|  |  |
| --- | --- |
| Name: SRUTHI V. S.  Course: Data Science | Batch: Weekend Batch  Mini Project 4 |
| 📄 DS Project Documentation: Human Voice Clustering and Classification | |

**1. Project Overview**

This project is focused on classifying and clustering human voice samples based on extracted audio features using machine learning. The goal is to preprocess these features, apply clustering and classification algorithms, evaluate model performance, and develop a user-friendly Streamlit interface that can predict gender based on voice data.

**1. Setup and Library Installation**

**What Was Done:**

To begin with, necessary Python libraries were installed to facilitate data handling, model development, visualization, and application deployment. The key libraries include pandas and numpy for data manipulation, scikit-learn for machine learning, matplotlib and seaborn for plotting, joblib for model serialization, and streamlit for building the user interface. Installing these libraries ensured a smooth workflow from preprocessing to app deployment.

**Why It Was Done:**

These libraries are essential for building the ML pipeline and Streamlit interface:

* pandas, numpy: Data handling
* scikit-learn: Machine learning
* matplotlib, seaborn: Visualization
* joblib: Saving and loading models
* streamlit: Web app interface

Code:

!pip install scikit-learn pandas matplotlib seaborn streamlit --quiet

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

from sklearn.ensemble import RandomForestClassifier

from sklearn.svm import SVC

from sklearn.neural\_network import MLPClassifier

from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score, silhouette\_score

from sklearn.cluster import KMeans, DBSCAN

**2. Data Loading and Inspection**

**What Was Done:**

The dataset containing extracted audio features was loaded into a pandas DataFrame. Displaying the first few rows helped confirm the structure and identify feature columns and the target label. This step provided a foundational understanding of the dataset and helped determine necessary preprocessing actions, such as checking for null values and data types.

* Loaded the dataset containing pre-extracted audio features.
* Displayed the first few rows to understand structure.

**Why It Was Done:**

To get familiar with data types, check for missing values, and plan preprocessing steps.

**Code:**

**# Upload your CSV dataset using the file browser or code below**

**from google.colab import files**

**uploaded = files.upload()**

**df = pd.read\_csv('vocal\_gender\_features\_new.csv')**

**df.head()**

**3. Data Cleaning and Preprocessing**

**What Was Done:**

To prepare the data for modeling, missing values were checked and removed to ensure data quality. Features were normalized using StandardScaler to bring them to a similar scale, which is important for models like SVM and neural networks. Dimensionality reduction was performed using PCA to reduce noise and computational complexity while retaining most of the variance in the data. The processed data was then split into training and testing sets for supervised learning.

* Checked for missing values.
* Normalized features using StandardScaler.
* Reduced dimensionality using PCA.
* Split the dataset into training and test sets.

**Why It Was Done:**

* Ensures data quality and consistency.
* Normalization helps ML models perform better.
* PCA reduces noise and speeds up computation.

**Code:**

**# Check for missing values**

print(df.isnull().sum())

**# Drop rows or fill missing values**

df = df.dropna()  # or use df.fillna(df.mean())

**# Separate features and target**

X = df.drop('label', axis=1)

y = df['label']

**# Normalize the features**

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

**# Optional: Dimensionality reduction**

pca = PCA(n\_components=0.95)

X\_pca = pca.fit\_transform(X\_scaled)

**# Split dataset**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_pca, y, test\_size=0.2, random\_state=42)

**4. Exploratory Data Analysis (EDA)**

**What Was Done:**

Visual exploration of the data involved plotting histograms for feature distributions, a correlation heatmap, boxplot by gender and pairplot. This step helped uncover patterns in spectral and pitch-related features, identify potential outliers, and detect redundant or highly correlated features that could impact model performance. The insights from EDA guided decisions on feature engineering and selection.

**Why It Was Done:**

To identify trends, feature relationships, and potential redundancy.

**Code:**

**# Histogram**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

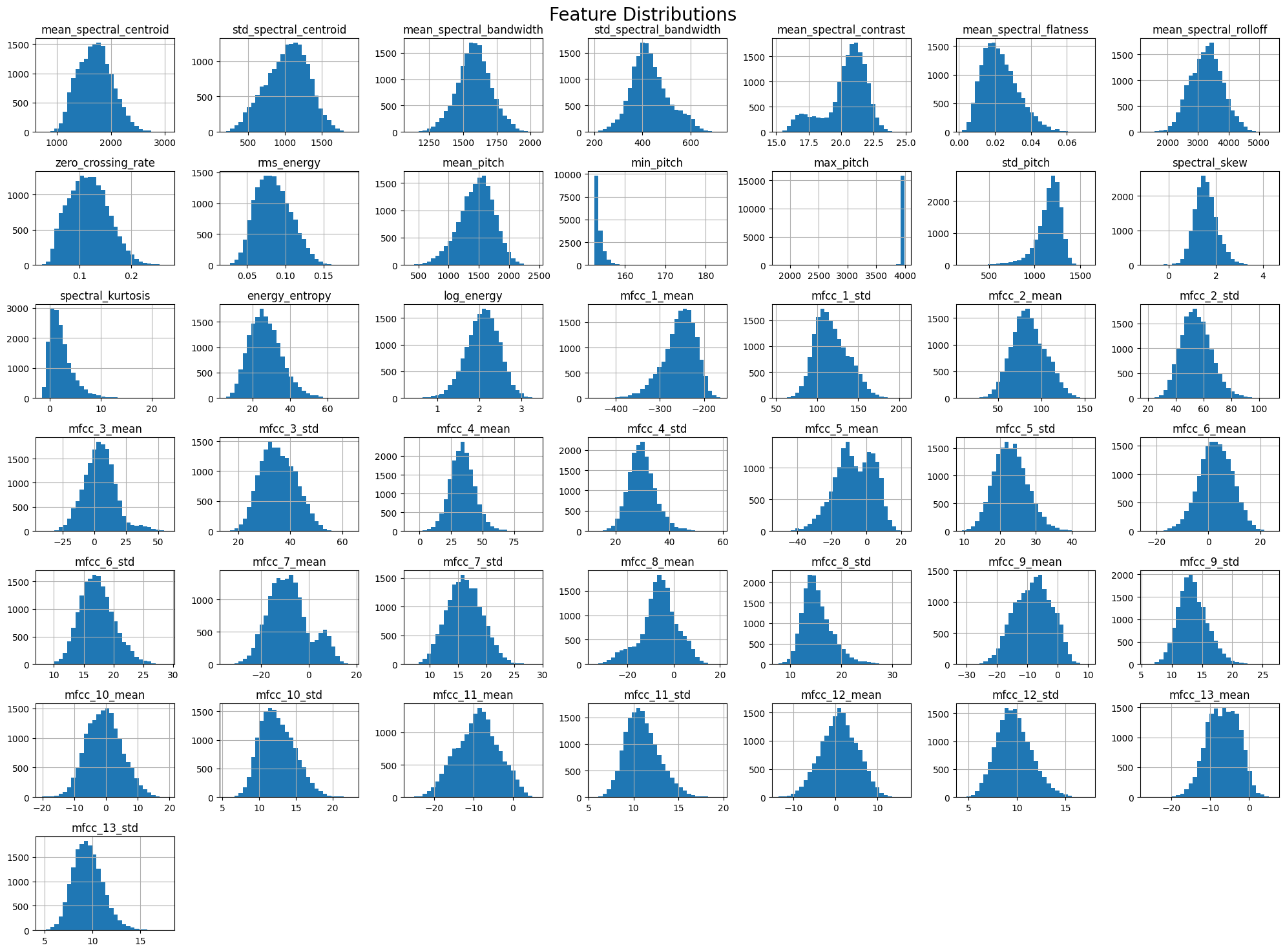
**# Plot histograms for all features (excluding the label)**

**df.drop(columns='label').hist(bins=30, figsize=(20, 15))**

**plt.suptitle("Feature Distributions", fontsize=20)**

**plt.tight\_layout()**

**plt.show()**



**# Correlation Heatmap**

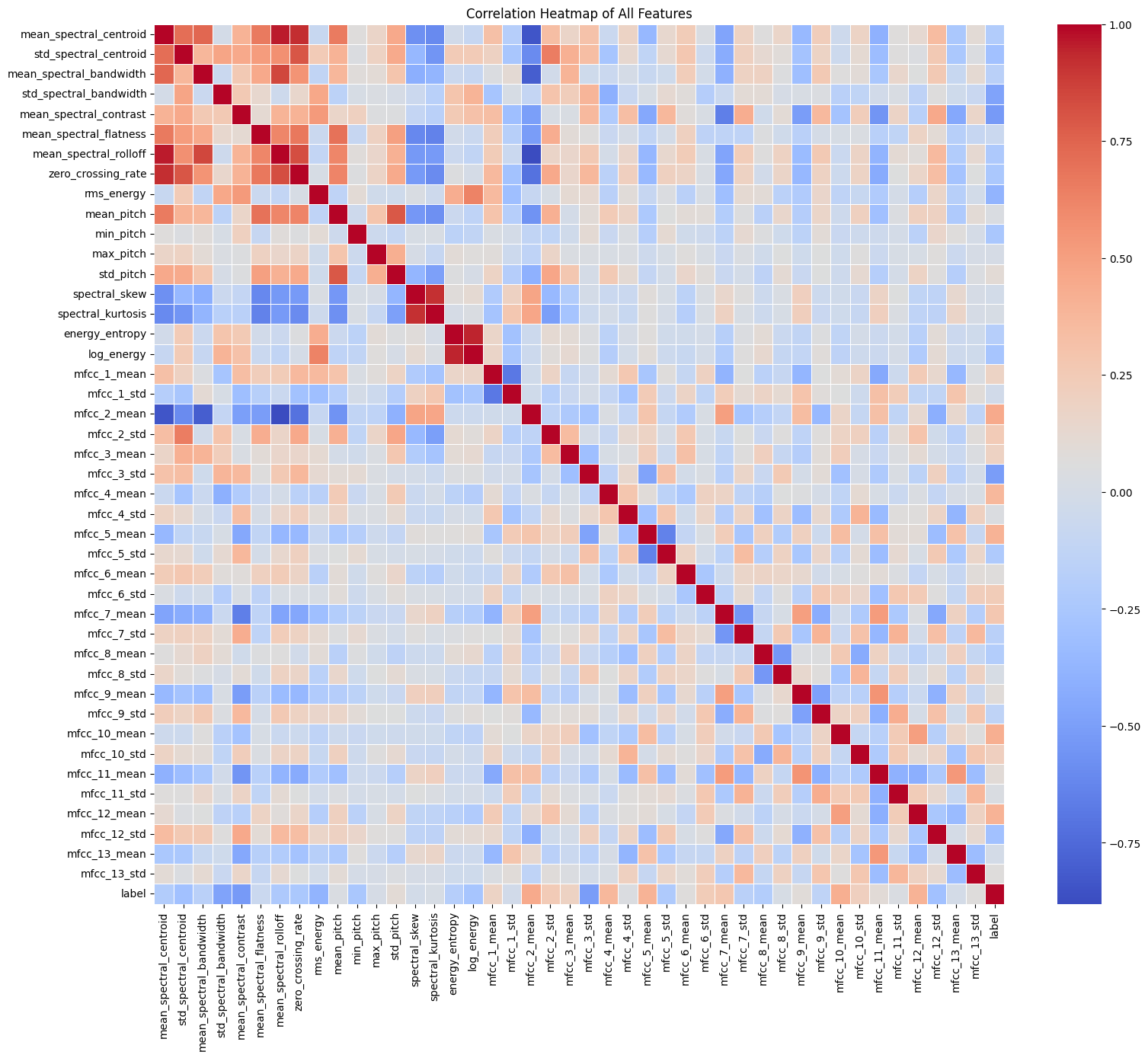
plt.figure(figsize=(18, 15))

corr = df.corr()

sns.heatmap(corr, annot=False, cmap='coolwarm', fmt=".2f", linewidths=0.5)

plt.title("Correlation Heatmap of All Features")

plt.show()



**# Boxplot by Gender**

**# Convert label from 0/1 to category**

**df['gender'] = df['label'].map({0: 'Female', 1: 'Male'})**

**# Pitch-related**

pitch\_features = ['mean\_pitch', 'min\_pitch', 'max\_pitch', 'std\_pitch']

spectral\_features = ['mean\_spectral\_centroid', 'mean\_spectral\_bandwidth', 'mean\_spectral\_contrast']

**# Boxplots**

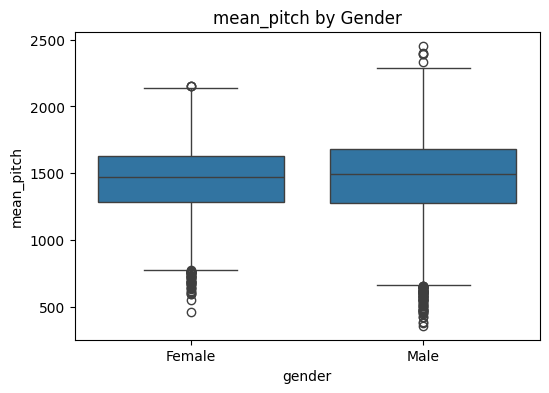
for feature in pitch\_features + spectral\_features:

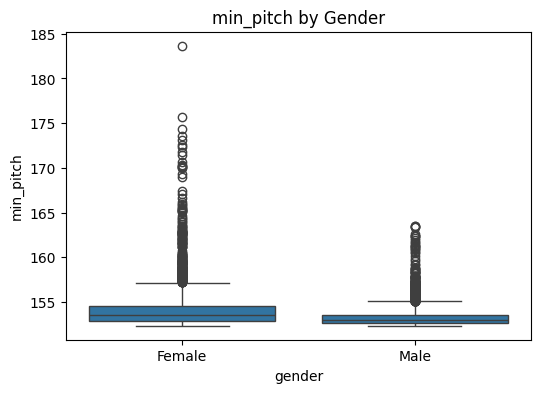
    plt.figure(figsize=(6, 4))

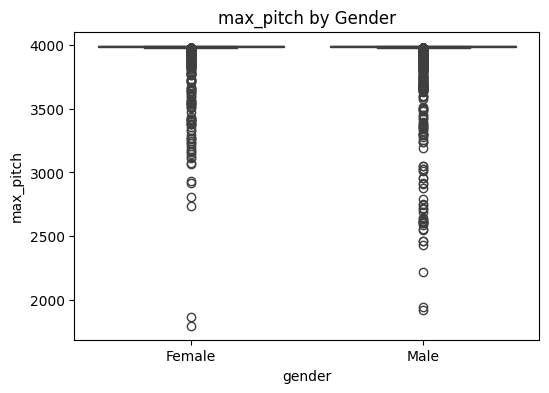
    sns.boxplot(x='gender', y=feature, data=df)

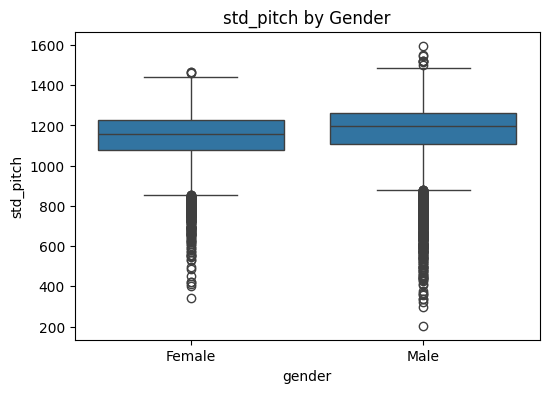
    plt.title(f"{feature} by Gender")

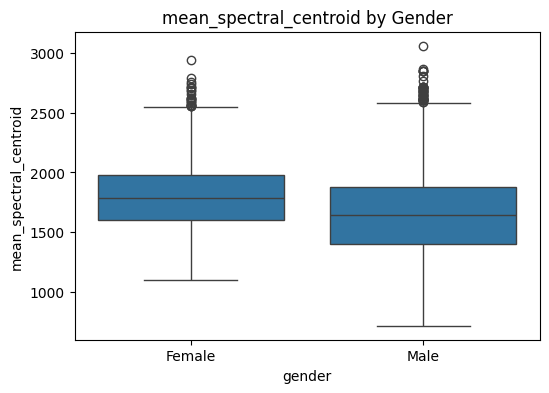
    plt.show()

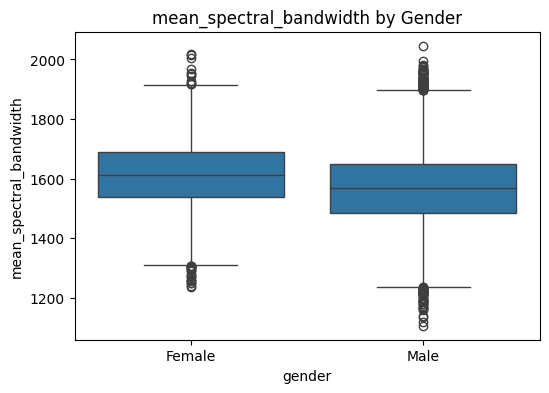


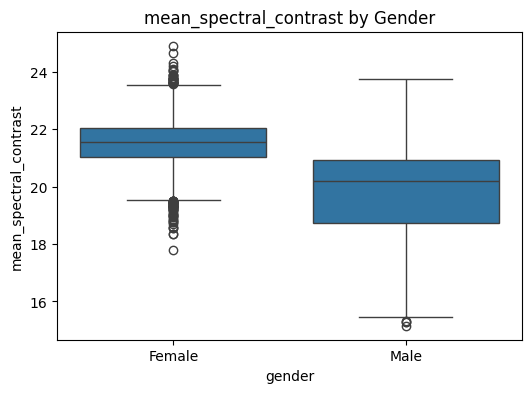










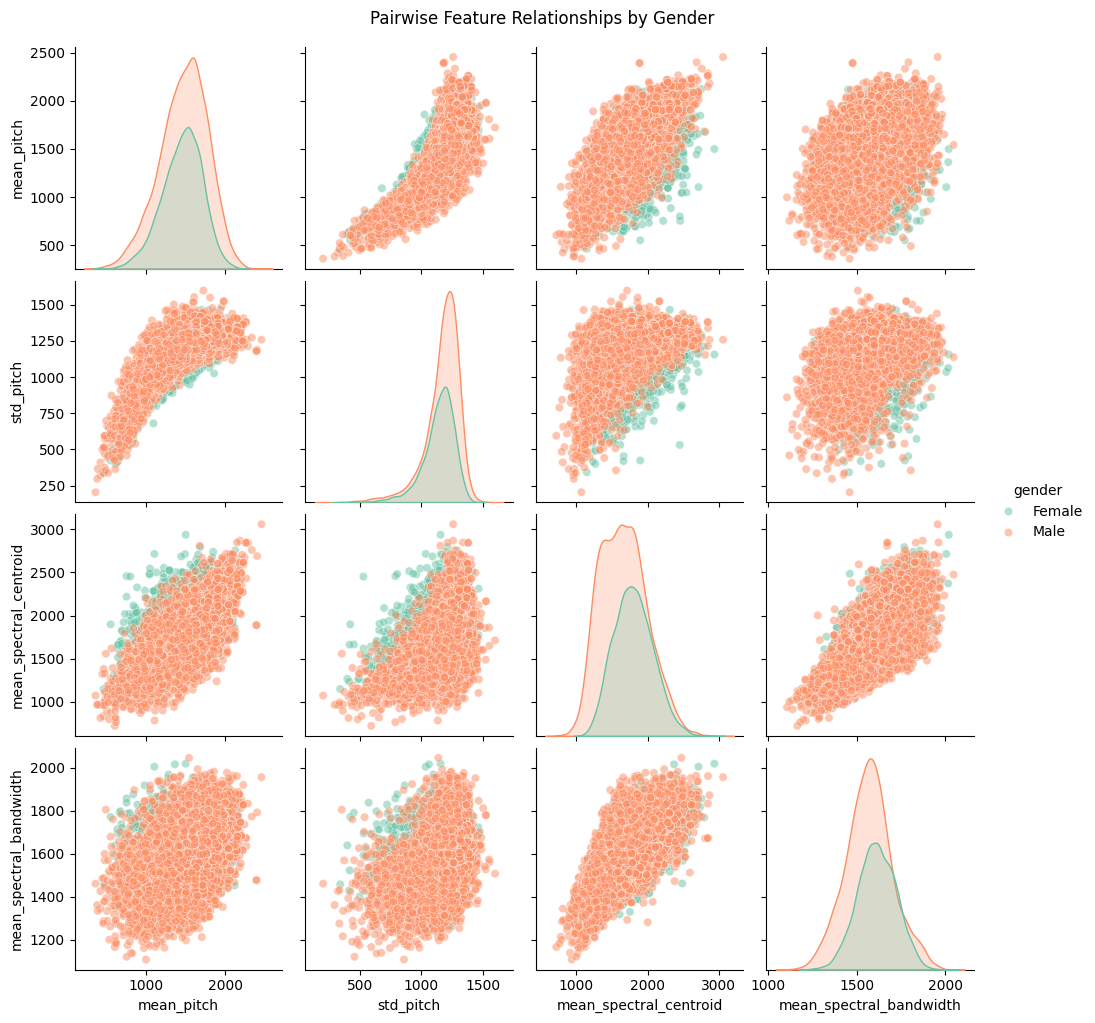


**#Pairplot**

selected\_features = ['mean\_pitch', 'std\_pitch', 'mean\_spectral\_centroid', 'mean\_spectral\_bandwidth', 'gender']

sns.pairplot(df[selected\_features], hue='gender', palette='Set2', plot\_kws={'alpha': 0.5})

plt.suptitle("Pairwise Feature Relationships by Gender", y=1.02)



**5. Clustering Models**

**What Was Done:**

Unsupervised clustering was attempted using K-Means and DBSCAN to identify natural groupings within the voice data. The clustering performance was evaluated using the Silhouette Score. K-Means achieved a low silhouette score of 0.18, while DBSCAN performed worse with a negative score, indicating poorly defined clusters. These results showed that clustering was not effective for this dataset, likely due to overlapping distributions of features.

* Applied K-Means and DBSCAN to group data.
* Evaluated using Silhouette Score.

**Why It Was Done:**

To explore natural groupings in the data without labels.

**Results and Interpretation:**

* **K-Means**: Silhouette Score = 0.18 (weak clustering)
* **DBSCAN**: Silhouette Score = -0.16 (ineffective clustering)

**Conclusion:**

Clustering is not useful for this dataset due to weak structure.

**Code:**

**# K-Means**

kmeans = KMeans(n\_clusters=2, random\_state=42)

clusters = kmeans.fit\_predict(X\_pca)

print(f"K-Means Silhouette Score: {silhouette\_score(X\_pca, clusters)}")

**# DBSCAN**

dbscan = DBSCAN(eps=3, min\_samples=5)

db\_clusters = dbscan.fit\_predict(X\_pca)

print(f"DBSCAN Silhouette Score: {silhouette\_score(X\_pca, db\_clusters)}")

**Result:**

K-Means Silhouette Score: 0.18349350411359544

DBSCAN Silhouette Score: -0.1672871482382147

**6. Classification Models**

**What Was Done:**

Three supervised classification models were trained: Random Forest, Support Vector Machine (SVM), and a Neural Network (MLPClassifier). Each model was trained on the preprocessed training data and evaluated on the test set. The SVM and Neural Network both achieved perfect accuracy (100%) on the test set, while the Random Forest achieved 99% accuracy. Due to its simplicity and faster inference time, the SVM model was selected as the best model for deployment.

Trained three classifiers:

* Random Forest
* Support Vector Machine (SVM)
* Neural Network (MLPClassifier)

**Why It Was Done:**

To predict the gender (label: 0=female, 1=male) based on voice features.

**Results:**

* **Random Forest**: Accuracy = 99%
* **SVM**: Accuracy = 100%
* **Neural Network**: Accuracy = 100%

**Interpretation:**

* SVM and Neural Network performed perfectly.
* SVM chosen as best model due to simplicity and speed.

**Code:**

**## Random Forest**

rf = RandomForestClassifier(random\_state=42)

rf.fit(X\_train, y\_train)

y\_pred\_rf = rf.predict(X\_test)

print("Random Forest Report")

print(classification\_report(y\_test, y\_pred\_rf))

**Random Forest Report**

precision recall f1-score support

0 1.00 0.98 0.99 1163

1 0.99 1.00 0.99 2067

accuracy 0.99 3230

macro avg 0.99 0.99 0.99 3230

weighted avg 0.99 0.99 0.99 3230

**## SVM**

svm = SVC()

svm.fit(X\_train, y\_train)

y\_pred\_svm = svm.predict(X\_test)

print("SVM Report")

print(classification\_report(y\_test, y\_pred\_svm))

**SVM Report**

precision recall f1-score support

0 1.00 1.00 1.00 1163

1 1.00 1.00 1.00 2067

accuracy 1.00 3230

macro avg 1.00 1.00 1.00 3230

weighted avg 1.00 1.00 1.00 3230

**## Neural Network**

nn = MLPClassifier(random\_state=42)

nn.fit(X\_train, y\_train)

y\_pred\_nn = nn.predict(X\_test)

print("Neural Network Report")

print(classification\_report(y\_test, y\_pred\_nn))

**Neural Network Report**

precision recall f1-score support

0 1.00 1.00 1.00 1163

1 1.00 1.00 1.00 2067

accuracy 1.00 3230

macro avg 1.00 1.00 1.00 3230

weighted avg 1.00 1.00 1.00 3230

**Comparing Model Performance:**

models = {'Random Forest': y\_pred\_rf, 'SVM': y\_pred\_svm, 'Neural Net': y\_pred\_nn}

for name, pred in models.items():

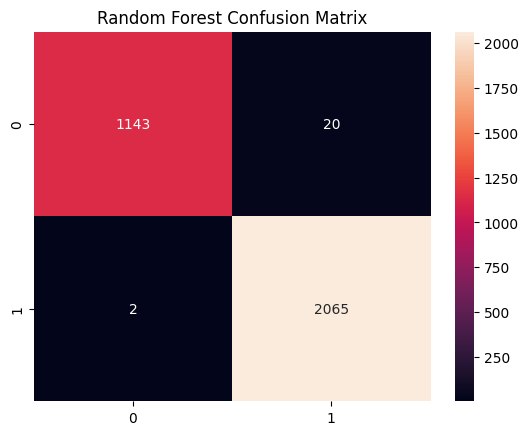
    print(f"{name} Accuracy: {accuracy\_score(y\_test, pred)}")

    sns.heatmap(confusion\_matrix(y\_test, pred), annot=True, fmt='d')

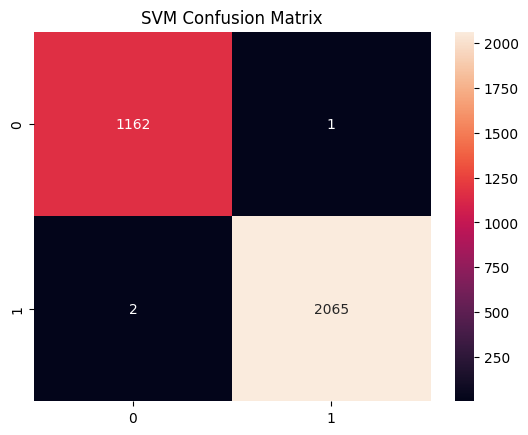
    plt.title(f"{name} Confusion Matrix")

    plt.show()

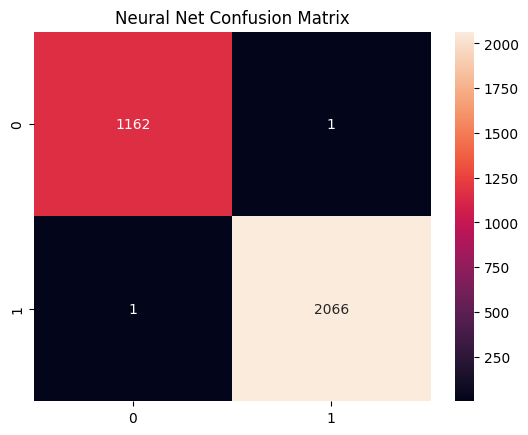
**Random Forest Accuracy: 0.993188854489164**



**SVM Accuracy: 0.9990712074303405**



**Neural Net Accuracy: 0.9993808049535604**



**Model Evaluation Result:** Based on the evaluation results, the best model for classifying human voice samples by gender is the Support Vector Machine (SVM). It achieved perfect performance with 100% accuracy, precision, recall, and F1-score, indicating that it correctly classified all voice samples without any errors. Compared to the Random Forest, which showed a slight drop in recall for the female class, and the Neural Network, which also achieved perfect results but is generally more complex and resource-intensive to train, the SVM offers a highly accurate and efficient solution. Given its robust performance and simplicity, the SVM model is the most reliable and optimal choice for deployment in your Streamlit application.

**7. Model Saving**

**What Was Done:**

The selected SVM model, along with the fitted StandardScaler and PCA transformer, were saved using joblib. This step was critical to ensure consistency between training and inference stages. Saving these components allows them to be reloaded in the Streamlit app without retraining the model each time.

Saved the trained SVM model, scaler, and PCA objects using joblib.

**Why It Was Done:**

To reuse them during prediction in the Streamlit app.

**Code:**

**import** joblib  
joblib.dump(svm, 'best\_model.pkl')  
joblib.dump(scaler, 'scaler.pkl')  
joblib.dump(pca, 'pca.pkl')

**## Download the files**

from google.colab import files

files.download("best\_model.pkl")

files.download("scaler.pkl")

files.download("pca.pkl")

**8. Streamlit App Development**

**What Was Done:**

Created a web interface that allows users to upload a CSV file and get gender predictions.

A Streamlit application was developed to provide a user interface for uploading CSV files containing extracted voice features. Upon file upload, the app preprocesses the data using the saved scaler and PCA, then uses the SVM model to make predictions on gender. The predictions are displayed in a readable format indicating whether the voice is male or female. This application allows end-users to leverage the machine learning model without needing programming skills.

**Why It Was Done:**

To make the model accessible for real-time predictions without needing to write code.

Key Features:

* Upload CSV with extracted features
* Preprocessing using saved scaler and PCA
* Predict using saved SVM model

**Sample Streamlit Code:**

import streamlit as st

import pandas as pd

import joblib

**# Load saved model, scaler, and PCA**

model = joblib.load("best\_model.pkl")

scaler = joblib.load("scaler.pkl")

pca = joblib.load("pca.pkl")

st.title("Voice Gender Prediction App")

uploaded\_file = st.file\_uploader("Upload CSV with voice features", type="csv")

if uploaded\_file is not None:

data = pd.read\_csv(uploaded\_file)

st.write("Uploaded Data Preview:", data.head())

if st.button("Predict"):

try:

**# Apply preprocessing**

data\_scaled = scaler.transform(data)

data\_pca = pca.transform(data\_scaled)

**# Predict**

predictions = model.predict(data\_pca)

prediction\_labels = ["Female" if p == 0 else "Male" for p in predictions]

st.write("Predictions:")

st.write(prediction\_labels)

except Exception as e:

st.error(f"Error in prediction: {e}")

**How to Run: Ran in Terminal**

streamlit run app.py

9. App execution:

A CSV file with 5 synthetic samples was created to test the Streamlit app.

