

# **GENDER RECOGNITION BY VOICE**

Project report submitted in partial fulfilment of the requirement for the  
award of the Degree of B.Tech in  
Computer Science and Engineering

BY

**SPARSH SAXENA**  
**2013662**

**SHIVANI DEOLI**  
**2013690**

Under the Guidance  
of

**RAHUL KUMAR**

SME AI



**Department of Computer Science and Engineering**

**Graphic Era (Deemed to be University)**

**Dehradun-248002**

**2022**





# **GENDER RECOGNITION BY VOICE**

Project report submitted in partial fulfilment of the requirement for the  
award of the Degree of B.Tech in  
Computer Science and Engineering

BY

**SPARSH SAXENA**  
**2013662**

**SHIVANI DEOLI**  
**2013690**

Under the Guidance  
of  
**RAHUL KUMAR**

SME AI



**Department of Computer Science and Engineering**  
**Graphic Era (Deemed to be University)**  
**Dehradun-248002**  
**2022**

## **CERTIFICATE**

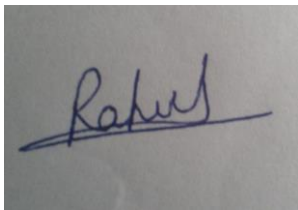
This is to certify that the project report entitled GENDER RECOGNIZATION BY VOICE being submitted by

SPARSH SAXENA (2013662)

SHIVANI DEOLI (2013690)

in partial fulfilment for the award of the Degree of Bachelor of Technology in Computer Science and Engineering (or Information Technology) to the Graphic Era Deemed to be University is a record of bonafied work carried out under my guidance and supervision.

The results embodied in this project report have not been submitted to any other University or Institute for the award of any Degree or Diploma.



Mr. Rahul Kumar

SME AI

Dr. D.P. Singh

Head of Department

## **ACKNOWLEDGEMENT**

We would like to express our sincere gratitude to several individuals and organizations for supporting us throughout our Graduate study. First, we wish to express our sincere gratitude to our mentor, Mr. Rahul Kumar, for his enthusiasm, patience, insightful comments, helpful information, practical advice and unceasing ideas that have helped us tremendously at all times in our research and building of this project. His immense knowledge, profound experience and professional expertise in the field has enabled us to complete this research successfully. Without his support and guidance, this project would not have been possible. We could not have imagined having a better supervisor in our study.

We also wish to express our sincere thanks to Graphic Era University for accepting us into the graduate program.

We are also grateful to the university staff for their consistent support and assistance.

Thanks for all your encouragement!

## **ABSTRACT**

Gender recognition by voice is a technique in which you can determine the gender category of a speaker by processing speech signals. We will be trying to classify gender by voice using machine learning in Python and make a graphical user interface to make the program user friendly.

Speech recognition has various applications including human to machine interaction, sorting of telephone calls by gender categorization, video categorization with tagging and so on. Currently, machine learning is a popular trend which has been widely utilized in various fields and applications, exploiting the recent development in digital technologies and the advantage of storage capabilities from electronic media. Recently, research focuses on the combination of ensemble learning techniques with the semi-supervised learning framework aiming to build more accurate classifiers. In this paper, we focus on gender recognition by voice utilizing a new ensemble semi-supervised self-labelled algorithm. Our preliminary numerical experiments demonstrate the classification efficiency of the proposed algorithm in terms of accuracy, leading to the development of stable and robust predictive models.

## **Contributions**

### **Sparsh Saxena**

- Worked on jupyter notebook to create machine learning model.
- Learned and worked on IBM SPSS Modeler.
- UI Development
- Worked on different app functionalities
- Decided the Front-End application workflow

### **Shivani Deoli**

- Documentation
- Worked on R language

## TABLE OF CONTENT

<b>1. INTRODUCTION.....</b>	<b>1</b>
<b>1.1 AIMS .....</b>	<b>2</b>
<b>1.1.1 Back -end .....</b>	<b>2</b>
<b>1.1.2 Front-end .....</b>	<b>2</b>
<b>1.2 OBJECTIVE .....</b>	<b>2</b>
<b>1.2.1 Application UI .....</b>	<b>2</b>
<b>1.2.2 Machine Learning Model .....</b>	<b>2</b>
<b>2. LITERATURE SURVEY .....</b>	<b>3</b>
<b>2.1 Gender Recognition by Voice Using an Improved Self-Labeled Algorithm.....</b>	<b>3</b>
<b>2.2 Voice Gender Recognition Using Deep Learning.....</b>	<b>5</b>
<b>3. PROBLEM SUMMARY .....</b>	<b>6</b>
<b>3.1 Technology Used / Software Requirements / Modules and their Functionalities.....</b>	<b>6</b>
<b>4. Software Design.....</b>	<b>9</b>
<b>5.1 Hardware Requirement.....</b>	<b>10</b>
<b>5.2 Software Requirement.....</b>	<b>10</b>
<b>6. CODE TEMPLATES .....</b>	<b>13</b>
<b>6.1 Back -End .....</b>	<b>13</b>
<b>6.1.1 Selecting Algorithms for Baseline model .....</b>	<b>13</b>
<b>6.1.2 Jupyter Notebook Model Training.....</b>	<b>16</b>
<b>6.1.3 R Script to Extract Features from Audio File .....</b>	<b>32</b>
<b>6.2 Front-End .....</b>	<b>34</b>
<b>7. TESTING.....</b>	<b>43</b>
<b>7.1. A case where user does not upload any audio. Then a dummy data will be created and user will be notified. ....</b>	<b>43</b>
<b>7.2. A case where user tries to upload a file other than wav format. Our GUI constraints user to upload file other than wav. ....</b>	<b>44</b>
<b>7.3 A case where user tries to save multiple times on the same data.....</b>	<b>44</b>
<b>7.4 A case where user uploads voice of male and our ML model predicts it correctly... </b>	<b>45</b>
<b>7.5 A case where user uploads voice of female and our ML model predicts it correctly.</b>	<b>46</b>
<b>8. OUTPUT SCREENS .....</b>	<b>46</b>
<b>9. CONCLUSIONS .....</b>	<b>51</b>
<b>10. Further Enhancements .....</b>	<b>51</b>



<b>11. References.....</b>	<b>52</b>
----------------------------	-----------

## TABLE OF FIGURES

Figure 1.1 .....	1
Figure 3.1 .....	7
Figure 4.1 .....	9
Figure 6.1 .....	14
Figure 6.2 .....	15
Figure 6.3 .....	15
Figure 6.4 .....	16
Figure 6.5 .....	16
Figure 6.6 .....	17
Figure 6.7 .....	17
Figure 6.8 .....	18
Figure 6.9 .....	19
Figure 6.10 .....	20
Figure 6.11 .....	21
Figure 6.12 .....	21
Figure 6.13 .....	22
Figure 6.14 .....	23
Figure 6.15 .....	23
Figure 6.16 .....	24
Figure 6.17 .....	25
Figure 6.18 .....	26
Figure 6.19 .....	27
Figure 6.20 .....	27
Figure 6.21 .....	28
Figure 6.22 .....	29
Figure 6.23 .....	30
Figure 6.24 .....	31
Figure 6.25 .....	33
Figure 6.26 .....	34
Figure 6.27 .....	35
Figure 6.28 .....	35
Figure 6.29 .....	36
Figure 6.30 .....	37
Figure 6.31 .....	38
Figure 6.32 .....	40
Figure 6.33 .....	41
Figure 7.1 .....	43
Figure 7.2 .....	44
Figure 7.3 .....	45
Figure 7.4 .....	45

Figure 7.5 .....	46
Figure 8.1 .....	47
Figure 8.2 .....	47
Figure 8.3 .....	48
Figure 8.4 .....	48
Figure 8.5 .....	49
Figure 8.6 .....	49
Figure 8.7 .....	50
Figure 8.8 .....	50

# 1. INTRODUCTION

Determining a person's gender as male or female, based upon a sample of their voice seems to initially be an easy task. Often, the human ear can easily detect the difference between a male or female voice within the first few spoken words.

Gender recognition can be useful in many fields, including automatic speech recognition, in which it can help improve the performance of these systems. It can also be used in categorizing calls by gender, or you can add it as a feature to a virtual assistant that is able to distinguish the talker's gender.

The model is constructed using 3,168 recorded samples of male and female voices, speech, and utterances. The samples are processed using acoustic analysis and then applied to an artificial intelligence/machine learning algorithm to learn gender-specific traits. The resulting program achieves 97% accuracy on the test set.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	meanfreq	sd	median	Q25	Q75	IQR	skew	kurt	sp.ent	sfm	mode	centroid	meanfun	minfun	maxfun	meandom	mindom	maxdom	dfrange	modindx	label
2	0.059781	0.064241	0.032027	0.015071	0.090193	0.075122	12.86346	274.4029	0.893369	0.491918	0	0.059781	0.084279	0.015702	0.275862	0.007813	0.007813	0.007813	0	0	male
3	0.066009	0.06731	0.040229	0.019414	0.092666	0.073252	22.42329	634.6139	0.892193	0.513724	0	0.066009	0.107937	0.015826	0.25	0.009014	0.007813	0.054688	0.046875	0.052632	male
4	0.077316	0.083829	0.036718	0.008701	0.131908	0.123207	30.75715	1024.928	0.846389	0.478905	0	0.077316	0.098706	0.015656	0.271186	0.00799	0.007813	0.015625	0.007813	0.046512	male
5	0.151228	0.072111	0.158011	0.096582	0.207955	0.111374	1.232831	4.177296	0.963322	0.727232	0.083878	0.151228	0.088965	0.017798	0.25	0.201497	0.007813	0.5625	0.554688	0.247119	male
6	0.13512	0.079146	0.124656	0.07872	0.206045	0.127325	1.101174	4.333713	0.971955	0.783568	0.104261	0.13512	0.106398	0.016931	0.266667	0.712813	0.007813	5.484375	5.476563	0.208274	male
7	0.132786	0.079557	0.11909	0.067958	0.209592	0.141634	1.932562	8.308895	0.963181	0.738307	0.112555	0.132786	0.110132	0.017112	0.253968	0.298222	0.007813	2.726563	2.71875	0.12516	male
8	0.150762	0.074463	0.160106	0.092899	0.205718	0.112819	1.530643	5.987498	0.967573	0.762638	0.086197	0.150762	0.105945	0.02623	0.266667	0.47962	0.007813	5.3125	5.304688	0.123992	male
9	0.160514	0.076767	0.144337	0.110532	0.231962	0.12143	1.397156	4.766611	0.959255	0.719858	0.128324	0.160514	0.093052	0.017758	0.144144	0.301339	0.007813	0.539063	0.53125	0.283937	male
10	0.142239	0.078018	0.138587	0.088206	0.208587	0.120381	1.099746	4.070284	0.970723	0.770992	0.219103	0.142239	0.096729	0.017957	0.25	0.336476	0.007813	2.164063	2.15625	0.148272	male
11	0.134329	0.08035	0.121451	0.07558	0.201957	0.126377	1.190368	4.78731	0.975246	0.804505	0.011699	0.134329	0.105881	0.0193	0.262295	0.340365	0.015625	4.695313	4.679688	0.08992	male
12	0.157021	0.071943	0.16816	0.10143	0.21674	0.11531	0.979442	3.974223	0.965249	0.733693	0.096358	0.157021	0.088894	0.022069	0.117647	0.460227	0.007813	2.8125	2.804688	0.2	male
13	0.138551	0.077054	0.127527	0.087314	0.202739	0.115426	1.62677	6.291365	0.966004	0.752042	0.012101	0.138551	0.104199	0.019139	0.262295	0.246094	0.007813	2.71875	2.710938	0.132351	male
14	0.137343	0.080877	0.124263	0.083145	0.209227	0.126082	1.378728	5.008952	0.963514	0.73615	0.108434	0.137343	0.092644	0.016789	0.213333	0.481671	0.015625	5.015625	5	0.0885	male
15	0.181225	0.060042	0.190953	0.128839	0.229532	0.100693	1.36943	5.4756	0.937446	0.53708	0.219827	0.181225	0.131504	0.025	0.275862	1.277114	0.007813	2.804688	2.796875	0.14655	male
16	0.183115	0.066882	0.191233	0.129149	0.240152	0.111004	3.568104	35.38475	0.940333	0.571394	0.049987	0.183115	0.102799	0.020833	0.275862	1.245739	0.203125	6.742188	6.539063	0.139332	male
17	0.174272	0.069411	0.190874	0.115602	0.228279	0.112677	4.485038	61.76491	0.950972	0.635199	0.050027	0.174272	0.102046	0.018328	0.246154	1.621299	0.007813	7	6.992188	0.209311	male
18	0.190846	0.06579	0.207951	0.13228	0.244357	0.112076	1.562304	7.83435	0.938546	0.53881	0.050129	0.190846	0.113323	0.017544	0.275862	1.434115	0.007813	6.320313	6.3125	0.25478	male
19	0.171247	0.074872	0.152807	0.122391	0.243617	0.121227	3.20717	25.76556	0.936954	0.58642	0.059958	0.171247	0.079718	0.015671	0.262295	0.106279	0.007813	0.570313	0.5625	0.138355	male
20	0.168346	0.074121	0.145618	0.115756	0.239824	0.124068	2.704335	18.4847	0.934523	0.559742	0.060033	0.168346	0.083484	0.015717	0.231884	0.146563	0.007813	3.125	3.117188	0.059537	male
21	0.173631	0.073352	0.153569	0.12368	0.244234	0.120554	2.804975	20.85754	0.930917	0.518269	0.060027	0.173631	0.09013	0.015702	0.210526	0.193044	0.007813	2.820313	2.8125	0.068124	male
22	0.172754	0.076903	0.177736	0.12007	0.245368	0.125298	2.967765	20.07811	0.925539	0.523081	0.059953	0.172754	0.093574	0.015764	0.2	0.235877	0.007813	0.71875	0.710938	0.235069	male
23	0.181015	0.074369	0.169299	0.128673	0.254175	0.125502	2.587325	12.28143	0.915284	0.475317	0.059957	0.181015	0.098643	0.016145	0.275862	0.209844	0.007813	3.695313	3.6875	0.05994	male
24	0.163536	0.072449	0.145543	0.11393	0.227449	0.113519	3.58765	28.65378	0.927015	0.542422	0.059941	0.163536	0.062542	0.015686	0.197531	0.059622	0.007813	0.445313	0.4375	0.091699	male
25	0.170213	0.075105	0.146053	0.123989	0.250126	0.126137	2.816793	13.76458	0.913832	0.487966	0.059944	0.170213	0.077698	0.015702	0.192771	0.101563	0.007813	0.5625	0.554688	0.161791	male
26	0.160422	0.076615	0.144824	0.120924	0.237244	0.116319	6.253208	85.49193	0.93303	0.567424	0.060078	0.160422	0.098944	0.016097	0.275862	0.206756	0.007813	3.953125	3.945313	0.07389	male
27	0.1647	0.075362	0.147018	0.118698	0.240475	0.121777	4.208608	43.68189	0.940669	0.60402	0.059965	0.1647	0.082963	0.01564	0.253968	0.143353	0.007813	1.0625	1.054688	0.125926	male
28	0.169579	0.075635	0.186468	0.116706	0.238549	0.121843	4.269923	45.89525	0.929498	0.543709	0.059966	0.169579	0.082451	0.016211	0.271186	0.148438	0.007813	3.609375	3.601563	0.050841	male
29	0.169021	0.071778	0.143168	0.125801	0.248315	0.122515	3.079273	14.3403	0.902275	0.477746	0.128148	0.169021	0.130598	0.015842	0.225352	0.335313	0.007813	0.710938	0.703125	0.397354	male
30	0.16734	0.072841	0.141739	0.122174	0.24	0.117826	2.192126	8.15241	0.913763	0.539479	0.134783	0.16734	0.120052	0.016244	0.262295	0.298678	0.007813	0.679688	0.671875	0.384778	male
31	0.180528	0.070867	0.142385	0.129541	0.252477	0.122936	2.799969	12.19036	0.853115	0.313426	0.133578	0.180528	0.126607	0.017039	0.177778	0.234863	0.007813	0.507813	0.5	0.329241	male
32	0.153039	0.074031	0.158065	0.092731	0.214366	0.121634	0.885702	3.523982	0.973218	0.807552	0.216774	0.153039	0.067407	0.01626	0.205128	0.261563	0.007813	1.023438	1.015625	0.191346	male
33	0.166659	0.075961	0.142781	0.122075	0.245077	0.123002	3.281848	16.61948	0.915785	0.560407	0.12702	0.166659	0.12772	0.017279	0.275862	0.274503	0.007813	0.632813	0.625	0.3325	male
34	0.175659	0.071652	0.144192	0.131058	0.256527	0.125469	3.736487	21.66667	0.876749	0.40391	0.134411	0.175659	0.132726	0.016563	0.228571	0.257813	0.007813	0.648438	0.640625	0.203437	male
35	0.174826	0.071533	0.146471	0.123529	0.247059	0.123529	2.576732	10.11012	0.875392	0.436706	0.12	0.174826	0.124685	0.016754	0.25	0.799006	0.007813	4.171875	4.164063	0.205816	male
36	0.174065	0.073122	0.140183	0.127706	0.252844	0.125138	3.26304	16.42405	0.875611	0.414839	0.133211	0.174065	0.124533	0.016343	0.231884	0.244945	0.007813	0.632813	0.625	0.4225	male
37	0.15953	0.075273	0.146853	0.121399	0.239371	0.117972	1.993501	6.870351	0.921132	0.582531	0.133636	0.15953	0.120254	0.018412	0.168421	0.372869	0.007813	2.609375	2.601563	0.216817	male

Figure 1.1

## **1.1 AIMS**

**1.1.1 Back -end** - Create and deploy/export machine learning model trained on provided dataset of acoustic measures of voice (males/females) to use in front-end.

**1.1.2 Front-end** - Create a simple UI (User Interface) through which user will interact with our application and use the different functionalities present in our application.

## **1.2 OBJECTIVE**

### **1.2.1 Application UI**

- a) User interface should be simple and light weight.
- b) Less number of clicks and keystrokes should be required to accomplish the task.
- c) Interface design should be intuitive. Intuitive user interface design is one that is easy to learn so that user can pick it up quickly and easily.

### **1.2.2 Machine Learning Model**

- a) The model should be light weight.
- b) The model should be able to process voice and give result.

## **2. LITERATURE SURVEY**

### **2.1 Gender Recognition by Voice Using an Improved Self-Labeled Algorithm**

Research work done by Ioannis E. Livieris, Emmanuel Pintelas and Panagiotis Pintelas. Speech constitutes one of the most popular and significant means for humans to communicate, express their emotions, cognitive states, and intentions to each other. Speech is produced by humans using a natural biological mechanism in which lungs discharge the air and convert it to speech passing through the vocal cords and organs including the tongue, teeth, lips etc. In general, a speech and voice recognition system can be used for gender identification. A natural voice recognition system is the human ear. The human ear has an excellent mechanism which can efficiently distinguish the gender by voice and speech based on attributes like frequency and loudness. In a similar way, a machine can be taught to do the same thing by choosing and incorporating the right features from voice data on a machine learning algorithm.

Gender recognition is a technique which is often utilized to determine the gender category of a speaker by processing speech signals. Speech signals taken from a recorded speech can be used to acquire acoustic attributes such as duration, intensity, frequency, and filtering. Some applications where gender recognition can be useful are speech emotion recognition, human to machine interaction, sorting of telephone calls by gender categorization, automatic salutations, muting sounds for a gender and audio/video categorization with tagging.

As technology is growing in a rapid way, machine learning is a research field which has had major developments; thus it has been widely established as a popular trend. Machine learning is a subset of artificial intelligence which utilizes algorithms and data to teach computers make decisions on specific problems in various fields like finance, banking, and medicine etc. Along this line, several studies for gender recognition and identification by voice using machine learning and data mining techniques have been conducted. However, the development of an accurate prediction model for gender recognition by voice is still considered a rather difficult and challenging task. In 2006, Vogt, T.; André, E. Improving automatic emotion recognition from speech via gender differentiation. In Proceedings of the Language Resources and Evaluation Conference, Genoa, Italy. and Harb, H.; Chen,

L. A general audio classifier based on human perception motivated model. *Multimed. Tools Appl.* in 2007 conducted an extensive experimental analysis and pointed out the difficulties of this classification problem since speech signals are highly time-varying and have very high randomness. This is mainly due to the fact that the progress in the field has been hampered by the lack of available labeled data for efficiently training a supervised classifier. Furthermore, in order to train efficiently a classifier and be able to make accurate predictions, it often needs a large amount of labeled data. Nevertheless, the process of finding sufficient labeled data for training classifiers to make accurate predictions is often an expensive and time-consuming task as it requires human efforts, while in contrast finding unlabeled data in general is significantly easier. To address the problem of insufficient labeled data, semi-supervised learning (SSL) algorithms constitute the appropriate methodology to exploit the hidden information found in the unlabeled set aiming to build more accurate classifiers. In the literature, several classes of SSL algorithms have been proposed and evaluated, each of them based on different methodologies and techniques related to the link between the distribution of labeled and unlabeled data. Self-labeled algorithms probably constitute the most popular and widely utilized class of SSL algorithms. The algorithms of this class follow an iterative procedure, augmenting an initial labeled dataset using their own predictions from a large pool of unlabeled datasets. Triguero et al. [14] proposed an in-depth taxonomy of self-labeled algorithms based on their main characteristics and made exhaustive research of their classification efficacy on various datasets.

Ensemble learning (EL) is another way of obtaining better results for a higher classification accuracy, which has been developed over the last decades. The main object of this methodology is the combination of several prediction models, in order to build a more accurate model rather than using a single one. Furthermore, the development of algorithms which hybridize SSL and EL approaches is another recent methodology which can be beneficial to each other and can build more robust classification algorithms, leading to even better classification results.

In this work, we propose a new ensemble-based self-labeled algorithm, called iCST-Voting, for gender recognition by voice. This algorithm combines the individual predictions of three of the most popular and efficient self-labeled

methods i.e., Co-training, Self-training, and Tri-training utilizing an ensemble as base learner. Our experimental results reveal the efficiency of this algorithm compared against state-of-the-art self-labeled algorithms.

The remainder of this paper is organized as follows: Section 2 presents a synopsis of related work on gender recognition by voice. Section 3 presents a brief description of the self-labeled algorithms. Section 4 presents the proposed classification algorithm. Section 5 presents the datasets and our experimental results. Finally, Section 6 presents our concluding remarks and some directions for future research.

## **2.2 Voice Gender Recognition Using Deep Learning**

Research work done by Mucahit Buyukyilmaz<sup>1</sup>, and Ali Osman Cibikdiken. In this article, a Multilayer Perceptron (MLP) deep learning model has been described to recognize voice gender. The data set have 3,168 recorded samples of male and female voices. The samples are produced by using acoustic analysis. An MLP deep learning algorithm has been applied to detect genderspecific traits. Our model achieves 96.74% accuracy on the test data set. Also, the interactive web page has been built for recognition gender of voice.

Acoustic analysis of the voice depends on parameter settings specific to sample characteristics such as intensity, duration, frequency, and filtering. The acoustic properties of the voice and speech can be used to detect gender of speaker. warbleR R package is designed for acoustic analysis. The data set which has acoustic parameters can be obtained with this analysis. The data set can be trained with different machine learning algorithms. In this paper, MLP has been used to obtain model. The results have been compared with related work. A web page has been designed to detect the gender of voice by using obtained model.

Used a frequency-based baseline model, logistic regression model, classification, and regression tree (CART) model, random forest model, boosted tree model, Support Vector Machine (SVM) model, XGBoost model, stacked model for recognition of voices data set.



### **3. PROBLEM SUMMARY**

The main aim of the project is to recognize gender of the person by using his/her voice. For this purpose, we will be working on both back-end and front-end for ease of the user. In back-end we will be training a machine learning model using various machine learning algorithms on historical data then making an executable application or a web application for the front-end.

#### **3.1 Technology Used / Software Requirements / Modules and their Functionalities**

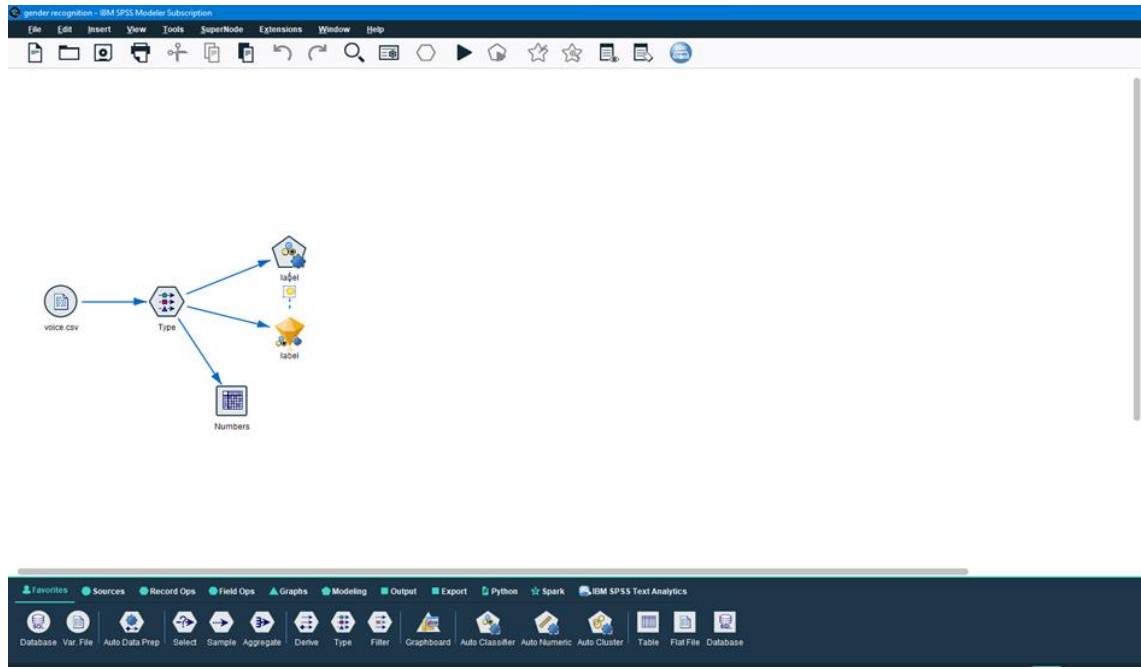
**a) Jupyter Notebook** - A Jupyter Notebook document is a browser-based REPL containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots, and rich media.

In this project, we used jupyter notebook for building back-end from scratch. Used jupyter notebook for training machine learning model using various machine learning algorithms.

**b) IBM SPSS Modeler** - SPSS Modeler is a leading visual data science and machine learning (ML) solution designed to help enterprises accelerate time to value by speeding up operational tasks for data scientists. Organizations worldwide use it for data preparation and discovery, predictive analytics, model management and deployment, and ML to monetize data assets.

IBM SPSS Modeler is a data mining and text analytics software application from IBM. It is used to build predictive models and conduct other analytic tasks. It has a visual interface which allows users to leverage statistical and data mining algorithms without programming

In this project, we used IBM SPSS Modeler for analyzing different machine learning model and selecting the best algorithms for training our machine learning model in jupyter notebook.



*Figure 3.1*

- c) **Tkinter** - Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.
- d) **Python3** - Python is a high-level, interpreted, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features such as list comprehensions, cycle-detecting garbage collection, reference counting, and Unicode support. Python 3.0, released in 2008, was a major revision that is not

completely backward compatible with earlier versions. Python 2 was discontinued with version 2.7.18 in 2020.

- e) **R** - R is a programming language for statistical computing and graphics supported by the R Core Team and the R Foundation for Statistical Computing. Created by statisticians Ross Ihaka and Robert Gentleman, R is used among data miners and statisticians for data analysis and developing statistical software. Users have created packages to augment the functions of the R language.

According to user surveys and studies of scholarly literature databases, R is one of the most commonly used programming languages used in data mining.

- f) **Warble R** - Acoustic analysis of the voice depends on parameter settings specific to sample characteristics such as intensity, duration, frequency, and filtering. The acoustic properties of the voice and speech can be used to detect gender of speaker. warbleR R package is designed for acoustic analysis.

- g) **Visual Studio Code** - Visual Studio Code is a source-code editor that can be used with a variety of programming languages, including Java, JavaScript, Go, Node.js, Python, C++, and Fortran.

#### 4. Software Design

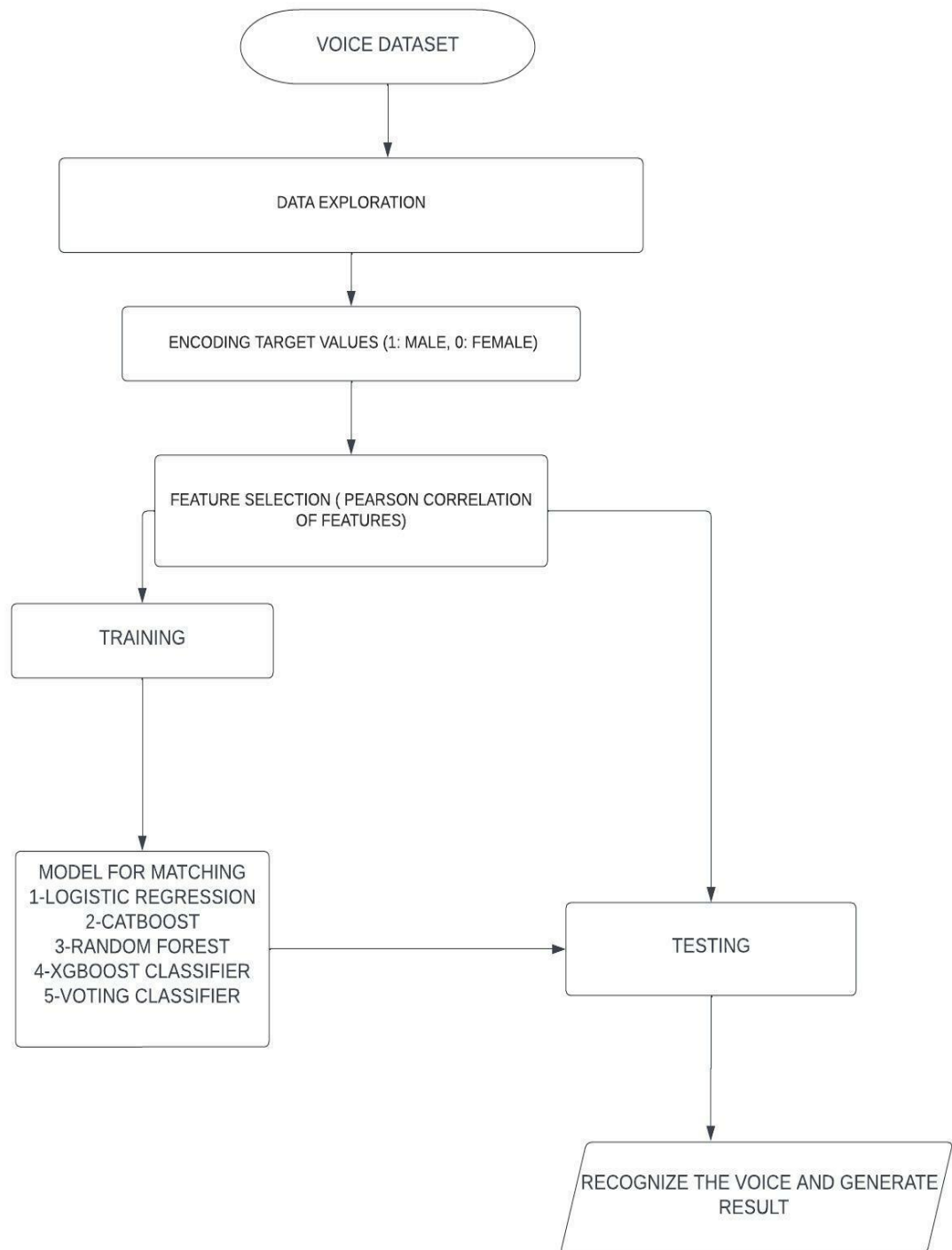


Figure 4.1

## 5. SOFTWARE AND HARDWARE REQUIREMENTS

The project was built on ubuntu environment using jupyter notebook running on visual studio code with python 3.10 kernel. Some of the modules and their functions will be explained further in this report.

### 5.1 Hardware Requirement

The project was built and compiled on a system with following specification

- RAM – 16 Gigabytes
- PROCESSOR – Intel Core i7-9750H (2.6 gigahertz) (6 cores/12 threads)
- STORAGE – 1 Terabyte (NVME Solid State Drive)

### 5.2 Software Requirement

The project was built using the following software (either already present or installed explicitly) –

- UBUNTU 22.04 (Dual Boot System)
- Visual Studio Code (Snap Version)
- Python 3.10
- Jupyter Notebook
- R

### 5.3 Some Module Installation

The main reason to choose ubuntu as the development environment was because of the availability of vast software for development and easiness to install.

Software and packages can be installed by running following commands in ubuntu terminal

- 1) **Python 3.10** – Python is pre- installed software in ubuntu operating system.
- 2) **Python-pip** - pip is the package installer for Python. You can use pip to install packages from the Python Package Index and other indexes. Python-pip can be installed using the command:  

```
$ sudo apt install python3-pip
```
- 3) **Pandas** - pandas is a software library written for the Python programming language for data manipulation and analysis. Pandas library can be installed using the command:

\$ pip install pandas

- 4) **Sklearn** - Scikit-learn also known as sklearn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting, k-means and many more.

Sklearn can be installed using the command:

\$ pip install sklearn

- 5) **Catboost** - Catboost is an open-source software library developed by Yandex. It provides a gradient boosting framework which among other features attempts to solve for Categorical features using a permutation driven alternative compared to the classical algorithm. Catboost can be installed using the command:

\$ pip install Catboost

- 6) **XGBoost** – XGBoost (eXtreme Gradient Boosting) is an open-source software library which provides a regularizing gradient boosting framework. XGBoost can be installed using the command:

\$ pip install xgboost

- 7) **Libportaudio2** - PortAudio is a portable audio I/O library designed for cross-platform support of audio. It uses a callback mechanism to request audio processing. Audio can be generated in various formats, including 32-bit floating point, and will be converted to the native format internally. Libportaudio2 can be installed using the command:

\$ sudo apt-get install libportaudio2

- 8) **Sounddevice** - This Python module provides bindings for the PortAudio library and a few convenience functions to play and record NumPy arrays containing audio signals. Sounddevice can be installed using the command:

\$ pip install sounddevice

- 9) **Scipy** - SciPy (Science Python) is a free and open-source Python library used for scientific computing and technical computing. Scipy can be installed using the command:

\$ pip install scipy

**10) Numpy** – Numpy (Numerical Python) is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. Numpy can be installed using the command:

```
$ pip install numpy
```

**11) R** - We need R language to extract the acoustic properties of voice which will be recorded by the user. R language can be installed in ubuntu by following commands:

```
$ sudo apt install r-base
```

```
$ sudo apt install gfortran libsndfile1-dev libfftw3-dev
```

```
$ R
```

With the last command you just opened R console, so you can run the follows command to install packages:

```
> install.packages("tuneR")
```

```
> install.packages("seewave")
```

```
> install.packages("pbapply")
```

```
> install.packages("fftw")
```

```
> q()
```

Note: q() is to close the "r console".

**12) Tkinter** - Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit. Tkinter can be installed using the command:

```
$ sudo apt-get install python3-tk
```

## **6. CODE TEMPLATES**

### **6.1 Back -End**

The back-end of our project consists of training our machine learning model using different technologies and different machine learning algorithms.

#### **6.1.1 Selecting Algorithms for Baseline model**

To analyze gender by voice and speech, a training database was required. A database was built using thousands of samples of male and female voices, each labeled by their gender of male or female.

Each voice sample is stored as a .WAV file, which is then pre-processed for acoustic analysis using the specan function from the WarbleR R package. Specan measures 22 acoustic parameters on acoustic signals for which the start and end times are provided.

The output from the pre-processed WAV files were saved into a CSV file, containing 3168 rows and 21 columns (20 columns for each feature and one label column for the classification of male or female).

The data we collected to train our machine learning model was preprocessed and cleaned that is, there were no null values, invalid values, outlier/noise.

Some of the Acoustic properties measured were-:



Data frame with the following acoustic parameters:

- ``duration``: length of signal
- ``meanfreq``: mean frequency (In kHz)
- ``sd``: standard deviation of frequency
- ``median``: median frequency (In kHz)
- ``q25``: first quantile (In kHz)
- ``q75``: third quantile (In kHz)
- ``Iqr``: Interquantile range (In kHz)
- ``skew``: skewness (see note in ``specprop`` description)
- ``kurt``: kurtosis (see note in ``specprop`` description)
- ``sp.ent``: spectral entropy
- ``sfn``: spectral flatness
- ``mode``: mode frequency
- ``centroid``: frequency centroid (see ``specprop``)
- ``peakf``: peak frequency (frequency with highest energy)
- ``meanfun``: average of fundamental frequency measured across acoustic signal
- ``minfun``: minimum fundamental frequency measured across acoustic signal
- ``maxfun``: maximum fundamental frequency measured across acoustic signal
- ``meandom``: average of dominant frequency measured across acoustic signal
- ``mindom``: minimum of dominant frequency measured across acoustic signal
- ``maxdom``: maximum of dominant frequency measured across acoustic signal
- ``dfrange``: range of dominant frequency measured across acoustic signal
- ``modindx``: modulation Index. Calculated as the accumulated absolute difference between adjacent measurements of fundamental frequencies divided by the frequency range

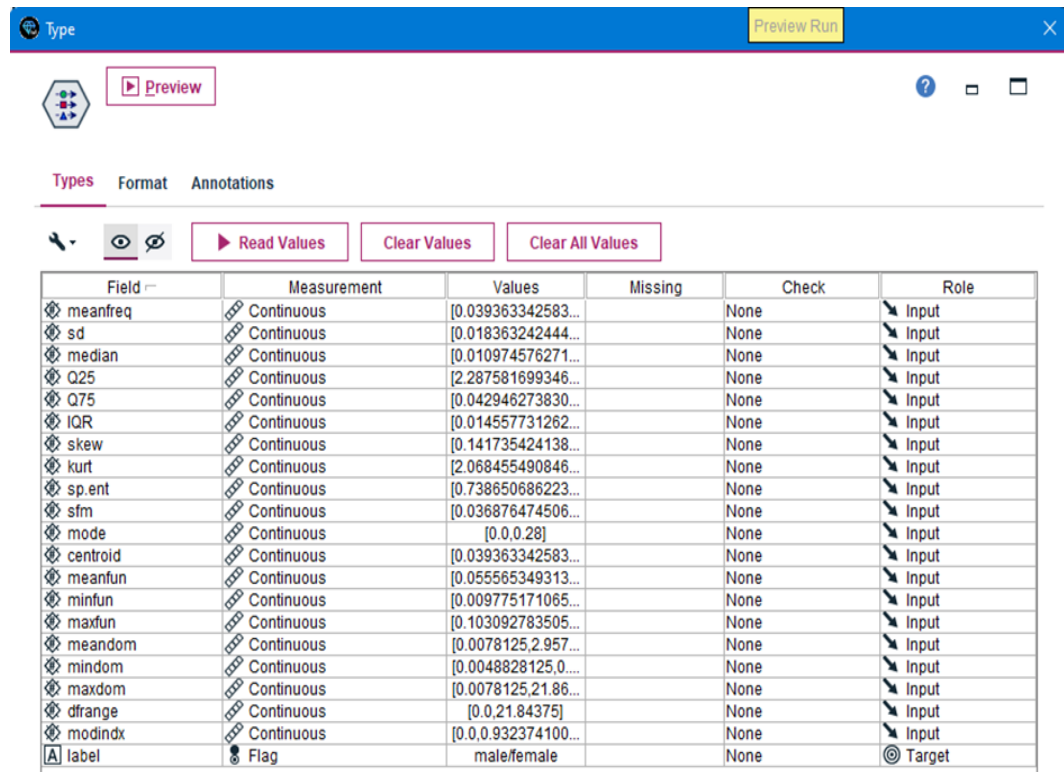
*Figure 6.1*

To select appropriate machine learning model and give ourselves a head start, we used IBM SPSS Modeler.

Steps involved to select machine learning algorithm in SPSS Modeler -:

1. Drag and drop source file (voice.csv) in stream canvas in SPSS Modeler.

2. Drag and drop 'type node' from 'Field Ops' palette to select data type of features available in dataset. And set 'label' feature role as target and measurement as Flag.



Field	Measurement	Values	Missing	Check	Role
meanfreq	Continuous	[0.039363342583...		None	Input
sd	Continuous	[0.01836324244...		None	Input
median	Continuous	[0.010974576271...		None	Input
Q25	Continuous	[2.287581699346...		None	Input
Q75	Continuous	[0.042946273830...		None	Input
IQR	Continuous	[0.014557731262...		None	Input
skew	Continuous	[0.141735424138...		None	Input
kurt	Continuous	[2.068455490846...		None	Input
sp.ent	Continuous	[0.738650686223...		None	Input
sfm	Continuous	[0.036876474506...		None	Input
mode	Continuous	[0.0,0.28]		None	Input
centroid	Continuous	[0.039363342583...		None	Input
meanfun	Continuous	[0.055565349313...		None	Input
minfun	Continuous	[0.009775171065...		None	Input
maxfun	Continuous	[0.103092783505...		None	Input
meandom	Continuous	[0.0078125,2.957...		None	Input
mindom	Continuous	[0.0048828125,0...		None	Input
maxdom	Continuous	[0.0078125,21.86...		None	Input
dfrange	Continuous	[0.0,21.84375]		None	Input
modindx	Continuous	[0.0,0.932374100...		None	Input
label	Flag	male/female		None	Target

Figure 6.2

3. Link source node and type node for data flow from one node to another node.
4. Drag and drop 'Auto Classifier' node in stream canvas from 'Modeling' palette to train and select appropriate machine learning algorithm.
5. Link 'type' node and 'Auto Classifier' Node to complete our stream.
6. Click on run button from the menu. The stream will look like as shown below.

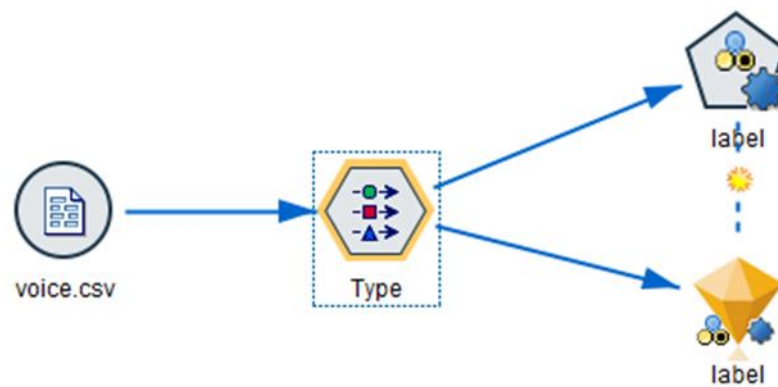
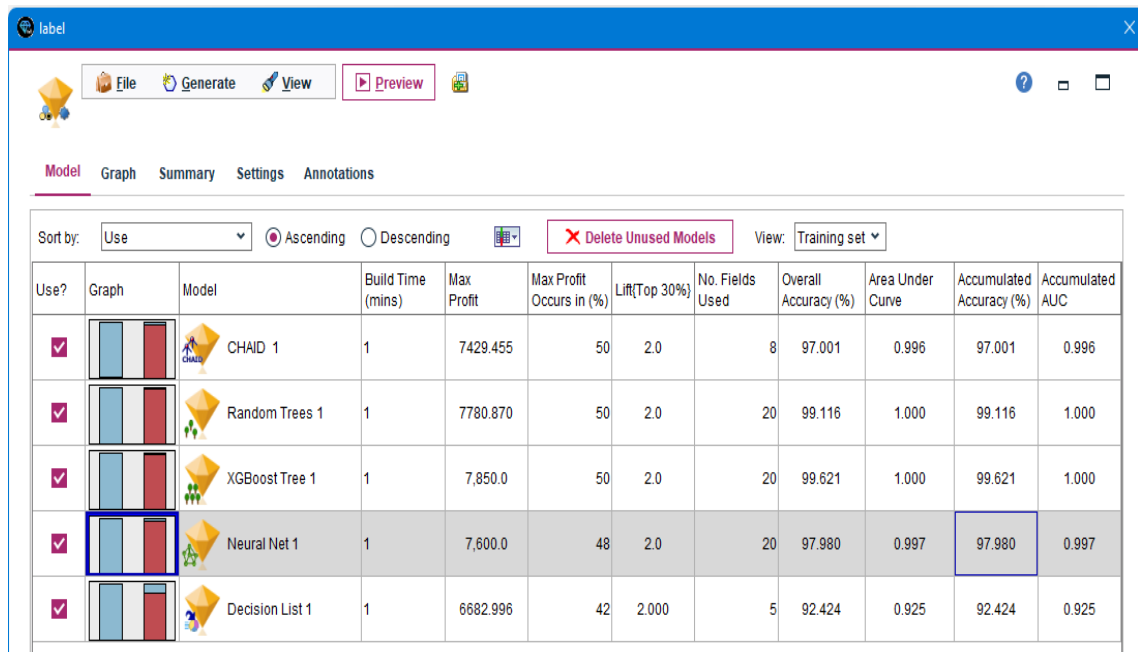


Figure 6.3

7. After the stream is executed, machine learning models with most accuracy are as shown:



Use?	Graph	Model	Build Time (mins)	Max Profit	Max Profit Occurs in (%)	Lift(Top 30%)	No. Fields Used	Overall Accuracy (%)	Area Under Curve	Accumulated Accuracy (%)	Accumulated AUC
<input checked="" type="checkbox"/>		CHAID 1	1	7429.455	50	2.0	8	97.001	0.996	97.001	0.996
<input checked="" type="checkbox"/>		Random Trees 1	1	7780.870	50	2.0	20	99.116	1.000	99.116	1.000
<input checked="" type="checkbox"/>		XGBoost Tree 1	1	7,850.0	50	2.0	20	99.621	1.000	99.621	1.000
<input checked="" type="checkbox"/>		Neural Net 1	1	7,600.0	48	2.0	20	97.980	0.997	97.980	0.997
<input checked="" type="checkbox"/>		Decision List 1	1	6682.996	42	2.000	5	92.424	0.925	92.424	0.925

Figure 6.4

After selecting the appropriate machine learning model, we trained and exported our machine learning model using jupyter notebook.

### 6.1.2 Jupyter Notebook Model Training

Steps involved in training our machine learning using jupyter notebook were:

#### 1) Importing Important Libraries

IMPORTANT LIBRARIES

```
In [ ]: import pandas as pd
        from sklearn.model_selection import train_test_split
        from sklearn.preprocessing import LabelEncoder
        from sklearn.preprocessing import MinMaxScaler

In [ ]: from catboost import CatBoostClassifier
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.linear_model import LogisticRegression
        from sklearn.ensemble import VotingClassifier
        from xgboost import XGBClassifier
```

Figure 6.5

## 2) Loading Dataset in Jupyter Notebook

IMPORTING DATASET

```
In [ ]: df = pd.read_csv('./voice.csv')
```

Figure 6.6

## 3) Exploring Data

DATA EXPLORATION

```
In [ ]: df
```

Out[ ]:

	meanfreq	sd	median	Q25	Q75	IQR	skew	kurt	sp.ent	sfm	...	centroid	meanfun
0	0.059781	0.064241	0.032027	0.015071	0.090193	0.075122	12.863462	274.402906	0.893369	0.491918	...	0.059781	0.084279
1	0.066009	0.067310	0.040229	0.019414	0.092666	0.073252	22.423285	634.613855	0.892193	0.513724	...	0.066009	0.107937
2	0.077316	0.083829	0.036718	0.008701	0.131908	0.123207	30.757155	1024.927705	0.846389	0.478905	...	0.077316	0.098706
3	0.151228	0.072111	0.158011	0.096582	0.207955	0.111374	1.232831	4.177296	0.963322	0.727232	...	0.151228	0.088965
4	0.135120	0.079146	0.124656	0.078720	0.206045	0.127325	1.101174	4.333713	0.971955	0.783568	...	0.135120	0.106398
...	...	...	...	...	...	...	...	...	...	...	...	...	...
3163	0.131884	0.084734	0.153707	0.049285	0.201144	0.151859	1.762129	6.630383	0.962934	0.763182	...	0.131884	0.182790
3164	0.116221	0.089221	0.076758	0.042718	0.204911	0.162193	0.693730	2.503954	0.960716	0.709570	...	0.116221	0.188980
3165	0.142056	0.095798	0.183731	0.033424	0.224360	0.190936	1.876502	6.604509	0.946854	0.654196	...	0.142056	0.209918
3166	0.143659	0.090628	0.184976	0.043508	0.219943	0.176435	1.591065	5.388298	0.950436	0.675470	...	0.143659	0.172375
3167	0.165509	0.092884	0.183044	0.070072	0.250827	0.180756	1.705029	5.769115	0.938829	0.601529	...	0.165509	0.185607

3168 rows × 21 columns

Figure 6.7

```

In [ ]: df.isnull().sum()

Out[ ]: meanfreq      0
        sd            0
        median        0
        Q25           0
        Q75           0
        IQR           0
        skew          0
        kurt          0
        sp.ent        0
        sfm           0
        mode          0
        centroid      0
        meanfun       0
        minfun        0
        maxfun        0
        meandom       0
        mindom        0
        maxdom        0
        dfrange       0
        modindx       0
        label         0
        dtype: int64

In [ ]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3168 entries, 0 to 3167
Data columns (total 21 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   meanfreq    3168 non-null   float64
1   sd          3168 non-null   float64
2   median      3168 non-null   float64
3   Q25         3168 non-null   float64
4   Q75         3168 non-null   float64
5   IQR         3168 non-null   float64
6   skew        3168 non-null   float64
7   kurt        3168 non-null   float64
8   sp.ent      3168 non-null   float64
9   sfm         3168 non-null   float64
10  mode        3168 non-null   float64
11  centroid    3168 non-null   float64
12  meanfun     3168 non-null   float64
13  minfun      3168 non-null   float64
14  maxfun      3168 non-null   float64
15  meandom     3168 non-null   float64
16  mindom      3168 non-null   float64
17  maxdom      3168 non-null   float64
18  dfrange     3168 non-null   float64
19  modindx     3168 non-null   float64
20  label       3168 non-null   object
dtypes: float64(20), object(1)
memory usage: 519.9+ KB

```

Figure 6.8

```
In [ ]: df.shape
Out[ ]: (3168, 21)

In [ ]: df.label
Out[ ]: 0      male
1      male
2      male
3      male
4      male
...
3163   female
3164   female
3165   female
3166   female
3167   female
Name: label, Length: 3168, dtype: object
```

*Figure 6.9*

Here we can see we don't have any null values, invalid values, and any outlier. After exploring the dataset, we can see that our independent variables/features (20 columns) are continuous and our dependent variable/feature or known as target variable is binary (male or female). So, we can convert it into 0 or 1, where 1 signifies Male and 0 signifies Female.

#### **4) Encoding Labels**

To achieve the above, we need to perform encoding and convert our binary classification in string to 0 or 1. The process of encoding "label" column is shown below.

```

In [ ]: encoding_columns = [ "label"]
Encoder = LabelEncoder()
for column in encoding_columns :
    print("column",column )
    df[ column ] = Encoder.fit_transform(tuple(df[ column ]))

column label

In [ ]: df

Out[ ]:

```

	kurt	sp.ent	sfm	...	centroid	meanfun	minfun	maxfun	meandom	mindom	maxdom	dfrange	modindx	label
	274.402906	0.893369	0.491918	...	0.059781	0.084279	0.015702	0.275862	0.007812	0.007812	0.007812	0.000000	0.000000	1
	634.613855	0.892193	0.513724	...	0.066009	0.107937	0.015826	0.250000	0.009014	0.007812	0.054688	0.046875	0.052632	1
	1024.927705	0.846389	0.478905	...	0.077316	0.098706	0.015656	0.271186	0.007990	0.007812	0.015625	0.007812	0.046512	1
	4.177296	0.963322	0.727232	...	0.151228	0.088965	0.017798	0.250000	0.201497	0.007812	0.562500	0.554688	0.247119	1
	4.333713	0.971955	0.783568	...	0.135120	0.106398	0.016931	0.266667	0.712812	0.007812	5.484375	5.476562	0.208274	1
	...	...	...	...	...	...	...	...	...	...	...	...	...	...
	6.630383	0.962934	0.763182	...	0.131884	0.182790	0.083770	0.262295	0.832899	0.007812	4.210938	4.203125	0.161929	0
	2.503954	0.960716	0.709570	...	0.116221	0.188980	0.034409	0.275862	0.909856	0.039062	3.679688	3.640625	0.277897	0
	6.604509	0.946854	0.654196	...	0.142056	0.209918	0.039506	0.275862	0.494271	0.007812	2.937500	2.929688	0.194759	0
	5.388298	0.950436	0.675470	...	0.143659	0.172375	0.034483	0.250000	0.791360	0.007812	3.593750	3.585938	0.311002	0
	5.769115	0.938829	0.601529	...	0.165509	0.185607	0.062257	0.271186	0.227022	0.007812	0.554688	0.546875	0.350000	0

Figure 6.10

Here, in label column we can see that all Male is converted into 1 and all Female value is converted into 0.

## 5) Normalizing the data

Normalization is a technique often applied as part of data preparation for machine learning. The goal of normalization is to change the values of numeric columns in the dataset to use a common scale, without distorting differences in the ranges of values or losing information. Normalization is also required for some algorithms to model the data correctly.

For example, assume your input dataset contains one column with values ranging from 0 to 1, and another column with values ranging from 10,000 to 100,000. The great difference in the scale of the numbers could cause problems when you attempt to combine the values as features during modeling.



Normalization avoids these problems by creating new values that maintain the general distribution and ratios in the source data, while keeping values within a scale applied across all numeric columns used in the model.

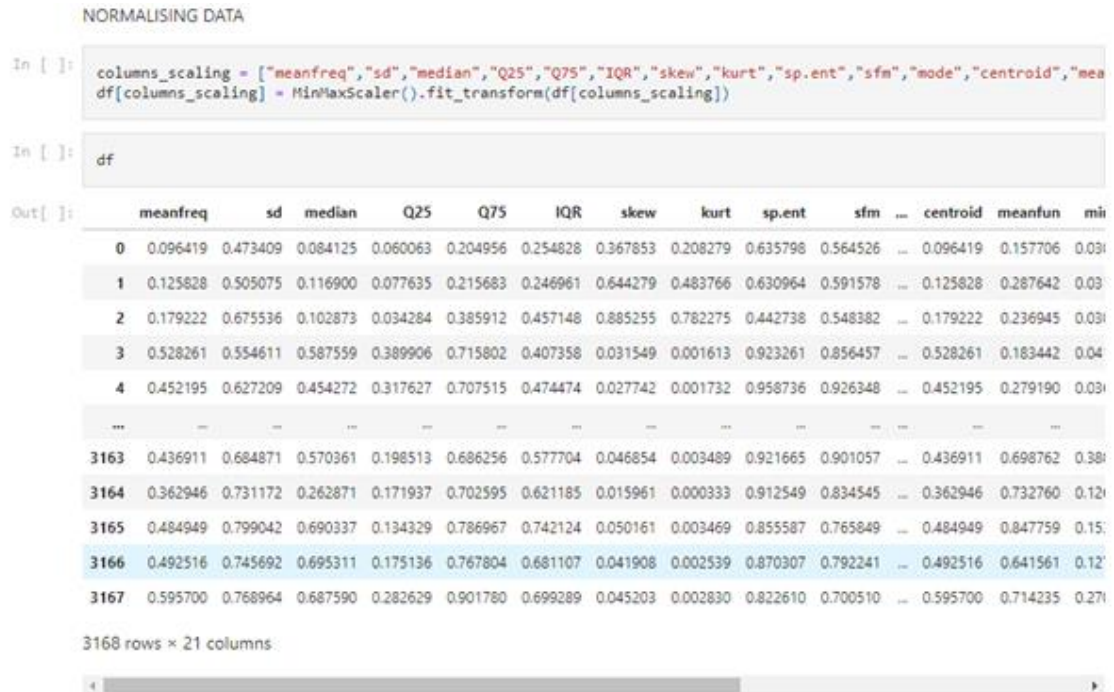


Figure 6.11

Here we can see our data is normalized between 0 and 1 using MinMaxScaler.

## 6) Feature Selection

Feature selection is the process of reducing the number of input variables when developing a predictive model. It is desirable to reduce the number of input variables to both reduce the computational cost of modeling and, in some cases, to improve the performance of the model.



Figure 6.12



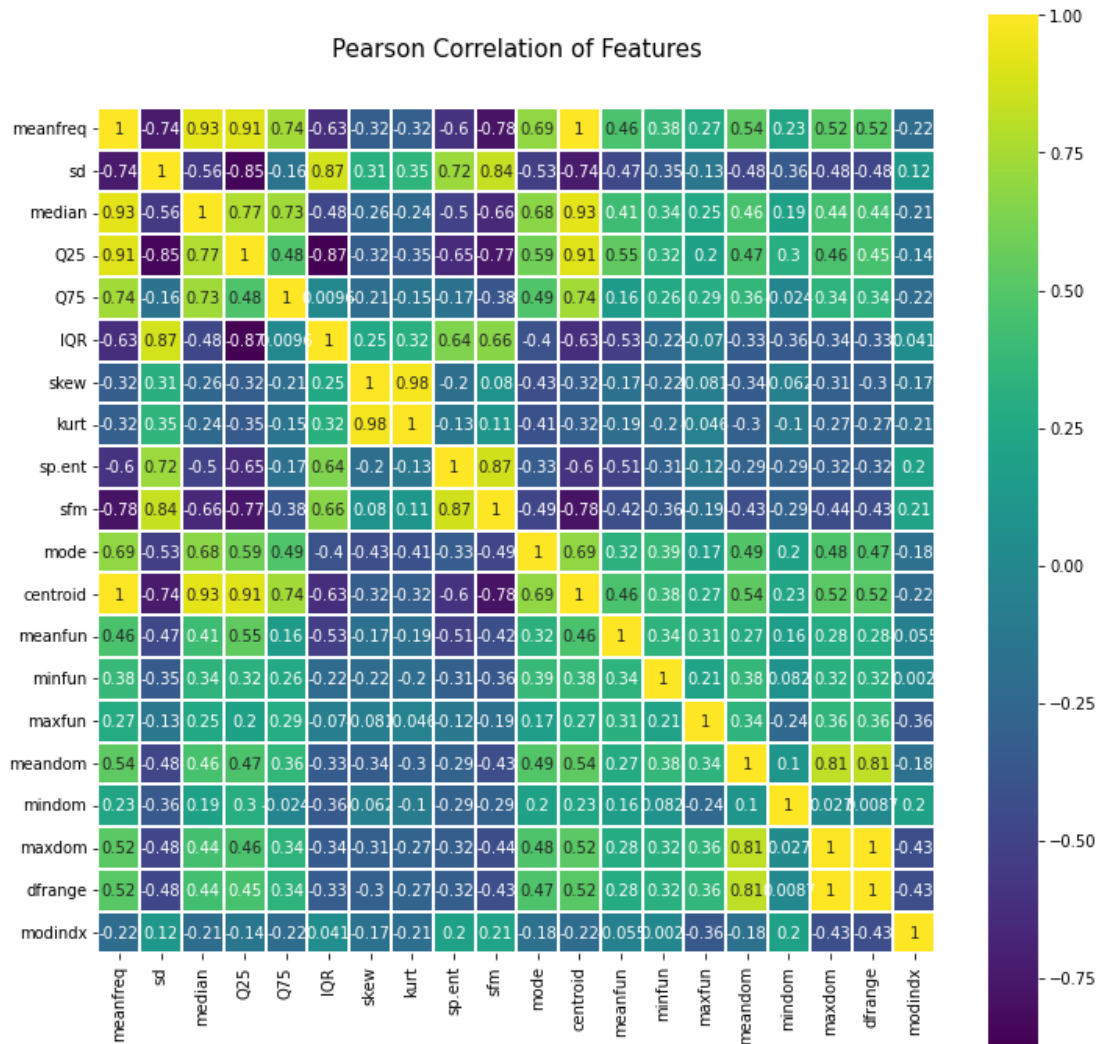


Figure 6.13

While looking at the plot, we can figure out some interesting correlations. If you look at meanfreq vs centroid their correlation is maximum possible value of 1. Same is the case with maxdom and dfrange. So essentially, we could filter out these features and still get an equivalent performance as they aren't adding any new information.

Pearson's Correlation method is used for finding the association between the continuous features and the class feature.

```
df = df.drop('centroid',axis=1)

df = df.drop('maxdom',axis=1)
```

Figure 6.14

7) Splitting data into test and train using train\_test\_split

We performed splitting in our dataset in train and test. Ratio (train: test::0.7: 0.3) or train = 70% of data and test = 30% of data.

```
In [ ]: x = df.drop("label",axis=1)
        y = df["label"]

In [ ]: X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=2)

In [ ]: X_test.shape

Out[ ]: (951, 20)

In [ ]: X_train

Out[ ]:
```

	meanfreq	sd	median	Q25	Q75	IQR	skew	kurt	sp.ent	sfm	mode	centroid	meanfur
1876	0.719393	0.274532	0.717657	0.651222	0.756676	0.175297	0.051910	0.004514	0.707063	0.301095	0.630225	0.719393	0.561524
1564	0.553560	0.451480	0.473543	0.445432	0.770681	0.402854	0.091584	0.012915	0.750398	0.553083	0.399394	0.553560	0.331706
2336	0.927098	0.207848	0.923945	0.925923	0.935628	0.063245	0.076286	0.007131	0.329248	0.208454	0.890411	0.927098	0.660146
308	0.741660	0.403821	0.739527	0.575573	0.909004	0.401702	0.071391	0.008560	0.637927	0.182622	0.000000	0.741660	0.425192
2844	0.875210	0.154105	0.860907	0.881289	0.846021	0.022741	0.086357	0.008939	0.218481	0.183751	0.791745	0.875210	0.738225
...	...	...	...	...	...	...	...	...	...	...	...	...	...
2514	0.734667	0.165525	0.734975	0.735009	0.720207	0.052805	0.083562	0.008928	0.449808	0.269678	0.705882	0.734667	0.627973
2347	0.877997	0.162214	0.869386	0.859938	0.885949	0.083668	0.057026	0.004173	0.443190	0.206904	0.768452	0.877997	0.608316
1608	0.393571	0.689842	0.538519	0.128690	0.677488	0.641799	0.038794	0.002080	0.831621	0.724009	0.691134	0.393571	0.723091
2541	0.746960	0.169346	0.750087	0.738239	0.762467	0.090436	0.067016	0.005735	0.414524	0.235839	0.660944	0.746960	0.648486
2575	0.911841	0.135529	0.906858	0.880092	0.906623	0.082766	0.057236	0.004214	0.329327	0.134200	0.865731	0.911841	0.713886

2217 rows x 20 columns

Figure 6.15

```

In [ ]: y_train
Out[ ]: 1876    0
        1564    1
        2336    0
        308     1
        2844    0
        ..
        2514    0
        2347    0
        1608    0
        2541    0
        2575    0
        Name: label, Length: 2217, dtype: int64

```

*Figure 6.16*

## 8) Training Machine Learning Model

In this step, we trained our machine learning model based on the model we choose using IBM SPSS Modeler to achieve maximum accuracy.

```

cat = CatBoostClassifier()
randomforest = RandomForestClassifier()
xgb = XGBClassifier()
lgr = LogisticRegression()

```

```

estimator = []
estimator.append(('catboost',cat))
estimator.append(('randomforest',randomforest))
estimator.append(('xgb',xgb))
estimator.append(('lgr',lgr))

```

```

voting = VotingClassifier(estimators= estimator,voting = 'soft')
voting.fit(X_train,y_train)

```

Learning rate set to 0.014474

0:	learn: 0.6634346	total: 161ms	remaining: 2m 40s
1:	learn: 0.6336754	total: 166ms	remaining: 1m 22s
2:	learn: 0.6081259	total: 172ms	remaining: 57.2s
3:	learn: 0.5903566	total: 177ms	remaining: 44.1s
4:	learn: 0.5651287	total: 182ms	remaining: 36.2s
5:	learn: 0.5451984	total: 186ms	remaining: 30.8s
6:	learn: 0.5227123	total: 190ms	remaining: 27s
7:	learn: 0.5046087	total: 195ms	remaining: 24.2s
8:	learn: 0.4843424	total: 200ms	remaining: 22s
9:	learn: 0.4657945	total: 204ms	remaining: 20.2s
10:	learn: 0.4455297	total: 209ms	remaining: 18.8s
11:	learn: 0.4269345	total: 214ms	remaining: 17.6s
12:	learn: 0.4115255	total: 218ms	remaining: 16.5s
13:	learn: 0.3968832	total: 222ms	remaining: 15.7s
14:	learn: 0.3837349	total: 227ms	remaining: 14.9s
15:	learn: 0.3704792	total: 231ms	remaining: 14.2s
16:	learn: 0.3590101	total: 236ms	remaining: 13.6s
17:	learn: 0.3456744	total: 241ms	remaining: 13.1s
18:	learn: 0.3345047	total: 246ms	remaining: 12.7s
19:	learn: 0.3225562	total: 250ms	remaining: 12.2s
20:	learn: 0.3101763	total: 255ms	remaining: 11.9s
21:	learn: 0.3004072	total: 260ms	remaining: 11.5s
22:	learn: 0.2916018	total: 264ms	remaining: 11.2s
23:	learn: 0.2807871	total: 269ms	remaining: 10.9s
24:	learn: 0.2731066	total: 274ms	remaining: 10.7s
25:	learn: 0.2630479	total: 278ms	remaining: 10.4s
26:	learn: 0.2534657	total: 284ms	remaining: 10.2s
27:	learn: 0.2447225	total: 288ms	remaining: 10s
28:	learn: 0.2369604	total: 293ms	remaining: 9.81s
29:	learn: 0.2293946	total: 298ms	remaining: 9.65s
30:	learn: 0.2230168	total: 303ms	remaining: 9.47s
31:	learn: 0.2167960	total: 308ms	remaining: 9.32s

Figure 6.17

## 9) Evaluating our trained model on test data

After training our ML model we tested our model on test data and evaluated its performance based on accuracy.

```
y_pred = voting.predict(X_test)
```

```
y_pred
```

```
array([1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0,  
       1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0,  
       0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1,  
       0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,  
       1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1,  
       0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0,  
       1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1,  
       1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1,  
       1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1,  
       0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1,  
       0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0,  
       0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0,  
       0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0,  
       1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0,  
       1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0,  
       0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0,  
       0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,  
       0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1,  
       0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1,  
       0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1,  
       0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1,  
       0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1,  
       0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1,  
       0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0,  
       0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0,  
       0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,  
       1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1,  
       0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1,  
       1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0,  
       0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0,  
       0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1,  
       0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1,  
       1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1,  
       0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1,  
       1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0,  
       1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0,  
       1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0,  
       0, 0, 0, 1, 0], dtype=int64)
```

Figure 6.18

The above array is the prediction of our trained model on the test data. This array will be compared with actual/true value and the accuracy will be calculated

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn import metrics
```

```
metrics.accuracy_score(y_test, y_pred)
```

```
0.9789695057833859
```

Figure 6.19

Here, using accuracy score we can see that we got accuracy of 97.8%. To check how many labels were mis predicted and correctly predicted by our ML model we will use confusion matrix.

```
t = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix= t, display_labels= voting.classes_)
disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x133cbc081c0>
```

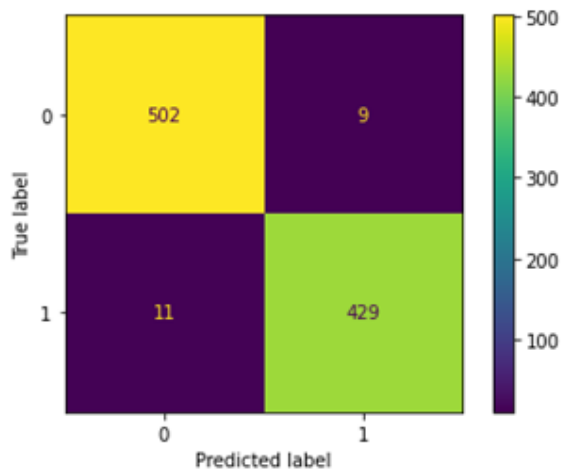


Figure 6.20

Here we can conclude the following points:

- Our model predicted Females which were Females were 502/511. (Correctly predicted Females)

- Our model predicted Males which were Females were 009/511. (Mis prediction of Females)
- Our model predicted Female which were Males were 011/440. (Mis prediction of Males)
- Our model predicted Males which were Males were 429/440. (Correctly predicted Males).

## 10) Exporting trained model for front end

We will now export our ML model to use it in our front end.

```
import pickle
```

```
file_name = 'finalized_model.sav'  
pickle.dump(voting, open(file_name, 'wb'))
```

```
loaded_model = pickle.load(open('finalized_model.sav', 'rb'))
```

*Figure 6.21*

This will save our trained model in the root folder for further use in any application.

## 11) Recording voice and recognizing gender in real-time

After training our model now we need to test it on real-world and see its performance on unseen data which is totally different from the dataset provided as the real world data can contain noise. As a matter of fact, in an object like voice, noise (background noise captured by mic) can have a significant impact on the result of our model.



```
import sounddevice as sd
from scipy.io.wavfile import write
from scipy.io import wavfile
import numpy as np
```

```
fs = 44100 # Sample rate
seconds = 3 # Duration of recording
```

```
myrecording = sd.rec(int(seconds * fs), samplerate=fs, channels=1)
print("Speak Now")
sd.wait() # Wait until recording is finished
print("Recording Done!")
write('output.wav', fs, myrecording) # Save as WAV file
```

```
Speak Now
Recording Done!
```

*Figure 6.22*

In this code snippet, a male voice was captured for 3 seconds from 1 channel. Where 1 channel is mono audio and 2 channel is stereo and so on. Our model was trained only on mono audio. Hence, only 1 channel will be used to record audio. The recorded voice will be stored in the root folder named "output.wav". Currently our program can only process wav file formats.

## **12) Extracting features from the recorded voice**

Now, that a voice is recorded for testing our trained model. We first need to extract the features of the recorded voice containing same features as used to train our model. To extract the features from the voice we need to implement Warble R library from the R language. The used R script will be discussed later in this report. To extract features, we will run the R script from Jupyter notebook using os (operating system) library.



```
import os
```

```
os.system('Rscript extractor_feature.r')
```

Warning message:  
In (0:(n - 1)) \* f : NAs produced by integer overflow  
0

```
voice = pd.read_csv('my_voice.csv')
```

```
#columns_scaling = ["meanfreq", "sd", "median", "Q25", "Q75", "IQR", "skew", "kurt", "sp.ent", "sfm", "mode", "centroid", "meanfun", "minfun", "maxfun", "meandom", "m
#voice[columns_scaling] = MinMaxScaler().fit_transform(voice[columns_scaling])
```

```
voice
```

	meanfreq	sd	median	Q25	Q75	IQR	skew	kurt	sp.ent	sfm	mode	centroid	meanfun	minfun	maxfun	meandom	mindom	maxi
0	0.150715	0.093451	0.135161	0.116138	0.255304	0.139166	9.159192	130.723983	0.841195	0.368963	0	0.150715	0.131587	0.043151	0.279114	0.069255	0	0.62

```
voice = voice.drop('centroid',axis=1)
```

```
voice = voice.drop('maxdom',axis=1)
```

```
type(voice)
```

pandas.core.frame.DataFrame

Figure 6.23

In the above code snippet we used `os` library and ran a terminal command that will run the “extractor.r” script and extract features of the wav file present in the root folder named as “output.wav” and save its features in the root folder with extension csv and filename as “my\_voice.csv”. Now we can use this csv file to provide a real-world input to our trained model and test it.

### 13) Testing our model on real-world input

As discussed, the audio file provided to the input was of the gender “male”. Let’s perform prediction on the features of the voice containing in the filename “my\_voice.csv”.

```
gender = voting.predict(voice)

if(gender[0] == 0):
    print("Female")
else:
    print("Male")

Male
```

*Figure 6.24*

The prediction done by our trained model will be in 0 or 1 as this is a binary classification problem. As discussed, while encoding our “label” feature, we saw that “female” label was changed to 0 and “male” label was changed to 1. Here, after predicting our model predicted the provided voice as “male” which is correct. We also provided “female” voice to our model which was predicted correctly by our model. Which will be demonstrated in the GUI section of this report.

### **6.1.3 R Script to Extract Features from Audio File**

To test our model on real-world input. That is, an audio file. We need to extract its features so that it can act as a real-world input for our machine learning model. To do so, we need a program in which the Warble R package of R language is implemented that will extract all the features of voice which are required by our machine learning model.

```

meanfreq <- analysis$mean/1000
sd <- analysis$sd/1000
median <- analysis$median/1000
Q25 <- analysis$Q25/1000
Q75 <- analysis$Q75/1000
IQR <- analysis$IQR/1000
skew <- analysis$skewness
kurt <- analysis$kurtosis
sp.ent <- analysis$sh
sfm <- analysis$sfm
mode <- analysis$mode/1000
centroid <- analysis$cent/1000

#Frequency with amplitude peaks
peakf <- 0#seewave::fpeaks(songspec, f = r@samp.rate, w1 = w1, nmax = 3, plot = FALSE)[1, 1]

#Fundamental frequency parameters
ff <- seewave::fund(r, f = r@samp.rate, ovlp = 50, threshold = threshold,
                    fmax = 280, ylim=c(0, 280/1000), plot = FALSE, w1 = w1)[, 2]
meanfun<-mean(ff, na.rm = T)
minfun<-min(ff, na.rm = T)
maxfun<-max(ff, na.rm = T)

#Dominant frequency parameters
y <- seewave::dfreq(r, f = r@samp.rate, w1 = w1, ylim=c(0, 280/1000), ovlp = 0, plot = F, threshold
meandom <- mean(y, na.rm = TRUE)
mindom <- min(y, na.rm = TRUE)
maxdom <- max(y, na.rm = TRUE)
dfrange <- (maxdom - mindom)
duration <- (end[i] - start[i])

#modulation index calculation
changes <- vector()
for(j in which(!is.na(y))){
  change <- abs(y[j] - y[j + 1])
  changes <- append(changes, change)
}
if(mindom==maxdom) modindx<-0 else modindx <- mean(changes, na.rm = T)/dfrange

#save results
return(c(duration, meanfreq, sd, median, Q25, Q75, IQR, skew, kurt, sp.ent, sfm, mode,
          centroid, peakf, meanfun, minfun, maxfun, meandom, mindom, maxdom, dfrange, modindx))

```

Figure 6.25

The above R script will extract all the 21 acoustic features of the voice which are required by our ML model for prediction.

After completing our back-end. Now we need to implement a user-friendly program so that any user can use it without any hassle. To achieve the same, we need a GUI (Graphical User Interface) that will contain all the features required to accomplish project task with only a few clicks. To implement a GUI we will use Tkinter package in python.

## 6.2 Front-End

To start with GUI / Front-end we made a python script as running jupyter notebook cells is not practical. With a python script only one run command will do the trick.

First we imported all the libraries required as shown –

```
1  import shutil
2  from tkinter import *
3  from tkinter.ttk import *
4  from tkinter.filedialog import askopenfile
5  import os
6  import pickle
7  from tracemalloc import is_tracing
8  import pandas as pd
9  import webbrowser
10 from sklearn.model_selection import train_test_split
11 from sklearn.preprocessing import LabelEncoder
12 from sklearn.preprocessing import MinMaxScaler
13 from catboost import CatBoostClassifier
14 from sklearn.ensemble import RandomForestClassifier
15 from sklearn.linear_model import LogisticRegression
16 from sklearn.ensemble import VotingClassifier
17 from xgboost import XGBClassifier
18 from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
19 from sklearn import metrics
```

*Figure 6.26*

After importing all the libraries we need a window to show all the elements like buttons, widgets, text. This will be achieved with the following code snippet

```

root = Tk()

root.title('Major Project')

#width x height
root.geometry("1280x720")

#width,height
root.minsize(1280,720)

root.maxsize(1920,1080)
scrollbar = Scrollbar(root)
scrollbar.pack(side = RIGHT, fill = Y )

```

*Figure 6.27*

Here root works as a window of our GUI. Where we have defined the minimum and maximum size of the window. Now, the title of the project and a logo is added to the GUI with the following code.

```

#GEU LOGO
photo1 = PhotoImage(file = "/home/sparsh/pyvscode/Major_Project/GenderPredict-main/Resources/download.png")
photo1_label = Label(image=photo1)
photo1_label.pack()

#MAJOR PROJECT TITLE
Heading = Label(text = "Gender Recognition Using Voice",font=('Helvetica', 22, 'bold'))
Heading.pack(pady=45)

```

*Figure 6.28*

After adding a logo and title, we will add a record button so that a user can record his/her voice. This will be achieved by the same method we implemented during our back end. So the code snippet for record button and its function will look like following-

```

#RECORDING VOICE
def record(event):
    import sounddevice as sd
    from scipy.io.wavfile import write
    from scipy.io import wavfile
    import numpy as np
    fs = 44100 # Sample rate
    seconds = 3 # Duration of recording
    myrecording = sd.rec(int(seconds * fs), samplerate=fs, channels=1)
    #print("Speak Now")
    for widgets in record_frame.winfo_children():
        widgets.destroy()
    Recording = Label(record_frame, text = "Recording...")
    Recording.pack()
    sd.wait() # Wait until recording is finished
    for widgets in record_frame.winfo_children():
        widgets.destroy()
    Recorded = Label(record_frame, text = "Recording Done!")
    Recorded.pack()
    write('/home/sparsh/pyvscod/Major_Project/GenderPredict-main/output1.wav', fs, myrecording) # Save as WAV file

Record_png = PhotoImage(file = "/home/sparsh/pyvscod/Major_Project/GenderPredict-main/Resources/record.png")
Record_button = Button(root, text = 'Click Me !', image = Record_png, cursor="hand2")
Record_button.pack()
Record_button.bind('<Button-1',record)

record_frame = Frame(root,borderwidth=6)
record_frame.pack()
record_text = Label(record_frame,text="Record\n OR")
record_text.pack()

```

*Figure 6.29*

So, after recording the voice, voice will be saved in the root folder with the filename “output.wav”. It is not always necessary that user will take recording every time he/she

wants to use the program. So, we implemented a feature to upload a file in wav format. The upload method is implemented as following-

```
#UPLOAD FILE
def open_file():
    global file
    file = askopenfile(filetypes=[('Waveform Audio File', '*wav')])
    if file is not None:
        filepath = os.path.abspath(file.name)
        location = Label(upload_frame, text='File Chosen: ' + str(filepath))
        location.pack()
        src = str(filepath)
        dst = "/home/sparsh/pyvscod/Major_Project/GenderPredict-main/output.wav"
        shutil.copyfile(src, dst)

upload_frame = Frame(root, borderwidth=2)
upload_frame.pack()
upload_button = Button(upload_frame, text = 'Choose File', command=lambda:open_file(), cursor="hand2")
upload_button.pack()
upload_label = Label(upload_frame, text = 'Upload audio in wav format')
upload_label.pack()
```

*Figure 6.30*

**Note – Frame** method create a frame in GUI which takes a argument of window. Here, window of our GUI is implemented as root.

**Note - Button** method creates a botton in the window canvas in desired location which can be a window (root) or a frame. Text parameter in Button method apply a text to Button. Command parameter tells the button what to do when pressed.

**Note - Label** apply text to a window (root) or a frame. Text parameter contains a string which will be displayed on the desired location.

In the above code when a user will press the upload button, a dialog box to choose a file will open and the desired file in wav format will be copied in the root folder.



After recording or uploading a audio file in wav format, user will need to predict the gender in that audio file. So, the GUI contains a button to predict the gender. The implementation of gender prediction will done by loading the save ML model. The code snippet for the same will look like –

```
#PREDICTION
def predict(event):
    for widgets in predict_frame.winfo_children():
        widgets.destroy()

    model = pickle.load(open('/home/sparsh/pyvscod/Major_Project/GenderPredict-main/finalized_model.sav', 'rb'))
    os.chdir('/home/sparsh/pyvscod/Major_Project/GenderPredict-main/')
    os.system('Rscript extractor_feature.r')
    voice = pd.read_csv('my_voice.csv')
    voice = voice.drop('centroid',axis=1)
    voice = voice.drop('maxdom',axis=1)
    gender = model.predict(voice)
    if(gender[0] == 0):
        prediction = Label(predict_frame,text="Female",font=('comicsans', 22, 'bold'))
        prediction.pack()
    else:
        prediction = Label(predict_frame,text = "Male",font=('comicsans', 22, 'bold'))
        prediction.pack()

predict_frame = Frame(root, borderwidth=6)
predict_frame.pack()
predict_button = Button(predict_frame,text = 'Predict Gender')
predict_button.pack()
predict_button.bind('<Button-1>',predict)
```

*Figure 6.31*

In the above code, we first cleared the previous prediction (if any). Then, extracted the features of the voice recorded or uploaded by the user. Then, we loaded our ML model and used it to predict the gender in the voice and display it in the GUI.

Now, if the prediction done by the user is correct (if the user knows for sure). He/she can save it in the dataset to increase its accuracy and the ability to predict more accurately in real-world. Now, the challenge that arise is, that the user can click save button multiple times on the same voice which will increase redundancy, size of dataset, and time to train ML model on the new dataset. Also, removing the save button after clicking once will make user to run the program again which will be very inconvenient. Hence, we have provided a method which will remove the redundant rows once the user presses the save button. Hence the implementation of the same will look as follows –

```

#IF PREDICTION IS CORRECT WE WILL SAVE THE VOICE FEATURES AND SAVE TO DATAFRAME
def save(event):
    for widgets in saved_frame.winfo_children():
        widgets.destroy()
    import csv
    from csv import writer
    model = pickle.load(open('/home/sparsh/pyvscode/Major_Project/GenderPredict-main/finalized_model.sav', 'rb'))
    voice = pd.read_csv('/home/sparsh/pyvscode/Major_Project/GenderPredict-main/my_voice.csv')
    voice = voice.drop('centroid',axis=1)
    voice = voice.drop('maxdom',axis=1)
    gender = model.predict(voice)
    with open('my_voice.csv','r') as read_obj:
        csv_reader = csv.reader(read_obj)
        list_of_csv = list(csv_reader)
    if(gender[0] == 0):
        predicted_gender = "female"
        list_of_csv[1].append("{}".format(predicted_gender))
    else:
        predicted_gender = "male"
        list_of_csv[1].append("{}".format(predicted_gender))

    with open ('voice.csv','a') as f_object:
        writer_object = writer(f_object)
        writer_object.writerow(list_of_csv[1])
        f_object.close()

    df=pd.read_csv("/home/sparsh/pyvscode/Major_Project/GenderPredict-main/voice.csv")
    df.drop_duplicates(subset=["meanfreq","sd","median","Q25","Q75","IQR","skew","kurt","sp.ent","sfm","mode","cent
    df.to_csv("/home/sparsh/pyvscode/Major_Project/GenderPredict-main/voice.csv",index=False)
    saved = Label(saved_frame,text = "Saved Thank You!",font=('comicsans', 22, 'bold'))
    saved.pack()

correct_frame = Frame(root,borderwidth=2)
correct_frame.pack()
correct_label = Label(correct_frame,text="*Please click the save button if the prediction is correct*")
correct_label.pack()
correct_button = Button(correct_frame,text = "Save")
correct_button.pack()
correct_button.bind('<Button-1',save)
saved_frame = Frame(root)
saved_frame.pack()

```

Figure 6.32

In the above code, when a prediction is validated by the user and he/she press the save button, The features of the voice recorded or uploaded will be saved in the dataset with the gender label included. And, if the user tries to save redundant data it will be

discarded. Now, using the new data, user can train the model and increase its accuracy further as the dataset grows. To achieve the same we have included a train model button which will train the ML model on the new dataset and will display its new accuracy on the GUI. The code snippet for the same is shown as follows-

```
#TRAIN NEW MODEL
def train_model(event):
    for widgets in accurate_frame.winfo_children():
        widgets.destroy()
    df = pd.read_csv('/home/sparsh/pyvscode/Major_Project/GenderPredict-main/voice.csv')
    encoding_columns = [ "label"]
    Encoder = LabelEncoder()
    for column in encoding_columns :
        df[ column ] = Encoder.fit_transform(tuple(df[ column ]))
    df = df.drop('centroid',axis=1)
    df = df.drop('maxdom',axis=1)
    x = df.drop("label",axis=1)
    y = df["label"]
    X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=2)
    cat = CatBoostClassifier()
    randomforest = RandomForestClassifier()
    xgb = XGBClassifier()
    lgr = LogisticRegression(max_iter=7200)
    estimator = []
    estimator.append(('catboost',cat))
    estimator.append(('randomforest',randomforest))
    estimator.append(('xgb',xgb))
    estimator.append(('lgr',lgr))
    voting = VotingClassifier(estimators= estimator,voting ='soft')
    voting.fit(X_train,y_train)
    y_pred = voting.predict(X_test)
    accu = metrics.accuracy_score(y_test,y_pred)
    accu_label = Label(accurate_frame,text="accuracy = "+str(accu*100))
    accu_label.pack()
    file_name = 'finalized_model.sav'
    pickle.dump(voting, open(file_name,'wb'))
train_frame = Frame(root)
train_frame.pack()
train_label = Label(train_frame,text = "Click the button below to train on the new dataset")
train_label.pack()
train_button = Button(train_frame,text = "Train")
train_button.pack()
train_button.bind('<Button-1>',train_model)
accurate_frame = Frame(root)
accurate_frame.pack()
```

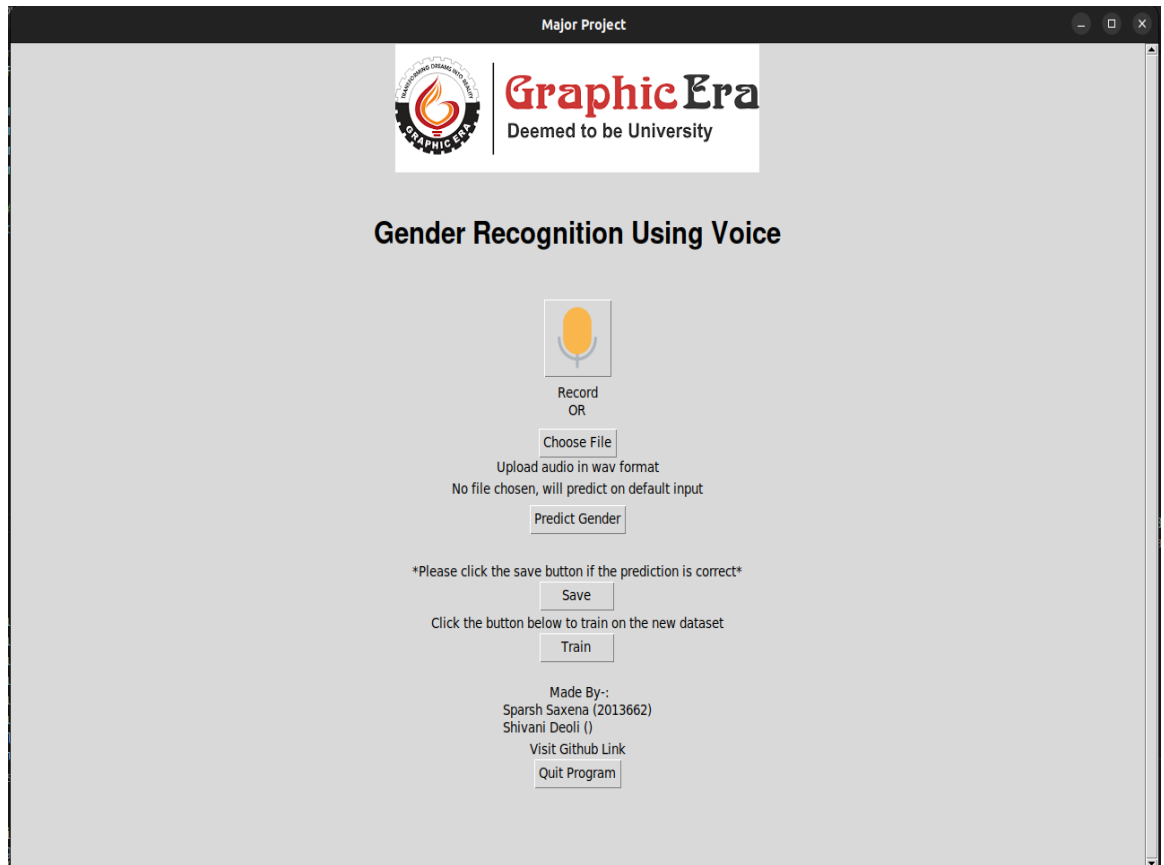
Figure 6.33

In the above code, when a user will press train button the ML model will trained in the same way as discussed in back-end phase and the new trained model will be saved in the root folder for further uses.

Hence, we implemented a GUI with some intermediate features that can increase model accuracy, also anyone can alter the code to do the same. The implemented GUI is flexible and more features can be added to the same. The GUI is easy to use, and can accomplish task with few clicks without any prior knowledge.

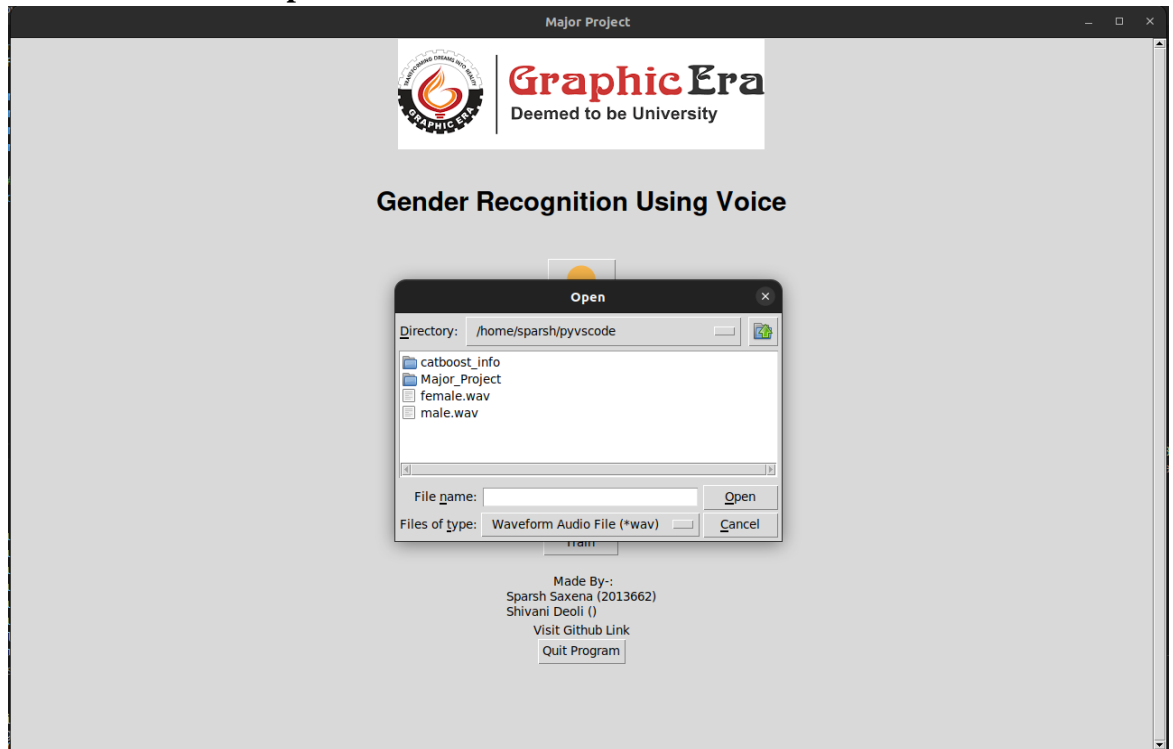
## 7. TESTING

**7.1. A case where user does not upload any audio. Then a dummy data will be created and user will be notified.**



*Figure 7.1*

**7.2. A case where user tries to upload a file other than wav format. Our GUI constraints user to upload file other than wav.**



*Figure 7.2*

**7.3 A case where user tries to save multiple times on the same data.** To counter the action a robust mechanism will prevent the user saving same data by deleting the redundant data without letting the user know, but user will see a prompt that the data is saved.

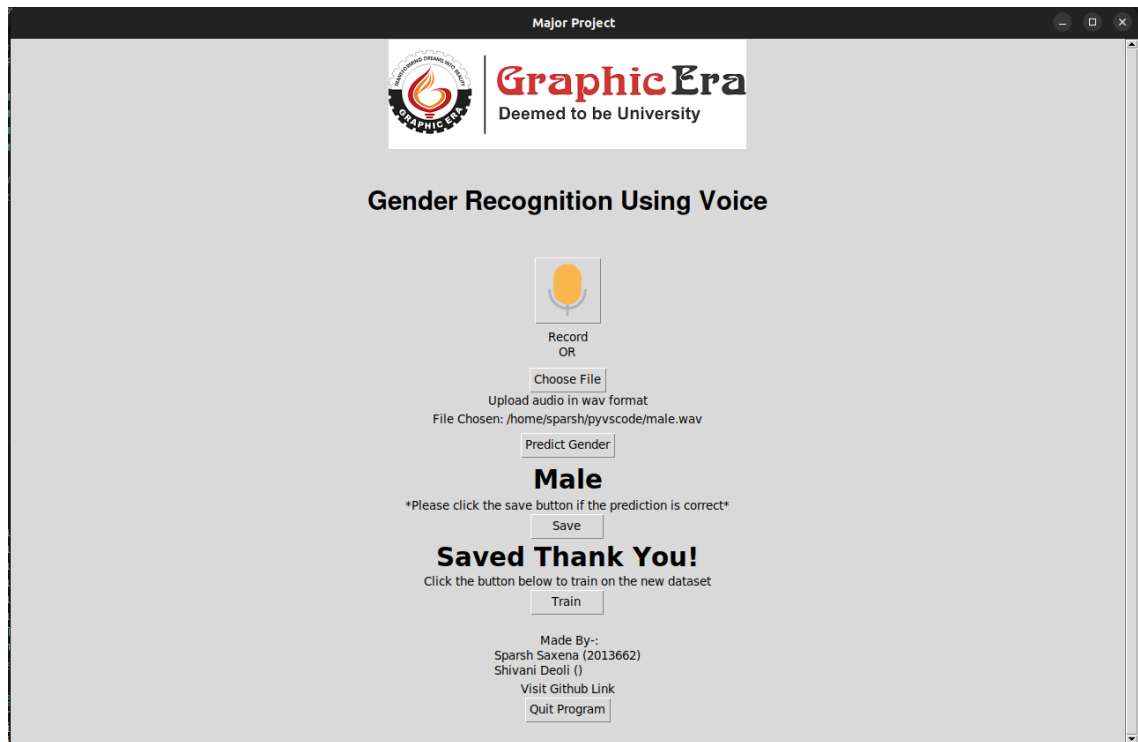


Figure 7.3

7.4 A case where user uploads voice of male and our ML model predicts it correctly.

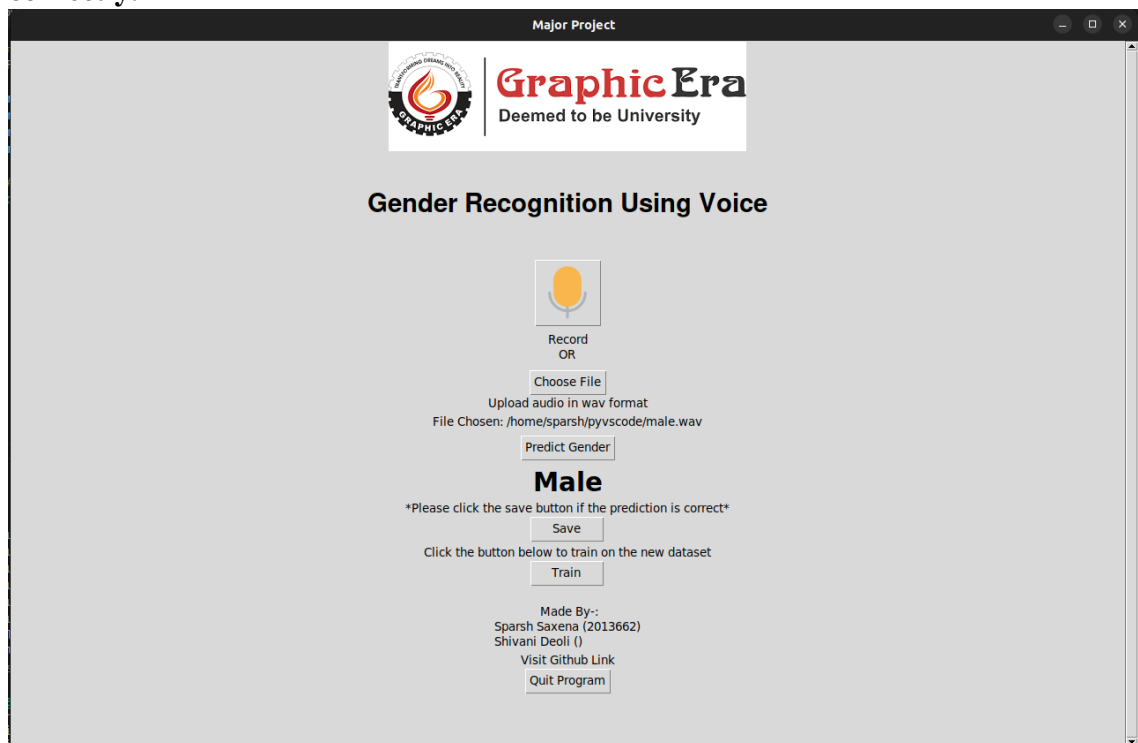
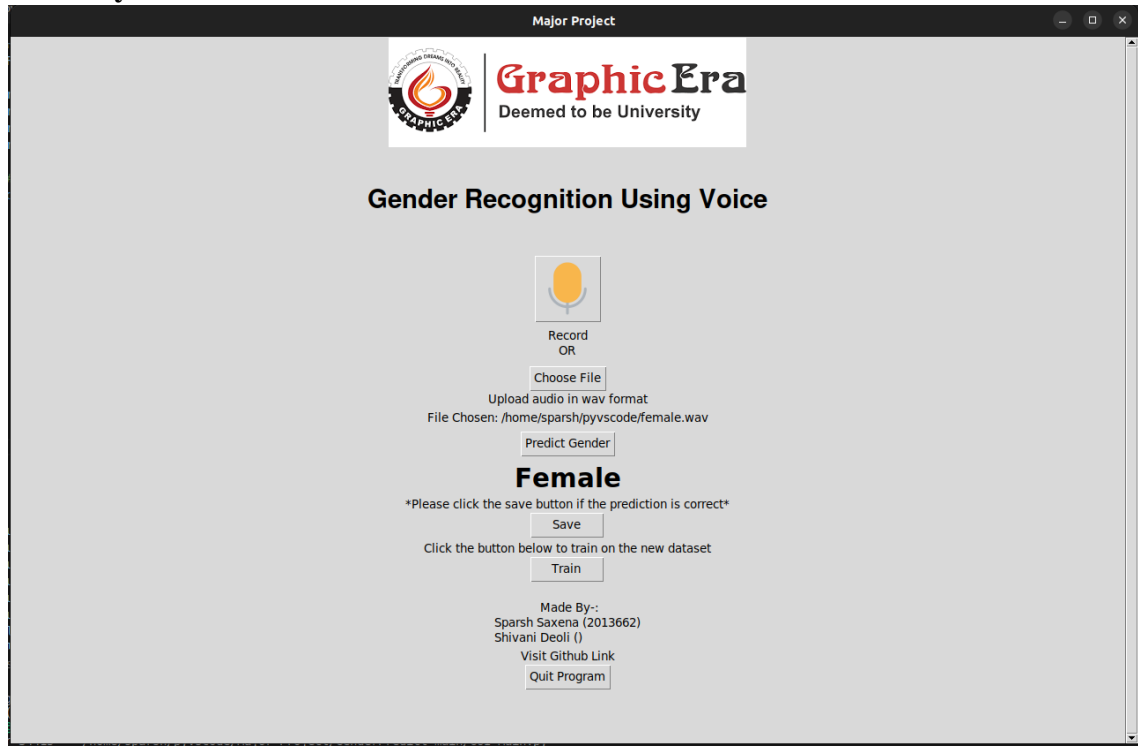


Figure 7.4



**7.5 A case where user uploads voice of female and our ML model predicts it correctly.**



*Figure 7.5*

## **8. OUTPUT SCREENS**

Output screens of GUI are shown below.

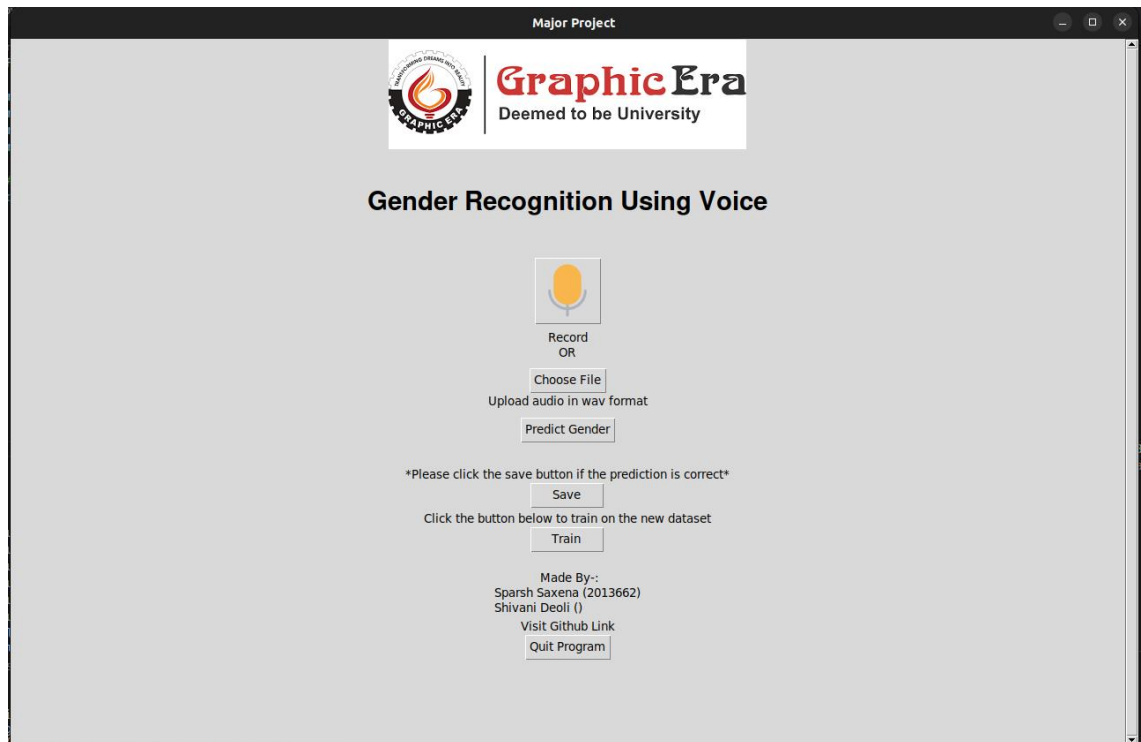


Figure 8.1

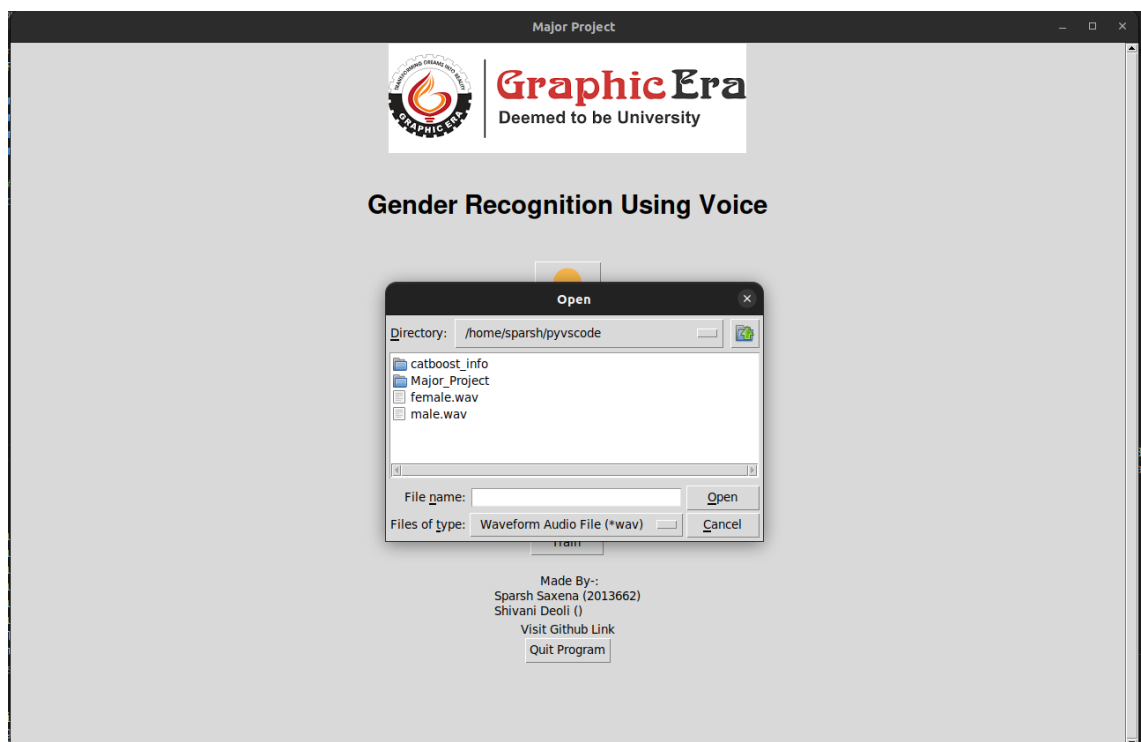


Figure 8.2

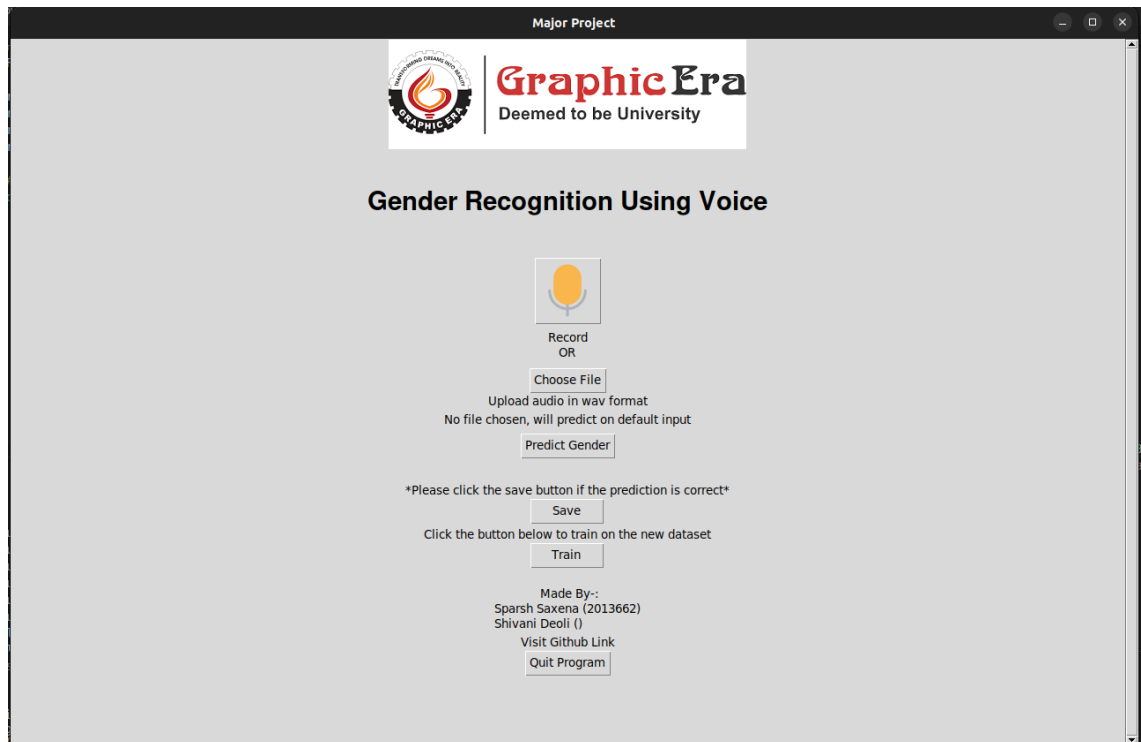


Figure 8.3

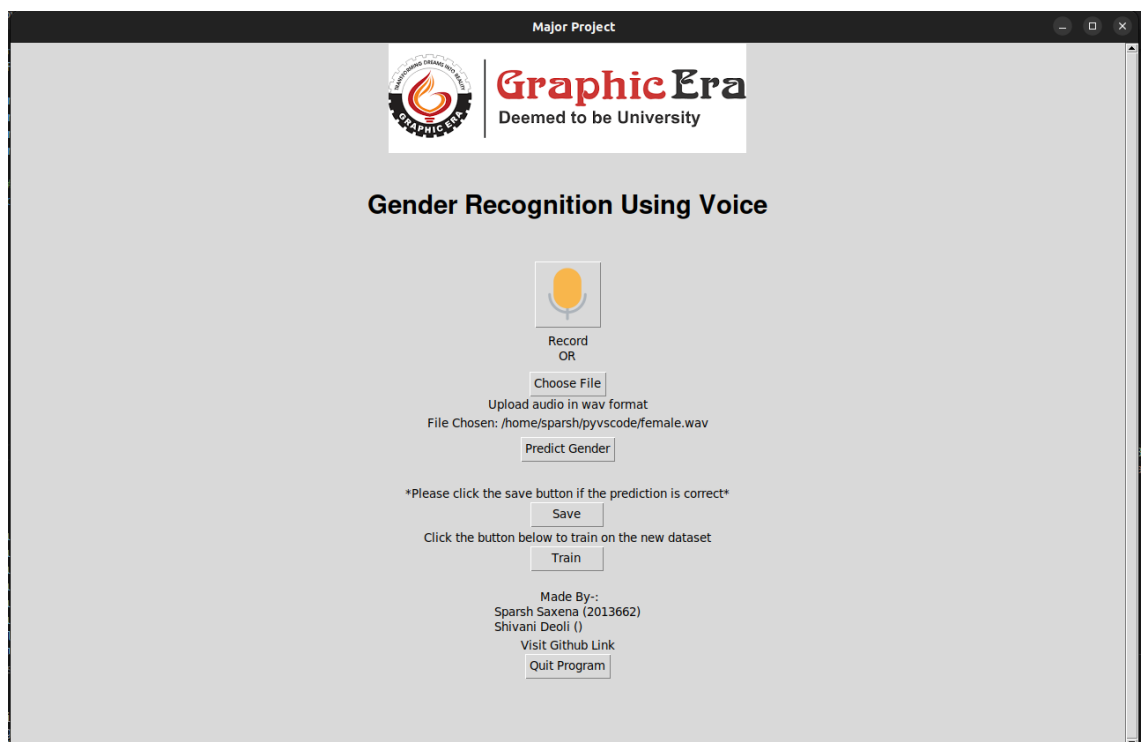


Figure 8.4

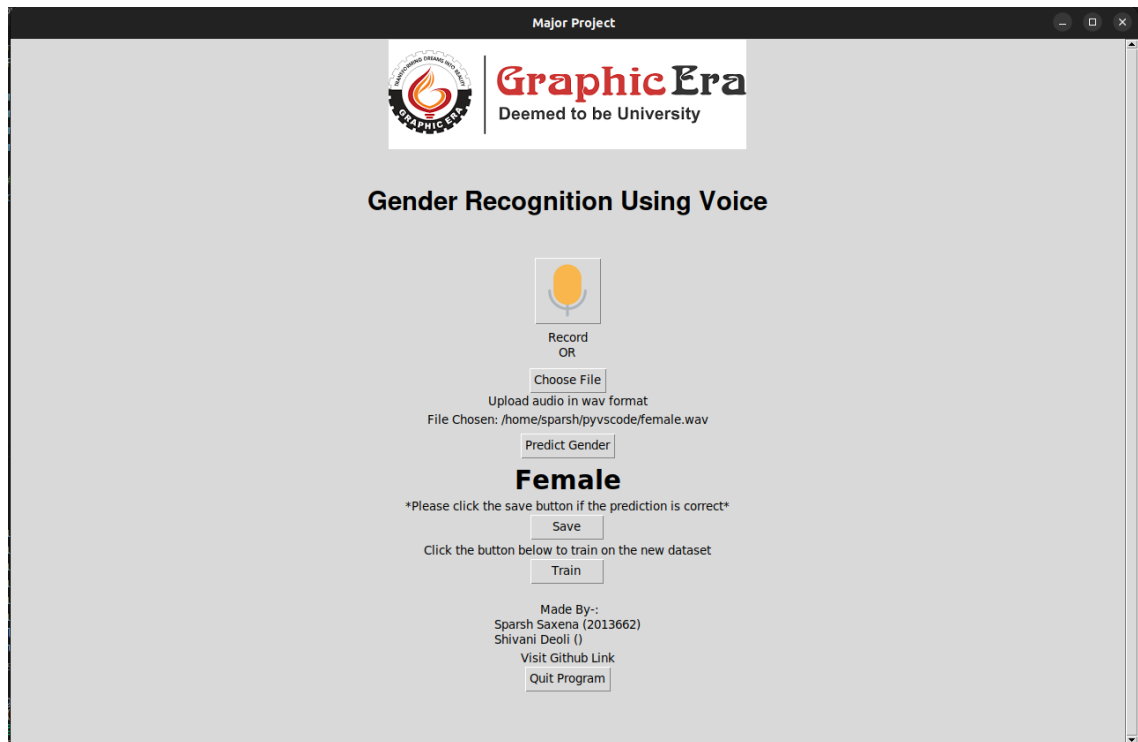


Figure 8.5

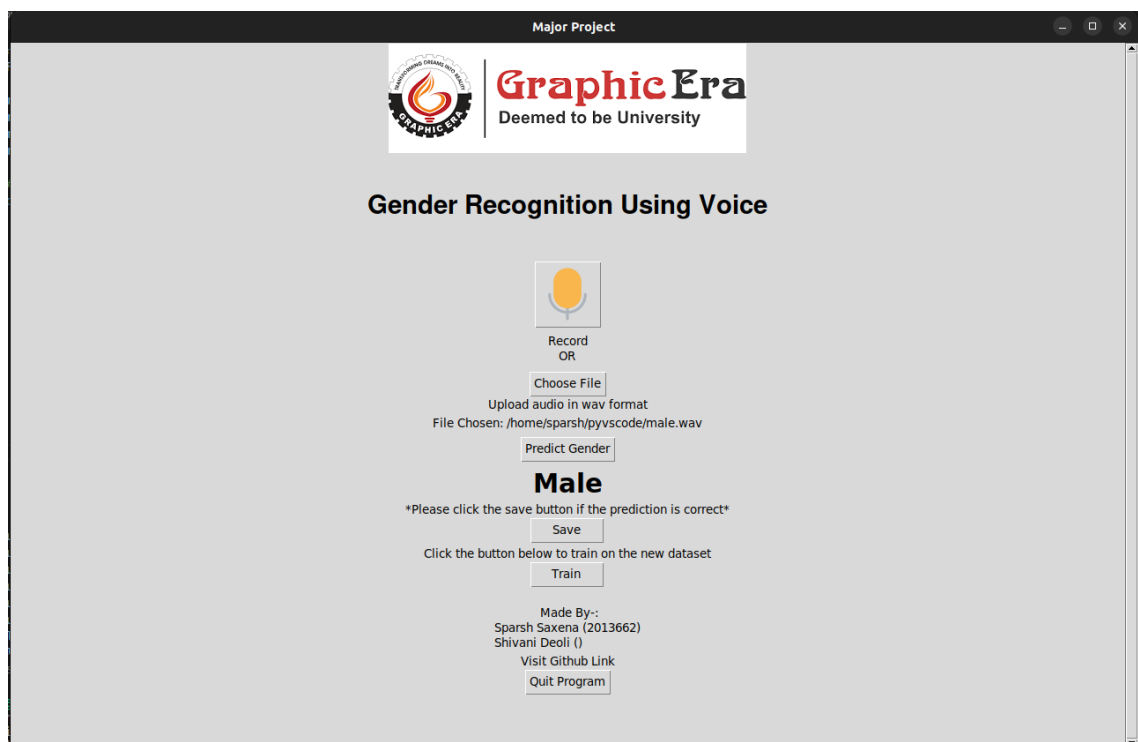


Figure 8.6

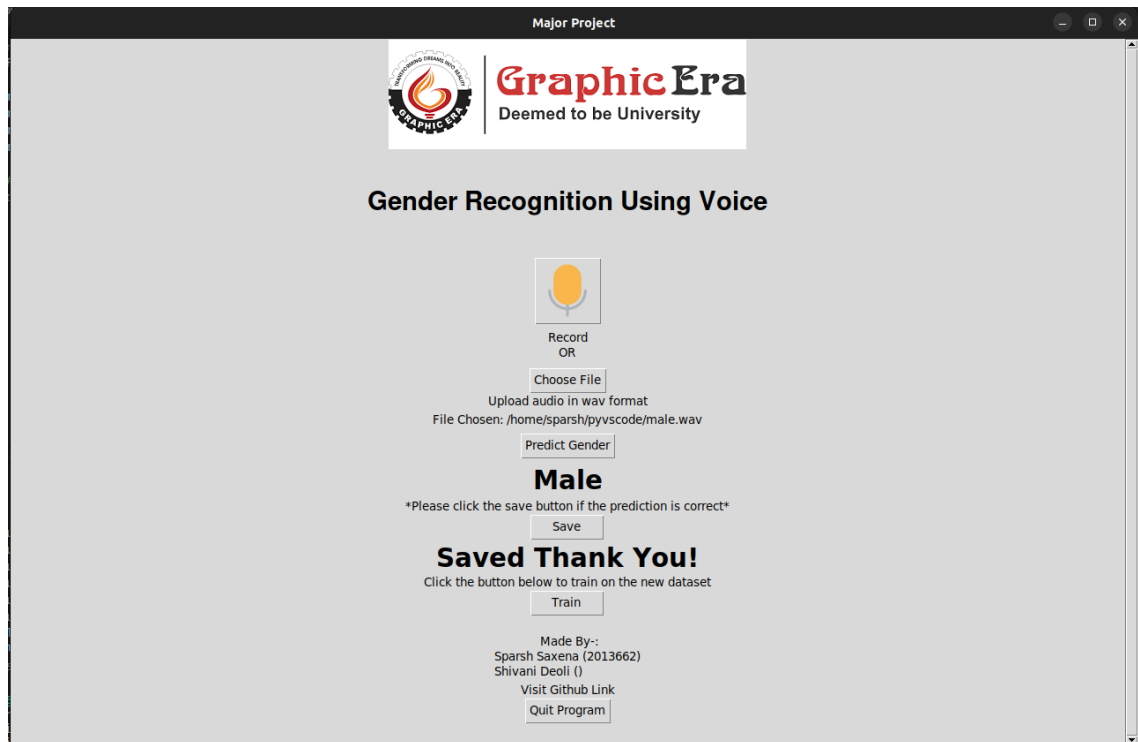


Figure 8.7

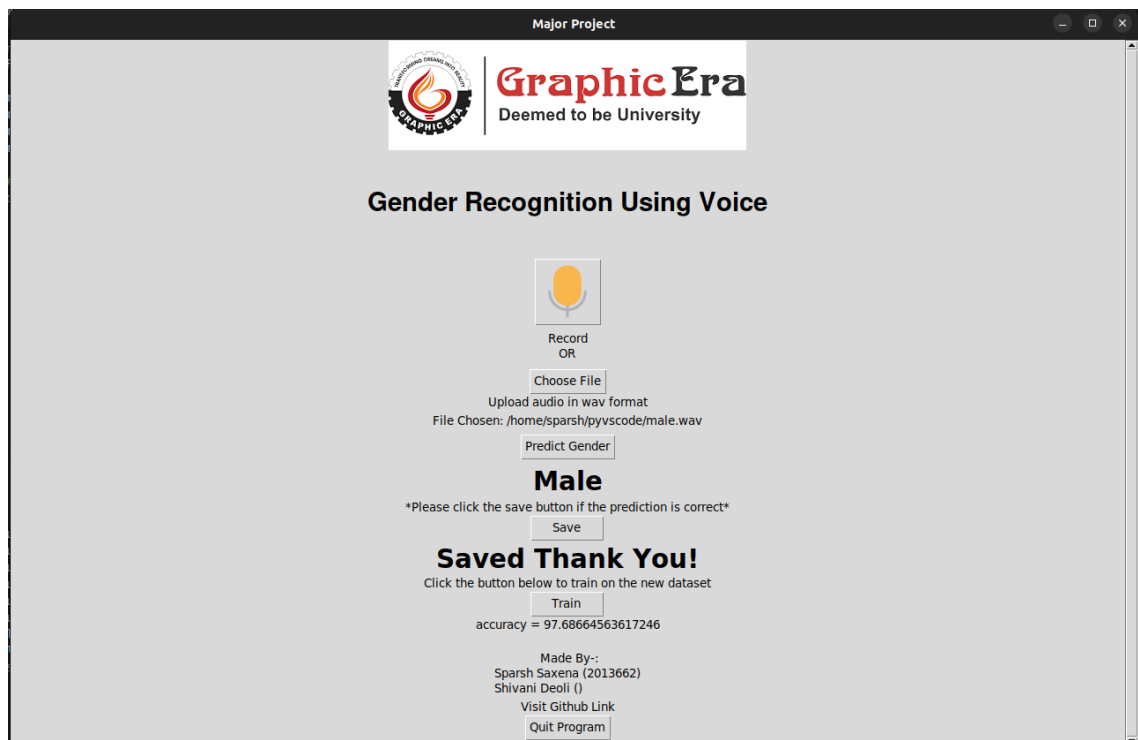


Figure 8.8

## 9. CONCLUSIONS

Recognizing the gender of human voice has been considered one of the challenging tasks because of its importance in various applications. The contributions are threefold including-

- (a)- Studying the extracted features by examining the correlation between each other
- (b)- Building classification models using different ML techniques from distinct families
- (c)- Evaluating the natural feature selection techniques in finding the optimal subset of relevant features on classification performance.

In phase-I we created and deployed the back end of the application/project and achieved brilliant accuracy of 97.8% and we created front end in the Phase-II of the project.

The machine learning model has been built and the final step includes integration of both application UI and machine learning model which is covered in Phase-II of the major project.

## 10. Further Enhancements

In the future work, more experiments will be conducted to use many feature categories, ML techniques, and other natural feature selection techniques. Further improvements will be made to the model to make it fast and efficient.

Our future work is focused on working with more regional datasets and improving the prediction accuracy of our work. Moreover, we want to implement our methodology in a real-time system.

If a person mimics voice of different gender, then he actually is, that time we could not get the desired result. To avoid it we can include face recognition model with the current model so that we can have visual identification of that person also.

## 11. References

- (a) Becker, K., 2022. Identifying the Gender of a Voice using Machine Learning. [online] Primary Objects. Available at: <<http://www.primaryobjects.com/2016/06/22/identifying-the-gender-of-a-voice-using-machine-learning/>> [Accessed 10 May 2022].
- (b) Kaggle.com. 2022. Gender Recognition by Voice. [online] Available at: <<https://www.kaggle.com/datasets/primaryobjects/voicegender>> [Accessed 10 May 2022].
- (c) Cran.r-project.org. 2022. *Streamline Bioacoustic Analysis [R package warbleR version 1.1.27]*. [online] Available at: <<https://cran.r-project.org/web/packages/warbleR/index.html>> [Accessed 11 May 2022].
- (d) Python, R., 2022. *Playing and Recording Sound in Python – Real Python*. [online] Realpython.com. Available at: <<https://realpython.com/playing-and-recording-sound-python/>> [Accessed 11 May 2022].
- (e) Atlantis-press.com. 2022. [online] Available at: <<https://www.atlantis-press.com/article/25868884.pdf>> [Accessed 11 May 2022].





