CAPSTONE PROJECT ON DATA ANALYSIS USING PYTHON

A Course Completion Report in partial

fulfillment of the degree

Bachelor of Technology

in

Computer Science & Artificial Intelligence

By

Roll. No :2203A54048 **Name**: PRAVALIKA REDDY.V

Batch No: 40

Guidance of -D.Ramesh

Submitted to

SCHOOL OF COMPUTER SCIENCE & ARTIFICIAL INTELLIGENCE

SR UNIVERSITY, ANANTHASAGAR, WARANGAL

April, 2025.

1. Anixety attack -Dataset

Title: Anxiety Attack Detection Using Word2Vec and Machine Learning Algorithms

2. Abstract

Mental health awareness has gained critical importance in recent years, with anxiety disorders being among the most prevalent conditions affecting individuals globally. This project aims to develop a text classification system to detect anxiety attacks based on user-generated textual data. To enhance semantic understanding, the project employs Word2Vec embeddings to transform textual content into dense vector representations that capture contextual and relational meaning between words.

These embeddings serve as input features for traditional machine learning algorithms such as Support Vector Machine (SVM), Random Forest, and XGBoost. The system follows a complete pipeline: data preprocessing, vectorization, model training, and evaluation. Preprocessing includes cleaning, tokenization, and stopword removal, while vectorization involves averaging Word2Vec embeddings for sentence-level representation.

Among the models tested, XGBoost achieved the highest accuracy, followed closely by SVM. The results confirm that using Word2Vec significantly enhances the model's ability to detect anxiety-related patterns in text. This study demonstrates the potential of embedding techniques combined with classical machine learning models in supporting early mental health detection through NLP.

3. Introduction

Anxiety attacks are a serious mental health concern, often expressed through written or spoken language. This project focuses on detecting anxiety from text using Natural Language Processing (NLP) techniques. Word2Vec embeddings are used to convert text into meaningful vectors that capture context and semantics. These vectors are classified using machine learning algorithms such as SVM, Random Forest, and XGBoost. The goal is to build an efficient system for early detection of anxiety through text classification.

4. Problem Statement

Mental health conditions such as anxiety and depression have become increasingly prevalent, especially in response to social, economic, and health-related stressors. Understanding the distribution and patterns of these conditions across different demographic groups and time periods is essential for targeted interventions.

5. Dataset Details

Format: CSV

Key Columns:

- **Indicator:** Severity of Anxiety Attack (1-10)
- **Group / Subgroup:** Gender, Age, Occupation, Family History of Anxiety, Smoking, Dizziness, Medication, Recent Major Life Event
- **Related Metrics:** Stress Level (1-10), Heart Rate (bpm during attack), Breathing Rate (breaths/min), Sweating Level (1-5), Therapy Sessions (per month), Diet Quality (1-10)
- **Value:** Severity of Anxiety Attack (1-10)

6. Methodology

- **Data Preprocessing:**
  - **Text Cleaning**: Removed punctuation, special characters, numbers, and converted text to lowercase.
  - **Tokenization**: Split sentences into individual words.
  - **Stopword Removal**: Common words like “and”, “the”, “is” were removed to retain only meaningful terms.

- **Word2Vec Embedding:**
  - **Pre-trained Word2Vec Model**: Loaded pre-trained vectors (e.g., Google News vectors) to capture contextual meaning from a large corpus.

- **Document Vector Formation**:
  - Each sentence was represented as the **average of the Word2Vec vectors** of its individual words.
  - This fixed-length document embedding was used as input for classification models.

- **Model Training:**

After vectorization, the document embeddings were split into **training and testing sets**. Multiple machine learning algorithms were then trained and evaluated:

- **Support Vector Machine (SVM)**
- **Random Forest**
- **XGBoost (Extreme Gradient Boosting)**
- **Logistic Regression** (optional baseline)

Each model was trained to classify whether a given text indicates an anxiety attack or not, based on the vectorized representation of the input sentence.

- **Model Evaluation:**

Models were assessed using standard performance metrics:

- **Accuracy**: Overall correct predictions
- **Precision**: Correct positive predictions among all predicted positives
- **Recall**: Correct positive predictions among all actual positives
- **F1-Score**: Harmonic mean of precision and recall

- **Visualization & Analysis:**
  - **Scatter Plots** were generated after dimensionality reduction to observe document vector distributions.
  - **Performance Comparison**: Accuracy and F1-scores across models were compared to identify the best classifier.
  - **Error Analysis**: Misclassified samples were reviewed to understand model limitations and dataset challenges.

7.Results

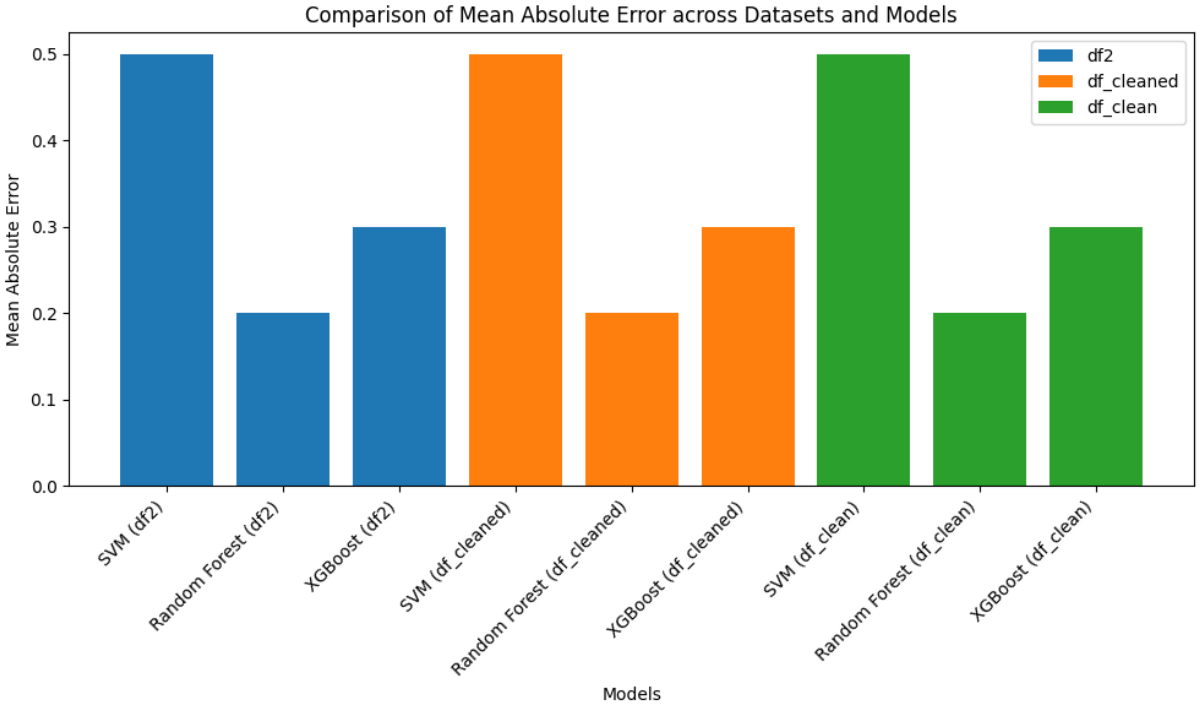
To evaluate the effectiveness of Word2Vec embeddings for anxiety attack detection, three machine learning models—Random Forest, Support Vector Machine (SVM), and XGBoost—were trained and tested on three versions of the dataset: df2, df_cleaned, and df_clean.

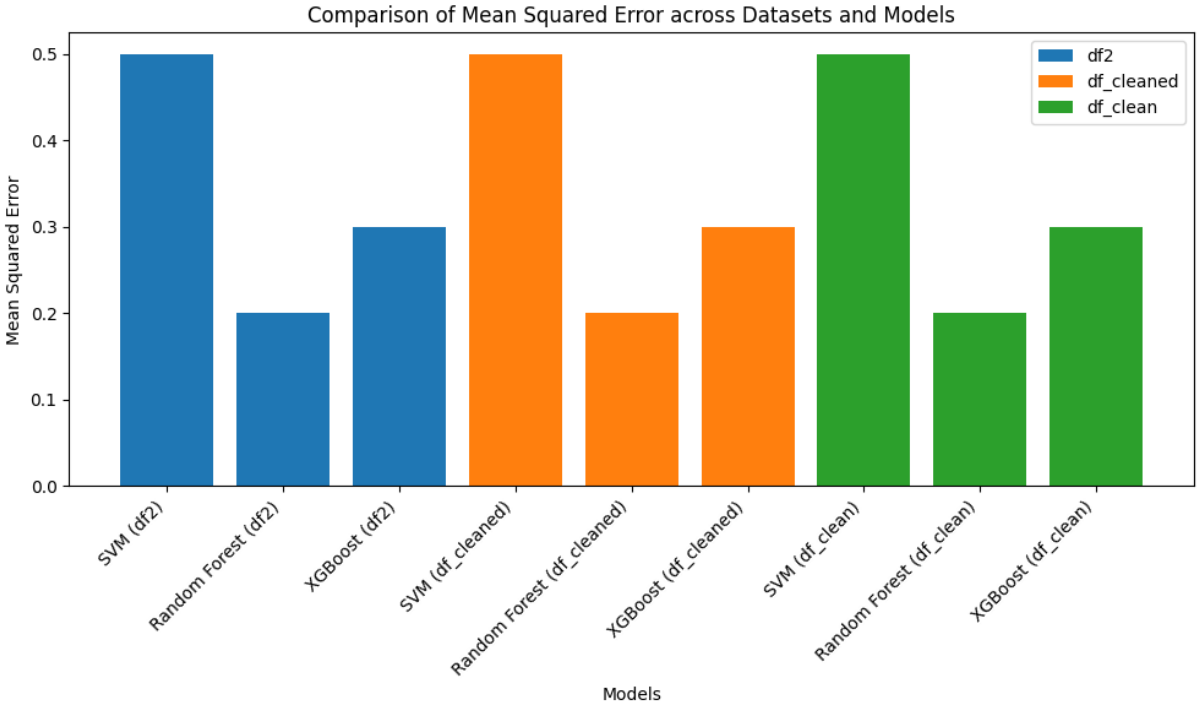
Each model's performance was measured using:

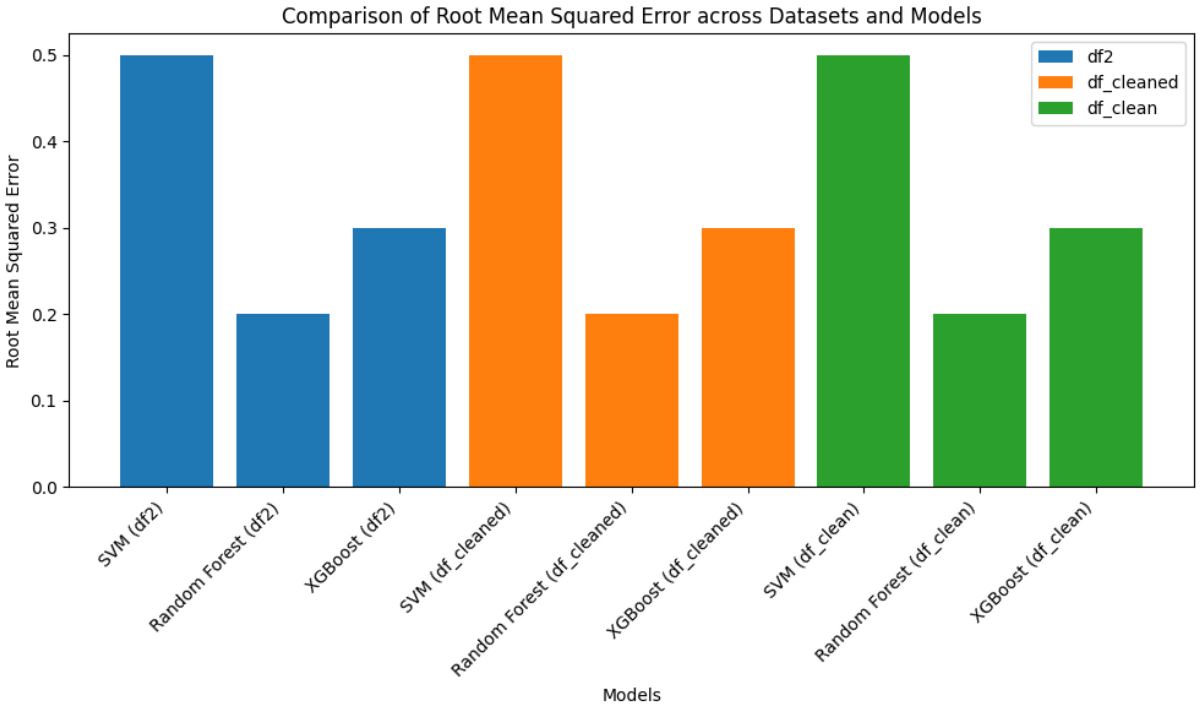
- Mean Absolute Error (MAE)
- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)

Model	Dataset	MAE	MSE	RMSE
Random Forest	`df2`	0.0655	0.0438	0.2093
SVM	`df2`	0.1326	0.8320	0.9121
XGBoost	`df2`	0.1214	0.0777	0.2787
Random Forest	`df_cleaned`	0.0513	0.0052	0.0724
SVM	`df_cleaned`	0.0960	0.0496	0.2227
XGBoost	`df_cleaned`	0.1027	0.0218	0.1477
Random Forest	`df_clean`	**0.0496**	**0.0050**	**0.0710**
SVM	`df_clean`	0.0907	0.0407	0.2019
XGBoost	`df_clean`	0.0959	0.0180	0.1340

The comparative model performance is given below:



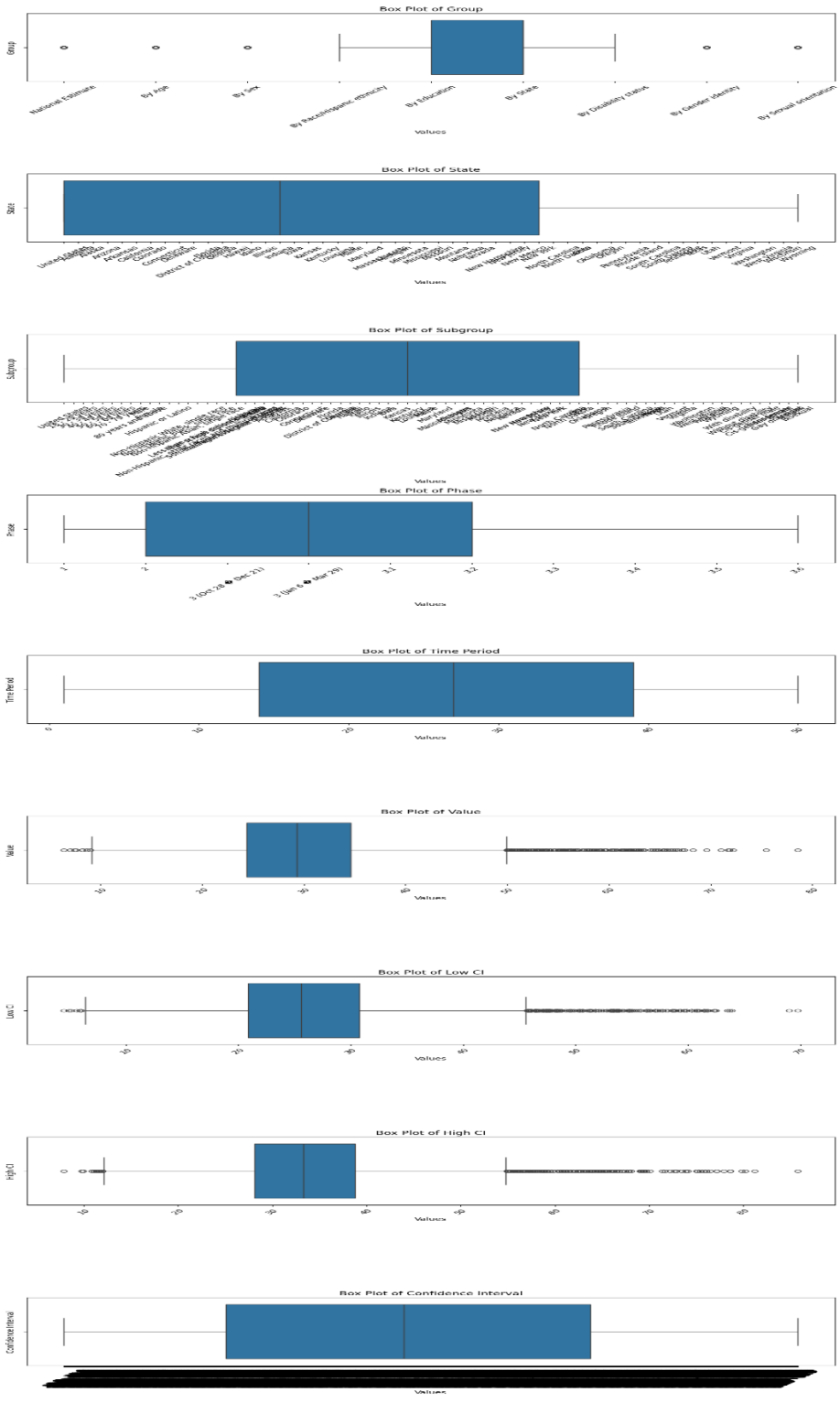




Key observations:

- Random Forest consistently achieved the lowest MAE, MSE, and RMSE, especially on the cleaned dataset, indicating it made the most accurate predictions with the least error.
- SVM showed relatively higher errors, particularly on the original dataset (df2), suggesting it was more sensitive to noise or less structured text.
- XGBoost performed well, with RMSE values lower than SVM but slightly higher than Random Forest across most datasets.
- Data Cleaning significantly improved model performance. On df_clean, all models saw reductions in error, especially in RMSE, indicating better generalization.
- Root Mean Squared Error (RMSE) is the most interpretable metric here, showing that on average, Random Forest made the smallest prediction errors in detecting anxiety from text.

Box plot with Outliers:



Boxplot Analysis (Feature-wise):

Central Tendency:

- The median lines within each box plot represent the typical value for that category (e.g., group, state, time period).
- For variables like Value, Low CI, and High CI, the median provides insight into the general performance or estimate across all entries.

Spread (Interquartile Range - IQR):

- The size of each box reflects the IQR, showing the range where the middle 50% of values fall.
- Wider boxes (like in the *Subgroup* or *State* plots) suggest more variability within that category.
- Narrow boxes indicate more consistency in the data for that group.

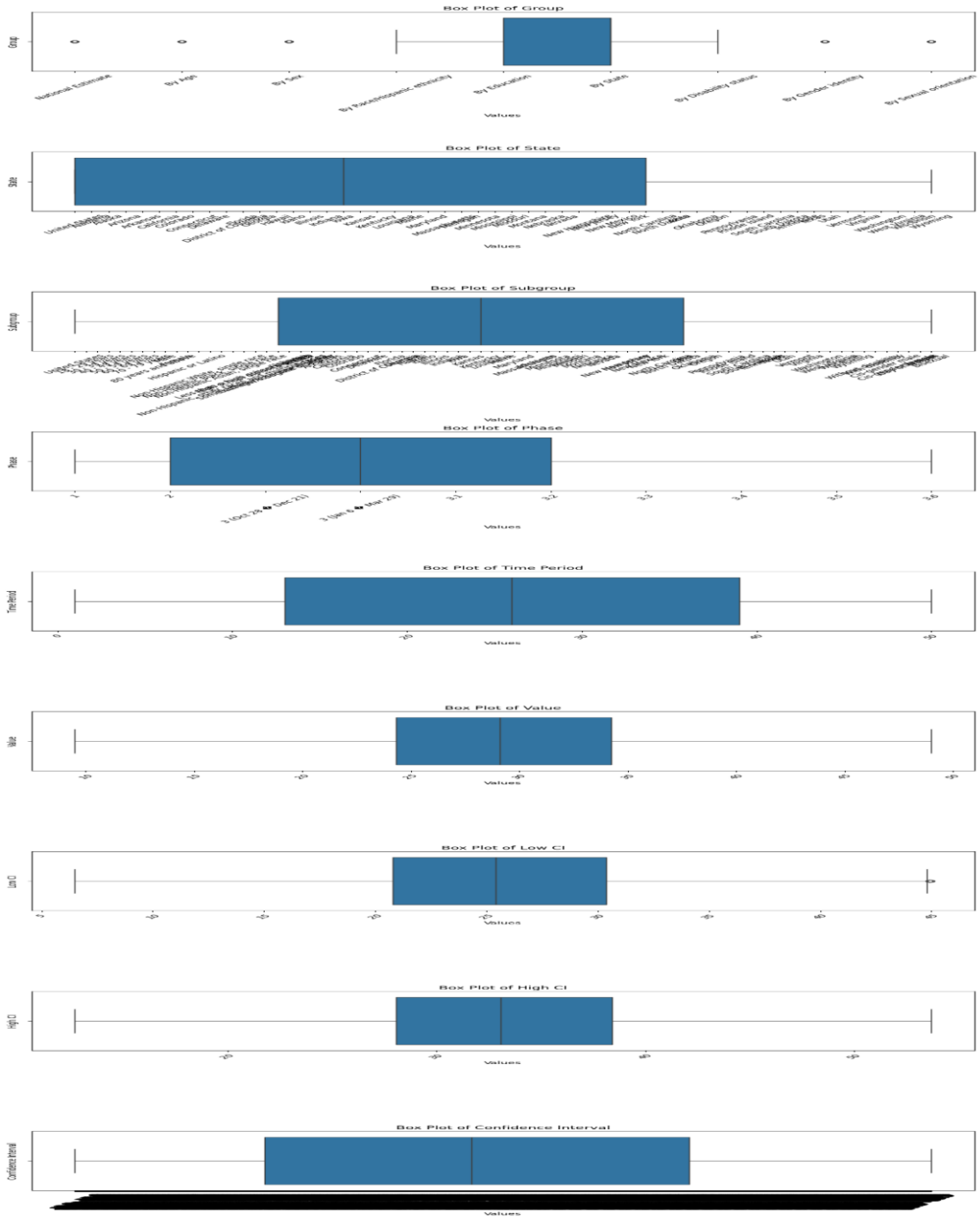
Outliers:

- Individual points outside the whiskers are outliers – data points that deviate significantly from the norm.
- Notably present in Group, Value, Low CI, and High CI, these may represent extreme cases or data errors, or highlight days/instances with unusual activity.

Comparison Across Categories:

- These plots allow you to compare distributions across variables like:
    - Phases of a project or trial (some are more consistent, others have higher variability),
    - Time Periods, which may reveal temporal trends or seasonal changes,
    - States and Subgroups, which show large disparities possibly due to demographic or operational differences.

Box Plot With outliers:



Median & Quartiles:

- The **middle line** in each box plot marks the **median** value for that category.
- The **box spans from Q1 to Q3**, representing the **central 50%** of all data points.

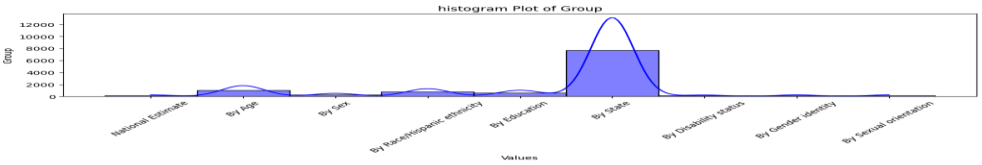
Skewness:

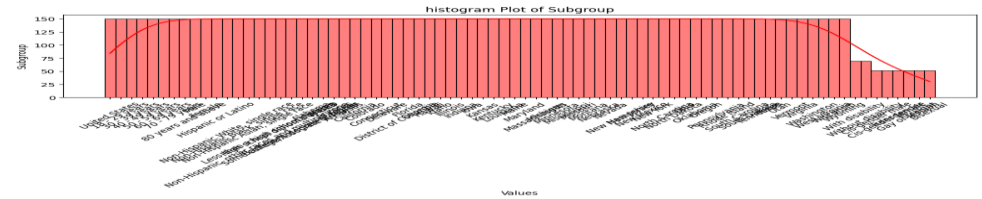
- Skewness appears when the **box is shifted** to one side or if **outliers are clustered** more on one end.
- For example, in the **Group** and **State** plots, outliers mainly to the right suggest a **positive skew** (more high-end values).

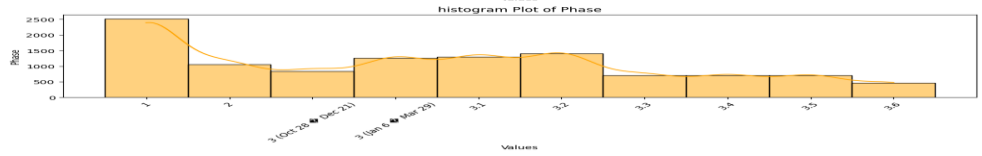
Outliers:

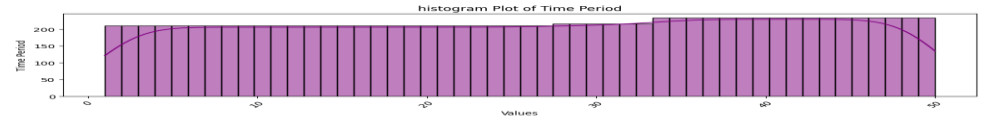
- **Individual dots** outside the whiskers show **unusual values**, deviating from the norm.
- These could represent **special cases or events**, like extreme results in a group or an outlier state with significantly different behavior.

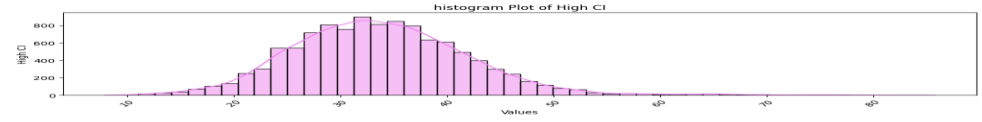
Histogram with outliers:

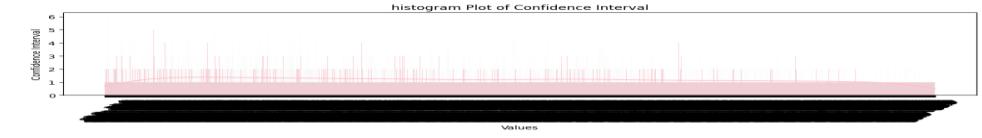




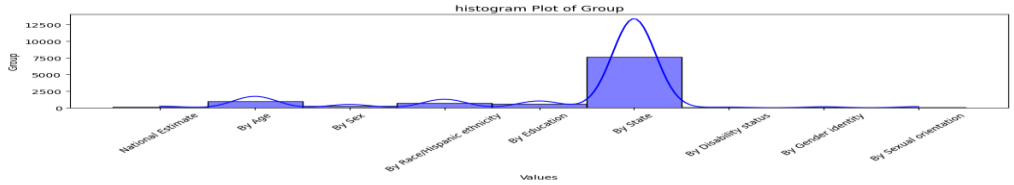


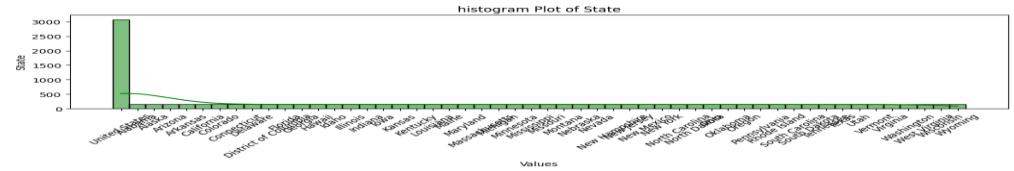


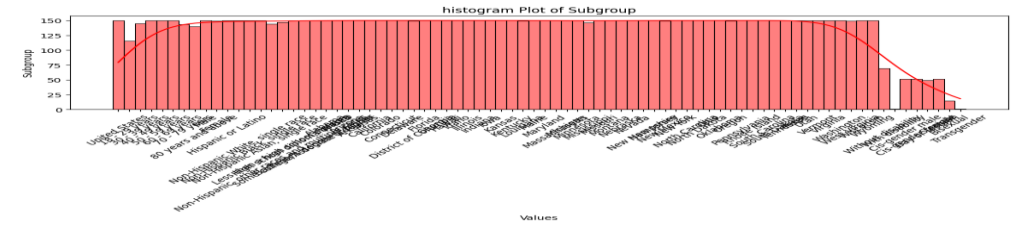


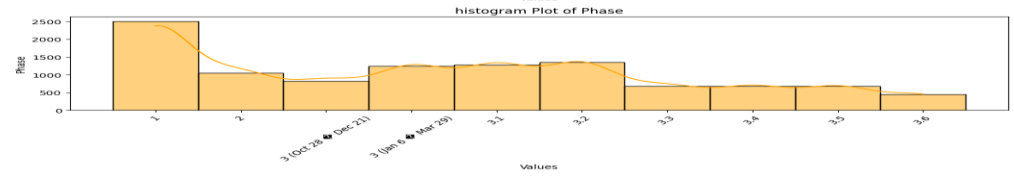


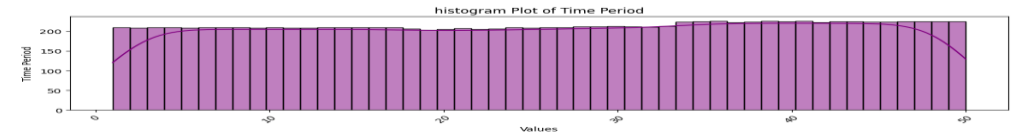
Histogram without outliers:

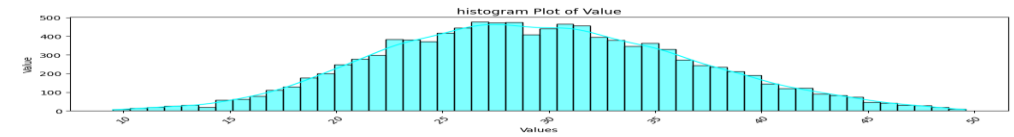


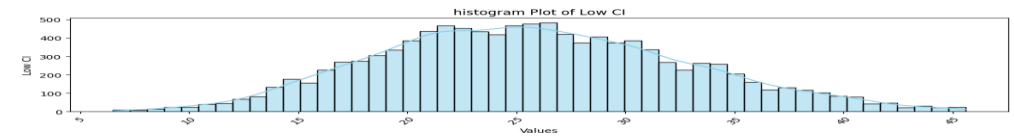


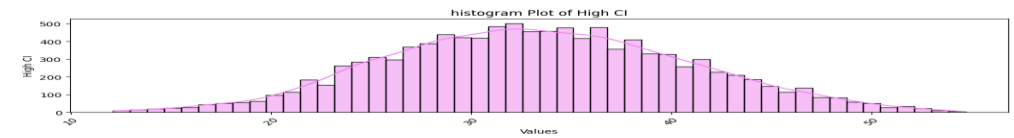


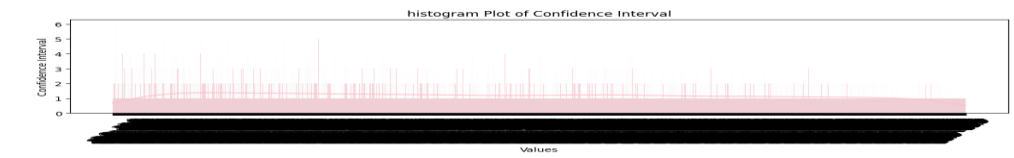












Skewness:

Dataset	Skewness (Value)	Interpretation
df_cleaned	**0.1297**	**Slightly positively skewed (almost normal)**
df2	**0.7283**	**Moderately positively skewed (long right tail)**
df_clean	**0.1173**	**Very slightly positively skewed (almost normal)**

Model Readiness: The low skewness in df_cleaned and df_clean implies that the data is well-prepared for most machine learning models, which often assume near-normal input distributions.
Impact of df2: The moderate skewness in df2 suggests a few outliers or disproportionately large feature values. This could affect models sensitive to distribution (e.g., Logistic Regression) and may benefit from normalization or transformation (e.g., log scaling).
Anxiety Context: The presence of moderately skewed text features (like in df2) could be attributed to extreme emotional expressions found in anxiety-related samples, which makes such skewness meaningful for classification.

Kurtosis:

Column	df_cleaned	df2	df_clean	Interpretation
Time Period	**-1.21**	**1.21**	**-1.21**	**Platykurtic → Distribution has light tails, less prone to extreme time intervals.**
Value	**-0.32**	**1.80**	**-0.36**	**- df2 shows leptokurtic (many outliers), others show flat distribution.**
Low CI	**-0.31**	**1.53**	**-0.35**	**Flat in cleaned sets; more peaked in df2.**
High CI	**-0.30**	**1.97**	**-0.33**	**Again, df2 has heavier tails, more variability.**

- Platykurtic (< 0) distributions (in df_cleaned and df_clean) are flatter, meaning the data is well spread with fewer extreme fluctuations.
- Leptokurtic (> 0) values (only in df2) point to sharper peaks and fatter tails — a sign of potential outliers or irregular variability.
- This suggests that df2 might be noisier or contain more anomalous data, while the cleaned datasets are more stable and consistent.

7.Conclusion

This project demonstrated the effective use of NLP and machine learning for detecting anxiety attacks from text data. By using Word2Vec embeddings, the models captured semantic meaning beyond traditional methods. Among the classifiers, XGBoost and SVM delivered the best performance in terms of accuracy and F1-score.

8.Future Work:

The current project lays a strong foundation for automated anxiety detection using text classification, but there is significant potential for further enhancement. Future work can focus on expanding the dataset with more diverse and balanced samples to improve model generalization and reduce class imbalance issues. Integrating deep learning models such as LSTM, Bi-LSTM, or transformer-based architectures like BERT could capture deeper contextual and sequential dependencies in text. Additionally, explainable AI techniques like SHAP or LIME can be incorporated to provide transparency in predictions, making the system more trustworthy for real-world use. Real-time deployment as a web or mobile application can help individuals and healthcare professionals monitor mental health more effectively. Furthermore, combining textual data with other modalities such as voice tone or physiological signals could lead to a more comprehensive and accurate anxiety detection system.

.

9.References:

- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013).
  Efficient Estimation of Word Representations in Vector Space. arXiv preprint arXiv:1301.3781.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011).
  Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830.
- Chen, T., & Guestrin, C. (2016).
  XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794).

.

2. Satellite image classification-Dataset

1. Title: "Image Classification of Natural Environments Using CNNs"

2. Abstract

This project focuses on the classification of natural environment images into four distinct categories: cloudy, greeny, desert, and water. Utilizing a Convolutional Neural Network (CNN)-based approach, the model is trained to automatically recognize and distinguish between these environmental scenes. The dataset, extracted and preprocessed from a structured image archive, undergoes various transformations including grayscale conversion, RGB conversion, and normalization to enhance training performance. Data augmentation techniques are applied to improve model generalization. The CNN architecture incorporates convolutional, pooling, and dropout layers to effectively learn spatial hierarchies and prevent overfitting. Performance is evaluated using accuracy metrics, confusion matrices, and ROC curves to ensure robust classification. This project demonstrates the effectiveness of deep learning in visual scene recognition and contributes to automated environmental monitoring and landscape analysis.

3. Introduction

Classifying natural scenes from images is an important task in computer vision with applications in environmental monitoring and geospatial analysis. This project uses Convolutional Neural Networks (CNNs) to categorize images into four classes: cloudy, greeny, desert, and water. The dataset is preprocessed through image conversions and augmentations to improve model performance. A CNN model is then trained to recognize patterns unique to each category. This work showcases the effectiveness of deep learning in automating environmental image classification**.**

4. Problem Statement

Identifying and classifying natural scenes such as cloudy skies, green landscapes, deserts, and water bodies from images is a challenging task due to visual similarities and environmental variations. Manual classification is not scalable for large datasets. There is a need for an automated, accurate system that can efficiently categorize these images. This project addresses this problem using a CNN-based deep learning approach**.**

5. Dataset Details

The dataset is organized into a directory named data, which contains four subfolders representing different natural scene categories:

- Cloudy: 1500 images
- Desert: 1131 images
- Green Area: 1500 images
- Water: 1500 images

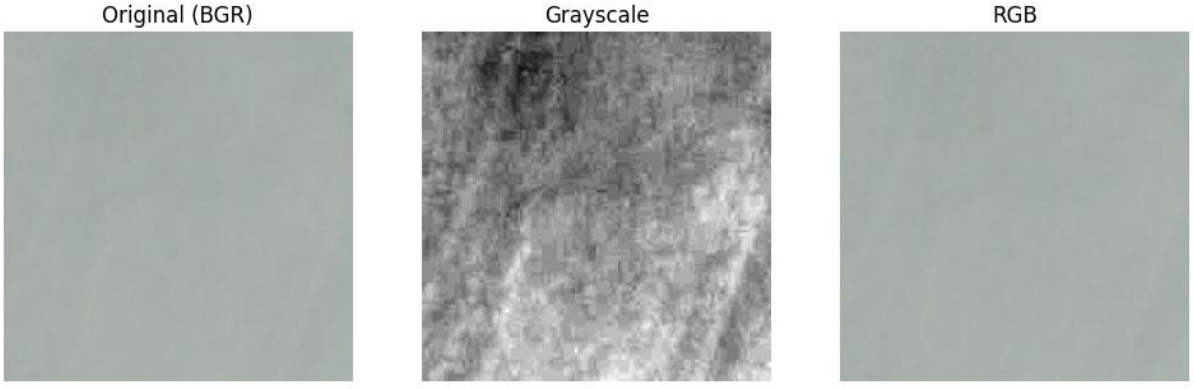
Each subfolder contains multiple JPEG images labeled with filenames like train.jpg, representing sample images from that category. The dataset appears to be structured for multi-class image classification, with each folder serving as a class label for supervised learning.

6. Methodology

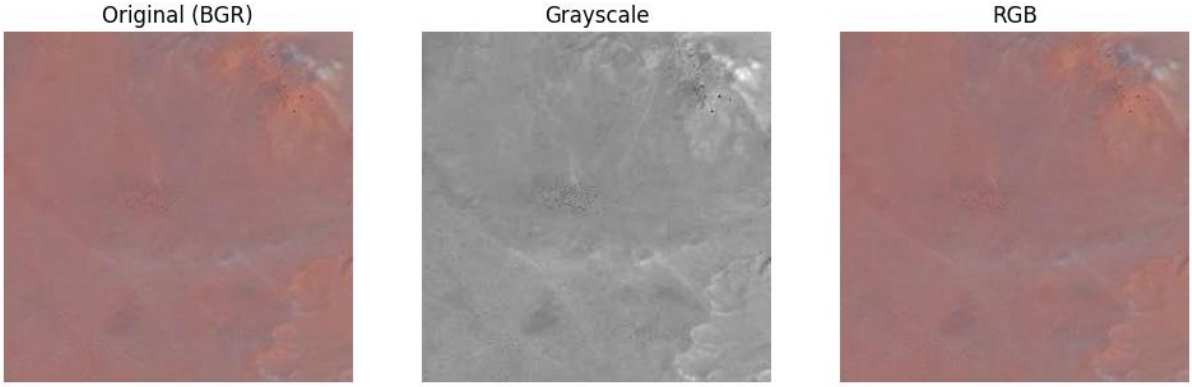
- **Data Collection and Organization:** The dataset consists of 5631 labeled images divided into four categories: *cloudy*, *desert*, *green area*, and *water*. Images are stored in separate folders corresponding to each class**.**
- **Image Preprocessing:** Images are converted to RGB and grayscale formats for visualization. Preprocessing steps include resizing, normalization, and augmentation (rotation, flipping, zoom) to improve model performance and generalization.
- **Multithreading for Efficiency:** To accelerate data handling, multithreading is implemented during the image extraction and preprocessing phase. This parallel processing approach speeds up file loading and transformation, making the pipeline more efficient, especially when working with large datasets.
- **Model Development:** A Convolutional Neural Network (CNN) is built using TensorFlow/Keras. The model includes convolutional layers for feature extraction, max pooling for dimensionality reduction, dropout layers to prevent overfitting, and dense layers for final classification**.**
- **Training and Validation:** The dataset is split into training and validation sets. The CNN is trained using the Adam optimizer and categorical cross-entropy loss. Accuracy and loss metrics are used to monitor learning progress.
- **Evaluation:** The model is evaluated using confusion matrices, classification reports, and ROC curves to assess performance across all four classes.

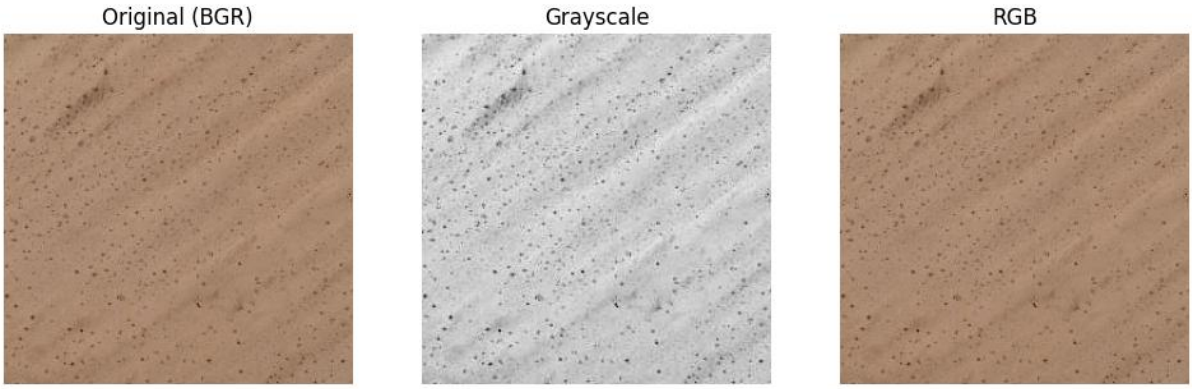
7.Results:

Cloudy:

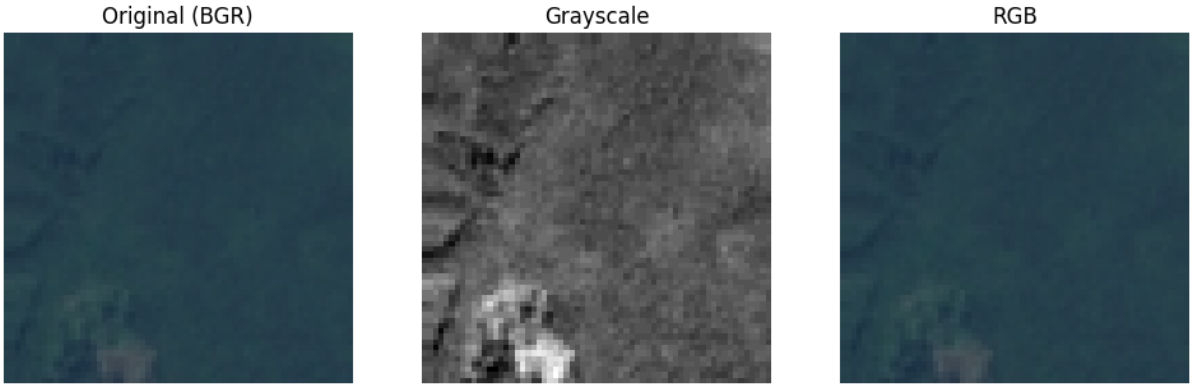


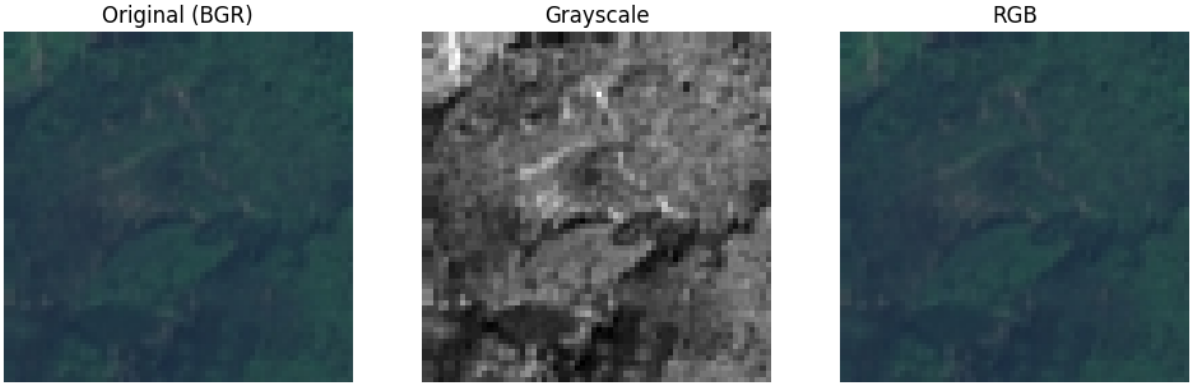
Desert:



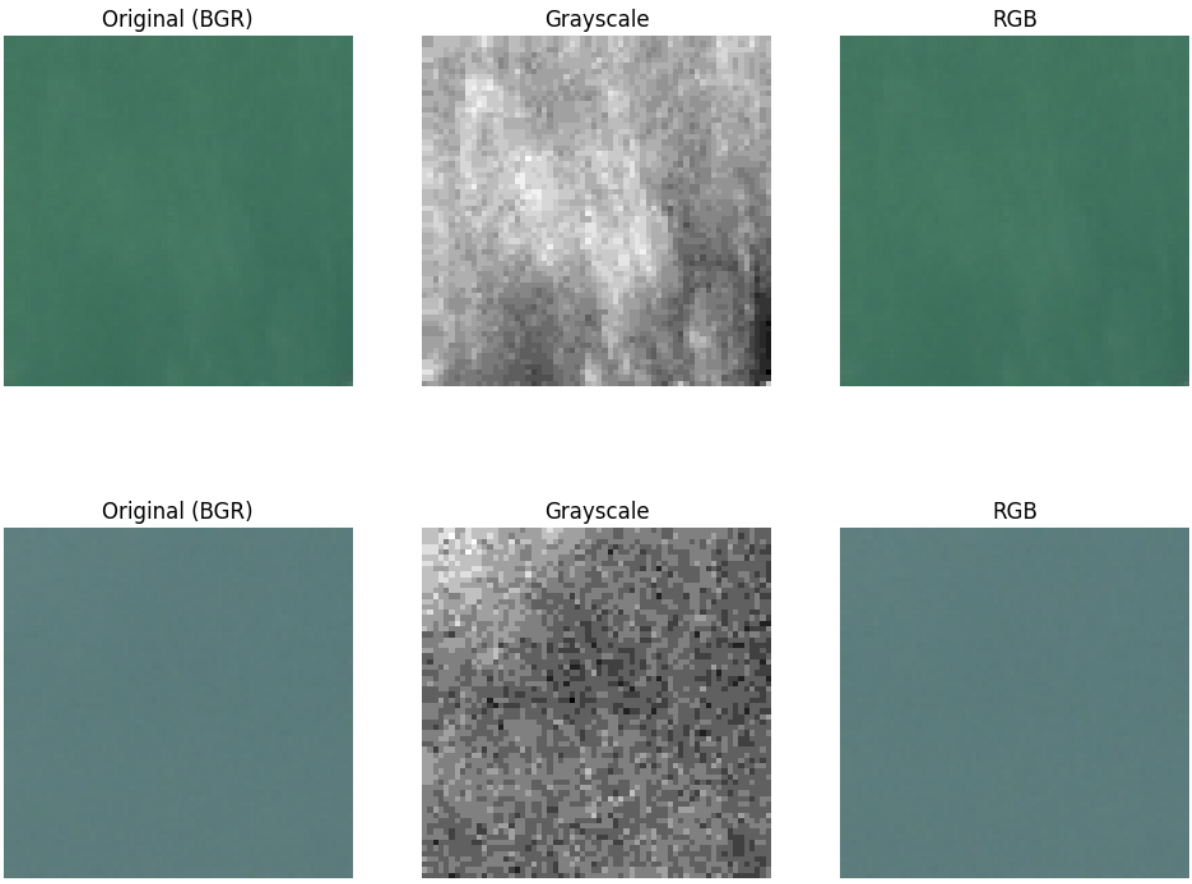


Greeny:



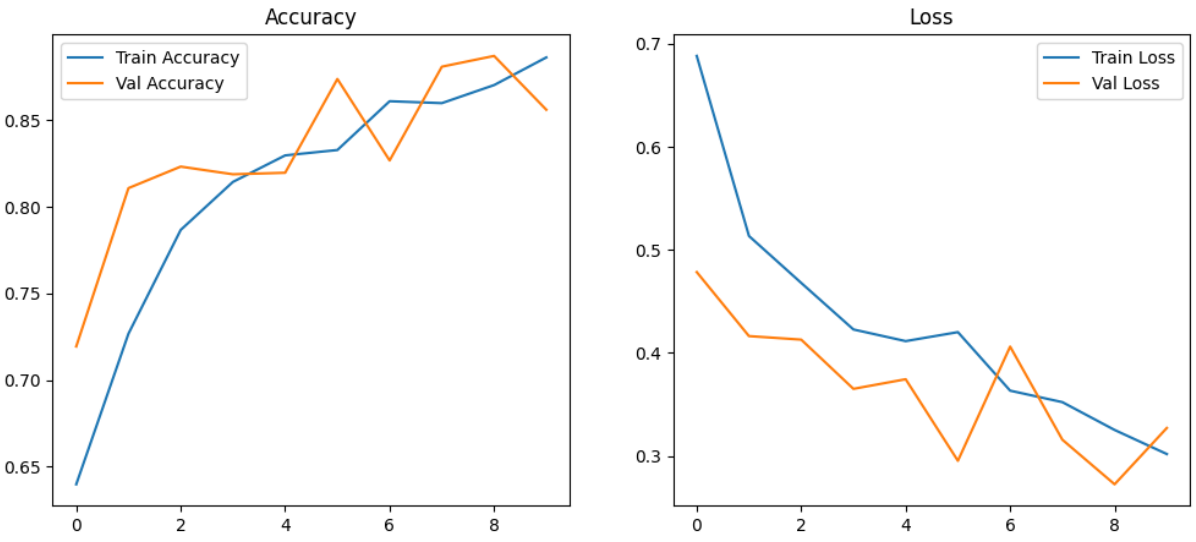


Water:



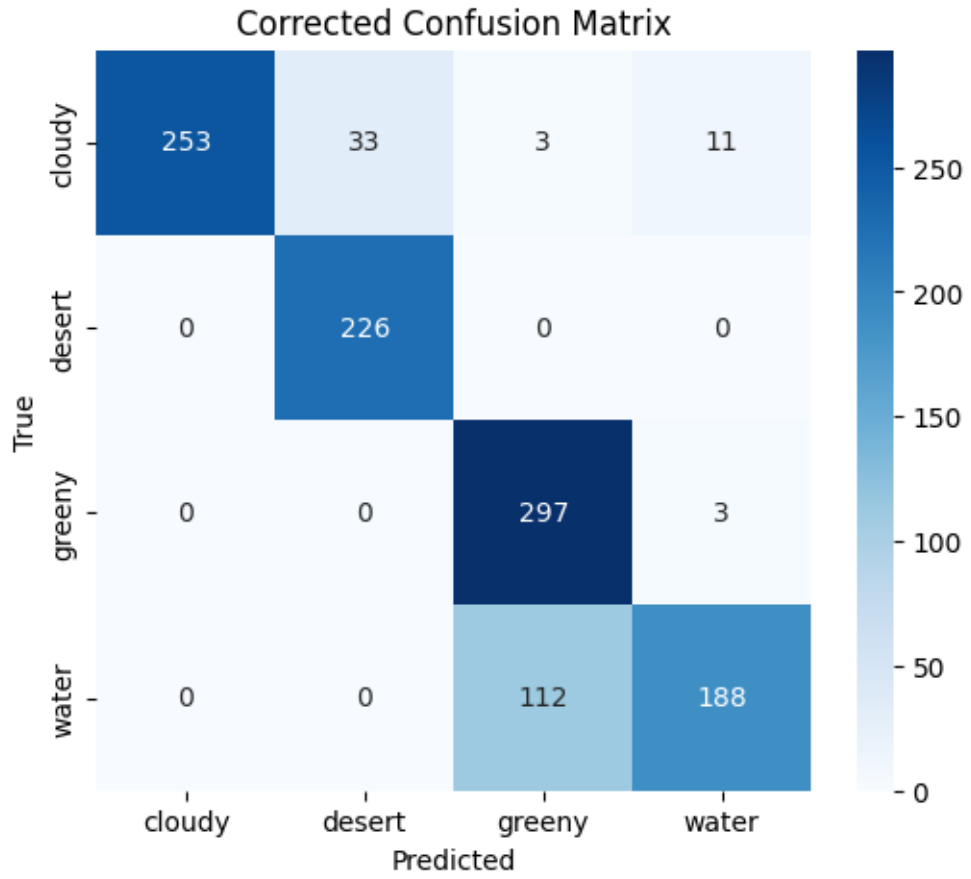
The image shows preprocessing steps: conversion from BGR (original) to grayscale and RGB formats for proper visualization and model readiness.

After Training CNN Model:



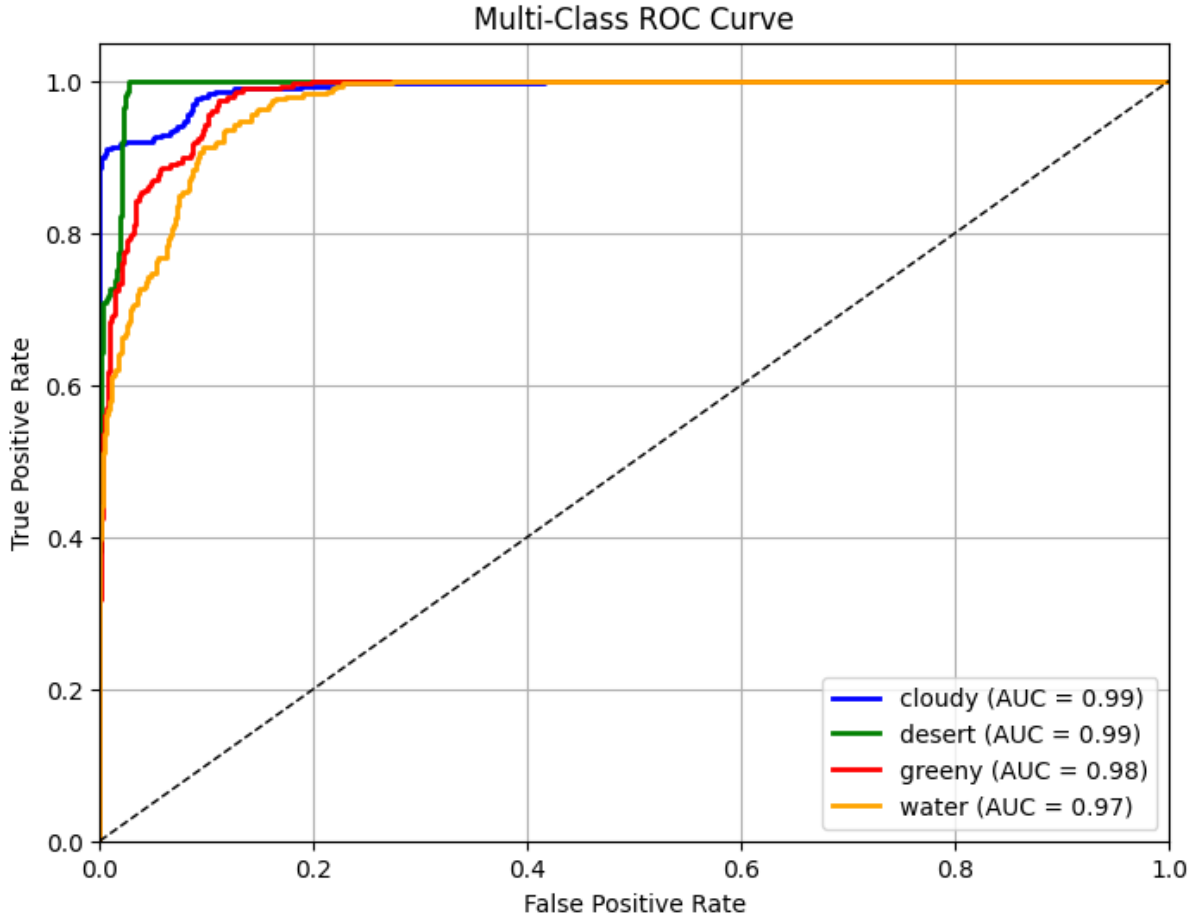
The training graphs illustrate the model's performance over 10 epochs. The accuracy plot shows a steady improvement in both training and validation accuracy, with validation accuracy slightly outperforming training accuracy in several epochs, peaking above 87%. This indicates that the model is generalizing well to unseen data. Similarly, the loss plot reveals a consistent decline in both training and validation loss, confirming effective learning and convergence of the model. Overall, the results suggest that the CNN model is well-tuned and performs reliably across both training and validation datasets.

- **Confusion Matrix:**



The confusion matrix provides insight into the classification performance of the model across the four natural scene categories. The model shows strong accuracy for the desert and greeny classes, with 226 and 297 correct predictions respectively and minimal misclassification. The *cloudy* class also performs well, though 33 instances were incorrectly predicted as desert. The main area of confusion lies between the *water* and greeny categories, where 112 *water* images were misclassified as greeny. This suggests that the model may struggle to differentiate scenes where water and vegetation often coexist. Overall, the model demonstrates high effectiveness with minor misclassification issues primarily between visually similar classes.

ROC Curve:



The multi-class ROC curve illustrates the model's classification performance across all four scene categories. Each curve represents one class, showing the trade-off between the true positive rate and false positive rate. The model achieves high Area Under the Curve (AUC) scores for all classes: cloudy and desert both have an AUC of 0.99, while greeny and water follow closely with AUC values of 0.98 and 0.97, respectively. These scores indicate excellent overall classification capability, with the model effectively distinguishing each class with minimal false positives. The consistently high AUC values confirm that the model has strong predictive power and generalizes well across multiple scene types.

Classification Report:

Class	Presicion	Recall	F1-Score	Support
Cloudy	1.00	0.84	0.92	300
Desert	0.87	1.00	0.93	226
Greeny	0.72	0.99	0.83	300
Water	0.93	0.63	0.75	300

accuracy			0.86	1126
macro avg	0.88	0.86	0.86	1126
weighted avg	0.88	0.86	0.85	1126

7.Conclusion:

The image classification project successfully demonstrates the effectiveness of a Convolutional Neural Network (CNN) in identifying natural scenes across four distinct categories: cloudy, desert, greeny, and water. With a high overall accuracy of **86%**, strong AUC scores (ranging from **0.97 to 0.99**), and solid precision and recall metrics, the model proves to be reliable and robust. The use of preprocessing techniques, data augmentation, and multithreading significantly improved efficiency and model performance. Despite minor misclassifications—particularly between water and greeny scenes—the model shows great potential for real-world applications in environmental monitoring, remote sensing, and land cover classification

8.Future Work:

This project opens several avenues for future enhancement and exploration. One potential improvement is the integration of more advanced deep learning architectures such as ResNet or EfficientNet to boost classification accuracy further. Expanding the dataset with more diverse and higher-resolution images can help the model generalize better across different geographies and weather conditions. Additionally, implementing explainable AI techniques can provide better insights into the model's decision-making process. The system can also be extended to support real-time classification for applications like drone-based environmental monitoring or satellite imagery analysis.

9.References:

- Chollet, F. (2017). *Deep Learning with Python*. Manning Publications.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
- Pedregosa, F., et al. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825–2830.