PROJECT3 - REPORT
SNLI TEXT ENTAILMENT

GIVEN TASKS:

- perform preprocessing the snli corpus dataset containing train, dev, test sets
- Training natural language inference model using deep learning strategies so that we can predict the gold label when premise and hypothesis are given
- Implement inference API which takes premise, hypothesis as inputs and generates predictions
- Implement Explainability API which takes premise,hypothesis, and inference label as inputs and generate predictions from it to check the correctness of our model

PRE-PROCESSING:

As a part of pre-processing,I:

1. Ensure that classes are balanced while exploring the data
2. Convert sentences into lowercase
3. Remove rows where gold labels are not mentioned
4. Remove all columns except sentence1,sentence2,gold label
5. Remove rows that have empty space in the sentences column
6. Removed all rows with null values
7. Tokenised the sentences
8. Removed stop words and punctuations from sentences
9. Perform lemmatization on each word and again joined all words to form a string
10. Mapped three gold labels as integers such that: {contradiction:0,Entailment:1,Neutral:2}
11. I loaded that preprocessed datasets as pickle files in the folder.

MODEL TRAINING AND TESTING:

- I chose the Bert base uncased pre-trained model as it can capture complex patterns in sentences and generate good predictions for our task of textual entailment
- Using the torch library we can generate feature token ids,segmentids,attention mask ids from pairs of sentences so that they are in the form to feed into the network and labels are also converted to torch tensors.
- Now the model is finetuned by getting trained via these ‘train’ data sent via batch loaders and we make sure that for each epoch we have the model using torch only if validation loss(evaluating the model on ‘dev’ data) is less than best validation loss so as to avoid overfitting
- Training is continued until there is no point where validation loss becomes lesser and the model got saved in a .pth file.
- This model is tested on the ‘test’ set to get an accuracy of around 0.87

DEPLOYING THE MODEL:

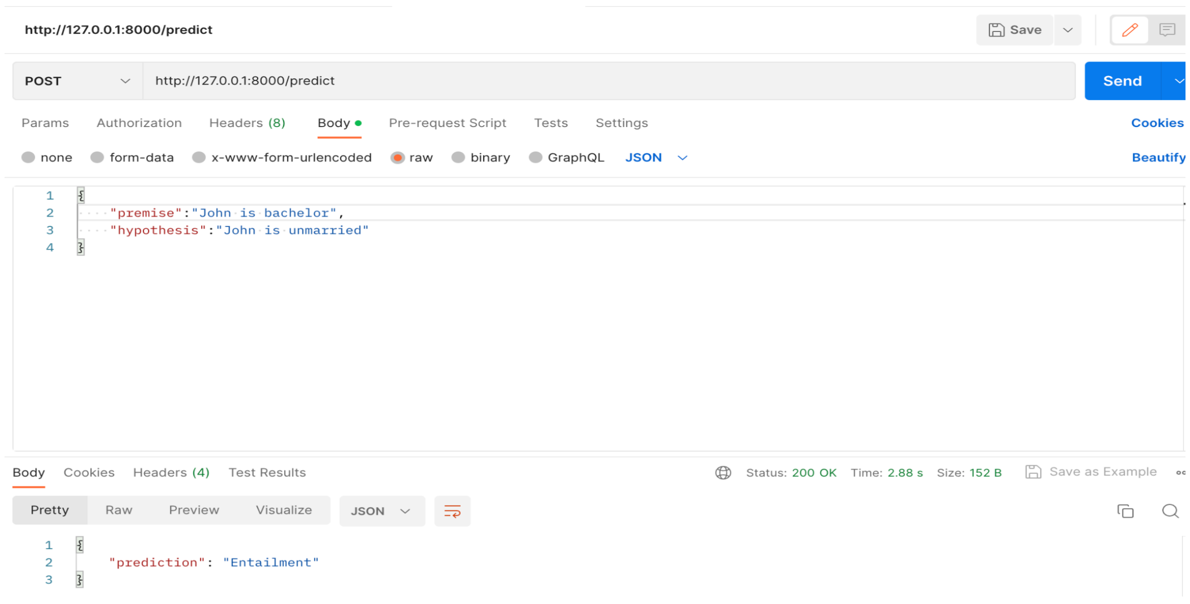
For the purpose of deploying the model via api I chose to use **FastApi** framework which is a modern, fast (high-performance), web framework for building APIs with Python 3.6+ based on standard Python type hints. It is built on top of the Starlette framework for the web parts and the Pydantic library for the data validation and parsing.

FastAPI has gained a lot of popularity due to its simplicity, ease of use, and high performance. It is commonly used for building RESTful APIs, microservices, and web sockets.

IMPLEMENTING INFERENCE API:

- I created a class so that we can take the premise and hypothesis as inputs from the user.
- I created Python script named inference_api
- I instantiated an object named app on FastAPI
- I created a post request endpoint named ‘predict’ on it under which I defined a function ‘predict_relation’ which takes inputs premise, hypothesis and returns prediction as output regarding how they are related i.e one amongst contradiction, neutral or entailment.
- After executing this,if we run the command ‘uvicorn inference_api:app –reload’ on the terminal,uvicorn gets run on a server with a particular url which we should we note down and after starting server process, application startup gets completed and once this process going on we can paste url address at any API platforms like postbox and we can paste it as post request and give inputs as JSON type in body section,to get json output fetched by server describing the relation between pair of sentences.

Demonstration:



http://127.0.0.1:8000/predict

POST http://127.0.0.1:8000/predict

```
{
    "premise":"John is bachelor",
    "hypothesis":"John is unmarried"
}
```

Status: 200 OK Time: 2.88 s Size: 152 B

```
{
    "prediction": "Entailment"
}
```

IMPLEMENTING EXPLAINABILITY API:

For this we take premise, hypothesis, and inference label as inputs from the user and get explanations out of it using explainable ai methods like Shapley,lime etc.I used shapley method for it which historically is a solution concept used in game theory that involves fairly distributing both gains and costs to several actors working in coalition

For this also we are going to use fast API and can generate output using Postbox.

The shap_values key contains the computed Shapley values for each token in the input sentence. Each row corresponds to a different class (in this case, contradiction and entailment). The expected_value key contains the model's expected output value for each class. The instance key contains the input example used to generate the Shapley values. The input_features key lists the tokens used as input features for the Shapley values computation.

The outputs of Shapley values can be useful in several ways like:

Feature importance: Shapley values can be used to determine the contribution of each feature in a prediction, which can help improve the model's performance, interpretability, and fairness.