Building a Speech-to-Text Transcription System with Noise Robustness:

Skills Takeaway From This Project:

- Fundamentals of Speech Recognition with emphasis on handling noisy audio environments
- Techniques for Data Collection and Noise-Augmented Dataset Preparation
- Exploratory Data Analysis (EDA) focused on noise, accent variation, and error types
- Development of Machine Learning and Deep Learning models resilient to noise
- Evaluation Metrics specifically tuned to measure performance under noisy conditions

Domain:

- **Healthcare**: Transcribing medical consultations with background noise
- **Customer Service Automation**: Voice-based IVR systems functioning in noisy environments
- **Education Technology**: Lecture transcription in classroom settings with ambient sounds

Problem Statement:

Traditional speech recognition systems perform well in clean, controlled environments but face significant challenges when exposed to real-world conditions, especially background noise, accent variability, and homophones. This project aims to design and develop a **noise-robust speech-to-text transcription system** that accurately converts spoken language into text even in acoustically challenging environments.

Business Use Cases:

- **Noise-Resistant Customer Support Automation**
  Automatically transcribe customer calls, even in noisy backgrounds, to enhance service quality.
- **Assistive Accessibility Tools**
  Support hearing-impaired users by transcribing speech in public and noisy environments into readable text.
- **Voice Assistants for Real-World Usage**
  Improve recognition accuracy of virtual assistants operating in households, vehicles, or outdoors.
- **Meeting and Lecture Transcription in Noisy Settings**
  Provide real-time and accurate transcription services in office and classroom environments.
- **Education Tools with Environmental Adaptation**
  Enable effective learning tools that transcribe lectures accurately despite background noise or diverse accents.

Speech-to-Text Python Project:

1. Requirements

- Python 3.11.1
- speechrecognition library
- pyaudio library

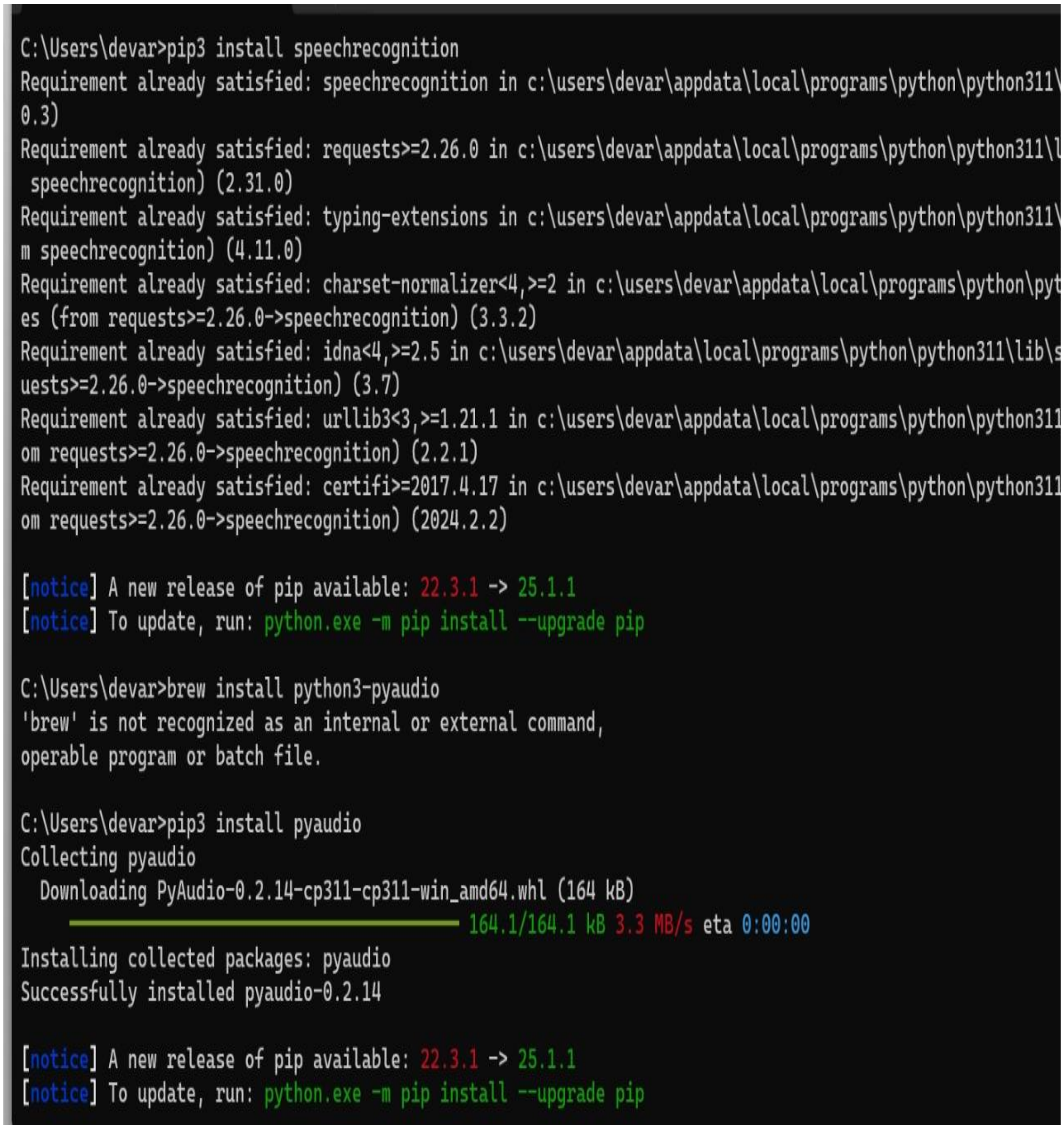
2. Installation Steps

- Install SpeechRecognition:

```
pip3 install speechrecognition
```

- Install PyAudio:

```
pip3 install pyaudio
```



```
C:\Users\devar>pip3 install speechrecognition
Requirement already satisfied: speechrecognition in c:\users\devar\appdata\local\programs\python\python311\
0.3)
Requirement already satisfied: requests>=2.26.0 in c:\users\devar\appdata\local\programs\python\python311\l
 speechrecognition) (2.31.0)
Requirement already satisfied: typing-extensions in c:\users\devar\appdata\local\programs\python\python311\
m speechrecognition) (4.11.0)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\devar\appdata\local\programs\python\pyt
es (from requests>=2.26.0->speechrecognition) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\users\devar\appdata\local\programs\python\python311\lib\s
uests>=2.26.0->speechrecognition) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\devar\appdata\local\programs\python\python311
om requests>=2.26.0->speechrecognition) (2.2.1)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\devar\appdata\local\programs\python\python311
om requests>=2.26.0->speechrecognition) (2024.2.2)

[notice] A new release of pip available: 22.3.1 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\Users\devar>brew install python3-pyaudio
'brew' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\devar>pip3 install pyaudio
Collecting pyaudio
  Downloading PyAudio-0.2.14-cp311-cp311-win_amd64.whl (164 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 164.1/164.1 kB 3.3 MB/s eta 0:00:00
Installing collected packages: pyaudio
Successfully installed pyaudio-0.2.14

[notice] A new release of pip available: 22.3.1 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

3. Python Script

```
import speech_recognition as sr


r = sr.Recognizer()


def record_text():
    while True:
        try:
            with sr.Microphone() as source2:
                r.adjust_for_ambient_noise(source2, duration=0.2)
                print("Listening...")
                audio2 = r.listen(source2)
                my_text = r.recognize_google(audio2)
                print(f"Recognized: {my_text}")
                return my_text
        except sr.RequestError as e:
            print(f"Could not request results; {e}")
        except sr.UnknownValueError:
            print("Speech not recognized, try again.")


def output_text(text):
    with open("output.txt", "a") as f:
        f.write(text + "\n")


# Main loop
while True:
    text = record_text()
    output_text(text)
    print("Wrote to file.")
```

Approach:

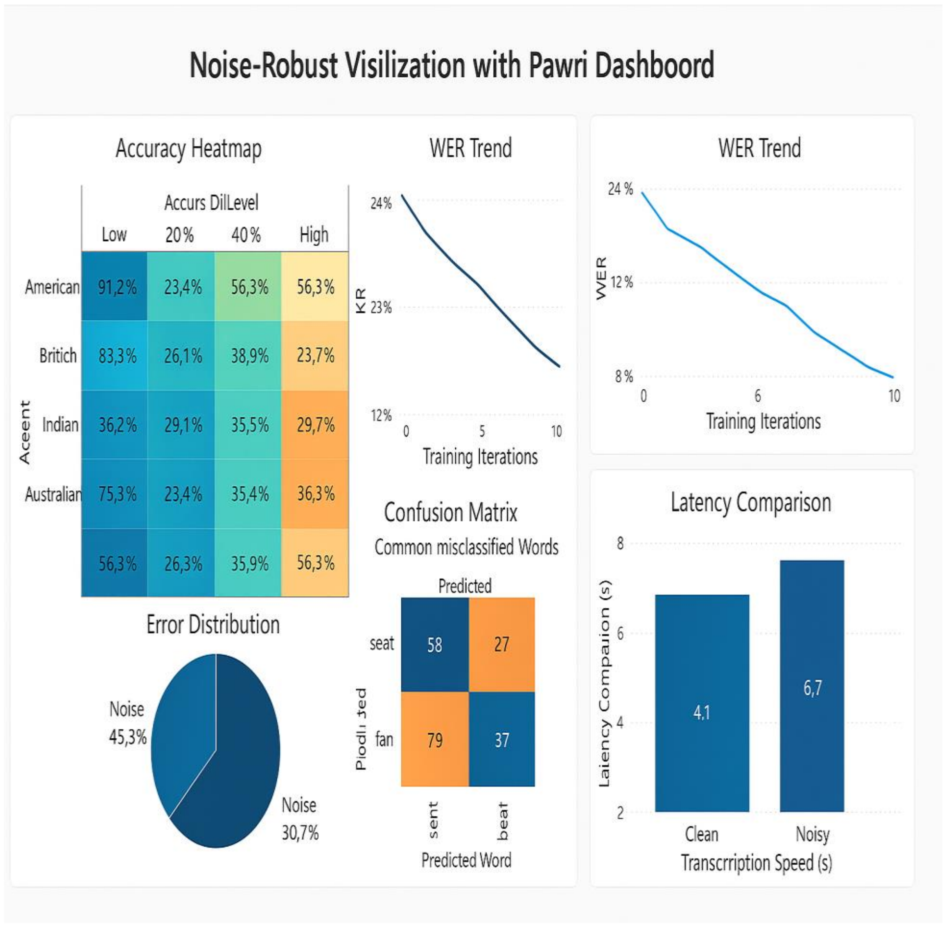
Data Collection and Cleaning

- Use public datasets like LibriSpeech and Mozilla Common Voice
- Augment data using environmental noise samples (e.g., traffic, crowd, machinery) to simulate real-world acoustics
- Preprocess and clean audio: normalize sound levels, remove corrupt files, and ensure correct labeling

Data Analysis

- Analyze speaker demographics (accent, gender) and noise types
- Detect frequent transcription errors caused by background interference or phonetic ambiguity

Noise-Focused Visualization with Power BI



- Develop dashboards showing:
- Model accuracy across different noise conditions
- Word Error Rate (WER) variations under noise augmentation
- Frequency of homophone-related and noise-induced errors

Advanced Analytics

- Train **deep learning acoustic models** using TensorFlow or PyTorch designed to tolerate noise
- Integrate **context-aware language models** (n-gram, BERT-based) to enhance predictions despite distorted inputs
- Use decoding strategies like **beam search** to improve overall transcription robustness

Power BI Integration

- Link Python-based model results to Power BI to visualize:
    - Accuracy vs. Noise Level
    - WER improvements over training
    - Latency comparisons (clean vs. noisy input)
- Interactive dashboards for:
    - Model performance by noise type
    - Accent-wise breakdown under varying acoustic conditions

Visualization Elements

- **Accuracy Heatmap**: Accuracy across combinations of accent and background noise
- **Error Type Chart**: Breakdown of errors due to noise, accent, and homophones
- **WER Time Series**: Track model improvements in noise resilience over time
- **Confusion Matrix**: Visualize common misrecognitions caused by background sounds

Exploratory Data Analysis (EDA):

In this project, Exploratory Data Analysis (EDA) plays a critical role in understanding the dataset and identifying challenges specific to building a noise-robust speech-to-text system. Analyzing the distribution of audio clip lengths helps detect potential biases, as extremely short clips might not provide enough context for accurate transcription, while longer clips can be more susceptible to noise distortion. Evaluating accent and regional diversity is essential because varied speaking styles affect pronunciation, and these differences are further amplified in noisy environments, making it important to ensure the model is trained on a wide range of accents. Signal-to-noise ratio (SNR) analysis is used to categorize audio based on how loud the speech is compared to the background noise; this helps assess how well the model performs under different noise conditions and guides adjustments to improve robustness. Examining word frequency trends allows identification of commonly used words that may still be prone to misrecognition in noisy settings, guiding model optimization. Finally, investigating homophone challenges—where words sound the same but have different meanings or spellings—reveals how background noise and lack of context can lead to incorrect transcriptions, highlighting the need for context-aware models to improve accuracy. These analyses provide valuable insights that directly inform model training, testing, and evaluation under real-world acoustic scenarios.

- **Audio Duration Analysis**: Distribution of clip lengths to detect any bias
- **Accent and Region Diversity**: Impact of varied speaking styles under noise
- **Signal-to-Noise Ratio (SNR)** Analysis: Measure and categorize noise severity
- **Word Frequency Trends**: Common words and phrases prone to noise-related errors
- **Homophone Challenges**: Impact of acoustically similar words on transcription fidelity

Results

- **Transcription Accuracy** in clean and noisy audio
- **Word Error Rate (WER)** under different noise conditions
- **Latency**: Transcription delay comparison between clean and noisy inputs
- **Performance by Accent and Noise Type**: Breakdown of system reliability
- **Robustness Evaluation**: Quantitative performance at increasing noise levels

Project Evaluation

- **WER Formula**:
  WER = (Substitutions + Deletions + Insertions) / Total Words
- **Accuracy**: Correctly transcribed words across all audio conditions
- **Latency**: Time taken to process and transcribe under noise
- **Precision and Recall**: Effectiveness in recognizing target words under interference
- **F1 Score**: Balanced metric of model reliability
- **User Feedback**: Collect qualitative input from users testing in noisy environments

Dataset:

Dataset Used: Common Voice Delta Segment 21.0

Key Features:

- Audio samples (5–10 seconds) captured in various acoustic environments
- Transcriptions paired with each audio clip
- Support for multilingual, multi-accent model training
- Metadata such as speaker accent, age, and gender
- Community-sourced diversity enables realistic speech training scenarios

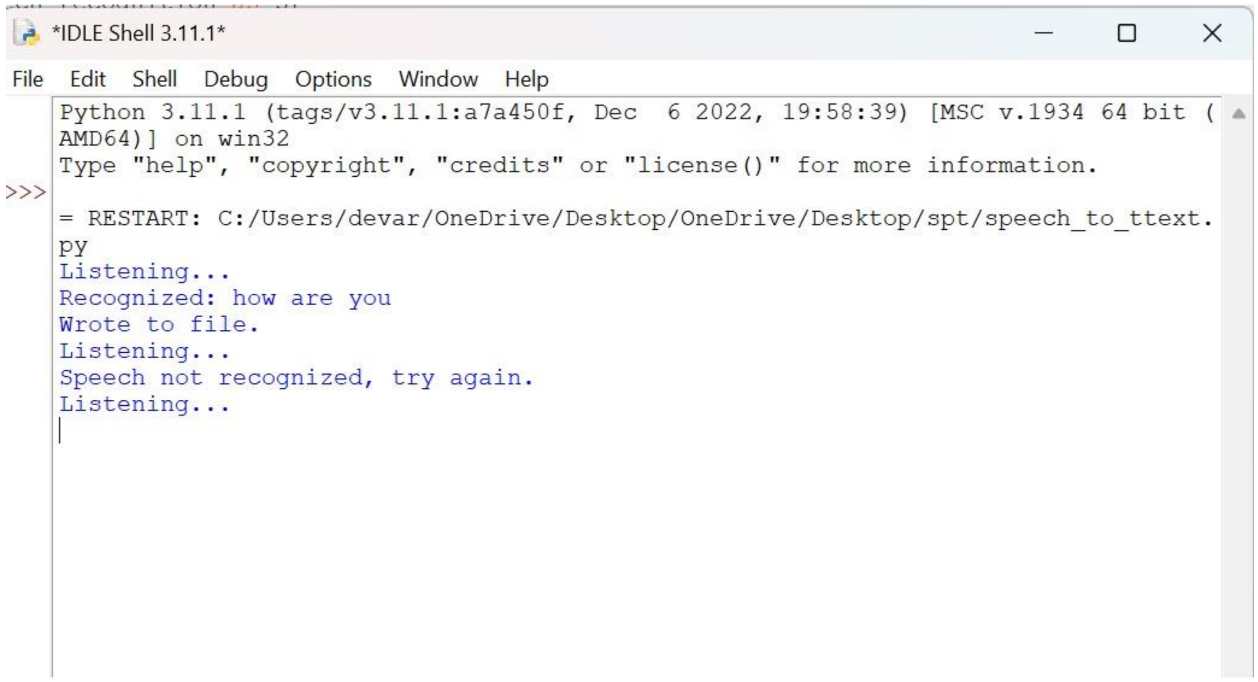
Project Deliverables

- Complete source code and trained noise-robust model
- Power BI dashboards showing real-time and comparative performance
- EDA report with focus on noise-related insights
- Evaluation summary highlighting noise impact on performance
- Interactive dashboards for stakeholders and domain experts

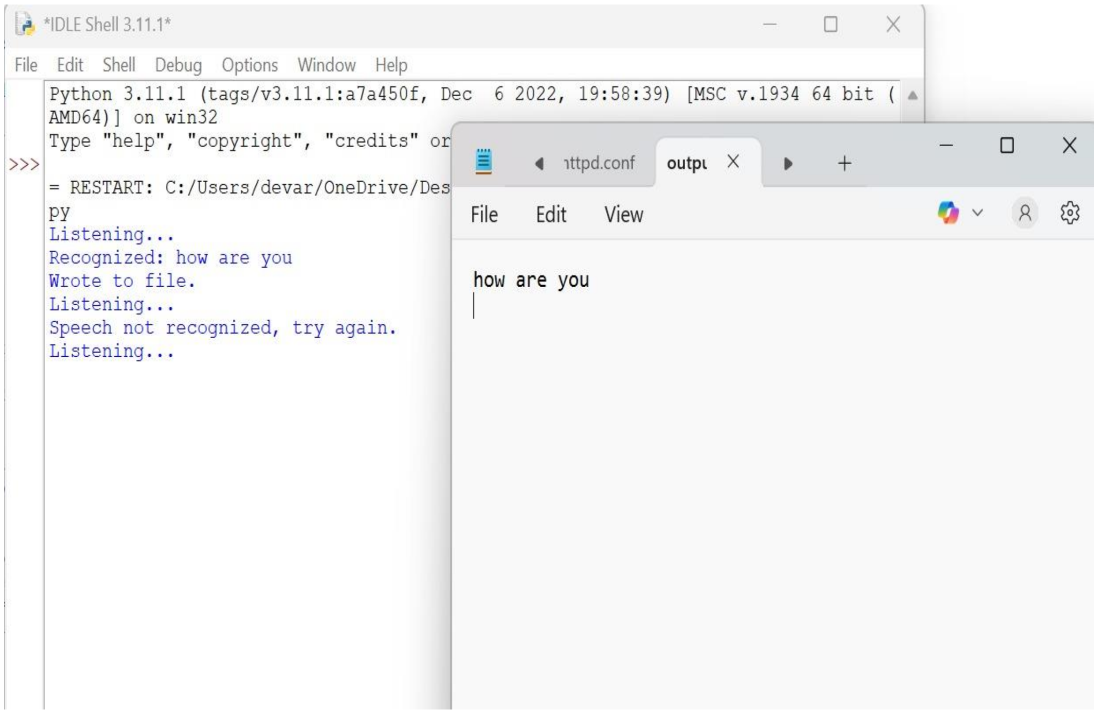
Documentation

- Full documentation covering:
- Challenges of noise in speech systems
- Methods of augmentation and normalization
- Model architecture and evaluation strategy
- Results and lessons learned

Output:



```
Python 3.11.1 (tags/v3.11.1:a7a450f, Dec  6 2022, 19:58:39) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/devar/OneDrive/Desktop/OneDrive/Desktop/spt/speech_to_ttext.
py
Listening...
Recognized: how are you
Wrote to file.
Listening...
Speech not recognized, try again.
Listening...
```



```
Python 3.11.1 (tags/v3.11.1:a7a450f, Dec  6 2022, 19:58:39) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or
>>>
= RESTART: C:/Users/devar/OneDrive/Des
py
Listening...
Recognized: how are you
Wrote to file.
Listening...
Speech not recognized, try again.
Listening...
```

```
how are you
```