
    #punctuation = dict.fromkeys(i for i in range(sys.maxunicode)
                                #if unicodedata.category(chr(i)).startswith
('P'))

    #text = text.translate(punctuation)
    text=re.sub('</?.*?>',' <>', text)
    text=re.sub('\\d|\\W+|_',' ',text)
    text=re.sub('[^a-zA-Z]'," ", text)

    return text
```

# Preprocessing

In [6]:
```
file ='controversial-comments.jsonl'
# possible orient value: split, records, index, columns, and values)
# The following file is a subset of above file with the 1st 233537 lines.
comments = pd.read_json("controversial-comments_small.jsonl",orient="colu
mns",lines=True)
comments.head()
```

Out[6]:

| | con | txt |
|---|---|---|
| **0** | 0 | Well it's great that he did something about th... |
| **1** | 0 | You are right Mr. President. |
| **2** | 0 | You have given no input apart from saying I am... |
| **3** | 0 | I get the frustration but the reason they want... |
| **4** | 0 | I am far from an expert on TPP and I would ten... |

In [7]:
```
print('Size: ', len(comments), '\n',
      'Shape: ', comments.info(), '\n',
      'Categories: ', comments.con.unique())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 233538 entries, 0 to 233537
Data columns (total 2 columns):
 #   Column  Non-Null Count   Dtype 
---  ------  --------------   ----- 
 0   con     233538 non-null  int64 
 1   txt     233538 non-null  object
dtypes: int64(1), object(1)
memory usage: 3.6+ MB
Size:  233538
 Shape:  None
 Categories:  [0 1]
```

In [8]:
```
# since the size is humongus, I will take sample of the 2 categories.
# by trial, sample of 50000 from each category can be easily handled by m
y machine

size = 50000    # sample size
replace = True  # with replacement
fn = lambda obj: obj.loc[np.random.choice(obj.index, size, replace),:]

controversy = comments.groupby('con', as_index=False).apply(fn)
del comments
controversy.shape
```

Out[8]:
```
(100000, 2)
```

In [9]:
```
controversy['txt'] = controversy['txt'].apply(lambda x:to_lower(x))
controversy['txt'] = controversy['txt'].apply(lambda x:remove_puncs(x))

controversy.reset_index(drop=True, inplace=True)

controversy.head()
```

Out[9]:

| | con | txt |
|---|---|---|
| **0** | 0 | i shit you not i ve had donald supporters tell... |
| **1** | 0 | deleted |
| **2** | 0 | have to say i agree with this i want to know b... |
| **3** | 0 | i guess the perfect example of this is the per... |
| **4** | 0 | it was one of his main campaign promises there... |

In [10]:
```
controversy['txt'] = [word_tokenize(string) for string in controversy['txt']]
controversy.head()
```

Out[10]:

| | con | txt |
|---|---|---|
| **0** | 0 | [i, shit, you, not, i, ve, had, donald, suppor... |
| **1** | 0 | [deleted] |
| **2** | 0 | [have, to, say, i, agree, with, this, i, want,... |
| **3** | 0 | [i, guess, the, perfect, example, of, this, is... |
| **4** | 0 | [it, was, one, of, his, main, campaign, promis... |

In [11]:
```
# Load stop words
stop_words = stopwords.words('english')
controversy['txt'] = controversy['txt'].apply(lambda x: [item for item in x if item not in stop_words])
controversy.head()
```

Out[11]:

| | con | txt |
|---|---|---|
| **0** | 0 | [shit, donald, supporters, tell, pro, gay, inv... |
| **1** | 0 | [deleted] |
| **2** | 0 | [say, agree, want, know, sides, echo, chamber,... |
| **3** | 0 | [guess, perfect, example, pervasiveness, trump... |
| **4** | 0 | [one, main, campaign, promises, dozens, videos... |

In [13]:
```
controversy.to_pickle('./Data/pkld_controversy.pkl')
```

In [ ]:
```
```

In [ ]:
```
```