In [1]:

```
# Change directory to VSCode workspace root so that relative path loads work correctly. Turn this addition off with the DataScience.changeDirOnImportExport setting
import os
try:
        os.chdir(os.path.join(os.getcwd(), 'Python'))
        print(os.getcwd())
except:
        pass
```

```
C:\Users\jonat\Documents\Masters\DSC 550 - Data Mining\Python
```

In [3]:

```
import numpy as np
import pandas as pd
import re
import string

import keras
#import keras_metrics as km
import tensorflow as tf
from pathlib import Path
from pandas import Series, DataFrame
from sklearn.externals import joblib

from sklearn.neural_network import MLPRegressor, MLPClassifier
from sklearn.feature_extraction.text import TfidfVectorizer
from keras.wrappers.scikit_learn import KerasClassifier
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras import backend as K
from keras.utils import np_utils

from nltk import wordpunct_tokenize
from nltk.corpus import stopwords

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import GridSearchCV, cross_validate, cross_val_score, train_test_split, StratifiedKFold, cross_val_predict
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.preprocessing import StandardScaler, LabelEncoder

from keras.datasets import mnist
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras import backend as K
```

In [4]:

```
base_path = Path('../Python/data/reddit/')
path_categories = base_path / 'categorized-comments.jsonl'
```

In [5]:

```
df_cat = pd.read_json(path_categories, lines=True, orient='columns') # Multiclass dataset
```

In [6]:

```
df = df_cat.sample(frac=0.001)
```

In [7]:

```
df['cat'].unique()

# encode class values as integers
encoder = LabelEncoder()
encoder.fit(df['cat'])
encoded_Y = encoder.transform(df['cat'])
# convert integers to dummy variables (i.e. one hot encoded)
dummy_y = np_utils.to_categorical(encoded_Y)
```

In [8]:

```
def my_preprocessor(text):
    text = re.sub("\S*\d\S*", "", text).strip() # Strip out any numbers
    text = text.translate(str.maketrans('','', string.punctuation)) # Strip out pun
ctuation
    return text.lower() # Return lowercase values
```

In [9]:

```
# Create Vector model of categories dataset
cv_cat = TfidfVectorizer(preprocessor=my_preprocessor,
                    stop_words='english')
vectors_cv_cat = cv_cat.fit_transform(df['txt'])
```

In [10]:

```
vectors_cv_cat.todense().shape[1]
```

Out[10]:

```
8332
```

In [11]:

```
models = ['MLPClassifier']
columnnames = ['Model']

# Set scoring tests to run
scoring = {'Accuracy': 'accuracy',
               'Precision': 'precision_macro',
               'Recall': 'recall_macro',
               'F1': 'f1_macro'}
```

In [12]:

```
# Create a DataFrame which uses the models and datasets variables as initial values
df_results = pd.DataFrame(columns=columnnames)
df_results['Model'] = models

# Create empty DataFrame columns for test results
for k, v in scoring.items():
    df_results[k] = ""
```

In [13]:

```
N_FEATURES = vectors_cv_cat.todense().shape[1]
N_CLASSES = 4

def insert_results(results, modelname, df):
    """Insert results of validation tests into a dataframe

    Args:
        results (dict): Dictionary results from validate_results()
        modelname (str): Name of the model we are testing
        df (dataframe): Dataframe we are inserting results into.
    """
    for column, test in scoring.items():
        df.loc[(df['Model'] == modelname),column] = np.mean(results['test_' + test])

def validate_results(model, features, target, scoring, cv=2):
    """Run validation tests for a given model and provide scoring results.

    Args:
        model (model): Model function, e.g., MultinomialNB()
        features (matrix): Matrix of features
        target (array): array of target values
        scoring (dict): Dictionary of scoring methods to return.
        cv (int, optional): Number of cross validation folds. Defaults to 3.

    Returns:
        dict: Dictionary of scoring results.
    """
    cv_results = cross_validate(model,
                features,
                target,
                cv=cv,
                scoring=list(scoring.values()),
                n_jobs=-1,
                return_train_score=False)
    return cv_results

# create model
def create_model():
    model = Sequential()
    model.add(Dense(500, activation='relu', input_shape=(N_FEATURES,)))
    model.add(Dense(150, activation='relu'))
    model.add(Dense(N_CLASSES, activation='softmax'))

    # Compile model
    model.compile(loss='categorical_crossentropy', optimizer='rmsprop', metrics=['accuracy'])

    #model.fit(X_train, y_train, epochs=150, batch_size=10, verbose=0)

    return model
```

In [14]:

```
MLP_results = validate_results(MLPClassifier(hidden_layer_sizes=[500, 150], verbose=True),
                                 vectors_cv_cat,
                                 df['cat'],
                                 scoring)
```

In [15]:

```
insert_results(MLP_results, 'MLPClassifier', df_results)
```

In [16]:
```
df_results
```

Out[16]:

	Model	Accuracy	Precision	Recall	F1
0	MLPClassifier	0.528757	0.486375	0.385112	0.386993

In [17]:

```
y_pred = cross_val_predict(MLPClassifier(hidden_layer_sizes=[500, 150], verbose=True),
                           vectors_cv_cat,
                           df['cat'],
                           cv=3)
```

```
Iteration 1, loss = 1.33585712
Iteration 2, loss = 1.18810724
Iteration 3, loss = 1.05777524
Iteration 4, loss = 0.88433685
Iteration 5, loss = 0.67159739
Iteration 6, loss = 0.46861778
Iteration 7, loss = 0.32154177
Iteration 8, loss = 0.22990916
Iteration 9, loss = 0.17218759
Iteration 10, loss = 0.13976977
Iteration 11, loss = 0.12134324
Iteration 12, loss = 0.11354554
Iteration 13, loss = 0.10886636
Iteration 14, loss = 0.10728182
Iteration 15, loss = 0.10575724
Iteration 16, loss = 0.10606593
Iteration 17, loss = 0.10614762
Iteration 18, loss = 0.10455946
Iteration 19, loss = 0.10317087
Iteration 20, loss = 0.10319957
Iteration 21, loss = 0.10268554
Iteration 22, loss = 0.10367511
Iteration 23, loss = 0.10295088
Iteration 24, loss = 0.10301576
Iteration 25, loss = 0.10198731
Iteration 26, loss = 0.10350602
Iteration 27, loss = 0.10303513
Iteration 28, loss = 0.10575203
Iteration 29, loss = 0.10516754
Iteration 30, loss = 0.10177056
Iteration 31, loss = 0.10302135
Iteration 32, loss = 0.10237974
Iteration 33, loss = 0.10323765
Iteration 34, loss = 0.10311555
Iteration 35, loss = 0.10227349
Iteration 36, loss = 0.10287395
Iteration 37, loss = 0.10161480
Iteration 38, loss = 0.10362822
Iteration 39, loss = 0.10328037
Iteration 40, loss = 0.10048534
Iteration 41, loss = 0.10297032
Iteration 42, loss = 0.10215868
Iteration 43, loss = 0.10144068
Iteration 44, loss = 0.10134544
Iteration 45, loss = 0.10285610
Iteration 46, loss = 0.10142437
Iteration 47, loss = 0.10158242
Iteration 48, loss = 0.10237468
Iteration 49, loss = 0.10222916
Iteration 50, loss = 0.10237243
Iteration 51, loss = 0.10116248
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs.
Stopping.
Iteration 1, loss = 1.33342269
Iteration 2, loss = 1.20007621
Iteration 3, loss = 1.06699365
Iteration 4, loss = 0.90571730
Iteration 5, loss = 0.69846872
Iteration 6, loss = 0.49267753
Iteration 7, loss = 0.34061307
Iteration 8, loss = 0.24466722
Iteration 9, loss = 0.18616183
Iteration 10, loss = 0.15072582
```

In [18]:
```
conf_mat = confusion_matrix(df['cat'], y_pred)
```

In [19]:
```
conf_mat
```

Out[19]:
```
array([[143,   5,  82, 158],
       [ 33,   8,  31,  80],
       [ 29,   6, 455, 273],
       [ 59,   9, 293, 683]], dtype=int64)
```

In [20]:
```
model = KerasClassifier(build_fn=create_model, epochs=150, batch_size=10, verbos
e=0)
```

In [21]:
```
# evaluate using 3-fold cross validation
kfold = StratifiedKFold(n_splits=3, shuffle=True, random_state=7)
results = cross_val_score(model, vectors_cv_cat, df['cat'], cv=kfold)
```

```
WARNING:tensorflow:From C:\Users\jonat\Anaconda3\lib\site-packages\tensorflow\py
thon\framework\op_def_library.py:263: colocate_with (from tensorflow.python.fram
ework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From C:\Users\jonat\Anaconda3\lib\site-packages\tensorflow\py
thon\ops\math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is dep
recated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
```

In [24]:
```
results.mean()
```

Out[24]:
```
0.5215361132383879
```

In [25]:
```
y_pred = cross_val_predict(model,
                           vectors_cv_cat,
                           df['cat'],
                           cv=3)
```

In [26]:
```
conf_mat = confusion_matrix(df['cat'], y_pred)
```

In [27]:
```
conf_mat
```

Out[27]:
```
array([[124,  16,  94, 154],
       [ 28,  14,  34,  76],
       [ 33,  10, 449, 271],
       [ 62,  15, 304, 663]], dtype=int64)
```

In [28]:
```
report = classification_report(df['cat'], y_pred)
print(report)
```

```
                        precision    recall  f1-score   support

                  news       0.50      0.32      0.39       388
science_and_technology       0.25      0.09      0.14       152
                sports       0.51      0.59      0.55       763
           video_games       0.57      0.64      0.60      1044

             micro avg       0.53      0.53      0.53      2347
             macro avg       0.46      0.41      0.42      2347
          weighted avg       0.52      0.53      0.52      2347
```

In [29]:

```
# Set that the color channel value will be first
K.set_image_data_format("channels_first")

# Set seed
np.random.seed(0)

# Set image information
channels = 1
height = 28
width = 28

# Load data and target from MNIST data
(data_train, target_train), (data_test, target_test) = mnist.load_data()
```

In [30]:

```
# Reshape training image data into features
data_train = data_train.reshape(data_train.shape[0], channels, height, width)

# Reshape test image data into features
data_test = data_test.reshape(data_test.shape[0], channels, height, width)

# Rescale pixel intensity to between 0 and 1
features_train = data_train / 255
features_test = data_test / 255

# One-hot encode target
target_train = np_utils.to_categorical(target_train)
target_test = np_utils.to_categorical(target_test)
number_of_classes = target_test.shape[1]
```

In [31]:

```
# Start neural network
network = Sequential()

# Add convolutional layer with 64 filters, a 5x5 window, and ReLU activation functi
on
network.add(Conv2D(filters=64,
                   kernel_size=(5, 5),
                   input_shape=(channels, width, height),
                   activation='relu'))

# Add max pooling layer with a 2x2 window
network.add(MaxPooling2D(pool_size=(2, 2)))

# Add dropout layer
network.add(Dropout(0.5))

# Add layer to flatten input
network.add(Flatten())

# # Add fully connected layer of 128 units with a ReLU activation function
network.add(Dense(128, activation="relu"))

# Add dropout layer
network.add(Dropout(0.5))

# Add fully connected layer with a softmax activation function
network.add(Dense(number_of_classes, activation="softmax"))
```

```
WARNING:tensorflow:From C:\Users\jonat\Anaconda3\lib\site-packages\keras\backen
d\tensorflow_backend.py:3445: calling dropout (from tensorflow.python.ops.nn_op
s) with keep_prob is deprecated and will be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep
_prob`.
```

In [32]:

```
# Compile neural network
network.compile(loss="categorical_crossentropy", # Cross-entropy
                optimizer="rmsprop", # Root Mean Square Propagation
                metrics=["accuracy"]) # Accuracy performance metric

# Train neural network
network.fit(features_train, # Features
            target_train, # Target
            epochs=2, # Number of epochs
            verbose=0, # Don't print description after each epoch
            batch_size=1000, # Number of observations per batch
            validation_data=(features_test, target_test)) # Data for evaluation
```

Out[32]:

```
<keras.callbacks.History at 0x266396251d0>
```

In [ ]: