**Neural Networks & Deep Learning**

**Name**: Maddikunta Dakshyani

#**700 No**: 700666204

* **Task 1:** Save the model and use the saved model to predict on new text data (ex, “A lot of good things are happening. We are respected again throughout the world, and that's a great [thing.@realDonaldTrump](mailto:thing.@realDonaldTrump)”)

1. Data Loading and Preprocessing:

* + The code starts by importing necessary libraries and loading the dataset from a CSV file named 'Sentiment (3).csv'.
  + It preprocesses the text data by converting it to lowercase and removing non-alphanumeric characters.

2. Tokenization and Padding:

* Tokenization is performed using the `Tokenizer` class from Keras, with a maximum of 2000 words.
* The text sequences are padded to ensure uniform length using `pad\_sequences` from Keras.

3. Model Architecture:

* The model architecture is defined using a Sequential neural network from Keras.
* It consists of an Embedding layer with 128 output dimensions, followed by an LSTM layer with 196 units.
* A Dense layer with softmax activation is used for multi-class classification (positive, neutral, negative).

4. Model Compilation and Training:

* The model is compiled with categorical cross-entropy loss and Adam optimizer, with accuracy as the evaluation metric.
* It is then trained on the training data (`X\_train`, `Y\_train`) with one epoch and a batch size of 32.

5. Model Evaluation:

* After training, the model is evaluated on the test data (`X\_test`, `Y\_test`) using the `evaluate` method.
* The evaluation results (loss and accuracy) are printed.

6. Model Saving and Loading:

* The trained model is saved to a file named 'sentimentAnalysis.h5' using the `save` method.
* It is then loaded back into memory using the `load\_model` function from Keras.

7. Predicting on New Data:

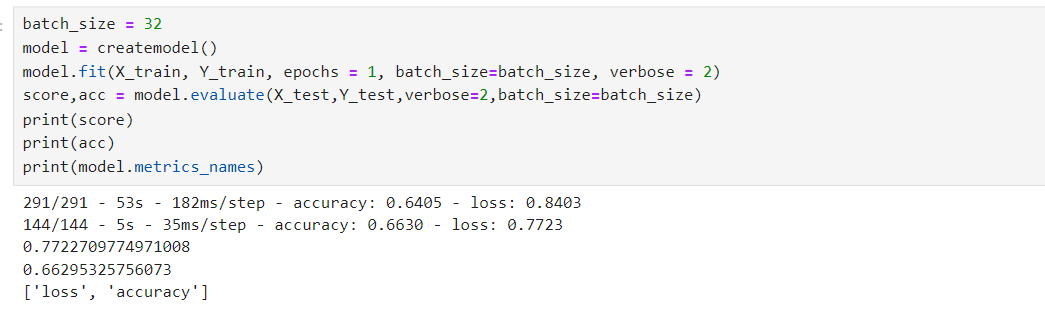
* The sentence is tokenized, padded, and passed through the model to obtain predicted sentiment probabilities.
* The sentiment is determined based on the highest probability class (positive, neutral, or negative).

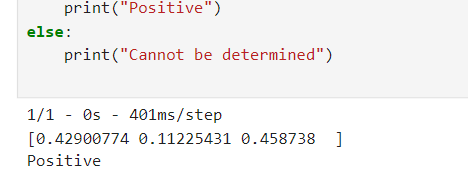
8. Label Encoding:

* The target labels (sentiments) are encoded using LabelEncoder from scikit-learn.
* This encoding allows conversion of categorical labels into numerical format for model training.

9. Train-Test Split:

* The dataset is split into training and testing sets using the `train\_test\_split` function from scikit-learn.
* 67% of the data is used for training and 33% for testing.

****

****

* **Task 2:** Apply GridSearchCV on the source code provided in the class

1. Importing Necessary Libraries:

* The code imports KerasClassifier from keras.wrappers.scikit\_learn for integrating Keras models with scikit-learn and GridSearchCV from sklearn.model\_selection for hyperparameter tuning.

2. Model Initialization:

* The KerasClassifier is initialized with build\_fn=createmodel, where createmodel is a function that defines the Keras model architecture.

3. Hyperparameter Specification:

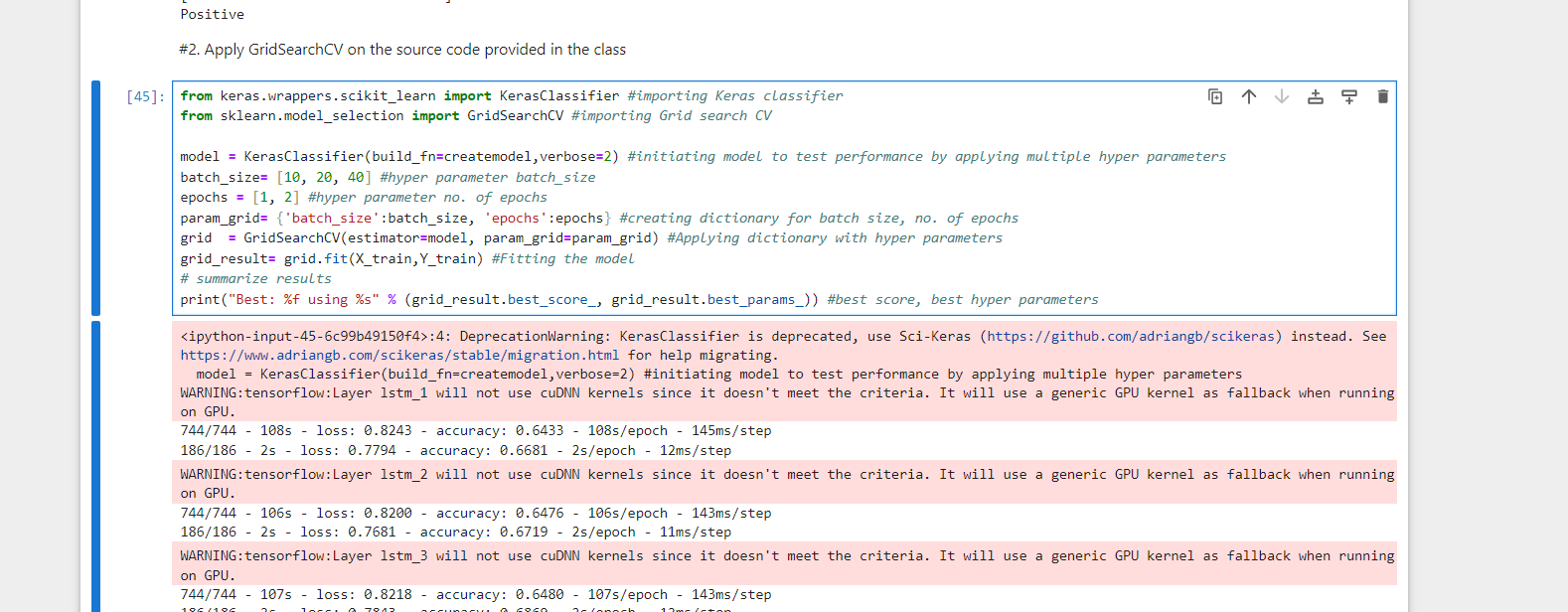
* Hyperparameters such as batch\_size and epochs are specified along with their possible values (batch\_size=[10, 20, 40] and epochs=[1, 2]).

4. Grid Search:

* A param\_grid dictionary is created to specify the grid of hyperparameters to search.
* GridSearchCV is instantiated with the model (estimator=model) and the parameter grid (param\_grid).

5. Model Training and Best Parameters:

* The grid search is performed on the training data (X\_train, Y\_train) using fit.
* The best mean score achieved during cross-validation and the corresponding best hyperparameters are printed using grid\_result.best\_score\_ and grid\_result.best\_params\_.



GitHub Link: <https://github.com/dxm62040ucm/Assign9/blob/main/Assignment8_Dakshyani_700666204.ipynb>

Video Link:

<https://drive.google.com/file/d/15RIiO_HVCpiTSe_Oz-JMoCWU4pJJgK3E/view?usp=drive_link>

**THE END**