IE 506: Machine Learning - Principles and Techniques **Jan-May 2023**

Challenging Assignment : Multi Label Classification

1 Introduction

Multi-label classification is a supervised learning problem that involves predicting multiple labels for a given input instance. This problem is encountered in many real-world applications such as image and text classification, where an input instance can belong to multiple categories simultaneously. Unlike traditional binary or multi-class classification, multi-label classification requires the model to predict the presence or absence of multiple labels for a single input instance.

In this assignment, we aim to solve a multi-label classification problem using the "one vs rest" approach. This approach involves training a separate binary classifier for each label, where the positive class is defined as the instances that belong to that label and the negative class is defined as the instances that do not belong to that label. The final prediction is made by aggregating the predictions of all the binary classifiers.

2 Problem Statement:

The objective of this project is to develop a multi-label classification model that can accurately predict the labels associated with each data point in a given dataset. The dataset is provided in the form of a text file where each line represents a data point with its corresponding set of labels and features. The features are represented in the form of `FeatureID:val`, where `FeatureID` is the ID of the feature and `val` is its value.

Our first objective is to preprocess the dataset and convert it into a suitable form for our model. This includes cleaning the data, handling missing values, and converting the features into a suitable format for machine learning models.

Our second objective is to train an Adaboost model on the preprocessed data using the "one vs rest" approach to solve the multi-label classification problem. We will evaluate the performance of the model using two evaluation metrics: F1 score and accuracy.

The F1 score is calculated as:

$$\text{F1 score} = \frac{1}{n}\sum_{i=1}^{n}\frac{|y^i \cap \hat{y}^i|}{|y^i| + |\hat{y}^i|} \tag{1}$$

where n is the total number of data points, y^i is the true set of labels for the i-th data point, and $\hat{y}^i$ is the predicted set of labels for the i-th data point.

Accuracy is calculated as:

$$\text{Accuracy} = \frac{1}{n}\sum_{i=1}^{n}\frac{|y^i \cap \hat{y}^i|}{|y^i \cup \hat{y}^i|} \tag{2}$$

where n is the total number of data points, y^i is the true set of labels for the i-th data point, and $\hat{y}^i$ is the predicted set of labels for the i-th data point.

Our final objective is to identify the best hyperparameters for the Adaboost model using k-fold cross-validation. We will perform a grid search over a range of hyperparameters and select the combination that results in the highest F1 score and accuracy.

3 Solution Approach:

We aim to train a model that predicts the possible labels assigned to a given data point. To achieve this, we adopt the "1 vs rest" method and use an Adaboost classifier for each class. We train binary classifiers corresponding to each label. The approach can be described as follows:

Let there be k labels in the dataset. For each label, we follow these steps:

Define the input space $D = (x^i, y^i)$ where $x^i \in \mathbf{R^d}$ and $y^i \in +1, -1$ for all $i \in 1, 2, \ldots, N$. Here, $y^i = 1$ if the label is present in the label set of the data point, otherwise $y^i = -1$.

Initialize the weight vector $w_i = \frac{1}{N}$ for all $i \in 1, 2, \ldots, N$.

For T rounds, we perform the following steps:

a. Train a weak classifier $h(x)$ with examples weighted using current weights w_i^t:

$$\epsilon_t = \sum_{i=1}^{n} w_i^t \mathbf{I}(h_t(x^i) \neq y^i) \tag{3}$$

b. Compute the weight of the weak classifier:

$$\alpha_t = \frac{1}{2} * \mathbf{ln} \frac{1 - \epsilon_t}{\epsilon_t} \tag{4}$$

c. Update the weight vector as follows:

$$w_i^{t+1} = w_i^t \mathbf{exp}(-1 \alpha_t y^i h(x^i)) \tag{5}$$

d. Normalize the weight vector:

$$w_i^{t+1} = \frac{w_i^{t+1}}{\sum_{i=1}^{N} w_i^{t+1}} \tag{6}$$

Aggregate the weak classifiers as follows:

$$\mathbf{H(x)} = \mathbf{sign}(\sum_{i=1}^{T} \alpha_t h_t(x^i)) \tag{7}$$

Here, $\mathbf{H(x)}$ is our prediction by our one-class binary classifier for our data instance x^i.

We repeat steps 1 to 4 for each label in the dataset.

We aggregate the predictions of all classifiers to create a set of predictions.

Thus, our approach involves training multiple binary classifiers using Adaboost, one for each label, and aggregating the predictions to arrive at the final set of labels for a given data point

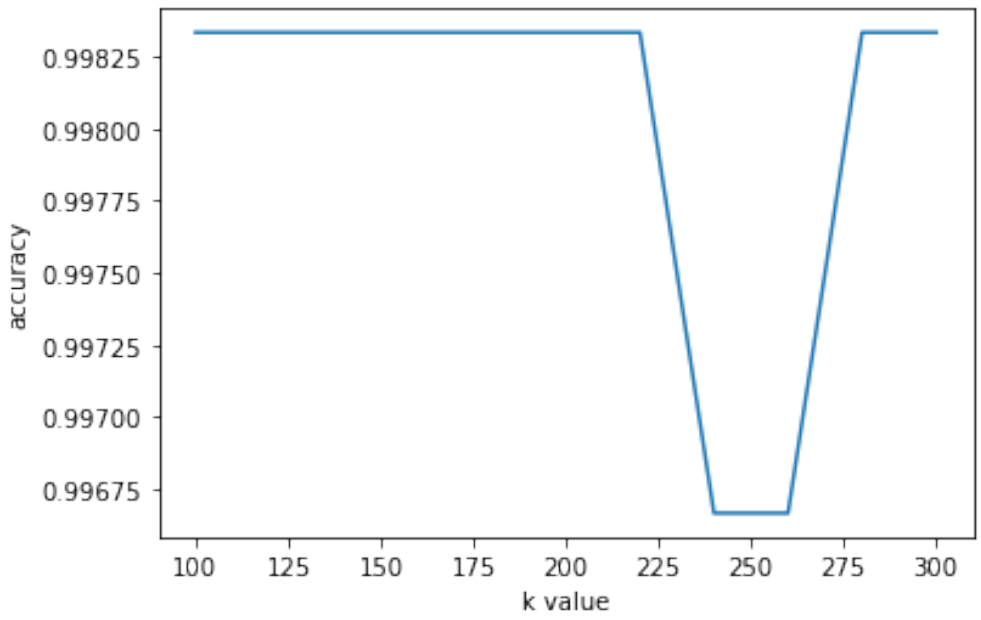
4 Experiments:

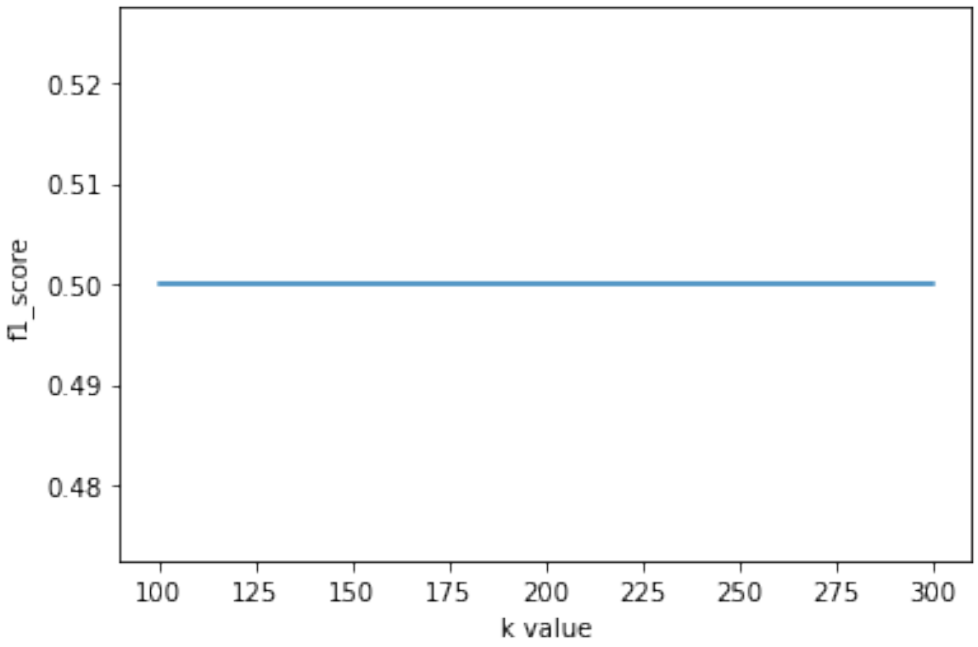
For our project, we utilized AdaBoost algorithm to tune the hyperparameter k and obtain the best value of k, which turned out to be 100. We then proceeded with this value of k to train our model using the training dataset (a multilabel text file). The AdaBoost algorithm is used to train k weak base classifiers and obtain corresponding alpha values for each label.

Next, we applied the aggregation function to the alpha values and classifiers to predict the labels for the test dataset (a sample test data file). This gave us predictions for each class label, which were then used to create a set of predicted labels for each data instance.

Finally, we evaluated the performance of our model by computing the f1 score and accuracy of the predicted labels for the test dataset using relevant functions. The f1 score and accuracy metrics helped us assess the effectiveness of our model in predicting the correct labels for the test dataset.

Overall, our project successfully implemented the AdaBoost algorithm with k-fold cross-validation to fine-tune the hyperparameter and train a model to predict labels for a multilabel dataset. The f1 score and accuracy metrics allowed us to evaluate the performance of our model and assess its effectiveness in predicting the correct labels for the test dataset.





5 Code Introduction:

The code file for this project is designed to perform several tasks that can be explained as follows. Firstly, it can easily read any text file that has a similar format to the provided data files. Secondly, the code is written in a way that allows for the input of a dataframe, which returns the accuracy for each individual as well as the combined predictions in the form of a list of sets. Lastly, the available functions for computing f1 score and accuracy take input in the form of numpy arrays.