Project Scoping – Speech Emotion Recognition

Group 3 - Team Members:

- Team Member 1: Debanjan Saha
- Team Member 2: Ajay Bana
- Team Member 3: Akhil Krishna Nair
- Team Member 4: Sai Venkat Madamanchi
- Team Member 5: Siddharth Banyal
- Team Member 6: Venkatesh Gopinath Bogem

Table of Contents

1. Introduction

Speech emotion recognition (SER) is a fascinating application of machine learning that involves analyzing human speech to determine the speaker's emotional state. It operates on the premise that vocal expressions contain a wealth of emotional information, manifesting in variations in tone, pitch, volume, and speech rate. By capturing and interpreting these acoustic nuances, SER systems aim to bridge the communicative gap between humans and machines, allowing for more intuitive and empathetic interactions across various technological domains.

Emotion extraction from speech is integral to numerous applications where understanding human emotion is beneficial. In customer service, it enables automated systems to respond appropriately to a customer's mood, improving the service quality and experience. In mental health, it can provide therapists with additional insights into a client's emotional well-being, particularly when changes in mood might not be as overtly expressed. It's also a step forward in creating emotionally intelligent AI that can adapt responses based on human emotions, fostering more natural and engaging interactions.

Our approach to developing a speech emotion recognition system will involve collecting a diverse dataset of spoken emotional expressions. We'll extract salient audio features and train machine learning models to classify these emotions accurately. Key phases will include data preprocessing, feature extraction, model selection, training, and validation. The application of deep learning techniques, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), will allow our models to capture the complex patterns inherent in emotional speech.

Existing applications of speech emotion recognition are broad and impactful. In interactive voice response (IVR) systems, SER can redirect calls based on the caller's emotional state, ensuring that frustrated customers are quickly attended to by human operators. AI personal assistants use emotion detection to tailor responses to the user's current mood, enhancing the user experience. Beyond customer service, SER is leveraged in security systems for stress detection, in entertainment for dynamic game experiences, and in automotive industries for monitoring driver alertness and emotional state. These applications underscore the growing importance of emotionally aware AI in our daily lives.

2. Dataset Information

2.1 Dataset Introduction:

Our dataset comprises a diverse collection of audio samples sourced from renowned databases like the Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS), Toronto Emotional Speech Set (TESS), Surrey Audio-Visual Expressed Emotion (SAVEE), and Crowd Sourced Emotional Multimodal Actors Dataset (CREMA-D), augmented with audio features to enhance the model's robustness across various demographics and contexts.

2.2 Data Card:

a. RAVDESS Dataset

Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) Speech audio-only files (16bit, 48kHz .wav) from the RAVDESS. Full dataset of speech and song, audio and video (24.8 GB) available from Zenodo. Construction and perceptual validation of the RAVDESS is described in the author's Open Access paper in PLoS ONE [1].

Our portion of the RAVDESS dataset contains 1440 files: 60 trials per actor x 24 actors = 1440. The RAVDESS contains 24 professional actors (12 female, 12 male), vocalizing two lexically-matched statements in a neutral North American accent. Speech emotions includes calm, happy, sad, angry, fearful, surprise, and disgust expressions. Each expression is produced at two levels of emotional intensity (normal, strong), with an additional neutral expression.

File naming convention:

Each of the 1440 files has a unique filename. The filename consists of a 7-part numerical identifier (e.g., 03-01-06-01-02-01-12.wav). These identifiers define the stimulus characteristics:

Filename identifiers:

- Modality (01 = full-AV, 02 = video-only, 03 = audio-only).
- Vocal channel (01 = speech, 02 = song).
- Emotion (01 = neutral, 02 = calm, 03 = happy, 04 = sad, 05 = angry, 06 = fearful, 07 = disgust, 08 = surprised).
- Emotional intensity (01 = normal, 02 = strong). NOTE: There is no strong intensity for the 'neutral' emotion.
- Statement (01 = "Kids are talking by the door", 02 = "Dogs are sitting by the door").
- Repetition (01 = 1st repetition, 02 = 2nd repetition).
- Actor (01 to 24. Odd numbered actors are male, even numbered actors are female).

Filename example: 03-01-06-01-02-01-12.wav

- Audio-only (03)
- Speech (01)
- Fearful (06)
- Normal intensity (01)
- Statement "dogs" (02)
- 1st Repetition (01)
- 12th Actor (12)
- Female, as the actor ID number is even.

Link to the original entire Dataset:- RAVDESS

b. TESS Dataset

A study was done to analyse the recognition of emotional speech for a young and an old speaker. The TESS (Toronto Emotional Speech Set) [2] dataset is female only and is of very high-quality audio. For almost 20 hours, each actor individually recorded the stimuli in a sound-attenuating booth. Three female undergraduate students with normal hearing, listened to the recordings and categorized them into one of the seven emotions for each actor. Most of the other dataset is skewed towards male speakers and thus brings about a slightly imbalance representation, but not this one. Because of that, this dataset would serve a very good training dataset for the emotion classifier in terms of generalisation (not overfitting)

There are a set of 200 target words ("stimuli") were spoken in the carrier phrase "Say the word _____' by two actresses (aged 26 and 64 years) and recordings were made of the set portraying each of seven emotions (anger, disgust, fear, happiness, pleasant surprise, sadness, and neutral). There are 2800 data points (audio files) in total.

The dataset is organised such that each of the two female actor and their emotions are contained within its own folder. And within that, all 200 target words audio file can be found. The format of the audio file is a WAV format.

Link to the original Dataset:- TESS

b. CREMA-D Dataset

The CREMA-D (Crowd Sourced Emotional Multimodal Actors) [3] dataset is an audio-visual dataset uniquely suited for the study of multi-modal emotion expression and perception, which consists of facial and vocal emotional expressions in sentences spoken in a range of basic emotional states, and can be used to probe other questions concerning the audio-visual perception of emotion. What's interesting is that this dataset is the sheer variety of data which helps train a model that can be generalised across new datasets. Many audio datasets use a limited number of speakers which leads to a lot of information leakage. CREMA-D has many speakers. For this fact, the CREMA-D is a very good dataset to use to ensure the model does not overfit.

CREMA-D is a data set of 7,442 original clips from 91 actors. These clips were from 48 male and 43 female actors between the ages of 20 and 74 coming from a variety of races and ethnicities (African America, Asian, Caucasian, Hispanic, and Unspecified). Actors spoke from a selection of 12 sentences. The sentences were presented using one of six different emotions (Anger, Disgust, Fear, Happy, Neutral, and Sad) and four different emotion levels (Low, Medium, High, and Unspecified).

Link to the original Dataset:-CREMA-D

d. SAVEE Dataset

The SAVEE (Surrey Audio-Visual Expressed Emotion) [4] database was recorded from four native English male speakers (identified as DC, JE, JK, KL), postgraduate students and researchers at the University of Surrey aged from 27 to 31 years. Emotion has been described psychologically in discrete categories: anger, disgust, fear, happiness, sadness and surprise. This is supported by the cross-cultural studies of Ekman [5] and studies of automatic emotion recognition tended to focus on recognizing these [6-8]. The authors added neutral to provide recordings of 7 emotion categories. The text material consisted of 15 TIMIT [9] sentences per emotion: 3 common, 2 emotion-specific and 10 generic sentences that were different for each emotion and phonetically-balanced. The 3 common and 2 × 6 = 12 emotion-specific sentences were recorded as neutral to give 30 neutral sentences.

This results in a total of 120 utterances per speaker, for example:

- Common: She had your dark suit in greasy wash water all year.
- Anger: Who authorized the unlimited expense account?
- Disgust: Please take this dirty table cloth to the cleaners for me.
- Fear: Call an ambulance for medical assistance.
- Happiness: Those musicians harmonize marvelously.
- Sadness: The prospect of cutting back spending is an unpleasant one for any governor.
- Surprise: The carpet cleaners shampooed our oriental rug.
- Neutral: The best way to learn is to solve extra problems.

Link to the original Dataset:- SAVEE

2.3 Data Rights and Privacy:

Data Compliance: The dataset aligns with GDPR, exemplifying adherence to the highest standards of data protection and privacy.

Privacy Considerations: Prioritizing privacy, the dataset is anonymization, safeguarding PII information. By meticulously removing personally identifiable details, the dataset ensures the utmost privacy for consumers.

3. Data Planning and Splits

Data preprocessing and augmentation are crucial in audio analysis to ensure models are trained on diverse, representative data, improving their ability to generalize to unseen samples. Preprocessing can include normalizing audio levels, trimming silence, and extracting relevant features like MFCCs, Chromagram, Mel Spectrograms, etc. which capture the timbral aspects of sound.

Augmentation, such as pitch shifting, time stretching, scale shifting and addition of Additive White Gaussian Noise (AGWN) to artificially expand the dataset, introducing variability that simulates different real-world conditions and speaker

variations. This process mitigates overfitting by making models more robust to variations in real-world audio signals.

A brief explanation of the kind of audio files in each dataset:

1. Surrey Audio-Visual Expressed Emotion (SAVEE): A collection of audio-visual emotional data from male actors, designed for research in human-computer interaction, psychology, and emotion analysis.

2. Toronto Emotional Speech Set (TESS): Features a range of emotional speech samples from female speakers, useful for studies in speech processing and emotional recognition.

3. Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A well-cited dataset containing audio and video recordings of actors expressing different emotions through speech and song, supporting emotion recognition research.

4. CREMA-D: A dataset of video and audio recordings where actors deliver lines with different emotional tones, intended for use in automated facial and vocal emotion recognition systems.

Each of the above mentioned dataset has its unique characteristics and applications in emotion recognition, speech processing, and multimodal sentiment analysis research.

When using multiple datasets like SAVEE, TESS, RAVDESS, and CREMA-D to train a model, we are considering diversity and balance across the datasets for the data split. First, we combine all datasets and shuffle them to ensure a mix of sources. Split the combined dataset into training (64%), validation (16%), and test (20%) sets, maintaining a distribution of all emotional categories and speakers across splits. This approach ensures the model learns general patterns applicable across different datasets, enhancing its robustness and ability to generalize.

4. GitHub Repository

- GitHub Repository Link: GitHub-repo
- **Folder Structure:**

```
.
├── LICENSE                 ## MIT License
├── README.md               ## START FROM HERE
├── assets                  ## assets
├── docs                    ## Documentation
├── pipeline
│   └── airflow             ## Airflow pipeline
│       ├── Dockerfile                  ## For Containerization
│       ├── config                      ## Airflow secrets
│       ├── dags                        ## Main DAGs directory
│       │   ├── README.md
│       │   ├── __init__.py
│       │   ├── build                   ## mlcore package build
│       │   ├── data_pipeline.py        ## data pipeline dag
│       │   ├── data_process.py         ## dummy dag
│       │   ├── dvc.yaml                ## DVC pipeline
```

```
│       │   ├── gcp.py                  ## GCP utility
│       │   ├── dist                    ## distribution
│       │   ├── logs                    ## running logs
│       │   │   └── running_logs.log
│       │   ├── setup.py                ## install packages script
│       │   ├── src                     ## source folder
│       │   │   ├── logs
│       │   │   ├── mlcore              ## core module for the project
│       │   │   │   ├── __init__.py
│       │   │   │   ├── artifacts                   ## artifacts
│       │   │   │   │   ├── data_ingestion          ## data
│       │   │   │   │   │   ├── cremad
│       │   │   │   │   │   ├── ravdess
│       │   │   │   │   │   ├── savee
│       │   │   │   │   │   └── tess
│       │   │   │   │   ├── data_transformation     ## files from data transformation
│       │   │   │   │   │   ├── data_parts.parquet
│       │   │   │   │   │   ├── train_data.parquet
│       │   │   │   │   │   ├── test_data.parquet
│       │   │   │   │   │   └── val_data.parquet
│       │   │   │   │   └── data_validation         ## files from data validation
│       │   │   │   │       ├── metadata.csv
│       │   │   │   │       └── status.txt
│       │   │   │   ├── components                  ## core modules
│       │   │   │   │   ├── __init__.py
│       │   │   │   │   ├── data_ingestion.py       ## ingestion script
│       │   │   │   │   ├── data_transformation.py  ## transformation script
│       │   │   │   │   ├── data_validation.py      ## validation script
│       │   │   │   │   ├── model_evaluation.py     ## evaluation script
│       │   │   │   │   └── model_trainer.py        ## model training script
│       │   │   │   ├── config                      ## configuration items
│       │   │   │   │   ├── __init__.py
│       │   │   │   │   ├── config.yaml             ## paths for various configuration files
│       │   │   │   │   └── configuration.py        ## configuration datastructures
│       │   │   │   ├── constants
│       │   │   │   │   ├── __init__.py             ## path to constants
│       │   │   │   │   ├── params.yaml             ## model parameters, data augmentation
│       │   │   │   │   └── schema.yaml             ## dataset schema
│       │   │   │   ├── entity                      ## frozen dataclass sets
│       │   │   │   │   ├── __init__.py
│       │   │   │   │   └── config_entity.py        ## dataclass schema settings
│       │   │   │   ├── logs
│       │   │   │   │   └── running_logs.log        ## logs
│       │   │   │   └── utils
│       │   │   │       ├── __init__.py
│       │   │   │       └── common.py               ## common utility functions
│       │   │   └── mlcore.egg-info                 ## distribution dependencies
│       │   ├── stage_01_data_ingestion.py          ## ingestion pipeline script
│       │   ├── stage_02_data_validation.py         ## validation pipeline script
│       │   ├── stage_03_data_transformation.py     ## transformation pipeline script
│       │   ├── stage_04_model_trainer.py           ## model training pipeline script
│       │   ├── stage_05_model_evaluation.py        ## model evaluation pipeline script
│       │   └── main.py                             ## All-in-one pipeline script
│       ├── docker-compose.yaml                     ## For starting containers
│       ├── logs
│       ├── plugins
│       └── requirements.txt
├── requirements.txt
└── tree.txt
```

The code has been modularized and containerized as per best practices, and we have also set various Coding Standards [Here] in place. The entire code uses sets of configuration files, where configuration items for the each and every step can be controlled without modifying the code.
Here is a brief snippet of our configuration file:

```yaml
artifacts_root: src/mlcore/artifacts

### ============================================================
###                 INGESTION CONFIGURATION
### ============================================================

data_ingestion_ravdess:
  root_dir: src/mlcore/artifacts/data_ingestion
  source_URL: 'ravdess:https%3A%2F%2Fstorage.googleapis.com%2Fkaggle-data-sets%2F107620%2F256618%2Fbundle%
  local_data_path: src/mlcore/artifacts/data_ingestion/ravdess

data_ingestion_tess:
  root_dir: src/mlcore/artifacts/data_ingestion
  source_URL: 'tess:https%3A%2F%2Fstorage.googleapis.com%2Fkaggle-data-sets%2F316368%2F639622%2Fbundle%2F
  local_data_path: src/mlcore/artifacts/data_ingestion/tess

data_ingestion_cremad:
  root_dir: src/mlcore/artifacts/data_ingestion
  source_URL: 'cremad:https%3A%2F%2Fstorage.googleapis.com%2Fkaggle-data-sets%2F325566%2F653195%2Fbundle%
  local_data_path: src/mlcore/artifacts/data_ingestion/cremad

data_ingestion_savee:
  root_dir: src/mlcore/artifacts/data_ingestion
  source_URL: 'savee:https%3A%2F%2Fstorage.googleapis.com%2Fkaggle-data-sets%2F338555%2F671851%2Fbundle%2
  local_data_path: src/mlcore/artifacts/data_ingestion/savee

### ============================================================
###                DATA VALIDATION CONFIGURATION
### ============================================================

data_validation:
  root_dir: src/mlcore/artifacts/data_validation
  unzip_ravdess_dir: src/mlcore/artifacts/data_ingestion/ravdess
  unzip_tess_dir: src/mlcore/artifacts/data_ingestion/tess/TESS Toronto emotional speech set data
  unzip_cremad_dir: src/mlcore/artifacts/data_ingestion/cremad/AudioWAV
  unzip_savee_dir: src/mlcore/artifacts/data_ingestion/savee/ALL
  local_output_path: src/mlcore/artifacts/data_validation/metadata.csv
  validation_status: src/mlcore/artifacts/data_validation/status.txt

### ============================================================
###              DATA TRANSFORMATION CONFIGURATION
### ============================================================

data_transformation:
  root_dir: src/mlcore/artifacts/data_transformation
  metadata_path: src/mlcore/artifacts/data_validation/metadata.csv
  output_path: src/mlcore/artifacts/data_transformation
```

5. Project Scope

5.1 Problems

Existing work in speech emotion recognition is facing several challenges, including variability in speech due to different languages, accents, and individual speaker characteristics. The emotional state can be subjective, leading to inconsistencies in labelled data. Limited availability of diverse, annotated datasets hinders training robust models. Furthermore, extracting relevant features from raw audio that accurately capture emotional nuances is complex. There's also the need for models to perform well in real-world, noisy environments, which can degrade recognition accuracy.

The primary objective of this project is to develop and implement a machine learning pipeline to help recognize an emotion in a speech in real time. By leveraging advanced machine learning and deep learning models, we aim to enhance the

existing emotion recognition, ensuring optimal resource utilization while making accurate emotion classifications of speech.

5.2 Current Solutions

Current solutions in the industry for Speech Emotion Recognition (SER) leverage advanced machine learning and deep learning models, integrating them into various applications. These solutions are employed in customer service to analyze caller sentiment, in mental health apps for monitoring emotional well-being, and in virtual assistants to respond appropriately to user emotions. Companies are also exploring multimodal emotion recognition, combining audio with visual cues to enhance accuracy. Cloud-based APIs and services that offer emotion recognition capabilities are becoming increasingly available, making SER more accessible to developers and businesses.

Amazon Alexa, Google Assistant, and Apple's Siri are increasingly incorporating aspects of Speech Emotion Recognition (SER) to enhance user interaction. While explicit details of their SER capabilities are not extensively publicized, these platforms are believed to use voice tone and pattern analysis to improve response accuracy and user experience. The focus is on creating more empathetic and context-aware interactions, potentially adjusting responses based on perceived user emotions. These advancements represent ongoing research and development efforts to integrate SER into widespread consumer technology, aiming for more intuitive and human-like interactions with AI assistants.

In SER, voice tone and pattern analysis involves extracting features from speech, such as pitch, energy, and rate, to identify emotional states. Machine learning algorithms analyze these features to classify emotions. This process allows systems like voice assistants to understand user sentiment, enabling them to respond in ways that are more aligned with the user's emotional state, thereby making interactions feel more natural and empathetic.

Existing virtual assistants primarily operate through cloud-based solutions but increasingly incorporate edge computing elements. This hybrid approach enables quick responses to basic commands locally on the device, enhancing privacy and reducing latency, while more complex queries and processing are handled in the cloud. This strategy optimizes both the performance and capabilities of these virtual assistants.

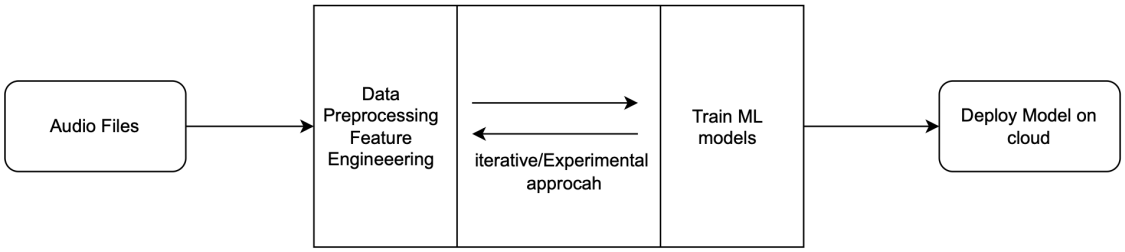
5.3 Proposed Solutions

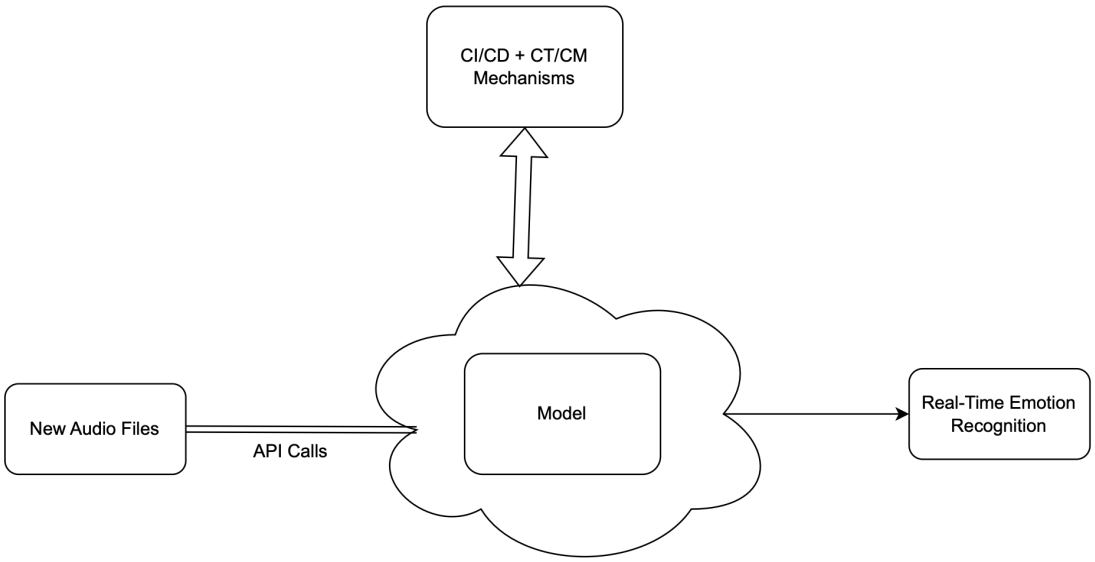
5.3.1 Predictive Analytics and Monitoring Software

In Speech Emotion Recognition (SER), machine learning and historical vocal data are utilized to identify emotional states from speech. By examining patterns in tone, pitch, and speech rate, SER tools offer nuanced insights, enabling technologies like virtual assistants to respond empathetically. Software platforms equipped with

analytical dashboards allow for the monitoring and improvement of SER applications, enhancing interaction quality between humans and AI systems. This proactive approach to understanding human emotions through speech advances the development of more intuitive user experiences.

6. Current Approach Flow Chart and Bottleneck Detection





7. Metrics, Objectives, and Business Goals

7.1 Business Goals

Implementing Speech Emotion Recognition (SER) for customer service in a tech company can significantly enhance business operations and user satisfaction. Accurate SER enables personalized customer interactions, optimizing service quality and efficiency. It aids in timely identifying and addressing customer frustrations, improving resolution rates.

Additionally, SER contributes to staff training by highlighting effective communication patterns. Data-driven insights from SER analytics can further refine customer engagement strategies, ensuring a consistently positive experience and fostering brand loyalty. This innovative approach aligns with modern expectations for empathetic, responsive service, setting the company apart in competitive markets.

7.2 Objectives

The primary objective of this project is to develop and implement a machine learning pipeline to help recognize an emotion in a speech in real time. By leveraging advanced machine learning and deep learning models, in a cloud-based machine learning platform. The accuracy of the methodology used to classify a speech into its respective emotion is the main focus of this project.

7.3 Success Metrics

This project is focused on SER, incorporating Continuous Training (CT), alongside CI/CD, Continuous Monitoring (CM), and dynamic dashboards for real-time metrics, the success criteria can be streamlined as follows:

1. Automated CI/CD and Continuous Training (CT) Workflow:
- Efficient automation of data ingestion, model retraining, evaluation, and deployment processes to adapt to new audio files and speeches.
- Seamless integration and deployment of updates with minimal manual effort, ensuring the model stays current with the latest data and algorithms.

2. Continuous Monitoring (CM) and Dashboards:
- Effective real-time monitoring of model performance (e.g., forecasting accuracy) and operational metrics (e.g., latency, throughput).
- Interactive dashboards that provide insights into model health, data quality, and the impact of weather on energy consumption.
- Automated alerts for model drift, data anomalies, or performance degradation, prompting timely adjustments.

3. Model and Data Management:
- Robust version control for models and datasets, enabling traceability and quick rollback if needed.
- High-quality data ingestion and preprocessing to ensure accurate and reliable classifications.

4. Scalability and Efficiency:
- Scalable architecture to handle varying volumes of audio files and formats.
- Optimized resource management, balancing computational costs with classification accuracy and timeliness.

5. Adaptability and Continuous Improvement:
- Flexibility to incorporate new audio sources, new audio formats that require new ETL processes.Commitment to iterative improvement through regular feedback loops and model updates.

Success in this context is defined not just by technical robustness but also by the model's ability to deliver actionable insights, drive operational efficiencies, and adapt to evolving data landscapes and business needs.

8. Failure Analysis

To address potential risks, a comprehensive failure analysis strategy is essential.

Failures in data pipelines can significantly impact data analysis, business intelligence, and decision-making processes. Here are some common examples of failures in pipelines:

- Missing or incomplete data due to extraction errors or source system availability issues. Transformation errors, leading to inaccurate analytics and business intelligence insights.
- Slow processing due to inefficient code or inadequate hardware resources, causing delays in data availability.
- Inability to handle increased data volumes or new audio sources, leading to system overload or significant performance degradation.
- Hardcoded or inflexible designs that make it difficult to adapt to changing data processing requirements.
- Failures in external services or data sources that the pipeline depends on, cause data ingestion issues.
- Inconsistent data across different stages of the pipeline or when integrating data from multiple sources.

Building software solutions to SER using audio files involves complex data processing and modeling steps. Several potential failures can occur during this process, affecting the accuracy and reliability. Here are some common issues:

- Including features that have little to no predictive power can reduce model performance.
- Failing to include critical audio parameters or derived features that significantly influence energy consumption.
- Building a model that is too complex for the available data can lead to overfitting, where the model performs well on training data but poorly on unseen data.
- Conversely, a model that is too simple may not capture the underlying relationships between the audio features and the inherent emotion, resulting in underfitting.
- Failing to account for variations in noise, voice and complex emotions in real-time audio data.
- Failure to regularly update the model with new data or retrain it to adapt to new audio files.

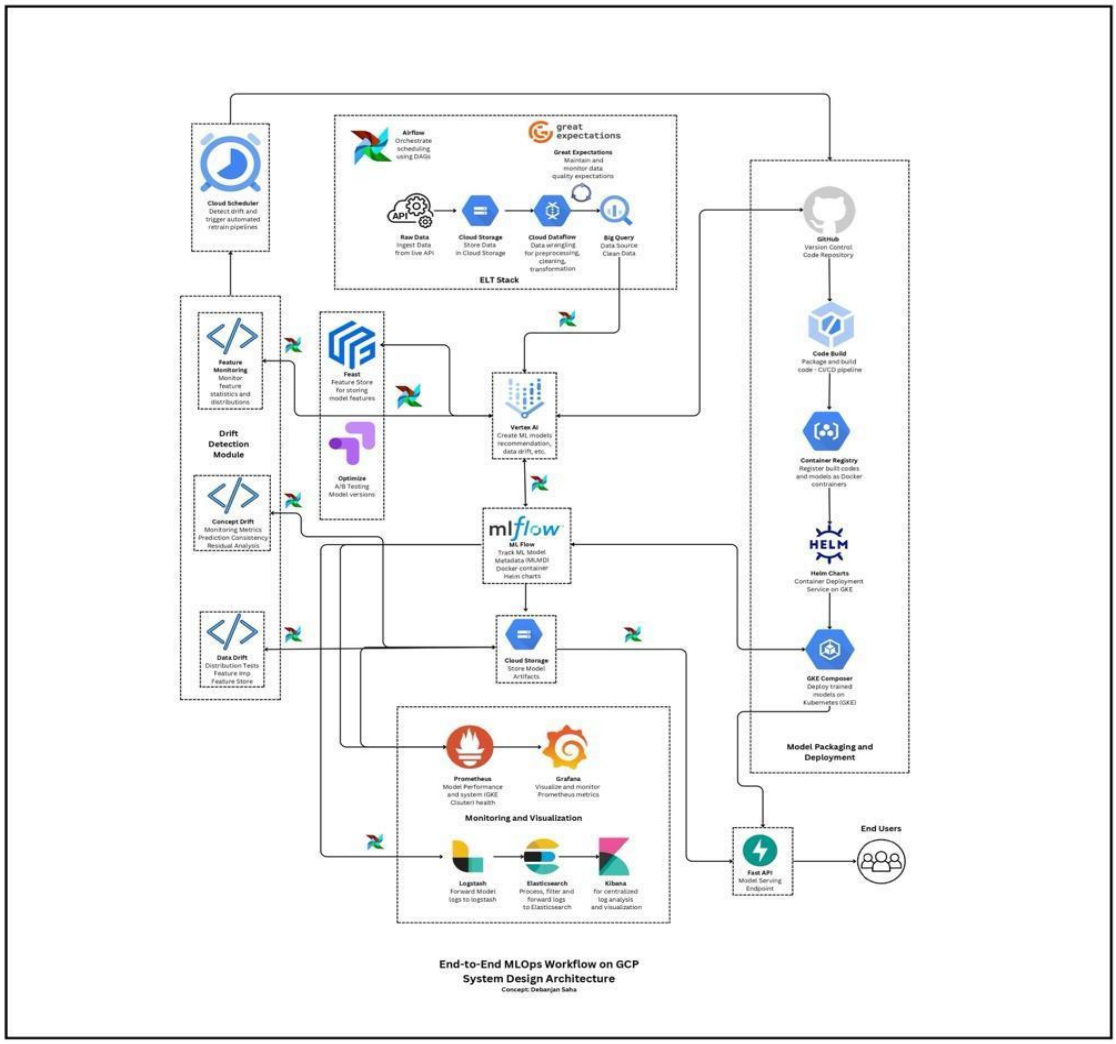
Effective monitoring, robust error handling, and regular maintenance are essential to mitigate these failures. Implementing best practices in data pipeline design, such as

ensuring data quality, scalability, security, and fault tolerance, can help in preventing these issues and maintaining reliable data processing systems.

In summary, a proactive approach involving ongoing monitoring, well-defined triggers, and thorough documentation serves as a foundation for robust failure analysis. This strategy enables quick identification and mitigation of issues, contributing to the overall effectiveness of machine learning systems and their adaptability to changing conditions.

9. Deployment Infrastructure

Our deployment infrastructure will be hosted on Google Cloud Platform (GCP) using Google Kubernetes Engine (GKE) for efficient container orchestration. This choice provides scalability, ease of management, and integration with various GCP services.



9.1 Infrastructure Components:

9.1.1 GCP GKE Cluster:

- GKE will serve as the foundation for hosting our machine learning model containers. It allows for automated scaling, management, and Helm charts for Kubernetes orchestration, ensuring a robust and resilient deployment.

9.1.2 Docker Containers:

- The machine learning model, along with its dependencies, will be containerized using Docker. This ensures consistent deployment across different environments, facilitating reproducibility.

9.1.3 MLFlow for Model Tracking:

- MLFlow will be integrated into our MLOps pipeline for comprehensive model tracking and management. It provides capabilities for tracking experiments, packaging code, and sharing and deploying models.

9.1.3 Airflow for Orchestration:

- Most of our non-kubernetes components will be orchestrated using Apache Airflow as it allows the creation of DAGs which facilitate seamless scheduling periodic tasks like executing data flows, data pre-processing, running experiments, executing custom tasks, and much more.

9.2 Deployment Process:

9.2.1 CI/CD Pipeline:

- A continuous integration and continuous deployment (CI/CD) pipeline will be established to automate the deployment process using Code Build. This pipeline will include steps for testing, building Docker images, deploying to GKE, and managing MLFlow experiments.

9.2.2 Kubernetes Deployments:

- Helm charts containing Kubernetes manifests will define the deployment specifications for our machine learning model and accompanying services. These manifests will be version-controlled and applied to the GKE cluster as part of the CI/CD process.

9.2.3 MLFlow Integration:

- MLFlow server components will be deployed as part of the GKE cluster. MLFlow Tracking will be integrated to log and organize experiments, parameters, metrics, and artifacts. MLFlow Models will enable easy model versioning and deployment.

10. Monitoring Plan

A robust monitoring plan is essential for ensuring the continuous health, performance, and reliability of our electricity demand forecasting model. The monitoring plan encompasses various aspects of the MLOps pipeline, including model performance,

system metrics, and data quality. The integration of Prometheus, Grafana, and the ELK stack will play a pivotal role in capturing and visualizing these metrics.

10.1 Monitoring Components:

10.1.1 Model Performance Metrics:

- **Metrics Tracked:** accuracy, precision, recall, and F-1score.
- **Monitoring Frequency:** Real-time monitoring updated once every 100 emotion classifications.
- **Alerts:** Trigger alerts if **tracked metrics** deviates significantly from the baseline or exceeds a predefined threshold.

10.1.2 Resource Utilization:

- **Metrics Tracked:** CPU and memory usage of the deployed model containers.
- **Monitoring Frequency:** Real-time monitoring with Prometheus.
- **Alerts:** Notify if resource utilization approaches predefined limits to prevent performance degradation.

10.1.3 Data Quality Checks:

- **Metrics Tracked:** Missing values, outliers, and distribution shifts in incoming data.
- **Monitoring Frequency:** Daily batch checks and real-time streaming checks.
- **Alerts:** Flag anomalies in the data distribution or significant data quality issues.

10.1.4 MLFlow Tracking:

- **Metrics Tracked:** Experiment metrics, model versions, and deployment artifacts.
- **Monitoring Frequency:** Continuous tracking with every model update.
- **Alerts:** Notify if there are discrepancies in logged metrics or issues with model versions.

10.1.5 Log Management:

- **Logs Tracked:** Deployment logs, application logs, and error logs.
- **Monitoring Frequency:** Real-time log streaming with the ELK stack.
- **Alerts:** Alert on critical errors or unusual patterns in logs that may indicate issues.

10.2 Visualization and Reporting:

10.2.1 Grafana Dashboards:

- Customized Grafana dashboards will provide a visual representation of model performance, resource utilization, and other critical metrics. These dashboards will enable the operations team to quickly identify trends and potential issues.

10.2.2 Kibana Visualizations:

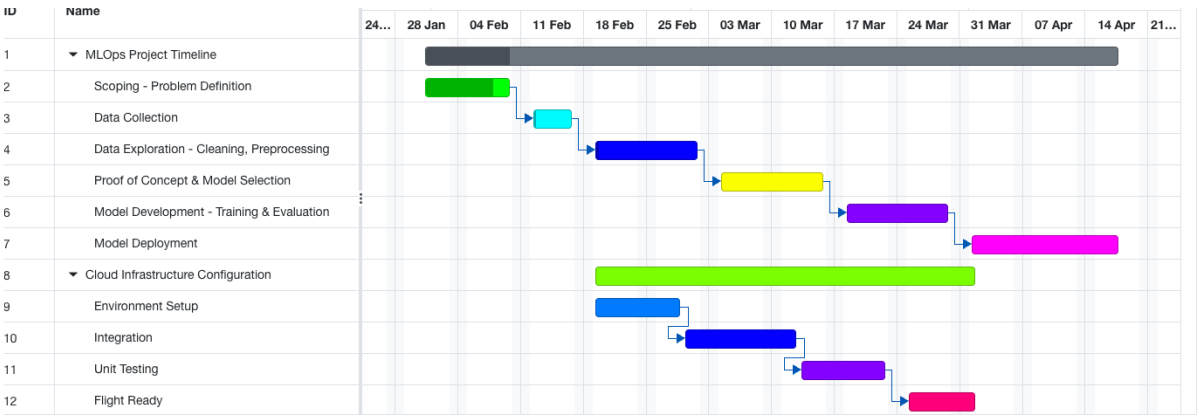
- Kibana will be used to create visualizations for log data, allowing for efficient analysis of log patterns and facilitating troubleshooting.

11. Success and Acceptance Criteria

Success defined by achieving accuracy goals. Acceptance criteria include stakeholder approval and successful deployment.

12. Timeline Planning

We have planned a phase-by-phase execution of this project starting from scoping to data collection, creating a proof-of-concept (POC), model building, testing, evaluating, to final deployment and monitoring. While some of the critical components of our architecture are being developed we also plan to bring up our cloud development environment in Google Cloud Platform. Below is a gantt chart showing our development activities in phase by phase manner.



- Data Collection and Preprocessing: 3 weeks
- Model Development and Training: 4 weeks
- Testing and Validation: 3 weeks
- Deployment and Monitoring: 3 weeks

13. Additional Information

13.1 References

1. Livingstone, S.R., & Russo, F.A. (2018). The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English. *PLoS ONE, 13*.
2. Dupuis, K., & Pichora-Fuller, M.K. (2011). Recognition of emotional speech for younger and older talkers: Behavioural findings from the toronto emotional speech set. *Canadian Acoustics, 39*, 182-183.

3. Cao, H., Cooper, D.G., Keutmann, M.K., Gur, R.C., Nenkova, A., & Verma, R. (2014). CREMA-D: Crowd-Sourced Emotional Multimodal Actors Dataset. *IEEE Transactions on Affective Computing, 5*, 377-390.
4. Jackson, Philip & ul haq, Sana. (2011). Surrey Audio-Visual Expressed Emotion (SAVEE) database.
5. Ekman, P. (2003). Emotions Revealed: Recognizing Faces and Feelings to Improve Communication and Emotional Life.
6. Sharma, R., Pachori, R.B., & Sircar, P. (2020). Automated emotion recognition based on higher order statistics and deep learning algorithm. *Biomed. Signal Process. Control., 58*, 101867.
7. Tuncer, T., Dogan, S., & Acharya, U.R. (2021). Automated accurate speech emotion recognition system using twine shuffle pattern and iterative neighborhood component analysis techniques. *Knowl. Based Syst., 211*, 106547.
8. Tawari, A., & Trivedi, M.M. (2010). Speech Emotion Analysis: Exploring the Role of Context. *IEEE Transactions on Multimedia, 12*, 502-509.
9. Lopes, C., & Perdigão, F. (2012). TIMIT Acoustic-Phonetic Continuous Speech Corpus.