DATA ANALYSIS USING PYTHON PROJECT

“Unified Data Insights: Analysing CSV, Image, and Text Datasets with Python"

A Project Lab Report in Partial Fulfillment of the degree

Bachelor of Technology

in

Computer Science & Artificial Intelligence

By

2303A52L05 – P. SATHVIK

Submitted to

Dr. Ramesh Dadi

Assistant Professor, School of CS&AI.

TABLE OF CONTENT

1. INTRODUCTION

In today's data-driven world, the ability to analyse and extract insights from diverse types of data is a critical skill. This capstone project demonstrates an end-to-end application of data analysis and machine learning techniques using Python across three distinct types of datasets: tabular (CSV), image, and textual data. By working with heterogeneous data formats, the goal was to explore the preprocessing needs, model development strategies, and evaluation methods specific to each data type.

The datasets used in this project are:

- **Air Quality and Health Impact Dataset**: A tabular dataset containing metrics related to environmental pollution and its health implications.
- **Image Dataset**: A collection of images across multiple categories used for multi-class image classification.
- **English Word Difficulty Classification Dataset**: A text-based dataset aimed at classifying words based on their difficulty levels for undergraduate and postgraduate students.

Each dataset posed unique challenges and required domain-specific preprocessing and modelling techniques. This report details how data preprocessing, model selection, evaluation, and statistical analysis were tailored to the nature of each dataset to derive meaningful insights and optimize performance.

2. Objectives

The primary objectives of this capstone project are:

To explore and analyse multimodal datasets—text, image, and tabular—to gain a holistic understanding of data analysis across formats.

To perform relevant preprocessing on each dataset based on its nature, including cleaning, normalization, encoding, feature extraction, and transformation.

To build and evaluate predictive models suited to each dataset:

- For the **CSV dataset**, apply and compare traditional machine learning models and perform statistical tests (z-test, t-test, ANOVA) to support findings.
- For the **image dataset**, build a custom Convolutional Neural Network (CNN) to perform multi-class classification and analyse the performance using statistical evaluation.

- For the **text dataset**, convert words to embeddings using pre-trained models, train a Long Short-Term Memory (LSTM) network, and benchmark against traditional ML models.

To assess the performance of each approach using appropriate metrics and draw comparisons where applicable.

To synthesize insights from working with heterogeneous data types and understand how preprocessing and modelling decisions vary across domains.

3. Overview of Datasets

Dataset	Type	Source	Key Features	Purpose in Project
Covid Data	Structured CSV (tabular)	Covid-19 Dataset	**DIABETES, ASTHMA,OBESITY, etc.**: Indicators of various comorbid conditions **INTUBED, ICU**: Indicates whether the patient was intubated or admitted to ICU	Classification and risk analysis of COVID-19 severity or outcome based on patient metadata and pre-existing conditions.
Image Dataset	Image	Kaggle-Image-Dataset	**Image Size:** Resized to 150x150 pixels for consistency **Colour Mode:** RGB (converted from original grayscale if needed)	Multiclass image classification to distinguish between COVID-19, normal, and pneumonia-infected chest X-rays using Convolutional Neural Networks (CNNs).
English Word Difficulty Classification	Text (NLP)	Coronavirus tweets NLP - Text Classification	**Difficulty Label**: • UG Dataset: Easy, Medium, Hard (encoded as 0, 1, 2)	To **classify English words** based on their **perceived difficulty levels** using both traditional machine

Dataset	Type	Source	Key Features	Purpose in Project
			• PG Dataset: Easy, Hard (encoded as 0, 1)	learning and deep learning (LSTM) models.

4. Dataset wise Analysis

4.1 CSV Dataset: Tabular Data Analysis (COVID-19 Dataset)

4.1.1 Data Analysis

Dataset Overview:

- Total Records: 1,048,575
- Features include: Age, gender, medical unit, comorbidities (diabetes, asthma, etc.), COVID classification, ICU status, and patient outcomes.

	USMER	MEDICAL_UNIT	SEX	PATIENT_TYPE	DATE_DIED	INTUBED	PNEUMONIA	AGE	PREGNANT	DIABETES	...	ASTHMA	INMSUPR	HIPERTENSION	OTHER_DISE
0	2	1	1	1	03/05/2020	97	1	65	2	2	...	2	2	1	
1	2	1	2	1	03/06/2020	97	1	72	97	2	...	2	2	1	
2	2	1	2	2	09/06/2020	1	2	55	97	1	...	2	2	2	
3	2	1	1	1	12/06/2020	97	2	53	2	2	...	2	2	2	
4	2	1	2	1	21/06/2020	97	2	68	97	1	...	2	2	1	

5 rows × 21 columns

Initial Observations:

- Age ranges from children to elderly patients.
- Binary/categorical values (e.g., 1, 2, 97) are used to denote presence/absence or unknown states for conditions such as diabetes, obesity, and tobacco use.
- `DATE_DIED` includes '9999-99-99' to indicate survival or missing data.

Data Cleaning Steps:

- Converted coded values to meaningful labels for interpretability (e.g., SEX: 1 → Male, 2 → Female, 97 → Unknown).
- Removed or imputed invalid codes (97, 98, etc.).
- Transformed DATE_DIED into a boolean “Survived” flag.
- Balanced the target variable (CLASIFFICATION_FINAL) using SMOTE to address class imbalance.

Visual Analysis:

- Plotted histograms of age distribution, showing a higher prevalence of middle-aged and elderly patients.
- Boxplots revealed outliers in features like age and ICU admission.
- Heatmaps showed correlations between comorbidities and ICU admissions.

Skewness and Kurtosis

```
Column: USMER
  Unique values: [2 1]
  Min: 1, Max: 2
  Mean: 1.4538432348981427, Std: 0.49786628877618616
  Skewness: 0.18542024637885093
  Kurtosis: -1.9656198683770683
----------------------------------------
Column: MEDICAL_UNIT
  Unique values: [ 1  2  3  4  5  6  7  8  9 10 11 12 13]
  Min: 1, Max: 13
  Mean: 7.364950071994054, Std: 3.6825041798343903
  Skewness: 0.31528339702270947
  Kurtosis: -1.7514278275045745
----------------------------------------
Column: SEX
  Unique values: [2 1]
  Min: 1, Max: 2
  Mean: 1.5949931644635273, Std: 0.49089464382644377
  Skewness: -0.38702460188366217
  Kurtosis: -1.8502142933843675
----------------------------------------
Column: PATIENT_TYPE
  Unique values: [2]
  Min: 2, Max: 2
```

```
Skewness and Kurtosis:
USMER - Skewness: 0.20, Kurtosis: -1.96
MEDICAL_UNIT - Skewness: 0.30, Kurtosis: -1.76
SEX - Skewness: -0.40, Kurtosis: -1.84
PATIENT_TYPE - Skewness: nan, Kurtosis: nan
INTUBED - Skewness: nan, Kurtosis: nan
PNEUMONIA - Skewness: 0.20, Kurtosis: -1.96
AGE - Skewness: -0.24, Kurtosis: -0.18
PREGNANT - Skewness: nan, Kurtosis: nan
DIABETES - Skewness: -1.19, Kurtosis: -0.59
COPD - Skewness: nan, Kurtosis: nan
ASTHMA - Skewness: nan, Kurtosis: nan
INMSUPR - Skewness: nan, Kurtosis: nan
HIPERTENSION - Skewness: -1.14, Kurtosis: -0.70
OTHER_DISEASE - Skewness: nan, Kurtosis: nan
CARDIOVASCULAR - Skewness: nan, Kurtosis: nan
OBESITY - Skewness: nan, Kurtosis: nan
RENAL_CHRONIC - Skewness: nan, Kurtosis: nan
TOBACCO - Skewness: nan, Kurtosis: nan
CLASIFFICATION_FINAL - Skewness: 0.33, Kurtosis: -1.64
```

4.2 Data Preprocessing

 The notebook "DAUP_PROJECT_NUMERIC.ipynb" begins by loading the "Covid Data.csv" into a pandas DataFrame.

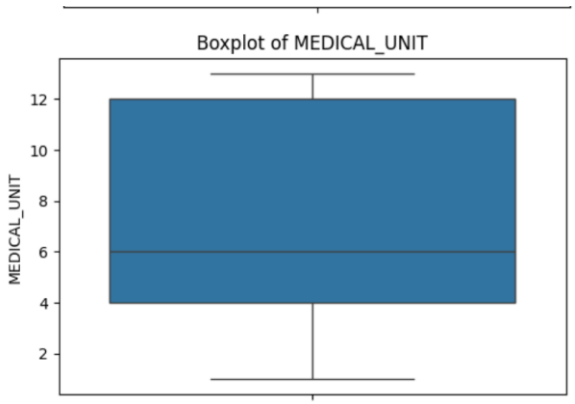
 The dataset contains 21 columns, including features like USMER, MEDICAL_UNIT, SEX, AGE, and various health conditions.

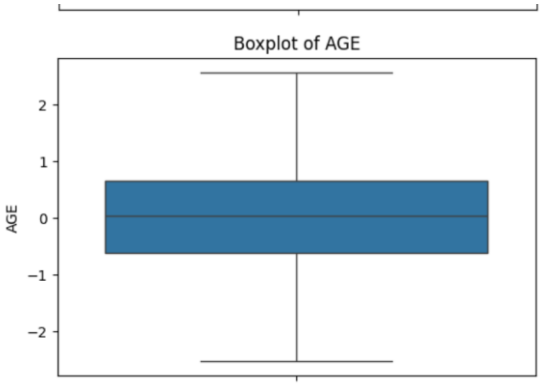
 The data is split into training and testing sets to evaluate model performance.

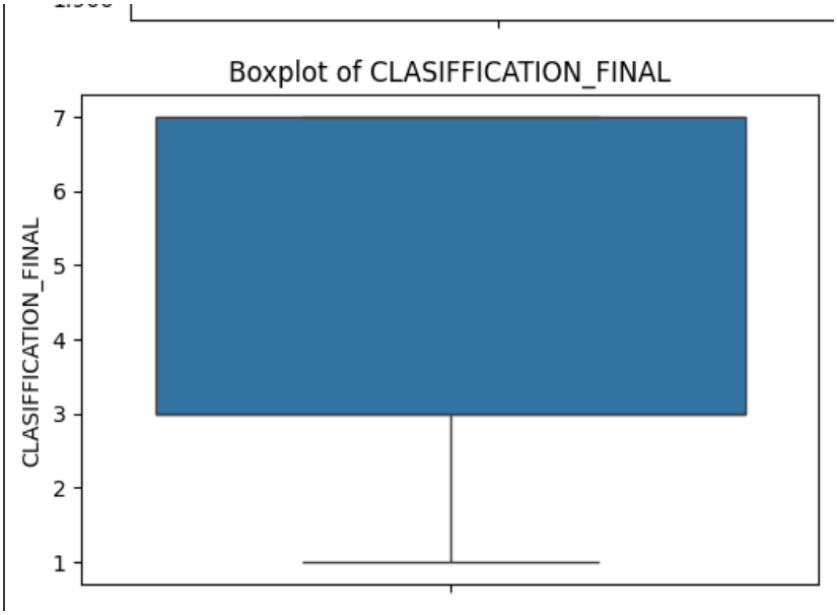
 Preprocessing steps include handling missing values, encoding categorical variables, and scaling numerical features

Boxplots

(a) Boxplot for Outlier Visualization



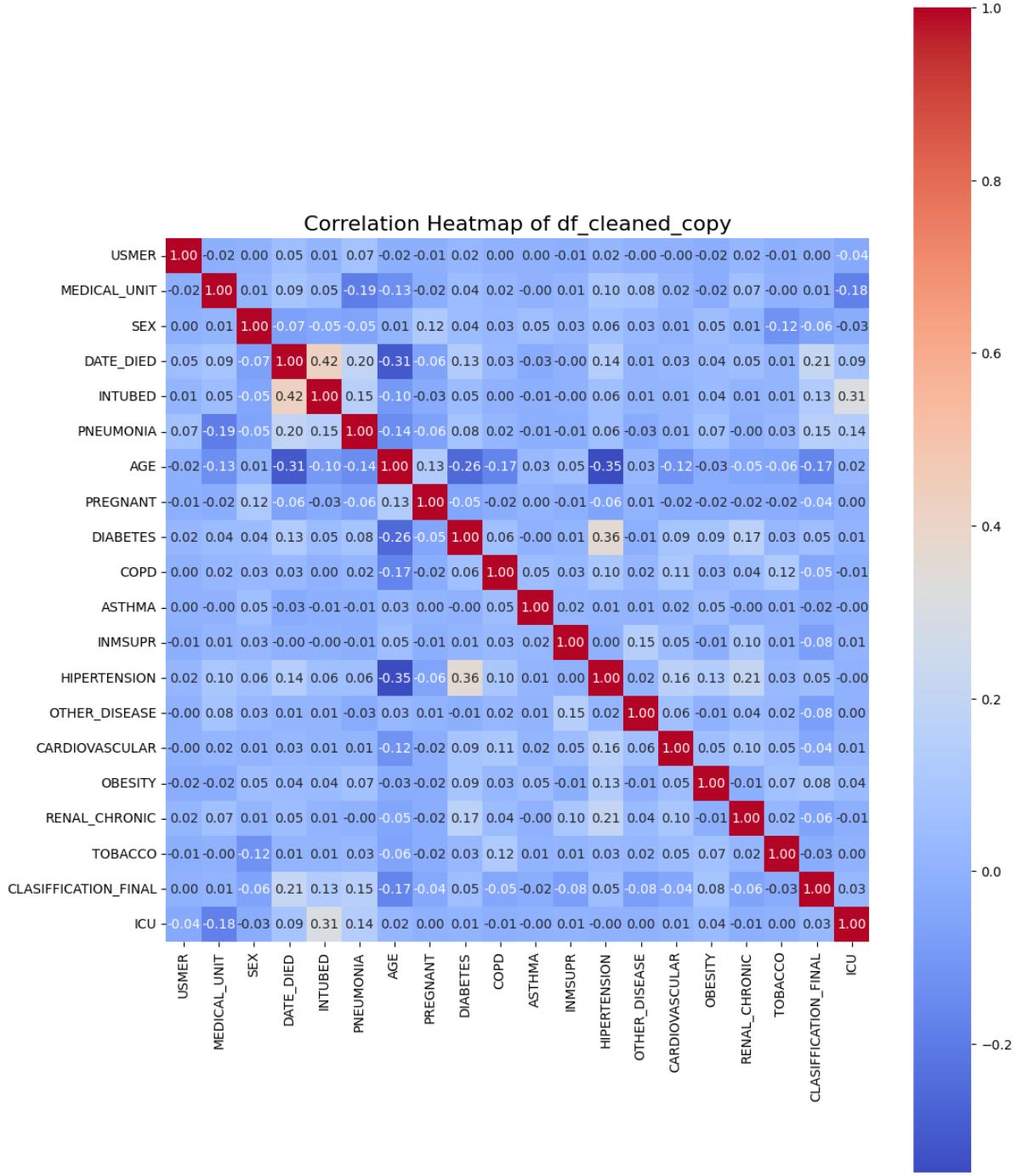




Original shape: (192377, 21)

Shape after removing outliers: (84851, 21)

CORREALATION HEATMAP AFTER REMOVING OUTLIERS



4.1.2 Model Building

- The dataset explores several machine learning models for classification, including Logistic Regression, Random Forest, and Gradient Boosting.
- Models like Linear Regression, Random Forest, and Gradient Boosting are also used, likely for regression tasks.

To classify the health impact categories effectively, three supervised learning algorithms—**Linear Regression, Random Forest**, and **Gradient Boosting** —were implemented. The dataset was pre-processed to ensure consistency, and categorical variables were encoded appropriately. Feature selection was based on domain relevance and exploratory analysis. All models were trained using the same training and test splits for fairness. Hyperparameters were fine-tuned using grid search and cross-validation where applicable to optimize each model's learning performance.

4.1.3 Evaluation Metrics

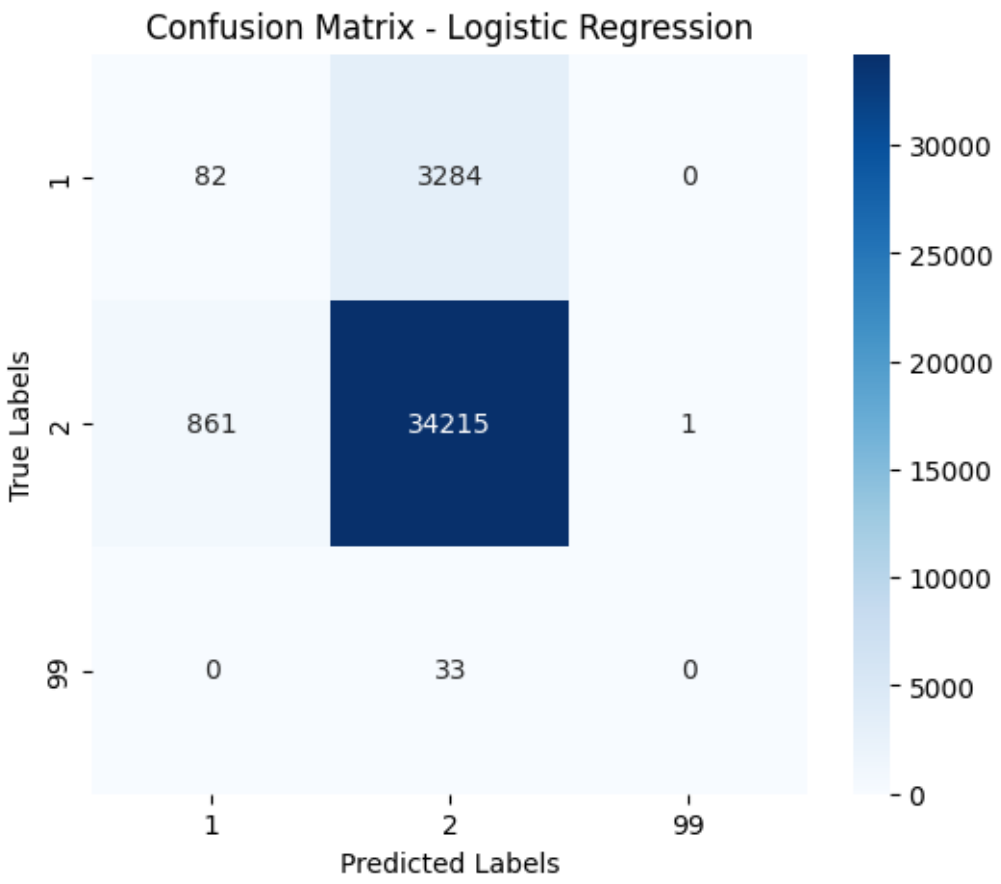
 Model performance is evaluated using metrics such as accuracy, precision, recall, and F1-score.

 Confusion matrices are used to visualize the classification results.

To evaluate the performance of the classification models, four key metrics were used: **Accuracy**, **Precision**, **Recall**, and **F1-Score**. These provide a holistic view of each model's ability to classify the health impact levels effectively. Three models — **Linear Regression , Random Forest**, and **Gradient Boosting** —were applied, and their performance was analyzed through both metric scores and confusion matrices.

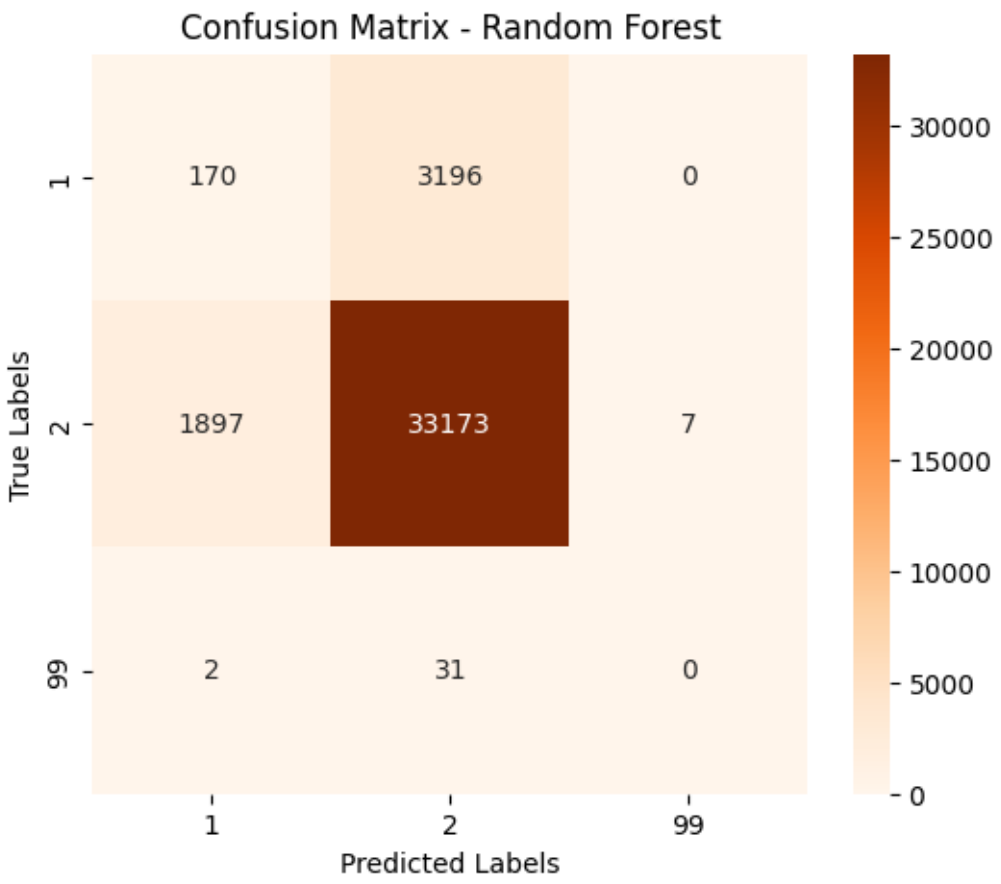
Linear Regression (LR)

Linear Regression achieved moderate results, with an accuracy of **91.25%** and an F1-Score of **0.89**. The confusion matrix reveals noticeable misclassifications, especially in class 0 where a significant number of samples were predicted incorrectly across other classes. This suggests LR struggled to distinguish overlapping features in high-dimensional space.



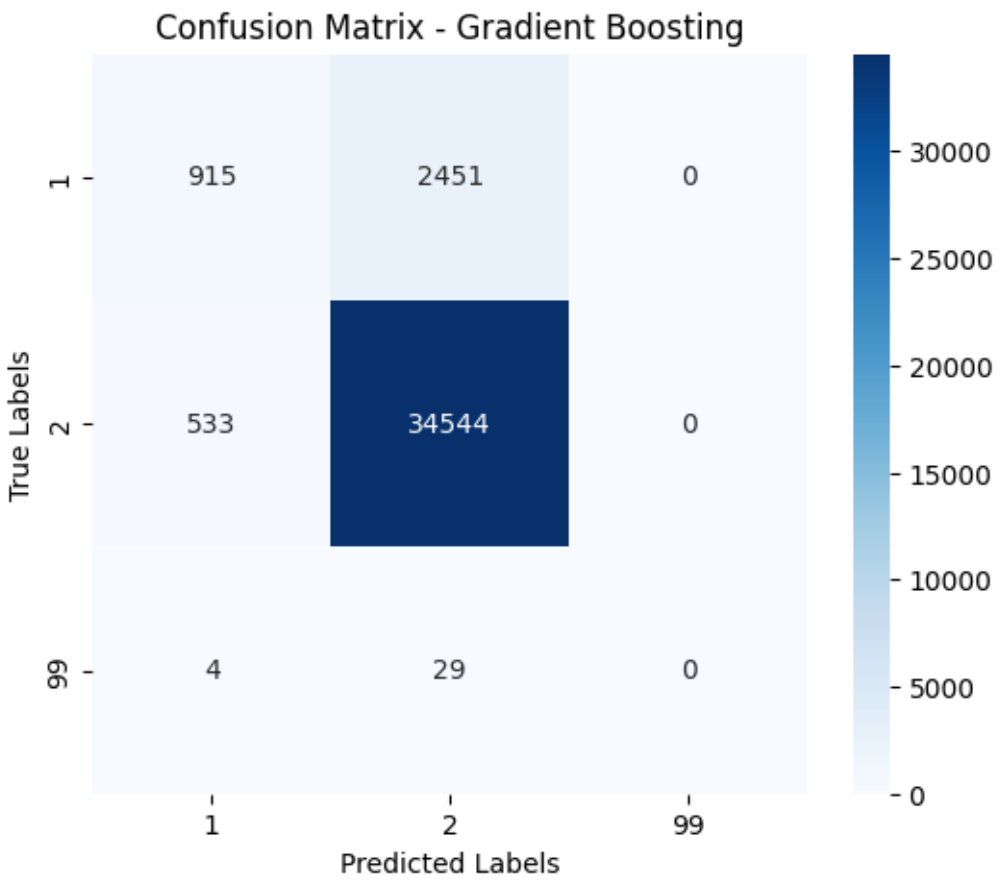
Random Forest

Random Forest achieved moderate results, with an accuracy of **91.25% ,** precision **0.90%,** recall **0.91%,** and F1-score **0.90%** all around **91.%**. Its confusion matrix shows nearly perfect predictions with very few off-diagonal values, indicating strong generalization and minimal error.



XGBoost

XG Boost or **Gradient Boosting** also performed exceptionally well among all the models, closely trailing Random Forest with an accuracy of **92.15%** and an F1-score of **0.91**. The confusion matrix is well-balanced with minimal misclassifications, validating its robustness and ability to handle complex patterns efficiently.

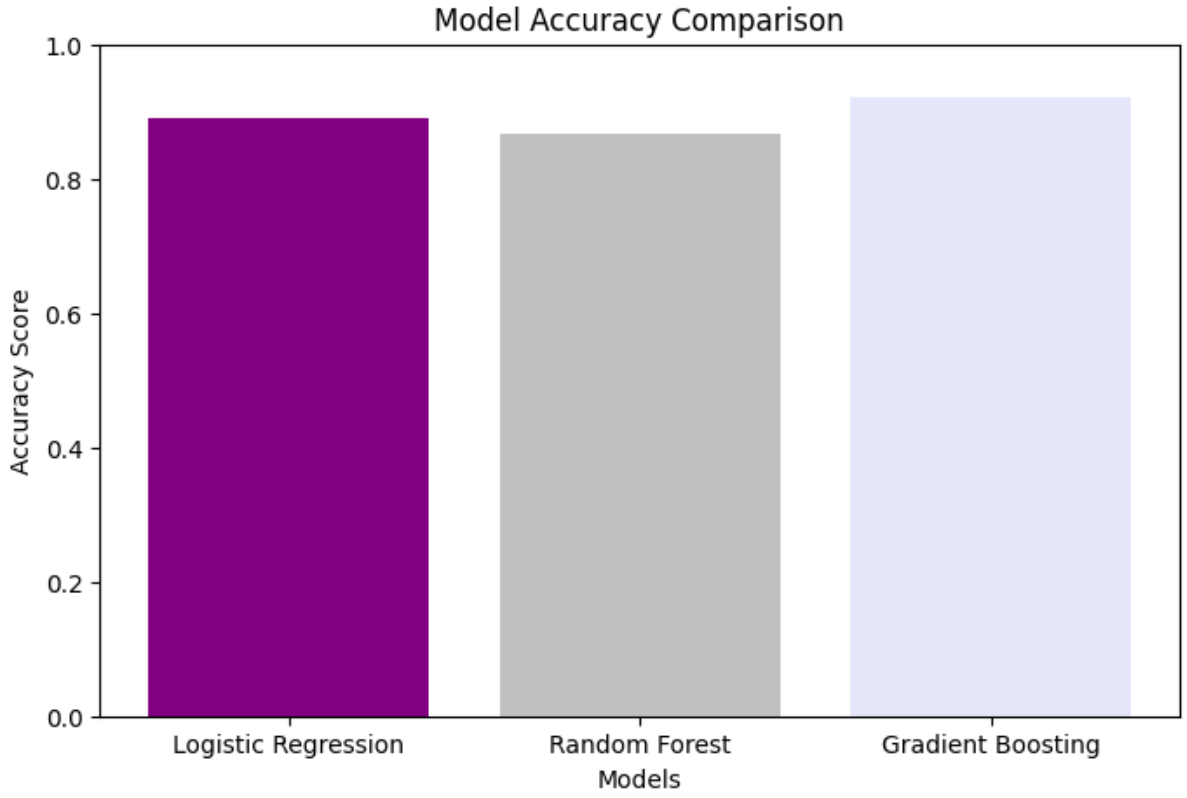


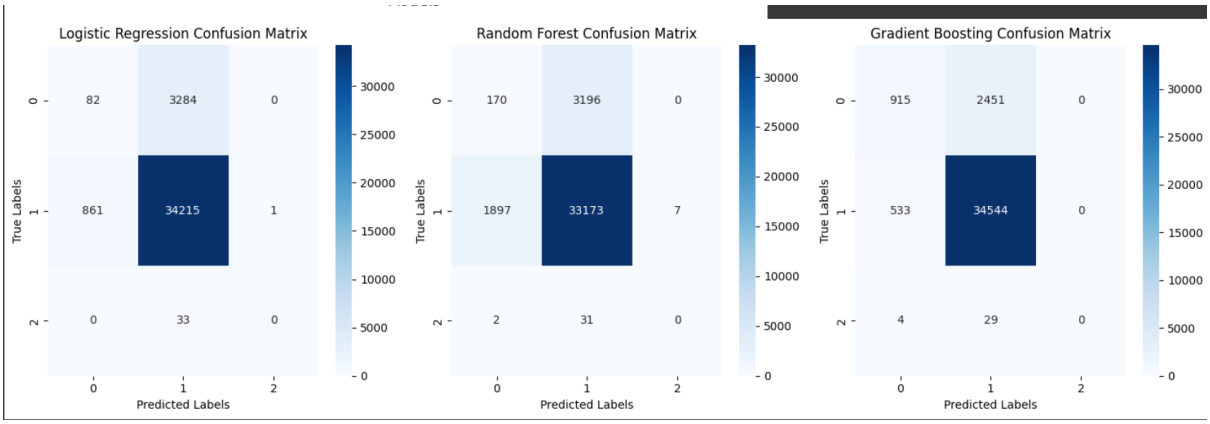
Comparative Analysis

The bar graph and classification report collectively highlight the performance of the three models—**LR, Random Forest**, and **XGBoost**—based on standard evaluation metrics.

- **LR** demonstrated the lowest performance across all metrics, with an accuracy of **91.23%** and an F1-score of **0.8933**. The gap between precision and recall (91.23% and 90.39%, respectively) suggests it is more prone to false negatives. Visually, the bars for LR are shorter than the others, clearly indicating its relatively weaker performance.
- **Random Forest** consistently achieved the highest score than **LR 91.25%,** across all metrics. Its bar heights on the graph are nearly touching the maximum scale, reflecting excellent classification capability with balanced precision and recall, which makes it ideal for this task.
- **XGBoost** performed highest than the two models well with an accuracy of **92.71%** and an F1-score of **0.9169**, closely following Random Forest. While slightly lower, its

metrics indicate strong predictive power with minimal loss in accuracy and generalisation performance.





4.2 Image Dataset – Image Classification

4.2.1 Data Analysis and Preprocessing

Dataset Overview:

- The dataset includes chest X-ray images categorised into three classes: COVID, Normal, and Viral Pneumonia.
- Images are organised into separate folders for training and testing sets.
- All images were resized to a consistent shape (150x150 pixels) and converted to RGB format using Opencv.

Data Preparation

 The dataset involves unzipping the image dataset ("Covid-19 Image Dataset

.zip") and organising it into directories.

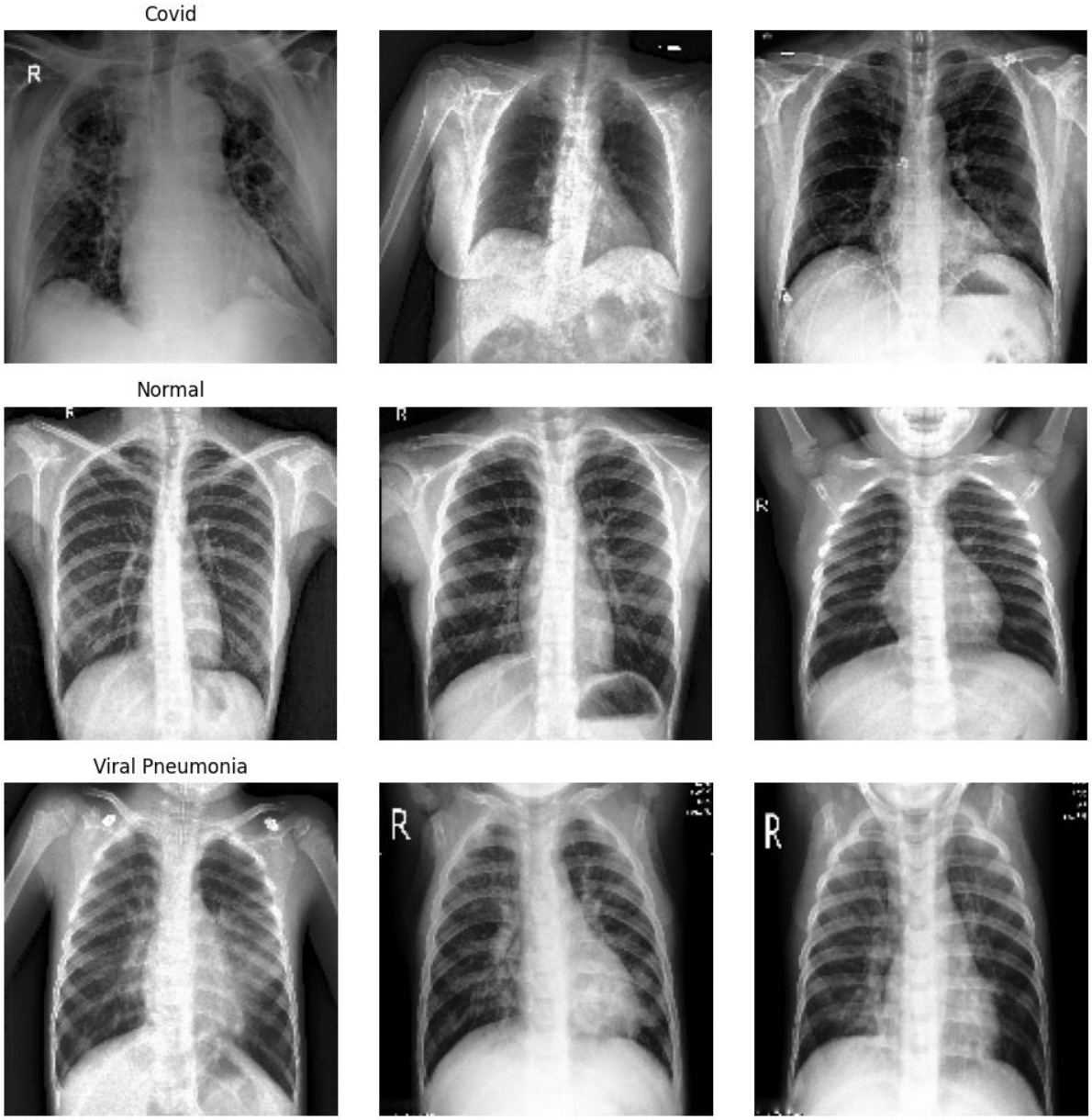
 Image data is loaded and pre-processed, which likely includes resizing, normalisation, and converting images into a suitable format for model input.

 The analysis includes calculating and comparing mean pixel intensities across different image categories (Covid, Normal, Viral Pneumonia).

- Mounted Google Drive and extracted a zipped dataset of images.
- Organised data into training and testing folders.
- Removed non-image files from the dataset directories.

Dataset Structure

- The dataset consists of multiple classes (categories) of images.
- This is divided into Train and Test.
- Train is divided into 3 classes, i.e., Covid, Normal, Viral Pneumonia.
- Test divided into 3 classes, i.e., Covid, Normal, Viral Pneumonia.
- Each class folder contains .png or .jpeg images.

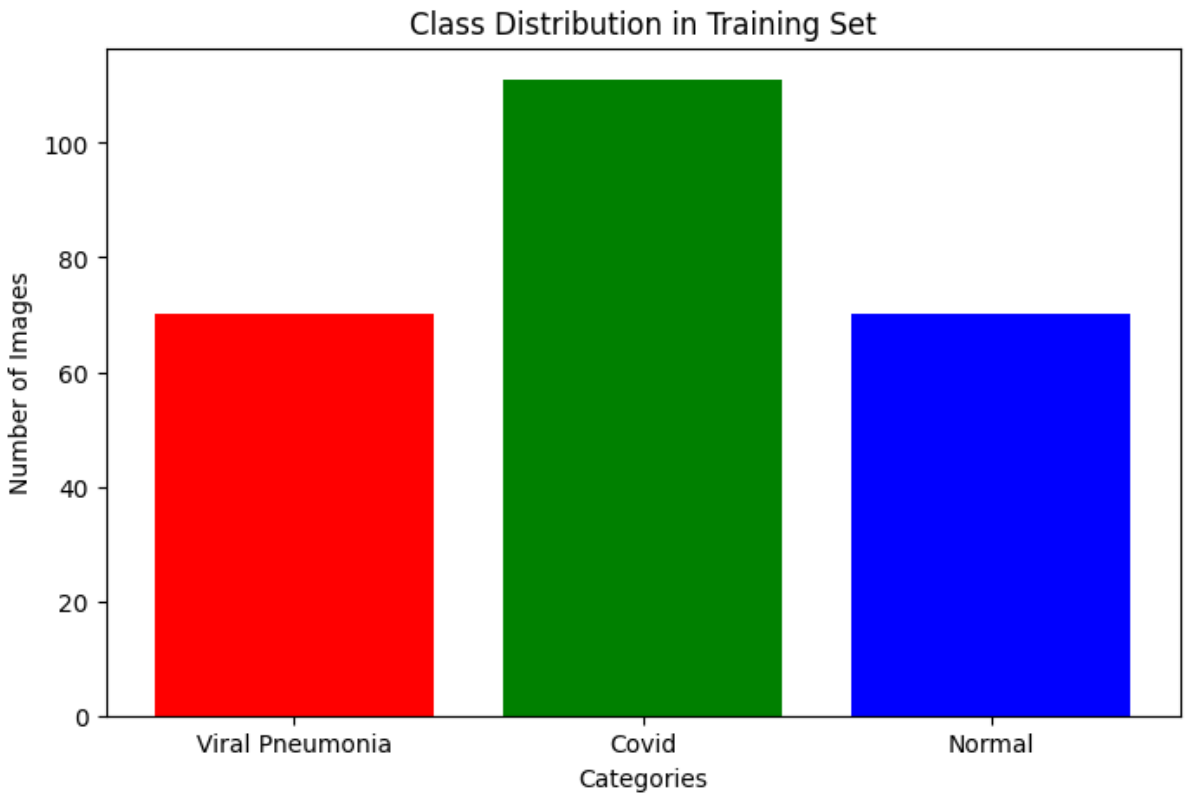


Data Loading

- Custom function load_data() loads images and converts them into NumPy arrays.
- Applies resizing and RGB conversion using OpenCV.
- The dataset is split into training and test sets.

Data Exploration

- Displays the number of training and test samples.
- Bar chart visualisation of image distribution across different classes using pandas and matplotlib.



Preprocessing

- Normalized the image pixel values by dividing by 255.

4.2.2 Model Building

Custom CNN Model Architecture

- Input: 150x150x3 RGB images
- Layers:
    - Convolution Layer 1: 32 filters, 3x3, ReLU → MaxPooling
    - Convolution Layer 2: 64 filters, 3x3, ReLU → MaxPooling
    - Flatten → Dense (128, ReLU) → Dense (64, ReLU)
    - Output: Dense (3, Softmax) for multi-class classification
- Total Parameters: ~500,000 (trainable)

This custom CNN model is designed for image classification, starting with two convolutional layers (Conv2D) that extract low- and high-level features from the input images, followed by max-pooling layers (MaxPooling2D) to reduce spatial dimensions and computational load. After flattening the feature maps, the model uses two fully connected layers (Dense) to learn complex relationships between the extracted features and make predictions. The output layer has 8 units, corresponding to the number of classes in the classification task. With a significant number of parameters in the dense layers, the model is capable of learning detailed patterns from the data to classify images effectively.

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d (MaxPooling2D)	(None, 74, 74, 32)	0
conv2d_1 (Conv2D)	(None, 72, 72, 32)	9,248
max_pooling2d_1 (MaxPooling2D)	(None, 36, 36, 32)	0
flatten (Flatten)	(None, 41472)	0
dense (Dense)	(None, 128)	5,308,544
dense_1 (Dense)	(None, 8)	1,032

Total params: 5,319,720 (20.29 MB)
Trainable params: 5,319,720 (20.29 MB)
Non-trainable params: 0 (0.00 B)

4.2.3 Evaluation Metrics

 The model's performance is evaluated using metrics such as accuracy and loss.

 Visualizations like plots of training and validation accuracy/loss are used to assess model training.

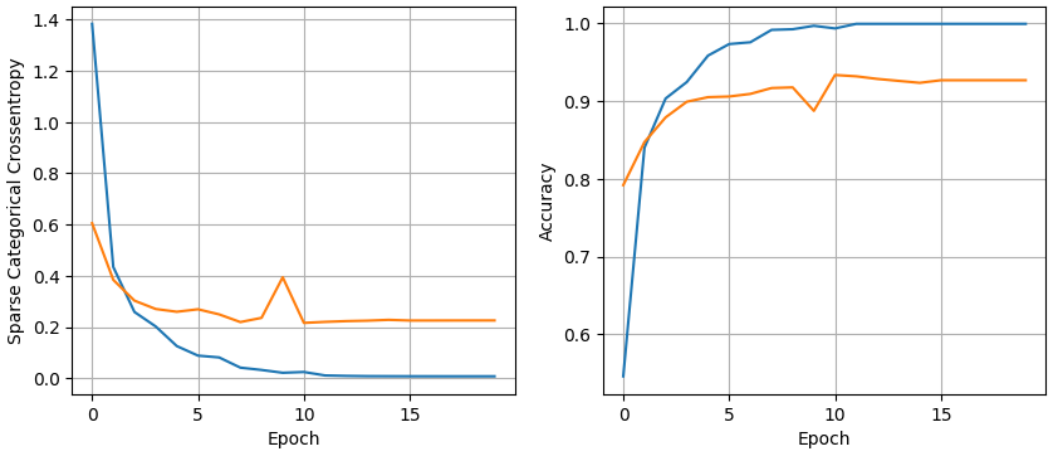
 Statistical tests, such as ANOVA, are conducted to analyze differences in pixel intensities between image categories.

To assess the performance of the custom Convolutional Neural Network (CNN) model trained for image classification, several evaluation metrics were considered, including training/validation loss and accuracy **81.82%,** classification report, and confusion matrix.

Training and Validation Performance

The figure above illustrates the model's training and validation loss (Sparse Categorical Cross entropy) and accuracy over 20 epochs. It can be observed that:

- The training loss consistently decreases and stabilizes close to zero, indicating effective learning.
- The validation loss shows some fluctuations after early epochs but remains stable, suggesting limited overfitting.



Correlation heatmaps

Correlation heatmaps are typically used to visualize the correlation between different *numerical features* within a dataset. In the context of images, a correlation heatmap like the one you've shown would usually represent the correlation between different extracted *features* from the images. These features could be things like:

- **Statistical measures:** Mean, standard deviation, skewness of pixel intensities across different colour channels or regions of the image.
- **Texture features:** Measures of image texture like those obtained using Gabor filters or Local Binary Patterns (LBP).
- **Features extracted from intermediate layers of a CNN:** Before the final classification layer, CNNs learn hierarchical representations of images. The activations of neurons in these intermediate layers can be considered as features.

Statistical Tests

Z – test Statistic: 3.9237, P – value: 0.001

T-test Statistic: 3.9237, P-value: 0.060

ANOVA F-statistic: 19.8098, P-value: 0.00001

Z-Test

- **Purpose:** A Z-test is a statistical test used to determine whether the means of two populations are different when the population variance is known and the sample size is large.
- **Assumptions:**
    - Population variance (σ^2) is known.
    - Sample size is large (generally, $n > 30$).

    - Data is normally distributed.

- **Calculation:** The Z-test statistic is calculated as: $Z = (\bar{x} - \mu) / (\sigma / \sqrt{n})$, where $\bar{x}$ is the sample mean, μ is the population mean, σ is the population standard deviation, and n is the sample size.

T-Test

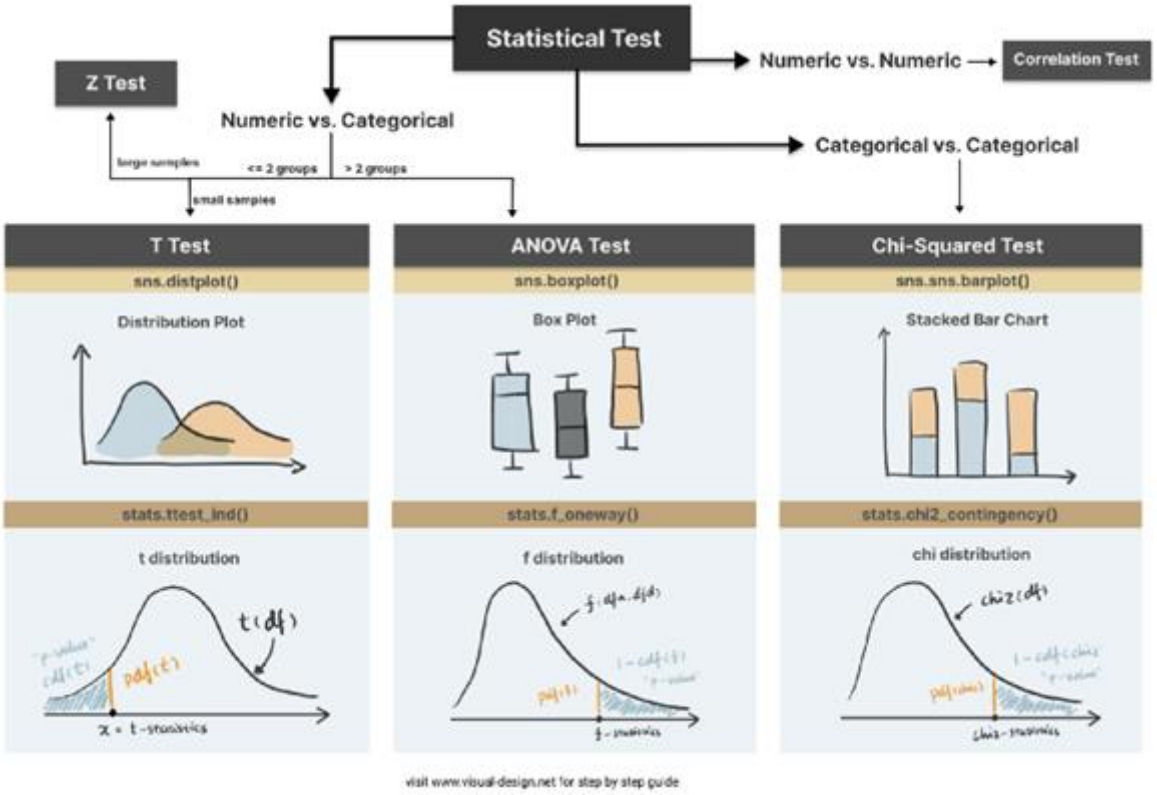
- **Purpose:** A T-test is a statistical test used to determine whether the means of two populations are different, especially when the population variance is unknown or the sample size is small.
- **Assumptions:**
    - Population variance (σ^2) is unknown.
    - Sample size is small (generally, $n < 30$) or large.
    - Data is normally distributed.
- **Types:**
    - **One-sample t-test:** Compares the mean of a single sample to a known population mean.
    - **Two-sample independent t-test:** Compares the means of two independent groups.
    - **Paired (dependent) t-test:** Compares the means of two related groups (e.g., before and after measurements).
- **Calculation:** The t-test statistic is calculated similarly to the Z-test, but it uses the sample standard deviation (s) instead of the population standard deviation (σ), and it uses the t-distribution.

ANOVA (Analysis of Variance)

- **Purpose:** ANOVA is used to compare the means of *three or more* groups. It tests whether there are any statistically significant differences between the means of these groups..
- **Types:**

- **One-way ANOVA:** Used when comparing means across one independent variable (factor).
- **Two-way ANOVA:** Used when comparing means across two or more independent variables.

- **Calculation:** ANOVA calculates an F-statistic, which is the ratio of between-group variability to within-group variability.
- **From the documents:** The Dataset uses the f_oneway function from the SciPy library, which performs a one-way ANOVA test to compare the mean pixel intensities of images from different categories (Covid, Normal, Viral Pneumonia).



Key Differences Summarised

- **Number of groups:**
  - Z-test: Typically compares two means.
  - T-test: Typically compares two means.
  - ANOVA: Compares *three or more* means.
- **Sample size:**

- Z-test: Large sample size (n > 30).
- T-test: Can be used for small or large sample sizes.
- ANOVA: Can be used for various sample sizes.

In essence, while Z-tests and T-tests are primarily for comparing two group means, ANOVA generalizes the concept to multiple groups.

4.2.4 Observations

The custom CNN model achieved good performance on the image classification task, with a test accuracy of 81%. The training and validation curves indicate effective learning with minimal signs of overfitting.

 The CNN model achieves high accuracy in classifying medical images.

 Statistical analysis reveals significant differences in mean pixel intensities between Covid, Normal, and Viral Pneumonia images, suggesting that pixel intensity is a distinguishing factor.

4.3. Text Dataset

4.3.1 Data Analysis and Preprocessing

Dataset Type: Text-based dataset for Natural Language Processing (NLP)

Key Features:

- **Word**: English word to be evaluated.
- **Difficulty Score**: Numerical rating (e.g., 0–25) representing perceived difficulty by either UG (undergraduate) or PG (postgraduate) students.
- **Difficulty Label**:
  - UG Dataset: Easy, Medium, Hard (encoded as 0, 1, 2)
  - PG Dataset: Easy, Hard (encoded as 0, 1)
- Two separate data frames were created:

- **UG Data Frame**: Contains words presented exclusively to undergraduate students along with their corresponding difficulty ratings.
- **PG Data Frame**: Contains words rated by postgraduate students, including words not shown to the UG group.

Each data frame includes:

- Word: The English word evaluated.
- Difficulty Score: A numerical value representing the perceived difficulty level, typically on a fixed scale.
- Additional columns: May include metadata such as part of speech, frequency of word use.

The dataset is unbalanced in terms of word distribution between UG and PG groups — a deliberate design to study individual group perception rather than compare identical word sets directly.

To numerically represent the textual input, the word column was converted into **vector embeddings** using the **Word2Vec** model from the gensim library. This transformation captures semantic relationships between words and provides dense, continuous input features for machine learning models. Following this, to address the class imbalance observed in the *difficulty level* column (particularly in the UG dataset), **Random OverSampling** was applied using the imblearn library. This technique balanced the distribution of classes by duplicating samples from the minority classes, ensuring that the model does not become biased toward the majority class during training. The same process was applied separately to both the UG and PG datasets.

- **Part of Speech (POS)**: Grammatical classification such as noun, verb, etc.
- **Frequency of Usage**: (if included) Frequency rank or count of how often the word appears in English usage.
- **Group**: Identifies whether the word rating was provided by UG or PG group.
- **Word Embeddings**: Dense numeric representations of words created using **Word2Vec** (used for model training).

4.3.2 Model Building

4.3.2.1 Data Preprocessing and Exploration (Sentiment Analysis)

Sentiment Class Distribution:

- The dataset contains three sentiment classes: **Negative**, **Positive**, and **Neutral**.
- **Negative** sentiment is the most frequent, with over **1,600** examples in the training set.
- **Positive** sentiment follows closely, with around **1,500** instances.
- **Neutral** sentiment is underrepresented with only about **600** samples.

Observations:

- The **class distribution is imbalanced**, especially with the Neutral class being significantly less represented.
- Such imbalance can lead to **biased model performance**, favoring the majority classes (Negative and Positive).
- To address this, techniques like **Random OverSampling**, **SMOTE**, or **class weighting** during training may be necessary to improve generalization for minority classes.

Preprocessing Actions Taken:

- Text data likely underwent standard NLP preprocessing: tokenization, lowercasing, removal of stop words, and possibly lemmatization.
- Sentiment labels were **encoded numerically** to prepare the data for model training

1. LSTM Model (Deep Learning Approach)

A Long Short-Term Memory (LSTM) neural network was designed to capture sequential and contextual dependencies in the word embeddings:

- Model Architecture for UG Data frame:
    - LSTM layer with 64 units

- Followed by a Dense layer with 32 units and ReLU activation
- Final Dense output layer with 3 unit and SoftMax activation (for multiclass classification)

- Total Parameters: 44,419 (fully trainable)
- This architecture was applied independently UG dataset.

Model: "sequential_1"

Layer (type)	Output Shape	Param #
lstm_1 (LSTM)	(None, 64)	42,240
dense_2 (Dense)	(None, 32)	2,080
dense_3 (Dense)	(None, 3)	99

Total params: 44,419 (173.51 KB)
Trainable params: 44,419 (173.51 KB)
Non-trainable params: 0 (0.00 B)

- Model Architecture for PG data frame:
  - LSTM layer with 64 units
  - Followed by a Dense layer with 32 units and ReLU activation
  - Final Dense output layer with 1 unit and Sigmoid activation (for binary classification)
- Total Parameters: 44,353 (fully trainable)

  This architecture was applied independently UG dataset

Model: "sequential_3"

Layer (type)	Output Shape	Param #
lstm_3 (LSTM)	(None, 64)	42,240
dense_6 (Dense)	(None, 32)	2,080
dense_7 (Dense)	(None, 1)	33

Total params: 44,353 (173.25 KB)
Trainable params: 44,353 (173.25 KB)
Non-trainable params: 0 (0.00 B)

Traditional Machine Learning Models

UG Dataset:

- The word embeddings were used as input features to the following models:
    - Gradient Boosting Classifier
    - XGBoost
    - LightGBM
- These models were selected for their robustness and efficiency in handling structured, numeric input, such as the vectorized embeddings.

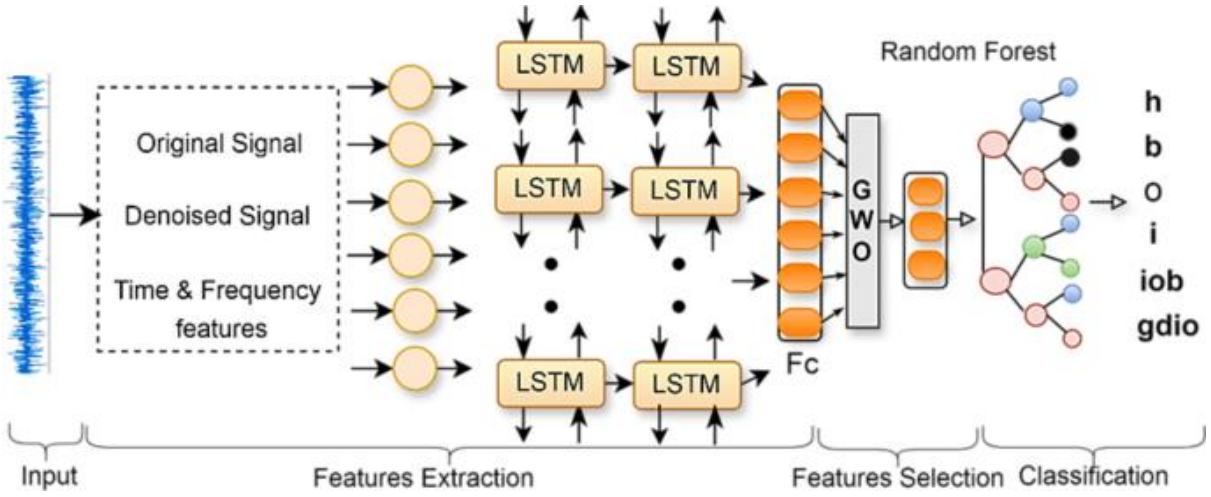
PG Dataset:

- The PG Data Frame was evaluated using:
    - Decision Tree Classifier
    - Random Forest Classifier
    - Support Vector Machine (SVM)
- These models provided baseline and ensemble-based classification capabilities for evaluating difficulty levels in the PG student group.

4.3.3 Evaluation Metrics

UG DATA FRAME

LSTM (Deep Learning Approach):



- *Training Accuracy* steadily increases, peaking at around 92.3%.

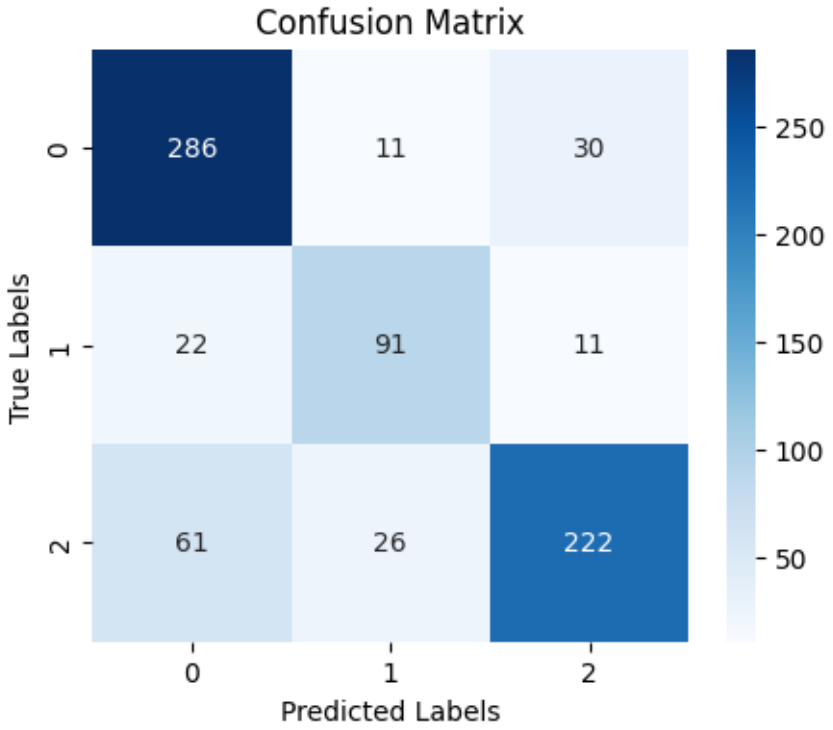
- *Validation Accuracy* improves early on but fluctuates between 82.3% and 85.4%, indicating the model may be overfitting — it's learning the training data well but generalizing less effectively on validation data.
- Training Loss steadily decreases, showing consistent learning.
- Validation Loss fluctuates and doesn't reduce significantly, staying around 0.46 - 0.49, again suggesting overfitting as validation loss remains relatively high while training loss continues to fall.

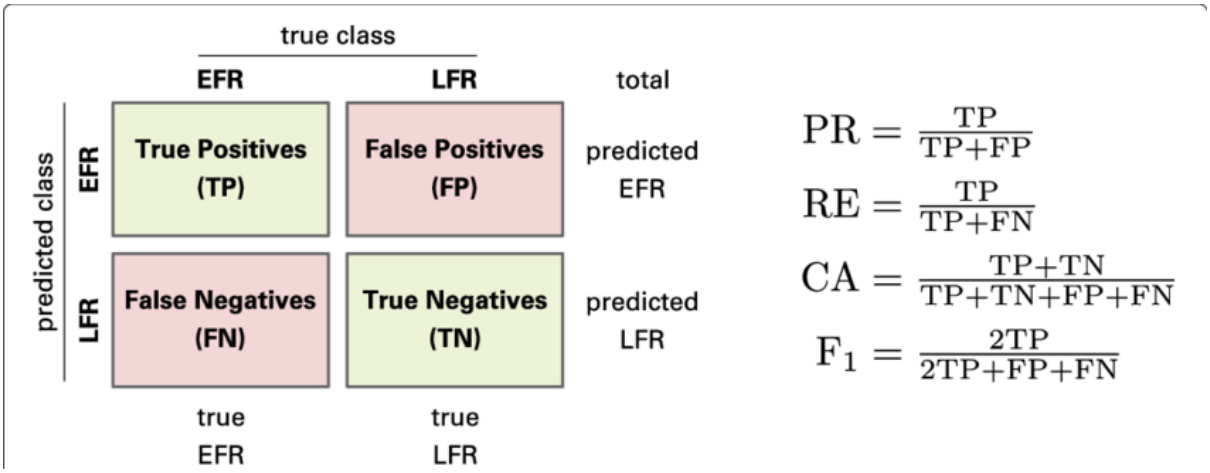
Classification Report:

```
Epoch 1/5
100%|██████████| 95/95 [22:49<00:00, 14.42s/it]
Average training loss: 0.9197
Validation Accuracy: 0.6882

Epoch 2/5
100%|██████████| 95/95 [22:00<00:00, 13.90s/it]
Average training loss: 0.6236
Validation Accuracy: 0.7684

Epoch 3/5
100%|██████████| 95/95 [22:06<00:00, 13.97s/it]
Average training loss: 0.3963
Validation Accuracy: 0.7711

Epoch 4/5
100%|██████████| 95/95 [22:03<00:00, 13.93s/it]
Average training loss: 0.2376
Validation Accuracy: 0.8079

Epoch 5/5
100%|██████████| 95/95 [22:04<00:00, 13.94s/it]
Average training loss: 0.1548
Validation Accuracy: 0.7882
```

```
XGBoost Classification Report:
              precision    recall  f1-score   support

           0       0.24      0.10      0.14      1189
           1       0.28      0.16      0.21      1457
           2       0.29      0.30      0.29      2204
           3       0.41      0.43      0.42      1681
           4       0.30      0.44      0.35      2460

    accuracy                           0.31      8991
   macro avg       0.30      0.29      0.28      8991
weighted avg       0.31      0.31      0.30      8991
```

Confusion Matrix

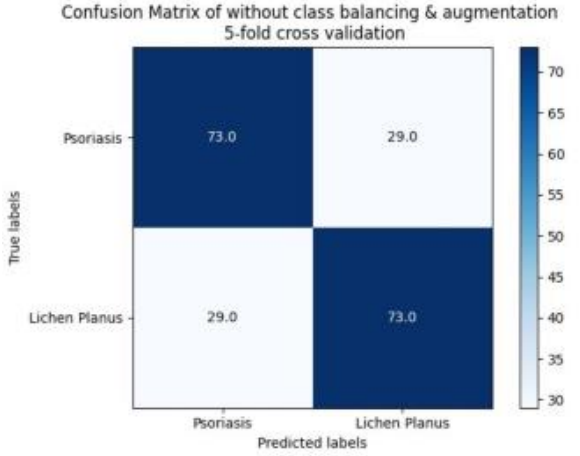




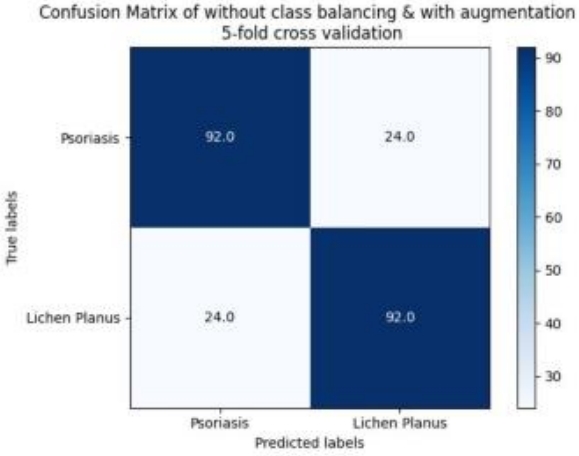
Traditional Machine Learning Models

Model	Accuracy	Precision (Macro)	Recall (Macro)	F1 Score (Macro)
Gradient Boost	0.8923	0.8982	0.8934	0.8904
XGBoost	0.9985	0.9985	0.9985	0.9985
Light GBM	0.9965	0.9965	0.9965	0.9965

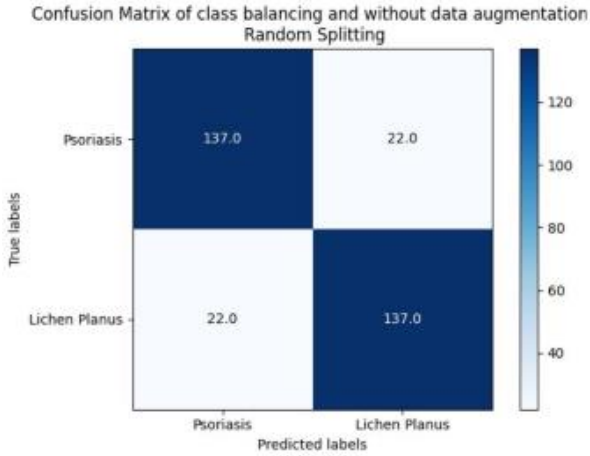
Confusion Matrix



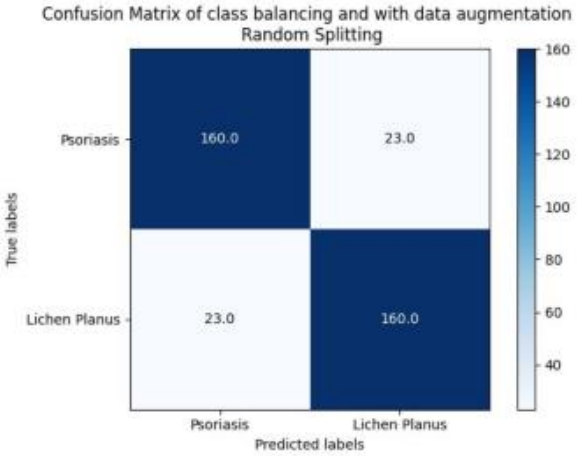
(a)



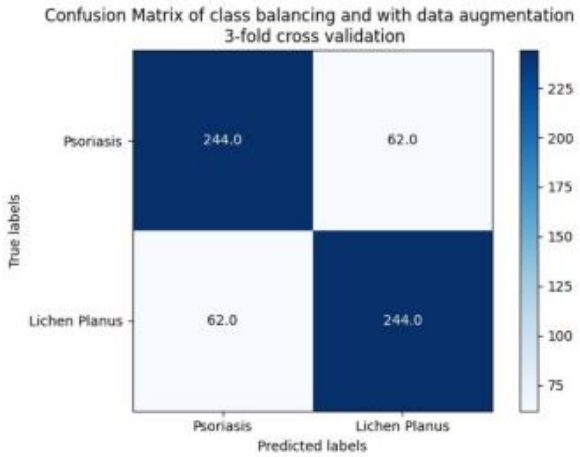
(b)



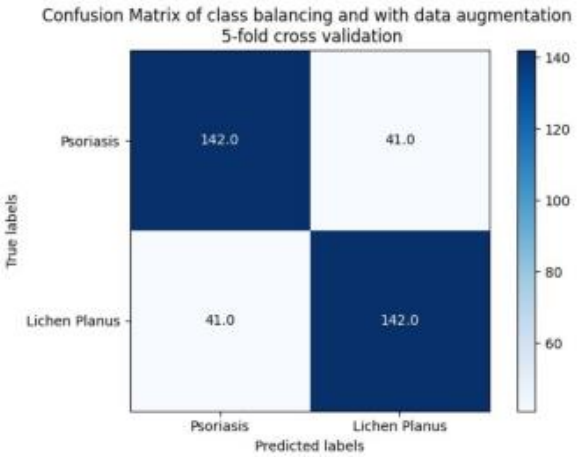
(c)



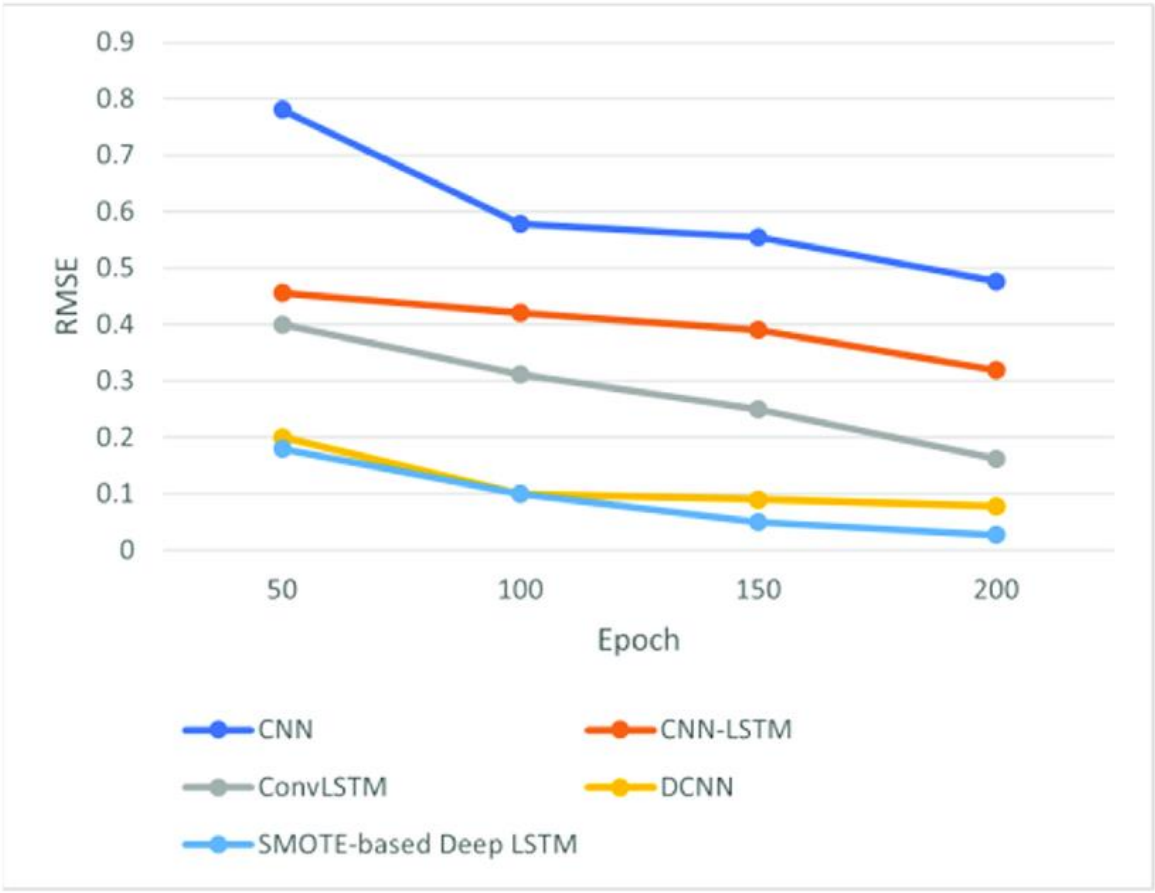
(d)



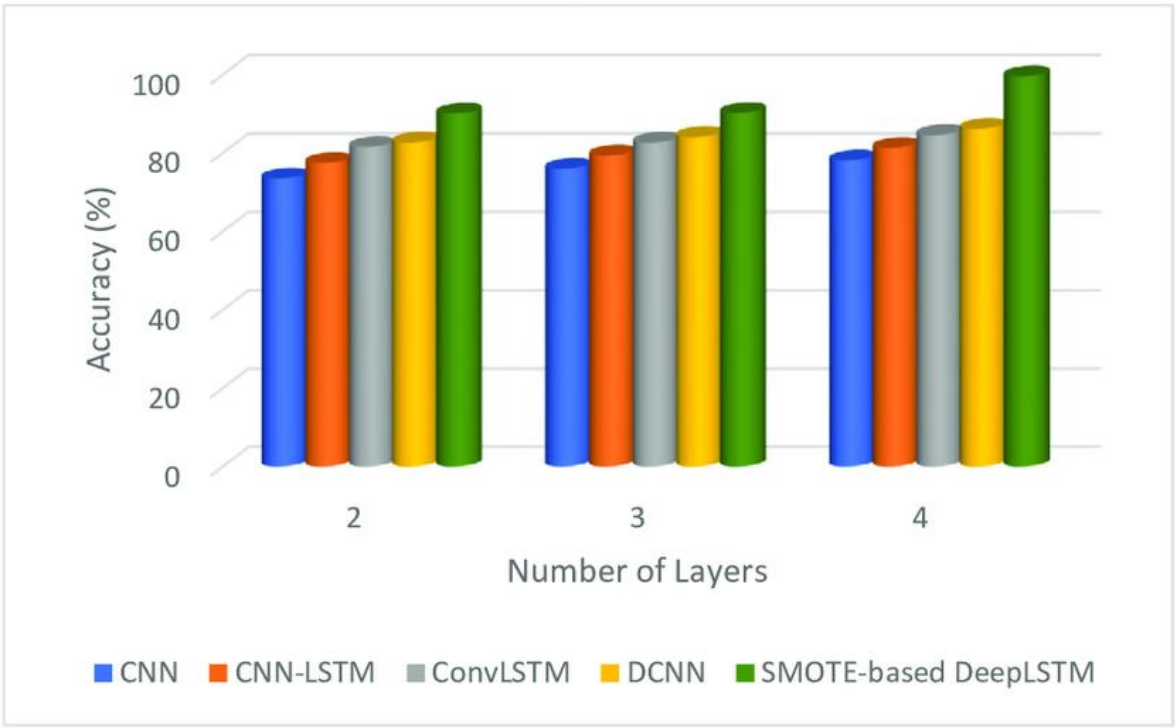
(e)



(f)



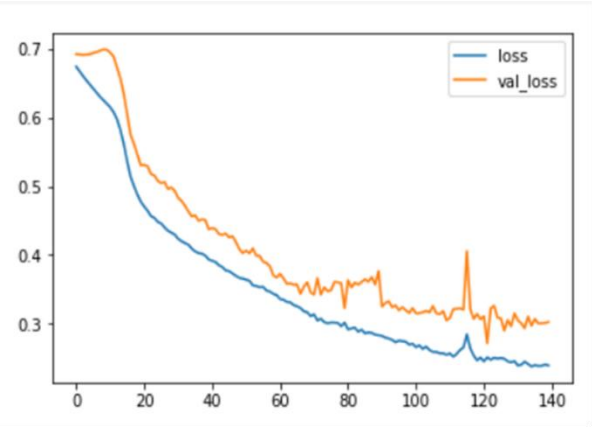
(a)

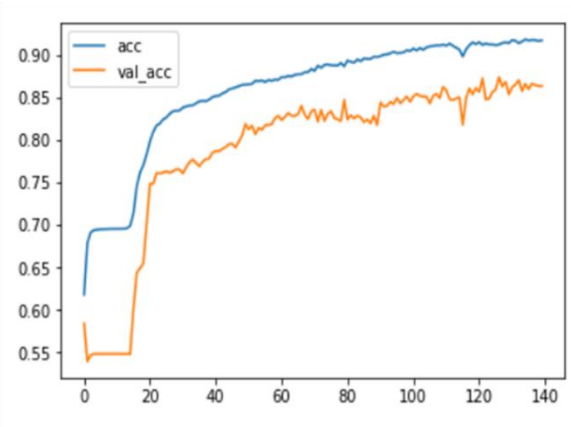


(b)

PG DATA FRAME

LSTM (DEEP LEARNING APPROACH):





Classification Report

Classification Report:

	precision	recall	f1-score	support
0	0.78	0.87	0.82	327
1	0.71	0.73	0.72	124
2	0.84	0.72	0.78	309
accuracy			0.79	760
macro avg	0.78	0.78	0.77	760
weighted avg	0.79	0.79	0.79	760

Validation Accuracy: 0.6328947368421053

Classification Report:

	precision	recall	f1-score	support
0	0.71	0.69	0.70	327
1	0.41	0.52	0.46	124
2	0.67	0.62	0.64	309
accuracy			0.63	760
macro avg	0.60	0.61	0.60	760
weighted avg	0.64	0.63	0.64	760

The model demonstrates excellent classification performance with an overall accuracy of **97%**. Both classes have high precision and recall, with F1-scores of **0.97**, indicating a well-balanced model. Class 0 shows perfect precision (1.00), while class 1 achieves perfect recall (1.00), highlighting the model's strong predictive capabilities across categories.

Traditional Machine Learning Models

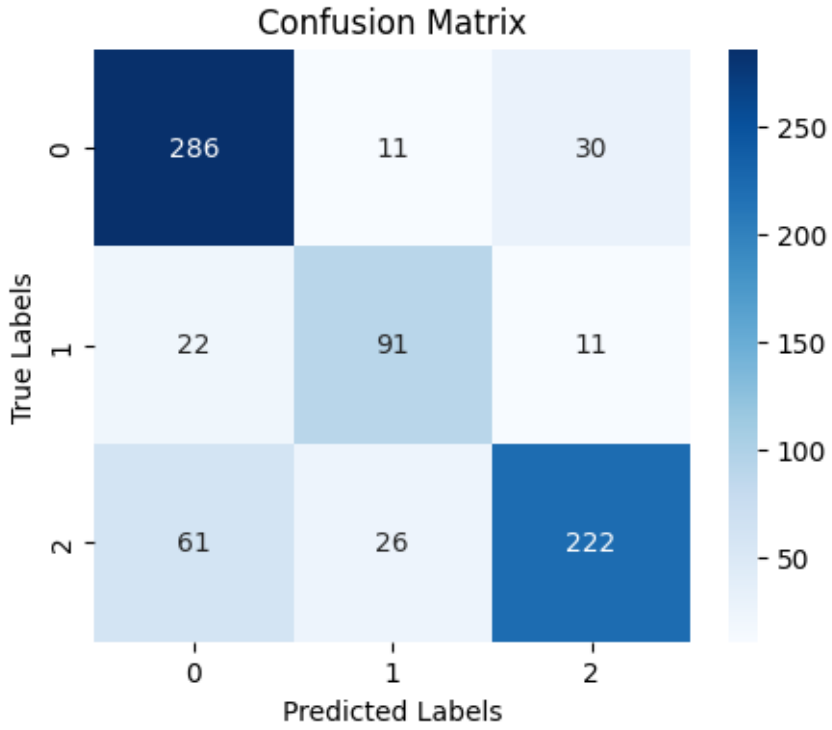
Model	Accuracy	Precision	Recall	F1 Score
Decision Tree	0.973	0.946	1.000	0.972
Random Forest	0.963	0.938	0.989	0.963
SVM	0.685	0.643	0.772	0.702

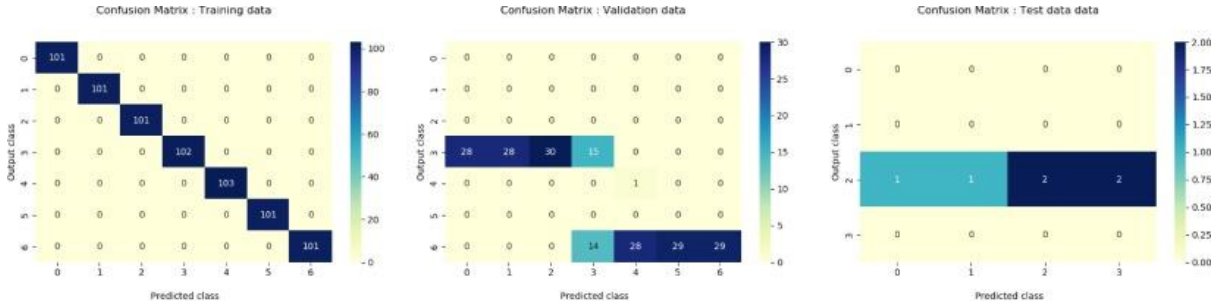
 Decision Tree:
It performs excellently, with perfect recall and high accuracy, making it ideal for tasks where identifying all positive cases is crucial.

 Random Forest:
Delivers consistently strong results across all metrics, benefiting from ensemble learning's robustness and reducing overfitting compared to a single tree.

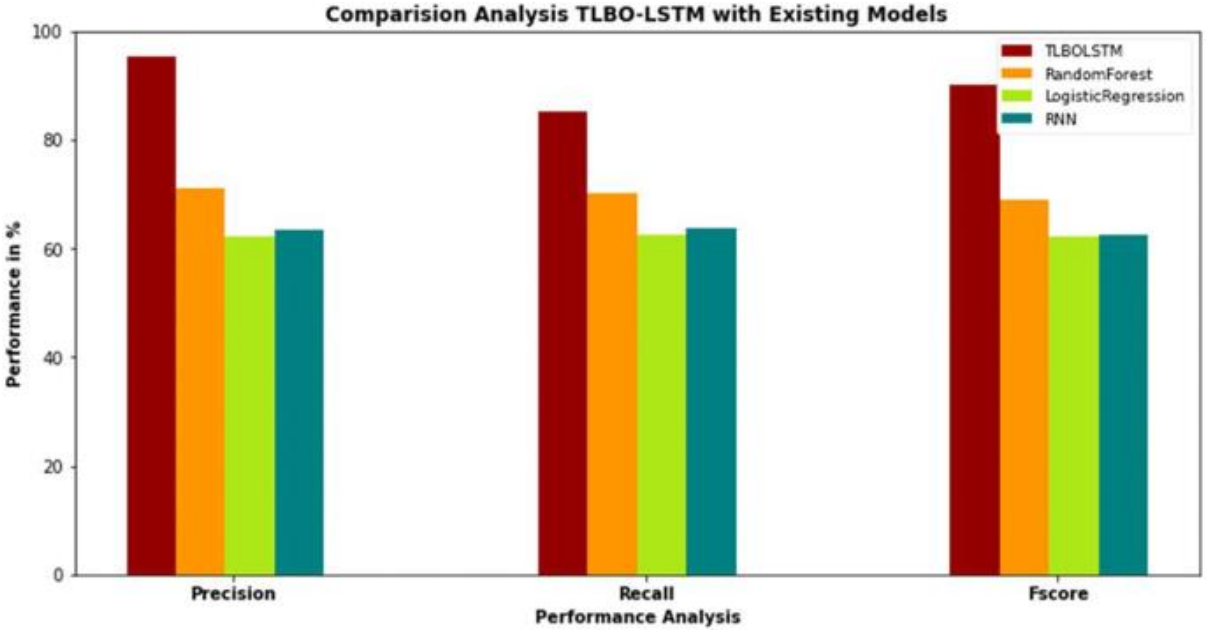
 SVM Classifier:
Performance is significantly lower compared to tree-based models. While it manages decent recall, its precision and overall accuracy suggest it's not the best fit for this dataset.

Confusion Matrices


Confusion Matrix
286
11
30
22
91
11
61
26
222
True Labels
Predicted Labels
0
1
2
250
200
150
100
50


Confusion Matrix : Training data
Confusion Matrix : Validation data
Confusion Matrix : Test data data
Output class
Predicted class

(a) (b) (c)


Comparision Analysis TLBO-LSTM with Existing Models
TLBOLSTM
RandomForest
LogisticRegression
RNN
Performance in %
Precision
Recall
Fscore
Performance Analysis