REPORT OF TASK 1:-

AIM: - Create a feature to predict the emotion using the audio file where we can record a voice as well as upload a voice note in the model. For an example If the voice note contains happy emotion we should say its happy likewise we should predict all emotions using voice note. This should work for only female voices not for other voices. If we upload or record other thing female voice it should say like upload female voice.

Task to be complete for complete this project:-

1) Create a model that can differentiate between the male and female voice.
2) Create a model that can predict the emotions of voice notes.
3) Merge/combine them to fulfil the demand of project.

1) Creating a model that can differentiate between the male and female voice

Gender Classification Using Voice Data

1. Introduction

This project involves using a logistic regression model to classify gender based on voice features. The data used for training and testing the model is sourced from a dataset of voice recordings with labelled genders.

2. Data Description

The dataset consists of 3168 samples, each with 20 numerical features related to voice frequencies and one categorical feature indicating gender (label). The features include:

- 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
- kurt: Kurtosis
- sp.ent: Spectral entropy
- sfm: Spectral flatness measure
- mode: Mode frequency (in kHz)
- centroid: Frequency centroid (in kHz)
- 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: Mean 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

3. Data Preprocessing

3.1 Label Encoding

The categorical labels were encoded as follows:

- male labeled as 1
- female labeled as 0

3.2 Feature and Target Separation

The features were separated from the target variable:

- X: Features (excluding the label column)
- y: Target variable (label)

3.3 Train-Test Split

The data was split into training and testing sets using an 80-20 split ratio with a random seed of 42 for reproducibility.

3.4 Feature Scaling

The features were standardized using StandardScaler from scikit-learn to ensure that each feature has a mean of 0 and a standard deviation of 1.

4. Model Training

A logistic regression model was trained on the standardized training data. The model was fitted using the training set X_train and y_train.

5. Model Evaluation

The trained model was used to make predictions on the test set X_test. The accuracy of the model was calculated using the accuracy_score function from scikit-learn. The results were as follows:

- Logistic Regression Training Score: 97.40%
- Logistic Regression Testing Accuracy: 98.11%

6. Saving the Model and Scaler

The trained logistic regression model and the scaler were saved to files using the joblib library:

- Model file: gender_prediction_model.pkl
- Scaler file: scaler.pkl

7. Prediction on New Data

A function extract_features was defined to extract relevant features from a .wav audio file. The extracted features were then standardized using the saved scaler, and the logistic regression model was used to predict the gender.

An example prediction was made on a new .wav file, resulting in a predicted gender of "female."

8. Conclusion

The logistic regression model demonstrated high accuracy in classifying gender based on voice features. The model and scaler were successfully saved for future use, and a function for extracting features from new audio files was implemented, allowing for easy and accurate gender predictions on new voice data.

2) Creating a model that can predict the emotion of voice
Emotion Classification Using Voice Data

1. Introduction

This project involves using an LSTM (Long Short-Term Memory) model to classify emotions based on voice features. The data used for training and testing the model is sourced from a dataset of voice recordings with labeled emotions.

2. Data Description

The dataset consists of 5600 samples, each with 40 MFCC (Mel-Frequency Cepstral Coefficients) features and one categorical feature indicating emotion (label). The labels represent different emotions such as 'happy', 'sad', 'fear', 'angry', 'neutral', 'disgust', and 'ps'.

3. Data Preprocessing

3.1 Data Loading

The dataset was loaded from a directory containing various audio files, each named to include the emotion label.

3.2 Dataframe Creation

A dataframe was created with two columns:

- `Speech`: Paths to the audio files.
- `Label`: Emotion labels extracted from the filenames.

3.3 Visualization

The distribution of different emotion labels was visualized using a count plot to ensure balanced classes for training.

3.4 Waveform and Spectrogram Visualization

Functions `wave` and `spectra` were defined to visualize the waveform and spectrogram of audio files for different emotions, aiding in understanding the audio features.

3.5 Feature Extraction

MFCC features were extracted from the audio files using the `librosa` library. Each audio file was processed to extract 40 MFCC features, which were then averaged.

3.6 Feature and Target Preparation

The MFCC features were compiled into a numpy array `X`. The target variable `y` was created using one-hot encoding for the emotion labels.

3.7 Data Reshaping

The feature array `X` was reshaped to fit the LSTM model's input requirements, resulting in a shape of `(5600, 40, 1)`.

4. Model Training

4.1 Model Architecture

A sequential LSTM model was constructed with the following layers:

- LSTM layer with 123 units
- Dense layer with 64 units and ReLU activation
- Dropout layer with a rate of 0.2
- Dense layer with 32 units and ReLU activation
- Dropout layer with a rate of 0.2

- Output dense layer with 7 units and softmax activation

4.2 Compilation

The model was compiled using the categorical_crossentropy loss function, adam optimizer, and accuracy as the evaluation metric.

4.3 Training

The model was trained on the dataset for 100 epochs with a batch size of 512, using 20% of the data for validation.

4.4 Evaluation

The training and validation accuracy and loss were plotted over epochs to visualize the model's performance.

5. Model Evaluation

The model demonstrated satisfactory performance with high accuracy on both training and validation sets.

6. Saving the Model and Data

The trained LSTM model and the processed dataset were saved to files for future use:

- Model file: emotion_detection_by_speech.h5
- Data file: toronto-emotional-speech-set-tess.csv

7. Prediction on New Data

7.1 Loading the Model and Data

The saved model and data were loaded to recreate the encoder and allow predictions on new audio files.

7.2 Feature Extraction for New Data

A function extract_mfcc was defined to extract MFCC features from new .wav audio files.

7.3 Making Predictions

The extracted features were processed and fed into the model to predict the emotion. The predicted emotion was retrieved using the inverse transform of the one-hot encoder.

8. Conclusion

The LSTM model demonstrated high accuracy in classifying emotions based on voice features. The model and encoder were successfully saved for future use, and a function for extracting features from new audio files was implemented, allowing for easy and accurate emotion predictions on new voice data.

3) I use streamlit framework to create a GUI in which I tried to combine them.

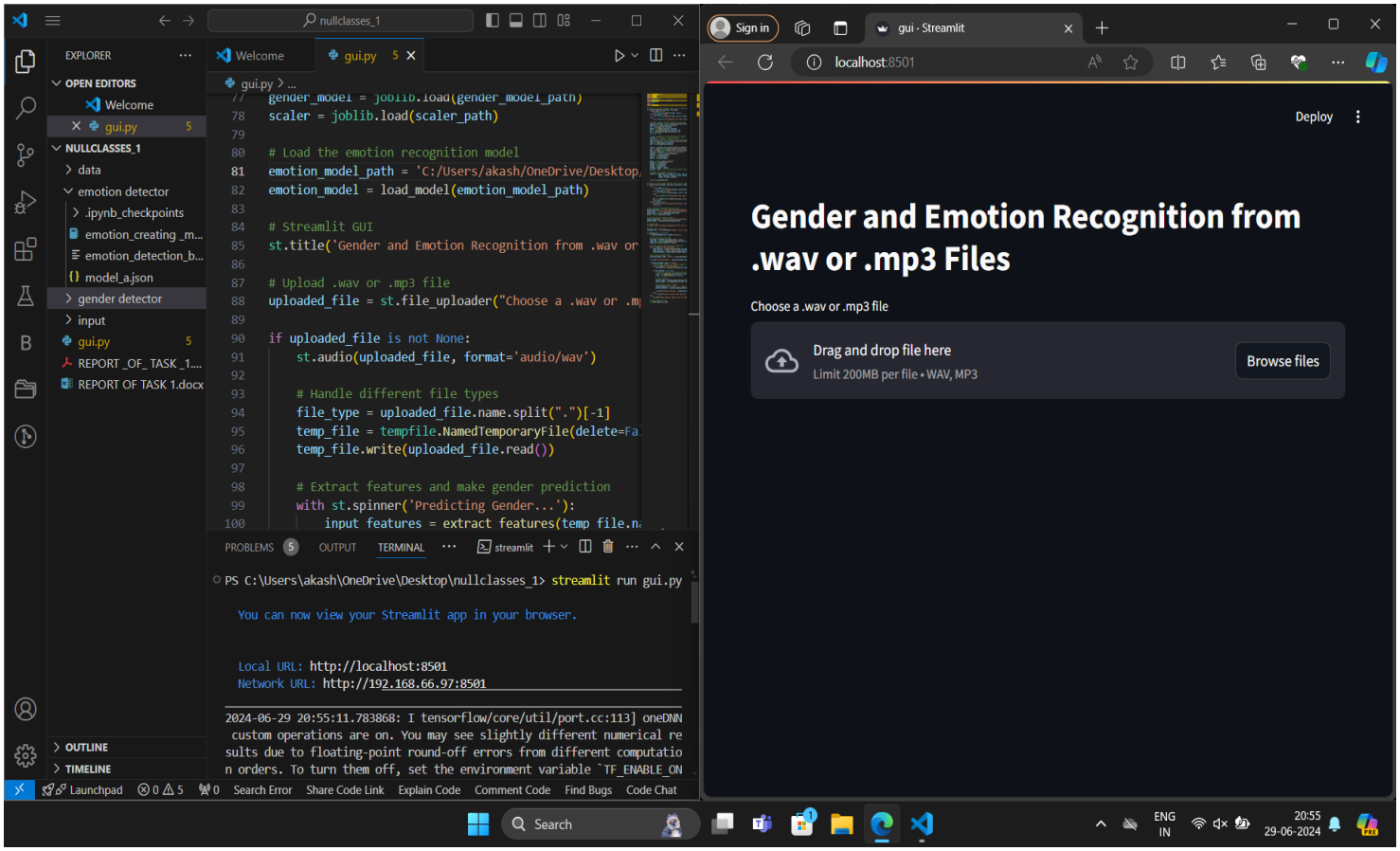
Introduction to Streamlit in Python

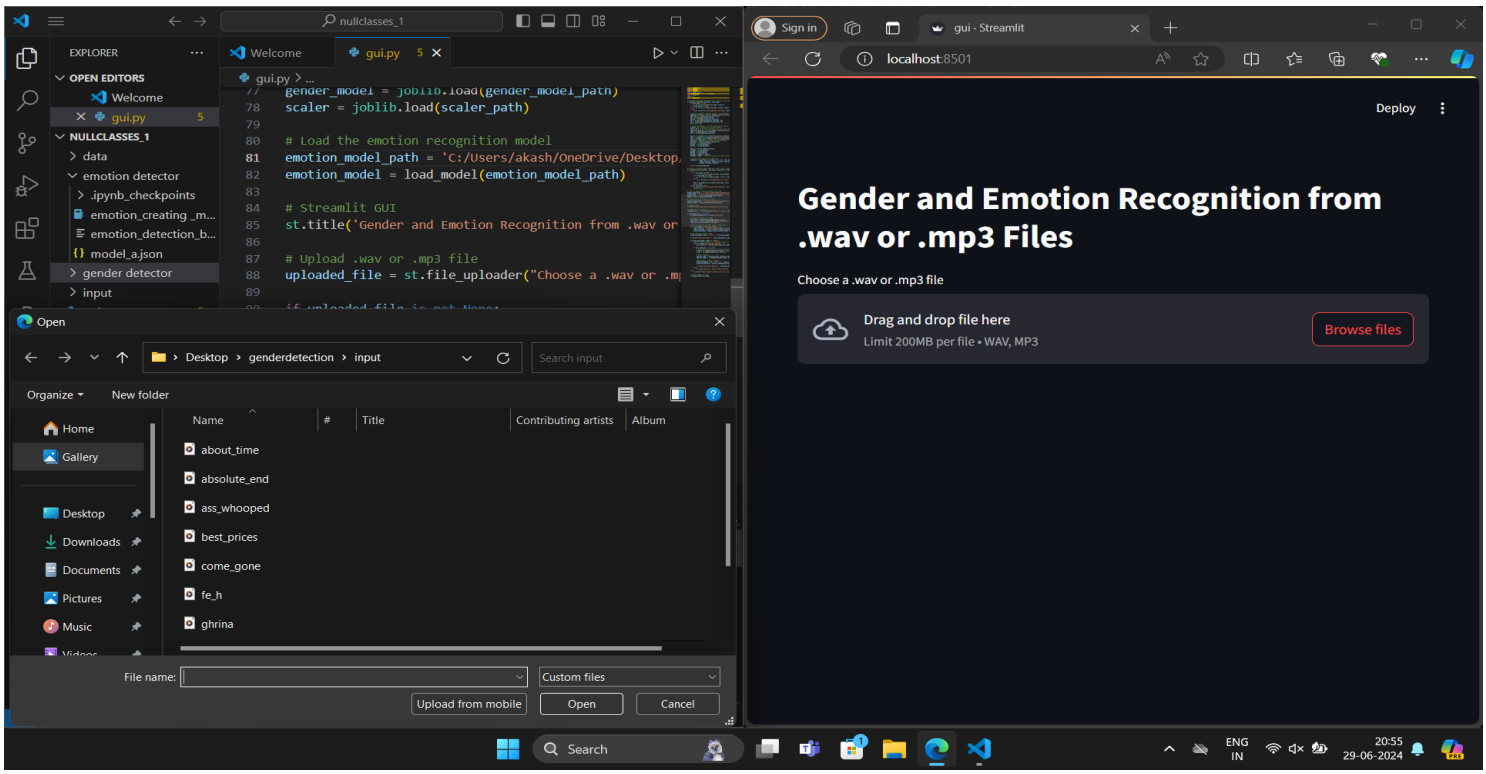
Streamlit is an open-source app framework for creating and sharing data applications built in Python. It's designed to be simple and quick to use, allowing data scientists and machine learning engineers to convert their data scripts into interactive web applications in just a few lines of code.

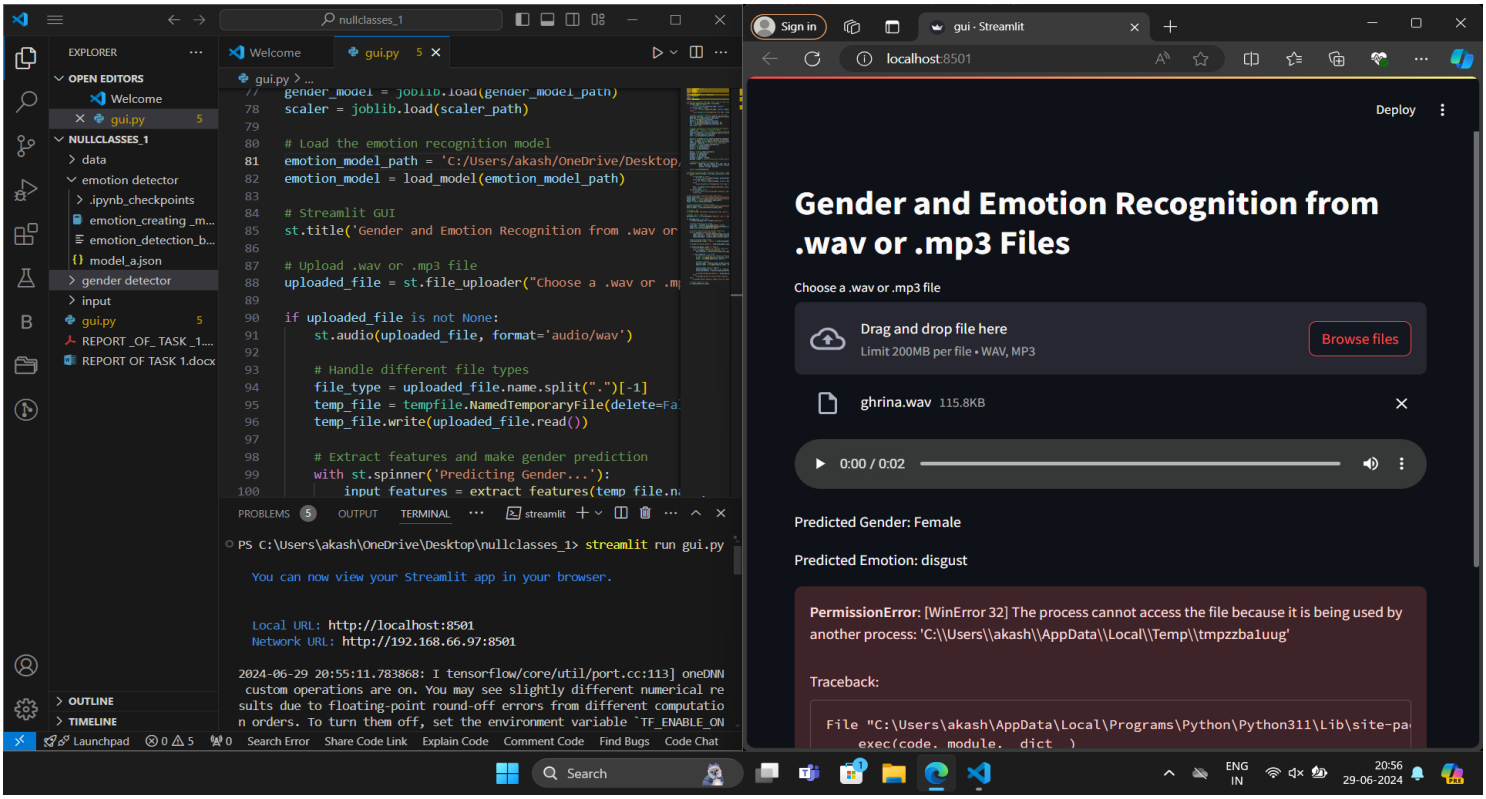
Key Features of Streamlit:

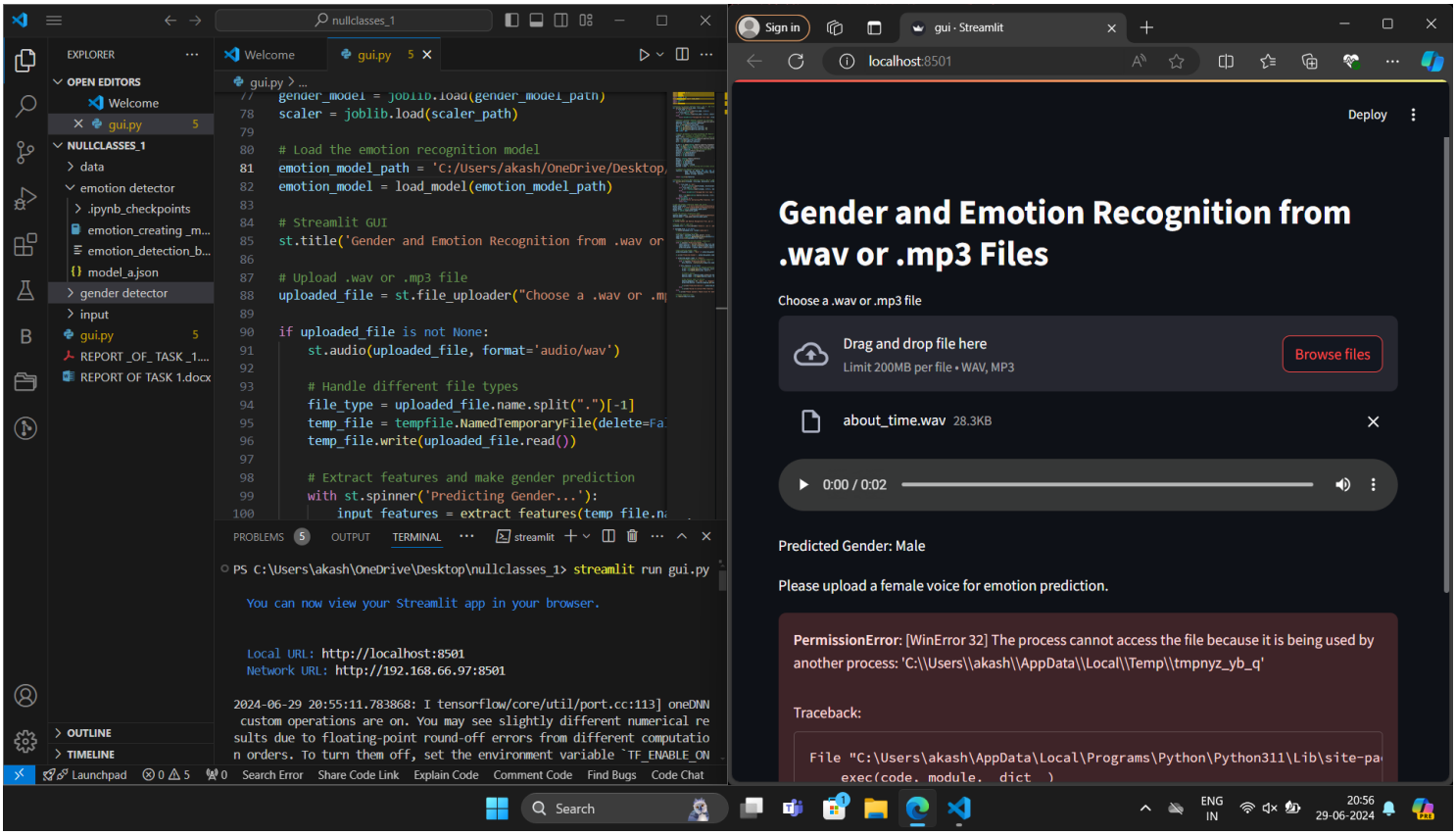
- **Simplicity**: Streamlit's API is easy to learn and use. You can turn data scripts into web apps with minimal effort.
- **Interactivity**: It provides widgets to add interactivity to your app, like sliders, text inputs, and buttons.
- **Data Visualization**: Supports popular visualization libraries such as Matplotlib, Plotly, Altair, and others.
- **Auto-Refresh**: Automatically refreshes and updates the app when you save your source code

#output:-









By ~ Akash kumar