Evaluating Speech Separation and Impacting Features

Deeksha Prabhakar
Illinois Institute of Technology

Supervised by: Vijay Gurbani and Daniel Pluth

December 2022

TABLE OF CONTENTS

LIST OF PLOTS

ABSTRACT

Speaker separation involves separating individual speakers from a mixture of voices or background noise, known as the "cocktail party problem." This refers to the ability to focus on a specific sound while filtering out other distractions.

In this analysis, we propose the idea of obtaining features present in the original data and then evaluating the impact they have on the ability of the model to separate the mixed audio streams.

We first start by taking out 400 audio streams from the Voxceleb dataset and combining them to form the 200 single utterances. Once we obtained the mixes, we used the pre-trained Speechbrain model, sepformer-whamr. This model will then proceed to separate the audio mixes given as input and obtain 2 outputs that should be as close as possible to the original ones.

We then try to obtain a feature list from the 400 chosen audios and then go on to assess the effect that certain features have on the model's capability to distinguish between multiple audio sources in a mixed recording.

We use two analysis parameters- permutation feature importance and shap values to conclude which features have more effect on separation.

OBJECTIVE

In this project, our main aim is to obtain a list of features that capture the data well and then advance to obtaining the most contributing features i.e the features that will lead to a better separation or a worse separation.
The dataset will be prepared in a way that these feature values can be used as input variables and then go on to using various models like Logistic Regression, Decision Trees, SVM (both rbf and linear kernel), XGBoost, AdaBoost, to obtain the above-mentioned results. These results shall then be analyzed to conclude the features that affect separating the audio streams the most.

METHODOLOGY

A. Dataset Preparation:

We first listened to the audios and extracted the distinguishing features present in the original audio streams (audio streams before mixing).

The list of features that was obtained from our initial screening is:

1. Gender - Male or Female
2. Presence of Two or More Speakers
3. Presence of Background Noise if any
4. Pitch of the Audio Streams
5. Intensity of the Audio Streams

We used the parselmouth tool and managed to extract the pitch and intensity from the audio files. Parselmouth uses Praat on its backend.
This was followed by another round of listening to all the audio streams to compile a comprehensive list describing the features that are present or absent in that particular audio file.

We first started by exploring the data and compiling the features in a way that they represent the mixes.
We started by taking individual values for each of these features and then proceeded to encode the values for their combinations.

1. **Gender:**
   Encryption Key used:

Value of Feature for mix1_original1	Value of Feature for mix1_original2	Encoded Value
Male	Male	0
Male	Female	1
Female	Male	1
Female	Female	2

To demonstrate,

if mix1_original1 was a male voice and mix1_original2 was also a male voice and these are combined to get mix1, we assign the combination a value of 0 and so on.

2. **Presence of Two or More Speakers:**

Encryption Key used:

Value of Feature for mix1_original1	Value of Feature for mix1_original2	Encoded Value
No	No	0
No	Yes	1
Yes	No	1
Yes	Yes	2

For example,

if mix1_original1 was characterized by the presence of one speaker only and mix1_original2 was characterized by the presence of more than one speaker and these are combined to get mix1, we assign the combination a value of 1 and so on.

3. **Background Noise:**

We observed that the main types of background noise present in the audio are noise, echo, static, music and studio audience.

Encryption Key used:

Value of Feature for mix1_original1	Value of Feature for mix1_original2	Encoded Value
None	None	0
None	Noise	1
None	Static/ Electronic Distortion	2
None	Cough	3
None	Music	4

None	Echo	5
None	Wind	6
None	Beeping	7
None	Crash	8
Noise	Noise	9
Noise	Music	10
Static/ Electronic Distortion	Music	11
Noise	People in Background	12
None	Studio Audience	13
Echo	Static/ Electronic Distortion	14
Echo	Noise	15
Music	Music	16
Static/ Electronic Distortion	Noise	17
Studio Audience	Music	18
Static/ Electronic Distortion	Studio Audience	19

For example,

if mix1_original1 was had echo and mix1_original2 had some static/electronics distortion and these are combined to get mix1, we assign the combination a value of 14 and so on.

4. **Pitch (min):**

For this feature's value we compared the pitch value obtained from mix1_original1 and mix1_original2, and picked the lower one of the two.

5. **Pitch (diff):**

For this feature's value we obtained the pitch(max) in the same way as described above and obtained pitch(diff) by subtracting the pitch(min) value from pitch(max).

6. **Intensity (min):**
   For this feature's value we compared the intensity value obtained from mix1_original1 and mix1_original2, and picked the lower one of the two.

7. **Intensity (diff):**
   For this feature's value we obtained the intensity(max) in the same way as described above and obtained intensity(diff) by subtracting the intensity(min) value from intensity(max).

8. **WER-Amazon:**
   For this feature's value we obtained the values of Word Error Rate defined by comparing transcripts from Amazon transcribe and Human transcription.
   We took this value before mixing and after separation and then took the difference between these two values.
   We then combined these values for mix1_original1 and mix1_original2 by taking an average of the two and similarly for other audios as well.

   For example,
   Let's take the example of mix1_original1 and mix1_original2:

Audio File	WER before - human-Amazon	WER after - human-Amazon	WER - Amazon
mix1_original1	11.1	18.5	8.7
mix1_original2	20	30	

   We obtained the value of 8.7 as-
   WER - Amazon = ½ * ((18.5-11.1)+ (30-20)) = 8.7
   We've then considered 15% as the threshold value i.e WER values greater than 15 are considered as badly separated and encoded as 0 and values less than or equal to 15 are considered as good separation and encoded as 1.

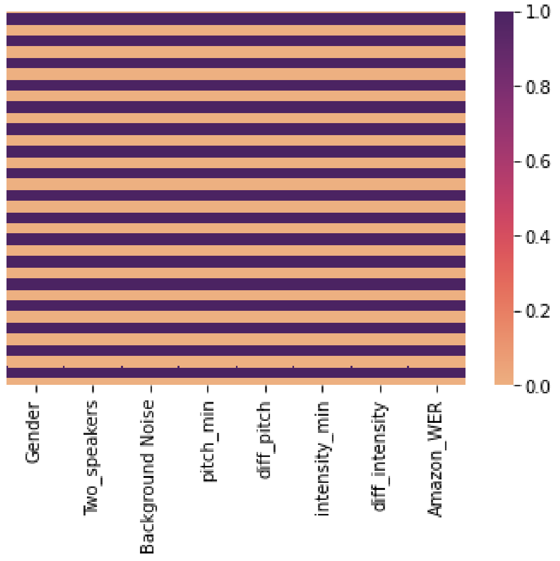
This led to the formation of our dataset that we will use in the model training phase to obtain feature importance.

B. Data Exploration And Analysis:

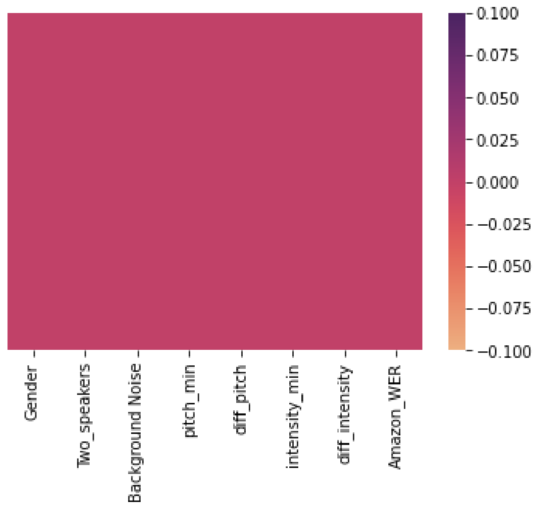
Now that our dataset is ready, we performed various exploratory data analysis techniques to get a good look at our data and understand the values we have.
We will consider WER- Amazon as our target variable and gender, presence of speaker, Background Noise, pitch(min), pitch(diff), intensity(min), and intensity(diff) as predictor variables in our model training phase.

EDA Steps:

1. Removing null values:
   Since we encoded the values of features for mixes and not individual audio streams we had null values in our dataset. We removed them as the first step

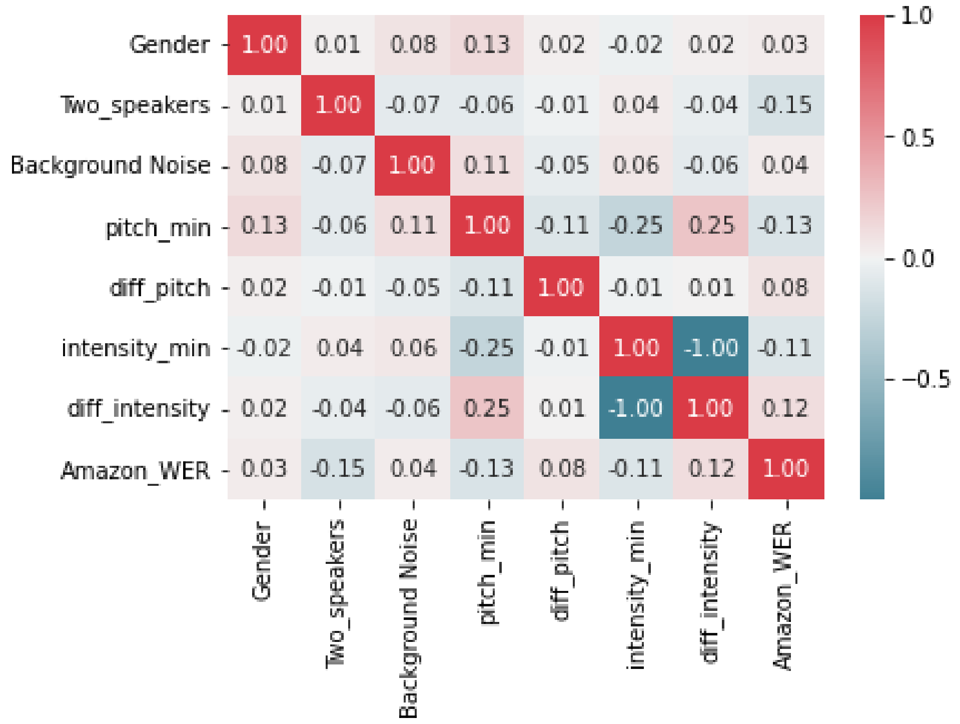


Plot 1.1: Data before removing all null values

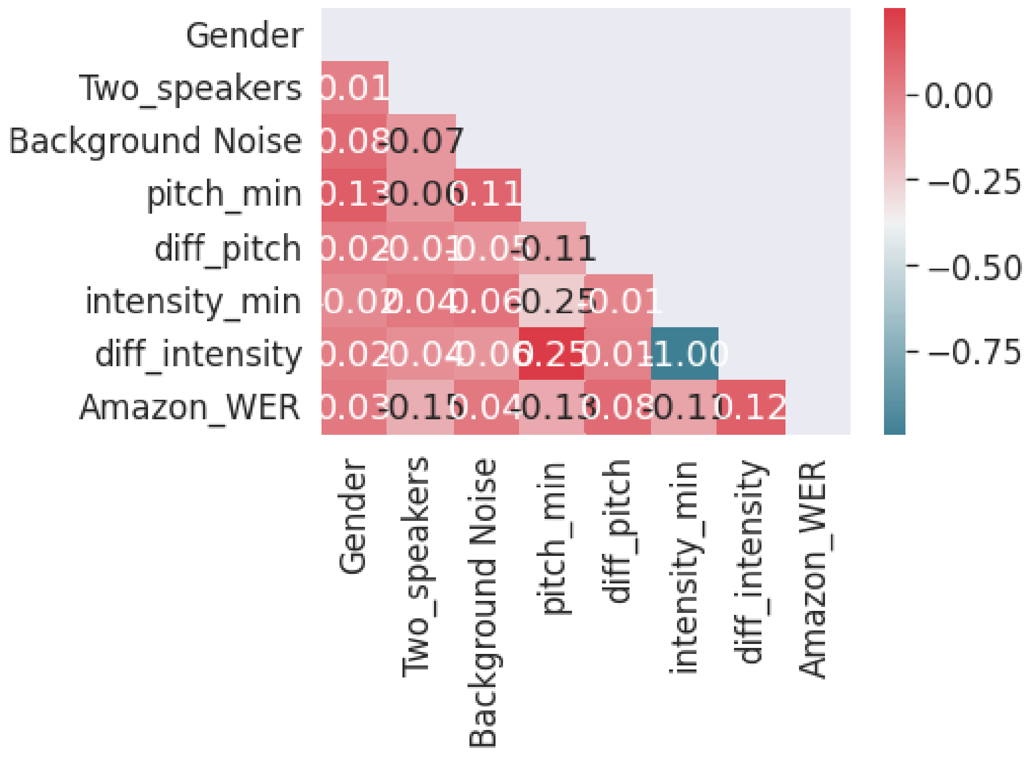


Plot 1.2: Data after removing all null values

2. Obtaining correlation Matrix:
   We obtained correlation values between the various variables and obtained the following:

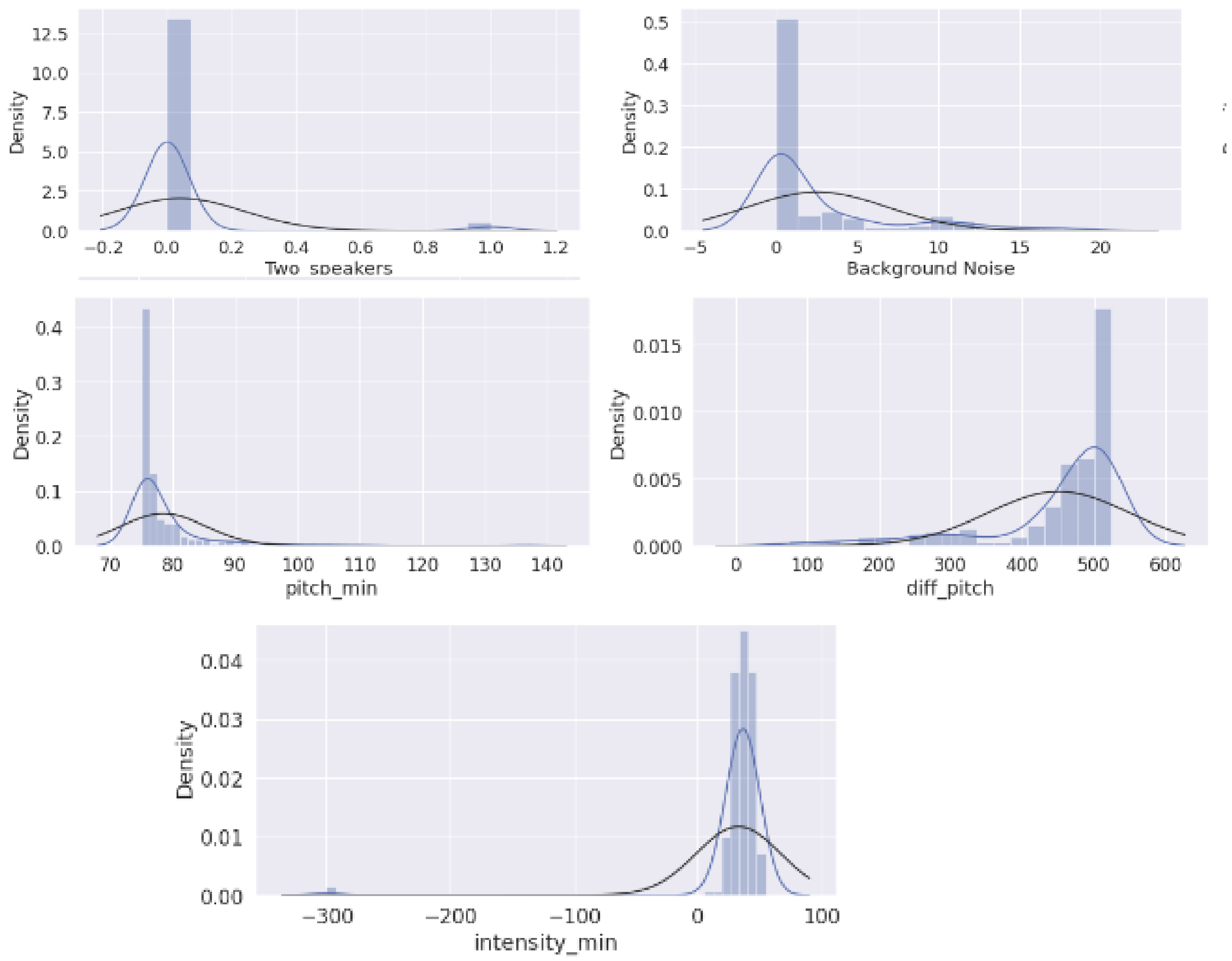


	Gender	Two_speakers	Background Noise	pitch_min	diff_pitch	intensity_min	diff_intensity	Amazon_WER
Gender	1.00	0.01	0.08	0.13	0.02	-0.02	0.02	0.03
Two_speakers	0.01	1.00	-0.07	-0.06	-0.01	0.04	-0.04	-0.15
Background Noise	0.08	-0.07	1.00	0.11	-0.05	0.06	-0.06	0.04
pitch_min	0.13	-0.06	0.11	1.00	-0.11	-0.25	0.25	-0.13
diff_pitch	0.02	-0.01	-0.05	-0.11	1.00	-0.01	0.01	0.08
intensity_min	-0.02	0.04	0.06	-0.25	-0.01	1.00	-1.00	-0.11
diff_intensity	0.02	-0.04	-0.06	0.25	0.01	-1.00	1.00	0.12
Amazon_WER	0.03	-0.15	0.04	-0.13	0.08	-0.11	0.12	1.00



Plot 2.1: Correlation Values between variables

3. Distribution Plots:
   We then obtained the distribution plots to better understand the variables and their values. We obtained the following results:



Plot 3.1: Distributions of various variables

4. Dealing with categorical and numerical variables:
   On analysis, it was seen that gender, presence of two speakers, and background noise are categorical values. We then went on to one-hot encode them to obtain proper values for training our models.
   We also scaled our numerical variables (pitch(min), pitch(diff), intensity(min), intensity(diff)) so that they don't interfere with the weights assigned to variables during training.

C. Model Training and Feature Importance:

In this phase, we used our dataset to train various models and then obtain important features using the following two parameters:

1. **Permutation Feature Importance:**

   Permutation importance uses models differently. It is calculated after a model has been fitted. So we won't change the model or change what predictions we'd get for a given value of height, sock count, etc.
   It causes us to think that if we randomly shuffle a single column of the validation data, leaving the target and all other columns in place, how would that affect the accuracy of predictions in that now-shuffled data.
   Randomly re-ordering a single column should cause less accurate predictions, since the resulting data no longer corresponds to anything observed in the real world. Model accuracy especially suffers if we shuffle a column that the model relied on heavily for predictions.
   It follows the following procedure:

1. Take a trained model.
2. Shuffle the values in a single column, and make predictions using the resulting dataset. Use these predictions and the true target values to calculate how much the loss function suffered from shuffling. That performance deterioration measures the importance of the variable you just shuffled.
3. Return the data to the original order (undoing the shuffle from step 2). Now repeat step 2 with the next column in the dataset, until you have calculated the importance of each column.

2. **SHAP values:**

   SHAP (SHapley Additive exPlanations) values are a method to explain the output of any machine learning model. They provide an explanation for the predictions of the model by providing a feature importance score for each input feature.

SHAP values are based on the concept of Shapley values from cooperative game theory, which provide a way to distribute a value among a group of individuals based on their contributions. In the case of machine learning, the value being distributed is the model's prediction, and the individuals are the input features.

Each feature is assigned a SHAP value, which represents the feature's contribution to the model's output. A positive SHAP value means that the feature has a positive impact on the model's prediction, while a negative SHAP value means that the feature has a negative impact. The sum of all the SHAP values for all the features equals the model's prediction.

SHAP values can be used to understand how a model is making its predictions and identify which features are most important for specific predictions. They can also be used for model interpretability, comparison of different models, and to detect bias and fairness in a model.

In general, SHAP values have a few properties that make them a powerful tool for interpreting a model:

1. They are model-agnostic, meaning that they can be used with any type of model.
2. They are locally accurate, meaning that they provide the feature importance for each specific prediction, rather than just a global feature importance score.
3. They have theoretical guarantees of consistency and completeness, meaning that they provide a complete and consistent explanation of the model's predictions.

We used the following models to obtain feature importance:

1. **Logistic Regression:**

   It is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes). It is used to predict a binary outcome (1 / 0, Yes / No, True / False) given a set of independent variables.

   In logistic regression, the relationship between the independent variable(s) and the outcome variable is modeled using probability. Specifically, the logistic function is used to model the probability that a given input belongs to a certain class.

   The logistic function, also known as the sigmoid function, is defined as:

   $$p = 1 / (1 + e^{(-x)})$$

   where p is the probability that the input belongs to the positive class, and x is the input to the function (a linear combination of the independent variables).

   The goal of training a logistic regression model is to find the best parameters (also known as coefficients or weights) for the independent variables such that the model is most accurate at predicting the outcome.

2. **Decision Trees:**

   A decision tree is a type of algorithm used for both classification and regression tasks in machine learning. It is a flowchart-like tree structure, where each internal node represents a test on an attribute, each branch represents the outcome of a test, and each leaf node represents a class label. The topmost node in the decision tree is known as the root node.

The algorithm starts at the root node and splits the data into subsets based on the values of the input features. This process is repeated recursively for each subset of the data, creating a new internal node at each step. The process stops when a stopping criterion is met, such as reaching a maximum tree depth or having a minimum number of samples in a leaf node.

In classification tasks, the leaf nodes represent class labels, and the goal is to predict the class label for new data by traversing the tree from the root node to a leaf node. In regression tasks, the leaf nodes represent a predicted value, and the goal is to predict a numeric value for new data.

Decision trees are simple to understand and interpret, and they can handle both categorical and numerical data.

3. SVM:

Support Vector Machines (SVMs) are a type of supervised learning algorithm that can be used for both classification and regression tasks. The basic idea behind SVMs is to find a decision boundary that separates different classes in the data as best as possible. This decision boundary is called the "maximum margin hyperplane," and the points that are closest to it are called "support vectors.”

a. SVM with rbf kernel:

The radial basis function (RBF) kernel is one of the most popular kernel functions used in SVMs. It is a non-linear kernel that can transform the input data into a higher-dimensional space in which a linear decision boundary can separate the classes. The RBF kernel calculates the similarity between the input data and a landmark point, which is usually chosen at random, based on the distance of the input data from the landmark point.

The equation for an RBF kernel is

$$K(x,x') = e\text{^}(-\gamma||x-x'||\text{^}2)$$

Where x and x' are input data, γ is a parameter that can be adjusted, and ||x-x'|| is the Euclidean distance between x and x'. The value of γ determines the width of the decision boundary. A smaller value of γ produces a wide boundary, while a larger value of γ produces a tight boundary.

The key advantage of using an RBF kernel is that it can model non-linear decision boundaries without the need to explicitly compute higher-dimensional representations of the data.

b. **SVM with Linear Kernel:**
One way to create this boundary is through the use of a linear kernel. A kernel is a function that maps the input data into a higher dimensional space, where it may be easier to find a linear boundary. The linear kernel is simply the dot product of the input data, so it does not transform the input data into a higher dimensional space, but instead uses the raw data to create the boundary. This can be computationally efficient and is often used when the data is not linearly separable in the original feature space.

In simple words, a linear kernel only checks the linear boundary which is a straight line that separates the data, whereas in rbf (radial basis function) kernel we can separate the data via hyperplane as well as a non-linear boundary which can help in cases where data is not linearly separable.

4. XGBoost:

It is used for supervised learning problems such as regression, classification, and ranking. The key idea behind XGBoost is to build an ensemble of decision trees using the gradient boosting framework, where each tree is trained to correct the mistakes of the previous tree. This results in a model that is able to capture complex non-linear patterns in the data.

The algorithm is implemented using a variant of gradient boosting called tree boosting, which uses decision trees as base learners. Decision trees are a type of model that splits the input space into smaller regions based on the value of the input features. Each split results in a new decision tree node, which is used to make a prediction. XGBoost uses a specific type of decision tree called an "extensive gradient boosting" decision tree, which is able to handle missing values and categorical variables more effectively.

One of the key advantages of XGBoost is its ability to handle large datasets with a high number of features. Another key advantage of XGBoost is that it has built-in regularization in the form of L1 and L2 regularization, which can help prevent overfitting.

5. AdaBoost:

AdaBoost, or Adaptive Boosting, is an ensemble machine learning algorithm that is used to improve the performance of other base classifiers. The algorithm works by training multiple base classifiers, where each classifier is trained on a weighted version of the data. The weighting is based on the performance of the previous classifiers: observations that are misclassified by previous classifiers are given higher weights so that future classifiers focus more on these observations. The final prediction is made by combining the predictions of all the base classifiers using a weighted majority vote.

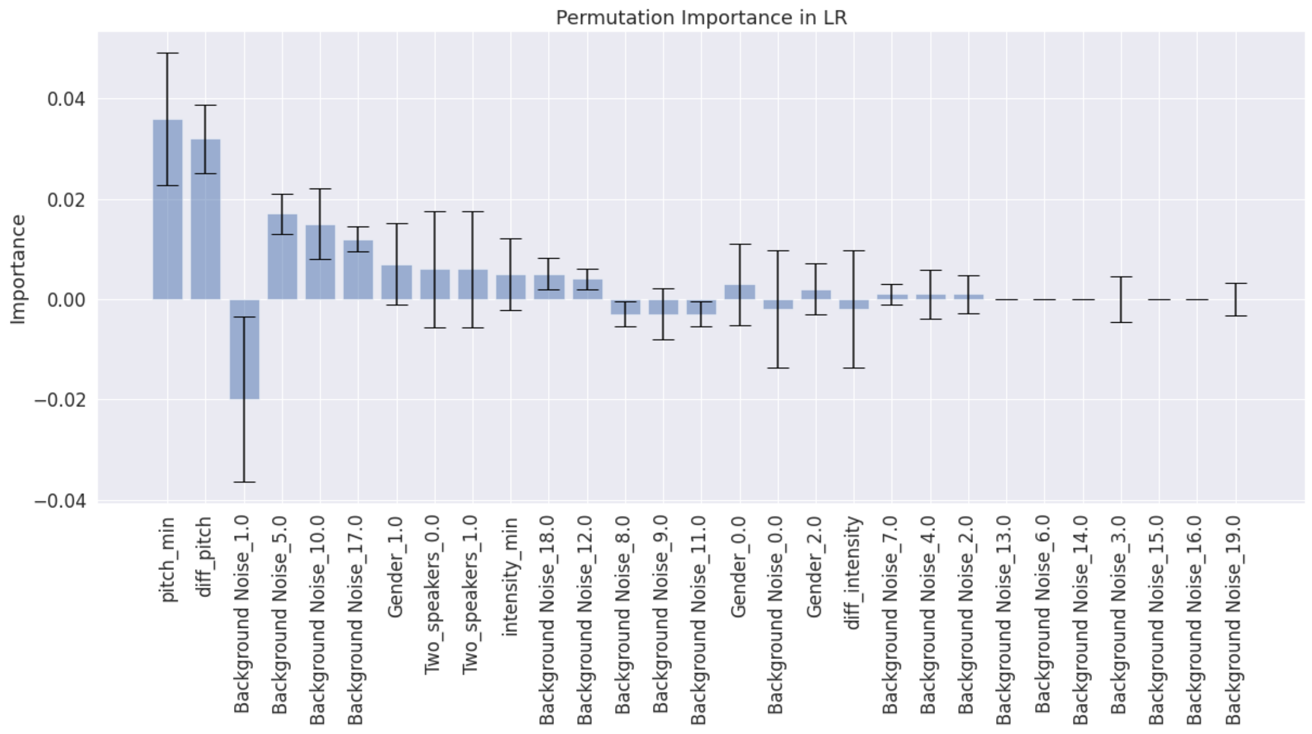
AdaBoost is an iterative algorithm that starts by training a base classifier on the entire dataset. The algorithm then evaluates the classifier's performance and assigns a weight to each observation based on whether it was correctly classified or not. In the next iteration, the algorithm trains a new classifier on the weighted data, and so on. The number of iterations, or the number of classifiers to be trained, is a user-specified parameter.

One of the strengths of AdaBoost is that it can be used with a variety of base classifiers, including decision trees, logistic regression, and artificial neural networks. Additionally, it is relatively simple to implement and interpret, and often works well even when the base classifiers are not very strong.

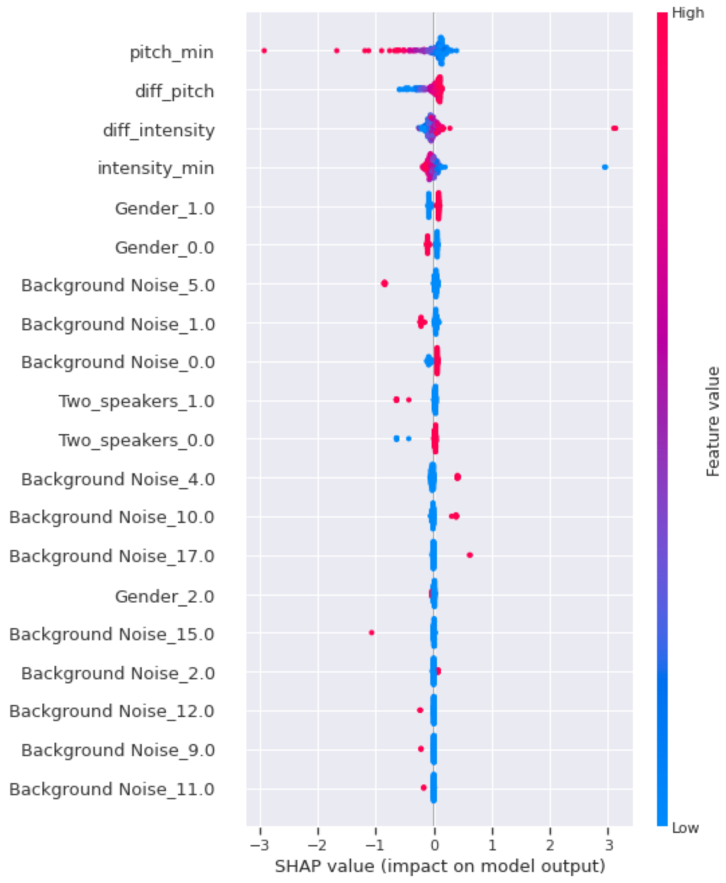
RESULTS

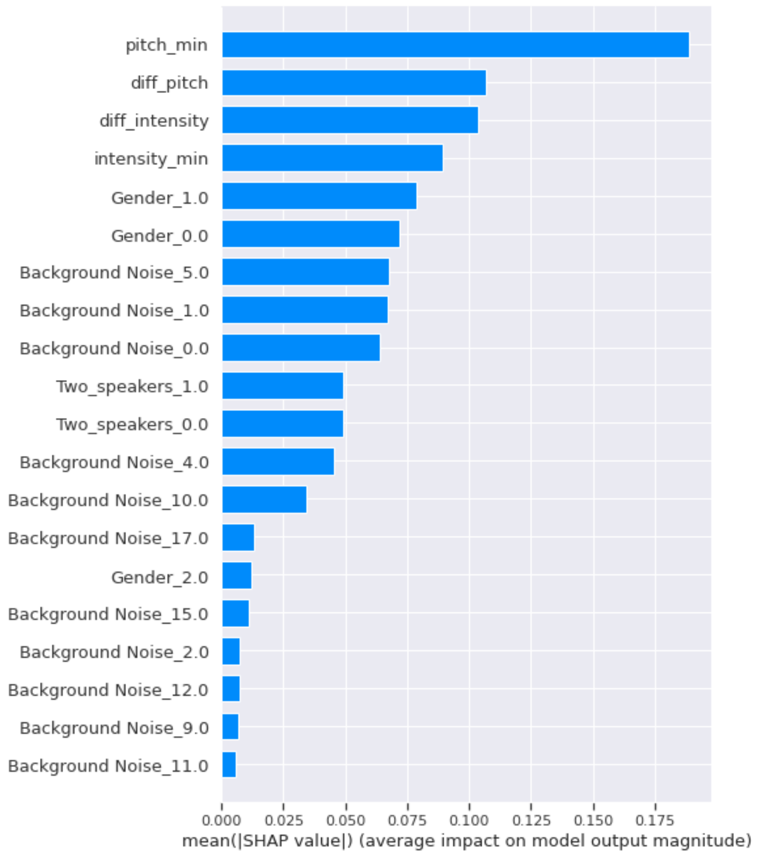
After training the above models and obtaining permutation importance as well as SHAP values, the following results were obtained:

1. Logistic Regression:



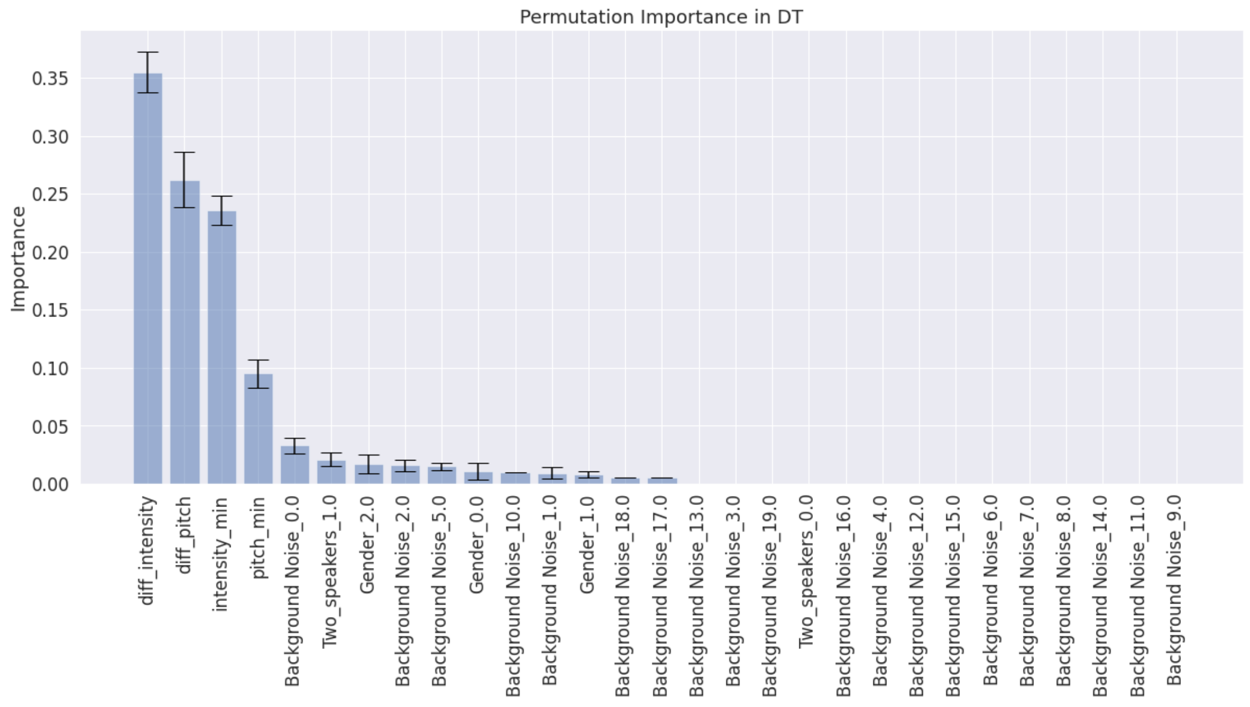
Plot 4.1.1: Permutation Importance in Logistic Regression



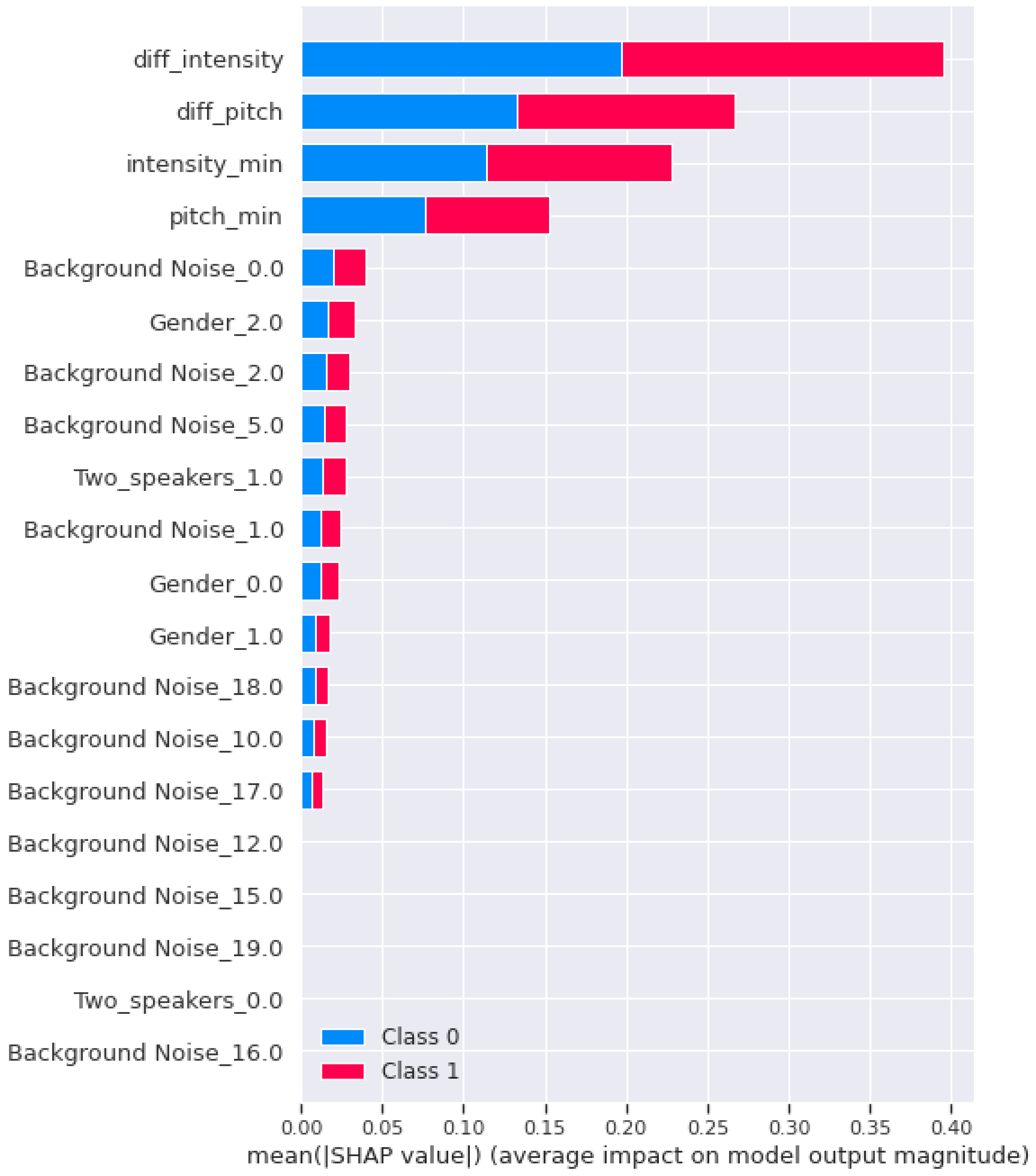


Plot 4.1.2: SHAP values from Logistic Regression

2. Decision Trees:

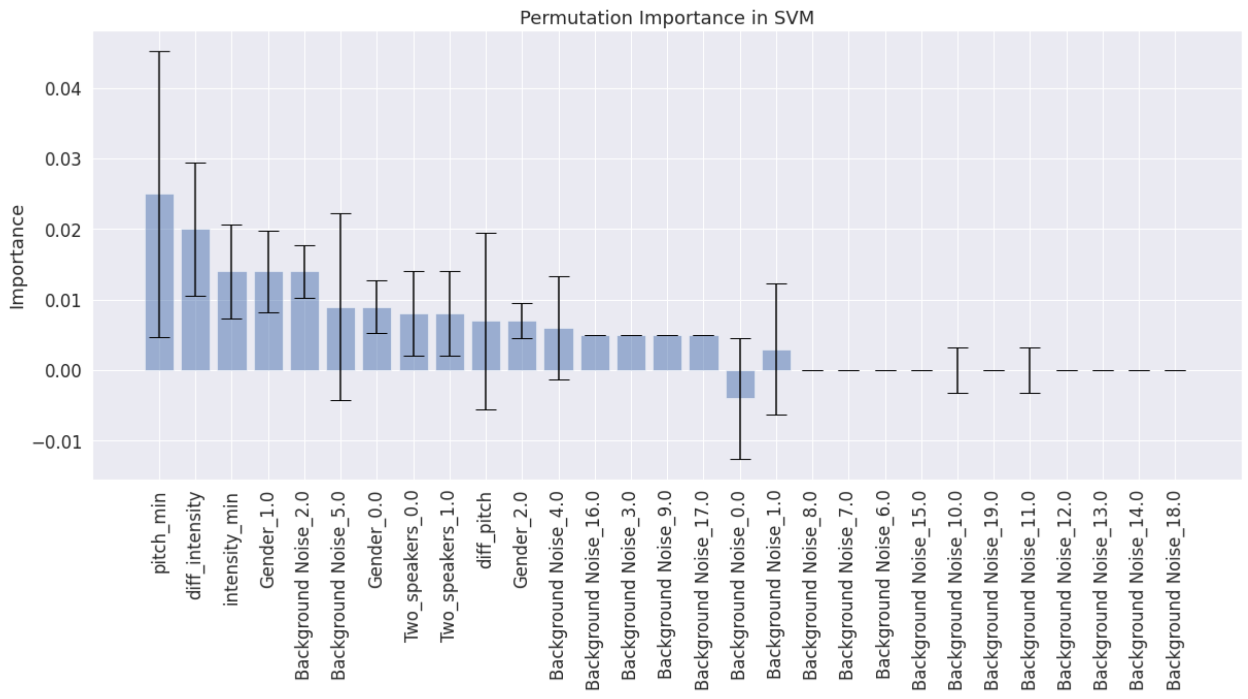


Plot 4.2.1: Permutation Importance in Decision Trees

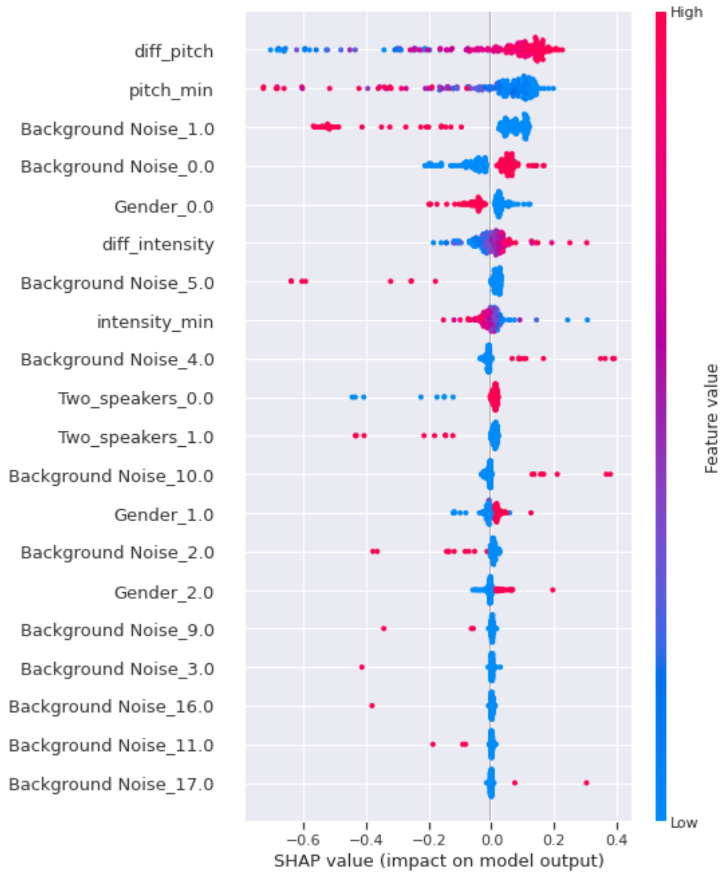


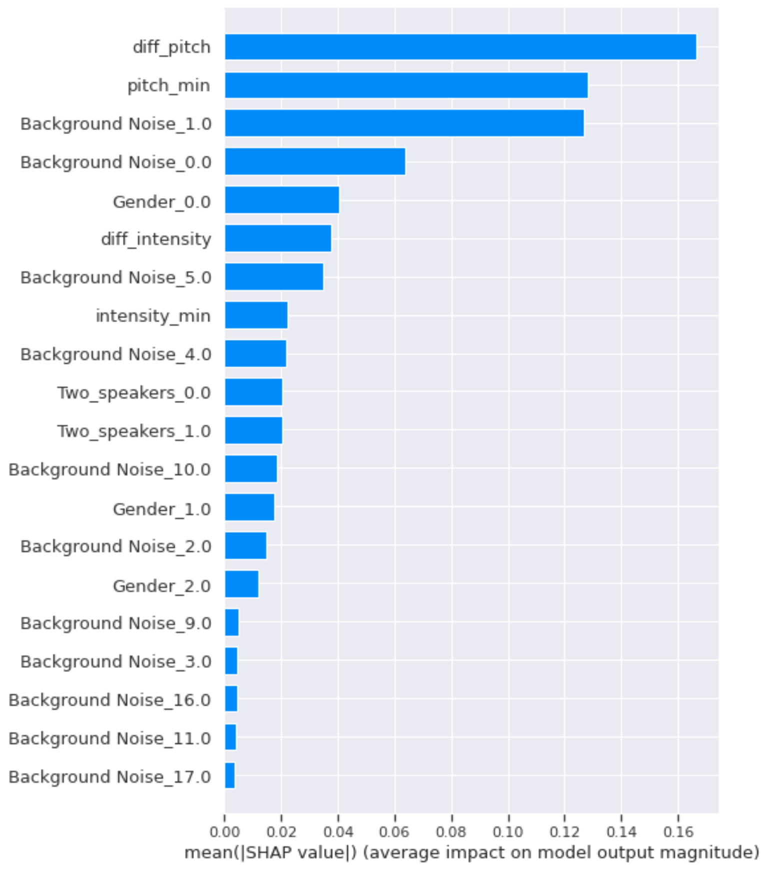
Plot 4.2.2: SHAP values from Decision Trees

3. SVM with rbf kernel:



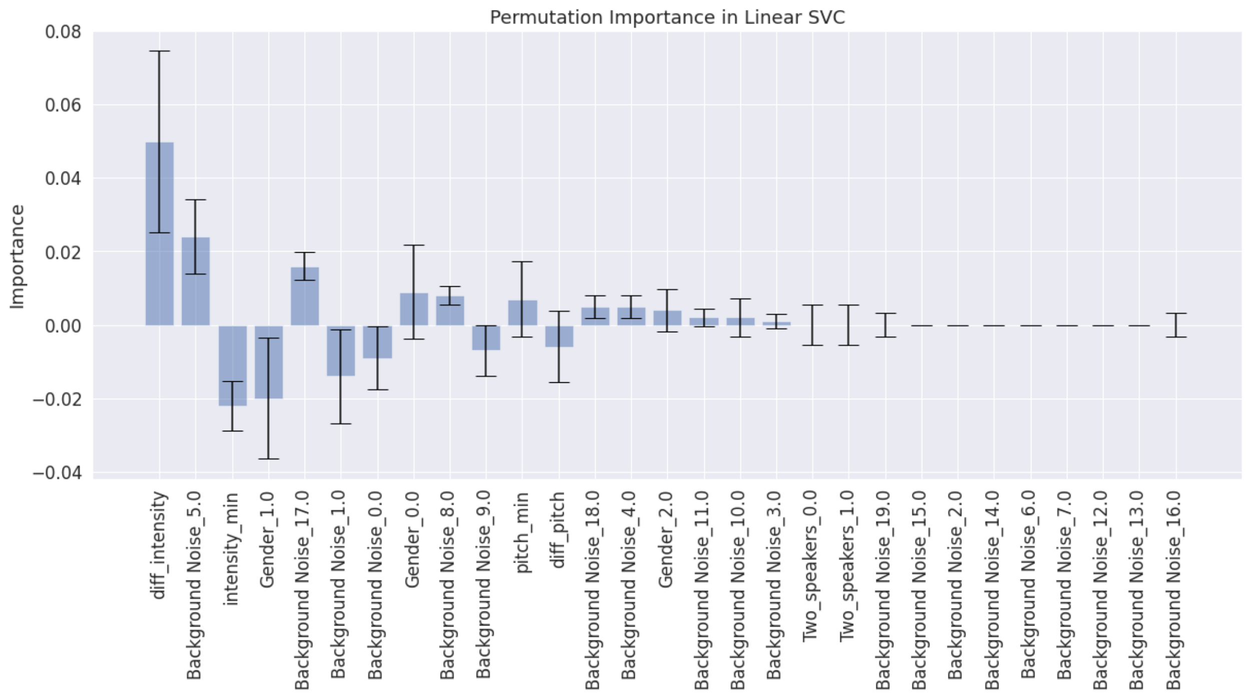
Plot 4.3.1: Permutation Importance in SVM with rbf kernel



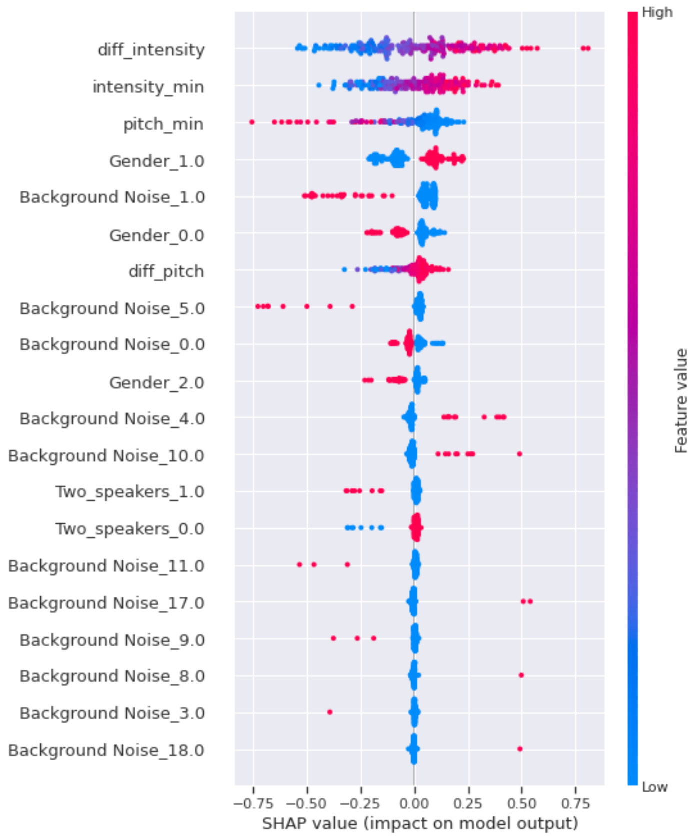


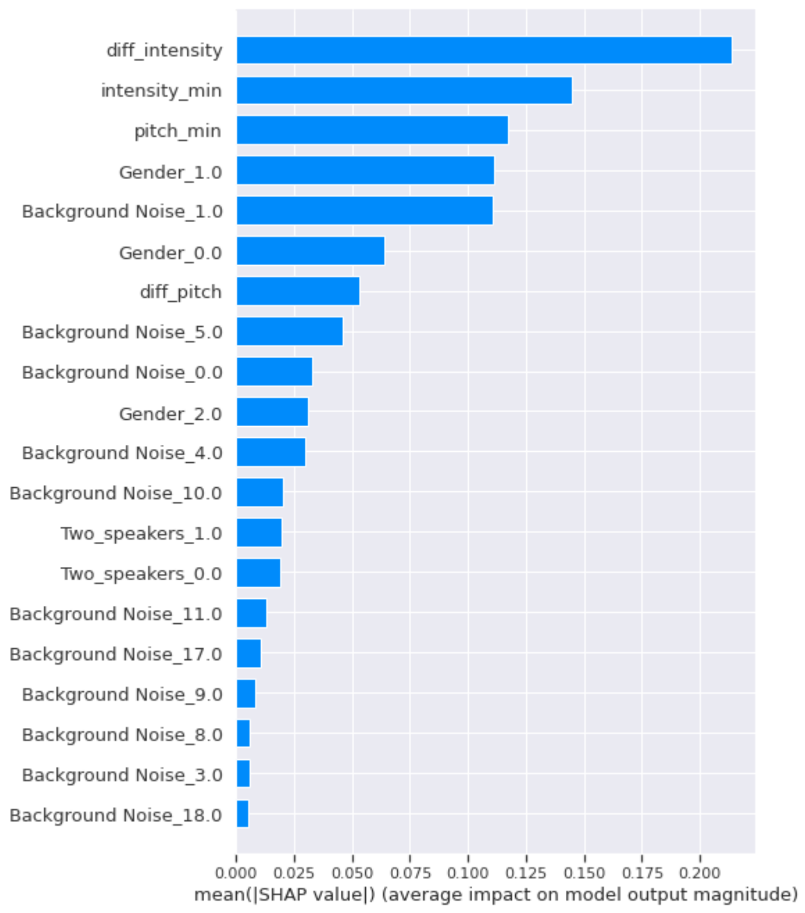
Plot 4.3.2: SHAP values from SVM with rbf kernel

4. SVM with linear kernel:



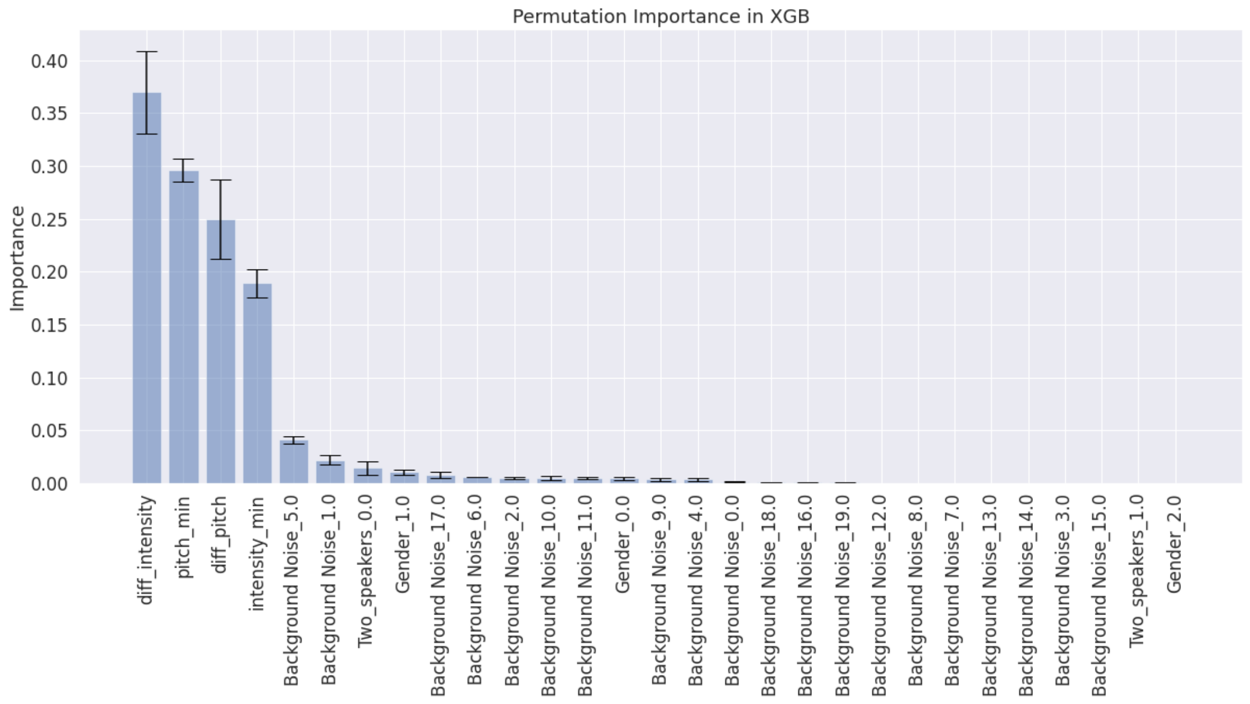
Plot 4.4.1: Permutation Importance in SVM with linear kernel



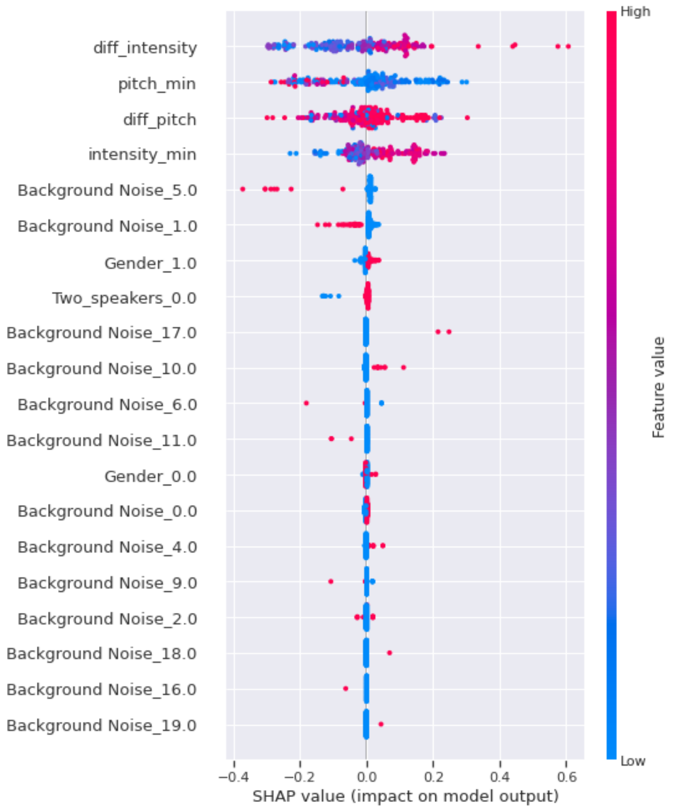


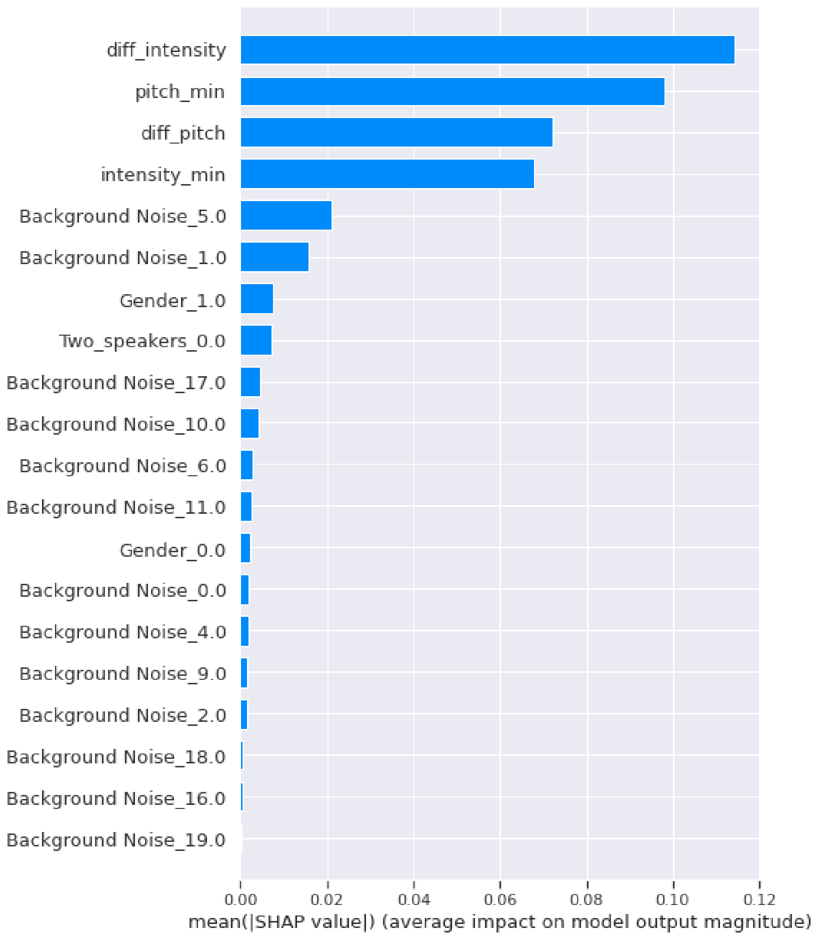
Plot 4.4.2: SHAP values from SVM with linear kernel

5. XGBoost:



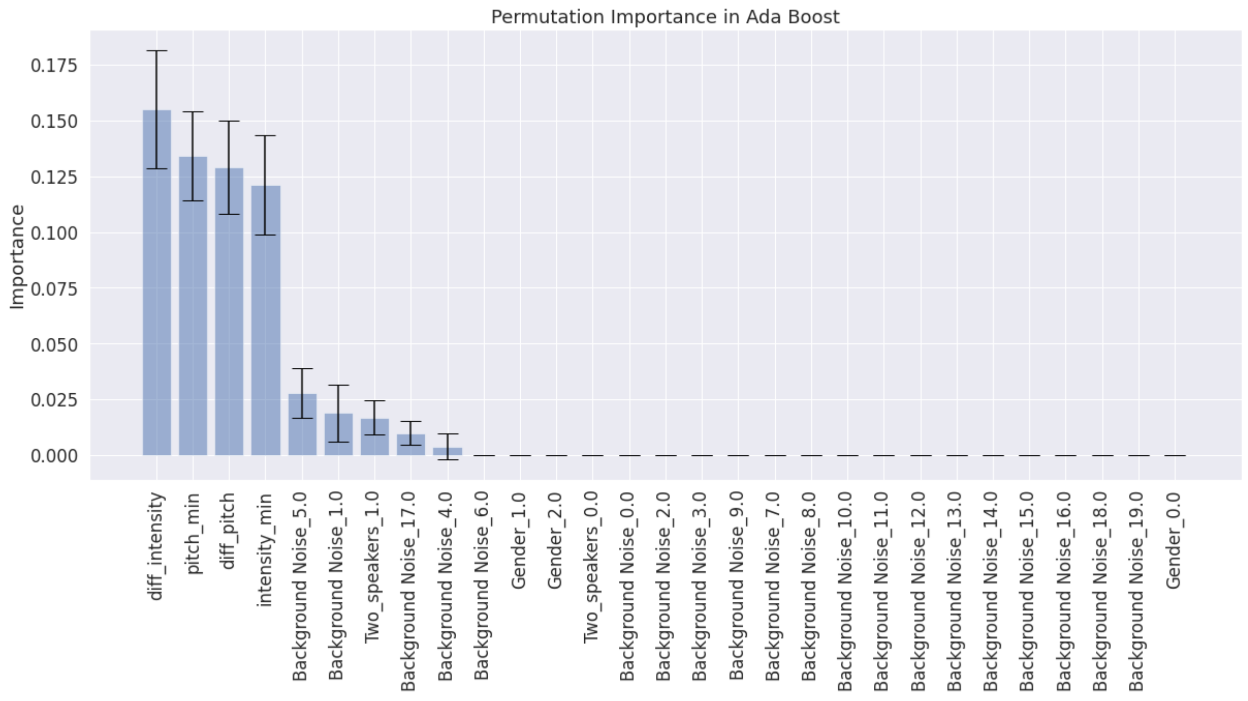
Plot 4.5.1: Permutation Importance in XGBoost



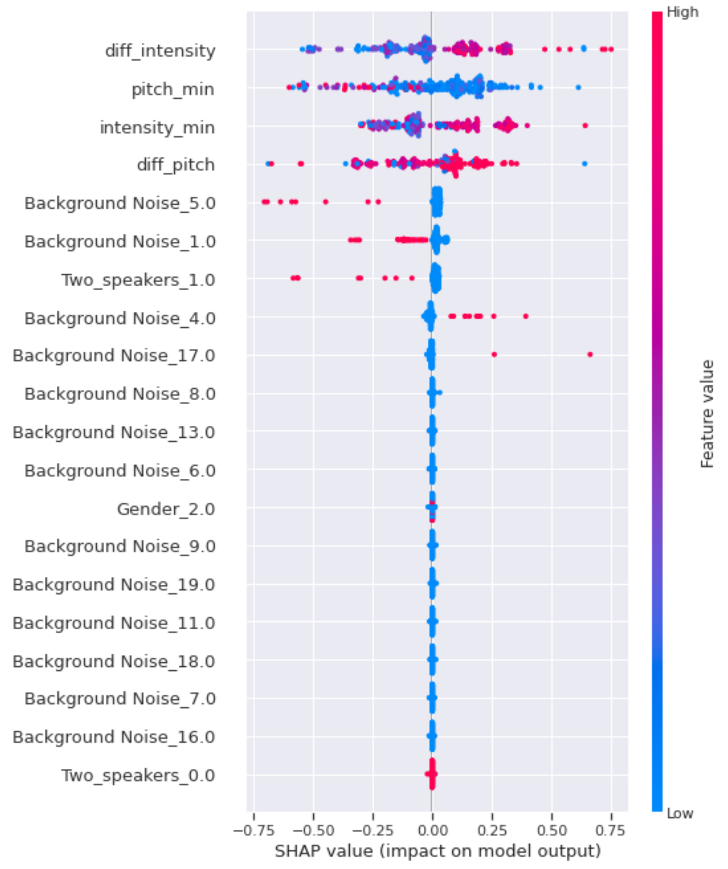


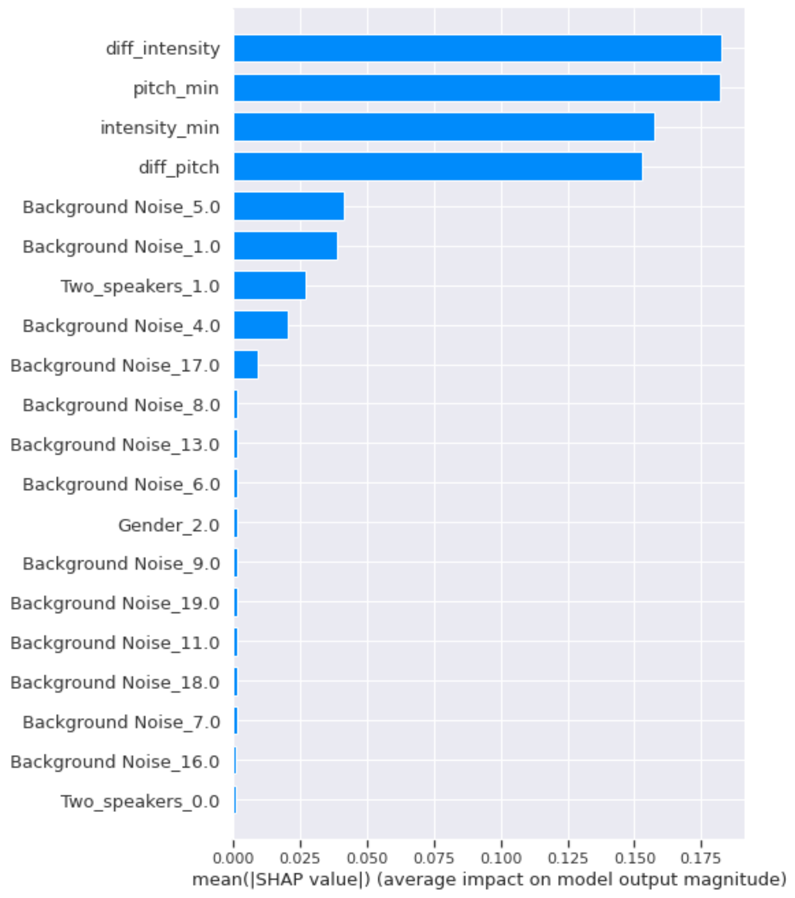
Plot 4.5.2: SHAP values from XGBoost

6. AdaBoost:



Plot 4.6.1: Permutation Importance in AdaBoost





Plot 4.6.2: SHAP values from AdaBoost

CONCLUSION

After evaluating the above results we can conclude that the features having the most importance are:

1. Pitch (min)
2. Pitch (diff)
3. Intensity (min)
4. Intensity (diff)