Big Data Report for MLSS Project

SATHVIK RAO MP	PES1UG19CS436
SIDDHANTH M	PES1UG19CS481
UTHPAL P	PES1UG19CS548
VISHWAS R	PES1UG19CS579

Dataset chosen : Spam Detection dataset

Design details :

- The entire project can be comprehended easily in terms of modules.
- The main.py script is the driver code which initialises the streaming context, connects to the stream, converts the incoming stream in JSON format to dataframe RDD and eventually calls pre-processing module on the entire batch of the stream.
- The preprocess.py script takes in the entire batch data and does some necessary pre-processing on the data. It splits the data into train and test set and incrementally fits the model on the training set and assesses the performance of the model using the test set.
- The model.py script initialises all the models with their hyperparameters.
- The evaluate.py script is used to assess the performance of the model on the test set.
- The paraTuning.py script is used for hyperparameter tuning of the models.

Surface level implementation details of each unit :

main.py : It connects to the stream and uses various discretised streams (using maps, flatMaps etc on each DS to give another DS) to convert the stream in JSON format to dataframe RDD. Then it calls foreachRDD method on entire batch data.

Preprocess.py : It gets the entire batch data as a dataframe. We have three columns – Subject, Body, Spam/Ham. First, we concatenate the Subject column and Body column into one column named Text to simplify further steps. Then we use Pipelines containing tokenization, stop-word removal, hashing vectorizer and TF-IDF to pre-process the Text column. We convert the Spam/Ham labels to numeric 0 and 1. We split the data into train and test set and fit the model incrementally on train set and assess the performance of the model at each iteration. At the end of the stream, we save the model using pickle.

Model.py : It has two methods fetchNewModel and fetchTrainedModel. When we are training the model, we use fetchNewModel to get new instance of the scikit-learn model with the best hyperparameters. When we are testing the model, we use fetchTrainedModel to get an instance of the already trained model with its pre-trained weights.

Evaluate.py : It uses scikit learn's classification report module to assess the performance of the trained model at each iteration of the stream.

ParaTuning.py : It uses GridSearchCV to try out all the combinations of the hyperparameters and gives as output the best combination of the hyperparameter which outperforms every other combination of the hyperparameters.

Reason behind design decisions :

The design is very easy to follow the flow of the program. Its highly modularised. The design helps us in getting all the necessary data of model's performance like its accuracy at each iteration of the stream which helps us in plotting various graphs. The three models we have chosen for classifying mail as spam/ham are MultinomialNB, Passive Aggressive Classifier and Multilayer Perceptron. The multilayer perceptron gives us the best accuracy.

Takeaway from the project :

- How to build models on streaming data and use it to do predictions on streaming data.

- Affect of training batch_size on model’s performance.
- Affect of test batch_size on model’s performance.
- Hyperparameter tuning on model to find the optimal hyperparameters.
- Analysing different model’s performance.