Use fasttext sentence embeddings (get_sentence_vector, see description here) to encode the data and train a logistic regression classifier. Select the best configuration based on the validation set, apply it to the test set. (20)

a. Apply TfidfVectorizer to the training data, use idf -weighted & normalized vectors to obtain a new representation. Train & test a logistic regression classifier. (20)

File main.py

```
              precision    recall  f1-score   support

           0       0.70      0.76      0.73        50
           1       0.74      0.68      0.71        50

    accuracy                           0.72       100
   macro avg       0.72      0.72      0.72       100
weighted avg       0.72      0.72      0.72       100

Accuracy (fastText): 0.72
              precision    recall  f1-score   support

           0       0.84      0.86      0.85        50
           1       0.86      0.84      0.85        50

    accuracy                           0.85       100
   macro avg       0.85      0.85      0.85       100
weighted avg       0.85      0.85      0.85       100

Accuracy (TF-IDF): 0.85
```

On the result of accuracy for the `random_state=20`

```
Different states were chosen to vector division
Best result was showed by 20 and 42
```

Substitute the fasttext model with a SentenceTransformers model from the list, e.g. allMiniLM-L6-v2. Use SentenceTransformers' encode method instead of fasttext's get_sentence_vector. Train, tune, and test a logistic regression classifier. (15)

a. Experiment with another SentenceTransformer model. Motivate your choice. (15)

File All_miniLM.py

```
              precision    recall  f1-score   support

           0       0.78      0.95      0.86        44
           1       0.96      0.79      0.86        56

    accuracy                           0.86       100
   macro avg       0.87      0.87      0.86       100
weighted avg       0.88      0.86      0.86       100

Accuracy: 0.86
```

The result for the allMiniLM-L6-v2 is 0.86. But there is slight confusion from recall part in negative data set. Need to investigate why is so high in the negative part.

File paraphare.py

```
              precision    recall  f1-score   support

           0       0.75      0.93      0.83        44
           1       0.93      0.75      0.83        56

    accuracy                           0.83       100
   macro avg       0.84      0.84      0.83       100
weighted avg       0.85      0.83      0.83       100

Accuracy: 0.83
```

This model is one of the smallest and fastests 'paraphrase-MiniLM-L3-v2'
than all-MiniLM-L6-v2, which might be more efficient for applications where inference speed is critical. Also I did not like waiting too much to test my theories that is why I stated with the fastest. Result if accuracy 0.83.

Also he model is fine-tuned for paraphrasing tasks in Yelp reviews. Which means it is good at saying negative or positive sentences in different ways. Which I thought will give some advantage from that previous model. But difference is negligible.

5. Define a Stanza pipeline with a sentiment analysis model. The Stanza model performs threeway classification. Using the validation set, decide which class to assign to the texts classified as neutral. Apply the classifier to the test set in batch mode. (15)

FILE Staza.py

First
If we assume that it 0: 0, 1 : 1, 2:1
Neutral will be taken as positive result then
Validation Set Accuracy: 0.95
Test Set Accuracy: 0.93

Second
If we assume the 0:1, 1:0, 2:1
Neutrall will be taken as negative then
Validation Set Accuracy: 0.9577777777777777
Test Set Accuracy: 0.95

Which shows better results and suggest that neutral comments are often perceived negative from perspective of human mind.

6. Summarize the evaluation results (accuracy on the test set) of all tested configurations, analyze/compare results. (15)

Using fastText and Logistic Regression
Got an accuracy of 72%. The results were fiary good even for both positive and negative sentiments.

Applying TF-IDF and Logistic Regression
Improved accuracy to 85%.
This method showed better results than fastText, meaning it did a better job overall.

Employing SentenceTransformers (all-MiniLM-L6-v2) with Logistic Regression
Reached a slightly higher accuracy of 86%.
This model was particularly good at identifying negative comments, which might show a correlation to pick out negative sentiments more effectively. But it seemed slower in testing that onthers

Using SentenceTransformers (paraphrase-MiniLM-L3-v2) with Logistic Regression

Scored an accuracy of 83%.
Chosen for its speed, this model was a bit less accurate than the all-MiniLM-L6-v2 model and wasn't as precise in classifying sentiments.

Utilizing Stanza for Sentiment Analysis
Recorded an accuracy of 93% or 95%, depending on how neutral sentiments were classified. The analysis showed that viewing neutral comments as negative slightly improved the accuracy, hinting that neutral reviews might often carry a somewhat negative meaning in this context.

Overall Findings
The Stanza model had the top accuracy, possibly because it's specifically made for segmentaitons.

SentenceTransformer models did better than fastText, likely because they are more advanced in understanding the meaning of sentences.

TF-IDF was also effective, especially when used with logistic regression.

Treating neutral reviews as negative for Stanza's analysis seemed to resonate with the data, suggesting that people may see neutral feedback as more negative.

More complex model may give you more accurate results but could be harder to manage in a live environment due to its need for more computing power.