CS 4260/5260 – Project 2

Due Date: Mar 24 (Monday), 11:59 PM

> **Project to be completed in groups of 2-3 students each**
>
> One person from each group should notify me before the end of Friday Feb 26 listing who will be in your group. If you prefer to work alone, you should indicate this via email as well. Students for whom I receive no word will be assigned to a group at my discretion.

Project Description:

In this project, you will implement a collection of supervised learning algorithms to make predictions using a simple real-world dataset.

Dataset Description:

I have included in a dataset called data.csv. This dataset was obtained from Kaggle and a link to the original archive is provided at the end of these requirements. The data describes 30,000 songs available on the Spotify music platform.

This dataset allows for predicting the popularity of a song given a variety of attributes like danceability, loudness, tempo, valence, etc. The dataset also includes a number of categorical attributes that will not be used for training. You may make use of these attributes if you wish, but this will make implementation more complex. I list the attributes that I recommend omitting in the appendix at the end of this document. All attributes not included in that list are required to be considered during training. For decision trees, it is NOT required that every required attribute be used as a splitting attribute in the final tree.

The class label, track_popularity, is a number in the range 1-100. You should augment your dataset by converting this to a binary label where the track is considered popular if track_popularity >= 50 and not popular if track_popularity < 50.

Expectations:

For this project, you are responsible for implementing the following minimum functionality:

1. Your program must read in the dataset from the provided csv file. It should be able to isolate the class label (track_popularity) and remove or ignore the attributes that will not be used in training.
2. You are to implement **5-fold cross validation** for all algorithms required. Using 5-fold cross validation, your program should report the **mean** and **standard deviation** of the accuracy across folds.
3. You are to implement the three algorithms below:
    a. Decision Trees
        i. Your program should use **TDIDT** to learn a single decision tree. The TDIDT procedure should be defined in your source code. Solutions that rely primarily on pre-built functions (e.g. sklearn.tree) **will not** receive full credit.

1. You may optoinally use pre-built functions to train as a comparison point to your own solution. Your report must include at a minimum the results for your TDIDT implementation.

ii. You may use any reasonable choice of splitting criteria **selection function** (e.g. entropy, GINI impurity, etc.) and **termination function** criteria (e.g. class purity, change in information, etc.). Ther termination function should always account for maximum tree depth.

iii. Your program should support specifying a **maximum tree depth**. Your report should include results for max depth = 1, 3, and 5.

b. Neural Network

i. Your program should create and learn neural networks with the following architecture configurations at minimum. You are allowed to define your networks statically (i.e. you do not have to generate them programmatically) and using any data structure (e.g. graph, matrices, tensors, etc.).

1. A single perceptron (one neuron, no hidden layers)
2. 1 hidden layer, 5 nodes per hidden layer, 1 output neuron
3. 3 hidden layers, 5 nodes per hidden layer, 1 output neuron

ii. For all networks, you can choose any **non-linear activation function**, a reasonable learning rate, and epoch count. Justify your choices in the report.

iii. Learning should use the **backpropagation algorithm** described in class. You must implement this algorithm in your code. Training using prebuilt training functions (e.g. fit, backward, etc.) within a NN library (e.g. pytorch, tensorflow, scipy, etc.) will not receive full credit.

1. You may optionally use pre-built functions to train as a comparison point to your own solution. Your report must include at a minimum the results for your backpropagation implementation.

c. K-nearest neighbor

i. Your program should implement a simple k-nearest neighbor algorithm.

ii. Your program should support specifying any value of k. Your report should include results for k=1, 3, and 5.

iii. You may optionally include efficiency improvements mentioned in class.

4. You are to include a **report that describes the choices you made** regarding configurations of each algorithm (hyperparameters chosen, etc.) and the **measured performance on 5-fold cross validation (minimum: mean and std).** You should give your reasoning for making the decisions you made and any observations you made regarding model performance during development and testing. In summary, your report should include, at minimum, the required information about:

a. Decision trees:

i. max depth = {1, 3, 5}

b. Neural network:

i. 5 nodes per hidden layer, hidden layer count = {0, 1, 3}

c. K-nn:

i. k = {1,3, 5}

d. A short description of each group member's contribution to the project.

Submission Guidelines:

The deliverables for this project include the source code and documentation as defined above. Unlike project 1, this project requires a report. Submissions are to be made to Brightspace. If submitting multiple files, you should compress them into a zip file format. Only one submission is required per group. Submissions from multiple group members will result in only the most recent submission being graded.

Programming Language:

I default to and recommend Python for this project. If you prefer, you are also allowed to use another language of your choice **with approval from the instructor**. The following languages do not require pre-approval: Python, C, Java. Using a language for which you have not received approval will result in a deduction. Language approval from project 1 does not transfer.

You may use standard libraries for any purpose that does not trivialize the learning process. If you are unsure whether a library or function would be disallowed, you should ask the instructor.

Honor Code

Your group is to complete this project individually. You may discuss the project with other students or with AI tools. You are NOT allowed to share or receive code from other students outside your group. You ARE allowed to use AI tools for code generation, documentation, discussion, etc, but I highly recommend keeping this to a minimum so that you can get the most out of this project. For more information about the honor policy, you should consult the syllabus or inquire with the instructor.

Appendix

The data used in this project was obtained under an open license from https://www.kaggle.com/datasets/joebeachcapital/30000-spotify-songs

The list of 13 attributes that are used in your training algorithm (but not necessarily present in the final model for decision trees specifically) are:

> *danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, tempo, duration_ms,* ***track_popularity (class attribute)***

The 10 attributes that I recommend omitting from both training and inferences are:

> *track_id, track_name, track_artist, track_album_id, track_album_name, track_album_release_date, playlist_name, playlist_id, playlist_genre, playlist_subgenre*