In [1]:
```
# Danny: FastText
```

In [1]:
```
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.model_selection import train_test_split
```

In [2]:
```
# Connect with drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [3]:
```
#Read dataset
folder_path = '/content/drive/MyDrive/COMP 652/Final Project/Colab Notebooks
dataset = pd.read_csv(folder_path)
# dataset
```

In [4]:
```
#Rename column Unnamed: 0 to Id
dataset = dataset.rename(columns={"Unnamed: 0": "Id"})

#Drop the null value rows
dataset.dropna()

#Find the unique column values and set the labels for the same.
col_values= dataset['status'].unique()

# Create a dictionary to map unique status values to numbers
status_mapping = {status: i for i, status in enumerate(col_values)}

# Create the new column 'status_numeric' based on the mapping
dataset['labels'] = dataset['status'].map(status_mapping)
```

In [5]:
```
import pandas as pd
import re
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer

# Download required NLTK resources (only once)
nltk.download('punkt_tab')
nltk.download('stopwords')
nltk.download('wordnet')

# Initialize preprocessing tools
stop_words = set(stopwords.words('english'))
stemmer = PorterStemmer()
lemmatizer = WordNetLemmatizer()

# Function to preprocess text
def preprocess(text):
```

```
    if isinstance(text, str):
      text = text.lower()     # Lowercase
      text = re.sub(r'[^\w\s]', '', text)     # Remove punctuation
      tokens = word_tokenize(text)     # Tokenize
      tokens = [t for t in tokens if t not in stop_words]     # Remove stopw
      tokens = [stemmer.stem(t) for t in tokens]     # Apply stemming
      tokens = [lemmatizer.lemmatize(t) for t in tokens]     # Apply lemmati
      return ' '.join(tokens)     # Apply lemmatization
    else:
      return text

# Apply to DataFrame
dataset['clean_text'] = dataset['statement'].apply(preprocess)

dataset.head()
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
```

Out[5]:

	Id	statement	status	labels	clean_text
0	0	oh my gosh	Anxiety	0	oh gosh
1	1	trouble sleeping, confused mind, restless hear...	Anxiety	0	troubl sleep confus mind restless heart tune
2	2	All wrong, back off dear, forward doubt. Stay ...	Anxiety	0	wrong back dear forward doubt stay restless re...
3	3	I've shifted my focus to something else but I'...	Anxiety	0	ive shift focu someth el im still worri
4	4	I'm restless and restless, it's been a month n...	Anxiety	0	im restless restless month boy mean

FastText

In [6]:
```
!pip install fasttext
```

```
Collecting fasttext
  Downloading fasttext-0.9.3.tar.gz (73 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0.0/73.4 kB ? eta -:--:--
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━- 71.7/73.4 kB 2.3 MB/s eta 0:0
0:01
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 73.4/73.4 kB 1.8 MB/s eta 0:0
0:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Collecting pybind11>=2.2 (from fasttext)
  Using cached pybind11-2.13.6-py3-none-any.whl.metadata (9.5 kB)
Requirement already satisfied: setuptools>=0.7.0 in /usr/local/lib/python3.1
1/dist-packages (from fasttext) (75.2.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packa
ges (from fasttext) (2.0.2)
Using cached pybind11-2.13.6-py3-none-any.whl (243 kB)
Building wheels for collected packages: fasttext
  Building wheel for fasttext (pyproject.toml) ... done
  Created wheel for fasttext: filename=fasttext-0.9.3-cp311-cp311-linux_x86_
64.whl size=4313503 sha256=f768a6f72143a3c31cd5a3619e32b6cbf6026c69b809294d9
714d3391eed8144
  Stored in directory: /root/.cache/pip/wheels/65/4f/35/5057db0249224e9ab55a
513fa6b79451473ceb7713017823c3
Successfully built fasttext
Installing collected packages: pybind11, fasttext
Successfully installed fasttext-0.9.3 pybind11-2.13.6
```

In [7]:
```
import fasttext
from sklearn.model_selection import train_test_split, StratifiedKFold
```

In [17]:
```
# dataset['labeled_data'] = '__label__' + dataset['status'].astype(str) + '
# concatenated_dataset = '__label__' + dataset['status'].astype(str) + ' ' +
# concatenated_dataset = concatenated_dataset.dropna()
# dataset = dataset.dropna(subset=['labeled_data'])
```

In [46]:
```
# preprocess data
fasttext_dataset = dataset.dropna(subset=['clean_text']) # drop where clean_
fasttext_dataset = fasttext_dataset.dropna(subset=['status']) # drop where s
fasttext_dataset['clean_text'] = fasttext_dataset['clean_text'].astype(str).
fasttext_dataset['status'] = fasttext_dataset['status'].astype(str).replace(
fasttext_dataset = fasttext_dataset[fasttext_dataset['clean_text'] != ''] #

# declare the data and its labels
x = fasttext_dataset['clean_text']
y = fasttext_dataset['status']

# split the training and test datasets. 80% for training, 20% for test
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, ran
```

In [9]:
```
# function to flatten nested lists. used for y_predictions
def flatten_list(x:list):
    lst = []
    for element in x:
```

```
            lst.extend(flatten_list(element))
        else:
            lst.append(element)
    return lst
```

In [ ]:
```
# !pip install fasttext
```

In [54]:
```
# Hyperparameters
# 1st item: learning rate
# 2nd item: wordNgrams
# 3rd item: loss function (ns = negative sampling, hs = hierarchical softmax
hyperparameters = {
    0:[.1, 1, 'ns'],
    1:[.5, 1, 'ns'],
    2:[1.0, 1, 'ns'],
    3:[.1, 2, 'ns'],
    4:[.5, 2, 'ns'],
    5:[1.0, 2, 'ns'],
    6:[.1, 3, 'ns'],
    7:[.5, 3, 'ns'],
    8:[1.0, 3, 'ns'],
    9:[.1, 1, 'hs'],
    10:[.5, 1, 'hs'],
    11:[1.0, 1, 'hs'],
    12:[.1, 2, 'hs'],
    13:[.5, 2, 'hs'],
    14:[1.0, 2, 'hs'],
    15:[.1, 3, 'hs'],
    16:[.5, 3, 'hs'],
    18:[1.0, 3, 'hs'],
}
```

In [ ]:
```
import fasttext
from sklearn.model_selection import train_test_split, StratifiedKFold

kfolds = StratifiedKFold(n_splits=18, shuffle=True, random_state=42)

accuracies_negativesampling = []
accuracies_hierarchicalsoftmax = []

for idx, (train_index, val_index) in enumerate(kfolds.split(x_train, y_train

    # training sets
    x_train_fold = x_train.iloc[train_index]
    y_train_fold = y_train.iloc[train_index]

    # validation sets
    x_val_fold =  x_train.iloc[val_index]
    y_val_fold = y_train.iloc[val_index]

    # concatenate the y and x data to match the fasttext supervised paramete
    fasttext_model_data = '__label__' + y_train.astype(str) + ' ' + x_train.

    # substitute any NaN values with ''
    fasttext_model_data = fasttext_model_data.fillna('')
```

```
    # output a text file of the fasttext training data, since the fasttext m
    fasttext_model_data.to_csv('/content/drive/MyDrive/COMP 652/Final Projec

    # select parameters for hypteruning
    learning_rate = hyperparameters[idx][0]
    wordNgram = hyperparameters[idx][1]
    loss_function = hyperparameters[idx][2]

    # train the fasttext model with the text file
    fasttext_model = fasttext.train_supervised(input = '/content/drive/MyDri
                                               lr = learning_rate,
                                               wordNgrams = wordNgram,
                                               loss = loss_function)

    # get predicted labels
    y_predictions, _ = fasttext_model.predict(list(x_val_fold))
    y_predictions = flatten_list(y_predictions)
    y_predictions = [prediction.replace('__label__', '') for prediction in y

    # calculate the accuracy and append results to accuracies variable
    num_correct = np.sum(y_predictions == y_val_fold) # number of matches fo
    accuracy = float(num_correct) / len(y_val_fold) # calculate the accuracy

    if idx < 9:
      accuracies_negativesampling.append(accuracy)
    else:
      accuracies_hierarchicalsoftmax.append(accuracy)

    print(f"Fold {idx+1} learning rate={learning_rate}, wordNgram={wordNgram
```

```
0
Fold 1 learning rate=0.1, wordNgram=1, loss_function=ns, accuracy=0.8189
1
```

In [ ]:
```
# plot the accuracies for each loss function type
# negative sampling
plt.plot(accuracies_negativesampling)
plt.show()

# hierarchical softmax
plt.plot(accuracies_hierarchicalsoftmax)
plt.show()
```

In [ ]:
```
from sklearn.metrics import roc_auc_score
# Plot AUROC
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

# calculate confusion matrix
cm = confusion_matrix(y_val_fold, y_prediction)
print(f"Confusion matrix (cm): \n {cm}")

# calculate the ROC curve
false_pos_rate, true_pos_rate, thresholds = roc_curve(y_val_fold, y_predicti
roc_auc = auc(false_pos_rate, true_pos_rate)
# plot the ROC curve
```

```
plt.figure(figsize=(8, 6))
plt.plot(false_pos_rate, true_pos_rate, color="blue", lw=2, label=f"ROC curv
plt.xlabel("False Positive Rate", fontsize=12)
plt.ylabel("True Positive Rate", fontsize=12)
plt.title("ROC Curve", fontsize=12)
plt.legend(loc="lower right", fontsize=12)
plt.show()
```