Assignment 2: Tree-based Models

Due Sunday, 7/18, at 11:59PM

Notes

This assignment is worth 100 points total. You must show code to back up your results i.e. if you report just a number, you will get zero credit. Submit your code in a Jupyter Notebook. Include your name in the name of the code file. Clearly label each section of your code that responds to a particular question. There should be a labeled section for Question 1, 2, ...etc. For example, you could call the first section "Question 1: Reading in data".

Background

In this assignment, you will be working with Spotify data. You will explore using tree- based models. The label in this data is the **target** column, which represents whether a song / track is a hit (1) or a flop (0). You're allowed to use any columns as predictors except the artist and uri fields. If you want to use the track column, you'll need to do some feature engineering first (you're not required to use this column, however).

For problems 2, 3, 4, and 5, you should be able to follow very similar logic, except changing the different models. If you get stuck on any problem, let me know via Slack or email.

1. Read in the Spotify data and perform the standard 70 / 30 split into train / validation. Perform any data visualization you think would be relevant for this dataset. (15 points)

2. Train a decision tree using the training data. When you train the decision tree model, use a search grid (either grid search or randomized search) to optimize the hyperparameters. Report the AUC on the training set and the validation set. (15 points)

3. Repeat 2, except use a random forest. (15 points)

4. Repeat 3, except use a gradient boosting model (GradientBoostingClassifier).

(15 points)

5.Repeat 4, except use XGBoost. Remember you will need to install the xgboost package to do this. If you have any issues with installation, send me a message on Slack / email. Which model performed the best from 2, 3, 4, and 5 based off AUC? (15 points)

6.Pick one of the models above and find the optimal probability threshold based off F1-Score (the optimal threshold should be based off the training set). There are a couple of examples in the lecture notebooks for how to do this. Using the optimal threshold, get the following metrics for both the train and validation sets (20 points): a. Accuracy b. F1-Score c. Precision d. Recall

7. Suppose you have a dataset with a binary label (1 or 0). Calculate the Gini impurity of a dataset in the following scenarios. (5 points):

a.The dataset has 75% positive labels and 25% negative labels.

b.The dataset has 90% positive labels and 10% negative labels

---

Data Dictionary:

Track – Name of the song

Artist – name of the first artist listed for the song / track

Uri – unique identifier for the song / track

Danceability – Numerical value measuring how suitable the track is for dancing based on a combination of musical elements, including tempo

Energy – Measure between 0 and 1 of the intensity and activity of the track

Key - Estimated overall key of the track (integers map to standard pitch class notation)

Loudness – Average of the loudness across the length of the track (measured in decibels)

Mode – indicator of the modality (major or minor) of the track

Speechiness – measure of the presence of spoken words in the track. The more exclusively speech-like the track is (e.g. poetry, talk show), the closer this value will be to 1

Acousticness – a confidence value between 0 and 1 whether the track is acoustic (a value of 1 means high confidence track is acoustic)

Instrumentalness – Predicts whether the song is purely instrumental (no vocals). A score closer to 1 means higher chance of the track having no vocals

Liveness – detects the presence of audience in the recording. A value closer to 1 implies a higher chance the track was recorded live

Valence – A measure from 0 to 1 scoring the musical positiveness of the track. Higher scores imply the song is happier or cheerful. Lower scores mean the track is more likely sad or angry.