1. **Requirements:** In order to run the source code, Python 3.7 and necessary packages need to be installed. The commands to install packages are in `install.sh`.

2. **Classical models**
   Source code: classical/classical_clf.py
   Explanation:
   - Text processing: convert all text into lowercase, tokenize the tweet sentence into words. Remove stopwords and non-alpha words. For each word, find its post of speech using `nltk` and we map it to a root work by using a `WordNetLemmatizer`. Some data needs to be downloaded for use with `nltk`.
   - Vectorization: The text in tweets need to be encoded as numbers for use as input of models. Each tweet is encoded by a vector using `TfidfVectorizer`. `LabelEncoder` is used for encoding intensity class.
   - After fitting X and Y vectors from training data for each algorithm, we use those models to predict new X in testing data.

   Run: cd to `classical` folder, run the command: `python3 classical_clf.py`. The result is printed to standard output. A sample result is available in `output.txt` in the same folder.

3. **Deep learning models**
   Source code: classical/classical_clf.py
   Explanation:
   - Preprocessing text is similar to the process of building classical models.
   - Word embeddings: Each word is represented by an array of D numbers representing its semantic. If V is the vocabulary, then we need a matrix of size |V|xD to store word embeddings.
   - To get the semantic of a word, we use the package `spacy` and `en_core_web_lg` model.
   - In order to convert a tweek into a sequence of numbers, keras's Tokenizer is utilized, this is different from TfidfVectorizer that is used for classical models. For conversion of intensity class we use `to_categorical` from `keras`.
   - Define the model: I follow steps in this link to define the model: https://medium.com/swlh/step-by-step-building-a-multi-class-text-classification-model-with-keras-f78a0209a61a. I don't understand the theory of added layers, just know that the embedding matrix is used for adding an `Embedding` layer.
   - Training: Use the `train_X_sequences` and `train_Y` from the conversion step.
   - Predicting on the `test_X_sequences` from the conversion step and print result.

   Run: cd to `deep_learning` folder, run the command: `python3 cnn.py`