ASSIGNMENT 4

Date	**31 October 2022**
Team ID	PNT2022TMID41278
Project Name	Emerging Methods for Early Detection of Forest Fires

```
Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1BuzOmo9BAXkrsPk9gOKeCV1JA_rd4_Vy

**Import the required libraries**
"""

# Commented out IPython magic to ensure Python compatibility.
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style("darkgrid")
# %matplotlib inline

import string
import nltk
from nltk.corpus import stopwords

from wordcloud import WordCloud
from sklearn.feature_extraction.text import CountVectorizer
from nltk.stem import WordNetLemmatizer

from sklearn.model_selection import train_test_split
from sklearn import metrics

"""**Read** **the Dataset**

"""

messages = pd.read_csv('../content/spam.csv',encoding = 'latin-1')
messages.head()

"""**Preprocessing the data**

"""

messages.isnull()

messages.isnull().sum()

messages.isnull().sum()/len(messages)

messages.dropna(axis = 1)

messages.fillna(0)

messages.mean()

messages.mode()

messages.median()

messages.std()

messages.describe()

messages.describe(include=['object'])
```

```
messages.describe(include='all')

messages.drop(["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"], axis=1, inplace=True)
```

```
messages.columns = ["SpamHam","Tweet"]

sns.countplot(messages["SpamHam"])
```

```
import nltk
nltk.download('stopwords')

from sklearn.preprocessing import LabelEncoder
lb_enc = LabelEncoder()
y = lb_enc.fit_transform(messages["SpamHam"])

import pandas as pd
import numpy as np
import re
import collections
import contractions
import seaborn as sns
import matplotlib.pyplot as plt
plt.style.use('dark_background')
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
import warnings
warnings.simplefilter(action='ignore', category=Warning)
import keras
from keras.layers import Dense, Embedding, LSTM, Dropout
from keras.models import Sequential
from keras.preprocessing.text import Tokenizer

import pickle
```

```
pip install contractions
```

```
pip install pad_sequences
```

```
from sklearn.preprocessing import LabelEncoder
lb_enc = LabelEncoder()
y = lb_enc.fit_transform(messages["SpamHam"])

tokenizer = Tokenizer() #initializing the tokenizer
tokenizer.fit_on_texts(messages)# fitting on the sms data
text_to_sequence = tokenizer.texts_to_sequences(messages)

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list the files in
the input directory

# we need to fit model with sequence of tokens with specific lengt
from keras.preprocessing.text import Tokenizer
from keras.models import Sequential
# normal LSTM/GRU and the Version with Cuda
from keras.layers import Dense, Embedding, GRU, LSTM, Dropout, Bidirectional
from keras.callbacks import TensorBoard, EarlyStopping, ModelCheckpoint


# keras wrapper for k-fold cross-validation
from keras.wrappers.scikit_learn import KerasClassifier
# normsl cross validation
from sklearn.model_selection import cross_val_score, train_test_split
# cross validation for hyperparameter tuning
from sklearn.model_selection import GridSearchCV
```

```
x_raw = []
y_raw = []

with open("spam.csv", encoding = "ISO-8859-1") as f:
    for line in f:
        y_raw.append(line.split()[0])
        x_raw.append(' '.join(i for i in line.split()[1:]))

y = [1 if i=='ham' else 0 for i in y_raw]
print(max(len(s) for s in x_raw))
print(min(len(s)for s in x_raw))
sorted_X = sorted(len(s) for s in x_raw)
print(sorted_X[len(sorted_X) // 2])

tokenizer = Tokenizer()
tokenizer.fit_on_texts(x_raw)
sequences = tokenizer.texts_to_sequences(x_raw)

vocab_size = len(tokenizer.word_index)+1
print(vocab_size)

sum([len(x) for x in sequences])

X_train, X_test, y_train, y_test = train_test_split(sequences, y, test_size = 0.2,
random_state= 0)

"""**Create** **model**"""

model = Sequential()
model.add(Embedding(input_dim=vocab_size, output_dim=embedding_size, input_length=max_len))
model.add(Dropout(0.8))
model.add(LSTM(140, return_sequences=False))
model.add(Dropout(0.8))
model.add(Dense(1, activation='sigmoid', name='Classification'))
model.summary()

"""**Add Layers (LSTM, Dense-(Hidden Layers), Output)**"""

#LSTM hyperparameters
n_lstm = 20
drop_lstm =0.2

#LSTM Spam detection architecture
model1 = Sequential()
model1.add(LSTM(n_lstm, dropout=drop_lstm, return_sequences=True))
model1.add(LSTM(n_lstm, dropout=drop_lstm, return_sequences=True))
model1.add(Dense(1, activation='sigmoid'))

model1.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics=['accuracy'])

"""**Hidden Layer**"""

model.add(Dense(300,activation='relu'))
model.add(Dense(150,activation='relu'))

"""**Output Layer**"""

model.add(Dense(4,activation='softmax'))
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
len(X_train)

"""Text Preprocessing

"""

messages.head()
```

```
messages.keys

def remove_url(text):
    re_url = re.compile('https?://\S+|www\.\S+')
    return re_url.sub('', text)
messages['SpamHam'] = messages['SpamHam'].apply(remove_url)

def remove_url(text):
    re_url = re.compile('https?://\S+|www\.\S+')
    return re_url.sub('', text)

messages['SpamHam'] = messages['SpamHam'].apply(remove_url)

exclude = string.punctuation

def remove_punc(text):
    return text.translate(str.maketrans('', '', exclude))

messages['SpamHam'] = messages['SpamHam'].apply(remove_punc)

X = messages["SpamHam"]
y = messages['SpamHam'].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size= 0.2, random_state= 42,
stratify = y)

"""**Compile The Model**"""

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

"""**Fit the model**"""

def train_model(model):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    y_prob = model.predict_proba(X_test)
    accuracy = round(accuracy_score(y_test, y_pred), 3)
    precision_score = round(precision_score(y_test, y_pred), 3)
    recall = round(recall_score(y_test, y_pred), 3)

    print(f'Accuracy of the model: {accuracy}')
    print(f'Precision Score of the model: {precision}')
    print(f'Recall Score of the model: {recall}')

    sns.set_context('notebook', font_scale= 1.3)
    fig, ax = plt.subplots(1, 2, figsize = (25,  8))
    ax1 = plot_confusion_matrix(y_test, y_pred, ax= ax[0], cmap= 'YlGnBu')
    ax2 = plot_roc(y_test, y_prob, ax= ax[1], plot_macro= False, plot_micro= False, cmap=
'summer')

from keras.models import Sequential
from keras.layers import Dense
from sklearn.datasets import make_blobs
from sklearn.preprocessing import MinMaxScaler
# generate 2d classification dataset
x, y = make_blobs(n_samples=100, centers=2, n_features=2, random_state=1)
scalar = MinMaxScaler()
scalar.fit(x)
x = scalar.transform(x)
# define and fit the final model
model = Sequential()
model.add(Dense(4, input_shape=(2,), activation='relu'))
model.add(Dense(4, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam')
model.fit(x, y, epochs=10, verbose=5)
```

```
"""**Save the model**"""

ls model.save('spam.h5')

"""**Test the model**"""

ls

clf = MultinomialNB()
clf.fit(X_train,y_train)
print("Accuracy of Model",clf.score(X_test,y_test)*100,"%")
```