VOICE-BASED GENDER CLASSIFICATION USING ML & DL

Done By: S. Afreen 21BEC1017

AIM:

This report investigates the gender classification of voice signals using a variety of deep learning (DL) and machine learning (ML) models. The research includes an advanced deep learning model called Artificial Neural Network (ANN) in addition to three main machine learning models: Decision Tree, Random Forest, and Logistic Regression.

This work attempts to assess these models' accuracy in identifying gender from voice data through extensive testing and analysis.

OBJECTIVE:

This report aims to assess how effectively voice signals can be used for gender classification using Machine Learning (ML) and Deep Learning (DL) methods, such as Decision Tree, Random Forest, Logistic Regression, and Artificial Neural Network (ANN).

The goal is to offer a thorough understanding of how these models perform relative to each other through meticulous experimentation and analysis. These findings are essential for informed decision-making in various fields that employ gender classification applications.

By examining the precision and effectiveness of each algorithm, this study seeks to provide valuable insights into voice-based gender classification, thereby fostering technological advancements and real-world applications.

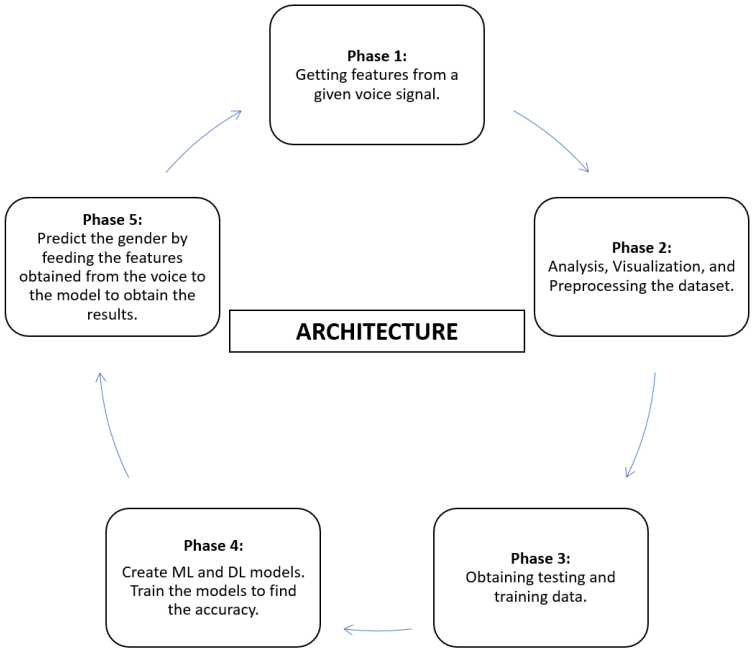
INTRODUCTION:

Voice signal gender classification is important in many fields, including speech recognition, healthcare, security, and human-computer interaction. In addition to enabling personalized experiences, the ability to reliably identify a person's gender from their voice is essential for applications like voice authentication and gender-specific healthcare interventions. More sophisticated computational techniques—such as Machine Learning (ML) and Deep Learning (DL)—are being developed, and this has led to efforts to create gender classification techniques that are more accurate.

The goal of this project is to assess the effectiveness of various computer models in classifying gender based on speech signals. The study specifically looks at four main models: Random Forest, Decision Tree, Logistic Regression, and Artificial Neural Network (ANN). Every model has unique benefits and drawbacks, so a careful comparison is necessary to determine which is more appropriate for practical use.

This project has two main goals: first, to evaluate how well these computer models perform in correctly identifying gender from voice signals; second, to learn more about the factors that affect each model's efficacy. By achieving these goals, the aim is to advance the development of gender classification systems capable of addressing diverse needs across various domains. Consequently, this report delves into a comprehensive analysis of model performance, aiming to inform the refinement of gender classification methodologies for broader societal benefit.

DESIGN ARCHITECTURE:



Phase 1:

The process begins with loading the audio file utilizing the librosa library, which provides the audio time series ('y') and the corresponding sampling rate ('sr'). Subsequently, various features are computed from the audio signal, including statistical measures and spectral characteristics.

The features extracted encompass a range of statistical descriptors such as mean, standard deviation, median, quartiles, interquartile range (IQR), skewness, and kurtosis. Additionally, spectral features like spectral centroid, spectral flatness measures, mode, and pitch-related attributes are computed. These features capture essential characteristics of the voice signal, enabling subsequent analysis and classification tasks.

Upon computation, the extracted features are stored in a dictionary format for further processing. Each feature is associated with its corresponding value, facilitating easy retrieval and manipulation for subsequent analysis. Finally, the extracted features are printed to the console for inspection and verification.

Overall, Phase 1 lays the foundation for subsequent stages of the project by extracting informative features from the voice signal, which are pivotal for subsequent model training and gender classification tasks.

Phase 2:

The dataset comprises 2292 instances and 20 features related to voice signals, encompassing statistical measures, spectral characteristics, and fundamental frequency measurements, facilitating gender classification analysis and modeling. This phase involves the analysis, visualization, and preprocessing of the dataset containing 20 features related to voice signals. These features include statistical measures such as mean frequency, standard deviation of frequency, median frequency, quartiles, interquantile range, skewness, kurtosis, spectral entropy, spectral flatness, mode frequency, frequency centroid, peak frequency, and various fundamental frequency measurements.

The process begins with importing the dataset using the pandas library and examining its structure and contents. Descriptive statistics such as mean, standard deviation, and quartiles are computed for each feature, providing insights into their distributions and characteristics.

Visualizations such as histograms, scatter plots, box plots, and kernel density estimations are employed to explore the distribution and relationships between different features, as well as to identify any potential outliers or anomalies. Additionally, preprocessing steps such as outlier removal using the interquartile range (IQR) method and label encoding of categorical variables are performed to ensure data quality and compatibility for subsequent analysis.

Further analysis includes computing correlation coefficients between features and the target variable, as well as visualizing the correlations using heatmaps and bar plots. This aids in identifying features that are strongly correlated with the target variable, potentially influencing gender classification outcomes.

Finally, the dataset is split into independent variables (features) and dependent variables (target labels), preparing it for subsequent modeling and classification tasks. This phase lays the groundwork for data understanding and preparation, essential for building effective gender classification models.

Phase 3:

This phase involves splitting the dataset into training and testing subsets, facilitating the development and evaluation of predictive models for gender classification.

Utilizing the train_test_split function from the sklearn.model_selection module, the dataset is divided into training and testing data. The training data, constituting 80% of the original dataset, is utilized to train the model and enable it to learn patterns from the data. Conversely, the testing data, comprising 20% of the dataset, serves as unseen data to evaluate the model's performance and generalization ability on new, real-world data.

The resultant subsets include:

- x_train: Training dataset containing independent variables, used for model training.

- x_test: Testing dataset containing independent variables, employed for evaluating model performance.

- y_train: Corresponding labels for the training data, facilitating supervised learning.

- y_test: Labels for the testing data, utilized for assessing model accuracy and performance metrics.

This division ensures that the model is trained on enough data while also providing a separate set of data for unbiased evaluation. Additionally, it enables the calculation of accuracy and other performance metrics on unseen data, aiding in the assessment of model effectiveness.

Phase 4:

This phase involves creating Machine Learning (ML) and Deep Learning (DL) models to perform gender classification based on the provided dataset. These models are trained to achieve optimal accuracy in predicting gender from voice signals.

1. Artificial Neural Network (ANN):

- The TensorFlow library is utilized to implement an Artificial Neural Network model.

- The model architecture comprises multiple layers of densely connected neurons with activation functions such as 'relu' and 'softmax'.

- The model is compiled with appropriate loss function, optimizer, and evaluation metric.

- Training is performed on the training dataset with specified epochs and batch size.

- The trained model is evaluated on the testing dataset to determine accuracy.

- Additionally, the model is applied to predict gender from new data points, and the predicted labels are inverse transformed to obtain the actual gender classifications.

2. Decision Tree Classifier:

- The scikit-learn library is employed to create a Decision Tree Classifier model.

- The model is trained on the training dataset and subsequently used to predict gender labels for the testing dataset.

- Accuracy of the model is calculated using evaluation metrics such as accuracy_score.

- Classification report provides detailed information on model performance including precision, recall, and F1-score.

3. Random Forest Classifier:

- Similar to Decision Tree Classifier, a Random Forest Classifier model is constructed using the RandomForestClassifier class from scikit-learn.

- The model is trained and evaluated on the training and testing datasets, respectively.

- Accuracy and classification report metrics are computed to assess model performance.

4. Logistic Regression:

- Logistic Regression model is created using the LogisticRegression class from scikit-learn.

- The model is trained on the training dataset and used to predict gender labels for the testing dataset.

- Accuracy of the model is calculated, and the classification report provides detailed performance metrics.

Finally, a visualization is generated to compare the accuracies of different models, aiding in the selection of the most effective model for gender classification. Additionally, the trained models are applied to predict gender labels for new data points, demonstrating their practical utility.

Phase 5:

This phase involves utilizing the trained ML and DL models to predict gender based on the features extracted from voice signals. These features are fed into the models, and the models produce predictions indicating the likely gender associated with each set of features. This phase demonstrates the practical application of the trained models for real-time gender classification tasks, enabling the automated determination of gender from voice data.

IMPLEMENTATION CODE:

PHASE 1: Audio extraction

```
import librosa
import numpy as np
from scipy.stats import skew, kurtosis

def extract_audio_features(audio_file_path):
    # Load audio file using librosa, 'y' contains the audio time series, and
'sr' is the sampling rate
    y, sr = librosa.load(audio_file_path, sr=None)

    # Calculate mean of the spectral centroid
    meanfreq = np.mean(librosa.feature.spectral_centroid(y=y, sr=sr))

    # Calculate standard deviation of the spectral centroid
    sd = np.std(librosa.feature.spectral_centroid(y=y, sr=sr))

    # Calculate median of the audio signal
    median = np.median(y)

    # Calculate the 25th percentile (first quartile) of the audio signal
    Q25 = np.percentile(y, 25)

    # Calculate the 75th percentile (third quartile) of the audio signal
    Q75 = np.percentile(y, 75)

    # Calculate the interquartile range (IQR)
    IQR = Q75 - Q25

    # Calculate skewness and kurtosis of the spectral centroid
    spectral_centroid = librosa.feature.spectral_centroid(y=y, sr=sr)[0]
    skewness = skew(spectral_centroid)
    kurt = kurtosis(spectral_centroid)

    # Calculate spectral flatness measures
    sp_ent = np.mean(librosa.feature.spectral_flatness(y=y))
    sfm = np.mean(librosa.feature.spectral_flatness(y=y))

    # Calculate the mean of the spectral centroid
    centroid = np.mean(librosa.feature.spectral_centroid(y=y, sr=sr))

    # Calculate mode of the audio signal using NumPy
    mode_value = float(np.argmax(np.bincount(y.astype(int))))

    # Calculate pitch-related features
    pitches = librosa.core.pitch_tuning(y)
    meanfun = np.mean(pitches)
```

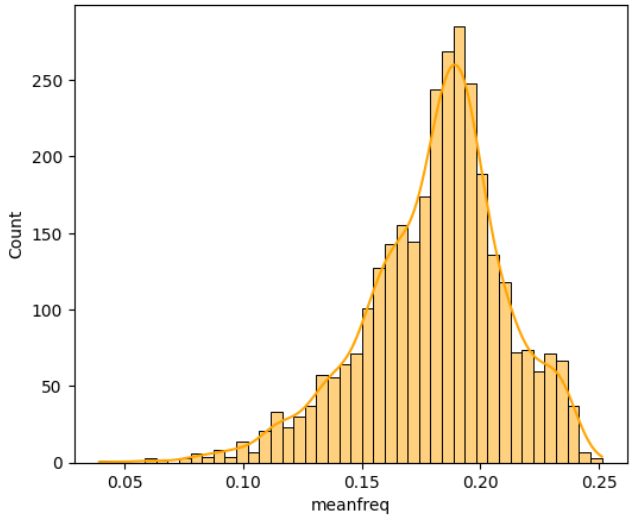
```
    minfun = np.min(pitches)
    maxfun = np.max(pitches)
    meandom = np.mean(librosa.feature.tempogram(y=y, sr=sr))
    mindom = np.min(librosa.feature.tempogram(y=y, sr=sr))
    maxdom = np.max(librosa.feature.tempogram(y=y, sr=sr))
    dfrange = (maxdom - mindom)
    modindx = np.mean(librosa.feature.spectral_rolloff(y=y, sr=sr))

    # Store the extracted features in a dictionary
    return {
        'meanfreq': [meanfreq],
        'sd': [sd],
        'median': [median],
        'Q25': [Q25],
        'Q75': [Q75],
        'IQR': [IQR],
        'skew': [skewness],
        'kurt': [kurt],
        'sp.ent': [sp_ent],
        'sfm': [sfm],
        'mode': [mode_value],
        'centroid': [centroid],
        'meanfun': [meanfun],
        'minfun': [minfun],
        'maxfun': [maxfun],
        'meandom': [meandom],
        'mindom': [mindom],
        'maxdom': [maxdom],
        'dfrange': [dfrange],
        'modindx': [modindx],
    }

# Path to the audio file
audio_file_path = 'C:\Gender Classfication project\Gender Classfication
project\path\girl_1.mp3'

# Extract features from the audio file
features = extract_audio_features(audio_file_path)

# Print the extracted features
print(features)
```

PHASE 2 & 3: ANALYSIS, VISUALIZATION, PREPROCESSING, TRAINING AND TESTING DATA

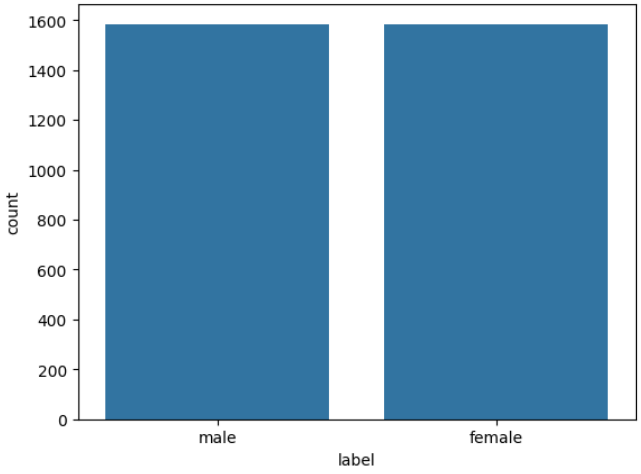
```
df = pd.read_csv("voice.csv") #importing the csv file dataset using pandas library
df.head(10)
df.shape #shape command shows the rows and columns of the dataset
```

(3168, 21)

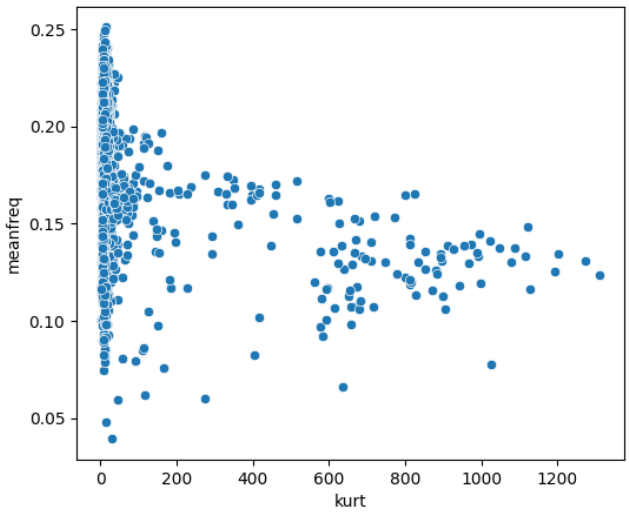
```
df.info() #info command gives description about the type of data (64 bits float data or object type data or any other type) present in each column of the dataset
df.describe() #describe command gives a numerical analysis of all the numerical values for each column
fig=plt.figure(figsize=(6,5))
sns.histplot(data=df,x='meanfreq',color='orange',kde=True)
plt.show()
```



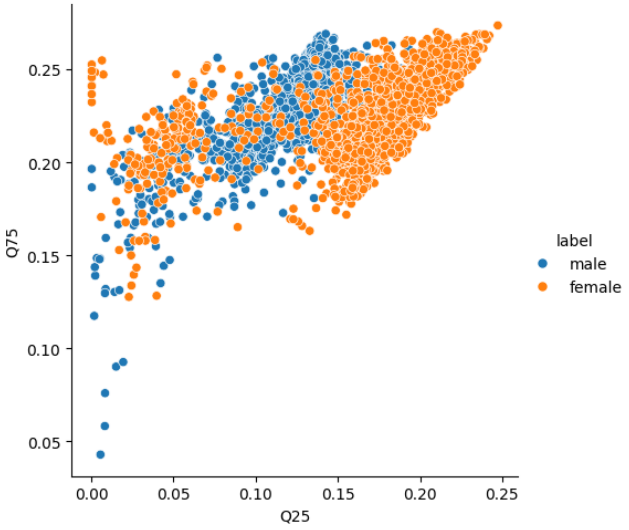

```
sns.countplot(x="label",data=df)
```



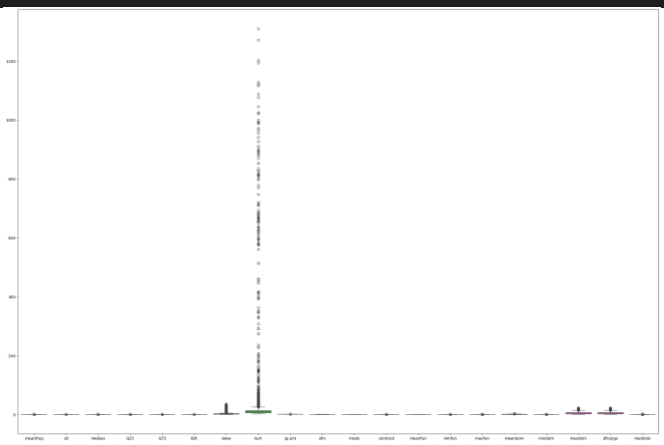

```
fig=plt.figure(figsize=(6,5))
sns.scatterplot(data=df,x='kurt',y='meanfreq')
plt.show()
```




```
sns.relplot(x="Q25", y="Q75",data=df,hue="label")
```

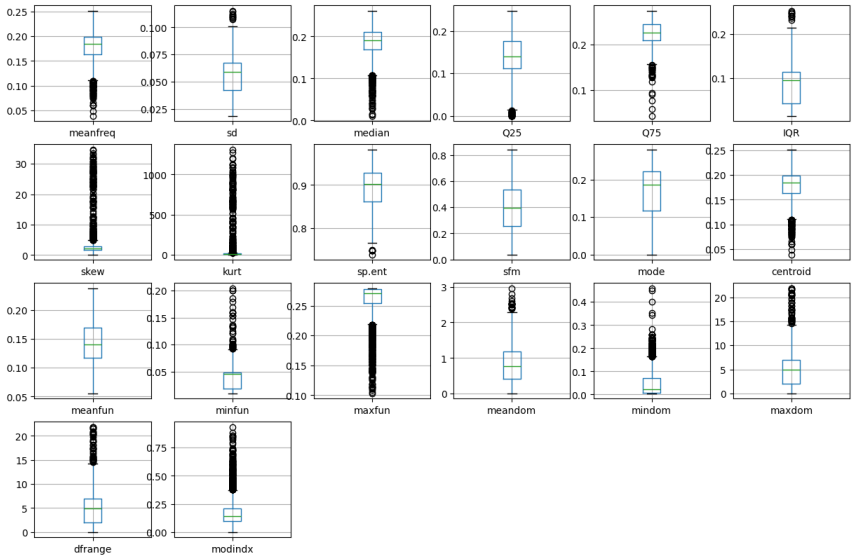



```
fig=plt.figure(figsize=(30,20))
sns.boxplot(df)
plt.show()
```

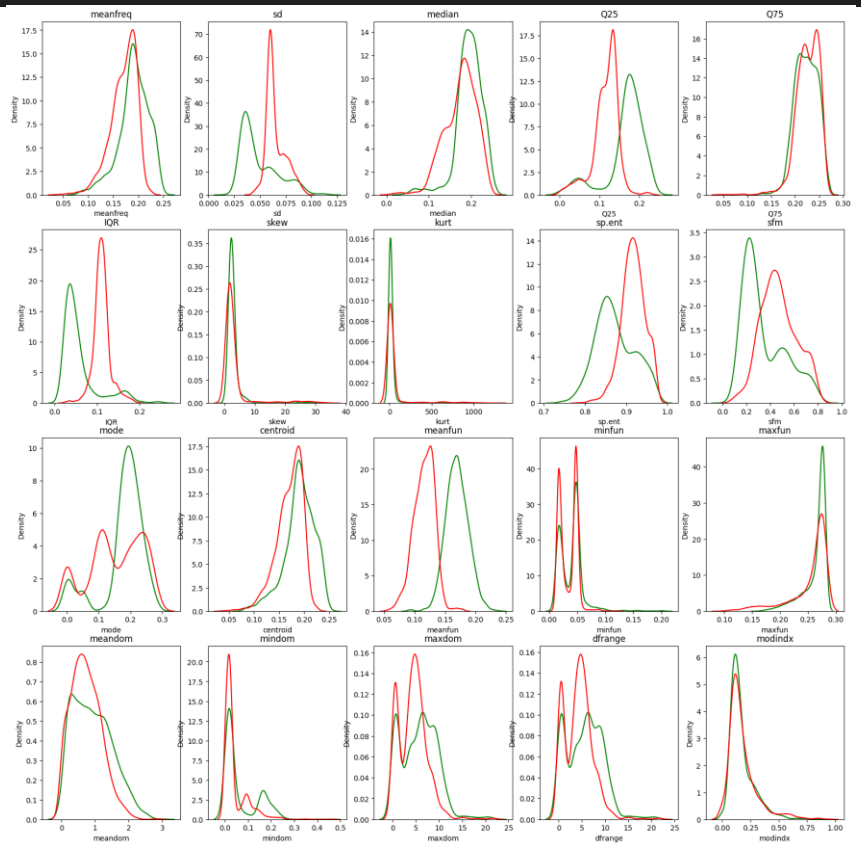


```
plt.figure(figsize=(15,15))
i=1
for col in df.iloc[:,:-1]:
```

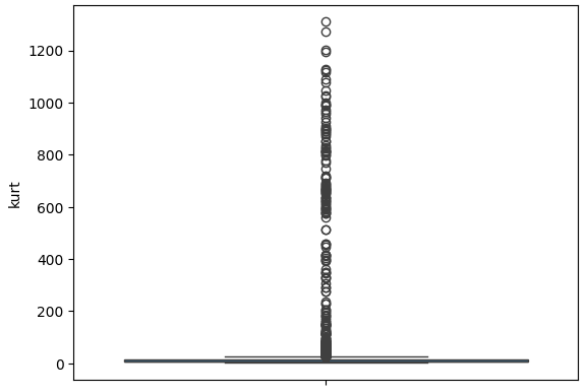
```
    plt.subplot(6,6,i)
    df[[col]].boxplot()
    i+=1
```




```
plt.subplots(4,5,figsize=(20,20))
for k in range(1,21):
    plt.subplot(4,5,k)
    plt.title(df.columns[k-1])
      sns.kdeplot(df.loc[df['label']  ==  'female',  df.columns[k-1]],  color=
'green', label='F')
    sns.kdeplot(df.loc[df['label'] == 'male', df.columns[k-1]], color= 'red',
label='M')
```



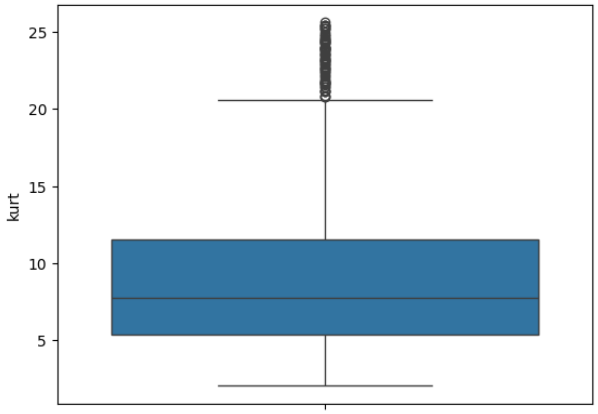
```
sns.boxplot(df['kurt'])
```



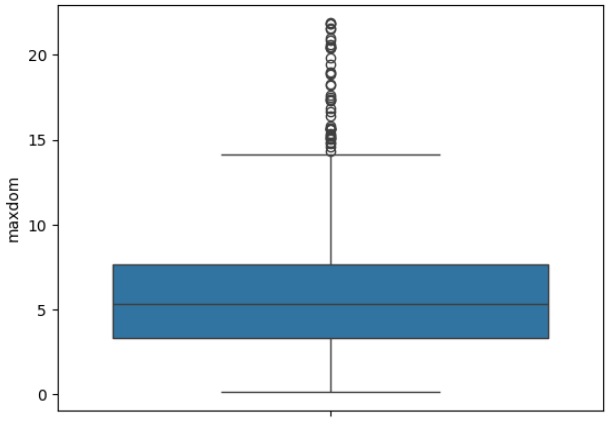

```
q1 = df['kurt'].quantile(0.25)
q3 = df['kurt'].quantile(0.75)
IQR=q3-q1
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
print(upper_limit)
print(lower_limit)
```

```
25.61794301571637
-6.299490840118888
```

```
df = df[df['kurt']<upper_limit]
sns.boxplot(df['kurt'])
```



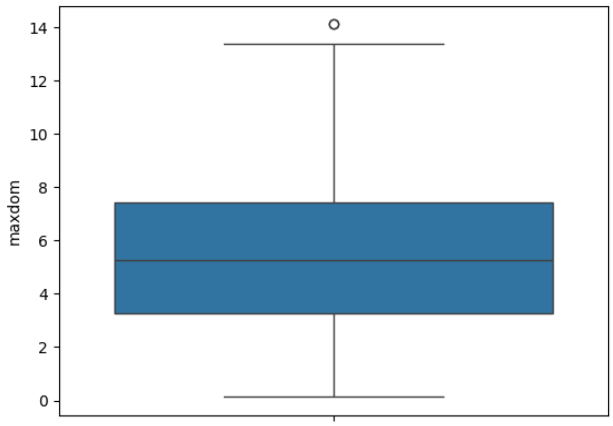

```
sns.boxplot(df['maxdom'])
```



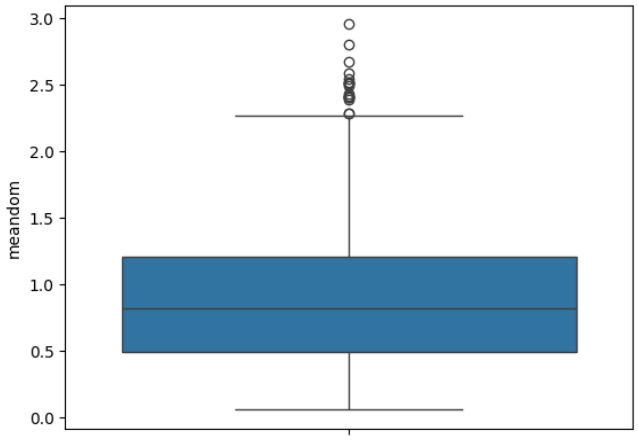
```
q1 = df['maxdom'].quantile(0.25)
q3 = df['maxdom'].quantile(0.75)
IQR=q3-q1
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
print(upper_limit)
print(lower_limit)
```

```
14.1708984375
-3.1806640625
```

```
df = df[df['maxdom']<upper_limit]
sns.boxplot(df['maxdom'])
```



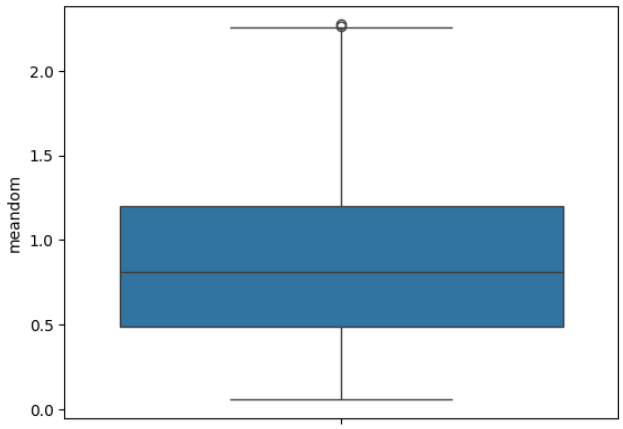

```
sns.boxplot(df['meandom'])
```



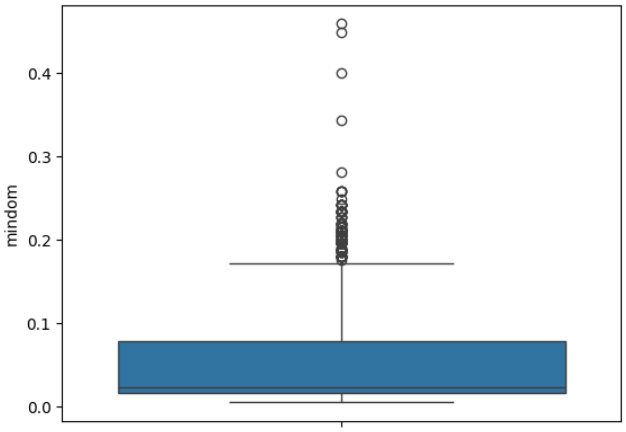

```
q1 = df['meandom'].quantile(0.25)
q3 = df['meandom'].quantile(0.75)
IQR=q3-q1
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
print(upper_limit)
print(lower_limit)
```

```
2.279893755305599
-0.5820809900254647
```

```
df = df[df['meandom']<upper_limit]
sns.boxplot(df['meandom'])
```



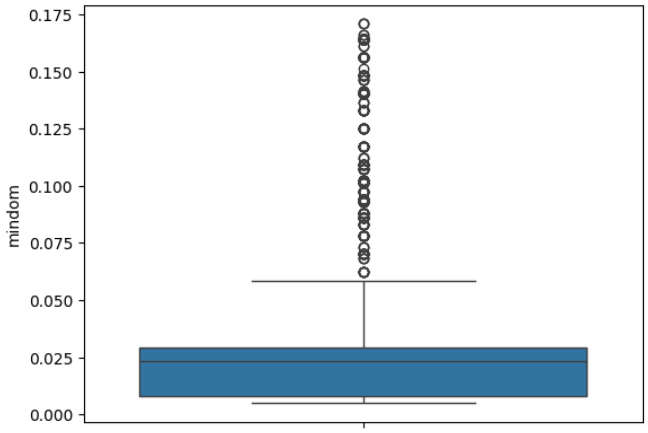

```
sns.boxplot(df['mindom'])
```



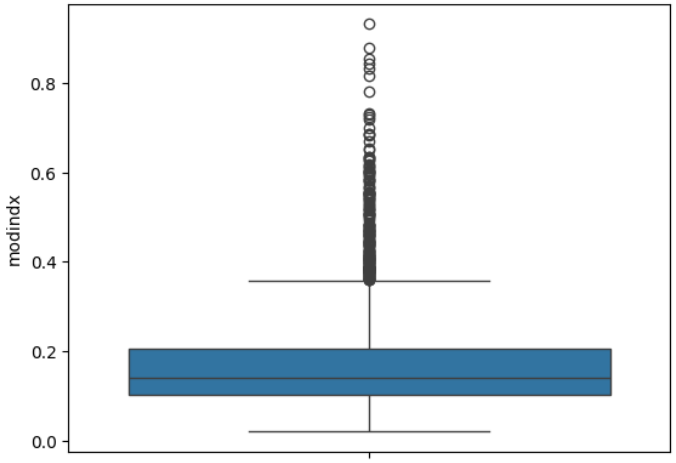

```
q1 = df['mindom'].quantile(0.25)
q3 = df['mindom'].quantile(0.75)
IQR=q3-q1
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
print(upper_limit)
print(lower_limit)
```

0.171875
-0.078125

```
df = df[df['mindom']<upper_limit]
sns.boxplot(df['mindom'])
```



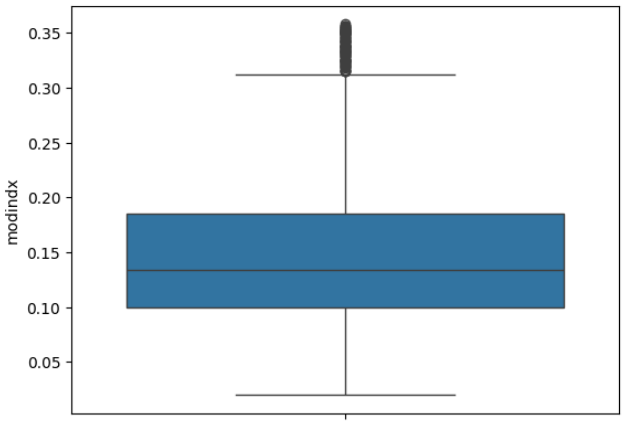
```
sns.boxplot(df['modindx'])
```



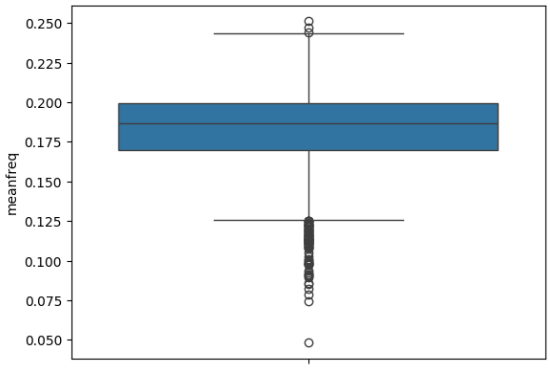

```
q1 = df['modindx'].quantile(0.25)
q3 = df['modindx'].quantile(0.75)
IQR=q3-q1
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
print(upper_limit)
print(lower_limit)
```

```
0.358318698652736
-0.051447757362466026
```

```
df = df[df['modindx']<upper_limit]
sns.boxplot(df['modindx'])
```



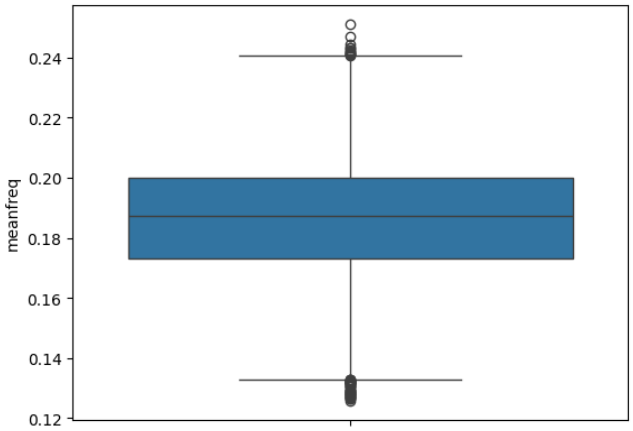

```
sns.boxplot(df['meanfreq'])
```



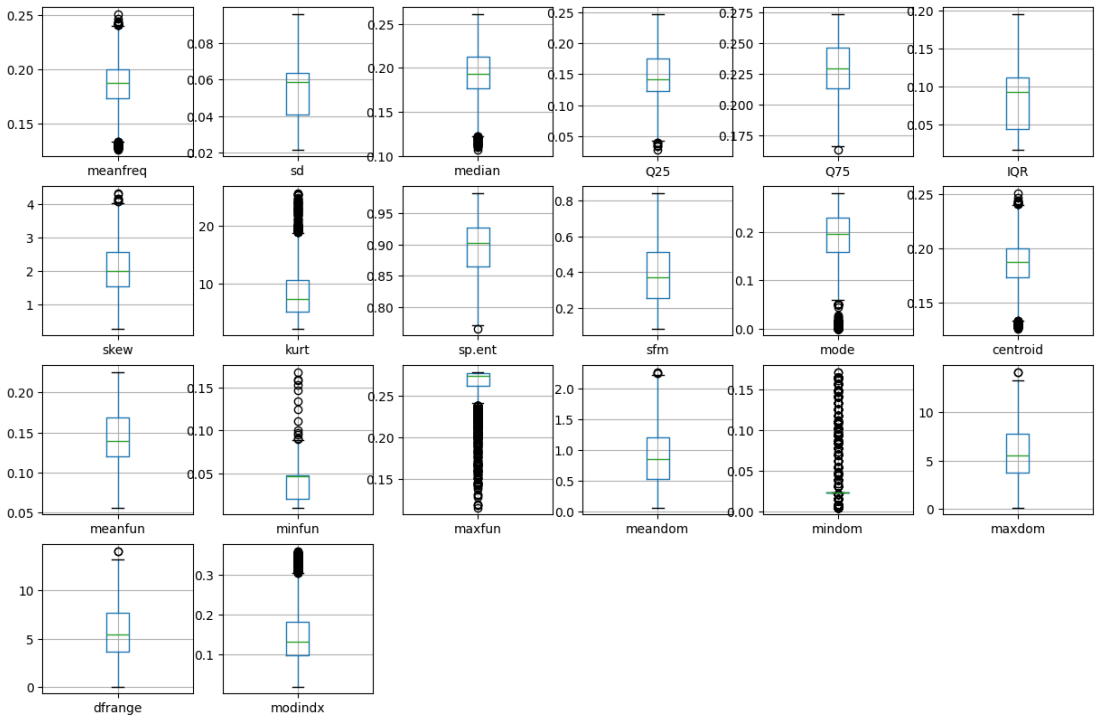
```
q1 = df['meanfreq'].quantile(0.25)
q3 = df['meanfreq'].quantile(0.75)
IQR=q3-q1
upper_limit=q3+1.5*IQR
lower_limit=q1-1.5*IQR
print(upper_limit)
print(lower_limit)
```

0.24411877286509415
0.12544410072336515

```
df = df[df['meanfreq']>lower_limit]
sns.boxplot(df['meanfreq'])
```




```
plt.figure(figsize=(15,15))
i=1
for col in df.iloc[:,:-1]:
    plt.subplot(6,6,i)
    df[[col]].boxplot()
    i+=1
```



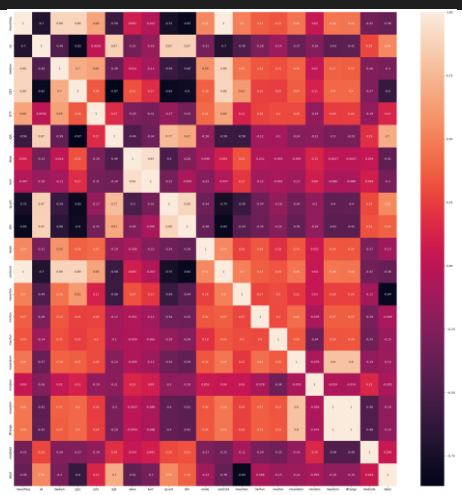
```
df.iloc[:,:-1].skew()
```

```
meanfreq   -0.115514
sd          0.023828
median     -0.451013
Q25        -0.131292
Q75        -0.236446
IQR         0.064380
skew        0.451095
kurt        1.291360
sp.ent     -0.400872
sfm         0.475558
mode       -1.054007
centroid   -0.115514
meanfun     0.107445
minfun      0.874763
maxfun     -2.512841
meandom     0.430349
mindom      1.853433
maxdom      0.028377
dfrange     0.032113
modindx     1.047374
dtype: float64
```

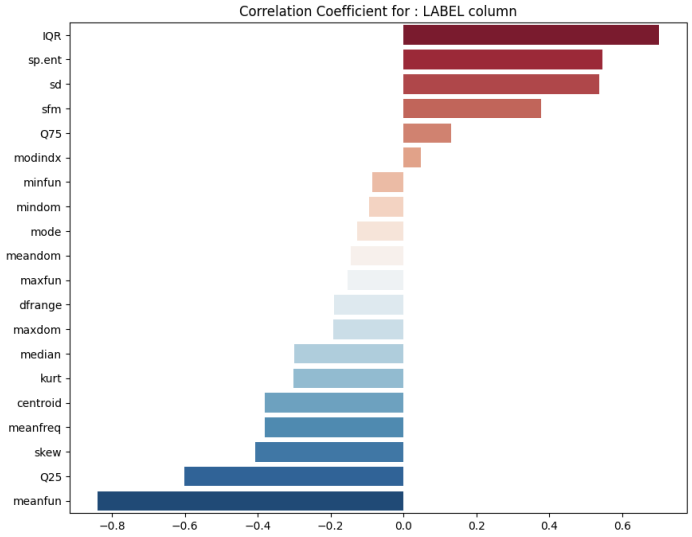
```
#LABEL ENCODING THE pdclass COLUMN
#Label encoding enables to give label based values for the columns of object datatype
from sklearn.preprocessing import LabelEncoder
lb_make = LabelEncoder()
df['label'] = lb_make.fit_transform(df['label'])
df.sample(3)
df.label.value_counts()
```

```
label
1    1192
0    1100
Name: count, dtype: int64
```

```
df.corr()
plt.figure(figsize=(30,30))
sns.heatmap(df.corr(),annot=True)
plt.show()
```



```
def target_coeff(dataframe,target):
    data = dataframe.corr()[target].sort_values(ascending=False)
    indices = data.index
    labels = []
    corr = []
    for i in range(1, len(indices)):
        labels.append(indices[i])
        corr.append(data[i])
    plt.figure(figsize=(10,8),dpi=100)
    sns.barplot(x=corr, y=labels, palette="RdBu")
    plt.title('Correlation Coefficient for : {} column'.format(target.upper()))
    plt.show()
target_coeff(df,'label')
```




```
#INDEPENDENT VARIABLES
x=df.iloc[:,0:20] ##0-first column, 21st-last column, excludes the label in the
22nd column
x.head() ##first 5 rows display
x.shape
```

```
(2292, 20)
```

```
#DEPENDENT VARIABLES
y=df.label
y.head()
```

```
3    1
4    1
5    1
6    1
7    1
Name: label, dtype: int32
```

```
y.shape
```

```
(2292,)
```

```
#Training and Testing dataset splitting is done.
#Training data is 80% of the original data which will help to run the model and also prepare the model to work with the testing data.
#Testing data is 20% of the original data which will help the model to work with real-time data in future.
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

```
((1833, 20), (459, 20), (1833,), (459,))
```

```
acc=[]
model=[]
```

PHASE 4: ML & DL MODELS

ANN:

```
import tensorflow as tf
print(tf.__version__)
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Dropout,Activation,Flatten
from tensorflow.keras.optimizers import Adam
from sklearn import metrics
from tensorflow.keras.layers import Input
### No of classes
num_labels=y.shape[0]
ANNmodel = Sequential()

# First layer
ANNmodel.add(Dense(128,  input_shape=(20,),  activation='relu'))   #  Increased number of neurons
ANNmodel.add(Dropout(0.2))  # Reduced dropout rate

# Second layer
ANNmodel.add(Dense(256, activation='relu'))  # Increased number of neurons
ANNmodel.add(Dropout(0.3))  # Adjusted dropout rate

# Third layer
ANNmodel.add(Dense(128, activation='relu'))  # Reduced number of neurons
ANNmodel.add(Dropout(0.2))  # Reduced dropout rate

# Final layer
ANNmodel.add(Dense(num_labels, activation='softmax'))

ANNmodel.summary()
```

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 128)	2,688
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 256)	33,024
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 128)	32,896
dropout_2 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 2292)	295,668

Total params: 364,276 (1.39 MB)

Trainable params: 364,276 (1.39 MB)

```
ANNmodel.compile(loss='categorical_crossentropy',metrics=['accuracy'],optimize
r='adam')
ANNmodel.compile(optimizer='adam',      loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
from datetime import datetime

num_epochs = 100  # Increased number of epochs
num_batch_size = 64  # Adjusted batch size

checkpointer                                                              =
ModelCheckpoint(filepath='saved_models/audio_classification.keras',
                               verbose=1, save_best_only=True)

early_stopping = EarlyStopping(monitor='val_loss', patience=20, verbose=1)  #
Early stopping callback

start = datetime.now()

ANNmodel.fit(x_train, y_train,
              batch_size=num_batch_size,
              epochs=num_epochs,
              validation_split=0.2,
             callbacks=[checkpointer, early_stopping],  # Added early stopping
callback
              verbose=1)
duration = datetime.now() - start
print("Training completed in time: ", duration)
```

```
Epoch 1/100
19/23 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.2482 - loss: 6.2533
Epoch 1: val_loss improved from inf to 0.78784, saving model to
saved_models/audio_classification.keras
23/23 ━━━━━━━━━━━━━━━━━━━━ 2s 18ms/step - accuracy: 0.2789 - loss: 5.7640 -
val_accuracy: 0.5041 - val_loss: 0.7878
Epoch 2/100
13/23 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.4880 - loss: 0.9244
Epoch 2: val_loss improved from 0.78784 to 0.70603, saving model to
saved_models/audio_classification.keras
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.4963 - loss: 0.9056 -
val_accuracy: 0.5232 - val_loss: 0.7060
Epoch 3/100
12/23 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.5068 - loss: 0.7933
Epoch 3: val_loss improved from 0.70603 to 0.67437, saving model to
saved_models/audio_classification.keras
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.5178 - loss: 0.7809 -
val_accuracy: 0.5858 - val_loss: 0.6744
Epoch 4/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.5053 - loss: 0.7488
Epoch 4: val_loss did not improve from 0.67437
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.5058 - loss: 0.7482 -
val_accuracy: 0.6158 - val_loss: 0.6749
Epoch 5/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.5968 - loss: 0.6761
Epoch 5: val_loss improved from 0.67437 to 0.66529, saving model to
saved_models/audio_classification.keras
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.5954 - loss: 0.6769 -
val_accuracy: 0.6294 - val_loss: 0.6653
Epoch 6/100
12/23 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.5900 - loss: 0.6932
Epoch 6: val_loss improved from 0.66529 to 0.66050, saving model to
saved_models/audio_classification.keras
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.5860 - loss: 0.6911 -
val_accuracy: 0.6213 - val_loss: 0.6605
Epoch 7/100
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.6014 - loss: 0.6716
...
21/23 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9553 - loss: 0.1275
Epoch 100: val_loss did not improve from 0.09234
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9552 - loss: 0.1271 -
val_accuracy: 0.9074 - val_loss: 0.2192
Training completed in time:  0:00:21.338195
```

```
#validation accuracy
test_accuracy=ANNmodel.evaluate(x_test,y_test,verbose=0)
print(test_accuracy[1])
```

0.9172113537788391

```
ANNmodel.predict(x_test)
y_pred=ANNmodel.predict(x_test)
y_pred
```

DECISION TREE:

```
DT_model=DecisionTreeClassifier()
DT_model.fit(x_train,y_train)
y_pred=DT_model.predict(x_test)
y_pred
#the model predicted the result with the input of testing data of independent variables (x_test) since it has been trained previously with the training dataset
y_test #original result of the independent variables of the testing data of dependent variables.
x=accuracy_score(y_test,y_pred)
print(x)
acc.append(x)
model.append('Decision Tree Classifier')
```

```
0.9629629629629629
```

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      0.96      0.96       220
           1       0.96      0.97      0.96       239

    accuracy                           0.96       459
   macro avg       0.96      0.96      0.96       459
weighted avg       0.96      0.96      0.96       459
```

RANDOM FOREST:

```
RF_model = RandomForestClassifier(n_estimators=100,random_state=42)
RF_model.fit(x_train, y_train)
y_pred=RF_model.predict(x_test)
y_pred
#the model predicted the result with the input of testing data of independent variables (x_test) since it has been trained previously with the training dataset
y_test #original result of the independent variables of the testing data of dependent variables.
x=accuracy_score(y_test,y_pred)
print(x)
acc.append(x)
model.append('Random Forest Classifier')
```

```
0.9738562091503268
```

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      0.97      0.97       220
           1       0.97      0.98      0.97       239

    accuracy                           0.97       459
   macro avg       0.97      0.97      0.97       459
weighted avg       0.97      0.97      0.97       459
```

LOGISTIC REGRESSION:

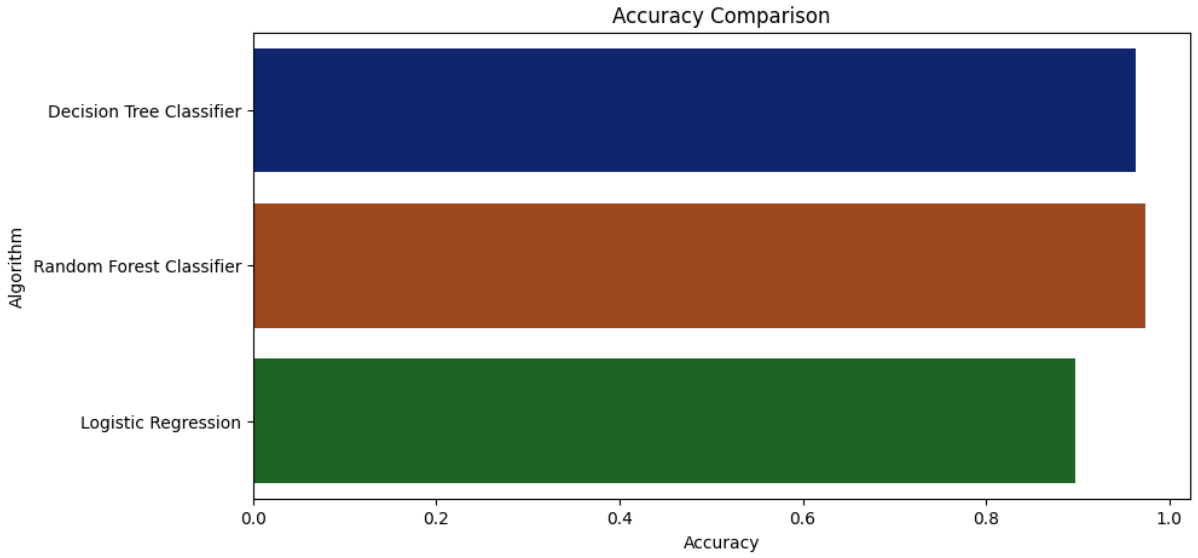
```
LR_model=LogisticRegression()
LR_model.fit(x_train,y_train)
y_pred=LR_model.predict(x_test)
y_pred
y_test
x=accuracy_score(y_test,y_pred)
print(x)
acc.append(x)
model.append('Logistic Regression')
```

```
0.8976034858387799
```

RESULT:

PHASE 5: PREDICTION AND ACCURACY RESULTS

```
plt.figure(figsize=[10,5],dpi = 100)
plt.title('Accuracy Comparison')
plt.xlabel('Accuracy')
plt.ylabel('Algorithm')
sns.barplot(x = acc,y = model,palette='dark')
```



RANDOM FOREST PREDICTION: (for voice signal 1)

```
data_dict = {
    'meanfreq': [2192.3291053431626], 'sd': [1220.3091879537865], 'median': [-
0.0004272461], 'Q25': [-0.01605224609375], 'Q75': [0.008514404296875], 'IQR':
[0.024566650390625],        'skew':        [1.4053504430510502],        'kurt':
[1.3383442870536895], 'sp.ent': [0.056253217], 'sfm': [0.056253217], 'mode':
[0.0], 'centroid': [2192.3291053431626], 'meanfun': [-0.38], 'minfun': [-0.38],
'maxfun':    [-0.38],    'meandom':    [0.10655758482835208],    'mindom':    [-
1.5849987772704835e-16],  'maxdom':  [1.0],  'dfrange':  [1.0000000000000002],
'modindx': [4372.310450819672]
}
data = pd.DataFrame(data_dict)
```

```
prediction=RF_model.predict(data)
print(prediction)
lb_make.inverse_transform(prediction)
```

```
array(['male'], dtype=object)
```

ANN PREDICTION: (for voice signal 2)

```
data_dict = {
      'meanfreq': [2474.127023919006], 'sd': [2194.621448232875], 'median':
[0.0], 'Q25': [-0.04960154742002487], 'Q75': [0.03940417990088463], 'IQR':
[0.0890057273209095], 'skew': [2.6191388679266474], 'kurt':
[7.055743334716519], 'sp.ent': [0.041512117], 'sfm': [0.041512117], 'mode':
[0.0], 'centroid': [2474.127023919006], 'meanfun': [-0.14999999999999997],
'minfun': [-0.14999999999999997], 'maxfun': [-0.14999999999999997], 'meandom':
[0.06314910571901232], 'mindom': [-1.5017708637785386e-16], 'maxdom': [1.0],
'dfrange': [1.0000000000000002], 'modindx': [4159.272927135678]
}
data = pd.DataFrame(data_dict)
prediction=ANNmodel.predict(data)
# Ensure prediction values are integer indices
prediction_indices = np.argmax(prediction, axis=1)

# Inverse transform the integer indices to labels
inverse_labels = lb_make.inverse_transform(prediction_indices)

# Print the inverse transformed labels
print(inverse_labels)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
['female']
```

- **ANN accuracy:** 91.7%
- **Decision Tree accuracy:** 96.3%
- **Random forest accuracy:** 97.4%
- **Logistic Regression accuracy:** 89.7%

Among the ML models, the Random Forest Classifier achieved the highest accuracy of 97.4%, while for DL model, the Artificial Neural Network (ANN) attained an accuracy of 91.7%. Therefore, the Random Forest Classifier is selected for prediction from the ML models, while ANN is utilized for prediction from the DL model.

CONCLUSION:

The effectiveness of several deep learning (DL) and machine learning (ML) models for gender classification from voice signals has been evaluated in this work. Promising findings emerged from the analysis, with the Random Forest Classifier topping the list of machine learning models with an accuracy of 97.4%. With an accuracy of 91.7%, the Artificial Neural Network (ANN) showed competitive performance for DL models.

These results highlight the usefulness of deep learning and machine learning techniques for gender classification tasks, underscoring the significance of choosing suitable models for particular applications. High accuracy and reliability make the Random Forest Classifier a strong option for real-world gender classification scenarios.