Voice data set

**Problem statement:**

The dataset has different attributes of “sound” for males and females

The dataset includes the following columns:

●       meanfreq: Mean frequency of the voice signal

●       sd: Standard deviation of the frequency

●       median: Median frequency

●       Q25: First quantile (25%) of the frequency

●       Q75: Third quantile (75%) of the frequency

●       IQR: Interquartile range

●       skew: Skewness of the frequency distribution

●       kurt: Kurtosis, a measure of the "tailedness" of the frequency distribution

●       sp.ent: Spectral entropy, a measure of signal complexity

●       sfm: Spectral flatness measure

●       centroid: Frequency centroid (mean frequency)

●       meanfun: Mean fundamental frequency measured across acoustic signal

●       minfun: Minimum fundamental frequency measured across acoustic signal

●       maxfun: Maximum fundamental frequency measured across acoustic signal

●       meandom: Mean dominant frequency of the acoustic signal

●       mindom: Minimum dominant frequency of the acoustic signal

●       maxdom: Maximum dominant frequency of the acoustic signal

●       dfrange: Dominant frequency range

●       modindx: Modulation index

●       label: Male or female voice - the target variable

The objective is to build a model which will accurately predict and classify any sound as male, or female based on its attributes.

**Data Analysis Journey**

Step 1 : after loading the data we do the initial inspection to understand the structure of the dataset

Step 2 : then we are checking the count of missing values

Step 3 :#handling categorical data after segregating the categorical and continuous columns

We are listing out the categorical columns and applying encoding to the relevant string columns

Correlation of the dataset

Step 5 : feature scaling we are standardizing the numerical features such that the model can capture the patterns accurately

Step 6 : dividing into train test split to split the data for building the model

**Splitting the Dataset**

• Split the dataset into training and test sets to make predictions using the test data breaking dataset to train and test to 75% train data and 25% test data

* Applying Machine Learning Models

• Applied Logistic Regression model, decision tree model and random forest model

* Comparing Model Accuracy and Tuning

• Cross-Validation Without Hyperparameter Tuning

A DecisionTreeClassifier is initialized, and cross-validation is performed using StratifiedKFold to ensure balanced splits of the dataset.

• Cross-validation scores are printed along with their mean

• Cross-Validation with Hyperparameter Tuning:

• A parameter grid is defined for hyperparameter tuning.

• GridSearchCV is used to perform an exhaustive search over specified parameter values for the DecisionTreeClassifier using cross-validation.

• The best parameters and cross-validation score are printed.

• Cross-validation scores are printed for the best model along with their mean.

• Visualization:

• The decision tree is visualized for the best model obtained from hyperparameter tuning using plot\_tree.

* Evaluate the best model on the test set
* Confusion Matrix with interpretation, precision and recall
* Applied random forest classifier
* Cross-validation scores are printed along with their mean
* Cross validation with hyperparameter
* Apply feature importances with the help of random forest
* Creating a data frame for feature importances
* Sorting by importance
* Plot the feature importance
* Applying SVM (support vector machine) for cross check of model performance
* Runing SVM with default hyperparameter
* Applying SVM model with kernel with linear
* Applying SVM model with kernel with RBF
* Apply SVM with polynomial kernel
* Apply SVM with sigmoid kernel
* Performing K-fold cross validation with different kernels
* Applying regularization parameter C and checking out the accuracy score with kernel as linear

**Conclusion**

* While applying decision tree we got overfitting scenario, so we refitted the decision tree model with customization of parameters
* Considering the accuracy scores of Random Forest (with cross-validation and hyperparameter tuning along with their default scores), we concluded that the model gave the best accuracy score: 0.97
* Best Model Metrics: Random Forest
* Accuracy: 0.97
* Precision: 0.98
* Recall: 0.97
* F1 Score: 0.97