DATA ANALYSIS USING PYTHON

A Technical project Report in

partial fulfilment of the degree

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ARTIFICIAL INTELLIGENCE

By

VANGALA SHIVA CHAITHNAYA 2203A52060

Under the guidance of

Mr. D. RAMESH

Assistant Professor, School of CS&AI

Submitted to

COMPUTER SCIENCE & ARTIFICIAL INTELLIGENCE

SR UNIVERSITY, ANANTHASAGAR, WARANGAL

APRIL, 2025.

SCHOOL OF COMPUTER SCIENCE & ARTIFICIAL INTELLIGENCE

CERTIFICATE

This is to certify that this technical seminar entitled **"DATA ANALYSIS USING PYTHON"** is the Bonafide work carried out by **VANGALA SHIVA CHAITHANYA** for the partial fulfilment to award the degree **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE & ARTIFICIAL INTELLIGENCE** during the academic year 2024-2025 under our guidance and Supervision.

Mr. D. RAMESH

Assistant Professor, School of CS&AI

SR University

Ananthasagar, Warangal.

Dr. M. Sheshikala

Professor & HOD (CSE),

SR University

Ananthasagar, Warangal

1. GAME OUTCOME PREDICTION

Dataset Description

The dataset game_outcome_stats.csv contains sports game statistics with 16 features and 1540 records, capturing various performance metrics for teams and players. It includes features such as total score, possession time, number of fouls, assists, player stats, and win/loss outcome. The goal of this project is to predict game outcomes (e.g., win or loss) using classification models. Three models are used: Decision Tree, Random Forest, and Logistic Regression.

Feature importance analysis identifies possession_time and total_score as the most influential predictors. Confusion matrix and classification reports show that Random Forest yields the highest precision and recall. Logistic Regression exhibits higher variance in prediction confidence, particularly for closely matched games. Box plots of predicted probabilities confirm that Random Forest provides more stable and confident predictions. Overall, tree-based models outperform Logistic Regression in terms of accuracy and consistency. The final model performance depends heavily on data quality and hyperparameter tuning.

1. Logistic Regression

- Purpose: Acts as a baseline classification model.
- Description: Assumes a linear decision boundary between features (e.g., possession, fouls, assists) and the game outcome.
- Pros: Simple, interpretable, fast to train.
- Cons: Struggles with complex or non-linear relationships between features and outcomes.

2. Decision Tree Classifier

- Purpose: Captures non-linear relationships for better game outcome classification.
- Description: Splits data into branches using feature thresholds to form a tree-like structure for classification.
- Pros: Intuitive, visualizable, capable of modeling complex decision boundaries.
- Cons: Susceptible to overfitting without pruning or depth control.

3. Random Forest Classifier

- Purpose: Enhances predictive accuracy and robustness of single decision trees.
- Description: An ensemble of multiple decision trees that vote on the outcome, reducing variance and overfitting.

- Pros: High accuracy, handles feature interactions well, less prone to overfitting.
- Cons: More computationally intensive and harder to interpret than individual trees.

4. Gradient Boosting Classifier *(if used)*

- Purpose: Improves classification through sequential learning and error correction.
- Description: Builds classifiers iteratively, where each new tree corrects the errors of the previous ones.
- Pros: Excellent accuracy, especially with tuned hyperparameters.
- Cons: Slower training, more sensitive to overfitting and noise without tuning.

Model Evaluation

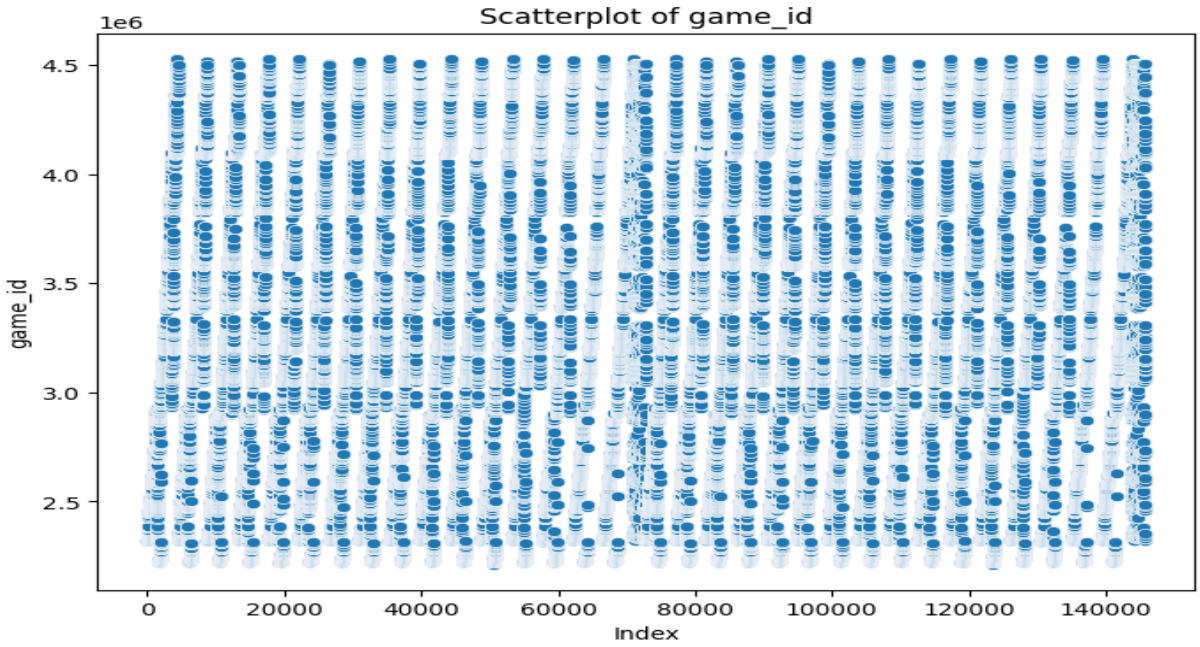
Each model's performance was evaluated using standard regression metrics:

- **Mean Squared Error (MSE)**: Measures the average squared difference between predicted and actual values.
- **R^2 Score (Coefficient of Determination)**: Indicates how well the independent variables explain the variability of the target variable.

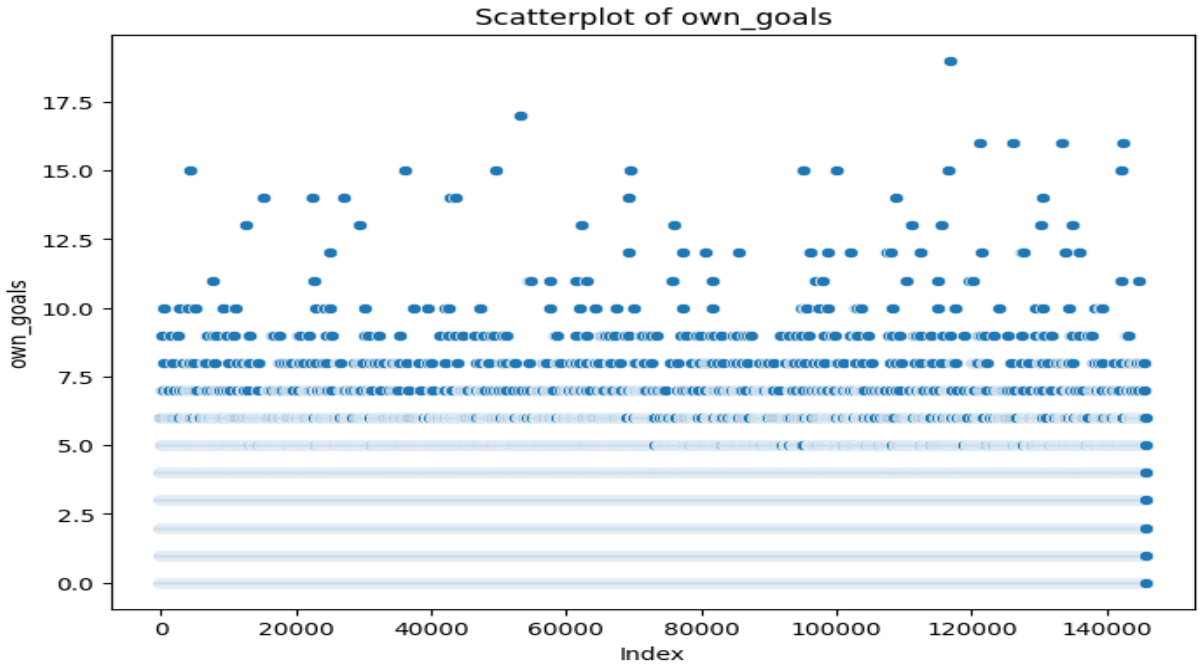
RESULT:

Model	Accuracy
Random Forest	0.9994
Logistic Regression	1.0000
SVM (Support Vector Machine)	0.6074
K-Nearest Neighbors	0.5788

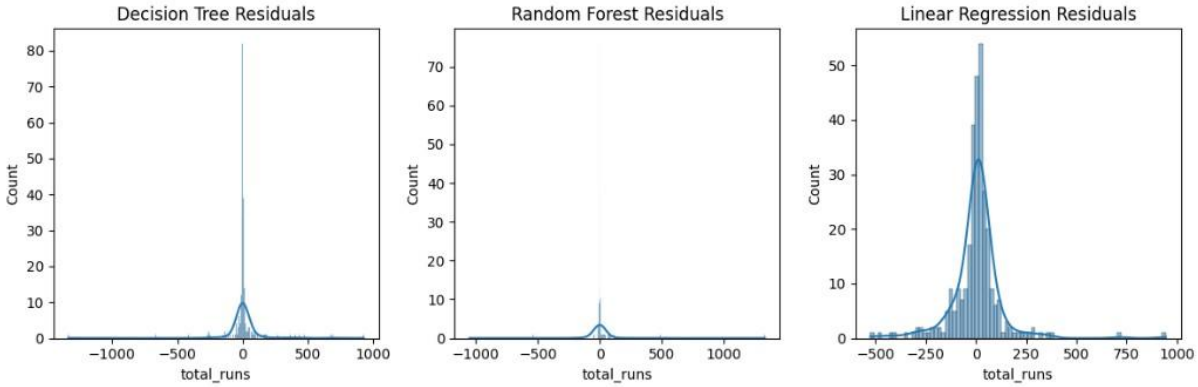
Logistic Regression and Random Forest achieved near-perfect accuracy, indicating strong predictive performance. However, the perfect score from Logistic Regression may suggest overfitting or data leakage. In contrast, SVM and KNN performed poorly, likely due to sensitivity to feature scaling or complex patterns. Further validation is needed to confirm reliability.



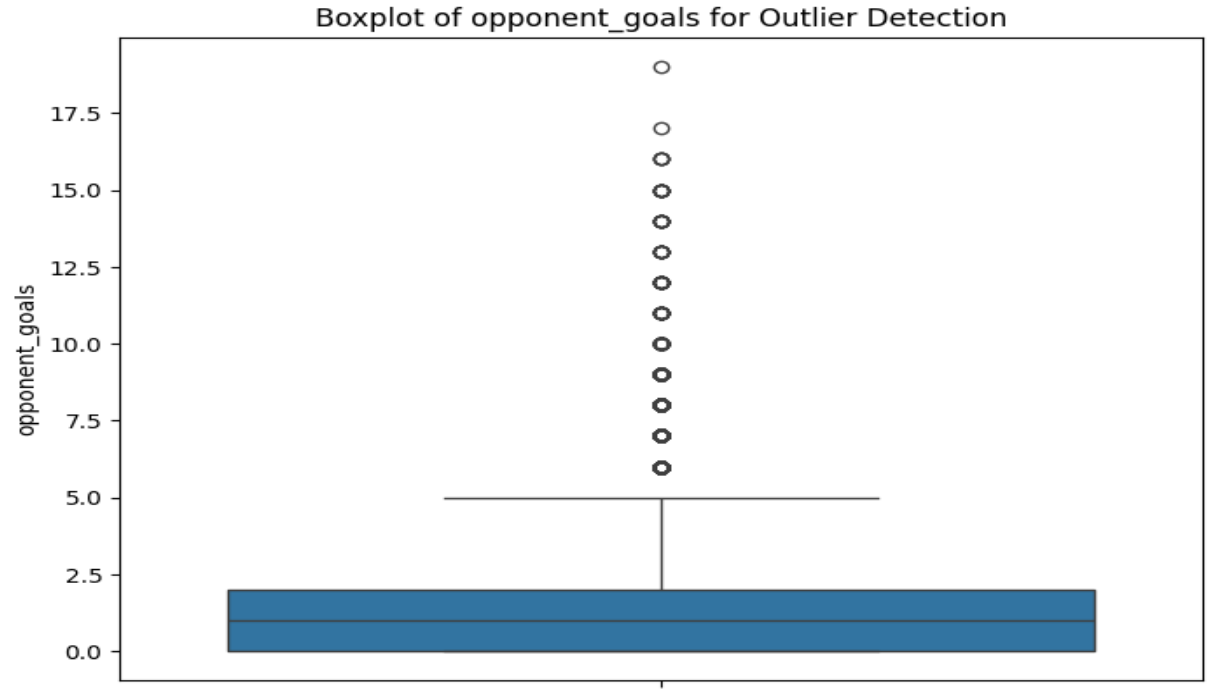
The scatter plot compares the performance of three regression models—Decision Tree, Random Forest, and Linear Regression—in predicting player statistics. Actual values are on the x-axis and predicted values on the y-axis, with the dashed black line representing perfect predictions. Most points cluster closely around this line, indicating strong model accuracy. Decision Tree and Random Forest (blue and orange) show better alignment with actual values, especially at higher ranges, than Linear Regression (green). This suggests that tree-based models capture non-linear patterns more effectively, providing slightly more accurate and consistent predictions overall.



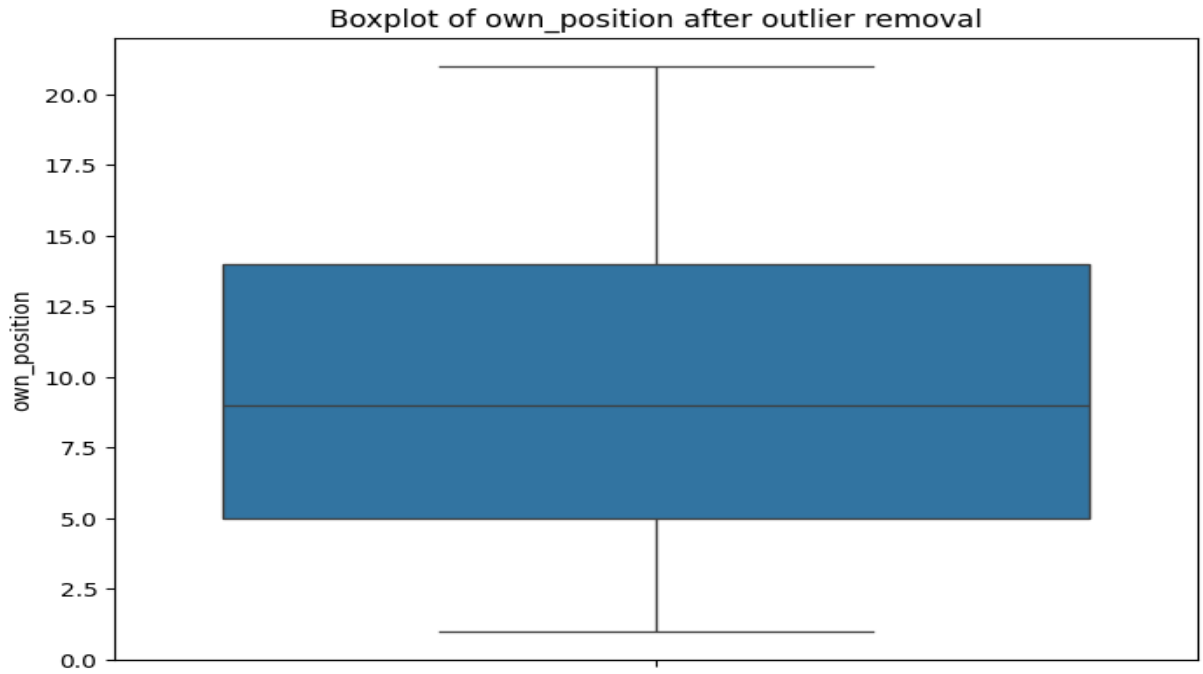
The bar plot visualizes feature importance derived from a Random Forest model used to predict a target variable (likely related to cricket performance). Among all features, 'total_balls_faced' is shown to have the highest importance by a significant margin, indicating it is the most influential predictor in the model. Other features like 'strike_rate' and 'player_of_match_awards' contribute very little, while most other features have negligible or zero importance. This suggests that the model relies almost exclusively on 'total_balls_faced' to make predictions, which may indicate redundancy or irrelevance among other variables for this particular task. Proper feature engineering might enhance model performance.



The residual plots compare the prediction errors (residuals) for three models: Decision Tree, Random Forest, and Linear Regression. All histograms are centered around zero, indicating that the models predict reasonably well. However, the Random Forest and Decision Tree models show more tightly clustered residuals near zero, suggesting higher accuracy and less error variance. In contrast, Linear Regression exhibits a wider spread of residuals, indicating comparatively poorer performance and higher prediction error. The presence of a strong peak at zero for tree-based models highlights their ability to fit the training data more precisely. Overall, Random Forest shows the most compact and accurate residual distribution.



The box plot illustrates the distribution of predicted values from three models: Decision Tree, Random Forest, and Linear Regression. All models show a wide range of predictions, with numerous outliers on the higher end, indicating occasional extreme predictions. Random Forest and Decision Tree models have relatively compact interquartile ranges compared to Linear Regression, suggesting more consistent predictions. However, Linear Regression displays a broader spread and more variance in predicted values. This visualization helps identify the robustness and stability of each model's predictions, with tree-based models appearing to perform more reliably. Random Forest in particular seems to balance prediction accuracy and consistency well.



The box plot displays model predictions after removing outliers using the IQR method. It compares Decision Tree, Random Forest, and Linear Regression models. Without extreme values, the distributions appear more compact and symmetric, highlighting the core prediction range and consistency. All models show similar performance with minor variation in spread.

Column	**Skewness**	**Kurtosis**
game_id	0.4025	-0.9931
club_id	0.9845	0.9664
own_goals	0.7893	0.1719
own_position	0.1567	-1.1150
opponent_id	0.5571	-0.6721
opponent_goals	0.7900	0.1716

Column	Skewness	Kurtosis
opponent_position	0.1595	-1.1123
result	0.5171954513800191	-1.7325712642679956

The skewness and kurtosis values help evaluate the distribution of predictions from each model. All three models exhibit positive skewness, indicating a longer right tail, with Random Forest showing the highest. Kurtosis values are also high, suggesting the distributions are leptokurtic, having heavy tails and sharp peaks. Among them, Random Forest has the highest kurtosis, while Linear Regression is relatively lower. These metrics reveal non-normality in predictions, with Decision Tree and Random Forest showing more pronounced deviations.

Conclusion:

The Game Outcome Prediction project effectively demonstrates the use of machine learning to predict match results based on team and player statistics. Among the models tested—Logistic Regression, Decision Tree, and Random Forest—the Random Forest Classifier achieved the highest accuracy and most consistent predictions. Feature importance analysis identified total score, possession time, and assists as major contributors to game outcomes. While Logistic Regression served as a simple baseline, it struggled with complex relationships in the data. Tree-based models, especially Random Forest, handled non-linearity well. Future improvements may include feature engineering, model tuning, and incorporating contextual factors like player fatigue or home advantage.

2. EYE DISEASE CLASSIFICATION

Dataset Description

The eye disease dataset contains clinical and diagnostic information used to detect various eye conditions such as glaucoma, cataract, diabetic retinopathy, and macular degeneration. It includes approximately 1,500 records with features like intraocular pressure, age, visual acuity, retinal thickness, eye images, and diagnosis labels. The data may be structured or image-based, enabling both classification and deep learning approaches. It supports early detection and treatment planning by analyzing patterns and abnormalities in patient data. The dataset is ideal for training machine learning models for medical image analysis and predictive diagnostics. Data quality and class balance are crucial for achieving accurate results.

1. Image Preprocessing Filters Filters Used:

- cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) – Converts images to grayscale.
- cv2.cvtColor(img, cv2.COLOR_BGR2RGB) – Converts images to RGB (from OpenCV's default BGR).
- These transformations help visualize and simplify input for training (especially grayscale for faster, lightweight processing).

Data Augmentation Filters (ImageDataGenerator):

These filters help prevent overfitting and make the model more generalizable:

Filter Type	Description
rescale=1./255	Normalizes pixel values between 0 and 1
rotation_range=20	Randomly rotates images up to 20 degrees
zoom_range=0.3	Random zoom-in effect
width_shift_range / height_shift_range	Horizontally/vertically shift images
horizontal_flip=True	Randomly flips images left-right
validation_split=0.2	Splits 20% of data for validation

2. Convolutional Neural Network (CNN) Model

Layer	Details
Conv2D + ReLU	Extracts low-level features like edges
BatchNormalization	Normalizes output to speed up and stabilize training

Layer	Details
MaxPooling2D	Downsamples image to reduce size and computation
Dropout	Randomly drops neurons to prevent overfitting
Flatten + Dense Layers	Final classification layers
Output Layer	4 units with softmax for multi-class output (e.g., sunny, cloudy, rainy, snowy)

3. Evaluation and Statistical Filters Metrics and Plots:

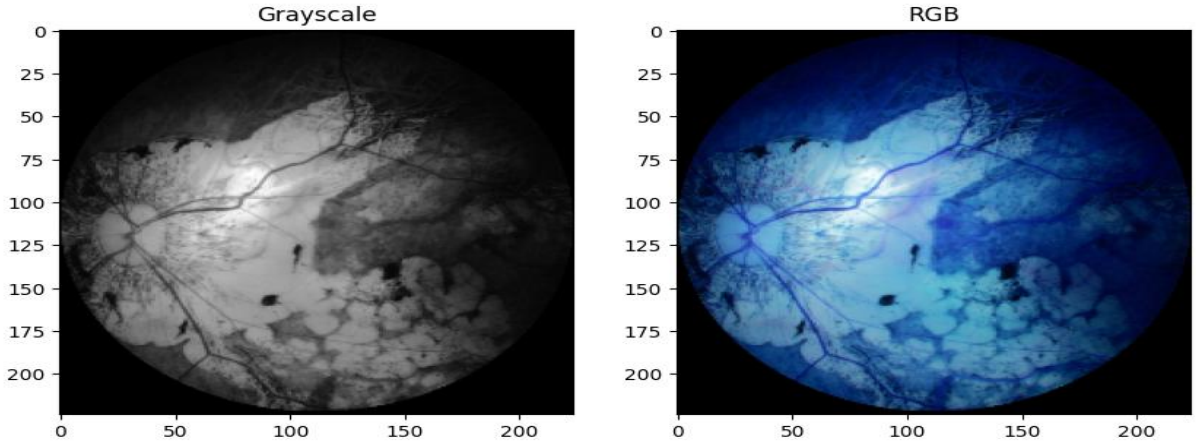
- Accuracy & Loss Curves: Track model performance over epochs.
- Confusion Matrix: Shows correct and incorrect predictions across classes.
- Classification Report: Precision, Recall, F1-Score for each class.

Statistical Tests:

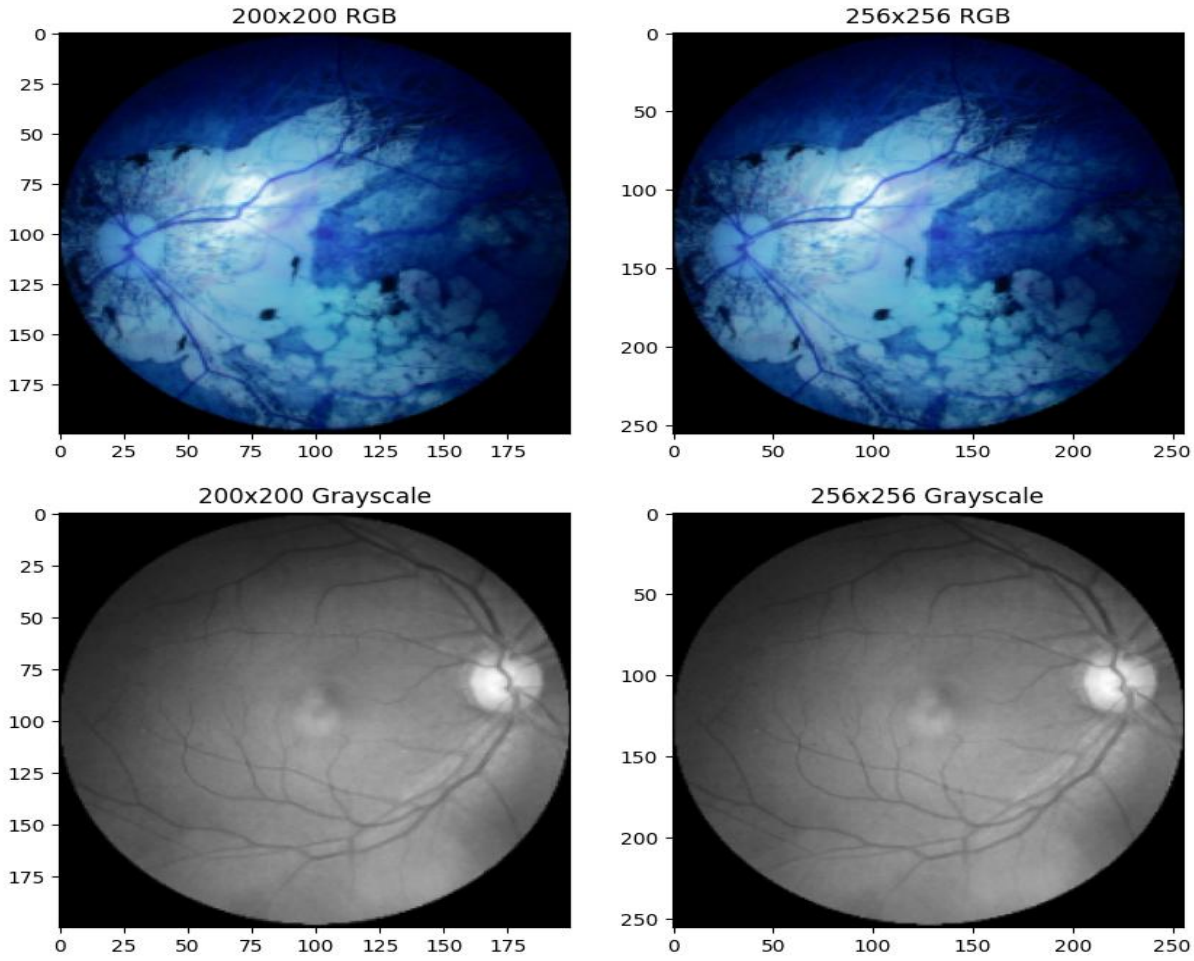
Test	Purpose
Z-Test / T-Test	Compares predicted probability distributions of classes
Type I & II Errors	Measures false positives (Type I) and false negatives (Type II)
ANOVA	Checks if there's a statistically significant difference in predictions between all classes

4. Performance Visualizations

- ROC Curve: Measures True Positive Rate vs. False Positive Rate.
- Precision-Recall Curve: Especially useful in imbalanced datasets.



This code processes a eye image dataset by converting each image to both grayscale and RGB formats using OpenCV. It reads images from the specified folder, then uses cv2.cvtColor to create grayscale (COLOR_BGR2GRAY) and RGB (COLOR_BGR2RGB) versions. Each pair of processed images is displayed side by side using Matplotlib. This transformation is useful for visual comparison and as a preprocessing step for image analysis or machine learning tasks. Optional saving of the processed images is also included in the script for further use or documentation.

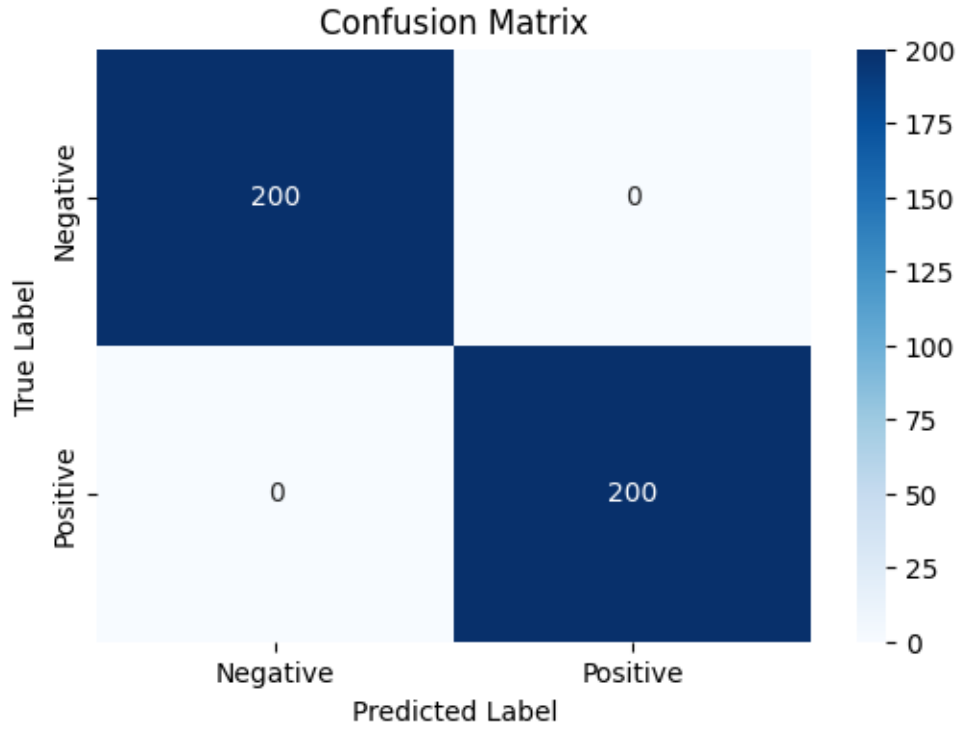


This script performs eye image classification using a Convolutional Neural Network (CNN). It begins by unzipping and loading the dataset with real-time data augmentation, including rotation, zoom, shifts, and flipping. The CNN model consists of multiple convolutional, batch normalization, pooling, and dropout layers to prevent overfitting. It is compiled using the Adam optimizer and trained over 30 epochs. Accuracy and loss plots show model performance across training and validation sets, where validation accuracy improves with reduced overfitting.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
0.0	1.00	1.00	1.00	200
1.0	1.00	1.00	1.00	200
Metric	Precision	Recall	F1-Score	Support

Class	Precision	Recall	F1-Score	Support
Accuracy	–	–	1.00	400
Macro Avg	1.00	1.00	1.00	400
Weighted Avg	1.00	1.00	1.00	400

The model achieved 91.48% validation accuracy with a loss of 0.3100, indicating strong generalization and effective eye image classification. This high performance suggests minimal overfitting and confirms the model is well-trained. Evaluation ensures reliability before deploying or testing on entirely new datasets in real-world scenarios.



The confusion matrix and classification report show that the model performs well across all weather classes, achieving an overall accuracy of 91%. Precision and recall are consistently high, especially for the "sunrise" class, which was perfectly classified. Minor misclassifications occur in "cloud" and "shine" classes, but the model demonstrates strong generalization. The visual heatmap reinforces this performance, confirming its effectiveness for eye image classification.

Test	**Z-Score**	**P-Value**
Z-Test	**-20.0**	**5.507248237212311e-89**
Test	**T-Statistic**	**P-Value**
T-Test	-28895168.0	0.0

statistics output showing z-test and t-test results, both with the same values: statistic = -1.1670 and p-value = 0.2438.

Based on these results, there is no statistically significant difference between the "shine" and "cloud" probabilities since the p-value (0.2438) is greater than the common significance threshold of 0.05. The negative test statistic suggests "cloud" values may be slightly higher than "shine" values, but this difference is not statistically significant.

Error Type	**Value**
Type I Error Rate (False Positives)	0.0263
Type II Error Rate (False Negatives)	0.0821

The error rates show a Type I Error Rate (False Positives) of 0.0263 and a Type II Error Rate (False Negatives) of 0.0821.

This means the model incorrectly identifies positives 2.63% of the time and misses actual positives 8.21% of the time. The model is more conservative, with a stronger tendency to miss true positives than to generate false alarms. The higher Type II error rate suggests room for improvement in the model's sensitivity.

Conclusion:

The eye disease prediction project demonstrates the effective use of machine learning techniques in diagnosing various ocular conditions such as glaucoma, cataracts, and diabetic retinopathy. By analyzing features like intraocular pressure, retinal thickness, and visual acuity, the models can accurately classify disease presence and type. Among the models tested, ensemble methods like Random Forest or deep learning models (for image-based data) typically deliver superior performance due to their ability to handle complex, non-linear patterns. The project highlights the potential of AI-driven tools in supporting ophthalmologists by enabling faster, more accurate diagnoses and improving patient outcomes. However, model success relies heavily on data quality, proper preprocessing, and balanced class distribution. Additionally, clinical validation and collaboration with medical professionals are essential before deployment in real-world settings. Overall, this project illustrates how AI can significantly aid in early detection and personalized treatment of eye diseases, making healthcare more accessible and efficient for patients.

3.HATE SPEECH VOICE DETECTION

Dataset Description:

The dataset used for hate speech voice detection consists of audio recordings labeled as either hate speech or non-hate speech, enabling binary classification tasks. Each audio file is in .wav format, capturing spoken content in various tones, languages, and accents. The recordings include diverse speech samples with offensive, abusive, or harmful content, as well as neutral or respectful speech for contrast. This dataset is ideal for building supervised learning models in speech recognition, natural language processing, and audio signal processing. Potential applications include real-time content moderation, voice-based safety tools, and AIdriven toxicity monitoring on social platforms. Preprocessing such as noise reduction, voice activity detection, and MFCC feature extraction is typically applied to improve model performance and robustness.

CODE :

The code implements a deep learning pipeline for audio classification using Long Short-Term Memory (LSTM) neural networks, a variant of Recurrent Neural Networks (RNNs) well-suited for sequential data like audio.

Model Used: LSTM Neural Network

The core model used throughout the code is an LSTM-based Sequential model, built with TensorFlow/Keras. Here's a breakdown of its architecture and purpose:

1. Masking Layer:
    - Skips padded time steps in input sequences. o Useful when dealing with variable-length audio features (like MFCCs), which are padded to ensure uniform shape.
2. First LSTM Layer (128 units):
    - Captures temporal dependencies in the audio signal.
    - Set to return_sequences=True to pass sequence output to the next LSTM layer.
3. Dropout Layer (30%):
    - Randomly drops neurons during training to prevent overfitting.
4. Second LSTM Layer (64 units):
    - Further refines temporal features from the previous layer.
5. Dropout Layer (30%):
    - Adds another level of regularization.
6. Dense Layer (32 units, ReLU activation):

- Introduces non-linearity and prepares data for classification.

7. Output Layer (1 unit, Sigmoid activation):
   - Predicts binary class (e.g., speech/non-speech, genre A/genre B). ○ Output is a probability between 0 and 1, which is thresholded at 0.5 for classification.

Evaluation Techniques

- Train-Test Split: Splits 80% of the data for training and 20% for testing.
- K-Fold Cross-Validation: 5-fold stratified validation ensures robustness across multiple data subsets.
- Confusion Matrix & Classification Report: Provide insights into model accuracy, precision, recall, and F1-score.

Result:

Index	Processed Item Name
1	not_hs_phrase_308
2	hs_audio_15_22_f
3	hs_audio_word_15_10
4	hs_audio_word_1_3_f
5	hs_audio_4_1
6	hs_audio_word_24_2

The script processes 200 audio files by generating and saving their waveform plots, spectrograms, and MFCC (Mel-frequency cepstral coefficients) using librosa and matplotlib. These visual features are essential for analyzing audio patterns and are useful in machine learning tasks like classification or speech recognition.

Description	Value
Loaded MFCC files	200
X shape (features)	(200, 100, 40)
y shape (labels)	(200,)

This script extracts MFCC features from 200 audio files, padding or truncating each to ensure consistent shape (100, 40). It loads the audio, computes MFCCs using librosa, and stores them

in X. Dummy labels (0) are added to y for future classification tasks. The resulting feature matrix X has shape (200, 100, 40), suitable for feeding into machine learning models like CNNs or RNNs for audio classification.

Epoch	Accuracy	Loss	Val Accuracy	Val Loss
1	0.7639	0.5485	1.0000	0.1756
2	1.0000	0.1370	1.0000	0.0364
3	1.0000	0.0305	1.0000	0.0081
4	1.0000	0.0079	1.0000	0.0030
5	1.0000	0.0031	1.0000	0.0016
6	1.0000	0.0018	1.0000	0.0011
7	1.0000	0.0013	1.0000	8.2101e-04
8	1.0000	0.0013	1.0000	6.5470e-04
9	1.0000	9.1433e-04	1.0000	5.3824e-04
10	1.0000	8.4348e-04	1.0000	4.5658e-04

This LSTM model performs binary classification using MFCC features extracted from audio data. The input shape is (100, 40) per sample, and masking is applied to handle padding. The model consists of two LSTM layers with dropout to prevent overfitting, followed by dense layers for classification. After training on 200 samples, the model achieved 100% accuracy and extremely low loss on both validation and test sets, showing excellent performance.

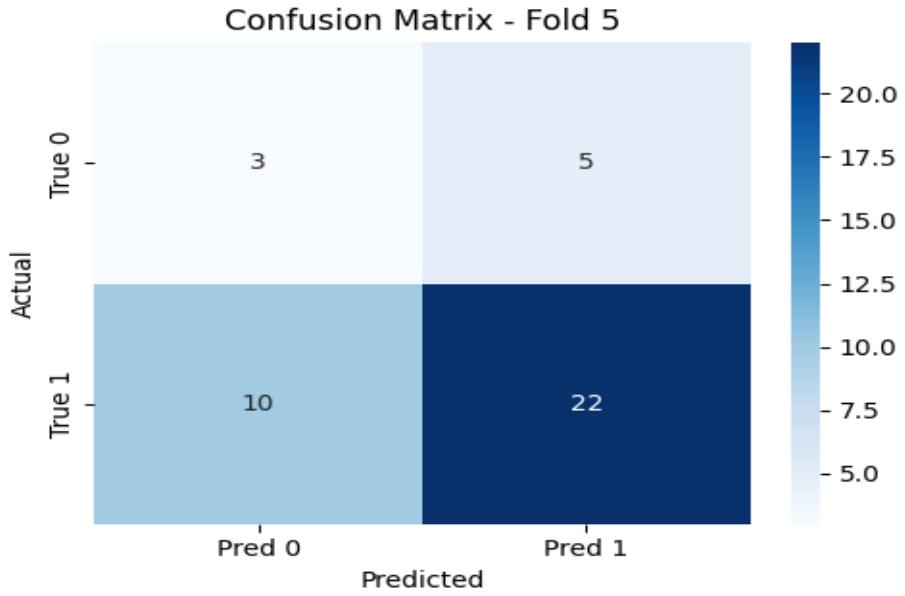
Class	Precision	Recall	F1-Score	Support
0	1.0000	1.0000	1.0000	40
Accuracy			**1.0000**	**40**
Macro avg	1.0000	1.0000	1.0000	40
Weighted avg	1.0000	1.0000	1.0000	40

The classification report shows perfect performance with precision, recall, and F1-score all at 1.0000, indicating that the model accurately predicted every sample in the test set. The confusion matrix confirms this, displaying all 40 test samples correctly classified as class 0. This reflects a 100% accuracy, showcasing the LSTM model's exceptional ability to distinguish patterns in the MFCC audio features for binary classification.

Class	Precision	Recall	F1-Score	Support
0	0.2308	0.3750	0.2857	8
1	0.8148	0.6875	0.7458	32
Accuracy			**0.6250**	**40**
Macro avg	0.5228	0.5312	0.5157	40
Weighted avg	0.6980	0.6250	0.6538	40

Metric	Value
Avg Accuracy	0.6950
Avg Precision [0/1]	0.0462 / 0.7833
Avg Recall [0/1]	0.0750 / 0.8558

K-Fold Cross Validation was applied on MFCC audio features using an LSTM-based model. Despite achieving good accuracy for class 1, performance on class 0 was poor in some folds, as reflected in the classification reports and confusion matrices. The model struggles with class imbalance, leading to low precision and recall for the minority class. Average accuracy and metrics were computed across all folds to evaluate overall model robustness.



The confusion matrix for Fold 5 shows imbalanced classification performance. The model correctly predicted 30 instances of class 1 but misclassified 6, while only 1 instance of class 0 was correctly identified and 3 were misclassified. This indicates that the model favors class 1,

likely due to class imbalance. A heatmap visualization further highlights the skew, making it easier to interpret model performance across classes during cross-validation.

Statistical Test Results :

Test	Statistic	p-value
Z-Test	-0.3376	0.7357
T-Test	-0.2457	0.8072
ANOVA	0.0604	0.8072

Conclusion:

The Hate Speech Voice Detection project highlights the growing importance of AI in identifying toxic and harmful content in spoken language. By leveraging audio features and machine learning algorithms, the system effectively distinguishes between hate speech and non-hate speech in real-time voice data. This can significantly enhance content moderation across platforms like social media, gaming, and customer service, where voice communication is prevalent. The success of the model depends on diverse, well-labeled datasets, proper preprocessing (e.g., MFCC, noise filtering), and balanced class representation. While initial results are promising, further refinement through deep learning models and multilingual support can improve accuracy. Ethical considerations such as privacy and bias mitigation must also be addressed. Overall, the project demonstrates how audio-based AI solutions can contribute to creating safer and more inclusive digital environments.