GENDER RECOGNIZATION USING VOICE

(USING ACOUSTIC PARAMETERIC VALUES)

GROUP MEMBERS

Name : B. Madhava Reddy
College : Lovely Professional University
University reg.no : 12101758

Name : G. Mallikarjuna Reddy
College : Lovely Professional University
University reg.no : 12102343

CONTENT TABLE

Acknowledgement

I take this opportunity to express my profound gratitude and deep regards to my faculty, **Prof. Arnab Chakraborty** for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by him time to time shall carry me a long way in the journey of life on which I am about to embark.

I am obliged to my project team members for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my assignment.

B.Madhava Reddy

G.Mallikarjuna Reddy

Project Objective

Project Objective: Gender Recognition using Voice (Acoustic Parameter Values)

1.Problem Description: The problem addressed in this project is gender recognition using voice recordings. The goal is to develop a machine learning model that can accurately classify the gender of individuals based on acoustic parameter values extracted from their voice samples.

2.Objective: The objective of this project is to build an effective gender recognition model using machine learning techniques and acoustic parameter values derived from voice recordings. By analyzing the acoustic features present in voice samples, the model aims to accurately predict the gender of a speaker.

The specific objectives of the project are as follows:

•Collect a diverse dataset of voice recordings from individuals of different genders.
•Extract relevant acoustic parameter values from the voice recordings, such as pitch, formants, vocal tract length, and intensity.

•Preprocess the dataset to handle missing values, normalize the data, and address outliers or noise in the voice recordings.

•Train a machine learning model using the preprocessed dataset to predict the gender based on the extracted acoustic features.
•Evaluate the performance of the model using appropriate metrics such as accuracy, precision, recall, and F1 score.
•Deploy the trained model into a usable application or system that can predict the gender of speakers based on voice inputs.

By achieving these objectives, the project aims to provide a reliable and accurate gender recognition system that can be applied to various domains, including speaker identification, speech recognition, and human-computer interaction.

The project objective encompasses understanding the problem of gender recognition using voice, developing a machine learning model that utilizes acoustic parameter values, and ultimately creating a practical application or system for gender recognition based on voice inputs.

PROJECT SCOPE

The scope of the project includes the following boundary conditions:

1.Data Collection:

1. Collect a diverse dataset of voice recordings from individuals of different genders.
2. Ensure a balanced representation of male and female speakers in the dataset.
3. Obtain voice samples that cover a wide range of vocal characteristics.

2.Acoustic Parameter Extraction:

1. Extract relevant acoustic parameter values from the voice recordings.
2. Consider features such as pitch, formants, vocal tract length, and intensity.
3. Utilize libraries and tools for accurate and efficient feature extraction.

3.Data Preprocessing:

1. Handle missing values in the dataset, if any, using appropriate techniques.
2. Normalize the data to ensure consistent scaling and comparability of acoustic features.
3. Address outliers or noise in the voice recordings through filtering or data cleaning methods.

4.Model Selection and Training:

1. Explore and evaluate different machine learning algorithms for gender recognition.
2. Consider algorithms such as logistic regression, support vector machines, random forests, or neural networks.
3. Optimize model hyperparameters and evaluate performance using appropriate validation techniques.

5.Model Evaluation and Performance Metrics:

1. Assess the performance of the trained model using evaluation metrics like accuracy, precision, recall, and F1 score.
2. Employ cross-validation techniques to ensure reliable performance estimation.
3. Utilize ROC curve analysis to evaluate the trade-off between true positive rate and false positive rate.

6.Model Deployment:

1. Implement the trained model into a practical application or system.
2. Develop a user-friendly interface to accept voice inputs and provide real-time gender predictions.
3. Ensure the application is efficient, reliable, and scalable for diverse usage scenarios.

Data Description

Source of the data: Kaggle. The given dataset is a original dataset in Kaggle.

Data Description: The given train dataset has 3168 rows and 21 columns.

Information about the dataset

```
<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
```

The Dataset

The following acoustic properties of each voice are measured and included within the CSV:

- **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
- **sfm**: 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
- **label**: male or female

Number statistics of the given dataset

	count	mean	std	min	25%	50%	75%	max
meanfreq	3168.0	0.180907	0.029918	0.039363	0.163662	0.184838	0.199146	0.251124
sd	3168.0	0.057126	0.016652	0.018363	0.041954	0.059155	0.067020	0.115273
median	3168.0	0.185621	0.036360	0.010975	0.169593	0.190032	0.210618	0.261224
Q25	3168.0	0.140456	0.048680	0.000229	0.111087	0.140286	0.175939	0.247347
Q75	3168.0	0.224765	0.023639	0.042946	0.208747	0.225684	0.243660	0.273469
IQR	3168.0	0.084309	0.042783	0.014558	0.042560	0.094280	0.114175	0.252225
skew	3168.0	3.140168	4.240529	0.141735	1.649569	2.197101	2.931694	34.725453
kurt	3168.0	36.568461	134.928661	2.068455	5.669547	8.318463	13.648905	1309.612887
sp.ent	3168.0	0.895127	0.044980	0.738651	0.861811	0.901767	0.928713	0.981997
sfm	3168.0	0.408216	0.177521	0.036876	0.258041	0.396335	0.533676	0.842936
mode	3168.0	0.165282	0.077203	0.000000	0.118016	0.186599	0.221104	0.280000
centroid	3168.0	0.180907	0.029918	0.039363	0.163662	0.184838	0.199146	0.251124
meanfun	3168.0	0.142807	0.032304	0.055565	0.116998	0.140519	0.169581	0.237636
minfun	3168.0	0.036802	0.019220	0.009775	0.018223	0.046110	0.047904	0.204082
maxfun	3168.0	0.258842	0.030077	0.103093	0.253968	0.271186	0.277457	0.279114
meandom	3168.0	0.829211	0.525205	0.007812	0.419828	0.765795	1.177166	2.957682
mindom	3168.0	0.052647	0.063299	0.004883	0.007812	0.023438	0.070312	0.458984
maxdom	3168.0	5.047277	3.521157	0.007812	2.070312	4.992188	7.007812	21.867188
dfrange	3168.0	4.994630	3.520039	0.000000	2.044922	4.945312	6.992188	21.843750
modindx	3168.0	0.173752	0.119454	0.000000	0.099766	0.139357	0.209183	0.932374

DataType of each column that present in the datset .

```
meanfreq    float64
sd          float64
median      float64
Q25         float64
Q75         float64
IQR         float64
skew        float64
kurt        float64
sp.ent      float64
sfm         float64
mode        float64
centroid    float64
meanfun     float64
minfun      float64
maxfun      float64
meandom     float64
mindom      float64
maxdom      float64
dfrange     float64
modindx     float64
label        object
dtype: object
```

DATA-PREPROCESSING

Now we will pre-process the data. The methodology followed is given below:

- Checking for null values.
  - If null values are present, we will fill them or drop the row containing the null value based on the dataset.
- Checking for outliers.
  - If outliers are present, they will either be removed or replaced by following a suitable method depending on the dataset.

Data Pre-processing

As the given dataset had Categorical and Non-categorical data mixed, we converted the categorical data into non-categorical data accordingly. We converted the binary categories into 0 and 1. We converted the other categorical attributes into suitable numerical values. The following table shows the conversion record:

Non-Numeric to Numeric Change Table

column	Initial-Value	Final-Value
label	male	1
	female	0

Table 3: Categorical to Numerical change in values

```
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 object
```

← BEFORE LABEL ENCODING

AFTER LABEL ENCODING →

```
0       1
1       1
2       1
3       1
4       1
       ..
3163    0
3164    0
3165    0
3166    0
3167    0
Name: label, Length: 3168, dtype: int64 int64
```

We searched for null values in our dataset and formed the following table:

	Total_count	Null_count
meanfreq	3168	0
sd	3168	0
median	3168	0
Q25	3168	0
Q75	3168	0
IQR	3168	0
skew	3168	0
kurt	3168	0
sp.ent	3168	0
sfm	3168	0
mode	3168	0
centroid	3168	0
meanfun	3168	0
minfun	3168	0
maxfun	3168	0
meandom	3168	0
mindom	3168	0
maxdom	3168	0
dfrange	3168	0
modindx	3168	0
label	3168	0

To visualise the null values, we made a heatmap plot using seaborn library function heatmap. The heatmap plot is given below:

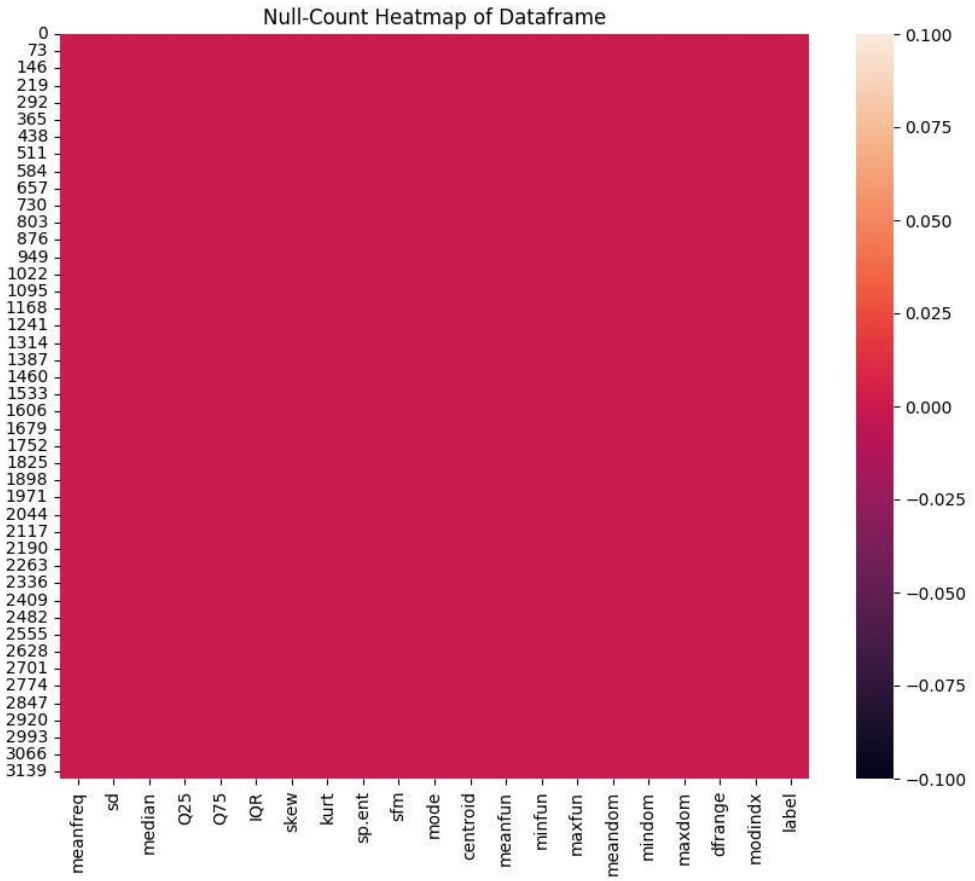
Fig 1: Heatmap of the given Data

The heatmap shows that the dataset has null values To remove the null values, we had the following methodology:

- filling Null values in 'Credit_History' with 0 (as stated in the dataset description from source)
- filling Null values in binary columns with mode value. This was applied in columns 'Self_Employed' and 'Gender'.
- filling Null values with mean +- std in non-binary columns. This was applied in column 'LoanAmount' and "Loan_Amount_Term'.

After removing the null values, the following heatmap was obtained:

Heatmap implying absence of null values.

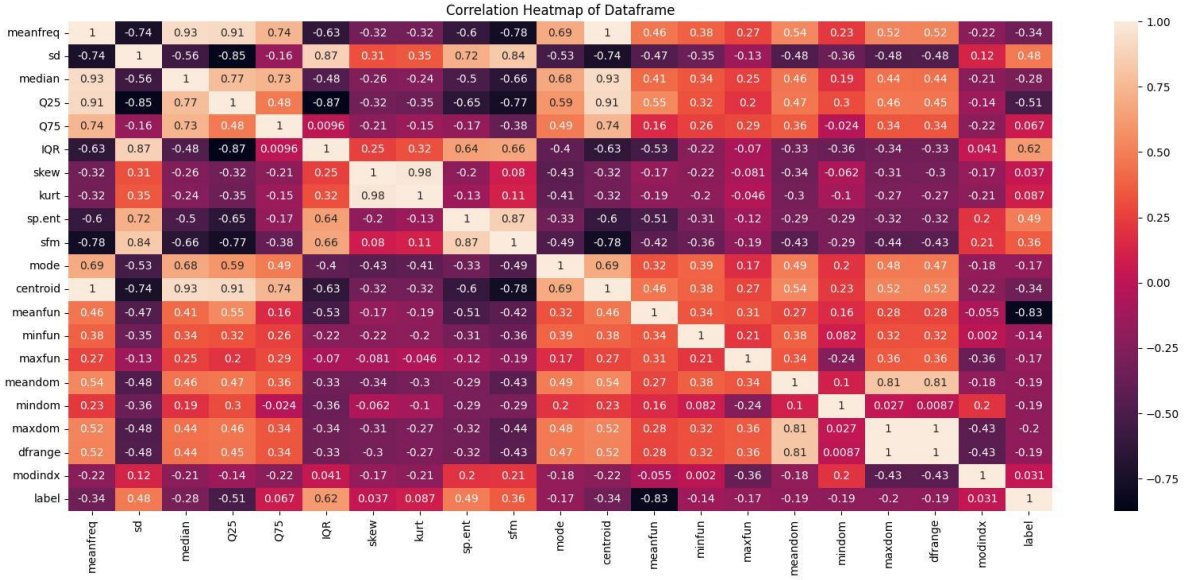


Now we have successfully handled Null values and converted non-numeric values to Numeric values.

So, we are moving on to find if there are any outliers in our data and find the correlations of different attributes to our target .

The following table gives the correlation value of each attribute with our target attribute i.e. ‘label’:

Correlation values with Target Attribute



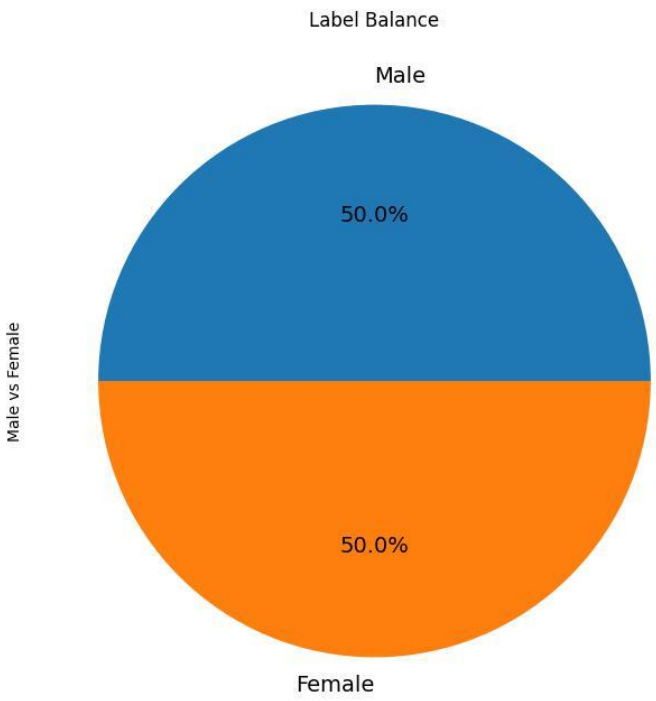
Outliers are extreme values that deviate from other observations on data, they may indicate a variability in a measurement, experimental errors or a novelty. In other words, an outlier is an observation that diverges from an overall pattern on a sample.

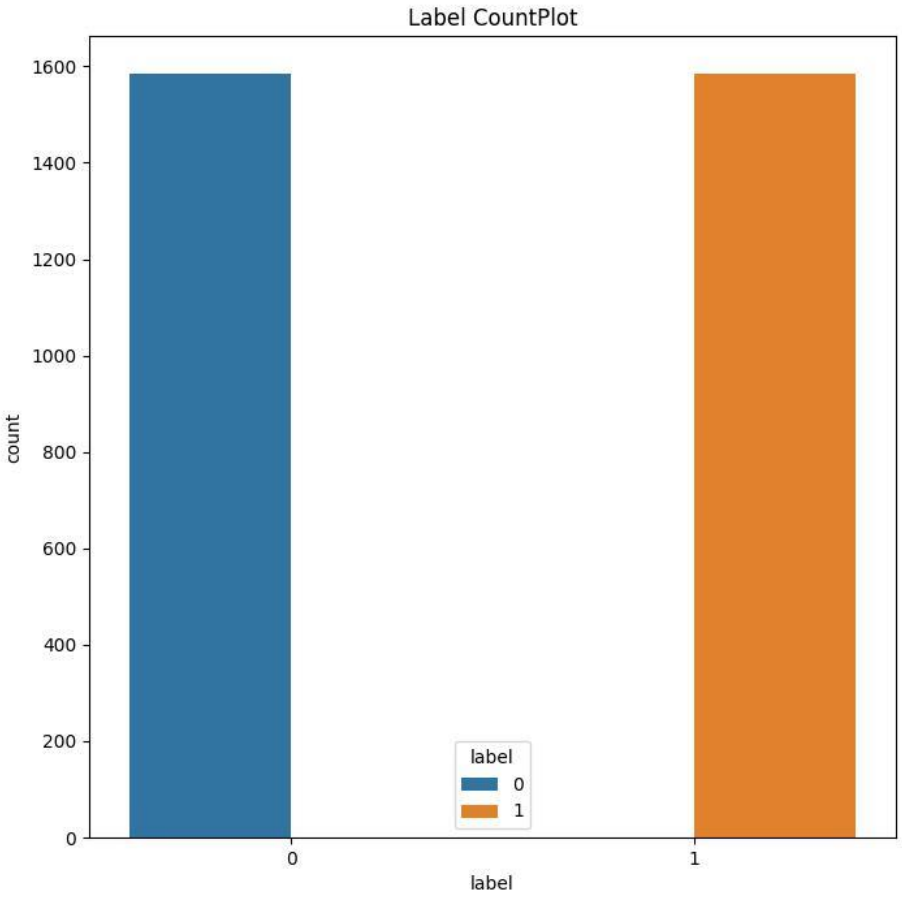
Most common causes of outliers on a data set:

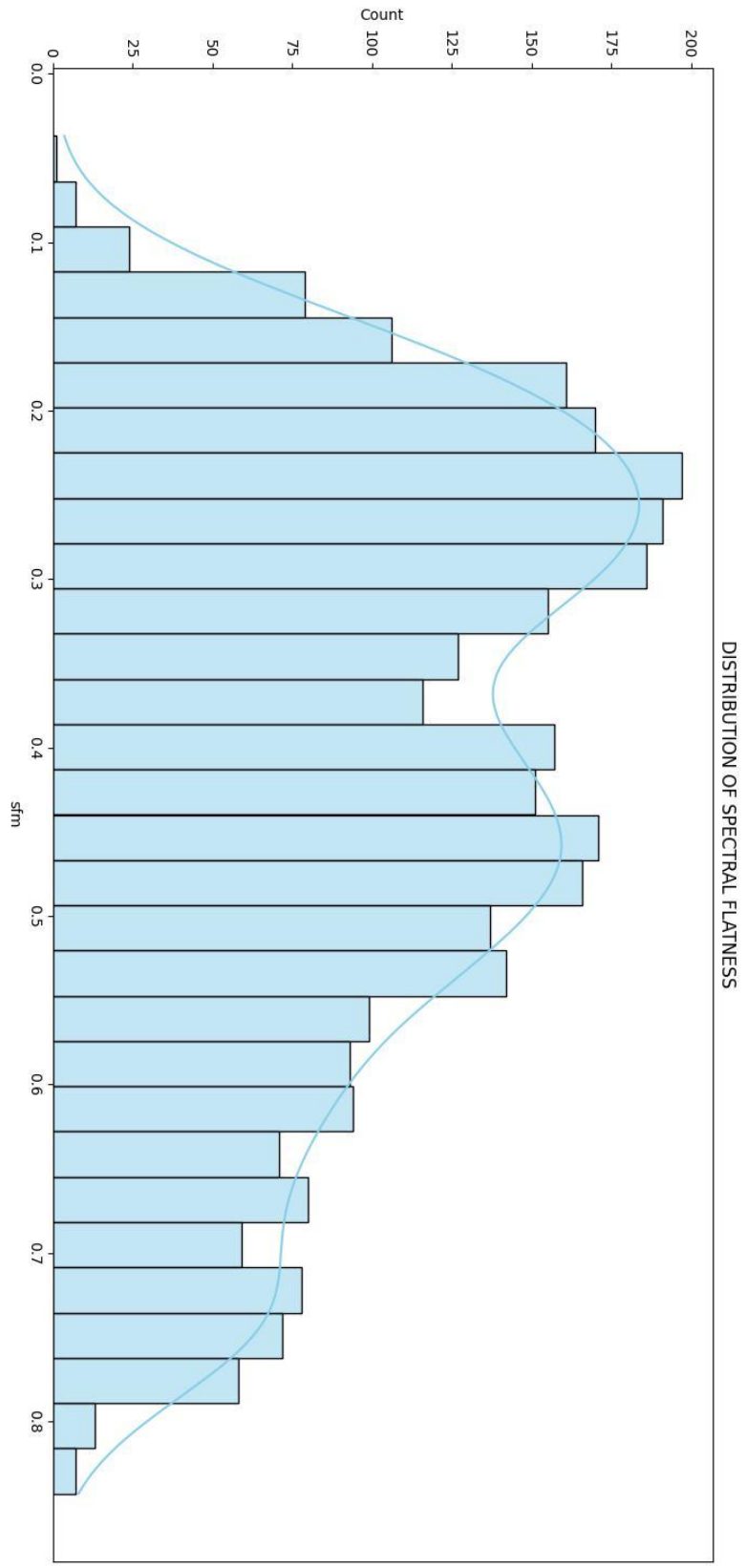
- Data entry errors (human errors)
- Measurement errors (instrument errors)
- Experimental errors (data extraction or experiment planning/executing errors)
- Intentional (dummy outliers made to test detection methods)
- Data processing errors (data manipulation or data set unintended mutations)
- Sampling errors (extracting or mixing data from wrong or various sources) •

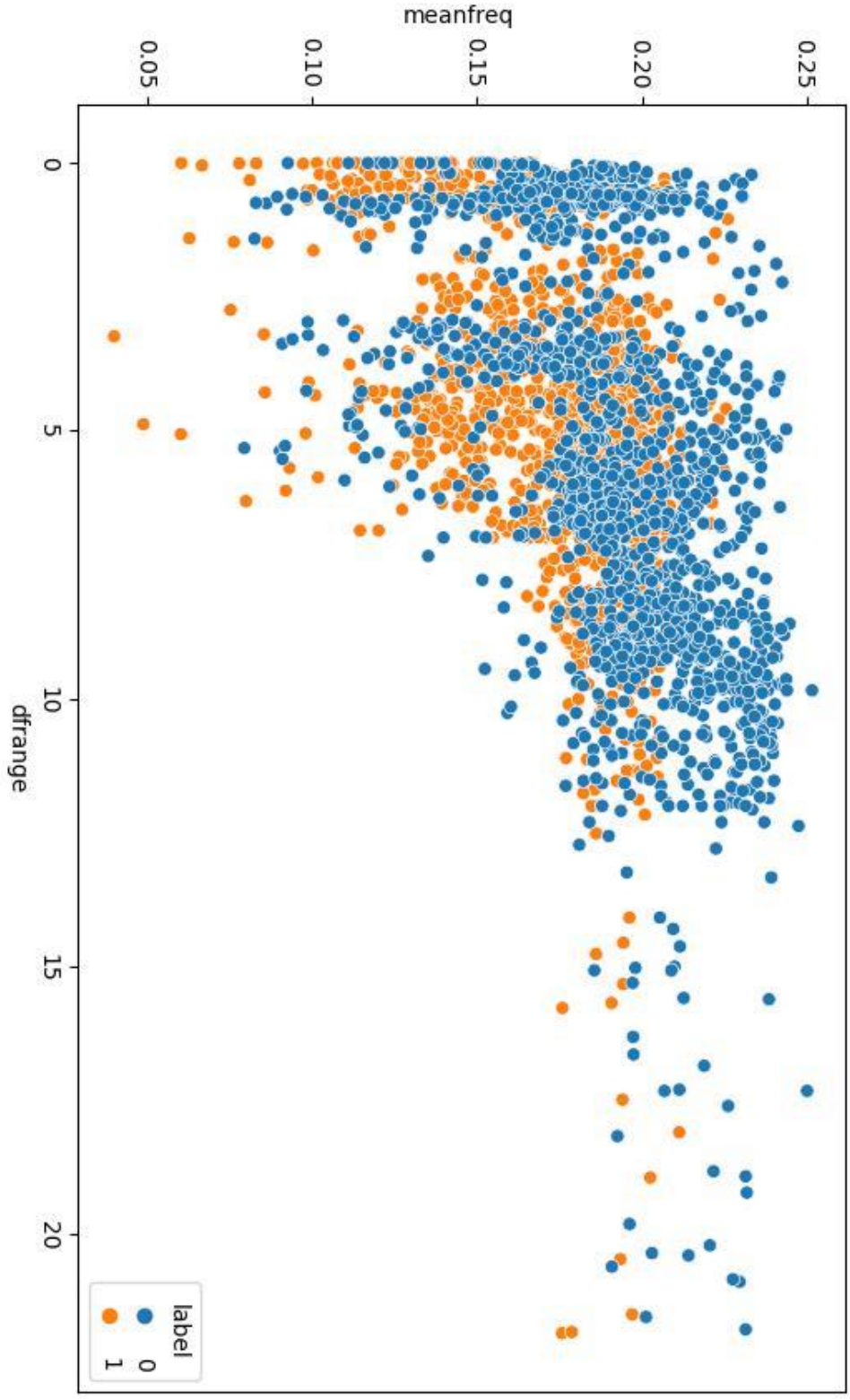
Natural (not an error, novelties in data)

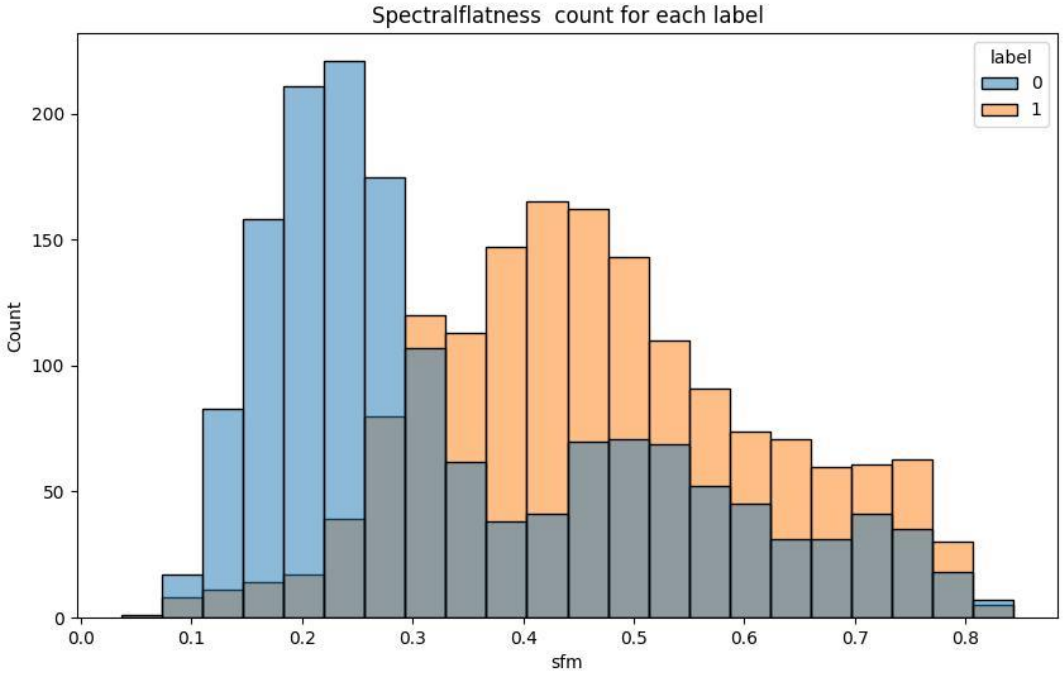
DATA-VISUALIZATION

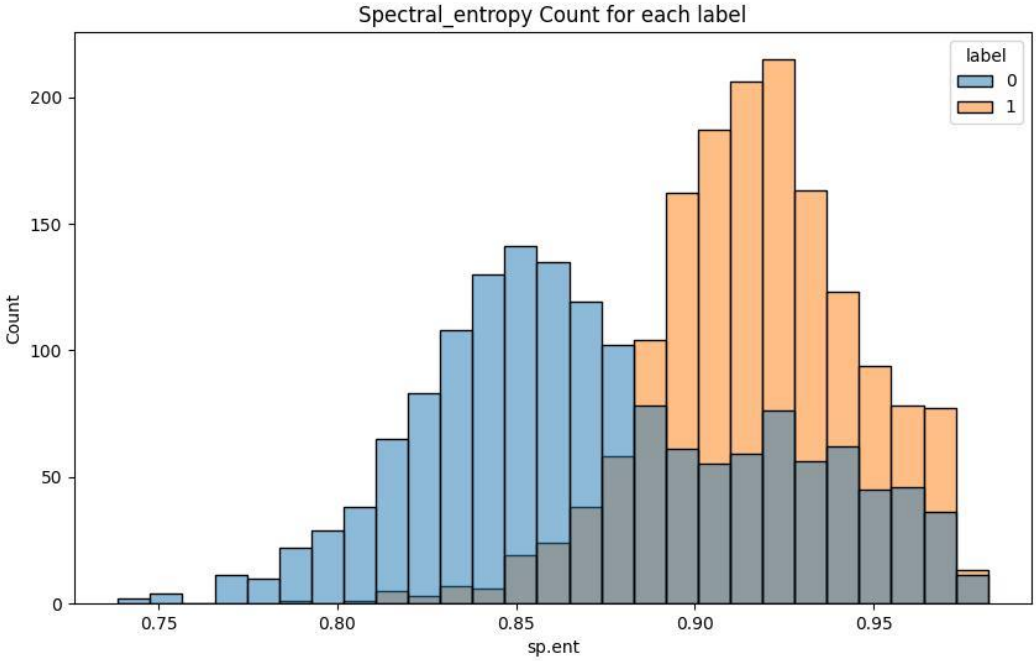





Count
0
25
50
75
100
125
150
175
200
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
sfm
DISTRIBUTION OF SPECTRAL FLATNESS


meanfreq
0.05
0.10
0.15
0.20
0.25
dfrange
0
5
10
15
20
label
0
1


Spectralflatness count for each label
label
0
1
Count
sfm
0
50
100
150
200
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8


Spectral_entropy Count for each label
label
0
1
Count
sp.ent
0
50
100
150
200
0.75
0.80
0.85
0.90
0.95

MODEL BUILDING

Splitting data for training and testing purpose

We split the given train dataset into two parts for training and testing purpose. The split ratio we used is 0.8 which indicates we used 80% data for training purpose and 20% data for testing purpose. We will be using the same split ratio for all the models trained.

1.RANDOM FOREST

- It can handle high-dimensional datasets with numerous features.
- It is resistant to overfitting and tends to generalize well to unseen data.
- It can handle both categorical and numerical features without requiring extensive data preprocessing.
- It provides measures of feature importance, allowing insights into the most influential features for classification.

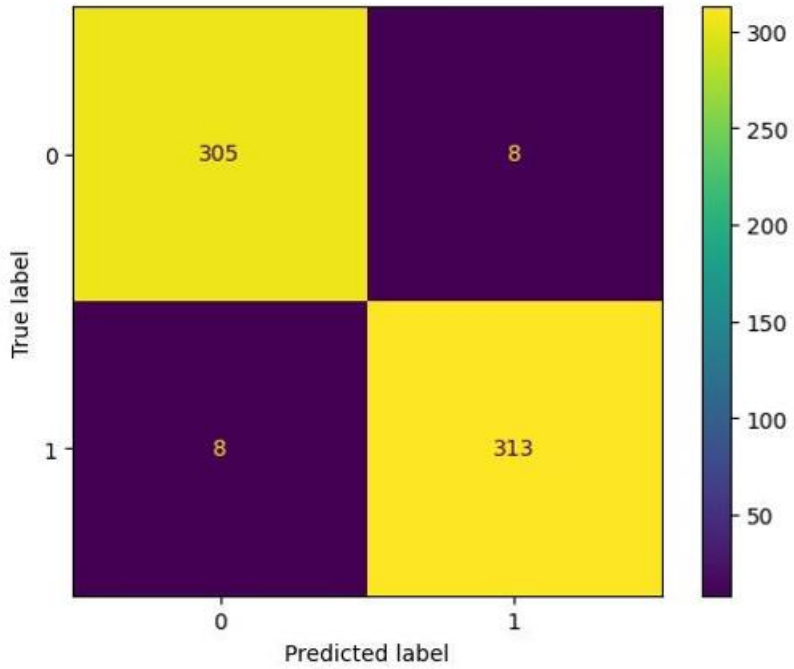
Furthermore, the Random Forest Classifier has the capability to handle imbalanced datasets, noise, and missing values to some extent. It is also relatively less sensitive to hyperparameter tuning compared to other models.

The object description of the Random Forest Classifier used is given below:

Table : Object Parameter Table for Random Forest Classifier

Object Name	**RandomForestClassifier**
Parameters	**Value**
bootstrap	True
class_weight	None
criterion	'gini'
max_depth	None
max_features	'auto'
max_leaf_nodes	None
min_impurity_decrease	0.0
min_impurity_split	None
min_samples_leaf	1
min_samples_split	2
min_weight_fraction_leaf	0.0
n_estimators	100
n_jobs	2
oob_score	False
random_state	0
verbose	1

Now we created a confusion matrix to view the actual and predicted test results. Given below is the confusion matrix:



Confusion matrix of Random Forest Classifier

We trained the Random Forest Classifier Model although it was not our goal, to find out the feature importance of the given attributes and choose the principal attributes for training our goal models. Random Forest Classifier has an inbuilt feature importance function that gives the value of importance of each attribute and helps in determining the target attribute.

The following table consists of the feature importance obtained from our Random Forest Classifier:

Feature importance table obtained from Random Forest Classifier

Feature _name	feature_important_score
1. meanfreq	0.01326822
2. sd	0.06863572
3. median	0.0167935
4. Q25	0.11866658
5. Q75	0.00947794
6. IQR	0.22054806
7. skew	0.01139696
8. kurt	0.01054253
9. sp.ent	0.05009396
10. sfm	0.02994125
11. mode	0.02068543
12. centroid	0.01926841
13. meanfun	0.3588864
14. minfun	0.00985604
15. maxfun	0.00548978
16. meandom	0.00748746
17. mindom	0.00716134
18. maxdom	0.00690919
19. dfrange	0.00738769
20. modindx	0.00750353

K Nearest Neighbor(KNN)

1.Training Phase:

1. The algorithm starts by storing the entire training dataset, which consists of labeled data points. Each data point is represented by a set of features and assigned a corresponding class label.
2. During the training phase, the algorithm simply memorizes the training data, as KNN is a lazy learner and does not build an explicit model.

2.Prediction Phase:

1. When a new unlabeled data point is given for prediction, the algorithm identifies the K nearest neighbors to that data point from the training dataset.
2. The "K" value is a user-defined hyperparameter that determines the number of neighbors to consider.
3. The distance metric, such as Euclidean distance or Manhattan distance, is used to measure the similarity between data points in the feature space.
4. The most common approach is to calculate the distances between the new data point and all other data points in the training set.
5. The K nearest neighbors are the data points with the smallest distances to the new data point.

3.**Classification** :

•For classification tasks, KNN assigns the class label to the new data point based on the majority class among its K nearest neighbors.

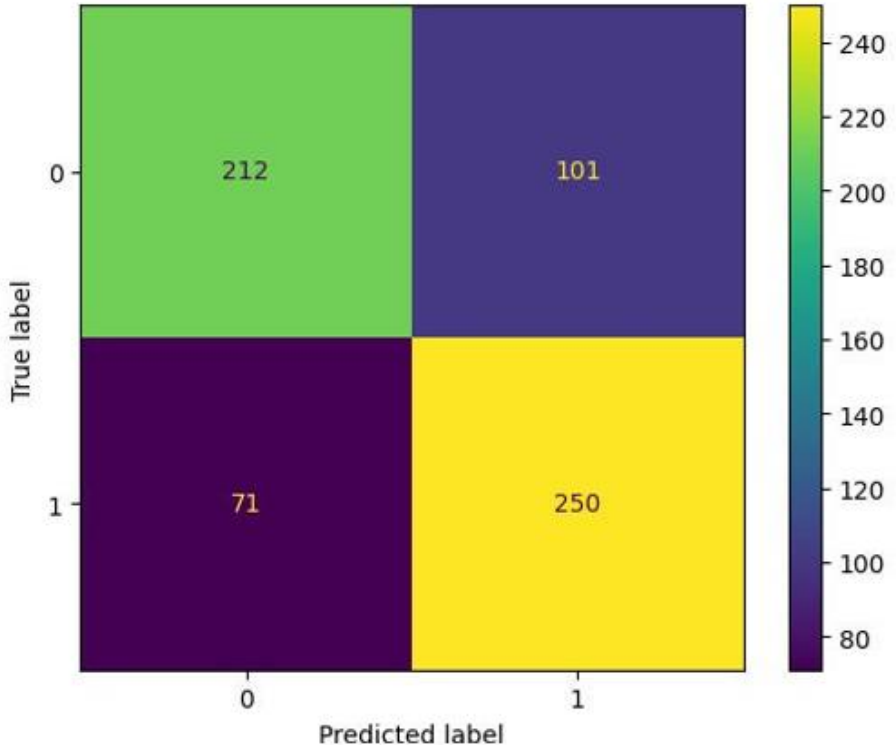
•Each neighbor's vote is given equal weight, regardless of its proximity to the new data point.

•In regression tasks, KNN predicts the value of the new data point by calculating the average or weighted average of the target values of its K nearest neighbors.

Object Parameter Table for k-NN Classifier

Object Name	**KNeighborsClassifier**
Parameters	**Value**
algorithm	'auto'
leaf_size	30
metric	'minkowski'
metric_params	None
n_neighbors	5
p	2
weights	'uniform'

Now we created a confusion matrix to view the actual and predicted test results. Given below is the confusion matrix:



Confusion matrix of k-NN Classifier

It's important to note that KNN assumes the feature space is continuous and that nearby points in the feature space are more likely to share the same class or have similar values. Additionally, the choice of K and the distance metric can significantly impact the performance of the algorithm, and they should be carefully selected based on the specific problem and data characteristics.

From the Receiver Operating Characteristic (ROC) graph, we find the Area Under Curve (AUC) for our KNN classifier model is 0.540741

AUC value of KNN = 71.75%

Now we will be preparing the classification report of our k-NN model.

Classification Report table of k-NN classifier

Classification Report of k-NN classifier model				
Accuracy			0.71	
	precision	**recall**	**f1-score**	**support**
0	0.72	0.69	0.71	312
1	0.71	0.75	0.73	322

Naive Bayes Classifier

A Naive Bayes classifier is a probabilistic machine learning model that's used for classification task. The crux of the classifier is based on the Bayes theorem. The fundamental Naive Bayes assumption is that each feature makes an:

•independent

•equal

contribution to the outcome. The assumptions made by Naive Bayes are not generally correct in real-world situations. In-fact, the independence assumption is never correct but often works well in practice.

Bayes Theorem:

Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes' theorem is stated mathematically as the following equation:

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)}$$

where A and B are events and P(B) is the probability of occurrence of event B.

Basically, we are trying to find probability of event A, given the event B is true. Event B is also termed as evidence.

P(A) is the priori of A (the prior probability, i.e. Probability of event before evidence is seen). The evidence is an attribute value of an unknown instance (here, it is event B).

P(A|B) is a posteriori probability of B, i.e. probability of event after

evidence is seen. The class-data relation from the Bayes Theorem can be

obtained as follows:

$$P(Class|Data) = \frac{P(Class)P(Data|Class)}{}$$

Where,

•P(Class|Data) = Posterior
•P(Class) = Prior
•P(Data|Class) = Likelihood
•P(Data) = Marginal Probability In other words, it can be written as:

$$Posterior = \frac{Prior * Likelihood}{Marginal\ Probability}$$

In application, we do not need to calculate the Marginal Probability for classification. We only need to calculate the numerator of the posterior for classification.

Types of Naive Bayes Classifier:

Multinomial Naive Bayes:

This is mostly used for document classification problem, i.e. whether a document belongs to the category of sports, politics, technology etc. The features/predictors used by the classifier are the frequency of the words present in the document.

Bernoulli Naive Bayes:

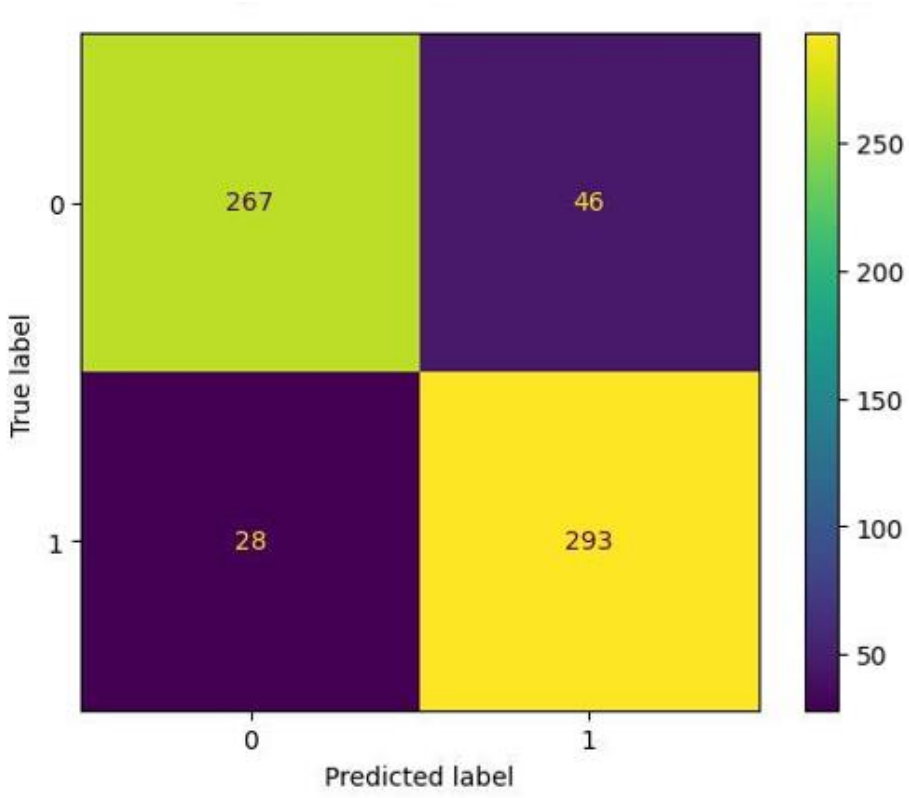
This is similar to the multinomial Naive Bayes but the predictors are Boolean variables. The parameters that we use to predict the class variable take up only values yes or no, for example if a word occurs in the text or not.

Gaussian Naive Bayes:

When the predictors take up a continuous value and are not discrete, we assume that these values are sampled from a gaussian distribution.

The object description of the Naive Bayes used is given below:

Now we created a confusion matrix to view the actual and predicted test results. Given below is the confusion matrix:



Now we will be preparing the classification report of our Gaussian Naive Bayes model.

Classification Report of Gaussian Naive Bayes model				
Accuracy			0.88	
	precision	**recall**	**f1-score**	**support**
0	0.90	0.86	0.88	312
1	0.91	0.89	0.94	322

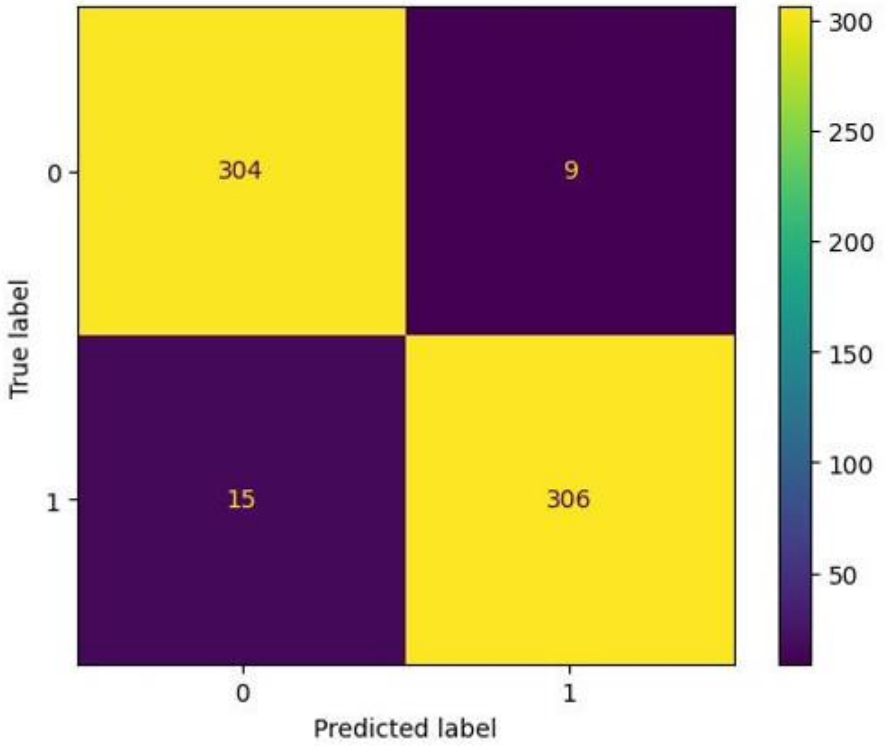
Decision Tree

Decision tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label. A tree has many analogies in real life, and turns out that it has influenced a wide area of machine learning, covering both classification and regression. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions. Though a commonly used tool in data mining for deriving a strategy to reach a particular goal, it's also widely used in machine learning.

Table 17: Object Parameter Table for Decision Tree Classifier

Object Name	**DecisionTreeClassifier**
Parameters	**Value**
class_weight	None
criterion	'gini'
max_depth	None
max_features	None
max_leaf_nodes	None
min_impurity_decrease	0.0
min_impurity_split	None
min_samples_leaf	1
min_samples_split	2
min_weight_fraction_leaf	0.0
presort	False
random_state	None
splitter	'best'

Now we created a confusion matrix to view the actual and predicted test results. Given below is the confusion matrix:



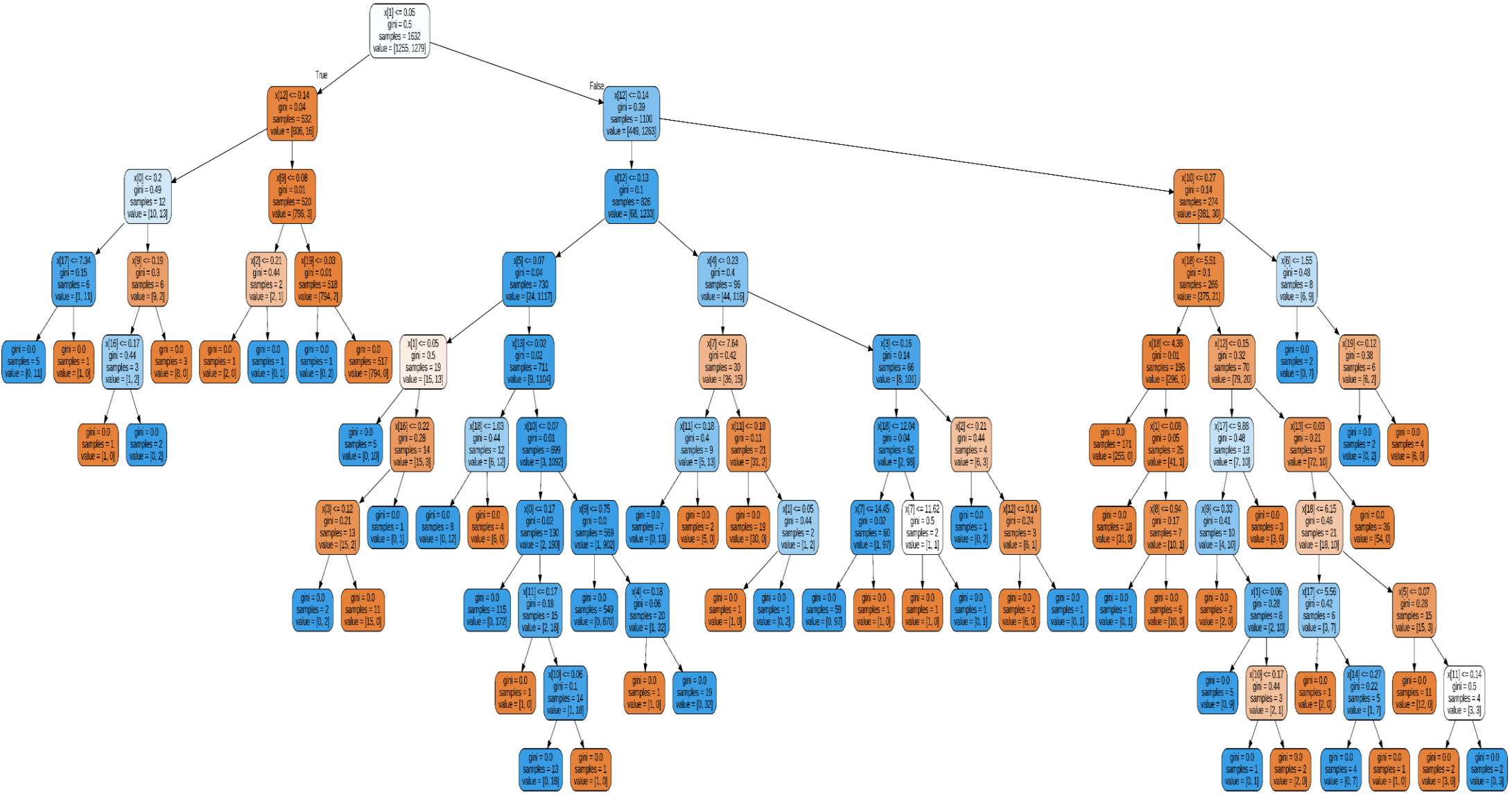
Confusion matrix of Decision Tree Classifier

AUC value of Decision Tree = 0.96

Now we will be preparing the classification report of our Decision Tree

Classification Report of Decision Tree classifier model				
Accuracy			0.96	
	precision	**recall**	**f1-score**	**support**
0	0.96	0.97	0.96	312
1	0.96	0.97	0.96	322

The decision tree prepared by the trained model is given in the next page.


x[1] <= 0.05
gini = 0.5
samples = 1632
value = [1255, 1279]
True
False
x[12] <= 0.14
gini = 0.04
samples = 532
value = [806, 16]
x[12] <= 0.14
gini = 0.39
samples = 1100
value = [449, 1263]
x[0] <= 0.2
gini = 0.49
samples = 12
value = [10, 13]
x[9] <= 0.08
gini = 0.01
samples = 520
value = [796, 3]
x[12] <= 0.13
gini = 0.1
samples = 826
value = [68, 1233]
x[10] <= 0.27
gini = 0.14
samples = 274
value = [381, 30]
x[17] <= 7.34
gini = 0.15
samples = 6
value = [1, 11]
x[9] <= 0.19
gini = 0.3
samples = 6
value = [9, 2]
x[2] <= 0.21
gini = 0.44
samples = 2
value = [2, 1]
x[19] <= 0.03
gini = 0.01
samples = 518
value = [794, 2]
x[5] <= 0.07
gini = 0.04
samples = 730
value = [24, 1117]
x[4] <= 0.23
gini = 0.4
samples = 96
value = [44, 116]
x[18] <= 5.51
gini = 0.1
samples = 266
value = [375, 21]
x[6] <= 1.55
gini = 0.48
samples = 8
value = [6, 9]
gini = 0.0
samples = 5
value = [0, 11]
gini = 0.0
samples = 1
value = [1, 0]
x[16] <= 0.17
gini = 0.44
samples = 3
value = [1, 2]
gini = 0.0
samples = 3
value = [8, 0]
gini = 0.0
samples = 1
value = [2, 0]
gini = 0.0
samples = 1
value = [0, 1]
gini = 0.0
samples = 1
value = [0, 2]
gini = 0.0
samples = 517
value = [794, 0]
x[1] <= 0.05
gini = 0.5
samples = 19
value = [15, 13]
x[13] <= 0.02
gini = 0.02
samples = 711
value = [9, 1104]
x[7] <= 7.64
gini = 0.42
samples = 30
value = [36, 15]
x[3] <= 0.16
gini = 0.14
samples = 66
value = [8, 101]
x[18] <= 4.38
gini = 0.01
samples = 196
value = [296, 1]
x[12] <= 0.15
gini = 0.32
samples = 70
value = [79, 20]
gini = 0.0
samples = 2
value = [0, 7]
x[19] <= 0.12
gini = 0.38
samples = 6
value = [6, 2]
gini = 0.0
samples = 1
value = [1, 0]
gini = 0.0
samples = 1
value = [0, 2]
gini = 0.0
samples = 5
value = [0, 10]
x[16] <= 0.22
gini = 0.28
samples = 14
value = [15, 3]
x[18] <= 1.03
gini = 0.44
samples = 12
value = [6, 12]
x[10] <= 0.07
gini = 0.01
samples = 699
value = [3, 1092]
x[11] <= 0.18
gini = 0.4
samples = 9
value = [5, 13]
x[11] <= 0.18
gini = 0.11
samples = 21
value = [31, 2]
x[18] <= 12.04
gini = 0.04
samples = 62
value = [2, 98]
x[2] <= 0.21
gini = 0.44
samples = 4
value = [6, 3]
gini = 0.0
samples = 171
value = [255, 0]
x[1] <= 0.06
gini = 0.05
samples = 25
value = [41, 1]
x[17] <= 9.86
gini = 0.48
samples = 13
value = [7, 10]
x[13] <= 0.03
gini = 0.21
samples = 57
value = [72, 10]
gini = 0.0
samples = 2
value = [0, 2]
gini = 0.0
samples = 4
value = [6, 0]
x[3] <= 0.12
gini = 0.21
samples = 13
value = [15, 2]
gini = 0.0
samples = 1
value = [0, 1]
gini = 0.0
samples = 8
value = [0, 12]
gini = 0.0
samples = 4
value = [6, 0]
x[0] <= 0.17
gini = 0.02
samples = 130
value = [2, 190]
x[9] <= 0.75
gini = 0.0
samples = 569
value = [1, 902]
gini = 0.0
samples = 7
value = [0, 13]
gini = 0.0
samples = 2
value = [5, 0]
gini = 0.0
samples = 19
value = [30, 0]
x[1] <= 0.05
gini = 0.44
samples = 2
value = [1, 2]
x[7] <= 14.45
gini = 0.02
samples = 60
value = [1, 97]
x[7] <= 11.62
gini = 0.5
samples = 2
value = [1, 1]
gini = 0.0
samples = 1
value = [0, 2]
x[12] <= 0.14
gini = 0.24
samples = 3
value = [6, 1]
gini = 0.0
samples = 18
value = [31, 0]
x[8] <= 0.94
gini = 0.17
samples = 7
value = [10, 1]
x[9] <= 0.33
gini = 0.41
samples = 10
value = [4, 10]
gini = 0.0
samples = 3
value = [3, 0]
x[18] <= 6.15
gini = 0.46
samples = 21
value = [18, 10]
gini = 0.0
samples = 36
value = [54, 0]
gini = 0.0
samples = 2
value = [0, 1]
x[11] <= 0.17
gini = 0.18
samples = 15
value = [2, 18]
gini = 0.0
samples = 549
value = [0, 870]
x[4] <= 0.18
gini = 0.06
samples = 20
value = [1, 32]
gini = 0.0
samples = 59
value = [0, 97]
gini = 0.0
samples = 1
value = [1, 0]
gini = 0.0
samples = 1
value = [1, 0]
gini = 0.0
samples = 1
value = [0, 1]
gini = 0.0
samples = 2
value = [6, 0]
gini = 0.0
samples = 1
value = [0, 1]
gini = 0.0
samples = 1
value = [0, 1]
gini = 0.0
samples = 1
value = [0, 1]
gini = 0.0
samples = 2
value = [0, 2]
gini = 0.0
samples = 2
value = [2, 0]
x[10] <= 0.06
gini = 0.1
samples = 14
value = [1, 18]
gini = 0.0
samples = 1
value = [1, 0]
gini = 0.0
samples = 19
value = [0, 32]
gini = 0.0
samples = 13
value = [0, 18]
gini = 0.0
samples = 1
value = [1, 0]
gini = 0.0
samples = 5
value = [0, 9]
x[10] <= 0.17
gini = 0.44
samples = 3
value = [2, 1]
gini = 0.0
samples = 1
value = [2, 0]
x[14] <= 0.27
gini = 0.22
samples = 5
value = [1, 7]
gini = 0.0
samples = 11
value = [12, 0]
x[11] <= 0.14
gini = 0.5
samples = 4
value = [3, 3]
gini = 0.0
samples = 1
value = [0, 1]
gini = 0.0
samples = 2
value = [2, 0]
gini = 0.0
samples = 4
value = [0, 7]
gini = 0.0
samples = 1
value = [1, 0]
gini = 0.0
samples = 2
value = [3, 0]
gini = 0.0
samples = 2
value = [0, 3]

Logistic Regression

Logistic Regression (also called Logit Regression) is commonly used to estimate the probability that an instance belongs to a particular class (e.g., what is the probability that this email is spam?). If the estimated probability is greater than 50%, then the model predicts that the instance belongs to that class (called the positive class, labelled "1"), or else it predicts that it does not (i.e., it belongs to the negative class, labelled "0"). This makes it a binary classifier.

Binary logistic regression major assumptions:

•The dependent variable should be dichotomous in nature (e.g., presence vs. absent).

•There should be no outliers in the data, which can be assessed by converting the continuous predictors to standardized scores, and removing values below -3.29 or greater than 3.29.

•There should be no high correlations (multicollinearity) among the predictors. This can be assessed by a correlation matrix among the predictors. Some authors suggest that as long correlation coefficients among independent variables are less than 0.90 the assumption is met.

At the centre of the logistic regression analysis is the task estimating the log odds of an event. Mathematically, logistic regression estimates a multiple linear regression function defined as:

logit(p) $= \log\left(\frac{p(y=1)}{1-(p=1)}\right) = \beta_0 + \beta_1 x_{i2} + \beta_2 \cdot x_{i2} + \_ + \beta_p \cdot x_{in}$

for

i = 1……n.

The object description of the Logistic Regression Classifier used is given below:

Object Name	**LogisticRegression**
Parameters	**Value**
C=1.0	C=1.0
solver	'lbfgs'
class_weight	None
dual	False
fit_intercept	True
intercept_scaling	1
max_iter	100
multi_class	'warn'
n_jobs	None
penalty	'l2'
random_state=None	None
tol	0.0001
verbose	0
warm_start	False

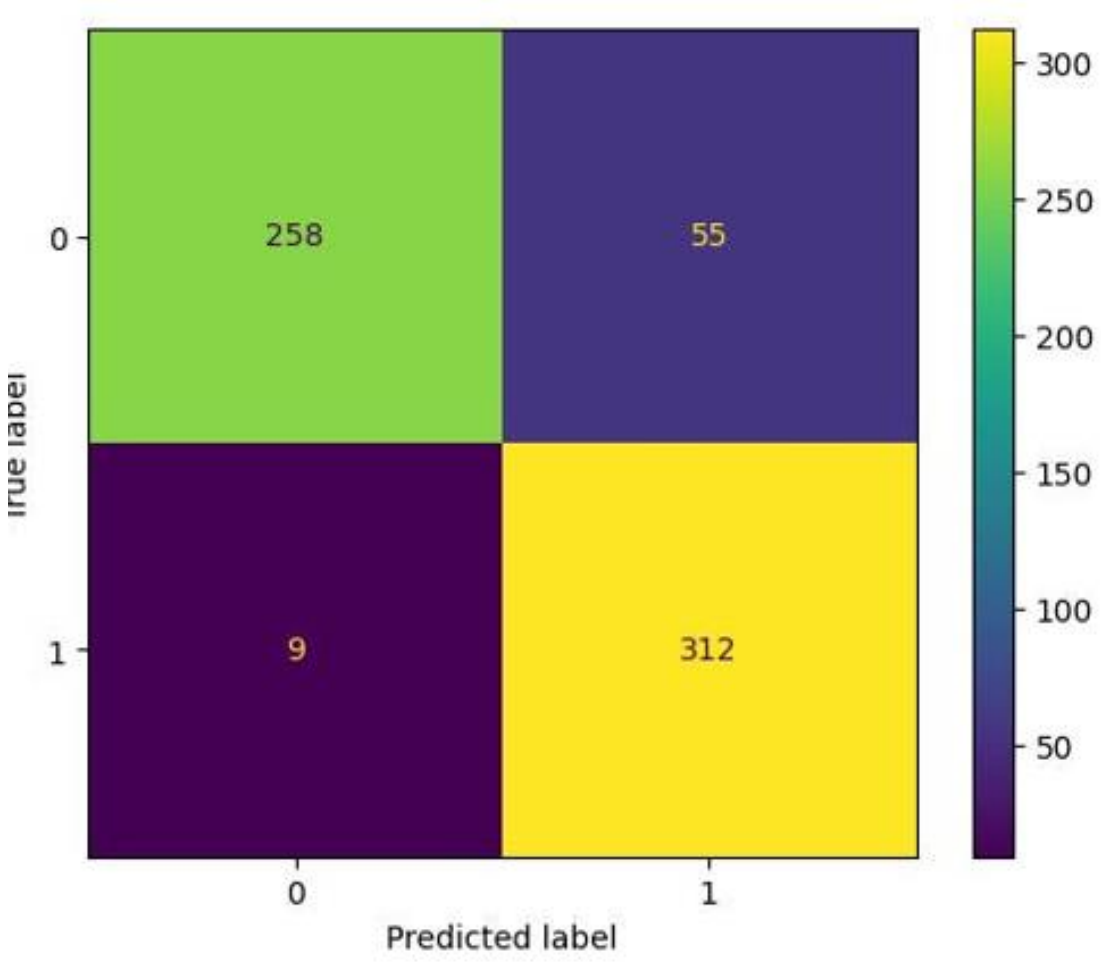
The intercept of the trained model is:

```
0.44840069
```

Table 21: Coefficients of attributes if the trained Logistic Regression Model

Feature _name	feature_coefficients
1. meanfreq	1.10763152e+00
2. sd	1.74636485e+00
3. median	-1.37654350e+00
4. Q25	-6.87749409e+00
5. Q75	2.72434597e+00
6. IQR	9.60184006e+00
7. skew	-3.71503007e-01
8. kurt	1.00609192e-02
9. sp.ent	4.59590249e+00
10. sfm	-8.00937412e-01
11. mode	1.24458278e+00
12. centroid	-1.10763152e+00
13. meanfun	-1.51311407e+01
14. minfun	1.01690995e-01
15. maxfun	-2.28065710e+00
16. meandom	3.13922111e-01
17. mindom	-1.49491999e+00
18. maxdom	-8.15295519e-01
19. dfrange	6.79624470e-01
20. modindx	-1.88547318e+00

Now we created a confusion matrix to view the actual and predicted test results. Given below is the confusion matrix:



AUC value of Logistic Regression = 0.881481.

Now we will be preparing the classification report of our Logistic Regression model.

Classification Report of Logistic Regression Classifier model				
Accuracy			0.8754	
	precision	**recall**	**f1-score**	**support**
0	0.95	0.79	0.86	312
1	0.82	0.96	0.89	322

Comparison of the Models trained :-

We trained 5 models using the 5 algorithms viz.

1.k-Nearest Neighbour
2.Gaussian Naive Bayes
3.Decision Tree and
4.Logistic Regression
5.Random Forest

The 5 models had different accuracy. The comparison of the accuracies of the models are given below:

Table 24: Accuracy Comparison table

Model	Accuracy %
k-Nearest Neighbour	71.76
Gaussian Naive Bayes	88.32
Decision Tree	96.53
Logistic Regression	87.53
Random Forest	97.94

The following bar graph shows the accuracy comparison in graphical way:

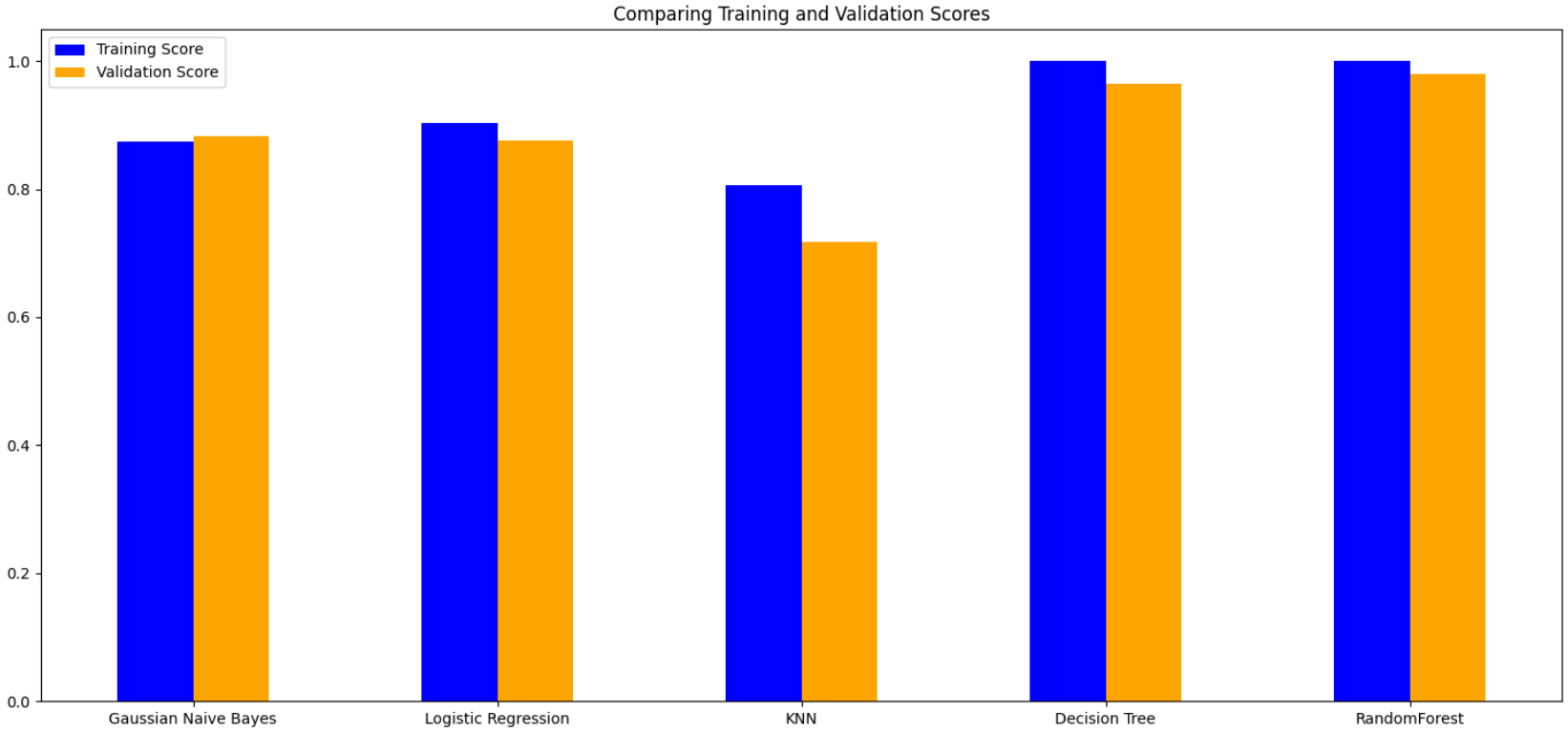


Fig 12: Comparison of accuracy of the 4 different models trained

Thus, from the above comparison we can see that Gaussian Naive Bayes classifier has the highest accuracy. So, our selected model is Gaussian Naive Bayes Classifier Model.

Code

GENDER RECOGIZATION USING VOICE (ACOUSTIC VALUES)

Problem Statement

Gender Recognition by Voice and Speech Analysis - This is a dataset originally created to identify a voice as male or female, based upon acoustic properties of the voice and speech. The dataset consists of 3,168 recorded voice samples, collected from male and female speakers. The voice samples are pre-processed by acoustic analysis in R using the seewave and tuneR packages, with an analyzed frequency range of 0hz-280hz (human vocal range).

Aim - Split the given dataset into training and validation datasets and build a classifier on the training data and validate its performance on the validation dataset. ¶

Step 1: Importing required libraries

In [53]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import BernoulliNB, GaussianNB

import sklearn
from sklearn import metrics

import warnings
warnings.filterwarnings('ignore')
```

Step 2: Data-Analysis

1.Importing the dataset:

In [5]:
```
voice_data=pd.read_csv('/content/drive/MyDrive/Colab Notebooks/PROJECT/voice.csv')
voice_data.head()
```

Out[5]:

	meanfreq	sd	median	Q25	Q75	IQR	skew	kurt	sp.ent	sfm	...	centroid	meanfun	minfun	maxfun	me
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	0.015702	0.275862	0
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	0.015826	0.250000	0
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	0.015656	0.271186	0
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	0.017798	0.250000	0
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	0.016931	0.266667	0

5 rows × 21 columns

2.Data-proporties

Shape:

In [6]:
```
Shape=voice_data.shape
print(f" no.of rows={Shape[0]}\n no.of columns={Shape[1]}")
```

```
 no.of rows=3168
 no.of columns=21
```

Columns:

In [7]:
```
cols=pd.DataFrame(data=voice_data.columns,index=range(1,22))
cols
```

Out[7]:

	0
1	meanfreq
2	sd
3	median
4	Q25
5	Q75
6	IQR
7	skew
8	kurt
9	sp.ent
10	sfm
11	mode
12	centroid
13	meanfun
14	minfun
15	maxfun
16	meandom
17	mindom
18	maxdom
19	dfrange
20	modindx
21	label

Information of the dataset:

In [8]:
```
voice_data.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
```

DataType of each column

```
In [9]: voice_data.dtypes
```

```
Out[9]: meanfreq    float64
        sd          float64
        median      float64
        Q25         float64
        Q75         float64
        IQR         float64
        skew        float64
        kurt        float64
        sp.ent      float64
        sfm         float64
        mode        float64
        centroid    float64
        meanfun     float64
        minfun      float64
        maxfun      float64
        meandom     float64
        mindom      float64
        maxdom      float64
        dfrange     float64
        modindx     float64
        label        object
        dtype: object
```

Description of the dataset

```
In [10]: voice_data.describe().T
```

Out[10]:

	count	mean	std	min	25%	50%	75%	max
meanfreq	3168.0	0.180907	0.029918	0.039363	0.163662	0.184838	0.199146	0.251124
sd	3168.0	0.057126	0.016652	0.018363	0.041954	0.059155	0.067020	0.115273
median	3168.0	0.185621	0.036360	0.010975	0.169593	0.190032	0.210618	0.261224
Q25	3168.0	0.140456	0.048680	0.000229	0.111087	0.140286	0.175939	0.247347
Q75	3168.0	0.224765	0.023639	0.042946	0.208747	0.225684	0.243660	0.273469
IQR	3168.0	0.084309	0.042783	0.014558	0.042560	0.094280	0.114175	0.252225
skew	3168.0	3.140168	4.240529	0.141735	1.649569	2.197101	2.931694	34.725453
kurt	3168.0	36.568461	134.928661	2.068455	5.669547	8.318463	13.648905	1309.612887
sp.ent	3168.0	0.895127	0.044980	0.738651	0.861811	0.901767	0.928713	0.981997
sfm	3168.0	0.408216	0.177521	0.036876	0.258041	0.396335	0.533676	0.842936
mode	3168.0	0.165282	0.077203	0.000000	0.118016	0.186599	0.221104	0.280000
centroid	3168.0	0.180907	0.029918	0.039363	0.163662	0.184838	0.199146	0.251124
meanfun	3168.0	0.142807	0.032304	0.055565	0.116998	0.140519	0.169581	0.237636
minfun	3168.0	0.036802	0.019220	0.009775	0.018223	0.046110	0.047904	0.204082
maxfun	3168.0	0.258842	0.030077	0.103093	0.253968	0.271186	0.277457	0.279114
meandom	3168.0	0.829211	0.525205	0.007812	0.419828	0.765795	1.177166	2.957682
mindom	3168.0	0.052647	0.063299	0.004883	0.007812	0.023438	0.070312	0.458984
maxdom	3168.0	5.047277	3.521157	0.007812	2.070312	4.992188	7.007812	21.867188
dfrange	3168.0	4.994630	3.520039	0.000000	2.044922	4.945312	6.992188	21.843750
modindx	3168.0	0.173752	0.119454	0.000000	0.099766	0.139357	0.209183	0.932374

Step 3 : Data Preprocess

1.Data cleaning (if null values present) ,

2.Label Encoding (for Target column) ,

3.Shuffle the dataset ,

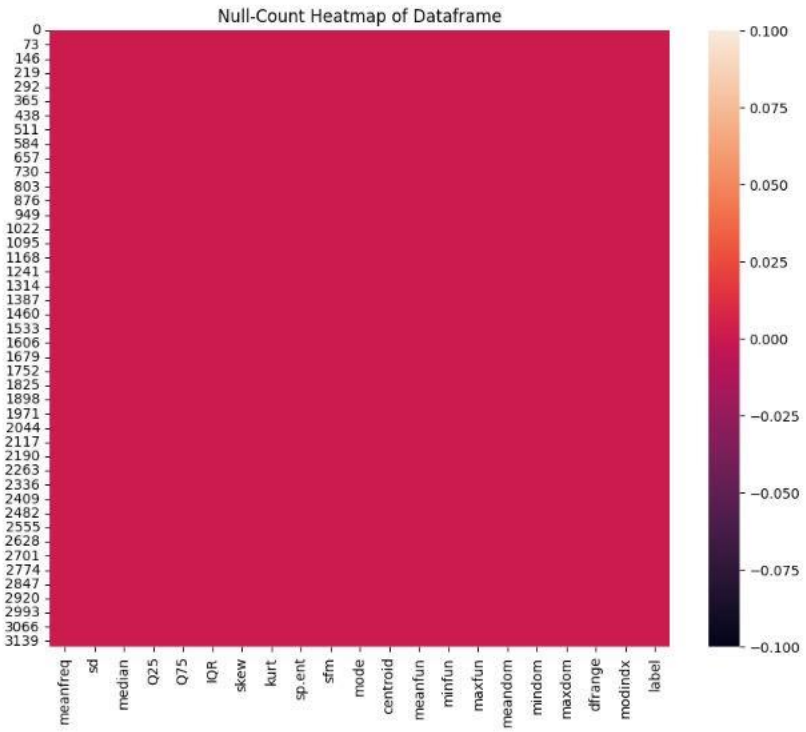
4.Split the data into training and testing sets.

In [11]:
```
#checking for nullvalues
df1=voice_data.count()
df2=voice_data.isnull().sum()
null_info=pd.concat([df1,df2],axis=1)
null_info=null_info.rename(columns={0:'Total_count',1:'NUll_count'})
null_info
```

Out[11]:

	Total_count	NUll_count
meanfreq	3168	0
sd	3168	0
median	3168	0
Q25	3168	0
Q75	3168	0
IQR	3168	0
skew	3168	0
kurt	3168	0
sp.ent	3168	0
sfm	3168	0
mode	3168	0
centroid	3168	0
meanfun	3168	0
minfun	3168	0
maxfun	3168	0
meandom	3168	0
mindom	3168	0
maxdom	3168	0
dfrange	3168	0
modindx	3168	0
label	3168	0

In [12]:
```
#NULL-COUNT Heatmap

plt.figure(figsize=(10, 8))
dataplot = sns.heatmap(voice_data.isnull())
plt.title("Null-Count Heatmap of Dataframe")
plt.savefig('null_count_heatmap.jpg')
plt.show()
```



In [13]:
```
#before label encoding
print(voice_data['label'],voice_data['label'].dtype)
```

```
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 object
```

In [14]:
```
#Encoding the label because label is obj type system only understand numeric values
label_encoder=LabelEncoder()
voice_data['label']=label_encoder.fit_transform(voice_data['label'])
```

In [15]:
```
# after label encoding
print(voice_data['label'],voice_data['label'].dtype)
```

```
0       1
1       1
2       1
3       1
4       1
       ..
3163    0
3164    0
3165    0
3166    0
3167    0
Name: label, Length: 3168, dtype: int64 int64
```

In [16]:
```
#befor shuffling the data
voice_data.head(5)
```

Out[16]:

	meanfreq	sd	median	Q25	Q75	IQR	skew	kurt	sp.ent	sfm	...	centroid	meanfun	minfun	maxfun	me
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	0.015702	0.275862	0
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	0.015826	0.250000	0
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	0.015656	0.271186	0
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	0.017798	0.250000	0
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	0.016931	0.266667	0

5 rows × 21 columns

In [17]:
```
#after shuffling the data
shuffled_voice_data=voice_data.sample(frac=1).reset_index(drop=True)
shuffled_voice_data.head(5)
```

Out[17]:

	meanfreq	sd	median	Q25	Q75	IQR	skew	kurt	sp.ent	sfm	...	centroid	meanfun	minfun	maxfun	meanc
0	0.166046	0.057188	0.177352	0.108921	0.208147	0.099226	1.529905	5.848867	0.931908	0.444206	...	0.166046	0.095105	0.016377	0.262295	0.481
1	0.187154	0.053048	0.201888	0.167039	0.223119	0.056080	2.974554	14.106474	0.873341	0.386045	...	0.187154	0.159235	0.016129	0.238806	0.637
2	0.210379	0.044542	0.207963	0.187810	0.247841	0.060031	2.092086	8.047210	0.880201	0.311898	...	0.210379	0.158242	0.047013	0.274286	1.335
3	0.235271	0.029317	0.236563	0.223198	0.251933	0.028735	2.141051	7.623588	0.803982	0.124648	...	0.235271	0.185272	0.048780	0.279070	1.992
4	0.189991	0.060324	0.204712	0.126556	0.239846	0.113291	0.961993	3.121155	0.923729	0.397331	...	0.189991	0.129917	0.048096	0.279070	1.271

5 rows × 21 columns

In [18]:
```
#splitting the data
x=shuffled_voice_data.iloc[:,:-1]
y=shuffled_voice_data['label']
```

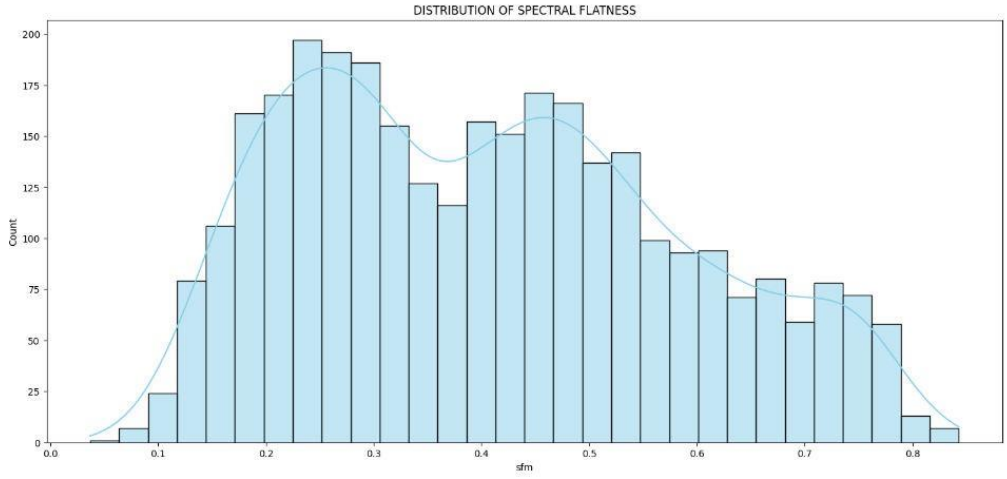
In [19]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
print("no.of samples in train_set:",x_train.shape[0])
print("no.of samples in test_set:",x_test.shape[0])
```

```
no.of samples in train_set: 2534
no.of samples in test_set: 634
```

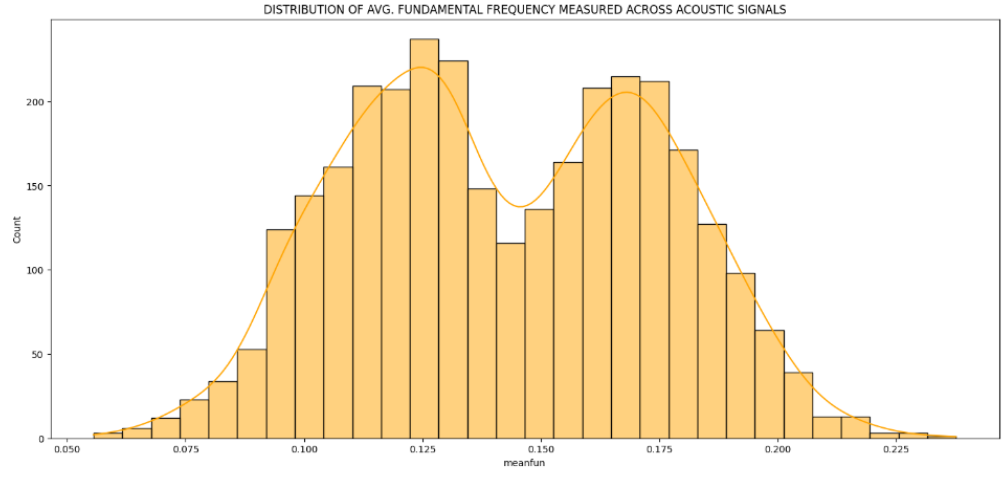
Explotary Data Analysis(EDA)

An approach to analyze the data using visual techniques. It is used to discover trends, patterns, or to check assumptions with the help of statistical summary and graphical representations.

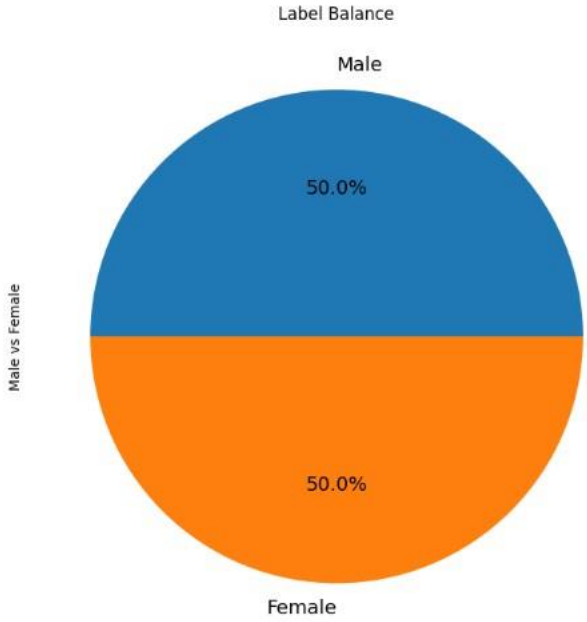
In [20]:
```
#checking the distribution of data.

plt.figure(figsize=(18,8))
sns.histplot(voice_data.sfm, color='skyblue',bins=30,kde=True)
plt.title("DISTRIBUTION OF SPECTRAL FLATNESS")
plt.savefig('DISTRIBUTION_OF_SPECTRAL_FLATNESS.jpg')
plt.show()
```



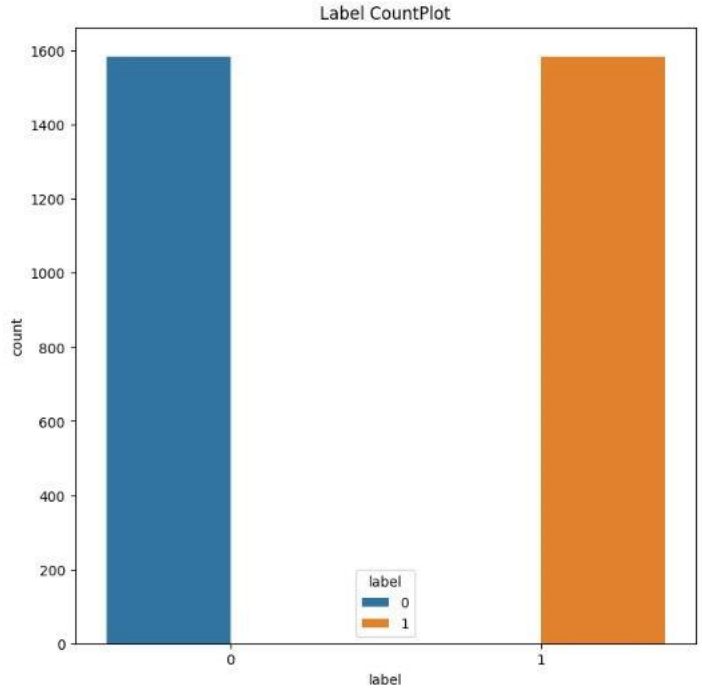
In [21]:
```
plt.figure(figsize=(18,8))
sns.histplot(voice_data.meanfun, color='orange',bins=30,kde=True)
plt.title("DISTRIBUTION OF AVG. FUNDAMENTAL FREQUENCY MEASURED ACROSS ACOUSTIC SIGNALS")
plt.savefig('Avg_Fundamenta_fre.jpg')
plt.show()
```



In [22]:
```
#Label Balance verification

plt.figure(figsize=(10, 8))
voice_data.label.value_counts().plot(kind="pie",
                                     fontsize=14,
                                     labels=["Male", "Female"],
                                     ylabel="Male vs Female",
                                     autopct='%1.1f%%');
plt.title("Label Balance")
plt.savefig('label_balance.jpg')
plt.show()
```

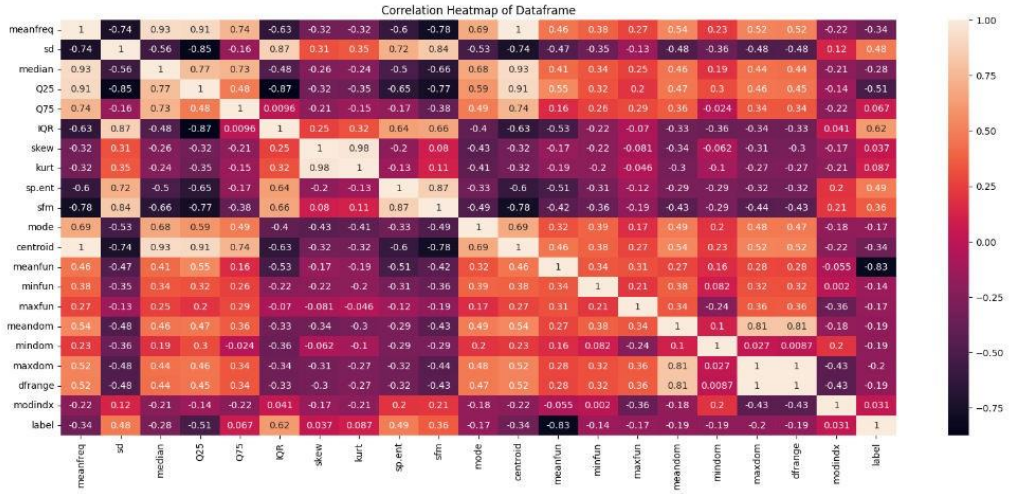


In [23]:
```
plt.figure(figsize=(8, 8))
sns.countplot(x="label",data=voice_data, hue='label')
plt.title("Label CountPlot")
plt.savefig('label_countplot.jpg')
plt.show()
```

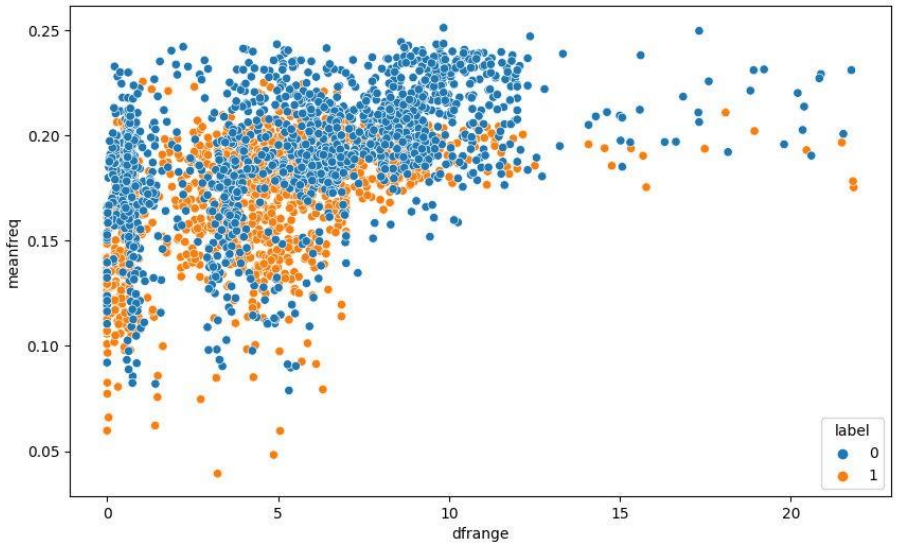


In [ ]:
```
#correlation values
corr_values=voice_data.corr()
```

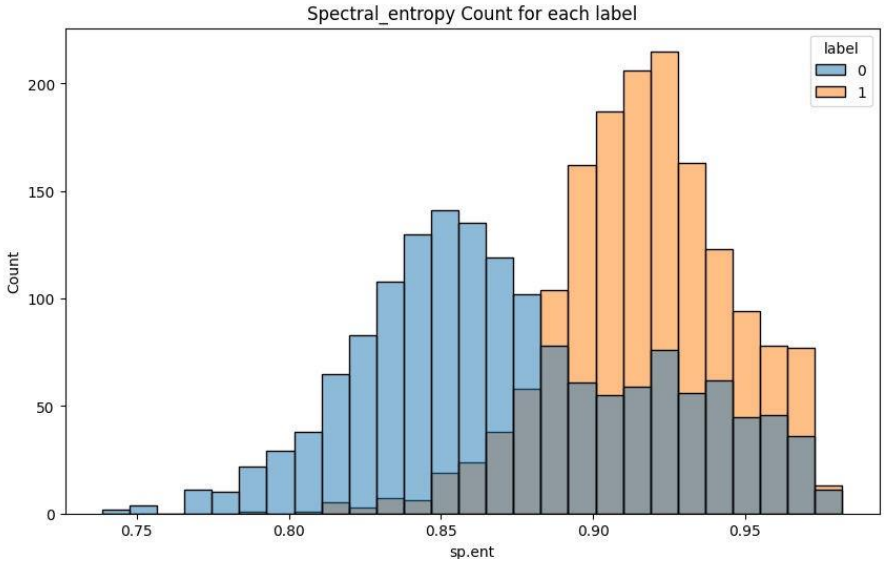
In [25]:
```
#Correlation Heatmap

plt.figure(figsize=(20, 8))
dataplot = sns.heatmap(voice_data.corr(), annot=True)
plt.title("Correlation Heatmap of Dataframe")
plt.savefig('correlation_heatmap.jpg')
plt.show()
```



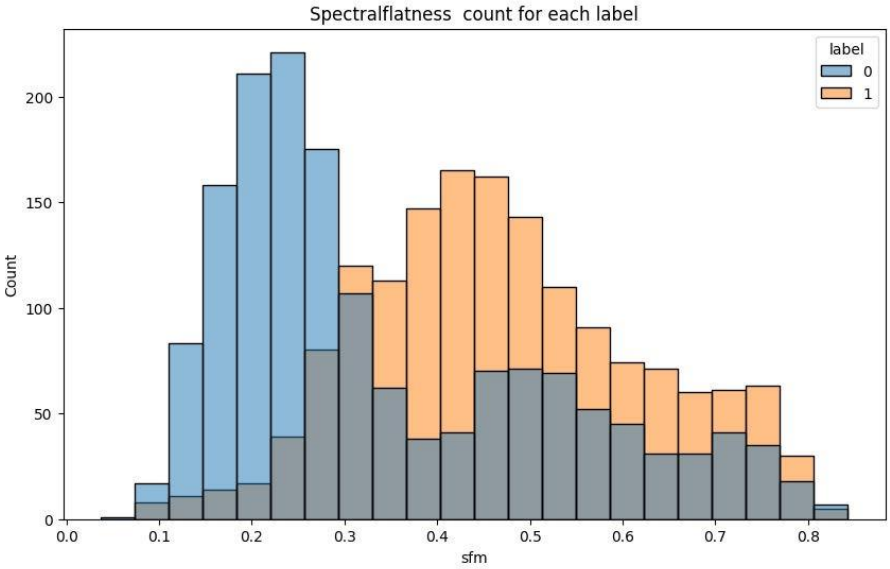
In [26]:
```
plt.figure(figsize=(10,6))
sns.scatterplot(data=voice_data, x="dfrange", y="meanfreq", hue="label");
plt.savefig('scatterplot.jpg')

#dfrange: range of dominant frequency measured across acoustic signal
#meanfreq: mean frequency (in kHz)
```



In [27]:
```
#sp.ent: spectral entropy

plt.figure(figsize=(10,6))
sns.histplot(data=voice_data,x="sp.ent", hue="label");
plt.title('Spectral_entropy Count for each label')
plt.savefig('sp_ent.jpg')
```



In [28]:
```
#sfm: spectral flatness

plt.figure(figsize=(10,6))
sns.histplot(data=voice_data,x="sfm", hue="label");
plt.title('Spectralflatness  count for each label')
plt.savefig('sfn.jpg')
```



MODELS BUILDING

REGRESSION MODEL

In [30]:
```
lr_classifier_model=LogisticRegression()
lr_classifier_model.fit(x_train,y_train)
```

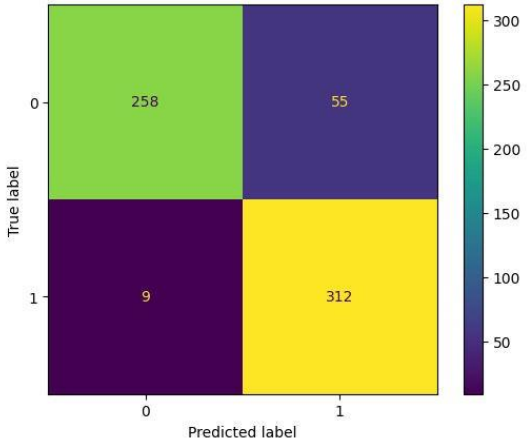
Out[30]:
```
LogisticRegression()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [31]:
```
lr_test_accuracy = lr_classifier_model.score(x_test, y_test)
print(f"Validation Accuracy of Logisitic Regression Classifier is: {(lr_test_accuracy )*100:.2f}%")
```

```
Validation Accuracy of Logisitic Regression Classifier is: 89.91%
```

In [35]:
```
sklearn.metrics.ConfusionMatrixDisplay.from_estimator(lr_classifier_model, x_test, y_test)
```

Out[35]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fd500bf39a0>



KNN Classifier

In [39]:
```
knn_classifier_model = KNeighborsClassifier(n_neighbors=5) # We always use odd numbers for this to avoid ties
knn_classifier_model.fit(x_train, y_train)
```

Out[39]: KNeighborsClassifier()

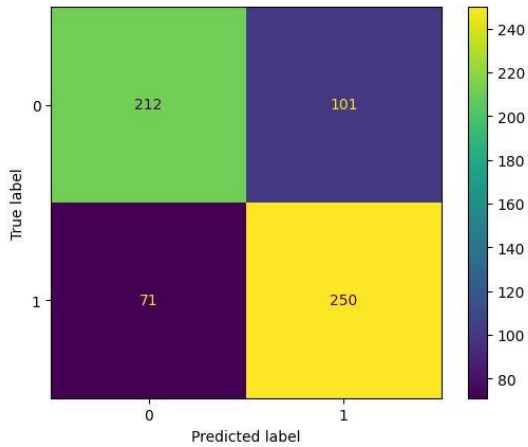
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [41]:
```
knn_test_accuracy = knn_classifier_model.score(x_test, y_test)
print(f"Validation Accuracy of KNN Clf. is: {(knn_test_accuracy)*100:.2f}%")
```

```
Validation Accuracy of KNN Clf. is: 72.87%
```

In [42]:
```
sklearn.metrics.ConfusionMatrixDisplay.from_estimator(knn_classifier_model, x_test, y_test)
```

Out[42]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fd50119dc00>



Decision Tree Classifier

In [44]:
```
dt_classifier_model = DecisionTreeClassifier()
dt_classifier_model.fit(x_train, y_train)
```

Out[44]: DecisionTreeClassifier()

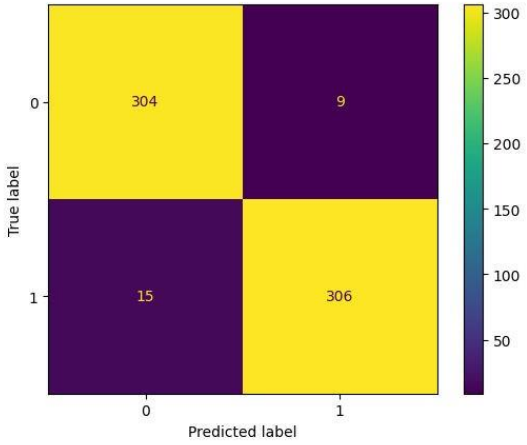
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [46]:
```
dt_test_accuracy = dt_classifier_model.score(x_test, y_test)
print(f"Validation Accuracy of Decision Tree Clf. is: {(dt_test_accuracy)*100:.2f}%")
```

```
Validation Accuracy of Decision Tree Clf. is: 96.21%
```

In [47]:
```
sklearn.metrics.ConfusionMatrixDisplay.from_estimator(dt_classifier_model, x_test,y_test)
```

Out[47]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fd509b25cf0>
```



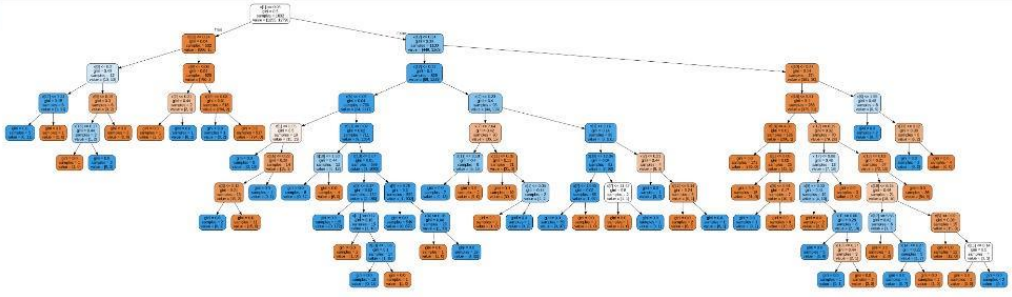
Random Forest

In [49]:
```
rfc_model = RandomForestClassifier()
rfc_model.fit(x_train, y_train)
```

Out[49]:
```
RandomForestClassifier()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [66]:
```
from sklearn.tree import export_graphviz

estimator = rfc_model.estimators_[5]

export_graphviz(estimator,
                out_file='tree.dot',
                rounded = True, proportion = False,
                precision = 2, filled = True)

from subprocess import call
call(['dot', '-Tpng', 'tree.dot', '-o', 'tree.png', '-Gdpi=600'])

from IPython.display import Image
Image(filename = 'tree.png')
```

Out[66]:

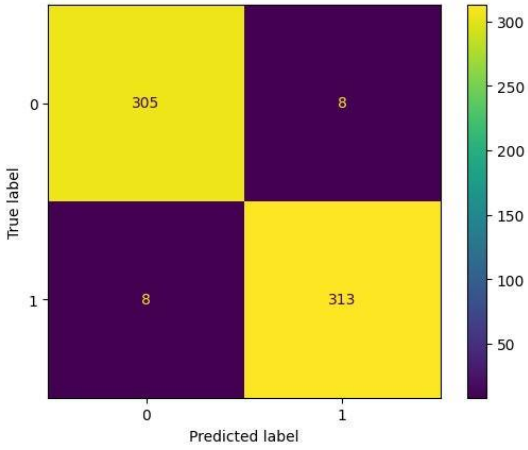


In [51]:
```
rfc_test_accuracy = rfc_model.score(x_test,y_test)
print(f"Validation Accuracy of Random Forest Classifier is: {(rfc_test_accuracy)*100:.2f}%")
```

```
Validation Accuracy of Random Forest Classifier is: 97.48%
```

In [52]:
```
sklearn.metrics.ConfusionMatrixDisplay.from_estimator(rfc_model, x_test,y_test)
```

Out[52]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fd500a35f60>
```



Naive Baye's

In [55]:
```
nb_classifier_model = GaussianNB()
nb_classifier_model.fit(x_train, y_train)
```

Out[55]:
```
GaussianNB()
```

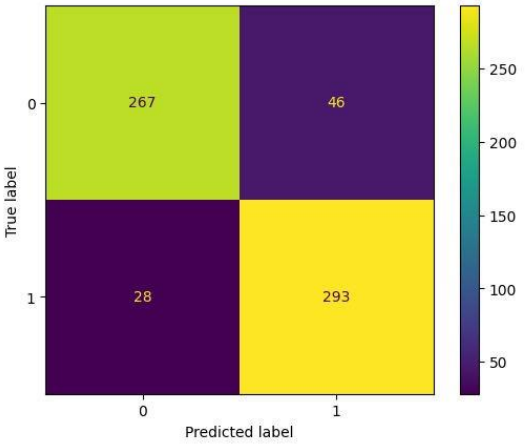
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [56]:
```
nb_test_accuracy = nb_classifier_model.score(x_test,y_test)
print(f"Validation Accuracy of Naive Bayes Classifier is: {(nb_test_accuracy)*100:.2f}%")
```

```
Validation Accuracy of Naive Bayes Classifier is: 88.33%
```

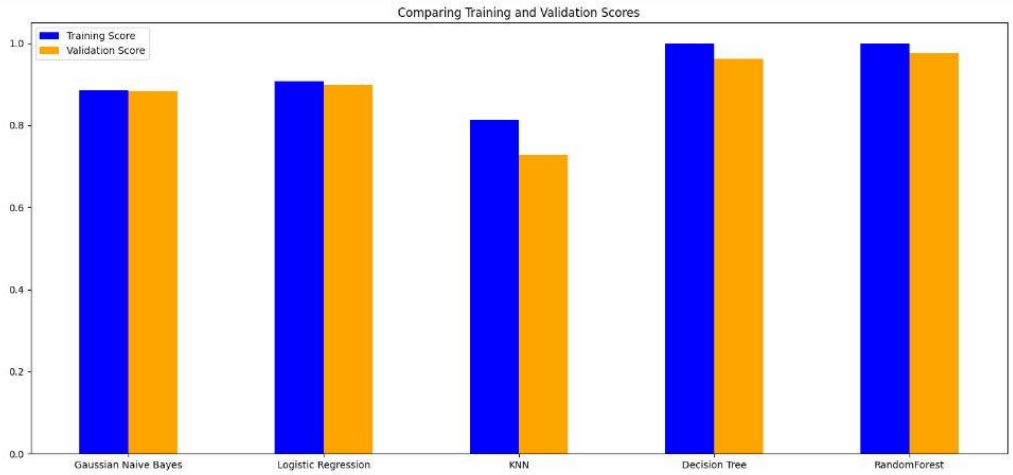
In [59]:
```
sklearn.metrics.ConfusionMatrixDisplay.from_estimator(nb_classifier_model, x_test, y_test)
```

Out[59]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fd550137f40>
```



COMPARING ALL MODEL USING ACCURACY

In [61]:
```
trainScores = [nb_classifier_model.score(x_train, y_train),lr_classifier_model.score(x_train, y_train) ,knn_classifier_model.
valScores = [nb_test_accuracy,lr_test_accuracy,knn_test_accuracy,dt_test_accuracy,rfc_test_accuracy]
indices = ['Gaussian Naive Bayes', 'Logistic Regression', 'KNN','Decision Tree', 'RandomForest']
scores = pd.DataFrame({'Training Score': trainScores,'Validation Score': valScores}, index=indices)
plot = scores.plot.bar(figsize=(18, 8), rot=0, color=['blue', 'orange'])
plt.title('Comparing Training and Validation Scores')
plt.show()
```



In [65]:
```
models = pd.DataFrame({
    'Model': ['Gaussian Naive Bayes', 'Logistic Regression', 'KNN','Decision Tree', 'RandomForest'],
    'Score': [nb_test_accuracy,lr_test_accuracy,knn_test_accuracy,dt_test_accuracy,rfc_test_accuracy]})

models.sort_values(by='Score', ascending=False)
```

Out[65]:

	Model	Score
4	RandomForest	0.974763
3	Decision Tree	0.962145
1	Logistic Regression	0.899054
0	Gaussian Naive Bayes	0.883281
2	KNN	0.728707

FUTURE SCOPE

Title: Future Scope of Gender Recognition Using Voice

1.Introduction:

1. Brief overview of gender recognition using voice and its importance in various applications.
2. Discussion on the current state of gender recognition technologies and their limitations.
3. Introduction to the concept of future scope and advancements in the field.

2.Advancements in Acoustic Feature Extraction:

1. Exploration of novel acoustic features that can improve gender recognition accuracy.
2. Investigation into advanced signal processing techniques for feature extraction.
3. Integration of multi-modal data (e.g., speech and facial cues) for enhanced gender recognition.

3.Deep Learning Approaches:

1. Utilization of deep learning models, such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and transformer-based architectures.
2. Investigation into transfer learning techniques to leverage large-scale pre-trained models.

4. Robustness to Vocal Variations:
    1. Addressing challenges related to vocal variations due to age, accent, language, and emotional states.
    2. Development of adaptive models capable of recognizing gender across diverse populations.
    3. Investigation into the fusion of demographic information with voice-based features for improved accuracy.
5. Real-Time and Embedded Systems:
    1. Optimization of gender recognition algorithms for real-time and resource-constrained environments.
    2. Exploration of lightweight models and feature selection techniques for efficient implementation.
    3. Integration of gender recognition in voice-enabled devices, IoT systems, and edge computing platforms.
6. Mitigating Bias and Ensuring Fairness:
    1. Investigation into biases and fairness issues in gender recognition algorithms.
    2. Development of methods to mitigate bias and ensure fair representation across different genders and demographic groups.

7.Privacy and Ethical Considerations:

1. Addressing privacy concerns related to the collection and storage of voice data for gender recognition.
2. Ensuring data protection and compliance with relevant privacy regulations.
3. Incorporating ethical guidelines and responsible practices in the development and deployment of gender recognition systems.

8.Real-World Applications:

1. Analysis of potential applications of gender recognition using voice in various domains, such as healthcare, customer service, security, and entertainment.
2. Exploration of use cases in voice assistants, call centers, gender-based marketing, and personalized user experiences.
3. Discussion on the potential impact of gender recognition technologies on social dynamics and gender equality.

9.Collaborative Research and Data Sharing:

Encouraging collaboration among researchers, industry professionals, and policymakers to advance gender recognition technologies.

CONCLUSION

In conclusion, gender recognition using voice is an evolving field with significant potential and numerous applications. Through this project, we have explored the current state of gender recognition technologies, methodologies, challenges, and future prospects.
We have observed that while existing gender recognition systems show promising results, there is ample room for improvement and innovation. Advancements in acoustic feature extraction techniques offer the possibility of capturing more nuanced voice characteristics, leading to enhanced accuracy in gender recognition.

Looking ahead, the future applications of gender recognition using voice are vast and varied. It can revolutionize industries such as healthcare, customer service, security, and entertainment, enabling personalized user experiences and gender-based marketing strategies. However, it is important to consider the potential societal impact and ensure that these technologies are developed and deployed responsibly

In conclusion, gender recognition using voice has immense potential to shape the future of voice-enabled technologies and services. With ongoing advancements in acoustic feature extraction, deep learning approaches, real-time implementation, fairness considerations, privacy and ethical practices, and collaborative research efforts, we can expect significant improvements in accuracy, fairness, and societal impact. By embracing these future prospects, we can create more inclusive and equitable voice-based systems and contribute to a more diverse and technologically advanced future.

Certificate

This is to certify that Mr. Madhava Reddy of Lovely Professional University, registration number :-12101758, has successfully completed a project on Gender Recognization Using Voice using *Machine Learning with Python* under the guidance of Prof. Arnab Chakraborty.

Prof. Arnab Chakraborty Globsyn

Finishing School

Certificate

This is to certify that Mr. Mallikarjuna Reddy of Lovely Professional University, registration number :-12102343, has successfully completed a project on Gender Recognization Using Voice using *Machine Learning with Python* under the guidance of Prof. Arnab Chakraborty.

Prof. Arnab Chakraborty Globsyn

Finishing School