Project Aim: The project aims at classifying the text as title or non-title.

1. **Model:** For this project, the ensemble model XGBoost was employed.
    a. The XGBoost model handles imbalanced data very well.
    b. It catches non-linear relationships too.
    c. Being an ensemble model, it delivers a really good performance.

2. **Vectorizer:** The text was converted from string to numbers (vectors) using the TF-IDF vectoriser as it gave good accuracy.
    a. The model cannot read text, so it is mandatorily converted to vector form.
    b. TF-IDF works on word frequency, denoting each word's importance.

3. **Evaluation Metric:** To consider both precision and recall, the f1-score was considered, giving a value of 0.99. This helps in overcoming false positives and false negatives that arise when the data is imbalanced.

4. **Improvements with Extra Time:**
    - Vectorisers like word2vec could be used instead of TF-IDF.
        - TF-IDF struggles with out of vocabulary words.
        - TF-IDF could create a large sparse matrix when the vocabulary is huge.
    - A word2vec stacked with RNN LSTM could have been used.
        - Many-to-one RNN performs well in this case when the dataset is very large.
        - LSTM can make the model perform better as it can retain the details of title/non-title for a short time.
    - With more computational power, a BERT model could be employed.
        - Rather than just going for vectorization, BERT uses an attention mechanism, which gets a better understanding of the context.

5. **Time:** All these values are approximate
    - Designing Solution: 2 hours
    - Training: This completely depends on the provided data. For the given data, the model was trained in 68 seconds.
    - Prediction: Again, this completely depends on the provided data. For the given data, the predictions were made in about 5 seconds.

6. **Setup and Run**
    - Setup
        - Python 3.9 has been used for the entire project.
        - Run "pip3 install -r requirements.txt" to install all the required dependencies.
    - Run
      Since this is a CLI app, the project runs through the console.
        - For model training:
          python3 app.py --train --data "Training data path"
        - For making predictions:
          Python3 app.py --predict --test-data "Test data path"