Take Home Project

**Preprocessing step:**

I started out by importing the dataset using pandas. I then went and dropped all the columns that didn't directly contribute to the audiogram, so essentially dropping everything except the dB values from both ears from 500Hz-8kHz. There are some row with missing values were represented as "\*\*" so I replaced those with NaN values. I then removed every row that contained at least one NaN value. I stack the dat for each ears, left, and right. Then, I looked at dataset, and noticed there are some duplicate values. Ilooked at ditrubation if each column, and noticed, I could remove them after comparing distriubtion of each column before and after removing duplicates. I also converted the data type of columns to float from object. I looked ar describtion of all columns, and noticed that the range of databetween -5, 99.

I looked at heatmap correlation, and noticed,'3k' column has biggere correlation with '2k', '4k', and '6k'. Thus based on correlation matrix, I decided to predict first, '3k','8k', '1k', and '500k'.

**Jupyter notebooks:**

In the first jupyter notebook, I tried diffeent alogirthms, such as Random Forest, SVR Regressor, Gradient Boosting Regressor, and KNN. In order to keep the order of prediction, I used RegressorChain object from sklearn modul.After that, I tune the four algorithms, and based on MSE, and MAE I decided to choose Random Forest model.

In the second jupyter notebook, I decided to remove duplicates step by step, For instance, if I want to predict '3k' for the first model, I dropped '500k' column, and then removed the duplicated from the datset, and then saved index values for considering the predication of '500k'. In the seond step I dropped '1k', and removed duplicates, and so on, until I just left with datset with three columns '2k', '4k', '6k', and target value of '3k'. I created random forest with gridsearch to tune the model, and saved the model.

Here I plotted a flochart of my approach.



All in all, I created four Random Forest models, so I could use them to predict the rest of gain frequencies. Since the dataset was too large, my pc took along time to train and tune each models. Here I decided to focus on the worlflow rather than just accuracy because I beleive If I had more time I could have tuned the hyperparameters, and tried diffrent algorithms.

I the last notebook that I ran on COLAB while my pc was running the model, and tune it, I tried the 2nd jupyter notebook more general, I created class "RemoveDuplicate", and two function, "grid\_Serch\_pipeline", "ModelRF", to simplify the workflow. Moreover, if someone wants to use different machine learning algorithm with the sam workflow, he could use this to create the datset fir each step.

**Final file for using in command line Terminal:**

I created a python file so that I could take use inputs, and predcit the rest of gain frequncies. I wrote this part of code in VSCode, and ran it on my pc, so I could test my model.